EMNLP 2018

2018 Conference on Empirical Methods in Natural Language Processing

Proceedings of the Conference

Oct. 31-Nov. 4
Brussels, Belgium

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

Diamond Tier

Google

facebook

Bloomberg®

salesforce

Platinum Tier

amazon

Bai du 百度

grammarly

京东

NAVER LABS EUROPE

visit.brussels

fwo Research Foundation Flanders
Opening new horizons

KU LEUVEN
Department of Computer Science

Gold Tier

CVTE
视源股份

ebay™

Microsoft

NAVER | LINE

ORACLE®

POLYAI

Silver Tier

HUAWEI

duolingo

figure eight

NUANCE

Bronze Tier & Supporter Tier

NEXT CANADA

AI2 ALLEN INSTITUTE for ARTIFICIAL INTELLIGENCE

textkernel
Machine Intelligence for People and Jobs

text IQ

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-948087-84-1

Preface by the General Chair

Welcome to EMNLP 2018 in Brussels, Belgium! I hope that EMNLP 2018 will be a memorable experience filled with exciting research presentations, outstanding keynote speakers, and many stimulating conversations with colleagues and friends, both old and new. In the evenings, I hope that you will explore the wonderful city of Brussels! Admire the historic buildings, visit great museums, and enjoy the Belgian cuisine.

EMNLP 2018 will have an extensive technical program that includes 14 workshops, 6 tutorials, 3 invited speakers, 351 long paper presentations, 198 short paper presentations, 10 TACL paper presentations, and 29 demos. I want to give special thanks to the Program Co-Chairs: David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii. EMNLP 2018 received a record-breaking 2,231 valid submissions, a 48% increase over EMNLP 2017! Despite the massive volume of submissions, the PC chairs put tremendous care into every decision, big and small, and gracefully handled numerous inquiries and requests. Their commitment to a high-quality program was inspiring. Of course, the PC Chairs did not handle the workload alone: we all owe an enormous debt of gratitude to the 60 Area Chairs and 1,436 reviewers (yes, 1,400+ reviewers!) who took on the critical responsibilities of reviewing paper submissions and providing feedback to the program chairs. Thank you all!

The Demo Co-Chairs, Eduardo Blanco and Wei Lu, also had to handle an usually high workload. EMNLP 2018 received 77 demo submissions, which was a 40% increase over EMNLP 2017. They recruited an additional 150+ reviewers to assess the demo submissions, and managed the review process seamlessly. Thank you, Eduardo and Wei!

The EMNLP 2018 Workshop Co-Chairs, Marieke van Erp and Vincent Ng, and the EMNLP 2018 Tutorial Co-Chairs, Mausam and Lu Wang, oversaw the selection of workshops and tutorials and coordinated planning with the organizers. Many thanks to Marieke, Vincent, Mausam, and Lu for orchestrating terrific workshop and tutorial programs! And I am thrilled to have three exceptional NLP researchers as our keynote speakers: Johan Bos, Julia Hirschberg, and Gideon Mann.

For EMNLP 2018, we added two new initiatives to provide financial support to conference participants. We introduced *childcare grants* that offer financial support for childcare expenses incurred to participate in the conference. These childcare grants give families broad flexibility to utilize many different types of childcare services and arrangements. A giant thank you to Swapna Somasundaran for navigating this new terrain and drafting the policy guidelines, and to both Swapna and Aoife Cahill for overseeing the application and decision processes.

EMNLP 2018 also introduced travel scholarships for non-students to attend the conference. The non-student travel scholarships provide support to people with financial need who might otherwise not be able to participate in the conference. We created this initiative with an eye toward supporting researchers from historically under-representated regions and young researchers or others with limited financial means. I want to thank Anna Rumshisky and Hugo Van hamme for managing these new non-student travel scholarships, along with the student travel scholarship and volunteer applications.

Nitin Madnani deserves special thanks for taking charge of *both* the EMNLP 2018 web site and conference app, and doing a fantastic job. Nitin was a pleasure to work with, and incredibly responsive, answering emails impossibly fast and accommodating every request. Thanks also to our Publicity Chair, Mohit Iyyer, for advertising the conference on social media and elsewhere, and our Video Chair, Bonan Min, for handling the video recordings.

Managing the conference proceedings and compiling the handbook are extremely time-consuming and tedious jobs, but less visible to conference attendees, who only see the final products. Enormous thanks to the Publication Chairs, Micha Elsner (junior chair) and Preethi Raghavan (senior chair), the Conference Handbook Chair, Kai-Wei Chang, the Handbook Advisor, Joachim Bingel, and the Handbook Proofreader, He He. They were truly a fantastic team!

And a huge thanks to the Local Organization Committee from KU Leuven: Marie-Francine Moens (chair), Dominique De Brabanter, Frieda Steurs, and Hugo Van hamme, as well as the Local Sponsorship Chair, Katrien Beuls from Vrije Universiteit Brussel. Their hard work and enthusiasm were key to making EMNLP 2018 a success! In addition, a large conference like EMNLP depends heavily on sponsorship, and I want to sincerely thank *all* of the EMNLP 2018 sponsors for their generous support!

I must also thank SIGDAT for their support throughout this past year. With extra thanks to Noah Smith, the SIGDAT Secretary-Treasurer and EMNLP liaison, for his support of our new initiatives, valuable feedback, and quick responses. Finally, I owe a huge debt of gratitude to Priscilla Rasmussen, who managed so many aspects of the conference that I can't even begin to name them. A heartfelt thanks to Priscilla for her hard work, sage advice, and for patiently answering an endless stream of questions from me, and many others.

And thanks to all of YOU for participating in EMNLP 2018! I hope you enjoy the conference and your time in Brussels!

EMNLP 2018 General Chair

Ellen Riloff, University of Utah, USA

Preface by the Program Committee Co-Chairs

Welcome to EMNLP 2018! This year's technical program consists of three invited talks, 224 oral presentations (including three papers appearing in the Transactions of the ACL), and 335 poster presentations (including seven papers appearing in TACL). To our knowledge, it's the largest NLP conference ever, and it would not have been possible without the help of our program committee members.

This year, we organized the program committee into eight broad areas. For each area, one of our amazing senior area chairs (Jordan Boyd-Graber, Xavier Carreras, Yejin Choi, Philipp Koehn, Alessandro Moschitti, Slav Petrov, Massimo Poesio, and Kam-Fai Wong) headed a team of several other area chairs (totaling 52 across all areas) and an army of reviewers (totaling 1,436 across all areas). We'd like to especially thank those reviewers who agreed to review a full load of papers, which turned out to be a bit heavier than expected!

2018 appears to have been the year of experimentation with new review forms, and EMNLP was no exception. We eliminated many of the traditional numerical ratings and divided up the free-form comments into a small number of separate free-response questions. We tried to make this new structure mirror the structure of a typical review, while encouraging more comprehensive reviewing.

Extrapolating exponentially from the past two years, we expected to receive about 1,800 submissions, but the eventual number exceeded our expectations at 2,231 (excluding empty and duplicate submissions). After removal of invalid submissions and some withdrawals, we sent 2,116 papers out for review. Despite the growing number of submissions, we tried to keep acceptance rates at the same level as past years. The acceptance statistics are shown below.

	Long	Short	Total	TACL
Submitted	1,376	855	2,231	–
Accepted as talk	140 (10.2%)	81 (9.5%)	221 (9.9%)	3
Accepted as poster	211 (15.3%)	117 (13.7%)	328 (14.7%)	7
Accepted (total)	351 (25.5%)	198 (23.2%)	549 (24.6%)	10

As in past years, three awards will be given for Best Long Paper, Best Short Paper, and Best Resource Paper. We solicited recommendations for awards from reviewers and area chairs. Following these recommendations, we sent 7 long papers, 5 short papers, and 5 resource papers to three committees chosen from among the area chairs. In the end, we selected two winners for Best Long Paper and one each for Best Short Paper and Best Resource Paper, all of which will be presented in a final plenary session.

We are delighted to have keynote addresses from three giants of our field: Johan Bos, on the future of computational semantics; Julia Hirschberg, on deception detection in speech; and Gideon Mann, on the use of NLP in finance.

In addition to our entire program committee, we would also like to give thanks to:

- Our general chair, Ellen Riloff,
- Priscilla Rasmussen,
- Last year's program chairs, Rebecca Hwa and Sebastian Riedel,
- Our local chair, Marie-Francine Moens, and the local organizing committee,
- Our web and publicity chairs, Nitin Madnani and Mohit Iyyer,
- Our publications chairs, Micha Elsner and Preethi Raghavan,
- Our handbook chair, Kai-Wei Chang,
- Rich Gerber and the technical support team at SoftConf.

Again, welcome! We hope that you enjoy this year's conference, and return home with new insights, ideas, and opportunities!

EMNLP 2018 Program Co-Chairs

David Chiang, University of Notre Dame, USA
Julia Hockenmaier, University of Illinois Urbana-Champaign, USA
Jun'ichi Tsujii, Artificial Intelligence Research Center, Japan

Organizing Committee

General Chair
Ellen Riloff, University of Utah, USA

Program Committee Co-chairs
David Chiang, University of Notre Dame, USA
Julia Hockenmaier, University of Illinois Urbana-Champaign, USA
Junichi Tsujii, Artificial Intelligence Research Center, Japan

Local Arrangements Committee
Marie-Francine Moens, KU Leuven, Belgium (chair)
Dominique De Brabanter, KU Leuven, Belgium
Frieda Steurs, KU Leuven, Belgium
Hugo Van hamme, KU Leuven, Belgium

Local Sponsorship Chair
Katrien Beuls, Vrije Universiteit Brussel, Belgium

Workshop Co-chairs
Marieke van Erp, KNAW Humanities Cluster, Netherlands
Vincent Ng, University of Texas at Dallas, USA

Tutorial Co-chairs
Mausam, Indian Institute of Technology Delhi, India
Lu Wang, Northeastern University, USA

Demos Co-chairs
Eduardo Blanco, University of North Texas, USA
Wei Lu, Singapore University of Technology and Design, Singapore

Publications Sr Chair
Preethi Raghavan, IBM, USA

Publications Jr Chair
Micha Elsner, Ohio State University, USA

Publicity Chair
Mohit Iyyer, University of Massachusetts at Amherst, USA

Video Chair
Bonan Min, BBN Technologies, USA

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

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

Conference Handbook Advisor
Joachim Bingel, University of Copenhagen, Denmark

Handbook Proofreader
He He, Stanford University, USA

Student Scholarship Co-chair and Student Volunteer Coordinator
Anna Rumshisky, University of Massachusetts at Lowell, USA
Hugo Van hamme, KU Leuven, Belgium

Childcare Policy and Grant Coordinator
Swapna Somasundaran, Educational Testing Service, USA
Aoife Cahill, Educational Testing Service, USA

Program Committee

Program Committee Co-chairs
David Chiang, University of Notre Dame, USA
Julia Hockenmaier, University of Illinois Urbana-Champaign, USA
Junichi Tsujii, Artificial Intelligence Research Center, Japan

Area Chairs
Information Extraction, Information Retrieval, and Question Answering
Jordan Boyd-Graber, University of Maryland, USA (senior chair)
Isabelle Augenstein, University of Copenhagen, Denmark
Ming-Wei Chang, Google, USA
Doug Downey, Northwestern University, USA
Ruihong Huang, Texas A&M University, USA
Mausam, IIT Delhi, India
Makoto Miwa, Toyota Technological Institute, Japan
William Wang, University of California at Santa Barbara, USA
Scott Yih, Allen Institute for Artificial Intelligence, USA

Text Mining and Information Retrieval
Alessandro Moschitti, University of Trento, Italy (senior chair)
Sophia Ananiadou, University of Manchester, United Kingdom
Hsin-Hsi Chen, National Taiwan University
Marius Pasca, Google, USA
Xiang Ren, University of Southern California, USA
Alan Ritter, Ohio State University, USA
David Smith, Northeastern University, USA

Social Media, Computational Social Science, and Sentiment/Opinion Analysis
Kam-Fai Wong, Chinese University of Hong Kong (senior chair)
Eiji Aramaki, Nara Institute of Science and Technology, Japan
Mona Diab, George Washington University, USA
Yulan He, Aston University, United Kingdom
Dirk Hovy, Bocconi University, Italy
Rada Mihalcea, University of Michigan, USA
Alice Oh, KAIST, Korea
Wei Xu, Ohio State University, USA

Morphology, Syntax, and Psycholinguistics
Slav Petrov, Google, USA (senior chair)
Liang Huang, Oregon State University, USA
Roger Levy, Massachusetts Institute of Technology, USA
Stephan Oepen, University of Oslo, Norway
Emily Pitler, Google, USA
Reut Tsarfaty, Open University of Isra

Semantics

Massimo Poesio, Queen Mary University of London, United Kingdom (senior chair)
Omri Abend, Hebrew University of Jerusalem, Israel
Raffaella Bernardi, University of Trento, Italy
Michael Lewis, Facebook, USA
Yulia Tsvetkov, Carnegie Mellon University, USA
Benjamin Van Durme, Johns Hopkins University, USA
Nianwen Xue, Brandeis University, USA

Discourse, Dialogue, Summarization, Generation, and Multimodal NLP

Yejin Choi, University of Washington, USA (senior chair)
Mohit Bansal, University of North Carolina, USA
Michel Galley, Microsoft, USA
Grzegorz Chrupała, Tilburg University, Netherlands
Haizhou Li, National University of Singapore
Fei Liu, University of Central Florida, USA
Karen Livescu, Toyota Technological Institute - Chicago, USA
Meg Mitchell, Google, USA
Rashmi Prasad, Interactions, USA
Xiaojun Wan, Peking University, China
Zhou Yu, University of California at Davis, USA

Machine Translation and Multilinguality

Philipp Koehn, Johns Hopkins University, USA (senior chair)
Arianna Bisazza, Leiden University, Netherlands
Qun Liu, Dublin City University, Ireland
Yang Liu, Tsinghua University, China
Zhaopeng Tu, Tencent, China
Taro Watanabe, Google, Japan

Machine Learning

Xavier Carreras, dMetrics, USA (senior chair)
Shay Cohen, University of Edinburgh, United Kingdom
Kevin Gimpel, Toyota Technological Institute - Chicago, USA
Stefan Riezler, Heidelberg University, Germany
Karl Stratos, Toyota Technological Institute - Chicago, USA
Jun Suzuki, Tohoku University, Japan

Reviewers

We would like to recognize the following 100 reviewers with the Best Reviewer Award.

Natalie Ahn, Emilia Apostolova, Miguel Ballesteros, Daniel Bauer, Yonatan Belinkov, Darina Benikova, Luciana Benotti, Robert Berwick, Chloé Braud, Hendrik Buschmeier, Jan Buys, Dallas Card, Asli Celikyilmaz, Arun Chaganty, Kai-Wei Chang, Colin Cherry, Maximin Coavoux, Zeyu Dai, Cedric De Boom, Miryam de Lhoneux, Mark Dredze, Greg Durrett, Ondrej Dusek, Jason Eisner, Michael Elhadad, M. Amin Farajian, Shi Feng, Simone Filice, Stefan L. Frank, Lea Frermann, Matthias Gallé, Ekaterina

Garmash, Michaela Geierhos, Lieke Gelderloos, Kallirroi Georgila, Alborz Geramifard, Matthew R. Gormley, Yvette Graham, Jiatao Gu, Bo Han, Shuguang Han, Abram Handler, Hua He, Lisa Anne Hendricks, Daniel Hershcovich, Jack Hessel, Gerold Hintz, Ari Holtzman, Peter Jansen, Ȧkos Kádár, Sarvnaz Karimi, Ekaterina Kochmar, Parisa Kordjamshidi, Gaurav Kumar, Adhiguna Kuncoro, Yanran Li, Jing Li, Chen Li, Lucelene Lopes, Adam Lopez, Kate Loveys, Chandler May, Nafise Sadat Moosavi, Andrea Moro, Philippe Muller, Shashi Narayan, Dong Nguyen, Hitoshi Nishikawa, Peyman Passban, Laura Perez-Beltrachini, Ana-Maria Popescu, Piotr Przybyla, Peng Qi, Will Radford, Roi Reichart, Anna Rohrbach, Shigehiko Schamoni, Yves Scherrer, Sebastian Schuster, Matthew Shardlow, Tianze Shi, Carina Silberer, Kevin Small, Luca Soldaini, Gabriel Stanovsky, Kristina Striegnitz, Jannik Strötgen, Liling Tan, Hao Tang, Yi Tay, Mariët Theune, Yuen-Hsien Tseng, Subhashini Venugopalan, Henning Wachsmuth, Eric Wallace, Bonnie Webber, Michael White, Shuly Wintner, Andrew Yates, Justine Zhang.

We also thank the remaining reviewers for their hard work.

Mourad Abbas, Muhammad Abdul-Mageed, Amjad Abu-Jbara, Oliver Adams, Heike Adel, Stergos Afantenos, Apoorv Agarwal, Željko Agić, Roee Aharoni, Chaitanya Ahuja, Zeynep Akata, Alan Akbik, Ahmet Aker, Cem Akkaya, Chris Alberti, Hanan Aldarmaki, Nikolaos Aletras, Afra Alishahi, Alexandre Allauzen, Tim Althoff, Bharat Ram Ambati, Waleed Ammar, Antonios Anastasopoulos, Jesse Anderton, Nicholas Andrews, Anietie Andy, Gabor Angeli, Marianna Apidianaki, Jun Araki, Kenji Araki, Yuki Arase, Ehsaneddin Asgari, Ramón Astudillo, Giuseppe Attardi, Eleftherios Avramidis, Amittai Axelrod, Wilker Aziz, Yoram Bachrach, Hessam Bagherinezhad, Fan Bai, Simon Baker, Mithun Balakrishna, Niranjan Balasubramanian, Tyler Baldwin, David Bamman, Rafael E. Banchs, Siddhartha Banerjee, Trapit Bansal, Roy Bar-Haim, Libby Barak, Alistair Baron, Marco Baroni, Alberto Barrón-Cedeño, Guntis Barzdins, Pierpaolo Basile, Valerio Basile, Joost Bastings, Riza Theresa Batista-Navarro, Vishwash Batra, Timo Baumann, Rachel Bawden, Frederic Bechet, Daniel Beck, Srikanta Bedathur, Núria Bel, Eric Bell, Kedar Bellare, Iz Beltagy, Anja Belz, Farah Benamara, Jonathan Berant, Taylor Berg-Kirkpatrick, Raffaella Bernardi, Nicola Bertoldi, Laurent Besacier, Steven Bethard, Chandra Bhagavatula, Suma Bhat, Archna Bhatia, Sumit Bhatia, Pushpak Bhattacharyya, Chris Biemann, Lidong Bing, Alexandra Birch, Yonatan Bisk, Johannes Bjerva, Jari Björne, Frédéric Blain, Eduardo Blanco, Su Lin Blodgett, Michael Bloodgood, Reihane Boghrati, Nikolay Bogoychev, Bernd Bohnet, Danushka Bollegala, Daniele Bonadiman, Francesca Bonin, Kalina Bontcheva, Georgeta Bordea, Alexey Borisov, Johan Bos, Antoine Bosselut, Jan A. Botha, Houda Bouamor, Fethi Bougares, Samuel Bowman, Ryan Boyd, Matko Bošnjak, Jonathan Brennan, Chris Brew, Chris Brockett, Austin Brockmeier, Thomas Brovelli (Meyer), Caroline Brun, Dominique Brunato, Paul Buitelaar, Florin Bulgarov, Stephan Busemann, Bill Byrne, Donna Byron, Benjamin Börschinger, José G. C. de Souza, Elena Cabrio, Aoife Cahill, Iacer Calixto, Jose Camacho-Collados, Erik Cambria, Nicola Cancedda, Marie Candito, Hailong Cao, Yuan Cao, Ziqiang Cao, Giuseppe Carenini, Francisco Casacuberta, Thiago Castro Ferreira, Daniel Cer, Mauro Cettolo, Soumen Chakrabarti, Yllias Chali, Yee Seng Chan, Muthu Kumar Chandrasekaran, Baobao Chang, Chia-Hui Chang, Yin-Wen Chang, Rajen Chatterjee, Snigdha Chaturvedi, Stergios Chatzikyriakidis, Wanxiang Che, Ciprian Chelba, Bin Chen, Boxing Chen, Chen Chen, Chung-Chi Chen, Danqi Chen, Huan-Yuan Chen, John Chen, Kaiping Chen, Kehai Chen, Muhao Chen, Tao Chen, Tongfei Chen, Wenhu Chen, Wenliang Chen, Xilun Chen, Yidong Chen, Yubo Chen, Yun-Nung Chen, Zhiyuan Chen, Jianpeng Cheng, Pu-Jen Cheng, Yong Cheng, Jackie Chi Kit Cheung, Jen-Tzung Chien, Hai Leong Chieu, Laura Chiticariu, Kyunghyun Cho, Eunsol Choi, Heeyoul

Choi, Leshem Choshen, Prafulla Kumar Choubey, Christos Christodoulopoulos, Chenhui Chu, Tagyoung Chung, Kenneth Church, Philipp Cimiano, Volkan Cirik, Alexander Clark, Kevin Clark, Ann Clifton, Arman Cohan, Daniel Cohen, Raphael Cohen, Nigel Collier, Michael Collins, John Conroy, Matthieu Constant, Danish Contractor, Bonaventura Coppola, Gregory F. Coppola, Caio Corro, Marta R. Costa-jussà, Ryan Cotterell, Benoit Crabbé, Danilo Croce, James Cross, Montse Cuadros, Heriberto Cuayahuitl, Yiming Cui, Aron Culotta, Anna Currey, Iria da Cunha, Özlem Çetinoğlu, Ido Dagan, Andrew Dai, Zihang Dai, Joachim Daiber, Jeff Dalton, Bhavana Dalvi, Falavigna Daniele, Lena Dankin, Kareem Darwish, Amitava Das, Dipanjan Das, Rajarshi Das, Pradeep Dasigi, Gaël de Chalendar, Adrià de Gispert, Daniël de Kok, Eric De La Clergerie, Renato de Mori, Luciano Del Corro, Claudio Delli Bovi, Louise Deléger, Thomas Demeester, David Demeter, Dina Demner-Fushman, Steve DeNeefe, Lingjia Deng, Pascal Denis, Michael Denkowski, Matthew Denny, Tejaswini Deoskar, Valeria de Paiva, Nina Dethlefs, Chris Develder, Jacob Devlin, Bhuwan Dhingra, Luigi Di Caro, Giuseppe Di Fabbrizio, Jana Diesner, Shuoyang Ding, Simon Dobnik, Ellen Dodge, Jesse Dodge, Tobias Domhan, Li Dong, MeiXing Dong, Shichao Dong, A. Seza Doğruöz, Eduard Dragut, Mark Dras, Markus Dreyer, Lan Du, Xiangyu Duan, Haim Dubossarsky, Pablo Duboue, Kevin Duh, Nadir Durrani, Chris Dyer, Marc Dymetman, Valery Dzutsati, Hiroshi Echizen'ya, Richard Eckart de Castilho, Thomas Effland, Steffen Eger, Markus Egg, Patrick Ehlen, Maud Ehrmann, Vladimir Eidelman, Andreas Eisele, Jacob Eisenstein, Asif Ekbal, Layla El Asri, Heba Elfardy, Ahmed Elgohary, Mihail Eric, Hugo Jair Escalante, Cristina España-Bonet, Kurt Junshean Espinosa, Luis Espinosa Anke, Allyson Ettinger, Keelan Evanini, James Fan, Hao Fang, Hui Fang, Licheng Fang, Manaal Faruqui, Benoit Favre, Maryam Fazel-Zarandi, Marcello Federico, Christian Federmann, Geli Fei, Anna Feldman, Yang Feng, Yansong Feng, Raquel Fernández, Daniel Fernández-González, Francis Ferraro, Elena Filatova, Katja Filippova, Mark Finlayson, Orhan Firat, Mark Fishel, Nicholas FitzGerald, Jeffrey Flanigan, Dan Flickinger, Michael Flor, Radu Florian, Antske Fokkens, José A. R. Fonollosa, Maxwell Forbes, Tommaso Fornaciari, George Foster, Samuel Fraiberger, Anette Frank, Dayne Freitag, Markus Freitag, André Freitas, Jesse Freitas, Daniel Fried, Xiao Fu, Hagen Fuerstenau, Atsushi Fujii, Fumiyo Fukumoto, Richard Futrell, Alona Fyshe, Michael Gamon, Zhe Gan, Kuzman Ganchev, Octavian-Eugen Ganea, Juri Ganitkevitch, Mercedes García-Martínez, Claire Gardent, Dan Garrette, Milica Gasic, Albert Gatt, Tao Ge, Ulrich Germann, Mehdi Ghanimifard, Debanjan Ghosh, Sucheta Ghosh, George Giannakopoulos, Daniel Gildea, Dimitra Gkatzia, Goran Glavaš, Alfio Gliozzo, Koldo Gojenola, Yoav Goldberg, Dan Goldwasser, Sharon Goldwater, Juan Carlos Gomez, Graciela Gonzalez-Hernandez, Jesús González-Rubio, Hugo Gonçalo Oliveira, Jonathan Gordon, Kyle Gorman, Cyril Goutte, Kartik Goyal, Pawan Goyal, Natalia Grabar, Roger Granada, Mark Granroth-Wilding, Eleni Gregoromichelaki, Scott Grimm, Ralph Grishman, Cyril Grouin, Adam Grycner, Anupam Guha, Lin Gui, Tao Gui, Camille Guinaudeau, Hongyu Guo, Jiafeng Guo, Jiang Guo, Weiwei Guo, Nitish Gupta, Sonal Gupta, Francisco Guzmán, Jeremy Gwinnup, Carlos Gómez-Rodríguez, Nizar Habash, Maryam Habibi, Gholamreza Haffari, Masato Hagiwara, Udo Hahn, Jan Hajic, Hannaneh Hajishirzi, Dilek Hakkani-Tur, John Hale, Keith Hall, William L. Hamilton, Thierry Hamon, Jialong Han, Na-Rae Han, Oul Han, Xianpei Han, Sanda Harabagiu, Christian Hardmeier, Daniel Hardt, Orin Hargraves, David Harwath, Kazi Saidul Hasan, Sadid A. Hasan, Mark Hasegawa-Johnson, Homa Hashemi, Kazuma Hashimoto, Eva Hasler, Katsuhiko Hayashi, Devamanyu Hazarika, He He, Luheng He, Shizhu He, Xiangnan He, Yifan He, Zhongjun He, Kenneth Heafield, Carmen Heger, Michael Heilman, Benjamin Heinzerling, James Henderson, John Henderson, Matthew Henderson, Aron Henriksson, Aurélie Herbelot, Ulf Hermjakob, Raquel Hervas, Jonathan Herzig, John Hewitt, Ryuichiro Higashinaka, Derrick Higgins, Tsutomu Hirao, Graeme Hirst, Julian Hitschler, Hieu Hoang, Vu Cong Duy Hoang, Nathan Hodas, Chris Hokamp, Kristy Hollingshead, Chester Holtz, Yu Hong, Joseph Hoover, Ales Horak, Mohammad Javad Hosseini, Yufang Hou, Eduard Hovy, Estevam Hruschka, Baotian Hu, Yuheng Hu, Zhiting Hu, Bo Huang, Guoping Huang, Haoran Huang, Hen-Hsen Huang, Lifu Huang, Minlie Huang, Po-Sen Huang, Ruihong Huang, Shujian Huang, Xiaojiang Huang, Xuanjing Huang, Zhongqiang Huang, Matthias Huck, Samar Husain, Rebecca Hwa, Jena D. Hwang, Seung-won Hwang, Ignacio Iacobacci, Adrian Iftene, Gonzalo Iglesias, Ryu Iida, Loring Ingraham, Naoya Inoue, Kentaro Inui, Takashi Inui, Radu Tudor Ionescu, Molly Ireland, Aminul Islam, Alexei V. Ivanov, Srinivasan Iyer, Mohit Iyyer, Kokil Jaidka, Prachi Jain, Adam Jatowt, Sujay Kumar Jauhar, Sébastien Jean, Laura Jehl, Yacine Jernite, Rahul Jha, Heng Ji, Yangfeng Ji, Robin Jia, Ping Jian, Hui Jiang, Jing Jiang, Jingtian Jiang, Meng Jiang, Wenbin Jiang, Antonio Jimeno Yepes, Yohan Jo, Anders Johannsen, Richard Johansson, Kristen Johnson, Kristiina Jokinen, Gareth Jones, Kenneth Joseph, Mandar Joshi, Shafiq Joty, Meizhi Ju, Marcin Junczys-Dowmunt, Preethi Jyothi, Gerhard Jäger, Nobuhiro Kaji, Herman Kamper, Pallika Kanani, Hiroshi Kanayama, Dongyeop Kang, Dimitri Kartsaklis, Arzoo Katiyar, Makoto P. Kato, Daisuke Kawahara, Hideto Kazawa, Chris Kedzie, Aniruddha Kembhavi, Casey Kennington, Mitesh M. Khapra, Huda Khayrallah, Douwe Kiela, Yuta Kikuchi, Halil Kilicoglu, Jin-Dong Kim, Jooyeon Kim, Najoung Kim, Seokhwan Kim, Suin Kim, Sun Kim, Sunghwan Mac Kim, Yoon Kim, Young-Bum Kim, Irwin King, Eliyahu Kiperwasser, Svetlana Kiritchenko, Nikita Kitaev, Judith Klavans, Alexandre Klementiev, Roman Klinger, Alistair Knott, Rebecca Knowles, Sosuke Kobayashi, Thomas Kober, Simon Kocbek, Kazunori Komatani, Rik Koncel-Kedziorski, Grzegorz Kondrak, Xiang Kong, Ioannis Konstas, Yannis Korkontzelos, Leila Kosseim, Lili Kotlerman, Zornitsa Kozareva, Mikhail Kozhevnikov, Julia Kreutzer, Jayant Krishnamurthy, Kriste Krstovski, Canasai Kruengkrai, Udo Kruschwitz, Germán Kruszewski, Lun-Wei Ku, Marco Kuhlmann, Roland Kuhn, Vivek Kulkarni, Shankar Kumar, Jonathan K. Kummerfeld, Gourab Kundu, Tsung-Ting Kuo, Sadao Kurohashi, Polina Kuznetsova, Maximilian Köper, Sandra Kübler, Majid Laali, Gorka Labaka, Wai Lam, Mathias Lambert, Patrik Lambert, Vasileios Lampos, Gerasimos Lampouras, Wuwei Lan, Ni Lao, Mirella Lapata, Romain Laroche, Jey Han Lau, Alon Lavie, Carolin Lawrence, Angeliki Lazaridou, Phong Le, Joseph Le Roux, Robert Leaman, Chia-ying Lee, Hung-yi Lee, John Lee, Lin-shan Lee, Sungjin Lee, Yoong Keok Lee, Young-Suk Lee, Els Lefever, Tao Lei, Gaël Lejeune, Alessandro Lenci, Piroska Lendvai, Chee Wee (Ben) Leong, James Lester, Gregor Leusch, Effi Levi, Roger Levy, Baoli Li, Binyang Li, Cheng-Te Li, Chenliang Li, Haibo Li, Juanzi Li, Junhui Li, Junyi Jessy Li, Lei Li, Liangyou Li, Lishuang Li, Miles Li, Piji Li, Sujian Li, Xin Li, Xiujun Li, Yanen Li, Yunyao Li, Zhenghua Li, Zhixing Li, Maria Liakata, Chen Liang, Paul Pu Liang, Ming Liao, Xiangwen Liao, Anne-Laure Ligozat, Nut Limsopatham, Bill Yuchen Lin, Chenghua Lin, Chuan-Jie Lin, Dekang Lin, Xi Victoria Lin, Xiao Ling, Tal Linzen, Pierre Lison, Marina Litvak, Bing Liu, Changsong Liu, Fei Liu, Jiangming Liu, Jingjing Liu, Kai Liu, Kang Liu, Lemao Liu, Liyuan Liu, Qian Liu, Shujie Liu, Xiaodong Liu, Yang Liu, Yang Liu, Yiqun Liu, Zhengzhong Liu, Zhiyuan Liu, Nikola Ljubešić, Elena Lloret, José Lopes, Oier Lopez de Lacalle, Aurelio Lopez-Lopez, Annie Louis, Ryan Lowe, Sharid Loáiciga, Qin Lu, Wei Lu, Yi Luan, Wencan Luo, Zhunchen Luo, Anh Tuan Luu, Teresa Lynn, Chunchuan Lyu, Jing Ma, Mingbo Ma, Shuming Ma, Wei-Yun Ma, Xuezhe Ma, Klaus Macherey, Wolfgang Macherey, Pranava Swaroop Madhyastha, Walid Magdy, Wolfgang Maier, Jean Maillard, Peter Makarov, Prodromos Malakasiotis, Alfredo Maldonado, Andreas Maletti, Igor Malioutov, Shervin Malmasi, Radhika Mamidi, Suresh Manandhar, Gideon Mann, Ana Marasovic, Daniel Marcu, Mitchell Marcus, Alex Marin, André F. T. Martins, Bruno Martins, David Martins de Matos, Sebastian Martschat, Héctor Martínez Alonso, Marco Matassoni, Prashant Mathur, Yuichiroh Matsubayashi, Yuji Matsumoto, Takuya Matsuzaki, Austin Matthews, Abhinav Maurya, Arne Mauser, Jonathan May, Stephen Mayhew, Diana Maynard, Karen Mazidi, David McAllester, Diana McCarthy, David McClosky, John Philip McCrae, Stephen McGregor, Tara McIntosh, Kathy McKeown, Hongyuan Mei, Yelena Mejova, Pablo Mendes, Arul Menezes, Fandong Meng, Helen Meng, Mohsen Mesgar, Florian Metze, Haitao Mi, Yishu Miao, Slava Mikhaylov, Timothy Miller, Tristan Miller, Bonan Min, Anne-Lyse Minard, Pasquale Minervini, Michael Minock, Paramita Mirza, Abhijit Mishra, Dipendra Misra, Jeff Mitchell, Akiva Miura, Daichi Mochihashi, Ashutosh Modi, Muhidin Mohamed, Saif Mohammad, Michael Mohler, Luis Gerardo Mojica de la Vega, Manuel Montes, Christof Monz, Taesun Moon, Roser Morante, Véronique Moriceau, Hajime Morita, Alessandro Moschitti, Dana Movshovitz-Attias, Nikola Mrkšić, Tingting Mu, Animesh Mukherjee, Dragos Munteanu, Yugo Murawaki, Kenton Murray, Sung-Hyon Myaeng, Maria Nadejde, Masaaki Nagata, Ryo Nagata, Ajay Nagesh, Tetsuji Nakagawa, Mikio Nakano, Yukiko Nakano, Ndapa Nakashole, Hideki Nakayama, Preslav Nakov, Jason Naradowsky, Karthik Narasimhan, Alexis Nasr, Roberto Navigli, Adeline Nazarenko, Mark-Jan Nederhof, Matteo Negri, Aida Nematzadeh, Goran Nenadic, Yael Netzer, Graham Neubig, Guenter Neumann, Dominick Ng, Hwee Tou Ng, Jun-Ping Ng, Vincent Ng, Minh-Quoc Nghiem, Axel-Cyrille Ngonga Ngomo, Thien Huu Nguyen, Truc-Vien T. Nguyen, Viet-An Nguyen, Jian-Yun Nie, Jan Niehues, Ivelina Nikolova, Nobal Bikram Niraula, Xing Niu, Joakim Nivre, Hiroshi Noji, Thanapon Noraset, Joel Nothman, Pierre Nugues, Aurélie Névéol, Brendan O'Connor, Tim O'Gorman, Yusuke Oda, Kemal Oflazer, Jong-Hoon Oh, Kiyonori Ohtake, Naoaki Okazaki, Manabu Okumura, Constantin Orasan, Vicente Ordonez, Petya Osenova, Simon Ostermann, Myle Ott, Katja Ovchinnikova, Diarmuid Ó Séaghdha, Arzucan Özgür, Lilja Øvrelid, Ulrike Pado, Martha Palmer, Joao Palotti, Shimei Pan, Xiaoman Pan, Alexander Panchenko, Alexandros Papangelis, Nikos Papasarantopoulos, Nikolaos Pappas, Ankur Parikh, Niki Parmar, Patrick Paroubek, Carla Parra Escartín, Tommaso Pasini, Siddharth Patwardhan, Michael J. Paul, Adam Pauls, Ellie Pavlick, Adam Pease, Baolin Peng, Hao Peng, Haoruo Peng, Nanyun Peng, Xiaochang Peng, Sergio Penkale, Ethan Perez, Julien Perez, Gabriele Pergola, Sandro Pezzelle, Karl Pichotta, Olivier Pietquin, Mohammad Taher Pilehvar, Manfred Pinkal, Juan Pino, Yuval Pinter, Barbara Plank, Emmanouil Antonios Platanios, Julien Plu, Massimo Poesio, Adam Poliak, Edoardo Maria Ponti, Simone Paolo Ponzetto, Andrei Popescu-Belis, Maja Popović, Fred Popowich, Soujanya Poria, Christopher Potts, Amir Pouran Ben Veyseh, Forough Poursabzi-Sangdeh, Vinodkumar Prabhakaran, Shrimai Prabhumoye, Daniel Preoţiuc-Pietro, Emily Prud'hommeaux, Laurent Prévot, Jay Pujara, Matthew Purver, Valentina Pyatkin, Sampo Pyysalo, Verónica Pérez-Rosas, Ashequl Qadir, Jing Qian, Longhua Qian, Lianhui Qin, Pengda Qin, Long Qiu, Minghui Qiu, Lizhen Qu, Meng Qu, Chris Quirk, Ella Rabinovich, Alessandro Raganato, Altaf Rahman, Taraka Rama, Bhuvana Ramabhadran, Vikram Ramanarayanan, Maya Ramanath, Carlos Ramisch, Delip Rao, Sudha Rao, Mohammad Sadegh Rasooli, Pushpendre Rastogi, Sagnik Ray Choudhury, Simon Razniewski, Ines Rehbein, Marek Rei, Ehud Reiter, Steffen Remus, Gary Ren, Zhaochun Ren, Ludovic Rheault, Corentin Ribeyre, Giuseppe Riccardi, Matthew Richardson, Mark Riedl, Martin Riedl, Jason Riesa, German Rigau, Fabio Rinaldi, Brian Riordan, Kirk Roberts, Tim Rocktäschel, Pedro Rodriguez, Ina Roesiger, Marcus Rohrbach, Oleg Rokhlenko, Stephen Roller, Salvatore Romeo, Sara Rosenthal, Alessandro Rossi, Paolo Rosso, Benjamin Roth, Michael Roth, Sascha Rothe, Subhro Roy, Alla Rozovskaya, Sebastian Ruder, Rachel Rudinger, Frank Rudzicz, Vasile Rus, Alexander Rush, Irene Russo, Delia Rusu, Attapol Rutherford, Derek Ruths, Ashish Sabharwal, Mrinmaya Sachan, Kugatsu Sadamitsu, Fatiha Sadat, Markus Saers, Kenji Sagae, Rishiraj Saha Roy, Magnus Sahlgren, Sunil Sahu, Hassan Sajjad, Keisuke Sakaguchi, Tetsuya Sakai, Sakriani Sakti, Mohammad Salameh, Elizabeth Salesky, Avneesh Saluja, Rajhans Samdani, Germán Sanchis-Trilles, Enrico Santus, Maarten Sap, Naomi Saphra, Zahra Sarabi, Ryohei Sasano, Agata Savary, Hassan Sawaf, Asad Sayeed, Carolina Scarton, Odette Scharenborg, Tatjana Scheffler, Niko Schenk, David Schlangen, Jonathan Schler, Natalie Schluter, Allen Schmaltz, Helmut Schmid, Nathan Schneider, William Schuler, Björn Schuller, H. Andrew Schwartz, Lane Schwartz, Roy Schwartz, Stephanie Schwartz, Holger Schwenk, Djamé Seddah, Frederique Segond, Satoshi Sekine, Ethan Selfridge, Jean Senellart, Rico Sennrich, Minjoon Seo, Christophe Servan, Burr Settles, Lei Sha, Izhak Shafran, Kashif Shah, Samira Shaikh, Rebecca Sharp, Lanbo She, Shiqi Shen, Wade Shen, Fiona Shen-Bayh, Baoxu Shi, Bei Shi, Bowen Shi, Xiaodong Shi, Hyopil Shin, Chaitanya Shivade, Avi Sil, Fabrizio Silvestri, Yanchuan Sim, Patrick Simianer, Dan Simonson, Kiril Simov, Sameer Singh, Steve Skiena, Neil Smalheiser, Sharon Small, Noah A. Smith, Parinaz Sobhani, Richard Socher, Artem Sokolov, Hyun-Je Song, Kaisong Song, Linfeng Song, Yan Song, Yang Song, Yangqiu Song, Radu Soricut, Aitor Soroa, Daniil Sorokin, Axel Soto, Lucia Specia, Matthias Sperber, Richard Sproat, Vivek Srikumar, Somayajulu Sripada, Shashank Srivastava, Miloš Stanojević, Mark Steedman, Josef Steinberger, Shane Steinert-Threlkeld, Pontus Stenetorp, Mark Stevenson, Suzanne Stevenson, Ian Stewart, Svetlana Stoyanchev, Veselin Stoyanov, Carlo Strapparava, Emma Strubell, Keh-Yih Su, Pei-Hao Su, Yu Su, Jinsong Su, Amarnag Subramanya, Katsuhito Sudoh, Alane Suhr, Elior Sulem, Md Arafat Sultan, Fei Sun, Huan Sun, Le Sun, Ming Sun, Mihai Surdeanu, Hisami Suzuki, Swabha Swayamdipta, Idan Szpektor, Jakub Szymanik, Felipe Sánchez-Martínez, Anders Søgaard, Jan Šnajder, Sanja Štajner, Hiroya Takamura, Sho Takase, David Talbot, Partha Talukdar, Akihiro Tamura, Chenhao Tan, Jiwei Tan, Niket Tandon, Duyu Tang, Jian Tang, Jiliang Tang, Jintao Tang, Siliang Tang, Chris Tanner, Xavier Tannier, Chongyang Tao, Fangbo Tao, Makarand Tapaswi, Simone Teufel, Kapil Thadani, Jesse Thomason, Fei Tian, Ran Tian, Jörg Tiedemann, Christoph Tillmann, Gaurav Singh Tomar, Nadi Tomeh, Marc Tomlinson, Sara Tonelli, Antonio Toral, Kentaro Torisawa, Shubham Toshniwal, Ke Tran, Isabel Trancoso, Rocco Tripodi, Adam Trischler, Chen-Tse Tsai, Richard Tzong-Han Tsai, Yuta Tsuboi, Yoshimasa Tsuruoka, Lifu Tu, Gokhan Tur, Marco Turchi, Ferhan Ture, Kateryna Tymoshenko, Oscar Täckström, Stefan Ultes, Lyle Ungar, Shyam Upadhyay, Olga Uryupina, Dmitry Ustalov, Masao Utiyama, Antonio Uva, Sowmya Vajjala, Tim Van de Cruys, Rob van der Goot, Lonneke van der Plas, Menno van Zaanen, Vincent Vandeghinste, David Vandyke, Sumithra Velupillai, Alakananda Vempala, Sriram Venkatapathy, Ashish Venugopal, Patrick Verga, Marc Verhagen, Yannick Versley, Guido Vetere, David Vilar, David Vilares, Aline Villavicencio, Andreas Vlachos, Rob Voigt, Svitlana Volkova, Ngoc Thang Vu, Thuy Vu, Ivan Vulić, Yogarshi Vyas, Byron Wallace, Matthew Walter, Stephen Wan, Alex Wang, Baoxun Wang, Chao Wang, Chenguang Wang, Chuan Wang, Daling Wang, Dingquan Wang, Houfeng Wang, Longyue Wang, Lu Wang, Mingxuan Wang, Quan Wang, Rui Wang, Shaojun Wang, Shuohang Wang, Wei Wang, Xiaolong Wang, Xin Wang, Xing Wang, Zhichun Wang, Zhiguo Wang, Zhongqing Wang, Leo Wanner, Nigel Ward, Zeerak Waseem, Shinji Watanabe, Nick Webb, Ingmar Weber, Julie Weeds, Furu Wei, Zhongyu Wei, Gerhard Weikum, David Weir, Yu Weiren, Ralph Weischedel, David Weiss, Dirk Weissenborn, Johannes Welbl, Charles Welch, Laura Wendlandt, Aaron Steven White, Michael Wiegand, John Wieting, Derry Tanti Wijaya, Jake Williams, Jason D Williams, Steven Wilson, Sam Wiseman, Guillaume Wisniewski, Travis Wolfe, Derek F. Wong, Tak-Lam Wong, Hua Wu, Jiawei Wu, Yunfang Wu, Zhaohui Wu, Joern Wuebker, Tong Xiao, Xinyan Xiao, Yanghua Xiao, Pengtao Xie, Deyi Xiong, Wenhan Xiong, Kun Xu, Ruifeng Xu, Tong Xu, Wei Xu, Wenduan Xu, Yadollah Yaghoobzadeh, Takehiro Yamamoto, Rui Yan, Baosong Yang, Bishan Yang, Diyi Yang, Liu Yang, Weiwei Yang, Yaqin Yang, Yi Yang, Yiben Yang, Zhilin Yang, Zichao Yang, Helen Yannakoudakis, Tae Yano, Jin-ge Yao, Wenlin Yao, Ziyu Yao, Mark Yatskar, Lana Yeganova, Wen-wai Yim, Seid Muhie Yimam, Pengcheng Yin, Qingyu Yin, Wenpeng Yin, Dani Yogatama, Naoki Yoshinaga, Masaharu Yoshioka, Dian Yu, Jianfei Yu, Kai Yu, Liang-Chih Yu, Mo Yu, Ning Yu, François Yvon, Roberto Zamparelli, Marcos Zampieri, Fabio Massimo Zanzotto, Rowan Zellers, Daojian Zeng, Jichuan Zeng, Xingshan Zeng, Chrysoula Zerva, Luke Zettlemoyer, Deniz Zeyrek, Ke Zhai, Biao Zhang, Congle Zhang, Dongdong Zhang, Hao Zhang, Jiajun Zhang, Jingyi Zhang, Lei Zhang, Meishan Zhang, Min Zhang, Min Zhang, Qi Zhang, Sheng Zhang, Wei-Nan Zhang, Xingxing Zhang, Yongfeng Zhang, Yuan Zhang, Yuchen Zhang, Yue Zhang, Yuhao Zhang, Bing Zhao, Dongyan Zhao, Hai Zhao, Jun Zhao, Kai Zhao, Lin Zhao, Sendong Zhao, Tiancheng

Zhao, Wayne Xin Zhao, Alisa Zhila, Victor Zhong, Guodong Zhou, Hao Zhou, Junsheng Zhou, Ming Zhou, Hao Zhu, Lixing Zhu, Muhua Zhu, Qi Zhu, Xiaodan Zhu, Yukun Zhu, Ayah Zirikly, Yftah Ziser, Michael Zock, Chengqing Zong.

Invited Speaker: Johan Bos, University of Groningen
The Moment of Meaning and the Future of Computational Semantics

Abstract: There are many recent advances in semantic parsing: we see a rising number of semantically annotated corpora and there is exciting technology (such as neural networks) to be explored. In this talk I will discuss what role computational semantics could play in future natural language processing applications (including fact checking and machine translation). I will argue that we should not just look at semantic parsing, but that things can get really interesting when we can use language-neutral meaning representations to draw (transparent) inferences. The main ideas will be exemplified by the parallel meaning bank, a new corpus comprising texts annotated with formal meaning representations for English, Dutch, German and Italian.

Bio: Johan Bos is Professor of Computational Semantics at the University of Groningen (Netherlands). He received his doctorate from the Computational Linguistics Department at the University of the Saarland (Germany) and held post-doc positions at the University of Edinburgh (UK) and the La Sapienza University in Rome (Italy). In 2010, he moved to his current position in Groningen, leading the computational semantics group. Bos is the developer of Boxer, a state-of-the-art wide-coverage semantic parser for English, initiator of the Groningen Meaning Bank, a large semantically-annotated corpus of texts, and inventor of Wordrobe, a game with a purpose for semantic annotation. Bos received a $1.5-million Vici grant from NWO (Netherlands Organisation for Scientific Research) in 2015 to investigate the role of meaning in human and machine translation. A concrete outcome of this project is the Parallel Meaning Bank containing detailed meaning representations for English, German, Dutch and Italian sentences.

Invited Speaker: Julia Hirschberg, Columbia University
Truth or Lie? Spoken Indicators of Deception in Speech

Abstract: Detecting deception from various forms of human behavior is a longstanding research goal which is of considerable interest to the military, law enforcement, corporate security, social services and mental health workers. However, both humans and polygraphs are very poor at this task. We describe more accurate methods we have developed to detect deception automatically from spoken language. Our classifiers are trained on the largest cleanly recorded corpus of within-subject deceptive and non-deceptive speech that has been collected. To distinguish truth from lie we make use of acoustic-prosodic, lexical, demographic, and personality features. We further examine differences in deceptive behavior based upon gender, personality, and native language (Mandarin Chinese vs. English), comparing our systems to human performance. We extend our studies to identify cues in trusted speech vs. mistrusted speech and how these features differ by speaker and by listener. Why does a listener believe a lie?

Bio: Julia Hirschberg is Percy K. and Vida L. W. Hudson Professor and Chair of Computer Science at Columbia University. She previously worked at Bell Laboratories and AT&T Labs where she created the HCI Research Department. She has been editor of Computational Linguistics and Speech Communication, is a fellow of AAAI, ISCA, ACL, ACM, and IEEE, and a member of the National Academy of Engineering. She received the IEEE James L. Flanagan Speech and Audio Processing Award and the ISCA Medal for Scientific Achievement. She currently serves on the IEEE Speech and Language Processing Technical Committee, is co-chair of the CRA-W Board, and has worked for diversity for many years at AT&T and Columbia. She works on spoken language processing and NLP, studying text-to-speech synthesis, spoken dialogue systems, entrainment in conversation, detection of deceptive and emotional speech, hedging behavior, and linguistic code-switching (language mixing).

Invited Speaker: Gideon Mann, Bloomberg L.P.

Understanding the News that Moves Markets

Abstract: Since the dawn of human civilization, finance and language technology have been connected. However, only recently have advances in statistical language understanding, and an ever-increasing thirst for market advantage, led to the widespread application of natural language technology across the global capital markets. This talk will review the ways in which language technology is enabling market participants to quickly understand and respond to major world events and breaking business news. It will outline the state of the art in applications of NLP to finance and highlight open problems that are being addressed by emerging research.

Bio: Gideon Mann is the Head of Data Science at Bloomberg L.P., where he guides the strategic direction for machine learning, natural language processing (NLP) and search across the company. He is part of the leadership team for the Office of the CTO. He served as a founding member of both the Data for Good Exchange (D4GX), an annual conference on data science applications for social good, and the Shift Commission on Work, Workers and Technology. He has also been active in academic research in fact extraction, weakly-supervised learning, and distributed optimization. Recently, he has also been interested in applications of machine learning to problems in software engineering. From 2007 to 2014, he worked at Google Research in New York City, and his team built core machine learning libraries, released the Google Prediction API, and developed Colaboratory. Mann graduated Brown University in 1999 and received a Ph.D. from The Johns Hopkins University in 2006.

Table of Contents

Privacy-preserving Neural Representations of Text
Maximin Coavoux, Shashi Narayan and Shay B. Cohen 1

Adversarial Removal of Demographic Attributes from Text Data
Yanai Elazar and Yoav Goldberg 11

DeClarE: Debunking Fake News and False Claims using Evidence-Aware Deep Learning
Kashyap Popat, Subhabrata Mukherjee, Andrew Yates and Gerhard Weikum 22

It's going to be okay: Measuring Access to Support in Online Communities
Zijian Wang and David Jurgens 33

Detecting Gang-Involved Escalation on Social Media Using Context
Serina Chang, Ruiqi Zhong, Ethan Adams, Fei-Tzin Lee, Siddharth Varia, Desmond Patton, William Frey, Chris Kedzie and Kathy McKeown 46

Reasoning about Actions and State Changes by Injecting Commonsense Knowledge
Niket Tandon, Bhavana Dalvi, Joel Grus, Wen-tau Yih, Antoine Bosselut and Peter Clark 57

Collecting Diverse Natural Language Inference Problems for Sentence Representation Evaluation
Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White and Benjamin Van Durme 67

Textual Analogy Parsing: What's Shared and What's Compared among Analogous Facts
Matthew Lamm, Arun Chaganty, Christopher D. Manning, Dan Jurafsky and Percy Liang 82

SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference
Rowan Zellers, Yonatan Bisk, Roy Schwartz and Yejin Choi 93

TwoWingOS: A Two-Wing Optimization Strategy for Evidential Claim Verification
Wenpeng Yin and Dan Roth 105

Associative Multichannel Autoencoder for Multimodal Word Representation
Shaonan Wang, Jiajun Zhang and Chengqing Zong 115

Game-Based Video-Context Dialogue
Ramakanth Pasunuru and Mohit Bansal 125

simNet: Stepwise Image-Topic Merging Network for Generating Detailed and Comprehensive Image Captions
Fenglin Liu, Xuancheng Ren, Yuanxin Liu, Houfeng Wang and Xu Sun 137

Multimodal Language Analysis with Recurrent Multistage Fusion
Paul Pu Liang, Ziyin Liu, AmirAli Bagher Zadeh and Louis-Philippe Morency 150

Temporally Grounding Natural Sentence in Video
Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie and Tat-Seng Chua 162

PreCo: A Large-scale Dataset in Preschool Vocabulary for Coreference Resolution
Hong Chen, Zhenhua Fan, Hao Lu, Alan Yuille and Shu Rong 172

Adversarial Transfer Learning for Chinese Named Entity Recognition with Self-Attention Mechanism
Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao and Shengping Liu 182

Using Linguistic Features to Improve the Generalization Capability of Neural Coreference Resolvers
Nafise Sadat Moosavi and Michael Strube 193

Neural Segmental Hypergraphs for Overlapping Mention Recognition
Bailin Wang and Wei Lu 204

Variational Sequential Labelers for Semi-Supervised Learning
Mingda Chen, Qingming Tang, Karen Livescu and Kevin Gimpel 215

Joint Representation Learning of Cross-lingual Words and Entities via Attentive Distant Supervision
Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Chengjiang Li, Xu Chen and Tiansi Dong 227

Deep Pivot-Based Modeling for Cross-language Cross-domain Transfer with Minimal Guidance
Yftah Ziser and Roi Reichart 238

Multi-lingual Common Semantic Space Construction via Cluster-consistent Word Embedding
Lifu Huang, Kyunghyun Cho, Boliang Zhang, Heng Ji and Kevin Knight 250

Unsupervised Multilingual Word Embeddings
Xilun Chen and Claire Cardie 261

CLUSE: Cross-Lingual Unsupervised Sense Embeddings
Ta Chung Chi and Yun-Nung Chen 271

Adversarial Propagation and Zero-Shot Cross-Lingual Transfer of Word Vector Specialization
Edoardo Maria Ponti, Ivan Vulić, Goran Glavaš, Nikola Mrkšić and Anna Korhonen 282

Improving Cross-Lingual Word Embeddings by Meeting in the Middle
Yerai Doval, Jose Camacho-Collados, Luis Espinosa Anke and Steven Schockaert 294

WikiAtomicEdits: A Multilingual Corpus of Wikipedia Edits for Modeling Language and Discourse
Manaal Faruqui, Ellie Pavlick, Ian Tenney and Dipanjan Das 305

On the Relation between Linguistic Typology and (Limitations of) Multilingual Language Modeling
Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart and Anna Korhonen 316

A Fast, Compact, Accurate Model for Language Identification of Codemixed Text
Yuan Zhang, Jason Riesa, Daniel Gillick, Anton Bakalov, Jason Baldridge and David Weiss . . . 328

Personalized Microblog Sentiment Classification via Adversarial Cross-lingual Multi-task Learning
Weichao Wang, Shi Feng, Wei Gao, Daling Wang and Yifei Zhang 338

Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks
Zhichun Wang, Qingsong Lv, Xiaohan Lan and Yu Zhang 349

Cross-lingual Lexical Sememe Prediction
Fanchao Qi, Yankai Lin, Maosong Sun, Hao Zhu, Ruobing Xie and Zhiyuan Liu 358

Neural Cross-Lingual Named Entity Recognition with Minimal Resources
Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith and Jaime Carbonell 369

A Stable and Effective Learning Strategy for Trainable Greedy Decoding
Yun Chen, Victor O.K. Li, Kyunghyun Cho and Samuel Bowman 380

Addressing Troublesome Words in Neural Machine Translation
Yang Zhao, Jiajun Zhang, Zhongjun He, Chengqing Zong and Hua Wu 391

Top-down Tree Structured Decoding with Syntactic Connections for Neural Machine Translation and Parsing
Jetic Gū, Hassan S. Shavarani and Anoop Sarkar 401

XL-NBT: A Cross-lingual Neural Belief Tracking Framework
Wenhu Chen, Jianshu Chen, Yu Su, Xin Wang, Dong Yu, Xifeng Yan and William Yang Wang 414

Contextual Parameter Generation for Universal Neural Machine Translation
Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig and Tom Mitchell 425

Back-Translation Sampling by Targeting Difficult Words in Neural Machine Translation
Marzieh Fadaee and Christof Monz 436

Multi-Domain Neural Machine Translation with Word-Level Domain Context Discrimination
Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin and Jianqiang Zhao . . . 447

A Discriminative Latent-Variable Model for Bilingual Lexicon Induction
Sebastian Ruder, Ryan Cotterell, Yova Kementchedjhieva and Anders Søgaard 458

Non-Adversarial Unsupervised Word Translation
Yedid Hoshen and Lior Wolf 469

Semi-Autoregressive Neural Machine Translation
Chunqi Wang, Ji Zhang and Haiqing Chen 479

Understanding Back-Translation at Scale
Sergey Edunov, Myle Ott, Michael Auli and David Grangier 489

Bootstrapping Transliteration with Constrained Discovery for Low-Resource Languages
Shyam Upadhyay, Jordan Kodner and Dan Roth 501

NORMA: Neighborhood Sensitive Maps for Multilingual Word Embeddings
Ndapa Nakashole 512

Adaptive Multi-pass Decoder for Neural Machine Translation
Xinwei Geng, Xiaocheng Feng, Bing Qin and Ting Liu 523

Improving the Transformer Translation Model with Document-Level Context
Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang and Yang Liu 533

MTNT: A Testbed for Machine Translation of Noisy Text
Paul Michel and Graham Neubig 543

SimpleQuestions Nearly Solved: A New Upperbound and Baseline Approach
Michael Petrochuk and Luke Zettlemoyer 554

Phrase-Indexed Question Answering: A New Challenge for Scalable Document Comprehension
Minjoon Seo, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi and Hannaneh Hajishirzi 559

Ranking Paragraphs for Improving Answer Recall in Open-Domain Question Answering
Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko and Jaewoo Kang 565

Cut to the Chase: A Context Zoom-in Network for Reading Comprehension
Sathish Reddy Indurthi, Seunghak Yu, Seohyun Back and Heriberto Cuayahuitl 570

Adaptive Document Retrieval for Deep Question Answering
Bernhard Kratzwald and Stefan Feuerriegel 576

Why is unsupervised alignment of English embeddings from different algorithms so hard?
Mareike Hartmann, Yova Kementchedjhieva and Anders Søgaard 582

Quantifying Context Overlap for Training Word Embeddings
Yimeng Zhuang, Jinghui Xie, Yinhe Zheng and Xuan Zhu 587

Neural Latent Relational Analysis to Capture Lexical Semantic Relations in a Vector Space
Koki Washio and Tsuneaki Kato 594

Generalizing Word Embeddings using Bag of Subwords
Jinman Zhao, Sidharth Mudgal and Yingyu Liang 601

Neural Metaphor Detection in Context
Ge Gao, Eunsol Choi, Yejin Choi and Luke Zettlemoyer 607

Distant Supervision from Disparate Sources for Low-Resource Part-of-Speech Tagging
Barbara Plank and Željko Agić 614

Unsupervised Bilingual Lexicon Induction via Latent Variable Models
Zi-Yi Dou, Zhi-Hao Zhou and Shujian Huang 621

Learning Unsupervised Word Translations Without Adversaries
Tanmoy Mukherjee, Makoto Yamada and Timothy Hospedales 627

Adversarial Training for Multi-task and Multi-lingual Joint Modeling of Utterance Intent Classification
Ryo Masumura, Yusuke Shinohara, Ryuichiro Higashinaka and Yushi Aono 633

Surprisingly Easy Hard-Attention for Sequence to Sequence Learning
Shiv Shankar, Siddhant Garg and Sunita Sarawagi 640

Joint Learning for Emotion Classification and Emotion Cause Detection
Ying Chen, Wenjun Hou, Xiyao Cheng and Shoushan Li 646

Exploring Optimism and Pessimism in Twitter Using Deep Learning
Cornelia Caragea, Liviu P. Dinu and Bogdan Dumitru 652

Predicting News Headline Popularity with Syntactic and Semantic Knowledge Using Multi-Task Learning
Sotiris Lamprinidis, Daniel Hardt and Dirk Hovy 659

Hybrid Neural Attention for Agreement/Disagreement Inference in Online Debates
Di Chen, Jiachen Du, Lidong Bing and Ruifeng Xu 665

Increasing In-Class Similarity by Retrofitting Embeddings with Demographic Information
Dirk Hovy and Tommaso Fornaciari 671

A Syntactically Constrained Bidirectional-Asynchronous Approach for Emotional Conversation Generation
Jingyuan Li and Xiao Sun 678

Auto-Dialabel: Labeling Dialogue Data with Unsupervised Learning
Chen Shi, Qi Chen, Lei Sha, Sujian Li, Xu Sun, Houfeng Wang and Lintao Zhang 684

Extending Neural Generative Conversational Model using External Knowledge Sources
Prasanna Parthasarathi and Joelle Pineau 690

Modeling Temporality of Human Intentions by Domain Adaptation
Xiaolei Huang, Lixing Liu, Kate Carey, Joshua Woolley, Stefan Scherer and Brian Borsari 696

An Auto-Encoder Matching Model for Learning Utterance-Level Semantic Dependency in Dialogue Generation
Liangchen Luo, Jingjing Xu, Junyang Lin, Qi Zeng and Xu Sun 702

A Dataset for Document Grounded Conversations
Kangyan Zhou, Shrimai Prabhumoye and Alan W Black 708

Out-of-domain Detection based on Generative Adversarial Network
Seonghan Ryu, Sangjun Koo, Hwanjo Yu and Gary Geunbae Lee 714

Listening Comprehension over Argumentative Content
Shachar Mirkin, Guy Moshkowich, Matan Orbach, Lili Kotlerman, Yoav Kantor, Tamar Lavee, Michal Jacovi, Yonatan Bilu, Ranit Aharonov and Noam Slonim 719

Using active learning to expand training data for implicit discourse relation recognition
Yang Xu, Yu Hong, Huibin Ruan, Jianmin Yao, Min Zhang and Guodong Zhou 725

Learning To Split and Rephrase From Wikipedia Edit History
Jan A. Botha, Manaal Faruqui, John Alex, Jason Baldridge and Dipanjan Das 732

BLEU is Not Suitable for the Evaluation of Text Simplification
Elior Sulem, Omri Abend and Ari Rappoport 738

S2SPMN: A Simple and Effective Framework for Response Generation with Relevant Information
Jiaxin Pei and Chenliang Li 745

Improving Reinforcement Learning Based Image Captioning with Natural Language Prior
Tszhang Guo, Shiyu Chang, Mo Yu and Kun Bai 751

Training for Diversity in Image Paragraph Captioning
Luke Melas-Kyriazi, Alexander Rush and George Han 757

A Graph-theoretic Summary Evaluation for ROUGE
Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong and Fang Chen 762

Guided Neural Language Generation for Abstractive Summarization using Abstract Meaning Representation
Hardy Hardy and Andreas Vlachos 768

Evaluating Multiple System Summary Lengths: A Case Study
Ori Shapira, David Gabay, Hadar Ronen, Judit Bar-Ilan, Yael Amsterdamer, Ani Nenkova and Ido Dagan 774

Neural Latent Extractive Document Summarization
Xingxing Zhang, Mirella Lapata, Furu Wei and Ming Zhou 779

On the Abstractiveness of Neural Document Summarization
Fangfang Zhang, Jin-ge Yao and Rui Yan 785

Automatic Essay Scoring Incorporating Rating Schema via Reinforcement Learning
Yucheng Wang, Zhongyu Wei, Yaqian Zhou and Xuanjing Huang 791

Identifying Well-formed Natural Language Questions
Manaal Faruqui and Dipanjan Das 798

Self-Governing Neural Networks for On-Device Short Text Classification
Sujith Ravi and Zornitsa Kozareva 804

HFT-CNN: Learning Hierarchical Category Structure for Multi-label Short Text Categorization
Kazuya Shimura, Jiyi Li and Fumiyo Fukumoto 811

A Hierarchical Neural Attention-based Text Classifier
Koustuv Sinha, Yue Dong, Jackie Chi Kit Cheung and Derek Ruths 817

Labeled Anchors and a Scalable, Transparent, and Interactive Classifier
Jeffrey Lund, Stephen Cowley, Wilson Fearn, Emily Hales and Kevin Seppi 824

Coherence-Aware Neural Topic Modeling
Ran Ding, Ramesh Nallapati and Bing Xiang 830

Utilizing Character and Word Embeddings for Text Normalization with Sequence-to-Sequence Models
Daniel Watson, Nasser Zalmout and Nizar Habash 837

Topic Intrusion for Automatic Topic Model Evaluation
Shraey Bhatia, Jey Han Lau and Timothy Baldwin 844

Supervised and Unsupervised Methods for Robust Separation of Section Titles and Prose Text in Web Documents
Abhijith Athreya Mysore Gopinath, Shomir Wilson and Norman Sadeh 850

SwitchOut: an Efficient Data Augmentation Algorithm for Neural Machine Translation
Xinyi Wang, Hieu Pham, Zihang Dai and Graham Neubig 856

Improving Unsupervised Word-by-Word Translation with Language Model and Denoising Autoencoder
Yunsu Kim, Jiahui Geng and Hermann Ney 862

Decipherment of Substitution Ciphers with Neural Language Models
Nishant Kambhatla, Anahita Mansouri Bigvand and Anoop Sarkar 869

Rapid Adaptation of Neural Machine Translation to New Languages
Graham Neubig and Junjie Hu 875

Compact Personalized Models for Neural Machine Translation
Joern Wuebker, Patrick Simianer and John DeNero 881

Self-Governing Neural Networks for On-Device Short Text Classification
Sujith Ravi and Zornitsa Kozareva 887

Supervised Domain Enablement Attention for Personalized Domain Classification
Joo-Kyung Kim and Young-Bum Kim 894

A Deep Neural Network Sentence Level Classification Method with Context Information
Xingyi Song, Johann Petrak and Angus Roberts 900

Towards Dynamic Computation Graphs via Sparse Latent Structure
Vlad Niculae, André F. T. Martins and Claire Cardie 905

Convolutional Neural Networks with Recurrent Neural Filters
Yi Yang 912

Exploiting Rich Syntactic Information for Semantic Parsing with Graph-to-Sequence Model
Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen and Vadim Sheinin 918

Retrieval-Based Neural Code Generation
Shirley Anugrah Hayati, Raphael Olivier, Pravalika Avvaru, Pengcheng Yin, Anthony Tomasic and Graham Neubig 925

SQL-to-Text Generation with Graph-to-Sequence Model
Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng and Vadim Sheinin 931

Generating Syntactic Paraphrases
Emilie Colin and Claire Gardent 937

Neural-Davidsonian Semantic Proto-role Labeling
Rachel Rudinger, Adam Teichert, Ryan Culkin, Sheng Zhang and Benjamin Van Durme 944

Conversational Decision-Making Model for Predicting the King's Decision in the Annals of the Joseon Dynasty
JinYeong Bak and Alice Oh 956

Toward Fast and Accurate Neural Discourse Segmentation
Yizhong Wang, Sujian Li and Jingfeng Yang 962

A Dataset for Telling the Stories of Social Media Videos
Spandana Gella, Mike Lewis and Marcus Rohrbach 968

Cascaded Mutual Modulation for Visual Reasoning
Yiqun Yao, Jiaming Xu, Feng Wang and Bo Xu 975

How agents see things: On visual representations in an emergent language game
Diane Bouchacourt and Marco Baroni 981

Attention-Based Capsule Networks with Dynamic Routing for Relation Extraction
Ningyu Zhang, Shumin Deng, Zhanling Sun, Xi Chen, Wei Zhang and Huajun Chen 986

Put It Back: Entity Typing with Language Model Enhancement
Ji Xin, Hao Zhu, Xu Han, Zhiyuan Liu and Maosong Sun 993

Event Detection with Neural Networks: A Rigorous Empirical Evaluation
Walker Orr, Prasad Tadepalli and Xiaoli Fern 999

PubSE: A Hierarchical Model for Publication Extraction from Academic Homepages
Yiqing Zhang, Jianzhong Qi, Rui Zhang and Chuandong Yin 1005

A Neural Transition-based Model for Nested Mention Recognition
Bailin Wang, Wei Lu, Yu Wang and Hongxia Jin 1011

Genre Separation Network with Adversarial Training for Cross-genre Relation Extraction
Ge Shi, Chong Feng, Lifu Huang, Boliang Zhang, Heng Ji, Lejian Liao and Heyan Huang . . . 1018

Effective Use of Context in Noisy Entity Linking
David Mueller and Greg Durrett 1024

Exploiting Contextual Information via Dynamic Memory Network for Event Detection
Shaobo Liu, Rui Cheng, Xiaoming Yu and Xueqi Cheng 1030

Do explanations make VQA models more predictable to a human?
Arjun Chandrasekaran, Viraj Prabhu, Deshraj Yadav, Prithvijit Chattopadhyay and Devi Parikh 1036

Facts That Matter
Marco Ponza, Luciano Del Corro and Gerhard Weikum 1043

Entity Tracking Improves Cloze-style Reading Comprehension
Luong Hoang, Sam Wiseman and Alexander Rush 1049

Adversarial Domain Adaptation for Duplicate Question Detection
Darsh Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo and Preslav Nakov 1056

Translating a Math Word Problem to a Expression Tree
Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang and Xiaojiang Liu 1064

Semantic Linking in Convolutional Neural Networks for Answer Sentence Selection
Massimo Nicosia and Alessandro Moschitti 1070

A dataset and baselines for sequential open-domain question answering
Ahmed Elgohary, Chen Zhao and Jordan Boyd-Graber 1077

Improving the results of string kernels in sentiment analysis and Arabic dialect identification by adapting them to your test set
Radu Tudor Ionescu and Andrei M. Butnaru 1084

Parameterized Convolutional Neural Networks for Aspect Level Sentiment Classification
Binxuan Huang and Kathleen Carley 1091

Improving Multi-label Emotion Classification via Sentiment Classification with Dual Attention Transfer Network
Jianfei Yu, Luis Marujo, Jing Jiang, Pradeep Karuturi and William Brendel 1097

Learning Sentiment Memories for Sentiment Modification without Parallel Data
Yi Zhang, Jingjing Xu, Pengcheng Yang and Xu Sun 1103

Joint Aspect and Polarity Classification for Aspect-based Sentiment Analysis with End-to-End Neural Networks
Martin Schmitt, Simon Steinheber, Konrad Schreiber and Benjamin Roth 1109

Representing Social Media Users for Sarcasm Detection
Y. Alex Kolchinski and Christopher Potts 1115

Syntactical Analysis of the Weaknesses of Sentiment Analyzers
Rohil Verma, Samuel Kim and David Walter 1122

Is Nike female? Exploring the role of sound symbolism in predicting brand name gender
Sridhar Moorthy, Ruth Pogacar, Samin Khan and Yang Xu 1128

Improving Large-Scale Fact-Checking using Decomposable Attention Models and Lexical Tagging
Nayeon Lee, Chien-Sheng Wu and Pascale Fung 1133

Harnessing Popularity in Social Media for Extractive Summarization of Online Conversations
Ryuji Kano, Yasuhide Miura, Motoki Taniguchi, Yan-Ying Chen, Francine Chen and Tomoko Ohkuma 1139

Identifying Locus of Control in Social Media Language
Masoud Rouhizadeh, Kokil Jaidka, Laura Smith, H. Andrew Schwartz, Anneke Buffone and Lyle Ungar 1146

Somm: Into the Model
Shengli Hu 1153

Fine-Grained Emotion Detection in Health-Related Online Posts
Hamed Khanpour and Cornelia Caragea 1160

The Remarkable Benefit of User-Level Aggregation for Lexical-based Population-Level Predictions
Salvatore Giorgi, Daniel Preoţiuc-Pietro, Anneke Buffone, Daniel Rieman, Lyle Ungar and H. Andrew Schwartz 1167

Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement
Jason Lee, Elman Mansimov and Kyunghyun Cho 1173

Large Margin Neural Language Model
Jiaji Huang, Yi Li, Wei Ping and Liang Huang 1183

Targeted Syntactic Evaluation of Language Models
Rebecca Marvin and Tal Linzen 1192

Rational Recurrences
Hao Peng, Roy Schwartz, Sam Thomson and Noah A. Smith 1203

Efficient Contextualized Representation: Language Model Pruning for Sequence Labeling
Liyuan Liu, Xiang Ren, Jingbo Shang, Xiaotao Gu, Jian Peng and Jiawei Han 1215

Automatic Event Salience Identification
Zhengzhong Liu, Chenyan Xiong, Teruko Mitamura and Eduard Hovy 1226

Temporal Information Extraction by Predicting Relative Time-lines
Artuur Leeuwenberg and Marie-Francine Moens 1237

Jointly Multiple Events Extraction via Attention-based Graph Information Aggregation
Xiao Liu, Zhunchen Luo and Heyan Huang 1247

RESIDE: Improving Distantly-Supervised Neural Relation Extraction using Side Information
Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya and Partha Talukdar 1257

Collective Event Detection via a Hierarchical and Bias Tagging Networks with Gated Multi-level Attention Mechanisms
Yubo Chen, Hang Yang, Kang Liu, Jun Zhao and Yantao Jia 1267

Valency-Augmented Dependency Parsing
Tianze Shi and Lillian Lee 1277

Unsupervised Learning of Syntactic Structure with Invertible Neural Projections
Junxian He, Graham Neubig and Taylor Berg-Kirkpatrick 1292

Dynamic Oracles for Top-Down and In-Order Shift-Reduce Constituent Parsing
Daniel Fernández-González and Carlos Gómez-Rodríguez 1303

Constituent Parsing as Sequence Labeling
Carlos Gómez-Rodríguez and David Vilares 1314

Synthetic Data Made to Order: The Case of Parsing
Dingquan Wang and Jason Eisner 1325

Tell-and-Answer: Towards Explainable Visual Question Answering using Attributes and Captions
Qing Li, Jianlong Fu, Dongfei Yu, Tao Mei and Jiebo Luo 1338

Learning a Policy for Opportunistic Active Learning
Aishwarya Padmakumar, Peter Stone and Raymond Mooney 1347

RecipeQA: A Challenge Dataset for Multimodal Comprehension of Cooking Recipes
Semih Yagcioglu, Aykut Erdem, Erkut Erdem and Nazli Ikizler-Cinbis 1358

TVQA: Localized, Compositional Video Question Answering
Jie Lei, Licheng Yu, Mohit Bansal and Tamara Berg 1369

Localizing Moments in Video with Temporal Language
Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell and Bryan Russell 1380

Card-660: Cambridge Rare Word Dataset - A Reliable Evaluation Framework for Rare Word Representation Models
Mohammad Taher Pilehvar, Dimitri Kartsaklis, Victor Prokhorov and Nigel Collier 1391

Leveraging Gloss Knowledge in Neural Word Sense Disambiguation by Hierarchical Co-Attention
Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui and Baobao Chang 1402

Weeding out Conventionalized Metaphors: A Corpus of Novel Metaphor Annotations
Erik-Lân Do Dinh, Hannah Wieland and Iryna Gurevych 1412

Streaming word similarity mining on the cheap
Olof Görnerup and Daniel Gillblad 1425

Memory, Show the Way: Memory Based Few Shot Word Representation Learning
Jingyuan Sun, Shaonan Wang and Chengqing Zong 1435

Disambiguated skip-gram model
Karol Grzegorczyk and Marcin Kurdziel 1445

Picking Apart Story Salads
Su Wang, Eric Holgate, Greg Durrett and Katrin Erk 1455

Dynamic Meta-Embeddings for Improved Sentence Representations
Douwe Kiela, Changhan Wang and Kyunghyun Cho 1466

A Probabilistic Model for Joint Learning of Word Embeddings from Texts and Images
Melissa Ailem, Bowen Zhang, Aurélien Bellet, Pascal Denis and Fei Sha 1478

Transfer and Multi-Task Learning for Noun–Noun Compound Interpretation
Murhaf Fares, Stephan Oepen and Erik Velldal 1488

Dissecting Contextual Word Embeddings: Architecture and Representation
Matthew Peters, Mark Neumann, Luke Zettlemoyer and Wen-tau Yih 1499

Preposition Sense Disambiguation and Representation
Hongyu Gong, Jiaqi Mu, Suma Bhat and Pramod Viswanath 1510

Auto-Encoding Dictionary Definitions into Consistent Word Embeddings
Tom Bosc and Pascal Vincent 1522

Spot the Odd Man Out: Exploring the Associative Power of Lexical Resources
Gabriel Stanovsky and Mark Hopkins 1533

Neural Multitask Learning for Simile Recognition
Lizhen Liu, Xiao Hu, Wei Song, Ruiji Fu, Ting Liu and Guoping Hu 1543

Structured Alignment Networks for Matching Sentences
Yang Liu, Matt Gardner and Mirella Lapata 1554

Compare, Compress and Propagate: Enhancing Neural Architectures with Alignment Factorization for Natural Language Inference
Yi Tay, Anh Tuan Luu and Siu Cheung Hui 1565

Convolutional Interaction Network for Natural Language Inference
Jingjing Gong, Xipeng Qiu, Xinchi Chen, Dong Liang and Xuanjing Huang 1576

Lessons from Natural Language Inference in the Clinical Domain
Alexey Romanov and Chaitanya Shivade 1586

Question Generation from SQL Queries Improves Neural Semantic Parsing
Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen and Ming Zhou 1597

SemRegex: A Semantics-Based Approach for Generating Regular Expressions from Natural Language Specifications
Zexuan Zhong, Jiaqi Guo, Wei Yang, Jian Peng, Tao Xie, Jian-Guang Lou, Ting Liu and Dongmei Zhang 1608

Decoupling Structure and Lexicon for Zero-Shot Semantic Parsing
Jonathan Herzig and Jonathan Berant 1619

A Span Selection Model for Semantic Role Labeling
Hiroki Ouchi, Hiroyuki Shindo and Yuji Matsumoto 1630

Mapping Language to Code in Programmatic Context
Srinivasan Iyer, Ioannis Konstas, Alvin Cheung and Luke Zettlemoyer 1643

SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task
Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li and Dragomir Radev 1653

Cross-lingual Decompositional Semantic Parsing
Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh and Benjamin Van Durme 1664

Learning to Learn Semantic Parsers from Natural Language Supervision
Igor Labutov, Bishan Yang and Tom Mitchell 1676

DeepCx: A transition-based approach for shallow semantic parsing with complex constructional triggers
Jesse Dunietz, Jaime Carbonell and Lori Levin 1691

What It Takes to Achieve 100% Condition Accuracy on WikiSQL
Semih Yavuz, Izzeddin Gur, Yu Su and Xifeng Yan 1702

Better Transition-Based AMR Parsing with a Refined Search Space
Zhijiang Guo and Wei Lu 1712

Heuristically Informed Unsupervised Idiom Usage Recognition
Changsheng Liu and Rebecca Hwa 1723

Coming to Your Senses: on Controls and Evaluation Sets in Polysemy Research
Haim Dubossarsky, Eitan Grossman and Daphna Weinshall 1732

Predicting Semantic Relations using Global Graph Properties
Yuval Pinter and Jacob Eisenstein 1741

Learning Scalar Adjective Intensity from Paraphrases
Anne Cocos, Veronica Wharton, Ellie Pavlick, Marianna Apidianaki and Chris Callison-Burch1752

Pointwise HSIC: A Linear-Time Kernelized Co-occurrence Norm for Sparse Linguistic Expressions
Sho Yokoi, Sosuke Kobayashi, Kenji Fukumizu, Jun Suzuki and Kentaro Inui 1763

Neural Related Work Summarization with a Joint Context-driven Attention Mechanism
Yongzhen Wang, Xiaozhong Liu and Zheng Gao 1776

Improving Neural Abstractive Document Summarization with Explicit Information Selection Modeling
Wei Li, Xinyan Xiao, Yajuan Lyu and Yuanzhuo Wang 1787

Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization
Shashi Narayan, Shay B. Cohen and Mirella Lapata 1797

Improving Abstraction in Text Summarization
Wojciech Kryściński, Romain Paulus, Caiming Xiong and Richard Socher 1808

Content Selection in Deep Learning Models of Summarization
Chris Kedzie, Kathleen McKeown and Hal Daume III 1818

Improved Semantic-Aware Network Embedding with Fine-Grained Word Alignment
Dinghan Shen, Xinyuan Zhang, Ricardo Henao and Lawrence Carin 1829

Learning Context-Sensitive Convolutional Filters for Text Processing
Dinghan Shen, Martin Renqiang Min, Yitong Li and Lawrence Carin 1839

Deep Relevance Ranking Using Enhanced Document-Query Interactions
Ryan McDonald, George Brokos and Ion Androutsopoulos 1849

Learning Neural Representation for CLIR with Adversarial Framework
Bo Li and Ping Cheng 1861

AD3: Attentive Deep Document Dater
Swayambhu Nath Ray, Shib Sankar Dasgupta and Partha Talukdar 1871

Gromov-Wasserstein Alignment of Word Embedding Spaces
David Alvarez-Melis and Tommi Jaakkola 1881

Deep Probabilistic Logic: A Unifying Framework for Indirect Supervision
Hai Wang and Hoifung Poon 1891

Deriving Machine Attention from Human Rationales
Yujia Bao, Shiyu Chang, Mo Yu and Regina Barzilay 1903

Semi-Supervised Sequence Modeling with Cross-View Training
Kevin Clark, Minh-Thang Luong, Christopher D. Manning and Quoc Le 1914

A Probabilistic Annotation Model for Crowdsourcing Coreference
Silviu Paun, Jon Chamberlain, Udo Kruschwitz, Juntao Yu and Massimo Poesio 1926

A Deterministic Algorithm for Bridging Anaphora Resolution
Yufang Hou 1938

A Knowledge Hunting Framework for Common Sense Reasoning
Ali Emami, Noelia De La Cruz, Adam Trischler, Kaheer Suleman and Jackie Chi Kit Cheung 1949

Mapping Text to Knowledge Graph Entities using Multi-Sense LSTMs
Dimitri Kartsaklis, Mohammad Taher Pilehvar and Nigel Collier 1959

Differentiating Concepts and Instances for Knowledge Graph Embedding
Xin Lv, Lei Hou, Juanzi Li and Zhiyuan Liu 1971

One-Shot Relational Learning for Knowledge Graphs
Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo and William Yang Wang 1980

Regular Expression Guided Entity Mention Mining from Noisy Web Data
Shanshan Zhang, Lihong He, Slobodan Vucetic and Eduard Dragut 1991

HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding
Shib Sankar Dasgupta, Swayambhu Nath Ray and Partha Talukdar 2001

Neural Adaptation Layers for Cross-domain Named Entity Recognition
Bill Yuchen Lin and Wei Lu 2012

Entity Linking within a Social Media Platform: A Case Study on Yelp
Hongliang Dai, Yangqiu Song, Liwei Qiu and Rijia Liu 2023

Annotation of a Large Clinical Entity Corpus
Pinal Patel, Disha Davey, Vishal Panchal and Parth Pathak 2033

Visual Supervision in Bootstrapped Information Extraction
Matthew Berger, Ajay Nagesh, Joshua Levine, Mihai Surdeanu and Helen Zhang 2043

Learning Named Entity Tagger using Domain-Specific Dictionary
Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren and Jiawei Han 2054

Zero-Shot Open Entity Typing as Type-Compatible Grounding
Ben Zhou, Daniel Khashabi, Chen-Tse Tsai and Dan Roth 2065

Attention-Guided Answer Distillation for Machine Reading Comprehension
Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang and Ming Zhou2077

Interpretation of Natural Language Rules in Conversational Machine Reading
Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard and Sebastian Riedel 2087

A State-transition Framework to Answer Complex Questions over Knowledge Base
Sen Hu, Lei Zou and Xinbo Zhang 2098

A Multi-answer Multi-task Framework for Real-world Machine Reading Comprehension
Jiahua Liu, Wan Wei, Maosong Sun, Hao Chen, Yantao Du and Dekang Lin 2109

Logician and Orator: Learning from the Duality between Language and Knowledge in Open Domain
Mingming Sun, Xu Li and Ping Li 2119

MemoReader: Large-Scale Reading Comprehension through Neural Memory Controller
Seohyun Back, Seunghak Yu, Sathish Reddy Indurthi, Jihie Kim and Jaegul Choo 2131

Multi-Granular Sequence Encoding via Dilated Compositional Units for Reading Comprehension
Yi Tay, Anh Tuan Luu and Siu Cheung Hui 2141

Neural Compositional Denotational Semantics for Question Answering
Nitish Gupta and Mike Lewis 2152

Cross-Pair Text Representations for Answer Sentence Selection
Kateryna Tymoshenko and Alessandro Moschitti 2162

QuAC: Question Answering in Context
Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang and Luke Zettlemoyer 2174

Knowledge Base Question Answering via Encoding of Complex Query Graphs
Kangqi Luo, Fengli Lin, Xusheng Luo and Kenny Zhu 2185

Neural Relation Extraction via Inner-Sentence Noise Reduction and Transfer Learning
Tianyi Liu, Xinsong Zhang, Wanhao Zhou and Weijia Jia 2195

Graph Convolution over Pruned Dependency Trees Improves Relation Extraction
Yuhao Zhang, Peng Qi and Christopher D. Manning 2205

Multi-Level Structured Self-Attentions for Distantly Supervised Relation Extraction
Jinhua Du, Jingguang Han, Andy Way and Dadong Wan 2216

N-ary Relation Extraction using Graph-State LSTM
Linfeng Song, Yue Zhang, Zhiguo Wang and Daniel Gildea 2226

Hierarchical Relation Extraction with Coarse-to-Fine Grained Attention
Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun and Peng Li 2236

Label-Free Distant Supervision for Relation Extraction via Knowledge Graph Embedding
Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu and Huajun Chen 2246

Extracting Entities and Relations with Joint Minimum Risk Training
Changzhi Sun, Yuanbin Wu, Man Lan, Shiliang Sun, Wenting Wang, Kuang-Chih Lee and Kewen Wu 2256

Large-scale Exploration of Neural Relation Classification Architectures
Hoang-Quynh Le, Duy-Cat Can, Sinh T. Vu, Thanh Hai Dang, Mohammad Taher Pilehvar and Nigel Collier 2266

Possessors Change Over Time: A Case Study with Artworks
Dhivya Chinnappa and Eduardo Blanco 2278

Using Lexical Alignment and Referring Ability to Address Data Sparsity in Situated Dialog Reference Resolution
Todd Shore and Gabriel Skantze 2288

Subgoal Discovery for Hierarchical Dialogue Policy Learning
Da Tang, Xiujun Li, Jianfeng Gao, Chong Wang, Lihong Li and Tony Jebara 2298

Supervised Clustering of Questions into Intents for Dialog System Applications
Iryna Haponchyk, Antonio Uva, Seunghak Yu, Olga Uryupina and Alessandro Moschitti 2310

Towards Exploiting Background Knowledge for Building Conversation Systems
Nikita Moghe, Siddhartha Arora, Suman Banerjee and Mitesh M. Khapra 2322

Decoupling Strategy and Generation in Negotiation Dialogues
He He, Derek Chen, Anusha Balakrishnan and Percy Liang 2333

Large-scale Cloze Test Dataset Created by Teachers
Qizhe Xie, Guokun Lai, Zihang Dai and Eduard Hovy 2344

emrQA: A Large Corpus for Question Answering on Electronic Medical Records
Anusri Pampari, Preethi Raghavan, Jennifer Liang and Jian Peng 2357

HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering
Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov and Christopher D. Manning 2369

Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering
Todor Mihaylov, Peter Clark, Tushar Khot and Ashish Sabharwal 2381

Evaluating Theory of Mind in Question Answering
Aida Nematzadeh, Kaylee Burns, Erin Grant, Alison Gopnik and Tom Griffiths 2392

A Unified Syntax-aware Framework for Semantic Role Labeling
Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li and Luo Si 2401

Semantics as a Foreign Language
Gabriel Stanovsky and Ido Dagan 2412

An AMR Aligner Tuned by Transition-based Parser
Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin and Ting Liu 2422

Dependency-based Hybrid Trees for Semantic Parsing
Zhanming Jie and Wei Lu 2431

Policy Shaping and Generalized Update Equations for Semantic Parsing from Denotations
Dipendra Misra, Ming-Wei Chang, Xiaodong He and Wen-tau Yih 2442

Sentence Compression for Arbitrary Languages via Multilingual Pivoting
Jonathan Mallinson, Rico Sennrich and Mirella Lapata 2453

Unsupervised Cross-lingual Transfer of Word Embedding Spaces
Ruochen Xu, Yiming Yang, Naoki Otani and Yuexin Wu 2465

XNLI: Evaluating Cross-lingual Sentence Representations
Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk and Veselin Stoyanov 2475

Joint Multilingual Supervision for Cross-lingual Entity Linking
Shyam Upadhyay, Nitish Gupta and Dan Roth 2486

Fine-grained Coordinated Cross-lingual Text Stream Alignment for Endless Language Knowledge Acquisition
Tao Ge, Qing Dou, Heng Ji, Lei Cui, Baobao Chang, Zhifang Sui, Furu Wei and Ming Zhou . 2496

WECA:A WordNet-Encoded Collocation-Attention Network for Homographic Pun Recognition
Yufeng Diao, Hongfei Lin, Di Wu, Liang Yang, Kan Xu, Zhihao Yang, Jian Wang, Shaowu Zhang, Bo Xu and Dongyu Zhang 2507

A Hybrid Approach to Automatic Corpus Generation for Chinese Spelling Check
Dingmin Wang, Yan Song, Jing Li, Jialong Han and Haisong Zhang 2517

Neural Quality Estimation of Grammatical Error Correction
Shamil Chollampatt and Hwee Tou Ng 2528

Transferring from Formal Newswire Domain with Hypernet for Twitter POS Tagging
Tao Gui, Qi Zhang, Jingjing Gong, Minlong Peng, di liang, Keyu Ding and Xuanjing Huang . 2540

Free as in Free Word Order: An Energy Based Model for Word Segmentation and Morphological Tagging in Sanskrit
Amrith Krishna, Bishal Santra, Sasi Prasanth Bandaru, Gaurav Sahu, Vishnu Dutt Sharma, Pavankumar Satuluri and Pawan Goyal 2550

A Challenge Set and Methods for Noun-Verb Ambiguity
Ali Elkahky, Kellie Webster, Daniel Andor and Emily Pitler 2562

What do character-level models learn about morphology? The case of dependency parsing
Clara Vania, Andreas Grivas and Adam Lopez 2573

Learning Better Internal Structure of Words for Sequence Labeling
Yingwei Xin, Ethan Hart, Vibhuti Mahajan and Jean David Ruvini 2584

ICON: Interactive Conversational Memory Network for Multimodal Emotion Detection
Devamanyu Hazarika, Soujanya Poria, Rada Mihalcea, Erik Cambria and Roger Zimmermann2594

Discriminative Learning of Open-Vocabulary Object Retrieval and Localization by Negative Phrase Augmentation
Ryota Hinami and Shin'ichi Satoh 2605

Grounding Semantic Roles in Images
Carina Silberer and Manfred Pinkal 2616

Commonsense Justification for Action Explanation
Shaohua Yang, Qiaozi Gao, Sari Sadiya and Joyce Chai 2627

Learning Personas from Dialogue with Attentive Memory Networks
Eric Chu, Prashanth Vijayaraghavan and Deb Roy 2638

Grounding language acquisition by training semantic parsers using captioned videos
Candace Ross, Andrei Barbu, Yevgeni Berzak, Battushig Myanganbayar and Boris Katz 2647

Translating Navigation Instructions in Natural Language to a High-Level Plan for Behavioral Robot Navigation
Xiaoxue Zang, Ashwini Pokle, Marynel Vázquez, Kevin Chen, Juan Carlos Niebles, Alvaro Soto and Silvio Savarese 2657

Mapping Instructions to Actions in 3D Environments with Visual Goal Prediction
Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin and Yoav Artzi 2667

Deconvolutional Time Series Regression: A Technique for Modeling Temporally Diffuse Effects
Cory Shain and William Schuler 2679

Is this Sentence Difficult? Do you Agree?
Dominique Brunato, Lorenzo De Mattei, Felice Dell'Orletta, Benedetta Iavarone and Giulia Venturi 2690

Neural Transition Based Parsing of Web Queries: An Entity Based Approach
Rivka Malca and Roi Reichart 2700

An Investigation of the Interactions Between Pre-Trained Word Embeddings, Character Models and POS Tags in Dependency Parsing
Aaron Smith, Miryam de Lhoneux, Sara Stymne and Joakim Nivre 2711

Depth-bounding is effective: Improvements and evaluation of unsupervised PCFG induction
Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler and Lane Schwartz 2721

Incremental Computation of Infix Probabilities for Probabilistic Finite Automata
Marco Cognetta, Yo-Sub Han and Soon Chan Kwon 2732

Syntax Encoding with Application in Authorship Attribution
Richong Zhang, Zhiyuan Hu, Hongyu Guo and Yongyi Mao 2742

Sanskrit Word Segmentation Using Character-level Recurrent and Convolutional Neural Networks
Oliver Hellwig and Sebastian Nehrdich 2754

Session-level Language Modeling for Conversational Speech
Wayne Xiong, Lingfeng Wu, Jun Zhang and Andreas Stolcke 2764

Towards Less Generic Responses in Neural Conversation Models: A Statistical Re-weighting Method
Yahui Liu, Wei Bi, Jun Gao, Xiaojiang Liu, Jian Yao and Shuming Shi 2769

Training Millions of Personalized Dialogue Agents
Pierre-Emmanuel Mazare, Samuel Humeau, Martin Raison and Antoine Bordes 2775

Towards Universal Dialogue State Tracking
Liliang Ren, Kaige Xie, Lu Chen and Kai Yu 2780

Semantic Parsing for Task Oriented Dialog using Hierarchical Representations
Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar and Mike Lewis 2787

The glass ceiling in NLP
Natalie Schluter 2793

Reducing Gender Bias in Abusive Language Detection
Ji Ho Park, Jamin Shin and Pascale Fung 2799

SafeCity: Understanding Diverse Forms of Sexual Harassment Personal Stories
Sweta Karlekar and Mohit Bansal 2805

Learning multiview embeddings for assessing dementia
Chloé Pou-Prom and Frank Rudzicz 2812

WikiConv: A Corpus of the Complete Conversational History of a Large Online Collaborative Community
Yiqing Hua, Cristian Danescu-Niculescu-Mizil, Dario Taraborelli, Nithum Thain, Jeffery Sorensen and Lucas Dixon 2818

Marginal Likelihood Training of BiLSTM-CRF for Biomedical Named Entity Recognition from Disjoint Label Sets
Nathan Greenberg, Trapit Bansal, Patrick Verga and Andrew McCallum 2824

Adversarial training for multi-context joint entity and relation extraction
Giannis Bekoulis, Johannes Deleu, Thomas Demeester and Chris Develder 2830

Structured Multi-Label Biomedical Text Tagging via Attentive Neural Tree Decoding
Gaurav Singh, James Thomas, Iain Marshall, John Shawe-Taylor and Byron C. Wallace 2837

Deep Exhaustive Model for Nested Named Entity Recognition
Mohammad Golam Sohrab and Makoto Miwa 2843

Evaluating the Utility of Hand-crafted Features in Sequence Labelling
Minghao Wu, Fei Liu and Trevor Cohn 2850

Improved Dependency Parsing using Implicit Word Connections Learned from Unlabeled Data
Wenhui Wang, Baobao Chang and Mairgup Mansur 2857

A Framework for Understanding the Role of Morphology in Universal Dependency Parsing
Mathieu Dehouck and Pascal Denis 2864

The Lazy Encoder: A Fine-Grained Analysis of the Role of Morphology in Neural Machine Translation
Arianna Bisazza and Clara Tump 2871

Imitation Learning for Neural Morphological String Transduction
Peter Makarov and Simon Clematide 2877

An Encoder-Decoder Approach to the Paradigm Cell Filling Problem
Miikka Silfverberg and Mans Hulden 2883

Generating Natural Language Adversarial Examples
Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava and Kai-Wei Chang 2890

Multi-Head Attention with Disagreement Regularization
Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu and Tong Zhang 2897

Deep Bayesian Active Learning for Natural Language Processing: Results of a Large-Scale Empirical Study
Aditya Siddhant and Zachary C. Lipton 2904

Bayesian Compression for Natural Language Processing
Nadezhda Chirkova, Ekaterina Lobacheva and Dmitry Vetrov 2910

Multimodal neural pronunciation modeling for spoken languages with logographic origin
Minh Nguyen, Gia H Ngo and Nancy Chen 2916

Chinese Pinyin Aided IME, Input What You Have Not Keystroked Yet
Yafang Huang and Hai Zhao 2923

Estimating Marginal Probabilities of n-grams for Recurrent Neural Language Models
Thanapon Noraset, Doug Downey and Lidong Bing 2930

How to represent a word and predict it, too: Improving tied architectures for language modelling
Kristina Gulordava, Laura Aina and Gemma Boleda 2936

The Importance of Generation Order in Language Modeling
Nicolas Ford, Daniel Duckworth, Mohammad Norouzi and George Dahl 2942

Document-Level Neural Machine Translation with Hierarchical Attention Networks
Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas and James Henderson 2947

Three Strategies to Improve One-to-Many Multilingual Translation
Yining Wang, Jiajun Zhang, Feifei Zhai, Jingfang Xu and Chengqing Zong 2955

Multi-Source Syntactic Neural Machine Translation
Anna Currey and Kenneth Heafield 2961

Fixing Translation Divergences in Parallel Corpora for Neural MT
Minh Quang Pham, Josep Crego, Jean Senellart and François Yvon 2967

Adversarial Evaluation of Multimodal Machine Translation
Desmond Elliott 2974

Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion
Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou and Edouard Grave 2979

Learning When to Concentrate or Divert Attention: Self-Adaptive Attention Temperature for Neural Machine Translation
Junyang Lin, Xu Sun, Xuancheng Ren, Muyu Li and Qi Su 2985

Accelerating Asynchronous Stochastic Gradient Descent for Neural Machine Translation
Nikolay Bogoychev, Kenneth Heafield, Alham Fikri Aji and Marcin Junczys-Dowmunt 2991

Learning to Jointly Translate and Predict Dropped Pronouns with a Shared Reconstruction Mechanism
Longyue Wang, Zhaopeng Tu, Andy Way and Qun Liu 2997

Getting Gender Right in Neural Machine Translation
Eva Vanmassenhove, Christian Hardmeier and Andy Way 3003

Towards Two-Dimensional Sequence to Sequence Model in Neural Machine Translation
Parnia Bahar, Christopher Brix and Hermann Ney 3009

End-to-End Non-Autoregressive Neural Machine Translation with Connectionist Temporal Classification
Jindřich Libovický and Jindřich Helcl 3016

Prediction Improves Simultaneous Neural Machine Translation
Ashkan Alinejad, Maryam Siahbani and Anoop Sarkar 3022

Training Deeper Neural Machine Translation Models with Transparent Attention
Ankur Bapna, Mia Chen, Orhan Firat, Yuan Cao and Yonghui Wu 3028

Context and Copying in Neural Machine Translation
Rebecca Knowles and Philipp Koehn 3034

Encoding Gated Translation Memory into Neural Machine Translation
Qian Cao and Deyi Xiong 3042

Automatic Post-Editing of Machine Translation: A Neural Programmer-Interpreter Approach
Thuy-Trang Vu and Gholamreza Haffari 3048

Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation
Yilin Yang, Liang Huang and Mingbo Ma 3054

Multi-Multi-View Learning: Multilingual and Multi-Representation Entity Typing
Yadollah Yaghoobzadeh and Hinrich Schütze 3060

Word Embeddings for Code-Mixed Language Processing
Adithya Pratapa, Monojit Choudhury and Sunayana Sitaram 3067

On the Strength of Character Language Models for Multilingual Named Entity Recognition
Xiaodong Yu, Stephen Mayhew, Mark Sammons and Dan Roth 3073

Code-switched Language Models Using Dual RNNs and Same-Source Pretraining
Saurabh Garg, Tanmay Parekh and Preethi Jyothi 3078

Part-of-Speech Tagging for Code-Switched, Transliterated Texts without Explicit Language Identification
Kelsey Ball and Dan Garrette 3084

Zero-shot User Intent Detection via Capsule Neural Networks
Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang and Philip Yu 3090

Hierarchical Neural Networks for Sequential Sentence Classification in Medical Scientific Abstracts
Di Jin and Peter Szolovits 3100

Investigating Capsule Networks with Dynamic Routing for Text Classification
Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao and Soufei Zhang 3110

Topic Memory Networks for Short Text Classification
Jichuan Zeng, Jing Li, Yan Song, Cuiyun Gao, Michael R. Lyu and Irwin King 3120

Few-Shot and Zero-Shot Multi-Label Learning for Structured Label Spaces
Anthony Rios and Ramakanth Kavuluru 3132

Automatic Poetry Generation with Mutual Reinforcement Learning
Xiaoyuan Yi, Maosong Sun, Ruoyu Li and Wenhao Li 3143

Variational Autoregressive Decoder for Neural Response Generation
Jiachen Du, Wenjie Li, Yulan He, Ruifeng Xu, Lidong Bing and Xuan Wang 3154

Integrating Transformer and Paraphrase Rules for Sentence Simplification
Sanqiang Zhao, Rui Meng, Daqing He, Andi Saptono and Bambang Parmanto 3164

Learning Neural Templates for Text Generation
Sam Wiseman, Stuart Shieber and Alexander Rush 3174

Multi-Reference Training with Pseudo-References for Neural Translation and Text Generation
Renjie Zheng, Mingbo Ma and Liang Huang 3188

Knowledge Graph Embedding with Hierarchical Relation Structure
Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin and Qing He 3198

Embedding Multimodal Relational Data for Knowledge Base Completion
Pouya Pezeshkpour, Liyan Chen and Sameer Singh 3208

Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction
Yi Luan, Luheng He, Mari Ostendorf and Hannaneh Hajishirzi 3219

Playing 20 Question Game with Policy-Based Reinforcement Learning
Huang Hu, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, wei wu and Zhan Chen . . 3233

Multi-Hop Knowledge Graph Reasoning with Reward Shaping
Xi Victoria Lin, Richard Socher and Caiming Xiong 3243

Neural Transductive Learning and Beyond: Morphological Generation in the Minimal-Resource Setting
Katharina Kann and Hinrich Schütze 3254

Implicational Universals in Stochastic Constraint-Based Phonology
Giorgio Magri 3265

Explaining Character-Aware Neural Networks for Word-Level Prediction: Do They Discover Linguistic Rules?
Fréderic Godin, Kris Demuynck, Joni Dambre, Wesley De Neve and Thomas Demeester 3275

Adapting Word Embeddings to New Languages with Morphological and Phonological Subword Representations
Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R. Mortensen and Jaime Carbonell 3285

A Computational Exploration of Exaggeration
Enrica Troiano, Carlo Strapparava, Gözde Özbal and Serra Sinem Tekiroglu 3296

Building Context-aware Clause Representations for Situation Entity Type Classification
Zeyu Dai and Ruihong Huang 3305

Hierarchical Dirichlet Gaussian Marked Hawkes Process for Narrative Reconstruction in Continuous Time Domain
Yeon Seonwoo, Alice Oh and Sungjoon Park 3316

Investigating the Role of Argumentation in the Rhetorical Analysis of Scientific Publications with Neural Multi-Task Learning Models
Anne Lauscher, Goran Glavaš, Simone Paolo Ponzetto and Kai Eckert 3326

Neural Ranking Models for Temporal Dependency Structure Parsing
Yuchen Zhang and Nianwen Xue 3339

Causal Explanation Analysis on Social Media
Youngseo Son, Nipun Bayas and H. Andrew Schwartz 3350

LRMM: Learning to Recommend with Missing Modalities
Cheng Wang, Mathias Niepert and Hui Li 3360

Content Explorer: Recommending Novel Entities for a Document Writer
Michal Lukasik and Richard Zens 3371

A Genre-Aware Attention Model to Improve the Likability Prediction of Books
Suraj Maharjan, Manuel Montes, Fabio A. González and Thamar Solorio 3381

Thread Popularity Prediction and Tracking with a Permutation-invariant Model
Hou Pong Chan and Irwin King 3392

IARM: Inter-Aspect Relation Modeling with Memory Networks in Aspect-Based Sentiment Analysis
Navonil Majumder, Soujanya Poria, Alexander Gelbukh, Md Shad Akhtar, Erik Cambria and Asif Ekbal 3402

Limbic: Author-Based Sentiment Aspect Modeling Regularized with Word Embeddings and Discourse Relations
Zhe Zhang and Munindar Singh 3412

An Interpretable Neural Network with Topical Information for Relevant Emotion Ranking
Yang Yang, Deyu ZHOU and Yulan He 3423

Multi-grained Attention Network for Aspect-Level Sentiment Classification
Feifan Fan, Yansong Feng and Dongyan Zhao 3433

Attentive Gated Lexicon Reader with Contrastive Contextual Co-Attention for Sentiment Classification
Yi Tay, Anh Tuan Luu, Siu Cheung Hui and Jian Su 3443

Contextual Inter-modal Attention for Multi-modal Sentiment Analysis
Deepanway Ghosal, Md Shad Akhtar, Dushyant Chauhan, Soujanya Poria, Asif Ekbal and Pushpak Bhattacharyya 3454

Adaptive Semi-supervised Learning for Cross-domain Sentiment Classification
Ruidan He, Wee Sun Lee, Hwee Tou Ng and Daniel Dahlmeier 3467

ExtRA: Extracting Prominent Review Aspects from Customer Feedback
Zhiyi Luo, Shanshan Huang, Frank F. Xu, Bill Yuchen Lin, Hanyuan Shi and Kenny Zhu 3477

Cross-Lingual Cross-Platform Rumor Verification Pivoting on Multimedia Content
Weiming Wen, Songwen Su and Zhou Yu 3487

Extractive Adversarial Networks: High-Recall Explanations for Identifying Personal Attacks in Social Media Posts
Samuel Carton, Qiaozhu Mei and Paul Resnick 3497

Automatic Detection of Vague Words and Sentences in Privacy Policies
Logan Lebanoff and Fei Liu 3508

Multi-view Models for Political Ideology Detection of News Articles
Vivek Kulkarni, Junting Ye, Steve Skiena and William Yang Wang 3518

Predicting Factuality of Reporting and Bias of News Media Sources
Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass and Preslav Nakov 3528

Legal Judgment Prediction via Topological Learning
Haoxi Zhong, Guo Zhipeng, Cunchao Tu, Chaojun Xiao, Zhiyuan Liu and Maosong Sun 3540

Hierarchical CVAE for Fine-Grained Hate Speech Classification
Jing Qian, Mai ElSherief, Elizabeth Belding and William Yang Wang 3550

Residualized Factor Adaptation for Community Social Media Prediction Tasks
Mohammadzaman Zamani, H. Andrew Schwartz, Veronica Lynn, Salvatore Giorgi and Niranjan Balasubramanian 3560

Framing and Agenda-setting in Russian News: a Computational Analysis of Intricate Political Strategies
Anjalie Field, Doron Kliger, Shuly Wintner, Jennifer Pan, Dan Jurafsky and Yulia Tsvetkov . . 3570

Identifying the sentiment styles of YouTube's vloggers
Bennett Kleinberg, Maximilian Mozes and Isabelle van der Vegt 3581

Native Language Identification with User Generated Content
Gili Goldin, Ella Rabinovich and Shuly Wintner 3591

Beyond Error Propagation in Neural Machine Translation: Characteristics of Language Also Matter
Lijun Wu, Xu Tan, Di He, Fei Tian, Tao Qin, Jianhuang Lai and Tie-Yan Liu 3602

A Study of Reinforcement Learning for Neural Machine Translation
Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai and Tie-Yan Liu 3612

Meta-Learning for Low-Resource Neural Machine Translation
Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li and Kyunghyun Cho 3622

Unsupervised Statistical Machine Translation
Mikel Artetxe, Gorka Labaka and Eneko Agirre 3632

A Visual Attention Grounding Neural Model for Multimodal Machine Translation
Mingyang Zhou, Runxiang Cheng, Yong Jae Lee and Zhou Yu 3643

Sentiment Classification towards Question-Answering with Hierarchical Matching Network
Chenlin Shen, Changlong Sun, Jingjing Wang, Yangyang Kang, Shoushan Li, Xiaozhong Liu, Luo Si, Min Zhang and Guodong Zhou 3654

Cross-topic Argument Mining from Heterogeneous Sources
Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai and Iryna Gurevych 3664

Summarizing Opinions: Aspect Extraction Meets Sentiment Prediction and They Are Both Weakly Supervised
Stefanos Angelidis and Mirella Lapata 3675

CARER: Contextualized Affect Representations for Emotion Recognition
Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu and Yi-Shin Chen 3687

Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency
Zhuang Ma and Michael Collins 3698

CaLcs: Continuously Approximating Longest Common Subsequence for Sequence Level Optimization
Semih Yavuz, Chung-Cheng Chiu, Patrick Nguyen and Yonghui Wu 3708

Pathologies of Neural Models Make Interpretations Difficult
Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez and Jordan Boyd-Graber 3719

Phrase-level Self-Attention Networks for Universal Sentence Encoding
Wei Wu, Houfeng Wang, Tianyu Liu and Shuming Ma 3729

BanditSum: Extractive Summarization as a Contextual Bandit
Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof and Jackie Chi Kit Cheung 3739

A Word-Complexity Lexicon and A Neural Readability Ranking Model for Lexical Simplification
Mounica Maddela and Wei Xu 3749

Learning Latent Semantic Annotations for Grounding Natural Language to Structured Data
Guanghui Qin, Jin-Ge Yao, Xuening Wang, Jinpeng Wang and Chin-Yew Lin 3761

Syntactic Scaffolds for Semantic Structures
Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer and Noah A. Smith 3772

Hierarchical Quantized Representations for Script Generation
Noah Weber, Leena Shekhar, Niranjan Balasubramanian and Nate Chambers 3783

Semantic Role Labeling for Learner Chinese: the Importance of Syntactic Parsing and L2-L1 Parallel Data
Zi Lin, Yuguang Duan, Yuanyuan Zhao, Weiwei Sun and Xiaojun Wan 3793

A Teacher-Student Framework for Maintainable Dialog Manager
Weikang Wang, Jiajun Zhang, Han Zhang, Mei-Yuh Hwang, Chengqing Zong and Zhifei Li . 3803

Discriminative Deep Dyna-Q: Robust Planning for Dialogue Policy Learning
Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu and Yun-Nung Chen 3813

A Self-Attentive Model with Gate Mechanism for Spoken Language Understanding
Changliang Li, Liang Li and Ji Qi 3824

Learning End-to-End Goal-Oriented Dialog with Multiple Answers
Janarthanan Rajendran, Jatin Ganhotra, Satinder Singh and Lazaros Polymenakos 3834

AirDialogue: An Environment for Goal-Oriented Dialogue Research
Wei Wei, Quoc Le, Andrew Dai and Jia Li 3844

QuaSE: Sequence Editing under Quantifiable Guidance
Yi Liao, Lidong Bing, Piji Li, Shuming Shi, Wai Lam and Tong Zhang 3855

Paraphrase Generation with Deep Reinforcement Learning
Zichao Li, Xin Jiang, Lifeng Shang and Hang Li 3865

Operation-guided Neural Networks for High Fidelity Data-To-Text Generation
Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan and Chin-Yew Lin 3879

Generating Classical Chinese Poems via Conditional Variational Autoencoder and Adversarial Training
Juntao Li, Yan Song, Haisong Zhang, Dongmin Chen, Shuming Shi, Dongyan Zhao and Rui Yan 3890

Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks
Yao Zhao, Xiaochuan Ni, Yuanyuan Ding and Qifa Ke 3901

Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task
Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang and Dragomir Radev 3911

Unsupervised Natural Language Generation with Denoising Autoencoders
Markus Freitag and Scott Roy 3922

Answer-focused and Position-aware Neural Question Generation
Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma and Shi Wang 3930

Diversity-Promoting GAN: A Cross-Entropy Based Generative Adversarial Network for Diversified Text Generation
Jingjing Xu, Xuancheng Ren, Junyang Lin and Xu Sun 3940

Towards a Better Metric for Evaluating Question Generation Systems
Preksha Nema and Mitesh M. Khapra 3950

Stylistic Chinese Poetry Generation via Unsupervised Style Disentanglement
Cheng Yang, Maosong Sun, Xiaoyuan Yi and Wenhao Li 3960

Generating More Interesting Responses in Neural Conversation Models with Distributional Constraints
Ashutosh Baheti, Alan Ritter, Jiwei Li and Bill Dolan 3970

Better Conversations by Modeling, Filtering, and Optimizing for Coherence and Diversity
Xinnuo Xu, Ondřej Dušek, Ioannis Konstas and Verena Rieser 3981

Incorporating Background Knowledge into Video Description Generation
Spencer Whitehead, Heng Ji, Mohit Bansal, Shih-Fu Chang and Clare Voss 3992

Multimodal Differential Network for Visual Question Generation
Badri Narayana Patro, Sandeep Kumar, Vinod Kumar Kurmi and Vinay Namboodiri 4002

Entity-aware Image Caption Generation
Di Lu, Spencer Whitehead, Lifu Huang, Heng Ji and Shih-Fu Chang 4013

Learning to Describe Differences Between Pairs of Similar Images
Harsh Jhamtani and Taylor Berg-Kirkpatrick 4024

Object Hallucination in Image Captioning
Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell and Kate Saenko 4035

Abstractive Text-Image Summarization Using Multi-Modal Attentional Hierarchical RNN
Jingqiang Chen and Hai Zhuge 4046

Keyphrase Generation with Correlation Constraints
Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan and Zhoujun Li 4057

Closed-Book Training to Improve Summarization Encoder Memory
Yichen Jiang and Mohit Bansal 4067

Improving Neural Abstractive Document Summarization with Structural Regularization
Wei Li, Xinyan Xiao, Yajuan Lyu and Yuanzhuo Wang 4078

Iterative Document Representation Learning Towards Summarization with Polishing
Xiuying Chen, Shen Gao, Chongyang Tao, Yan Song, Dongyan Zhao and Rui Yan 4088

Bottom-Up Abstractive Summarization
Sebastian Gehrmann, Yuntian Deng and Alexander Rush 4098

Controlling Length in Abstractive Summarization Using a Convolutional Neural Network
Yizhu Liu, Zhiyi Luo and Kenny Zhu 4110

APRIL: Interactively Learning to Summarise by Combining Active Preference Learning and Reinforcement Learning
Yang Gao, Christian M. Meyer and Iryna Gurevych 4120

Adapting the Neural Encoder-Decoder Framework from Single to Multi-Document Summarization
Logan Lebanoff, Kaiqiang Song and Fei Liu 4131

Semi-Supervised Learning for Neural Keyphrase Generation
Hai Ye and Lu Wang 4142

MSMO: Multimodal Summarization with Multimodal Output
Junnan Zhu, Haoran Li, Tianshang Liu, Yu Zhou, Jiajun Zhang and Chengqing Zong 4154

Frustratingly Easy Model Ensemble for Abstractive Summarization
Hayato Kobayashi 4165

Automatic Pyramid Evaluation Exploiting EDU-based Extractive Reference Summaries
Tsutomu Hirao, Hidetaka Kamigaito and Masaaki Nagata 4177

Learning to Encode Text as Human-Readable Summaries using Generative Adversarial Networks
Yaushian Wang and Hung-yi Lee 4187

Joint Multitask Learning for Community Question Answering Using Task-Specific Embeddings
Shafiq Joty, Lluís Màrquez and Preslav Nakov 4196

What Makes Reading Comprehension Questions Easier?
Saku Sugawara, Kentaro Inui, Satoshi Sekine and Akiko Aizawa 4208

Commonsense for Generative Multi-Hop Question Answering Tasks
Lisa Bauer, Yicheng Wang and Mohit Bansal 4220

Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text
Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov and William Cohen 4231

A Nil-Aware Answer Extraction Framework for Question Answering
Souvik Kundu and Hwee Tou Ng 4243

Exploiting Deep Representations for Neural Machine Translation
Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi and Tong Zhang 4253

Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures
Gongbo Tang, Mathias Müller, Annette Rios and Rico Sennrich 4263

Simplifying Neural Machine Translation with Addition-Subtraction Twin-Gated Recurrent Networks
Biao Zhang, Deyi Xiong, jinsong su, Qian Lin and Huiji Zhang 4273

Speeding Up Neural Machine Translation Decoding by Cube Pruning
Wen Zhang, Liang Huang, Yang Feng, Lei Shen and Qun Liu 4284

Revisiting Character-Based Neural Machine Translation with Capacity and Compression
Colin Cherry, George Foster, Ankur Bapna, Orhan Firat and Wolfgang Macherey 4295

A Skeleton-Based Model for Promoting Coherence Among Sentences in Narrative Story Generation
Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai and Xu Sun 4306

NEXUS Network: Connecting the Preceding and the Following in Dialogue Generation
Xiaoyu Shen, Hui Su, Wenjie Li and Dietrich Klakow 4316

A Neural Local Coherence Model for Text Quality Assessment
Mohsen Mesgar and Michael Strube 4328

Deep Attentive Sentence Ordering Network
Baiyun Cui, Yingming Li, Ming Chen and Zhongfei Zhang 4340

Getting to "Hearer-old": Charting Referring Expressions Across Time
Ieva Staliūnaitė, Hannah Rohde, Bonnie Webber and Annie Louis 4350

Making "fetch" happen: The influence of social and linguistic context on nonstandard word growth and decline
Ian Stewart and Jacob Eisenstein 4360

Analyzing Correlated Evolution of Multiple Features Using Latent Representations
Yugo Murawaki 4371

Capturing Regional Variation with Distributed Place Representations and Geographic Retrofitting
Dirk Hovy and Christoph Purschke 4383

Characterizing Interactions and Relationships between People
Farzana Rashid and Eduardo Blanco 4395

Why Swear? Analyzing and Inferring the Intentions of Vulgar Expressions
Eric Holgate, Isabel Cachola, Daniel Preoţiuc-Pietro and Junyi Jessy Li 4405

Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP tasks
Steffen Eger, Paul Youssef and Iryna Gurevych 4415

Hard Non-Monotonic Attention for Character-Level Transduction
Shijie Wu, Pamela Shapiro and Ryan Cotterell 4425

Speed Reading: Learning to Read ForBackward via Shuttle
Tsu-Jui Fu and Wei-Yun Ma 4439

Modeling Localness for Self-Attention Networks
Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao and Tong Zhang 4449

Chargrid: Towards Understanding 2D Documents
Anoop R Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne and Jean Baptiste Faddoul 4459

Simple Recurrent Units for Highly Parallelizable Recurrence
Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai and Yoav Artzi 4470

NPRF: A Neural Pseudo Relevance Feedback Framework for Ad-hoc Information Retrieval
Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun and Jungang Xu . . 4482

Co-Stack Residual Affinity Networks with Multi-level Attention Refinement for Matching Text Sequences
Yi Tay, Anh Tuan Luu and Siu Cheung Hui 4492

Spherical Latent Spaces for Stable Variational Autoencoders
Jiacheng Xu and Greg Durrett 4503

Learning Universal Sentence Representations with Mean-Max Attention Autoencoder
Minghua Zhang, Yunfang Wu, Weikang Li and Wei Li 4514

Word Mover's Embedding: From Word2Vec to Document Embedding
Lingfei Wu, Ian En-Hsu Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar and Michael J. Witbrock 4524

Multilingual Clustering of Streaming News
Sebastião Miranda, Arturs Znotins, Shay B. Cohen and Guntis Barzdins 4535

Multi-Task Label Embedding for Text Classification
Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang and Yaohui Jin 4545

Semantic-Unit-Based Dilated Convolution for Multi-Label Text Classification
Junyang Lin, Qi Su, Pengcheng Yang, Shuming Ma and Xu Sun 4554

MCapsNet: Capsule Network for Text with Multi-Task Learning
Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang and Yaohui Jin 4565

Uncertainty-aware generative models for inferring document class prevalence
Katherine Keith and Brendan O'Connor 4575

Challenges of Using Text Classifiers for Causal Inference
Zach Wood-Doughty, Ilya Shpitser and Mark Dredze 4586

Direct Output Connection for a High-Rank Language Model
Sho Takase, Jun Suzuki and Masaaki Nagata 4599

Disfluency Detection using Auto-Correlational Neural Networks
Paria Jamshid Lou, Peter Anderson and Mark Johnson 4610

Pyramidal Recurrent Unit for Language Modeling
Sachin Mehta, Rik Koncel-Kedziorski, Mohammad Rastegari and Hannaneh Hajishirzi 4620

On Tree-Based Neural Sentence Modeling
Haoyue Shi, Hao Zhou, Jiaze Chen and Lei Li 4631

Language Modeling with Sparse Product of Sememe Experts
Yihong Gu, Jun Yan, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin and Leyu Lin 4642

Siamese Network-Based Supervised Topic Modeling
Minghui Huang, Yanghui Rao, Yuwei Liu, Haoran Xie and Fu Lee Wang 4652

GraphBTM: Graph Enhanced Autoencoded Variational Inference for Biterm Topic Model
Qile Zhu, Zheng Feng and Xiaolin Li 4663

Modeling Online Discourse with Coupled Distributed Topics
Akshay Srivatsan, Zachary Wojtowicz and Taylor Berg-Kirkpatrick 4673

Learning Disentangled Representations of Texts with Application to Biomedical Abstracts
Sarthak Jain, Edward Banner, Jan-Willem van de Meent, Iain J Marshall and Byron C. Wallace 4683

Multi-Source Domain Adaptation with Mixture of Experts
Jiang Guo, Darsh Shah and Regina Barzilay 4694

A Neural Model of Adaptation in Reading
Marten van Schijndel and Tal Linzen 4704

Understanding Deep Learning Performance through an Examination of Test Set Difficulty: A Psychometric Case Study
John Lalor, Hao Wu, Tsendsuren Munkhdalai and Hong Yu 4711

Lexicosyntactic Inference in Neural Models
Aaron Steven White, Rachel Rudinger, Kyle Rawlins and Benjamin Van Durme 4717

Dual Fixed-Size Ordinally Forgetting Encoding (FOFE) for Competitive Neural Language Models
Sedtawut Watcharawittayakul, Mingbin Xu and Hui Jiang 4725

The Importance of Being Recurrent for Modeling Hierarchical Structure
Ke Tran, Arianna Bisazza and Christof Monz 4731

Joint Learning for Targeted Sentiment Analysis
Dehong Ma, Sujian Li and Houfeng Wang 4737

Revisiting the Importance of Encoding Logic Rules in Sentiment Classification
Kalpesh Krishna, Preethi Jyothi and Mohit Iyyer 4743

A Co-Attention Neural Network Model for Emotion Cause Analysis with Emotional Context Awareness
Xiangju Li, Kaisong Song, Shi Feng, Daling Wang and Yifei Zhang 4752

Modeling Empathy and Distress in Reaction to News Stories
Sven Buechel, Anneke Buffone, Barry Slaff, Lyle Ungar and Joao Sedoc 4758

Interpretable Emoji Prediction via Label-Wise Attention LSTMs
Francesco Barbieri, Luis Espinosa Anke, Jose Camacho-Collados, Steven Schockaert and Horacio Saggion 4766

A Tree-based Decoder for Neural Machine Translation
Xinyi Wang, Hieu Pham, Pengcheng Yin and Graham Neubig 4772

Greedy Search with Probabilistic N-gram Matching for Neural Machine Translation
Chenze Shao, Xilin Chen and Yang Feng 4778

Exploring Recombination for Efficient Decoding of Neural Machine Translation
Zhisong Zhang, Rui Wang, Masao Utiyama, Eiichiro Sumita and Hai Zhao 4785

Has Machine Translation Achieved Human Parity? A Case for Document-level Evaluation
Samuel Läubli, Rico Sennrich and Martin Volk 4791

Automatic Reference-Based Evaluation of Pronoun Translation Misses the Point
Liane Guillou and Christian Hardmeier 4797

FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation
Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu and Maosong Sun 4803

A strong baseline for question relevancy ranking
Ana Gonzalez, Isabelle Augenstein and Anders Søgaard 4810

Learning Sequence Encoders for Temporal Knowledge Graph Completion
Alberto Garcia-Duran, Sebastijan Dumančić and Mathias Niepert 4816

Similar but not the Same: Word Sense Disambiguation Improves Event Detection via Neural Representation Matching
Weiyi Lu and Thien Huu Nguyen 4822

Learning Word Representations with Cross-Sentence Dependency for End-to-End Co-reference Resolution
Hongyin Luo and Jim Glass 4829

Word Relation Autoencoder for Unseen Hypernym Extraction Using Word Embeddings
Hong-You Chen, Cheng-Syuan Lee, Keng-Te Liao and Shou-de Lin 4834

Refining Pretrained Word Embeddings Using Layer-wise Relevance Propagation
Akira Utsumi 4840

Learning Gender-Neutral Word Embeddings
Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang and Kai-Wei Chang 4847

Learning Concept Abstractness Using Weak Supervision
Ella Rabinovich, Benjamin Sznajder, Artem Spector, Ilya Shnayderman, Ranit Aharonov, David Konopnicki and Noam Slonim 4854

Word Sense Induction with Neural biLM and Symmetric Patterns
Asaf Amrami and Yoav Goldberg 4860

InferLite: Simple Universal Sentence Representations from Natural Language Inference Data
Jamie Kiros and William Chan 4868

Similarity-Based Reconstruction Loss for Meaning Representation
Olga Kovaleva, Anna Rumshisky and Alexey Romanov 4875

What can we learn from Semantic Tagging?
Mostafa Abdou, Artur Kulmizev, Vinit Ravishankar, Lasha Abzianidze and Johan Bos 4881

Conditional Word Embedding and Hypothesis Testing via Bayes-by-Backprop
Rujun Han, Michael Gill, Arthur Spirling and Kyunghyun Cho 4890

Classifying Referential and Non-referential It Using Gaze
Victoria Yaneva, Le An Ha, Richard Evans and Ruslan Mitkov 4896

State-of-the-art Chinese Word Segmentation with Bi-LSTMs
Ji Ma, Kuzman Ganchev and David Weiss 4902

Sanskrit Sandhi Splitting using seq2(seq)2
Rahul Aralikatte, Neelamadhav Gantayat, Naveen Panwar, Anush Sankaran and Senthil Mani 4909

Unsupervised Neural Word Segmentation for Chinese via Segmental Language Modeling
Zhiqing Sun and Zhi-Hong Deng 4915

LemmaTag: Jointly Tagging and Lemmatizing for Morphologically Rich Languages with BRNNs
Daniel Kondratyuk, Tomáš Gavenčiak, Milan Straka and Jan Hajič 4921

Recovering Missing Characters in Old Hawaiian Writing
Brendan Shillingford and Oiwi Parker Jones 4929

When data permutations are pathological: the case of neural natural language inference
Natalie Schluter and Daniel Varab 4935

Bridging Knowledge Gaps in Neural Entailment via Symbolic Models
Dongyeop Kang, Tushar Khot, Ashish Sabharwal and Peter Clark 4940

The BQ Corpus: A Large-scale Domain-specific Chinese Corpus For Sentence Semantic Equivalence Identification
Jing Chen, Qingcai Chen, Xin Liu, Haijun Yang, Daohe Lu and Buzhou Tang 4946

Interpreting Recurrent and Attention-Based Neural Models: a Case Study on Natural Language Inference
Reza Ghaeini, Xiaoli Fern and Prasad Tadepalli 4952

Towards Semi-Supervised Learning for Deep Semantic Role Labeling
Sanket Vaibhav Mehta, Jay Yoon Lee and Jaime Carbonell 4958

Identifying Domain Adjacent Instances for Semantic Parsers
James Ferguson, Janara Christensen, Edward Li and Edgar Gonzàlez 4964

Mapping natural language commands to web elements
Panupong Pasupat, Tian-Shun Jiang, Evan Liu, Kelvin Guu and Percy Liang 4970

Wronging a Right: Generating Better Errors to Improve Grammatical Error Detection
Sudhanshu Kasewa, Pontus Stenetorp and Sebastian Riedel 4977

Modeling Input Uncertainty in Neural Network Dependency Parsing
Rob van der Goot and Gertjan van Noord 4984

Parameter sharing between dependency parsers for related languages
Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein and Anders Søgaard 4992

Grammar Induction with Neural Language Models: An Unusual Replication
Phu Mon Htut, Kyunghyun Cho and Samuel Bowman 4998

Data Augmentation via Dependency Tree Morphing for Low-Resource Languages
Gozde Gul Sahin and Mark Steedman 5004

How Much Reading Does Reading Comprehension Require? A Critical Investigation of Popular Benchmarks
Divyansh Kaushik and Zachary C. Lipton 5010

MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling
Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan and Milica Gasic 5016

Linguistically-Informed Self-Attention for Semantic Role Labeling
Emma Strubell, Patrick Verga, Daniel Andor, David Weiss and Andrew McCallum 5027

Phrase-Based & Neural Unsupervised Machine Translation
Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer and Marc'Aurelio Ranzato . 5039

Conference Program

Friday, November 2, 2018

09:00–09:30 ***Opening remarks (Gold Hall)***

09:30–10:30 ***Keynote I: Julia Hirschberg "Truth or Lie? Spoken Indicators of Deception in Speech" (Gold Hall)***

10:30–11:00 ***Coffee Break***

11:00–12:30 ***Long Papers and Demos (Orals and Posters) I***

Session 1A: Social Applications I (Gold Hall)

11:00–11:18 *Privacy-preserving Neural Representations of Text*
Maximin Coavoux, Shashi Narayan and Shay B. Cohen

11:18–11:36 *Adversarial Removal of Demographic Attributes from Text Data*
Yanai Elazar and Yoav Goldberg

11:36–11:54 *DeClarE: Debunking Fake News and False Claims using Evidence-Aware Deep Learning*
Kashyap Popat, Subhabrata Mukherjee, Andrew Yates and Gerhard Weikum

11:54–12:12 *It's going to be okay: Measuring Access to Support in Online Communities*
Zijian Wang and David Jurgens

12:12–12:30 *Detecting Gang-Involved Escalation on Social Media Using Context*
Serina Chang, Ruiqi Zhong, Ethan Adams, Fei-Tzin Lee, Siddharth Varia, Desmond Patton, William Frey, Chris Kedzie and Kathy McKeown

Friday, November 2, 2018 (continued)

Session 1B: Semantics I (Copper Hall)

11:00–11:18 *Reasoning about Actions and State Changes by Injecting Commonsense Knowledge*
Niket Tandon, Bhavana Dalvi, Joel Grus, Wen-tau Yih, Antoine Bosselut and Peter Clark

11:18–11:36 *Collecting Diverse Natural Language Inference Problems for Sentence Representation Evaluation*
Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White and Benjamin Van Durme

11:36–11:54 *Textual Analogy Parsing: What's Shared and What's Compared among Analogous Facts*
Matthew Lamm, Arun Chaganty, Christopher D. Manning, Dan Jurafsky and Percy Liang

11:54–12:12 *SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference*
Rowan Zellers, Yonatan Bisk, Roy Schwartz and Yejin Choi

12:12–12:30 *TwoWingOS: A Two-Wing Optimization Strategy for Evidential Claim Verification*
Wenpeng Yin and Dan Roth

Session 1C: Vision (Silver Hall)

11:00–11:18 *Associative Multichannel Autoencoder for Multimodal Word Representation*
Shaonan Wang, Jiajun Zhang and Chengqing Zong

11:18–11:36 *Game-Based Video-Context Dialogue*
Ramakanth Pasunuru and Mohit Bansal

11:36–11:54 *simNet: Stepwise Image-Topic Merging Network for Generating Detailed and Comprehensive Image Captions*
Fenglin Liu, Xuancheng Ren, Yuanxin Liu, Houfeng Wang and Xu Sun

11:54–12:12 *Multimodal Language Analysis with Recurrent Multistage Fusion*
Paul Pu Liang, Ziyin Liu, AmirAli Bagher Zadeh and Louis-Philippe Morency

12:12–12:30 *Temporally Grounding Natural Sentence in Video*
Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie and Tat-Seng Chua

Session 1D: Entities and Coreference (Hall 100)

11:00–11:18 *PreCo: A Large-scale Dataset in Preschool Vocabulary for Coreference Resolution*
Hong Chen, Zhenhua Fan, Hao Lu, Alan Yuille and Shu Rong

11:18–11:36 *Adversarial Transfer Learning for Chinese Named Entity Recognition with Self-Attention Mechanism*
Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao and Shengping Liu

11:36–11:54 *Using Linguistic Features to Improve the Generalization Capability of Neural Coreference Resolvers*
Nafise Sadat Moosavi and Michael Strube

11:54–12:12 *Neural Segmental Hypergraphs for Overlapping Mention Recognition*
Bailin Wang and Wei Lu

12:12–12:30 *Variational Sequential Labelers for Semi-Supervised Learning*
Mingda Chen, Qingming Tang, Karen Livescu and Kevin Gimpel

Session 1E: Machine Translation and Multilingual Methods (Posters and Demos, Grand Hall 2)

Joint Representation Learning of Cross-lingual Words and Entities via Attentive Distant Supervision
Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Chengjiang Li, Xu Chen and Tiansi Dong

Deep Pivot-Based Modeling for Cross-language Cross-domain Transfer with Minimal Guidance
Yftah Ziser and Roi Reichart

Multi-lingual Common Semantic Space Construction via Cluster-consistent Word Embedding
Lifu Huang, Kyunghyun Cho, Boliang Zhang, Heng Ji and Kevin Knight

Unsupervised Multilingual Word Embeddings
Xilun Chen and Claire Cardie

CLUSE: Cross-Lingual Unsupervised Sense Embeddings
Ta Chung Chi and Yun-Nung Chen

Adversarial Propagation and Zero-Shot Cross-Lingual Transfer of Word Vector Specialization
Edoardo Maria Ponti, Ivan Vulić, Goran Glavaš, Nikola Mrkšić and Anna Korhonen

Improving Cross-Lingual Word Embeddings by Meeting in the Middle
Yerai Doval, Jose Camacho-Collados, Luis Espinosa Anke and Steven Schockaert

WikiAtomicEdits: A Multilingual Corpus of Wikipedia Edits for Modeling Language and Discourse
Manaal Faruqui, Ellie Pavlick, Ian Tenney and Dipanjan Das

On the Relation between Linguistic Typology and (Limitations of) Multilingual Language Modeling
Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart and Anna Korhonen

A Fast, Compact, Accurate Model for Language Identification of Codemixed Text
Yuan Zhang, Jason Riesa, Daniel Gillick, Anton Bakalov, Jason Baldridge and David Weiss

Personalized Microblog Sentiment Classification via Adversarial Cross-lingual Multi-task Learning
Weichao Wang, Shi Feng, Wei Gao, Daling Wang and Yifei Zhang

Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks
Zhichun Wang, Qingsong Lv, Xiaohan Lan and Yu Zhang

Cross-lingual Lexical Sememe Prediction
Fanchao Qi, Yankai Lin, Maosong Sun, Hao Zhu, Ruobing Xie and Zhiyuan Liu

Neural Cross-Lingual Named Entity Recognition with Minimal Resources
Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith and Jaime Carbonell

A Stable and Effective Learning Strategy for Trainable Greedy Decoding
Yun Chen, Victor O.K. Li, Kyunghyun Cho and Samuel Bowman

Addressing Troublesome Words in Neural Machine Translation
Yang Zhao, Jiajun Zhang, Zhongjun He, Chengqing Zong and Hua Wu

Top-down Tree Structured Decoding with Syntactic Connections for Neural Machine Translation and Parsing
Jetic Gū, Hassan S. Shavarani and Anoop Sarkar

XL-NBT: A Cross-lingual Neural Belief Tracking Framework
Wenhu Chen, Jianshu Chen, Yu Su, Xin Wang, Dong Yu, Xifeng Yan and William Yang Wang

Contextual Parameter Generation for Universal Neural Machine Translation
Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig and Tom Mitchell

Back-Translation Sampling by Targeting Difficult Words in Neural Machine Translation
Marzieh Fadaee and Christof Monz

Multi-Domain Neural Machine Translation with Word-Level Domain Context Discrimination
Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin and Jianqiang Zhao

A Discriminative Latent-Variable Model for Bilingual Lexicon Induction
Sebastian Ruder, Ryan Cotterell, Yova Kementchedjhieva and Anders Søgaard

Non-Adversarial Unsupervised Word Translation
Yedid Hoshen and Lior Wolf

Semi-Autoregressive Neural Machine Translation
Chunqi Wang, Ji Zhang and Haiqing Chen

Understanding Back-Translation at Scale
Sergey Edunov, Myle Ott, Michael Auli and David Grangier

Bootstrapping Transliteration with Constrained Discovery for Low-Resource Languages
Shyam Upadhyay, Jordan Kodner and Dan Roth

NORMA: Neighborhood Sensitive Maps for Multilingual Word Embeddings
Ndapa Nakashole

Adaptive Multi-pass Decoder for Neural Machine Translation
Xinwei Geng, Xiaocheng Feng, Bing Qin and Ting Liu

Improving the Transformer Translation Model with Document-Level Context
Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang and Yang Liu

MTNT: A Testbed for Machine Translation of Noisy Text
Paul Michel and Graham Neubig

Demo: CytonMT: an Efficient Neural Machine Translation Open-source Toolkit Implemented in C++
Xiaolin Wang, Masao Utiyama, and Eiichiro Sumita

Demo: SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing
Taku Kudo and John Richardson

12:30–13:45 ***Lunch***

13:45–14:45 ***Short Papers (Orals and Posters) I***

Session 2A: Question Answering I (Gold Hall)

13:45–13:57 *SimpleQuestions Nearly Solved: A New Upperbound and Baseline Approach*
Michael Petrochuk and Luke Zettlemoyer

13:57–14:09 *Phrase-Indexed Question Answering: A New Challenge for Scalable Document Comprehension*
Minjoon Seo, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi and Hannaneh Hajishirzi

14:09–14:21 *Ranking Paragraphs for Improving Answer Recall in Open-Domain Question Answering*
Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko and Jaewoo Kang

14:21–14:33 *Cut to the Chase: A Context Zoom-in Network for Reading Comprehension*
Sathish Reddy Indurthi, Seunghak Yu, Seohyun Back and Heriberto Cuayahuitl

14:33–14:45 *Adaptive Document Retrieval for Deep Question Answering*
Bernhard Kratzwald and Stefan Feuerriegel

Session 2B: Semantics II (Copper Hall)

13:45–13:57 *Why is unsupervised alignment of English embeddings from different algorithms so hard?*
Mareike Hartmann, Yova Kementchedjhieva and Anders Søgaard

13:57–14:09 *Quantifying Context Overlap for Training Word Embeddings*
Yimeng Zhuang, Jinghui Xie, Yinhe Zheng and Xuan Zhu

14:09–14:21 *Neural Latent Relational Analysis to Capture Lexical Semantic Relations in a Vector Space*
Koki Washio and Tsuneaki Kato

14:21–14:33 *Generalizing Word Embeddings using Bag of Subwords*
Jinman Zhao, Sidharth Mudgal and Yingyu Liang

14:33–14:45 *Neural Metaphor Detection in Context*
Ge Gao, Eunsol Choi, Yejin Choi and Luke Zettlemoyer

Session 2C: Multilingual Methods I (Silver Hall)

13:45–13:57 *Distant Supervision from Disparate Sources for Low-Resource Part-of-Speech Tagging*
Barbara Plank and Željko Agić

13:57–14:09 *Unsupervised Bilingual Lexicon Induction via Latent Variable Models*
Zi-Yi Dou, Zhi-Hao Zhou and Shujian Huang

14:09–14:21 *Learning Unsupervised Word Translations Without Adversaries*
Tanmoy Mukherjee, Makoto Yamada and Timothy Hospedales

14:21–14:33 *Adversarial Training for Multi-task and Multi-lingual Joint Modeling of Utterance Intent Classification*
Ryo Masumura, Yusuke Shinohara, Ryuichiro Higashinaka and Yushi Aono

14:33–14:45 *Surprisingly Easy Hard-Attention for Sequence to Sequence Learning*
Shiv Shankar, Siddhant Garg and Sunita Sarawagi

Session 2D: Social Media (Hall 100)

13:45–13:57 *Joint Learning for Emotion Classification and Emotion Cause Detection*
Ying Chen, Wenjun Hou, Xiyao Cheng and Shoushan Li

13:57–14:09 *Exploring Optimism and Pessimism in Twitter Using Deep Learning*
Cornelia Caragea, Liviu P. Dinu and Bogdan Dumitru

14:09–14:21 *Predicting News Headline Popularity with Syntactic and Semantic Knowledge Using Multi-Task Learning*
Sotiris Lamprinidis, Daniel Hardt and Dirk Hovy

14:21–14:33 *Hybrid Neural Attention for Agreement/Disagreement Inference in Online Debates*
Di Chen, Jiachen Du, Lidong Bing and Ruifeng Xu

14:33–14:45 *Increasing In-Class Similarity by Retrofitting Embeddings with Demographic Information*
Dirk Hovy and Tommaso Fornaciari

Session 2E: Short Posters I (Grand Hall 2)

A Syntactically Constrained Bidirectional-Asynchronous Approach for Emotional Conversation Generation
Jingyuan Li and Xiao Sun

Auto-Dialabel: Labeling Dialogue Data with Unsupervised Learning
Chen Shi, Qi Chen, Lei Sha, Sujian Li, Xu Sun, Houfeng Wang and Lintao Zhang

Extending Neural Generative Conversational Model using External Knowledge Sources
Prasanna Parthasarathi and Joelle Pineau

Modeling Temporality of Human Intentions by Domain Adaptation
Xiaolei Huang, Lixing Liu, Kate Carey, Joshua Woolley, Stefan Scherer and Brian Borsari

An Auto-Encoder Matching Model for Learning Utterance-Level Semantic Dependency in Dialogue Generation
Liangchen Luo, Jingjing Xu, Junyang Lin, Qi Zeng and Xu Sun

A Dataset for Document Grounded Conversations
Kangyan Zhou, Shrimai Prabhumoye and Alan W Black

Out-of-domain Detection based on Generative Adversarial Network
Seonghan Ryu, Sangjun Koo, Hwanjo Yu and Gary Geunbae Lee

Listening Comprehension over Argumentative Content
Shachar Mirkin, Guy Moshkowich, Matan Orbach, Lili Kotlerman, Yoav Kantor, Tamar Lavee, Michal Jacovi, Yonatan Bilu, Ranit Aharonov and Noam Slonim

Using active learning to expand training data for implicit discourse relation recognition
Yang Xu, Yu Hong, Huibin Ruan, Jianmin Yao, Min Zhang and Guodong Zhou

Learning To Split and Rephrase From Wikipedia Edit History
Jan A. Botha, Manaal Faruqui, John Alex, Jason Baldridge and Dipanjan Das

BLEU is Not Suitable for the Evaluation of Text Simplification
Elior Sulem, Omri Abend and Ari Rappoport

S2SPMN: A Simple and Effective Framework for Response Generation with Relevant Information
Jiaxin Pei and Chenliang Li

Improving Reinforcement Learning Based Image Captioning with Natural Language Prior
Tszhang Guo, Shiyu Chang, Mo Yu and Kun Bai

Training for Diversity in Image Paragraph Captioning
Luke Melas-Kyriazi, Alexander Rush and George Han

A Graph-theoretic Summary Evaluation for ROUGE
Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong and Fang Chen

Guided Neural Language Generation for Abstractive Summarization using Abstract Meaning Representation
Hardy Hardy and Andreas Vlachos

Evaluating Multiple System Summary Lengths: A Case Study
Ori Shapira, David Gabay, Hadar Ronen, Judit Bar-Ilan, Yael Amsterdamer, Ani Nenkova and Ido Dagan

Friday, November 2, 2018 (continued)

Neural Latent Extractive Document Summarization
Xingxing Zhang, Mirella Lapata, Furu Wei and Ming Zhou

On the Abstractiveness of Neural Document Summarization
Fangfang Zhang, Jin-ge Yao and Rui Yan

Automatic Essay Scoring Incorporating Rating Schema via Reinforcement Learning
Yucheng Wang, Zhongyu Wei, Yaqian Zhou and Xuanjing Huang

Identifying Well-formed Natural Language Questions
Manaal Faruqui and Dipanjan Das

15:00–15:12 *Self-Governing Neural Networks for On-Device Short Text Classification*
Sujith Ravi and Zornitsa Kozareva

HFT-CNN: Learning Hierarchical Category Structure for Multi-label Short Text Categorization
Kazuya Shimura, Jiyi Li and Fumiyo Fukumoto

A Hierarchical Neural Attention-based Text Classifier
Koustuv Sinha, Yue Dong, Jackie Chi Kit Cheung and Derek Ruths

Labeled Anchors and a Scalable, Transparent, and Interactive Classifier
Jeffrey Lund, Stephen Cowley, Wilson Fearn, Emily Hales and Kevin Seppi

Coherence-Aware Neural Topic Modeling
Ran Ding, Ramesh Nallapati and Bing Xiang

Utilizing Character and Word Embeddings for Text Normalization with Sequence-to-Sequence Models
Daniel Watson, Nasser Zalmout and Nizar Habash

Topic Intrusion for Automatic Topic Model Evaluation
Shraey Bhatia, Jey Han Lau and Timothy Baldwin

Supervised and Unsupervised Methods for Robust Separation of Section Titles and Prose Text in Web Documents
Abhijith Athreya Mysore Gopinath, Shomir Wilson and Norman Sadeh

Friday, November 2, 2018 (continued)

14:45–15:00 ***Mini-Break***

15:00–16:00 ***Short Papers (Orals and Posters) II***

Session 3A: Machine Translation I (Gold Hall)

15:00–15:12 *SwitchOut: an Efficient Data Augmentation Algorithm for Neural Machine Translation*
Xinyi Wang, Hieu Pham, Zihang Dai and Graham Neubig

15:12–15:24 *Improving Unsupervised Word-by-Word Translation with Language Model and Denoising Autoencoder*
Yunsu Kim, Jiahui Geng and Hermann Ney

15:24–15:36 *Decipherment of Substitution Ciphers with Neural Language Models*
Nishant Kambhatla, Anahita Mansouri Bigvand and Anoop Sarkar

15:36–15:48 *Rapid Adaptation of Neural Machine Translation to New Languages*
Graham Neubig and Junjie Hu

15:48–16:00 *Compact Personalized Models for Neural Machine Translation*
Joern Wuebker, Patrick Simianer and John DeNero

Session 3B: Machine Learning I (Copper Hall)

15:00–15:12 *Self-Governing Neural Networks for On-Device Short Text Classification*
Sujith Ravi and Zornitsa Kozareva

15:12–15:24 *Supervised Domain Enablement Attention for Personalized Domain Classification*
Joo-Kyung Kim and Young-Bum Kim

15:24–15:36 *A Deep Neural Network Sentence Level Classification Method with Context Information*
Xingyi Song, Johann Petrak and Angus Roberts

15:36–15:48 *Towards Dynamic Computation Graphs via Sparse Latent Structure*
Vlad Niculae, André F. T. Martins and Claire Cardie

15:48–16:00 *Convolutional Neural Networks with Recurrent Neural Filters*
Yi Yang

Session 3C: Semantic Parsing / Generation (Silver Hall)

15:00–15:12 *Exploiting Rich Syntactic Information for Semantic Parsing with Graph-to-Sequence Model*
Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen and Vadim Sheinin

15:12–15:24 *Retrieval-Based Neural Code Generation*
Shirley Anugrah Hayati, Raphael Olivier, Pravalika Avvaru, Pengcheng Yin, Anthony Tomasic and Graham Neubig

15:24–15:36 *SQL-to-Text Generation with Graph-to-Sequence Model*
Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng and Vadim Sheinin

15:36–15:48 *Generating Syntactic Paraphrases*
Emilie Colin and Claire Gardent

15:48–16:00 *Neural-Davidsonian Semantic Proto-role Labeling*
Rachel Rudinger, Adam Teichert, Ryan Culkin, Sheng Zhang and Benjamin Van Durme

Session 3D: Vision / Discourse (Hall 100)

15:00–15:12 *Conversational Decision-Making Model for Predicting the King's Decision in the Annals of the Joseon Dynasty*
JinYeong Bak and Alice Oh

15:12–15:24 *Toward Fast and Accurate Neural Discourse Segmentation*
Yizhong Wang, Sujian Li and Jingfeng Yang

15:24–15:36 *A Dataset for Telling the Stories of Social Media Videos*
Spandana Gella, Mike Lewis and Marcus Rohrbach

15:36–15:48 *Cascaded Mutual Modulation for Visual Reasoning*
Yiqun Yao, Jiaming Xu, Feng Wang and Bo Xu

15:48–16:00 *How agents see things: On visual representations in an emergent language game*
Diane Bouchacourt and Marco Baroni

Session 3E: Short Posters II (Grand Hall 2)

Attention-Based Capsule Networks with Dynamic Routing for Relation Extraction
Ningyu Zhang, Shumin Deng, Zhanling Sun, Xi Chen, Wei Zhang and Huajun Chen

Put It Back: Entity Typing with Language Model Enhancement
Ji Xin, Hao Zhu, Xu Han, Zhiyuan Liu and Maosong Sun

Event Detection with Neural Networks: A Rigorous Empirical Evaluation
Walker Orr, Prasad Tadepalli and Xiaoli Fern

PubSE: A Hierarchical Model for Publication Extraction from Academic Homepages
Yiqing Zhang, Jianzhong Qi, Rui Zhang and Chuandong Yin

A Neural Transition-based Model for Nested Mention Recognition
Bailin Wang, Wei Lu, Yu Wang and Hongxia Jin

Genre Separation Network with Adversarial Training for Cross-genre Relation Extraction
Ge Shi, Chong Feng, Lifu Huang, Boliang Zhang, Heng Ji, Lejian Liao and Heyan Huang

Effective Use of Context in Noisy Entity Linking
David Mueller and Greg Durrett

Exploiting Contextual Information via Dynamic Memory Network for Event Detection
Shaobo Liu, Rui Cheng, Xiaoming Yu and Xueqi Cheng

Do explanations make VQA models more predictable to a human?
Arjun Chandrasekaran, Viraj Prabhu, Deshraj Yadav, Prithvijit Chattopadhyay and Devi Parikh

Facts That Matter
Marco Ponza, Luciano Del Corro and Gerhard Weikum

Entity Tracking Improves Cloze-style Reading Comprehension
Luong Hoang, Sam Wiseman and Alexander Rush

Adversarial Domain Adaptation for Duplicate Question Detection
Darsh Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo and Preslav Nakov

Translating a Math Word Problem to a Expression Tree
Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang and Xiaojiang Liu

Semantic Linking in Convolutional Neural Networks for Answer Sentence Selection
Massimo Nicosia and Alessandro Moschitti

A dataset and baselines for sequential open-domain question answering
Ahmed Elgohary, Chen Zhao and Jordan Boyd-Graber

Improving the results of string kernels in sentiment analysis and Arabic dialect identification by adapting them to your test set
Radu Tudor Ionescu and Andrei M. Butnaru

Parameterized Convolutional Neural Networks for Aspect Level Sentiment Classification
Binxuan Huang and Kathleen Carley

Improving Multi-label Emotion Classification via Sentiment Classification with Dual Attention Transfer Network
Jianfei Yu, Luis Marujo, Jing Jiang, Pradeep Karuturi and William Brendel

Learning Sentiment Memories for Sentiment Modification without Parallel Data
Yi Zhang, Jingjing Xu, Pengcheng Yang and Xu Sun

Joint Aspect and Polarity Classification for Aspect-based Sentiment Analysis with End-to-End Neural Networks
Martin Schmitt, Simon Steinheber, Konrad Schreiber and Benjamin Roth

Representing Social Media Users for Sarcasm Detection
Y. Alex Kolchinski and Christopher Potts

Syntactical Analysis of the Weaknesses of Sentiment Analyzers
Rohil Verma, Samuel Kim and David Walter

Is Nike female? Exploring the role of sound symbolism in predicting brand name gender
Sridhar Moorthy, Ruth Pogacar, Samin Khan and Yang Xu

Improving Large-Scale Fact-Checking using Decomposable Attention Models and Lexical Tagging
Nayeon Lee, Chien-Sheng Wu and Pascale Fung

Harnessing Popularity in Social Media for Extractive Summarization of Online Conversations
Ryuji Kano, Yasuhide Miura, Motoki Taniguchi, Yan-Ying Chen, Francine Chen and Tomoko Ohkuma

Identifying Locus of Control in Social Media Language
Masoud Rouhizadeh, Kokil Jaidka, Laura Smith, H. Andrew Schwartz, Anneke Buffone and Lyle Ungar

Somm: Into the Model
Shengli Hu

Fine-Grained Emotion Detection in Health-Related Online Posts
Hamed Khanpour and Cornelia Caragea

The Remarkable Benefit of User-Level Aggregation for Lexical-based Population-Level Predictions
Salvatore Giorgi, Daniel Preoţiuc-Pietro, Anneke Buffone, Daniel Rieman, Lyle Ungar and H. Andrew Schwartz

16:00–16:30 ***Coffee Break***

16:30–18:00 ***Long Papers and Demos (Orals and Posters) II***

Session 4A: Language Models (Gold Hall)

16:30–16:48 *Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement*
Jason Lee, Elman Mansimov and Kyunghyun Cho

16:48–17:06 *Large Margin Neural Language Model*
Jiaji Huang, Yi Li, Wei Ping and Liang Huang

17:06–17:24 *Targeted Syntactic Evaluation of Language Models*
Rebecca Marvin and Tal Linzen

17:24–17:42 *Rational Recurrences*
Hao Peng, Roy Schwartz, Sam Thomson and Noah A. Smith

17:42–18:00 *Efficient Contextualized Representation: Language Model Pruning for Sequence Labeling*
Liyuan Liu, Xiang Ren, Jingbo Shang, Xiaotao Gu, Jian Peng and Jiawei Han

Session 4B: Information Extraction (Copper Hall)

16:30–16:48 *Automatic Event Salience Identification*
Zhengzhong Liu, Chenyan Xiong, Teruko Mitamura and Eduard Hovy

16:48–17:06 *Temporal Information Extraction by Predicting Relative Time-lines*
Artuur Leeuwenberg and Marie-Francine Moens

17:06–17:24 *Jointly Multiple Events Extraction via Attention-based Graph Information Aggregation*
Xiao Liu, Zhunchen Luo and Heyan Huang

17:24–17:42 *RESIDE: Improving Distantly-Supervised Neural Relation Extraction using Side Information*
Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya and Partha Talukdar

17:42–18:00 *Collective Event Detection via a Hierarchical and Bias Tagging Networks with Gated Multi-level Attention Mechanisms*
Yubo Chen, Hang Yang, Kang Liu, Jun Zhao and Yantao Jia

Session 4C: Syntactic Parsing (Silver Hall)

16:30–16:48 *Valency-Augmented Dependency Parsing*
Tianze Shi and Lillian Lee

16:48–17:06 *Unsupervised Learning of Syntactic Structure with Invertible Neural Projections*
Junxian He, Graham Neubig and Taylor Berg-Kirkpatrick

17:06–17:24 *Dynamic Oracles for Top-Down and In-Order Shift-Reduce Constituent Parsing*
Daniel Fernández-González and Carlos Gómez-Rodríguez

17:24–17:42 *Constituent Parsing as Sequence Labeling*
Carlos Gómez-Rodríguez and David Vilares

17:42–18:00 *Synthetic Data Made to Order: The Case of Parsing*
Dingquan Wang and Jason Eisner

Session 4D: Visual QA (Hall 100)

16:30–16:48 *Tell-and-Answer: Towards Explainable Visual Question Answering using Attributes and Captions*
Qing Li, Jianlong Fu, Dongfei Yu, Tao Mei and Jiebo Luo

16:48–17:06 *Learning a Policy for Opportunistic Active Learning*
Aishwarya Padmakumar, Peter Stone and Raymond Mooney

17:06–17:24 *RecipeQA: A Challenge Dataset for Multimodal Comprehension of Cooking Recipes*
Semih Yagcioglu, Aykut Erdem, Erkut Erdem and Nazli Ikizler-Cinbis

17:24–17:42 *TVQA: Localized, Compositional Video Question Answering*
Jie Lei, Licheng Yu, Mohit Bansal and Tamara Berg

17:42–18:00 *Localizing Moments in Video with Temporal Language*
Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell and Bryan Russell

Session 4E: Semantics III (Posters and Demos, Grand Hall 2)

Card-660: Cambridge Rare Word Dataset - a Reliable Benchmark for Infrequent Word Representation Models
Mohammad Taher Pilehvar, Dimitri Kartsaklis, Victor Prokhorov and Nigel Collier

Leveraging Gloss Knowledge in Neural Word Sense Disambiguation by Hierarchical Co-Attention
Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui and Baobao Chang

Weeding out Conventionalized Metaphors: A Corpus of Novel Metaphor Annotations
Erik-Lân Do Dinh, Hannah Wieland and Iryna Gurevych

Streaming word similarity mining on the cheap
Olof Görnerup and Daniel Gillblad

Memory, Show the Way: Memory Based Few Shot Word Representation Learning
Jingyuan Sun, Shaonan Wang and Chengqing Zong

Disambiguated skip-gram model
Karol Grzegorczyk and Marcin Kurdziel

Picking Apart Story Salads
Su Wang, Eric Holgate, Greg Durrett and Katrin Erk

Dynamic Meta-Embeddings for Improved Sentence Representations
Douwe Kiela, Changhan Wang and Kyunghyun Cho

A Probabilistic Model for Joint Learning of Word Embeddings from Texts and Images
Melissa Ailem, Bowen Zhang, Aurélien Bellet, Pascal Denis and Fei Sha

Transfer and Multi-Task Learning for Noun–Noun Compound Interpretation
Murhaf Fares, Stephan Oepen and Erik Velldal

Dissecting Contextual Word Embeddings: Architecture and Representation
Matthew Peters, Mark Neumann, Luke Zettlemoyer and Wen-tau Yih

Preposition Sense Disambiguation and Representation
Hongyu Gong, Jiaqi Mu, Suma Bhat and Pramod Viswanath

Auto-Encoding Dictionary Definitions into Consistent Word Embeddings
Tom Bosc and Pascal Vincent

Spot the Odd Man Out: Exploring the Associative Power of Lexical Resources
Gabriel Stanovsky and Mark Hopkins

(TACL) Linear Algebraic Structure of Word Senses, with Applications to Polysemy
Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski

Neural Multitask Learning for Simile Recognition
Lizhen Liu, Xiao Hu, Wei Song, Ruiji Fu, Ting Liu and Guoping Hu

Structured Alignment Networks for Matching Sentences
Yang Liu, Matt Gardner and Mirella Lapata

Compare, Compress and Propagate: Enhancing Neural Architectures with Alignment Factorization for Natural Language Inference
Yi Tay, Anh Tuan Luu and Siu Cheung Hui

Convolutional Interaction Network for Natural Language Inference
Jingjing Gong, Xipeng Qiu, Xinchi Chen, Dong Liang and Xuanjing Huang

Lessons from Natural Language Inference in the Clinical Domain
Alexey Romanov and Chaitanya Shivade

Question Generation from SQL Queries Improves Neural Semantic Parsing
Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen and Ming Zhou

SemRegex: A Semantics-Based Approach for Generating Regular Expressions from Natural Language Specifications
Zexuan Zhong, Jiaqi Guo, Wei Yang, Jian Peng, Tao Xie, Jian-Guang Lou, Ting Liu and Dongmei Zhang

Decoupling Structure and Lexicon for Zero-Shot Semantic Parsing
Jonathan Herzig and Jonathan Berant

A Span Selection Model for Semantic Role Labeling
Hiroki Ouchi, Hiroyuki Shindo and Yuji Matsumoto

Mapping Language to Code in Programmatic Context
Srinivasan Iyer, Ioannis Konstas, Alvin Cheung and Luke Zettlemoyer

SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task
Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li and Dragomir Radev

Cross-lingual Decompositional Semantic Parsing
Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh and Benjamin Van Durme

Learning to Learn Semantic Parsers from Natural Language Supervision
Igor Labutov, Bishan Yang and Tom Mitchell

DeepCx: A transition-based approach for shallow semantic parsing with complex constructional triggers
Jesse Dunietz, Jaime Carbonell and Lori Levin

What It Takes to Achieve 100% Condition Accuracy on WikiSQL
Semih Yavuz, Izzeddin Gur, Yu Su and Xifeng Yan

Better Transition-Based AMR Parsing with a Refined Search Space
Zhijiang Guo and Wei Lu

Demo: TRANX: A Transition-based Neural Abstract Syntax Parser for Semantic Parsing and Code Generation
Xiaolin Wang, Masao Utiyama, and Eiichiro Sumita

Demo: Visual Interrogation of Attention-Based Models for Natural Language Inference and Machine Comprehension
Shusen Liu, Tao Li, Zhimin Liu, Vivek Srikumar, Valerio Pascucci, and Peer-Timo Bremer

Demo: Magnitude: A Fast, Efficient Universal Vector Embedding Utility Package
Ajay Patel, Alexander Sands, Chris Callison-Burch, and Marianna Apidianaki

Demo: Universal Sentence Encoder for English
Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil

Saturday, November 3, 2018

09:00–10:30 ***Long Papers and Demos (Orals and Posters) III***

Session 5A: Semantics IV (Gold Hall)

09:00–09:18 *Heuristically Informed Unsupervised Idiom Usage Recognition*
Changsheng Liu and Rebecca Hwa

09:18–09:36 *Coming to Your Senses: on Controls and Evaluation Sets in Polysemy Research*
Haim Dubossarsky, Eitan Grossman and Daphna Weinshall

09:36–09:54 *Predicting Semantic Relations using Global Graph Properties*
Yuval Pinter and Jacob Eisenstein

09:54–10:12 *Learning Scalar Adjective Intensity from Paraphrases*
Anne Cocos, Veronica Wharton, Ellie Pavlick, Marianna Apidianaki and Chris Callison-Burch

10:12–10:30 *Pointwise HSIC: A Linear-Time Kernelized Co-occurrence Norm for Sparse Linguistic Expressions*
Sho Yokoi, Sosuke Kobayashi, Kenji Fukumizu, Jun Suzuki and Kentaro Inui

Session 5B: Summarization (Copper Hall)

09:00–09:18 *Neural Related Work Summarization with a Joint Context-driven Attention Mechanism*
Yongzhen Wang, Xiaozhong Liu and Zheng Gao

09:18–09:36 *Improving Neural Abstractive Document Summarization with Explicit Information Selection Modeling*
Wei Li, Xinyan Xiao, Yajuan Lyu and Yuanzhuo Wang

09:36–09:54 *Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization*
Shashi Narayan, Shay B. Cohen and Mirella Lapata

09:54–10:12 *Improving Abstraction in Text Summarization*
Wojciech Kryściński, Romain Paulus, Caiming Xiong and Richard Socher

10:12–10:30 *Content Selection in Deep Learning Models of Summarization*
Chris Kedzie, Kathleen McKeown and Hal Daume III

Session 5C: IR / Text Mining (Silver Hall)

09:00–09:18 *Improved Semantic-Aware Network Embedding with Fine-Grained Word Alignment*
Dinghan Shen, Xinyuan Zhang, Ricardo Henao and Lawrence Carin

09:18–09:36 *Learning Context-Sensitive Convolutional Filters for Text Processing*
Dinghan Shen, Martin Renqiang Min, Yitong Li and Lawrence Carin

09:36–09:54 *Deep Relevance Ranking Using Enhanced Document-Query Interactions*
Ryan McDonald, George Brokos and Ion Androutsopoulos

09:54–10:12 *Learning Neural Representation for CLIR with Adversarial Framework*
Bo Li and Ping Cheng

10:12–10:30 *AD3: Attentive Deep Document Dater*
Swayambhu Nath Ray, Shib Sankar Dasgupta and Partha Talukdar

Session 5D: Machine Learning II (Hall 100)

09:00–09:18 *Gromov-Wasserstein Alignment of Word Embedding Spaces*
David Alvarez-Melis and Tommi Jaakkola

09:18–09:36 *Deep Probabilistic Logic: A Unifying Framework for Indirect Supervision*
Hai Wang and Hoifung Poon

09:36–09:54 *Deriving Machine Attention from Human Rationales*
Yujia Bao, Shiyu Chang, Mo Yu and Regina Barzilay

09:54–10:12 *Semi-Supervised Sequence Modeling with Cross-View Training*
Kevin Clark, Minh-Thang Luong, Christopher D. Manning and Quoc Le

10:12–10:30 (TACL) Comparing Bayesian Models of Annotation
Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio

Session 5E: Information Extraction, Question Answering (Posters and Demos, Grand Hall 2)

A Probabilistic Annotation Model for Crowdsourcing Coreference
Silviu Paun, Jon Chamberlain, Udo Kruschwitz, Juntao Yu and Massimo Poesio

A Deterministic Algorithm for Bridging Anaphora Resolution
Yufang Hou

A Knowledge Hunting Framework for Common Sense Reasoning
Ali Emami, Noelia De La Cruz, Adam Trischler, Kaheer Suleman and Jackie Chi Kit Cheung

Mapping Text to Knowledge Graph Entities using Multi-Sense LSTMs
Dimitri Kartsaklis, Mohammad Taher Pilehvar and Nigel Collier

Differentiating Concepts and Instances for Knowledge Graph Embedding
Xin Lv, Lei Hou, Juanzi Li and Zhiyuan Liu

One-Shot Relational Learning for Knowledge Graphs
Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo and William Yang Wang

Regular Expression Guided Entity Mention Mining from Noisy Web Data
Shanshan Zhang, Lihong He, Slobodan Vucetic and Eduard Dragut

HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding
Shib Sankar Dasgupta, Swayambhu Nath Ray and Partha Talukdar

Neural Adaptation Layers for Cross-domain Named Entity Recognition
Bill Yuchen Lin and Wei Lu

Entity Linking within a Social Media Platform: A Case Study on Yelp
Hongliang Dai, Yangqiu Song, Liwei Qiu and Rijia Liu

Annotation of a Large Clinical Entity Corpus
Pinal Patel, Disha Davey, Vishal Panchal and Parth Pathak

Visual Supervision in Bootstrapped Information Extraction
Matthew Berger, Ajay Nagesh, Joshua Levine, Mihai Surdeanu and Helen Zhang

Learning Named Entity Tagger using Domain-Specific Dictionary
Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren and Jiawei Han

Zero-Shot Open Entity Typing as Type-Compatible Grounding
Ben Zhou, Daniel Khashabi, Chen-Tse Tsai and Dan Roth

Attention-Guided Answer Distillation for Machine Reading Comprehension
Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang and Ming Zhou

Interpretation of Natural Language Rules in Conversational Machine Reading
Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard and Sebastian Riedel

A State-transition Framework to Answer Complex Questions over Knowledge Base
Sen Hu, Lei Zou and Xinbo Zhang

A Multi-answer Multi-task Framework for Real-world Machine Reading Comprehension
Jiahua Liu, Wan Wei, Maosong Sun, Hao Chen, Yantao Du and Dekang Lin

Logician and Orator: Learning from the Duality between Language and Knowledge in Open Domain
Mingming Sun, Xu Li and Ping Li

MemoReader: Large-Scale Reading Comprehension through Neural Memory Controller
Seohyun Back, Seunghak Yu, Sathish Reddy Indurthi, Jihie Kim and Jaegul Choo

Multi-Granular Sequence Encoding via Dilated Compositional Units for Reading Comprehension
Yi Tay, Anh Tuan Luu and Siu Cheung Hui

Neural Compositional Denotational Semantics for Question Answering
Nitish Gupta and Mike Lewis

Cross-Pair Text Representations for Answer Sentence Selection
Kateryna Tymoshenko and Alessandro Moschitti

QuAC: Question Answering in Context
Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang and Luke Zettlemoyer

Knowledge Base Question Answering via Encoding of Complex Query Graphs
Kangqi Luo, Fengli Lin, Xusheng Luo and Kenny Zhu

Neural Relation Extraction via Inner-Sentence Noise Reduction and Transfer Learning
Tianyi Liu, Xinsong Zhang, Wanhao Zhou and Weijia Jia

Graph Convolution over Pruned Dependency Trees Improves Relation Extraction
Yuhao Zhang, Peng Qi and Christopher D. Manning

Multi-Level Structured Self-Attentions for Distantly Supervised Relation Extraction
Jinhua Du, Jingguang Han, Andy Way and Dadong Wan

N-ary Relation Extraction using Graph-State LSTM
Linfeng Song, Yue Zhang, Zhiguo Wang and Daniel Gildea

Hierarchical Relation Extraction with Coarse-to-Fine Grained Attention
Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun and Peng Li

Label-Free Distant Supervision for Relation Extraction via Knowledge Graph Embedding
Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu and Huajun Chen

Extracting Entities and Relations with Joint Minimum Risk Training
Changzhi Sun, Yuanbin Wu, Man Lan, Shiliang Sun, Wenting Wang, Kuang-Chih Lee and Kewen Wu

Large-scale Exploration of Neural Relation Classification Architectures
Hoang-Quynh Le, Duy-Cat Can, Sinh T. Vu, Thanh Hai Dang, Mohammad Taher Pilehvar and Nigel Collier

Possessors Change Over Time: A Case Study with Artworks
Dhivya Chinnappa and Eduardo Blanco

Demo: CogCompTime: A Tool for Understanding Time in Natural Language
Qiang Ning, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth

Demo: DERE: A Task and Domain-Independent Slot Filling Framework for Declarative Relation Extraction
Heike Adel, Laura Ana Maria Bostan, Sean Papay, Sebastian Padó, and Roman Klinger

Demo: Integrating Knowledge-Supported Search into the INCEpTION Annotation Platform
Beto Boullosa, Richard Eckart de Castilho, Naveen Kumar, Jan-Christoph Klie, and Iryna Gurevych

Demo: OpenKE: An Open Toolkit for Knowledge Embedding
Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li

Demo: An Interactive Web-Interface for Visualizing the Inner Workings of the Question Answering LSTM
Ekaterina Loginova and Günter Neumann

Demo: An Interface for Annotating Science Questions
Michael Boratko, Harshit Padigela, Divyendra Mikkilineni, Pritish Yuvraj, Rajarshi Das, Andrew McCallum, Maria Chang, Achille Fokoue, Pavan Kapanipathi, Nicholas Mattei, Ryan Musa, Kartik Talamadupula, and Michael Witbrock

Demo: Interactive Instance-based Evaluation of Knowledge Base Question Answering
Daniil Sorokin and Iryna Gurevych

10:30–11:00 ***Coffee Break***

11:00–12:30 ***Long Papers and Demos (Orals and Posters) IV***

Session 6A: Dialogue I (Gold Hall)

11:00–11:18 *Using Lexical Alignment and Referring Ability to Address Data Sparsity in Situated Dialog Reference Resolution*
Todd Shore and Gabriel Skantze

11:18–11:36 *Subgoal Discovery for Hierarchical Dialogue Policy Learning*
Da Tang, Xiujun Li, Jianfeng Gao, Chong Wang, Lihong Li and Tony Jebara

11:36–11:54 *Supervised Clustering of Questions into Intents for Dialog System Applications*
Iryna Haponchyk, Antonio Uva, Seunghak Yu, Olga Uryupina and Alessandro Moschitti

11:54–12:12 *Towards Exploiting Background Knowledge for Building Conversation Systems*
Nikita Moghe, Siddhartha Arora, Suman Banerjee and Mitesh M. Khapra

12:12–12:30 *Decoupling Strategy and Generation in Negotiation Dialogues*
He He, Derek Chen, Anusha Balakrishnan and Percy Liang

Session 6B: Question Answering II (Copper Hall)

11:00–11:18 *Large-scale Cloze Test Dataset Created by Teachers*
Qizhe Xie, Guokun Lai, Zihang Dai and Eduard Hovy

11:18–11:36 *emrQA: A Large Corpus for Question Answering on Electronic Medical Records*
Anusri Pampari, Preethi Raghavan, Jennifer Liang and Jian Peng

11:36–11:54 *HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering*
Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov and Christopher D. Manning

11:54–12:12 *Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering*
Todor Mihaylov, Peter Clark, Tushar Khot and Ashish Sabharwal

12:12–12:30 *Evaluating Theory of Mind in Question Answering*
Aida Nematzadeh, Kaylee Burns, Erin Grant, Alison Gopnik and Tom Griffiths

Session 6C: Semantics V (Silver Hall)

11:00–11:18 *A Unified Syntax-aware Framework for Semantic Role Labeling*
Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li and Luo Si

11:18–11:36 *Semantics as a Foreign Language*
Gabriel Stanovsky and Ido Dagan

11:36–11:54 *An AMR Aligner Tuned by Transition-based Parser*
Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin and Ting Liu

11:54–12:12 *Dependency-based Hybrid Trees for Semantic Parsing*
Zhanming Jie and Wei Lu

12:12–12:30 *Policy Shaping and Generalized Update Equations for Semantic Parsing from Denotations*
Dipendra Misra, Ming-Wei Chang, Xiaodong He and Wen-tau Yih

Session 6D: Multilingual Methods II (Hall 100)

11:00–11:18 *Sentence Compression for Arbitrary Languages via Multilingual Pivoting*
Jonathan Mallinson, Rico Sennrich and Mirella Lapata

11:18–11:36 *Unsupervised Cross-lingual Transfer of Word Embedding Spaces*
Ruochen Xu, Yiming Yang, Naoki Otani and Yuexin Wu

11:36–11:54 *XNLI: Evaluating Cross-lingual Sentence Representations*
Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk and Veselin Stoyanov

11:54–12:12 *Joint Multilingual Supervision for Cross-lingual Entity Linking*
Shyam Upadhyay, Nitish Gupta and Dan Roth

12:12–12:30 *Fine-grained Coordinated Cross-lingual Text Stream Alignment for Endless Language Knowledge Acquisition*
Tao Ge, Qing Dou, Heng Ji, Lei Cui, Baobao Chang, Zhifang Sui, Furu Wei and Ming Zhou

Session 6E: Syntax, Morphology, Vision and Language I (Posters and Demos, Grand Hall 2)

WECA:A WordNet-Encoded Collocation-Attention Network for Homographic Pun Recognition
Yufeng Diao, Hongfei Lin, Di Wu, Liang Yang, Kan Xu, Zhihao Yang, Jian Wang, Shaowu Zhang, Bo Xu and Dongyu Zhang

A Hybrid Approach to Automatic Corpus Generation for Chinese Spelling Check
Dingmin Wang, Yan Song, Jing Li, Jialong Han and Haisong Zhang

Neural Quality Estimation of Grammatical Error Correction
Shamil Chollampatt and Hwee Tou Ng

Transferring from Formal Newswire Domain with Hypernet for Twitter POS Tagging
Tao Gui, Qi Zhang, Jingjing Gong, Minlong Peng, di liang, Keyu Ding and Xuanjing Huang

Free as in Free Word Order: An Energy Based Model for Word Segmentation and Morphological Tagging in Sanskrit
Amrith Krishna, Bishal Santra, Sasi Prasanth Bandaru, Gaurav Sahu, Vishnu Dutt Sharma, Pavankumar Satuluri and Pawan Goyal

A Challenge Set and Methods for Noun-Verb Ambiguity
Ali Elkahky, Kellie Webster, Daniel Andor and Emily Pitler

What do character-level models learn about morphology? The case of dependency parsing
Clara Vania, Andreas Grivas and Adam Lopez

Learning Better Internal Structure of Words for Sequence Labeling
Yingwei Xin, Ethan Hart, Vibhuti Mahajan and Jean David Ruvini

ICON: Interactive Conversational Memory Network for Multimodal Emotion Detection
Devamanyu Hazarika, Soujanya Poria, Rada Mihalcea, Erik Cambria and Roger Zimmermann

Discriminative Learning of Open-Vocabulary Object Retrieval and Localization by Negative Phrase Augmentation
Ryota Hinami and Shin'ichi Satoh

Grounding Semantic Roles in Images
Carina Silberer and Manfred Pinkal

Commonsense Justification for Action Explanation
Shaohua Yang, Qiaozi Gao, Sari Sadiya and Joyce Chai

Learning Personas from Dialogue with Attentive Memory Networks
Eric Chu, Prashanth Vijayaraghavan and Deb Roy

Grounding language acquisition by training semantic parsers using captioned videos
Candace Ross, Andrei Barbu, Yevgeni Berzak, Battushig Myanganbayar and Boris Katz

Translating Navigation Instructions in Natural Language to a High-Level Plan for Behavioral Robot Navigation
Xiaoxue Zang, Ashwini Pokle, Marynel Vázquez, Kevin Chen, Juan Carlos Niebles, Alvaro Soto and Silvio Savarese

Mapping Instructions to Actions in 3D Environments with Visual Goal Prediction
Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin and Yoav Artzi

Deconvolutional Time Series Regression: A Technique for Modeling Temporally Diffuse Effects
Cory Shain and William Schuler

Is this Sentence Difficult? Do you Agree?
Dominique Brunato, Lorenzo De Mattei, Felice Dell'Orletta, Benedetta Iavarone and Giulia Venturi

Neural Transition Based Parsing of Web Queries: An Entity Based Approach
Rivka Malca and Roi Reichart

An Investigation of the Interactions Between Pre-Trained Word Embeddings, Character Models and POS Tags in Dependency Parsing
Aaron Smith, Miryam de Lhoneux, Sara Stymne and Joakim Nivre

Depth-bounding is effective: Improvements and evaluation of unsupervised PCFG induction
Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler and Lane Schwartz

(TACL) In-Order Transition-based Constituent Parsing
Jiangming Liu and Yue Zhang

(TACL) Surface Statistics of an Unknown Language Indicate How to Parse It
Dingquan Wang and Jason Eisner

Incremental Computation of Infix Probabilities for Probabilistic Finite Automata
Marco Cognetta, Yo-Sub Han and Soon Chan Kwon

Syntax Encoding with Application in Authorship Attribution
Richong Zhang, Zhiyuan Hu, Hongyu Guo and Yongyi Mao

Sanskrit Word Segmentation Using Character-level Recurrent and Convolutional Neural Networks
Oliver Hellwig and Sebastian Nehrdich

(TACL) Universal Word Segmentation: Implementation and Interpretation
Yan Shao, Christian Hardmeier, and Joakim Nivre

Demo: MorAz: an Open-source Morphological Analyzer for Azerbaijani Turkish
Berke Özenç, Razieh Ehsani, and Ercan Solak

Demo: Juman++: A Morphological Analysis Toolkit for Scriptio Continua
Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi

12:30–13:45 ***Lunch***

13:45–14:45 ***Short Papers (Orals and Posters) III***

Saturday, November 3, 2018 (continued)

Session 7A: Dialogue II (Gold Hall)

13:45–13:57 *Session-level Language Modeling for Conversational Speech*
Wayne Xiong, Lingfeng Wu, Jun Zhang and Andreas Stolcke

13:57–14:09 *Towards Less Generic Responses in Neural Conversation Models: A Statistical Re-weighting Method*
Yahui Liu, Wei Bi, Jun Gao, Xiaojiang Liu, Jian Yao and Shuming Shi

14:09–14:21 *Training Millions of Personalized Dialogue Agents*
Pierre-Emmanuel Mazare, Samuel Humeau, Martin Raison and Antoine Bordes

14:21–14:33 *Towards Universal Dialogue State Tracking*
Liliang Ren, Kaige Xie, Lu Chen and Kai Yu

14:33–14:45 *Semantic Parsing for Task Oriented Dialog using Hierarchical Representations*
Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar and Mike Lewis

Session 7B: Social Applications II (Copper Hall)

13:45–13:57 *The glass ceiling in NLP*
Natalie Schluter

13:57–14:09 *Reducing Gender Bias in Abusive Language Detection*
Ji Ho Park, Jamin Shin and Pascale Fung

14:09–14:21 *SafeCity: Understanding Diverse Forms of Sexual Harassment Personal Stories*
Sweta Karlekar and Mohit Bansal

14:21–14:33 *Learning multiview embeddings for assessing dementia*
Chloé Pou-Prom and Frank Rudzicz

14:33–14:45 *WikiConv: A Corpus of the Complete Conversational History of a Large Online Collaborative Community*
Yiqing Hua, Cristian Danescu-Niculescu-Mizil, Dario Taraborelli, Nithum Thain, Jeffery Sorensen and Lucas Dixon

Session 7C: NER (Silver Hall)

13:45–13:57 *Marginal Likelihood Training of BiLSTM-CRF for Biomedical Named Entity Recognition from Disjoint Label Sets*
Nathan Greenberg, Trapit Bansal, Patrick Verga and Andrew McCallum

13:57–14:09 *Adversarial training for multi-context joint entity and relation extraction*
Giannis Bekoulis, Johannes Deleu, Thomas Demeester and Chris Develder

14:09–14:21 *Structured Multi-Label Biomedical Text Tagging via Attentive Neural Tree Decoding*
Gaurav Singh, James Thomas, Iain Marshall, John Shawe-Taylor and Byron C. Wallace

14:21–14:33 *Deep Exhaustive Model for Nested Named Entity Recognition*
Mohammad Golam Sohrab and Makoto Miwa

14:33–14:45 *Evaluating the Utility of Hand-crafted Features in Sequence Labelling*
Minghao Wu, Fei Liu and Trevor Cohn

Session 7D: Morphology / Parsing (Hall 100)

13:45–13:57 *Improved Dependency Parsing using Implicit Word Connections Learned from Unlabeled Data*
Wenhui Wang, Baobao Chang and Mairgup Mansur

13:57–14:09 *A Framework for Understanding the Role of Morphology in Universal Dependency Parsing*
Mathieu Dehouck and Pascal Denis

14:09–14:21 *The Lazy Encoder: A Fine-Grained Analysis of the Role of Morphology in Neural Machine Translation*
Arianna Bisazza and Clara Tump

14:21–14:33 *Imitation Learning for Neural Morphological String Transduction*
Peter Makarov and Simon Clematide

14:33–14:45 *An Encoder-Decoder Approach to the Paradigm Cell Filling Problem*
Miikka Silfverberg and Mans Hulden

Session 7E: Short Posters III (Grand Hall 2)

Generating Natural Language Adversarial Examples
Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava and Kai-Wei Chang

Multi-Head Attention with Disagreement Regularization
Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu and Tong Zhang

Deep Bayesian Active Learning for Natural Language Processing: Results of a Large-Scale Empirical Study
Aditya Siddhant and Zachary C. Lipton

Bayesian Compression for Natural Language Processing
Nadezhda Chirkova, Ekaterina Lobacheva and Dmitry Vetrov

Multimodal neural pronunciation modeling for spoken languages with logographic origin
Minh Nguyen, Gia H Ngo and Nancy Chen

Chinese Pinyin Aided IME, Input What You Have Not Keystroked Yet
Yafang Huang and Hai Zhao

Estimating Marginal Probabilities of n-grams for Recurrent Neural Language Models
Thanapon Noraset, Doug Downey and Lidong Bing

How to represent a word and predict it, too: Improving tied architectures for language modelling
Kristina Gulordava, Laura Aina and Gemma Boleda

The Importance of Generation Order in Language Modeling
Nicolas Ford, Daniel Duckworth, Mohammad Norouzi and George Dahl

Document-Level Neural Machine Translation with Hierarchical Attention Networks
Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas and James Henderson

Three Strategies to Improve One-to-Many Multilingual Translation
Yining Wang, Jiajun Zhang, Feifei Zhai, Jingfang Xu and Chengqing Zong

Multi-Source Syntactic Neural Machine Translation
Anna Currey and Kenneth Heafield

Fixing Translation Divergences in Parallel Corpora for Neural MT
Minh Quang Pham, Josep Crego, Jean Senellart and François Yvon

Adversarial Evaluation of Multimodal Machine Translation
Desmond Elliott

Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion
Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou and Edouard Grave

Learning When to Concentrate or Divert Attention: Self-Adaptive Attention Temperature for Neural Machine Translation
Junyang Lin, Xu Sun, Xuancheng Ren, Muyu Li and Qi Su

Accelerating Asynchronous Stochastic Gradient Descent for Neural Machine Translation
Nikolay Bogoychev, Kenneth Heafield, Alham Fikri Aji and Marcin Junczys-Dowmunt

Learning to Jointly Translate and Predict Dropped Pronouns with a Shared Reconstruction Mechanism
Longyue Wang, Zhaopeng Tu, Andy Way and Qun Liu

Getting Gender Right in Neural Machine Translation
Eva Vanmassenhove, Christian Hardmeier and Andy Way

Towards Two-Dimensional Sequence to Sequence Model in Neural Machine Translation
Parnia Bahar, Christopher Brix and Hermann Ney

End-to-End Non-Autoregressive Neural Machine Translation with Connectionist Temporal Classification
Jindřich Libovický and Jindřich Helcl

Prediction Improves Simultaneous Neural Machine Translation
Ashkan Alinejad, Maryam Siahbani and Anoop Sarkar

Training Deeper Neural Machine Translation Models with Transparent Attention
Ankur Bapna, Mia Chen, Orhan Firat, Yuan Cao and Yonghui Wu

Context and Copying in Neural Machine Translation
Rebecca Knowles and Philipp Koehn

Encoding Gated Translation Memory into Neural Machine Translation
Qian Cao and Deyi Xiong

Automatic Post-Editing of Machine Translation: A Neural Programmer-Interpreter Approach
Thuy-Trang Vu and Gholamreza Haffari

Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation
Yilin Yang, Liang Huang and Mingbo Ma

Multi-Multi-View Learning: Multilingual and Multi-Representation Entity Typing
Yadollah Yaghoobzadeh and Hinrich Schütze

Word Embeddings for Code-Mixed Language Processing
Adithya Pratapa, Monojit Choudhury and Sunayana Sitaram

On the Strength of Character Language Models for Multilingual Named Entity Recognition
Xiaodong Yu, Stephen Mayhew, Mark Sammons and Dan Roth

Code-switched Language Models Using Dual RNNs and Same-Source Pretraining
Saurabh Garg, Tanmay Parekh and Preethi Jyothi

Part-of-Speech Tagging for Code-Switched, Transliterated Texts without Explicit Language Identification
Kelsey Ball and Dan Garrette

14:45–15:00 ***Mini-Break***

15:00–16:00 ***Keynote II: Gideon Mann "Understanding the News that Moves Markets" (Gold Hall)***

16:00–16:30 ***Coffee Break***

16:30–18:00 *Long Papers and Demos (Orals and Posters) V*

Session 8A: Text Categorization (Gold Hall)

16:30–16:48 *Zero-shot User Intent Detection via Capsule Neural Networks*
Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang and Philip Yu

16:48–17:06 *Hierarchical Neural Networks for Sequential Sentence Classification in Medical Scientific Abstracts*
Di Jin and Peter Szolovits

17:06–17:24 *Investigating Capsule Networks with Dynamic Routing for Text Classification*
Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao and Soufei Zhang

17:24–17:42 *Topic Memory Networks for Short Text Classification*
Jichuan Zeng, Jing Li, Yan Song, Cuiyun Gao, Michael R. Lyu and Irwin King

17:42–18:00 *Few-Shot and Zero-Shot Multi-Label Learning for Structured Label Spaces*
Anthony Rios and Ramakanth Kavuluru

Session 8B: Generation (Copper Hall)

16:30–16:48 *Automatic Poetry Generation with Mutual Reinforcement Learning*
Xiaoyuan Yi, Maosong Sun, Ruoyu Li and Wenhao Li

16:48–17:06 *Variational Autoregressive Decoder for Neural Response Generation*
Jiachen Du, Wenjie Li, Yulan He, Ruifeng Xu, Lidong Bing and Xuan Wang

17:06–17:24 *Integrating Transformer and Paraphrase Rules for Sentence Simplification*
Sanqiang Zhao, Rui Meng, Daqing He, Andi Saptono and Bambang Parmanto

17:24–17:42 *Learning Neural Templates for Text Generation*
Sam Wiseman, Stuart Shieber and Alexander Rush

17:42–18:00 *Multi-Reference Training with Pseudo-References for Neural Translation and Text Generation*
Renjie Zheng, Mingbo Ma and Liang Huang

Session 8C: Knowledge Graphs (Silver Hall)

16:30–16:48 *Knowledge Graph Embedding with Hierarchical Relation Structure*
Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin and Qing He

16:48–17:06 *Embedding Multimodal Relational Data for Knowledge Base Completion*
Pouya Pezeshkpour, Liyan Chen and Sameer Singh

17:06–17:24 *Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction*
Yi Luan, Luheng He, Mari Ostendorf and Hannaneh Hajishirzi

17:24–17:42 *Playing 20 Question Game with Policy-Based Reinforcement Learning*
Huang Hu, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, wei wu and Zhan Chen

17:42–18:00 *Multi-Hop Knowledge Graph Reasoning with Reward Shaping*
Xi Victoria Lin, Richard Socher and Caiming Xiong

Session 8D: Morphology / Phonology (Hall 100)

16:30–16:48 *Neural Transductive Learning and Beyond: Morphological Generation in the Minimal-Resource Setting*
Katharina Kann and Hinrich Schütze

16:48–17:06 *Implicational Universals in Stochastic Constraint-Based Phonology*
Giorgio Magri

17:06–17:24 *Explaining Character-Aware Neural Networks for Word-Level Prediction: Do They Discover Linguistic Rules?*
Fréderic Godin, Kris Demuynck, Joni Dambre, Wesley De Neve and Thomas Demeester

17:24–17:42 *Adapting Word Embeddings to New Languages with Morphological and Phonological Subword Representations*
Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R. Mortensen and Jaime Carbonell

17:42–18:00 (TACL) Recurrent Neural Networks in Linguistic Theory: Revisiting Pinker and Prince (1988) and the Past Tense Debate
Christo Kirov and Ryan Cotterell

Session 8E: Sentiment, Social Applications, Multimodal Semantics, Discourse (Posters and Demos, Grand Hall 2)

A Computational Exploration of Exaggeration
Enrica Troiano, Carlo Strapparava, Gözde Özbal and Serra Sinem Tekiroglu

Building Context-aware Clause Representations for Situation Entity Type Classification
Zeyu Dai and Ruihong Huang

Hierarchical Dirichlet Gaussian Marked Hawkes Process for Narrative Reconstruction in Continuous Time Domain
Yeon Seonwoo, Alice Oh and Sungjoon Park

Investigating the Role of Argumentation in the Rhetorical Analysis of Scientific Publications with Neural Multi-Task Learning Models
Anne Lauscher, Goran Glavaš, Simone Paolo Ponzetto and Kai Eckert

Neural Ranking Models for Temporal Dependency Structure Parsing
Yuchen Zhang and Nianwen Xue

Causal Explanation Analysis on Social Media
Youngseo Son, Nipun Bayas and H. Andrew Schwartz

LRMM: Learning to Recommend with Missing Modalities
Cheng Wang, Mathias Niepert and Hui Li

Content Explorer: Recommending Novel Entities for a Document Writer
Michal Lukasik and Richard Zens

A Genre-Aware Attention Model to Improve the Likability Prediction of Books
Suraj Maharjan, Manuel Montes, Fabio A. González and Thamar Solorio

Thread Popularity Prediction and Tracking with a Permutation-invariant Model
Hou Pong Chan and Irwin King

IARM: Inter-Aspect Relation Modeling with Memory Networks in Aspect-Based Sentiment Analysis
Navonil Majumder, Soujanya Poria, Alexander Gelbukh, Md Shad Akhtar, Erik Cambria and Asif Ekbal

Limbic: Author-Based Sentiment Aspect Modeling Regularized with Word Embeddings and Discourse Relations
Zhe Zhang and Munindar Singh

An Interpretable Neural Network with Topical Information for Relevant Emotion Ranking
Yang Yang, Deyu ZHOU and Yulan He

Multi-grained Attention Network for Aspect-Level Sentiment Classification
Feifan Fan, Yansong Feng and Dongyan Zhao

Attentive Gated Lexicon Reader with Contrastive Contextual Co-Attention for Sentiment Classification
Yi Tay, Anh Tuan Luu, Siu Cheung Hui and Jian Su

Contextual Inter-modal Attention for Multi-modal Sentiment Analysis
Deepanway Ghosal, Md Shad Akhtar, Dushyant Chauhan, Soujanya Poria, Asif Ekbal and Pushpak Bhattacharyya

Adaptive Semi-supervised Learning for Cross-domain Sentiment Classification
Ruidan He, Wee Sun Lee, Hwee Tou Ng and Daniel Dahlmeier

ExtRA: Extracting Prominent Review Aspects from Customer Feedback
Zhiyi Luo, Shanshan Huang, Frank F. Xu, Bill Yuchen Lin, Hanyuan Shi and Kenny Zhu

Cross-Lingual Cross-Platform Rumor Verification Pivoting on Multimedia Content
Weiming Wen, Songwen Su and Zhou Yu

Extractive Adversarial Networks: High-Recall Explanations for Identifying Personal Attacks in Social Media Posts
Samuel Carton, Qiaozhu Mei and Paul Resnick

Automatic Detection of Vague Words and Sentences in Privacy Policies
Logan Lebanoff and Fei Liu

Multi-view Models for Political Ideology Detection of News Articles
Vivek Kulkarni, Junting Ye, Steve Skiena and William Yang Wang

Saturday, November 3, 2018 (continued)

Predicting Factuality of Reporting and Bias of News Media Sources
Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass and Preslav Nakov

Legal Judgment Prediction via Topological Learning
Haoxi Zhong, Guo Zhipeng, Cunchao Tu, Chaojun Xiao, Zhiyuan Liu and Maosong Sun

Hierarchical CVAE for Fine-Grained Hate Speech Classification
Jing Qian, Mai ElSherief, Elizabeth Belding and William Yang Wang

Residualized Factor Adaptation for Community Social Media Prediction Tasks
Mohammadzaman Zamani, H. Andrew Schwartz, Veronica Lynn, Salvatore Giorgi and Niranjan Balasubramanian

Framing and Agenda-setting in Russian News: a Computational Analysis of Intricate Political Strategies
Anjalie Field, Doron Kliger, Shuly Wintner, Jennifer Pan, Dan Jurafsky and Yulia Tsvetkov

Identifying the sentiment styles of YouTube's vloggers
Bennett Kleinberg, Maximilian Mozes and Isabelle van der Vegt

Native Language Identification with User Generated Content
Gili Goldin, Ella Rabinovich and Shuly Wintner

Demo: Visualization of the Topic Space of Argument Search Results in args.me
Yamen Ajjour, Henning Wachsmuth, Dora Kiesel, Patrick Riehmann, Fan Fan, Giuliano Castiglia, Rosemary Adejoh, Bernd Fröhlich, and Benno Stein

Saturday, November 3, 2018 (continued)

Demo: A Multilingual Information Extraction Pipeline for Investigative Journalism
Gregor Wiedemann, Seid Muhie Yimam, and Chris Biemann

Demo: When science journalism meets artificial intelligence : An interactive demonstration
Raghuram Vadapalli, Bakhtiyar Syed, Nishant Prabhu, Balaji Vasan Srinivasan, and Vasudeva Varma

19:00–22:00 ***Social Event (Royal Museums of Fine Arts of Belgium)***

Sunday, November 4, 2018

09:00–10:30 ***Long Papers and Demos (Orals and Posters) VI***

Session 9A: Machine Translation II (Gold Hall)

09:00–09:18 *Beyond Error Propagation in Neural Machine Translation: Characteristics of Language Also Matter*
Lijun Wu, Xu Tan, Di He, Fei Tian, Tao Qin, Jianhuang Lai and Tie-Yan Liu

09:18–09:36 *A Study of Reinforcement Learning for Neural Machine Translation*
Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai and Tie-Yan Liu

09:36–09:54 *Meta-Learning for Low-Resource Neural Machine Translation*
Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li and Kyunghyun Cho

09:54–10:12 *Unsupervised Statistical Machine Translation*
Mikel Artetxe, Gorka Labaka and Eneko Agirre

10:12–10:30 *A Visual Attention Grounding Neural Model for Multimodal Machine Translation*
Mingyang Zhou, Runxiang Cheng, Yong Jae Lee and Zhou Yu

Sunday, November 4, 2018 (continued)

Session 9B: Sentiment I (Copper Hall)

09:00–09:18 *Sentiment Classification towards Question-Answering with Hierarchical Matching Network*
Chenlin Shen, Changlong Sun, Jingjing Wang, Yangyang Kang, Shoushan Li, Xiaozhong Liu, Luo Si, Min Zhang and Guodong Zhou

09:18–09:36 *Cross-topic Argument Mining from Heterogeneous Sources*
Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai and Iryna Gurevych

09:36–09:54 *Summarizing Opinions: Aspect Extraction Meets Sentiment Prediction and They Are Both Weakly Supervised*
Stefanos Angelidis and Mirella Lapata

09:54–10:12 *CARER: Contextualized Affect Representations for Emotion Recognition*
Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu and Yi-Shin Chen

10:12–10:30 (TACL) Adversarial Deep Averaging Networks for Cross-Lingual Sentiment Classification
Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger

Session 9C: Machine Learning III (Silver Hall)

09:00–09:18 *Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency*
Zhuang Ma and Michael Collins

09:18–09:36 *CaLcs: Continuously Approximating Longest Common Subsequence for Sequence Level Optimization*
Semih Yavuz, Chung-Cheng Chiu, Patrick Nguyen and Yonghui Wu

09:36–09:54 *Pathologies of Neural Models Make Interpretations Difficult*
Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez and Jordan Boyd-Graber

09:54–10:12 *Phrase-level Self-Attention Networks for Universal Sentence Encoding*
Wei Wu, Houfeng Wang, Tianyu Liu and Shuming Ma

10:12–10:30 *BanditSum: Extractive Summarization as a Contextual Bandit*
Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof and Jackie Chi Kit Cheung

Session 9D: Semantics VI (Hall 100)

09:00–09:18 *A Word-Complexity Lexicon and A Neural Readability Ranking Model for Lexical Simplification*
Mounica Maddela and Wei Xu

09:18–09:36 *Learning Latent Semantic Annotations for Grounding Natural Language to Structured Data*
Guanghui Qin, Jin-Ge Yao, Xuening Wang, Jinpeng Wang and Chin-Yew Lin

09:36–09:54 *Syntactic Scaffolds for Semantic Structures*
Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer and Noah A. Smith

09:54–10:12 *Hierarchical Quantized Representations for Script Generation*
Noah Weber, Leena Shekhar, Niranjan Balasubramanian and Nate Chambers

10:12–10:30 *Semantic Role Labeling for Learner Chinese: the Importance of Syntactic Parsing and L2-L1 Parallel Data*
Zi Lin, Yuguang Duan, Yuanyuan Zhao, Weiwei Sun and Xiaojun Wan

Session 9E: Generation, Dialog, Summarization; Vision and Language II (Posters and Demos, Grand Hall 2)

A Teacher-Student Framework for Maintainable Dialog Manager
Weikang Wang, Jiajun Zhang, Han Zhang, Mei-Yuh Hwang, Chengqing Zong and Zhifei Li

Discriminative Deep Dyna-Q: Robust Planning for Dialogue Policy Learning
Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu and Yun-Nung Chen

A Self-Attentive Model with Gate Mechanism for Spoken Language Understanding
Changliang Li, Liang Li and Ji Qi

Learning End-to-End Goal-Oriented Dialog with Multiple Answers
Janarthanan Rajendran, Jatin Ganhotra, Satinder Singh and Lazaros Polymenakos

AirDialogue: An Environment for Goal-Oriented Dialogue Research
Wei Wei, Quoc Le, Andrew Dai and Jia Li

(TACL) Polite Dialogue Generation Without Parallel Data
Tong Niu and Mohit Bansal

QuaSE: Sequence Editing under Quantifiable Guidance
Yi Liao, Lidong Bing, Piji Li, Shuming Shi, Wai Lam and Tong Zhang

Paraphrase Generation with Deep Reinforcement Learning
Zichao Li, Xin Jiang, Lifeng Shang and Hang Li

Operation-guided Neural Networks for High Fidelity Data-To-Text Generation
Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan and Chin-Yew Lin

Generating Classical Chinese Poems via Conditional Variational Autoencoder and Adversarial Training
Juntao Li, Yan Song, Haisong Zhang, Dongmin Chen, Shuming Shi, Dongyan Zhao and Rui Yan

Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks
Yao Zhao, Xiaochuan Ni, Yuanyuan Ding and Qifa Ke

Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task
Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang and Dragomir Radev

Unsupervised Natural Language Generation with Denoising Autoencoders
Markus Freitag and Scott Roy

Answer-focused and Position-aware Neural Question Generation
Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma and Shi Wang

Diversity-Promoting GAN: A Cross-Entropy Based Generative Adversarial Network for Diversified Text Generation
Jingjing Xu, Xuancheng Ren, Junyang Lin and Xu Sun

Towards a Better Metric for Evaluating Question Generation Systems
Preksha Nema and Mitesh M. Khapra

Stylistic Chinese Poetry Generation via Unsupervised Style Disentanglement
Cheng Yang, Maosong Sun, Xiaoyuan Yi and Wenhao Li

Generating More Interesting Responses in Neural Conversation Models with Distributional Constraints
Ashutosh Baheti, Alan Ritter, Jiwei Li and Bill Dolan

Better Conversations by Modeling, Filtering, and Optimizing for Coherence and Diversity
Xinnuo Xu, Ondřej Dušek, Ioannis Konstas and Verena Rieser

Incorporating Background Knowledge into Video Description Generation
Spencer Whitehead, Heng Ji, Mohit Bansal, Shih-Fu Chang and Clare Voss

Multimodal Differential Network for Visual Question Generation
Badri Narayana Patro, Sandeep Kumar, Vinod Kumar Kurmi and Vinay Namboodiri

Entity-aware Image Caption Generation
Di Lu, Spencer Whitehead, Lifu Huang, Heng Ji and Shih-Fu Chang

Learning to Describe Differences Between Pairs of Similar Images
Harsh Jhamtani and Taylor Berg-Kirkpatrick

Object Hallucination in Image Captioning
Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell and Kate Saenko

Abstractive Text-Image Summarization Using Multi-Modal Attentional Hierarchical RNN
Jingqiang Chen and Hai Zhuge

Keyphrase Generation with Correlation Constraints
Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan and Zhoujun Li

Closed-Book Training to Improve Summarization Encoder Memory
Yichen Jiang and Mohit Bansal

Improving Neural Abstractive Document Summarization with Structural Regularization
Wei Li, Xinyan Xiao, Yajuan Lyu and Yuanzhuo Wang

Iterative Document Representation Learning Towards Summarization with Polishing
Xiuying Chen, Shen Gao, Chongyang Tao, Yan Song, Dongyan Zhao and Rui Yan

Bottom-Up Abstractive Summarization
Sebastian Gehrmann, Yuntian Deng and Alexander Rush

Controlling Length in Abstractive Summarization Using a Convolutional Neural Network
Yizhu Liu, Zhiyi Luo and Kenny Zhu

APRIL: Interactively Learning to Summarise by Combining Active Preference Learning and Reinforcement Learning
Yang Gao, Christian M. Meyer and Iryna Gurevych

Adapting the Neural Encoder-Decoder Framework from Single to Multi-Document Summarization
Logan Lebanoff, Kaiqiang Song and Fei Liu

Semi-Supervised Learning for Neural Keyphrase Generation
Hai Ye and Lu Wang

MSMO: Multimodal Summarization with Multimodal Output
Junnan Zhu, Haoran Li, Tianshang Liu, Yu Zhou, Jiajun Zhang and Chengqing Zong

Frustratingly Easy Model Ensemble for Abstractive Summarization
Hayato Kobayashi

Automatic Pyramid Evaluation Exploiting EDU-based Extractive Reference Summaries
Tsutomu Hirao, Hidetaka Kamigaito and Masaaki Nagata

Learning to Encode Text as Human-Readable Summaries using Generative Adversarial Networks
Yaushian Wang and Hung-yi Lee

Demo: Visualizing Group Dynamics based on Multiparty Meeting Understanding
Ni Zhang, Tongtao Zhang, Indrani Bhattacharya, Heng Ji, and Rich Radke

Demo: PizzaPal: Conversational Pizza Ordering using a High-Density Conversational AI Platform
Antoine Raux, Yi Ma, Paul Yang, and Felicia Wong

Demo: Developing Production-Level Conversational Interfaces with Shallow Semantic Parsing
Arushi Raghuvanshi, Lucien Carroll, and Karthik Raghunathan

Demo: SyntaViz: Visualizing Voice Queries through a Syntax-Driven Hierarchical Ontology
Md Iftekhar Tanveer and Ferhan Ture

Demo: LIA: A Natural Language Programmable Personal Assistant
Igor Labutov, Shashank Srivastava, and Tom Mitchell

Demo: Data2Text Studio: Automated Text Generation from Structured Data
Longxu Dou, Guanghui Qin, Jinpeng Wang, Jin-Ge Yao, and Chin-Yew Lin

Demo: Demonstrating Par4Sem - A Semantic Writing Aid with Adaptive Paraphrasing
Seid Muhie Yimam and Chris Biemann

10:30–11:00 ***Coffee Break***

11:00–12:30 ***Long Papers and Demos (Orals and Posters) VII***

Sunday, November 4, 2018 (continued)

Session 10A: Question Answering III (Gold Hall)

11:00–11:18 *Joint Multitask Learning for Community Question Answering Using Task-Specific Embeddings*
Shafiq Joty, Lluís Màrquez and Preslav Nakov

11:18–11:36 *What Makes Reading Comprehension Questions Easier?*
Saku Sugawara, Kentaro Inui, Satoshi Sekine and Akiko Aizawa

11:36–11:54 *Commonsense for Generative Multi-Hop Question Answering Tasks*
Lisa Bauer, Yicheng Wang and Mohit Bansal

11:54–12:12 *Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text*
Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov and William Cohen

12:12–12:30 *A Nil-Aware Answer Extraction Framework for Question Answering*
Souvik Kundu and Hwee Tou Ng

Session 10B: Machine Translation III (Copper Hall)

11:00–11:18 *Exploiting Deep Representations for Neural Machine Translation*
Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi and Tong Zhang

11:18–11:36 *Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures*
Gongbo Tang, Mathias Müller, Annette Rios and Rico Sennrich

11:36–11:54 *Simplifying Neural Machine Translation with Addition-Subtraction Twin-Gated Recurrent Networks*
Biao Zhang, Deyi Xiong, jinsong su, Qian Lin and Huiji Zhang

11:54–12:12 *Speeding Up Neural Machine Translation Decoding by Cube Pruning*
Wen Zhang, Liang Huang, Yang Feng, Lei Shen and Qun Liu

12:12–12:30 *Revisiting Character-Based Neural Machine Translation with Capacity and Compression*
Colin Cherry, George Foster, Ankur Bapna, Orhan Firat and Wolfgang Macherey

Session 10C: Discourse (Silver Hall)

11:00–11:18 *A Skeleton-Based Model for Promoting Coherence Among Sentences in Narrative Story Generation*
Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai and Xu Sun

11:18–11:36 *NEXUS Network: Connecting the Preceding and the Following in Dialogue Generation*
Xiaoyu Shen, Hui Su, Wenjie Li and Dietrich Klakow

11:36–11:54 *A Neural Local Coherence Model for Text Quality Assessment*
Mohsen Mesgar and Michael Strube

11:54–12:12 *Deep Attentive Sentence Ordering Network*
Baiyun Cui, Yingming Li, Ming Chen and Zhongfei Zhang

12:12–12:30 *Getting to "Hearer-old": Charting Referring Expressions Across Time*
Ieva Staliūnaitė, Hannah Rohde, Bonnie Webber and Annie Louis

Session 10D: Evolution / Sociolinguistics (Hall 100)

11:00–11:18 *Making "fetch" happen: The influence of social and linguistic context on nonstandard word growth and decline*
Ian Stewart and Jacob Eisenstein

11:18–11:36 *Analyzing Correlated Evolution of Multiple Features Using Latent Representations*
Yugo Murawaki

11:36–11:54 *Capturing Regional Variation with Distributed Place Representations and Geographic Retrofitting*
Dirk Hovy and Christoph Purschke

11:54–12:12 *Characterizing Interactions and Relationships between People*
Farzana Rashid and Eduardo Blanco

12:12–12:30 *Why Swear? Analyzing and Inferring the Intentions of Vulgar Expressions*
Eric Holgate, Isabel Cachola, Daniel Preoţiuc-Pietro and Junyi Jessy Li

Session 10E: Machine Learning (Posters and Demos, Grand Hall 2)

Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP tasks
Steffen Eger, Paul Youssef and Iryna Gurevych

Hard Non-Monotonic Attention for Character-Level Transduction
Shijie Wu, Pamela Shapiro and Ryan Cotterell

Speed Reading: Learning to Read ForBackward via Shuttle
Tsu-Jui Fu and Wei-Yun Ma

Modeling Localness for Self-Attention Networks
Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao and Tong Zhang

Chargrid: Towards Understanding 2D Documents
Anoop R Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne and Jean Baptiste Faddoul

Simple Recurrent Units for Highly Parallelizable Recurrence
Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai and Yoav Artzi

NPRF: A Neural Pseudo Relevance Feedback Framework for Ad-hoc Information Retrieval
Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun and Jungang Xu

Co-Stack Residual Affinity Networks with Multi-level Attention Refinement for Matching Text Sequences
Yi Tay, Anh Tuan Luu and Siu Cheung Hui

Spherical Latent Spaces for Stable Variational Autoencoders
Jiacheng Xu and Greg Durrett

Learning Universal Sentence Representations with Mean-Max Attention Autoencoder
Minghua Zhang, Yunfang Wu, Weikang Li and Wei Li

Word Mover's Embedding: From Word2Vec to Document Embedding
Lingfei Wu, Ian En-Hsu Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar and Michael J. Witbrock

Sunday, November 4, 2018 (continued)

Multilingual Clustering of Streaming News
Sebastião Miranda, Arturs Znotins, Shay B. Cohen and Guntis Barzdins

Multi-Task Label Embedding for Text Classification
Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang and Yaohui Jin

Semantic-Unit-Based Dilated Convolution for Multi-Label Text Classification
Junyang Lin, Qi Su, Pengcheng Yang, Shuming Ma and Xu Sun

MCapsNet: Capsule Network for Text with Multi-Task Learning
Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang and Yaohui Jin

Uncertainty-aware generative models for inferring document class prevalence
Katherine Keith and Brendan O'Connor

Challenges of Using Text Classifiers for Causal Inference
Zach Wood-Doughty, Ilya Shpitser and Mark Dredze

Direct Output Connection for a High-Rank Language Model
Sho Takase, Jun Suzuki and Masaaki Nagata

Disfluency Detection using Auto-Correlational Neural Networks
Paria Jamshid Lou, Peter Anderson and Mark Johnson

Pyramidal Recurrent Unit for Language Modeling
Sachin Mehta, Rik Koncel-Kedziorski, Mohammad Rastegari and Hannaneh Hajishirzi

On Tree-Based Neural Sentence Modeling
Haoyue Shi, Hao Zhou, Jiaze Chen and Lei Li

Language Modeling with Sparse Product of Sememe Experts
Yihong Gu, Jun Yan, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin and Leyu Lin

(TACL) Language Modeling for Morphologically Rich Languages: Character-Aware Modeling for Word-Level Prediction
Daniela Gerz, Ivan Vulic, Edoardo Maria, Jason Naradowsky, Roi Reichart, and Anna Korhonen

(TACL) Low-rank RNN Adaptation for Context-Aware Language Modeling
Aaron Jaech and Mari Ostendorf

Siamese Network-Based Supervised Topic Modeling
Minghui Huang, Yanghui Rao, Yuwei Liu, Haoran Xie and Fu Lee Wang

GraphBTM: Graph Enhanced Autoencoded Variational Inference for Biterm Topic Model
Qile Zhu, Zheng Feng and Xiaolin Li

Modeling Online Discourse with Coupled Distributed Topics
Akshay Srivatsan, Zachary Wojtowicz and Taylor Berg-Kirkpatrick

Learning Disentangled Representations of Texts with Application to Biomedical Abstracts
Sarthak Jain, Edward Banner, Jan-Willem van de Meent, Iain J Marshall and Byron C. Wallace

Multi-Source Domain Adaptation with Mixture of Experts
Jiang Guo, Darsh Shah and Regina Barzilay

Demo: Sisyphus, a Workflow Manager Designed for Machine Translation and Automatic Speech Recognition
Jan-Thorsten Peter, Eugen Beck, and Hermann Ney

Demo: APLenty: annotation tool for creating high-quality datasets using active and proactive learning
Minh-Quoc Nghiem and Sophia Ananiadou

Demo: KT-Speech-Crawler: Automatic Dataset Construction for Speech Recognition from YouTube Videos
Egor Lakomkin, Sven Magg, Cornelius Weber, and Stefan Wermter

Demo: Term Set Expansion based NLP Architect by Intel AI Lab
Jonathan Mamou, Oren Pereg, Moshe Wasserblat, Alon Eirew, Yael Green, Shira Guskin, Peter Izsak, and Daniel Korat

12:30–13:45 ***Lunch***

13:00–13:45 ***Business Meeting***

13:45–14:45 ***Short Papers (Orals and Posters) IV***

Session 11A: Analyzing Models (Gold Hall)

13:45–13:57 *A Neural Model of Adaptation in Reading*
Marten van Schijndel and Tal Linzen

13:57–14:09 *Understanding Deep Learning Performance through an Examination of Test Set Difficulty: A Psychometric Case Study*
John Lalor, Hao Wu, Tsendsuren Munkhdalai and Hong Yu

14:09–14:21 *Lexicosyntactic Inference in Neural Models*
Aaron Steven White, Rachel Rudinger, Kyle Rawlins and Benjamin Van Durme

14:21–14:33 *Dual Fixed-Size Ordinally Forgetting Encoding (FOFE) for Competitive Neural Language Models*
Sedtawut Watcharawittayakul, Mingbin Xu and Hui Jiang

14:33–14:45 *The Importance of Being Recurrent for Modeling Hierarchical Structure*
Ke Tran, Arianna Bisazza and Christof Monz

Session 11B: Sentiment II (Copper Hall)

13:45–13:57 *Joint Learning for Targeted Sentiment Analysis*
Dehong Ma, Sujian Li and Houfeng Wang

13:57–14:09 *Revisiting the Importance of Encoding Logic Rules in Sentiment Classification*
Kalpesh Krishna, Preethi Jyothi and Mohit Iyyer

14:09–14:21 *A Co-Attention Neural Network Model for Emotion Cause Analysis with Emotional Context Awareness*
Xiangju Li, Kaisong Song, Shi Feng, Daling Wang and Yifei Zhang

14:21–14:33 *Modeling Empathy and Distress in Reaction to News Stories*
Sven Buechel, Anneke Buffone, Barry Slaff, Lyle Ungar and Joao Sedoc

14:33–14:45 *Interpretable Emoji Prediction via Label-Wise Attention LSTMs*
Francesco Barbieri, Luis Espinosa Anke, Jose Camacho-Collados, Steven Schockaert and Horacio Saggion

Session 11C: Machine Translation IV (Silver Hall)

13:45–13:57 *A Tree-based Decoder for Neural Machine Translation*
Xinyi Wang, Hieu Pham, Pengcheng Yin and Graham Neubig

13:57–14:09 *Greedy Search with Probabilistic N-gram Matching for Neural Machine Translation*
Chenze Shao, Xilin Chen and Yang Feng

14:09–14:21 *Exploring Recombination for Efficient Decoding of Neural Machine Translation*
Zhisong Zhang, Rui Wang, Masao Utiyama, Eiichiro Sumita and Hai Zhao

14:21–14:33 *Has Machine Translation Achieved Human Parity? A Case for Document-level Evaluation*
Samuel Läubli, Rico Sennrich and Martin Volk

14:33–14:45 *Automatic Reference-Based Evaluation of Pronoun Translation Misses the Point*
Liane Guillou and Christian Hardmeier

Session 11D: QA / Knowledge Graphs (Hall 100)

13:45–13:57 *FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation*
Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu and Maosong Sun

13:57–14:09 *A strong baseline for question relevancy ranking*
Ana Gonzalez, Isabelle Augenstein and Anders Søgaard

14:09–14:21 *Learning Sequence Encoders for Temporal Knowledge Graph Completion*
Alberto Garcia-Duran, Sebastijan Dumančić and Mathias Niepert

14:21–14:33 *Similar but not the Same: Word Sense Disambiguation Improves Event Detection via Neural Representation Matching*
Weiyi Lu and Thien Huu Nguyen

14:33–14:45 *Learning Word Representations with Cross-Sentence Dependency for End-to-End Co-reference Resolution*
Hongyin Luo and Jim Glass

Session 11E: Short Posters IV (Grand Hall 2)

Word Relation Autoencoder for Unseen Hypernym Extraction Using Word Embeddings
Hong-You Chen, Cheng-Syuan Lee, Keng-Te Liao and Shou-de Lin

Refining Pretrained Word Embeddings Using Layer-wise Relevance Propagation
Akira Utsumi

Learning Gender-Neutral Word Embeddings
Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang and Kai-Wei Chang

Learning Concept Abstractness Using Weak Supervision
Ella Rabinovich, Benjamin Sznajder, Artem Spector, Ilya Shnayderman, Ranit Aharonov, David Konopnicki and Noam Slonim

Word Sense Induction with Neural biLM and Symmetric Patterns
Asaf Amrami and Yoav Goldberg

Sunday, November 4, 2018 (continued)

InferLite: Simple Universal Sentence Representations from Natural Language Inference Data
Jamie Kiros and William Chan

Similarity-Based Reconstruction Loss for Meaning Representation
Olga Kovaleva, Anna Rumshisky and Alexey Romanov

What can we learn from Semantic Tagging?
Mostafa Abdou, Artur Kulmizev, Vinit Ravishankar, Lasha Abzianidze and Johan Bos

Conditional Word Embedding and Hypothesis Testing via Bayes-by-Backprop
Rujun Han, Michael Gill, Arthur Spirling and Kyunghyun Cho

Classifying Referential and Non-referential It Using Gaze
Victoria Yaneva, Le An Ha, Richard Evans and Ruslan Mitkov

State-of-the-art Chinese Word Segmentation with Bi-LSTMs
Ji Ma, Kuzman Ganchev and David Weiss

Sanskrit Sandhi Splitting using seq2(seq)2
Rahul Aralikatte, Neelamadhav Gantayat, Naveen Panwar, Anush Sankaran and Senthil Mani

Unsupervised Neural Word Segmentation for Chinese via Segmental Language Modeling
Zhiqing Sun and Zhi-Hong Deng

LemmaTag: Jointly Tagging and Lemmatizing for Morphologically Rich Languages with BRNNs
Daniel Kondratyuk, Tomáš Gavenčiak, Milan Straka and Jan Hajič

Recovering Missing Characters in Old Hawaiian Writing
Brendan Shillingford and Oiwi Parker Jones

When data permutations are pathological: the case of neural natural language inference
Natalie Schluter and Daniel Varab

Bridging Knowledge Gaps in Neural Entailment via Symbolic Models
Dongyeop Kang, Tushar Khot, Ashish Sabharwal and Peter Clark

The BQ Corpus: A Large-scale Domain-specific Chinese Corpus For Sentence Semantic Equivalence Identification
Jing Chen, Qingcai Chen, Xin Liu, Haijun Yang, Daohe Lu and Buzhou Tang

Interpreting Recurrent and Attention-Based Neural Models: a Case Study on Natural Language Inference
Reza Ghaeini, Xiaoli Fern and Prasad Tadepalli

Towards Semi-Supervised Learning for Deep Semantic Role Labeling
Sanket Vaibhav Mehta, Jay Yoon Lee and Jaime Carbonell

Identifying Domain Adjacent Instances for Semantic Parsers
James Ferguson, Janara Christensen, Edward Li and Edgar Gonzàlez

Mapping natural language commands to web elements
Panupong Pasupat, Tian-Shun Jiang, Evan Liu, Kelvin Guu and Percy Liang

Wronging a Right: Generating Better Errors to Improve Grammatical Error Detection
Sudhanshu Kasewa, Pontus Stenetorp and Sebastian Riedel

Modeling Input Uncertainty in Neural Network Dependency Parsing
Rob van der Goot and Gertjan van Noord

Parameter sharing between dependency parsers for related languages
Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein and Anders Søgaard

Grammar Induction with Neural Language Models: An Unusual Replication
Phu Mon Htut, Kyunghyun Cho and Samuel Bowman

Data Augmentation via Dependency Tree Morphing for Low-Resource Languages
Gozde Gul Sahin and Mark Steedman

14:45–15:00 ***Mini-Break***

15:00–16:00 ***Keynote III: Johan Bos "The Moment of Meaning and the Future of Computational Semantics" (Gold Hall)***

16:00–16:30 ***Coffee Break***

16:30–18:00 ***Best Paper Awards and Closing (Gold Hall)***

16:30–16:42 *How Much Reading Does Reading Comprehension Require? A Critical Investigation of Popular Benchmarks*
Divyansh Kaushik and Zachary C. Lipton

16:42–17:00 *MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling*
Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan and Milica Gasic

17:00–17:18 *Linguistically-Informed Self-Attention for Semantic Role Labeling*
Emma Strubell, Patrick Verga, Daniel Andor, David Weiss and Andrew McCallum

17:18–17:36 *Phrase-Based & Neural Unsupervised Machine Translation*
Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer and Marc'Aurelio Ranzato

Privacy-preserving Neural Representations of Text

Maximin Coavoux **Shashi Narayan** **Shay B. Cohen**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
{mcoavoux,scohen}@inf.ed.ac.uk,shashi.narayan@ed.ac.uk

Abstract

This article deals with adversarial attacks towards deep learning systems for Natural Language Processing (NLP), in the context of privacy protection. We study a specific type of attack: an attacker eavesdrops on the hidden representations of a neural text classifier and tries to recover information about the input text. Such scenario may arise in situations when the computation of a neural network is shared across multiple devices, e.g. some hidden representation is computed by a user's device and sent to a cloud-based model. We measure the privacy of a hidden representation by the ability of an attacker to predict accurately specific private information from it and characterize the tradeoff between the privacy and the utility of neural representations. Finally, we propose several defense methods based on modified training objectives and show that they improve the privacy of neural representations.

1 Introduction

This article presents an adversarial scenario meant at characterizing the privacy of neural representations for NLP tasks, as well as defense methods designed to improve the privacy of those representations. A deep neural network constructs intermediate hidden representations to extract features from its input. Such representations are trained to predict a label, and therefore should contain useful features for the final prediction. However, they might also encode information about the input that a user wants to keep private (e.g. personal data) and can be exploited for adversarial usages.

We study a specific type of attack on neural representations: an attacker eavesdrops on the hidden representations of novel input examples (that are not in the training set) and tries to recover information about the content of the input text (Figure 1). A typical scenario where such attacks would occur is when the computation of a deep neural net is shared between several devices (Li et al., 2017). For example, a user's device computes a representation of a textual input, and sends it a to cloud-based neural network to obtain, e.g. the topic of the text or its sentiment. The scenario is illustrated in Figure 1.

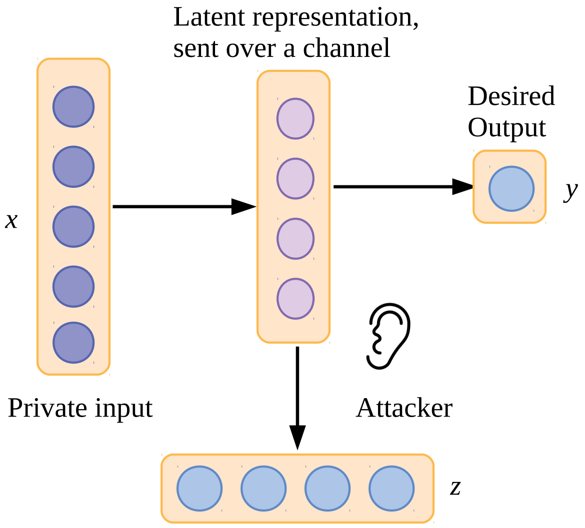

Figure 1: General setting illustration. The main classifier predicts a label y from a text x, the attacker tries to recover some private information $\mathbf{z}$ contained in x from the latent representation used by the main classifier.

Private information can take the form of key phrases *explicitly* contained in the text. However, it can also be *implicit*. For example, demographic information about the author of a text can be predicted with above chance accuracy from linguistic cues in the text itself (Rosenthal and McKeown, 2011; Preoţiuc-Pietro et al., 2015).

Independently of its explicitness, some of this private information correlates with the output labels, and therefore will be learned by the network. In such a case, there is a tradeoff between the utility of the representation (measured by the accuracy of the network) and its privacy. It might be

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1–10
Brussels, Belgium, October 31 - November 4, 2018.

necessary to sacrifice some accuracy in order to satisfy privacy requirements.

However, this is not the case of all private information, since some of it is not relevant for the prediction of the text label. Still, private information might be learned incidentally. This non-intentional and incidental learning also raises privacy concerns, since an attacker with an access to the hidden representations, may exploit them to recover information about the input.

In this paper we explore the following situation: (i) a *main classifier* uses a deep network to predict a label from textual data; (ii) an *attacker* eavesdrops on the hidden layers of the network and tries to recover information about the input text of unseen examples. In contrast to previous work about neural networks and privacy (Papernot et al., 2016; Carlini et al., 2018) we do not protect the privacy of examples from the training set, but the privacy of unseen examples provided, e.g., by a user.

An example of a potential application would be a spam detection service with the following constraints: the service provider does not access verbatim emails sent to users, only their vector representations. Theses vector representations should not be usable to gather information about the user's contacts or correspondents, i.e. protect the user from profiling.

This paper makes the following contributions:[1]

- We propose a metric to measure the privacy of the neural representation of an input for Natural Language Processing tasks. The metric is based on the ability of an attacker to recover information about the input from the latent representation only.
- We present defense methods designed against this type of attack. The methods are based on modified training objectives and lead to an improved privacy-accuracy tradeoff.

2 Adversarial Scenario

In the scenario we propose, each example consists of a triple $(x, y, \mathbf{z})$, where x is a natural language text, y is a single label (e.g. topic or sentiment), and $\mathbf{z}$ is a vector of private information contained in x. Our base setting has two entities: (i) a *main classifier* whose role is to learn to predict y from x, (ii) an *attacker* who learns to predict $\mathbf{z}$ from the latent representation of x used by the main classifier. We illustrate this setting in Figure 1.

In order to evaluate the utility and privacy of a specific model, we proceed in three phases:

Phase 1. Training of the main classifier on (x, y) pairs and evaluation of its accuracy;

Phase 2. Generation of a dataset of pairs $(\mathbf{r}(x), \mathbf{z})$ for the attacker, $\mathbf{r}$ is the representation function of the main classifier ($\mathbf{r}$ is defined in Section 2.1);

Phase 3. Training of the attacker's network and evaluation of its performance for measuring privacy.

In the remainder of this section, we describe the main classifier (Section 2.1), and the attacker's model (Section 2.2).

2.1 Text Classifier

As our base model, we chose a standard LSTM architecture (Hochreiter and Schmidhuber, 1997) for sequence classification. LSTM-based architectures have been applied to many NLP tasks, including sentiment classification (Wang et al., 2016) and text classification (Zhou et al., 2016).

First, an LSTM encoder computes a fixed-size representation $\mathbf{r}(x)$ from a sequence of tokens $x = (x_1, x_2, \ldots, x_n)$ projected to an embedding space. We use $\boldsymbol{\theta}_r$ to denote the parameters used to construct $\mathbf{r}$. They include the parameters of the LSTM, as well as the word embeddings. Then, the encoder output $\mathbf{r}(x)$ is fed as input to a feedforward network with parameters $\boldsymbol{\theta}_p$ that predicts the label y of the text, with a softmax output activation. In the standard setting, the model is trained to minimize the negative log-likelihood of y labels:

$$\mathcal{L}_m(\boldsymbol{\theta}_r, \boldsymbol{\theta}_p) = \sum_{i=1}^{N} - \log P(y^{(i)}|x^{(i)}; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p),$$

where N is the number of training examples.

2.2 Attacker's Classifier

Once the main model has been trained, we assume that its parameters $\boldsymbol{\theta}_r$ and $\boldsymbol{\theta}_p$ are fixed. We generate a new dataset made of pairs $(\mathbf{r}(x), \mathbf{z}(x))$, where $\mathbf{r}(x)$ is the hidden representation used by the main model and $\mathbf{z}(x)$ is a vector of private categorical variables. In practice, $\mathbf{z}$ is a vector of binary variables, (representing e.g. demographic information about the author). In our experiments, we use the same training examples x for the main

---

[1] The source code used for the experiments described in this paper is available at `https://github.com/mcoavoux/pnet`.

classifier and the classifier of the attacker. However, since the attacker has access to the representation function $\mathbf{r}$ parameterized by $\boldsymbol{\theta}_r$, they can generate a dataset from any corpus containing the private variables they want to recover. In other words, it is not necessary that they have access to the original training corpus to train their classifier.

The attacker trains a second feedforward network on the new dataset $\{(\mathbf{r}(x^{(i)}), \mathbf{z}^{(i)})\}_{i \leq N}$. This classifier uses a sigmoid output activation to compute the probabilities of each binary variable in $\mathbf{z}$:

$$P(\mathbf{z}|\mathbf{r}(x); \boldsymbol{\theta}_a) = \sigma(\text{FeedForward}(\mathbf{r}(x))).$$

It is trained to minimize the negative log-likelihood of $\mathbf{z}$:

$$\begin{aligned}\mathcal{L}_a(\boldsymbol{\theta}_a) &= \sum_{i=1}^{N} -\log P(\mathbf{z}^{(i)}|\mathbf{r}(x^{(i)}); \boldsymbol{\theta}_a) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{K} -\log P(\mathbf{z}_j^{(i)}|\mathbf{r}(x^{(i)}); \boldsymbol{\theta}_a),\end{aligned}$$

assuming that the K variables in $\mathbf{z}$ are independent. Since the parameters used to construct $\mathbf{r}$ are fixed, the attacker only acts upon its own parameters $\boldsymbol{\theta}_a$ to optimize this loss.

We use the performance of the attacker's classifier as a proxy for privacy. If its accuracy is high, then an eavesdropper can easily recover information about the input document. In contrast, if its accuracy is low (i.e. close to that of a most-frequent label baseline), then we may reasonably conclude that $\mathbf{r}$ does not encode enough information to reconstruct x, and mainly contains information that is useful to predict y.

In general, the performance of a single attacker does not provide sufficient evidence to conclude that the input representation $\mathbf{r}$ is robust to an attack. It should be robust to any type of reconstruction method. In the scope of this paper though, we only experiment with a feedforward network reconstructor, i.e. a powerful learner.

In the following sections, we propose several training method modifications aimed at obfuscating private information from the hidden representation $\mathbf{r}(x)$. Intuitively, the aim of these modifications is to minimize some measure of information between $\mathbf{r}$ and $\mathbf{z}$ to make the prediction of $\mathbf{z}$ hard. An obvious choice for that measure would be the Mutual Information (MI) between $\mathbf{r}$ and $\mathbf{z}$. However, MI is hard to compute due to the continuous distribution of $\mathbf{r}$ and does not lend itself well to stochastic optimization.

3 Defenses Against Adversarial Attacks

In this section, we present three training methods designed as defenses against the type of attack we described in Section 2.2. The first two methods are based on two neural networks with rival objective functions (Section 3.1). The last method is meant at discouraging the model to cluster together training examples with similar private variables $\mathbf{z}$ (Section 3.2).

3.1 Adversarial Training

First, we propose to frame the training of the main classifier as a two-agent process: the main agent and an adversarial generator, exploiting a setting similar to Generative Adversarial Networks (GAN, Goodfellow et al., 2014). The generator learns to reconstruct examples from the hidden representation, whereas the main agent learns (i) to perform its main task (ii) to make the task of the generator difficult.

We experiment with two types of generators: a classifier that predicts the binary attributes $\mathbf{z}(x)$ used as a proxy for the reconstruction of x (Section 3.1.1) and a character-based language model that directly optimizes the likelihood of the training examples (Section 3.1.2).

3.1.1 Adversarial Classification: Multidetasking

In order not to make $\mathbf{r}(x)$ a good representation for reconstructing $\mathbf{z}$, we make two modifications to the training setup of the main model (Phase 1):

- We use a duplicate adversarial classifier, with parameters $\boldsymbol{\theta}'_a$, that tries to predict $\mathbf{z}$ from $\mathbf{r}(x)$. It is trained simultaneously with the main classifier. Its training examples are generated on the fly, and change overtime as the main classifier updates its own parameters. This classifier simulates an attack during training.
- We modify the objective function of the main classifier to incorporate a penalty when the adversarial classifier is good at reconstructing $\mathbf{z}$. In other words, the main classifier tries to update its parameters so as to confuse the duplicate attacker.

Formally, for a single data point $(x, y, \mathbf{z})$, the adversarial classifier optimizes:

$$\mathcal{L}_{a'}(x, y, \mathbf{z}; \boldsymbol{\theta}'_a)_= -\log P(\mathbf{z}|\mathbf{r}(x); \boldsymbol{\theta}'_a),$$

whereas the main classifier optimizes:

$$\begin{aligned}\mathcal{L}_m(x, y, \mathbf{z}; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) = & -\alpha \log P(y|x; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) \\ & -\beta \log P(\neg\mathbf{z}|\mathbf{r}(x); \boldsymbol{\theta}'_a).\end{aligned}$$

The first term of this equation is the log-likelihood of the y labels. The second term is designed to deceive the adversary. The hyperparameters $\alpha > 0$ and $\beta > 0$ control the relative importance of both terms.

As in a GAN, the losses of both classifiers are interdependent, but their parameters are distinct: the adversary can only update $\boldsymbol{\theta}'_a$ and the main classifier can only update $\boldsymbol{\theta}_r$ and $\boldsymbol{\theta}_p$.

The duplicate adversarial classifier is identical to the classifier used to evaluate privacy after the main model has been trained and its parameters are fixed. However, both classifiers are completely distinct: the former is used during the training of the main model (Phase 1) to take privacy into account whereas the latter is used to evaluate the privacy of the final model (Phase 3), as is described in Section 2.

3.1.2 Adversarial Generation

The second type of generator we use is a character-based LSTM language model that is trained to reconstruct full training examples. For a single example $(x; y)$, the hidden state of the LSTM is initialized with $\mathbf{r}(x)$, computed by the main model. The generator optimizes:

$$\begin{aligned}\mathcal{L}_g(x, y; \boldsymbol{\theta}_\ell; \boldsymbol{\theta}_r) &= -\log P(x|\mathbf{r}(x); \boldsymbol{\theta}_\ell) \\ &= -\sum_{i=1}^{C} \log P(x_i|x_1^{i-1}, \mathbf{r}(x); \boldsymbol{\theta}_\ell),\end{aligned}$$

where $\boldsymbol{\theta}_\ell$ is the set of parameters of the LSTM generator, x_i is the i^{th} character in the document, and C is the length of the document in number of characters. The generator has no control over $\mathbf{r}(x)$, and optimizes the objective only by updating its own parameters $\boldsymbol{\theta}_\ell$.

Conversely, the loss of the main model is modified as follows:

$$\begin{aligned}\mathcal{L}_m(x, y; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) = & -\alpha \log P(y|x; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) \\ & -\beta \mathcal{L}_g(x, y; \boldsymbol{\theta}_\ell, \boldsymbol{\theta}_r).\end{aligned}$$

The first term maximizes the likelihood of the y labels whereas the second term is meant at making the reconstruction difficult by maximizing the loss of the generator. As in the loss function described in the previous section, α and β control the relative importance of both terms. Once again, the main classifier can optimize the second term only by updating $\boldsymbol{\theta}_r$, since it has no control over the parameters of the adversarial generator.

A key property of this defense method is that it has no awareness of what the private variables $\mathbf{z}$ are. Therefore, it has the potential to protect the neural representation against an attack on any private information. From a broader perspective, the goal of this defense method is to specialize the hidden representation $\mathbf{r}(x)$ to the task at hand (sentiment or topic prediction) and to avoid learning anything not relevant to it.

3.2 Declustering

The last strategy we employ to make the task of the attacker harder is based on the intuition that private variables $\mathbf{z}$ are easier to predict from $\mathbf{r}$ when the main model learns implicitly to cluster examples with similar $\mathbf{z}$ in the same regions of the representation space.

In order to avoid such implicit clustering, we add a term to the training objective of the main model that penalizes pairs of examples (x, x') that (i) have similar reconstructions $\mathbf{z}(x) \approx \mathbf{z}(x')$ (ii) have hidden representations $\mathbf{r}(x)$ and $\mathbf{r}(x')$ in the same region of space. We use the following modified loss for a single example:

$$\begin{aligned}\mathcal{L}_m(x, y, \mathbf{z}; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) &= -\log P(y|x; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) \\ &+\alpha(0.5 - \ell(\mathbf{z}, \mathbf{z}'))||\mathbf{r}(x) - \mathbf{r}(x')||_2^2,\end{aligned}$$

where $(x', \mathbf{z}')$ is another example sampled uniformly from the training set, α is a hyperparameter controlling the importance of the second term, and $\ell(\cdot, \cdot) \in [0, 1]$ is the normalized Hamming distance.

4 Experiments

Our experiments are meant to characterize the privacy-utility tradeoff of neural representations on text classification tasks, and evaluating if the proposed defense methods have a positive impact on it. We first describe the datasets we used (Section 4.1) and the experimental protocol (Section 4.2), then we discuss the results (Section 4.3). We found that in the normal training regime, where no defense is taken into account, the adversary can recover private information with higher accuracy than a most frequent class baseline. Furthermore, we found that the de-

Dataset	Train	Dev	Test
TP US	22142	2767	2767
TP Germany	12596	1574	1574
TP Denmark	82193	10274	10274
TP France	9136	1141	1141
TP UK	48647	6080	6080
AG news	11657	1457	1457
DW corpus	5435	1772	1830
Blog posts	5144	642	642

Table 1: Sizes of datasets in number of examples.

fenses we implemented have a positive effect on the accuracy-privacy tradeoff.

4.1 Datasets

We experiment with two text classification tasks: sentiment analysis (Section 4.1.1) and topic classification (Section 4.1.2). The sizes of each dataset are summarized in Table 1.

4.1.1 Sentiment Analysis

We use the Trustpilot dataset (Hovy et al., 2015) for sentiment analysis. This corpus contains reviews associated with a sentiment score on a five point scale, and self-reported information about the users. We use the five subcorpora corresponding to five areas (Denmark, France, Germany, United Kingdom, United States).

We filter examples containing both the birth year and gender of the author of the review and use these variables as the private information. As in previous work on this dataset (Hovy, 2015; Hovy and Søgaard, 2015), we bin the age of the author into two categories (‘under 35’ and ‘over 45’). Finally, we randomly split each subcorpus into a training set (80%), a development set (10%) and a test (10%).

As an additional experimental setting, we use both demographic variables (gender and age) as input to the main model. We do so by adding two additional tokens at the beginning of the input text, one for each variable. It has been shown that those variables can be used to improve text classification (Hovy, 2015). Also, we would like to evaluate whether the attacker’s task is easier when the variables to predict are *explicitly* in the input, compared to when these information are only potentially and implicitly in the input. In other words, this setting simulates the case where private information may be used by the model to improve classification, but should not be exposed too obviously. In the rest of this section, we use RAW to denote the setting where only the raw text is used as input and +DEMO, the setting where the demographic variables are also used as input.

4.1.2 Topic Classification

We perform topic classification on two genres of documents: news articles and blog posts.

News article For topic classification of news article, we use two datasets: the AG news corpus[2] (Del Corso et al., 2005) and the English part of the Deutsche Welle (DW) news corpus (Pappas and Popescu-Belis, 2017).

For the AG corpus, following Zhang et al. (2015), we construct the dataset by extracting documents belonging to the four most frequent topics, and use the concatenation of the ‘title’ and ‘description’ fields as the input to the classifier. We randomly split the corpus into a training set (80%), a development set (10%) and a test set (10%). For the DW dataset, we use the ‘text’ field as input, and the standard split. We kept only documents belonging to the 20 most frequent topics.

The attacker tries to detect which named entities appear in the input text (each coefficient in $\mathbf{z}(x)$ indicates whether a specific named entity occurs in the text). For both datasets, we used the named entity recognition system from the NLTK package (Bird et al., 2009) to associate each example with the list of named entities that occur in it. We select the five most frequent named entities with type ‘person’, and only keep examples containing at least one of these named entities. This filtering is necessary to avoid a very unbalanced dataset (since each selected named entity appears usually in very few articles).

Blog posts We used the blog authorship corpus presented by Schler et al. (2006), a collection of blog posts associated with the age and gender of the authors, as provided by the authors themselves. Since the blog posts have no topic annotation, we ran the LDA algorithm (Blei et al., 2003) on the whole collection (with 10 topics). The LDA outputs a distribution on topics for each blog post. We selected posts with a single dominating topic ($> 80\%$) and discarded the other posts. We binned age into two category (under 20 and over 30). We

---

[2] `http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`

	Baselines				Best adversaries			
	Lower bound (most frequent class)		Upper bound (trained)		+DEMO		RAW	
	Gender	Age	Gender	Age	Gender	Age	Gender	Age
TP (Denmark)	61.6	58.4	70.5	78.0	68.5	75.3	62.0	63.4
TP (France)	61.0	50.1	69.0	63.4	61.0	57.1	61.0	60.6
TP (Germany)	75.2	50.9	75.2	75.2	75.2	60.4	75.2	58.6
TP (UK)	58.8	56.7	70.0	76.3	66.4	63.5	59.9	61.8
TP (US)	63.5	63.7	74.1	74.8	81.3	74.9	64.7	63.9
Blogs	50.0	50.3	65.7	56.1	-	-	63.9	55.8

Table 2: Comparisons between baselines and best adversaries. All metrics reported in this table are accuracies.

used the age and gender of the author as the private variables. These variables have a very unbalanced distribution in the dataset, we randomly select examples to obtain uniform distributions of private variables. Finally, we split the corpus into a training set (80%), a validation set and a test set (10% each).

4.2 Protocol

Evaluation For the main task, we report a single accuracy measure. For measuring the privacy of a representation, we compute the following metrics:

- For demographic variables (sentiment analysis and blog post topic classification): $1 - X$, where X is the average of the accuracy of the attacker on the prediction of gender and age;
- For named entities (news topic classification): $1 - F$, where F is an F-score computed over the set of binary variables in $\mathbf{z}$ that indicate the presence of named entities in the input example.

Training protocol We implemented our model using Dynet (Neubig et al., 2017). The feedforward components (both of the main model and of the attacker) have a single hidden layer of 64 units with a ReLU activation. Word embeddings have 32 units. The LSTM encoder has a single layer of varying sizes, since it is expected that the amount of information that can be learned depends on the size of these representations. We used the Adam optimizer (Kingma and Ba, 2014) with the default learning rate, and 0.2 dropout rate for the LSTM. We used $\alpha = 0.1$ for the declustering method, based on preliminary experiments. For the other defense methods, we used $\alpha = \beta = 1$ and did not experiment with other values.

For each dataset, and each LSTM state dimension ($\{8, 16, 32, 64, 128\}$), we train the main model for 8 epochs (sentiment classification) or 16 epochs (topic classification), and select the model with the best accuracy on the development set. Then, we generate the dataset for the attacker, train the adversarial model for 16 epochs and select the model with the worst privacy on the development set (i.e. the most successful attacker).

It has to be noted that we select the models that implement defenses on their accuracy, rather than their privacy or a combination thereof. In practice, we could also base the selection strategy on a privacy budget: selecting the most accurate model with privacy above a certain threshold.

4.3 Results

This section discusses results for the sentiment analysis task (Section 4.3.1) and the topic classification task (Section 4.3.2).

4.3.1 Sentiment Analysis

How private are neural representations? Before discussing the effect of proposed defense methods, we motivate empirically our approach by showing that adversarial models can recover private information with reasonable accuracy when the attack is targeted towards a model that implements none of the presented defense methods.

To do so, we compare the accuracy of adversarial models to two types of baselines:

- As a lower bound, we use the most frequent class baseline.
- As an upper bound, we trained a classifier that can optimize the hidden representations ($\mathbf{r}$) for the attacker's tasks. In other words, this baseline is trained to predict de-

Corpus	Standard		M-Detask.		A-Gener.		Decl. $\alpha = 0.1$	
	Main	Priv.	Main	Priv.	Main	Priv.	Main	Priv.
Germany	85.1	32.2	-0.6	-0.3	-1.3	**+0.6**	-0.8	**+1.9**
baseline	78.6	36.9						
Denmark	82.6	28.1	-0.2	**+4.4**	-0.1	**+6.0**	-0.3	**+7.6**
baseline	70.4	40.0						
France	75.1	41.1	-0.8	**+0.7**	-1.4	-6.4	-1.5	-18.2
baseline	69.2	44.4						
UK	87.0	39.3	-0.5	**+0.9**	-0.2	**+0.2**	-0.1	**+0.3**
baseline	77.1	42.2						
US	85.0	33.9	-0.1	**+2.6**	-0.2	**+1.8**	**+0.7**	**+2.2**
baseline	79.4	36.4						

Table 3: Results on the test sets of the Trustpilot dataset, +DEMO setting. *Main* is the accuracy on sentiment analysis. *Priv.* is the privacy measure (i.e. the inverse accuracy of the attacker: higher is better, see Section 4.2). The baselines are most-frequent class classifiers. The values reported for the defense methods indicate absolute differences with the standard training regime (no defense implemented) for both metrics.

Corpus	Standard		M-Detask.		A-Gener.		Decl. $\alpha = 0.1$	
	Main	Priv.	Main	Priv.	Main	Priv.	Main	Priv.
Germany	85.5	32.1	**+0.3**	**+0.5**	-0.8	**+0.9**	-1.7	**+2.2**
baseline	78.6	36.9						
Denmark	82.3	37.3	-0.6	**+0.6**	-0.1	-0.3	-0.2	-0.1
baseline	70.4	40.0						
France	72.7	40.6	**+1.8**	-0.1	**+1.9**	-0.4	-0.3	-0.1
baseline	69.2	44.4						
UK	86.9	40.1	-0.2	**+1.0**	-0.0	**+1.2**	-0.0	0.0
baseline	77.1	42.2						
US	84.5	36.1	-1.1	**+0.2**	**+0.5**	**+0.1**	**+0.3**	**+0.5**
baseline	79.4	36.4						

Table 4: Results on the test sets of the Trustpilot dataset, RAW setting. See Section 4.2 and caption of Table 3 for details about the metrics.

Corpus	Standard		M-Detask.		A-Gener.		Decl. $\alpha = 0.1$	
	Main	Priv.	Main	Priv.	Main	Priv.	Main	Priv.
AG news	76.5	33.7	-14.5	**+14.5**	**+0.2**	-7.8	-2.5	**+8.6**
baseline	57.8							
DW news	44.3	78.3	-5.7	**+21.7**	**+5.9**	**+13.1**	-5.4	**+18.4**
baseline	22.1							
Blogs	58.3	40.8	-0.8	**+3.4**	**+1.1**	**+0.9**	-0.2	**+1.2**
baseline	47.8	49.8						

Table 5: Results for topic classification (test sets). See Section 4.2 and caption of Table 3 for details about the metrics.

mographic variables from x, as if it were the main task.

In Table 2, we compare both baselines to the best adversary in the two settings (RAW and +DEMO) among the models trained with no defenses. First of all, we observe that apart from gender on the German dataset, the trained baseline outperforms the most frequent class baseline by a wide margin (8 to 25 absolute difference). Second of all, the attacker is able to outperform the most frequent class baseline overall, even in the RAW setting. In more details, for age, the adversary is well over the baseline in all cases except US. On the other hand, gender seems harder to predict: the adversary outperforms the most frequent class baseline only in the +DEMO setting.

The same pattern is visible for the blog post dataset, also presented in the last line of Table 2: the best adversaries are 14 points over the baseline for gender and 5 points for age, i.e. almost as good as a model that can fine tune the hidden representations.

These results justify our approach, since they demonstrate that hidden representations learn private information about the input, and can be exploited to recover this information with reasonable accuracy.

Effect of defenses We report results for the main task accuracy and the representation privacy in Table 3 for the +DEMO setting and in Table 4 for the RAW setting. Recall that the privacy measure (Priv.) is computed by $1 - X$ where X is the average accuracy of the attacker on gender and age predictions. When this privacy metric is higher, it is more difficult to exploit the hidden representation of the network to recover information about x. The 'Standard' columns contain the accuracy and privacy of the base model described in Section 2. The next columns present the absolute variation in accuracy and privacy for the three defense methods presented in Section 3: Multidetasking, Adversarial Generation, and Declustering. We also report for each corpus the most frequent class baseline for the main task accuracy, and the privacy of the most frequent class baselines on private variables (i.e. the upper bound for privacy).

The three modified training methods designed as defenses have a positive effect on privacy. Despite a model selection based on accuracy, they lead to an improvement in privacy on all datasets, except on the France subcorpus. In most cases, we observe only a small decrease in accuracy, or even an improvement at times (e.g. multidetasking on the Germany dataset, RAW setting), thus improving the tradeoff between the utility and the privacy of the text representations.

4.3.2 Topic Classification

We report results on topic classification in Table 5.

News articles For the news corpora, the privacy metric is based on the F-score on the binary variables $\mathbf{z}$ indicating the presence or absence of a named entity in the text. First of all, we observe that defense methods that explicitly use $\mathbf{z}$ (i.e. multidetasking and declustering), have a very positive effect on privacy, but also a detrimental effect on the main task. We hypothesize that this is due to the strong correlations between the main task labels y and the private information $\mathbf{z}$. As a result, improving the privacy of the neural representations comes at a cost in accuracy.

In contrast, the adversarial generation defense method lead to an improvement in accuracy, that is quite substantial for the DW corpus. We speculate that this is due to the secondary term in the objective function of the main model (Section 3.1.2) that helps avoiding overfitting the main task or learning spurious features.

Blog posts On the blog post dataset, the effects are smaller, which we attribute to the nature of the task of the attacker. The defense methods consistently improve privacy and, in one case, accuracy. The best effects on the tradeoff are achieved with the multidetasking and adversarial generation methods.

5 Discussion

The main result of our experiments is that the defenses we propose improve privacy with usually a small effect, either positive or negative, on accuracy, thus improving the tradeoff between the utility and the privacy of neural representations.

An important direction for future work is the choice of a strategy for model selection. The tradeoff between utility and privacy can be controlled in many ways. For example, the importance of both terms in the loss functions in Section 3.1 can be controlled to favor either privacy or utility. In the scope of this paper, we did not perform thorough hyperparameter tuning, but believe that doing so is important for achieving better results, since the effects of defense method can be more drastic than desired in some cases, as exemplified on the news corpora (Table 5).

Overall, we found that the multidetasking approach lead to the more stable improvements and should be preferred in most cases, since it is also the less computationnally expensive defense. On the other hand, the adversarial generation method does not require the specification of private variables, and thus is a more general approach.

6 Related Work

The deployment of machine learning in both academic and industrial contexts raises concerns about adversarial uses of machine learning, as well as concerns about attacks specifically targeted at these algorithms that often rely on large amounts of data, including personal data.

More generally, the framework of differential privacy (Dwork, 2006) provides privacy guarantees for the problem of releasing information without compromising confidential data, and usually involves adding noise in the released information. It has been applied to the training of deep learning models (Abadi et al., 2016; Papernot et al., 2016; Papernot et al., 2018), and Bayesian topic models (Schein et al., 2018).

The notion of privacy is particularly crucial to NLP, since it deals with textual data, oftentimes user-generated data, that contain a lot of private information. For example, textual data contain a lot of signal about authors (Hovy and Spruit, 2016). and can be leveraged to predict demographic variables (Rosenthal and McKeown, 2011; Preoţiuc-Pietro et al., 2015). Oftentimes, this information is not explicit in the text but latent and related to the usage of various linguistic traits. Our work is based on a stronger hypothesis: this latent information is still present in vectorial representations of texts, even if the representations have not been supervised by these latent variables.

Li et al. (2017) study the privacy of unsupervised representations of images, and measures their privacy with the peak signal to noise ratio between an original image and its reconstruction by an attacker. They find a tradeoff between the privacy of the learned representations and the accuracy of an image classification model that uses these representations as inputs. Our setting is complementary since it is applied to NLP tasks, but explores a similar problem in the case of representations learned with a task supervision.

A related problem is the unintended memorization of private data from the training set and has been addressed by Carlini et al. (2018). They tackle this problem in the context of text generation (machine translation, language modelling).

If an attacker has access to e.g. a trained language model, they are likely to be able to generate sentences from the training set, since the language model is trained to assign high probabilities to those sentences. Such memorization is problematic when the training data contains private information and personal data. The experimental setting we explore is different from these works: we assume that the attacker has access to a hidden layer of the network and tries to recover information about an input example that is not in the training set.

In a recent study, Li et al. (2018) proposed a method based on GAN designed to improve the robustness and privacy of neural representations, applied to part-of-speech tagging and sentiment analysis. They use a training scheme with two agents similar to our multidetasking strategy (Section 3.1.1), and found that it made neural representations more robust and accurate. However, they only use a single adversary to alter the training of the main model and to evaluate the privacy of the representations, with the risk of overestimating privacy. In contrast, once the parameters of our main model are fixed, we train a new classifier from scratch to evaluate privacy.

7 Conclusion

We have presented an adversarial scenario and used it to measure the privacy of hidden representations in the context of two NLP tasks: sentiment analysis and topic classification of news article and blog posts. We have shown that in general, it is possible for an attacker to recover private variables with higher than chance accuracy, using only hidden representations. In order to improve the privacy of hidden representations, we have proposed defense methods based on modifications of the training objective of the main model. Empirically, the proposed defenses lead to models with a better privacy.

Acknowledgments

We thank the anonymous reviewers and members of the Cohort for helpful feedback on previous versions of the article. We gratefully acknowledge the support of the European Union under the Horizon 2020 SUMMA project (grant agreement 688139), and the support of Huawei Technologies.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 308–318, New York, NY, USA. ACM.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, 1st edition. O'Reilly Media, Inc.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. 2018. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *CoRR*, abs/1802.08232.

Gianna M. Del Corso, Antonio Gullí, and Francesco Romani. 2005. Ranking a stream of news. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 97–106, New York, NY, USA. ACM.

Cynthia Dwork. 2006. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, volume 4052, pages 1–12, Venice, Italy. Springer Verlag.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

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

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 752–762, Beijing, China. Association for Computational Linguistics.

Dirk Hovy, Anders Johannsen, and Anders Søgaard. 2015. User review sites as a resource for large-scale sociolinguistic studies. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 452–461, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the 53rd Annual Meeting of the Association for*

Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 483–488, Beijing, China. Association for Computational Linguistics.

Dirk Hovy and Shannon L. Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598, Berlin, Germany. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Meng Li, Liangzhen Lai, Naveen Suda, Vikas Chandra, and David Z. Pan. 2017. Privynet: A flexible framework for privacy-preserving deep neural network training with A fine-grained privacy control. *CoRR*, abs/1709.06161.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 25–30, Melbourne, Australia. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. 2016. Semi-supervised knowledge transfer for deep learning from private training data. *CoRR*, abs/1610.05755.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable Private Learning with PATE. *ArXiv e-prints*, abs/1802.08908.

Nikolaos Pappas and Andrei Popescu-Belis. 2017. Multilingual hierarchical attention networks for document classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1015–1025, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015. An analysis of the user occupational class through twitter content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1754–1764, Beijing, China. Association for Computational Linguistics.

Sara Rosenthal and Kathleen McKeown. 2011. Age prediction in blogs: A study of style, content, and online behavior in pre- and post-social media generations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 763–772, Portland, Oregon, USA. Association for Computational Linguistics.

Aaron Schein, Zhiwei Steven Wu, Mingyuan Zhou, and Hanna Wallach. 2018. Locally Private Bayesian Inference for Count Models. *ArXiv e-prints*, abs/1803.08471.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James Pennebaker. 2006. Effects of age and gender on blogging. In *Computational Approaches to Analyzing Weblogs - Papers from the AAAI Spring Symposium, Technical Report*, volume SS-06-03, pages 191–197.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495, Osaka, Japan. The COLING 2016 Organizing Committee.

Adversarial Removal of Demographic Attributes from Text Data

Yanai Elazar[†] and **Yoav Goldberg**[†*]
[†]Computer Science Department, Bar-Ilan University, Israel
[*]Allen Institute for Artificial Intelligence
{yanaiela,yoav.goldberg}@gmail.com

Abstract

Recent advances in Representation Learning and Adversarial Training seem to succeed in removing unwanted features from the learned representation. We show that demographic information of authors is encoded in—and can be recovered from—the intermediate representations learned by text-based neural classifiers. The implication is that decisions of classifiers trained on textual data are not agnostic to—and likely condition on—demographic attributes. When attempting to remove such demographic information using adversarial training, we find that while the adversarial component achieves chance-level development-set accuracy during training, a post-hoc classifier, trained on the encoded sentences from the first part, still manages to reach substantially higher classification accuracies on the same data. This behavior is consistent across several tasks, demographic properties and datasets. We explore several techniques to improve the effectiveness of the adversarial component. Our main conclusion is a cautionary one: do not rely on the adversarial training to achieve invariant representation to sensitive features.

1 Introduction

Consider automated systems that are used for determining credit ratings, setting insurance policy rates, or helping in hiring decisions about individuals. We would like such decisions to not take into account factors such as the gender or the race of the individual, or any other factor which we deem to be irrelevant to the decision. We refer to such irrelevant factors as *protected attributes*. The naive solution of not including protected attributes in the features to a Machine Learning system is insufficient: other features may be highly correlated with—and thus predictive of—the protected attributes (Pedreshi et al., 2008). For example, in Credit Score modeling, text might help in credit score decisions (Ghailan et al., 2016). By using the raw text as is, a discrimination issue might arise, as textual information can be predictive of some demographic factors (Hovy et al., 2015) and author's attributes might correlate with target variables (Zhao et al., 2017).

In this paper we are interested in language-based features. It is well established that textual information can be predictive of age, race, gender, and many other social factors of the author (Koppel et al., 2002; Burger et al., 2011; Nguyen et al., 2013; Weren et al., 2014; Verhoeven and Daelemans, 2014; Rangel et al., 2016; Verhoeven et al., 2016; Blodgett et al., 2016), or even the audience of the text (Voigt et al., 2018).

Thus, any system that incorporates raw text into its decision process is at risk of indirectly conditioning on such signals. Recent advances in representation learning suggest *adversarial training* as a mean to hide the protected attributes from the decision function (Section 2). We perform a series of experiments and show that: (1) Information about race, gender and age is indeed encoded into intermediate representations of neural networks, even when training for seemingly unrelated tasks and the training data is balanced in terms of the protected attributes (Section 4); (2) The adversarial training method is indeed effective for reducing the amount of protected encoded information... (3) ...but in some cases even though the adversarial component seems to be doing a perfect job, a fair amount of protected information still remains, and can be extracted from the encoded representations (Section 5.1).

This suggests that when working with text data it is very easy to condition on sensitive properties by mistake. Even when explicitly using the adversarial training method to remove such properties, one should not blindly trust the adversary, and be careful to ensure the protected attributes are in-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 11–21
Brussels, Belgium, October 31 - November 4, 2018.

deed fully removed. We explore means for improving the effectiveness of the adversarial training procedure (section 5.2).[1]

However, while successful to some extent, none of the methods fully succeed in removing all demographic information. Our main message, then, remains cautionary: **if the goal is to ensure fairness or invariant representation, do not trust adversarial removal of features from text inputs for achieving it.**

2 Learning Setup

We follow a setup in which we have some labeled data D composed of documents $x_1, ..., x_n$ and task labels $y_1, ..., y_n$. We wish to train a classifier f that accurately predicts the main task labels y_i. Each data point x_i is also associated with a *protected attribute* z_i, and we want the decision $y_i = f(x_i)$ to be oblivious to z_i. Following (Ganin and Lempitsky, 2015; Xie et al., 2017), we structure f as an encoder $h(x)$ that maps x into a representation vector $\mathbf{h}_x$, and a classifier $c(h(x))$ that is used for predicting y based on $\mathbf{h}_x$. If $\mathbf{h}_{x_i}$ is not predictive of z_i, then the main task prediction $f(x_i) = c(h(x_i))$ does not depend on z_i.

We say that a protected attribute z has *leaked* if we can train a classifier $c'(\mathbf{h}_{x_i})$ to predict z_i with an accuracy beyond chance level, and that the protected attribute is *guarded* if we cannot train such a classifier. We say that a classifier $f(x) = c(h(x))$ is *guarded* if z is guarded, and that it is *leaky* with respect to z if z leaked.

Adversarial Training In order to make f oblivious to z, we follow the adversarial training setup (Goodfellow et al., 2014; Ganin and Lempitsky, 2015; Beutel et al., 2017; Xie et al., 2017). During training, an adversarial classifier $adv(\mathbf{h}_x)$ is trained to predict z, while the encoder h is trained to make adv fail. Concretely, the training procedure tries to jointly optimize both quantities:

$$\begin{aligned} &\arg\min_{adv} && L(adv(h(x_i)), z_i) \\ &\arg\min_{h,c} && L(c(h(x_i)), y_i) - L(adv(h(x_i)), z_i) \end{aligned}$$

where $L(y', y)$ is the loss function (in our case, cross entropy). This objective results in creating the representation $\mathbf{h}_x$ s.t. it's maximally informative for the main task, while at the same time minimally informative of the protected attribute. The optimization is performed in practice using the gradient-reversal layer (GRL) method (Ganin and Lempitsky, 2015). The GRL is a layer g_λ that is inserted between the encoded vector $\mathbf{h}_x$ and the adversarial classifier adv. During the forward pass the layer acts as the identity, while during backpropagation it scales the gradients passed through it by $-\lambda$, causing the encoder to receive the opposite gradients from the adversary. The metaparameter λ controls the intensity of the reversal layer. This results in the objective:

$$\arg\min_{h,c,adv} L(c(h(x_i)), y_i) + L(adv(g_\lambda(h(x_i))), z_i)$$

Attacker Network To test the effectiveness of the adversarial training, we use an *attacker network* $att(\mathbf{h}_x)$. After the classifier $c(h(x))$ is fully trained, we use the encoder to obtain representations $\mathbf{h}$, and train the attacker network to predict z based on $\mathbf{h}$, without access to the encoder or to the original inputs x that resulted in $\mathbf{h}$. If, after training, the attacker can predict z on unseen examples with an accuracy of beyond chance level, then the attribute z leaked to the representation, and the classifier is *not guarded.*

Network Architecture In our setup, an example x_i is a sequence of tokens $w_1, ..., w_{m_i}$ and the encoder is a one layer LSTM network that reads in the associated embedding vectors and returns the final state: $\mathbf{h} = LSTM(\mathbf{w}_{1:m})$. The classifier c and the adversarial adv are both multi-layer perceptrons with one hidden layer, sharing the same hidden layer size and activation function (tanh).[2]

3 Data, Tasks, and Protected Attributes

To perform our experiments, we need a reasonably large dataset in which the data-points x contain textual information, and for which we have both main-task labels y and protected attribute labels z. While our motivating example used prediction tasks for credit rating, insurance rates or hiring decisions, to the best of our knowledge there are no publicly available datasets for these sensitive tasks that meet our criteria. We thus opted to use much less sensitive main-tasks, for which we can obtain the needed data. We focus on Twitter messages, and our protected attributes are binary-race (non-hispanic Whites vs. non-hispanic Blacks),

---

[1] The code and data acquisition are available in: `https://github.com/yanaiela/demog-text-removal`

[2] Further details regarding the architecture and training parameters can be found in the supplementary materials.

binary-gender (Male vs. Female)[3] and binary-age (18-34 vs. 35+). As main tasks we chose *binary emoji-based sentiment prediction* and *binary tweet-mention prediction*. Both the sentiment and the mention prediction tasks are not inherently correlated with race, gender or age. Protected attributes leakage in these seemingly benign main-tasks is a strong indicator that such leakage is likely to occur also in more sensitive tasks.

Main Tasks: Sentiment and Mention-detection Both tasks can be derived automatically from twitter data. We construct a binary "sentiment" task by identifying a subset of emojis which are associated with positive and negative sentiment,[4] identifying tweets containing these emojis, assigning them with the corresponding sentiment and removing the emojis. Tweets containing emojis from both sentiment lists are discarded. The binary mention task is to determine if a tweet mentions another user, i.e, classifying conversational vs. non-conversational tweets. We derive this dataset by identifying tweets that include @mentions tokens, and removing all such tokens from the tweets.

Protected: Race The race annotation is based on the dialectal tweets (DIAL) corpus from (Blodgett et al., 2016), consisting of 59.2 million tweets by 2.8 million users. Each tweet is associated with predicted "race" information which was predicted using a technique that takes into account the geolocation of the author and the words in the tweet. We focus on the AAE (African-American English) and SAE (Standard American English) categories, which we use as proxies for non-Hispanic blacks and non-Hispanic whites.

We chose only annotations with confidence (the probability of the authors' race) of above 80%. Due to its construction, the race annotations in this dataset are highly correlated with the language being used. As such, the data reflects an extreme case in which the underlying language is very predictive of the protected attribute.

Protected: Age and Gender We use data from the PAN16 dataset (Rangel et al., 2016), containing manually annotated Age and Gender information of 436 Twitter users, along with up to 1k tweets for each user. User annotation was performed by consulting the user's LinkedIn profile. Gender was determined by considering the user's name and photograph, discarding unclear cases. Age range was determined by birth-date which was published on the user's profile, or by mapping their degree starting date.

Data-splits From the DIAL corpus we extracted 166K and 10K tweets for training and development purpose respectively (after cleaning and extracting relevant tweets), whereas for the PAN16 dataset we collected 160K tweets for training and 10K for development. The train/development split in both phases of the training (*task-training* and *attacker-training*) is the same. This is the worst possible scenario for the attacker, as it is training on the exact representations the adversary attempted to remove the protected attribute from. Each split is balanced with respect to both the main and the protected labels: a random prediction of each variable is likely to result in 50% accuracy.

Metrics Throughout this paper, we measure leakage using accuracy. We say that the protected attribute has leaked if an *attacker* manages to predict the protected attribute with better than 50% accuracy, which is always the probability of that attribute ($P(Z) = 0.5$). In Appendix A we relate our metric to more standard fairness metrics, and prove that in our setup a guarded predictor guarantees demographic parity, equality of odds, and equality of opportunity. Note however that we also show empirically that such guarded predictors are very hard to attain in practice.

4 Baselines and Data Leakage

In-dataset Accuracy Upper-bounds We begin by examining how well can we perform on each task (both main-tasks and protected attributes) when training the encoder and classifier directly on that task, without any adversarial component. This provides an upper bound on the protected attribute leakage for the main tasks results. The results in Table 1 indicate that the classifiers achieve reasonable accuracies for the main tasks.[5] For

---

[3] While gender is a non-binary construct, many decisions in the real-world are unfortunately still influenced by hard binary gender categories. We thus consider binary-gender to be a useful approximation in our context.

[4] Complete list is available in Appendix C

[5] While the sentiment score may seem low, we manually verified the erroneous predictions and found out that many of them are indeed ambiguous with respect to sentiment, e.g. sentences like "I can't take Amanda seriously 😂" and "You make me so angry, yet you make me so happy. 😩" which were predicted negative and positive respectively, but their gold label was the opposite.

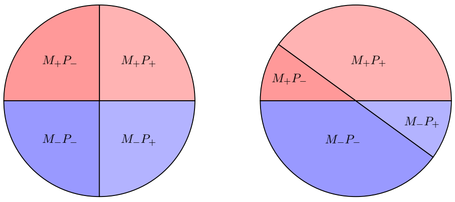

(a) balanced (b) unbalanced

Figure 1: Balanced (a) vs. Unbalanced (b) dataset. **Red(M+)/Blue(M-)**: Main Task. **Light(P+)/Dark(P-)**: Protected attribute. Each class is globally balanced, but in (b) the proportion of the protected attribute within each main task split is unbalanced.

the protected attributes, *race* is highly predictable (83.9%) while *age* and *gender* can also be recovered at above 64% accuracy.

Data	Task	Accuracy
DIAL	Sentiment	67.4
	Mention	81.2
	Race	83.9
PAN16	Mention	77.5
	Gender	67.7
	Age	64.8

Table 1: Accuracies when training directly towards a single task.

Leakage When training directly for the protected attributes, we can recover them with relatively high accuracies. But is information about them being encoded when we train on the main tasks? In this set of experiments, we encode the training and validation sets using the encoder trained on the main task, and train the attacker network to predict the protected attributes based on these vectors. This experiment suggests an upper bound on the amount of leakage of protected attributes when we do not actively attempt to prevent it. The Balanced section in Table 2 summarizes the validation-set accuracies. While the numbers are lower than when training directly (Table 1), they are still high enough to extract meaningful and possibly highly sensitive information (e.g. DIAL Race direct prediction is 83.9% while DIAL Race leakage on the balanced Sentiment task is 64.5%).

Leakage: Unbalanced Data The datasets we considered were perfectly balanced with respect to both main task and protected attribute labels (Figure 1a). Such extreme case is not representative of real-world datasets, in which a dataset may be well balanced w.r.t. the main task labels but not the protected attribute. For example, when training a classifier to predict a fit for managerial position based on Curriculum Vitae (CV) of candidates, the CV dataset may be perfectly balanced according to the managerial / non-managerial variable, but, because of existing social biases, CVs of females might be under-represented in the managerial category and over-represented in the non-managerial one. In such a situation, the classifier may perpetuate the bias by learning to favor males over females for managerial positions. We simulate this more realistic scenario by constructing unbalanced datasets in which the main tasks (sentiment/mention) remain balanced but the protected class proportions within each main class are not, as demonstrated in Figure 1b. For example, in the sentiment/gender case, we set the positive-sentiment class to contain 80% male and 20% female tweets, while the negative-sentiment class contains 20% male and 80% female tweets. We then follow the leakage experiment on the unbalanced datasets. The attacker is trained and tested on a balanced dataset. Otherwise, the attacker can perform quite well on the male/female task simply by learning to predict sentiment, which does not reflect leakage of gender data to the representation. When training the attacker on balanced data, its decisions cannot rely on the sentiment information encoded in the vectors, and must look for encoded information about the protected attributes. The results in Table 2 indicate that both task accuracy and attribute leakage are stronger in the unbalanced case.

Leakage: Real-world Example The above experiments used artificially constructed datasets. Here, we demonstrate leakage using a popular encoder trained for emotion detection: the DeepMoji encoder (Felbo et al., 2017) trained to predict the most suitable emoji usage for a sentence (one of 64 in total), based on 1.2 billion tweets. The model is advertised as a good encoder for encoding sentences into a representation that is highly predictive of sentiment, mood, emotion and sarcasm. Does it also capture *protected attributes*? We encode the sentences of the different *protected attributes* using the DeepMoji encoder and train three different *attackers* to predict race, gender and age. The best scores on the development set are 84.7%, 67.2% and 67.1% respectively. This should not come as a surprise, as indeed some

			Balanced		Unbalanced	
Data	Task	Protected Attribute	Task Acc	Leakage	Task Acc	Leakage
DIAL	Sentiment	Race	67.4	64.5	79.5	73.5
	Mention	Race	81.2	71.5	86.0	73.8
PAN16	Mention	Gender	77.5	60.1	76.8	64.0
		Age	74.7	59.4	77.5	59.7

Table 2: Protected attribute leakage: balanced & unbalanced data splits.

emoji usage is highly correlated with these properties.

5 Mitigating Data Leakage

Leakage of protected attributes information into the internal representation of the network when training on seemingly unrelated tasks is very common. We explore the means of mitigating such leakage.

5.1 Adversarial Training

We repeat the experiments in Table 2 with an adversarial component (Ganin and Lempitsky, 2015) as described in Section 2, in order to actively remove the protected attribute information from the encoded representation during training. Note that the adversarial objective is in odds with the main-task one: by removing the protected attribute information from the encoder, we may also hurt its ability to encode information about the main task.

Figure 2 shows the main task and adversary prediction accuracies on the development set as training progresses, for the Sentiment/Race pair.

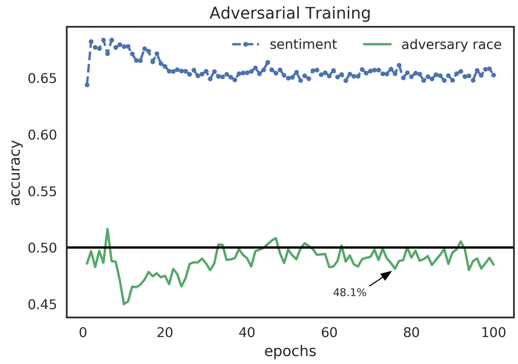

Figure 2: Main task and Adversary accuracy curves for Sentiment/Race.

After an initial peak in task prediction accuracy, the adversary prediction drops and starts to fluctuate around chance level (50%), as desired, along with a drop in main task accuracy as well. The adversary's accuracy remain around chance level throughout the entire training process, suggesting that the adversarial training is indeed effective for removing the protected attribute information from the encoded representation. These trends are persistent for all main-task/protected-attribute pairs we tried.

However, training the attacker network on the resulting encoder vectors reveals a different story. For example, when considering the encoder after 50 training epochs (adversary accuracy of 49.0%), the attacker reaches 56.0% accuracy: substantially higher than the adversarial's success rate, despite sharing the exact same architecture, and being trained and tested on the exact same dataset.

Table 3 summarizes the attacker's recovery rate on the adversarialy-trained encoders for the different settings. In all cases, the adversarial's success rate is around 50%, while the attacker's rate is substantially higher. The attacker's rates are lower than in Table 1, indicating the adversarial training is effective in removing *some* of the protected attribute. However, **a substantial amount of information managed to leak past the adversary, despite its seemingly perfect performance**.

Data	Task	Protected Attribute	Task Acc	Leakage	Δ
DIAL	Sentiment	Race	64.7	56.0	5.0
	Mention	Race	81.5	63.1	9.2
PAN16	Mention	Gender	75.6	58.5	8.0
	Mention	Age	72.5	57.3	6.9

Table 3: Performances on different datasets with an adversarial training. Δ is the difference between the attacker score and the corresponding adversary's accuracy.

Attacker's Accuracy on Unseen Data We verify that the attacker's recovery accuracy persists also on the more realistic scenario in which the attacker is applied to encoded sentences that did not participate in the adversarial training. We constructed an additional dataset of 166K completely unseen samples from the Sentiment/Race case. As expected, the attacker works even better in this case, reaching an accuracy of 59.7% Vs. 56.0% on the original development set.

5.2 Strengthening the Adversarial Component

We explore means of strengthening the adversarial component, by tuning its capacity and its weight, as well as by using a novel adversarial-ensemble configuration.

Capacity We increase the capacity of the adversarial component by increasing its hidden dimension, while keeping the attacker's hidden dimension constant at 300 dimensions. We try hidden dimensions of size 500, 1000, 2000, 5000 and 8000.

Weight We experiment with different weighting of the adversarial component during training by tuning the λ parameter, trying the values 0.5, 1.0 (default), 1.5, 2, 3, 5 (with values above 5 the main task training became extremely unstable, not raising above 50%).

Ensemble An alternative to using larger λ values is to introduce several adversaries. The potential benefit of this approach is that rather than focusing harder on removing a single feature, here the different adversaries could each focus on a different aspect of the representation. This approach is potentially better suited to deal with language variability. Concretely, we suggest the following adaptation to the adversarial loss to incorporate k adversaries with different random initializations:

$$L_y(c(h(x)), y) + \sum_{j=1}^{k} L_z(adv_j(g_\lambda(h(x))), z)$$

Other Attempts We also experienced with several other techniques: reinitializing the adversarial weights every t epochs; training the adversary without propagating the error to the encoder components for t epochs and only then starting to propagate; using adversaries with more hidden layers; adding dropout on the encoded vectors and within the encoder. None of these yielded improvements over the above methods.

Results All methods are effective to some extent, Table 4 summarizes the results.

Increasing the capacity of the adversarial network helped reduce the protected attribute's leakage, though different capacities work best on each setup. On the Sentiment/Race task, none of the higher dimensional adversaries worked better than the 300-dim one, on the PAN16 dataset it did. On PAN16/Gender the 8000-dim adversary performed best, and on PAN16/Age, the 500-dim one.

Increasing the weight of the adversary through the λ parameter also has a positive effect on the result (except on the Sentiment/Race pair). However, too large λ values make training unstable, and require many more epochs for the main-task to stabilize around a satisfying accuracy.

The adversarial ensemble method with 2 adversaries achieves 57.4% on Sentiment/Race, as opposed to 56.0% with a single one, but when using 5 different adversaries, we achieve 54.8%. On the PAN16 dataset larger ensembles are more effective. However, a potential issue with the ensemble method is that larger ensembles reduces training stability, similar to increasing the λ value. For example, with 5 adversaries, the main-task accuracy remained at random for 5 epochs, and only begun rising at the 6th epoch. Using 10 adversaries, the main task could not be trained.

To summarize, while all methods are effective to some extent, it appears that (a) no method and parameter setting performs equally well across the different setups; and (b) no method succeeds in completely preventing the leakage of the protected attributes. Combining the different methods (ensembles of larger networks, larger networks with larger λ, etc.) did not improve the results.

Unbalanced Data Results We repeated the same set of experiments on the unbalanced Sentiment/Race corpus (Table 5). In this setup, the results are somewhat similar: increasing the adversarial capacity and λ is ineffective, and even increases the attacker's recovery rate. However, using an ensemble of 5 adversaries does manage to reduce the leakage, but it is still far from a satisfying result.

6 Analysis

The gap between the adversary's dev-set accuracy and the after-the-fact attacker accuracy on the same data is surprising. To better understand the phenomenon, we perform further analysis on the Sentiment/Race pair with the default single adversary.

Embedding Vs. RNN Recall that the *attacker network* tries to extract as much information from

		DIAL			PAN16					
Method	Parameter	Sentiment	Race	Δ	Mention	Gender	Δ	Mention	Age	Δ
No Adversary Baseline	-	67.4	14.5	-	77.5	10.1	-	74.7	9.4	-
Standard Adversary	(300/1.0/1)	64.7	6.0	5.0	75.6	8.5	8.0	72.5	7.3	6.9
Adv-Capacity	500	64.1	6.7	5.2	73.8	8.1	6.7	71.4	**4.3**	4.1
	1000	63.4	7.1	4.9	75.2	8.9	7.0	71.6	6.3	4.0
	2000	65.2	8.1	6.9	76.1	6.7	6.4	71.9	6.0	5.7
	5000	63.9	**6.2**	3.7	74.5	5.6	1.6	73.0	10.2	9.6
	8000	65.0	7.1	4.8	75.7	**5.4**	4.2	71.9	9.8	7.3
λ	0.5	63.9	**6.8**	6.2	75.6	7.8	6.8	73.1	4.8	3.4
	1.5	64.9	7.4	5.4	75.6	**4.9**	2.4	72.5	6.8	5.8
	2.0	64.2	7.3	5.9	76.0	-7.2	6.7	72.1	8.5	7.7
	3.0	65.8	10.2	10.1	73.7	6.4	6.1	72.5	-6.3	5.2
	5.0	50.0	-	-	73.6	6.5	5.7	69.0	**3.2**	2.9
Ensemble	2	62.4	7.4	5.4	74.8	6.4	5.0	72.8	8.8	8.3
	3	66.5	6.5	5.0	75.3	4.9	3.1	72.1	6.7	6.0
	5	63.8	**4.8**	2.6	74.3	**4.1**	3.0	70.1	**5.7**	5.4

Table 4: Results of different adversarial configurations. **Sentiment/Mention**: main task accuracy. **Race/Gender/Age**: protected attribute recovery difference from 50% rate by the attacker (values below 50% are as informative as those above it). **Δ**: the difference between the attacker score and the corresponding adversary's accuracy. The bold numbers are the best *oblivious* classifiers within each configuration.

Method	Param	Sentiment	Race
No Adversary Baseline	-	79.5	23.5
Standard Adversary	1.0	76.8	10.6
Adv-Capacity	500	74.8	**13.8**
	1000	70.5	18.4
	2000	73.9	18.5
	5000	71.5	19.4
	8000	73.6	18.7
Lambda	0.5	75.0	15.5
	1.5	71.2	18.2
	2.0	73.0	12.1
	3.0	71.5	**12.0**
	5.0	50.0	-
Ensemble	2	70.6	20.8
	3	73.6	17.9
	5	71.5	**8.6**

Table 5: Unbalanced Sentiment/Race with the different methods. **Sentiment**: task accuracy. **Race**: Attacker's recovery accuracy beyond 50%.

the encoder's output as possible. The encoder consists of two components: (1) Embedding Matrix and (2) an RNN. Therefore, the leakage can be caused due to one of them (or due to their combination).

We conduct the following experiment to determine which part affects the leakage more: we create a new encoder by composing 2 existing encoders: an encoder with high leakage (*Leaky*, using the baseline encoder) and an encoder with low leakage (*Guarded*, using the 5-Ensemble adversary). We fuse the two encoders by combining the embedding matrix of the *Leaky* encoder with the RNN module of the *Guarded* encoder, and vice versa. This yields two new encoders: an encoder with a "leaky" Embedding Matrix module and a "strong" RNN module (*Leaky-EMB*), and an encoder with a "strong" Embedding Matrix module and a "leaky" RNN module (*Leaky-RNN*). We compare encoders *Leaky-EMB* and *Leaky-RNN* to gauge which module has a greater contribution to the data leakage. We train attacker-networks over the encoders' output to predict the protected attributes.

		Embedding	
		Leaky	Guarded
RNN	Leaky	64.5	67.8
	Guarded	59.3	54.8

Table 6: Accuracies of the protected attribute with different encoders.

Table 6 summarize the results, implying that the leakage is caused mainly by the RNN, and less by the Embedding Matrix.[6]

---

[6] A discrepancy exists to some extent in the new encoders, as their parts originate from different models that were trained separately. To test if the fusion is valid, we train a different classifier on top of the new encoders to predict the main task. The combination of the leaked RNN with the guarded embeddings results in 65.4% on the sentiment task and the other combination results in 60.9% as opposed to 67.5% and 63.8% on the leaked and guarded models, respectively. As the new models are on par with the original ones, we conclude that the new encoders are valid.

Consistent Leakage: Examples Inspection We are interested in tweets whose protected attribute (race) is correctly predicted by the adversary. However, at accuracy rates below 60%, many of the correct predictions could be attributed to chance. To identify the relevant examples, we repeated the Sentiment/Race default adversary experiment 10 times with different random seeds. We then trained 10 attacker networks, and used each of them to label all examples in the development set. We then looked for tweets which are consistently and correctly classified by at least 9 attackers.[7] Table 7 shows some of these cases. Many of them include tokens (*Naw*, *Bestfrand*, *tan*) and syntactic structures (*Going over Bae house*) which are indeed predictive, though not the most salient features.

Leakage via Embeddings Even though we found out the RNN is much more responsible to the leakage then the Embedding, those still contribute to the leakage and are easier to inspect. Therefore, we turn to inspect the encoders' Embedding. We hypothesize that a possible reason for the adversarial network's inability to completely remove the protected race information is word frequency. Namely, rare words, which might be strongly identified with one group, didn't get enough updates during training and therefore remained predictive towards one of the groups. To quantify this, we compared two vocabularies: words appearing in tweets where the predictions were consistently predicted (9 or 10 out of 10 times) by the different attackers, and words appearing in tweets that were randomly distributed (50%) between the attackers. If our hypothesis is correct, we expect words from the second group to be more frequent than words in the first group. We discard words appearing in both groups, and associate each word with its training set frequency. One-tailed Mann-Whitney U test (Mann and Whitney, 1947) showed the effect is highly significant with $p < e^{-12}$.

Data Overfitting? Standard ML setups often suffer from overfitting on the training data, especially when using neural-networks which tend to memorize the data they encounter. In the adversarial setup, the overfitting could result in the encoder-adversary pair working together to perfectly clean the attributes from the training data, without generalization. Such overfitting could explain the attacker success. Is this what happened? We test this hypothesis by using the same *attacker networks* experiments solely on the training data. We train the *attackers* on 90% of the training data while using the rest 10% as held-out. If overfitting has occurred, the accuracy is likely to result in 50% accuracy. Alas, this is not the case. Table 8 summarize the training accuracies of the *attacker network*. The Mention/Race task achieves the highest score of 64.3% whereas the Mention/Gender task achieves the lowest - 58.1%. Even though when trained directly to predict these attributes without the adversarial setup, the training accuracies are much higher, a substantial amount of signal is still left, even in the training data.

7 Related Work

The fact that intermediary vector representations that are trained for one task are predictive of another is not surprising: it is at the core of the success of NLP methods for deriving "generic" word and sentence representations (e.g. Word2vec (Mikolov et al., 2013), Skip-thought vectors (Kiros et al., 2015), Contextualized Word Representations (Melamud et al., 2016; Peters et al., 2018) etc.). While usually considered a positive feature, it can often have undesired consequences one should be aware of and potentially control for. Several works document biases and stereotypes that are captured by unsupervised word embeddings (Bolukbasi et al., 2016; Caliskan et al., 2017) and ways of mitigating them (Bolukbasi et al., 2016; Zhang et al., 2018). Bias and stereotyping were also documented on a common NLP dataset (Rudinger et al., 2017). While these work are concerned with the learned representations encoding unwanted biases about the world, our concern is with capturing potentially sensitive demographic information about individual authors of the text.

Removing sensitive attributes (demographic or otherwise) from intermediate representations in order to achieve fair classification has been explored by solving an optimization problem (Zemel et al., 2013), as well as by employing adversarial training (Edwards and Storkey, 2015; Louizos et al., 2015; Xie et al., 2017; Zhang et al., 2018), focusing on structured features. Adversarial training was also applied for Image anonymization

---

[7] 776 correct and 946 consistent examples in total

AAE (“non-hispanic blacks”)	SAE (“non-hispanic whites”)
My Brew Eattin	I want to be tan again
_ Naw im cool	Why is it so hot in the house ?!
Tonoght was cool	Been doing Spanish homework for 2 hours .
My momma Bestfrand died	I wish I was still in Spain
Enoy yall day	Ahhhhh so much homework .
Going over Bae house	_TWITTER-ENTITY_ I miss you too !
She not texting or calling ? Ok	I want to move to california
Real relationships go thru real shit	Lol , I don’t even go here .
About to spend my entire check IDGAF	Ahhhhh so much homework .
Getting ready for school	I’m so tired .

Table 7: Examples for correct dialectal/race predictions, which were predicted consistently by at least 9 different attacker-classifiers.

Data	Task	Protected Attribute	Δ
DIAL	Sentiment	Race	12.2
	Mention	Race	14.3
PAN16	Mention	Gender	8.1
	Mention	Age	9.7

Table 8: Attacker’s performance on different datasets. Results are on a training set 10% held-out. Δ is the difference between the attacker score and the corresponding adversary’s accuracy.

(Edwards and Storkey, 2015; Feutry et al., 2018). In contrast, we consider features that are based on short user-authored text.

Several works apply adversarial training to textual data, in order to learn encoders that are invariant to some properties of the text (Chen et al., 2016; Conneau et al., 2017; Zhang et al., 2017; Xie et al., 2017). As their main motivation is to remove information about domain or language in order to improve transfer learning, domain adaptation, or end task accuracy, they were less concerned with the ability to recover information from the resulting representation, and did not evaluate it directly as we do here.

Recent work on creating private representation in the text domain (Li et al., 2018) share our motivation of removing unintended demographic attributes from the learned representation using adversarial training. However, they report only the discrimination accuracies of the adversarial component, and do not train another classifier to verify that the representations are indeed clear of the protected attribute. As our work shows, trusting the adversary is insufficient, and external verification is crucial.

Finally, our work is motivated by the desire for fairness. We use a definition in which a fair classification is one that does not condition on a certain attribute (*fairness by blindness*), and evaluate the ability to achieve text-derived representations that are blind to a property we wish to protect. Many other definitions of fairness exist, including *demographic parity*, *equality of odds* and *equality of opportunity* (see e.g. discussion in (Hardt et al., 2016; Beutel et al., 2017)). Under our setup, blindness guarantees these metrics (Appendix A).

8 Conclusions

We show that demographic information leaks into intermediate representations of neural networks trained on text data. Systems that train on text data and do not want to condition on demographic information must take active steps against accidental conditioning. Our experiments suggest that:
(1) Adversarial training is effective for mitigating protected attribute leakage, but, when dealing with text data, may fail to remove it completely.
(2) When using the adversarial training method, the adversary score during training *cannot be trusted*, and must be verified with an externally-trained attacker, preferably on unseen data.
(3) Tuning the capacity and weight of the adversary, as well as using an ensemble of several adversaries, can improve the results. However, no single method is the most effective in all cases.

Acknowledgments

We would like to thank Moni Shahar, Felix Kreuk, Yova Kementchedjhieva and the BIU NLP lab for fruitful conversation and helpful comments. We also thank Su Lin Blodgett for her help in supplying the DIAL dataset and clarifications. This work was supported in part by the The Israeli Science Foundation (grant number 1555/15) and German Research Foundation via the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. 2017. Data decisions and theoretical implications when adversarially learning fair representations. *arXiv preprint arXiv:1707.00075*.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of african-american english. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1119–1130, Austin, Texas. Association for Computational Linguistics.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

John D Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1301–1309. Association for Computational Linguistics.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Harrison Edwards and Amos Storkey. 2015. Censoring representations with an adversary. *arXiv preprint arXiv:1511.05897*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Clément Feutry, Pablo Piantanida, Yoshua Bengio, and Pierre Duhamel. 2018. Learning anonymized representations with adversarial neural networks. *arXiv preprint arXiv:1802.09386*.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189.

Omar Ghailan, Hoda MO Mokhtar, and Osman Hegazy. 2016. Improving credit scorecard modeling through applying text analysis. *institutions*, 7(4).

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Moritz Hardt, Eric Price, Nati Srebro, et al. 2016. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323.

Dirk Hovy, Anders Johannsen, and Anders Søgaard. 2015. User review sites as a resource for large-scale sociolinguistic studies. In *Proceedings of the 24th International Conference on World Wide Web*, pages 452–461. International World Wide Web Conferences Steering Committee.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Moshe Koppel, Shlomo Argamon, and Anat Rachel Shimoni. 2002. Automatically categorizing written texts by author gender. *Literary and linguistic computing*, 17(4):401–412.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 25–30. Association for Computational Linguistics.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. 2015. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*.

Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Dong Nguyen, Rilana Gravel, Dolf Trieschnigg, and Theo Meder. 2013. " how old do you think i am?" a study of language and age in twitter. In *ICWSM*.

Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. 2008. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 560–568. ACM.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Francisco Rangel, Paolo Rosso, Ben Verhoeven, Walter Daelemans, Martin Potthast, and Benno Stein. 2016. Overview of the 4th author profiling task at pan 2016: cross-genre evaluations. In *Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings/Balog, Krisztian [edit.]; et al.*, pages 750–784.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 74–79.

Ben Verhoeven and Walter Daelemans. 2014. Clips stylometry investigation (csi) corpus: A dutch corpus for the detection of age, gender, personality, sentiment and deception in text. In *LREC 2014-NINTH INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION*, pages 3081–3085.

Ben Verhoeven, Walter Daelemans, and Barbara Plank. 2016. Twisty: A multilingual twitter stylometry corpus for gender and personality profiling. In *LREC*.

Rob Voigt, David Jurgens, Vinodkumar Prabhakaran, Dan Jurafsky, and Yulia Tsvetkov. 2018. Rtgender: A corpus for studying differential responses to gender. In *LREC*.

Edson RD Weren, Anderson U Kauer, Lucas Mizusaki, Viviane P Moreira, J Palazzo M de Oliveira, and Leandro K Wives. 2014. Examining multiple features for author profiling. *Journal of information and data management*, 5(3):266.

Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy, and Graham Neubig. 2017. Controllable invariance through adversarial feature learning. In *Advances in Neural Information Processing Systems*, pages 585–596.

Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. *arXiv preprint arXiv:1801.07593*.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association of Computational Linguistics*, 5(1):515–528.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2989.

DeClarE: Debunking Fake News and False Claims using Evidence-Aware Deep Learning

Kashyap Popat[1], **Subhabrata Mukherjee**[2], **Andrew Yates**[1], **Gerhard Weikum**[1]
[1]Max Planck Institute for Informatics, Saarbrücken, Germany
[2]Amazon Inc., Seattle, USA
`{kpopat,ayates,weikum}@mpi-inf.mpg.de, subhomj@amazon.com`

Abstract

Misinformation such as fake news is one of the big challenges of our society. Research on automated fact-checking has proposed methods based on supervised learning, but these approaches do not consider external evidence apart from labeled training instances. Recent approaches counter this deficit by considering external sources related to a claim. However, these methods require substantial feature modeling and rich lexicons. This paper overcomes these limitations of prior work with an end-to-end model for evidence-aware credibility assessment of arbitrary textual claims, without any human intervention. It presents a neural network model that judiciously aggregates signals from external evidence articles, the language of these articles and the trustworthiness of their sources. It also derives informative features for generating user-comprehensible explanations that makes the neural network predictions transparent to the end-user. Experiments with four datasets and ablation studies show the strength of our method.

1 Introduction

Motivation: Modern media (e.g., news feeds, microblogs, etc.) exhibit an increasing fraction of misleading and manipulative content, from questionable claims and “alternative facts” to completely faked news. The media landscape is becoming a twilight zone and battleground. This societal challenge has led to the rise of fact-checking and debunking websites, such as *Snopes.com* and *PolitiFact.com*, where people research claims, manually assess their credibility, and present their verdict along with evidence (e.g., background articles, quotations, etc.). However, this manual verification is time-consuming. To keep up with the scale and speed at which misinformation spreads, we need tools to automate this debunking process.

State of the Art and Limitations: Prior work on “truth discovery” (see Li et al. (2016) for survey)[1] largely focused on structured facts, typically in the form of subject-predicate-object triples, or on social media platforms like Twitter, Sina Weibo, etc. Recently, methods have been proposed to assess the credibility of claims in natural language form (Popat et al., 2017; Rashkin et al., 2017; Wang, 2017), such as news headlines, quotes from speeches, blog posts, etc.

The methods geared for general text input address the problem in different ways. On the one hand, methods like Rashkin et al. (2017); Wang (2017) train neural networks on labeled claims from sites like *PolitiFact.com*, providing credibility assessments without any explicit feature modeling. However, they use only the text of questionable claims and no external evidence or interactions that provide limited context for credibility analysis. These approaches also do not offer any explanation of their verdicts. On the other hand, Popat et al. (2017) considers external evidence in the form of other articles (retrieved from the Web) that confirm or refute a claim, and jointly assesses the language style (using subjectivity lexicons), the trustworthiness of the sources, and the credibility of the claim. This is achieved via a pipeline of supervised classifiers. On the upside, this method generates user-interpretable explanations by pointing to informative snippets of evidence articles. On the downside, it requires substantial feature modeling and rich lexicons to detect bias and subjectivity in the language style.

Approach and Contribution: To overcome the limitations of the prior works, we present *DeClarE*[2], an end-to-end neural network model for assessing and explaining the credibility of arbi-

---

[1] As fully objective and unarguable truth is often elusive or ill-defined, we use the term *credibility* rather than “truth”.

[2] Debunking Claims with Interpretable Evidence

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 22–32
Brussels, Belgium, October 31 - November 4, 2018.

trary claims in natural-language text form. Our approach combines the best of both families of prior methods. Similar to Popat et al. (2017), DeClarE incorporates external evidence or counter-evidence from the Web as well as signals from the language style and the trustworthiness of the underlying sources. However, our method does not require any feature engineering, lexicons, or other manual intervention. Rashkin et al. (2017); Wang (2017) also develop an end-to-end model, but DeClarE goes far beyond in terms of considering external evidence and joint interactions between several factors, and also in its ability to generate user-interpretable explanations in addition to highly accurate assessments. For example, given the natural-language input claim *"the gun epidemic is the leading cause of death of young African-American men, more than the next nine causes put together"* by Hillary Clinton, DeClarE draws on evidence from the Web to arrive at its verdict *credible*, and returns annotated snippets like the one in Table 6 as explanation. These snippets, which contain evidence in the form of statistics and assertions, are automatically extracted from web articles from sources of varying credibility.

Given an input claim, DeClarE searches for web articles related to the claim. It considers the *context* of the claim via word embeddings and the (language of) web articles captured via a bidirectional LSTM (biLSTM), while using an *attention* mechanism to focus on parts of the articles according to their relevance to the claim. DeClarE then aggregates all the information about claim source, web article contexts, attention weights, and trustworthiness of the underlying sources to assess the claim. It also derives informative features for interpretability, like source embeddings that capture trustworthiness and salient words captured via attention. Key contributions of this paper are:

- **Model:** An end-to-end neural network model which automatically assesses the credibility of natural-language claims, without any hand-crafted features or lexicons.
- **Interpretability:** An *attention* mechanism in our model that generates user-comprehensible explanations, making credibility verdicts transparent and interpretable.
- **Experiments:** Extensive experiments on four datasets and ablation studies, demonstrating effectiveness of our method over state-of-the-art baselines.

2 End-to-end Framework for Credibility Analysis

Consider a set of N claims $\langle C_n \rangle$ from the respective origins/sources $\langle CS_n \rangle$, where $n \in [1, N]$. Each claim C_n is reported by a set of M articles $\langle A_{m,n} \rangle$ along with their respective sources $\langle AS_{m,n} \rangle$, where $m \in [1, M]$. Each corresponding tuple of claim and its origin, reporting articles and article sources – $\langle C_n, CS_n, A_{m,n}, AS_{m,n} \rangle$ forms a training instance in our setting, along with the credibility label of the claim used as ground-truth during network training. Figure 1 gives a pictorial overview of our model. In the following sections, we provide a detailed description of our approach.

2.1 Input Representations

The input claim C_n of length l is represented as $[c_1, c_2, ..., c_l]$ where $c_l \in \Re^d$ is the d-dimensional word embedding of the l-th word in the input claim. The source/origin of the claim CS_n is represented by a d_s-dimensional embedding vector $cs_n \in \Re^{d_s}$.

A reporting article $A_{m,n}$ consisting of k tokens is represented by $[a_{m,n,1}, a_{m,n,2}, ..., a_{m,n,k}]$, where $a_{m,n,k} \in \Re^d$ is the d-dimensional word embedding vector for the k-th word in the reporting article $A_{m,n}$. The claim and article word embeddings have shared parameters. The source of the reporting article $AS_{m,n}$ is represented as a d_s-dimensional vector, $as_{m,n} \in \Re^{d_s}$. For the sake of brevity, we drop the notation subscripts n and m in the following sections by considering only a single training instance – the input claim C_n from source CS_n, the corresponding article $A_{m,n}$ and its sources $AS_{m,n}$ given by: $\langle C, CS, A, AS \rangle$.

2.2 Article Representation

To create a representation of an article, which may capture task-specific features such as whether it contains objective language, we use a bidirectional Long Short-Term Memory (LSTM) network as proposed by Graves et al. (2005). A basic LSTM cell consists of various gates to control the flow of information through timesteps in a sequence, making LSTMs suitable for capturing long and short range dependencies in text that may be difficult to capture with standard recurrent neural networks (RNNs). Given an input word embedding of tokens $\langle a_k \rangle$, an LSTM cell performs various non-linear transformations to generate a hidden vector state h_k for each token at each timestep k.

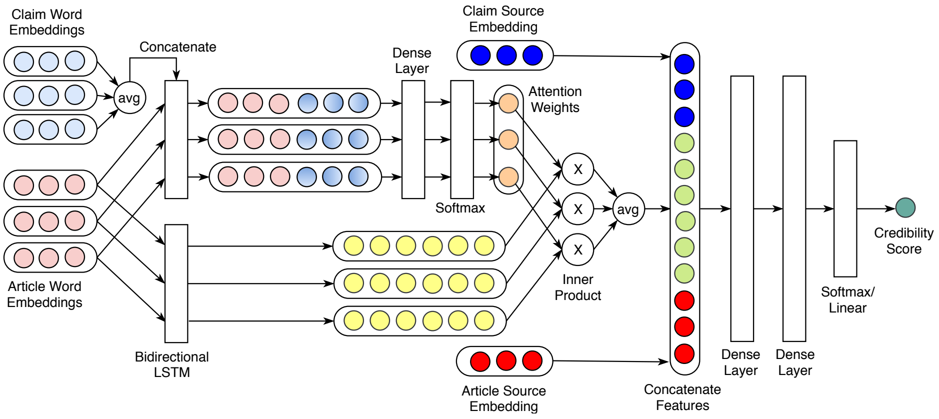

Figure 1: Framework for credibility assessment. Upper part of the pipeline combines the article and claim embeddings to get the claim specific attention weights. Lower part of the pipeline captures the article representation through biLSTM. Attention focused article representation along with the source embeddings are passed through dense layers to predict the credibility score of the claim.

We use bidirectional LSTMs in place of standard LSTMs. Bidirectional LSTMs capture both the previous timesteps (past features) and the future timesteps (future features) via forward and backward states respectively. Correspondingly, there are two hidden states that capture past and future information that are concatenated to form the final output as: $h_k = [\overrightarrow{h_k}, \overleftarrow{h_k}]$.

2.3 Claim Specific Attention

As we previously discussed, it is important to consider the relevance of an article with respect to the claim; specifically, focusing or *attending* to parts of the article that discuss the claim. This is in contrast to prior works (Popat et al., 2017; Rashkin et al., 2017; Wang, 2017) that ignore either the article or the claim, and therefore miss out on this important interaction.

We propose an attention mechanism to help our model focus on salient words in the article with respect to the claim. To this end, we compute the importance of each term in an article with respect to an overall representation of the corresponding claim. Additionally, incorporating attention helps in making our model transparent and interpretable, because it provides a way to generate the most salient words in an article as evidence of our model's verdict.

Following Wieting et al. (2015), the overall representation of an input claim is generated by taking an average of the word embeddings of all the words therein:

$$\bar{c} = \frac{1}{l}\sum_{l} c_l$$

We combine this overall representation of the claim with each article term:

$$\hat{a}_k = a_k \oplus \bar{c}$$

where, $\hat{a}_k \in \Re^{d+d}$ and $\oplus$ denotes the concatenate operation. We then perform a transformation to obtain claim-specific representations of each article term:

$$a'_k = \mathbf{f}(W_a \hat{a}_k + b_a)$$

where W_a and b_a are the corresponding weight matrix and bias terms, and $\mathbf{f}$ is an activation function[3], such as $ReLU$, $tanh$, or the identity function. Following this, we use a softmax activation to calculate an attention score α_k for each word in the article capturing its relevance to the claim context:

$$\alpha_k = \frac{\exp(a'_k)}{\sum_k \exp(a'_k)} \quad (1)$$

2.4 Per-Article Credibility Score of Claim

Now that we have article term representations given by $\langle h_k \rangle$ and their relevance to the claim given by $\langle \alpha_k \rangle$, we need to combine them to predict the claim's credibility. In order to create an

---

[3] In our model, the *tanh* activation function gives best results.

attention-focused representation of the article considering both the claim and the article's language, we calculate a weighted average of the hidden state representations for all article tokens based on their corresponding attention scores:

$$g = \frac{1}{k} \sum_{k} \alpha_k \cdot h_k \tag{2}$$

We then combine all the different feature representations: the claim source embedding (cs), the attention-focused article representation (g), and the article source embedding (as). In order to merge the different representations and capture their joint interactions, we process them with two fully connected layers with non-linear activations.

$$\begin{aligned} d_1 &= relu(W_c(g \oplus cs \oplus as) + b_c) \\ d_2 &= relu(W_d d_1 + b_d) \end{aligned}$$

where, W and b are the corresponding weight matrix and bias terms.

Finally, to generate the overall credibility label of the article for classification tasks, or credibility score for regression tasks, we process the final representation with a final fully connected layer:

$$\text{Classification:} \quad s = sigmoid(d_2) \tag{3}$$

$$\text{Regression:} \quad s = linear(d_2) \tag{4}$$

2.5 Credibility Aggregation

The credibility score in the above step is obtained considering a single reporting article. As previously discussed, we have M reporting articles per claim. Therefore, once we have the per-article credibility scores from our model, we take an average of these scores to generate the overall credibility score for the claim:

$$cred(C) = \frac{1}{M} \sum_{m} s_m \tag{5}$$

This aggregation is done after the model is trained.

3 Datasets

We evaluate our approach and demonstrate its generality by performing experiments on four different datasets: a general fact-checking website, a political fact-checking website, a news review community, and a SemEval Twitter rumour dataset.

3.1 Snopes

Snopes (`www.snopes.com`) is a general fact-checking website where editors manually investigate various kinds of rumors reported on the Internet. We used the Snopes dataset provided by Popat et al. (2017). This dataset consists of rumors analyzed on the Snopes website along with their credibility labels (*true* or *false*), sets of reporting articles, and their respective web sources.

3.2 PolitiFact

PolitiFact is a political fact-checking website (`www.politifact.com`) in which editors rate the credibility of claims made by various political figures in US politics. We extract all articles from PolitiFact published before December 2017. Each article includes a claim, the speaker (political figure) who made the claim, and the claim's credibility rating provided by the editors.

PolitiFact assigns each claim to one of six possible ratings: *true, mostly true, half true, mostly false, false* and *pants-on-fire*. Following Rashkin et al. (2017), we combine *true, mostly true* and *half true* ratings into the class label *true* and the rest as *false* – hence considering only binary credibility labels. To retrieve the reporting articles for each claim (similar to Popat et al. (2017)), we issue each claim as a query to a search engine[4] and retrieve the top 30 search results with their respective web sources.

3.3 NewsTrust

NewsTrust is a news review community in which members review the credibility of news articles. We use the NewsTrust dataset made available by Mukherjee and Weikum (2015). This dataset contains NewsTrust stories from May 2006 to May 2014. Each story consists of a news article along with its source, and a set of reviews and ratings by community members. NewsTrust aggregates these ratings and assigns an overall credibility score (on a scale of 1 to 5) to the posted article. We map the attributes in this data to the inputs expected by DeClarE as follows: the title and the web source of the posted (news) article are mapped to the input claim and claim source, respectively. Reviews and their corresponding user identities are mapped to reporting articles and article sources, respectively. We use this dataset for the regression task of predicting the credibility score of the posted article.

---

[4] We use the Bing search API.

Dataset	SN	PF	NT	SE
Total claims	4341	3568	5344	272
True claims	1164	1867	-	127
False claims	3177	1701	-	50
Unverified claims	-	-	-	95
Claim sources	-	95	161	10
Articles	29242	29556	25128	3717
Article sources	336	336	251	89

Table 1: Data statistics (SN: Snopes, PF: PolitiFact, NT: NewsTrust, SE: SemEval).

3.4 SemEval-2017 Task 8

As the fourth dataset, we consider the benchmark dataset released by SemEval-2017 for the task of determining credibility and stance of social media content (Twitter) (Derczynski et al., 2017). The objective of this task is to predict the credibility of a questionable tweet (*true*, *false* or *unverified*) along with a confidence score from the model. It has two sub-tasks: (i) a *closed* variant in which models only consider the questionable tweet, and (ii) an *open* variant in which models consider both the questionable tweet and additional context consisting of snapshots of relevant sources retrieved immediately before the rumor was reported, a snapshot of an associated Wikipedia article, news articles from digital news outlets, and preceding tweets about the same event. Testing and development datasets provided by organizers have 28 tweets (1021 reply tweets) and 25 tweets (256 reply tweets), respectively.

3.5 Data Processing

In order to have a minimum support for training, claim sources with less than 5 claims in the dataset are grouped into a single dummy claim source, and article sources with less than 10 articles are grouped similarly (5 articles for SemEval as it is a smaller dataset).

For Snopes and PolitiFact, we need to extract relevant snippets from the reporting articles for a claim. Therefore, we extract snippets of 100 words from each reporting article having the maximum relevance score: $sim = sim_{\text{bow}} \times sim_{\text{semantic}}$ where sim_{bow} is the fraction of claim words that are present in the snippet, and sim_{semantic} represents the cosine similarity between the average of claim word embeddings and snippet word embeddings. We also enforce a constraint that the sim score is at least δ. We varied δ from 0.2 to 0.8 and found 0.5 to give the optimal performance on a withheld dataset. We discard all articles related to Snopes and PolitiFact websites from our datasets to have an unbiased model. Statistics of the datasets after pre-processing is provided in Table 1. All the datasets are made publicly available at `https://www.mpi-inf.mpg.de/dl-cred-analysis/`.

Parameter	SN	PF	NT	SE
Word embedding length	100	100	300	100
Claim source embedding length	-	4	8	4
Article source embedding length	8	4	8	4
LSTM size (for each pass)	64	64	64	16
Size of fully connected layers	32	32	64	8
Dropout	0.5	0.5	0.3	0.3

Table 2: Model parameters used for each dataset (SN: Snopes, PF: PolitiFact, NT: NewsTrust, SE: SemEval).

4 Experiments

We evaluate our approach by conducting experiments on four datasets, as described in the previous section. We describe our experimental setup and report our results in the following sections.

4.1 Experimental Setup

When using the Snopes, PolitiFact and NewsTrust datasets, we reserve 10% of the data as validation data for parameter tuning. We report 10-fold cross validation results on the remaining 90% of the data; the model is trained on 9-folds and the remaining fold is used as test data. When using the SemEval dataset, we use the data splits provided by the task's organizers. The objective for Snopes, PolitiFact and SemEval experiments is binary (credibility) classification, while for NewsTrust the objective is to predict the credibility score of the input claim on a scale of 1 to 5 (i.e., credibility regression). We represent terms using pre-trained GloVe Wikipedia 6B word embeddings (Pennington et al., 2014). Since our training datasets are not very large, we do not tune the word embeddings during training. The remaining model parameters are tuned on the validation data; the parameters chosen are reported in Table 2. We use Keras with a Tensorflow backend to implement our system. All the models are trained using Adam optimizer (Kingma and Ba, 2014) (learning rate: 0.002) with categorical cross-entropy loss for classification and mean squared error loss for regression task. We use L2-regularizers with the

Dataset	**Configuration**	***True* Claims Accuracy (%)**	***False* Claims Accuracy (%)**	**Macro F1-Score**	**AUC**
Snopes	LSTM-text	64.65	64.21	0.66	0.70
	CNN-text	67.15	63.14	0.66	0.72
	Distant Supervision	**83.21**	**80.78**	**0.82**	**0.88**
	DeClarE (Plain)	74.37	78.57	0.78	0.83
	DeClarE (Plain+Attn)	78.34	78.91	0.79	0.85
	DeClarE (Plain+SrEmb)	77.43	79.80	0.79	0.85
	DeClarE (Full)	78.96	78.32	0.79	0.86
PolitiFact	LSTM-text	63.19	61.96	0.63	0.66
	CNN-text	63.67	63.31	0.64	0.67
	Distant Supervision	62.53	62.08	0.62	0.68
	DeClarE (Plain)	62.67	69.05	0.66	0.70
	DeClarE (Plain+Attn)	65.53	68.49	0.66	0.72
	DeClarE (Plain+SrEmb)	66.71	69.28	0.67	0.74
	DeClarE (Full)	**67.32**	**69.62**	**0.68**	**0.75**

Table 3: Comparison of various approaches for credibility classification on Snopes and PolitiFact datasets.

fully connected layers as well as dropout. For all the datasets, the model is trained using each claim-article pair as a separate training instance.

To evaluate and compare the performance of DeClarE with other state-of-the-art methods, we report the following measures:

- Credibility Classification (Snopes, PolitiFact and SemEval): accuracy of the models in classifying *true* and *false* claims separately, macro F1-score and Area-Under-Curve (AUC) for the ROC (Receiver Operating Characteristic) curve.
- Credibility Regression (NewsTrust): Mean Square Error (MSE) between the predicted and true credibility scores.

4.2 Results: Snopes and Politifact

We compare our approach with the following state-of-the-art models: (i) LSTM-text, a recent approach proposed by Rashkin et al. (2017). (ii) CNN-text: a CNN based approach proposed by Wang (2017). (iii) Distant Supervision: state-of-the-art distant supervision based approach proposed by Popat et al. (2017). (iv) DeClare (Plain): our approach with only biLSTM (no attention and source embeddings). (v) DeClarE (Plain+Attn): our approach with only biLSTM and attention (no source embeddings). (vi) DeClarE (Plain+SrEmb): our approach with only biLSTM and source embeddings (no attention). (vii) DeClarE (Full): end-to-end system with biLSTM, attention and source embeddings.

The results when performing credibility classification on the Snopes and PolitiFact datasets are shown in Table 3. DeClarE outperforms LSTM-text and CNN-text models by a large margin on both datasets. On the other hand, for the Snopes dataset, performance of DeClarE (Full) is slightly lower than the Distant Supervision configuration (p-value of 0.04 with a pairwise t-test). However, the advantage of DeClarE over Distant Supervision approach is that it does not rely on hand crafted features and lexicons, and can generalize well to arbitrary domains without requiring any seed vocabulary. It is also to be noted that both of these approaches use external evidence in the form of reporting articles discussing the claim, which are not available to the LSTM-text and CNN-text baselines. This demonstrates the value of external evidence for credibility assessment.

On the PolitiFact dataset, DeClarE outperforms all the baseline models by a margin of 7-9% AUC (p-value of $9.12e{-}05$ with a pairwise t-test) with similar improvements in terms of Macro F1. A performance comparison of DeClarE's various configurations indicates the contribution of each component of our model, i.e, biLSTM capturing article representations, attention mechanism and source embeddings. The additions of both the attention mechanism and source embeddings improve performance over the plain configuration in all cases when measured by Macro F1 or AUC.

4.3 Results: NewsTrust

When performing credibility regression on the NewsTrust dataset, we evaluate the models in terms of mean squared error (MSE; lower is better) for credibility rating prediction. We use the

Configuration	**MSE**
CNN-text	0.53
CCRF+SVR	0.36
LSTM-text	0.35
DistantSup	0.35
DeClarE (Plain)	0.34
DeClarE (Full)	**0.29**

Table 4: Comparison of various approaches for credibility regression on NewsTrust dataset.

first three models described in Section 4.2 as baselines. For CNN-text and LSTM-text, we add a linear fully connected layer as the final layer of the model to support regression. Additionally, we also consider the state-of-the-art CCRF+SVR model based on Continuous Conditional Random Field (CCRF) and Support Vector Regression (SVR) proposed by Mukherjee and Weikum (2015). The results are shown in Table 4. We observe that DeClarE (Full) outperforms all four baselines, with a 17% decrease in MSE compared to the best-performing baselines (i.e., LSTM-text and Distant Supervision). The DeClarE (Plain) model performs substantially worse than the full model, illustrating the value of including attention and source embeddings. CNN-text performs substantially worse than the other baselines.

4.4 Results: SemEval

On the SemEval dataset, the objective is to perform credibility classification of a tweet while also producing a classification confidence score. We compare the following approaches and consider both variants of the SemEval task: (i) *NileTMRG* (Enayet and El-Beltagy, 2017): the best performing approach for the *close* variant of the task, (ii) *IITP* (Singh et al., 2017): the best performing approach for the *open* variant of the task, (iii) DeClare (Plain): our approach with only biLSTM (no attention and source embeddings), and (iv) DeClarE (Full): our end-to-end system with biLSTM, attention and source embeddings.

We use the evaluation measure proposed by the task's organizers: macro F1-score for overall classification and Root-Mean-Square Error (RMSE) over confidence scores. Results are shown in Table 5. We observe that DeClarE (Full) outperforms all the other approaches — thereby, re-affirming its power in harnessing external evidence.

Configuration	**Macro Accuracy**	**RMSE**
IITP (Open)	0.39	0.746
NileTMRG (Close)	0.54	0.673
DeClarE (Plain)	0.46	0.687
DeClarE (Full)	**0.57**	**0.604**

Table 5: Comparison of various approaches for credibility classification on SemEval dataset.

5 Discussion

5.1 Analyzing Article Representations

In order to assess how our model separates articles reporting false claims from those reporting true ones, we employ dimensionality reduction using Principal Component Analysis (PCA) to project the article representations (g in Equation 2) from a high dimensional space to a 2d plane. The projections are shown in Figure 2a. We observe that DeClarE obtains clear separability between credible versus non-credible articles in Snopes dataset.

5.2 Analyzing Source Embeddings

Similar to the treatment of article representations, we perform an analysis with the claim and article source embeddings by employing PCA and plotting the projections. We sample a few popular news sources from Snopes and claim sources from PolitiFact. These news sources and claim sources are displayed in Figure 2b and Figure 2c, respectively. From Figure 2b we observe that DeClarE clearly separates fake news sources like *nationalreport*, *empirenews*, *huzlers*, etc. from mainstream news sources like *nytimes*, *cnn*, *wsj*, *foxnews*, *washingtonpost*, etc. Similarly, from Figure 2c we observe that DeClarE locates politicians with similar ideologies and opinions close to each other in the embedding space.

5.3 Analyzing Attention Weights

Attention weights help understand what DeClarE focuses on during learning and how it affects its decisions – thereby, making our model transparent to the end-users. Table 6 illustrates some interesting claims and salient words (highlighted) that DeClarE focused on during learning. Darker shades indicate higher weights given to the corresponding words. As illustrated in the table, DeClarE gives more attention to important words in the reporting article that are relevant to the claim and also

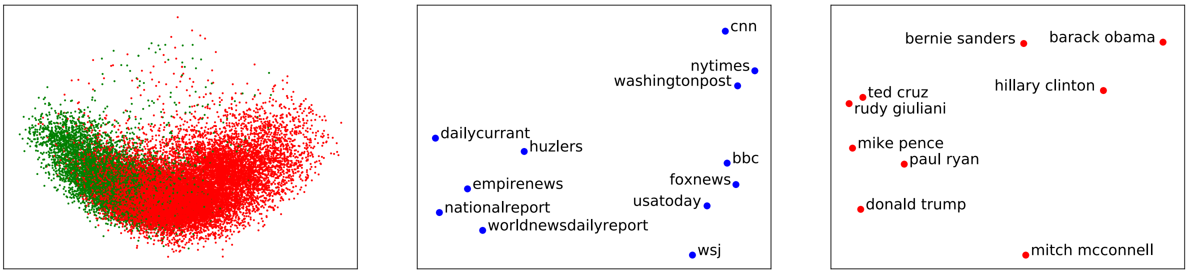

(a) Projections of article representations using PCA; DeClarE obtains clear separation between representations of non-credible articles (*red*) vs. true ones (*green*).

(b) Projections of article source representations using PCA; DeClarE clearly separates fake news sources from authentic ones.

(c) Projections of claim source representations using PCA; DeClarE clusters politicians of similar ideologies close to each other in the embedding space.

Figure 2: Dissecting the article, article source and claim source representations learned by DeClarE.

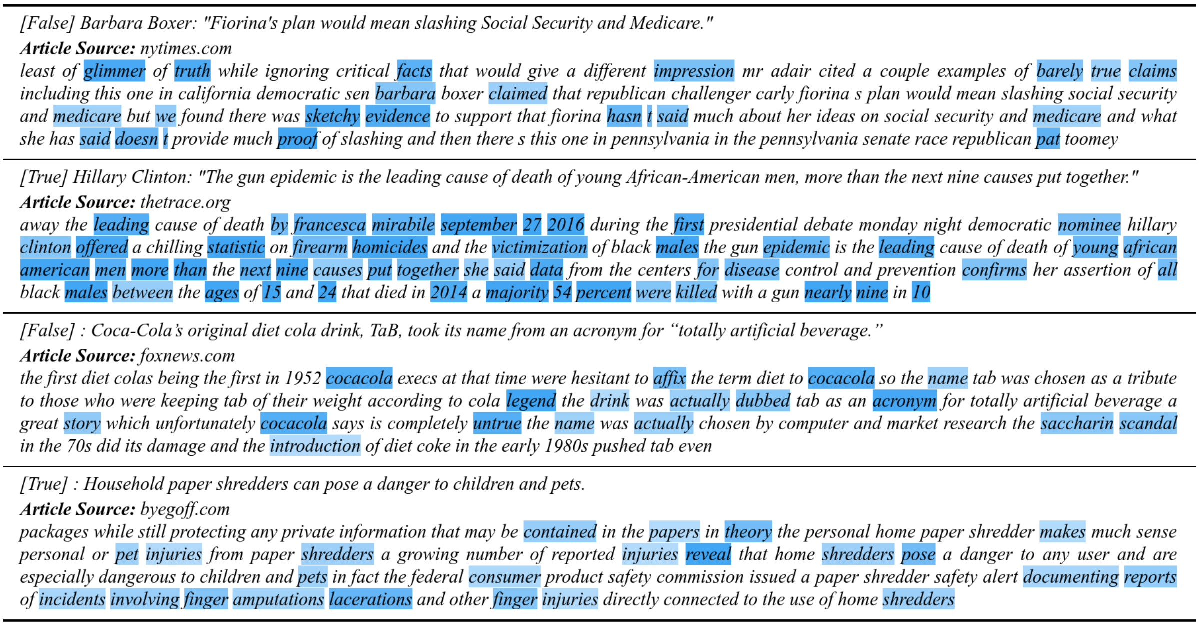

[False] Barbara Boxer: "Fiorina's plan would mean slashing Social Security and Medicare."
Article Source: *nytimes.com*
least of glimmer of truth while ignoring critical facts that would give a different impression mr adair cited a couple examples of barely true claims including this one in california democratic sen barbara boxer claimed that republican challenger carly fiorina s plan would mean slashing social security and medicare but we found there was sketchy evidence to support that fiorina hasn t said much about her ideas on social security and medicare and what she has said doesn t provide much proof of slashing and then there s this one in pennsylvania in the pennsylvania senate race republican pat toomey

[True] Hillary Clinton: "The gun epidemic is the leading cause of death of young African-American men, more than the next nine causes put together."
Article Source: *thetrace.org*
away the leading cause of death by francesca mirabile september 27 2016 during the first presidential debate monday night democratic nominee hillary clinton offered a chilling statistic on firearm homicides and the victimization of black males the gun epidemic is the leading cause of death of young african american men more than the next nine causes put together she said data from the centers for disease control and prevention confirms her assertion of all black males between the ages of 15 and 24 that died in 2014 a majority 54 percent were killed with a gun nearly nine in 10

[False] : Coca-Cola's original diet cola drink, TaB, took its name from an acronym for "totally artificial beverage."
Article Source: *foxnews.com*
the first diet colas being the first in 1952 cocacola execs at that time were hesitant to affix the term diet to cocacola so the name tab was chosen as a tribute to those who were keeping tab of their weight according to cola legend the drink was actually dubbed tab as an acronym for totally artificial beverage a great story which unfortunately cocacola says is completely untrue the name was actually chosen by computer and market research the saccharin scandal in the 70s did its damage and the introduction of diet coke in the early 1980s pushed tab even

[True] : Household paper shredders can pose a danger to children and pets.
Article Source: *byegoff.com*
packages while still protecting any private information that may be contained in the papers in theory the personal home paper shredder makes much sense personal or pet injuries from paper shredders a growing number of reported injuries reveal that home shredders pose a danger to any user and are especially dangerous to children and pets in fact the federal consumer product safety commission issued a paper shredder safety alert documenting reports of incidents involving finger amputations lacerations and other finger injuries directly connected to the use of home shredders

Table 6: Interpretation via attention (weights) ([*True*]/[*False*] indicates the verdict from DeClarE).

play a major role in deciding the corresponding claim's credibility. In the first example on Table 6, highlighted words such as "..*barely true*..." and "..*sketchy evidence*..." help our system to identify the claim as *not credible*. On the other hand, highlighted words in the last example, like, "..*reveal*..." and "..*documenting reports*..." help our system to assess the claim as *credible*.

6 Related Work

Our work is closely related to the following areas:

Credibility analysis of Web claims: Our work builds upon approaches for performing credibility analysis of natural language claims in an open-domain Web setting. The approach proposed in Popat et al. (2016, 2017) employs stylistic language features and the stance of articles to assess the credibility of the natural language claims. However, their model heavily relies on hand-crafted language features. Rashkin et al. (2017); Wang (2017) propose neural network based approaches for determining the credibility of a textual claim, but it does not consider external sources like web evidence and claim sources. These can be important evidence sources for credibility analysis. The method proposed by Samadi et al. (2016) uses the Probabilistic Soft Logic (PSL) framework to estimate source reliability and claim correctness. Vydiswaran et al. (2011) proposes an iterative algorithm which jointly learns the veracity of textual claims and trustworthiness of the sources. These approaches do not consider the deeper semantic aspects of language, however. Wiebe and Riloff (2005); Lin et al. (2011); Recasens et al. (2013) study the problem of detecting bias in language, but do not consider credibility.

Truth discovery: Prior approaches for truth discovery (Yin et al., 2008; Dong et al., 2009, 2015; Li et al., 2011, 2014, 2015; Pasternack and Roth, 2011, 2013; Ma et al., 2015; Zhi et al., 2015; Gao et al., 2015; Lyu et al., 2017) have focused on structured data with the goal of addressing the problem of conflict resolution amongst multi-source data. Nakashole and Mitchell (2014) proposed a method to extract conflicting values from the Web in the form of Subject-Predicate-Object (SPO) triplets and uses language objectivity analysis to determine the true value. Like the other truth discovery approaches, however, this approach is mainly suitable for use with structured data.

Credibility analysis in social media: Mukherjee et al. (2014); Mukherjee and Weikum (2015) propose PGM based approaches to jointly infer a statement's credibility and the reliability of sources using language specific features. Approaches like (Castillo et al., 2011; Qazvinian et al., 2011; Yang et al., 2012; Xu and Zhao, 2012; Gupta et al., 2013; Zhao et al., 2015; Volkova et al., 2017) propose supervised methods for detecting deceptive content in social media platforms like Twitter, Sina Weibo, etc. Similarly, approaches like Ma et al. (2016); Ruchansky et al. (2017) use neural network methods to identify fake news and rumors on social media. Kumar et al. (2016) studies the problem of detecting hoax articles on Wikipedia. All these rely on domain-specific and community-specific features like retweets, likes, upvotes, etc.

7 Conclusion

In this work, we propose a completely automated end-to-end neural network model, DeClarE, for evidence-aware credibility assessment of natural language claims without requiring hand-crafted features or lexicons. DeClarE captures signals from external evidence articles and models joint interactions between various factors like the context of a claim, the language of reporting articles, and trustworthiness of their sources. Extensive experiments on real world datasets demonstrate our effectiveness over state-of-the-art baselines.

References

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 675–684, New York, NY, USA. ACM.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 69–76.

Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. 2009. Integrating conflicting data: The role of source dependence. *Proc. VLDB Endow.*, 2(1):550–561.

Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-based trust: Estimating the trustworthiness of web sources. *Proc. VLDB Endow.*, 8(9):938–949.

Omar Enayet and Samhaa R. El-Beltagy. 2017. Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 470–474.

Jing Gao, Qi Li, Bo Zhao, Wei Fan, and Jiawei Han. 2015. Truth discovery and crowdsourcing aggregation: A unified perspective. *PVLDB*, 8(12):2048–2049.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*, ICANN'05, pages 799–804, Berlin, Heidelberg. Springer-Verlag.

Aditi Gupta, Hemank Lamba, Ponnurangam Kumaraguru, and Anupam Joshi. 2013. Faking sandy: Characterizing and identifying fake images on twitter during hurricane sandy. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 729–736, New York, NY, USA. ACM.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Srijan Kumar, Robert West, and Jure Leskovec. 2016. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 591–602, Republic and

Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. 2014. A confidence-aware approach for truth discovery on long-tail data. *Proc. VLDB Endow.*, 8(4):425–436.

Xian Li, Weiyi Meng, and Clement Yu. 2011. T-verifier: Verifying truthfulness of fact statements. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 63–74, Washington, DC, USA. IEEE Computer Society.

Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. *SIGKDD Explor. Newsl.*, 17(2):1–16.

Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2015. On the discovery of evolving truth. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 675–684, New York, NY, USA. ACM.

Chenghua Lin, Yulan He, and Richard Everson. 2011. Sentence subjectivity detection with weakly-supervised learning. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1153–1161. Asian Federation of Natural Language Processing.

Shanshan Lyu, Wentao Ouyang, Huawei Shen, and Xueqi Cheng. 2017. Truth discovery by claim and source embedding. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 2183–2186, New York, NY, USA. ACM.

Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. 2015. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 745–754, New York, NY, USA. ACM.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3818–3824. AAAI Press.

Subhabrata Mukherjee and Gerhard Weikum. 2015. Leveraging joint interactions for credibility analysis in news communities. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15.

Subhabrata Mukherjee, Gerhard Weikum, and Cristian Danescu-Niculescu-Mizil. 2014. People on drugs: Credibility of user statements in health communities. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 65–74, New York, NY, USA. ACM.

Ndapandula Nakashole and Tom M. Mitchell. 2014. Language-aware truth assessment of fact candidates. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1009–1019.

Jeff Pasternack and Dan Roth. 2011. Making better informed trust decisions with generalized fact-finding. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2324–2329.

Jeff Pasternack and Dan Roth. 2013. Latent credibility analysis. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1009–1020, New York, NY, USA. ACM.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*, EMNLP '14.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility assessment of textual claims on the web. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2173–2178, New York, NY, USA. ACM.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion.

Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1589–1599, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP '17.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association* *for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1659. Association for Computational Linguistics.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 797–806, New York, NY, USA. ACM.

Mehdi Samadi, Partha Talukdar, Manuela Veloso, and Manuel Blum. 2016. Claimeval: Integrated and flexible framework for claim evaluation using credibility of sources. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 222–228. AAAI Press.

Vikram Singh, Sunny Narayan, Md. Shad Akhtar, Asif Ekbal, and Pushpak Bhattacharyya. 2017. IITP at semeval-2017 task 8 : A supervised approach for rumour evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 497–501.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 647–653. Association for Computational Linguistics.

V.G. Vinod Vydiswaran, ChengXiang Zhai, and Dan Roth. 2011. Content-driven trust propagation framework. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 974–982, New York, NY, USA. ACM.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 422–426.

Janyce Wiebe and Ellen Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'05, pages 486–497, Berlin, Heidelberg. Springer-Verlag.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Qiongkai Xu and Hai Zhao. 2012. Using deep linguistic features for finding deceptive opinion spam. In *Proceedings of COLING 2012: Posters*, pages 1341–1350. The COLING 2012 Organizing Committee.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, MDS '12, pages 13:1–13:7, New York, NY, USA. ACM.

Xiaoxin Yin, Jiawei Han, and Philip S. Yu. 2008. Truth discovery with multiple conflicting information providers on the web. *IEEE Trans. on Knowl. and Data Eng.*, 20(6):796–808.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1395–1405, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Shi Zhi, Bo Zhao, Wenzhu Tong, Jing Gao, Dian Yu, Heng Ji, and Jiawei Han. 2015. Modeling truth existence in truth discovery. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1543–1552, New York, NY, USA. ACM.

It's going to be okay:
Measuring Access to Support in Online Communities

Zijian Wang
Symbolic Systems Program
Stanford University
`zijwang@stanford.edu`

David Jurgens
School of Information
University of Michigan
`jurgens@umich.edu`

Abstract

People use online platforms to seek out support for their informational and emotional needs. Here, we ask what effect does revealing one's gender have on receiving support. To answer this, we create (i) a new dataset and method for identifying supportive replies and (ii) new methods for inferring gender from text and name. We apply these methods to create a new massive corpus of 102M online interactions with gender-labeled users, each rated by degree of supportiveness. Our analysis shows wide-spread and consistent disparity in support: identifying as a woman is associated with higher rates of support—but also higher rates of disparagement.

1 Introduction

Despite substantial efforts to reduce gender disparities in online social contexts, gender gaps persist and, increasingly, negatively affect women through online harassment (Duggan, 2017). Online social platforms still serve a critical role for individuals as they seek to fill informational and emotional needs, frequently by interacting with others (Goswami et al., 2010; Chuang and Yang, 2012; Hether et al., 2016). The supportive replies of others help promote personal well-being (MacGeorge et al., 2011), yet unsupportive replies can not only lead to distress but discourage online engagement altogether. Given gender disparity in the receipt of anti-social behavior, to what degree does this disparity persist in individuals' receipt of support? We answer this question, illustrated in Figure 1, by examining supportive and unsupportive message rates across millions of online interactions, using a new computational model of support. Our work is motivated by an agenda of promoting supportive online platforms where people can participate equally.

Comment: `KatieZ22`: I'm nervous about my differential calculus exam next week. My current idea is work through problems on previous exams. But yiiiikes.
∟ **Reply**: `PizzaMagic`: You can ace that test! Your plan seems smart and you have plenty of time to prepare. :)

Figure 1: In this fictitious example, `KatieZ22` receives a supportive reply from `PizzaMagic`. In choosing their names, each user has chosen a particular gender performance, signaling female and gender-anonymity, respectively. In online settings, such gender performances evoke stereotypes that affect how others interact and provide access to online resources. Our study asks what effect does this gender signaling have on individuals receiving support and disparagement?

This work connects with the growing body of computational studies of gender disparity in online behavior (e.g., Lam et al., 2011; Magno and Weber, 2014; Garimella and Mihalcea, 2016; Li et al., 2018); our work here examines this disparity along a new dimension, support, and unlike prior work, examines disparity along the full spectrum of both pro-social (supportive) and anti-social (unsupportive) behaviors. Prior works have also examined the language of support in online support forums for health-related issues (Biyani et al., 2014; Wang et al., 2012; De Choudhury and De, 2014; Althoff et al., 2016; De Choudhury and Kiciman, 2017), often with the aim of improving people's access. Here, we aim to study support in general, everyday interactions, drawing upon theories of how support is expressed in language (Cutrona and Suhr, 1992; Wright et al., 2003).

Our investigation provides four main contributions. First, we introduce a new task of rating the supportiveness of a message and provide an accompanying dataset of 9,032 post-reply pairs with annotations (§2). Second, using this data, we develop a new computational model for automatically identifying supportive and unsupportive replies (§3), using theory-based features that operationalize linguistic strategies for giving support. Third, we develop a new state-of-the-art sys-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 33–45
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

tem for classifying the gender of a username (§4) and construct a massive dataset of over 102M post-reply pairs from three online platforms (§5), where participants are labeled by gender. Further, the text of each post is rated for its gender predictiveness, enabling studying gender performance at the name and textual levels. Finally, we apply our support classifier to our social interaction dataset to reveal wide-spread disparity on the basis of gender (§6). Our results show that when gender is performed, female performances are associated *both* with higher rates of supportive comments *and* with higher rates of unsupportive comments, highlighting that gender disparity is not just for negative behaviors online.

2 The Language of Support

Individuals engage in online platforms for a variety of reasons and supportive responses to this engagement can take many forms (Shumaker and Brownell, 1984; Vaux, 1985), from informational support like advice to emotional support like expressions of sympathy. Responders may choose from different linguistic support strategies depending on the speaker and context (Cutrona and Suhr, 1992). For example, given an individual commenting on a Wikipedia talk page about their idea for adding new content, a responder may point to an additional resource they can use, whereas given an individual posting to Reddit for relationship advice, a responder may express sympathy. Our goal is to study the language and behavior of *everyday* supportive or unsupportive interactions as they occur on three large social platforms: Reddit, StackExchange, and Wikipedia. These platforms represent common settings people seek out to engage in discussions and ask for help. Therefore, we annotate a dataset by degrees of support and analyze how linguistic expectations of support manifest in online interactions.

Data and Annotation Post and reply pairs were selected from the three platforms. Many of these interactions are short and therefore to increase diversity, annotated pairs were sampled by balancing by platform and the lengths of posts and replies seen in each.

As a social activity, support is often expressed by drawing upon other social strategies such as politeness (Feng et al., 2013). To help focus annotators' attention on support specifically and disentangle related social cues, we pair our support annotation with contrastive annotation questions for three other related phenomena: agreement (with the post's message), politeness, and offensiveness. Annotators were asked to rate support on a five-point Likert scale from very unsupportive and very supportive, with analogous questions each for agreement and politeness; offensiveness was rated on a five point scale from inoffensive to very offensive.

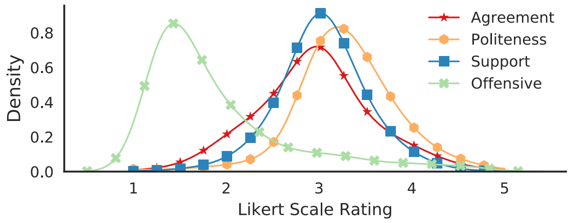

Figure 2: Annotators' rating distributions.

Rating	Example Reply
1.33	see your arse mate, stop talking out of it.
2.0	We dont like your kind.
3.0	I was referring to <link>
3.33	thanks chief.
3.66	Not many people, apparently, but I'm listening!
4.0	Im in as well! Meet up at the Tavern?!
4.33	I like your style!
4.66	Love this. Thank you, more please.

Table 1: Examples of annotator ratings of Support from 1 to 5. Additional examples are in Supplemental §2.

All data was annotated using CrowdFlower with detailed instructions and example replies for each level of the Likert scale. Each task presented five post-reply pairs and with detailed instructions that ask annotators to focus on rating each reply along these four dimensions. Annotators were required to pass a training phase where they had at least 70% agreement with a gold standard annotation on 10 items. After training, each task included one control question, which was used to remove annotators whose agreement with the gold standard fell below 70%.

In total, 9,032 instances were annotated by three annotators, who had a Krippendorff's α of 0.766, indicating substantial agreement on the data (Artstein and Poesio, 2008). Figure 2 shows the distribution of ratings. Support is positively correlated with agreement (r=0.71) and politeness (r=0.51), though annotators rated replies as having more politeness than support on average. Support annotation examples are shown in Table 1. Offensiveness was negatively correlated (r=-0.38) with support.

Analysis Supportive replies can use a variety of strategies for offering support, depending on need

	Strategy	Support	In Top-25%	Example
INFORMATIONAL	Suggestion advice	0.043	16.3%	You might try...
	Referral	0.091	1.3%	Please see [URL]
	Situational appraisal	-0.071^{**}	1.5%	Your situation sounds like...
	Teaching	-0.065^{***}	2.7%	The reason that's happening...
TANGIBLE	Direct offer to do something	-0.020	0.04%	Do you want me to?
	Willingness	0.266^{***}	1.6%	I could help you...
ESTEEM	Compliment	0.337^{***}	23.8%	Great idea!
	Validation	0.248^{***}	27.2%	You're right about...
	Relief of blame	0.490^{***}	1.2%	It's not your fault that...
	Companionship Reminder	-0.089^{*}	0.7%	Your friends and family still...
EMOTIONAL	Sympathy	0.041	0.1%	Sorry to hear that
	Listening	-0.104^{***}	3.1%	Why did you feel...
	Empathy	0.067	0.08%	I know how you feel...
	Encouragement	0.423^{***}	1.6%	Go for it
	Accommodation	0.035	74.7%	
	Emotion	0.081^{***}	27.2%	

Table 2: Support strategies and their presence in our data, as shown through the mean supportiveness rating (-2 to 2) for replies using that strategy and the percentage of posts in the top 25% of the most-supportive that employ the strategy. For supportiveness, posts are compared with all others not employing that strategy, with significance measured using the Mann-Whitney U test. Throughout the paper, *** denotes $p<0.001$, ** $p<0.01$, and * $p<0.05$.

and context. Cutrona and Suhr (1992) proposed a broad taxonomy of support strategies based on in-person interactions, such as offering an appraisal of the current situation or seeking to relieve the other person of blame. We examine to what degree are these strategies employed in online, relatively-anonymous settings and whether their usage online is associated with higher perceived support. To test this, support strategies were automatically identified using a combination of regular expressions for lexical patterns and dependency-parsed trees, together with specialized lexicons matching each strategy and rules for detecting negation. For example, suggestions were detected by identifying a second-person subject with a modal verb indicating possibility (Quirk et al., 1985, p. 219).

Many of the strategies suggested by Cutrona and Suhr (1992) for expressing support in person were also observed online; Table 2 shows the average supportiveness rating for replies containing each strategy, where supportiveness is centered to $[-2, 2]$. Further, their effect on perception of supportiveness, while small, is significant and positive for many. However, we do observe two notable negative trends. First informational support strategies of aiding a person by reassessing their situation (e.g., offering a new perspective) and by teaching were considered less supportive. We observed that in several cases these strategies were employed when an individual was not seeking support, in which case unrequested new information can appear condescending. Indeed, replies employing these two support strategies were still rated polite, 0.30 and 0.31 mean politeness respectively, suggesting the context in which new information is given weighs heavily on whether it is treated as supporting the individual.

Support can also be conveyed implicitly through unconscious stylistic choices. Rains (2016) notes linguistic accommodation is frequently observed in supportive responses, where individuals match the function word frequency of the original communication (Bucholtz and Hall, 2005; Danescu-Niculescu-Mizil et al., 2011). Shown in the bottom of Table 2, we observe only a weak non-significant positive association between support and accommodation (as measured using Danescu-Niculescu-Mizil et al. (2011)), though the single post-reply unit of analysis limits our ability to detect long-term accommodation across multiple dialog turns. Liviatan et al. (2008) and Li and Feng (2014) found that supportive replies often to contain more emotional language, which evokes a personal connection and intensity (Spottswood et al., 2013; Braithwaite et al., 1999). Shown in the bottom of Table 2, we find a significant and positive association where more emotive posts are viewed as more supportive.

Finally, we note that a few strategies suggested by Cutrona and Suhr (1992) were not seen in our annotated dataset. Strategies of offering to participate, using physical affection, assurances of confidentiality, material and financial loans, and prayer were not seen; a broader scan of our unannotated data did find these attested but rare in practice. We attribute this rareness to the public, online nature of the interactions, in contrast to the interpersonal setting studied by Cutrona and Suhr (1992).

3 Computational Model of Support

Our primary objective is to measure the relationship between identity and support in online social systems. Therefore, we next develop a classifier to automatically label replies with support.

Features As a social activity, the language of support draws upon multiple lexical and stylistic cues. We base on classifier on *theory-inspired* and data-driven features. The first set consists of the operationalized linguistic strategies for expressing support, shown in Table 2. Further, in constructing our feature set, we build upon past linguistic analyses of related social-situated language. Wellman and Wortley (1990) note that the availability of support is related to social distance, which is in part expressed linguistically through the degree of formality (Hovy, 1987; Sigley, 1997). Therefore, we include features from Pavlick and Tetreault (2016), which examined linguistic markers of formality. Advice giving is a core component of many theories of support (MacGeorge et al., 2011) and such advice is frequently wrapped in politeness language (Feng et al., 2013), e.g., hedging suggestions rather than imposing direction, which provides face-saving opportunities for the person receiving support (Clark and Schunk, 1980). Therefore, we include the feature set of Danescu-Niculescu-Mizil et al. (2013), which though focused on requests, provides many general lexical patterns for politeness.

Beyond these, we include features motivated by observational studies of support. In analyzing online support groups, Alpers et al. (2005) found that LIWC (Pennebaker and Stone, 2003) was valid as a construct for analyzing messages and compared similarly human judgments about the categories. To capture emotional language (Li and Feng, 2014; Liviatan et al., 2008), we include the NRC emotion lexicons of Mohammad and Turney (2013). Given the informational support strategy, we include lexicons from argumentation for capturing explanatory replies (Teufel, 2000).

Support may be given in response to stressors, which change in nature throughout a person's lifetime (Vaux, 1985; Segrin, 2003). To potentially capture variation in the language of support based on the posting individuals, we include features known to be associated with age such as elongation and capitalization (Goswami et al., 2009; Barbieri, 2008), grammatical differences in sentence construction and length (Hovy and Søgaard, 2015), and a lexicon for age of acquisition (Kuperman et al., 2012).

Data-driven features include (1) lexical features capturing the presence of n-grams, their relative frequency, (2) grammatical features from dependency-parsed triples, which are also backed off to parts of speech, (3) word lexicons for formality, sentiment, and subjectivity, (4) style features such as word and sentence length, complexity, and use of contractions, and (5) the average word vector for the sentence.

In total, our model includes 23,903 features, the bulk of which are n-grams and dependency triples. A detailed listing of all features is provided in Supplemental §1.

One notable feature that we did not include was the presence of self-disclosure in a reply, which has been linked to high-social support as a way of conveying connection and empathy (Wright et al., 2003). While computational models for self-disclosure have been proposed (Bak et al., 2014; De Choudhury and De, 2014), we were unable to scale these methods to the size of our analysis.

Task Setup Support ratings are discretized to create a ternary classification task with labels $\{-1, 0, 1\}$, denoting unsupportive, neutral, and supportive comments. Ratings were discretized by treating all those ratings $\leq$-0.67 as negative and those with a rating $\geq$0.67 as positive.

A Random Forest classifier was trained on all 23,903 features; random forests are robust to overfitting even with large numbers of features, making them suitable for this high-dimensional feature space (Fernández-Delgado et al., 2014). Furthermore, random forests are able to learn conjunctive features, allowing us to learn how combinations of strategies are employed to yield support. As the majority of posts are neutral, we mitigate the class imbalance using SMOTE (Chawla et al., 2002) to generate synthetic examples in the

Model	Macro-F1
Our Model	**0.52**
Bigram Features	0.43
Unigram Features	0.42
Support Features (Table 2)	0.40
Majority	0.29
Random	0.26

Table 3: Support classification performance

training fold using the 5 nearest neighbors, taking care to avoid contamination of the test set. The classifier is implemented using Scikit-learn (Pedregosa et al., 2011) and syntactic processing was done using spaCy (Honnibal and Johnson, 2015). Word vectors are the publicly released Google-News word2vec vectors (Mikolov et al., 2013).

Three works have examined related tasks where Biyani et al. (2014) and Khanpour et al. (2018) classify posts in online cancer support groups as providing informational or emotional support and Wang et al. (2012) classify the degree of support along these dimensions. Here, we solve a more general task that includes unsupportive comments and is in the general domain.

Evaluation We compare our full model for predicting support against three models: our 14 features for detecting support strategies from Table 2, a model trained on the subset of unigram features (4,352), and a model trained on bigram features (8,897), the latter of which is known to be a strong lexical baseline (Wang and Manning, 2012). All models were tested using five-fold cross-validation with Macro-F1 for evaluation and including baselines for labeling instances at random or choosing the most frequent.

Our full model obtains substantial improvements over all baselines and models, as shown in Table 3. Further, the simple support strategy features provide a large and statically-significant improvement over the two baselines. The model using support strategy features performs similarity to the unigram model, despite having two orders of magnitude fewer features. A follow-up analysis on cross-platform performance, described in Supplemental §1.2, showed that while within-platform performance was relatively high (0.54 Macro F1 for Reddit and 0.53 for Wikipedia), performance for the more technical StackExchange site was lower both within (0.44) and across (0.40 when trained on Reddit and 0.42 when on Wikipedia).

Examining our full model's most important features showed that the two support strategies for validation and compliments (cf. Table 2) were the most important features, followed closely by lexicons for emotion: Anger in LIWC, Disgust in NRC, and the positive sentiment in Liu et al. (2005), all of which were motivated by theory. These results confirm that our theory-inspired features are both salient for supportiveness and effective as features.

4 Inferring Gender

As a part of interacting, individuals present a view of themselves as an interlocutor, revealing aspects about themselves such as gender through explicit means (Marwick, 2013; Allen and Wiles, 2016), e.g., profile pictures, or through implicit—and potentially unconscious—cues such as stylistic choices in language (e.g., Eckert and McConnell-Ginet, 2003; Bamman et al., 2014). In the relatively anonymous and deindividuated online setting, these identity cues can have a profound impact on how other perceive and interact with them (e.g., Mickelson et al., 1995; Herring, 2003; Ammari et al., 2014; Megarry, 2014) and these minimal gender cues give rise to full-fledged social stereotypes and, potentially, the negative behavior that comes when treating someone as a stereotype (Kiesler et al., 1984; Lea and Spears, 1991; Postmes et al., 1998; Wang et al., 2009). Here, we develop methods for inferring gender from two signals: (1) names that users chose; and (2) implicit cues conveyed by linguistic features.

4.1 Gender from Names

Prior work has developed models for inferring gender from username alone (e.g., Tang et al., 2011; Liu and Ruths, 2013; Jaech and Ostendorf, 2015; Knowles et al., 2016). Here, we develop a new character-based neural model that incorporates rich gender-labeled username information for identifying additional gender-salience cues in usernames from roles and attributes, e.g., `SuperDad1` or `AspiringActress99`.

Data Individuals convey their gender in multiple ways beyond using gender-associated names. Therefore, to capture this variety, we collect usernames from two online platforms where users have self-declared their perceived gender. First, Twitter usernames and screen names were collected from a 10% sample from 2014 to 2017. Here, we identify usernames whose biography contains an explicit mention of their gender, e.g., by stating a gendered role "mom to two kids" or specifying pronoun preferences "he/him/his." Gendered profiles were collected for 4,900,250 individuals using a selection of lexical patterns with aggressive filtering to remove false positives. Second, we collect 283,427 usernames from Reddit identified through self-declarations of gender in /R/RELATIONSHIPS, e.g., "I [23F] need to talk to my boyfriend [27M]", and 84,068 usernames where the user has chosen a gender-indicating flair (a visual icon displayed within the subreddit). These two sources provide much-needed variation for gender in usernames beyond those mirroring full names.

Model Given a username, we infer gender using a character-based encoder consisting of three stacked LSTM networks (Hochreiter and Schmidhuber, 1997). Following platform restrictions on usernames, character sequences are restricted to being in ascii range and are embedded into 16 dimensional vectors as input. Adopting best practices (Ioffe and Szegedy, 2015), batch normalization is applied prior to the dense layer used to compute the gender prediction. LSTMs were sized at 256 after limited hyperparameter tuning on development data. We optimize with Adam (Kingma and Ba, 2014) with a learning rate of 0.002.

Training and Evaluation All data is partitioned into 80% train, 10% development, and 10% test splits. As some usernames are repeated in different communities, we keep only one unique instance prior to partitioning to avoid leakage between partitions.

Training mini-batches were balanced for both genders, which yielded better performance in tests on the development data. We compare the performance of our model on the test set against two current state-of-art systems available off the shelf for inferring gender from usernames, demographer (Knowles et al., 2016) and Jaech and Ostendorf (2015). Demographer is trained on names from the Social Security Administration and the method of Jaech and Ostendorf (2015) is trained on usernames from OkCupid and uses 3.5M Snapchat usernames for self-learning to improve accuracy.

As shown in Table 4, our model outperforms both systems by substantial margins for both Twitter and Reddit data. In tests on data from both papers reported in Supplemental §3, our model also outperforms their systems. High accuracy is not expected for these models in most domains, as many usernames do not signal gender.

Method	Twitter	Reddit
Our Model	**0.7785**	**0.6299**
Jaech and Ostendorf (2015)	0.7028	0.5935
Knowles et al. (2016)	0.6520	0.5216

Table 4: Gender inference (Macro-F1)

4.2 Gender from Text

Gender can also manifest through more subtle, stylistic cues (e.g., Schnoebelen, 2012; Flekova and Gurevych, 2013; Bamman et al., 2014; Volkova et al., 2015; Garimella and Mihalcea, 2016; Carpenter et al., 2016). Thus, even when a person chooses a neutral username, their linguistic style may reveal their gender. Therefore, we construct a regression model to infer the degree to which either gender is expressed through text.

Data and Model Gender-labeled post data was constructed using held-out data from our three platforms where posts were authored by a user with a high-confidence gender prediction. Posts were randomly sampled across forums (e.g., subreddits) from the held-out data to achieve gender parity with 555K posts for Wikipedia, and 58K for StackExchange; Reddit was subsampled to 1M posts total due to its size.

Features were selected by drawing upon prior work: (i) stylistic features like punctuation and number frequencies, casing, word length and (ii) content features including n-grams, sentiment, and specialized lexicons like LIWC. A full listing of features is reported in Supplemental §1.

Following prior work (Bamman et al., 2014), a logistic regression model was trained for each platform using L2 regression; we adopt separate models for each to better adapt to any platform-specific gender variation.

Evaluation Models were evaluated using AUC with five-fold cross validation, with 0.661 AUC for StackExchange, 0.700 for Wikipedia, and 0.661 for Reddit. The models perform substantially better than random choice (0.5) for the challenging task of inferring gender from a single post, as many posts contain no signal of gender. Additional analyses are reported in Supplemental §1.2.

5 Gender-Salient Interaction Data

To quantify the social support people receive online, we examine communications from three major online communities: Reddit, StackExchange, and Wikipedia. We refer to a communication between individuals as a *post* with a *reply*, defining

	Male	Female	Neutral
Reddit	1,017,455	213,849	1,211,813
SE	2,431,234	289,864	836,531
Wiki.	80,876	14,073	46,954

Table 5: Users with high-confidence or neutral gender

Gender: Female	**Gender: Male**	**Gender: Neutral**
Shanakitty	AdamMcAdamson	kazkeb
cassycas	ChipCarlson11	xuebin671
kelseylenae	BaBa_Dad	ConfigurationalYes
Mrs_BruceWayne	BarryCA67	thelizardof_Oz
madelaine00x	BenJewish	oibird
norma-gaspard	dojoguy	Merpageddon

Table 6: High-confidence and neutral name examples.

these for each platform next.

Reddit Reddit data was selected using a longitudinal sample of one month (July) per year, from 2006 to 2017, and a continuous sample of one full year's data in 2017. Our initial data consists of the top 10,000 subreddits ordered by the total number of posts in the data. From these communities we restrict our analysis to a comment and its first reply, which reduces confounds from multiparty communication. These post-reply pairs were further filtered to remove non-English posts using Google CLD2 (McCandless, 2010), yielding 434.29M candidate communications.

StackExchange StackExchange (SE) contains substantial social interaction in the comment to posts and replies (Ahn et al., 2013; Danescu-Niculescu-Mizil et al., 2013). These communications often expand beyond the immediate topic. Directed communication within these comments is frequently signaled using an explicit mention starting with an "@," which we use to identify pairs. In total, we collected post-reply pairs from the full history of all StackExchange, yielding 3.16M pairs across 162 sites.

Wikipedia Wikipedia features an active social component in its talk pages, with more personal communication–or even personal attacks–during debates around appropriateness or suggested changes (Bender et al., 2011). Similar to Reddit, we construct post-reply pairs by identifying each comment and its first response on a talk page, yielding 26.7M pairs from 387K talk pages.

Assigning Gender All post-reply pairs were labeled using our post classifier (§4.1). Posts with *high-confidence* gender predictions (softmax probability $>$ 0.9 or $<$ 0.1) were labeled with the predicted gender. To contrast the effects of having a gendered name, we construct a complementary dataset where the posting user's username is effectively gender *neutral*, e.g., `user1209`; these neutral names are chosen from those with near-chance probability in the output softmax $0.45 < p < 0.55$. The relative counts of high-confidence and neutral gender names in each platform are shown in Table 5, along with examples in Table 6.

Restricting the dataset to pairs where we have a salient identity, our final dataset for analysis consists of 49.58M, 0.72M, and 3.69M pairs for gender in Reddit, StackExchange, and Wikipedia; and 46.19M, 201.7K, and 1.60M for neutral in each, respectively. Where possible, we also record any high-salience identities for replying users.

6 Gender and Support

The gender cues provided through computer mediated communication provide enough information that a person will fill in the result with a stereotype (Lea and Spears, 1991; Spears and Lea, 1992). What effect might this stereotyping have for access to support? While establishing full causality for an answer is infeasible in our current observational study, we take the first step by quantifying whether disparity in support exists and examine what contextual factors may affect support giving. Using our classifier, we label the 102M post-reply pairs from our dataset (§5), which includes both high-confidence and gender-neutral users.

Model To quantify access to support, we construct separate mixed-effect logistic regression models for predicting the dependent variable of whether a post will receive a supportive reply and for whether it will receive an unsupportive reply. Random effects are added for each community within a platform, which capture the variance in support rates between communities, e.g., due to differences in community norms, topic, or size. As fixed effects, we include a categorical variable for name-inferred gender, always using the gender-neutral condition as the reference coding, which, critically, allows us to examine the changes of support for revealing gender relative to users whose identity is effectively anonymous. We include the predicted probability of a user's gender from their writing (§4.2), centered to $[-0.5, 0.5]$ such that 0 denotes a gender-neutral writing style. Finally, we include interaction terms for writing and names to capture effects of joint gender performance.

Results Three main results are observed. First,

	SUPPORTIVE			UNSUPPORTIVE		
	Reddit	**StackExchange**	**Wikipedia**	**Reddit**	**StackExchange**	**Wikipedia**
intercept	−2.3775***	−2.4729***	−2.9298***	−3.0232***	−4.3734***	−2.9787***
♀name	0.5480***	0.5409***	0.7376***	0.2927***	0.6071***	0.3543***
♂name	0.4652***	0.6639***	0.6422***	0.3230***	0.6445***	0.3313***
♀📝	1.1174***	−1.0778***	0.5261***	0.2520*	0.7398***	−0.1451**
♀name ∧ ♀📝	0.8802***	1.2751***	0.1155	0.2662*	0.0537	0.1695
♂name ∧ ♀📝	0.0535	0.8119***	−0.2070***	0.2016	0.3556*	0.4750***

Table 7: Logistic regression coefficients for predicting whether a post will receive a supportive reply (left) or unsupportive reply (right) on the basis of a gendered name and writing (denoted 📝), with coefficients for name and writing interactions. Name gender is categorical with the reference coding is neutral. The writing coefficient is center at 0 (neutral) with positive values being more female.

the use of a gender-conveying name is associated with both higher rates of supportive comments and unsupportive comments. Our results agree with those from the small scale study of Feng et al. (2013) who found that accounts with human pictures and person-sounding usernames receive higher social support. Indeed, while several studies have touted the benefits of anonymity online for discussing sensitive topics (Campbell and Wright, 2002; Wright, 2002a,b), our results suggest that selecting a gender neutral name may lead to lower support overall. Our findings also reinforce the observation that the personal-anonymity online does not lead to equal support due to cues about identity (Postmes and Spears, 2002).

Second, the rates of supportive and unsupportive comments are significantly associated with *both* kinds of gender performances (i.e., names and writing style). These results suggest that online audiences are sensitive to both kinds of overt and implicit gender displays and that even innocuous choices such as gendered usernames can shape our online interactions.

Third, when gender is performed in together name and writing, female performances are consistently associated across all three platforms with higher rates of receiving supportive replies and unsupportive replies. Note that this trend is seen in the cumulative effect on support after combining coefficients for the interactions term with the coefficients for writing and name. We illustrate this cumulative effect for two types of gender performances in Reddit, shown as separate axes in Figure 3. Indeed, when a user has a gendered username, the cumulative effect of male writing performance is consistently associated with fewer supportive replies. In small-scale interpersonal studies, Abbey et al. (1991) and Barbee et al. (1993) note men are more likely to receive unsupportive comments; however, we did not observe this disparity in our online setting.

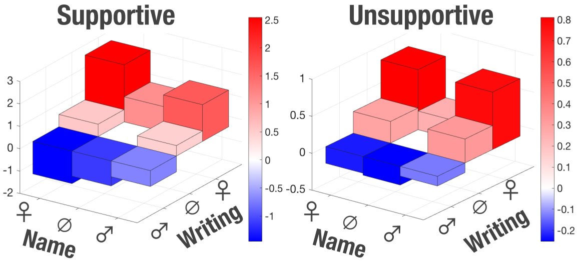

Figure 3: Cumulative effect of name and writing performances on the log-odds coefficients for receiving comments of each type in Reddit when the gender is maximally performed (cf. Table 7), excluding the intercept effect for clarity (∅ denotes neutral performance). Positive values (red) denote increased comment frequency and negative values (blue) show decreased frequency. This plot shows that as female performance becomes more salient (shown bottom to top visually), such users receive increasingly-higher rates of supportive and unsupportive replies in Reddit.

Does the replier's gender matter? Mickelson et al. (1995) note that men and women differ in *how* they receive support, with the gender composition of the interacting pair driving the kind of supportive behavior. Here, we examine whether men and women differ in the *rates* they give support to one another, using gendered names as a proxy for identity. Because only Reddit has sufficient data, we construct a mixed-effect regression model for Reddit using the 4.5M post-reply pairs where the replier has a high-confidence or neutral gender and include the replier's gender as a factor with interactions for the poster. The results shown in Table 8 reveal two main conclusions. First, men and women give supportive comments at different rates, with women being far more likely to leave supportive replies and less likely to leave unsupportive replies. Second, the interaction terms show that there is minimal dyadic interaction between the gender identities of the poster and replier with

	Sup.	Unsup.
intercept	-2.391^{***}	-3.107^{***}
P:♀name	0.561^{***}	0.288^{***}
P:♂name	0.450^{***}	0.330^{***}
P:♀📝	1.263^{***}	0.290^{***}
R:♀name	0.249^{***}	-0.153^{***}
R:♂name	-0.057^{***}	-0.036^{**}
P:♀name ∧ P:♀📝	0.914^{***}	0.212^{**}
P:♂name ∧ P:♀📝	0.082	0.131
P:♀name ∧ R:♀name	-0.103^{***}	-0.037
P:♂name ∧ R:♀name	-0.033	0.026
P:♀name ∧ R:♂name	-0.004	0.036^{*}
P:♂name ∧ R:♂name	0.031^{*}	-0.003
P:♀📝 ∧ R:♀name	0.209	-0.202
P:♀📝 ∧ R:♂name	-0.331^{**}	-0.228^{*}
P:♀name ∧ P:♀📝 ∧ R:♀name	-0.343	0.376
P:♀name ∧ P:♀📝 ∧ R:♂name	0.329	0.312
P:♂name ∧ P:♀📝 ∧ R:♀name	0.067	0.178
P:♂name ∧ P:♀📝 ∧ R:♂name	0.024	0.203

Table 8: Regression coefficients for Reddit when the gender identity of the replier is known. The post author's identity is denoted with a P and replier's with R.

respect to rates of giving supportive or unsupportive comments. We only observe significant interactions indicating (1) replying users with female names are *less* likely to leave supportive replies to posting users with female names and (2) replying users with male names are *i*) *more* likely to leave supportive replies to other users with male names, *ii*) *less* likely to leave supportive comments if the writing appears more female, *iii*) *more* likely to leave unsupportive comments if the posters name is female, and *iv*) *less* likely to leave unsupportive comments if the writing appears more female.

Limitations The observations of our study should still be viewed within its practical limitations, of which we note two. First, in examining the content of replies, we do not control for potential direct or indirect requests for help in text that may ultimately affect the rates of support. This issue could be a potential confound, as the cultural norms for masculinity often promote self reliance (Addis and Mahalik, 2003), ultimately leading to gendered differences in requests. Second, this observational study cannot establish causality between gender displays and support; while the disparity is real, exogenous factors could potentially explain the disparity without finding gender displays as a cause, though the mixed effects still control for some contextual variability in the different support frequencies across communities. In spite of these limitations, we view this work as an important first step for demonstrating gender disparity in support—both positive and negative—and inviting future work to establish a causal explanation.

7 Ethical Considerations

The use of gender as a variable in NLP requires that we also discuss ethical considerations resulting from this work, as it directly relates to identity and the dignity of persons being studied. Following the guidelines of Larson (2017) for using gender in NLP, our use of gender is intentional and central to this study on gender disparities in received support. We base our notion of gender as one of linguistic performance (DeFrancisco et al., 2013), in which individuals adapt their style and name to emphasize or de-emphasize certain aspects of their gender identity (Eckert, 2008). Accordingly, we have opted represent gender performance along a graded scale, though we recognize that this representation does not capture non-binary gender identities.

The gender inference methods introduced here raise ethical considerations as they ultimately enable automatic identification of gender for any person on the basis of name or writing (Hamidi et al., 2018). Such technology could be used to unfairly identify and target persons of either gender for malicious behavior or may harm through misgendering individuals. Ultimately, we decided that such risk was acceptable given the positive impact of our study on revealing gender disparity. We hope to also use our method to better support privacy-preserving behavior (Allen and Wiles, 2016; Reddy and Knight, 2016) by helping individuals identify and change names or statements that would indicate a particular gender. Further, we hope that when used in combination with our support classifier and a larger context of gendered interactions (Voigt et al., 2018), these technologies can identify healthy communities that are supportive of all people.

8 Conclusion

Individuals use social media to support their informational and emotional needs. Our study has shown wide-spread disparity in the levels of support individuals receive on the basis of their perceived gender. Our results were made possible through the development of a new massive 102M post-reply dataset tagged with high-salience and neutral gender and the introduction of a new task, annotated dataset, and model for classifying supportive messages. All data, code, and annotation guidelines are publicly released at `https://github.com/davidjurgens/support`.

Acknowledgments

The authors would like to thank Sarita Schoenebeck, Eric Gilbert, and Libby Hemphill for their helpful feedback and the anonymous reviewers for their comments and suggestions. The authors gratefully acknowledge the Volkswagen Foundation for their support in part of this work.

References

Antonia Abbey, Frank M Andrews, and L Jill Halman. 1991. The importance of social relationships for infertile couples' well-being. In *Infertility*, pages 61–86. Springer.

Michael E Addis and James R Mahalik. 2003. Men, masculinity, and the contexts of help seeking. *American Psychologist*, 58(1):5.

June Ahn, Brian S Butler, Cindy Weng, and Sarah Webster. 2013. Learning to be a better q'er in social q&a sites: Social norms and information artifacts. *Proceedings of the Association for Information Science and Technology*, 50(1):1–10.

Ruth ES Allen and Janine L Wiles. 2016. A rose by any other name: Participants choosing research pseudonyms. *Qualitative Research in Psychology*, 13(2):149–165.

Georg W Alpers, Andrew J Winzelberg, Catherine Classen, Heidi Roberts, Parvati Dev, Cheryl Koopman, and C Barr Taylor. 2005. Evaluation of computerized text analysis in an internet breast cancer support group. *Computers in Human Behavior*, 21(2):361–376.

Tim Althoff, Kevin Clark, and Jure Leskovec. 2016. Large-scale analysis of counseling conversations: An application of natural language processing to mental health. *Transactions of the Association for Computational Linguistics (TACL)*, 4:463.

Tawfiq Ammari, Sarita Yardi Schoenebeck, and Meredith Ringel Morris. 2014. Accessing social support and overcoming judgment on social media among parents of children with special needs. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

JinYeong Bak, Chin-Yew Lin, and Alice Oh. 2014. Self-disclosure topic model for classifying and analyzing twitter conversations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1986–1996.

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2014. Gender identity and lexical variation in social media. *Journal of Sociolinguistics*, 18(2):135–160.

Anita P Barbee, Michael R Cunningham, Barbara A Winstead, Valerian J Derlega, Mary R Gulley, Pamela A Yankeelov, and Perri B Druen. 1993. Effects of gender role expectations on the social support process. *Journal of Social Issues*, 49(3):175–190.

Federica Barbieri. 2008. Patterns of age-based linguistic variation in American English. *Journal of Sociolinguistics*, 12(1):58–88.

Emily M Bender, Jonathan T Morgan, Meghan Oxley, Mark Zachry, Brian Hutchinson, Alex Marin, Bin Zhang, and Mari Ostendorf. 2011. Annotating social acts: Authority claims and alignment moves in wikipedia talk pages. In *Proceedings of the Workshop on Languages in Social Media*, pages 48–57. Association for Computational Linguistics.

Prakhar Biyani, Cornelia Caragea, Prasenjit Mitra, and John Yen. 2014. Identifying emotional and informational support in online health communities. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 827–836.

Dawn O Braithwaite, Vincent R Waldron, and Jerry Finn. 1999. Communication of social support in computer-mediated groups for people with disabilities. *Health Communication*, 11(2):123–151.

Mary Bucholtz and Kira Hall. 2005. Identity and interaction: A sociocultural linguistic approach. *Discourse Studies*, 7(4-5):585–614.

Kristen Campbell and Kevin B Wright. 2002. On-line support groups: An investigation of relationships among source credibility, dimensions of relational communication, and perceptions of emotional support. *Communication Research Reports*, 19(2):183–193.

Jordan Carpenter, Daniel Preotiuc-Pietro, Lucie Flekova, Salvatore Giorgi, Courtney Hagan, Margaret L Kern, Anneke EK Buffone, Lyle Ungar, and Martin EP Seligman. 2016. Real men don't say "cute": Using automatic language analysis to isolate inaccurate aspects of stereotypes. *Social Psychological and Personality Science*.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research (JAIR)*, 16:321–357.

Katherine Y Chuang and Christopher C Yang. 2012. Interaction patterns of nurturant support exchanged in online health social networking. *Journal of Medical Internet Research*, 14(3).

Herbert H Clark and Dale H Schunk. 1980. Polite responses to polite requests. *Cognition*, 8(2):111–143.

Carolyn E Cutrona and Julie A Suhr. 1992. Controllability of stressful events and satisfaction with spouse support behaviors. *Communication Research*, 19(2):154–174.

Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words!: linguistic style accommodation in social media. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 745–754. ACM.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Munmun De Choudhury and Sushovan De. 2014. Mental health discourse on reddit: Self-disclosure, social support, and anonymity. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*.

Munmun De Choudhury and Emre Kiciman. 2017. The language of social support in social media and its effect on suicidal ideation risk. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*, pages 32–41.

Victoria Pruin DeFrancisco, Catherine Helen Palczewski, and Danielle D McGeough. 2013. *Gender in communication: A critical introduction*. Sage Publications.

Maeve Duggan. 2017. Online harassment 2017. *Pew Research Center*.

Penelope Eckert. 2008. Variation and the indexical field. *Journal of Sociolinguistics*, 12(4):453–476.

Penelope Eckert and Sally McConnell-Ginet. 2003. *Language and gender*. Cambridge University Press.

Bo Feng, Siyue Li, and Na Li. 2013. Is a profile worth a thousand words? how online support-seeker's profile features may influence the quality of received support messages. *Communication Research*, 43(2):253–276.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. 2014. Do we need hundreds of classifiers to solve real world classification problems. *Journal of Machine Learning Research (JMLR)*, 15(1):3133–3181.

Lucie Flekova and Iryna Gurevych. 2013. Can we hide in the web? large scale simultaneous age and gender author profiling in social media. In *Proceedings of the Conference and Labs of the Evaluation Forum (CLEF)*.

Aparna Garimella and Rada Mihalcea. 2016. Zooming in on gender differences in social media. In *Proceedings of the COLING Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media (PEOPLES)*, pages 1–10.

Sumit Goswami, Sudeshna Sarkar, and Mayur Rustagi. 2009. Stylometric analysis of bloggers' age and gender. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*.

Suparna Goswami, Felix Köbler, Jan Marco Leimeister, and Helmut Krcmar. 2010. Using online social networking to enhance social connectedness and social support for the elderly. In *Proceedings of the International Conference on Information Systems (ICIS)*. Association for Information Systems.

Foad Hamidi, Morgan Klaus Scheuerman, and Stacy M Branham. 2018. Gender recognition or gender reductionism?: The social implications of embedded gender recognition systems. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, page 8. ACM.

Susan C Herring. 2003. Gender and power in on-line communication. *The Handbook of Language and Gender*, pages 202–228.

Heather J Hether, Sheila T Murphy, and Thomas W Valente. 2016. A social network analysis of supportive interactions on prenatal sites. *Digital Health*, 2:2055207616628700.

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

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1373–1378.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the Annual Meeting of the Assocation for Computational Linguistics (ACL)*, pages 483–488.

Eduard Hovy. 1987. Generating natural language under pragmatic constraints. *Journal of Pragmatics*, 11(6):689–719.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 448–456.

Aaron Jaech and Mari Ostendorf. 2015. What your username says about you. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2032–2037.

Hamed Khanpour, Cornelia Caragea, and Prakhar Biyani. 2018. Identifying emotional support in online health communities. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Sara Kiesler, Jane Siegel, and Timothy W McGuire. 1984. Social psychological aspects of computer-mediated communication. *American Psychologist*, 39(10):1123.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Rebecca Knowles, Josh Carroll, and Mark Dredze. 2016. Demographer: Extremely simple name demographics. In *EMNLP Workshop on NLP and Computational Social Science*, pages 108–113. Association for Computational Linguistics.

Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition ratings for 30,000 english words. *Behavior Research Methods*, 44(4):978–990.

Shyong Tony K Lam, Anuradha Uduwage, Zhenhua Dong, Shilad Sen, David R Musicant, Loren Terveen, and John Riedl. 2011. Wp: clubhouse?: An exploration of wikipedia's gender imbalance. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration (WikiSym)*, pages 1–10. ACM.

Brian N Larson. 2017. Gender as a variable in natural-language processing: Ethical considerations. In *Proceedings of the First Workshop on Ethics in Natural Language Processing*.

Martin Lea and Russell Spears. 1991. Computer-mediated communication, de-individuation and group decision-making. *International Journal of Man-Machine Studies*, 34(2):283–301.

Jia Li, Xuan Liu, and Min Sun. 2018. Research on gender differences in online health communities. *International Journal of Medical Informatics*.

Siyue Li and Bo Feng. 2014. What to say to an online support-seeker? the influence of others' responses and support-seekers' replies. *Human Communication Research*, 41(3):303–326.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 342–351. ACM.

Wendy Liu and Derek Ruths. 2013. What's in a name? using first names as features for gender inference in twitter. In *AAAI Spring Symposium: Analyzing Microtext*, pages 10–16.

Ido Liviatan, Yaacov Trope, and Nira Liberman. 2008. Interpersonal similarity as a social distance dimension: Implications for perception of others' actions. *Journal of Experimental Social Psychology*, 44(5):1256–1269.

Erina L MacGeorge, Bo Feng, and Brant R Burleson. 2011. Supportive communication. *Handbook of Interpersonal Communication*, pages 317–354.

Gabriel Magno and Ingmar Weber. 2014. International gender differences and gaps in online social networks. In *Proceedings of the International Conference on Social Informatics (SocInfo)*, pages 121–138. Springer.

Alice Marwick. 2013. Online identity. *A companion to new media dynamics*.

Michael McCandless. 2010. Accuracy and performance of Google's compact language detector. `http://blog.mikemccandless.com/2011/10/accuracy-and-performance-of-googles.html`.

Jessica Megarry. 2014. Online incivility or sexual harassment? conceptualising women's experiences in the digital age. In *Women's Studies International Forum*, volume 47, pages 46–55. Elsevier.

Kristin D Mickelson, Vicki S Helgeson, and Erica Weiner. 1995. Gender effects on social support provision and receipt. *Personal Relationships*, 2(3):211–224.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. *Artificial Intelligence*, 29(3):436–465.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association of Computational Linguistics (TACL)*, 4(1):61–74.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830.

James W Pennebaker and Lori D Stone. 2003. Words of wisdom: language use over the life span. *Journal of Personality and Social Psychology*, 85(2):291.

Tom Postmes and Russell Spears. 2002. Behavior online: Does anonymous computer communication reduce gender inequality? *Personality and Social Psychology Bulletin*, 28(8):1073–1083.

Tom Postmes, Russell Spears, and Martin Lea. 1998. Breaching or building social boundaries? side-effects of computer-mediated communication. *Communication Research*, 25(6):689–715.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. A comprehensive grammar of the english language. *London and New York: Longman*.

Stephen A Rains. 2016. Language style matching as a predictor of perceived social support in computer-mediated interaction among individuals coping with illness. *Communication Research*, 43(5):694–712.

Sravana Reddy and Kevin Knight. 2016. Obfuscating gender in social media writing. In *Proceedings of Workshop on Natural Language Processing and Computational Social Science*, pages 17–26.

Tyler Joseph Schnoebelen. 2012. *Emotions are relational: positioning and the use of affective linguistic resources*. Ph.D. thesis, Stanford University.

Chris Segrin. 2003. Age moderates the relationship between social support and psychosocial problems. *Human Communication Research*, 29(3):317–342.

Sally A Shumaker and Arlene Brownell. 1984. Toward a theory of social support: Closing conceptual gaps. *Journal of Social Issues*, 40(4):11–36.

Robert J Sigley. 1997. Text categories and where you can stick them: a crude formality index. *International Journal of Corpus Linguistics*, 2(2):199–237.

Russell Spears and Martin Lea. 1992. Social influence and the influence of the “social” in computer-mediated communication. In Martin Lea, editor, *Contexts of Computer-Mediated Communication*, pages 30–65). Harvester Wheatsheaf.

Erin L Spottswood, Joseph B Walther, Amanda J Holmstrom, and Nicole B Ellison. 2013. Person-centered emotional support and gender attributions in computer-mediated communication. *Human Communication Research*, 39(3):295–316.

Cong Tang, Keith Ross, Nitesh Saxena, and Ruichuan Chen. 2011. What’s in a name: A study of names, gender inference, and gender behavior in facebook. In *International Conference on Database Systems for Advanced Applications*, pages 344–356. Springer.

Simone Teufel. 2000. *Argumentative zoning: Information extraction from scientific text*. Ph.D. thesis, University of Ediburgh.

Alan Vaux. 1985. Variations in social support associated with gender, ethnicity, and age. *Journal of Social Issues*, 41(1):89–110.

Rob Voigt, David Jurgens, Vinodkumar Prabhakaran, Dan Jurafsky, and Yulia Tsvetkov. 2018. Rtgender: A corpus for studying differential responses to gender. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Svitlana Volkova, Yoram Bachrach, Michael Armstrong, and Vijay Sharma. 2015. Inferring latent user properties from texts published in social media. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 4296–4297.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 90–94. Association for Computational Linguistics.

Yi-Chia Wang, Robert Kraut, and John M Levine. 2012. To stay or leave?: the relationship of emotional and informational support to commitment in online health support groups. In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW)*, pages 833–842. ACM.

Zuoming Wang, Joseph B Walther, and Jeffrey T Hancock. 2009. Social identification and interpersonal communication in computer-mediated communication: What you do versus who you are in virtual groups. *Human Communication Research*, 35(1):59–85.

Barry Wellman and Scot Wortley. 1990. Different strokes from different folks: Community ties and social support. *American Journal of Sociology*, 96(3):558–588.

Kevin Wright. 2002a. Motives for communication within on-line support groups and antecedents for interpersonal use. *Communication Research Reports*, 19(1):89–98.

Kevin Wright. 2002b. Social support within an online cancer community: An assessment of emotional support, perceptions of advantages and disadvantages, and motives for using the community from a communication perspective. *Journal of Applied Communication Research*, 30(3):195–209.

Kevin B Wright, Sally B Bell, Kevin B Wright, and Sally B Bell. 2003. Health-related support groups on the internet: Linking empirical findings to social support and computer-mediated communication theory. *Journal of Health Psychology*, 8(1):39–54.

Detecting Gang-Involved Escalation on Social Media Using Context

Serina Chang[1], Ruiqi Zhong[1], Ethan Adams[1], Fei-Tzin Lee[1], Siddharth Varia[1],
Desmond Patton[2], William Frey[2], Chris Kedzie[1], and Kathleen McKeown[1]
[1]Department of Computer Science, Columbia University
[2]School of Social Work, Columbia University

`{sc3003, rz2383, ea2678, fl2301, sv2504, dp2787, w.frey}@columbia.edu`
`{kedzie, kathy}@cs.columbia.edu`

Abstract

Gang-involved youth in cities such as Chicago have increasingly turned to social media to post about their experiences and intents online. In some situations, when they experience the *loss* of a loved one, their online expression of emotion may evolve into *aggression* towards rival gangs and ultimately into real-world violence. In this paper, we present a novel system for detecting Aggression and Loss in social media. Our system features the use of domain-specific resources automatically derived from a large unlabeled corpus, and contextual representations of the emotional and semantic content of the user's recent tweets as well as their interactions with other users. Incorporating context in our Convolutional Neural Network (CNN) leads to a significant improvement.

1 Introduction

In cities such as Chicago, gang-involved youth have increasingly turned to social media to post about their experience, often expressing grief when friends or family members are shot and killed. As grief turns to anger, their posts turn to retribution and ultimately to plans for revenge (Patton et al., 2018b). Research in this space has shown that online posts often affect life in the real world (Moule et al., 2013; Patton et al., 2013; Pyrooz et al., 2015; Patton et al., 2016, 2017a). In some communities, violence outreach workers manually scour online spaces to identify such possibilities and intervene to diffuse situations. A tool that identifies Aggression or Loss posts could help them filter irrelevant posts, but resources to develop a tool like this are scarce.

In this paper, we present automatic approaches for constructing resources and context features in this domain, and apply them to detecting Aggression and Loss in the social media posts of gang-involved youth in Chicago. We exploit both a small labeled dataset (4,936 posts) and a much larger unlabeled dataset (approximately 1 million posts), which we constructed using a method that enabled us to gather Twitter posts representative of the community we study. We incorporate our approaches into a CNN system, as well as a Support Vector Machine (SVM) to match the architecture of prior work, thus enabling analysis of the impact in different frameworks[1].

Key features of our system are the use of domain-specific word embeddings and a lexicon automatically induced from our unlabeled dataset. When classifying an individual tweet, our system considers the content and emotional impact of the tweets in the author's recent history. If applicable, our system additionally takes into account a model of the pairwise interactions between the author and other users in the tweet referenced via either retweet or mention.

We compare our approaches with previous work that used a smaller dataset (800 tweets) and hand-curated resources with an SVM (Blevins et al., 2016). By integrating our induced domain-specific and context information in a CNN, we achieve a significant increase over their reported results.

Our contributions include:

- A new labeled dataset, six times larger than that of prior work;
- Domain-specific resources, automatically induced from our constructed unlabeled dataset;
- Context features that capture semantic and emotion content in the user's recent posts as

---

[1] We will make tweet IDs for the data available to researchers who sign an MOU specifying their intended use of the data and their agreement with our ethical guidelines. Contact Serina Chang (sc3003@columbia.edu) or Kathleen McKeown (kathy@cs.columbia.edu). Our code is available at https://github.com/serinachang5/contextifier.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 46–56
Brussels, Belgium, October 31 - November 4, 2018.

well as their interactions with other users in the dataset.

Our approach brings us one step closer to building a useful tool that can help reduce gang violence in urban neighborhoods. In the remainder of the paper, we present related work, the dataset that we used, and our methodology. We conclude with an error analysis and a discussion of the impact of our contributions.

2 Related Work

Researchers have begun to explore how online data can be used to help prevent gun violence. Pavlick et al. 2016 are creating the Gun Violence Data Base by crowdsourcing annotations on newspaper articles that report on gun violence, labeling the sections of text that report on incidents, the shooter, and the victim. Researchers have also explored identifying deaths from police shootings with semi-supervised methods for both CNNs and logistic regression (Keith et al., 2017) and found that logistic regression using a soft-labeling approach gave the best results. Researchers studying gun control issues analyzed social media for posts related to any issue around guns in the year following the Sandy Hook elementary school shooting (Benton et al., 2016) and argued that online media can be used to understand trends in gun violence and gun-related behaviors (Ayers et al., 2016).

Closely related research aims to automatically identify gang members' Twitter profiles (Balasuriya et al., 2016). After collecting profiles using bootstrapping, they trained different classifiers on the tweets and meta-information about the authors. Further research analyzes the social networks of gangs (Radil et al., 2010) and predicts gang affiliation based on the analysis of graffiti style features (Piergallini et al., 2014).

The most relevant work in automatically analyzing social media posts by gang-involved youth is that of Blevins et al. 2016. The labeled dataset that Blevins and collaborators used is extremely challenging, in part due to its size, but also because it contains text in a particular dialect of English – African American English (AAE) – which has very little core NLP tool support. Other research investigating the development of tools for understanding AAE in social media (Blodgett et al., 2016) shows that existing tools (e.g., dependency parsers) perform poorly on this language. Previous work by Patton on a subset of our dataset notes that due to the linguistic style, tweets from gang-involved youth in Chicago can be challenging for outsiders to interpret and thus are often open to misinterpretation and potential criminalization (Patton et al., 2017b).

The challenges of interpreting our data are further compounded by the usual difficulties with Twitter data. Twitter data is sometimes handled by translating it to Standard American English (SAE) through the use of a phrasebook. The NoSlang Slang Translator (NoSlang, 2018b), and the accompanying NoSlang Drug Slang Translator (NoSlang, 2018a), have been used in other tasks to translate social media communication (Sarker et al., 2016), (Han and Baldwin, 2011).

To engineer features for an SVM classifier, Blevins et al. 2016 learned a part-of-speech (POS) tagger for their data and constructed a word level translation phrasebook to map emojis and slang to the Dictionary of Affect in Language (DAL) in order to identify their emotion.

In contrast to Blevins' translation approach, we leverage our large unlabeled dataset to automatically induce resources, such as word embeddings, that function well within the domain of our task. Previous research on domain-specific word embeddings includes work in cybersecurity (Roy et al., 2017), disease surveillance (Ghosh et al., 2016), and construction (Tixier et al., 2016). These domain-specific word embeddings tend to improve performance on tasks within that domain.

Context has been used in previous research on detecting hate speech in social media. Qian et al. 2018 found significant improvements by collecting the entire history of a user's tweets and feeding them to a encoder to create an intra-user representation, which was used as input to a Bidirectional LSTM. They also used a representation of tweets similar to the tweet being classified. While their approach captures a user profile based on everything the user has posted, in our approach we investigate how the recent history of tweets and interactions with others can improve classification. Others also make use of a user profile, though not one learned from unlabeled data (Dadvar et al., 2013).

3 Data

Our dataset consists of two parts: first, a collection of 4,936 tweets authored or retweeted by Gakirah Barnes, a powerful female Chicago gang member, and her top communicators, as well as additional Twitter users in the same demographic, annotated by social work researchers who have been studying Gakirah and the associated Chicago gangs. Second, we use a much larger collection of approximately one million unlabeled tweets automatically scraped from 279 users in the same social network. This social network is comprised of 214 users snowball-sampled from Gakirah Barnes' top 14 communicators. Traditionally, snowball sampling has been used to recruit hard-to-reach research subjects (Atkinson and Flint, 2001) and we have adapted it for social media. The remaining 65 users were added to this network by retaining those with the highest IQI score [2] from the full list of Gakirah's Twitter followers. Our tweets thus form a representative sample of Twitter dialogue between youth from Chicago neighborhoods with high levels of gang activity during that time period.

Table 1: Example tweets

No.	**Tweet Text**	**Label**
1	#FreeDaDommmmm [URL]	Loss
2	Damn juss peeped shorty on tha news out here @USER ..smh.. crazyy.. #RIPShorty	Loss
3	I'm smokin on Dat DMoney man Im high as fuck	Aggress
4	Lost Ty to Sum Fuck Shit dont Fuck around wit Fuck rounds n u a type of Niggas Ion fuck wit 👿💯	Loss
5	My bro Mooki thirsty he jus wana 👏 sum 💩 🔫 😈 💯	Aggress

The social work researchers performed a detailed, qualitative analysis of a subset of the dataset, with a focus on analyzing how context influences determination of a label. For example, they note that an aggressive tweet may reference a previous event, and will often use coded language to do so. Since much of the language used in our data differs significantly from standard American English, local youth active in similar environments served as consultants to answer questions about the language, as they were able to interpret the slang terms present in these tweets. The social work researchers conducted a fine-grained analysis using an online tool for annotation, identifying insults, threats, bragging, hypervigilance and challenges to authority, all of which were collapsed into a single category, Aggression. Posts including distress, sadness, loneliness and death were collapsed into the category Loss. The Other category includes discussion of other aspects of their life, such as friendships, relationships, drugs, general conversations, and happiness. We developed our system (as did Blevins et al. 2016) on the collapsed labels, as the task is difficult even with three-way categorization.

Each tweet in a subset of the entire dataset consisting of 3,000 tweets was reviewed by two different annotators. Inter-rater reliability between raters was tracked, with dissimilar annotations flagged for further review. Flagged tweets were further analyzed by the social work researchers, which included youth from Chicago who currently live in the same community as, or an adjacent one to, that in which the deceased Gakirah Barnes resided, to adjudicate disagreement. Among the set of tweets coded by two annotators, inter-annotator agreement on the Aggression class was high even before adjudication, with a Cohen's kappa coefficient of .94; agreement on the Loss class was somewhat lower, with a Cohen's kappa of .83. Examples of labeled Twitter posts from Gakirah and her followers are shown in Table 1 [3].

In order to mitigate potential issues with training and test data being drawn from different time periods or having different distributions of labels, we shuffled our data and drew stratified samples with equal distribution across classes for our training, validation, and test sets for each of the cross validation folds, using 64%, 16%, and 20% of our data for each respectively. The Aggression and Loss classes are relatively small, reflecting their low distribution in real life: we have only 329 Aggression tweets and 734 Loss tweets, with the Other class comprising the remaining 3,873 tweets.

4 Methods

We approach this classification task using a standard CNN classifier architecture (Kim, 2014; Collobert et al., 2011) as our starting point. We initially experimented with both character and word level CNNs but found the word level to be 1.6 macro-F1

---

[2] https://www.brookings.edu/wp-content/uploads/2016/06/isis_twitter_census_berger_morgan.pdf

[3] Our data was scraped from publicly available posts and was determined exempt by our organization's IRB. User names are replaced with USER in the table, and text has been modified to render tweets unsearchable.

points better than the character level, so we only include the word level here. We leveraged the unlabeled corpora by constructing domain-specific embeddings and a lexicon that better fit our unique and low-resource domain. We then integrated our domain-specific resources into the CNN to represent the given tweet as well as to represent context features. Our context features represent a window of the user's recent tweets as well as the interactions of the author with other users via references in their tweets.

4.1 Domain-Specific Resources

We exploited the large unlabeled corpus to build two domain-specific resources for this task: word embeddings and a task-specific lexicon.

4.1.1 Word Embeddings

Word embeddings have proven useful in representing the semantic content of sentences. The semantic representation of a word by its associated embedding, however, depends on its usage in the corpus the embedding was trained on, and so off-the-shelf word embeddings do not always adapt well to tasks with a unique domain (Roy et al., 2017), (Ghosh et al., 2016), (Tixier et al., 2016). Thus, we were motivated to use our unlabeled corpus to create domain-specific word embeddings. We used the Word2Vec (Mikolov et al., 2013) CBOW model to train the embeddings which is the default training algorithm available in Gensim [4]. We used a window size of 5 words with a minimum word count of 5 to train $w \in \mathbb{R}^{300}$. The CBOW model was trained for 20 epochs.

4.1.2 Computing a Lexicon of Aggression and Loss

Given the domain-specific nature of our users' language, we could not rely on standard NLP lexicons to represent emotion in their tweets. For our task, the two emotions of interest are Aggression and Loss. Previous work (Blevins et al., 2016) used a phrasebook to translate the domain-specific words of their corpus to Standard American English so that they could access emotion in the Dictionary of Affect in Language (DAL) (Whissell, 2009), but this approach does not generalize to capture new words.

We therefore adapted the SENTPROP algorithm (Hamilton et al., 2016) to automatically induce a lexicon of Aggression and Loss from our unlabeled corpus. The SENTPROP algorithm constructs a lexical graph out of the word embeddings, then propagates labels from the seed sets over the unlabeled nodes via a random walk method. The resulting output for each word indicates the probability of a random walk from the seed set landing on that node. We chose SENTPROP as an induction method because it performs especially well for domain-specific corpora, and it is resource-light and interpretable.

We created word embeddings by employing an SVD-based method that was reported by the SENTPROP authors to perform optimally with their algorithm. We first constructed the positive point-wise mutual information matrix, M^{PPMI}, over the unlabeled corpus, then computed singular value decomposition (SVD) to derive $M^{PPMI} = U\Sigma V^{\intercal}$. The word embedding for word w_i was thus given by U_i, truncated to a standard length of dimension 300. To construct our seed sets, we asked our annotators to consider words for Loss and Aggression which they associated most strongly with each class. They generated a set of 29 words for Aggression and a set of 40 words for Loss, which we include in our appendix.

We ran SENTPROP with our SVD-based embeddings and the seed sets from our annotators. We used the output probabilities from the random walks to map words to their association with Aggression and Loss, thus forming our lexicon of Aggression and Loss. Finally, we scaled the probabilities per class to mean 0 and variance 1.

4.2 Context Features

Our context features utilize the domain-specific resources that we induced from the unlabeled corpora. To capture context, we first considered the author's recent history, separately exploring representations by our domain-specific word embeddings and by the SENTPROP lexicon (SPLex). If applicable, we also considered the interactions between the author and other users who were referenced in the tweet, either via retweet or mention.

4.2.1 User History

To obtain the user's recent history, we ordered all the tweets chronologically and bucketed them by author. Thus, for any given tweet occurring at time t, a_t, we were able to retrieve previous tweets $a_{t-1}, a_{t-2}, \ldots$ by that user. We treated recent history as a sliding window and fetched tweets within

---

[4] `https://radimrehurek.com/gensim/models/word2vec.html`

the past d days from when the current tweet was tweeted, such that recent history tweets would be the set $\{a_{t-1}, \ldots, a_{t-k}\}$, where $t - k < d$.

To represent the tweets within the context of recent history, we first combined word level representations into tweet level, then tweet level representations into context level. At each stage of combination, we tried both summing and averaging. Thus, our recent history representations were built by aggregating either word embeddings or SPLex scores, which maintained their dimensionality of 300 or 2, respectively.

We also considered three types of tweets that would be relevant to a user. The user's own tweets (`SELF`) would always be relevant; we experimented with also including tweets where the user was retweeted (`RETWEET`) and tweets where the user was mentioned (`MENTION`). We included these parameters as additional sources of context because a user's tweet may be a response to a recent mention or retweet from another user.

We also experimented with weighting the most recent tweets more heavily than further tweets within the recent history window. This became especially important when we experimented with larger windows of a month or more, since tweets from a few days ago are more likely to be related to the current tweet than tweets from a few weeks ago. To model this diminishing relevance, we introduced a weighting protocol with a variable half-life where weights decay exponentially over time. The parameter we tuned was the half-life ratio r, which is the proportion of the window size d that corresponds to the window's half-life. Then, before combining tweet level representations into context level, we multiplied each tweet representation b_i by its weight, $2^{-\frac{\Delta t}{f}}$, where $\Delta t = t - i$ is the distance in days between the context tweet a_i and the current tweet a_t, and $f = d * r$ is the half life.

4.2.2 User Interactions

As an additional context feature, we modeled the pairwise interactions between users. To identify interactions, we iterated through our unlabeled and labeled corpora and checked which users were involved in each tweet. We counted a user as involved in a tweet if they posted the tweet or were referenced via retweet or mention. For each pair of users, we aggregated all their tweets of mutual involvement into one document and averaged the document's word embeddings to create a representation of their pairwise interactions in $\mathbb{R}^{300}$.

5 Experiments

We experimented with the efficacy of our domain-specific resources, the impact of different context parameters, and the contribution of context to predicting Aggression and Loss.

5.1 Corpus pre-processing

For word level models, we preprocess each tweet by: i) lowercasing every character, ii) replacing every user mention and url with special tokens "user" and "url", iii) considering each emoji an individual token, whether space separated or not, and iv) removing emoji modifiers to reduce sparsity, just as we used lowercasing. We select the top 40K tokens based on frequency, replacing the remaining tokens with "UNKNOWN". We zero-pad or trim tweets so that tweet length will be 50 when passed to our CNN model. Similarly, we only consider users who occur (as author, source of retweet, or in mention) in the labeled and unlabeled corpus at least twice, resulting in 35,656 users in total.

We extract the author of the tweets from metadata, and user mentions and original posters of retweets from the Twitter text, based on their Twitter display name. We used Twitter display name rather than user ID because we cannot collect user ID for interaction features.

5.2 CNN Architecture

For this 3-way classification task, we train two models; the first model predicts whether a tweet has the Aggression label and the second predicts for Loss. Each model maps a sequence of tokens to a probability value for a class. Here we define the architecture of our CNN model. Our input c is a token index sequence of length 50. We map each token index to a vector $\in \mathbb{R}^{300}$ with a trainable embedding matrix, followed by dropout 0.5. We apply a 1D Convolutional layer with kernel sizes 1 and 2, filter size 200 each, to the embedded token sequence, followed by ReLU activation, max pooling and dropout 0.5. We concatenate the output of max pooling for kernel sizes 1 and 2, stack another dense layer h with dimension 256, and connect the output of h to the final single output unit with sigmoid activation.

In the prediction phase, for each data point, we classify it as Aggression if the the first model produces the probability score above threshold t_A. If it is not predicted as Aggression, then we classify it as Loss if the second model produces a score above a threshold t_L. The remaining tweets are classified as Other. t_A and t_L are tuned on the validation set.

We incorporate context information into the neural network in the following way. Each type of context feature takes the form of a real vector: both word embedding user history and word embedding user interaction features are in $\mathbb{R}^{300}$, and SPLex user history features are in $\mathbb{R}^2$. We concatenate these feature vectors with the last layer h before the final classification output.

5.3 SVM Baseline

We used as our baseline method a linear-kernel SVM classifier as used by Blevins et al. 2016. We obtained code from the authors and trained on our larger dataset. In this method, after basic preprocessing is performed to replace urls and user mentions with special tokens, unigram, bigram, part-of-speech tag, and emotion features are extracted. Feature selection is performed to prune the feature space. The part-of-speech tagger used in Blevins et al. 2016 was developed for use on this domain; emotion features are computed using scores for each tweet word taken from the Dictionary of Affect in Language (DAL). We performed gridsearch to re-tune the loss function, the regularization penalty type, and the penalty parameter C, but found that the original settings for these parameters still performed best even on our new development set. We also tuned the class weights used: while the model performed best on the original data with balanced class weights, we found that less extreme balancing performed better here (weights 2, 1, and 0.12 for Aggression, Loss, and Other, respectively).

While we retrained the SVM on our new training set, we did not modify the additional components used for feature selection such as the phrase table or the specialized part-of-speech tagger, as we had no additional data available for this. This indicates the difficulty of generalizing to new data with unseen vocabulary, and is one of the disadvantages of using manually-created specialized feature sets such as these.

5.4 Domain Experiments

In order to test the efficacy of our domain-specific word embeddings, we compared them with a number of other embedding types. Our baseline method was Pennington et al. 2014's GloVe embeddings pretrained on a general Twitter dataset, available from their website[5]. We trained a parallel set of word embeddings on the African American English (AAE) corpus of around 1.1 million tweets provided by Blodgett et al. 2016, and another set on a corpus of a location-specific set of tweets that we scraped, drawn from users who posted from a specific area within the South Side of Chicago where the gangs we study are based. We also compared performance with a randomly initialized word embedding matrix.

5.5 Context Experiments

We first explored the impact of the user history parameters, tuning them separately for representations by our domain-specific word embeddings and by SPLex. We kept these representations separate because we expected them to capture different types of context: word embeddings should capture the semantic content of the user's history, while SPLex scores should capture something closer to the user's emotional state leading up to the tweet.

With each representation, we experimented with summing versus averaging word embeddings to yield a tweet level representation, and similarly experimented with summing and averaging from tweet embeddings to context level representations. We varied the size of the context window, d, trying 2 days, 1 week, 1 month, 2 months, and 3 months. We also varied the half-life ratio, r = .25, .5, .75, or no weighting. Lastly, we tried including different types of posts in the user history.

Once we tuned the user history parameters, we experimented with adding our context features (user history and user interactions) to the best tweet level model we could achieve without context. For our CNN, our best tweet level model used our domain-specific word embeddings as pretrained weights for the embedding layer (CNN-DS in Table 3). To evaluate the impact of our resources in different frameworks, we additionally experimented with the contribution of context in an SVM. The best tweet level SVM included the averaged domain-specific word embeddings and summed SPLex scores of the tokens in the tweet (SVM-DS).

6 Results and Discussion

We report results comparing different embeddings and comparing parameters for context. We use the best results from these experiments to produce our

---

[5] `https://nlp.stanford.edu/projects/glove/`

Figure 1: Diagram of our steps to generate domain-specific and context features for our neural net system.

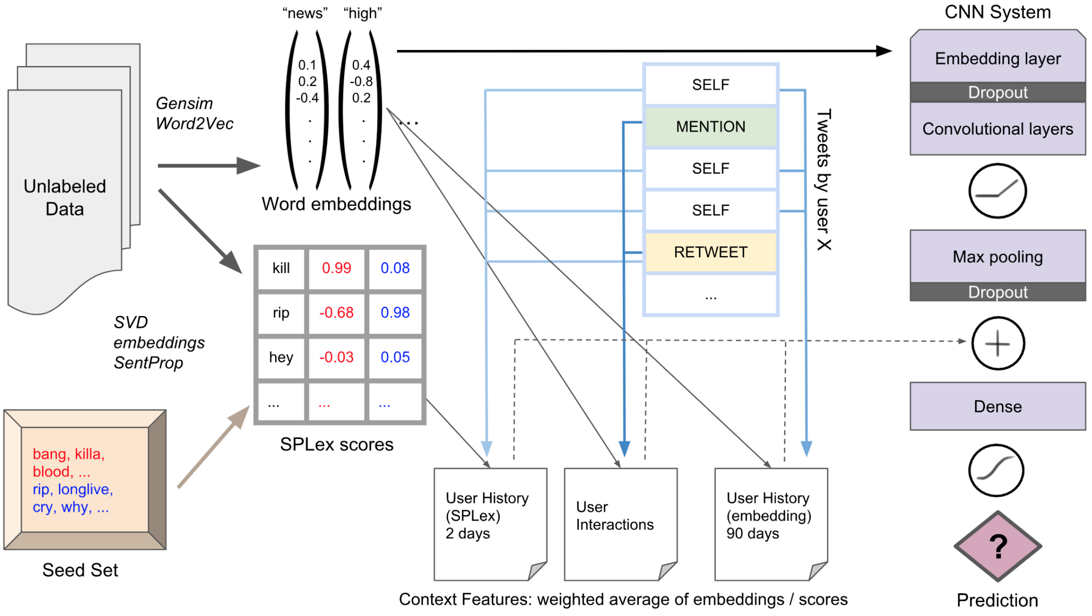

final systems in the SVM and CNN frameworks. The best resulting architecture for the CNN framework is illustrated in Fig. 1.

6.1 Comparison of Embeddings

Experiments were performed using five-fold cross-validation over the labeled data and were repeated five times for each fold to account for variance between runs. Reported F-scores, shown in Table 2, are averaged across runs and across folds.

Word embeddings trained on our unlabeled corpus outperformed other embeddings by over 4 points. Related datasets such as the location-specific or AAE corpus did not provide helpful semantic information, as their embeddings did not even beat random initialization. This was not an effect of corpus size, since these corpora contained 800,000 and 1.1 million tweets, respectively, compared to the 1 million in our unlabeled corpus. Thus, we attribute the difference to the importance of deriving embeddings directly from our community of interest, demonstrating that the language of our community is more specific than AAE in general and that our snowballing method was able to capture a better representation of user language than a location driven method.

6.2 User History Parameters

Experiments were performed using five-fold cross-validation and F-scores computed as in the word embedding experiments. We found that user history represented by domain-specific word embeddings performed optimally when we averaged from word to tweet level and from tweet to context level. The best window size was $d = 90$ days, including only `SELF` posts, and using a half-life ratio of $r = 0.25$. For user history represented by SPLex, we found the best method of combination to be summing, at both the word and tweet level. We hypothesize this is because summing captures not only the presence but also the number or density of highly indicative Aggression or Loss words posted by the user over the context window. The best window size was $d = 2$ days, including both `SELF` and `RETWEET` posts, without half-life weighting.

Our approach was designed to implement and

Table 2: Results comparing different embeddings with CNN. GN refers to Google News, LS to location specific embeddings, GT to Glove Twitter embeddings, and DS to our domain specific embeddings. A, L and O refer to Aggression, Loss, and Other respectively.

Embeddings Type	F1			Macro F1
	A	L	O	
GN	27.9	66.6	86.9	60.5
AAE	27.3	69.8	86.5	61.2
LS	31.3	68.3	87.9	62.5
Random Init.	29.3	70.5	88.9	62.9
GT	29.0	71.1	89.0	63.0
DS	**37.9**	**73.4**	**90.3**	**67.20**

Table 3: Comparison of different models. The below pairs of algorithms achieve statistical significance $p < 0.002$ for each class (the higher performing algorithm comes first): i) CNN-Context vs. CNN-DS; ii) CNN-DS vs. SVM-Retrained; iii) SVM-Context vs. SVM-DS. SVM-Context outperforms SVM-Retrained in the Aggression class by a robust margin (5 points).

Model	Aggression			Loss			Other			Macro F1
	P	R	F	P	R	F	P	R	F	
SVM-Retrained(baseline)	36.4	31.3	33.7	73.7	68.8	71.2	89.8	**92.0**	90.9	65.3
SVM-DS	32.4	38.9	35.4	66.9	72.9	69.8	90.8	87.7	89.2	64.8
SVM-Context	35.0	43.7	38.8	68.6	74.0	71.2	**91.6**	88.2	89.9	66.6
CNN-DS	35.7	41.1	38.2	**78.9**	70.3	74.3	90.7	91.4	91.0	67.9
CNN-Context	**38.3**	**46.4**	**42.0**	78.8	**73.2**	**75.9**	91.3	91.7	**91.5**	**69.8**

test previous insights about the domain, particularly that context plays a role in the interpretation of posts. The short time frame for SPLex user history corresponds with the 2 day window found in Patton et al. 2018b's research and reflects the fact that emotional states may fluctuate often and within a certain number of days. In contrast, word embeddings improved consistently as we extended the context window from 2 days to 90 days. Since word embedding user history is meant to capture the user's semantics, a larger window size means the representation can be drawn from more tweets, and thus reflects a more representative sample of the user's semantics around this time period.

6.3 Comparison of Best Systems

To develop a more stable measurement of comparison between different systems, we create four independent sets of 5-fold cross validation splits on our data set (altogether 20 folds); to account for randomness in neural net training, we train each neural net model 5 times and take the majority vote of the predictions. For each class, we calculate the statistical significance of F-score based on the predictions on the concatenated test sets of all 20 folds using the Approximate Randomization Test (Riezler and Maxwell 2005) with the Bonferroni correction for multiple comparisons. Results are shown in Table 3.

Adding context contributed to a significant improvement in both the CNN and SVM frameworks, demonstrating the independent value of our context features over domain-specific resources. For contrast, we also compared our context features with user profiles built from averaging the word embeddings in all of the user's tweets. Our pairwise and user history features outperformed user profiles by .7 points, demonstrating that it is valuable to provide dynamic representations of users that can adjust to their recent posts or their interactions with other users, as opposed to stereotyping their overall behavior.

Additionally, we compare the impact of our domain-specific resources to those used by Blevins et al. (2016). In particular, we expect that their emotion scores will not generalize to the new vocabulary in our large unlabeled corpus (see Section 4.1). Our domain-specific resources alone without context raise our SVM to comparable performance with the Blevins et al. retrained baseline, and the resources push our CNN without context over this baseline. This demonstrates that our automatic methods can do as well as if not better than phrasebook methods, and they are significantly more efficient to generate.

7 Error Analysis

In this section we provide an analysis of the tradeoffs of each classifier by analyzing some of the examples in Table 1.

Context vs non-context CNN. Our best CNN - a system which incorporated context - was able to correctly predict tweets 3 and 4, whereas our baseline using only our pretrained Word2Vec embeddings was not. Correctly classifying tweet 4 relies on the knowledge that the referenced user, DMoney, is a deceased member of a rival gang of the poster. In tweet 3, the poster is saying that he has seen Gakirah's death on the news; this is an expression of loss.

Domain-specific vocabulary. Our CNN trained on domain-specific word embeddings is able to correctly classify tweet 5, while the one trained on Twitter word embeddings did not pick up the aggressive content. This user is talking about how their friend is ready to kill someone. This tweet contains the word *thirsty* but in this domain-specific context it means being ready and having an urge (although it would not always refer to killing).

Hashtags and character sequences. Despite their strengths, both our best CNN and our best SVM classifiers were still unable to correctly classify some of the trickier cases. There were certain types of tweets they were categorically unable to recognize: tweet 1 features a hashtag that refers to an incarcerated acquaintance of the poster, but as both our CNN and SVM models operate at the word level, this tag would have appeared simply as a rare or unknown token to them.

Anger miscategorized as Aggression. At times, the classifier categorized posts that express anger as Aggression. For example, in tweet 4 the author uses profanity to express grief related to the loss of a friend. In addition, the devil face emoji, which is sometimes used to express aggression, is also used in the context of anger. While the best CNN model managed to correctly predict this as Loss, the SVM miscategorized it as Aggression.

8 Ethics

Our ethics guidelines include just treatment of the users who provide our data, removal of identifying information for publication, and the inclusion of Chicago-based community members as domain experts in the analysis and validation of our findings.

There are risks involved with detecting Aggression and Loss in social media data using automatic detection systems. These risks include possible misidentifications of tweets, increased police involvement, and loss of privacy, which all have the potential to harm marginalized communities and people. Our mitigation strategies begin by partnering with violence prevention organizations and incorporating domain experts (Frey et al., 2018) to ensure the highest ethical standards for interpreting social media posts and for the dissemination and use of our research for violence prevention. Through insights gained from these partnerships, we developed our own risk mitigation strategies: de-identifying each tweet and rendering it unsearchable through textual modification without altering meaning; encrypting our social media corpus to protect user identities; and relying on violence prevention organizations' expertise in deciding if and when to involve law enforcement to prevent the unethical use of our data (e.g., hyper-surveillance of communities of color).

9 Conclusion and Future Work

Our approach shows that integrating emotions and semantic content of a user's recent posts is an important component for the task of predicting Aggression and Loss in social media posts of gang-involved youth. Furthermore, using domain-specific embeddings and an Aggression-Loss lexicon induced from a corpus of language constructed to represent our specific community of users is also critical to success. Our experiments reveal that our snowballing technique is more effective than a location based approach and that fitting our community is more complex than resorting to their demographic, as captured in the AAE corpus of Blodgett et al. (2016).

Our work has real life implications for the use of machine learning to identify unique characteristics in social media data that may indicate the process by which gun violence may occur (Patton et al., 2018a). Our partnership between computer scientists, social work researchers and practitioners has advanced plans to create applications to help outreach workers in Chicago identify factors related to potential violence, potentially allowing them to prevent and intervene in aggressive online activity. The tool, which would be co-created with community stakeholders, would enable quick scanning of large quantities of social media posts that outreach workers would be unable to perform manually.

We expect our methods to be generalizable because we compute embeddings and lexicons from neighborhood-specific data and do not rely on large, hand-crafted resources such as dictionaries. However, we hope to test generalizability in future work by applying our methods to other gang-related corpora, because there is variation in language, local concepts, and behavior across gangs. In the future, we are also interested in further experimenting with the context features introduced in this work; for instance, by extending our pairwise interaction features to take into account direction between users. Finally, we intend to explore other types of context, such as reference to specific events that may trigger the emotions of either Aggression or Loss.

10 Acknowledgements

This research is supported in part by DARPA contract 55630053. The authors also thank the anonymous reviewers for their thoughtful comments.

References

R. Atkinson and J. Flint. 2001. Accessing hidden and hard-to-reach populations: Snowball research strategies. *Social research update* 33:1–4.

J. W. Ayers, B. M. Althouse, E. C. Leas, T. Alcorn, and M. Dredze. 2016. Big media data can inform gun violence prevention. In *Bloomberg Data for Good Exchange*.

L. Balasuriya, S. Wijeratne, D. Doran, and A. Sheth. 2016. Finding street gang members on twitter. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*. pages 685–692.

A. Benton, B. Hancock, G. Coppersmith, J. W. Ayers, and M. Dredze. 2016. After sandy hook elementary: A year in the gun control debate on twitter. In *Bloomberg Data for Good Exchange*.

T. Blevins, R. Kwiatkowski, J. Macbeth, K. McKeown, D. Patton, and O. Rambow. 2016. Automatically processing tweets from gang-involved youth: Towards detecting loss and aggression. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 2196–2206.

S. L. Blodgett, L. Green, and B. O'Connor. 2016. Demographic dialectal variation in social media: A case study of african-american english. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1119–1130. https://aclweb.org/anthology/D16-1120.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

M. Dadvar, D. Trieschnigg, R. Ordelman, and F. de Jong. 2013. Improving cyberbullying detection with user context. In *European Conference on Information Retrieval*. Springer, pages 693–696.

W. R. Frey, D. U. Patton, M. B. Gaskell, and K. A. McGregor. 2018. Artificial intelligence and inclusion: Formerly gang-involved youth as domain experts for analyzing unstructured twitter data. *Social Science Computer Review* https://doi.org/0894439318788314.

S. Ghosh, P. Chakraborty, E. Cohn, J. S. Brownstein, and N. Ramakrishnan. 2016. Designing domain specific word embeddings: Applications to disease surveillance. *CoRR* abs/1603.00106. http://arxiv.org/abs/1603.00106.

W. L. Hamilton, K. Clark, J. Leskovec, and D. Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. *EMNLP* abs/1606.02820. https://arxiv.org/abs/1606.02820.

B. Han and T. Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 368–378. http://dl.acm.org/citation.cfm?id=2002472.2002520.

K. A. Keith, A. Handler, M. Pinkham, C. Magliozzi, J. McDuffie, and B. T. O'Connor. 2017. Identifying civilians killed by police with distantly supervised entity-event extraction. In *EMNLP*.

Y. Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

R. K. Moule, D. C. Pyrooz, and S. H. Decker. 2013. From 'What the f#@% is a Facebook?'to 'Who doesn't use Facebook?': The role of criminal lifestyles in the adoption and use of the Internet. *Social Science Research* 42(6):1411–1421.

NoSlang. 2018a. Noslang drug slang translator. Accessed: 2018-05-20. http://www.noslang.com/drugs/dictionary.php.

NoSlang. 2018b. Noslang slang translator. Accessed: 2018-05-20. http://www.noslang.com/.

D. U. Patton, R. D. Eschmann, and D. A. Butler. 2013. Internet banging: New trends in social media, gang violence, masculinity and hip hop. *Computers in Human Behavior* 29(5):A54 – A59. https://doi.org/10.1016/j.chb.2012.12.035.

D. U. Patton, R. D. Eschmann, C. Elsaesser, and E. Bocanegra. 2016. Sticks, stones and facebook accounts: What violence outreach workers know about social media and urban-based gang violence in chicago. *Computers in Human Behavior* 65:591 – 600. https://doi.org/https://doi.org/10.1016/j.chb.2016.05.052.

D. U. Patton, S. Patel, J. Hong, M. Ranney, M. Crandall, and L. Dungy. 2017a. Tweets, gangs, and guns: A snapshot of gang communications in detroit 32.

D. U. Patton, N. Sanchez, D. Fitch, J. Macbeth, and P. Leonard. 2017b. I know god's got a day 4 me: Violence, trauma, and coping among gang-involved twitter users. *Social Science Computer Review* 35(2):226–243. https://doi.org/10.1177/0894439315613319.

D.U. Patton, K. McGreggor, and G. Slutkin. 2018a. Youth gun violence prevention in a digital age. *Pediatrics* pages 2017–2438. https://doi.org/10.1542/peds.2017-2438.

D.U. Patton, O. Rambow, J. Auerbach, K. Li, and W. Frey. 2018b. Expressions of loss predict aggressive comments on twitter among gang involved youth in chicago. *Nature Partner Journal: Digital Medicine* .

E. Pavlick, H. Ji, X. Pan, and C. Callison-Burch. 2016. The gun violence database: A new task and data set for nlp .

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

M. Piergallini, A. S. Dogruöz, P. Gadde, D. Adamson, and C. P. Rosé. 2014. Modeling the use of graffiti style features to signal social relations within a multi-domain learning paradigm. pages 107–115.

D. C. Pyrooz, S. H. Decker, and R. K. Moule Jr. 2015. Criminal and routine activities in online settings: Gangs, offenders, and the internet. *Justice Quarterly* 32(3):471–499. https://doi.org/10.1080/07418825.2013.778326.

J. Qian, M. ElSherief, E. M. Belding, and W. Y. Wang. 2018. Leveraging intra-user and inter-user representation learning for automated hate speech detection. *CoRR* abs/1804.03124. http://arxiv.org/abs/1804.03124.

S. M. Radil, C. Flint, and G. E. Tita. 2010. Spatializing social networks: Using social network analysis to investigate geographies of gang rivalry, territoriality, and violence in los angeles. *Annals of the Association of American Geographers* 100(2):307–326.

Stefan Riezler and John T Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for mt. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. pages 57–64.

A. Roy, Y. Park, and S. Pan. 2017. Learning domain-specific word embeddings from sparse cybersecurity texts. *CoRR* abs/1709.07470. http://arxiv.org/abs/1709.07470.

A. Sarker, K. O'Connor, R. Ginn, M. Scotch, K. Smith, D. Malone, and G. Gonzalez. 2016. Social media mining for toxicovigilance: Automatic monitoring of prescription medication abuse from twitter. *Drug Safety* 39(3):231–240. https://doi.org/10.1007/s40264-015-0379-4.

A. J.-P. Tixier, M. Vazirgiannis, and M. R. Hallowell. 2016. Word embeddings for the construction domain. *CoRR* abs/1610.09333. http://arxiv.org/abs/1610.09333.

C. Whissell. 2009. Using the revised dictionary of affect in language to quantify the emotional undertones of samples of natural language. *Psychological Reports* 105:509–521.

Reasoning about Actions and State Changes by Injecting Commonsense Knowledge

Niket Tandon[*], **Bhavana Dalvi Mishra**[*], **Joel Grus**,
Wen-tau Yih, **Antoine Bosselut**, **Peter Clark**
Allen Institute for AI, Seattle, WA
`{nikett,bhavanad,joelg,scottyih,antoineb,peterc}@allenai.org`

Abstract

Comprehending procedural text, e.g., a paragraph describing photosynthesis, requires modeling actions and the state changes they produce, so that questions about entities at different timepoints can be answered. Although several recent systems have shown impressive progress in this task, their predictions can be globally inconsistent or highly improbable. In this paper, we show how the predicted effects of actions in the context of a paragraph can be improved in two ways: (1) by incorporating global, commonsense constraints (e.g., a non-existent entity cannot be destroyed), and (2) by biasing reading with preferences from large-scale corpora (e.g., trees rarely move). Unlike earlier methods, we treat the problem as a neural structured prediction task, allowing hard and soft constraints to steer the model away from unlikely predictions. We show that the new model significantly outperforms earlier systems on a benchmark dataset for procedural text comprehension (+8% relative gain), and that it also avoids some of the nonsensical predictions that earlier systems make.

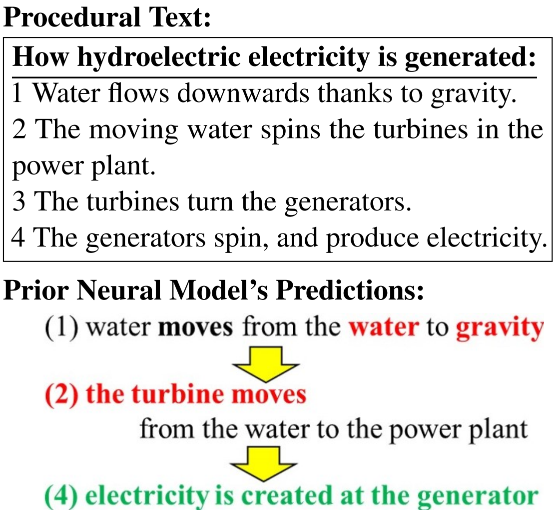

Procedural Text:

How hydroelectric electricity is generated:
1 Water flows downwards thanks to gravity.
2 The moving water spins the turbines in the power plant.
3 The turbines turn the generators.
4 The generators spin, and produce electricity.

Prior Neural Model's Predictions:

(1) water **moves** from the **water** to **gravity**

(2) the turbine moves from the water to the power plant

(4) electricity is created at the generator

Figure 1: Poor predictions (in red) made by a prior neural model (ProGlobal) applied to an (abbreviated) paragraph from the ProPara dataset. ProGlobal predicts entity locations at each sentence, but the implied movements violate commonsense constraints (e.g., an object cannot move from itself (1)) and corpus-based preferences (e.g., it is rare to see turbines move (2)).

1 Introduction

Procedural text is ubiquitous (e.g., scientific protocols, news articles, how-to guides, recipes), but is challenging to comprehend because of the dynamic nature of the world being described. Comprehending such text requires a model of the actions described in the text and the state changes they produce, so that questions about the states of entities at different timepoints can be answered (Bosselut et al., 2018).

Despite these challenges, substantial progress has been made recently in this task. Recent work – such as EntNet (Henaff et al., 2017), QRN (Seo et al., 2017b), ProLocal/ProGlobal (Dalvi et al., 2018), and NPN (Bosselut et al., 2018) – has focused on learning to predict individual entity states at various points in the text, thereby approximating the underlying dynamics of the world. However, while these models can learn to make local predictions with fair accuracy, their results are often globally unlikely or inconsistent. For example, in Figure 1, the neural ProGlobal model from Dalvi et al. (2018) learns to predict the impossible action of an object moving from itself (1), and the unlikely action of a turbine changing location (2). We observe similar mistakes in other neural models, indicating that these models have little notion of global consistency. Unsurprisingly, mistakes in local predictions compound as the process becomes longer, further reducing the plausibility of the overall result.

---

[*] Niket Tandon and Bhavana Dalvi Mishra contributed equally to this work.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 57–66
Brussels, Belgium, October 31 - November 4, 2018.

To address this challenge, we treat process comprehension as a *structured prediction task* and apply hard and soft constraints during reading. During training, our model, called ProStruct, learns to search for the most likely action sequence that is consistent with global constraints (e.g., entities cannot be destroyed after they have already been destroyed) and priors from background knowledge (e.g., turbines rarely change location). The model is trained end-to-end, with gradients backpropagating through the search path. We find that this approach significantly outperforms existing approaches on a benchmark dataset for process comprehension, mainly by avoiding the nonsensical predictions that earlier systems make.

Our contributions are twofold. First, we reformulate procedural text comprehension in a novel way: as a (neural) structured prediction task. This lets hard and soft constraints steer the model away from unlikely and nonsensical predictions. Second, we present a novel, end-to-end model that integrates these constraints and achieves state-of-the-art performance on an existing process comprehension dataset (Dalvi et al., 2018).

2 Related Work

Our work builds off a recent body of work that focuses on using neural networks to explicitly track the states of entities while reading long texts. These works have focused on answering simple commonsense questions (Henaff et al., 2017), tracking entity states in scientific processes (Dalvi et al., 2018; Clark et al., 2018), tracking ingredients in cooking recipes (Bosselut et al., 2018), and tracking the emotional reactions and motivations of characters in simple stories (Rashkin et al., 2018). Our work extends these methods and addresses their most common issues by using background knowledge about entities to prune the set of state changes they can experience as the model reads new text.

Prior to these neural approaches, some earlier systems for process comprehension did make use of world knowledge, and motivated this work. Like us, the system ProRead (Berant et al., 2014; Scaria et al., 2013) also treated process comprehension as structure prediction, using an Integer Linear Programming (ILP) formalism to enforce global constraints (e.g., if the result of event1 is the agent of event2, then event1 must enable event2). Similarly, Kiddon et al. (2015) used corpus-based priors to guide extraction of an “action graph” from recipes. Our work here can viewed as incorporating these approaches within the neural paradigm.

Neural methods for structure prediction have been used extensively in other areas of NLP, and we leverage these methods here. In particular we use a neural encoder-decoder architecture with beam search decoding, representative of several current state-of-the-art systems (Bahdanau et al., 2014; Wiseman and Rush, 2016; Vinyals et al., 2015). As our model’s only supervision signal comes from the final prediction (of state changes), our work is similar to previous work in semantic parsing that extracts structured outputs from text with no intermediate supervision (Krishnamurthy et al., 2017).

State tracking also appears in other areas of AI, such as dialog. A typical dialog state tracking task (e.g., the DSTC competitions) involves gradually uncovering the user’s state (e.g., their constraints, preferences, and goals for booking a restaurant), until an answer can be provided. Although this context is somewhat different (the primary goal being state discovery from weak dialog evidence), state tracking techniques originally designed for procedural text have been successfully applied in this context also (Liu and Perez, 2017).

Finally, our model learns to search over the best candidate structures using hard constraints and soft KB priors. Previous work in Neural Machine Translation (NMT) has used sets of example-specific lexical constraints in beam search decoding to only produce translations that satisfy every constraint in the set (Hokamp and Liu, 2017). In contrast, our work uses a set of global example-free constraints to prune the set of possible paths the search algorithm can explore. Simultaneously, a recent body of work has explored encoding soft constraints as an additional loss term in the training objective for dialogue (Wen et al., 2015), machine translation (Tu et al., 2016), and recipe generation (Kiddon et al., 2016). Our work instead uses soft constraints to re-rank candidate structures and is not directly encoded in the loss function.

3 Problem Definition

We first define the general task that we are addressing, before presenting our approach.

3.1 General Formulation

We define the task as follows. **Given:**

- A **paragraph of procedural text** S = an ordered set of sentences $\{s_1, ..., s_T\}$ describing

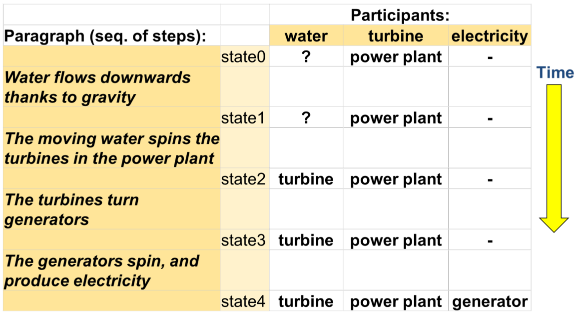

Paragraph (seq. of steps):		Participants: water	turbine	electricity
	state0	?	power plant	-
Water flows downwards thanks to gravity				
	state1	?	power plant	-
The moving water spins the turbines in the power plant				
	state2	turbine	power plant	-
The turbines turn generators				
	state3	turbine	power plant	-
The generators spin, and produce electricity				
	state4	turbine	power plant	generator

Figure 2: How the (simplified) paragraph in Figure 1 is annotated in ProPara. Each filled row shows the location of entities between each step ("?" denotes "unknown", "-" denotes "does not exist"). For example, in the last line (state4), the water is at the turbine.

a sequence of **actions**[1] about a given **topic** (a word or phrase).

- A **set of entities** $E = \{e_j\}$ representing the entities mentioned in the procedure or process. Each entity e_j is denoted by the set of its mentions in the paragraph, e.g., {`leaf, leaves`}
- A **set of properties** $P = \{p_k\}$ of entities to be tracked (e.g., location, existence)

predict:

- The **state** of each entity e_j after each sentence s_k, where an entity's state is the values of all its properties $\{p_k\}$. For example, in Figure 2, the state of the water after step 2 is {location(water) = turbine; exists(water) = true}.

This task definition covers the tasks used in earlier procedural text comprehension datasets. In bAbI tasks 1-3, a single propert (location) was tracked for a single entity throughout a paragraph (Weston et al., 2015). In the state tracking task of Bosselut et al. (2018), six properties (temperature, shape, etc.) were tracked for each ingredient in the recipe.

3.2 Data

In our work, we use the ProPara dataset (Dalvi et al., 2018) for both illustration and evalution. ProPara contains 488 paragraphs (3100 sentences) of a particular genre of procedural text, namely science processes (e.g., how hydroelectricity is generated). The dataset tracks two entity properties, existence and location, for all entities involved in each process, resulting in 81,000 annotations in the dataset. Figure 2 gives a (simplified) example of the data, visualized as an (entity x sentence) grid, where each column tracks a different entity (time progressing vertically downwards), and each row denotes the entities' state (existence and location) after each sentence. To evaluate the predictions, a set of templated questions whose answers can be computed from the predictions is posed (e.g., "What was destroyed, when and where?").

4 Model

We now describe our model, called PROSTRUCT.

4.1 Overview

We approach the task by predicting the state *changes* that occur at each step of the text, using a vocabulary (size K) of the possible state change types that can occur given the domain and properties being modeled. For example, for the ProPara dataset, we model $K = 4$ types of state change: ***`move`*****,** ***`create`*****,** ***`destroy`*****,** and ***`none`***. ***`move`*** changes an entity's location from one place to another, `create` from non-existence to a location, and *`destroy`* from a location to non-existence. State changes can be parameterized by text spans in the paragraph, e.g., ***`move`*** takes a before and after location parameter. If a parameterized state change is predicted, then the model also must predict its parameter values from the paragraph.

Previous models for process comprehension make a sequence of local predictions about the entities' states, one sentence at a time, maintaining a (typically neural) state at each sentence. However, none have the ability to reverse earlier predictions should an inconsistency arise later in the sequence. PROSTRUCT overcomes this limitation by reformulating the task as structured prediction. To do this, it uses a neural encoder-decoder from the semantic parsing literature (Krishnamurthy et al., 2017; Yin and Neubig, 2017) combined with a search procedure that integrates soft and hard constraints for finding the best candidate structure.

For each sentence and entity, the encoder first uses a bidirectional LSTM to encode the sentence and indicator variables identifying which entity is currently being considered (Figure 3). It then produces a (distributed) representation of the action that the sentence describes as being applied to that entity. During decoding, the model decodes each action embedding into a distribution over possible state changes that might result, then performs

---

[1] We use a broad definition of action to mean any event that changes the state of the world (including non-volitional events such as roots absorbing water).

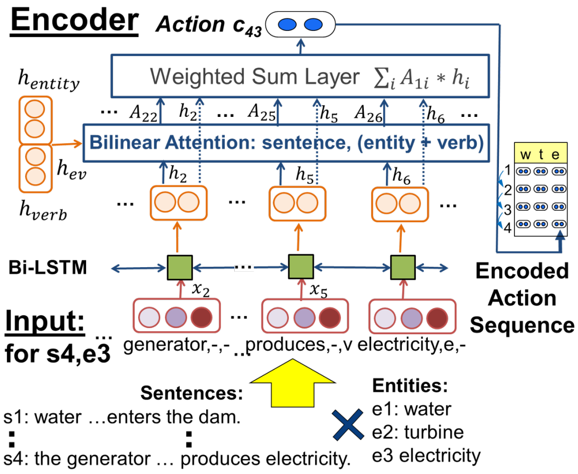

Figure 3: The encoder, illustrated for the ProPara domain with the paragraph from Figure 1. During encoding, ProStruct creates an action embedding c_{tj} representing the action at step t on entity e_k, for all entities at all steps. The overall action sequence (right-hand box) is the collection of these embeddings, for each entity (listed horizontally) and each step (listed vertically downwards).

a search over the space of possible state change sequences. Each node in the space is a partial sequence of state changes, and each edge is a prediction of the next state changes to add to the sequence (Figure 4).

During training, the model only follows the path along the gold sequence, and optimizes a loss function that drives up the likelihood of predictions along that path (thus driving down the probabilities for alternative, incorrect paths). At test time, the model does not have access to the gold path, and instead performs a beam search of the space to find the best candidate sequence.

Most importantly, by mapping the state change prediction problem to structured prediction, we can perform a search over the set of candidate paths that allows us to introduce hard and soft constraints that capture commonsense knowledge. Hard constraints are used to prune the search space (Equation 4 later), and soft constraints bias the search away from unlikely state changes via an additional term in the scoring function (Equations 5 and 6).

4.2 Encoder

The encoder operates over every $(s_t, e_j) \in S \times E$ pair to create an encoded representation c_{tj} of the action described by sentence s_t, as applied to entity e_j. In other words, we can consider the overall action to be represented by $|E|$ embeddings, one for each of the entities in E, encoding the action's effects on each. This novel feature allows us to model different effects on different entities by the same action. For example, a conversion action may simultaneously destroy one entity and create another. Figure 3 shows the encoder operating on s_4: "The generator spins, and produces electricity" and e_3: `electricity` from Figure 1.

Without loss of generality, we define an arbitrary sentence in S as $s_t = \{w_0, ..., w_I\}$. Each word w_i in the input sentence is encoded as a vector $x_i = [v_w : v_e : v_v]$, which is the concatenation of a pre-trained word embedding v_w for w_i, an indicator variable v_e for whether w_i is a reference to the specified entity e_j, and an indicator variable v_v for whether w_i is a verb. We use GloVe vectors as pre-trained embeddings (Pennington et al., 2014) and a POS tagger to extract verbs (Spacy, 2018).

Then, a BiLSTM is used to encode the word representations extracted above, yielding a contextualized vector h_i for each embedded word x_i that is the concatenated output of the backward and forward hidden states produced by the BiLSTM for word w_i. An attention over the contextualized embeddings h_i is performed to predict a distribution of weights over the sentence:

$$a_i = h_i * B * h_{ev} + b \tag{1}$$

$$c_{tj} = \sum_{i=1}^{I} a_i * h_i \tag{2}$$

where a_i is the attention weight for each contextualized embedding, c_{tj} is the vector encoding the action for the sentence-entity pair (s_t, e_j), B and b are learned parameters, and h_{ev} is the concatenation of the contextual embeddings of the hidden states where the entity h_e and verb h_v are mentioned:

$$h_{ev} = [\mu(\{h_i : x_i[v_e] = 1\}); \mu(\{h_i : x_i[v_v] = 1\}] \tag{3}$$

where μ is an average function, and $x_i[v_e]$ and $x_i[v_v]$ correspond to the entity indicator and verb indicator variables defined above for any word w_i, respectively. The output vector c_{tj} encodes the action at step s_t on entity e_j. This vector is computed for all steps and entities, populating a grid of the actions on each entity at each step (Figure 3).

4.3 Decoder

To decode the action vectors c_{tj} into their resulting state changes they imply, each is passed through a feedforward layer to generate $logit(\pi_t^j)$, a set of logistic activations over the K possible state changes π_t^j for entity e_j in sentence s_t. (For ProPara, there are $K = 4$ possible state changes: `move`, `create`, `destroy`, `none`). These logits denote how likely each state change π_t^j is for entity e_j at sentence s_t. The decoder then explores the search space of possible state change sequences for the **whole** paragraph (Figure 4), using these likelihoods to score each visited sequence (Equation 6).

Let π_t be the set of state changes for *all* entities at time t, i.e., $\pi_t = \{\pi_t^j\}_{j=1..|E|}$, and let Π_t be the sequence of state changes from time 1 to t, i.e., $\Pi_t = [\pi_1, ..., \pi_t]$. Each node in the search space is a Π_t, and each edge adds a π_{t+1} to it so that it becomes Π_{t+1}:

$$\Pi_t \xrightarrow{\pi_{t+1}} \Pi_{t+1}$$

Given there are K possible values for π_t^j, the number of possible configurations for π_t at time t (i.e., the branching factor during search) is exponential: $K^{|E|}$, where $|E|$ is the number of entities in the paragraph.

To explore this exponential number of paths, after every sentence s_t, we prune branches $\Pi_t \rightarrow \Pi_{t+1}$ where Π_{t+1} is impossible according to background knowledge (described in Section 5.1). We define the boolean function over state change sequences:

$$\begin{aligned} \text{allowable}(\Pi) &= 1 \text{ if hard constraints satisfied} \\ &= 0 \text{ otherwise} \end{aligned} \tag{4}$$

and prune paths Π_{t+1} where $allowable(\Pi_{t+1}) = 0$. For example for ProPara, a state transition such as *`DESTROY`* → *`MOVE`* is not allowed because a hard constraint prohibits non-existent entities from being moved (Section 5.1).

While hard constraints remove impossible state change predictions, there may also be other state changes that are implausible with respect to background knowledge. For example, commonsense dictates that it is unlikely (but not impossible) for plants to be destroyed during photosynthesis. Accordingly, our inference procedure should discourage (but not prohibit) predicting plant destruction when reading about photosynthesis. To discourage unlikely state changes, we make use of soft constraints that estimate the likelihood of a particular state change associated with an entity, denoted as:

$$P(\pi^j | e_j, topic) \tag{5}$$

In Section 5.2, we describe how these likelihoods can be estimated from large-scale corpora. We add this bias as an additional term (the second term below) when scoring the addition of π_{t+1} to the sequence so far Π_t:

$$\phi'(\pi_{t+1}) = \sum_{j=1}^{|E|} \Big(\lambda \; logit(\pi_{t+1}^j) + (1 - \lambda) \; \log P(\pi_{t+1}^j | e_j, topic) \Big) \tag{6}$$

where λ is a learned parameter controlling the degree of bias.

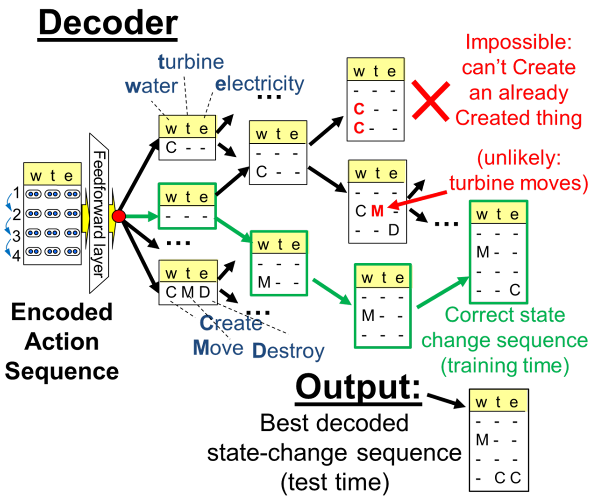

Figure 4: The decoder, illustrated for the ProPara domain. Each action embedding c_{tj} is first passed through a feedforward layer to generate a distribution over the (here $K = 4$) possible state changes that could result, for each entity (listed horizontally) at each step (listed vertically downwards). The decoder then explores the space of state-change sequences, using these distributions to guide the search. During end-to-end training, ProStruct follows the correct (green) path, and backpropagates to drive up probabilities along this path. During testing, the system performs a beam search to find the most globally plausible sequence.

During search, when making a transition along a path from Π_t to a valid Π_{t+1}, Π_{t+1} is scored by accumulating normalized scores along the path:

$$\phi(\Pi_{t+1}) = \phi(\Pi_t) + \frac{\phi'(\pi_{t+1})}{\sum_{\pi'_{t+1} \in \Pi_{t+1}} \phi'(\pi'_{t+1})} \tag{7}$$

Continuing state transitions in this manner, when we reach the finished state (i.e., last sentence), our objective is to maximize the score of the state changes produced when reading each sentence. During training, we only materialize a valid node when $\Pi_t \in \Pi_t^*$ where Π_t^* is the set of nodes along the gold path.

We use this constrained decoding to predict the state change sequence. For state changes that take additional parameters, e.g., in the ProPara model a *`move`* is parameterized by the before and after locations, we also predict those parameter values during decoding. This is done using standard span prediction layers (inspired by BiDAF, Seo et al. (2017a)) on top of the encoded input.

The model is trained to minimize the joint loss of predicting the correct state changes and correct state change parameters for every sentence in the paragraph:

$$\mathcal{L} = -\sum_{t=1}^{T}\Big(\log P(\pi_t) + \sum_{j=1}^{|E|}\sum_{p\in\text{param}(\pi_t^j)} \log P(v_{pjt}|\pi_t^j)\Big) \tag{8}$$

where $\text{param}(\pi_t^j)$ are the parameters of state change π_t^j, and v_{pjt} are the values of those parameters. For example, `move` is parameterized by before/after locations, and the 2nd loss term refers to the predicted values of those locations.

At test time, instead of following the gold state change path, we use beam search. After reading any sentence, we explore the top-k states sorted by the score $\phi'(\pi_t)$ that satisfy hard constraints. This way, we predict a sequence of state changes that have maximum score while being sensible w.r.t. hard constraints.

5 Incorporating Commonsense Knowledge

By formulating procedural text comprehension as a structured prediction task, we can introduce commonsense knowledge as hard and soft constraints into the model, allowing nonsensical and unlikely predictions to be avoided, and allowing the system to recover from early mistakes.

5.1 Hard Constraints

Hard constraints are introduced by defining the (boolean) function over a candidate sequence of state changes:

$$\text{allowable}(\Pi)$$

used in Equation 4.

While this function can be defined in any way, for the ProPara application we use six constraints. The first three below are based on basic "laws of physics" or commonsense (CS) and are universally applicable:

CS-1: An entity must **exist** before it can be **moved** or **destroyed**
CS-2: An entity cannot be **created** if it already **exists**
CS-3: An entity cannot change until it is mentioned in the paragraph

The next three constraints are observed in the training data:

D-1: Maximum number of toggles for an entity between Exists and not Exist $\leq f_{max\_toggles}$
D-2: Max fraction of entities that are changed per sentence $\leq f_{entities\_per\_sentence}$
D-3: Max fraction of sentences in which an entity changes $\leq f_{sentences\_per\_entity}$

The thresholds used in D-1, D-2 and D-3 are hyperparameters that can be tuned on the dev set.

5.2 Soft Constraints

Soft constraints are introduced by defining the prior probabilities used in Equation 6:

$$P(\pi^j|e_j, topic)$$

that entity e_j undergoes state change π^j in a sentence of text about *topic*. These probabilities are used to re-rank the candidate event sequences during decoding (see Equation 6).

While any method can be used to estimate these probabilities, we describe our corpus-based approach here. Although it was designed for ProPara, it generalizes easily to other domains, and is itself a contribution of this work. For a given state change π^j, entity e_j, and topic, we first gather a corpus of Web sentences mentioning that topic (using Bing search APIs), then count the number of times x that the entity is described as undergoing that state change (e.g., that `water` is said to `MOVE`). To determine this frequency, we first convert the sentences into a set of SRL frames (verb + role-argument pairs) using an off-the-shelf SRL labeler. We then use an existing rulebase, derived from VerbNet, that contains rules that map SRL frames to state changes, e.g., `e1/ARG0 "absorbs"/VERB e2/ARG1` $\Longrightarrow$ `e2 MOVES` (Clark et al., 2018). Although the rules and SRL labels are incomplete and noisy, redundancy in the corpus provides some robustness when estimating the frequency x. Finally, the observed frequency x is converted to a likelihood using a logistic transformation:

$$P(\pi^j|e_j, topic) = \frac{1}{1 + exp^{-(x-x_0)}} \tag{9}$$

where, x_0 is a hyperparameter tuned on the dev set.

5.3 Commonsense Constraints for New Domains

The commonsense constraints we have used for ProPara are general, covering the large variety of topics contain in ProPara (e.g., electricity, photosynthesis, earthquakes). However, if one wants to apply ProStruct to other genres of procedural text (e.g., fictional text, newswire articles), or broaden the state change vocabulary, different commonsense constraints may be needed. Note that our model architecture itself is agnostic to the source and quantity of hard and soft constraints. For example, one might leverage commonsense rules from existing ontologies such as SUMO (Niles and Pease, 2001) or Cyc (Lenat et al., 1985) to identify new hard constraints; and our corpus-based method could be extended to cover new state change types should the state change vocabulary be extended.

6 Evaluation

We evaluate our model using the ProPara dataset, and compare against several strong baselines published with the original dataset (Dalvi et al., 2018).

6.1 Evaluation setup

Given a paragraph and set of entities as input, the task is to answer four templated questions, whose answers are deterministically computed from the state change sequence:

Q1. What are the inputs to the process?
Q2. What are the outputs of the process?
Q3. What conversions occur, when and where?
Q4. What movements occur, when and where?

Inputs are defined as entities that existed at the start of the process, but not at the end. Outputs are entities that did not exist at the start, but did at the end. A conversion is when some entities are destroyed and others created. Finally, a movement is an event where an entity changes location.

For each process, as every question can have multiple answers, we compute a separate F1 score for each question by comparing the gold and predicted answers. For Q1 and Q2, this is straightforward as answers are atomic (i.e., individual names of entities). For Q3, as each answer is a 4-tuple (*convert-from, convert-to, location, sentence-id*), some answers may only be partially correct. To score partial correctness, we pair gold and predicted answers by requiring the *sentence-id* in each to be the same, and then score each pair by the Hamming distance of their tuples. For Q4, each answer is also a 4-tuple (*entity, from-location, to-location, sentence-id*), and the same procedure is applied. The four F1 scores are then macro-averaged. The total number of items to predict in the train/dev/test partitions is 7043/913/1095.

6.2 Baselines

We compare results using the following process comprehension models:

Recurrent Entity Networks (EntNet) (Henaff et al., 2017) are a state-of-the-art model for the bAbI tasks (Weston et al., 2015). The model uses a dynamic memory to maintain a representation of the world state as sentences are read, with a gated update at each step. These states are decoded to answer questions after each sentence is read.

Query Reduction Networks (QRN) (Seo et al., 2017b) perform a gated propagation of their hidden state across each time-step. Given a question, the hidden state is used to modify the query to keep pointing to the answer at each step.

ProLocal (Dalvi et al., 2018) predicts the state changes described in individual sentences, and then uses commonsense rules of inertia to propagate state values forwards and backwards in time.

ProGlobal (Dalvi et al., 2018) predicts states of an entity across all time steps. It considers the entire paragraph while predicting states for an entity, and learns to predict location spans at time-step $t + 1$ based on location span predictions at t.

7 Results

7.1 Comparison with Baselines

We compare our model (which make use of world knowledge) with the four baseline systems on the ProPara dataset. All models were trained on the training partition, and the best model picked based on prediction accuracy on the dev partition. Table 1 shows the precision, recall, and F1 for all models on the the test partition. PROSTRUCT significantly outperforms the baselines, suggesting that world knowledge helps PROSTRUCT avoid spurious predictions. This hypothesis is supported by the fact that the ProGlobal model has the highest recall and worst precision, indicating that it is over-generating state change predictions. Conversely, the ProLocal model has the highest precision, but its recall is much lower, likely because it makes predictions for individual sentences, and thus has no access to information in surrounding sentences that may suggest a state change is occurring.

We also examined the role of the constraint rules (both hard and soft) on efficiency. With all rules disabled, the training does not complete even one epoch in more than three hours. Because the number of valid states is exponential in the number of

	Precision	Recall	F1
ProLocal	77.4	22.9	35.3
QRN	55.5	31.3	40.0
EntNet	50.2	33.5	40.2
ProGlobal	46.7	52.4	49.4
ProStruct	74.2	42.1	**53.7**

Table 1: Results on the prediction task (test set).

	Precision	Recall	F1
ProStruct	70.4	47.8	56.9
- Soft constraints	61.9	47.4	53.7
- Hard constraints†	69.6	47.0	56.1

† Partial ablation, ablated at test only (training without these is computationally infeasible).

Table 2: Ablating world knowledge on the dev set.

entities, the training is particularly slow on paragraphs with many entities. In contrast, with all rules enabled, training takes less than 10 minutes per epoch. This illustrates that the constraints are not only contributing to the model scores, but also helping make the search efficient.

7.2 Ablations and Analysis

To explore the impact of world knowledge, we also performed two ablations on the dev set: Removing soft constraints (at both training and test time), and a partial ablation of removing hard constraints at test time only - note that hard constraints cannot be removed during training because model training time becomes prohibitively large without them, thus qualifying this second ablation. Table 4 shows that F1 drops when each type of knowledge is removed, illustrating that they are helping. The smaller drop for hard constraints suggests that they have primarily been incorporated into the network during training due to this ablation being partial.

Qualitatively, we compared dev set examples where the predicted event sequence changed, comparing predictions made without world knowledge to those made with world knowledge. For readability, we only show the event type predictions (M,C,D, and N (shown as "-")) and not their from-location/to-location arguments. If a prediction changes from X (without knowledge) to Y (with knowledge), we write this "$X \rightarrow Y$". For cases where the prediction changed, we show incorrect predictions in red, and correct predictions in green.

We first compare predictions made with and without the **BK** (corpus-based background knowledge, the soft constraints). Table 3 shows a paragraph about the process of nuclear-powered electricity generation, in the problematic prediction of the generator moving (M) was predicted in the second to last sentence. However, the background knowledge contains no examples of generators being moved. As a result, it drives the probability mass away from the move (M) prediction, resulting in a no state change (N) prediction instead.

Table 4 shows a second example where, without knowledge, no event was predicted for the spark entity. However, BK contains many examples of sparks being created (reflecting text about this topic), shifting the probability mass towards this prediction, resulting in the correct C (create).

Finally, Table 5 shows an example of a hard constraint preventing a nonsensical prediction (namely, electricity is created after it already exists).

7.3 Error Analysis

There are also many cases where incorrect predictions are made. The main causes are summarized below, and offer opportunities for future work.

Implicit reference is a challenge for ProStruct, where an entity affected by an event is not mentioned until a later sentence in the paragraph. For example, in the following ProPara paragraph snippet about combustion engines:

> "...(3) A spark ignites fuel...(4) The pressure pushes the piston down...."

both `spark` and `pressure` are created in sentence 3, even though `pressure` is not mentioned until the subsequent sentence. Recognizing this type of implicit mention is very hard. It is possible that BK could help in such situations, particularly if `ignite` were often associated with creating pressure in the context of a combustion engines, but we did not see such examples in practice.

A second challenge is **coreference**, in particular when different entities have similar names. For example, again for combustion, a snippet looks:

> ...(2) the fuel is injected... (6) the spent

	Without vs. with BK				
	Fuel	Heat	Turbine	Generator	Elec.
Fuel produces heat.	D	C	-	-	-
... Steam spins turbine.	-	-	-	-	-
Generator is turned.	-	-	-	M → N	-
Makes electricity.	-	-	-	-	C

Table 3: BK improves precision. In a nuclear powered electricity generation scenario, BK drives the probability mass away from the generator movement, as a generator does not generally change location.

	Without vs. with BK		
	Fuel	Air	Spark
Fuels burns in the chamber.	D	-	-
The burning fuel creates energy.	-	-	-
The upward motion cause air ...	-	M	-
The piston compresses the air.	-	-	-
A spark ignites the fuel and air ...	-	-	N → C
...	...	...	...

Table 4: BK improves coverage. BK has a strong signal that a spark is usually created in combustion engines, shifting the probability mass towards spark-creation.

	Without and with constraints		
	Electricity	Signals	...
Electricity enters supply unit.	M	-	-
The supply gives electricity to transistors.	C → D	-	
...	...	...	...
The energy is used to complete ...	-	-	

Table 5: Hard constraints avoid nonsensical predictions. In this example without ***CS*-2**, the electricity is predicted to be created after it already exists (impossible). This mistake is avoided using the constraints.

> fuel is ejected. (7) new fuel is injected....

Here `fuel` and `spent fuel` are the same entity, while `new fuel` is a different entity. Correctly tracking these references is challenging (in this case, ProStruct misidentifies (7) as describing an event on the original `fuel/spent fuel`).

A third, related problem is **pronoun resolution**. For example, in:

> The sound continues to bounce off of things and produce echoes until it is totally absorbed or dissipated.

the word `it` confuses ProStruct, and it predicts that the `echo` (rather than the `sound`) is destroyed. We observe several such failure cases.

Finally, we observed **BK retrieval failures** when there was appropriate background knowledge that was expressed in a lexically different way. Consider the example in Table 6 about oil formation. Without BK, the model correctly predicts that `sediment` is destroyed (D). However, BK has few examples of `sediment` being destroyed, and so biases the prediction away from this (correct) choice to an incorrect choice. Further examination of BK shows that it does in fact have knowledge about this destruction, but that is expressed using the word `deposit` instead (e.g., "deposits break down"). A soft (neural) means of accessing BK would help alleviate this problem.

	Without BK vs. with BK		
	Algae	Plankton	Sediment
Algae and plankton die.	D	D	-
The dead algae and plankton ...	-	-	-
The sediment breaks down.	-	-	D → M

Table 6: BK lookup limitation: though BK knows that `deposits` can be destroyed (broken down), it does not equate this with (synonymous) `sediments` being destroyed, hence biases model away from correct answer.

8 Conclusions

Answering questions about procedural text remains challenging, requiring models of actions and the state changes they produce. Predictions made locally throughout the text may together be globally inconsistent or improbable. We have shown how the predicted effects of actions can be improved by treating the task as a structured prediction problem, allowing commonsense knowledge to be injected to avoid an overall inconsistent or improbable set of predictions. In particular, we have shown how two kinds of knowledge can be exploited: hard constraints to exclude impossible and nonsensical state changes, and soft constraints to encourage likely state changes. The resulting system significantly outperforms previous state-of-the-art systems on a challenging dataset, and our ablations and analysis suggest that the knowledge is playing an important role. Our code is available at `https://github.com/allenai/propara`.

Acknowledgements

We thank Oren Etzioni for his insightful feedback and encouragement for this work. We are grateful to Paul Allen whose long-term vision continues to inspire our scientific endeavors.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Proc. EMNLP'14*.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. *6th International Conference on Learning Representations (ICLR)*.

Peter Clark, Bhavana Dalvi Mishra, and Niket Tandon. 2018. What happened? Leveraging VerbNet to predict the effects of actions in procedural text. *arXiv preprint arXiv:1804.05435*.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1595–1604, New Orleans, Louisiana. Association for Computational Linguistics.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *ICLR*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *ACL*.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *Proc. EMNLP'15*, pages 982–992.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proc. EMNLP'16*, pages 329–339.

Jayant Krishnamurthy, Pradeep Dasigi, and Matthew Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Douglas B Lenat, Mayank Prakash, and Mary Shepherd. 1985. Cyc: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI magazine*, 6(4):65.

Fei Liu and Julien Perez. 2017. Dialog state tracking, a machine reading approach using memory network. In *EACL*.

Ian Niles and Adam Pease. 2001. Towards a standard upper ontology. In *FOIS*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Hannah Rashkin, Antoine Bosselut, Maarten Sap, Kevin Knight, and Yejin Choi. 2018. Modeling naive psychology of characters in simple commonsense stories. In *ACL*.

Aju Thalappillil Scaria, Jonathan Berant, Mengqiu Wang, Peter Clark, Justin Lewis, Brittany Harding, and Christopher D. Manning. 2013. Learning biological processes with global constraints. In *EMNLP*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017a. Bidirectional attention flow for machine comprehension. In *Proc. ICLR'17*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017b. Query-reduction networks for question answering. In *ICLR*.

Spacy. 2018. Spacy tokenizer API reference page. `https://spacy.io/api/annotation#pos-tagging`. Accessed: 2018-04-10.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. In *ACL*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *NIPS*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei hao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *ACL*.

Collecting Diverse Natural Language Inference Problems for Sentence Representation Evaluation

Adam Poliak[1] **Aparajita Haldar**[1,2] **Rachel Rudinger**[1] **J. Edward Hu**[1]
Ellie Pavlick[3] **Aaron Steven White**[4] **Benjamin Van Durme**[1]
[1]Johns Hopkins University, [2] BITS Pilani, Goa Campus, India
[3]Brown University, [4]University of Rochester

Abstract

We present a large scale collection of diverse natural language inference (NLI) datasets that help provide insight into how well a sentence representation captures distinct types of reasoning. The collection results from recasting 13 existing datasets from 7 semantic phenomena into a common NLI structure, resulting in over half a million labeled context-hypothesis pairs in total. We refer to our collection as the DNC: ***D**iverse **N**atural Language Inference **C**ollection*. The DNC is available online at `http://www.decomp.net`, and will grow over time as additional resources are recast and added from novel sources.

1 Introduction

A plethora of new natural language inference (NLI)[1] datasets has been created in recent years (Bowman et al., 2015; Williams et al., 2017; Lai et al., 2017; Khot et al., 2018). However, these datasets do not provide clear insight into what type of reasoning or inference a model may be performing. For example, these datasets cannot be used to evaluate whether competitive NLI models can determine if an event occurred, correctly differentiate between figurative and literal language, or accurately identify and categorize named entities. Consequently, these datasets cannot answer how well sentence representation learning models capture distinct semantic phenomena necessary for general natural language understanding (NLU).

To answer these questions, we introduce the **D**iverse **N**LI **C**ollection (DNC), a large-scale NLI dataset that tests a model's ability to perform diverse types of reasoning. DNC is a collection of NLI problems, each requiring a model to perform a unique type of reasoning. Each NLI dataset contains labeled context-hypothesis pairs that we recast from semantic annotations for specific structured prediction tasks. We extend various prior works on challenge NLI datasets (Zhang et al., 2017), and define recasting as leveraging existing datasets to create NLI examples (Glickman, 2006; White et al., 2017). We recast annotations from a total of 13 datasets across 7 NLP tasks into labeled NLI examples. The tasks include event factuality, named entity recognition, gendered anaphora resolution, sentiment analysis, relationship extraction, pun detection, and lexicosyntactic inference. Currently, the DNC contains over half a million labeled examples. Table 1 includes NLI pairs that test specific types of reasoning.

Event Factuality	► Find him before he finds the dog food The finding did not happen	✓
	► I'll need to ponder The pondering happened	✗
Relation Extraction	► Ward joined Tom in their native Perth Ward was born in Perth	✓
	► Stefan had visited his son in Bulgaria Stefan was born in Bulgaria	✗
Puns	► Kim heard masks have no face value Kim heard a pun	✓
	► Tod heard that thrift is better than annuity Tod heard a pun	✗

Table 1: Example sentence pairs for different semantic phenomena. ► indicates the line is a context and the following line is its corresponding hypothesis. ✓ and ✗ respectively indicate that the context entails, or does not entail the hypothesis. Appendix A includes more recast examples.

Using a *hypothesis-only* NLI model, with access to just hypothesis sentences, as a strong baseline (Tsuchiya, 2018; Gururangan et al., 2018; Poliak et al., 2018b), our experiments demonstrate how DNC can be used to probe a model's ability to capture different types of semantic reasoning

---

[1] The task of determining if a hypothesis would likely be inferred from a context, or premise; also known as Recognizing Textual Entailment (RTE) (Dagan et al., 2006, 2013).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 67–81
Brussels, Belgium, October 31 - November 4, 2018.

necessary for general NLU. In short, this work answers a recent plea to the community to test "more kinds of inference" than in previous challenge sets (Chatzikyriakidis et al., 2017).

2 Motivation & Background

Compared to eliciting NLI datasets directly, i.e. asking humans to author contexts and/or hypothesis sentences, recasting can 1) help determine whether an NLU model performs distinct types of reasoning; 2) limit types of biases observed in previous NLI data; and 3) generate examples cheaply, potentially at large scales.

NLU Insights Popular NLI datasets, e.g. Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) and its successor Multi-NLI (Williams et al., 2017), were created by eliciting hypotheses from humans. Crowd-source workers were tasked with writing one sentence each that is entailed, neutral, and contradicted by a caption extracted from the Flickr30k corpus (Young et al., 2014). Although these datasets are widely used to train and evaluate sentence representations, a high accuracy is not indicative of what types of reasoning NLI models perform. Workers were free to create any type of hypothesis for each context and label. Such datasets cannot be used to determine how well an NLI model captures many desired capabilities of language understanding systems, e.g. paraphrastic inference, complex anaphora resolution (White et al., 2017), or compositionality (Pavlick and Callison-Burch, 2016; Dasgupta et al., 2018). By converting prior annotation of a specific phenomenon into NLI examples, recasting allows us to create a diverse NLI benchmark that tests a model's ability to perform distinct types of reasoning.

Limit Biases Studies indicate that many NLI datasets contain significant biases. Examples in the early Pascal RTE datasets could be correctly predicted based on syntax alone (Vanderwende and Dolan, 2006; Vanderwende et al., 2006). Statistical irregularities, and annotation artifacts, within class labels allow a hypothesis-only model to significantly outperform the majority baseline on at least six recent NLI datasets (Poliak et al., 2018b). Class label biases may be attributed to the human-elicited protocol. Moreover, examples in such NLI datasets may contain racial and gendered stereotypes (Rudinger et al., 2017).

We limit some biases by not relying on humans to generate hypotheses. Recast NLI datasets may still contain some biases, e.g. non-uniform distributions over NLI labels caused by the distribution of labels in the original dataset that we recast.[2] Experimental results using Poliak et al. (2018b)'s hypothesis-only model indicate to what degree the recast datasets retain some biases that may be present in the original semantic datasets.

NLI Examples at Large-scale Generating NLI datasets from scratch is costly. Humans must be paid to generate or label natural language text. This linearly scales costs as the amount of generated NLI-pairs increases. Existing annotations for a wide array of semantic NLP tasks are freely available. By leveraging existing semantic annotations already invested in by the community we can generate and label NLI pairs at little cost and create large NLI datasets to train data hungry models.

Why These Semantic Phenomena? A long term goal is to develop NLU systems that can achieve human levels of understanding and reasoning. Investigating how different architectures and training corpora can help a system perform human-level general NLU is an important step in this direction. DNC contains recast NLI pairs that are easily understandable by humans and can be used to evaluate different sentence encoders and NLU systems. These semantic phenomena cover distinct types of reasoning that an NLU system may often encounter in the wild. While higher performance on these benchmarks might not be conclusive proof of a system achieving human-level reasoning, a system that does poorly should not be viewed as performing human-level NLU. We argue that these semantic phenomena play integral roles in NLU. There exist more semantic phenomena integral to NLU (Allen, 1995) and we plan to include them in future versions of the DNC.

Previous Recast NLI Example sentences in RTE1 (Dagan et al., 2006) were extracted from MT, IE, and QA datasets, with the process referred to as 'recasting' in the thesis by Glickman (2006). NLU problems were reframed under the NLI framework and candidate sentence pairs were extracted from existing NLP datasets and then labeled under NLI (Dagan et al., 2006). Years later, this term was independently used by White et al.

---

[2] In a corpus with part-of-speech tags, the distribution of labels for the word "the" will likely peak at the `Det` tag.

(2017), who proposed to "leverage existing large-scale semantic annotation collections as a source of targeted textual inference examples." The term 'recasting' was limited to automatically converting existing semantic annotations into labeled NLI examples without manual intervention. We adopt the broader definition of 'recasting' since our NLI examples were automatically or manually generated from prior NLU datasets.

Applied Framework versus Inference Probing Traditionally, NLI has not been viewed as a downstream, applied NLP task.[3] Instead, the community has often used it as "a generic evaluation framework" to compare models for distinct downstream tasks (Dagan et al., 2006) or to determine whether a model performs distinct types of reasoning (Cooper et al., 1996). These two different evaluation goals may affect which datasets are recast. We target both goals as we recast applied tasks and linguistically focused phenomena.

3 Recasting Semantic Phenomena

We describe efforts to recast 7 semantic phenomena from a total of 13 datasets into labeled NLI examples. Many of the recasting methods rely on simple templates that do not include nuances and variances typical of natural language. This allows us to specifically test how sentence representations capture distinct types of reasoning. When recasting, we preserve each dataset's train/dev/test split. If a dataset does not contain such a split, we create a random split with roughly a 80:10:10 ratio. Table 2 reports statistics about each recast dataset.

Event Factuality (EF) Event factuality prediction is the task of determining whether an event described in text occurred. Determining whether an event occurred enables accurate inferences, e.g. monotonic inferences, based on the event (Rudinger et al., 2018b).[4] Incorporating factuality has been shown to improve NLI (Sauri and Pustejovsky, 2007).

We recast event factuality annotations from UW (Lee et al., 2015), MEANTIME (Minard et al., 2016), and Decomp (Rudinger et al., 2018b). We use sentences from original datasets as contexts and templates (1a) and (1b) as hypotheses.[5]

(1) a. The *Event* happened
  b. The *Event* did not happen

If the predicate denoting the *Event* was annotated as having happened in the factuality dataset, the context paired with (1a) is labeled as ENTAILED and the same context paired with (1b) is labeled as NOT-ENTAILED. Otherwise, we swap the labels.

Named Entity Recognition (NER) Distinct types of entities have different properties and relational objects (Prince, 1978) that can help infer facts from a given context. For example, if a system can detect that an entity is a name of a nation, then that entity likely has a leader, a language, and a culture (Prince, 1978; Van Durme, 2010). When classifying NLI pairs, a model can determine if an object mentioned in the hypothesis can be a relational object typically associated with the type of entity described in the context. NER tags can also be directly used to determine if a hypothesis is likely to not be entailed by a context, such as when entities in contexts and hypotheses do not share NER tags (Castillo and Alemany, 2008; Sammons et al., 2009; Pakray et al., 2010).

Given a sentence annotated with NER tags, we recast the annotations by preserving the original sentences as contexts and creating hypotheses using the template "*NP* is a *Label*."[6] For ENTAILED hypotheses we replace *Label* with the correct NER label of the *NP*; for NOT-ENTAILED hypotheses, we choose an incorrect label from the prior distribution of NER tags for the given phrase. This prevents us from adding additional biases besides any class-label statistical irregularities present in the original data. We apply this procedure on the Gronigen Meaning Bank (Bos et al., 2017) and the ConLL-2003 Shared Task (Tjong Kim Sang and De Meulder, 2003).

Gendered Anaphora Resolution (GAR) The ability to perform pronoun resolution is essential to language understanding, in many cases requiring common-sense reasoning about the world (Levesque et al., 2012). White et al. (2017) show that this task can be directly recast as an NLI problem by transforming Winograd schemas into NLI sentence pairs.

Using a similar formula Rudinger et al. (2018a) introduce *Winogender schemas*, minimal sentence pairs that differ only by pronoun gender. With this

---

[3] This changed as large NLI datasets have recently been used to train, or pre-train, models to perform NLI, or other tasks (Conneau et al., 2017; Pasunuru and Bansal, 2017).

[4] Appendix B.1 provides an example.

[5] We replace *Event* with the event described in the context.

[6] We ensure grammatical hypotheses by appropriately conjugating "is a" when needed.

Sem. Phenomena	Dataset	# pairs	Automated
Event Factuality	Decomp (Rudinger et al., 2018b)	42K (41,888)	✓
	UW (Lee et al., 2015)	5K (5,094)	✓
	MeanTime (Minard et al., 2016)	.7K (738)	✓
Named Entity Recognition	Groningen (Bos et al., 2017)	260K (261,406)	✓
	CoNLL (Tjong Kim Sang and De Meulder, 2003)	60K (59,970)	✓
Gendered Anaphora	Winogender (Rudinger et al., 2018a)	.4K (464)	✗
Lexicosyntactic Inference	VerbCorner (Hartshorne et al., 2013)	135K (138, 648)	✓
	MegaVeridicality (White and Rawlins, 2018)	11K (11,814)	✓
	VerbNet (Schuler, 2005)	2K (1, 759)	✓✗
Puns	(Yang et al., 2015)	9K (9,492)	✓
	SemEval 2017 Task 7 (Miller et al., 2017)	8K (8, 054)	✓
Relationship Extraction	FACC1 (Gabrilovich et al., 2013)	25K (25,132)	✓✗
Sentiment Analysis	(Kotzias et al., 2015)	6K (6,000)	✓
Combined	Diverse NLI Collection (DNC)	570K (570,459)	
—	SNLI (Bowman et al., 2015)	570K	
—	Multi-NLI (Williams et al., 2017)	433K	

Table 2: Statistics summarizing the recast datasets. The first column refers to the original annotation that was recast, the 'Combined' row refers to the combination of our recast datasets. The second column indicates the datasets that were recast, and the 3rd column reports how many labeled NLI pairs were extracted from the corresponding dataset. The last column indicates whether the recasting method was fully-automatic without human involvement (✓), manual (✗), or used a semi-automatic method that included human intervention (✓✗). The Multi-NLI and SNLI numbers contextualize the scale of our dataset.

adapted pronoun resolution task, they demonstrate the presence of systematic gender bias in coreference resolution systems. We recast Winogender schemas as an NLI task, introducing a potential method of detecting gender bias in NLI systems or sentence embeddings. In recasting, the context is the original, unmodified Winogender sentence; the hypothesis is a short, manually constructed sentence having a correct (ENTAILED) or incorrect (NOT-ENTAILED) pronoun resolution.

Lexicosyntactic Inference (Lex) While many inferences in natural language are triggered by lexical items alone, there exist pervasive inferences that arise from interactions between lexical items and their syntactic contexts. This is particularly apparent among propositional attitude verbs – e.g. *think*, *want*, *know* – which display complex distributional profiles (White and Rawlins, 2016). For instance, the verb *remember* can take both finite clausal complements and infinitival clausal complements.

(2) a. Jo didn't **remember** *that she ate*
    b. Jo didn't **remember** *to eat*

This small change in the syntactic structure gives rise to large changes in the inferences that are licensed: (2a) presupposes that *Jo ate* while (2b) entails that *Jo didn't eat*. We recast data from three datasets that are relevant to these sorts of lexicosyntactic interactions.

Lex #1: MegaVeridicality (MV) White and Rawlins (2018) build the MegaVeridicality dataset by selecting verbs from the MegaAttitude dataset (White and Rawlins, 2016) based on their grammatical acceptability in the [NP _ that S] and [NP was _ed that S] frames.[7] They then asked annotators to answer questions of the form in (3) using three possible responses: *yes*, *maybe or maybe not*, and *no* (Karttunen et al., 2014).

(3) a. Someone {knew, didn't know} that a particular thing happened.
    b. Did that thing happen?

We use the same procedure to annotate sentences containing verbs that take various types of infinitival complement: [NP _ for NP to VP], [NP _ to VP], [NP _ NP to VP], and [NP was _ed to VP].[8]

To recast these annotations, we assign the context sentences like (3a) to the majority class – *yes*, *maybe or maybe not*, *no* – across 10 different annotators, after applying an ordinal model-based normalization to their responses. We then pair each context sentence with three hypotheses.

(4) a. That thing happened
    b. That thing may or may not have happened
    c. That thing didn't happen

---

[7] NP is always instantiated by *someone*; and S is always instantiated by *a particular thing happened*.

[8] NP is always instantiated by either *someone*, *a particular person*, or *a particular thing*; and VP is always instantiated by *happen*, *do a particular thing*, or *have a particular thing*.

If annotated *yes*, *maybe or maybe not*, or *no*, the pair (3a)-(4a), (3a)-(4b), or (3a)-(4c) is respectively assigned ENTAILED and the other pairings are assigned NOT-ENTAILED; train/dev/test split labels are randomly assigned to every pair that context sentence appears in.

Lex #2: Recasting VerbNet (VN) We create additional lexicosyntactic NLI examples from VerbNet (Schuler, 2005). VerbNet contains classes of verbs that each can have multiple frames. Each frame contains a mapping from syntactic arguments to thematic roles, which are used as arguments in Neo-Davidsonian first-order logical predicates (5b) that describe the frame's semantics. Each frame additionally contains an example sentence (5a) that we use as our NLI context and we create templates (5c) from the most frequent semantic predicates to generate hypotheses (5d).

(5) a. Michael swatted the fly
  b. cause(E, $Agent$)
  c. *Agent* caused the E
  d. Michael caused the swatting

We use the Berkeley Parser (Petrov et al., 2006) to match tokens in an example sentence with the thematic roles and then fill in the templates with the matched tokens (5d). We also decompose multi-argument predicates into unary predicates to increase the number of hypotheses we generate. On average, each context is paired with 4.5 hypotheses. We generate NOT-ENTAILED hypotheses by filling in templates with incorrect thematic roles. [9] We partition the recast NLI examples into train/development/test splits such that all example sentences from a VerbNet class (which we use a NLI hypothesis) appear in only one partition of our dataset. In turn, the recast VerbNet dataset's partition is not exactly 80:10:10.

Lex #3: Recasting VerbCorner (VC) The third dataset testing lexicosyntactic inference that we recast is VerbCorner (VC) (Hartshorne et al., 2013). VC decomposes VerbNet predicates into simple semantic properties and "elicit[s] reliable semantic judgments corresponding to VerbNet predicates" via crowd-sourcing. The semantic judgments focus on movement, physical contact, application of force, change of physical or mental state, and valence, all of which "may be central organizing principles for a human's . . . conceptualization of the world." (Hartshorne et al., 2013).

Each sentence in VC is judged based on the decomposed semantic properties. We convert each semantic property into declarative statements[10] to create hypotheses and pair them with the original sentences which we preserve as contexts. The NLI pair is ENTAILED or NOT-ENTAILED depending on the given sentence's semantic judgment.

Figurative Language (Puns) Figurative language demonstrates natural language's expressiveness and wide variations. Understanding and recognizing figurative language "entail[s] cognitive capabilities to abstract and meta-represent meanings beyond *physical* words" (Reyes et al., 2012). Puns are prime examples of figurative language that may perplex general NLU systems as they are one of the more regular uses of linguistic ambiguity (Binsted, 1996) and rely on a wide-range of phonetic, morphological, syntactic, and semantic ambiguity (Pepicello and Green, 1984; Binsted, 1996; Bekinschtein et al., 2011).

We recast puns from Yang et al. (2015) and Miller et al. (2017) using templates to generate contexts (6a) and hypotheses (6b), (6c). We replace *Name* with names sampled from a distribution based on US census data,[11] and *Pun* with the original sentence. If the original sentence was labeled as containing a pun, the (6a)-(6b) pair is labeled as ENTAILED and (6a)-(6c) is labeled as NOT-ENTAILED, otherwise we swap the labels.

(6) a. *Name* heard that *Pun*
  b. *Name* heard a pun
  c. *Name* did not hear a pun

Relation Extraction (RE) The goal of the relation extraction (RE) task is to infer the real-world relationships between pairs of entities from natural language text. The task is "grounded" in the sense that the input is natural language text and the output is ⟨`entity1`, `relation`, `entity2`⟩ tuples defined in the schema of some knowledge base. RE requires a system to understand the many different surface forms which may entail the same underlying relation, and to distinguish those from surface forms which involve the same entities but do not entail the relation of interest. For example, (7a) is entailed by (7b) and (7c) but not by (7d).

---

[9] This is similar to Aharon et al. (2010)'s template matching to generate entailment rules from FrameNet (Baker et al., 1998).

[10] We list the declarative statements in Appendix B.2.1.

[11] `http://www.ssa.gov/oact/babynames/names.zip`

(7) a. *Name* was born in *Place*
b. *Name* is from *Place*
c. *Name*, a *Place* native, . . .
d. *Name* visited *Place*

Natural language surface forms are often used in RE in a weak-supervision setting (Mintz et al., 2009; Hoffmann et al., 2011; Riedel et al., 2013). That is, if `entity1` and `entity2` are known to be related by `relation`, it is assumed that every sentence observed which mentions both `entity1` and `entity2` is assumed to be a realization of `relation`: i.e. (7d) would (falsely) be taken as evidence of the `birthPlace` relation.

Here we first generate hypotheses and then corresponding contexts. To generate hypotheses, we begin with entity-relation triples extracted from DBPedia infoboxes: e.g. ⟨*Barack Obama*, `birthPlace`, *Hawaii*⟩. These relation predicates were extracted directly from Wikipedia infoboxes and are not cleaned. As a result, many relations are redundant with one another (`birthPlace`, `hometown`) and some relations do not correspond to obvious natural language glosses based on the name alone (`demographics1Info`). Thus, we construct a template for each predicate `p` by manually inspecting 1) a sample of entities which are related by `p` 2) a sample of sentences in which those entities co-occur and 3) the most frequent natural language strings which join entities related by `p` according to a OpenIE triple database (Schmitz et al., 2012; Fader et al., 2011) extracted from a large text corpus. We then manually write a simple template (e.g. Mention1 *was born in* Mention2) for `p`, ignoring any unclear relations. In total, we end up with 574 unique relations, expressed by 354 unique templates.

For each such hypothesis generated, we create a number of contexts. We begin with the FACC1 corpus (Gabrilovich et al., 2013) which contains natural language sentences from ClueWeb in which entities have been automatically linked to disambiguated Freebase entities, when possible. Then, given a tuple ⟨`entity1`, `relation`, `entity2`⟩, we find every sentence which contains both `entity1` and `entity2`. Since many of these sentences are false positives (7d), we have human annotators vet each context/hypothesis pair, using the ordinal entailment scale described in Zhang et al. (2017). We include optional binary labels by converting pairs labeled as $1 - 4$ and 5 to ENTAILED and NOT-ENTAILED respectively.[12] We apply pruning methods (described in Appendix B.4) to combat issues related to noisy, ungrammatical hypotheses and disagreement between multiple annotators.

Subjectivity (Sentiment) Some of the previously discussed semantic phenomena deal with objective information – did an event occur or what type of entities does a specific name represent. Subjective information is often expressed differently (Wiebe et al., 2005), making it important to use other tests to probe whether an NLU system understands language that expresses subjective information. We are interested in determining whether general NLU models capture 'subjective clues' that can help identify and understand emotions, opinions, and sentiment within a subjective text (Wilson et al., 2006).

We recast a sentiment analysis dataset since the task is the "expression of subjectivity as either a positive or negative opinion" (Taboada, 2016). We extract sentences from product, movie, and restaurant reviews labeled as containing positive or negative sentiment (Kotzias et al., 2015). Contexts (8a) and hypotheses (8b), (8c) are generated using the following templates:

(8) a. When asked about *Item*, *Name* said *Review*
b. *Name* liked the *Item*
c. *Name* did not like the *Item*

Item is replaced with either "product", "movie", or "restaurant", and the *Name* is sampled as previously discussed. If the original sentence contained positive (negative) sentiment, the (8a)-(8b) pair is labeled as ENTAILED (NOT-ENTAILED) and (8a)-(8c) is labeled as NOT-ENTAILED (ENTAILED).

3.1 Noise in Recast Data

Recasting can create noisy NLI examples that may potentially enable a model to achieve a high accuracy by learning dataset specific characteristics that are unrelated to NLU. For example, Poliak et al. (2018a,b) previously noted the association between ungrammaticality and NOT-ENTAILED examples based on how White et al. (2017) recast the FrameNet+ dataset (Pavlick et al., 2015).

---

[12] Following the label set in SNLI, Zhang et al. (2017) converted pairs labeled with 1 as CONTRADICTION, $2 - 4$ as NEUTRAL and 5 to ENTAILMENT. Since here we are generally interested in binary classification, we merge the CONTRADICTION and NEUTRAL examples as NOT-ENTAILED.

Recast Data Model	NER	EF	RE	Puns	Sentiment	GAR	VC	MV	VN
Majority (MAJ)	50.00	50.00	59.53	50.00	50.00	50.00	50.00	66.67	53.66
No Pre-training									
InferSent	**92.50**	83.07	61.89	60.36	50.00	–	88.60	**85.96**	46.34
Hyp-only	91.48	69.14	64.78	60.36	50.00	–	76.82	77.83	46.34
Pre-trained DNC									
InferSent *(update)*	92.47	**83.86**	74.38	**93.17**	81.00	–	**89.00**	85.62	76.83
InferSent *(fixed)*	92.20	81.07	74.11	87.76	77.33	**50.65**	88.59	83.84	67.68
Hyp-only *(update)*	91.60	71.07	70.57	60.02	46.83	–	76.78	77.83	68.90
Hyp-only *(fixed)*	91.37	69.74	65.97	56.44	48.17	50.00	76.78	77.83	59.15
Pre-trained Multi-NLI									
InferSent *(update)*	92.37	83.03	**76.08**	92.48	**83.50**	–	88.45	85.11	**78.05**
InferSent *(fixed)*	52.99	54.88	66.75	56.04	56.50	**50.65**	45.33	55.92	45.73
Hyp-only *(update)*	91.62	70.64	69.91	60.36	49.33	–	76.82	77.83	68.29
Hyp-only *(fixed)*	52.55	66.33	52.96	60.59	50.00	50.43	41.31	46.28	48.78

Table 3: NLI accuracies on test data. Columns correspond to each semantic phenomena and rows correspond to the model used. Columns are ordered from larger to smaller in size, but the last three (VC, MV, VN) are separated since they fall under lexicosyntactic inference. *(update)* refers to a model that was initialized with pre-trained parameters and then re-trained on the corresponding recast data. *(fixed)* refers to a model that was trained and then evaluated on these data sets. Bold numbers in each column indicate which settings were responsible for the highest accuracy on the specific recast dataset.

In the DNC, most of the noisy examples are in the recast VerbNet and Relation Extraction portions. In recast VerbNet, some examples are noisy because of incorrect subject-verb agreement.[13] Since more noisy examples appeared in the Relation Extraction set, we relied on Amazon Mechanical Turk workers to flag ungrammatical hypotheses in the recast dataset, and we remove NLI pairs with ungrammatical hypotheses.[14]

4 Experiments

Our experiments demonstrate how these recast datasets may be used to evaluate how well models capture different types of semantic reasoning necessary for general language understanding. We also include results from a hypothesis-only model as a strong baseline. This may reveal whether the recast datasets retain statistical irregularities from the original, task-specific annotations.

4.1 Models

For demonstrating how well an NLI model performs these fine-grained types of reasoning, we use `InferSent` (Conneau et al., 2017). `InferSent` independently encodes a context and hypothesis with a bi-directional LSTM and combines the sentence representations by concatenating the individual sentence representations, their element-wise subtraction and product. The combined representation is then fed into a MLP with a single hidden layer. The hypothesis-only model is a modified version of `InferSent` that only accesses hypotheses (Poliak et al., 2018b). We report experimental details in Appendix C.

4.2 Results

Table 3 reports the models' accuracies across the recast NLI datasets. Even though we categorize VerbNet, MegaVeridicality, and VerbCorner as lexicosyntatic inference, we train and evaluate models separately on these three datasets because we use different strategies to individually recast them. When evaluating NLI models, our baseline is the maximum between the accuracies of the hypothesis-only model and the majority class label (MAJ). In six of the eight recast datasets that we use to train our models the hypothesis-only model outperforms MAJ. The two datasets where the hypothesis-only model does not outperform MAJ are Sentiment and VN, each of which contain less than $10K$ examples.[15] We do not train on GAR because of its small size.

Our results suggest that `InferSent`, when not pre-trained on any other data, might capture specific semantic phenomena better than other seman-

---

[13] "Her *teeth was* cared for" or "*Floss were* used".

[14] See Appendix B.4 for details.

[15] This is similar to Poliak et al. (2018b)'s results where a hypothesis-only model did not outperform MAJ on datasets with $\leq$ 10K examples.

tic phenomena. `InferSent` seems to learn the most about determining if an event occurred, since the difference between its accuracy and that of the hypothesis-only baseline (+13.93) is largest on the recast EF dataset compared to the other recast annotations. The model seems to similarly learn to perform (or detect) the type of lexicosyntactic inference present in VC and MV. Interestingly, the hypothesis-only model outperforms `InferSent` on the recast RE.

Hypothesis Only Baseline The hypothesis-only model can demonstrate how likely it is that an NLI label applies to a hypothesis, regardless of its context and indicates how well each recast dataset tests a model's ability to perform each specific type of reasoning when performing NLI. The high hypothesis-only accuracy on the recast NER dataset may demonstrate that the hypothesis-only model is able to detect that the distribution of class labels for a given word may be peaky. For example, *Hong Kong* appears 130 times in the training set and is always labeled as a location. Based on this, in future work we may consider different methods to recast NER annotations into labeled NLI examples, or limit the dataset's training size.

Pre-training models on DNC We would like to know whether initializing models with pre-trained parameters improves scores. We notice that when we pre-train our models on DNC, for the larger datasets, a pre-trained model does not seem to significantly outperform randomly initializing the parameters. For the smaller datasets, specifically Puns, Sentiment and VN, a pre-trained model significantly outperforms random initialization.[16]

We are also interested to know whether fine-tuning these pre-trained models on each category (*update*) improves a model's ability to perform well on the category compared to keeping the pre-trained models' parameters static (*fixed*). Across all of the recast datasets, updating the pre-trained model's parameters during training improves `InferSent`'s accuracies more than keeping the model's parameters fixed. When updating a model pre-trained on the entire DNC, we see the largest improvements on VN (+9.15).

Models trained on Multi-NLI Williams et al. (2017) argue that Multi-NLI "[makes] it possible to evaluate systems on nearly the full complexity of the language." However, how well does Multi-NLI test a model's capability to understand the diverse semantic phenomena captured in DNC? We posit that if a model, trained on and performing well on Multi-NLI, does not perform well on our recast datasets, then Multi-NLI might not evaluate a model's ability to understand the "full complexity" of language as argued.[17]

When trained on Multi-NLI, our `InferSent` model achieves an accuracy of 70.22% on (matched) Multi-NLI.[18] When we test the model on the recast datasets (without updating the parameters), we see significant drops.[19] On the datasets testing a model's lexicosyntactic inference capabilities, the model performs below the majority class baseline. On the NER, EF, and Puns datasets its performs below the hypothesis-only baseline. We also notice that on three of the datasets (EF, Puns, and VN), the fixed hypothesis-only model outperforms the fixed `InferSent` model.

These results might suggest that Multi-NLI does not evaluate whether sentence representations capture these distinct semantic phenomena. This is a bit surprising for some of the recast phenomena. We would expect Multi-NLI's fiction section (especially its humor subset) in the training set to contain some figurative language that might be similar to puns, and the travel guides (and possibly telephone conversations) to contain text related to sentiment.

Pre-training on DNC or Multi-NLI? Initializing a model with parameters pre-trained on DNC or Multi-NLI often outperforms random initialization.[20] Is it better to pre-train on DNC or Multi-NLI? On five of the recast datasets, using a model pre-trained on DNC outperforms a model pre-trained on Multi-NLI. The results are flipped on the two datasets focused on downstream tasks (Sentiment and RE) and MV. However, the differences between pre-training on the DNC or Multi-NLI are small. From this, it is unclear whether pre-training on DNC is better than Multi-NLI.

Size of Pre-trained DNC Data We randomly sample 10K and 20K examples from each

---

[16] By 32.81, 31.00, and 30.83 points respectively.

[17] We treat Multi-NLI's NEUTRAL and CONTRADICTION labels as equivalent to the DNC's NOT-ENTAILED label.

[18] Although this is about 10 points below SoTA, we believe that the pre-trained model performs well enough to evaluate whether Multi-NLI tests a model's capability to understand the diverse semantic phenomena in the DNC.

[19] `InferSent` (*pre-trained, fixed*) in Table 3.

[20] Pre-training does not improve accuracies on NER or MV.

datasets' training set to investigate what happens if we train our models on a subsample of each training set instead of the entire DNC. Although we noticed a slight decrease across each recast test set, the decrease was not significant. We leave this investigating for a future thorough study.

5 Related Work

Exploring what linguistic phenomena neural models learn Many tests have been used to probe how well neural models learn different linguistic phenomena. Linzen et al. (2016) use "number agreement in English subject-verb dependencies" to show that LSTMs learn about syntax-sensitive dependencies. In addition to syntax (Shi et al., 2016), researchers have used other labeling tasks to investigate whether neural machine translation (NMT) models learn different linguistic phenomena (Belinkov et al., 2017a,b; Dalvi et al., 2017; Marvin and Koehn, 2018). Recently, Poliak et al. (2018a) used recast NLI datasets to investigate semantics captured by NMT encoders.

Targeted Tests for Natural Language Understanding We follow a long line of work focused on building datasets to test how well NLU systems perform distinct types of semantic reasoning. FraCaS uses a limited number of sentence-pairs to test whether systems understand semantic phenomena, e.g. generalized quantifiers, temporal references, and (nominal) anaphora (Cooper et al., 1996). FraCas cannot be used to train neural models – it includes just roughly 300 high-quality instances manually created by linguists. MacCartney (2009) created the *FraCaS textual inference test suite* by automatically "convert[ing] each FraCaS question into a declarative hypothesis." Levesque et al. (2012)'s Winograd Schema Challenge forces a model to choose between two possible answers for a question based on a sentence describing an event.

Recent benchmarks test whether NLI models handle adjective-noun composition (Pavlick and Callison-Burch, 2016), other types of composition (Dasgupta et al., 2018), paraphrastic inference, anaphora resolution, and semantic proto-roles (White et al., 2017). Concurrently, Conneau et al. (2018)'s benchmark can be used to probe whether sentence representations capture many linguistic properties. It includes syntactic and surface form tests but does not focus on as a wide range of semantic phenomena as in the DNC. Glockner et al. (2018) introduce a modified version of SNLI to test how well NLI models perform when requiring lexical and world knowledge.

Wang et al. (2018)'s GLUE dataset is intended to evaluate and potentially train a sentence representation to perform well across different NLP tasks. This continues an aspect of the initial RTE collection, designed to be representative of downstream tasks like QA, MT, and IR (Dagan et al., 2010). While GLUE is therefore concerned with applied tasks, DNC, as well as Naik et al. (2018)'s NLI *stress* tests, is concerned with *probing* the capabilities of NLU models to capture explicitly distinguished aspects of meaning. While one may conjecture that the latter is needed to be "solved" to eventually "solve" the former, it may be that these goals only partially overlap. Some NLP researchers might focus on probing for semantic phenomena in sentence representations while others may be more interested in developing single sentence representations that can help models perform well on a wide array of downstream tasks.

6 Conclusion

We described how we recast a wide range of semantic phenomena from many NLP datasets into labeled NLI sentence pairs. These examples serve as a diverse NLI framework that may help diagnose whether NLU models capture and perform distinct types of reasoning. Our experiments demonstrate how to use this framework as an NLU benchmark. The DNC is actively growing as we continue recasting more datasets into labeled NLI examples. We encourage dataset creators to recast their datasets in NLI and invite them to add their recast datasets into the DNC. The collection, along with baselines and trained models are available online at `http://www.decomp.net`.

Acknowledgements

We thank Diyi Yang for help with the PunsOfTheDay dataset, the JSALT "Sentence Representation" team for insightful discussions, and three anonymous reviewers for feedback. This work was supported by the JHU HLTCOE, DARPA LORELEI and AIDA, NSF-BCS (1748969/1749025), and NSF-GRFP (1232825). The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

References

Roni Ben Aharon, Idan Szpektor, and Ido Dagan. 2010. Generating entailment rules from framenet. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 241–246. Association for Computational Linguistics.

James Allen. 1995. *Natural language understanding*. Pearson.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Tristan A Bekinschtein, Matthew H Davis, Jennifer M Rodd, and Adrian M Owen. 2011. Why clowns taste funny: the relationship between humor and semantic ambiguity. *Journal of Neuroscience*, 31(26):9665–9671.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Kim Binsted. 1996. *Machine humour: An implemented model of puns*. Ph.D. thesis, University of Edinburgh, Edinburgh, Scotland.

Johan Bos, Valerio Basile, Kilian Evang, Noortje J Venhuizen, and Johannes Bjerva. 2017. The groningen meaning bank. In *Handbook of Linguistic Annotation*, pages 463–496. Springer.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Julio Javier Castillo and Laura Alonso Alemany. 2008. An approach using named entities for recognizing textual entailment. In *Notebook Papers of the Text Analysis Conference, TAC Workshop*.

Stergios Chatzikyriakidis, Robin Cooper, Simon Dobnik, and Staffan Larsson. 2017. An overview of natural language inference data collection: The way forward? In *Proceedings of the Computing Natural Language Inference Workshop*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Alexis Conneau, GermÃČÂąn Kruszewski, Guillaume Lample, LoÃČÂŕc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. Recognizing textual entailment: Rational, evaluation and approaches–erratum. *Natural Language Engineering*, 16(1):105–105.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 142–151, Taipei, Taiwan. Asian Federation of Natural Language Processing.

I. Dasgupta, D. Guo, A. Stuhlmüller, S. J. Gershman, and N. D. Goodman. 2018. Evaluating Compositionality in Sentence Embeddings. *ArXiv e-prints*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1535–1545. Association for Computational Linguistics.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). *Note: http://lemurproject.org/clueweb09/FACC1/Cited by*, 5.

Oren Glickman. 2006. *Applied textual entailment*. Ph.D. thesis, Bar Ilan University.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Joshua K Hartshorne, Claire Bonial, and Martha Palmer. 2013. The verbcorner project: Toward an empirically-based semantic decomposition of verbs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1438–1442.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA. Association for Computational Linguistics.

Lauri Karttunen, Stanley Peters, Annie Zaenen, and Cleo Condoravdi. 2014. The Chameleon-like Nature of Evaluative Adjectives. In *Empirical Issues in Syntax and Semantics 10*, pages 233–250. CSSP-CNRS.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. In *AAAI*.

Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth. 2015. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606. ACM.

Alice Lai, Yonatan Bisk, and Julia Hockenmaier. 2017. Natural language inference from multiple premises. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 100–109, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Kenton Lee, Yoav Artzi, Yejin Choi, and Luke Zettlemoyer. 2015. Event detection and factuality assessment with non-expert supervision. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1648.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 552–561. AAAI Press.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Bill MacCartney. 2009. *Natural language inference*. Ph.D. thesis, Stanford University.

Rebecca Marvin and Philipp Koehn. 2018. Exploring Word Sense Disambiguation Abilities of Neural Machine Translation Systems. In *Proceedings of the 13th Conference of The Association for Machine Translation in the Americas (Volume 1: Research Track*, pages 125–131.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of english puns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 719–729.

Tristan Miller, Christian Hempelmann, and Iryna Gurevych. 2017. Semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 58–68, Vancouver, Canada. Association for Computational Linguistics.

Tristan Miller and Mladen Turković. 2016. Towards the automatic detection and identification of english puns. *The European Journal of Humour Research*, 4(1):59–75.

Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begona Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son. 2016. Meantime, the newsreader multilingual event and time corpus. In *Language Resources and Evaluation Conference (LREC)*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Partha Pakray, Santanu Pal, Soujanya Poria, Sivaji Bandyopadhyay, and Alexander F Gelbukh. 2010. Ju_cse_tac: Textual entailment recognition system at tac rte-6. In *TAC Workshop*.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multitask video captioning with video and entailment generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1273–1283.

Ellie Pavlick and Chris Callison-Burch. 2016. Most "babies" are "little" and most "problems" are "huge": Compositional entailment in adjective-nouns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2164–2173. Association for Computational Linguistics.

Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015. Framenet+: Fast paraphrastic tripling of framenet. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 408–413, Beijing, China. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

William J Pepicello and Thomas A Green. 1984. *Language of riddles: new perspectives*. The Ohio State University Press.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Adam Poliak, Yonatan Belinkov, James Glass, and Benjamin Van Durme. 2018a. On the evaluation of semantic phenomena in neural machine translation using natural language inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 513–523, New Orleans, Louisiana. Association for Computational Linguistics.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018b. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Ellen F Prince. 1978. On the function of existential presupposition in discourse. In *Papers from the... Regional Meeting. Chicago Ling. Soc. Chicago, Ill.*, volume 14, pages 362–376.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 74–79, Valencia, Spain. Association for Computational Linguistics.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018a. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14, New Orleans, Louisiana. Association for Computational Linguistics.

Rachel Rudinger, Aaron Steven White, and Benjamin Van Durme. 2018b. Neural models of factuality. In *Proceedings of the 2018 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 731–744, New Orleans, Louisiana. Association for Computational Linguistics.

Mark Sammons, VG Vinod Vydiswaran, Tim Vieira, Nikhil Johri, Ming-Wei Chang, Dan Goldwasser, Vivek Srikumar, Gourab Kundu, Yuancheng Tu, Kevin Small, et al. 2009. Relation alignment for textual entailment recognition. In *TAC Workshop*.

Roser Sauri and James Pustejovsky. 2007. Determining modality and factuality for text entailment. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 509–516. IEEE.

Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Karin Kipper Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Maite Taboada. 2016. Sentiment analysis: an overview from linguistics. *Annual Review of Linguistics*, 2:325–347.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *11th International Conference on Language Resources and Evaluation (LREC2018)*.

Benjamin Van Durme. 2010. *Extracting Implicit Knowledge from Text*. Ph.D. thesis, University of Rochester, Rochester, NY 14627.

Lucy Vanderwende and William B Dolan. 2006. What syntax can contribute in the entailment task. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 205–216. Springer.

Lucy Vanderwende, Arul Menezes, and Rion Snow. 2006. Microsoft research at rte-2: Syntactic contributions in the entailment task: an implementation. In *Second PASCAL Challenges Workshop*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Aaron Steven White, Pushpendre Rastogi, Kevin Duh, and Benjamin Van Durme. 2017. Inference is everything: Recasting semantic resources into a unified evaluation framework. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 996–1005, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Aaron Steven White and Kyle Rawlins. 2016. A computational model of s-selection. In *Semantics and linguistic theory*, volume 26, pages 641–663.

Aaron Steven White and Kyle Rawlins. 2018. The role of veridicality and factivity in clause selection. In *Proceedings of the 48th Annual Meeting of the North East Linguistic Society*, page to appear, Amherst, MA. GLSA Publications.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. 2006. Recognizing strong and weak opinion clauses. *Computational intelligence*, 22(2):73–99.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376. Association for Computational Linguistics.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *Transactions of the Association of Computational Linguistics*, 5(1):379–395.

A More Recast NLI Examples

Table 4 includes examples from all of the recast NLI datasets. We include one ENTAILED and one NOT-ENTAILED example from each dataset that tests a distinct type of reasoning.

B Recasting Semantic Phenomena

Here we add secondary information about the original datasets and our recasting efforts.

B.1 Event Factuality

We demonstrate how determining whether an event occurred can enable accurate inferences based on the event. Consider the following sentences:

(9) a. She walked a beagle
b. She walked a dog
c. She walked a brown beagle

If the *walking* occurred, (9a) entails (9b) but not (9c). If we negate the action in sentences (9a), (9b), and (9c) to respectively become:

(10) a. She did not walk a beagle
b. She did not walk a dog
c. She did not walk a brown beagle

The new hypothesis (10c) is now entailed by the context (10a) while (10b) is not.

B.2 Lexicosyntactic Inference

B.2.1 VerbCorner

When recasting VerbCorner, we use the following templates for hypotheses, assigning them as ENTAILED and NOT-ENTAILED based on the positive or negative answers to the annotation task questions about the context sentence.

(11) a. Someone {moved/did not move} from their location
b. Something touched another thing / Nothing touched anything else
c. Someone or something {applied/did not apply} force onto something
d. Someone or something {changed/did not change} physically
e. Someone {changed/did not change} their thoughts, feelings, or beliefs
f. Something {good/neutral/bad} happened

B.3 Figurative Language

Puns in Yang et al. (2015) were originally extracted from `punsoftheday.com`, and sentences without puns came from newswire and proverbs. The sentences are labeled as containing a pun or not. Puns in Miller et al. (2017) were sampled from prior pun detection datasets (Miller and Gurevych, 2015; Miller and Turković, 2016) and includes new examples generated from scratch for the shared task; the original labels denote whether the sentences contain homographic, heterographic, or no pun at all. Here, we are only interested in whether a sentence contains a pun or not instead of discriminating between homographic and heterographic puns.

B.4 Relation Extraction

Since hypotheses were automatically generated from Wikipedia infoboxes, many examples are noisy and ungrammatical. We presented hypotheses (independent of their corresponding contexts) to Mechanical Turk workers and asked them to label each sentence as containing no grammatical error, minor grammatical issues, or major grammatical issues. We removed the $2,056$ NLI examples with hypothesis containing major grammatical issues, resulting in $28,041$ labeled pairs. Interestingly, almost 70% of those examples where labeled between $1-4$, which we view as NOT-ENTAILED. We release the ungrammatical NLI examples as supplementary data.

A second source of noise in the recast relation extraction dataset can be caused by disagreement amongst multiple annotators. Examples in our training and development sets are annotated by a single annotator while we use 3- to 5-way redundancy to annotate the test examples. To guarantee high-quality test examples, we only include examples with 100% inner-annotator agreement. Additionally, we remove the 16 examples labeled with 4 from our NOT-ENTAILED examples in this pruned test set since some of these examples are arguably entailments. Consequently, the test set contains 761 examples, out of the original $3,670$ test examples. Nevertheless, we separately release all $3,670$ test examples and include the original annotations as well, enabling others to consider other methods to collapse the multi-way annotations.

Semantic Phenomena	✓	✗
Event Factuality	I would like to learn how The learning did not happen	I'll not say anything The saying happened
Named Entity Recognition	Ms. Rice said the United States must work intensively Ms. is a person 's title	Afghan officials are welcoming the Netherlands' decision The Netherlands is an event
Gendered Anaphora	The student met with the architect to view her blueprints for inspiration The architect has blueprints	The appraiser told the buyer that he had paid too much for the painting The appraiser had purchased a painting
MegaVeridicality	Someone assumed that a particular thing happened That thing might or might not have happened	A particular person craved to do a particular thing That person did that thing
VerbNet	The Romans destroyed the city The Romans caused the destroying	Andre presented the plaque Andre was transferred
VerbCorner	Molly wheeled Lisa to Rachel Someone moved from their location	Kyle bewildered Mark Someone or something changed physically
Relation Extraction	At least 100,000 Chinese live in Lhasa, outnumbering Tibetans two to one Tibetans live in Lhasa	Tropical storm Humberto is expected to reach the Texas coast tonight Humberto hit Texas
Puns	Jorden heard that my skiing skills are really going downhill Jorden heared a pun	Caiden heard that fretting cares make grey hairs Caiden heared a pun
Sentiment Analysis	When asked about the product, Liam said, "Don't waste your money" Liam did not like the product	When asked about the movie, Angel said, "A bit predictable" Angel liked the movie

Table 4: Example sentence pairs for different semantic phenomena. The ✓ and ✗ columns respectively indicate that the context entails, or does not entail the hypothesis. Each cell's first and second line respectively represent a context and hypothesis.

B.5 Sentiment

Kotzias et al. (2015) compiled examples from previous sources. The movie dataset came from Maas et al. (2011), the Amazon product reviews were released by McAuley and Leskovec (2013) add the restaurant reviews were sourced from the Yelp dataset challenge.[21]

C Experimental Details

In all our experiments, we use pre-computed GloVe embeddings (Pennington et al., 2014) and use the OOV vector for words that do not have a defined embedding. We follow Conneau et al. (2017)'s procedure to train our models. During training, our models are optimized with SGD. Our initial learning rate is 0.1 with a decay rate of 0.99. Our models train for at most 20 epochs and can optionally terminate early when the learning rate is less than 10^{-5}. If the accuracy deceases on the development set in any epoch, the learning rate is divided by 5. As described in Poliak et al. (2018b), our hypothesis-only model feeds the hypotheses' encoded representation directly into the MLP.

---

[21] `http://www.yelp.com/dataset_challenge`

Textual Analogy Parsing: What's Shared and What's Compared among Analogous Facts

Matthew Lamm[1,3] **Arun Tejasvi Chaganty**[2,3*]
Christopher D. Manning[1,2,3] **Dan Jurafsky**[1,2,3] **Percy Liang**[2,3]
[1]Stanford Linguistics [2]Stanford Computer Science [3]Stanford NLP Group
{mlamm, jurafsky}@stanford.edu
{chaganty, manning, pliang}@cs.stanford.edu

Abstract

To understand a sentence like "whereas only 10% of White Americans live at or below the poverty line, 28% of African Americans do" it is important not only to identify individual facts, e.g., poverty rates of distinct demographic groups, but also the higher-order relations between them, e.g., the disparity between them. In this paper, we propose the task of Textual Analogy Parsing (TAP) to model this higher-order meaning. The output of TAP is a frame-style meaning representation which explicitly specifies what is shared (e.g., poverty rates) and what is compared (e.g., White Americans vs. African Americans, 10% vs. 28%) between its component facts. Such a meaning representation can enable new applications that rely on discourse understanding such as automated chart generation from quantitative text. We present a new dataset for TAP, baselines, and a model that successfully uses an ILP to enforce the structural constraints of the problem.

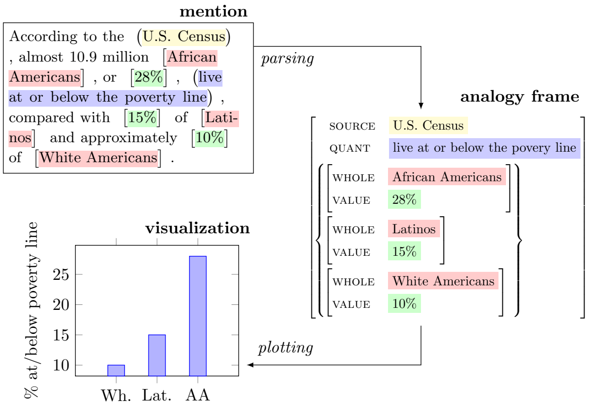

According to the (U.S. Census), almost 10.9 million [African Americans], or [28%], (live at or below the poverty line), compared with [15%] of [Latinos] and approximately [10%] of [White Americans].

Figure 1: In textual analogy parsing (TAP), one maps analogous facts to semantic role representations and identifies analogical relations between them. Automated chart generation from text is a motivating application of TAP.

1 Introduction

The task of information extraction by and large seeks to populate a knowledge base with individuated facts extracted from text (Sarawagi, 2008). For example, given the sentence:

(E1) [According to the U.S. Census, whereas only 10% of White Americans live at or below the poverty line today]$_{\mathrm{C1}}$, [28% of African Americans do.]$_{\mathrm{C2}}$[1]

one would extract two independent facts about voter registration, about the two distinct demographic groups. On the other hand, the theory of discourse maintains that part of the above sentence's meaning inheres in the fact that clauses C1 and C2 are juxtaposed (Kehler, 2002). Thus the author intends that we consider them in relation to each other, inviting us to note, for example, a disparity of wealth distribution *between* demographic groups. To fail to capture this is to miss out on an important aspect of text understanding.

We propose the task of Textual Analogy Parsing (TAP) to explicitly capture such relational meaning between analogous facts in text. Concretely, TAP first maps a set of analogous facts to semantic role (SRL) representations, and then identifies the roles along which they are similar (the shared content) and along which they are distinct (the compared content)—see Figure 1. The resulting representation, the TAP frame, is a deeper representation than the one output by shallow discourse parsers (Taboada and Mann, 2006; Prasad et al., 2007; Pitler et al., 2009; Prasad et al., 2010; Surdeanu et al., 2015). Given (E1) above, a shallow discourse parser would classify the relation of contrast between C1 and C2—indicating that some salient differences exist in the meanings of the juxtaposed phrases—but without identifying the na-

---

[*] Author contributed significantly.

[1] Data in E1 and the figure sentence from Morris (2014).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 82–92
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

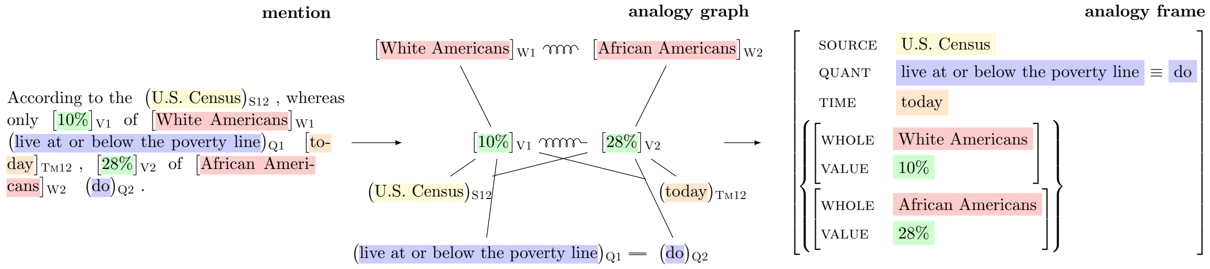

Figure 2: The mapping from utterance to TAP frame. Vertices in the graph are labeled with abbreviated semantic roles. Single lines represent edges between a VALUE and other roles in its associated fact. Double lines represent coreference and synonymy. Springs represent analogy. Note that vertices connected by equivalence arcs, or any span which connects to both V1 and V2 via fact relations (i.e., scope), map to the *shared content* of the TAP frame. Analogous spans map to the *compared* content.

ture of those differences.

We focus on applying TAP to quantitative facts, because TAP frames can be used to create graphical plots from sentences with numbers, as in Figure 1. This new application could help to simplify complex quantitative text on the web (Barrio et al., 2016; Leonhardt et al., 2017). We thus created an expert-annotated dataset of TAP frames over quantitative facts in the Wall Street Journal corpus (Marcus et al., 1999).

We model TAP by jointly predicting SRL representations of facts in a sentence, and higher-order semantic relations between them. Our main findings are that a neural architecture outperforms a log-linear baseline, well-chosen linguistic features help performance, and so does the use of an integer-linear programming (ILP) decoder that enforces the structural constraints of the task. Nevertheless, both quantitative and qualitative evaluation reveal room for improvement on TAP.

In sum, our main contributions are (1) a new task, Textual Analogy Parsing (TAP), that combines shallow semantic parsing with discourse meaning, (2) a dataset of TAP frames from quantitative newswire, and (3) a preliminary study of a new application, automated chart generation from text. All data and code, including standardized evaluation scripts, are made freely available.

2 A Semantic Representation of Analogy

Let us revisit the example sentence from the previous section (E1), where a pair of analogous quantitative facts about poverty rates of different demographic groups are presented in contrast. Individually, these can be represented using the semantic role structures in Figure 3, but representing them separately in this way fails to capture the fact that they are analogous, i.e., structurally and semantically similar but distinct.

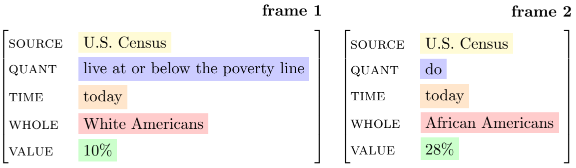

Figure 3: Two analogous quantitative facts represented independently, using the QSRL schema (Lamm et al., 2018).

Instead, we can explicitly show points of similarity and difference between them in the two-tiered frame structure in Figure 2, which we call a TAP frame. The outer tier of the TAP frame contains *shared content*, or information pertinent to all of the facts in question, and the inner tier contains *compared content*, the information that varies across the set of facts.

Mapping from an utterance to a TAP frame requires three types of relational reasoning. Firstly, one must decompose the utterance into a set of facts, where a fact is represented as a set of semantic roles. Then, one must identify the shared content across facts by aligning roles that are semantically equivalent, in the sense that they are either the same span, are coreferent, or are synonymous. For example, in Figure 2 the phrase ‘U.S. Census’ occurs as the SOURCE of both facts because it scopes over the entire sentence in which they appear. Additionally, one must identify the compared content by aligning roles that are analogous, in the sense that they are semantically similar but nevertheless distinct. For example, the phrases ‘White Americans’ and ‘African Americans’ are analogous in our running sentence, playing the same role in their respective facts, while signifying distinct demographic groups.

(a)	[New England Electric]$_{\text{A1}}$ had [offered]$_{\text{Q1}}$ [\$2 billion]$_{\text{V1}}$ to acquire (PS of New Hampshire)$_{\text{TH123}}$, well below the [\$2.29 billion]$_{\text{V2}}$ value [United Illuminating]$_{\text{A2}}$ places on its (bid)$_{\text{Q2}}$ and the [\$2.25 billion]$_{\text{V3}}$ [Northeast]$_{\text{A3}}$ says its (bid)$_{\text{Q3}}$ is worth.
(b)	(First Boston)$_{\text{S12}}$ estimated that (UAL)$_{\text{TH12}}$ was (worth)$_{\text{Q12}}$ [\$ 250 to \$ 344 a share]$_{\text{V1}}$ based on [UAL's results for the 12 months ending last June 30]$_{\text{C1}}$, but only [\$ 235 to \$ 266]$_{\text{V2}}$ based on [a management estimate of results for 1989]$_{\text{C2}}$

Table 1: Representative sentences from the Quantitative TAP dataset. Co-indexing (e.g., A1/Q1) indicates when spans are part of the same QSRL fact. Parentheses indicate shared content spans and brackets indicate compared content spans. To parse (a), one must recognize that '*to acquire PS of New Hampshire*' is elided but nevertheless an implied TH(eme) in two of the clauses, and that '*offered*' and '*bid*' are contextually synonymous Q(uantities). Moreover, one must note that the A(gents) are analogous, and hence part of the compared content. In (b), '*First Boston*', '*UAL*' and '*worth*', contribute a S(ource), TH(eme), and Q(uantity) to the shared content respectively. Here, C(ause) roles are compared content.

	Train ($n = 1000$)			Test ($n = 100$)		
	av.	max	tot.	av.	max	tot.
Count	1.4	3	1383	1.4	3	133
Length	2.6	16	–	2.6	7	–

Table 2: Dataset statistics (average per sentence, max per sentence, and total over the dataset) for the number of analogy frames (Count) and the number of values compared within each frame (Length).

3 The Quantitative TAP Dataset

Motivated by the application of automated graphical plot generation from text, we annotated a dataset of quantitative TAP frames from the Penn Treebank WSJ corpus (Marcus et al., 1999).

As our SRL representation of quantitative facts, we employ the Quantitative Semantic Role Labeling (QSRL) framework we previously defined in Lamm et al. (2018). Having identified a numerical VALUE in text (e.g., 10%), QSRL asks, "what does this number measure?" to determine its associated QUANTITY (e.g., a poverty rate). It might also identify, for example, the WHOLE out of which this percentage is measured (e.g., the set of African Americans), and the TIME at which the quantity took on the value (e.g., today), etc. We employ all fifteen QSRL roles in our annotations.

Our annotations not only capture the relation between a quantitative predicate and its arguments, but also the higher-order analogy relations between them. The distinction is reflected in the sentences in Table 1 from the dataset: Colored spans are co-indexed when they participate in the same quantitative fact; spans with like roles surrounded by parentheses are shared content, meaning that they are either synonymous or co-referent; spans with like roles surrounded by brackets are compared content, meaning that they are analogous but semantically distinct.

To identify instances of quantitative analogy in the WSJ corpus, we first prune out any sentence having fewer than three numerical mentions, where a numerical mention is defined as a contiguous sequence of CD POS tags. Of those left, we manually identify those containing one or more quantitative analogies, i.e., ones in which numerical values are compared content. We estimate the incidence of these to be around 20%. A linguist then annotated 1,100 of these for analogy relationships. See Table 2 for a summary.

Using an independent set of expert annotations on 100 of these sentences, we measured a significant per-token label agreement of 0.882 and edge label agreement of 0.991 using Krippendorff's α.[2]

Table 1 highlights some of the challenging linguistic phenomena in the data. With respect to identifying the shared content of a TAP frame, these can be coarsely divided into two sets. Firstly, in scope, ellipsis, and gapping, a single syntactic element serves as a role in multiple QSRL frames. This is exemplified by the phrase 'PS of New Hampshire' in Table 1(a): It is mentioned explicitly as a THEME of the first fact, and only implied in the second two. Based on a random sample of 100 train sentences, we estimate that 86% of frames in the data exhibit these phenomena. Secondly, in synonymy and coreference, multiple elements appear in a sentence but contribute the same role to the shared content, e.g., 'offered' and 'bid' in Table 1(a). We estimate that 31% of frames in

---

[2] High edge agreement should be expected because edges are type-constrained and thus easy to identify. Additionally, we computed agreement after matching overlapping spans.

the data exhibit these phenomena.

One must learn to identify analogy relationships over a diverse set of compared content roles, with distinct semantic properties: in Table 1(a), AGENT is a compared content role, whereas in Table 1(b), CAUSE is.

4 Modeling TAP in the Quantitative Setting

We model TAP by generating a typed analogy graph over spans of an input text that is isomorphic to the set of TAP frames in that text, e.g., Figure 2. Each vertex in the graph corresponds to a role-labeled span, and edges represent semantic relations between them.

In this graph, each fact is uniquely identified by a VALUE vertex, which is connected via a FACT edge to all of its associated roles. Any two shared content vertices across facts are connected by an EQUIVALENCE edge, indicating that they are coreferent or synonymous. A single vertex can also be shared across facts by linking via a FACT edge to more than one VALUE vertex, suggesting a scopal relationship. Finally, any two vertices which are compared content in the graph are linked via an ANALOGY edge.

More formally, given an utterance $\mathbf{x}$ with tokens $x_1, \ldots, x_n$, let G be a graph with vertices V and edges E. For a vertex $v = (i, j, l) \in V$, $1 \leq i < j \leq n$ are the start and end token indices of a span in $\mathbf{x}$ with role $l \in \mathcal{L}_Q \overset{\text{def}}{=} \{\text{VALUE}, \ldots, \text{QUANT}\}$, the set of QSRL roles. For an edge $e = (v, v', l) \in E$, $v, v' \in V$ and $l \in \mathcal{L}_\mathcal{R} \overset{\text{def}}{=} \{\text{FACT}, \text{EQUIVALENCE}, \text{ANALOGY}\}$.

For G so defined to encode a set of valid TAP frames, it must satisfy certain constraints:

1. **Well-formedness constraints.** For any two vertices $v, v' \in V$, their associated spans must not overlap. Furthermore, every vertex must participate in at least one FACT edge, i.e., no disconnected vertices.
2. **Typing constraints.** FACT relations are always drawn from a VALUE vertex to a non-VALUE vertex. ANALOGY and EQUIVALENCE are only ever drawn between two vertices of the same role.
3. **Unique facts.** If a VALUE vertex v is connected to two distinct vertices v' and v'' of the same role via a FACT edge, then $\text{EQUIVALENCE}(v', v'')$ exists.
4. **Transitivity constraints.** ANALOGY and EQUIVALENCE edges are transitive: if $\text{EQUIVALENCE}(v, v') \in E$ and $\text{EQUIVALENCE}(v', v'') \in E$ then $\text{EQUIVALENCE}(v, v'') \in E$ also. This also holds for ANALOGY edges, but only when v, v' and v'' are VALUE vertices.
5. **Analogy.** There must be at least one pair of analogous VALUE vertices, and for each such pair, there must be a pair of analogous facts connected to them: if v, v' are two VALUE vertices with $\text{ANALOGY}(v, v') \in E$, then there must also exist w, w' as two non-VALUE vertices with $\text{FACT}(v, w) \in E$, $\text{FACT}(v', w') \in E$, $\text{ANALOGY}(w, w') \in E$.

Note that while these constraints rely on the choice of VALUE as the role that grounds quantitative facts, they reflect the general idea that analogy is a structured mapping between meaning representations.

5 A Neural and ILP Model for TAP

We now present a neural and ILP model that predicts analogy graphs as defined in Section 4. Given a sentence, the neural model predicts a distribution over role-labeled spans with edges denoting semantic relations between them. Then, we use an ILP to decode while enforcing the TAP constraints defined in Section 4. Figure 4 presents an overview of the architecture.

Context-sensitive word embeddings. We first encode the words in a sentence by embedding each token using fixed word embeddings. We also concatenate a few linguistic features to the word embeddings, such as named entity tags and dependency relations. These features are generated using CoreNLP (Manning et al., 2014) and represented by randomly-initialized, learned embeddings for symbols together with the fixed word embedding of each token's dependency head and the dependency path length between adjacent tokens. The token embeddings are then passed through several stacked convolutional layers (Kim, 2014). While the first convolutional layer can only capture local information, subsequent layers allow for longer-distance reasoning.

Span prediction. Next, we feed the outputs of a single fully-connected hidden layer to a conditional random field (CRF) (Lafferty et al., 2001),

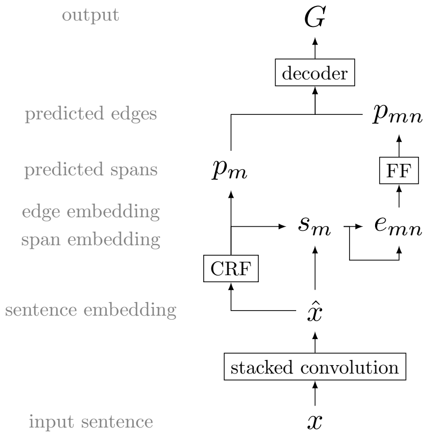

Figure 4: An overview of the proposed neural model: The sentence embedding represents features across the entire sentence using multiple convolutional layers. We then use a conditional random field (CRF) layer to predict labeled spans p_m and to generate span and edge embeddings. We use a feedforward (FF) layer on the edge embeddings to predict edge labels p_{mn}. Together, p_m and p_{mn} form a distribution over edges and labels that we decode into TAP frames.

which defines a joint distribution over per-token role labels. We thus obtain spans from this distribution corresponding to vertices of the graph described in Section 4 by merging contiguous role-labels in the maximum likelihood label sequence predicted by the CRF.

Edge prediction with PATHMAX features. For edge prediction, we use the spans identified above to construct span and edge embeddings: for every span (i, j) that was predicted, we construct a span vector $s_m = \sum_{k=i}^{j} \hat{x}_k$. We also construct a role-label score vector for the span, p_m by summing the role-label probability vectors of its constituent tokens. Then, for every vertex pair (m, n), we construct an edge representation e_{mn}. The basis of this representation is simply the concatenation of the span representations, the sum of the span representations, their respective role-label score vectors p_m and p_n, and relative token distances.

To capture long-distance phenomena like scope, we also incorporate features into e_{mn} from the dependency paths between the two spans by max-pooling the (learned) dependency relation embeddings along the path between the tokens.[3] When computing the representation between two spans, we take the average of the path embedding between each pair of tokens within them. We call this extension PATHMAX.

The resulting edge representation e_{mn} is passed through a single fully-connected hidden layer and an output layer to predict a distribution over edge labels p_{mn}, for each pair of spans.

Training. The supervised data described in Section 3 provides gold spans and edges between them. Thus we define a loss function with two terms: one for the log-likelihood of the span labels output by the CRF model, and one for the cross-entropy loss on the edge labels. We train the span and edge components of the model jointly.

Decoding. We consider two methods for decoding the span-level and edge-level label distributions p_m and $p_{m,n}$ into a labeled graph respecting the constraints described in Section 4.

As a simple greedy method to enforce these constraints, we begin by picking the most likely role for each span and edge and then discarding any edges and spans that violate the well-formedness (1) and typing constraints (2). We then enforce transitivity constraints (4) by incrementally building a cluster of analogous and equivalent spans. We then resolve the unique facts constraint (3) by keeping only the span with highest FACT edge score. Finally, for every cluster of analogous VALUE spans, we check that the analogy constraint (5) holds and if not, discard the cluster.

We also implement an optimal decoder that encodes the TAP constraints as an ILP (Roth and Yih, 2004; Do et al., 2012). The ILP tries to find an optimal decoding according to the model, subject to hard constraints imposed on the solution space. For example, we require that solutions satisfy the 'connected spans' constraint:

$$\forall s \exists s' : e(s, s', \text{FACT})$$

In plain English, this says that every span s in a solution must be connected via a FACT edge to some other span s'. See the supplementary material for the full list of constraints we employ. We solve the ILPs with Gurobi (Gurobi Optimization, Inc., 2018).

6 Experiments

We now describe the experimental setup of our neural model (Section 5) on the dataset of TAP frames we created (Section 3). Results and discussion are reported in Section 7.

---

[3] The dependency paths are directed but unlexicalized.

Evaluation metrics. The primary metric we use to measure the accuracy of a system on frame prediction is the precision, recall and F_1 between the labeled vertex-edge-vertex triples predicted by the model and those in the gold parse. If there are multiple predicted spans that overlap with a single gold span or vice versa, we find a matching of predicted and gold spans that maximizes overlap.

In addition to the primary metric, we also report precision, recall and F_1 when predicting labeled (non-VALUE) spans and predicting labeled edges before performing any decoding.[4] We also use the matching process described above for both these sets of metrics. Standardized evaluation code is provided with the dataset.

Experimental setup. We compare the neural models presented in Section 5 in addition to a log-linear baseline. The log-linear baseline uses the same fixed word embeddings as the neural model in addition to the named entity and dependency parse features described in Section 5. The key difference is that instead of learning a sentence embedding or hidden layers, the log-linear model simply uses a CRF to predict span labels directly from fixed input features, and then uses a single sigmoid layer to predict edge labels from deterministic edge embeddings, e_{mn}.

For the neural models, we used three convolutional layers for sentence embedding with a filter size of 3. Every layer other than the input layer used a hidden dimension of 50 with ReLU nonlinearities. We introduced a single dropout layer ($p = 0.5$) between every two layers in the network (including at the input). We used 50-dimensional GloVe embeddings (Pennington et al., 2014) learned from Wikipedia 2014 and Gigaword 5 as pre-trained word embeddings, and initialized the embeddings for the features randomly. We chose relatively low input- and hidden-vector dimension because of the size of our data. The network was trained for 15 epochs using ADADELTA (Zeiler, 2012) with a learning rate of 1.0. All models were implemented in `PyTorch` (Paszke et al., 2017).

7 Results and Discussion

			Frame prediction		
Model	Feats.	Dec.	P	R	F_1
Log-linear	✓	gr.	46.3	21.8	29.7
Log-linear	✓	opt.	37.1	27.5	31.6
Neural	×	gr.	50.7	38.4	43.7
Neural	×	opt.	52.8	48.6	50.6
Neural	✓	gr.	54.9	57.4	56.1
Neural	✓	opt.	**56.4**	**68.8**	**62.0**

Table 3: Performance of models on the test data. Combining the neural model with linguistic features and using an optimal decoder to enforce semantic constraints led to the best performance.

	Span prediction		
Model	P	R	F_1
Log-linear (all feats.)	**42.8**	**82.3**	**56.3**
Neural (no feats.)	41.7	79.1	54.6
Neural (all feats.)	41.5	79.2	54.4
w/o NER	41.6	79.1	54.5
w/o dep.	41.2	77.5	53.8
w/o CRF	36.1	73.1	48.3

Table 4: Performance of models on labeled (non-VALUE) span prediction during cross-validation prior to decoding. We found using a CRF to be the most important aspect: simply using fixed word vectors with a CRF (i.e., the log-linear model) was sufficient to predict spans.

Frame prediction results on the test set are summarized in Table 3. Our three main findings are that (i) the neural network model far outperforms the log-linear model on our frame metric, (ii) including linguistic features further increases performance, and (iii) so does using an optimal decoder over a greedy method.

Quantitative error analysis. To better understand which aspects of our model contribute to the task, we perform an ablation study on the span and edge predictions of our model *prior to decoding*.

With respect to span prediction (Table 4), we found that the fixed word vectors, along with a CRF, were able to capture the information needed to identify QSRL role-spans. Indeed, the log-linear baseline, which directly uses these word vectors as features for a CRF, did the best at span prediction. We believe that the drop in performance from introducing hidden layers with the neural models is a result of the model updating its span representations to do better edge prediction.[5]

---

[4] We exclude VALUE spans from span scores because they are easy to predict and thus inflate model performance.

[5] In a separate experiment, the neural model outperformed the log-linear model when they were trained only to do span prediction.

Model	Edge prediction P	Edge prediction R	Edge prediction F_1
Log-linear (all feats.)	33.6	15.7	18.9
Neural (no feats.)	73.7	65.8	68.7
Neural (all feats.)	74.4	**75.6**	**74.7**
w/o NER	74.8	72.2	73.1
w/o dep.	73.4	65.0	68.2
w/o PATHMAX.	72.8	64.0	67.2

Table 5: Performance of models on labeled edge prediction during cross-validation prior to decoding. We found that both dependency label (dep.) and path features (PATHMAX) help significantly.

While the log-linear model did well at predicting spans, it did a poor job predicting edges, indicating that learning to extract higher-order features from learned span embeddings is necessary for identifying semantic relations between them (Table 5). We also found that linguistic features were important: in particular, we found that syntactic features – the dependency path features (PATHMAX) and dependency labels – played a big role in edge prediction, followed by type information from NER tags.

Qualitative error analysis. Our model is tasked with jointly identifying QSRL parses of analogous facts in a sentence, and ANALOGY and EQUIVALENCE relations among them. As described in Section 4, these pieces interact in mutually constraining ways, and thus it is possible for local errors to have global effects on predicted frames.

In Figure 5, for example, the model correctly identifies the gold TIME spans as part of a TAP frame, but mistakenly predicts that they are linked by EQUIVALENCE, and thus modify the same VALUE span. In the gold parse, they are linked by ANALOGY, and modify distinct VALUE spans. As a result of this misclassification, the model leaves out an entire QSRL fact from the resulting parse.

In many cases, the model successfully identifies compared content roles between QSRL facts. In Figure 6, we show an example where it does not manage to do so. Here, unable to identify the ANALOGY relation between the phrases '*Those with a bullish view*' and '*the dollar bears*', the model instead chooses two identical sequences 'the dollar' as the non-VALUE compared content. Inspecting edge probability scores from the model *before decoding* reveals that the neural model thinks that the first instance of 'the dollar' in the sentence is semantically analogous to the second; it can be confused by surface similarity into classifying ANALOGY relations.

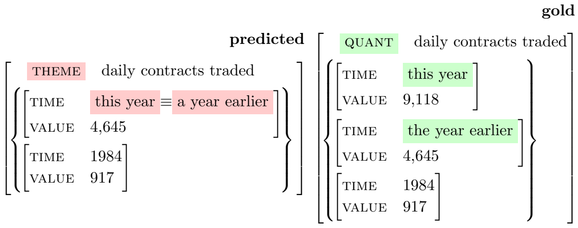

Figure 5: TAP frames for the sentence, '*This year ... daily contracts traded totaled 9,118, up from 4,645 a year earlier and from 917 in 1984.*' The model not only misclassifies the QSRL role of '*daily contracts traded*', but also mistakenly identifies an EQUIVALENCE between '*this year*' and '*the year earlier*'. As a result, the VALUE 9,118 is left without a compared content role, and is dropped.

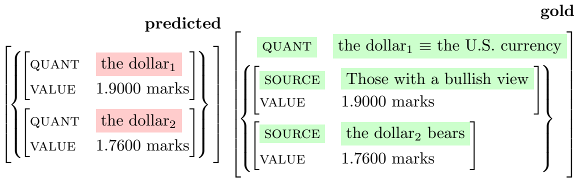

Figure 6: TAP frames for the sentence '*Those with a bullish view see [the dollar]$_1$ trading up near 1.900 marks... while [the dollar]$_2$ bears see the U.S. currency trading around 1.7600 marks*'. Among other errors, the model failed to identify analogous SOURCE spans and instead predicts that the two instances of the phrase '*the dollar*' (indicated with indexing) in the sentence contribute non-VALUE compared content.

Application to plot generation. As we have seen, textual analogy is frequently used to compare quantities along some axis of differentiation. For example, one might compare the stock prices of different companies, or describe the change in some quantity's value over time. Such analogy relationships can alternately be expressed in the form of a plot.

Indeed, there is a natural correspondence between charts and TAP frames over quantitative facts: VALUES of a quantitative TAP frame are plotted against other compared content roles, and elements of the shared content correspond with scopal chart elements, such as titles. This mapping is well-defined provided analogous values share units. We present some initial results exploring this direction.

In Figure 7, we deterministically plot TAP

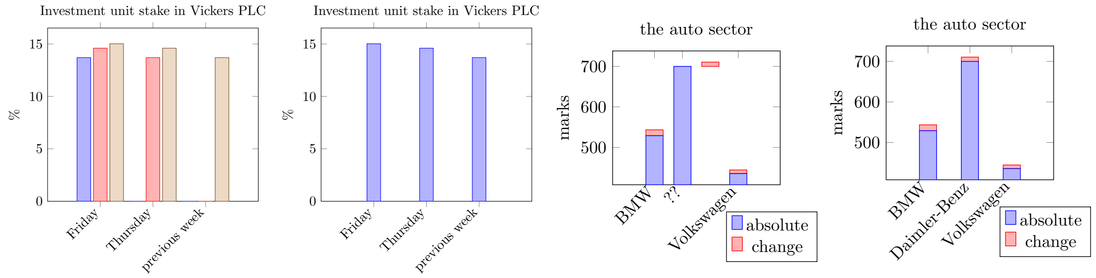

(a) Before constraints (b) After constraints (c) Before constraints (d) After constraints

Figure 7: Charts generated from TAP frames. Charts (a) and (b) are generated from the sentence '*Vicker's PLC ... raised its stake in the company Friday to 15.02% from about 14.6% Thursday and from 13.6% the previous week.*' Before imposing constraints, the neural model assigns multiple values to the TIME arguments '*Thursday*' and '*Friday*', over-extending their scope. Imposing structural constraints ensures the correct assignment of TIMES to VALUES. Charts (c) and (d) are generated from the sentence '*In the auto sector, Bayerische Motoren Werke plunged 14.5 marks to 529 marks, Daimler-Benz dropped 10.5 to 700, and Volkswagen slumped 9 to 435.5.*' Here, the model fails to associate an absolute (blue) and relative (red) VALUE pair with a THEME role. The imposition of global constraints corrects this, linking them to the THEME 'Diamler-Benz'.

frames generated by our system both before and after the imposition of global analogy constraints, for two sentences in the data. In the first sentence, VALUE spans are plotted against the TIME spans the model associates with their respective facts. In the second sentence, two analogy frames are plotted together, one reflecting the absolute values of the stock prices mentioned (blue) and the other reflecting the changes in prices mentioned (red). Units are extracted from VALUE spans using simple pattern matching. Chart titles are only illustrative and were generated by stitching together shared content identified by our system.

Note that with the imposition of global constraints reflecting the structure of analogy, the system yields well-formed charts. Without these constraints, generated charts either have multiple y-axis values assigned to the same x-axis value, or have floating y-axis values with no grounding on the x-axis.

8 Related Work

Analogy. In the cognitive science literature, analogy is a general form of relational reasoning unique to human cognition (Tversky and Gati, 1978; Holyoak and Thagard, 1996; Goldstone and Son, 2005; Penn et al., 2008; Holyoak, 2012). Our model of textual analogy is particularly influenced by Structure Mapping Theory (Falkenhainer et al., 1989; Gentner and Markman, 1997), an influential cognitive model of analogy as a structure-preserving map between concepts.

Within the NLP community, there has been much work focused on inferring lexical analogies between generic concepts, e.g., *tennis:racket::baseball:bat* (Mikolov et al., 2013; Turney, 2013), from global distributional statistics. Such analogies are generic, type-level patterns whose structure exists in the nature of the language; here, we are interested in specific analogies whose structure is conveyed by a particular sentence.

Discourse and Information Extraction. TAP is an information extraction task that synthesizes ideas from semantic role labeling on the one hand and discourse parsing on the other. The former produces predicate-argument representations of individual facts in a text (Baker et al., 1998; Gildea and Jurafsky, 2002; Palmer et al., 2005); the latter identifies discourse relations between syntactic clauses (Taboada and Mann, 2006; Prasad et al., 2007; Pitler et al., 2009; Prasad et al., 2010; Surdeanu et al., 2015).

TAP first maps from syntax to a set of SRL-style representations, and then identifies structurally-constrained, higher-order relations among them. It is in this sense reminiscent of, but distinct from, work on causal processes by Berant et al. (2014).

Numbers in NLP. There has been some work on understanding numbers in text. This includes quantitative reasoning (Kushman et al., 2014; Roy et al., 2015), numerical information extraction (Madaan et al., 2016), and techniques for making numbers more easily interpretable in text (Chaganty and Liang, 2016; Kim et al., 2016).

If pursued further, the application of plotting quantitative text that we discuss in this paper could help to clarify quantitative text on the web (Larkin and Simon, 1987; Barrio et al., 2016).

Neural modeling. Recent work has shown the promise of sophisticated neural models on semantic role labeling (He et al., 2017). Similar to other such sequence prediction models, e.g., those for named entity recognition (Lample et al., 2016) or semantic role labeling (Zhou and Xu, 2015), our span prediction utilizes a neural CRF. Our model also has an edge-prediction component, which benefits from a simplified version of the PathLSTM model of Roth and Lapata (2016). Our edge-prediction model also uses an embedding concatenation component, which was inspired by recent work on neural coreference resolution (Lee et al., 2017). He et al. (2017) also impose semantic constraints during prediction, but use A* search instead of an ILP.

9 Conclusion

In this paper we have presented a new task, textual analogy parsing, or TAP. Given a sentence about a set of analogous facts, TAP outputs a frame representation that expresses the points of similarity and difference in their meanings.

We note that in the particular case of quantitative text, TAP frames correspond with charts. We develop a new dataset of TAP frames from quantitative newswire, and compare a variety models for TAP. Our best model employs a globally optimal decoder to enforce the structural constraints of analogy; its outputs can be mapped to well-formed charts of quantitative information extracted from text.

We view this work to be an exciting step in the direction of deeper discourse modeling. Future work might further extend the recovery of analogy as part of information extraction. This might include TAP outside of the quantitative domain, or TAP at the paragraph level.

Acknowledgments

We would like to thank the members of the Stanford NLP Group for reviewing early versions of the paper, and would also like to thank the anonymous reviewers for their thoughtful feedback.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90.

Pablo J Barrio, Daniel G Goldstein, and Jake M Hofman. 2016. Improving Comprehension of Numbers in the News. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2729–2739.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling Biological Processes for Reading Comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510.

Arun Chaganty and Percy Liang. 2016. How Much is 131 Million Dollars? Putting Numbers in Perspective with Compositional Descriptions. In *ACL*, pages 578–587.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint Inference for Event Timeline Construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 677–687.

Brian Falkenhainer, Kenneth D Forbus, and Dedre Gentner. 1989. The Structure-Mapping Engine: Algorithm and Examples. *Artificial Intelligence*, 41(1):1–63.

Dedre Gentner and Arthur B Markman. 1997. Structure Mapping in Analogy and Similarity. *American Psychologist*, 52(1):45.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational linguistics*, 28(3):245–288.

Robert L Goldstone and Ji Y Son. 2005. The Transfer of Scientific Principles Using Concrete and Idealized Simulations. *The Journal of the Learning Sciences*, 14(1):69–110.

Gurobi Optimization, Inc. 2018. Gurobi Optimizer Reference Manual. `http://www.gurobi.com`.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep Semantic Role Labeling: What Works and What's Next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 473–483.

Keith Holyoak and Paul Thagard. 1996. *Mental Leaps: Analogy in Creative Thought*. MIT Press, Cambridge, Mass.

Keith J Holyoak. 2012. Analogy and Relational Reasoning. *The Oxford Handbook of Thinking and Reasoning*, pages 234–259.

Andrew Kehler. 2002. *Coherence, Reference, and the Theory of Grammar*. CSLI Publications, Stanford, CA.

Yea-Seul Kim, Jessica Hullman, and Maneesh Agrawala. 2016. Generating Personalized Spatial Analogies for Distances and Areas. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 38–48.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to Automatically Solve Algebra Word Problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 271–281.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML 2001, pages 282–289.

Matthew Lamm, Arun Chaganty, Dan Jurafsky, Christopher D. Manning, and Percy Liang. 2018. QSRL: A Semantic Role-Labeling Schema for Quantitative Text. In *First Financial Narrative Processing Workshop at LREC 2018*, Miyazaki, Japan.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Jill H Larkin and Herbert A Simon. 1987. Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive science*, 11(1):65–100.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end Neural Coreference Resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197.

David Leonhardt, Jodi Rudoren, Jon Galinsky, Karron Skog, Marc Lacey, Tom Giratikanon, and Tyson Evans. 2017. Journalism That Stands Apart: The Report of the 2020 Group. Technical report, The New York Times.

Aman Madaan, Ashish Mittal, G Ramakrishnan Mausam, Ganesh Ramakrishnan, and Sunita Sarawagi. 2016. Numerical Relation Extraction with Minimal Supervision. In *AAAI*, pages 2764–2771.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Michell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3 ldc99t42.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Monique W. Morris. 2014. *Black Stats*. The New Press, New York, NY.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational linguistics*, 31(1):71–106.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Workshop on Autodifferentiation*.

Derek C Penn, Keith J Holyoak, and Daniel J Povinelli. 2008. Darwin's mistake: Explaining the discontinuity between human and nonhuman minds. *Behavioral and Brain Sciences*, 31(2):109–130.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 2, pages 683–691.

Rashmi Prasad, Aravind K Joshi, and Bonnie L Webber. 2010. Exploiting Scope for Shallow Discourse Parsing. In *LREC*.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie L Webber. 2007. The Penn Discourse Treebank 2.0 Annotation Manual. Technical report, Institute for Research in Cognitive Sciences.

Dan Roth and Wen-tau Yih. 2004. A Linear Programming Formulation for Global Inference in Natural Language Tasks. Technical report, Illinois University at Urbana-Champaign, Dept of Computer Science.

Michael Roth and Mirella Lapata. 2016. Neural Semantic Role Labeling with Dependency Path Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1192–1202.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

Sunita Sarawagi. 2008. Information Extraction. *Foundations and Trends in Databases*, 1(3):261–377.

Mihai Surdeanu, Tom Hicks, and Marco Antonio Valenzuela-Escarcega. 2015. Two Rractical Rhetorical Structure Theory Parsers. In *HLT-NAACL*, pages 1–5.

Maite Taboada and William C Mann. 2006. Rhetorical Structure Theory: Looking back and moving ahead. *Discourse Studies*, 8(3):423–459.

Peter D Turney. 2013. Distributional semantics beyond words: Supervised learning of analogy and paraphrase. *arXiv preprint arXiv:1310.5042*.

Amos Tversky and Itamar Gati. 1978. Studies of Similarity. *Cognition and Categorization*, 1(1978):79–98.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Jie Zhou and Wei Xu. 2015. End-to-end Learning of Semantic Role Labeling Using Recurrent Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1127–1137.

SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference

Rowan Zellers[♠] **Yonatan Bisk**[♠] **Roy Schwartz**[♠♡] **Yejin Choi**[♠♡]
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♡]Allen Institute for Artificial Intelligence
{rowanz,ybisk,roysch,yejin}@cs.washington.edu
https://rowanzellers.com/swag

Abstract

Given a partial description like "she opened the hood of the car," humans can reason about the situation and anticipate what might come next ("then, she examined the engine"). In this paper, we introduce the task of grounded commonsense inference, unifying natural language inference and commonsense reasoning.

We present SWAG, a new dataset with 113k multiple choice questions about a rich spectrum of grounded situations. To address the recurring challenges of the annotation artifacts and human biases found in many existing datasets, we propose *Adversarial Filtering* (AF), a novel procedure that constructs a de-biased dataset by iteratively training an ensemble of stylistic classifiers, and using them to filter the data. To account for the aggressive adversarial filtering, we use state-of-the-art language models to massively oversample a diverse set of potential counterfactuals. Empirical results demonstrate that while humans can solve the resulting inference problems with high accuracy (88%), various competitive models struggle on our task. We provide comprehensive analysis that indicates significant opportunities for future research.

On stage, a woman takes a seat at the piano. She
a) sits on a bench as her sister plays with the doll.
b) smiles with someone as the music plays.
c) is in the crowd, watching the dancers.
d) nervously sets her fingers on the keys.

A girl is going across a set of monkey bars. She
a) jumps up across the monkey bars.
b) struggles onto the monkey bars to grab her head.
c) gets to the end and stands on a wooden plank.
d) jumps up and does a back flip.

The woman is now blow drying the dog. The dog
a) is placed in the kennel next to a woman's feet.
b) washes her face with the shampoo.
c) walks into frame and walks towards the dog.
d) tried to cut her face, so she is trying to do something very close to her face.

Table 1: Examples from SWAG; the correct answer is **bolded**. Adversarial Filtering ensures that stylistic models find all options equally appealing.

1 Introduction

When we read a story, we bring to it a large body of implicit knowledge about the physical world. For instance, given the context "on stage, a woman takes a seat at the piano," shown in Table 1, we can easily infer what the situation might *look* like: a woman is giving a piano performance, with a crowd watching her. We can furthermore infer her likely *next* action: she will most likely set her fingers on the piano keys and start playing.

This type of natural language inference requires commonsense reasoning, substantially broadening the scope of prior work that focused primarily on linguistic entailment (Chierchia and McConnell-Ginet, 2000). Whereas the dominant entailment paradigm asks if two natural language sentences (the 'premise' and the 'hypothesis') describe the same set of possible worlds (Dagan et al., 2006; Bowman et al., 2015), here we focus on whether a (multiple-choice) ending describes a possible (*future*) world that can be anticipated from the situation described in the premise, even when it is not strictly entailed. Making such inference necessitates a rich understanding about everyday physical situations, including object affordances (Gibson, 1979) and frame semantics (Baker et al., 1998).

A first step toward grounded commonsense inference with today's deep learning machinery is to create a large-scale dataset. However, recent work has shown that human-written datasets are susceptible to *annotation artifacts*: unintended stylistic patterns that give out clues for the gold labels (Gururangan et al., 2018; Poliak et al., 2018). As a result, models trained on such datasets with hu-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 93–104
Brussels, Belgium, October 31 - November 4, 2018.

man biases run the risk of over-estimating the actual performance on the underlying task, and are vulnerable to adversarial or out-of-domain examples (Wang et al., 2018; Glockner et al., 2018).

In this paper, we introduce Adversarial Filtering (AF), a new method to automatically detect and reduce stylistic artifacts. We use this method to construct **SWAG**: an adversarial dataset with 113k multiple-choice questions. We start with pairs of temporally adjacent video captions, each with a context and a follow-up event that we *know* is physically possible. We then use a state-of-the-art language model fine-tuned on this data to massively oversample a diverse set of possible negative sentence endings (or *counterfactuals*). Next, we filter these candidate endings aggressively and adversarially using a committee of trained models to obtain a population of de-biased endings with similar stylistic features to the real ones. Finally, these filtered counterfactuals are validated by crowd workers to further ensure data quality.

Extensive empirical results demonstrate unique contributions of our dataset, complementing existing datasets for natural langauge inference (NLI) (Bowman et al., 2015; Williams et al., 2018) and commonsense reasoning (Roemmele et al., 2011; Mostafazadeh et al., 2016; Zhang et al., 2017). **First**, our dataset poses a new challenge of grounded commonsense inference that is easy for humans (88%) while hard for current state-of-the-art NLI models (<60%). **Second**, our proposed adversarial filtering methodology allows for cost-effective construction of a large-scale dataset while substantially reducing known annotation artifacts. The generality of adversarial filtering allows it to be applied to build future datasets, ensuring that they serve as reliable benchmarks.

2 SWAG: Our new dataset

We introduce a new dataset for studying physically grounded commonsense inference, called **SWAG**.[1] Our task is to predict which event is most likely to occur next in a video. More formally, a model is given a context $\boldsymbol{c} = (\boldsymbol{s}, \boldsymbol{n})$: a complete sentence $\boldsymbol{s}$ and a noun phrase $\boldsymbol{n}$ that begins a second sentence, as well as a list of possible verb phrase sentence endings $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_4\}$. See Figure 1 for an example triple $(\boldsymbol{s}, \boldsymbol{n}, \boldsymbol{v}_i)$. The model must then select the most appropriate verb phrase $\boldsymbol{v}_{\hat{i}} \in \boldsymbol{V}$.

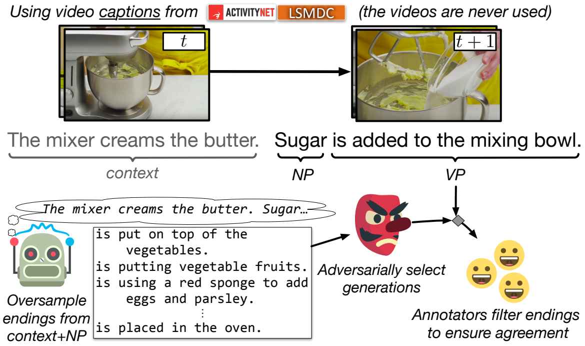

Figure 1: Overview of the data collection process. For a pair of sequential video captions, the second caption is split into noun and verb phrases. A language model generates many negative endings, of which a difficult subset are human-annotated.

Overview Our corpus consists of 113k multiple choice questions (73k training, 20k validation, 20k test) and is derived from pairs of consecutive video captions from ActivityNet Captions (Krishna et al., 2017; Heilbron et al., 2015) and the Large Scale Movie Description Challenge (LSMDC; Rohrbach et al., 2017). The two datasets are slightly different in nature and allow us to achieve broader coverage: ActivityNet contains 20k YouTube clips containing one of 203 activity types (such as doing gymnastics or playing guitar); LSMDC consists of 128k movie captions (audio descriptions and scripts). For each pair of captions, we use a constituency parser (Stern et al., 2017) to split the second sentence into noun and verb phrases (Figure 1).[2] Each question has a human-verified gold ending and 3 distractors.

3 A solution to annotation artifacts

In this section, we outline the construction of **SWAG**. We seek dataset diversity while minimizing *annotation artifacts*, conditional stylistic patterns such as length and word-preference biases. For many NLI datasets, these biases have been shown to allow shallow models (e.g. bag-of-words) obtain artificially high performance.

To avoid introducing easily "gamed" patterns, we present Adversarial Filtering (AF), a generally-applicable treatment involving the iterative refinement of a set of assignments to increase the entropy under a chosen model family. We then discuss how we generate counterfactual endings, and

---

[1] Short for **S**ituations **W**ith **A**dversarial **G**enerations.

[2] We filter out sentences with rare tokens ($\leq$3 occurrences), that are short ($l \leq 5$), or that lack a verb phrase.

Algorithm 1 Adversarial filtering (AF) of negative samples. During our experiments, we set $N^{easy} = 2$ for refining a population of $N^{-} = 1023$ negative examples to $k = 9$, and used a 80%/20% train/test split.

```
while convergence not reached do
    • Split the dataset $\mathcal{D}$ randomly up into training and testing portions $\mathcal{D}^{tr}$ and $\mathcal{D}^{te}$.
    • Optimize a model $f_\theta$ on $\mathcal{D}^{tr}$.
    for index $i$ in $\mathcal{D}^{te}$ do
        • Identify easy indices:
        $\mathcal{A}_i^{easy} = \{j \in \mathcal{A}_i : f_\theta(x_i^+) > f_\theta(x_{i,j}^-)\}$
        • Replace $N^{easy}$ easy indices $j \in \mathcal{A}_i^{easy}$ with adversarial indices $k \notin \mathcal{A}_i$ satisfying $f_\theta(x_{i,k}^-) > f_\theta(x_{i,j}^-)$.
    end for
end while
```

finally, the models used for filtering.

3.1 Formal definition

In this section, we formalize what it means for a dataset to be *adversarial*. Intuitively, we say that an adversarial dataset for a model f is one on which f will not generalize, even if evaluated on test data from the same distribution. More formally, let our input space be $\mathcal{X}$ and the label space be $\mathcal{Y}$. Our trainable classifier f, taking parameters θ is defined as $f_\theta : \mathcal{X} \rightarrow \mathbb{R}^{|\mathcal{Y}|}$. Let our dataset of size N be defined as $\mathcal{D} = \{(x_i, y_i)\}_{1 \le i \le N}$, and let the loss function over the dataset be $L(f_\theta, \mathcal{D})$. We say that a dataset is *adversarial* with respect to f if we expect high empirical error I over all leave-one-out train/test splits (Vapnik, 2000):

$$I(\mathcal{D}, f) = \frac{1}{N} \sum_{i=1}^{N} L(f_{\theta_i^\star}, \{(x_i, y_i)\}), \qquad (1)$$

$$\text{where } \theta_i^\star = \operatorname*{argmin}_{\theta} L(f_\theta, \mathcal{D} \setminus \{(x_i, y_i)\}), \qquad (2)$$

with regularization terms omitted for simplicity.

3.2 Adversarial filtering (AF) algorithm

In this section, we outline an approach for generating an adversarial dataset $\mathcal{D}$, effectively maximizing empirical error I with respect to a family of trainable classifiers f. Without loss of generality, we consider the situation where we have N *contexts*, each associated with a single positive example $(x_i^+, 1) \in \mathcal{X} \times \mathcal{Y}$, and a large population of context-specific negative examples $(x_{i,j}^-, 0) \in \mathcal{X} \times \mathcal{Y}$, where $1 \le j \le N^-$ for each i. For instance, the negative examples could be incorrect relations in knowledge-base completion (Socher et al., 2013), or all words in a dictionary for a single-word cloze task (Zweig and Burges, 2011).

Our goal will be to filter the population of negative examples for each instance i to a size of $k \ll N^-$. This will be captured by returning a set of *assignments* $\mathcal{A}$, where for each instance the assignment will be a k-subset $\mathcal{A}_i = [1 \ldots N^-]^k$. The filtered dataset will then be:

$$\mathcal{D}^{AF} = \{(x_i, 1), \{(x_{i,j}^-, 0)\}_{j \in \mathcal{A}_i}\}_{1 \le i \le N} \qquad (3)$$

Unfortunately, optimizing $I(\mathcal{D}^{AF}, f)$ is difficult as $\mathcal{A}$ is global and non-differentiable. To address this, we present Algorithm 1. On each iteration, we split the data into dummy 'train' and 'test' splits. We train a model f on the training portion and obtain parameters θ, then use the remaining test portion to reassign the indices of $\mathcal{A}$. For each context, we replace some number of 'easy' negatives in $\mathcal{A}$ that f_θ classifies correctly with 'adversarial' negatives outside of $\mathcal{A}$ that f_θ misclassifies.

This process can be thought of as increasing the overall entropy of the dataset: given a strong model f_θ that is compatible with a random subset of the data, we aim to ensure it cannot generalize to the held-out set. We repeat this for several iterations to reduce the generalization ability of the model family f over arbitrary train/test splits.

3.3 Generating candidate endings

To generate counterfactuals for **SWAG**, we use an LSTM (Hochreiter and Schmidhuber, 1997) language model (LM), conditioned on contexts from video captions. We first pretrain on BookCorpus (Zhu et al., 2015), then finetune on the video caption datasets. The architecture uses standard best practices and was validated on held-out perplexity of the video caption datasets; details are in the appendix. We use the LM to sample N^-=1023 unique endings for a partial caption.[3]

Importantly, we *greedily* sample the endings, since beam search decoding biases the generated endings to be of lower perplexity (and thus easily distinguishable from found endings). We find this process gives good counterfactuals: the generated endings tend to use *topical* words, but often make little sense physically, making them perfect for our task. Further, the generated endings are marked as "gibberish" by humans only 9.1% of the time (Sec 3.5); in that case the ending is filtered out.

---

[3] To ensure that the LM generates unique endings, we split the data into five validation folds and train five separate LMs, one for each set of training folds. This means that each LM never sees the found endings during training.

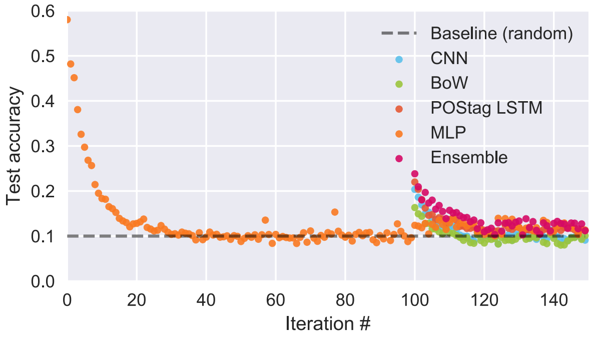

Figure 2: Test accuracy by AF iteration, under the negatives given by $\mathcal{A}$. The accuracy drops from around 60% to close to random chance. For efficiency, the first 100 iterations only use the MLP.

3.4 Stylistic models for adversarial filtering

In creating SWAG, we designed the model family f to pick up on low-level *stylistic features* that we posit should not be predictive of whether an event happens next in a video. These stylistic features are an obvious case of annotation artifacts (Cai et al., 2017; Schwartz et al., 2017).[4] Our final classifier is an ensemble of four stylistic models:

1. A multilayer perceptron (MLP) given LM perplexity features and context/ending lengths.
2. A bag-of-words model that averages the word embeddings of the second sentence as features.
3. A one-layer CNN, with filter sizes ranging from 2-5, over the second sentence.
4. A bidirectional LSTM over the 100 most common words in the second sentence; uncommon words are replaced by their POS tags.

We ensemble the models by concatenating their final representations and passing it through an MLP. On every adversarial iteration, the ensemble is trained jointly to minimize cross-entropy.

The accuracies of these models (at each iteration, evaluated on a 20% split of the test dataset before indices of $\mathcal{A}$ get remapped) are shown in Figure 2. Performance decreases from 60% to close to random chance; moreover, confusing the perplexity-based MLP is not sufficient to lower performance of the ensemble. Only once the other stylistic models are added does the ensemble accuracy drop substantially, suggesting that our approach is effective at reducing stylistic artifacts.

Imagine that you are watching a video clip. The clip has a caption, but it is missing the final phrase. Please choose the best 2 caption endings, and classify each as:

- **likely**, if it completes the caption in a reasonable way;
- **unlikely**, if it sounds ridiculous or impossible;
- **gibberish** if it has such serious errors that it doesn't feel like a valid English sentence.

Example: Someone is shown sitting on a fence and talking to the camera while pointing out horses. Someone

- stands in front of a podium. (**likely**, **second best**)
- rides a horse using a hose. (**unlikely**)
- is shown riding a horse. (**likely**, **best**)
- , the horse in a plaza field. (**gibberish**)

Figure 3: Mechanical Turk instructions (abridged).

3.5 Human verification

The final data-collection step is to have humans verify the data. Workers on Amazon Mechanical Turk were given the caption context, as well as six candidate endings: one found ending and five adversarially-sampled endings. The task was twofold: Turkers ranked the endings independently as likely, unlikely, or gibberish, and selected the best and second best endings (Fig 3).

We obtained the correct answers to each context in two ways. If a Turker ranks the found ending as either best or second best (73.7% of the time), we add the found ending as a gold example, with negatives from the generations not labelled best or gibberish. Further, if a Turker ranks a generated ending as best, and the found ending as second best, then we have reason to believe that the generation is good. This lets us add an additional training example, consisting of the generated best ending as the gold, and remaining generations as negatives.[5] Examples with $\leq$3 non-gibberish endings were filtered out.[6]

We found after 1000 examples that the annotators tended to have high agreement, also generally choosing found endings over generations (see Table 2). Thus, we collected the remaining 112k examples with one annotator each, periodically verifying that annotators preferred the found endings.

4 Experiments

In this section, we evaluate the performance of various NLI models on SWAG. Recall that models

---

[4] A broad definition of annotation artifacts might include aspects besides lexical/stylistic features: for instance, certain events are less likely semantically regardless of the context (e.g. riding a horse using a hose). For this work, we erred more conservatively and only filtered based on style.

[5] These two examples share contexts. To prevent biasing the test and validation sets, we didn't perform this procedure on answers from the evaluation sets' context.

[6] To be data-efficient, we reannotated filtered-out examples by replacing gibberish endings, as well as generations that outranked the found ending, with candidates from $\mathcal{A}$.

	Label distribution by ending type		Inter-annotator agreement	
Labels	Found end	Gen. end	α	ppa
Best	53.5%	9.3%	0.43	72%
Second Best	20.2%	15.9%		
Neither	26.3%	74.8%		
Likely	80.3%	33.3%	0.39	64%
Unlikely	19.0%	57.5%		
Gibberish	0.7%	9.1%		

Table 2: Annotators tend to label the found ending as likely and within the top 2 (column 2), in other cases the example is filtered out. Both label groups have high inter-annotator agreement, in terms of Krippendorff's α and pairwise percent agreement.

for our dataset take the following form: given a sentence and a noun phrase as context $\boldsymbol{c} = (\boldsymbol{s}, \boldsymbol{n})$, as well as a list of possible verb phrase endings $\boldsymbol{V} = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_4\}$, a model f_θ must select a verb $\hat{\imath}$ that hopefully matches i_{gold}:

$$\hat{\imath} = \operatorname*{argmax}_i f_\theta(\boldsymbol{s}, \boldsymbol{n}, \boldsymbol{v}_i) \tag{4}$$

To study the amount of bias in our dataset, we also consider models that take as input just the ending verb phrase $\boldsymbol{v}_i$, or the entire second sentence $(\boldsymbol{n}, \boldsymbol{v}_i)$. For our learned models, we train f by minimizing multi-class cross-entropy. We consider three different types of word representations: 300d GloVe vectors from Common Crawl (Pennington et al., 2014), 300d Numberbatch vectors retrofitted using ConceptNet relations (Speer et al., 2017), and 1024d ELMo contextual representations that show improvement on a variety of NLP tasks, including standard NLI (Peters et al., 2018). We follow the final dataset split (see Section 2) using two training approaches: training on the found data, and the found and highly-ranked generated data. See the appendix for more details.

4.1 Unary models

The following models predict labels from *a single span* of text as input; this could be the ending only, the second sentence only, or the full passage.

a. fastText (Joulin et al., 2017): This library models a single span of text as a bag of n-grams, and tries to predict the probability of an ending being correct or incorrect independently.[7]

b. Pretrained sentence encoders We consider two types of pretrained RNN sentence encoders, SkipThoughts (Kiros et al., 2015) and InferSent (Conneau et al., 2017). SkipThoughts was trained by predicting adjacent sentences in book data, whereas InferSent was trained on supervised NLI data. For each second sentence (or just the ending), we feed the encoding into an MLP.

c. LSTM sentence encoder Given an arbitrary span of text, we run a two-layer BiLSTM over it. The final hidden states are then max-pooled to obtain a fixed-size representation, which is then used to predict the potential for that ending.

4.2 Binary models

The following models predict labels from *two spans* of text. We consider two possibilties for these models: using just the second sentence, where the two text spans are $\boldsymbol{n}, \boldsymbol{v}_i$, or using the context and the second sentence, in which case the spans are $\boldsymbol{s}, (\boldsymbol{n}, \boldsymbol{v}_i)$. The latter case includes many models developed for the NLI task.

d. Dual Bag-of-Words For this baseline, we treat each sentence as a bag-of-embeddings $(\mathbf{c}, \mathbf{v}_i)$. We model the probability of picking an ending i using a bilinear model: $\text{softmax}_i(\mathbf{c}\mathbf{W}\mathbf{v}_i^T)$.[8]

e. Dual pretrained sentence encoders Here, we obtain representations from SkipThoughts or InferSent for each span, and compute their pairwise compatibility using either 1) a bilinear model or 2) an MLP from their concatenated representations.

f. SNLI inference Here, we consider two models that do well on SNLI (Bowman et al., 2015): Decomposable Attention (Parikh et al., 2016) and ESIM (Chen et al., 2017). We use pretrained versions of these models (with ELMo embeddings) on SNLI to obtain 3-way entailment, neutral, and contradiction probabilities for each example. We then train a log-linear model using these 3-way probabilities as features.

g. SNLI models (retrained) Here, we train ESIM and Decomposable Attention on our dataset: we simply change the output layer size to 1 (the potential of an ending $\boldsymbol{v}_i$) with a softmax over i.

4.3 Other models

We also considered the following models:

h. Length: Although length was used by the adversarial classifier, we want to verify that human validation didn't reintroduce a length bias. For this baseline, we always choose the shortest ending.

i. ConceptNet As our task requires world knowledge, we tried a rule-based system on top of the

---

[7] The fastText model is trained using binary cross-entropy; at test time we extract the prediction by selecting the ending with the highest positive likelihood under the model.

[8] We also tried using an MLP, but got worse results.

			Ending only				2nd sentence only				Context+2nd sentence			
			found only		found+gen		found only		found+gen		found only		found+gen	
		Model	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
	misc	Random	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0
		Length	26.7	27.0	26.7	27.0								
		ConceptNet					26.0	26.0	26.0	26.0				
Unary models		fastText	27.5	26.9	29.9	29.0	29.2	27.8	29.8	29.0	29.4	28.0	30.3	29.8
	Sentence encoders	SkipThoughts	32.4	32.1	32.2	31.8	33.0	32.4	32.8	32.3				
		InferSent	30.6	30.2	32.0	31.9	33.2	32.0	34.0	32.6				
	LSTM sequence model	LSTM+GloVe	31.9	31.8	32.9	32.4	32.7	32.4	34.3	33.5	43.1	43.6	45.6	45.7
		LSTM+Numberbatch	32.4	32.6	32.3	31.9	31.9	31.9	34.1	32.8	39.9	40.2	41.2	40.5
		LSTM+ELMo	**43.6**	**42.9**	**43.3**	**42.3**	**47.4**	**46.7**	**46.3**	**46.0**	51.4	50.6	51.3	50.4
Binary models	DualBoW	DualBoW+GloVe					31.3	31.3	31.9	31.2	34.5	34.7	32.9	33.1
		DualBoW+Numberbatch					31.9	31.4	31.6	31.3	35.1	35.1	34.2	34.1
	Dual sentence encoders	SkipThoughts-MLP					34.6	33.9	36.2	35.5	33.4	32.3	37.4	36.4
		SkipThoughts-Bilinear					36.0	35.7	34.7	34.5	36.5	35.6	35.3	34.9
		InferSent-MLP					32.9	32.1	32.8	32.7	35.9	36.2	39.5	39.4
		InferSent-Bilinear					32.0	31.3	31.6	31.3	40.5	40.3	39.0	38.4
	SNLI inference	SNLI-ESIM									36.4	36.1	36.2	36.0
		SNLI-DecompAttn									35.8	35.8	35.8	35.7
	SNLI models (retrained)	DecompAttn+GloVe					29.8	30.3	31.1	31.7	47.4	47.6	48.5	48.6
		DecompAttn+Numberbatch					32.4	31.7	32.5	31.9	47.4	48.0	48.0	48.3
		DecompAttn+ELMo					43.4	43.4	40.6	40.3	47.7	47.3	46.0	45.4
		ESIM+GloVe					34.8	35.1	36.3	36.7	51.9	52.7	52.5	52.5
		ESIM+Numberbatch					33.1	32.6	33.0	32.4	46.5	46.4	44.0	44.6
		ESIM+ELMo					46.0	45.7	45.9	44.8	**59.1**	**59.2**	**58.7**	**58.5**
	Human	1 turker									82.8			
		3 turkers									85.1			
		5 turkers									**88.0**			
		Expert									85.0			

Table 3: Performance of all models in accuracy (%). All models substantially underperform humans, although performance increases as more context is provided (left to right). We optionally train on found endings only, or found and human-validated generated endings (found+gen).

ConceptNet knowledge base (Speer et al., 2017). For an ending sentence, we use the spaCy dependency parser to extract the head verb and its dependent object. The ending score is given by the number of ConceptNet causal relations[9] between synonyms of the verb and synonyms of the object.

j. Human performance To benchmark human performance, five Mechanical Turk workers were asked to answer 100 dataset questions, as did an 'expert' annotator (the first author of this paper). Predictions were combined using a majority vote.

4.4 Results

We present our results in Table 3. The best model that only uses the ending is the LSTM sequence model with ELMo embeddings, which obtains 43.6%. This model, as with most models studied, greatly improves with more context: by 3.1% when given the initial noun phrase, and by an additional 4% when also given the first sentence.

Further improvement is gained from models that compute pairwise representations of the inputs. While the simplest such model, DualBoW, obtains only 35.1% accuracy, combining InferSent sentence representations gives 40.5% accuracy (InferSent-Bilinear). The best results come from pairwise NLI models: when fully trained on SWAG, ESIM+ELMo obtains 59.2% accuracy.

When comparing machine results to human results, we see there exists a lot of headroom. Though there likely is some noise in the task, our results suggest that humans (even untrained) converge to a consensus. Our in-house "expert" annotator is outperformed by an ensemble of 5 Turk workers (with 88% accuracy); thus, the effective upper bound on our dataset is likely even higher.

5 Analysis

5.1 SWAG versus existing NLI datasets

The past few years have yielded great advances in NLI and representation learning, due to the availability of large datasets like SNLI and MultiNLI

---

[9] We used the relations 'Causes', 'CapableOf', 'ReceivesAction', 'UsedFor', and 'HasSubevent'. Though their coverage is low (30.4% of questions have an answer with ≥1 causal relation), the more frequent relations in ConceptNet, such as 'IsA', at best only indirectly relate to our task.

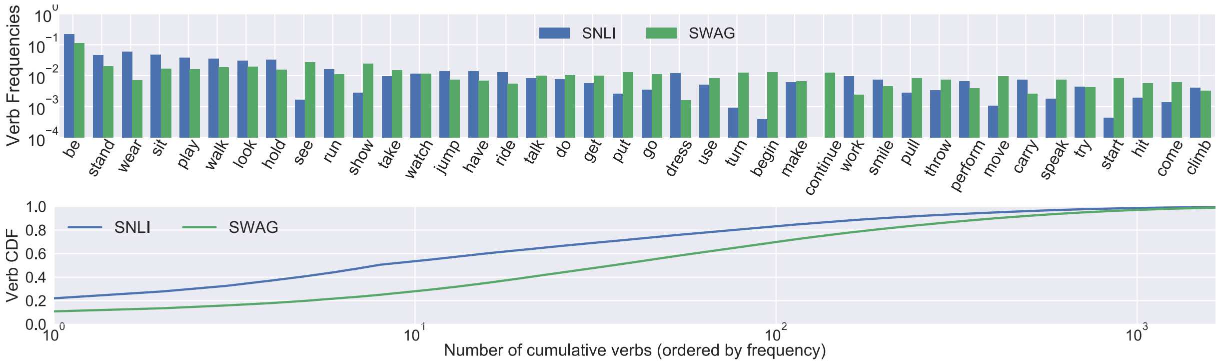

Figure 4: Top: Distribution of the 40 top verbs in the union of SNLI and **SWAG**. Our dataset shows a greater variety of dynamic verbs, such as "move", as well as temporal verbs such as "start" and "come." "Continue" is cut off for SNLI (it has frequency $6 \cdot 10^{-5}$). Bottom: CDF for verbs in SNLI and **SWAG**.

(Bowman et al., 2015; Williams et al., 2018). With the release of **SWAG**, we hope to continue this trend, particularly as our dataset largely has the same input/output format as other NLI datasets. We observe three key differences between our dataset and others in this space:

First, as noted in Section 1, **SWAG** requires a unique type of temporal reasoning. A state-of-the-art NLI model such as ESIM, when bottlenecked through the SNLI notion of entailment (SNLI-ESIM), only obtains 36.1% accuracy.[10] This implies that these datasets necessitate different (and complementary) forms of reasoning.

Second, our use of videos results in wide coverage of dynamic and temporal situations Compared with SNLI, with contexts from Flickr30K (Plummer et al., 2017) image captions, **SWAG** has more active verbs like 'pull' and 'hit,' and fewer static verbs like 'sit' and 'wear' (Figure 4).[11]

Third, our dataset suffers from few lexical biases. Whereas fastText, a bag of n-gram model, obtains 67.0% accuracy on SNLI versus a 34.3% baseline (Gururangan et al., 2018), fastText obtains only 29.0% accuracy on **SWAG**.[12]

5.2 Error analysis

We sought to quantify how human judgments differ from the best studied model, ESIM+ELMo. We randomly sampled 100 validation questions that ESIM+ELMo answered incorrectly, for each extracting both the gold ending and the model's preferred ending. We asked 5 Amazon Mechanical Turk workers to pick the better ending (of which they preferred the gold endings 94% of the time) and to select one (or more) multiple choice reasons explaining why the chosen answer was better.

Reason	Explanation	Freq.
Situational	The good ending is better *in context*.	53.7%
Plausibility	The bad ending is implausible *regardless of context*.	14.4%
Novelty	The bad ending seems redundant; it is entailed by the context.	1.8%
Weirdness	The bad ending is semantically or grammatically malformed, e.g. 'the man is getting out of the horse.'	18.1%
Ambiguous	Both endings seem equally likely.	12.0%

Table 4: Justifications for ranking the gold answer over a wrong answer chosen by ESIM+ELMo.

The options, and the frequencies, are outlined in Table 4. The most common reason for the turkers preferring the correct answer is situational (52.3% of the time), followed by weirdness (17.5%) and plausibility (14.4%). This suggests that ESIM+ELMo already does a good job at filtering out weird and implausible answers, with the main bottleneck being grounded physical understanding. The ambiguous percentage is also relatively low (12.0%), implying significant headroom.

5.3 Qualitative examples

Last, we show several qualitative examples in Table 5. Though models can do decently well by identifying complex alignment patterns between the two sentences (e.g. being "up a tree" implies that "tree" is the end phrase), the incorrect model predictions suggest this strategy is insuffi-

---

[10] The weights of SNLI-ESIM pick up primarily on entailment probability (0.59), as with neutral (0.46), while contradiction is negatively correlated (-.42).

[11] Video data has other language differences; notably, character names in LSMDC were replaced by 'someone'

[12] The most predictive individual words on SWAG are infrequent in number: 'dotted' with $P(+|\text{dotted}) = 77\%$ with 10.3 counts, and $P(-|\text{similar}) = 81\%$ with 16.3 counts. (Counts from negative endings were discounted 3x, as there are 3 times as many negative endings as positive endings).

A waiter brings a fork. The waiter **a) starts to step away. (74.76%)** b) adds spaghetti to the table. (21.57%) c) brings a bunch of pie to the food (2.67%) d) drinks from the mug in the bowl. (0.98%)	He is up a tree. Someone **a) stands underneath the tree. (97.44%)** b) is at a pool table holding a cup. (1.14%) c) grabs a flower from a paper. (0.96%) d) is eating some cereal. (0.45%)
An old man rides a small bumper car. Several people a) get in the parking lot. (76.58%) b) wait in the car. (15.28%) **c) get stuck with other bumper cars. (6.75%)** d) are running down the road. (1.39%)	He pours the raw egg batter into the pan. He a) drops the tiny pan onto a plate. (93.48%) **b) lifts the pan and moves it around to shuffle the eggs. (4.94%)** c) stirs the dough into a kite. (1.53%) d) swirls the stir under the adhesive. (0.05%)

Table 5: Example questions answered by the best model, ESIM+Elmo, sorted by model probability. Correct model predictions are in **blue**, incorrect model predictions are red. The right answers are **bolded**.

cient. For instance, answering "An old man rides a small bumper car" requires knowledge about *bumper cars* and how they differ from regular cars: bumper cars are tiny, don't drive on roads, and don't work in parking lots, eliminating the alternatives. However, this knowledge is difficult to extract from existing corpora: for instance, the ConceptNet entry for Bumper Car has only a single relation: bumper cars are a type of vehicle. Other questions require intuitive physical reasoning: e.g, for "he pours the raw egg batter into the pan," about what happens next in making an omelet.

5.4 Where to go next?

Our results suggest that SWAG is a challenging testbed for NLI models. However, the adversarial models used to filter the dataset are purely stylistic and focus on the second sentence; thus, subtle artifacts still likely remain in our dataset. These patterns are ostensibly picked up by the NLI models (particularly when using ELMo features), but the large gap between machine and human performance suggests that more is required to solve the dataset. As models are developed for commonsense inference, and more broadly as the field of NLP advances, we note that AF can be used again to create a more adversarial version of SWAG using better language models and AF models.

6 Related Work

Entailment NLI There has been a long history of NLI benchmarks focusing on linguistic entailment (Cooper et al., 1996; Dagan et al., 2006; Marelli et al., 2014; Bowman et al., 2015; Lai et al., 2017; Williams et al., 2018). Recent NLI datasets in particular have supported learning broadly-applicable sentence representations (Conneau et al., 2017); moreover, models trained on these datasets were used as components for performing better video captioning (Pasunuru and Bansal, 2017), summarization (Pasunuru and Bansal, 2018), and generation (Holtzman et al., 2018), confirming the importance of NLI research. The NLI task requires a variety of commonsense knowledge (LoBue and Yates, 2011), which our work complements. However, previous datasets for NLI have been challenged by unwanted annotation artifacts, (Gururangan et al., 2018; Poliak et al., 2018) or scale issues. Our work addresses these challenges by constructing a new NLI benchmark focused on grounded commonsense reasoning, and by introducing an adversarial filtering mechanism that substantially reduces known and easily detectable annotation artifacts.

Commonsense NLI Several datasets have been introduced to study NLI beyond linguistic entailment: for inferring likely causes and endings given a sentence (COPA; Roemmele et al., 2011), for choosing the most sensible ending to a short story (RocStories; Mostafazadeh et al., 2016; Sharma et al., 2018), and for predicting likelihood of a hypothesis by regressing to an ordinal label (JOCI; (Zhang et al., 2017)). These datasets are relatively small: 1k examples for COPA and 10k cloze examples for RocStories.[13] JOCI increases the scale by generating the hypotheses using a knowledge graph or a neural model. In contrast to JOCI where the task was formulated as a regression task on the degree of plausibility of the hypothesis, we frame commonsense inference as a multiple choice question to reduce the potential ambiguity in the labels and to allow for direct comparison between machines and humans. In addition, SWAG's use of adversarial filtering increases diversity of situations and counterfactual generation quality.

---

[13] For RocStories, this was by design to encourage learning from the larger corpus of 98k sensible stories.

Last, another related task formulation is sentence completion or cloze, where the task is to predict a single word that is removed from a given context (Zweig and Burges, 2011; Paperno et al., 2016).[14] Our work in contrast requires longer textual descriptions to reason about.

Vision datasets Several resources have been introduced to study temporal inference in vision. The Visual Madlibs dataset has 20k image captions about hypothetical next/previous events (Yu et al., 2015); similar to our work, the test portion is multiple-choice, with counterfactual answers retrieved from similar images and verified by humans. The question of 'what will happen next?' has also been studied in photo albums (Huang et al., 2016), videos of team sports, (Felsen et al., 2017) and egocentric dog videos (Ehsani et al., 2018). Last, annotation artifacts are also a recurring problem for vision datasets such as Visual Genome (Zellers et al., 2018) and Visual QA (Jabri et al., 2016); recent work was done to create a more challenging VQA dataset by annotating complementary image pairs (Goyal et al., 2016).

Reducing gender/racial bias Prior work has sought to reduce demographic biases in word embeddings (Zhang et al., 2018) as well as in image recognition models (Zhao et al., 2017). Our work has focused on producing a dataset with minimal annotation artifacts, which in turn helps to avoid some gender and racial biases that stem from elicitation (Rudinger et al., 2017). However, it is not perfect in this regard, particularly due to biases in movies (Schofield and Mehr, 2016; Sap et al., 2017). Our methodology could potentially be extended to construct datasets free of (possibly intersectional) gender or racial bias.

Physical knowledge Prior work has studied learning grounded knowledge about objects and verbs: from knowledge bases (Li et al., 2016), syntax parses (Forbes and Choi, 2017), word embeddings (Lucy and Gauthier, 2017), and images and dictionary definitions (Zellers and Choi, 2017). An alternate thread of work has been to learn scripts: high-level representations of event chains (Schank and Abelson, 1975; Chambers and Jurafsky, 2009). SWAG evaluates both of these strands.

7 Conclusion

We propose a new challenge of physically situated commonsense inference that broadens the scope of natural language inference (NLI) with commonsense reasoning. To support research toward commonsense NLI, we create a large-scale dataset SWAG with 113k multiple-choice questions. Our dataset is constructed using Adversarial Filtering (AF), a new paradigm for robust and cost-effective dataset construction that allows datasets to be constructed at scale while automatically reducing annotation artifacts that can be easily detected by a committee of strong baseline models. Our adversarial filtering paradigm is general, allowing potential applications to other datasets that require human composition of question answer pairs.

Acknowledgements

We thank the anonymous reviewers, members of the ARK and xlab at the University of Washington, researchers at the Allen Institute for AI, and Luke Zettlemoyer for their helpful feedback. We also thank the Mechanical Turk workers for doing a fantastic job with the human validation. This work was supported by the National Science Foundation Graduate Research Fellowship (DGE-1256082), the NSF grant (IIS-1524371, 1703166), the DARPA CwC program through ARO (W911NF-15-1-0543), the IARPA DIVA program through D17PC00343, and gifts by Google and Facebook. The views and conclusions contained herein are those of the authors and should not be interpreted as representing endorsements of IARPA, DOI/IBC, or the U.S. Government.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

Zheng Cai, Lifu Tu, and Kevin Gimpel. 2017. Pay attention to the ending: Strong neural baselines for the

---

[14] Prior work on sentence completion filtered negatives with heuristics based on LM perplexities. We initially tried something similar, but found the result to still be gameable.

roc story cloze task. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 616–622.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and Their Participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 602–610, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668.

Gennaro Chierchia and Sally McConnell-Ginet. 2000. *Meaning and Grammar (2Nd Ed.): An Introduction to Semantics*. MIT Press, Cambridge, MA, USA.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Robin Cooper, Dick Crouch, JV Eijckl, Chris Fox, JV Genabith, J Japars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. A framework for computational semantics (fracas). Technical report, Technical report, The FraCaS Consortium.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Kiana Ehsani, Hessam Bagherinezhad, Joseph Redmon, Roozbeh Mottaghi, and Ali Farhadi. 2018. Who let the dogs out? modeling dog behavior from visual data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Panna Felsen, Pulkit Agrawal, and Jitendra Malik. 2017. What will happen next? forecasting player moves in sports videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3342–3351.

Maxwell Forbes and Yejin Choi. 2017. Verb physics: Relative physical knowledge of actions and objects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 266–276.

JJ Gibson. 1979. The ecological approach to visual perception. *Houghton Mifflin Comp.*

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. *arXiv preprint arXiv:1612.00837*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112. Association for Computational Linguistics.

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970.

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

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649. Association for Computational Linguistics.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1233–1239.

Allan Jabri, Armand Joulin, and Laurens van der Maaten. 2016. Revisiting visual question answering baselines. In *European conference on computer vision*, pages 727–739. Springer.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 427–431.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-Captioning Events in Videos. In *International Conference on Computer Vision (ICCV)*.

Alice Lai, Yonatan Bisk, and Julia Hockenmaier. 2017. Natural language inference from multiple premises. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 100–109, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Peter LoBue and Alexander Yates. 2011. Types of common-sense knowledge needed for recognizing textual entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 329–334. Association for Computational Linguistics.

Li Lucy and Jon Gauthier. 2017. Are distributional representations ready for the real world? evaluating word vectors for grounded perceptual meaning. In *Proceedings of the First Workshop on Language Grounding for Robotics*, pages 76–85.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA). ACL Anthology Identifier: L14-1314.

Nasrin Mostafazadeh, Nathanael Chambers, Xiadong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *NAACL*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1273–1283, Vancouver, Canada. Association for Computational Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2017. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *Int. J. Comput. Vision*, 123(1):74–93.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis Only Baselines in Natural Language Inference. In *Joint Conference on Lexical and Computational Semantics (StarSem)*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele. 2017. Movie Description. *International Journal of Computer Vision*, 123(1):94–120.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the First ACL*

Workshop on Ethics in Natural Language Processing, pages 74–79.

Maarten Sap, Marcella Cindy Prasettio, Ari Holtzman, Hannah Rashkin, and Yejin Choi. 2017. Connotation frames of power and agency in modern films. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2329–2334.

Roger C. Schank and Robert P. Abelson. 1975. Scripts, plans, and knowledge. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'75, pages 151–157, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Alexandra Schofield and Leo Mehr. 2016. Gender-distinguishing features in film dialogue. In *Proceedings of the Fifth Workshop on Computational Linguistics for Literature*, pages 32–39.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Li Zilles, Yejin Choi, and Noah A. Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the ROC story cloze task. In *Proc. of CoNLL*.

Rishi Sharma, James Allen, Omid Bakhshandeh, and Nasrin Mostafazadeh. 2018. Tackling the story ending biases in the story cloze test. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 752–757.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*, pages 4444–4451.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 818–827.

Vladimir Vapnik. 2000. *The Nature of Statistical Learning Theory*, 2 edition. Information Science and Statistics. Springer-Verlag, New York.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Licheng Yu, Eunbyung Park, Alexander C. Berg, and Tamara L. Berg. 2015. Visual Madlibs: Fill in the blank Image Generation and Question Answering. *ICCV*.

Rowan Zellers and Yejin Choi. 2017. Zero-shot activity recognition with verb attribute induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. 2018. Neural motifs: Scene graph parsing with global context. In *Conference on Computer Vision and Pattern Recognition*.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Conference on Artificial Intelligence, Ethics and Society*.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal Common-sense Inference. *Transactions of the Association for Computational Linguistics*, 5:379–395.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2989.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*.

Geoffrey Zweig and Christopher JC Burges. 2011. The microsoft research sentence completion challenge. Technical report, Citeseer.

TwoWingOS: A Two-Wing Optimization Strategy for Evidential Claim Verification

Wenpeng Yin, Dan Roth
University of Pennsylvania
{wenpeng,danroth}@seas.upenn.edu

Abstract

Determining whether a given claim is supported by evidence is a fundamental NLP problem that is best modeled as Textual Entailment. However, given a large collection of text, finding evidence that could support or refute a given claim is a challenge in itself, amplified by the fact that different evidence might be needed to support or refute a claim. Nevertheless, most prior work decouples evidence identification from determining the truth value of the claim given the evidence.

We propose to consider these two aspects jointly. We develop TwoWingOS (**two-wing** **o**ptimization **s**trategy), a system that, while identifying appropriate evidence for a claim, also determines whether or not the claim is supported by the evidence. Given the claim, TwoWingOS attempts to identify a subset of the evidence candidates; given the predicted evidence, it then attempts to determine the truth value of the corresponding claim. We treat this challenge as coupled optimization problems, training a joint model for it. TwoWingOS offers two advantages: (i) Unlike pipeline systems, it facilitates flexible-size evidence set, and (ii) Joint training improves both the claim verification and the evidence identification. Experiments on a benchmark dataset show state-of-the-art performance.[1]

1 Introduction

A claim, e.g., "Marilyn Monroe worked with Warner Brothers", is an assertive sentence that may be true or false. While the task of claim verification *will not* tell us the absolute truth of this claim, it is expected to determine whether the claim is supported by evidence in a given text collection. Specifically, given a claim and a text corpus, *evidential claim verification*, demonstrated in Figure 1, aims at identifying text snippets in the corpus that act as evidence that supports or refutes the claim.

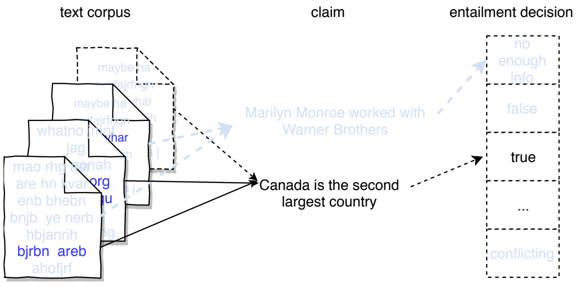

Figure 1: Illustration of evidential claim verification task. For a claim, we determine its truth value by evidence identified from a text corpus.

This problem has broad applications. For example, knowledge bases (KB), such as Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007), can be augmented with a new relational statement such as "(Afghanistan, is_source_of, Kushan Dynasty)". This needs to be first verified by a claim verification process and supported by evidence (Roth et al., 2009; Chaganty et al., 2017). More broadly, claim verification is a key component in any technical solution addressing recent concerns about the trustworthiness of online content (Vydiswaran et al., 2011; Pasternack and Roth, 2013; Hovy et al., 2013). In both scenarios, we care about whether or not a claim holds, and seek reliable evidence in support of this decision.

Evidential claim verification requires that we address three challenges. First, to locate text snippets in the given corpus that can potentially be used to determine the truth value of the given claim. This differs from the conventional textual entailment (TE) problem (Dagan et al., 2013) as here we first look for the premises given a hypothesis. Clearly, the evidence one seeks depends on the claim, as well as on the eventual entailment

---

[1] cogcomp.org/page/publication_view/847

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 105–114
Brussels, Belgium, October 31 - November 4, 2018.

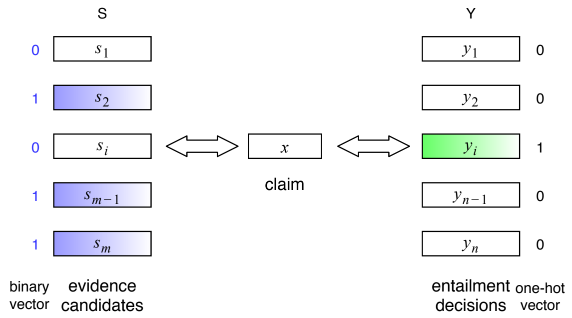

Figure 2: TwoWingOS, a generic two-wing optimization framework. A subset of the evidence candidates $S_e = \{s_1, \ldots, s_{m-1}, s_m\}$ is chosen via a binary vector (left), and an n-valued entailment decision $y_i \in Y$ is chosen (right), with respect to the claim x.

decision – the same claim would require different supporting than refuting evidence. This motivates us to develop an approach that can transfer knowledge from claim verification to evidence identification. Second, the evidence for a claim might require aggregating information from multiple sentences and even multiple documents (rf. #3 in Table 4). Therefore, a *set*, rather than a collection of independent text snippets, should be chosen to act as evidence. And, finally, in difference from TE, given a set of evidence sentences as a premise, the truth value of the claim should depend on *all of the evidence*, rather than on a single sentence there.

The discussion above suggests that claim verification and evidence identification are tightly coupled. Claim should influence the identification of appropriate evidence, and "trusted evidence boosts the claim's veracity" (Vydiswaran et al., 2011). Consequently, we propose TwoWingOS, a two-wing optimization strategy[2], to support this process. As shown in Figure 2, we consider a set of sentences S as the candidate evidence space, a claim x, and a decision space Y for the claim verification. In the optimal condition, a one-hot vector over Y indicates which decision to make towards the claim, and a binary vector over S indicates a subset of sentences S_e (in blue in Figure 2) to act as evidence.

Prior work mostly approached this problem as a pipeline procedure – first, given a claim x, determine S_e by some similarity matching; then, conduct textual entailment over (S_e, x) pairs. Our framework, TwoWingOS, optimizes the two subtasks jointly, so that both claim verification and evidence identification can enhance each other. TwoWingOS is a generic framework making use of a shared representation of the claim to co-train evidence identification and claim verification.

TwoWingOS is tested on the FEVER benchmark (Thorne et al., 2018), showing ≈30% F_1 improvement for evidence identification, and ≈23% accuracy increase in claim verification. Our analysis shows that (i) entity mentions in claims provide a strong clue for retrieving relevant passages; (ii) composition of evidence clues across sentences helps claim verification; and that (iii) the joint training scheme provides significant benefits of a pipeline architecture.

2 Related Work

Most work focuses on the dataset construction while lacking advanced models to handle the problem. Vlachos and Riedel (2014) propose and define the "fact checking" problem, without a concrete solution. Ferreira and Vlachos (2016) release the dataset "Emergent" for rumor debunking. Each claim is accompanied by an article headline as evidence. Then a three-way logistic regression model is used over some rule-based features. No need to search for evidence. Wang (2017) release a larger dataset for fake news detection, and propose a hybrid neural network to integrate the statement and the speaker's meta data to do classification. However, the presentation of evidences is ignored. Kobayashi et al. (2017) release a similar dataset to (Thorne et al., 2018), but they do not consider the evaluation of evidence reasoning.

Some work mainly pays attention to determining whether the claim is true or false, assuming evidence facts are provided or neglecting presenting evidence totally, e.g., (Angeli and Manning, 2014) – given a database of true facts as premises, predicting whether an unseen fact is true and should belong to the database by natural logic inference. Open-domain question answering (QA) against a text corpus (Yin et al., 2016; Chen et al., 2017; Wang et al., 2018) can also be treated as claim verification problem, if we treat (question, correct answer) as a claim. However, little work has studied how well a QA system can identify all the answer evidence.

Only a few works considered improving the evidence presentation in claim verification problems.

---

[2] By "two-wing optimization", we mean that the same object, i.e., the claim, is mapped into two target spaces in a joint optimization scheme.

Roth et al. (2009) introduce the task of Entailed Relation Recognition – given a set of short paragraphs and a relational fact in the triple form of (argument$_1$, relation, argument$_2$), finding the paragraphs that can entail this fact. They first use Expanded Lexical Retrieval to rank and keep the top-k paragraphs as candidates, then build a TE classifier over each (candidate, statement) pair. The work directly related to us is by Thorne et al. (2018). Given claims and a set of Wikipages, Thorne et al. (2018) use a retrieval model based on TF-IDF to locate top-5 sentences in top-5 pages as evidence, then utilize a neural entailment model to classify (evidence, claim) pairs.

In contrast, our work tries to optimize the claim verification as well as the evidence identification in a joint training scheme, which is more than just supporting or refuting the claims.

3 The TWOWINGOS Model

Figure 2 illustrates the two-wing optimization problem addressed in this work: given a collection of evidence candidates S=$\{s_1, s_2, \cdots, s_i, \cdots, s_m\}$, a claim x and a decision set $Y = \{y_1 \cdots, y_n\}$, the model TWOWINGOS predicts a binary vector p over S and a one-hot vector o over Y against the ground truth, a binary vector q and a one-hot vector z, respectively. A binary vector over S means a subset of sentences (S_e) act as evidence, and the one-hot vector indicates a single decision (y_i) to be made towards the claim x given the evidence S_e. Next, we use two separate subsections to elaborate the process of evidence identification (i.e., optimize p to q) and the claim verification (i.e., optimize o to z).

3.1 Evidence identification

A simple approach to identifying evidence is to detect the top-k sentences that are lexically similar to the claim, as some pipeline systems (Roth et al., 2009; Thorne et al., 2018) do. However, a claim-unaware fixed k is less optimal, adding noise or missing key supporting factors, consequently limiting the performance.

In this work, we approach the evidence by modeling sentences S=$\{s_1, \cdots, s_i, \cdots, s_m\}$ with the claim x as context in a supervised learning scheme. For each s_i, the problem turns out to be learning a probability: how likely s_i can entail the claim conditioned on other candidates as context, as shown by the blue items in Figure 2.

To start, a piece of text t ($t \in S \cup \{x\}$) is represented as a sequence of l hidden states, forming a feature map $\mathbf{T} \in \mathbb{R}^{d \times l}$, where d is the dimensionality of hidden states. We first stack a vanilla CNN (convolution & max-pooling) (LeCun et al., 1998) over $\mathbf{T}$ to get a representation for t. As a result, each evidence candidate s_i has a representation $\mathbf{s}_i$, and the claim x has a representation $\mathbf{x}$. To get a probability for each s_i, we need first to build its *claim-aware representation* $\mathbf{r}_i$.

Coarse-grained representation. We directly concatenate the representation of s_i and x, generated by the vanilla CNN, as:

$$\mathbf{r}_i = [\mathbf{s}_i, \mathbf{x}, \mathbf{s}_i \cdot \mathbf{x}^T] \tag{1}$$

This coarse-grained approach makes use of merely the sentence-level representations while neglecting more fine-grained interactions between the sentences and the claim.

Fine-grained representation. Instead of directly employing the sentence-level representations, here we explore claim-aware representations for each word in sentence s_i, then compose them as the sentence representation $\mathbf{r}_i$, inspired by the Attentive Convolution (Yin and Schütze, 2017).

For each word s_i^j in s_i, we first calculate its matching score towards each word x^z in x, by dot product over their hidden states. Then the representation of the claim, as the context for the word s_i^j, is formed as:

$$\mathbf{c}_i^j = \sum_z \mathrm{softmax}(\mathbf{s}_i^j \cdot (\mathbf{x}^z)^T) \cdot \mathbf{x}^z \tag{2}$$

Now, word s_i^j has left context $\mathbf{s}_i^{j-1}$, right context $\mathbf{s}_i^{j+1}$ in s_i, and the claim-aware context $\mathbf{c}_i^j$ from x. A convolution encoder generates its claim-aware representation $\mathbf{i}_i^j$:

$$\mathbf{i}_i^j = \tanh(\mathbf{W} \cdot [\mathbf{s}_i^{j-1}, \mathbf{s}_i^j, \mathbf{s}_i^{j+1}, \mathbf{c}_i^j] + \mathbf{b}) \tag{3}$$

where parameters $\mathbf{W} \in \mathbb{R}^{d \times 4d}$, $\mathbf{b} \in \mathbb{R}^d$.

To compose those claim-aware word representations as the representation for sentence s_i, we use a max-pooling over $\{\mathbf{i}_i^j\}$ along with j, generating $\mathbf{i}_i$.

We use term $f_{\mathrm{int}}(s_i, x)$ to denote this whole process, so that:

$$\mathbf{i}_i = f_{\mathrm{int}}(s_i, x) \tag{4}$$

At this point, the fine-grained representation for evidence candidate s_i is:

$$\mathbf{r}_i = [\mathbf{s}_i, \mathbf{x}, \mathbf{s}_i \cdot \mathbf{x}^T, \mathbf{i}_i] \tag{5}$$

Loss function. With a claim-aware representation $\mathbf{r}_i$, the sentence s_i subsequently gets a probability, acting as the evidence, $\alpha_i \in (0,1)$ via a non-linear sigmoid function:

$$\alpha_i = \mathrm{sigmoid}(\mathbf{v} \cdot \mathbf{r}_i^{\mathrm{T}}) \tag{6}$$

where parameter vector $\mathbf{v}$ has the same dimensionality as $\mathbf{r}_i$.

In the end, all evidence candidates in S have a ground-truth binary vector q and the predicted probability vector α; then loss l_{ev} ("ev": **ev**idence) is implemented as a binary cross-entropy:

$$l_{ev} = \sum_{i=1}^{m} -(q_i \log(\alpha_i) + (1-q_i)\log(1-\alpha_i)) \tag{7}$$

As the output of this evidence identification module, we binarize the probability vector α by $p_i = [\alpha_i > 0.5]$ ("$[x]$" is 1 if x is true or 0 otherwise). p_i indicates s_i is evidence or not. All $\{s_i\}$ with $p_i = 1$ act as evidence set S_e.

3.2 Claim verification

As shown in Figure 2, to figure out an entailment decision y_i for the claim x, the evidence S_e possibly consists of more than one sentence. Furthermore, those evidence sentences are not necessarily in textual order nor from the same passage. So, we need a mechanism that enables each evidence or even each word inside to be aware of the content from other evidence sentences. Similar to the aforementioned approach to evidence identification, we come up with three methods, with different representation granularity, to learn a representation for (S_e, x), i.e., the input for claim verification, shown in Figure 3.

Coarse-grained representation. In this case, we treat S_e as a whole, constructing its representation $\mathbf{e}$ by summing up the representations of all sentences in S_e in a weighted way:

$$\mathbf{e} = \sum_{i=1}^{m} \alpha_i \cdot p_i \cdot \mathbf{s}_i \tag{8}$$

where α_i, from Equation 6, is the probability of s_i being the evidence.

Then the (S_e, x) pair gets a coarse-grained concatenated representation: $[\mathbf{e}, \mathbf{x}]$. It does not model the interactions within the evidence nor the interactions between the evidence and the claim. Based on our experience in evidence identification module, the representation of a sentence is better learned by composing context-aware word-level representations. Next, we introduce how to learn fine-grained representation for the (S_e, x) pair.

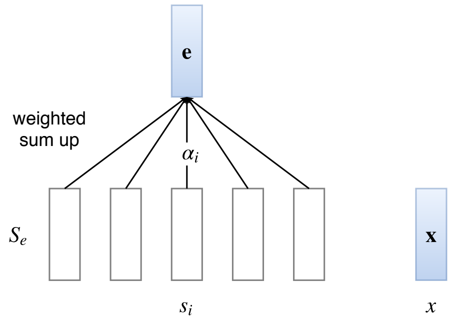

(a) Coarse-grained representations

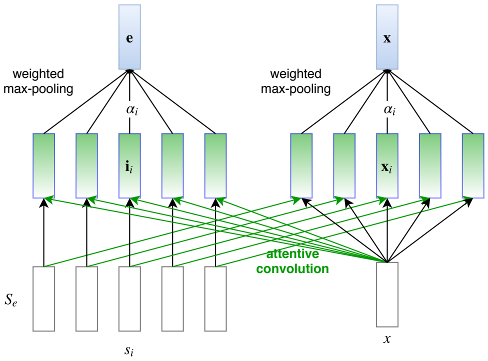

(b) Single-channel fine-grained representations

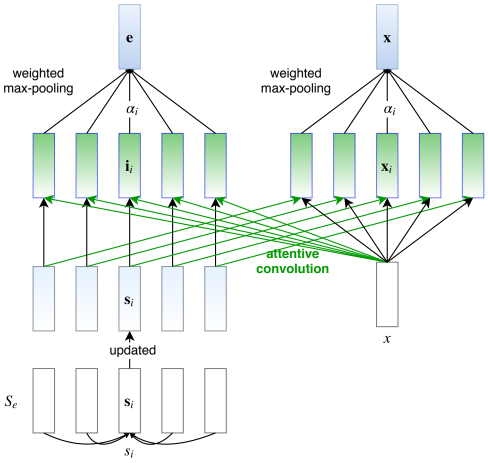

(c) Two-channel fine-grained representations

Figure 3: Three representation learning methods in claim verification. Green arrows act as context in attentive convolution.

Single-channel fine-grained representation. By "single-channel," we mean each sentence s_i is aware of the claim x as its single context.

For a single pair (s_i, x), we utilize the function $f_{\text{int}}()$ in Equation 4 to build the fine-grained representations for both s_i and x, obtaining $\mathbf{i}_i =$

$f_{\text{int}}(s_i, x)$ for s_i and $\mathbf{x}_i = f_{\text{int}}(x, s_i)$ for x.

For (S_e, x), we compose all the $\{\mathbf{i}_i\}$ and all the $\{\mathbf{x}_i\}$ along with i, via a weighted max-pooling:

$$\mathbf{e} = \text{maxpool}_i(\alpha_i \cdot p_i \cdot \mathbf{i}_i) \tag{9}$$

$$\mathbf{x} = \text{maxpool}_i(\alpha_i \cdot p_i \cdot \mathbf{x}_i) \tag{10}$$

This weighted max-pooling ensures that the sentences with higher probabilities of being evidence have a higher chance to present their features. As a result, (S_e, x) gets a concatenated representation: [**e**, **x**]

Two-channel fine-grained representation. By "two-channel," we mean that each evidence s_i is aware of two kinds of context, one from the claim x, the other from the remaining evidences.

Our first step is to accumulate evidence clues within S_e. To start, we concatenate all sentences in S_e as a fake long sentence $\hat{S}$ consisting of hidden states $\{\hat{\mathbf{s}}\}$. Similar to Equation 2, for each word s_i^j in sentence s_i, we *accumulate all of its related clues* ($\mathbf{c}_i^j$) from $\hat{S}$ as follows:

$$\mathbf{c}_i^j = \sum_z \text{softmax}(\mathbf{s}_i^j \cdot (\hat{\mathbf{s}}^z)^T) \cdot \hat{\mathbf{s}}^z \tag{11}$$

Then we update $\mathbf{s}_i^j$, the representation of word s_i^j, by element-wise addition:

$$\mathbf{s}_i^j = \mathbf{s}_i^j \oplus \mathbf{c}_i^j \tag{12}$$

This step enables the word s_i^j to "see" all related clues from S_e. The reason we add $\mathbf{s}_i^j$ and $\mathbf{c}_i^j$ is motivated by a simple experience: Assume the claim "Lily lives in the biggest city in Canada", and one sentence contains a clue "$\cdots$ Lily lives in Toronto $\cdots$" and another sentence contains a clue "$\cdots$ Toronto is Canada's largest city$\cdots$". The most simple yet effective approach to aggregating the two clues is to sum up their representation vectors (Blacoe and Lapata, 2012) (we do not concatenate them, as those clues have no consistent textual order across different $\mathbf{s}_i^j$).

After updating the representation of each word in s_i, we perform the aforementioned "single-channel fine-grained representation" between the updated s_i and the claim x, generating [**e**, **x**].

Loss function. For the claim verification input (S_e, x), we forward its representation [**e**, **x**] to a logistic regression layer in order to infer a probability distribution o over the label space Y:

	#SUPPORTED	#REFUTED	#NEI
train	80,035	29,775	35,639
dev	3,333	3,333	3,333
test	3,333	3,333	3,333

Table 1: Statistics of claims in FEVER dataset

$$\mathbf{o} = \text{softmax}(\mathbf{W} \cdot [\mathbf{e}, \mathbf{x}] + \mathbf{b}) \tag{13}$$

where $\mathbf{W} \in \mathbb{R}^{n \times 2d}$, $\mathbf{b} \in \mathbb{R}^n$

The loss l_{cv} ("cv": claim verification) is implemented as negative log-likelihood:

$$l_{\text{cv}} = -\log(\mathbf{o} \cdot \mathbf{z}^T) \tag{14}$$

where **z** is the ground truth one-hot label vector for the claim x on the space Y.

3.3 Joint optimization

Given the loss l_{ev} in evidence identification and the loss l_{cv} in claim verification, the overall training loss is represented by:

$$l = l_{\text{ev}} + l_{\text{cv}} \tag{15}$$

To ensure that we jointly train the two coupled subtasks with intensive knowledge communication instead of simply putting two pipeline neural networks together, our TWOWINGOS has following configurations:

- Both subsystems share the same set of word embeddings as parameters; the vanilla CNNs for learning sentence and claim representations share parameters as well.
- The output binary vector p by the evidence identification module is forwarded to the module of claim verification, as shown in Equations 8-10.
- Though the representation of a claim's decision y_i is not put explicitly into the module of evidence identification, the claim's representation **x** will be fine-tuned by the y_i, so that the evidence candidates can get adjustment from the decision y_i, since the claims are shared by two modules.

4 Experiments

4.1 Setup

Dataset. In this work, we use FEVER (Thorne et al., 2018). The claims in FEVER were generated from the introductory parts of about 50K

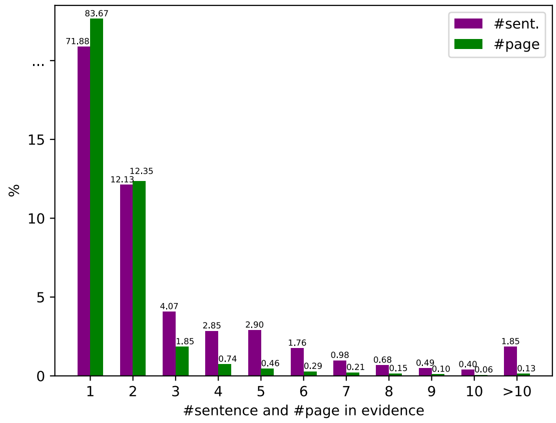

Figure 4: Distribution of #sentence and #pages in FEVER evidence

Wikipedia pages of a June 2017 dump. Annotators construct claims about a single fact of the title entity with arbitrarily complex expressions and entity forms. To increase the claim complexity so that claims would not be trivially verified, annotators adopt two routes: (i) Providing additional knowledge: Annotators can explore a dictionary of terms that were (hyper-)linked, along with their pages; (ii) Mutate claims in six ways: negation, paraphrasing, substitution of a relation/entity with a similar/dissimilar one, and making the claims more general/specific. All resulting claims have 9.4 tokens in average. Apart from claims, FEVER also provides a Wikipedia corpus in size of about 5.4 million.

Each claim is labeled as SUPPORTED, REFUTED or NOTENOUGHINFO (NEI). In addition, evidence sentences, from any wiki page, are required to be provided for SUPPORTED and REFUTED. Table 1 lists the data statistics. Figure 4 shows the distributions of sentence sizes and page sizes in FEVER's evidence set. We can see that roughly 28% of the evidence covers more than one sentence, and approximately 16.3% of the evidence covers more than one wiki page.

This task has three evaluations: (i) NOSCOREEV – accuracy of claim verification, neglecting the validity of evidence; (ii) SCOREEV – accuracy of claim verification with a requirement that the predicted evidence fully covers the gold evidence for SUPPORTED and REFUTED; (iii) F_1 – between the predicted evidence sentences and the ones chosen by annotators. We use the officially released evaluation scorer [3].

Wiki page retrieval[4]**.** For each claim, we search in the given dictionary of wiki pages in the form of {title: sentence_list}, and keep the top-5 ranked pages for fair comparison with Thorne et al. (2018). Algorithm 1 briefly shows the steps of wiki page retrieval. To speed up, we first build an inverted index from words to titles, then for each claim, we only search in the titles that cover at least one claim word.

```
Input: A claim, wiki={title: page_vocab}
Output: A ranked top-$k$ wiki titles
Generate entity_mentions from the claim;
while each title do
    if claim.vocab∩title.vocab is empty then
        discard this title
    else
        title_score = the max recall value of title.vocab
          in claim and in entity_mentions of the claim;
        if title_score = 1.0 then
            title.score = title_score
        else
            page_score = recall of claim in
              page_vocab;
            title.score = title_score + page_score
        end
    end
end
Sort titles by title.score in descending order
```

Algorithm 1: Algorithm description of wiki page retrieval for FEVER claims.

All sentences of the top-5 retrieved wiki pages are kept as evidence candidates for claims in train, dev and test. It is worth mentioning that this page retrieval step is a reasonable preprocessing which controls the complexity of evidence searching in real-world, such as the big space – 5.4 million – in this work.

Training setup. All words are initialized by 300D Word2Vec (Mikolov et al., 2013) embeddings, and are fine-tuned during training. The whole system is trained by AdaGrad (Duchi et al., 2011). Other hyperparameter values include: learning rate 0.02, hidden size 300, mini-batch size 50, filter width 3.

Baselines. In this work, we first consider the two systems reported by Thorne et al. (2018): (i) MLP: A multi-layer perceptron with one hidden layer, based on TF-IDF cosine similarity between the claim and the evidence (all evidence sentences are concatenated as a longer text piece) (Riedel et al., 2017); (ii) Decomp-Att (Parikh et al., 2016): A decomposable attention model that develops atten-

---

[3] https://github.com/sheffieldnlp/fever-scorer

[4] Our retrieval results are released as well.

k	(Thorne et al., 2018)		ours	
	rate	acc_ceiling	rate	acc_ceiling
1	25.31	50.21	76.58	84.38
5	55.30	70.20	89.63	93.08
10	65.86	77.24	91.19	94.12
25	75.92	83.95	92.81	95.20
50	82.49	90.13	93.36	95.57
100	86.59	91.06	94.19	96.12

Table 2: Wikipage retrieval evaluation on dev. "rate": claim proportion, e.g., $x\%$, if its gold passages are fully retrieved (for "SUPPORT" and "REFUTE" only); "acc_ceiling": $\frac{x\%\cdot(\#\mathrm{S}+\#\mathrm{R})+\#\mathrm{N}}{\#\mathrm{S}+\#\mathrm{R}+\#\mathrm{N}}$, the upper bound of accuracy for three classes if the coverage $x\%$ satisfies.

tion mechanisms to decompose the problem into subproblems to solve in parallel. Note that both systems first employed an IR system to keep top-5 relevant sentences from the retrieved top-5 wiki pages as *static evidence* for claims.

We further consider the following variants of our own system TwoWingOS:

- *Coarse-coarse*: Both evidence identification and claim verification adopt coarse-grained representations.

To further study our system, we test this "coarse-coarse" in three setups: (i) "pipeline" – train the two modules independently. Forward the predicted evidence to do entailment for claims; (ii) "diff-CNN" – joint training with separate CNN parameters to learn sentence/claim representations; (iii) "share-CNN" – joint training with shared CNN parameters.

The following variants are in joint training.

- *Fine&sentence-wise*: Given the evidence with multiple sentences, a natural baseline is to do entailment reasoning for each (sentence, claim), then compose. We do entailment reasoning between each predicted evidence sentence and the claim, generating a probability distribution over the label space Y. Then we sum up all the distribution vectors element-wise, as an ensemble system, to predict the label;

- Four combinations of different grained representation learning: "*coarse&fine(single)*", "*coarse&fine(two)*", "*fine&coarse*" and "*fine&fine(two)*". "Single" and "two" refer to the single/two-channel cases respectively.

4.2 Results

Performance of passage retrieval. Table 2 compares our wikipage retriever with the one in (Thorne et al., 2018), which used a document retriever[5] from DrQA (Chen et al., 2017).

Our document retrieval module surpasses the competitor by a big margin in terms of the coverage of gold passages: 89.63% vs. 55.30% ($k = 5$ in all experiments). Its powerfulness should be attributed to: (i) Entity mention detection in the claims. (ii) As wiki titles are entities, we have a bi-channel way to match the claim with the wiki page: one with the title, the other with the page body, as shown in Algorithm 1.

Performance on FEVER Table 3 lists the performances of baselines and the TwoWingOS variants on FEVER (dev&test). From the *dev* block, we observe that:

- TwoWingOS (from "share-CNN") surpasses prior systems in big margins. Overall, fine-grained schemes in each subtask contribute more than the coarse-grained counterparts;

- In the three setups – "pipeline", "diff-CNN" and "share-CNN" – of *coarse-coarse*, "pipeline" gets better scores than (Thorne et al., 2018) in terms of evidence identification. "Share-CNN" has comparable F_1 as "diff-CNN" while gaining a lot on NoScoreEv (72.32 vs. 39.22) and ScoreEv (50.12 vs. 21.04). This clearly shows that the claim verification gains much knowledge transferred from the evidence identification module. Both "diff-CNN" and "share-CNN" perform better than "pipeline" (except for the slight inferiority at ScoreEv: 21.04 vs. 22.26).

- Two-channel fine-grained representations show more effective than the single-channel counterpart in claim verification (NoScoreEv: 78.77 vs. 75.65, ScoreEv: 53.64 vs. 52.65). As we expected, evidence sentences should *collaborate* in inferring the truth value of the claims. Two-channel setup enables an evidence candidate aware of other candidates as well as the claim.

- In the last three rows of *dev*, there is no clear difference among their evidence identification scores. Recall that "sent-wise" is essentially an ensemble system over each (sentence, claim) entailment result. "Coarse-grained", instead, first sums up all sentence representation, then performs ($\sum(\mathit{sentence})$, claim) reasoning. We can also treat this "sum up" as an ensemble. Their comparison shows that *these two kinds of tricks do not*

---

[5] It compares passages and claims as TF-IDF weighted bag-of-bigrams.

		system	claim verification		evidence identification		
			NoScoreEv	ScoreEv	recall	precision	F_1
dev		MLP	41.86	19.04	44.22	10.44	16.89
		Decomp-Att	52.09	32.57	44.22	10.44	16.89
	TwoWingOS	coarse&coarse					
		pipeline	35.72	22.26	53.75	29.42	33.80
		diff-CNN	39.22	21.04	46.88	43.01	44.86
		share-CNN	72.32	50.12	45.55	40.77	43.03
		coarse&fine(single)	75.65	52.65	45.81	42.53	44.11
		coarse&fine(two)	78.77	53.64	45.78	39.23	42.25
		fine&sent-wise	71.02	53.43	52.70	**48.31**	50.40
		fine&coarse	71.48	53.17	52.75	47.30	49.87
		fine&fine(two)	**78.90**	**56.16**	**53.81**	47.73	**50.59**
test		(Thorne et al., 2018)	50.91	31.87	45.89	10.79	17.47
		TwoWingOS	**75.99**	**54.33**	**49.91**	**44.68**	**47.15**

Table 3: Performance on *dev* and *test* of FEVER. TwoWingOS outperforms prior systems if vanilla CNN parameters are shared by evidence identification and claim verification subsystems. It gains more if fine-grained representations are adopted in both subtasks.

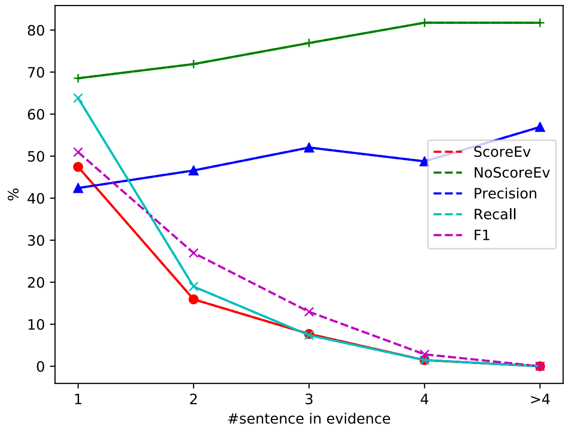

Figure 5: Performance vs. #sentence in evidence. Our system has robust precisions. The overall performance NoScoreEv is not influenced by the decreasing recall; this verifies the fact that the truth value of most claims can be determined by a single identified evidence sentence.

make much difference.

If we adopt "two-channel fine-grained representation" in claim verification, big improvements are observed in both NoScoreEv (+7.42%) and ScoreEv (+3%).

In the *test* block, our system (fine&fine(two)) beats the prior top system across all measurements by big margins – F_1: 47.15 vs. 17.47; ScoreEv: 54.33 vs. 31.87; NoScoreEv: 75.99 vs. 50.91.

In both *dev* and *test* blocks, we can observe that our evidence identification module consistently obtains balanced recall and precision. In contrast, the pipeline system by Thorne et al. (2018) has much higher recall than precision (45.89 vs. 10.79). It is worth mentioning that the ScoreEv metric is highly influenced by the recall value, since ScoreEv is computed on the claim instances whose evidences are fully retrieved, regardless of the precision. So, ideally, a system can set all sentences as evidence, so that ScoreEv can be promoted to be equal to NoScoreEv. Our system is more reliable in this perspective.

Performance vs. #sent. in evidence. Figure 5 shows the results of the five evaluation measures against different sizes of gold evidence sentences in test set. We observe that: (i) Our system has robust precisions across #sentence; however, the recall decreases. This is not that surprising, since the more ground-truth sentences in evidence, the harder it is to retrieve all of them; (ii) Due to the decrease in recall, the ScoreEv also gets influenced for bigger #sentence. Interestingly, high precision and worse recall in evidence with more sentences still make consistently strong overall performance, i.e., NoScoreEv. This should be due to the fact that the majority (83.18% (Thorne et al., 2018)) of claims can be correctly entailed by a single ground truth sentence, even if any remaining ground truth sentences are unavailable.

Error analysis. The case #1 in Table 4 shows that our system identifies two pieces of evidence

#	G/P	claim	gold evidence	predicted evidence
1	0/1	Telemundo is an English-language television network.	(Telemundo, 0) (Telemundo, 1) (Telemundo, 4) (Telemundo, 5) (Hispanic_and_Latino_Americans, 0)	(Telemundo, 0) (Telemundo, 4) (Fourth_television_network, 0) (Fourth_television_network, 4)
2	1/2	Home for the Holidays stars a famous American actor.	(Anne_Bancroft, 0) (Charles_Durning, 0) (Holly_Hunter, 0) (Home_for_the_Holidays_(1995_film), 5)	$\emptyset$
3	0/2	Both hosts of Weekly Idol were born in 1983.	(Weekly_Idol, 0) (Weekly_Idol, 1) (Defconn, 0)	(Weekly_Idol, 1)

Table 4: Error cases of TWOWINGOS in FEVER. "G/P": gold/predicted label ("0": refute; "1": support; "2": not enough information). To save space, we use "(title, i)" to denote the i^{th} sentence in the corresponding wiki page.

(i.e., (Telemundo, 0) and (Telemundo, 4)) correctly; however, it falsely predicts the claim label. (Telemundo, 0): *Telemundo is an American Spanish-language terrestrial television* $\cdots$. We can easily find that the keyword "Spanish-language" should refute the claim. However, both "Spanish-language" in this evidence and the "English-language" in the claim are unknown tokens with randomly initialized embeddings. This hints that a more careful data preprocessing may be helpful. In addition, to refute the claim, another clue comes from the combination of (Telemundo, 4) and (Hispanic_and_Latino_Americans, 0). (Telemundo, 4): "*The channel* $\cdots$ *aimed at Hispanic and Latino American audiences*"; (Hispanic_and_Latino_Americans, 0): "*Hispanic Americans and Latino Americans* $\cdots$ *are descendants of people from countries of Latin America and Spain.*". Our system only retrieved (Telemundo, 4). And this clue is hard to grasp as it requires some background knowledge – people from Latin America and Spain usually are not treated as English-speaking.

In the case #2, our system fails to identify any evidence. This is due to the failure of our passage retrieval module: it detects entity mentions "Home", "Holidays" and "American", and the top-5 retrieved passages are "Home", "Home_for_the_Holidays", "American_Home", "American" and "Home_for_the_Holidays_(song)", which unfortunately cover none of the four ground truth passages. Interestingly, (i) given the falsely retrieved passages, our system predicts "no sentence is valid evidence" (denoted as $\emptyset$ in Table 4); (ii) given the empty evidence, our system predicts "NoEnoughInfo" for this claim. Both make sense.

In the case #3, a successful classification of the claim requires information aggregation over the three gold evidence sentences: (Weekly_Idol, 0): "*Weekly Idol is a South Korean variety show* $\cdots$"; (Weekly_Idol, 1): "*The show is hosted by comedian Jeong Hyeong-don and rapper Defconn.*"; (Defconn, 0): "*Defconn (born Yoo Dae-joon; January 6 , 1977) is a* $\cdots$". To successfully retrieve the three sentences as a whole set of evidence is challenging in evidence identification. Additionally, this example relies on the recognition and matching of digital numbers (1983 vs. 1977), which is beyond the expressivity of word embeddings, and is expected to be handled by rules more easily.

5 Summary

In this work, we build TWOWINGOS, a two-wing optimization framework to address the claim verification problem by presenting precise evidence. Differing from a pipeline system, TWOWINGOS ensures the evidence identification module and the claim verification module are trained jointly, in an end-to-end scheme. Experiments show the superiority of TWOWINGOS in the FEVER benchmark.

Acknowledgments

We thank group colleagues (Nitish Gupta and Jennifer Sheffield) and Dr. Mo Yu from IBM AI Foundations Lab for providing insightful comments and critiques. This work was supported by Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Gabor Angeli and Christopher D. Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of EMNLP*, pages 534–545.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP-CoNLL*, pages 546–556.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Arun Tejasvi Chaganty, Ashwin Paranjape, Percy Liang, and Christopher D. Manning. 2017. Importance sampling for unbiased on-demand evaluation of knowledge base population. In *Proceedings of EMNLP*, pages 1038–1048.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of ACL*, pages 1870–1879.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzoto. 2013. Recognizing textual entailment: Models and applications.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of NAACL*, pages 1163–1168.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard H. Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of NAACL*, pages 1120–1130.

Mio Kobayashi, Ai Ishii, Chikara Hoshino, Hiroshi Miyashita, and Takuya Matsuzaki. 2017. Automated historical fact-checking by passage retrieval, word statistics, and virtual question-answering. In *Proceedings of IJCNLP*, pages 967–975.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of EMNLP*, pages 2249–2255.

Jeff Pasternack and Dan Roth. 2013. Latent credibility analysis. In *Proceedings of WWW*, pages 1009–1020.

Benjamin Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *CoRR*, abs/1707.03264.

Dan Roth, Mark Sammons, and V. G. Vinod Vydiswaran. 2009. A framework for entailed relation recognition. In *Proceedings of ACL*, pages 57–60.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*, pages 697–706.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *Proceedings of NAACL*.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the Workshop on Language Technologies and Computational Social Science@ACL*, pages 18–22.

V. G. Vinod Vydiswaran, ChengXiang Zhai, and Dan Roth. 2011. Content-driven trust propagation framework. In *Proceedings of SIGKDD*, pages 974–982.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R^3: Reinforced ranker-reader for open-domain question answering. In *Proceedings of AAAI*.

William Yang Wang. 2017. "Liar, Liar Pants on Fire": A new benchmark dataset for fake news detection. In *Proceedings of ACL*, pages 422–426.

Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. 2016. Attention-based convolutional neural network for machine comprehension. In *Proceedings of the NAACL Workshop on Human-Computer Question Answering*, pages 15–21.

Wenpeng Yin and Hinrich Schütze. 2017. Attentive convolution. *CoRR*, abs/1710.00519.

Associative Multichannel Autoencoder for Multimodal Word Representation

Shaonan Wang[1,2], **Jiajun Zhang**[1,2], **Chengqing Zong**[1,2,3]
[1] National Laboratory of Pattern Recognition, CASIA, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
`{shaonan.wang,jjzhang,cqzong}@nlpr.ia.ac.cn`

Abstract

In this paper we address the problem of learning multimodal word representations by integrating textual, visual and auditory inputs. Inspired by the re-constructive and associative nature of human memory, we propose a novel *associative multichannel autoencoder* (AMA). Our model first learns the associations between textual and perceptual modalities, so as to predict the missing perceptual information of concepts. Then the textual and predicted perceptual representations are fused through reconstructing their original and associated embeddings. Using a gating mechanism our model assigns different weights to each modality according to the different concepts. Results on six benchmark concept similarity tests show that the proposed method significantly outperforms strong unimodal baselines and state-of-the-art multimodal models.

1 Introduction

Representing the meaning of a word is a prerequisite to solve many linguistic and non-linguistic problems, such as retrieving words with the same meaning, finding the most relevant images or sounds of a word and so on. In recent years we have seen a surge of interest in building computational models that represent word meanings from patterns of word co-occurrence in corpora (Turney and Pantel, 2010; Mikolov et al., 2013; Pennington et al., 2014; Clark, 2015; Wang et al., 2018b). However, word meaning is also tied to the physical world. Many behavioral studies suggest that human semantic representation is grounded in the external environment and sensorimotor experience (Landau et al., 1998; Barsalou, 2008). This has led to the development of multimodal representation models that utilize both textual and perceptual information (e.g., images, sounds).

As evidenced by a range of evaluations (Andrews et al., 2009; Bruni et al., 2014; Silberer et al., 2016), multimodal models can learn better semantic word representations (a.k.a. embeddings) than text-based models. However, most existing models still have a number of drawbacks. First, they ignore the associations between modalities, and thus lack the ability of information transferring between modalities. Consequently they cannot handle words without perceptual information. Second, they integrate textual and perceptual representations with simple concatenation, which is insufficient to effectively fuse information from various modalities. Third, they typically treat the representations from different modalities equally. This is inconsistent with many psychological findings that information from different modalities contributes differently to the meaning of words (Paivio, 1990; Anderson et al., 2017).

In this work, we introduce the associative multichannel autoencoder (AMA), a novel multimodal word representation model that addresses all the above issues. Our model is built upon the stacked autoencoder (Bengio et al., 2007) to learn semantic representations by integrating textual and perceptual inputs. Inspired by the re-constructive and associative nature of human memory, we propose two associative memory modules as extensions. One is to learn associations between modalities (e.g., associations between textual and visual features), so as to reconstruct corresponding perceptual information of concepts. The other is to learn associations between related concepts, by reconstructing embeddings of both target words and their associated words. Furthermore, we propose a gating mechanism to learn the importance weights of different modalities to each word.

To summarize, our main contributions in this work are two-fold:

- We present a novel associative multichannel autoencoder for multimodal word representation, which is capable of utilizing associations between different modalities and related

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 115–124
Brussels, Belgium, October 31 - November 4, 2018.

concepts, and assigning different importance weights to each modality according to different words. Results on six standard benchmarks demonstrate that our methods outperform strong unimodal baselines and state-of-the-art multimodal models.

- Our model successfully integrates cognitive insights of the re-constructive and associative nature of semantic memory in humans, suggesting that rich information contained in human cognitive processing can be used to enhance NLP models. Furthermore, our results shed light on the fundamental questions of how to learn semantic representations, such as the plausibility of reconstructing perceptual information, associating related concepts and grounding word symbols to external environment.

2 Background and Related Work

2.1 Cognitive Grounding

A large body of research evidences that human semantic memory is inherently re-constructive and associative (Collins and Loftus, 1975; Anderson and Bower, 2014). That is, memories are not exact static copies of reality, but are rather reconstructed from their stimuli and associated concepts each time they are retrieved. For example, when we see a *dog*, not only the concept itself, but also the corresponding perceptual information and associated words will be jointly activated and reconstructed. Moreover, various theories state that the different sources of information contribute differently to the semantic representation of a concept (Wang et al., 2010; Ralph et al., 2017). For instance, Dual Coding Theory (Hiscock, 1974) posits that concrete words are represented in the brain in terms of a perceptual and linguistic code, whereas abstract words are encoded only in the linguistic modality.

In these respects, our method employs a retrieval and representation process analogous to that of humans, in which the retrieval of perceptual information and associated words is triggered and mediated by a linguistic input. The learned cross-modality mapping and reconstruction of associated words are inspired by the human mental model of associations between different modalities and related concepts. Moreover, word meaning is tied to both linguistic and physical environment, and relies differently on each modality inputs (Wang et al., 2018a). These are also captured by our multimodal representation model.

2.2 Multimodal Models

The existing multimodal representation models can be generally classified into two groups: 1) *Jointly training models* build multimodal representations with raw inputs of textual and perceptual resources. 2) *Separate training models* independently learn textual and perceptual representations and integrate them afterwards.

2.2.1 Jointly training models

A class of models extends Latent Dirichlet Allocation (Blei et al., 2003) to jointly learn topic distributions from words and perceptual units (Andrews et al., 2009; Silberer and Lapata, 2012; Roller and Schulte im Walde, 2013). Recently introduced work is an extension of the Skip-gram model (Mikolov et al., 2013). For instance, Hill and Korhonen (2014) propose a corpus fusion method that inserts the perceptual features of concepts in the training corpus, which is then used to train the Skip-gram model. Lazaridou et al. (2015) propose MMSkip model, which injects visual information in the process of learning textual representations by adding a max-margin objective to minimize the distance between textual and visual vectors. Kiela and Clark (2015) adopt the MMSkip to learn multimodal vectors with auditory perceptual inputs.

These methods can implicitly propagate perceptual information to word representations and at the same time learn multimodal representations. However, they utilize raw text corpus in which words having perceptual information account for a small portion. This weakens the effect of introducing perceptual information and consequently leads to the slight improvement of textual vectors.

2.2.2 Separate training models

The simplest approach is concatenation which fuses textual and visual vectors by concatenating them. It has been proven to be effective in learning multimodal representations (Bruni et al., 2014; Hill et al., 2014; Collell et al., 2017). Variations of this method employ transformation and dimension reduction on the concatenation result, including application of singular value decomposition (SVD) (Bruni et al., 2014) or canonical correlation analysis (CCA) (Hill et al., 2014). There is also work using deep learning methods to project different modality inputs into a common space, including restricted Boltzman machines (Ngiam et al., 2011; Srivastava and Salakhutdinov, 2012), autoencoders (Silberer and Lapata, 2014; Silberer et al., 2016), and recursive neural networks (Socher et al., 2013). However, the above methods can only generate multimodal vectors of those words that have perceptual information, thus reducing multimodal vocabulary drastically.

An empirically superior model addresses this problem by predicting missing perceptual information firstly. This includes Hill et al. (2014) who utilize the ridge regression method to learn a mapping matrix from textual modality to visual modality, and Collell et al. (2017) who employ a feed-forward neural network to learn the mapping relation between textual vectors and visual vectors. Applying the mapping function on textual representations, they obtain the predicted visual vectors for all words in textual vocabulary. Then they calculate multimodal representations by concatenating textual and predicted visual vectors. However, the above methods learn separate mapping functions and fusion models, which are somewhat inelegant. In this paper we employ a neural-network mapping function to integrate these two processes into a unified multimodal models.

According to this classification, our method falls into the second group. However, existing models ignore either the associative relations among modalities, associative relations among relative words, or the different contributions of each modality. This paper aims to integrate more perceptual information and the human-like associative memory into a unified multimodal model to learn better word representations.

3 Associative Multichannel Autoencoder

We first provide a brief description of the basic multichannel autoencoder for learning multimodal word representations (Figure 1). Then we extend the model with two associative memory modules and a gating mechanism (Figure 2) in the next sections.

3.1 Basic Mutichannel Autoencoder

An autoencoder is an unsupervised neural network which is trained to reconstruct a given input from its latent representation (Bengio, 2009). In this work, we propose a variant of autoencoder called multichannel autoencoder, which maps multimodal inputs into a common space.

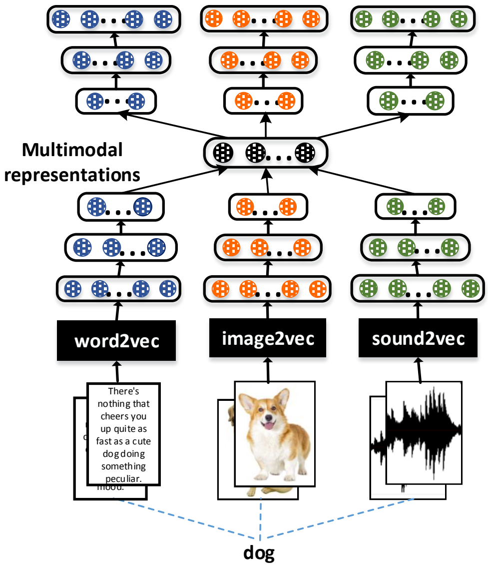

Figure 1: Architecture of the multichannel autoencoder with inputs of textual, visual and auditory sources.

Our model extends the unimodal and bimodal autoencoder (Ngiam et al., 2011; Silberer and Lapata, 2014) to induce semantic representations integrating textual, visual and auditory information. As shown in Figure 1, our model first transforms input textual vector x_t, visual vector x_v and auditory vector x_a to hidden representations:

$$\begin{aligned} h_t &= g(W_t x_t + b_t) \\ h_v &= g(W_v x_v + b_v) \\ h_a &= g(W_a x_a + b_a). \end{aligned} \tag{1}$$

Then the hidden representations are concatenated together and mapped to a common space:

$$h_m = g(W_m[h_t; h_v; h_a] + b_m). \tag{2}$$

The model is trained to reconstruct the hidden representations of the three modalities from the multimodal representation h_m:

$$[\hat{h}_t; \hat{h}_v; \hat{h}_a] = g(W'_m h_m + b_{\hat{m}}), \tag{3}$$

and finally to reconstruct the original embeddings of textual, visual and auditory inputs:

$$\begin{aligned} \hat{x}_t &= g(W'_t \hat{h}_t + b_{\hat{t}}) \\ \hat{x}_v &= g(W'_v \hat{h}_v + b_{\hat{v}}) \\ \hat{x}_a &= g(W'_a \hat{h}_a + b_{\hat{a}}), \end{aligned} \tag{4}$$

where $\hat{x}_t$, $\hat{x}_v$, $\hat{x}_a$ are the reconstruction of input vectors x_t, x_v, x_a, and $\hat{h}_t$, $\hat{h}_v$, $\hat{h}_a$

are the reconstruction of hidden representations h_t, h_v, h_a. The learning parameters $\{W_t, W_v, W_a, W_t', W_v', W_a', W_m, W_m'\}$ are weight matrices, $\{b_t, b_v, b_a, b_{\hat{t}}, b_{\hat{v}}, b_{\hat{a}}, b_m, b_{\hat{m}}\}$ are bias vectors. Here $[\cdot\,;\,\cdot]$ denotes the vector concatenation, and g denotes the non-linear function which we use $tanh(\cdot)$.

Training a single-layer autoencoder corresponds to optimizing the learning parameters to minimize the overall loss between inputs and their reconstructions. Following (Vincent et al., 2010), we use squared loss:

$$\min_{\theta_1} \sum_{i=1}^{n} (||x_t^i - \hat{x}_t^i||^2 + ||x_v^i - \hat{x}_v^i||^2 + ||x_a^i - \hat{x}_a^i||^2), \tag{5}$$

where i denotes the i^{th} word, and the model parameters are $\theta_1 = \{W_t, W_v, W_a, W_m, W_t', W_v', W_a', W_m', b_t, b_v, b_a, b_m, b_{\hat{t}}, b_{\hat{v}}, b_{\hat{a}}, b_{\hat{m}}\}$.

Autoencoders can be stacked to create deep networks. To enhance the quality of semantic representations, we employ a stacked multichannel autoencoder, which is composed of multiple hidden layers that are stacked together.

3.2 Integrating Modality Associations

In reality, the words that have corresponding images or sounds are only a small subset of the textual vocabulary. To obtain the perceptual vectors for each word, we need associations between modalities (i.e., text-to-vision and text-to-audition mapping functions), that transform the textual vectors into visual and auditory ones. Previous methods learn separate mapping functions and fusion models, which are somewhat inelegant. Here we employ a neural-network mapping function to incorporate this modality association module into multimodal models.

Take text-to-vision mapping as an example. Suppose that $T \in \mathbb{R}^{m_t \times n_t}$ is the textual representation containing m_t words, $V \in \mathbb{R}^{m_v \times n_v}$ is the visual representation containing m_v ($\ll m_t$) words, where n_t and n_v are dimensions of the textual and visual representations respectively. The textual and visual representations of the i^{th} concept are denoted as T_i and V_i respectively. Our goal is to learn a mapping function $f : g(W_p T + b_p)$ from textual to visual space such that the prediction $f(T_i)$ is similar to the actual visual vector V_i. The set of visual representations along with their corresponding textual representations are used to learn the mapping function. To train the model, we employ a square loss:

$$\min_{\theta_2} \sum_{i=1}^{m_v} ||f(T_i) - V_i||^2, \tag{6}$$

where the training parameters are $\theta_2 = \{W_p, b_p\}$. We adopt the same method to learn the text-to-audition mapping function.

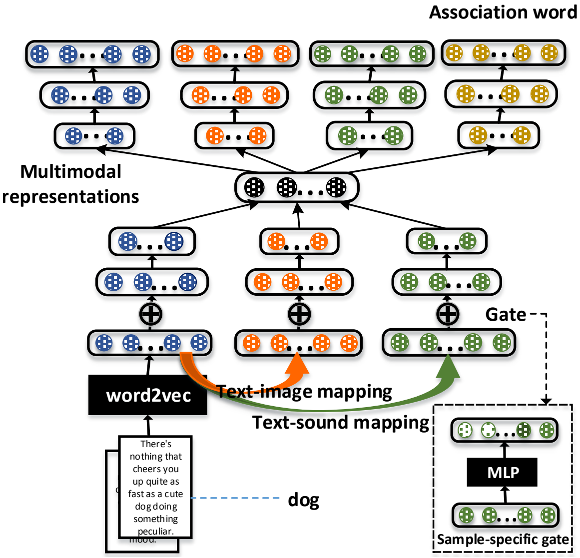

Figure 2: Architecture of the proposed associative multichannel autoencoder.

3.3 Integrating Word Associations

Word associations are a proxy for an aspect of human semantic memory that is not sufficiently captured by the usual training objectives of multimodal models. Therefore we assume that incorporating the objective of word associations helps to learn better semantic representations. To achieve this, we propose to reconstruct the vector of associated word from the corresponding multimodal semantic representation. Specifically, in the decoding process we change the equation (3) to:

$$[\hat{h}_t, \hat{h}_v, \hat{h}_a, \hat{h}_{asc}] = g(W_m' h_m + b_{\hat{m}}), \tag{7}$$

and equation (4) to:

$$\begin{aligned} \hat{x}_t &= g(W_t' \hat{h}_t + b_{\hat{t}}) \\ \hat{x}_v &= g(W_v' \hat{h}_v + b_{\hat{v}}) \\ \hat{x}_a &= g(W_a' \hat{h}_a + b_{\hat{a}}) \\ \hat{x}_{asc} &= g(W_{asc} \hat{h}_{asc} + b_{asc}). \end{aligned} \tag{8}$$

To train the model, we add an additional objective function, which is the mean square error between the embeddings of the associated word y and their re-constructive embeddings $\hat{x}_{asc}$:

$$\min_{\theta_3} \sum_{i=1}^{n} ||y^i - \hat{x}^i_{asc}||^2, \tag{9}$$

where y^i and x^i are the embeddings of a pair of associated words. Here, y is the concatenation of three unimodal vectors $[y_t; y_v; y_a]$. The parameters of word association module are $\theta_3 = \{W_t, W_v, W_a, W_m, \hat{W}_m, W_{asc}, b_t, b_v, b_a, b_m, b_{\hat{m}}, b_{asc}\}$. This additional criterion drives the learning towards a semantic representation capable of reconstructing its associated representation.

3.4 Integrating a Gating Mechanism

Considering that the meaning of each word has different dependencies on textual and perceptual information, we propose the sample-specific gate to assign different weights to each modality according to different words. The weight parameters are calculated by the following feed-forward neural networks:

$$\begin{aligned} g_t &= g(W_{gt}x_t + b_{gt}) \\ g_v &= g(W_{gv}x_v + b_{gv}) \\ g_a &= g(W_{ga}x_a + b_{ga}), \end{aligned} \tag{10}$$

where g_t, g_v and g_a are value or vector gate of textual, visual and auditory representations respectively. For the value gate, W_{gt}, W_{gv} and W_{ga} are vectors, and b_{gt}, b_{gv} and b_{ga} are value parameters. For the vector gate, the parameters W_{gt}, W_{gv} and W_{ga} are matrices, b_{gt}, b_{gv} and b_{ga} are vectors. The value gate controls the importance weights of different input representations as a whole, whereas the vector gate can adjust the importance weights of each dimension of input representations.

Finally, we compute element-wise multiplication of the textual, visual and auditory representations with their corresponding gates:

$$\begin{aligned} x_{gt} &= x_t \odot g_t \\ x_{gv} &= x_v \odot g_v \\ x_{ga} &= x_a \odot g_a. \end{aligned} \tag{11}$$

The x_{gt}, x_{gv} and x_{ga} can be seen as the weighted textual, visual and auditory representations. The parameters of our gating mechanism is trained together with that of the proposed model.

3.5 Model Training

To train the AMA model, we use overall objective function of equation $(5) + (6) + (9)$. In the training phase, model inputs are textual vectors, the corresponding visual vectors, auditory vectors, and association words (Figure 2). In the testing phase, we only need textual inputs to generate multimodal word representations.

4 Experimental Setup

4.1 Datasets

Textual vectors. We use 300-dimensional GloVe vectors[1] which are trained on the Common Crawl corpus consisting of 840B tokens and a vocabulary of 2.2M words[2].

Visual vectors. Our source of visual vectors are collected from ImageNet (Russakovsky et al., 2015) which covers a total of 21,841 WordNet synsets (Fellbaum, 1998) that have 14,197,122 images. For our experiments, we delete words with fewer than 50 images or words not in the Glove vectors, and sample at most 100 images for each word. To generate a visual vector for each word, we use the forward pass of a pre-trained VGG-net model[3] and extract the hidden representation of the last layer as the feature vector. Then we use averaged feature vectors of the multiple images corresponding to the same word. Finally, we get 8,048 visual vectors of 128 dimensions.

Auditory vectors. For auditory data, we gather audio files from Freesound[4], in which we select words with more than 10 audio files and sample at most 50 sounds for one word. To extract auditory features, we use the VGG-net model which is pre-trained on Audioset[5]. The final auditory vectors are averaged feature vectors of multiple audios of the same word, which contains 9,988 words of 128 dimensions[6].

Word associations. We use the word association data collected by (De Deyne et al., 2016), in which each word pair is generated by at least

---

[1] `http://nlp.stanford.edu/projects/glove`

[2] We have tried skip-gram vectors and get the same conclusions.

[3] `http://www.vlfeat.org/matconvnet/`

[4] `http://www.freesound.org/`

[5] `https://research.google.com/audioset`

[6] We build auditory vectors with the released code at: `https://github.com/tensorflow/models/tree/master/research/audioset`

one subject[7]. This dataset includes mostly words with similar meaning (e.g., occasionally & sometimes, adored & loved, supervisor & boss) and related words (e.g., eruption & volcano, cortex & brain, umbrella & rain). We calculate the association score for each word pair (cue word + target word) as: *the number of person who generated the word pair divided by the total number of people who were presented with the cue word.* For training, we select pairs of associated words above a threshold of 0.15 and delete those that are not in the Glove vocabulary, which results in 7,674 word association data sets[8]. For the development set, we randomly sample 5,000 word association collections together with their association scores.

4.2 Model Settings

Our models are implemented with PyTorch (Paszke et al., 2017), optimized with Adam (Kingma and Ba, 2014). We set the initial learning rate to 0.05, and batch size to 64. We tune the number of layers over 1, 2, 3, the size of multimodal vectors over 100, 200, 300, and the size of each layer in textual channel over 300, 250, 200, 150, 100 and in visual/auditory channel over 128, 120, 90, 60. We train the model for 500 epochs and select the best parameters on the development set. All models are trained for 3 times and the average results are reported in Table 1.

To test the effect of each module, we separately train the following models: multichannel autoencoder with modality association (AMA-M), with modality and word associations (AMA-MW), with modality and word associations plus value/vector gate (AMA-MW-Gval/vec).

For AMA-M model, we initialize the text-to-vision and text-to-audition mapping functions with pre-trained mapping matrices, which are parameters of one-layer feed-forward neural networks. The network uses input of the textual vectors, output of visual or auditory vectors, and is trained with SGD for 100 epochs. We initialize the network biases as zeros and network weights with He-initialisation (He et al., 2015). The best parameters of AMA-M model are 2 hidden layers, with textual channel size of 300, 250 and 150, visual/auditory channel size of 128, 90, 60. For AMA-MW model, we use the best AMA-M model parameters as initialization, and train the model with word association data. The optimal parameter of association channel size is 300, 350, 556 (or 428 for bimodal inputs). For AMA-MW-Gval and AMA-MW-Gvec, we adopt the same training strategy as AMA-MW model. The code for training and evaluation can be found at: `https://github.com/wangshaonan/Associative-multichannel-autoencoder`.

5 Experiments

5.1 Evaluation Tasks

We test the baseline and proposed models on six standard evaluation benchmarks, covering two different tasks: (i) Semantic relatedness: Men-3000 (Bruni et al., 2014) and Wordrel-252 (Agirre et al., 2009); (ii) Semantic similarity: Simlex-999 (Hill et al., 2016), Semsim-7576 (Silberer and Lapata, 2014), Wordsim-203 and Simverb-3500 (Gerz et al., 2016). All test sets contain a list of word pairs along with their subject ratings.

We employ Spearman's correlation method to evaluate the performance of our models. This method calculates the correlation coefficients between model predictions and subject ratings, in which the model prediction is the cosine similarity between semantic representations of two words.

5.2 Baseline Multimodal Models

Most of existing multimodal models only utilize textual and visual modalities. For fair comparison, we re-implement several representative systems with our own textual and visual vectors. The **Concatenation (CONC) model** (Kiela and Bottou, 2014) is simple concatenation of normalized textual and visual vectors. The **Mapping** (Collell et al., 2017) and **Ridge** (Hill et al., 2014) models first learn a mapping matrix from textual to visual modality using feed-forward neural network and ridge regression respectively. After applying the mapping function on the textual vectors, they obtain the predicted visual vectors for all words in textual vocabulary. Then they concatenate the normalized textual and predicted visual vectors to get multimodal word representations. The **SVD** (Bruni et al., 2014) and **CCA** (Hill et al., 2014) models first concatenate normalized textual and visual vectors, and then conduct SVD or CCA transformations on the concatenated vectors.

For multimodal models with textual, visual and

---

[7] The dataset can be found at: `https://simondedeyne.me/data`.

[8] We have done experiments with Synonyms (which are extracted from WordNet and PPDB corpora), and the results are not as good as using word associations.

Table 1: Spearman's correlations between model predictions and human ratings on six evaluation datasets. Here T, V, A denote textual, visual and auditory. TV denotes bimodal inputs of textual and visual. TVA denotes trimodal inputs of textual, visual and auditory. The bold scores are the best results per column in bimodal models and trimodal models respectively. For each test, ALL corresponds to the whole testing set, V/A to those word pairs for which we have textual&visual vectors in bimodal models or textual&visual&auditory in trimodal models, and ZS (zero-shot) denotes word pairs for which we have only textual vectors. The #inst. denotes the number of word pairs.

	MEN			SIMLEX			SEMSIM			SIMVERB			WORDSIMM			WORDREL		
	ALL	V/A	ZS	ALL	V/A	ZS	ALL	V/A	ZS	ALLL	V/A	ZS	ALL	V/A	ZS	ALL	V/A	ZS
Kiela & Bottou 2014	-	0.72	-	-	-		-	-	-	-	-	-	-	-	-	-	-	-
Silberer & Lapata 2014	-	-	-	-	-	-	0.70	-	-	-	-	-	-	-	-	-	-	-
Lazaridou et al., 2015	0.75	0.76	-	0.40	0.53	-	0.72	0.72	-	-	-	-	-	-	-	-	-	-
Collell et al., 2017	0.811	0.819	0.802	0.410	0.388	0.422	0785	0.791	0.764	0.286	0.371	0.285	0.781	0.698	0.766	0.629	0.797	0.601
Glove-textual (V)	0.802	0.799	0.788	0.408	0.371	0.429	0.744	0.751	0.716	0.283	0.320	0.282	0.798	0.688	0.779	0.682	0.759	0.661
Glove-textual (A)	0.802	0.801	0.830	0.408	0.399	0.456	0.744	0.715	0.762	0.283	0.129	0.397	0.798	0.805	0.785	0.682	0.708	0.652
CNN-visual	-	0.566	-	-	0.406	-	-	0.502	-	-	0.235	-	-	0.526	-	-	0.422	-
Predicted-visual	0.698	0.757	0.656	0.372	0.458	0.347	0.702	0.700	0.709	0.212	0.194	0.211	0.596	0.621	0.557	0.412	0.604	0.384
CNN-auditory	-	0.266	-	-	0.053	-	-	0.159	-	-	0	-	-	0.231	-	-	0.088	-
Predicted-auditory	0.558	0.555	0.597	0.270	0.251	0.296	0.547	0.531	0.559	0.157	0.074	0.227	0.515	0.496	0.544	0.388	0.400	0.372
CONC (TV)	-	0.786	-	-	0.442	-	-	0.709	-	-	0.437	-	-	0.665	-	-	0.666	-
Mapping (TV)	0.806	0.815	0.782	0.408	0.407	0.410	0.769	0.771	0.709	0.282	0.358	0.272	0.781	0.696	0.768	0.650	0.751	0.594
Ridge (TV)	0.806	0.816	0.786	0.418	0.405	0.429	0.764	0.766	0.756	0.287	0.329	0.285	0.786	0.689	0.771	0.660	0.765	0.640
SVD (TV)	0.806	0.816	0.786	0.418	0.405	0.429	0.764	0.766	0.756	0.287	0.330	0.286	0.786	0.689	0.771	0.660	0.764	0.640
CCA (TV)	0.816	0.833	0.798	0.478	**0.507**	0.493	0.656	0.666	0.619	0.333	0.276	0.334	0.757	0.754	0.704	0.626	0.733	0.599
AMA-M (TV)	0.836	0.822	0.834	0.445	0.460	0.471	**0.781**	**0.784**	0.769	0.324	0.403	0.323	0.807	0.754	0.769	0.681	0.814	0.648
AMA-MW (TV)	0.838	0.824	0.822	0.471	0.446	**0.509**	0.757	0.738	0.723	0.343	0.421	0.340	**0.814**	**0.780**	0.737	0.707	0.744	0.659
AMA-MW-Gval (TV)	**0.845**	**0.835**	**0.841**	0.476	0.472	0.506	0.776	0.778	0.767	**0.352**	0.396	**0.352**	0.808	0.758	0.763	**0.726**	0.796	**0.705**
AMA-MW-Gvec (TV)	0.840	0.831	0.835	**0.485**	0.486	0.505	0.766	0.769	**0.778**	0.343	**0.523**	0.342	0.811	0.769	**0.778**	0.694	**0.846**	0.661
CONC (TVA)	-	0.778	-	-	0.451	-	-	0.661	-	-	0.503	-	-	0.687	-	-	0.593	-
Ridge (TVA)	0.805	0.812	0.791	0.417	0.428	0.420	0.764	0.725	0.781	0.286	0.557	0.285	0.785	0.733	0.762	0.659	0.716	0.646
AMA-M (TVA)	0.831	0.814	0.832	0.452	0.488	0.472	**0.778**	**0.741**	**0.793**	0.333	0.531	0.332	0.805	0.751	0.784	0.685	0.703	0.670
AMA-MW (TVA)	0.838	0.826	0.838	0.481	**0.508**	0.508	0.762	0.726	0.777	**0.358**	**0.605**	**0.357**	**0.814**	**0.821**	0.787	**0.734**	**0.819**	**0.711**
AMA-MW-Gval (TVA)	**0.849**	**0.832**	**0.851**	**0.488**	0.500	**0.509**	0.772	0.729	0.790	0.347	0.598	0.347	0.810	0.806	0.782	0.730	0.761	0.710
AMA-MW-Gvec (TVA)	0.843	0.815	0.843	0.477	0.505	0.497	0.767	0.733	0.781	0.346	0.564	0.346	0.812	0.779	**0.788**	0.723	0.729	0.705
#inst.-visual	3000	1065	1935	999	261	738	7546	5757	1789	3500	41	3459	201	45	158	245	28	224
#inst.-auditory	3000	2732	268	999	741	258	7546	2816	4730	3500	1362	2138	201	129	72	245	153	92
#inst.-visual-auditory	3000	964	2036	999	238	761	7546	2322	5224	3500	22	3478	201	30	171	245	25	220

auditory inputs, we implement **CONC** and **Ridge** as baseline models. The trimodal CONC model simply concatenates normalized textual, visual and auditory vectors. The trimodal Ridge model first learns text-to-vision and text-to-audition mapping matrices with ridge regression method. Then it applies the mapping functions on the textual vectors to get the predicted visual and auditory vectors. Finally, the normalized textual, predicted-visual and predicted-auditory vectors are concatenated to get the multimodal representations.

All above baseline models are implemented with Sklearn[9]. Same as the proposed AMA model, the hyper-parameters of baseline models are tuned on the development set using Spearman's correlation method. In Ridge model, the optimal regularization parameter is 0.6. The Mapping model is trained with SGD for maximum 100 epochs with early stopping, and the optimal learning rate is 0.001. The output dimension of SVD and CCA models are 300.

5.3 Results and Discussion

As shown in Table 1, we divide all models into six groups: (1) existing multimodal models (with textual and visual inputs) in which results are reprinted from Collell et al. (2017). (2) Unimodal models with textual, (predicted) visual or (pre-

---

[9] `http://scikit-learn.org/`

dicted) auditory inputs. (3) Our re-implementation of baseline bimodal models with textual and visual inputs (TV). (4) Our AMA models with textual and visual inputs. (5) Our implementation of trimodal baseline models with textual, visual and auditory inputs (TVA). (6) Our AMA model with textual, visual and auditory inputs.

Overall performance Our AMA models (in group 4 and 6) clearly outperform their baseline unimodal and multimodal models (in group 2, 3 and 5). We use Wilcoxon signed-rank test to check if significant difference exists between two models. Results show that our multimodal models perform significantly better ($p < 0.05$) than all baseline models.

As shown clearly, our bimodal and trimodal AMA models achieve better performance than baselines in both V/A (visual or auditory, the testing data that have associated visual or auditory vectors) and ZS (zero-shot, the testing data that do not have associated visual or auditory vectors) region. In other words, our models outperform baseline models on words with or without perceptual information. The good results in ZS region also indicate that our models have good generalization capacity.

Unimodal baselines As shown in group 2, the Glove vectors are much better than CNN-visual and CNN-auditory vectors, in which CNN-auditory has the worst performance on capturing concept similarities. Comparing with visual and auditory vectors, the predicted visual and auditory vectors achieve much better performance. This indicates that the predicted vectors contain richer information than purely perceptual representations and are more useful for building semantic representations.

Multimodal baselines For bimodal models (group 3), the CONC model that combines Glove and visual vectors performs worse than Glove on four out of six datasets, suggesting that simple concatenation might be suboptimal. The Mapping and Ridge models, which combine Glove and predicted visual vectors, improve over Glove on five out of six datasets in ALL regions. This reinforces the conclusion that the predicted visual vectors are more useful in building multimodal models. The SVD model gets similar results as Ridge model. The CCA model maps different modality inputs into a common space, achieving better results on some datasets and worse results on the others. The improvement on three benchmark tests shows the potential of mapping multimodal inputs into a common space.

The above results can also be observed in the trimodal CONC and Ridge models (group 5). Overall, the trimodal models, which utilize additional auditory inputs, get slightly worse performance than bimodal models. This is partly caused by the fusion method of concatenation. Note that our proposed AMA models are more effective with trimodal inputs as shown in group 6.

Our multimodal models With either bimodal or trimodal inputs, the proposed AMA-M model outperforms all baseline models by a large margin. Specifically our AMA-M model achieves an relative improvement of 4.1% on average (4.5% with trimodal inputs) over the state-of-the-art Ridge model. This illustrates that our AMA models can productively combine textual and perceptual representations. Moreover, our AMA-MW model, which employs word associations, achieves an average improvement of 1.5% (2.7% with trimodal inputs) over the AMA-M model. That is to say, the representation ability of multimodal models can be clearly improved by learning associative relations between words. Furthermore, the AMA-MW-Gval model improves the AMA-MW model by 1.3% (0.3% with trimodal inputs) on average, illustrating that the gating mechanism (especially the value gate) helps to learn better semantic representations.

In addition, we explore the effect of word association data size. We find that the decrease of association data has no discernible effect on model performance: when using 100%, 80%, 60%, 40%, 20% of the data, the average results are 0.6479, 0.6409, 0.6361, 0.6430, 0.6458 in bimodal model. The same trend is observed in trimodal models.

6 Conclusions and Future Work

We have proposed a cognitively-inspired multimodal model — associative multichannel autoencoder — which utilizes the associations between modalities and related words to learn multimodal word representations. Performance improvement on six benchmark tests shows that our models can efficiently fuse different modality inputs and build better semantic representations.

Ultimately, the present paper sheds light on the fundamental questions of how to learn word meanings, such as the plausibility of reconstructing perceptual information, associating related concepts and grounding word symbols to external environment. We believe that one of the promising future directions is to learn from how humans learn and store semantic word representations to build a more effective computational model.

Acknowledgement

The research work descried in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002103 and also supported by the Natural Science Foundation of China under Grant No. 61333018. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve this paper.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL*, pages 19–27.

Andrew J. Anderson, Douwe Kiela, Stephen Clark, and Massimo Poesio. 2017. Visually grounded and textual semantic models differentially decode brain activity associated with concrete and abstract nouns. *TACL*, 5:17–30.

John R Anderson and Gordon H Bower. 2014. *Human associative memory*. Psychology press.

Mark Andrews, Gabriella Vigliocco, and David Vinson. 2009. Integrating experiential and distributional data to learn semantic representations. *Psychological review*, 116(3):463.

Lawrence W Barsalou. 2008. Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, 49(2014):1–47.

Stephen Clark. 2015. Vector space models of lexical meaning. *Handbook of Contemporary Semantic Theory, The*, pages 493–522.

Guillem Collell, Teddy Zhang, and Marie-Francine Moens. 2017. Imagined visual representations as multimodal embeddings. In *AAAI*.

Allan M. Collins and Elizabeth F. Loftus. 1975. A spreading-activation theory of semantic processing. *Psychological Review*, 82:407–428.

Simon De Deyne, Amy Perfors, and Daniel J Navarro. 2016. Predicting human similarity judgments with distributional models: The value of word associations. *COLING*, pages 1861–1870.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Felix Hill and Anna Korhonen. 2014. Learning abstract concept embeddings from multi-modal data: Since you probably can't see what i mean. In *EMNLP*, pages 255–265.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Multi-modal models for concrete and abstract concept meaning. *Transactions of the Association for Computational Linguistics*, 2:285–296.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Merrill Hiscock. 1974. Imagery and verbal processes. *Psyccritiques*, 19(6):487.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *EMNLP*, pages 36–45.

Douwe Kiela and Stephen Clark. 2015. Multi-and cross-modal semantics beyond vision: Grounding in auditory perception. In *EMNLP*, pages 2461–2470.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Barbara Landau, Linda Smith, and Susan Jones. 1998. Object perception and object naming in early development. *Trends in cognitive sciences*, 2(1):19–24.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. *ACL*, pages 153–163.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *ICML*, pages 689–696.

Allan Paivio. 1990. *Mental representations: A dual coding approach*. Oxford University Press.

Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2017. Pytorch.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew A Lambon Ralph, Elizabeth Jefferies, Karalyn Patterson, and Timothy T Rogers. 2017. The neural and computational bases of semantic cognition. *Nature Reviews Neuroscience*, 18(1):42.

Stephen Roller and Sabine Schulte im Walde. 2013. A multimodal lda model integrating textual, cognitive and visual modalities. In *EMNLP*, pages 1146–1157.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2016. Visually grounded meaning representations. *IEEE transactions on pattern analysis and machine intelligence*.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *EMNLP*, pages 1423–1433.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *ACL*, pages 721–732.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943.

Nitish Srivastava and Ruslan R Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

Jing Wang, Julie A. Conder, David N. Blitzer, and Svetlana. V. Shinkareva. 2010. Neural representation of abstract and concrete concepts: A meta-analysis of neuroimaging studies. *Human brain mapping*, 31(10):1459–1468.

Shaonan Wang, Jiajun Zhang, Nan Lin, and Chengqing Zong. 2018a. Investigating inner properties of multimodal representation and semantic compositionality with brain-based componential semantics. In *AAAI*, pages 5964–5972.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2018b. Learning multimodal word representation via dynamic fusion methods. In *AAAI*, pages 5973–5980.

Game-Based Video-Context Dialogue

Ramakanth Pasunuru and **Mohit Bansal**
UNC Chapel Hill
{ram, mbansal}@cs.unc.edu

Abstract

Current dialogue systems focus more on textual and speech context knowledge and are usually based on two speakers. Some recent work has investigated static image-based dialogue. However, several real-world human interactions also involve dynamic visual context (similar to videos) as well as dialogue exchanges among multiple speakers. To move closer towards such multimodal conversational skills and visually-situated applications, we introduce a new video-context, many-speaker dialogue dataset based on live-broadcast soccer game videos and chats from Twitch.tv. This challenging testbed allows us to develop visually-grounded dialogue models that should generate relevant temporal and spatial event language from the live video, while also being relevant to the chat history. For strong baselines, we also present several discriminative and generative models, e.g., based on tridirectional attention flow (TriDAF). We evaluate these models via retrieval ranking-recall, automatic phrase-matching metrics, as well as human evaluation studies. We also present dataset analyses, model ablations, and visualizations to understand the contribution of different modalities and model components.

1 Introduction

Dialogue systems or conversational agents which are able to hold natural, relevant, and coherent interactions with humans have been a long-standing goal of artificial intelligence and machine learning. There has been a lot of important previous work in this field for decades (Weizenbaum, 1966; Isbell et al., 2000; Rambow et al., 2001; Rieser et al., 2005; Georgila et al., 2006; Rieser and Lemon, 2008; Ritter et al., 2011), including recent work on introduction of large textual-dialogue datasets (e.g., Lowe et al. (2015); Serban et al. (2016)) and end-to-end neural network based models (Sordoni et al., 2015; Vinyals and Le, 2015; Su et al., 2016; Luan et al., 2016; Li et al., 2016; Serban et al., 2017a,b).

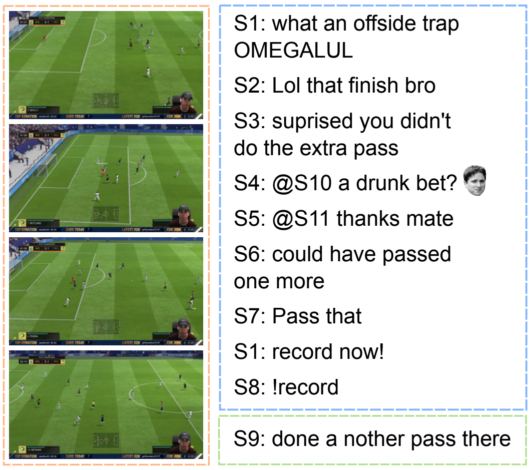

S1: what an offside trap OMEGALUL

S2: Lol that finish bro

S3: suprised you didn't do the extra pass

S4: @S10 a drunk bet?

S5: @S11 thanks mate

S6: could have passed one more

S7: Pass that

S1: record now!

S8: !record

S9: done a nother pass there

Figure 1: Sample example from our many-speaker, video-context dialogue dataset, based on live soccer game chat. The task is to predict the response (bottom-right) using the video context (left) and the chat context (top-right).

Current dialogue tasks are usually focused on the textual or verbal context (conversation history). In terms of multimodal dialogue, speech-based spoken dialogue systems have been widely explored (Eckert et al., 1997; Singh et al., 2000; Young, 2000; Janin et al., 2003; Celikyilmaz et al., 2017; Wen et al., 2015; Su et al., 2016; Mrkšić et al., 2016), as well as work on gesture and haptics based dialogue (Johnston et al., 2002; Cassell, 1999; Foster et al., 2008). In order to address the additional advantage of using visually-grounded context knowledge in dialogue, recent work introduced the visual dialogue task (Das et al., 2017; de Vries et al., 2017; Mostafazadeh et al., 2017). However, the visual context in these tasks is lim-

---

We release all data, code, and models at: `https://github.com/ramakanth-pasunuru/video-dialogue`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 125–136
Brussels, Belgium, October 31 - November 4, 2018.

ited to one static image. Moreover, the interactions are between two speakers with fixed roles (one asks questions and the other answers).

Several situations of real-world dialogue among humans involve more 'dynamic' visual context, i.e., video-style information of the world moving around us (both spatially and temporally). Further, several human conversations involve more than two speakers, with changing roles. In order to develop such dynamically-visual multimodal dialogue models, we introduce a new 'many-speaker, video-context chat' testbed, along with a new dataset and models for the same. Our dataset is based on live-broadcast soccer (FIFA-18) game videos from the 'Twitch.tv' live video streaming platform, along with the spontaneous, many-speaker live chats about the game. This challenging testbed allows us to develop dialogue models where the generated response is required to be relevant to the temporal and spatial events in the live video, as well as be relevant to the chat history (with potential impact towards video-grounded applications such as personal assistants, intelligent tutors, and human-robot collaboration).

We also present several strong discriminative and generative baselines that learn to retrieve and generate bimodal-relevant responses. We first present a triple-encoder discriminative model to encode the video, chat history, and response, and then classify the relevance label of the response. We then improve over this model via tridirectional attention flow (TriDAF). For the generative models, we model bidirectional attention flow between the video and textual chat context encoders, which then decodes the response. We evaluate these models via retrieval ranking-recall, phrase-matching metrics, as well as human evaluation studies. We also present dataset analysis as well as model ablations and attention visualizations to understand the contribution of the video vs. chat modalities and the model components.

2 Related Work

Early dialogue systems had components of natural language (NL) understanding unit, dialogue manager, and NL generation unit (Bates, 1995). Statistical learning methods were used for automatic feature extraction (Dowding et al., 1993; Mikolov et al., 2013), dialogue managers incorporated reward-driven reinforcement learning (Young et al., 2013; Shah et al., 2016), and the generation units have been extended with seq2seq neural network models (Vinyals and Le, 2015; Serban et al., 2016; Luan et al., 2016).

In addition to the focus on textual dialogue context, using multimodal context brings more potential for having real-world grounded conversations. For example, spoken dialogue systems have been widely explored (Singh et al., 2000; Gurevych and Strube, 2004; Georgila et al., 2006; Eckert et al., 1997; Young, 2000; Janin et al., 2003; De Mori, 2007; Wen et al., 2015; Su et al., 2016; Mrkšić et al., 2016; Hori et al., 2016; Celikyilmaz et al., 2015, 2017), as well as gesture and haptics based dialogue (Johnston et al., 2002; Cassell, 1999; Foster et al., 2008). Additionally, dialogue systems for digital personal assistants are also well explored (Myers et al., 2007; Sarikaya et al., 2016; Damacharla et al., 2018). In the visual modality direction, some important recent attempts have been made to use static image based context in dialogue systems (Das et al., 2017; de Vries et al., 2017; Mostafazadeh et al., 2017), who proposed the 'visual dialog' task, where the human can ask questions on a static image, and an agent interacts by answering these questions based on the previous chat context and the image's visual features. Also, Celikyilmaz et al. (2014) used visual display information for on-screen item resolution in utterances for improving personal digital assistants.

In contrast, we propose to employ dynamic video-based information as visual context knowledge in dialogue models, so as to move towards video-grounded intelligent assistant applications. In the video+language direction, previous work has looked at video captioning (Venugopalan et al., 2015) as well as Q&A and fill-in-the-blank tasks on videos (Tapaswi et al., 2016; Jang et al., 2017; Maharaj et al., 2017) and interactive 3D environments (Das et al., 2018; Yan et al., 2018; Gordon et al., 2017; Anderson et al., 2017). There has also been early related work on generating sportscast commentaries from simulation (RoboCup) soccer videos represented as non-visual state information (Chen and Mooney, 2008). Also, Liu et al. (2016a) presented some initial ideas on robots learning grounded task representations by watching and interacting with humans performing the task (i.e., by converting human demonstration videos to Causal And-Or graphs). On the other hand, we propose a new video-chat dataset where the

2:35:44 lluckbox : @joefasho_ true lol
2:35:46 Anselm2 : @ikenna44 can you update the record pls
2:35:48 Flame_96 : u guys should all ptb to make the game more cancer
2:36:03 L7Gasm : honestly if i saw inception ptb
2:36:03 Moobot : How to earn free bits to donate! https://blog.twitch.tv/a-new-way-to-cheer-ef9e48701ae8#.jzb8vdv7x
2:36:07 L7Gasm : unsub
2:36:10 Flame_96 : then maybe ea would wake up and make a good game
2:36:19 melvin109 : !record
2:36:19 Moobot : 11 - 4
2:36:20 L7Gasm : JK i love u @InceptionXx
2:36:21 : Your mic is picking up a lot of static background noise, have you got mic boost turned on?
2:36:21 Anselm2 : yeah me too
2:36:22 Matt344 : @Flame_96 Imagine everyone PTB, most games on Champions Channel will be 1-0
2:36:24 Meeng6 : !record

Figure 2: Sample page of live broadcast of FIFA-18 game on twitch.tv with concurrent user chat.

dialogue models need to generate the next response in the sequence of chats, conditioned both on the raw video features as well as the previous textual chat history. Moreover, our new dataset presents a many-speaker conversation setting, similar to previous work on meeting understanding and Computer Supported Cooperative Work (CSCW) (Janin et al., 2003; Waibel et al., 2001; Schmidt and Bannon, 1992). In the live video stream direction, Fu et al. (2017) and Ping and Chen (2017) used real-time comments to predict the frame highlights in a video, and Barbieri et al. (2017) presented emotes and troll prediction.

3 Twitch-FIFA Dataset

3.1 Dataset Collection and Processing

For our new video-context dialogue task, we used the publicly accessible Twitch.tv live broadcast platform, and collected videos of soccer (FIFA-18) games along with the users' live chat conversations about the game. This dataset has videos involving various realistic human actions and events in a complex sports environment and hence serves as a good testbed and first step towards multimodal video-based dialogue data. An example is shown in Fig. 1 (and an original screenshot example in Fig. 2), where the users perform a complex 'many-speaker', 'multimodal' dialogue. Overall, we collected 49 FIFA-18 game videos along with their users' chat, and divided them into 33 videos for training, 8 videos for validation, and 8 videos for testing. Each such video is several hours long, providing a good amount of data (Table 2).

To extract triples (instances) of video context, chat context, and response from this data, we divide these videos based on the fixed time frames instead of fixed number of utterances in order to maintain conversation topic clusters (because of the sparse nature of chat utterances count over the time). First, we use 20-sec context windows to extract the video clips and users utterances in this time frame, and use it as our video and chat contexts, resp. Next, the chat utterances in the immediately-following 10-sec window (response window) that do not overlap with the next instance's context window are considered as potential responses.[1] Hence, there are only two instances (triples) in a 60-sec long video, i.e., 20-sec video+chat context window and 10-sec response window, and there is no overlap between the instances. Now, out of these potential responses, to only allow the response that has at least some good coherence and relevance with the chat context's topic, we choose the first (earliest) response that has high similarity with some other utterance in this response window (using 0.5 BLEU-4 threshold, based on manual inspection).[2]

	Relevance to Video+Chat
filtered response wins	34%
1st response wins	3%
Non-distinguishable	63% (56 both-good, 7 both-bad)

Table 1: Human evaluation of our dataset, comparing our filtered responses versus the first response in the window (for relevance w.r.t. video and chat contexts).

Human Quality Evaluation of Data Filtering Process: To evaluate the quality of the responses that result from our filtering process described above, we performed an anonymous (randomly shuffled w/o identity) human comparison between the response selected by our filtering process vs. the first response from the response window without any filtering, based on relevance w.r.t. video and chat context. Table 1 presents the results on 100 sample size, showing that humans in a blind-test found 90% (34+56) of our filtered responses as valid responses, verifying that our response selection procedure is reasonable. Furthermore, out of these 90% valid responses, we found that 55% are chat-only relevant, 11% are video-only relevant, and 24% are both video+chat relevant.

In order to make the above procedure safe and to make the dataset more challenging, we also discourage frequent responses (top-20 most-frequent

---

[1] We use non-overlapping windows because: (1) the utterances are non-uniformly distributed in time and hence if we have a shifting window, sometimes a particular data instance/chunk becomes very sparse and contains almost zero utterances; (2) we do not want overlap between response of one window with the context of the next window, so as to avoid the encoder already having seen the response (as part of context) that the decoder needs to generate for the other window.

[2] Based on intuition that if multiple speakers are saying the same response in that 10-second window, then this response should be more meaningful/relevant w.r.t. chat context.

Statistics	Train	Val	Test
#Videos	33	8	8
Total Hours	58.4	11.9	15.4
Final Filtered #Instances	10,150	2,153	2,780
Avg. Chat Context Length	69.0	63.5	71.2
Avg. Response Length	6.5	6.5	6.1

Table 2: Twitch-FIFA dataset's chat statistics (lengths are defined in terms of number of words).

generic utterances) unless no other response satisfies the similarity condition, hence suppressing the frequent responses.[3] If we couldn't find any utterance based on the multi-response matching procedure described above, then we just consider the first utterance in the 10-second window as the response.[4] We also make sure that the chat context window has at least 4 utterances, otherwise we exclude that context window and also the corresponding response window from the dataset. After all this processing, our final resulting dataset contains $10,510$ samples in training, $2,153$ samples in validation, and $2,780$ samples in test.[5]

3.2 Dataset Analysis

Dataset Statistics Table 2 presents the full statistics on train, validation, and test sets of our Twitch-FIFA dataset, after the filtering process described in Sec. 3.1. As shown, the average chat context length in the dataset is around 68 words, and the average response length is 6.3 words.

Chat Context Size Fig. 3 presents the study of number of utterances in the chat context vs. the number of such training samples. As we limit the minimum number of utterances to 4, chat context with less than 4 utterances is not present in the dataset. From the Fig. 3, it is clear that as the number of utterances in the chat context increases, the number of such training samples decrease.

Frequent Words Fig. 4 presents the top-20 frequent words (excluding stop words) and their corresponding frequency in our Twitch-FIFA dataset. Most of these frequent words are related to soccer vocabulary. Also, some of these frequent words are twitch emotes (e.g. 'kappa', 'inceptionlove').

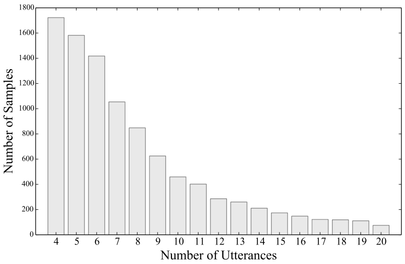

Figure 3: Distribution of #utterances in chat context (w.r.t. the #training examples for each case).

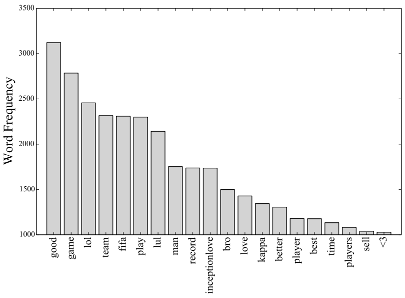

Figure 4: Frequent words in our Twitch-FIFA dataset.

4 Models

Let $v = \{v_1, v_2, .., v_m\}$ be the video context frames, $u = \{u_1, u_2, .., u_n\}$ be the textual chat (utterance) context tokens, and $r = \{r_1, r_2, .., r_k\}$ be response tokens generated (or retrieved).

4.1 Baselines

Our simple non-trained baselines are Most-Frequent-Response (re-rank the candidate responses based on their frequency in the training set), Chat-Response-Cosine (re-rank the candidate responses based on their similarity score w.r.t. the chat context), and Nearest-Neighbor (find the K-best similar chat contexts in the training set, take their corresponding responses, and then re-rank the candidate responses based on mean similarity score w.r.t. this K-best response set). For trained baselines, we use logistic regression and Naive Bayes methods. We use the final state of a Twitch-trained RNN Language Model to represent the chat context and response. Please see supplementary for full details.

4.2 Discriminative Models

4.2.1 Triple Encoder

For our simpler discriminative model, we use a 'triple encoder' to encode the video context, chat context, and response (see Fig. 5), as an extension of the dual encoder model in Lowe et al. (2015). The task here is to predict the given train-

---

[3] Note that this filtering suppresses the performance of simple frequent-response baseline described in Sec. 4.1.

[4] Other preprocessing steps include: omit the utterances in the response window which refer to a speaker name out of the current chat context; remove non-representative utterances, e.g., those with hyperlinks; replace (anonymize) all the user identities mentioned in the utterances with a common tag (i.e., anonymizing due to similar intuitions from the Q&A community (Hermann et al., 2015)).

[5] Note that this is substantially larger than or comparable to most current video captioning datasets. We plan to further extend our dataset based on diverse games and video types.

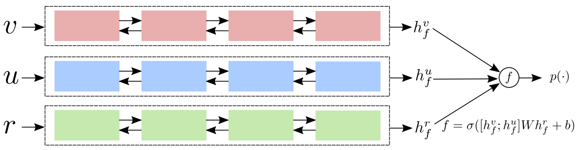

Figure 5: Overview of our 'triple encoder' discriminative model, with bidirectional-LSTM-RNN encoders for video, chat context, and response.

ing triple (v, u, r) as positive or negative. Let h_f^v, h_f^u, and h_f^r be the final state information of the video, chat, and response LSTM-RNN (bidirectional) encoders respectively; then the probability of a positive training triple is defined as follows:

$$p(v, u, r; \theta) = \sigma([h_f^v; h_f^u]^T W h_f^r + b) \qquad (1)$$

where W and b are trainable parameters. Here, W can be viewed as a similarity matrix which will bring the context $[h_f^v; h_f^u]$ into the same space as the response h_f^r, and get a suitable similarity score.

For optimizing our discriminative model, we use max-margin loss function similar to Mao et al. (2016) and Yu et al. (2017). Given a positive training triple (v, u, r), let the corresponding negative training triples be (v', u, r), (v, u', r), and (v, u, r'), i.e., one modality is wrong at a time in each of these three (see Sec. 5 for the negative example selection). The max-margin loss is:

$$\begin{aligned} L(\theta) = \sum [&\max(0, M + \log p(v', u, r) - \log p(v, u, r)) \\ &+ \max(0, M + \log p(v, u', r) - \log p(v, u, r)) \\ &+ \max(0, M + \log p(v, u, r') - \log p(v, u, r))] \end{aligned} \qquad (2)$$

where the summation is over all the training triples in the dataset. M is a tunable margin hyperparameter between positive and negative training triples.

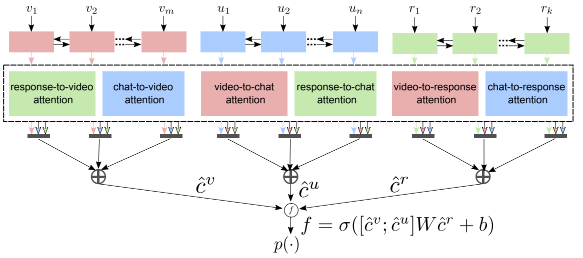

Figure 6: Overview of our tridirectional attention flow (TriDAF) model with all pairwise modality attention modules, as well as self-attention on video context, chat context, and response as inputs.

4.2.2 Tridirectional Attention Flow (TriDAF)

Our tridirectional attention flow model learns stronger joint spaces between the three modalities in a mutual-information way. We use bidirectional attention flow mechanisms (Seo et al., 2017) between the video and chat contexts, between the video context and the response, as well as between the chat context and the response, hence enabling attention flow across all three modalities, as shown in Fig. 6. We name this model Tridirectional Attention Flow or TriDAF. We will next discuss the bidirectional attention flow mechanism between video and chat contexts, but the same formulation holds true for bidirectional attention between video context and response, and between chat context and response. Given the video context hidden state h_i^v and chat context hidden state h_j^u at time steps i and j respectively, the bidirectional attention mechanism is based on the similarity score:

$$S_{i,j}^{(v,u)} = w_{S^{(v,u)}}^T [h_i^v; h_j^u; h_i^v \odot h_j^u] \qquad (3)$$

where $S_{i,j}^{(v,u)}$ is a scalar, $w_{S^{(v,u)}}$ is a trainable parameter, and $\odot$ denote element-wise multiplication. The attention distribution from chat context to video context is defined as $\alpha_{i:} = softmax(S_{i:})$, hence the chat-to-video context vector $c_i^{v \leftarrow u} = \sum_j \alpha_{i,j} h_j^u$. Similarly, the attention distribution from video context to chat context is defined as $\beta_{j:} = softmax(S_{:j})$, hence the video-to-chat context vector $c_j^{u \leftarrow v} = \sum_i \beta_{j,i} h_i^v$.

We then compute similar bidirectional attention flow mechanisms between the video context and response, and between the chat context and response. Then, we concatenate each hidden state and its corresponding context vector from other two modalities, e.g., $\hat{h}_i^v = [h_i^v; c_i^{v \leftarrow u}; c_i^{v \leftarrow r}]$ for the i^{th} timestep of the video context. Finally, we add self-attention mechanism (Lin et al., 2017) across the concatenated hidden states of each of the three modules.[6] If $\hat{h}_i^v$ is the final concatenated vector of the video context at time step i, then the self-attention weights α^s for this video context are the softmax of e^s:

$$e_i^s = V_a^v \tanh(W_a^v \hat{h}_i^v + b_a^v) \qquad (4)$$

where V_a^v, W_a^v, and b_a^v are trainable self-attention parameters. The final representation vector of the full video context after self-attention is $\hat{c}^v = \sum_i \alpha_i^s \hat{h}_i^v$. Similarly, the final representation vectors of the chat context and the response are $\hat{c}^u$ and $\hat{c}^r$, respectively. Finally, the probability that

---

[6] In our preliminary experiments, we found that adding self-attention is 0.92% better in recall@1 and faster than passing the hidden states through another layer of RNN, as done in Seo et al. (2017).

the given training triple (v, u, r) is positive is:

$$p(v, u, r; \theta) = \sigma([\hat{c}^v; \hat{c}^u]^T W \hat{c}^r + b) \quad (5)$$

Again, here also we use max-margin loss (Eqn. 2).

4.3 Generative Models

4.3.1 Seq2seq with Attention

Our simpler generative model is a sequence-to-sequence model with bilinear attention mechanism (similar to Luong et al. (2015)). We have two encoders, one for encoding the video context and another for encoding the chat context, as shown in Fig. 7. We combine the final state information from both encoders and give it as initial state to the response generation decoder. The two encoders and the decoder are all two-layer LSTM-RNNs. Let h_i^v and h_j^u be the hidden states of video and chat encoders at time step i and j respectively. At each time step t of the decoder with hidden state h_t^r, the decoder attends to parts of video and chat encoders and uses the combined information to generate the next token. Let α_t and β_t be the attention weight distributions for video and chat encoders respectively with video context vector $c_t^v = \sum_i \alpha_{t,i} h_i^v$ and chat context vector $c_t^u = \sum_j \beta_{t,j} h_j^u$. The attention distribution for video encoder is defined as (and the same holds for chat encoder):

$$e_{t,i} = h_t^{rT} W_a^v h_i^v; \quad \alpha_t = \text{softmax}(e_t) \quad (6)$$

where W_a^v is a trainable parameter. Next, we concatenate the attention-based context information (c_t^v and c_t^u) and decoder hidden state (h_t^r), and do a non-linear transformation to get the final hidden state $\hat{h}_t^r$ as follows:

$$\hat{h}_t^r = \tanh(W_c[c_t^v; c_t^u; h_t^r]) \quad (7)$$

where W_c is again a trainable parameter. Finally, we project the final hidden state information to vocabulary size and give it as input to a *softmax* layer to get the vocabulary distribution $p(r_t|r_{1:t-1}, v, u; \theta)$. During training, we minimize the cross-entropy loss defined as follows:

$$L_{\text{XE}}(\theta) = -\sum \sum_t \log p(r_t|r_{1:t-1}, v, u; \theta) \quad (8)$$

where the final summation is over all the training triples in the dataset.

Further, to train a stronger generative model with negative training examples (which teaches the model to give higher generative decoder probability to the positive response as compared to all the negative ones), we use a max-margin loss (similar to Eqn. 2 in Sec. 4.2.1):

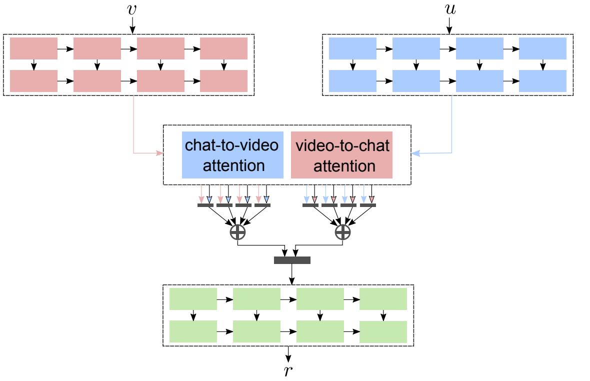

Figure 7: Overview of our generative model with bidirectional attention flow between video context and chat context during response generation.

$$\begin{aligned} L_{\text{MM}}(\theta) = \sum [&\max(0, M + \log p(r|v', u) - \log p(r|v, u)) \\ &+ \max(0, M + \log p(r|v, u') - \log p(r|v, u)) \\ &+ \max(0, M + \log p(r'|v, u) - \log p(r|v, u))] \end{aligned} \quad (9)$$

where the summation is over all the training triples in the dataset. Overall, the final joint loss function is a weighted combination of cross-entropy loss and max-margin loss: $L(\theta) = L_{\text{XE}}(\theta) + \lambda L_{\text{MM}}(\theta)$, where λ is a tunable hyperparameter.

4.3.2 Bidirectional Attention Flow (BiDAF)

The stronger version of our generative model extends the two-encoder-attention-decoder model above to add bidirectional attention flow (BiDAF) mechanism (Seo et al., 2017) between video and chat encoders, as shown in Fig. 7. Given the hidden states h_i^v and h_j^u of video and chat encoders at time step i and j, the final hidden states after the BiDAF are $\hat{h}_i^v = [h_i^v; c_i^{v \leftarrow u}]$ and $\hat{h}_j^u = [h_i^u; c_j^{u \leftarrow v}]$ (similar to as described in Sec. 4.2.2), respectively. Now, the decoder attends over these final hidden states, and the rest of the decoder process is similar to Sec 4.3.1 above, including the weighted joint cross-entropy and max-margin loss.

5 Experimental Setup

Evaluation We first evaluate both our discriminative and generative models using retrieval-based recall@k scores, which is a concrete metric for such dialogue generation tasks (Lowe et al., 2015). For our discriminative models, we simply rerank the given responses (in a candidate list of size 10, based on 9 negative examples; more details below)

Models	r@1	r@2	r@5
BASELINES			
Most-Frequent-Response	10.0	16.0	20.9
Naive Bayes	9.6	20.9	51.5
Logistic Regression	10.8	21.8	52.5
Nearest Neighbor	11.4	22.6	53.2
Chat-Response-Cosine	11.4	22.0	53.2
DISCRIMINATIVE MODEL			
Dual Encoder (C)	17.1	30.3	61.9
Dual Encoder (V)	16.3	30.5	61.1
Triple Encoder (C+V)	18.1	33.6	68.5
TriDAF+Self Attn (C+V)	20.7	35.3	69.4
GENERATIVE MODEL			
Seq2seq +Attn (C)	14.8	27.3	56.6
Seq2seq +Attn (V)	14.8	27.2	56.7
Seq2seq + Attn (C+V)	15.7	28.0	57.0
Seq2seq + Attn + BiDAF (C+V)	16.5	28.5	57.7

Table 3: Performance of our baselines, discriminative models, and generative models for recall@k metrics on our Twitch-FIFA test set. C and V represent chat and video context, respectively.

in the order of the probability score each response gets from the model. If the positive response is within the top-k list, then the recall@k score is 1, otherwise 0, following previous Ubuntu-dialogue work (Lowe et al., 2015). For the generative models, we follow a similar approach, but the reranking score for a candidate response is based on the log probability score given by the generative models' decoder for that response, following the setup of previous visual-dialog work (Das et al., 2017). In our experiments, we use recall@1, recall@2, and recall@5 scores. For completeness, we also report the phrase-matching metric scores: METEOR (Denkowski and Lavie, 2014) and ROUGE (Lin, 2004) for our generative models. We also present human evaluation.

Training Details For negative samples, during training, for every positive triple (video, chat, response) in the training set, we sample 3 random negative triples. For validation/test, we sample 9 random negative responses elsewhere from the validation/test set. Also, the negative samples don't come from the video corresponding to the positive response. More details of negative samples and other training details (e.g., dimension/vocab sizes, visual feature details, validation-based hyperparamater tuning and model selection), are discussed in the supplementary.

6 Results and Analysis

6.1 Human Evaluation of Dataset

First, the overall human quality evaluation of our dataset (shown in Table 1) demonstrates that it contains 90% responses relevant to video and/or chat context. Next, we also do a blind human study on the recall-based setup (on a set of 100 samples from the validation set), where we anonymize the positive response by randomly mixing it with 9 tricky negative responses in the retrieval list, and ask the user to select the most relevant response for the given video and/or chat context. We found that human performance on this task is around 55% recall@1, demonstrating that this 10-way-discriminative recall-based task setup is reasonably challenging for humans,[7] but also that there is a lot of scope for future model improvements because the chance baseline is only 10% and the best-performing model so far (see Sec. 6.3) achieves only 22% recall@1 (on dev set), and hence there is a large 33% gap.

6.2 Baseline Results

Table 3 displays all our primary results. We first discuss results of our simple non-trained and trained baselines (see Sec. 4.1). The 'Most-Frequent-Response' baseline, which just ranks the 10-sized response retrieval list based on their frequency in the training data, gets only around 10% recall@1.[8] Our other non-trained baselines: 'Chat-Response-Cosine' and 'Nearest Neighbor', which ranks the candidate responses based on (Twitch-trained RNN encoder's vector) cosine similarity with chat-context and K-best training contexts' response vectors, respectively, achieves slightly better scores. We also show that our simple trained baselines (logistic regression and nearest neighbor) also achieve relatively low scores, indicating that a simple, shallow model will not work on this challenging dataset.

6.3 Discriminative Model Results

Next, we present the recall@k retrieval performance of our various discriminative models in Ta-

---

[7] This relatively low human recall@1 performance is because this is a challenging, 10-way-discriminative evaluation, i.e., the choice comes w.r.t. 9 tricky negative examples along with just 1 positive example (hence chance-baseline is only 10%). Note that these negative examples are an artifact of specifically recall-based evaluation only, and will not affect the more important real-world task of response generation (for which our dataset's response quality is 90%, as shown in Table 1). Moreover, our dataset filtering (see Sec. 3.1) also 'suppresses' simple baselines and makes the task even harder.

[8] Note that the performance of this baseline is worse than the random choice baseline (recall@1:10%, recall@2:20%, recall@5:50%) because our dataset filtering process already suppresses frequent responses (see Sec. 3.1), in order to provide a challenging dataset for the community.

Models	METEOR	ROUGE-L
Multiple References		
Seq2seq + Atten. (C)	2.59	8.44
Seq2seq + Atten. (V)	2.66	8.34
Seq2seq + Atten. (C+V) $\otimes$	3.03	8.84
$\otimes$ + BiDAF (C+V)	3.70	9.82

Table 4: Performance of our generative models on phrase matching metrics.

Models	Relevance
Seq2seq + Atten. (C+V) wins	41.0 %
BiDAF wins	34.0 %
Non-distinguishable	25.0 %

Table 5: Human evaluation comparing the baseline and BiDAF generative models.

Models	recall@1	recall@2	recall@5
1 neg.	18.21	32.19	64.05
3 neg.	22.20	35.90	68.09

Table 6: Ablation (dev) of one vs. three negative examples for TriDAF self-attention discriminative model.

ble 3: dual encoder (chat context only), dual encoder (video context only), triple encoder, and TriDAF model with self-attention. Our dual encoder models are significantly better than random choice and all our simple baselines above, and further show that they have complementary information because using both of them together (in 'Triple Encoder') improves the overall performance of the model. Finally, we show that our novel TriDAF model with self-attention performs significantly better than the triple encoder model.[9]

6.4 Generative Model Results

Next, we evaluate the performance of our generative models with both retrieval-based recall@k scores and phrase matching-based metrics as discussed in Sec. 5 (as well as human evaluation). We first discuss the retrieval-based recall@k results in Table 3. Starting with a simple sequence-to-sequence attention model with video only, chat only, and both video and chat encoders, the recall@k scores are better than all the simple baselines. Moreover, using both video+chat context is again better than using only one context modality. Finally, we show that the addition of the bidirectional attention flow mechanism improves the performance in all recall@k scores.[10] Note that generative model scores are lower than the discriminative models on retrieval recall@k metric, which is expected (see discussion in previous visual dialogue work (Das et al., 2017)), because discriminative models can tune to the biases in the response candidate options, but generative models are more useful for real-world tasks such as generation of novel responses word-by-word from scratch in Siri/Alexa/Cortana style applications (whereas discriminative models can only rank the pre-given list of responses).

We also evaluate our generative models with phrase-level matching metrics: METEOR and ROUGE-L, as shown in Table 4. Again, our BiDAF model is stat. significantly better than non-BiDAF model on both METEOR ($p < 0.01$) and ROUGE-L ($p < 0.02$) metrics. Since dialogue systems can have several diverse, non-overlapping valid responses, we consider a multi-reference setup where all the utterances in the 10-sec response window are treated as valid responses.[11]

6.5 Human Evaluation of Models

Finally, we also perform human evaluation to compare our top two generative models, i.e., the video+chat seq2seq with attention and its extension with BiDAF (Sec. 4.3), based on a 100-sized sample. We take the generated response from both these models, and randomly shuffle these pairs to anonymize model identity. We then ask two annotators (for 50 task instances each) to score the responses of these two models based on relevance. Note that the human evaluators were familiar with Twitch FIFA-18 video games and also the Twitch's unique set of chat mannerisms and emotes. As shown in Table 5, our BiDAF based generative model performs better than the non-BiDAF one, which is already quite a strong video+chat encoder model with attention.

7 Ablations and Analysis

7.1 Negative Training Pairs

We also compare the effect of different negative training triples that we discussed in Sec. 5. Table 6 shows the comparison between one negative

---

[9] Statistical significance of $p < 0.01$ for recall@1, based on the bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994) with 100K samples.

[10] Stat. signif. $p < 0.05$ for recall@1 w.r.t. Seq2seq+Atten (video+chat); $p < 0.01$ w.r.t. chat- and video-only models.

[11] Liu et al. (2016b) discussed that BLEU and most phrase matching metrics are not good for evaluating dialogue systems. Also, generative models have very low phrase-matching metric scores because the generated response can be valid but still very different from the ground truth reference (Lowe et al., 2015; Liu et al., 2016b; Li et al., 2016). We present results for the relatively better metrics like paraphrase-enabled METEOR for completeness, but still focus on retrieval recall@k and human evaluation.

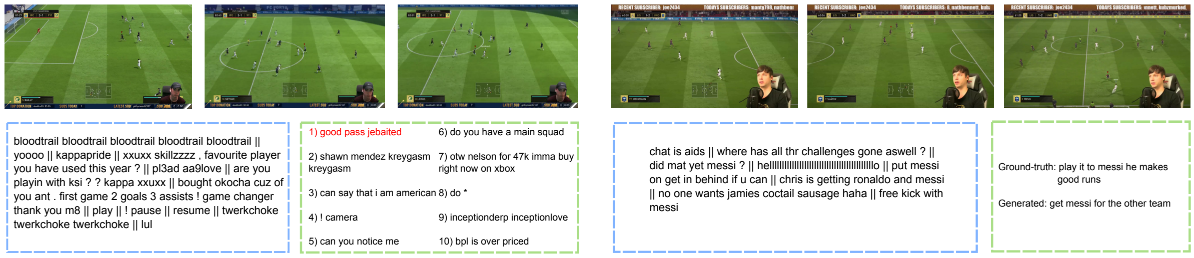

bloodtrail bloodtrail bloodtrail bloodtrail bloodtrail || yoooo || kappapride || xxuxx skillzzzz , favourite player you have used this year ? || pl3ad aa9love || are you playin with ksi ? ? kappa xxuxx || bought okocha cuz of you ant . first game 2 goals 3 assists ! game changer thank you m8 || play || ! pause || resume || twerkchoke twerkchoke twerkchoke || lul

1) good pass jebaited
2) shawn mendez kreygasm kreygasm
3) can say that i am american
4) ! camera
5) can you notice me
6) do you have a main squad
7) otw nelson for 47k imma buy right now on xbox
8) do *
9) inceptionderp inceptionlove
10) bpl is over priced

chat is aids || where has all thr challenges gone aswell ? || did mat yet messi ? || hellllllllllllllllllllllllllllllllllllllo || put messi on get in behind if u can || chris is getting ronaldo and messi || no one wants jamies coctail sausage haha || free kick with messi

Ground-truth: play it to messi he makes good runs

Generated: get messi for the other team

Figure 8: Output retrieval (left) and generative (right) examples from TriDAF and BiDAF models, resp.

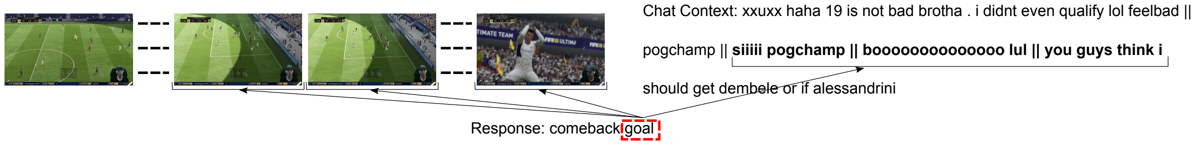

Chat Context: xxuxx haha 19 is not bad brotha . i didnt even qualify lol feelbad || pogchamp || **siiiii pogchamp || booooooooooooooo lul || you guys think i** should get dembele or if alessandrini

Response: comeback goal

Figure 9: Attention visualization: generated word ‘goal’ in response is intuitively aligning to goal-related video frames (top-3-weight frames highlighted) and context words (top-10-weight words highlighted).

training triple (with just a negative response) vs. three negative training triples (one with negative video context, one with negative chat context, and another with negative response), showing that using the 3-negative examples setup is substantially better.

7.2 Discriminative Loss Functions

Table 7 shows the performance comparison between the classification loss and max-margin loss on our TriDAF with self-attention discriminative model (Sec. 4.2.2). We observe that max-margin loss performs better than the classification loss, which is intuitive because max-margin loss tries to differentiate between positive and negative training example triples.

Models	recall@1	recall@2	recall@5
Classification loss	19.32	33.72	66.60
Max-margin loss	22.20	35.90	68.09

Table 7: Ablation of classification vs. max-margin loss on our TriDAF discriminative model (on dev set).

7.3 Generative Loss Functions

For our best generative model (BiDAF), Table 8 shows that using a joint loss of cross-entropy and max-margin is better than just using only cross-entropy loss optimization (Sec. 4.3.1). Max-margin loss provides knowledge about the negative samples for the generative model, hence improves the retrieval-based recall@k scores.

7.4 Attention Visualization and Examples

Finally, we show some interesting output examples from both our discriminative and generative models as shown in Fig. 8. Additionally, Fig. 9 visualizes that our models can learn some correct attention alignments from the generated output response word to the appropriate (goal-related) video frames as well as chat context words.

Models	recall@1	recall@2	recall@5
Cross-entropy (XE)	13.12	23.45	54.78
XE+Max-margin	15.61	27.39	57.02

Table 8: Ablation of cross-entropy loss vs. cross-entropy+maxmargin loss for our BiDAF-based generative model (on dev set).

8 Conclusion

We presented a new game-chat based video-context, many-speaker dialogue task and dataset. We also presented several baselines and state-of-the-art discriminative and generative models on this task. We hope that this testbed will be a good starting point to encourage future work on the challenging video-context dialogue paradigm. In future work, we plan to investigate the effects of multiple users, i.e., the multi-party aspect of this dataset. We also plan to explore advanced video features such as activity recognition, person identification, etc.

Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by DARPA YFA17-D17AP00022, ARO-YIP Award W911NF-18-1-0336, Google Faculty Research Award, Bloomberg Data Science Research Grant, and NVidia GPU awards. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the funding agency.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2017. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*.

Francesco Barbieri, Luis Espinosa-Anke, Miguel Ballesteros, Horacio Saggion, et al. 2017. Towards the understanding of gaming audiences by modeling twitch emotes. In *Third Workshop on Noisy User-generated Text (W-NUT 2017)*.

Madeleine Bates. 1995. Models of natural language understanding. *Proceedings of the National Academy of Sciences*, 92(22):9977–9982.

Justine Cassell. 1999. Embodied conversation: integrating face and gesture into automatic spoken dialogue systems.

Asli Celikyilmaz, Li Deng, and Dilek Hakkani-Tur. 2017. Deep learning for spoken and text dialog systems. *Deep Learning in Natural Language Processing (eds. Li Deng and Yang Liu)*.

Asli Celikyilmaz, Zhaleh Feizollahi, Dilek Hakkani-Tur, and Ruhi Sarikaya. 2014. Resolving referring expressions in conversational dialogs for natural user interfaces. In *EMNLP*, pages 2094–2104.

Asli Celikyilmaz, Zhaleh Feizollahi, Dilek Hakkani-Tur, and Ruhi Sarikaya. 2015. A universal model for flexible item selection in conversational dialogs. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 361–367. IEEE.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135. ACM.

Praveen Damacharla, Parashar Dhakal, Sebastian Stumbo, Ahmad Y Javaid, Subhashini Ganapathy, David A Malek, Douglas C Hodge, and Vijay Devabhaktuni. 2018. Effects of voice-based synthetic assistant on performance of emergency care provider in training. *International Journal of Artificial Intelligence in Education*, pages 1–22.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *CVPR*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *CVPR*.

Renato De Mori. 2007. Spoken language understanding: a survey. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 365–376. IEEE.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

John Dowding, Jean Mark Gawron, Doug Appelt, John Bear, Lynn Cherny, Robert Moore, and Douglas Moran. 1993. Gemini: A natural language system for spoken-language understanding. In *ACL*, pages 54–61.

Wieland Eckert, Esther Levin, and Roberto Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 80–87. IEEE.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Mary Ellen Foster, Ellen Gurman Bard, Markus Guhe, Robin L Hill, Jon Oberlander, and Alois Knoll. 2008. The roles of haptic-ostensive referring expressions in cooperative, task-based human-robot dialogue. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 295–302. ACM.

Cheng-Yang Fu, Joon Lee, Mohit Bansal, and Alexander C Berg. 2017. Video highlight prediction using audience chat reactions. In *EMNLP*.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Ninth International Conference on Spoken Language Processing*.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2017. Iqa: Visual question answering in interactive environments. *arXiv preprint arXiv:1712.03316*.

Iryna Gurevych and Michael Strube. 2004. Semantic similarity applied to spoken dialogue summarization. In *Proceedings of the 20th international conference on Computational Linguistics*, page 764. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701.

Takaaki Hori, Hai Wang, Chiori Hori, Shinji Watanabe, Bret Harsham, Jonathan Le Roux, John R Hershey, Yusuke Koji, Yi Jing, Zhaocheng Zhu, et al. 2016. Dialog state tracking with attention-based sequence-to-sequence learning. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 552–558. IEEE.

Charles Lee Isbell, Michael Kearns, Dave Kormann, Satinder Singh, and Peter Stone. 2000. Cobot in lambdamoo: A social statistics agent. In *AAAI/IAAI*, pages 36–41.

Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. 2017. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *CVPR*, pages 2680–8.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. 2003. The icsi meeting corpus. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–I. IEEE.

Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker, and Preetam Maloor. 2002. Match: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 376–383. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *ACL*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Changsong Liu, Joyce Y Chai, Nishant Shukla, and Song-Chun Zhu. 2016a. Task learning through visual demonstration and situated dialogue. In *AAAI Workshop: Symbiotic Cognitive Systems*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016b. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Yi Luan, Yangfeng Ji, and Mari Ostendorf. 2016. Lstm based conversation models. *arXiv preprint arXiv:1603.09457*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421.

Tegan Maharaj, Nicolas Ballas, Anna Rohrbach, Aaron Courville, and Christopher Pal. 2017. A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering. *CVPR*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *CVPR*, pages 11–20.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios P Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. *arXiv preprint arXiv:1701.08251*.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2016. Neural belief tracker: Data-driven dialogue state tracking. *arXiv preprint arXiv:1606.03777*.

Karen Myers, Pauline Berry, Jim Blythe, Ken Conley, Melinda Gervasio, Deborah L McGuinness, David Morley, Avi Pfeffer, Martha Pollack, and Milind Tambe. 2007. An intelligent personal assistant for task and time management. *AI Magazine*, 28(2):47.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Qing Ping and Chaomei Chen. 2017. Video highlights detection and summarization with lag-calibration based on concept-emotion mapping of crowd-sourced time-sync comments. In *EMNLP Workshop on New Frontiers in Summarization*.

Owen Rambow, Srinivas Bangalore, and Marilyn Walker. 2001. Natural language generation in dialog systems. In *Proceedings of the first international conference on Human language technology research*, pages 1–4. Association for Computational Linguistics.

Verena Rieser, Ivana Kruijff-Korbayová, and Oliver Lemon. 2005. A corpus collection and annotation framework for learning multimodal clarification strategies. In *6th SIGdial Workshop on DISCOURSE and DIALOGUE*.

Verena Rieser and Oliver Lemon. 2008. Learning effective multimodal dialogue strategies from wizard-of-oz data: Bootstrapping and evaluation. In *ACL*, pages 638–646.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *EMNLP*, pages 583–593. Association for Computational Linguistics.

Ruhi Sarikaya, Paul A Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiaohu Liu, et al. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 391–397. IEEE.

Kjeld Schmidt and Liam Bannon. 1992. Taking cscw seriously. *Computer Supported Cooperative Work (CSCW)*, 1(1-2):7–40.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pages 3288–3294.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Pararth Shah, Dilek Hakkani-Tür, and Larry Heck. 2016. Interactive reinforcement learning for task-oriented dialogue management. In *NIPS Deep Learning for Action and Interaction Workshop*.

Satinder P Singh, Michael J Kearns, Diane J Litman, and Marilyn A Walker. 2000. Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems*, pages 956–962.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *ACL*.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *CVPR*, pages 4631–4640.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence-video to text. In *CVPR*, pages 4534–4542.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of ICML Deep Learning Workshop*.

Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *CVPR*.

Alex Waibel, Michael Bett, Florian Metze, Klaus Ries, Thomas Schaaf, Tanja Schultz, Hagen Soltau, Hua Yu, and Klaus Zechner. 2001. Advances in automatic meeting record creation and access. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 597–600. IEEE.

Joseph Weizenbaum. 1966. Elizaa computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Claudia Yan, Dipendra Misra, Andrew Bennnett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. 2018. CHALET: Cornell house agent learning environment. *arXiv preprint arXiv:1801.07357*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Steve J Young. 2000. Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 358(1769):1389–1402.

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. 2017. A joint speaker-listener-reinforcer model for referring expressions. In *CVPR*.

simNet: Stepwise Image-Topic Merging Network for Generating Detailed and Comprehensive Image Captions

Fenglin Liu[1]*, **Xuancheng Ren**[2]*, **Yuanxin Liu**[1], **Houfeng Wang**[2] **and Xu Sun**[2]
[1]School of ICE, Beijing University of Posts and Telecommunications
[2]MOE Key Laboratory of Computational Linguistics, School of EECS, Peking University
lfl@bupt.edu.cn, renxc@pku.edu.cn, yuanxinLIU@bupt.edu.cn
{wanghf, xusun}@pku.edu.cn

Abstract

The encode-decoder framework has shown recent success in image captioning. Visual attention, which is good at detailedness, and semantic attention, which is good at comprehensiveness, have been separately proposed to ground the caption on the image. In this paper, we propose the *S*tepwise *I*mage-Topic *M*erging *N*etwork (*simNet*) that makes use of the two kinds of attention at the same time. At each time step when generating the caption, the decoder adaptively merges the attentive information in the extracted topics and the image according to the generated context, so that the visual information and the semantic information can be effectively combined. The proposed approach is evaluated on two benchmark datasets and reaches the state-of-the-art performances.[1]

1 Introduction

Image captioning attracts considerable attention in both natural language processing and computer vision. The task aims to generate a description in natural language grounded on the input image. It is a very challenging yet interesting task. On the one hand, it has to identify the objects in the image, associate the objects, and express them in a fluent sentence, each of which is a difficult subtask. On the other hand, it combines two important fields in artificial intelligence, namely, natural language processing and computer vision. More importantly, it has a wide range of applications, including text-based image retrieval, helping visually impaired people see (Wu et al., 2017), human-robot interaction (Das et al., 2017), etc.

Soft-Attention: a open laptop computer sitting on top of a table
ATT-FCN: a dog sitting on a desk with a laptop computer and mouse
simNet: a open laptop computer and mouse sitting on a table with a dog nearby

Figure 1: Examples of using different attention mechanisms. Soft-Attention (Xu et al., 2015) is based on visual attention. The generated caption is **detailed** in that it knows the visual attributes well (e.g. *open*). However, it omits many objects (e.g. *mouse* and *dog*). ATT-FCN (You et al., 2016) is based on semantic attention. The generated caption is more **comprehensive** in that it includes more objects. However, it is bad at associating details with the objects (e.g. missing *open* and mislocating *dog*). simNet is our proposal that effectively merges the two kinds of attention and generates a detailed and comprehensive caption.

Models based on the encoder-decoder framework have shown success in image captioning. According to the pivot representation, they can be roughly categorized into models based on visual information (Vinyals et al., 2015; Chen and Zitnick, 2015; Mao et al., 2014; Karpathy and Li, 2015, 2017), and models based on conceptual information (Fang et al., 2015; You et al., 2016; Wu et al., 2016). The later explicitly provides the visual words (e.g. *dog*, *sit*, *red*) to the decoder instead of the image features, and is more effective in image captioning according to the evaluation on benchmark datasets. However, the models based on conceptual information have a major drawback that it is hard for the model to associate the details with the specific objects in the image, because the visual words are inherently unordered in semantics. Figure 1 shows an example. For semantic attention, although *open* is provided as a visual word, due to the insufficient use of visual information, the model gets confused about what objects *open* should be associated with and thus discards *open* in the caption. The model may even associate the details incorrectly, which is the case

---

*Equal Contributions

[1] The code is available at `https://github.com/lancopku/simNet`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 137–149
Brussels, Belgium, October 31 - November 4, 2018.

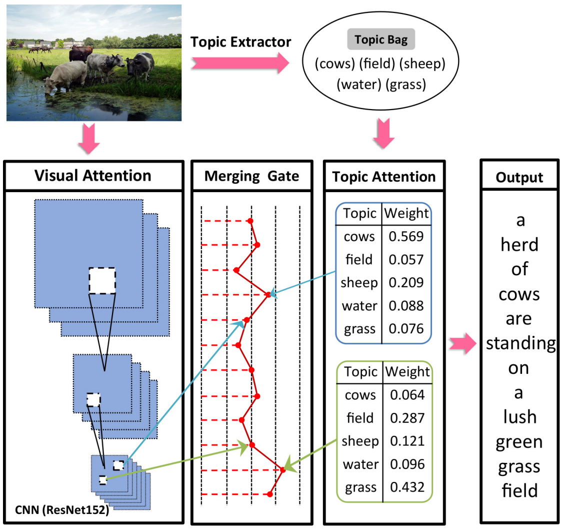

Figure 2: Illustration of the main idea. The visual information captured by CNN and the conceptual information in the extracted topics are first condensed by attention mechanisms respectively. The merging gate then adaptively adjusts the weight between the visual information and the conceptual information for generating the caption.

for the position of the dog. In contrast, models based on the visual information often are accurate in details but have difficulty in describing the image comprehensively and tend to only describe a subregion.

In this work, we get the best of both worlds and integrate visual attention and semantic attention for generating captions that are both detailed and comprehensive. We propose a **Stepwise Image-Topic Merging Network** as the decoder to guide the information flow between the image and the extracted topics. At each time step, the decoder first extracts focal information from the image. Then, it decides which topics are most probable for the time step. Finally, it attends differently to the visual information and the conceptual information to generate the output word. Hence, the model can efficiently merge the two kinds of information, leading to outstanding results in image captioning.

Overall, the main contributions of this work are:

- We propose a novel approach that can effectively merge the information in the image and the topics to generate cohesive captions that are both detailed and comprehensive. We refine and combine two previous competing attention mechanisms, namely visual attention and semantic attention, with an importance-based merging gate that effectively combines and balances the two kinds of information.
- The proposed approach outperforms the state-of-the-art methods substantially on two benchmark datasets, Flickr30k and COCO, in terms of SPICE, which correlates the best with human judgments. Systematic analysis shows that the merging gate contributes the most to the overall improvement.

2 Related Work

A large number of systems have been proposed for image captioning. Neural models based on the encoder-decoder framework have been attracting increased attention in the last few years in several multi-discipline tasks, such as neural image/video captioning (NIC) and visual question answering (VQA) (Vinyals et al., 2015; Karpathy and Li, 2015; Venugopalan et al., 2015; Zhao et al., 2016; Zhang et al., 2017). State-of-the-art neural approaches (Anderson et al., 2018; Liu et al., 2018; Lu et al., 2018) incorporate the attention mechanism in machine translation (Bahdanau et al., 2014) to generate grounded image captions. Based on what they attend to, the models can be categorized into visual attention models and semantic attention models.

Visual attention models pay attention to the image features generated by CNNs. CNNs are typically pre-trained on the image recognition task to extract general visual signals (Xu et al., 2015; Chen et al., 2017; Lu et al., 2017). The visual attention is expected to find the most relevant image regions in generating the caption. Most recently, image features based on predicted bounding boxes are used (Anderson et al., 2018; Lu et al., 2018). The advantages are that the attention no longer needs to find the relevant generic regions by itself but instead find relevant bounding boxes that are object orientated and can serve as semantic guides. However, the drawback is that predicting bounding boxes is difficult, which requires large datasets (Krishna et al., 2017) and complex models (Ren et al., 2015, 2017a).

Semantic attention models pay attention to a predicted set of semantic concepts (Fang et al., 2015; You et al., 2016; Wu et al., 2016). The semantic concepts are the most frequent words in the captions, and the extractor can be trained using various methods but typically is only trained on the given image captioning dataset. This kind

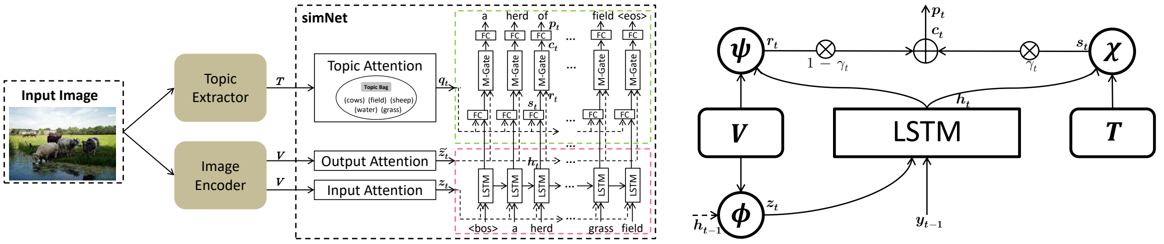

(a) The overall framework.

(b) The data flow in the proposed simNet.

Figure 3: Illustration of the proposed approach. In the right plot, we use ϕ, ψ, χ to denote input attention, output attention, and topic attention, respectively.

of approach can be seen as the extension of the earlier template-based slotting-filling approaches (Farhadi et al., 2010; Kulkarni et al., 2013).

However, few work studies how to combine the two kinds of attention models to take advantage of both of them. On the one hand, due to the limited number of visual features, it is hard to provide comprehensive information to the decoder. On the other hand, the extracted semantic concepts are unordered, making it hard for the decoder to portray the details of the objects correctly.

This work focuses on combining the visual attention and the semantic attention efficiently to address their drawbacks and make use of their merits. The visual attention is designed to focus on the attributes and the relationships of the objects, while the semantic attention only includes words that are objects so that the extracted topics could be more accurate. The combination is controlled by the importance-based merging mechanism that decides at each time step which kind of information should be relied on. The goal is to generate image captions that are both detailed and comprehensive.

3 Approach

Our proposed model consists of an image encoder, a topic extractor, and a stepwise merging decoder. Figure 3 shows a sketch. We first briefly introduce the image encoder and the topic extractor. Then, we introduce the proposed stepwise image-topic merging decoder in detail.

3.1 Image Encoder

For an input image, the image encoder expresses the image as a series of visual feature vectors $\boldsymbol{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k\}, \boldsymbol{v}_i \in \mathbb{R}^g$. Each feature corresponds to a different perspective of the image. The visual features serve as descriptive guides of the objects in the image for the decoder. We use a ResNet152 (He et al., 2016), which is commonly used in image captioning, to generate the visual features. The output of the last convolutional layer is used as the visual information:

$$\boldsymbol{V} = \boldsymbol{W}^{V,I}\mathrm{CNN}(\boldsymbol{I}) \qquad (1)$$

where $\boldsymbol{I}$ is the input image, and $\boldsymbol{W}^{V,I}$ shrinks the last dimension of the output.[2]

3.2 Topic Extractor

Typically, identifying an object requires a combination of visual features, and considering the limited capacity of the visual features, it is hard for the conventional decoder to describe the objects in the image comprehensively. An advance in image captioning is to provide the decoder with the semantic concepts in the image directly so that the decoder is equipped with an overall perspective of the image. The semantic concepts can be objects (e.g. *person*, *car*), attributes (e.g. *off*, *electric*), and relationships (e.g. *using*, *sitting*). We only use the words that are objects in this work, the reason of which is explained later. We call such words **topics**. The topic extractor concludes a list of candidate topic embeddings $\boldsymbol{T} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_m\}, \boldsymbol{w}_i \in \mathbb{R}^e$ from the image, where e is the dimension of the topic word embeddings. Following common practice (Fang et al., 2015; You et al., 2016), we adopt the weakly-supervised approach of Multiple Instance Learning (Zhang et al., 2006) to build a topic extractor. Due to limited space, please refer to Fang et al. (2015) for detailed explanation.

Different from existing work that uses all the most frequent words in the captions as valid semantic concepts or visual words, we only include the object words (nouns) in the topic word list. Existing work relies on attribute words and rela-

---

[2] For conciseness, all the bias terms of linear transformations in this paper are omitted.

tionship words to provide visual information to the decoder. However, it not only complicates the extracting procedure but also contributes little to the generation. For an image containing many objects, the decoder is likely to combine the attributes with the objects arbitrarily, as such words are specific to certain objects but are provided to the decoder unordered. In contrast, our model has visual information as additional input and we expect that the decoder should refer to the image for such kind of information instead of the extracted concepts.

3.3 Stepwise Image-Topic Merging Decoder

The essential component of the decoder is the proposed stepwise image-topic merging network. The decoder is based on an LSTM (Hochreiter and Schmidhuber, 1997). At each time step, it combines the textual caption, the attentive visual information, and the attentive conceptual information as the context for generating an output word. The goal is achieved by three modules, the visual attention, the topic attention, and the merging gate.

Visual Attention as Output The visual attention attends to attracting parts of the image based on the state of the LSTM decoder. In existing work (Xu et al., 2015), only the previous hidden state $\boldsymbol{h}_{t-1} \in \mathbb{R}^d$ of the LSTM is used in computation of the visual attention:

$$\boldsymbol{Z}_t = \tanh(\boldsymbol{W}^{Z,V}\boldsymbol{V} \oplus \boldsymbol{W}^{Z,h}\boldsymbol{h}_{t-1}) \tag{2}$$

$$\boldsymbol{\alpha}_t = \text{softmax}(\boldsymbol{Z}_t\boldsymbol{w}^{\alpha,Z}) \tag{3}$$

where $\boldsymbol{W}^{Z,V} \in \mathbb{R}^{k\times g}, \boldsymbol{W}^{Z,h} \in \mathbb{R}^{k\times d}, \boldsymbol{w}^{\alpha,Z} \in \mathbb{R}^k$ are the learnable parameters. We denote the matrix-vector addition as $\oplus$, which is calculated by adding the vector to each column of the matrix. $\boldsymbol{\alpha}_t \in \mathbb{R}^k$ is the attentive weights of $\boldsymbol{V}$ and the attentive visual input $\boldsymbol{z}_t \in \mathbb{R}^g$ is calculated as

$$\boldsymbol{z}_t = \boldsymbol{V}\boldsymbol{\alpha}_t \tag{4}$$

The visual input $\boldsymbol{z}_t$ and the embedding of the previous output word $\boldsymbol{y}_{t-1}$ are the input of the LSTM.

$$\boldsymbol{h}_t = \text{LSTM}(\begin{bmatrix}\boldsymbol{z}_t\\ \boldsymbol{y}_{t-1}\end{bmatrix}, \boldsymbol{h}_{t-1}) \tag{5}$$

However, there is a noticeable drawback that the previous output word y_{t-1}, which is a much stronger indicator than the previous hidden state $\boldsymbol{h}_{t-1}$, is not used in the attention. As $\boldsymbol{z}_t$ is used as the input, we call it **input attention**. To overcome that drawback, we add another attention that incorporates the current hidden state $\boldsymbol{h}_t$, which is based on the last generated word y_{t-1}:

$$\widetilde{\boldsymbol{Z}}_t = \tanh(\widetilde{\boldsymbol{W}}^{Z,V}\boldsymbol{V} \oplus \widetilde{\boldsymbol{W}}^{Z,h}\boldsymbol{h}_t) \tag{6}$$

$$\widetilde{\boldsymbol{\alpha}}_t = \text{softmax}(\widetilde{\boldsymbol{Z}}_t\widetilde{\boldsymbol{w}}^{\alpha,Z}) \tag{7}$$

$$\widetilde{\boldsymbol{z}}_t = \boldsymbol{V}\widetilde{\boldsymbol{\alpha}}_t \tag{8}$$

The procedure resembles the input attention, and we call it **output attention**. It is worth mentioning that the output attention is essentially the same with the spatial visual attention proposed by Lu et al. (2017). However, they did not see it from the input-output point of view nor combine it with the input attention.

The attentive visual output is further transformed to $\boldsymbol{r}_t = \tanh(\boldsymbol{W}^{s,z}\widetilde{\boldsymbol{z}}_t), \boldsymbol{W}^{s,z} \in \mathbb{R}^{e\times g}$, which is of the same dimension as the topic word embedding to simplify the following procedure.

Topic Attention In an image caption, different parts concern different topics. In the existing work (You et al., 2016), the conceptual information is attended based on the previous output word:

$$\boldsymbol{\beta}_t = \text{softmax}(\boldsymbol{T}^\mathsf{T}\boldsymbol{U}\boldsymbol{y}_{t-1}) \tag{9}$$

where $\boldsymbol{U} \in \mathbb{R}^{e\times e}, \boldsymbol{\beta}_t \in \mathbb{R}^m$. The profound issue is that this approach neglects the visual information. It should be beneficial to provide the attentive visual information when selecting topics. The hidden state of the LSTM contains both the information of previous words and the attentive input visual information. Therefore, the model attends to the topics based on the hidden state of the LSTM:

$$\boldsymbol{Q}_t = \tanh(\boldsymbol{W}^{Q,T}\boldsymbol{T} \oplus \boldsymbol{W}^{Q,h}\boldsymbol{h}_t) \tag{10}$$

$$\boldsymbol{\beta}_t = \text{softmax}(\boldsymbol{Q}_t\boldsymbol{w}^{\beta,Q}) \tag{11}$$

where $\boldsymbol{W}^{Q,T} \in \mathbb{R}^{m\times e}, \boldsymbol{W}^{Q,h} \in \mathbb{R}^{m\times d}, \boldsymbol{w}^{\beta,Q} \in \mathbb{R}^m$ are the parameters to be learned. $\boldsymbol{\beta}_t \in \mathbb{R}^m$ is the weight of the topics, from which the attentive conceptual output $\boldsymbol{q}_t \in \mathbb{R}^e$ is calculated:

$$\boldsymbol{q}_t = \boldsymbol{T}\boldsymbol{\beta}_t \tag{12}$$

The topic attention $\boldsymbol{q}_t$ and the hidden state $\boldsymbol{h}_t$ are combined as the contextual information $\boldsymbol{s}_t$:

$$\boldsymbol{s}_t = \tanh(\boldsymbol{W}^{s,q}\boldsymbol{q}_t + \boldsymbol{W}^{s,h}\boldsymbol{h}_t) \tag{13}$$

where $\boldsymbol{W}^{s,q} \in \mathbb{R}^{e\times e}, \boldsymbol{W}^{s,h} \in \mathbb{R}^{e\times d}$ are learnable parameters.

Merging Gate We have prepared both the visual information $\boldsymbol{r}_t$ and the contextual information $\boldsymbol{s}_t$. It is not reasonable to treat the two kinds of information equally when the decoder generates different types of words. For example, when generating descriptive words (e.g., *behind*, *red*), $\boldsymbol{r}_t$ should matter more than $\boldsymbol{s}_t$. However, when generating object words (e.g., *people*, *table*), $\boldsymbol{s}_t$ is more important. We introduce a novel score-based merging mechanism to make the model adaptively learn to adjust the balance:

$$\gamma_t = \sigma(S(\boldsymbol{s}_t) - S(\boldsymbol{r}_t)) \tag{14}$$

$$\boldsymbol{c}_t = \gamma_t \boldsymbol{s}_t + (1 - \gamma_t)\boldsymbol{r}_t \tag{15}$$

where σ is the sigmoid function, $\gamma_t \in [0, 1]$ indicates how important the topic attention is compared to the visual attention, and S is the scoring function. The scoring function needs to evaluate the importance of the topic attention. Noticing that Eq. (10) and Eq. (11) have a similar purpose, we define S similarly:

$$S(\boldsymbol{s}_t) = \tanh(\boldsymbol{W}^{S,h}\boldsymbol{h}_t + \boldsymbol{W}^{S,s}\boldsymbol{s}_t) \cdot \boldsymbol{w}^S \tag{16}$$

$$S(\boldsymbol{r}_t) = \tanh(\boldsymbol{W}^{S,h}\boldsymbol{h}_t + \boldsymbol{W}^{S,r}\boldsymbol{r}_t) \cdot \boldsymbol{w}^S \tag{17}$$

where $\cdot$ denotes dot product of vectors, $\boldsymbol{W}^{S,s} \in \mathbb{R}^{m \times e}, \boldsymbol{W}^{S,r} \in \mathbb{R}^{m \times e}$ are the parameters to be learned, and $\boldsymbol{W}^{S,h}, \boldsymbol{w}^s$ share the weights of $\boldsymbol{W}^{Q,h}, \boldsymbol{w}^{\beta,Q}$ from Eq. (10) and Eq. (11), respectively.

Finally, the output word is generated by:

$$y_t \sim \boldsymbol{p}_t = \text{softmax}(\boldsymbol{W}^{p,c}\boldsymbol{c}_t) \tag{18}$$

where each value of $\boldsymbol{p}_t \in \mathbb{R}^{|D|}$ is a probability indicating how likely the corresponding word in vocabulary D is the current output word. The whole model is trained using maximum log likelihood and the loss function is the cross entropy loss.

In all, our proposed approach encourages the model to take advantage of all the available information. The adaptive merging mechanism makes the model weigh the information elaborately.

4 Experiment

We describe the datasets and the metrics used for evaluation, followed by the training details and the evaluation of the proposed approach.

4.1 Datasets and Metrics

There are several datasets containing images and their captions. We report results on the popular Microsoft COCO (Chen et al., 2015) dataset and the Flickr30k (Young et al., 2014) dataset. They contain 123,287 images and 31,000 images, respectively, and each image is annotated with 5 sentences. We report results using the widely-used publicly-available splits in the work of Karpathy and Li (2015). There are 5,000 images each in the validation set and the test set for COCO, 1,000 images for Flickr30k.

We report results using the COCO captioning evaluation toolkit (Chen et al., 2015) that reports the widely-used automatic evaluation metrics SPICE, CIDEr, BLEU, METEOR, and ROUGE. SPICE (Anderson et al., 2016), which is based on scene graph matching, and CIDEr (Vedantam et al., 2015), which is based on n-gram matching, are specifically proposed for evaluating image captioning systems. They both incorporate the consensus of a set of references for an example. BLEU (Papineni et al., 2002) and METOR (Banerjee and Lavie, 2005) are originally proposed for machine translation evaluation. ROUGE (Lin and Hovy, 2003; Lin, 2004) is designed for automatic evaluation of extractive text summarization. In the related studies, it is concluded that SPICE correlates the best with human judgments with a remarkable margin over the other metrics, and is expert in judging detailedness, where the other metrics show negative correlations, surprisingly; CIDEr and METEOR follows with no particular precedence, followed by ROUGE-L, and BLEU-4, in that order (Anderson et al., 2016; Vedantam et al., 2015).

4.2 Settings

Following common practice, the CNN used is the ResNet152 model (He et al., 2016) pre-trained on ImageNet.[3] There are 2048 7 × 7 feature maps, and we project them into 512 feature maps, i.e. g is 512. The word embedding size e is 256 and the hidden size d of the LSTM is 512. We only keep caption words that occur at least 5 times in the training set, resulting in 10,132 words for COCO and 7,544 for Flickr30k. We use the topic extractor pre-trained by Fang et al. (2015) for 1,000 concepts on COCO. We only use 568 manually-annotated object words as topics. For an image, only the top 5 topics are selected, which means m is 5. The same topic extractor is used for Flickr30k, as COCO provides adequate generality. The caption words and the topic words share the same embeddings. In training, we first train the model without visual attention (freezing the CNN parameters) for 20 epochs with the batch size of 80. The learning rate for the LSTM is 0.0004. Then, we switch to jointly train the full model with a learning rate of 0.00001, which exponentially decays with the number of epochs so that it is halved every 50 epochs. We also use momen-

---

[3] We use the pre-trained model from `torchvision`.

Flickr30k	SPICE	CIDEr	METEOR	ROUGE-L	BLEU-4
HardAtt (Xu et al., 2015)	-	-	0.185	-	0.199
SCA-CNN (Chen et al., 2017)	-	-	0.195	-	0.223
ATT-FCN (You et al., 2016)	-	-	0.189	-	0.230
SCN-LSTM (Gan et al., 2017)	-	-	0.210	-	0.257
AdaAtt (Lu et al., 2017)	0.145	0.531	0.204	0.467	0.251
NBT (Lu et al., 2018)	0.156	0.575	0.217	-	0.271
SR-PL (Liu et al., 2018)*†	0.158	**0.650**	0.218	**0.499**	**0.293**
simNet	**0.160**	0.585	**0.221**	0.489	0.251

Table 1: Performance on the Flickr30k Karpathy test split. The symbol * denotes directly optimizing CIDEr. The symbol † denotes using extra data for training, thus not directly comparable. Nonetheless, our model supersedes all existing models in SPICE, which correlates the best with human judgments.

COCO	SPICE	CIDEr	METEOR	ROUGE-L	BLEU-4
HardAtt (Xu et al., 2015)	-	-	0.230	-	0.250
ATT-FCN (You et al., 2016)	-	-	0.243	-	0.304
SCA-CNN (Chen et al., 2017)	-	0.952	0.250	0.531	0.311
LSTM-A (Yao et al., 2017)	0.186	1.002	0.254	0.540	0.326
SCN-LSTM (Gan et al., 2017)	-	1.012	0.257	-	0.330
Skeleton (Wang et al., 2017)	-	1.069	0.268	0.552	0.336
AdaAtt (Lu et al., 2017)	0.195	1.085	0.266	0.549	0.332
NBT (Lu et al., 2018)	0.201	1.072	0.271	-	0.347
DRL (Ren et al., 2017b)*	-	0.937	0.251	0.525	0.304
TD-M-ATT (Chen et al., 2018)*	-	1.116	0.268	0.555	0.336
SCST (Rennie et al., 2017)*	-	1.140	0.267	0.557	0.342
SR-PL (Liu et al., 2018)*†	0.210	1.171	0.274	**0.570**	0.358
Up-Down (Anderson et al., 2018)*†	0.214	**1.201**	0.277	0.569	**0.363**
simNet	**0.220**	1.135	**0.283**	0.564	0.332

Table 2: Performance on the COCO Karpathy test split. Symbols, * and †, are defined similarly. Our model outperforms the current state-of-the-art Up-Down substantially in terms of SPICE.

tum of 0.8 and weight decay of 0.999. We use Adam (Kingma and Ba, 2014) for parameter optimization. For fair comparison, we adopt early stop based on CIDEr within maximum 50 epochs.

4.3 Results

We compare our approach with various representative systems on Flickr30k and COCO, including the recently proposed NBT that is the state-of-the-art on the two datasets in comparable settings. Table 1 shows the result on Flickr30k. As we can see, our model outperforms the comparable systems in terms of all of the metrics except BLEU-4. Moreover, our model overpasses the state-of-the-art with a comfortable margin in terms of SPICE, which is shown to correlate the best with human judgments (Anderson et al., 2016).

Table 2 shows the results on COCO. Among the directly comparable models, our model is arguably the best and outperforms the existing models except in terms of BLEU-4. Most encouragingly, our model is also competitive with Up-Down (Anderson et al., 2018), which uses much larger dataset, Visual Genome (Krishna et al., 2017), with dense annotations to train the object detector, and directly optimizes CIDEr. Especially, our model outperforms the state-of-the-art substantially in SPICE and METEOR. Breakdown of SPICE F-scores over various subcategories (see Table 3) shows that our model is in dominant lead in almost all subcategories. It proves the effectiveness of our approach and indicates that our model is quite data efficient.

For the methods that directly optimize CIDEr, it is intuitive that CIDEr can improve significantly. The similar improvement of BLEU-4 is evidence that optimizing CIDEr leads to more n-gram matching. However, it comes to our notice that the improvements of SPICE, METEOR, and ROUGE-L are far less significant, which suggests there may be a gaming situation where the n-gram matching is wrongfully exploited by the model in reinforcement learning. As shown by Liu et al. (2017), it is most reasonable to jointly optimize

Methods	SPICE							CIDEr	METEOR	ROUGE-L	BLEU-4
	All	Objects	Attributes	Relations	Color	Count	Size				
Baseline (Plain Encoder-Decoder Network)	0.150	0.295	0.048	0.039	0.022	0.004	0.023	0.762	0.220	0.495	0.251
Up-Down (Anderson et al., 2018)*†	0.214	0.391	0.100	0.065	0.114	0.184	0.032	**1.201**	0.277	**0.569**	**0.363**
Baseline + Input Att.	0.164	0.316	0.060	0.044	0.030	0.038	0.024	0.840	0.233	0.512	0.273
Baseline + Output Att.	0.181	0.329	0.094	0.053	0.089	0.184	0.044	0.968	0.253	0.534	0.301
Baseline + Input Att. + Output Att.	0.187	0.338	0.101	0.055	**0.115**	0.161	**0.048**	1.038	0.259	0.542	0.311
Baseline + Topic Att.	0.184	0.348	0.074	0.051	0.047	0.064	0.037	0.915	0.250	0.517	0.260
Baseline + Topic Att. + MGate	0.189	0.355	0.080	0.051	0.055	0.090	0.033	0.959	0.256	0.527	0.281
Baseline + Input Att. + Output Att. + Topic Att.	0.206	0.381	0.091	0.060	0.075	0.094	0.045	1.068	0.273	0.556	0.320
simNet (Full Model)	**0.220**	**0.394**	**0.109**	**0.070**	0.088	**0.202**	0.045	1.135	**0.283**	0.564	0.332

Table 3: Results of incremental analysis. For a better understanding of the differences, we further list the breakdown of SPICE F-scores. *Objects* indicates comprehensiveness, and the others indicate detailedness. Additionally, we report the performance of the current state-of-the-art Up-Down for further comparison, which uses extra dense-annotated data for pre-training and directly optimizes CIDEr.

Method	Precision	Recall	F1
Topics (m=5)	49.95	38.91	42.48
All words (m=5)	**84.01**	17.99	29.49
All words (m=10)	70.90	30.18	42.05
All words (m=20)	52.51	**44.53**	**47.80**

Table 4: Performance of visual word extraction.

Method	S	C	M	R	B
Topics (m=5)	**0.220**	**1.135**	**0.283**	**0.564**	**0.332**
All words (m=5)	0.197	1.047	0.264	0.550	0.314
All words (m=10)	0.201	1.076	0.256	0.528	0.293
All words (m=20)	0.209	1.117	0.276	0.561	0.329

Table 5: Effect of using different visual words.

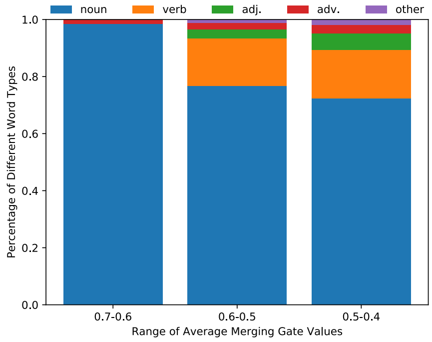

Figure 4: Average merging gate values according to word types. As we can see, object words (noun) dominate the high value range, while attribute and relation words are assigned lower values, indicating the merging gate learns to efficiently combine the information.

all the metrics at the same time.

We also evaluate the proposed model on the COCO evaluation server, the results of which are shown in Appendix A.1, due to limited space.

5 Analysis

In this section, we analyze the contribution of each component in the proposed approach, and give examples to show the strength and the potential improvements of the model. The analysis is conducted on the test set of COCO.

Topic Extraction The motivation of using objects as topics is that they are easier to identify so that the generation suffers less from erroneous predictions. This can be proved by the F-score of the identified topics in the test set, which is shown in Table 4. Using top-5 object words is at least as good as using top-10 all words. However, using top-10 all words introduces more erroneous visual words to the generation. As shown in Table 5, when extracting all words, providing more words to the model indeed increases the captioning performance. However, even when top-20 all words are used, the performance is still far behind using only top-5 object words and seems to reach the performance ceiling. It proves that for semantic attention, it is also important to limit the absolute number of incorrect visual words instead of merely the precision or the recall. It is also interesting to check whether using other kind of words can reach the same effect. Unfortunately, in our experiments, only using verbs or adjectives as semantic concepts works poorly.

To examine the contributions of the sub-modules in our model, we conduct a series of experiments. The results are summarized in Table 3. To help with the understanding of the differences, we also report the breakdown of SPICE F-scores.

Visual Attention Our input attention achieves similar results to previous work (Xu et al., 2015),

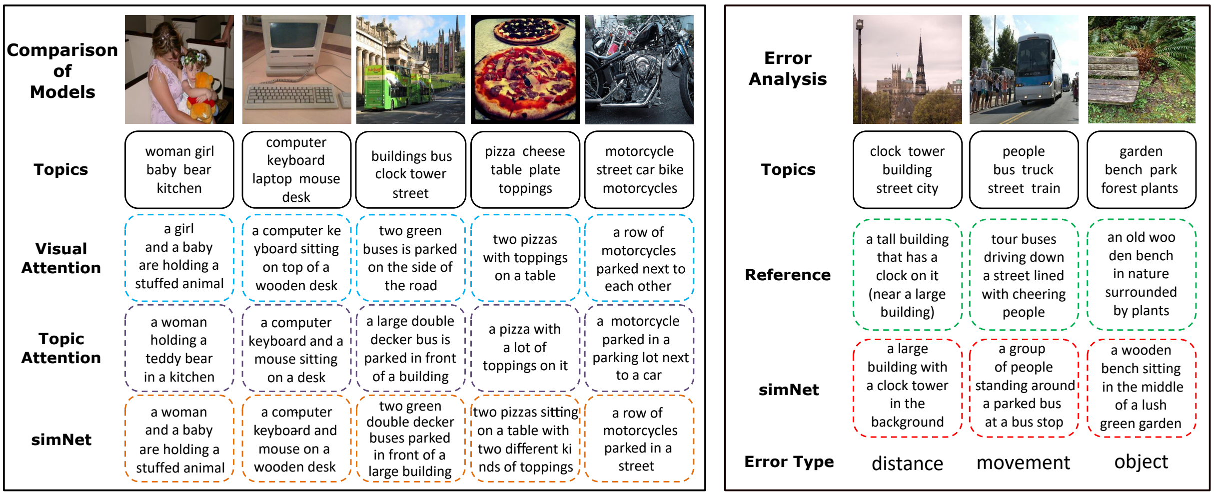

Figure 5: Examples of the generated captions. The left plot compares simNet with visual attention and topic attention. Visual attention is good at portraying the relations but is less specific in objects. Topic attention includes more objects but lacks details, such as material, color, and number. The proposed model achieves a very good balance. The right plot shows the error analysis of the proposed simNet.

if not better. Using only the output attention is much more effective than using only the input attention, with substantial improvements in all metrics, showing the impact of information gap caused by delayed input in attention. Combining the input attention and the output attention can further improve the results, especially in color and size descriptions.

Topic Attention As expected, compared with visual attention, the topic attention is better at identifying objects but worse at identifying attributes. We also apply the merging gate to the topic attention, but it now merges $\boldsymbol{q}_t$ and $\boldsymbol{h}_t$ instead of $\boldsymbol{s}_t$ and $\boldsymbol{r}_t$. With the merging gate, the model can balance the information in caption text and extracted topics, resulting in better overall scores. While it overpasses the conventional visual attention, it lags behind the output attention.

Merging Gate Combing the visual attention and the topic attention directly indeed results in a huge boost in performance, which confirms our motivation. However, directly combining them also causes lower scores in attributes, color, count, and size, showing that the advantages are not fully made use of. The most dramatic improvements come from applying the merging gate to the combined attention, showing that the proposed balance mechanism can adaptively combine the two kinds of information and is essential to the overall performance. The average merging gate value summarized in Figure 4 suggests the same.

We give some examples in the left plot of Figure 5 to illustrate the differences between the models more intuitively. From the examples, it is clear that the proposed simNet generates the best captions in that more objects are described and many informative and detailed attributes are included, such as the quantity and the color.

Visualization Figure 6 shows the visualization of the topic attention and the visual attention with running examples. As we can see, the topic attention is active when generating a phrase containing the related topic. For example, *bathroom* is always most attended when generating *a bathroom*. The merging gate learns to direct the information flow efficiently. When generating words such as *on* and *a*, it gives lower weight to the topic attention and prefers the visual attention. As to the visual attention, the output attention is much more focused than the input attention. As we hypothesized, the conventional input attention lacks the information of the last generated word and does not know what to look for exactly. For example, when generating *bathroom*, the input attention does not know the previous generated word is *a*, and it loses its focus, while the output attention is relatively more concentrated. Moreover, the merging gate learns to overcome the erroneous topics, as shown in the second example. When generating *chair*, the topic attention is focused on a wrong object *bed*, while the visual attention attends correctly to the chair, and especially the output attention attends to the armrest. The merging gate effectively remedies

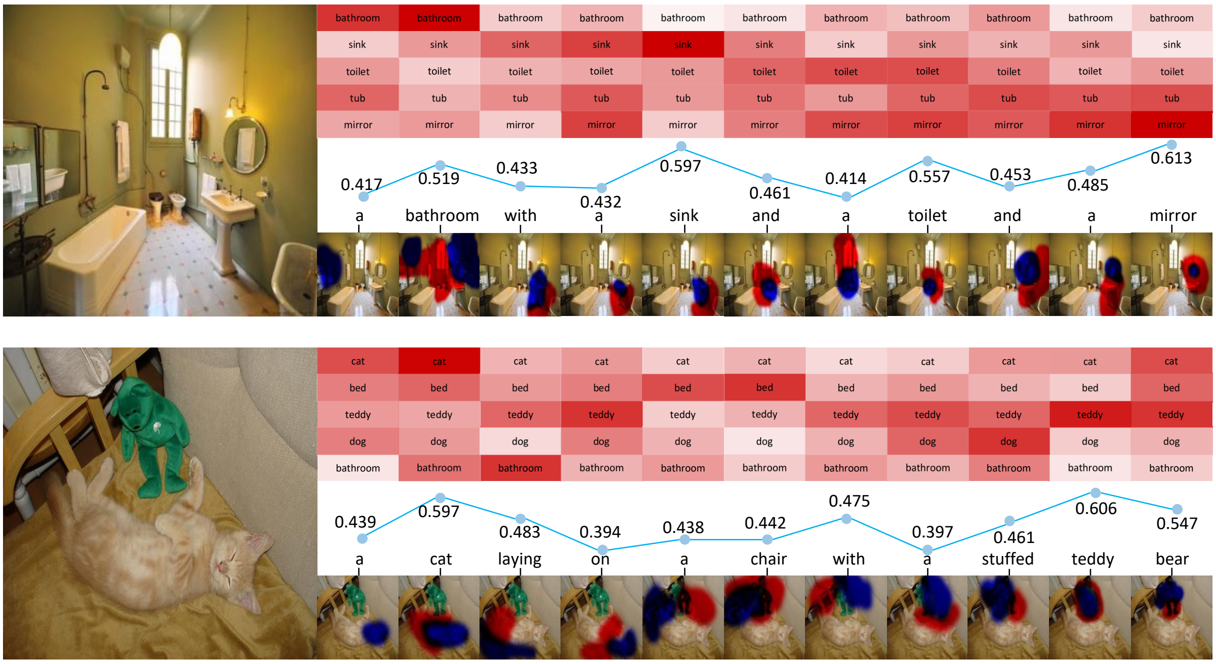

Figure 6: Visualization. Please view in color. Here, we give two running examples. The upper part of each example shows the attention weights of each of 5 extracted topics. Deeper color means larger in value. The middle part shows the value of the merging gate that determines the importance of the topic attention. The lower part shows the visualization of visual attention. The attended region is covered with color. The blue shade indicates the output attention. The red shade indicates the input attention.

the misleading information from the topic attention and outputs a lower weight, resulting in the model correctly generating the word *chair*.

Error Analysis We conduct error analysis using the proposed (full) model on the test set to provide insights on how the model may be improved. We find 123 out of 1000 generated captions that are not satisfactory. There are mainly three types of errors, i.e. distance (32, 26%), movement (22, 18%), and object (60, 49%), with 9 (7%) other errors. Distance error takes place when there is a lot of objects and the model cannot grasp the foreground and the background relationship. Movement error means that the model fails to describe whether the objects are moving. Those two kinds of errors are hard to eliminate, as they are fundamental problems of computer vision waiting to be resolved. Object error happens when there are incorrect extracted topics, and the merging gate regards the topic as grounded in the image. In the given example, the incorrect topic is *garden*. The tricky part is that the topic is seemingly correct according to the image features or otherwise the proposed model will choose other topics. A more powerful topic extractor may help with the problem but it is unlikely to be completely avoided.

6 Conclusions

We propose the stepwise image-topic merging network to sequentially and adaptively merge the visual and the conceptual information for improved image captioning. To our knowledge, we are the first to combine the visual and the semantic attention to achieve substantial improvements. We introduce the stepwise merging mechanism to efficiently guide the two kinds of information when generating the caption. The experimental results demonstrate the effectiveness of the proposed approach, which substantially outperforms the state-of-the-art image captioning methods in terms of SPICE on COCO and Flickr30k datasets. Quantitative and qualitative analysis show that the generated captions are both detailed and comprehensive in comparison with the existing methods.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (No. 61673028). We thank all the anonymous reviewers for their constructive comments and suggestions. Xu Sun is the corresponding author of this paper.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: semantic propositional image caption evaluation. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V*, volume 9909 of *Lecture Notes in Computer Science*, pages 382–398. Springer.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and VQA. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, June 29, 2005*, pages 65–72. Association for Computational Linguistics.

Hui Chen, Guiguang Ding, Sicheng Zhao, and Jungong Han. 2018. Temporal-difference learning with sampling baseline for image captioning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press.

Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. 2017. SCA-CNN: spatial and channel-wise attention in convolutional networks for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6298–6306. IEEE Computer Society.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325.

Xinlei Chen and C. Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 2422–2431. IEEE Computer Society.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1080–1089. IEEE Computer Society.

Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Kumar Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1473–1482. IEEE Computer Society.

Ali Farhadi, Seyyed Mohammad Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David A. Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV*, volume 6314 of *Lecture Notes in Computer Science*, pages 15–29. Springer.

Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2017. Semantic compositional networks for visual captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1141–1150. IEEE Computer Society.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

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

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3128–3137. IEEE Computer Society.

Andrej Karpathy and Fei-Fei Li. 2017. Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):664–676.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2013. BabyTalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 35(12):2891–2903.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop, Barcelona, Spain, July, 2004*, pages 74–81. Association for Computational Linguistics.

Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram cooccurrence statistics. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics.

Junyang Lin, Xu Sun, Xuancheng Ren, Shuming Ma, Jinsong Su, and Qi Su. 2018. Deconvolution-based global decoding for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3260–3271. Association for Computational Linguistics.

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2017. Improved image captioning via policy gradient optimization of spider. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 873–881. IEEE Computer Society.

Xihui Liu, Hongsheng Li, Jing Shao, Dapeng Chen, and Xiaogang Wang. 2018. Show, tell and discriminate: Image captioning by self-retrieval with partially labeled data. *CoRR*, abs/1803.08314.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3242–3250. IEEE Computer Society.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2018. Neural baby talk. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*.

Shuming Ma, Xu Sun, Junyang Lin, and Xuancheng Ren. 2018. A hierarchical end-to-end model for jointly improving text summarization and sentiment classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4251–4257. ijcai.org.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. 2014. Deep captioning with multimodal recurrent neural networks (m-RNN). *CoRR*, abs/1412.6632.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318. ACL.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2017a. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.

Zhou Ren, Xiaoyu Wang, Ning Zhang, Xutao Lv, and Li-Jia Li. 2017b. Deep reinforcement learning-based image captioning with embedding reward. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1151–1159. IEEE Computer Society.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1179–1195. IEEE Computer Society.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: consensus-based image description evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4566–4575. IEEE Computer Society.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence - video to text. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 4534–4542. IEEE Computer Society.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3156–3164. IEEE Computer Society.

Yufei Wang, Zhe Lin, Xiaohui Shen, Scott Cohen, and Garrison W. Cottrell. 2017. Skeleton key: Image captioning by skeleton-attribute decomposition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 7378–7387. IEEE Computer Society.

Qi Wu, Chunhua Shen, Lingqiao Liu, Anthony R. Dick, and Anton van den Hengel. 2016. What value do explicit high level concepts have in vision to language problems? In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 203–212. IEEE Computer Society.

Shaomei Wu, Jeffrey Wieland, Omid Farivar, and Julie Schiller. 2017. Automatic alt-text: Computer-generated image descriptions for blind users on a social network service. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW 2017, Portland, OR, USA, February 25 - March 1, 2017*, pages 1180–1192. ACM.

Jingjing Xu, Xu Sun, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018a. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 979–988. Association for Computational Linguistics.

Jingjing Xu, Yi Zhang, Qi Zeng, Xuancheng Ren, Xiaoyan Cai, and Xu Sun. 2018b. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, Brussels, Belgium, October 31-November 4, 2018*. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.

Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. 2017. Boosting image captioning with attributes. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 4904–4912. IEEE Computer Society.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4651–4659. IEEE Computer Society.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Cha Zhang, John C. Platt, and Paul A. Viola. 2006. Multiple instance boosting for object detection. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 1417–1424. MIT Press.

Hanwang Zhang, Zawlin Kyaw, Shih-Fu Chang, and Tat-Seng Chua. 2017. Visual translation embedding network for visual relation detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3107–3115. IEEE Computer Society.

Zhou Zhao, Hanqing Lu, Deng Cai, Xiaofei He, and Yueting Zhuang. 2016. Partial multi-modal sparse coding via adaptive similarity structure regularization. In *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19, 2016*, pages 152–156. ACM.

COCO	BLEU-1		BLEU-2		BLEU-3		BLEU-4		METEOR		ROUGE-L		CIDEr	
	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40
HardAtt (Xu et al., 2015)	0.705	0.881	0.528	0.779	0.383	0.658	0.277	0.537	0.241	0.322	0.516	0.654	0.865	0.893
ATT-FCN (You et al., 2016)	0.731	0.900	0.565	0.815	0.424	0.709	0.316	0.599	0.250	0.335	0.535	0.682	0.943	0.958
SCA-CNN (Chen et al., 2017)	0.712	0.894	0.542	0.802	0.404	0.691	0.302	0.579	0.244	0.331	0.524	0.674	0.912	0.921
LSTM-A (Yao et al., 2017)	0.739	0.919	0.575	0.842	0.436	0.740	0.330	0.632	0.256	0.350	0.542	0.700	0.984	1.003
SCN-LSTM (Gan et al., 2017)	0.740	0.917	0.575	0.839	0.436	0.739	0.331	0.631	0.257	0.348	0.543	0.696	1.003	1.013
AdaAtt (Lu et al., 2017)†	0.748	0.920	0.584	0.845	0.444	0.744	0.336	0.637	0.264	0.359	0.550	0.705	1.042	1.059
TD-M-ATT (Chen et al., 2018)*†	0.757	0.913	0.591	0.836	0.441	0.726	0.324	0.609	0.259	0.342	0.547	0.689	1.059	1.090
SCST (Rennie et al., 2017)*†	0.781	0.937	0.619	0.860	0.470	0.759	0.352	0.645	0.270	0.355	0.563	0.707	1.147	1.167
Up-Down (Anderson et al., 2018)*†‡	**0.802**	**0.952**	**0.641**	**0.888**	**0.491**	**0.794**	**0.369**	**0.685**	**0.276**	**0.367**	**0.571**	**0.724**	**1.179**	**1.205**
simNet	0.766	0.941	0.605	0.874	0.462	0.778	0.350	0.671	0.267	0.362	0.558	0.716	1.087	1.111

Table 6: Performance on the online COCO evaluation server. The SPICE metric is unavailable for our model, thus not reported. c5 means evaluating against 5 references, and c40 means evaluating against 40 references. The symbol * denotes directly optimizing CIDEr. The symbol † denotes model ensemble. The symbol ‡ denotes using extra data for training, thus not directly comparable. Our submission does not use the three aforementioned techniques. Nonetheless, our model is second only to Up-Down and surpasses almost all the other models in published work, especially when 40 references are considered.

A Supplementary Material

A.1 Results on COCO Evaluation Server

Table 6 shows the performance on the online COCO evaluation server[4]. We put it in the appendix because the results are incomplete and the SPICE metric is not available for our submission, which correlates the best with human evaluation. The SPICE metrics are only available at the leaderboard on the COCO dataset website[5], which, unfortunately, has not been updated for more than a year. Our submission does not directly optimize CIDEr, use model ensemble, or use extra training data. The three techniques typically result in orthogonal improvements (Lu et al., 2017; Rennie et al., 2017; Anderson et al., 2018). Moreover, the SPICE results are missing, in which the proposed model has the most advantage. Nonetheless, our model is second only to Up-Down (Anderson et al., 2018) and surpasses almost all the other models in published work, especially when 40 references are considered.

---

[4] `https://competitions.codalab.org/competitions/3221`

[5] `http://cocodataset.org/#captions-leaderboard`

Multimodal Language Analysis with Recurrent Multistage Fusion

Paul Pu Liang[1], **Ziyin Liu**[2], **Amir Zadeh**[2], **Louis-Philippe Morency**[2]
[1]Machine Learning Department, [2]Language Technologies Institute
Carnegie Mellon University
`{pliang,ziyinl,abagherz,morency}@cs.cmu.edu`

Abstract

Computational modeling of human multimodal language is an emerging research area in natural language processing spanning the language, visual and acoustic modalities. Comprehending multimodal language requires modeling not only the interactions within each modality (intra-modal interactions) but more importantly the interactions between modalities (cross-modal interactions). In this paper, we propose the Recurrent Multistage Fusion Network (RMFN) which decomposes the fusion problem into multiple stages, each of them focused on a subset of multimodal signals for specialized, effective fusion. Cross-modal interactions are modeled using this multistage fusion approach which builds upon intermediate representations of previous stages. Temporal and intra-modal interactions are modeled by integrating our proposed fusion approach with a system of recurrent neural networks. The RMFN displays state-of-the-art performance in modeling human multimodal language across three public datasets relating to multimodal sentiment analysis, emotion recognition, and speaker traits recognition. We provide visualizations to show that each stage of fusion focuses on a different subset of multimodal signals, learning increasingly discriminative multimodal representations.

1 Introduction

Computational modeling of human multimodal language is an upcoming research area in natural language processing. This research area focuses on modeling tasks such as multimodal sentiment analysis (Morency et al., 2011), emotion recognition (Busso et al., 2008), and personality traits recognition (Park et al., 2014). The multimodal temporal signals include the language (spoken words), visual (facial expressions, gestures) and acoustic modalities (prosody, vocal expressions). At its core, these multimodal signals are highly structured with two prime forms of interactions: intra-modal and cross-modal interactions (Rajagopalan et al., 2016). Intra-modal interactions refer to information within a specific modality, independent of other modalities. For example, the arrangement of words in a sentence according

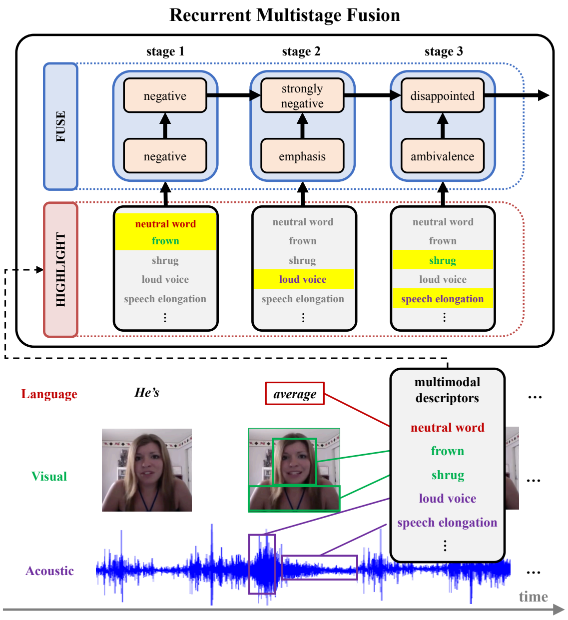

Figure 1: An illustrative example for Recurrent Multistage Fusion. At each recursive stage, a subset of multimodal signals is highlighted and then fused with previous fusion representations. The first fusion stage selects the neutral word and frowning behaviors which create an intermediate representation reflecting negative emotion when fused together. The second stage selects the loud voice behavior which is locally interpreted as emphasis before being fused with previous stages into a strongly negative representation. Finally, the third stage selects the shrugging and speech elongation behaviors that reflect ambivalence and when fused with previous stages is interpreted as a representation for the disappointed emotion.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 150–161
Brussels, Belgium, October 31 - November 4, 2018.

to the generative grammar of a language (Chomsky, 1957) or the sequence of facial muscle activations for the presentation of a frown. Cross-modal interactions refer to interactions between modalities. For example, the simultaneous co-occurrence of a smile with a positive sentence or the delayed occurrence of a laughter after the end of a sentence. Modeling these interactions lie at the heart of human multimodal language analysis and has recently become a centric research direction in multimodal natural language processing (Liu et al., 2018; Pham et al., 2018; Chen et al., 2017), multimodal speech recognition (Sun et al., 2016; Gupta et al., 2017; Harwath and Glass, 2017; Kamper et al., 2017), as well as multimodal machine learning (Tsai et al., 2018; Srivastava and Salakhutdinov, 2012; Ngiam et al., 2011).

Recent advances in cognitive neuroscience have demonstrated the existence of multistage aggregation across human cortical networks and functions (Taylor et al., 2015), particularly during the integration of multisensory information (Parisi et al., 2017). At later stages of cognitive processing, higher level semantic meaning is extracted from phrases, facial expressions, and tone of voice, eventually leading to the formation of higher level cross-modal concepts (Parisi et al., 2017; Taylor et al., 2015). Inspired by these discoveries, we hypothesize that the computational modeling of cross-modal interactions also requires a *multistage fusion* process. In this process, cross-modal representations can build upon the representations learned during earlier stages. This decreases the burden on each stage of multimodal fusion and allows each stage of fusion to be performed in a more specialized and effective manner.

In this paper, we propose the Recurrent Multistage Fusion Network (RMFN) which automatically decomposes the multimodal fusion problem into multiple recursive stages. At each stage, a subset of multimodal signals is highlighted and fused with previous fusion representations (see Figure 1). This divide-and-conquer approach decreases the burden on each fusion stage, allowing each stage to be performed in a more specialized and effective way. This is in contrast with conventional fusion approaches which usually model interactions over multimodal signals altogether in one iteration (e.g., early fusion (Baltrušaitis et al., 2017)). In RMFN, temporal and intra-modal interactions are modeled by integrating our new multistage fusion process with a system of recurrent neural networks. Overall, RMFN jointly models intra-modal and cross-modal interactions for multimodal language analysis and is differentiable end-to-end.

We evaluate RMFN on three different tasks related to human multimodal language: sentiment analysis, emotion recognition, and speaker traits recognition across three public multimodal datasets. RMFN achieves state-of-the-art performance in all three tasks. Through a comprehensive set of ablation experiments and visualizations, we demonstrate the advantages of explicitly defining multiple recursive stages for multimodal fusion.

2 Related Work

Previous approaches in human multimodal language modeling can be categorized as follows:

Non-temporal Models: These models simplify the problem by using feature-summarizing temporal observations (Poria et al., 2017). Each modality is represented by averaging temporal information through time, as shown for language-based sentiment analysis (Iyyer et al., 2015; Chen et al., 2016) and multimodal sentiment analysis (Abburi et al., 2016; Nojavanasghari et al., 2016; Zadeh et al., 2016; Morency et al., 2011). Conventional supervised learning methods are utilized to discover intra-modal and cross-modal interactions without specific model design (Wang et al., 2016; Poria et al., 2016). These approaches have trouble modeling long sequences since the average statistics do not properly capture the temporal intra-modal and cross-modal dynamics (Xu et al., 2013).

Multimodal Temporal Graphical Models: The application of graphical models in sequence modeling has been an important research problem. Hidden Markov Models (HMMs) (Baum and Petrie, 1966), Conditional Random Fields (CRFs) (Lafferty et al., 2001), and Hidden Conditional Random Fields (HCRFs) (Quattoni et al., 2007) were shown to work well on modeling sequential data from the language (Misawa et al., 2017; Ma and Hovy, 2016; Huang et al., 2015) and acoustic (Yuan and Liberman, 2008) modalities. These temporal graphical models have also been extended for modeling multimodal data. Several methods have been proposed including multi-view HCRFs where the potentials of the HCRF are designed to model data from multiple views (Song et al., 2012), multi-layered CRFs with latent variables to learn hidden spatio-temporal dynamics from multi-view data (Song et al., 2012), and multi-view Hierarchical Sequence Summarization models that recursively build up hierarchical representations (Song et al., 2013).

Multimodal Temporal Neural Networks: More recently, with the advent of deep learning, Recurrent Neural Networks (Elman, 1990; Jain and Medsker, 1999) have been used extensively for language and speech based sequence modeling (Zilly et al., 2016; Soltau et al., 2016), sentiment analysis (Socher et al., 2013; dos Santos and Gatti, 2014; Glorot et al., 2011; Cambria, 2016), and emotion recognition (Han et al., 2014; Bertero et al., 2016; Lakomkin et al., 2018). Long-short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997a) have also been extended for multimodal settings (Rajagopalan et al., 2016) and by learning binary gating mechanisms to remove noisy modalities (Chen et al., 2017). Recently, more advanced models were proposed to model both intra-modal and cross-modal interactions. These use Bayesian ranking algorithms (Herbrich et al., 2007) to model both person-independent and person-dependent features (Liang et al., 2018), generative-discriminative objectives to learn either joint (Pham et al., 2018) or factorized multimodal representations (Tsai et al., 2018), external memory mechanisms to synchronize multimodal data (Zadeh et al., 2018a), or low-rank tensors to approximate expensive tensor products (Liu et al., 2018). All these methods assume that cross-modal interactions should be discovered all at once rather than across multiple stages, where each stage solves a simpler fusion problem. Our empirical evaluations show the advantages of the multistage fusion approach.

3 Recurrent Multistage Fusion Network

In this section we describe the Recurrent Multistage Fusion Network (RMFN) for multimodal language analysis (Figure 2). Given a set of modalities $\{l(anguage), v(isual), a(coustic)\}$, the signal from each modality $m \in \{l, v, a\}$ is represented as a temporal sequence $\mathbf{X}^m = \{\mathbf{x}_1^m, \mathbf{x}_2^m, \mathbf{x}_3^m, \cdots, \mathbf{x}_T^m\}$, where $\mathbf{x}_t^m$ is the input at time t. Each sequence $\mathbf{X}^m$ is modeled with an intra-modal recurrent neural network (see subsection 3.3 for details). At time t, each intra-modal recurrent network will output a unimodal representation $\mathbf{h}_t^m$. The Multistage Fusion Process uses a recursive approach to fuse all unimodal representations $\mathbf{h}_t^m$ into a cross-modal representation $\mathbf{z}_t$ which is then fed back into each intra-modal recurrent network.

3.1 Multistage Fusion Process

The Multistage Fusion Process (MFP) is a modular neural approach that performs multistage fusion to model cross-modal interactions. Multistage fusion is a divide-and-conquer approach which decreases the burden on each stage of multimodal fusion, allowing each stage to be performed in a more specialized and effective way. The MFP has three main modules: `HIGHLIGHT`, `FUSE` and `SUMMARIZE`.

Two modules are repeated at each stage: `HIGHLIGHT` and `FUSE`. The `HIGHLIGHT` module identifies a subset of multimodal signals from $[\mathbf{h}_t^l, \mathbf{h}_t^v, \mathbf{h}_t^a]$ that will be used for that stage of fusion. The `FUSE` module then performs two subtasks simultaneously: a local fusion of the highlighted features and integration with representations from previous stages. Both `HIGHLIGHT` and `FUSE` modules are realized using memory-based neural networks which enable coherence between stages and storage of previously modeled cross-modal interactions. As a final step, the `SUMMARIZE` module takes the multimodal representation of the final stage and translates it into a cross-modal representation $\mathbf{z}_t$.

Figure 1 shows an illustrative example for multistage fusion. The `HIGHLIGHT` module selects "neutral words" and "frowning" expression for the first stage. The local and integrated fusion at this stage creates a representation reflecting negative emotion. For stage 2, the `HIGHLIGHT` module identifies the acoustic feature "loud voice". The local fusion at this stage interprets it as an expression of emphasis and is fused with the previous fusion results to represent a strong negative emotion. Finally, the highlighted features of "shrug" and "speech elongation" are selected and are locally interpreted as "ambivalence". The integration with previous stages then gives a representation closer to "disappointed".

3.2 Module Descriptions

In this section, we present the details of the three multistage fusion modules: `HIGHLIGHT`, `FUSE` and `SUMMARIZE`. Multistage fusion begins with the concatenation of intra-modal network outputs $\mathbf{h}_t = \bigoplus_{m \in M} \mathbf{h}_t^m$. We use superscript $^{[k]}$ to denote the indices of each stage $k = 1, \cdots, K$ during K total stages of multistage fusion. Let Θ denote the neural network parameters across all modules.

`HIGHLIGHT`: At each stage k, a subset of the multimodal signals represented in $\mathbf{h}_t$ will be au-

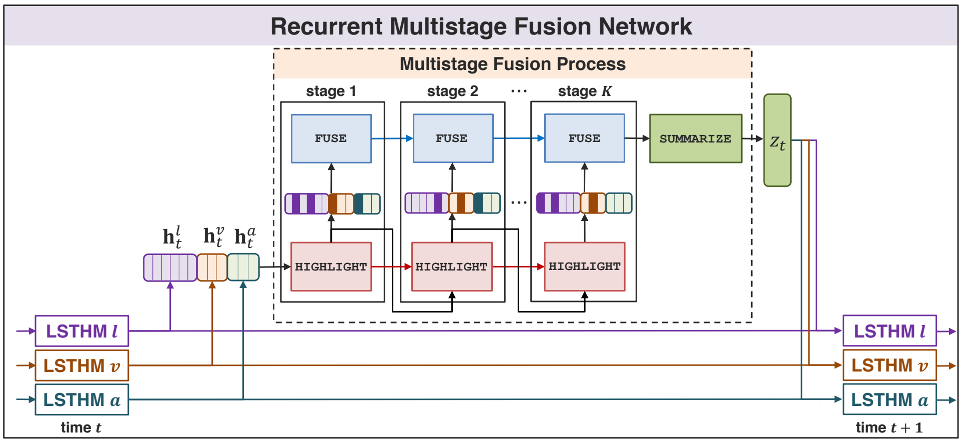

Figure 2: The Recurrent Multistage Fusion Network for multimodal language analysis. The Multistage Fusion Process has three modules: `HIGHLIGHT`, `FUSE` and `SUMMARIZE`. Multistage fusion begins with the concatenated intra-modal network outputs $\mathbf{h}_t^l, \mathbf{h}_t^v, \mathbf{h}_t^a$. At each stage, the `HIGHLIGHT` module identifies a subset of multimodal signals and the `FUSE` module performs local fusion before integration with previous fusion representations. The `SUMMARIZE` module translates the representation at the final stage into a cross-modal representation $\mathbf{z}_t$ to be fed back into the intra-modal recurrent networks.

tomatically highlighted for fusion. Formally, this module is defined by the process function f_H:

$$\mathbf{a}_t^{[k]} = f_H(\mathbf{h}_t \,;\, \mathbf{a}_t^{[1:k-1]}, \Theta) \tag{1}$$

where at stage k, $\mathbf{a}_t^{[k]}$ is a set of attention weights which are inferred based on the previously assigned attention weights $\mathbf{a}_t^{[1:k-1]}$. As a result, the highlights at a specific stage k will be dependent on previous highlights. To fully encapsulate these dependencies, the attention assignment process is performed in a recurrent manner using a LSTM which we call the `HIGHLIGHT` LSTM. The initial `HIGHLIGHT` LSTM memory at stage 0, $\mathbf{c}_t^{\texttt{HIGHLIGHT}[0]}$, is initialized using a network $\mathcal{M}$ that maps $\mathbf{h}_t$ into LSTM memory space:

$$\mathbf{c}_t^{\texttt{HIGHLIGHT}[0]} = \mathcal{M}(\mathbf{h}_t \,;\, \Theta) \tag{2}$$

This allows the memory mechanism of the `HIGHLIGHT` LSTM to dynamically adjust to the intra-modal representations $\mathbf{h}_t$. The output of the `HIGHLIGHT` LSTM $\mathbf{h}_t^{\texttt{HIGHLIGHT}[k]}$ is softmax activated to produce attention weights $\mathbf{a}_t^{[k]}$ at every stage k of the multistage fusion process:

$$\mathbf{a}_t^{[k]}{}_j = \frac{\exp\left(\mathbf{h}_t^{\texttt{HIGHLIGHT}[k]}{}_j\right)}{\sum_{d=1}^{|\mathbf{h}_t^{\texttt{HIGHLIGHT}[k]}|} \exp\left(\mathbf{h}_t^{\texttt{HIGHLIGHT}[k]}{}_d\right)} \tag{3}$$

and $\mathbf{a}_t^{[k]}$ is fed as input into the `HIGHLIGHT` LSTM at stage $k+1$. Therefore, the `HIGHLIGHT` LSTM functions as a decoder LSTM (Sutskever et al., 2014; Cho et al., 2014) in order to capture the dependencies on previous attention assignments. Highlighting is performed by element-wise multiplying the attention weights $\mathbf{a}_t^{[k]}$ with the concatenated intra-modal representations $\mathbf{h}_t$:

$$\tilde{\mathbf{h}}_t^{[k]} = \mathbf{h}_t \odot \mathbf{a}_t^{[k]} \tag{4}$$

where $\odot$ denotes the Hadamard product and $\tilde{\mathbf{h}}_t^{[k]}$ are the attended multimodal signals that will be used for the fusion at stage k.

`FUSE`: The highlighted multimodal signals are simultaneously fused in a local fusion and then integrated with fusion representations from previous stages. Formally, this module is defined by the process function f_F:

$$\mathbf{s}_t^{[k]} = f_F(\tilde{\mathbf{h}}_t^{[k]} \,;\, \mathbf{s}_t^{[1:k-1]}, \Theta) \tag{5}$$

where $\mathbf{s}_t^{[k]}$ denotes the integrated fusion representations at stage k. We employ a `FUSE` LSTM to simultaneously perform the local fusion and the integration with previous fusion representations. The `FUSE` LSTM input gate enables a local fusion while the `FUSE` LSTM forget and output gates enable integration with previous fusion results. The initial `FUSE` LSTM memory at stage 0, $\mathbf{c}_t^{\texttt{FUSE}[0]}$, is initialized using random orthogonal matrices (Arjovsky et al., 2015; Le et al., 2015).

`SUMMARIZE`: After completing K recursive stages of `HIGHLIGHT` and `FUSE`, the `SUMMARIZE` operation generates a cross-modal representation using all final fusion representations $\mathbf{s}_t^{[1:K]}$. Formally, this operation is defined as:

$$\mathbf{z}_t = \mathcal{S}(\mathbf{s}_t^{[1:K]}\ ;\ \Theta) \tag{6}$$

where $\mathbf{z}_t$ is the final output of the multistage fusion process and represents all cross-modal interactions discovered at time t. The summarized cross-modal representation is then fed into the intra-modal recurrent networks as described in the subsection 3.3.

3.3 System of Long Short-term Hybrid Memories

To integrate the cross-modal representations $\mathbf{z}_t$ with the temporal intra-modal representations, we employ a system of Long Short-term Hybrid Memories (LSTHMs) (Zadeh et al., 2018b). The LSTHM extends the LSTM formulation to include the cross-modal representation $\mathbf{z}_t$ in a hybrid memory component:

$$\mathbf{i}_{t+1}^m = \sigma(\mathbf{W}_i^m\ \mathbf{x}_{t+1}^m + \mathbf{U}_i^m\ \mathbf{h}_t^m + \mathbf{V}_i^m\ \mathbf{z}_t + \mathbf{b}_i^m) \tag{7}$$
$$\mathbf{f}_{t+1}^m = \sigma(\mathbf{W}_f^m\ \mathbf{x}_{t+1}^m + \mathbf{U}_f^m\ \mathbf{h}_t^m + \mathbf{V}_f^m\ \mathbf{z}_t + \mathbf{b}_f^m) \tag{8}$$
$$\mathbf{o}_{t+1}^m = \sigma(\mathbf{W}_o^m\ \mathbf{x}_{t+1}^m + \mathbf{U}_o^m\ \mathbf{h}_t^m + \mathbf{V}_o^m\ \mathbf{z}_t + \mathbf{b}_o^m) \tag{9}$$
$$\bar{\mathbf{c}}_{t+1}^m = \mathbf{W}_{\bar{c}}^m\ \mathbf{x}_{t+1}^m + \mathbf{U}_{\bar{c}}^m\ \mathbf{h}_t^m + \mathbf{V}_{\bar{c}}^m\ \mathbf{z}_t + \mathbf{b}_{\bar{c}}^m \tag{10}$$
$$\mathbf{c}_{t+1}^m = \mathbf{f}_t^m \odot \mathbf{c}_t^m + \mathbf{i}_t^m \odot tanh(\bar{\mathbf{c}}_{t+1}^m) \tag{11}$$
$$\mathbf{h}_{t+1}^m = \mathbf{o}_{t+1}^m \odot tanh(\mathbf{c}_{t+1}^m) \tag{12}$$

where σ is the (hard-)sigmoid activation function, $tanh$ is the tangent hyperbolic activation function, $\odot$ denotes the Hadamard product. $\mathbf{i}, \mathbf{f}$ and $\mathbf{o}$ are the input, forget and output gates respectively. $\bar{\mathbf{c}}_{t+1}^m$ is the proposed update to the hybrid memory $\mathbf{c}_t^m$ at time $t+1$ and $\mathbf{h}_t^m$ is the time distributed output of each modality. The cross-modal representation $\mathbf{z}_t$ is modeled by the Multistage Fusion Process as discussed in subsection 3.2. The hybrid memory $\mathbf{c}_t^m$ contains both intra-modal interactions from individual modalities $\mathbf{x}_t^m$ as well as the cross-modal interactions captured in $\mathbf{z}_t$.

3.4 Optimization

The multimodal prediction task is performed using a final representation $\mathcal{E}$ which integrate (1) the last outputs from the LSTHMs and (2) the last cross-modal representation $\mathbf{z}_T$. Formally, $\mathcal{E}$ is defined as:

$$\mathcal{E} = (\bigoplus_{m \in M} \mathbf{h}_T^m) \bigoplus \mathbf{z}_T \tag{13}$$

where $\oplus$ denotes vector concatenation. $\mathcal{E}$ can then be used as a multimodal representation for supervised or unsupervised analysis of multimodal language. It summarizes all modeled intra-modal and cross-modal representations from the multimodal sequences. RMFN is differentiable end-to-end which allows the network parameters Θ to be learned using gradient descent approaches.

4 Experimental Setup

To evaluate the performance and generalization of RMFN, three domains of human multimodal language were selected: multimodal sentiment analysis, emotion recognition, and speaker traits recognition.

4.1 Datasets

All datasets consist of monologue videos. The speaker's intentions are conveyed through three modalities: language, visual and acoustic.

Multimodal Sentiment Analysis involves analyzing speaker sentiment based on video content. Multimodal sentiment analysis extends conventional language-based sentiment analysis to a multimodal setup where both verbal and non-verbal signals contribute to the expression of sentiment. We use **CMU-MOSI** (Zadeh et al., 2016) which consists of 2199 opinion segments from online videos each annotated with sentiment in the range [-3,3].

Multimodal Emotion Recognition involves identifying speaker emotions based on both verbal and nonverbal behaviors. We perform experiments on the **IEMOCAP** dataset (Busso et al., 2008) which consists of 7318 segments of recorded dyadic dialogues annotated for the presence of human emotions happiness, sadness, anger and neutral.

Multimodal Speaker Traits Recognition involves recognizing speaker traits based on multimodal communicative behaviors. **POM** (Park et al., 2014) contains 903 movie review videos each annotated for 12 speaker traits: confident (con), passionate (pas), voice pleasant (voi), credible (cre), vivid (viv), expertise (exp), reserved (res), trusting (tru), relaxed (rel), thorough (tho), nervous (ner), persuasive (per) and humorous (hum).

4.2 Multimodal Features and Alignment

GloVe word embeddings (Pennington et al., 2014), Facet (iMotions, 2017) and COVAREP (Degottex et al., 2014) are extracted for the language, visual and acoustic modalities respectively [1]. Forced alignment is performed using P2FA (Yuan and Liberman, 2008) to obtain the exact utterance times

---

[1] Details on feature extraction are in supplementary.

Dataset	CMU-MOSI				
Task	Sentiment				
Metric	A2 ↑	F1 ↑	A7 ↑	MAE ↓	Corr ↑
SOTA3	$76.5^{\diamond}$	$74.5^{\dagger}$	$33.2^{\#}$	$0.968^{\S}$	$0.622^{\flat}$
SOTA2	$77.1^{\S}$	$77.0^{\S}$	$34.1^{\star}$	$0.965^{\star}$	$0.625^{\S}$
SOTA1	$77.4^{\star}$	$77.3^{\star}$	$34.7^{\S}$	$0.955^{\diamond}$	$0.632^{\star}$
RMFN	**78.4**	**78.0**	**38.3**	**0.922**	**0.681**
Δ_{SOTA}	**↑ 1.0**	**↑ 0.7**	**↑ 3.6**	**↓ 0.033**	**↑ 0.049**

Table 1: Sentiment prediction results on CMU-MOSI. Best results are highlighted in bold and Δ_{SOTA} shows improvement over previous state of the art (SOTA). Symbols denote baseline model which achieves the reported performance: MFN: $\star$, MARN: §, GME-LSTM(A): $\diamond$, TFN: †, MV-LSTM: $\#$, EF-LSTM: $\flat$. The RMFN outperforms the current SOTA across all evaluation metrics. Improvements are highlighted in green.

Dataset	IEMOCAP Emotions							
Task	Happy		Sad		Angry		Neutral	
Metric	A2 ↑	F1 ↑	A2 ↑	F1 ↑	A2 ↑	F1 ↑	A2 ↑	F1 ↑
SOTA3	$86.1^{\times}$	$83.6^{\S}$	$83.2^{\bullet}$	$81.7^{\bullet}$	$85.0^{\star}$	$84.2^{\S}$	$68.2^{\flat}$	$66.7^{\#}$
SOTA2	$86.5^{\star}$	$84.0^{\star}$	$83.4^{\dagger}$	$82.1^{\star}$	$85.1^{\#}$	$84.3^{\#}$	$68.8^{\flat}$	$68.5^{\flat}$
SOTA1	$86.7^{\S}$	$84.2^{\flat}$	$83.5^{\star}$	$82.8^{\dagger}$	$\mathbf{85.2}^{\flat}$	$84.5^{\flat}$	$\mathbf{69.6}^{\star}$	$\mathbf{69.2}^{\star}$
RMFN	**87.5**	**85.8**	**83.8**	**82.9**	85.1	**84.6**	69.5	69.1
Δ_{SOTA}	**↑ 0.8**	**↑ 1.6**	**↑ 0.3**	**↑ 0.1**	–	**↑ 0.1**	–	–

Table 2: Emotion recognition results on IEMOCAP. Best results are highlighted in bold and Δ_{SOTA} shows improvement over previous SOTA. Symbols denote baseline model which achieves the reported performance: MFN: $\star$, MARN: §, BC-LSTM: $\bullet$, TFN: †, MV-LSTM: $\#$, EF-LSTM: $\flat$, SVM: $\times$. The RMFN outperforms the current SOTA across evaluation metrics except Δ_{SOTA} entries in gray. Improvements are highlighted in green.

of each word. We obtain the aligned video and audio features by computing the expectation of their modality feature values over each word utterance time interval (Tsai et al., 2018).

4.3 Baseline Models

We compare to the following models for multimodal machine learning: **MFN** (Zadeh et al., 2018a) synchronizes multimodal sequences using a multi-view gated memory. It is the current state of the art on CMU-MOSI and POM. **MARN** (Zadeh et al., 2018b) models intra-modal and cross-modal interactions using multiple attention coefficients and hybrid LSTM memory components. **GME-LSTM(A)** (Chen et al., 2017) learns binary gating mechanisms to remove noisy modalities that are contradictory or redundant for prediction. **TFN** (Zadeh et al., 2017) models unimodal, bimodal and trimodal interactions using tensor products. **BC-LSTM** (Poria et al., 2017) performs context-dependent sentiment analysis and emotion recognition, currently state of the art on IEMOCAP. **EF-LSTM** concatenates the multimodal inputs and uses that as input to a single LSTM (Hochreiter and Schmidhuber, 1997b). We also implement the Stacked, (**EF-SLSTM**) (Graves et al., 2013) Bidirectional (**EF-BLSTM**) (Schuster and Paliwal, 1997) and Stacked Bidirectional (**EF-SBLSTM**) LSTMs. For descriptions of the remaining baselines, we refer the reader to **EF-HCRF** (Quattoni et al., 2007), **EF/MV-LDHCRF** (Morency et al., 2007), **MV-HCRF** (Song et al., 2012), **EF/MV-HSSHCRF** (Song et al., 2013), **MV-LSTM** (Rajagopalan et al., 2016), **DF** (Nojavanasghari et al., 2016), **SAL-CNN** (Wang et al., 2016), **C-MKL** (Poria et al., 2015), **THMM** (Morency et al., 2011), **SVM** (Cortes and Vapnik, 1995; Park et al., 2014) and **RF** (Breiman, 2001).

4.4 Evaluation Metrics

For classification, we report accuracy Ac where c denotes the number of classes and F1 score. For regression, we report Mean Absolute Error MAE and Pearson's correlation r. For MAE lower values indicate stronger performance. For all remaining metrics, higher values indicate stronger performance.

5 Results and Discussion

5.1 Performance on Multimodal Language

Results on CMU-MOSI, IEMOCAP and POM are presented in Tables 1, 2 and 3 respectively[2]. We achieve state-of-the-art or competitive results for all domains, highlighting RMFN's capability in human multimodal language analysis. We observe that RMFN does not improve results on IEMOCAP neutral emotion and the model outperforming RMFN is a memory-based fusion baseline (Zadeh et al., 2018a). We believe that this is because neutral expressions are quite idiosyncratic. Some people may always look angry given their facial configuration (e.g., natural eyebrow raises of actor Jack Nicholson). In these situations, it becomes useful to compare the current image with a memorized or aggregated representation of the speaker's face. Our proposed multistage fusion approach can easily be extended to memory-based fusion methods.

---

[2] Results for all individual baseline models are in supplementary. State-of-the-art (SOTA)1/2/3 represent the three best performing baseline models on each dataset.

Dataset	POM Speaker Personality Traits											
Task	Con	Pas	Voi	Cre	Viv	Exp	Res	Rel	Tho	Ner	Per	Hum
Metric	A7 ↑	A7 ↑	A7 ↑	A7 ↑	A7 ↑	A7 ↑	A5 ↑	A5 ↑	A5 ↑	A5 ↑	A7 ↑	A6 ↑
SOTA3	$26.6^{\bullet}$	$31.0^{\dagger}$	$34.0^{\flat}$	$29.6^{\flat}$	$35.0^{\S}$	$31.0^{\flat}$	$34.0^{\heartsuit}$	$50.7^{\#}$	$45.8^{\bullet}$	$44.8^{\flat}$	$28.1^{\times}$	$40.4^{\heartsuit}$
SOTA2	$29.1^{\S}$	$34.0^{\S}$	$34.5^{\S}$	$31.5^{\S}$	$36.5^{\bullet}$	$31.5^{\S}$	$36.9^{\S}$	$52.2^{\S}$	$46.8^{\S}$	$47.3^{\S}$	$31.0^{\S}$	$44.8^{\S}$
SOTA1	$34.5^{\star}$	$35.5^{\star}$	**37.4***	$34.5^{\star}$	$36.9^{\star}$	$36.0^{\star}$	$38.4^{\star}$	$53.2^{\star}$	$47.3^{\star}$	$47.8^{\star}$	$34.0^{\star}$	**47.3***
RMFN	**37.4**	**38.4**	**37.4**	**37.4**	**38.9**	**38.9**	**39.4**	**53.7**	**48.3**	**48.3**	**35.0**	46.8
Δ_{SOTA}	**↑ 2.9**	**↑ 2.9**	**0.0**	**↑ 2.9**	**↑ 2.0**	**↑ 3.9**	**↑ 1.0**	**↑ 0.5**	**↑ 1.0**	**↑ 0.5**	**↑ 1.0**	–

Table 3: Results for personality trait recognition on POM. Best results are highlighted in bold and Δ_{SOTA} shows improvement over previous SOTA. Symbols denote baseline model which achieves the reported performance: MFN: $\star$, MARN: §, BC-LSTM: $\bullet$, TFN: †, MV-LSTM: #, EF-LSTM: $\flat$, RF: $\heartsuit$, SVM: $\times$. The MFP outperforms the current SOTA across all evaluation metrics except the Δ_{SOTA} entries highlighted in gray. Improvements are highlighted in green.

Dataset	CMU-MOSI				
Task	Sentiment				
Metric	A2 ↑	F1 ↑	A7 ↑	MAE ↓	Corr ↑
RMFN-R1	75.5	75.5	35.1	0.997	0.653
RMFN-R2	76.4	76.4	34.5	0.967	0.642
RMFN-R3	**78.4**	**78.0**	**38.3**	**0.922**	**0.681**
RMFN-R4	76.0	76.0	36.0	0.999	0.640
RMFN-R5	75.5	75.5	30.9	1.009	0.617
RMFN-R6	70.4	70.5	30.8	1.109	0.560
RMFN	**78.4**	**78.0**	**38.3**	**0.922**	**0.681**

Table 4: Effect of varying the number of stages on CMU-MOSI sentiment analysis performance. Multistage fusion improves performance as compared to single stage fusion.

Dataset	CMU-MOSI				
Task	Sentiment				
Metric	A2 ↑	F1 ↑	A7 ↑	MAE ↓	Corr ↑
MARN	77.1	77.0	34.7	0.968	0.625
RMFN (no MFP)	76.5	76.5	30.8	0.998	0.582
RMFN (no `HIGHLIGHT`)	77.9	77.9	35.9	0.952	0.666
RMFN	**78.4**	**78.0**	**38.3**	**0.922**	**0.681**

Table 5: Comparison studies of RMFN on CMU-MOSI. Modeling cross-modal interactions using multistage fusion and attention weights are crucial in multimodal language analysis.

5.2 Analysis of Multistage Fusion

To achieve a deeper understanding of the multistage fusion process, we study five research questions. (Q1): whether modeling cross-modal interactions across multiple stages is beneficial. (Q2): the effect of the number of stages K during multistage fusion on performance. (Q3): the comparison between multistage and independent modeling of cross-modal interactions. (Q4): whether modeling cross-modal interactions are helpful. (Q5): whether attention weights from the `HIGHLIGHT` module are required for modeling cross-modal interactions.

Q1: To study the effectiveness of the multistage fusion process, we test the baseline RMFN-R1 which performs fusion in only one stage instead of across multiple stages. This model makes the strong assumption that all cross-modal interactions can be modeled during only one stage. From Table 4, RMFN-R1 underperforms as compared to RMFN which performs multistage fusion.

Q2: We test baselines RMFN-RK which perform K stages of fusion. From Table 4, we observe that increasing the number of stages K increases the model's capability to model cross-modal interactions up to a certain point ($K = 3$) in our experiments. Further increases led to decreases in performance and we hypothesize this is due to overfitting on the dataset.

Q3: To compare multistage against independent modeling of cross-modal interactions, we pay close attention to the performance comparison with respect to MARN which models multiple cross-modal interactions all at once (see Table 5). RMFN shows improved performance, indicating that multistage fusion is both effective and efficient for human multimodal language modeling.

Q4: RMFN (no MFP) represents a system of LSTHMs without the integration of $\mathbf{z}_t$ from the MFP to model cross-modal interactions. From Table 5, RMFN (no MFP) is outperformed by RMFN, confirming that modeling cross-modal interactions is crucial in analyzing human multimodal language.

Q5: RMFN (no `HIGHLIGHT`) removes the `HIGHLIGHT` module from MFP during multistage fusion. From Table 5, RMFN (no `HIGHLIGHT`) underperforms, indicating that highlighting multimodal representations using attention weights are important for modeling cross-modal interactions.

5.3 Visualizations

Using an attention assignment mechanism during the `HIGHLIGHT` process gives more interpretability to the model since it allows us to visualize the attended multimodal signals at each stage and

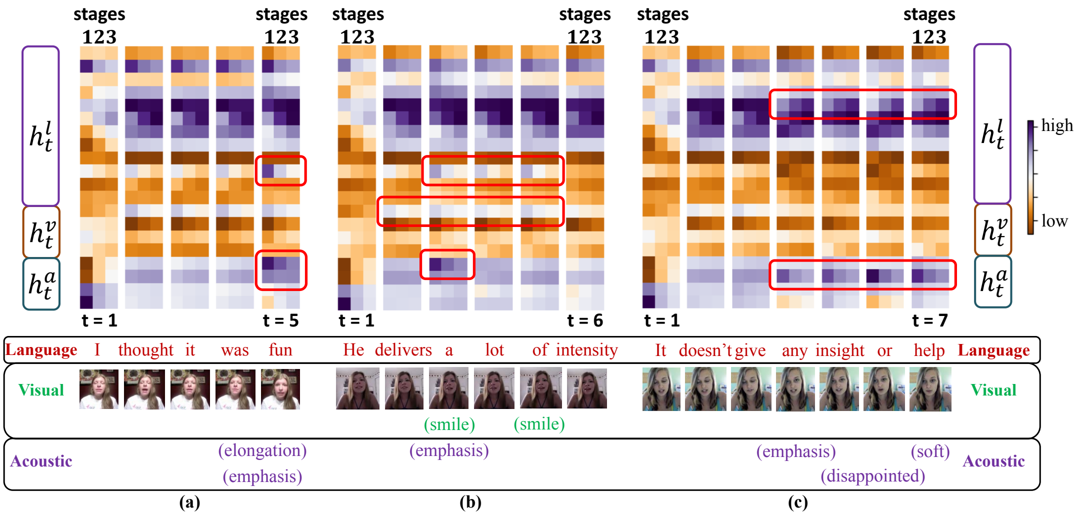

Figure 3: Visualization of learned attention weights across stages 1,2 and 3 of the multistage fusion process and across time of the multimodal sequence. We observe that the attention weights are diverse and evolve across stages and time. In these three examples, the red boxes emphasize specific moments of interest. (a) Synchronized interactions: the positive word "fun" and the acoustic behaviors of emphasis and elongation ($t = 5$) are synchronized in both attention weights for language and acoustic features. (b) Asynchronous trimodal interactions: the asynchronous presence of a smile ($t = 2 : 5$) and emphasis ($t = 3$) help to disambiguate the language modality. (c) Bimodal interactions: the interactions between the language and acoustic modalities are highlighted by alternating stages of fusion ($t = 4 : 7$).

time step (see Figure 3). Using RMFN trained on the CMU-MOSI dataset, we plot the attention weights across the multistage fusion process for three videos in CMU-MOSI. Based on these visualizations we first draw the following general observations on multistage fusion:

Across stages: Attention weights change their behaviors across the multiple stages of fusion. Some features are highlighted by earlier stages while other features are used in later stages. This supports our hypothesis that RMFN learns to specialize in different stages of the fusion process.

Across time: Attention weights vary over time and adapt to the multimodal inputs. We observe that the attention weights are similar if the input contains no new information. As soon as new multimodal information comes in, the highlighting mechanism in RMFN adapts to these new inputs.

Priors: Based on the distribution of attention weights, we observe that the language and acoustic modalities seem the most commonly highlighted. This represents a prior over the expression of sentiment in human multimodal language and is closely related to the strong connections between language and speech in human communication (Kuhl, 2000).

Inactivity: Some attention coefficients are not active (always orange) throughout time. We hypothesize that these corresponding dimensions carry only intra-modal dynamics and are not involved in the formation of cross-modal interactions.

5.4 Qualitative Analysis

In addition to the general observations above, Figure 3 shows three examples where multistage fusion learns cross-modal representations across three different scenarios.

Synchronized Interactions: In Figure 3(a), the language features are highlighted corresponding to the utterance of the word "fun" that is highly indicative of sentiment ($t = 5$). This sudden change is also accompanied by a synchronized highlighting of the acoustic features. We also notice that the highlighting of the acoustic features lasts longer across the 3 stages since it may take multiple stages to interpret all the new acoustic behaviors (elongated tone of voice and phonological emphasis).

Asynchronous Trimodal Interactions: In Figure 3(b), the language modality displays ambiguous sentiment: "delivers a lot of intensity" can be inferred as both positive or negative. We observe that the circled attention units in the visual and acoustic features correspond to the asynchronous presence of a smile ($t = 2 : 5$) and phonological emphasis ($t = 3$) respectively. These nonverbal behaviors resolve ambiguity in language and result in an overall display of positive sentiment. We further note the coupling of attention weights that highlight the language, visual and acoustic features across stages ($t = 3 : 5$), further emphasizing the coordination of all three modalities during multistage fusion despite their asynchronous occurrences.

Bimodal Interactions: In Figure 3(c), the language modality is better interpreted in the context of acoustic behaviors. The disappointed tone and soft voice provide the nonverbal information useful for sentiment inference. This example highlights the bimodal interactions ($t = 4 : 7$) in alternating stages: the acoustic features are highlighted more in earlier stages while the language features are highlighted increasingly in later stages.

6 Conclusion

This paper proposed the Recurrent Multistage Fusion Network (RMFN) which decomposes the multimodal fusion problem into multiple stages, each focused on a subset of multimodal signals. Extensive experiments across three publicly-available datasets reveal that RMFN is highly effective in modeling human multimodal language. In addition to achieving state-of-the-art performance on all datasets, our comparisons and visualizations reveal that the multiple stages coordinate to capture both synchronous and asynchronous multimodal interactions. In future work, we are interested in merging our model with memory-based fusion methods since they have complementary strengths as discussed in subsection 5.1.

Acknowledgements

This material is based upon work partially supported by the National Science Foundation (Award #1833355) and Samsung. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of National Science Foundation or Samsung, and no official endorsement should be inferred. The authors thank Yao Chong Lim, Venkata Ramana Murthy Oruganti, Zhun Liu, Ying Shen, Volkan Cirik, and the anonymous reviewers for their constructive comments on this paper.

References

Harika Abburi, Rajendra Prasath, Manish Shrivastava, and Suryakanth V Gangashetty. 2016. Multimodal sentiment analysis using deep neural networks. In *International Conference on Mining Intelligence and Knowledge Exploration*, pages 58–65. Springer.

Martín Arjovsky, Amar Shah, and Yoshua Bengio. 2015. Unitary evolution recurrent neural networks. *CoRR*, abs/1511.06464.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2017. Multimodal machine learning: A survey and taxonomy. *arXiv preprint arXiv:1705.09406*.

Leonard E Baum and Ted Petrie. 1966. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563.

Dario Bertero, Farhad Bin Siddique, Chien-Sheng Wu, Yan Wan, Ricky Chan, and Pascale Fung. 2016. Real-time speech emotion and sentiment recognition for interactive dialogue systems.

Leo Breiman. 2001. Random forests. *Mach. Learn.*, 45(1):5–32.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette Chang, Sungbok Lee, and Shrikanth S. Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Journal of Language Resources and Evaluation*, 42(4):335–359.

E. Cambria. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107.

Minghai Chen, Sen Wang, Paul Pu Liang, Tadas Baltrušaitis, Amir Zadeh, and Louis-Philippe Morency. 2017. Multimodal sentiment analysis with word-level fusion and reinforcement learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, ICMI 2017.

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Q. Weinberger, and Claire Cardie. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *CoRR*, abs/1606.01614.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton and Co., The Hague.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Gilles Degottex, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer. 2014. Covarepa collaborative voice analysis repository for speech technologies. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 960–964. IEEE.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 513–520, USA. Omnipress.

A. Graves, A. r. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

Abhinav Gupta, Yajie Miao, Leonardo Neves, and Florian Metze. 2017. Visual features for context-aware speech recognition. *CoRR*, abs/1712.00489.

Kun Han, Dong Yu, and Ivan Tashev. 2014. Speech emotion recognition using deep neural network and extreme learning machine.

David F. Harwath and James R. Glass. 2017. Learning word-like units from joint audio-visual analysis. *CoRR*, abs/1701.07481.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. Trueskill™: A bayesian skill rating system. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 569–576. MIT Press.

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

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *CoRR*, abs/1508.01991.

iMotions. 2017. Facial expression analysis.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*.

L. C. Jain and L. R. Medsker. 1999. *Recurrent Neural Networks: Design and Applications*, 1st edition. CRC Press, Inc., Boca Raton, FL, USA.

Herman Kamper, Shane Settle, Gregory Shakhnarovich, and Karen Livescu. 2017. Visually grounded learning of keyword prediction from untranscribed speech. *CoRR*, abs/1703.08136.

Patricia K. Kuhl. 2000. A new view of language acquisition. *Proceedings of the National Academy of Sciences*, 97(22):11850–11857.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Egor Lakomkin, Cornelius Weber, Sven Magg, and Stefan Wermter. 2018. Reusing neural speech representations for auditory emotion recognition. *CoRR*, abs/1803.11508.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941.

Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. 2018. Multimodal local-global ranking fusion for emotion recognition. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, ICMI 2018.

Zhun Liu, Ying Shen, Varun Bharadhwaj Lakshminarasimhan, Paul Pu Liang, AmirAli Bagher Zadeh, and Louis-Philippe Morency. 2018. Efficient low-rank multimodal fusion with modality-specific factors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2247–2256. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. ACL 2016.

Shotaro Misawa, Motoki Taniguchi, Yasuhide Miura, and Tomoko Ohkuma. 2017. Character-based bidirectional lstm-crf with words and characters for japanese named entity recognition. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 97–102. Association for Computational Linguistics.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 169–176. ACM.

Louis-Philippe Morency, Ariadna Quattoni, and Trevor Darrell. 2007. Latent-dynamic discriminative models for continuous gesture recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. 2011. Multimodal deep learning. In *ICML*, pages 689–696. Omnipress.

Behnaz Nojavanasghari, Deepak Gopinath, Jayanth Koushik, Tadas Baltrušaitis, and Louis-Philippe Morency. 2016. Deep multimodal fusion for persuasiveness prediction. In *Proceedings of the 18th*

ACM International Conference on Multimodal Interaction, ICMI 2016, pages 284–288, New York, NY, USA. ACM.

German I. Parisi, Jun Tani, Cornelius Weber, and Stefan Wermter. 2017. Emergence of multimodal action representations from neural network self-organization. *Cognitive Systems Research*, 43:208 – 221.

Sunghyun Park, Han Suk Shim, Moitreya Chatterjee, Kenji Sagae, and Louis-Philippe Morency. 2014. Computational analysis of persuasiveness in social multimedia: A novel dataset and multimodal prediction approach. In *Proceedings of the 16th International Conference on Multimodal Interaction*, ICMI '14, pages 50–57, New York, NY, USA. ACM.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Hai Pham, Thomas Manzini, Paul Pu Liang, and Barnabas Poczos. 2018. Seq2seq2sentiment: Multimodal sequence to sequence models for sentiment analysis. In *Proceedings of Grand Challenge and Workshop on Human Multimodal Language (Challenge-HML)*, pages 53–63, Melbourne, Australia. Association for Computational Linguistics.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2539–2544.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 873–883.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 439–448. IEEE.

Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Michael Collins, and Trevor Darrell. 2007. Hidden conditional random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1848–1852.

Shyam Sundar Rajagopalan, Louis-Philippe Morency, Tadas Baltrušaitis, and Goecke Roland. 2016. Extending long short-term memory for multi-view structured learning. In *European Conference on Computer Vision*.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78. ACL.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

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

Hagen Soltau, Hank Liao, and Hasim Sak. 2016. Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition. *CoRR*, abs/1610.09975.

Yale Song, Louis-Philippe Morency, and Randall Davis. 2012. Multi-view latent variable discriminative models for action recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2120–2127. IEEE.

Yale Song, Louis-Philippe Morency, and Randall Davis. 2013. Action recognition by hierarchical sequence summarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3562–3569.

Nitish Srivastava and Ruslan R Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2222–2230. Curran Associates, Inc.

F. Sun, D. Harwath, and J. Glass. 2016. Look, listen, and decode: Multimodal speech recognition with images. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 573–578.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

P. Taylor, J. N. Hobbs, J. Burroni, and H. T. Siegelmann. 2015. The global landscape of cognition: hierarchical aggregation as an organizational principle of human cortical networks and functions. *Scientific Reports*, 5:18112 EP –.

Yao-Hung Hubert Tsai, Paul Pu Liang, Amir Zadeh, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2018. Learning factorized multimodal representations. *arXiv preprint arXiv:1806.06176*.

Haohan Wang, Aaksha Meghawat, Louis-Philippe Morency, and Eric P Xing. 2016. Select-additive learning: Improving cross-individual generalization in multimodal sentiment analysis. *arXiv preprint arXiv:1609.05244*.

Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*.

Jiahong Yuan and Mark Liberman. 2008. Speaker identification on the scotus corpus. *Journal of the Acoustical Society of America*, 123(5):3878.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Empirical Methods in Natural Language Processing, EMNLP*.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018a. Memory fusion network for multiview sequential learning. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. 2018b. Multi-attention recurrent network for human communication comprehension. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems*, 31(6):82–88.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent Highway Networks. *arXiv preprint arXiv:1607.03474*.

Temporally Grounding Natural Sentence in Video

Jingyuan Chen[1]* **Xinpeng Chen**[2]* **Lin Ma**[2] **Zequn Jie**[2] **Tat-Seng Chua**[1]
[1]National University of Singapore [2]Tencent AI Lab
{jingyuanchen91, jschenxinpeng, forest.linma, zequn.nus}@gmail.com, dcscts@nus.edu.sg

Abstract

We introduce an effective and efficient method that grounds (*i.e.*, localizes) natural sentences in long, untrimmed video sequences. Specifically, a novel Temporal GroundNet (TGN)[1] is proposed to temporally capture the evolving fine-grained frame-by-word interactions between video and sentence. TGN sequentially scores a set of temporal candidates ended at each frame based on the exploited frame-by-word interactions, and finally grounds the segment corresponding to the sentence. Unlike traditional methods treating the overlapping segments separately in a sliding window fashion, TGN aggregates the historical information and generates the final grounding result in one single pass. We extensively evaluate our proposed TGN on three public datasets with significant improvements over the state-of-the-arts. We further show the consistent effectiveness and efficiency of TGN through an ablation study and a runtime test.

1 Introduction

We examine the task of Natural Sentence Grounding in Video (NSGV). Given an untrimmed video and a natural sentence, the goal is to determine the start and end timestamps of the segment in the video which corresponds to the given sentence, as shown in Figure 1 (a). Comparing with the other video researches, such as bidirectional video-sentence retrieval (Xu et al., 2015b), video attractiveness prediction (Chen et al., 2018, 2016), and video captioning (Pasunuru and Bansal, 2017; Wang et al., 2018a,b), NSGV needs to model not only the characteristics of sentence and video but also the fine-grained interactions between the two modalities, which is even more challenging.

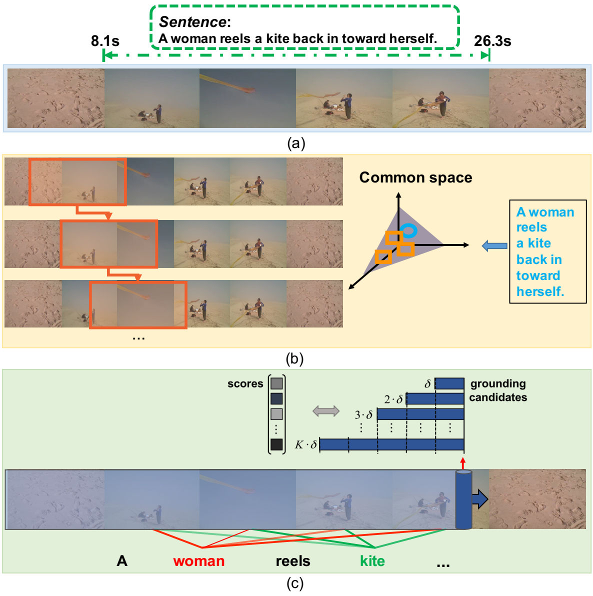

Figure 1: (a) The Natural Sentence Grounding in Video (NSGV) task. (b) A common space based matching method performs in a sliding window fashion. (c) Our proposed Temporal GroundNet (TGN) localizes the candidate video segments at multiple scales in a single processing pass. The frames in the video and the words in the sentence interact attentively to perform fine-grained frame-by-word matchings for grounding sentence in video.

Recently, several related works (Gao et al., 2017; Hendricks et al., 2017) leverage one temporal sliding window approach over video sequences to generate video segment candidates, which are then independently combined (Gao et al., 2017) or compared (Hendricks et al., 2017) with the given sentence to make the grounding prediction. Although the existing works have achieved promising performances, they are still suffering from inferior effectiveness and efficiency. First, existing methods project the video segment and sentence into one common space, as shown in Figure 1 (b),

---

[*] Work done while Jingyuan Chen and Xinpeng Chen were Research Interns with Tencent AI Lab.

[1] The project homepage is `https://jingyuanchen.github.io/archive/tgn.html`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 162–171
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

where the two generated embedding vectors are used to perform the matching between video segment and sentence. Such a matching is only performed in the global segment and sentence level and thus not expressive enough, which ignores the fine-grained matching relations between video frames and the words in sentences. Second, in order to handle the diverse temporal scales and locations of the candidate segments, exhaustive matching between the large amount of overlapping segments and the sentence is required. As such, the sliding window methods are very computationally expensive.

In order to tackle the above two limitations, we introduce a novel Temporal GroundNet (TGN) model, the first dynamic single-stream deep architecture for the NSGV task that takes full advantage of fine-grained interactions between video frames and words in a sentence, as shown in Figure 1 (c). TGN sequentially processes video frames, where at each time step we rely on a novel multimodal interactor to exploit the evolving fine-grained frame-by-word interactions. Then, TGN works on the yielded interaction status to simultaneously score a set of temporal candidates of multiple scales and finally localize the video segment that corresponds to the sentence. More importantly, our proposed TGN is able to analyze an untrimmed video frame by frame without resorting to handling overlapping temporal video segments.

2 Related Work

2.1 Grounding Natural Language in Image

Grounding natural language in image is also known as natural language object retrieval. The task is to localize an image region described by natural language, which involves comprehending and modeling different spatial contexts, such as spatial configurations (Hu et al., 2016), attributes (Yu et al., 2018), and relationships between objects (Hu et al., 2017). Specifically, the task is usually formulated as a ranking problem over a set of candidate regions in a given image, where candidate spatial locations come from region proposal methods (Uijlings et al., 2013; Jie et al., 2016b,a; Ren et al., 2017) such as EdgeBox (Zitnick and Dollár, 2014). Earlier studies (Mao et al., 2016; Rohrbach et al., 2016) score the generated candidate regions according to their appearances and spatial features along with features of the entire image. However, these methods fail to incorporate the interactions between objects, because the scoring process of each region proposal is isolated. More recent studies (Hu et al., 2017; Nagaraja et al., 2016) improve the performance with the aid of modeling relationships between objects.

2.2 Grounding Natural Language in Video

Analogous to spatial grounding in image, this work studies a similar problem—temporal natural language grounding in video. Earlier works (Yu and Siskind, 2013; Lin et al., 2014) learn the semantics of sentences, which are then matched to visual concepts via exploiting object appearance, motion and spatial relationships. However, they are limited to a small set of objects. Recently, larger datasets (Gao et al., 2017; Hendricks et al., 2017) are constructed to support more flexible groundings. The methods proposed in (Gao et al., 2017; Hendricks et al., 2017) learn a common embedding space shared by video segment features and sentence representations, in which their similarities are measured. Specifically, moment context network (MCN) (Hendricks et al., 2017) learns a shared embedding for video clip-level features and language features. The video features integrate local video features, global features, and temporal endpoint features. Cross-modal temporal regression localizer (CTRL) (Gao et al., 2017) contains four modules, specifically a visual encoder extracting clip-level features with context, a sentence encoder yielding its embedding through LSTM, a multimodal processing network generating the fused representations via element-wise operations, and a temporal regression network producing the alignment scores and location offsets. One limitation of those common space matching methods is that the video segment generation process is computationally expensive, as they carry out overlapping sliding window matching (Gao et al., 2017) or exhaustive search (Hendricks et al., 2017). Another weakness is that they exploit the relationships between textual and visual modalities by conducting a simple concatenation (Gao et al., 2017) or measuring a squared distance loss (Hendricks et al., 2017), which ignores the evolving fine-grained video-sentence interactions. In this paper, a novel model TGN is proposed to deal with the aforementioned limitations for the task of natural sentence grounding in video.

3 Approach

Given a long and untrimmed video sequence V and a natural sentence S, the NSGV task is to localize a video segment $V_s = \{f_t\}_{t=t_b}^{t_e}$ from V, beginning at t_b and ending at t_e, which corresponds to and expresses the same semantic meaning as the given sentence S. In order to perform the grounding, each video is represented as $V = \{f_t\}_{t=1}^{T}$, where T is the total number of frames and f_t denotes the feature representation of the t-th video frame. Similarly, each sentence is represented as $S = \{w_n\}_{n=1}^{N}$, where w_n is the embedding vector of the n-th word in the sentence and N denotes the total number of words.

We propose a novel model, namely Temporal GroundNet (TGN), to tackle the NSGV problem. As illustrated in Figure 2, TGN consists of three modules. 1) Encoder: visual and textual encoders are used to compose the video frame representations and word embeddings, respectively. 2) Interactor: a multimodal interactor learns the frame-by-word interactions between the video and sentence. 3) Grounder: a grounder generates the temporal localization in one single pass. Please note that these three modules are fully coupled together, which can thus be trained in an end-to-end fashion.

3.1 Encoder

With the obtained video frame features $V = \{f_t\}_{t=1}^{T}$ and word embeddings of the sentence $S = \{w_n\}_{n=1}^{N}$, we employ two long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) to sequentially process the two different modalities, *i.e.*, video and sentence, independently. Specifically, one LSTM sequentially models the video V, yielding the hidden states $\{\mathbf{h}_t^v\}_{t=1}^{T}$, while the other LSTM processes the sequential words in the sentence S, resulting in its corresponding hidden states $\{\mathbf{h}_n^s\}_{n=1}^{N}$. Owing to natural behaviors and characteristics of LSTMs, both $\{\mathbf{h}_t^v\}_{t=1}^{T}$ and $\{\mathbf{h}_n^s\}_{n=1}^{N}$ can encode and aggregate the contextual evidences (Wang and Jiang, 2016b) from the sequential video frame representations and word embeddings of the sentence, respectively, meanwhile casting aside the irrelevant information.

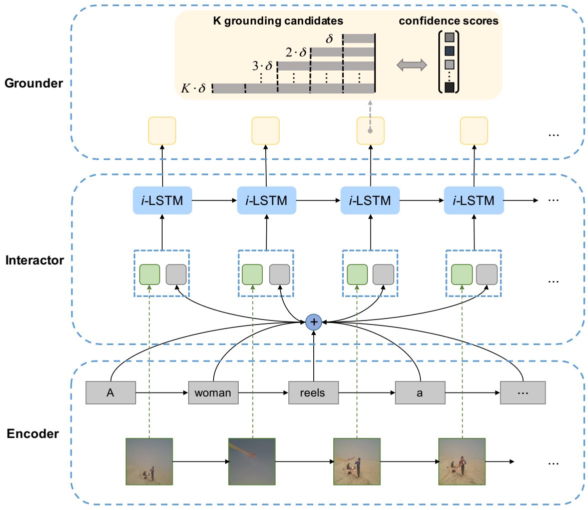

Figure 2: The architecture of our proposed TGN model. TGN consists of three modules. The visual and textural encoders aggregate the contextual evidences from the sequential video frame representations and word embeddings of the sentence, respectively. The multimodal interactor learns the fine-grained frame-by-word interactions between the video and sentence. The grounder yields the temporal grounding of the sentence in the video sequence via one single pass.

3.2 Interactor

Based on the hidden states of the video and sentence yielded from the leveraged encoders, we design a multimodal interactor to perform the frame-by-word interactions between the video and sentence. First, the frame-specific sentence feature is generated through summarizing the sentence hidden states by considering their relationships with the specific video frame at each time step. Afterwards, an interaction LSTM, dubbed i-LSTM, is performed to aggregate frame-by-word interactions.

3.2.1 Frame-Specific Sentence Feature

Directly operating on the clip-level and sentence-level features generated by the encoders cannot well exploit the frame-by-word relationships between video and sentence that evolve over time. Inspired by (Wang and Jiang, 2016a; Feng et al., 2018), we introduce one novel frame-specific sentence feature, which adaptively summarizes the hidden states of the sentence $\{\mathbf{h}_n^s\}_{n=1}^{N}$ with respect to the t-th video frame:

$$\mathbf{H}_t^s = \sum_{n=1}^{N} \alpha_t^n \mathbf{h}_n^s, \tag{1}$$

where $\mathbf{H}_t^s$ denotes the summarized sentence representation specified by the t-th video frame. At each time step t, we utilize the hidden state $\mathbf{h}_t^v$ to selectively attend the words and summarize them accordingly. The attention weight α_t^n encodes the degree to which the n-th word in the sentence is aligned with the t-th video frame. As the processing of video frames proceeds, the attention weights dynamically change regarding to the current video frame. As such, the generated frame-specific sentence features $\{\mathbf{H}_t^s\}_{t=1}^T$ consider the frame-by-word relationships between all the video frames and all the words in the sentence.

As the generation of frame-specific sentence feature is deeply coupled with the following interaction LSTM, we will explain the calculation of the attention weight α_t^n later.

3.2.2 Interaction LSTM (i-LSTM)

In order to accurately ground the sentence in a video, the multimodal interation behaviors between the video and sentence need to be comprehensively modeled. Previous approaches on multimodal interactions were limited to concatenation (Zhu et al., 2016), element-wise product or sum (Gao et al., 2017), and bilinear pooling (Fukui et al., 2016). These methods are not expressive enough since they ignore the evolving fine-grained interactions across video and sentence, particularly the frame-by-word interactions. In this paper, we propose a novel multimodal interaction model, which is realized by LSTM. We term it interaction LSTM (i-LSTM), which sequentially processes the video sequence frame by frame, holding deep interactions with the words in the sentence.

In order to well capture the complicated temporal interactions between the video and sentence, at each time step t, the input of the i-LSTM is formed by concatenating the t-th video hidden state $\mathbf{h}_t^v$ and the t-th frame-specific sentence feature $\mathbf{H}_t^s$ as: $\mathbf{r}_t = \mathbf{h}_t^v \parallel \mathbf{H}_t^s$. $\mathbf{r}_t$ is then fed into the i-LSTM unit to yield the t-th intermediate interaction status between the video and sentence:

$$\mathbf{h}_t^r = i\text{-LSTM}(\mathbf{r}_t, \mathbf{h}_{t-1}^r), \tag{2}$$

where $\mathbf{h}_t^r$ is the yielded hidden state, encoding the fine-grained interactions between the word and video frame. $\mathbf{h}_t^r$ will be further used to perform the grounding process. Due to the inherent properties and characteristics of LSTMs, important cues regarding to grounding up to the current stage will be "remembered", while non-essential ones will be "forgotten".

Now we go back to the generation of attention weight α_t^n in Eq. (1), based on the obtained visual hidden states h_t^v and textual hidden state h_n^s as well as the yielded interaction status h_{t-1}^r in the previous step. The widely used soft-attention mechanism (Xu et al., 2015a; Chen et al., 2017) is used to generate the attention weights in a frame-by-word manner. As aforementioned, the i-LSTM models the evolving frame-by-word interactions between the sentence and video. Therefore, the attention weight between the n-th word $\mathbf{h}_n^s$ and the t-th video frame $\mathbf{h}_t^v$ is determined by not only the content of the video and sentence but also their interaction status. Thus, we design one network to compute the relevance score of one video frame with respect to each word:

$$\beta_t^n = \mathbf{w}^\intercal \tanh(\mathbf{W}^S\mathbf{h}_n^s + \mathbf{W}^V\mathbf{h}_t^v + \mathbf{W}^R\mathbf{h}_{t-1}^r + \mathbf{b}) + c, \tag{3}$$

where vector $\mathbf{w}$, matrices $\mathbf{W}^*$, bias vector $\mathbf{b}$, and bias c are the network parameters to be learned. $\mathbf{h}_{t-1}^r$ is the hidden state of the i-LSTM at $t-1$ time step. The final word-level attention weights are obtained by:

$$\alpha_t^n = \frac{\exp(\beta_t^n)}{\sum_{j=1}^N \exp(\beta_t^j)}. \tag{4}$$

The obtained attention weight α_t^n is thereafter to generate the frame-specific sentence feature as in Eq. (1).

3.3 Grounder

In this section, we introduce the grounder, which works on the yielded interaction status $\mathbf{h}_t^r$ from i-LSTM, to localize the video segment that corresponds to the sentence. Our proposed grounder works in one single pass without introducing overlapping sliding windows, which thus results in a fast runtime. As shown in Figure 2, at each time step t, the grounder efficiently scores a set of K grounding candidates by considering multiple time scales (Buch et al., 2017) that end at time step t. Specifically, we use different K for different datasets, which is determined by the distribution of the lengths of all ground-truth groundings in a certain dataset. To simplify the following discussions, the lengths of K time scales are assumed to be an arithmetic sequence with the common difference δ and all the temporal candidates are sorted by increasing lengths. In other words, the length of the k-th candidate is $k\delta$. Note that all grounding candidates considered at time t have a fixed ending boundary.

Specifically, at each time step t, the grounder will classify each temporal candidate in consideration as a positive grounding or a negative one with respect to the given sentence. Considering multiple time scales, the grounder will generate the confidence scores $\mathbf{C}_t = (c_t^1, c_t^2, ..., c_t^K)$ that correspond to the set of K visual grounding candidates, all ending at time step t. The hidden state $\mathbf{h}_t^r$ generated by i-LSTM at time t, representing the interaction status between the sentence and video sequence up to the current position, is naturally suited to yield the confidence scores for the different time scales ending at time step t. In this paper, the confidence scores, indicating the sentence grounding, are generated by a fully-connected layer with sigmoid nonlinearity:

$$\mathbf{C}_t = \sigma(\mathbf{W}^K \mathbf{h}_t^r + \mathbf{b}_t^r), \tag{5}$$

where $\mathbf{W}^K$ and $\mathbf{b}_t^r$ are the corresponding parameters, and σ denotes the nonlinear sigmoid function.

3.4 Training

The training samples collected in $\mathcal{X}$ for NSGV are video-sentence pairs. Specifically, each video V is temporally associated with a set of sentence annotations: $A = \{(S_i, t_i^b, t_i^e)\}_{i=1}^M$, where M is the number of annotated sentences of the video, and S_i is a sentence description of a video clip, with t_i^b and t_i^e indicating the beginning and ending time in the video. Each training sample corresponds to a ground-truth matrix $\mathbf{y} \in \mathbb{R}^{T \times K}$ with binary entries. We use y_t^k to denote the (t, k)-th entry of the ground-truth matrix. y_t^k is interpreted as whether the k-th grounding candidate at time step t corresponds to the given natural sentence. Concretely, the entry y_t^k is set as 1, indicating that the corresponding video segment (ends at time step t with length $k\delta$) has a temporal Intersection-over-Union (IoU) with (t^b, t^e) larger than a threshold θ. Otherwise y_t^k is set as 0.

For a training pair $(V, S) \in \mathcal{X}$, the objective at time step t is given by a weighted binary cross entropy loss $\mathcal{L}(t, V, S)$:

$$-\sum_{k=1}^{K} w_0^k y_t^k \log c_t^k + w_1^k (1 - y_t^k) \log(1 - c_t^k), \tag{6}$$

where the weights w_0^k and w_1^k are calculated according to the frequencies of positive and negative samples in the training set with length $k\delta$. y_t^k is the ground-truth value and c_t^k denotes the prediction results by our proposed model.

Our TGN backpropagates at every time step t to learn all the parameters of the fully-coupled three modules: encoder, interactor, and grounder. The objective of all training video-sentence pairs $\mathcal{X}$ is defined as:

$$\mathcal{L}_{\mathcal{X}} = \sum_{(V,S)\in\mathcal{X}} \sum_{t=1}^{T} \mathcal{L}(t, V, S). \tag{7}$$

3.5 Inference

During the inference stage, given a testing video V and a sentence S, the textual and visual encoders first generate hidden states for each word and video frame, respectively. Then, the interactor sequentially goes through the video frame by frame to yield the frame-by-word interaction status. At each position t, a K-dimensional score vector $\mathbf{C}_t$ is generated by the grounder. Therefore, after processing the last frame in the video, a $T \times K$ score matrix is obtained for the whole video, with the (t, k)-th entry in the matrix indicating the probability that the video segment ended at position t with length $k\delta$ in video V corresponds to sentence S. Eventually, the evaluation is reduced to a ranking problem over all the grounding candidates based on the generated scores.

4 Experiments

In this section, we evaluate the effectiveness of our proposed TGN on the NSGV task. We begin by describing the datasets used for evaluation, followed by the introduction of the experimental settings including the baselines, configurations, as well as the evaluation metrics. Afterwards, we demonstrate the effectiveness of TGN by comparing with the state-of-the-art approaches and efficiency through a runtime test.

4.1 Datasets

We experiment on three publicly accessible datasets: DiDeMo (Hendricks et al., 2017), TACoS (Regneri et al., 2013), and ActivityNet Captions (Fabian Caba Heilbron and Niebles, 2015). These datasets consist of videos as well as their associated temporally annotated sentences.

DiDeMo[2] consists of 10464 25-50 second long videos. The same split provided by (Hendricks et al., 2017) is used for a fair comparison, with 33008, 4180, and 4022 video-sentence pairs for training, validation, and testing, respectively.

---

[2] `https://goo.gl/JpbAhg`.

TACoS[3] consists of 127 videos selected from the MPII Cooking Composite Activities video corpus (Rohrbach et al., 2012). The same split as in (Gao et al., 2017) is used, consisting of 10146, 4589, and 4083 video-sentence pairs for training, validation, and testing, respectively.

ActivityNet Captions[4] consists of $19,209$ videos amounting to 849 hours. The public split is used for our experiments, which has 37421, 17505, and 17031 video-sentence pairs for training, validation, and testing, respectively.

4.2 Experimental Settings

4.2.1 Baselines

We compare our proposed TGN against the following two state-of-the-art models, specifically, the MCN (Hendricks et al., 2017), CTRL (Gao et al., 2017), visual-semantic alignment with LSTM (VSA-RNN) (Karpathy and Li, 2015), and visual-semantic alignment with skip thought vector (VSA-STV) (Kiros et al., 2015). For fair comparisons, we compare the results of MCN on DiDeMo and the results of CTRL, VSA-RNN, VSA-STV on TACoS reported in their papers.

4.2.2 Evaluation Metrics

A grounding of one natural sentence in a video is considered as "correct" if its temporal IoU with the ground-truth boundary is above a threshold θ. To be consistent with the baselines, we adopt R@N, IoU=θ, and mean IoU (mIoU) as our evaluation metrics. R@N, IoU=θ represents the percentage of testing samples which have at least one of the top-N results with IoU larger than θ. mIoU means the average IoU over all testing samples.

4.2.3 Configurations

Generally, the video frame features are usually extracted with a time resolution. For the videos in DiDeMo and TACoS, we sample every 5 second as done by (Hendricks et al., 2017). As the videos in DiDeMo are 25-30 second long, the video feature length is reduced to 6. For videos in ActivityNet Captions, we sample every second. To extract visual features, we consider both appearance and optical flow features. Specifically, we study four widely-used visual features: VGG16 (Simonyan and Zisserman, 2014), C3D (Tran et al., 2015), Inception-V4 (Szegedy et al., 2017), and optical flow (Wang et al., 2016). Please note that when comparing with specific baseline methods, we use the same features as baseline methods, specifically, VGG16 and optical flow for MCN and C3D for CTRL, VSA-RNN, and VSA-STV.

Table 1: Performance comparisons of different methods on DiDeMo. The best performance for each metric entry is highlighted in boldface.

Method	R@1 IoU=1	R@5 IoU=1	mIoU
MFP	19.40	66.38	26.65
MCN-VGG16	13.10	44.82	25.13
MCN-Flow	18.35	56.25	31.46
MCN-Fusion	19.88	62.39	33.51
MCN-Fusion+TEF	28.10	78.21	41.08
TGN-VGG16	24.28	71.43	38.62
TGN-Flow	27.52	76.94	42.84
TGN-Fusion	**28.23**	**79.26**	**42.97**

For sentences, we tokenize each sentence by Stanford CoreNLP (Manning et al., 2014) and use the 300-D word embeddings from GloVe (Pennington et al., 2014) to initialize the models. The words not found in GloVe are initialized as zero vectors. The hidden state dimensions of all LSTMs (including the video, sentence, and interaction LSTMs) are set as 512. We use the Adam (Kingma and Ba, 2014) optimizer with β_1 = 0.5 and β_2 = 0.999. The initial learning rate is set to 0.001. We train the network for 200 iterations, and the learning rate is gradually decayed over time. The mini-batch size is set to 64.

4.3 Experimental Results and Analysis

4.3.1 Comparisons with State-of-the-Arts

Experiments on DiDeMo. Table 1 illustrates the performance comparisons on the DiDeMo dataset. In addition to MCN, we also compare with the baseline Moment Frequency Prior (MFP) in (Hendricks et al., 2017), which selects segments corresponding to the positions of videos in the training dataset with most annotations. First, TGN with different features can significantly outperforms the "prior baseline" MFP, which retrieves segments corresponding to the most common start and end points in the dataset. Second, it can be observed that with the same visual features, specifically VGG16 and optical flow, TGN significantly outperforms MCN. And the performance of TGN with optical flow is better than that with VGG16. One possible reason is that the videos in DiDeMo are relatively short, which only contain a single event. In such a case, the action information plays

---

[3] `https://goo.gl/ajmsva`.

[4] `https://goo.gl/N355bG`.

Table 2: Performance comparisons of different methods on TACoS. The best performance for each metric entry is highlighted in boldface.

Method	R@1 IoU=0.5	R@1 IoU=0.3	R@1 IoU=0.1	R@5 IoU=0.5	R@5 IoU=0.3	R@5 IoU=0.1
VSA-RNN	4.78	6.91	8.84	9.10	13.90	19.05
VSA-STV	7.56	10.77	15.01	15.5	23.92	32.82
CTRL-C3D	13.30	18.32	24.32	25.42	36.39	48.73
TGN-C3D	**18.90**	**21.77**	**41.87**	**31.02**	**39.06**	**53.40**

a more critical role. This finding is also consistent with (Hendricks et al., 2017). By fusing the results obtained by VGG16 and optical flow together, the performance can be further boosted, as demonstrated by TGN-Fusion and MCN-Fusion. Third, MCN introduces the temporal endpoint feature (TEF) as prior knowledge, which indicates when a segment occurs in a video. With TEF, the performance of MCN can be significantly improved. However, it is still inferior to our proposed TGN.

MCN is designed as an enumeration-based approach. Each video in the DiDeMo dataset is split into six five-second chunks which are considered as the time unit for localization. Therefore, in total there are only $\mathrm{C}_7^2 = 7 \times 6/2 = 21$ different ways of localization for DiDeMo videos. Therefore, although MCN can be effectively applied to videos with several chunks due to the small search space, it is not practical for untrimmed long videos. In the Section 4.3.3, we will evaluate and compare the efficiencies of MCN, CTRL, and our proposed TGN.

Experiments on TACoS. Table 2 illustrates the experimental results on TACoS. First, it can be observed that CTRL performs much better than VSA-RNN and VSA-STV. The reasons lie in twofold (Gao et al., 2017). On one hand, CTRL utilizes a multilayer alignment network to learn better alignment. On the other hand, VSA-RNN and VSA-STV do not encode temporal context information of video. Second, with the same visual feature, specifically C3D, TGN-C3D significantly outperforms CTRL-C3D. This is due to the fact that TGN exploits not only the contextual information but also the fine-grained interaction behaviors. More concretely, TGN considers the frame-by-word correlations by introducing an attentive combinations of the words in the sentence, where each weight encodes the degree to which the word is aligned with each specific frame. This mechanism is beneficial to capturing the informative semantics in the sentences for alignment.

Table 3: Performance comparisons of different visual features on ActivityNet Captions. The best performance for each metric entry is highlighted in boldface.

Feature	R@1 IoU=0.5	R@1 IoU=0.3	R@1 IoU=0.1	R@5 IoU=0.5	R@5 IoU=0.3	R@5 IoU=0.1
C3D	27.93	43.81	69.59	**44.20**	54.56	78.66
VGG16	23.90	42.24	65.76	40.17	51.82	76.21
Inception-V4	**28.47**	**45.51**	**70.06**	43.33	**57.32**	**79.10**

Table 4: Ablation studies on TACoS. The best performance for each metric entry is highlighted in boldface.

Feature	R@1 IoU=0.5	R@1 IoU=0.3	R@1 IoU=0.1	R@5 IoU=0.5	R@5 IoU=0.3	R@5 IoU=0.1
NA	5.53	7.67	24.23	15.20	18.94	41.25
NM	13.89	18.60	41.41	26.60	31.74	47.70
TGN	**18.90**	**21.77**	**41.87**	**31.02**	**39.06**	**53.40**

Experiments on ActivityNet Captions. Besides the two benchmarks, we also evaluate our model on the ActivityNet Captions dataset. Different CNNs are used to encode video visual information. Specifically, we consider VGG16, C3D, and Inception-V4. The results are included in Table 3. First, our proposed TGN can perform effectively on long untrimmed videos. Second, Inception-V4 performs generally better than VGG16 and C3D, which is consistent with the finding in (Canziani et al., 2016). Therefore, more powerful visual representations of video features will undoubtedly improve the the performance of our proposed TGN on the NSGV task.

Some qualitative results of our proposed TGN on ActiveityNet Captions dataset is illustrated in Figure 3. It can be observed that with different visual features, different grounding results are obtained. For the first and second examples, TGN with VGG16 and Inception-V4 generates more accurate groundings than that with C3D, while TGN with C3D yields more accurate grounding results for the third example. More specifically, our proposed TGN with VGG16 and Inception-V4 can well identify the visual information related with the sentence, i.e. "`A man in a red shirt claps his hands`".

4.3.2 Effect of Frame-by-Word Attention

We examine the effect of the frame-by-word attention in interactor. We ablate TGN into two other methods. 1) NA: There is no attention layer in this model. After obtaining the sequential hidden states of the sentence, mean pooling is used to generate the representation for the whole sentence.

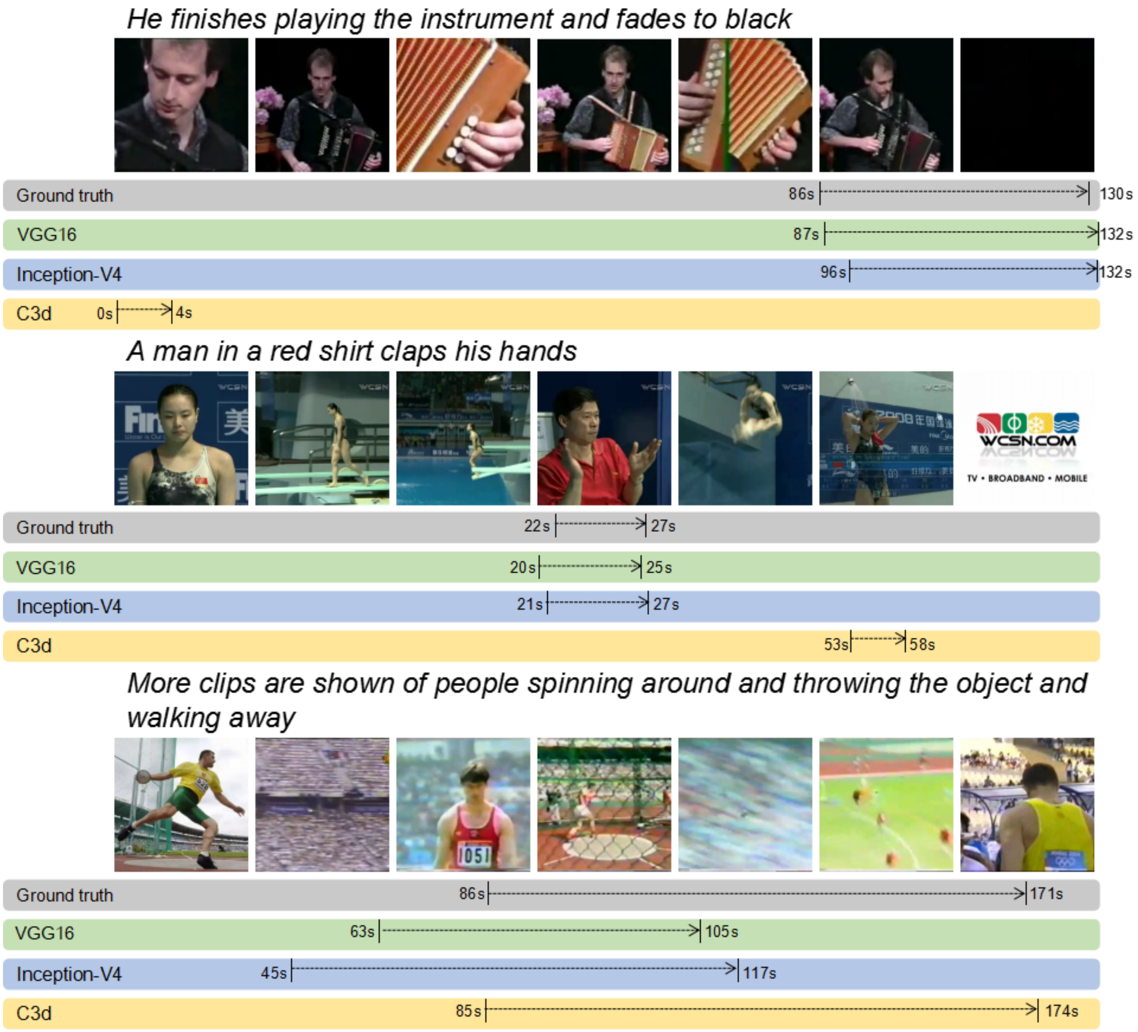

Figure 3: The qualitative grounding results of our TGN model on the ActiveityNet Captions dataset with different visual features.

Then the generated representation is concatenated with video representation, based on which the scores for multiple grounding candidates are predicted. 2) NM: The idea of generating frame-specific sentence feature is still reserved in the NM model. The difference between NM and TGN is that there is no interaction LSTM in NM. Specifically, when calculating the attention weight for each word as in Eq. (3), the hidden state $\mathbf{h}^r_{t-1}$ indicating the interaction status is not incorporated.

The quantitative results are displayed in Table 4. First, when the attention mechanism is applied (NM), the performance is improved as compared with utilizing mean pooling (NA) for sentence features. The better performance demonstrates that our assumption about the evolving frame-by-word correlations between two modalities is reasonable. This also indicates that it is necessary to discriminate the contribution of each word in a sentence to perform the NSGV task. Second, utilizing the interaction LSTM module (TGN) achieves better performance than simply concatenating the video representation and the attentive sentence representation (NM). This result indicates that the interaction LSTM yields better interaction status between these two modalities, which can thereby benefit the final grounding.

We provide some qualitative examples in Figure 4 for a better understanding of the frame-by-word attention. Meanwhile, the grounding results yielded by TGN-Fusion (considering both VGG16 and optical flow) are also illustrated. This experiment is designed to verify whether the frame-by-word attention mechanism in interactor is useful to highlight the representative concepts in the sentence. The attention weights α for two testing samples in DiDeMo are illustrated in Figure 4, where the darker the color is, the larger

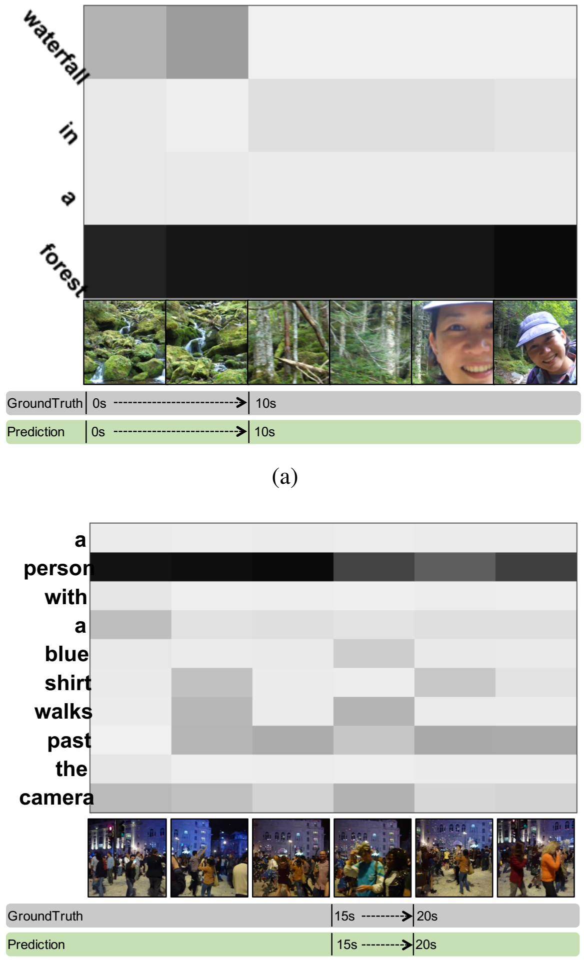

(a)

(b)

Figure 4: Visualization results on frame-by-word attention. The darker the color is, the larger its represented attention value is.

Table 5: Efficiency comparison in terms of frame per second.

	CTRL	MCN	TGN
FPS	562	286	1,363

the attention weight is. It can be observed that some words well match the frames. For example, in Figure 4 (a), the concept “`forest`” appears across all the video frames presenting an evenly distributed attention weights, while the other concept “`waterfall`” only presents in the first two frames. In addition to nouns, the adjective “`blue`” in Figure 4 (b) also receives relatively higher attention weights in relevant frames. Lastly, for stop words like “`a`”, “`the`” and “`in`”, their attention weights, which are very small, also present an even distribution.

4.3.3 Efficiency

We evaluate the efficiency of our proposed TGN, by comparing its runtime with MCN and CTRL on a Tesla M40 GPU. The efficiency is measured by frames per second (FPS) as shown in Table 5. Please not that the feature extraction time is excluded. It can be observed that our TGN model achieves much faster processing speeds, with 1,363 fps vs. 562 and 286 for CTRL and MCN, respectively. The reason mainly attributes to that the proposed TGN only process each video in one single pass without processing overlapped sliding windows.

5 Conclusion

In this paper, we focused on the task of natural sentence grounding in video that is believed to offer a comprehensive understanding of bridging computer vision and natural language processing. Towards this task, we proposed an end-to-end Temporal GroundNet (TGN) by incorporating the evolving fine-grained frame-by-word interactions across video-sentence modalities to generate a visual grounding tailored to each given natural sentence. Moreover, TGN performs efficiently, which only needs to process the video sequence in one single pass. Extensive experiments on three real-world datasets clearly demonstrate the effectiveness and efficiency of the proposed TGN.

References

Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. 2017. Sst: Single-stream temporal action proposals. In *CVPR*, pages 5534–5542.

Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. 2016. An analysis of deep neural network models for practical applications. *CoRR*, abs/1605.07678.

Jingyuan Chen, Xuemeng Song, Liqiang Nie, Xiang Wang, Hanwang Zhang, and Tat-Seng Chua. 2016. Micro tells macro: Predicting the popularity of micro-videos via a transductive model. In *MM*, pages 898–907.

Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *SIGIR*, pages 335–344.

Xinpeng Chen, Jingyuan Chen, Lin Ma, Jian Yao, Wei Liu, Jiebo Luo, and Tong Zhang. 2018. Fine-grained video attractiveness prediction using multimodal deep learning on a large real-world dataset. In *WWW*.

Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970.

Yang Feng, Lin Ma, Wei Liu, Tong Zhang, and Jiebo Luo. 2018. Video re-localization. In *ECCV*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*, pages 457–468.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017. TALL: temporal activity localization via language query. In *ICCV*, pages 5277–5285.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan C. Russell. 2017. Localizing moments in video with natural language. In *ICCV*, pages 5804–5813.

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

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2017. Modeling relationships in referential expressions with compositional modular networks. In *CVPR*, pages 4418–4427.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016. Natural language object retrieval. In *CVPR*, pages 4555–4564.

Zequn Jie, Xiaodan Liang, Jiashi Feng, Xiaojie Jin, Wen Lu, and Shuicheng Yan. 2016a. Tree-structured reinforcement learning for sequential object localization. In *NIPS*, pages 127–135.

Zequn Jie, Xiaodan Liang, Jiashi Feng, Wen Feng Lu, Francis Eng Hock Tay, and Shuicheng Yan. 2016b. Scale-aware pixelwise object proposal networks. *IEEE Trans. Image Processing*, 25(10):4525–4539.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*, pages 3294–3302.

Dahua Lin, Sanja Fidler, Chen Kong, and Raquel Urtasun. 2014. Visual semantic search: Retrieving videos via complex textual queries. In *CVPR*, pages 2657–2664.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL*, pages 55–60. The Association for Computer Linguistics.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *CVPR*, pages 11–20.

Varun K. Nagaraja, Vlad I. Morariu, and Larry S. Davis. 2016. Modeling context between objects for referring expression understanding. In *ECCV*, pages 792–807.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. In *ACL*, pages 1273–1283.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *TACL*, 1:25–36.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2017. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149.

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *ECCV*, pages 817–834.

Marcus Rohrbach, Michaela Regneri, Mykhaylo Andriluka, Sikandar Amin, Manfred Pinkal, and Bernt Schiele. 2012. Script data for attribute-based recognition of composite activities. In *ECCV*, pages 144–157.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284.

Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497.

Jasper R. R. Uijlings, Koen E. A. van de Sande, Theo Gevers, and Arnold W. M. Smeulders. 2013. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171.

Bairui Wang, Lin Ma, Wei Zhang, and Wei Liu. 2018a. Reconstruction network for video captioning. In *CVPR*.

Jingwen Wang, Wenhao Jiang, Lin Ma, Wei Liu, and Yong Xu. 2018b. Bidirectional attentive fusion with context gating for dense video captioning. In *CVPR*.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, pages 20–36.

Shuohang Wang and Jing Jiang. 2016a. Learning natural language inference with LSTM. In *NAACL*, pages 1442–1451.

Shuohang Wang and Jing Jiang. 2016b. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015a. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057.

Ran Xu, Caiming Xiong, Wei Chen, and Jason J. Corso. 2015b. Jointly modeling deep video and compositional text to bridge vision and language in a unified framework. In *AAAI*, pages 2346–2352.

Haonan Yu and Jeffrey Mark Siskind. 2013. Grounded language learning from video described with sentences. In *ACL*, pages 53–63.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L. Berg. 2018. Mattnet: Modular attention network for referring expression comprehension. *CoRR*, abs/1801.08186.

Yuke Zhu, Oliver Groth, Michael S. Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *CVPR*, pages 4995–5004.

C. Lawrence Zitnick and Piotr Dollár. 2014. Edge boxes: Locating object proposals from edges. In *ECCV*, pages 391–405.

PreCo: A Large-scale Dataset in Preschool Vocabulary for Coreference Resolution

Hong Chen[1], **Zhenhua Fan**[1, 2], **Hao Lu**[1], **Alan L. Yuille**[3] **and Shu Rong**[1]
[1]Preschool Lab, Yitu Tech
[2]Shandong Normal University
[3]Department of Cognitive Science and Department of Computer Science,
Johns Hopkins University
`{hong.chen,zhenhua.fan,hao.lv,shu.rong}@yitu-inc.com`
`ayuille1@jhu.edu`

Abstract

We introduce PreCo, a large-scale English dataset for coreference resolution. The dataset is designed to embody the core challenges in coreference, such as entity representation, by alleviating the challenge of low overlap between training and test sets and enabling separated analysis of mention detection and mention clustering. To strengthen the training-test overlap, we collect a large corpus of 38K documents and 12.5M words which are mostly from the vocabulary of English-speaking preschoolers. Experiments show that with higher training-test overlap, error analysis on PreCo is more efficient than the one on OntoNotes, a popular existing dataset. Furthermore, we annotate singleton mentions making it possible for the first time to quantify the influence that a mention detector makes on coreference resolution performance. The dataset is freely available at `https://preschool-lab.github.io/PreCo/`.

1 Introduction

Coreference resolution, identifying mentions that refer to the same entities, is an important NLP problem. Resolving coreference is critical for many downstream applications, such as reading comprehension, translation, and text summarization. Identifying a mention depends not only on its lexicons but also its contexts, and requires representations of all the entities before the mention. This is still a challenging task for the approaches based on the cutting-edge word2vec-like lexical representation. For example, it is hard to identify the mention "he" between two entities "Tom" and "Jerry" because they have almost the same word embeddings.

A number of datasets have been proposed to study the coreference resolution problem, such as MUC (Hirschman and Chinchor, 1997), ACE (Doddington et al., 2004), and OntoNotes (Pradhan et al., 2012). The most popular one is OntoNotes, and recent work on coreference resolution (Clark and Manning, 2016a,b; Lee et al., 2017; Peters et al., 2018) evaluated their models on it. Other datasets were rarely studied after OntoNotes was published.

Previous work (Sadat Moosavi and Strube, 2017) suggests that the overlap between training and test sets makes significant impact on the performance of current coreference resolvers. In OntoNotes, which has relatively low training-test overlap, this impact is mixed together with the core challenges of coreference resolution. For example, consider the failure of referencing "them" to "the wounded" in "..., the wounded were carried off so fast and it was difficult to count them". It is hard to tell whether the algorithm can succeed if the currently low-frequency phrase "the wounded" has not been seen enough times in the training set. From a machine learning perspective, high overlap is needed to ensure that the training and test datasets have similar statistics.

Another limitation of OntoNotes is that it only has annotations for non-singleton mentions, while singleton mentions are not annotated. Most of the algorithms for coreference resolution have two steps: mention detection and mention clustering (Wiseman et al., 2016; Clark and Manning, 2016a,b). The lack of singleton mention annotations makes training and evaluation of mention detectors more difficult.

To address both limitations of OntoNotes, we build a new dataset, PreCo. To alleviate the negative impact of low training-test overlap, we restrict the data domain and collect a sufficient amount of data to achieve a relatively high training-test overlap. Restricting the data domain is a common way to enable better studies of unsolved NLP tasks, such as language modeling (Hill et al., 2015) and

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 172–181
Brussels, Belgium, October 31 - November 4, 2018.

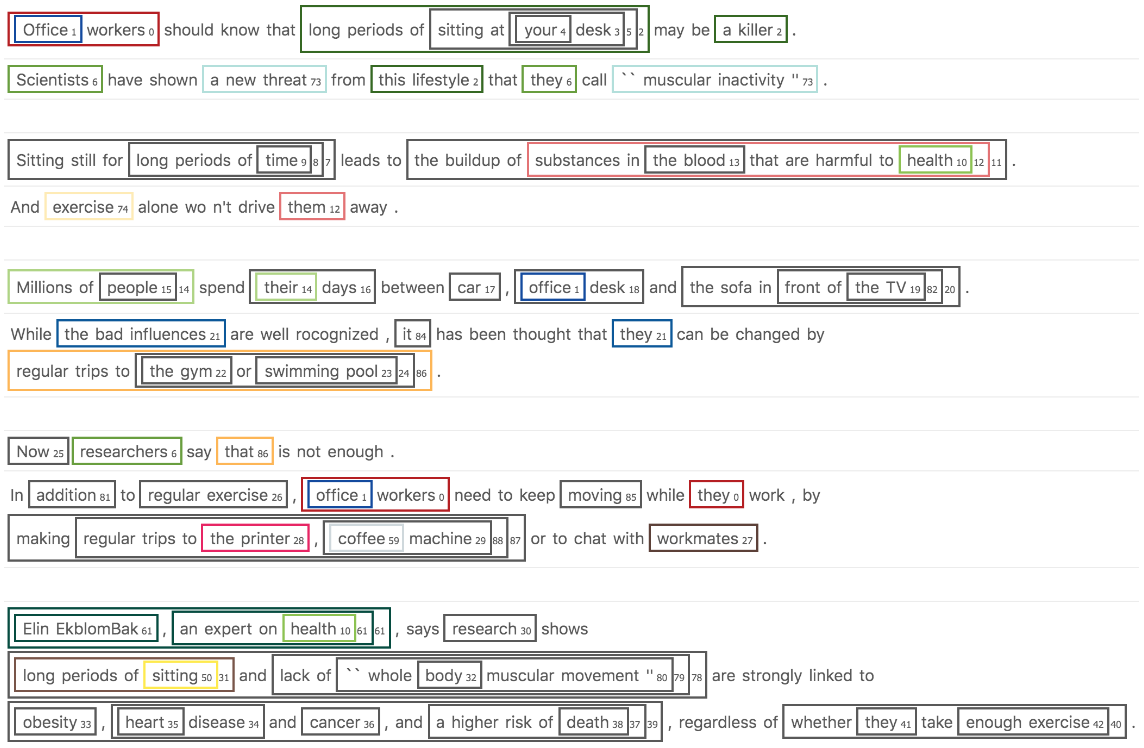

Figure 1: An Example from PreCo. In the example, mentions are indicated by boxes, and mention clustering is indicated by the subscripted numbers. If two mentions have the same number, they refer to the same entity.

visual question answering (Johnson et al., 2017).

We select our data from English reading comprehension tests for middle and high school Chinese students, which has several advantages. On one hand, the vocabulary size is appropriate. The English vocabulary of a typical Chinese high school student contains about 3000 commonly used words. This is similar to the vocabulary of a preschool English-speaking child (Wikipedia, 2018). Most words from the English tests are in this limited vocabulary. On the other hand, it is practical to collect enough data of this type from the Internet. With 12.5M words, PreCo is about 10 times larger than OntoNotes. Large scale datasets, e.g. ImageNet (Deng et al., 2009), SQuAD (Rajpurkar et al., 2016), have played an important role for driving computer vision and NLP forward.

We use the rate of out-of-vocabulary (OOV) words between training and test sets to measure their overlap. PreCo shows much higher training-test overlap than OntoNotes by having an OOV rate of 0.8%, which is about 1/3 of OntoNotes's 2.1%. At the same time, PreCo presents a good challenge for coreference resolution research since its documents are in the open domain and have various writing styles. We test a state-of-the-art system (Peters et al., 2018) on PreCo and get an F1 score of 81.5. However, a modest human performance (87.9, which will be described in 4.1) is much higher, verifying there remain challenges.

To help training and evaluation of mention detection, we annotate singleton mentions in PreCo. Besides singleton mentions, we follow most other annotation rules of OntoNotes to label the new dataset. We show that in a state-of-the-art coreference resolution system (Peters et al., 2018), we can improve the model performance from 77.3 to 81.6 F1 on a training set of 2.5K PreCo documents by using an oracle mention detector, and the remaining gap of 18.4 F1 to the perfect 100 F1 can only be reduced by improving mention clustering. This indicates that future work should concern more about mention clustering than mention detection.

The advantages of our proposed dataset over existing ones in coreference resolution can be summarized as follows:

- Its OOV rate is about 1/3 of OntoNotes.
- It has about 10 times larger corpus size than OntoNotes.
- It has annotated singleton mentions.

2 Related Work

Existing Datasets. The first two resources for coreference resolution study were MUC-6 and MUC-7 (Hirschman and Chinchor, 1997). The MUC datasets are too small for training and testing, containing a total of 127 documents with 65K words. The next standard dataset was ACE (Doddington et al., 2004) which has a much larger corpus of 1M words. But its annotations are restricted to a small subset of entities and are less consistent. OntoNotes (Pradhan et al., 2012) was presented to overcome those limitations. Machine learning based approaches, especially deep learning based, benefitted from this well annotated and large-scale (1.3M words) dataset. Continuous research on OntoNotes over the past 6 years improved performance by 10 F1 score (Durrett and Klein, 2013; Peters et al., 2018). Datasets after OntoNotes, such as WikiCoref (Ghaddar and Langlais, 2016), are seldom studied. Therefore, we mainly compare PreCo with OntoNotes in this paper. With a much larger scale, PreCo builds on the advantages of OntoNotes. Some of these existing datasets also have corpus in other languages, but we just focus on coreference resolution in English.

Out-of-domain Evaluation. (Sadat Moosavi and Strube, 2017) show that if coreference resolvers mainly rely on lexical representation, as it is the case in state-of-the-art ones, they are weak at generalizing to unseen domains. Even in the seen domains, the low degree of overlap for non-pronominal mentions between the training and test sets cause serious deterioration of coreference resolution performance. As a conclusion, (Sadat Moosavi and Strube, 2017) suggested that out-of-domain evaluation is a must in the literature. But we think the problem can be relieved by expanding the training data for the target domains to increase overlap, so that the field can pay more attention to the other challenges of coreference resolution.

Data Simplification. Many simplified datasets were built to enable better study on unsolved tasks. Such simplifications can guide researchers to the core problems and make data collection easier. For example, (Hill et al., 2015) introduced the Children's Book Test to distinguish the task of predicting syntactic function words from that of predicting low-frequency words for language model. The dataset helped them to develop a generalizable model with explicit memory representations. The reading comprehension dataset SQuAD (Rajpurkar et al., 2016) imposes the constraint that every answer is always a segment of the input text. This constraint benefits both labeling and evaluation of the dataset, which has significant influences in terms of benchmarks. Similarly, the reinforcement learning literature develops algorithms by studying games instead of the real world environment (Mnih et al., 2013). We hope that, with high training-test overlap, PreCo can serve as a valuable resource for research on coreference resolution.

3 Dataset Creation

We discuss the data collection and annotation in this section. The overview of the process is shown in Figure 2.

3.1 Corpus Collection

We crawl English tests from several web sites. The web pages often contain the full English tests in a lot of formats. We build an annotation website and hire annotators to manually extract the relevant contents. We have a total of 80 part-time Chinese annotators, most of whom are university students. They are required to have a minimum score in standard English tests. During annotation training, the annotators read the annotation rules, and take several practice tasks, in which they annotate sample articles, and their results are compared with ground truth side by side for them to study. Before formal annotation, the annotators will need to pass an assessment.

Some data cleaning is done during annotation, such as unifying paragraph separators, etc. The questions with answers in these tests are also extracted for future research. Finally, we use NLTK's sentence and word tokenizer (Bird et al., 2009) to tokenize the crawled text.

In addition to having annotators manually clean the data, we also use heuristic rules to further clean the data. For example, in some cases the whitespaces between two words are missing. We use a spell checker to identify and correct most of these cases. We also use heuristic rules to fix some sentence partition boundaries, e.g., to make sure opening quotes are placed at the beginning of a sentence, instead of being wrongly placed at the end of a previous sentence (closing quotes are handled similarly).

In addition to the crawled data, we include the

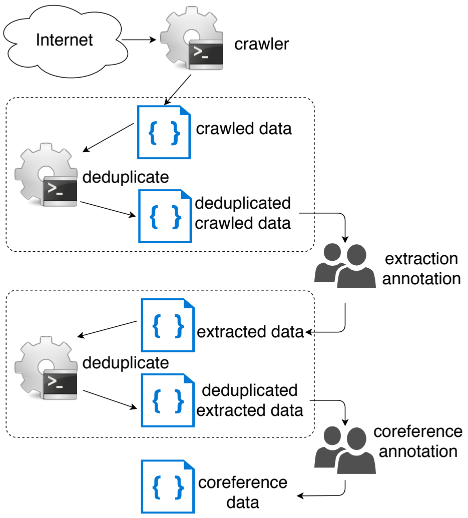

Figure 2: Overview of dataset creation.

documents from the RACE dataset (Lai et al., 2017). RACE is a reading comprehension dataset from English tests for middle and high school Chinese students, which has similar types of data sources as PreCo. About 2/3 of PreCo documents are from the RACE dataset.

Since documents are from several data sources, we want to remove duplicated documents, and documents that are not exactly the same but have a high rate of repetitions. The similarity of two documents D_1 and D_2 is estimated using the bag-of-words model. Assume S_1 and S_2 are bag-of-words multisets to represent the two documents. The similarity between D_1 and D_2 is defined as $\max(\frac{|S_1 \cap S_2|}{|S_1|}, \frac{|S_1 \cap S_2|}{|S_2|})$. If the similarity between two documents are larger than 0.9, we remove the shorter one. This process is referred as *deduplicate* in Figure 2.

3.2 Data Partition

The dataset has a total of 38K documents. We use 500 documents for the development set, 500 documents for the test set, and the rest 37K documents for the training set. The development and test documents were randomly selected from RACE's development and test sets.

3.3 Coreference Annotation and Refinement

We manually annotate coreferences on these documents. The annotation rules are slightly different from OntoNotes (Pradhan et al., 2012). We modify some of the rules to make the definition of coreference more consistent and easier to be understood by the annotators. The major differences are listed in Table 1. Figure 1 shows an example document in PreCo with annotations.

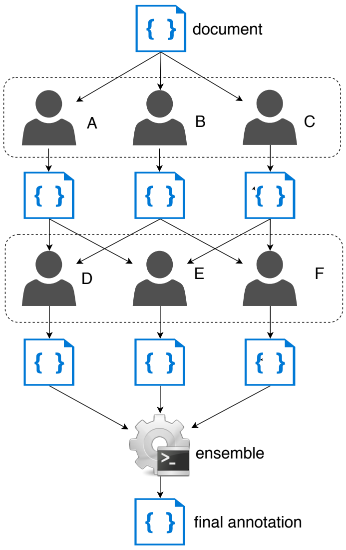

Figure 3: Process of annotation refinement. A document is firstly annotated by 3 annotators A, B, and C, independently. Then another annotator D merges annotations from A and B. Similarly, annotator E merges annotations from A and C, and annotator F merges annotations from B and C. Finally, annotations from D, E and F are merged using an ensemble algorithm.

Good quality control of annotation is essential, since the rules are complicated and coreference resolution depends on meticulous reading of the whole document over and over. We found that annotators get low recall and insufficient precision mainly because of negligence, as opposed to the lack of annotation rules or other ambiguities. For example, two co-referred mentions could be far apart and require careful searches, and an annotator may miss it. Therefore we further refine annotations as shown in Figure 3. Annotators can think about the complicated inconsistent cases when merging annotations, and the voting process will fix some errors while preserving the mentions and coreferences that are found only once by individual annotators.

The quality of different annotation processes is shown in Table 2. OntoNotes took 2 individual annotations for each document and got an adjudicated version based on them. Taking the adjudicated version as ground truth, the average MUC score (Vilain et al., 1995) [1] of individual annota-

---

[1] MUC score is one of the metrics to evaluate the quality of coreference resolution.

Type	Example	OntoNotes	PreCo
verbs	Sales [grew] 10%. [The growth] is exciting.	Verbs can be coreferred.	Usually, verbs cannot be coreferred. Certain gerunds can.
generic mentions	[Parents] are usually busy. [Parents] should get involved.	Generic mentions can only be coreferred by pronouns.	Generic mentions can be coreferred directly.
non-proper modifiers	[Wheat] is important. [Wheat] fields are everywhere.	Non-proper modifiers cannot be coreferred.	Non-proper modifiers can be coreferred as generic mentions.
copular structures	[John] is [a good teacher].	The referent and the attribute cannot be coreferred.	The referent and the attribute can be coreferred.
appositives	[[John]$_a$, [a linguist I know]$_b$]$_c$, ...	Sub-spans are not coreferred with the whole-span. a and b are not coreferent with c.	Sub-spans are coreferred with the whole-span. a and b are coreferred with c.
misc.	The [U.S.] policy ... [Secretary of State] [Colin Powell] ...	Nationality acronyms and job titles in appositives cannot be coreferred.	Nationality acronyms and all job titles can be coreferred.

Table 1: Major differences of annotation rules between PreCo and OntoNotes. The annotation rules of OntoNotes are described in (OntoNotes Guidelines)

tions is 89.6, and the inter-annotator MUC score is 83.0. The corresponding numbers for PreCo are 85.3 and 77.5. The actual gap of individual annotation quality between OntoNotes and PreCo is not as large as it looks like. Note that, OntoNotes's two individual coreference annotations of each document are based on the same syntactic annotations of the document, so they could be more consistent than PreCo's which are annotated on raw text. Therefore, if we want to fairly compare PreCo with OntoNotes, we should take into account OntoNotes's inter-annotator consistency of syntactic parsing annotations. As it has a rough upper bound of 98.5 F1 score according to the reannotation of English Treebank on OntoNotes by the principal annotator a year after the original annotation (Weischedel et al., 2011), we could infer that the individual annotation quality of PreCo is quite close to OntoNotes.

Labeling the whole dataset is costly because each annotation from scratch or comparison takes an average of about 10 minutes. Prompts from an algorithm do not help since they do not speed up the annotation much but instead introduce biases. We observed some biases when using an algorithm to help annotation. We have two models, M_1 and M_2, and we have a test set T which is annotated manually, and a test set T' which uses prompts from model M_1 to help annotation. While M_1 and M_2 have similar performance on T, M_1's performance is much higher than M_2's on T', which shows the biases.

Because of limited annotation resources, we have only finished the refinements on the development and test sets with the process shown in Figure 3. We refine the training set annotations as follows: for each document, two annotators annotate it separately, and a third annotator compares and merges the two annotations. We use a training set of 2.5K documents to quantify the impact of this annotation refinement to model performance. Table 3 shows the model performances of the training set that is annotated once, and the training set of the merged annotation. The performance difference is quite significant. Furthermore, the difference is consistent with Table 2: the "AB-merge" model has a similar precision as the "Once" model, but it has a much higher recall. It indicates that a further refinement of the training set such as DEF-voting could be essential. A more interesting question is: how to make the definition of coreference more consistent and executable? We leave it as future work.

Process	Avg. Prec	Avg. Rec	Avg. F1
Once	87.3	71.7	78.7
ABC-voting	93.5	76.1	83.9
AB-merge	87.5	88.3	87.9
DEF-voting	100.0	100.0	100.0

Table 2: Annotation quality. DEF-voting is taken as the ground truth to evaluate other annotation processes. The annotation "AB-merge" is merged by annotator G, who is different from D, E and F.

Annotation	Avg. Prec	Avg. Rec	Avg. F1
Once	79.3	69.1	73.9
AB-merge	78.1	76.5	77.3

Table 3: The annotation quality's impact on model performance. Each row shows the development set performance of the EE2E-Coref model (training details in Section 4.1) trained by data of different annotation quality. Each training set contains 2.5K documents. In the training set "Once", each document is annotated by one annotator. In the training set "AB-merge", each document is annotated by two annotators independently, and the annotations are compared and merged by a third annotator.

3.4 Dataset Properties

Table 4 shows some properties of OntoNotes and PreCo. As intended, PreCo has a lower OOV rate than OntoNotes. For a training set with vocabulary $\mathcal{V}$ and a test set with n tokens $[t_1, t_2, ..., t_n]$, ignoring the tokens with non-alphabetic characters, the OOV rate is defined by:

$$\frac{\sum_i o(t_i)}{n}, \text{where } o(t_i) = \begin{cases} 0 & \text{if } t_i \in \mathcal{V} \\ 1 & \text{if } t_i \notin \mathcal{V} \end{cases}$$

The OOV rate can be extended to the rate of low-frequency words which also indicates the training-test overlap, by simply replacing $\mathcal{V}$ in the definition above with the non-low-frequency vocabulary of the training set. We find that the OOV rate is consistent to the rates of low-frequency words in different levels. So we use the OOV rate for convenience.

In PreCo, about 50.8% of the mentions are singleton mentions. Figure 4 shows the distribution of cluster sizes within non-singleton clusters. The distribution is similar between OntoNotes and PreCo.

4 Analysis

To verify our assumption that PreCo embodies the core challenges of coreference, we evaluate a strong baseline coreference resolver on it. Specifically, we (i) estimate the room for improvement of the baseline system to show that the dataset is challenging, (ii) study the impact of training-test overlap to model performance and error analysis to show the advantages of PreCo, and (iii) quantitatively evaluate the mention detector to understand the bottlenecks of the coreference resolution system.

Property	OntoNotes	PreCo
Training documents	2.8K	37K
Training tokens	1.3M	12.2M
Dev-test documents	0.7K	1K
Dev-test tokens	0.3M	0.3M
Tokens per document	467	330
OOV rate	2.1%	0.8%
Non-singleton mentions		
Mention length	2.29	2.02
Mention density	0.12	0.16
Cluster size	4.40	4.49
Cluster density	0.027	0.035
Singleton mentions		
Mention length	N/A	3.32
Mention density	N/A	0.16
Singleton mention rate	N/A	50.8%

Table 4: Properties of OntoNotes and PreCo. The mention (cluster) density is defined by: number of mentions (clusters) / number of tokens.

4.1 Baseline Performance

We use the end-to-end neural coreference resolver, E2E-Coref (Lee et al., 2017), enhanced by the deep contextualized word representations (Peters et al., 2018) as the baseline system, and we refer to this system as EE2E-Coref. This is the state-of-the-art model on OntoNotes, achieving a test average F1 score of 70.4, which is the main evaluation metric for coreference resolution. The metric is computed by averaging the F1 of MUC, B^3, and $CEAF_{\phi 4}$, which are three metrics of coreference resolution that have different focuses.

Our implementation EE2E-Coref[2] gets 81.5 Avg. F1 score on PreCo. We follow the setting of most hyperparameters on OntoNotes and do grid-search for the decay parameter of the learning rate and the size of the hidden layers on the development set, since these two hyperparameters are relatively sensitive to the scale of the training data. The F1 score increment from OntoNotes to PreCo is probably due to the higher overlap between the training and test sets in PreCo.

---

[2] It gets an F1 score of 70.0±0.3 on OntoNotes, slightly lower than the F1 score reported in the original paper.

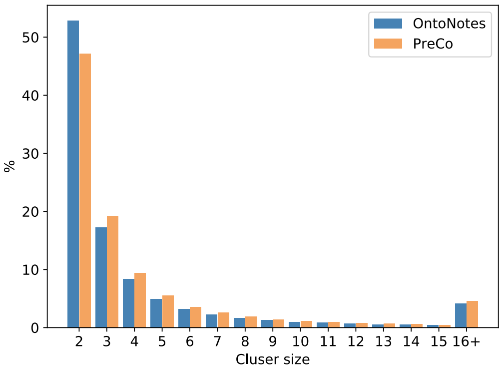

Figure 4: Distribution of cluster sizes within non-singleton clusters. We ignore singleton clusters in this figure so that it is easier to compare between OntoNotes and PreCo.

[<His father> and he] get off the car. [They] find the old man lying near the taxi. The banana skin is near him. The old man looks at [them] and says, "Teach [**your**] child to throw the banana skin to the right place!"
He gave his last few coins to [a beggar], but then he saw <another one>, and forgot that he did not have any money. He asked <the man> if <he> would like to have lunch with him, and [**the beggar**] accepted, so they went into a small restaurant and had a good meal.
[Holmes] and <Dr. Watson> went on a camping trip. After a good meal and a bottle of wine, they lay down in a tent for the night and went to sleep. Some hours later, Holmes woke up and pushed [**his friend**].

Table 5: Error cases of EE2E-Coref on PreCo. Each bold mention is incorrectly referred to the entity in []s. The mentions of its gold entity are in <>s.

We demonstrate three typical error cases made by EE2E-Coref on PreCo in Table 5. Coreference resolution in these cases requires good understanding of multiple sentences, which is an open problem in NLP. A capable entity representation for "them", "another one" or "Dr. Watson" may help to resolve these error cases. We also compare the performance of EE2E-Coref with human performance to estimate the room for improvement on PreCo. As described in Section 3.4, human annotators get low recall mostly due to negligence. So we use the AB-merge annotation to estimate human's ability on coreference resolution. The gap of performance between model and human is 6.4 F1 score, from 81.5 to 87.9. The actual gap is larger, since AB-merge still has some missed coreference annotations due to negligence. This shows that the dataset is challenging and encourages future research. The error cases show the challenges as well.

Note that PreCo is not a general purpose dataset. Our motivation of designing PreCo is to make it easier to improve coreference resolution algorithms, e.g., to make error analysis easier. It is not a goal of PreCo to generalize well on corpus from other domains. Furthermore, we find that there are a certain amount of annotation errors in the development and test sets. We suggest that researchers working on PreCo should be careful about these errors, especially after a model gets F1 score beyond 90.0.

4.2 Impact of Training-test Overlap

Training-test overlap makes significant impact on error analysis. Consider an error case of coreference resolution, if there are low-frequency words in the related mentions, then it will be hard to tell whether the algorithm can succeed if the words has not been seen enough times in the training set. We call an error case LFW if there are low-frequency words[3] in its related mentions[4]. Therefore, the lower LFW rate a training set contains, the more precisely it may expose the drawbacks of the algorithm.

To study the impact of training-test overlap, actually, the training-dev overlap, we pick different subsets from the training data and evaluate the models trained on them. At first, we control overlap by picking different sizes of the training data randomly. Figure 5(a) shows that, as the training data size grows, the OOV rate, which is the overlap indicator, decreases and the F1 score of EE2E-Coref increases significantly. Figure 5(b) shows that when training set size increases, the OOV rate and the LFW rate drop together. Then, to remove the impact of data size, we pick training sets which have a fixed size but different overlaps with the development set vocabulary. The OOV rates and F1 scores of these subsets are shown in Figure 5(c). This experiment verifies the positive cor-

---

[3] In our experiments, a word is defined as low-frequency if it appears in the training set less than 10 times.

[4] There are 3 kinds of error cases of coreference resolution: false-new, false-link and wrong-link. In our experiments, the related mentions include: the current mention in all 3 kinds of cases, the nearest gold antecedent in false-new and wrong-link and the false referred antecedent in false-link and wrong-link.

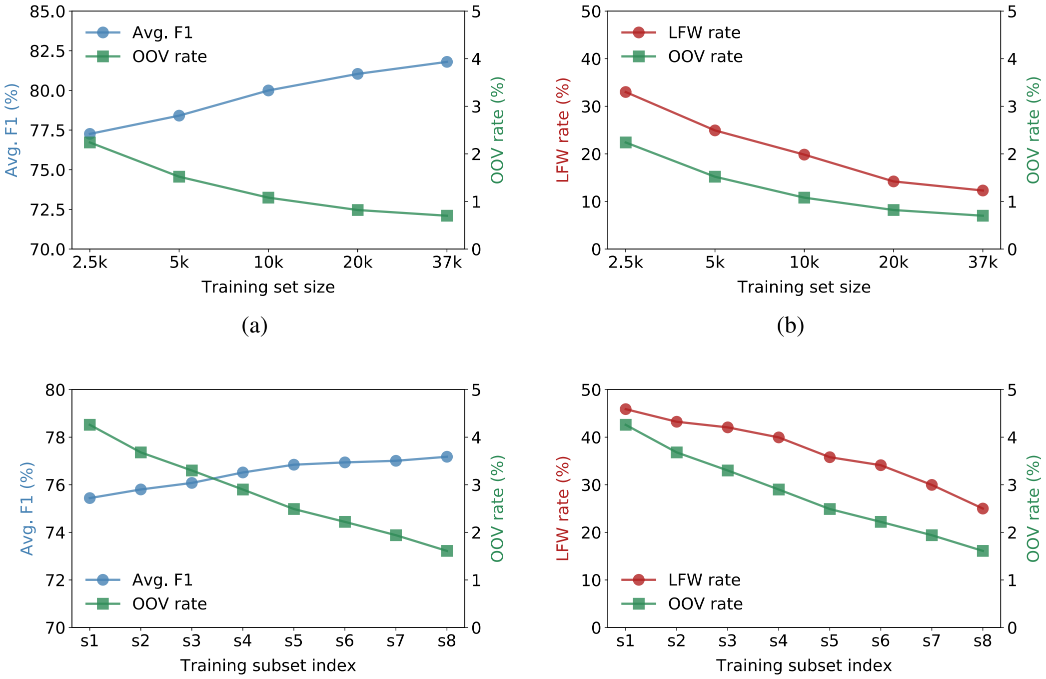

(a) (b)

(c) (d)

Figure 5: Impact of training-dev overlap. (a) and (b) show the impact of training set sizes. (c) and (d) show the impact of the training-dev OOV rate, when the training sets have the same size of 2.5K documents. The 8 subsets, s1-s8, consist of documents ranked by their overlaps with the development set vocabulary.

relation between training-dev overlap and coreference resolution performance suggested by (Sadat Moosavi and Strube, 2017). Figure 5(d) shows that for training sets with the same size, the OOV rate and the LFW rate also drop together.

We observe that the training set of 2.5K documents in Figure 5(a) has a higher model performance than all the training sets in Figure 5(c). This is not expected. One hypothesis is that the lower performance in Figure 5(c) is due to the smaller diversity of these training sets, which are selected to have certain training-dev OOV rates.

The training-dev LFW rate of OntoNotes is 34.8%. As a comparison, the number for PreCo is 12.3%. A subset of PreCo with a similar token number to OntoNotes has a LFW rate of 33.0%. This indicates that research of coreference algorithms on PreCo will be much more efficient than on OntoNotes. Even if we can ignore the LFW error cases, there are others related to low-frequency word senses, phrases and sentence structures, which are hard to filter out. They will also obscure the error analysis. It is reasonable to believe that training-dev overlap impacts the rate of these error cases in a similar way to impact LFW rate.

4.3 Mention Detection

Since most coreference systems consist of a mention detection module and a mention clustering module, an important question is: with a perfect mention detection module, what is the model performance on coreference resolution? The answer would help us understand the bottlenecks of the entire system, by quantifying the impact of the mention detection module on the final F1 score. (Lee et al., 2017) gave an answer by taking ground truth non-singleton mentions as the input of the coreference resolver for both training and evaluation, assuming that the perfect mention detector can also make perfect anaphoricity decisions, e.g., to decide whether a mention should be linked to an antecedent. But this assumption can be violated since mention detectors usually take local information but anaphoricity decisions usually need more context, nearly as much as entity identification. The anaphoricity decisions should be made in the mention clustering module.

Mention	OntoNotes	PreCo
detected	66.7	77.3
*all	N/A	81.6
*non-singleton	85.2	89.2

Table 6: Coreference resolution performances on development set under different mention detection qualities. A prefixed * denotes ground truth. The model trained on OntoNotes is E2E-Coref (Lee et al., 2017) while the one trained on PreCo is EE2E-Coref. The PreCo training set contains the same 2.5K documents as in Table 3.

We argue that a better way to answer the question is to take all ground truth mentions (including singletons) for coreference. This operation is not feasible in OntoNotes since it does not have annotations for singleton mentions. We do this on PreCo and the results are shown in Table 6. There is an obvious difference between the F1 scores achieved with all gold mentions and non-singleton gold mentions. Therefore, the room for improvement by better mention detection is not as enormous as suggested in (Lee et al., 2017). The major challenge remained in coreference resolution is mention clustering.

5 Conclusion

In this paper, we propose a large-scale coreference resolution dataset to overcome the limitations of existing ones. Our dataset, PreCo, features higher training-test overlap, about 10 times larger scale than previous datasets, and singleton mention annotations. By evaluating a state-of-the-art coreference resolver, we show that there is a wide gap between the model and human performance, which demonstrated challenges of the dataset. We verified the expectation that PreCo's higher training-test overlap helps research on coreference resolution. For the first time, we quantified the impact of mention detector to the entire system, thanks to our singleton mention annotations. We make the dataset public, and hope it will stimulate further research on coreference resolution.

References

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982.

Abbas Ghaddar and Phillippe Langlais. 2016. Wikicoref: An english coreference-annotated corpus of wikipedia articles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Lynette Hirschman and Nancy Chinchor. 1997. Muc-7 coreference task definition. In *Proceedings of MUC-7*. Applications International Corporation.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *CoRR*, abs/1707.07045.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. NIPS Deep Learning Workshop 2013.

OntoNotes Guidelines. 2007. Ontonotes english coreference guidelines.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. *ArXiv e-prints*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning (CoNLL 2012)*, Jeju, Korea.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Nafise Sadat Moosavi and Michael Strube. 2017. Lexical features in coreference resolution: To be used with caution.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pages 45–52, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. Ontonotes: A large training corpus for enhanced processing. *Handbook of Natural Language Processing and Machine Translation. Springer*.

Wikipedia. 2018. Vocabulary development. [Online; accessed 18-April-2018].

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 994–1004. Association for Computational Linguistics.

Adversarial Transfer Learning for Chinese Named Entity Recognition with Self-Attention Mechanism

Pengfei Cao[1,2], **Yubo Chen**[1], **Kang Liu**[1,2], **Jun Zhao**[1,2] and **Shengping Liu**[3]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Beijing Unisound Information Technology Co., Ltd, Beijing, 100028, China
{pengfei.cao, yubo.chen, kliu, jzhao}@nlpr.ia.ac.cn, liushengping@unisound.com

Abstract

Named entity recognition (NER) is an important task in natural language processing area, which needs to determine entities boundaries and classify them into pre-defined categories. For Chinese NER task, there is only a very small amount of annotated data available. Chinese NER task and Chinese word segmentation (CWS) task have many similar word boundaries. There are also specificities in each task. However, existing methods for Chinese NER either do not exploit word boundary information from CWS or cannot filter the specific information of CWS. In this paper, we propose a novel adversarial transfer learning framework to make full use of task-shared boundaries information and prevent the task-specific features of CWS. Besides, since arbitrary character can provide important cues when predicting entity type, we exploit self-attention to explicitly capture long range dependencies between two tokens. Experimental results on two different widely used datasets show that our proposed model significantly and consistently outperforms other state-of-the-art methods.

1 Introduction

The task of named entity recognition (NER) is to recognize the named entities in given text. NER is a preliminary and important task in natural language processing (NLP) area and can be used in many downstream NLP tasks, such as relation extraction (Bunescu and Mooney, 2005), event extraction (Chen et al., 2015) and question answering (Yao and Van Durme, 2014). In recent years, numerous methods have been carefully studied for NER task, including Hidden Markov Models (HMMs) (Bikel et al., 1997), Support Vector Machines (SVMs) (Isozaki and Kazawa, 2002) and Conditional Random Fields (CRFs) (Lafferty et al., 2001). Currently, with the development of deep learning, neural networks (Lample et al., 2016; Peng and Dredze, 2016; Luo and Yang, 2016) have been introduced to NER task. All these methods need to **determine entities boundaries** and **classify them into pre-defined categories**.

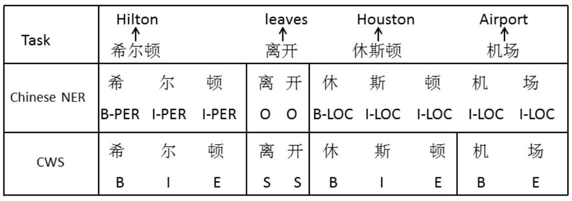

Task	Hilton ↑ 希尔顿			leaves ↑ 离开		Houston ↑ 休斯顿			Airport ↑ 机场	
Chinese NER	希	尔	顿	离	开	休	斯	顿	机	场
	B-PER	I-PER	I-PER	O	O	B-LOC	I-LOC	I-LOC	I-LOC	I-LOC
CWS	希	尔	顿	离	开	休	斯	顿	机	场
	B	I	E	S	S	B	I	E	B	E

Figure 1: An example of illustrating the similarities and specificities between Chinese NER and CWS.

Although great improvements have been achieved by these methods on Chinese NER task, some issues still have not been well addressed. One significant drawback is that there is only a very small amount of annotated data available. Weibo NER dataset (Peng and Dredze, 2015; He and Sun, 2017a) and Sighan2006 NER dataset (Levow, 2006) are two widely used datasets for Chinese NER task, containing 1.3k and 45k training examples, respectively. On the two datasets, the highest F1 scores are 48.41% and 89.21%, respectively. As a basic task in NLP area, the performance is not satisfactory. Fortunately, Chinese word segmentation (CWS) task is to recognize word boundaries and the amount of supervised training data for CWS is abundant compared with NER. There are many similarities between Chinese NER task and CWS task, which we call task-shared information. As shown in Figure 1, given a sentence “希尔顿离开休斯顿机场 (Hilton leaves Houston Airport)”, the two tasks have the same boundaries for some words such as “希尔顿 (Hilton)” and “离开 (leaves)”, while Chinese NER has more coarse-grained boundaries

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 182–192
Brussels, Belgium, October 31 - November 4, 2018.

than CWS task for certain word such as “休斯顿机场 (Houston Airport)” in the example of Figure 1, which we call task-specific information. In order to incorporate word boundary information from CWS task into NER task, Peng and Dredze (2016) propose a joint model that performs Chinese NER with CWS task. However, their proposed model only focuses on task-shared information between Chinese NER and CWS, and ignores filtering the specificities of each task, which will bring noise for both of the tasks. For example, the CWS task splits “休斯顿机场 (Houston Airport)” into “休斯顿 (Houston)” and “机场 (Airport)”, while the NER task takes “休斯顿机场 (Houston Airport)” as a whole entity. Thus, how to exploit task-shared information and prevent the noise brought by CWS task to Chinese NER task is a challenging problem.

Another issue is that most proposed models cannot explicitly model long range dependencies when predicting entity type. Though bidirectional long short term memory (BiLSTM) can learn long-distance dependencies, it cannot conduct direct connections between arbitrary two characters. As shown in Figure 1, if the model only focuses on the word “希尔顿 (Hilton)”, it can be a person or organization. However, when the model explicitly captures the dependencies between “希尔顿 (Hilton)” and “离开 (leaves)”, it is easy to classify “希尔顿 (Hilton)” into “person” category. Context information is very crucial for determining the entity type. While in the sentence “我将住在希尔顿 (I will be staying at the Hilton)”, the entity type of “希尔顿 (Hilton)” is “organization”. Thus, how to better capture the global dependencies of the whole sentence is another challenging problem.

To address the above problems, we propose an adversarial transfer learning framework to integrate the task-shared word boundary information into Chinese NER task in this paper. The adversarial transfer learning is incorporating adversarial training into transfer learning. To better capture long range dependencies and synthesize the information of the sentence, we extend self-attention mechanism into the framework. Specifically, we try to improve Chinese NER task performance by incorporating shared boundary information from CWS task. To prevent the specific information of CWS task from lowering the performance of the Chinese NER task, we introduce adversarial training to ensure that the Chinese NER task only exploits task-shared word boundary information. Then, for tackling the long range dependency problems, we utilize self-attention to synthesize the hidden representation of BiLSTM. Finally, we evaluate our model on two different widely used Chinese NER datasets. Experimental results show that our proposed model achieves better performance than other state-of-the-art methods and gains new benchmarks.

In summary, the contributions of this paper are as follows:

- We propose an adversarial transfer learning framework to incorporate task-shared word boundary information from CWS task into Chinese NER task. To our best knowledge, it is the first work to apply adversarial transfer learning method into NER task.

- We introduce self-attention mechanism into our model, which aims to capture the global dependencies of the whole sentence and learn inner structure features of sentence.

- We conduct our experiment on two different widely used Chinese NER datasets, and the experimental results demonstrate that our proposed model significantly and consistently outperforms previous state-of-the-art methods. We release the source code publicly for further research[1].

2 Related Work

NER Many methods have been proposed for NER task. Early studies on NER often exploit SVMs (Isozaki and Kazawa, 2002), HMMs (Bikel et al., 1997) and CRFs (Lafferty et al., 2001), heavily relying on feature engineering. Zhou et al. (2013) formulate Chinese NER as a joint identification and categorization task. In recent years, neural network models have been introduced to NER task (Collobert et al., 2011; Huang et al., 2015; Peng and Dredze, 2016). Huang et al. (2015) exploit BiLSTM to extract features and feed them into CRF decoder. After that, the BiLSTM-CRF model is usually exploited as the baseline. Lample et al. (2016) use a character LSTM to represent spelling characteristics. In addition, Wang et al. (2017) propose a gated convolutional neural network (GCNN) model for Chinese NER. Peng and Dredze (2016) propose a joint model for Chinese

---

[1] https://github.com/CPF-NLPR/AT4ChineseNER

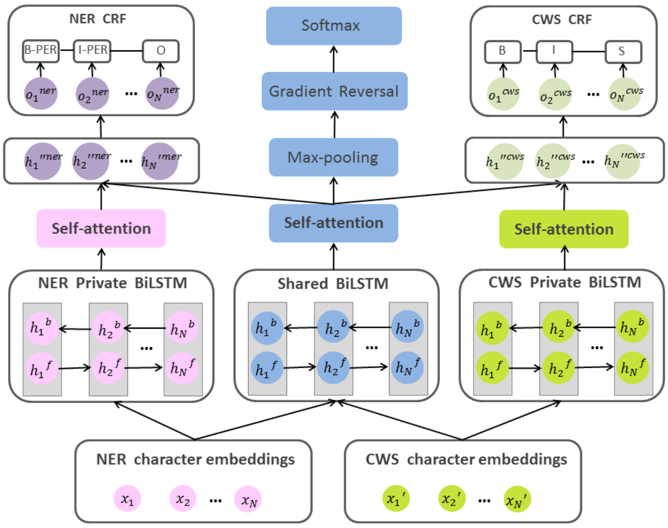

Figure 2: The general architecture of our proposed model. The left and right part are Chinese NER space and CWS private space, respectively, including embedding layer, feature extractor (Private BiLSTM), self-attention and CRF layer. The middle part is shared space consisting of feature extractor (Shared BiLSTM), self-attention and task discriminator.

NER, which are jointly trained with CWS task. However, the specific features brought by CWS task can lower the performance of the Chinese NER task.

Adversarial Training Adversarial networks have achieved great success in computer vision (Goodfellow et al., 2014; Denton et al., 2015). In NLP area, adversarial training has been introduced for domain adaptation (Ganin and Lempitsky, 2014; Zhang et al., 2017; Gui et al., 2017), cross-lingual transfer learning (Chen et al., 2016; Kim et al., 2017), multi-task learning (Chen et al., 2017; Liu et al., 2017) and crowdsourcing learning (Yang et al., 2018). Bousmalis et al. (2016) propose shared-private model in domain separation network. Different from these works, we exploit adversarial network to jointly train Chinese NER task and CWS task, aiming to extract task-shared word boundary information from CWS task. To our knowledge, it is the first work to apply adversarial transfer learning framework to Chinese NER task.

Self-Attention Self-attention has been introduced to machine translation by Vaswani et al. (2017) for capturing global dependencies between input and output and achieves state-of-the-art performance. For language understanding task, Shen et al. (2017) exploit self-attention to learn long range dependencies. Tan et al. (2017) apply self-attention to semantic role labelling task and achieve state-of-the-art results. We are the first to introduce self-attention mechanism to Chinese NER task.

3 Method

In this paper, we propose a novel adversarial transfer learning framework that will learn task-shared word boundary information from CWS task, filter specific information of CWS and explicitly capture the long range dependencies between arbitrary two characters in sentence. The architecture of our proposed model is illustrated in Figure 2. The model mainly consists of five components: embedding layer, shared-private feature extractor, self-attention, task-specific CRF and task discriminator. In the following sections, we will describe each part of our proposed model in detail.

3.1 Embedding Layer

Similar to other neural network models, the first step of our proposed model is to map discrete characters into the distributed representations. For a given Chinese sentence $\mathbf{x} = \{c_1, c_2, \ldots, c_N\}$ from Chinese NER dataset or CWS dataset, we lookup embedding vector from pre-trained embedding matrix for each character c_i as $\mathbf{x}_i \in \mathbb{R}^{d_e}$.

3.2 Shared-Private Feature Extractor

Long short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a variant of recurrent neural network (RNN) (Elman, 1990), which enables to address the gradient vanishing and exploding problems in RNN via introducing gate mechanism and memory cell. The unidirectional LSTM only leverages information from the past, ignoring the future information. In order to incorporate information from both sides of sequence, we adopt BiLSTM to extract features. Specially, the hidden state of BiLSTM could be expressed as follows:

$$\overrightarrow{\mathbf{h}_i} = \overrightarrow{\mathrm{LSTM}}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{x}_i) \quad (1)$$

$$\overleftarrow{\mathbf{h}_i} = \overleftarrow{\mathrm{LSTM}}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{x}_i) \quad (2)$$

$$\mathbf{h}_i = \overrightarrow{\mathbf{h}_i} \oplus \overleftarrow{\mathbf{h}_i} \quad (3)$$

where $\overrightarrow{\mathbf{h}_i} \in \mathbb{R}^{d_h}$ and $\overleftarrow{\mathbf{h}_i} \in \mathbb{R}^{d_h}$ are the hidden states of the forward and backward LSTM at position i, respectively. $\oplus$ denotes concatenation operation.

As shown in Figure 2, we propose a shared-private feature extractor, which assigns a private BiLSTM layer and shared BiLSTM layer for task $k \in \{NER, CWS\}$. The private BiLSTM layer is used to extract task-specific features, and the shared BiLSTM layer is used to learn task-shared word boundaries. Formally, for any sentence in dataset of task k, the hidden states of shared and private BiLSTM layer can be computed as follows:

$$\mathbf{s}_i^k = \mathrm{BiLSTM}(\mathbf{x}_i^k, \mathbf{s}_{i-1}^k; \theta_s) \quad (4)$$

$$\mathbf{h}_i^k = \mathrm{BiLSTM}(\mathbf{x}_i^k, \mathbf{h}_{i-1}^k; \theta_k) \quad (5)$$

where θ_s and θ_k are the shared BiLSTM parameters and private BiLSTM parameters of task k, respectively.

3.3 Self-Attention

Inspired by the self-attention applied to machine translation (Vaswani et al., 2017) and semantic role labelling (Tan et al., 2017), we exploit self-attention to explicitly learn the dependencies between any two characters in sentence and capture the inner structure information of sentence. In this paper, we adopt the multi-head self-attention mechanism. $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N\}$ denotes the output of private BiLSTM. Correspondingly, $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N\}$ is the output of shared BiLSTM. We will take the self-attention in private space as example to illustrate how it works. The scaled dot-product attention can be precisely described as follows:

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{\mathbf{d}}})\mathbf{V} \quad (6)$$

where $\mathbf{Q} \in \mathbb{R}^{N\times 2d_h}$, $\mathbf{K} \in \mathbb{R}^{N\times 2d_h}$ and $\mathbf{V} \in \mathbb{R}^{N\times 2d_h}$ are query matrix, keys matrix and value matrix, respectively. In our setting, $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{H}$. $\mathbf{d}$ is the dimension of hidden units of BiLSTM, which equals to $2d_h$.

Multi-head attention first linearly projects the queries, keys and values h times by using different linear projections. Then h projections perform the scaled dot-product attention in parallel. Finally, these results of attention are concatenated and once again projected to get the new representation. Formally, the multi-head attention can be expressed as follows:

$$head_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \quad (7)$$

$$\mathbf{H}' = (head_i \oplus \ldots \oplus head_h)\mathbf{W}_o \quad (8)$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{2d_h\times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{2d_h\times d_k}$ and $\mathbf{W}_i^V \in \mathbb{R}^{2d_h\times d_k}$ are trainable projection parameters and $d_k = 2d_h/h$. $\mathbf{W}_o \in \mathbb{R}^{2d_h\times 2d_h}$ is also trainable parameter.

3.4 Task-Specific CRF

For a sentence in dataset of task k, we compute the final representation via concatenating the representations from private space and shared space after self-attention layer:

$$\mathbf{H}''^k = \mathbf{H}'^k \oplus \mathbf{S}'^k \quad (9)$$

where $\mathbf{H}'^k$ and $\mathbf{S}'^k$ are the outputs of private self-attention and shared self-attention of task k, respectively.

Considering the dependencies between successive labels, we exploit CRF (Lafferty et al., 2001) to inference tags instead of making tagging decisions using $\mathbf{h}_i''$ independently. Due to the difference of labels, we introduce a specific CRF layer for each task. Given a sentence $\mathbf{x} = \{c_1, c_2, \ldots, c_N\}$ with a predicted tag sequence $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$, the CRF tagging process can be formalized as follows:

$$\mathbf{o}_i = \mathbf{W}_s\mathbf{h}_i'' + \mathbf{b}_s \quad (10)$$

$$s(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N}(\mathbf{o}_{i,y_i} + \mathbf{T}_{y_{i-1},y_i}) \quad (11)$$

$$\bar{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathbf{Y}_x} s(\mathbf{x}, \mathbf{y}) \quad (12)$$

where $\mathbf{W}_s \in \mathbb{R}^{|T|\times 4d_h}$ and $\mathbf{b}_s \in \mathbb{R}^{|T|}$ are trainable parameters. $|T|$ denotes the number of output labels. $\mathbf{o}_{i,y_i}$ represents the score of the y_i-th tag of the character c_i. $\mathbf{T}$ is a transition score matrix which defines the scores of two successive labels. $\mathbf{Y}_x$ represents all candidate tag sequences for given sentence $\mathbf{x}$. In decoding, we use Viterbi algorithm to get the predicted tag sequence $\bar{\mathbf{y}}$.

For training, we exploit negative log-likelihood objective as the loss function. The probability of the ground-truth label sequence is computed by:

$$p(\hat{\mathbf{y}}|\mathbf{x}) = \frac{e^{s(\mathbf{x},\hat{\mathbf{y}})}}{\sum_{\widetilde{\mathbf{y}}\in\mathbf{Y}_x} e^{s(\mathbf{x},\widetilde{\mathbf{y}})}} \tag{13}$$

where $\hat{\mathbf{y}}$ denotes the ground-truth label sequence. Given T training examples $(\mathbf{x}^{(i)};\hat{\mathbf{y}}^{(i)})$, the loss function L_{Task} can be defined as follows:

$$L_{Task} = -\sum_{i=1}^{T} \log p(\hat{\mathbf{y}}^{(i)}|\mathbf{x}^{(i)}) \tag{14}$$

We use gradient back-propagation method to minimize the loss function.

3.5 Task Discriminator

Inspired by adversarial networks (Goodfellow et al., 2014), we incorporate adversarial training into shared space to guarantee that specific features of tasks do not exist in shared space. We propose a task discriminator to estimate which task the sentence comes from. Formally, the task discriminator can be expressed as follows:

$$\mathbf{s}'^{k} = \text{Maxpooling}(\mathbf{S}'^{k}) \tag{15}$$

$$D(\mathbf{s}'^{k};\theta_d) = \text{softmax}(\mathbf{W}_d\mathbf{s}'^{k} + \mathbf{b}_d) \tag{16}$$

where θ_d indicates the parameters of task discriminator. $\mathbf{W}_d \in \mathbb{R}^{K\times 2d_h}$ and $\mathbf{b}_d \in \mathbb{R}^{K}$ are trainable parameters. K is the number of tasks.

Besides the task loss L_{Task}, we introduce an adversarial loss L_{Adv} to prevent specific features of CWS task from creeping into shared space. The adversarial loss trains the shared model to produce shared features such that the task discriminator cannot reliably recognize which task the sentence comes from. The adversarial loss can be computed as follows:

$$L_{Adv} = \min_{\theta_s}(\max_{\theta_d}\sum_{k=1}^{K}\sum_{i=1}^{T_k}\log D(E_s(\mathbf{x}_k^{(i)}))) \tag{17}$$

where θ_s denotes the trainable parameters of shared BiLSTM. E_s denotes the shared feature extractor. T_k is the number of training examples of task k. $\mathbf{x}_k^{(i)}$ is the i-th example of task k. There is a minimax optimization that the shared BiLSTM generates a representation to mislead the task discriminator and the discriminator tries its best to correctly determine the type of task.

We add a gradient reversal layer (Ganin and Lempitsky, 2014) below the softmax layer to address the minimax optimization problem. In the training phrase, we minimize the task discriminator errors, and through gradient reversal layer the gradients will become opposed sign to adversarially encourage the shared feature extractor to learn task-shared word boundary information. After training phrase, the shared feature extractor and task discriminator reach a point where the discriminator cannot differentiate the tasks according to the representations learned from shared feature extractor.

3.6 Training

The final loss function of our proposed model can be written as follows:

$$L = L_{NER}\cdot I(\mathbf{x}) + L_{CWS}\cdot(1 - I(\mathbf{x})) + \lambda L_{Adv} \tag{18}$$

where λ is a hyper-parameter. L_{NER} and L_{CWS} can be computed via Eq.14. $I(\mathbf{x})$ is a switching function to identify which task the input comes from. It is defined as follows:

$$I(\mathbf{x}) = \begin{cases} 1, & if \quad \mathbf{x}\in\mathcal{D}_{NER} \\ 0, & if \quad \mathbf{x}\in\mathcal{D}_{CWS} \end{cases} \tag{19}$$

where $\mathcal{D}_{NER}$ and $\mathcal{D}_{CWS}$ are Chinese NER training corpora and CWS training corpora, respectively.

In the training phrase, at each iteration, we first select a task from $\{NER, CWS\}$ in turn. Then, we sample a batch of training instances from the given task to update the parameters. We use Adam (Kingma and Ba, 2014) algorithm to optimize the final loss function. Since Chinese NER task and CWS task may have different convergence rate, we repeat the above iterations until early stopping according to the Chinese NER task performance.

4 Experiments

4.1 Datasets

To evaluate our proposed model on Chinese NER, we experiment on two different widely used datasets, including Weibo NER dataset (WeiboNER) (Peng and Dredze, 2015; He and Sun,

Dataset	Task	# Train sent	# Dev sent	# Test sent
WeiboNER	Chinese NER	1350	270	270
SighanNER	Chinese NER	41728	4636	4365
MSR	CWS	86924	—	3985

Table 1: Statistics of the datasets.

Models	P(%)	R(%)	F1(%)
CRF (Peng and Dredze, 2015)	56.98	25.26	35.00
CRF+word (Peng and Dredze, 2015)	64.94	25.77	36.90
CRF+character (Peng and Dredze, 2015)	57.89	34.02	42.86
CRF+character+position (Peng and Dredze, 2015)	57.26	34.53	43.09
Joint(cp) (main) (Peng and Dredze, 2015)	57.98	35.57	44.09
Pipeline Seg.Repr.+NER (Peng and Dredze, 2016)	64.22	36.08	46.20
Jointly Train Char.Emb (Peng and Dredze, 2016)	63.16	37.11	46.75
Jointly Train LSTM Hidden (Peng and Dredze, 2016)	63.03	38.66	47.92
Jointly Train LSTM+Emb (main) (Peng and Dredze, 2016)	63.33	39.18	48.41
BiLSTM+CRF+adversarial+self-attention	55.72	**50.68**	**53.08**

Table 2: NER results for named entities on the original WeiboNER dataset (Peng and Dredze, 2015). There are three blocks. The first two blocks contain the main and simplified models proposed by Peng and Dredze (2015) and Peng and Dredze (2016), respectively. The last block lists the performance of our proposed model.

2017a) and SIGHAN2006 NER dataset (SighanNER) (Levow, 2006). We use the MSR dataset (from SIGHAN2005) for CWS task.

The WeiboNER is annotated with four entity types (person, location, organization and geo-political entities), including named entities and nominal mentions. The SighanNER is simplified Chinese, which contains three entity types (person, location and organization). For WeiboNER, we use the same training, development and testing splits as Peng and Dredze (2015). Since the SighanNER does not have development set, we sample 10% data of training set as development set. We use MSR dataset to improve the performance of the Chinese NER task. Table 1 gives the details of the three datasets.

4.2 Settings

For evaluation, we use the Precision (P), Recall (R) and F1 score as metrics in our experiment.

For hyper-parameter configurations, we adjust them according to the performance on development set of Chinese NER task. We set the character embedding size d_e to 100. The dimensionality of LSTM hidden states d_h is 120. The initial learning rate is set to 0.001. The loss weight coefficient λ is set to 0.06. We set the dropout rate to 0.3. The number of projections h is 8. We set the batch size of SighanNER and WeiboNER as 64 and 20, respectively.

For trainable parameters initialization, we use xavier initializer (Glorot and Bengio, 2010) to initialize parameters. The character embeddings used in our experiment are pre-trained on Baidu Encyclopedia corpus and Weibo corpus by using word2vec toolkit (Mikolov et al., 2013).

4.3 Compared with State-of-the-art Methods

In this section, we will give the experimental results of our proposed model and previous state-of-the-art methods on WeiboNER dataset and SighanNER dataset, respectively.

4.3.1 Evaluation on WeiboNER

We compare our proposed model with the latest models on WeiboNER dataset. Table 2 shows the experimental results for named entities on the original WeiboNER dataset.

In the first block of Table 2, we give the performance of the main model and baselines proposed by Peng and Dredze (2015). They propose a CRF-based model to jointly train the embeddings with NER task, which achieves better results than pipeline models. In addition, they consider the po-

Models	Named Entity			Nominal Mention			Overall
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	F1(%)
Peng and Dredze (2015)	74.78	39.81	51.96	71.92	53.03	61.05	56.05
Peng and Dredze (2016)	66.67	47.22	55.28	74.48	54.55	62.97	58.99
He and Sun (2017a)	66.93	40.67	50.60	66.46	53.57	59.32	54.82
He and Sun (2017b)	61.68	48.82	54.50	74.13	53.54	62.17	58.23
BiLSTM+CRF+adv+self-attention	59.51	**50.00**	54.34	71.43	47.90	57.35	58.70

Table 3: Experimental results on the updated WeiboNER dataset (He and Sun, 2017a). There are two blocks. The first block is the performance of latest models. The second block reports the performance of our proposed model. With the limited length of the page, we use "adv" to denote "adversarial".

Models	P(%)	R(%)	F1(%)
Chen et al. (2006)	91.22	81.71	86.20
Zhou et al. (2006)	88.94	84.20	86.51
Luo and Yang (2016)	91.30	87.22	89.21
BiLSTM+CRF+adversarial+self-attention	**91.73**	**89.58**	**90.64**

Table 4: Results on SighanNER dataset. There are two blocks. The first block reports the result of previous methods. The second block gives the performance of our proposed model.

sition of each character in a word to train character and position embeddings.

In the second block of Table 2, we report the performance of the main model and baselines proposed by Peng and Dredze (2016). Aiming to incorporate word boundary information into the NER task, they propose an integrated model that can joint training CWS task, improving the F1 score from 46.20% to 48.41% as compared with pipeline model (Pipeline Seg.Repr.+NER).

In the last block of Table 2, we give the experimental result of our proposed model (BiLSTM+CRF+adversarial+self-attention). We can observe that our proposed model significantly outperforms other models. Compared with the model proposed by Peng and Dredze (2016), our method gains 4.67% improvement in F1 score. Interestingly, WeiboNER dataset and MSR dataset are different domains. The WeiboNER dataset is social media domain, while the MSR dataset can be regard as news domain. The improvement of performance indicates that our proposed adversarial transfer learning framework may not only learn task-shared word boundary information from CWS task but also tackle the domain adaptation problem.

We also conduct an experiment on the updated WeiboNER dataset. Table 3 lists the performance of the latest models and our proposed model on the updated dataset. In the first block of Table 3, we report the performance of the latest models. The model proposed by Peng and Dredze (2015) achieves F1 score of 56.05% on overall performance. He and Sun (2017b) propose an unified model for Chinese NER task to exploit the data from out-of-domain corpus and in-domain unlabelled texts. The unified model improves the F1 score from 54.82% to 58.23% compared with the model proposed by He and Sun (2017a).

In the second block of Table 3, we give the result of our proposed model. It can be observed that our proposed model achieves a very competitive performance. Compared with the latest model proposed by He and Sun (2017b), our model improves the F1 score from 58.23% to 58.70% on overall performance. The improvement demonstrates the effectiveness of our proposed model.

4.3.2 Evaluation on SighanNER

Table 4 lists the comparisons on SighanNER dataset. We observe that our proposed model achieves new state-of-the-art performance.

In the first block, we give the performance of previous methods for Chinese NER task on SighanNER dataset. Chen et al. (2006) propose a character-based CRF model for Chinese NER task. Zhou et al. (2006) introduce a pipeline model, which first segments the text with character-level CRF model and then applies word-level CRF to tag. Luo and Yang (2016) first train a word segmenter and then use word segmentation as addi-

Models	SighanNER			WeiboNER		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
BiLSTM+CRF	89.84	88.42	89.13	58.99	44.93	51.01
BiLSTM+CRF+transfer	90.60	89.19	89.89	60.00	46.03	52.09
BiLSTM+CRF+adversarial	90.52	89.56	90.04	61.94	45.48	52.45
BiLSTM+CRF+self-attention	90.62	88.81	89.71	57.81	47.67	52.25
BiLSTM+CRF+adversarial+self-attention	**91.73**	**89.58**	**90.64**	55.72	**50.68**	**53.08**

Table 5: Comparison between our proposed model and simplified models on SighanNER dataset and original WeiboNER dataset.

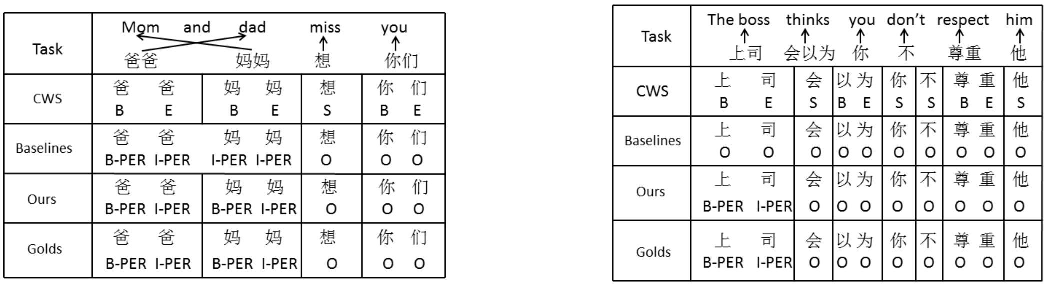

(a) Example for the effectiveness of boundary information.

(b) Example for the effectiveness of self-attention.

Figure 3: The analysis of Chinese NER cases from WeiboNER dataset.

tional features for sequence tagging. Although the model achieves competitive performance, giving the F1 score of 89.21%, it suffers from the error propagation problem.

In the second block, we report the result of our proposed model. Compared with the state-of-the-art model proposed by Luo and Yang (2016), our method improves the F1 score from 89.21% to 90.64% without any additional features, which demonstrates the effectiveness of our proposed model.

4.4 Effectiveness of Adversarial Transfer Learning and Self-Attention

Table 5 provides the experimental results of our proposed model and baseline as well as its simplified models on SighanNER dataset and WeiboNER dataset. The simplified models are described as follows:

- BiLSTM+CRF: The model is used as strong baseline in our work, which is trained using Chinese NER training data.
- BiLSTM+CRF+transfer: We apply transfer learning to BiLSTM+CRF model without adversarial loss and self-attention mechanism.
- BiLSTM+CRF+adversarial: Compared with BiLSTM+CRF+transfer model, the BiLSTM+CRF+adversarial model incorporates adversarial training.
- BiLSTM+CRF+self-attention: The model integrates the self-attention mechanism based on BiLSTM+CRF model.

From the experimental results of Table 5, we have following observations:

- **Effectiveness of transfer learning.** BiLSTM+CRF+transfer improves F1 score from 89.13% to 89.89% as compared with BiLSTM+CRF on SighanNER dataset and achieves 1.08% improvement on WeiboNER dataset, which indicates the word boundary information from CWS is very effective for Chinese NER task.
- **Effectiveness of adversarial training.** By introducing adversarial training, BiLSTM+CRF+adversarial boosts the performance as compared with BiLSTM+CRF+transfer model, showing 0.15% and 0.36% improvement on SighanNER dataset and WeiboNER dataset, respectively. It proves that adversarial training can prevent specific features of CWS task from creeping into shared space.
- **Effectiveness of self-attention mechanism.** When compared with BiLSTM+CRF, the

BiLSTM+CRF+self-attention significantly improves the performance on the two different datasets with the help of information learned from self-attention, which verifies that the self-attention mechanism is effective for Chinese NER task.

We observe that our proposed adversarial transfer learning framework and self-attention lead to noticeable improvements over the baseline, improving F1 score from 51.01% to 53.08% on WeiboNER dataset and giving 1.51% improvement on SighanNER dataset.

4.5 Detailed Analysis

4.5.1 Case Study

Word boundary information from CWS task is very important for Chinese NER task, especially when different entities appear together, . We take a sentence in WeiboNER test set as example for illustrating the effectiveness of our proposed model. As shown in Figure 4(a), when two "person" entities appearing together, our proposed method exploits word segmentation information to determine the boundary between them and then make correct taggings. In Figure 4(b), when labelling the word "上司 (the boss)", the self-attention explicitly learns the dependencies with "尊重 (respect)", therefore, our model enables to correctly classify the word into "person" category. It verifies that the self-attention is very effective for Chinese NER task.

4.5.2 Error Analysis

According to the results of Table 2 and Table 4, our proposed model achieves 4.67% and 1.43% improvement as compared with previous state-of-the-art methods on WeiboNER dataset and SighanNER dataset, respectively. However, the overall performance on WeiboNER dataset is relatively low. Two reasons can be explained for this issue. One reason is that the number of training examples in WeiboNER dataset is very limited as compared with SighanNER dataset. There are only 1.3k examples in WeiboNER training corpora, which is not enough to train deep neural networks. Another reason is that the expression is informal in social media, lowering the performance on WeiboNER dataset. While the greater improvement on WeiboNER dataset proves that our method is helpful to solve the problem.

5 Conclusions

In this paper, we propose a novel adversarial transfer learning framework for Chinese NER task, which can exploit task-shared word boundaries features and prevent the specific information of CWS task. Besides, we introduce self-attention mechanism to capture the dependencies of arbitrary two characters and learn the inner structure information of sentence. Experiments on two different widely used datasets demonstrate that our method significantly and consistently outperforms previous state-of-the-art models.

Acknowledgments

The research work is supported by the Natural Science Foundation of China (No.61533018 and No.61702512), and the independent research project of National Laboratory of Pattern Recognition. This work is also supported in part by Beijing Unisound Information Technology Co., Ltd.

References

Daniel M Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of the fifth conference on Applied natural language processing*, pages 194–201. Association for Computational Linguistics.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 724–731. Association for Computational Linguistics.

Aitao Chen, Fuchun Peng, Roy Shan, and Gordon Sun. 2006. Chinese named entity recognition with conditional probabilistic models. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 173–176.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. *arXiv preprint arXiv:1704.07556*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 167–176.

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

Emily L Denton, Soumith Chintala, Rob Fergus, et al. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Yaroslav Ganin and Victor Lempitsky. 2014. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2411–2420.

Hangfeng He and Xu Sun. 2017a. F-score driven max margin neural network for named entity recognition in chinese social media. *EACL 2017*, page 713.

Hangfeng He and Xu Sun. 2017b. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *AAAI*, pages 3216–3222.

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

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

Hideki Isozaki and Hideto Kazawa. 2002. Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

Wencan Luo and Fan Yang. 2016. An empirical study of automatic chinese word segmentation for spoken language understanding and named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–248.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 149–155.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2017. Deep semantic role labeling with self-attention. *arXiv preprint arXiv:1712.01586*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Chunqi Wang, Wei Chen, and Bo Xu. 2017. Named entity recognition with gated convolutional neural networks. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 110–121. Springer.

YaoSheng Yang, Meishan Zhang, Wenliang Chen, Wei Zhang, Haofen Wang, and Min Zhang. 2018. Adversarial learning for chinese ner from crowd annotations. *arXiv preprint arXiv:1801.05147*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 956–966.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *arXiv preprint arXiv:1701.00188*.

Junsheng Zhou, Liang He, Xinyu Dai, and Jiajun Chen. 2006. Chinese named entity recognition with a multi-phase model. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 213–216.

Junsheng Zhou, Weiguang Qu, and Fen Zhang. 2013. Chinese named entity recognition via joint identification and categorization. *Chinese journal of electronics*, 22(2):225–230.

Using Linguistic Features to Improve the Generalization Capability of Neural Coreference Resolvers

Nafise Sadat Moosavi[1,2*] and **Michael Strube**[1]
[1]Heidelberg Institute for Theoretical Studies gGmbH
[2]Research Training Group AIPHES
moosavi@ukp.informatik.tu-darmstadt.de, michael.strube@h-its.org

Abstract

Coreference resolution is an intermediate step for text understanding. It is used in tasks and domains for which we do not necessarily have coreference annotated corpora. Therefore, generalization is of special importance for coreference resolution. However, while recent coreference resolvers have notable improvements on the CoNLL dataset, they struggle to generalize properly to new domains or datasets. In this paper, we investigate the role of linguistic features in building more generalizable coreference resolvers. We show that generalization improves only slightly by merely using a set of additional linguistic features. However, employing features and subsets of their values that are informative for coreference resolution, considerably improves generalization. Thanks to better generalization, our system achieves state-of-the-art results in out-of-domain evaluations, e.g., on WikiCoref, our system, which is trained on CoNLL, achieves on-par performance with a system designed for this dataset.

1 Introduction

Coreference resolution is the task of recognizing different expressions that refer to the same entity. The referring expressions are called mentions. For instance, the sentence "[Susan]$_1$ sent [her]$_1$ daughter to a boarding school" contains two coreferring mentions. "her" is an anaphor which refers to the antecedent "Susan".

The availability of coreference information benefits various Natural Language Processing (NLP) tasks including automatic summarization, question answering, machine translation and information extraction. Current coreference developments are almost only targeted at improving scores on the CoNLL official test set. However, the superiority of a coreference resolver on the CoNLL evaluation sets does not necessarily indicate that it also performs better on new datasets. For instance, the ranking model of Clark and Manning (2016a), the reinforcement learning model of Clark and Manning (2016b) and the end-to-end model of Lee et al. (2017) are three recent coreference resolvers, among which the model of Lee et al. (2017) performs the best and that of Clark and Manning (2016b) performs the second best on the CoNLL development and test sets. However, if we evaluate these systems on the WikiCoref dataset (Ghaddar and Langlais, 2016a), which is consistent with CoNLL with regard to coreference definition and annotation scheme, the performance ranking would be in a reverse order[1].

In Moosavi and Strube (2017a), we investigate the generalization problem in coreference resolution and show that there is a large overlap between the coreferring mentions in the CoNLL training and evaluation sets. Therefore, higher scores on the CoNLL evaluation sets do not necessarily indicate a better coreference model. They may be due to better memorization of the training data. As a result, despite the remarkable improvements in coreference resolution, the use of coreference resolution in other applications is mainly limited to the use of simple rule-based systems, e.g. Lapata and Barzilay (2005),Yu and Ji (2016), and Elsner and Charniak (2008).

In this paper, we explore the role of linguistic features for improving generalization. The incorporation of linguistic features is considered as a potential solution for building more generalizable NLP systems[2]. While linguistic features[3]

---

[*] This author is currently employed by the Ubiquitous Knowledge Processing (UKP) Lab, Technische Universität Darmstadt, https://www.ukp.tu-darmstadt.de.

[1] The single model of Lee et al. (2017) is used here.

[2] E.g. there is a dedicated workshop for this topic `https://sites.google.com/view/relsnnlp`.

[3] We refer to features that are based on linguistic intu-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 193–203
Brussels, Belgium, October 31 - November 4, 2018.

were shown to be important for coreference resolution, e.g. Uryupina (2007) and Bengtson and Roth (2008), state-of-the-art systems no longer use them and mainly rely on word embeddings and deep neural networks. Since all recent systems are using neural networks, we focus on the effect of linguistic features on a neural coreference resolver.

The contributions of this paper are as follows:

- We show that linguistic features are more beneficial for a neural coreference resolver if we incorporate features and subsets of their values that are informative for discriminating coreference relations. Otherwise, employing linguistic features with all their values only slightly affects the performance and generalization.
- We propose an efficient discriminative pattern mining algorithm, called EPM, for determining (feature, value) pairs that are informative for the given task. We show that while the informativeness of EPM mined patterns is on-par with those of its counterparts, it scales best to large datasets.[4]
- By improving generalization, we achieve state-of-the-art performance on all examined out-of-domain evaluations. Our out-of-domain performance on WikiCoref is on-par with that of Ghaddar and Langlais (2016b)'s coreference resolver, which is a system specifically designed for WikiCoref and uses its domain knowledge.

2 Importance of Features in Coreference

Uryupina (2007)'s thesis is one of the most thorough analyses of linguistically motivated features for coreference resolution. She examines a large set of linguistic features, i.e. string match, syntactic knowledge, semantic compatibility, discourse structure and salience, and investigates their interaction with coreference relations. She shows that even imperfect linguistic features, which are extracted using error-prone preprocessing modules, boost the performance and argues that coreference resolvers could and should benefit from linguistic theories. Her claims are based on analyses on the MUC dataset. Ng and Cardie (2002), Yang et al. (2004), Ponzetto and Strube (2006), Bengtson and Roth (2008), and Recasens and Hovy (2009) also study the importance of features in coreference resolution.

Apart from the mentioned studies, which are mainly about the importance of individual features, studies like Björkelund and Farkas (2012), Fernandes et al. (2012), and Uryupina and Moschitti (2015) generate new features by combining basic features. Björkelund and Farkas (2012) do not use a systematic approach for combining features. Fernandes et al. (2012) use the Entropy guided Feature Induction (EFI) approach (Fernandes and Milidiú, 2012) to automatically generate discriminative feature combinations. The first step is to train a decision tree on a dataset in which each sample consists of features describing a mention pair. The EFI approach traverses the tree from the root in a depth-first order and recursively builds feature combinations. Each pattern that is generated by EFI starts from the root node. As a result, EFI tends to generate long patterns. A decision tree does not represent all patterns of data. Therefore, it is not possible to explore all feature combinations from a decision tree.

Uryupina and Moschitti (2015) propose an alternative approach to EFI. They formulate the problem of generating feature combinations as a pattern mining approach. They use the Jaccard Item Mining (JIM) algorithm[5] (Segond and Borgelt, 2011). They show that the classifier that uses the JIM features significantly outperforms the one that employs the EFI features.

3 Baseline Coreference Resolver

deep-coref (Clark and Manning, 2016a) and e2e-coref (Lee et al., 2017) are among the best performing coreference resolvers from which e2e-coref performs better on the CoNLL test set. deep-coref is a pipelined system, i.e. a mention detection first determines the list of candidate mentions with their corresponding features. It contains various coreference models including the mention-pair, mention-ranking, and entity-based models. The mention-ranking model of deep-coref has three variations: (1) "ranking" uses the slack-rescaled max-margin training objective of Wiseman et al. (2015), (2) "reinforce" is a variation of the "ranking" model in which the hyper-parameters are set in a reinforcement learning framework (Sutton and Barto, 1998), and (3) "top-

---

itions, e.g. string match, or are acquired from linguistic preprocessing modules, e.g. POS tags, as linguistic features.

[4] The EPM code is available at `https://github.com/ns-moosavi/epm`

[5] `http://www.borgelt.net/jim.html`

pairs" is a simple variation of the "ranking" model that uses a probabilistic objective function and is used for pretraining the "ranking" model.

e2e-coref is an end-to-end system that jointly models mention detection and coreference resolution. It considers all possible (start, end) word spans of each sentence as candidate mentions. Apart from a single model, e2e-coref includes an ensemble of five models.

We use deep-coref as the baseline in our experiments. The reason is that some of the examined features require the head of each mention to be known, e.g. head match, while e2e-coref mentions do not have specific heads and heads are automatically determined using an attention mechanism. We also observe that if we limit e2e-coref candidate spans to those that correspond to deep-coref's detected mentions, the performance of e2e-coref drops to a level on-par with deep-coref[6].

4 Examined Features

The examined linguistic features include string match, syntactic, shallow semantic and discourse features. **Mention-based** features include:

- Mention type: proper, nominal or pronominal
- Fine mention type: proper, definite or indefinite nominal, or the citation form of pronouns
- Gender: female, male, neutral, unknown
- Number: singular, plural, unknown
- Animacy: animate, inanimate, unknown
- Named entity type: person, location, organization, date, time, number, etc.
- Dependency relation: enhanced dependency relation (Schuster and Manning, 2016) of the head word to its parent
- POS tags of the first, last, head, two words preceding and following of each mention

Pairwise features include:

- Head match: both mentions have the same head, e.g. "red hat" and "the hat"
- String of one mention is contained in the other, e.g. "Mary's hat" and "Mary"
- Head of one mention is contained in the other, e.g. "Mary's hat" and "hat"
- Acronym, e.g. "Heidelberg Institute for Theoretical Studies" and "HITS"
- Compatible pre-modifiers: the set of pre-modifiers of one mention is contained in that of the other, e.g. "the red hat that she is wearing" and "the red hat"
- Compatible[7] gender, e.g. "Mary" and "women"
- Compatible number, e.g. "Mary" and "John"
- Compatible animacy, e.g. "those hats" and "it"
- Compatible attributes: compatible gender, number and animacy, e.g. "Mary" and "she"
- Closest antecedent that has the same head and compatible premodifiers, e.g. "this new book" and "This book" in "Take a look at this new book. This book is one of the best sellers."
- Closest antecedent that has compatible attributes, e.g. the antecedent "Mary" and the anaphor "she" in the sentence "John saw Mary, and she was in a hurry"
- Closest antecedent that has compatible attributes and is a subject, e.g. the antecedent "Mary" and the anaphor "she" in the sentence "Mary saw John, but she was in a hurry"
- Closest antecedent that has compatible attributes and is an object, e.g. "Mary" and "she" in "John saw Mary, and she was in a hurry"

The last three features are similar to the discourse-level features discussed by Uryupina (2007), which are created by combining *proximity*, *agreement* and *salience* properties. She shows that such features are useful for resolving pronouns. we estimate proximity by considering the distance of two mentions. The salience is also incorporated by discriminating subject or object antecedents. We do not use any gold information. All features are extracted using Stanford CoreNLP (Manning et al., 2014).

5 Impact of Linguistic Features

In this section, we examine the effect of employing all linguistic features described in Section 4 in a neural coreference resolver, i.e. deep-coref. We use *MUC* (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), $CEAF_e$ (Luo, 2005), *LEA* (Moosavi and Strube, 2016), and the *CoNLL* score (Pradhan et al., 2014), i.e. the average F_1 value of *MUC*, B^3, and $CEAF_e$, for evaluations.

The results of employing those features in deep-coref's "ranking" and "top-pairs" models on the

---

[6] The CoNLL score of the e2e-coref single model on the CoNLL development set drops from 67.36 to 65.81, while that of the deep-coref "ranking" model is 66.09.

[7] One value is unknown, or both values are identical.

CoNLL development set are reported in Table 1.

	MUC	B^3	CEAF$_e$	CoNLL	LEA
ranking	74.31	64.23	59.73	66.09	60.47
+linguistic	74.35	63.96	60.19	66.17	60.20
top-pairs	73.95	63.98	59.52	65.82	60.07
+linguistic	74.32	64.45	60.19	66.32	60.62

Table 1: Impact of linguistic features on deep-coref models on the CoNLL development set.

The rows "ranking" and "top-pairs" show the base results of deep-coref's "ranking" and "top-pairs" models, respectively. "+linguistic" rows represents the results for each of the mention-ranking models in which the feature set of Section 4 is employed. The gender, number, animacy and mention type features, which have less than five values, are converted to binary features. Named entity and POS tags, and dependency relations are represented as learned embeddings.

We observe that incorporating all the linguistic features bridges the gap between the performance of "top-pairs" and "ranking". However, it does not improve significantly over "ranking". Henceforth, we use the "top-pairs" model of deep-coref as the baseline model to incorporate linguistic features.

To assess the impact on generalization, we evaluate "top-pairs" and "+linguistic"[8] models that are trained on CoNLL, on WikiCoref (see Table 2). We observe that the impact on generalization is also not notable, i.e. the CoNLL score improves only by 0.5pp over "ranking".

	MUC	B^3	CEAF$_e$	CoNLL	LEA
ranking	63.10	48.43	47.18	52.90	44.40
top-pairs	63.09	48.42	46.05	52.52	44.21
+linguistic	63.99	49.63	46.60	53.40	45.66

Table 2: Out-of-domain evaluation of deep-coref models on the WikiCoref dataset.

Based on an ablation study, while our feature set contains numerous features, the resulting improvements of "linguistic" over "top-pairs" mainly comes from the last four pairwise features in Section 4, which are carefully designed features.

6 Better Exploiting Linguistic Features

As discussed by Moosavi and Strube (2017a), there is a large lexical overlap between the coreferring mentions of the CoNLL training and evaluation sets. As a result, lexical features provide a very strong signal for resolving coreference relations.

For linguistic features to be more effective in current coreference resolvers, which rely heavily on lexical features, they should also provide a strong signal for coreference resolution.

Additional linguistic features are not necessarily all informative for coreference resolution, especially if they are extracted automatically and are noisy. Besides, for features with multiple values, e.g. mention-based features, only a small subset of values may be informative.

To better exploit linguistic features, we only employ (feature, value) pairs[9] that are informative for coreference resolution. Coreference resolution is a complex task in which features have complex interactions (Recasens and Hovy, 2009). As a result, we cannot determine the informativeness of feature-values in isolation.

We use a discriminative pattern mining approach (Cheng et al., 2007, 2008; Batal and Hauskrecht, 2010) that examines all combinations of feature-values, up to a certain length, and determines which feature-values are informative when they are considered in combination.

Due to the large data size (all mention-pairs of the CoNLL training data) and the high dimensionality of feature-values, compared to common evaluation sets of pattern mining methods, the existing discriminative pattern mining approaches were not applicable to our data. In this section, we propose an efficient discriminative pattern mining approach, called Efficient Pattern Miner (EPM), that is scalable to large NLP datasets. The most important properties of EPM are (1) it examines all frequent feature-values combinations, up to the desired length, (2) it is scalable to large datasets, and (3) it is only data dependent and independent of the coreference resolver.

6.1 Notation

We use the following notations and definitions throughout this section:

- $D = \{X_i, c(X_i)\}_{i=1}^{n}$: set of n training samples. X_i is the set of feature-values that describes the ith sample. $c(X_i) \in C$ is the label of X_i, e.g. coreferent and non-coreferent.
- $A = \{a_1, \ldots, a_l\}$: set of all feature-values present in D. Each $a_i \in A$ is called an item, e.g. $a_i =$"anaphor type=proper".

---

[8] i.e. "top-pairs+linguistic"

[9] Henceforth, we refer to them as feature-values.

- p: pattern $p = \{a_{i_1}, \dots, a_{i_k}\}$ is a set of one or more items, e.g. p ={“anaphor type=proper”, “antecedent type=proper”}.
- $support(p, c_i)$: the number of samples that contain pattern p and are labeled with c_i.

6.2 Data Structure

For representing the input samples, we use the Frequent Pattern Tree (FP-Tree) structure that is the data structure of the FP-Growth algorithm (Han et al., 2004), i.e. one of the most common algorithms for frequent pattern mining. FP-Tree provides a structure for representing all existing patterns of data in a compressed form. Using the FP-Tree structure allows an efficient enumeration of all frequent patterns. In the FP-Tree structure, items are arranged in descending order of frequency. Frequency of an item corresponds to $\sum_{c_i \in C} support(a_i, c_i)$. Except for the root, which is a null node, each node n contains an item $a_i \in A$. It also contains the support values of a_i in the subpath of the tree that starts from the root and ends with n, i.e. $support_n(a_i, c_j)$.

The FP-Tree construction method (Han et al., 2004) is as follows: (a) scan D to collect the set of all items, i.e. A. Compute $support(a_i, c_j)$ for each item $a_i \in A$ and label $c_j \in C$. Sort A’s members in descending order according to their frequencies, i.e. $\sum_{c_i \in C} support(a_i, c_i)$. (b) create a null-labeled node as the root, and (c) scan D again. For each $(X_i, c(X_i)) \in D$:

1. Order all items $a_j \in X_i$ according to the order in A.
2. Set the current node (T) to the root.
3. Consider $X_i = [a_k|\bar{X}_i]$, where a_k is the first (ordered) item of x_i, and $\bar{X}_i = X_i - a_k$. If T has a child n that contains a_k then increment $support_n(a_k, c(X_i))$ by one. Otherwise, create a new node n that contains a_k with $support_n(a_k, c(X_i)) = 1$. Add n to the tree as a child of T.
4. If $\bar{X}_i$ is non-empty, set T to n. Assign $X_i = \bar{X}_i$ and go to step 3.

As an example, assume D contains the following two samples:

X_1={ana-type=NAM, ant-type=NAM, head-match=F}, $C(X_1) = 0$

X_2={ana-type=NAM, ant-type=NAM, head-match=T}, $C(X_2) = 1$

Based on these samples A={ana-type=NAM, ant-type=NAM, head-match=F, head-match=T}, $support(a_i, 0)_{a_i \in A}$= {1,1,1,0}, and $support(a_i, 1)_{a_i \in A}$={1,1,0,1}. If we sort A based on a_i’s frequencies ($support(a_i, 0) + support(a_i, 1)$), the ordering of A’s items will remain the same.

The FP-Tree construction steps for the above samples are demonstrated in Figure 1. ana-type, ant-type, and head-match features are abbreviated as ana, ant, and head, respectively.

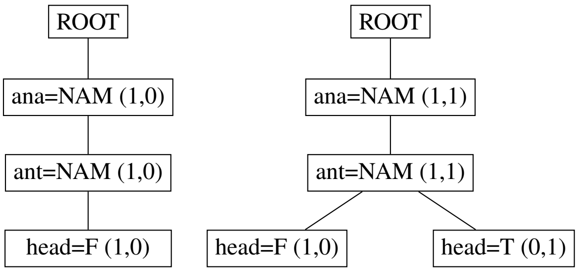

Figure 1: Left to right: (partially) constructed FP-Tree for the example in Section 6.2.

From an initial FP-Tree (T) that represents all existing patterns, one can easily obtain a new FP-Tree in which all patterns include a given pattern p. This can be done by only including sub-paths of T that contain pattern p. The new tree is called conditional FP-Tree of p, T_p. An example of conditional FP-Tree is included in the supplementary materials.

6.3 Informativeness Measures

We use a *discriminative power* and an *information novelty* measure for determining informativeness. We also use a *frequency* measure which determines the required minimum frequency of a pattern in training samples. It helps to avoid overfitting to the properties of the training data.

Discriminative power: We use the G^2 likelihood ratio test (Agresti, 2007) in order to choose patterns whose association with the class variable is statistically significant.[10] The G^2 test is successfully used for text analysis (Dunning, 1993).

Information Novelty: A large number of redundant patterns can be generated by adding irrelevant items to a base pattern that is discriminative itself.

---

[10] A pattern is considered discriminative if the corresponding p-value is less than a fixed threshold (0.01).

We consider the pattern p as novel if (1) p predicts the target class label c significantly better than all of its containing items, and (2) p predicts c significantly better than all of its sub-patterns that satisfy the frequency, discriminative power, and the first information novelty conditions. Similar to Batal and Hauskrecht (2010), we employ a binomial distribution to determine information novelty.

6.4 Mining Algorithm

The EPM algorithm is summarized in Algorithm 1. It takes FP-Tree T, pattern p on which T is conditioned, and set of items ($A_j \subset A$) whose combinations with p will be examined. Initially, p is empty and the FP-Tree is constructed based on all frequent items of data and $A_j = A$. Resulting patterns are collected in P.

For each $a_i \in A_j$, the algorithm builds new pattern q by combining a_i with p. $frequent(q)$ checks whether q meets the frequency condition. If q is frequent, the algorithm continues the search process. Otherwise, it is not possible to build any frequent pattern out of a non-frequent one. *Discriminative power* and the first condition of *information novelty* are then checked for pattern q.

```
Algorithm $EPM$ ($T$, $p$, $A_j$)
    foreach $a_i \in A_j$ do
        $q = p \cup \{a_i\}$
        if $Frequent(q)$ then
            if $Discriminative(q)$ then
                if $Novel(q)$ then
                    $P = P \cup q$
                end
            end
            if $|q| >= \Theta_l$ then
                continue
            end
            construct $T_q$ = $q$'s conditional tree
            $EPM(T_q, q, ancestors(a_i))$
        end
    end
```

Algorithm 1: The EPM algorithm.

We use a threshold (Θ_l) for the maximum length of mined patterns. Θ_l can be set to large values if more complex and specific patterns are desirable.

If $|q|$ is smaller than Θ_l, the conditional FP-Tree T_q is built that represents patterns of T that include the pattern q. The mining algorithm then continues to recursively search for more specific patterns by combining q with the items included in $ancestors(a_i)$, which keeps the list of all ancestors of a_i in the original FP-Tree. EPM examines all frequent patterns of up to length Θ_l.

If we use a statistical test multiple times, the risk of making false discoveries increases (Webb, 2006). To tackle this, we apply the Bonferroni correction for multiple tests in a post-pruning function after the mining process. This function also applies the second information novelty condition on the resulting patterns.

7 Why Use EPM?

In this section, we explain why EPM is a better alternative compared to its counterparts for large NLP datasets. We compare EPM with two efficient discriminative pattern mining algorithms, i.e. Minimal Predictive Patterns (MPP) (Batal and Hauskrecht, 2010) and Direct Discriminative Pattern Mining (DDPMine) (Cheng et al., 2008), on standard machine learning datasets.

MPP selects patterns that are significantly more predictive than all their sub-patterns. It measures significance by the binomial distribution. For each pattern of length l, MPP checks $2^l - 1$ sub-patterns. DDPMine is an iterative approach that selects the most discriminative pattern at each iteration and reduces the search space of the next iteration by removing all samples that include the selected pattern. DDPMine uses the FP-Tree structure.

We show that EPM scales best and compares favorably based on the informativeness of resulting patterns. Due to its efficiency, EPM can handle large datasets similar to ones that are commonly used in various NLP tasks.

7.1 Experimental Setup

We use the same FP-Tree implementation for DDPMine and EPM. In all algorithms, we consider a pattern as frequent if it occurs in 10% of the samples of one of the classes. We use $\Theta_l = 3$ for both MPP and EPM.

We perform 5-times repeated 5-fold cross validation and the results are averaged. In each validation, all experiments are performed on the same split. We use a linear SVM, i.e. LIBLINEAR 2.11 (Fan et al., 2008), as the baseline classifier.

We use several datasets from the UCI machine learning repository (Lichman, 2013) whose characteristics are presented in the first three columns of Table 3, i.e. the number of (1)

	Data characteristics			# Patterns			Micro-F				Macro-F			
Dataset	#Features	#FI	n	DDP	MPP	EPM	Orig	DDP	MPP	EPM	Orig	DDP	MPP	EPM
cmc	(0/2/7)	24	1473	4	99	23	77.5	77.4	76.2	77.3	57.3	57.1	57.7	59.4
nursery	(0/0/8)	27	12690	4	258	198	97.5	98.2	99.9	99.8	49.4	79.4	99.8	98.8
sick	(6/1/22)	36	2800	5	627	89	94.6	94.7	96.1	95.8	62.6	64.8	81.0	75.6
kr-v-k	(0/0/16)	40	28056	7	71	63	99.1	99.1	99.6	99.6	49.8	49.8	87.8	88.4
german	(0/7/13)	51	1000	8	548	97	70.7	70.9	73.1	72.7	49.6	55.2	65.3	64.2
connect-4	(0/0/42)	75	67557	-	-	907	90.5	-	-	90.5	47.5	-	-	56.6
census	(1/12/28)	76	299284	-	-	5618	93.8	-	-	93.8	48.4	-	-	51.6
poker	(0/10/0)	85	1025010	-	-	14216	23.1	-	-	49.6	22.4	-	-	44.5

Table 3: Evaluating the informativeness of DDPMine, MPP and EPM patterns on standard datasets.

(real/integer/nominal) features (#Features), (2) frequent items (#FI), and (3) samples (n). We use one[the minority class]-vs-all technique for datasets with more than two classes.

7.2 How Informative are EPM Patterns?

To evaluate the informativeness of mined patterns, the common practice is to add them as new features to the feature set of the baseline classifier; the more informative the patterns, the greater impact they would have on the overall performance. All patterns are added as binary features, i.e. the feature is true for samples that contain all items of the corresponding pattern.

The effect of the patterns of DDPMine, MPP and EPM on the overall accuracy is presented in Table 3. The columns #Patterns show the number of patterns mined by each of the algorithms. The *Orig* columns show the results of the SVM using the original feature sets. The *DDP*, *MPP*, and *EPM* columns show the results of the SVM on the datasets for which the feature set is extended by the features mined by DDPMine, MPP, and EPM, respectively. The results of the 5-repeated 5-fold cross validation are reported if each single validation takes less than 10 hours.

Based on the results of Table 3 (1) EPM efficiently scales to larger datasets, (2) MPP and EPM patterns considerably improves the performance, and (3) EPM has on-par results with MPP while it mines considerably fewer patterns.

7.3 How Does it Scale?

Figure 2 compares EPM mining time (in seconds) with those of DDPMine and MPP. The parameter in the parentheses is the pattern size threshold, e.g. $\Theta_l = 4$ for EPM(4). The experiments that take more than two days are terminated and are not included. EPM is notably faster in comparison to the other two approaches. It is notable that the examined datasets are considerably smaller than the coreference data, which includes more than 33 million samples and 200 frequent feature-values.

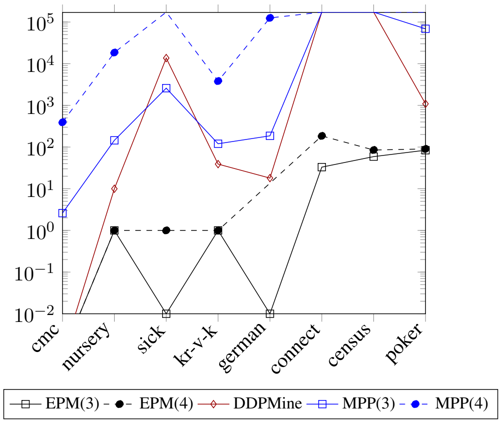

Figure 2: Comparison of mining times (seconds).

8 Impact of Informative Feature-values

8.1 Experimental Setup

For determining informative feature-values, we extract all features for all mention-pairs[11] of the CoNLL training data and then apply EPM on this data. In order to prevent learning annotation errors and specific properties of the training data, we consider a pattern as frequent if it occurs in coreference relations of at least m different coreferring anaphors ($m = 20$). Since the majority of mention-pairs are non-coreferent and we are not interested in patterns for non-coreferring relations, we also consider the coreference probability of each pattern p, i.e. $\frac{|\{X_i|p\in X_i \wedge c(X_i)=coreferent\}|}{|\{X_i|p\in X_i\}|}$, in the post-pruning function. The coreference probability should be higher than a threshold (60% in our experiments), so we only mine patterns that are informative for coreferring mentions.

For the coreference resolution experiments, instead of incorporating informative patterns, we incorporate feature-values that are included in the

---

[11] Each mention is paired with all the preceding mentions.

		MUC			B^3			CEAF$_e$			CoNLL	LEA		
		R	P	F_1	R	P	F_1	R	P	F_1		R	P	F_1
deep-coref	ranking	70.43	79.57	74.72	58.08	69.26	63.18	54.43	64.17	58.90	65.60	54.55	65.68	59.60
	reinforce	69.84	79.79	74.48	57.41	70.96	63.47	55.63	63.83	59.45	65.80	53.78	67.23	59.76
	top-pairs	69.41	79.90	74.29	57.01	70.80	63.16	54.43	63.74	58.72	65.39	53.31	67.09	59.41
	+EPM	71.16	79.35	75.03	59.28	69.70	64.07	56.52	64.02	60.04	66.38	55.63	66.11	60.42
	+JIM	69.89	80.45	74.80	57.08	71.58	63.51	55.36	64.20	59.45	65.93	53.46	67.97	59.85
e2e	single	74.02	77.82	75.88	62.58	67.45	64.92	59.16	62.96	61.00	67.27	58.90	63.79	61.25
	ensemble	73.73	80.95	77.17	61.83	72.10	66.57	60.11	65.62	62.74	68.83	58.48	68.81	63.23

Table 4: Comparisons on the CoNLL test set. The F_1 gains that are statistically significant: (1) "+EPM" compared to "top-pairs", "ranking" and "JIM", (2) "+EPM" compared to "reinforce" based on MUC, B^3 and LEA, (3) "single" compared to "+EPM" based on MUC and B^3, and (4) "ensemble" compared to other systems. Significance is measured based on the approximate randomization test ($p < 0.05$) (Noreen, 1989).

informative patterns mined by EPM. The reason is that deep-coref, or any other recent coreference resolver, uses a deep neural network, which has a fully automated feature generation process. We add these feature-values as binary features.

By setting Θ_l to five,[12] EPM results in 13 pairwise feature-values, 112 POS tags, i.e. 53 POS for anaphors and 59 for antecedents, 25 dependency relations, 26 mention types (mention types or fine mention types), and finally, 14 named entity tags.[13]

Based on the observation in Section 5, we use the top-pairs model of deep-coref as the baseline to employ additional features, i.e. "+EPM" is the top-pairs model in which EPM feature-values are incorporated.

8.2 Impact on In-domain Performance

The performance of the "+EPM" model compared to recent state-of-the-art coreference models on the CoNLL test set is presented in Table 4. The "single" and "ensemble" rows represent the results of the single and ensemble models of e2e-coref.

We also compare EPM with the pattern mining approach used by Uryupina and Moschitti (2015), i.e. Jaccard Item Mining (JIM). For a fair comparison, while Uryupina and Moschitti (2015) used mined patterns for extracting feature templates, we use them for selecting feature-values. We run the JIM algorithm on the same data and with the same setup as that of EPM.[14] This results in nine pairwise features, 260 POS tags, 38 dependency relations, 32 mention types, and 18 named entity tags. The "+JIM" row shows the results of deep-coref top-pairs model in which these feature-values are incorporated. As we see, EPM feature-values result in significantly better performance than those of JIM while the number of EPM feature-values is considerably less than JIM.

	MUC	B^3	CEAF$_e$	CoNLL	LEA
+EPM	74.92	65.03	60.88	66.95	61.34
-pairwise	74.37	64.55	60.46	66.46	60.71
-type	74.71	64.87	61.00	66.86	61.07
-dep	74.57	64.79	60.65	66.67	61.01
-NER	74.61	65.05	60.93	66.86	61.27
-POS	74.74	65.04	60.88	66.89	61.30
+pairwise	74.25	64.33	60.02	66.20	60.57

Table 5: Impact of different EPM feature groups on the CoNLL development set.

Feature Ablation Table 5 shows the effect of each group of EPM feature-values, i.e. pairwise features, mention types, dependency relations, named entity tags and POS tags, on the performance of "+EPM". The performance of "+EPM" from which each of the above feature groups is removed, one feature group at a time, is represented as "-pairwise", "-types", "-dep", "-NER", and "-POS", respectively. The POS and named entity tags have the least and the pairwise features have the most significant effect. Since pairwise features have the most significant effect, we also perform an experiment in which only pairwise features are incorporated in the "top-pairs" model, i.e. "+pairwise". The results of "-pairwise" compared to "+pairwise" show that pairwise feature-values have a significant impact, but only when they are considered in combination with other EPM

---

[12] We observe that using larger Θ_l values will result in many over-specified patterns.

[13] Following the previous studies that show different features are of different importance for various types of mentions, e.g. Denis and Baldridge (2008) and Moosavi and Strube (2017b), we mine a separate set of patterns for each type of anaphor. These resulting feature-values are the union of informative feature-values for all types of anaphora.

[14] We set the minimum frequency, maximum pattern length and $score^+$ threshold parameters of JIM to 20, 5 and 0.6.

		MUC			B^3			CEAF$_e$			CoNLL	LEA		
		R	P	F$_1$	R	P	F$_1$	R	P	F$_1$		R	P	F$_1$
e2edeep-coref	ranking	57.72	69.57	63.10	41.42	58.30	48.43	42.20	53.50	47.18	52.90	37.57	54.27	44.40
	reinforce	62.12	58.98	60.51	46.98	45.79	46.38	44.28	46.35	45.29	50.73	42.28	41.70	41.98
	top-pairs	56.31	71.74	63.09	39.78	61.85	48.42	40.80	52.85	46.05	52.52	35.87	57.58	44.21
	+EPM	58.23	74.05	**65.20**	43.33	63.90	51.64	43.44	56.33	**49.05**	**55.30**	39.70	59.81	**47.72**
	single	60.14	64.46	62.22	45.20	51.75	48.25	38.18	43.50	40.67	50.38	40.70	47.56	43.86
	ensemble	59.58	71.60	65.04	44.64	60.91	51.52	40.38	49.17	44.35	53.63	40.73	56.97	47.50
	G&L	66.06	62.93	64.46	57.73	48.58	**52.76**	46.76	49.54	48.11	55.11	-	-	-

Table 6: Out-of-domain evaluation on the WikiCoref dataset. The highest F_1 scores are boldfaced.

feature-values.

		in-domain		out-of-domain	
		CoNLL	LEA	CoNLL	LEA
		pt (Bible)			
deep-coref	ranking	75.61	71.00	66.06	57.58
	+EPM	76.08	71.13	**68.14**	**60.74**
e2e-coref	single	77.80	73.73	65.22	58.26
	ensemble	**78.88**	**74.88**	65.45	59.71
		wb (weblog)			
deep-coref	ranking	61.46	53.75	57.17	48.74
	+EPM	61.97	53.93	**61.52**	**53.78**
e2e-coref	single	62.02	53.09	60.69	52.69
	ensemble	**64.76**	**57.54**	60.99	52.99

Table 7: In-domain and out-of-domain evaluations for the pt and wb genres of the CoNLL test set. The highest scores are boldfaced.

8.3 Impact on Generalization

We use the same setup as that of Moosavi and Strube (2017a) for evaluating generalization including (1) training on the CoNLL data and testing on WikiCoref[15] and (2) excluding a genre of the CoNLL data from training and development sets and testing on the excluded genre. Similar to Moosavi and Strube (2017a), we use the *pt* and *wb* genres for the latter evaluation setup.

The results of the first evaluation setup are shown in Table 6. The best performance on WikiCoref is achieved by Ghaddar and Langlais (2016a) ("G&L" in Table 6) who introduced WikiCoref and design a domain-specific coreference resolver that makes use of the Wikipedia markups of a document as well as links to Freebase, which are annotated in WikiCoref.

Incorporating EPM feature-values improves the performance by about three points. While "+EPM" does not use the WikiCoref data during training, and unlike "G&L", it does not employ any domain-specific features, it achieves on-par performance with that of "G&L". This indeed shows the effectiveness of informative feature-values in improving generalization.

The second set of generalization experiments is reported in Table 7. "in-domain" columns show the results when the evaluation genres were included in training and development sets while the "out-of-domain" columns show the results when the evaluation genres were excluded. As we can see, "+EPM" generalizes best, and in out-of-domain evaluations, it considerably outperforms the ensemble model of e2e-coref, which has the best performance on the CoNLL test set.

9 Conclusions

In this paper, we show that employing linguistic features in a neural coreference resolver significantly improves generalization. However, the incorporated features should be informative enough to be taken into account in the presence of lexical features, which are very strong features in the CoNLL dataset. We propose an efficient algorithm to determine informative feature-values in large datasets. As a result of a better generalization, we achieve state-of-the-art results in all examined out-of-domain evaluations.

Acknowledgments

The authors would like to thank Mark-Christoph Müller, Benjamin Heinzerling, Alex Judea, Steffen Eger and the anonymous reviewers for their helpful comments and feedbacks. This work has been supported by the Klaus Tschira Foundation, Heidelberg, Germany and the German Research Foundation (DFG) as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

---

[15] WikiCoref only contains 30 documents, which is not enough for training neural coreference resolvers.

References

Alan Agresti. 2007. *An Introduction to Categorical Data Analysis.* John Wiley & Sons.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation,* Granada, Spain, 28–30 May 1998, pages 563–566.

Iyad Batal and Milos Hauskrecht. 2010. Constructing classification features using minimal predictive patterns. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 869–878.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 294–303.

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 49–55.

Hong Cheng, Xifeng Yan, Jiawei Han, and Chih-Wei Hsu. 2007. Discriminative frequent pattern analysis for effective classification. In *Proceedings of the IEEE 23rd International Conference on Data Engineering (ICDE 2007)*, pages 716–725.

Hong Cheng, Xifeng Yan, Jiawei Han, and Philip S Yu. 2008. Direct discriminative pattern mining for effective classification. In *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE 2008)*, pages 169–178.

Kevin Clark and Christopher D. Manning. 2016a. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016.

Kevin Clark and Christopher D. Manning. 2016b. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* Austin, Tex., 1–5 November 2016, pages 2256–2262.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 660–669.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Micha Elsner and Eugene Charniak. 2008. Coreference-inspired coherence modeling. In *Proceedings ACL-HLT 2008 Conference Short Papers,* Columbus, Ohio, 15–20 June 2008, pages 41–44.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Eraldo R Fernandes and Ruy L Milidiú. 2012. Entropy-guided feature generation for structured learning of Portuguese dependency parsing. In *Proceedings of the International Conference on Computational Processing of the Portuguese Language*, pages 146–156. Springer.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 41–48.

Abbas Ghaddar and Philippe Langlais. 2016a. Coreference in Wikipedia: Main concept resolution. In *Proceedings of the 20th Conference on Computational Natural Language Learning,* Berlin, Germany, 7–11 August 2016, pages 229–238.

Abbas Ghaddar and Philippe Langlais. 2016b. WikiCoref: An English coreference-annotated corpus of Wikipedia articles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation,* Portorož, Slovenia, 23–28 May 2016.

Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. 2004. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1-5):53–87.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence,* Edinburgh, Scotland, 30 July – 5 August 2005, pages 1085–1090.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark.

M. Lichman. 2013. UCI machine learning repository.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pages 25–32.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Nafise Sadat Moosavi and Michael Strube. 2016. Which coreference evaluation metric do you trust? A proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016, pages 632–642.

Nafise Sadat Moosavi and Michael Strube. 2017a. Lexical features in coreference resolution: To be used with caution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Vancouver, B.C., Canada, 30 July –4 August 2017.

Nafise Sadat Moosavi and Michael Strube. 2017b. Use generalized representations, but do not forget surface features. In *Proceedings of the 2nd Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2017)*, pages 1–7, Valencia, Spain.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics,* Philadelphia, Penn., 7–12 July 2002, pages 104–111.

Eric W. Noreen. 1989. *Computer Intensive Methods for Hypothesis Testing: An Introduction*. Wiley, New York, N.Y.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics,* New York, N.Y., 4–9 June 2006, pages 192–199.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Baltimore, Md., 22–27 June 2014, pages 30–35.

Marta Recasens and Eduard Hovy. 2009. A deeper look into features for coreference resolution. In *Proceedings of the 7th Discourse Anaphora and Anaphor Resolution*, pages 29–42.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France.

Marc Segond and Christian Borgelt. 2011. Item set mining based on cover similarity. *Advances in Knowledge Discovery and Data Mining*, pages 493–505.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Olga Uryupina. 2007. *Knowledge acquisition for coreference resolution*. Ph.D. thesis, Saarland University.

Olga Uryupina and Alessandro Moschitti. 2015. A state-of-the-art mention-pair model for coreference resolution. In *Proceedings of STARSEM 2015: The Fourth Joint Conference on Lexical and Computational Semantics,* Denver, Col., 4–5 June 2015, pages 289–298.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, Cal. Morgan Kaufmann.

Geoffrey I. Webb. 2006. Discovering significant patterns. *Machine Learning*, 68(1):1–39.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Beijing, China, 26–31 July 2015, pages 1416–1426.

Xiaofeng Yang, Jian Su, Guodung Zhou, and Chew Lim Tan. 2004. Improving pronoun resolution by incorporating coreferential information of candidates. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, 21–26 July 2004, pages 128–135.

Dian Yu and Heng Ji. 2016. Unsupervised person slot filling based on graph mining. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016, pages 44–53.

Neural Segmental Hypergraphs for Overlapping Mention Recognition

Bailin Wang
University of Massachusetts
Amherst
`bailinwang@cs.umass.edu`

Wei Lu
Singapore University of Technology
and Design
`luwei@sutd.edu.sg`

Abstract

In this work, we propose a novel *segmental hypergraph* representation to model overlapping entity mentions that are prevalent in many practical datasets. We show that our model built on top of such a new representation is able to capture features and interactions that cannot be captured by previous models while maintaining a low time complexity for inference. We also present a theoretical analysis to formally assess how our representation is better than alternative representations reported in the literature in terms of representational power. Coupled with neural networks for feature learning, our model achieves the state-of-the-art performance in three benchmark datasets annotated with overlapping mentions.[1]

1 Introduction

One of the most crucial steps towards building a natural language understanding system is the identification of basic semantic chunks in text. Such a task is typically characterized by the named entity recognition task (Grishman, 1997; Tjong Kim Sang and De Meulder, 2003), or the more general mention recognition task, where mentions are defined as references to entities that could be named, nominal or pronominal (Florian et al., 2004). The extracted mentions can be used in various downstream tasks for performing further semantic related tasks, including question answering (Abney et al., 2000), relation extraction (Mintz et al., 2009; Liu et al., 2017), event extraction (Riedel and McCallum, 2011; Li et al., 2013), and coreference resolution (Soon et al., 2001; Ng and Cardie, 2002; Chang et al., 2013).

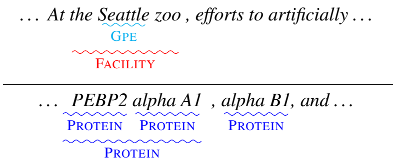

Figure 1: Examples of overlapping mentions.

One popular approach to the task of mention extraction is to regard it as a sequence labeling problem, with the underlying primary assumption being that the mentions are non-overlapping spans in the text. However, as highlighted in several prior research efforts (Alex et al., 2007; Finkel and Manning, 2009; Lu and Roth, 2015), mentions may overlap with one another in practice. Thus, models based on such a simplified assumption may result in sub-optimal performance for a down-stream task when they are deployed in practice. For example, consider a phrase "*At the Seattle zoo, . . .*" shown in Figure 1, the relation LOCATEDIN between the mentions "*the Seattle zoo*" (of type FACILITY) and "*Seattle*" (of type GPE: Geo-political entities) will not be extracted unless both of these two overlapping mentions could be extracted. Similarly, there are 4 mentions of the same type (PROTEIN) in the text span "*. . . PEBP2 alpha A1, alpha B1 . . .*" taken from the biomedical domain. A downstream question answering system may fail to return the correct answer as desired, if the mention extraction system it relies on is unable to extract all these valid mentions.

Various approaches to extracting overlapping mentions have been proposed in the past decade. The cascaded approach (Alex et al., 2007) builds a pipeline of sequence labeling models using conditional random fields (CRF) (Lafferty et al., 2001). However, the model is unable to handle overlapping mentions of the same type. Finkel and Manning (2009) presented a parsing based approach to nested mention extraction. Due to the chart-based parsing algorithm involved, the model has a cubic time complexity in the number of words in the sen-

---

[1] We make our system and code available at: `http://statnlp.org/research/ie`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 204–214
Brussels, Belgium, October 31 - November 4, 2018.

tence. A recent approach by Lu and Roth (2015) introduced a hypergraph representation for capturing overlapping mentions, which was shown fast and effective. The work was improved by Muis and Lu (2017), who proposed a sequence labeling approach that assigns tags to gaps between words. However, both approaches suffer from the *structural ambiguity* issue during inference, as we will further discuss in this paper.

We summarize our contributions as:

1. We propose a novel *segmental hypergraph* representation that is capable of modeling arbitrary combinations of (potentially overlapping) mentions in a given sentence. The model has a $\mathcal{O}(cmn)$ time complexity (m is the number of mention types, n is the number of words in a sentence, and c is the maximal number of words for each mention), and is able to capture features that cannot be captured by existing approaches.
2. Theoretically, we show that our approach based on such a new representation does not have the limitations associated with some recently proposed state-of-the-art approaches for overlapping mention extraction.
3. We show through extensive experiments on standard data that by exploiting both word-level and span-level features learned from neural networks, our model is able to achieve the state-of-the-art performance for recognizing overlapping mentions.

Our model is also general and robust. Further experiments show that our model yields competitive results when evaluated on data that does not have overlapping mentions annotated when comparing against other recently proposed state-of-the-art neural models that are capable of extracting non-overlapping mentions only.

2 Related Work

Overlapping Mention Recognition

One of the earliest research efforts on handling overlapping mentions is a rule-based approach (Zhang et al., 2004; Zhou et al., 2004; Zhou, 2006) that is evaluated on the GENIA dataset (Kim et al., 2003). The authors first detected the innermost mentions and then relied on rule-based post-processing methods to identify overlapping mentions. McDonald et al. (2005) presented a multilabel classification algorithm to model overlapping segments in a sentence systematically.

Alex et al. (2007) proposed several ways to combine multiple conditional random fields (CRF) (Lafferty et al., 2001) for such tasks. Their best results were obtained by cascading several CRF models in a specific order while each model is responsible for detecting mentions of a particular type. Outputs of one model can also serve as features to the next model. However, such an approach cannot model overlapping mentions of the same type, which frequently appear in practice.

Finkel and Manning (2009) approached this task from a parsing perspective by constructing a constituency tree, mapping each mention to a node in the tree. This approach assumes one mention is *contained by* the other when they overlap. While such an assumption largely holds in practice, it comes with a cost – the chart-based parser suffers from its cubic time complexity, making it not scalable to large datasets involving long sentences. Based on the same idea, Wang et al. (2018) proposed a scalable transition-based approach to construct a constituency forest (a collection of constituency trees).

Instead of relying on structured models, Xu et al. (2017) proposed a local classifier for each possible span. However, this local approach is unable to capture the interactions between spans. Similar to (Alex et al., 2007), Ju et al. (2018) dynamically stacked multiple flat layers which recognize mentions sequentially from innermost mentions to outermost mentions.

Our work is inspired by the model of Lu and Roth (2015), who introduced a *mention hypergraph* representation for capturing overlapping mentions. Their model was shown fast and effective, and was improved by the mention separator model (Muis and Lu, 2017). However, we note that (as also highlighted in their papers) both models suffer from the *structural ambiguity* issue during inference, which we will discuss later. Our new representation does not have this limitation.[2] Recently, Katiyar and Cardie (2018) also proposed a hypergraph-based representation based on the BILOU tagging scheme. Their model is trained greedily using neural networks by viewing the hypergraph construction procedure as a multi-label assignment process.

Neural Models for Mention Recognition

Recently, neural network based approaches to entity or mention recognition have received signifi-

---

[2] A model comparison can be found later in Table 1.

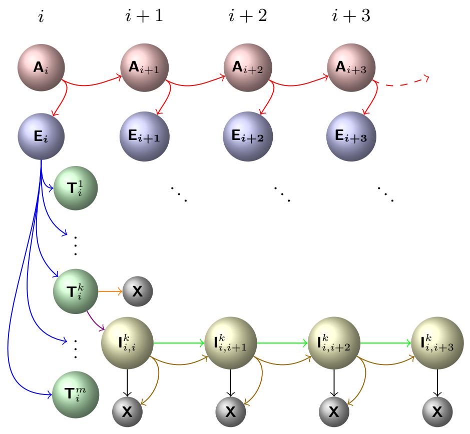

Figure 2: An example of partial segmental hypergraph (hyperedges of different types in different colors).

cant attention. They have been proven effective, even in the absence of handcrafted features. Collobert et al. (2011) used convolutional neural networks (CNN) over word sequences, paired with a CRF output layer. Huang et al. (2015) replaced the CNN with a bidirectional long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997). Strubell et al. (2017) proposed an iterated dilated CNN to improve computational efficiency. Beyond word-level compositions, several methods incorporated character-level compositions with character embeddings, either through CNN (Chiu and Nichols, 2016; Ma and Hovy, 2016) or LSTM (Lample et al., 2016).

3 Segmental Hypergraph

A segmental hypergraph is a representation that aims at representing all possible combinations of (potentially overlapping) mentions in a given sentence. It belongs to a class of directed hypergraphs (Gallo et al., 1993), where each hyperedge e consists of a single designated parent node (*head* of e) and an ordered list of child nodes (*tail* of e). Specifically, our segmental hypergraph consists of the following 5 types of nodes:

- $\mathbf{A}_i$ encodes all such mentions that start with the i-th or a later word
- $\mathbf{E}_i$ encodes all mentions that start exactly with the i-th word
- $\mathbf{T}_i^k$ represents all mentions of type k starting with the i-th word
- $\mathbf{I}_{i,j}^k$ represents all mentions of type k that contain the j-th word and start with the i-th word
- $\mathbf{X}$ marks the end of a mention.

Hyperedges connecting these nodes are designed to indicate how the semantics of a parent node can be re-expressed in terms of its child nodes. Figure 2 gives a partial segmental hypergraph representing all combinations of mentions within the span $[i, i+3]$ consisting of 4 words. There are 4 types of hyperedges:

1. A hyperedge $\{\mathbf{A}_i \rightarrow (\mathbf{A}_{i+1}, \mathbf{E}_i)\}$ from $\mathbf{A}_i$ to its children implies the fact that $\mathbf{A}_i$ consists of those mentions that either "start exactly with the i-th word" ($\mathbf{E}_i$), or "start with a word that appears strictly after the i-th word" ($\mathbf{A}_{i+1}$).
2. A hyperedge $\{\mathbf{E}_i \rightarrow (\mathbf{T}_i^1, \ldots, \mathbf{T}_i^m)\}$ from $\mathbf{E}_i$ to its children implies that we should consider all possible types for the mentions (possibly of length 0) that start with the i-th word.
3. Two hyperedges $\{\mathbf{T}_i^k \rightarrow \mathbf{I}_{i,i}^k\}$ and $\{\mathbf{T}_i^k \rightarrow \mathbf{X}\}$ from $\mathbf{T}_i^k$ indicate that either there exists at least one mention starting with the i-th word (the former hyperedge), or there does not exist any such mention (the latter hyperedge).
4. Three hyperedges $\{\mathbf{I}_{i,j}^k \rightarrow \mathbf{I}_{i,j+1}^k\}$, $\{\mathbf{I}_{i,j}^k \rightarrow \mathbf{X}\}$, and $\{\mathbf{I}_{i,j}^k \rightarrow (\mathbf{I}_{i,j+1}^k, \mathbf{X})\}$ from $\mathbf{I}_{i,j}^k$ indicate the following three cases respectively: 1) both the j-th and $(j+1)$-th words belong to at least one mention that starts with the i-th word, 2) there exists one mention that starts with the i-th word and ends with the j-th word, and 3) both cases are valid.

Essentially, the complete hypergraph compactly encodes the whole search space of all possible mentions that can ever appear within a sentence, where such mentions may or may not overlap with one another. When we traverse the complete segmental hypergraph by following the directions as specified by the hyperedges, selecting only one outgoing hyperedge at a time at each node, we arrive at a *hyperpath*[3] – a rooted, directed substructure contained by the original hypergraph.

Figure 3 shows an example. Here, "*Israeli UN Ambassador*" of type PERSON is captured by the following sequence of nodes (along a hyperpath): "$\mathbf{A}_1$, $\mathbf{E}_1$, $\mathbf{T}_1^2$, $\mathbf{I}_{1,1}^2$, $\mathbf{I}_{1,2}^2$, $\mathbf{I}_{1,3}^2$, $\mathbf{X}$", while "*Israeli UN Ambassador Danny*" of type PERSON corresponds to the following node sequence: "$\mathbf{A}_1$, $\mathbf{E}_1$, $\mathbf{T}_1^2$, $\mathbf{I}_{1,1}^2$, $\mathbf{I}_{1,2}^2$, $\mathbf{I}_{1,3}^2$, $\mathbf{I}_{1,4}^2$, $\mathbf{X}$". Similarly, the following sequence "$\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{E}_2$, $\mathbf{T}_2^1$, $\mathbf{I}_{2,2}^1$, $\mathbf{X}$" represents the mention "*UN*" of type ORGANIZATION. As we can see, such node sequences together form a single hyperpath that encodes this specific combination of mentions that overlap with one another.

---

[3] Each hyperpath is a *hypertree* (Brandstädt et al., 1998).

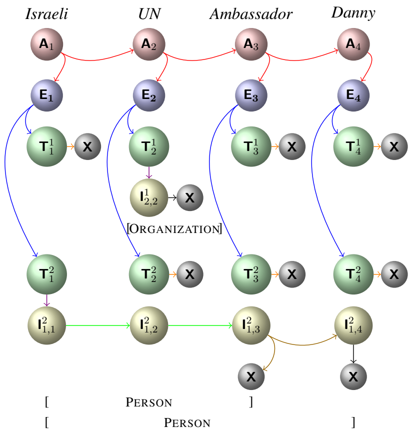

Figure 3: A specific hyperpath for encoding three mentions. For brevity, we only show two types.

More details on segmental hypergraph and hyperpaths are in the supplementary material.

Theoretical Analysis

Our proposed segmental hypergraph representation has the following theoretical property:

Theorem 3.1. (Structural Ambiguity Free) *For any sentence and its segmental hypergraph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*, let* $\mathcal{S}$ *be the set of all possible mention combinations for the given sentence, and* $\mathcal{P}$ *be the set of all hyperpaths contained by* $\mathcal{G}$*, there is a one-to-one correspondence between elements in* $\mathcal{P}$ *and* $\mathcal{S}$*.*

Due to space, we provide a proof sketch and include more details in the supplementary material. **Proof Sketch** We note that each hyperpath is uniquely characterized by its collection of hyperedges that involve **X** nodes. These hyperedges uniquely determine the collection of mentions. Conversely, a collection of mentions can be uniquely characterized by a collection of such hyperedges, which yields a unique hyperpath. ☐

Note that such a theorem states that our novel representation has no *structural ambiguity*, a nice property that both mention hypergraph model of (Lu and Roth, 2015) and mention separator model of (Muis and Lu, 2017) do not hold. As the authors have mentioned in their papers, for a given sub-structure in their model, there exist multiple ways of interpreting the combination of mentions. Specifically, in both representations, the decisions on where the *beginning* and the *end* of a mention are made *locally*. Such a design will lead to the structural ambiguity as there will be multiple interpretations to the mentions given a particular collection of positions marked as beginning and end of mentions. To illustrate, consider a phrase with 4 words "*A B C D*" where there are only two overlapping mentions "*B C*" and "*A B C D*". In both of the previous approaches, their models would make local predictions and assign both "*A*" and "*B*" as left boundaries, and both "*C*" and "*D*" as right boundaries. However, based on such local predictions one could also interpret "*A B C*" as a mention – this is where the ambiguity arises. In contrast, our model enjoys the structural ambiguity free property as it uses our newly defined **I** nodes (together with **X** nodes) to jointly capture the complete boundary information of mentions. Table 1 shows a full comparison. [4]

4 Learning

We adopt a log-linear approach to model the conditional probability of each hyperpath as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp f(\boldsymbol{x}, \boldsymbol{y})}{\sum_{\boldsymbol{y}'} \exp f(\boldsymbol{x}, \boldsymbol{y}')} \tag{1}$$

where $f(\boldsymbol{x}, \boldsymbol{y})$ is the score function for any pair of input sentence $\boldsymbol{x}$ and output mention combination $\boldsymbol{y}$, which corresponds to a unique hyperpath $\mathcal{G}_{\boldsymbol{y}}$. Our objective is to minimize the negative log-likelihood of all instances in the training set $\mathcal{D}$:

$$-\sum_{(\boldsymbol{x}, \boldsymbol{y}^*) \in \mathcal{D}} \log p(\boldsymbol{y}^*|\boldsymbol{x}) \tag{2}$$

We define features over each hyperedge within the hyperpath $\mathcal{G}_{\boldsymbol{y}}$. The score function can be decomposed into the following form:

$$f(\boldsymbol{x}, \boldsymbol{y}) = \sum_{e \in \mathcal{G}_{\boldsymbol{y}}} \psi(e, \boldsymbol{x}) \tag{3}$$

where $e \in \mathcal{G}_{\boldsymbol{y}}$ denotes a hyperedge that appears within the hyperpath $\mathcal{G}_{\boldsymbol{y}}$, and $\psi(e, \boldsymbol{x})$ is a score defined over e when the input sentence is $\boldsymbol{x}$.

Apart from *word-level* features, the segmental hypergraph also allows *span-level* features to be defined. The node $\mathbf{I}^k_{i,j}$ corresponds to a particular span $[i, j]$ over which we can extract our local features. The hyperedge between **I** nodes can capture the interactions between partial mentions and hyperedge between $\mathbf{I}^k_{i,j}$ and **X** precisely represents the mention $[i, j]$ with type k. We note that such features and interactions cannot be captured by the models of (Lu and Roth, 2015) and (Muis and Lu, 2017). Such a unique property makes our segmental hypergraph model more expressive than theirs.

---

[4] The mention hypergraph (Lu and Roth, 2015) also suffers from the *spurious structures* issue, while we do not. We refer the readers to (Muis and Lu, 2017) for details.

	Spurious Structures	Structural Ambiguity	Only Nested Mentions	Pipeline Approach	Different Types Only	Time Complexity
Alex et al. (2007)	NO	NO	NO	YES	YES	$\mathcal{O}(mn)$
Finkel & Manning (2009)	NO	NO	YES	NO	NO	$\mathcal{O}(\|G\|n^3)$
Lu & Roth (2015)	YES	YES	NO	NO	NO	$\mathcal{O}(mn)$
Muis & Lu (2017)	NO	YES	NO	NO	NO	$\mathcal{O}(mn)$
Wang et al. (2018)	NO	NO	YES	NO	NO	$\mathcal{O}(mn)$
This work	**NO**	**NO**	**NO**	**NO**	**NO**	$\mathcal{O}(cmn)$

Table 1: Model comparison. $|G|$ is the number of rules in grammar G.

4.1 Softmax-Margin Training

Inspired by (Mohit et al., 2012), we consider the softmax-margin (Gimpel and Smith, 2010) in our model. The function $\psi(e, \boldsymbol{x})$ is defined as follows:

$$\psi(e, \boldsymbol{x}) = \phi(e, \boldsymbol{x}) + \Delta(e, \mathcal{G}_{\boldsymbol{y}^*}) \tag{4}$$

where $\phi(e, \boldsymbol{x})$ is a feature function, and $\Delta(e, \mathcal{G}_{\boldsymbol{y}^*})$ is the cost function that defines the margin:

$$\Delta(e, \mathcal{G}_{\boldsymbol{y}^*}) = \begin{cases} \beta & \mathrm{TX}[e] \wedge e \notin \mathcal{G}_{\boldsymbol{y}^*} \\ 1 & \mathrm{TI}[e] \wedge e \notin \mathcal{G}_{\boldsymbol{y}^*} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Here, $\boldsymbol{y}^*$ is the gold mention combination, and $\mathrm{TX}[e]$ and $\mathrm{TI}[e]$ are indicator functions that return true if e is between T and X and between T and I respectively, and false otherwise. We set $\beta \geq 1$ such that the cost function will assign more penalty to false negatives than to false positives.

4.2 Feature Representation

We use two bidirectional LSTMs to learn word-level and span-level feature representations that can be used in our approach, resulting in our *neural segmental hypergraph* model. We first map the i-th word in a sentence to its pre-trained word embedding $\mathbf{e}_i$, and its POS tag to its embedding $\mathbf{p}_i$ if it exists. The final representation for i-th word is the concatenation of them: $\mathbf{v}_i = [\mathbf{e}_i, \mathbf{p}_i]$. Next, we use the a bidirectional LSTM to capture context-specific information for each word, resulting in the word-level features:

$$\mathbf{h}^w_i = [\mathrm{biLSTM}_1(\mathbf{v}_0, ..., \mathbf{v}_n)]_i \tag{6}$$

Such representations are then used as inputs to a second LSTM to generate span-level features: Inspired by (Kong et al., 2016), we compute all possible span embeddings efficiently with time complexity $\mathcal{O}(cn)$ using dynamic programming, with n being the number of words in the input $\boldsymbol{x}$ and c being the maximal length of a mention.

$$\mathbf{h}^s_{i:j} = \mathrm{biLSTM}_2(\mathbf{h}^w_i, ..., \mathbf{h}^w_j) \tag{7}$$

Recall that there are 4 types of hyperedges in our hypergraph, over which we can define the score functions. Since every valid mention hyperpath contains the first and second kind of hyperedges, defining scores over such hyperedges are unnecessary as their scores would serve as a constant factor that can be eliminated in the overall loss function of the log-linear model. Thus we only need to define the score functions on the latter two types of hyperedges. For hyperedges that only involve two nodes, we use a linear layer to compute their scores:

$$\phi(\{\mathsf{T}^k_i \rightarrow \mathsf{X}\}, \boldsymbol{x}) = \mathbf{W}^{(k)}_{\mathrm{TX}} \cdot \mathbf{h}^w_i \tag{8}$$

$$\phi(\{\mathsf{T}^k_i \rightarrow \mathsf{I}^k_{i,j}\}, \boldsymbol{x}) = \mathbf{W}^{(k)}_{\mathrm{TI}} \cdot \mathbf{h}^w_i \tag{9}$$

$$\phi(\{\mathsf{I}^k_{i,j} \rightarrow \mathsf{I}^k_{i,j+1}\}, \boldsymbol{x}) = \mathbf{W}^{(k)}_{\mathrm{II}} \cdot [\mathbf{h}^s_{i:j}, \mathbf{h}^s_{i:j+1}] \tag{10}$$

$$\phi(\{\mathsf{I}^k_{i,j} \rightarrow \mathsf{X}\}, \boldsymbol{x}) = \mathbf{W}^{(k)}_{\mathrm{IX}} \cdot \mathbf{h}^s_{i:j} \tag{11}$$

where matrices $\mathbf{W}_{\mathrm{TX}}, \mathbf{W}_{\mathrm{TI}} \in \mathbb{R}^{d_1 \times m}$, $\mathbf{W}_{\mathrm{II}} \in \mathbb{R}^{2d_2 \times m}$, $\mathbf{W}_{\mathrm{IX}} \in \mathbb{R}^{d_2 \times m}$, with superscript (k) referring to the k-th column of the matrix, d_1 is the dimension of $\mathbf{h}^w$, d_2 is the dimension of $\mathbf{h}^s$, and m is the number of mention types.

For the hyperedges that involve more than two nodes, the score is computed as follows:

$$\begin{aligned} &\phi(\{\mathsf{I}^k_{i,j} \rightarrow (\mathsf{X}, \mathsf{I}^k_{i,j+1})\}, \boldsymbol{x}) \\ &= \mathbf{W}'^{(k)}_{\mathrm{II}} \cdot [\mathbf{h}^s_{i:j}, \mathbf{h}^s_{i:j+1}] + \mathbf{W}'^{(k)}_{\mathrm{IX}} \cdot \mathbf{h}^s_{i:j} \end{aligned} \tag{12}$$

where $\mathbf{W}'_{\mathrm{II}} \in \mathbb{R}^{2d_2 \times m}$, $\mathbf{W}'_{\mathrm{IX}} \in \mathbb{R}^{d_2 \times m}$. Note that in this work, we set $\mathbf{W}'_{\mathrm{II}} = \mathbf{W}_{\mathrm{II}}$ and $\mathbf{W}'_{\mathrm{IX}} = \mathbf{W}_{\mathrm{IX}}$ to reduce the number of free parameters.

Learning uses stochastic gradient descent with the update rule of Adam (Kingma and Ba, 2014) and a gradient clipping of 3.0. Dropout (Srivastava et al., 2014) for input vectors $\mathbf{v}$ and ℓ_2 regularization are used to reduce overfitting; both are tuned during the development process.

4.3 Character-level Representation

To make fair comparisons with recent models (Ju et al., 2018; Wang et al., 2018) that additionally incorporate character-level components in capturing orthographic and morphological features of words, we follow Lample et al. (2016) to use a bidirectional LSTM that takes the character embeddings as input. Specifically, the character-level representation $\mathbf{ch}_i$ for each word is obtained by concatenating the last hidden vectors of the forward and backward LSTMs. When this component is activated, the representation of each word is changed to: $\mathbf{v}_i = [\mathbf{e}_i, \mathbf{p}_i, \mathbf{ch}_i]$.

5 Inference

Inference can be done efficiently using a generalized inside-outside style message-passing algorithm (Baker, 1979). The partition function of (1) can be computed using the inside algorithm applied to the complete hypergraph $\mathcal{G}$, where we traverse from leaf nodes $\mathsf{\mathbf{X}}$ to the root node $\mathsf{\mathbf{A}}_1$, passing messages to a parent node $\mathsf{\mathbf{p}}$ from its child nodes:

$$\mu[\mathsf{\mathbf{p}}] \leftarrow \log\Big(\sum_{e:h(e)\equiv\mathsf{\mathbf{p}}} \exp\big(\psi(e,\boldsymbol{x}) + \sum_{\mathsf{\mathbf{c}}\in\mathcal{T}(e)} \mu[\mathsf{\mathbf{c}}]\big)\Big) \tag{13}$$

where $h(e)$ is the head of the hyperedge e, and $\mathcal{T}(e)$ is the collection of nodes that form the tail of e – they are the child nodes of $h(e)$ given e. The message passing step for the outside algorithm can be defined analogously. It can be verified that such a message passing algorithm, that is analogous to the sum-product belief propagation algorithm (Kschischang et al., 2001) used in standard graphical models, will converge after one forward and one backward pass.

For decoding, we perform the standard MAP inference on top of the complete hypergraph to find the most probable hyperpath. The resulting procedure is similar to the max-product message passing algorithm, where we consider only the feature function ϕ for constructing the messages:

$$\mu[\mathsf{\mathbf{p}}] \leftarrow \max_{e:h(e)\equiv\mathsf{\mathbf{p}}} \Big(\phi(e,\boldsymbol{x}) + \sum_{\mathsf{\mathbf{c}}\in\mathcal{T}(e)} \mu[\mathsf{\mathbf{c}}]\Big) \tag{14}$$

During inference, each node corresponds to a sum/max computation. Since one node is incident to 3 hyperedges maximally, the time complexity of inference algorithm can be implied by the number of nodes in the graph, which is $\mathcal{O}(cmn)$, where c is the maximal length for any mention. This complexity is the same as that of a zero-th order semi-Markov CRF model (Sarawagi and Cohen, 2005). Please refer to the supplementary material for a detailed explanation of the inference algorithm.

	ACE-2004		GENIA	
	Train (%)	Test (%)	Train (%)	Test (%)
# sentences	6,799	879	14,836	1,855
with o.l.	2,683 (39)	272 (42)	3,199 (22)	448 (24)
# mentions	22,207	3,031	46,473	5,600
o.l.	10,170 (46)	1,418 (47)	8,337 (18)	1,217 (22)
o.l. (st)	5,431 (24)	780 (26)	4,613 (10)	634 (11)
o.l. (st & slb)	2,188 (10)	307 (10)	2,133 (5)	287 (5)
lengh > 6	1,439 (6)	199 (7)	2,449 (5)	301 (5)
max lengh	57	43	28	19

Table 2: Statistics (ACE04, GENIA). *o.l.*: overlapping mentions, *st/slb*: same type/left boundary.

6 Experiments

6.1 Datasets

We mainly evaluate our models on the standard ACE-2004, ACE-2005 (Doddington et al., 2004), and GENIA (Kim et al., 2003) datasets with the same splits used by previous works (Lu and Roth, 2015; Muis and Lu, 2017). Sample data statistics of these datasets are listed in Table 2. [5] We can see that overlapping mentions frequently appear in such datasets. For ACE2004, over 46% of the mentions overlap with one another. GENIA focuses on biomedical entity recognition[6] and overlapping mentions are also common in it. Most mentions (over 93%) are not longer than 6 tokens which we select as maximal length (c) for the restricted models.

6.2 Baseline Approaches

We consider the following baseline models:

- CRF (LINEAR): a linear-chain CRF model. Since the linear-chain CRF cannot handle overlapping structures, we only use the outer-most mentions for learning. Specifically, every outer-most mention is labeled based on the BILOU tagging scheme, which was empirically shown to be better than the BIO scheme (Ratinov and Roth, 2009).
- CRF (CASCADED): the cascaded CRF based approach following (Alex et al., 2007). Note that this approach cannot model the overlapping mentions of the same type.
- Semi-CRF: the semi-Markov CRF model (Sarawagi and Cohen, 2005). The semi-CRF model is also only trained on the outer-most mentions. It can also capture span-level fea-

---

[5] See supplementary material for complete data statistics.

[6] Following previous works, we used version 3.02p which comes with annotated POS tags (Tateisi, 2004) . Following (Finkel and Manning, 2009), we collapse *DNA*, *RNA* and *protein* subtypes into *DNA*, *RNA* and *protein* respectively, keep cell line and cell type and remove mentions of other types.

		ACE-2004			ACE-2005			GENIA		
		P	R	F_1	P	R	F_1	P	R	F_1
Non-Neural	CRF (LINEAR)	71.8	40.8	52.1	69.5	44.5	54.2	77.1	63.3	69.5
	CRF (CASCADED)	78.4	46.4	58.3	74.8	49.1	59.3	75.9	66.1	70.6
	Semi-CRF (c=6)	76.1	41.4	53.6	72.8	45.0	55.6	74.5	66.0	70.0
	Semi-CRF (c=n)	66.7	42.0	51.5	67.5	46.1	54.8	74.2	65.8	69.7
	Finkel and Manning (2009)	-	-	-	-	-	-	75.4	65.9	70.3
	Lu and Roth (2015)	70.0	56.9	62.8	66.3	59.2	62.5	74.2	66.7	70.3
	Muis and Lu (2017)	72.7	58.0	64.5	69.1	58.1	63.1	75.4	66.8	70.8
	SH (-NN, c=6)	69.4	57.0	62.0	70.3	55.8	62.2	77.0	66.1	71.1
	SH (-NN, c=n)	71.1	60.6	65.4	69.5	60.7	64.8	76.2	67.5	71.6
Neural	FOFE (Xu et al., 2017) (c=6)	68.2	54.3	60.5	67.4	55.1	60.6	71.2	64.3	67.6
	FOFE (Xu et al., 2017) (c=n)	57.3	46.8	51.5	56.3	44.6	49.8	63.2	59.3	61.2
	Katiyar and Cardie (2018)	73.6	71.8	72.7	70.6	70.4	70.5	**79.8**	68.2	73.6
	Ju et al. (2018)[7]	-	-	-	74.2	70.3	72.2	78.5	71.3	74.7
	Wang et al. (2018)	74.9	71.8	73.3	74.5	71.5	73.0	78.0	70.2	73.9
	SH (c=6)	79.1	67.3	72.7	75.7	69.6	72.5	76.6	71.0	73.7
	SH (c=6) + *char*	**80.1**	67.5	73.3	75.9	70.0	72.8	76.8	71.8	74.2
	SH (c=n)	77.7	72.1	74.5	76.6	71.9	74.2	76.1	72.9	74.5
	SH (c=n) + *char*	78.0	**72.4**	**75.1**	**76.8**	**72.3**	**74.5**	77.0	**73.3**	**75.1**

Table 3: Main results. SH: segmental hypergraphs (our approach).

tures defined over a complete segment. Similar to our model, semi-CRF typically comes with a length restriction (c) which indicates the maximal length of a mention.

- Finkel and Manning (2009): a parsing-based approach for recognizing nested mentions that reported results on the GENIA dataset.
- Lu and Roth (2015): the model that makes use of mention hypergraphs for recognizing overlapping mentions.
- Muis and Lu (2017): the model that makes use of mention separators to tag gaps between words for recognizing overlapping mentions.
- FOFE (Xu et al., 2017): a local classifier based on neural networks that runs on every possible span to detect mentions. The maximal mention length (c) can also be used here.
- Katiyar and Cardie (2018): a hypergraph-based model that uses LSTM for learning feature representations.
- Ju et al. (2018): a cascaded model that makes use of multiple LSTM-CRF layers to recognize mentions in an inside-out manner.
- Wang et al. (2018): a neural transition-based model that construct nested mentions through a sequence of actions.
- SH (-NN): a non-neural version of our segmental hypergraph model that excludes the LSTMs but employs handcrafted features. [8]

	ACE-2004		ACE-2005		GENIA	
	(c=6)	(c=n)	(c=6)	(c=n)	(c=6)	(c=n)
SH	72.7	74.5	72.5	74.2	73.7	74.5
-D	71.5	73.1	71.3	72.9	72.1	72.8
-SM	72.0	73.3	71.8	73.5	72.4	73.3
-P	71.5	72.7	71.2	73.0	72.0	73.2

Table 4: Results of various ablations. D: dropout, SM: softmax-margin, P: pre-trained embeddings.

As discussed earlier, we also evaluate the variants of our model that takes character-level representations (+*char*).

6.3 Training

Pre-trained embeddings GloVe (Pennington et al., 2014) of dimension 100 are used to initialize the trainable word vectors for experiments in ACE and GENIA datasets.[9] The embeddings for POS tags are initialized randomly with dimension 32. Early stopping is used based on the performance of development set. The value β used in softmax-margin is chosen from [1, 3] with step size 0.5.

6.4 Experimental Results

Main results can be found in Table 3. Using the same set of handcrafted features, our unrestricted non-neural model SH (-NN, c=n) achieves the best performance compared with other non-neural models, revealing the effectiveness of our newly proposed segmental hypergraph representation. It achieves around 1-2% gain in terms of F_1 compared with mention hypergraph of Lu and Roth (2015) and mention separator of Muis and Lu (2017), showing the necessity of eliminating structural ambiguity. CRF (LINEAR) and Semi-CRF do not perform well due to incapability of handling

---

[7] Note that in ACE2005, Ju et al. (2018) did their experiments with a different split than Lu and Roth (2015) which we follow as our split.

[8] To make a proper comparison, we use the same handcrafted features used by (Lu and Roth, 2015), which were proven effective in previous approaches.

[9] We also additionally tried using embeddings trained on PubMed for GENIA but the performance was comparable.

overlapping mentions. In contrast, the pipeline approach CRF (CASCADED) performs better.

Our unrestricted neural segmental hypergraph model SH (c=n) already achieves the best results among all previous models in ACE datasets, showing the effectiveness of our neural segmental hypergraph. The improvement mainly comes from its ability to recall more mentions. In GENIA, even without using external features like Brown clustering features as all non-neural models do, our neural models still get significant improvements. Compared with the non-neural SH (-NN) which has around 4.2M parameters, our neural model SH only has 1.9M parameters yet it still performs better. We empirically see that the representations learned by LSTM can better capture complex contextual dependencies in sentences. The character-level representations (+ *char*) make both restricted and unrestricted SH perform even better. Particularly, SH (c=n) + *char* achieves the best results in all datasets compared with other recent neural models (Katiyar and Cardie, 2018; Ju et al., 2018; Wang et al., 2018).

One hypothesis we may have is that, without length restriction, a model will enjoy the benefit of recalling more long mentions, but also will be exposed to more false positives. This poses a challenge for a model – whether it is capable of balancing these two factors. Empirically, we find that the length restriction (c=6) improves the precision of semi-CRF and SH at the expense of the recall, providing some evidence to support the hypothesis. However, in terms of F_1, the unrestricted semi-CRF performs worse while unrestricted SH performs better compared to their restricted counterparts. The reason is that the span-level handcrafted features that the semi-CRF relies on can be very sparse when mentions are overly long. We empirically found this issue is alleviated in the model SH (-NN), possibly due to its ability in capturing interactions between neighboring spans. Even with length restriction, SH still yields competitive results, making it attractive in processing large-scale datasets considering its linear time complexity. Furthermore, we find that as c increases, SH performs better consistently in terms of F_1. The choice of c then becomes a tradeoff between time complexity and performance. Please refer to the supplementary material for details.

Compared with the local approach FOFE, our global approach gives a much better performance, showing its effectiveness in capturing interactions between spans. Moreover, FOFE's performance suffers significantly in the absence of the length restriction. The reason is that it will generate much more negative training instances under this setting, which makes its learning more challenging.

	Overlapping			Non-Overlapping			*w/s*
	P	R	F_1	P	R	F_1	
Lu and Roth (2015)	68.1	52.6	59.4	64.1	65.1	64.6	503
Muis and Lu (2017)	70.4	55.0	61.8	67.2	63.4	65.2	253
Wang et al. (2018)	77.4	70.5	73.8	76.1	69.6	72.7	1445
SH (c=6)	80.2	68.3	73.8	74.8	70.0	72.3	248
SH (c=n)	80.6	73.6	76.9	75.5	71.5	73.4	157

Table 5: Results on different types of sentences (ACE05), *w/s*: # of words decoded per second.

6.5 Additional Analyses

To understand our model better, we conduct some further experiments in this section.

Ablation study

We first conduct an ablation study by removing dropout, softmax-margin and pre-trained embeddings from our model respectively. The results are shown in Table 4. The dropout and pre-trained embeddings can improve the performance of our model significantly and this behavior is consistent with previous neural models for NER (Chiu and Nichols, 2016; Lample et al., 2016). Meanwhile, our new cost function based on softmax margin training also contributes significantly to the good performance of our model across these datasets.

How well does it handle overlapping mentions?

To further understand how well our model can handle overlapping mentions, we split the test data into two portions: sentences with and without overlapping mentions. We compare our model with the two state-of-the-art models and report results on ACE-05 in Table 5.[10] In both portions, SH achieves significant improvements, especially in the portion with overlapping mentions. This observation indicates that our model can better capture the structure of overlapping mentions than these two previous models. It also helps explain why the margin of improvement is larger in ACE than in GENIA since the former has more overlapping mentions than the latter, as shown in Table 2. Compared with the model with length restriction c, the unrestricted model mainly benefits from its ability to recall more overlapping mentions.

---

[10] Full results are listed in the supplementary material.

Model	F_1
SH (c=6)	89.6
SH (c=6) + *char*	90.5
SH (c=n)	89.2
SH (c=n) + *char*	90.2
Collobert et al. (2011)	88.7
Chiu and Nichols (2016)	90.9
Lample et al. (2016)	90.9
Ma and Hovy (2016)	91.2
Xu et al. (2017)	90.7
Strubell et al. (2017)	90.5

Table 6: Additional results on CoNLL-2003.

Running time

Since other compared models also feature linear time complexity (see Table 1), we examine the decoding speed in terms of the number of words processed per second. We re-implement the models of Lu and Roth (2015) and Muis and Lu (2017) using the same platform as ours (PyTorch) and run them on the same machine (CPU: Intel i5 2.7 GHz). The model of (Wang et al., 2018) is also tested with the same environment. Results on ACE-05 are listed in Table 5. The length bound (c=6) makes our model much faster, resulting in a speed comparable to the model of Muis and Lu (2017). The transition-based model by (Wang et al., 2018) has the best scalability partially because of its greedy strategy for decoding.

What if the data has no overlapping mentions?

To assess the robustness of our model and understand whether it could serve as a general mention extraction model, we additionally evaluate our model on CoNLL 2003 dataset which is annotated with non-overlapping mentions only. We compared our model with recent state-of-the-art neural network based models. For a fair comparison, we used the Collobert et al. (2011) embeddings widely used by previous models, and ignored POS tag features even though they are available. Results are in Table 6. Only neural models without using external features are included. [11] By only relying on word (and character) embeddings, our model achieves competitive results compared with other state-of-the-art neural models that also do not exploit external features, yet these models are mostly designed to handle only non-overlapping mentions. The only exception is the FOFE approach by (Xu et al., 2017) as we discussed earlier.

Notes on mention interactions

The dependencies between overlapping mentions can be very beneficial. SH can capture a specific kind of interaction between neighboring spans. Such interactions happen between mentions that share the same type and the same left boundary. As we can see from the sentence in Figure 3, one mention could also serve as a pre-modifier for another mention and both could share the same type. As shown in Table 2, there are over 8% such mentions in ACE and over 4% in GENIA. Specifically, SH relies on the hyperedges between **I** nodes to capture such interactions explicitly. To verify the effectiveness of this connection, we zero the weights between **I** nodes. The ablated model only achieves around 70.0% in ACEs and 71.4% in GENIA, implying the impact of this dependency connection. On the other hand, it also reveals the potential direction of improving SH by explicitly modeling more dependencies between mentions, such as the dependencies between mentions with different types. LSTM that serves as feature representation may capture such interactions implicitly, but building the connections could still be an important aspect for improvement.

7 Conclusion and Future Work

In this work, we propose a novel neural segmental hypergraph model that is able to capture overlapping mentions. We show that our model has some theoretical advantages over previous state-of-the-art approaches for recognizing overlapping mentions. Through extensive experiments, we show that our model is general and robust in handling both overlapping and non-overlapping mentions. The model achieves the state-of-the-art results in three standard datasets for recognizing overlapping mentions. We anticipate this model could be leveraged in other similar sequence modeling tasks that involve predicting overlapping structures such as recognizing overlapping and discontinuous entities (Muis and Lu, 2016) which frequently exist in the biomedical domain.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. This work was done after the first author visited Singapore University of Technology and Design. This work is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156.

---

[11] See the supplementary material for complete results.

References

Steven Abney, Michael Collins, and Amit Singhal. 2000. Answer extraction. In *Proc. of the sixth conference on applied natural language processing*.

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Proc. of BioNLP*.

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

Andreas Brandstädt, Feodor Dragan, Victor Chepoi, and Vitaly Voloshin. 1998. Dually chordal graphs. *SIAM Journal on Discrete Mathematics*, 11(3):437–455.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proc. of EMNLP*.

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

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

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Proc. of LREC*.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proc. of EMNLP*.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proc. of HLT-NAACL*.

Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. 1993. Directed hypergraphs and applications. *Discrete applied mathematics*, 42(2-3):177–201.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin crfs: Training log-linear models with cost functions. In *Proc. of HLT-NAACL*.

Ralph Grishman. 1997. Information extraction: Techniques and challenges. In *International Summer School on Information Extraction*, pages 10–27. Springer.

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

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

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proc. of NAACL-HLT*.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proc. of NAACL-HLT*.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *Proc. of ICLR*.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL-HLT*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. of ACL*.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. In *Proc. of EMNLP*.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proc. of EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of ACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Flexible text segmentation with structured multilabel classification. In *Proc. of HLT-EMNLP*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proc. of ACL-IJCNLP*.

Behrang Mohit, Nathan Schneider, Rishav Bhowmick, Kemal Oflazer, and Noah A Smith. 2012. Recall-oriented learning of named entities in arabic wikipedia. In *Proc. of EACL*.

Aldrian Obaja Muis and Wei Lu. 2016. Learning to recognize discontiguous entities. In *Proc. of EMNLP*.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proc. of EMNLP*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of CoNLL*.

S. Riedel and A. McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proc. of EMNLP*.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In *Proc. of NIPS*.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proc. of EMNLP*.

Yuka Tateisi. 2004. Part-of-speech annotation of biology research abstracts. In *Proc. of LREC*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018. A neural transition-based model for nested mention recognition. In *Proc. of EMNLP*.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proc. of ACL*.

Jie Zhang, Dan Shen, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2004. Enhancing hmm-based biomedical named entity recognition by studying special phenomena. *Journal of Biomedical Informatics*, 37(6):411–422.

Guodong Zhou. 2006. Recognizing names in biomedical texts using mutual information independence model and svm plus sigmoid. *International Journal of Medical Informatics*, 75(6):456–467.

Guodong Zhou, Jie Zhang, Jian Su, Dan Shen, and Chewlim Tan. 2004. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics*, 20(7):1178–1190.

Variational Sequential Labelers for Semi-Supervised Learning

Mingda Chen **Qingming Tang** **Karen Livescu** **Kevin Gimpel**
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
{mchen,qmtang,klivescu,kgimpel}@ttic.edu

Abstract

We introduce a family of multitask variational methods for semi-supervised sequence labeling. Our model family consists of a latent-variable generative model and a discriminative labeler. The generative models use latent variables to define the conditional probability of a word given its context, drawing inspiration from word prediction objectives commonly used in learning word embeddings. The labeler helps inject discriminative information into the latent space. We explore several latent variable configurations, including ones with hierarchical structure, which enables the model to account for both label-specific and word-specific information. Our models consistently outperform standard sequential baselines on 8 sequence labeling datasets, and improve further with unlabeled data.

1 Introduction

Sequence labeling tasks in natural language processing (NLP) often have limited annotated data available for model training. In such cases regularization can be important, and it can be helpful to use additional unlabeled data. One approach for both regularization and semi-supervised training is to design latent-variable generative models and then develop neural variational methods for learning and inference (Kingma and Welling, 2014; Rezende and Mohamed, 2015).

Neural variational methods have been quite successful for both generative modeling and representation learning, and have recently been applied to a variety of NLP tasks (Mnih and Gregor, 2014; Bowman et al., 2016; Miao et al., 2016; Serban et al., 2017; Zhou and Neubig, 2017; Hu et al., 2017). They are also very popular for semi-supervised training; when used in such scenarios, they typically have an additional task-specific prediction loss (Kingma et al., 2014; Maale et al., 2016; Zhou and Neubig, 2017; Yang et al., 2017b). However, it is still unclear how to use such methods in the context of sequence labeling.

In this paper, we apply neural variational methods to sequence labeling by combining a latent-variable generative model and a discriminatively-trained labeler. We refer to this family of procedures as variational sequential labelers (VSLs). Learning maximizes the conditional probability of each word given its context and minimizes the classification loss given the latent space. We explore several models within this family that use different kinds of conditional independence structure among the latent variables within each time step. Intuitively, the multiple latent variables can disentangle information pertaining to label-oriented and word-specific properties.

We study VSLs in the context of named entity recognition (NER) and several part-of-speech (POS) tagging tasks, both on English Twitter data and on data from six additional languages. Without unlabeled data, our models consistently show 0.5-0.8% accuracy improvements across tagging datasets and 0.8 F_1 improvement for NER. Adding unlabeled data further improves the model performance by 0.1-0.3% accuracy or 0.2 F_1 score. We obtain the best results with a hierarchical structure using two latent variables at each time step.

Our models, like generative latent variable models in general, have the ability to naturally combine labeled and unlabeled data. We obtain small but consistent performance improvements by adding unlabeled data. In the absence of unlabeled data, the variational loss acts as regularizer on the learned representation of the supervised sequence prediction model. Our results demonstrate that this regularization improves performance even when only labeled data is used. We also compare different ways of applying the classification loss when using a latent variable hierar-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 215–226
Brussels, Belgium, October 31 - November 4, 2018.

chy, and find that the most effective structure also provides the cleanest separation of information in the latent space.

2 Related Work

There is a growing amount of work applying neural variational methods to NLP tasks, including document modeling (Mnih and Gregor, 2014; Miao et al., 2016; Serban et al., 2017), machine translation (Zhang et al., 2016), text generation (Bowman et al., 2016; Serban et al., 2017; Hu et al., 2017), language modeling (Bowman et al., 2016; Yang et al., 2017b), and sequence transduction (Zhou and Neubig, 2017), but we are not aware of any such work for sequence labeling. Before the advent of neural variational methods, there were several efforts in latent variable modeling for sequence labeling (Quattoni et al., 2007; Sun and Tsujii, 2009).

There has been a great deal of work on using variational autoencoders in semi-supervised settings (Kingma et al., 2014; Maale et al., 2016; Zhou and Neubig, 2017; Yang et al., 2017b). Semi-supervised sequence labeling has a rich history (Altun et al., 2006; Jiao et al., 2006; Mann and McCallum, 2008; Subramanya et al., 2010; Søgaard, 2011). The simplest methods, which are also popular currently, use representations learned from large amounts of unlabeled data (Miller et al., 2004; Owoputi et al., 2013; Peters et al., 2017). Recently, Zhang et al. (2017) proposed a structured neural autoencoder that can be jointly trained on both labeled and unlabeled data.

Our work involves multi-task losses and is therefore also related to the rich literature on multi-task learning for sequence labeling (Plank et al., 2016; Augenstein and Søgaard, 2017; Bingel and Søgaard, 2017; Rei, 2017, *inter alia*).

Another related thread of work is learning interpretable latent representations. Zhou and Neubig (2017) factorize an inflected word into lemma and morphology labels, using continuous and categorical latent variables. Hu et al. (2017) interpret a sentence as a combination of an unstructured latent code and a structured latent code, which can represent attributes of the sentence.

There have been several efforts in combining variational autoencoders and recurrent networks (Gregor et al., 2015; Chung et al., 2015; Fraccaro et al., 2016). While the details vary, these models typically contain latent variables at each time step in a sequence. This prior work mainly focused on ways of parameterizing the time dependence between the latent variables, which gives them more power in modeling distributions over observation sequences. In this paper, we similarly use latent variables at each time step, but we adopt stronger independence assumptions which leads to simpler models and inference procedures. Also, the models cited above were developed for modeling data distributions, rather than for supervised or semi-supervised learning, which is our focus here.

The key novelties in our work compared to the prior work mentioned above are the proposed sequential variational labelers and the investigation of latent variable hierarchies within these models. The empirical effectiveness of latent hierarchical structure in variational modeling is a key contribution of this paper and may be applicable to the other applications discussed above. Recent work, contemporaneous with this submission, similarly showed the advantages of combining hierarchical latent variables and variational learning for conversational modeling, in the context of a non-sequential model (Park et al., 2018).

3 Proposed Methods

We begin by describing variational autoencoders and the notation we will use in the following sections. We denote the input word sequence by $x_{1:T}$, the corresponding label sequence by $l_{1:T}$, the input words other than the word at position t by x_{-t}, the generative model by $p_\theta(\cdot)$, and the posterior inference model by $q_\phi(\cdot)$.

3.1 Background: Variational Autoencoders

We review variational autoencoders (VAEs) by describing a VAE for an input sequence $x_{1:T}$. When using a VAE, we assume a generative model that generates an input using a latent variable z, typically assumed to follow a multivariate Gaussian distribution. We seek to maximize the marginal likelihood of inputs $x_{1:T}$ when marginalizing out the latent variable z. Since this is typically intractable, especially when using continuous latent variables, we instead maximize a lower bound on the marginal log-likelihood (Kingma and Welling,

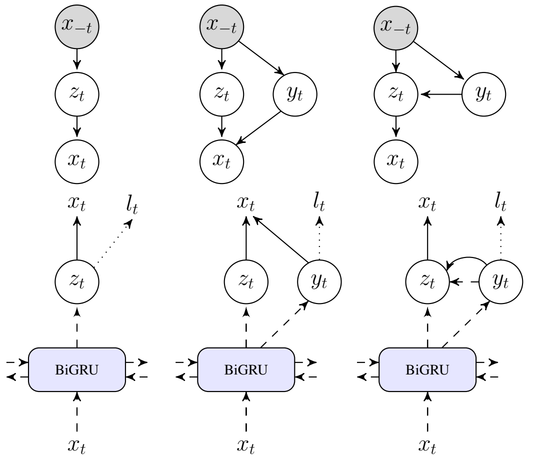

(a) VSL-G (b) VSL-GG-Flat (c) VSL-GG-Hier

Figure 1: Variational sequential labelers. The first row shows the original graphical models of each variant where shaded circles are observed variables. The second row shows how we perform inference and learning, showing inference models (in dashed lines), generative models (in solid lines), and classifier (in dotted lines). All models are trained to maximize $p_\theta(x_t|x_{-t})$ and predict the label l_t.

2014):

$$\log\ p_\theta(x_{1:T}) \geq \mathop{\mathbb{E}}_{z\sim q_\phi(\cdot|x_{1:T})}\left[\log p_\theta(x_{1:T}|z) - \log\frac{q_\phi(z|\ x_{1:T})}{p_\theta(z)}\right] = \underbrace{\mathop{\mathbb{E}}_{z\sim q_\phi(\cdot|x_{1:T})}[\log p_\theta(x_{1:T}|z)]}_{\text{Reconstruction Loss}} - \underbrace{KL(q_\phi(z|x_{1:T})\|p_\theta(z))}_{\text{KL divergence}} \tag{1}$$

where we have introduced the variational posterior q parametrized by new parameters ϕ. q is referred to as an "inference model" as it encodes an input into the latent space. We also have the generative model probabilities p parametrized by θ. The parameters are trained in a way that reflects a classical autoencoder framework: encode the input into a latent space, decode the latent space to reconstruct the input. These models are therefore referred to as "variational autoencoders".

The lower bound consists of two terms: reconstruction loss and KL divergence. The KL divergence term provides a regularizing effect during learning by ensuring that the learned posterior remains close to the prior over the latent variables.

3.2 Variational Sequential Labelers

We now introduce variational sequential labelers (VSLs) and propose several variants for sequence labeling tasks. Although the latent structure varies, a VSL maximizes the conditional probability of $p_\theta(x_t|x_{-t})$ and minimizes a classification loss using the latent variables as the input to the classifier. Unlike VAEs, VSLs do not autoencode the input, so they are more similar to recent conditional variational formulations (Sohn et al., 2015; Miao et al., 2016; Zhou and Neubig, 2017). Intuitively, the VSL variational objective is to find the information that is useful for predicting the word x_t from its surrounding context, which has similarities to objectives for learning word embeddings (Collobert et al., 2011; Mikolov et al., 2013). This objective serves as regularization for the labeled data and as an unsupervised objective for the unlabeled data.

All of our models use latent variables for each position in the sequence. These characteristics are shown in the visual depictions of our models in Figure 1. We consider variants with multiple latent variables per time step and attach the classifier to only particular variables. This causes the different latent variables to capture different characteristics.

In the following sections, we will describe various latent variable configurations that we will evaluate empirically in subsequent sections.

3.3 Single Latent Variable

We begin by defining a basic VSL and corresponding parametrization, which will also be used in other variants. This first model (which we call VSL-G and show in Figure 1a) has a Gaussian latent variable at each time step. VSL-G uses two training objectives; the first is similar to the lower bound on log-likelihood used by VAEs:

$$\log\ p_\theta(x_t|x_{-t}) \geq \mathop{\mathbb{E}}_{z_t\sim q_\phi(\cdot|\ x_{1:T},t)}[\log p_\theta(x_t|\ z_t) - \log\frac{q_\phi(z_t|\ x_{1:T},t)}{p_\theta(z_t|\ x_{-t})}] = \mathop{\mathbb{E}}_{z_t\sim q_\phi(\cdot|\ x_{1:T},t)}[\log p_\theta(x_t|\ z_t)] - KL(q_\phi(z_t|\ x_{1:T},t)\|\ p_\theta(z_t|\ x_{-t})) = U_0(x_{1:T},t) \tag{2}$$

VSL-G additionally uses a classifier f on the latent variable z_t which is trained with the following objective:

$$C_0(x_{1:T},l_t) = \mathop{\mathbb{E}}_{z_t\sim q_\phi(\cdot|x_{1:T},t)}[-\log f(l_t|z_t)] \tag{3}$$

The final loss is

$$L(x_{1:T},l_{1:T}) = \sum_{t=1}^{T}[C_0(x_{1:T},l_t) - \alpha U_0(x_{1:T},t)]$$

where α is a trade-off hyperparameter. α is set to zero during supervised training but it is tuned based on development set performance during semi-supervised training. The same procedure is adopted for the other VSL models below.

For the generative model, we parametrize $p_\theta(x_t|z_t)$ as a feedforward neural network with two hidden layers and ReLU (Nair and Hinton, 2010) as activation function. As reconstruction loss, we use cross-entropy over the words in the vocabulary. We defer the descriptions of the parametrization of $p_\theta(z_t|\ x_{-t})$ to Section 3.6.

We now discuss how we parametrize the inference model $q_\phi(z_t|x_{1:T}, t)$. We use a bidirectional gated recurrent unit (BiGRU; Chung et al., 2014) network to produce a hidden vector h_t at position t. The BiGRU is run over the input $x_{1:T}$, where each x_t is the concatenation of a word embedding and the concatenated final hidden states from a character-level BiGRU. The inference model $q_\phi(z_t|x_{1:T}, t)$ is then a single layer feedforward neural network that uses h_t as input. When parametrizing the posterior over latent variables in the following models below, we use this same procedure to produce hidden vectors with a BiGRU and then use them as input to feedforward networks. The structure of our inference model is similar to those used in previous state-of-the-art models for sequence labeling (Lample et al., 2016; Yang et al., 2017a).

In order to focus more on the effect of our variational objective, the classifier we use is always the same as our baseline model (see Section 4.3), which is a one layer feedforward neural network without a hidden layer, and it is also used in test-time prediction.

3.4 Flat Latent Variables

We next consider ways of factorizing the functionality of the latent variable into label-specific and other word-specific information. We introduce VSL-GG-Flat (shown in Figure 1b), which has two conditionally independent Gaussian latent variables at each time step, denote z_t and y_t for time step t. The variational lower bound is derived as follows:

$$\begin{aligned}
&\log\ p_\theta(x_t|x_{-t}) \geq \\
&\quad \mathop{\mathbb{E}}_{z_t, y_t \sim q_\phi(\cdot|x_{1:T}, t)} [\log p_\theta(x_t|\ z_t, y_t) \\
&\quad - \log \frac{q_\phi(z_t|x_{1:T}, t)}{p_\theta(z_t|x_{-t})} - \log \frac{q_\phi(y_t|x_{1:T}, t)}{p_\theta(y_t|x_{-t})}] \\
&\quad = \mathop{\mathbb{E}}_{z_t, y_t \sim q_\phi(\cdot|x_{1:T}, t)} [\log p_\theta(x_t|z_t, y_t)] \\
&\quad - KL(q_\phi(z_t|x_{1:T}, t) \| p_\theta(z_t|x_{-t})) \\
&\quad - KL(q_\phi(y_t|x_{1:T}, t) \| p_\theta(y_t|x_{-t})) \\
&\quad = U_1(x_{1:T}, t)
\end{aligned} \tag{4}$$

The classifier f is on the latent variable y_t and its loss is

$$C_1(x_{1:T}, l_t) = \mathop{\mathbb{E}}_{y_t \sim q_\phi(\cdot|x_{1:T}, t)} [-\log f(l_t|y_t)] \tag{5}$$

The final loss for the model is

$$L(x_{1:T}, l_{1:T}) = \sum_{t=1}^{T} [C_1(x_{1:T}, l_t) - \alpha U_1(x_{1:T}, t)] \tag{6}$$

Where α is a trade-off hyperparameter.

Similarly to the VSL-G model, $q_\phi(z_t|x_{1:T}, t)$ and $q_\phi(y_t|x_{1:T}, t)$ are parametrized by single layer feedforward neural networks using the hidden state h_t as input.

3.5 Hierarchical Latent Variables

We also explore hierarchical relationships among the latent variables. In particular, we introduce the VSL-GG-Hier model which has two Gaussian latent variables with the hierarchical structure shown in Figure 1c. This model encodes the intuition that the word-specific latent information z_t may differ depending on the class-specific information of y_t.

For this model, the derivations are similar to Equations (4) and (5). The first is:

$$\begin{aligned}
&\log\ p_\theta(x_t|x_{-t}) \geq \\
&\quad \mathop{\mathbb{E}}_{z_t, y_t \sim q_\phi(\cdot|x_{1:T}, t)} [\log p_\theta(x_t|z_t) \\
&\quad - \log \frac{q_\phi(z_t|y_t, x_{1:T}, t)}{p_\theta(z_t|y_t, x_{-t})} - \log \frac{q_\phi(y_t|x_{1:T}, t)}{p_\theta(y_t|x_{-t})}] \\
&\quad = \mathop{\mathbb{E}}_{z_t, y_t \sim q_\phi(\cdot|x_{1:T}, t)} [\log p_\theta(x_t|z_t)] \\
&\quad - KL(q_\phi(z_t|y_t, x_{1:T}, t) \| p_\theta(z_t|y_t, x_{-t})) \\
&\quad - KL(q_\phi(y_t|x_{1:T}, t) \| p_\theta(y_t|x_{-t})) \\
&\quad = U_2(x_{1:T}, t)
\end{aligned} \tag{7}$$

The classifier f uses y_t as input and is trained with the following loss:

$$C_2(x_{1:T}, l_t) = \underset{y_t \sim q_\phi(\cdot|x_{1:T}, t)}{\mathbb{E}} [-\log f(l_t|y_t)] \quad (8)$$

Note that C_1 and C_2 have the same form. The final loss is

$$L(x_{1:T}, l_{1:T}) = \sum_{t=1}^{T} [C_2(x_{1:T}, l_t) - \alpha\, U_2(x_{1:T}, t)] \quad (9)$$

Where α is a trade-off hyperparameter.

The hierarchical posterior $q_\phi(z_t|y_t, x_{1:T}, t)$ is parametrized by concatenating the hidden vector h_t and the random variable y_t and then using them as input to a single layer feedforward network.

3.6 Parametrization of Priors

Traditional variational models assume extremely simple priors (e.g., multivariate standard Gaussian distributions). Recently there have been efforts to learn the prior and posterior jointly during training (Fraccaro et al., 2016; Serban et al., 2017; Tomczak and Welling, 2018). In this paper, we follow this same idea but we do not explicitly parametrize the prior $p_\theta(z_t|x_{-t})$. This is partially due to the lack of computationally-efficient parametrization options for $p_\theta(z_t|x_{-t})$. In addition, since we are not seeking to do generation with our learned models, we can let part of the generative model be parametrized implicitly.

More specifically, the approach we use is to learn the priors by updating them iteratively. During training, we first initialize the priors of all examples as multivariate standard Gaussian distributions. As training proceeds, we use the last optimized posterior as our current prior based on a particular "update frequency" (see supplementary material for more details).

Our learned priors are implicitly modeled as

$$p_\theta^k(z_t|x_{-t}) \approx \sum_x q_\phi^{k-1}(z_t|X_t = x, x_{-t}, t) p_{\text{data}}(X_t = x|x_{-t}) \quad (10)$$

where p_{data} is the empirical data distribution, X_t is a random variable corresponding to the observation at position t, and k is the prior update time step. The intuition here is that the prior is obtained by marginalizing over values for the missing observation represented by the random variable X_t. The posterior q_ϕ^{k-1} is as defined in our latent variable models. We assume $p_{\text{data}}(X_t = x|x_{-t}) = 0$ for $x_{1:T} \notin$ training set. For context x_{-t} that can pair with multiple values of X_t, its prior is the data-dependent weighted average posterior. For simplicity of implementation and efficient computation, however, if context x_{-t} can pair with multiple values in our training data, we ignore this fact and simply use instance-dependent posteriors. Another way to view this is as conditioning on the index of the training examples while parametrizing the above. That is

$$p_\theta^{k,i}(z_t|x_{-t}) \leftarrow q_\phi^{k-1,i}(z_t|x_{1:T}, t) \quad (11)$$

where i is the index of the instance.

3.7 Training

In this subsection, we introduce techniques we have used to address difficulties during training.

Reparametrization Trick. It is challenging to use gradient descent for a random variable as it involves a non-differentiable sampling procedure. Kingma and Welling (2014) introduced a reparametrization trick to tackle this problem. They parametrize a Gaussian random variable z as $u_\varphi(x) + g_\psi(x) \circ \epsilon$ where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and $u_\varphi(x)$, $g_\psi(x)$ are deterministic and differentiable functions, so the gradient can go through $u_\varphi(\cdot)$ and $g_\psi(\cdot)$. In our experiments, we use one sample for each time step during training. For evaluation at test time, we use the mean value $u_\varphi(x)$.

KL Divergence Weight Annealing. Although the use of prior updating lets us avoid tuning the weight of the KL divergence, the simple priors can still hinder learning during the initial stages of training. To address this, we follow the method described by Bowman et al. (2016) to add weights to all KL divergence terms and anneal the weights from a small value to 1.

4 Experiments

We describe key details of our experimental setup in the subsections below but defer details about hyperparameter tuning to the supplementary material. Our implementation is available at `https://github.com/mingdachen/vsl`

4.1 Datasets

We evaluate our model on the CoNLL 2003 English NER dataset (Tjong Kim Sang and De Meulder, 2003) and 7 POS tagging datasets: the

Twitter tagging dataset of Gimpel et al. (2011) and Owoputi et al. (2013), and 6 languages from the Universal Dependencies (UD) 1.4 dataset (McDonald et al., 2013).

Twitter POS Dataset. The Twitter dataset has 25 tags. We use OCT27TRAIN and OCT27DEV as the training set, OCT27TEST as the development set, and DAILY547 as the test set. We randomly sample {1k, 2k, 3k, 4k, 5k, 10k, 20k, 30k, 60k} tweets from 56 million English tweets as our unlabeled data and tune the amount of unlabeled data based on development set accuracy.

UD POS Datasets. The UD datasets have 17 tags. We use French, German, Spanish, Russian, Indonesian and Croatian. We follow the same setup as Zhang et al. (2017), randomly sampling 20% of the original training set as our labeled data and 50% as unlabeled data. There is no overlap between the labeled and unlabeled data. See Zhang et al. (2017) for more details about the setup.

NER Dataset. We use the BIOES labeling scheme and report micro-averaged F_1. We preprocessed the text by replacing all digits with 0. We randomly sample 10% of the original training set as our labeled data and 50% as unlabeled data. We also ensure there is no overlap between the labeled and unlabeled data.

4.2 Pretrained Word Embeddings

For all experiments, we use pretrained 100-dimensional word embeddings. For Twitter, we trained skip-gram embeddings (Mikolov et al., 2013) on a dataset of 56 million English tweets. For the UD datasets, we trained skip-gram embeddings on Wikipedia for each of the six languages. For NER, we use 100-dimensional pretrained GloVe (Pennington et al., 2014) embeddings. Our models perform better with word embeddings kept fixed during training while for the baselines the word embeddings are fine tuned as this improves the baseline performance.

4.3 Baselines

Our primary baseline is a BiGRU tagger where the input consists of the concatenation of a word embedding and the concatenation of the final hidden states of a character-level BiGRU. This BiGRU architecture is identical to that used in the inference networks in our VSL models. Predictions are made based on a linear transformation given the current hidden state. The output dimensionality of the transformation is task-dependent (e.g., 25 for Twitter tagging). We use the standard per-position cross entropy loss for training.

	dev.		test	
	acc.	ULΔ	acc.	ULΔ
BiGRU baseline	90.8	-	90.6	-
VSL-G	91.1	+0.1	-	-
VSL-GG-Flat	91.4	+0.1	-	-
VSL-GG-Hier	***91.6***	***+0.3***	***91.6***	*+0.3*

(a) Twitter tagging accuracies (%)

	dev.		test	
	F_1	ULΔ	F_1	ULΔ
BiGRU baseline	87.6	-	83.7	-
VSL-G	87.8	+0.1	-	-
VSL-GG-Flat	88.0	+0.1	-	-
VSL-GG-Hier	***88.4***	***+0.2***	***84.7***	*+0.0*

(b) NER F_1 score (%)

Table 1: For dev and test, we show results when only using labeled data and the change in performances ("ULΔ") when adding unlabeled data. Bold is highest in each column. Italic is the best model including unlabeled data. We only show test results for the baseline and our best-performing model, which achieves 91.9% accuracy on the Twitter test set and 84.7% F_1 on the NER test set when using unlabeled data.

We also report results from the best systems from Zhang et al. (2017), namely the NCRF and NCRF-AE models. Both use feedforward networks as encoders and conditional random field layers for capturing sequential information. The NCRF-AE model additionally can benefit from unlabeled data.

5 Results

Table 1a shows results on the Twitter development and test sets. All of our VSL models outperform the baseline and our best VSL models outperform the BiGRU baseline by 0.8–1% absolute. When comparing different latent variable configurations, we find that a hierarchical structure performs best. Without unlabeled data, our models already outperform the BiGRU baseline. Adding unlabeled data enlarges the gap between the baseline and our models by up to 0.1–0.3% absolute.

Table 1b shows results on the CoNLL 2003 NER development and test sets. We observe similar trends as in the Twitter data, except that the model does not show improvement on the test set when adding unlabeled data.

	French		German		Indonesian		Spanish		Russian		Croatian	
	acc.	ULΔ	acc.	ULΔ	acc.	ULΔ	acc.	ULΔ	acc.	ULΔ	acc.	ULΔ
NCRF	93.4	-	90.4	-	88.4	-	91.2	-	86.6	-	86.1	-
NCRF-AE	93.7	+0.2	90.8	+0.2	89.1	+0.3	91.7	+0.5	87.8	+1.1	87.9	+1.2
BiGRU baseline	95.9	-	92.6	-	92.2	-	94.7	-	95.2	-	95.6	-
VSL-G	96.1	+0.0	92.8	+0.0	92.3	+0.0	94.8	+0.1	95.3	+0.0	95.6	+0.1
VSL-GG-Flat	96.1	+0.0	93.0	**+0.1**	92.4	**+0.1**	95.0	+0.1	95.5	**+0.1**	95.8	+0.1
VSL-GG-Hier	***96.4***	***+0.1***	***93.3***	***+0.1***	***92.8***	***+0.1***	***95.3***	***+0.2***	***95.9***	***+0.1***	***96.3***	***+0.2***

Table 2: Tagging accuracies (%) on UD test sets. For each language, we show test accuracy ("acc.") when only using labeled data and the change in test accuracy ("ULΔ") when adding unlabeled data. Results for NCRF and NCRF-AE are from Zhang et al. (2017), though results are not strictly comparable because we used pretrained word embeddings for all languages on Wikipedia. Bold is highest in each column, excluding the NCRF variants. Italic is the best accuracy including the unlabeled data.

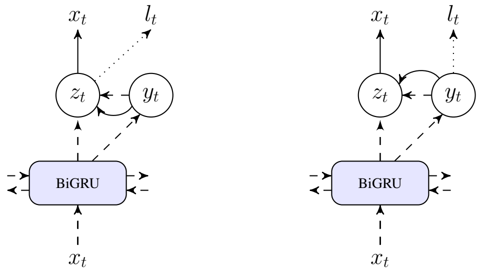

(a) VSL-GG-Hier with classification loss on z

(b) VSL-GG-Hier

Figure 2: Comparison of attaching classification loss to different latent variables in VSL-GG-Hier.

Table 2 shows our results on the UD datasets. The trends are broadly consistent with those of Table 1a and 1b. The best performing models use hierarchical structure in the latent variables. There are some differences across languages. For French, German, Indonesian and Russian, VSL-G does not show improvement when using unlabeled data. This may be resolved with better tuning, since the model actually shows improvement on the dev set.

Note that results reported by Zhang et al. (2017) and ours are not strictly comparable as their word embeddings were only pretrained on the UD training sets while ours were pretrained on Wikipedia. Nonetheless, they also mentioned that using embeddings pretrained on larger unlabeled data did not help. We include these results to show that our baselines are indeed strong compared to prior results reported in the literature.

	Twitter		NER		UD average	
	acc.	ULΔ	F_1	ULΔ	acc.	ULΔ
classifier on y	91.6	+0.3	88.4	+0.2	95.0	+0.1
classifier on z	91.1	+0.2	87.8	+0.1	94.4	+0.0

Table 3: Twitter and NER dev results (%), UD averaged test accuracies (%) for two choices of attaching the classification loss to latent variables in the VSL-GG-Hier model. All previous results for VSL-GG-Hier used the classification loss on y.

6 Discussion

6.1 Effect of Position of Classification Loss

We investigate the effect of attaching the classifier to different latent variables. In particular, for the VSL-GG-Hier model, we compare the attachment of the classifier between z and y. See Figure 2. The results in Table 3 suggest that attaching the reconstruction and classification losses to the same latent variable (z) harms accuracy although attaching the classifier to z effectively gives the classifier an extra layer. We can observe why this occurs by looking at the latent variable visualizations in Figure 3d. Compared with Figure 3e, where the two variables are more clearly disentangled, the latent variables in Figure 3d appear to be capturing highly similar information.

6.2 Effect of Latent Hierarchy

To verify our assumption of the latent structure, we visualize the latent space for Gaussian models using t-SNE (Maaten and Hinton, 2008) in Figure 3. The BiGRU baseline (Figure 3a) and the VSL-G (Figure 3b) do not show significant differences. However, when using multiple latent variables, the different latent variables capture different characteristics. In the VSL-GG-Flat model (Figure 3c), the y variable (the upper plot) reflects

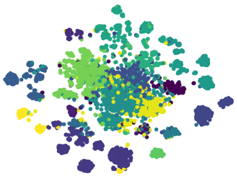

(a) BiGRU Baseline

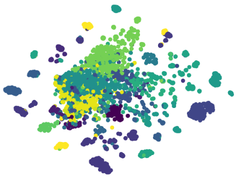

(b) VSL-G

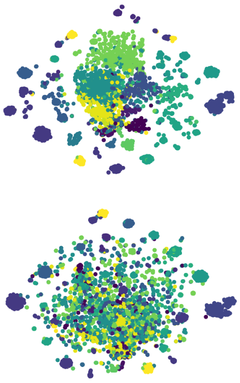

(c) VSL-GG-Flat

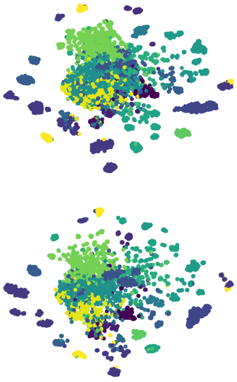

(d) VSL-GG-Hier, classification loss on z

(e) VSL-GG-Hier

Figure 3: t-SNE visualization of Gaussian latent variables and baseline hidden states for Twitter development set. In plot 3c, 3d, and 3e, the upper subplot is latent variable y and the lower is z. Each point in the plot is a token and the color represents the true tag of the token.

the clustering of the tagging space much more closely than the z variable (the lower plot). Since both variables are used to reconstruct the word, but only the y variable is trained to predict the tag, it appears that z is capturing other information useful for reconstructing the word. However, since they are both used for reconstruction, the two spaces show signs of alignment; that is, the "tag" latent variable y does not show as clean a separation into tag clusters as the y variable in the VSL-GG-Hier model in Figure 3e.

In Figure 3e (VSL-GG-Hier), the clustering of words with respect to the tag is clearest. This may account for the consistently better performance of this model relative to the others. The z variable reflects a space that is conditioned on y but that diverges from it, presumably in order to better reconstruct the word. The closer the latent variable is to the decoder output, the weaker the tagging information becomes while other word-specific information becomes more salient.

	Twitter		NER	
	acc.	no VR	F_1	no VR
BiGRU baseline	90.8	-	87.6	-
VSL-G	91.1	90.9	87.8	87.7
VSL-GG-Flat	91.4	90.9	88.0	87.8
VSL-GG-Hier	91.6	91.0	88.4	87.9

Table 4: Results on Twitter and NER dev sets. For each model, we show supervised results for the models with variational regularization ("acc." or F_1) and results when replacing variational components with their deterministic counterparts ("no VR").

Figure 3d shows that VSL-GG-Hier with classification loss on z, which consistently underperforms both the VSL-GG-Flat and VSL-GG-Hier models in our experiments, appears to be capturing the same latent space in both variables. Since the z variable is used to both predict the tag and reconstruct the word, it must capture both the tag and word reconstruction spaces, and may be limited by capacity in doing so. The y variable does not seem to be contributing much modeling power, as its space is closely aligned to that of z.

6.3 Effect of Variational Regularization

We investigate the beneficial effects of variational frameworks ("variational regularization") by replacing our variational components in VSLs with their deterministic counterparts, which do not have randomness in the latent space and do not use the KL divergence term during optimization. Note that these BiGRU encoders share the same architectures as their variational posterior counterparts and still use both the classification and reconstruction losses. While other subsets of losses could be considered in this comparison, our motivation is to compare two settings that correspond to well-known frameworks. The "no VR" setting corresponds roughly to the combination of a classifier and a traditional autoencoder. We note that these experiments do not use any unlabeled data.

The results in Table 4 demonstrate that compared to the baseline BiGRU, adding the reconstruction loss ("VSL-G, no VR") yields only 0.1 improvement for both Twitter and NER. Although adding hierarchical structure further improves performance, the improvements are small (+0.1 and +0.2 for Twitter and NER respectively). For VSL-

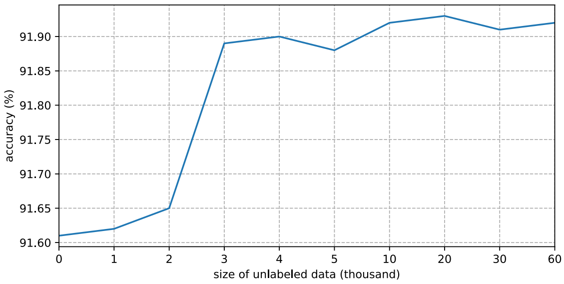

Figure 4: Twitter dev accuracies (%) when varying the amount of unlabeled data.

GG-Hier, variational regularization accounts for relatively large differences of 0.6 for Twitter and 0.5 for NER. These results show that the improvements do not come solely from adding a reconstruction objective to the learning procedure. In limited preliminary experiments, we did not find a benefit from adding unlabeled data under the "no VR" setting.

6.4 Effect of Unlabeled Data

In order to examine the effect of unlabeled data, we report our Twitter dev accuracies when varying the unlabeled data size. We choose VSL-GG-Hier as the model for this experiment since it benefits the most from unlabeled data. As Figure 4 shows, gradually adding unlabeled data helps a little at the beginning. Further adding unlabeled data boosts the accuracy of the model. The improvements that come from unlabeled data quickly plateau after the amount of unlabeled data goes beyond 10,000. This suggests that with little unlabeled data, the model is incapable of fully utilizing the information in the unlabeled data. However if the amount of unlabeled data is too large, the supervised training signal becomes too weak to extract something useful from the unlabeled data.

We also notice that when there is a large amount of unlabeled data, it is always better to pretrain the prior first using a small α (e.g., 0.1) and then use it as a warm start to train a new model using a larger α (e.g., 1.0). Tuning the weight of the KL divergence could achieve a similar effect, but it may require tuning the weight for labeled data and unlabeled data separately. We prefer to pretrain the prior as it is simpler and involves less hyperparameter tuning.

7 Conclusion

We introduced variational sequential labelers for semi-supervised sequence labeling. They consist of latent-variable generative models with flexible parametrizations for the variational posterior (using RNNs over the entire input sequence) and a classifier at each time step. Our best models use multiple latent variables arranged in a hierarchical structure. We demonstrate systematic improvements in NER and POS tagging accuracy across 8 datasets over a strong baseline. We also find small, but consistent, improvements by using unlabeled data.

Acknowledgments

We would like to thank NVIDIA for donating GPUs used in this research, the anonymous reviewers for their comments that improved this paper, and Google for a faculty research award to K. Gimpel that partially supported this research. This research was funded by NSF grant 1433485.

References

Y. Altun, D. McAllester, and M. Belkin. 2006. Maximum margin semi-supervised learning for structured variables. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 33–40. MIT Press.

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 341–346. Association for Computational Linguistics.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169. Association for Computational Linguistics.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21. Association for Computational Linguistics.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2980–2988. Curran Associates, Inc.

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

Marco Fraccaro, Søren Kaae Sø nderby, Ulrich Paquet, and Ole Winther. 2016. Sequential neural models with stochastic layers. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2199–2207. Curran Associates, Inc.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47. Association for Computational Linguistics.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1462–1471, Lille, France. PMLR.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia. PMLR.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 209–216, Sydney, Australia. Association for Computational Linguistics.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Lars Maale, Casper Kaae Snderby, Sren Kaae Snderby, and Ole Winther. 2016. Auxiliary deep generative models. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1445–1453, New York, New York, USA. PMLR.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Gideon S. Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of ACL-08: HLT*, pages 870–878. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97. Association for Computational Linguistics.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1727–1736, New York, New York, USA. PMLR.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT-NAACL 2004: Main Proceedings*, pages 337–342, Boston, Massachusetts, USA. Association for Computational Linguistics.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1791–1799, Bejing, China. PMLR.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390. Association for Computational Linguistics.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A hierarchical latent structure for variational conversation modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1792–1801. Association for Computational Linguistics.

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

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418. Association for Computational Linguistics.

Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Morency Collins, and Trevor Darrell. 2007. Hidden conditional random fields. *IEEE transactions on pattern analysis and machine intelligence*, 29(10).

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130. Association for Computational Linguistics.

Danilo Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France. PMLR.

Iulian Vlad Serban, Alexander G. Ororbia, Joelle Pineau, and Aaron Courville. 2017. Piecewise latent variables for neural variational text processing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 422–432. Association for Computational Linguistics.

Anders Søgaard. 2011. Semi-supervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 48–52, Portland, Oregon, USA. Association for Computational Linguistics.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3483–3491. Curran Associates, Inc.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176, Cambridge, MA. Association for Computational Linguistics.

Xu Sun and Jun'ichi Tsujii. 2009. Sequential labeling with latent variables: An exact inference algorithm and its efficient approximation. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 772–780. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Jakub Tomczak and Max Welling. 2018. Vae with a vampprior. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1214–1223, Playa Blanca, Lanzarote, Canary Islands. PMLR.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017a. Transfer learning for sequence tagging with hierarchical recurrent networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017b. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3881–3890, International Convention Centre, Sydney, Australia. PMLR.

Biao Zhang, Deyi Xiong, jinsong su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530. Association for Computational Linguistics.

Xiao Zhang, Yong Jiang, Hao Peng, Kewei Tu, and Dan Goldwasser. 2017. Semi-supervised structured prediction with neural crf autoencoder. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1701–1711. Association for Computational Linguistics.

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 310–320. Association for Computational Linguistics.

Joint Representation Learning of Cross-lingual Words and Entities via Attentive Distant Supervision

Yixin Cao[1,2] **Lei Hou**[2*] **Juanzi Li**[2] **Zhiyuan Liu**[2]
Chengjiang Li[2] **Xu Chen**[2] **Tiansi Dong**[3]
[1]School of Computing, National University of Singapore, Singapore
[2]Department of CST, Tsinghua University, Beijing, China
[3]B-IT, University of Bonn, Bonn, Germany
`{caoyixin2011,iamlockelightning,successcx}@gmail.com`
`{houlei,liuzy,lijuanzi}@tsinghua.edu.cn`
`dongt@bit.uni-bonn.de`

Abstract

Joint representation learning of words and entities benefits many NLP tasks, but has not been well explored in cross-lingual settings. In this paper, we propose a novel method for joint representation learning of cross-lingual words and entities. It captures mutually complementary knowledge, and enables cross-lingual inferences among knowledge bases and texts. Our method does not require parallel corpora, and automatically generates comparable data via distant supervision using multi-lingual knowledge bases. We utilize two types of regularizers to align cross-lingual words and entities, and design knowledge attention and cross-lingual attention to further reduce noises. We conducted a series of experiments on three tasks: word translation, entity relatedness, and cross-lingual entity linking. The results, both qualitatively and quantitatively, demonstrate the significance of our method.

1 Introduction

Multi-lingual knowledge bases (KB) store millions of entities and facts in various languages, and provide rich background structural knowledge for understanding texts. On the other hand, text corpus contains huge amount of statistical information complementary to KBs. Many researchers leverage both types of resources to improve various natural language processing (NLP) tasks, such as machine reading (Yang and Mitchell, 2017), question answering (He et al., 2017; Hao et al., 2017).

Most existing work jointly models KB and text corpus to enhance each other by learning word and entity representations in a unified vector space. For example, Wang et al. (2014); Yamada et al. (2016); Cao et al. (2017) utilize the co-occurrence information to align similar words and entities with similar embedding vectors. Toutanova et al. (2015); Wu et al. (2016); Han et al. (2016); Weston et al. (2013a); Wang and Li (2016) represent entities based on their textual descriptions together with the structured relations. These methods focused on mono-lingual settings. However, for cross-lingual tasks (e.g., cross-lingual entity linking), these approaches need to introduce additional tools to do translation, which suffers from extra costs and inevitable errors (Ji et al., 2015, 2016).

In this paper, we carry out cross-lingual joint representation learning, which has not been fully researched in the literature. We aim at creating a unified space for words and entities in various languages, and easing cross-lingual semantic comparison, which will benefit from the complementary information in different languages. For instance, two different meanings of word *center* in English are expressed by two different words in Chinese: *center* as *the activity-specific building* is expressed by 中心, *center* as *the basketball player role* is 中锋.

Our main challenge is the limited availability of parallel corpus, which is usually either expensive to obtain, or only available for certain narrow domains (Gouws et al., 2015). Many work has been done to alleviate the problem. One school of methods uses adversarial technique or domain adaption to match linguistic distribution (Zhang et al., 2017b; Barone, 2016; Cao et al., 2016). These methods do not require parallel corpora. The weakness is that the training process is unstable and that the high complexity restricts the methods only to small-scale data. Another line of work uses pre-existing multi-lingual resources to automatically generate "pseudo bilingual documents" (Vulic and Moens, 2015, 2016). However, negative results have been observed due to the occasional poor quality of training data (Vulic and Moens, 2016). All above methods only focus on words. We consider both words and entities, which

---

[*] Corresponding author.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 227–237
Brussels, Belgium, October 31 - November 4, 2018.

makes the parallel data issue more challenging.

In this paper, we propose a novel method for joint representation learning of cross-lingual words and entities. The basic idea is to capture mutually complementary knowledge in a shared semantic space, which enables joint inference among cross-lingual knowledge base and texts without additional translations. We achieve it by (1) utilizing an existing multi-lingual knowledge base to automatically generate cross-lingual supervision data, (2) learning mono-lingual word and entity representations, (3) applying cross-lingual sentence regularizer and cross-lingual entity regularizer to align similar words and entities with similar embeddings. The entire framework is trained using a unified objective function, which is efficient and applicable to arbitrary language pairs that exist in multi-lingual KBs.

Particularly, we build a bilingual entity network from inter-language links [1] in KBs for regularizing cross-lingual entities through a variant of skip-gram model (Mikolov et al., 2013c). Thus, mono-lingual structured knowledge of entities are not only extended to cross-lingual settings, but also augmented from other languages. On the other hand, we utilize **distant supervision** to generate comparable sentences for cross-lingual sentence regularizer to model co-occurrence information across languages. Compared with “pseudo bilingual documents”, comparable sentences achieve higher quality, because they rely not only on the shared semantics at document level, but also on cross-lingual information at sentence level. We further introduce two attention mechanisms, knowledge attention and cross-lingual attention, to select informative data in comparable sentences.

Our contributions can be concluded as follows:

- We proposed a novel method that jointly learns representations of not only cross-lingual words but also cross-lingual entities in a unified vector space, aiming to enhance the embedding quality from each other via complementary semantics.
- Our proposed model introduces distant supervision coupled with attention mechanisms to generate comparable data as cross-lingual supervision, which can benefit many cross-lingual analysis.
- We did qualitative analysis to have an intuitive impression of our embeddings, and quantitative analysis in three tasks: word translation, entity relatedness, and cross-lingual entity linking. Experiment results show that our method demonstrates significant improvements in all three tasks.

2 Related Work

Jointly representation learning of words and entities attracts much attention in the fields of Entity Linking (Zhang et al., 2017a; Cao et al., 2018), Relation Extraction (Weston et al., 2013b) and so on, yet little work focuses on cross-lingual settings. Inspiringly, we investigate the task of cross-lingual word embedding models (Ruder et al., 2017), and classify them into three groups according to parallel corpora used as supervisions: (i) methods requiring **parallel corpus with aligned words** as constraint for bilingual word embedding learning (Klementiev et al., 2012; Zou et al., 2013; Wu et al., 2014; Luong et al., 2015; Ammar et al., 2016; Soricut and Ding, 2016). (ii) methods using **parallel sentences** (i.e. translated sentence pairs) as the semantic composition of multi-lingual words (Gouws et al., 2015; Kociský et al., 2014; Hermann and Blunsom, 2014; Chandar et al., 2014; Shi et al., 2015; Mogadala and Rettinger, 2016). (iii) methods requiring **bilingual lexicon** to map words from one language into the other (Mikolov et al., 2013b; Faruqui and Dyer, 2014; Xiao and Guo, 2014).

The major weakness of these methods is the limited availability of parallel corpora. One remedy is to use existing multi-lingual resources (i.e. multi-lingual KB). Camacho-Collados et al. (2015) combines several KBs (Wikipedia, WordNet and BabelNet) and leverages multi-lingual synsets to learn word embeddings at sense level through an extra post-processing step. Artetxe et al. (2017) starts from a small bilingual lexicon and using a self-learning approach to induce the structural similarity of embedding spaces. Vulic and Moens (2015, 2016) collect comparable documents on same themes from multi-lingual Wikipedia, shuffle and merge them to build “pseudo bilingual documents” as training corpora. However, the quality of “pseudo bilingual documents” are difficult to control, resulting in poor performance in several cross-lingual tasks (Vulic and Moens, 2016).

Another remedy matches linguistic distribu-

---

[1] `https://en.wikipedia.org/wiki/Help:Interlanguage_links`

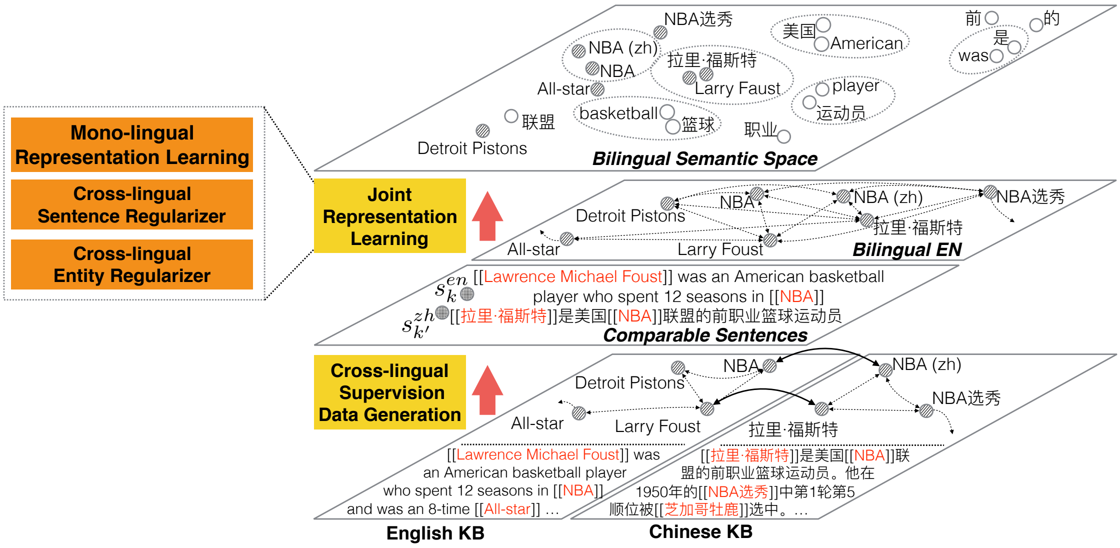

Figure 1: The overview framework of our method. The inputs and outputs of each step are listed in the three levels. Particularly, there are three main components of joint representation learning. Red texts with brackets are anchors, dashed lines denote entity relations, and solid lines are cross-lingual links.

tion via adversarial training (Barone, 2016; Zhang et al., 2017b; Lample et al., 2018), domain adaption (Cao et al., 2016). However, these methods suffer from the instability of training process and the high complexity. This either limits the scalability of vocabulary size or relies on a strong distribution assumption.

Inspired by Vulic and Moens (2016), we generate highly qualified **comparable sentences** via distant supervision, which is one of the most promising approaches to addressing the issue of sparse training data, and performs well in relation extraction (Lin et al., 2017a; Mintz et al., 2009; Zeng et al., 2015; Hoffmann et al., 2011; Surdeanu et al., 2012). Our comparable sentences may further benefit many other cross-lingual analysis, such as information retrieval (Dong et al., 2014).

3 Preliminaries and Framework

3.1 Preliminaries

Given a multi-lingual KB, we take (i) text corpus, (ii) entity and their relations, (iii) a set of anchors as inputs, and learn embeddings for each word and each entity in various languages. For clarity, we use English and Chinese as sample languages in the rest of the paper, and use superscript $y \in \{en, zh\}$ to denote language-specific parameters[2].

We use multi-lingual Wikipedia as KB including a set of entities $\mathcal{E}^y = \{e_i^y\}$ and their articles. We concatenate these articles together, and form text corpus $\mathcal{D}^y = \langle w_1^y, \dots, w_i^y, \dots, w_{|\mathcal{D}|}^y \rangle$. Hyper links in articles are denoted by Anchors $\mathcal{A}^y = \{\langle w_i^y, e_j^y \rangle\}$, which indicates that word w_i^y refers to entity e_j^y. $\mathcal{G}^y = (\mathcal{E}^y, \mathcal{R}^y)$ is the mono-lingual Entity Network (EN), where $\mathcal{R}^y = \{\langle e_i^y, e_j^y \rangle\}$ if there is a link between e_i^y, e_j^y. We use interlanguage links in Wikipedia as cross-lingual links $\mathcal{R}^{en-zh} = \{\langle e_i^{en}, e_{i'}^{zh} \rangle\}$, indicating e_i^{en}, $e_{i'}^{zh}$ refer to the same thing in English and Chinese. **Cross-lingual word and entity representation learning** is to map words and entities in different languages into a unified semantic space. Each word and entity obtain their embedding vectors[3] $\mathbf{w}_i^y$ and $\mathbf{e}_j^y$.

3.2 Framework

To alleviate the heavy burden of limited parallel corpora and additional translation efforts, we utilize existing multi-lingual resources to distantly supervise cross-lingual word and entity representation learning, so that the shared embedding space supports joint inference among KB and texts across languages. As shown in Figure 1, our framework has two steps: (1) **Cross-lingual Su-**

---

[2] We choose English and Chinese as example languages because they are top-ranked according to total number of speakers, the full list can be found in `https://en.wikipedia.org/wiki/Lists_of_languages_by_number_of_speakers`.

[3] For the cross-lingual linked entities sharing the same strings (e.g., *NBA* and *NBA (zh)*), which is an infrequent situation between languages, we use separated representations to keep training objective consistent and avoid confusion.

pervision Data Generation builds a bilingual entity network and generates comparable sentences based on cross-lingual links; (2) **Joint Representation Learning** learns cross-lingual word and entity embeddings using a unified objective function. Our assumption throughout the entire framework is as follows: *The more words/entities two contexts share, the more similar they are.*

As shown in Figure 1, we build a bilingual EN $\mathcal{G}^{en-zh}$ by using $\mathcal{G}^{en}, \mathcal{G}^{zh}$ and cross-lingual links $\mathcal{R}^{en-zh}$. Thus, entities in different languages shall be connected in a unified network to facilitate cross-lingual entity alignments. Meanwhile, from KB articles, we extract comparable sentences $\mathcal{S}^{en-zh} = \{\langle s_k^{en}, s_k^{zh} \rangle\}$ as high qualified parallel data to align similar words in different languages.

Based on generated cross-lingual data $\mathcal{G}^{en-zh}, \mathcal{S}^{en-zh}$ and mono-lingual data $\mathcal{D}^y$, $\mathcal{A}^y$, where $y \in \{en, zh\}$, we jointly learn cross-lingual word and entity embeddings through three components: (1) **Mono-lingual Representation Learning**, which learns mono-lingual word and entity embeddings for each language by modeling co-occurrence information through a variant of skip-gram model (Mikolov et al., 2013c). (2) **Cross-lingual Entity Regularizer**, which aligns entities that refer to the same thing in different languages by extending the mono-lingual model to bilingual EN. For example, entity *Foust* in English and entity 福斯特 *(Foust)* in Chinese are closely embedded in the semantic space because they share common neighbors in two languages, *All-star* and *NBA* 选秀 *(draft)*, etc.. (3) **Cross-lingual Sentence Regularizer**, which models cross-lingual co-occurrence at sentence level in order to learn translated words to have most similar embeddings. For example, English word *basketball* and the translated Chinese word 篮球 frequently co-occur in a pair of comparable sentences, therefore, their vector representations shall be close in the semantic space. The above components are trained jointly under a unified objective function.

4 Cross-lingual Supervision Data Generation

This section introduces how to build a bilingual entity network $\mathcal{G}^{en-zh}$ and comparable sentences $\mathcal{S}^{en-zh}$ from a multi-lingual KB.

4.1 Bilingual Entity Network Construction

Entities with cross-lingual links refer to the same thing, which implies they are equivalent across languages. Conventional knowledge representation methods only add edges between e_i^{en} and $e_{i'}^{zh}$ indicating a special "equivalent" relation (Zhu et al., 2017). Instead, we build $\mathcal{G}^{en-zh} = (\mathcal{E}^{en} \cup \mathcal{E}^{zh}, \mathcal{R}^{en} \cup \mathcal{R}^{zh} \cup \tilde{\mathcal{R}}^{en-zh})$ by enriching the neighbors of cross-lingual linked entities. That is, we add edges $\tilde{\mathcal{R}}^{en-zh}$ between two mono-lingual ENs by letting all neighbors of e_i^{en} be neighbors of $e_{i'}^{zh}$, and vice versa, if $\langle e_i^{en}, e_{i'}^{zh} \rangle \in \mathcal{R}^{en-zh}$.

$\mathcal{G}^{en-zh}$ extends $\mathcal{G}^{en}$ and $\mathcal{G}^{zh}$ to bilingual settings in a natural way. It not only keeps a consistent objective in mono-lingual ENs—entities, no matter in which language, will be embedded closely if share common neighbors—but also enhances each other with more neighbors in the foreign language.

Following the method in Zhu et al. (2017), there will be no edge between Chinese entity 福斯特 *(Foust)* and English entity *Pistons*, which implies a wrong fact that 福斯特 *(Foust)* does not belong to *Pistons*. Our method enriches the missing relation between entities 福斯特 *(Foust)* and 活塞队 *(Pistons)* in incomplete Chinese KB through corresponding English common neighbors, *Allstar*, *NBA*, etc., as illustrated in Figure 1.

4.2 Comparable Sentences Generation

To supervise the cross-lingual representation learning of words, we automatically generate comparable sentences as cross-lingual training data. Comparable sentences are not translated paired sentences, but sentences with the same topic in different languages. As shown in the middle layer (Figure 1), the pair of sentences are comparable sentences: (1) "*Lawrence Michael Foust was an American basketball player who spent 12 seasons in NBA*", (2) "拉里·福斯特 *(Lawrence Foust)* 是 *(was)* 美国 *(American) NBA* 联盟 *(association)* 的 *(of)* 前 *(former)* 职业 *(professional)* 篮球 *(basketball)* 运动员 *(player)*".

Inspired by the distant supervision technique in relation extraction, we assume that sentence s_k^{en} in Wikipedia articles of entity e_i^{en} explicitly or implicitly describes e_i^{en} (Yamada et al., 2017), and that s_k^{en} shall express a relation between e_i^{en} and e_j^{en} if another entity e_j^{en} is in s_k^{en}. Meanwhile, we find a comparable sentence $s_{k'}^{zh}$ in another language which satisfies $s_{k'}^{zh}$ containing $e_{j'}^{zh}$

in Wikipedia articles of Chinese entity $e_{i'}^{zh}$, where $\langle e_i^{en}, e_{i'}^{zh}\rangle, \langle e_j^{en}, e_{j'}^{zh}\rangle \in \mathcal{R}^{en-zh}$. As shown in Figure 1, the sentences in the second level are comparable due to the similar theme of the relation between entity *Foust* and *NBA*. To find this type of sentences, we search the anchors in the English aritcle and Chinese article of cross-lingual entity *Foust*, respectively, and extract the sentences including another crosslingual entity *NBA*. Comparable sentences can be regarded as cross-lingual contexts.

Unfortunately, comparable sentences suffer from two issues caused by distant supervision:

Wrong labelling. Take English as sample, there may be several sentences $s_{k,l}^{en}|_{l=1}^{L}$ containing the same entity e_j^{en} in the article of e_i^{en}. A straightforward solution is to concatenate them into a longer sentence s_k^{en}, but this increases the chance to include unrelated sentences.

Unbalanced information. Sometimes the pair of sentences convey unbalanced information, e.g., the English sentence in the middle layer (Figure 1) contains *Foust spent 12 seasons in NBA* while the comparable Chinese sentence not.

To address the issues, we propose knowledge attention and cross-lingual attention to filter out unrelated information at sentence level, and at word level respectively.

5 Joint Representation Learning

As shown in Figure 2, there are three components in learning cross-lingual word and entity representations, which are trained jointly. In this section, we will describe them in detail.

5.1 Mono-lingual Representation Learning

Following Yamada et al. (2016); Cao et al. (2017), we learn mono-lingual word/entity embeddings based on corpus $\mathcal{D}^y$, anchors $\mathcal{A}^y$ and entity network $\mathcal{G}^y$. Capturing the cooccurrence information among words and entities, these embeddings serve as the foundation and will be further extended to bilingual settings using the proposed cross-lingual regularizers, which will be detailed in the next section. Monolingually, we utilize a variant of Skip-gram model (Mikolov et al., 2013c) to predict the contexts given current word/entity:

$$\mathcal{L}_m = \sum_{y\in\{en,zh\}} \sum_{x_i^y \in \{\mathcal{D}^y,\mathcal{A}^y,\mathcal{G}^y\}} \log P(\mathcal{C}(x_i^y)|x_i^y)$$

where x_i^y is either a word or an entity, and $\mathcal{C}(x_i^y)$ denotes: (i) contextual words in a pre-defined window of x_i^y if $x_i^y \in \mathcal{D}^y$, (ii) neighbor entities that linked to x_i^y if $x_i^y \in \mathcal{G}^y$, (iii) contextual words of w_j^y if x_i^y is entity e_i^y in an anchor $\langle w_j^y, e_i^y\rangle \in \mathcal{A}^y$.

5.2 Cross-lingual Entity Regularizer

The bilingual EN $\mathcal{G}^{en-zh}$ merges entities in different languages into a unified network, resulting in the possibility of using the same objective as in mono-lingual ENs. Thus, we naturally extend mono-lingual function to cross-lingual settings:

$$\mathcal{L}_e = \sum_{e_i^y \in \{\mathcal{G}^{en-zh}\}} \log P(\mathcal{C}'(e_i^y)|e_i^y)$$

where $\mathcal{C}'(e_i^y)$ denotes cross-lingual contexts—neighbor entities in different languages that linked to e_i^y. Thus, by jointly learning mono-lingual representation with cross-lingual entity regularizer, words and entities share more common contexts, and will have similar embeddings. As shown in Figure 1, English entity *NBA* co-occurs with words *basketball* and *player* in texts, so they are embedded closely in the semantic space. Meanwhile, cross-lingual linked entities *NBA* and *NBA (zh)* have similar representations due to the most common neighbor entities, e.g., *Foust*.

5.3 Cross-lingual Sentence Regularizer

Comparable sentences provide cross-lingual co-occurrence of words, thus, we can use them to learn similar embeddings for the words that frequently co-occur by minimizing the Euclidean distance as follows:

$$\mathcal{L}_s = \sum_{\langle s_k^{en}, s_{k'}^{zh}\rangle \in S^{en-zh}} ||\mathbf{s}_k^{en} - \mathbf{s}_{k'}^{zh}||^2$$

where $\mathbf{s}_k^{en}, \mathbf{s}_{k'}^{zh}$ are sentence embeddings. Take English as sample language, we define it as the average sum of word vectors weighted by the combination of two types of attentions:

$$\mathbf{s}_k^{en} = \sum_{l=1}^{L} \psi(e_m^{en}, s_{k,l}^{en}) \sum_{w_i^{en} \in s_{k,l}^{en}} \psi'(w_i^{en}, w_j^{zh}) \mathbf{w}_i^{en}$$

where $s_{k,l}^{en}|_{l=1}^{L}$ are sentences containing the same entity (as mentioned in Section 4.2), and $\psi(e_m^{en}, s_{k,l}^{en})$ is knowledge attention that aims

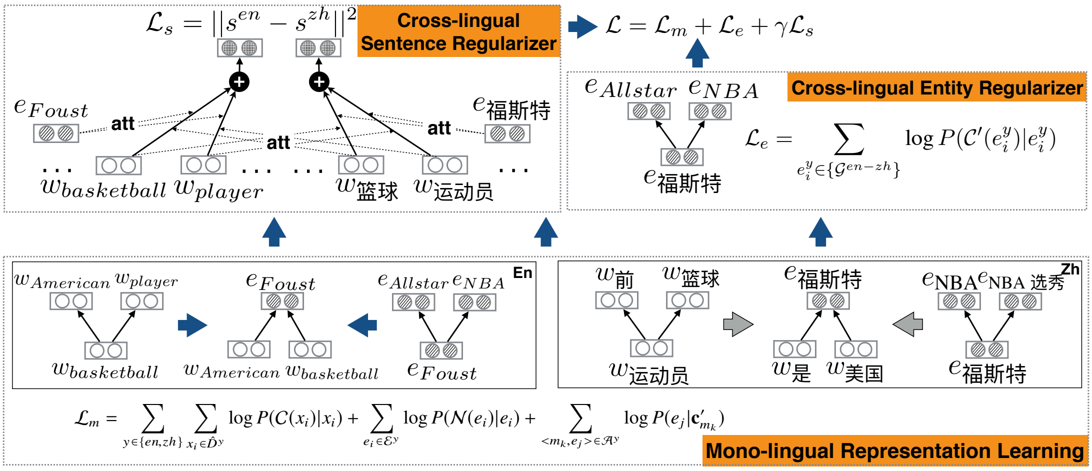

Figure 2: The nerual model for jointly representation learning.

at filtering out wrong labelling sentences, and $\psi'(w_i^{en}, w_j^{zh})$ is cross-lingual attention to deal with the unbalanced information through possible aligned words.

Next, we will introduce the two types of attentions in detail.

Knowledge Attention

Suppose that sentences $s_{k,l}^{en}|_{l=1}^{L}$ contain the same entities in articles of entity e_m^{en}, the wrong labelling errors increase, because some $s_{k,l}^{en}$ is almost irrelevant to e_m^{en}. Knowledge attention assigns smaller weights to wrong labelled sentences, and higher weights to related sentences. Thus, we define it proportional to the similarity between $s_{k,l}^{en}$ and e_m^{en}:

$$\psi(e_m^{en}, s_{k,l}^{en}) \propto sim(\mathbf{e}_m^{en}, \sum_{w_i^{en} \in s_{k,l}^{en}} \mathbf{w}_i^{en})$$

where sim is similarity measurement. We use cosine similarity in the presented work. Knowledge attention is normalized to satisfy $\sum_l^L \psi(e_m^{en}, s_{k,l}^{en}) = 1$.

Cross-lingual Attention

Inspired by self-attention mechanism (Lin et al., 2017b), we motivate cross-lingual attention focusing on potential information from comparable sentences themselves. The intuition is to find possible aligned words between languages, and filter out the words without alignments. We define it according to the maximum similarity computed by our cross-lingual word embeddings:

$$\psi'(w_i^{en}, w_j^{zh}) \propto \max_{w_i^{en} \in s_k^{en}, w_j^{zh} \in s_{k'}^{zh}} sim(\mathbf{w}_i^{en}, \mathbf{w}_j^{zh})$$

We set a threshold for discarding non-aligned words if $\psi'(w_i^{en}, w_j^{zh}) < \theta$, and make a normalization for selected words. We set $\theta = 0$ in experiments. Thus, unbalanced information is trimmed to the common meanings between s_k^{en} and $s_{k'}^{zh}$. For example (Figure 1), words *American*, *basketball*, *player* are selected due to their aligned Chinese words 美国, 篮球, 运动员, while *12 seasons* in s_k^{en} or 前 *(former)* in $s_{k'}^{zh}$ are discarded due to low attentions.

The reason of using such regularizer lies in two points: (1) the embeddings of cross-lingual aligned words become closer within the pair of comparable sentences, and meanwhile (2) the distance between their contexts is also minimized, which keeps the same way as used in mono-lingual word embeddings training—the words sharing more contexts have similar embeddings. In this way, our regularizer follows a similar assumption with (Gouws et al., 2015): *The more frequently two words occur in parallel/comparable sentence pairs, the closer their representation will be.*

5.4 Training

All above components are jointly trained using the overall objective function as follows:

$$\mathcal{L} = \mathcal{L}_m + \mathcal{L}_e + \gamma\mathcal{L}_s$$

where γ is a hyper-parameter to tune the effect of cross-lingual sentence regularizer, and set to 1 in experiments. We use Softmax as probability function, and negative sampling and SGD for efficient optimization (Mikolov et al., 2013a).

6 Experiments

In this section, we describe some qualitative analysis with nearest neighbors and quantitative experiments with the tasks of word translation, entity relatedness and cross-lingual entity linking to verify the quality of cross-lingual word embeddings, entity embeddings and the joint inference among them, respectively. The codes of our proposed model can be found in `https://github.com/TaoMiner/MultiLingualEmbedding`.

6.1 Experiment Settings

	Word		Entity	
	vocab (m)	token (b)	vocab (m)	token (b)
En	1.99	1.90	3.94	0.41
Zh	0.55	0.17	0.58	0.06
Es	0.70	0.48	0.70	0.04
Ja	0.46	0.45	0.88	0.08
It	0.67	0.40	1.09	0.12
Tr	0.33	0.05	0.22	0.01

Table 1: Multi-lingual KB Statistics.

We choose Wikipedia, the April 2017 dump, as multi-lingual KB and six popular languages for evaluation. The preprocessing consists of following steps: converting texts into lower cases, filtering out symbols and low frequency words and entities (less than 5), and tokenizing Chinese corpus using Jieba[4] and Japanese corpus using mecab[5]. The statistics is listed in Table 1. For brevity, we adopt two-letter abbreviations: 'En', 'Zh', 'Es', 'Ja', 'It' and 'Tr' for English, Chinese, Spanish, Japanese, Italian and Turkish, respectively. The token sub-column denotes the total number of word/entity in the entire training corpus, and we use 'm' to denote million and 'b' for billion.

For cross-lingual settings, we choose five language pairs to compare with state-of-the-art methods, whose statistics is listed in Table 2.

We trained our method using the suggested parameters in Skip-gram model (Mikolov et al., 2013c) and evaluate the embeddings shared by all tasks for fairly comparison. We set training epoch as 2 to ensure convergence, which costs nearly 20 hours on the server with 64 core CPU and 188GB memory. The embedding dimension is set to 200 and context window size is 5. For each positive example, we sample 5 negative examples.

	Cross-lingual Links (m)	Comparable Sentences(m)	Bilingual EN	
			$\mathcal{E}$(m)	$\mathcal{R}$(b)
Es-En	0.82	4.66	4.64	0.58
Zh-En	0.51	2.02	4.52	0.57
Ja-Zh	0.26	1.04	1.46	0.19
It-En	0.74	3.83	5.03	0.68
Tr-En	0.15	0.75	4.16	0.44

Table 2: Cross-lingual Data Statistics.

6.2 Qualitative Analysis

Translation words (Chinese)
篮球 (+), 篮球队 (basketball team), 湖人 (lakers), 男子篮球 (men's basketball), 湖人队 (the lakers), 国王队 (the Kings), 美式足球 (American football), 中锋 (center)
Nearest entities (Chinese)
NBA, 篮球 (Basketball) , 控球后卫 (Point guard), NBA 选秀 (draft), 香港男子甲一组男子篮球联赛 (Hong Kong men's top basketball league), 橄榄球 (American football), 东方篮球队 (Eastern basketball team)
Nearest words
nba, wnba, player, twyman, professional, pick, 76ers
Nearest entities
Professional sports, Varsity letter, Sports agent, All-America, Final four, All-star, College basketball

Table 3: Cross-lingual nearest words and entities of English word *basketball*.

We manually checked nearest neighbors to have a straightforward impression of the quality of our embeddings. The nearest neighbors of English word *basketball* is listed in Table 3.

As Table 3 shows, we find the correct translation ranked at top 1 (marked by +), and the listed words as well as English nearest words are all basketball related, indicating a higher quality of our cross-lingual word embeddings. Interestingly, we found that although all nearest entities are sports related, e.g., *NBA* or *Professional sports*, there is an obvious culture divergence between Chinese entities and English entities, such as *Hong Kong basketball league* v.s. *All-America*.

6.3 Word Translation

Following (Zhang et al., 2017b), we test our cross-lingual word embeddings on benchmark dataset including over 2,000 bilingual word pairs on average. The ground truth is obtained from Open

---

[4] `https://github.com/fxsjy/jieba`

[5] `http://taku910.github.io/mecab/`

	Es-En		It-En		Ja-Zh		Tr-En		Zh-En	
	large	small	large	small	large	small	large	small	large	small
TM	-	48.61	-	37.95	-	26.67	-	11.15	4.79	21.79
IA	-	60.41	-	46.52	-	36.35	-	17.11	7.08	32.29
Bilbowa	53	65.96	-	-	-	-	-	-	-	-
BWESG	48.88	66.38	36.84	51.29	30.93	37.80	21.36	35.59	20.57	29.17
Adversarial	-	71.97	-	58.60	-	43.02	-	17.18	7.92	43.31
Ours-noatt	68.34	77.1	62.22	65.90	37.00	42.30	**57.47**	**60.51**	**35.90**	42.80
Ours	**70.41**	**78.50**	**63.07**	**67.85**	**41.30**	**46.70**	54.40	59.31	35.66	**44.67**

Table 4: Word Translation.

Multilingual WordNet[6] or Google translation. We compare all methods using the same vocabulary, and analyze the vocabulary size's impact by setting a nearly 5k small scale and 50k large scale.

We choose several state-of-the-art methods as baseline, using different level of parallel data: (1) TM (Mikolov et al., 2013b), IA (Zhang et al., 2016) are pioneers and popular transformation based methods using **bilingual lexicon**. (2) Bilbowa (Gouws et al., 2015) is typical work using **parallel sentences** and performs quite well. (3) BWESG (Vulic and Moens, 2016) is similar to our method and achieves best performance in the literature of using **comparable data**. (4) Adversarial model (Zhang et al., 2017b) is the state-of-the-arts **without parallel data**. Besides, we remove attention from our method to investigate the impacts from attention mechanisms, marked with *Ours-noatt*.

For fair comparison, we report the results in original paper (Zhang et al., 2017b) except Bilbowa and BWESG, which didn't report their results on the same benchmark datasets. So, we carefully implement them using released codes on the same training corpus as ours with suggested parameters. Nevertheless, we do not have performance reports of Zh-En, It-En, Tr-En and Ja-Zh with Bilbowa due to the lack of parallel data used in the original paper. As shown in Table 4, we can see:

- Our proposed method significantly outperforms all the baseline methods with average gains of 21% and 9.1% on large and small vocabulary. This proves the high quality of our generated cross-lingual data and the effectiveness of our joint framework.
- The pair of languages have similar culture achieves better performance (Es-En, It-En, Tr-En, Ja-Zh) than that have different cultural origins, e.g., Zh-En.
- Languages with richer corpus have better translations because adequate training data helps to capture more accurate cross-lingual semantics (Es-En, It-En, Tr-En v.s. Ja-Zh).
- Our method has less performance reduction between small and large vocabulary than methods based on parallel word pairs, because we adopt a consistent objective function which aligns cross-lingual semantics, and simultaneously keeps their own monolingual semantics.
- Attention mechanisms further improve the performance, mainly because they help to select the most informative words and sentences, filtering out unrelated data.

6.4 Entity Relatedness

With respect to our entity embeddings, we have conducted experiments to evaluate English entity relatedness following (Ganea and Hofmann, 2017; Hoffart et al., 2011), in which the dataset contains 3,314 entities, and each entity has 91 candidate entities labeled with 1 or 0, indicating whether they are semantically related. Given an entity, we rank candidate entities according to their similarity based on our embeddings, and evaluate the ranking quality through two standard metrics: normalized discounted cumulative gain (NDCG) (Järvelin and Kekäläinen, 2002) and mean average precision (MAP) (Manning et al., 2008).

To give a comprehensive fair comparison, we choose several widely used and state-of-the-art methods as our baselines, and compare with the results in the original papers: (1) WLM (Milne and Witten, 2008), the popular semantic similarity measurement based on Wikipedia anchor links. (2) ALIGN (Yamada et al., 2016) and MPME (Cao et al., 2017), state-of-the-arts that jointly learn word and entity embeddings using mono-lingual EN. (3) Deep Joint (DJ) model (Ganea and Hofmann, 2017), deep neural model that achieves the

---

[6] `http://compling.hss.ntu.edu.sg/omw`

best performance of entity relatedness.

	NDCG			MAP
	@1	@5	@10	
WLM	.54	.52	.55	.48
ALIGN (d=500)	.59	.56	.59	.52
MPME	.61	.61	.65	.58
DJ (d=300)	.63	.61	.64	.58
Ours (Zh-En)	.62	.62	.66	.59
Ours (Es-En)	.61	.61	.65	.59
Ours (Tr-En)	.62	.62	.65	.59
Ours (It-En)	.61	.61	.65	.58
Ours-e (Es-En)	.62	.62	.67	.61
Ours-e (Es-En,epoch=5)	**.64**	**.64**	**.68**	**.62**

Table 5: Entity Relatedness.

Table 5 shows the results of baseline methods as well as our methods based on different languages. We also test the cases of our method without training cross-lingual words, marked as Ours-e. We can see our method outperforms all baseline methods by introducing cross-lingual information, and all bilingual ENs lead to similar results. Strangely, ALIGN and DJ with more embedding dimensions seemly fails to capture overall relatedness (performance reduction from top@1 to top@5). The best performance of Ours-e implies that training cross-lingual word slightly harms the performance of entity embeddings. We can introduce additional sense embeddings in future (Cao et al., 2017).

Although favorable improvements has been achieved by using our English entity embeddings, it shall be fewer than that of other languages, because resources of English are already quite rich, and even richer than many other languages, thus contributions from other languages will be less significant than vice versa. Due to the limitation of the publication, we neglect to report experiment results on the vice versa direction.

6.5 Cross-lingual Entity Linking

Entity linking, the task of identifing the language-specific reference entity for mentions in texts, raises the key challenges of comparing the relevance between entities and contextual words around the mentions (Cao et al., 2015; Nguyen et al., 2016). Recently, the surge of cross-lingual analysis pushes the entity linking task on cross-lingual settings (Ji et al., 2015). Therefore, we comprehensively measure our joint inference ability among words and entities using the tri-lingual EL benchmark dataset KBP2015, which consists of 944 documents and 38,831 mentions, and divides them into 444 and 500 documents for training and evaluation. Note that the main purpose of it is not to beat other EL models but to evaluate the quality of our embeddings, so we adopt a simple classifier GBRT (Gradient Boost Regression Tree) based method as in (Cao et al., 2017; Yamada et al., 2016), replace with our cross-lingual embeddings, and filter out mentions that are out of our vocabulary.

	English	Spanish	Chinese
Top system	73.7	80.4	83.1
Second system	66.2	71.5	78.1
Ours	73.9	79.1	81.3

Table 6: Tri-lingual Entity Linking.

Table 6 shows the top 1 linking accuracy (%). We can see our method performs much better than the second ranked system, and is competitive with the top ranked system. Considering that the systems utilize additional translation tools (Ji et al., 2015), we conclude that our embeddings are high qualified for joint inference among entities and words in different languages.

7 Conclusions

In this paper, we propose a novel method to jointly learn cross-lingual word and entity representations that enables effective inference among cross-lingual knowledge bases and texts. Instead of parallel data, we use distant supervision over multi-lingual KB to generate high quality comparable data as cross-lingual supervision signals for two types of regularizer. We introduce attention mechanism to further improve the training quality. A series of experiments on several tasks verify the effectiveness of our methods as well as the quality of cross-lingual word and entity embeddings.

In the future, we will enrich semantics of low-resourced languages by cross-lingual linking to rich-resourced languages, and extend more cross-lingual words and entities to multi-lingual settings.

8 Acknowledgments

The work is supported by NSFC key project (No. 61533018, U1736204, 61661146007), Ministry of Education and China Mobile Research Fund (No. 20181770250), and THUNUS NExT++ Co-Lab. Partial financial support from P3ML project funded by BMBF of Germany under grant number 01/S17064 is greatly acknowledged.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *ACL*.

Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. In *Rep4NLP@ACL*.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Nasari: a novel approach to a semantically-aware representation of items. In *HLT-NAACL*.

Hailong Cao, Tiejun Zhao, Shu Zhang, and Yao Meng. 2016. A distribution-based model to learn bilingual word embeddings. In *COLING*.

Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Neural collective entity linking. In *COLING*.

Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juan-Zi Li. 2017. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *ACL*.

Yixin Cao, Juanzi Li, Xiaofei Guo, Shuanhu Bai, Heng Ji, and Jie Tang. 2015. Name list only? target entity disambiguation in short texts. In *EMNLP*.

A. P. Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *NIPS*.

Meiping Dong, Yong Cheng, Yang Liu, Jia Xu, Maosong Sun, Tatsuya Izuha, and Jie Hao. 2014. Query lattice for translation retrieval. In *COLING*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *EACL*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *EMNLP*.

Stephan Gouws, Yoshua Bengio, and Gregory S. Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *ICML*.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2016. Joint representation learning of text and knowledge for knowledge graph completion. *CoRR*.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *ACL*.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *ACL*.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *ACL*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *TOIS*.

Heng Ji, Joel Nothman, Hoa Trang Dang, and Sydney Informatics Hub. 2016. Overview of tac-kbp2016 tri-lingual edl and its impact on end-to-end cold-start kbp. In *TAC*.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of tac-kbp2015 tri-lingual entity discovery and linking. In *TAC*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *COLING*.

Tomás Kociský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. In *ACL*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, Hervé Jégou, et al. 2018. Word translation without parallel data. *ICLR*.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017a. Neural relation extraction with multi-lingual attention. In *ACL*.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017b. A structured self-attentive sentence embedding. *CoRR*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *VS@HLT-NAACL*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to information retrieval.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *CoRR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *NIPS*.

David Milne and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *AAAI*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL/IJCNLP*.

Aditya Mogadala and Achim Rettinger. 2016. Bilingual word embeddings from parallel and non-parallel corpora for cross-language text classification. In *HLT-NAACL*.

Thien Huu Nguyen, Nicolas Fauceglia, Mariano Rodriguez Muro, Oktie Hassanzadeh, Alfio Massimiliano Gliozzo, and Mohammad Sadoghi. 2016. Joint learning of local and global features for entity linking via neural networks. In *COLING*.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. *arXiv preprint arXiv:1706.04902*.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning cross-lingual word embeddings via matrix co-factorization. In *ACL*.

Radu Soricut and Nan Ding. 2016. Multilingual word embeddings using multigraphs. *CoRR*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*.

Ivan Vulic and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *ACL*.

Ivan Vulic and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *JAIR*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *EMNLP*.

Zhigang Wang and Juan-Zi Li. 2016. Text-enhanced representation learning for knowledge graph. In *IJCAI*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013a. Connecting language and knowledge bases with embedding models for relation extraction. In *ACL*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013b. Connecting language and knowledge bases with embedding models for relation extraction. In *EMNLP*.

Haiyang Wu, Daxiang Dong, Xiaoguang Hu, Dianhai Yu, Wei He, Hua Wu, Haifeng Wang, and Ting Liu. 2014. Improve statistical machine translation with context-sensitive bilingual semantic embedding model. In *EMNLP*.

Jiawei Wu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Knowledge representation via joint learning of sequential text and knowledge graphs. *CoRR*.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *CoNLL*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning distributed representations of texts and entities from knowledge base. *TACL*.

Bishan Yang and Tom M. Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *ACL*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*.

Jing Zhang, Yixin Cao, Lei Hou, Juanzi Li, and Hai-Tao Zheng. 2017a. Xlink: an unsupervised bilingual entity linking system. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Adversarial training for unsupervised bilingual lexicon induction. In *ACL*.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi S. Jaakkola. 2016. Ten pairs to tag - multilingual pos tagging via coarse mapping between embeddings. In *HLT-NAACL*.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*.

Will Y. Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*.

Deep Pivot-Based Modeling for Cross-language Cross-domain Transfer with Minimal Guidance

Yftah Ziser and **Roi Reichart**
Faculty of Industrial Engineering and Management, Technion, IIT
syftah@campus.technion.ac.il, roiri@technion.ac.il

Abstract

While cross-domain and cross-language transfer have long been prominent topics in NLP research, their combination has hardly been explored. In this work we consider this problem, and propose a framework that builds on pivot-based learning, structure-aware Deep Neural Networks (particularly LSTMs and CNNs) and bilingual word embeddings, with the goal of training a model on labeled data from one (language, domain) pair so that it can be effectively applied to another (language, domain) pair. We consider two setups, differing with respect to the unlabeled data available for model training. In the *full* setup the model has access to unlabeled data from both pairs, while in the *lazy* setup, which is more realistic for truly resource-poor languages, unlabeled data is available for both domains but only for the source language. We design our model for the lazy setup so that for a given target domain, it can train once on the source language and then be applied to any target language without re-training. In experiments with nine English-German and nine English-French domain pairs our best model substantially outperforms previous models even when it is trained in the lazy setup and previous models are trained in the full setup.[1]

1 Introduction

The field of Natural Language Processing (NLP) has made impressive progress in the last two decades and text processing applications are now performed in a quality that was beyond imagination only a few years ago. With this success, it is only natural that researchers seek ways to apply NLP algorithms in as many languages and textual domains as possible. However, the success of NLP algorithms most often relies on the availability of non-trivial supervision such as corpora annotated with linguistic classes or structures, and for multilingual applications often also on parallel corpora. This resource bottleneck seriously challenges the world-wide accessibility of NLP technology.

To address this problem substantial efforts have been put into the development of cross-domain (*CD*, (Daumé III, 2007; Ben-David et al., 2010)) and cross-language (CL) transfer methods. For both areas, while a variety of methods have been developed for many tasks throughout the years (§ 2), with the prominence of deep neural networks (DNNs) the focus of modern methods is shifting towards learning data representations that can serve as a bridge across domains and languages.

For CD, this includes: (a) pre-DNN work ((Blitzer et al., 2006, 2007), known as *structural correspondence learning (SCL)*), that models the connections between *pivot features* – features that are frequent in the source and the target domains and are highly correlated with the task label in the source domain – and the other, non-pivot, features; (b) DNN work (Glorot et al., 2011; Chen et al., 2012) which employs compress-based noise reduction to learn cross-domain features; and recently also (c) works that combine the two approaches (Ziser and Reichart, 2017, 2018) (henceforth ZR17 and ZR18). For CL, the picture is similar: multilingual representations (usually word embeddings) are prominent in the transfer of NLP algorithms from one language to another (e.g. (Upadhyay et al., 2016)).

In this paper we aim to take CL and CD transfer a significant step forward and design methods that can adapt NLP algorithms simultaneously across languages and domains. We consider this research problem fundamental to our field as manually annotated resources are often scarce in many domains, even for languages that are consid-

---

[1] Our code is publicly available at https://github.com/yftah89/PBLM-Cross-language-Cross-domain

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 238–249
Brussels, Belgium, October 31 - November 4, 2018.

ered resource-rich. With effective cross-language cross-domain (CLCD) methods it is sufficient to have training resources in a single domain of one language in order to solve the task in any other (language, domain) pair.

As a first step, our focus in this work is on the task of sentiment classification that has been extensively researched in the CD literature. Surprisingly, even for this task we are aware of only one previous work that aims to perform CLCD learning (Fernández et al., 2016). However, this work does not employ modern DNN techniques and is substantially outperformed by our methods.

Our approach to CLCD learning is rooted in the family of methods that combine the power of both DNNs and pivot-based ideas, and is based on two principles. First, we build on the recent progress in learning multilingual word embeddings (Ruder et al., 2017). Such embeddings help close the lexical gap between languages as they map their different vocabularies to a shared vector space. Second, we follow (Prettenhofer and Stein, 2010, 2011; Fernández et al., 2016) and redefine the concept of pivot features for CLCD setups (§ 5). While these authors already employed this idea in order to design pivot-based methods in CL (Prettenhofer and Stein, 2010, 2011) and CLCD (Fernández et al., 2016) for text classification and sentiment analysis, their algorithms do not employ DNNs and multilingual embeddings. In this paper we show that it is the combination of bilingual word embeddings *(BEs)* and structure aware DNNs with the re-defined pivots that leads to high quality CLCD models.

Aiming to facilitate transfer to resource poor languages and domains, our methods rely on as little supervision as possible. Particularly, we explore two scenarios. In the first, *full CLCD* setup, models have access to manually annotated reviews from the source (language, domain) pair, and unannotated reviews from both the source and the target (language, domain) pairs. In the second, *lazy CLCD* setup, models have access only to source language reviews - annotated reviews from the source domain, and unannotated reviews from both the source and the target domains.

We consider the lazy setup to be the desired standard setup of CLCD learning for two reasons. First, in true resource-poor languages we expect it to be hard to find a sufficient number of reviews from many domains, even if they are unannotated (imagine for example trying to obtain 50K unlabeled spinner reviews in Swahili). Second, it allows a *train once, adapt everywhere* mode: instead of training a separate model for each target language, in this setup for each target domain only a single model is trained on the source language, and the target language is considered only at test time through BEs (§ 5). Notice that in order to allow the lazy setup, the BEs should be trained such that the source language embeddings have no knowledge about any particular target language. In § 5 we discuss the BEs we employ (Smith et al., 2017), which have this property.

We create CLCD variants of DNN- and pivot-based methods originally designed to learn effective representations for CD learning. To the best of our knowledge, there are three such methods, which employ two types of DNNs (§ 4): (a) AE-SCL and AE-SCL-SR (Ziser and Reichart, 2017) that integrate pivot-based ideas (SCL) with autoencoder-based (AE) noise reduction; and (b) pivot-based language modeling (PBLM, (Ziser and Reichart, 2018)) that combines pivot-based ideas with LSTMs for representation learning, and integrates this architecture with an LSTM or a CNN for task classification. In § 5 we discuss how to employ these methods for CLCD transfer where the lexical gap between languages is bridged by pivot translation and BEs, and show that PBLM allows for more effective transfer.

We address the task of binary sentiment classification and experiment with nine English-German and nine English-French domain pairs (§ 6, 7). Our PBLM-based models substantially outperform all previous models, even when the PBLM model is trained in the lazy setup and the previous models are trained in the full setup.

2 Previous Work

We briefly survey work on CL and CD learning and on multilingual word embeddings. We focus on aspects that are relevant to our work rather than on a comprehensive survey of the extensive previous work on these problems.

Cross-language transfer CL has been explored extensively in NLP. Example applications include POS tagging (Täckström et al., 2013), syntactic parsing (Guo et al., 2015; Ammar et al., 2016), text classification (Shi et al., 2010; Prettenhofer and Stein, 2010) and sentiment analysis (Wan, 2009; Zhou et al., 2016) among others.

Our work is mostly related to two works: (a) Cross-lingual SCL (CL-SCL, (Prettenhofer and Stein, 2010, 2011)); and (b) Distributional Correspondence Indexing (DCI, (Fernández et al., 2016)) – in both cases pivot features were redefined to support CL (in (a)) and CLCD (in (b)) with non-DNN models, in order to perform sentiment analysis. Below we show how we combine this idea with modern DNNs and BEs to substantially improve CLCD learning.

Cross-domain transfer In NLP, CD transfer (a.k.a domain adaptation) has been addressed for many tasks, including sentiment classification (Bollegala et al., 2011b), POS tagging (Schnabel and Schütze, 2013), syntactic parsing (Reichart and Rappoport, 2007; McClosky et al., 2010; Rush et al., 2012) and relation extraction (Jiang and Zhai, 2007; Bollegala et al., 2011a), if to name a handful of examples.

Several approaches to CD transfer have been proposed in the ML literature, including instance reweighting (Huang et al., 2007; Mansour et al., 2009), sub-sampling from both domains (Chen et al., 2011) and learning joint target and source feature representations. Representation learning, the latter, has become prominent in the DNN era, and is the approach we take here. As noted in § 1 we adopt CD models that integrate pivot-based learning with DNNs to perform CLCD.

Multilingual word embeddings Multilingual word embeddings learning is an active field of research. For example, Ruder et al. (2017) compare 49 papers that have addressed the problem since 2011. Such embeddings are of importance as they provide means of bridging the lexical gap between languages, which supports CL transfer.

Surveying this extensive literature is well beyond our scope. Since our focus is on performing CLCD with minimal supervision, we quote Ruder et al. (2017) that categorize multilingual embedding methods with respect to two criteria on the data they require for their training: (a) type of alignment (word, sentence or document); and (b) comparability (parallel data: exact translation, vs. comparable data: data that is only similar). The BEs we use in our work are those of Smith et al. (2017) that require several thousands translated words as a supervision signal. That is, except from BEs induced using comparable word alignment signals – words aligned through indirect signals such as related images or through comparability of their features (e.g. POS tags) – the BEs we employ belong to the class of the most minimal supervision. In addition, as noted in § 1, in order to allow the lazy CLCD setup, we would like BEs where the source language embeddings are induced with no knowledge of the target language, and we indeed choose such BEs (§ 5).

3 Task Definition

The task we address is cross-language cross-domain (CLCD) learning. Formally, we are given a set of labeled examples from language L_s and domain D_s (denoted as the pair (L_s, D_s)). Our goal is to train an algorithm that will be able to correctly label examples from language L_t and domain D_t (L_t, D_t). The same label set, T, is used across the participating source and target domains and languages.

The setup we consider is similar in spirit to the setup known as unsupervised domain adaptation (e.g. (Blitzer et al., 2007; Ziser and Reichart, 2017, 2018)). When taking the representation learning approach to CLCD learning, the training pipeline usually consists of two steps. In the first step, the representation learning model is trained on unlabeled data from the source and target languages and domains, with the goal of generating a joint representation for the source and the target. Below we describe the unlabeled data in the full and the lazy CLCD setups. In the second step, a classifier for the supervised task is trained on the (L_s, D_s) labeled data. To facilitate language and domain transfer, every example that is fed to the task classifier in this second step is first represented by the representation model that was trained with unlabeled data at the first step. This is true both when the task classifier is trained and at test time when it is applied to data from (L_t, D_t).

We consider two setups which differ with respect to the unlabeled examples available for the representation learning model. In the full CLCD setup, the training algorithm has access to unlabeled examples from both (L_s, D_s) and (L_t, D_t). Since for truly resource poor languages it may be challenging to find a sufficient number of unlabeled examples from (L_t, D_t), we also consider the lazy setup where the training algorithm has access to unlabeled examples from (L_s, D_s) and (L_s, D_t) – that is, target domain unlabeled examples are available only in the source language.

4 Preliminaries

In this paper we aim to adapt CD models that integrate the power of DNNs and of pivot-based learning so that they can be applied to CLCD learning. In this section we hence briefly describe the works in this line. We start with the concept of domain adaptation using pivot-based methods, continue with works that are based on autoencoders and end with works that are based on sequence modeling with LSTMs.

Pivot based domain adaptation This approach was proposed by Blitzer et al. (2006, 2007), through their SCL method. Its main idea is to divide the shared feature space of the source and the target domains to a set of pivot features that are: (a) frequent in both domains; and (b) have a strong correlation with the task label in the source domain labeled data. The features which do not comply with at least one of these criteria form a complementary set of non-pivot features.

In SCL, after the original feature set is divided into the pivot and non-pivot subsets, this division is utilized in order to map the original feature space of both domains into a shared, low-dimensional, real-valued feature space. To do so, a binary classifier is defined for each of the pivot features. This classifier takes the non-pivot features of an input example as its representation, and is trained on the unlabeled data from both the source and the target domains, to predict whether its associated pivot feature appears in the example or not. Note that no human annotation is required for the training of these classifiers, the supervision signal is in the unlabeled data. The matrix whose columns are the weight vectors of the classifiers is post-processed with singular value decomposition (SVD) and the derived matrix maps feature vectors from the original space to the new.

Since the presentation of SCL, pivot-based cross-domain learning has been researched extensively (e.g. (Pan et al., 2010; Gouws et al., 2012; Bollegala et al., 2015; Yu and Jiang, 2016; Yang et al., 2017)).

4.1 Autoencoder Based Methods

An autoencoder (AE) is comprised of an encoder function e and a decoder function d, and its output is a reconstruction of its input x: $r(x) = d(e(x))$. The model is trained to minimize a loss between x and $r(x)$. Over the last decade AEs have become prominent in CD learning with methods such as Stacked Denoising Autoencoders (SDA, (Vincent et al., 2008; Glorot et al., 2011) and marginalized SDA (MSDA, (Chen et al., 2012)) outperforming earlier state-of-the-art methods that were based on the concept of pivots but did not employ DNNs (Blitzer et al., 2006, 2007). A survey of AE-based models in CD learning can be found in ZR17.

ZR17 combined AEs and pivot-based modeling for CD learning. Their basic model (AE-SCL) is a feed-forward NN where the non-pivot features of the input example are encoded into a hidden representation that is then decoded into the pivot features of the example. Their advanced model (AE-SCL-SR) is identical in structure but its reconstruction matrix is fixed and consists of pre-trained embeddings of the pivot features, so that input examples with similar pivots are biased to have similar hidden representations. Since no CL learning was attempted in that work, the pre-trained embeddings used in AE-SCL-SR are monolingual. Both models are illustrated in Figure 1.

After one of the above representation models is trained with unlabeled data from the source and target domains, it is employed when training the task (sentiment analysis) classifier and when applying this classifier to test data. ZR17 learned a standard linear classifier (logistic regression), and fed it with the hidden representation of AE-SCL or AE-SCL-SR. They demonstrated the superiority of their models (especially, AE-SCL-SR) over non-DNN pivot-based methods and a variety of AE-based methods that do not consider pivots.

4.2 LSTM Based Methods

ZR18 observed that AE-based representation learning models do not exploit the structure of their input examples. Obviously, this can negatively impact text classification tasks, such as sentiment analysis. They hence proposed a structure-aware representation learning model, named Pivot Based Language Modeling (PBLM, Figure 2a).

PBLM is an LSTM fed with the embeddings of the input example words. As is standard in the LSTM literature, it is possible to feed the model with 1-hot word vectors and multiply them by a (randomly initialized) embeddings matrix (as done by ZR18) or to feed the model with pre-trained embeddings. In this paper we consider both options, taking advantage of the second in order to feed the model with BEs.

In contrast to standard LSTM-based language

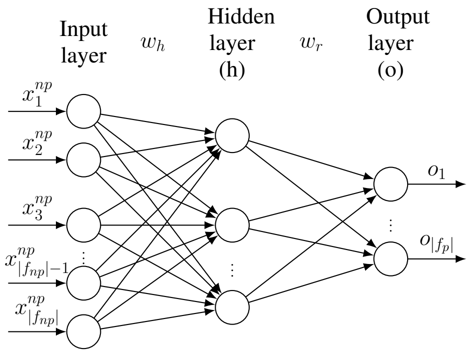

Figure 1: The AE-SCL and AE-SCL-SR models (figure imported from ZR17). x^{np} is a binary vector indicating whether each of the non-pivot features appears in the input example. x^{p} is a similar vector defined with respect to pivot features. o, the output vector of the model, provides the probability that each of the pivot features appears in the example. The loss function of both models is the cross-entropy loss between o and x^{p}. While in AE-SCL both the encoding matrix w^{h} and the reconstruction matrix w^{r} are optimized, in AE-SCL-SR w^{r} consists of pre-trained word embeddings.

models that predict at each point the most likely next input word, PBLM predicts the next input unigram or bigram if one of these is a pivot (if both are, it predicts the bigram) and NONE otherwise. Similarly to AE-SCL and AE-SCL-SR, PBLM is trained with unlabeled data from both the source and target domains.

consider the example in Figure 2a, provided in ZR18 for adaptation of a sentiment classifier between English book reviews and English reviews of kitchen appliances. PBLM learns the connection between witty - an adjective that is often used to describe books, but not kitchen appliances - and great - a common positive adjective in both domains, and hence a pivot feature. Another example in ZR18 for the same domain pair (see Figure 1 in their paper) is: ”I was at first very excited with my new Zyliss salad spinner - it is easy to spin and looks great”, from this sentence PBLM learns the connection between easy - an adjective that is often used to describe kitchen appliances, but not books - and great. That is, PBLM is able to learn the connection between witty and easy to facilitate adaptation between the domains.

PBLM can naturally feed a structure-aware task classifier. Particularly, in the PBLM-CNN architecture that we consider here (Figure 2b),[2] the PBLM's softmax layer (that computes the probabilities of each pivot to be the next unigram/bigram) is cut and a matrix whose columns are the PBLM's h_t vectors is fed to the CNN.

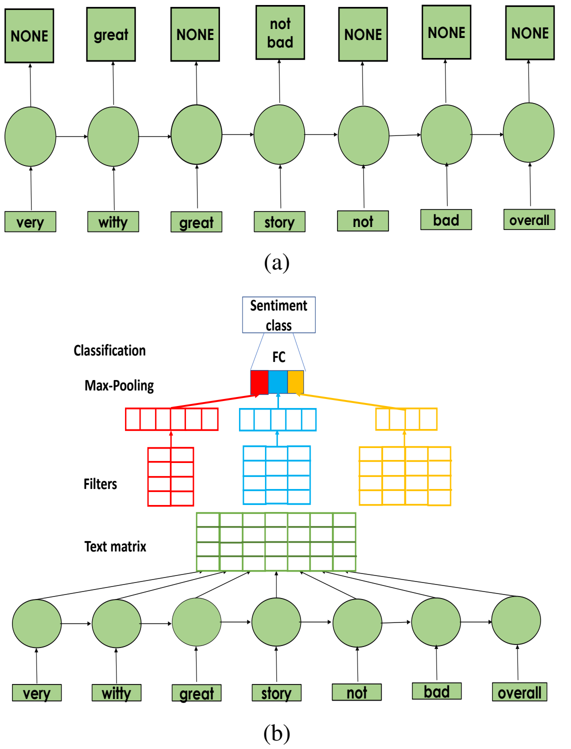

(a)

(b)

Figure 2: The PBLM model (figure imported form ZR18). (a) The PBLM representation learning model. (b) Adapting a classifier with PBLM: the PBLM-CNN model where PBLM representations are fed into a CNN task classifier.

ZR18 demonstrated the superiority of PBLM-CNN over AE-SCL, AE-SCL-SR and a variety of other previous models, emphasizing the importance of structure-awareness in CD transfer. We next discuss the adaptation of these models so that they can perform CLCD learning.

5 Cross-language Cross-domain Transfer

The models described in the previous section employ pivot-based learning (all models) and allow a convenient integration of BEs (AE-SCL-SR and PBLM). Below we discuss how we adapt these models so that they can perform CLCD learning.

---

[2] ZR18 also considered a PBLM-LSTM architecture where the PBLM representations feed an LSTM classifier. We focus on PBLM-CNN which demonstrated superior performance in 13 of 20 of their experimental setups.

Pivot translation We follow (Prettenhofer and Stein, 2010, 2011; Fernández et al., 2016) and redefine pivot features to be features that: (a) are frequent in (L_s, D_s) and that *their translation* is frequent in (L_t, D_t) ; and (b) are highly correlated with the task label in (L_s, D_s) . Note, that except for the translation requirement in (a) this is the classical definition of pivot features (§ 1).

Translated pivots are integrated into the models in a way that creates a shared cross-lingual output space. For both PBLM and the AE-based models a source language pivot feature and its translation are considered to be the same predicted class of the model. Consider, for example, a setup where we learn representations in order to adapt a classifier from (English, books) to (French, music). The pivot feature magnificent(English)/magnifique(French) will be considered the same PBLM prediction when trained on the unlabeled data from both (L_s, D_s) and (L_t, D_t). Similarly, in AE-SCL and AE-SCL-SR magnificent and magnifique will be assigned the same coordinate in the x^p (gold standard pivot indicators) and o (model output) vectors. In the lazy setup, where training is done with unlabeled data from (English, books) and (English, music) pivot translation is irrelevant as the representation learning model is trained only in the source language.

Note that when only pivot translation is used to make the CD methods address CLCD learning, the input space is not shared across languages. Instead, 1-hot vectors are used to encode the vocabularies of both languages, whose overlap is limited. This mismatch is somewhat reduced when training on unlabeled data from both (L_s, D_s) and (L_t, D_t). That is, we rely on the trained parameters of the models to align the input spaces when trained on unlabeled data from both (L_s, D_s) and (L_t, D_t).

In § 7 we show that this technique alone leads to improved CLCD results compared to existing methods. The lazy setup, however, is not supported by this technique, as training is not performed on unlabeled data from the target language. We next describe how to integrate BEs into our models, which provides a shared input layer that is crucial for both full and lazy CLCD.

Multilingual word embeddings Translated pivot features provide the models with a shared output layer. But can we use the same mechanism in order to map the input layers of the models into a shared cross-lingual space ?

Unfortunately, word-level translation does not seem like the right solution to this problem, due to two reasons. First, word-level translation is inherently ambiguous – it is very frequent that the set of senses associated with a given word in one language, is not identical to the set of senses associated with any other word in another given language. Moreover, large scale word-level translation may impose prohibitively high costs – either financial or in human time. Hence word-level translation is feasible mostly when dealing with a relatively small number of pivot features. The input layers of the models, consisting of words from the entire vocabulary (PBLM) or of non-pivot unigrams and bigrams (AE-SCL and AE-SCL-SR), require a cheaper and more stable mapping.

Our solution is hence based on BEs which embed words from the source and the target language in a shared vector space. As discussed in § 2 the BEs we use are those of Smith et al. (2017) that require several thousands of translated word pairs as a supervision signal, which reflects a low supervision level (Ruder et al., 2017). While bilingual word embedding models do not provide accurate word-level translation (to the level that such translation is possible), they do embed words from the two languages that have similar meaning with similar vectors, in terms of euclidean distance.

The BEs of Smith et al. (2017) also have the property required for our lazy setup: they are induced such that the source language embeddings have no knowledge of any particular target language. The embedding algorithm achieves that by learning two sets of monolingual embeddings and then aligning them with an SVD-based method.

Once we obtain the BEs, it is straightforward to integrate them into the PBLM model. We start by considering the full CLCD setup. When PBLM is applied to text from (L_s, D_s) – both when it is trained with unlabeled data (Figure 2a) and when it is used as part of the task classifier, when this classifier is trained with labeled data (Figure 2b) – the BEs of the source language words are fed into the model. Likewise, when PBLM is applied to text from (L_t, D_t) – both when it is trained with unlabeled data and when it is used as part of the task classifier when this classifier is applied to test data – it is fed with the bilingual representations of the target language words. In the lazy setup, the details are very similar except that PBLM is not trained with unlabeled data from (L_t, D_t), only with unlabeled data from (L_s, D_s) and (L_s, D_t).

Unfortunately, BEs do not provide a sufficient solution for the AE-based models. In AE-SCL the input layer consists of a non-pivots indicator vector, x^{np}, that cannot be replaced by embedding vectors in a straight forward manner. In AE-SCL-SR the input layer is identical to that of AE-SCL, but this model replaces the reconstruction matrix w^r with a matrix whose rows consist of pre-trained word embeddings of the pivot features. Hence, similarly to PBLM we can construct a w^r matrix with the source language BEs when this model is applied to source language data, and with target language BEs when this model is applied to target language data. This construction of w^r provides an additional shared cross-lingual layer, added to the translated pivot features of the output layer.

Consequently, an inherent limitation of the AE-based models when it comes to CLCD transfer, is that they cannot be employed in the lazy setup. The intersection of their input spaces when applied to the source and the target languages is limited to the vectors representing the shared vocabulary items (see above in this section). Hence, these models have to be trained with unlabeled data from both languages in order to align the input layers of the two languages with each other.

6 Experiments

Task and data [3] As in our most related previous work (Prettenhofer and Stein, 2010, 2011; Fernández et al., 2016) we experiment with the Websis-CLS-10 dataset (Prettenhofer and Stein, 2010) consisting of Amazon product reviews written in 4 languages (English, German, French and Japanese), from 3 product domains (Books (B), DVDs (D) and Music (M)). Due to our extensive experimental setup we leave Japanese for future.[4]

For each (language, domain) pair the dataset includes 2000 train and 2000 test documents, labeled as positive or negative, and between 9,358 to 50,000 unlabeled documents. As in the aforementioned related works, we consider English as the source language, as it is likely to have labeled documents from the largest number of domains.

Following ZR18 we also consider a more challenging setup where the English source domain consists of user airline (A) reviews (Nguyen, 2015). We use the dataset of ZR18, consisting of 1000 positive and 1000 negative reviews in the labeled set, and 39396 reviews as the unlabeled set.

We employ a 5-fold cross-validation protocol. In all folds 1600 (English, D_s) train-set examples are randomly selected for training and 400 for development. The German and French test sets are used in all folds. All sets contain the same number of positive and negative reviews. For each model we report averaged performance across the folds.

The BEs were downloaded from the author's github. More details are in the appendix.

Models and baselines Our main model is PBLM+BE that is trained in the full setup and employs both translated pivots for CL output alignment and BEs for CL input alignment (§ 5). We also experiment with PBLM+BE+Lazy: the same model employed in the lazy setup, and with PBLM: a model similar to PBLM+BE except that BEs are not employed.

We further experiment with AE-SCL that employs translated pivots for CL output alignment and AE-SCL-SR that does the same and also integrates BEs into its fixed reconstruction matrix. Following ZR17 and ZR18 the linear classifier we use is logistic regression. To compare to previous work, we implemented the CL-SCL and the DCI models, for which we use the cosine kernel that performs best in (Fernández et al., 2016).

To consider the power of BEs, we experiment with a classifier fed with the BEs of the input document's words. We consider both a CNN classifier (where the BEs are fed into the columns of the CNN input matrix) and logistic regression (where the embeddings of the document's words are averaged) and report results with CNN as they are superior. We denote this model with BE+CNN.

For reference, we also compare to a setup where $L_s = L_t$, and to a setup where $L_s = L_t$ and $D_s = D_t$. For these setups we report results with a linear classifier with unigram and bigram features, as it outperforms both a linear classifier and a CNN with BE features. The models are denoted with Linear-IL and Linear-ILID, respectively (IL stands for in-language and ID for in-domain).

Pivot features For all models we consider unigrams and bigrams as features. To divide these

---

[3] The URLs of the code (previous models and standard packages) and data we used, are in the appendix.

[4] We add an English domain to our experiments. Moreover, training the models we consider here is substantially more time consuming as we employ DNNs, as opposed to previous methods that use linear classifiers.

Product Review Domains (Websis-CLS-10,(Prettenhofer and Stein, 2010)), CLCD														
	English-German							English-French						
S-T	**D-B**	**M-B**	**B-D**	**M-D**	**B-M**	**D-M**	**All**	**D-B**	**M-B**	**B-D**	**M-D**	**B-M**	**D-M**	**All**
PBLM Models														
P+BE	**78.7**	**78.6**	**80.6**	**79.2**	**81.7**	**78.5**	**79.5**	**81.1**	**74.7**	**76.3**	**75.0**	**75.1**	**76.8**	**76.5**
PBLM	70.9	62.9	74.5	66.5	75.0	75.5	71.0	76.0	67.9	70.3	69.9	67.3	70.4	70.3
Lazy	74.8	74.0	75.1	72.8	73.3	73.7	73.9	74.2	73.1	75.3	74.4	74.1	72.4	73.9
Autoencoder+pivot Models														
A-S-SR	68.3	62.5	69.4	69.9	70.2	69	67.4	69.3	68.9	70.9	70.7	67	71.4	69.7
A-SCL	67.9	63.7	68.7	63.8	69.0	70.1	67.2	68.6	66.1	69.2	69.4	66.7	68.1	68.0
Pivot-based (no DNN) Models														
C-SCL	65.9	62.5	65.1	65.2	71.2	69.8	66.7	70.3	63.8	68.8	66.8	66.0	70.1	67.6
DCI	67.1	60.6	66.9	66.7	68.9	68.2	66.4	71.2	65.4	69.1	67.5	66.7	71.4	68.6
CLCD without CD Learning														
CNN	62.8	63.8	65.3	68.7	71.6	72.0	67.3	69.5	59.7	63.7	65.7	65.9	67.0	65.2

Airline (English, (Nguyen, 2015)) to Product Review Domains (German or French), CLCD								
	English-German				English-French			
Source-Target	**A-B**	**A-D**	**A-M**	**All**	**A-B**	**A-D**	**A-M**	**All**
PBLM Models								
P+BE	**67.9**	62.5	63.6	64.6	63.5	**66.9**	64.8	65.1
PBLM	60.9	59.6	60.1	60.2	60.9	61.9	58.9	60.5
Lazy	66.3	**65.0**	**66.6**	**66.0**	**65.7**	65.6	**69.0**	**66.8**
Autoencoder+pivot Models								
A-S-SR	55.8	57.5	60.8	58	55.8	52.9	56.3	55.7
A-SCL	55.9	56.2	58.2	56.8	55.8	52.9	56.4	55.0
Pivot-based (no DNN) Models								
C-SCL	56.6	52.6	53.7	54.3	52.7	54.5	53.1	53.4
DCI	55.9	52.1	54.5	54.1	53.1	53.7	53.9	53.5
CLCD without CD Learning								
CNN	59.4	61.2	61.3	60.6	57.9	55.3	56.2	56.5

Product Review Domains (Websis-CLS-10,(Prettenhofer and Stein, 2010)), Within Language														
	German-German							French-French						
S-T	**D-B**	**M-B**	**B-D**	**M-D**	**B-M**	**D-M**	**All**	**D-B**	**M-B**	**B-D**	**M-D**	**B-M**	**D-M**	**All**
In-language cross-domain learning (no CD technique is employed)														
IL	81.5	78.9	77.8	76.7	77.6	79.8	78.7	80.2	78.2	79.2	79.7	78.5	79.7	79.3
In-language, In-domain learning														
S-T	**B-B**	–	**D-D**	–	**M-M**	–	**All**	**B-B**	–	**D-D**	–	**M-M**	–	**All**
ILID	84.2	–	81.5	–	83.3	–	83	84.1	–	79.2	–	85.8	–	83

Table 1: Sentiment accuracy. Top: CLCD transfer in the product domains. Middle: CLCD transfer from the English airline domain to the French and German product domains. Bottom: within language learning for the target languages. "All" refers to the average over the setups. We shorten some abbreviations: P+BE stands for PBLM+BE, Lazy for PBLM+BE+Lazy, A-S-SR for AE-SCL-SR, A-SCL for AE-SCL, C-SCL for CL-SCL, CNN for BE+CNN, IL for Linear-IL and ILID for Linear-ILID.

features into pivots and non-pivots we follow (Blitzer et al., 2007; Ziser and Reichart, 2017, 2018). Pivots are translated with Google translate. Pivot features are frequent in the unlabeled data of both the source and the target (language, domain) pairs: we require them to appear at least 10 times in each. Among those frequent features we select the ones with the highest mutual information with the task (sentiment) label in the source (language, domain) labeled data. For non-pivot features we consider unigrams and bigrams that appear at least 10 times in one of the (language, domain) pairs.

Hyper-parameter tuning For all models we follow the tuning process described in the original papers. Details are in the appendix.

7 Results

Our results (Table 1) support the integration of structure-aware DNNs, translated pivots and BEs as advocated in this paper. Indeed, PBLM+BE which integrates all these factors and trained in the full setup is the best performing model in all 12 product setups (top table) and in 2 of 6 airline-product setups (middle table). PBLM+BE+lazy, the same model when trained in the lazy setup in which no target language unlabeled data is available for training, is the second best model in 9 of 12 product-product setups (in the other three setups only PBLM+BE and PBLM perform better) and is the best performing model in 4 of 6 airline-product setup and on average across these setups.

To better understand this last surprising result of the airline-product setups, we consider the pivot selection process (§ 6): (a) sort the source features by their mutual information with the source domain sentiment label; and (b) iterate over the pivots and exclude the ones whose translation frequency is not high enough in the target domain.

Let's examine the number of feature candidates that should be considered (in step (b)) from the list of criterion (a) in order to get 100 pivots. In product to product domain pairs: 182; In airline to product domain pairs: 304 (numbers are averaged across setups). In the lazy setup (where no pivot translation is performed) the corresponding numbers are: product to product domain pairs: 148; airline to product domain pairs: 173.

Hence, for domain pairs that involve airline and product, in the full setup many good pivots are *lost in translation* which affects the representation learning quality of PBLM+BE. While PBLM+BE+lazy does not get access to target language data, many more of its pivot features are preserved. We hypothesize that this can be one reason to the surprising superior performance of PBLM+BE+lazy when adapting from airline to product domains.

The success of PBLM+BE+lazy provides a particularly strong support to the validity of our approach, as this model lacks a major source of supervision available to the other CLCD models. As noted in § 1, we believe that the lazy setup is crucial for the future of CLCD learning.

Excluding BEs (PBLM) or changing the model to not generate a shared cross-lingual input layer (AE-SCL-SR that is also unaware of the review structure) results in substantial performance degradation. PBLM is better on average for all four CLCD setups, which emphasizes the importance of structure-awareness. Excluding both BEs and structure-awareness (AE) yields further degradation in most cases and on average. Yet, this degradation is minor (0.5% - 1.7% in the averages of the different setups), suggesting that the way AE-SCL-SR employs BEs, which is useful for CD transfer (ZR17), is less effective for CLCD.

CL-SCL and DCI, that employ pivot translation but neither DNNs nor BEs, lag behind the PBLM-based models and often also the AE-based models, although they outperform the latter in some cases. Likewise, BE+CNN, where BEs are employed but without any other CLCD learning technique, is also substantially outperformed by the PBLM-based models, but it does better than the AE-based models with the airline source domain.

Finally, comparison to the within-language models of the bottom table allows us to quantify the gap between current CLCD models and standard models that do not perform CD and/or CL transfer. The averaged differences between our best product-product model, PBLM-BE, and Linear-ILID are 3.5% (English-German) and 6.5% (English-French). When adapting from the airline domain the gap is much larger: averaged gaps of 17% and 16.2% from the best performing PBLM+BE-lazy, for English-German and English-French, respectively. This is not a surprise as ZR18 already demonstrated the challenging nature of within-language airline-product transfer. We consider our results to be encouraging, especially given the improvement over previous work, and the smaller gaps in the product-product setups.

8 Conclusions

We addressed the problem of CLCD transfer in sentiment analysis and proposed methods based on pivot-based learning, structure-aware DNNs and BEs. We considered full and lazy training, and designed a lazy model that, for a given target domain, can be trained with unlabeled data from the source language only and then be applied to any target language without re-training. Our models outperform previous models across 18 CLCD setups, even when ours are trained in the lazy setup and previous models are trained in the full setup.

In future work we wish to improve our results for large domain gaps and for more dissimilar languages, particularly in the important lazy setup. As our airline-product results indicate, increasing the domain gap harms our results, and we expect the same with more diverse language pairs.

Acknowledgements

We thank the anonymous reviewers and the members of the Technion NLP group for their useful comments. We also thank Ivan Vulić for guiding us in the world of multilingual word embeddings.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics* 4.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning* 79(1-2):151–175.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL (TACL)* 5:135–146.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proc. of ACL*.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2011a. Relation adaptation: learning to extract novel relations with minimum supervision. In *Proc. of IJCAI*.

Danushka Bollegala, David Weir, and John Carroll. 2011b. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *Proc. of ACL*.

Minmin Chen, Yixin Chen, and Kilian Q Weinberger. 2011. Automatic feature decomposition for single view co-training. In *Proc. of ICML*.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proc. of ICML*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*.

Alejandro Moreo Fernández, Andrea Esuli, and Fabrizio Sebastiani. 2016. Distributional correspondence indexing for cross-lingual and cross-domain sentiment classification. *Journal of artificial intelligence research* 55(1):131–163.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *In proc. of ICML*. pages 513–520.

Stephan Gouws, GJ Van Rooyen, MIH Medialab, and Yoshua Bengio. 2012. Learning structural correspondences across different linguistic domains with synchronous neural language models. In *Proc. of the xLite Workshop on Cross-Lingual Technologies, NIPS*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings ACL-IJCNLP)*.

Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. 2007. Correcting sample selection bias by unlabeled data. In *Proc. of NIPS*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proc. of ACL*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation with multiple sources. In *Proc. of NIPS*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Proc. of NAACL*.

Quang Nguyen. 2015. The airline review dataset. `https://github.com/quankiquanki/skytrax-reviews-dataset`. Scraped from `www.airlinequality.com`.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*. ACM, pages 751–760.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of ACL*.

Peter Prettenhofer and Benno Stein. 2011. Cross-lingual adaptation using structural correspondence learning. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3(1):13.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proc. of ACL*.

Sebastian Ruder, Ivan Vuli, and Anders Sgaard. 2017. A survey of cross-lingual word embedding models. In *arXiv preprint arXiv:1706.04902*.

Alexander M Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and pos tagging using inter-sentence consistency constraints. In *Proc. of EMNLP-CoNLL*.

Tobias Schnabel and Hinrich Schütze. 2013. Towards robust cross-domain domain adaptation for part-of-speech tagging. In *Proc. of IJCNLP*.

Lei Shi, Rada Mihalcea, and Mingjun Tian. 2010. Cross language text classification by model translation and semi-supervised learning. In *Proceedings of EMNLP*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *proceedings of ICLR*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics* 1:1–12.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of ACL*.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proc. of ICML*.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of ACL-IJCNLP*.

Wei Yang, Wei Lu, and Vincent Zheng. 2017. A simple regularization-based algorithm for learning cross-domain word embeddings. In *Proc. of EMNLP*.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proc. of EMNLP*.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Attention-based lstm network for cross-lingual sentiment classification. In *Proceedings of EMNLP*.

Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *Proc. of CoNLL*.

Yftah Ziser and Roi Reichart. 2018. Pivot based language modeling for improved neural domain adaptation. In *Proc. of NAACL-HLT*.

A Hyper-parameter Tuning

As promised in Section 6 of the main paper we detail here our hyper-parameter tuning process.

For all models, we tune the number of pivot features among $[100, 200, 300, 400, 500]$. For PBLM, the input embedding size (when no word embeddings are used) is tuned among $[128, 256]$, and the hidden representation dimension is selected from $[128, 256, 512]$. The size of the hidden layer of AE-SCL and AE-SCL-SR is set to 300.

The dimension of our bilingual embeddings is 300, as decided by (Smith et al., 2017). For all CNN models we use 256 filters of size $3 \times |embedding|$ and perform max pooling for each of the 256 vectors to generate a single 1×256 vector that is fed into the classification layer. In the SVD step of CL-SCL we tune the output dimension among $[50, 75, 100, 125, 150]$.

For AE-SCL and AE-SCL-SR, we follow ZR17 and represent each example fed into the sentiment classifier with its $w^h x^{np}$ vector. Unlike ZR17 we do not concatenate this representation with a bag of unigrams and bigrams representation of the example – due to the cross-lingual nature of our task. As in the original papers, the input features of AE-SCL, AE-SCL-SR, CL-SCL and DCI are word unigrams and bigrams.

All the algorithms in the paper that involve a CNN or a LSTM are trained with the ADAM algorithm (Kingma and Ba, 2015). For this algorithm we follow ZR18 and use the parameters described in the original ADAM article:

- Learning rate: $lr = 0.001$.
- Exponential decay rate for the 1st moment estimates: $\beta_1 = 0.9$.
- Exponential decay rate for the 2nd moment estimates: $\beta_2 = 0.999$.
- Fuzz factor: $\epsilon = 1e-08$.
- Learning rate decay over each update: $decay = 0.0$.

B Code and Data

Here we provide the URLs of the code and data we used in this paper:

- The Websis-CLS-10 dataset (Prettenhofer and Stein, 2010) `http://www.uni-weimar.de/en/media/chairs/webis/research/corpora/corpus-webis-cls-10/`
- Bilingual word embeddings (Smith et al., 2017): `https://github.com/Babylonpartners/fastText_multilingual`. The authors employed their method to monolingual fastText embeddings (Bojanowski et al., 2017) – the embeddings of 78 languages were aligned with the English embeddings.
- The bilingual embeddings are based on the fastText Facebook embeddings (Bojanowski et al., 2017): `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`

- Logistic regression classifier: `http://scikit-learn.org/stable/`

- PBLM: We use the code from the author's github: `https://github.com/yftah89/PBLM-Domain-Adaptation`

- AE-SCL and AE-SCL-SR: We use the code from the author's github: `https://github.com/yftah89/Neural-SCLDomain-Adaptation.`

- We reimplemented the CL-SCL (Prettenhofer and Stein, 2011) and the DCI (Fernández et al., 2016) models.

Multi-lingual Common Semantic Space Construction via Cluster-consistent Word Embedding

Lifu Huang[1], **Kyunghyun Cho**[2], **Boliang Zhang**[1], **Heng Ji**[1], **Kevin Knight**[3]
[1] Rensselaer Polytechnic Institute
`{huangl7,zhangb8,jih}@rpi.edu`
[2] New York University, CIFAR Global Scholar
`kyunghyun.cho@nyu.edu`
[3] Didi Labs and University of Southern California
`knight@isi.edu`

Abstract

We construct a multilingual common semantic space based on distributional semantics, where words from multiple languages are projected into a shared space via which all available resources and knowledge can be shared across multiple languages. Beyond word alignment, we introduce multiple cluster-level alignments and enforce the word clusters to be consistently distributed across multiple languages. We exploit three signals for clustering: (1) neighbor words in the monolingual word embedding space; (2) character-level information; and (3) linguistic properties (e.g., apposition, locative suffix) derived from linguistic structure knowledge bases available for thousands of languages. We introduce a new cluster-consistent correlational neural network to construct the common semantic space by aligning words as well as clusters. Intrinsic evaluation on monolingual and multilingual QVEC tasks shows our approach achieves significantly higher correlation with linguistic features which are extracted from manually crafted lexical resources than state-of-the-art multi-lingual embedding learning methods do. Using low-resource language name tagging as a case study for extrinsic evaluation, our approach achieves up to 14.6% absolute F-score gain over the state of the art on cross-lingual direct transfer. Our approach is also shown to be robust even when the size of bilingual dictionary is small.[1]

1 Introduction

More than 3,000 languages have electronic record, e.g., at least a portion of the Christian Bible had been translated into 2,508 different languages. However, the training data for mainstream natural language processing (NLP) tasks such as Information Extraction (IE) and Machine Translation (MT) is only available for dozens of dominant languages. In this paper we aim to construct a multilingual common semantic space where words in multiple languages are mapped into a distributed, language-agnostic semantic continuous space, so that resources and knowledge can be shared across languages.

Previous multilingual embedding methods align the semantic distributions of words from multiple languages within the common semantic space. Though several recent attempts (Artetxe et al., 2017, 2018; Conneau et al., 2017) have shown that it is possible to extract multilingual word embedding from a pair of potentially unaligned corpora in multiple languages, we claim that it is necessary to impose more constraints to preserve linguistic properties and facilitate downstream NLP tasks, such as cross-lingual IE, and MT. We find that words also can be clustered through explicit (e.g., sharing affixes of certain linguistic functions) or implicit clues (e.g., sharing neighbors from monolingual word embedding) and such clusters should also be consistent across multiple languages. To do so, we design a new algorithm, called cluster-consistent multilingual word embedding, that extracts multilingual word embedding vectors which preserve the natural clustering structures of words across multiple languages.

We propose to create clusters through three kinds of signals as follows, without any extra human annotation effort. Then we aggregate the embedding vectors of words in each cluster and ensure that the clusters (or the words therein) are consistent across multiple languages.

Neighbor based clustering and alignment. We build our common space based on correlational neural network (CorrNet) which is an extension of autoencoder framework by enabling cross-lingual reconstruction. In contrast to previous work (Chandar et al., 2016; Rajendran et al.,

---

[1] The resources and programs are available for research purpose: https://github.com/wilburOne/CommonSpace/

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 250–260
Brussels, Belgium, October 31 - November 4, 2018.

2015), we extend CorrNet to *neighbor-consistent correlation network* by using each word's neighbors (the nearest words within monolingual semantic space) to ensure that the cross-lingual mapping from and to the common semantic space is locally smooth. For instance, the neighboring words of *China* in English (*Japan*, *India* and *Taiwan*) should be close to the neighboring words of *Cina* in Italian (*Beijing*, *Korea*, *Japan*) in the common semantic space. In other words, we encourage the consistency of neighborhoods across multiple languages.

Character based clustering and alignment. Many related languages share very similar character set, and many words that refer to the same concept share similar compositional characters or patterns, e.g., *China* (English), *Kina* (Danish), and *Cina* (Italian).

Linguistic property based clustering and alignment. Many languages also share linguistic properties, e.g., apposition, conjunction, and plural suffix (English (*-s* / *-es*), Turkish (*-lar* / *-ler*), Somali (*-o*)). Linguists have created a wide variety of linguistic property knowledge bases, which are readily available for thousands of languages. For example, the CLDR (Unicode Common Locale Data Repository)[2] includes closed word classes and affixes indicating various linguistic properties. We propose to take advantage of these language-universal resources to create clusters, where the words within one cluster share the same linguistic property, and build alignment between clusters for common semantic space construction.

We evaluate our approach on monolingual and multilingual QVEC (Tsvetkov et al., 2015) tasks, which measure the quality of word embeddings based on the alignment of the embeddings to linguistic feature vectors extracted from manually crafted linguistic resources, as well as an extrinsic evaluation on name tagging for low-resource languages. Experiments demonstrate that our framework is effective at capturing linguistic properties and significantly outperforms state-of-the-art multi-lingual embedding learning methods.

2 Related Work

Multilingual word embeddings have advanced many multilingual NLP tasks, such as machine translation (Zou et al., 2013; Mikolov et al., 2013b; Madhyastha and España-Bonet, 2017), dependency parsing (Guo et al., 2015; Ammar et al., 2016a), and name tagging (Zhang et al., 2017a; Tsai and Roth, 2016; Zhang et al., 2018; Cheung et al.; Zhang et al., 2017b; Feng et al., 2017). Using bilingual aligned words, previous methods project multiple monolingual embeddings into a shared semantic space using linear mappings (Mikolov et al., 2013b; Rothe et al., 2016; Zhang et al., 2016; Baroni et al., 2015; Xing et al., 2015; Smith et al., 2017) or canonical correlation analysis (CCA) (Ammar et al., 2016b; Faruqui and Dyer, 2014; Lu et al., 2015). Compared with CCA, which only optimizes the correlation for each individual pair of languages, linear mapping based methods can jointly optimize all the languages in the common semantic space. We focus on learning linear mappings to construct the common semantic space and adopt correlational neural networks (CorrNet) (Chandar et al., 2016; Rajendran et al., 2015) as the basic model. In contrast to previous work which only exploited monolingual word semantics, we introduce multiple cluster-level alignments and design a new cluster consistent CorrNet to align both words and clusters.

Another branch of approaches for multilingual word embeddings are based on parallel or comparable data, such as parallel sentences (AP Chandar et al., 2014; Gouws et al., 2015; Luong et al., 2015; Hermann and Blunsom, 2014; Schwenk et al., 2017), phrase translations (Duong et al., 2016) and comparable documents (Vulic and Moens, 2015). Moreover, to reduce the need of bilingual alignment, several approaches have been designed to learn cross-lingual embeddings based on a small seed dictionary (Vulic and Korhonen, 2016; Zhang et al., 2016; Artetxe et al., 2017), or even with no supervision (Cao et al., 2016; Zhang et al., 2017d,c; Conneau et al., 2017; Artetxe et al., 2018). However, such methods are still limited to bilingual word embedding learning and remaining to be explored for common semantic space construction.

3 Approach

3.1 Overview

Figure 1 shows the overview of our neural architecture. We project all monolingual word embeddings into a common semantic space based on word-level as well as cluster-level alignments and learn the transformation functions. First, on

---

[2] cldr.unicode.org

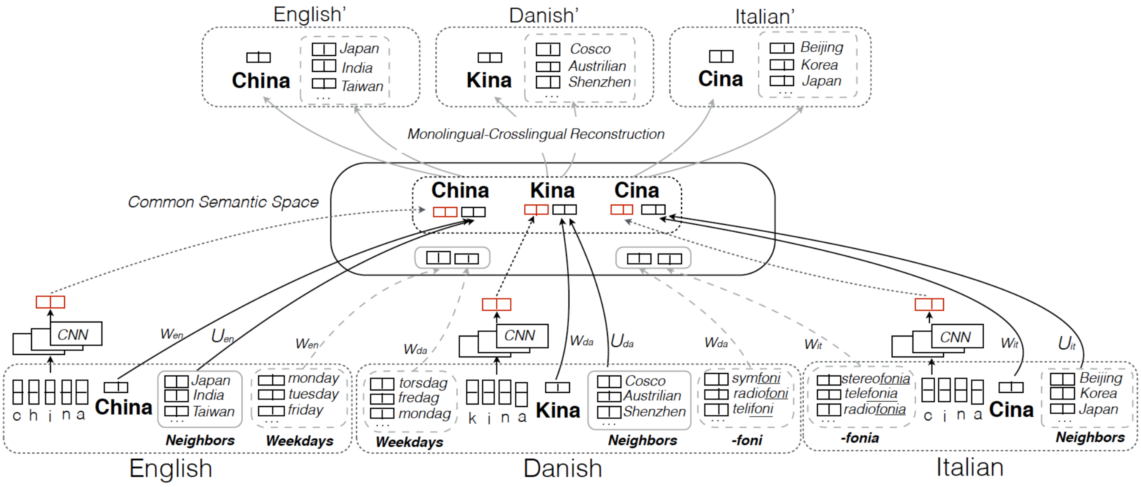

Figure 1: Architecture Overview. In each monolingual semantic space, the words within solid rectangle denote a neighbor based cluster and the words within dotted rectangle denote a linguistic property based cluster.

word-level, we build a neighborhood-consistent CorrNet to augment word representations with neighbor based clusters and align them in the common semantic space. In addition, we apply a language-independent convolutional neural networks to compose character-level word representation and concatenate it with word representation in the common semantic space. Finally, we construct clusters based on linguistic properties, including closed word classes and affixes, and align them in the common semantic space. We jointly optimize for all the alignments in the common semantic space for each pair of languages.

3.2 Basic Model

We briefly describe the basic model for learning the common semantic space: correlational neural networks (CorrNets) (Chandar et al., 2016; Rajendran et al., 2015). It combines the advantages of canonical correlation analysis (CCA) and autoencoder (AE).

Given the bilingual aligned word pairs between two languages l_1 and l_2, we first use their monolingual word embeddings to initialize each word with a vector and obtain $M_{l_1} \in \mathbb{R}^{|V_{l_1}|\times d_{l_1}}$ and $M_{l_2} \in \mathbb{R}^{|V_{l_2}|\times d_{l_2}}$, where V_{l_1} and V_{l_2} are the bilingual dictionary of l_1 and l_2. $V_{l_1}^i$ is the translation of $V_{l_2}^i$, and d_{l_1} and d_{l_2} are the vector dimensionalities. Then for each language we learn a linear projection function to project M_{l_1} and M_{l_2} into the common semantic space:

$$H_{l_1} = \sigma(M_{l_1} \cdot W_{l_1} + b_{l_1}) ,$$

$$H_{l_2} = \sigma(M_{l_2} \cdot W_{l_2} + b_{l_2}) ,$$

where $H_{l_1} \in \mathbb{R}^{|V_{l_1}|\times h}$ and $H_{l_2} \in \mathbb{R}^{|V_{l_2}|\times h}$ are the vector representations for V_{l_1} and V_{l_2} in the common semantic space respectively. h is the vector dimensionality in the shared semantic space. $W_{l_1} \in \mathbb{R}^{d_{l_1}\times h}$ and $W_{l_2} \in \mathbb{R}^{d_{l_2}\times h}$ are the transformation matrices, and b_{l_1} and b_{l_2} are the bias vectors. σ denotes Sigmoid function.

After we project the monolingual embeddings into the common semantic space, we further reconstruct M_{l_1} and M_{l_2} from H_{l_1} and H_{l_2} separately:

$$M'_{l_1} = \sigma(H_{l_1} \cdot W_{l_1}^{\top} + b'_{l_1}) ,$$

$$M^*_{l_1} = \sigma(H_{l_2} \cdot W_{l_1}^{\top} + b'_{l_1}) ,$$

$$M'_{l_2} = \sigma(H_{l_2} \cdot W_{l_2}^{\top} + b'_{l_2}) ,$$

$$M^*_{l_2} = \sigma(H_{l_1} \cdot W_{l_2}^{\top} + b'_{l_2}) ,$$

where b'_{l_1}, b'_{l_2} are the bias vectors. M'_{l_1} and M'_{l_2} are the monolingual reconstructions of M_{l_1} and M_{l_2} from the common space, and $M^*_{l_1}$ and $M^*_{l_2}$ are cross-lingual reconstructions. $W_{l_1}^{\top}$ and $W_{l_2}^{\top}$ are the transposes of W_{l_1} and W_{l_2} respectively.

To learn the common semantic representations, we minimize the distance between the aligned word vectors as well as the loss of monolingual and cross-lingual reconstruction:

$$O_W = \sum_{\{l_i,l_j\}\in A} L(M'_{l_i}, M_{l_i}) + L(M^*_{l_i}, M_{l_i})$$
$$+ L(M'_{l_j}, M_{l_j}) + L(M^*_{l_j}, M_{l_j}) + L(H_{l_i}, H_{l_j}) ,$$

where l denotes any language that we want to project into the common semantic space, A denotes all bilingual dictionaries, and L denotes a similarity metric. In our work, we use cosine similarity as the similarity metric.

3.3 Neighborhood-Consistent CorrNet

CorrNet can project multiple monolingual word embeddings into a common semantic space using bilingual word alignment. However, the same concepts may have different semantic bias in various languages. For example, the top five nearest words of the concept "*China*" are: (*Japan, India, Taiwan, Chinese, Asia*) in English, (*Cosco, Shenzhen, Australian, Shanghai, manufacturing*) in Danish, and (*Beijing, Korea, Japan, aluminum, copper*) in Italian respectively. In order to ensure the consistency of the neighborhoods within the common semantic space and make the cross-lingual mapping locally smooth, we propose to augment monolingual word representation with its top-N nearest neighboring words from the original monolingual semantic space.[3]

Given the monolingual embeddings of the bilingual aligned words for two languages l_1 and l_2, M_{l_1} and M_{l_2}, for each word, we extract the top-N nearest neighbors and construct the neighborhood clusters. Each cluster $t_l = \{w_1, w_2, ..., w_{|t_l|}\}$ in language l is represented by

$$c_{t_l} = \frac{1}{|t_l|} \sum_{w \in t_l} E_w \ ,$$

where E_w denotes the monolingual word embedding for w.

We obtain all the neighborhood cluster vector representations C_{l_1}, C_{l_2} for l_1 and l_2. We incorporate the neighborhood cluster information into the common semantic space when projecting monolingual embeddings:

$$\begin{aligned} H_{l_1} &= \sigma(M_{l_1} \cdot W_{l_1} + C_{l_1} \cdot U_{l_1} + b_{l_1}), \\ H_{l_2} &= \sigma(M_{l_2} \cdot W_{l_2} + C_{l_2} \cdot U_{l_2} + b_{l_2}), \end{aligned} \qquad (1)$$

Besides the monolingual and cross-lingual reconstructions for M_{l_1} and M_{l_2} in CorrNets, we also add monolingual and cross-lingual reconstructions for the neighborhood clusters:

$$C'_{l_1} = \sigma(H_{l_1} \cdot U_{l_1}^\top + b^*_{l_1}) \ ,$$

$$C^*_{l_1} = \sigma(H_{l_2} \cdot U_{l_1}^\top + b^*_{l_1}) \ ,$$

$$C'_{l_2} = \sigma(H_{l_2} \cdot U_{l_2}^\top + b^*_{l_2}) \ ,$$

$$C^*_{l_2} = \sigma(H_{l_1} \cdot U_{l_2}^\top + b^*_{l_2}) \ ,$$

In addition to optimizing the loss functions described in the Section 3.2, we further optimize the monolingual and cross-lingual reconstruction for neighborhood clusters:

$$\begin{aligned} O_N = \sum_{\{l_i, l_j\} \in A} & L(C'_{l_i}, C_{l_i}) + L(C^*_{l_i}, C_{l_i}) \\ & + L(C'_{l_j}, C_{l_j}) + L(C^*_{l_j}, C_{l_j}) \ , \end{aligned}$$

3.4 Character-Level Word Alignment

Bilingual word alignment is not always enough or available to induce a common semantic space, especially for low-resource languages. Although the words that refer to the same concept are not exactly the same in multiple languages, they often share a set of similar characters, especially in related languages written in the same script. For example, the same entity is spelled slightly differently in three languages: *Semsettin Gunaltay* in English, *Şemsettin Günaltay* in Turkish, and *Semsetin Ganoltey* in Somali. Beyond word-level alignment, we introduce character-level alignment by composing word representations from its compositional characters using convolutional neural networks (CNN). For each language, we adopt a language-independent CNN to generate character-level word representation.

Character Lookup Embeddings Let S_l be the character set for language l and $E_{S_l} \in \mathbb{R}^{|S_l| \times d}$ be the character lookup embeddings, where d is the dimensionality of each character embedding. Here, we use a simple yet effective method to induce character embeddings from word embeddings[4]. For each character c, we initialize its embedding by averaging the embeddings of all words which contain the character. The character embeddings will be further tuned by the model.

Character-Level CNN (Kim et al., 2016) The input layer is a sequence of characters of length k for each word. Each character is represented by a d-dimensional lookup embedding. Thus each input sequence is represented as a feature map of dimensionality $d \times k$.

We use the convolution layer to learn the representation for each sliding n-gram characters. We make p_i as the concatenated embeddings of n continuous columns from the input matrix, where n

---

[3] We set $N = 10$ in our experiments since it performed best on the intrinsic evaluation among {2, 5, 10, 20, 50}.

[4] This approach is proved to be better than random initialization of character embeddings.

is the filter width. We then apply the convolution weights $W \in \mathbb{R}^{d \times nd}$ to p_i with a biased vector $b \in \mathbb{R}^d$, i.e., $p_i^{'} = \tanh(W \cdot p_i + b)$. All n-gram representations $p_i^{'}$ are used to generate the word representation y by max-pooling.

In our experiments, we apply multiple filters with various widths to obtain the representation for word w_i^l. The final character-level word representation $\hat{w}_i^l$ is the concatenation of all word representations with varying filter widths.

Cross-lingual Mapping Given the bilingual aligned word pairs, we directly minimize the distance of the character-level word representations in the common semantic space by:

$$O_{char} = \sum_{\{l_i, l_j\} \in A} L(\hat{W}_{l_i}^{char}, \hat{W}_{l_j}^{char})$$

The final word representation of w_i^l in the common semantic space is the concatenation of character-level word presentation $\hat{w}_i^l$ and projected word representation h_i^l.

3.5 Linguistic Property Alignment

Linguists have made great efforts at building linguistic property knowledge bases for thousands of languages in the world. These knowledge bases include a large number of topological properties (phonological, lexical and grammatical) which we will use to build a high-level alignment between words across languages. We exploit the following resources:

- **CLDR** (Unicode Common Locale Data Repository)[5] which includes multilingual gazetteers for months, weekdays, cardinal and ordinal numbers;
- **Wiktionary**[6] which is a multilingual, web-based collaborative project to create an English content dictionary, includes word and prefix/suffix dictionaries for 1,247 languages;
- **Panlex**[7] database which contains 1.1 billion pairwise translations among 21 million expressions in about 10,000 language varieties.

We mainly exploit two types of linguistic properties to extract word clusters. The first type is language-independent closed word classes, such as colors, weekdays, and months. Table 1 shows some examples of the word clusters we automatically extracted from CLDR and Wiktionary for English. The second type of word clusters is generated based on morphological information, including affixes that indicate various linguistic properties. These properties tend to be consistent across many languages. For example, "-like" is a suffix denoting "similar to" in English, while in Danish "-agtig" performs the same function. Wiktionary and Panlex include the affix alignments between English and any other languages. We filtered out the many-to-many affix alignments and obtained hundreds of alignments between each language and English. For each affix, we derive a set of word pairs (*basic word*, *extended word with affix*) by first selecting all the word pairs where *basic word* $+$ *affix* $=$ *extended word*, then ranking all word pairs based on the cosine similarity of their monolingual word embedding. Finally we select the top ranked 20 word pairs to form the cluster for each affix.

Class Name	Words / Word Pairs
Colors	white, yellow, red, blue, green ...
Weekdays	monday, tuesday, friday, sunday ...
Months	january, february, march, april ...
numbers	one, two, three, four, five ...
pronouns	i, me, you, he, she, her, they ...
prepositions	of, in, on, for, from, about ...
conjunctions	but, and, so, or, when, while ...
clothes	hat, shirt, pants, skirt, socks ...
-like	(god, godlike), (bird, birdlike) ...
-able	(accept, acceptable), (adopt, adoptable) ...
micro-	(gram, microgram), (chip, microchip) ...
auto-	(maker, automaker), (gas, autogas) ...

Table 1: Examples of closed word classes and linguistic properties based clusters for English

We extract a set of word clusters from each language, and align the clusters based on their functions defined in CLDR, Wiktionary and Panlex. For each language l, each cluster $r_i^l \in R^l$ contains a set of words or word-pairs sharing the same function. We use the average operation to obtain an overall vector representation for each cluster M_l^R.[8] Then, we project the cluster-level vectors into the shared semantic space and minimize the distance between them:

$$H_{l_i}^R = \sigma(M_{l_i}^R \cdot W_{l_i} + b_{l_i}^R) \,,$$

$$H_{l_j}^R = \sigma(M_{l_j}^R \cdot W_{l_j} + b_{l_j}^R) \,,$$

---

[5] http://cldr.unicode.org/index/charts

[6] https://en.wiktionary.org

[7] http://panlex.org/

[8] For each word pair, we use the vector of the extend word minus the vector of the basic word as the vector representation of the word pair.

Parameter Name	Value
Monolingual Word Embedding Size	512
Multilingual Word Embedding Size	512
# of Filters in Convolution Layer	20
Filter Widths	1, 2, 3
Batch Size	500
Initial Learning Rate	0.5
Optimizer	Adadelta

Table 2: Hyper-parameters.

$$O_R = \sum_{\{l_i,l_j\}\in A} L(H^R_{l_i}, H^R_{l_j}) \,,$$

where W is the same as the W used in Section 3.3 for each language. We finally optimize the sum of the losses by finding the parameters $\theta = \{W_l, b_l, b_l^{'}, U_l, b_l^{*}, \text{CNN}_l, b_l^R\}$, where l denotes a specific language:

$$O_\theta = O_W + O_N + O_{char} + O_R$$

4 Experiments

4.1 Experiment Setup

Previous work (Ammar et al., 2016b; Duong et al., 2017) evaluated multilingual word embeddings on a series of intrinsic (e.g., monolingual and cross-lingual word similarity, word translation) and extrinsic (e.g., multilingual document classification, multilingual dependency parsing) evaluation tasks. In order to evaluate the quality of the multilingual embeddings, we use QVEC (Tsvetkov et al., 2015) tasks (details will be described in Section 4.2) as the intrinsic evaluation platform. In addition, to demonstrate the effectiveness of our common semantic space for knowledge transfer, especially for low-resource scenarios, we adopt the low-resource language name tagging task for extrinsic evaluation.

For fair comparison with state-of-the-art methods on building multi-lingual embeddings (Ammar et al., 2016b; Duong et al., 2017), we use the same monolingual data and bilingual dictionaries as in their work. We build multilingual word embeddings for 3 languages (*English, Italian, Danish*) and 12 languages (*Bulgarian, Czech, Danish, German, Greek, English, Spanish, Finnish, French, Hungarian, Italian, Swedish*) respectively. The monolingual data for each language is the combination of the Leipzig Corpora Collection[9] and Europarl.[10] The bilingual dictionaries are the same as those used in Ammar et al. (2016b).[11]

For each task, we evaluate the performance of our common semantic space in comparison with previously published multilingual word embeddings (MultiCluster, MultiCCA, MultiSkip, and MultiCross)[12]. MultiCluster (Ammar et al., 2016b) groups multilingual words into clusters based on bilingual dictionaries and forces all the words from various languages within one cluster share the same embedding. MultiCCA (Ammar et al., 2016b; Faruqui and Dyer, 2014) uses CCA to estimate linear projections for each pair of languages. MultiSkip is an extension of the multilingual skip-gram model (Luong et al., 2015), which requires parallel data. MultiCross is an approach to unify bilingual word embeddings into a shared semantic space using post hoc linear transformations (Duong et al., 2017).

Table 2 lists the hyper-parameters used in the experiments.

4.2 Intrinsic Evaluation: QVEC

In order to evaluate the quality of multilingual embeddings, we adopt QVEC (Tsvetkov et al., 2015) as the intrinsic evaluation measure. It evaluates the quality of word embeddings based on the alignment of distributional word vectors to linguistic feature vectors extracted from manually crafted lexical resources, e.g., SemCor (Miller et al., 1993). For each word, each dimension of its linguistic feature vector defines the probability of that word belongs to a supersense (e.g., NN.FOOD) which is summarized from WordNet (Fellbaum, 1998).

QVEC is computed as

$$\text{QVEC} = \max_{\sum_j a_{ij} \leq 1} \sum_{i=1}^{D} \sum_{j=1}^{P} r(x_i, s_j) \times a_{ij},$$

where $x \in \mathbb{R}^{D\times 1}$ denotes a distributional word vector and $s \in \mathbb{R}^{P\times 1}$ denotes a linguistic word vector. D and P denote the sizes of vectors respectively. $a_{ij} = 1$ iff x_i is aligned to s_j, otherwise $a_{ij} = 0$. $r(x_i, s_j)$ is the Pearson's correlation between x_i and s_j. QVEC-CCA (Ammar et al., 2016b) is extended from QVEC by using CCA to measure the correlation between the distributional matrix and the linguistic vector matrix, instead of cumulative dimension-wise correlation.

---

[9] http://wortschatz.uni-leipzig.de/en/download/

[10] http://www.statmt.org/europarl/index.html

[11] http://128.2.220.95/multilingual/data/

[12] For fair comparison, we use the development sets of the intrinsic evaluation tasks in Ammar et al. (2016b) to select the best model.

		3 Languages				12 Languages			
		Monolingual		Multilingual		Monolingual		Multilingual	
		QVEC	QVEC-CCA	QVEC	QVEC-CCA	QVEC	QVEC-CCA	QVEC	QVEC-CCA
	MultiCluster	10.8	63.6	9.1	**45.8**	10.4	62.7	9.3	44.5
	MultiCCA	10.8	63.8	8.5	43.9	10.8	63.9	8.5	43.7
	MultiSkip	7.8	57.3	7.3	36.2	8.4	59.1	7.2	36.5
	MultiCross	-	-	-	-	11.9	46.4	8.6	31.0
CorrNet	W	14.8	63.6	11.3	43.4	14.7	63.8	13.2	43.9
	W+N	15.8	64.2	**13.1**	43.9	15.8	64.6	14.0	44.8
	W+N+C	15.3	66.2	12.2	44.5	16.0	66.6	14.0	44.7
	W+N+L	**16.2**	66.1	**13.1**	44.8	16.1	64.7	13.8	44.9
	W+N+C+L	**16.2**	**67.3**	12.4	45.4	**16.3**	**66.7**	**14.1**	**45.2**

Table 3: QVEC and QVEC-CCA scores. W: word alignment. N: neighbor based clustering and alignment. C: character based clustering and alignment. L: linguistic property based clustering and alignment.

As shown in Table 3, our approaches outperform previous approaches in almost all cases[13]. Specifically, by augmenting word representation with neighboring words in the common semantic space as in Eq. (1), the performance for monolingual and multilingual QVEC and QVEC-CCA tasks is consistently improved. In addition, by aligning character-level compositional representations and linguistic property based clusters in the shared semantic space, the monolingual and multilingual representation quality is further improved.

4.3 Impact of Bilingual Dictionary Size

In order to show the impact of the size of bilingual lexicons, we use three languages as a case study, and gradually reduce the size of the lexicons for each pair of languages from 40,000 to 10,000 and further to 2,000, 1,000, 500 and 250. For following experiments, we use MultiCluster and MultiCCA as baselines [14]. Table 4 shows the results. We observe that both MultiCCA and CorrNet approaches are sensitive to the size of the bilingual lexicons. Our approach on the other hand can maintain high performance, even when the size of bilingual lexicons is reduced to 250. The performances of MultiCluster based on various sizes of bilingual dictionary are close because it jointly trains the embedding of multiple languages from scratch and by default takes advantage of identical strings among all the languages.

4.4 Low-Resource Name Tagging

We evaluate the quality of multilingual embeddings on a downstream task by using the embeddings as input features. Here, we use low-resource language name tagging as a target task, which aims to automatically identify and named entities from text and classify them into certain types, including Person (PER), Location (LOC), Organization (ORG), and Geo-Political Entities (GPE). We experiment with two sets of languages. The first set *Amh+Tig* consists of Amharic and Tigrinya. Both languages share the same Ge'ez script and descend from the proto-Semitic language family. The other set *Eng+Uig+Tur* consists of one high-resource language (English), one medium-resource language (Turkish) and one low-resource language (Uighur). It also consists of two distinct language scripts: English and Turkish use Latin script while Uighur uses Arabic script.

We use an LSTM-CRF architecture (Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016) for name tagging. It takes only word embedding as input and predict a tag for each word. Table 5 shows the statistics of training, development, and test sets for each language released by Linguistic Data Consortium (LDC).[15] For each language pair, we combine the bilingual aligned words extracted from Wiktionary and monolingual dictionaries based on identical strings.[16] We evaluate the quality from several aspects:

---

[13] We conduct paired t-test between CorrNet W+N+Ch+L and all the other models on 10 randomly sampled subsets. The differences are all statistically significant while all p-values are less than 0.05

[14] MultiSkip requires parallel corpora to train cross-lingual embeddings while the original implementation of MultiCross is not public.

[15] The annotations are from: Amh (LDC2016E87), Tig (LDC2017E27), Uig (LDC2016E70), Tur (LDC2014E115), Eng (Tjong Kim Sang and De Meulder, 2003). We combined these corpora with Wikipedia dump to train word embeddings with Word2Vec toolkit (Mikolov et al., 2013a).

[16] We extracted 23,781 pairs of words for Amh and Tig, 16,868 pairs for Eng and Tur, 3,353 pairs for Eng and Uig, and 3,563 pairs for Tur and Uig.

		QVEC		QVEC-CCA	
		Monolingual	Multilingual	Monolingual	Multilingual
40,000	multiCCA	10.8	8.5	63.8	43.9
	multiCluster	10.8	9.1	63.6	**45.8**
	CorrNet W	14.8	11.3	63.6	43.4
	CorrNet W+N+C+L	**16.2**	**12.4**	**67.3**	45.4
10,000	multiCCA	9.8	6.5	63.6	42.3
	multiCluster	10.6	9.5	62.4	44.7
	CorrNet W	14.8	11.3	63.4	43.0
	CorrNet W+N+C+L	**15.7**	**12.4**	**68.0**	**45.1**
2,000	multiCCA	9.9	6.2	63.6	40.9
	multiCluster	10.5	9.3	62.5	**44.8**
	CorrNet W	**14.5**	7.1	62.0	39.2
	CorrNet W+N+C+L	**14.5**	**11.4**	**68.0**	**44.8**
1,000	multiCCA	12.3	6.9	63.5	38.2
	multiCluster	10.5	9.3	62.5	**44.8**
	CorrNet W	**13.7**	9.4	63.0	40.0
	CorrNet W+N+C+L	13.6	**10.5**	**66.4**	43.0
500	multiCCA	12.3	5.5	63.5	36.0
	multiCluster	10.5	9.3	62.6	**44.7**
	CorrNet W	13.3	9.1	62.8	39.4
	CorrNet W+N+C+L	**13.4**	**9.5**	**66.2**	42.7
250	multiCCA	12.3	5.3	63.5	35.0
	multiCluser	10.5	9.2	62.7	**44.9**
	CorrNet W	13.8	9.3	62.5	39.3
	CorrNet W+N+C+L	**13.9**	**9.8**	**65.9**	42.2

Table 4: Results using bilingual lexicons with varying sizes (40,000, 10,000, 2,000, 1,000, 500, 250) and three languages. CorrNet W+N+C+L is the proposed approach with all the cluster types.

	Amh	Tig	Uig	Tur	Eng
Train	1,506	1,585	1,500	1,500	14,029
Dev	167	176	190	378	3,250
Test	711	440	476	470	3,453

Table 5: # of Sentences for name tagging

		Multilingual			
Train & Test	Mono-lingual	Multi-CCA	Multi-Cluster	CorrNet W	CorrNet W+N+C+L
Amh	52.0	50.6	53.4	52.4	**55.8**
Tig	78.2	78.4	76.4	77.9	**78.5**
Uig	63.3	59.6	60.1	61.9	**65.2**
Tur	62.9	47.7	54.0	59.3	**64.9**

Table 6: Comparison on Monolingual Embedding Quality: name tagging performance (F-score, %) using monolingual embedding and multilingual embeddings.

Monolingual embedding quality evaluation Table 6 shows the name tagging performance for each language using the original monolingual embeddings and multilingual embeddings. For all languages, the multilingual embeddings learned from our approach significantly outperform those learned from MultiCCA and MultiCluster, which shows the effectiveness of our approach. More importantly, the multilingual embeddings learned from our approach also outperform original monolingual embeddings, which demonstrates that by projecting multiple languages into one common space, the monolingual embedding quality can be further improved.

Cross-lingual direct transfer In this setting, we train a name tagger on one or two languages using multilingual embeddings and test it on a new language without any annotated data. Table 7 shows the performance. For most testing languages, our approach achieves better performance than MultiCCA and MultiCluster. The closer that the languages are, such as Amharic and Tigrinya, the better performance is achieved.

Cross-lingual mutual enhancement We finally show the improvement by adding more cross-lingual annotated data and also using multilingual embeddings in Table 8. The multilingual embeddings learned by our approach consistently outperforms MultiCCA and MultiCluster. Particularly, when there are not enough annotated examples, the performance could be improved by incorporating annotated examples from other languages. This is evident for Amharic, Tigrinya and Uighur. For Turkish, we notice that a larger extra anno-

Train	Test	Multi-CCA	Multi-Cluster	CorrNet W	CorrNet W+N+C+L
Amh	Tig	15.5	29.7	28.3	**33.7**
Tig	Amh	11.1	**24.7**	12.8	23.3
Eng	Uig	4.8	9.1	13.3	**15.5**
Tur	Uig	0.4	11.4	19.8	**25.0**
Eng+Tur	Uig	8.3	10.5	17.3	**23.3**
Eng	Tur	17.6	21.4	18.3	**22.4**
Uig	Tur	6.9	12.8	**13.2**	10.7
Eng+Uig	Tur	20.4	23.3	14.5	**27.0**

Table 7: Comparison on Cross-lingual Direct Transfer: name tagging performance (F-score, %) when the tagger was trained on 1-2 source languages and tested on a target language.

Train	Test	Multi-CCA	Multi-Cluster	CorrNet W	CorrNet W+N+C+L
Tig+Amh	Amh	52.9	54.7	52.1	**56.5**
Amh+Tig	Tig	78.0	76.9	78.1	**78.7**
Eng+Uig	Uig	64.8	62.2	65.1	**67.7**
Tur+Uig	Uig	63.6	58.9	63.6	**65.8**
Eng+Tur+Uig	Uig	65.8	64.8	64.6	**68.5**
Eng+Tur	Tur	50.3	56.1	59.3	**65.5**
Uig+Tur	Tur	51.4	52.7	57.8	**62.7**
Eng+Uig+Tur	Tur	48.1	54.3	56.6	**61.5**

Table 8: Comparison on Cross-lingual Mutual Enhancement: name tagging performance (F-score, %) when the training set for the tagger was enhanced by annotated examples in other languages.

tated set from other languages (e.g., Uig+Tur or Eng+Uig+Tur) doesn't necessarily result in improvement. This is partially due to the use of Arabic script in Uighur, which differs from Turkish and English. Thus we suggest to project closely related languages using the same script into the common semantic space.

We take Turkish name tagging as a case study to show the benefit of the common semantic space with extra English annotations. The monolingual model failed to identify *Belgrad´da* as a geopolitical entity (GPE) because it doesn't occur in Turkish training data. However, by adding English annotations, the tagger successfully tags it as a GPE since it's semantically close to *Belgrade* in the common semantic space according to their character level compositional embeddings and *Belgrade* is frequently tagged as GPE in English annotations. In another example, using Turkish annotations only, *Kraliyet Donanması´na* is mistakenly tagged as a GPE since it's following *da* and all entity mentions following *da* in Turkish annotations are annotated as GPE. After adding English annotations into training, it is correctly tagged as an ORG because *da* is well aligned with *in* in the common semantic space according to the linguistic property alignment between Turkish and English, and many entity mentions following *in* are annotated as ORG in English annotations.

5 Conclusions and Future Work

We construct a common semantic space for multiple languages based on a cluster-consistent correlational neural network. It combines word-level alignment and multi-level cluster alignment, including neighbor based clusters, character-level compositional word representations, and linguistic property based clusters induced from the readily available language-universal linguistic knowledge bases. Our approach achieved significantly higher performance than state-of-the-art multilingual embedding learning methods through both intrinsic and extrinsic evaluations. In the future, we will further extend our approach to multi-lingual multimedia common semantic space construction.

Acknowledgments

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract # FA8650-17-C-9116, and U.S. DARPA LORELEI Program # HR0011-15-C-0115. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. Kyunghyun Cho also thanks support by eBay, TenCent, NVIDIA and CIFAR.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016a. Many languages, one parser. *arXiv preprint arXiv:1602.01595*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016b. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Sarath AP Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C

Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proceedings of NIPS*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of ACL*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. *arXiv preprint arXiv:1805.06297*.

Marco Baroni, Angeliki Lazaridou, and Georgiana Dinu. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning.

Hailong Cao, Tiejun Zhao, Shu Zhang, and Yao Meng. 2016. A distribution-based model to learn bilingual word embeddings. In *Proceedings of COLING*.

Sarath Chandar, Mitesh M Khapra, Hugo Larochelle, and Balaraman Ravindran. 2016. Correlational neural networks. *Neural computation*.

Leon Cheung, Thamme Gowda, Ulf Hermjakob, Nelson Liu, Jonathan May, Alexandra Mayn, Nima Pourdamghani, Michael Pust, Kevin Knight, Nikolaos Malandrakis, et al. Elisa system description for lorehlt 2017.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. *arXiv preprint arXiv:1606.09403*.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2017. Multilingual training of crosslingual word embeddings. In *Proceedings of EMNLP*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EMNLP*.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Xiaocheng Feng, Lifu Huang, Bing Qin, Ying Lin, Heng Ji, and Ting Liu. 2017. Multi-level cross-lingual attentive neural architecture for low resource name tagging. *Tsinghua Science and Technology*, 22(6):633–645.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of ICML*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of ACL*.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributional semantics. In *Proceedings of ACL*.

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of HLT-NAACL*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of HLT-NAACL*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Pranava Swaroop Madhyastha and Cristina España-Bonet. 2017. Learning bilingual projections of embeddings for vocabulary expansion in machine translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

George A Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. 1993. A semantic concordance. In *Proceedings of HLT*.

Janarthanan Rajendran, Mitesh M Khapra, Sarath Chandar, and Balaraman Ravindran. 2015. Bridge correlational neural networks for multilingual multimodal representation learning. *arXiv preprint arXiv:1510.03519*.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. *arXiv preprint arXiv:1602.07572*.

Holger Schwenk, Ke Tran, Orhan Firat, and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. *arXiv preprint arXiv:1704.04154*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of NAACL-HLT*.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual wikification using multilingual embeddings. In *Proceedings of HLT-NAACL*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of EMNLP*.

Ivan Vulic and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of ACL*.

Ivan Vulic and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of ACL*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of HLT-NAACL*.

Boliang Zhang, Ying Lin, Xiaoman Pan, Di Lu, Jonathan May, Kevin Knight, and Heng Ji. 2018. Elisa-edl: A cross-lingual entity extraction, linking and localization system. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 41–45.

Boliang Zhang, Xiaoman Pan, Ying Lin, Tongtao Zhang, Kevin Blissett, Samia Kazemi, Spencer Whitehead, Lifu Huang, and Heng Ji. 2017a. Rpi blender tac-kbp2017 13 languages edl system.

Boliang Zhang, Xiaoman Pan, Ying Lin, Tongtao Zhang, Kevin Blissett, Samia Kazemi, Spencer Whitehead, Lifu Huang, and Heng Ji. 2017b. Rpi blender tac-kbp2017 13 languages edl system. In *TAC*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017c. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of ACL*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017d. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of EMNLP*.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi S Jaakkola. 2016. Ten pairs to tag-multilingual pos tagging via coarse mapping between embeddings. In *Proceedings of HLT-NAACL*.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*.

Unsupervised Multilingual Word Embeddings

Xilun Chen
Department of Computer Science
Cornell Unversity
Ithaca, NY, 14853, USA
xlchen@cs.cornell.edu

Claire Cardie
Department of Computer Science
Cornell Unversity
Ithaca, NY, 14853, USA
cardie@cs.cornell.edu

Abstract

Multilingual Word Embeddings (MWEs) represent words from multiple languages in a single distributional vector space. Unsupervised MWE (UMWE) methods acquire multilingual embeddings without cross-lingual supervision, which is a significant advantage over traditional supervised approaches and opens many new possibilities for low-resource languages. Prior art for learning UMWEs, however, merely relies on a number of independently trained Unsupervised Bilingual Word Embeddings (UBWEs) to obtain multilingual embeddings. These methods fail to leverage the interdependencies that exist among many languages. To address this shortcoming, we propose a fully unsupervised framework for learning MWEs[1] that directly exploits the relations between all language pairs. Our model substantially outperforms previous approaches in the experiments on multilingual word translation and cross-lingual word similarity. In addition, our model even beats supervised approaches trained with cross-lingual resources.

1 Introduction

Continuous distributional word representations (Turian et al., 2010) have become a common technique across a wide variety of NLP tasks. Recent research, moreover, proposes cross-lingual word representations (Klementiev et al., 2012; Mikolov et al., 2013a) that create a shared embedding space for words across *two* (Bilingual Word Embeddings, BWE) or *more* languages (Multilingual Word Embeddings, MWE). Words from different languages with similar meanings will be close to one another in this cross-lingual embedding space. These embeddings have been found beneficial for a number of cross-lingual and even monolingual NLP tasks (Faruqui and Dyer, 2014; Ammar et al., 2016).

The most common form of cross-lingual word representations is the BWE, which connects the lexical semantics of two languages. Traditionally for training BWEs, cross-lingual supervision is required, either in the form of parallel corpora (Klementiev et al., 2012; Zou et al., 2013), or in the form of bilingual lexica (Mikolov et al., 2013a; Xing et al., 2015). This makes learning BWEs for low-resource language pairs much more difficult. Fortunately, there are attempts to reduce the dependence on bilingual supervision by requiring a very small parallel lexicon such as identical character strings (Smith et al., 2017), or numerals (Artetxe et al., 2017). Furthermore, recent work proposes approaches to obtain unsupervised BWEs without relying on *any* bilingual resources (Zhang et al., 2017; Lample et al., 2018b).

In contrast to BWEs that only focus on a pair of languages, MWEs instead strive to leverage the interdependencies among multiple languages to learn a multilingual embedding space. MWEs are desirable when dealing with multiple languages simultaneously and have also been shown to improve the performance on some bilingual tasks thanks to its ability to acquire knowledge from other languages (Ammar et al., 2016; Duong et al., 2017). Similar to training BWEs, cross-lingual supervision is typically needed for training MWEs, and the prior art for obtaining *fully unsupervised* MWEs simply maps all the languages independently to the embedding space of a chosen target language[2] (usually English) (Lample et al., 2018b). There are downsides, however, when using a single fixed target language with no interaction between any of the two source languages. For instance, French and Italian are very similar, and the fact that each of them is individually converted to a less similar language, English for example, in

---

[1] Code: `https://github.com/ccsasuke/umwe`

[2] Henceforth, we refer to this method as BWE-Pivot as the target language serves as a pivot to connect other languages.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 261–270
Brussels, Belgium, October 31 - November 4, 2018.

order to produce a shared embedding space will inevitably degrade the quality of the MWEs.

For certain multilingual tasks such as translating between any pair of N given languages, another option for obtaining UMWEs exists. One can directly train UBWEs for each of such language pairs (referred to as BWE-Direct). This is seldom used in practice, since it requires training $O(N^2)$ BWE models as opposed to only $O(N)$ in BWE-Pivot, and is too expensive for most use cases. Moreover, this method still does not fully exploit the language interdependence. For example, when learning embeddings between French and Italian, BWE-Direct only utilizes information from the pair itself, but other Romance languages such as Spanish may also provide valuable information that could improve performance.

In this work, we propose a novel unsupervised algorithm to train MWEs using *only* monolingual corpora (or equivalently, monolingual word embeddings). Our method exploits the interdependencies between any two languages and maps all monolingual embeddings into a shared multilingual embedding space via a two-stage algorithm consisting of (i) Multilingual Adversarial Training (`MAT`) and (ii) Multilingual Pseudo-Supervised Refinement (`MPSR`). As shown by experimental results on multilingual word translation and cross-lingual word similarity, our model is as efficient as BWE-Pivot yet outperforms both BWE-Pivot and BWE-Direct despite the latter being much more expensive. In addition, our model achieves a higher overall performance than state-of-the-art *supervised* methods in these experiments.

2 Related Work

There is a plethora of literature on learning cross-lingual word representations, focusing either on a pair of languages, or multiple languages at the same time (Klementiev et al., 2012; Zou et al., 2013; Mikolov et al., 2013a; Gouws et al., 2015; Coulmance et al., 2015; Ammar et al., 2016; Duong et al., 2017, *inter alia*). One shortcoming of these methods is the dependence on cross-lingual supervision such as parallel corpora or bilingual lexica. Abundant research efforts have been made to alleviate such dependence (Vulić and Moens, 2015; Artetxe et al., 2017; Smith et al., 2017), but consider only the case of a single pair of languages (BWEs). Furthermore, fully unsupervised methods exist for learning BWEs (Zhang et al., 2017; Lample et al., 2018b; Artetxe et al., 2018a). For unsupervised MWEs, however, previous methods merely rely on a number of independent BWEs to separately map each language into the embedding space of a chosen target language (Smith et al., 2017; Lample et al., 2018b).

Adversarial Neural Networks have been successfully applied to various cross-lingual NLP tasks where annotated data is not available, such as cross-lingual text classification (Chen et al., 2016), unsupervised BWE induction (Zhang et al., 2017; Lample et al., 2018b) and unsupervised machine translation (Lample et al., 2018a; Artetxe et al., 2018b). These works, however, only consider the case of two languages, and our `MAT` method (§3.1) is a generalization to multiple languages.

Mikolov et al. (2013a) first propose to learn cross-lingual word representations by learning a linear mapping between the monolingual embedding spaces of a pair of languages. It has then been observed that enforcing the linear mapping to be orthogonal could significantly improve performance (Xing et al., 2015; Artetxe et al., 2016; Smith et al., 2017). These methods solve a linear equation called the orthogonal Procrustes problem for the optimal orthogonal linear mapping between two languages, given a set of word pairs as supervision. Artetxe et al. (2017) find that when using weak supervision (e.g. digits in both languages), applying this Procrustes process iteratively achieves higher performance. Lample et al. (2018b) adopt the iterative Procrustes method with pseudo-supervision in a fully unsupervised setting and also obtain good results. In the MWE task, however, the multilingual mappings no longer have a closed-form solution, and we hence propose the `MPSR` algorithm (§3.2) for learning multilingual embeddings using gradient-based optimization methods.

3 Model

In this work, our goal is to learn a single multilingual embedding space for N languages, without relying on *any* cross-lingual supervision. We assume that we have access to monolingual embeddings for each of the N languages, which can be obtained using unlabeled monolingual corpora (Mikolov et al., 2013b; Bojanowski et al., 2017). We now present our unsupervised MWE (UMWE) model that jointly maps the monolingual embeddings of all N languages into a single space by explicitly leveraging the interdependencies between arbitrary language pairs, but is computationally as efficient as learning $O(N)$ BWEs (instead of $O(N^2)$).

Denote the set of languages as $\mathscr{L}$ with $|\mathscr{L}| = N$. Suppose for each language $l \in \mathscr{L}$ with vocabulary $\mathcal{V}_l$, we have a set of d-dimensional monolingual word embeddings $\mathcal{E}_l$ of size $|\mathcal{V}_l| \times d$. Let $\mathcal{S}_l$ denote the monolingual embedding space for l, namely the distribution of the monolingual embeddings of l. If a set of embeddings $\mathcal{E}$ are in an embedding space $\mathcal{S}$, we write $\mathcal{E} \vdash \mathcal{S}$ (e.g. $\forall l : \mathcal{E}_l \vdash \mathcal{S}_l$). Our models learns a set of encoders $\mathcal{M}_l$, one for each language l, and the corresponding decoders $\mathcal{M}_l^{-1}$. The encoders map all $\mathcal{E}_l$ to a single target space $\mathcal{T}$: $\mathcal{M}_l(\mathcal{E}_l) \vdash \mathcal{T}$. On the other hand, a decoder $\mathcal{M}_l^{-1}$ maps an embedding in $\mathcal{T}$ back to $\mathcal{S}_l$.

Previous research (Mikolov et al., 2013a) shows that there is a strong linear correlation between the vector spaces of two languages, and that learning a complex non-linear neural mapping does not yield better results. Xing et al. (2015) further show that enforcing the linear mappings to be orthogonal matrices achieves higher performance. Therefore, we let our encoders $\mathcal{M}_l$ be orthogonal linear matrices, and the corresponding decoders can be obtained by simply taking the transpose: $\mathcal{M}_l^{-1} = \mathcal{M}_l^{\top}$. Thus, applying the encoder or decoder to an embedding vector is accomplished by multiplying the vector with the encoder/decoder matrix.

Another benefit of using linear encoders and decoders (also referred to as *mappings*) is that we can learn $N - 1$ mappings instead of N by choosing the target space $\mathcal{T}$ to be the embedding space of a specific language (denoted as the *target language*) without losing any expressiveness of the model. Given a MWE with an arbitrary $\mathcal{T}$, we can construct an equivalent one with only $N-1$ mappings by multiplying the encoders of each language $\mathcal{M}_l$ to the decoder of the chosen target language $\mathcal{M}_t^{\top}$:

$$\mathcal{M}_t' = \mathcal{M}_t^{\top}\mathcal{M}_t = I$$
$$\mathcal{M}_l'\mathcal{E}_l = (\mathcal{M}_t^{\top}\mathcal{M}_l)\mathcal{E}_l \vdash \mathcal{S}_t$$

where I is the identity matrix. The new MWE is isomorphic to the original one.

We now present the two major components of our approach, Multilingual Adversarial Training (§3.1) and Multilingual Pseudo-Supervised Refinement (§3.2).

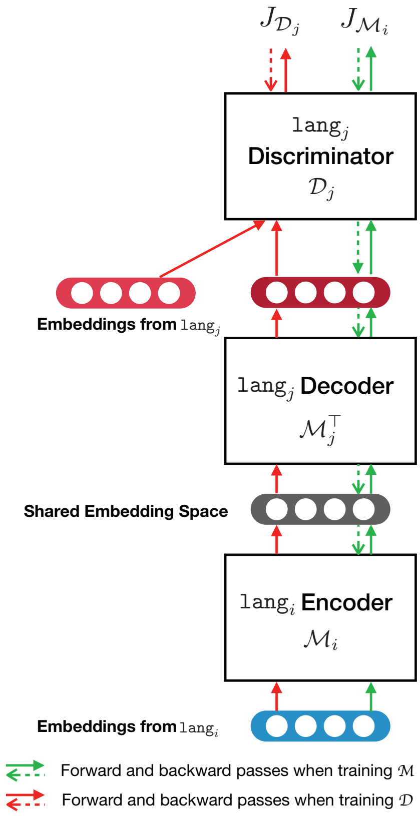

Figure 1: Multilingual Adversarial Training (Algorithm 1). `lang`$_i$ and `lang`$_j$ are two randomly selected languages at each training step. $J_{\mathcal{D}_j}$ and $J_{\mathcal{M}_i}$ are the objectives of $\mathcal{D}_j$ and $\mathcal{M}_i$, respectively (Eqn. 1 and 2).

3.1 Multilingual Adversarial Training

In this section, we introduce an adversarial training approach for learning multilingual embeddings without cross-lingual supervision. Adversarial Training is a powerful technique for minimizing the divergence between complex distributions that are otherwise difficult to directly model (Goodfellow et al., 2014). In the cross-lingual setting, it has been successfully applied to unsupervised cross-lingual text classification (Chen et al., 2016) and unsupervised bilingual word embedding learning (Zhang et al., 2017; Lample et al., 2018b). However, these methods only consider one pair of languages at a time, and do not fully exploit the cross-lingual relations in the multilingual setting.

Figure 1 shows our Multilingual Adversarial Training (`MAT`) model and the training procedure is described in Algorithm 1. Note that as explained in §3, the encoders and decoders adopted in practice are orthogonal linear mappings while the shared embedding space is chosen to be the same space as a selected target language.

Algorithm 1 Multilingual Adversarial Training

```
Require: Vocabulary $\mathcal{V}_i$ for each language lang$_i \in \mathscr{L}$. Hyperparameter $k \in \mathbb{N}$.
 1: repeat
 2:     ▷ $\mathcal{D}$ iterations
 3:     for diter $= 1$ to $k$ do
 4:         loss$_d = 0$
 5:         for all lang$_j \in \mathscr{L}$ do
 6:             Select at random lang$_i \in \mathscr{L}$
 7:             Sample a batch of words $x_i \sim \mathcal{V}_i$
 8:             Sample a batch of words $x_j \sim \mathcal{V}_j$
 9:             $\hat{x}_t = \mathcal{M}_i(x_i)$                ▷ encode to $\mathcal{T}$
10:             $\hat{x}_j = \mathcal{M}_j^\top(\hat{x}_t)$            ▷ decode to $\mathcal{S}_j$
11:             $y_j = \mathcal{D}_j(x_j)$                  ▷ real vectors
12:             $\hat{y}_j = \mathcal{D}_j(\hat{x}_j)$              ▷ converted vectors
13:             loss$_d$ += $L_d(1, y_j) + L_d(0, \hat{y}_j)$
14:         Update all $\mathcal{D}$ parameters to minimize loss$_d$
15:     ▷ $\mathcal{M}$ iteration
16:     loss $= 0$
17:     for all lang$_i \in \mathscr{L}$ do
18:         Select at random lang$_j \in \mathscr{L}$
19:         Sample a batch of words $x_i \sim \mathcal{V}_i$
20:         $\hat{x}_t = \mathcal{M}_i(x_i)$                    ▷ encode to $\mathcal{T}$
21:         $\hat{x}_j = \mathcal{M}_j^\top(\hat{x}_t)$                ▷ decode to $\mathcal{S}_j$
22:         $\hat{y}_j = \mathcal{D}_j(\hat{x}_j)$
23:         loss += $L_d(1, \hat{y}_j)$
24:     Update all $\mathcal{M}$ parameters to minimize loss
25:     orthogonalize($\mathcal{M}$)                      ▷ see §3.3
26: until convergence
```

In order to learn a multilingual embedding space without supervision, we employ a series of *language discriminators* $\mathcal{D}_l$, one for each language $l \in \mathscr{L}$. Each $\mathcal{D}_l$ is a binary classifier with a sigmoid layer on top, and is trained to identify how likely a given vector is from $\mathcal{S}_l$, the embedding space of language l. On the other hand, to train the mappings, we convert a vector from a random language `lang`$_i$ to another random language `lang`$_j$ (via the target space $\mathcal{T}$ first). The objective of the mappings is to confuse $\mathcal{D}_j$, the language discriminator for `lang`$_j$, so the mappings are updated in a way that $\mathcal{D}_j$ cannot differentiate the converted vectors from the real vectors in $\mathcal{S}_j$. This multilingual objective enables us to explicitly exploit the relations between all language pairs during training, leading to improved performance.

Formally, for any language `lang`$_j$, the objective that $\mathcal{D}_j$ is minimizing is:

$$J_{\mathcal{D}_j} = \mathop{\mathbb{E}}_{i\sim\mathscr{L}} \mathop{\mathbb{E}}_{\substack{x_i\sim\mathcal{S}_i\\ x_j\sim\mathcal{S}_j}} \Big[\!\Big[L_d\left(1, \mathcal{D}_j(x_j)\right) + L_d\left(0, \mathcal{D}_j(\mathcal{M}_j^\top \mathcal{M}_i x_i)\right)\Big]\!\Big] \tag{1}$$

where $L_d(y, \hat{y})$ is the loss function of $\mathcal{D}$, which is chosen as the *cross entropy loss* in practice. y is the language label with $y = 1$ indicates a real embedding from that language.

Furthermore, the objective of $\mathcal{M}_i$ for `lang`$_i$ is:

$$J_{\mathcal{M}_i} = \mathop{\mathbb{E}}_{j\sim\mathscr{L}} \mathop{\mathbb{E}}_{\substack{x_i\sim\mathcal{S}_i\\ x_j\sim\mathcal{S}_j}} L_d\left(1, \mathcal{D}_j(\mathcal{M}_j^\top \mathcal{M}_i x_i)\right) \tag{2}$$

where $\mathcal{M}_i$ strives to make $\mathcal{D}_j$ believe that a converted vector to `lang`$_j$ is instead real. This adversarial relation between $\mathcal{M}$ and $\mathcal{D}$ stimulates $\mathcal{M}$ to learn a shared multilingual embedding space by making the converted vectors look as authentic as possible so that $\mathcal{D}$ cannot predict whether a vector is a genuine embedding from a certain language or converted from another language via $\mathcal{M}$.

In addition, we allow `lang`$_i$ and `lang`$_j$ to be the same language in (1) and (2). In this case, we are encoding a language to $\mathcal{T}$ and back to itself, essentially forming an adversarial autoencoder (Makhzani et al., 2015), which is reported to improve the model performance (Zhang et al., 2017). Finally, on Line 5 and 17 in Algorithm 1, a for loop is used instead of random sampling. This is to ensure that in each step, every discriminator (or mapping) is getting updated at least once, so that we do not need to increase the number of training iterations when adding more languages. Computationally, when compared to the BWE-Pivot and BWE-Direct baselines, one step of `MAT` training costs similarly to N BWE training steps, and in practice we train `MAT` for the same number of iterations as training the baselines. Therefore, `MAT` training scales linearly with the number of languages similar to BWE-Pivot (instead of quadratically as in BWE-Direct).

3.2 Multilingual Pseudo-Supervised Refinement

Using `MAT`, we are able to obtain UMWEs with reasonable quality, but they do not yet achieve state-of-the-art performance. Previous research on learning unsupervised BWEs (Lample et al., 2018b) observes that the embeddings obtained from adversarial training do a good job aligning the frequent words between two languages, but performance degrades when considering the full vocabulary. They hence propose to use an iterative refinement method (Artetxe et al., 2017) to repeatedly refine the embeddings obtained from the adversarial training. The idea is that we can anchor on the more accurately predicted relations between frequent words to improve the mappings learned by adversarial training.

Algorithm 2 Multilingual Pseudo-Supervised Refinement

```
Require: A set of (pseudo-)supervised lexica of word pairs
    between each pair of languages $\texttt{Lex}(\texttt{lang}_i, \texttt{lang}_j)$.
repeat
    $\texttt{loss} = 0$
    for all $\texttt{lang}_i \in \mathscr{L}$ do
        Select at random $\texttt{lang}_j \in \mathscr{L}$
        Sample $(x_i, x_j) \sim \texttt{Lex}(\texttt{lang}_i, \texttt{lang}_j)$
        $t_i = \mathcal{M}_i(x_i)$                          ▷ encode $x_i$
        $t_j = \mathcal{M}_j(x_j)$                          ▷ encode $x_j$
        $\texttt{loss} \mathrel{+}= L_r(t_i, t_j)$          ▷ refinement loss
    Update all $\mathcal{M}$ parameters to minimize $\texttt{loss}$
    $\texttt{orthogonalize}(\mathcal{M})$                   ▷ see §3.3
until convergence
```

When learning MWEs, however, it is desirable to go beyond aligning each language with the target space individually, and instead utilize the relations between all languages as we did in `MAT`. Therefore, we in this section propose a generalization of the existing refinement methods to incorporate a multilingual objective.

In particular, `MAT` can produce an approximately aligned embedding space. As mentioned earlier, however, the training signals from $\mathcal{D}$ for rare words are noisier and may lead to worse performance. Thus, the idea of Multilingual Pseudo-Supervised Refinement (`MPSR`) is to induce a dictionary of highly confident word pairs for every language pair, used as pseudo supervision to improve the embeddings learned by `MAT`. For a specific language pair $(\texttt{lang}_i, \texttt{lang}_j)$, the pseudo-supervised lexicon $\texttt{Lex}(\texttt{lang}_i, \texttt{lang}_j)$ is constructed from *mutual nearest neighbors* between $\mathcal{M}_i\mathcal{E}_i$ and $\mathcal{M}_j\mathcal{E}_j$, among the most frequent $15k$ words of both languages.

With the constructed lexica, the `MPSR` objective is:

$$J_r = \mathop{\mathbb{E}}_{(i,j)\sim\mathscr{L}^2} \; \mathop{\mathbb{E}}_{(x_i,x_j)\sim\texttt{Lex}(i,j)} L_r(\mathcal{M}_i x_i, \mathcal{M}_j x_j) \tag{3}$$

where $L_r(x, \hat{x})$ is the loss function for `MPSR`, for which we use the *mean square loss*. The `MPSR` training is depicted in Algorithm 2.

Cross-Lingual Similarity Scaling (CSLS) When constructing the pseudo-supervised lexica, a distance metric between embeddings is needed to compute nearest neighbors. Standard distance metrics such as the Euclidean distance or cosine similarity, however, can lead to the *hubness* problem in high-dimensional spaces when used to calculate nearest neighbors (Radovanović et al., 2010; Dinu and Baroni, 2015). Namely, some words are very likely to be the nearest neighbors of many others (hubs), while others are not the nearest neighbor of any word. This problem is addressed in the literature by designing alternative distance metrics, such as the inverted softmax (Smith et al., 2017) or the CSLS (Lample et al., 2018b). In this work, we adopt the CSLS similarity as a drop-in replacement for cosine similarity whenever a distance metric is needed. The CSLS similarity (whose negation is a distance metric) is calculated as follows:

$$\begin{aligned} \mathrm{CSLS}(x, y) = {} & 2\cos(x, y) \\ & - \frac{1}{n} \sum_{y' \in N_Y(x)} \cos(x, y') \\ & - \frac{1}{n} \sum_{x' \in N_X(y)} \cos(x', y) \end{aligned} \tag{4}$$

where $N_Y(x)$ is the set of n nearest neighbors of x in the vector space that y comes from: $Y = \{y_1, ..., y_{|Y|}\}$, and vice versa for $N_X(y)$. In practice, we use $n = 10$.

3.3 Orthogonalization

As mentioned in §3, orthogonal linear mappings are the preferred choice when learning transformations between the embedding spaces of different languages (Xing et al., 2015; Smith et al., 2017). Therefore, we perform an orthogonalization update (Cisse et al., 2017) after each training step to ensure that our mappings $\mathcal{M}$ are (approximately) orthogonal:

$$\forall l : \mathcal{M}_l = (1 + \beta)\mathcal{M}_l - \beta \mathcal{M}_l \mathcal{M}_l^\top \mathcal{M}_l$$

where β is set to 0.001.

3.4 Unsupervised Multilingual Validation

In order to do model selection in the unsupervised setting, where no validation set can be used, a surrogate validation criterion is required that does not depend on bilingual data. Previous work shows promising results using such surrogate criteria for model validation in the bilingual case (Lample et al., 2018b), and we in this work adopt a variant adapted to our multilingual setting:

$$\begin{aligned} V(\mathcal{M}, \mathcal{E}) &= \mathop{\mathbb{E}}_{(i,j)\sim P_{ij}} \texttt{mean\_csls}(\mathcal{M}_j^\top \mathcal{M}_i \mathcal{E}_i, \mathcal{E}_j) \\ &= \sum_{i \neq j} p_{ij} \cdot \texttt{mean\_csls}(\mathcal{M}_j^\top \mathcal{M}_i \mathcal{E}_i, \mathcal{E}_j) \end{aligned}$$

where p_{ij} forms a probability simplex. In this work, we let all $p_{ij} = \frac{1}{N(N-1)}$ so that $V(\mathcal{M}, \mathcal{E})$ reduces to the macro average over all language pairs. Using different p_{ij} values can place varying weights on different language pairs, which might be desirable in certain scenarios.

The `mean_csls` function is an unsupervised bilingual validation criterion proposed by Lample et al. (2018b), which is the mean CSLS similarities between the most frequent $10k$ words and their translations (nearest neighbors).

4 Experiments

In this section, we present experimental results to demonstrate the effectiveness of our unsupervised MWE method on two benchmark tasks, the multilingual word translation task, and the SemEval-2017 cross-lingual word similarity task. We compare our `MAT+MPSR` method with state-of-the-art unsupervised and supervised approaches, and show that ours outperforms previous methods, supervised or not, on both tasks.

Pre-trained $300d$ fastText (monolingual) embeddings (Bojanowski et al., 2017) trained on the Wikipedia corpus are used for all systems that require monolingual word embeddings for learning cross-lingual embeddings.

4.1 Multilingual Word Translation

In this section, we consider the task of word translation between arbitrary pairs of a set of N languages. To this end, we use the recently released multilingual word translation dataset on six languages: English, French, German, Italian, Portuguese and Spanish (Lample et al., 2018b). For any pair of the six languages, a ground-truth bilingual dictionary is provided with a train-test split of 5000 and 1500 unique source words, respectively. The $5k$ training pairs are used in training supervised baseline methods, while all unsupervised methods do not rely on any cross-lingual resources. All systems are tested on the 1500 test word pairs for each pair of languages.

For comparison, we adopted a state-of-the-art unsupervised BWE method (Lample et al., 2018b) and generalize it for the multilingual setting using the two aforementioned approaches, namely BWE-Pivot and BWE-Direct, to produce unsupervised baseline MWE systems. English is chosen as the pivot language in BWE-Pivot. We further incorporate the supervised BWE-Direct (Sup-BWE-Direct) method as a baseline, where each BWE is trained on the $5k$ gold-standard word pairs via the orthogonal Procrustes process (Artetxe et al., 2017; Lample et al., 2018b).

Table 1 presents the evaluation results, wherein the numbers represent *precision@1*, namely how many times one of the correct translations of a source word is retrieved as the top candidate. All systems retrieve word translations using the CSLS similarity in the learned embedding space. Table 1a shows the detailed results for all 30 language pairs, while Table 1b summarizes the results in a number of ways. We first observe the training cost of all systems summarized in Table 1b. #BWEs indicates the training cost of a certain method measured by how many BWE models it is equivalent to train. BWE-Pivot needs to train $2(N-1)$ BWEs since a separate BWE is trained for each direction in a language pair for increased performance. BWE-Direct on the other hand, trains an individual BWE for all (again, directed) pairs, resulting a total of $N(N-1)$ BWEs. The supervised Sup-BWE-Direct method trains the same number of BWEs as BWE-Direct but is much faster in practice, for it does not require the unsupervised adversarial training stage. Finally, while our `MAT+MPSR` method does not train independent BWEs, as argued in §3.1, the training cost is roughly equivalent to training $N-1$ BWEs, which is corroborated by the real training time shown in Table 1b.

We can see in Table 1a that our `MAT+MPSR` method achieves the highest performance on all but 3 language pairs, compared against both the unsupervised and supervised approaches. When looking at the overall performance across all language pairs, BWE-Direct achieves a $+0.6\%$ performance gain over BWE-Pivot at the cost of being much slower to train. When supervision is available, Sup-BWE-Direct further improves another 0.4% over BWE-Direct. Our `MAT+MPSR` method, however, attains an impressive 1.3% improvement against Sup-BWE-Direct, despite the lack of cross-lingual supervision.

To provide a more in-depth examination of the results, we first consider the Romance language pairs, such as fr-es, fr-it, fr-pt, es-it, it-pt and their reverse directions. BWE-Pivot performs notably worse than BWE-Direct on these pairs, which validates our hypothesis that going through a less similar language (English) when translating between

	en-de	en-fr	en-es	en-it	en-pt	de-fr	de-es	de-it	de-pt	fr-es	fr-it	fr-pt	es-it	es-pt	it-pt
Supervised methods with cross-lingual supervision															
Sup-BWE-Direct	73.5	81.1	81.4	77.3	79.9	73.3	67.7	69.5	59.1	82.6	83.2	78.1	83.5	87.3	81.0
Unsupervised methods without cross-lingual supervision															
BWE-Pivot	74.0	82.3	81.7	77.0	80.7	71.9	66.1	68.0	57.4	81.1	79.7	74.7	81.9	85.0	78.9
BWE-Direct	74.0	82.3	81.7	77.0	80.7	73.0	65.7	66.5	58.5	83.1	83.0	77.9	83.3	87.3	80.5
`MAT+MPSR`	**74.8**	**82.4**	**82.5**	**78.8**	**81.5**	**76.7**	**69.6**	**72.0**	**63.2**	**83.9**	**83.5**	**79.3**	**84.5**	**87.8**	**82.3**

	de-en	fr-en	es-en	it-en	pt-en	fr-de	es-de	it-de	pt-de	es-fr	it-fr	pt-fr	it-es	pt-es	pt-it
Supervised methods with cross-lingual supervision															
Sup-BWE-Direct	72.4	**82.4**	82.9	76.9	**80.3**	69.5	68.3	67.5	63.7	85.8	87.1	84.3	87.3	91.5	81.1
Unsupervised methods without cross-lingual supervision															
BWE-Pivot	72.2	82.1	83.3	**77.7**	80.1	68.1	67.9	66.1	63.1	84.7	86.5	82.6	85.8	91.3	79.2
BWE-Direct	72.2	82.1	83.3	**77.7**	80.1	69.7	68.8	62.5	60.5	86	87.6	83.9	87.7	92.1	80.6
`MAT+MPSR`	**72.9**	81.8	**83.7**	77.4	79.9	**71.2**	**69.0**	**69.5**	**65.7**	**86.9**	**88.1**	**86.3**	**88.2**	**92.7**	**82.6**

(a) Detailed Results

	Training Cost		*Single Source*						*Single Target*						
	#BWEs	time	en-xx	de-xx	fr-xx	es-xx	it-xx	pt-xx	xx-en	xx-de	xx-fr	xx-es	xx-it	xx-pt	**Overall**
Supervised methods with cross-lingual supervision															
Sup-BWE-Direct	$N(N-1)$	$4h$	78.6	68.4	79.2	81.6	80.0	80.2	79.0	68.5	82.3	82.1	78.9	77.1	78.0
Unsupervised methods without cross-lingual supervision															
BWE-Pivot	$2(N-1)$	$8h$	79.1	67.1	77.1	80.6	79.0	79.3	**79.1**	67.8	81.6	81.2	77.2	75.3	77.0
BWE-Direct	$N(N-1)$	$23h$	79.1	67.2	79.2	81.7	79.2	79.4	**79.1**	67.1	82.6	82.1	78.1	77.0	77.6
`MAT+MPSR`	$N-1$	$5h$	**80.0**	**70.9**	**79.9**	**82.4**	**81.1**	**81.4**	79.1	**70.0**	**84.1**	**83.4**	**80.3**	**78.8**	**79.3**

(b) Summarized Results

Table 1: Multilingual Word Translation Results for English, German, French, Spanish, Italian and Portuguese. The reported numbers are *precision@1* in percentage. All systems use the nearest neighbor under the CSLS distance for predicting the translation of a certain word.

similar languages will result in reduced accuracy. Our `MAT+MPSR` method, however, overcomes this disadvantage of BWE-Pivot and achieves the best performance on all these pairs through an explicit multilingual learning mechanism without increasing the computational cost.

Furthermore, our method also beats the BWE-Direct approach, which supports our second hypothesis that utilizing knowledge from languages beyond the pair itself could improve performance. For instance, there are a few pairs where BWE-Pivot outperforms BWE-Direct, such as de-it, it-de and pt-de, even though it goes through a third language (English) in BWE-Pivot. This might suggest that for some less similar language pairs, leveraging a third language as a bridge could in some cases work better than only relying on the language pair itself. German is involved in all these language pairs where BWE-Pivot outperforms than BWE-Direct, which is potentially due to the similarity between German and the pivot language English. We speculate that if choosing a different pivot language, there might be other pairs that could benefit. This observation serves as a possible explanation of the superior performance of our multilingual method over BWE-Direct, since our method utilizes knowledge from all languages during training.

4.2 Cross-Lingual Word Similarity

In this section, we evaluate the quality of our MWEs on the cross-lingual word similarity (CLWS) task, which assesses how well the similarity in the cross-lingual embedding space corresponds to a human-annotated semantic similarity score. The high-quality CLWS dataset from SemEval-2017 (Camacho-Collados et al., 2017) is

	en-de	en-es	de-es	en-it	de-it	es-it	en-fa	de-fa	es-fa	it-fa	**Average**
Supervised methods with cross-lingual supervision											
Luminoso	**.769**	**.772**	**.735**	**.787**	**.747**	**.767**	.595	.587	.634	.606	.700
NASARI	.594	.630	.548	.647	.557	.592	.492	.452	.466	.475	.545
Unsupervised methods without cross-lingual supervision											
BWE-Pivot	.709	.711	.703	.709	.682	.721	.672	.655	.701	.688	.695
BWE-Direct	.709	.711	.703	.709	.675	.726	.672	.662	.714	.695	.698
`MAT+MPSR`	.711	.712	.708	.709	.684	.730	**.680**	**.674**	**.720**	**.709**	**.704**

Table 2: Results for the SemEval-2017 Cross-Lingual Word Similarity task. Spearman's ρ is reported. Luminoso (Speer and Lowry-Duda, 2017) and NASARI (Camacho-Collados et al., 2016) are the two top-performing systems for SemEval-2017 that reported results on all language pairs.

used for evaluation. The dataset contains word pairs from any two of the five languages: English, German, Spanish, Italian, and Farsi (Persian), annotated with semantic similarity scores.

In addition to the BWE-Pivot and BWE-Direct baseline methods, we also include the two best-performing systems on SemEval-2017, Luminoso (Speer and Lowry-Duda, 2017) and NASARI (Camacho-Collados et al., 2016) for comparison. Note that these two methods are supervised, and have access to the Europarl[3] (for all languages but Farsi) and the OpenSubtitles2016[4] parallel corpora.

Table 2 shows the results, where the performance of each model is measured by the Spearman correlation. When compared to the BWE-Pivot and the BWE-Direct baselines, `MAT+MPSR` continues to perform the best on all language pairs. The qualitative findings stay the same as in the word translation task, except the margin is less significant. This might be because the CLWS task is much more lenient compared to the word translation task, where in the latter one needs to correctly identify the translation of a word out of hundreds of thousands of words in the vocabulary. In CLWS though, one can still achieve relatively high correlation in spite of minor inaccuracies.

On the other hand, an encouraging result is that when compared to the state-of-the-art supervised results, our `MAT+MPSR` method outperforms NASARI by a very large margin, and achieves top-notch overall performance similar to the competition winner, Luminoso, without using any bitexts. A closer examination reveals that our unsupervised method lags a few points behind Luminoso on the European languages wherein the supervised methods have access to the large-scale high-quality Europarl parallel corpora. It is the low-resource language, Farsi, that makes our unsupervised method stand out. All of the unsupervised methods outperform the supervised systems from SemEval-2017 on language pairs involving Farsi, which is not covered by the Europarl bitexts. This suggests the advantage of learning unsupervised embeddings for lower-resourced languages, where the supervision might be noisy or absent. Furthermore, within the unsupervised methods, `MAT+MPSR` again performs the best, and attains a higher margin over the baseline approaches on the low-resource language pairs, vindicating our claim of better multilingual performance.

5 Conclusion

In this work, we propose a fully unsupervised model for learning multilingual word embeddings (MWEs). Although methods exist for learning high-quality unsupervised BWEs (Lample et al., 2018b), little work has been done in the unsupervised multilingual setting. Previous work relies solely on a number of unsupervised BWE models to generate MWEs (e.g. BWE-Pivot and BWE-Direct), which does not fully leverage the interdependencies among all the languages. Therefore, we propose the `MAT+MPSR` method that explicitly exploits the relations between all language pairs without increasing the computational cost. In our experiments on multilingual word translation and cross-lingual word similarity (SemEval-2017), we show that `MAT+MPSR` outperforms existing unsupervised and even supervised models, achieving new state-of-the-art performance.

For future work, we plan to investigate how our

---

[3] `http://opus.nlpl.eu/Europarl.php`

[4] `http://opus.nlpl.eu/OpenSubtitles2016.php`

method can be extended to work with other BWE frameworks, in order to overcome the instability issue of Lample et al. (2018b). As pointed out by recent work (Søgaard et al., 2018; Artetxe et al., 2018a), the method by Lample et al. (2018b) performs much worse on certain languages such as Finnish, etc. More reliable multilingual embeddings might be obtained on these languages if we adapt our multilingual training framework to work with the more robust methods proposed recently.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR*, abs/1602.01925.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018b. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26, Vancouver, Canada. Association for Computational Linguistics.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv e-prints 1606.01614v5*.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. 2017. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 854–863, International Convention Centre, Sydney, Australia. PMLR.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, fast cross-lingual word-embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1109–1113, Lisbon, Portugal. Association for Computational Linguistics.

Georgiana Dinu and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *International Conference on Learning Representations, Workshop Track*.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2017. Multilingual training of crosslingual word embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 894–904, Valencia, Spain. Association for Computational Linguistics.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471. Association for Computational Linguistics.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474, Mumbai, India. The COLING 2012 Organizing Committee.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herv Jgou. 2018b. Word translation without parallel data. In *International Conference on Learning Representations*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 3111–3119, USA. Curran Associates Inc.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.*, 11:2487–2531.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of ICLR*.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788. Association for Computational Linguistics.

Robert Speer and Joanna Lowry-Duda. 2017. Conceptnet at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 85–89.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 719–725, Beijing, China. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, Denver, Colorado. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970, Vancouver, Canada. Association for Computational Linguistics.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington, USA. Association for Computational Linguistics.

CLUSE: Cross-Lingual Unsupervised Sense Embeddings

Ta-Chung Chi and Yun-Nung Chen
National Taiwan University, Taipei, Taiwan
`r06922028@ntu.edu.tw` `y.v.chen@ieee.org`

Abstract

This paper proposes a modularized sense induction and representation learning model that jointly learns bilingual sense embeddings that align well in the vector space, where the cross-lingual signal in the English-Chinese parallel corpus is exploited to capture the collocation and distributed characteristics in the language pair. The model is evaluated on the Stanford Contextual Word Similarity (SCWS) dataset to ensure the quality of monolingual sense embeddings. In addition, we introduce Bilingual Contextual Word Similarity (BCWS), a large and high-quality dataset for evaluating cross-lingual sense embeddings, which is the first attempt of measuring whether the learned embeddings are indeed aligned well in the vector space. The proposed approach shows the superior quality of sense embeddings evaluated in both monolingual and bilingual spaces.[1]

1 Introduction

Word embeddings have recently become the basic component in most NLP tasks for its ability to capture semantic and distributed relationships learned in an unsupervised manner. The higher similarity between word vectors can indicate similar meanings of words. Therefore, embeddings that encode semantics have been shown to serve as the good initialization and benefit several NLP tasks. However, word embeddings do not allow a word to have different meanings in different contexts, which is a phenomenon known as polysemy. For example, "*apple*" may have different meanings in *fruit* and *technology* contexts. Several attempts have been proposed to tackle this problem by inferring multi-sense word representations (Reisinger and Mooney, 2010; Neelakantan et al., 2014; Li and Jurafsky, 2015; Lee and Chen, 2017).

These approaches relied on the "one-sense per collocation" heuristic (Yarowsky, 1993), which assumes that presence of nearby words correlates with the sense of the word of interest. However, this heuristic provides only a weak signal for discriminating sense identities, and it requires a large amount of training data to achieve competitive performance.

Considering that different senses of a word may be translated into different words in a foreign language, Guo et al. (2014) and Šuster et al. (2016) proposed to learn multi-sense embeddings using this additional signal. For example, "*bank*" in English can be translated into *banc* or *banque* in French, depending on whether the sense is financial or geographical. Such information allows the model to identify which sense a word belongs to. However, the drawback of these models is that the trained foreign language embeddings are not aligned well with the original embeddings in the vector space.

This paper addresses these limitations by proposing a bilingual modularized sense induction and representation learning system. Our learning framework is the first pure sense representation learning approach that allows us to utilize two different languages to disambiguate words in English. To fully use the linguistic signals provided by bilingual language pairs, it is necessary to ensure that the embeddings of each foreign language are related to each other (i.e., they align well in the vector space). We solve this by proposing an algorithm that jointly learns sense representations between languages. The contributions of this paper are four-fold:

- We propose the first system that maintains purely sense-level cross-lingual representation learning with linear-time sense decoding.
- We are among the first to propose a single ob-

---

[1] The code and dataset are available at `http://github.com/MiuLab/CLUSE`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 271–281
Brussels, Belgium, October 31 - November 4, 2018.

jective for modularized bilingual sense embedding learning.
- We are the first to introduce a high-quality dataset for directly evaluating bilingual sense embeddings.
- Our experimental results show the state-of-the-art performance for both monolingual and bilingual contextual word similarities.

2 Related Work

There are a lot of prior works focusing on representation learning, while this work mainly focuses on bridging the work about sense embeddings and cross-lingual embeddings and introducing a newly collected bilingual data for better evaluation.

Sense Embeddings Reisinger and Mooney (2010) first proposed multi-prototype embeddings to address the lexical ambiguity when using a single embedding to represent multiple meanings of a word. Huang et al. (2012); Neelakantan et al. (2014); Li and Jurafsky (2015); Bartunov et al. (2016) utilized neural networks as well as the Bayesian non-parametric method to learn sense embeddings. Lee and Chen (2017) first utilized a reinforcement learning approach and proposed a modularized framework that separates learning of senses from that of words. However, none of them leverages the bilingual signal, which may be helpful for disambiguating senses.

Cross-Lingual Word Embeddings Klementiev et al. (2012) first pointed out the importance of learning cross-lingual word embeddings in the same space and proposed the cross-lingual document classification (CLDC) dataset for extrinsic evaluation. Gouws et al. (2015) trained directly on monolingual data and extracted a bilingual signal from a smaller set of parallel data. Kočiskỳ et al. (2014) used a probabilistic model that simultaneously learns alignments and distributed representations for bilingual data by marginalizing over word alignments. Hermann and Blunsom (2014) learned word embeddings by minimizing the distances between compositional representations between parallel sentence pairs. Šuster et al. (2016) reconstructed the bag-of-words representation of semantic equivalent sentence pairs to learn word embeddings. Shi et al. (2015) proposed a training algorithm in the form of matrix decomposition, and induced cross-lingual constraints for simultaneously factorizing monolingual matrices. Luong et al. (2015) extended the skip-gram model to bilingual corpora where contexts of bilingual word pairs were jointly predicted. Wei and Deng (2017) proposed a variational autoencoding approach that explicitly models the underlying semantics of the parallel sentence pairs and guided the generation of the sentence pairs. Although the above approaches aimed to learn cross-lingual embeddings jointly, they fused different meanings of a word in one embedding, leading to lexical ambiguity in the vector space model.

Cross-Lingual Sense Embeddings Guo et al. (2014) adopted the heuristics where different meanings of a polysemous word usually can be represented by different words in another language and clustered bilingual word embeddings to induce senses. Šuster et al. (2016) proposed an encoder, which uses parallel corpora to choose a sense for a given word, and a decoder that predicts context words based on the chosen sense. Bansal et al. (2012) proposed an unsupervised method for clustering the translations of a word, such that the translations in each cluster share a common semantic sense. Upadhyay et al. (2017) leveraged cross-lingual signals in more than two languages. However, they either used pretrained embeddings or learned only for the English side, which is undesirable since cross-lingual embeddings shall be jointly learned such that they aligned well in the embedding space.

Evaluation Datasets Several datasets can be used to justify the performance of learned sense embeddings. Huang et al. (2012) presented SCWS, the first and only dataset that contains word pairs and their sentential contexts for measuring the quality of sense embeddings. However, it is a monolingual dataset constructed in English, so it cannot evaluate cross-lingual semantic word similarity. On the other hand, while Camacho-Collados et al. (2017) proposed a cross-lingual semantic similarity dataset, it ignored the contextual words but kept only word pairs, making it impossible to judge sense-level similarity. In this paper, we present an English-Chinese contextual word similarity dataset in order to benchmark the experiments about bilingual sense embeddings.

3 CLUSE: Cross-Lingual Unsupervised Sense Embeddings

Our proposed model borrows the idea about modularization from Lee and Chen (2017), which treats the sense induction and representation mod-

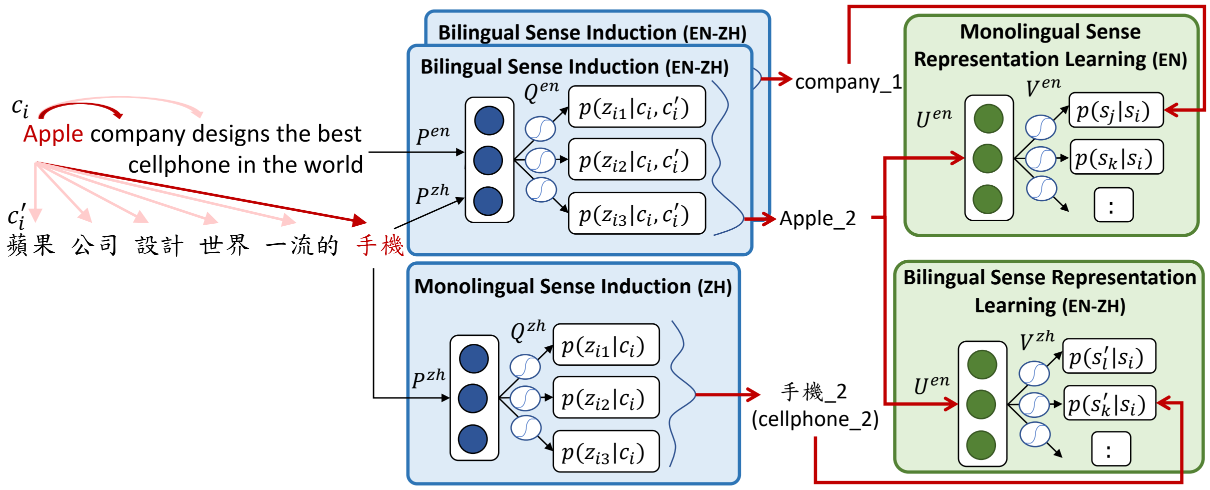

Figure 1: Sense induction modules decide the senses of words, and two sense representation learning modules optimize the sense collocated likelihood for learning sense embeddings within a language and between two languages. Two languages are treated equally and optimized iteratively.

ules separately to avoid mixing word-level and sense-level embeddings together.

Our model consists of four different modules illustrated in Figure 1, where sense induction modules decide the senses of words, and two sense representation learning modules optimize the sense collocated likelihood for learning sense embeddings within a language and between two languages in a joint manner. All modules are detailed below.

3.1 Notations

We denote our parallel corpus without word alignment C, where C^{en} is for the English part and C^{zh} is for the Chinese part. Our English vocabulary is W^{en} and Chinese vocabulary is W^{zh}. Moreover, C^{en}_t and C^{zh}_t are the t-th sentence-level parallel sentences in English and Chinese respectively. In the following sections, we treat English as the major language and Chinese as an additional bilingual signal, while their roles can be mutually exchanged. Specifically, English and Chinese iteratively become the major language during the training procedure.

3.2 Bilingual Sense Induction Module

The bilingual sense induction module takes a parallel sentence pair as input and determines which sense identity a target word belongs to given the bilingual contextual information. Formally, for the t-th English sentence C^{en}_t, we aim to decode the most probable sense $z_{ik} \in Z_i$ for the i-th word $w_i \in W^{en}$ in C^{en}_t, where Z_i is the set of sense candidates for w_i and $1 \le k \le |Z_i|$. We assume that the meaning of w_i can be determined by its surrounding words, or the so-called local context, $c_i = \{w_{i-m}, \cdots, w_{i+m}\}$, where m is the size of context window.

Aside from monolingual information, it is desirable to exploit the parallel sentences as additional bilingual contexts to enable cross-lingual embedding learning. Note that word alignment is not required in this work, so we consider the whole parallel bilingual sentence during training. Considering training efficiency, we sample M words in the parallel bilingual sentence with their original relative order or pad it to M for those shorter than M. Formally, given the t-th parallel bilingual sentence C^{zh}_t, the bilingual context of w_i is therefore $c'_i = \{w'_0, \cdots, w'_{M-1}\}$ and $w' \in W^{zh}$.

To ensure efficiency, continuous bag-of-words (CBOW) model is applied, where it takes word-level input tokens and outputs sense-level identities. Specifically, given an English *word* embedding matrix P^{en}, the local context can be modeled as the average of word embeddings from its context, $\frac{1}{|c_i|}\sum_{w_j \in c_i} P^{en}_j$. Similarly, we can model the bilingual contextual information given Chinese word embedding matrix P^{zh} using the CBOW formulation and obtain $\frac{1}{M}\sum_{w'_j \in c'_i} P^{zh}_j$. We linearly combine the contextual information from different languages as:

$$\bar{C} = \alpha \cdot \frac{1}{|c_i|} \sum_{w_j \in c_i} P^{en}_j + (1-\alpha) \cdot \frac{1}{M} \sum_{w'_j \in c'_i} P^{zh}_j. \tag{1}$$

The likelihood of selecting each sense identity z_{ik} for w_i can be formulated in the form of *Bernoulli* distribution with a sigmoid function $\sigma(\cdot)$:

$$p(z_{ik} \mid c_i, c'_i) = \sigma((Q^{en}_{ik})^T \bar{C}), \qquad (2)$$

where Q^{en} is a 3-dimensional tensor with each dimension denotes W^{en}, z_{ik} for a specific word i in W^{en}, and the corresponding latent variable, respectively. Therefore, Q^{en}_{ik} will retrieve the latent variable of k-th sense of i-th English word. Finally, we can induce the sense identity, z^*_{ik}, given the contexts of a word w_i from different languages, c_i and c'_i.

$$z^*_{ik} = \arg\max_{z_{ik}} p(z_{ik} \mid c_i, c'_i) \qquad (3)$$

In order to allow the module to explore other potential sense identities, we apply an ϵ-greedy algorithm (Mnih et al., 2013) for exploration in the training procedure.

3.3 Monolingual Sense Induction Module

This module is the degraded version of bilingual sense induction module when $\alpha = 1$, which occurs where *no* parallel bilingual signal exists. In other words, every bilingual sense induction module will experience the degradation during the training process presented in Algorithm 1. The only difference is that it cannot access the bilingual information. The purpose of this module is to maintain the stability of sense induction and to decode the sampled bilingual sense identity which will later be used in the bilingual sense representation learning module. As shown in Figure 1, given the monolingual context of a word, this module selects its sense identity using (2) and (3) with $\alpha = 1$.

3.4 Monolingual Sense Representation Learning Module

Given the decoded sense identities from the sense induction module, the skip-gram architecture (Mikolov et al., 2013) is applied considering that it only requires two decoded sense identities for stochastic training. We first create an input English sense representation matrix U^{en} and an English collocation estimation matrix V^{en} as the learning targets. Given a target word w_i and its collocated word w_j in the t-th English sentence C^{en}_t, we map them to their sense identities as $z^*_{ik} = s_i$ and $z^*_{jl} = s_j$ by the sense induction module and maximize the sense collocation likelihood. The skip-gram objective can be formulated as $p(s_j \mid s_i)$:

$$p(s_j \mid s_i) = \frac{\exp((U^{en}_{s_i})^T V^{en}_{s_j})}{\sum_{s_k} \exp((U^{en}_{s_i})^T V^{en}_{s_k})}, \qquad (4)$$

where s_k iterates over all possible English sense identities in the denominator. This formulation shares the same architecture as skip-gram but extends to rely on senses. Note that the Chinese sense representation learning module is built similarly.

3.5 Bilingual Sense Representation Learning Module

To ensure sense embeddings of two different languages align well, we hypothesize that the target sense identity s_i not only predicts the sense identity s_j of w_j in C^{en}_t but also one sampled sense identity s'_l of w'_l from the parallel sentence C^{zh}_t, where s'_l is decoded by the Chinese monolingual sense induction module. Specifically, the *bilingual* skip-gram objective can be formulated using the English sense embedding matrix U^{en} and the bilingual collocation estimation matrix V^{zh} as:

$$p(s'_l \mid s_i) = \frac{\exp((U^{en}_{s_i})^T V^{zh}_{s'_l})}{\sum_{s'_k} \exp((U^{en}_{s_i})^T V^{zh}_{s'_k})}, \qquad (5)$$

where s'_k iterates over all possible Chinese sense identities in the denominator.

3.6 Joint Learning

In this learning framework, the gradient cannot be back-propagated from the representation module to the induction module due to the usage of $\arg\max$ operator. It is therefore desirable to connect these two modules in a way such that they can improve each other by their own estimations. In one direction, forwarding the prediction of the sense induction module to the sense representation learning module is trivial, while in another direction, we treat the estimated collocation likelihood as the reward for the induction module.

First note that calculating the partition function in the denominator of (4) and (5) is intractable since it involves a computationally expensive summation over all sense identities. In practice, we adopt the negative sampling strategy technique (Mikolov et al., 2013) and rewrite (4) and (5) as:

$$\log p(s_j \mid s_i) = \log \sigma((U^{en}_{s_i})^T V^{en}_{s_j}) + \sum_{k=1}^{N} \mathbb{E}_{s_k \sim p_{\text{neg}}(s)}[\sigma(-(U^{en}_{s_i})^T V^{en}_{s_k})], \tag{6}$$

$$\log p(s'_l \mid s_i) = \log \sigma((U^{en}_{s_i})^T V^{zh}_{s'_l}) + \sum_{k=1}^{N} \mathbb{E}_{s'_k \sim p_{\text{neg}}(s')}[\sigma(-(U^{en}_{s_i})^T V^{zh}_{s'_k})], \tag{7}$$

where $p_{\text{neg}}(s)$ and $p_{\text{neg}}(s')$ is the distribution over all English senses and all Chinese senses for negative samples respectively, and N is the number of negative sample. The rewritten objective for optimizing two sense representation learning modules is the same as maximizing (6) and (7). Moreover, we can utilize the probability of correctly classifying the skip-gram sense pair as the reward signal. The intuition is that a correctly decoded sense identity is more likely to predict its neighboring sense identity compared to incorrectly decoded ones.

This learning framework can now be viewed as a reinforcement learning agent solving one-step Markov Decision Process (Sutton and Barto, 1998; Lee and Chen, 2017). For bilingual modules, the state, action, and reward correspond to bilingual context $\bar{C}$, sense z_{ik}, and $\sigma((U^{en}_{s_i})^T V^{zh}_{s'_l})$ respectively. As for the monolingual modules, the state, action, and reward correspond to monolingual context c_t, sense z_{ik}, and $\sigma((U^{en}_{s_i})^T V^{en}_{s_j})$. Finally, we can optimize both bilingual and monolingual sense induction modules (P and Q from (2) by minimizing the cross entropy loss between decoded sense probability and reward. We also include an entropy regularization term as suggested in (Šuster et al., 2016) to let the sense induction module converge faster and make more confident predictions. Formally,

$$\min H(\sigma((U^{en}_{s_i})^T V^{zh}_{s'_l}), p(z_{ik} \mid c_i, c'_i)) + \lambda E(p(z_{ik} \mid c_i, c'_i)) \tag{8}$$

$$\min H(\sigma((U^{en}_{s_i})^T V^{en}_{s_j}), p(z_{ik} \mid c_i)) + \lambda E(p(z_{ik} \mid c_i)) \tag{9}$$

E is the entropy of selection probability weighted by λ. Note that the major language is switched iteratively among two languages. Algorithm 1 presents the full learning procedure.

Algorithm 1 Bilingual Sense Embedding Learning Algorithm

```
Input: $C^{en}, C^{zh}, W^{en}, W^{zh}$
Output: $P^{en}, P^{zh}, Q^{en}, Q^{zh}, U^{en}, U^{zh}, V^{en}, V^{zh}$
loop until converge
    MAIN(en, zh, 0.4)    ▷ 0.4 is just an example weight
    MAIN(zh, en, 0.4)
end loop
function MAIN(maj, bi, $\alpha$)
    $t, i, j, k, l \leftarrow$ GETTRAINDATA(maj)
    $s_i, pred_i \leftarrow$ INDUCESENSE(maj, bi, $t$, $i$, $\alpha$)
    $s_j, \_ \leftarrow$ INDUCESENSE(maj, bi, $t$, $j$, $\alpha$)
    $s'_l, pred'_l \leftarrow$ INDUCESENSE(bi, bi, $t$, $k$, 1.0)
    $s'_k, \_ \leftarrow$ INDUCESENSE(bi, bi, $t$, $l$, 1.0)
    $r \leftarrow$ TRAINSRL(maj, maj, $s_i$, $s_j$)
    $r' \leftarrow$TRAINSRL(maj, bi, $s_i$, $s'_l$)
    $r'' \leftarrow$TRAINSRL(bi, bi, $s'_l$, $s'_k$)
    TRAINSI(maj, bi, $r, pred_i$)
    TRAINSI(maj, bi, $r', pred_i$)
    TRAINSI(bi, bi, $r'', pred'_l$)
end function
function INDUCESENSE(maj, bi, $t$, $i$, $\alpha$)
    calculate $\alpha$-weighted $\bar{C}$ by (1)
    select $z^*_{ik}$ by (2) and (3)
    return $z^*_{ik}, p(z^*_{ik} \mid \bar{C})$
end function
function TRAINSRL(maj, bi, $s_i$, $s_j$)
    if maj==bi then
        optimize $U^{maj}$, $V^{maj}$ by (6) given $s_i$, $s_j$
    else
        optimize $U^{maj}$, $V^{bi}$ by (7) given $s_i$, $s_j$
    end if
    return collocation prob of $(s_i, s_j)$
end function
function TRAINSI(maj, bi, r, pred)
    if maj==bi then
        optimize $P^{maj}$, $Q^{maj}$ by (9) given r, pred
    else
        optimize $P^{maj}$, $Q^{bi}$ by (8) given r, pred
    end if
end function
```

4 New Dataset—Bilingual Contextual Word Similarity (BCWS)

We propose a new dataset to measure the bilingual contextual word similarity. English and Chinese are chosen as our language pair for three reasons:

1. They are the top widely used languages in the world.
2. English and Chinese belong to completely different language families, making it interesting to explore syntactic and semantic difference among them.
3. Chinese is a language that requires segmentation, this dataset can also help researchers experiment on different segmentation levels and investigate how segmentation affects the

English Sentence	**Chinese Sentence**	**Score**
Judges must give both sides an equal opportunity to <**state**> their cases.	我非常喜歡這個故事，它<**告訴**>我們一些重要的啓示。 (I like this story a lot, which <**tells**> us some important inspiration.)	7.00
It was of negligible <**importance**> prior to 1990, with antiquated weapons and few members.	黃斑部病變的預防及早期治療是相當<**重要**>的。 (The prevention and early treatment of macular lesions is very <**important**>.)	6.94
Due to the San Andreas Fault bisecting the hill, one side has <**cold**> water, the other has hot.	水果攤老闆似乎很意外眞有人買這<**冷**>貨，露出「你眞内行」的眼神與我聊了幾句。 (The owner of the fruit stall seemed surprised that someone bought this <**unpopular**> product, talking me few words about “you are such a pro”.)	3.70

Table 1: Sentence pair examples and average annotated scores in BCWS.

sense similarity.

This dataset also provides a *direct* measure to determine whether the two language embeddings align well in the vector space. Note that we focus on word-level, and this is different from (Klementiev et al., 2012), which also measured the cross-lingual embedding similarity but rely on the ambiguous document-level classification.

Our dataset contains 2093 question pairs, where each pair consists of exactly one English and one Chinese sentence; note that they are **not** parallel but with their own sentential contexts shown in Table 1. Eleven raters[2] were recruited to annotate this dataset. Each rater gives a score ranging from 1.0 (different) to 10.0 (same) for each question to indicate the semantic similarity of bilingual word pairs based on sentential clues. The annotated dataset shows very high intra-rater consistency; we leave one rater out and calculate Spearman correlation between the rater and the average of the rest, and the average number is about 0.83, indicating the human-level performance (the average number in SCWS is 0.52).

We describe the construction of BCWS below.

Chinese Multi-Sense Word Extraction We utilize the Chinese Wikipedia dump to extract the most frequent 10000 Chinese words that are *nouns*, *adjective*, and *verb* based on Chinese Wordnet (Huang et al., 2010). In order to test the sense-level representations, we discard single-sense words to ensure that the selected words are polysemous. Also, the words with more than 20 senses are deleted, since those senses are too fine-grained and even hard for human to disambiguate. We denote the list of Chinese words l_c.

English Candidate Word Extraction We have to find an English counterpart for each Chinese word in l_c. We utilize *BabelNet* (Navigli and Ponzetto, 2010), a free and open-sourced knowledge resource, to serve as our bilingual dictionary. To be more concrete, we first query the selected Chinese word using the free API call provided by Babelnet to retrieve all *WordNet* senses[3]. For example, the Chinese word “制服” has two major meanings:

- *a type of clothing worn by members of an organization*
- *force to submit or subdue*.

Hence, we can obtain two candidate English words “*uniform*” and “*subjugate*”. Each word in l_c retrieves its associated English candidate words and obtain the dictionary D.

Enriching Semantic Relationship Note that D is merely a simple translation mapping between Chinese and English words. It is desirable that we have a more complicated and interesting relationship between bilingual word pairs. Hence, we traverse D and for each English word we find its *hyponyms*, *hypernyms*, *holonyms* and *attributes*, and add the additional words into D. In our example, we may obtain {制服:[uniform, subjugate, livery, clothing, repress, dominate, enslave, dragoon...]}. We sample 2 English words if the number of English candidate words is more than 5, 3 English words if more than 10, and 1 English word oth-

---

[2] They are all Chinese native speaker whose scores are at least 29 in the TOEFL reading section or 157 in the GRE verbal section.

[3] *BabelNet* contains sense definitions from various resources such as Wordnet, Wikitionary, Wikidata, etc

erwise to form the final bilingual pair. For example, a bilingual word pair (制服, enslave) can be formed accordingly. After this step, we obtain 2093 bilingual word pairs P.

Adding Contextual Information Given the bilingual word pairs P, appropriate contexts should be found in order to form the full sentences for human judgment. For each Chinese word, we randomly sample one example sentence in Chinese WordNet that matches the PoS tag we selected in section 4. For each English word, we traverse the whole English Wikipedia dump to find the sentences that contain the target English word. We then sample one sentence where the target word is tagged as the matched PoS tag[4].

5 Experiments

5.1 Experimental Setup

Two sets of parallel data are used in the experiments, one for English-Chinese (EN-ZH) and another for English-German (EN-DE). UM-corpus (Tian et al.) is used for EN-ZH training, while Europarl corpus (Koehn, 2005) is used for EN-DE training. UM-corpus contains 15,764,200 parallel sentences with 381,921,583 English words and 572,277,658 unsegmented Chinese words. Europarl contains 1,920,209 parallel sentences with 44,548,491 German words and 47,818,827 English words. We evaluate our proposed model on the benchmark monolingual dataset, SCWS, and on the bilingual dataset, our proposed BCWS, where the evaluation metrics are actually introduced in section 5.4.

5.2 Hyperparameter Settings

In our experiments, we use a mini-batch size of 512, context window size for major language is set to $m = 5$ and we sample $M = 20$ words for bilingual context. For the exploration of sense induction module, we set $\epsilon = 0.05$. The λ of entropy regularization is set to 1.[5] For negative sampling in (6) and (7), we pick $N = 25$. The fixed learning rate is set to 0.025. The embedding dimension is 300 and the sense number per word is set to 3 for both Chinese, German, and English ($|Z_i| = 3$). This setting is for a fair comparison with prior works.

5.3 Baseline

The baselines for comparison can be categorized into three:

- *Monolingual sense embeddings*: Lee and Chen (2017) is the current state-of-the-art model of monolingual sense embedding evaluated on SCWS. We re-train the sense embeddings using the same data but only in English for fair comparison.
- *Cross-lingual word embeddings*: Luong et al. (2015) treated words from different languages the same and trained cross-lingual embeddings in the same space. Conneau et al. (2017) utilized adversarial training to map pretrained word embeddings into another language space.
- *Cross-lingual sense embeddings*: Upadhyay et al. (2017) utilized more than two languages to learn multilingual embeddings. We report the number shown in the paper for comparison.

5.4 Evaluation Metric

Reisinger and Mooney (2010) introduced two contextual similarity estimations, AvgSimC and MaxSimC. AvgSimC is a *soft* measurement that addresses the contextual information with a probability estimation:

$$\text{AvgSimC}(w_i, \bar{C}_t, w_j, \bar{C}_{t'}) = \sum_{k=1}^{|Z_i|} \sum_{l=1}^{|Z_j|} \pi(z_{ik}|\bar{C}_t)\pi(z_{jl}|\bar{C}_{t'})d(z_{ik}, z_{jl}),$$

AvgSimC weights the similarity measurement of each sense pair z_{ik} and z_{jl} by their probability estimations. On the other hand, MaxSimC is a *hard* measurement that only considers the most probable senses:

$$\text{MaxSimC}(w_i, \bar{C}_t, w_j, \bar{C}_{t'}) = d(z_{ik}, z_{jl}),$$
$$z_{ik} = \arg\max_{z_{ik'}} \pi(z_{ik'}|\bar{C}_t),$$
$$z_{jl} = \arg\max_{z_{jl'}} \pi(z_{jl'}|\bar{C}_{t'}).$$

$d(z_{ik}, z_{jl})$ refers to the cosine similarity between $U^{maj}_{z_{ik}}$ and $U^{bi}_{z_{jl}}$ in the bilingual case (BCWS) and $U^{maj}_{z_{ik}}$ and $U^{maj}_{z_{jl}}$ in the monolingual case (SCWS).

5.5 Bilingual Embedding Evaluation

Cross-lingual sense embeddings are the main contribution of this paper. Table 2 shows that all results from the proposed model are significantly

---

[4] We use the NLTK PoS tagger to obtain the tags.

[5] We tried different values of $\lambda = 0.001, 0.5$, and the model converges approximately 12, 5 times slower compared to $\lambda = 1$.

Model	α	EN-ZH		EN-DE
		Bilingual/BCWS	Mono(EN)/SCWS	Mono(EN)/SCWS
1) Monolingual Sense Embeddings				
Lee and Chen (2017)			**66.8** / 65.5	**63.8** / 63.4
2) Cross-Lingual Word Embeddings				
Luong et al. (2015)		49.2	61.1	62.1
Conneau et al. (2017)		52.5	65.5	**64.0**
3) Cross-Lingual Sense Embeddings				
Upadhyay et al. (2017)		-	45.0*	-
Proposed	0.1	55.8 / 55.8	65.6 / 65.6	**63.8** / 63.9
	0.3	55.7 / 55.7	64.9 / 65.1	**63.8 / 64.0**
	0.5	56.3 / 56.3	65.8 / 66.0	63.6 / 63.9
	0.7	**56.7 / 56.7**	65.6 / 65.8	63.1 / 63.2
	0.9	56.0 / 56.0	66.0 / **66.2**	62.9 / 63.1

Table 2: Contextual similarity results evaluated on the SCWS/BCWS dataset, where the reported numbers indicate Spearman's rank correlation $\rho \times 100$ on AvgSimC / MaxSimC. * indicates that Upadhyay et al. (2017) trained the sense embeddings using a different parallel dataset.

better than the baselines that learn cross-lingual word embeddings. It indicates that the sense-level information is critical for precise vector representations. In addition, all results for AvgSimC and MaxSimC are the same in the proposed model, showing that the learned selection distribution is reliable for sense decoding.

5.6 Monolingual Embedding Evaluation

Because our model considers multiple languages and learns the embeddings jointly, the multilingual objective makes learning more difficult due to more noises. In order to ensure the quality of the monolingual sense embeddings, we also evaluate our learned English sense embeddings on the benchmark SCWS data. Comparing the results between training on EN-ZH and training on EN-DE, all results using EN-ZH are better than ones using EN-DE. The probable reason is that the language difference between English and Chinese is larger than English and German; parallel Chinese sentences therefore provide informative cues for learning better sense embeddings. Furthermore, our proposed model achieves comparable or superior performance than the current state-of-the-art monolingual sense embeddings proposed by Lee and Chen (2017) when trained on our monolingual data.

5.7 Sensitivity of Bilingual Contexts

To investigate how much the bilingual sense induction module relies on another language, the results with different α are shown in the table.

Model	EN2DE	DE2EN
1) Sentence-Level Training		
Hermann and Blunsom (2014)	83.7	71.4
AP et al. (2014)	**91.8**	72.8
Wei and Deng (2017)	91.0	**80.4**
2) Word-Level Training		
Klementiev et al. (2012)	77.7	71.1
Gouws et al. (2015)	86.5	75.0
Kočiský et al. (2014)	83.1	75.4
Shi et al. (2015)	91.3	**77.2**
Luong et al. (2015)	**86.4**	75.6
Conneau et al. (2017)	78.7	67.1
Proposed	81.8	76.0

Table 3: Accuracy on cross-lingual document classification (%).

To justify the usefulness of utilizing bilingual signal, we compare our model with Lee and Chen (2017), which used monolingual signal in a similar modular framework. Our method outperforms theirs in terms of MaxSimC on both EN-ZH and EN-DE. However, this trend is not observed on AvgSimC. The reason may be that bilingual signal is indicative but noisy, which largely affects AvgSimC due to its weighted sum operation. MaxSimC only picks the most probable senses, which makes it robust to noises.

In addition, our performance slightly degrades as α increases for EN-DE, and the best performance is obtained when α is small, indicating that

Target	kNN Senses (EN)	kNN Senses (ZH)
apple_0	fruit, cake, sweet	蘋果, 春天, 蛋糕, iphone, 雞蛋, 巧克力, 葡萄 (apple, spring, cake, iphone, egg, chocolate, purples)
apple_1	iphone, cake, google, stores	蘋果, iphone, 微軟, 競爭對手, 春天, 谷歌 (apple, iphone, microsoft, competitor, spring, google)
uniform_0	dressed, worn, tape, wearing, cloth	均匀,光滑,衣服,鞋子,穿著,服裝 (even, smooth, clothes, shoes, wearing, clothing)
uniform_1	particle, computed, varying, gradient	態,粉末,縱向,等離子體,剪切,剛度 (phase, powder, longitudinal, plasma, cut, stiffness)

Table 4: Words with similar senses obtained by kNN.

bilingual signal does help. However, this trend is not observed on EN-ZH, because English is very different from Chinese, such that it can benefit little from Chinese than from German.

5.8 Extrinsic Evaluation

We further evaluate our bilingual sense embeddings using a downstream task, cross-lingual document classification (CLDC), with a standard setup (Klementiev et al., 2012). To be more concrete, a set of labeled documents in language *A* is available to train a classifier, and we are interested in classifying documents in another language *B* at test time, which tests semantic transfer of information across different languages. We use the averaged sense embeddings as word embeddings for a fair comparison.

The result is shown in Table 3. We can see that our proposed model achieves comparable performance or even superior performance to most prior work on the DE2EN direction; however, the same conclusion does not hold for the EN2DE direction. The reason may be that we test the model that works best on BCWS and hence not able to tune hyperparameters on the development set of CLDC. In addition, we use the average of sense vectors as input word embeddings, which may induce some noises into the resulting vectors. In sum, the comparable performance of the downstream task shows the practical usage and the potential extension of the proposed model.

5.9 Qualitative Analysis

Some examples of our learned sense embeddings are shown in Table 4. It is obvious to see that the first sense of *Apple* is related to *fruit and things to eat*, while the second one means the *tech company Apple Inc*. Most English and Chinese nearest neighbors match the meanings of the induced senses, but there are still some noises that are underlined. For example, *cake* should be the neighbor of the first sense rather than the second one. The same observation applies to *iphone* and *spring*. In our second example for *uniform*, the first sense is related to *outfit and clothes*, while the second is related to *engineering terms*. However, *even* appears in the *outfit and clothes* sense, which is incorrect. The reason may be that the size of the parallel corpus is not large enough for the model to accurately distinguish all senses via unsupervised learning. Hence, utilizing external resources such as bilingual dictionaries or designing a new model that can use existing large monolingual corpora like Wikipedia can be our future work.

6 Conclusion

This paper is the first purely sense-level cross-lingual representation learning model with efficient sense induction, where several monolingual and bilingual modules are jointly optimized. The proposed model achieves superior performance on both bilingual and monolingual evluation datasets. A newly collected dataset for evaluating bilingual contextual word similarity is presented, which provides potential research directions for future work.

Acknowledgement

We would like to thank reviewers for their insightful comments on the paper. This work was financially supported from the Young Scholar Fellowship Program by Ministry of Science and Technology (MOST) in Taiwan, under Grant 107-2636-E-002-004.

References

Sarath Chandar AP, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861.

Mohit Bansal, John DeNero, and Dekang Lin. 2012. Unsupervised translation sense clustering. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 773–782. Association for Computational Linguistics.

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *Artificial Intelligence and Statistics*, pages 130–138.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *International Conference on Machine Learning*, pages 748–756.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 497–507.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*.

Chu-Ren Huang, Shu-Kai Hsieh, Jia-Fei Hong, Yun-Zhu Chen, I-Li Su, Yong-Xiang Chen, and Sheng-Wei Huang. 2010. Chinese wordnet: Design, implementation, and application of an infrastructure for cross-lingual knowledge processing. *Journal of Chinese Information Processing*, 24(2):14–23.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. *Proceedings of COLING 2012*, pages 1459–1474.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. *arXiv preprint arXiv:1405.0947*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Guang-He Lee and Yun-Nung Chen. 2017. MUSE: Modularizing unsupervised sense embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 327–337.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in neural information processing systems*, pages 3111–3119.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop*.

Roberto Navigli and Simone Paolo Ponzetto. 2010. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 216–225. Association for Computational Linguistics.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient nonparametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning cross-lingual word embeddings via matrix co-factorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint*

Conference on Natural Language Processing (Volume 2: Short Papers), volume 2, pages 567–572.

Simon Šuster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. In *Proceedings of NAACL-HLT*, pages 1346–1356.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Liang Tian, Derek F Wong, Lidia S Chao, Paulo Quaresma, and Francisco Oliveira. Um-corpus: A large english-chinese parallel corpus for statistical machine translation.

Shyam Upadhyay, Kai-Wei Chang, Matt Taddy, Adam Kalai, and James Zou. 2017. Beyond bilingual: Multi-sense word embeddings using multilingual context. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 101–110.

Liangchen Wei and Zhi-Hong Deng. 2017. A variational autoencoding approach for inducing cross-lingual word embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4165–4171. AAAI Press.

David Yarowsky. 1993. One sense per collocation. In *Proceedings of the workshop on Human Language Technology*, pages 266–271. Association for Computational Linguistics.

Adversarial Propagation and Zero-Shot Cross-Lingual Transfer of Word Vector Specialization

Edoardo M. Ponti[1]*, **Ivan Vulić**[1]*, **Goran Glavaš**[2], **Nikola Mrkšić**[3], **Anna Korhonen**[1]

[1]Language Technology Lab, DTAL, University of Cambridge
[2]Data and Web Science Group, University of Mannheim
[3]PolyAI
[1]{ep490,iv250,alk23}@cam.ac.uk
[2]goran@informatik.uni-mannheim.de
[3]nikola@poly-ai.com

Abstract

Semantic *specialization* is a process of fine-tuning pre-trained distributional word vectors using external lexical knowledge (e.g., WordNet) to accentuate a particular semantic relation in the specialized vector space. While post-processing specialization methods are applicable to arbitrary distributional vectors, they are limited to updating only the vectors of words occurring in external lexicons (i.e., *seen words*), leaving the vectors of all other words unchanged. We propose a novel approach to specializing the full distributional vocabulary. Our *adversarial post-specialization* method propagates the external lexical knowledge to the full distributional space. We exploit words seen in the resources as training examples for learning a global specialization function. This function is learned by combining a standard L_2-distance loss with a adversarial loss: the adversarial component produces more realistic output vectors. We show the effectiveness and robustness of the proposed method across three languages and on three tasks: word similarity, dialog state tracking, and lexical simplification. We report consistent improvements over distributional word vectors and vectors specialized by other state-of-the-art specialization frameworks. Finally, we also propose a cross-lingual transfer method for *zero-shot specialization* which successfully specializes a full target distributional space without any lexical knowledge in the target language and without any bilingual data.

1 Introduction

Word representation learning is a mainstay of modern Natural Language Processing (NLP), and its usefulness has been proven across a wide spectrum of NLP applications (Collobert et al., 2011; Chen and Manning, 2014; Melamud et al., 2016b, *inter alia*). Standard *distributional* word vector models are grounded in the distributional hypothesis (Harris, 1954), that is, they leverage information about word co-occurrences in large text corpora (Mikolov et al., 2013; Pennington et al., 2014; Levy and Goldberg, 2014; Bojanowski et al., 2017). This dependence on contextual signal results in a well-known tendency to conflate semantic similarity with other types of semantic association (Hill et al., 2015; Schwartz et al., 2015; Vulić et al., 2017) in the induced word vector spaces.[1]

A common remedy is to move beyond purely unsupervised word representation learning, in a process referred to as *semantic specialization* or *retrofitting*. Specialization methods exploit lexical knowledge from external resources, such as WordNet (Fellbaum, 1998) or the Paraphrase Database (Ganitkevitch et al., 2013) to refine the semantic properties of pre-trained vectors and *specialize* the distributional spaces for a particular relation, e.g., synonymy (i.e., true similarity) (Faruqui et al., 2015; Mrkšić et al., 2017) or hypernymy (Nickel and Kiela, 2017; Nguyen et al., 2017; Vulić and Mrkšić, 2018).

The best-performing specialization models (cf. Mrkšić et al. 2017) are deployed as *post-processors of the vector space*: distributional vectors are *fine-tuned* to satisfy linguistic constraints extracted from external resources to offer improved support to downstream NLP applications (Faruqui, 2016). Such models are versatile as they can be applied to arbitrary distributional spaces, but they have a major drawback: they *locally* update only vectors of words present in linguistic constraints (i.e., *seen* words), whereas vectors of all other (i.e., *unseen*) words remain intact (see Figure 1).

---

[*] Both authors equally contributed to this work.

[1] For instance, it is difficult to discern synonyms from antonyms in distributional vector spaces: this has a negative impact on language understanding tasks such as statistical dialog modeling or text simplification (Glavaš and Štajner, 2015; Faruqui et al., 2015; Mrkšić et al., 2016; Kim et al., 2016)

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 282–293
Brussels, Belgium, October 31 - November 4, 2018.

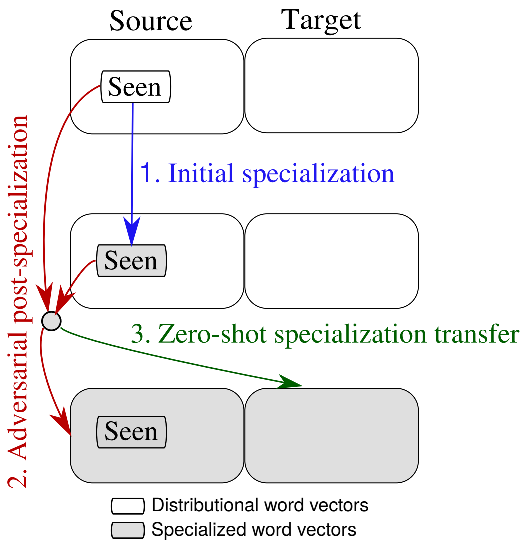

Figure 1: High-level illustration of the adversarial post-specialization process and cross-lingual zero-shot specialization, described in detail in §2.

Vulić et al. (2018) have recently proposed a model which, based on the updates of vectors of seen words, learns a global specialization function that can be applied to the large subspace of unseen words. Their global method, termed *post-specialization* and implemented as a deep feed-forward network, effectively specializes *all* distributional vectors.

In this paper, we propose a new approach to post-specialization which addresses the following two research questions: **a)** Is it possible to use a more sophisticated learning approach to yield more realistic specialized vectors for the full vocabulary? **b)** Given that specialization methods inherently require a large number of constraints, is it possible to specialize distributional word vectors where such resources are scarce or non-existent? Our novel model learns the global specialization function by casting the feed-forward specialization network as a generator component of an adversarial architecture, see Figure 2. The corresponding discriminator component learns to discern original specialized vectors (produced by any *local* specialization model) from vectors produced by transforming distributional vectors with the feed-forward post-specialization network (i.e., the generator).

We show that the proposed adversarial model yields state-of-the-art performance on standard word similarity benchmarks, outperforming the post-specialization model of Vulić et al. (2018). We further demonstrate the effectiveness of the proposed model in two downstream tasks: lexical text simplification and dialog state tracking. Finally, we demonstrate that, by coupling our adversarial specialization model with any unsupervised model for inducing bilingual vector spaces, such as the algorithm proposed by Conneau et al. (2018), we can successfully perform zero-shot language transfer of the specialization, that is, we can specialize distributional spaces of languages without any linguistic constraints in those languages, and without any bilingual data.

2 Methodology

The post-specialization procedure (Vulić et al., 2018) is a two-step process. First, a subspace of vectors for words observed in external resources is fine-tuned using any off-the-shelf specialization model, such as the original retrofitting model (Faruqui et al., 2015), counter-fitting (Mrkšić et al., 2016), dLCE (Nguyen et al., 2016), or state-of-the-art ATTRACT-REPEL (AR) specialization (Mrkšić et al., 2017; Vulić et al., 2017). We outline the initial specialization algorithms in §2.1. In the second step, the initial specialization is propagated to the entire vocabulary, including words not observed in the resources, relying on an adversarial architecture augmented with a distance loss. This adversarial post-specialization model, compatible with any specialization model, is described in §2.2.

Finally, in §2.3, we introduce a *cross-lingual zero-shot specialization model* which transfers the specialization to a target language without any lexical resources. An overview of the proposed methodology from this section is provided in Figure 1.

2.1 Initial Specialization

Linguistic Constraints. Adopting the nomenclature from Mrkšić et al. (2017), post-processing models are generally guided by two broad sets of constraints: **1)** ATTRACT constraints specify which words should be close to each other in the fine-tuned vector space (e.g. synonyms like *graceful* and *amiable*); **2)** REPEL constraints describe which words should be pulled away from each other (e.g. antonyms like *innocent* and *sinful*). Earlier post-processors (Faruqui et al., 2015; Jauhar et al., 2015; Wieting et al., 2015) operate only with ATTRACT constraints, and are thus not suited to model both aspects contributing to the specialization process.

We first outline the state-of-the-art ATTRACT-REPEL specialization model (Mrkšić et al., 2017) which leverages both sets of constraints. Here, we again stress two important aspects relevant to our post-specialization model: **a)** all initial specialization models fine-tune only representations for the subspace of words seen in the external constraints, while all other words remain unaffected by specialization; **b)** post-specialization is not tied to ATTRACT-REPEL in particular; it is applicable on top of any other post-processor.[2]

Specialization of Seen Words. The key idea is to inject the knowledge from linguistic constraints into pre-trained distributional word vectors. Given a set A of ATTRACT word pairs and a set R of REPEL word pairs, each word pair (v_l, v_r) from the vocabulary $\mathcal{V}_s$ of seen words present in these sets can be represented as a vector pair $(\mathbf{x}_l, \mathbf{x}_r)$.

The optimization is driven by mini-batches of ATTRACT pairs $\mathcal{B}_A$ (batch size k_A), and of REPEL pairs $\mathcal{B}_R$ (size k_R). For both of these, two sets of negative example pairs of equal size are drawn from the $2(k_A + k_R)$ vectors occurring in $\mathcal{B}_A$ and $\mathcal{B}_R$. This defines the mini-batches $T_A(B_A) = [(\mathbf{t}_l^1, \mathbf{t}_r^1) \dots (\mathbf{t}_l^{k_A}, \mathbf{t}_r^{k_A})]$ and $T_R(B_R) = [(\mathbf{t}_l^1, \mathbf{t}_r^1) \dots (\mathbf{t}_l^{k_R}, \mathbf{t}_r^{k_R})]$. Negative examples $\mathbf{t}_l$ and $\mathbf{t}_r$ for ATTRACT (or REPEL) pairs are the nearest (or farthest) neighbours by cosine similarity to $\mathbf{x}_l$ and $\mathbf{x}_r$, respectively. They ensure that the paired vectors for words in the constraints are closer to each other (or more distant for antonyms) than to their respective negative examples.

The overall objective function consists of three terms. The first term pulls ATTRACT pairs together:

$$
\begin{aligned}
Att(\mathcal{B}_A, \mathcal{T}_A) = \sum_{i=1}^{k_A} \Big[& \tau\big(\delta_A + \mathbf{x}_l^i \mathbf{t}_l^i - \mathbf{x}_l^i \mathbf{x}_r^i\big) + \\
& + \tau\big(\delta_A + \mathbf{x}_r^i \mathbf{t}_r^i - \mathbf{x}_l^i \mathbf{x}_r^i\big) \Big] \quad (1)
\end{aligned}
$$

$\tau(z) = \max(0, z)$ is the standard rectifier (Nair and Hinton, 2010). δ_A is the ATTRACT margin: it specifies the tolerance for the difference between the two distances (with the other pair member and with the negative example). The second term, $Rep(\mathcal{B}_R, \mathcal{T}_R)$, is similar but now pushes REPEL pairs away from each other, relying on the REPEL margin δ_R:

$$
\begin{aligned}
Rep(\mathcal{B}_R, \mathcal{T}_R) = \sum_{i=1}^{k_R} [& \tau\left(\delta_R - \mathbf{x}_l^i \mathbf{t}_l^i + \mathbf{x}_l^i \mathbf{x}_r^i\right) + \\
& + \tau\left(\delta_R - \mathbf{x}_r^i \mathbf{t}_r^i + \mathbf{x}_l^i \mathbf{x}_r^i\right)] \quad (2)
\end{aligned}
$$

The final term is tasked to *preserve* the quality of the original vectors through L_2-regularization:

$$
Pre(\mathcal{B}_A, \mathcal{B}_R) = \sum_{\mathbf{x}_i \in \mathcal{B}_A \cup \mathcal{B}_R} \lambda_P ||\mathbf{y}_i - \mathbf{x}_i||_2 \quad (3)
$$

$\mathbf{y}_i$ is the vector specialized from the original distributional vector $\mathbf{x}_i$, and λ_P is a regularization hyper-parameter. The optimizer finally minimizes the following objective: $\mathcal{L}_{AR} = Att(\mathcal{B}_A, \mathcal{T}_A) + Rep(\mathcal{B}_R, \mathcal{T}_R) + Pre(\mathcal{B}_A, \mathcal{B}_R)$.

2.2 Adversarial Post-Specialization

Motivation. The AR method affects only a subset of the full vocabulary $\mathcal{V}$, and consequently only a (small) subspace of the original space $\mathbf{X}$ (see Figure 1). In particular, it specializes the embeddings $\mathbf{X}_s$ corresponding to $\mathcal{V}_s$, the vocabulary of words observed in the constraints. It leaves the embeddings $\mathbf{X}_u$ corresponding to all other (unseen) words $\mathcal{V}_u$ identical.

Nevertheless, the perturbation underwent by the original observed embeddings can provide evidence about the general effects of specialization. In particular, it allows to learn a global mapping function $f : \mathbf{X} \in \mathbb{R}^d \rightarrow \mathbf{Y} \in \mathbb{R}^d$ for d-dimensional vectors. The parameters for this function can be trained in a supervised fashion from pairs of original and initially specialized word embeddings $(\mathbf{x}_i^{(s)}, \mathbf{y}_i^{(s)})$ from $\mathcal{V}_s$, as illustrated by Figure 2. Subsequently, the mapping can be applied to distributional word vectors $\mathbf{x}_u$ from the vocabulary of unseen words $\mathcal{V}_u$ to predict $\hat{\mathbf{y}}_u$, their specialized counterpart. This procedure, called post-specialization, effectively propagates the information stored in the external constraints to the entire word vector space.

However, this mapping should not just model the inherent transformation, but also ensure that the resulting vector is 'natural'. In particular, assuming that word representations lie on a manifold, the mapping should return one of its values. The intuition behind our formulation of the training objective is that: **a)** an L_2-distance loss can retrieve a faithful mapping whereas **b)** an adversarial loss can prevent unrealistic outputs, as already proven in the the visual domain (Pathak et al., 2016; Ledig et al., 2017; Odena et al., 2017).

---

[2] We have empirically validated the robustness of the proposed adversarial post-specialization by applying it also on top of other post-processing methods: retrofitting (Faruqui et al., 2015) and counter-fitting (Mrkšić et al., 2016). For brevity, we only report the (best) results with ATTRACT-REPEL, the best-performing initial/local specialization model.

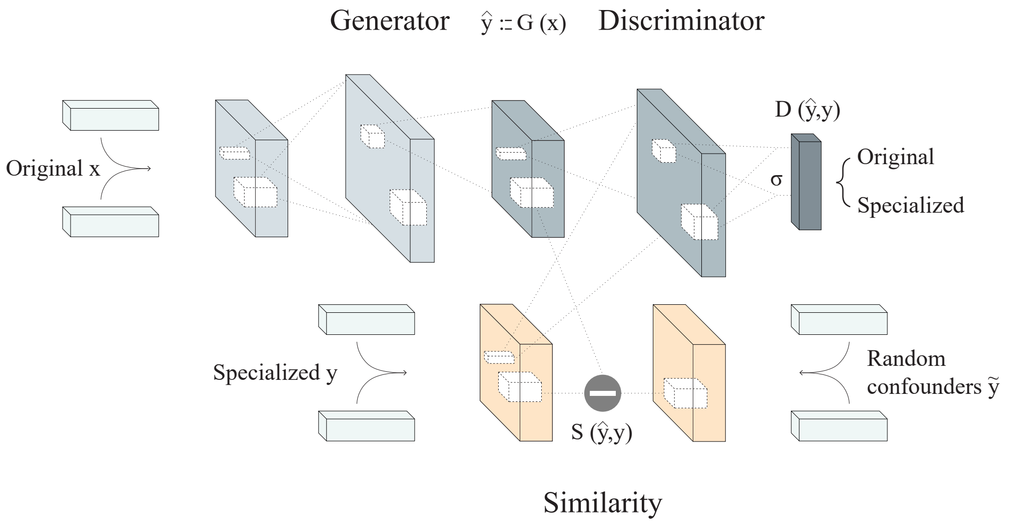

Figure 2: Architecture of the AuxGAN: an adversarial generator-discriminator loss (above) is combined with a max-margin L_2-distance loss with random confounders (below).

Objective Function. The pairs of original and specialized embeddings for seen words allow to train the global mapping function. In principle, this can be any differentiable parametrized function $G(\mathbf{x};\theta_G)$. Vulić et al. (2018) showed that non-linear functions ensure a better mapping than linear transformations which seem inadequate to mimic the complex perturbations of the specialization process, guided by possibly millions of pairwise constraints. Our preliminary experiments corroborate this intuition. Thus, in this work we also opt for implementing $G(\mathbf{x};\theta_G)$ as a deep neural network. Each of the l hidden layers of size h non-linearly transforms its input. The output layer is a linear transformation into the prediction $\hat{\mathbf{y}} \in \mathcal{R}^d$.

The parameters θ_G are learned by minimizing the L_2 distance between the training pairs. In particular, the loss is a contrastive margin-based ranking loss with negative sampling (MM) as proposed by Weston et al. (2011, *inter alia*). The gist of this loss is that the first component increases the cosine similarity *cos* of predicted and initially specialized vectors of the same word up to a margin δ_{MM}. On the other hand, the second component encourages the predicted vectors to distance themselves from k *random confounders*. These are negative examples sampled uniformly from the batch $\mathcal{B}$ excluding the current vector:

$$\mathcal{L}_{MM} = \sum_{i=1}^{||\mathcal{V}_s||} \sum_{j=1|j\neq i}^{k} \tau[\delta_{MM} - \cos(G(\mathbf{x}_i^{(s)};\theta_G), \mathbf{y}_i^{(s)}) + \\ + \cos(G(\mathbf{x}_i^{(s)};\theta_G), \mathbf{y}_j^{(s)})] \quad (4)$$

One of the original contributions of this work is combining the L_2 distance with an adversarial loss, resulting in an auxiliary-loss Generative Adversarial Network (AuxGAN) as shown in Figure 2. The role of the adversarial component, as mentioned above, is to 'soften' the mapping and guarantee realistic outputs from the target distribution.

The mapping can be considered a generator $G(\mathbf{x}|\theta_G)$. On top of this, a discriminator $D(\mathbf{x}|\theta_D)$, implemented also as a multi-layer neural net, tries to distinguish whether a vector is sampled from the predicted vectors or the AR-specialized vectors. Its output layer performs binary classification through *softmax*. The objective minimizes the loss $\mathcal{L}_D$:

$$\mathcal{L}_D = -\sum_{i=1}^{n} \log P(\text{specialized} = 0|G(\mathbf{x}_i;\theta_G);\theta_D) - \\ - \sum_{i=1}^{m} \log P(\text{specialized} = 1|\mathbf{y}_i;\theta_D) \quad (5)$$

In a two-player game (Goodfellow et al., 2014), the generator is trained to fool the discriminator by maximizing $\log(1 - P(0|G(\mathbf{x}_i;\theta_G);\theta_D))$. However, to avoid vanishing gradients of G early on, the loss $\mathcal{L}_G$ is reformulated by swapping the labels of Eq. (5) as follows:

$$\mathcal{L}_G = -\sum_{i=1}^{n} \log P(\text{specialized} = 1|G(\mathbf{x}_i;\theta_G);\theta_D) - \\ - \sum_{i=1}^{m} \log P(\text{specialized} = 0|\mathbf{y}_i;\theta_D) \quad (6)$$

During the optimization procedure through stochastic gradient descent, we alternate among s steps for

$\mathcal{L}_D$, one step for $\mathcal{L}_G$, and one step for $\mathcal{L}_{MM}$ to avoid the overfitting of D. The reason why $s \geq 1$ is that D can be kept close to a minimum of its loss function by updating G less frequently.

2.3 Zero-shot Transfer to Other Languages

Once the AuxGAN has learned a global mapping function $G(\mathbf{x}; \theta_G)$ in a resource-rich language, it can be directly applied to unseen words. In this work, we propose a method to additionally post-specialize the whole vocabulary $\mathcal{V}_t$ of a resource-poor target language. We assume a real-world scenario where no target language constraints are available to specialize it directly.

What is more, we assume that no bilingual data or dictionaries are available either. Hence, we rely on unsupervised cross-lingual word embedding induction, and in particular on Conneau et al. (2018)'s method. By virtue of these assumptions, there is no limitation to the range of potential target languages that can be specialized. Incidentally, please note that the proposed transfer method is equally applicable on top of other cross-lingual word embedding induction methods. These may require more bilingual supervision to learn the cross-lingual vector space.[3]

After learning the shared cross-lingual word embedding space in an unsupervised fashion (Conneau et al., 2018), the global post-specialization function learnt on the seen source language vectors is applied to the target language vectors, since they lie in the same shared space (see Figure 1 again). By virtue of the transfer, linguistic constraints in the source language can enhance the distributional vectors of target language vocabularies.

Conneau et al. (2018) learn a shared cross-lingual vector space as follows. They first learn a coarse initial mapping between two monolingual embedding spaces in two different languages through a GAN where the generator is a linear transformation with an orthogonal matrix $\hat{\mathbf{W}}$. Its loss is identical to Eq. (5) and Eq. (6), but unlike our AuxGAN model it discriminates between embeddings drawn from the source language and the target language distributions. Using the shared space, they extract for each source vector the closest target vector according to a distance metric designed to mitigate the hubness problem (Radovanović et al., 2010), the Cross-Domain Similarity Local Scaling (CSLS).

This creates a bilingual synthetic dictionary that allows to further refine the coarse initial mapping. In particular, the optimal parameters for the linear mapping minimizing the L_2-distance between source-target pairs are provided by the closed-form Procrustes solution (Schönemann, 1966) based on singular value decomposition (SVD):

$$\hat{\mathbf{W}} = \arg\min_W ||\mathbf{W}\,\mathbf{X}_t - \mathbf{X}_s||_F = \mathbf{U}\mathbf{V}^\top$$
$$\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top = \mathrm{SVD}(\mathbf{X}_t\mathbf{X}_s^\top) \quad (7)$$

where $||\cdot||_F$ is the Frobenius norm. After mapping the original target embeddings into the shared space with this method, we post-specialize them with the function outlined in §2.2, learnt on the source language. This yields the specialized target vectors $\hat{\mathbf{Y}}_t = G(\hat{\mathbf{W}}\,\mathbf{X}_t; \theta_G)$.

3 Experimental Setup

Distributional Vectors. We estimate the robustness of adversarial post-specialization by experimenting with three widely used collections of distributional English vectors. 1) SGNS-W2 vectors are trained on the cleaned and tokenized Polyglot Wikipedia (Al-Rfou et al., 2013) using Skip-Gram with Negative Sampling (SGNS) (Mikolov et al., 2013) by Levy and Goldberg (2014) with bag-of-words contexts (window size is 2). 2) GLOVE-CC are GloVe vectors trained on the Common Crawl (Pennington et al., 2014). 3) FASTTEXT are vectors trained on Wikipedia with a SGNS variant that builds word vectors by summing the vectors of their constituent character n-grams (Bojanowski et al., 2017). All vectors are 300-dimensional.[4]

Constraints and Initial Specialization. We experiment with the sets of linguistic constraints used in prior work (Zhang et al., 2014; Ono et al., 2015; Vulić et al., 2018). These constraints, extracted from WordNet (Fellbaum, 1998) and Roget's Thesaurus (Kipfer, 2009), comprise a total of 1,023,082 synonymy/ATTRACT word pairs and 380,873 antonymy/REPEL pairs.

Note that the sets of constraints cover only a fraction of the full distributional vocabulary, providing direct motivation for post-specialization methods

---

[3] See the recent survey papers on cross-lingual word embeddings and their typology (Upadhyay et al., 2016; Vulić and Korhonen, 2016; Ruder et al., 2017)

[4] Experiments with other standard word vectors, such as CONTEXT2VEC (Melamud et al., 2016a) and dependency-based embeddings (Bansal et al., 2014) show similar trends and lead to same conclusions.

which are able to specialize the full vocabulary. For instance, only 15.3% of the SGNS-W2 vocabulary words are seen words present in the constraints.[5]

The constraints are initially injected into the distributional vector space (see Figure 1 again) using ATTRACT-REPEL, a state-of-the-art specialization model, for which we adopt the original suggested model setup (Mrkšić et al., 2017).[6] Hyperparameter values are set to: $\delta_A = 0.6$, $\delta_R = 0.0$, $\lambda_P = 10^{-9}$. The models are trained for 5 epochs with Adagrad (Duchi et al., 2011), with batch sizes set to $k_A = k_R = 50$, again as in the original work.

AuxGAN Setup and Hyper-Parameters. Both the generator and the discriminator are feed-forward nets with $l = 2$ hidden layers, each of size $h = 2048$, and LeakyReLU as non-linear activation (Maas et al., 2013). The dropout for the input and hidden layers of the generator is 0.2 and for the input layer of the discriminator 0.1. In evaluation, the noise is blanketed out in order to ensure a deterministic mapping (Isola et al., 2017). Moreover, we smooth the golden labels for prediction by a factor of 0.1 to make the model less vulnerable to adversarial examples (Szegedy et al., 2016).

We train our model with SGD for 10 epochs of 1 million iterations each, feeding mini-batches of size 32. For each pair in a batch we generate 25 negative examples; $s = 5$ (see §2.2). As a way to normalize the mini-batches (Salimans et al., 2016), these are constructed to contain exclusively either original or specialized vectors. At each epoch, the initial learning rate of 0.1 is decayed by a factor of 0.98, or 0.5 if the score on the validation set (computed as the average cosine similarity between the predicted and AR-specialized embeddings)[7] has not increased. The hyper-parameters k and δ_{MM} are tuned via grid search on the validation set.

Zero-Shot Specialization Setup. The GAN discriminator for learning a shared cross-lingual vector space (see §2.3) has hyper-parameters identical to the AuxGAN. The generator instead is a linear layer initialized as an identity matrix and enforced to lie on the manifold of orthogonal matrices during training (Cisse et al., 2017). No dropout is used. The unsupervised validation metric for early stopping is the cosine distance between dictionary pairs extracted with the CSLS similarity metric.

4 Results and Discussion

4.1 Word Similarity

Evaluation Setup. We first evaluate adversarial post-specialization intrinsically, using two standard word similarity benchmarks for English: SimLex-999 (Hill et al., 2015) and SimVerb-3500 (Gerz et al., 2016), a dataset containing human similarity ratings for 3,500 verb pairs.[8] The evaluation measure is Spearman's ρ rank correlation between gold and predicted word pair similarity scores.

We evaluate word vectors in two settings, similar to Vulić et al. (2018). **a)** In the synthetic DISJOINT setting, we discard all linguistic constraints that contain any of the words found in SimLex or SimVerb. This means that all test words from SimLex and SimVerb are effectively unseen words, and through this setting we are able to *in vitro* evaluate the model's ability to generalize the specialization function to unseen words. **b)** In the FULL setting we leverage all constraints. This is a standard "real-life" scenario where some test words do occur in the constraints, while the mapping is learned for the remaining words. We use the FULL setting in all subsequent downstream applications (§4.2).

We compare our model to ATTRACT-REPEL (AR), which specializes only the vectors of words occurring in the constraints. We also provide comparisons to a post-specialization model of Vulić et al. (2018) which specializes the full vocabulary, but substitutes the AuxGAN architecture from §2.2 with a deep 5-layer feed-forward neural net also based on the max-margin loss (see Eq. (4)) to learn the mapping function (POST-DFFN).

Results and Analysis. The results are summarized in Table 1. The scores suggest that the proposed adversarial post-specialization model is universally useful and robust: we observe gains over input distributional word vectors for all three vector collections. The results in the DISJOINT setting illustrate the core limitation of the initial specialization/post-processing models and indicate the extent of improvement achieved when generalizing the specialization function to unseen words

---

[5] The respective coverage for the 200K most frequent GLOVE-CC and FASTTEXT words is only 13.3% and 14.6%.

[6] `https://github.com/nmrksic/attract-repel`

[7] The score is computed as the average cosine similarity between the original and specialized embeddings.

[8] Unlike WordSim-353 (Finkelstein et al., 2002) or MEN (Bruni et al., 2014), SimLex and SimVerb provide explicit guidelines to discern between true semantic similarity and (more broad) conceptual relatedness, so that related but non-similar words (e.g. *tiger* and *jungle*) have a low rating.

	Setting: DISJOINT						Setting: FULL					
	GLOVE-CC		FASTTEXT		SGNS-W2		GLOVE-CC		FASTTEXT		SGNS-W2	
	SL	SV	SL	SV	SL	SV	SL	SV	SL	SV	SL	SV
Distributional (X)	.407	.280	.383	.247	.414	.272	.407	.280	.383	.247	.414	.272
Specialized: ATTRACT-REPEL	.407	.280	.383	.247	.414	.272	.781	.761	.764	.744	.778	.761
Post-Specialized: POST-DFFN	.645	.531	.503	.340	.553	.430	.785	.764	.768	.745	.781	.763
Post-Specialized: AUXGAN	**.652**	**.552**	**.513**	**.394**	**.581**	**.434**	.789	.764	.766	.741	.782	.762

Table 1: Spearman's ρ correlation scores for three standard English distributional vectors spaces on English SimLex-999 (SL) and SimVerb-3500 (SV). POST-DFFN (Vulić et al., 2018) uses a deep non-linear feed-forward network to learn the mapping function f. AUXGAN is our adversarial model (see §2.2).

through adversarial post-specialization. Moreover, the scores suggest that the more sophisticated adversarial post-specialization method (AUXGAN) outperforms POST-DFFN across a large number of experimental runs, verifying its effectiveness.

We observe only modest and inconsistent gains over ATTRACT-REPEL and POST-DFFN in the FULL setting. However, the explanation of this finding is straightforward: 99.2% of SimLex words and 99.9% of SimVerb words are present in the external constraints, making this an unrealistic evaluation scenario. The usefulness of the initial ATTRACT-REPEL specialization is less pronounced in real-life downstream applications in which such high coverage cannot be guaranteed, as shown in §4.2.

4.2 Downstream Tasks

We next evaluate the embedding spaces specialized with the AuxGAN method in two tasks in which discerning semantic similarity from semantic relatedness is crucial: lexical text simplification (LS) and dialog state tracking (DST).

4.2.1 Lexical Text Simplification

The goal of lexical simplification is to replace complex words (typically words that are used less often in language and are therefore less familiar to readers) with their simpler synonyms, without infringing the grammaticality and changing the meaning of the text. Replacing complex words with related words instead of true synonyms affects the original meaning (e.g., *Ferrari pilot Vettel* vs *Ferrari airplane Vettel*) and often yields ungrammatical text (e.g., *they drink all pizzas*).

LS Using Word Vectors. We use Light-LS, a publicly available LS tool based on word embeddings (Glavaš and Štajner, 2015). Light-LS generates and then ranks substitution candidates based on similarity in the input word vector space. The quality of the space thus directly affects LS performance: by plugging any word vector space into Light-LS, we extrinsically evaluate that embedding space for LS. Furthermore, the better the embedding space captures true semantic similarity, the better the substitutions made by Light-LS.

	GLOVE-CC	FASTTEXT	SGNS-W2
Vector space	Acc	Acc	Acc
Distributional	.660	.578	.560
Specialized: AR	.676	.698	.644
Post-Specialized:			
POST-DFFN	**.723**	.723	.709
AUXGAN	.717	**.739**	**.721**

Table 2: Lexical simplification results for three (post-specialized) distributional spaces.

Evaluation Setup. We use the standard LS dataset of Horn et al. (2014). It contains 500 sentences with indicated complex words (one word per sentence) that have to be substituted with simpler synonyms. For each word, simplifications were crowdsourced from 50 human annotators. Following prior work (Horn et al., 2014; Glavaš and Štajner, 2015), we evaluate the performance of Light-LS using the metric that quantifies both the quality and the frequency of word replacements: *Accurracy (Acc)* metric is the number of correct simplifications made divided by the total number of complex words.

Results and Analysis. Scores for all three pretrained vector spaces are shown in Table 2. Similar to the word similarity task, embedding spaces produced with post-specialization models outperform the vectors produced with AR and original distributional vectors. The gains are now more pronounced in the real-life FULL setup, as only 59.6 % of all indicated complex words and substitution candidates from the LS dataset are covered in the external con-

GLOVE-CC word vectors	JGA
Distributional	.797
Specialized: ATTRACT-REPEL	.817
Post-Specialized: POST-DFFN	.829
Post-Specialized: AUXGAN	**.836**

Table 3: English DST performance (joint goal accuracy). GLOVE-CC word vectors.

straints. Adversarial post-specialization (AUXGAN) has a slight edge over the post-specialization with a simple feed-forward network (POST-DFFN) for FASTTEXT and SGNS-W2 embeddings, but not for GLOVE-CC vectors. In general, the fact that both post-specialization methods outperform ATTRACT-REPEL by a wide margin shows the importance of specializing the full word vector space for downstream NLP applications.

4.2.2 Dialog State Tracking

Finally, we evaluate the importance of full-vocabulary (adversarial) post-specialization in another language understanding task: dialog state tracking (DST) (Henderson et al., 2014; Williams et al., 2016), which is a standard task to measure the impact of specialization in prior work (Mrkšić et al., 2017). A DST model is typically the first component of a dialog system pipeline (Young, 2010), tasked with capturing user's goals and updating the dialog belief state at each dialog turn. Distinguishing similarity from relatedness is crucial for DST (e.g., a dialog system should not recommend an *"expensive restaurant in the west"* when asked for an *"affordable pub in the north"*).

Evaluation Setup. To evaluate the effects of specialized word vectors on DST, following prior work we utilize the Neural Belief Tracker (NBT), a statistical DST model that makes inferences purely based on pre-trained word vectors (Mrkšić et al., 2017).[9] Again, as in prior work the DST evaluation is based on the Wizard-of-Oz (WOZ) v2.0 dataset (Wen et al., 2017; Mrkšić et al., 2017), comprising 1,200 dialogues split into training (600 dialogues), development (200), and test data (400). We report the standard DST metric: *joint goal accuracy (JGA)*, the proportion of dialog turns where all the user's search goal constraints were correctly identified, computed as average over 5 NBT runs.

	Similarity (ρ)		LS (Acc)		DST (JGA)	
Vector space	IT	DE	IT	DE	IT	DE
Distrib.	.297	.417	.308	-	.681	.621
AUXGAN	**.431**	**.525**	**.392**	-	**.714**	**.651**

Table 4: Results of zero-shot specialization applied to IT and DE FASTTEXT distributional vectors.

Results and Analysis. We show English DST performance in the FULL setting in Table 3. Only NBT performance with GLOVE-CC vectors is reported for brevity, as similar performance gains are observed with the other two pre-trained vector collections. The results confirm our findings established in the other two tasks: **a)** initial AR specialization of distributional vectors is useful, but **b)** it is crucial to specialize the full vocabulary for improved performance (e.g., 57% of all WOZ words are present in the constraints), and **c)** the more sophisticated AUXGAN model yields additional gains.

4.3 Cross-Lingual Zero-Shot Specialization

Evaluation Setup. Large collections of linguistic constraints do not exist for many languages. Therefore, we test if the specialization knowledge from a resoure-rich language (i.e., English) can be transferred to resource-lean target languages (see §2.3). We simulate resource-lean scenarios using two target languages: Italian (IT) and German (DE).[10] We evaluate zero-specialized IT and DE FASTTEXT vectors, using English FASTTEXT vectors as the source, on the same three tasks as before. We report the same evaluation measures, using the following evaluation data: 1) IT and DE SimLex-999 datasets (Leviant and Reichart, 2015) for word similarity; 2) IT lexical simplification data (SIMPITIKI) (Tonelli et al., 2016); 3) IT and DE WOZ data (Mrkšić et al., 2017) for DST.

Results and Analysis. The results are summarized in Table 4. The gains over the original distributional vectors are substantial across all three tasks and for both languages. This finding indicates that the semantic content of distributional vectors can be enriched even for languages without any readily available lexical resources.

The gap between performances of language transfer and the monolingual setting is explained

---

[9] `https://github.com/nmrksic/neural-belief-tracker`; For full model details, we refer the reader to the original paper.

[10] Note that the two languages are not resource-poor, but we treat them as such in our experiments. This choice of languages was determined by the availability of high-quality evaluation data to measure the effects of zero-shot specialization.

by the noise introduced by the bilingual vector alignment and the different ways concepts are lexicalized across languages, as studied by semantic typology (Ponti et al., 2018). Nonetheless, in the long run, these transfer results hold promise to support the specialization of vector spaces even for resource-lean languages, and their applications.

5 Related Work

Vector Space Specialization. Specialization methods embed external information into vector spaces. Some of them integrate external linguistic constraints into distributional training and *jointly* optimize distributional and non-distributional objectives: they modify the prior or the regularization (Yu and Dredze, 2014; Xu et al., 2014; Bian et al., 2014; Kiela et al., 2015), or use a variant of the SGNS-style objective (Liu et al., 2015; Ono et al., 2015; Osborne et al., 2016).

Other models inject external knowledge from available lexical resources (e.g., WordNet, PPDB) into pre-trained word vectors as a *post-processing step* (Faruqui et al., 2015; Rothe and Schütze, 2015; Wieting et al., 2015; Nguyen et al., 2016; Mrkšić et al., 2016; Cotterell et al., 2016; Mrkšić et al., 2017). They offer a portable, flexible, and lightweight approach to incorporating external knowledge into *arbitrary* vector spaces, outperforming less versatile joint models and yielding state-of-the-art results on language understanding tasks (Mrkšić et al., 2016; Kim et al., 2016; Vulić et al., 2017). By design, these methods fine-tune only vectors of words seen in external resources.

Vulić et al. (2018) suggest that specializing the full vocabulary is beneficial for downstream applications. Comparing to their work, we show that a more sophisticated adversarial post-specialization can yield further gains across different tasks and boost full-vocabulary specialization in resource-lean settings through cross-lingual transfer.

Generative Adversarial Networks. GANs were originally devised to generate images from input noise variables (Goodfellow et al., 2014). The generation process is typically conditioned on discrete labels or data from other modalities, such as text (Mirza and Osindero, 2014). Otherwise, the condition can take the form of real data in input rather than (or in addition to) noise: in this case, the generator parameters are better conceived as a mapping function. For instance, it can bridge between pixel-to-pixel (Isola et al., 2017) or character-to-pixel (Reed et al., 2016) transformations.

The GAN objective can be mixed with more traditional loss functions: in these cases, apart from trying to fool the discriminator, the generator also minimizes the distance between input and target data (Pathak et al., 2016; Li and Wand, 2016; Ledig et al., 2017). The distance can be formulated as the mean squared error between the input and the target (Pathak et al., 2016), their feature maps (Li and Wand, 2016), both (Zhu et al., 2016), or a loss calculated on feature maps of a deep convolutional network (Ledig et al., 2017).

In the textual domain, adversarial models have been proven to support domain adaptation (Ganin et al., 2016) and language transfer (Chen et al., 2016) by learning domain/language-invariant latent features. Adversarial training also powers unsupervised mapping between monolingual vector spaces to learn cross-lingual word embeddings (Zhang et al., 2017; Conneau et al., 2018). In this work, we show how to apply adversarial techniques to the problem of vector specialization, which has a substantial impact on language understanding tasks.

6 Conclusion and Future Work

We have presented adversarial post-specialization, a novel model supported by adversarial training which specializes word vectors for the full vocabulary of the input distributional vector space, including words unseen in external lexical resources. We have also introduced a method for zero-shot specialization of word vectors in languages without any external resources. The benefits of adversarial post-specialization and its zero-shot transfer have been demonstrated across three tasks (word similarity, lexical text simplification, and dialog state tracking) and for three languages.

In future work, we will explore more sophisticated adversarial models such as Cycle-GAN (Zhu et al., 2017). Moreover, we will experiment with bootstrapping approaches to extract new lexical constraints from post-specialized embeddings. We also plan to extend the method to asymmetric relations (e.g., hypernymy) and to more target (resource-lean) languages. The code is available at `https://github.com/cambridgeltl/adversarial-postspec`.

Acknowledgements

This work is supported by the ERC Consolidator Grant LEXICAL (no 648909).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*, pages 183–192.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of ACL*, pages 809–815.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Proceedings of ECML-PKDD*, pages 132–148.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. 2017. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of ICML*, pages 854–863.

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

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proceedings of ICLR (Conference Track)*.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of ACL*, pages 1651–1660.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Manaal Faruqui. 2016. *Diverse Context for Learning Word Representations*. Ph.D. thesis, Carnegie Mellon University.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*, pages 1606–1615.

Christiane Fellbaum. 1998. *WordNet*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):59:1–59:35.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*, pages 758–764.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of EMNLP*, pages 2173–2182.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of ACL*, pages 63–68.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*, pages 2672–2680.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Matthew Henderson, Blaise Thomson, and Jason D. Wiliams. 2014. The Second Dialog State Tracking Challenge. In *Proceedings of SIGDIAL*, pages 263–272.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using Wikipedia. In *Proceedings of ACL*, pages 458–463.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of CVPR*, pages 1125–1134.

Sujay Kumar Jauhar, Chris Dyer, and Eduard H. Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of NAACL*, pages 683–693.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of EMNLP*, pages 2044–2048.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016. Intent detection using semantically enriched word embeddings. In *Proceedings of SLT*.

Barbara Ann Kipfer. 2009. *Roget's 21st Century Thesaurus (3rd Edition)*. Philip Lief Group.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of CVPR*, pages 4681–4690.

Ira Leviant and Roi Reichart. 2015. Separated by an un-common language: Towards judgment language informed vector space modeling. *CoRR*, abs/1508.00106.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*, pages 302–308.

Chuan Li and Michael Wand. 2016. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proceedings of ECCV*, pages 702–716.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of ACL*, pages 1501–1511.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of ICML*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016a. Context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of CoNLL*, pages 51–61.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016b. The role of context types and dimensionality in learning word embeddings. In *Proceedings of NAACL-HLT*, pages 1030–1040.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*, pages 1777–1788.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT*, pages 142–148.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the ACL*, 5:309–324.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of EMNLP*, pages 233–243.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of ACL*, pages 454–459.

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Proceedings of NIPS*, pages 6341–6350.

Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of ICML*, pages 2642–2651.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of NAACL-HLT*, pages 984–989.

Dominique Osborne, Shashi Narayan, and Shay Cohen. 2016. Encoding prior knowledge with eigenword embeddings. *Transactions of the ACL*, 4:417–430.

Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. 2016. Context encoders: Feature learning by inpainting. In *Proceedings of CVPR*, pages 2536–2544.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Edoardo Maria Ponti, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina

Shutova, and Anna Korhonen. 2018. Modeling language variation and universals: A survey on typological linguistics for natural language processing. *arXiv preprint arXiv:1807.00914*.

Milos Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. In *Proceedings of ICML*, pages 1060–1069.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of ACL*, pages 1793–1803.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual embedding models. *CoRR*, abs/1706.04902.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training GANs. In *Proceedings of NIPS*, pages 2234–2242.

Peter H Schönemann. 1966. A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31(1):1–10.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of CoNLL*, pages 258–267.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of CVPR*, pages 2818–2826.

Sara Tonelli, Alessio Palmero Aprosio, and Francesca Saltori. 2016. SIMPITIKI: A simplification corpus for Italian. In *Proceedings of CLiC-it*.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of ACL*, pages 1661–1670.

Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. 2018. Post-specialisation: Retrofitting vectors of words unseen in lexical resources. In *Proceedings of NAACL-HLT*, pages 516–527.

Ivan Vulić and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of ACL*, pages 247–257.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising word vectors for lexical entailment. In *Proceedings of NAACL-HLT*, pages 1134–1145.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of ACL*, pages 56–68.

Ivan Vulić, Roy Schwartz, Ari Rappoport, Roi Reichart, and Anna Korhonen. 2017. Automatic selection of context configurations for improved class-specific word representations. In *Proceedings of CoNLL*, pages 112–122.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. WSABIE: Scaling up to large vocabulary image annotation. In *Proceedings of IJCAI*, pages 2764–2770.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL*, 3:345–358.

Jason D. Williams, Antoine Raux, and Matthew Henderson. 2016. The Dialog State Tracking Challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of CIKM*, pages 1219–1228.

Steve Young. 2010. Cognitive User Interfaces. *IEEE Signal Processing Magazine*.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of ACL*, pages 545–550.

Jingwei Zhang, Jeremy Salwen, Michael Glass, and Alfio Gliozzo. 2014. Word semantic representations using bayesian probabilistic tensor factorization. In *Proceedings of EMNLP*, pages 1522–1531.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of ACL*, volume 1, pages 1959–1970.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. 2016. Generative visual manipulation on the natural image manifold. In *Proceedings of ECCV*, pages 597–613.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of CVPR*, pages 2223–2232.

Improving Cross-Lingual Word Embeddings by Meeting in the Middle

Yerai Doval
Grupo COLE, Escola Superior
de Enxeñaría Informática
Universidade de Vigo, Spain
`yerai.doval@uvigo.es`

Jose Camacho-Collados, Luis Espinosa-Anke
and Steven Schockaert
School of Computer Science and Informatics
Cardiff University, UK
`camachocolladosj@cardiff.ac.uk`
`espinosa-ankel@cardiff.ac.uk`
`schockaerts1@cardiff.ac.uk`

Abstract

Cross-lingual word embeddings are becoming increasingly important in multilingual NLP. Recently, it has been shown that these embeddings can be effectively learned by aligning two disjoint monolingual vector spaces through linear transformations, using no more than a small bilingual dictionary as supervision. In this work, we propose to apply an additional transformation after the initial alignment step, which moves cross-lingual synonyms towards a middle point between them. By applying this transformation our aim is to obtain a better cross-lingual integration of the vector spaces. In addition, and perhaps surprisingly, the monolingual spaces also improve by this transformation. This is in contrast to the original alignment, which is typically learned such that the structure of the monolingual spaces is preserved. Our experiments confirm that the resulting cross-lingual embeddings outperform state-of-the-art models in both monolingual and cross-lingual evaluation tasks.

1 Introduction

Word embeddings are one of the most widely used resources in NLP, as they have proven to be of enormous importance for modeling linguistic phenomena in both supervised and unsupervised settings. In particular, the representation of words in cross-lingual vector spaces (henceforth, *cross-lingual word embeddings*) is quickly gaining in popularity. One of the main reasons is that they play a crucial role in transferring knowledge from one language to another, specifically in downstream tasks such as information retrieval (Vulić and Moens, 2015b), entity linking (Tsai and Roth, 2016) and text classification (Mogadala and Rettinger, 2016), while at the same time providing improvements in multilingual NLP problems such as machine translation (Zou et al., 2013).

There exist different approaches for obtaining these cross-lingual embeddings. One of the most successful methodological directions, which constitutes the main focus of this paper, attempts to learn bilingual embeddings via a two-step process: first, word embeddings are trained on monolingual corpora and then the resulting monolingual spaces are aligned by taking advantage of bilingual dictionaries (Mikolov et al., 2013b; Faruqui and Dyer, 2014; Xing et al., 2015).

These *alignments* are generally modeled as linear transformations, which are constrained such that the structure of the initial monolingual spaces is left unchanged. This can be achieved by imposing an orthogonality constraint on the linear transformation (Xing et al., 2015; Artetxe et al., 2016). Our hypothesis in this paper is that such approaches can be further improved, as they rely on the assumption that the internal structure of the two monolingual spaces is identical. In reality, however, this structure is influenced by language-specific phenomena, e.g., the fact that Spanish distinguishes between masculine and feminine nouns (Davis, 2015) as well as the specific biases of the different corpora from which the monolingual spaces were learned. Because of this, monolingual embedding spaces are not isomorphic (Søgaard et al., 2018; Kementchedjhieva et al., 2018). On the other hand, simply dropping the orthogonality constraints leads to overfitting, and is thus not effective in practice.

The solution we propose is to start with existing state-of-the-art alignment models (Artetxe et al., 2017; Conneau et al., 2018), and to apply a further transformation to the resulting initial alignment. For each word w with translation w', this additional transformation aims to map the vector representations of both w and w' onto their average, thereby creating a cross-lingual vector space which intuitively corresponds to the average of the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 294–304
Brussels, Belgium, October 31 - November 4, 2018.

two aligned monolingual vector spaces. Similar to the initial alignment, this mapping is learned from a small bilingual lexicon.

Our experimental results show that the proposed additional transformation does not only benefit cross-lingual evaluation tasks, but, perhaps surprisingly, also monolingual ones. In particular, we perform an extensive set of experiments on standard benchmarks for bilingual dictionary induction and monolingual and cross-lingual word similarity, as well as on an extrinsic task: cross-lingual hypernym discovery.

Code and pre-trained embeddings to reproduce our experiments and to apply our model to any given cross-lingual embeddings are available at `https://github.com/yeraidm/meemi`.

2 Related Work

Bilingual word embeddings have been extensively studied in the literature in recent years. Their nature varies with respect to the supervision signals used for training (Upadhyay et al., 2016; Ruder et al., 2018). Some common signals to learn bilingual embeddings come from parallel (Hermann and Blunsom, 2014; Luong et al., 2015; Levy et al., 2017) or comparable corpora (Vulić and Moens, 2015a; Søgaard et al., 2015; Vulić and Moens, 2016), or lexical resources such as WordNet, ConceptNet or BabelNet (Speer et al., 2017; Mrksic et al., 2017; Goikoetxea et al., 2018). However, these sources of supervision may be scarce, limited to certain domains or may not be directly available for certain language pairs.

Another branch of research exploits pre-trained monolingual embeddings with weak signals such as bilingual lexicons for learning bilingual embeddings (Mikolov et al., 2013b; Faruqui and Dyer, 2014; Ammar et al., 2016; Artetxe et al., 2016). Mikolov et al. (2013b) was one of the first attempts into this line of research, applying a linear transformation in order to map the embeddings from one monolingual space into another. They also noted that more sophisticated approaches, such as using multilayer perceptrons, do not improve with respect to their linear counterparts. Xing et al. (2015) built upon this work by normalizing word embeddings during training and adding an orthogonality constraint. In a complementary direction, Faruqui and Dyer (2014) put forward a technique based on canonical correlation analysis to obtain linear mappings for both monolingual embedding spaces into a new shared space. Artetxe et al. (2016) proposed a similar linear mapping to Mikolov et al. (2013b), generalizing it and providing theoretical justifications which also served to reinterpret the methods of Faruqui and Dyer (2014) and Xing et al. (2015). Smith et al. (2017) further showed how orthogonality was required to improve the consistency of bilingual mappings, making them more robust to noise. Finally, a more complete generalization providing further insights on the linear transformations used in all these models can be found in Artetxe et al. (2018a).

These approaches generally require large bilingual lexicons to effectively learn multilingual embeddings (Artetxe et al., 2017). Recently, however, alternatives which only need very small dictionaries, or even none at all, have been proposed to learn high-quality embeddings via linear mappings (Artetxe et al., 2017; Conneau et al., 2018). More details on the specifics of these two approaches can be found in Section 3.1. These models have in turn paved the way for the development of machine translation systems which do not require any parallel corpora (Artetxe et al., 2018b; Lample et al., 2018). Moreover, the fact that such approaches only need monolingual embeddings, instead of parallel or comparable corpora, makes them easily adaptable to different domains (e.g., social media or web corpora).

In this paper we build upon these state-of-the-art approaches by applying an additional transformation, which aims to map each word and its translation onto the average of their vector representations. This strategy bears some resemblance with the idea of learning meta-embeddings (Yin and Schütze, 2016). Meta-embeddings are vector space representations which aggregate several pre-trained word embeddings from a given language (e.g., trained using different corpora and/or different word embedding models). Empirically it was found that such meta-embeddings can often outperform the individual word embeddings from which they were obtained. In particular, it was recently argued that word vector averaging can be a highly effective approach for learning such meta-embeddings (Coates and Bollegala, 2018). The main difference between such approaches and our work is that because we rely on a small dictionary, we cannot simply average word vectors, since for most words we do not know the corresponding translation. Instead, we train a regression model to predict this average word vector from the vector representation of the given word only, i.e., without using the vector representation of its translation.

3 Methodology

Our approach for improving cross-lingual embeddings consists of three main steps, where the first two steps are the same as in existing methods. In particular, given two monolingual corpora, a word vector space is first learned independently for each language. This can be achieved with common word embedding models, e.g., Word2vec (Mikolov et al., 2013a), GloVe (Pennington et al., 2014) or FastText (Bojanowski et al., 2017). Second, a linear alignment strategy is used to map the monolingual embeddings to a common bilingual vector space (Section 3.1). Third, a final transformation is applied on the aligned embeddings so the word vectors from both languages are refined and further integrated with each other (Section 3.2). This third step is the main contribution of our paper.

3.1 Aligning monolingual spaces

Once the monolingual word embeddings have been obtained, a linear transformation is applied in order to integrate them into the same vector space. This linear transformation is generally carried out using a supervision signal, typically in the form of a bilingual dictionary. In the following we explain two state-of-the-art models performing this linear transformation.

`VecMap` (Artetxe et al., 2017). `VecMap` uses an orthogonal transformation over normalized word embeddings. An iterative two-step procedure is also implemented in order to avoid the need of starting with a large seed dictionary (e.g., in the original paper it was tested with a very small bilingual dictionary of just 25 pairs). In this procedure, first, the linear mapping is estimated using a small bilingual dictionary, and then, this dictionary is augmented by applying the learned transformation to new words from the source language. Lastly, the process is repeated until some convergence criterion is met.

`MUSE` (Conneau et al., 2018). In this case, the transformation matrix is learned through an iterative Procrustes alignment (Schönemann, 1966).[1] The anchor points needed for this alignment can be obtained either through a supplied bilingual dictionary or through an unsupervised model. This unsupervised model is trained using adversarial learning to obtain an initial alignment of the two monolingual spaces, which is then refined by the Procrustes alignment using the most frequent words as anchor points. A new distance metric for the embedding space, referred to as cross-domain similarity local scaling, is also introduced. This metric, which takes into account the nearest neighbors of both source and target words, was shown to better handle high-density regions of the space, thus alleviating the hubness problem of word embedding models (Radovanović et al., 2010; Dinu et al., 2015), which arises when a few points (known as hubs) become the nearest neighbors of many other points in the embedding space.

3.2 Meeting in the middle

After the initial alignment of the monolingual word embeddings, our proposed method leverages an additional linear model to refine the resulting bilingual word embeddings. This is because the methods presented in the previous section apply constraints to ensure that the structure of the monolingual embeddings is largely preserved. As already mentioned in the introduction, conceptually this may not be optimal, as embeddings for different languages and trained from different corpora can be expected to be structured somewhat differently. Empirically, as we will see in the evaluation, after applying methods such as `VecMap` and `MUSE` there still tend to be significant gaps between the vector representations of words and their translations. Our method directly attempts to reduce these gaps by moving each word vector towards the middle point between its current representation and the representation of its translation. In this way, by bringing the two monolingual fragments of the space closer to each other, we can expect to see an improved performance on cross-lingual evaluation tasks such as bilingual dictionary induction. Importantly, the internal structure of the two monolingual fragments themselves is also affected by this step. By aver-

---

[1] Very recently, Kementchedjhieva et al. (2018) showed that projecting both monolingual embedding spaces onto a third space (instead of directly onto each other) using a generalized Procrustes analysis facilitates the learning of alignments.

aging between the representations obtained from different languages, we hypothesize that the impact of language-specific phenomena and corpus specific biases will be reduced, thereby ending up with more "neutral" monolingual embeddings.

In the following, we detail our methodological approach. First, we leverage the same bilingual dictionary that was used to obtain the initial alignment (Section 3.1). Specifically, let $D = \{(w, w')\}$ be the given bilingual dictionary, where $w \in V$ and $w' \in V'$, with V and V' representing the vocabulary of the first and second language, respectively. For pairs $(w, w') \in D$, we can simply compute the corresponding average vector $\vec{\mu}_{w,w'} = \frac{\vec{v}_w + \vec{v}_{w'}}{2}$. Then, using the pairs in D as training data, we learn a linear mapping X such that $X\vec{v}_w \approx \vec{\mu}_{w,w'}$ for all $(w, w') \in D$. This mapping X can then be used to predict the averages for words outside the given dictionary. To find the mapping X, we solve the following least squares linear regression problem:

$$E = \sum_{(w,w') \in D} \|X\vec{w} - \vec{\mu}_{w,w'}\|^2 \qquad (1)$$

Similarly, for the other language, we separately learn a mapping X' such that $X'\vec{v}_{w'} \approx \vec{\mu}_{w,w'}$.

It is worth pointing out that we experimented with several variants of this linear regression formulation. For example, we also tried using a multilayer perceptron to learn non-linear mappings, and we experimented with several regularization terms to penalize mappings that deviate too much from the identity mapping. None of these variants, however, were found to improve on the much simpler formulation in (1), which can be solved exactly and efficiently. Furthermore, one may wonder whether the initial alignment is actually needed, since e.g., Coates and Bollegala (2018) obtained high-quality meta-embeddings without such an alignment set. However, when applying our approach directly to the initial monolingual non-aligned embedding spaces, we obtained results which were competitive but slightly below the two considered alignment strategies.

4 Evaluation

We test our bilingual embedding refinement approach on both intrinsic and extrinsic tasks. In Section 4.1 we describe the common training setup for all experiments and language pairs. The languages we considered are English, Spanish, Italian, German and Finnish. Throughout all the experiments we use publicly available resources in order to make comparisons and reproducibility of our experiments easier.

4.1 Cross-lingual embeddings training

Corpora. In our experiments we make use of web-extracted corpora. For English we use the 3B-word UMBC WebBase Corpus (Han et al., 2013), while we chose the Spanish Billion Words Corpus (Cardellino, 2016) for Spanish. For Italian and German, we use the itWaC and sdeWaC corpora from the WaCky project (Baroni et al., 2009), containing 2 and 0.8 billion words, respectively.[2] Lastly, for Finnish, we use the Common Crawl monolingual corpus from the Machine Translation of News Shared Task 2016[3], composed of 2.8B words. All corpora are tokenized and lowercased.

Monolingual embeddings. The monolingual word embeddings are trained with the Skipgram model from FastText (Bojanowski et al., 2017) on the corpora described above. The dimensionality of the vectors was set to 300, with the default FastText hyperparameters.

Bilingual dictionaries. We use the bilingual dictionaries packaged together by Artetxe et al. (2017), each one conformed by 5000 word translations. They are used both for the initial bilingual mappings and then again for our linear transformation.

Initial mapping. Following previous works, for the purpose of obtaining the initial alignment, English is considered as source language and the remaining languages are used as target. We make use of the open-source implementations of **`VecMap`**[4] (Artetxe et al., 2017) and **`MUSE`**[5] (Conneau et al., 2018), which constitute strong baselines for our experiments (cf. Section 3.1). Both of them were used with the recommended parameters and in their supervised setting, using the aforementioned bilingual dictionaries.

Meeting in the Middle. Then, once the initial cross-lingual embeddings are trained, and as explained in Section 3.2, we obtain our linear transformation by using the exact solution to the least

---

[2] UMBC, Spanish Billion-Words and ItWaC are the official corpora of the hypernym discovery SemEval task (Section 4.2.3) for English, Spanish and Italian, respectively.

[3] `http://www.statmt.org/wmt16/translation-task.html`

[4] `github.com/artetxem/vecmap`

[5] `github.com/facebookresearch/MUSE`

Model	EN-ES			EN-IT			EN-DE			EN-FI		
	$P@1$	$P@5$	$P@10$	$P@1$	$P@5$	$P@10$	$P@1$	$P@5$	$P@10$	$P@1$	$P@5$	$P@10$
VecMap	36.0	59.8	65.6	35.5	57.2	63.9	31.7	54.2	60.2	17.2	36.4	43.7
VecMap$_\mu$	**37.8**	**61.5**	**67.1**	**36.3**	**59.2**	**66.3**	**33.5**	**57.3**	**61.7**	**18.5**	**40.9**	**48.3**
MUSE	37.1	59.0	65.2	36.3	57.3	62.9	32.5	53.7	59.0	18.2	35.2	42.4
MUSE$_\mu$	**38.3**	**62.3**	**67.2**	**37.0**	**59.0**	**65.7**	**33.7**	**57.0**	**62.2**	**19.4**	**41.1**	**49.0**

Table 1: Bilingual dictionary induction results. Precision at k ($P@K$) performance for Spanish (ES), Italian (IT), German (DE) and Finnish (FI), using English (EN) as source language.

squares linear regression problem. To this end, we use the same bilingual dictionaries as in the previous step. Henceforth, we will refer to our transformed models as **VecMap**$_\mu$ and **MUSE**$_\mu$, depending on the initial mapping.

4.2 Experiments

We test our cross-lingual word embeddings in two intrinsic tasks, i.e., bilingual dictionary induction (Section 4.2.1) and word similarity (Section 4.2.2), and an extrinsic task, i.e., cross-lingual hypernym discovery (Section 4.2.3).

4.2.1 Bilingual dictionary induction

The dictionary induction task consists in automatically generating a bilingual dictionary from a source to a target language, using as input a list of words in the source language.

Experimental setting For this task, and following previous works, we use the English-Italian test set released by Dinu et al. (2015) and those released by Artetxe et al. (2017) for the remaining language pairs. These test sets have no overlap with respect to the training and development sets, and contain around 1900 entries each. Given an input word from the source language, word translations are retrieved through a nearest-neighbor search of words in the target language, using cosine distance. Note that this gives us a ranked list of candidates for each word from the source language. Accordingly, the performance of the embeddings is evaluated with the precision at k ($P@k$) metric, which evaluates for what percentage of test pairs, the correct answer is among the k highest ranked candidates.

Results As can be seen in Table 1, our refinement method consistently improves over the baselines (i.e., `VecMap` and `MUSE`) on all language pairs and metrics. The higher scores indicate that the two monolingual embedding spaces become more tightly integrated because of our additional transformation. It is worth highlighting here the case of English-Finnish, where the gains obtained in $P@5$ and $P@10$ are considerable. This might indicate that our approach is especially useful for morphologically richer languages such as Finnish, where the limitations of the previous bilingual mappings are most apparent.

Analysis When analyzing the source of errors in $P@1$, we came to similar conclusions as Artetxe et al. (2017).[6] Several source words are translated to words that are closely related to the one in the gold reference in the target language; e.g., for the English word *essentially* we obtain *básicamente* (*basically*) instead of *fundamentalmente* (*fundamentally*) in Spanish, both of them closely related, or the closest neighbor for *dirt* being *mugre* (*dirt*) instead of *suciedad* (*dirt*), which in fact was among the five closest neighbors. We can also find multiple examples of the higher performance of our models compared to the baselines. For instance, in the English-Spanish cross-lingual models, after the initial alignment, we can find that *seconds* has *minutos* (*minutes*) as nearest neighbour, but after applying our additional transformation, *seconds* becomes closest to *segundos* (*seconds*). Similarly, *paint* initially has *tintado* (*tinted*) as the closest Spanish word, and then *pintura* (*paint*).

4.2.2 Word similarity

We perform experiments on both monolingual and cross-lingual word similarity. In monolingual similarity, models are tested in their ability to determine the similarity between two words in the same language, whereas in cross-lingual similarity the words belong to different languages. While in the monolingual setting the main objective is to test the quality of the monolingual subsets of the bilin-

---

[6] The results on this task are lower than those reported in Artetxe et al. (2017). This is due to the different corpora and embedding algorithms used to train the monolingual embeddings. In particular, they use corpora including Wikipedia, which is comparable across languages.

Model	`English`								`Spanish`				`Italian`				`German`					
	SemEval		**WordSim**		**SimLex**		**RG-65**		**SemEval**		**RG-65**		**SemEval**		**WordSim**		**SemEval**		**WordSim**		**RG-65**	
	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ
`VecMap`	74.1	73.9	67.9	67.0	42.0	40.7	77.8	**77.5**	70.0	71.4	86.6	88.0	67.2	**69.0**	64.0	66.9	70.1	70.1	72.7	72.2	80.2	79.7
`VecMap`$_\mu$	**75.0**	**74.8**	**70.5**	**70.1**	**43.8**	**41.8**	**78.0**	76.6	**71.5**	**72.1**	**87.6**	**89.4**	**68.4**	68.9	**65.3**	**67.3**	**70.9**	**70.7**	72.7	**72.4**	**81.0**	**81.3**
`MUSE`	74.2	74.2	68.3	67.6	42.6	41.5	**78.6**	**78.4**	70.5	71.9	86.6	88.3	67.4	**69.2**	64.1	66.9	69.8	69.8	**72.5**	**72.5**	80.3	**80.1**
`MUSE`$_\mu$	**75.0**	**74.8**	**70.8**	**70.4**	**44.2**	**42.4**	78.3	77.5	**71.8**	**72.3**	**87.7**	**89.3**	**68.6**	69.1	**65.3**	**67.2**	**70.4**	**70.2**	72.2	72.1	80.3	80.0

Table 2: Monolingual word similarity results. Pearson (r) and Spearman (ρ) correlation.

gual vector space, the cross-lingual setting constitutes a straightforward benchmark to test the quality of bilingual embeddings.

Experimental setting For monolingual word similarity we use the English SimLex-999 (Hill et al., 2015), and the language-specific versions of SemEval-17[7] (Camacho-Collados et al., 2017), WordSim-353[8] (Finkelstein et al., 2002), and RG-65 (Rubenstein and Goodenough, 1965). The corresponding cross-lingual datasets from SemEval-18, WordSim-353 and RG-65 were considered for the cross-lingual word similarity evaluation[9]. Cosine similarity is again used as comparison measure.

Results Tables 2 and 3 show the monolingual[10] and cross-lingual word similarity results[11], respectively. For both the monolingual and cross-lingual settings, we can notice that our models generally outperform the corresponding baselines. Moreover, in cases where no improvement is obtained, the differences tend to be minimal, with the exception of RG-65, but this is a very small test set for which larger variations can thus be expected. In contrast, there are a few cases where substantial gains were obtained by using our model. This is most notable for English WordSim and SimLex in the monolingual setting.

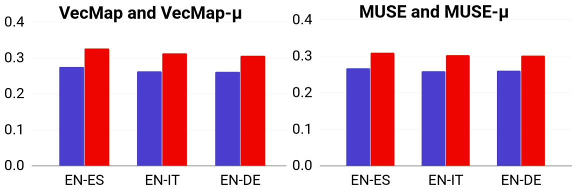

Figure 1: Comparative average similarity between `VecMap` and `MUSE` (blue) and our proposed model (red) on the SemEval cross-lingual similarity datasets.

Analysis In order to further understand the movements of the space with respect to the original `VecMap` and `MUSE` spaces, Figure 1 displays the average similarity values on the SemEval cross-lingual datasets (the largest among all benchmarks) of each model. As expected, the figure clearly shows how our model consistently brings the words from both languages closer on all language pairs. Furthermore, this movement is performed smoothly across all pairs, i.e., our model does not make large changes to specific words but rather small changes overall. This can be verified by inspecting the standard deviation of the difference in similarity after applying our transformation. These standard deviation scores range from 0.031 (English-Spanish for `VecMap`) to 0.039 (English-Italian for `MUSE`), which are relatively small given that the cosine similarity scale ranges from -1 to 1.

As a complement of this analysis we show some qualitative results which give us further insights on the transformations of the vector space after our average approximation. In particular, we analyze the reasons behind the higher quality displayed by our bilingual embeddings in monolingual settings. While `VecMap` and `MUSE` do not transform the initial monolingual spaces, our model transforms both spaces simultaneously. In this analysis we focus on the source language of our experiments (i.e., English). We found interesting patterns which are learned by our model and

---

[7] The original datasets of SemEval-17 contained also multiwords, but for consistency we use the version containing single words only.

[8] WordSim datasets consist of the similarity re-scoring for several languages of Leviant and Reichart (2015), downloaded from `http://leviants.com/ira.leviant/MultilingualVSMdata.html`

[9] The WordSim-353 and RG-65 cross-lingual datasets (Camacho-Collados et al., 2015) were downloaded at `http://lcl.uniroma1.it/similarity-datasets/`

[10] The English results correspond to the averaged performance of the English fragments of English-Spanish, English-Italian and English-German cross-lingual embeddings.

[11] The results of the original `VecMap` in cross-lingual similarity are comparable or better to those reported in Artetxe et al. (2017) on the three datasets used in their evaluation.

Model	English-Spanish				English-Italian				English-German					
	SemEval		RG-65		SemEval		WordSim		SemEval		WordSim		RG-65	
	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ
VecMap	71.7	**71.6**	82.1	82.4	69.6	69.6	60.2	**63.1**	71.6	71.3	64.1	**65.9**	78.1	78.8
VecMap$_\mu$	71.7	71.3	82.1	**82.8**	**70.2**	**69.9**	**61.3**	63.0	**72.0**	**71.5**	**64.2**	65.4	**78.6**	**79.7**
MUSE	72.0	**72.0**	81.9	82.3	69.4	69.4	59.9	62.7	70.4	70.1	63.5	65.1	78.4	79.5
MUSE$_\mu$	**72.2**	71.8	**82.3**	**82.5**	**70.5**	**70.1**	**61.2**	62.7	**71.9**	**71.4**	**64.1**	**65.3**	**78.8**	**80.5**

Table 3: Cross-lingual word similarity results. Pearson (r) and Spearman (ρ) correlation.

help understand these monolingual gains. For example, a recurring pattern is that words in English which are translated to the same word, or to semantically close words, in the target language end up closer together after our transformation. For example, in the case of English-Spanish the following pairs were among the pairs whose similarity increased the most by applying our transformation: *cellphone-telephone*, *movie-film*, *book-manuscript* or *rhythm-cadence*, which are either translated to the same word in Spanish (i.e., *teléfono* and *película* in the first two cases) or are already very close in the Spanish space. More generally, we found that word pairs which move together the most tend to be semantically very similar and belong to the same domain, e.g., *car-bicycle*, *opera-cinema*, or *snow-ice*.

4.2.3 Cross-lingual hypernym discovery

Modeling hypernymy is a crucial task in NLP, with direct applications in diverse areas such as semantic search (Hoffart et al., 2014; Roller and Erk, 2016), question answering (Prager et al., 2008; Yahya et al., 2013) or textual entailment (Geffet and Dagan, 2005). Hypernyms, in addition, are the backbone of lexical ontologies (Yu et al., 2015), which are in turn useful for organizing, navigating and retrieving online content (Bordea et al., 2016). Thus, we propose to evaluate the contribution of cross-lingual embeddings towards the task of hypernym discovery, i.e., given an input word (e.g., *cat*), retrieve or discover its most likely (set of) valid hypernyms (e.g., *animal*, *mammal*, *feline*, and so on). Intuitively, by leveraging a bilingual vector space condensing the semantics of two languages, one of them being English, the need for large amounts of training data in the target language may be reduced.

Experimental setting We follow Espinosa-Anke et al. (2016) and learn a (cross-lingual) linear transformation matrix between the hyponym and hypernym spaces, which is afterwards used to predict the most likely (set of) hypernyms, given an unseen hyponym. Training and evaluation data come from the SemEval 2018 Shared Task on Hypernym Discovery (Camacho-Collados et al., 2018). Note that current state-of-the-art systems aimed at modeling hypernymy (Shwartz et al., 2016; Bernier-Colborne and Barriere, 2018) combine large amounts of annotated data along with language-specific rules and cue phrases such as Hearst Patterns (Hearst, 1992), both of which are generally scarcely (if at all) available for languages other than English. Therefore, we report experiments with training data only from English (11,779 hyponym-hypernym pairs), and "enriched" models informed with relatively few training pairs (500, 1k and 2k) from the target languages. Evaluation is conducted with the same metrics as in the original SemEval task, i.e., Mean Reciprocal Rank (MRR), Mean Average Precision (MAP) and Precision at 5 (P@5). These measures explain a model's behavior from complementary prisms, namely how often at least one valid hypernym was highly ranked (MRR), and in cases where there is more than one correct hypernym, to what extent they were all correctly retrieved (MAP and P@5). Finally, as in the previous experiments, we report comparative results between our proposed models and the two competing baselines (`VecMap` and `MUSE`). As an additional informative baseline, we include the highest scoring unsupervised system at the SemEval task for both Spanish and Italian (`BestUns`), which is based on the distributional models described in Shwartz et al. (2017).

Results The results listed in Table 4 indicate several trends.[12] First and foremost, in terms of

---

[12] Note that this task is harder than hypernymy detection (Upadhyay et al., 2018). Hypernymy detection is framed as a binary classification task, while in hypernym discovery hypernyms have to be retrieved from the whole vocabulary.

Train data	Model	Spanish			Italian		
		MAP	MRR	P@5	MAP	MRR	P@5
-	**BestUns**	2.4	5.5	2.5	3.9	8.7	3.9
EN	**VecMap**	**6.4**	**16.5**	**6.0**	4.5	10.6	4.3
	VecMap$_\mu$	6.1	15.4	5.7	**5.6**	**13.3**	**5.4**
	MUSE	5.9	14.1	5.5	4.9	11.1	4.7
	MUSE$_\mu$	**6.2**	**14.8**	**5.8**	**5.1**	**11.7**	**4.9**
EN + 500	**VecMap**	**7.3**	**18.2**	**7.0**	6.1	14.0	5.8
	VecMap$_\mu$	7.0	17.6	6.6	**6.8**	**16.2**	**6.4**
	MUSE	6.4	15.9	6.1	5.3	12.0	5.0
	MUSE$_\mu$	**6.9**	**16.9**	**6.6**	**6.0**	**13.4**	**5.7**
EN + 1k	**VecMap**	**7.9**	19.2	**7.6**	7.0	16.4	6.6
	VecMap$_\mu$	7.8	19.2	7.4	**7.5**	**18.1**	**7.0**
	MUSE	7.2	17.3	6.9	6.2	13.8	5.8
	MUSE$_\mu$	**7.8**	**18.8**	**7.5**	**6.5**	**14.2**	**6.3**
EN + 2k	**VecMap**	8.0	19.1	7.7	8.2	19.1	7.5
	VecMap$_\mu$	**8.2**	**19.9**	**7.9**	**8.7**	**20.7**	**8.1**
	MUSE	7.2	17.2	6.8	7.2	15.8	7.0
	MUSE$_\mu$	**8.3**	**19.5**	**8.0**	**7.6**	**17.0**	**7.2**

Table 4: Results on the hypernym discovery task.

model-wise comparisons, we observe that our proposed alterations of both `VecMap` and `MUSE` improve their quality in a consistent manner, across most metrics and data configurations. In Italian our proposed model shows an improvement across all configurations. However, in Spanish `VecMap` emerges as a highly competitive baseline, with our model only showing an improved performance when training data in this language abounds (in this specific case there is an increase from 17.2 to 19.5 points in the MRR metric). This suggests that the fact that the monolingual spaces are closer in our model is clearly beneficial when hybrid training data is given as input, opening up avenues for future work on weakly-supervised learning. Concerning the other baseline, `MUSE`, the contribution of our proposed model is consistent for both languages, again becoming more apparent in the Italian split and in a fully cross-lingual setting, where the improvement in MRR is almost 3 points (from 10.6 to 13.3). Finally, it is noteworthy that even in the setting where no training data from the target language is leveraged, all the systems based on cross-lingual embeddings outperform the best unsupervised baseline, which is a very encouraging result with regards to solving tasks for languages on which training data is not easily accessible or not directly available.

Analysis A manual exploration of the results obtained in cross-lingual hypernym discovery reveals a systematic pattern when comparing, for example, `VecMap` and our model. It was shown in Table 4 that the performance of our model gradually increased alongside the size of the training data in the target language until surpassing `VecMap` in the most informed configuration (i.e., EN+2k). Specifically, our model seems to show a higher presence of generic words in the output hypernyms, which may be explained by these being closer in the space. In fact, out of 1000 candidate hyponyms, our model correctly finds *person* 143 times, as compared to the 111 of `VecMap`, and this systematically occurs with generic types such as *citizen* or *transport*. Let us mention, however, that the considered baselines perform remarkably well in some cases. For example, the English-only `VecMap` configuration (EN), unlike ours, correctly discovered the following hypernyms for *Francesc Macià* (a Spanish politician and soldier): *politician*, *ruler*, *leader* and *person*. These were missing from the prediction of our model in all configurations until the most informed one (EN+2k).

5 Conclusions and Future Work

We have shown how to refine bilingual word embeddings by applying a simple transformation which moves cross-lingual synonyms closer towards their average representation. Before applying this strategy, we start by aligning the monolingual embeddings of the two languages of interest. For this initial alignment, we have considered two state-of-the-art methods from the literature, namely `VecMap` (Artetxe et al., 2017) and `MUSE` (Conneau et al., 2018), which also served as our baselines. Our approach is motivated by the fact that these alignment methods do not change the structure of the individual monolingual spaces. However, the internal structure of embeddings is, at least to some extent, language-specific, and is moreover affected by biases of the corpus from which they are trained, meaning that after the initial alignment significant gaps remain between the representations of cross-lingual synonyms. We tested our approach on a wide array of datasets from different tasks (i.e., bilingual dictionary induction, word similarity and cross-lingual hypernym discovery) with state-of-the-art results.

This paper opens up several promising avenues for future work. First, even though both languages are currently being treated symmetrically, the initial monolingual embedding of one of the languages may be more reliable than that of the other. In such cases, it may be of interest to replace the vectors $\vec{\mu}_{w,w'}$ by a weighted average of the monolingual word vectors. Second, while we have only considered bilingual scenarios in this paper, our approach can naturally be applied to scenarios involving more languages. In this case, we would first choose a single target language, and obtain alignments between all the other languages and this target language. To apply our model, we can then simply learn mappings to predict averaged word vectors across all languages. Finally, it would also be interesting to use the obtained embeddings in downstream applications such as language identification or cross-lingual sentiment analysis, and extend our analysis to other languages, with a particular focus on morphologically-rich languages (after seeing our success with Finnish), for which the bilingual induction task has proved more challenging for standard cross-lingual embedding models (Søgaard et al., 2018).

Acknowledgments

Yerai Doval is supported by the Spanish Ministry of Economy, Industry and Competitiveness (MINECO) through projects FFI2014-51978-C2-2-R, TIN201785160C21-R and TIN201785160C22-R; the Spanish State Secretariat for Research, Development and Innovation (which belongs to MINECO) and the European Social Fund (ESF) under an FPI fellowship (BES-2015-073768) associated to project FFI2014-51978-C2-1-R; and by the Galician Regional Government under project ED431D 2017/12. Jose Camacho-Collados, Luis Espinosa-Anke and Steven Schockaert have been supported by ERC Starting Grant 637277.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018b. Unsupervised neural machine translation. In *Proceedings of the Sixth International Conference on Learning Representations*.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Gabriel Bernier-Colborne and Caroline Barriere. 2018. Crim at semeval-2018 task 9: A hybrid approach to hypernym discovery. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, New Orleans, Louisiana.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5(1):135–146.

Georgeta Bordea, Els Lefever, and Paul Buitelaar. 2016. Semeval-2016 task 13: Taxonomy extraction evaluation (texeval-2). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1081–1091.

Jose Camacho-Collados, Claudio Delli Bovi, Luis Espinosa-Anke, Sergio Oramas, Tommaso Pasini, Enrico Santus, Vered Shwartz, Roberto Navigli, and Horacio Saggion. 2018. SemEval-2018 Task 9: Hypernym Discovery. In *Proceedings of SemEval*, New Orleans, LA, United States.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A Framework for the Construction of Monolingual and Cross-lingual Word Similarity Datasets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing – Short Papers*, pages 1–7, Beijing, China.

Cristian Cardellino. 2016. Spanish Billion Words Corpus and Embeddings. *http://crscardellino.me/SBWCE/*.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding–computing meta-embeddings by averaging source word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 194–198.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proceedings of ICLR*.

Ernest Davis. 2015. Does the world look different in different languages? *Artif. Intell.*, 229:202–209.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of ICLR, Workshop track*.

Luis Espinosa-Anke, Jose Camacho-Collados, Claudio Delli Bovi, and Horacio Saggion. 2016. Supervised distributional hypernym discovery via domain adaptation. In *Proceedings of EMNLP*, pages 424–435.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

Lev Finkelstein, Gabrilovich Evgeniy, Matias Yossi, Rivlin Ehud, Solan Zach, Wolfman Gadi, and Ruppin Eytan. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 107–114. Association for Computational Linguistics.

Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. 2018. Bilingual embeddings with random walks over multilingual wordnets. *Knowledge-Based Systems*.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC EBIQUITY-CORE: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, pages 44–52.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING 1992*, pages 539–545.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 58–68.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Johannes Hoffart, Dragan Milchevski, and Gerhard Weikum. 2014. Stics: searching with strings, things, and cats. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 1247–1248. ACM.

Yova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing procrustes analysis for better bilingual dictionary induction. In *Proceedings of the Conference on Computational Natural Language Learning*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *Proceedings of the Sixth International Conference on Learning Representations*.

Ira Leviant and Roi Reichart. 2015. Separated by an un-common language: Towards judgment language informed vector space modeling. *arXiv preprint arXiv:1508.00106*.

Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 765–774.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Aditya Mogadala and Achim Rettinger. 2016. Bilingual word embeddings from parallel and non-parallel corpora for cross-language text classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 692–702.

Nikola Mrksic, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gai, Anna Korhonen, and Steve Young. 2017. Semantic Specialisation of Distributional Word Vector Spaces using Monolingual and Cross-Lingual Constraints. *TACL*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

John Prager, Jennifer Chu-Carroll, Eric W Brown, and Krzysztof Czuba. 2008. Question answering by predictive annotation. In *Advances in Open Domain Question Answering*, pages 307–347. Springer.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(Sep):2487–2531.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *Proceedings of EMNLP*, pages 2163–2172, Austin, Texas.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2018. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*.

Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of ACL*, Berlin, Germany.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of EACL*, Valencia, Spain. Association for Computational Linguistics.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of ICLR*.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual NLP. In *Proceedings of ACL*, pages 1713–1722.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. *arXiv preprint arXiv:1805.03620*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of AAAI*.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual wikification using multilingual embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 589–598.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1661–1670.

Shyam Upadhyay, Yogarshi Vyas, Marine Carpuat, and Dan Roth. 2018. Robust cross-lingual hypernymy detection using dependency context. In *Proceedings of NAACL*.

Ivan Vulić and Marie-Francine Moens. 2015a. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 719–725.

Ivan Vulić and Marie-Francine Moens. 2015b. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 363–372. ACM.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1107–1116. ACM.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. In *Proceedings of IJCAI*, pages 1390–1397.

Will Y. Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*, pages 1393–1398, Seattle, USA.

WikiAtomicEdits: A Multilingual Corpus of Wikipedia Edits for Modeling Language and Discourse

Manaal Faruqui* **Ellie Pavlick*** **Ian Tenney** **Dipanjan Das**
Google AI Language

Abstract

We release a corpus of 43 million *atomic edits* across 8 languages. These edits are mined from Wikipedia edit history and consist of instances in which a human editor has inserted a single contiguous phrase into, or deleted a single contiguous phrase from, an existing sentence. We use the collected data to show that the language generated during editing differs from the language that we observe in standard corpora, and that models trained on edits encode different aspects of semantics and discourse than models trained on raw, unstructured text. We release the full corpus as a resource to aid ongoing research in semantics, discourse, and representation learning.

1 Introduction

Written language often undergoes several rounds of revision as human authors determine exactly what information they want their words to convey. On Wikipedia, this process is carried out collectively by a large community at a rate of nearly two revisions per second (Yang et al., 2017). While Wikipedia's revision history contains arbitrarily complex edits, our corpus and analysis focuses on *atomic insertion edits*: instances in which an editor has inserted a single, contiguous span of text into an existing complete sentence (Table 1). This restriction allows us to make several assumptions which we believe make the data an especially powerful source of signal. Namely, we can assume that 1) some information was not communicated by the original sentence, 2) that information *should* have been communicated (according to a human editor), and 3) that information *is* communicated by the inserted phrase. Thus, we believe that a large data set of such edits is inherently valuable for researchers modeling inference and discourse and that the data can yield insights about representation at both the phrase and the sentence level.

Adding new relevant information
She died there **in 1949** after a long illness.
Refining claim/Resolving ambiguity
Finlay announced he'd be on the 1000th episode of "WWE Monday Night Raw"**, but he wasn't.**
Improving Discourse/Fluency
It is **also** being evaluated as a potential biological control for the invasive plant . . .

Table 1: Example atomic insertions (in bold) from the corpus and the types of semantic and discourse phenomena that such edits capture.

We mine Wikipedia edit history to create a corpus of 43 million atomic insertion and deletion edits covering 8 languages. We argue that the corpus contains distinct semantic signals not present in raw text. We thus focus our experiments on answering the following questions:

1. How is language that is inserted during editing different from general Wikipedia text?
2. What can we learn about language by observing the editing process that we cannot readily learn by observing only the final edited text?

Specifically, the contributions of this paper are:

- A new corpus (WikiAtomicEdits) of 26M atomic insertions and 17M atomic deletions covering 8 languages (§3 and §4): `http://goo.gl/language/wiki-atomic-edits`.
- Linguistic analysis showing that inserted language differs measurably from the language observed in general Wikipedia text (§5).

---

*Both authors contributed equally.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 305–315
Brussels, Belgium, October 31 - November 4, 2018.

- Language modeling experiments showing that models trained on WikiAtomicEdits encode different aspects of semantics and discourse than models trained on raw, unstructured text (§6).

2 Theoretical Motivation

We borrow the idea of an *atomic edit* from prior work in natural language inference, specifically natural logic (Lakoff, 1970; Van Benthem, 1986). MacCartney (2009) defines an atomic edit e applied to a natural language expression s as the insertion, deletion, or substitution of a sub-expression p such that both the original expression s and the resulting expression $e(s)$ are well-formed semantic constituents. E.g. s = *"She died from an illness"*, p = *"in 1949"*, and $e(s)$ = *"She died in 1949 from an illness"*. This formulation is desirable because it exposes a relationship between the surface form and the semantics of natural language while remaining agnostic about the underlying semantic representation. That is, the difference in "meaning" between s and $e(s)$ is exactly the "meaning" of p (in context), regardless of how that meaning is represented.

We adopt this philosophy in creating our corpus. We focus our analysis specifically on atomic insertion edits. We make the assumption that editors on Wikipedia are attempting to communicate true information[1] and to do so effectively. Insertion edits are thus particularly interesting because the underlying generation process admits the following assumptions:

1. The original sentence s does not effectively communicate some piece of information.
2. A reasonable reader of s would like/expect this information to be communicated.
3. This information is communicated by the inserted phrase p (in the context of $e(s)$).

We therefore believe that the supervision provided by insertion edits can improve our understanding of semantics, discourse, and composition, and that the data released will be valuable for research in these areas. The goal of our experiments is to establish that the signal provided in these edits is distinct from what one could easily obtain given currently available text corpora.

Language	Ins	Del	Total
German	3.3	1.9	5.2
English	13.7	9.3	23.0
Spanish	1.4	0.9	2.3
French	2.0	2.0	4.0
Italian	1.0	0.6	1.6
Japanese	2.2	1.3	3.5
Russian	1.4	0.9	2.3
Chinese	0.7	0.4	1.1
Total	25.7	17.2	42.9

Table 2: The number of instances (in millions) of atomic insertions/deletions for each language.

3 WikiAtomicEdits: Corpus Creation

3.1 Extracting Edits

Wikipedia edits can be accessed through Wikipedia dumps. The edits are stored as diffs on the entire Wikipedia page, meaning some processing is required to reconstruct the changes that were made at the sentence level. We use historical snapshots of each Wikipedia document and compare against subsequent snapshots to extract sentence-level edits. We strip the HTML tags and Wikipedia markup of the page and then run a sentence splitter (Gillick, 2009) to obtain a list of sentences for each snapshot. Rather than run a full, quadratic-time (Myers, 1986) sequence alignment to compare the two lists of sentences, which is infeasible for long articles, we propose an efficient precision-oriented approximation.

Given n sentences in one snapshot ("base") and m sentences in a subsequent one ("edited"), we assume that most edits are local and restrict our attention to a fixed-size window. For each sentence s_i in the base snapshot, we compute pairwise BLEU scores (Papineni et al., 2002) between s_i and the sentences $\{t_j\}_{j=i-k}^{i+k}$ ($k = 5$) in the edited snapshot. We consider the sentence with the highest BLEU score in this window as a candidate. If the sentences are not identical and the difference consists of an insertion or deletion of a single contiguous phrase[2], we add this example to the corpus. For each article, we run this algorithm over the most recent 100,000 snapshots as of February 2018. We extract edits for 8 languages. Statistics are shown in Table 2.

---

[1] This is true for the majority of edits, although about 13% of edits are "spam" (§4.3).

[2] We use the Python 2.7 `difflib` library to compute a minimal diff at the byte level.

3.2 Insertions vs. Deletions

We use the algorithm described above to extract both atomic insertions and atomic deletions. However, we chose to omit the deletions from our linguistic (§5) and language modeling (§6) analyses for two reasons. First, our intuition is that spans which are deleted by an editor are more likely to be "bad" phrases (e.g. spam, false information, or grammatical errors introduced by a previous editor). To confirm this, we manually inspected 100 of each type of edit. We found that indeed deletions contained a higher proportion of spam text and malformed English (16/100) than did insertions (7/100). Second, while insertions permit a clean set of assumptions about the relationship between the original sentence and the edited sentence (§2), it is more difficult to make generalizations about deletions. Specifically, it is difficult to say whether the original sentence *should not* communicate the information in the deleted phrase (i.e. the phrase contains false, irrelevant, or otherwise erroneous information) or rather the original sentence/surrounding context *already communicates* the information in the deleted phrase (i.e. the deleted phrase is redundant). As such, deletions are a noisier target for analysis. Nonetheless, we recognize that the deletions provide a related and likely useful signal. We thus include deletions in our corpus but leave their deeper linguistic analysis for future work.

4 Corpus Quality & Reproducibility

4.1 Annotation

Given the data collected as above, we now investigate whether the extracted edits are sufficiently clean to be useful for computational language analysis and modeling. To do this, we focus our attention specifically on the English, Spanish, and German subcorpora, as these are languages for which we could find a sufficient number of native speakers to perform the necessary annotation for our analysis. Thus, the discussion and results in this section may not be representative of the other languages in the corpus.

We are interested specifically in two questions. First, we want to measure the overall corpus quality: how many of the inserted phrases represent meaningful edits and how many are simply noise (e.g. from editor or preprocessing error)? Second, we want to understand, at least in part, the reproducibility of the corpus: could we expect a different group of human editors to produce the same edits as those observed?

	en	es	de
No Error	78%	55%	85%
Possible Error	13%	30%	9%
Clear Error	9%	15%	6%

Table 3: Corpus quality for three languages for which we were able to collect annotations. "No Error"/"Clear Error" means annotators agreed unanimously that the edit was/was not an error; "Possible Error" means annotations were mixed.

To address these questions, we collect annotations in a semi-generative manner. Each annotator is shown a sentence s and a phrase p to be inserted, and is asked to insert p into s in order to form a new sentence $e(s)$. If s is not a complete and well-formed sentence, or if there is no location at which p can be inserted such that $e(s)$ would be a complete and well-formed sentence, annotators are instructed to mark the edit as an error. We use the "error" labels in order to study corpus quality (§4.2) and use the annotators' insertion location to estimate reproducibility (§4.3).

We collect labels for 5,000 English edits, and 1,000 each for Spanish and German edits using a crowd-sourcing platform. We collect 5-way annotations for English and 3-way annotations for Spanish and German. Our choices of languages and the differing levels of redundancy were due to availability of annotators. We will release these 7,000 edits and their annotations with the corpus.

4.2 Corpus Quality

To measure corpus quality, we compute the proportion of edits marked as errors by our annotators. Table 3 shows our results. For English, in 78% of cases our annotators agreed unanimously that p could be inserted meaningfully into s (55% for Spanish; 85% for German). These numbers reassure us that, while there is some noise, the majority of the corpus represents legitimate edits with meaningful signal. For more discussion of the errors refer to Supplementary Material.

4.3 Agreement and Ambiguity

We next explore the extent to which the edits in the corpus are reproducible. In an ideal world, we would like to answer the question: given the same original sentences, would a different group of human editors produce the same edits? Answering this directly would require annotators with domain expertise and is infeasible in practice. However, we can use our crowdsourced annotation to answer a restricted variant of this question: given a sentence s and an insertable phrase p, do humans agree on where p belongs in s? We can measure agreement in this setting straightforwardly using exact match, and can interpret human performance as that of a "perfect" language model. I.e. we can interpret disagreement as evidence that reproducing the particular edit is dependent on exogenous information not available in the language of s alone (e.g. knowledge of the underlying facts being discussed, or of the author's individual style).

Based on our annotation experiment, we find that individual annotators agree with the original editor 66% of the time for English, 72% for Spanish, and 85% for German.[3] More interesting than how often humans disagree on this task, however, is *why* they disagree. To better understand this, we take a sample of 100 English sentences in which at least one human annotator disagreed with the original editor and no annotator marked the edit as an error. We then manually inspect the sample and record whether or not the annotators' choices of different insertion points give rise to sentences with different semantic meaning or simply to sentences with different discourse structure.

In particular, we consider three categories for the observed disagreements: 1) the sentences are **meaning equivalent** from a truth-conditional perspective, 2) the sentences contain **significant differences in meaning** from a truth-conditional perspective, or 3) the sentences contain **minor differences or ambiguities** in meaning (but would likely be considered equivalent from the point of view of most readers). We also include an error category, for when the disagreement stems from a single annotator making an erroneous choice. Examples of each category are given in Table 4. Note that the assessment of the truth conditions of the sentence and their equivalence is based on our judgment, and many of these judgments are subjective. We will release our annotations for this analysis with the corpus, to enable reproducibility and refinement in future research.

Table 5 shows our results. We found 49% to be meaning equivalent (i.e. the edit's location effected discourse structure only), and 22% to have significant differences in meaning (i.e. the edit's location fundamentally changed the meaning of the sentence). An additional 13% exhibited minor differences or ambiguities in meaning, and in the remaining 16% of cases, the disagreement appeared to be due to annotator error.

5 Corpus Linguistic Analysis

We now turn our attention to exploring the language in the corpus itself. In this section and in §6, our focus is on the questions put forth in the introduction: 1) how does the language that is inserted during editing differ from language that is observed in general? and 2) what can we learn about language by observing the editing process that we cannot readily learn by observing only raw text? Here, we explore these questions from a corpus linguistics perspective. The analysis in this section is based predominantly on the 14M insertion edits from the English subcorpus (Table 2).

5.1 Manual Categorization of Insertions

We first characterize the types of insertions in terms of the function they serve. Manually inspecting the edits, we identify four high-level categories. Note that we do not intend these categories to be formal or exhaustive, but rather to be illustrative of the types of semantic and discourse phenomena in the corpus: i.e. to give sense of the balance between semantic, pragmatic, and grammatical edits in the corpus. The categories we identify are as follows:

1. **Extension**: the explicit addition of new information that the author of the original sentence did *not* intend to communicate.[4]
2. **Refinement**: the addition of information that the author of the original sentence either intended to communicate or assumed the reader would already know. This category includes hedges, non-restrictive modifiers, and other clarifications or scoping-down of claims.
3. **Fluency / Discourse**: grammatical fixes, as well as the insertion of discourse connectives (*"thus"*), presuppositions (*"also"*), and editorializations (*"very"*).

---

[3] We consider cases which the annotator marks as "error" to be a disagreement with the original editor.

[4] Whether or not the author "intended" to communicate something is based on our judgment. Since this annotation is intended to be exploratory, we allow a degree of informality.

Meaning Equivalent
Paul Wheelahan**, the son of a mounted policeman,** was born in Bombala, South Wales...
Paul Wheelahan was born in Bombala, South Wales**, the son of a mounted policeman,**...
Minor Difference / Ambiguity
She moved to Australia **in 1964** and attended the University of New South Wales...
She moved to Australia and attended the University of New South Wales **in 1964**...
Significant Difference in Meaning
...he and Bart have to share a raft with Ned Flanders and **his youngest son,** Todd Flanders.
...he and **his youngest son,** Bart have to share a raft with Ned Flanders and Todd Flanders.

Table 4: Examples of sentences falling into three disagreement categories, defined in terms of the truth conditions of the edited sentence. See text for a more detailed explanation.

Meaning Equivalent	49
Significant Differences in Meaning	22
Minor Differences/Ambiguities	13
Annotator Error	16

Table 5: Analysis of 100 sentences for which at least one annotator disagreed with the gold label and no annotator marked as an error.

4. **Referring Expressions** (RE): changes in the name of an entity that do not change the underlying referent, such as adding a first name (*"Andrew"*) or a title (*"Dr."*). RE edits could fall within our definition of "refinement", but because they are especially prevalent we annotate them as a separate category.

We also define an **Error** category for spam, vandalism, and other "mistake" edits.

We manually categorize 100 randomly-sampled edits. The breakdown is shown in Table 6. In our sample, the majority (43%) were extensions, and the second most frequent where refinements (24%). No single category dominates and all are well-represented, suggesting that a variety of phenomena can be studied using this corpus.

5.2 Comparing Insertions to Raw Text

Understanding the high-level functions of edits, as above, provides some insight into the type of linguistic signals contained in the data. However, we are particularly interested in whether the language used for these functions is noticeably different from general Wikipedia text. That is: it is not obvious that the language humans use to e.g. extend or refine an existing claim should necessarily be different, in aggregate, from the language used to formulate these claims in general. We thus explore whether this is the case.

We first compare the distribution of parts of speech observed for the inserted phrases to the distribution of parts of speech that we observe in Wikipedia overall–i.e. in the sentences appearing in the final, published version of Wikipedia, not only the edit history. In order to compare the relative frequencies in a straightforward way, we look only at edits in which a single word was inserted.[5] Figure 1 shows our results for English, Spanish, and German. We see, for example, that in English, adjectives and adverbs combined make up nearly 30% of all inserted words, three and a half times higher than the frequency of adjectives/adverbs observed in the general Wikipedia corpus, and that proper nouns are inserted at a higher rate than would be suggested given their base frequency.

Looking more carefully, we see that the nature of the edits for each part of speech are qualitatively different as well. To explore this further, we look at which words appear at substantially higher rate as insertions than they do in the general Wikipedia corpus. We compute this as follows: for a word w with part of speech pos, we compute the number of times w occurs as an insertion per thousand insertions of any word of type pos, and compare this to the rate of occurrence of w per thousand occurrences of any word of type pos within the general Wikipedia corpus. Table 7 shows our results for English (Spanish and German are given in the Supplementary Material). In particular, we see that many words which are inserted at a significantly higher-than-baseline rate reflect "refinement"-type edits. Many of these are words which the original author may have commu-

---

[5] In our corpus 30% of inserted phrases are a single word, and 70% are less than five words. We compared frequencies for longer POS sequences as well, but it did not yield particular insight over looking at single POS tags.

Category	Freq.	Example
Extend	43%	The population was 39,000 in 2004**, measured at 29,413 at the 2011 Census**.
Refine	24%	. . . began an investigation into Savile 's **apparent** history of abuse. . .
RE	11%	**Andrew** Sugerman has been involved in the production of motion pictures. . .
Fluency	9%	Philippine coconut jam**, meanwhile,** is made from coconut cream. . .
Error	13%	The team **are well - known as a loser team in the past 5 years.The team** is. . .

Table 6: High-level categories into which we manually characterize edits, to understand the variety of phenomena captured by the corpus. Frequencies are based on our annotation of a sample of 100 edits.

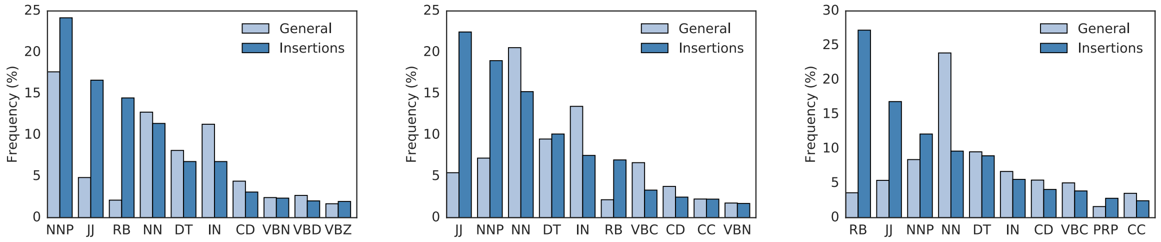

(a) English (b) Spanish (c) German

Figure 1: Most frequent POS tags for English, Spanish, and German single-word insertions. Dark blue bars show the relative frequency among inserted phrases and light blue bars show the relative frequency among phrases observed in Wikipedia in general.

nicated implicitly but the editor chose to state explicitly, such as whether or not a person is a *"current"*/*"former"* public figure[6] or is *"famous"*. On the other hand, words which are inserted at a significantly lower-than-baseline rate are those which would be unlikely to be omitted by the original author. For example, if an event is famously the *"first"* or the *"only"* one of its kind, it is highly unlikely for the original author describing that event not to use these words explicitly.

6 Language Modeling Analysis

We next explore the corpus from a language modeling perspective. Again, we are interested in understanding how the signal captured by the editing process is distinct from that captured by the final edited text alone, and in characterizing the types of signals we can learn from modeling the insertions directly. We investigate this through two tasks: first, given a sentence s and insertable phrase p, predict the index i at which p should appear in s (§6.1), and second, given a sentence s and an index i, generate candidate phrases that would be appropriate to insert into s at i (§6.2).

	NNP	JJ	RB
Over	City 16:2 Sir 7:1 US 7:1 John 6:3 Roman 4:1	former 34:6 current 11:2 famous 9:2 professional 10:3 fictional 5:1	also 187:91 currently 40:7 very 24:11 then 45:33 allegedly 10:1
Under	New 1:5 United 2:5 I 2:4 9 de 2:4 School 1:3	first 9:29 only 2:20 other 12:26 total 2:13 such 3:13	not 35:96 first 9:68 all 1:35 only 22:47 about 4:29

Table 7: Words that appear as insertions at significantly higher rates (top row) and significantly lower rates (bottom row) than their rate of occurance in Wikipedia in general. We compute "rate" as simply the observed occurrence of the given word per thousand occurrences of any word with the given POS. Table shows each word followed by (rate as insertion):(rate in general)

6.1 Predicting Insertion Locations

Task. This task–given a phrase p and a sentence s, choose the best index i in s at which to insert p–is identical to the task we asked humans to perform in §4. We consider two simple models for performing this task: a basic language model and a discriminative model trained on the insertion data. We report performance as overall accuracy. We analyze whether a model which is trained to

---

[6] We note that the addition of *"former"* is likely tied to changes in the real world (Wijaya et al., 2015).

model insertions directly captures something different than a general language model in terms of the types of errors each model makes.

Models. We evaluate two models. First, we evaluate a standard language modeling baseline (**General LM**), in which we simply insert the phrase p at every possible point in s and chose the index which yields the lowest perplexity. We use the LSTM language model from Jozefowicz et al. (2016), which obtained SOTA results on language modeling on the one billion words benchmark for English (Chelba et al., 2013). We train this language model for each language on an average of $\sim$ 500 million tokens from Wikipedia. Second, we evaluate a discriminative model specifically trained on the insertion data (**Discriminative Model**). This model represents the base sentence using a sentence encoder that produces a context-dependent representation of every word index in the sentence, and then at test time, compares the learned representation of each index with the representation of the phrase p to be inserted. We use a 256-dimensional 2-layer biLSTM encoder, initialized with FastText 300-dimensional word vectors (Mikolov et al., 2018; Grave et al., 2018).[7] We hold out 50K and 10K insertion edits for each language as development and test sets, and use the remaining edits (insertions and deletions) as training data. This provides us with at least 1 million examples for training in each language (cf. Table 2). See Supplementary Material for additional details.

Results. Table 8 shows the accuracy of each model for each language. We see that the discriminitve model trained on insertions directly performs better than the general LM by at least 1% absolute accuracy on every language, and by 3.8% absolute on average. It is worth emphasizing that this performance improvement is despite the fact that the general LM was trained with, on average, four times the number of tokens[8] and is a much larger model–the general LM has $\sim$ 2 billion parameters (Jozefowicz et al., 2016) compared to $\sim$ 1 million for the discriminative model.

	General LM	Discr. Model
German	68.1	72.9
English	58.7	68.4
Spanish	67.0	70.1
French	69.9	73.4
Italian	69.0	72.9
Japanese	73.0	74.2
Russian	72.9	74.3
Chinese	65.5	68.9
Average	68.0	71.8

Table 8: Insertion accuracy on the test set.

More interesting than raw performance is the difference in the types of errors that the models make. For each model, we take a random sample of 50 examples on which the model made a correct prediction and 50 examples on which the model made an incorrect prediction. We annotate these 200 examples[9] according to the edit type classification discussed in §5.1. Table 9 shows the results. We find a significant difference[10] ($p < 0.01$) between the types of edits on which the General LM makes correct predictions and the types on which it makes incorrect predictions. Specifically, the General LM appears to be especially good at predicting location for fluency/discourse edits, and especially poor at predicting the location of refinement edits. In contrast, we do not see any significant bias in the errors made by the discriminative model compared to its correct predictions ($p = 0.23$). We interpret this as evidence that the insertion data captures some semantic signal that is not readily gleaned from raw text corpora.

6.2 Predicting Insertion Phrases

Task. In a final set of experiments, we explore a generative version of the language modeling task: given a sentence s and an specified index i, generate a phrase p which would be appropriate to insert into s at i. We are interested in what such a model can learn about the nature of how sentences are extended: what type of information would be relevant from a semantic perspective, and natural from a discourse perspective to insert at a given point? We train two models for this task, one trained on the training split of the WikiAtomicEdits corpus, and one baseline trained on a comparable set of phrasal insertions not derived from human edits. We evaluate on the same 10K held-out insertion

---

[7] `https://fasttext.cc/docs/en/crawl-vectors.html`

[8] The number of tokens in the WikiAtomicEdits is computed as the the total number of words in the edited sentence $e(s)$ after the insertion. Refer to Supplementary Material for more detailed statistics on the size of the dataset.

[9] To avoid bias, the 200 examples are shuffled and the annotator does not know which group (correct/incorrect, or which model) each example belongs to.

[10] We use the chi-squared test provided by scipy.stats.

	Base Freq.	General LM		Discr. Model	
		✓	✗	✓	✗
Extend	25	21	19	25	21
Refine	14	7	18	13	14
RE	6	7	9	4	9
Fluency	5	15	4	8	6

Table 9: Relationship between model accuracy and insertion type, based on a sample of 50 correct (✓) and 50 incorrect (✗) predictions from each model. Base frequency is shown for reference and is based on our analysis from §5.1. The General LM shows a bias in accuracy by insertion type. This bias is not observed for the discriminative model.

edits as in §6.1, and measure performance using both a strict "exact match" as well as a softer similarity metric.

Model. We use an standard sequence-to-sequence model (Sutskever et al., 2014), modifying the input with a special token denoting the insertion point. For example, given the input [*" Angel " is a song recorded by <ins> pop music duo Eurythmics .*], the model would be trained to produce the target phrase [*the British*]. We use a two-layer bidirectional encoder using the same 300-dimensional FastText embeddings as in §6.1, and a sequence decoder with attention (Bahdanau et al., 2015) using a learned wordpiece model (Schuster and Nakajima, 2012) with a vocabulary of 16,000.

Experimental Design. We train one version of this model on the same set of 23M English examples as the discriminative insertion model from §6.1; we refer to the model trained on this data as **Edits**. For comparison, we train an identical model on a set of simulated insertions which we create by sampling sentences from Wikipedia and removing contiguous spans of tokens, which we then treat as the insertion phrases. To ensure that this data is reasonably comparable to the WikiAtomicEdits data, we parse the sampled sentences (Andor et al., 2016) and only remove a span if it represents a full subtree of the dependency parse and is not the subject of the sentence.[11] We generate 23M such "psuedo-edits" for training, the same size as the WikiAtomicEdits training set. We refer to the model trained on this data as **General**.

Results. We look at the top 10 phrases proposed by each model, as decoded by beam search. In addition to reporting standard LM perplexity, we compute two measures of performance, which are intended to provide an intuitive picture of how well each model captures the nature of the information that is introduced by the human editors. Specifically, we compute **Exact Match** as the proportion of sentences for which the model produced the gold phrase (i.e. the phrase inserted by the human editor) somewhere among the top 10 phrases. We also compute **Similarity@1** as the mean cosine similarity of each top-ranked phrase and respective gold phrase over the test set. We use the sum of the Glove embeddings (Pennington et al., 2014) of each word in the phrase as a simple approximation of the phrase vector.

Table 11 shows the results. We see that, compared to the model trained on General Wikipedia, the model trained on WikiAtomicEdits generates edits which are more similar to the human insertions, according to all of our metrics. Table 10 provides a few qualitative examples of how the phrases generated by the Edits model differ from those generated by the General model. Specifically, we see that the Edits model proposes phrases which better capture the discourse function of the human edit: e.g. providing context for/elaboration on a previously-stated fact. We note that this does not mean that training on Edits is inherently "better" than on General text, but rather that the supervision encoded by the WikiAtomicEdits corpus encodes aspects of language that are distinct from those easily learned from existing resources.

7 Related Work

Wikipedia Edits. Wikipedia edit history has been used as a source of supervision for a variety of NLP tasks, including sentence compression and simplification (Yamangil and Nelken, 2008; Yatskar et al., 2010), paraphrasing (Max and Wisniewski, 2010), entailment (Zanzotto and Pennacchiotti, 2010; Cabrio et al., 2012), and writing assistance (Zesch, 2012; Cahill et al., 2013; Grundkiewicz and Junczys-Dowmunt, 2014). User edits from Wikipedia and elsewhere have also been analyzed extensively for insight into the editing process and the types of edits made (Daxenberger and Gurevych, 2012, 2013; Yang et al., 2017)

---

[11] Not all of the inserted phrases in WikiAtomicEdits are well-formed constituents. However, generating psuedo-edits using this heuristic provided a cleaner, more realistic comparison than using fully-random spans.

She is cited as the first female superstar of Hindi Cinema **and India ’s Meryl Streep**	
Edits	General
and is the best actress of the film	in japan
and is the best actress of the Indian cinema	in june
and is the best actress of the film industry	in 2011

He is married to Aida Leanca **and has two children**	
Edits	General
and has a daughter	in january
, and has a daughter	in june
, and has a daughter and a daughter	in january 2012

Table 10: Predicted phrase insertions from model trained on Edits vs. General corpus. The Edits model better captures the discourse function of the human edit, e.g. elaborating on the previously-stated fact, while the General model gives syntactically-appropriate but generic insertions.

	Edits	General
Log Perplexity	8.32	9.23
Exact Match	13.1%	8.0%
Similarity@1	0.54	0.48

Table 11: Comparison of how closely each model’s generated phrases match the phrase inserted by the human editor. “Edits” was trained on WikiAtomicEdits and “General” was trained on comparable data not derived from human edits. We consider the top 10 phrases generated by each model.

and to better understand argumentation (Tan and Lee, 2014). Particular attention has been given to spam edits (Adler et al., 2011) and editor quality (Leskovec et al., 2010). Our work differs in that WikiAtomicEdits is much larger than currently available corpora, both by number of languages and by size of individual languages. In addition, our focus on atomic edits should facilitate more controlled studies of semantics and discourse.

Sentence Representation and Generation. We view the WikiAtomicEdits corpus as being especially valuable for ongoing work in sentence representation and generation, which requires models of what “good” sentences look like and how they are constructed. Recent work has attempted to model sentence generation by re-writing existing sentences, either using crowdsourced edit examples (Narayan et al., 2017) or unsupervised heuristics (Guu et al., 2018); in contrast, we provide a large corpus of natural, human-produced edits.

Also related is recent work in sentence representation learning from raw text (Kiros et al., 2015; Peters et al., 2018), bitext (McCann et al., 2017), and other supervised tasks including NLI (Conneau et al., 2017). Especially related is work on learning representations from weakly-labelled discourse relations (Nie et al., 2017; Jernite et al., 2017), as the WikiAtomicEdits corpus captures similar types of discourse signal.

Description of Data Release

Our full corpus is available for download at `http://goo.gl/language/wiki-atomic-edits`. The data contains 26M atomic insertions and 17M atomic deletions covering 8 languages. All sentences (both the original sentence s, and the edited sentence $e(s)$) have been POS-tagged and dependency parsed (Andor et al., 2016) as well as scored using a SOTA LM (Jozefowicz et al., 2016). We also release the 5K 5-way human insertion annotations for English, and 1K 3-way annotations each for Spanish and German, as described in §4.

8 Conclusion

We have introduced the WikiAtomicEdits corpus, derived from Wikipedia’s edit history, which contains 43M examples of atomic insertions and deletions in 8 languages. We have shown that the language in this corpus is meaningfully different from the language we observe in general, and that models trained on this corpus encode different aspects of semantics and discourse than models trained on raw text. These results suggest that the corpus will be valuable to ongoing research in semantics, discourse, and representation learning.

References

B. Thomas Adler, Luca De Alfaro, Santiago M. Mola-Velasco, Paolo Rosso, and Andrew G. West. 2011. Wikipedia vandalism detection: Combining natural language, metadata, and reputation features. In *Proc. of CICLing*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Elena Cabrio, Bernardo Magnini, and Angelina Ivanova. 2012. Extracting context-rich entailment rules from wikipedia revision history. In *Proc. of the 3rd Workshop on the People's Web Meets NLP*.

Aoife Cahill, Nitin Madnani, Joel Tetreault, and Diane Napolitano. 2013. Robust systems for preposition error correction using wikipedia revisions. In *Proc. of NAACL*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Johannes Daxenberger and Iryna Gurevych. 2012. A corpus-based study of edit categories in featured and non-featured Wikipedia articles. In *Proce. of COLING*.

Johannes Daxenberger and Iryna Gurevych. 2013. Automatically classifying edit categories in wikipedia revisions. In *Proc. of EMNLP*.

Dan Gillick. 2009. Sentence boundary detection and the problem with the U.S. In *Proc. of NAACL*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proc. of LREC*.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2014. The wiked error corpus: A corpus of corrective wikipedia edits and its application to grammatical error correction. In *Advances in Natural Language Processing – Lecture Notes in Computer Science*, volume 8686, pages 478–490. Springer.

K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics (TACL)*.

Yacine Jernite, Samuel R Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proc. of NIPS*.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*, 22(1/2):151–271.

Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Governance in social media: A case study of the Wikipedia promotion process. In *Proc. of ICWSM*.

Bill MacCartney. 2009. *Natural language inference*. Stanford University.

Aurélien Max and Guillaume Wisniewski. 2010. Mining naturally-occurring corrections and paraphrases from wikipedia's revision history. In *Proc. of LREC*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proc. of NIPS*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proc. of LREC*.

Eugene W Myers. 1986. An O(ND) difference algorithm and its variations. *Algorithmica*, 1(1-4):251–266.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *Proc. of EMNLP*.

Allen Nie, Erin D Bennett, and Noah D Goodman. 2017. Dissent: Sentence representation learning from explicit discourse relations. *arXiv preprint arXiv:1710.04334*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *Proc. of ICASSP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

Chenhao Tan and Lillian Lee. 2014. A corpus of sentence-level revisions in academic writing: A step towards understanding statement strength in communication. In *Proc. of ACL*.

Johan Van Benthem. 1986. *Natural Logic*. Springer Netherlands.

Derry Tanti Wijaya, Ndapandula Nakashole, and Tom Mitchell. 2015. "A spousal relation begins with a deletion of engage and ends with an addition of divorce": Learning state changing verbs from wikipedia revision history. In *Proc. of EMNLP*.

Elif Yamangil and Rani Nelken. 2008. Mining wikipedia revision histories for improving sentence compression. In *Proc. of ACL*.

Diyi Yang, Aaron Halfaker, Robert Kraut, and Eduard Hovy. 2017. Identifying semantic edit intentions from revisions in wikipedia. In *Proc. of EMNLP*.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *Proc. of NAACL*.

Fabio Massimo Zanzotto and Marco Pennacchiotti. 2010. Expanding textual entailment corpora from wikipedia using co-training. In *Proc. of the 2nd Workshop on The People's Web Meets NLP*.

Torsten Zesch. 2012. Measuring contextual fitness using error contexts extracted from the wikipedia revision history. In *Proc. of EACL*.

On the Relation between Linguistic Typology and (Limitations of) Multilingual Language Modeling

Daniela Gerz[1]*, **Ivan Vulić**[1]*, **Edoardo Maria Ponti**[1], **Roi Reichart**[2], **Anna Korhonen**[1]

[1]Language Technology Lab, DTAL, University of Cambridge
[2]Faculty of Industrial Engineering and Management, Technion, IIT
[1]`{dsg40, iv250, ep490, alk23}@cam.ac.uk`
[2]`roiri@ie.technion.ac.il`

Abstract

A key challenge in cross-lingual NLP is developing general language-independent architectures that are equally applicable to any language. However, this ambition is largely hampered by the variation in structural and semantic properties, i.e. the typological profiles of the world's languages. In this work, we analyse the implications of this variation on the language modeling (LM) task. We present a large-scale study of state-of-the art *n-gram based* and *neural* language models on 50 typologically diverse languages covering a wide variety of morphological systems. Operating in the *full vocabulary LM setup* focused on *word-level prediction*, we demonstrate that a coarse typology of morphological systems is predictive of absolute LM performance. Moreover, *fine-grained typological features* such as exponence, flexivity, fusion, and inflectional synthesis are borne out to be responsible for the proliferation of low-frequency phenomena which are organically difficult to model by statistical architectures, or for the meaning ambiguity of character n-grams. Our study strongly suggests that these features have to be taken into consideration during the construction of next-level language-agnostic LM architectures, capable of handling morphologically complex languages such as Tamil or Korean.

1 Introduction

Deep learning has allowed NLP algorithms to dispose of manually-crafted features, and to virtually achieve language independence. However, their performance still varies noticeably across languages due to different underlying data distributions (Bender, 2013; O'Horan et al., 2016). Linguistic typology, the systematic comparison of the world's languages, holds promise to explain these idiosyncrasies and interpret statistical models in terms of variation in language structures (Ponti et al., 2017).

In order to evaluate how cross-lingual structural variation hinders the design of effective general-purpose algorithms, we propose the task of language modeling (LM) as a testbed. In particular, we opt for a full-vocabulary setup where no word encountered at training time is treated as an unknown symbol, in order to **a)** ensure a fair comparison across languages with different word frequency rates and **b)** avoid setting an arbitrary threshold on vocabulary size (Cotterell et al., 2018).

Although there has recently been a tendency towards expanding test language samples, the datasets considered in previous works (Botha and Blunsom, 2014; Vania and Lopez, 2017; Kawakami et al., 2017; Cotterell et al., 2018) are not entirely adequate yet to represent the typological variation and to ground cross-lingual generalisations empirically. Hence, we test several LM architectures (including n-gram, neural, and character-aware models) on a novel and wider set of 50 languages sampled according to stratification principles.

Through this large-scale multilingual analysis, we shed new light on the current limitations of standard LM models and offer support to further developments in multilingual NLP. In particular, we demonstrate that the previous fixed-vocabulary assumption in fact ignores the limitations of language modeling for morphologically rich languages. Moreover, we find a strong correlation across the board between LM model performances and the type of morphological system adopted in each language.

To motivate this correlation we show how fine-grained typological properties interact with the frequency distribution (Zipf, 1949) by regulating word boundaries and the proliferation of word forms; and 2) with the mapping between morphemes (here intended as character n-grams) and meaning, by possibly blurring it.

The paper is organised as follows. After provid-

---

[*] Both authors equally contributed to this work.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 316–327
Brussels, Belgium, October 31 - November 4, 2018.

ing a short overview of multilingual LM and its possible setups (§2), we describe the cross-lingual variation in morphological systems and propose a novel typologically diverse dataset for LM in §3. We outline the data in §4 and benchmarked language models in §5. Finally, we discuss the results in light of linguistic typology in §6.

2 Multilingual Language Modeling

A language model computes a probability distribution over sequences of word tokens, and is typically trained to maximise the likelihood of word input sequences. The LM objective is expressed as:

$$P(w_1, ...w_n) = \prod_i P(w_i|w_1, ...w_{i-1}) \qquad (1)$$

w_i is a word token with the index i in the sequence. LM is considered a central task in NLP and language understanding, with applications in speech recognition (Mikolov et al., 2010), text summarisation (Filippova et al., 2015; Rush et al., 2015), and information retrieval (Ponte and Croft, 1998; Zamani and Croft, 2016). The importance of language modeling has been accentuated even more in representation learning recently, where it is used as a novel form of unsupervised pre-training (and an alternative to static word embeddings) for the benefit of a variety of NLP applications (Peters et al., 2018; Howard and Ruder, 2018).

Related Work: Datasets and Evaluation. Language modeling is predominantly tested on English and other Western European languages. Standard English LM benchmarks are the Penn Treebank (PTB) (Marcus et al., 1993) and the 1 Billion Word Benchmark (BWB) (Chelba et al., 2013). Datasets extracted from BBC News (Greene and Cunningham, 2006) and IMDB Movie Reviews (Maas et al., 2011) are also used for LM evaluation in English (Wang and Cho, 2016; Miyamoto and Cho, 2016; Press and Wolf, 2017).

For multilingual LM evaluation, Botha and Blunsom (2014) extract datasets for Czech, French, Spanish, German, and Russian from the 2013 Workshop on Statistical Machine Translation (WMT) data (Bojar et al., 2013). Kim et al. (2016) reuse these datasets and add Arabic. Ling et al. (2015) evaluate on English, Portuguese, Catalan, German and Turkish datasets extracted from Wikipedia. Kawakami et al. (2017) evaluate on 7 European languages using Wikipedia data, including Finnish.

To the best of our knowledge, the largest datasets used in previous work are from (Müller et al., 2012; Cotterell et al., 2018) and amount to 21 languages from the Europarl data (Koehn, 2005). Despite the large coverage of languages, these sets are still restricted only to the languages of the European Union. On the other hand, the most typologically diverse dataset thus far was released by Vania and Lopez (2017). It includes 10 languages representing some morphological systems.

This short survey of related work demonstrates a clear tendency towards extending LM evaluation to other languages, abandoning English-centric assumptions, and focusing on language-agnostic LM architectures. However, a comprehensive evaluation set that systematically covers a wide and balanced spectrum of typologically diverse languages is still missing. The novel dataset we discuss in this paper aims at bridging this gap (see §4).

Fixed vs. Full Vocabulary Setup. A majority of language models rely on the fixed-vocabulary assumption: they use a special symbol <UNK> that represents all words not present in the fixed vocabulary V, which are termed out-of-vocabulary (OOV). Selecting the set V typically slips under the radar, and can be seen as "something of a black art" despite its enormous impact on final LM performance (Cotterell et al., 2018).[1] Standard LM setups either fix the vocabulary V to the top n most frequent words, typically with $n = 10,000$ or $n = 5,000$ (Mikolov et al., 2010; Ling et al., 2015; Vania and Lopez, 2017; Lee et al., 2017, *inter alia*), or include in V only words with a frequency below a certain threshold (typically 2 or 5) (Heafield et al., 2013).

The rationale behind fixing the set V is **a)** to make the language model more robust to handling OOVs and to effectively bypass the problem of unreliable word estimates for low-frequency and unseen words (by ignoring them), and **b)** to enable direct comparisons of absolute perplexity scores across different models. However, this posits a critical challenge as cross-linguistic evaluation becomes uneven. In fact, we witness a larger proportion of vocabulary words replaced by <UNK> in morphologically rich languages because of their higher OOV rates (see Table 3). What is more, while the fixed-vocabulary assumption artificially

---

[1] For instance, Vania and Lopez (2017) report perplexity scores of ≈20 for Finnish when V is fixed to the $5k$ most frequent words. The same model in the full-vocabulary setup obtains perplexity scores of ≈2,000.

FI	*Kreikkalaiset sijoittivat geometrian synnyn muinaiseen Egyptiin , jossa sitä tarvittiin maanmittaukseen .*
FI (MIN-5)	*<UNK> <UNK> <UNK> synnyn <UNK> Egyptiin , jossa sitä tarvittiin <UNK> .*
FI (10K)	*<UNK> <UNK> <UNK> <UNK> <UNK> <UNK> , jossa sitä <UNK> <UNK> .*
KO	그 뒤 한시 백일장에서 장원하여 신동으로 알려졌다. 그러나 그의 집은 지독하게 가난했다 .
KO (MIN-5)	그 뒤 <UNK> <UNK> <UNK> <UNK> 알려졌다 . 그러나 그의 집은 <UNK> <UNK> .
KO (10K)	그 뒤 <UNK> <UNK> <UNK> <UNK> 알려졌다 . 그러나 그의 <UNK> <UNK> <UNK> .

Table 1: Examples from Finnish and Korean LM datasets after applying the standard fixed-vocabulary assumption. MIN=5: only words with corpus frequency above 5 are retained in the final fixed vocabulary V; 10K: V comprises the $10k$ most frequent words.

Type	Fusion	Exponence	Flexivity	Synthesis
Isolating	low	1:1	1:1	low
Fusional	mid	many:1	1:many	mid
Introflexive	high	many:1	1:many	mid
Agglutinative	mid	1:1	1:1	high

Table 2: Traditional morphological types described in terms of selected features from WALS.

improves the perplexity measure, it actually makes the models less useful, especially in morphologically rich languages, as exemplified in Table 1.

Our goal is to get a clear picture on how different typological features and the corresponding corpus frequency distributions affect LM performance, without the influence of the unrealistic fixed-vocabulary assumption. Therefore, we work in the *full-vocabulary LM setup* (Adams et al., 2017; Grave et al., 2017). This means that we explicitly decide to retain also infrequent words in the modeled data: V contains all words occurring at least once in the training set, only unseen words from test data are treated as OOVs. We believe that this setup leads to an evaluation that pinpoints the crucial limitations of standard LM architectures.[2]

Why Not Open Vocabulary Setup? Recent neural LM architectures have also focused on handling large vocabularies and unseen words using character-aware modeling (Luong and Manning, 2016; Jozefowicz et al., 2016; Kawakami et al., 2017, *inter alia*). This setup is commonly referred to as the *open-vocabulary* setup. However, two distinct approaches with crucial modeling differences are referred to by the same term in the literature. **a)** *Word-level generation* constructs word vectors for arbitrary words from constituent subword-level components, but word-level prediction is still evaluated based on the fixed-vocabulary assumption. **b)** *Character-level generation* predicts characters instead of words.

Given that character-level prediction and word-level prediction operate on entirely different sets of symbols, their performance is hardly comparable. Still, Jozefowicz et al. (2016) report that, in a hybrid setup which evaluates character-level prediction based on word-level perplexity with the fixed-vocabulary assumption, current state-of-the-art word-level prediction models (i.e., the ones we discuss in §5) still significantly outperform such hybrid character-level prediction approaches. Therefore, we operate in the full-vocabulary setup.

3 Typology of Morphological Systems

Aiming for a comprehensive multilingual LM evaluation in this study, we survey all possible types of morphological systems (Haspelmath and Sims, 2013), which possibly lead to different performances. Traditionally, languages have been grouped into the four main categories: *isolating*, *fusional*, *introflexive* and *agglutinative*, based on their position along a spectrum measuring the preference on breaking up concepts in many words (on one extreme) or rather compose them into single words (on the other extreme).

The mono-dimensionality of this spectrum has recently been challenged as languages exhibit a multitude of morphological features that do not covary across languages (Plank, 2017; Ponti et al., 2018). The typological database WALS (Dryer and Haspelmath, 2013) documents several of them that are relevant for LM: *inflectional synthesis*, *fusion*, *exponence*, and *flexivity*. Note that the prototypes of traditional categories can be approximated in terms of these features, as shown in Table 2, although more combinations are possible.

Languages specify different subsets of grammatical categories (such as tense for verbs, or num-

---

[2] For instance, as discussed later in §3 and validated empirically in §6, the vocabularies of morphologically rich languages are inherently larger: it is simply more difficult to learn and make LM predictions in such languages.

ber for nouns), and for each category different values are available in each language: for instance, Finnish has less tense values (it lacks a future), whereas Slovene has more number values (including a dual) compared to English. The feature **inflectional synthesis** for verbs (Bickel and Nichols, 2013) measures how many categories appear on the maximally inflected verb per language. More available categories enlarge the vocabulary (and consequently the OOV rate) with forms instantiating all possible combinations of their values.

Another crucial aspect is how the available grammatical categories are expressed, which can be described by fusion, exponence, and flexivity. **Fusion** measures the degree of connectedness between a grammatical marker to another word. The marker can be (from lower to higher fusion) a separate word, a clitic, an affix, or can affect the form of the root itself (e.g. an umlaut or a tone).

Exponence measures the number of categories (e.g., tense, number) a single morpheme tends to convey. Exponence is separative if one grammatical category is conveyed by one morpheme (1:1), and cumulative if multiple categories are grouped into one morpheme (many:1).

Flexivity indicates the possibility that the value of a grammatical category be mapped into different morphological forms (1:many). In other terms, lemmas belonging to the same part-of-speech are divided into inflectional classes (such as declension classes for nouns or conjugation classes for verbs), each characterised by a different paradigm, that is, a different set of value-to-form mappings.

The three last features are illustrated by the examples Ex. (2)-Ex. (5), all uttering the sentence "*I will guard the doors and I will not open (them)*".[3]

(2) *tôi* *sẽ* *bảo* *vệ* *cửa* *và* *tôi* *sẽ*
I FUT guard door and I FUT
không *mở*
NEG open (Vietnamese)

(3) *kapı-lar-ı* *koruy-acağ-ım* *ve*
door-PL-ACC guard-FUT-1SG and
aç-may-acağ-ım
open-NEG-FUT-1SG (Turkish)

(4) *sorvegli-erò* *le* *port-e* *e* *non*
guard-FUT.1SG DEF door-PL and NEG
apr-irò
open-FUT.1SG (Italian)

(5) *'e-šmor* *'al* *ha-d'lat-ót* *v'-lo*
1SG-guard.FUT on DEF-door-PL and-NEG
'e-ftach *otán*
1SG-wait.FUT them (Hebrew)

In particular, consider how tense and person are expressed on verbs. Vietnamese in Ex. (2) puts two particles *tôi* and *sẽ* before the verb, which are distinct (separate exponence), autonomous from the root (no fusion), and fixed (absence of flexivity). Turkish in Ex. (3) attaches suffixes: *-acak-* for tense and *-ım* for person. These are distinct (separate exponence), joined to the roots (concatenative fusion), and (phonologically determined variants of) the same morpheme (1:1 flexivity). Italian in Ex. (4) uses affixes *-erò* and *-irò*: they are concatenated to the root with respect to fusion, convey both tense and person (cumulative exponence), and are dissimilar (presence of flexivity). Finally, in Ex. (5) for Hebrew the consonant pattern of the verb *š-m-r* is interdigitated by the vowel *-o-* for tense, and preceded by a prefix *'e-* for person. The first phenomenon alters the root itself (introflexive fusion), is distinct from the second (separate exponence), and changes its realisation based on the verb's lemma (presence of flexivity).

The above evidence strongly motivates us, as well as recent previous work (Vania and Lopez, 2017; Kawakami et al., 2017; Cotterell et al., 2018), to approach LM with models that are aware of the inner structure of their input words, and to benchmark these modeling choices on a typologically diverse range of languages, as shown in §4.

4 Data

Selection of Languages. Our selection of test languages is guided by the following goals: **a)** we have to ensure the coverage of typological properties from §3, and **b)** we want to analyse a large set of languages which extends and surpasses other work in the LM literature (see §2).

Since cross-lingual NLP aims at modeling *extant* languages rather than *possible* languages (including, e.g., extinct ones), creating a balanced sample is challenging. In fact, attested languages, intended as a random variable, are extremely sparse and not independent-and-identically-distributed (Cotterell and Eisner, 2017). First, available and reliable data exist only for a fraction of the world's languages. Second, these data are biased because their features may not stem from the underlying distribution, i.e., from what is naturally possible/frequent, but rather

---

[3] All morphological glosses follow the Leipzig glossing rules, listed at `https://www.eva.mpg.de/lingua/resources/glossing-rules.php`

can be inherited by genealogical relatedness or borrowed by areal proximity (Bakker, 2010). To mitigate these biases, theoretical works resorted to stratification approaches, where each subgroup of related languages is sampled independently. maximizing their diversity (Dryer, 1989, *inter alia*). We perform our selection in the same spirit.

We start from the Polyglot Wikipedia (PW) project (Al-Rfou et al., 2013) which provides cleaned and tokenised Wikipedia data in 40 languages. However, the majority of the PW languages are similar from the perspective of genealogy (26/40 are Indo-European), geography (28/40 are Western European), and typology (26/40 are fusional). Consequently, the PW set is not a representative sample of the world's languages.

To amend this limitation, we source additional languages with the data coming from the same domain, Wikipedia, considering candidates in descending order of corpus size cleaned and preprocessed by the Polyglot tokeniser (Al-Rfou et al., 2013). Since fusional languages are already represented in the PW, we add new languages from other morphological types: isolating (*Min Nan, Burmese, Khmer*), agglutinative (*Basque, Georgian, Kannada, Tamil, Mongolian, Javanese*), and introflexive languages (*Amharic*).

Partition. We construct datasets for all 50 languages by extracting the first 40K sentences for each language, and split them into train (34K), validation (3K), and test (3K). This choice has been motivated by the following observations: **a)** we require similarly-sized datasets from the same domain for all languages; **b)** the size of the datasets has to be similar to the standard English PTB dataset (Marcus et al., 1993) which has been utilised to guide LM development in English for more than 20 years. The final list of 50 languages along with their language codes (ISO 639-1), morphological type (i.e., isolating, fusional, introflexive, agglutinative), and corpus statistics is provided in Table 3.

5 Models and Experimental Setup

Benchmarked Language Models. The availability of LM evaluation sets in a large number of diverse languages, as described in §4, gives an opportunity to conduct a full-fledged multilingual analysis of representative LM architectures for word-level prediction. First, we evaluate a state-of-the-art model from the *n-gram* family of models (Goodman, 2001) from the KenLM package.[4] It is based on 5-grams with extended Kneser-Ney smoothing (Kneser and Ney, 1995; Heafield et al., 2013). We refer to this model as **KN5**.

Modern LM architectures are almost exclusively based on recurrent neural networks (RNNs), and especially on Long-Short-Term Memory networks (LSTMs). (Mikolov et al., 2010; Sundermeyer et al., 2015; Chen et al., 2016, *inter alia*). They map a sequence of input words to embedding vectors using a look-up matrix and then perform word-level prediction by passing the vectors to the **LSTM**.

Finally, we also evaluate a character-aware variant of the neural LSTM LM architecture. We use the **Char-CNN-LSTM** model (Kim et al., 2016) due to its public availability and strong performance in several languages. In this model, each character is embedded and passed through a convolutional neural network with max-over-time pooling (LeCun et al., 1989), followed by a highway network transformation (Srivastava et al., 2015) to build word representations from their constituent characters. By resorting to character-level information, the model is able to provide better parameter estimates for lower-frequency words, which is particularly important for morphologically rich languages. The CNN-based word representations are then processed in a sequence by a regular LSTM network to obtain word-level predictions.

Evaluation Setup. We report *perplexity* scores (Jurafsky and Martin, 2017, chapter 4.2.1) using the *full* vocabulary for each respective LM dataset. This means that we explicitly decide to retain also infrequent words in the data and analyse the difficulty of modeling such words in morphologically rich languages (see §2 for the discussion).

In the full-vocabulary setup, the set V comprises all words occurring at least once in the training set. Unseen test words are mapped to *one* <UNK> vector, sampled from the the space of trained word vectors relying on a normal distribution and the same fixed random seed for all models. On the other hand, KN5 by design has a slightly different way of handling unseen test words: they are regarded as outliers and assigned low-probability estimates.

Training Setup and Parameters. For LSTM and Char-CNN-LSTM language models, we reproduce the standard LM setup of Zaremba et al. (2015) and parameter choices of Kim et al. (2016).

---

[4] `https://github.com/kpu/kenlm`

Batch size is 20 and a sequence length is 35, where one step corresponds to one word token. The maximum word length is chosen dynamically based on the longest word in the corpus. The corpus is processed continuously; the RNN hidden states reset at the beginning of each epoch. Parameters are optimised with SGD, and the gradient is averaged over the batch size and sequence length. We then scale the averaged gradient by the sequence length (=35) and clip to 5.0 for more stable training. The learning rate is 1.0, decayed by 0.5 after each epoch if the validation perplexity does not improve. All models are trained for 15 epochs, which is typically sufficient for model convergence. Finally, KN5 is trained relying on the suggested parameters from the KenLM package.

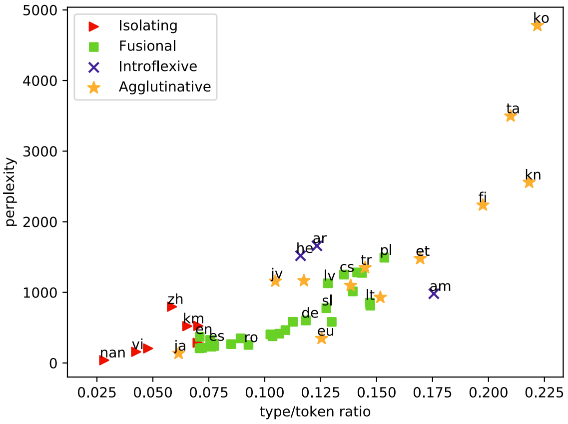

Figure 1: Perplexity scores with the Char-CNN-LSTM language model (Kim et al., 2016) on PTB-sized language modeling data in 50 languages as a function of type-to-token ratios in training data.

6 Results and Discussion

In this section, we present our main empirical findings on the connection between LM performance and corpus statistics emerging from different typological profiles (see §3). Before proceeding, we stress that the absolute perplexity scores across different languages are not directly comparable, but their values provide evidence on the difficulty and limitations of language modeling in each language, considering the fact that all language models were trained on similarly-sized datasets. The results for all three benchmarked language models on all 50 languages are summarised in Table 3.

Comparison of Language Models. A quick inspection of the results from Table 3 reveals that the Char-CNN-LSTM model is the best-performing model overall. We report the best results with that model for 48/50 languages and across all traditional morphological types. Gains over the simpler recurrent LM architecture (i.e., the LSTM model) are present for all 50/50 languages. In short, this means that character-level information on the input side of neural architectures, in addition to leading to fewer parameters, is universally beneficial for the final performance of word-level prediction, as also suggested by Kim et al. (2016) on a much smaller set of languages. By relying on character-level knowledge, Char-CNN-LSTM model provides better estimates for lower-frequency words.

Moreover, the results show that KN5 is a competitive baseline for several languages (e.g., Kannada, Thai, Amharic). This further highlights the importance of testing models on a typologically diverse set of languages: despite the clear superiority of neural LM architectures such as Char-CNN-LSTM in a large number of languages, the results and the marked outliers still suggest that there is currently no “one-size-fits-all” model.

In general, large perplexity scores for certain languages (e.g., agglutinative languages such as Finnish, Korean, Tamil, or introflexive languages), especially when compared to performance on English on a similarly-sized dataset, clearly point at the limitations of all the “language-agnostic” LM architectures. As suggested by Jozefowicz et al. (2016), LM performance in English can be boosted by simply collecting more data and working with large vocabularies (e.g., reducing the number of relevant OOVs). However, this solution is certainly not applicable to a majority of the world’s languages (Bird, 2011; Gandhe et al., 2014; Adams et al., 2017), see later in §6: *Further Discussion.*

Frequency Analysis and Traditional Morphological Types. We now analyse all languages in our collection according to word-level frequency properties also listed in Table 3 for all 50 languages. We report: 1) the vocabulary size (i.e., the total number of vocabulary words in each training dataset); 2) the total number of test words not occurring in the corresponding training data; 3) the total number of tokens in both training and test data; and finally 4) type-to-token ratios (TTR) in training data. We also plot absolute perplexity scores of Char-CNN-LSTM (Kim et al., 2016), the best-performing model overall (see §6), in relation to TTR ratios in Figure 1.

	Data Stats					Baseline Models		
Language (code)	Vocab Size (Train)	New Test Vocab	Number Tokens (Train)	Number Tokens (Test)	Type / Token (Train)	KN5	LSTM	Char-CNN-LSTM
× Amharic (am)	89749	4805	511K	39.2K	0.18	1252	1535	**981**
× Arabic (ar)	89089	5032	722K	54.7K	0.12	2156	2587	**1659**
☐ Bulgarian (bg)	71360	3896	670K	49K	0.11	610	651	**415**
☐ Catalan (ca)	61033	2562	788K	59.4K	0.08	358	318	**241**
☐ Czech (cs)	86783	4300	641K	49.6K	0.14	1658	2200	**1252**
☐ Danish (da)	72468	3618	663K	50.3K	0.11	668	710	**466**
☐ German (de)	80741	4045	682K	51.3K	0.12	930	903	**602**
☐ Greek (el)	76264	3767	744K	56.5K	0.10	607	538	**405**
☐ English (en)	55521	2480	783K	59.5K	0.07	533	494	**371**
☐ Spanish (es)	60196	2721	781K	57.2K	0.08	415	366	**275**
★ Estonian (et)	94184	3907	556K	38.6K	0.17	1609	2564	**1478**
★ Basque (eu)	81177	3365	647K	47.3K	0.13	560	533	**347**
☐ Farsi (fa)	52306	2041	738K	54.2K	0.07	355	263	**208**
★ Finnish (fi)	115579	6489	585K	44.8K	0.20	2611	4263	**2236**
☐ French (fr)	58539	2575	769K	57.1K	0.08	350	294	**231**
× Hebrew (he)	83217	3862	717K	54.6K	0.12	1797	2189	**1519**
☐ Hindi (hi)	50384	2629	666K	49.1K	0.08	473	426	**326**
☐ Croatian (hr)	86357	4371	620K	48.1K	0.14	1294	1665	**1014**
★ Hungarian (hu)	101874	5015	672K	48.7K	0.15	1151	1595	**929**
▷ Indonesian (id)	49125	2235	702K	52.2K	0.07	454	359	**286**
☐ Italian (it)	70194	2923	787K	59.3K	0.09	567	493	**349**
★ Japanese (ja)	44863	1768	729K	54.6K	0.06	169	156	**136**
★ Javanese (jv)	65141	4292	622K	52K	0.10	1387	1443	**1158**
★ Georgian (ka)	80211	3738	580K	41.1K	0.14	1370	1827	**1097**
▷ Khmer (km)	37851	1303	579K	37.4K	0.07	586	637	**522**
★ Kannada (kn)	94660	4604	434K	29.4K	0.22	**2315**	5310	2558
★ Korean (ko)	143794	8275	648K	50.6K	0.22	5146	10063	**4778**
☐ Lithuanian (lt)	81501	3791	554K	41.7K	0.15	1155	1415	**854**
☐ Latvian (lv)	75294	4564	587K	45K	0.13	1452	1967	**1129**
▷ Malay (ms)	49385	2824	702K	54.1K	0.07	776	725	**525**
★ Mongolian (mng)	73884	4171	629K	50K	0.12	1392	1716	**1165**
▷ Burmese (my)	20574	755	576K	46.1K	0.04	209	212	**182**
▷ Min-Nan (nan)	33238	1404	1.2M	65.6K	0.03	61	43	**39**
☐ Dutch (nl)	60206	2626	708K	53.8K	0.08	397	340	**267**
☐ Norwegian (no)	69761	3352	674K	47.8K	0.10	534	513	**379**
☐ Polish (pl)	97325	4526	634K	47.7K	0.15	1741	2641	**1491**
☐ Portuguese (pt)	56167	2394	780K	59.3K	0.07	342	272	**214**
☐ Romanian (ro)	68913	3079	743K	52.5K	0.09	384	359	**256**
☐ Russian (ru)	98097	3987	666K	48.4K	0.15	1128	1309	**812**
☐ Slovak (sk)	88726	4521	618K	45K	0.14	1560	2062	**1275**
☐ Slovene (sl)	83997	4343	659K	49.2K	0.13	1114	1308	**776**
☐ Serbian (sr)	81617	3641	628K	46.7K	0.13	790	961	**582**
☐ Swedish (sv)	77499	4109	688K	50.4K	0.11	843	832	**583**
★ Tamil (ta)	106403	6017	507K	39.6K	0.21	**3342**	6234	3496
▷ Thai (th)	30056	1300	628K	49K	0.05	233	241	**206**
▷ Tagalog (tl)	72416	3791	972K	66.3K	0.07	379	298	**219**
★ Turkish (tr)	90840	4608	627K	45K	0.14	1724	2267	**1350**
☐ Ukranian (uk)	89724	4983	635K	47K	0.14	1639	1893	**1283**
▷ Vietnamese (vi)	32055	1160	754K	61.9K	0.04	197	190	**158**
▷ Chinese (zh)	43672	1653	746K	56.8K	0.06	1064	826	**797**
▷ **Isolating** (avg)	40930	1825	759K	54K	0.05	440	392	**326**
☐ **Fusional** (avg)	73499	3532	689K	51.3K	0.11	842	969	**618**
× **Introflexive** (avg)	87352	4566	650K	49.5K	0.14	1735	2104	**1386**
★ **Agglutinative** (avg)	91051	4687	603K	45K	0.16	1898	3164	**1727**

Table 3: Test perplexities for 50 languages (ISO 639-1 codes sorted alphabetically) in the full-vocabulary prediction LM setup; **Left**: Basic statistics of LM evaluation data (see §4 and §5). **Right**: Results with all three language models in our comparison. Best absolute perplexity scores for each language are in bold, but note that the absolute scores in the KN5 column are not directly comparable to the scores obtained with neural models due to a different handling of OOVs at test time (see §5).

In isolating and some fusional languages (e.g., Vietnamese, Thai, English) the TTR tends to be small: we have a comparatively low number of infrequent words. Agglutinative languages such as Finnish, Estonian, and Korean are on the other side of the spectrum. Introflexive and fusional languages, typically over-represented in prior work (see the discussion in §3), are found in the middle.

This emerges clearly in Figure 1, grouping isolating languages to the left side of the x-axis, followed by fusional languages (Germanic and Romance first to the left, and then Balto-Slavic to the right), and placing agglutinative languages towards the far right. Crucially, TTR is an excellent predictor of LM performance. To measure the correlation between this corpus statistics variable and absolute LM performance, we compute their Pearson's r correlation. We find a strong positive correlation, with a value of $r = 0.83$ and significance $p < 0.001$.

We do observe a strong link between each language's morphological type, and the corresponding perplexity score. A transition in terms of the spectrum of morphological systems (see §3) can be traced again on the y-axis of Figure 1, roughly following the reported LM performance: from isolating, over fusional and introflexive to agglutinative languages. In fact, a correlation exists also between traditional morphological types and LM performance. We assessed its strength with the one-way ANOVA statistical test, obtaining a value of $\eta^2 = 0.37$ and a significance of $p < 0.001$.

Finally, it should be noted that the choice of TTP over other corpus statistics such as vocabulary size is motivated by the fact that the corpora are comparable, and not parallel. Because of this, the variation of V may stem from the contents rather than the intrinsic linguistic properties. As a counter-check, the correlation between V and LM performance is in fact milder, with $r = 0.64$. Yet, notwithstanding the stronger correlation, TTP is unable to explain the results entirely. Only through finer-grained typological features it becomes possible to justify several outliers, as shown in the next subsection.

Fine-Grained Typological Analysis. Among the relevant typological features (see §3 and Table 2), fusion and inflectional synthesis have the largest impact on word-level predictions. In fact, the former determines the word boundaries, whereas the latter regulates the amount of possible morpheme combinations. Consider their effect on the frequency distribution of words, expressed as follows (Zipf, 1949):

$$f = \frac{\frac{1}{k^s}}{\sum_{n=1}^{V} \frac{1}{n^s}} \qquad (6)$$

f is the frequency, k the rank, and $s \geq 0$ the exponent characteristic of the distribution. If high, both typological features enlarge V and s, assigning less probability mass to each word.

Low fusion means a preference for separate words (as in isolating languages such as Vietnamese and Chinese), leading to a smaller vocabulary with less (but more frequent) words. This property, additionally boosted by low inflectional synthesis, facilitates statistical language modeling in isolating languages. Vice versa, high fusion results in preference for concatenation of morphemes or introflection, and consequently sparser vocabularies. Yet, this distinction cannot justify the figures by itself, as it equates agglutinative languages and traditional fusional languages. Here, inflectional synthesis is also at play. Through the statistical test of one-way ANOVA, we found a weak effect of $\eta^2 = 0.09$ for fusion and a medium effect of $\eta^2 = 0.21$ for inflection synthesis.

On the other hand, the fine-grained typological features of exponence and flexivity play a role in the ambiguity of the mapping between morphemes and meanings or grammatical functions. This turns out to be especially relevant for character-aware models. The intuition is that if the mapping is straightforward, injecting character information is more advantageous. To validate this claim, we evaluate the ANOVA between exponence of nouns and verbs and the difference in perplexity between LSTM and Char-CNN-LSTM.[5] We report a weak, although existent, correlation with value $\eta^2 = 0.07$ and $\eta^2 = 0.04$, respectively.

Further Discussion. Importantly, our large-scale multilingual LM study strongly indicates that due to diverse typological profiles, certain languages and language groups are inherently more complex to language-model when relying on established statistical models, even when such models are constructed as widely applicable and (arguably) language-agnostic. This finding supports preliminary results from prior work (Botha and Blunsom, 2014; Adams et al., 2017; Cotterell et al., 2018), and is also backed by insights from linguistic theory on variance of language complexity in general and variance of morphological complexity in specific (McWhorter, 2001; Evans and Levinson, 2009). More broadly and along the same line, earlier research in statistical machine translation (SMT) has also shown that typological factors such as the amount of reordering, the morphological complexity, as well as genealogical relatedness of languages are crucial in predicting success in SMT (Birch et al., 2008; Paul et al., 2009; Daiber, 2018).

Our results indicate that the artificial fixed-

---

[5] Unfortunately no values are available in WALS for the feature of flexivity besides a limited domain.

vocabulary assumption from prior work produces overly optimistic perplexity scores, and its limitation is even more pronounced in morphologically rich languages, which inherently contain a large number of infrequent words due to their productive morphological systems. The typical solution to collect more data (Jozefowicz et al., 2016; Kawakami et al., 2017) mitigates this effect to a certain extent, but stills suffers from the Zipfian hypothesis (1949), and it cannot be guaranteed for resource-poor languages where obtaining sufficient monolingual data is also a challenge (Adams et al., 2017).

Therefore, another solution is to resort to other sources of information which are not purely contextual/distributional. For instance, a promising line of current and future research is to (learn to) exploit subword-level patterns captured in an unsupervised manner (Pinter et al., 2017; Herbelot and Baroni, 2017) or integrate existing morphological generation and inflection tools and regularities (Cotterell et al., 2015; Vulić et al., 2017; Bergmanis et al., 2017) into language models to reduce data sparsity, and improve language modeling for morphologically rich languages. For instance, a recent enhancement of the Char-CNN-LSTM language model that enforces similarity between parameters of morphologically related words leads to large perplexity gains across a large number of languages, with the most prominent gains reported for morphologically complex languages (Gerz et al., 2018).

Given the recent success and improved performance with LM-based pre-training methodology (Peters et al., 2018; Howard and Ruder, 2018) across a wide variety of syntactic and semantic NLP tasks in English, improving language models for other languages might have far-reaching consequences for multilingual NLP in general. Typological information coded in typological databases (Ponti et al., 2018) offer invaluable support to language modeling (e.g., knowledge on word ordering, morphological regularities), but such typologically-informed LM architectures are still non-existent.

7 Conclusion

In this paper, we have run a large-scale study on Language Modeling (LM) across several architectures and a collection of 50 typologically diverse languages. We have demonstrated that typological properties of languages, such as their morphological systems, have an enormous impact on the performance of allegedly "language-agnostic" models. We have found that the corpus statistics most predictive of LM performance is type-to-token ratio (TTR), as demonstrated by their strong Pearson's correlation. In turn, the value of TTR is motivated by fine-grained typological features that define the type of morphological system within a language. In fact, such features affect the word boundaries and the number of morphemes per word, affecting the word frequency distribution for each language.

We have also observed that injecting character information into word representations is always beneficial because this mitigates the above-mentioned sparsity issues. However, the extent of the gain in perplexity partly depends on some typological properties that regulate the ambiguity of the mapping between morphemes (here modeled as character n-grams) and their meaning.

We hope that NLP/LM practitioners will find the datasets for 50 languages put forth in this work along with benchmarked LMs useful for future developments in (language-agnostic as well as typologically-informed) multilingual language modeling. This study calls for next-generation solutions that will additionally leverage typological knowledge for improved language modeling. Code and data are available at: `http://people.ds.cam.ac.uk/dsg40/lmmrl.html`.

Acknowledgements

This work is supported by the ERC Consolidator Grant LEXICAL (no 648909). The authors would like to thank the anonymous reviewers for their helpful suggestions.

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of EACL*, pages 937–947.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*, pages 183–192.

Dik Bakker. 2010. Language sampling. In *The Oxford handbook of linguistic typology*, pages 100–127. Oxford University Press.

Emily M. Bender. 2013. *Linguistic fundamentals for natural language processing: 100 essentials from morphology and syntax*. Morgan & Claypool Publishers.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of CoNLL*, pages 31–39.

Balthasar Bickel and Johanna Nichols. 2013. *Inflectional Synthesis of the Verb*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Alexandra Birch, Miles Osborne, and Philipp Koehn. 2008. Predicting success in machine translation. In *Proceedings of EMNLP*, pages 745–754.

Steven Bird. 2011. Bootstrapping the language archive: New prospects for natural language processing in preserving linguistic heritage. *Linguistic Issues in Language Technology*, 6(4).

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the 8th Workshop on Statistical Machine Translation*, pages 1–44.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of ICML*, pages 1899–1907.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. In *Proceedings of INTERPSEECH*, pages 2635–2639.

Xie Chen, Xunying Liu, Yanmin Qian, MJF Gales, and Philip C Woodland. 2016. CUED-RNNLM: An open-source toolkit for efficient training and evaluation of recurrent neural network language models. In *Proceedings of ICASSP*, pages 6000 –6004.

Ryan Cotterell and Jason Eisner. 2017. Probabilistic typology: Deep generative models of vowel inventories. In *Proceedings of ACL*, pages 1182–1192.

Ryan Cotterell, Sebastian J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of NAACL-HLT*.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-Markov models. In *Proceedings of CoNLL*, pages 164–174.

Joachim Daiber. 2018. *Typologically Robust Statistical Machine Translation*. Ph.D. thesis, University of Amsterdam.

Matthew S Dryer. 1989. Large linguistic areas and language sampling. *Studies in Language. International Journal sponsored by the Foundation "Foundations of Language"*, 13(2):257–292.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Nicholas Evans and Stephen C. Levinson. 2009. The myth of language universals: Language diversity and its importance for cognitive science. *Behavioral and Brain Sciences*, 32(5):429–448.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of EMNLP*, pages 360–368.

Ankur Gandhe, Florian Metze, and Ian Lane. 2014. Neural network language models for low resource languages. In *Proceedings of INTERSPEECH*, pages 2615–2619.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018. Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *Transactions of the ACL*, 6:451–465.

Joshua T. Goodman. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

Edouard Grave, Moustapha Cissé, and Armand Joulin. 2017. Unbounded cache model for online language modeling with open vocabulary. In *Proceedings of NIPS*, pages 6044–6054.

Derek Greene and Padraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of ICML*, pages 377–384.

Martin Haspelmath and Andrea Sims. 2013. *Understanding morphology*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of ACL*, pages 690–696.

Aurelie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of EMNLP*, pages 304–309.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of ACL*, pages 328–339.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. In *Proceedings of ICML*.

Dan Jurafsky and James H. Martin. 2017. *Speech and Language Processing*, volume 3. Pearson.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2017. Learning to create and reuse words in open-vocabulary neural language modeling. In *Proceedings of ACL*, pages 1492–1502.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI*, pages 2741–2749.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *Proceedings of ICASSP*, pages 181–184.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit*, pages 79–86.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Handwritten digit recognition with a back-propagation network. In *Proceedings of NIPS*, pages 396–404.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. *CoRR*, abs/1705.07393.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernández Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP*, pages 1520–1530.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of ACL*, pages 1054–1063.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL*, pages 142–150.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

John McWhorter. 2001. The world's simplest grammars are Creole grammars. *Linguistic Typology*, 5(2):125–66.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*, pages 1045–1048.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. In *Proceedings of EMNLP*, pages 1992–1997.

Thomas Müller, Hinrich Schütze, and Helmut Schmid. 2012. A comparative investigation of morphological language modeling for the languages of the European Union. In *Proceedings of NAACL-HLT*, pages 386–395.

Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, and Anna Korhonen. 2016. Survey on the use of typological information in natural language processing. In *Proceedings of COLING*, pages 1297–1308.

Michael Paul, Hirofumi Yamamoto, Eiichiro Sumita, and Satoshi Nakamura. 2009. On the importance of pivot language selection for statistical machine translation. In *Proceedings of NAACL-HLT*, pages 221–224.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of EMNLP*, pages 102–112.

Frans Plank. 2017. Split morphology: How agglutination and flexion mix. *Linguistic Typology*, 21(2017):1–62.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of SIGIR*, pages 275–281.

Edoardo Maria Ponti, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen. 2018. Modeling language variation and universals: A survey on typological linguistics for natural language processing. *CoRR*, abs/1807.00914.

Edoardo Maria Ponti, Ivan Vulić, and Anna Korhonen. 2017. Decoding sentiment from distributed representations of sentences. In *Proceedings of *SEM*, pages 22–32.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of EACL*, pages 157–163.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, pages 379–389.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. In *Proceedings of the ICML Deep Learning Workshop*.

Martin Sundermeyer, Hermann Ney, and Ralf Schluter. 2015. From feedforward to recurrent LSTM neural networks for language modeling. *IEEE Transactions on Audio, Speech and Language Processing*, 23(3):517–529.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of ACL*, pages 2016–2027.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of ACL*, pages 56–68.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *Proceedings of ACL*, pages 1319–1329.

Hamed Zamani and W. Bruce Croft. 2016. Embedding-based query language models. In *Proceedings of ICTIR*, pages 147–156.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. In *Proceedings of ICLR (Conference Papers)*.

George Kingsley Zipf. 1949. *Human behavior and the principle of least effort: An introduction to human ecology*.

A Fast, Compact, Accurate Model for Language Identification of Codemixed Text

Yuan Zhang, Jason Riesa, Daniel Gillick, Anton Bakalov, Jason Baldridge, David Weiss
Google AI Language
`{zhangyua, riesa, dgillick, abakalov, jridge, djweiss}@google.com`

Abstract

We address fine-grained multilingual language identification: providing a language code for every token in a sentence, including codemixed text containing multiple languages. Such text is prevalent online, in documents, social media, and message boards. We show that a feed-forward network with a simple globally constrained decoder can accurately and rapidly label both codemixed and monolingual text in 100 languages and 100 language pairs. This model outperforms previously published multilingual approaches in terms of both accuracy and speed, yielding an 800x speed-up and a 19.5% averaged absolute gain on three codemixed datasets. It furthermore outperforms several benchmark systems on monolingual language identification.

1 Introduction

Codemixed text is common in user-generated content, such as web articles, tweets, and message boards, but most current language ID models ignore it. Codemixing involves language switches within and across constituents, as seen in these English-Spanish and English-Hindi examples.

(1) dame [$_{NP}$ ese book that you told me about]
Give me this book that you told me about.

(2) [$_{NP}$ aapki profile photo] [$_{VP}$ pyari hai]
Your profile photo is lovely.

Codemixing is the norm in many communities, e.g. speakers of both Hindi and English. As much as 17% of Indian Facebook posts (Bali et al., 2014) and 3.5% of all tweets (Rijhwani et al., 2017) are codemixed. This paper addresses fine-grained (token-level) language ID, which is needed for many multilingual downstream tasks, including syntactic analysis (Bhat et al., 2018), machine translation and dialog systems. Consider this example, which seeks a Spanish translation for the English word *squirrel*:

(3) como se llama un squirrel en español
What do you call a squirrel in Spanish?

Per-token language labels are needed; a system cannot handle the whole input while assuming it is entirely English or Spanish.

Fine-grained language ID presents new challenges beyond sentence- or document-level language ID. Document-level labels are often available in metadata, but token-level labels are not. Obtaining token-level labels for hundreds of languages is infeasible: candidate codemixed examples must be identified and multilingual speakers are required to annotate them. Furthermore, language ID models typically use character- and word-level statistics as signals, but shorter inputs have greater ambiguity and less context for predictions. Moreover, codemixing is common in informal contexts that often have non-standard words, misspellings, transliteration, and abbreviations (Baldwin et al., 2013). Consider (4), a French-Arabic utterance that has undergone transliteration, abbreviation and diacritic removal.

(4) cv bien hmd w enti
ça va bien **al**h**a**md**ullilah** w**a** enti
ça va bien الحمد اللّه وانت
It's going well, thank God, and you?

Language ID models must be fine-grained and robust to surface variations to handle such cases.

We introduce **CMX**, a fast, accurate language ID model for **C**ode**M**i**X**ed text that tackles these challenges. CMX first outputs a language distribution for every token independently with efficient feed-forward classifiers. Then, a decoder chooses labels using both the token predictions and global constraints over the entire sentence. This decoder

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 328–337
Brussels, Belgium, October 31 - November 4, 2018.

produces high-quality predictions on monolingual texts as well as codemixed inputs. We furthermore show how selective, grouped dropout enables a blend of character and word-level features in a single model without the latter overwhelming the former. This dropout method is especially important for CMX's robustness on informal texts.

We also create synthetic training data to compensate for the lack of token-level annotations. Based on linguistic patterns observed in real-world codemixed texts, we generate two million codemixed examples in 100 languages. In addition, we construct and evaluate on a new codemixed corpus of token-level language ID labels for 25k codemixed sentences (330k tokens). This corpus contains examples derived from user-generated posts that contain English mixed with Spanish, Hindi or Indonesian.

Language ID of monolingual text has been extensively studied (Hughes et al., 2006; Baldwin and Lui, 2010; Lui and Baldwin, 2012; King and Abney, 2013), but language ID for codemixed text has received much less attention. Some prior work has focused on identifying larger language spans in longer documents (Lui et al., 2014; Jurgens et al., 2017) or estimating proportions of multiple languages in a text (Lui et al., 2014; Kocmi and Bojar, 2017). Others have focused on token-level language ID; some work is constrained to predicting word-level labels from a single language pair (Nguyen and Doğruöz, 2013; Solorio et al., 2014; Molina et al., 2016a; Sristy et al., 2017), while others permit a handful of languages (Das and Gambäck, 2014; Sristy et al., 2017; Rijhwani et al., 2017). In contrast, CMX supports 100 languages. Unlike most previous work–with Rijhwani et al. 2017 a notable exception–we do not assume a particular language pair at inference time. Instead, we only assume a large fixed set of language pairs as a general constraint for all inputs.

We define and evaluate CMX and show that it strongly outperforms state-of-the-art language ID models on three codemixed test sets covering ten languages, and a monolingual test set including 56 languages. It obtains a 19.5% absolute gain on codemixed data and a 1.1% absolute gain (24% error reduction) on the monolingual corpus. Our analysis reveals that the gains are even more pronounced on shorter text, where the language ID task naturally becomes more difficult. In terms of runtime speed, CMX is roughly 800x faster than existing token-level models when tested on the same machine. Finally, we demonstrate a resource-constrained but competitive variant of CMX that reduces memory usage from 30M to 0.9M.

2 Data

We create synthetic codemixed training examples to address the expense and consequent paucity of token-level language ID labels. We also annotate real-world codemixed texts to measure performance of our models, understand code-mixing patterns and measure the impact of having such examples as training data.

Synthetic data generation from monolingual text. For training models that support hundreds of languages, it is simply infeasible to obtain manual token-level annotations to cover every codemixing scenario (Rijhwani et al., 2017). However, it is often easy to obtain sentence-level language labels for monolingual texts. This allows projection of sentence-level labels to each token, but a model trained only on such examples will lack codemixed contexts and thus rarely switch within a sentence. To address this, we create synthetic training examples that mix languages within the same sequence.

To this end, we first collect a monolingual corpus of 100 languages from two public resources: the W2C corpus[1] and the Corpus Crawler project.[2] Then we generate a total of two million synthetic codemixed examples for all languages.

In generating each training example, we first sample a pair of languages uniformly.[3] We sample from a set of 100 language pairs, mainly including the combination of English and a non-English language. The full set is listed in the supplemental material. Then we choose uniformly between generating an *intra-mix* or *inter-mix* example, which are two of the most prominent types of codemixing in the real world (Barman et al., 2014; Das and Gambäck, 2014).[4] An *intra-mix* sentence like (1) starts with one language and switches to another language, while an *inter-mix* sentence like (2) has

---

[1] `http://ufal.mff.cuni.cz/w2c`

[2] `https://github.com/googlei18n/corpuscrawler`

[3] Both our collected codemixed data and Barman et al. (2014) indicate that more than 95% of codemixed instances are bilingual.

[4] The two types of codemixing have roughly equal proportions in our labeled corpus.

	en/es	en/hi	en/id
Number of tokens	98k	140k	94k
Number of sentences	9.5k	9.9k	5.3k

Table 1: Statistics of our YouTube and Google+ dataset, GY-Mix.

Test Set	Languages
Twitter-Mix	en, es
Web-Mix6	cs, en, eu, hu, hr, sk
GY-Mix	en, es, hi, id
KB-Mono56	56 languages

Table 2: The languages of each testing corpora in our experiments. The first three sets primarily include codemixed texts while the last one (KB-Mono56) is monolingual.

an overall single language with words from a second language in the middle. To generate an example, we uniformly draw phrases from our monolingual corpus for the chosen target languages, and then concatenate or mix phrases randomly. The shorter phrase in *inter-mix* examples contains one or two tokens, and the maximum length of each example is eight tokens.

Manual annotations on real-world codemixed text. We obtain candidates by sampling codemixed public posts from Google+[5] and video comments from YouTube,[6] limited to three language pairs with frequent code switching: English-Spanish, English-Hindi[7] and English-Indonesian. All texts are tokenized and lowercased by a simple rule-based model before annotation. Both the candidate selection and the annotation procedures are done by linguists proficient in both languages. The final annotated corpus contains 24.7k sentences with 334k tokens; 30% are monolingual, 67% are bilingual and 3% have more than two languages. Finally, we create an 80/10/10 split (based on tokens) for training, development and testing, respectively. Table 1 gives the token and sentence counts per language. In the rest of the paper, we refer to this dataset as GY-Mix.

Evaluation datasets. We evaluate on four datasets, three codemixed and one monolingual. For a fair comparison, we report accuracies on subsets of these test sets that include languages supported by all tested models. Examples with Hindi words written in Latin script are also removed because the benchmark systems we compare to do not support it.

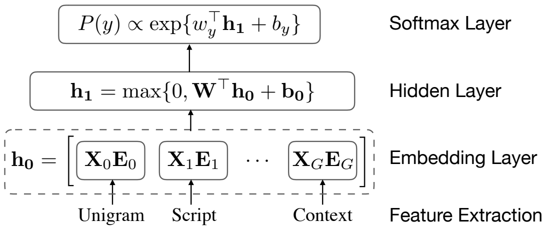

Figure 1: Basic feed-forward network unit for scoring each token in the input and predicting possible languages. Multiple features are embedded, concatenated, and fed into a hidden layer with ReLU activation.

- **Twitter-Mix**: Codemixed data from the EMNLP 2016 shared task (Molina et al., 2016b).
- **Web-Mix6**: Codemixed data crawled from multilingual web pages (King and Abney, 2013), using a subset of six languages.
- **GY-Mix**: The test set of our token-level codemixed data (en-es, en-hi, and en-id).
- **KB-Mono56:** Monolingual test set of Kocmi and Bojar (2017), using a subset of 56 languages.

Table 2 summarizes the final language setting of each test set used in our experiments.

3 Identifying Language Spans in Codemixed Text

CMX uses two stages to assign language codes to every token in a sentence. First, it predicts a distribution over labels for each token independently with a feed-forward network that uses character and token features from a local context window. Then, it finds the best assignment of token labels for an entire sentence using greedy search, subject to a set of global constraints. Compared to sequence models like CRFs or RNNs, this two-stage strategy has several major advantages for fine-grained language ID: (1) it does not require annotated codemixed text over hundreds of languages and their mixed pairings, (2) learning independent classifiers followed by greedy decoding

---

[5] `https://plus.google.com/`

[6] `https://www.youtube.com/`

[7] Hindi texts found in both Devanagari and Latin scripts.

Features	Window	D	V
Character n-gram	+/- 1	16	1000-5000
Script	0	8	27
Lexicon	+/- 1	16	100

Table 3: Feature spaces of CMX. The window column indicates that CMX uses character n-gram and lexicon features extracted from the previous and following tokens as well as the current one.

is significantly faster than structured training (especially considering the large label set inherent in language ID), and (3) it is far easier to implement.

3.1 Token Model

Simple feed-forward networks have achieved near state-of-the-art performance in a wide range of NLP tasks (Botha et al., 2017; Weiss et al., 2015). CMX follows this strategy, with embedding, hidden, and softmax layers as shown in Figure 1. The inputs to the network are grouped feature matrices, e.g. character, script and lexicon features. Each group g's features are represented by a sparse matrix $\mathbf{X}_g \in \mathbb{R}^{F_g \times V_g}$, where F_g is the number of feature templates and V_g is the vocabulary size of the feature group. The network maps sparse features to dense embedding vectors and concatenates them to form the embedding layer:

$$\mathbf{h}_0 = vec[\mathbf{X}_g \mathbf{E}_g | \forall g] \quad (1)$$

where $\mathbf{E}_g \in \mathbb{R}^{V_g \times D_g}$ is a learned embedding matrix per group. The final size of the embedding layer $|\mathbf{h}_0| = \sum_g F_g D_g$ is the sum of all embedded feature sizes. CMX uses both discrete and continuous features. We use a single hidden layer with size 256 and apply a rectified linear unit (ReLU) over hidden layer outputs. A final softmax layer outputs probabilities for each language. The network is trained per-token with cross-entropy loss.

We explain the extraction process of each feature type below. Table 3 summarizes the three types of features and their sizes used in CMX. Character and lexicon features are extracted for the previous and following tokens as well as the current token to provide additional context.

Character n-gram features We apply character n-gram features with $n = [1, 4]$. RNNs or CNNs would provide more flexible character feature representations, but our initial experiments did not show significant gains over simpler n-gram features. We use a distinct feature group for each n. The model averages the embeddings according to the fractions of each n-gram string in the input token. For example, if the token is *banana*, then one of the extracted trigrams is *ana* and the corresponding fraction is 2/6. Note that there are six trigrams in total due to an additional boundary symbol at both ends of the token.

Following Botha et al. (2017), we use feature hashing to control the size V and avoid storing a big string-to-id map in memory during runtime. The feature id of an n-gram string x is given by $\mathcal{H}(x) \bmod V_g$ (Ganchev and Dredze, 2008), where $\mathcal{H}$ is a well-behaved hash function. We set $V = 1000, 1000, 5000, 5000$ for $n = 1, 2, 3, 4$ respectively; these values yield good performance and are far smaller than the number of n-gram types.

Script features Some text scripts are strongly correlated with specific languages. For example, Hiragana is only used in Japanese and Hangul is only used in Korean. Each character is assigned one of the 27 types of scripts based on its unicode value. The final feature vector contains the normalized counts of all character scripts observed in the input token.

Lexicon features This feature group is backed by a large lexicon table which holds a language distribution for each token observed in the monolingual training data. For example, the word *mango* occurs 48% of the time in English documents and 18% in Spanish ones. The table contains about 6.2 million entries. We also construct an additional prefix table of language distributions for 6-gram character prefixes. If the input token matches an entry in the lexicon table (or failing that, the prefix table), our model extracts the following three groups of features.

- *Language distribution.* The language distribution itself is included as the feature vector.
- *Active languages.* As above, but feature values are set to 1 for all non-zero probabilities. For example, the word *mango* has feature value 1 on both English and Spanish.
- *Singletons.* If the token is associated with only one language, return a one-hot vector whose only non-zero value is the position indicating that language.

The size of all lexicon feature vectors is equal to the number of supported languages.

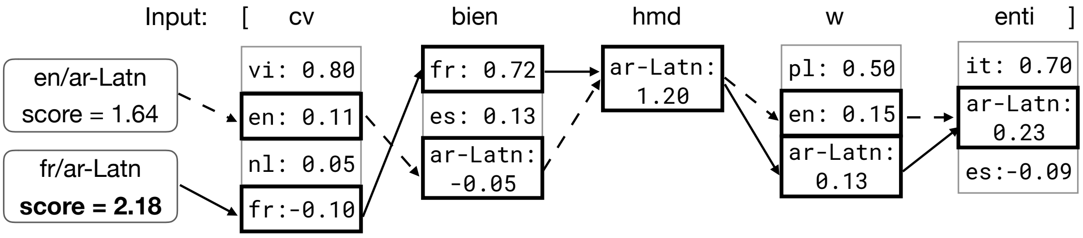

Figure 2: Example of our decoding algorithm with global constraints for example (4) for two allowed language pairs, en/ar-Latn and fr/ar-Latn.

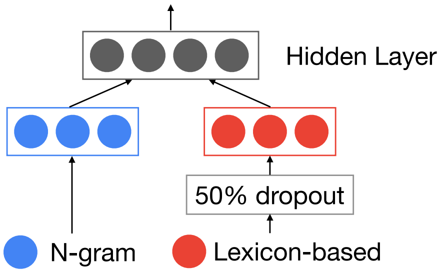

Figure 3: Our selective feature dropout method. The model randomly sets the lexicon feature vectors to zero with 50% probability while n-gram features are always used.

3.2 Selective Feature Dropout

Preliminary experiments showed poor performance, especially on informal texts, when all three types of features are simply merged. Consider the following example outputs on misspelled word *Ennnglish*, for which no lexicon features fire.

Input: *Ennnglish*

With Lexicon Features	*W/o Lexicon Features*
• $p(sv) = 0.27$	• $p(en) = 0.74$
• $p(da) = 0.24$	• $p(nl) = 0.10$
• $p(nl) = 0.18$	• $p(fy) = 0.06$
• …	• …

Without dropout, the model with lexicon features does not make effective use of the token's character n-grams and makes a catastrophically wrong prediction. The core problem is that lexicon features are both prevalent and highly predictive for language ID; during training, this dampens the updating of weights of n-gram features and thus diminishes their overall utility.

To address this issue and make CMX more robust to noisy inputs, we selectively apply a grouped feature dropout strategy that stochastically down-weights lexicon features during training. Figure 3 illustrates the idea: for each input, after feature extraction, the vector of lexicon features is randomly set to zero. This way, the model must rely entirely on n-gram features for this particular input. Note that our feature dropout is different from standard dropout in at least two ways: (1) dropout happens to entire feature groups rather than on individual neurons, (2) we selectively apply dropout only on a subset of features. After tuning the dropout rate on development data (Figure 5) we choose a dropout rate of 50%. Section 4.3 explains the tuning procedure.

3.3 Decoding with Global Constraints

Given a trained model, the goal of decoding is to find the sequence of per-token languages that maximizes the overall score. The simple, greedy strategy of picking the top prediction for each token over-predicts too many languages in a single sentence. For example, on average the greedy method predicts more than 1.7 languages per sentence on monolingual inputs. Because the token classier uses a window including only the previous, current, and next token, it has a quite limited view on the entire sequence.

Motivated by this observation, we add the following **global constraint** in decoding: only monolingual outputs or codemixed outputs from a fixed set of language pairs are permitted. We choose a set of 100 language pairs, primarily including the combination of English and a non-English language. The full set is listed in the supplemental material.

Finally, we introduce a straightforward variant of greedy decoding that finds the optimal language assignment in the presence of these global constraints. We independently find the best assignment under each allowed language combination (monolingual or language pair) and return the one with the highest score.

Figure 2 shows paths for example (4) with two allowed language pairs: en/ar-Latn and

DATASET	Twitter-Mix	Web-Mix6	GY-Mix				Average
LANGUAGES	es/en	6 Langs	es/en	hi/en	hi/hi-Latn/en	id/en	
LanideNN	71.3	52.1	65.7	79.6	–	22.9	58.3
EquiLID	87.9	63.5	71.0	81.9	–	64.9	73.9
CMX-small	88.8	91.0	89.9	98.2	85.0	86.7	90.9
CMX	**92.4**	**93.2**	**91.8**	**98.4**	**87.4**	**91.1**	**93.4**

Table 4: **Codemixed Texts:** Token-level accuracy (%) of different approaches on codemixed texts. "CMX-small" corresponds to our small model without lexicon features and vocabulary tables. The *hi/hi-Latn/en* column shows the accuracy on texts in English, Latin Hindi and Devanagari Hindi; the baseline models do not support identification of text in Hindi in Latin script. *Average* shows averaged accuracy on all sets except *hi/hi-Latn/en*. Boldface numbers indicate the best accuracy for each testing set.

fr/ar-Latn.[8] The two paths in dashed and solid lines indicate the best assignment for each language pair respectively. Because scoring is independent across tokens, each subtask is computed in $O(N)$ time. The total decoding time is $O(N|\mathcal{L}|)$ where $\mathcal{L}$ is the constraint set, and the global optimality of this algorithm is guaranteed because the assignment found in each subtask is optimal.

4 Experiments

4.1 Training Setup

We train CMX on the concatenation of three datasets: (a) GY-Mix's training portion, (b) synthetic codemixed data and (c) a monolingual corpus that covers 100 languages. Every token in the training set spawns a training instance. Our training set consists of 38M tokens in total, which is on the same magnitude as the sizes of training data reported in previous work (Jurgens et al., 2017; Joulin et al., 2016).

We use mini-batched averaged stochastic gradient descent (ASGD) (Bottou, 2010) with momentum (Hinton, 2012) and exponentially decaying learning rates to learn the parameters of the network. We fix the mini-batch size to 256 and the momentum rate to 0.9. We tune the initial learning rate and the decay step using development data.

4.2 Main Results

Codemixed Texts Table 4 lists our main results on the codemixed datasets. We primarily compare our approach against two benchmark systems: EquiLID (Jurgens et al., 2017) and LanideNN (Kocmi and Bojar, 2017). Both achieved state-of-the-art performance on several monolingual and codemixed language ID datasets. LanideNN makes a prediction for every character, so we convert its outputs to per-token predictions by a voting method over characters in each word. For both benchmarks, we use the public pre-trained model provided by the authors. The EquiLID model uses 53M parameters, LanideNN uses 3M, and CMX only uses 0.28M parameters.[9]

Across all datasets, CMX consistently outperforms both benchmark systems by a large margin. On average, our full model (CMX) is 19.5% more accurate than EquiLID (93.4% vs. 73.9%); the gain is even larger compared to LanideNN. Note that none of the models are trained on the Twitter-Mix or the Web-Mix6 dataset, so these two datasets provide an evaluation on the out-domain performance of each approach. In this setting CMX also yields significant improvement in accuracy, e.g. a 4.5% (absolute) gain over EquiLID on the Twitter-Mix dataset.

An Even Smaller Model We further compare between CMX and a variant we call CMX-small, which has no access to lexicon resources or lexicon features. This smaller variant has only 237k parameters and reduces the memory footprint from 30M to 0.9M during runtime, while the (average) loss on accuracy is only 2.5%. This comparison demonstrates that our approach is also an excellent fit for resource-constrained environments, such as on mobile phones.

Monolingual Texts In addition to EquiLID and LanideNN, we further compare CMX against

---

[8] Scores are sorted. Some languages omitted for illustration purposes.

[9] We explain how we compute the number of parameters of our model in the supplemental material.

MODEL	Sent Acc.	Char/Sec
CODEMIXING MODELS		
LanideNN	94.6	0.17k
EquiLID	95.1	0.25k
CMX-small	94.6	265.5k
CMX	**96.6**	206.1k
MONOLINGUAL MODELS		
Langid.py	92.8	183.8k
fastText-small	92.5	2,671.1k
fastText-full	94.4	2,428.3k
CLD2	95.5	4,355.0k

Table 5: **Monolingual Texts:** Sentence-level accuracy (%) on KB-Mono56. Monolingual models make per-sentence predictions only.

Langid.py (Lui and Baldwin, 2012), CLD2[10] and fastText (Joulin et al., 2016, 2017)—all are popular off-the-shelf tools for monolingual language ID. Sentence-level predictions for EquiLID and LanideNN models are obtained by simple voting. Table 5 summarizes sentence-level accuracy of different approaches on the KB-Mono56 test set. CMX achieves the best sentence-level accuracy over all monolingual and codemixing benchmark systems. The resource-constrained CMX-small also performs strongly, obtaining 94.6% accuracy on this test set.

Our approach also maintains high performance on very short texts, which is especially important for many language identification contexts such as user-generated content. This is demonstrated in Figure 4, which plots the cumulative accuracy curve on KB-Mono56 over sentence length (as measured by the number of non-whitespace characters). For example, points at x=50 show the averaged accuracies over sentences with no more than 50 characters. We compare CMX against the best performing monolingual and codemixing benchmark systems. The relative gain is more prominent on shorter sentences than on longer ones. For example, the improvement is 4.6% on short sentences ($\leq$30 characters), while the gain on segments $\leq$150 characters is 1.9%. Similar patterns are seen with respect to other systems.

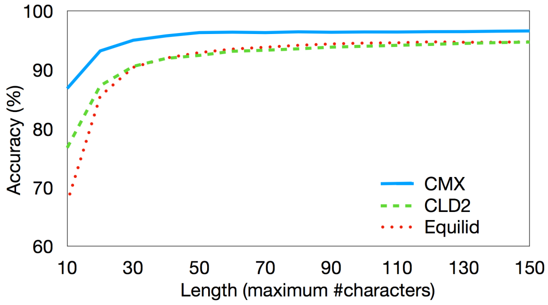

Figure 4: Sentence-level accuracy (y-axis) on KB-Mono56 as a function of the maximum number of non-whitespace characters in a sentence (x-axis). For example, the point at $x = 50$ denotes the accuracy on all the sentences with $\leq$ 50 characters.

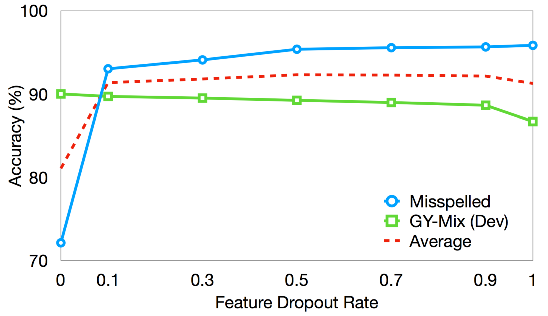

Figure 5: Accuracy on development sets with various feature dropout rate values p.

Inference Speed Table 5 also shows the inference speed of each method in characters per second, tested on a machine with a 3.5GHz Intel Xeon processor and 32GB RAM. CMX (written in C++) is far faster than other fine-grained systems, e.g. it has an 800x speed-up over EquiLID. It is not surprising that monolingual models are faster than CMX, which makes a prediction for every token rather than once for the entire sequence. Of course, monolingual models do not support language ID on codemixed inputs, and furthermore CMX performs the best even on monolingual texts.

4.3 Analysis

Feature Dropout Rate To analyze how the feature dropout rate impacts the model performance, we create a set of *synthetically misspelled* tokens by random duplication or replacement of one or two characters. In addition, we ensure that every token has at least one language-unique character, so a model with character n-gram features should be able to easily identify the language of this token. Figure 5 shows the tuning results for dropout values on misspelled tokens and the GY-Mix development set. Without feature dropout (p=0.0), our model only gets 72.1% on misspelled tokens,

---

[10] `https://github.com/CLD2Owners/cld2`

TRAINING DATA	Twitter-Mix	Web-Mix6	GY-Mix (Test)	KB-Mono56
All Training Corpora	92.4	93.2	93.6	95.1
w/o GY-Mix (Train)	88.5	92.9	89.3	95.0
w/o Synthetic	92.1	88.8	92.5	95.1

Table 6: Token-level accuracy of our full model with different training sets, removing either GY-Mix annotations or synthetic codemixed corpus at a time.

METHOD	GY-Mix	KB-Mono56	#Lang/Sent
Independent	87.8	91.9	1.78
Switching Penalty	89.4	93.1	1.58
Bilingually Constrained	93.6	95.1	1.27
Gold	-	-	1.15

Table 7: Token-level accuracy of different decoding methods on GY-Mix and KB-Mono56, as well as the averaged number of predicted languages in each sentence.

indicating that n-gram features are not properly trained. The proposed feature dropout method effectively addresses this issue, improving the accuracy to 95.3% with $p \geq 0.5$. We choose $p = 0.5$ (Figure 3) because it gives the best trade-off on the two tuning sets. The curves in Figure 5 also show that model performance is robust across a wide range of dropout rates between the two extremes, so the strategy is effective, but is not highly sensitive and does not require careful tuning.

Impact of Decoding Algorithm Table 7 shows a comparison over different decoding strategies, including (a) *independent* greedy prediction for each token, (b) adding a *switching penalty* and decoding with Viterbi, (c) and our *bilingually constrained* decoding. For the second method, we add a fixed transition matrix that gives a penalty score $\log p$ for every code switch in a sentence. We choose $p = 0.5$, which gives the best overall results on the development set. Our approach outperforms *switching penalty* by more than 2% on both GY-Mix and KB-Mono56. To analyze the reason behind this difference we show the average number of languages in each sentence in Table 7. Both baseline approaches on average predict more than 1.5 languages per sentence while the oracle number based on gold labels is only 1.15. Our global bilingual constraints effectively address this over-prediction issue, reducing the average number of predicted languages to 1.27. We also measure the running time of all methods. The decoding speed of our method is 206k char/sec (Table 5), while the *independent* method is 220k char/sec. Our decoding with global constraints thus only increases the running time by a factor of 1.07.

Codemixed Training Datasets Our training data consists of two codemixed corpora: manual annotations on real-world data (GY-Mix) and a synthetic corpus. To analyze their contribution, we remove each corpus in turn from the training set and report the results in Table 6. Adding the GY-Mix training set mainly improves accuracy on GY-Mix test and Twitter-Mix, while the gains from the synthetic data are greatest on Web-Mix6. This shows that synthetic data helps CMX generalize to a broader range of languages since GY-Mix has language overlap only with Twitter-Mix, not Web-Mix6. The two examples below further demonstrate the benefit of synthetic examples:

With Synthetic Data

- [Translate $]_{\text{en}}$ [**maçã** $]_{\text{pt}}$ [to English $]_{\text{en}}$
- [Translate $]_{\text{en}}$ [**Apfel** $]_{\text{de}}$ [to English $]_{\text{en}}$

Without Synthetic Data

- [Translate **maçã** to $]_{\text{pt}}$ [English $]_{\text{en}}$
- [Translate **Apfel** to English $]_{\text{en}}$

Both examples are likely potential queries "Translate *apple* to English" with *apple* replaced by its translation in German(de) or Portuguese(pt). The underlying language pairings never appear in GY-Mix. CMX with synthetic training data is able to correctly identify the single token inter-mixed in a sentence, while the model trained without synthetic data fails on both cases.

Contribution of Features CMX has three types of features: character n-gram, script, and lexicon features. n-gram features play a crucial role as back-off from lexicon features. Consider informal Latin script inputs, like *hellooooo*, for which no lexicon features fire. Foregoing n-gram features results in abysmal performance (<20%) on this type of input because script features alone are inadequate. The main impact of script features is to avoid embarrassing mistakes on inputs that can be easily identified from their scripts. Finally, note that removing lexicon features corresponds to the CMX-small model. On monolingual inputs (Table 5), the lexicon features in CMX provide a 2.0% absolute improvement in accuracy.

5 Conclusions

CMX is a fast and compact model for fine-grained language identification. It outperforms related models on codemixed and monolingual texts, which we show on several datasets covering text in a variety of languages and gathered from diverse sources. Furthermore, it is particularly robust to the idiosyncrasies of short informal text.

Acknowledgments

We thank Emily Pitler, Slav Petrov, John Alex, Daniel Andor, Kellie Webster, Vera Axelrod, Kuzman Ganchev, Jan Botha, and Manaal Faruqui for helpful discussions during this work, and our anonymous reviewers for their thoughtful comments and suggestions. We also thank Elixabete Gomez, Héctor Alcalde, Knot Pipatsrisawat, and their stellar team of linguists who helped us to annotate and curate much of our data.

References

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 356–364.

Timothy Baldwin and Marco Lui. 2010. Language identification: The long and the short of the matter. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the ACL*, pages 229–237.

Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. 2014. I am borrowing ya mixing, an analysis of english-hindi code mixing in facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 116–126, Doha, Qatar. Association for Computational Linguistics.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 13–24.

Irshad Ahmad Bhat, Riyaz A. Bhat, and Manish Shrivastava. 2018. Universal dependency parsing for hindi-english code-switching. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

Jan A Botha, Emily Pitler, Ji Ma, Anton Bakalov, Alex Salcianu, David Weiss, Ryan McDonald, and Slav Petrov. 2017. Natural language processing with small feed-forward networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2879–2885.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 378–387.

Kuzman Ganchev and Mark Dredze. 2008. Small statistical models by random feature mixing. In *Proceedings of the ACL-08: HLT Workshop on Mobile Language Processing*, pages 19–20.

Geoffrey E Hinton. 2012. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer.

Baden Hughes, Timothy Baldwin, Steven Bird, Jeremy Nicholson, and Andrew MacKinlay. 2006. Reconsidering language identification for written language resources. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 485–488.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*, pages 427–431.

David Jurgens, Yulia Tsvetkov, and Dan Jurafsky. 2017. Incorporating dialectal variability for socially equitable language identification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 51–57.

Ben King and Steven Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proccedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tom Kocmi and Ondřej Bojar. 2017. Lanidenn: Multilingual language identification on character window. In *European Conference of the Association for Computational Linguistics*.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012: System Demonstrations*, pages 25–30. Association for Computational Linguistics.

Marco Lui, Jey Han Lau, and Timothy Baldwin. 2014. Automatic detection and language identification of multilingual documents. *Transactions of the Association for Computational Linguistics*, pages 28–40.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016a. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 40–49, Austin, Texas. Association for Computational Linguistics.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016b. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 40–49.

Nguyen and A. Seza Doğruöz. 2013. Word level language identification in online multilingual communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 857–862.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. 2017. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1971–1982, Vancouver, Canada. Association for Computational Linguistics.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72, Doha, Qatar. Association for Computational Linguistics.

Nagesh Bhattu Sristy, N. Satya Krishna, B. Shiva Krishna, and Vadlamani Ravi. 2017. Language identification in mixed script. In *FIRE '17: Forum for Information Retrieval Evaluation*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, ACL 2015*.

Personalized Microblog Sentiment Classification via Adversarial Cross-lingual Multi-task Learning

Weichao Wang[1] , **Shi Feng**[1] , **Wei Gao**[2] , **Daling Wang**[1] , **Yifei Zhang**[1]
[1]School of Computer Science and Engineering, Northeastern University, Shenyang, China
`wangwecha@gmail.com, {fengshi,wangdaling,zhangyifei}@cse.neu.edu.cn`
[2]Victoria University of Wellington, Wellington, New Zealand
`wei.gao@vuw.ac.nz`

Abstract

Sentiment expression in microblog posts can be affected by user's personal character, opinion bias, political stance and so on. Most of existing personalized microblog sentiment classification methods suffer from the insufficiency of discriminative tweets for personalization learning. We observed that microblog users have consistent individuality and opinion bias in different languages. Based on this observation, in this paper we propose a novel user-attention-based Convolutional Neural Network (CNN) model with adversarial cross-lingual learning framework. The user attention mechanism is leveraged in CNN model to capture user's language-specific individuality from the posts. Then the attention-based CNN model is incorporated into a novel adversarial cross-lingual learning framework, in which with the help of user properties as bridge between languages, we can extract the language-specific features and language-independent features to enrich the user post representation so as to alleviate the data insufficiency problem. Results on English and Chinese microblog datasets confirm that our method outperforms state-of-the-art baseline algorithms with large margins.

1 Introduction

The past decade has witnessed an exploding interest in microblog sentiment classification from both academic and commercial communities due to its inherent challenges and wide applications (Liu, 2012; Jawa and Hasija, 2015). However, most existing algorithms largely ignore latent personal distinctions among microblog users (Wu et al., 2015; Sun et al., 2015; Yu et al., 2015). In fact, the sentiment expression in posts can be affected by user's language habit, personal character, opinion bias and so on, and even the same sentence can deliver quite different sentiment polarities depending on user's underlying context.

Table 1 illustrates a simple case where two users are talking about house price. In the first two example posts, we can see that the **Seller**, as a real estate agent, is excited about the growth of the price, while as a consumer, the **Buyer** feels sad for the situation. Without knowing the users' background, it would be difficult for the traditional methods to detect the sentiment polarities of posts 1 and 2.

In recent years, personalized sentiment classification considering user's individuality has emerged as one of the prevailing research topics in the field. Song et al. (2015b, 2016) utilized a latent factor model to capture personal sentiment variations of microblog users. Zhao et al. (2017) exploited the social relation information to facilitate user representation learning. In rating prediction research, Tang et al. (2015) embedded user and product level information into a neural network model. Chen et al. (2016a) further improved the model by incorporating global user and product information as attentions into a hierarchical LSTM model. These studies have achieved promising results on personalized sentiment classification. However, most of them suffer from the insufficiency of discriminative posts for personalization learning.

We observed that the microblog users have consistent individuality and opinion bias in different languages on different platforms. Posts 3 and 4 in Table 1 are posted by the same couple of **Seller** and **Buyer** in Chinese on Sina Weibo[1]. Both of them expressed their own stance or opinion across the two platforms as consistently as what each has said in English on Twitter, indicating the stable underlying individuality of each user. Jointly analyzing microblog posts in different languages can help us enrich the user features and ex-

---

[1] `www.weibo.com`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 338–348
Brussels, Belgium, October 31 - November 4, 2018.

ID	Source	Post	Label
1	Twitter	**Seller**: The house prices are getting higher and higher!	(positive)
2	Twitter	**Buyer**: The house prices are getting higher and higher!	(negative)
3	Weibo	**Seller**: 房价一直在提高，我们会有不错的效益。 (*The house prices have been rising and we will have good benefits.*)	(positive)
4	Weibo	**Buyer**: 过高的房价已经超出了我的承受范围。 (*High house prices have exceeded my tolerance.*)	(negative)

Table 1: A toy example of microblogs of the same users in English and Chinese. Note that posts 3 and 4 are brand new and not the translation of posts 1 and 2.

tract more precise personalized view and opinion bias. Furthermore, the microblog posts on Twitter and Weibo share many similarities in social and textual styles regardless of different languages used. By cross-lingual learning from the same (or similar) users' posts, we can extract the common language-independent features to enhance the representations for both monolingual and cross-lingual sentiment classification. We hypothesize that general cross-lingual sentiment words such as adjectives have similar effect on the sentiment of sentences in different languages, which can be considered generally independent of language. It is thus meaningful to leverage the individuality features to bridge the language gap and exploit the global language-independent sentiment knowledge learned to help alleviate data insufficiency for personalized sentiment classification.

Convolutional neural networks (CNN) are effective in short text analysis thanks to the n-grams of text that contain rich and accurate information for capturing user's attention. Besides, adversarial multi-task learning provides us with a feasible method to bridge the semantics of posts of different languages by utilizing the correlation between the two related language-specific tasks. In this paper, we propose a user-attention-based CNN model with adversarial multi-task learning (dubbed as ACNN-AMT). Firstly, we incorporate user attention into CNN model to capture user's language-specific individuality from the posts. Secondly, we propose an adversarial multi-task learning framework to co-train cross-lingual sentiment features with the help of user's consistent individuality. The main contributions of our paper are three-fold:

- We propose a novel user attention mechanism over n-gram semantics of short posts to capture the language-specific individuality and opinion bias of users for personalized sentiment classification.
- We incorporate the user-attention-based CNN model into a novel adversarial cross-lingual multi-task learning framework. With the help of consistency of user individuality as bridge across two languages, we learn language-specific features and language-independent features to enrich the user post representation, and alleviate the problem of insufficiency of personal data.
- We conduct comprehensive experiments on real-world Sina Weibo datasets and Twitter datasets to confirm the effectiveness of our proposed model. The experimental results demonstrate that our model consistently outperforms strong baseline methods by a large margin. Our datasets have been made publicly available[2].

2 Related Work

Recently, personalized sentiment classification has attracted lots of attention in research community. Song et al. (2015b) proposed a personalized sentiment classification method based on latent factor model to capture personal sentiment variations. Tang et al. (2015) was first to incorporate user and product information into a neural network model for personalized rating prediction of products. Wu and Huang (2016) proposed a personalized microbolg sentiment classification framework via multi-task learning to capture the personal properties and take advantage of common sentiment knowledge shared among all users. Chen et al. (2016a) adopted a hierarchical neural network to incorporate global user and product information into the sentiment model via attention mechanism. Zhao et al. (2017) exploited both the

---

[2] `https://github.com/githubfordata/data`

semantics of posts and the social relations among users for microblog sentiment classification with recurrent random-walk network model. Although these methods have achieved reasonable results, they are limited by insufficient personalized data that the learning of personal sentiment features and user representations has to rely on.

Generative adversarial network (Goodfellow et al., 2014) has been successfully applied to Natural Language Processing, especially domain and language adaptation, including measuring and matching different feature space distribution between source and target. Ganin and Lempitsky (2015) presented an adversarial approach to domain adaptation for transferring knowledge from source domain to target domains. Chen et al. (2016b) proposed an adversarial deep averaging network to transfer sentiment knowledge from labeled source language to low-resource target language where only unlabeled data exists. Liu et al. (2017) tried an adversarial multi-task learning framework for text classification, intending to refrain the shared and private latent feature spaces from interfering each other. These methods only took multi-domain or multi-language setting into account, aiming to address the problem caused by low-resource languages or specific domains, but they ignored the important factors of users across different domains or languages, which is a very important factor in sentiment analysis.

Unlike the previous studies, we incorporate cross-lingual sentiment consistency and personalized user sentiment model from the viewpoint of multi-task learning for alleviating the insufficiency of personal data and improving the performance of personalization in sentiment analysis.

3 Attention-based Personalization Model

In CNN, we can use attention mechanism to predict the importance probability of each component with respect to attention query (Bahdanau et al., 2014). Given a post, the user usually focuses on its specific portions depending on one's individuality or interest. The user attention can be incorporated into CNN model to learn appropriate user-specific features and capture crucial components that the user cares about.

Given a data example (x, u, y) in a monolingual dataset, the post content x is denoted as:

$$x = w_1 \oplus w_2 \oplus \cdots \oplus w_i \oplus \cdots \oplus w_n \qquad (1)$$

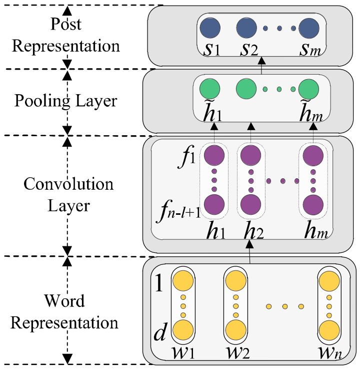

Figure 1: Convolutional Neural Network for post representation.

where n is number of words contained in x, $\oplus$ is the concatenation operator, each word w_i is mapped to its d-dimensional embedding representation $w_i \in \mathbb{R}^d$, the user of the post is represented by a continuous vector of o dimensions $u \in \mathbb{R}^o$ which is initialized as zero-valued vector, and y is the sentiment label of x. Given all training examples, our goal is to learn appropriate text and user representations for classifying sentiment class of different users' new posts.

3.1 CNN-based Text Representation

We use CNN (Kim, 2014; Kalchbrenner et al., 2014) for learning the semantic representation s of input post x, which is shown in Figure 1. The semantic representations of n-grams with sliding window are mapped to fixed and lower dimension via the convolutional filters. Let l be the width of convolutional filter, W and b be the parameters and bias of linear mapping, which is shared among all layers in a filter. Each sliding window with fixed-length window size of l starting from the k-th word is denoted as $\Omega_k = w_k \oplus w_{k+1} \oplus \cdots \oplus w_{k+l-1}$, where $\Omega_k \in \mathbb{R}^{d\times l}$. The convolutional layer output $f_k \in \mathbb{R}^m$ of the window Ω_k is calculated by applying the linear mapping below, following a convolution operation:

$$f_k = W \cdot \Omega_k + b \qquad (2)$$

where $W \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$, and m is the output length of each linear mapping. With the convolution operation, $n - l + 1$ features in total, i.e., one for each window, are obtained and denoted as $\mathbf{f} = [f_1, f_2, \cdots, f_{n-l+1}]^\top$ For each column vector h_j for $j = 1, \ldots, m$ in $\mathbf{f}$, a scalar $\tilde{h}_j$ is calculated as the representation of h_j. By concatenating all scalars, the output vector of a post representation

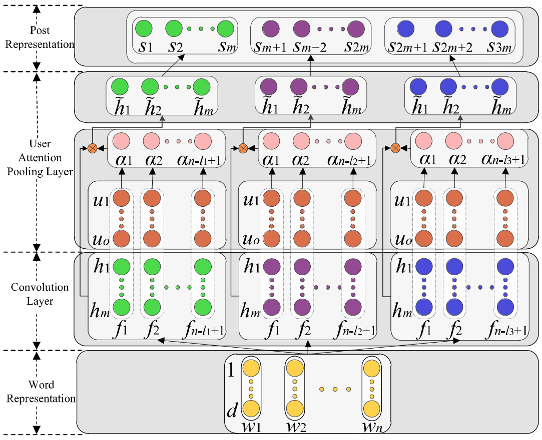

Figure 2: Our user-attention-based CNN model (ACNN) for post representation. The convolution windows (in convolution layer) in green, purple and blue color correspond to three sizes of n-grams used, i.e., l_1, l_2, and l_3, respectively.

$s = [s_1, s_2, \ldots, s_m]$ with fixed length of m is generated. Note that s can be generated from $\tilde{h}$ with the similar linear mapping as Eq. 2, but for simplicity we treat it the same as $\tilde{h}$ in our experiment.

3.2 Incorporating User Attention

The model incorporating user attention mechanism is shown in Figure 2.

The users information are distinguished by user id, and each user u is embedded as vector $u \in \mathbb{R}^o$ for capturing their opinion bias and individuality. All the users embedding will be regarded as attention query parts in user attention mechanism, and the model gives all input windows different attention weights according to the individuality of post users for semantic representation. We treat user embeddings as model parameters, which can be updated with the optimization algorithm in training process (see Section 4).

Formally, the final post representation $s = \tilde{h}$ is a weighted sum of all sliding windows of convolution layer:

$$\tilde{h} = \sum_{k=1}^{n-l+1} \alpha_k f_k \tag{3}$$

where α_k measures the importance of the k-th n-gram (i.e., window) for current user. The attention weight α_k of each window's hidden state can be defined as:

$$\alpha_k = \frac{\exp(\sigma(f_k, u))}{\sum_{j=1}^{n-l+1} \exp(\sigma(f_j, u))} \tag{4}$$

where σ is a scoring function which scores the importance of each n-gram by taking into account both post and user representations.

For each input window, the attention score is calculated for getting the attentive degree of input to current user attention query by combining them together linearly. The scoring function σ is defined as:

$$\sigma(f_k, u) = v \cdot \tanh\left(W_f \cdot f_k + W_u \cdot u + b\right) \tag{5}$$

where $W_f \in \mathbb{R}^{m \times m}$ is a weight matrix for hidden states of input windows of a post, $W_u \in \mathbb{R}^{m \times o}$ is weight matrix for the user of the current post, and $v \in \mathbb{R}^m$ is the weight vector, which is randomly initialized, for getting a score number for the function. v is needed since a matrix $\mathbb{R}^{m \times 1}$ is obtained by $\tanh(\cdot)$, and for getting a scalar, we need a matrix $\mathbb{R}^{1 \times m}$ for matrix multiplication.

To make our model more accurate, we apply three kinds of convolutional filters to capture the semantics of n-grams corresponding to three levels of granularities, i.e., we let $l \in \{l_1, l_2, l_3\}$ corresponding to unigram, bi-gram and tri-gram, respectively, and we concatenate the representations $\tilde{h}$ of these n-grams of these three granularities as the representation of each post. Similar as (Tang et al., 2015), for modeling user-sentiment consistencies, we also concatenate the representation of users and that of posts as the final semantic representation. Note that for the current post, the user embedding is the same among three different granularities, but the model parameters W_f and W_u among the three granularities are different because the convolutional representation of post f_k is dependent on specific granularity of n-grams used.

4 Adversarial Learning for Personalized Sentiment Classification

Multi-task learning can utilize the correlation among different tasks to learn related knowledge for better performance (Evgeniou and Pontil, 2004). The posts of each user are often insufficient for personalized sentiment analysis with a monolingual dataset. By exploiting posts of both Chinese and English in multi-task learning, we can enrich the original monolingual posts of each user to alleviate the data insufficiency problem. Inspired by the adversarial network that can build the mapping between generative distribution and target distribution (Goodfellow et al., 2014),

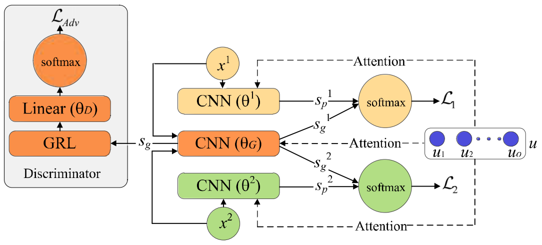

Figure 3: The ACNN-AMT model for sentiment classification. The left hand side is the language discriminator of the adversarial network; the CNNs in the middle are the feature generators of language-specific sentiment classification tasks; and the right hand side are the decision functions of the corresponding sentiment classifiers. Both sentiment classification tasks share the same user representation where the bridge is established via user's consistent individuality across languages. The adversarial learning is realized by gradient reversal layer (GRL).

we conjecture that the sentiment knowledge between two different languages can be mapped to the same feature space to further improve the effect of multi-task learning with a feature generator and a language category discriminator. Formally, given a data sample (x^t, u^t, y^t) from the posts of the t-th language (i.e., task), we aim to predict the sentiment class of the posts with our adversarial cross-lingual multi-task learning method.

4.1 Adversarial Multi-task Learning

Intuitively, a user may post on different microblogging platforms in different languages, but the individuality and opinion bias embedded in the posts of different languages are supposed to be consistent. By regarding user's individuality as bridge between languages, we adopt multi-task learning to utilize the correlation of user's individuality across Chinese and English platforms for learning better user representation.

As shown in Figure 3, we illustrate our user attention based CNN model with adversarial multi-task learning (ACNN-AMT). All tasks share a same collection of users embedding u via the attention mechanism as a cross-lingual bridge. Each task is a sentiment classification model, which is decomposed into two components, i.e., a global ACNN model trained with posts from all languages for extracting common sentiment features (the orange-color CNN model), a language-specific ACNN model which is trained with posts from a specific language (the yellow- or green-color CNN model) for generating language-specific features, and a softmax classifier for predicting sentiment classification. Therefore, this multi-task model is both a sentiment classifier and meanwhile a generator for producing post representations at global and language-specific levels. Both tasks share the same global ACNN model.

For any input post x^t of the t-th task[3], we can compute its global representation s_g^t and task-specific representation s_p^t with the ACNN model. The final features s^t of post x^t are concatenation of the features from the specific and global space and user embedding. Formally, we define s_g^t, s_p^t and s^t as follows:

$$\begin{aligned} s_g^t &= \text{ACNN}(x^t, u^t, \theta_G) \\ s_p^t &= \text{ACNN}(x^t, u^t, \theta^t) \\ s^t &= s_g^t \oplus s_p^t \oplus u^t \end{aligned} \tag{6}$$

where θ_G contain parameters of the global task-independent model (which acts as the generator in the adversarial network), and θ^t contain the model parameters of the t-th task, and u^t is the user embedding of the t-th task which is constant across different tasks.

Because Chinese and English have different lexical and linguistic structures, the global sentiment classifier could learn little knowledge from English for Chinese sentiment classification, and vice versa. Nevertheless, if we could learn sentiment features independent of language category, the global sentiment classifier will be well trained by posts of both languages. To get such language-independent features, we introduce adversarial training into the global model. We formulate the global model as a generator and the language classification model as a discriminator using Generative Adversarial Network (GAN) (Goodfellow et al., 2014). During training, the generator is trying to generate language-independent sentiment features that would confuse the discriminator as to these features' language category, so that the discriminator is forced to struggle to become a strong classifier for correctly predicting the language category of the generated representation, until both the generator and discriminator cannot improve any more. The language discriminative model D parameterized by θ_D and θ_G is defined as follows:

$$\hat{y}(\theta_D, \theta_G) = \text{softmax}(\tanh(W_D s^g(\theta_G) + b_D)) \tag{7}$$

---

[3] For simplicity, we consider a two-task case here, i.e., $t = \{1, 2\}$. But our model is generic which can easily consider more tasks.

where $\hat{y}(\cdot)$ is the predicted language category, θ_D contain parameters $W_D \in \mathbb{R}^{T\times 3m}$ which are the weights (T is number of language categories) and $b_D \in \mathbb{R}^{T\times 3m}$ which are the bias terms, and $s^g(\theta_G) \in \mathbb{R}^{3m}$ is the language-independent sentiment features extracted by global ACNN model parameterized by θ_G.

Finally, we get the adversarial loss by cross-entropy error between gold language distribution and our predicted language distribution:

$$\mathcal{L}_{Adv} = \sum_{t=1}^{T}\sum_{i=1}^{N^t} y_i^t \log(\hat{y}_i^t) \tag{8}$$

where y_i^t denotes the ground-truth language label indicating that the i-th post comes from the t-th task (or language), $\hat{y}_i^t$ is the predicted language of the post, and N^t is the number of posts in the t-th task. We use the gradient reversal layer (GRL) (Ganin and Lempitsky, 2015) to perform the adversarial learning. The objective of GRL is to optimize θ_G and θ_D via the min-max steps as $\min_{\theta_G}\max_{\theta_D}\mathcal{L}_{Adv}$. After the adversarial training, we can obtain stronger language category discriminator and language-independent sentiment features.

4.2 Sentiment Classification

For the t-th task, the predicted post sentiment distribution is defined as follows:

$$\hat{z}^t = \mathrm{softmax}(\tanh(W^t s^t + b^t)) \tag{9}$$

where $W^t \in \mathbb{R}^{C\times(6m+o)}$ are the weights and $b^t \in \mathbb{R}^{(6m+o)}$ are the biases. For each sentiment category, cross-entropy error between gold sentiment distribution and our predicted sentiment distribution is defined as sentiment loss:

$$\mathcal{L}^t = -\sum_{c=1}^{C}\sum_{i=1}^{N_c^t} z_{c,i}^t \log(\hat{z}_{c,i}^t) \tag{10}$$

where $z_{c,i}^t$ denotes the ground-truth label indicating sentiment of the i-th post of the t-th task, $\hat{z}_{c,i}^t$ is the predicted sentiment label of the post, C is the number of sentiment classes, and N_c^t is the number of posts of class c in the t-th task.

Similar to (Liu et al., 2017), we add a squared Frobenius norm term to the loss function:

$$\mathcal{L}_{Frobenius} = \sum_{t=1}^{T}\|{G^t}^{\top}P^t\|_F^2 \tag{11}$$

where $\|\cdot\|_F^2$ is the squared Frobenius norm, G^t and P^t are matrics whose rows are s_g^t and s_p^t of each input sentence, respectively.

The final loss function of our model can be written as:

$$\mathcal{L} = \beta\mathcal{L}^1 + \lambda\mathcal{L}^2 + \gamma\mathcal{L}_{Adv} + \delta\mathcal{L}_{Frobenius} \tag{12}$$

where $\mathcal{L}^1$ and $\mathcal{L}^2$ are the loss functions of sentiment classification models of the two languages, and the hyper parameters β, λ, γ and δ are the weights of their corresponding losses.

To minimize the overall loss function $\mathcal{L}$, we use Adadelta (Zeiler, 2012) to update the model parameters. In particular, we employ Adadelta to update θ_D to maximize the adversarial loss term $\mathcal{L}_{Adv}$, and update other parameters to minimize the overall loss function $\mathcal{L}$.

5 Experiment

5.1 Experimental Setting

Our experiments are conducted on two real-world microblog sentiment datasets. The first dataset was crawled using our built-in-house crawler from Sina Weibo, which contain only Chinese posts. Weibo users have their personal profile, and their homepage links to Twitter may also appear in it. We extracted the Twitter homepage links for crawling our Twitter dataset via Twitter API, and the dataset contain only English posts. We labeled the posts with sentiment polarity based on frequently used emoticons as emoticons which are given by users were commonly regarded as appropriate personal sentiment labels (Song et al., 2015a,b; Wu and Huang, 2016).

The statistics of the two datasets are summarized in Table 2 and 3, where 96 bilingual users were found appearing in both datasets. In the experiment, we used all users, i.e., 755 from Twitter, and 379 from Weibo, assuming that similar users can be embedded into similar user representation space. The bilingual users will especially bring strong effects on personalization learning because the posts of bilingual users can be better represented with consistent individuality than those of monolingual users.

For each user's posts in different languages, we split them into training, development and testing sets by user volumes under the ratio of 7:2:1 randomly, and used Stanford CoreNLP (Manning et al., 2014) for word segmentation of Chinese and English posts. For evaluation, we use precision,

Dataset	classes	users	posts	positive	negative	posts/user	vocabulary	words/post
Twitter	2	755	41,210	28,782	12,428	54.58	27,211	10.70
Weibo	2	379	40,081	25,367	14,714	105.75	25,702	15.74

Table 2: Statistics of Weibo and Twitter datasets.

Dataset	users	posts	positive	negative	posts/user	vocabulary	words/post
Twitter	96	6,942	4,879	2,063	72.31	10,181	9.84
Weibo		13,563	7,448	6,115	141.28	14,031	15.19

Table 3: Statistics of Weibo and Twitter data of bilingual users.

recall and F-score to measure the classification results with 10-fold cross validation, and take average over ten folds for obtaining the values of the three evaluation metrics.

For Chinese posts, we trained our Word2Vec model (Mikolov et al., 2013) on our crawled 30M Weibo corpus. For English posts, we used the 200d GloVe vectors (Pennington et al., 2014) as word embeddings. We set the dimensionality of user embedding to 100.

The other model parameters were initialized by randomly sampling from uniform distribution in $[-0.1, 0.1]$. We set the mini-batch size to 32. Besides, we employed dropout technique (Srivastava et al., 2014) and the dropout rate was set to 0.5. We tuned the hyper-parameters on the development sets, and empirically set β to 0.8, λ to 1, γ to 0.1, and δ to 0.01. The width of three convolutional filter was fixed to 1, 2, and 3, corresponding to unigram, bi-gram and tri-gram, respectively.

5.2 Baselines

We compared our model with several strong baseline methods for microblog sentiment classification:

SVM + Wordvec: It averages the embeddings of all words in a post, which are regarded as features of a SVM classifer using LibSVM library (Chang and Lin, 2011).

LSTM: It uses traditional LSTM model for sentiment classification.

CNN: It uses CNN model (Kim, 2014) for sentiment classification.

NSC + UPA: It uses user and product attention mechanism over word-level and sentence-level semantics based on LSTM (Chen et al., 2016a). The original method focused on modeling product review texts. Because microblog posts contain users but no particular products information, we implemented the method with word-level modeling and user attention for fair comparison.

UPNN: It uses vector space model for modeling user preferences and product qualities, which is integrated with word-level representation and post-level representation (Tang et al., 2015). Original UPNN models user preference and product characteristic. We modified it to only model user preferences as a baseline method for comparison.

ASP-MTL: It uses LSTM model with adversarial multi-task learning to extract the domain-invariant features for cross-domain text classification (Liu et al., 2017). We regard domain category as language category for cross-lingual sentiment classification.

5.3 Result Comparisons

Table 4 shows the comparison of results among different models. We separate results into two groups which are monolingual results and cross-lingual results respectively.

In the first group, the SVM with averaging word embedding performs poorly because SVM cannot capture semantic information as well as deep neural networks. The CNN model performs better than LSTM, because LSTM model is easier to overfit. The NSC with UPA yields better results than LSTM, and UPNN yields better results than CNN, because they exploit the impact of user information. Our ACNN model archives the best performance by exploiting user information on top of the semantics of n-grams.

In the second group, the ASP-MTL performs better than LSTM, because global sentiment classifier is well trained by language-independent sentiment features with adversarial training. Our ACNN-AMT model significantly outperforms all the other baseline methods, confirming the importance of both user information and global

Models	Twitter			Weibo		
	P	R	F	P	R	F
SVM+Wordvec	77.18	68.74	70.63	72.60	68.07	68.88
CNN	75.60	77.02	76.41	73.57	73.60	73.55
LSTM	74.02	75.15	74.59	71.11	72.27	72.26
NSC+UPA	77.76	77.81	77.74	78.47	78.72	78.57
UPNN	81.02	80.67	80.21	79.29	80.28	79.77
ACNN	81.35	80.43	**80.85**	80.31	79.85	**80.02**
ASP-MTL	75.24	75.28	75.21	72.54	73.48	72.94
ACNN-AMT(full model)	81.90	80.73	**81.24**	82.43	81.74	**82.03**

Table 4: Comparison of microblog sentiment classification results.

language-independent sentiment features.

5.4 Effect of Attention Mechanism and Multi-task Learning

Table 5 shows the effect of user attention in combination with pure CNN and multi-task CNN (CNN-MT) models. We can observe that: 1) The ACNN model and ACNN-MT model yield better sentiment classification results compared with CNN model and CNN-MT model without attention. It demonstrates that our user attention mechanism can capture crucial components for each user. 2) The CNN-MT model improves slightly compared with CNN model, and the reason is that there are substantial differences between Chinese and English languages, curbing from extracting global common sentiment features. The ACNN-MT model yields much better results than ACNN model, confirming that multi-task learning with user to bridge the gap between two languages can help generate more accurate user representation. We can also see that the ACNN-MT model improves more obviously on Weibo dataset. This is because larger proportion of Weibo posts than that of Twitter posts share users cross-lingually.

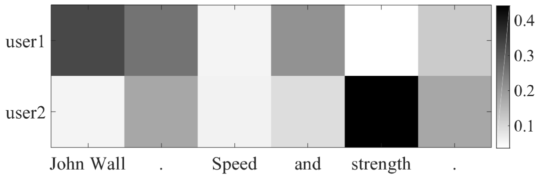

Figure 4: Visualization of attentions over unigram of Twitter. The darker color means higher attention weight.

In Figure 4 and Figure 5, we visualize the attention weights over n-grams to justify the effectiveness of our ACNN model. In Figure 4, we choose a tweet posted by user1 with content “John Wall. Speed and strength”. The user1 is a fan of “John Wall”, and the user2 has never talked about him. For the word “John Wall”[4], user1 has the largest attention weight, but user2 almost pays no attention, demonstrating that our ACNN model can capture crucial components for each user.

In Figure 5, we visualize the weights of a Chinese post posted by user1 with content “奇才 后卫 约翰沃尔 在 美国 国家队 表演赛 一显身手” (“The Wizards’s guard John Wall displayed his skills in the exhibition match of US national team”). As shown in this post, user1’s pursuit of John Wall is unchanged in Chinese, indicating that analyzing texts jointly could help extract more precise individuality of a user. For unigram, user1 does not have the largest attention weight for word “ 约翰沃尔” (“John Wall”). Nonetheless, for bi-grams and tri-grams, user1 has the largest attention weights on “后卫 约翰沃尔” (“guard John Wall”) and “奇才 后卫 约翰沃尔” (“Wizards guard John Wall”), respectively, and the reason is that n-grams contain more abundant and accurate semantic information that is crucial for each user to compose appropriate post representation.

5.5 Effect of Adversarial Mechanism

Table 6 shows the effect of adversarial mechanism. Chen et al. (2016b) has proved the important role of bilingual word embeddings in cross-lingual classification with adversarial learning. We train our bilingual word embeddings with MUSE (Conneau et al., 2017) so that semantically similar words in different languages are drawn closely in the embedding space. When the model is trying to classify Twitter posts, we transform Chi-

---

[4] The “John Wall” is the name of a basketball star, and we treat it as a single word.

Models	Twitter			Weibo		
	P	R	F	P	R	F
CNN	75.60	77.02	76.41	73.57	73.60	73.55
ACNN	81.35	80.43	80.85	80.31	79.85	80.02
CNN-MT	76.62	76.84	76.69	73.95	73.75	73.72
ACNN-MT	80.91	80.95	**80.89**	81.20	81.63	**81.53**

Table 5: Effect of attention mechanism and multi-task learning.

Models	Twitter			Weibo		
	P	R	F	P	R	F
CNN-MT	76.62	76.84	76.69	73.95	73.75	73.72
CNN-AMT	77.14	77.74	77.41	74.18	74.47	74.25
ACNN-MT	80.91	80.95	80.89	81.20	81.63	81.53
ACNN-AMT	81.90	80.73	**81.24**	82.43	81.74	**82.03**

Table 6: Effect of adversarial mechanism.

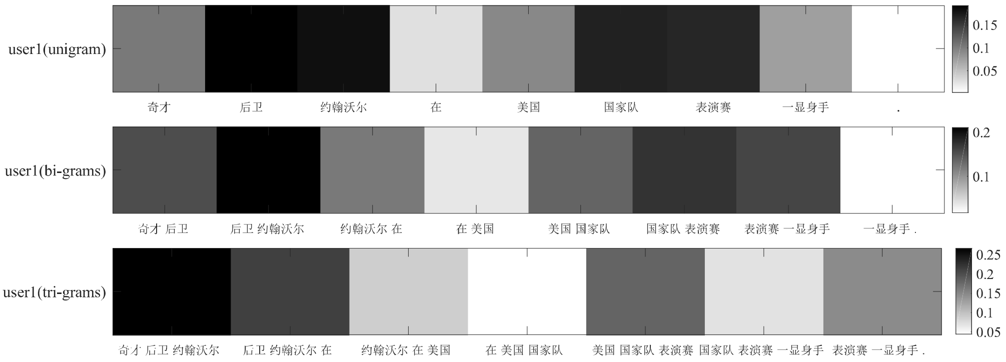

Figure 5: Visualization of attentions over n-grams of Weibo. (奇才 : *Wizards*; 后卫 : *rearguard*; 约翰沃尔: *John Wall*; 在: *in*; 美国: *US*; 国家队: *national team*; 表演赛: *exhibition game*; 一显身手: *display his skills.*)

nese word embeddings into English word embeddings for alignment, and vice versa. By comparing models with adversarial mechanism (i.e., CNN-AMTL, ACNN-AMTL) and models without adversarial mechanism (i.e., CNN-MTL, ACNN-MTL), the former yield better results, suggesting that the adversarial learning could help generate language-independent sentiment features, and improve the effect of sentiment classification.

6 Conclusion

In this paper, we proposed an ACNN model for capturing user's individuality on microblogs, and an ACNN-AMT model for alleviating the data insufficiency problem in personalized sentiment classification. By exploiting the user information, our ACNN model could put emphasis on the key portions of posts that reflect users' individuality. By utilizing the consistent individuality and opinion bias of microblog users across different languages, our ACNN-AMT model could extract the language-specific features and language-independent features to enrich the user-specific post representation. Experimental results on English and Chinese microblog datasets confirm the clear advantage of our method over state-of-the-art baseline algorithms.

Acknowledgments

The work was supported by the National Key R&D Program of China under grant 2018YFB1004700, and National Natural Science Foundation of China (61772122, 61872074).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016a. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016b. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Vibhu Jawa and Varun Hasija. 2015. A sentiment and interest based approach for product recommendation. In *Modelling and Simulation (UKSim), 2015 17th UKSim-AMSS International Conference on*, pages 75–80. IEEE.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Kaisong Song, Ling Chen, Wei Gao, Shi Feng, Daling Wang, and Chengqi Zhang. 2016. Persentiment: A personalized sentiment classification system for microblog users. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 255–258. ACM.

Kaisong Song, Shi Feng, Wei Gao, Daling Wang, Ling Chen, and Chengqi Zhang. 2015a. Build emotion lexicon from microblogs by combining effects of seed words and emoticons in a heterogeneous graph. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, pages 283–292. ACM.

Kaisong Song, Shi Feng, Wei Gao, Daling Wang, Ge Yu, and Kam-Fai Wong. 2015b. Personalized sentiment classification based on latent individuality of microblog users. In *IJCAI*, pages 2277–2283.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Xiao Sun, Fei Gao, Chengcheng Li, and Fuji Ren. 2015. Chinese microblog sentiment classification based on convolution neural network with content extension method. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pages 408–414. IEEE.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1014–1023.

Fangzhao Wu and Yongfeng Huang. 2016. Personalized microblog sentiment classification via multitask learning. In *AAAI*, pages 3059–3065.

Fangzhao Wu, Yangqiu Song, and Yongfeng Huang. 2015. Microblog sentiment classification with contextual knowledge regularization. In *AAAI*, pages 2332–2338.

Zhiwei Yu, Raymond K Wong, Chi-Hung Chi, and Fang Chen. 2015. A semi-supervised learning approach for microblog sentiment classification. In *Smart City/SocialCom/SustainCom (SmartCity), 2015 IEEE International Conference on*, pages 339–344. IEEE.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhou Zhao, Hanqing Lu, Deng Cai, Xiaofei He, and Yueting Zhuang. 2017. Microblog sentiment classification via recurrent random walk network learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3532–3538. AAAI Press.

Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks

Zhichun Wang, Qingsong Lv, Xiaohan Lan, Yu Zhang
College of Information Science and Technique
Beijing Normal University, Beijing, 100875, China
`zcwang@bnu.edu.cn` `{lqs,lanxh,zybnu}@mail.bnu.edu.cn`

Abstract

Multilingual knowledge graphs (KGs) such as DBpedia and YAGO contain structured knowledge of entities in several distinct languages, and they are useful resources for cross-lingual AI and NLP applications. Cross-lingual KG alignment is the task of matching entities with their counterparts in different languages, which is an important way to enrich the cross-lingual links in multilingual KGs. In this paper, we propose a novel approach for cross-lingual KG alignment via graph convolutional networks (GCNs). Given a set of pre-aligned entities, our approach trains GCNs to embed entities of each language into a unified vector space. Entity alignments are discovered based on the distances between entities in the embedding space. Embeddings can be learned from both the structural and attribute information of entities, and the results of structure embedding and attribute embedding are combined to get accurate alignments. In the experiments on aligning real multilingual KGs, our approach gets the best performance compared with other embedding-based KG alignment approaches.

1 Introduction

Knowledge graphs (KGs) represent human knowledge in the machine-readable format, are becoming the important basis of many applications in the areas of artificial intelligence and natural language processing. Multilingual KGs such as DBpedia (Bizer et al., 2009), YAGO (Suchanek et al., 2008; Rebele et al., 2016), and BabelNet (Navigli and Ponzetto, 2012) are especially valuable if cross-lingual applications are to be built. Besides the knowledge encoded in each distinct language, multilingual KGs also contain rich cross-lingual links that match the equivalent entities in different languages. The cross-lingual links play an important role to bridge the language gap in a multilingual KG; however, not all the equivalent entities are connected by cross-lingual links in most multilingual KGs. Therefore, increasingly more research work studies the problem of cross-lingual KG alignment, aiming to match entities in different languages in a multilingual KG automatically.

Traditional cross-lingual KG alignment approaches either rely on machine translation technique or defining various language-independent features to discover cross-lingual links. Most recently, several embedding-based approaches have been proposed for cross-lingual KG alignment, including MTransE (Chen et al., 2017) and JAPE (Sun et al., 2017). Given two KGs and a set of pre-aligned entities of them, embedding-based approaches project entities into low-dimensional vector spaces; entities are matched based on the computations on their vector representations. Following very similar ideas as above, JE (Hao et al., 2016) and ITransE (Zhu et al., 2017) are embedding-based approaches for matching entities between heterogeneous KGs, and they can also work for the problem of cross-lingual KG alignment. The above embedding-based approaches can achieve promising performance without machine translation or feature engineering.

However, we find that the above approaches all try to jointly model the cross-lingual knowledge and the monolingual knowledge in one unified optimization problem. The loss of two kinds of knowledge has to be carefully balanced during the optimization. For example, JE, MTransE, and ITransE all use hyper-parameters to weight the loss of entity alignments in the loss functions of their models; JAPE uses the pre-aligned entities to combine two KGs as one, and adds weight to the scores of negative samples in its loss function. In the above approaches, entities' embeddings have to encode both the structural information in KGs and the equivalent relations of entities. Further-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 349–357
Brussels, Belgium, October 31 - November 4, 2018.

more, the attributes of entities (e.g., the age of a people, the population of a country) have not been fully utilized in the existing models. MTransE and ITransE cannot use attributional information in KGs; although JAPE includes the attribute types in the model, the attribute values of entities are ignored. We believe that considering the attribute values can further improve the results of KG alignment.

Having the above observations, we propose a new embedding-based KG alignment approach which directly models the equivalent relations between entities by using graph convolutional networks (GCNs). GCN is a kind of convolutional network which directly operates on graph-structured data; it generates node-level embeddings by encoding information about the nodes' neighborhoods. The adjacencies of two equivalent entities in KGs usually contain other equivalent entities, so we choose GCNs to generate neighborhood-aware embeddings of entities, which are used to discover entity alignments. Our approach can also provide a simple and effective way to include entities' attribute values in the alignment model. More specifically, our approach has the following advantages:

- Our approach uses the entity relations in each KG to build the network structure of GCNs, and it only considers the equivalent relations between entities in model training. Our approach has small model complexity and can achieve encouraging alignment results.
- Our approach only needs pre-aligned entities as training data, and it does not require any pre-aligned relations or attributes between KGs.
- Entity relations and entity attributes are effectively combined in our approach to improve the alignment results.

In the experiments on aligning real multilingual KGs, our approach gets the best performance compared with the baseline methods.

The rest of this paper is organized as follows, Section 2 reviews some related work, Section 3 introduces some background knowledge, Section 4 describes our proposed approach, Section 5 presents the evaluation results, Section 6 is the conclusion and future work.

2 Related Work

2.1 KG Embedding

In the past few years, much work has been done on the problem of KG embedding. KG embedding models embed entities and relations in a KG into a low-dimensional vector space while preserving the original knowledge. The embeddings are usually learned by minimizing a global loss function of all the entities and relations in a KG, which can be further used for relation prediction, information extraction, and some other tasks. TransE is a representative KG embedding approach (Bordes et al., 2013), which projects both entities and relations into the same vector space; if a triple (h, r, t) holds, TransE wants that $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$. The embeddings are learned by minimizing a margin-based ranking criterion over the training set. TransE model is simple but powerful, and it gets promising results on link prediction and triple classification problems. To further improve TransE, several enhanced models based on it have been proposed, including TransR (Lin et al., 2015), TransH (Wang et al., 2014) and TransD (Ji et al., 2015) etc. By introducing new representations of relational translation, later approaches achieve better performance at the cost of increasing model complexity. There are many other KG embedding approaches, recent surveys (Wang et al., 2017; Nickel et al., 2016) give detailed introduction and comparison.

2.2 Embedding-based KG Alignment

Here we introduce the KG Alignment approaches most related to ours, and discuss the main differences between our approach and them.

JE (Hao et al., 2016) jointly learns the embeddings of multiple KGs in a uniform vector space to align entities in KGs. JE uses a set of seed entity alignments to connect two KGs, and then learns the embeddings by using a modified TransE model, which adds a loss of entity alignments in its global loss function.

MTransE (Chen et al., 2017) encodes entities and relations of each KG in a separated embedding space by using TransE; it also provides transitions for each embedding vector to its cross-lingual counterparts in other spaces. The loss function of MTransE is the weighted sum of two component models' loss (i.e., knowledge model and alignment model). To train the alignment model, MTransE needs a set of aligned triples of two KGs.

JAPE (Sun et al., 2017) combines structure embedding and attribute embedding to match entities in different KGs. Structure embedding follows the TransE model, which learns vector representations of entities in the overlay graph of two KGs. Attribute embedding follows the Skip-gram model, which aims to capture the correlations of attributes. To get desirable results, JAPE needs the relations and attributes of two KGs to be aligned in advance.

ITransE (Zhu et al., 2017) is a joint knowledge embedding approach for multiple KGs, which is also suitable for the cross-lingual KG alignment problem. ITransE first learns both entity and relation embeddings following TransE; then it learns to map knowledge embeddings of different KGs into a joint space according to a set of seed entity alignments. ITransE performs iterative entity alignment by using the newly discovered entity alignments to update joint embeddings of entities. ITransE requires all relations being shared among KGs.

The above approaches follow the similar framework to match entities in different KGs. They all rely on TransE model to learn entity embeddings, and then define some kinds of transformation between embeddings of aligned entities. Compared with these approaches, our approach uses an entirely different framework; it uses GCNs to embed entities in a unified vector space, where aligned entities are expected to be as close as possible. Our approach only focuses on matching entities in two KGs, and it does not learn embeddings of relations. MTransE, JAPE, and ITransE all require relations being aligned or shared in KGs; our approach does not need this kind of prior knowledge.

3 Problem Formulation

KGs represent knowledge about real-world entities as triples. Here we consider two kinds of triples in KGs: relational triples, and attributional triples. Relational triples represents relations between entities, and it has the form $\langle entity_1, relation, entity_2 \rangle$. Attributional triples describe attributes of entities, and it has the form $\langle entity, attribute, value \rangle$. For example in the data of YAGO, *graduatedFrom* is a relation, and (*Albert_Einstein, graduatedFrom, ETH_Zurich*) is a relational triple; *diedOnDate* is an attribute, and (*Albert_Einstein, diedOnDate, 1955*) is an attributional triple. Both relational and attributional triples describe important information about entities, we will take both of them into account in the task of cross-lingual KG alignment.

Formally, we represent a KG as $G = (E, R, A, T_R, T_A)$, where E, R, A are sets of entities, relations and attributes, respectively; $T_R \subset E \times R \times E$ is the set of relational triples, $T_A \subset E \times A \times V$ is the set of attributional triples, where V is the set of attribute values.

Let $G_1 = (E_1, R_1, A_1, T_1^R, T_1^A)$ and $G_2 = (E_2, R_2, A_2, T_2^R, T_2^A)$ be two KGs in different languages, and $S = \{(e_{i_1}, e_{i_2}) | e_{i_1} \in E_1, e_{i_2} \in E_2\}_{i=1}^m$ be a set of pre-aligned entity pairs between G_1 and G_2. We define the task of cross-lingual KG alignment as finding new entity alignments based on the existing ones. In multilingual KGs such as DBpedia and YAGO, the cross-lingual links in them can be used to build the sets of pre-aligned entity pairs. The already known entity alignments are used as seeds or training data in the process of KG alignment.

4 The Proposed Approach

The framework of our proposed approach is shown in Figure 1. Given two KGs G_1 and G_2 in different languages, and a set of known aligned entity pairs $S = \{(e_{i_1}, e_{i_2})\}_{i=1}^m$ between them, our approach automatically find new entity alignments based on GCN-based entity embeddings. The basic idea of our approach is to use GCNs to embed entities from different languages into a unified vector space, where equivalent entities are expected to be as close as possible. Entity alignments are predicted by applying a pre-defined distance function to entities' GCN-representations.

4.1 GCN-based Entity Embedding

GCNs (Bruna et al., 2014; Henaff et al., 2015; Defferrard et al., 2016; Kipf and Welling, 2017) are a type of neural network that directly operates on graph data. GCNs allow end-to-end learning of prediction pipelines whose inputs are graphs of arbitrary size and shape. The inputs of a GCN are feature vectors of nodes and the structure of the graph; the goal of a GCN is to learn a function of features on the input graph and produces a node-level output. GCNs can encode information about the neighborhood of a node as a real-valued vector, which was usually used for classification or regression. When solving the problem of KG alignment, we assume that (1) equivalent entities tend

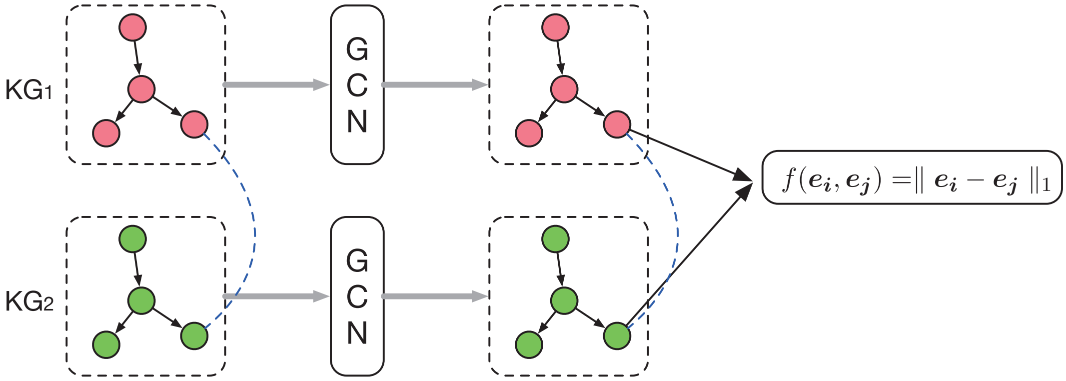

Figure 1: Framework of our approach (dashed blue lines connects equivalent entities in two KGs)

to have similar attributes, and (2) equivalent entities are usually neighbored by some other equivalent entities. GCNs can combine the attribute information and the structure information together, therefore our approach uses GCNs to project entities into low-dimensional vector space, where equivalent entities are close to each other.

A GCN-model consists of multiple stacked GCN layers. The input to the l-th layer of the GCN model is a vertex feature matrix, $H^{(l)} \in \mathbb{R}^{n \times d^{(l)}}$, where n is the number of vertices and $d^{(l)}$ is the number of features in the l-th layer. The output of the l-th layer is a new feature matrix $H^{(l+1)}$ by the following convolutional computation:

$$H^{(l+1)} = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \qquad (1)$$

where σ is an activation function; A is a $n \times n$ connectivity matrix that represents the structure information of the graph; $\hat{A} = A + I$, and I is the identity matrix; $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$; $W^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l+1)}}$ is the weight matrix of the l-th layer in the GCN, $d^{(l+1)}$ is the dimensionality of new vertex features.

Structure and Attribute Embedding. In our approach, GCNs are used to embed entities of two KGs in a unified vector space. To utilize both structure and attribute information of entities, our approach assigns two feature vectors to each entity in GCN layers, structure feature vector h_s and attribute feature vector h_a. In the input layer, $h_s^{(0)}$ is randomly initialized and updated during the training process; $h_a^{(0)}$ is the attribute vectors of entities and it is fixed during the model training. Let H_s and H_a be the structure and attribute feature matrices of all the entities, we redefine the convolutional computation as:

$$\begin{aligned} &[H_s^{(l+1)}; H_a^{(l+1)}] \\ &= \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}[H_s^{(l)}W_s^{(l)}; H_a^{(l)}W_a^{(l)}]\right) \end{aligned} \qquad (2)$$

where $W_s^{(l)}$ and $W_a^{(l)}$ are the weight matrices for structure features and attribute features in the l-th layer, respectively; [;] denotes the concatenation of two matrices. The activation function σ is chosen as $ReLU(\cdot) = max(0, \cdot)$.

Model Configuration. More specifically, our approach uses two 2-layer GCNs, and each GCN processes one KG to generate embeddings of its entities. As defined in Section 3, we denote two KGs as $G_1 = (E_1, R_1, A_1, T_1^R, T_1^A)$ and $G_2 = (E_2, R_2, A_2, T_2^R, T_2^A)$; and let their corresponding GCN models be denoted as GCN_1 and GCN_2. As for the structure feature vectors of entities, we set the dimensionality of feature vectors to d_s in all the layers of GCN_1 and GCN_2; and two GCN models share the weight matrices $W_s^{(1)}$ and $W_s^{(2)}$ for the structure features in two layers. As for the attribute vectors of entities, we set the dimensionality of output feature vectors to d_a. Because two KGs may have different number of attributes (i.e. $|A_1| \neq |A_2|$), the dimensionalities of the input attribute feature vectors in two GCN models are different. The first layer of each GCN model transforms the input attribute feature vectors into vectors of size d_a; and two GCN-models generate attribute embeddings of the same dimensionality. Table 1 outlines the parameters of two GCNs in our approach. The final outputs of two GCNs are $(d_s + d_a)$-dimensional embeddings of entities, which are further used to discover entity alignments.

Parameter	GCN_1	GCN_2
Initial Structure Feature Matrices	$H_{s1}^{(0)} \in \mathbb{R}^{\lvert E_1\rvert \times d_s}$	$H_{s2}^{(0)} \in \mathbb{R}^{\lvert E_2\rvert \times d_s}$
Weight Matrix for Structure Features in Layer 1	$W_s^{(1)} \in \mathbb{R}^{d_s \times d_s}$	
Weight Matrix for Structure Features in Layer 2	$W_s^{(2)} \in \mathbb{R}^{d_s \times d_s}$	
Output Structure Embeddings	$H_{s1}^{(2)} \in \mathbb{R}^{\lvert E_1\rvert \times d_s}$	$H_{s2}^{(2)} \in \mathbb{R}^{\lvert E_2\rvert \times d_s}$
Initial Attribute Feature Matrices	$H_{a1}^{(0)} \in \mathbb{R}^{\lvert E_1\rvert \times \lvert A_1\rvert}$	$H_{a2}^{(0)} \in \mathbb{R}^{\lvert E_2\rvert \times \lvert A_2\rvert}$
Weight Matrix for Attribute Features in Layer 1	$W_{a1}^{(1)} \in \mathbb{R}^{\lvert A_1\rvert \times d_a}$	$W_{a2}^{(1)} \in \mathbb{R}^{\lvert A_2\rvert \times d_a}$
Weight Matrix for Attribute Features in Layer 2	$W_a^{(2)} \in \mathbb{R}^{d_a \times d_a}$	
Output Attribute Embeddings	$H_{a1}^{(2)} \in \mathbb{R}^{\lvert E_1\rvert \times d_a}$	$H_{a2}^{(2)} \in \mathbb{R}^{\lvert E_2\rvert \times d_a}$

Table 1: Parameters of two GCNs

Computation of Connectivity Matrix. In a GCN model, the connectivity matrix A defines the neighborhoods of entities in the convolutional computation. For an undirected graph, the adjacency matrix can be directly used as A_s. But KGs are relational multi-graphs, entities are connected by typed relations. Therefore, we design a particular method for computing A of a KG; we let $a_{ij} \in A$ indicate to what extent the information of alignments propagates from the i-th entity to the j-th entity. The probability of two entities being equivalent differs greatly considering they connect to aligned entities by different relations (e.g., *hasParent* vs. *hasFriend*). Therefore, we compute two measures, which are called functionality and inverse functionality, for each relation:

$$fun(r) = \frac{\#Head\_Entities\_of\_r}{\#Triples\_of\_r} \tag{3}$$

$$ifun(r) = \frac{\#Tail\_Entities\_of\_r}{\#Triples\_of\_r} \tag{4}$$

where $\#Triples\_of\_r$ is the number of triples of relation r; $\#Head\_Entities\_of\_r$ and $\#Tail\_Entities\_of\_r$ are the numbers of head entities and tail entities of r, respectively. To measure the influence of the i-th entity over the j-the entity, we set $a_{ij} \in A$ as:

$$a_{ij} = \sum_{\langle e_i,r,e_j\rangle \in G} ifun(r) + \sum_{\langle e_j,r,e_i\rangle \in G} fun(r) \tag{5}$$

4.2 Alignment prediction

Entity alignments are predicted based on the distances between entities from two KGs in the GCN-representation space. For entities e_i in G_1 and v_j in G_2, we compute the following distance measure between them:

$$\begin{aligned} D(e_i, v_j) = &\beta \frac{f(\boldsymbol{h_s}(e_i), \boldsymbol{h_s}(v_j)}{d_s} \\ &+ (1-\beta)\frac{f(\boldsymbol{h_a}(e_i), \boldsymbol{h_a}(v_j))}{d_a} \end{aligned} \tag{6}$$

where $f(\boldsymbol{x}, \boldsymbol{y}) = \| \boldsymbol{x} - \boldsymbol{y} \|_1$, $\boldsymbol{h_s}(\cdot)$ and $\boldsymbol{h_a}(\cdot)$ denote the structure embedding and attribute embedding of an entity, respectively; d_s and d_a are dimensionalities of structure embeddings and attribute embeddings; β is a hyper-parameter that balances the importance of two kinds of embeddings.

The distance is expected to be small for equivalent entities and large for non-equivalent ones. For a specific entity e_i in G_1, our approach computes the distances between e_i and all the entities in G_2, and returns a list of ranked entities as candidate alignments. The alignment can be also performed from G_2 to G_1. In the experiments, we report the results of both directions of KG alignment.

4.3 Model Training

To enable GCNs to embed equivalent entities as close as possible in the vector space, we use a set of known entity alignments S as training data to train GCN models. The model training is performed by minimizing the following margin-based ranking loss functions:

$$\begin{aligned} L_s = \sum_{(e,v)\in S} \sum_{(e',v')\in S'_{(e,v)}} &[f(\boldsymbol{h_s}(e), \boldsymbol{h_s}(v)) + \gamma_s \\ &- f(\boldsymbol{h_s}(e'), \boldsymbol{h_s}(v')]_+ \end{aligned} \tag{7}$$

$$
\begin{aligned}
L_a = \sum_{(e,v)\in S} \sum_{(e',v')\in S'_{(e,v)}} & [f(\boldsymbol{h_a}(e), \boldsymbol{h_a}(v)) + \gamma_a \\
& - f(\boldsymbol{h_a}(e'), \boldsymbol{h_a}(v'))]_+
\end{aligned}
\tag{8}
$$

where $[x]_+ = \max\{0, x\}$, $S'_{(e,v)}$ denotes the set of negative entity alignments constructed by corrupting (e, v), i.e. replacing e or v with a randomly chosen entity in G_1 or G_2; $\gamma_s, \gamma_a > 0$ are margin hyper-parameters separating positive and negative entity alignments. L_s and L_a are loss functions for structure embedding and attribute embedding, respectively; they are independent of each other and hence are optimized separately. We adopt stochastic gradient descent (SGD) to minimize the above loss functions.

5 Experiment

5.1 Datasets

We use the DBP15K datasets in the experiments, which were built by Sun et al. (2017). The datasets were generated from DBpedia, a large-scale multilingual KG containing rich inter-language links between different language versions. Subsets of Chinese, English, Japanese and French versions of DBpedia are selected following certain rules. Table 2 outlines the detail information of the datasets. Each dataset contains data two KGs in different languages and 15 thousand inter-language links connecting equivalent entities in two KGs. In the experiments, the known equivalent entity pairs are used for model training and testing.

5.2 Experiment Settings

In the experiments, we compared our approach with JE, MTransE and JAPE. We also build JAPE′, a variant of JAPE which does not use pre-aligned relations and attributes. Because the approach ITransE performs iterative alignment and it requires two KGs sharing the same relations, we do not include it in the comparison. The inter-language links in each dataset are used as the gold standards of entity alignments. For all the compared approaches, we use 30% of inter-language links for training and 70% of them for testing; the split of training and testing are the same for all approaches. We use $Hits@k$ as the evaluation measure to assess the performance of all the approaches. $Hits@k$ measures the proportion of correctly aligned entities ranked in the top k candidates. For the parameters of our approach, we set $d_s = 1,000$, $d_a = 100$; the margin $\gamma_s = \gamma_a = 3$ in the loss function, and β in the distance measure is empercally set to 0.9.

5.3 Results

Table 3 shows the results of all the compared approaches on DBP15K datasets. We report $Hits@1$, $Hits@10$ and $Hits@50$ of approaches on each dataset. Because we use the same datasets as in (Sun et al., 2017), the results of JE, MTransE, and JAPE are obtained from (Sun et al., 2017). For JAPE and JAPE′, each of them has three variants: Structure Embedding without negative triples (SE w/o neg.), Structure Embedding (SE), Structure and attribute joint embedding (SE+AE). We use GCN(SE) and GCN(SE+AE) to denote two variants of our approach: one only uses relational triples to perform structure embedding, and the other uses both relational and attributional triples to perform structure and attribute embedding.

GCN(SE) vs. GCN(SE+AE)

We first compare the results of GCN(SE) and GCN(SE+AE) to see whether the attributional information is helpful in the KG alignment task. According to the results, adding attributes in our approach do lead to slightly better results. The improvements range from 1% to 10%, which are very similar to the improvements of JAPE(SE) over JAPE(SE+AE). It shows that the KG alignment mainly relays on the structural information in KGs, but the attributional information is still useful. Our approach uses the same framework for embedding structure and attribute information, the combination of two kinds of embeddings works effectively.

GCN(SE+AE) vs. Baselines

On the dataset of DBP15K$_{ZH-EN}$, JAPE(SE+AE) performs best and gets five best $Hits@k$ values; our approach GCN(SE+AE) gets the best $Hits@1$ in the alignment direction of ZH→EN. The results of GCN(SE+AE) and JAPE gets very close results regarding $Hits@1$ and $Hits@10$ in the direction of ZH→EN. In the alignment direction of EN→ZH, JAPE(SE+AE) outperforms GCN(SE+AE) by about 2-3%. But it should be noticed that JAPE uses additional aligned relations and attributes as its inputs,

Datasets		Entities	Relations	Attributes	Rel. triples	Attr. triples
$DBP15K_{ZH-EN}$	Chinese	66,469	2,830	8,113	153,929	379,684
	English	98,125	2,317	7,173	237,674	567,755
$DBP15K_{JA-EN}$	Japanese	65,744	2,043	5,882	164,373	354,619
	English	95,680	2,096	6,066	233,319	497,230
$DBP15K_{FR-EN}$	French	66,858	1,379	4,547	192,191	528,665
	English	105,889	2,209	6,422	278,590	576,543

Table 2: Details of the datasets

$DBP15K_{ZH-EN}$		$ZH \rightarrow EN$			$EN \rightarrow ZH$		
		Hits@1	Hits@10	Hits@50	Hits@1	Hits@10	Hits@50
*JE		21.27	42.77	56.74	19.52	39.36	53.25
*MTransE		30.83	61.41	79.12	24.78	52.42	70.45
*JAPE	SE w/o neg.	38.34	68.86	84.07	31.66	59.37	76.33
	SE	39.78	72.35	87.12	32.29	62.79	80.55
	SE + AE	41.18	**74.46**	**88.90**	**40.15**	**71.05**	**86.18**
JAPE′	SE w/o neg.	30.10	62.58	80.28	23.04	52.91	72.17
	SE	30.54	66.41	83.94	23.91	57.02	77.31
	SE + AE	33.32	69.28	86.40	33.02	66.92	85.15
GCN	SE	38.42	70.34	81.24	34.43	65.68	77.03
	SE + AE	**41.25**	74.38	86.23	36.49	69.94	82.45
$DBP15K_{JA-EN}$		$JA \rightarrow EN$			$EN \rightarrow JA$		
		Hits@1	Hits@10	Hits@50	Hits@1	Hits@10	Hits@50
*JE		18.92	39.97	54.24	17.80	38.44	52.48
*MTransE		27.86	57.45	75.94	23.72	49.92	67.93
*JAPE	SE w/o neg.	33.10	63.90	80.80	29.71	56.28	73.84
	SE	34.27	66.39	83.61	31.40	60.80	78.51
	SE + AE	36.25	68.50	85.35	38.37	67.27	82.65
JAPE′	SE w/o neg.	28.90	60.61	80.03	25.34	53.36	71.94
	SE	29.35	63.31	82.76	26.37	57.35	76.87
	SE + AE	31.06	64.11	81.57	32.45	62.21	79.08
GCN	SE	38.21	72.49	82.69	36.90	68.50	79.51
	SE + AE	**39.91**	**74.46**	**86.10**	**38.42**	**71.81**	**83.72**
$DBP15K_{FR-EN}$		$FR \rightarrow EN$			$EN \rightarrow FR$		
		Hits@1	Hits@10	Hits@50	Hits@1	Hits@10	Hits@50
*JE		15.38	38.84	56.50	14.61	37.25	54.01
*MTransE		24.41	55.55	74.41	21.26	50.60	69.93
*JAPE	SE w/o neg.	29.55	62.18	79.36	25.40	56.55	74.96
	SE	29.63	64.55	81.90	26.55	60.30	78.71
	SE + AE	32.39	66.68	83.19	32.97	65.91	82.38
JAPE′	SE w/o neg.	28.23	60.99	78.47	24.68	55.25	74.19
	SE	27.58	62.03	79.98	24.93	58.95	77.79
	SE + AE	30.21	65.81	82.57	31.42	63.86	80.95
GCN	SE	36.51	73.42	85.93	36.08	72.37	85.44
	SE + AE	**37.29**	**74.49**	**86.73**	**36.77**	**73.06**	**86.39**

Table 3: Results comparison of cross-lingual KG alignment (* marks the results obtained from (Sun et al., 2017))

while our approach does not use these kinds of prior knowledge. If compared with JAPE′, GCN(SE+AE) performs better than it regrading $Hits@1$ and $Hits@10$. While compared with

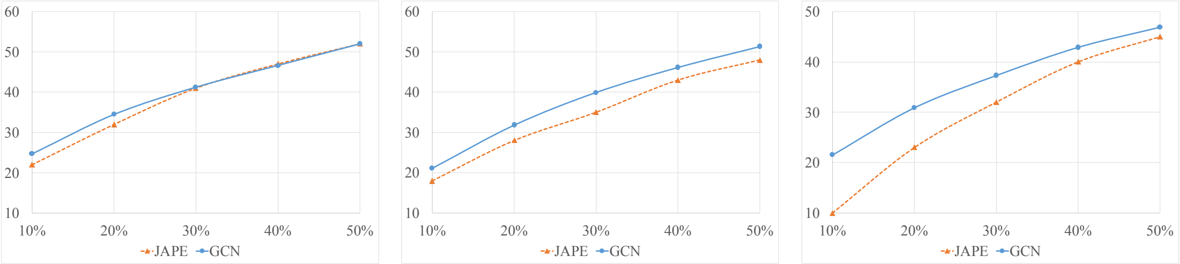

(a) ZH-EN (b) JA-EN (c) FR-EN

Figure 2: GCN and JAPE using different sizes of training data (horizontal coordinates: proportions of pre-aligned entities used in training data; vertical coordinates: $Hits@1$)

JE and MTransE, GCN(SE+AE) outperform them by more than 10% in most cases. On the datasets of DBP15K$_{JA-EN}$ and DBP15K$_{FR-EN}$, GCN(SE+AE) outperforms all the compared approaches regarding all the $Hits@k$ measures. Even compared with JAPE which uses extra relation and attribute alignments, GCN(SE+AE) still gets better results than it.

Comparing with all the baselines, both GCN(SE) and GCN(SE+AE) outperform JE and MTransE significantly. Among all the baselines, JAPE is the strongest one; it might due to its ability of using both relational and attributional triples, and the extra alignments of relations and attributes that it consumes. Our approach achieves better results than JAPE on two datasets; Although JAPE performs better than our approach, the differences between their results are small. If there are no existing relation and attribute alignments between two KGs, our approach will have distinct advantage over JAPE.

GCN vs. JAPE using different sizes of training data

To investigate how the size of training set affects the results of our approach, we further compare our approach with JAPE by using different number of pre-aligned entities as training data. For JAPE, the pre-aligned entities are used as seeds to make their vectors overlapped. In our approach, all the pre-aligned entities are used to train GCN models. Intuitively, the more pre-aligned entities used, the better results should be obtained by both GCN and JAPE.

Here we use different proportions of pre-aligned entities as training data, which ranges 10% to 50% with step 10%; all the rest of pre-aligned entities are used for testing. Figure 2 shows the $Hits@1$ of two approaches in three datasets. It shows that both approaches perform better as the size of training data increases. And our approach always outperforms JAPE except using 40% pre-aligned entities as training data in Figure 2(a). Especially in the tasks of aligning Japanese to English and French to English, our approach has a distinct advantage over JAPE.

6 Conclusion and Future Work

This paper presents a new embedding-based KG alignment approach which discovers entity alignments based on the entity embeddings learned by GCNs. Our approach can make use of both the relational and the attributional triples in KGs to discover the entity alignments. We evaluate our method on the data of real multilingual KGs, and the results show the advantages of our approach over the compared baselines.

In the future work, we will explore more advanced GCN models for KG alignment task, such as Relational GCNs (Schlichtkrull et al., 2017) and Graph Attention Networks (GATs) (Velickovic et al., 2017). Furthermore, how to iteratively discover new entity alignments in the framework of our approach is another interesting direction that we will study in the future.

Acknowledgments

The work is supported by the National Natural Science Foundation of China (No. 61772079) and the National Key R&D Program of China (No. 2017YFC0804004).

References

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia-a crystallization point for the web of data. *Web Semantics:*

science, services and agents on the world wide web, 7(3):154–165.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of Advances in neural information processing systems (NIPS2013)*, pages 2787–2795.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *Proceedings of International Conference on Learning Representations (ICLR2014)*.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (AAAI2017)*, pages 1511–1517.

Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings Advances in Neural Information Processing Systems (NIPS2016)*, pages 3844–3852.

Yanchao Hao, Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint embedding method for entity alignment of knowledge bases. In *Proceedings of China Conference on Knowledge Graph and Semantic Computing (CCKS2016)*, pages 3–14.

Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 687–696.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference on Learning Representations (ICLR2017)*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI2015)*, volume 15, pages 2181–2187.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Thomas Rebele, Fabian M. Suchanek, Johannes Hoffart, Joanna Asia Biega, Erdal Kuzey, and Gerhard Weikum. 2016. Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *Proceedings of the Fifteenth International Semantic Web Conference (ISWC2016)*, pages 177–185.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *Proceedings of the Sixteenth International Semantic Web Conference (ISWC2017)*, pages 628–644.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *CoRR*, abs/1710.10903.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-eighth AAAI Conference on Artificial Intelligence (AAAI2014)*, volume 14, pages 1112–1119.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via joint knowledge embeddings. In *Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence (IJCAI2017)*, pages 4258–4264.

Cross-lingual Lexical Sememe Prediction

Fanchao Qi[1*], **Yankai Lin**[1*], **Maosong Sun**[1,2†], **Hao Zhu**[1], **Ruobing Xie**[3], **Zhiyuan Liu**[1]
[1]Department of Computer Science and Technology, Tsinghua University
Institute for Artificial Intelligence, Tsinghua University
State Key Lab on Intelligent Technology and Systems, Tsinghua University
[2]Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University
[3]Search Product Center, WeChat Search Application Department, Tencent, China
{qfc17, linyk14}@mails.tsinghua.edu.cn, sms@tsinghua.edu.cn
zhuhao15@mails.tsinghua.edu.cn
xrbsnowing@163.com, liuzy@tsinghua.edu.cn

Abstract

Sememes are defined as the minimum semantic units of human languages. As important knowledge sources, sememe-based linguistic knowledge bases have been widely used in many NLP tasks. However, most languages still do not have sememe-based linguistic knowledge bases. Thus we present a task of cross-lingual lexical sememe prediction, aiming to automatically predict sememes for words in other languages. We propose a novel framework to model correlations between sememes and multi-lingual words in low-dimensional semantic space for sememe prediction. Experimental results on real-world datasets show that our proposed model achieves consistent and significant improvements as compared to baseline methods in cross-lingual sememe prediction. The codes and data of this paper are available at `https://github.com/thunlp/CL-SP`.

1 Introduction

Words are regarded as the smallest meaningful unit of speech or writing that can stand by themselves in human languages, but not the smallest indivisible semantic unit of meaning. That is, the meaning of a word can be represented as a set of semantic components. For example, "*Man = human + male + adult*" and "*Boy = human + male + child*". In linguistics, the minimum semantic unit of meaning is named sememe (Bloomfield, 1926). Some people believe that semantic meanings of concepts such as words can be composed of a limited closed set of sememes. And sememes can help us comprehend human languages better.

Unfortunately, the lexical sememes of words are not explicit in most human languages. Hence, people construct sememe-based linguistic knowledge bases (KBs) via manually annotating every words with a pre-defined closed set of sememes. HowNet (Dong and Dong, 2003) is one of the most well-known sememe-based linguistic KBs. Different from WordNet (Miller, 1995) which focuses on the relations between senses, it annotates each word with one or more relevant sememes. As illustrated in Fig. 1, the word *apple* has two senses including *apple (fruit)* and *apple (brand)* in HowNet. The sense *apple (fruit)* has one sememe *fruit*, and the sense *apple (brand)* has five sememes including *computer*, *PatternValue*, *able*, *bring* and *SpecificBrand*. There exist about 2,000 sememes and over 100 thousand labeled Chinese and English words in HowNet. HowNet has been widely used in various NLP applications such as word similarity computation (Liu and Li, 2002), word sense disambiguation (Zhang et al., 2005), question classification (Sun et al., 2007) and sentiment classification (Dang and Zhang, 2010).

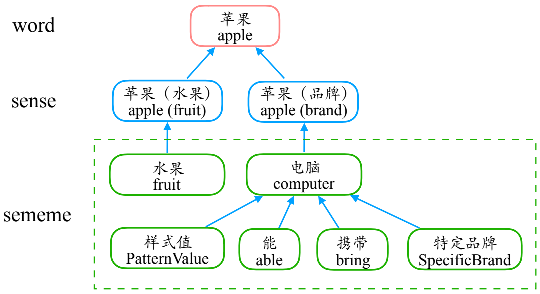

Figure 1: An example of HowNet.

However, most languages do not have such sememe-based linguistic KBs, which prevents us understanding and utilizing human languages to a greater extent. Therefore, it is important to build sememe-based linguistic KBs for various languages. Manual construction for sememe-based linguistic KBs requires efforts of many linguistic experts, which is time-consuming and

---

[*] Indicates equal contribution
[†] Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 358–368
Brussels, Belgium, October 31 - November 4, 2018.

labor-intensive. For example, the construction of HowNet has cost lots of Chinese linguistic experts more than 10 years.

To address the issue of the high labor cost of manual annotation, we propose a new task, cross-lingual lexical sememe prediction (CLSP) which aims to automatically predict lexical sememes for words in other languages. CLSP aims to assist in the annotation of linguistic experts. There are two critical challenges for CLSP: (1) There is not a consistent one-to-one match between words in different languages. For example, English word "beautiful" can refer to Chinese words of either "美丽" or "漂亮". Hence, we cannot simply translate HowNet into another language. And how to recognize the semantic meaning of a word in other languages becomes a critical problem. (2) Since there is a gap between the semantic meanings of words and sememes, we need to build semantic representations for words and sememes to capture the semantic relatedness between them.

To tackle these challenges, in this paper, we propose a novel model for CLSP, which aims to transfer sememe-based linguistic KBs from source language to target language. Our model contains three modules including (1) monolingual word embedding learning which is intended for learning semantic representations of words for source and target languages respectively; (2) cross-lingual word embedding alignment which aims to bridge the gap between the semantic representations of words in two languages; (3) sememe-based word embedding learning whose objective is to incorporate sememe information into word representations. For simplicity, we do not consider the hierarchy information in HowNet in this paper.

In experiments, we take Chinese as source language and English as target language to show the effectiveness of our model. Experimental results show that our proposed model could effectively predict lexical sememes for words with different frequencies in other languages. Our model also has consistent improvements on two auxiliary experiments including bilingual lexicon induction and monolingual word similarity computation by jointly learning the representations of sememes, words in source and target languages.

2 Related Work

Since HowNet was published (Dong and Dong, 2003), it has attracted wide attention of researchers. Most of related works focus on applying HowNet to specific NLP tasks (Liu and Li, 2002; Zhang et al., 2005; Sun et al., 2007; Dang and Zhang, 2010; Fu et al., 2013; Niu et al., 2017; Zeng et al., 2018; Gu et al., 2018). To the best of our knowledge, only Xie et al. (2017) and Jin et al. (2018) conduct studies of augmenting HowNet by recommending sememes for new words. However, both of the two works are aimed to recommend sememes for monolingual words and not applicable to cross-lingual circumstance. Accordingly, our work is the first effort to automatically perform cross-lingual sememe prediction to enrich sememe-based linguistic KBs.

Our novel model adopts the method of word representation learning (WRL). Recent years have witnessed great advances in WRL. Models like Skip-gram, CBOW (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014) are immensely popular and achieve remarkable performance in many NLP tasks. However, most WRL methods learn distributional information of words from large corpora while the valuable information contained in semantic lexicons are disregarded. Therefore, some works try to inject semantic information of KBs into WRL (Faruqui et al., 2015; Liu et al., 2015; Mrkšic et al., 2016; Bollegala et al., 2016). Nevertheless, these works are all applied to word-based KBs such as WordNet, few works pay attention to how to incorporate the knowledge from sememe-based linguistic KBs.

There also have been plenty of studies working on cross-lingual WRL (Upadhyay et al., 2016; Ruder, 2017). Most of them require parallel corpora (Zou et al., 2013; AP et al., 2014; Hermann and Blunsom, 2014; Kočiskỳ et al., 2014; Gouws et al., 2015; Luong et al., 2015; Coulmance et al., 2015). Some of them adopt unsupervised or weakly supervised methods (Mikolov et al., 2013b; Vulić and Moens, 2015; Conneau et al., 2017; Artetxe et al., 2017). There are also some works using a seed lexicon as the cross-lingual signal (Dinu et al., 2014; Faruqui and Dyer, 2014; Lazaridou et al., 2015; Shi et al., 2015; Lu et al., 2015; Gouws et al., 2015; Wick et al., 2016; Ammar et al., 2016; Duong et al., 2016; Vulić and Korhonen, 2016).

In terms of our cross-lingual sememe prediction task, parallel data-based bilingual WRL methods are unsuitable because most language pairs have no large parallel corpora. Besides, unsupervised methods are not appropriate either as they are generally hard to learn high-quality bilingual word embeddings. Therefore, we choose the seed lexicon method in our model, and further introduce matching mechanism that is inspired by Zhang et al. (2017) to enhance its performance.

3 Methodology

In this section, we introduce our novel model for CLSP. Here we define the language with sememe annotations as source language and the language without sememe annotations as target language. The main idea of our model is to learn word embeddings of source and target languages jointly in a unified semantic space, and then predict sememes for words in target language according to the words with similar semantic meanings in source language.

Our method consists of three parts: monolingual word representation learning, cross-lingual word embedding alignment and sememe-based word representation learning. Hence, we define the objective function of our method corresponding to the three parts:

$$\mathcal{L} = \mathcal{L}_{mono} + \mathcal{L}_{cross} + \mathcal{L}_{sememe}. \quad (1)$$

Here, the monolingual term $\mathcal{L}_{mono}$ is designed for learning monolingual word embeddings from non-parallel corpora for source and target languages respectively. The cross-lingual term $\mathcal{L}_{cross}$ aims to align cross-lingual word embeddings in a unified semantic space. And $\mathcal{L}_{sememe}$ can draw sememe information into word representation learning and conduce to better word embeddings for sememe prediction. In the following subsections, we introduce the three parts in detail.

3.1 Monolingual Word Representation

Monolingual word representation is responsible for explaining regularities in monolingual corpora of source and target languages. Since the two corpora are non-parallel, $\mathcal{L}_{mono}$ comprises two monolingual sub-models that are independent of each other:

$$\mathcal{L}_{mono} = \mathcal{L}^S_{mono} + \mathcal{L}^T_{mono}, \quad (2)$$

where the superscripts S and T denote source and target languages respectively.

As a common practice, we choose the well established Skip-gram model to obtain monolingual word embeddings. Skip-gram model is aimed at maximizing the predictive probability of context words conditioned on the centered word. Formally, taking the source side for example, given a training word sequence $\{w^S_1, \cdots, w^S_n\}$, Skip-gram model intends to minimize:

$$\mathcal{L}^S_{mono} = -\sum_{c=K+1}^{n-K} \sum_{-K \le k \le K, k \ne 0} \log P(w^S_{c+k} | w^S_c), \quad (3)$$

where K is the size of the sliding window. $P(w^S_{c+k}|w^S_c)$ stands for the predictive probability of one of the context words conditioned on the centered word w^S_c, formalized by the following softmax function:

$$P(w^S_{c+k}|w^S_c) = \frac{\exp(\mathbf{w}^S_{c+k} \cdot \mathbf{w}^S_c)}{\sum_{w^S_s \in V^S} \exp(\mathbf{w}^S_s \cdot \mathbf{w}^S_c)}, \quad (4)$$

in which V^s indicates the word vocabulary of source language. $\mathcal{L}^T_{mono}$ can be formulated similarly.

3.2 Cross-lingual Word Embedding Alignment

Cross-lingual word embedding alignment aims to build a unified semantic space for the words in source and target languages. Inspired by Zhang et al. (2017), we align the cross-lingual word embeddings with signals of a seed lexicon and self-matching.

Formally, $\mathcal{L}_{cross}$ is composed of two terms including alignment by seed lexicon $\mathcal{L}_{seed}$ and alignment by matching $\mathcal{L}_{match}$:

$$\mathcal{L}_{cross} = \lambda_s \mathcal{L}_{seed} + \lambda_m \mathcal{L}_{match}, \quad (5)$$

where λ_s and λ_m are hyperparameters for controlling relative weightings of the two terms.

Alignment by Seed Lexicon

The seed lexicon term $\mathcal{L}_{seed}$ encourages word embeddings of translation pairs in a seed lexicon $\mathcal{D}$ to be close, which can be achieved via a L_2 regularizer:

$$\mathcal{L}_{seed} = \sum_{\langle w^S_s, w^T_t \rangle \in \mathcal{D}} \|\mathbf{w}^S_s - \mathbf{w}^T_t\|^2, \quad (6)$$

in which w^S_s and w^T_t indicate the words in source and target languages in the seed lexicon respectively.

Alignment by Matching Mechanism

As for the matching process, it is founded on an assumption that each target word should be matched to a single source word or a special empty word, and vice versa. The goal of the matching process is to find the matched source (target) word for each target (source) word and maximize the matching probabilities for all the matched word pairs. The loss of this part can be formulated as:

$$\mathcal{L}_{match} = \mathcal{L}_{match}^{T2S} + \mathcal{L}_{match}^{S2T}, \tag{7}$$

where $\mathcal{L}_{match}^{T2S}$ is the term for target-to-source matching and $\mathcal{L}_{match}^{S2T}$ is the term for source-to-target matching.

Next, we give a detailed explanation of target-to-source matching, and the source-to-target matching is defined in the same way. We first introduce a latent variable $m_t \in \{0, 1, \cdots, |V^S|\}$ $(t = 1, 2, \cdots, |V^T|)$ for each target word w_t^T, where $|V^S|$ and $|V^T|$ indicate the vocabulary size of source and target languages respectively. Here, m_t specifies the index of the source word that w_t^T matches with, and $m_t = 0$ signifies the empty word is matched. Then we have $\mathbf{m} = \{m_1, m_2, \cdots, m_{|V^T|}\}$, and can formalize the target-to-source matching term:

$$\begin{aligned} \mathcal{L}_{match}^{T2S} &= -\log P(\mathcal{C}^T|\mathcal{C}^S) \\ &= -\log \sum_{\mathbf{m}} P(\mathcal{C}^T, \mathbf{m}|\mathcal{C}^S), \end{aligned} \tag{8}$$

where $\mathcal{C}^T$ and $\mathcal{C}^S$ denote the target and source corpus respectively. Here, we simply assume that the matching processes of target words are independent of each other. Therefore, we have:

$$\begin{aligned} P(\mathcal{C}^T, \mathbf{m}|\mathcal{C}^S) &= \prod_{w^T \in \mathcal{C}^T} P(w^T, \mathbf{m}|\mathcal{C}^S) \\ &= \prod_{t=1}^{|V^T|} P(w_t^T | w_{m_t}^S)^{c(w_t^T)}, \end{aligned} \tag{9}$$

where $w_{m_t}^S$ is the source word that w_t^T matches with, and $c(w_t^T)$ is the number of times w_t^T occurs in the target corpus.

3.3 Sememe-based Word Representation

Sememe-based word representation is intended for improving word embeddings for sememe prediction by introducing the information of sememe-based linguistic KBs of source language. In this section, we present two methods of sememe-based word representation.

Word Relation-based Approach

A simple and intuitive method is to let words with similar sememe annotations tend to have similar word embeddings, which we name word relation-based approach. To begin with, we construct a synonym list from sememe-based linguistic KBs of source language, where we regard words sharing a certain number of sememes as synonyms. Next, we force synonyms to have closer word embeddings.

Formally, we let $\mathbf{w}_i^S$ be original word embedding of w_i^S and $\hat{\mathbf{w}}_i^S$ be its adjusted word embedding. And let $\text{Syn}(w_i^S)$ denote the synonym set of word w_i^S. Then the loss function is:

$$\begin{aligned} \mathcal{L}_{sememe} = \sum_{w_i^S \in V^S} \Big[& \alpha_i \|\mathbf{w}_i^S - \hat{\mathbf{w}}_i^S\|^2 + \\ & \sum_{w_j^S \in \text{Syn}(w_i^S)} \beta_{ij} \|\hat{\mathbf{w}}_i^S - \hat{\mathbf{w}}_j^S\|^2 \Big], \end{aligned} \tag{10}$$

where α and β control the relative strengths of the two terms. It should be noted that the idea of forcing similar words to have close word embeddings is similar to the state-of-the-art retrofitting approach (Faruqui et al., 2015). However, retrofitting approach cannot be applied here because sememe-based linguistic KBs such as HowNet cannot directly provide its needed synonym list.

Sememe Embedding-based Approach

Simple and effective as the word relation-based approach is, it cannot make full use of the information of sememe-based linguistic KBs because it disregards the complicated relations between sememes and words as well as relations between different sememes. To address this limitation, we propose sememe embedding-based approach, which learns both sememe and word embeddings jointly.

In this approach, we represent sememes with distributed vectors as well and place them into the same semantic space as words. Similar to SPSE (Xie et al., 2017), which learns sememe embeddings by decomposing word-sememe matrix and sememe-sememe matrix, our method utilizes sememe embeddings as regularizers to learn better word embeddings. Different from SPSE, we do not use pre-trained word embeddings. Instead, we learn word embeddings and sememe embeddings simultaneously.

More specifically, from HowNet we can extract a source-side word-sememe matrix M^S with $M^S_{sj} = 1$ indicating word w^S_s is annotated with sememe x_j, otherwise $M^S_{sj} = 0$. Hence by factorizing M^S, we can define the loss function as:

$$\mathcal{L}_{sememe} = \sum_{w^S_s \in V^S, x_j \in X} (\mathbf{w}^S_s \cdot \mathbf{x}_j + b_s + b'_j - M^S_{sj})^2, \tag{11}$$

where b_s and b'_j are the biases of w^S_s and x_j, and X denotes sememe set.

In this approach, we obtain word and sememe embeddings in a unified semantic space. The sememe embeddings bear all the information about the relationships between words and sememes, and they inject the information into word embeddings. Therefore, the word embeddings are expected to be more suitable for sememe prediction.

3.4 Training and Prediction

Training

When training monolingual word embeddings, we use negative sampling following Mikolov et al. (2013a). In the optimization of sememe part, we adopt the iterative updating method following Faruqui et al. (2015) for word relation-based approach and stochastic gradient descent (SGD) for sememe embedding-based approach. As for the optimization of the seed lexicon term of cross-lingual part, we also apply SGD.

Nevertheless, due to the existence of the latent variable, optimization of the matching process in cross-lingual part poses a challenge. We settle on Viterbi EM algorithm to address the problem. Next, we still take the target-to-source side as an example and give a detailed description of the training process using Viterbi EM algorithm.

Viterbi EM algorithm alternates between a Viterbi E step and a subsequent M step. The Viterbi E step aims to find the most probable matched word pairs given the current parameters. Considering the independence, we can seek the match for each word individually:

$$\hat{m}_t = \mathop{\arg\max}_{s \in \{0,1,\cdots,|V^S|\}} P(w^T_t | w^S_s). \tag{12}$$

As for the parametrization of the matching probability, there are various choices. For computational simplicity, we select cosine similarity:

$$P(w^T_t | w^S_s) = \begin{cases} \epsilon & \text{if } s = 0, \\ cos(\mathbf{w}^T_t, \mathbf{w}^S_s) & \text{otherwise,} \end{cases} \tag{13}$$

where ϵ is a hyperparameter indicating the probability of matching the empty word. Therefore, the Viterbi E step computes matching by:

$$\tilde{m}_t = \mathop{\arg\max}_{s \in \{1,\cdots,|V^S|\}} cos(\mathbf{w}^T_t, \mathbf{w}^S_s), \tag{14}$$

$$\hat{m}_t = \begin{cases} \tilde{m}_t & \text{if } cos(\mathbf{w}^T_t, \mathbf{w}^S_{\tilde{m}_t}) > \epsilon, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

From this, we can see that ϵ serves as a threshold to keep out unreliable matched pairs.

The Viterbi M step performs maximization as if the latent variable has been observed in the Viterbi E step. Thus, we can treat the matched pairs as correct translations, and use a L_2 regularizer as well. Consequently, the M step computes:

$$(\hat{\mathbf{w}}^S, \hat{\mathbf{w}}^T) = \mathop{\arg\max}_{\mathbf{w}^S, \mathbf{w}^T} \mathcal{M}(\mathbf{w}^S, \mathbf{w}^T), \tag{16}$$

where $\mathcal{M}(\mathbf{w}^S, \mathbf{w}^T)$ is defined as:

$$\mathcal{M}(\mathbf{w}^S, \mathbf{w}^T) = -\sum_{t=1}^{|V^T|} \mathbb{I}[\tilde{m}_t \neq 0] \frac{c(w^T_t)}{|\mathcal{C}^T|} \|\mathbf{w}^T_t - \mathbf{w}^S_{\tilde{m}_t}\|^2. \tag{17}$$

Prediction

Since we assume that words with similar sememe annotations are similar and similar words should have similar sememes, which resembles collaborative filtering in personalized recommendation, we can recommend sememes for target words according to their most similar source words.

Formally, we define the score function $P(x_j | w^T_t)$ of sememes x_j given a target word w^T_t as:

$$P(x_j | w^T_t) = \sum_{w^S_s \in V^S} cos(\mathbf{w}^S_s, \mathbf{w}^T_t) \cdot M^S_{sj} \cdot c^{r_s}, \tag{18}$$

where r_s is the descending rank of word similarity $cos(\mathbf{w}^S_s, \mathbf{w}^T_t)$ for the source word w^S_s, and $c \in (0, 1)$ is a hyperparameter. Thus, c^{r_s} is a declined confidence factor which can eliminate the noise from irrelevant source words and concentrate on the most similar source words when predicting sememes for target words.

4 Experiments

In this section, we first introduce the dataset used in the experiments and then describe the experimental settings of both baseline method and our model. Next, we present the experimental results of different methods on the task of cross-lingual lexical sememe prediction. And then we conduct detailed analysis and exhaustive case studies. Following this, we investigate the effect of word frequency on cross-lingual sememe prediction results. Finally, we perform further quantitative analysis through two sub-tasks including bilingual lexicon induction and word similarity computation.

4.1 Dataset

We use sememe annotations in HowNet for sememe prediction. HowNet annotates sememes for $118,346$ Chinese words and $104,025$ English words. The number of sememes in total is $1,983$. Since some sememes only appear few times in HowNet, which are expected to be unimportant, we filter out those low-frequency sememes. Specifically, the frequency threshold is 5, and the final number of distinct sememes used in our experiments is $1,400$.

In our experiments, Chinese is source language and English is target language. To learn Chinese and English monolingual word embeddings, we extract about 2.0G text from Sogou-T[1] and Wikipedia[2] respectively. And we use THULAC[3] (Li and Sun, 2009) for Chinese word segmentation.

As for seed lexicon, we build it in a similar way to Zhang et al. (2017). First, we employ Google Translation API[4] to translate the source side (Chinese) vocabulary. Then the translations in the target language (English) are queried again in the reverse direction to translate back to the source language (Chinese). And we only keep the translation pairs whose back translated words match with the original source words.

In the task of bilingual lexicon induction, we opt for Chinese-English Translation Lexicon Version 3.0[5] to be the gold standard. In the task of word similarity computation, we choose WordSim-240 and WordSim-297 (Jin and Wu, 2012) datasets for Chinese, and WordSim-353 (Finkelstein et al., 2002) and SimLex-999 (Hill et al., 2015) datasets for English to evaluate the performance of our model. These datasets contain word pairs as well as human-assigned similarity scores. The word vectors are evaluated by ranking the word pairs according to their cosine similarities, and measuring Spearman's rank correlation coefficient with the human ratings.

4.2 Experimental Settings

We empirically set the dimension of word and sememe embeddings to 200. And the embeddings are all randomly initialized. In monolingual word embedding learning, we follow the optimal parameter settings in Mikolov et al. (2013a). We set the window size K to 5, down-sampling rate for high-frequency words to 10^{-5}, learning rate to 0.025 and the number of negative samples to 5. In cross-lingual word embedding alignment, the seed lexicon term weight λ_s is 0.01, and the matching term weight λ_m is $1,000$. In sememe-based word representation, the number of shared sememes for synonyms in the word relation-based approach is 2. In the training of matching process, we set ϵ to 0.5 empirically. When predicting sememes for words in target language, we only consider 100 most similar source words for each target word and the attenuation parameter c is 0.8. The testing set for cross-lingual lexical sememe prediction contains $2,000$ randomly selected English words from the vocabulary.

4.3 Cross-lingual Lexical Sememe Prediction

We evaluate our model by recommending sememes for English words. In HowNet, many words have multiple sememes, so that sememe prediction can be regarded as a multi-label classification task. We use mean average precision (MAP) and F_1 score to evaluate the sememe prediction results.

We compare our model that incorporates sememe information with word relation-based approach (named CLSP-WR) and our model which jointly trains word and sememe embeddings (named CLSP-SE) with a baseline method BiLex (Zhang et al., 2017), a bilingual WRL model without incorporation of sememe information. For BiLex, we use its trained bilingual word embeddings to predict sememes for the words in target language with our sememe prediction approach.

Table 1 exhibits the evaluation results of cross-lingual lexical sememe prediction with different

---

[1] Sogou-T is a corpus of web pages provided by a Chinese commercial search engine. `https://www.sogou.com/labs/resource/t.php`

[2] `https://dumps.wikimedia.org/`

[3] `http://thulac.thunlp.org/`

[4] `https://cloud.google.com/translate/`

[5] `https://catalog.ldc.upenn.edu/LDC2002L27`

Method	Seed Lexicon	Sememe Prediction	
		MAP	F_1 Score
BiLex	1000	27.57	16.08
	2000	33.79	22.33
	4000	35.78	25.74
	6000	38.29	28.71
CLSP-WR	1000	28.12	18.55
	2000	33.78	23.64
	4000	38.30	27.74
	6000	41.23	30.64
CLSP-SE	1000	31.78	18.22
	2000	37.70	24.31
	4000	40.77	29.33
	6000	**43.16**	**32.49**

Table 1: Evaluation results of cross-lingual lexical sememe prediction with different seed lexicon sizes.

seed lexicon sizes in {1000, 2000, 4000, 6000[6]}. From the table, we can clearly see that:

(1) Our two models perform much better compared with BiLex in all the seed lexicon size settings. It indicates that incorporating sememe information into word embeddings can effectively improve the performance of predicting sememes for target words. The reason is that both of our models make words with similar sememe annotations have similar embeddings, and as a result, we can recommend better sememes for target words according to its related source words.

(2) CLSP-SE model achieves better results than CLSP-WR model. The reason is that by representing sememes in a latent semantic space, CLSP-SE model can further capture the relatedness between sememes as well as the relatedness between words and sememes, which is helpful for modeling the representations of those words with similar sememes.

4.4 Case Study

In case study, we conduct qualitative analysis to explain the effectiveness of our models with detailed cases. We show two examples of cross-lingual word sememe prediction, in which we predict sememes for *handcuffs* and *canoeist*. Fig. 2 shows the embeddings of five closest Chinese and English words to *handcuffs* and *canoeist*, and the vector of each word is projected down to two dimensions using t-SNE (Maaten and Hinton, 2008).

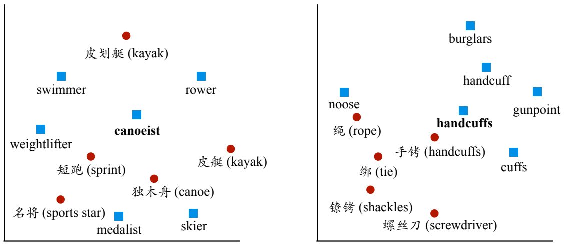

Figure 2: Two examples of nearest English and Chinese words.

Table 2 lists top-5 sememes we predict for the two words and the sememes annotated for each word in HowNet are in boldface. In the table, we also exhibit the annotated sememes of the five closest Chinese words.

In the first example, our model finds the best translated word for *handcuffs* in Chinese 手铐 "handcuffs", whose sememe annotations are exactly the same as those of *handcuffs*. In addition, the second closest Chinese word 镣铐 "shackles" is a synonym for 手铐 "handcuffs" and also has the same sememe annotations. Therefore, our model predicts all the correct sememes successfully. From the prediction results of this example, we notice that our model can accurately predict general sememes like 用具 "tool" and 人 "human", which are supposed to be difficult to predict.

In the second example, accurate Chinese translated counterpart for *canoeist* does not exist, but our model still hits all the three annotated sememes in the top-5 predicted sememes. By observing the most similar Chinese words, we can find that although these words do not have the same meaning as *canoeist*, they are related to *canoeist* in different aspects. For example, 短跑 "sprint" and *canoeist* are both in the sports domain so that they share the sememes 锻炼 "exercise" and 体育 "sport". 名将 "sports star" has the meaning of sports star and it can provide the sememe 人 "human" in sememe prediction. Furthermore, it is noteworthy that our model predicts 船 "ship" due to the nearest Chinese words 独木舟 "canoe" and 皮艇 "kayak", whereas 船 "ship" is not annotated for *canoeist* in HowNet. It is obvious that 船 "ship" is an appropriate sememe for *canoeist*. Since HowNet is manually annotated by experts, misannotated words always exist inevitably, which in some cases underestimates our models.

---

[6] The largest seed lexicon size is 6000 because that is the maximum number of translation word pairs that we can obtain from the bilingual corpora.

Type	Words	Sememes
English Word	handcuffs	**用具 “tool”**, **警 “police”**, **扣住 “detain”**, **人 “human”**, **有罪 “guilty”**
5 Nearest Chinese Words	手铐 “handcuffs”	**有罪 “guilty”**, **警 “police”**, **人 “human”**, **扣住 “detain”**, **用具 “tool”**
	镣铐 “shackles”	**有罪 “guilty”**, **警 “police”**, **人 “human”**, **扣住 “detain”**, **用具 “tool”**
	绑 “tie”	包扎 “wrap”
	螺丝刀 “screwdriver”	**用具 “tool”**, 放松 “loosen”, 勒紧 “tighten”
	绳 “rope”	线 “linear”, 材料 “material”, 拴连 “fasten”
English Word	canoeist	**锻炼 “exercise”**, **人 “human”**, **体育 “sport”**, 事情 “fact”, 船 “ship”
5 Nearest Chinese Words	短跑 “sprint”	事情 “fact” **锻炼 “exercise”** **体育 “sport”**
	独木舟 “canoe”	船 “ship”
	皮艇 “kayak”	船 “ship”
	名将 “sports star”	著名 “famous”, **人 “human”**, 官 “official”, 军 “military”
	皮划艇 “kayak”	事情 “fact”, **锻炼 “exercise”**, **体育 “sport”**

Table 2: Two examples of cross-lingual lexical sememe prediction.

4.5 Effect of Word Frequency

To explore how frequencies of target words affect cross-lingual sememe prediction results, we split the testing set into four subsets according to word frequency and then calculate the sememe prediction MAP and F_1 score for each subset. The results are shown in Table 3.

Method	Word Frequency	Sememe Prediction	
		MAP	F_1 Score
BiLex	<200	30.35	21.83
	200 - 500	34.83	25.95
	501 - 2500	40.21	28.62
	>2500	47.56	35.80
CLSP-WR	<200	34.73	24.41
	200 - 500	39.50	29.49
	501 - 2500	43.92	33.87
	>2500	47.33	34.99
CLSP-SE	<200	36.54	27.49
	200 - 500	41.46	30.09
	501 - 2500	45.35	35.01
	>2500	**49.34**	**37.16**

Table 3: Evaluation results of cross-lingual lexical sememe prediction with different word frequencies. The number of words in each frequency range is 497, 458, 522 and 523 respectively.

From the table we can see that: (1) The more frequently a target word appears in the corpus, the better its predicted sememes are. It is because high-frequency words normally have better word embeddings, which are crucial to sememe prediction. (2) Our models evidently perform better than BiLex in different word frequencies, especially in low frequency. It indicates that by considering external information of HowNet, our models are more robust and can competently handle sparse scenarios.

4.6 Further Quantitative Analysis

In this section, we conduct two typical auxiliary experiments to further analyze the superiority of our models quantitatively.

Bilingual Lexicon Induction

Our models learn bilingual word embeddings in one unified semantic space. Here we use translation top-1 and top-5 average precision (P@1 and P@5) to evaluate bilingual lexicon induction performance of our models and BiLex. The seed lexicon size also varies in {1000, 2000, 4000, 6000}.

Method	Seed Lexicon	Lexicon Induction	
		P@1	P@5
BiLex	1000	6.48	10.78
	2000	10.84	15.84
	4000	19.48	23.96
	6000	25.89	29.59
CLSP-WR	1000	6.89	11.28
	2000	11.96	18.08
	4000	19.50	25.78
	6000	25.83	31.03
CLSP-SE	1000	6.60	11.04
	2000	11.90	18.62
	4000	19.26	25.11
	6000	**26.91**	**32.17**

Table 4: Bilingual lexicon induction performance with different seed lexicon sizes.

The results are shown in Table 4. From this table, we observe that our models, especially CLSP-SE model, enhance the performance of word translation compared to BiLex no matter how large the seed lexicon is. It indicates that our models can bind bilingual word embeddings better.

Word Similarity Computation

We also evaluate the task of monolingual word similarity computation on WordSim-240 (WS-240) and WordSim-297 (WS-297) datasets for Chinese, and WordSim-353 (WS-353) and SimLex-999 (SL-999) datasets for English.

Method	Chinese (source)		English (target)	
	WS-240	WS-297	WS-353	SL-999
BiLex	60.36	62.17	60.46	27.22
CLSP-WR	61.27	65.25	60.46	27.22
CLSP-SE	60.84	65.62	62.47	28.79

Table 5: Performance on monolingual word similarity computation with seed lexicon size 6000.

Table 5 shows the results of monolingual word similarity computation on four datasets. From the table, we find that: (1) Our models perform better than BiLex on both Chinese word similarity datasets. It signifies incorporating sememe information helps learn better monolingual embeddings; (2) CLSP-WR model does not enhance English word similarity results but CLSP-SE model does. It is because CLSP-WR model only post-processes Chinese word embeddings and keeps English word embeddings unchanged while CLSP-SE model undertakes bilingual alignment and sememe information incorporation together, which makes English word embeddings improve with Chinese word embeddings.

5 Conclusion and Future Work

In this paper, we introduce a new task of cross-lingual sememe prediction. This task is very important because the construction of sememe-based linguistic knowledge bases in various languages is beneficial to better understanding these languages. We propose a simple and effective model for this task, including monolingual word representation learning, cross-lingual word representation alignment and sememe-based word representation learning. Experimental results on real-world datasets show that our model achieves consistent and significant improvements compared to baseline method in cross-lingual sememe prediction.

In the future, we will explore the following research directions: (1) In this paper, for simplification, we ignore the rich hierarchy information in HowNet and also ignore the fact that a word may have multiple senses. We will extend our models to consider the structure information of sememe and multiple senses of words; (2) In fact, our framework for cross-lingual lexical sememe prediction can be transferred to other cross-lingual tasks. We will explore the effectiveness of our model in these tasks such as cross-lingual information retrieval.

Acknowledgments

This research is funded by the National 973 project (No. 2014CB340501). It is also partially supported by the NExT++ project, the National Research Foundation, Prime Minister’s Office, Singapore under its IRC@Singapore Funding Initiative. Hao Zhu is supported by Tsinghua University Initiative Scientific Research Program. We also thank the anonymous reviewers for their valuable comments and suggestions.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Sarath Chandar AP, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proceedings of NIPS*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of ACL*.

Leonard Bloomfield. 1926. A set of postulates for the science of language. *Language*, 2(3):153–164.

Danushka Bollegala, Mohammed Alsuhaibani, Takanori Maehara, and Ken-ichi Kawarabayashi. 2016. Joint word representation learning using a corpus and a semantic lexicon. In *Proceedings of AAAI*.

Alexis Conneau, Guillaume Lample, Marc’Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, fast cross-lingual word-embeddings. In *Proceedings of EMNLP*.

Lei Dang and Lei Zhang. 2010. Method of discriminant for chinese sentence sentiment orientation based on hownet. *Application Research of Computers*, 4:43.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*.

Zhendong Dong and Qiang Dong. 2003. Hownet-a hybrid language and knowledge resource. In *Proceedings of NLP-KE*.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. In *Proceedings of EMNLP*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the EACL*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Xianghua Fu, Guo Liu, Yanyan Guo, and Zhiqiang Wang. 2013. Multi-aspect sentiment analysis for chinese online social reviews based on topic modeling and hownet lexicon. *Knowledge-Based Systems*, 37:186–195.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: fast bilingual distributed representations without word alignments. In *Proceedings of ICML*.

Yihong Gu, Jun Yan, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin, and Leyu Lin. 2018. Language modeling with sparse product of sememe experts. In *Proceedings of EMNLP*.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual distributed representations without word alignment. In *Proceedings of ICLR*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Huiming Jin, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin, and Leyu Lin. 2018. Incorporating chinese characters of words for lexical sememe prediction. In *Proceedings of ACL*.

Peng Jin and Yunfang Wu. 2012. SemEval-2012 Task 4: Evaluating chinese word similarity. In *Procedings of *SEM*.

Tomáš Kočiskỳ, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. In *Proceedings of ACL*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of ACL-IJCNLP*.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for chinese word segmentation. *Computational Linguistics*, 35(4):505–512.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of ACL-IJCNLP*.

Qun Liu and Sujian Li. 2002. Word similarity computing based on hownet. *International Journal of Computational Linguistics & Chinese Language Processing*, 7(2):59–76.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of NAANL-HLT*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*.

Laurens van der Maaten and Geoffrey E Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nikola Mrkšic, Diarmuid OSéaghdha, Blaise Thomson, Milica Gašic, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT*.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Sebastian Ruder. 2017. A survey of cross-lingual embedding models. *arXiv preprint arXiv:1706.04902*.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning cross-lingual word embeddings via matrix co-factorization. In *Proceedings of ACL-IJCNLP*.

Jingguang Sun, Dongfeng Cai, Dexin Lv, and Yanju Dong. 2007. Hownet based chinese question automatic classification. *Journal of Chinese Information Processing*, 21(1):90–95.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of ACL*.

Ivan Vulić and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of ACL*.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of ACL-IJCNLP*.

Michael Wick, Pallika Kanani, and Adam Craig Pocock. 2016. Minimally-constrained multilingual embeddings via artificial code-switching. In *Proceedings of AAAI*.

Ruobing Xie, Xingchi Yuan, Zhiyuan Liu, and Maosong Sun. 2017. Lexical sememe prediction via word embeddings and matrix factorization. In *Proceedings of AAAI*.

Xiangkai Zeng, Cheng Yang, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. 2018. Chinese liwc lexicon expansion via hierarchical classification of word embeddings with sememe attention. In *Proceedings of AAAI*.

Meng Zhang, Haoruo Peng, Yang Liu, Huan-Bo Luan, and Maosong Sun. 2017. Bilingual lexicon induction from non-parallel data with minimal supervision. In *Proceedings of AAAI*.

Yuntao Zhang, Ling Gong, and Yongcheng Wang. 2005. Chinese word sense disambiguation using hownet. In *Proceedings of International Conference on Natural Computation*.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*.

Neural Cross-Lingual Named Entity Recognition with Minimal Resources

Jiateng Xie,[1] **Zhilin Yang,**[1]**Graham Neubig,**[1]
Noah A. Smith,[2,3] **and Jaime Carbonell**[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington
[3]Allen Institute for Artificial Intelligence
{jiatengx,zhiliny,gneubig,jgc}@cs.cmu.edu,
nasmith@cs.washington.edu

Abstract

For languages with no annotated resources, unsupervised transfer of natural language processing models such as named-entity recognition (NER) from resource-rich languages would be an appealing capability. However, differences in words and word order across languages make it a challenging problem. To improve mapping of lexical items across languages, we propose a method that finds translations based on bilingual word embeddings. To improve robustness to word order differences, we propose to use self-attention, which allows for a degree of flexibility with respect to word order. We demonstrate that these methods achieve state-of-the-art or competitive NER performance on commonly tested languages under a cross-lingual setting, with much lower resource requirements than past approaches. We also evaluate the challenges of applying these methods to Uyghur, a low-resource language.[1]

1 Introduction

Named entity recognition (NER), the task of detecting and classifying named entities from text into a few predefined categories such as people, locations or organizations, has seen the state-of-the-art greatly advanced by the introduction of neural architectures (Collobert et al., 2011; Huang et al., 2015; Chiu and Nichols, 2016; Lample et al., 2016; Yang et al., 2016; Ma and Hovy, 2016; Peters et al., 2017; Liu et al., 2018; Peters et al., 2018). However, the success of these methods is highly dependent on a reasonably large amount of annotated training data, and thus it remains a challenge to apply these models to languages with limited amounts of labeled data. Cross-lingual NER attempts to address this challenge by transferring knowledge from a high-resource source language with abundant entity labels to a low-resource target language with few or no labels. Specifically, in this paper we attempt to tackle the extreme scenario of *unsupervised transfer*, where no labeled data is available in the target language. Within this paradigm, there are two major challenges to tackle: how to effectively perform lexical mapping between the languages, and how to address word order differences.

To cope with the first challenge of lexical mapping, a number of methods use parallel corpora to project annotations between languages through word alignment (Ehrmann et al., 2011; Kim et al., 2012; Wang and Manning, 2014; Ni et al., 2017). Since parallel corpora may not be always available, Mayhew et al. (2017) proposed a "cheap translation" approach that uses a bilingual dictionary to perform word-level translation. The above approaches provide a reasonable proxy for the actual labeled training data, largely because the words that participate in entities can be translated relatively reliably given extensive parallel dictionaries or corpora (e.g., with 1 million word pairs or sentences). Additionally, as a side benefit of having explicitly translated words, models can directly exploit features extracted from the surface forms (e.g. through character-level neural feature extractors), which has proven essential for high accuracy in the monolingual scenario (Ma and Hovy, 2016). However, these methods are largely predicated on the availability of large-scale parallel resources, and thus, their applicability to low-resource languages is limited.

In contrast, it is also possible to learn lexical mappings through bilingual word embeddings (BWE). These bilingual embeddings can be obtained by using a small dictionary to project two sets of embeddings into a consistent space (Mikolov et al., 2013a; Faruqui and Dyer,

---

[1] The source code is available at `https://github.com/thespectrewithin/cross-lingual_NER`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 369–379
Brussels, Belgium, October 31 - November 4, 2018.

2014; Artetxe et al., 2016; Smith et al., 2017), or even in an entirely unsupervised manner using adversarial training or identical character strings (Zhang et al., 2017; Artetxe et al., 2017; Lample et al., 2018). Many approaches in the past have leveraged the shared embedding space for cross-lingual applications (Guo et al., 2015; Ammar et al., 2016b; Zhang et al., 2016; Fang and Cohn, 2017), including NER (Bharadwaj et al., 2016; Ni et al., 2017). The minimal dependency on parallel resources makes the embedding-based method much more suitable for low-resource languages. However, since different languages have different linguistic properties, it is hard, if not impossible, to align the two embedding spaces perfectly (see Figure 1). Meanwhile, because surface forms are not available, character-level features cannot be used, resulting in reduced tagging accuracy (as demonstrated in our experiments).

To address the above issues, we propose a new lexical mapping approach that combines the advantages of both discrete dictionary-based methods and continuous embedding-based methods. Specifically, we first project embeddings of different languages into the shared BWE space, then learn discrete word translations by looking for nearest neighbors in this projected space, and finally train a model on the translated data. This allows our method to inherit the benefits of both embedding-based and dictionary-based methods: its resource requirements are low as in the former, but it suffers less from misalignment of the embedding spaces and has access to character-level information like the latter.

Turning to differences in word ordering, to our knowledge there are no methods that explicitly deal with this problem in unsupervised cross-lingual transfer for NER. Our second contribution is a method to alleviate this issue by incorporating an order-invariant self-attention mechanism (Vaswani et al., 2017; Lin et al., 2017) into our neural architecture. Self-attention allows re-ordering of information within a particular encoded sequence, which makes it possible to account for word order differences between the source and the target languages.

In our experiments, we start with models trained in English as the source language on the CoNLL 2002 and 2003 datasets and transfer them into Spanish, Dutch, and German as the target languages. Our approach obtains new state-of-the-art cross-lingual results in Spanish and Dutch, and competitive results in German, even without a dictionary, completely removing the need for resources such as Wikipedia and parallel corpora. Next, we transfer English using the same approach into Uyghur, a truly low-resource language. With significantly fewer cross-lingual resources, our approach can still perform competitively with previous best results.

2 Approach

We establish our problem setting (§2.1), then present our methods in detail (§2.2), and provide some additional motivation (§2.3).

2.1 Problem Setting

NER takes a sentence as the input and outputs a sequence of labels corresponding to the named entity categories of the words in the sentence, such as location, organization, person, or none. In standard supervised NER, we are provided with a labeled corpus of sentences in the target language along with tags indicating which spans correspond to entities of each type.

As noted in the introduction, we study the problem of unsupervised cross-lingual NER: given labeled training data only in a separate source language, we aim to learn a model that is able to perform NER in the target language. This transfer can be performed using a variety of resources, including parallel corpora (Täckström et al., 2012; Ni et al., 2017), Wikipedia (Nothman et al., 2013), and large dictionaries (Ni et al., 2017; Mayhew et al., 2017). In this work, we limit ourselves to a setting where we have the following resources, making us comparable to other methods such as Mayhew et al. (2017) and Ni et al. (2017):

- Labeled training data in the source language.
- Monolingual corpora in both source and target languages.
- A dictionary, either a small pre-existing one, or one induced by unsupervised methods.

2.2 Method

Our method follows the process below:

1. Train separate word embeddings using monolingual corpora using standard embedding training methods (§2.2.1).
2. Project word embeddings in the two languages into a shared embedding space by optimizing

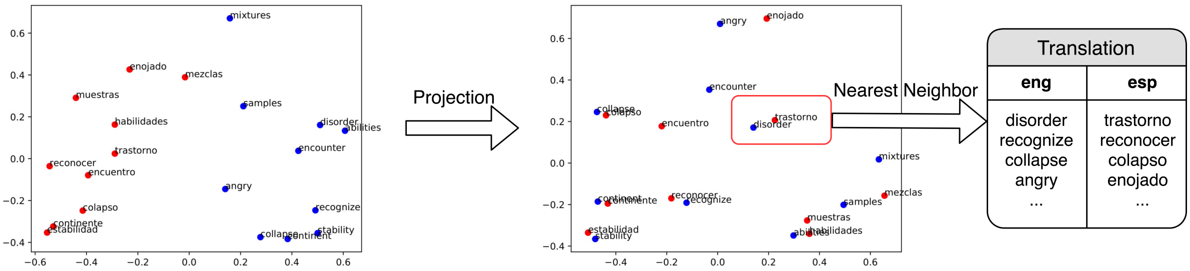

Figure 1: Example of the result of our approach on Spanish-English words not included in the dictionary (embeddings are reduced to 2 dimensions for visual clarity). We first project word embeddings into a shared space, and then use the nearest neighbors for word translation. Notice that the word pairs are not perfectly aligned in the shared embedding space, but after word translation we obtain correct alignments.

the word embedding alignment using the given dictionary (§2.2.2).

3. For each word in the source language training data, translate it by finding its nearest neighbor in the shared embedding space (§2.2.3).
4. Train an NER model using the translated words along with the named entity tags from the English corpus (§2.2.4).

We consider each in detail.

2.2.1 Learning Monolingual Embeddings

Given text in the source and target language, we first independently learn word embedding matrices X and Y in the source and target languages respectively. These embeddings can be learned on monolingual text in both languages with any of the myriad of word embedding methods (Mikolov et al., 2013b; Pennington et al., 2014; Bojanowski et al., 2017).

2.2.2 Learning Bilingual Embeddings

Next, we learn a cross-lingual projection of X and Y into a shared space. Assume we are given a dictionary $\{x_i, y_i\}_{i=1}^{D}$, where x_i and y_i denote the embeddings of a word pair. Let $X_D = [x_1, x_2, \cdots, x_D]^\top$ and $Y_D = [y_1, y_2, \cdots, y_D]^\top$ denote two embedding matrices consisting of word pairs from the dictionary.

Following previous work (Zhang et al., 2016; Artetxe et al., 2016; Smith et al., 2017), we optimize the following objective:

$$\min_{W} \sum_{i=1}^{d} \|Wx_i - y_i\|^2 \;\; \text{s.t.} \;\; WW^\top = I,$$

where W is a square parameter matrix. This objective can be further simplified as

$$\max_{W} \text{Tr}(X_D W Y_D^\top) \;\; \text{s.t.} \;\; WW^\top = I.$$

Here, the transformation matrix W is constrained to be orthogonal so that the dot product similarity of words is invariant with respect to the transformation both within and across languages.

To optimize the above objective (the Procrustes problem), we decompose the matrix $Y_D^\top X_D$ using singular value decomposition. Let the results be $Y_D^\top X_D = U \sum V^\top$, then $W = UV^\top$ gives the exact solution. We define the similarity matrix between X and Y to be $S = YWX^\top = YU(XV)^\top$, where each column contains the cosine similarity between source word x_i and all target words y_i. We can then define $X' = XV$ and $Y' = YU$, which are X and Y transformed into a shared embedding space.

To refine the alignment in this shared space further, we iteratively perform a self-learning refinement step k [2] times by:

1. Using the aligned embeddings to generate a new dictionary that consists of mutual nearest neighbors obtained using the same metric as introduced below.
2. Solving the Procrustes problem based on the newly generated dictionary to get a new set of bilingual embeddings.

The bilingual embeddings at the end of the kth step, X'_k and Y'_k, will be used to perform translation.

2.2.3 Learning Word Translations

To learn actual word translations, we next proceed to perform nearest-neighbor search in the

---

[2] We use $k = 3$ in this paper.

common space. Instead of using a common distance metric such as cosine similarity, we adopt the cross-domain similarity local scaling (CSLS) metric (Lampple et al., 2018), which is designed to address the hubness problem common to the shared embedding space (Dinu and Baroni, 2014). Specifically,

$$\text{CSLS}(x_i, y_j) = 2\cos(x_i, y_j) - r_T(x_i) - r_S(y_j)$$

where $r_T(x_i) = \frac{1}{K}\sum_{y_t \in N_T(x_i)} \cos(x_i, y_t)$ denotes the mean cosine similarity between x_i and its K neighbors y_t. Using this metric, we find translations for each source word s by selecting target word $\hat{t_s}$ where $\hat{t_s} = \arg\max_t \text{CSLS}(x_s, y_t)$.

2.2.4 Training the NER Model

Finally, we translate the entire English NER training data into the target language by taking English sentences $S = s_1, s_2, ..., s_n$ and translating them into target sentences $\hat{T} = \hat{t_1}, \hat{t_2}, ..., \hat{t_n}$. The label of each English word is copied to be the label of the target word. We can then train an NER model directly using the translated data. Notably, because the model has access to the surface forms of the target sentences, it can use the character sequences of the target language as part of its input.

During learning, all word embeddings are normalized to lie on the unit ball, allowing every training pair an equal contribution to the objective and improving word translation accuracy (Artetxe et al., 2016). When training the NER model, however, we do not normalize the word embeddings, because preliminary experiments showed the original unnormalized embeddings gave superior results. We suspect this is due to frequency information conveyed by vector length, an important signal for NER. (Named entities appear less frequently in the monolingual corpus.)

2.3 Discussion

Figure 1 shows an example of the embeddings and translations learned with our approach trained on Spanish and English data from the experiments (see §4 for more details). As shown in the figure, there is usually a noticeable difference between the word embeddings of a word pair in different languages, which is inevitable because different languages have distinct traits and different monolingual data, and as a result it is intrinsically hard to learn a perfect alignment. This indicates that models trained directly on data using the source embeddings may not generalize well to the slightly different embeddings of the target language.

Instead of directly modeling the shared embedding space (Guo et al., 2015; Zhang et al., 2016; Fang and Cohn, 2017; Ni et al., 2017), we leverage the shared embedding space for word translation. As shown in Figure 1, unaligned word pairs can still be translated correctly with our method, as the embeddings are still closer to the correct translations than the closest incorrect one.

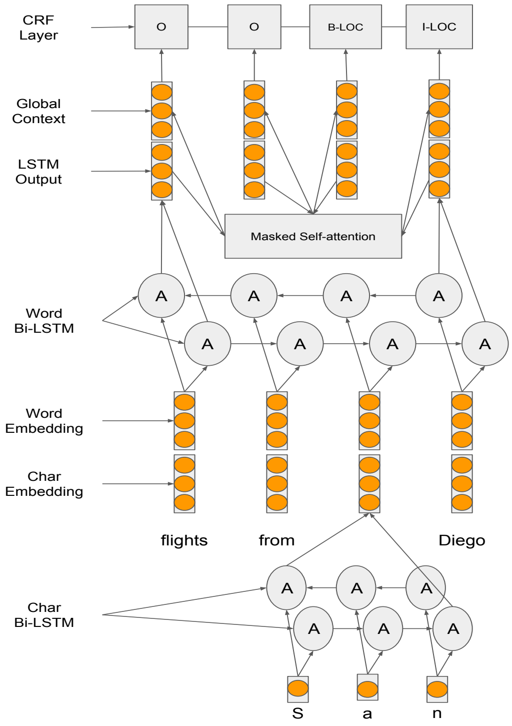

Figure 2: Self-attentive Bi-LSTM-CRF Model

3 NER Model Architecture

We describe the model we use to perform NER. We will first describe the basic hierarchical neural CRF tagging model (Lample et al., 2016; Ma and Hovy, 2016; Yang et al., 2016), and introduce the self-attention mechanism that we propose to deal with divergence of word order.

3.1 Hierarchical Neural CRF

The hierarchical CRF model consists of three components: a character-level neural network, either an RNN or a CNN, that allows the model to capture subword information, such as morphological variations and capitalization patterns; a word-level neural network, usually an RNN, that consumes word representations and produces context sensitive hidden representations for each word; and a linear-chain CRF layer that models the dependency between labels and performs inference.

In this paper, we closely follow the architecture proposed by Lample et al. (2016), and use bidirectional LSTMs for both the character level and word level neural networks. Specifically, given an input sequence of words $(w_1, w_2, ..., w_n)$, and each word's corresponding character sequence, the model first produces a representation for each word, x_i, by concatenating its character representation with its word embedding. Subsequently, the word representations of the input sequence $(x_1, x_2, \cdots, x_n)$ are fed into a word level Bi-LSTM, which models the contextual dependency within each sentence and outputs a sequence of context sensitive hidden representations $(h_1, h_2, \cdots, h_n)$. A CRF layer is then applied on top of the word level LSTM and takes in as its input the sequence of hidden representations $(h_1, h_2, \cdots, h_n)$, and defines the joint distribution of all possible output label sequences. The Viterbi algorithm is used during decoding.

3.2 Self-Attention

The training-time inputs to our model are in essence corrupted sentences from the target language (e.g., Spanish), which have a different order from natural target sentences. We propose to alleviate this problem by adding a self-attention layer (Vaswani et al., 2017) on top of the word-level Bi-LSTM. Self-attention provides each word with a context feature vector based on *all* the words of a sentence. As the context vectors are obtained irrespective of the words' positions in a sentence, at test time, the model is more likely to see vectors similar to those seen at training time, which we posit introduces a level of flexibility with respect to the word order, and thus may allow for better generalization.

Let $H = [h_1, h_2, \cdots, h_n]^\top$ be a sequence of word-level hidden representations. We apply a single layer MLP on H to obtain the queries Q and keys $K = \tanh(HW + b)$, where $W \in \mathbb{R}^{d \times d}$ is a parameter matrix and $b \in \mathbb{R}^d$ is a bias term, with d being the hidden state size. The output of attention layer is defined as:

$$\begin{aligned} H^a &= \text{softmax}(QK^\top) \odot (E - I)H \\ &= [h_1^a, h_2^a, ..., h_3^a]^\top \end{aligned}$$

where I is an identity matrix and E is an all-one matrix. The term $(E - I)$ serves as an attention mask that prevents the weights from centering on the word itself, as we would like to provide each word with sentence level context. The outputs from the self-attention layer are then concatenated with the original hidden representations to form the final inputs to the CRF layer, which are $([h_1, h_1^a], [h_2, h_2^a], ..., [h_3, h_3^a])$.

4 Experiments

To examine the effectiveness of both of our proposed methods, we conduct four sets of experiments. First, we evaluate our model both with and without provided dictionaries on a benchmark NER dataset and compare with previous state-of-the-art results. Second, we compare our methods against a recently proposed dictionary-based translation baseline (Mayhew et al., 2017) by directly applying our model on their translated data.[3] Subsequently, we conduct an ablation study to further understand our proposed methods. Lastly, we apply our methods to a truly low-resource language, Uyghur.

4.1 Experimental Settings

We evaluate our proposed methods on the benchmark CoNLL 2002 and 2003 NER datasets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003), which contain 4 European languages, English, German, Dutch and Spanish. For all experiments, we use English as the source language and translate its training data into the target language. We train a model on the translated data, and test it on the target language. For each experiment, we run our models 5 times using different seeds and report the mean and standard deviation, as suggested by Reimers and Gurevych (2017).

Word Embeddings For all languages, we use two different embedding methods, fastText (Bojanowski et al., 2017) and GloVe (Pennington et al., 2014), to perform word-embedding based translations and train the NER model, respectively. For fastText, we use the publicly available embeddings trained on Wikipedia for all languages. For GloVe, we use the publicly available embeddings pre-trained on Gigaword and Wikipedia for English. For Spanish, German and Dutch, we use Spanish Gigaword and Wikipedia, German WMT News Crawl data and Wikipedia, and Dutch

---

[3] We thank the authors of Mayhew et al. (2017) for sharing their data.

Wikipedia, respectively, to train the GloVe word embeddings. We use a vocabulary size of 100,000 for both embedding methods.

Dictionary We consider three different settings to obtain the seed dictionary, including two methods that do not use parallel resources:

1. Use identical character strings shared between the two vocabularies as the seed dictionary.
2. Lample et al. (2018)'s method of using adversarial learning to induce a mapping that aligns the two embedding spaces, and the mutual nearest neighbors in the shared space will be used as a dictionary. The learning procedure is formulated as a two player game, where a discriminator is trained to distinguish words from the two embedding spaces, and a linear mapping is trained to align the two embedding spaces and thus fool the discriminator.
3. Use a provided dictionary. In our experiments, we use the ones provided by Lample et al. (2018),[4] each of which contain 5,000 source words and about 10,000 entries.

Translation We follow the general procedure described in Section 2, and replace each word from the English training data with its corresponding word in the target language. For out-of-vocabulary (OOV) words, we simply keep them as-is. We capitalize the resulting sentences following the pattern of the original English words. Note that for German, simply following the English capitalization pattern does not work, because all nouns in German are capitalized. To handle this problem, we count the number of times each word is capitalized in Wikipedia, and capitalize the word if the probability is greater than 0.6.

Network Parameters For our experiments, we set the character embedding size to be 25, character level LSTM hidden size to be 50, and word level LSTM hidden size to be 200. For OOV words, we initialize an unknown embedding by uniformly sampling from range $[-\sqrt{\frac{3}{\text{emb}}}, +\sqrt{\frac{3}{\text{emb}}}]$, where emb is the size of embedding, 100 in our case. We replace each number with 0 when used as input to the character level Bi-LSTM.

Network Training We use SGD with momentum to train the NER model for 30 epochs, and select the best model on the target language development set. We choose the initial learning rate to be $\eta_0 = 0.015$, and update it using a learning decay mechanism after each epoch, $\eta_t = \frac{\eta_0}{1+\rho t}$, where t is the number of completed epoch and $\rho = 0.05$ is the decay rate. We use a batch size of 10 and evaluate the model per 150 batches within each epoch. We apply dropout on the inputs to the word-level Bi-LSTM, the outputs of the word-level Bi-LSTM, and the outputs of the self-attention layer to prevent overfitting. The self-attention dropout rate is set to 0.5 when using our translated data, and 0.2 when using cheap-translation data. We use 0.5 for all other dropouts. The word embeddings are not fine-tuned during training.

4.2 Results

Table 1 presents our results on transferring from English to three other languages, alongside results from previous studies. Here "BWET" (bilingual word embedding translation) denotes using the hierarchical neural CRF model trained on data translated from English. As can be seen from the table, our methods outperform previous state-of-the-art results on Spanish and Dutch by a large margin and perform competitively on German even without using any parallel resources. We achieve similar results using different seed dictionaries, and produce the best results when adding the self-attention mechanism to our model.

Despite the good performance on Spanish and Dutch, our model does not outperform the previous best result on German, and we speculate that there are a few reasons. First, German has rich morphology and contains many compound words, making the word embeddings less reliable. Our supervised result on German indicates the same problem, as it is about 8 F_1 points worse than Spanish and Dutch. Second, these difficulties become more pronounced in the cross-lingual setting, leading to a noisier embedding space alignment, which lowers the quality of BWE-based translation. We believe that this is a problem with all methods using word embeddings. In such cases, more resource-intensive methods may be necessary.

4.2.1 Comparison with Dictionary-Based Translation

Table 1 also presents results of a comparison between our proposed BWE translation method and the "cheap translation" baseline of (Mayhew et al., 2017). The size of the dictionaries used by both

---

[4] `https://github.com/facebookresearch/MUSE`

Model	Spanish	Dutch	German	Extra Resources
* Täckström et al. (2012)	59.30	58.40	40.40	parallel corpus
* Nothman et al. (2013)	61.0	64.00	55.80	Wikipedia
* Tsai et al. (2016)	60.55	61.60	48.10	Wikipedia
* Ni et al. (2017)	65.10	65.40	58.50	Wikipedia, parallel corpus, 5K dict.
*† Mayhew et al. (2017)	65.95	66.50	**59.11**	Wikipedia, 1M dict.
* Mayhew et al. (2017) (only Eng. data)	51.82	53.94	50.96	1M dict.
Our methods:				
BWET (id.c.)	71.14 ± 0.60	70.24 ± 1.18	57.03 ± 0.25	–
BWET (id.c.) + self-att.	**72.37** ± 0.65	70.40 ± 1.16	**57.76** ± 0.12	–
BWET (adv.)	70.54 ± 0.85	70.13 ± 1.04	55.71 ± 0.47	–
BWET (adv.) + self-att.	71.03 ± 0.44	**71.25** ± 0.79	56.90 ± 0.76	–
BWET	71.33 ± 1.26	69.39 ± 0.53	56.95 ± 1.20	10K dict.
BWET + self-att.	71.67 ± 0.86	70.90 ± 1.09	57.43 ± 0.95	10K dict.
* BWET on data from Mayhew et al. (2017)	66.53 ± 1.12	69.24 ± 0.66	55.39 ± 0.98	1M dict.
* BWET + self-att. on data from Mayhew et al. (2017)	66.90 ± 0.65	69.31 ± 0.49	55.98 ± 0.65	1M dict.
* Our supervised results	86.26 ± 0.40	86.40 ± 0.17	78.16 ± 0.45	annotated corpus

Table 1: NER F_1 scores. *Approaches that use more resources than ours ("Wikipedia" means Wikipedia is used not as a monolingual corpus, but to provide external knowledge). †Approaches that use multiple languages for transfer. "Only Eng. data" is the model used in Mayhew et al. (2017) trained on their data translated from English without using Wikipedia and other languages. The "data from Mayhew et al. (2017)" is the same data translated from only English they used. "Id.c." indicates using identical character strings between the two languages as the seed dictionary. "Adv." indicates using adversarial training and mutual nearest neighbors to induce a seed dictionary. Our supervised results are obtained using models trained on annotated corpus from CoNLL.

approaches are given in the right-most column. Using our model on their translated data from English outperforms the baseline scores produced by their models over all languages, a testament to the strength of our neural CRF baseline. The results produced by our model on their data indicate that our approach is effective, as we manage to outperform their approaches on all three languages using much smaller dictionaries and even without dictionaries. Also, we see that self-attention is effective when applied on their data, which also does not carry the correct word order.

4.2.2 Why Does Translation Work Better?

In this section, we study the effects of different ways of using bilingual word embeddings and the resulting induced translations. As we pointed out previously, finding translations has two advantages: (1) the model can be trained on the exact points from the target embedding space, and (2) the model has access to the target language's original character sequences. Here, we conduct ablation studies over these two variables. Specifically, we consider the following three variants.[5]

- **Common space** This is the most common setting for using bilingual word embeddings, and has recently been applied in NER (Ni et al., 2017). In short, the source and target word embeddings are cast into a common space, namely $X' = XV$ and $Y' = YU$, and the model is trained with the source side embedding and the source character sequence, and directly applied on the target side.

- **Replace** In this setting, we replace each original word embedding x_i with its nearest neighbor y_i in the common space but do not perform translation. This way, the model will be trained with target word embeddings and source-side character sequences.

- **Translation** This is our proposed approach, where the model is trained on both exact points in the target space and target language character sequences.

The three variants are compared in Table 2. The "common space" variant performs the worst by a large margin, confirming our hypothesis that discrepancy between the two embedding spaces harms the model's ability to generalize. From the comparison between the "replace" and "translation," we observe that having access to the target language's character sequence helps performance, especially for German, perhaps due in part to its capitalization patterns, which differ from English. In this case, we have to lower-case all the words for character inputs in order to prevent the model from overfitting the English capitalization pattern.

---

[5] In this study, we use GloVe for learning bilingual embeddings and word translations instead of fastText.

Model	Spanish	Dutch	German
Common space	65.40 ± 1.22	66.15 ± 1.62	43.73 ± 0.94
Replace	68.21 ± 1.22	69.37 ± 1.33	48.59 ± 1.21
Translation	**69.21** ± 0.95	**69.39** ± 1.21	**53.94** ± 0.66

Table 2: Comparison of different ways of using bilingual word embeddings, within our method (NER F_1).

	Model	Uyghur Unsequestered Set	Extra Resources
*†	Mayhew et al. (2017)	**51.32**	Wikipedia, 100K dict.
*	Mayhew et al. (2017) (only Eng. data)	27.20	Wikipedia, 100K dict.
	BWET	25.73 ± 0.89	5K dict.
	BWET + self-att.	26.38 ± 0.34	5K dict.
*	BWET on data from Mayhew et al. (2017)	30.20 ± 0.98	Wikipedia, 100K dict.
*	BWET + self-att. on data from Mayhew et al. (2017)	30.68 ± 0.45	Wikipedia, 100K dict.
*	Combined (see text)	31.61 ± 0.46	Wikipedia, 100K dict., 5K dict.
*	Combined + self-att.	**32.09** ± 0.61	Wikipedia, 100K dict., 5K dict.

Table 3: NER F_1 scores on Uyghur. *Approaches using language-specific features and resources ("Wikipedia" means Wikipedia is used not as a monolingual corpus, but to provide external knowledge). †Approaches that transfer from multiple languages and use language-specific techniques.

4.3 Case Study: Uyghur

In this section, we directly apply our approach to Uyghur, a truly low-resource language with very limited monolingual and parallel resources. We test our model on 199 annotated evaluation documents from the DARPA LORELEI program (the "unsequestered set") and compare with previously reported results in the cross-lingual setting by Mayhew et al. (2017). Similar to our previous experiments, we transfer from English, use fastText embeddings trained on Common Crawl and Wikipedia[6] and a provided dictionary to perform translation, and use GloVe trained on a monolingual corpus that has 30 million tokens to perform NER. Results are presented in Table 3.

Our method performs competitively, considering that we use a much smaller dictionary than Mayhew et al. (2017) and no knowledge from Wikipedia in Uyghur. Our best results come from a combined approach: using word embeddings to translate words that are not covered by Mayhew et al. (2017)'s dictionary (last line of Table 3). Note that for the CoNLL languages, Mayhew et al. (2017) used Wikipedia for the Wikifier features (Tsai et al., 2016), while for Uyghur they used it for translating named entities, which is crucial for low-resource languages when some named entities are not covered by the dictionary or the translation is not reliable. We suspect that the unreliable translation of named entities is the major reason why our method alone performs worse but performs better when combined with their data that has access to higher quality translations of named entities.

The table omits results using adversarial learning and identical character strings, as both failed (F_1 scores around 10). We attribute these failures to the low quality of Uyghur word embeddings and the fact that the two languages are distant. Also, Uyghur is mainly written in Arabic script, making the identical character method inappropriate. Overall, this reveals a practical challenge for multilingual embedding methods, where the underlying distributions of the text in the two languages are divergent.

5 Related Work

Cross-Lingual Learning Cross-lingual learning approaches can be loosely classified into two categories: annotation projection and language-independent transfer.

Annotation projection methods create training data by using parallel corpora to project annotations from the source to the target language. Such approaches have been applied to many tasks under the cross-lingual setting, such as POS tagging (Yarowsky et al., 2001; Das and Petrov, 2011; Täckström et al., 2013; Fang and Cohn, 2016), mention detection (Zitouni and Florian, 2008) and parsing (Hwa et al., 2005; McDonald et al., 2011).

Language independent transfer-based approaches build models using language independent and delexicalized features. For instance,

---

[6] `https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md`

Zirikly and Hagiwara (2015) transfers word cluster and gazetteer features through the use of comparable copora. Tsai et al. (2016) links words to Wikipedia entries and uses the entry category as features to train language independent NER models. Recently, Ni et al. (2017) propose to project word embeddings into a common space as language independent features. These approaches utilize such features by training a model on the source language and directly applying it to the target language.

Another way of performing language independent transfer resorts to multi-task learning, where a model is trained jointly across different languages by sharing parameters to allow for knowledge transfer (Ammar et al., 2016a; Yang et al., 2017; Cotterell and Duh, 2017; Lin et al., 2018). However, such approaches usually require some amounts of training data in the target language for bootstrapping, which is different from our unsupervised approach that requires no labeled resources in the target language.

Bilingual Word Embeddings There have been two general paradigms in obtaining bilingual word vectors besides using dictionaries: through parallel corpora and through joint training. Approaches based on parallel corpora usually learn bilingual word embeddings that can produce similar representations for aligned sentences (Hermann and Blunsom, 2014; Chandar et al., 2014). Jointly-trained models combine the common monolingual training objective with a cross-lingual training objective that often comes from parallel corpus (Zou et al., 2013; Gouws et al., 2015). Recently, unsupervised approaches also have been used to align two sets of word embeddings by learning a mapping through adversarial learning or self-learning (Zhang et al., 2017; Artetxe et al., 2017; Lample et al., 2018).

6 Conclusion

In this paper, we propose two methods to tackle the cross-lingual NER problem under the unsupervised transfer setting. To address the challenge of lexical mapping, we find translations of words in a shared embedding space built from a seed lexicon. To alleviate word order divergence across languages, we add a self-attention mechanism to our neural architecture. With these methods combined, we are able to achieve state-of-the-art or competitive results on commonly tested languages under a cross-lingual setting, with lower resource requirements than past approaches. We also evaluate the challenges of applying these methods to an extremely low-resource language, Uyghur.

Acknowledgments

We thank Stephen Mayhew for sharing the data, and Zihang Dai for meaningful discussion.

This research was sponsored by Defense Advanced Research Projects Agency Information Innovation Office (I2O) under the Low Resource Languages for Emergent Incidents (LORELEI) program, issued by DARPA/I2O under Contract No. HR0011-15-C0114. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation here on.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016a. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016b. Massively multilingual word embeddings. `https://arxiv.org/pdf/1602.01925`.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *EMNLP*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *ACL*, pages 451–462.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *EMNLP*, pages 1462–1472.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *NIPS*, pages 1853–1861.

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

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

Ryan Cotterell and Kevin Duh. 2017. Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields. In *IJCNLP*, pages 91–96.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *ACL*, pages 600–609.

Georgiana Dinu and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *CoRR*, abs/1412.6568.

Maud Ehrmann, Marco Turchi, and Ralf Steinberger. 2011. Building a multilingual named entity-annotated corpus using annotation projection. In *RANLP*, pages 118–124.

Meng Fang and Trevor Cohn. 2016. Learning when to trust distant supervision: An application to low-resource POS tagging using cross-lingual projection. In *CoNLL*, pages 178–186.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *ACL*, pages 587–593.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *ACL*, pages 462–471.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *ICML*, pages 748–756.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *ACL*, volume 1, pages 1234–1244.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *ACL*, pages 58–68.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(3):311–325.

Sungchul Kim, Kristina Toutanova, and Hwanjo Yu. 2012. Multilingual named entity recognition using parallel data and metadata from wikipedia. In *ACL*, pages 694–702.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*, pages 260–270.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herv Jgou. 2018. Word translation without parallel data. In *ICLR*.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *ACL*, pages 799–809.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

L. Liu, J. Shang, F. Xu, X. Ren, H. Gui, J. Peng, and J. Han. 2018. Empower sequence labeling with task-aware neural language model. In *AAAI*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*, pages 1064–1074.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *EMNLP*, pages 2526–2535.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *EMNLP*, pages 62–72.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *ACL*, pages 1470–1480.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194:151–175.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*, pages 1756–1765.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, pages 2227–2237.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *EMNLP*, pages 338–348.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *ICLR*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan T. McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *TACL*, 1:1–12.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *NAACL*, pages 477–487.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *CoNLL*, pages 1–4.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*, pages 142–147.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *CoNLL*, pages 219–228.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

Mengqiu Wang and Christopher D. Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics (TACL)*, 2(5):55–66.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *CoRR*, abs/1603.06270.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks.

D. Yarowsky, G. Ngai, and R. Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *HLT*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *ACL*, volume 1, pages 1959–1970.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi S. Jaakkola. 2016. Ten pairs to tag - multilingual POS tagging via coarse mapping between embeddings. In *NAACL*, pages 1307–1317.

Ayah Zirikly and Masato Hagiwara. 2015. Cross-lingual transfer of named entity recognizers without parallel corpora. In *ACL*, pages 390–396. Association for Computational Linguistics.

Imed Zitouni and Radu Florian. 2008. Mention detection crossing the language barrier. In *EMNLP*, pages 600–609.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.

A Stable and Effective Learning Strategy for Trainable Greedy Decoding

Yun Chen[†], **Victor O.K. Li**[†], **Kyunghyun Cho**[‡,⋆], **Samuel R. Bowman**[‡]
[†]The University of Hong Kong, [‡]New York University, [⋆]CIFAR Global Scholar
yun.chencreek@gmail.com, vli@eee.hku.hk,
kyunghyun.cho@nyu.edu, bowman@nyu.edu

Abstract

Beam search is a widely used approximate search strategy for neural network decoders, and it generally outperforms simple greedy decoding on tasks like machine translation. However, this improvement comes at substantial computational cost. In this paper, we propose a flexible new method that allows us to reap nearly the full benefits of beam search with nearly no additional computational cost. The method revolves around a small neural network *actor* that is trained to observe and manipulate the hidden state of a previously-trained decoder. To train this actor network, we introduce the use of a pseudo-parallel corpus built using the output of beam search on a base model, ranked by a target quality metric like BLEU. Our method is inspired by earlier work on this problem, but requires no reinforcement learning, and can be trained reliably on a range of models. Experiments on three parallel corpora and three architectures show that the method yields substantial improvements in translation quality and speed over each base system.

1 Introduction

Neural network sequence decoders yield state-of-the-art results for many text generation tasks, including machine translation (Bahdanau et al., 2015; Luong et al., 2015; Gehring et al., 2017; Vaswani et al., 2017; Dehghani et al., 2018), text summarization (Rush et al., 2015; Ranzato et al., 2015; See et al., 2017; Paulus et al., 2017) and image captioning (Vinyals et al., 2015; Xu et al., 2015). These decoders generate tokens from left to right, at each step giving a distribution over possible next tokens, conditioned on both the input and all the tokens generated so far. However, since the space of all possible output sequences is infinite and grows exponentially with sequence length, heuristic search methods such as greedy decoding or beam search (Graves, 2012; Boulanger-Lewandowski et al., 2013) must be used at decoding time to select high-probability output sequences. Unlike greedy decoding, which selects the token of the highest probability at each step, beam search expands all possible next tokens at each step, and maintains the k most likely prefixes, where k is the beam size. Greedy decoding is very fast—requiring only a single run of the underlying decoder—while beam search requires an equivalent of k such runs, as well as substantial additional overhead for data management. However, beam search often leads to substantial improvement over greedy decoding. For example, Ranzato et al. (2015) report that beam search (with $k = 10$) gives a 2.2 BLEU improvement in translation and a 3.5 ROUGE-2 improvement in summarization over greedy decoding.

Various approaches have been explored recently to improve beam search by improving the method by which candidate sequences are scored (Li et al., 2016; Shu and Nakayama, 2017), the termination criterion (Huang et al., 2017), or the search function itself (Li et al., 2017). In contrast, Gu et al. (2017) have tried to directly improve greedy decoding to decode for an arbitrary decoding objective. They add a small *actor* network to the decoder and train it with a version of policy gradient to optimize sequence objectives like BLEU. However, they report that they are seriously limited by the instability of this approach to training.

In this paper, we propose a procedure to modify a trained decoder to allow it to generate text greedily with the level of quality (according to metrics like BLEU) that would otherwise require the relatively expensive use of beam search. To do so, we follow Cho (2016) and Gu et al. (2017) in our use of an *actor* network which manipulates the decoder's hidden state, but introduce a stable and effective procedure to train this actor. In our

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 380–390
Brussels, Belgium, October 31 - November 4, 2018.

training procedure, the actor is trained with ordinary backpropagation on a model-specific artificial parallel corpus. This corpus is generated by running the un-augmented model on the training set with large-beam beam search, and selecting outputs from the resulting k-best list which score highly on our target metric.

Our method can be trained quickly and reliably, is effective, and can be straightforwardly employed with a variety of decoders. We demonstrate this for neural machine translation on three state-of-the-art architectures: RNN-based (Luong et al., 2015), ConvS2S (Gehring et al., 2017) and Transformer (Vaswani et al., 2017), and three corpora: IWSLT16 German-English,[1] WMT15 Finnish-English[2] and WMT14 German-English.[3]

2 Background

2.1 Neural Machine Translation

In sequence-to-sequence learning, we are given a set of source–target sentence pairs and tasked with learning to generate each target sentence (as a sequence of words or word-parts) from its source sentence. We first use an encoding model such as a recurrent neural network to transform a source sequence into an encoded representation, then generates the target sequence using a neural decoder.

Given a source sentence $\mathbf{x} = \{x_1, ..., x_{T_s}\}$, a neural machine translation system models the distribution over possible output sentences $\mathbf{y} = \{y_1, ..., y_T\}$ as:

$$P(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}) = \prod_{t=1}^{T} P(y_t|y_{<t}, \mathbf{x}; \boldsymbol{\theta}), \qquad (1)$$

where $\boldsymbol{\theta}$ is the set of model parameters.

Given a parallel corpus $D_{x,y}$ of source–target sentence pairs, the neural machine translation model can be trained by maximizing the log-likelihood:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{ \sum_{\langle \mathbf{x},\mathbf{y} \rangle \in D_{x,y}} \log P(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}) \right\}. \qquad (2)$$

2.2 Decoding

Given estimated model parameters $\hat{\boldsymbol{\theta}}$, the decision rule for finding the translation with the highest probability for a source sentence $\mathbf{x}$ is given by

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \left\{ P(\mathbf{y}|\mathbf{x}; \hat{\boldsymbol{\theta}}) \right\}. \qquad (3)$$

However, since such exact inference requires the intractable enumeration of large and potentially infinite set of candidate sequences, we resort to approximate decoding algorithms such as greedy decoding, beam search, noisy parallel decoding (NPAD; Cho, 2016), or trainable greedy decoding (Gu et al., 2017).

Greedy Decoding In this algorithm, we generate a single sequence from left to right, by choosing the token that is most likely at each step. The output $\hat{\mathbf{y}} = \{\hat{y}_1, ..., \hat{y}_T\}$ can be represented as

$$\hat{y}_t = \underset{y}{\operatorname{argmax}} \left\{ P(y|\hat{y}_{<t}, \mathbf{x}; \hat{\boldsymbol{\theta}}) \right\}. \qquad (4)$$

Despite its low computational complexity of $O(|V| \times T)$, the translations selected by this method may be far from optimal under the overall distribution given by the model.

Beam Search Beam search decodes from left to right, and maintains $k > 1$ hypotheses at each step. At each step t, beam search considers all possible next tokens conditioned on the current hypotheses, and picks the k with the overall highest scores $\prod_{t'=1}^{t} P(y_{t'}|y_{<t'}, \mathbf{x}; \hat{\boldsymbol{\theta}})$. When all the hypotheses are complete (they end in an end-of-the-sentence symbol or reach a predetermined length limit), it returns the hypothesis with the highest likelihood. Tuning to find a roughly optimal beam size k can yield improvements in performance with sizes as high as 30 (Koehn and Knowles, 2017; Britz et al., 2017). However, the complexity of beam search grows linearly in beam size, with high constant terms, making it undesirable in some applications where latency is important, such as in on-device real-time translation.

NPAD Noisy parallel approximate decoding (NPAD; Cho, 2016) is a parallel decoding algorithm that can be used to improve greedy decoding or beam search. The main idea is that a better translation with a higher probability may be found by injecting unstructured random noise into the hidden state of the decoder network. Positive results with NPAD suggest that small manipulations to the decoder hidden state can correspond to substantial but still reasonable changes to the output sequence.

---

[1] https://wit3.fbk.eu/

[2] http://www.statmt.org/wmt15/translation-task.html

[3] http://www.statmt.org/wmt14/translation-task

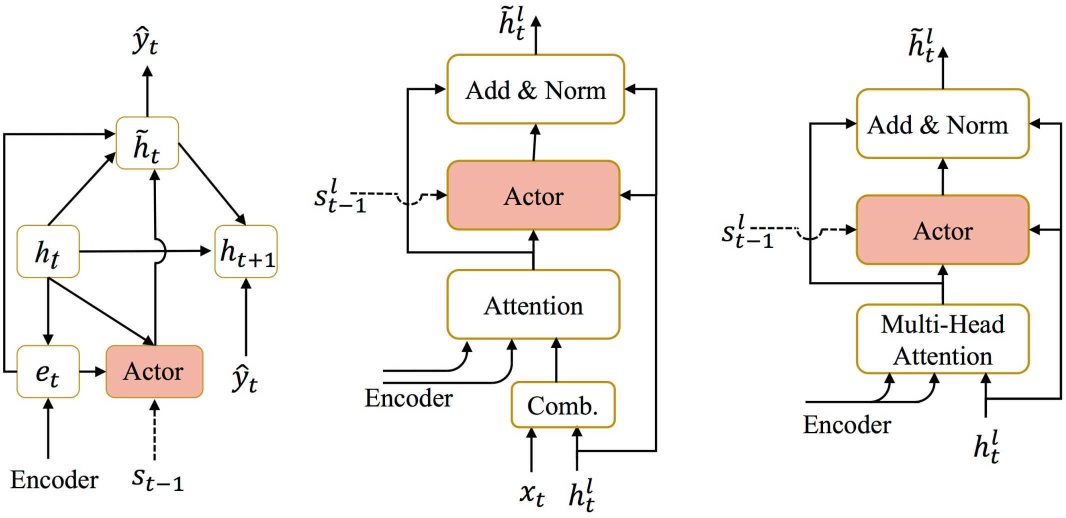

(a) RNN (b) ConvS2S (c) Transformer

Figure 1: A single step of a generic actor interacting with a decoder of each of three types. The dashed arrows denote an optional recurrent connection in the actor network.

Trainable Greedy Decoding Approximate decoding algorithms generally approximate the maximum-a-posteriori inference described in Equation 3. This is not necessarily the optimal basis on which to generate text, since (i) the conditional log-probability assigned by a trained NMT model does not necessarily correspond well to translation quality (Tu et al., 2017), and (ii) different application scenarios may demand different decoding objectives (Gu et al., 2017). To solve this, Gu et al. (2017) extend NPAD by replacing the unstructured noise with a small feedforward actor neural network. This network is trained using a variant of policy gradient reinforcement learning to optimize for a target quality metric like BLEU under greedy decoding, and is then used to guide greedy decoding at test time by modifying the decoder's hidden states. Despite showing gains over the equivalent actorless model, their attempt to directly optimize the quality metric makes training unstable, and makes the model nearly impossible to optimize fully. This paper offers a stable and effective alternative approach to training such an actor, and further develops the architecture of the actor network.

3 Methods

We propose a method for training a small actor neural network, following the trainable greedy decoding approach of Gu et al. (2017). This actor takes as input the current decoder state $\mathbf{h}_t$, an attentional context vector $\mathbf{e}_t$ for the source sentence, and optionally the previous hidden state $\mathbf{s}_{t-1}$ of the actor, and produces a vector-valued action $\mathbf{a}_t$ which is used to update the decoder hidden state. The actor function can take on a variety of forms, and we explore four: a feedforward network with one hidden layer (*ff*), feedforward network with two hidden layers (*ff2*), a GRU recurrent network (*rnn*; Cho et al., 2014), and gated feedforward network (*gate*).

The feedforward *ff* actor function is computed as

$$\begin{aligned} \mathbf{z}_t &= \sigma([\mathbf{h}_t, \mathbf{e}_t]W^i + \mathbf{b}^i), \\ \mathbf{a}_t &= \tanh(\mathbf{z}_t W^o + \mathbf{b}^o), \end{aligned} \tag{5}$$

the *ff2* actor is computed as

$$\begin{aligned} \mathbf{z}_t^1 &= \sigma([\mathbf{h}_t, \mathbf{e}_t]W^i + \mathbf{b}^i), \\ \mathbf{z}_t^2 &= \sigma(\mathbf{z}_t^1 W^z + \mathbf{b}^z), \\ \mathbf{a}_t &= \tanh(\mathbf{z}_t^2 W^o + \mathbf{b}^o), \end{aligned} \tag{6}$$

the *rnn* actor is computed as

$$\begin{aligned} \mathbf{z}_t &= \sigma([\mathbf{h}_t, \mathbf{e}_t]U^z + \mathbf{s}_{t-1}W^z), \\ \mathbf{r}_t &= \sigma([\mathbf{h}_t, \mathbf{e}_t]U^r + \mathbf{s}_{t-1}W^r), \\ \tilde{\mathbf{s}}_t &= \tanh\left([\mathbf{h}_t, \mathbf{e}_t]U^h + (\mathbf{s}_{t-1} \circ \mathbf{r}_t)W^h\right), \\ \mathbf{s}_t &= (1 - \mathbf{z}_t) \circ \tilde{\mathbf{s}}_t + \mathbf{z}_t \circ \mathbf{s}_{t-1}, \\ \mathbf{a}_t &= \mathbf{s}_t U, \end{aligned} \tag{7}$$

and the *gate* actor is computed as

$$\begin{aligned}\mathbf{z}_t &= \sigma([\mathbf{h}_t, \mathbf{e}_t]U^z),\\ \mathbf{a}_t &= \mathbf{z}_t \circ \tanh([\mathbf{h}_t, \mathbf{e}_t]U).\end{aligned} \tag{8}$$

Once the action $\mathbf{a}_t$ has been computed, the hidden state $\mathbf{h}_t$ is simply replaced with the updated state $\tilde{\mathbf{h}}_t$:

$$\tilde{\mathbf{h}}_t = f(\mathbf{h}_t, \mathbf{e}_t) + \mathbf{a}_t. \tag{9}$$

Figure 1 shows a single step of the actor interacting with the underlying neural decoder of each of the three NMT architectures we use: the RNN-based model of Luong et al. (2015), ConvS2S (Gehring et al., 2017), and Transformer (Vaswani et al., 2017). We add the actor at the decoder layer immediately after the computation of the attentional context vector. For the RNN-based NMT, we add the actor network only to the last decoder layer, the only place attention is used. Here, it takes as input the hidden state of the last decoder layer $\mathbf{h}_t^L$ and the source context vector $\mathbf{e}_t$, and outputs the action $\mathbf{a}_t$, which is added back to the attention vector $\tilde{\mathbf{h}}_t^L$. For ConvS2S and Transformer, we add an actor network to each decoder layer. This actor is added to the sublayer which performs multi-head or multi-step attention over the output of the encoder stack. It takes as input the decoder state $\mathbf{h}_t^l$ and the source context vector $\mathbf{e}_t^l$, and outputs an action $\mathbf{a}_t^l$ which is added back to get $\tilde{\mathbf{h}}_t^l$.

Training To overcome the severe instability reported by Gu et al. (2017), we introduce the use of a *pseudo-parallel corpus* generated from the underlying NMT model (Gao and He, 2013; Auli and Gao, 2014; Kim and Rush, 2016; Chen et al., 2017; Freitag et al., 2017; Zhang et al., 2017) for actor training. This corpus includes pairs that both (i) have a high model likelihood, so that we can coerce the model to generate them without much additional training or many new parameters and, (ii) represent high-quality translations, measured according to a target metric like BLEU. We do this by generating sentences from the original unaugmented model with large-beam beam search and selecting the best sentence from the resulting k-best list according to the decoding objective.

More specifically, let $\langle \mathbf{x}, \mathbf{y} \rangle$ be a sentence pair in the training data and $\mathcal{Z} = \{\mathbf{z}^1, ..., \mathbf{z}^k\}$ be the k-best list from beam search on the pretrained NMT model, where k is the beam size. We define the objective score of the translation $\mathbf{z}$ w.r.t. the gold-standard translation $\mathbf{y}$ according to a target metric such as BLEU (Papineni et al., 2002), NIST (Doddington, 2002), negative TER (Snover et al., 2006), or METEOR (Lavie and Denkowski, 2009) as $O(\mathbf{z}, \mathbf{y})$. Then we choose the sentence $\tilde{\mathbf{z}}$ that has the highest score to become our new target sentence:

$$\tilde{\mathbf{z}} = \underset{\mathbf{z}=\mathbf{z}^1,..,\mathbf{z}^k}{\operatorname{argmax}} O(\mathbf{z}, \mathbf{y}). \tag{10}$$

Once we obtain the pseudo-corpus $D_{x,z} = \{\langle \mathbf{x_i}, \tilde{\mathbf{z}}_i \rangle\}_{i=1}^n$, we keep the underlying model fixed and train the actor by maximizing the log-likelihood of the actor parameters with these pairs:

$$\hat{\boldsymbol{\theta}}_a = \underset{\boldsymbol{\theta}_a}{\operatorname{argmax}} \left\{ \sum_{\langle \mathbf{x}, \mathbf{z} \rangle \in D_{x,z}} \log P(\mathbf{z} | \mathbf{x}; \hat{\boldsymbol{\theta}}, \boldsymbol{\theta}_a) \right\} \tag{11}$$

In this way, the actor network is trained to manipulate the neural decoder's hidden state at decoding time to induce it to produce better-scoring outputs under greedy or small-beam decoding.

4 Experiments

4.1 Setting

We evaluate our approach on IWSLT16 German-English, WMT15 Finnish-English, and WMT14 De-En translation in both directions with three strong translation model architectures.

For IWSLT16, we use *tst2013* and *tst2014* for validation and testing, respectively. For WMT15, we use *newstest2013* and *newstest2015* for validation and testing, respectively. For WMT14, we use *newstest2013* and *newstest2014* for validation and testing, respectively. All the data are tokenized and segmented into subword symbols using byte-pair encoding (BPE; Sennrich et al., 2016) to restrict the size of the vocabulary. Our primary evaluations use tokenized and cased BLEU. For METEOR and TER evaluations, we use multeval[4] with tokenized and case-insensitive scoring. All the underlying models are trained from scratch, except for ConvS2S WMT14 English-German translation, for which we use the trained model (as well as training data) provided by Gehring et al. (2017).[5]

---

[4] https://github.com/jhclark/multeval

[5] https://s3.amazonaws.com/fairseq-py/models/wmt14.v2.en-de.fconv-py.tar.bz2

	BLEU↑			tok/s↑			BLEU↑			tok/s↑		
	greedy	beam4	tg	greedy	beam4	tg	greedy	beam4	tg	greedy	beam4	tg
IWSLT16	De → En						En → De					
RNN	23.57	24.90	23.59	62.8	45.0	60.4	20.05	21.11	19.88	48.1	32.5	45.7
ConvS2S	27.44	28.80	28.74	191.1	87.2	167.5	22.88	24.02	24.42	136.5	64.0	124.0
Transformer	27.15	28.74	28.36	63.9	31.0	59.8	23.87	25.03	25.46	57.9	26.5	51.2
WMT15	Fi → En						En → Fi					
RNN	12.45	13.22	13.02	51.5	33.1	43.4	9.77	10.81	10.57	44.0	31.2	43.8
ConvS2S	15.43	16.86	17.17	24.8	11.4	16.2	12.65	13.97	14.33	25.0	11.7	16.9
Transformer	13.76	14.61	14.49	31.4	13.4	29.8	12.38	13.55	12.95	29.8	12.8	27.9
WMT14	De → En						En → De					
RNN	23.08	24.62	24.54	38.4	26.6	36.4	18.87	20.59	19.89	33.2	22.4	32.5
ConvS2S	27.52	28.79	28.56	22.5	9.9	14.6	24.86	25.71	26.04	19.9	9.1	13.6
Transformer	26.44	27.31	26.96	32.9	14.3	30.9	22.01	22.74	22.31	28.5	12.2	26.1

Table 1: Generation quality (BLEU) and speed (tokens/sec). Speed is measured for sentence-by-sentence generation without mini-batching on the test set on CPU. We show the result by the underlying model with greedy decoding (greedy), beam search with $k = 4$ (beam4) and our trainable greedy decoder (tg).

	BLEU↑			
	tg	tg+beam4	tg	tg+beam4
IWSLT16	De → En		En → De	
RNN	23.59	25.03	19.88	20.72
ConvS2S	28.74	29.50	24.42	24.74
Transformer	28.36	28.95	25.46	25.89
WMT15	Fi → En		En → Fi	
RNN	13.02	13.49	10.57	11.04
ConvS2S	17.17	17.51	14.33	14.87
Transformer	14.49	14.79	12.95	13.45
WMT14	De → En		En → De	
RNN	24.54	24.86	19.89	20.56
ConvS2S	28.56	28.46	26.04	26.08
Transformer	26.96	27.21	22.31	21.92

Table 2: Generation quality (BLEU↑) using the proposed trainable greedy decoder without and with beam search ($k = 4$). Results without beam search (*tg*) are also appeared in Table 1.

RNN We use OpenNMT-py (Klein et al., 2017)[6] to implement our model. It is composed of an encoder with two-layer bidirectional RNN, and a decoder with another two-layer RNN. We refer to OpenNMT's default setting (*rnn_size* = 500, *word_vec_size* = 500) and the setting in Artetxe et al. (2018) (*rnn_size* = 600, *word_vec_size* = 300), and choose similar hyper-parameters: *rnn_size* = 500, *word_vec_size* = 300 for IWSLT16 and *rnn_size* = 600, *word_vec_size* = 500 for WMT. We use the input-feeding decoder and global attention with the general alignment function (Luong et al., 2015).

ConvS2S We implement our model based on fairseq-py.[7] We follow the settings in *fconv_iwslt_de_en* and *fconv_wmt_en_de* for IWSLT16 and WMT, respectively.

Transformer We implement our model based on the code from Gu et al. (2018).[8] We follow their hyperparameter settings for all experiments.

In the results below, we focus on the *gate* actor and pseudo-parallel corpora constructed by choosing the sentence with the best BLEU score from the k-best list produced by beam search with $k = 35$. Experiments motivating these choices are shown later in this section.

4.2 Results and Analysis

The results (Table 1) show that the use of the actor makes it practical to replace beam search with greedy decoding in most cases: We lose little or no performance, and doing so yields an increase in decoding efficiency, even accounting for the small overhead added by the actor. Among the three architectures, ConvS2S—the one with the most and largest layers—performs best. We conjecture that this gives the decoder more flexibility with which to guide decoding. In cases where model throughput is less important, our method can also be combined with beam search at test time to yield results somewhat better than either could achieve alone. Table 2 shows the result when combining our method with beam search.

---

[6] https://github.com/OpenNMT/OpenNMT-py

[7] https://github.com/facebookresearch/fairseq-py

[8] https://github.com/salesforce/nonauto-nmt

src	Am Vormittag wollte auch die Arbeitsgruppe Migration und Integration ihre Beratungen fortsetzen .
ref	During the morning , the Migration and Integration working group also sought to continue its discussions .
greedy	The morning also wanted to continue its discussions on migration and integration .
beam4	In the morning , the working group on migration and integration also wanted to continue its discussions .
beam35*	In the morning , the migration and integration working group also wanted to continue its discussions .
tg	The morning , the Migration and Integration Working Group wanted to continue its discussions .
tg+beam4	In the morning , the Migration and Integration Working Group wanted to continue its discussions .
src	Die meisten Mails werden unterwegs mehrfach von Software-Robotern gelesen .
ref	The majority of e-mails are read several times by software robots en route to the recipient .
greedy	Most mails are read by software robots on the go .
beam4	Most mails are read by software robots on the go .
beam35*	Most e-mails are read several times by software robots on the road .
tg	Most mails are read several times by software robots on the road .
tg+beam4	Most mails are read several times by software robots on the road .
src	Ich suche schon seit einiger Zeit eine neue Wohnung fr meinen Mann und mich .
ref	I have been looking for a new home for my husband and myself for some time now .
greedy	I have been looking for a new apartment for some time for my husband and myself .
beam4	I have been looking for a new apartment for some time for my husband and myself .
beam35*	I have been looking for a new apartment for my husband and myself for some time now .
tg	I have been looking for a new apartment for my husband and myself for some time now .
tg+beam4	I have been looking for a new apartment for my husband and myself for some time now .

Table 3: Translation examples from the WMT14 De-En test set with Transformer. We show translations generated by the underlying transformer using greedy decoding, beam search with $k = 4$, and beam search with $k = 35$ and the oracle BLEU scorer ($\star$). We also show the translations using our trainable greedy decoder both without and with beam search. Phrases of interest are underlined.

	IWSLT16 De-En				WMT14 De-En			
	ref	greedy	k35*	tg	ref	greedy	k35*	tg
Base Model	20.4	65.3	61.5	64.2	23.5	65.2	63.8	65.1
+Trainable Greedy Decoder	19.1	70.4	65.3	75.1	18.9	76.0	72.6	82.8

Table 4: Word-level likelihood (%) averaged by sentence for the IWSLT16 and WMT14 De-En test sets with Transformer. Each row represents the model used to evaluate word-level likelihood, and each column represents a different source of translations, including the reference (ref), greedy decoding on the base model (greedy), beam search with $k = 35$ on the base model and the BLEU scorer (k35*), and trainable greedy decoder (tg).

Examples Table 3 shows a few selected translations from the WMT14 German-English test set. In manual inspection of these examples and others, we find that the actor encourages models to recover missing tokens, optimize word order, and correct prepositions.

Likelihood We also compare word-level likelihood for different decoding results assigned by the base model and the actor-augmented model. For a sentence pair $\langle \mathbf{x}, \mathbf{y} \rangle$, word-level likelihood is defined as

$$P_w = \frac{1}{T} \sum_{t=1}^{T} P(y_t | y_{<t}, \mathbf{x}; \boldsymbol{\theta}). \qquad (12)$$

Table 4 shows the word-level likelihood averaged over the test set for IWSLT16 and WMT14 German to English translation with Transformer. Our trainable greedy decoder learns a much more peaked distribution and assigns a much higher probability mass to its greedy decoding result than the base model. When evaluated under the base model, the translations from trainable greedy decoding have smaller likelihood than the translations from greedy decoding using the base model for both datasets. This indicates that the trainable greedy decoder is able to find a sequence that is not highly scored by the underlying model, but that corresponds to a high value of the target metric.

Magnitude of Action Vector We also record the L_2 norm of the action, decoder hidden state, and attentional source context vectors on the validation set. Figure 2 shows these values over the course of training on the IWSLT16 De-En validation set with Transformer. The norm of the action starts

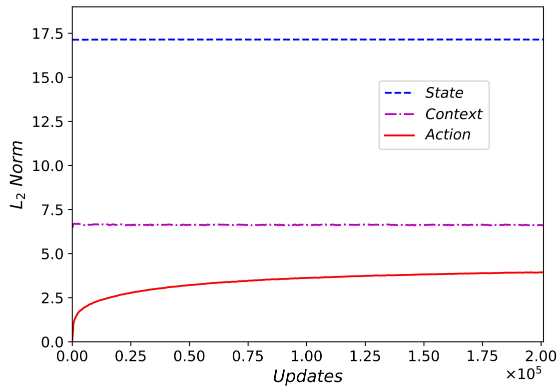

Figure 2: The norms of the three activation vectors on the IWSLT16 De-En validation set with Transformer. *Action*, *Context* and *State* represent the norm of the action, attentional source context vector and decoder hidden state, respectively.

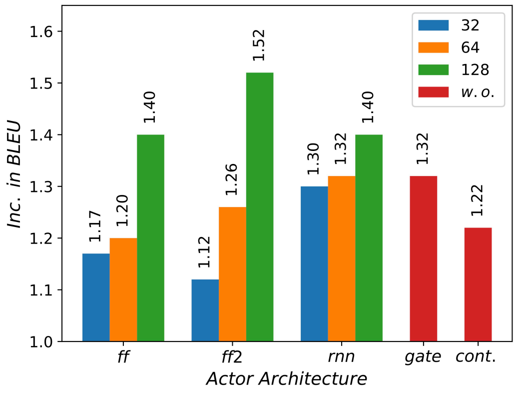

Figure 3: The effect of the actor architecture and hidden state size on trainable greedy decoding results over the IWSLT16 De-En validation set with Transformer (BLEU↑), shown with a baseline (*cont.*) in which the underlying model, rather than the actor, is trained on the pseudo-parallel corpus. The Y-axis starts from 1.0. *w.o.* indicates an actor with no hidden layer. 0.0 corresponds to 33.04 BLEU.

small, increases rapidly early in training, and converges to a value well below that of the decoder hidden state. This suggests that the action adjusts the decoders hidden state only slightly, rather than overwriting it.

4.3 Effects of Model Settings

Actor Architecture Figure 3 shows the trainable greedy decoding result on IWSLT16 De-En validation set with different actor architectures. We observe that our approach is stable across different actor architectures and is relatively insensitive to the hyperparameters of the actor. For the same type of actor, the performance increases gradually with the hidden layer size. The use of a recurrent connection within the actor does not meaningfully improve performance, possibly since all actors can use the recurrent connections of the underlying decoder. Since the gate actor contains no additional hyperparameters and was observed to learn quickly and reliably, we use it in all other experiments.

Here, we also explore a simple alternative to the use of the actor: creating a pseudo-parallel corpus with each model, and then training each model, unmodified and entirety, directly on this new corpus. This experiment (*cont.* in Figure 3) yields results that are comparable to, but not better than, the results seen with the actors. However, this comes with substantially greater computational complexity at training time, and, if the same trained model is to be optimized for multiple target metrics, greater storage costs as well.

Beam Size Figure 4a shows the effect of the beam size used to generate the pseudo-parallel corpus on the IWSLT16 De-En validation set with Transformer. Trainable greedy decoding improves over greedy decoding even when we set $k = 1$, namely, running *greedy* decoding on the unaugmented model to construct the new training corpus. With increased beam size k, the BLEU score consistently increases, but we observe diminishing returns beyond roughly $k = 35$, and we use that value elsewhere.

Training Corpus Construction There are a variety of ways one might use the output of beam search to construct a pseudo-parallel corpus: We could use the single highest-scoring output (by BLEU, or our target metric) for each input (*top1*), use all 35 beam search outputs (*full*), use all those outputs that score higher than the threshold, namely the base model's greedy decoding output (*thd*), or combine the *top1* results with the gold-standard translations (*comb.*). We show the effect of training corpus construction in Figure 4b. *para* denotes the baseline approach of training the actor with the original parallel corpus used to train the underlying NMT model. Among the four novel approaches, *full* obtains the worst performance, since the beam search outputs contain translations that are far from the gold-standard translation. We choose the best performing *top1* strategy.

	Obj.	BLEU↑	MTR↑	TER↓
Greedy	-	27.15	29.0	54.4
Beam4	-	28.74	29.9	51.9
	sBLEU	28.36	29.7	52.0
	sMTR	28.36	29.6	51.8
	-sTER	28.05	29.6	51.4

Table 5: Results when trained with different decoding objectives on IWSLT16 De-En translation using Transformer. MTR denotes METEOR. We report greedy decoding and beam search ($k = 4$) results using the original model, and results with trainable greedy decoding (lower half).

Decoding Objectives As our approach is capable of using an arbitrary decoding objective, we investigate the effect of different objectives on BLEU, METEOR (MTR) and TER scores with Transformer for IWSLT16 De-En translation. Table 5 shows the final result on the test set. When trained with one objective, our model yields relatively good performance on that objective. For example, negative sentence-level TER (i.e., -sTER) leads to -3.0 TER improvement over greedy decoding and -0.5 TER improvement over beam search. However, since these objectives are all well correlated with each other, training with different objectives do not differ dramatically.

5 Related Work

Data Distillation Our work is directly inspired by work on knowledge distillation, which uses a similar pseudo-parallel corpus strategy, but aims at training a compact model to approximate the function learned by a larger model or an ensemble of models (Hinton et al., 2015). Kim and Rush (2016) introduce knowledge distillation in the context of NMT, and show that a smaller student network can be trained to achieve similar performance to a teacher model by learning from pseudo-corpus generated by the teacher model. Zhang et al. (2017) propose a new strategy to generate a pseudo-corpus, namely, fast sequence-interpolation based on the greedy output of the teacher model and the parallel corpus. Freitag et al. (2017) extend knowledge distillation on an ensemble and oracle BLEU teacher model. However, all these approaches require the expensive procedure of retraining the full student network.

Pseudo-Parallel Corpora in Statistical MT Pseudo-parallel corpora generated from beam search have been previously used in statistical machine translation (SMT) (Chiang, 2012; Gao and He, 2013; Auli and Gao, 2014; Dakwale and Monz, 2016). Gao and He (2013) integrate a recurrent neural network language model as an additional feature into a trained phrase-based SMT system and train it by maximizing the expected BLEU on k-best list from the underlying model. Our work revisits a similar idea in the context trainable greedy decoding for neural MT.

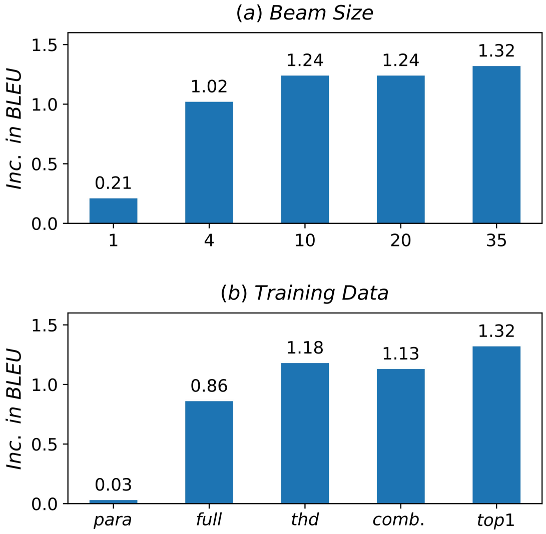

Figure 4: (a) The effect of beam size on the IWSLT16 De-En validation with Transformer and (b) the effect of the training corpus composition in the same setting. *para*: parallel corpus; *full*: all 35 beam search outputs; *thd*: beam search outputs that score higher than the base model's greedy decoding output; *top1*: beam search output with the highest bleu score; *comb.*: *top1+para*. 0.0 corresponds to 33.04 BLEU.

Decoding for Multiple Objectives Several works have proposed to incorporate different decoding objectives into training. Ranzato et al. (2015) and Bahdanau et al. (2016) use reinforcement learning to achieve this goal. Shen et al. (2016) and Norouzi et al. (2016) train the model by defining an objective-dependent loss function. Wiseman and Rush (2016) propose a learning algorithm tailored for beam search. Unlike these works that optimize the entire model, Li et al. (2017) introduce an additional network that predicts an arbitrary decoding objective given a source sentence and a prefix of translation. This prediction is used as an auxiliary score in beam search. All of these methods focus primarily on improving beam search results, rather than those with greedy decoding.

6 Conclusion

This paper introduces a novel method, based on an automatically-generated pseudo-parallel corpus, for training an actor-augmented decoder to optimize for greedy decoding. Experiments on three models and three datasets show that the training strategy makes it possible to substantially improve the performance of an arbitrary neural sequence decoder on any reasonable translation metric in either greedy or beam-search decoding, all with only a few trained parameters and minimal additional training time.

As our model is agnostic to both the model architecture and the target metric, we see the exploration of more diverse and ambitious model–target metric pairs as a clear avenue for future work.

Acknowledgments

This work was partly supported by Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI), Samsung Electronics (Improving Deep Learning using Latent Structure) and the Facebook Low Resource Neural Machine Translation Award. KC thanks support by eBay, TenCent, NVIDIA and CIFAR. This project has also benefited from financial support to SB by Google and Tencent Holdings.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Michael Auli and Jianfeng Gao. 2014. Decoder integration and expected bleu training for recurrent neural network language models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 136–142, Baltimore, Maryland. Association for Computational Linguistics.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Joseph Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In Proceedings of International Conference on Learning Representations (ICLR)*.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2013. Audio chord recognition with recurrent neural networks. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, Copenhagen, Denmark. Association for Computational Linguistics.

Yun Chen, Yang Liu, Yong Cheng, and Victor O.K. Li. 2017. A teacher-student framework for zero-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1925–1935. Association for Computational Linguistics.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13(Apr):1159–1187.

Kyunghyun Cho. 2016. Noisy parallel approximate decoding for conditional recurrent language model. *preprint arXiv:1605.03835*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Praveen Dakwale and Christof Monz. 2016. Improving statistical machine translation performance by oracle-bleu model re-estimation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 38–44.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *preprint arXiv:1807.03819*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. *preprint arXiv:702.01802*.

Jianfeng Gao and Xiaodong He. 2013. Training mrf-based phrase translation models using gradient ascent. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, pages 450–459, Atlanta, Georgia. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *preprint arXiv:211.3711*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. *In Proceedings of International Conference on Learning Representations (ICLR)*.

Jiatao Gu, Kyunghyun Cho, and Victor O.K. Li. 2017. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978, Copenhagen, Denmark. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *preprint arXiv:1503.02531*.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2134–2139.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Alon Lavie and Michael J Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine translation*, 23(2-3):105–115.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *preprint arXiv:1611.08562*.

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2017. Learning to decode for future success. *preprint arXiv:1701.06549*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Mohammad Norouzi, Samy Bengio, zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1723–1731. Curran Associates, Inc.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *preprint arXiv:1705.04304*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *preprint arXiv:1511.06732*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

Raphael Shu and Hideki Nakayama. 2017. Later-stage minimum bayes-risk decoding for neural machine translation. *preprint arXiv:1704.03169*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3156–3164. IEEE.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Xiaowei Zhang, Wei Chen, Feng Wang, Shuang Xu, and Bo Xu. 2017. Towards compact and fast neural machine translation using a combined method. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1475–1481, Copenhagen, Denmark. Association for Computational Linguistics.

Addressing Troublesome Words in Neural Machine Translation

Yang Zhao[1,2], **Jiajun Zhang**[1,2], **Zhongjun He**[4], **Chengqing Zong**[1,2,3], **and Hua Wu**[4]
[1]National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
[4]Baidu Inc., Beijing, China
{yang.zhao, jjzhang, cqzong}@nlpr.ia.ac.cn, {hezhongjun,wu_hua} @baidu.com

Abstract

One of the weaknesses of Neural Machine Translation (NMT) is in handling low-frequency and ambiguous words, which we refer as troublesome words. To address this problem, we propose a novel memory-enhanced NMT method. First, we investigate different strategies to define and detect the troublesome words. Then, a contextual memory is constructed to memorize which target words should be produced in what situations. Finally, we design a hybrid model to dynamically access the contextual memory so as to correctly translate the troublesome words. The extensive experiments on Chinese-to-English and English-to-German translation tasks demonstrate that our method significantly outperforms the strong baseline models in translation quality, especially in handling troublesome words.

1 Introduction

Neural machine translation (NMT) based on the encoder-decoder architecture becomes the new state-of-the-art due to distributed representation and end-to-end learning (Cho et al., 2014; Bahdanau et al., 2015; Junczys-Dowmunt et al., 2016; Gehring et al., 2017; Vaswani et al., 2017).

However, the current NMT is a global model that maximizes the performance on the overall data and has problems in handling low-frequency words and ambiguous words[1], we refer these words as troublesome words and define them in Section 3.1.

Some previous work attempt to tackle the translation problem of low-frequency words. Sennrich et al. (2016) propose to decompose the words into subwords which are used as translation units so that the low-frequency words can be represented by frequent subword sequences. Arthur et al. (2016) and Feng et al. (2017) try to incorporate a translation lexicon into NMT in order to obtain the correct translation of low-frequency words. However, the former method still faces the low-frequency problem of subwords. And the latter one has a drawback that they use lexicons without considering specific contexts. Fig. 1 shows an example, in which "aerkate" is an infrequent word and the baseline NMT incorrectly translates it into a pronoun "he". Incorporation of bilingual lexicon rectifies the mistake but wrongly converts "chengzhang" into an incorrect target word "growth" since an entry "(chengzhang, growth)" in the bilingual lexicon is somewhat wrongly used without taking the contexts into account. Furthermore, these two kinds of methods mainly focus on low-frequency words that are just a part of the troublesome words.

In this paper, we categorize the words (including infrequent words and ambiguous words) which are difficult to translate as troublesome words and propose a novel memory-augmented framework to

Source: 阿尔卡特 宣称 去年 第四 季 销售 成长 近 百分之三十

Pinyin: **aerkate** cheng qunian disi ji xiaoshou **chengzhang** jin baifenzhisanshi

Reference: **alcatel** says sales in fourth quarter last year **grew** nearly 30 %

NMT: **he** said sales **grew** nearly 30 percent in fourth quarter of last year

NMT+LexiconTable: **alcatel** said sales **growth** nearly 30 percent in fourth quarter of last year

Figure 1: The NMT model produces a wrong translation for the low-frequency word "aerkat". While introducing an external lexicon table without contextual information, the model incorrectly translates the ambiguous word "chengzhang" into "growth".

---

[1] In this work, we consider a source word is ambiguous if it has multiple translations with high entropy of probability distribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 391–400
Brussels, Belgium, October 31 - November 4, 2018.

address them. Our method first investigates different strategies to define the troublesome words. Then, these words and their contexts in the training data are memorized with a contextual memory which is finally accessed dynamically during decoding to solve the translation problem of the troublesome words.

Specifically, we first decode all the source sentences of the bilingual training data with baseline NMT and define the troublesome source words according to the distance between the predicted words and the gold words. The troublesome words associated with their hidden contextual representations are stored in a memory which memorizes the correct translation and the corresponding contextual information. During decoding, we activate the contextual memory when we encounter the troublesome words and employ the contextual similarity between the test sentence and the memory to determine appropriate target words. We test our methods on Chinese-to-English and English-to-German translation tasks. The experimental results demonstrate that the translation performance can be significantly improved and a large portion of troublesome words can be correctly translated. The contributions are listed as follows:

1) We are the first to define and handle the troublesome words in neural machine translation.

2) We propose to memorize not only the bilingual lexicons but also their contexts with a contextual memory.

3) We design a dynamic approach to correctly translate the troublesome words by combining the contextual memory and the NMT model.

2 Neural Machine Translation

NMT contains an encoder and a decoder. The encoder transforms a source sentence $X = \{x_1, x_2, ..., x_{Tx}\}$ into a set of context vectors $C = (h_1^m, h_2^m, ..., h_{Tx}^m)$ by using m stacked Long Short Term Memory (LSTM) layers (Hochreiter and Schmidhuber, 1997) . h_j^m is the hidden state of the top layer in encoder. The bottom layer of encoder is a bi-direction LSTM layer to collect the context from the left side and right side.

The decoder generates one target word at a time by computing $p_i^N(y_i|y_{<i}, C)$ as follows:

$$p_i^N(y_i|y_{<i}, C) = softmax(W_{y_i}\widetilde{z}_i + b_s) \quad (1)$$

where $\widetilde{z}_i$ is the attention output:

$$\widetilde{z}_i = tanh(W_z[z_i^m; c_i]) \quad (2)$$

c_i can be calculated as follows:

$$c_i = \sum_{j=1}^{Tx} a_{ij} h_i^m \quad (3)$$

where $a_{i,j}$ is the attention weight:

$$a_{i,j} = \frac{h_j^m z_i^m}{\sum_j h_j^m z_i^m} \quad (4)$$

where z_i^m is the hidden state of the top layer in decoder. More detailed introduction can be found in (Luong et al., 2015).

Notation. In this paper, we denote the whole source vocabulary by $V_S = \{s_m\}_{m=1}^{|V_S|}$ and target vocabulary by $V_T = \{t_n\}_{n=1}^{|V_T|}$, where s_m is the source word and t_n is the target word. We denote a source sentence by X and a target sentence by Y. Each source word in X is denoted by x_j. Each target word in Y is denoted by y_i. Accordingly, a target word can be denoted not only by t_n, but also by y_i. This does not contradict. t_n means this target word is the n^{th} word in vocabulary V_T, and y_i means this target word is the i^{th} word in sentence Y. Similarly, we denote a source word by s_m and x_j.

3 Method Description

Our method contains three parts: 1) definition and detection of the troublesome words (Section 3.1); 2) contextual memory construction (Section 3.2); and 3) hybrid approach combining contextual memory and baseline NMT model (Section 3.3).

3.1 Troublesome Word Definition

Generally speaking, troublesome words are those that are difficult to translate for the baseline NMT system $BNMT$. Fig. 2 shows the main process to detect the troublesome words. Given each training sentence pair (X, Y), $BNMT$ decodes the source sentence X and outputs the predicted probability of each gold target word $p_i^N(y_i)$. We call y_i an **exception** if $p_i^N(y_i)$ satisfies the predefined **exception criteria** introduced below. The source word x_j is an exception (a candidate troublesome word) if (x_j, y_i) is an entry in the word alignment A[2]. Suppose x_j appears N times in the training data and there are M exceptions among all its aligned

---

[2] The word alignments A is extracted using the fast-align tool (Dyer et al., 2013) on the bilingual training data with both source-to-target and target-to-source directions.

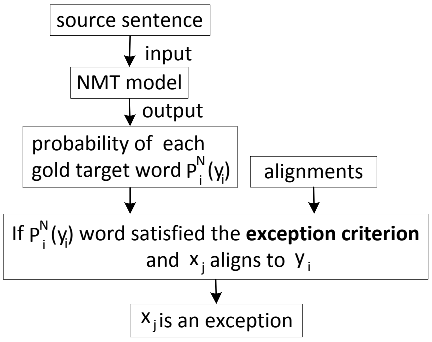

Figure 2: The main process to detect an exception.

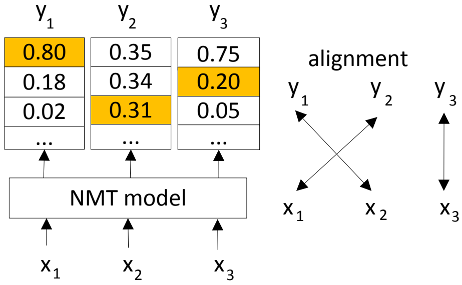

Figure 3: A toy example to show the process: if $p_i^N(y_i)$ (left) satisfies the predefined exception criteria and x_j aligns to y_i, then x_j is an exception.

gold target words. Then, the **exception rate** $r(x_j)$ will be M/N.

Definition: x_j is a **troublesome word** if $r(x_j) > \epsilon$ in which ϵ is a predefined threshold.

Exception Criteria. As discussed before, we need an exception criterion to measure whether a gold target word is an exception or not. In this paper, we investigate three exception criteria. Here, we introduce each of them through a toy example shown in Fig. 3, in which the source sentence is $X = \{x_1, x_2, x_3\}$ and the gold target sentence is $Y = \{y_1, y_2, y_3\}$. The left shows the probability distribution of all target vocabulary $p_i^N(V_T)$ at each decoding step i, where the probability of the gold target word is highlighted in yellow. The right shows the word alignments between X and Y.

1) Absolute Criterion. A gold target word y_i is an exception if its predicted probability $p_i^N(y_i)$ is lower than a predefined threshold, namely $p_i^N(y_i) < p_0$. In Fig. 3, $p_i^N(y_i)$ at each decoding step is respectively 0.8, 0.31 and 0.2. If we set $p_0 = 0.5$, $p_2^N(y_2)$ and $p_3^N(y_3)$ are lower than threshold p_0. x_1 and x_3 are both exceptions according to the alignments.

2) Gap Criterion. For this criterion, we utilize the predicted probability gap between the gold target word and the top one. Specifically, the gap can be calculated by:

$$g(y_i) = max(p_i^N(V_T)) - p_i^N(y_i) \qquad (5)$$

where $max(p_i^N(V_T))$ is the top one in the probability distribution at the i^{th} decoding step. y_i is an exception if $g(y_i) > g_0$. In Fig. 3, the largest predicted probabilities at each decoding step $max(p_i^N(V_T))$ are respectively 0.8, 0.35 and 0.75. Thus, the gap is 0.0, 0.04 and 0.55. If $g_0 = 0.1$, x_3 is an exception since $g(y_3) > g_0$ and x_3 aligns to y_3.

3) Ranking Criterion. This criterion is based on the ranking of $p_i^N(y_i)$ in $p_i^N(V_T)$ (denoted by $rank(y_i)$). If $rank(y_i) > rank_0$, then y_i is an exception. In Fig. 3, the ranking of each gold target word is 1, 3 and 2. If we set $rank_0 = 2$, then $rank(y_2) = 3 > rank_0$ and x_1 is an exception due to the alignment between x_1 and y_2.

Using the above exception criteria and the definition of troublesome words, we can detect all the source-side troublesome words in the bilingual training data.

3.2 Contextual Memory Construction

For a troublesome word, we now introduce how to build a contextual memory $\mathbb{M}$ to store its translation knowledge. Specifically, the contextual memory contains five elements:

$$\{s_m, t_n, c(s_m, t_n), p^L(s_m, t_n), r(s_m)\} \qquad (6)$$

each of them is described as follows:

- s_m is a troublesome source word.
- t_n is a gold target word for s_m.
- $c(s_m, t_n)$ is the context of lexicon pair (s_m, t_n). Here, we use the hidden states of encoder h_j to represent the context, since it contains the information from left ($\overrightarrow{h_j}$) and right ($\overleftarrow{h_j}$). Note that when we traverse the training data and memorize the contexts of all troublesome words, there must be many cases in which the same pair (s_m, t_n) appears in different contexts. In order to reduce the memory size and fuse different contexts of a same lexicon pair, we merge these memories by averaging the contexts. Assume there are K different contexts for (s_m, t_n), and they

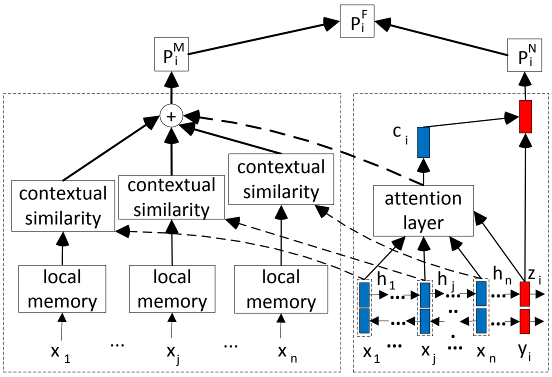

Figure 4: The architecture of contextual memory-augmented NMT.

are denoted by $h_k(s_m, t_n)$. The average context of (s_m, t_n) can be calculated by:

$$c(s_m, t_n) = \frac{\sum_{k=1}^{K} h_k(s_m, t_n)}{K} \quad (7)$$

Note that the context here is defined on the source side.

- $p^L(s_m, t_n)$ is the lexicon translation probability. It is the average of source-to-target and target-to-source probabilities calculated through maximum likelihood estimation on word alignments.
- $r(s_m)$ is the exception rate of s_m introduced in Section 3.1 and it can indicate the translation difficulty of a source word. We will use $r(s_m)$ to determine the dynamic weights of contextual memories in Section 4.

Noise Reduction. As we know, the training data and word alignments are not perfect and may introduce noise to the contextual memory. To reduce the noise, we employ two strategies.

1) To improve the quality of the alignments A, we derive the alignment results from source-to-target and target-to-source, respectively. We only save the alignments which exist in both directions.

2) We eliminate the lexicon pairs whose translation probabilities are too small. For a lexicon pair (s_m, t_n), if its lexicon translation probability is smaller than 0.01, we treat this lexicon pair as a noisy sample and eliminate it from our memory.

3.3 Integrating Contextual Memory into NMT

In this section, we integrate the contextual memory into NMT to handle troublesome words. The overall framework is depicted in Fig. 4 and the integration process can be divided into four steps:

Step 1. Given a test sentence X, the first step is to find the troublesome words in X and collect corresponding **local memories** from the **global contextual memory** $\mathbb{M}$. For each source word x_j, we retrieve from $\mathbb{M}$ if it is a troublesome word and obtain the local memory as follows:

$$\{x_j, t_n, c(x_j, t_n), p^L(x_j, t_n), r(x_j)\} \quad (8)$$

Step 2. The next step is to measure the contextual similarity between the context in the test sentence X and the context in $\mathbb{M}$. For the troublesome word $x_j \in X$, we still use the encoder hidden state h_j to represent the context in X. The corresponding context in $\mathbb{M}$ is $c(x_j, t_n)$ in Eq. (8). Here, we use a feed-forward network to measure this similarity[3]:

$$d_j(t_n) = \\ sigmoid(v_d^T * tanh(W_h * h_j + W_c * c(x_j, t_n))) \quad (9)$$

where v_d, W_h and W_c are learnable parameters. The sigmoid function guarantees the similarity score is in the range $(0, 1)$. This similarity $d_j(t_n)$ will determine whether or not to adopt the target translation word t_n in $\mathbb{M}$.

Step 3. The next task is calculating the probability $p_i^M(t_n)$ of t_n at each decoding step i. $p_i^M(t_n)$ is the probability predicted by the contextual memory $\mathbb{M}$ and is calculated by:

$$p_i^M(t_n) = \sum_{j}^{Tx} a_{i,j} * d_j(t_n) * p^L(x_j, t_n) \quad (10)$$

where $a_{i,j}$ is the attention weight, $d_j(t_n)$ is the context similarity in Eq. (9), and $p^L(x_j, t_n)$ is the lexicon translation probability.

Step 4. The final task is to combine the memory predicted probability (p_i^M in Eq. (10)) and the NMT predicted one (p_i^N in Eq. (1)). Here, we propose a dynamic strategy to balance these two probabilities:

$$p_i^F(t_n) = \lambda_i * p_i^M(t_n) + (1 - \lambda_i) * p_i^N(t_n) \quad (11)$$

where $p_i^F(t_n)$ is the final probability of the target word t_n, λ_i is the dynamic weight to adjust

---

[3] In our preliminary experiment, we also try cosine distance to measure this similarity, while the performance of cosine distance is lower than the current feed-forward network method.

the contribution from the memory and NMT. Here we explain the reason why we apply the dynamic manner. Recall that for each source troublesome word s_m, we calculate its exception rate (similar to error rate). If a troublesome word has a lower exception rate, indicating that this source word is easier to be translated for the neural model. In this case, p_i^N is more reliable. Thus we design the dynamic weight λ_i according to the exception rate $r(x_j)$:

$$\begin{aligned} \lambda_i &= sigmoid(\beta_\gamma * \gamma_i) \\ \gamma_i &= \sum_{j}^{Tx} a_{i,j} * r(x_j) \end{aligned} \tag{12}$$

where β_γ is a learnable parameter. From Eq. (12), the dynamic weight λ_i is determined by both of the attention weight $a_{i,j}$, and the exception rate $r(x_j)$.

Training the parameters. As discussed above, our method contains some parameters (v_d, W_h, W_c and β_γ) to be learned. We denote the parameters introduced by our method by θ^M and the parameters in NMT by θ^N. To make it efficient, given the aligned training data $D = \{X^{(d)}, Y^{(d)}\}_{d=1}^{|D|}$, we keep θ^N unchanged and optimize θ^M by maximizing the following objective function.

$$L(\theta^M) = \frac{1}{|D|} \sum_{d=1}^{|D|} \sum_{i}^{T_y} log\ p_i^F(y_i^{(d)}; \theta^M) \tag{13}$$

where p_i^F can be calculated by Eq. (11).

4 Experimental Settings

We test the proposed methods on Chinese-to-English (CH-EN) and English-to-German (EN-DE) translation. In CH-EN translation, we use LDC corpus which includes 2.1M sentence pairs for training. NIST 2003 dataset is used for validation. NIST04-06 and 08 datasets are used for testing. In EN-DE translation, we use WMT 2014 EN-DE dataset, which includes 4.5M sentence pairs for training. 2012-2013 datasets are used for validation and 2014 dataset is used for testing.

We use the Zoph_RNN toolkit[4] to implement all described methods. In all experiments, the encoder and decoder include two stacked LSTM layers. The word embedding dimension and the size of hidden layers are both set to 1,000. The mini-batch size is set to 128. We discard the training sentence pairs whose length exceeds 100. We run a total of 20 iterations for all translation tasks. We test all methods based on two granularities: words and sub-words. For word granularity, we limit the vocabulary to 30K (CH-EN) and 50K (EN-DE) for both the source and target languages. For sub-word granularity, we use the BPE method (Sennrich et al., 2016) to merge 30K (CH-EN) and 32K (EN-DE) steps. The beam size is set to 12. We use case-insensitive 4-gram BLEU (Papineni et al., 2002) for translation quality evaluation.

We compare our method with other relevant methods as follows:

1) **Baseline**: It is the baseline NMT system with global attention (Luong et al., 2015; Zoph and Knight, 2016; Jean et al., 2015).

2) **Arthur**: It is the state-of-the-art method which incorporates discrete translation lexicons into NMT (Arthur et al., 2016). We implement Arthur et al. (2016)'s method in two different ways. In the first way, we fix the **Baseline** unchanged, and utilize Arthur et al. (2016)'s method in the test phase. We denote this system by **Arthur(test)**. In second way, we allow **Baseline** to be retrained by Arthur et al. (2016)'s method, and denote the system by **Arthur(train+test)**. We replicate the Arthurs work using the bias method with the hyper parameter being set to 0.001 as reported in their paper.

3) **X+MEM**: It is our proposed memory augment method for any neural model **X**, in which we define the troublesome word by using the gap criterion with threshold $g_0 = 0.1$. We set threshold $\epsilon = 0.05$, which is fine-tuned in validation set. It means if the exception rate of a source word exceeds 0.05, we treat this word as a troublesome word.

5 Results on CH-EN Translation

5.1 Our methods vs. Baseline

Table 1 reports the main translation results of CH-EN translation. We first compare **Baseline+MEM** with **Baseline**. As shown in row 1 and row 5 in Table 1, **Baseline+MEM** can improve over **Baseline** on all test datasets, and the average improvement is 1.37 BLEU points. The results show that our method could significantly outperform the baseline model.

---

[4] `https://github.com/isi-nlp/Zoph_RNN`. We extend this toolkit with global attention.

#	Model	03	04	05	06	08	Avg.	△
1	Baseline	41.01	42.94	40.31	40.57	30.96	39.16	-
2	Arthur(test)	41.34	43.31	40.79	40.84	31.11	39.48	-
3	Arthur(train+test)	41.88	43.75	41.16	41.63	31.47	39.98	-
4	Baseline(sub-word)	43.93	44.74	42.46	43.01	32.53	41.33	-
5	Baseline+MEM	$42.74^{\dagger}$	$43.94^{\dagger}$	$42.15^{\dagger}$	$41.94^{\dagger}$	$31.86^{\dagger}$	40.53	+1.37
6	Arthur(train+test)+MEM	$43.04^{\dagger}$	44.65^{*}	$42.19^{\dagger}$	$42.59^{\dagger}$	32.05^{*}	40.90	+0.92
7	Baseline(sub-word)+MEM	$44.98^{\dagger}$	$45.51^{\dagger}$	$43.93^{\dagger}$	$43.95^{\dagger}$	$33.33^{\dagger}$	42.34	+1.01

Table 1: The main results of CH-EN translation. △ shows the BLEU points improvement of system "X+MEM" than system X. "*" indicates that system "X+MEM" is statistically significant better ($p < 0.05$) than system X and "†" indicates $p < 0.01$.

Model	#Pairs	Size	Time	BLEU
Baseline	-	-	0.406	39.16
Arthur(test)	938K	58M	0.429	39.48
Tword	125K	7.4M	0.423	39.77
+Context	125K	893M	0.508	40.19
+Dynamic	125K	893M	0.511	40.53
All+Context	938K	6.4G	1.829	40.23

Table 2: The effects of lexicon pairs only containing troublesome words (Tword), context and dynamic weights. Column **#Pairs** shows the number of lexicon pairs. Column **Time** shows the average decoding time (seconds) of per sentence.

Source: 阿尔卡特 宣称 去年 第四 季 销售 成长 近 百分之三十

Pinyin: aerkate cheng qunian disi ji xiaoshou chengzhang jin baifenzhisanshi

Reference: alcatel says sales in fourth quarter last year grew nearly 30 %

Baseline: he said sales grew nearly 30 percent in fourth quarter of last year

Arthur: alcatel says sales growth nearly 30 percent in fourth quarter of last year

Baseline+MEM: alcatel says sales grew nearly 30 percent in fourth quarter of last year

Figure 5: An example to show that considering context could produce a better translation result. In the example, **Arthur(test)** translates *chengzhang* into a wrong target word *growth*, while **Baseline+MEM** could overcome this mistake with the help of the context modeling.

5.2 Results on Sub-words

We also test the proposed method when the translation unit is sub-word. The baseline and our method using sub-word as translation unit are respectively denoted by **Baseline(sub-word)** and **Baseline(sub-word)+MEM**. The results are shown in row 4 and row 7. From the results, **Baseline(sub-word)+MEM** outperforms **Baseline(sub-word)** by 1.01 BLEU points, indicating that adopting sub-words as translation units still faces the problem of troublesome tokens, and our method could alleviate this problem.

5.3 Our Method vs. Method Using Translation Lexicon

We also compare our method with Arthur et al. (2016)'s method which incorporates a translation lexicon into NMT. Here, the comparison is conducted in two ways based on whether the baseline neural model is fixed or retrained.

Fixed Baseline. Comparing **Arthur(test)** (row 2 in Table 1) and **Baseline+MEM** (row 5 in Table 1), we can see that our proposed method can surpass **Arthur(test)** with 1.05 BLEU points. As there are three differences between our methods and **Arthur(test)**, we take the following experiments to evaluate the effect of each difference.

The first difference is that our memory only stores the lexicon pairs for troublesome words, while **Arthur(test)** utilizes all the available lexicon pairs. We implement another system which is similar to **Arthur(test)**, except that we only utilize the troublesome lexicon pairs. We denote the system by **Tword**. The results are reported in Table 2. From the results, we can find that **Tword** obtains better translation results than **Arthur(test)** while using much fewer lexicon pairs (125K vs. 938K).

The second difference is that we take the context into consideration. When we add the context on the basis of **Tword** (denoted by **+Context**), it further improves the baseline system by 1.03 BLEU points, indicating the importance of the context. Fig.5 shows the mentioned example in Section 1, in which **Arthur(test)** translates *chengzhang* into a wrong target word *growth*, while **Baseline+MEM** could overcome this mistake with the help of the context modeling.

We also implement another system, in which we build the contextual memory for all source words.

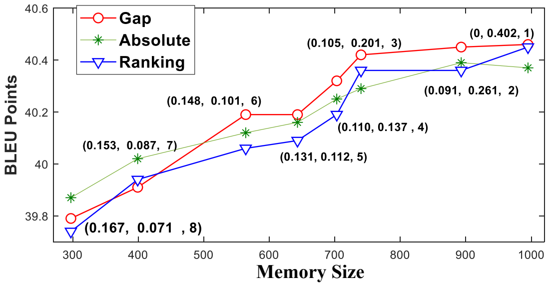

Figure 6: The comparison of different criteria. The gap criterion outperforms others with the increase of the memory size.

We denote the system by **All+Context** and the results are reported in Table 2. As shown in Table 2, **All+Context** surpasses **Arthur(test)** with 0.75 BLEU points while at the cost of 6.4G memory footprint and 1.829s time consuming. However, if we only build the contextual memory for the troublesome words, comparing to **All+Context**, there is only a slighter BLEU points decline (40.19 vs. 40.23) while sharply reduces memory size to 893M and decoding time to 0.511s, showing that our strategy of only building the contextual memory for troublesome words is effective.

The final difference is that we employ the dynamic strategy to balance between NMT and the contextual memory. When we employ this dynamic strategy (denoted by **+Dynamic**), the improvement can further reach 1.37 BLEU points.

Retrained Baseline. In the second comparison, we allow the baseline model to be retrained by Arthur's method (**Arthur(train+test)**). We then implement our method using **Arthur(train+test)** as baseline (denoted by **Arthur(train+test)+MEM**). Comparing the results of these two methods in Table 1 (line 3 and 6), our method is still effective on the retrained model. The average gains are 0.92 BLEU points.

5.4 Effects of Different Exception Criteria

In our method, we investigate three exception criteria to define the troublesome words. The following experiment is conducted to compare their performances. For fairness, the comparison of the three criteria is conducted under the same number of contextual memory, which can be achieved by adjusting the respective thresholds (p_0, g_0 and $rank_0$). The results are reported in Fig. 6, in which the x axis represents the size of contextual memory, the y axis denotes BLEU score, and the numbers in the bracket from left to right are the respective thresholds of **gap**, **absolute** and **ranking**. As shown in Fig. 6, all the three criteria can improve the translation quality. When the memory size is relatively small, absolute criterion performs best. With the size increases, the gap criterion achieves a higher performance than others.

Type	Baseline	Baseline+MEM
Low+Amb	38.15	39.75
Low	38.56	40.03
Amb	39.86	40.67
Others	40.49	40.76

Table 3: The BLEU score on different kinds of sentences. **Low** denotes low frequency words, **Amb** denotes ambiguous words. **Low+Amb** denotes the low frequency and ambiguous words.

Type	Tword	Error	Rectify	Deterio
Total	374	153	70 (45.8%)	11
Low+Amb	77	30	16 (53.3%)	3
Low	144	59	30 (50.8%)	4
Amb	117	48	20 (41.7%)	4
Others	36	16	4 (25%)	0

Table 4: The manual analysis on the word level. Column **Tword** shows the number of troublesome words in sentence. Column **Error** shows the number of errors made by **Baseline** when translates the troublesome words. The number (ratio) of rectification caused by our method is reported in column **Rectify**. Column **Deterio** shows the number of deterioration (the original translation is correct, while our method produces the incorrect translation) caused by our method.

Note that our current criteria only consider one single factor. The combination of different criteria may be more beneficial, and we leave this as our future work.

5.5 Results on Low-Frequency Words and Ambiguous Words

We further analyze our method on specific troublesome words, such as low-frequency words and ambiguous words. Here, we use the following definition in our analysis.

Low-frequency words: The words whose frequency is lower than 100.

Ambiguous words: Assume a word s_m contains K candidate translations with a probability p_k^L. If the entropy of probability distribution $-\sum_{k=1}^{K} p_k^L log p_k^L > E_0$ ($E_0 = 1.5$ in this paper), we treat this word as an ambiguous word.

Therefore, the sentences containing troublesome words can be divided into four different parts: 1) sentences which contain both low-frequency and ambiguous words (**Low+Amb**,

Model	Rectify	Deterio
Arthur(test)	51	17
Baseline+MEM	70	11

Table 5: The numbers of rectification (**Rectify**) and deterioration (**Deterio**) caused by different models.

986 sentences), 2) sentences which contain low-frequency words but no ambiguous words (**Low**, 1427 sentences). 3) sentences which contain ambiguous words while do not contain low-frequency words (**Amb**, 1301 sentences). 4) Other sentences (**Others**, 832 sentences). The results are reported in Table 3. From this table, we observe that our proposed method improves the translation quality on all kinds of sentences. **Low+Amb** performs best (**Low** the second), indicating that our method is most effective in dealing with low-frequency words. The improvement on **Amb** is 0.81 BLEU points, showing that our method can also well handle the ambiguous words.

We also conduct a manual analysis to figure out how many troublesome words could be rectified by our method. We randomly select 200 testing sentences, and count the following three numbers: 1) the number of troublesome words in the sentence (**Tword**), 2) the number of mistakes produced by **Baseline** (**Error**), 3) the number (ratio) of rectification using our method (**Rectify**). 4) The number of deterioration caused by our method (**Deterio**). The statistics are reported in Table 4. From the results, we can get similar conclusions that our method is most effective on low-frequency and ambiguous words with the rectification rate 50.8% and 41.7% respectively.

We can notice that the proposed method also produces 11 deterioration cases (**Deterio**) when rectifying the troublesome words. As a comparison, we also count the total rectification and deterioration numbers of **Arthur(test)**. The results are reported in Table 5. These results show that our method could rectify more words (51 vs. 70) with less deterioration (17 vs. 11) than **Arthur(test)**.

6 Results on EN-DE Translation

We also test our method on EN-DE translation and the results are reported in Table 6. We can see that our method is still effective on EN-DE translation. Specifically, when the translation unit is word, the proposed method improves the baseline by 1.13 BLEU points. The improvement is 0.76 BLEU points when the translation unit is sub-word.

Model	Unit	EN-DE	
		dev	test
Baseline	word	20.28	21.04
Baseline+MEM	word	$21.34^{\dagger}$	$22.17^{\dagger}$
Baseline	sub-word	22.10	22.85
Baseline+MEM	sub-word	$23.05^{\dagger}$	23.61^{*}

Table 6: The results on EN-DE translation. "*" indicates that it is statistically significantly better ($p < 0.05$) than system X and "†" indicates $p < 0.01$.

7 Related Work

The related work can be divided into three categories and we describe each of them as follows:

Neural Turing Machine for NMT. Our idea is first inspired by the Neural Turing Machine (NTM) (Graves et al., 2014, 2016) and memory network (Weston et al., 2014). (Wang et al., 2017a) used special NTM memory to extend the decoder in the attention-based NMT. In their method, the memory is used to provide temporary information from source to assist the decoding process. In contrast, our work uses memory to store contextual knowledge in the training data.

Smaller translation granularity. Our work is also inspired by the other studies to deal with the low-frequency and ambiguous words (Vickrey et al., 2005; Zhai et al., 2013; Rios et al., 2017; Carpuat and Wu, 2007; Li et al., 2016). Among them, the most relevant is the work that decomposes the low-frequency words into smaller granularities, e.g, hybrid word-character model (Luong and Manning, 2016), sub-word model (Sennrich et al., 2016) or word piece model (Wu et al., 2016). These methods mainly focus on low-frequency words that are just a subset of the troublesome words. Furthermore, our experimental results show that even using a smaller translation unit, the NMT model still faces the problem of troublesome tokens and our method could alleviate this problem.

Combining SMT and NMT. Our ideas are also inspired by the work which combines SMT and NMT. Earlier studies were mostly based on the SMT framework, and have been deeply discussed by the review paper in Zhang and Zong (2015). Later, the researchers transfer to NMT framework, e.g. (Wang et al., 2017b; Zhang and Zong, 2016; Zhou et al., 2017; Tu et al., 2016; Mi et al., 2016; He et al., 2016; Dahlmann et al., 2017; Wang et al., 2017c,d; Gu et al., 2018; Zhao et al., 2018). The most relevant studies are Arthur et al. (2016) and

Feng et al. (2017). They incorporate the lexicon pairs into NMT to improve the translation quality. There are three differences between our method and theirs. First, we only utilize the lexicon pairs for the troublesome words, rather than using all lexicon pairs. Second, we take contextual information into consideration for memory construction. Third, we design a dynamic strategy to balance the memory and NMT. The experiments show the superiority of our proposed methods.

8 Conclusions

To address troublesome words in NMT, we have proposed a novel memory-enhanced framework. We first define and detect the troublesome words, then construct a contextual memory to store the translation knowledge and finally access the contextual memory dynamically to correctly translate the troublesome words. The extensive experiments on Chinese-to-English and English-to-German translation tasks demonstrate that our method significantly outperforms the strong baseline models in translation quality, especially in handling the troublesome words.

Acknowledgments

The research work described in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2016QY02D0303 and the Natural Science Foundation of China under Grant No. 61333018. The research work in this paper also has been supported by Beijing Advanced Innovation for Language Resources of Beijing Language and Culture University.

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. *In Proceedings of EMNLP 2016*, pages 1557–1567.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In Proceedings of ICLR 2015*.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *In proceedings of EMNLP 2007*, pages 61–72.

Kyunghyun Cho, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *In Proceedings of EMNLP 2014*, pages 1724–1734.

Leonard Dahlmann, Evgeny Matusov, Pavel Petrushkov, and Shahram Khadivi. 2017. Neural machine translation leveraging phrase-based models in a hybrid search. *In proceedings of EMNLP 2017*, pages 1422–1431.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *In proceedings of NAACL 2013*.

Yang Feng, Shiyue Zhang, Andi Zhang, Dong Wang, and Andrew Abel. 2017. Memory-augmented neural machine translation. *In proceedings of EMNLP 2017*, pages 1401–1410.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1601.03317*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. 2018. Search engine guided nonparametric neural machine translation. *In proceedings of AAAI 2018*.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with smt features. *In Proceedings of AAAI 2016*.

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

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. *In Proceedings of ACL 2015*, pages 124–129.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? a case study on 30 translation directions. *arXiv preprint arXiv:1610.01108*.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown word in neural machine translation. *In Proceedings of IJCAI 2016*, pages 2852–2858.

Minh Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *In proceedings of EMNLP 2016*, pages 1054–1063.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *In Proceedings of EMNLP 2015*, pages 1412–1421.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding model for neural machine translation. *In Proceedings of EMNLP 2016*, pages 955–960.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *In Proceedings of ACL 2002*, pages 311–318.

Annette Rios, Laura Mascarell, and Rico Sennrich. 2017. Improving word sense disambiguation in neural machine translation with sense embeddings. In *In proceedings of WMT 2017*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *In Proceedings of ACL 2016*, pages 1715–1725.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. *In Proceedings of ACL 2016*, pages 76–85.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and ukasz Kaiser. 2017. Attention is all you need. *arXiv preprint arXiv:1601.03317*.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word-sense disambiguation for machine translation. In *In proceedings of ACL 2005*, pages 771–778.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2017a. Memory-enhanced decoder for neural machine translation. *In proceedings of EMNLP 2017*, pages 278–286.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017b. Neural machine translation advised by statistical machine translation. *In proceedings of AAAI 2017*.

Xing Wang, Zhaopeng Tu, Deyi Xiong, and Min Zhang. 2017c. Translating phrases in neural machine translation. *In proceedings of EMNLP 2017*, pages 1432–1442.

Yining Wang, Yang Zhao, Jiajun Zhang, Chengqing Zong, and Zhengshan Xue. 2017d. Towards neural machine translation with partially aligned corpora. *In proceedings of IJCNLP 2017*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Feifei Zhai, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2013. Handling ambiguities of bilingual predicate-argument structures for statistical machine translation. In *In proceedings of ACL 2013*, pages 1127–1136.

Jiajun Zhang and Chengqing Zong. 2015. Deep neural networks in machine translation: An overview. *IEEE Intelligent Systems*, 30(5):16–25.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. *In Proceedings of EMNLP 2016*, pages 1535–1545.

Yang Zhao, Yining Wang, Jiajun Zhang, and Chengqing Zong. 2018. Phrase table as recommendation memory for neural machine translation. *In proceedings of IJCAI 2018*, pages 4609–4615.

Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. 2017. Neural system combination for machine translation. *In Proceedings of ACL 2017*, pages 378–384.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *In Proceedings of NAACL 2016*, pages 30–34.

Top-down Tree Structured Decoding with Syntactic Connections for Neural Machine Translation and Parsing

Jetic Gū, Hassan S. Shavarani, Anoop Sarkar
TASC 9404, 8888 University Drive,
Simon Fraser University,
Burnaby, BC V5A 1S6, Canada
`{jeticg, sshavara, anoop}@sfu.ca`

Abstract

The addition of syntax-aware decoding in Neural Machine Translation (NMT) systems requires an effective tree-structured neural network, a syntax-aware attention model and a language generation model that is sensitive to sentence structure. We exploit a top-down tree-structured model called DRNN (Doubly-Recurrent Neural Networks) first proposed by Alvarez-Melis and Jaakola (2017) to create an NMT model called Seq2DRNN that combines a sequential encoder with tree-structured decoding augmented with a syntax-aware attention model. Unlike previous approaches to syntax-based NMT which use dependency parsing models our method uses constituency parsing which we argue provides useful information for translation. In addition, we use the syntactic structure of the sentence to add new connections to the tree-structured decoder neural network (Seq2DRNN+SynC). We compare our NMT model with sequential and state of the art syntax-based NMT models and show that our model produces more fluent translations with better reordering. Since our model is capable of doing translation and constituency parsing at the same time we also compare our parsing accuracy against other neural parsing models.

1 Introduction

Neural machine translation (NMT) models were initially proposed as extensions of sequential neural language models (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015) or convolutions over n-grams in the decoder (Kalchbrenner and Blunsom, 2013). Early methods for discriminative training of machine translation models showed that the loss functions for translation were not sensitive to the production of certain important words such as verbs, without which the output sentence might be uninterpretable by humans. A good solution was to penalise such bad outputs using tree structures which get very low scores if important words like verbs are missing (Chiang, 2005; Zollmann and Venugopal, 2006; Galley et al., 2006). To this end, there has been a push to incorporate some syntax into NMT models: Sennrich and Haddow (2016) incorporate POS tags and dependency information from the source side of a translation pair in NMT models. Stahlberg et al. (2016) use source language syntax to guide the decoder of an NMT system to follow hierarchical structures (Hiero) rules (Chiang, 2005). Eriguchi et al. (2016) and Bastings et al. (2017) use tree-structured encoders to exploit source language syntax. Aharoni and Goldberg (2017) take the approach of serialising the parse trees to use in a sequential decoder. Eriguchi et al. (2017) propose an NMT+RNNG model, which explores the possibilities of using dependency syntax trees from the target language using StackLSTMs (Dyer et al., 2015, 2016) to aid a sequential decoder. These approaches showed promising improvements in translation quality but all the models in previous work, even the model in Eriguchi et al. (2017) which uses RNNG, are bottom-up tree structured decoders.

In contrast, we use a top-down tree-structured model called DRNN (Doubly-Recurrent Neural Networks) first proposed by Alvarez-Melis and Jaakkola (2017) to model structural syntactic information for NMT. We call our novel NMT model Seq2DRNN, using DRNNs as a tree-structured decoder combined with a sequential encoder and a novel syntax-aware attention model.

All the previous work in syntax-aware NMT mentioned above has focused on dependency parsing as the syntactic model. In contrast, we wish to pursue phrase structure (aka constituency) based syntax-based NMT. We provide some analysis that shows that constituency information can help recover information in NMT decoding.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 401–413
Brussels, Belgium, October 31 - November 4, 2018.

We perform extensive experiments comparing our model against other state-of-the-art sequence to sequence and syntax-aware NMT models and show that our model can improve translation quality and reordering quality. The model performs translation and constituency parsing simultaneously so we also compare our parsing accuracy to other neural parsing models.

2 Model Description

In this paper, source sentence will be written as $\mathbf{f} = x_1, x_2, ..., x_n$, target sentence as $\mathbf{e} = y_1, y_2, ..., y_m$ where y_j is a word. Additionally, p_k represents a non-terminal symbol (constituent, phrase) in the target sentence constituency tree. $[\mathbf{v}; \mathbf{u}]$ stands for concatenation of vectors $\mathbf{v}$ and $\mathbf{u}$. $\mathbf{W}(x)$ is the word embedding of word x.

The design of our NMT system follows the encoder-decoder model (also known as a sequence to sequence model) proposed by Cho et al. (2014) and Sutskever et al. (2014). Our system uses a standard bidirectional gated RNN (BiLSTM or bidirectional Long Short-Term Memory) (Huang et al., 2015) as the encoder and our proposed tree-structured RNN as the decoder.

2.1 Sequence to Sequence NMT (Seq2Seq)

Neural machine translation models generally consist of an encoder, a decoder and an attention model (Luong et al., 2015; Cho et al., 2015). The encoder is used to produce hidden representations of the source sentence, which is fed into the decoder along with the attention information to produce the translation output sequence.

A common approach is to use bidirectional LSTMs as encoder, produce forward hidden states $\overrightarrow{\mathbf{h}}_i$ and backward hidden states $\overleftarrow{\mathbf{h}}_i$, and the final representation $\mathbf{h}_i^{\text{enc}}$ is the concatenation of both:

$$\begin{aligned}\overrightarrow{\mathbf{h}}_i &= \overrightarrow{\text{RNN}}_{\text{enc}}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{W}_x(x_i)) \\ \overleftarrow{\mathbf{h}}_i &= \overleftarrow{\text{RNN}}_{\text{enc}}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{W}_x(x_i)) \\ \mathbf{h}_i^{\text{enc}} &= [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]\end{aligned} \tag{1}$$

The decoder takes the output of the encoder and generates a sequence in target language. The attention mechanism provides additional context vectors $\mathbf{c}_j$ which is a weighted average contribution of each $\mathbf{h}_i$ source side encoding.

$$\begin{aligned}\mathbf{h}_j^{\text{dec}} &= \text{RNN}_{\text{dec}}(\mathbf{h}_{j-1}^{\text{dec}}, [\mathbf{o}_{j-1}; \mathbf{c}_j]) \\ \mathbf{o}_j &= \text{softmax}(\mathbf{U}\mathbf{h}_j^{\text{dec}} + \mathbf{b})\end{aligned} \tag{2}$$

Here, $\mathbf{U}$ is the readout matrix and $\mathbf{b}$ is the bias vector. $\mathbf{o}_j$ is the output word embedding.

2.2 NMT with a Tree-Structured Decoder (Seq2DRNN)

The output translation from a translation system should convey the same meaning as the input. This includes the correct word choices but also the right information structure. Sentence structure can be viewed as starting with an action or state (described via verbs or other predicates) and the entities or propositions involved in that activity or state (usually described via arguments to verbs). Thus certain words in the output translation, like verbs, are crucial to the understanding of the target language sentence but only provide marginal value in n-gram matching evaluations like the BLEU score. Tree representations, produced via dependency parsing and constituency parsing, are useful because they are sensitive to this information structure. Our tree-structured decoder uses a neural network to generate trees (described in §2.2.1), which is incorporated into an NMT model (our novel encoder-decoder model is in §2.2.2) which translates and produces a parse tree. Our new Syntactic Connection method (SynC) is described in §2.2.4 which is combined with the Seq2DRNN model (Seq2DRNN+SynC) and the attention mechanism (§2.2.3).

2.2.1 Doubly-Recurrent Neural Network

The Doubly-Recurrent Neural Network model (Alvarez-Melis and Jaakkola, 2017) takes a vector representation as input and generates a tree. Alvarez-Melis and Jaakkola (2017) show that the DRNN model can effectively reconstruct trees but they do not use DRNNs within a full-scale NMT system. We also use DRNNs for phrase-structure (aka constituency) tree structures rather than dependency trees as in previous work. DRNN decoding proceeds top-down; the generation of nodes at depth d depend solely on the state of nodes at depth $< d$. Unlike previous work in tree-structured decoding for NMT by Dyer et al. (2016) and Eriguchi et al. (2017), the output sentence generation is not done in sequence, where the target word y_j is generated after all $y_{<j}$ are generated. DRNN first predicts the structure of the sentence and then expands each component to predict words. When generating y_j, information regarding the structure of words from 1 to $j - 1$ and $j + 1$ to m can be used to aid prediction of y_j.

A DRNN consists of two recurrent neural network units, which separately process ancestral and fraternal information about nodes in the tree. Assuming a node is v, its immediate parent node is $P(v)$ and its closest sibling on the left side (appears in the target language sequence just before v) is $S(v)$. The label of node v is $\mathbf{z}_v$. Then the ancestral hidden representation $\mathbf{h}^a$ and fraternal representation $\mathbf{h}^f$ of a node are calculated with Equation 3.

$$\begin{aligned} \mathbf{h}_v^a &= \text{RNN}_{\text{dec}}^a(\mathbf{h}_{P(v)}^a, \mathbf{z}_{P(v)}) \\ \mathbf{h}_v^f &= \text{RNN}_{\text{dec}}^f(\mathbf{h}_{S(v)}^f, \mathbf{z}_{S(v)}) \end{aligned} \tag{3}$$

$\mathbf{h}_v^a$ and $\mathbf{h}_v^f$ are then combined to produce the hidden state of node v for prediction (predictive hidden state $\mathbf{h}_v$, Equation 4), which is used to predict the labels of node v.

$$\mathbf{h}_v = \tanh(\mathbf{U}^f \mathbf{h}_v^f + \mathbf{U}^a \mathbf{h}_v^a) \tag{4}$$

During the label prediction, DRNN first makes topological decisions: whether (i) the current node is a leaf node (node with no children, α_v); then (ii) whether the current node has siblings on its right-hand side (γ_v). Both predictions are done using sigmoid activations:

$$\begin{aligned} o_v^a &= \sigma(\mathbf{u}^a \mathbf{h}_v) \\ \alpha_v &= 1 \text{ if } o_v^a \text{ is activated.} \end{aligned} \tag{5}$$

$$\begin{aligned} o_v^f &= \sigma(\mathbf{u}^f \mathbf{h}_v) \\ \gamma_v &= 1 \text{ if } o_v^f \text{ is activated.} \end{aligned} \tag{6}$$

Then, label representation $\mathbf{o}_v$ is predicted using α_v and γ_v, and the predictive hidden state $\mathbf{h}_v$:

$$\mathbf{o}_v = \text{softmax}(\mathbf{U}_o \mathbf{h}_v + \alpha_v \mathbf{u}_a + \gamma_v \mathbf{u}_f) \tag{7}$$

At inference time each node at the same depth is expanded independently, therefore the whole process can be parallelised. This parallelism advantage is not observed in any of the sequential decoders that generate output sequence strictly from left to right nor from right to left (§4 has more discussion).

2.2.2 Parsing and Translating with DRNN

A DRNN is capable of producing a tree structure with labels given an input vector representation. If we train the DRNN to produce parse trees from the output of an encoder RNN, this system will be able to translate and parse at the same time. (Alvarez-Melis and Jaakkola, 2017) in their paper provided a proof-of-concept NMT experiment using dependency trees. Instead of an single RNN unit to process fraternal information they had multiple for modelling the fraternal information on syntactic tree structure because dependency parse trees differentiate between left and right children. The model itself also disregards the sequentiality of natural language, and lack attention mechanisms to make it work in exchange for a strict top-bottom decoding procedure.

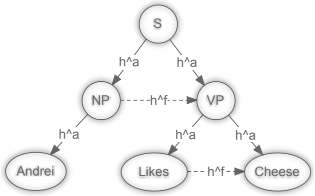

Figure 1: Seq2DRNN on Constituency Tree

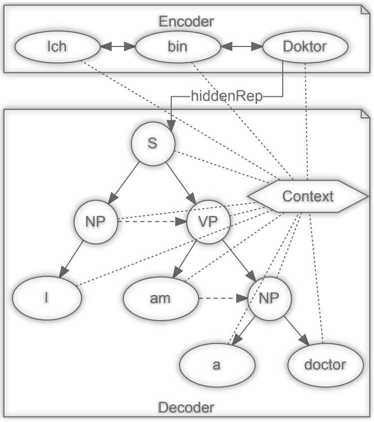

Figure 2: Seq2DRNN Encoder-Decoder

We use constituency parse trees to represent sentences in the target language (Figure 1) because constituency or phrase-structure trees are more amenable to top-down derivation compared to dependency trees. It is also easier to model for DRNN, and presumably more capable at handling unknown words which is common in NMT systems with limited vocabulary size.

Each node on the tree represents either a terminal symbol (a word) or a non-terminal symbol (a clause or phrase type). The sub-tree dominated by a non-terminal node is the clause or phrase identified with this non-terminal node label.

A conventional bidirectional RNN (BiLSTM) encoder (Cho et al., 2014; Sutskever et al., 2014) is used to produce hidden states for the decoder (see Figure 2).

We use breadth-first search to implement the Seq2DRNN decoder. Two queues are used here: *current* queue which is the queue containing all of the nodes on the currently being processed depth, and *next* queue with nodes on the next depth (Algorithm 1 has all the details).

The decoding process starts from top to bottom, from root to its children, then to its grandchildren, and so on until the leaf nodes which are the output words.

In our implementation, sentence clauses (S nodes) are generated as the children of the root node to generalise over sentence types and in case there are multiple sentences in a single translation pair.

Initially, the *current* queue will only have one entry: the root node, which is initialised with the hidden representation of the source sentence.

Each node in the *current* queue is expanded in the following manner: first generate all of its siblings and add them to the *current* queue, and if any node happens to be non-terminal, generate its first child and add it to the *next* queue. After the *current* queue is empty, make *next* the new *current* queue and start working on nodes at the next depth.

For training, we use back-propagation through trees using the approach in Goller and Küchler (1996). In the forward pass, the source sentence is encoded into a hidden representation and fed into the decoder. The decoder generates the tree, predicts the labels of every node from root to leaves. Then in the backward pass, gradients are calculated and used to update the parameters. The loss calculation includes losses in topological predictions: o_v^a and o_v^f (Equations 5 and 6) and label predictions: $\mathbf{o}_v$ (Equation 7).

$$\begin{aligned}\mathrm{Loss(e)} = &\sum_v \mathrm{Loss}^{\mathrm{label}}(\mathbf{o}_v, \hat{\mathbf{o}_v})+\\ &\alpha \sum_v \mathrm{Loss}^{\mathrm{topo}}(o_v^a, \hat{o_v^a})+\\ &\alpha \sum_v \mathrm{Loss}^{\mathrm{topo}}(o_v^f, \hat{o_v^f})\end{aligned} \tag{8}$$

Here α is a hyper-parameter.

2.2.3 Attention Mechanism

Attention mechanisms usually work by adding an additional context vector during label prediction. We use a variation of an existing attention mechanism proposed by Luong et al. (2015). In our attention model, we produce a context vector $\mathbf{c}_v$ for every node v by looking at all hidden states produced by the encoder $\mathbf{h}_i^{\mathrm{enc}}$, then calculating the weights and adding up the weighted hidden states.

$$\mathrm{weight}_{v,i} = \mathbf{V}_a \tanh(\mathbf{W}_a \mathbf{h}_v + \mathbf{U}_a \mathbf{h}_i^{\mathrm{enc}}) \in \mathbb{R} \tag{9}$$

$$\mathbf{c}_v = \sum_{i=1}^{n} \hat{\mathrm{weight}}_{v,i} \mathbf{h}_i^{\mathrm{enc}} \tag{10}$$

After the calculation in Equation 9, the weights are normalised with a softmax function before being used to calculate the context vector. The attention module allows the generation of labels to pay more attention to specific token representations of words in the input sentence.

Algorithm 1 Seq2DRNN Decoder

```
procedure DECODE(hiddenRep)
    currentQueue ← Node from hiddenRep
    nextQueue ← empty
loop:
    if currentQueue is not empty then
        node ← currentQueue.pop()
        Generate labels of node
        if node has siblings then
            currentQueue ← sibling(node)
        if node has children then
            nextQueue ← child(node)
        goto loop
    ▷ all nodes at current depth are generated
    ▷ move on to the next depth
    if nextQueue is not empty then
        currentQueue ← nextQueue
        nextQueue ← empty
        goto loop.
    ▷ both queues should be empty now
```

2.2.4 SynC: Syntactic Connections for Language Generation (Seq2DRNN+SynC)

A conventional Seq2Seq model uses an RNN language model (Cho et al., 2014; Sutskever et al., 2014) conditioned on the input representation produced by the encoder to generate the output one word at a time (Equation 11). The prediction of a word y_j is directly conditioned on previously generated words where $\mathbf{c}_j$ is the context vector.

$$\mathrm{P}(\mathbf{e}) = \prod_j \mathrm{P}(y_j | y_{<j}, \mathbf{c}_j) \tag{11}$$

The problem with this word-level language model is that it treats a sentence as a plain sequence of symbols regardless of its syntactic construction. Sentences may contain multiple subordinate clauses and their boundaries are not well-modelled by sequential language models.

We propose a new method to connect the hidden units in the Seq2DRNN decoder that pays attention to contextual tree relationships. The prediction of the representation of a word or a constituent z_j (if a constituent then p_j, if a word then y_j) is defined as follows:

$$\begin{aligned}&\mathrm{P}(z_j \mid y_{<j}, \mathbf{c}_j) =\\ &\quad \mathrm{P}(z_j \mid y_{<j}, y_k(\forall k, z_j \in p_k \vee\\ &\quad \mathrm{precedes}(y_k, z_j)) \mid \mathbf{c}_j)\end{aligned} \tag{12}$$

The generation of the representation of a word/constituent z_j, which is part of the clause

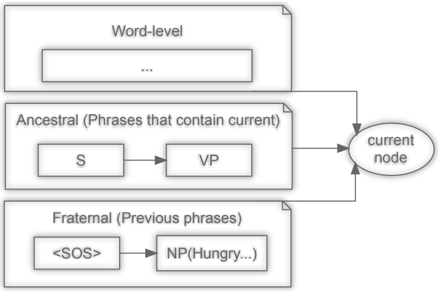

Figure 3: SynC example: the three types of information explicitly modelled when generating the current node/word *likes* in *Andrei when starving likes cheese*.

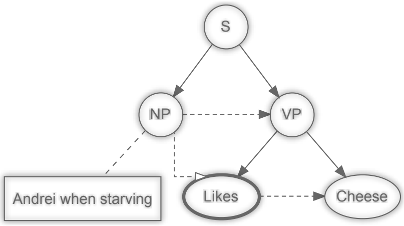

Figure 4: SynC in action; when generating the word *likes* in *Andrei when starving likes cheese*, the prediction will be made knowing that the preceding clause is a noun phrase.

that contain it ($z_j \in p_k$), with clauses before which ($\text{precedes}(p_k, z_j)$), is conditioned on the following information: i) Word-level: previously generated words $y_{<j}$; ii) Ancestral Clause: the clauses that contain the current word p_k, i.e. ($\forall k z_j \in p_k$); iii) Fraternal Clause: the clauses that precede the current clause p_k, i.e. ($\forall k\, \text{precedes}(p_k, z_j)$).

In practice, the generation of a node looks at the following representations:

1. Word-level: an RNN unit that produces the representation of previous words as a sequence $y_{<j}$;
2. Ancestral: treating the ancestors of the current node as a sequence (from root to the immediate parent), the representation of that sequence: $p_k(\forall k, z_j \in p_k)$;
3. Fraternal: treating the previous siblings of the current node as well as the previous siblings of its parent node and so on as a sequence, the representation of that sequence: $p_k(\forall k, \text{precedes}(p_k, z_j))$.

SynC creates connections in the tree-structured decoder that pays attention to the structural context of generation of each terminal or non-terminal symbol in the phrase structure tree. For example in English, it is common for verb phrases to follow a noun phrase. But that noun phrase could itself be a subordinate clause with its own verb phrases. In this case, our goal is to explicitly model the fact that the previous phrase is a noun phrase instead of just the entire sequence of words.

SynC can be easily incorporated in the proposed Seq2DRNN model (Seq2DRNN+SynC). In addition to the fraternal RNN unit that focuses on preceding sibling nodes, and the ancestral DRNN unit that focus on parent nodes, a node would also look at its parent's previous sibling state (the hidden vector representation of preceding clauses from the very beginning of the sentence). When a non-terminal symbol is expanded into a sub-tree, it's first child will not have a previous sibling to provide fraternal information ($S(v) = \text{Null}$, as in Equ 3). In this case, SynC establishes connection between its first child and its parent's fraternal information provider for such fraternal RNN state ($S(v) = S(P(v))$).

$$\mathbf{h}_v^f = \begin{cases} S(v) \neq \text{Null}, & \text{RNN}_{\text{dec}}^f(\mathbf{h}_{S(v)}^f, \mathbf{z}_{S(v)}) \\ S(v) = \text{Null}, & S(v) := S(P(v)), \\ & \text{RNN}_{\text{dec}}^f(\mathbf{h}_{S(v)}^f, \mathbf{z}_{S(v)}) \end{cases} \tag{13}$$

An example is shown in Figure 4. In this case, a word-level language model will regard *starving* as the previous word, which is less helpful for the prediction of a verb phrase *likes cheese*.

3 Experiments

3.1 Model Training

Experiments in this paper utilise constituency trees on the target side, these trees are obtained by using the Stanford Lexical Parser (Klein and Manning, 2003a) which we chose for its speed and accuracy prior to training.

This procedure of pre-parsing data is not required at test time, our NMT system would take a sentence as input and produces the translation in target language along with its constituency tree as output.

We use the German-English dataset from IWSLT2017 [1] for our experiments, and tst2010-2015 as the test set (Table 1).

To compare with other decoders that utilise target-side syntactic information, we also evaluate on three more datasets from News Commentary v8 using newstest2016 as testset (Table 2).

We replace all rarely occurring words with

---

[1] The International Workshop on Spoken Language Translation Evaluation 2017: https://sites.google.com/site/iwsltevaluation2017/TED-tasks

Train pairs	226,572
Test pairs	8,079
Unique source words	128,857
Unique target words	61,566
Average source sentence length	21
Average target sentence length	20

Table 1: IWSLT2017 Dataset information

Language pair	DE-EN	CS-EN	RU-EN
Train pairs	166,313	134,453	131,492
Test pairs	2,999	2,999	2,998
Uniq. src lex	149,318	153,173	159,074
Uniq. tgt lex	68,415	59,909	64,220
Avg. src len	25	22	25
Avg. tgt len	25	25	26

Table 2: News Commentary v8 Dataset information

	BLEU	RIBES	Perplx.
Seq2Seq	22.83	81.5	1.828
Seq2DRNN	23.53	80.4	**1.644**
Seq2DRNN+SynC	**25.36**	**82.6**	1.750

Table 3: IWSLT17 Experiment results

UNK (Unknown) tokens. Only the top 50,000 most frequent words are kept.

3.2 Modelling details

The implementation of all models in this paper is done using DyNet (Neubig et al., 2017a) with Autobatching (Neubig et al., 2017b). We use Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) as RNN units. Each LSTM unit has 2 layers, with input and hidden dimension of 256. We use a minibatches of 64 samples. We use early stopping mechanism for all experiments and Adam optimiser (Kingma and Ba, 2015) as trainer.

Note that this configuration is significantly smaller in both dimension and batch size than those presented in IWSLT2017 due to hardware limitations. All experiments are carried out on a single GTX 1080 Ti GPU with 11GB of VRAM.

3.3 Results

Table 3 and Table 4 has the BLEU (Papineni et al., 2002) and RIBES (Isozaki et al., 2010) scores. In our IWSLT2017 tests, both Seq2DRNN and Seq2DRNN+SynC produce better results than the Seq2Seq baseline model in terms of BLEU scores, while Seq2DRNN+SynC also produces better RIBES scores indicating better reordering of phrases in the output. The Seq2DRNN+SynC model performs better than the Seq2DRNN model. Both Seq2Seq and Seq2DRNN+SynC are able to produce results with lower perplexities than the baseline Seq2Seq model on the test data.

In our News Commentary v8 tests, the same relative performance from Seq2DRNN(SynC) can be observed. The Seq2DRNN+SynC model is also able to out-perform the Str2Tree model proposed by Aharoni and Goldberg (2017) and NMT+RNNG by Eriguchi et al. (2017) in most cases. Note that Eriguchi et al. (2017) used dependency information instead of constituency information as presented in our work and Aharoni and Goldberg (2017)'s work.

Table 5 shows an example translation from all of the models we use in our experiments. Seq2Seq is able to translate with the correct vocabulary, but the sentences are often syntactically awkward. As the sentence length increases the syntactic fluency of Seq2Seq gets worse. Seq2DRNN is able to produce more syntactically fluent sentences since each lowest sub-clause contains typically ≤ 5 words. Seq2DRNN+SynC produces the best results in this example: produces more syntactically fluent sentences, chooses the right words in the right place more frequently.

We also took several examples from our IWSLT17 experiment and blank out certain nouns by replacing them with unknown tokens (Table 6). Note that in our training set, most sentences do not have unknown tokens, and those that do only have at most 1. Our assumptions of the observed patterns in this case are: i) the proposed models are more capable at handling unknown tokens; ii) while Seq2DRNN is more capable at retaining the structure of the sentence, it cannot rely on a wider context to predict certain common phrases with noises in the source sentence; iii) the proposed Seq2DRNN+SynC model is more capable at handling unknown words both in the sense of being better at retaining sentence structure and handling noisy input.

3.4 Attention Module

We visualise the attention weights of our Seq2DRNN+SynC model. Attention §2.2.3 computes a context vector for each node in the tree (a weighted sum of the source side vector representations). For the translation pair in Figure 2, we show the attention weight of each pair of word and node (Equation 9) in Figure 5.

The addition of syntax nodes in the output enables the attention model to be used more effectively and is also valuable for visual inspection of syntactic nodes in the output mapping to the input.

Dataset	DE-EN		CS-EN		RU-EN	
	BLEU	RIBES	BLEU	RIBES	BLEU	RIBES
Seq2Seq	16.61	73.8	11.22	69.6	12.03	69.6
Str2Tree	16.13	—	11.65	—	11.94	—
NMT+RNNG	16.41	75.0	12.06	**70.4**	12.46	71.0
Seq2DRNN	16.90	75.1	11.84	67.3	12.04	69.7
Seq2DRNN+SynC	**17.21**	**75.8**	**12.11**	70.3	**12.96**	**71.1**

Table 4: News Commentary v8 Experiment results. Seq2Seq and NMT+RNNG results are taken from Eriguchi et al. (2017), Str2Tree (string-to-linearised-tree) results (no RIBES scores) come from Aharoni and Goldberg (2017) All numbers reported here are of non-ensemble models.

Source	wir wiederholten diese übung mit denselben studenten. was glauben sie passiert nun? nun verstanden sie den vorteil des prototyping. so wurde aus demselben, schlechten team eines unter den besten. sie produzierten die höchste konstruktion in der geringsten zeit.
Literal	we repeated this exercise with the same students. now what do you believe happened? now they understand the value of prototyping. so the same terrible team became one of the very best. they produced the tallest construction in the shortest time.
Gold	we did the exercise again with the same students. what do you think happened then? so now they understand the value of prototyping. so the same team went from being the very worst to being among the very best. they produced the tallest structures in the least amount of time.
Seq2Seq	we repeated this with the same students. what happened you think differently? now, you know, the advantage of the design of the cycle. so, the same one of the team of the team among the best. it produced songs in the slightest building.
Seq2DRNN	well repeat these queries with the same students. what do you think of this? now it understood the advantage of the interests. that's been made of the same thing of one thing. they produced the highest construction of the best time at the best time.
Seq2DRNN+SynC	we repeated this practice with the same students. what do you think happened? now, they understood the value of prototyping. it was being made of the same thing of one of the best ones. they produced the highest construction in the best time.

Table 5: Translation Sample. *Gold* is the reference, and *Literal* is produced by a bilingual German-English speaker.

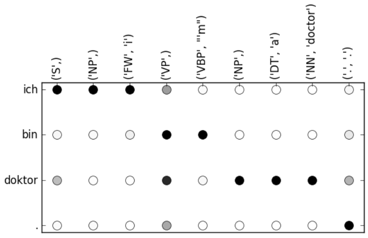

Figure 5: Attention Module in Seq2DRNN+SynC. "ich" means "I", "bin" is "am", "doktor"'s literal translation is "doctor". Darker colour means higher weight (relevance score) as calculated in Equation 9. The values of each column sum up to 1. The attention weights in this example perfectly align with the appropriate clauses. Additional example is provided in the appendix.

3.5 Parsing Quality

To evaluate the parsing quality, we follow the approach by Vinyals et al. (2015) and train a DRNN(SynC) model to produce English to English(Tree) translation. We use the same data and experiment settings that Vinyals et al. (2015) used: the Wall Street Journal Penn Treebank English corpus with golden constituency structure, 256 for input/hidden dimension and 3 layers of RNN. We evaluate on section 23 of the aforementioned WSJ data using EVALB[2]. The results are presented in Table 7.

Although falling short behind more specifically designed models like the RNNG by Dyer et al. (2016), our model is able to produce better results than the LSTM+AD model proposed by Vinyals et al. (2015), which is more comparable to ours since they are also using an NMT model to do constituency parsing. Since our work is more focusing on the translation aspect, optimising and designing a dedicated parser is slightly off-topic here. Nevertheless, it is worth noting that in 50.89% of the cases, Seq2DRNN+SynC was able to produce output that perfectly matches the reference. The same number for sentences with less than 40 words is 52.16%, while the F-measure increases to 90.5. This shows Seq2DRNN(SynC) when doing parsing can produce outputs of similar quality when handling longer sentences.

We also do evaluation on our translation results from the IWSLT dataset. Since translation results do not come with reference parse trees, we parse the output of our decoder using the same parser we used in our other experiments: the Stanford Parser. Constituency parsing evaluation is done using Precision/Recall/F1-scores on the output constituent spans (unlabelled) and spans and labels (labelled). The results are presented in Table 8 and Table 9. The parser we use gets F1 score of 87.04 on Penn Treebank English constituency

---

[2] https://nlp.cs.nyu.edu/evalb/

0	src	es war **zeit** zum **abendessen** und wir hielten ausschau nach einem **restaurant**.	die gute **nachricht** ist , dass die **person** , die das gesagt hat **ann coulter** war .
	S2S	and it was **time** for **dinner** , and we were looking for a **restaurant**.	the good **news** is that the **person** who said that was **ann coulter** .
	S2D	it was the **time** for **dinner** , and we were looking for a **restaurant**.	the good **news** is that the **person** who said that was **ann coulter** .
	S2D+L	it was **time** for **dinner** , and we were looking for a **restaurant**.	the good **news** is that the **person** who said that was **ann coulter** .
1	src	es war **zeit** zum **abendessen** und wir hielten ausschau nach einem **UNK**.	die gute **UNK** ist , dass die **person** , die das gesagt hat **ann coulter** war .
	S2S	and it was **time** for **dinner** , and we thought we were looking for a window.	the good **news** is that the **person** who said that the teacher was **ann coulter** .
	S2D	it was the **time** for **dinner** , and we were looking for a **UNK** pilot.	the good motivator is that the **person** who said that was **ann coulter** .
	S2D+L	it was **time** for **dinner** , and we were looking for a **UNK**.	the good **news** is that the **person** who said that was **ann coulter** .
2	src	es war **zeit** zum **UNK** und wir hielten ausschau nach einem **UNK**.	die gute **UNK** ist , dass die **UNK** , die das gesagt hat **ann coulter** war .
	S2S	and it was **time** for the time time , and we thought we looked at a window search.	the good **news** is that the **UNK** that the UNK , which was **ann coulter** .
	S2D	it was the second **time** , and we would look for a **UNK** pilot.	the good motivator is that the **UNK group** who said that was **ann coulter**.
	S2D+L	it was a **time** for **UNK** , and we were looking for a **UNK**.	the good **news** is that the **people** who said that was **ann coulter** .
3	src	es war **UNK** zum **UNK** und wir hielten ausschau nach einem **UNK**.	die gute **UNK** ist , dass die **UNK** , die das gesagt hat **UNK** war .
	S2S	UNK was a time time and UNK , looking for a UNK look for a look at a war.	UNK is not a cop 's needs to be ozzie good to good good , which is that the UNK UNK that said that it was said .
	S2D	it was **time** of time and guerrilla .	the good motivator is that the **UNK** planner that said this was a **UNK** video.
	S2D+L	it was **time** for the tone and night to watch the connection .	the good **news** is that the **people** that said that was mandated .

Table 6: Unknown noun experiment samples. Substituted and correct nouns are marked in **Bold**, while incorrect elements are marked in underline. Examples shown are: no UNK; 1 UNK; 2 UNKs; 3 UNKs. When there are no unknown tokens, all three compared models are able to produce reasonably good if not identical translations. When there is only one UNK token, Seq2DRNN often does not use the context to predict an appropriate word or phrase. In contrast, both the Seq2Seq and Seq2DRNN+SynC were able to correctly predict that *die gute UNK ist* could be translated to *the good news is*. When there are 2 UNK tokens in the source sentence, Seq2Seq produces more incorrect predictions, Seq2DRNN makes some mistakes, while Seq2DRNN+SynC is able to get the most parts correct. Finally, when we replace 3 nouns, all models fail to some degree while Seq2Seq's output is the worst.

Model	F-measure
Baseline (Vinyals et al., 2015)	< 70
LSTM+AD (Vinyals et al., 2015)	88.3
Petrov (2010)	91.8
Dyer et al. (2016)	**92.4**
Seq2DRNN	89.4
Seq2DRNN+SynC	89.9

Table 7: Parser scores. Numbers from (Vinyals et al., 2015) are of non-ensemble models.

	Unlabelled		
	Prec.	Rec.	F1
Seq2DRNN	96.87	96.93	96.90
Seq2DRNN+SynC	96.43	95.89	96.16

Table 8: IWSLT Translation result constituency unlabelled scores. Reference parse trees obtained using Stanford Parser.

	Labelled		
	Prec.	Rec.	F1
Seq2DRNN	91.63	91.69	91.66
Seq2DRNN+SynC	90.73	90.22	90.48

Table 9: IWSLT Translation result constituency labelled scores. Reference parse trees obtained using Stanford Parser.

parsing (Klein and Manning, 2003b).

The presence of SynC in the decoder influences parse tree construction: the Seq2DRNN+SynC F1 score is comparable but lower than Seq2DRNN.

4 Related Work

Recent research shows that modelling syntax is useful for various neural NLP tasks. Dyer et al. (2015, 2016); Vinyals et al. (2015); Luong et al. (2016) have works on language modelling and parsing, Tai et al. (2015) on semantic analysis, and Zhang et al. (2016) on sentence completion, etc.

Eriguchi et al. (2017) showed that NMT model can benefit from neural syntactical parsing models. Choe and Charniak (2016) showed that a neural parsing problem shares similarity to neural language modelling problem, which forms a building block of an NMT system. We can then make the assumption that structural syntactic information utilised in neural parsing models should be able to aid NMT, which is shown to be true here.

Zhang et al. (2016) proposed TreeLSTM which is another structured neural decoder. TreeLSTM is not only structurally more complicated but also uses external classifiers. Dong and Lapata (2016) also proposed a sequence-to-tree (Seq2Tree) model for question answering. Both of these models are not designed for NMT and lack a language model. While operate from top-to-bottom like Seq2DRNN(+SynC), TreeLSTM and Seq2Tree produce components that lack sequential continuity which we have shown to be non-negligible for language generation.

Aharoni and Goldberg (2017), Wu et al. (2017), and Eriguchi et al. (2017) experimented with NMT models that utilise target side structural syntax. Aharoni and Goldberg (2017) treated constituency trees as sequential strings (linearised-tree) and trained a Seq2Seq model to produce such sequences. Wu et al. (2017) proposed SD-NMT, which models dependency syntax trees by adding a shift-reduce neural parser to a standard RNN decoder. Eriguchi et al. (2017) in addition to Wu et al. (2017)'s work, proposed NMT+RNNG which uses a modified RNNG generator (Dyer et al., 2016) to process dependency instead of constituency information as originally proposed by Dyer et al. (2016), making it consequently a StackLSTM sequential decoder with additional RNN units so it is still a bottom-up tree-structured decoder rather than a top-down decoder like ours. Nevertheless, all of these research showed that target side syntax could improve NMT systems. We believe these models could also be augmented with SynC connections (with NMT+RNNG one has to instead use constituency information).

5 Conclusions

We propose an NMT model that utilises target side constituency syntax with a strictly top-down tree-structured decoder using Doubly-Recurrent Neural Networks (DRNN) incorporated into an encoder-decoder NMT model. We propose a new way of modelling language generation by establishing additional clause-based syntactic connections called SynC. Our experiments show that our proposed models can outperform a strong sequence to sequence NMT baseline and several rival models and do parsing competitively.

In the future we hope to incorporate source side syntax into the model. We plan to explore the applications of SynC in NMT with more structured attention mechanisms, and potentially a hybrid phrase-based NMT systems with SynC, in which the model can benefit from SynC to be more extensible when handling larger lexicons.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful remarks. The research was also partially supported by the Natural Sciences and Engineering Research Council of Canada grants NSERC RGPIN-2018-06437 and RGPAS-2018-522574 and a Department of National Defence (DND) and NSERC grant DGDND-2018-00025 to the third author.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting on Association for Computational Linguistics*.

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *International Conference on Learning Representations*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Empirical Methods in Natural Language Processing*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

KyungHyun Cho, Aaron C. Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. In *IEEE Transactions on Multimedia Special Issue on Deep Learning for Multimedia Computing*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and

Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing*.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Empirical Methods in Natural Language Processing*, pages 2331–2336.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54rd Annual Meeting on Association for Computational Linguistics*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting on Association for Computational Linguistics*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of North American Chapter of the Association for Computational Linguistics*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *54th Annual Meeting of the Association for Computational Linguistics*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *55th Annual Meeting of the Association for Computational Linguistics*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 961–968. Association for Computational Linguistics.

Christoph Goller and Andreas Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *In Proc. of the ICNN-96*, pages 347–352. IEEE.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 944–952, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Empirical Methods in Natural Language Processing*, pages 1700–1709.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Dan Klein and Christopher D. Manning. 2003a. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003b. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *International Conference on Learning Representations*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017a. Dynet: The dynamic neural network toolkit. In *arXiv preprint arXiv:1701.03980*.

Graham Neubig, Yoav Goldberg, and Chris Dyer. 2017b. On-the-fly operation batching in dynamic computation graphs. In *CoRR*, volume abs/1705.07860.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Slav Petrov. 2010. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 19–27, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *In Proceedings of the First Conference on Machine Translation*.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically guided neural machine translation. In *Proceedings of Association for Computational Linguistics*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting on Association for Computational Linguistics*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2773–2781, Cambridge, MA, USA. MIT Press.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–707. Association for Computational Linguistics.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Tree recurrent neural networks with application to language modeling. In *North American Chapter of the Association for Computational Linguistics*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 138–141. Association for Computational Linguistics.

A Additional Translation Samples (IWSLT2017 German-English)

The samples here are from our IWSLT2017 German-English testset. We compared the performances of all our proposed models as well as the baseline Seq2Seq model in Table 10.

We provide an additional example of our attention module visualisation in Figure 6 and for parser in Figure 7.

Source	1 m ist das höchste, was ich gesehen habe.
Literal	1 m is the highest that i've seen.
Reference	thirty-nine inches is the tallest structure i've seen.
Seq2Seq	and the highest thing **is i've seen**.
Seq2DRNN	one is the highest thing i've seen at that.
Seq2DRNN+SynC	feet is the highest thing i've seen.
Source	ich weiß nicht . sie wollten in die zeit zurck , bevor es autos gab oder twitter oder amerika sucht den superstar.
Literal	i dont know. they want to go back in time, before there were automobiles or twitter or america looking for superstar.
Reference	i dont know. they want to go back before there were automobiles or twitter or american idol.
Seq2Seq	i don't know. they **were in the days**, when **they were** cars before **cars or the earnings, or america, and the country**.
Seq2DRNN	i don't want to know before time, they wanted to go back before **the cars before they were cars or americans**.
Seq2DRNN+SynC	i don't know. they wanted to go back in time, **they wanted to go back into the before,** before there had cars or twitter **visitors**.

Table 10: Translation Samples. *Gold* is the reference, and *Literal* is produced by a bilingual German-English speaker. The reason we include the literal translation is that sometimes the reference translation from the corpus can have additional components or be non-literal translations.

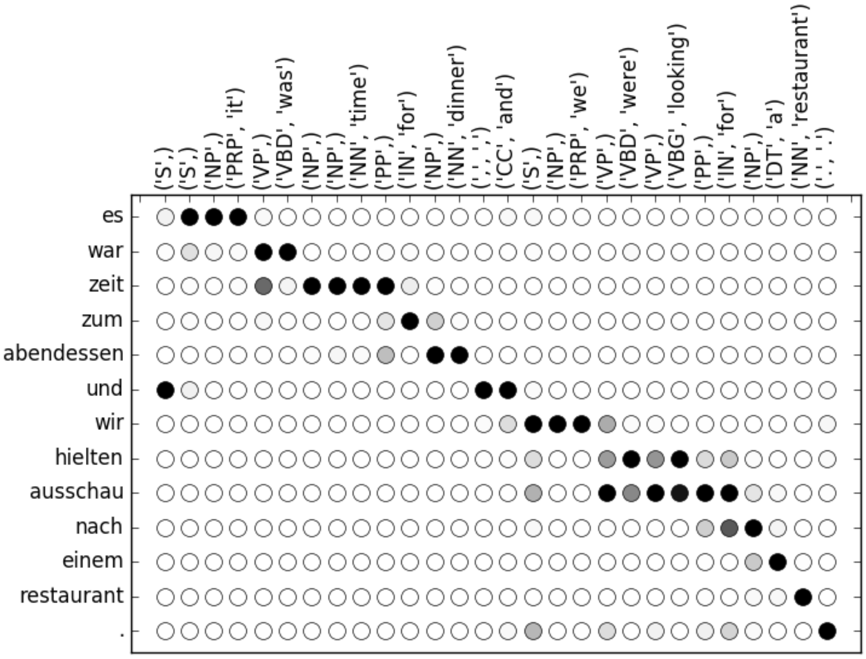

Figure 6: Attention visualisation (Seq2DRNN+SynC). Darker colour means higher attention weight as defined in Equ 9. The sentence is randomly selected from our IWSLT experiment.

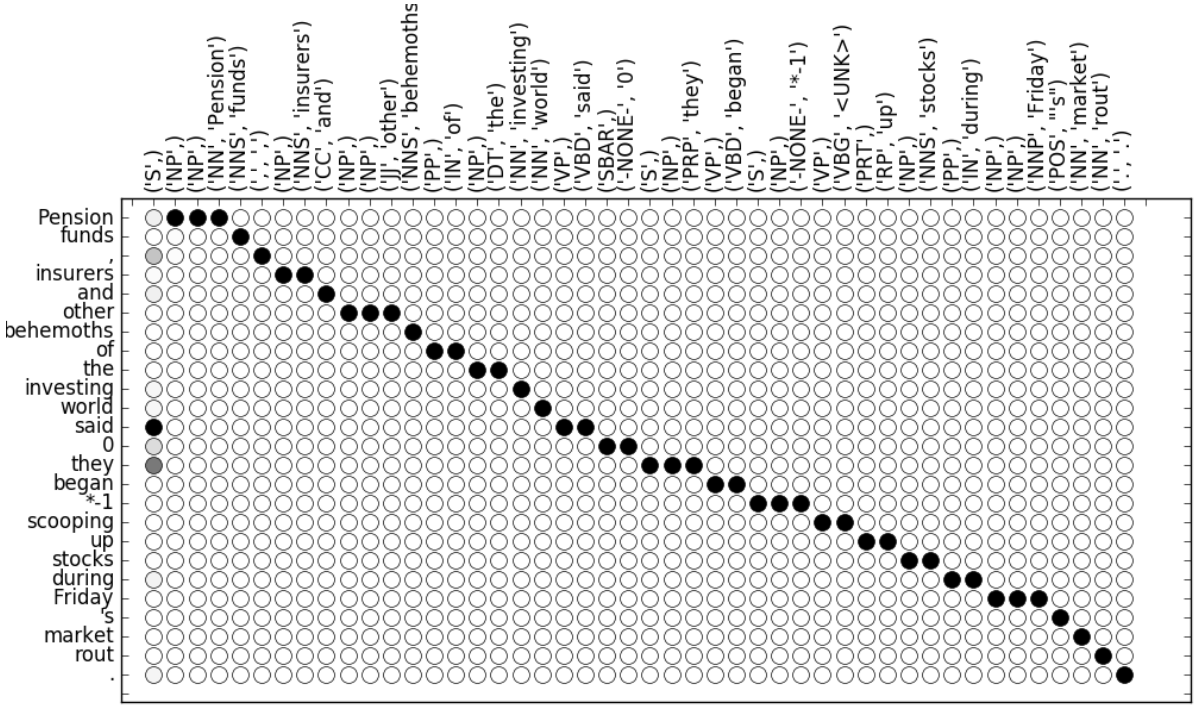

Figure 7: Parser attention visualisation (Seq2DRNN+SynC). Darker colour means higher attention weight as defined in Equ 9. The sentence is randomly selected from our PennTreebank experiment.

XL-NBT: A Cross-lingual Neural Belief Tracking Framework

Wenhu Chen[1], **Jianshu Chen**[2], **Yu Su**[3], **Xin Wang**[1], **Dong Yu**[2]
Xifeng Yan[1] **and William Wang**[1]
University of California, Santa Barbara, CA, USA[1]
Tencent AI Lab, Bellevue, WA, USA[2]
The Ohio State University, Columbus, Ohio[3]
`{wenhuchen,xwang,xyan,william}@cs.ucsb.edu, su.809@osu.edu`
`jianshuchen@tencent.com, dongyu@ieee.org`

Abstract

Task-oriented dialog systems are becoming pervasive, and many companies heavily rely on them to complement human agents for customer service in call centers. With globalization, the need for providing cross-lingual customer support becomes more urgent than ever. However, cross-lingual support poses great challenges—it requires a large amount of additional annotated data from native speakers. In order to bypass the expensive human annotation and achieve the first step towards the ultimate goal of building a universal dialog system, we set out to build a cross-lingual state tracking framework. Specifically, we assume that there exists a source language with dialog belief tracking annotations while the target languages have no annotated dialog data of any form. Then, we pre-train a state tracker for the source language as a teacher, which is able to exploit easy-to-access parallel data. We then distill and transfer its own knowledge to the student state tracker in target languages. We specifically discuss two types of common parallel resources: bilingual corpus and bilingual dictionary, and design different transfer learning strategies accordingly. Experimentally, we successfully use English state tracker as the teacher to transfer its knowledge to both Italian and German trackers and achieve promising results.

1 Introduction

Over the past few years, we have witnessed the burgeoning of real-world applications of dialog systems, with many academic, industrial, and startup efforts racing to lead the widely-believed next-generation human-machine interfaces. As a result, numerous task-oriented dialog systems such as virtual assistants and customer conversation services were developed (Wen et al., 2015; Rojas-Barahona et al., 2017; Bordes and Weston, 2017; Williams et al., 2017; Li et al., 2017), with Google Duplexbeing the most recent example.

With the rapid process of globalization, more countries have observed growing populations of immigrants, and more companies have moved forward to develop their overseas business sectors. To provide better customer service and bring down the cost of labor at call centers, the development of universal dialog systems has become a practical issue. A straightforward strategy is to separately collect training data and train dialog systems for each language. However, it is not only tedious but also expensive. Two settings naturally arise for more efficient usage of the training data: (1) Multi-lingual setting: we annotate data for multiple languages and train a single model, with possible innovations on joint training. (2) Cross-lingual setting: we annotate data and train a model for only one (popular) language, and transfer the learned knowledge to other languages. Here we are interested in the second case, and the important research question we ask is: *How can we build cross-lingual dialog systems that can support less popular, low- or even zero-resource languages*?

As an initial step towards cross-lingual dialog systems, we focus on the cornerstone of dialog systems – *dialog state tracking* (DST), or *belief tracking*, a key component for understanding user inputs and updating *belief state*, i.e., a system's internal representation of the state of conversation (Young et al., 2010). Based on the perceived belief state, the dialog manager can decide which action to take, and what verbal response to generate (Precup and Teh, 2017; Bordes and Weston, 2017).

DST models require a considerable amount of annotated data for training (Henderson et al., 2014b; Mrksic et al., 2015, 2017). For a common dialog shown in Figure 1, a typical data acquisition process (Rojas-Barahona et al., 2017) not only requires two human users to converse for multiple

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 414–424
Brussels, Belgium, October 31 - November 4, 2018.

turns but also requires annotators to identify user's intention in each turn. Such two-step annotation is very expensive, especially for rare languages.

We study the novel problem of *cross-lingual DST*, where one leverages the annotated data of a source language to train DST for a target language with *zero* annotated data (Figure 1); no conversation dialog or dialog state annotation is available for the target language. In order to deal with this zero-resource challenging scenario, we first decouple the state-of-the-art neural belief tracker framework (Mrksic et al., 2017) into sub-modules, namely *utterance encoder*, *context gate*, and *slot-value decoder*. By introducing a teacher-student framework, we are able to transfer knowledge across languages module by module, following the divide-and-conquer philosophy. Requiring no target-side dialog data, our method relies on other easy-to-access parallel resources to understand the connection between languages. Depending on the popularity and availability of target language resources, we study two kinds of parallel data: bilingual corpus and bilingual dictionary, and we respectively design two transfer learning strategies.

We use the popular *Wizard-of-Oz* (Rojas-Barahona et al., 2017) dataset as our DST benchmark to evaluate the effectiveness of our cross-lingual transfer learning. We specify English as the source (primary) language and two different European languages (German and Italian) as our zero-annotation target languages. Compared with an array of alternative transfer learning strategies, our cross-lingual DST models consistently achieve promising results in both scenarios for both zero-annotation languages. To ensure reproducibility, we release our code, training data and parallel resources in the github[1]. Our main contributions are three-fold:

- Towards building cross-lingual dialog systems, we are the first to study the cross-lingual dialog state tracking problem.
- We systematically study different scenarios for this problem based on the availability of parallel data and propose novel transfer learning methods to tackle the problem.
- We empirically demonstrate the efficacy of the proposed methods, showing that our methods can accurately track dialog states for languages with zero annotated data.

2 Related Work

2.1 Dialog State Tracking

Broadly speaking, the dialog belief tracking algorithms can be divided into three families: 1) hand-crafted rules 2) generative models, and 3) maximum-entropy model (Metallinou et al., 2013). Later on, many deep learning based discriminative models have surged to replace the traditional strategies (Henderson et al., 2014a; Mrksic et al., 2017; Williams et al., 2016) and achieved state-of-the-art results on various datasets. Though the discriminative models are reported to achieve fairly high accuracy, their applications are heavily restricted by the domain, ontology, and language. Recently, a pointer network based algorithm (Xu and Hu, 2018) and another multi-domain algorithm (Rastogi et al., 2017) have been proposed to break the ontology and domain boundary. Besides, (Mrkšić et al., 2017) has proposed an algorithm to train a unified framework to deal with multiple languages with annotated datasets. In contrast, our paper focuses on breaking the language boundary and transfer DST knowledge from one language into other zero-annotation languages.

2.2 Cross-Lingual Transfer Learning

Cross-lingual transfer learning has been a very popular topic during the years, which can be seen as a transductive process. In such process, the input domains of the source and target are different (Pan and Yang, 2010) since each language has its own distinct lexicon. By discovering the underlying connections between the source and target domain, we could design transfer algorithms for different tasks. Recently, algorithms have been successfully designed for POS tagging (Zhang et al., 2016; Kim et al., 2017), NER (Pan et al., 2017; Ni et al., 2017) as well as image captioning (Miyazaki and Shimizu, 2016). These methods first aim at discovering the relatedness between two languages and separate language-common modules from language-specific modules, then resort to external resources to transfer the knowledge across the language boundary. Our method addresses the transfer learning using a teacher-student framework and proposes to use the teacher to gradually guide the student to make more proper decisions.

---

[1] `https://github.com/wenhuchen/Cross-Lingual-NBT`

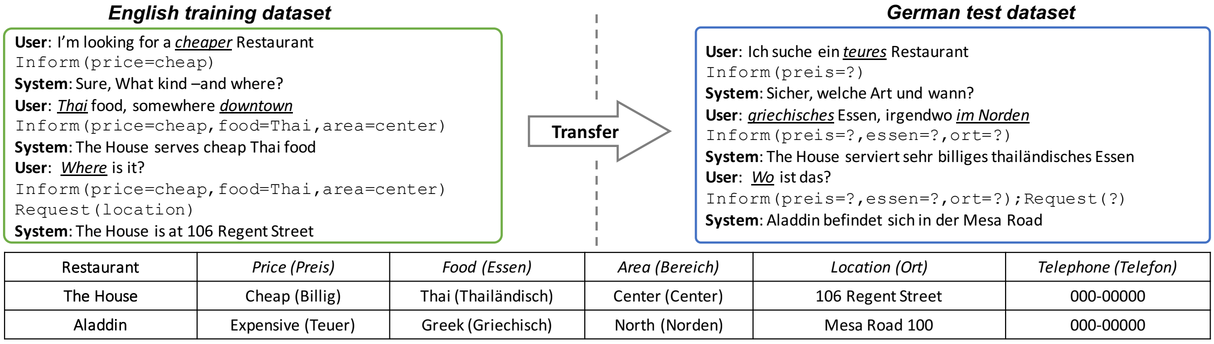

User: I'm looking for a *cheaper* Restaurant
`Inform(price=cheap)`
System: Sure, What kind –and where?
User: *Thai* food, somewhere *downtown*
`Inform(price=cheap, food=Thai, area=center)`
System: The House serves cheap Thai food
User: *Where* is it?
`Inform(price=cheap, food=Thai, area=center)`
`Request(location)`
System: The House is at 106 Regent Street

User: Ich suche ein *teures* Restaurant
`Inform(preis=?)`
System: Sicher, welche Art und wann?
User: *griechisches* Essen, irgendwo *im Norden*
`Inform(preis=?, essen=?, ort=?)`
System: The House serviert sehr billiges thailändisches Essen
User: *Wo* ist das?
`Inform(preis=?, essen=?, ort=?); Request(?)`
System: Aladdin befindet sich in der Mesa Road

Restaurant	*Price (Preis)*	*Food (Essen)*	*Area (Bereich)*	*Location (Ort)*	*Telephone (Telefon)*
The House	Cheap (Billig)	Thai (Thailändisch)	Center (Center)	106 Regent Street	000-00000
Aladdin	Expensive (Teuer)	Greek (Griechisch)	North (Norden)	Mesa Road 100	000-00000

Figure 1: Cross-lingual transfer learning for dialog state tracking, where the underlying database (the table above) is shared across languages. The source language has annotated dialogs and the ground truth states, but the target language has neither dialogs nor ground truth states (only a testing dataset for evaluation).

3 Problem Definition

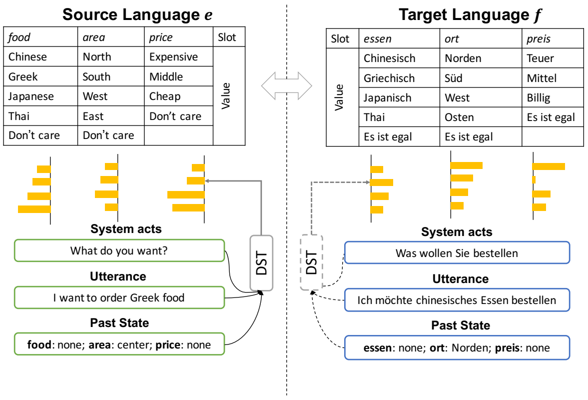

food	*area*	*price*	Slot
Chinese	North	Expensive	Value
Greek	South	Middle	
Japanese	West	Cheap	
Thai	East	Don't care	
Don't care	Don't care		

Slot	*essen*	*ort*	*preis*
Value	Chinesisch	Norden	Teuer
	Griechisch	Süd	Mittel
	Japanisch	West	Billig
	Thai	Osten	Es ist egal
	Es ist egal	Es ist egal	

What do you want?
I want to order Greek food
food: none; **area**: center; **price**: none

Was wollen Sie bestellen
Ich möchte chinesisches Essen bestellen
essen: none; **ort**: Norden; **preis**: none

Figure 2: Cross-lingual DST structure, the ontology and database between multiple languages are shared.

The dialog states are defined as a set of search constraints (i.e. informable slots or goals) that the user specified through the dialog and a set of attribute questions regarding the search results (i.e. requestable slots or requests). The objective of dialog state tracking (DST) is to predict and track the user intention (i.e., the values of the aforementioned slots) at each time step based on the current user utterance and the entire dialog history. As shown in Figure 2, for each slot, the DST computes an output distribution of the candidate values using three inputs: (i) *system response* a_t, which is the sentence generated by the system, (ii) *utterance* u_t, which is the sentence from the user, and (iii) *previous state*, which denotes the selected slot-value pairs. We define the ontology of the dialog system to be the set of all the possible words the dialog slot and value can take. In this paper, we are interested in learning a cross-lingual DST. Specifically, we assume that the DST for the *source* language has access to a human-annotated training dataset D while the DSTs for the *target* languages do not have access to annotated data in other languages except for testing data. We here mainly consider two different types of parallel resources to assist the transfer learning:

(1) Bilingual Corpus, where abundant bilingual corpora exist between the source and the target languages. This is often the case for common language pairs like German, Italian, and French, etc.

(2) Bilingual Dictionary, where public bilingual dictionaries exist between the source and the target languages, but large-scaled parallel corpus are harder to obtain. This can be the case for rarer languages like Finnish, Bulgarian, etc.

Furthermore, we assume that all the languages share a common multi-lingual database, whose column/row names and entry values are stored via multiple languages (see the database in Figure 1). That is, the ontology of dialog among different languages is known with a one-to-one mapping between them (e.g., greek=griechisch=greco, food=essen=cibo). Based on that, we could construct a mapping function M to associate the ontology terms from different languages with pre-designed language-agnostic concepts: for example, $M(foods) = M(Essen) = M(Cibo) =$ FOOD. We illustrate our problem definition in Figure 2.

4 Decoupled Neural Belief Tracker

We design our cross-lingual DST on top of the state-of-the-art Neural Belief Tracker (NBT) (Mrksic et al., 2017), which demonstrates many advantages (no hand-crafted lexicons, no linguistic knowledge required, etc). These nice properties are essential for our cross-lingual DST design because we are pursuing a general and simple framework regardless of the language properties. In short, NBT consists of a neural network that computes the matching score for every candidate slot-value pair (c_s, c_v) based on the following three inputs: (i) the system dialog acts $a_t = (t_q, t_s, t_v)$,[2] (ii) the user utterance u_t, and (iii) the candidate slot-value pair. And it identifies the user intents by evaluating the scores for all the slot-value pairs (see Figure 3). With a slight abuse of notation, we still use $c_s, c_v, t_s, t_v, t_q \in \mathbb{R}^H$ to denote the vector representations of themselves, where H is the embedding dimension. We will use pre-trained embedding vectors in our cross-lingual NBT, just like the original NBT and they will be fixed during training. To enable cross-lingual transfer learning, we first re-interpret the architecture of the original NBT by decomposing it into three components:

Utterance Encoding The first component is an utterance encoder, which maps the utterance $u_t = \{w_1, w_2, \cdots, w_N\}$ of a particular language into a semantic representation vector $r(u_t) \in \mathbb{R}^H$, where $w_i \in \mathbb{R}^H$ is the word vector for the i-th token and N is the length of the utterance. Note that the dimension of the semantic vector $r(u_t)$ is the same as that of the word vector. We implement the encoder using the same convolutional neural network (CNN) as the original NBT, with a slight modification of adding a top batch normalization layer. We will explain this change in section 5.

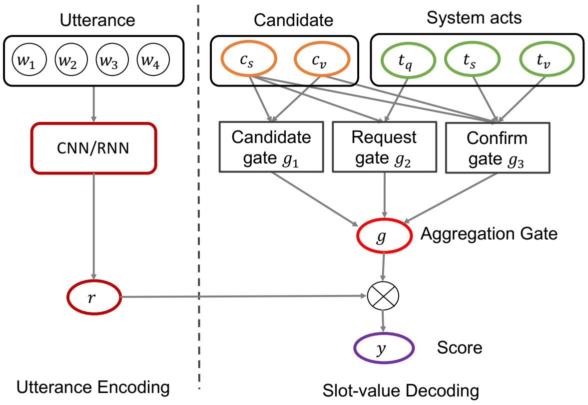

Figure 3: Our implementation of baseline NBT, slightly modified from (Mrksic et al., 2017).

Context Gate The second part is the context gate, which takes the system acts $a_t = (t_q, t_s, t_v)$ and the candidate slot-value pair (c_s, c_v) as its inputs and filter out the desired information from the encoded utterance. The context gate g is a sum of three separate gates:

$$g(c_s, c_v, a_t) = g_1 + g_2 + g_3 \tag{1}$$

where the individual gates are defined as:

$$\begin{aligned} g_1 &= \sigma(W_c^s(c_s + c_v) + b_c^s) \\ g_2 &= (c_s \cdot W_t^q t_q) \odot [1, \cdots, 1]^H \\ g_3 &= (c_s \cdot W_t^s t_s)(c_v \cdot W_t^v t_v) \odot [1, \cdots, 1]^H \end{aligned} \tag{2}$$

where $W_c^s, W_t^q, W_t^s, W_t^v \in \mathbb{R}^{H \times H}$ are the weight matrices, and $\odot$ and $\cdot$ denote the Hadamard product and the inner product, respectively. The three gates $g_1 \in \mathbb{R}^H, g_2 \in \mathbb{R}^H, g_3 \in \mathbb{R}^H$ model the relevance between the candidate slot and value, the system request and the system confirms, respectively. The transformation matrices W_t^q, W_t^s, W_t^v are added to the original NBT to increase the model flexibility of the gates.

Slot-Value Decoding The final component is a slot-value decoder, which predicts the score y of a given slot-value pair using the filtered information from the utterance representation r as:

$$y(c_s, c_v, u_t, a_t) = W_y^T [r(u_t) \odot g(c_s, c_v, a_t)] \tag{3}$$

where $W_y \in \mathbb{R}^{H \times 1}$ is the weight vector. The above expression computes the score for the slot-value pair based on the information from the current turn. We combine it with the information from previous turns to get the final score:

$$\begin{aligned} \hat{y}(c_v | u_t, a_t, c_s) =& \lambda y(c_s, c_v, u_t, a_t) + \\ & (1 - \lambda) \hat{y}(c_s, c_v, u_{t-1}, a_{t-1}) \end{aligned} \tag{4}$$

here λ is a combination weight. For each given slot c_s, NBT selects the single highest value for informable slots and selects all values above a certain threshold for request slots. Here we replace the multi-layer perceptron in the orginal NBT by a linear output layer (to be explained in section 5).

5 Cross-lingual Neural Belief Tracker

In this section, we develop a cross-lingual Neural Belief Tracker (XL-NBT) that distills knowledge from one NBT to another using a teacher-student framework. We assume the ontology mapping M is known a priori (see Figure 3). XL-NBT uses language-specific utterance encoder and context gate for each input language while sharing a common (language-agnostic) slot-value decoder across different languages (see Figure 3).

---

[2] t_q represents the system request, t_s, t_v represents the system confirmation. If the system wants to request some information from the user by asking "what's your favorite area?", then NBT sets t_q="AREA". If the system wants to confirm some information from a user by asking "should I try Persian restaurants in the north?" then NBT sets t_s, t_v="area, north".

The key idea is to optimize the language-specific components of the student network (NBT of the target language) so that their outputs are language-agnostic. This is achieved by making these outputs close to that of the teacher network (NBT of the source language), as we detail below.

5.1 Teacher-Student Framework

We are given a well-trained NBT for a source language e, and we want to learn an NBT for a target language f without any annotated training data. Therefore, we cannot learn the target-side NBT from standard supervised learning. Instead, we use a teacher-student framework to distill the knowledge from the source-side NBT (teacher network) into the target-side NBT (student network) (see Figure 4). Let $x_e \triangleq (c_s^e, c_v^e, u_t^e, a_t^e)$ be the input to the teacher network and let $x_f \triangleq (c_s^f, c_v^f, u_t^f, a_t^f)$ be the associated input to the student network. The standard teacher-student framework trains the student network by minimizing

$$\hat{J}_1 = \sum_{x_e, x_f} ||y(c_s^e, c_v^e, u_t^e, a_t^e) - y(c_s^f, c_v^f, u_t^f, a_t^f)||^2 \qquad (5)$$

where $y(c_s^e, c_v^e, u_t^e, a_t^e)$ and $y(c_s^f, c_v^f, u_t^f, a_t^f)$ denote the scores by the teacher and the student networks, respectively, and the slot-value pairs satisfy $M(c_v^f) = M(c_v^e)$ and $M(c_s^f) = M(c_s^e)$. However, the target-side inputs $(c_s^f, c_v^f, u_t^f, a_t^f)$ parallel to $(c_s^e, c_v^e, u_t^e, a_t^e)$ are usually not available in cross-lingual DST, and, even worse, the target-side utterance u_t^e is not available. We may have to generate synthetic input data for the student network or leverage external data sources. It is relatively easy to use the mapping $M(\cdot)$ to generate $(c_s^f, c_v^f, a_t^f))$ (i.e., the inputs of the target-side context gate) from the (c_s^e, c_v^e, a_t^e). But it is more challenging to obtain the parallel utterance data u_t^f from u_t^e). Therefore, we have to leverage external bilingual data sources to alleviate the problem. However, the external bilingual data are usually not in the same domain as the utterance, and hence they are not aligned with the slot-value pair and system acts (i.e., (c_s^e, c_v^e, a_t^e) or (c_s^f, c_v^f, a_t^f)). For this reason, we cannot perform the knowledge transfer by minimizing the cost (5). Instead, we need to develop a new cost function where the utterance is not required to be aligned with the slot-value pair and the system acts. To this end, let $g^e = g_e(c_s^e, c_v^e, a_t^e)$ and $g^f = g_f(c_s^f, c_v^f, a_t^f)$. And we substitute (3) into (5) and get:

$$\begin{aligned}
\hat{J}_1 &\leq ||W_y||^2 \sum_{c_v^f, c_v^e} ||r_e \odot g^e - r_f \odot g^f||^2 \\
&= ||W_y||^2 \sum_{c_v^f, c_v^e} ||g^e \odot (r_e - r_f) + r_f \odot (g^e - g^f)||^2 \\
&\leq ||W_y||^2 \sum_{c_v^f, c_v^e} ||g^e||^2 ||r_e - r_f||^2 + ||r_f||^2 ||g^e - g^f||^2
\end{aligned}$$

where $r_e = r_e(u_t^e)$ and $r_f = r_f(u_t^f)$. As we mentioned earlier, the weight W_y in the slot-value decoder is shared between the student and the teacher networks and will not be updated. The teacher-student optimization only adjusts the weights related to the language-specific parts in Figure 3 (i.e., utterance encoding and context gating). Therefore, the shared weight $||W_y||$ is seen as a constant. Furthermore, $\sum_{c_v^f, c_v^e} ||g^e||^2$ can be seen as a constant since the teacher gate is fixed. Since we use batch normalization layer to normalize the encoder output (described in Figure 3), $||r_f(u_t^f)||^2$ can also be treated as a constant C_2. Therefore, we formally write the upper bound of $\hat{J}_1$ as our surrogate cost function J:

$$J = C_1 ||r_e(u_t^e) - r_f(u_t^f)||^2 + C_2 \sum_{c_v^f, c_v^e} ||g^e - g^f||^2 \qquad (6)$$

The surrogate cost has successfully decoupled utterance encoder with context gate, and we use J_r and J_g to measure the encoder matching cost and the gate matching cost, respectively.

$$\begin{aligned}
J_r &= ||r_e(u_t^e) - r_f(u_t^f)||^2 \\
J_g &= \sum_{c_v^f, c_v^e} ||g^e - g^f||^2
\end{aligned} \qquad (7)$$

The encoder cost J_r is optimized to distill the knowledge from the teacher encoder to student encoder while gate cost J_g is optimized to distill the knowledge from teacher gate to student gate. This objective function successfully *decouples* the optimization of encoder and gate, thus we are able to optimize J_r and J_g separately from different data sources. Recall that we can easily simulate the target-side system acts, slot-value pairs (c_s^f, c_v^f, a^f) by using the ontology mapping M. Therefore, optimizing J_g is relatively easy. Formally, we write the gate matching cost as follows:

$$J_g = \sum_{\substack{a_t^e, c_s^e, c_v^e \\ a_t^f, c_s^f, c_v^f}} ||g_e(c_s^e, c_v^e, a_t^e) - g_f(c_s^f, c_v^f, a_t^f)||^2 \qquad (8)$$

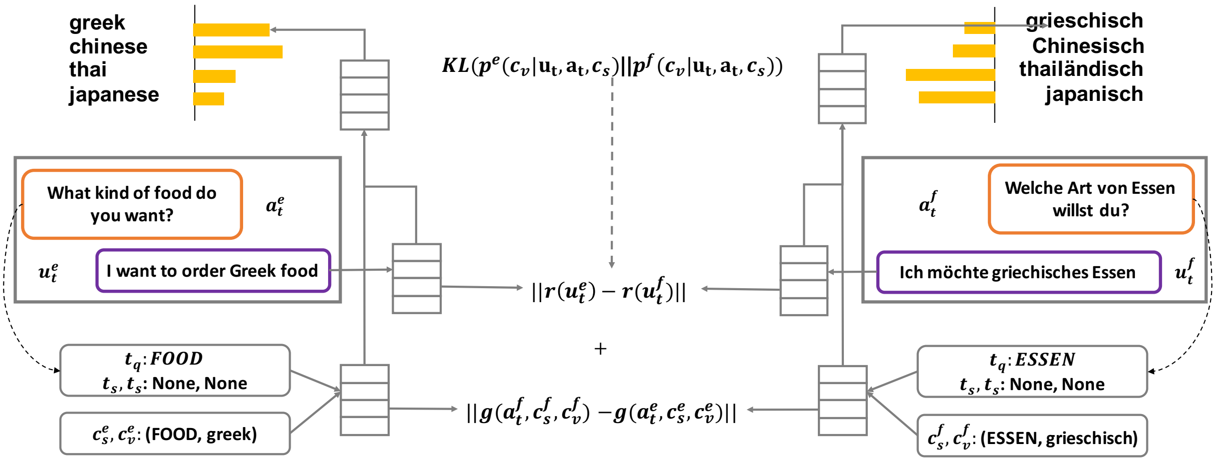

Figure 4: Teacher-Student Framework for cross-lingual transfer learning. The dotted line denotes the imaginary utterances, which expresses the same intention as the source side.

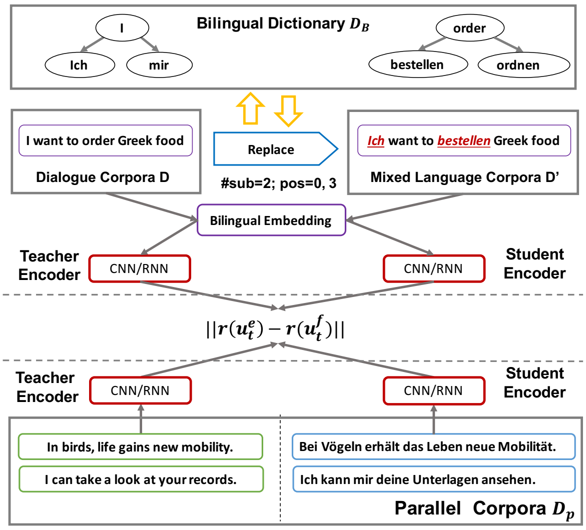

Figure 5: XL-NBT-C and XL-NBT-D for two scenarios

However, exact optimization of J_r is difficult and we have to approximate it using external parallel data. We consider two kinds of external resources (bilingual corpus and bilingual dictionary) in the sections 5.2-5.3 (see Figure 5 for the main idea).

5.2 Bilingual Corpus (XL-NBT-C)

In our first scenario, we assume there exists a parallel corpus D_p consisting of sentence pairs from the source language and the target language. In this case, the cost function (6) is approximated by

$$J = \mathop{\mathbb{E}}_{(m_e, m_f) \in D_p} ||r_e(m_e) - r_f(m_f)||^2 + \alpha J_g \qquad (9)$$

where α is the balancing factor and J_g is defined in (6). The cost function (9) is minimized by stochastic gradient descent. At test time, we switch the encoder to receive target language inputs.

5.3 Bilingual Dictionary (XL-NBT-D)

In the second scenario, we assume there exists no parallel corpus but a bilingual dictionary D_B that defines the correspondence between source words and target words (a one-to-many mapping $\{w : M_D(w)\}$). Likewise, it is infeasible to optimize the exact encoder cost J_r due to the lack of target-side utterances. We propose a word replacement strategy (to be described later) to generate synthetic parallel sentence $\hat{u}_t^f$ of "mixed" language. Then, we use the generated target parallel sentences to approximate the cost (6) by

$$J_r = \mathop{\mathbb{E}}_{u_t \in D} ||r_e(u_t^e) - r_f(\hat{u}_t^f)||^2 + \alpha J_g \qquad (10)$$

where α is the balancing factor. For word replacement, we first decide the number of words N_w to be replaced, then we draw N_w positions randomly from the source utterance and substitute the corresponding word w_i with their target word synonym from $M_D(w)$ based on the context as follows:

$$\begin{aligned} jp(N_w = i) &= \frac{\exp(-i/\tau)}{\sum_{i' < N} \exp(-i'/\tau)} \\ p(\hat{w}) &= \frac{\hat{w} \cdot h_{\hat{w}}}{\sum_{w' \in M(w_i)} w' \cdot h_{\hat{w}}} \end{aligned} \qquad (11)$$

where $h_{\hat{w}} = \sum_{k=-2:k\neq 0}^{2} w_{i+k}$ represents the context vector and N denotes the utterance length. The context similarity of context and the target-side synonym can better help us in choosing the most appropriate candidate from the list. In our following experiments, we adjust the temperature of τ to control the aggressiveness of replacement.

6 Experiments

6.1 Dataset

The Wizard of Oz (WOZ) (Rojas-Barahona et al., 2017) dataset is used for training and evaluation, which consists of user conversations with task-oriented dialog systems designed to help users find suitable restaurants around Cambridge, UK. The corpus contains three informable (i.e. goal-tracking) slots: FOOD, AREA, and PRICE. The users can specify values for these slots in order to find which best meet their criteria. Once the system suggests a restaurant, the users can ask about the values of up to eight requestable slots (PHONE NUMBER, ADDRESS, etc.). Multilingual WOZ 2.0 (Mrksic et al., 2017) has expanded this dataset to include more dialogs and more languages. The train, valid and test datasets for three different languages (English, German, Italian) are available online[3]. We use the English as source language where 600 dialogs are used for training, 200 for validation and 400 for testing. We use the German and Italian as the target language to transfer our knowledge from English DST system. In the experiments, we do not have access to any training or validation dataset for German and Italian, and we only have access to their testing dataset which is composed of 400 dialogs.

For external resource, we use the IWSLT2014 Ted Talk parallel corpus (Mauro et al., 2012) from the official website[4] for bilingual corpus scenario. In the IWSLT2014 parallel corpus, we only keep the sentences between 4 and 40 words and decrease the sentence pairs to around 150K. We use Panlex (Kamholz et al., 2014) as our data source and crawl translations for all the words appearing in the dialog datasets to build our bilingual dictionary. We specifically investigate two kinds of pre-trained embedding, and we use Glove (Pennington et al., 2014) as the monolingual embedding and MUSE (Conneau et al., 2017) as the bilingual embedding to see their impacts on the DST performance.

We split the raw DST corpus into turn-level examples. During training, we use the ground truth previous state V_{t-1} as inputs. At test time, we use the model searched states as the previous state to continue tracking intention until the end of the dialog. When the dialog terminates, we use two evaluation metrics introduced in Henderson et al. (2014a) to evaluate the DST performance: (1) *Goals*: the proportion of dialog turns where all the users search goal constraints were correctly identified. (2) *Requests*: similarly, the proportion of dialog turns where users requests for information were identified correctly. Our implementation is based on the NBT[5], the details of our system setting are described in the appendix.

6.2 Results

Here we highlight the baselines we use to compare with our cross-lingual algorithm as follows:
(1) Supervised: this baseline algorithm assumes the existence of annotated dialog belief tracking datasets, and it determines the upper bound of the DST model.
(2) w/o Transfer: this algorithm trains an English NBT, and then directly feeds target language into the embedding level as inputs during test time to evaluate the performance.
(3) Ontology-match: this algorithm directly uses exact string matching against the utterance to discover the perceived slot-value pairs, it directly assigns a high score to the appearing candidates.
(4) Translation-based: this system pre-trains a translator on the external bilingual corpus and then translates the English dialog and ontology into target language as "annotated" data, which is used to train the NBT in the target language domain (more details about the implementation, performance and examples are listed in the appendix).
(5) Word-By-Word (WBW): this system transforms the English dialog corpus into target language word by word using the bilingual dictionary, which is used to train the NBT in target side.
We demonstrate the results for our proposed algorithms and other competing algorithms in Table 2, from which we can easily conclude that that (i) our Decoupled NBT does not affect the performance, and (ii) our cross-lingual NBT framework is able to achieve significantly better accuracy for both languages in both parallel-resource scenarios.

Compare with Translator/WBW. With bilingual corpus, XL-NBT-C with pre-trained bilingual embedding can significantly outperform our Translator baseline (Klein et al., 2017). This is intuitive because the translation model requires

---

[3] `https://github.com/nmrksic/neural-belief-tracker/tree/master/data`

[4] `https://wit3.fbk.eu/mt.php?release=2014-01`

[5] `https://github.com/nmrksic/neural-belief-tracker`

Error Type	Examples
Modify Failure	Machine: I have two options that fit that description, golden wok Chinese restaurant and the Nirala which serves Indian food, do you have a preference? User: How about Nirala, whats the address and phone of that? Previous State: food=Chinese; Prediction: food=none; Groundtruth: food=Indian
Maintain Failure	Machine: there are $num places with a moderate price range. can you please tell me what kind of food you would like? User: well I want to eat in the north, whats up that way? Previous State: food=expensive; Prediction: food=none; Groundtruth: food=expensive
History Failure	Machine: Anatolia is located at $num bridge street city center. User: thank you goodbye! Previous State: food=Chinese; Prediction: food=Chinese;,Groundtruth: food=Turkish

Table 1: Here we show the frequent error types, the examples are translated to English for better understanding.

Language		German (student)		Italian (student)		English (teacher)	
Models		Goal	Request	Goal	Request	Goal	Request
w/ Supervised Dialog	NBT (Mrksic et al., 2017)	-	-	-	-	**0.84**	**0.91**
	Decoupled NBT (mono)	0.79	0.83	0.86	0.91	0.82	0.89
	Decoupled NBT (bilingual)	**0.80**	**0.84**	**0.88**	**0.91**	**0.84**	0.90
w/o Bilingual Data	w/o Transfer (mono)	0.15	0.10	0.15	0.11	-	-
	w/o Transfer (bilingual)	0.13	0.13	0.11	0.12	-	-
	Ontology Matching	0.24	0.21	0.23	0.21	-	-
w/ Bilingual Corpus	Translate (Klein et al., 2017)	0.41	0.42	0.48	0.51	-	-
	XL-NBT-C (mono)	0.48	0.54	0.65	0.60	-	-
	XL-NBT-C (bilingual)	**0.55**	**0.59**	**0.72**	**0.69**	-	-
w/ Bilingual Dictionary	Word-by-Word	0.22	0.25	0.25	0.27	-	-
	XL-NBT-D (mono)	0.14	0.15	0.23	0.22	-	-
	XL-NBT-D (bilingual)	**0.51**	**0.56**	**0.73**	**0.63**	-	-

Table 2: Experimental results for cross-lingual NBT and other baseline algorithms. All results are averaged over 5 runs. Here we use "mono" to refer to the experiments with pre-trained monolingual embedding, "bilingual" to refer to the experiments with pre-trained bilingual embedding.

both source-side encoding and target-side word-by-word decoding, while our XL-NBT only needs a bilingual source-encoding to align two vector space, which averts the compounded decoding errors. With the bilingual dictionary, the word-by-word translator is very weak and leading to many broken target sentences, which poses challenges for DST training. In comparison, our XL-NBT-D can control the replacement by adjusting its temperature to maintain the stability of utterance representation. Furthermore, for both cases, our teacher-student framework can make use of the knowledge learned in source-side NBT to assist its decision making, while translator-based methods learn from scratch.

Bilingual Corpus vs. Bilingual Dictionary. From the table, we can easily observe that bilingual corpus is obviously a more informative parallel resource to perform cross-lingual transfer learning. The accuracy of XL-NBT-D is lower than XL-NBT-C. We conjecture that our replacement strategy to generate "mixed" language utterance can sometimes break the semantic coherence and cause additional noises during the transfer process, which remarkably degrades the DST performance.

Monolingual vs. Bilingual embedding. From the table, we can observe that the bilingual embedding and monolingual embedding does not make much difference in supervised training. However, the gap in the bilingual corpus case is quite obvious. Monolingual embedding even causes the transfer to fail in a bilingual dictionary case. We conjecture that the bilingual word embedding already contain many alignment information between two languages, which largely eases the training of encoder matching objective.

German vs. Italian As can be seen, the transfer learning results for Italian are remarkably higher than German, especially for the "Goal" accuracy. We conjecture that it is due to German declension, which can produce many word forms. The very diverse word forms present great challenges for DST to understand its intention behind. Especially for the bilingual dictionary, German tends to have much longer replacement candidate lists than Italian, which introduces more noises to the replacement procedure.

Error Analysis Here we showcase the most frequent error types in subsection 6.1. From our observation, these three types of errors distribute evenly in the test dialogs. The error mainly comes from the unaligned utterance space, which leads to failure in understanding the intention of human utterance in the target language. This can lead the system to fail in modifying the dialog state or maintaining the previous dialog states.

6.3 Discussion

Here we want to further highlight the comparison between our transfer learning algorithm with the MT-based approach. Though our approach outperforms the standard Translator trained on IWSLT-2014, it does not necessarily claim that our transfer algorithm outperforms any translation methods on any parallel corpus. In our further ablation studies, we found that using Google Translator [6] can actually achieve a better score than our transfer algorithm, which is understandable considering the complexity of Google Translator and the much larger parallel corpus it leverages. By leveraging more close-to-domain corpus and comprehensive entity recognition/replacement strategy, the translator model is able to achieve a higher score. Apparently, we need to trade off the efficiency for the accuracy. For DST problem, it is an overkill to introduce a more complex translation algorithm, what we pursue is a simple yet efficient algorithm to achieve promising scores. It is also worth mentioning that our XL-NBT algorithm only takes several hours to achieve the reported score, while the translator model takes much more time and memory to train depending on the complexity. Thus, the simplicity and efficiency makes our model a better fit for rare-language and limited-budget scenarios.

6.4 Ablation Test

Here we investigate the effect‘ of hyper-parameter α, τ on the evaluation results. The α is used to balance the optimization of encoder constraint and gate constraint, where larger α means more optimization on gate constraint. The temperature τ is used to control the aggressiveness of the replacement XL-NBT-D, where smaller τ means more source words are replaced by target synonyms. From the table Table 3, we can observe that the experimental results are not very sensitive to α, a dramatic change of α will not harm the final results too much, we simply choose $\alpha = 1$ as the hyper-parameter. In contrast, the system is more sensitive to temperature. Too conservative replacement will lead to weak transfer, while too aggressive replacement will destroy the utterance representation. Therefore, we choose the a moderate temperature of $\tau = 0.1$ throughout our experiments. We also draw the learning curve (Precision vs. Iteration) in the Appendix for both XL-NBT-C and XL-NBT-D. The learning curves show that our algorithm is stable and converges quickly, and the reported results are highly reproducible.

α ablation (τ fixed to 0.1)			**τ ablation (α fixed to 1)**		
value	Goal	Request	value	Goal	Request
α=0	0.13	0.00	τ=0	0.14	0.08
α=0.1	0.46	0.54	τ=0.03	0.43	0.50
α=1	**0.51**	**0.56**	τ=0.1	**0.51**	**0.56**
α=5	0.46	0.54	τ=0.3	0.47	0.51
α=10	0.46	0.52	τ=1	0.44	0.52
α=100	0.44	0.50	τ=10	0.33	0.32

Table 3: Ablation test for hyper-parameter α and τ on English-to-German XL-NBT-D.

7 Conclusion

In our paper, we propose a novel teacher-student framework to perform cross-lingual transfer learning for DST. The key idea of our model is to decouple the current DST neural network into two separate modules and transfer them separately. We believe our method can be further extended into a general purpose multi-lingual transfer framework to resolve other NLP matching or classification problems.

8 Acknowledgement

We are gratefully supported by a Tencent AI Lab Rhino-Bird Gift Fund. We are also very thankful for the public belief tracking code and multi-lingual state-tracking datasets released by Nikola Mrksic from the University of Cambridge.

---

[6] `https://translate.google.com/`

References

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014a. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 360–365. IEEE.

Matthew Henderson, Blaise Thomson, and Steve J. Young. 2014b. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA*, pages 292–299.

David Kamholz, Jonathan Pool, and Susan M Colowick. 2014. Panlex: Building a resource for panlingual lexical translation. In *LREC*, pages 3145–3150.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Çelikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 733–743.

Cettolo Mauro, Girardi Christian, and Federico Marcello. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conference of European Association for Machine Translation*, pages 261–268.

Angeliki Metallinou, Dan Bohus, and Jason Williams. 2013. Discriminative state tracking for spoken dialog systems. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 466–475.

Takashi Miyazaki and Nobuyuki Shimizu. 2016. Cross-lingual image caption generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1780–1790.

Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 794–799.

Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1777–1788.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *arXiv preprint arXiv:1706.00374*.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1470–1480.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1946–1958.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Doina Precup and Yee Whye Teh, editors. 2017. *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*. PMLR.

Abhinav Rastogi, Dilek Hakkani-Tür, and Larry P. Heck. 2017. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2017, Okinawa, Japan, December 16-20, 2017*, pages 561–568.

Lina Maria Rojas-Barahona, Milica Gasic, Nikola Mrksic, Pei-Hao Su, Stefan Ultes, Tsung-Hsien Wen, Steve J. Young, and David Vandyke. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 438–449.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Peihao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1711–1721.

Jason Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 665–677.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1448–1457.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi S. Jaakkola. 2016. Ten pairs to tag - multilingual POS tagging via coarse mapping between embeddings. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1307–1317.

Contextual Parameter Generation for Universal Neural Machine Translation

Emmanouil Antonios Platanios†**, Mrinmaya Sachan**†**, Graham Neubig**‡**, Tom M. Mitchell**†
†Machine Learning Department, ‡Language Technologies Institute
Carnegie Mellon University
{e.a.platanios,mrinmays,gneubig,tom.mitchell}@cs.cmu.edu

Abstract

We propose a simple modification to existing neural machine translation (NMT) models that enables using a single *universal* model to translate between multiple languages while allowing for language specific parameterization, and that can also be used for *domain adaptation*. Our approach requires no changes to the model architecture of a standard NMT system, but instead introduces a new component, the *contextual parameter generator* (CPG), that generates the parameters of the system (e.g., weights in a neural network). This parameter generator accepts source and target language embeddings as input, and generates the parameters for the encoder and the decoder, respectively. The rest of the model remains unchanged and is shared across all languages. We show how this simple modification enables the system to use monolingual data for training and also perform *zero-shot* translation. We further show it is able to surpass state-of-the-art performance for both the IWSLT-15 and IWSLT-17 datasets and that the learned language embeddings are able to uncover interesting relationships between languages.

1 Introduction

Neural Machine Translation (NMT) directly models the mapping of a source language to a target language without any need for training or tuning any component of the system separately. This has led to a rapid progress in NMT and its successful adoption in many large-scale settings (Wu et al., 2016; Crego et al., 2016). The encoder-decoder abstraction makes it conceptually feasible to build a system that maps any source sentence in any language to a vector representation, and then decodes this representation into any target language. Thus, various approaches have been proposed to extend this abstraction for multilingual MT (Luong et al., 2016; Dong et al., 2015; Johnson et al., 2017; Ha et al., 2016; Firat et al., 2016a).

Prior work in multilingual NMT can be broadly categorized into two paradigms. The first, *universal NMT* (Johnson et al., 2017; Ha et al., 2016), uses a single model for all languages. Universal NMT lacks any language-specific parameterization, which is an oversimplification and detrimental when we have very different languages and limited training data. As verified by our experiments, the method of Johnson et al. (2017) suffers from high sample complexity and thus underperforms in limited data settings. The universal model proposed by Ha et al. (2016) requires a new coding scheme for the input sentences, which results in large vocabulary sizes that are difficult to scale. The second paradigm, *per-language encoder-decoder* (Luong et al., 2016; Firat et al., 2016a), uses separate encoders and decoders for each language. This does not allow for sharing of information across languages, which can result in overparameterization and can be detrimental when the languages are similar.

In this paper, we strike a balance between these two approaches, proposing a model that has the ability to learn parameters separately for each language, but also share information between similar languages. We propose using a new *contextual parameter generator* (CPG) which (a) generalizes all of these methods, and (b) mitigates the aforementioned issues of *universal* and *per-language encoder-decoder* systems. It learns language embeddings as a context for translation and uses them to generate the parameters of a shared translation model for all language pairs. Thus, it provides these models the ability to learn parameters separately for each language, but also share information between similar languages. The parameter generator is general and allows any existing NMT model to be enhanced in this way.[1] In addition, it has the following desirable features:

1. **Simple:** Similar to Johnson et al. (2017) and Ha et al. (2016), and in contrast with Luong et al. (2016) and Firat et al. (2016a), it can

---

[1] In fact, it could likely be applied in other scenarios, such as domain adaptation, as well.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 425–435
Brussels, Belgium, October 31 - November 4, 2018.

be applied to most existing NMT systems with some minor modification, and it is able to accommodate attention layers seamlessly.

2. **Multilingual:** Enables multilingual translation using the same single model as before.
3. **Semi-supervised:** Can use monolingual data.
4. **Scalable:** Reduces the number of parameters by employing extensive, yet controllable, sharing across languages, thus mitigating the need for large amounts of data, as in Johnson et al. (2017). It also allows for the decoupling of languages, avoiding the need for a large shared vocabulary, as in Ha et al. (2016).
5. **Adaptable:** Can adapt to support new languages, without requiring complete retraining.
6. **State-of-the-art:** Achieves better performance than pairwise NMT models and Johnson et al. (2017). In fact, our approach can surpass state-of-the-art performance.

We first introduce a modular framework that can be used to define and describe most existing NMT systems. Then, in Section 3, we introduce our main contribution, the *contextual parameter generator* (CPG), in terms of that framework. We also argue that the proposed approach takes us a step closer to a common universal interlingua.

2 Background

We first define the multi-lingual NMT setting and then introduce a modular framework that can be used to define and describe most existing NMT systems. This will help us distill previous contributions and introduce ours.

Setting. We assume that we have a set of source languages S and a set of target languages T. The total number of languages is $L = |S \cup T|$. We also assume we have a set of $C \leq |S| \times |T|$ pairwise parallel corpora, $\{P_1, \dots, P_C\}$, each of which contains a set of sentence pairs for a single source-target language combination. The goal of multilingual NMT is to build a model that, when trained using the provided parallel corpora, can learn to translate well between any pair of languages in $S \times T$. The majority of related work only considers pairwise NMT, where $|S| = |T| = 1$.

2.1 NMT Modules

Most NMT systems can be decomposed to the following modules (also visualized in Figure 1).

Preprocessing Pipeline. The data preprocessing pipeline handles tokenization, cleaning, normalizing the text data and building a *vocabulary*, i.e. a two-way mapping from preprocessed sentences to sequences of word indices that will be used for the translation. A commonly used proposal for defining the vocabulary is the byte-pair encoding (BPE) algorithm which generates subword unit vocabularies (Sennrich et al., 2016b). This eliminates the notion of out-of-vocabulary words, often resulting in increased translation quality.

Encoder/Decoder. The *encoder* takes in indexed source language sentences, and produces an intermediate representation that can later be used by a *decoder* to generate sentences in a target language. Generally, we can think of the encoder as a function, $\mathbf{f}^{(enc)}$, parameterized by $\theta^{(enc)}$. Similarly, we can think of the decoder as another function, $\mathbf{f}^{(dec)}$, parameterized by $\theta^{(dec)}$. The goal of learning to translate can then be defined as finding the values for $\theta^{(enc)}$ and $\theta^{(dec)}$ that result in the best translations. A large amount of previous work proposes novel designs for the encoder/decoder module. For example, using attention over the input sequence while decoding (Bahdanau et al., 2015; Luong et al., 2015) provides significant gains in translation performance.[2]

Parameter Generator. All modules defined so far have previously been used when describing NMT systems and are thus easy to conceptualize. However, in previous work, most models are trained for a given language pair, and it is not trivial to extend them to work for multiple pairs of languages. We introduce here the concept of the *parameter generator*, which makes it easy to define and describe multilingual NMT systems. This module is responsible for generating $\theta^{(enc)}$ and $\theta^{(dec)}$ for any given source and target language. Different parameter generators result in different numbers of learnable parameters and can thus be used to share information across different languages. Next, we describe related work, in terms of the parameter generator for NMT:

- **Pairwise:** In the simple and commonly used pairwise NMT setting (Wu et al., 2016; Crego et al., 2016), the parameter generator would generate separate parameters, $\theta^{(enc)}$ and $\theta^{(dec)}$, for each pair of source-target languages. This re-

---

[2] Note that depending on the vocabulary that is used and on whether it is one shared vocabulary across all languages, or one vocabulary per language, the output projection layer of the decoder (which produces probabilities over words) may be language dependent, or common across all languages. In our experiments, we used separate vocabularies and thus this layer was language-dependent.

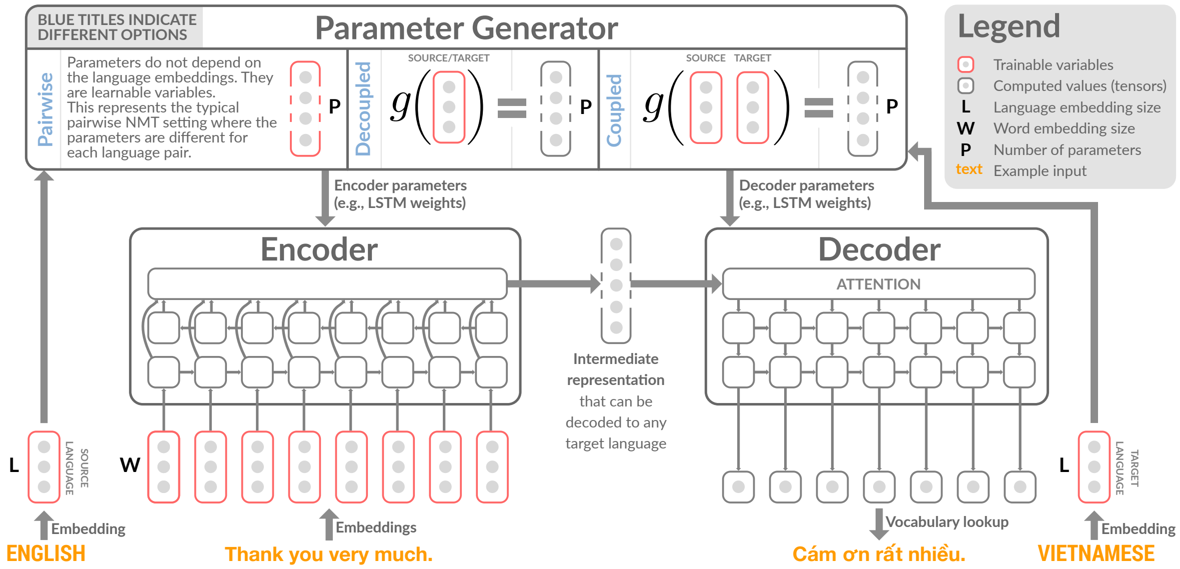

Figure 1: Overview of an NMT system, under our modular framework. Our main contribution lies in the parameter generator module (i.e., coupled or decoupled — each of the boxes with blue titles is a separate option). Note that g denotes a parameter generator network. In our experiments, we consider linear forms for this network. However, our contribution does not depend on the choices made regarding the rest of the modules; we could still use our parameter generator with different architectures for the encoder and the decoder, as well as using different kinds of vocabularies.

sults in no parameter sharing across languages, and thus $\mathcal{O}(ST)$ parameters.

- **Per-Language:** In the case of Dong et al. (2015), Luong et al. (2016) and Firat et al. (2016a), the parameter generator would generate separate encoder parameters, $\theta^{(enc)}$, for each source language, and separate decoder parameters, $\theta^{(dec)}$, for each target language. This leads to a reduction in the number of learnable parameters for multilingual NMT, from $\mathcal{O}(ST)$ to $\mathcal{O}(S+T)$. On one hand, Dong et al. (2015) train multiple models as a one-to-many multilingual NMT system that translates from one source language to multiple target languages. On the other hand, Luong et al. (2016) and Firat et al. (2016a) perform many-to-many translation. Luong et al. (2016), however, only report results for a single language pair and do not attempt multilingual translation. Firat et al. (2016a) propose an attention mechanism that is shared across all language pairs. We generalize the idea of multi-way multilingual NMT with the parameter generator network, described later.
- **Universal:** In the case of Ha et al. (2016) and Johnson et al. (2017), the authors propose using a single common set of encoder-decoder parameters for all language pairs. While Ha et al. (2016) embed words in a common semantic space across languages, Johnson et al. (2017) learn language embeddings that are in the same space as the word embeddings. Here, the parameter generator would provide the same parameters $\theta^{(enc)}$ and $\theta^{(dec)}$ for all language pairs. It would also create and keep track of learnable variables representing language embeddings that are prepended to the encoder input sequence. As we observed in our experiments, this system fails to perform well when the training data is limited. Finally, we believe that embedding languages in the same space as words is not intuitive; in our approach, languages are embedded in a separate space.

In contrast to all these related systems, we provide a simple, efficient, yet effective alternative – a parameter generator for multilingual NMT, that enables semi-supervised and zero-shot learning. We also learn language embeddings, similar to Johnson et al. (2017), but in our case they are separate from the word embeddings and are treated as a *context* for the translation, in a sense that will become clear in the next section. This notion of *context* is used to define parameter sharing across various encoders and decoders, and, as we discuss in our conclusion, is even applicable beyond NMT.

3 Proposed Method

We propose a new way to share information across different languages and to control the amount of sharing, through the parameter generator module. More specifically, we propose *contextual parameter generators*.

Contextual Parameter Generator. Let us denote the source language for a given sentence pair by ℓ_s and the target language by ℓ_t. Then, when using the contextual parameter generator, the parameters of the encoder are defined as $\theta^{(enc)} \triangleq g^{(enc)}(\mathbf{l_s})$, for some function $g^{(enc)}$, where $\mathbf{l_s}$ denotes a language embedding for the source language ℓ_s. Similarly, the parameters of the decoder are defined as $\theta^{(dec)} \triangleq g^{(dec)}(\mathbf{l_t})$ for some function $g^{(dec)}$, where $\mathbf{l_t}$ denotes a language embedding for the target language ℓ_t. Our generic formulation does not impose any constraints on the functional form of $g^{(enc)}$ and $g^{(dec)}$. In this case, we can think of the source language, ℓ_s, as a context for the encoder. The parameters of the encoder depend on its context, but its architecture is common across all contexts. We can make a similar argument for the decoder, and that is where the name of this parameter generator comes from. We can even go a step further and have a parameter generator that defines $\theta^{(enc)} \triangleq g^{(enc)}(\mathbf{l_s}, \mathbf{l_t})$ and $\theta^{(dec)} \triangleq g^{(dec)}(\mathbf{l_s}, \mathbf{l_t})$, thus coupling the encoding and decoding stages for a given language pair. In our experiments we stick to the previous, *decoupled*, form, because unlike Johnson et al. (2017), it has the potential to lead to an *interlingua*.

Concretely, because the encoding and decoding stages are decoupled, the encoder is not aware of the target language while generating it. Thus, we can take an encoded intermediate representation of a sentence and translate it to any target language. This is because, in this case, the intermediate representation is independent of any target language. This makes for a stronger argument that the intermediate representation produced by our encoder could be approaching a universal interlingua, more so than methods that are aware of the target language when they perform encoding.

3.1 Parameter Generator Network

We refer to the functions $g^{(enc)}$ and $g^{(dec)}$ as *parameter generator networks*. Even though our proposed NMT framework does not rely on a specific choice for $g^{(enc)}$ and $g^{(dec)}$, here we describe the functional form we used for our experiments. Our goal is to provide a simple form that works, and for which we can reason about. For this reason, we decided to define the parameter generator networks as simple linear transforms, similar to the factored adaptation model of Michel and Neubig (2018), which was only applied to the bias terms of the output softmax:

$$g^{(enc)}(\mathbf{l_s}) \triangleq \mathbf{W^{(enc)}} \mathbf{l_s}, \tag{1}$$

$$g^{(dec)}(\mathbf{l_t}) \triangleq \mathbf{W^{(dec)}} \mathbf{l_t}, \tag{2}$$

where $\mathbf{l_s}, \mathbf{l_t} \in \mathbb{R}^{\mathbf{M}}$, $\mathbf{W^{(enc)}} \in \mathbb{R}^{\mathbf{P^{(enc)}} \times \mathbf{M}}$, $\mathbf{W^{(dec)}} \in \mathbb{R}^{\mathbf{P^{(dec)}} \times \mathbf{M}}$, M is the language embedding size, $P^{(enc)}$ is the number of parameters of the encoder, and $P^{(dec)}$ is the number of parameters of the decoder.

Another way to interpret this model is that it imposes a low-rank constraint on the parameters. As opposed to our approach, in the base case of using multiple pairwise models to perform multilingual translation, each model has $P = P^{(enc)} + P^{(dec)}$ learnable parameters for its encoder and decoder. Given that the models are pairwise, for L languages, we have a total of $L(L-1)$ learnable parameter vectors of size P. On the other hand, using our contextual parameter generator we have a total of L vectors of size M (one for each language), and a single matrix of size $P \times M$. Then, the parameters of the encoder and the decoder, for a single language pair, are defined as a linear combination of the M columns of that matrix.

Controlled Parameter Sharing. We can further control parameter sharing by observing that the encoder/decoder parameters often have some "natural grouping". For example, in the case of recurrent neural networks we may have multiple weight matrices, one for each layer, as well as attention-related parameters. Based on this observations, we now propose a way to control how much information is shared across languages. The language embeddings need to represent all of the language-specific information and thus may need to be large in size. However, when computing the parameters of each group, only a small part of that information is relevant. Let $\theta^{(enc)} = \{\theta_j^{(enc)}\}_{j=1}^{G}$ and $\theta_j^{(enc)} \in \mathbb{R}^{P_j^{(enc)}}$, where G denotes the number of groups. Then, we define:

$$\theta_j^{(enc)} \triangleq \mathbf{W_j^{(enc)}} \mathbf{P_j^{(enc)}} \mathbf{l_s}, \tag{3}$$

where $\mathbf{W_j^{(enc)}} \in \mathbb{R}^{P_j^{(enc)} \times M'}$ and $\mathbf{P_j^{(enc)}} \in \mathbb{R}^{M' \times M}$, with $M' < M$ (and similarly for the decoder parameters). We can see now that $\mathbf{P_j^{(enc)}}$ is used to extract the relevant information (size M') for parameter group j, from the larger language embedding (size M). This allows us to control the parameter sharing across languages in the following way: if we want to increase the number of per-language parameters (i.e., the language embedding size) we can increase M while keeping M' small enough so that the total number of parameters does not explode. This would not have been possible without the proposed low-rank approximation for $\mathbf{W}^{(\mathbf{enc})}$, that uses the parameter grouping information.

Alternative Options. Given that our proposed approach does not depend on the specific choice of the parameter generator network, it might be interesting to design models that use side-information about the languages that are being used (such as linguistic information about language families and hierarchies). This is outside the scope of this paper, but may be an interesting future direction.

3.2 Semi-Supervised and Zero-Shot Learning

The proposed parameter generator also enables semi-supervised learning via back-translation. Concretely, monolingual data can be used to train the shared encoder/decoder networks to translate a sentence from some language to itself (similar to the idea of auto-encoders by Vincent et al. (2008)). This is possible and can help learning because of the fact that many of the learnable parameters are shared across languages.

Furthermore, zero-shot translation, where the model translates between language pairs for which it has seen no explicit training data, is also possible. This is because the same per-language parameters are used to translate to and from a given language, irrespective of the language at the other end. Therefore, as long as we train our model using some language pairs that involve a given language, it is possible to learn to translate in any direction involving that language.

3.3 Potential for Adaptation

Let us assume that we have trained a model using data for some set of languages, $\ell_1, \ell_2, \dots, \ell_m$. If we obtain data for some new language ℓ_n, we do not have to retrain the whole model from scratch. In fact, we can fix the parameters that are shared across all languages and only learn the embedding for the new language (along with the relevant word embeddings if not using a shared vocabulary). Assuming that we had a sufficient number of languages in the beginning, this may allow us to obtain reasonable translation performance for the new language, with a minimal amount of training.[3]

3.4 Number of Parameters

For the base case of using multiple pairwise models to perform multilingual translation, each model has $P + 2WV$ parameters, where $P = P^{(enc)} + P^{(dec)}$, W is the word embedding size, and V is the vocabulary size per language (assumed to be the same across languages, without loss of generality). Given that the models are pairwise, for L languages, we have a total of $L(L-1)(P + 2WV) = \mathcal{O}(L^2P + 2L^2WV)$ learnable parameters. For our approach, using the linear parameter generator network presented in Section 3.1, we have a total of $\mathcal{O}(PM + LWV)$ learnable parameters. Note that the number of encoder/decoder parameters has no dependence on L now, meaning that our model can easily scale to a large number of languages.

4 Experiments

In this section, we describe our experimental setup along with our results and key observations.

Setup. For all our experiments we use as the base NMT model an encoder-decoder network which uses a bidirectional LSTM for the encoder, and a two-layer LSTM with the attention model of Bahdanau et al. (2015) for the decoder. The word embedding size is set to 512. This is a common baseline model that achieves reasonable performance and we decided to use it as-is, without tuning any of its parameters, as extensive hyperparameter search is outside the scope of this paper.

During training, we use a label smoothing factor of 0.1 (Wu et al., 2016) and the AMSGrad optimizer (Reddi et al., 2018) with its default parameters in TensorFlow, and a batch size of 128 (due to GPU memory constraints). Optimization was stopped when the validation set BLEU score was maximized. The order in which language pairs are used while training was as follows: we always first sample a language pair (uniformly at random), and then sample a batch for that pair (uniformly at random).[4] During inference, we employ beam search with a beam size of 10 and the length normalization scheme of (Wu et al., 2016). We want to emphasize that we did not run experiments with other architectures or configurations, and thus this architecture was not chosen because it was favorable to our method, but rather because it was a frequently mentioned baseline in existing literature.

All experiments were run on a machine with a single Nvidia V100 GPU, and 24 GBs of system memory. Our most expensive experiment took about 10 hours to complete, which would

---

[3] This is due to the small number of parameters that need to be learned in this case. To put this into perspective, in most of our experiments we used language embeddings of size 8.

[4] We did not observe any "forgetting" effect, because we keep "re-visiting" all language pairs throughout training. It would be interesting to explore other sampling schemes, but it is outside the scope of this paper.

Table 1: Comparison of our proposed approach (shaded rows) with the base pairwise NMT model (PNMT) and the Google multilingual NMT model (GML) for the IWSLT-15 dataset. The *Percent Parallel* row shows what portion of the parallel corpus is used while training; the rest is being used only as monolingual data. Results are shown for the BLEU and Meteor metrics. CPG* represents the same model as CPG, but trained without using auto-encoding training examples. The best score in each case is shown in **bold**.

		BLEU				**Meteor**			
		PNMT	GML	CPG*	CPG	PNMT	GML	CPG*	CPG
100% Parallel Data	En→Cs	14.89	15.92	16.88	**17.22**	19.72	20.93	21.51	**21.72**
	Cs→En	24.43	25.25	26.44	**27.37**	27.29	27.46	28.16	**28.52**
	En→De	25.99	15.92	26.41	**26.77**	44.72	42.97	45.97	**46.30**
	De→En	30.93	29.60	31.24	**31.77**	30.73	29.90	30.95	**31.13**
	En→Fr	38.25	34.40	38.10	**38.32**	57.43	53.86	57.42	**57.68**
	Fr→En	37.40	35.14	37.11	**37.89**	34.83	33.14	34.34	**34.89**
	En→Th	23.62	22.22	26.03	**26.33**	-	-	-	-
	Th→En	15.54	14.03	16.54	**26.77**	21.58	21.02	22.78	**23.05**
	En→Vi	27.47	25.54	28.33	**29.03**	-	-	-	-
	Vi→En	24.03	23.19	25.91	**26.38**	27.59	26.96	28.23	**28.79**
	Mean	26.26	24.12	27.30	**27.80**	32.98	32.03	33.67	**34.01**
10% Parallel Data	En→Cs	5.71	8.18	8.40	**9.49**	12.18	14.97	15.25	**15.90**
	Cs→En	6.64	14.56	14.81	**15.38**	13.02	20.04	19.98	**20.87**
	En→De	11.70	14.60	15.09	**16.03**	29.98	33.74	34.88	**36.19**
	De→En	18.10	19.02	19.77	**20.25**	22.57	23.27	23.65	**24.40**
	En→Fr	24.47	25.15	24.00	**25.79**	44.10	44.84	44.95	**46.22**
	Fr→En	23.79	25.02	24.55	**27.12**	26.28	26.61	26.20	**28.18**
	En→Th	7.86	15.58	**18.41**	17.65	-	-	-	-
	Th→En	7.13	9.11	**10.19**	10.14	13.91	16.32	16.78	**16.92**
	En→Vi	18.01	17.51	**18.92**	18.90	-	-	-	-
	Vi→En	6.69	16.00	16.28	**16.86**	13.39	21.01	21.34	**22.28**
	Mean	13.01	16.47	17.04	**17.76**	21.93	25.10	25.38	**26.37**
1% Parallel Data	En→Cs	0.49	1.25	1.57	**2.38**	4.60	6.24	6.28	**8.38**
	Cs→En	1.10	1.76	1.87	**4.60**	6.29	7.13	7.08	**11.15**
	En→De	1.22	4.13	4.06	**6.46**	12.23	18.29	17.61	**23.83**
	De→En	1.46	3.42	3.86	**7.49**	7.58	8.79	8.95	**13.73**
	En→Fr	2.88	7.74	7.41	**12.45**	13.88	21.29	21.80	**30.36**
	Fr→En	4.05	5.22	5.06	**11.39**	9.58	9.86	9.83	**16.34**
	En→Th	1.22	5.72	8.01	**9.26**	-	-	-	-
	Th→En	1.42	1.66	1.65	**3.37**	6.08	7.22	5.89	**8.74**
	En→Vi	5.35	5.61	5.48	**8.00**	-	-	-	-
	Vi→En	2.01	3.57	3.64	**6.43**	7.86	8.76	8.48	**12.04**
	Mean	2.12	4.01	4.26	**7.18**	8.51	10.95	10.74	**15.58**

cost about $25 on a cloud computing service such as Google Cloud or Amazon Web Services, thus making our results reproducible, even by independent researchers.

Experimental Settings. The goal of our experiments is to show how, by using a simple modification of this model, (i) we can achieve significant improvements in performance, while at the same time (ii) being more data and computation efficient, and (iii) enabling support for zero-shot translation. To that end, we perform three types of experiments:

1. Supervised: In this experiment, we use full parallel corpora to train our models. Plain pairwise NMT models (PNMT) are compared to the same models modified to use our proposed decoupled parameter generator. We use two variants: (i) one which does not use auto-encoding of monolingual data while training (CPG*), and (ii) one which does (CPG). Please refer to Section 3.2 for more details.
2. Low-Resource: Similar to the supervised experiments except that we limit the size of the parallel corpora used in training. However, for GML and CPG the full monolingual corpus is used for auto-encoding training.
3. Zero-Shot: In this experiment, our goal is to evaluate how well a model can learn to translate between language pairs that it has not seen while training. For example, a model trained using parallel corpora between English and German, and English and French, will be evaluated in translating from German to French. PNMT can perform zero-shot translation in this setting using pivoting. This means that, in the previous example, we would first translate from German to English and then from English to French (using two pairwise models for a single translation). However, pivoting is prone to error propagation incurred when chaining multiple imperfect translations. The proposed CPG

Table 2: Comparison of our proposed approach (shaded rows) with the base pairwise NMT model (PNMT) and the Google multilingual NMT model (GML) for the IWSLT-17 dataset. Results are shown for the BLEU metric only because Meteor does not support `It`, `Nl`, and `Ro`. CPG^8 represents CPG using language embeddings of size 8. The "$_{C4}$" subscript represents the low-rank version of CPG for controlled parameter sharing (see Section 3.1), using rank 4, etc. The best score in each case is shown in **bold**.

		BLEU							
		PNMT	GML	CPG^8	CPG$^8_{C4}$	CPG$^8_{C2}$	CPG$^8_{C1}$	CPG$^{64}_{C8}$	CPG$^{512}_{C8}$
Supervised	De→En	21.78	21.25	**22.56**	20.78	22.09	21.23	21.50	22.38
	De→It	13.16	13.84	**14.73**	14.34	14.43	13.84	14.34	14.11
	De→Ro	10.85	11.95	12.24	12.37	**12.72**	10.37	11.32	11.94
	En→De	**19.75**	17.06	19.41	19.04	18.42	17.04	17.46	19.29
	En→It	27.70	25.74	27.57	27.11	**28.21**	26.26	27.26	27.48
	En→Nl	24.41	22.46	24.47	**25.15**	24.64	23.94	24.48	24.50
	En→Ro	19.23	18.60	20.83	**20.96**	18.69	17.23	20.20	20.86
	It→De	14.39	12.76	14.61	**15.06**	14.15	13.12	14.18	14.69
	It→En	29.84	27.96	**30.62**	30.10	29.44	29.22	29.56	30.18
	It→Nl	16.74	16.27	17.99	**18.11**	18.05	17.13	17.71	17.99
	Nl→En	26.30	24.78	26.31	26.17	25.74	26.15	**26.33**	26.20
	Nl→It	16.03	16.10	16.81	**17.50**	17.03	16.81	16.89	17.09
	Nl→Ro	12.84	12.48	14.01	**14.44**	12.56	11.79	12.38	13.66
	Ro→De	12.75	12.21	13.58	**13.66**	13.02	12.62	12.96	13.63
	Ro→En	24.33	22.88	23.83	23.88	24.20	23.58	**24.65**	23.57
	Ro→Nl	13.70	14.11	15.34	**15.51**	15.11	14.65	15.29	15.19
	Mean	18.99	18.15	19.68	**19.75**	19.28	18.44	19.16	19.74
Zero-Shot	De→Nl	12.75	12.50	12.74	**12.80**	11.65	12.41	12.67	12.75
	It→Ro	9.97	9.57	10.57	10.17	10.42	9.65	**10.69**	10.32
	Nl→De	11.32	10.47	11.52	11.20	11.28	10.89	**11.63**	11.45
	Ro→It	11.69	10.82	11.51	11.40	11.66	11.42	**11.78**	11.27
	Mean	11.43	10.84	11.59	11.39	11.25	11.09	**11.69**	11.44

models inherently support zero-shot translation and require no pivoting.

For the experiments using the CPG model without controlled parameter sharing, we use language embeddings of size 8. This is based merely on the fact that this is the largest model size we could fit on one GPU. Whenever possible, we compare against PNMT, GML by Johnson et al. (2017),[5] and other state-of-the-art results.

Datasets. We use the following datasets:

- IWSLT-15: Used for supervised and low-resource experiments only (this dataset does not support zero-shot learning). We report results for Czech (`Ch`), English (`En`), French (`Fr`), German (`De`), Thai (`Th`), and Vietnamese (`Vi`). This dataset contains ~90,000-220,000 training sentence pairs (depending on the language pair), ~500-900 validation pairs, and ~1,000-1,300 test pairs.
- IWSLT-17: Used for supervised and zero-shot experiments. We report results for Dutch (`Nl`), English (`En`), German (`De`), Italian (`It`), and Romanian (`Ro`). This dataset contains ~220,000 training sentence pairs (for all language pairs except for the zero-shot ones), ~900 validation pairs, and ~1,100 test pairs.

Data Preprocessing. We preprocess our data using a modified version of the Moses tokenizer (Koehn et al., 2007) that correctly handles escaped HTML characters. We also perform some Unicode character normalization and cleaning. While training, we only consider sentences up to length 50. For both datasets, we generate a per-language vocabulary consisting of the most frequently occurring words, while ignoring words that appear less than 5 times in the whole corpus, and capping the vocabulary size to 20,000 words.

Results. Our results for the IWSLT-15 experiments are shown in Table 1. It is clear that our approach consistently outperforms both the corresponding pairwise model and GML. Furthermore, its advantage grows larger in the low-resource setting (up to 5.06 BLEU score difference, or a 2.4× increase), which is expected due to the extensive parameter sharing in our model. For this dataset, there exist some additional published state-of-the-art results not shown in Tables 1 and 2. Huang et al. (2018) report a BLEU score of 28.07 for the `En→Vi` task, while our model is able to achieve a score of **29.03**. Furthermore, Ha et al. (2016) report a BLEU score of 25.87 for the

---

[5] We use our own implementation of `GML` in order to obtain a fair comparison, in terms of the whole MT pipeline. We have modified it to use the same per-language vocabularies that we use for our approaches, as the proposed shared BPE vocabulary fails to perform well for the considered datasets.

`En→De` task, while our model is able to achieve a score of **26.77**.[6] Our results for the IWSLT-17 experiments are shown in Table 2.[7] Again, our method consistently outperforms both PNMT and GML, in both the supervised and the zero-shot settings. Furthermore, the results indicate that our model performance is robust to different sizes of the language embeddings and the choice of M' for controllable parameter sharing. It only underperforms in the degenerate case where $M' = 1$. It is also worth noting that, in the fully supervised setting, GML, the current state-of-the-art in the multilingual setting, underperforms the pairwise models.

The presented results provide evidence that our proposed approach is able to significantly improve performance, without requiring extensive tuning.

Language Embeddings. An important aspect of our model is that it learns language embeddings. In Figure 2 we show pairwise cosine distances between the learned language embeddings for our fully supervised experiments. There are some interesting patterns that indicate that the learned language embeddings are reasonable. For example, we observe that German (`De`) and Dutch (`Nl`) are most similar for the IWSLT-17 dataset, with Italian (`It`) and Romanian (`Ro`) coming second. Furthermore, Romanian and German are the furthest apart for that dataset. These relationships agree with linguistic knowledge about these languages and the families they belong to. We see similar patterns in the IWSLT-15 results but we focus on IWSLT-17 here, because it is a larger, better quality, dataset with more supervised language pairs. These results are encouraging for analyzing such embeddings to discover relationships between languages that were previously unknown. For example, perhaps surprisingly, French (`Fr`) and Vietnamese (`Vi`) appear to be significantly related for the IWSLT-15 dataset results. This is likely due to French influence in Vietnamese because to the occupation of Vietnam by France during the 19th and 20th centuries (Marr, 1981).

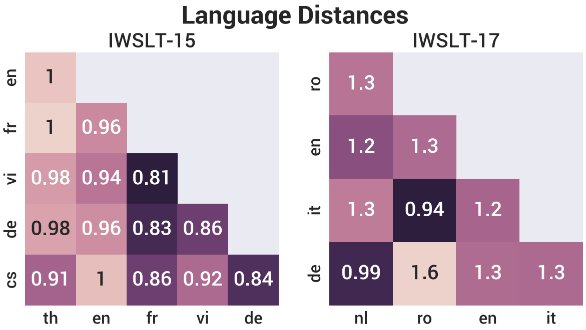

Figure 2: Pairwise cosine distance for all language pairs in the IWSLT-15 and IWSLT-17 datasets. **Darker** colors represent more similar languages.

4.1 Implementation and Reproducibility

Along with this paper we are releasing an implementation of our approach and experiments as part of a new Scala framework for machine translation.[8] It is built on top of TensorFlow Scala (Platanios, 2018) and follows the modular NMT design (described in Section 2.1) that supports various NMT models, including our baselines (e.g., Johnson et al. (2017)). It also contains data loading and preprocessing pipelines that support multiple datasets and languages, and is more efficient than other packages (e.g., `tf-nmt`[9]). Furthermore, the framework supports various vocabularies, among which we provide a new implementation for the byte-pair encoding (BPE) algorithm (Sennrich et al., 2016b) that is 2 to 3 orders of magnitude faster than the released one.[10] All experiments presented in this paper were performed using version `0.1.0` of the framework.

5 Related Work

Interlingual translation (Richens, 1958) has been the object of many research efforts. For a long time, before the move to NMT, most practical machine translation systems only focused on individual language pairs. Since the success of end-to-end NMT approaches such as the encoder-decoder framework (Sutskever et al., 2014; Bahdanau et al., 2015; Cho et al., 2014), recent work has tried to extend the framework to multi-lingual translation. An early approach was Dong et al. (2015) who performed one-to-many translation with a separate attention mechanism for each decoder. Luong et al. (2016) extended this idea with a focus on multi-task learning and multiple encoders and decoders, operating in a single shared vector space. The same architecture is used in

---

[6] We were unable to find reported state-of-the-art results for the rest of the language pairs.

[7] Note that, our results for IWSLT-17 are not comparable to those of the official challenge report (Cettolo et al., 2017), as we use less training data, a smaller baseline model, and our evaluation pipeline potentially differs. However, the numbers presented for all methods in this paper are comparable, as they were all obtained using the same baseline model and evaluation pipeline.

[8] `https://github.com/eaplatanios/symphony-mt`

[9] `https://github.com/tensorflow/nmt`

[10] `https://github.com/rsennrich/subword-nmt`

(Caglayan et al., 2016) for translation across multiple modalities. Zoph and Knight (2016) flipped this idea with a many-to-one translation model, however requiring the presence of a multi-way parallel corpus between all the languages, which is difficult to obtain. Lee et al. (2017) used a single character-level encoder across multiple languages by training a model on a many-to-one translation task. Closest to our work are more recent approaches, already described in Section 2 (Firat et al., 2016a; Johnson et al., 2017; Ha et al., 2016), that attempt to enforce different kinds of parameter sharing across languages.

Parameter sharing in multilingual NMT naturally enables semi-supervised and zero-shot learning. Unsupervised learning has been previously explored with key ideas such as back-translation (Sennrich et al., 2016a), dual learning (He et al., 2016), common latent space learning (Lample et al., 2018), etc. In the vein of multilingual NMT, Artetxe et al. (2018) proposed a model that uses a shared encoder and multiple decoders with a focus on unsupervised translation. The entire system uses cross-lingual embeddings and is trained to reconstruct its input using only monolingual data. Zero-shot translation was first attempted in (Firat et al., 2016b) who performed zero-zhot translation using their pre-trained multi-way multilingual model, fine-tuning it with pseudo-parallel data generated by the model itself. This was recently extended using a teacher-student framework (Chen et al., 2017). Later, zero-shot translation without any additional steps was attempted in (Johnson et al., 2017) using their shared encoder-decoder network. An iterative training procedure that leverages the duality of translations directly generated by the system for zero-shot learning was proposed by Lakew et al. (2017). For extremely low resource languages, Gu et al. (2018) proposed sharing lexical and sentence-level representations across multiple source languages with a single target language. Closely related is the work of Cheng et al. (2016) who proposed the joint training of source-to-pivot and pivot-to-target NMT models.

Ha et al. (2018) are probably the first to introduce a similar idea to that of having one network (called a *hypernetwork*) generate the parameters of another. However, in that work, the input to the hypernetwork are structural features of the original network (e.g., layer size and index). Al-Shedivat et al. (2017) also propose a related method where a neural network generates the parameters of a linear model. Their focus is mostly on interpretability (i.e., knowing which features the network considers important). However, to our knowledge, there is no previous work which proposes having a network generate the parameters of another deep neural network (e.g., a recurrent neural network), using some well-defined context based on the input data. This context, in our case, is the language of the input sentences to the translation model, along with the target translation language.

6 Conclusion and Future Directions

We have presented here a novel *contextual parameter generation* approach to neural machine translation. Our resulting system, which outperforms other state-of-the-art systems, uses a standard pairwise encoder-decoder architecture. However, it differs from earlier approaches by incorporating a component that generates the parameters to be used by the encoder and the decoder for the current sentence, based on the source and target languages, respectively. We refer to this novel component as the *contextual parameter generator*. The benefit of this approach is that it dramatically improves the ratio of the number of parameters to be learned, to the number of training examples available, by leveraging shared structure across different languages. Thus, our approach does not require any extra machinery such as back-translation, dual learning, pivoting, or multilingual word embeddings. It rather relies on the simple idea of *treating language as a context within which to encode/decode*. We also showed that the proposed approach is able to achieve state-of-the-art performance without requiring any tuning. Finally, we performed a basic analysis of the learned language embeddings, which showed that cosine distances between the learned language embeddings reflect well known similarities among language pairs such as German and Dutch.

In the future, we want to extend the concept of the contextual parameter generator to more general settings, such as translating between different modalities of data (e.g., image captioning). Furthermore, based on the discussion of Section 3.3, we hope to develop an adaptable, never-ending learning (Mitchell et al., 2018) NMT system.

Acknowledgments

We would like to thank Otilia Stretcu, Abulhair Saparov, and Maruan Al-Shedivat for the useful feedback they provided in early versions of this paper. This research was supported in part by AFOSR under grant FA95501710218.

References

Maruan Al-Shedivat, Avinava Dubey, and Eric P Xing. 2017. Contextual Explanation Networks. *CoRR*, abs/1705.10301.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised Neural Machine Translation. In *International Conference on Learning Representations*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*.

Ozan Caglayan, Walid Aransa, Yaxing Wang, Marc Masana, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, and Joost van de Weijer. 2016. Does Multimodality Help Human and Machine for Translation and Image Captioning? In *Proceedings of the First Conference on Machine Translation*, volume 2.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichi Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 Evaluation Campaign. In *Proceedings of the 14th International Workshop on Spoken Language Translation*.

Yun Chen, Yang Liu, Yong Cheng, and Victor O K Li. 2017. A Teacher-Student Framework for Zero-Resource Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1925–1935.

Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. 2016. Neural Machine Translation with Pivot Languages. *CoRR*, abs/1611.04928.

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Josep Maria Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, Satoshi Enoue, Chiyo Geiss, Joshua Johanson, Ardas Khalsa, Raoum Khiari, Byeongil Ko, Catherine Kobus, Jean Lorieux, Leidiana Martins, Dang-Chuan Nguyen, Alexandra Priori, Thomas Riccardi, Natalia Segal, Christophe Servan, Cyril Tiquet, Bo Wang, Jin Yang, Dakun Zhang, Jing Zhou, and Peter Zoldan. 2016. SYSTRAN's Pure Neural Machine Translation Systems. *CoRR*, abs/1610.05540.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism. In *Proceedings of NAACL-HLT*, pages 866–875.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman-Vural, and Kyunghyun Cho. 2016b. Zero-Resource Translation with Multi-Lingual Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O K Li. 2018. Universal Neural Machine Translation for Extremely Low Resource Languages. In *Proceedings of NAACL-HLT*, pages 344–354.

David Ha, Andrew Dai, and Quoc V Le. 2018. HyperNetworks. In *International Conference on Learning Representations*.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward Multilingual Neural Machine Translation with Universal Encoder and Decoder. In *Proceedings of the 13th International Workshop on Spoken Language Translation*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual Learning for Machine Translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards Neural Phrase-Based Machine Translation. In *International Conference on Learning Representations*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's Multilingual Neural Machine Translation System:Enabling Zero-Shot Translation. In *Transactions of the Association for Computational Linguistics*, volume 5, pages 339–351.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180.

Surafel M Lakew, Quintino F Lotito, Negri Matteo, Turchi Marco, and Federico Marcello. 2017. Improving Zero-Shot Translation of Low-Resource

Languages. In *Proceedings of the 14th International Workshop on Spoken Language Translation*, pages 113–119.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised Machine Translation Using Monolingual Corpora Only. In *International Conference on Learning Representations*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully Character-Level Neural Machine Translation without Explicit Segmentation. 5:365–378.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task Sequence to Sequence Learning. In *International Conference on Learning Representations*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

David G. Marr. 1981. Language and Literacy. In *Vietnamese Tradition on Trial, 1920-1945*, pages 136–189. University of California Press.

Paul Michel and Graham Neubig. 2018. Extreme Adaptation for Personalized Neural Machine Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 312–318.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir P. Mohamed, Ndapa Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2018. Never-Ending Learning. *Communications of the ACM*, 61(5):103–115.

Emmanouil A. Platanios. 2018. TensorFlow Scala. `https://github.com/eaplatanios/tensorflow_scala`.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*.

Richard H Richens. 1958. Interlingual Machine Translation. *The Computer Journal*, 1(3):144–147.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144.

Barret Zoph and Kevin Knight. 2016. Multi-Source Neural Translation. In *Proceedings of NAACL-HLT*, pages 30–34.

Back-Translation Sampling by Targeting Difficult Words in Neural Machine Translation

Marzieh Fadaee and **Christof Monz**
Informatics Institute, University of Amsterdam
Science Park 904, 1098 XH Amsterdam, The Netherlands
{m.fadaee,c.monz}@uva.nl

Abstract

Neural Machine Translation has achieved state-of-the-art performance for several language pairs using a combination of parallel and synthetic data. Synthetic data is often generated by back-translating sentences randomly sampled from monolingual data using a reverse translation model. While back-translation has been shown to be very effective in many cases, it is not entirely clear why. In this work, we explore different aspects of back-translation, and show that words with high prediction loss during training benefit most from the addition of synthetic data. We introduce several variations of sampling strategies targeting difficult-to-predict words using prediction losses and frequencies of words. In addition, we also target the contexts of difficult words and sample sentences that are similar in context. Experimental results for the WMT news translation task show that our method improves translation quality by up to 1.7 and 1.2 BLEU points over back-translation using random sampling for German→English and English→German, respectively.

1 Introduction

Neural machine translation (NMT) using a sequence-to-sequence model has achieved state-of-the-art performance for several language pairs (Bahdanau et al., 2015; Sutskever et al., 2014; Cho et al., 2014). The availability of large-scale training data for these sequence-to-sequence models is essential for achieving good translation quality.

Previous approaches have focused on leveraging monolingual data which is available in much larger quantities than parallel data (Lambert et al., 2011). Gulcehre et al. (2017) proposed two methods, shallow and deep fusion, for integrating a neural language model into the NMT system. They observe improvements by combining the scores of a neural language model trained on target monolingual data with the NMT system.

Sennrich et al. (2016a) proposed back-translation of monolingual target sentences to the source language and adding the synthetic sentences to the parallel data. In this approach a reverse model trained on parallel data is used to translate sentences from target-side monolingual data into the source language. This *synthetic* parallel data is then used in combination with the actual parallel data to re-train the model. This approach yields state-of-the-art results even when large parallel data are available and has become common practice in NMT (Sennrich et al., 2017; García-Martínez et al., 2017; Ha et al., 2017).

While back-translation has been shown to be very effective to improve translation quality, it is not exactly clear why it helps. Generally speaking, it mitigates the problem of overfitting and fluency by exploiting additional data in the target language. An important question in this context is how to select the monolingual data in the target language that is to be back-translated into the source language to optimally benefit translation quality. Pham et al. (2017) experimented with using domain adaptation methods to select monolingual data based on the cross-entropy between the monolingual data and in-domain corpus (Axelrod et al., 2015) but did not find any improvements over random sampling as originally proposed by Sennrich et al. (2016a). Earlier work has explored to what extent data selection of parallel corpora can benefit translation quality (Axelrod et al., 2011; van der Wees et al., 2017), but such selection techniques have not been investigated in the context of back-translation.

In this work, we explore different aspects of the back-translation method to gain a better understanding of its performance. Our analyses show that the quality of the synthetic data acquired with

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 436–446
Brussels, Belgium, October 31 - November 4, 2018.

a reasonably good model has a small impact on the effectiveness of back-translation, but the ratio of synthetic to real training data plays a more important role. With a higher ratio, the model gets biased towards noises in synthetic data and unlearns the parameters completely. Our findings show that it is mostly words that are difficult to predict in the target language that benefit from additional back-translated data. These are the words with high prediction loss during training when the translation model converges. We further investigate these difficult words and explore alternatives to random sampling of sentences with a focus on increasing occurrences of such words.

Our proposed approach is twofold: identifying difficult words and sampling with the objective of increasing occurrences of these words, and identifying contexts where these words are difficult to predict and sample sentences similar to the difficult contexts. With targeted sampling of sentences for back-translation we achieve improvements of up to 1.7 BLEU points over back-translation using random sampling.

2 Back-Translation for NMT

In this section, we briefly review a sequence-to-sequence NMT system and describe our experimental settings. We then investigate different aspects and modeling challenges of integrating the back-translation method into the NMT pipeline.

2.1 Neural Machine Translation

The NMT system used for our experiments is an encoder-decoder network with recurrent architecture (Luong et al., 2015). For training the NMT system, two sequences of tokens, $X = [x_1, \ldots, x_n]$ and $Y = [y_1, \ldots, y_m]$, are given in the source and target language, respectively.

The source sequence is the input to the encoder which is a bidirectional long short-term memory network generating a representation $\mathbf{s}_n$. Using an attention mechanism (Bahdanau et al., 2015), the attentional hidden state is:

$$\widetilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t])$$

where $\mathbf{h}_t$ is the target hidden state at time step t and $\mathbf{c}_t$ is the context vector which is a weighted average of $\mathbf{s}_n$.

The decoder predicts each target token y_t by computing the probability:

$$p(y_t|y_{<t}, \mathbf{s}_n) = \mathrm{softmax}(\mathbf{W}_o\widetilde{\mathbf{h}}_t)$$

For the token y_t, the conditional probability $p(y_t|y_{<t}, \mathbf{s}_n)$ during training quantifies the difficulty of predicting that token in the context $y_{<t}$. The prediction loss of token y_t is the negative log-likelihood of this probability.

During training on a parallel corpus $\mathbb{D}$, the cross-entropy objective function is defined as:

$$\mathcal{L} = \sum_{(X,Y)\in\mathbb{D}} \sum_{i=1}^{m} -\log p(y_i|y_{<i}, \mathbf{s}_n)$$

The objective of this function is to improve the model's estimation of predicting target words given the source sentence and the target context.

The model is trained end-to-end by minimizing the negative log likelihood of the target words.

2.2 Experimental Setup

For the translation experiments, we use English↔German WMT17 training data and report results on newstest 2014, 2015, 2016, and 2017 (Bojar et al., 2017).

As NMT system, we use a 2-layer attention-based encoder-decoder model implemented in OpenNMT (Klein et al., 2017) trained with embedding size 512, hidden dimension size 1024, and batch size 64. We pre-process the training data with Byte-Pair Encoding (BPE) using 32K merge operations (Sennrich et al., 2016b).

We compare the results to Sennrich et al. (2016a) by back-translating random sentences from the monolingual data and combine them with the parallel training data. We perform the random selection and re-training 3 times and report the averaged outcomes for the 3 models. In all experiments the sentence pairs are shuffled before each epoch.

We measure translation quality by single-reference case-sensitive BLEU (Papineni et al., 2002) computed with the `multi-bleu.perl` script from Moses.

2.3 Size of the Synthetic Data in Back-Translation

One selection criterion for using back-translation is the ratio of real to synthetic data. Sennrich et al. (2016a) showed that higher ratios of synthetic data leads to decreases in translation performance.

In order to investigate whether the improvements in translation performance increases with higher ratios of synthetic data, we perform three experiments with different sizes of synthetic data.

	Size	2014	2015	2016	2017
Baseline	4.5M	26.7	27.6	32.5	28.1
+ synthetic (1:1)	9M	28.7	29.7	36.3	30.8
+ synthetic (1:4)	23M	29.1	30.0	36.9	31.1
+ synthetic (1:10)	50M	22.8	23.6	29.2	23.9

Table 1: German→English translation quality (BLEU) of systems with different ratios of *real:syn* data.

Figure 1 shows the perplexity as a function of training time for different sizes of synthetic data. One can see that all systems perform similarly in the beginning and converge after observing increasingly more training instances. However, the model with the ratio of (1:10) synthetic data gets increasingly biased towards the noisy data after 1M instances. Decreases in performance with more synthetic than real data is also inline with findings of Poncelas et al. (2018).

Comparing the systems using ratios of (1:1) and (1:4), we see that the latter achieves lower perplexity during training. Table 1 presents the performance of these systems on the German→English translation task. The BLEU scores show that the translation quality does not improve linearly with the size of the synthetic data. The model trained on (1:4) real to synthetic ratio of training data achieves the best results, however, the performance is close to the model trained on (1:1) training data.

2.4 Direction of Back-Translation

Adding monolingual data in the target language to the training data has the benefit of introducing new context and improving the fluency of the translation model. The automatically generated translations in the source language while being erroneous, introduce new context for the source words and will not affect the translation model significantly.

Monolingual data is available in large quantities for many languages. The decision of the direction of back-translation is subsequently not based on the monolingual data available, but on the advantage of having more fluent source or target sentences.

Lambert et al. (2011) show that adding synthetic source and real target data achieves improvements in traditional phrase-based machine translation (PBMT). Similarly in previous works in NMT, back-translation is performed on monolingual data in the target language.

	Size	2014	2015	2016	2017
Baseline	4.5M	21.2	23.3	28.0	22.4
+ synthetic tgt	9M	22.4	25.3	29.8	23.7
+ synthetic src	9M	24.0	26.0	30.7	24.8

Table 2: English→German translation quality (BLEU) of systems using forward and reverse models for generating synthetic data.

We perform a small experiment to measure whether back-translating from source to target is also beneficial for improving translation quality. Table 2 shows that in both directions the performance of the translation system improves over the baseline. This is in contrast to the findings of Lambert et al. (2011) for PBMT systems where they show that using synthetic target data does not lead to improvements in translation quality.

Still, when adding monolingual data in the target language the BLEU scores are slightly higher than when using monolingual data in the source language. This indicates the moderate importance of having fluent sentences in the target language.

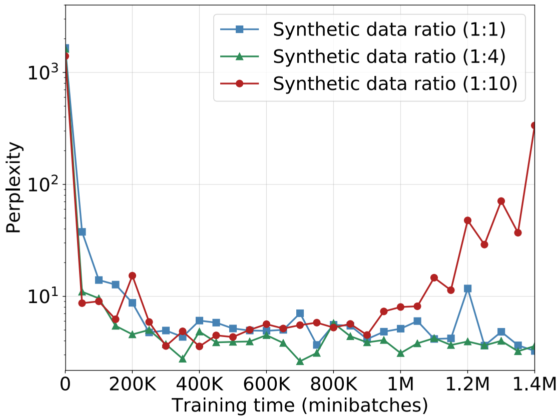

Figure 1: Training plots for systems with different ratios of $(real : syn)$ training data, showing perplexity on development set.

2.5 Quality of the Synthetic Data in Back-Translation

One selection criterion for back-translation is the quality of the synthetic data. Khayrallah and Koehn (2018) studied the effects of noise in the training data on a translation model and discovered that NMT models are less robust to many types of noise than PBMT models. In order for the NMT model to learn from the parallel data, the data should be fluent and close to the manually generated translations. However, automatically generating sentences using back-translation is not as accurate as manual translations.

	Size	2014	2015	2016	2017
Baseline	2.25M	24.3	24.9	29.5	25.6
+ synthetic	4.5M	26.0	26.9	32.2	27.5
+ ground truth	4.5M	26.7	27.6	32.5	28.1

Table 3: German→English translation quality (BLEU).

To investigate the *oracle gap* between the performance of manually generated and back-translated sentences, we perform a simple experiment using the existing parallel training data. In this experiment, we divide the parallel data into two parts, train the reverse model on the first half of the data and use this model to back-translate the second half. The manually translated sentences of the second half are considered as ground truth for the synthetic data.

Table 3 shows the BLEU scores of the experiments. As to be expected, re-training with additional parallel data yields higher performance than re-training with additional synthetic data. However, the differences between the BLEU scores of these two models are surprisingly small. This indicates that performing back-translation with a reasonably good reverse model already achieves results that are close to a system that uses additional manually translated data. This is inline with findings of Sennrich et al. (2016a) who observed that the same monolingual data translated with three translation systems of different quality and used in re-training the translation model yields similar results.

3 Back-Translation and Token Prediction

In the previous section, we observed that the reverse model used to back-translate achieves results comparable to manually translated sentences. Also, there is a limit in learning from synthetic data, and with more synthetic data the model unlearns its parameters completely.

In this section, we investigate the influence of the sampled sentences on the learning model. Fadaee et al. (2017) showed that targeting specific words during data augmentation improves the generation of these words in the right context. Specifically, adding synthetic data to the training data has an impact on the prediction probabilities of individual words. In this section, we further examine the effects of the back-translated synthetic data on the prediction of target tokens.

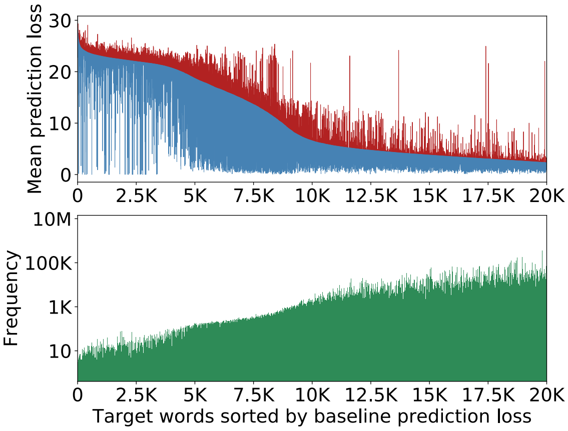

Figure 2: Top: Changes in mean token prediction loss after re-training with synthetic data sorted by mean prediction loss of the baseline system. Decreases and increases in values are marked blue and red, respectively. Bottom: Frequencies (log) of target tokens in the baseline training data.

As mentioned in Section 2.1, the objective function of training an NMT system is to minimize $\mathcal{L}$ by minimizing the prediction loss $-\log p(y_t|y_{<t}, \mathbf{s}_n)$ for each target token in the training data. The addition of monolingual data in the target language improves the estimation of the probability $p(Y)$ and consequently the model generates more fluent sentences.

Sennrich et al. (2016a) show that by using back-translation, the system with target-side monolingual data reaches a lower perplexity on the development set. This is expected since the domain of the monolingual data is similar to the domain of the development set. To investigate the model's accuracy independent from the domains of the data, we collect statistics of the target token prediction loss during training.

Figure 2 shows changes of token prediction loss when training converges and the weights are verging on being stable. The values are sorted by mean token prediction loss of the system trained on real parallel data.

We observe an effect similar to distributional smoothing (Chen and Goodman, 1996): While prediction loss increases slightly for most tokens, the largest decrease in loss occurs for tokens with high prediction loss values. This indicates that by randomly sampling sentences for back-translation, the model improves its estimation of tokens that were originally more difficult to predict, i.e., tokens that had a high prediction loss. Note that we compute the token prediction loss in just one pass over the training corpus with the final model and as a result it is not biased towards the order of the data.

This finding motivates us to further explore sampling criteria for back-translation that contribute considerably to the parameter estimation of the translation model. We propose that by oversampling sentences containing difficult-to-predict tokens we can maximize the impact of using the monolingual data. After translating sentences containing such tokens and including them in the training data, the model becomes more robust in predicting these tokens.

In the next two sections, we propose several methods of using the target token prediction loss to identify the most rewarding sentences for back-translating and re-training the translation model.

4 Targeted Sampling for Difficult Words

One of the main outcomes of using synthetic data is better estimation of words that were originally difficult to predict as measured by their high prediction losses during training. In this section, we propose three variations of how to identify these words and perform sampling to target these words.

Algorithm 1 Sampling for difficult words

Input: Difficult tokens $\mathfrak{D} = \{y_i\}_{i=1}^{D}$, monolingual corpus $\mathbb{M}$, number of required samples N
Output: Sampled sentences $S = \{S_i\}_{i=1}^{N}$ where each sentence S_i is sampled from $\mathbb{M}$

```
procedure DIFFSAMPLING ($\mathfrak{D}, \mathbb{M}, N$):
    Initialize $S = \{\}$
    repeat
        Sample $S_c$ from $\mathbb{M}$
        for all tokens $y$ in $S_c$ do
            if $y \in \mathfrak{D}$ then
                Add $S_c$ to $S$
    until $|S| = N$
    return $S$
```

4.1 Token Frequency as a Feature of Difficulty

Figure 2 shows that the majority of tokens with high mean prediction losses have low frequencies in the training data. Additionally, the majority of decreases in prediction loss after adding synthetic sentence pairs to the training data occurs with less frequent tokens.

Note that these tokens are not necessarily *rare* in the traditional sense; in Figure 2 the $10k$ less frequent tokens in the target vocabulary benefit most from back-translated data.

Sampling new contexts from monolingual data provides context diversity proportional to the token frequencies and less frequent tokens benefit most from new contexts. Algorithm 1 presents this approach where the list of difficult tokens is defined as:

$$\mathfrak{D} = \{\forall y_i \in V_t \colon freq(y_i) < \eta\}$$

V_t is the target vocabulary and η is the frequency threshold for deciding on the difficulty of the token.

4.2 Difficult Words with High Mean Prediction Losses

In this approach, we use the mean losses to identify difficult-to-predict tokens. The mean prediction loss $\hat{\ell}(y)$ of token y during training is defined as follows:

$$\hat{\ell}(y) = \frac{1}{n_y} \sum_{n=1}^{N} \sum_{t=1}^{|Y^n|} - \log p(y_t^n | y_{<t}^n, \mathbf{s}_n) \delta(y_t^n, y)$$

where n_y is the number of times token y is observed during training, i.e., the token frequency of y, N is the number of sentences in the training data, $|Y^n|$ is the length of target sentence n, and $\delta(y_t^n, y)$ is the Kronecker delta function, which is 1 if $y_t^n = y$ and 0 otherwise.

By specifically providing more sentences for difficult words, we improve the model's estimation and decrease the model's uncertainty in prediction. During sampling from the monolingual data, we select sentences that contain difficult words.

Algorithm 1 presents this approach where the list of difficult tokens is defined as:

$$\mathfrak{D} = \{\forall y_i \in V_t \colon \hat{\ell}(y_i) > \mu\}$$

V_t is the vocabulary of the target language and μ is the threshold on the difficulty of the token.

	De-En				En-De			
System	test2014	test2015	test2016	test2017	test2014	test2015	test2016	test2017
Baseline†	26.7	27.6	32.5	28.1	21.2	23.3	28.0	22.4
Random†	28.7	29.7	36.3	30.8	24.0	26.0	30.7	24.8
Freq	29.7	30.5	37.5	31.4	24.2	27.0	31.7	25.2
MeanPredLoss†	29.9	**30.9**	**37.8**	**32.1**	**24.7**	26.8	31.5	**25.5**
MeanPredLoss + stdPredLoss	**30.0**	30.9	37.7	31.9	24.1	26.9	31.0	25.3
Preserve PredLoss ratio	29.8	30.9	37.4	31.6	24.5	**27.2**	**31.8**	25.5

Table 4: German↔English translation quality (Bleu). Experiments marked † are averaged over 3 runs. MeanPredLoss and Freq are difficulty criteria based on mean token prediction loss and token frequency respectively. MeanPredLoss + stdPredLoss is experiments favoring tokens with skewed prediction losses. Preserve PredLoss ratio preserves the ratio of the distribution of difficult contexts.

4.3 Difficult Words with Skewed Prediction Losses

By using the mean loss for target tokens as defined above, we do not discriminate between differences in prediction loss for occurrences in different contexts. This lack of discrimination can be problematic for tokens with high loss variations. For instance, there can be a token with ten occurrences, out of which two have high and eight have low prediction loss values.

We hypothesize that if a particular token is easier to predict in some contexts and harder in others, the sampling strategy should be context sensitive, allowing to target specific contexts in which a token has a high prediction loss. In order to distinguish between tokens with a skewed and tokens with a more uniform prediction loss distribution, we use both mean and standard deviation of token prediction losses to identify difficult tokens.

Algorithm 1 formalizes this approach where the list of the difficult tokens is defined as:

$$\mathfrak{D} = \{\forall y_i \in V_t \colon \hat{\ell}(y_i) > \mu \wedge \sigma(\ell(y_i)) > \rho\}$$

$\hat{\ell}(y_i)$ is the mean and $\sigma(\ell(y_i))$ is the standard deviation of prediction loss of token y_i, V_t is the vocabulary list of the target language, and μ and ρ are the thresholds for deciding on the difficulty of the token.

4.4 Preserving Sampling Ratio of Difficult Occurrences

Above we examined the mean of prediction loss for each token over all occurrences, in order to identify difficult-to-predict tokens. However, the uncertainty of the model in predicting a difficult token varies for different occurrences of the token: one token can be easy to predict in one context, and hard in another. While the sampling step in the previous approaches targets these tokens, it does not ensure that the distribution of sampled sentences is similar to the distribution of problematic tokens in difficult contexts.

Algorithm 2 Sampling with ratio preservation

```
Input: Difficult tokens and the corresponding sentences in the bitext $\mathfrak{D} = \{y_t, Y_{y_t} = [y_1, \ldots, y_t, \ldots, y_m]\}$, monolingual corpus $\mathbb{M}$, number of required samples $N$
Output: Sampled sentences $S = \{S_i\}_{i=1}^{N}$ where each sentence $S_i$ is sampled from $\mathbb{M}$
procedure PredLossRatioSampling($\mathfrak{D}, \mathbb{M}, N$):
    Initialize $S = \{\}$
    $H(y_t) = \frac{N \times |(y_t, \cdot) \in \mathfrak{D}|}{|(y_\cdot, \cdot) \in \mathfrak{D}|}$
    repeat
        Sample $S_c$ from $\mathbb{M}$
        for all tokens $y$ in $S_c$ do
            if $|y \in S| < H(y)$ then
                Add $S_c$ to $S$
    until $|S| = N$
    return $S$
```

To address this issue, we propose an approach where we target the number of times a token occurs in difficult-to-predict contexts and sample sentences accordingly, thereby ensuring the same ratio as the distribution of difficult contexts. If token y_1 is difficult to predict in two contexts and token y_2 is difficult to predict in four contexts, the number of sampled sentences containing y_2 is double the number of sampled sentences containing y_1. Algorithm 2 formalizes this approach.

4.5 Results

We measure the translation quality of various models for German→English and English→German translation tasks. The results of the translation experiments are presented in Table 4. As baseline we compare our approach to Sennrich et al. (2016a). For all experiments we sample and back-translate sentences from the monolingual data, keeping a one-to-one ratio of back-translated versus original data (1:1).

We set the hyperparameters μ, ρ, and η to 5, 10, and 5000 respectively. The values of the hyperparameters are chosen on a small sample of the parallel data based on the token loss distribution.

As expected using random sampling for back-translation improves the translation quality over the baseline. However, all targeted sampling variants in turn outperform random sampling. Specifically, the best performing model for German→English, MEANPREDLOSS, uses the mean of prediction loss for the target vocabulary to oversample sentences including these tokens.

For the English→German experiments we obtain the best translation performance when we preserve the prediction loss ratio during sampling.

We also observe that even though the model targeting tokens with skewed prediction loss distributions (MEANPREDLOSS + STDPREDLOSS) improves over random selection of sentences, it does not outperform the model using only mean prediction losses.

5 Context-Aware Targeted Sampling

In the previous section, we proposed methods for identifying difficult-to-predict tokens and performed targeted sampling from monolingual data. While the objective was to increase the occurrences of difficult tokens, we ignored the context of these tokens in the sampled sentences.

Arguably, if a word is difficult to predict in a given context, providing more examples of the same or similar context can aid the learning process. In this section, we focus on the context of difficult-to-predict words and aim to sample sentences that are similar to the corresponding difficult context.

The general algorithm is described in Algorithm 3. In the following sections, we discuss different definitions of the local context (line 7 and line 9) and similarity measures (line 10) in this algorithm, and report the results.

5.1 Definition of Local Context

Prediction loss is a function of the source sentence and the target context. One reason that a token has high prediction loss is that the occurrence of the word is a deviation from what occurs more frequently in other occurrences of the context in the training data. This indicates an infrequent event, in particular a rare sense of the word, a domain that is different from other occurrences of the word, or an idiomatic expression.

<table>
<tr><td><i>source</i></td><td>wer glaube, dass das ende, sobald sie in Deutschland ank|ä|men, ir|re, erzählt B|ahr.</td></tr>
<tr><td><i>reference</i></td><td>if you think that this stops as soon as they arrive in Germany, you'd be wrong, says B|ahr.</td></tr>
<tr><td><i>NMT output</i></td><td>who believe that the end, as soon as they go to Germany, tells B|risk.</td></tr>
</table>

Table 5: An example from the synthetic data where the word *B|ahr* is incorrectly translated to *B|risk*. Subword unit boundaries are marked with '|'.

We identify *pairs* of tokens and sentences from parallel data where in each pair the NMT model assigns high prediction loss to the token in the given context. Note that a token can occur several times in this list, since it can be considered as difficult-to-predict in different sentences.

We propose two approaches to define the local context of the difficult token:

Neighboring tokens A straightforward way is to use positional context: tokens that precede and follow the target token, typically in a window of w tokens to each side. For sentence S containing a difficult token at index i the *context* function in Algorithm 3 is:

$$context(S, i) = [S^{i-w}, \ldots, S^{i-1}, S^{i+1}, \ldots, S^{i+w}]$$

where S^j is the token at index j in sentence S.

<table>
<tr><td><i>Sentence from bitext containing difficult token:</i></td></tr>
<tr><td>He attended Stan|ford University, where he double maj|ored in Spanish and History.</td></tr>
<tr><td><i>Sampled sentences from monolingual data:</i></td></tr>
<tr><td>– The group is headed by Aar|on K|ush|ner, a Stan|ford University gradu|ate who formerly headed a gre|eting card company.
– Ford just opened a new R&D center near Stan|ford University, a hot|bed of such technological research.
– Joe Grund|fest, a professor and a colleague at Stan|ford Law School, outlines four reasons why the path to the IP|O has become so steep for asp|iring companies.</td></tr>
</table>

Table 6: Results of targeted sampling for the difficult subword unit *'Stan'*.

Neighboring subword units In our analysis of prediction loss during training, we observe that several tokens that are difficult to predict are indeed subword units. Current state-of-the-art NMT systems apply BPE to the training data to address large vocabulary challenges (Sennrich et al., 2016b).

By using BPE the model generalizes common subword units towards what is more frequent in the training data. This is inherently useful since it allows for better learning of less frequent words. However, a side effect of this approach is that at times the model generates subword units that are not linked to any words in the source sentence.

As an example, in Table 5 German source and English reference translation show this problem. The word *B|ahr* consisting of two subword units is incorrectly translated into *B|risk*.

We address the insufficiency of the context for subword units with high prediction losses by targeting these tokens in sentence sampling.

Algorithm 3 formalizes this approach in sampling sentences from the monolingual data. For a sentence S containing a difficult subword at index i, the context function is defined as:

$$context(S, i) = [\ldots, S^{i-1}, S^{i+1}, \ldots]$$

where every token S^j in the local context is a subword unit and part of the same word as S^i. Table 6 presents examples of sampled sentences for the difficult subword unit *'Stan'*.

Sentence from bitext containing difficult word:
Bud|dy Hol|ly was part of the first group induc|ted into the **Rock** and R|oll Hall of F|ame on its formation in 1986.
Sampled sentences from monolingual data:
– A 2008 **Rock** and R|oll Hall of F|ame induc|t|ee, Mad|onna is ran|ked by the Gu|inn|ess Book of World Rec|ords as the top-selling recording artist of all time. – The **Rock** and R|oll Hall of Fam|ers gave birth to the California rock sound. – The winners were chosen by 500 voters, mostly musicians and other music industry veter|ans, who belong to the **Rock** and R|oll Hall of F|ame Foundation.

Table 7: Results of context-aware targeted sampling for the difficult token *'Rock'*

5.2 Similarity of the Local Contexts

In context-aware targeted sampling, we compare the context of a sampled sentence and the difficult context in the parallel data and select the sentence if they are *similar*. In the following, we propose two approaches for quantifying the similarities.

Algorithm 3 Sampling with context

```
Input: Difficult tokens and the corresponding sentences in the bitext $\mathfrak{D} = \{y_t, Y_{y_t} = [y_1, \ldots, y_t, \ldots, y_m]\}$, monolingual corpus $\mathbb{M}$, context function $context$, number of required samples $N$, similarity threshold $s$
Output: Sampled sentences $S = \{S_i\}_{i=1}^{N}$ where each sentence $S_i$ is sampled from $\mathbb{M}$
procedure CNTXTSAMPLING($\mathfrak{D}, \mathbb{M}, context, N, s$):
    Initialize $S = \{\}$
    repeat
        Sample $S_c$ from $\mathbb{M}$
        for all tokens $y_t$ in $S_c$ do
            if $y_t \in \mathfrak{D}$ then
                $C_m \leftarrow context(S_c, \text{indxof}(S_c, y_t))$
                for all $Y_{y_t}$ do
                    $C_p \leftarrow context(Y_{y_t}, \text{indxof}(Y_{y_t}, y_t))$

                    if $\text{Sim}(C_m, C_p) > s$ then
                        Add $S_c$ to $S$
    until $|S| = N$
    return $S$
```

Matching the local context In this approach we aim to sample sentences containing the difficult token, matching the exact context to the problematic context. By sampling sentences that match in a local window with the problematic context and differ in the rest of the sentence, we have more instances of the difficult token for training.

Algorithm 3 formalizes this approach where the similarity function is defined as:

$$\text{Sim}(C_m, C_p) = \frac{1}{c} \sum_{i=1}^{c} \delta(C_m^i, C_p^i)$$

C_m and C_p are the contexts of the sentences from monolingual and parallel data respectively, and c is the number of tokens in the contexts.

Word representations Another approach to sampling sentences that are similar to the problematic context is to weaken the matching assumption. Acquiring sentences that are similar in subject and not match the exact context words allows for lexical diversity in the training data. We use embeddings obtained by training the Skipgram model (Mikolov et al., 2013) on monolingual data to calculate the similarity of the two contexts.

For this approach we define the similarity function in Algorithm 3 as:

$$\text{Sim}(C_m, C_p) = cos(\mathbf{v}(C_m), \mathbf{v}(C_p))$$

where $\mathbf{v}(C_m)$ and $\mathbf{v}(C_p)$ are the averaged embeddings of the tokens in the contexts.

			De-En				**En-De**			
System			test2014	test2015	test2016	test2017	test2014	test2015	test2016	test2017
Baseline †			26.7	27.6	32.5	28.1	21.2	23.3	28.0	22.4
Random †			28.7	29.7	36.3	30.8	24.0	26.0	30.7	24.8
Difficulty criterion	**Context**	**Similarity**								
Freq	tokens	emb	30.0	30.8	37.6	31.7	24.4	26.3	31.5	25.6
PredLoss	SWords	match	29.1	30.1	36.9	31.0	23.8	26.2	28.8	23.2
PredLoss	tokens	match	29.7	30.6	37.6	31.8	24.3	27.4	31.6	25.5
PredLoss	tokens	emb	29.9	30.8	37.7	31.9	24.5	**27.5**	31.7	25.6
PredLoss	sentence	emb	24.9	25.5	30.1	26.2	22.0	24.6	27.9	22.5
MeanPredLoss	tokens	emb	**30.2**	**31.4**	**37.9**	**32.2**	**24.4**	27.2	**31.8**	**25.6**

Table 8: German↔English translation quality (Bleu). Experiments marked † are averaged over 3 runs. PredLoss is the contextual prediction loss and MeanPredLoss is the average loss. token and Sword are context selection definitions from neighboring tokens and subword units respectively. Note that token includes both subword units and full words. emb is computing context similarities with token embeddings and match is comparing the context tokens.

Table 7 presents examples of sampled sentences for the difficult word *rock*. In this example, the context where the word *'Rock'* has high prediction loss is about the *music genre* and not the most prominent sense of the word, *stone*. Sampling sentences that contain this word in this particular context provides an additional signal for the translation model to improve parameter estimation.

5.3 Results

The results of the translation experiments are given in Table 8. In these experiments, we set the hyperparameters s and w to 0.75 and 4, respectively. Comparing the experiments with different similarity measures, match and emb, we observe that in all test sets we achieve the best results when using word embeddings. This indicate that for targeted sampling, it is more beneficial to have diversity in the context of difficult words as opposed to having the exact ngrams.

When using embeddings as the similarity measure, it is worth noting that with a context of size 4 the model performs very well but fails when we increase the window size to include the whole sentence.

The experiments focusing on subword units (sword) achieve improvements over the baselines, however they perform slightly worse than the experiments targeting tokens (token).

The best Bleu scores are obtained with the mean of prediction loss as difficulty criterion (meanPredLoss) and using word representations to identify the most similar contexts. We observe that summarizing the distribution of the prediction losses by its mean is more beneficial than using individual losses. Our results motivate further explorations of using context for targeted sampling sentences for back-translation.

6 Conclusion

In this paper we investigated the effective method of back-translation for NMT and explored alternatives to select the monolingual data in the target language that is to be back-translated into the source language to improve translation quality.

Our findings showed that words with high prediction losses in the target language benefit most from additional back-translated data.

As an alternative to random sampling, we proposed targeted sampling and specifically targeted words that are difficult to predict. Moreover, we used the contexts of the difficult words by incorporating context similarities as a feature to sample sentences for back-translation. We discovered that using the prediction loss to identify weaknesses of the translation model and providing additional synthetic data targeting these shortcomings improved the translation quality in German↔English by up to 1.7 Bleu points.

Acknowledgments

This research was funded in part by the Netherlands Organization for Scientific Research (NWO) under project numbers 639.022.213 and 612.001.218. We also thank NVIDIA for their hardware support and the anonymous reviewers for their helpful comments.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362. Association for Computational Linguistics.

Amittai Axelrod, Yogarshi Vyas, Marianna Martindale, Marine Carpuat, and Johns Hopkins. 2015. Class-based n-gram language difference models for data selection. In *IWSLT (International Workshop on Spoken Language Translation)*, pages 180–187.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Stanley F Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

Kyunghyun Cho, B van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), 2014*.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Mercedes García-Martínez, Ozan Caglayan, Walid Aransa, Adrien Bardet, Fethi Bougares, and Loïc Barrault. 2017. Lium machine translation systems for wmt17 news translation task. In *Proceedings of the Second Conference on Machine Translation*, pages 288–295, Copenhagen, Denmark. Association for Computational Linguistics.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. 2017. On integrating a language model into neural machine translation. *Computer Speech & Language*, 45:137–148.

Thanh-Le. Ha, Jan Niehues, and Alexander Waibel. 2017. Effective Strategies in Zero-Shot Neural Machine Translation. *ArXiv e-prints*.

Huda Khayrallah and Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72. Association for Computational Linguistics.

Patrik Lambert, Holger Schwenk, Christophe Servan, and Sadaf Abdul-Rauf. 2011. Investigations on translation model adaptation using monolingual data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 284–293, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ngoc-Quan Pham, Jan Niehues, Thanh-Le Ha, Eunah Cho, Matthias Sperber, and Alexander Waibel. 2017. The karlsruhe institute of technology systems for the news translation task in wmt 2017. In *Proceedings of the Second Conference on Machine Translation*, pages 366–373.

Alberto Poncelas, Dimitar Shterionov, Andy Way, Gideon Maillette de Buy Wenniger, and Peyman Passban. 2018. Investigating Backtranslation in Neural Machine Translation. *ArXiv e-prints*.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams.

2017. The university of edinburgh's neural mt systems for wmt17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399, Copenhagen, Denmark. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1410, Copenhagen, Denmark. Association for Computational Linguistics.

Multi-Domain Neural Machine Translation with Word-Level Domain Context Discrimination

Jiali Zeng[1] **Jinsong Su**[1*] **Huating Wen**[1] **Yang Liu**[2] **Jun Xie**[3]
Yongjing Yin[1] **Jianqiang Zhao**[4]
[1]Xiamen University, Xiamen, China [2]Tsinghua University, Beijing, China
[3]Mobile Internet Group, Tencent Technology Co., Ltd, Beijing, China
[4]Meiya Pico information Co.,Ltd, Xiamen, China
`lemon@stu.xmu.edu.cn`, `{jssu,htwen,yinyongjing}@xmu.edu.cn`
`liuyang2011@tsinghua.edu.cn`, `stiffxie@tencent.com`, `zhaojq@300188.cn`

Abstract

With great practical value, the study of Multi-domain Neural Machine Translation (NMT) mainly focuses on using mixed-domain parallel sentences to construct a unified model that allows translation to switch between different domains. Intuitively, words in a sentence are related to its domain to varying degrees, so that they will exert disparate impacts on the multi-domain NMT modeling. Based on this intuition, in this paper, we devote to distinguishing and exploiting word-level domain contexts for multi-domain NMT. To this end, we jointly model NMT with monolingual attention-based domain classification tasks and improve NMT as follows: 1) Based on the sentence representations produced by a domain classifier and an adversarial domain classifier, we generate two gating vectors and use them to construct domain-specific and domain-shared annotations, for later translation predictions via different attention models; 2) We utilize the attention weights derived from target-side domain classifier to adjust the weights of target words in the training objective, enabling domain-related words to have greater impacts during model training. Experimental results on Chinese-English and English-French multi-domain translation tasks demonstrate the effectiveness of the proposed model. Source codes of this paper are available on Github https://github.com/DeepLearnXMU/WDCNMT.

1 Introduction

In recent years, neural machine translation (NMT) has achieved great advancement (Nal and Phil, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). However, two difficulties are encountered in the practical applications of NMT. On the one hand, training a NMT model for a specific domain requires a large quantity of parallel sentences in such domain, which is often not readily available. Hence, the much more common practice is to construct NMT models using mixed-domain parallel sentences. In this way, the domain-shared translation knowledge can be fully exploited. On the other hand, the translated sentences often belong to multiple domains, thus requiring a NMT model general to different domains. Since the textual styles, sentence structures and terminologies in different domains are often remarkably distinctive, whether such domain-specific translation knowledge is effectively preserved could have a direct effect on the performance of the NMT model. Therefore, how to simultaneously exploit the exclusive and shared translation knowledge of mixed-domain parallel sentences for multi-domain NMT remains a challenging task.

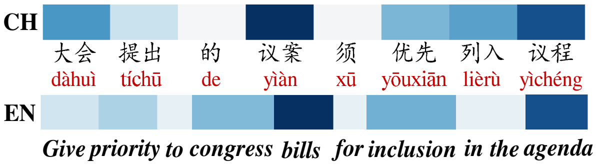

Figure 1: Word-level correlation heat map to *Laws* domain for a Chinese(CH)-English(EN) parallel sentence.

To tackle this problem, recently, researchers have carried out many constructive and in-depth studies (Kobus et al., 2016; Zhang et al., 2016; Pryzant et al., 2017; Farajian et al., 2017). However, most of these studies mainly focus on the utilization of domain contexts as a whole in NMT, while ignoring the discrimination of domain contexts at finer-grained level. In each sentence, some words are closely associated with its domain, while others are domain-independent. Intuitively, these two kinds of words play differ-

---

[*] Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 447–457
Brussels, Belgium, October 31 - November 4, 2018.

ent roles in multi-domain NMT, nevertheless, they are not being distinguished by the current models. Take the sentence shown in Figure 1 for example. The Chinese words “‘大会”(congress), “议案”(bills), “列入”(inclusion), and “议程”(agenda) are frequently used in *Laws* domain and imply the *Laws* style of the sentence, while other words in this sentence are common in all domains and they mainly indicate the semantic meaning of the sentence. Thus, it is reasonable to distinguish and encode these two types of words separately to capture domain-specific and domain-shared contexts, allowing the exclusive and shared knowledge to be exploited without any interference from the other. Meanwhile, the English words “*priority*”,“*government*”, “*bill*” and “*agenda*” are also closely related to *Laws* domain. To preserve the domain-related text style and idioms in generated translations, it is also reasonable for our model to pay more attention to these domain-related words than the others during model training. On this account, we believe that it is significant to distinguish and explore word-level domain contexts for multi-domain NMT.

In this paper, we propose a multi-domain NMT model with word-level domain context discrimination. Specifically, we first jointly model NMT with monolingual attention-based domain classification tasks. In source-side domain classification and adversarial domain classification tasks, we perform two individual attention operations on source-side annotations to generate the domain-specific and domain-shared vector representations of source sentence, respectively. Meanwhile, an attention operation is also placed on target-side hidden states to implement target-side domain classification. Then, we improve NMT with the following two approaches:

(1) According to the sentence representations produced by source-side domain classifier and adverisal domain classifier, we generate two gating vectors for each source annotation. With these two gating vectors, the encoded information of source annotation is selected automatically to construct domain-specific and domain-shared annotations, both of which are used to guide translation predictions via two attention mechanisms;

(2) Based on the attention weights of the target words from target-side domain classifier, we employ word-level cost weighting strategy to refine our model training. In this way, domain-specific target words will be assigned greater weights than others in the objective function of our model.

Our work demonstrates the benefits of separate modeling of the domain-specific and domain-shared contexts, which echoes with the successful applications of the multi-task learning based on shared-private architecture in many tasks, such as discourse relation recognition (Liu et al., 2017b), word segmentation (Chen et al., 2017b), text classification (Liu et al., 2017a), and image classification (Liu et al., 2016). Overall, the main contributions of our work are summarized as follows:

- We propose to construct domain-specific and domain-shared source annotations from initial annotations, of which effects are respectively captured for translation predictions.
- We propose to adjust the weights of target words in the training objective of NMT according to their relevance to different domains.
- We conduct experiments on large-scale multi-domain Chinese-English and English-French datasets. Experimental results demonstrate the effectiveness of our model.

2 Model

Figure 2 illustrates the architecture of our model, which includes a neural encoder equipped with a domain classifier and an adversarial domain classifier, and a neural decoder with two attention models and a target-side domain classifier.

2.1 Neural Encoder

As shown in the lower part of Figure 2, our encoder leverages the sentence representations produced by these two classifiers to construct domain-specific and domain-shared annotations from initial ones, preventing the exclusive and shared translation knowledge from interfering with each other. In our encoder, the input sentence $\mathbf{x}$=$x_1, x_2, ..., x_N$ are first mapped to word vectors and then fed into a bidirectional GRU (Cho et al., 2014) to obtain $\overrightarrow{\mathbf{h}}=\overrightarrow{h}_1, \overrightarrow{h}_2, ..., \overrightarrow{h}_N$ and $\overleftarrow{\mathbf{h}}=\overleftarrow{h}_1, \overleftarrow{h}_2, ..., \overleftarrow{h}_N$ in the left-to-right and right-to-left directions, respectively. These two sequences are then concatenated as $h_i = \{\overrightarrow{h}_i^\top, \overleftarrow{h}_i^\top\}^\top$ to form the word-level semantic representation of the input sentence.

Domain Classifier and Adversarial Domain Classifier. With annotations $\{h_i\}_{i=1}^N$, we employ

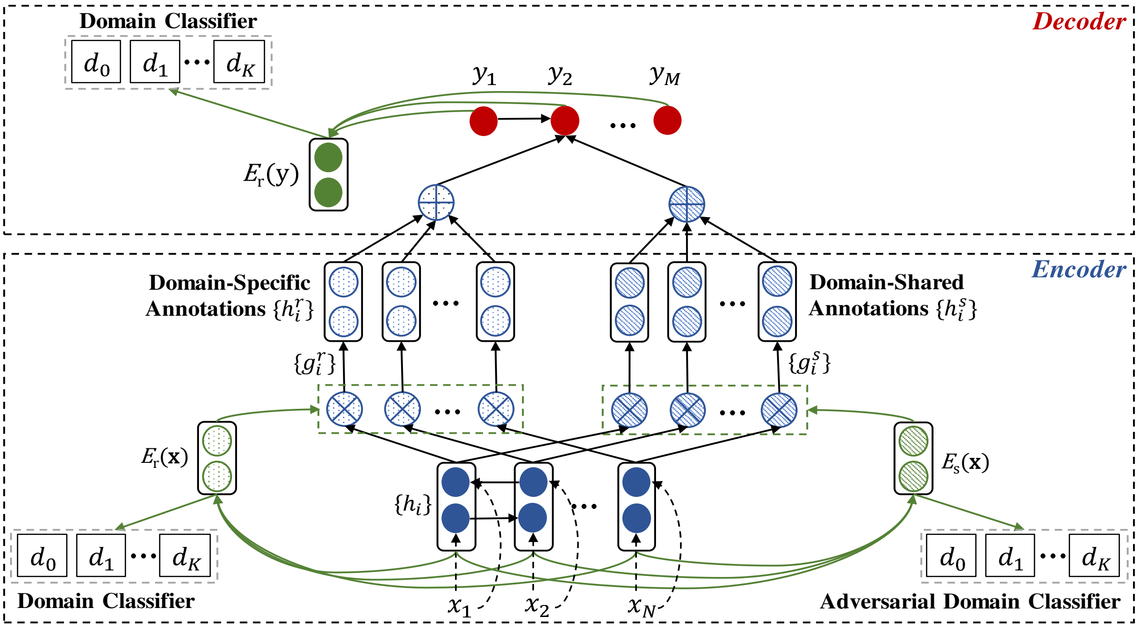

Figure 2: The architecture illustration of our model. Note that our two source-side domain classifiers are used to produce domain-specific and domain-shared annotations, respectively, and our target-side domain classifier is only used during model training.

two attention-like aggregators to generate the semantic representations of sentence $\mathbf{x}$, denoted by the vectors $E_r(\mathbf{x})$ and $E_s(\mathbf{x})$, respectively. Based on these two vectors, we employ the same neural network to model two classifiers with different context modeling objectives:

One is a domain classifier that aims to distinguish different domains in order to generate domain-specific source-side contexts. It is trained using the objective function $\mathcal{J}^s_{dc}(\mathbf{x};\boldsymbol{\theta}^s_{dc})$ = log $p(d|\mathbf{x};\boldsymbol{\theta}^s_{dc})$, where d is the domain tag of $\mathbf{x}$ and $\boldsymbol{\theta}^s_{dc}$ is its parameter set. The other is an adversarial domain classifier capturing source-side domain-shared contexts. To this end, we train it using the following adversarial loss functions:

$$\mathcal{J}^{s1}_{adc}(\mathbf{x};\boldsymbol{\theta}^{s1}_{adc}) = \log p(d|\mathbf{x};\boldsymbol{\theta}^{s1}_{adc},\boldsymbol{\theta}^{s2}_{adc}), \quad (1)$$

$$\mathcal{J}^{s2}_{adc}(\mathbf{x};\boldsymbol{\theta}^{s2}_{adc}) = H(p(d|\mathbf{x};\boldsymbol{\theta}^{s1}_{adc},\boldsymbol{\theta}^{s2}_{adc})), \quad (2)$$

where $H(p(\cdot))$=$-\sum_{k=1}^{K} p_k(\cdot)\log\, p_k(\cdot)$ is an entropy of distribution $p(\cdot)$ with K domain labels, $\boldsymbol{\theta}^{s1}_{adc}$ and $\boldsymbol{\theta}^{s2}_{adc}$ denote the parameters of softmax layer and the generation layer of $E_s(\mathbf{x})$ in this classifier, respectively. By this means, $E_r(\mathbf{x})$ and $E_s(\mathbf{x})$ are expected to encode the domain-specific and domain-shared semantic representations of $\mathbf{x}$, respectively. It should be noted that our utilization of domain classifiers is similar to adversarial training used in (Pryzant et al., 2017) which injects domain-shared contexts into annotations. However, by contrast, we introduce domain classifier and adversarial domain classifier simultaneously to distinguish different kinds of contexts for NMT more explicitly.

Here we describe only the modeling procedure of the domain classifier, while it is also applicable to the adversarial domain classifier. Specifically, $E_r(\mathbf{x})$ is defined as follows:

$$E_r(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i h_i, \quad (3)$$

$$\text{where} \quad \alpha_i = \frac{exp(e_i)}{\sum_{i'}^{N} exp(e_{i'})},$$

$$e_i = (v_a)^\top \tanh(W_a h_i),$$

and v_a and W_a are the relevant attention parameters. Then, we feed $E_r(\mathbf{x})$ into a fully connected layer with *ReLU* function (Ballesteros et al., 2015), and then pass its output through a *softmax* layer to implement domain classification

$$p(\cdot|\mathbf{x};\boldsymbol{\theta}^s_{dc})$$
$$= softmax(W^{s\top}_{dc}\ ReLU(E_r(\mathbf{x})) + b^s_{dc}), \quad (4)$$

where W^s_{dc} and b^s_{dc} are softmax parameters.

Domain-Specific and Domain-Shared Annotations. Since domain-specific and domain-shared contexts have different effects on NMT, and thus should be distinguished and separately captured by NMT model. Specifically, we first leverage the sentence representations $E_r(\mathbf{x})$ and $E_s(\mathbf{x})$ to generate two gating vectors, g_i^r and g_i^s, for annotation h_i in the following way:

$$g_i^r = sigmoid(W_{gr}^{(1)}E_r(\mathbf{x}) + W_{gr}^{(2)}h_i + b_{gr}), \tag{5}$$

$$g_i^s = sigmoid(W_{gs}^{(1)}E_s(\mathbf{x}) + W_{gs}^{(2)}h_i + b_{gs}), \tag{6}$$

where W_{gr}^*, W_{gs}^*, b_{gr} and b_{gs} denote the relevant matrices and bias, respectively. With these two vectors, we construct domain-specific and domain-shared annotations h_i^r and h_i^s from h_i:

$$h_i^r = g_i^r \odot h_i, \tag{7}$$

$$h_i^s = g_i^s \odot h_i. \tag{8}$$

2.2 Neural Decoder

The upper half of Figure 2 illustrates the architecture of our decoder. In particular, with the attention weights of target words from the domain classifier, we employ word-level cost weighting strategy to refine model training.

Formally, our decoder applies a nonlinear function $g(*)$ to define the conditional probability of translation $\mathbf{y}$=$y_1, y_2, ..., y_M$:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{M} p(y_j|\mathbf{x}, y_{<j}) = \prod_{j=1}^{M} g(y_{j-1}, s_j, c_j^r, c_j^s), \tag{9}$$

where the vector s_j denotes the GRU hidden state. It is updated as

$$s_j = GRU(s_{j-1}, y_{j-1}, c_j^r, c_j^s). \tag{10}$$

Here the vectors c_j^r and c_j^s represent the domain-specific and domain-shared contexts, respectively.

Domain-Specific and Domain-Shared Context Vectors. When generating y_j, we define c_j^r as a weighted sum of the domain-specific annotations $\{h_i^r\}$:

$$c_j^r = \sum_{i=1}^{N} \frac{\exp(e_{j,i}^r)}{\sum_{i'=1}^{N}\exp(e_{j,i'}^r)} \cdot h_i^r, \tag{11}$$

$$\text{where} \quad e_{j,i}^r = a(s_{j-1}, h_i^r),$$

and $a(*)$ is a feedforward neural network. Meanwhile, we produce c_j^s from the domain-shared annotations $\{h_i^s\}$ as in Eq. 11. By introducing c_j^r and c_j^s into s_j, our decoder is able to distinguish and simultaneously exploit two types of contexts for translation predictions.

Domain Classifier. We equip our decoder with a domain classifier with parameters $\boldsymbol{\theta_{tdc}}$, which maximizes the training objective i.e., $\mathcal{J}_{dc}^t(\mathbf{y}; \boldsymbol{\theta_{dc}^t}) = log\ p(d|\mathbf{y}; \boldsymbol{\theta_{dc}^t})$. To do this, we also apply attention operation to produce the domain-aware semantic representation $E_r(\mathbf{y})$ of $\mathbf{y}$,

$$E_r(\mathbf{y}) = \sum_{j=1}^{M} \beta_j s_j, \tag{12}$$

$$\text{where} \quad \beta_j = \frac{\exp(e_j)}{\sum_{j'}^{M}\exp(e_{j'})},$$

$$e_j = (v_b)^\top \tanh(W_b s_j),$$

and v_b and W_b are the related parameters. Likewise, we stack a domain classifier on top of $E_r(\mathbf{y})$. Note that this classifier is only used in model training to infer attention weights of target words. These weights measure their semantic relevance to different domains and can be utilized to adjust their cost weights in NMT training objective.

NMT Training Objective with Word-Level Cost Weighting. Formally, we define the objective function of NMT as follows:

$$\mathcal{J}_{nmt}(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta_{nmt}})$$

$$= \sum_{j=1}^{M}(1 + \beta_j)\log p(y_j|\mathbf{x}, y_{<j}; \boldsymbol{\theta_{nmt}}), \tag{13}$$

where β_j is the attention weight of y_j obtained by Eq. (12), and $\boldsymbol{\theta_{nmt}}$ denotes the parameter set of NMT. By this scaling strategy, domain-specific words are emphasized with a bonus, while domain-shared words are updated as usual.

Please note that scaling costs with a multiplicative scalar essentially changes the magnitude of parameter update but without changing its direction (Chen et al., 2017a). Besides, although our scaling strategy is similar to the cost weighting proposed by Chen et al. (2017a), our approach differs from it in two aspects: First, we employ word-level cost weighting rather than sentence-level one to refine NMT training; Second, our approach is less time-consuming for multi-domain NMT.

2.3 Overall Training Objective

Given a mixed-domain training corpus $\mathcal{D} = \{(\mathbf{x}, \mathbf{y}, d)\}$, we train the proposed model according to the following objective function:

$$
\begin{aligned}
\mathcal{J}(\mathcal{D};\boldsymbol{\theta}) = \sum_{(\mathbf{x},\mathbf{y},d)\in\mathcal{D}} & \{\mathcal{J}_{nmt}(\mathbf{x},\mathbf{y};\boldsymbol{\theta_{nmt}}) \\
& + \mathcal{J}^{s}_{dc}(\mathbf{x};\boldsymbol{\theta^{s}_{dc}}) + \mathcal{J}^{t}_{dc}(\mathbf{y};\boldsymbol{\theta^{t}_{dc}}) \\
& + \mathcal{J}^{s1}_{adc}(\mathbf{x};\boldsymbol{\theta^{s1}_{adc}}) + \lambda \cdot \mathcal{J}^{s2}_{adc}(\mathbf{x};\boldsymbol{\theta^{s2}_{adc}})\}
\end{aligned} \quad (14)
$$

where $\mathcal{J}_{nmt}(*)$, $\mathcal{J}^{s}_{dc}(*)$, $\mathcal{J}^{t}_{dc}(*)$ and $\mathcal{J}^{s*}_{adc}(*)$ are the objective functions of NMT, source-side domain classifier, target-side domain classifier, and source-side adversarial domain classifier, respectively, $\boldsymbol{\theta}$=$\{\boldsymbol{\theta_{nmt}}, \boldsymbol{\theta^{s}_{dc}}, \boldsymbol{\theta^{t}_{dc}}, \boldsymbol{\theta^{s1}_{adc}}, \boldsymbol{\theta^{s2}_{adc}}\}$, and λ is the hyper-parameter for adversarial learning.

Particularly, to ensure encoding accuracy of domain-shared contexts, we follow Chen et al. (2017b) to adopt an alternative two-phase strategy in training, where we alternatively optimize $\mathcal{J}(\mathcal{D};\boldsymbol{\theta})$ with $\boldsymbol{\theta^{s1}_{adc}}$ and $\{\boldsymbol{\theta}\text{-}\boldsymbol{\theta^{s1}_{adc}}\}$ respectively fixed at a time.

3 Experiment

To investigate the effectiveness of our model, we conducted multi-domain translation experiments on Chinese-English and English-French datasets.

3.1 Setup

Datasets. For Chinese-English translation, our data comes from UM-Corpus (Tian et al., 2014) and LDC[1]. To ensure data quality, we chose only the parallel sentences with domain label *Laws*, *Spoken*, and *Thesis* from UM-Corpus, and the LDC bilingual sentences related to *News* domain as our dataset. We used randomly selected sentences from UM-Corpus and LDC as development set, and combined the test set of UM-Corpus and randomly selected sentences from LDC to construct our test set. For English-French translation, we conducted experiments on the datasets of OPUS corpus[2], containing sentence pairs from *Medical*, *News*, and *Parliamentary* domains. We also divided these datasets into training, development and test sets. Table 1 provides the statistics of the corpora used in our experiments.

We performed word segmentation on Chinese sentences using *Stanford Segmenter*[3], and tokenized English and French sentences using *MOSES* script[4]. Then, we employed *Byte Pair Encoding* (Sennrich et al., 2016) to convert all words into subwords. The translation quality was evaluated by case-sensitive BLEU (Papineni et al., 2002).

Task	Domain	Train	Dev	Test
CH-EN	Laws	219K	600	456
	Spoken	219K	600	455
	Thesis	299K	800	625
	News	300K	800	650
EN-FR	Medical	1.09M	800	2000
	News	180K	800	2000
	Parliamentary	2.04M	800	2000

Table 1: Sentence numbers of data sets in our experiments.

Contrast Models. Since our model is essentially a standard attentional NMT model enhanced with **w**ord-level **d**omain **c**ontexts, we refer to it as **+WDC**. We compared it with the following models, namely:

- **OpenNMT**[5]. A famous open-source NMT system used widely in the NMT community trained on mix-domain training set.
- **DL4NMT-single** (Bahdanau et al., 2015). A reimplemented attentional NMT trained on a single domain dataset.
- **DL4NMT-mix** (Bahdanau et al., 2015). A reimplemented attentional NMT trained on mix-domain training set.
- **DL4NMT-finetune** (Luong and Manning, 2015). A reimplemented attentional NMT which is first trained using out-of-domain training corpus and then fine-tuned using in-domain dataset.
- **+Domain Control (+DC)** (Kobus et al., 2016). It directly introduces embeddings of source domain tag to enrich annotations of encoder.
- **+Multitask Learning (+ML1)** (Dong et al., 2015). It adopts a multi-task learning framework that shares encoder representation and separates the decoder modeling of different domains.
- **+Multitask Learning (+ML2)** (Pryzant et al., 2017). This model jointly trains

---

[1] https://www.ldc.upenn.edu/.

[2] http://opus.nlpl.eu/

[3] https://nlp.stanford.edu/

[4] http://www.statmt.org/moses/

[5] http://opennmt.net/.

NMT with domain classification via multi-task learning.

- **+Adversarial Discriminative Mixing (+ADM)** (Pryzant et al., 2017). It employs adversarial training to achieve the domain adaptation in NMT.

- **+Target Token Mixing (+TTM)** (Pryzant et al., 2017). This model is similar to +DC, with the only difference that it enriches source annotations by adding target-side domain tag rather than source-side one.

Note that our model uses two annotation sequences, thus we also compared it with the aforementioned models with two times of hidden state size ($2\times hd$). To further examine the effectiveness of the proposed components in our model, we also provided the performance of the following variants of our model:

- **+WDC(S)**. It only exploits the source-side word-level domain contexts for multi-domain NMT.

- **+WDC(T)**. It only employ word-level cost weighting on the target side to refine the model training.

Implementation Details. Following the common practice, we only used the training sentences within 50 words to efficiently train NMT models. Thus, 85.40% and 88.96% of the Chinese-English and English-French parallel sentences were covered in our experiments. In addition, we set the vocabulary size for Chinese-English and English-French as 32,000 and 32,000, respectively. In doing so, our vocabularies covered 99.97% Chinese words and 99.99% English words of the Chinese-English corpus, and almost 100% English words and 99.99% French words of the English-French corpus, respectively.

We applied *Adam* (Kingma and Ba, 2015) to train models and determined the best model parameters based on the model performance on development set. The used hyper-parameter were set as follows: β_1 and β_2 of Adam as 0.9 and 0.999, word embedding dimension as 500, hidden layer size as 1000, learning rate as 5×10^{-4}, batch size as 80, gradient norm as 1.0, dropout rate as 0.1, and beamsize as 10. Other settings were set following (Bahdanau et al., 2015).

Model	**Laws**	**Spoken**	**Thesis**	**News**
Contrast Models (**1**×**hd**)				
OpenNMT	45.82	9.15	13.93	19.73
DL4NMT-single	43.66	5.49	14.54	18.74
DL4NMT-mix	46.82	8.95	15.93	20.33
DL4NMT-finetune	54.19	8.77	16.71	21.55
+DC	49.83	9.18	16.71	20.58
+ML1	46.82	6.66	15.10	20.17
+ML2	48.95	9.45	15.85	20.48
+ADM	48.30	9.41	16.34	20.06
+TTM	49.05	9.36	16.42	20.44
Contrast Models (**2**×**hd**)				
DL4NMT-single	44.48	6.29	14.66	19.87
DL4NMT-mix	48.74	9.01	16.12	20.14
DL4NMT-finetune	54.69	9.07	17.11	21.85
+DC	50.43	9.38	16.45	20.44
+ML1	49.49	7.67	15.50	20.34
+ML2	50.05	9.35	16.03	20.64
+ADM	48.33	9.06	16.59	19.69
+TTM	49.92	9.01	16.38	21.04
Our Models				
+WDC(S)	54.55	10.12	17.22	22.16
+WDC(T)	51.94	9.76	17.72	21.02
+WDC	**55.03**	**10.20**	**18.04**	**22.29**

Table 2: Overall Evaluation of the Chinese-English translation task. $2\times hd$ = two times of hidden state size.

3.2 Results on Chinese-English Translation

We first determined the optimal hyper-parameter λ (see Eq. (14)) on the development set. To do this, we gradually varied λ from 0.1 to 1.0 with an increment of 0.1 in each step. Since our model achieved the best performance when λ=0.1, hence, we set λ=0.1 for all experiments thereafter.

Table 2 shows the overall experimental results. Using almost the same hyper-parameters, our reimplemented DL4NMT outperforms OpenNMT in all domains, demonstrating that our baseline is competitive in performance. Moreover, on all test sets of different domains, our model significantly outperforms other contrast models no matter which hyper-parameters they use. Furthermore, we arrive at the following conclusions:

First, our model surpasses DL4NMT-single, DL4NMT-mix and DL4NMT-finetune, all of which are commonly used in domain adaptation for NMT. Please note that DL4NMT-finetune requires multiple adapted NMT models to be constructed, while ours is a unified one that works well in all domains.

Second, compared with +DC, +ML2 and +ADM which all exploit source-side domain contexts for multi-domain NMT, our +WDC(S) still

(a) An Example Sentence in *Laws* Domain

(b) An Example Sentence in *Thesis* Domain

Figure 3: The correlation heat map of the gating vectors(blue/green) to domain-specific/domain-shared annotations in two example sentences. Note that domain-specific words "澳门"(Macao), "立法会"(Legislative Council), "产生"(Formation), "办法"(Method), "封闭"(Seal), "计算"(Calculation), "实验" (Experiment) are strengthened by g_i^r, while most of the domain-shared words "的"(of) and "与"(and) are focused by g_i^s.

exhibits better performance. This is because that these models focus on one aspect of domain contexts, while our model considers both domain-specific and domain-shared contexts on the source side.

Third, +WDC(T) also outperforms DL4NMT, revealing that it is reasonable and effective to emphasize domain-specific words in model training..

Last, +WDC achieves the best performance when compared with both +WDC(S) and +WDC(T). Therefore, we believe that word-level domain contexts on the both sides are complementary to each other, and utilizing them simultaneously is beneficial to multi-domain NMT.

3.3 Experimental Analysis

Furthermore, we conducted several visualization experiments to empirically analyze the individual effectiveness of the added model components.

3.3.1 Visualizations of Gating Vectors

We first visualized the gating vectors g_i^r and g_i^s to quantify their effects on extracting domain-specific and domain-shared contexts from initial source-side annotations. Since both g_i^r and g_i^s are high dimension vectors, which are difficult to be visualized directly, we followed Li et al. (2016) and Zhou et al. (2017) to visualize their individual contributions to the final output, which can be approximated by their first derivatives.

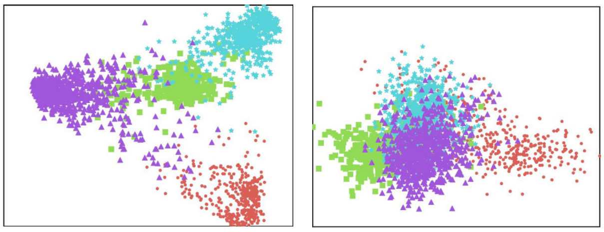

(a) Sentence Representation E_r(x) **(b) Sentence Representation E_s(x)**

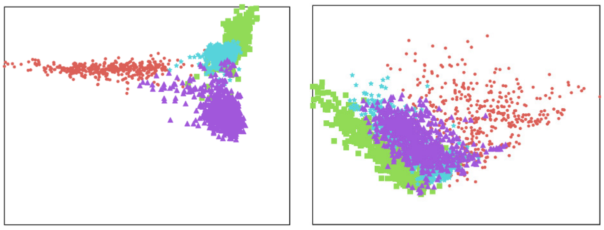

(c) Average of Sentence Annotations $\{h_i^r(x)\}$ **(d) Average of Sentence Annotations $\{h_i^s(x)\}$**

Figure 4: The visualization of the sentence representations and their corresponding average annotations, where the triangle-shaped(purple), circle-shaped(red), square-shaped(green) and pentagonal-shaped(blue) points denote *News*, *Laws*, *Spoken* and *Thesis* sentences, respectively.

Figure 3 shows the first derivative heat maps for two example sentences in *Laws* and *Thesis* domain, respectively. We can observe that without any loss of semantic meanings from source sentences, most of the domain-specific words are strengthened by g_i^r, while most of the domain-shared words, especially function words, are focused by g_i^s. This result is consistent with our expectation for the function of two gating vectors.

3.3.2 Visualizations of Sentence Representations and Annotations

Furthermore, we applied the *hypertools* (Heusser et al., 2018) to visualize the sentence representations $E_r(\mathbf{x})$ and $E_s(\mathbf{x})$, and the domain-specific and domain-shared annotation sequences $\{h_i^r\}_{i=1}^N$ and $\{h_i^s\}_{i=1}^N$. Here we represent each annotation sequence with its average vector in the figure.

As shown in Figure 4 (a) and (b), the sentence representation vectors and the average annotation vectors of different domains are clearly distributed in different regions. By contrast, their distributions are much more concentrated in Figure 4 (c) and (d). Thus, we conclude that our model is able to distinctively learn domain-specific and domain-shared contexts. Moreover, from Figure 4 (b), we observe that the sentence representation vectors of *Laws* domain does not completely coincide with

Domain	**Top10 Target Words**
Laws	*Article*, *Chapter*, *Principles*, *regulations*, *Provisions*, *Political*, *Servants*, *specify*, *China*, Municipal
Spoken	*meanly*, *Rusty*, *1910s*, *scours*, *mountaintops*, *paralyze*, *Puff*, *perpetrators*, *hitter*, *weightlifting*
Thesis	*aggregation*, *Activities*, *Computation*, *Alzheimer*, *nn*, *Contemporarily*, *EVALUATION*, *ethoxycarbonyl*, *sCRC*, *Announced*
News	*months*, *agency*, *outweighed*, *unconstitutionally*, *Congolese*, *session*, *Asia*, *news*, *hurts*, *francs*

Table 3: Examples of Domain-Specific Target Words.

Model	**Medical**	**Parliamentary**	**News**
Contrast Models (**1**×**hd**)			
OpenNMT	78.78	32.96	30.22
DL4NMT-single	77.34	33.28	29.56
DL4NMT-mix	78.48	33.16	31.62
DL4NMT-finetune	78.61	33.72	34.04
+DC	79.34	33.38	33.94
+ML1	77.29	33.39	31.92
+ML2	78.65	33.55	33.48
+ADM	76.74	33.06	33.43
+TTM	78.27	33.29	33.37
Contrast Models (**2**×**hd**)			
DL4NMT-single	78.50	33.38	30.23
DL4NMT-mix	78.84	33.19	33.28
DL4NMT-finetune	79.17	33.88	34.20
+DC	79.96	33.44	33.52
+ML1	78.38	33.20	31.90
+ML2	79.41	33.55	33.62
+ADM	79.31	33.50	33.34
+TTM	79.36	33.13	33.68
Our Models			
+WDC(S)	82.76	34.13	34.31
+WDC(T)	81.51	33.76	33.78
+WDC	**83.35**	**34.17**	**34.87**

Table 4: Overall Evaluation on the English-French translation task.

those of the other domains, this may be caused by the more formal and consistent sentence styles in *Laws* domain.

3.3.3 Illustrations of Domain-Specific Target Words

Lastly, for each domain, we presented the top ten target words with the highest weights learned by our target-side domain classifier. To do this, we calculated the average attention weight of each word in the training corpus as its corresponding domain weight.

As is clearly shown in Table 3 that most listed target words are closely related to their domains. This result validates the aforementioned hypothesis that some words are domain-dependent while others are domain-independent, and our target-side domain classifier is capable of distinguishing them with different attention weights.

3.4 Results on English-French Translation

Likewise, we determined the optimal λ=0.1 on the development set. Table 4 gives the results of English-French multi-domain translation. Similar to the previous experimental result in Section 3.2, our model continues to achieve the best performance compared to all contrast models using two different hidden state size settings, which demonstrates again that our model is effective and general to different language pairs in multi-domain NMT.

4 Related Work

In this work, we study on multi-domain machine translation in the field of domain adaptation for machine translation, which has attracted great attention since SMT (Clark et al., 2012; Huck et al., 2015; Sennrich et al., 2013). As for NMT, the dominant strategies for domain adaptation generally fall into two categories:

The first category is to transfer out-of-domain knowledge to in-domain translation. The conventional method is *fine-tuning*, which first trains the model on out-of-domain dataset and then fine-tunes it on in-domain dataset (Luong and Manning, 2015; Zoph et al., 2016; Servan et al., 2016). Freitag and Al-Onaizan (2016) proceeded further by ensembling the fine-tuned model with the original one. Chu et al. (2017) fine-tuned the model using the mix of in-domain and out-of-domain training corpora. From the perspective of data selection, Chen et al. (2017a) scaled the top-level costs of NMT system according to each training sentence's similarity to the development set. Wang et al. (2017a) explored the data selection strategy based on sentence embeddings for NMT domain adaptation. Moreover, Wang et al. (2017b) further proposed several sentence and domain weighting methods with a dynamic weight learning strategy. However, these approaches usually only perform well on target domain while being highly time consuming in transferring translation knowledge to all the constitute domains.

The second category is to directly use a mixed-domain training corpus to construct NMT model for the translated sentences derived from different domains. In this aspect, Kobus et al. (2016) incorporated domain information into NMT by appending a domain indicator token to each source sequence. Similarly, Johnson et al. (2016) added an artificial token to the input sequence to indicate the required target language. Contrastingly, Farajian et al. (2017) utilized the similarity between each test sentence and the training instances to dynamically set the hyper-parameters of the learning algorithm and update the generic model on the fly. Pryzant et al. (2017) proposed three novel models: *discriminative mixing* that jointly models NMT with domain classification, *adversarial discriminative mixing*, and *target token mixing* which appends a domain token to the target sequence. Sajjad et al. (2017) explored data concatenation, model stacking, data selection and multi-model ensemble to train multi-domain NMT. By exploiting domain as a tag or a feature, Tars and Fishel (2018) treated text domains as distinct languages in order to use multi-lingual approaches when implementing multi-domain NMT. Inspired by topic-based SMT, some researchers resorted to incorporating topical contexts into NMT. Chen et al. (2016) used the topic information of input sentence as an additional input to decoder. Zhang et al. (2016) enhanced the word representation by adding its topic embedding. However, these methods require to have explicit document boundaries between training data, which unfortunately do not exist in most datasets.

Overall, our work is related to the second type of approach with (Pryzant et al., 2017) and (Chen et al., 2017a) most related to ours. Unlike (Pryzant et al., 2017) applying adversarial training to only capture domain-shared translation knowledge, we further exploit domain-specific translation knowledge for multi-domain NMT. Also, in sharp contrast to (Chen et al., 2017a), our model not only exploits the source-side word-level domain contexts differently, but also employs a word-level cost weighting strategy for multi-domain NMT.

5 Conclusion and Future Work

In this work, we have explored how to utilize word-level domain contexts for multi-domain NMT. By jointly modeling NMT and domain classification tasks, we utilize the sentence representations of source-side domain classifier and adversarial domain classifier to construct domain-specific and domain-shared source annotations, which are then exploited by decoder. Moreover, using the attentional weights of target-side domain classifier, we adjust the weights of target words in the training objective to refine model training. Experimental results and in-depth analyses demonstrate the effectiveness of the proposed model.

In the future, we would like to extend the proposed word-level cost weighting strategy to source words. Besides, our method is also general to other NMT models. Therefore, we plan to apply our method to the NMT with complex architectures, for example, lattice-to-sequence NMT (Su et al., 2017), hierarchy-to-sequence NMT (Su et al., 2018), NMT with context-aware encoder (Zhang et al., 2017) and Transformer (Vaswani et al., 2017) and so on.

Acknowledgments

The authors were supported by National Natural Science Foundation of China (No. 61672440), the Fundamental Research Funds for the Central Universities (Grant No. ZK1024), and Scientific Research Project of National Language Committee of China (Grant No. YB135-49). We also thank the reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR 2015*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proc. of EMNLP 2015*.

Boxing Chen, Colin Cherry, George Foster, and Samuel Larkin. 2017a. Cost weighting for neural machine translation domain adaptation. In *Proc. of the First Workshop on Neural Machine Translation*.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided alignment training for topic-aware neural machine translation. *CoRR abs/1607.01628*.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017b. Adversarial multi-criteria learning for chinese word segmentation. In *Proc. of ACL 2017*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger

Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proc. of EMNLP 2014*.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proc. of ACL 2017*.

Jonathan H. Clark, Alon Lavie, and Chris Dyer. 2012. One system, many domains: Open-domain statistical machine translation via feature augmentation. In *Proc. of AMTA 2012*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proc. of ACL 2015*.

M. Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Proc. of WMT 2017*.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *CoRR abs/1612.06897*.

Andrew C. Heusser, Kirsten Ziman, Lucy L. W. Owen, and Jeremy R. Manning. 2018. Hypertools: a python toolbox for gaining geometric insights into high-dimensional data. *Journal of Machine Learning Research*.

M Huck, A Birch, and B Haddow. 2015. Mixed-domain vs. multi-domain statistical machine translation. In *Proc. of MT Summit 2015*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, , and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR abs/1611.04558*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR 2015*.

Catherine Kobus, Josep Crego, and Jean Senellart. 2016. Domain control for neural machine translation. *CoRR abs/1612.06140*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proc. of NAACL 2016*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Domain separation networks. In *Proc. of NIPS 2016*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adversarial multi-task learning for text classification. In *Proc. of ACL 2017*.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2017b. Implicit discourse relation classification via multi-task neural networks. In *Proc. of ACL 2017*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proc. of IWSLT 2015*.

Kalchbrenner Nal and Blunsom Phil. 2013. Recurrent continuous translation models. In *Proc. of EMNLP 2013*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL 2002*.

Reid Pryzant, Denny Britz, and Q Le. 2017. Effective domain mixing for neural machine translation. In *Proc. of WMT 2017*.

Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Yonatan Belinkov, and Stephan Vogel. 2017. Neural machine translation training in a multi-domain scenario. *CoRR abs/1708.08712*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL 2016*.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proc. of ACL 2013*.

Christophe Servan, Josep Crego, and Jean Senellart. 2016. Domain specialization: a post-training domain adaptation for neural machine translation. *CoRR abs/1612.06141*.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *Proc. of AAAI 2017*, pages 3302–3308.

Jinsong Su, Jiali Zeng, Deyi Xiong, Yang Liu, Mingxuan Wang, and Jun Xie. 2018. A hierarchy-to-sequence attentional neural machine translation model. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 26(3):623–632.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS 2014*.

Sander Tars and Mark Fishel. 2018. Multi-domain neural machine translation. *CoRR abs/1805.02282*.

Liang Tian, Derek F. Wong, Lidia S. Chao, Paulo Quaresma, Francisco Oliveira, Shuo Li, Yiming Wang, and Yi Lu. 2014. Um-corpus: A large english-chinese parallel corpus for statistical machine translation. In *Proc. of LREC 2014*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NIPS 2017*.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017a. Sentence embedding for neural machine translation domain adaptation. In *Proc. of ACL 2017*.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017b. Instance weighting for neural machine translation domain adaptation. In *Proc. of EMNLP 2017*.

Biao Zhang, Deyi Xiong, Jinsong Su, and Hong Duan. 2017. A context-aware recurrent encoder for neural machine translation. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 25(12):2424–2432.

Jian Zhang, Liangyou Li, Andy Way, and Qun Liu. 2016. Topic-informed neural machine translation. In *Proc. of COLING 2016*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proc. of ACL 2017*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *Proc. of EMNLP 2016*.

A Discriminative Latent-Variable Model for Bilingual Lexicon Induction

Sebastian Ruder[ə,ﬁ∗] **Ryan Cotterell**[∫,?∗] **Yova Kementchedjhieva**[3] **Anders Søgaard**[3]
[ə]Insight Research Centre, National University of Ireland, Galway, Ireland
[ﬁ]Aylien Ltd., Dublin, Ireland
[∫]The Computer Laboratory, University of Cambridge, Cambridge, UK
[?]Department of Computer Science, Johns Hopkins University, Baltimore, USA
[3]Department of Computer Science, University of Copenhagen, Copenhagen, Denmark
`sebastian@ruder.io,ryan.cotterell@jhu.com,{yova|soegaard}@di.ku.dk`

Abstract

We introduce a novel discriminative latent-variable model for the task of bilingual lexicon induction. Our model combines the bipartite matching dictionary prior of Haghighi et al. (2008) with a state-of-the-art embedding-based approach. To train the model, we derive an efficient Viterbi EM algorithm. We provide empirical improvements on six language pairs under two metrics and show that the prior theoretically and empirically helps to mitigate the hubness problem. We also demonstrate how previous work may be viewed as a similarly fashioned latent-variable model, albeit with a different prior.[1]

1 Introduction

Is there a more fundamental bilingual linguistic resource than a dictionary? The task of bilingual lexicon induction seeks to create a dictionary in a data-driven manner directly from monolingual corpora in the respective languages and, perhaps, a small seed set of translations. From a practical point of view, bilingual dictionaries have found uses in a myriad of NLP tasks ranging from machine translation (Klementiev et al., 2012) to cross-lingual named entity recognition (Mayhew et al., 2017). In this work, we offer a probabilistic twist on the task, developing a novel discriminative latent-variable model that outperforms previous work.

Our proposed model is a bridge between current state-of-the-art methods in bilingual lexicon induction that take advantage of word embeddings, e.g., the embeddings induced by Mikolov et al. (2013b)'s skip-gram objective, and older ideas in the literature that build explicit probabilistic models for the task. We propose a discriminative probability model, inspired by Irvine and Callison-Burch (2013), infused with the bipartite matching dictionary prior of Haghighi et al. (2008). However, like more recent approaches (Artetxe et al., 2017), our model operates directly over pretrained word embeddings, induces a joint cross-lingual embedding space, and scales to large vocabulary sizes. To train our model, we derive a generalized expectation-maximization algorithm (EM; Neal and Hinton, 1998) and employ an efficient matching algorithm.

Empirically, we experiment on three standard and three extremely low-resource language pairs. We evaluate intrinsically, comparing the quality of the induced bilingual dictionary, as well as analyzing the resulting bilingual word embeddings themselves. The latent-variable model yields gains over several previous approaches across language pairs. It also enables us to make implicit modeling assumptions explicit. To this end, we provide a reinterpretation of Artetxe et al. (2017) as a latent-variable model with an IBM Model 1–style (Brown et al., 1993) dictionary prior, which allows a clean side-by-side analytical comparison. Viewed in this light, the difference between our approach and Artetxe et al. (2017), the strongest baseline, is whether one-to-one alignments or one-to-many alignments are admitted between the words of the languages' respective lexicons. Thus, we conclude that our hard constraint on one-to-one alignments is primarily responsible for the improvements over Artetxe et al. (2017).

2 Background: Bilingual Lexicon Induction and Word Embeddings

Bilingual lexicon induction[2] is the task of finding word-level translations between the lexicons of two languages. For instance, the German word *Hund* and the English word *dog* are roughly semantically

---

[∗] The first two authors contributed equally.

[1] The code used to run the experiments is available at `https://github.com/sebastianruder/latent-variable-vecmap`.

[2] For the purposes of this paper, we use bilingual lexicon and (bilingual) dictionary synonymously. On the other hand, unmodified lexicon always refers to a word list in a single language.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 458–468
Brussels, Belgium, October 31 - November 4, 2018.

equivalent, so the pair *Hund–dog* should be an entry in a German–English bilingual lexicon. The task itself comes in a variety of flavors. We consider a version of the task that only relies on *monolingual* corpora in the tradition of Rapp (1995) and Fung (1995). In other words, the goal is to produce a bilingual lexicon primarily from unannotated raw text in each of the respective languages. Importantly, we avoid reliance on bitext, i.e. corpora with parallel sentences that are known translations of each other, e.g., EuroParl (Koehn, 2005). The bitext assumption is quite common in the literature; see Ruder et al. (2018, Table 2) for a survey. Additionally, we will assume the existence of a small seed set of word-level translations obtained from a dictionary; we also experiment with seed sets obtained from heuristics that do not rely on the existence of linguistic resources.

2.1 Graph-Theoretic Formulation

To ease the later exposition, we will formulate the task graph-theoretically. Let ℓ_{src} denote the source language and ℓ_{trg} the target language. Suppose the source language ℓ_{src} has n_{src} word types in its lexicon V_{src} and ℓ_{trg} has n_{trg} word types in its lexicon V_{trg}. We will write $v_{src}(i)$ for the i^{th} word type in ℓ_{src} and $v_{trg}(i)$ for the i^{th} word type in ℓ_{trg}. We can view the elements of V_{src} and V_{trg} as sets of vertices in a graph. Now consider the bipartite set of vertices $V = V_{trg} \cup V_{src}$. In these terms, a bilingual lexicon is just a bipartite graph $G = (E, V)$ and, thus, the task of bilingual lexicon induction is a combinatorial problem: the search for a 'good' edge set $E \subseteq V_{trg} \times V_{src}$. We depict such a bipartite graph in Figure 1. In §3, we will operationalize the notion of 'goodness' by assigning a weight w_{ij} to each possible edge between V_{trg} and V_{src}.

When the edge set E takes the form of a **matching**, we will denote it as $\mathbf{m}$.[3] In general, we will be interested in partial matchings, where many vertices have no incident edges. We will write $\mathcal{M}$ for the set of all partial matchings on the bipartite graph G. The set of vertices in V_{trg} (respectively V_{src}) with no incident edges will be termed $\mathbf{u}_{trg}$ (respectively $\mathbf{u}_{src}$). Note that for any matching $\mathbf{m}$, we have the identity $\mathbf{u}_{trg} = V_{trg} \setminus \{i : (i, j) \in \mathbf{m}\}$.

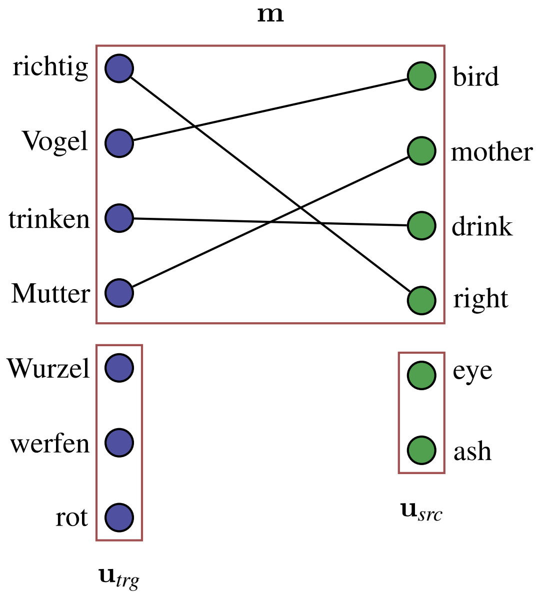

Figure 1: Partial lexicons of German and English shown as a bipartite graph. German is the target language and English is the source language. The $n_{trg} = 7$ German words are shown in blue and the $n_{src} = 6$ English words are shown in green. A bipartite matching $\mathbf{m}$ between the two sets of vertices is also depicted. The German nodes in $\mathbf{u}_{trg}$ are unmatched.

2.2 Word Embeddings

Word embeddings will also play a key role in our model. For the remainder of the paper, we will assume we have access to d-dimensional embeddings for each language's lexicon—for example, those provided by a standard model such as skip-gram (Mikolov et al., 2013b). Notationally, we define the real matrices $S \in \mathbb{R}^{d \times n_{src}}$ and $T \in \mathbb{R}^{d \times n_{trg}}$. Note that in this formulation $s_i \in \mathbb{R}^d$, the i^{th} column of S, is the word embedding corresponding to $v_{src}(i)$. Likewise, note that $t_i \in \mathbb{R}^d$, the i^{th} column of T, is the word embedding corresponding to $v_{trg}(i)$.

3 A Latent-Variable Model

The primary contribution of this paper is a novel latent-variable model for bilingual lexicon induction. The latent variable will be the edge set E, as discussed in §2.1. Given pretrained embeddings for the source and target languages, arranged into the matrices S and T, we define the density

$$p(T \mid S) := \sum_{\mathbf{m} \in \mathcal{M}} p(T \mid S, \mathbf{m}) \cdot p(\mathbf{m}) \quad (1)$$

where, recall from §2, $\mathcal{M}$ is the set of all bipartite matchings on the graph G and $\mathbf{m} \in \mathcal{M}$ is an individual matching. Note that, then, $p(\mathbf{m})$ is a

---

[3] A matching is an edge set where none of the edges share common vertices (West, 2000).

distribution over all bipartite matchings on G such as the matching shown in Figure 1. We will take $p(\mathbf{m})$ to be fixed as the uniform distribution for the remainder of the exposition, though more complicated distributions could be learned, of course. We further define the distribution

$$p_\theta(T \mid S, \mathbf{m}) := \prod_{(i,j)\in\mathbf{m}} p(t_i \mid s_j) \cdot \prod_{i\in\mathbf{u}_{\mathit{trg}}} p(t_i) \quad (2)$$

Recall we write $(i, j) \in \mathbf{m}$ to denote an edge in the matching. Furthermore, for notational simplicity, we have dropped the dependence of $\mathbf{u}_{\mathit{trg}}$ on $\mathbf{m}$. (Recall $\mathbf{u}_{\mathit{trg}} = V_{\mathit{trg}} \setminus \{i : (i, j) \in \mathbf{m}\}$). Next, we define the two densities present in equation (2) as Gaussians:

$$\begin{aligned} p_\theta(t \mid s) &:= \mathcal{N}(\Omega\, s, I) && (3)\\ &\propto \exp ||t - \Omega\, s||_2^2 \\ p_\theta(t) &:= \mathcal{N}(\mu, I) && (4) \end{aligned}$$

Given a fixed matching $\mathbf{m}$, we may create matrices $S_\mathbf{m} \in \mathbb{R}^{d\times|\mathbf{m}|}$ and $T_\mathbf{m} \in \mathbb{R}^{d\times|\mathbf{m}|}$ such that the rows correspond to word vectors of matched vertices (translations under the matching $\mathbf{m}$). Now, after some algebra, we see that we can rewrite $\prod_{(i,j)\in\mathbf{m}} p(t_i \mid s_i)$ in matrix notation:

$$\begin{aligned} p_\theta(T_\mathbf{m} \mid S_\mathbf{m}, \mathbf{m}) &= \prod_{(i,j)\in\mathbf{m}} p(t_i \mid s_j) && (5)\\ &\propto \prod_{(i,j)\in\mathbf{m}} \exp ||t_i - \Omega\, s_j||_2^2 \\ &= \exp \sum_{(i,j)\in\mathbf{m}} ||t_i - \Omega\, s_j||_2^2 \\ &= \exp ||T_\mathbf{m} - \Omega\, S_\mathbf{m}||_F^2 && (6) \end{aligned}$$

where $\Omega \in \mathbb{R}^{d\times d}$ is an orthogonal matrix of parameters to be learned. The result of this derivation, equation (6), will become useful during the discussion of parameter estimation in §4.

We define the model's parameters, to be optimized, as $\theta = (\Omega, \mu)$.

Modeling Assumptions and their Limitations In the previous section, we have formulated the induction of a bilingual lexicon as the search for an edge set E, which we treat as a latent variable that we marginalize out in equation (2). Specifically, we assume that E is a partial matching. Thus, for every $(i, j) \in \mathbf{m}$, we have $t_i \sim \mathcal{N}(\Omega\, s_j, I)$, that is, the embedding for $v_{\mathit{trg}}(i)$ is assumed to have been drawn from a Gaussian centered around the embedding for $v_{\mathit{src}}(j)$, after an orthogonal transformation. This gives rise to two modeling assumptions, which we make explicit: (i) There exists a single source for every word in the target lexicon *and* that source cannot be used more than once.[4] (ii) There exists an orthogonal transformation, after which the embedding spaces are more or less equivalent.

Assumption (i) may be true for related languages, but is likely false for morphologically rich languages that have a many-to-many relationship between the words in their respective lexicons. We propose to ameliorate this using a rank constraint that only considers the top n most frequent words in both lexicons for matching in §6. In addition, we experiment with priors that express different matchings in §7.

As for assumption (ii), previous work (Xing et al., 2015; Artetxe et al., 2017) has achieved some success using an orthogonal transformation; recently, however, Søgaard et al. (2018) demonstrated that monolingual embedding spaces are not approximately isomorphic and that there is a complex relationship between word form and meaning, which is only inadequately modeled by current approaches, which for example cannot model polysemy. Nevertheless, we will show that imbuing our model with these assumptions helps empirically in §6, giving them practical utility.

Why it Works: The Hubness Problem Why should we expect the bipartite matching prior to help, given that we know of cases when multiple source words *should match a target word*? One answer is because the bipartite prior helps us obviate the **hubness problem**, a common issue in word-embedding-based bilingual lexicon induction (Dinu et al., 2015). The hubness problem is an intrinsic problem of high-dimensional vector spaces where certain vectors will be *universal* nearest neighbors, i.e. they will be the nearest neighbor to a disproportionate number of other vectors (Radovanović et al., 2010). Thus, if we allow one-to-many alignments, we will find the embeddings of certain elements of V_{src} acting as hubs, i.e. the model will pick them to generate a disproportionate number of target embeddings, which reduces the quality of the embedding space.[5]

Another explanation for the positive effect of the

---

[4] This is true by the definition of a matching.

[5] In §5, we discuss the one-to-many alignment used in several of our baseline systems.

Algorithm 1 Viterbi EM for our latent-variable model

```
repeat
    // Viterbi E-Step
    $\mathbf{m}^\star \leftarrow \text{argmax}_{\mathbf{m}\in\mathcal{M}} \log p_\theta(\mathbf{m} \mid S, T)$
    $\mathbf{u}^\star_{trg} \leftarrow V_{trg} \setminus \{i : (i,j) \in \mathbf{m}^\star\}$
    // M-Step
    $U\Sigma V^\top \leftarrow \text{SVD}\left(T_{\mathbf{m}^\star} S^\top_{\mathbf{m}^\star}\right)$
    $\Omega^\star \leftarrow UV^\top$
    $\mu^\star \leftarrow {}^1\!/\!_{|\mathbf{u}^\star_{trg}|} \cdot \sum_{i\in\mathbf{u}^\star_{trg}} t_i$
    $\theta \leftarrow (\Omega^\star, \mu^\star)$
until converged
```

one-to-one alignment prior is its connection to the Wasserstein distance and computational optimal transport (Villani, 2008). Concurrent work (Grave et al., 2018) similarly has found the one-to-one alignment prior to be beneficial.

4 Parameter Estimation

We will conduct parameter estimation through Viterbi EM. We describe first the E-step, then the M-step. Viterbi EM estimates the parameters by alternating between the two steps until convergence. We give the complete pseudocode in Algorithm 1.

4.1 Viterbi E-Step

The E-step asks us to compute the posterior of latent bipartite matchings $p(\mathbf{m} \mid S, T)$. Computation of this distribution, however, is intractable as it would require a *sum* over all bipartite matchings, which is #P-hard (Valiant, 1979). Tricks from combinatorial optimization make it possible to *maximize* over all bipartite matchings in polynomial time. Thus, we fall back on the Viterbi approximation for the E-step (Brown et al., 1993; Samdani et al., 2012). The derivation will follow Haghighi et al. (2008). In order to compute

$$\mathbf{m}^\star = \underset{\mathbf{m}\in\mathcal{M}}{\text{argmax}} \log p_\theta(\mathbf{m} \mid S, T) \tag{7}$$

we construct a fully connected bipartite graph $G = (E, V_{src} \cup V_{trg})$, where $E = V_{src} \times V_{trg}$. We weight each arc $(i, j) \in E$ with the weight between the projected source word and target word embeddings: $w_{ij} = \log p(t_i \mid s_j) - \log p(t_i) = ||t_i - \Omega\, s_j||_2^2 - ||t_i - \mu||_2^2$, where the normalizers of both Gaussians cancel as both have the same covariance matrix, i.e., I. Note that in the case where the t_i and the s_j are of length 1, that is, $||t_i||_2 = ||s_j||_2 = 1$, and $\mu = \mathbf{0}$, we recover cosine distance between the vectors up to an additive constant as orthogonal matrices preserve length (the constant is always -1 as $||t_i||_2 = 1$). We may ignore this constant during the E-step's combinatorial optimization. Note the optimal partial matching will contain no edges with weight $w_{ij} < 0$. For this reason, we remove such edges from the bipartite graph. To find the maximal *partial* bipartite matching on G to compute $\mathbf{m}^\star$, we employ an efficient algorithm as detailed in the next section.

Finding a Maximal Bipartite Matching We frame finding an optimal one-to-one alignment between n_{src} source and n_{trg} words as a combinatorial optimization problem, specifically, a linear assignment problem (LAP; Bertsimas and Tsitsiklis, 1997). In its original formulation, the LAP requires assigning a number of agents (source words) to a number of tasks (target words) at a cost that varies based on each assignment. An optimal solution assigns each source word to exactly one target word and vice versa at minimum cost. The Hungarian algorithm (Kuhn, 1955) is one of the most well-known approaches for solving the LAP, but runs in $O((n_{src} + n_{trg})^3)$. This works for smaller vocabulary sizes,[6] but is prohibitive for matching cross-lingual word embeddings with large vocabularies for real-world applications.[7]

For each source word, most target words, however, are unlikely candidates for alignment. We thus propose to consider only the top k most similar target words for alignment with every source word. We sparsify the graph by weighting the edges for all other words with $-\infty$. The remaining weights w_{ij} are chosen as discussed above. We employ a version of the Jonker-Volgenant algorithm (Jonker and Volgenant, 1987; Volgenant, 1996), which has been optimized for LAP on sparse graphs, to find the maximum-weight matching $\mathbf{m}^\star$ on G.[8]

4.2 M-Step

Next, we will describe the M-step. Given an optimal matching $\mathbf{m}^\star$ computed in §4.1, we search for a matrix $\Omega \in \mathbb{R}^{d\times d}$. We additionally enforce the constraint that Ω is a real orthogonal matrix, i.e., $\Omega^\top\Omega = I$. Previous work (Xing et al., 2015;

---

[6] Haghighi et al. (2008) use the Hungarian algorithm to find a matching between 2000 source and target language words.

[7] For reference, in §6, we learn bilingual lexicons between embeddings of 200,000 source and target language words.

[8] After acceptance to EMNLP 2018, Edouard Grave pointed out that Sinkhorn propagation (Adams and Zemel, 2011; Mena et al., 2018) may have been a computationally more effective manner to deal with the latent matchings.

Artetxe et al., 2017) found that the orthogonality constraint leads to noticeable improvements.

Our M-step optimizes two objectives independently. First, making use of the result in equation (6), we optimize the following:

$$\begin{aligned}\log p(T_{\mathbf{m}^\star} \mid S_{\mathbf{m}^\star}, \mathbf{m}^\star) \quad (8)\\ = ||T_{\mathbf{m}^\star} - \Omega\, S_{\mathbf{m}^\star}||_F^2 + C\end{aligned}$$

with respect to Ω subject to $\Omega^\top \Omega = I$. (Note we may ignore the constant C during the optimization.) Second, we optimize the objective

$$\log \prod_{i \in \mathbf{u}_{trg}} p(t_i) = \sum_{i \in \mathbf{u}_{trg}} ||t_i - \mu||_2^2 + D \quad (9)$$

with respect to the mean parameter μ, which is simply an average. Note, again, we may ignore the constant D during optimization.

Optimizing equation (8) with respect to Ω is known as the orthogonal Procrustes problem (Schönemann, 1966; Gower and Dijksterhuis, 2004) and has a closed form solution that exploits the singular value decomposition (Horn and Johnson, 2012). Namely, we compute $U\Sigma V^\top = T_{\mathbf{m}}^\top S_{\mathbf{m}}$. Then, we directly arrive at the optimum: $\Omega^\star = UV^\top$. Optimizing equation (9) can also been done in closed form; the point which minimizes distance to the data points (thereby maximizing the log-probability) is the centroid: $\mu^\star = {}^1\!/\!_{|\mathbf{u}_{trg}|} \cdot \sum_{i \in \mathbf{u}_{trg}} t_i$.

5 Reinterpretation of Artetxe et al. (2017) as a Latent-Variable Model

The self-training method of Artetxe et al. (2017), our strongest baseline in §6, may also be interpreted as a latent-variable model in the spirit of our exposition in §3. Indeed, we only need to change the edge-set prior $p(\mathbf{m})$ to allow for edge sets other than those that are matchings. Specifically, a matching enforces a one-to-one alignment between types in the respective lexicons. Artetxe et al. (2017), on the other hand, allow for one-to-many alignments. We show how this corresponds to an alignment distribution that is equivalent to IBM Model 1 (Brown et al., 1993), and that Artetxe et al. (2017)'s self-training method is actually a form of Viterbi EM.

To formalize Artetxe et al. (2017)'s contribution as a latent-variable model, we lay down some more notation. Let $\mathcal{A} = \{1, \ldots, n_{src} + 1\}^{n_{trg}}$, where we define $(n_{src} + 1)$ to be `none`, a distinguished symbol indicating unalignment. The set $\mathcal{A}$ is to be interpreted as the set of all one-to-many alignments $\mathbf{a}$ on the bipartite vertex set $V = V_{trg} \cup V_{src}$ such that $a_i = j$ means the i^{th} vertex in V_{trg} is aligned to the j^{th} vertex in V_{src}. Note that $a_i = (n_{src} + 1) = \texttt{none}$ means that the i^{th} element of V_{trg} is unaligned. Now, by analogy to our formulation in §3, we define

$$\begin{aligned} p(T \mid S) &:= \sum_{\mathbf{a} \in \mathcal{A}} p(T \mid S, \mathbf{a}) \cdot p(\mathbf{a}) && (10)\\ &= \sum_{\mathbf{a} \in \mathcal{A}} \prod_{i=1}^{n_{trg}} p(t_i \mid s_{a_i}, a_i) \cdot p(a_i) && (11)\\ &= \prod_{i=1}^{n_{trg}} \sum_{a_i=1}^{n_{src}+1} p(t_i \mid s_{a_i}, a_i) \cdot p(a_i) && (12)\end{aligned}$$

The move from equation (11) to equation (12) is the dynamic-programming trick introduced in Brown et al. (1993). This reduces the number of terms in the expression from exponentially many to polynomially many. We take $p(\mathbf{a})$ to be a uniform distribution over all alignments with no parameters to be learned.

Artetxe et al. (2017)'s Viterbi E-Step In the context of Viterbi EM, it means the max over $\mathcal{A}$ will decompose additively s

$$\max_{\mathbf{a} \in \mathcal{A}} \log p(\mathbf{a} \mid S, T) = \sum_{i=1}^{n_{trg}} \max_{1 \le a_i \le (n_{src}+1)} \log p(a_i \mid S, T)$$

thus, we can simply find $\mathbf{a}^\star$ component-wise as follows:

$$a_i^\star = \underset{1 \le a_i \le (n_{src}+1)}{\operatorname{argmax}} \log p(a_i \mid t_i, s_{a_i}) \quad (13)$$

Artetxe et al. (2017)'s M-step The M-step remains unchanged from the exposition in §3 with the exception that we fit Ω given matrices $S_{\mathbf{a}}$ and $T_{\mathbf{a}}$ formed from a one-to-many alignment $\mathbf{a}$, rather than a matching $\mathbf{m}$.

Why a Reinterpretation? The reinterpretation of Artetxe et al. (2017) as a probabilistic model yields a clear analytical comparison between our method and theirs. The only difference between the two is the constraint on the bilingual lexicon that the model is allowed to induce.

6 Experiments

We first conduct experiments on bilingual dictionary induction and cross-lingual word similarity on three standard language pairs, English–Italian, English–German, and English–Finnish.

	English–Italian				English–German				English–Finnish			
	5,000	25	num	iden	5,000	25	num	iden	5,000	25	num	iden
Mikolov et al. (2013c)	34.93	00.00	0.00	1.87	35.00	0.00	0.07	19.20	25.91	0.00	0.00	7.02
Xing et al. (2015)	36.87	0.00	0.13	27.13	41.27	0.07	0.53	38.13	28.23	0.07	0.56	17.95
Zhang et al. (2016)	36.73	0.07	0.27	28.07	40.80	0.13	0.87	38.27	28.16	0.14	0.42	17.56
Artetxe et al. (2016)	39.27	0.07	0.40	31.07	41.87	0.13	0.73	41.53	**30.62**	0.21	0.77	22.61
Artetxe et al. (2017)	39.67	37.27	39.40	39.97	40.87	39.60	40.27	40.67	28.72	**28.16**	26.47	27.88
Ours (1:1)	41.00	39.63	40.47	41.07	**42.60**	**42.40**	**42.60**	**43.20**	29.78	0.07	3.02	**29.76**
Ours (1:1, rank constr.)	**42.47**	**41.13**	**41.40**	**41.80**	41.93	**42.40**	41.93	41.47	28.23	27.04	**27.60**	27.81

Table 1: Precision at 1 (P@1) scores for bilingual lexicon induction of different models with different seed dictionaries and languages on the full vocabulary.

		en-it	en-de	
	Dict	WS	RG	WS
Mikolov et al. (2013c)	5k	.627	.643	.528
Xing et al. (2015)	5k	.614	.700	.595
Zhang et al. (2016)	5k	.616	.704	.596
Artetxe et al. (2016)	5k	.617	.716	.597
Artetxe et al. (2017)	5k	.624	.742	.616
	25	.626	.749	.612
	num	**.628**	.739	.604
Ours (1:1)	5k	.621	.733	**.618**
	25	.621	.740	.617
	num	.624	.743	.617
Ours (1:1, rank constr.)	5k	.623	.741	.609
	25	.622	.753	.609
	num	.625	**.755**	.611

Table 2: Spearman correlations on English–Italian and English–German cross-lingual word similarity datasets.

6.1 Experimental Details

Datasets For bilingual dictionary induction, we use the English–Italian dataset by Dinu et al. (2015) and the English–German and English–Finnish datasets by Artetxe et al. (2017). For cross-lingual word similarity, we use the RG-65 and WordSim-353 cross-lingual datasets for English–German and the WordSim-353 cross-lingual dataset for English–Italian by Camacho-Collados et al. (2015).

Monolingual Embeddings We follow Artetxe et al. (2017) and train monolingual embeddings with word2vec, CBOW, and negative sampling (Mikolov et al., 2013a) on a 2.8 billion word corpus for English (ukWaC + Wikipedia + BNC), a 1.6 billion word corpus for Italian (itWaC), a 0.9 billion word corpus for German (SdeWaC), and a 2.8 billion word corpus for Finnish (Common Crawl).

Seed dictionaries Following Artetxe et al. (2017), we use dictionaries of 5,000 words, 25 words, and a numeral dictionary consisting of words matching the `[0-9]+` regular expression in both vocabularies.[9] In line with Søgaard et al. (2018), we additionally use a dictionary of identically spelled strings in both vocabularies.

Implementation details Similar to Artetxe et al. (2017), we stop training when the improvement on the average cosine similarity for the induced dictionary is below 1×10^{-6} between succeeding iterations. Unless stated otherwise, we induce a dictionary of 200,000 source and 200,000 target words as in previous work (Mikolov et al., 2013c; Artetxe et al., 2016). For optimal 1:1 alignment, we have observed the best results by keeping the top $k = 3$ most similar target words. If using a rank constraint, we restrict the matching in the E-step to the top 40,000 words in both languages.[10] Finding an optimal alignment on the $200{,}000 \times 200{,}000$ graph takes about 25 minutes on CPU;[11] with a rank constraint, matching takes around three minutes.

Baselines We compare our approach with and without the rank constraint to the original bilingual mapping approach by Mikolov et al. (2013c). In addition, we compare with Zhang et al. (2016) and Xing et al. (2015) who augment the former with an orthogonality constraint and normalization and an orthogonality constraint respectively. Finally, we compare with Artetxe et al. (2016) who add dimension-wise mean centering to Xing et al. (2015), and Artetxe et al. (2017).

Both Mikolov et al. (2013c) and Artetxe et al. (2017) are special cases of our famework and comparisons to these approaches thus act as an ablation study. Specifically, Mikolov et al. (2013c) does not

---

[9] The resulting dictionaries contain 2772, 2148, and 2345 entries for English–{Italian, German, Finnish} respectively.

[10] We validated both values with identical strings using the 5,000 word lexicon as validation set on English–Italian.

[11] Training takes a similar amount of time as (Artetxe et al., 2017) due to faster convergence.

employ orthogonal Procrustes, but rather allows the learned matrix Ω to range freely. Likewise, as discussed in §5, Artetxe et al. (2017) make use of a Viterbi EM style algorithm with a different prior over edge sets.[12]

6.2 Results

We show results for bilingual dictionary induction in Table 1 and for cross-lingual word similarity in Table 2. Our method with a 1:1 prior outperforms all baselines on English–German and English–Italian.[13] Interestingly, the 1:1 prior by itself fails on English–Finnish with a 25 word and numerals seed lexicon. We hypothesize that the prior imposes too strong of a constraint to find a good solution for a distant language pair from a poor initialization. With a better—but still weakly supervised—starting point using identical strings, our approach finds a good solution. Alternatively, we can mitigate this deficiency effectively using a rank constraint, which allows our model to converge to good solutions even with a 25 word or numerals seed lexicon. The rank constraint generally performs similarly or boosts performance, while being about 8 times faster. All approaches do better with identical strings compared to numerals, indicating that the former may be generally suitable as a default weakly-supervised seed lexicon.

On cross-lingual word similarity, our approach yields the best performance on WordSim-353 and RG-65 for English–German and is only outperformed by Artetxe et al. (2017) on English–Italian Wordsim-353.

7 Analysis

Vocabulary sizes The beneficial contribution of the rank constraint demonstrates that in similar languages, many frequent words will have one-to-one matchings, while it may be harder to find direct matches for infrequent words. We would thus expect the latent variable model to perform better if we only learn dictionaries for the top n most frequent words in both languages. We show results for our approach in comparison to the baselines in Figure 2 for English–Italian using a 5,000 word seed lexicon across vocabularies consisting of different numbers n of the most frequent words[14].

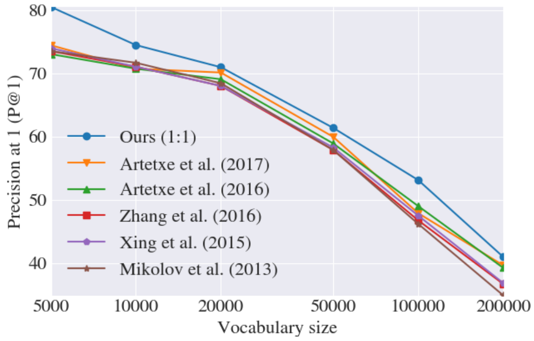

Figure 2: Bilingual dictionary induction results of our method and baselines for English–Italian with a 5,000 word seed lexicon across different vocabulary sizes.

The comparison approaches mostly perform similar, while our approach performs particularly well for the most frequent words in a language.

Different priors An advantage of having an explicit prior as part of the model is that we can experiment with priors that satisfy different assumptions. Besides the 1:1 prior, we experiment with a 2:2 prior and a 1:2 prior. For the 2:2 prior, we create copies of the source and target words V'_{src} and V'_{trg} and add these to our existing set of vertices $V' = (V_{trg} + V'_{trg}, V_{src} + V'_{src})$. We run the Viterbi E-step on this new graph G' and merge matched pairs of words and their copies in the end. Similarly, for the 1:2 prior, which allows one source word to be matched to two target words, we augment the vertices with a copy of the source words V'_{src} and proceed as above. We show results for bilingual dictionary induction with different priors across different vocabulary sizes in Figure 3.

The 2:2 prior performs best for small vocabulary sizes. As solving the linear assignment problem for larger vocabularies becomes progressively more challenging, the differences between the priors become obscured and their performance converges.

Hubness problem We analyze empirically whether the prior helps with the hubness problem. Following Lazaridou et al. (2015), we define the *hubness* $N_k(y)$ at k of a target word y as follows:

$$N_k(y) = |\{x \in Q \mid y \in NN_k(x, G)\}| \qquad (14)$$

where Q is a set of query source language words and $NN_k(x, G)$ denotes the k nearest neighbors

---

[12] Other recent improvements such as symmetric reweighting (Artetxe et al., 2018) are orthogonal to our method, which is why we do not explicitly compare to them here.

[13] Note that results are not directly comparable to (Conneau et al., 2018) due to the use of embeddings trained on different monolingual corpora (WaCKy vs. Wikipedia).

[14] We only use the words in the 5,000 word seed lexicon that are contained in the n most frequent words. We do not show results for the 25 word seed lexicon and numerals as they are not contained in the smallest n of most frequent words.

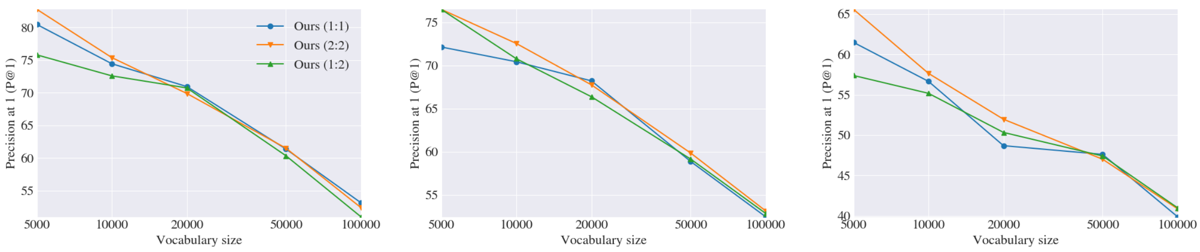

(a) English–Italian (b) English–German (c) English–Finnish

Figure 3: Bilingual dictionary induction results of our method with different priors using a 5,000 word seed lexicon across different vocabulary sizes.

Artetxe et al. (2017)	Ours (1:1)
luis (20)	gleichgültigkeit - '*indifference*' (14)
ungarischen - '*Hungarian*' (18)	heuchelei - '*hypocrisy*' (13)
jorge (17)	ahmed (13)
mohammed (17)	ideologie - '*ideology*' (13)
gewiß - '*certainly*' (17)	eduardo (13)

Table 3: Hubs in English–German cross-lingual embedding space with degree of hubness. Non-name tokens are translated.

	en-tr	en-bn	en-hi	et-fi
Artetxe et al. (2017)	28.93	0.87	2.07	30.18
Ours (1:1)	38.73	2.33	10.47	33.79
Ours (1:1, rank constr.)	**42.40**	**11.93**	**31.80**	**34.78**

Table 4: Bilingual dictionary induction results for English-{Turkish, Bengali, Hindi} and Estonian-Finnish.

of x in the graph G.[15] In accordance with Lazaridou et al. (2015), we set $k = 20$ and use the words in the evaluation dictionary as query terms. We show the target language words with the highest hubness using our method and Artetxe et al. (2017) for English–German with a 5,000 seed lexicon and the full vocabulary in Table 3.[16]

Hubs are fewer and occur less often with our method, demonstrating that the prior—to some extent—aids with resolving hubness. Interestingly, compared to Lazaridou et al. (2015), hubs seem to occur less often and are more meaningful in current cross-lingual word embedding models.[17] For instance, the neighbors of 'gleichgültigkeit' all relate to indifference and words appearing close to 'luis' or 'jorge' are Spanish names. This suggests that the prior might also be beneficial in other ways, e.g. by enforcing more reliable translation pairs for subsequent iterations.

Low-resource languages Cross-lingual embeddings are particularly promising for low-resource languages, where few labeled examples are typically available, but are not adequately reflected in current benchmarks (besides the English–Finnish language pair). We perform experiments with our method with and without a rank constraint and Artetxe et al. (2017) for three truly low-resource language pairs, English–{Turkish, Bengali, Hindi}. We additionally conduct an experiment for Estonian-Finnish, similarly to Søgaard et al. (2018). For all languages, we use fastText embeddings (Bojanowski et al., 2017) trained on Wikipedia, the evaluation dictionaries provided by Conneau et al. (2018), and a seed lexicon based on identical strings to reflect a realistic use case. We note that English does not share scripts with Bengali and Hindi, making this even more challenging. We show results in Table 4.

Surprisingly, the method by Artetxe et al. (2017) is unable to leverage the weak supervision and fails to converge to a good solution for English-Bengali and English-Hindi.[18] Our method without a rank constraint significantly outperforms Artetxe et al. (2017), while particularly for English-Hindi the rank constraint dramatically boosts performance.

Error analysis To illustrate the types of errors the model of Artetxe et al. (2017) and our method with a rank constraint make, we query both of them with words from the test dictionary of Artetxe et al.

---

[15] In other words, the hubness of a target word measures how often it occurs in the neighborhood of the query terms.

[16] We verified that hubs are mostly consistent across runs and similar across language pairs.

[17] Lazaridou et al. (2015) observed mostly rare words with N_{20} values of up to 50 and many with $N_{20} > 20$.

[18] One possible explanation is that Artetxe et al. (2017) particularly rely on numerals, which are normalized in the fastText embeddings.

Query	Gold	Artetxe et al. (2017)	Ours
unregierbar	ungovernable	untenable	uninhabitable
nikolai	nikolaj	feodor	nikolai
memoranden	memorandums	communiqus	memos
argentinier	argentinians	brazilians	argentines
trostloser	bleaker	dreary	dark-coloured
umverteilungen	redistributions	inequities	reforms
modischen	modish	trend-setting	modish
tranquilizer	tranquillizers	clonidine	opiates
sammelsurium	hotchpotch	assortment	mishmash
demagogie	demagogy	opportunism	demagogy
andris	andris	rehn	viktor
dehnten	halmahera	overran	stretched
deregulieren	deregulate	deregulate	liberalise
eurokraten	eurocrats	bureaucrats	eurosceptics
holte	holte	threw	grabbed
reserviertheit	aloofness	disdain	antipathy
reaktiv	reactively	reacting	reactive
danuta	danuta	julie	monika
scharfblick	perspicacity	sagacity	astuteness

Table 5: Example translations for German-English.

(2017) in German and seek their nearest neighbours in the English embedding space. P@1 over the German-English test set is 36.38 and 39.18 for Artetxe et al. (2017) and our method respectively. We show examples where nearest neighbours of the methods differ in Table 5.

Similar to Kementchedjhieva et al. (2018), we find that morphologically related words are often the source of mistakes. Other common sources of mistakes in this dataset are names that are translated to different names and nearly synonymous words being predicted. Both of these sources indicate that while the learned alignment is generally good, it is often not sufficiently precise.

8 Related work

Cross-lingual embedding priors Haghighi et al. (2008) first proposed an EM self-learning method for bilingual lexicon induction, representing words with orthographic and context features and using the Hungarian algorithm in the E-step to find an optimal 1:1 matching. Artetxe et al. (2017) proposed a similar self-learning method that uses word embeddings, with an implicit one-to-many alignment based on nearest neighbor queries. Vulić and Korhonen (2016) proposed a more strict one-to-many alignment based on symmetric translation pairs, which is also used by Conneau et al. (2018). Our method bridges the gap between early latent variable and word embedding-based approaches and explicitly allows us to reason over its prior.

Hubness problem The hubness problem is an intrinsic problem in high-dimensional vector spaces (Radovanović et al., 2010). Dinu et al. (2015) first observed it for cross-lingual embedding spaces and proposed to address it by re-ranking neighbor lists. Lazaridou et al. (2015) proposed a max-marging objective as a solution, while more recent approaches proposed to modify the nearest neighbor retrieval by inverting the softmax (Smi, 2017) or scaling the similarity values (Conneau et al., 2018).

9 Conclusion

We have presented a novel latent-variable model for bilingual lexicon induction, building on the work of Artetxe et al. (2017). Our model combines the prior over bipartite matchings inspired by Haghighi et al. (2008) and the discriminative, rather than generative, approach inspired by Irvine and Callison-Burch (2013). We show empirical gains on six language pairs and theoretically and empirically demonstrate the application of the bipartite matching prior to solving the hubness problem.

Acknowledgements

The authors acknowledge Edouard Grave and Arya McCarthy, who, aside from being generally awesome, provided feedback post-submission on the ideas. Sebastian is supported by Irish Research Council Grant Number EBPPG/2014/30 and Science Foundation Ireland Grant Number SFI/12/RC/2289, co-funded by the European Regional Development Fund. Ryan is supported by an NDSEG fellowship and a Facebook fellowship.

References

2017. Bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of ICLR*.

Ryan Prescott Adams and Richard S. Zemel. 2011. Ranking via Sinkhorn propagation. *arXiv preprint arXiv:1106.1925*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 451–462.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and Improving Bilingual Word Embedding Mappings with a Multi-Step Framework of Linear Transformations. In *Proceedings of AAAI 2018*.

D. Bertsimas and J.N. Tsitsiklis. 1997. *Introduction to linear optimization*. Athena Scientific.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A Framework for the Construction of Monolingual and Cross-lingual Word Similarity Datasets. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, (April 2016).

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word Translation Without Parallel Data. In *Proceedings of ICLR 2018*.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving Zero-Shot Learning by Mitigating the Hubness Problem. *ICLR 2015 Workshop track*, pages 1–10.

Pascale Fung. 1995. Compiling bilingual lexicon entries from a non-parallel english-chinese corpus. In *Third Workshop on Very Large Corpora*.

John C. Gower and Garmt B. Dijksterhuis. 2004. *Procrustes problems*. Oxford University Press.

Edouard Grave, Armand Joulin, and Quentin Berthet. 2018. Unsupervised Alignment of Embeddings with Wasserstein Procrustes. *arXiv preprint arXiv:1805.11222*.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning Bilingual Lexicons from Monolingual Corpora. In *Proceedings of ACL 2008*, June, pages 771–779.

Roger A. Horn and Charles R. Johnson. 2012. *Matrix Analysis*. Cambridge University Press.

Ann Irvine and Chris Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 518–523, Atlanta, Georgia. Association for Computational Linguistics.

Roy Jonker and Anton Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

Yova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing Procrustes Analysis for Better Bilingual Dictionary Induction. In *Proceedings of CoNLL 2018*.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 130–140, Avignon, France. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, volume 5, pages 79–86.

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and Pollution: Delving into Cross-Space Mapping for Zero-Shot Learning. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 270–280.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2536–2545, Copenhagen, Denmark. Association for Computational Linguistics.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. 2018. Learning latent permutations with Gumbel-Sinkhorn networks. *arXiv preprint arXiv:1802.08665*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR) Workshop*.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013c. Exploiting Similarities among Languages for Machine Translation.

R. M. Neal and G. E. Hinton. 1998. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers.

Milos Radovanović, Alexandros Nanopoulos, and Mirjana Ivanovic. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 320–322, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2018. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*.

Rajhans Samdani, Ming-Wei Chang, and Dan Roth. 2012. Unified expectation maximization. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 688–698, Montréal, Canada. Association for Computational Linguistics.

Peter H. Schönemann. 1966. A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31(1):1–10.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the Limitations of Unsupervised Bilingual Dictionary Induction. In *Proceedings of ACL 2018*.

Leslie G. Valiant. 1979. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201.

Cédric Villani. 2008. *Optimal transport: Old and new*, volume 338. Springer Science & Business Media.

A Volgenant. 1996. Linear and semi-assignment problems: a core oriented approach. *Computers & Operations Research*, 23(10):917–932.

Ivan Vulić and Anna Korhonen. 2016. On the Role of Seed Lexicons in Learning Bilingual Word Embeddings. *Proceedings of ACL*, pages 247–257.

Douglas B. West. 2000. *Introduction to Graph Theory*, 2 edition. Prentice Hall.

Chao Xing, Chao Liu, Dong Wang, and Yiye Lin. 2015. Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. *NAACL-2015*, pages 1005–1010.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten Pairs to Tag Multilingual POS Tagging via Coarse Mapping between Embeddings. In *Proceedings of NAACL-HLT 2016*, pages 1307–1317.

Non-Adversarial Unsupervised Word Translation

Yedid Hoshen
Facebook AI Research

Lior Wolf
Facebook AI Research and Tel Aviv University

Abstract

Unsupervised word translation from non-parallel inter-lingual corpora has attracted much research interest. Very recently, neural network methods trained with adversarial loss functions achieved high accuracy on this task. Despite the impressive success of the recent techniques, they suffer from the typical drawbacks of generative adversarial models: sensitivity to hyper-parameters, long training time and lack of interpretability. In this paper, we make the observation that two sufficiently similar distributions can be aligned correctly with iterative matching methods. We present a novel method that first aligns the second moment of the word distributions of the two languages and then iteratively refines the alignment. Extensive experiments on word translation of European and Non-European languages show that our method achieves better performance than recent state-of-the-art deep adversarial approaches and is competitive with the supervised baseline. It is also efficient, easy to parallelize on CPU and interpretable.

1 Introduction

Inferring word translations between languages is a long-standing research task. Earliest efforts concentrated on finding parallel corpora in a pair of languages and inferring a dictionary by force alignment of words between the two languages. An early example of this approach is the translation achieved using the Rosetta stone.

However, if most languages share the same expressive power and are used to describe similar human experiences across cultures, they should share similar statistical properties. Exploiting statistical properties of letters has been successfully employed by substitution crypto-analysis since at least the 9th century. It seems likely that one can learn to map between languages statistically, by considering the word distributions. As one specific example, it is likely that the set of elements described by the most common words in one language would greatly overlap with those described in a second language.

Another support for the plausibility of unsupervised word translation came with the realization that when words are represented as vectors that encode co-occurrences, the mapping between two languages is well captured by an affine transformation (Mikolov et al., 2013b). In other words, not only that one can expect the most frequent words to be shared, one can also expect the representations of these words to be similar up to a linear transformation.

A major recent trend in unsupervised learning is the use of Generative Adversarial Networks (GANs) presented by Goodfellow et al. (2014), in which two networks provide mutual training signals to each other: the generator and the discriminator. The discriminator plays an adversarial role to a generative model and is trained to distinguish between two distributions. Typically, these distributions are labeled as "real" and "fake", where "fake" denotes the generated samples.

In the context of unsupervised translation (Conneau et al., 2017; Zhang et al., 2017a,b), when learning from a source language to a target language, the "real" distribution is the distribution of the target language and the "fake" one is the mapping of the source distribution using the learned mapping. Such approaches have been shown recently to be very effective when employed on top of modern vector representations of words.

In this work, we ask whether GANs are necessary for achieving the level of success recently demonstrated for unsupervised word translation. Given that the learned mapping is simple and that the concepts described by the two languages are similar, we suggest to directly map every word

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 469–478
Brussels, Belgium, October 31 - November 4, 2018.

in one language to the closest word in the other. While one cannot expect that all words would match correctly for a random initialization, some would match and may help refine the affine transformation. Once an improved affine transformation is recovered, the matching process can repeat.

Naturally, such an iterative approach relies on a good initialization. For this purpose we employ two methods. First, an initial mapping is obtained by matching the means and covariances of the two distributions. Second, multiple solutions, which are obtained stochastically, are employed.

Using multiple stochastic solutions is crucial for languages that are more distant, e.g., more stochastic solutions are required for learning to translate between English and Arabic, in comparison to English and French. Evaluating multiple solutions relies on the ability to automatically identify the true matching without supervision and we present an unsupervised reconstruction-based criterion for determining the best stochastic solution.

Our presented approach is simple, has very few hyper-parameters, and is trivial to parallelize. It is also easily interpretable, since every step of the method has a clear goal and a clear success metric, which can also be evaluated without the ground truth bilingual lexicon. An extensive set of experiments shows that our much simpler and more efficient method is more effective than the state-of-the-art GAN based method.

2 Related Work

The earlier contributions in the field of word translation without parallel corpora were limited to finding matches between a small set of carefully selected words and translations, and relied on co-occurrence statistics (Rapp, 1995) or on similarity in the variability of the context before and after the word (Fung, 1995). Finding translations of larger sets of words was made possible in follow-up work by incorporating a seed set of matching words that is either given explicitly or inferred based on words that appear in both languages or are similar in edit distance due to a shared etymology (Fung and Yee, 1998; Rapp, 1999; Schafer and Yarowsky, 2002; Koehn and Knight, 2002; Haghighi et al., 2008; Irvine and Callison-Burch, 2013; Xia et al., 2016; Artetxe et al., 2017).

For example, Koehn and Knight (2002) matched English with German. Multiple heuristics were suggested based on hand crafted rules, including similarity in spelling and word frequency. A weighted linear combination is employed to combine the heuristics and the matching words are identified in a greedy manner. Haghighi et al. (2008) modeled the problem of matching words across independent corpora as a generative model, in which cross-lingual links are represented by latent variables, and employed an iterative EM method.

Another example that employs iterations was presented by Artetxe et al. (2017). Similarly to our method, this method relies on word vector embeddings, in their case the word2vec method (Mikolov et al., 2013a). Unlike our method, their method is initialized using seed matches.

Our core method incorporates a circularity term, which is also used in (Xia et al., 2016) for the task of NMT and later on in multiple contributions in the field of image synthesis (Kim et al., 2017; Zhu et al., 2017). This term is employed when learning bidirectional transformations to encourages samples from either domain to be mapped back to exactly the same sample when translated to the other domain and back. Since our transformations are linear, this is highly related to employing orthogonality as done in (Xing et al., 2015; Smith et al., 2017; Conneau et al., 2017) for the task of weakly or unsupervised word vector space alignment. Conneau et al. (2017) also employ a circularity term, but unlike our use of it as part of the optimization's energy term, there it is used for validating the solution and selecting hyperparameters.

Very recently, Zhang et al. (2017a,b); Conneau et al. (2017) have proposed completely unsupervised solutions. All three solutions are based on GANs. The methods differ in the details of the adversarial training, in the way that model selection is employed to select the best configuration and in the way in which matching is done after the distributions are aligned by the learned transformation.

Due to the min-max property of GANs, methods which rely on GANs are harder to interpret, since, for example, the discriminator D could focus on a combination of local differences between the distributions. The reliance on a discriminator also means that complex weight dependent metrics are implicitly used, and that these metrics evolve dynamically during training.

Our method does not employ GANs. Alternatives to GANs are also emerging in other domains. For example, generative methods were trained by iteratively fitting random ("noise") vectors by Bojanowski et al. (2017); In the recent image translation work of Chen and Koltun (2017), distinguishability between distribution of images was measured using activations of pretrained networks, a practice that is referred to as the "perceptual loss" (Johnson et al., 2016).

3 Non-Adversarial Word Translation

We present an approach for unsupervised word translation consisting of multiple parts: (i) Transforming the word vectors into a space in which the two languages are more closely aligned, (ii) Mini-Batch Cycle iterative alignment. There is an optional final stage of batch-based finetuning.

Let us define two languages $\mathcal{X}$ and $\mathcal{Y}$, each containing a set of N_X and N_Y words represented by the feature vectors $x_1..x_{N_X}$ and $y_1..y_{N_Y}$ respectively. Our objective is to find the correspondence function $f(n)$ such that for every x_n, $f(n)$ yields the index of the $\mathcal{Y}$ word that corresponds to the word x_n. If a set of possible correspondences is available for a given word, our objective is to predict one member of this set. In this unsupervised setting, no training examples of $f(n)$ are given.

3.1 Approximate Alignment with PCA

Each language consists of a set of words each parameterized by a word vector. A popular example of a word embedding method is FastText (Bojanowski et al., 2016), which uses the internal word co-occurrence statistics for each language. These word vectors are typically not expected to be aligned between languages and since the alignment method we employ is iterative, a good initialization is key.

Let us motivate our approach by a method commonly used in 3D point cloud matching. Let A be a set of 3D points and TA be the same set of points with a rotated coordinate system. Assuming non-isotropic distributions of points, transforming each set of points to its principle axes of variations (using PCA) will align the two point clouds. As noted by Daras et al. (2012), PCA-based alignment is common in the literature of point cloud matching.

Word distributions are quite different from 3D point clouds: They are much higher dimensional, and it is not obvious a priori that different languages present different "views" of the same "object" and share exactly the same axes of variation. The success of previous results, e.g. (Conneau et al., 2017), to align word vectors between languages using an orthonormal transformation does give credence to this approach. Our method relies on the assumption that many language pairs share some principle axes of variation. The empirical success of PCA initialization in this work supports this assumption.

For each language $[\mathcal{X}, \mathcal{Y}]$, we first select the N most frequent word vectors. In our implementation, we use $N = 5000$ and employ FastText vectors of dimension $D = 300$. We project the word vectors, after centering, to the top p principle components (we use $p = 50$).

3.2 Mini-Batch Cycle Iterative Closest Point

Although projecting to the top principle axes of variation would align a rotated non-isotropic point cloud, it does not do so in the general case. This is due to languages having different word distributions and components of variation.

We therefore attempt to find a transformation T that will align every word x_i from language $\mathcal{X}$ to a word $y_{f(i)}$ in language $\mathcal{Y}$. The objective is therefore to minimize:

$$\underset{T}{\operatorname{argmin}} \sum_i \min_{f(i)} |y_i - Tx_{f(i)}| \tag{1}$$

Eq. 1 is difficult to optimize directly and various techniques have been proposed for its optimization. One popular method used in 3D point cloud alignment is Iterative Closest Point (ICP). ICP solves Eq. 1 iteratively in two steps.

1. For each y_j, find the nearest Tx_i. We denote its index by $f_y(j) = i$
2. Optimize for T in $\sum_j \|y_j - Tx_{f_y(j)}\|$

In this work, we use a modified version of ICP which we call Mini-Batch Cycle ICP (MBC-ICP). MBC-ICP learns transformations T_{xy} for $\mathcal{X} \rightarrow \mathcal{Y}$ and T_{yx} for $\mathcal{Y} \rightarrow \mathcal{X}$. We include cycle-constraints ensuring that a word x transformed to the $\mathcal{Y}$ domain and back is unchanged (and similarly for every $\mathcal{Y} \rightarrow \mathcal{X} \rightarrow \mathcal{Y}$ transformation). The strength of the cycle constraints is parameterized by λ (we have $\lambda = 0.1$). We compute the nearest neighbor matches at the beginning of each epoch, and then optimize transformations T_{yx} and T_{xy} using mini-batch SGD with mini-batch size 128. Mini-batch rather than full-batch optimization greatly increases the success of the method. Experimental comparisons can be seen in the results section. Note we only compute the nearest neighbors at the beginning of each epoch, rather than for each mini-batch due to the computational expense.

Every iteration of the final MBC-ICP algorithm therefore becomes:

1. For each y_j, find the nearest $T_{xy}x_i$. We denote its index by $f_y(j)$
2. For each x_i, find the nearest $T_{yx}y_j$. We denote its index by $f_x(i)$
3. Optimize T_{xy} and T_{yx} using mini-batch SGD for a single epoch of $\{x_i\}$ and $\{y_j\}$ on: $\sum_j \|y_j - T_{xy}x_{f_y(j)}\| + \sum_i \|x_i - T_{yx}y_{f_x(i)}\| + \lambda \sum_i \|x_i - T_{yx}T_{xy}x_i\| + \lambda \sum_j \|y_j - T_{xy}T_{yx}y_j\|$

A good initialization is important for ICP-type methods. We therefore begin with the projected data in which the transformation is assumed to be relatively small and initialize transformations T_{xy} and T_{yx} with the identity matrix. We denote this step PCA-MBC-ICP.

Once PCA-MBC-ICP has generated the correspondences functions $f_x(i)$ and $f_y(j)$, we run a MBC-ICP on the original $300D$ word vectors (no PCA). We denote this step: RAW-MBC-ICP. We initialize the optimization using $f_x(i)$ and $f_y(j)$ learned before, and proceed with MBC-ICP. At the end of this stage, we recover transformations $\bar{T}_{xy}$ and $\bar{T}_{yx}$ that transform the $300D$ word vectors from $\mathcal{X} \rightarrow \mathcal{Y}$ and $\mathcal{Y} \rightarrow \mathcal{X}$ respectively.

Reciprocal pairs: After several iterations of MBC-ICP, the estimated transformations become quite reliable. We can therefore use this transformation to identify the pairs that are likely to be correct matches. We use the reciprocity heuristic: For every word $y \in \mathcal{Y}$ we find the nearest transformed word from the set $\{T_{xy}x | x \in X\}$. We also find the nearest neighbors for the $\mathcal{Y} \rightarrow \mathcal{X}$ transformation. If a pair of words is matched in both $\mathcal{X} \rightarrow \mathcal{Y}$ and $\mathcal{Y} \rightarrow \mathcal{X}$ directions, the pair is denoted reciprocal. During RAW-MBC-ICP, we use only reciprocal pairs, after the 50th epoch (this parameter is not sensitive).

In summary: we run PCA-MBC-ICP on the $5k$ most frequent words after transformation to principle components. The resulting correspondences $f_x(i)$ and $f_y(j)$ are used to initialize a RAW-MBC-ICP on the original 300D data (rather than PCA), using reciprocal pairs. The output of the method are transformation matrices T_{xy} and T_{yx}.

3.3 Fine-tuning

MBC-ICP is able to achieve very competitive performance without any further finetuning or use of large corpora. GAN-based methods on the other hand require iterative finetuning (Conneau et al., 2017; Hoshen and Wolf, 2018) to achieve competitive performance. To facilitate comparison with such methods, we also add a variant of our method with identical finetuning to (Conneau et al., 2017). As we show in the results section, fine-tuning European languages typically results in small improvements in accuracy (1-2%) for our method, in comparison to 10-15% for the previous work. Following (Xing et al., 2015; Conneau et al., 2017), fine-tuning is performed by running the Procrustes method iteratively on the full vocabulary of 200k words, initialized with the final transformation matrix from MBC-ICP. The Procrustes method uses SVD to find the optimal orthonormal matrix between $\mathcal{X}$ and $\mathcal{Y}$ given approximate matches. The new transformation is used to finetune the approximate matches. We run 5 iterations of successive transformation and matching estimation steps.

3.4 Matching Metrics

Although we optimize the nearest neighbor metric, we found that in accordance with (Conneau et al., 2017), neighborhood retrieval methods such as Inverted Soft-Max (ISF) (Smith et al., 2017) and Cross-domain Similarity Local Scaling (CSLS) improve final retrieval performance. We therefore evaluate using CSLS. The similarity between a word $x \in \mathcal{X}$ and a word $y \in \mathcal{Y}$ is computed as $2\cos(T_{xy}x, y) - r(T_{xy}x) - r(y)$, where $r(.)$ is the average cosine similarity between the word and its 10-NN in the other domain.

4 Multiple Stochastic Solutions

Our approach utilizes multiple stochastic solutions, to provide a good initialization for the MBC-ICP algorithm. There are two sources of stochasticity in our system: (i) The randomized nature of the PCA algorithm (it uses random matrices (Liberty et al., 2007)) (ii) The order of the training samples (the mini-batches) when training the transformations.

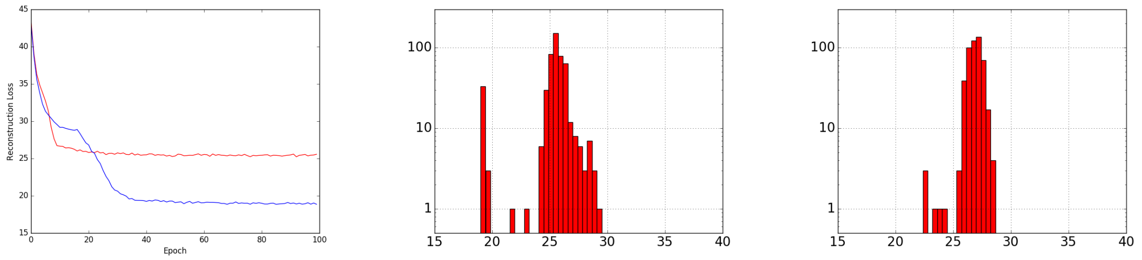

(a) (b) (c)

Figure 1: (a) Evolution of reconstruction loss as a function of epoch number for successful (Blue) and unsuccessful runs (Red). (b) The final reconstruction loss distribution for En-Fr. (c) A similar histogram for En-Ar.

The main issue faced by unsupervised learning in the case of multiple solutions, is either (i) choosing the best solution in case of a fixed parallel run budget, or (ii) finding a good stopping criterion if attempting to minimize the number of runs serially.

We use the reconstruction cost as an unsupervised metrics for measuring convergence of MBC-ICP. Specifically, we measure how closely every $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ is reconstructed by a transformed word from the other domain.

$$\sum_j \|y_j - T_{xy} x_{f_y(j)}\| + \sum_i \|x_i - T_{yx} y_{f_x(i)}\| \quad (2)$$

Although for isotropic distributions this has many degenerate solutions, empirically we find that values that are significantly lower than the median almost always correspond to a good solution.

The optimization profile of MBC-ICP is predictable and easily lends itself for choosing effective termination criteria. The optimization profile of a successfully converged and non-converging runs are presented in Fig. 1(a). The reconstruction loss clearly distinguish between the converged and non-converging runs. Fig. 1(b,c) presents the distribution of final reconstruction costs for 500 different runs for En-Fr and En-Ar.

5 Experiments

We evaluated our method extensively to ensure that it is indeed able to effectively and efficiently perform unsupervised word translation. As a strong baseline, we used the code and datasets from the MUSE repository by Conneau et al. (2017)[1]. Care was taken in order to make sure that we report these results as fairly as possible: (1) the results from the previous work were copied as is, except for En-It, where our runs indicated better baseline results. (2) For languages not reported, we ran the code with multiple options and report the best results obtained. One crucial option for GAN was whether to center the data or not. From communication with the authors we learned that, in nearly all non-European languages, centering the data is crucial. For European languages, not centering gave better results. For Arabic, centering helps in one direction but is very detrimental in the other. In all such cases, we report the best baseline result per direction. (3) For the supervised baseline, we report both the results from the original paper (in Tab. 1) and the results post Procrustes finetuning, which are better (Tab. 2). (4) Esperanto is not available in the MUSE repository at this time. We asked the authors for the data and will update the paper once available. Currently we are able to say (without the supervision data) that our method converges on En-Eo and Eo-En.

The evaluation concentrated on two aspects of the translation: (i) Word Translation Accuracy measured by the fraction of words translated to a correct meaning in the target language, and (ii) Runtime of the method.

We evaluated our method against the best methods from (Conneau et al., 2017). The supervised baseline method learns an alignment from 5k supervised matches using the Procrustes method. The mapping is then refined using the Procrustes method and CSLS matching on 200k unsupervised word vectors in the source and target languages. The unsupervised method proposed by (Conneau et al., 2017), uses generative adversarial domain mapping between the word vectors of the 50k most common words in each language. The mapping is then refined using the same procedure that is used in the supervised baseline.

A comparison of the word translation accura-

---

[1] https://github.com/facebookresearch/MUSE

cies before finetuning can be seen in Tab. 1. Our method significantly outperforms the method of (Conneau et al., 2017) on all of the evaluated European language pairs. Additionally, for these languages, our method performs comparably to the supervised baseline on all pairs except En-Ru for which supervision seems particularly useful. The same trends are apparent for simple nearest neighbors and CSLS although CSLS always performs better. For non-European languages, none of the unsupervised methods succeeds on all languages. We found that the GAN baseline fails on Farsi, Hindu, Bengali, Vietnamese and one direction of Japanese and Indonesian while our method does not succeed on Chinese, Japanese and Vietnamese. We conclude that the methods have complementary strengths, our method doing better on more languages. On languages where both methods succeed, MBC-ICP tends to do much better.

We present a comparison between the methods after finetuning and using the CSLS metric in Tab. 2. All methods underwent the same finetuning procedure. We can see that our method still outperforms the GAN method and is comparable to the supervised baseline on European languages. Another observation is that on most European language pairs, finetuning only makes a small difference for our method (1-2%). An unaligned vocabulary of 7.5k is sufficient to achieve most of the accuracy. This is in contrast with the GAN, that benefits greatly from finetuning on 200k words. Non-European language and English pairs are typically more challenging, finetuning helps much more for all unsupervised methods.

It is interesting to examine the languages on which each method could not converge. They typically fall into geographical and semantic clusters. The GAN method failed on Arabic and Hebrew, Hindu, Farsi and Bengali. Whereas our method failed on Japanese and Chinese. We suspect that different statistical properties favor each method.

We also compare the different methods in terms of training time required by the method. We emphasize that our method is trivially parallelizable, simply by splitting the random initializations between workers. The run time of each solution of MBC-ICP is 47 seconds on CPU. The run time of all solutions can therefore be as low as a single run, at linear increase in compute resources. As it runs on CPU, parallelization is not expensive. The average number of runs required for convergence depends on the language pair (see below, Fig. 2). We note that our method learns translations for both languages at the same time.

Table 1: Comparison of word translation accuracy (%) - without finetuning. Bold: best unsupervised method.

Pair	Supervised		Unsupervised			
	Baseline		GAN		Ours	
	nn	csls	nn	csls	nn	csls
European Languages						
En-Es	77.4	81.4	69.8	75.7	75.9	**81.1**
Es-En	77.3	82.9	71.3	79.7	76.0	**82.1**
En-Fr	74.9	81.1	70.4	77.8	74.8	**81.5**
Fr-En	76.1	82.4	61.9	71.2	75.0	**81.3**
En-De	68.4	73.5	63.1	70.1	66.9	**73.7**
De-En	67.7	72.4	59.6	66.4	67.1	**72.7**
En-Ru	47.0	51.7	29.1	37.2	36.8	**44.4**
Ru-En	58.2	63.7	41.5	48.1	48.4	**55.6**
En-It	75.7	77.3	54.3	65.1	71.1	**77.0**
It-En	73.9	76.9	55.0	64.0	70.4	**76.6**
Non-European Languages						
En-Fa	25.7	33.1	*	*	19.6	**29.0**
Fa-En	33.5	38.6	*	*	28.3	**28.3**
En-Hi	23.8	33.3	*	*	19.4	**30.3**
Hi-En	34.6	42.8	*	*	30.5	**38.9**
En-Bn	10.3	15.8	*	*	9.7	**13.5**
Bn-En	21.5	24.6	*	*	7.1	**14.5**
En-Ar	31.3	36.5	18.9	23.5	26.9	**33.3**
Ar-En	45.0	49.5	28.6	31	39.8	**45.5**
En-He	10.3	15.8	17.9	22.7	31.3	**38.9**
He-En	21.5	24.6	37.3	39.1	43.4	**50.8**
En-Zh	40.6	42.7	12.7	**16.0**	*	*
Zh-En	30.2	36.7	18.7	**25.1**	*	*
En-Ja	2.4	1.7	*	*	*	*
Ja-En	0.0	0.0	3.1	**3.6**	*	*
En-Vi	25.0	41.3	*	*	*	*
Vi-En	40.6	55.3	*	*	*	*
En-Id	55.3	65.6	18.9	23.5	39.4	**57.1**
Id-En	58.3	65.0	*	*	37.1	**58.1**

*Failed to converge

The current state-of-the-art baseline by (Conneau et al., 2017) requires around 3000 seconds on the GPU. It is not obvious how to parallelize such a method efficiently. It requires about 30 times longer to train than our method (with parallelization) and is not practical on a multi-CPU platform. The optional refinement step requires about 10 minutes. The performance increase of refinement for our method are typically quite modest and can be be skipped at the cost of 1-2% in accuracy, the GAN however requires finetuning to obtain competitive results. Another obvious advan-

Table 2: Word translation accuracy (%) - after finetuning and using CSLS. Bold: best unsupervised methods.

Pair	Supervised	Unsupervised	
	Baseline	GAN	Ours
European Languages			
En-Es	82.4	81.7	**82.1**
Es-En	83.9	83.3	**84.1**
En-Fr	82.3	**82.3**	**82.3**
Fr-En	83.2	82.1	**82.9**
En-De	75.3	74.0	**74.7**
De-En	72.7	72.2	**73.0**
En-Ru	50.7	44.0	**47.5**
Ru-En	63.5	59.1	**61.8**
En-It	78.1	76.9	**77.9**
It-En	78.1	76.7	**77.5**
Non-European Languages			
En-Fa	32.6	*	**34.6**
Fa-En	40.2	*	**41.5**
En-Hi	34.5	*	**34.6**
Hi-En	44.8	*	**44.5**
En-Bn	16.6	*	**14.7**
Bn-En	24.1	*	**21.9**
En-Ar	34.5	**35.3**	35.1
Ar-En	49.7	49.7	**50.6**
En-He	41.1	**41.6**	40.5
He-En	54.9	52.6	**52.9**
En-Zh	42.7	**32.5**	*
Zh-En	36.7	**31.4**	*
En-Ja	1.7	*	*
Ja-En	0.0	**4.2**	*
En-Vi	44.6	*	*
Vi-En	56.9	*	*
En-Id	68.0	67.8	**68.0**
Id-En	68.0	66.6	**68.0**

*Failed to converge

tage is that our method does not require a GPU.

Implementation: We used 100 iterations for the PCA-MBC-ICP stage on 50 PCA word vectors. This was run in parallel over 500 stochastic solutions. We selected the solution with the smallest unsupervised reconstitution criterion. This solution was used to initialize RAW-MBC-PCA, which we run for 100 iterations on the raw word vectors. The latter 50 iterations of RAW-MBC-ICP were carried out with only reciprocal pairs contributing to the optimization. Results were typically not sensitive to hyper-parameter choice, although longer optimization generally resulted in better performance.

Table 3: En-Es accuracy with and without PCA

Method	En-Es	Es-En
No PCA	0.0%	0.0%
With 300 PCs	0.0%	0.0%
With 50 PCs	**82.2%**	**83.8%**

Table 4: Fraction of converging runs per stochasticity.

Method	En-Es	En-Ar
No randomization	0.0%	0.0%
Randomized Ordering	0.0%	0.0%
Randomized PCA	9.8%	0.0%
Randomized Ordering + PCA	**16.8%**	**1.2%**

Ablation Analyses There are three important steps for the convergence of the ICP step: (i) PCA, (ii) Dimensionality reduction, (iii) Multiple stochastic solutions. In Tab. 3 we present the ablation results on the En-Es pair with PCA and no dimensionality reduction, with only the top 50 PCs and without PCA at all (best run out of 500 chosen using the unsupervised reconstruction loss). We can observe that the convergence rate without PCA and with PCA but without dimensionality reduction is much lower than with PCA, the best run without PCA has not succeeded in obtaining a good translation. This provides evidence that both PCA and dimensionality reduction are essential for the success of the method.

We experimented with the different factors of randomness between runs, to understand the causes of diversity allowing for convergence in the more challenging language pairs (such as En-Ar). We performed the following four experiments: i. Fixing PCA and Batch Ordering. ii. Fixing all data batches to have the same ordering in all runs, iii. Fix the PCA bases of all runs, iv. Let both PCA and batch ordering vary between runs.

Tab. 4 compares the results on En-Es and En-Ar for the experiments described above. It can be seen that using both sources of stochasticity is usually better. Although there is some probability the PCA will result in aligned principle components between the two languages, this usually does not happen and therefore using stochastic PCA is highly beneficial.

Convergence Statistics In Fig. 2 we present the statistics for all language pairs with Procrustes-ICP (P-ICP) vs MBC-ICP. In P-ICP, we first calculate the matches for the vocabulary, and then perform a batch estimate of the transformation using the P-ICP method (starting from PCA word

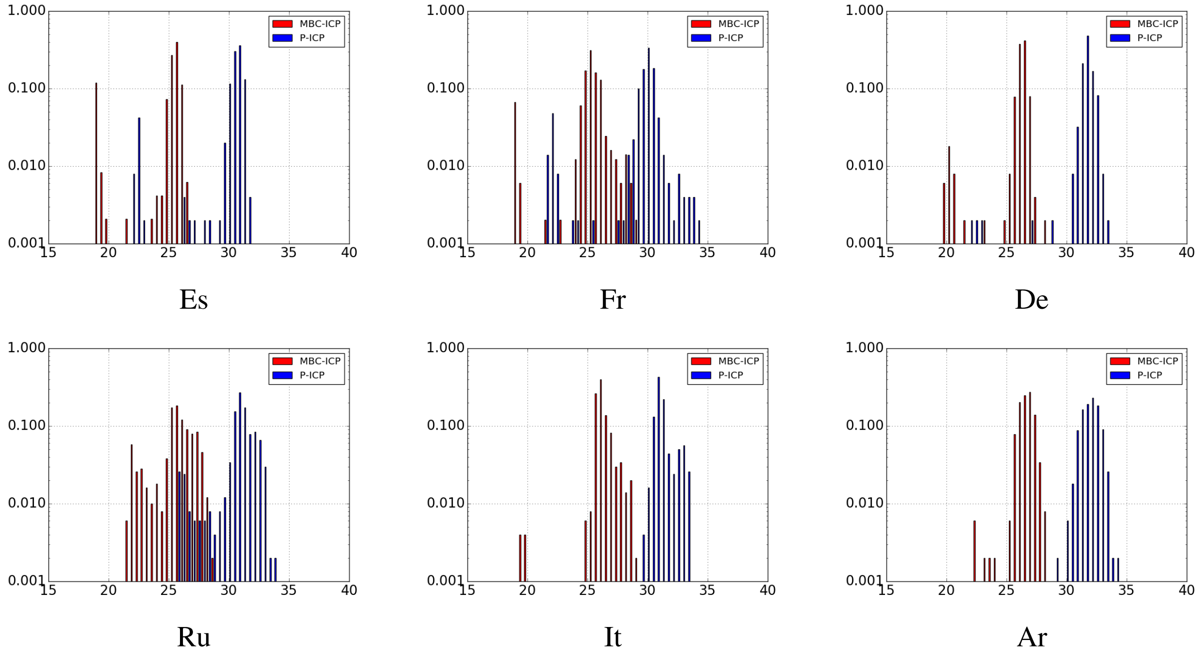

Figure 2: Histograms of the Reconstruction metric across 500 ICP runs for MBC-ICP (Red) and P-ICP (Blue). The comparison is shown for En-Es, En-Fr, En-De, En-Ru, En-It, En-Ar. On average MBC-ICP converges to much better minima. We can observe that MBC-ICP has many more converging runs than P-ICP. In fact for En-It and En-Ar, P-ICP does not converge even once in 500 runs.

vectors and T_{xy} initialized at identity). The only source of stochasticity in P-ICP is the PCA where in MBC-ICP the order of mini-batches provides further stochasticity. Adding random noise to the mapping initialization was not found to help. Each plot shows the histogram in log space for the number of runs that achieved unsupervised reconstruction loss within the range of the bin. The converged runs with lower reconstruction values typically form a peak which is quite distinct from the non-converged runs allowing for easy detection of converged runs. The rate of convergence generally correlates with our intuition for distance between languages (En-Ar much lower than En-Fr), although there are exceptions.

MBC-ICP converges much better than P-ICP: For the language pairs with a wide convergence range (En-Es, En-Fr, En-Ru) we can see that MBC-ICP converged on many more runs than P-ICP. For the languages with a narrow convergence range (En-Ar, En-It), P-ICP was not able to converge at all. We therefore conclude that the mini-batch update and batch-ordering stochasticity increase the convergence range and is important for effective unsupervised matching.

6 Conclusions

We have presented an effective technique for unsupervised word-to-word translation. Our method is simple and non-adversarial. We showed empirically that our method outperforms current state-of-the-art GAN methods in terms of pre and post finetuning word translation accuracy. Our method runs on CPU and is much faster than current methods when using parallelization. This will enable researchers from labs that do not possess graphical computing resources to participate in this exciting field. The proposed method is interpretable, i.e. every stage has an intuitive loss function with an easy to understand objective.

It is interesting to consider the relative performance between language pairs. Typically more related languages yielded better performance than more distant languages (but note that Indonesian performed better than Russian when translated to English). Even more interesting is contrasting the better performance of our method on West and South Asian languages, and GAN's better performance on Chinese.

Overall, our work highlights the potential benefits of considering alternatives to adversarial methods in unsupervised learning.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 451–462.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. 2017. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776* .

Qifeng Chen and Vladlen Koltun. 2017. Photographic image synthesis with cascaded refinement networks. In *ICCV*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087* .

Petros Daras, Apostolos Axenopoulos, and Georgios Litos. 2012. Investigating the effects of multiple factors towards more accurate 3-d object retrieval. *IEEE Transactions on Multimedia* 14(2):374–388.

Pascale Fung. 1995. Compiling bilingual lexicon entries from a non-parallel english-chinese corpus. In *Proceedings of the Third Workshop on Very Large Corpora*. Massachusetts Institute of Technology Cambridge, pages 173–183.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 414–420.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. pages 2672–2680.

Aria Haghighi, Percy Liang, Taylor Berg-kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-HLT*.

Yedid Hoshen and Lior Wolf. 2018. Identifying analogies across domains. In *International Conference on Learning Representations*.

Ann Irvine and Chris Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *HLT-NAACL*. pages 518–523.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*. Springer, pages 694–711.

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. 2017. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192* .

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition-Volume 9*. Association for Computational Linguistics, pages 9–16.

Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. 2007. Randomized algorithms for the low-rank approximation of matrices. *PNAS* .

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168* .

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 320–322.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, pages 519–526.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *proceedings of the 6th conference on Natural language learning-Volume 20*. Association for Computational Linguistics, pages 1–7.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859* .

Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. *arXiv preprint arXiv:1611.00179* .

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *HLT-NAACL*. pages 1006–1011.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017a. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1959–1970.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1934–1945.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

Semi-Autoregressive Neural Machine Translation

Chunqi Wang[*] **Ji Zhang** **Haiqing Chen**
Alibaba Group
{shiyuan.wcq, zj122146, haiqing.chenhq}@alibaba-inc.com

Abstract

Existing approaches to neural machine translation are typically autoregressive models. While these models attain state-of-the-art translation quality, they are suffering from low parallelizability and thus slow at decoding long sequences. In this paper, we propose a novel model for fast sequence generation — the semi-autoregressive Transformer (SAT). The SAT keeps the autoregressive property in global but relieves in local and thus are able to produce multiple successive words in parallel at each time step. Experiments conducted on English-German and Chinese-English translation tasks show that the SAT achieves a good balance between translation quality and decoding speed. On WMT'14 English-German translation, the SAT achieves 5.58× speedup while maintaining 88% translation quality, significantly better than the previous non-autoregressive methods. When produces two words at each time step, the SAT is almost lossless (only 1% degeneration in BLEU score).

1 Introduction

Neural networks have been successfully applied to a variety of tasks, including machine translation. The encoder-decoder architecture is the central idea of neural machine translation (NMT). The encoder first encodes a source-side sentence $\mathbf{x} = x_1 \dots x_m$ into hidden states and then the decoder generates the target-side sentence $\mathbf{y} = y_1 \dots y_n$ from the hidden states according to an autoregressive model

$$p(y_t | y_1 \dots y_{t-1}, \mathbf{x})$$

Recurrent neural networks (RNNs) are inherently good at processing sequential data. Sutskever et al. (2014); Cho et al. (2014) successfully applied RNNs to machine translation. Bahdanau et al. (2014) introduced attention mechanism into the encoder-decoder architecture and greatly improved NMT. GNMT (Wu et al., 2016) further improved NMT by a bunch of tricks including residual connection and reinforcement learning.

The sequential property of RNNs leads to its wide application in language processing. However, the property also hinders its parallelizability thus RNNs are slow to execute on modern hardware optimized for parallel execution. As a result, a number of more parallelizable sequence models were proposed such as ConvS2S (Gehring et al., 2017) and the Transformer (Vaswani et al., 2017). These models avoid the dependencies between dif-

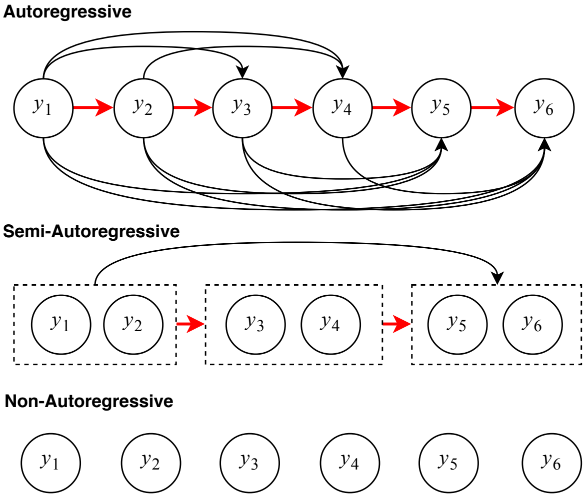

Figure 1: The different levels of autoregressive properties. Lines with arrow indicate dependencies. We mark the longest dependency path with bold red lines. The length of the longest dependency path decreases as we relieve the autoregressive property. An extreme case is *non-autoregressive*, where there is no dependency at all.

---

[*] Part of this work was done when the author was at Institute of Automation, Chinese Academy of Sciences.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 479–488
Brussels, Belgium, October 31 - November 4, 2018.

ferent positions in each layer thus can be trained much faster than RNN based models. When inference, however, these models are still slow because of the autoregressive property.

A recent work (Gu et al., 2017) proposed a non-autoregressive NMT model that generates all target-side words in parallel. While the parallelizability is greatly improved, the translation quality encounter much decrease. In this paper, we propose the semi-autoregressive Transformer (SAT) for faster sequence generation. Unlike Gu et al. (2017), the SAT is semi-autoregressive, which means it keeps the autoregressive property in global but relieves in local. As the result, the SAT can produce multiple successive words in parallel at each time step. Figure 1 gives an illustration of the different levels of autoregressive properties.

Experiments conducted on English-German and Chinese-English translation show that compared with non-autoregressive methods, the SAT achieves a better balance between translation quality and decoding speed. On WMT'14 English-German translation, the proposed SAT is 5.58× faster than the Transformer while maintaining 88% of translation quality. Besides, when producing two words at each time step, the SAT is almost lossless.

It is worth noting that although we apply the SAT to machine translation, it is not designed specifically for translation as Gu et al. (2017); Lee et al. (2018). The SAT can also be applied to any other sequence generation task, such as summary generation and image caption generation.

2 Related Work

Almost all state-of-the-art NMT models are autoregressive (Sutskever et al., 2014; Bahdanau et al., 2014; Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017), meaning that the model generates words one by one and is not friendly to modern hardware optimized for parallel execution. A recent work (Gu et al., 2017) attempts to accelerate generation by introducing a non-autoregressive model. Based on the Transformer (Vaswani et al., 2017), they made lots of modifications. The most significant modification is that they avoid feeding the previously generated target words to the decoder, but instead feeding the source words, to predict the next target word. They also introduced a set of latent variables to model the *fertilities* of source words to tackle the multimodality problem in translation. Lee et al. (2018) proposed another non-autoregressive sequence model based on iterative refinement. The model can be viewed as both a latent variable model and a conditional denoising autoencoder. They also proposed a learning algorithm that is hybrid of lower-bound maximization and reconstruction error minimization.

The most relevant to our proposed semi-autoregressive model is (Kaiser et al., 2018). They first autoencode the target sequence into a shorter sequence of discrete latent variables, which at inference time is generated autoregressively, and finally decode the output sequence from this shorter latent sequence in parallel. What we have in common with their idea is that we have not entirely abandoned autoregressive, but rather shortened the autoregressive path.

A related study on realistic speech synthesis is the parallel WaveNet (Oord et al., 2017). The paper introduced *probability density distillation*, a new method for training a parallel feed-forward network from a trained WaveNet (Van Den Oord et al., 2016) with no significant difference in quality.

There are also some work share a somehow simillar idea with our work: character-level NMT (Chung et al., 2016; Lee et al., 2016) and chunk-based NMT (Zhou et al., 2017; Ishiwatari et al., 2017). Unlike the SAT, these models are not able to produce multiple tokens (characters or words) each time step. Oda et al. (2017) proposed a bit-level decoder, where a word is represented by a binary code and each bit of the code can be predicted in parallel.

3 The Transformer

Since our proposed model is built upon the Transformer (Vaswani et al., 2017), we will briefly introduce the Transformer. The Transformer uses an encoder-decoder architecture. We describe the encoder and decoder below.

3.1 The Encoder

From the source tokens, learned embeddings of dimension d_{model} are generated which are then modified by an additive positional encoding. The positional encoding is necessary since the network does not leverage the order of the sequence by recurrence or convolution. The authors use additive encoding which is defined as:

$$PE(pos, 2i) = sin(pos/10000^{2i/d_{model}})$$
$$PE(pos, 2i+1) = cos(pos/10000^{2i/d_{model}})$$

where pos is the position of a word in the sentence and i is the dimension. The authors chose this function because they hypothesized it would allow the model to learn to attend by relative positions easily. The encoded word embeddings are then used as input to the encoder which consists of N blocks each containing two layers: (1) a multi-head attention layer, and (2) a position-wise feed-forward layer.

Multi-head attention builds upon scaled dot-product attention, which operates on a query Q, key K and value V:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where d_k is the dimension of the key. The authors scale the dot product by $1/\sqrt{d_k}$ to avoid the inputs to softmax function growing too large in magnitude. Multi-head attention computes h different queries, keys and values with h linear projections, computes scaled dot-product attention for each query, key and value, concatenates the results, and projects the concatenation with another linear projection:

$$H_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$
$$MultiHead(Q, K, V) = Concat(H_1, \dots H_h)$$

in which $W_i^Q, W_i^K \in \mathbb{R}^{d_{model} \times d_k}$ and $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$. The attention mechanism in the encoder performs attention over itself ($Q = K = V$), so it is also called self-attention.

The second component in each encoder block is a position-wise feed-forward layer defined as:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

where $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$, $b_1 \in \mathbb{R}^{d_{ff}}$, $b_2 \in \mathbb{R}^{d_{model}}$.

For more stable and faster convergence, residual connection (He et al., 2016) is applied to each layer, followed by layer normalization (Ba et al., 2016). For regularization, dropout (Srivastava et al., 2014) are applied before residual connections.

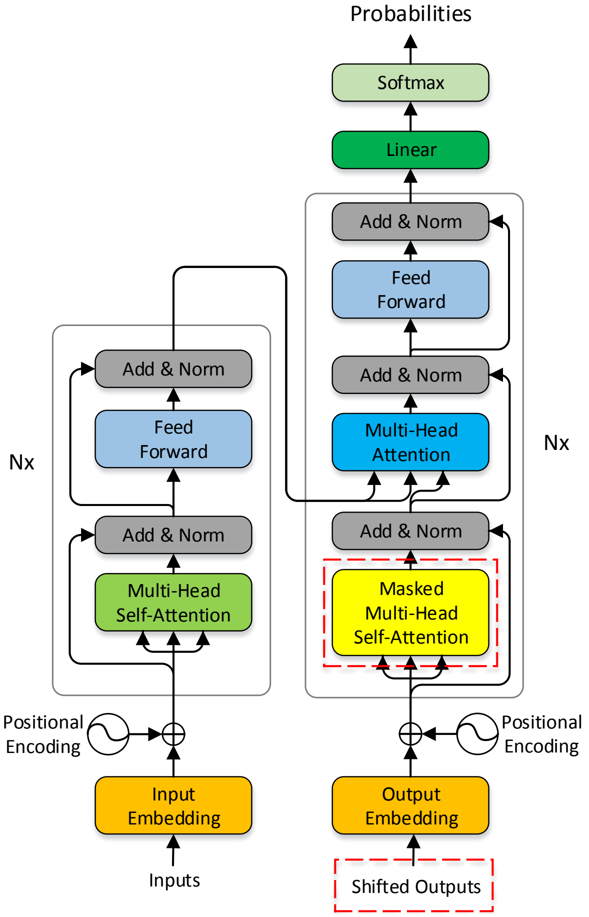

Figure 2: The architecture of the Transformer, also of the SAT, where the red dashed boxes point out the different parts of these two models.

3.2 The Decoder

The decoder is similar with the encoder and is also composed by N blocks. In addition to the two layers in each encoder block, the decoder inserts a third layer, which performs multi-head attention over the output of the encoder.

It is worth noting that, different from the encoder, the self-attention layer in the decoder must be masked with a causal mask, which is a lower triangular matrix, to ensure that the prediction for position i can depend only on the known outputs at positions less than i during training.

4 The Semi-Autoregressive Transformer

We propose a novel NMT model—the Semi-Autoregressive Transformer (SAT)—that can produce multiple successive words in parallel. As shown in Figure 2, the architecture of the SAT is almost the same as the Transformer, except some modifications in the decoder.

4.1 Group-Level Chain Rule

Standard NMT models usually factorize the joint probability of a word sequence $y_1 \dots y_n$ according to the word-level chain rule

$$p(y_1 \dots y_n|\mathbf{x}) = \prod_{t=1}^{n} p(y_t|y_1 \dots y_{t-1}, \mathbf{x})$$

resulting in decoding each word depending on all previous decoding results, thus hindering the parallelizability. In the SAT, we extend the standard word-level chain rule to the group-level chain rule.

We first divide the word sequence $y_1 \dots y_n$ into consecutive groups

$$G_1, G_2, \dots, G_{[(n-1)/K]+1} =$$
$$y_1 \cdots y_K, y_{K+1} \cdots y_{2K}, \dots, y_{[(n-1)/K]\times K+1} \cdots y_n$$

where $[\cdot]$ denotes floor operation, K is the group size, and also the indicator of parallelizability. The larger the K, the higher the parallelizability. Except for the last group, all groups must contain K words. Then comes the group-level chain rule

$$p(y_1 \dots y_n|\mathbf{x}) = \prod_{t=1}^{[(n-1)/K]+1} p(G_t|G_1 \dots G_{t-1}, \mathbf{x})$$

This group-level chain rule avoids the dependencies between consecutive words if they are in the same group. With group-level chain rule, the model no longer produce words one by one as the Transformer, but rather group by group. In next subsections, we will show how to implement the model in detail.

4.2 Long-Distance Prediction

In autoregressive models, to predict y_t, the model should be fed with the previous word y_{t-1}. We refer it as *short-distance prediction*. In the SAT, however, we feed y_{t-K} to predict y_t, to which we refer as *long-distance prediction*. At the beginning of decoding, we feed the model with K special symbols <s> to predict $y_1 \dots y_K$ in parallel. Then $y_1 \dots y_K$ are fed to the model to predict $y_{K+1} \dots y_{2K}$ in parallel. This process will continue until a terminator </s> is generated. Figure 3 gives illustrations for both short and long-distance prediction.

4.3 Relaxed Causal Mask

In the Transformer decoder, the causal mask is a lower triangular matrix, which strictly prevents earlier decoding steps from peeping information from later steps. We denote it as *strict causal mask*. However, in the SAT decoder, strict causal mask is not a good choice. As described in the previous subsection, in long-distance prediction, the model predicts y_{K+1} by feeding with y_1. With strict causal mask, the model can only access to y_1 when predict y_{K+1}, which is not reasonable since $y_1 \dots y_K$ are already produced. It is better to allow the model to access to $y_1 \dots y_K$ rather than only y_1 when predict y_{K+1}.

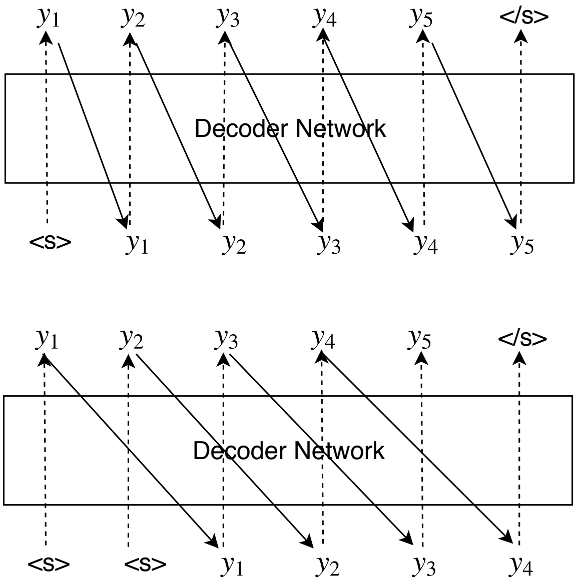

Figure 3: Short-distance prediction (top) and long-distance prediction (bottom).

Therefore, we use a coarse-grained lower triangular matrix as the causal mask that allows peeping later information in the same group. We refer to it as *relaxed causal mask*. Given the target length n and the group size K, relaxed causal mask $M \in \mathbb{R}^{n\times n}$ and its elements are defined below:

$$M[i][j] = \begin{cases} 1 & \text{if } j < ([(i-1)/K]+1) \times K \\ 0 & \text{other} \end{cases}$$

For a more intuitive understanding, Figure 4 gives a comparison between strict and relaxed causal mask.

4.4 The SAT

Using group-level chain rule instead of word-level chain rule, long-distance prediction instead of short-distance prediction, and relaxed causal

$$\begin{bmatrix} 1 & \mathbf{0} & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & \mathbf{0} & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & \mathbf{0} \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & \mathbf{1} & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & \mathbf{1} \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure 4: Strict causal mask (left) and relaxed causal mask (right) when the target length $n = 6$ and the group size $K = 2$. We mark their differences in bold.

Model	Complexity	Acceleration
Transformer	$N(a+b)$	1
SAT (beam search)	$\frac{N}{K}a + Nb$	$K(\frac{a+b}{a+Kb})$
SAT (greedy search)	$\frac{N}{K}(a+b)$	K

Table 1: Theoretical complexity and acceleration of the SAT. a denotes the time consumed on the decoder network (calculating a distribution over the target vocabulary) each time step and b denotes the time consumed on search (searching for top scores, expanding nodes and pruning). In practice, a is usually much larger than b since the network is deep.

mask instead of strict causal mask, we successfully extended the Transformer to the SAT. The Transformer can be viewed as a special case of the SAT, when the group size $K = 1$. The non-autoregressive Transformer (NAT) described in Gu et al. (2017) can also be viewed as a special case of the SAT, when the group size K is not less than maximum target length.

Table 1 gives the theoretical complexity and acceleration of the model. We list two search strategies separately: beam search and greedy search. Beam search is the most prevailing search strategy. However, it requires the decoder states to be updated once every word is generated, thus hinders the decoding parallelizability. When decode with greedy search, there is no such concern, therefore the parallelizability of the SAT can be maximized.

5 Experiments

We evaluate the proposed SAT on English-German and Chinese-English translation tasks.

5.1 Experimental Settings

Datasets For English-German translation, we choose the corpora provided by WMT 2014 (Bojar et al., 2014). We use the newstest2013 dataset for development, and the newstest2014 dataset for test. For Chinese-English translation, the corpora we use is extracted from LDC[1]. We chose the NIST02 dataset for development, and the NIST03, NIST04 and NIST05 datasets for test. For English and German, we tokenized and segmented them into subword symbols using byte-pair encoding (BPE) (Sennrich et al., 2015) to restrict the vocabulary size. As for Chinese, we segmented sentences into characters. For English-German translation, we use a shared source and target vocabulary. Table 2 summaries the two corpora.

	Sentence Number	Vocab Size	
		Source	Target
EN-DE	4.5M	36K	36K
ZH-EN	1.8M	9K	34K

Table 2: Summary of the two corpora.

Baseline We use the base Transformer model described in Vaswani et al. (2017) as the baseline, where $d_{model} = 512$ and $N = 6$. In addition, for comparison, we also prepared a lighter Transformer model, in which two encoder/decoder blocks are used ($N = 2$), and other hyper-parameters remain the same.

Hyperparameters Unless otherwise specified, all hyperparameters are inherited from the base Transformer model. We try three different settings of the group size K: $K = 2$, $K = 4$, and $K = 6$. For English-German translation, we share the same weight matrix between the source and target embedding layers and the pre-softmax linear layer. For Chinese-English translation, we only share weights of the target embedding layer and the pre-softmax linear layer.

Search Strategies We use two search strategies: beam search and greedy search. As mentioned in Section 4.4, these two strategies lead to different parallelizability. When beam size is set to 1, greedy search is used, otherwise, beam search is used.

Knowledge Distillation Knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016) describes a class of methods for training a smaller *student* network to perform better by learning from a larger *teacher* network. For NMT, Kim and Rush (2016) proposed a sequence-level knowledge distillation method. In this work, we apply this method to train the SAT using a pre-trained

---

[1] The corpora include LDC2002E18, LDC2003E14, LDC2004T08 and LDC2005T0.

Model	Beam Size	BLEU	Degeneration	Lattency	Speedup
Transformer	4	27.11	0%	346ms	1.00×
	1	26.01	4%	283ms	1.22×
Transformer, N=2	4	24.30	10%	163ms	2.12×
	1	23.37	14%	113ms	3.06×
NAT (Gu et al., 2017)	-	17.69	25%	39ms	15.6×
NAT (rescroing 10)	-	18.66	20%	79ms	7.68×
NAT (rescroing 100)	-	19.17	18%	257ms	2.36×
LT (Kaiser et al., 2018)	-	19.80	27%	105ms	-
LT (rescoring 10)	-	21.00	23%	-	-
LT (rescoring 100)	-	22.50	18%	-	-
IRNAT (Lee et al., 2018)	-	18.91	22%	-	1.98×
This Work					
SAT, K=2	4	26.90	1%	229ms	1.51×
	1	26.09	4%	167ms	2.07×
SAT, K=4	4	25.71	5%	149ms	2.32×
	1	24.67	9%	91ms	3.80×
SAT, K=6	4	24.83	8%	116ms	2.98×
	1	23.93	12%	62ms	5.58×

Table 3: Results on English-German translation. Latency is calculated on a single NVIDIA TITAN Xp without batching. For comparison, we also list results reported by Gu et al. (2017); Kaiser et al. (2018); Lee et al. (2018). Note that Gu et al. (2017); Lee et al. (2018) used PyTorch as their platform, but we and Kaiser et al. (2018) used TensorFlow. Even on the same platform, implementation and hardware may not exactly be the same. Therefore, it is not fair to directly compare BLEU and latency. A fairer way is to compare performance degradation and speedup, which are calculated based on their own baseline.

autoregressive Transformer network. This method consists of three steps: (1) train an autoregressive Transformer network (the *teacher*), (2) run beam search over the training set with this model and (3) train the SAT (the *student*) on this new created corpus.

Initialization Since the SAT and the Transformer have only slight differences in their architecture (see Figure 2), in order to accelerate convergence, we use a pre-trained Transformer model to initialize some parameters in the SAT. These parameters include all parameters in the encoder, source and target word embeddings, and pre-softmax weights. Other parameters are initialized randomly. In addition to accelerating convergence, we find this method also slightly improves the translation quality.

Training Same as Vaswani et al. (2017), we train the SAT by minimize cross-entropy with label smoothing. The optimizer we use is Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\varepsilon = 10^{-9}$. We change the learning rate during training using the learning rate funtion described in Vaswani et al. (2017). All models are trained for 10K steps on 8 NVIDIA TITAN Xp with each minibatch consisting of about 30k tokens. For evaluation, we average last five checkpoints saved with an interval of 1000 training steps.

Evaluation Metrics We evaluate the translation quality of the model using BLEU score (Papineni et al., 2002).

Implementation We implement the proposed SAT with *TensorFlow* (Abadi et al., 2016). The code and resources needed for reproducing the results are released at `https://github.com/chqiwang/sa-nmt`.

5.2 Results on English-German

Table 3 summaries results of English-German translation. According to the results, the translation quality of the SAT gradually decreases as K increases, which is consistent with intuition. When K = 2, the SAT decodes 1.51× faster than the Transformer and is almost lossless in translation quality (only drops 0.21 BLEU score). With K = 6, the SAT can achieve 2.98× speedup while the performance degeneration is only 8%.

When using greedy search, the acceleration becomes much more significant. When K = 6, the decoding speed of the SAT can reach about 5.58× of the Transformer while maintaining 88%

Model	b=1	b=16	b=32	b=64
Transformer	346ms	58ms	53ms	56ms
SAT, K=2	229ms	38ms	32ms	32ms
SAT, K=4	149ms	24ms	21ms	20ms
SAT, K=6	116ms	20ms	17ms	16ms

Table 4: Time needed to decode one sentence under various batch size settings. A single NVIDIA TIAN Xp is used in this test.

Model	K=1	K=2	K=4	K=6
Latency	1384ms	607ms	502ms	372ms

Table 5: Time needed to decode one sentence on CPU device. Sentences are decoded one by one without batching. K=1 denotes the Transformer.

of translation quality. Comparing with Gu et al. (2017); Kaiser et al. (2018); Lee et al. (2018), the SAT achieves a better balance between translation quality and decoding speed. Compared to the lighter Transformer (N = 2), with K = 4, the SAT achieves a higher speedup with significantly better translation quality.

In a real production environment, it is often not to decode sentences one by one, but batch by batch. To investigate whether the SAT can accelerate decoding when decoding in batches, we test the decoding latency under different batch size settings. As shown in Table 4, the SAT significantly accelerates decoding even with a large batch size.

It is also good to know if the SAT can still accelerate decoding on CPU device that does not support parallel execution as well as GPU. Results in Table 5 show that even on CPU device, the SAT can still accelerate decoding significantly.

5.3 Results on Chinese-English

Table 6 summaries results on Chinese-English translation. With K = 2, the SAT decodes 1.69× while maintaining 97% of the translation quality. In an extreme setting where K = 6 and beam size = 1, the SAT can achieve 6.41× speedup while maintaining 83% of the translation quality.

5.4 Analysis

Effects of Knowledge Distillation As shown in Figure 5, sequence-level knowledge distillation is very effective for training the SAT. For larger K, the effect is more significant. This phenomenon is echoing with observations by Gu et al. (2017); Oord et al. (2017); Lee et al. (2018). In addition, we tried word-level knowledge distillation (Kim and Rush, 2016) but only a slight improvement was observed.

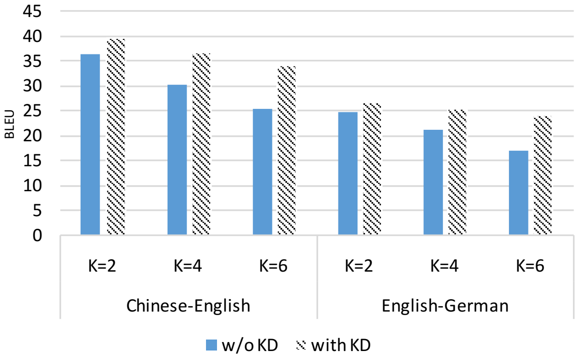

Figure 5: Performance of the SAT with and without sequence-level knowledge distillation.

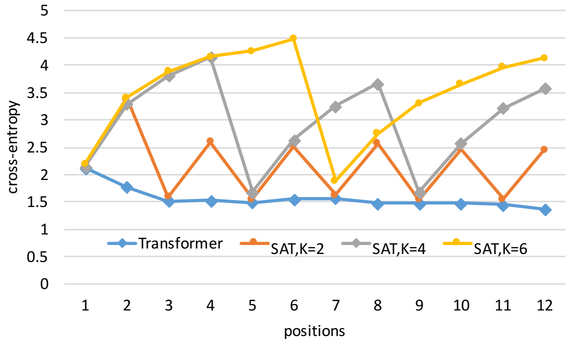

Figure 6: Position-wise cross-entropy for various models on English-German translation.

Position-Wise Cross-Entropy In Figure 6, we plot position-wise cross-entropy for various models. To compare with the baseline model, the results in the figure are from models trained on the original corpora, i.e., without knowledge distillation. As shown in the figure, position-wise cross-entropy has an apparent periodicity with a period of K. For positions in the same group, the position-wise cross-entropy increase monotonously, which indicates that the long-distance dependencies are always more difficult to model than short ones. It suggests the key to further improve the SAT is to improve the ability of modeling long-distance dependencies.

Case Study Table 7 lists three sample Chinese-English translations from the development set. As shown in the table, even when producing K = 6 words at each time step, the model can still gen-

Model	Beam Size	BLEU				Degeneration	Latency	Speedup
		NIST03	NIST04	NIST05	Averaged			
Transformer	4	40.74	40.54	40.48	40.59	0%	410ms	1.00×
	1	39.56	39.72	39.61	39.63	2%	302ms	1.36×
Transformer, N=2	4	37.30	38.55	36.87	37.57	7%	169ms	2.43×
	1	36.26	37.19	35.50	36.32	11%	117ms	3.50×
This Work								
SAT, K=2	4	39.13	40.04	39.55	39.57	3%	243ms	1.69×
	1	37.94	38.73	38.43	38.37	5%	176ms	2.33×
SAT, K=4	4	37.08	38.06	37.12	37.42	8%	152ms	2.70×
	1	35.77	36.43	35.04	35.75	12%	94ms	4.36×
SAT, K=6	4	34.61	36.29	35.06	35.32	13%	129ms	3.18×
	1	33.44	34.54	33.28	33.75	17%	64ms	6.41×

Table 6: Results on Chinese-English translation. Latency is calculated on NIST02.

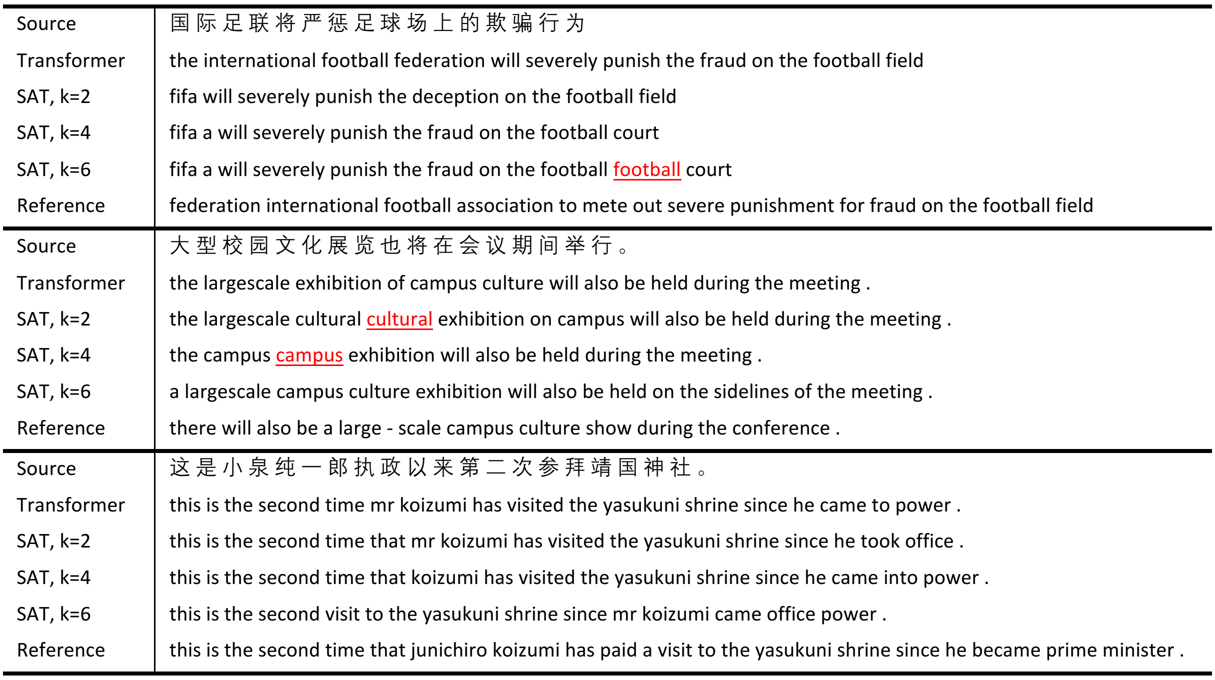

Source	国际足联将严惩足球场上的欺骗行为
Transformer	the international football federation will severely punish the fraud on the football field
SAT, k=2	fifa will severely punish the deception on the football field
SAT, k=4	fifa a will severely punish the fraud on the football court
SAT, k=6	fifa a will severely punish the fraud on the football football court
Reference	federation international football association to mete out severe punishment for fraud on the football field
Source	大型校园文化展览也将在会议期间举行。
Transformer	the largescale exhibition of campus culture will also be held during the meeting .
SAT, k=2	the largescale cultural cultural exhibition on campus will also be held during the meeting .
SAT, k=4	the campus campus exhibition will also be held during the meeting .
SAT, k=6	a largescale campus culture exhibition will also be held on the sidelines of the meeting .
Reference	there will also be a large - scale campus culture show during the conference .
Source	这是小泉纯一郎执政以来第二次参拜靖国神社。
Transformer	this is the second time mr koizumi has visited the yasukuni shrine since he came to power .
SAT, k=2	this is the second time that mr koizumi has visited the yasukuni shrine since he took office .
SAT, k=4	this is the second time that koizumi has visited the yasukuni shrine since he came into power .
SAT, k=6	this is the second visit to the yasukuni shrine since mr koizumi came office power .
Reference	this is the second time that junichiro koizumi has paid a visit to the yasukuni shrine since he became prime minister .

Table 7: Three sample Chinese-English translations by the SAT and the Transformer. We mark repeated words or phrases by red font and underline.

erate fluent sentences. As reported by Gu et al. (2017), instances of repeated words or phrases are most prevalent in their non-autoregressive model. In the SAT, this is also the case. This suggests that we may be able to improve the translation quality of the SAT by reducing the similarity of the output distribution of adjacent positions.

6 Conclusion

In this work, we have introduced a novel model for faster sequence generation based on the Transformer (Vaswani et al., 2017), which we refer to as the semi-autoregressive Transformer (SAT). Combining the original Transformer with group-level chain rule, long-distance prediction and relaxed causal mask, the SAT can produce multiple consecutive words at each time step, thus speedup decoding significantly. We conducted experiments on English-German and Chinese-English translation. Compared with previously proposed non-autoregressive models (Gu et al., 2017; Lee et al., 2018; Kaiser et al., 2018), the SAT achieves a better balance between translation quality and decoding speed. On WMT'14 English-German translation, the SAT achieves 5.58× speedup while maintaining 88% translation quality, significantly better than previous methods. When producing two words at each time step, the SAT is almost lossless (only 1% degeneration in BLEU score).

In the future, we plan to investigate better methods for training the SAT to further shrink the performance gap between the SAT and the Transformer. Specifically, we believe that the following two directions are worth study. First, use object function beyond maximum likelihood to improve the modeling of long-distance dependencies. Second, explore new method for knowledge distillation. We also plan to extend the SAT to allow the use of different group sizes K at different positions, instead of using a fixed value.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. We also thank Wenfu Wang, Hao Wang for helpful discussion and Linhao Dong, Jinghao Niu for their help in paper writting.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Shonosuke Ishiwatari, Jingtao Yao, Shujie Liu, Mu Li, Ming Zhou, Naoki Yoshinaga, Masaru Kitsuregawa, and Weijia Jia. 2017. Chunk-based decoder for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1901–1912.

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Pamar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. *international conference on learning representations*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*.

Yusuke Oda, Philip Arthur, Graham Neubig, Koichiro Yoshino, and Satoshi Nakamura. 2017. Neural machine translation via binary code prediction. *arXiv preprint arXiv:1704.06918*.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. 2017. Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of* *the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, and Klaus Macherey. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Hao Zhou, Zhaopeng Tu, Shujian Huang, Xiaohua Liu, Hang Li, and Jiajun Chen. 2017. Chunk-based biscale decoder for neural machine translation. *arXiv preprint arXiv:1705.01452*.

Understanding Back-Translation at Scale

Sergey Edunov$^{\triangle}$ **Myle Ott**$^{\triangle}$ **Michael Auli**$^{\triangle}$ **David Grangier**$^{\triangledown *}$
$^{\triangle}$Facebook AI Research, Menlo Park, CA & New York, NY.
$^{\triangledown}$Google Brain, Mountain View, CA.

Abstract

An effective method to improve neural machine translation with monolingual data is to augment the parallel training corpus with back-translations of target language sentences. This work broadens the understanding of back-translation and investigates a number of methods to generate synthetic source sentences. We find that in all but resource poor settings back-translations obtained via sampling or noised beam outputs are most effective. Our analysis shows that sampling or noisy synthetic data gives a much stronger training signal than data generated by beam or greedy search. We also compare how synthetic data compares to genuine bitext and study various domain effects. Finally, we scale to hundreds of millions of monolingual sentences and achieve a new state of the art of 35 BLEU on the WMT'14 English-German test set.

1 Introduction

Machine translation relies on the statistics of large parallel corpora, i.e. datasets of paired sentences in both the source and target language. However, bitext is limited and there is a much larger amount of monolingual data available. Monolingual data has been traditionally used to train language models which improved the fluency of statistical machine translation (Koehn, 2010).

In the context of neural machine translation (NMT; Bahdanau et al. 2015; Gehring et al. 2017; Vaswani et al. 2017), there has been extensive work to improve models with monolingual data, including language model fusion (Gulcehre et al., 2015, 2017), back-translation (Sennrich et al., 2016a) and dual learning (Cheng et al., 2016; He et al., 2016a). These methods have different advantages and can be combined to reach high accuracy (Hassan et al., 2018).

We focus on back-translation (BT) which operates in a semi-supervised setup where both bilingual and monolingual data in the target language are available. Back-translation first trains an intermediate system on the parallel data which is used to translate the target monolingual data into the source language. The result is a parallel corpus where the source side is *synthetic* machine translation output while the target is genuine text written by humans. The synthetic parallel corpus is then simply added to the real bitext in order to train a final system that will translate from the source to the target language. Although simple, this method has been shown to be helpful for phrase-based translation (Bojar and Tamchyna, 2011), NMT (Sennrich et al., 2016a; Poncelas et al., 2018) as well as unsupervised MT (Lample et al., 2018a).

In this paper, we investigate back-translation for neural machine translation at a large scale by adding hundreds of millions of back-translated sentences to the bitext. Our experiments are based on strong baseline models trained on the public bitext of the WMT competition. We extend previous analysis (Sennrich et al., 2016a; Poncelas et al., 2018) of back-translation in several ways. We provide a comprehensive analysis of different methods to generate synthetic source sentences and we show that this choice matters: sampling from the model distribution or noising beam outputs outperforms pure beam search, which is typically used, by 1.7 BLEU on average across several test sets. Our analysis shows that synthetic data based on sampling and noised beam search provides a stronger training signal than synthetic data based on argmax inference. We also study how adding synthetic data compares to adding real bitext in a controlled setup with the surprising finding that synthetic data can sometimes match the accuracy of real bitext. Our best setup achieves 35 BLEU on the WMT'14 English-German test set by rely-

---

*Work done while at Facebook AI Research.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 489–500
Brussels, Belgium, October 31 - November 4, 2018.

ing only on public WMT bitext as well as 226M monolingual sentences. This outperforms the system of DeepL by 1.7 BLEU who train on large amounts of high quality non-benchmark data. On WMT'14 English-French we achieve 45.6 BLEU.

2 Related work

This section describes prior work in machine translation with neural networks as well as semi-supervised machine translation.

2.1 Neural machine translation

We build upon recent work on neural machine translation which is typically a neural network with an encoder/decoder architecture. The encoder infers a continuous space representation of the source sentence, while the decoder is a neural language model conditioned on the encoder output. The parameters of both models are learned jointly to maximize the likelihood of the target sentences given the corresponding source sentences from a parallel corpus (Sutskever et al., 2014; Cho et al., 2014). At inference, a target sentence is generated by left-to-right decoding.

Different neural architectures have been proposed with the goal of improving efficiency and/or effectiveness. This includes recurrent networks (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015), convolutional networks (Kalchbrenner et al., 2016; Gehring et al., 2017; Kaiser et al., 2017) and transformer networks (Vaswani et al., 2017). Recent work relies on attention mechanisms where the encoder produces a sequence of vectors and, for each target token, the decoder attends to the most relevant part of the source through a context-dependent weighted-sum of the encoder vectors (Bahdanau et al., 2015; Luong et al., 2015). Attention has been refined with multi-hop attention (Gehring et al., 2017), self-attention (Vaswani et al., 2017; Paulus et al., 2018) and multi-head attention (Vaswani et al., 2017). We use a transformer architecture (Vaswani et al., 2017).

2.2 Semi-supervised NMT

Monolingual target data has been used to improve the fluency of machine translations since the early IBM models (Brown et al., 1990). In phrase-based systems, language models (LM) in the target language increase the score of fluent outputs during decoding (Koehn et al., 2003; Brants et al., 2007). A similar strategy can be applied to NMT (He et al., 2016b). Besides improving accuracy during decoding, neural LM and NMT can benefit from deeper integration, e.g. by combining the hidden states of both models (Gulcehre et al., 2017). Neural architecture also allows multi-task learning and parameter sharing between MT and target-side LM (Domhan and Hieber, 2017).

Back-translation (BT) is an alternative to leverage monolingual data. BT is simple and easy to apply as it does not require modification to the MT training algorithms. It requires training a target-to-source system in order to generate additional synthetic parallel data from the monolingual target data. This data complements human bitext to train the desired source-to-target system. BT has been applied earlier to phrase-base systems (Bojar and Tamchyna, 2011). For these systems, BT has also been successful in leveraging monolingual data for domain adaptation (Bertoldi and Federico, 2009; Lambert et al., 2011). Recently, BT has been shown beneficial for NMT (Sennrich et al., 2016a; Poncelas et al., 2018). It has been found to be particularly useful when parallel data is scarce (Karakanta et al., 2017).

Currey et al. (2017) show that low resource language pairs can also be improved with synthetic data where the source is simply a copy of the monolingual target data. Concurrently to our work, Imamura et al. (2018) show that sampling synthetic sources is more effective than beam search. Specifically, they sample multiple sources for each target whereas we draw only a single sample, opting to train on a larger number of target sentences instead. Hoang et al. (2018) and Cotterell and Kreutzer (2018) suggest an iterative procedure which continuously improves the quality of the back-translation and final systems. Niu et al. (2018) experiment with a multilingual model that does both the forward and backward translation which is continuously trained with new synthetic data.

There has also been work using source-side monolingual data (Zhang and Zong, 2016). Furthermore, Cheng et al. (2016); He et al. (2016a); Xia et al. (2017) show how monolingual text from both languages can be leveraged by extending back-translation to dual learning: when training both source-to-target and target-to-source models jointly, one can use back-translation in both directions and perform multiple rounds of BT. A similar idea is applied in unsupervised NMT (Lample et al., 2018a,b). Besides monolingual data, various approaches have been introduced to benefit from parallel data in other language pairs (Johnson et al., 2017; Firat et al., 2016a,b; Ha et al., 2016; Gu et al., 2018).

Data augmentation is an established technique in computer vision where a labeled dataset is supplemented with cropped or rotated input images. Recently, generative adversarial networks (GANs) have been successfully used to the same end (Antoniou et al., 2017; Perez and Wang, 2017) as well as models that learn distributions over image transformations (Hauberg et al., 2016).

3 Generating synthetic sources

Back-translation typically uses beam search (Sennrich et al., 2016a) or just greedy search (Lample et al., 2018a,b) to generate synthetic source sentences. Both are approximate algorithms to identify the maximum a-posteriori (MAP) output, i.e. the sentence with the largest estimated probability given an input. Beam is generally successful in finding high probability outputs (Ott et al., 2018a).

However, MAP prediction can lead to less rich translations (Ott et al., 2018a) since it always favors the most likely alternative in case of ambiguity. This is particularly problematic in tasks where there is a high level of uncertainty such as dialog (Serban et al., 2016) and story generation (Fan et al., 2018). We argue that this is also problematic for a data augmentation scheme such as back-translation. Beam and greedy focus on the head of the model distribution which results in very regular synthetic source sentences that do not properly cover the true data distribution.

As alternative, we consider sampling from the model distribution as well as adding noise to beam search outputs. First, we explore unrestricted sampling which generates outputs that are very diverse but sometimes highly unlikely. Second, we investigate sampling restricted to the most likely words (Graves, 2013; Ott et al., 2018a; Fan et al., 2018). At each time step, we select the k most likely tokens from the output distribution, renormalize and then sample from this restricted set. This is a middle ground between MAP and unrestricted sampling.

As a third alternative, we apply noising Lample et al. (2018a) to beam search outputs. Adding noise to input sentences has been very beneficial for the autoencoder setups of (Lample et al., 2018a; Hill et al., 2016) which is inspired by denoising autoencoders (Vincent et al., 2008). In particular, we transform source sentences with three types of noise: deleting words with probability 0.1, replacing words by a filler token with probability 0.1, and swapping words which is implemented as a random permutation over the tokens, drawn from the uniform distribution but restricted to swapping words no further than three positions apart.

4 Experimental setup

4.1 Datasets

The majority of our experiments are based on data from the WMT'18 English-German news translation task. We train on all available bitext excluding the ParaCrawl corpus and remove sentences longer than 250 words as well as sentence-pairs with a source/target length ratio exceeding 1.5. This results in 5.18M sentence pairs. For the back-translation experiments we use the German monolingual newscrawl data distributed with WMT'18 comprising 226M sentences after removing duplicates. We tokenize all data with the Moses tokenizer (Koehn et al., 2007) and learn a joint source and target Byte-Pair-Encoding (BPE; Sennrich et al., 2016) with 35K types. We develop on newstest2012 and report final results on newstest2013-2017; additionally we consider a held-out set from the training data of 52K sentence-pairs.

We also experiment on the larger WMT'14 English-French task which we filter in the same way as WMT'18 English-German. This results in 35.7M sentence-pairs for training and we learn a joint BPE vocabulary of 44K types. As monolingual data we use newscrawl2010-2014, comprising 31M sentences after language identification (Lui and Baldwin, 2012). We use newstest2012 as development set and report final results on newstest2013-2015.

The majority of results in this paper are in terms of case-sensitive tokenized BLEU (Papineni et al., 2002) but we also report test accuracy with detokenized BLEU using sacreBLEU (Post, 2018).

4.2 Model and hyperparameters

We re-implemented the Transformer model in pytorch using the fairseq toolkit.[1] All experiments

---

[1] Code available at `https://github.com/pytorch/fairseq`

are based on the Big Transformer architecture with 6 blocks in the encoder and decoder. We use the same hyper-parameters for all experiments, i.e., word representations of size 1024, feed-forward layers with inner dimension 4096. Dropout is set to 0.3 for En-De and 0.1 for En-Fr, we use 16 attention heads, and we average the checkpoints of the last ten epochs. Models are optimized with Adam (Kingma and Ba, 2015) using $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e-8$ and we use the same learning rate schedule as Vaswani et al. (2017). All models use label smoothing with a uniform prior distribution over the vocabulary $\epsilon = 0.1$ (Szegedy et al., 2015; Pereyra et al., 2017). We run experiments on DGX-1 machines with 8 Nvidia V100 GPUs and machines are interconnected by Infiniband. Experiments are run on 16 machines and we perform 30K synchronous updates. We also use the NCCL2 library and the torch distributed package for inter-GPU communication. We train models with 16-bit floating point operations, following Ott et al. (2018b). For final evaluation, we generate translations with a beam of size 5 and with no length penalty.

5 Results

Our evaluation first compares the accuracy of back-translation generation methods (§5.1) and analyzes the results (§5.2). Next, we simulate a low-resource setup to experiment further with different generation methods (§5.3). We also compare synthetic bitext to genuine parallel data and examine domain effects arising in back-translation (§5.4). We also measure the effect of upsampling bitext during training (§5.5). Finally, we scale to a very large setup of up to 226M monolingual sentences and compare to previous research (§5.6).

5.1 Synthetic data generation methods

We first investigate different methods to generate synthetic source translations given a back-translation model, i.e., a model trained in the reverse language direction (Section 5.1). We consider two types of MAP prediction: greedy search (greedy) and beam search with beam size 5 (beam). Non-MAP methods include unrestricted sampling from the model distribution (sampling), restricting sampling to the k highest scoring outputs at every time step with $k = 10$ (top10) as well as adding noise to the beam outputs (beam+noise). Restricted sampling is a middle-ground between beam search and unrestricted sampling, it is less likely to pick very low scoring outputs but still preserves some randomness. Preliminary experiments with top5, top20, top50 gave similar results to top10.

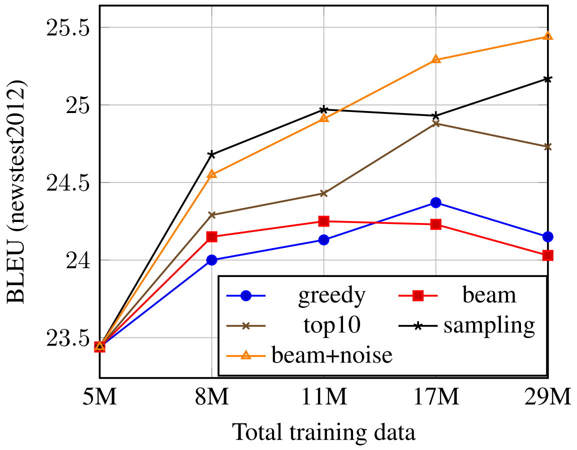

Figure 1: Accuracy of models trained on different amounts of back-translated data obtained with greedy search, beam search ($k = 5$), randomly sampling from the model distribution, restricting sampling over the ten most likely words (top10), and by adding noise to the beam outputs (beam+noise). Results based on newstest2012 of WMT English-German translation.

We also vary the amount of synthetic data and perform 30K updates during training for the bitext only, 50K updates when adding 3M synthetic sentences, 75K updates for 6M and 12M sentences and 100K updates for 24M sentences. For each setting, this corresponds to enough updates to reach convergence in terms of held-out loss. In our 128 GPU setup, training of the final models takes 3h 20min for the bitext only model, 7h 30min for 6M and 12M synthetic sentences, and 10h 15min for 24M sentences. During training we also sample the bitext more frequently than the synthetic data and we analyze the effect of this in more detail in §5.5.

Figure 1 shows that sampling and beam+noise outperform the MAP methods (pure beam search and greedy) by 0.8-1.1 BLEU. Sampling and beam+noise improve over bitext-only (5M) by between 1.7-2 BLEU in the largest data setting. Restricted sampling (top10) performs better than beam and greedy but is not as effective as unrestricted sampling (sampling) or beam+noise.

Table 1 shows results on a wider range of

	news2013	**news2014**	**news2015**	**news2016**	**news2017**	**Average**
bitext	27.84	30.88	31.82	34.98	29.46	31.00
+ beam	27.82	32.33	32.20	35.43	31.11	31.78
+ greedy	27.67	32.55	32.57	35.74	31.25	31.96
+ top10	28.25	33.94	34.00	36.45	32.08	32.94
+ sampling	28.81	34.46	34.87	37.08	32.35	33.51
+ beam+noise	29.28	33.53	33.79	37.89	32.66	33.43

Table 1: Tokenized BLEU on various test sets of WMT English-German when adding 24M synthetic sentence pairs obtained by various generation methods to a 5.2M sentence-pair bitext (cf. Figure 1).

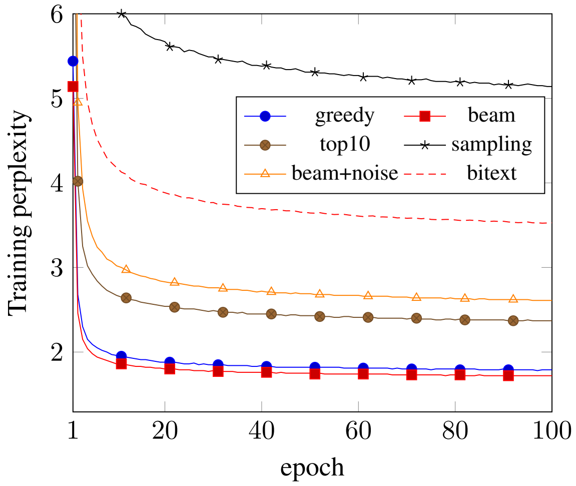

Figure 2: Training perplexity (PPL) per epoch for different synthetic data. We separately report PPL on the synthetic data and the bitext. Bitext PPL is averaged over all generation methods.

	Perplexity
human data	75.34
beam	72.42
sampling	500.17
top10	87.15
beam+noise	2823.73

Table 2: Perplexity of source data as assigned by a language model (5-gram Kneser–Ney). Data generated by beam search is most predictable.

test sets (newstest2013-2017). Sampling and beam+noise perform roughly equal and we adopt sampling for the remaining experiments.

5.2 Analysis of generation methods

The previous experiment showed that synthetic source sentences generated via sampling and beam with noise perform significantly better than those obtained by pure MAP methods. Why is this?

Beam search focuses on very likely outputs which reduces the diversity and richness of the generated source translations. Adding noise to beam outputs and sampling do not have this problem: Noisy source sentences make it harder to predict the target translations which may help learning, similar to denoising autoencoders (Vincent et al., 2008). Sampling is known to better approximate the data distribution which is richer than the argmax model outputs (Ott et al., 2018a). Therefore, sampling is also more likely to provide a richer training signal than argmax sequences.

To get a better sense of the training signal provided by each method, we compare the loss on the training data for each method. We report the cross entropy loss averaged over all tokens and separate the loss over the synthetic data and the real bitext data. Specifically, we choose the setup with 24M synthetic sentences. At the end of each epoch we measure the loss over 500K sentence pairs sub-sampled from the synthetic data as well as an equally sized subset of the bitext. For each generation method we choose the same sentences except for the bitext which is disjoint from the synthetic data. This means that losses over the synthetic data are measured over the *same* target tokens because the generation methods only differ in the source sentences. We found it helpful to upsample the frequency with which we observe the bitext compared to the synthetic data (§5.5) but we do not upsample for this experiment to keep conditions as similar as possible. We assume that when the training loss is low, then the model can easily fit the training data without extracting much learning signal compared to data which is harder to fit.

Figure 2 shows that synthetic data based on

source	Diese gegenstzlichen Auffassungen von Fairness liegen nicht nur der politischen Debatte zugrunde.
reference	These competing principles of fairness underlie not only the political debate.
beam	These conflicting interpretations of fairness are not solely based on the political debate.
sample	*Mr President,* these contradictory interpretations of fairness are not based solely on the political debate.
top10	Those conflicting interpretations of fairness are not solely at the heart of the political debate.
beam+noise	conflicting BLANK interpretations BLANK are of not BLANK based on the political debate.

Table 3: Example where sampling produces inadequate outputs. ”Mr President,” is not in the source. BLANK means that a word has been replaced by a filler token.

greedy or beam is much easier to fit compared to data from sampling, top10, beam+noise and the bitext. In fact, the perplexity on beam data falls below 2 after only 5 epochs. Except for sampling, we find that the perplexity on the training data is somewhat correlated to the end-model accuracy (cf. Figure 1) and that all methods except sampling have a lower loss than real bitext.

These results suggest that synthetic data obtained with argmax inference does not provide as rich a training signal as sampling or adding noise. We conjecture that the regularity of synthetic data obtained with argmax inference is not optimal. Sampling and noised argmax both expose the model to a wider range of source sentences which makes the model more robust to reordering and substitutions that happen naturally, even if the model of reordering and substitution through noising is not very realistic.

Next we analyze the richness of synthetic outputs and train a language model on real human text and score synthetic source sentences generated by beam search, sampling, top10 and beam+noise. We hypothesize that data that is very regular should be more predictable by the language model and therefore receive low perplexity. We eliminate a possible domain mismatch effect between the language model training data and the synthetic data by splitting the parallel corpus into three non-overlapping parts:

1. On 640K sentences pairs, we train a back-translation model,
2. On 4.1M sentence pairs, we take the source side and train a 5-gram Kneser-Ney language model (Heafield et al., 2013),
3. On the remaining 450K sentences, we apply the back-translation system using beam, sampling and top10 generation.

For the last set, we have genuine source sentences as well as synthetic sources from different generation techniques. We report the perplexity of our language model on all versions of the source data in Table 2. The results show that beam outputs receive higher probability by the language model compared to sampling, beam+noise and real source sentences. This indicates that beam search outputs are not as rich as sampling outputs or beam+noise. This lack of variability probably explains in part why back-translations from pure beam search provide a weaker training signal than alternatives.

Closer inspection of the synthetic sources (Table 3) reveals that sampled and noised beam outputs are sometimes not very adequate, much more so than MAP outputs, e.g., sampling often introduces target words which have no counterpart in the source. This happens because sampling sometimes picks highly unlikely outputs which are harder to fit (cf. Figure 2).

5.3 Low resource vs. high resource setup

The experiments so far are based on a setup with a large bilingual corpus. However, in resource poor settings the back-translation model is of much lower quality. Are non-MAP methods still more effective in such a setup? To answer this question, we simulate such setups by sub-sampling the training data to either 80K sentence-pairs or 640K sentence-pairs and then add synthetic data from sampling and beam search. We compare these smaller setups to our original 5.2M sentence bitext configuration. The accuracy of the

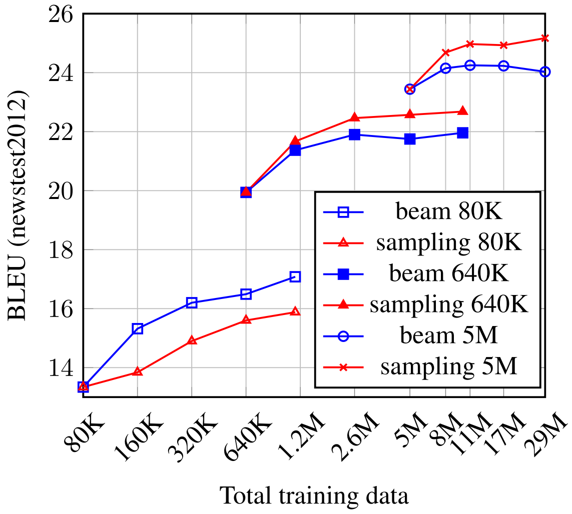

Figure 3: BLEU when adding synthetic data from beam and sampling to bitext systems with 80K, 640K and 5M sentence pairs.

German-English back-translation systems steadily increases with more training data: On newstest2012 we measure 13.5 BLEU for 80K bitext, 24.3 BLEU for 640K and 28.3 BLEU for 5M.

Figure 3 shows that sampling is more effective than beam for larger setups (640K and 5.2M bitexts) while the opposite is true for resource poor settings (80K bitext). This is likely because the back-translations in the 80K setup are of very poor quality and the noise of sampling and beam+noise is too detrimental for this brittle low-resource setting. When the setup is very small the very regular MAP outputs still provide useful training signal while the noise from sampling becomes harmful.

5.4 Domain of synthetic data

Next, we turn to two different questions: How does real human bitext compare to synthetic data in terms of final model accuracy? And how does the domain of the monolingual data affect results?

To answer these questions, we subsample 640K sentence-pairs of the bitext and train a back-translation system on this set. To train a forward model, we consider three alternative types of data to add to this 640K training set. We either add:

- the remaining parallel data (bitext),
- the back-translated target side of the remaining parallel data (BT-bitext),
- back-translated newscrawl data (BT-news).

The back-translated data is generated via sampling. This setup allows us to compare synthetic data to genuine data since BT-bitext and bitext share the same target side. It also allows us to estimate the value of BT data for domain adaptation since the newscrawl corpus (BT-news) is pure news whereas the bitext is a mixture of europarl and commoncrawl with only a small news-commentary portion. To assess domain adaptation effects, we measure accuracy on two held-out sets:

- newstest2012, i.e. pure newswire data.
- a held-out set of the WMT training data (valid-mixed), which is a mixture of europarl, commoncrawl and the small news-commentary portion.

Figure 4 shows the results on both validation sets. Most strikingly, BT-news performs almost as well as bitext on newstest2012 (Figure 4a) and improves the baseline (640K) by 2.6 BLEU. BT-bitext improves by 2.2 BLEU, achieving 83% of the improvement with real bitext. This shows that synthetic data can be nearly as effective as real human translated data when the domains match.

Figure 4b shows the accuracy on valid-mixed, the mixed domain valid set. The accuracy of BT-news is not as good as before since the domain of the BT data and the test set do not match. However, BT-news still improves the baseline by up to 1.2 BLEU. On the other hand, BT-bitext matches the domain of valid-mixed and improves by 2.7 BLEU. This trails the real bitext by only 1.3 BLEU and corresponds to 67% of the gain achieved with real human bitext.

In summary, synthetic data performs remarkably well, coming close to the improvements achieved with real bitext for newswire test data, or trailing real bitext by only 1.3 BLEU for valid-mixed. In absence of a large parallel corpus for news, back-translation therefore offers a simple, yet very effective domain adaptation technique.

5.5 Upsampling the bitext

We found it beneficial to adjust the ratio of bitext to synthetic data observed during training. In particular, we tuned the rate at which we sample data from the bitext compared to synthetic data. For example, in a setup of 5M bitext sentences and 10M synthetic sentences, an upsampling rate of 2 means that we double the frequency at which we

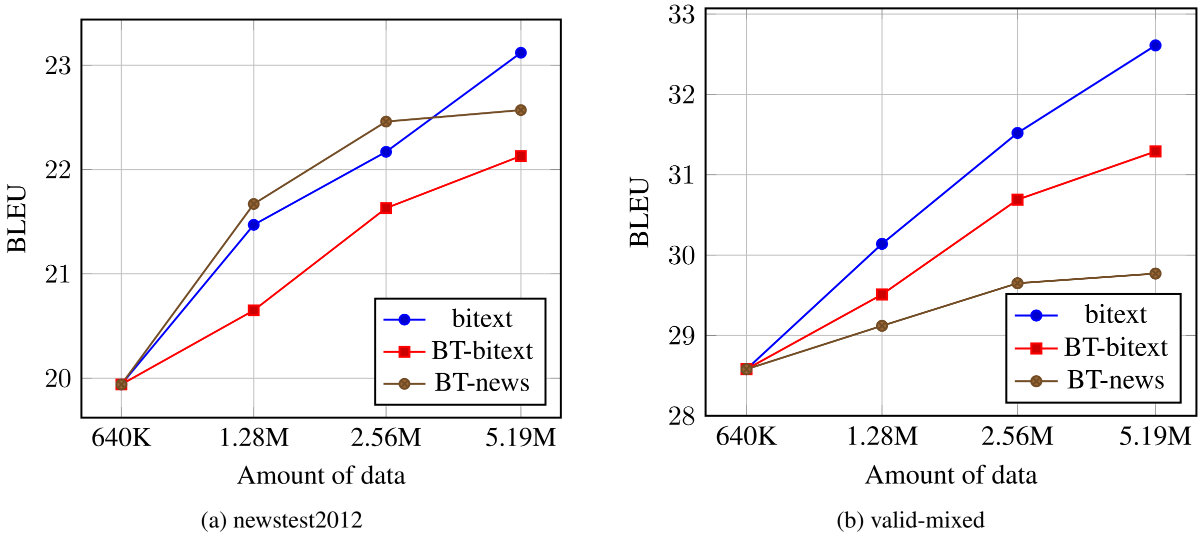

Figure 4: Accuracy on (a) newstest2012 and (b) a mixed domain valid set when growing a 640K bitext corpus with (i) real parallel data (bitext), (ii) a back-translated version of the target side of the bitext (BT-bitext), (iii) or back-translated newscrawl data (BT-news).

visit bitext, i.e. training batches contain on average an equal amount of bitext and synthetic data as opposed to 1/3 bitext and 2/3 synthetic data.

Figure 5 shows the accuracy of various upsampling rates for different generation methods in a setup with 5M bitext sentences and 24M synthetic sentences. Beam and greedy benefit a lot from higher rates which results in training more on the bitext data. This is likely because synthetic beam and greedy data does not provide as much training signal as the bitext which has more variation and is harder to fit. On the other hand, sampling and beam+noise require no upsampling of the bitext, which is likely because the synthetic data is already hard enough to fit and thus provides a strong training signal (§5.2).

5.6 Large scale results

To confirm our findings we experiment on WMT'14 English-French translation where we show results on newstest2013-2015. We augment the large bitext of 35.7M sentence pairs by 31M newscrawl sentences generated by sampling. To train this system we perform 300K training updates in 27h 40min on 128 GPUs; we do not upsample the bitext for this experiment. Table 4 shows tokenized BLEU and Table 5 shows detokenized BLEU.[2] To our knowledge, our baseline is the best reported result in the literature for newstest2014, and back-translation further improves upon this by 2.6 BLEU (tokenized).

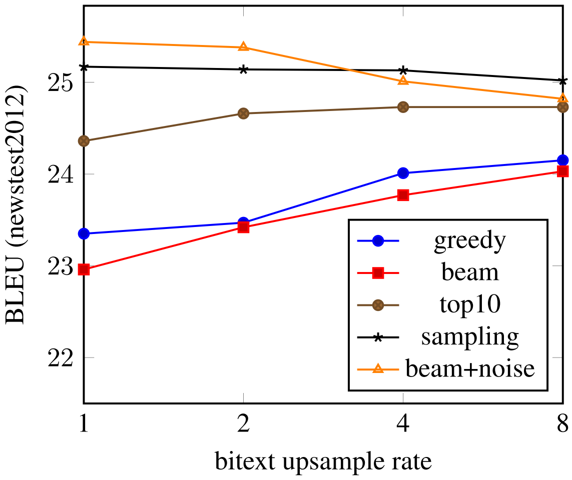

Figure 5: Accuracy when changing the rate at which the bitext is upsampled during training. Rates larger than one mean that the bitext is observed more often than actually present in the combined bitext and synthetic training corpus.

Finally, for WMT English-German we train on all 226M available monolingual training sentences and perform 250K updates in 22.5 hours on 128 GPUs. We upsample the bitext with a rate of 16 so that we observe every bitext sentence

---

[2] sacreBLEU signatures: BLEU+case.mixed+lang.en-fr+numrefs.1+smooth.exp+test.SET+tok.13a+version.1.2.7 with SET ∈ {wmt13, wmt14/full, wmt15}

[3] sacreBLEU signatures: BLEU+case.mixed+lang.en-LANG+numrefs.1+smooth.exp+test.wmt14/full+tok.13a+version.1.2.7 with LANG ∈ {de,fr}

	news13	**news14**	**news15**
bitext	36.97	42.90	39.92
+sampling	**37.85**	**45.60**	**43.95**

Table 4: Tokenized BLEU on various test sets for WMT English-French translation.

	news13	**news14**	**news15**
bitext	35.30	41.03	38.31
+sampling	**36.13**	**43.84**	**40.91**

Table 5: De-tokenized BLEU (sacreBLEU) on various test sets for WMT English-French.

16 times more often than each monolingual sentence. This results in a new state of the art of 35 BLEU on newstest2014 by using only WMT benchmark data. For comparison, DeepL, a commercial translation engine relying on high quality bilingual training data, achieves 33.3 tokenized BLEU .[4] Table 6 summarizes our results and compares to other work in the literature. This shows that back-translation with sampling can result in high-quality translation models based on benchmark data only.

6 Conclusions and future work

Back-translation is a very effective data augmentation technique for neural machine translation. Generating synthetic sources by sampling or by adding noise to beam outputs leads to higher accuracy than argmax inference which is typically used. In particular, sampling and noised beam outperforms pure beam by 1.7 BLEU on average on newstest2013-2017 for WMT English-German translation. Both methods provide a richer training signal for all but resource poor setups. We also find that synthetic data can achieve up to 83% of the performance attainable with real bitext. Finally, we achieve a new state of the art result of 35 BLEU on the WMT'14 English-German test set by using publicly available benchmark data only.

In future work, we would like to investigate an end-to-end approach where the back-translation model is optimized to output synthetic sources that are most helpful to the final forward model.

	En–De	En–Fr
a. Gehring et al. (2017)	25.2	40.5
b. Vaswani et al. (2017)	28.4	41.0
c. Ahmed et al. (2017)	28.9	41.4
d. Shaw et al. (2018)	29.2	41.5
DeepL	33.3	**45.9**
Our result	**35.0**	45.6
detok. sacreBLEU[3]	*33.8*	*43.8*

Table 6: BLEU on newstest2014 for WMT English-German (En–De) and English-French (En–Fr). The first four results use only WMT bitext (WMT'14, except for b, c, d in En–De which train on WMT'16). DeepL uses proprietary high-quality bitext and our result relies on back-translation with 226M newscrawl sentences for En–De and 31M for En–Fr. We also show detokenized BLEU (SacreBLEU).

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2017. Weighted transformer network for machine translation. *arxiv*, 1711.02132.

Antreas Antoniou, Amos J. Storkey, and Harrison Edwards. 2017. Data augmentation generative adversarial networks. *arXiv*, abs/1711.04340.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Nicola Bertoldi and Marcello Federico. 2009. Domain adaptation for statistical machine translation with monolingual resources. In *Workshop on Statistical Machine Translation (WMT)*.

Ondrej Bojar and Ales Tamchyna. 2011. Improving translation model by monolingual data. In *Workshop on Statistical Machine Translation (WMT)*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz Josef Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Conference on Natural Language Learning (CoNLL)*.

Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16:79–85.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Conference of the Association for Computational Linguistics (ACL)*.

---

[4] `https://www.deepl.com/press.html`

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ryan Cotterell and Julia Kreutzer. 2018. Explaining and generalizing back-translation through wake-sleep. *arXiv preprint arXiv:1806.04402*.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied Monolingual Data Improves Low-Resource Neural Machine Translation. In *Proc. of WMT*.

Tobias Domhan and Felix Hieber. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Angela Fan, Yann Dauphin, and Mike Lewis. 2018. Hierarchical neural story generation. In *Conference of the Association for Computational Linguistics (ACL)*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman-Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference of Machine Learning (ICML)*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv*, 1308.0850.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O. K. Li. 2018. Universal neural machine translation for extremely low resource languages. *arXiv*, 1802.05368.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv*, 1503.03535.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. 2017. On integrating a language model into neural machine translation. *Computer Speech & Language*, 45:137–148.

Thanh-Le Ha, Jan Niehues, and Alexander H. Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv*, 1611.04798.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv*, 1803.05567.

Soren Hauberg, Oren Freifeld, Anders Boesen Lindbo Larsen, John W. Fisher, and Lars Kai Hansen. 2016. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *AISTATS*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016a. Dual learning for machine translation. In *Conference on Advances in Neural Information Processing Systems (NIPS)*.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016b. Improved neural machine translation with smt features. In *Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, pages 151–157.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable Modified Kneser-Ney Language Model Estimation. In *Conference of the Association for Computational Linguistics (ACL)*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24.

Kenji Imamura, Atsushi Fujita, and Eiichiro Sumita. 2018. Enhancement of encoder and attention using target monolingual corpora in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 55–63.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics (TACL)*, 5:339–351.

Lukasz Kaiser, Aidan N. Gomez, and François Chollet. 2017. Depthwise separable convolutions for neural machine translation. *CoRR*, abs/1706.03059.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR*, abs/1610.10099.

Alina Karakanta, Jon Dehdari, and Josef van Genabith. 2017. Neural machine translation for low-resource languages without parallel corpora. *Machine Translation*, pages 1–23.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Philipp Koehn. 2010. *Statistical machine translation*. Cambridge University Press.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL Demo Session*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Patrik Lambert, Holger Schwenk, Christophe Servan, and Sadaf Abdul-Rauf. 2011. Investigations on translation model adaptation using monolingual data. In *Workshop on Statistical Machine Translation (WMT)*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. *arXiv*, 1803.05567.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xing Niu, Michael Denkowski, and Marine Carpuat. 2018. Bi-directional neural machine translation with synthetic parallel data. *arXiv preprint arXiv:1805.11213*.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018a. Analyzing uncertainty in neural machine translation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3956–3965.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018b. Scaling neural machine translation.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Conference of the Association for Computational Linguistics (ACL)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *International Conference on Learning Representations (ICLR) Workshop*.

Luis Perez and Jason Wang. 2017. The effectiveness of data augmentation in image classification using deep learning. *arxiv*, 1712.04621.

Alberto Poncelas, Dimitar Sht. Shterionov, Andy Way, Gideon Maillette de Buy Wenniger, and Peyman Passban. 2018. Investigating backtranslation in neural machine translation. *arXiv*, 1804.06189.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv*, 1804.08771.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. *Conference of the Association for Computational Linguistics (ACL)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Conference of the Association for Computational Linguistics (ACL)*.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proc. of NAACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Conference on Advances in Neural Information Processing Systems (NIPS)*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *arXiv preprint arXiv:1512.00567*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Conference on Advances in Neural Information Processing Systems (NIPS)*.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, , and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*.

Yingce Xia, Tao Qin, Wei Chen, Jiang Bian, Nenghai Yu, and Tie-Yan Liu. 2017. Dual supervised learning. In *International Conference on Machine Learning (ICML)*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bootstrapping Transliteration with Constrained Discovery for Low-Resource Languages

Shyam Upadhyay
University of Pennsylvania
Philadelphia, PA
`shyamupa@seas.upenn.edu`

Jordan Kodner
University of Pennsylvania
Philadelphia, PA
`jkodner@seas.upenn.edu`

Dan Roth
University of Pennsylvania
Philadelphia, PA
`danroth@seas.upenn.edu`

Abstract

Generating the English transliteration of a name written in a foreign script is an important and challenging step in multilingual knowledge acquisition and information extraction. Existing approaches to transliteration generation require a large (>5000) number of training examples. This difficulty contrasts with transliteration *discovery*, a somewhat easier task that involves picking a plausible transliteration from a given list. In this work, we present a bootstrapping algorithm that uses constrained discovery to improve generation, and can be used with as few as 500 training examples, which we show can be sourced from annotators in a matter of hours. This opens the task to languages for which large number of training examples are unavailable. We evaluate transliteration generation performance itself, as well the improvement it brings to cross-lingual candidate generation for entity linking, a typical downstream task. We present a comprehensive evaluation of our approach on nine languages, each written in a unique script.[1]

1 Introduction

Transliteration is the process of transducing names from one writing system to another (e.g., ओबामा in Devanagari to *Obama* in Latin script) while preserving their pronunciation (Knight and Graehl, 1998; Karimi et al., 2011). In particular, *back-transliteration* from foreign languages to English has applications in multilingual knowledge acquisition tasks including named entity recognition (Darwish, 2013) and information retrieval (Virga and Khudanpur, 2003). Two tasks feature prominently in the transliteration literature: *generation* (Knight and Graehl, 1998) which involves producing an appropriate transliteration for a given word in an open-ended way, and *discovery* (Sproat et al., 2006; Klementiev and Roth, 2008) which involves selecting an appropriate transliteration for a word from a list of candidates. This work develops transliteration generation approaches for low-resource languages.

Existing transliteration generation models require supervision in the form of source-target name pairs (≈5-10k), which are often collected from names in Wikipedia inter-language links (Irvine et al., 2010). However, most languages that use non-Latin scripts are under-represented in terms of such resources. Table 1 illustrates this issue, and the extra coverage one can achieve by extending to low-resource languages. A model that requires 50k name pairs as supervision can only support 6 languages, while one that just needs 500 could support 56. For a model to be widely applicable, it must function in low-resource settings.

# Name Pairs in Wikipedia		Languages	Scripts
$> 50,000$		6	5
$> 10,000$		18	14
$> 5,000$	*Previous Work*	24	15
$> 1,000$		45	22
> 500	*Our Approach*	56	23
> 0		93	30

Table 1: Cumulative number of person name pairs in Wikipedia inter-language links. While previous approaches for transliteration generation were applicable to only 24 languages (spanning 15 scripts), our approach is applicable to 56 languages (23 scripts). When counting scripts we exclude variants (e.g., all Cyrillic scripts and variants count as one).

We propose a new bootstrapping algorithm that uses a weak generation model to guide discovery of good transliterations, which in turn aids future bootstrapping iterations.[2] By carefully controlling the interaction of discovery and the generation model via constrained inference, we show

---

[1] code at `github.com/shyamupa/hma-translit`.

[2] All generative approaches are also capable of discovery, by using the posterior $P(y \mid x)$ to select the most likely candidate transliteration, while the opposite is not true.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 501–511
Brussels, Belgium, October 31 - November 4, 2018.

how to bootstrap a generation model using a dictionary of names in English, a list of words in the foreign script, and little initial supervision (≈500 name pairs). To the best of our knowledge, ours is the first work to accomplish transliteration generation in such a low-resource setting.

We demonstrate the practicality of our approach in truly low-resource scenarios and downstream applications through two case studies. First, in §8.1 we show that one can obtain the initial supervision from a *single* human annotator within a few hours for two languages – Armenian and Punjabi. This is a realistic scenario where language access is limited to a single native informant. Second, in §8.2 we show that our approach benefits a typical downstream application, namely candidate generation for cross-lingual entity linking, by improving recall on two low-resource languages – Tigrinya and Macedonian. We also present an analysis (§7) of the inherent challenges of transliteration, and the trade-off between native (i.e., source) and foreign (i.e., target) vocabulary.

2 Related Work

We briefly review the limitations of existing generation and discovery approaches, and provide an overview of how our work addresses them.

Transliteration Generation (Haizhou et al., 2004; Jiampojamarn et al., 2009; Ravi and Knight, 2009; Jiampojamarn et al., 2010; Finch et al., 2015, *inter alia*) requires generous amount of name pairs (≈5-10k) in order to learn to map words in the source script to the target script. While some approaches (Irvine et al., 2010; Tsai and Roth, 2018) use Wikipedia inter-language links to identify name pairs for supervision, a truly low-resource language (like Tigrinya) is likely to have limited Wikipedia presence as well.

Transliteration Discovery (Sproat et al., 2006; Chang et al., 2009) is considerably easier than generation, owing to the smaller search space. However, discovery often uses features derived from resources that are unavailable for low-resource languages, like comparable corpora (Sproat et al., 2006; Klementiev and Roth, 2008).

A key limitation of discovery is the assumption that the correct transliteration(s) is in the list of candidates $\mathcal{N}$. Since discovery models always pick *something* from $\mathcal{N}$, they can produce false positives, if no correct transliteration is present in $\mathcal{N}$. To overcome this, it is prudent to develop generation models which can handle input for which the transliteration does not belong in $\mathcal{N}$.

Our Work We show that a weak generation model can be iteratively improved using constrained discovery. In particular, our work uses a weak generation model to discover new training pairs, using constraints to drive the bootstrapping. Our generation model is inspired by the success of sequence to sequence generation models (Sutskever et al., 2014; Bahdanau et al., 2015) for string transduction tasks like inflection and derivation generation (Faruqui et al., 2016; Cotterell et al., 2017; Aharoni and Goldberg, 2017; Makarov et al., 2017). Our bootstrapping framework can be viewed as an instance of constraint driven learning (Chang et al., 2007, 2012).

3 Transliteration Generation with Hard Monotonic Attention - Seq2Seq(HMA)

We view generation as a string transduction task and use a sequence to sequence (Seq2Seq) generation model that uses *hard monotonic attention* (Aharoni and Goldberg, 2017), henceforth referred to as Seq2Seq(HMA). During generation, Seq2Seq(HMA) directly models the *monotonic* source-to-target sequence alignments, using a pointer that attends to a *single* input character at a time. Monotonic attention is a natural fit for transliteration because even though the number of characters needed to represent a sound in the source and target language vary, the sequence of sounds is presented in the same order.[3] We review Seq2Seq(HMA) below, and describe how it can be applied to transliteration generation.

Encoding Input Word Let Σ_f be the source alphabet and Σ_e be the English alphabet. Let $x = (x_1, x_2, \cdots, x_n)$ denote an input word where each character $x_i \in \Sigma_f$. The characters are first encoded using a embedding matrix $\mathbf{W} \in \mathbb{R}^{|\Sigma_f| \times d}$ to get character embeddings $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$ where each $\boldsymbol{x}_i \in \mathbb{R}^d$. These embeddings are fed into a bidirectional RNN encoder to generate encoded vectors $\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_n$ where each $\boldsymbol{h}_i \in \mathbb{R}^{2k}$, and k is the size of output vector of the forward (and backward) encoder. The encoded vectors $\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_n$ are then fed into the decoder.

---

[3] Many Indic scripts, that sometimes write vowels before the consonants they are pronounced after, seem to violate this claim, but Unicode representations of these scripts actually preserve the consonant-vowel order.

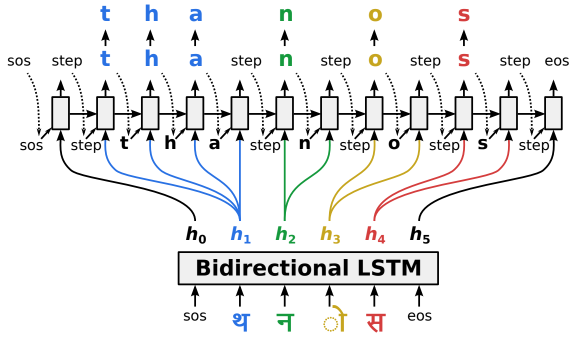

Figure 1: Transliteration using Seq2Seq transduction with Hard Monotonic Attention, or Seq2Seq(HMA). The figure shows how decoding proceeds for transliterating “थनोस” to “thanos”. During decoding, the model attends to a source character (e.g.,थ shown in blue) and outputs target characters (t, h, a) until a *step* action is generated, which moves the attention position forward by one character (to न), and so on.

Monotonic Decoding with Hard Attention Figure 1 illustrates the decoding process. The decoder RNN generates a sequence of actions $\{s_1, s_2, \cdots\}$, such that each $s_i \in \Sigma_e \cup \{step\}$. The *step* action controls an attention position a, attending on input character x_a, with encoded vector $\boldsymbol{h}_a$. Each action s_i is embedded into $\boldsymbol{s}_i \in \mathbb{R}^d$ using a output embedding matrix $\mathbf{A} \in \mathbb{R}^{(|\Sigma_e|+1)\times d}$. At any time during decoding, the decoder uses its last hidden state, the embedding of the previous action $\boldsymbol{s}_i$ and the encoded vector $\boldsymbol{h}_a$ of the current attended position to generate the next action s_{i+1}. If the generated action is *step*, the decoder increments the attention position by one. This ensures that the decoding is monotonic, as the attention position can only move forward or stay at the same position during generation. We use `Inference`(G, x) to refer to the above decoding process for a trained generation model G and input word x.

Training requires the oracle action sequence $\{s_i\}$ for input $x_{1:n}$ that generates the correct transliteration $y_{1:m}$. The oracle sequence is generated using the train name pairs and Algorithm 1 in Aharoni and Goldberg (2017), with the character-level alignment between $x_{1:n}$ and $y_{1:m}$ being generated using the algorithm in Cotterell et al. (2016).

Inference Strategies We describe an unconstrained and a constrained inference strategy to select the best transliteration $\hat{y}$ from a beam $\{y_i\}_{i=1}^k$ of transliteration hypotheses, sorted in descending order by likelihood. The constrained strategy use a name dictionary $\mathcal{N}$, to guide the inference. These strategies are applicable to any generation model.

- **Unconstrained (U)** selects the most likely item y_1 in the beam as $\hat{y}$.
- **Dictionary-Constrained (DC)** selects the highest scoring hypothesis that is present in $\mathcal{N}$, and defaults to y_1 if none are in $\mathcal{N}$.

It is tempting to disallow the model from generating hypotheses which are not in the dictionary $\mathcal{N}$. However, dictionaries are always incomplete, and restricting the search to generate from $\mathcal{N}$ inevitably leads to incorrect predictions if the correct transliteration is not in $\mathcal{N}$. This is essentially the same as the problem inherent to discovery models.

Other Strategies in Previous Work A related constrained inference strategy was proposed by Lin et al. (2016), who use a entity linking system (Wang et al., 2015) to correct and re-rank hypotheses, using any available context to aid hypothesis correction. Our constrained inference strategy is much simpler, requiring only a name dictionary $\mathcal{N}$. We experimentally show that our approach outperforms that of Lin et al. (2016).

4 Low-Resource Bootstrapping

Low-resource languages will have a limited number of name pairs for training a generation model. To learn a good generation model in this setting, we propose a new bootstrapping algorithm, that uses *constrained discovery* to mine name pairs to re-train the generation model. Our algorithm requires a small (≈500) seed list of name pairs $\mathcal{S}$ for supervision, a dictionary $\mathcal{N}$ containing names in English, and a list of words $\mathcal{V}_f$ in the foreign script.

Below we describe our algorithm and the constraints used to guide discovery of new name pairs.

4.1 The Bootstrapping Algorithm

Algorithm 1 shows the pseudo-code of the bootstrapping procedure. We initialize a weak generation model G_0 using a seed list of name pairs $\mathcal{S}$ (line 1). At iteration t, the current generation model G_t produces the top-k transliteration hypotheses $\{y_i\}_{i=1}^k$ for each word $x \in \mathcal{V}_f$ (line 5). A source word and hypothesis pair (x, y_i), is added to the set of mined name pairs $\mathcal{B}$ if they satisfy a set of discovery constraints (described below) (line 8). A new generation model G_{t+1} is trained for the next iteration using the union of the seed list $\mathcal{S}$ and the mined name pairs $\mathcal{B}$ (line 12). $\mathcal{B}$ is purged after every iteration (line 3) to prevent G_{t+1} from being influenced by possibly incorrect name pairs mined in

Algorithm 1 Bootstrapping a Transliteration Generation Model via Constrained Discovery

```
Input:
    English name dictionary $\mathcal{N}$; Seed training pairs $\mathcal{S}$;
    Vocabulary in the target language $\mathcal{V}_f$.
Hyper-parameters:
    initial minimum length threshold $L_0^{min}$;
    minimum likelihood threshold $\delta^{min}$;
    length ratio tolerance $\epsilon$.
Output: Generation model $G_T$
$G_0 = \texttt{train}(\mathcal{S})$                    ▷ init. generation model.
while not converged do
    $\mathcal{B} = \emptyset$                           ▷ purge mined set.
    for $x$ in $\mathcal{V}_f$ do
        $\{y_i\}_{i=1}^{k} = \text{argtop}_k\ \texttt{Inference}(G_t, x)$
        for $y_i$ in $\{y_i\}_{i=1}^{k}$ do
            if $(x, y_i)$ satisfies constraints in §4.2 then
                $\mathcal{B} = \mathcal{B} \cup \{(x, y_i)\}$   ▷ add to mined set.
            end if
        end for
    end for
    $G_{t+1} = \texttt{train}\ (\mathcal{S} \cup \mathcal{B})$
    $L_{t+1}^{min} = L_t^{min} - 1$                     ▷ reduce length threshold.
    $t = t + 1$                                         ▷ track iteration
end while
```

earlier iterations. The algorithm converges when accuracy@1 stops increasing on a development set. We note that our bootstrapping approach is applicable to any transliteration generation model.

To ensure that high quality name pairs are added to the mined set $\mathcal{B}$ during bootstrapping, we use the following discovery constraints.

4.2 Discovery Constraints

A word-transliteration pair (x, y) is added to the set of mined pairs $\mathcal{B}$, only if all the following constraints are satisfied,

1. $y \in \mathcal{N}$. i.e., y belongs in the dictionary.
2. $\text{P}(y \mid x) > \delta^{min}$. The model is sufficiently confident about the transliteration.
3. The ratio of lengths $\frac{|y|}{|x|}$ should be close to the average ratio estimated from $\mathcal{S}$ (Matthews, 2007). We encode this using the constraint $|\frac{|y|}{|x|} - r(\mathcal{S})| \leq \epsilon$, where ϵ is a tunable tolerance and $r(\mathcal{S})$ is the average ratio in $\mathcal{S}$.
4. $|y| > L_t^{min}$. We found that false positives were more likely to be short hypotheses in early iterations. As the model improves with each iteration, L_t^{min} is lowered to allow more new pairs to be mined.

We note that our bootstrapping algorithm can be formulated as an instance of constraint driven learning (Chang et al., 2007, 2012).

5 Experimental Setup

Unless otherwise specified, we evaluate all generation models using the best model prediction $\hat{y}$ using acc@1 against the reference transliteration y^*.

Training and Evaluation Dataset We use the train and development sets from the Named Entities Workshop 2015 (Duan et al., 2015) (NEWS2015) for Hindi (hi), Kannada (kn), Bengali (bn), Tamil (ta) and Hebrew (he) as our train and evaluation set.[4] The size of the train set was ∼12k, 10k, 14k, 10k and 10k respectively, and all evaluation sets were ∼1k.

For the low resource experiments, we subsample 500 examples from each train set in the NEWS2015 dataset using five random seeds and report the averaged results. We also set aside a 1k name pairs from the corresponding NEWS2015 train set of each language as development data. The foreign script portion of the remaining train data is used as $\mathcal{V}_f$ in the bootstrapping algorithm.

Model and Tuning Details We implemented Seq2Seq(HMA) using PyTorch.[5] We used 50 dimensional character embeddings, and single layer GRU (Cho et al., 2014) encoder with 20 hidden states for all experiments. The Adam (Kingma and Ba, 2014) optimizer was used with default hyperparameters, a learning rate of 0.001, a batch size of 1, and maximum of 20 iterations in all experiments. Beam search used a width of 10. For low-resource experiments, all bootstrapping parameters were tuned on the development data set aside above. L_0^{min} is chosen from $\{10, 15, 20, 25\}$.

Name Dictionary We use a name dictionary of 1.05 million names constructed from the English Wikipedia (dump dated 05/20/2017) by taking the list of title tokens in Wikipedia sorted by frequency, and removing tokens which appears only once.

5.1 Comparisons

We compare with the following generation models:

P&R (Pasternack and Roth, 2009) A probabilistic transliteration generation approach that learns latent alignments between substrings in the source and the target words. The model is trained to score all possible segmentation and their alignments, using an EM-like algorithm.

---

[4] Test set was not available since shared task concluded.

[5] github.com/pytorch

DirecTL+ (Jiampojamarn et al., 2009) A HMM-like discriminative string transduction model that predicts the output transliteration using many-to-many alignments between the source word and target transliteration. Following Jiampojamarn et al. (2009), we use the m2m-aligner (Jiampojamarn et al., 2007) to generate the many-to-many alignments, and the public implementation of DirecTL+ to train models.[6]

RPI-ISI (Lin et al., 2016) A transliteration approach that uses a language-independent entity linking system (Wang et al., 2015) to jointly correct and re-rank the hypotheses produced by the generation model. We compare to both the unconstrained inference (U) approach and the entity linking constrained inference (+EL) approach.

Seq2Seq w/ Att A sequence to sequence generation model which uses soft attention as described in (Bahdanau et al., 2015). This model does not enforce monotonicity at inference time, and serves as direct comparison for Seq2Seq(HMA).

6 Experiments

This section aims to analyze: **(a)** how effective is Seq2Seq(HMA) for transliteration generation when provided all available supervision (§6.1)? and **(b)** how effective is the bootstrapping algorithm in the low-resource setting when only 500 examples are available (§6.2)?

6.1 Full Supervision Setting

We compare Seq2Seq(HMA) with previous approaches when provided all available supervision, to see how it fares under standard evaluation.

Results in the unconstrained inference (U) setting (Table 2 top 5 rows) shows Seq2Seq(HMA), denoted by "Ours", outperforms previous approaches on Hindi, Kannada, and Bengali, with almost 3-4% gains. Improvements over the Seq2Seq with Attention (Seq2Seq w/ Att) model demonstrate the benefit of imposing the monotonicity constraint in the generation model. On Tamil and Hebrew, Seq2Seq(HMA) is at par with the best approaches, with negligible gap (∼0.3) in scores. Overall, we see that Seq2Seq(HMA) can achieve better (and sometimes competitive) scores than state-of-the-art approaches in full supervision settings. When comparing approaches which use constrained inference (Table 2, rows 6 and 7), we see that using dictionary-constrained inference (as in Ours(DC)) is more effective than using a entity-linking model for re-ranking (RPI-ISI + EL).

Lang. → **Approach ↓**	hi	kn	bn	ta	he	Avg.
Full Supervision Setting (5-10k examples)						
Seq2Seq w/ Att (U)	35.5	33.4	46.1	17.2	20.3	30.5
P&R (U)	37.4	31.6	45.4	**20.2**	18.7	30.7
DirecTL+ (U)	38.9	34.7	48.4	19.9	16.8	31.7
RPI-ISI (U)	40.3	29.8	49.4	**20.2**	21.5	32.2
Ours(U)	**42.8**	**38.9**	**52.4**	**20.5**	**23.4**	**35.6**
Approaches Using Constrained Inference						
RPI-ISI + EL	44.8	37.6	52.0	**29.0**	**37.2**	40.1
Ours(DC)	**51.8**	**43.3**	**56.6**	28.0	36.1	**43.2**
Low-Resource Setting (500 examples)						
Seq2Seq w/ Att (U)	17.0	13.6	14.5	6.0	9.5	12.1
P&R (U)	21.1	16.6	34.2	9.4	13.0	18.9
DirecTL+ (U)	26.6	25.3	35.5	11.8	10.7	22.0
Ours(U)	29.1	27.7	37.7	11.5	16.2	24.4
Ours(U) + Boot.	**40.1**	**35.1**	**50.3**	**17.8**	**22.8**	**33.2**

Table 2: Comparing different approaches on the NEWS 2015 dataset using acc@1 as the evaluation metric. "Ours" denotes the Seq2Seq(HMA) model, with (.) denoting the inference strategy. Numbers for RPI-ISI are from Lin et al. (2016).

6.2 Low-Resource Setting

In Table 2 (rows under "Low-Resource Setting"), we evaluate different models in a low-resource setting when provided only 500 name pairs as supervision. Results are averaged over 5 different random sub-samples of 500 examples.

The results clearly demonstrate that all generation models suffer a drop in performance when provided limited training data. Note that models like Seq2Seq with Attention suffer a larger drop than those which enforce monotonicity, suggesting that incorporating monotonicity into the inference step in the low-resource setting is essential. After bootstrapping our weak generation model using Algorithm 1, the performance improves substantially (last row in Table 2). On almost all languages, the generation model improves by at least 6%, with performance for Hindi and Bengali improving by more than 10%. Bootstrapping results for the languages are within 2-4% of the best model trained with all available supervision.

To better analyze the progress of the transliteration model during bootstrapping, we plot the accuracy@1 of the current transliteration model after each bootstrapping iteration for each of the languages (solid lines in Figure 2). For reference, we also show the best performance for a gener-

---

[6] `https://code.google.com/p/directl-p`

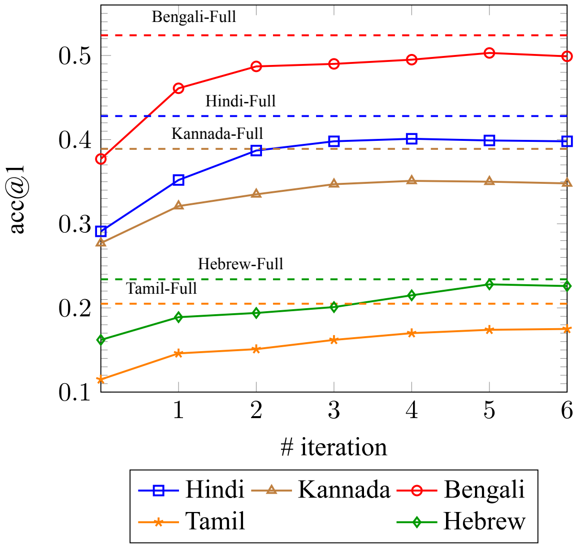

Figure 2: Plot showing acc@1 after each bootstrapping iteration for Hindi, Kannada, Bengali, Tamil and Hebrew, starting with only 500 training pairs as supervision. For comparison, the acc@1 of a model trained with all available supervision is also shown (respective dashed lines, marked X-Full).

ation model using all available supervision from §6.1 (dotted horizontal lines in Figure 2). From Figure 2, we can see that almost after 5 bootstrapping iterations, the generation model attains competitive performance to respective state-of-the-art models trained with full supervision.

6.3 Error Analysis

Though our model is state of the art, it does present a few weaknesses. We have found that the dictionary sometimes misleads the model during constrained inference. For example, the correct transliteration "vidyul" of the Hindi विद्युल, is not present in the dictionary, but another hypothesis "vidul" is. Another issue comes from the proportion of native (i.e., from the source language) and foreign (i.e., from English or other languages) names in the training data. It is usually not the case that the source and target scripts have the same transliteration rules. For example, य in Hindi might represent *ya* in English or Hindi names, but *ja* in German. Similarly, while अ should be *a* in Hindi names, it could be any of a few vowels in English. The NEWS2015 dataset does not report a native/foreign ratio, but by our estimation, it is about 70/30 for each language. This native and foreign names dichotomy are some of the inherent challenges in transliteration, that we discuss in detail in the next section.

7 Challenges Inherent to Transliteration

The fact that all models in Table 2 perform well or poorly on the same languages suggests that most of the observed performance variation is the result of factors *intrinsic* to the specific languages. Here we analyze some challenges that are inherent to the transliteration task, and explain why the performance ceiling is well under 100% for all languages, and lower for languages like Tamil and Hebrew than the others.

7.1 Source and Target-Specific Issues

Source-Driven Some transliteration errors are due to ambiguities in the source scripts. For instance, the Tamil script uses a single character to denote {*ta*, *da*, *tha*, *dha*}, a single character for {*ka*, *ga*, *kha*, *gha*}, etc., while the rest of the Indian scripts have unique characters for each of these. Thus, names like *Har**t**ley* and *Har**d**ley* are entirely indistinguishable in Tamil but are distinguishable in the other scripts. We illustrate this problem by transliterating back and forth between Tamil and Hindi. When transliterating Hindi→Tamil, the model achieves an accuracy of 31%, which drops to 15% when transliterating Tamil→Hindi, suggesting that the Tamil script is more ambiguous.

The Hebrew script also introduces error because it tends to omit vowels or write them ambiguously, leaving the model to guess between plausible choices. For example, the word מלך could be transliterated *melech* "king" just as easily as *malach* "he ruled." When Hebrew does write vowels, it reuses consonant letters, again ambiguously. For example, ה can be used to express *a* or *e*, so שמונה can be either *shmona* or *shmone* "eight masculine/feminine". The script also does not reliably distinguish *b* from *v* or *p* from *f*, among others.

All languages run into problems when they are faced with writing sounds that they do not natively distinguish. For example, Hindi does not make a distinction between *w* and *v*, so both *vest* and *west* are written as वेस्ट in its script.

These script-specific deficiencies explains why all models struggle on Tamil and Hebrew relative to the others. These issues cannot be completely resolved without memorizing individual source-target pairs and leveraging context.

Target-Driven Some errors arise from the challenges presented by target script (here Latin script for English). To handle English's notoriously convoluted orthography, a model has to infer silent let-

	Native	Foreign	Ratio
Hindi	45.1	31.4	1.44
Bengali	63.1	20.1	3.14
Kannada	42.6	23.1	1.84
Tamil	24.3	05.2	4.67

Table 3: Acc@1 for native and foreign words for four languages (§7.2). Ratio is native performance relative to foreign.

ters, decide whether to use *f* or *ph* for /f/; use *k*, *c*, *ck*, *ch*, or *q* for /k/, and so on. The problem is made worse because English is not the only language that uses Latin script. For example, German names like *Schmidt* should be written with *sch* instead of *sh*, and for French names like *Margot* and *Margeau* (which are pronounced the same), we have to resort to memorization. The arbitrariness extends into borrowings from the source languages as well. For example, the Indian name *Bangalore* is written with a silent-*e*, and the name *Lakshadweep* contains *ee*, instead of the expected *i*.

7.2 Disparity between Native and Foreign

All these issues come together to create a performance disparity between *native* names, which are well-integrated into the source language etymologically (Indian names like *Jasodhara* or *Ramanathan* for Hindi), and *foreign* names (French *Grenoble* or Japanese *Honshu* for Hindi), which are not. The above datasets include an unspecified mix of native and foreign names. This is a problem since any model must learn essentially separate transliteration schemes for each.

To quantify the effect of this, we annotate native and foreign names in the test split of the four Indian languages, and evaluate performance for both categories. Table 3 shows that our model performs significantly better on native names for all the languages. A possible reason for is that the source scripts were designed for writing native names (e.g., Tamil script lacks separate {*ta*, *da*, *tha*, *dha*} characters because the Tamil language does not distinguish these sounds). Furthermore, foreign names have a wide variety of origins with their own conventions as discussed in §7.1. The performance gap is proportionally greatest for Tamil, likely due to its script.

8 Case Studies

In this section, we evaluate the practical utility of our approach in low-resource settings and for downstream applications through two case studies. We first show that obtaining an adequate seed list is possible with a few hours of manual annotation (§8.1) from a *single* human annotator. We then show the positive impact that our approach has on a downstream task, by evaluating its contribution to candidate generation for Tigrinya and Macedonian entity linking (§8.2).

Language	Monolingual Corpus	Vocabulary
Punjabi	Corpus ILCI-II♠	30k
Armenian	TED♣	50k
Tigrinya	Habit Project♦	225k
Macedonian	TED♣	60k

♦=`habit-project.eu/wiki/TigrinyaCorpus`,
♠=`tdil-dc.in`,
♣=`github.com/ajinkyakulkarni14/TED-Multilingual-Parallel-Corpus`

Table 4: Corpora used for obtaining foreign vocabulary $\mathcal{V}_f$ for bootstrapping in the case studies in §8.1 and §8.2.

8.1 Manual Annotation

The manual annotation exercises simulate a low-resource setting with only a single human annotator is available. We judge the usability of the annotations by training models on them and evaluating the models on test sets of 1000 names each, obtained from Wikipedia inter-language links. For bootstrapping experiments, we use the corpora shown in Table 4 to obtain foreign vocabulary $\mathcal{V}_f$.

Languages Studied We investigate performance on two languages: Armenian and Punjabi.

Spoken in Armenia and Turkey, Armenian is an Indo-European language with no close relatives. It has Eastern and Western dialects with different spelling conventions. Armenian Wikipedia is primarily written in the Eastern dialect, while our annotator was a native Western speaker.[7]

Punjabi is an Indic language from Northwest India and Pakistan that is closely related to Hindi. Our annotator grew up primarily speaking Hindi.

Annotation Guidelines Annotators were given two tasks. First, they were asked to write two names and their English transliterations for each letter in the source script: one beginning with the letter and another containing it elsewhere. (e.g. "**J**ulia" and "Ben**j**amin" for the letter "j" if the source were English). The is done to ensure good coverage over the alphabet. Next, annotators were shown a list of English words and were asked to

---

[7] The annotator produced Western Armenian which was mechanically mapped to "Eastern" by swapping five Armenian character pairs: ղ/տ, պ/բ , ք/կ , ձ/ծ, ճ/ջ

Lang. → **Approach** ↓	Punjabi	Armenian
Ours(U)	33.4	49.9
Ours(U) + Bootstrapping	44.5	55.8
Annotation Time (hours)	5	4

Table 5: Acc@1 using human annotated seed set and bootstrapping the Seq2Seq(HMA) model. Both languages perform well relative to the other languages investigated so far. Both annotation sub-tasks took roughly the same time.

provide plausible transliteration(s) into the target script. The list had a mix of recognizable foreign (e.g., *Clinton*, *Helsinki*) and native names (e.g., *Sarkessian*, *Yerevan* for Armenian).

We collected about 600 and 500 annotated pairs respectively for Armenian and Punjabi. Table 5 shows that the performance of the models trained on the annotated data is comparable to that on the standard test corpora for other languages. This show that our approach is robust to human inconsistencies and regional spelling variations, and that obtaining an adequate seed list is possible with just a few hours of manual annotation.

8.2 Candidate Generation (CG)

Since transliteration is an intermediate step in many downstream multilingual information extraction tasks (Darwish, 2013; Kim et al., 2012; Jeong et al., 1999; Virga and Khudanpur, 2003; Chen et al., 2006), it is possibly to gauge its performance extrinsically by the impact it has on such tasks. We use the task of *candidate generation* (CG), which is a key step in cross-lingual entity linking.

The goal of cross-lingual entity linking (McNamee et al., 2011; Tsai and Roth, 2016; Upadhyay et al., 2018) is to ground spans of text written in any language to an entity in a knowledge base (KB). For instance, grounding **[Chicago]** in the following German sentence to `Chicago_(band)`.[8]

[Chicago] *wird in Woodstock aufzutreten.*

The role of CG in cross-lingual entity linking is to create a set of plausible entities given a string while ensuring the correct KB entity belongs to that set. For the above German sentence, it would provide a list of possible KB entities for the string *Chicago*: `Chicago_(band)`, `Chicago_(city)`, `Chicago_(font)`, etc., so that entity linking can select the band. Foreign scripts pose an additional challenge for CG because they must be transliterated before they are passed on to candidate generation. For instance, any mention of "Chicago" in Amharic must first be transliterated from **ቺካጎ**.

Most approaches for CG use Wikipedia inter-language links to generate the lists of candidates (Tsai and Roth, 2016). While recent approaches such as Tsai and Roth (2018) have resorted to name translation for CG, they require over 10k examples for languages written in non-Latin scripts, which is prohibitive for low-resource languages with little Wikipedia presence.

Candidate Generation with Transliteration We evaluate the extent to which our approach improves recall of a naive CG baseline that generates candidates by performing exact name match. For each span of text to be linked (or *query mention*), we first check if the naive name matching strategy finds any candidates in the KB. If none are found, the query mention is back-transliterated to English, and at most 20 candidates are generated using a inverted-index from English names to KB entities. The evaluation metric is recall@20, i.e., if the gold KB entity is in the top 20 candidates. We use Tigrinya and Macedonian as our test languages.

Tigrinya is a South Semitic language related to Amharic, written in the Ethiopic script, and spoken primarily in Eritrea and northern Ethiopia. The Tigrinya Wikipedia has <200 articles, so we use inter-language links (∼7.5k) from the Amharic Wikipedia instead to extract 1k name pairs for the seed set. We use the monolingual corpus in Table 4 for bootstrapping and evaluate on the *unsequestered set* provided under the NIST LoReHLT evaluation, containing 4,630 query mentions.

The Ethiopic script is an alphasyllabary, where each character is consonant-vowel pair. For example, the character **መ** is *mä*, **ሚ** with a tail is *mi*, and **ሞ** with a line is *mo*. With 26 consonants and 8 vowels, this leads to a set of >200 characters creating a sparsity problem since each character has its own Unicode code point. However, the code points are organized so that they can be automatically split[9] into unique consonant and vowel codes *without explicitly understanding the script*. We assign arbitrary ASCII codes to each consonant and vowel so that **መ**/*mä* becomes "D 1" and **ሞ**/*mo* becomes "D 6." This consonant-vowel splitting (CV-split) reduces the number of unique input characters to 55.

---

[8] Translation: Chicago will perform at Woodstock.

[9] Consonant = Unicode / 8; Vowel = Unicode % 8

Approach	Recall@20
Tigrinya	
Name match (baseline)	31.4
Ours	35.6
Ours (CV-split)	41.3
Ours (CV-split) + Bootstrapping	**46.2**
Macedonian	
Name match (baseline)	33.6
Ours	72.2
Ours + Bootstrapping	**76.8**

Table 6: Comparing candidate recall@20 for different approaches on Tigrinya and Macedonian. CV-split refers to consonant-vowel splitting. Using our transliteration generation model with bootstrapping yields the highest recall, improving significantly over a name match baseline.

Macedonian is a South Slavic language closely related to the languages of the former Yugoslavia and written in a local variant of the Cyrillic alphabet similar to Serbian's. We use the Macedonian test set constructed by McNamee et al. (2011) containing 1956 query mentions. A seed set of 1k name pairs was obtained from the inter-language Wikipedia links for Macedonian, and the monolingual corpus from Table 4 is used for bootstrapping.

Candidate Generation Results Table 6 shows the results for the two languages. For Tigrinya, candidate generation with transliteration improves on the baseline by 4.2%. Splitting the characters (CV-split) gives another 5.7%, and adding bootstrapping gives 4.9% more. Our approach yields an overall 14.8% improvement in recall over the baseline, showing that we can effectively exploit the little available supervision by bootstrapping. Macedonian yields more dramatic results, where transliteration provides 38.6% improvement (more than double the baseline), with bootstrapping providing another 4.6%. The differences between Tigrinya and Macedonian is likely due both to their test sets, corpora and writing systems.

9 Conclusion and Future Work

We presented a new transliteration generation model, namely Seq2Seq(HMA), and a new bootstrapping algorithm that can iteratively improve a weak generation model using constrained discovery. The model presented here achieves state-of-the-art results on typical training set sizes, and more importantly, works well in a low-resource setting with the aid of the bootstrapping algorithm. The key benefit of the bootstrapping approach is that it can "recover" most of the performance lost in the low-resource setting when little supervision is available by training with a smaller seed set, an English name dictionary, and a list of unannotated words in the target script. Additionally, our bootstrapping algorithm admits any generation model, giving it wide applicability. Through case studies, we showed that collecting an adequate seed list is practical with a few hours of annotation. The benefit of incorporating our transliteration approach in a downstream task, namely candidate generation, was also demonstrated. Finally, we discussed some of the inherent challenges of learning transliteration and the deficits of existing training sets.

There are several interesting directions for future work. Performing model combination, either by developing hybrid transliteration models (Nicolai et al., 2015) or by ensembling (Finch et al., 2016), can further improve low resource transliteration. Jointly leveraging similarities between related languages, such as writing systems or phonetic properties (Kunchukuttan et al., 2018), also shows promise for low-resource settings. Our analysis suggests value in revisiting "transliteration in context" approaches (Goto et al., 2003; Hermjakob et al., 2008), especially for languages like Hebrew. We would also like to expand on the analyses provided in §7 which uncover challenges inherent to the transliteration task, particularly the impact of the native/foreign distinction in the train and test data, the difficulties posed by specific scripts or pairs of scripts, and how these impact both back- and forward-transliteration. Recent work from Merhav and Ash (2018) suggests many useful analyses that we would like to incorporate.

Acknowledgments

The authors thank Mitch Marcus, Snigdha Chaturvedi, Stephen Mayhew, Nitish Gupta, Dan Deutsch, and the anonymous reviewers for their useful comments. We are grateful to the Armenian and Punjabi annotators for help with the case studies.

This work was supported under DARPA LORELEI by Contract HR0011-15-2-0025, Agreement HR0011-15-2-0023 with DARPA, and an NDSEG fellowship for the second author. Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological Inflection Generation with Hard Monotonic Attention. In *Proc. of ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proc. of ICLR*.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Yuancheng Tu. 2009. Unsupervised Constraint Driven Learning For Transliteration Discovery. In *Proc. of NAACL*.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding Semi-Supervision with Constraint-Driven Learning. In *Proc. of ACL*.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2012. Structured Learning with Constrained Conditional Models. *Machine Learning*, 88(3):399–431.

Hsin-Hsi Chen, Wen-Cheng Lin, Changhua Yang, and Wei-Hao Lin. 2006. Translating-Transliterating Named Entities for Multilingual Information Access. *Journal of the Association for Information Science and Technology*, 57(5).

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proc. of EMNLP*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 Shared Task—Morphological Reinflection. In *Proc. of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*.

Ryan Cotterell, Ekaterina Vylomova, Huda Khayrallah, Christo Kirov, and David Yarowsky. 2017. Paradigm Completion for Derivational Morphology. In *Proc. of EMNLP*.

Kareem Darwish. 2013. Named Entity Recognition using Cross-lingual Resources: Arabic as an Example. In *Proc. of ACL*.

Xiangyu Duan, Rafael E Banchs, Min Zhang, Haizhou Li, and A Kumaran, editors. 2015. *Proc. of the Fifth Named Entity Workshop*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological Inflection Generation Using Character Sequence to Sequence Learning. In *Proc. of NAACL-HLT*.

Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2015. Neural Network Transduction Models in Transliteration Generation. In *Proc. of the Fifth Named Entity Workshop*.

Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2016. Target-Bidirectional Neural Models for Machine Transliteration. In *Proc. of the Sixth Named Entity Workshop*.

Isao Goto, Naoto Kato, Noriyoshi Uratani, and Terumasa Ehara. 2003. Transliteration Considering Context Information based on the Maximum Entropy Method. In *Proc. of MT-Summit IX*, volume 125132.

Li Haizhou, Zhang Min, and Su Jian. 2004. A Joint Source-Channel Model for Machine Transliteration. In *Proc. of ACL*.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name Translation in Statistical Machine Translation - Learning When to Transliterate. *Proc. of ACL-HLT*.

Ann Irvine, Chris Callison-Burch, and Alexandre Klementiev. 2010. Transliterating from All Languages. In *Proc. of AMTA*.

Kil Soon Jeong, Sung-Hyon Myaeng, Jae Sung Lee, and K-S Choi. 1999. Automatic Identification and Back-Transliteration of Foreign Words for Information Retrieval. *Information Processing & Management*, 35(4):523–540.

Sittichai Jiampojamarn, Aditya Bhargava, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak. 2009. DirecTL: A Language-Independent Approach to Transliteration. In *Proc. of the 2009 Named Entities Workshop: Shared Task on Transliteration*.

Sittichai Jiampojamarn, Kenneth Dwyer, Shane Bergsma, Aditya Bhargava, Qing Dou, Mi-Young Kim, and Grzegorz Kondrak. 2010. Transliteration Generation and Mining with Limited Training Resources. In *Proc. of the 2010 Named Entities Workshop*.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion. In *Proc. of NAACL*.

Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. 2011. Machine Transliteration Survey. *ACM Computing Surveys*.

Sungchul Kim, Kristina Toutanova, and Hwanjo Yu. 2012. Multilingual Named Entity Recognition using Parallel Data and Metadata from Wikipedia. In *Proc. of ACL*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*.

Alex Klementiev and Dan Roth. 2008. Named Entity Transliteration and Discovery in Multilingual Corpora. In *Learning Machine Translation*. MIT Press.

Kevin Knight and Jonathan Graehl. 1998. Machine Transliteration. volume 24, pages 599–612. MIT Press.

Anoop Kunchukuttan, Mitesh Khapra, Gurneet Singh, and Pushpak Bhattacharyya. 2018. Leveraging Orthographic Similarity for Multilingual Neural Transliteration. In *Transactions of the Association for Computational Linguistics*, volume 6.

Ying Lin, Xiaoman Pan, Aliya Deri, Heng Ji, and Kevin Knight. 2016. Leveraging Entity Linking and Related Language Projection to Improve Name Transliteration. In *Proc. of the Sixth Named Entity Workshop*.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and Copy: UZH at SIGMORPHON 2017 Shared Task for Morphological Reinflection. In *Proc. of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*.

David Matthews. 2007. Machine Transliteration of Proper Names. *Master's Thesis, University of Edinburgh, Edinburgh, United Kingdom*.

Paul McNamee, James Mayfield, Dawn Lawrie, Douglas W Oard, and David S Doermann. 2011. Cross-Language Entity Linking. In *Proc. of IJCNLP*.

Yuval Merhav and Stephen Ash. 2018. Design Challenges in Named Entity Transliteration. In *Proc. of COLING*.

Garrett Nicolai, Bradley Hauer, Mohammad Salameh, Adam St Arnaud, Ying Xu, Lei Yao, and Grzegorz Kondrak. 2015. Multiple System Combination for Transliteration. In *Proc. of the Fifth Named Entity Workshop*.

Jeff Pasternack and Dan Roth. 2009. Learning Better Transliterations. In *Proc. of CIKM*.

Sujith Ravi and Kevin Knight. 2009. Learning Phoneme Mappings for Transliteration without Parallel Data. In *Proc. of NAACL-HLT*.

Richard Sproat, Tao Tao, and ChengXiang Zhai. 2006. Named Entity Transliteration with Comparable Corpora. In *Proc. of COLING-ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. of NIPS*.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual Wikification Using Multilingual Embeddings. In *Proc. of NAACL*.

Chen-Tse Tsai and Dan Roth. 2018. Learning Better Name Translation for Cross-Lingual Wikification. In *Proc. of AAAI*.

Shyam Upadhyay, Nitish Gupta, and Dan Roth. 2018. Joint Multilingual Supervision for Cross-lingual Entity Linking. In *Proc. of EMNLP*.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of Proper Names in Cross-lingual Information Retrieval. In *Proc. of the Workshop on Multilingual and Mixed-Language Named Entity Recognition*.

Han Wang, Jin Guang Zheng, Xiaogang Ma, Peter Fox, and Heng Ji. 2015. Language and Domain Independent Entity Linking with Quantified Collective Validation. In *Proc. of EMNLP*.

NORMA: Neighborhood Sensitive Maps for Multilingual Word Embeddings

Ndapa Nakashole
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
nnakashole@eng.ucsd.edu

Abstract

Inducing multilingual word embeddings by learning a linear map between embedding spaces of different languages achieves remarkable accuracy on related languages. However, accuracy drops substantially when translating between distant languages. Given that languages exhibit differences in vocabulary, grammar, written form, or syntax, one would expect that embedding spaces of different languages have different structures especially for distant languages. With the goal of capturing such differences, we propose a method for learning neighborhood sensitive maps, NORMA. Our experiments show that NORMA outperforms current state-of-the-art methods for word translation between distant languages.

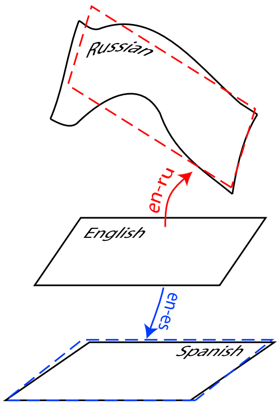

Figure 1: **Bottom:** By learning a linear map between embedding spaces of related languages, e.g., en-es, current methods achieve high accuracy on word translation. **Top:** For distant language pairs, e.g., en-ru, where differences are larger, word translation accuracy substantially degrades.

1 Introduction

The success of monolingual word embeddings has sparked interest in multilingual word embeddings. The goal is to learn word vectors where similar words have similar vector representations regardless of their language. Multilingual word embeddings are playing an increasingly prominent role in machine translation (Zou et al., 2013; Lample et al., 2018; Artetxe et al., 2018b). In addition, they are a promising avenue for cross-lingual model transfer (Guo et al., 2015; Täckström et al., 2012).

A prominent approach to learning multilingual word embeddings is to induce a mapping function between embedding spaces of different languages. However, there is a key assumption behind learning such a mapping function: that the embedding spaces of different languages exhibit similar structures (Mikolov et al., 2013a). Evidence that this assumption holds has mostly been through extrinsic evaluation metrics such as word translation accuracy. A notable exception is (Mikolov et al., 2013a), who showed empirical evidence on animals and numbers. Embeddings corresponding to a few numbers and animals in English and Spanish were projected down to two dimensions using PCA, and then manually rotated to accentuate similarity. Despite showing only these two concepts for two related languages, this work concluded that embedding spaces of different languages exhibit similar geometric arrangements. Additionally, work in this line of inquiry has continued to develop methods based on this assumption (Artetxe et al., 2018a; Conneau et al., 2018). Given that languages differ along dimensions such as vocabulary, grammar, written form, and syntax, one would expect that embedding spaces of different languages exhibit different structures. Indeed, recent work showed that assumptions of isomorphism and linearity do not hold (Søgaard et al., 2018; Nakashole and Flauger, 2018)

While these assumptions do not substantially affect accuracy when translating between related

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 512–522
Brussels, Belgium, October 31 - November 4, 2018.

languages, this is not the case for distant languages, see Figure 1. There is no established quantitative metric for measuring distances between languages. Language trees trace the evolution of languages but do not provide similarity scores. (Chiswick and Miller, 2005) learned similarity scores of 43 different languages to English by measuring how well Americans could learn a given language in a fixed period of time. Low scores on a standardized proficiency test were taken to indicate a large distance between the language and English. According to their scores, Japanese and Chinese are the most distant from English, Russian has a middle score, and French, Portuguese, Dutch, as expected, have some of the highest scores.

Additionally, linguists and psychologists have long studied the question of how language affects the way we think (Birner, 1999; Boroditsky, 2011). This influence would arise due to different languages organizing concepts differently.

We would like to model some aspects of the structural differences of languages when learning mapping functions between embedding spaces. To this end, we propose to learn neighborhood sensitive maps. We can, in principle, achieve neighborhood sensitive maps by training non-linear functions. However, training non-linear functions, in particular deep neural networks for this problem is difficult to optimize for this zero-shot (Lazaridou et al., 2015) learning problem, as we show in our experiments. Prior work alludes to similar observations(Mikolov et al., 2013a). For example, (Conneau et al., 2018) found that using non-linear mapping functions made training unstable[1].

In summary, our contributions are as follows:

- We propose a method for learning neighborhood sensitive maps, NORMA, which learns a single mapping function but in a departure from prior work, it *discovers neighborhoods*. NORMA avoids learning multiple mapping functions, thus enabling parameter sharing among neighborhoods. This is a more efficient use of training data than if we were to train multiple mapping functions for different neighborhoods as is done in (Zou et al., 2013).
- The neighborhoods are learned jointly while learning to translate, and we show that they are *interpretable*.
- Our experiments show that for word translation between distant languages, NORMA substantially outperforms methods that achieve the best performance when translating between related languages.
- Additionally, in the related language setting, we show that on rare words NORMA substantially outperforms state-of-the-art methods.

2 Related Work

The common approach to learning cross embedding space mapping functions is: first monolingual word embeddings for each language are trained independently; and second, a mapping function is learned, using supervised or unsupervised methods. The resulting mapping function enables translating words from the source to the target language.

Map Induction Methods. The earliest and simplest approach is to use a regularized least squares loss to induce a linear map $\mathbf{M}$ as follows: $\hat{\mathbf{M}} = \arg\min_{\mathbf{M}} ||\mathbf{MX} - \mathbf{Y}||_F + \lambda||\mathbf{M}||$, here $\mathbf{X}$ and $\mathbf{Y}$ are matrices that contain word embedding vectors for the source and target language (Mikolov et al., 2013a; Dinu et al., 2014; Vulic and Korhonen, 2016). Improved results were obtained by imposing an orthogonality constraint on $\mathbf{M}$ (Xing et al., 2015; Smith et al., 2017). Another loss function used in prior work is the max-margin loss, which has been shown to significantly outperform the least squares loss (Lazaridou et al., 2015; Nakashole and Flauger, 2017).

Another approach is to use canonical correlation analysis (CCA) to map two languages to a shared embedding space (Haghighi et al., 2008; Faruqui and Dyer, 2014; Lu et al., 2015; Ammar et al., 2016).

Most of the prior methods can be characterized as a series of linear transformations. In particular, (Artetxe et al., 2018a) propose a framework to differentiate prior methods in terms of which transformations they perform: embedding normalization, whitening, re-weighting, de-whitening, and dimensionality reduction.

Work on phrase translation proposed to induce many local maps that are individually trained (Zhao et al., 2015) on local neighborhoods. In

---

[1] https://openreview.net/forum?id=H196sainb

contrast, our approach trains a single function while taking into account neighborhood sensitivity. Our underlying motivation of neighborhood sensitivity is similar in spirit to the use of locally linear embeddings for nonlinear dimensionality reduction (Roweis and Saul, 2000).

Forms of Supervision. The methods we have described so far fall under supervised learning. In the supervised setting, a seed dictionary (5k word pairs is a typical size) is used to induce the mapping function. In (Artetxe et al., 2017) a semi-supervised approach is explored, whereby the method alternates between learning the map and generating an increasingly large dictionary. Completely unsupervised methods have recently been proposed using adversarial training (Barone, 2016; Zhang et al., 2017; Conneau et al., 2018). However, the underlying methods for learning the mapping function are similar to prior work such as (Xing et al., 2015). The limitations and strengths of unsupervised methods are detailed in (Søgaard et al., 2018)

Although in our our experiments we work in the supervised setting, NORMA can work with any form of supervision.

Translation Retrieval Methods. The most commonly used way to obtain a translation t of a source language word s is nearest neighbor retrieval, given by: $t = \arg\max_t \cos(\mathbf{M}x_s, y_t)$. Alternative retrieval methods have been proposed, such as the inverted nearest neighbor retrieval(Dinu et al., 2014), inverted softmax (Smith et al., 2017) and Cross-Domain Similarity Local Scaling (CSLS) (Conneau et al., 2018). Since we are interested in evaluating the quality of mapping functions, our experiments use standard nearest neighbor retrieval for all methods.

3 Local Maps in Embedding Space

Is it useful for maps to be neighborhood sensitive? To study this question we carried out experiments comparing performance of neighborhood-specific maps to global maps. A thorough analysis of this kind was carried out in our prior work (Nakashole and Flauger, 2018)

We created neighborhoods by first selecting the embeddings of a few words associated with specific topics such as diseases, or cities. We then added all nearby words, which are words whose cosine similarity to any of the selected words is $>=$0.5 [2]. We used three language pairs for local vs global map translation experiments: English to German, English to Portuguese, and English to Swedish. The neighborhoods and their train/test splits are:
$en - de$: medication(3,415/500), cities(2,083/500), and animals(990/500);
$en - pt$: diseases(1,670/300), chemicals(1,279/300), and names(1,986/300);
$en - sv$: flowers(1,537/200), insects(1,271/200), and names(1,416/200). The training and test data was obtained from subsets of Facebook AI MUSE lexicons[3]

For each of the neighborhoods, we evaluated translation accuracy both when using a locally trained map and when using a globally trained map. The difference is that the locally trained map is only trained using training data from the neighborhood, whereas the global map is trained using training data from the neighborhood but also from all other neighborhoods and more (˜10000 word pairs). That is, the training data for global maps is a superset of the local training data.

We trained all maps using linear transformations. As we will show in our experiments, optimizing neural network mapping functions for this problem fails. This is a similar observation to prior work (Mikolov et al., 2013a; Conneau et al., 2018)[1]. More details on models and experimental settings are described in Sections 4 and 5.

Figure 2 shows that for various neighborhoods, translation accuracy is higher when we train neighborhood-specific maps than one single global map. These results are similar to (Zou et al., 2013) who then trained many local maps. While we could also proceed to train many local maps, this requires identifying optimal neighborhoods. It also requires gathering sufficient training data for each of the neighborhoods independently. In our proposed method, NORMA, we avoid learning multiple maps, creating a single map, while modeling neighborhood information and promoting parameter sharing.

Overall, the results in Figure 2 are an indicator neighborhood sensitivity in maps is useful. This would particularly be useful for distant languages

---

[2] We found a 0.5 cutoff to be a good compromise between neighborhood purity, and size. However, our final method (Section 4) on which all our comparison experiments were based, automatically discovers neighborhoods based on ideas from sparse coding.

[3] https://github.com/facebookresearch/MUSE

where a single global map that is linear might not suffice since the underlying embedding structure for distant languages might differ more than those of related languages as depicted in Figure 1.

4 Model

In this section we introduce our model for learning neighborhood sensitive maps, NORMA. Our approach jointly discovers neighborhoods while learning to translate.

4.1 Reconstructive Neighborhood Discovery

Inspired by work on sparse coding (Lee et al., 2007), we discover neighborhoods by learning a reconstructive dictionary. We would like to learn a dictionary of neighborhoods on the source language side. To learn this dictionary, we set up a reconstruction objective, where for any given word embedding $x_i \in \mathbb{R}^d$, where d is the dimensionality of the word embeddings, we want to reconstruct x_i using a linear combination of K neighborhoods. Let $\mathbf{D} \in \mathbb{R}^{K \times d}$ be the neighborhood matrix, each row of $\mathbf{D}$ represents a d-dimensional vector which can be interpreted as representing the center of the neighborhood. Let $\mathbf{X} \in \mathbb{R}^{N \times d}$ be a set of N embedding vectors corresponding to words in the source language vocabulary[4]. We can learn a reconstructive dictionary of K neighborhoods with the following objective:

$$\mathbf{D}, \mathbf{V} = \arg\min_{\mathbf{D},\mathbf{V}} ||\mathbf{X} - \mathbf{VD}||_2^2 \qquad (1)$$

$\mathbf{D} \in \mathbb{R}^{K \times d}$ is the learned dictionary of neighborhoods, $K > d$ and thus the dictionary is overcomplete; $\mathbf{V} \in \mathbb{R}^{N \times K}$ are the learned neighborhood membership weights for $\mathbf{X}$. While we use the squared loss, other loss functions can be used (Lee et al., 2007). To encourage neighborhoods to be different from each other, one can impose an orthogonality constraint : $||\mathbf{DD}^T - \mathbf{I}||$ where $\mathbf{I}$ is the identity matrix. The reconstruction error with an orthogonality penalty is:

$$R(\theta) = ||\mathbf{X} - \mathbf{VD}||_2^2 + \lambda ||\mathbf{DD}^T - \mathbf{I}|| \qquad (2)$$

Where λ is a hyperparameter which controls the contribution of the orthogonality constraint to the reconstruction error.

4.2 Joint Neighborhood Discovery and Translation

Our approach ties neighborhood discovery to the word translation task. First, we obtain neighborhood 'factorized' representations by multiplying the input vector $\mathbf{X}$ by the dictionary of neighborhoods:

$$\mathbf{X}_{\mathcal{N}} = \mathbf{XD}^T,$$

where $\mathbf{X}_{\mathcal{N}} \in \mathbb{R}^{N \times K}$. Here again N refers to words in the source language vocabulary, English in the case of $en - de$ translation. And K is the number of neighborhoods.

Second, we obtain an intermediate representation of the input, which contains both the original input $\mathbf{X}$ and the neighborhood 'factorized' representations of the input $\mathbf{X}_{\mathcal{N}}$, through vector concatenation as follows:

$$\mathbf{X}_{\mathcal{I}} = [\mathbf{X}_{\mathcal{N}}; \mathbf{X}],$$

where $\mathbf{X}_{\mathcal{I}} \in \mathbb{R}^{N \times (K+d)}$.

To get the final representation of the input, we project $\mathbf{X}_{\mathcal{I}}$ into a low-dimensional vector of the same size as the original input:

$$\mathbf{X}_{\mathcal{F}} = \mathbf{X}_{\mathcal{I}} \mathbf{W}_f,$$

where $\mathbf{W}_f \in \mathbb{R}^{(K+d) \times d}$ is a set of learned parameters. And $\mathbf{X}_{\mathcal{F}} \in \mathbb{R}^{N \times d}$ is the resulting final representation.

We use these neighborhood sensitive representation $\mathbf{X}_{\mathcal{F}}$ as the input for learning the mapping function $\mathbf{W}$, instead of the original $\mathbf{X}$. We explore different ways for learning the mapping $\mathbf{W}$: first a linear mapping, and second, a single layer neural network with a leaky rectified linear unit (leaky ReLU[5]) non-linearity and a highway layer (Srivastava et al., 2015). As we will show in our experiments, training neural networks with more layers fails on this zero-shot learning problem.

For the linear map, the translation $\hat{y}_i$ is given by:

$$\hat{y}_i^{linear} = \mathbf{W} x_{\mathcal{F}_i} \qquad (3)$$

where $x_{\mathcal{F}_i} \in \mathbf{X}_{\mathcal{F}}$ is the neighborhood sensitive representation of x_i.

For the neural network map, using a single layer neural network, and a highway layer, the transla-

---

[4] Since the vocabulary size can be very large, in our experiments, we work in batches of N=50

[5] It outperformed other non-linearities such as $tanh$ in our initial experiments.

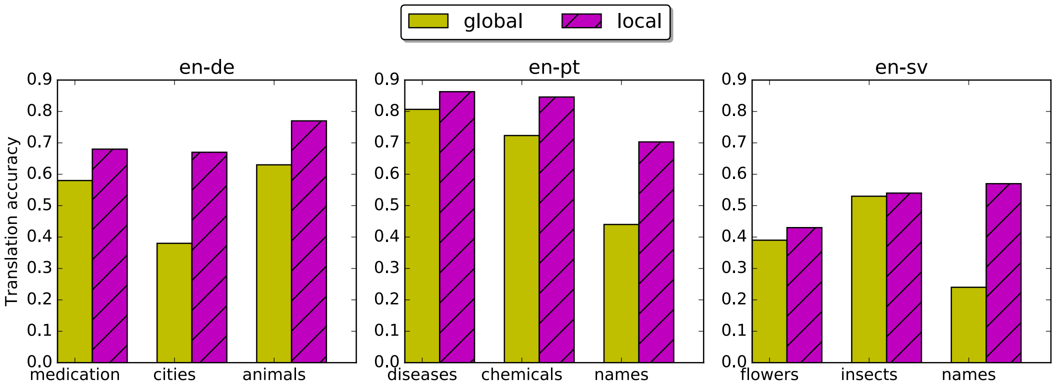

Figure 2: Accuracy of globally vs locally trained mapping functions for various neighborhoods on $en-de$, $en-pt$, and $en-sv$ translation.

tion $\hat{y}_i$ is given by:

$$h_i = \sigma_1(x_{\mathcal{F}_i}\mathbf{W})$$
$$t_i = \sigma_2(x_{\mathcal{F}_i}\mathbf{W^t})$$
$$\hat{y}_i^{nn} = t_i \times h_i + (1.0 - t_i) \times x_{\mathcal{F}_i} \qquad (4)$$

where σ_1 is a non-linearity. We use a leaky-ReLU non-linearity. σ_2 is the sigmoid function. $\mathbf{W^t}$ is another set of parameters in addition to $\mathbf{W}$.

4.3 Objective Function

We use the max-margin loss function to learn the parameters of the model:

$$L(\theta) = \sum_{i=1}^{m}\sum_{j\neq i}^{k} \max\Big(0, \gamma + d(y_i, \hat{y}_i^g) - d(y_j, \hat{y}_i^g)\Big), \qquad (5)$$

Where y_i is the true label; $\hat{y}_i^g$ is the prediction, which is either $\hat{y}_i^{linear}$ or $\hat{y}_i^{nn}$. The goal of the max-margin loss function is to rank correct training data pairs (x_i, y_i) higher than incorrect pairs (x_i, y_j) with a margin of at least γ. The margin γ is a hyper-parameter and the incorrect labels, y_j are selected randomly such that $j \neq i$. k is the number of incorrect examples per training instance, and $d(x, y) = (x-y)^2$ is the distance measure.

The joint neighborhood discovery and word translation objective is given by:

$$J(\theta) = L(\theta) + R(\theta) \qquad (6)$$

The neighborhood discovery part of the objective, $R(\theta)$, does not depend on availability of supervised data and only requires monolingual data on the source language side. Thus, we can discover neighborhoods in an unsupervised manner on a large set of monolingual word embeddings, then initialize using this pre-trained D which is then jointly optimized with the translation part of the objective $L(\theta)$. Importantly, this also means that our method can work with unsupervised methods for learning mapping functions such as those using adversarial training (Barone, 2016; Conneau et al., 2018).

5 Experimental Evaluation

In this section, we study the following questions: How does NORMA compare to state-of-the-art methods for learning mapping functions between embedding spaces of different languages? We study this question in three settings: when translating between distant languages, when translating between related languages, and lastly, when translating between related languages but on rare words. Additionally, we ask the following question: are the neighborhoods learned by NORMA meaningful?

To study these questions, we carried out experiments on word translation from English to two distant languages, a Slavic language (Russian), and a Sino-Tibetan language (Chinese). In addition, we carried out experiments on word translation between related languages (English, French, German and Portuguese).

Data and Experimental Setup. The Facebook AI MUSE[3] project (Conneau et al., 2018) provides train/test data for bilingual dictionaries of various language pairs, we use this data in our experiments. The MUSE dictionaries consist of

Method	Slavic & Sino-Tibetan				
	en-ru	en-zh	en-de	en-es	en-fr
NORMA-Linear	**50.33**	**43.27**	68.50	77.47	76.10
NORMA-Highway-NN	49.27	33.10	67.33	77.65	75.50
1 layer-NN	49.13	30.66	66.80	77.60	75.53
2 layer-NN	0	0	0	0	0
1 layer-Highway-NN	49.50	30.91	67.00	77.50	75.60
2 layer-Highway-NN	0	0	0	0	0
Artetxe et al . 2018	47.93	20.4	70.13	**79.6**	**79.30**
Conneau et al. 2018	37.30	30.90	**71.30**	79.10	78.10
Smith et al. 2017	46.33	39.60	69.20	78.80	78.13
Xing et al. 2015	44.50	41.0	67.07	77.33	75.47
Lazaridou et al. 2015	48.27	29.60	68.20	77.60	75.86
Faruqui and Dyer (2014)	35.47	32.20	55.67	72.33	69.27
Mikolov et al. 2013	42.47	19.80	60.07	74.20	71.60

Table 1: Precision at 1 comparison of NORMA to previously proposed mapping functions. We used FAIR/MUSE word translation lexicons train/test splits.

	en-ru	en-zh	en-de	en-es	en-fr
NOUN	42% /**55.1**	42% /42.1	39% /**74.6**	40% /**82.3**	42% /**80.0**
VERB	41% /47.3	39% /**47.6**	38% /64.4	40% /71.6	41% /70.0
ADJECTIVE	10% /34.4	11% /38.7	10% /56.1	9% /76.9	10% /71.3

Table 2: Part-of-Speech (POS) distributions of the MUSE test sets. Listed are the top 3 parts of speech, which account for ˜90% of the test data for all language pairs. X% /Y means the POS tag makes up X% of the test set, with accuracy Y.

5,000/1,500 word pairs for train/test data. Unless specified, we use the train/test split provided by MUSE. Development sets: the MUSE dictionaries that we used are very large. They contain over 100,000 entries for most language pairs, we tuned our models on data that was not part of the train and test sets.

We obtained pre-trained word embeddings from FastText (Bojanowski et al., 2017). In Equation 2, we did not find it helpful to encourage neighborhoods to be different, thus we set $\lambda = 0$. We set the margin γ in Equation 5 to be $\gamma = 0.4$. For the dictionary of neighborhoods D in Equation 1, we set the number of neighborhoods $K = 2,000$[6]. We use $N = 50$ batch size. We estimate model parameters using stochastic gradient descent.

Methods Under Comparison. We compare variations of NORMA to several previously proposed methods for generating mapping functions. The methods compared are: (Artetxe et al., 2018a; Conneau et al., 2018; Smith et al., 2017; Xing et al., 2015; Lazaridou et al., 2015; Faruqui and Dyer, 2014; Mikolov et al., 2013a). More detailed descriptions of these prior methods can be found in the related work section.

Our primary goal is to evaluate the quality of maps produced. While a number of prior work proposed various approaches for retrieval, which have been shown to improve accuracy by a few points, we compare all methods using the same retrieval method, nearest neighbor. Thus, for (Conneau et al., 2018), we report the results for the variant of their method called: *adv - Refine - NN*.

---

[6] We carried out experiments using different neighborhood sizes, and consistently found $K \approx 2000$ to outperform other choices.

5.1 English to Slavic and Sino Tibetan

State-of-the-art methods have mostly focused word translation evaluation on English to Latin languages or other nearby languages. (Artetxe et al., 2018a) performed experiments on en-es, en-de, en-it and en-fi, where concepts might still be organized in a relatively similar way. In (Conneau et al., 2018), the adversarial training method proposed was evaluated on Chinese, Russian, and Esperanto, but thorough comparison experiments to prior work on word translation were only performed on English to Italian.

We carried out en-ru and en-zh comparison experiments, and present the results in the second and third columns of Table 1. The two state-of-of-the art methods (Artetxe et al., 2018a) and (Conneau et al., 2018) are significantly outperformed by NorMA-Linear. On English to Russian, NorMA-Linear achieves 50.33 precision 1, outperforming both (Artetxe et al., 2018a) (Conneau et al., 2018), as well as other methods. On English to Chinese, NorMA-Linear achieves 43.37 precision 1, again ahead of other methods. The best performing variant of our method is NorMA-Linear. The neural networks with more than a single layer prove difficult to optimize for this problem, and produce accuracy of 0. This could be because the problem of cross-embedding space mapping is a zero-shot learning problem, which is much more difficult to train than a supervised problem, the setting in which deep learning methods have thrived so far.

5.2 English to Related Languages

We show experiments on English to related languages in the last three columns of Table 1. On these languages, indeed the most recently proposed methods (Artetxe et al., 2018a; Conneau et al., 2018) produce the best performing maps. However, NorMA-Linear is only 2-3 points behind these methods. This in contrast to English to Chinese where both (Artetxe et al., 2018a) and (Conneau et al., 2018) are behind NorMA - Linear, by more than 10 points.

A promising line of future work is to get NorMA-Linear to bridge the 2-3 point gap on related languages by exploring a best of both worlds approach, combining neighborhood sensitivity with the methods that achieve superior performance on nearby languages.

	en-pt	
	RARE	MUSE
NorMA-Linear	**57.67**	72.60
NorMA-Highway-NN	49.33	71.73
1 layer-NN	48.67	72.13
1 layer-Highway-NN	49.33	72.10
Artetxe et al . 2018	47.00	**77.73**
Lazaridou et al 2015	48.00	72.27

Table 3: Performance for en-pt on rare words (RARE), and the en-pt MUSE dataset, which as shown in Figure 3 contains a lot of frequent words.

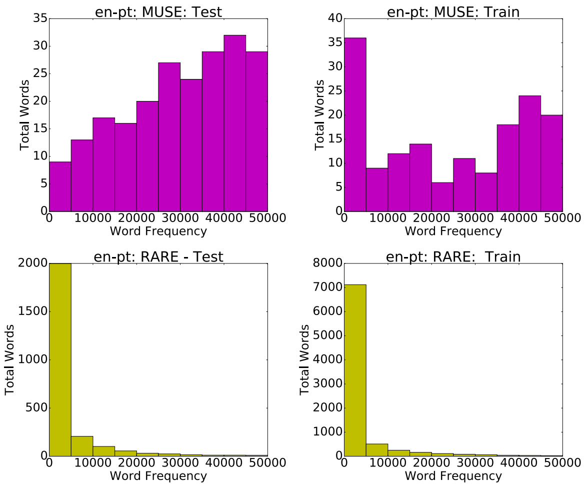

Figure 3: Top: Frequency distribution of MUSE dictionary test and train sets for en-pt. Bottom: Frequency distribution of the RARE words dataset.

5.3 Accuracy by Part-of-Speech

We assigned each word its majority part-of-speech by tagging the ClueWeb[7] corpus, which contains over 500 million webpages. We then evaluated translation precision of NorMA-Linear stratified by part-of-speech. The results are shown in Table 5 We found that, nouns and verbs make up about 80 percent of the MUSE test dictionaries, followed by adjectives (˜10%). We found that while nouns and verbs make up a large chunk of the test data, nouns are translated with much higher accuracy than verbs, except for English to Chinese. This finding will serve as a guide for future improvements to our method.

5.4 English to Languages: Rare Words

We analyzed the frequency distribution of the MUSE dictionaries. To get word frequency infor-

---

[7] `https://www.lemurproject.org/clueweb09.php/`

Neighborhood			
51	134	162	7
drugs zonisamide cocaine ritalin hospitalized pheniprazine overdose disorientation focusyn alfaxalone	criminally judicature prosecutory derogation restitutionary derogative jailable extradition sodomy crimes	chuanyao chuanyan zhiang thanong qiangbing pengpeng nguyan yuning liheng thanong	khoisan bantu sepedi otjiherero ndebeles hereros otjinene shona hutu witotoan

Table 4: Sample neighborhoods discovered by NORMA during en-de translation: 51 appears to represent drugs, 132: justice and crime; 162: Asian names, 7 : African names.

mation, we processed documents in the ClueWeb[7] corpus and recorded word occurrence frequency. We discovered that the MUSE dictionaries contain a lot of frequent words. The top half of Figure 3 shows frequency counts of the en-pt MUSE test dictionary. For readability we only show bins up to occurrence frequency of 50,000. We see that only about 50/1500 in the MUSE en-pt test data are infrequent, the rest are frequent words, occurring more than 10,000 times in the ClueWeb corpus.

We therefore created another test set for en-pt from the rest of the MUSE data which is not part of the train or test data, with the goal of creating a train/test of rare words. The bottom half of Figure 3 is a plot of frequency counts of train and test data for these rare words.

We then compared variations of NORMA to the best performing method on English to related languages, which is (Artetxe et al., 2018a). The comparison was done both on the regular MUSE test dataset for en-pt and the rare word dataset for en-pt. Since our method uses a max-margin loss much like (Lazaridou et al., 2015), we also compare to (Lazaridou et al., 2015).

Table 3 shows that NORMA-Linear outperforms (Artetxe et al., 2018a) by over 10 points on the RARE words dataset. On the regular MUSE dictionary, (Artetxe et al., 2018a) is ahead by about 5 points. On RARE, (Lazaridou et al., 2015) is behind NORMA-Linear by 9 points, whereas on the MUSE dictionary performance of (Lazaridou et al., 2015) and NORMA-Linear is about the same.

5.5 Neighborhood Interpretability

NORMA jointly discovers neighborhoods while learning to translate words. We now ask if the discovered neighborhoods semantically make sense. We can answer this question since each neighborhood vector can be seen as a "center" vector representing the words in the neighborhood. Thus we can consider words whose cosine similarity to the neighborhood vector is greater than some threshold, to be members of that neighborhood. As we mentioned, we found that setting the total number of neighborhoods to be discovered to $K = 2,000$ provided the best results. Of these 2,000 we show some of them in Table 4 obtained when training $en - de$. For each neighborhood, we show 10 words that appear among the top 100 words of that neighborhood. It can be seen that the neighborhoods represent some kind of "topics". For example, neighborhood number 51 appears to represent drugs, and drug-related concepts; number 132 contains justice and crime-related concepts; number 162 contains mostly Asian concepts and names, number 7 contains mostly African and names. We can see that the granularity of neighborhoods and their specificity varies.

6 Conclusions

We propose neighborhood sensitive maps for learning multilingual word embeddings, NORMA. Our method is motivated by the fact that languages differ along dimensions such as vocabulary, grammar, written form, and syntax, and therefore one would expect that embedding spaces of different languages exhibit different structures especially for distant languages.

Our method jointly discovers neighborhoods while learning to translate words. Experimental evaluation showed that NORMA substantially outperforms state-of-the-art (SOTA) methods on distant languages, while only being a few points behind on related languages. A promising line of future work is to explore a best of both worlds approach, combining neighborhood sensitivity with the methods that achieve superior performance on nearby languages.

Acknowledgments

We thank Raphael Flauger for useful discussions, and the anonymous reviewers for their constructive comments.

References

Hanan Aldarmaki, Mahesh Mohan, and Mona Diab. 2018. Unsupervised word mapping using structural similarities in monolingual embeddings. *Transactions of the Association of Computational Linguistics*, 6:185–196.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR*, abs/1602.01925.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 451–462.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *AAAI*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018b. Unsupervised neural machine translation. In *ICLR*.

Antonio Valerio Miceli Barone. 2016. Towards crosslingual distributed representations without parallel text trained with adversarial autoencoders. In *1st Workshop on Representation Learning for NLP*.

Betty Birner. 1999. Does the language i speak influence the way i think?.

Phil Blunsom and Karl Moritz Hermann. 2014. Multilingual distributed representations without word alignment. In *ICLR*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*.

Lera Boroditsky. 2011. How language shapes thought. *Scientific American*, 304(2):62–65.

A. P. Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *NIPS*, pages 1853–1861.

Barry R Chiswick and Paul W Miller. 2005. Linguistic distance: A quantitative measure of the distance between english and other languages. *Journal of Multilingual and Multicultural Development*, 26(1):1–11.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *EACL*, pages 462–471.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *ICML*, pages 748–756.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *NAACL*, pages 1386–1390.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *ACL*, pages 1234–1244.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*, pages 771–779.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *COLING*, pages 1459–1474.

Tomáš Kočiskỳ, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. *arXiv preprint arXiv:1405.0947*.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *ACL Workshop on Unsupervised Lexical Acquisition*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *ICLR*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *ACL*, pages 270–280.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. 2007. Efficient sparse coding algorithms. In *NIPS*, pages 801–808.

Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong. 2014. Learning small-size dnn with output-distribution-based criteria. In *INTERSPEECH*, pages 1910–1914.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. 2008. Predicting human brain activity associated with the meanings of nouns. *science*, 320(5880):1191–1195.

Ndapa Nakashole and Raphael Flauger. 2018. Characterizing departures from linearity in word translation. In *ACL*.

Ndapandula Nakashole and Raphael Flauger. 2017. Knowledge distillation for bilingual dictionary induction. In *EMNLP*, pages 2487–2496.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *ACL*.

Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning cross-lingual word embeddings via matrix co-factorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 567–572.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *ICLR*.

Anders Søgaard, Zeljko Agic, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual NLP. In *ACL*, pages 1713–1722.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Oscar Täckström, Ryan T. McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *NAACL*, pages 477–487.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res. (JAIR)*, 37:141–188.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *ACL*.

Ivan Vulic and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. ACL.

Ivan Vulic and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *ACL*, pages 719–725.

Derry Tanti Wijaya, Brendan Callahan, John Hewitt, Jie Gao, Xiao Ling, Marianna Apidianaki, and Chris Callison-Burch. 2017. Learning translations via matrix completion. In *EMNLP*, pages 1452–1463.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *HLT-NAACL*, pages 1006–1011.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1959–1970.

Kai Zhao, Hany Hassan, and Michael Auli. 2015. Learning translation models from monolingual continuous representations. In *NAACL*, pages 1527–1536.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.

Adaptive Multi-pass Decoder for Neural Machine Translation

Xinwei Geng **Xiaocheng Feng** **Bing Qin**[*] **Ting Liu**
Harbin Institute of Technology, China
`{xwgeng, xcfeng, qinb, tliu}@ir.hit.edu.cn`

Abstract

Although end-to-end neural machine translation (NMT) has achieved remarkable progress in the recent years, the idea of adopting multi-pass decoding mechanism into conventional NMT is not well explored. In this paper, we propose a novel architecture called *adaptive multi-pass decoder*, which introduces a flexible multi-pass polishing mechanism to extend the capacity of NMT via reinforcement learning. More specifically, we adopt an extra policy network to automatically choose a suitable and effective number of decoding passes, according to the complexity of source sentences and the quality of the generated translations. Extensive experiments on Chinese-English translation demonstrate the effectiveness of our proposed adaptive multi-pass decoder upon the conventional NMT with a significant improvement about 1.55 BLEU.

Source	*héshìzhù xiānshēng dē wěirènqī wéi yī nián , yǐ pèihé qí wéi fángwěihuì wěiyuán dē rèngqī jièmǎn rìqī , qítā xīn wěirèngwěiyuān dē rèngqī zé wéi liǎngnían .*
Reference	*all appointments are for two years , except that of mr ho sai - chu 's which is for one year in order to tie in with the expiry date of his appointment as an ha member .*
1st-pass	*mr ho sai - chu 's UNK is a year - long term of two years with a term of two years as the term of his term of office of the ha .*
2nd-pass	*mr ho sai - chu 's UNK is a year - long term of two years with a term of two years to serve as the term of office of the ha .*
3rd-pass	*mr ho sai - chu 's UNK is a year - long term of two years with a term of two years to tie in with the expiry date of his term of office .*
4th-pass	*mr ho sai - chu has been serving as a member of authority for a term of two years with a term of two years .*

Table 1: Translation examples of more decoding passes with the proposed multi-pass decoder.

1 Introduction

In the past several years, end-to-end neural machine translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014) has attracted increasing attention from both academic and industry communities. Compared with conventional statistical machine translation (SMT) (Brown et al., 1993; Koehn et al., 2003), which needs to explicitly model latent structures, NMT adopts a unified *encoder-decoder* framework to directly transform a source sentence into a target sentence. Furthermore, the introduction of attention mechanism (Bahdanau et al., 2014) enhances the capability of NMT in capturing long-distance dependencies.

Recently, a number of authors have endeavored to adopt the polishing mechanism into NMT. Similar to human cognitive process for writing a good paper, their models first create a complete draft and then polish it based on global understanding of the whole draft (Niehues et al., 2016; Chatterjee et al., 2016; Zhou et al., 2017; Xia et al., 2017; Junczys Dowmunt and Grundkiewicz, 2017) . Moreover, Zhang et al. (2018) introduces a backward decoder to better exploit the right-to-left target-side contexts. Generally these methods employ two separate decoders to accomplish the polishing task.

Although these polishing mechanism-based approaches demonstrate their effectiveness with two-pass decoding, the idea of multi-pass decoding is not well explored for NMT. Motivated by it, we first propose a novel multi-pass decoder to perform the translation procedure with a fixed number of decoding passes, referred to as decoding depth. According to the preliminary results, just as expected, multi-pass decoding really benefit to most translations. However, in some cases, the more decoding passes perhaps lead to the poor translation. For example in Table 1, the 3rd-pass de-

---

[*] Corresponding author.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 523–532
Brussels, Belgium, October 31 - November 4, 2018.

coding achieves a better result compared to 1st- and 2nd-pass decoding. Nevertheless, a drastic decrease arises, when we perform the 4th-pass decoding. Therefore, it's necessary to introduce a flexible multi-pass decoding, which has the ability to adaptively choose the suitable decoding passes.

Towards above goal, we further propose a novel framework called *adaptive multi-pass decoder* to automatically choose a proper decoding depth using reinforcement learning. Our model considers multi-pass decoding as a sequential decision making process, where continuing decoding or halt is chosen at each step. An extra policy network is employed to learn to automatically choose to continue next pass decoding or halt via reinforcement learning. For the purpose of making accurate and effective choices, the policy network employs recurrent neural network to capture the complexity of source sentence as well as the difference between the consecutive generated translations. Extensive experiments on Chinese-English translation show the proposed adaptive multi-pass decoder is capable of choosing a suitable decoding depth and significantly improves translation performance over conventional NMT model.

2 Background

Given a source sentence $\mathbf{x} = x_1, \dots, x_m, \dots, x_M$ and a target sentence $\mathbf{y} = y_1, \dots, y_n, \dots, y_N$, end-to-end neural machine translation directly models translation probability word by word as a single, large neural network:

$$P(\mathbf{y}|\mathbf{x};\theta) = \prod_{n=1}^{N} P(\mathbf{y_n}|\mathbf{x}, \mathbf{y}_{<n};\theta) \tag{1}$$

where θ is a set of model parameters and $\mathbf{y}_{<n}$ denotes a partial translation. Prediction of n-th word is generally made in an *encoder-decoder* framework:

$$P(\mathbf{y}_n|\mathbf{x}, \mathbf{y}_{<n};\theta) = g(\mathbf{y}_{n-1}, \mathbf{s}_n, \mathbf{c}_n) \tag{2}$$

where $g(\cdot)$ is a non-linear function, $\mathbf{y}_{n-1}$ denotes the previously generated word, $\mathbf{s}_n$ is n-th decoding hidden state, and $\mathbf{c}_n$ is a context vector for generating n-th target word. The decoder state $\mathbf{s}_n$ is computed by RNNs as follows:

$$\mathbf{s}_n = f(\mathbf{s}_{n-1}, \mathbf{y}_{n-1}, \mathbf{c}_n) \tag{3}$$

where $f(\cdot)$ is an activation function. Actually it's found gated RNN alternatives such as LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014) often achieve better performance than vanilla ones. $\mathbf{c}_n$ is a dynamic vector that selectively summarizes certain parts of source sentence at each decoding step:

$$\mathbf{c}_n = \sum_{m=1}^{M} \alpha_{n,m}\mathbf{h}_m \tag{4}$$

where $\alpha_{m,n}$ measures how well $\mathbf{x}_m$ and $\mathbf{y}_n$ are aligned, calculated by attention model (Bahdanau et al., 2014; Luong et al., 2015), and $\mathbf{h}_m$ is the encoder hidden state of the m-th source word. For the purpose of capturing both forward and backward contexts, bidirectional RNN (Schuster and Paliwal, 1997) is often employed as the encoder which converts the source sentence into an annotation sequence $\mathbf{h} = \{\mathbf{h}_1, \dots, \mathbf{h}_m, \dots, \mathbf{h}_M\}$, where $\mathbf{h}_m = [\overrightarrow{\mathbf{h}}_m, \overleftarrow{\mathbf{h}}_m]$ captures information about m-th word with respect to the preceding and following words in the source sentence respectively.

Although the introduction of RNNs as a decoder has resulted in substantial improvements in terms of translation quality, simultaneously it imposes a serious restriction on the capability of *encoder-decoder* framework caused by the structure of RNNs. That is, when the RNN decoder generates the t-th word $\mathbf{y}_t$ in decoding phase, only $\mathbf{y}_{<t}$ can be utilized, while the possible words $\mathbf{y}_{>t}$ are directly neglected. Thus, it's difficult to capture global information especially the ungenerated words for the current dominant RNN decoder without new significant innovation. Under the premise of preserving the original structure, a promising alternative to address the aforementioned issue is to incorporate with auxiliary neural networks to extend the RNN decoder.

Towards above goal, polishing mechanism-based methods first capture the global information through a complete draft created by SMT or NMT, and then take it as input to finally generate a translation. Compared with conventional NMT, polishing mechanism-based methods make a more accurate prediction at each time-step due to the extra global understanding, resulting in more fluent and grammatically correct translation. While these approaches have demonstrated the effectiveness, previous approaches follow pre-defined routes to perform the decoding procedure, not considering choosing a suitable decoding depth for the complexity of source sentences completely.

Therefore, it's important to develop a novel framework for making an accurate and effective choice about which decoding depth is appropriate for the source sentence.

3 Adaptive Multi-pass Decoder

In this section, we present an adaptive multi-pass decoder for neural machine translation, as illustrated in Figure 1. It could choose a proper decoding depth, depending on the complexity of the source sentence. As shown in Figure 1, our model includes three major components: an encoder to summarize source sentences with parameter set θ_e, a multi-pass decoder for multi-pass decoding with parameter set θ_d, and a policy network to choose a suitable depth with parameter set θ_p. The encoder of our model is identical to that of the dominant NMT which is modeled using a bidirectional RNN. Please refer to Bahdanau et al. (2014) for more details. We will elaborate the *multi-pass decoder* and *policy network* for adaptive multi-pass decoding in the following subsections.

3.1 Multi-pass Decoder

The multi-pass decoder is extended from the one of the dominant NMT model to leverage the target-side context. Similar to the dominant NMT model, our multi-pass decoder also performs the decoding under the semantic guide of source-side context captured by the encoder, whereas more importantly and differently, the global understanding through the target-side context provided by last pass decoding, is able to strongly assist our model to produce a better translation.

Given the source-side and target-side contexts separately captured by the encoder and last pass decoding, the multi-pass decoder learns to generate next target word, based on previous generated words. Using the multi-pass decoder with parameter set θ_d, we calculate the conditional probability of the translation $\hat{\mathbf{y}}^l$ at the l-th decoding pass as follows:

$$\begin{aligned}
&P(\hat{\mathbf{y}}^l|\mathbf{x},\hat{\mathbf{y}}^{l-1};\theta_e,\theta_d)\\
&=\prod_{n=1}^{N_l} P(\hat{\mathbf{y}}^l_n|\mathbf{x},\hat{\mathbf{y}}^l_{<n},\hat{\mathbf{y}}^{l-1};\theta_e,\theta_d)\\
&=\prod_{n=1}^{N_l} g^{dec}(\hat{\mathbf{y}}^l_{n-1},\mathbf{s}^{l,dec}_n,\mathbf{c}^{l,enc}_n,\mathbf{c}^{l,dec}_n) \qquad (5)
\end{aligned}$$

where $g^{dec}(\cdot)$ is a non-linear function, and $\mathbf{s}^{l,dec}_n$ denotes the n-th decoding state within the l-th decoding pass. N_l indicates the length of generated translation at the l-th decoding pass. The decoding state $\mathbf{s}^{l,dec}_n$ is obtained by RNNs as follows:

$$\mathbf{s}^{l,dec}_n = f^{dec}(\mathbf{s}^{l,dec}_{n-1},\hat{\mathbf{y}}^l_{n-1},\mathbf{c}^{l,enc}_n,\mathbf{c}^{l,dec}_n) \qquad (6)$$

where $f^{dec}(\cdot)$ is the GRU activation function. $\mathbf{c}^{l,enc}_n$ and $\mathbf{c}^{l,dec}_n$ denote source-side and target-side contexts at the n-th time step within the l-th decoding pass, respectively. It should be noted that when the multi-pass decoder performs the first decoding, there doesn't exist any generated translation. To address this case, the first-pass target-side context $\mathbf{c}^{1,dec}$ is set to zero.

Among the aforementioned contexts, $\mathbf{c}^{l,enc}_n$ is obtained as the weighted sum of the source-side hidden states $\{\mathbf{h}_m\}$, while we take the target-side hidden states $\{\mathbf{s}^{l-1,dec}_n\}$ produced by last pass decoding as input to compute $\mathbf{c}^{l,dec}_n$. Similar to the dominant NMT model, we adopt the attention model (Bahdanau et al., 2014; Luong et al., 2015) to calculate the weights, which indicate the alignment probability. We assume that $attn^{enc}$ denotes the encoder-decoder attention model, which takes the source annotations $\{\mathbf{h}_m\}$ as input, while $attn^{dec}$ are introduced to calculate the weight which measures how well the decoding state $\mathbf{s}^{l,dec}$ attends the last-pass hidden states $\{\mathbf{s}^{l-1,dec}_n\}$.

Assuming $\mathbf{s}^a$ indicates the decoding state, which attends the annotations $\{\mathbf{s}^b_k\}$ with a length K, our attention model calculates the context vector $\mathbf{s}^c$ as follows:

$$\mathbf{s}^c = \sum_{k=1}^{K} \alpha_k \mathbf{s}^b_k \qquad (7)$$

$$\alpha_k = \frac{\exp(e_k)}{\sum_{k'=1}^{K}\exp(e_{k'})} \qquad (8)$$

$$e_k = (v_a)^T \tanh(W_a\mathbf{s}^a + U_a\mathbf{s}^b_k) \qquad (9)$$

where v_a, W_a and U_a are the parameters of attention model.

Given a training $(\mathbf{x},\mathbf{y})$, the translation route can be demonstrated as: $\mathbf{x}\rightarrow\hat{\mathbf{y}}^1\rightarrow\ldots\rightarrow\hat{\mathbf{y}}^l\rightarrow\ldots\rightarrow\hat{\mathbf{y}}^{L(\mathbf{x},\mathbf{y})-1}\rightarrow\mathbf{y}$.The intermediate translations $\{\hat{\mathbf{y}}^l\}$ are generated by decoding. Given a training corpus $D=\{\mathbf{x},\mathbf{y}\}$, we define the object function using cross-entropy at last pass decoding as follows:

$$\begin{aligned}
J^{dec}(\theta_e,\theta_d) = -\frac{1}{|D|}\operatorname*{arg\,min}_{\theta_e,\theta_d}\sum_{(\mathbf{x},\mathbf{y})\in D}\\
\{\log P(\mathbf{y}|\mathbf{x},\hat{\mathbf{y}}^{L(\mathbf{x},\mathbf{y})-1};\theta_e,\theta_d)\} \qquad (10)
\end{aligned}$$

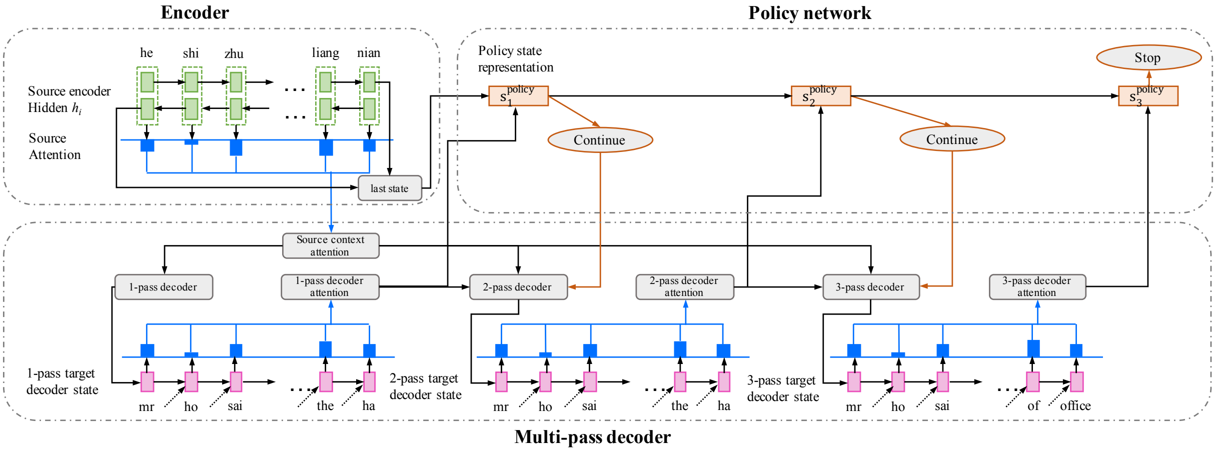

Figure 1: The architecture of our *adaptive multi-pass decoder*. Given the annotation sequence produced by the encoder, a policy network is adopted to choose a suitable action from the set $\{Continue, Stop\}$, which indicates continuing next pass decoding, or halt respectively. Different from the conventional decoder which only obtains the source-side context with the source attention model $attn^{enc}$, our multi-pass decoder also captures the target-side context of last-pass decoding with decoder attention model $attn^{dec}$. The policy network also use $attn^{policy}$ to collect useful information from the multi-pass decoding to choose an accurate and effective action to generate a good translation. Note that in this work the same parameters set of decoder and the corresponding attention is shared among different decoding passes. For this figure, we demonstrate a translation procedure with 3-pass decoding controlled by adaptive multi-pass decoder.

where $P(\mathbf{y}|\mathbf{x}, \hat{\mathbf{y}}^{L_{(\mathbf{x},\mathbf{y})}-1}; \theta_e, \theta_d)$ is conditional probability computed by multi-pass decoder. $L_{(\mathbf{x},\mathbf{y})}$ indicates the decoding depth for the instance $(\mathbf{x}, \mathbf{y})$. For effectiveness, note that all the intermediate translations $\{\hat{\mathbf{y}}^l\}$ are generated by greedy search in training and testing phase.

3.2 Policy Network

The multi-pass decoding can be converted into sequential decision making process, in which a policy is adopted to choose next pass decoding or halt. It's expected to automatically choose an accurate and effective decoding depth to generate a good translation. For example, if the source sentence is exhausted to obtain the corresponding translation such as the long sentences, we assume more decoding passes are needed to improve the translation, while only one pass decoding is enough to tackle the simple case.

Our main idea is to use reinforcement learning to control the decoding depth. We parameterize the available action $\mathbf{a}_l \in \{Continue, Stop\}$, where $Continue$ and $Stop$ indicate continuing next decoding pass and halt respectively, by a policy network $\pi(\mathbf{a}_l|\mathbf{s}_l^{policy}; \theta_p)$, where $\mathbf{s}_l^{policy}$ represents the policy state at the l-th decoding pass. For the purpose of making a better choice about the decoding depth and direction, it's necessary to consider whether or not the source sentence is easy to obtain a good translation and compared with the last pass decoding, whether the quality of translation can be improved. Thus, supervised by this guideline, the policy state $\mathbf{s}_l^{policy}$ is calculated by GRU to model the difference between the consecutive two decoding passes as follows:

$$\mathbf{s}_l^{policy} = f^{policy}(\mathbf{s}_{l-1}^{policy}, m_l) \tag{11}$$

where f^{policy} is the activation function, and m_l captures the useful information with respect to the policy network at the l-th decoding pass. In this work, we use the attention models $attn^{policy}$ to collect the decoding progress, denoted as m_l of the l-th decoding pass. In order to take account of the complexity of source sentence itself, the initial policy state $\mathbf{s}_0^{policy}$ is computed by $\mathbf{s}_0^{policy} = \tanh(W_{init}\mathbf{h}_M)$, where $\mathbf{h}_M$ is last state source annotations, and W_{init} is the parameters of initializing the policy state. Finally, we take the policy state $\mathbf{s}_l^{policy}$ as input to calculate the policy as follows:

$$\pi(\mathbf{a}_l|\mathbf{s}_l^{policy}; \theta_p) = softmax(W_p\mathbf{s}_l^{policy} + b_p) \tag{12}$$

where W_p and b_p are the parameters of the policy network. In this work we use REINFORCE algorithm (Williams, 1992), which is an instance of a broader class of algorithms called policy gradient methods (Sutton and Barto, 1998), to learn the parameter set θ_p such that the sequence of actions

$\mathbf{a} = \{\mathbf{a}_1, \ldots, \mathbf{a}_l, \ldots, \mathbf{a}_{L_{(\mathbf{x},\mathbf{y})}}\}$ maximizes the total expected reward. The expected reward for an instance is defined as:

$$J^{policy}(\theta_p) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s}^{policy};\theta_p)} r(\hat{\mathbf{y}}^{L_{(\mathbf{x},\mathbf{y})}}) \quad (13)$$

where $r(\hat{\mathbf{y}}^{L_{(\mathbf{x},\mathbf{y})}})$ is the reward at the $L_{(\mathbf{x},\mathbf{y})}$-th decoding pass. In this work, we use BLEU (Papineni et al., 2002) of the final translation $\hat{\mathbf{y}}^{L_{(\mathbf{x},\mathbf{y})}}$ generated by greedy search as input to compute our reward as follows:

$$r(\hat{\mathbf{y}}^{L_{(\mathbf{x},\mathbf{y})}}) = \text{BLEU}(\hat{\mathbf{y}}^{L_{(\mathbf{x},\mathbf{y})}}, \mathbf{y}) \quad (14)$$

4 Experiments

In this section, we describe experimental settings and report empirical results.

4.1 Setup

We evaluated the proposed *adaptive multi-pass decoder* on Chinese-English translation task. The evaluation metric was case-insensitive BLEU (Papineni et al., 2002) calculated by the `multi-bleu.perl`[1] script. The training corpus[2] consisted of 1.25M bilingual sentences with 27.9M Chinese words and 34.5M English words. We used the NIST 2002 (MT02) as the validation set for hyper-parameter optimization and model selection, and NIST 2003 (MT03), 2004 (MT04), 2005 (MT05) and 2006 (MT06) as test sets.

To effectively train the NMT model, we trained each model with sentences of length up to 50 words. Besides, we limited vocabulary size to 30K for both languages and map all the out-of-vocabulary words in the Chinese-English corpus to a special token UNK. We applied *Rmsprop* (Graves, 2013) to train models and selected the best model parameters according to the model performance on the development set. During this procedure, we set the following hyper-parameters: word embedding dimension as 620, hidden layer size as 1000, learning rate as 5×10^{-4}, batch size as 80, gradient norm as 1.0, and dropout rate as 0.3.

In the experiments, we compared our approach against the following state-of-the-art SMT and NMT systems:

1. *Moses*[3]: an open source phrase-based translation system with default configuration and a 4-gram language model trained on the target portion of training data. Note that we used all data to train MOSES (Koehn et al., 2007).

2. *RNNSearch*: a variant of the attention-based NMT system (Bahdanau et al., 2014) with slight changes from dl4mt tutorial[4].

3. *Deliberation Network*[5]: a re-implementation of attention-based NMT system with two independent left-to-right decoders (Xia et al., 2017). The first-pass decoder is identical to one of RNNSearch to generate a draft translation, while the second-pass decoder polishes it with an extra attention over the first pass decoder. The second-pass decoder is integrated with the first-pass decoder via reinforcement learning.

4. *ABDNMT*: As a comparison with the Deliberation Network, ABDNMT utilizes first-pass backward decoder to generate a translation with greedy search, and the second-pass forward decoder refines it with attention model (Zhang et al., 2018). For fairness, we replace the first-pass backward decoder with a forward decoder.

We set the beam size of all above-mentioned models as 10 in our work. Deliberation Network and ABDNMT were initialized with the pre-trained RNNSearch as Xia et al. (2017) and Zhang et al. (2018) described. Our multi-pass decoder was also initialized with RNNSearch and other parameters were randomly initialized from a uniform distribution on $[-0.1, 0.1]$. Besides, for effectiveness, we set the maximum decoding depth of our adaptive multi-pass decoder as 5.

4.2 Results on Chinese-English Translation

The experimental results of our model and baseline models on Chinese-English machine translation datasets are depicted in Table 2 .

---

[1] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

[2] The training corpus includes LDC2002E18, LDC2003E07, LDC2003E14, part of LDC2004T07, LDC2004T08 and LDC2005T06

[3] http://www.statmt.org/moses

[4] https://github.com/nyu-dl/dl4mt-tutorial

[5] We reproduce the deliberation network based on REINFORCE and gumbel-softmax (Jang et al., 2016), separately, but there still exists a gap with its best performance. We attribute this to that our reimplementation may be different from the original model in some unknown details.

System	#Params	Speed	MT02	MT03	MT04	MT05	MT06	Ave.
Moses	–	–	33.79	30.86	32.71	30.02	30.49	31.02
RNNSearch	83.99M	87	39.68	36.51	40.20	36.87	36.43	37.50
Deliberation Network	125.16M	162	40.98	37.82	40.56	37.67	37.20	38.31
ABDNMT	122.86M	132	41.12	38.01	41.20	38.07	37.59	38.71
2-pass decoder	87.81M	160	41.18	37.76	41.06	38.02	37.41	38.56
3-pass decoder	87.81M	245	41.28	37.99	40.72	37.86	37.63	38.55
4-pass decoder	87.81M	293	41.05	37.86	40.87	38.18	37.57	38.62
5-pass decoder	87.81M	322	40.88	37.70	40.84	38.06	**37.97**	38.64
Adaptive multi-pass decoder	96.01M	180	**41.42**	**38.39**	**41.43**	**38.54**	37.86	**39.05**

Table 2: Evaluation of the NIST Chinese-English translation task. The BLEU scores are case-insensitive. "Params" denotes the number the parameters in each model. The "Speed" denotes the generation speed in seconds on the development set. RNNSearch is an attention-based neural machine translation model(Bahdanau et al., 2014) with one-pass left-to-right decoding. RNNSearch(R2L) is a variant of RNNSearch with one-pass right-to-left decoding. As a comparison, Deliberation Network (Xia et al., 2017) and ABDNMT (Zhang et al., 2018) involve two independent decoders to adopt polishing mechanism to extend the ability of conventional NMT. Deliberation Network utilizes two left-to-right decoders coupled with reinforcement learning. However, ABDNMT exploits a backward decoder to perform first-pass right-to-left decoding. {2,3,4,5}-pass decoder utilizes our multi-pass decoder with a fixed number of decoding passes. Furthermore, adaptive multi-pass decoder involves a policy network to enhance our multi-pass decoder to choose a proper decoding depth.

Parameters RNNSearch, Deliberation Network and ABDNMT have 83.99M, 125.16M and 122.86M parameters, respectively. And the parameter size of our {2,3,4,5}-pass decoder and adaptive multi-pass decoder are about 87.81M and 96.01M, respectively.

Fixed Decoding Depth {2,3,4,5}-pass decoders perform the left-to-right decoding by the multi-pass decoder with a fixed number of decoding passes. In contrast to the related machine translation systems, our fixed number-pass decoder significantly outperforms Moses and RNNSearch by 7.53 and 1.05 BLEU points at least, as Table 2 presents. More importantly, our proposed multi-pass decoder obtains much better performance with an increase of only 3.82M parameters over RNNSearch. As a comparison with Deliberation Network involves two-pass decoding, the multi-pass decoder has a minimum increase of 0.24 BLEU score. Nevertheless, our multi-pass decoder proves its effectiveness due to the less parameters consumption of 37.35M in contrast to Deliberation Network. These results verify our hypothesis that the more decoding passes can polish the generated output to improve the translation quality. The underlying reason is that the attention component $attn^{dec}$ within our multi-pass decoder can capture the extra target-side contexts to obtain a global understanding to assist the translation procedure.

Towards the effect of the decoding depth set {2,3,4,5}, our multi-pass decoders obtain the approximate results, but the whole curve of BLEU is on an upward trend. Specifically, the multi-pass decoder with decoding depth 5 achieves the best performance with 38.64 BLEU, while the one with decoding depth 3 performs the worst among the decoding depth set with 38.55 BLEU. Although the average results of {2,3,4,5}-pass decoder are approximate, the distinction of {2,3,4,5}-pass decoder on NIST03, NIST04, NIST05, NIST06 and NIST08 is not negligible. These results indirectly prove the necessity of flexibility mechanism.

Adaptive Decoding Depth our proposed adaptive multi-pass decoder involves an extra policy network which controls the decoding depth according to the complexity of the source sentence and the differences between the consecutive generated translations. As shown in Table 2, the proposed adaptive multi-pass decoder obtains an improvement about 0.41 to 0.5 BLEU on average over the {2,3,4,5}-pass decoder, which demonstrates the effectiveness of the policy network. Specifically, the adaptive multi-pass decoder outperforms the multi-pass decoder with a fixed decoding depth by 0.69, 0.71, 0.68 and 0.45 BLEU scores on NIST03, NIST04, NIST05 and NIST06 datasets at most. In contrast to the Moses, RNNSearch, Deliberation Network and ABDNMT, the adaptive multi-pass decoder has the corresponding improvement about 8.03, 1.55, 0.74 and 0.34 BLEU points, respectively. More importantly, our adaptive multi-pass decoder outperforms ABDNMT, Deliberation Network model with a decrease of 26.85M, 29.15M parameters.

In order to further demonstrate the effective-

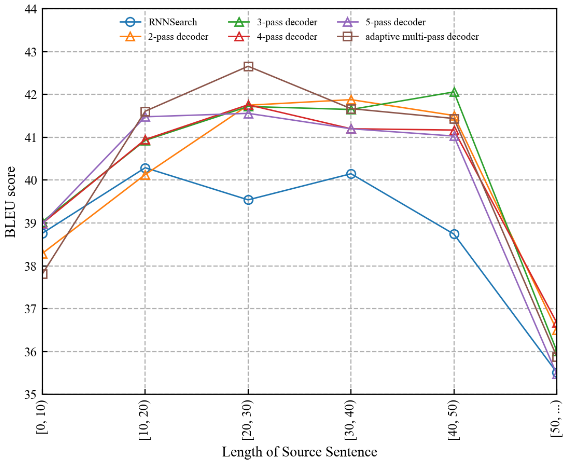

Figure 2: Performance of the generated translations with respect to the lengths of the source sentences on the development dataset.

ness of adaptively choosing the decoding depth, we investigate the ratio of decoding passes consumed by our multi-pass decoder on the development dataset, as shown in Table 3. Our adaptive multi-pass decoder chooses one-pass decoding in a high ratio of 46.57%, while in most about 53.43% cases our model leverages more than one pass decoding to produce a translation. The average decoding depth of our model is calculated as: $(1 \times 46.36\% + 2 \times 20.84\% + 3 \times 13.10\% + 4 \times 13.55\% + 5 \times 6.15\%) = 2.12$. Moreover, our ratio of the samples tends to decrease as the decoding depth rises on a whole. Since time consumption correlates with decoding depth, our adaptive multi-pass decoder proves its superior performance due to fewer parameters and less decoding passes.

Depth	1	2	3	4	5
Ratio(%)	46.57	20.45	13.00	13.60	6.38

Table 3: The ratio of decoding depth chosen by adaptive multi-pass decoder on the development dataset.

Time Consumption Due to the multi-pass decoding mechanism, the major limitation of our proposed multi-pass decoder is time cost. In training phrase, we spend more time training the multi-pass decoder than RNNSearch, Deliberation Network and ABDNMT. However, in testing phrase, as illustrated in Table 2, our adaptive multi-pass decoder spends about 180s completing the entire testing procedure, in comparison with the corresponding 87s, 162s, 132s of RNNSearch, Deliberation Network and ABDNMT, due to the auxiliary policy network. These results are consistent with above conclusion drew according to the decoding depth. Therefore, it's proven the necessity of our proposed auxiliary policy network to choose the decoding depth.

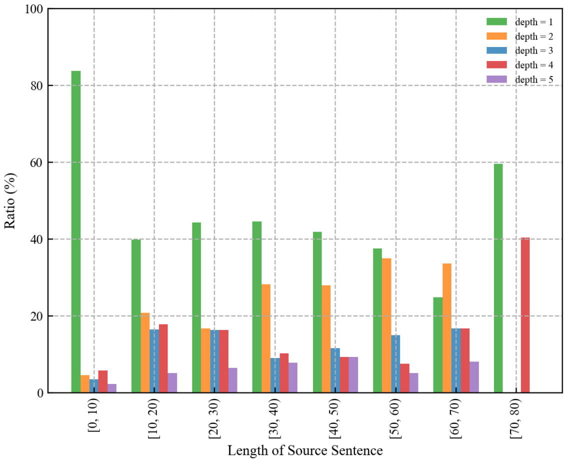

Figure 3: Ratio of decoding depth set {1,2,3,4,5} controlled by our adaptive multi-pass decoder with respect to each length segment of the source sentences on the development dataset.

4.3 Effect of Source Sentence Length

Following Bahdanau et al. (2014), we group sentences of similar lengths together and compute the BLEU score for each group, as shown in Figure 2. Obviously, our proposed adaptive multi-pass decoder outperforms RNNSearch in all length segments. Compared with {2,3,4,5}-pass decoders, our adaptive multi-pass decoder outperforms most even all the multi-pass decoders with fixed decoding depth in the length segments.

For the purpose of investigating the flexibility of policy network, we calculate the ratios of decoding depth set {1,2,3,4,5} on each sentence group with similar length, as illustrated in Figure 3. The ratio of one-pass decoding remains high level on each length segment, but explicitly is dominant on the length segment $[0, 10)$. In contrast, the ratios of remaining decoding depths show upwards trends on a whole. These results indirectly proves that our policy network has the capability of choosing the proper decoding depth. That is, when the source sentence is difficult to be translated such as some long sentences, more decoding passes are consumed to improve the translation quality, while in simple cases such as short source sentences, one-pass decoding is adequate.

4.4 Case Study

To better understand the effectiveness of flexible polishing mechanism adopted by policy network, Table 4 provides a Chinese-English transla-

Source	*xīnhuáshè běijīng 4 yuè 5 rì diàn dàmíngdǐngdǐng dē wēiruǎn gōngsī zǒngcái bǐěr · gàicí rìqián bèi jiānádà yì jiā guǎngbō diàn tái " shuàn le " yì bǎ , zài yúrénjié nàtiān bèi kāi le yīgè dàwángxiào .*
Reference	*xinhua news agency, beijing, april 5 , bill gates , the all - famous microsoft chairman, was duped by a canadian radio station the the other day and fell a victim to a big prank on april fools ' day.*
1st-pass	*xihua news agency report of april 5 th from beijing (by staff reporter UNK UNK) - the president of microsoft 's microsoft corporation , gates , was recently " UNK " by a radio station in canada and was hit by a UNK day on the day of the day .*
2nd-pass	*xinhua news agency, beijing, april 5, a fews days ago, microsoft 's president , microsoft corporation , was " UNK " by a radio station in canada .*
3rd-pass	***xinhua news agency, beijing, april 5 , microsoft 's president bill gates, the president of microsoft , was " UNK " by a radio station in canada in few days ago .***
4th-pass	*xinhua news agency, beijing, april 5 , microsoft 's president bill gates, the president of microsoft , was " UNK " by a radio station in canada in few days ago .*
5th-pass	*xinhua news agency, beijing, april 5 , microsoft 's president bill gates, the president of microsoft , was " UNK " by a radio station in canada in few days ago .*

Table 4: Translation examples at each decoding depth of adaptive multi-pass decoder.

tion example. Our proposed adaptive multi-pass decoder has the ability to polish the generated hypothesis again and again. As shown in Table 4, we force our adaptive multi-pass decoder to perform the multi-pass decoding with fixed depth sets {1,2,3,4,5}. The translation quality has an upwards trend with decoding depth 1 to 3, and the decoding with depth set {4,5} generates the identical translation as the decoding depth 3. Moreover, given the same source sentence, we use the proposed adaptive multi-pass decoder to choose the decoding depth. As expected, our adaptive multi-pass chooses 3-pass decoding which generates best translation and consumes least time, rather than {4,5}-pass decoding. Therefore, these results proves the effectiveness of our adaptive multi-pass decoder.

5 Related Work

In this work, we mainly focus on how to adopt adaptive polishing mechanism into NMT model, which has attracted intensive attention in recent years. We will elaborate polishing mechanism-based methods in the following pages.

The polishing mechanism-based approaches first generate a complete draft, and then improve the quality of it based on the global understanding of the whole draft. A related work is post-editing (Niehues et al., 2016; Chatterjee et al., 2016; Zhou et al., 2017; Junczys Dowmunt and Grundkiewicz, 2017): a source sentence e is first translated to f, and then f is refined by another model. Niehues et al. (2016) used phrase-based statistical machine translation (PBMT) to pre-translate the source sentence into target language, which was taken as input of NMT to generate the final translation. Zhou et al. (2017) combined phrase-based statistical machine translation (PBMT), hierarchical phrase-based statistical machine translation (HPMT) and NMT with a unified architecture, similar to the dominant NMT model. Compared with the dominant NMT model, two attention models were involved to compute the context vectors. Specifically, an attention model is utilized to calculate the context vector for each machine system, while the other attention model obtains the context vector over the all context vectors of machine systems.

In above works, the generating and refining are two separate processes. As a comparison, Xia et al. (2017) proposed deliberation network, which consists of two decoders: a first-pass decoder generates a draft, which is taken as input of second-pass decoder to obtain a better translation. All the components of deliberation network are coupled together and jointly optimized in an end-to-end way via reinforcement learning. Instead of first-pass forward decoder, Zhang et al. (2018) adopted a backward decoder to capture the right-to-left target-side contexts, which is taken as input to assist the second-pass forward decoder to obtain a better translation. Besides, the another difference with deliberation network is the second-pass decoder is integrated with the first-pass decoder without reinforcement learning.

For the purpose of exploring polishing mechanism, our model adopts adaptive multi-pass decoding strategy. Compared with the previous works which consumes no more than two decoding passes, our multi-pass decoder makes an attempt to perform the multi-pass decoding. More importantly, we adopt adaptive decoding depth controlled by policy network to extend the capacity of our multi-pass decoder.

6 Conclusion

In this paper, we propose a novel architecture called *adaptive multi-pass decoder* to adopt polishing mechanism into the NMT model via reinforcement learning. Towards this goal, a novel multi-pass decoder is introduced to generate the translation, conditioned on the source- and target-side contexts. Simultaneously, the multi-pass decoding is supervised by a policy network which learns to choose a suitable action from continuing next pass decoding or halt at each time step to maximize the BLEU of the final translation. As a result, our model has the capability of controlling the decoding depth to generate a better translation. Extensive experiments on Chinese-English translation demonstrate the effectiveness of the proposed adaptive multi-pass decoder.

In this paper, we focus on utilizing multi-pass decoder to polish the translation. Our proposed multi-pass decoder performs the multi-pass decoding mechanism with only forward decoding. One promising direction is to incorporate the backward decoding into our architecture. More specifically, we can extend the policy network to choose the backward decoding except for forward decoding and halting.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. We also thank Heng Gong and Shuang Chen for helpful discussion. This work was supported by the National Natural Science Foundation of China (NSFC) via grant 61632011, 61772156 and 61502120.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv e-prints*, abs/1409.0473.

Peter E. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2).

Rajen Chatterjee, José G. C. de Souza, Matteo Negri, and Marco Turchi. 2016. The fbk participation in the wmt 2016 automatic post-editing shared task. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 745–750. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Xiaocheng Feng, Jiang Guo, Bing Qin, Ting Liu, and Yongjie Liu. 2017. Effective deep memory networks for distant supervised relation extraction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, pages 19–25.

Xiaocheng Feng, Bing Qin, and Ting Liu. 2018. A language-independent neural network for event detection. *Science China Information Sciences*, 61(9):092106.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850.*

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

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144.*

Marcin Junczys Dowmunt and Roman Grundkiewicz. 2017. An exploration of neural sequence-to-sequence architectures for automatic post-editing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 120–129. Asian Federation of Natural Language Processing.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2016. Pre-translation for neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1828–1836. The COLING 2016 Organizing Committee.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1784–1794. Curran Associates, Inc.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. *arXiv preprint arXiv:1801.05122*.

Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. 2017. Neural system combination for machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 378–384. Association for Computational Linguistics.

Improving the Transformer Translation Model with Document-Level Context

Jiacheng Zhang[†], **Huanbo Luan**[†], **Maosong Sun**[†], **FeiFei Zhai**[#],
Jingfang Xu[#], **Min Zhang**[§] **and Yang Liu**[†‡*]
[†]Institute for Artificial Intelligence
State Key Laboratory of Intelligent Technology and Systems
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[‡]Beijing National Research Center for Information Science and Technology
[#]Sogou Inc., Beijing, China
[§]Soochow University, Suzhou, China

Abstract

Although the Transformer translation model (Vaswani et al., 2017) has achieved state-of-the-art performance in a variety of translation tasks, how to use document-level context to deal with discourse phenomena problematic for Transformer still remains a challenge. In this work, we extend the Transformer model with a new context encoder to represent document-level context, which is then incorporated into the original encoder and decoder. As large-scale document-level parallel corpora are usually not available, we introduce a two-step training method to take full advantage of abundant sentence-level parallel corpora and limited document-level parallel corpora. Experiments on the NIST Chinese-English datasets and the IWSLT French-English datasets show that our approach improves over Transformer significantly. [1]

1 Introduction

The past several years have witnessed the rapid development of neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015), which investigates the use of neural networks to model the translation process. Showing remarkable superiority over conventional statistical machine translation (SMT), NMT has been recognized as the new *de facto* method and is widely used in commercial MT systems (Wu et al., 2016). A variety of NMT models have been proposed to map between natural languages such as RNNencdec (Sutskever et al., 2014), RNNsearch (Bahdanau et al., 2015), ConvS2S (Gehring et al., 2017), and Transformer (Vaswani et al., 2017). Among them, the Transformer model has achieved state-of-the-art translation performance. The capability to minimize the path length between long-distance dependencies in neural networks contributes to its exceptional performance.

However, the Transformer model still suffers from a major drawback: it performs translation only at the sentence level and ignores document-level context. Document-level context has proven to be beneficial for improving translation performance, not only for conventional SMT (Gong et al., 2011; Hardmeier et al., 2012), but also for NMT (Wang et al., 2017; Tu et al., 2018). Bawden et al. (2018) indicate that it is important to exploit document-level context to deal with context-dependent phenomena which are problematic for machine translation such as coreference, lexical cohesion, and lexical disambiguation.

While document-level NMT has attracted increasing attention from the community in the past two years (Jean et al., 2017; Kuang et al., 2017; Tiedemann and Scherrer, 2017; Wang et al., 2017; Maruf and Haffari, 2018; Bawden et al., 2018; Tu et al., 2018; Voita et al., 2018), to the best of our knowledge, only one existing work has endeavored to model document-level context for the Transformer model (Voita et al., 2018). Previous approaches to document-level NMT have concentrated on the RNNsearch model (Bahdanau et al., 2015). It is challenging to adapt these approaches to Transformer because they are designed specifically for RNNsearch.

In this work, we propose to extend the Transformer model to take advantage of document-level context. The basic idea is to use multi-head self-attention (Vaswani et al., 2017) to compute the representation of document-level context, which is then incorporated into the encoder and decoder using multi-head attention. Since large-scale document-level parallel corpora are usually hard to acquire, we propose to train sentence-level model parameters on sentence-level paral-

---

[*] Corresponding author: Yang Liu.

[1] The source code is available at `https://github.com/Glaceon31/Document-Transformer`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 533–542
Brussels, Belgium, October 31 - November 4, 2018.

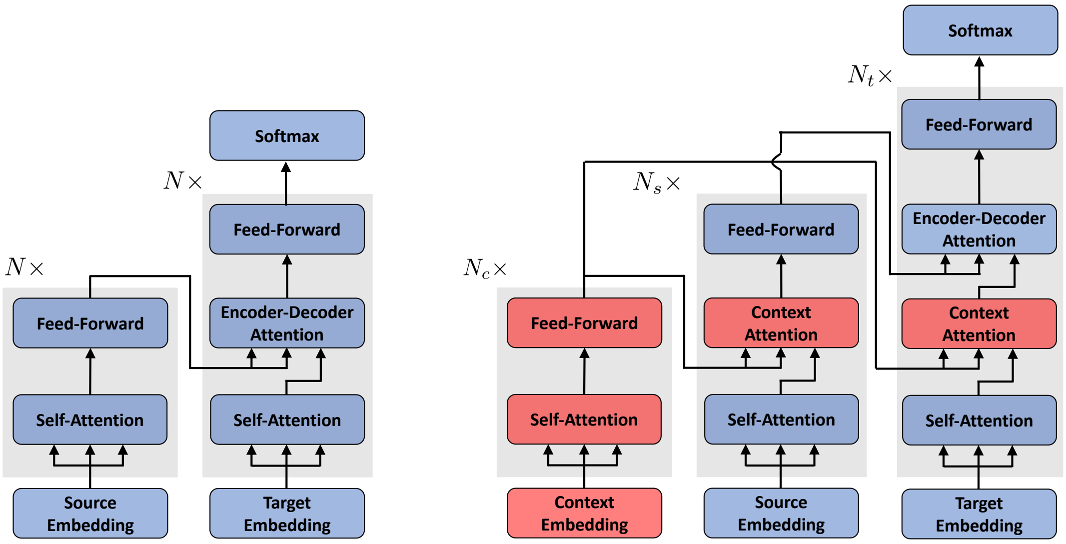

(a)

(b)

Figure 1: (a) The original Transformer translation model (Vaswani et al., 2017) and (b) the extended Transformer translation model that exploits document-level context. The newly introduced modules are highlighted in red.

lel corpora first and then estimate document-level model parameters on document-level parallel corpora while keeping the learned original sentence-level Transformer model parameters fixed. Our approach has the following advantages:

1. *Increased capability to capture context*: the use of multi-head attention, which significantly reduces the path length between long-range dependencies, helps to improve the capability to capture document-level context;
2. *Small computational overhead*: as all newly introduced modules are based on highly parallelizable multi-head attention, there is no significant slowdown in both training and decoding;
3. *Better use of limited labeled data*: our approach is capable of maintaining the superiority over the sentence-level counterpart even when only small-scale document-level parallel corpora are available.

Experiments show that our approach achieves an improvement of 1.96 and 0.89 BLEU points over Transformer on Chinese-English and French-English translation respectively by exploiting document-level context. It also outperforms a state-of-the-art cache-based method (Kuang et al., 2017) adapted for Transformer.

2 Approach

2.1 Problem Statement

Our goal is to enable the Transformer translation model (Vaswani et al., 2017) as shown in Figure 1(a) to exploit document-level context.

Formally, let $\mathbf{X} = \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k)}, \dots, \mathbf{x}^{(K)}$ be a source-language document composed of K source sentences. We use $\mathbf{x}^{(k)} = x_1^{(k)}, \dots, x_i^{(k)}, \dots, x_I^{(k)}$ to denote the k-th source sentence containing I words. $x_i^{(k)}$ denotes the i-th word in the k-th source sentence. Likewise, the corresponding target-language document is denoted by $\mathbf{Y} = \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(k)}, \dots, \mathbf{y}^{(K)}$ and $\mathbf{y}^{(k)} = y_1^{(k)}, \dots, y_j^{(k)}, \dots, y_J^{(k)}$ represents the k-th target sentence containing J words. $y_j^{(k)}$ denotes the j-th word in the k-th target sentence. We assume that $\langle \mathbf{X}, \mathbf{Y} \rangle$ constitutes a *parallel document* and each $\langle \mathbf{x}^{(k)}, \mathbf{y}^{(k)} \rangle$ forms a *parallel sentence*.

Therefore, the document-level translation probability is given by

$$P(\mathbf{Y}|\mathbf{X};\boldsymbol{\theta}) = \prod_{k=1}^{K} P(\mathbf{y}^{(k)}|\mathbf{X}, \mathbf{Y}_{<k};\boldsymbol{\theta}), \qquad (1)$$

where $\mathbf{Y}_{<k} = \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(k-1)}$ is a partial translation.

For generating $\mathbf{y}^{(k)}$, the source document $\mathbf{X}$ can be divided into three parts: (1) the k-th source sentence $\mathbf{X}_{=k} = \mathbf{x}^{(k)}$, (2) the source-side document-level context on the left $\mathbf{X}_{<k} = \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k-1)}$, and (3) the source-side document-level context on the right $\mathbf{X}_{>k} = \mathbf{x}^{(k+1)}, \ldots, \mathbf{x}^{(K)}$. As the languages used in our experiments (i.e., Chinese and English) are written left to right, we omit $\mathbf{X}_{>k}$ for simplicity.

We also omit the target-side document-level context $\mathbf{Y}_{<k}$ due to the translation error propagation problem (Wang et al., 2017): errors made in translating one sentence will be propagated to the translation process of subsequent sentences. Interestingly, we find that using source-side document-level context $\mathbf{X}_{<k}$, which conveys the same information with $\mathbf{Y}_{<k}$, helps to compute better representations on the target side (see Table 8).

As a result, the document-level translation probability can be approximated as

$$\begin{aligned} & P(\mathbf{Y}|\mathbf{X};\boldsymbol{\theta}) \\ \approx & \prod_{k=1}^{K} P(\mathbf{y}^{(k)}|\mathbf{X}_{<k}, \mathbf{x}^{(k)}; \boldsymbol{\theta}), \quad (2) \\ = & \prod_{k=1}^{K}\prod_{j=1}^{J} P(y_j^{(k)}|\mathbf{X}_{<k}, \mathbf{x}^{(k)}, \mathbf{y}_{<j}^{(k)}; \boldsymbol{\theta}), \quad (3) \end{aligned}$$

where $\mathbf{y}_{<j}^{(k)} = y_1^{(k)}, \ldots, y_{j-1}^{(k)}$ is a partial translation.

In this way, the document-level translation model can still be defined at the sentence level without sacrificing efficiency except that the source-side document-level context $\mathbf{X}_{<k}$ (or *context* for short) is taken into account.

In the following, we will introduce how to represent the context (Section 2.2), how to integrate the context (Section 2.3), and how to train the model especially when only limited training data is available (Section 2.4).

2.2 Document-level Context Representation

As document-level context often includes several sentences, it is important to capture long-range dependencies and identify relevant information. We use multi-head self-attention (Vaswani et al., 2017) to compute the representation of document-level context because it is capable of reducing the maximum path length between long-range dependencies to $O(1)$ (Vaswani et al., 2017) and determining the relative importance of different locations in the context (Bahdanau et al., 2015). Because of this property, multi-head self-attention has proven to be effective in other NLP tasks such as constituency parsing (Kitaev and Klein, 2018).

As shown in Figure 1(b), we use a self-attentive encoder to compute the representation of $\mathbf{X}_{<k}$. The input to the self-attentive encoder is a sequence of context word embeddings, represented as a matrix. Suppose $\mathbf{X}_{<k}$ is composed of M source words: $\mathbf{X}_{<k} = x_1, \ldots, x_m, \ldots, x_M$. We use $\boldsymbol{x}_m \in \mathbb{R}^{D\times 1}$ to denote the vector representation of x_m that is the sum of word embedding and positional encoding (Vaswani et al., 2017). Therefore, the matrix representation of $\mathbf{X}_{<k}$ is given by

$$\boldsymbol{X}_c = [\boldsymbol{x}_1; \ldots; \boldsymbol{x}_M], \quad (4)$$

where $\boldsymbol{X}_c \in \mathbb{R}^{D\times M}$ is the concatenation of all vector representations of all source contextual words.

The self-attentive encoder is composed of a stack of N_c identical layers. Each layer has two sub-layers. The first sub-layer is a multi-head self-attention:

$$\mathbf{A}^{(1)} = \mathrm{MultiHead}(\boldsymbol{X}_c, \boldsymbol{X}_c, \boldsymbol{X}_c), \quad (5)$$

where $\mathbf{A}^{(1)} \in \mathbb{R}^{D\times M}$ is the hidden state calculated by the multi-head self-attention at the first layer, $\mathrm{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ is a multi-head self-attention function that takes a query matrix $\mathbf{Q}$, a key matrix $\mathbf{K}$, and a value matrix $\mathbf{V}$ as inputs. In this case, $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \boldsymbol{X}_c$. This is why it is called self-attention. Please refer to (Vaswani et al., 2017) for more details.

Note that we follow Vaswani et al. (2017) to use residual connection and layer normalization in each sub-layer, which are omitted in the presentation for simplicity. For example, the actual output of the first sub-layer is:

$$\mathrm{LayerNorm}(\mathbf{A}^{(1)} + \boldsymbol{X}_c). \quad (6)$$

The second sub-layer is a simple, position-wise fully connected feed-forward network:

$$\mathbf{C}^{(1)} = \Big[\mathrm{FNN}(\mathbf{A}^{(1)}_{\cdot,1}); \ldots; \mathrm{FNN}(\mathbf{A}^{(1)}_{\cdot,M})\Big] \quad (7)$$

where $\mathbf{C}^{(1)} \in \mathbb{R}^{D\times M}$ is the annotation of $\mathbf{X}_{<k}$ after the first layer, $\mathbf{A}^{(1)}_{\cdot,m} \in \mathbb{R}^{D\times 1}$ is the column vector for the m-th contextual word, and $\mathrm{FNN}(\cdot)$ is a position-wise fully connected feed-forward network (Vaswani et al., 2017).

This process iterates N_c times as follows:

$$\mathbf{A}^{(n)} = \mathrm{MultiHead}\Big(\mathbf{C}^{(n-1)}, \mathbf{C}^{(n-1)}, \mathbf{C}^{(n-1)}\Big), \quad (8)$$

$$\mathbf{C}^{(n)} = \Big[\mathrm{FNN}(\mathbf{A}^{(n)}_{\cdot,1}); \ldots; \mathrm{FNN}(\mathbf{A}^{(n)}_{\cdot,M})\Big], \quad (9)$$

where $\mathbf{A}^{(n)}$ and $\mathbf{C}^{(n)}$ ($n = 1, \ldots, N_c$) are the hidden state and annotation at the n-th layer, respectively. Note that $\mathbf{C}^{(0)} = \boldsymbol{X}_c$.

2.3 Document-level Context Integration

We use multi-head attention to integrate $\mathbf{C}^{(N_c)}$, which is the representation of $\mathbf{X}_{<k}$, into both the encoder and the decoder.

2.3.1 Integration into the Encoder

Given the k-th source sentence $\mathbf{x}^{(k)}$, we use $\boldsymbol{x}_i^{(k)} \in \mathbb{R}^{D \times 1}$ to denote the vector representation of the i-th source word $x_i^{(k)}$, which is a sum of word embedding and positional encoding. Therefore, the initial matrix representation of $\mathbf{x}^{(k)}$ is

$$\boldsymbol{X} = [\boldsymbol{x}_1^{(k)}; \ldots; \boldsymbol{x}_I^{(k)}], \tag{10}$$

where $\boldsymbol{X} \in \mathbb{R}^{D \times I}$ is the concatenation of all vector representations of source words.

As shown in Figure 1(b), we follow (Vaswani et al., 2017) to use a stack of N_s identical layers to encode $\mathbf{x}^{(k)}$. Each layer consists of three sub-layers. The first sub-layer is a multi-head self-attention:

$$\mathbf{B}^{(n)} = \mathrm{MultiHead}\Big(\mathbf{S}^{(n-1)}, \mathbf{S}^{(n-1)}, \mathbf{S}^{(n-1)}\Big), \tag{11}$$

where $\mathbf{S}^{(0)} = \boldsymbol{X}$. The second sub-layer is context attention that integrates document-level context into the encoder:

$$\mathbf{D}^{(n)} = \mathrm{MultiHead}\Big(\mathbf{B}^{(n)}, \mathbf{C}^{(N_c)}, \mathbf{C}^{(N_c)}\Big). \tag{12}$$

The third sub-layer is a position-wise fully connected feed-forward neural network:

$$\mathbf{S}^{(n)} = \Big[\mathrm{FNN}(\mathbf{D}_{\cdot,1}^{(n)}); \ldots; \mathrm{FNN}(\mathbf{D}_{\cdot,I}^{(n)})\Big], \tag{13}$$

where $\mathbf{S}^{(n)} \in \mathbb{R}^{D \times I}$ is the representation of the source sentence $\mathbf{x}^{(k)}$ at the n-th layer ($n = 1, \ldots, N_s$).

2.3.2 Integration into the Decoder

When generating the j-th target word $y_j^{(k)}$, the partial translation is denoted by $\mathbf{y}_{<j}^{(k)} = y_1^{(k)}, \ldots, y_{j-1}^{(k)}$. We follow Vaswani et al. (2017) to offset the target word embeddings by one position, resulting in the following matrix representation of $\mathbf{y}_{<j}^{(k)}$:

$$\boldsymbol{Y} = [\boldsymbol{y}_0^{(k)}, \ldots, \boldsymbol{y}_{j-1}^{(k)}], \tag{14}$$

where $\mathbf{y}_0^{(k)} \in \mathbb{R}^{D \times 1}$ is the vector representation of a begin-of-sentence token and $\boldsymbol{Y} \in \mathbb{R}^{D \times j}$ is the concatenation of all vectors.

As shown in Figure 1(b), we follow (Vaswani et al., 2017) to use a stack of N_t identical layers to compute target-side representations. Each layer is composed of four sub-layers. The first sub-layer is a multi-head self-attention:

$$\mathbf{E}^{(n)} = \mathrm{MultiHead}\Big(\mathbf{T}^{(n-1)}, \mathbf{T}^{(n-1)}, \mathbf{T}^{(n-1)}\Big), \tag{15}$$

where $\mathbf{T}^{(0)} = \boldsymbol{Y}$. The second sub-layer is context attention that integrates document-level context into the decoder:

$$\mathbf{F}^{(n)} = \mathrm{MultiHead}\Big(\mathbf{E}^{(n)}, \mathbf{C}^{(N_c)}, \mathbf{C}^{(N_c)}\Big). \tag{16}$$

The third sub-layer is encoder-decoder attention that integrates the representation of the corresponding source sentence:

$$\mathbf{G}^{(n)} = \mathrm{MultiHead}\Big(\mathbf{F}^{(n)}, \mathbf{S}^{(N_s)}, \mathbf{S}^{(N_s)}\Big). \tag{17}$$

The fourth sub-layer is a position-wise fully connected feed-forward neural network:

$$\mathbf{T}^{(n)} = \Big[\mathrm{FNN}(\mathbf{G}_{\cdot,1}^{(n)}); \ldots; \mathrm{FNN}(\mathbf{G}_{\cdot,j}^{(n)}),\Big], \tag{18}$$

where $\mathbf{T}^{(n)} \in \mathbb{R}^{D \times j}$ is the representation at the n-th layer ($n = 1, \ldots, N_t$). Note that $\mathbf{T}^{(0)} = \boldsymbol{Y}$.

Finally, the probability distribution of generating the next target word $y_j^{(k)}$ is defined using a softmax layer:

$$P(y_j^{(k)} | \mathbf{X}_{<k}, \mathbf{x}^{(k)}, \mathbf{y}_{<j}^{(k)}; \boldsymbol{\theta}) \propto \exp(\mathbf{W}_o \mathbf{T}_{\cdot,j}^{(N_t)}) \tag{19}$$

where $\mathbf{W}_o \in \mathbb{R}^{|\mathcal{V}_y| \times D}$ is a model parameter, $\mathcal{V}_y$ is the target vocabulary, and $\mathbf{T}_{\cdot,j}^{(N_t)} \in \mathbb{R}^{D \times 1}$ is a column vector for predicting the j-th target word.

2.3.3 Context Gating

In our model, we follow Vaswani et al. (2017) to use residual connections (He et al., 2016) around each sub-layer to shortcut its input to its output:

$$\mathrm{Residual}(\mathbf{H}) = \mathbf{H} + \mathrm{SubLayer}(\mathbf{H}), \tag{20}$$

where $\mathbf{H}$ is the input of the sub-layer.

While residual connections prove to be effective for building deep architectures, there is one potential problem for our model: the residual connections after the context attention sub-layer might increase the influence of document-level context

$\mathbf{X}_{<k}$ in an uncontrolled way. This is undesirable because the source sentence $\mathbf{x}^{(k)}$ usually plays a more important role in target word generation.

To address this problem, we replace the residual connections after the context attention sub-layer with a *position-wise context gating* sub-layer:

$$\mathrm{Gating}(\mathbf{H}) = \lambda\mathbf{H} + (1-\lambda)\mathrm{SubLayer}(\mathbf{H}). \quad (21)$$

The gating weight is given by

$$\lambda = \sigma(\mathbf{W}_i\mathbf{H} + \mathbf{W}_s\mathrm{SubLayer}(\mathbf{H})), \quad (22)$$

where $\sigma(\cdot)$ is a sigmoid function, $\mathbf{W}_i$ and $\mathbf{W}_s$ are model parameters.

2.4 Training

Given a document-level parallel corpus D_d, the standard training objective is to maximize the log-likelihood of the training data:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\mathrm{argmax}} \left\{ \sum_{\langle \mathbf{X},\mathbf{Y}\rangle \in D_d} \log P(\mathbf{Y}|\mathbf{X};\boldsymbol{\theta}) \right\}. \quad (23)$$

Unfortunately, large-scale document-level parallel corpora are usually unavailable, even for resource-rich languages such as English and Chinese. Under small-data training conditions, document-level NMT is prone to underperform sentence-level NMT because of poor estimates of low-frequency events.

To address this problem, we adopt the idea of freezing some parameters while tuning the remaining part of the model (Jean et al., 2015; Zoph et al., 2016). We propose a two-step training strategy that uses an additional sentence-level parallel corpus D_s, which can be larger than D_d. We divide model parameters into two subsets: $\boldsymbol{\theta} = \boldsymbol{\theta}_s \cup \boldsymbol{\theta}_d$, where $\boldsymbol{\theta}_s$ is a set of original sentence-level model parameters (highlighted in blue in Figure 1(b)) and $\boldsymbol{\theta}_d$ is a set of newly-introduced document-level model parameters (highlighted in red in Figure 1(b)).

In the first step, sentence-level parameters $\boldsymbol{\theta}_s$ are estimated on the combined sentence-level parallel corpus $D_s \cup D_d$: [2]

$$\hat{\boldsymbol{\theta}}_s = \underset{\boldsymbol{\theta}_s}{\mathrm{argmax}} \sum_{\langle \mathbf{x},\mathbf{y}\rangle \in D_s \cup D_d} \log P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_s). \quad (24)$$

Note that the newly introduced modules (highlighted in red in Figure 1(b)) are inactivated in this step. $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_s)$ is identical to the original Transformer model, which is a special case of our model.

In the second step, document-level parameters $\boldsymbol{\theta}_d$ are estimated on the document-level parallel corpus D_d only:

$$\hat{\boldsymbol{\theta}}_d = \underset{\boldsymbol{\theta}_d}{\mathrm{argmax}} \sum_{\langle \mathbf{X},\mathbf{Y}\rangle \in D_d} \log P(\mathbf{Y}|\mathbf{X};\hat{\boldsymbol{\theta}}_s,\boldsymbol{\theta}_d). \quad (25)$$

Our approach is also similar to pre-training which has been widely used in NMT (Shen et al., 2016; Tu et al., 2018). The major difference is that our approach keeps $\hat{\boldsymbol{\theta}}_s$ *fixed* when estimating $\boldsymbol{\theta}_d$ to prevent the model from overfitting on the relatively smaller document-level parallel corpora.

3 Experiments

3.1 Setup

We evaluate our approach on Chinese-English and French-English translation tasks. In Chinese-English translation task, the training set contains 2M Chinese-English sentence pairs with 54.8M Chinese words and 60.8M English words. [3] The document-level parallel corpus is a subset of the full training set, including 41K documents with 940K sentence pairs. On average, each document in the training set contains 22.9 sentences. We use the NIST 2006 dataset as the development set and the NIST 2002, 2003, 2004, 2005, 2008 datasets as test sets. The development and test sets contain 588 documents with 5,833 sentences. On average, each document contains 9.9 sentences.

In French-English translation task, we use the IWSLT bilingual training data (Mauro et al., 2012) which contains 1,824 documents with 220K sentence pairs as training set. For development and testing, we use the IWSLT 2010 development and test sets, which contains 8 documents with 887 sentence pairs and 11 documents with 1,664 sentence pairs respectively. The evaluation metric for both tasks is case-insensitive BLEU score as calculated by the *multi-bleu.perl* script.

In preprocessing, we use byte pair encoding (Sennrich et al., 2016) with 32K merges to segment words into sub-word units for all languages. For the original Transformer model and our extended model, the hidden size is set to 512 and the

---

[2] It is easy to create a sentence-level parallel corpus from D_d.

[3] The training set consists of sentence-level parallel corpora LDC2002E18, LDC2003E07, LDC2003E14, news part of LDC2004T08 and document-level parallel corpora LDC2002T01, LDC2004T07, LDC2005T06, LDC2005T10, LDC2009T02, LDC2009T15, LDC2010T03.

# sent.	1	2	3
MT06	49.38	**49.69**	49.49

Table 1: Effect of context length on translation quality. The BLEU scores are calculated on the development set.

# Layer	MT06
1	**49.69**
2	49.38
3	49.54
4	49.59
5	49.31
6	49.43

Table 2: Effect of self-attention layer number (i.e., N_c) on translation quality. The BLEU scores are calculated on the development set.

filter size is set to 2,048. The multi-head attention has 8 individual attention heads. We set $N = N_s = N_t = 6$. In training, we use Adam (Kingma and Ba, 2015) for optimization. Each mini-batch contains approximately 24K words. We use the learning rate decay policy described by Vaswani et al. (2017). In decoding, the beam size is set to 4. We use the length penalty (Wu et al., 2016) and set the hyper-parameter α to 0.6. We use four Tesla P40 GPUs for training and one Tesla P40 GPU for decoding. We implement our approach on top of the open-source toolkit THUMT (Zhang et al., 2017). [4]

3.2 Effect of Context Length

We first investigate the effect of context length (i.e., the number of preceding sentences) on our approach. As shown in Table 1, using two preceding source sentences as document-level context achieves the best translation performance on the development set. Using more preceding sentences does not bring any improvement and increases computational cost. This confirms the finding of Tu et al. (2018) that long-distance context only has limited influence. Therefore, we set the number of preceding sentences to 2 in the following experiments. [5]

3.3 Effect of Self-Attention Layer Number

Table 2 shows the effect of self-attention layer number for computing representations of document-level context (see Section 2.2) on translation quality. Surprisingly, using only one self-attention layer suffices to achieve good performance. Increasing the number of self-attention layers does not lead to any improvements. Therefore, we set N_c to 1 for efficiency.

3.4 Comparison with Previous Work

In Chinese-English translation task, we compare our approach with the following previous methods:

1. (Wang et al., 2017): using a hierarchical RNN to integrate document-level context into the RNNsearch model. They use a document-level parallel corpus containing 1M sentence pairs. Table 3 gives the BLEU scores reported in their paper.

2. (Kuang et al., 2017): using a cache which stores previous translated words and topical words to incorporate document-level context into the RNNsearch model. They use a document-level parallel corpus containing 2.8M sentence pairs. Table 3 gives the BLEU scores reported in their paper.

3. (Vaswani et al., 2017): the state-of-the-art NMT model that does not exploit document-level context. We use the open-source toolkit THUMT (Zhang et al., 2017) to train and evaluate the model. The training dataset is our sentence-level parallel corpus containing 2M sentence pairs.

4. (Kuang et al., 2017)*: adapting the cache-based method to the Transformer model. We implement it on top of the open-source toolkit THUMT. We also use the same training data (i.e., 2M sentence pairs) and the same two-step training strategy to estimate sentence- and document-level parameters separately.

As shown in Table 3, using the same data, our approach achieves significant improvements over the original Transformer model (Vaswani et al., 2017) ($p < 0.01$). The gain on the concatenated test set (i.e., "All") is 1.96 BLEU points. It also outperforms the cache-based method (Kuang et al., 2017) adapted for Transformer significantly ($p < 0.01$), which also uses the two-step training strategy. Table 4 shows that our model also outperforms Transformer by 0.89 BLEU points on French-English translation task.

---

[4] `https://github.com/thumt/THUMT`

[5] If there is no preceding sentence, we simply use a single begin-of-sentence token.

Method	Model	MT06	MT02	MT03	MT04	MT05	MT08	All
(Wang et al., 2017)	RNNsearch	37.76	-	-	-	36.89	27.57	-
(Kuang et al., 2017)	RNNsearch	-	34.41	-	38.40	32.90	31.86	-
(Vaswani et al., 2017)	Transformer	48.09	48.63	47.54	47.79	48.34	38.31	45.97
(Kuang et al., 2017)*	Transformer	48.14	48.97	48.05	47.91	48.53	38.38	46.37
this work	Transformer	**49.69**	**50.96**	**50.21**	**49.73**	**49.46**	**39.69**	**47.93**

Table 3: Comparison with previous works on Chinese-English translation task. The evaluation metric is case-insensitive BLEU score. (Wang et al., 2017) use a hierarchical RNN to incorporate document-level context into RNNsearch. (Kuang et al., 2017) use a cache to exploit document-level context for RNNsearch. (Kuang et al., 2017)* is an adapted version of the cache-based method for Transformer. Note that "MT06" is not included in "All".

Method	Dev	Test
Transformer	29.42	35.15
this work	**30.40**	**36.04**

Table 4: Comparison with Transformer on French-English translation task. The evaluation metric is case-insensitive BLEU score.

	>	=	<
Human 1	24%	45%	31%
Human 2	20%	55%	25%
Human 3	12%	52%	36%
Overall	19%	51%	31%

Table 5: Subjective evaluation of the comparison between the original Transformer model and our model. ">" means that Transformer is better than our model, "=" means equal, and "<" means worse.

3.5 Subjective Evaluation

We also conducted a subjective evaluation to validate the benefit of exploiting document-level context. All three human evaluators were asked to compare the outputs of the original Transformer model and our model of 20 documents containing 198 sentences, which were randomly sampled from the test sets.

Table 5 shows the results of subjective evaluation. Three human evaluators generally made consistent judgements. On average, around 19% of Transformer's translations are better than that of our model, 51% are equal, and 31% are worse. This evaluation confirms that exploiting document-level context helps to improve translation quality.

3.6 Evaluation of Efficiency

We evaluated the efficiency of our approach. It takes the original Transformer model about 6.7 hours to converge during training and the training speed is 41K words/second. The decoding speed is 872 words/second. In contrast, it takes our model about 7.8 hours to converge in the second step of training. The training speed is 31K words/second. The decoding speed is 364 words/second.

Method	Training	Decoding
Transformer	41K	872
this work	31K	364

Table 6: Evaluation of training and decoding speed. The speed is measured in terms of word/second (wps).

Therefore, the training speed is only reduced by 25% thanks to the high parallelism of multi-head attention used to incorporate document-level context. The gap is larger in decoding because target words are generated in an autoregressive way in Transformer.

3.7 Effect of Two-Step Training

Table 7 shows the effect of the proposed two-step training strategy. The first two rows only use sentence-level parallel corpus to train the original Transformer model (see Eq. 24) and achieve BLEU scores of 39.53 and 45.97. The third row only uses the document-level parallel corpus to directly train our model (see Eq. 23) and achieves a BLEU score of 36.52. The fourth and fifth rows use the two-step strategy to take advantage of both sentence- and document-level parallel corpora and achieve BLEU scores of 40.22 and 47.93, respectively.

We find that document-level NMT achieves much worse results than sentence-level NMT (i.e., 36.52 vs. 39.53) when only small-scale document-level parallel corpora are available. Our two-step training method is capable of addressing this problem by exploiting sentence-level corpora, which

sent.	doc.	MT06	MT02	MT03	MT04	MT05	MT08	All
940K	-	36.20	42.41	43.12	41.02	40.93	31.49	39.53
2M	-	48.09	48.63	47.54	47.79	48.34	38.31	45.97
-	940K	34.00	38.83	40.51	38.30	36.69	29.38	36.52
940K	940K	37.12	43.29	43.70	41.42	41.84	32.36	40.22
2M	940K	49.69	50.96	50.21	49.73	49.46	39.69	47.93

Table 7: Effect of two-step training. "sent." denotes sentence-level parallel corpus and "doc." denotes document-level parallel corpus.

Integration	MT06	MT02	MT03	MT04	MT05	MT08	All
none	48.09	48.63	47.54	47.79	48.34	38.31	45.97
encoder	48.88	50.30	49.34	48.81	49.75	39.55	47.51
decoder	49.10	50.31	49.83	49.35	49.29	39.07	47.48
both	49.69	50.96	50.21	49.73	49.46	39.69	47.93

Table 8: Effect of context integration. "none" means that no document-level context is integrated, "encoder" means that the document-level context is integrated only into the encoder, "decoder" means that the document-level context is integrated only into the decoder, and "both" means that the context is integrated into both the encoder and the decoder.

Gating	MT06	MT02	MT03	MT04	MT05	MT08	All
w/o	49.33	50.56	49.74	49.29	50.11	39.02	47.55
w/	49.69	50.96	50.21	49.73	49.46	39.69	47.93

Table 9: Effect of context gating.

leads to significant improvements across all test sets.

3.8 Effect of Context Integration

Table 8 shows the effect of integrating document-level context to the encoder and decoder (see Section 2.3). It is clear that integrating document-level context into the encoder (Eq. 12) brings significant improvements (i.e., 45.97 vs. 47.51). Similarly, it is also beneficial to integrate document-level context into the decoder (Eq. 16). Combining both leads to further improvements. This observation suggests that document-level context does help to improve Transformer.

3.9 Effect of Context Gating

As shown in Table 9, we also validated the effectiveness of context gating (see Section 2.3.3). We find that replacing residual connections with context gating leads to an overall improvement of 0.38 BLEU point.

3.10 Analysis

We use an example to illustrate how document-level context helps translation (Table 10). In order to translate the source sentence, NMT has to disambiguate the multi-sense word "*yundong*", which is actually impossible without the document-level context. The exact meaning of "*rezhong*" is also highly context dependent. Fortunately, the sense of "*yundong*" can be inferred from the word "*saiche*" (car racing) in the document-level context and "*rezhong*" is the antonym of "*yanjuan*" (tired of). This example shows that our model learns to resolve word sense ambiguity and lexical cohesion problems by integrating document-level context.

4 Related Work

Developing document-level models for machine translation has been an important research direction, both for conventional SMT (Gong et al., 2011; Hardmeier et al., 2012; Xiong et al., 2013a,b; Garcia et al., 2014) and NMT (Jean et al., 2017; Kuang et al., 2017; Tiedemann and Scherrer, 2017; Wang et al., 2017; Maruf and Haffari, 2018; Bawden et al., 2018; Tu et al., 2018; Voita et al., 2018).

Most existing work on document-level NMT has focused on integrating document-level context into the RNNsearch model (Bahdanau et al.,

Context	*· · ·ziji ye yinwei queshao jingzheng duishou er dui* ***saiche*** *youxie* ***yanjuan*** *shi· · ·*
Source	*wo rengran feichang* ***rezhong*** *yu zhexiang* ***yundong****.*
Reference	I'm still very **fond of** the **sport**.
Transformer	I am still very **enthusiastic about** this **movement**.
Our work	I am still very **keen on** this **sport**.

Table 10: An example of Chinese-English translation. In the source sentence, "*yundong*" (sport or political movement) is a multi-sense word and "*rezhong*" (fond of) is an emotional word whose meaning is dependent on its context. Our model takes advantage of the words "*saiche*" (car racing) and "*yanjuan*" (tired of) in the document-level context to translate the source words correctly.

2015). These approaches can be roughly divided into two broad categories: computing the representation of the full document-level context (Jean et al., 2017; Tiedemann and Scherrer, 2017; Wang et al., 2017; Maruf and Haffari, 2018; Voita et al., 2018) and using a cache to memorize most relevant information in the document-level context (Kuang et al., 2017; Tu et al., 2018). Our approach falls into the first category. We use multi-head attention to represent and integrate document-level context.

Voita et al. (2018) also extended Transformer to model document-level context, but our work is different in modeling and training strategies. The experimental part is also different. While Voita et al. (2018) focus on anaphora resolution, our model is able to improve the overall translation quality by integrating document-level context.

5 Conclusion

We have presented a method for exploiting document-level context inside the state-of-the-art neural translation model Transformer. Experiments on Chinese-English and French-English translation tasks show that our method is able to improve over Transformer significantly. In the future, we plan to further validate the effectiveness of our approach on more language pairs.

Acknowledgments

Yang Liu is supported by the National Natural Science Foundation of China (No. 61432013), National Key R&D Program of China (No. 2017YFB0202204), National Natural Science Foundation of China (No. 61761166008), Advanced Innovation Center for Language Resources (TYR17002), and the NExT++ project supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC@Singapore Funding Initiative. This research is also supported by Sogou Inc.

References

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Sequence to sequence learning with neural networks. In *Proceedings of ICLR*.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating discourse phenomena in neural machine translation. In *Proceedings of NAACL*.

Eva Martínez Garcia, Cristina Espãna Bonet, and Lluíz Màrquez. 2014. Document-level machine translation with word vector models. In *Proceedings of EACL*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level statistical machine translation. In *Proceedings of EMNLP*.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-wide decoding for phrase-based statistical machine translation. In *Proceedings of EMNLP*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of CVPR*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL*.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? *CoRR*, abs/1704.05135.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder.

Shaohui Kuang, Deyi Xiong, Weihua Luo, and Guodong Zhou. 2017. Cache-based document-level neural machine translation. *CoRR*, abs/1711.11221.

Sameen Maruf and Gholamreza Haffari. 2018. Document context neural machine translation with memory networks. In *Proceedings of ACL*.

Cettolo Mauro, Girardi Christian, and Federico Marcello. 2012. Wit3: Web inventory of transcribed and translated talks. In *Proceedings of EAMT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *Transactions of the Association for Computational Linguistics*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of ACL*.

Longyue Wang, Zhaopeng Tu, Andy Way, and Liu Qun. 2017. Exploiting cross-sentence context for neural machine translation. In *Proceedings of EMNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Deyi Xiong, Guosheng Ben, Min Zhang, Yajuan Lv, and Qun Liu. 2013a. Modeling lexical cohesion for document-level machine translation. In *Proceedings of IJCAI*.

Deyi Xiong, Yang Ding, Min Zhang, and Chew Lim Tan. 2013b. Lexical chain based cohesion models for document-level statistical machine translation. In *Proceedings of EMNLP*.

Jiacheng Zhang, Yanzhuo Ding, Shiqi Shen, Yong Cheng, Maosong Sun, Huanbo Luan, and Yang Liu. 2017. Thumt: An open source toolkit for neural machine translation. *arXiv preprint arXiv:1706.06415*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of EMNLP*.

MTNT: A Testbed for Machine Translation of Noisy Text

Paul Michel and **Graham Neubig**
Language Technologies Institute
Carnegie Mellon University
`{pmichel1,gneubig}@cs.cmu.edu`

Abstract

Noisy or non-standard input text can cause disastrous mistranslations in most modern Machine Translation (MT) systems, and there has been growing research interest in creating noise-robust MT systems. However, as of yet there are no publicly available parallel corpora of with naturally occurring noisy inputs and translations, and thus previous work has resorted to evaluating on synthetically created datasets. In this paper, we propose a benchmark dataset for Machine Translation of Noisy Text (MTNT), consisting of noisy comments on Reddit[1] and professionally sourced translations. We commissioned translations of English comments into French and Japanese, as well as French and Japanese comments into English, on the order of 7k-37k sentences per language pair. We qualitatively and quantitatively examine the types of noise included in this dataset, then demonstrate that existing MT models fail badly on a number of noise-related phenomena, even after performing adaptation on a small training set of in-domain data. This indicates that this dataset can provide an attractive testbed for methods tailored to handling noisy text in MT.[2]

1 Introduction

#nlproc is actualy f*ing hARD tbh 😂

This handcrafted sentence showcases several types of noise that are commonly seen on social media: abbreviations (“#nlproc”), typographical errors (“actualy”), obfuscated profanities (“f*ing”), inconsistent capitalization (“hARD”), Internet slang (“tbh” for “to be honest”) and emojis (😂). Although machine translation has achieved significant quality improvements over the past few years due to the advent of Neural Machine Translation (NMT) (Kalchbrenner and Blunsom; Sutskever et al., 2014; Bahdanau et al., 2014; Wu et al., 2016), systems are still not robust to noisy input like this (Belinkov and Bisk, 2018; Khayrallah and Koehn). For example, Google Translate[3] translates the above example into French as:

#nlproc est en train de f * ing dur hb

which translates back into English as “#nlproc is in the process of [f * ing] hard hb”. This shows that noisy input can lead to erroneous translations that can be misinterpreted or even offensive.

Noise in social media text is a known issue that has been investigated in a variety of previous work (Eisenstein; Baldwin et al.). Most recently, Belinkov and Bisk (2018) have focused on the difficulties that character based NMT models have translating text with character level noise within individual words (from scrambling to simulated human errors such as typos or spelling/conjugation errors). This is a good first step towards noise-robust NMT systems, but as we demonstrate in §2, word-by-word replacement or scrambling of characters doesn’t cover all the idiosyncrasies of language on the Internet.

At this point, despite the obvious utility of creating noise-robust MT systems, and the scientific challenges contained therein, there is currently a bottleneck in that there is no standard open benchmark for researchers and developers of MT systems to test the robustness of their models to these and other phenomena found in noisy text on the Internet. In this work, we introduce MTNT, a new, realistic dataset aimed at testing robustness of MT systems to these phenomena. The dataset contains naturally created noisy source

---

[1] `www.reddit.com`

[2] The data is publicly available at `http://www.cs.cmu.edu/~pmichel1/mtnt/`.

[3] `translate.google.com` as of May 2018

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 543–553
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

sentences with professionally sourced translations both in a pair of typologically close languages (English and French) and distant languages (English and Japanese). We collect noisy comments from the Reddit online discussion website (§3) in English, French and Japanese, and ask professional translators to translate to and from English, resulting in approximately 1000 test samples and from 6k to 36k training samples in four language pairs (English-French (`en-fr`), French-English (`fr-en`), English-Japanese (`en-ja`) and Japanese-English (`ja-en`)). In addition, we release additional small monolingual corpora in those 3 languages to both provide data for semi-supervised adaptation approaches as well as noisy Language Modeling (LM) experiments. We test standard translation models (§5) and language models (§6) on our data to understand their failure cases and to provide baselines for future work.

2 Noise and Input Variations in Language on the Internet

2.1 Examples from Social Media Text

The term "noise" can encompass a variety of phenomena in natural language, with variations across languages (*e.g.* what is a typo in logographic writing systems?) and type of content (Baldwin et al.). To give the reader an idea of the challenges posed to MT and Natural Language Processing (NLP) systems operating on this kind of text, we provide a non-exhaustive list of types of noise and more generally input variations that deviate from standard MT training data we've encountered in Reddit comments:

- **Spelling/typographical errors**: "across" → "accross", "receive" → "recieve", "could have" → "could of", "temps" → "tant", "除く" → "覗く"
- **Word omission/insertion/repetition**: "je n'aime pas" → "j'aime pas","je pense" → "moi je pense"
- **Grammatical errors**: "a ton of" → "a tons of", "There are fewer people" → "There are less people"
- **Spoken language**: "want to" → "wanna", "I am" → "I'm", "je ne sais pas" → "chais pas", "何を笑っているの" → "何わろてんねん",
- **Internet slang**: "to be honest" → "tbh", "shaking my head" → "smh", "mort de rire" → "mdr", "笑" → "w"/"草"
- **Proper nouns** (with or without correct capitalization): "Reddit"→ "reddit"
- **Dialects**: African American Vernacular English, Scottish, Provençal, Québécois, Kansai, Tohoku...
- **Code switching**: "This is so cute" → "This is so kawaii", "C'est trop conventionel" → "C'est trop mainstream", "現在捏造中…" → "Now 捏造ing..."
- **Jargon**: on Reddit: "upvote", "downvote", "sub", "gild"
- **Emojis and other unicode characters**: ❤️,😀,😂,😍,😡, 😘, 😭
- **Profanities/slurs** (sometimes masked) "f*ck", "m*rde" …

2.2 Is Translating Noisy Text just another Adaptation Problem?

To a certain extent, translating noisy text is a type of *adaptation*, which has been studied extensively in the context of both Statistical Machine Translation (SMT) and NMT (Axelrod et al.; Li et al.; Luong and Manning, 2015; Chu et al.; Miceli Barone et al.; Wang et al.; Michel and Neubig, 2018). However, it presents many differences with previous domain adaptation problems, where the main goal is to adapt from a particular topic or style. In the case of noisy text, it will not only be the case that a particular word will be translated in a different way than it is in the general domain (e.g. as in the case of "sub"), but also that there will be increased lexical variation (e.g. due to spelling or typographical errors), and also inconsistency in grammar (e.g. due to omissions of critical words or mis-usage). The sum of these differences warrants that noisy MT be treated as a separate instance than domain adaptation, and our experimental analysis in 5.4 demonstrates that even after performing adaptation, MT systems still make a large number of noise-related errors.

3 Collection Procedure

We first collect noisy sentences in our three languages of interest, English, French and Japanese.

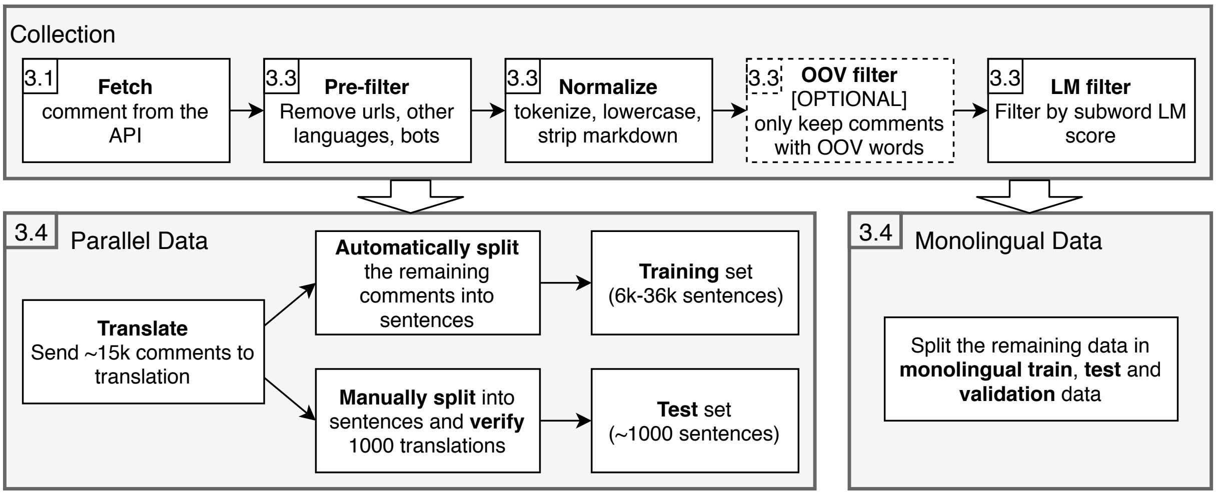

Figure 1: Summary of our collection process and the respective sections addressing them. We apply the same procedure for each language.

We refer to Figure 1 for an overview of the data collection and translation process.

We choose Reddit as a source of data because (1) its content is likely to exhibit noise, (2) some of its sub-communities are entirely run in different languages, in particular, English, French and Japanese, and (3) Reddit is a popular source of data in curated and publicly distributed NLP datasets (Tan et al.). We collect data using the public Reddit API. [4]

Note that the data collection and translation is performed at the comment level. We split the parallel data into sentences as a last step.

3.1 Data Sources

For each language, we select a set of communities ("subreddits") that we know contain many comments in that language:

English: Since an overwhelming majority of the discussions on Reddit are conducted in English, we don't restrict our collection to any community in particular.

French: `/r/france`, `/r/quebec` and `/r/rance`. The first two are among the biggest French speaking communities on Reddit. The third is a humor/sarcasm based offspring of `/r/france`.

Japanese: `/r/newsokur`, `/r/bakanewsjp`, `/r/newsokuvip`, `/r/lowlevelaware` and `/r/steamr`. Those are the biggest Japanese speaking communities, with over 2,000 subscribers.

We collect comments made during the 03/27/2018-03/29/3018 time period for English, 09/2018-03/2018 for French and 11/2017-03/2018 for Japanese. The large difference in collection time is due to the variance in comment throughput and relative amount of noise between the languages.

3.2 Contrast Corpora

Not all comments found on Reddit exhibit noise as described in Section 2. Because we would like to focus our data collection on noisy comments, we devise criteria that allow us to distinguish potentially noisy comments from clean ones. Specifically, we compile a *contrast* corpus composed of clean text that we can compare to, and find potentially noisy text that differs greatly from the contrast corpus. Given that our final goal is MT robust to noise, we prefer that these contrast corpora consist of the same type of data that is often used to train NMT models. We select different datasets for each language:

English: The English side of the preprocessed parallel training data provided for the German-English WMT 2017 News translation task,[5] as provided on the website. This amounts to ≈ 5.85 million sentences.

---

[4] In particular, we use this implementation: `praw.readthedocs.io/en/latest`, and our complete code is available at `http://www.cs.cmu.edu/~pmichel1/mtnt/`.

[5] `http://www.statmt.org/wmt17/translation-task.html`

French: The entirety of the French side of the parallel training data provided for the English-French WMT 2015 translation task.[6] This amounts to ≈ 40.86 million sentences.

Japanese: We aggregate three small/medium sized MT datasets: KFTT (Neubig, 2011), JESC (Pryzant et al.) and TED talks (Cettolo et al., 2012), amounting to $\approx$ 4.19 million sentences.

3.3 Identifying Noisy Comments

We now describe the procedure used to identify comments containing noise.

Pre-filtering First, we perform three pre-processing to discard comments that do not represent natural noisy text in the language of interest:

1. Comments containing a URL, as detected by a regular expression.
2. Comments where the author's username contains "bot" or "AutoModerator". This mostly removes automated comments from bots.
3. Comments in another language: we run `langid.py`[7] (Lui and Baldwin) and discard comments where $p(\text{lang} \mid \text{comment}) > 0.5$ for any language other than the one we are interested in.

This removes cases that are less interesting, i.e. those that could be solved by rule-based pattern matching or are not natural text created by regular users in the target language. Our third criterion in particular discards comments that are blatantly in another language while still allowing comments that exhibit code-switching or that contain proper nouns or typos that might skew the language identification. In preliminary experiments, we noticed that these criteria 14.47, 6.53 and 7.09 % of the collected comments satisfied the above criteria respectively.

Normalization After this first pass of filtering, we pre-process the comments before running them through our noise detection procedure. We first strip Markdown[8] syntax from the comments. For English and French, we normalize the punctuation, lowercase and tokenize the comments using the Moses tokenizer. For Japanese, we simply lowercase the alphabetical characters in the comments. Note that this normalization is done for the purpose of noise detection only. The collected comments are released without any kind of pre-processing. We apply the same normalization procedure to the contrast corpora.

Unknown words In the case of French and English, a clear indication of noise is the presence of out-of-vocabulary words (OOV): we record all lowercased words encountered in our reference corpus described in Section 3.2 and only keep comments that contain at least one OOV. Since we did not use word segmentation for the Japanese reference corpus, we found this method not to be very effective to select Japanese comments and therefore skipped this step.

Language model scores The final step of our noise detection procedure consists of selecting those comments with a low probability under a language model trained on the reference monolingual corpus. This approach mirrors the one used in Moore and Lewis and Axelrod et al. to select data similar to a specific domain using language model perplexity as a metric. We search for comments that have a low probability under a sub-word language model for more flexibility in the face of OOV words. We segment the contrast corpora with Byte-Pair Encoding (BPE) using the sentencepiece[9] implementation. We set the vocabulary sizes to $1,000$, $1,000$ and $4,000$ for English, French and Japanese respectively. We then use a 5-gram Kneser-Ney smoothed language model trained using `kenLM`[10] (Heafield et al.) to calculate the log probability, normalized by the number of tokens for every sentence in the reference corpus. Given a reddit comment, we compute the normalized log probability of each of its lines under our subword language model. If for any line this score is below the 1st percentile of scores in the reference corpus, the comment is labeled as noisy and saved.

3.4 Creating the Parallel Corpora

Once enough data has been collected, we isolate $15,000$ comments in each language by the follow-

---

[6] `http://www.statmt.org/wmt15/translation-task.html`

[7] `https://github.com/saffsd/langid.py`

[8] `https://daringfireball.net/projects/markdown`

[9] `https://github.com/google/sentencepiece`

[10] `https://kheafield.com/code/kenlm/`

	#samples	#src tokens	#trg tokens
`en-fr`	1,020	15,919	18,445
`fr-en`	1,022	16,662	16,038
`en-ja`	1,002	11,040	20,008
`ja-en`	1,020	23,997	33,429

Table 1: Test set numbers.

ing procedure:

- Remove all duplicates. In particular, this handles comments that might have been scraped twice or automatic comments from bots.
- To further weed out outliers (comments that are too noisy, *e.g.* ASCII art, wrong language. . . or not noisy enough), we discard comments that are on either end of the distribution of normalized LM scores within the set of collected comments. We only keep comments whose normalized score is within the 5-70 percentile for English (resp. 5-60 for French and 10-70 for Japanese). These numbers are chosen by manually inspecting the data.
- Choose $15,000$ samples at random.

We then concatenate the title of the thread where the comment was found to the text and send everything to an external vendor for manual translations. Upon reception of the translations, we noticed a certain amount of variation in the quality of translations, likely because translating social media text, with all its nuances, is difficult even for humans. In order to ensure the highest quality in the translations, we manually filter the data to segment the comments into sentences and weed out poor translations for our test data. We thereby retain around $1,000$ sentence pairs in each direction for the final test set.

We gather the samples that weren't selected for the test sets to be used for training or fine-tuning models on noisy data. We automatically split comments into sentences with a regular expression detecting sentence delimiters, and then align the source and target sentences. Should this alignment fail (*i.e.* the source comment contains a different number of sentences than the target comment after automatic splitting), we revert back to providing the whole comment without splitting. For the training data, we do not verify the correctness of translations as closely as for the test data. Finally, we isolate $\approx$ 900 samples in each direction to serve as validation data.

	#samples	#src tokens	#trg tokens
`en-fr`	36,058	841k	965k
`fr-en`	19,161	661k	634k
`en-ja`	5,775	281k	506k
`ja-en`	6,506	172k	128k

Table 2: Training sets numbers.

	#samples	#src tokens	#trg tokens
`en-fr`	852	16,957	18,948
`fr-en`	886	41,578	46,886
`en-ja`	852	40,124	46,886
`ja-en`	965	25,010	23,289

Table 3: Validation sets numbers.

Information about the size of the data can be found in Table 1, 2 and 3 for the test, training and validation sets respectively. We tokenize the English and French data with the Moses (Koehn et al.) tokenizer and the Japanese data with Kytea (Neubig et al., 2011) before counting the number of tokens in each dataset.

3.5 Monolingual Corpora

After the creation of the parallel train and test sets, a large number of unused comments remain in each language, which we provide as monolingual corpora. This additional data has two purposes: first, it serves as a resource for in-domain training using semi-supervised methods relying on monolingual data (e.g. Cheng et al.; Zhang and Zong). Second, it provides a language modeling dataset for noisy text in three languages.

We select $3,000$ comments at random in each dataset to form a validation set to be used to tune hyper-parameters, and provide the rest as training data. The data is provided with one comment per line. Newlines within individual comments are replaced with spaces. Table 4 contains information

		#samples	#tok	#char
`en`	train	81,631	3,99M	18,9M
	dev	3,000	146k	698k
`fr`	train	26,485	1,52M	7,49M
	dev	3,000	176k	867k
`ja`	train	32,042	943k	3.9M
	dev	3,000	84k	351k

Table 4: Monolingual data numbers.

		Spelling	Grammar	Emojis	Profanities
`en`	newstest2014	0.210	0.189	0.000	0.030
	newsdiscusstest2015	0.621	0.410	0.021	0.076
	MTNT (`en-fr`)	**2.180**	**0.559**	**0.289**	**0.239**
`fr`	newstest2014	2.776	0.091	0.000	0.245
	newsdiscusstest2015	1.686	0.457	0.024	0.354
	MTNT	**4.597**	**1.464**	**0.252**	**0.690**
`ja`	TED	0.011	0.266	0.000	0.000
	KFTT	0.021	0.228	0.000	0.000
	JESC	0.096	0.929	0.090	**0.058**
	MTNT	**0.269**	**1.527**	**0.156**	0.036

Table 5: Numbers, per 100 tokens, of quantifiable noise occurrences. For each language and category, the dataset with the highest amount of noise is highlighted.

on the size of the datasets. As with the parallel MT data, we provide the number of tokens after tokenization with the Moses tokenizer for English and French and Kytea for Japanese.

4 Dataset Analysis

In this section, we investigate the proposed data to understand how different categories of noise are represented and to show that our test sets contain more noise overall than established MT benchmarks.

4.1 Quantifying Noisy Phenomena

We run a series of tests to count the number of occurrences of some of the types of noise described in Section 2. Specifically we pass our data through spell checkers to count spelling and grammar errors. Due to some of these tests being impractical to run on a large scale, we limit our analysis to the test sets of MTNT.

We use slightly different procedures depending on the tools available for each language. We test for spelling and grammar errors in English data using Grammarly[11], an online resource for English spell-checking. Due to the unavailability of an equivalent of Grammarly in French and Japanese, we test for spelling and grammar error using the integrated spell-checker in Microsoft Word 2013[12]. Note that Word seems to count proper nouns as spelling errors, giving higher numbers of spelling errors across the board in French as compared to English.

For all languages, we also count the number of profanities and emojis using custom-made lists and regular expressions[13]. In order to compare results across datasets of different sizes, we report all counts per 100 words.

The results are recorded in the last row of each section in Table 5. In particular, for the languages with a segmental writing system, English and French, spelling errors are the dominant type of noise, followed by grammar error. Unsurprisingly, the former are much less present in Japanese.

4.2 Comparison to Existing MT Test Sets

Table 5 also provide a comparison with the relevant side of established MT test sets. For English and French, we compare our data to newstest2014[14] and newsdiscusstest2015[15] test sets. For Japanese, we compare with the test sets of the datasets described in Section 3.2.

Overall, MTNT contains more noise in all metrics but one (there are more profanities in JESC, a Japanese subtitle corpus). This confirms that MTNT indeed provides a more appropriate benchmark for translation of noisy or non-standard text.

Compared to synthetically created noisy test sets (Belinkov and Bisk, 2018) MTNT contains less systematic spelling errors and more varied types of noise (*e.g.* emojis and profanities) and is thereby more representative of naturally occurring noise.

---

[11] `https://www.grammarly.com/`

[12] `https://products.office.com/en-us/microsoft-word-2013`

[13] available with our code at `https://github.com/pmichel31415/mtnt`

[14] `http://www.statmt.org/wmt15/dev-v2.tgz`

[15] `http://www.statmt.org/wmt15/test.tgz`

5 Machine Translation Experiments

We evaluate standard NMT models on our proposed dataset to assess its difficulty. Our goal is not to train state-of-the art models but rather to test standard off-the-shelf NMT systems on our data, and elucidate what features of the data make it difficult.

5.1 Model Description

All our models are implemented in DyNet (Neubig et al., 2017) with the XNMT toolkit (?). We use approximately the same setting for all language pairs: the encoder is a bidirectional LSTM with 2 layers, the attention mechanism is a multi layered perceptron and the decoder is a 2 layered LSTM. The embedding dimension is 512, all other dimensions are 1024. We tie the target word embeddings and the output projection weights (Press and Wolf). We train with Adam (Kingma and Ba, 2014) with XNMT's default hyper-parameters, as well as dropout (with probability 0.3). We used BPE subwords to handle OOV words. Full configuration details as well as code to reproduce the baselines is available at `https://github.com/pmichel31415/mtnt`.

5.2 Training Data

We train our models on standard MT datasets:

- en ↔ fr: Our training data consists in the europarl-v7[16] and news-commentary-v10[17] corpora, totaling 2,164,140 samples, 54,611,105 French tokens and 51,745,611 English tokens (non-tokenized). We use the newsdiscussdev2015[14] dev set from WMT15 as validation data and evaluate the model on the newsdiscusstest2015[15] and newstest2014[14] test sets.

- en ↔ ja: We concatenate the respective train, validation and test sets of the three corpora mentioned in 3.2. In particular we detokenize the Japanese part of each dataset to make sure that any tokenization we perform will be uniform (in practice we remove ASCII spaces). This amounts to 3,900,772 training samples (34,989,346 English tokens without tokenization). We concatenate the dev sets associated with these corpora to serve as validation data and evaluate on each respective test set separately.

	`en-fr`	`fr-en`
newstest2014	33.52	28.93
newsdiscusstest2015	33.03	30.76
MTNT	21.77	23.27
MTNT (+tuning)	29.73	30.29
	`en-ja`	`ja-en`
TED	14.51	13.25
KFTT	20.82	20.77
JESC	15.77	18.00
MTNT	9.02	6.65
MTNT (+tuning)	12.45	9.82

Table 6: BLEU scores of NMT models on the various datasets.

5.3 Results

We use `sacreBLEU`[18], a standardized BLEU score evaluation script proposed by Post (2018), for BLEU evaluation of our benchmark dataset. It takes in detokenized references and hypotheses and performs its own tokenization before computing BLEU score. We specify the `intl` tokenization option. In the case of Japanese text, we run both hypothesis and reference through KyTea before computing BLEU score. We strongly encourage that evaluation be performed in the same manner in subsequent work, and will provide both scripts and an evaluation web site in order to facilitate reproducibility.

Table 6 lists the BLEU scores for our models on the relevant test sets in the two language pairs, including the results on MTNT.

5.4 Analysis

To better understand the types of errors made by our model, we count the n-grams that are over- and under- generated with respect to the reference translation. Specifically, we compare the count ratios of all 1- to 3-grams in the output and in the reference and look for the ones with the highest (over-generated) and lowest (under-generated) ratio.

We find that in English, the model under-generates the contracted form of the negative ("do not"/"don't") or of auxiliaries ("That is"/"I'm").

---

[16] `http://www.statmt.org/europarl/`

[17] `http://www.statmt.org/wmt15/training-parallel-nc-v10.tgz`

[18] `https://github.com/mjpost/sacreBLEU`

Source	Moi faire la gueule dans le métro me manque, c’est grave ?
Target	I miss sulking in the underground, is that bad?
Our model	I do not know what is going on in the metro, that is a serious matter.
+ fine-tuning	I do not want to be in the metro, it’s serious?
Source	:o ’tain je me disais bien que je passais à côté d’un truc vu les upvotes.
Target	:o damn I had the feeling that I was missing something considering the upvotes.
Our model	o, I was telling myself that I was passing over a nucleus in view of the Yupvoots.
+ fine-tuning	o, I was telling myself that I was going next to a nucleus in view of the <unk>upvotes.
Source	* C’est noël / pâques / pentecôte / toussaint : Pick One, je suis pas catho
Target	Christmas / Easter / Pentecost / All Saints: Pick One, I’m not Catholic!
Our model	<unk> It is a pale/poward, a palec<unk>te d’<unk>tat: Pick One, I am not a catho!
+ fine-tuning	<unk> It’s no<unk>l / pesc<unk>e /pentecate /mainly: Pick One, I’m not catho!

Table 7: Comparison of our model’s output before and after fine-tuning in `fr-en`.

Similarly, in French, our model over generates “de votre” (where “votre” is the formal 2nd person plural for “your”) and “n’ai pas” which showcases the “ne […] pas” negation, often dropped in spoken language. Conversely, the informal second person “tu” is under-generated, as is the informal and spoken contraction of “cela”, “ça”. In Japanese, the model under-generates, among others, the informal personal pronoun 俺 (“ore”) or the casual form だ (“da”) of the verb です (“desu”, to be). In `ja-en` the results are difficult to interpret as the model seems to produce incoherent outputs (*e.g.* “no, no, no…”) when the NMT system encounters sentences it has not seen before. The full list of n-grams with the top 5 and bottom 5 count ratios in each language pair is displayed in Table 8.

`fr-en`	`en-fr`	`ja-en`	`en-ja`
Over generated			
<unk>	<unk>	no, no,	※
it is not	qu’ils	i	が
I do not	de votre	no, no, no,	か？
That is	s’il	so on and	て
not have	n’ai pas	on and so	すか？
Under generated			
it’s	tu	|	？
I’m	ça	Is	よ。
I don’t	que tu	>	って
>	!	""The	俺
doesn’t	as	those	だ。

Table 8: Over and under generated n-grams in our model’s output for `en-fr`

5.5 Fine-Tuning

Finally, we test a simple domain adaptation method by fine-tuning our models on the training data described in Section 3.4. We perform one epoch of training with vanilla SGD with a learning rate of 0.1 and a batch size of 32. We do not use the validation data at all. As evidenced by the results in the last row of Table 6, this drives BLEU score up by 3.17 to 7.96 points depending on the language pair. However large this increase might be, our model still breaks on very noisy sentences. Table 7 shows three examples in `fr-en`. Although our model somewhat improves after fine-tuning, the translations remain inadequate in all cases. In the third case, our model downright fails to produce a coherent output. This shows that despite improving BLEU score, naive domain adaptation by fine-tuning doesn’t solve the problem of translating noisy text.

6 Language Modeling Experiments

In addition to our MT experiments, we report character-level language modeling results on the monolingual part of our dataset. We use the data described in Section 3.5 as training and validation sets. We evaluate the trained model on the source side of our `en-fr`, `fr-en` and `ja-en` test sets for English, French and Japanese respectively.

We report results for two models: a Kneser-Ney smoothed 6-gram model (implemented with KenLM) and an implementation of the AWD-LSTM proposed in (Merity et al., 2018)[19]. We report the Bit-Per-Character (bpc) counts in table 9.

---

[19] `https://github.com/salesforce/awd-lstm-lm`

	6-gram		AWD LSTM	
	dev	test	dev	test
English	2.081	2.179	1.706	1.810
French	1.906	2.090	1.449	1.705
Japanese	5.003	5.497	4.801	5.225

Table 9: Language modeling scores

We intend these results to serve as a baseline for future work in language modeling of noisy text in either of those three languages.

7 Related work

Handling noisy text has received growing attention among various language processing tasks due to the abundance of user generated content on popular social media platforms (Crystal, 2001; Herring, 2003; Danet and Herring, 2007). These contents are considered as noisy when compared to news corpora which have been the main data source for language tasks (Baldwin et al.; Eisenstein). They pose several unique challenges because they contain a larger variety of linguistic phenomena that are absent in the news domain and that lead to degraded quality when applying an model to out-of-domain data (Ritter et al.; Luong and Manning, 2015). Additionally, they are live examples of the Cmabrigde Uinervtisy (Cambridge University) effect, where state-of-the-art models become brittle while human's language processing capability is more robust (Sakaguchi et al., 2017; Belinkov and Bisk, 2018).

Efforts to address these challenges have been focused on creating in-domain datasets and annotations (Owoputi et al.; Kong et al.; Blodgett et al., 2017), and domain adaptation training (Luong and Manning, 2015). In MT, improvements were obtained for SMT (Formiga and Fonollosa). However, the specific challenges for neural machine translation have not been studied until recently (Belinkov and Bisk, 2018; Sperber et al.; Cheng et al., 2018). The first provides empirical evidence of non-trivial quality degradation when source sentences contain natural noise or synthetic noise within words, and the last two explore data augmentation and adversarial approaches of adding noise efficiently to training data to improve robustness.

Our work also contributes to recent advances in evaluating neural machine translation quality with regard to specific linguistic phenomena, such as manually annotated test sentences for English to French translation, in order to identify errors due to specific linguistic divergences between the two languages (Isabelle et al.), or automatically generated test sets to evaluate typical errors in English to German translation (Sennrich). Our contribution distinguishes itself from this previous work and other similar initiatives (Peterson, 2011) by providing an open test set consisting of naturally occurring text exhibiting a wide range of phenomena related to noisy input text from contemporaneous social media.

8 Conclusion

We proposed a new dataset to test MT models for robustness to the types of noise encountered in natural language on the Internet. We contribute parallel training and test data in both directions for two language pairs, English ↔ French and English ↔ Japanese, as well as monolingual data in those three languages. We show that this dataset contains more noise than existing MT test sets and poses a challenge to models trained on standard MT corpora. We further demonstrate that these challenges cannot be overcome by a simple domain adaptation approach alone. We intend this contribution to provide a standard benchmark for robustness to noise in MT and foster research on models, dataset and evaluation metrics tailored for this specific problem.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. How noisy social media text, how diffrnt social media sources? In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 356–364.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. *International Conference on Learning Representations*.

Su Lin Blodgett, Johnny Wei, and Brendan O'Connor. 2017. A dataset and classifier for recognizing social media english. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 56–61.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1965–1974.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391.

David Crystal. 2001. *Language and the Internet*. Cambridge University Press.

Brenda Danet and Susan Herring. 2007. *The Multilingual Internet: Language, Culture, and Communication Online*. Oxford University Press., New York.

Jacob Eisenstein. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369.

Lluís Formiga and José A. R. Fonollosa. Dealing with input noise in statistical machine translation. In *Proceedings of the Conference on Computational Linguistics 2012: Posters*, pages 319–328.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696.

Susan Herring, editor. 2003. *Media and Language Change*. Special issue of Journal of Historical Pragmatics 4:1.

Pierre Isabelle, Colin Cherry, and George Foster. A challenge set approach to evaluating machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2486–2496.

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Huda Khayrallah and Philipp Koehn. On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, pages 177–180.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012.

Mu Li, Yinggong Zhao, Dongdong Zhang, and Ming Zhou. Adaptive development data selection for log-linear model in statistical machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 662–670.

Marco Lui and Timothy Baldwin. langid.py: An off-the-shelf language identification tool. In *Proceedings of the Association for Computational Linguistics 2012 System Demonstrations*, pages 25–30.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. *International Conference on Learning Representations*.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1490–1495.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation.

Robert C. Moore and William Lewis. Intelligent selection of language model training data. In *Proceedings of the Association for Computational Linguistics 2010 Conference Short Papers*, pages 220–224.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980.*

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 529–533.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

Kay Peterson. 2011. Openmt12 evaluation.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771.*

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

R. Pryzant, Y. Chung, D. Jurafsky, and D. Britz. Jesc: Japanese-english subtitle corpus. *ArXiv e-prints.*

Alan Ritter, Sam Clark, Oren Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1524–1534. Association for Computational Linguistics.

Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2017. Robsut wrod reocginiton via semi-character recurrent neural network. In *AAAI*, pages 3281–3287.

Rico Sennrich. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382.

Matthias Sperber, Jan Niehues, and Alex Waibel. Toward robust neural machine translation for noisy input sequences. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 90–96.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th international conference on world wide web*, pages 613–624. International World Wide Web Conferences Steering Committee.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1483–1489.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144.*

Jiajun Zhang and Chengqing Zong. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

SimpleQuestions Nearly Solved:
A New Upperbound and Baseline Approach

Michael Petrochuk[*†] **Luke Zettlemoyer**[*]
[*]Paul G. Allen School of Computer Science & Engineering, Univ. of Washington, Seattle, WA
[†]Allen Institute for Artificial Intelligence, Seattle WA
{mikep5, lsz}@cs.washington.edu

Abstract

The SimpleQuestions dataset is one of the most commonly used benchmarks for studying single-relation factoid questions. In this paper, we present new evidence that this benchmark can be nearly solved by standard methods. First, we show that ambiguity in the data bounds performance at 83.4%; many questions have more than one equally plausible interpretation. Second, we introduce a baseline that sets a new state-of-the-art performance level at 78.1% accuracy, despite using standard methods. Finally, we report an empirical analysis showing that the upperbound is loose; roughly a quarter of the remaining errors are also not resolvable from the linguistic signal. Together, these results suggest that the SimpleQuestions dataset is nearly solved.

1 Introduction

We present new evidence that the SimpleQuestions benchmark (Bordes et al., 2015) can be nearly solved by standard methods. First, we show that ambiguity in the data bounds performance; there are often questions have more than one equally plausible interpretation. Second, we introduce a baseline that sets a new state-of-the-art performance level, despite using standard methods. Finally, we report an empirical analysis showing that the upperbound is loose.

The simple questions task involves mapping an English question (e.g. "Who wrote Gulliver's travels?") to an analogous Freebase (Bollacker et al., 2008) query, used to answer the question. The query consists of a Freebase relation (e.g. /film/film/story_by) and subject (e.g. 090s_0 [gulliver's travels]). To understand how we might bound performance on the SimpleQuestions dataset, our first contribution in this paper, consider the following examples:

a. who wrote gulliver's travels?
(film/film/story_by, 090s_0 [gulliver's travels, TV miniseries])

b. Name a character from gullivers travels.
(book/book/characters, 0btc7 [gulliver's travels])

In example (a) the phrase "Gulliver's travels" is mapped to a TV miniseries, while in (b) it is mapped to a book. This introduces an unintended ambiguity, since either mapping is equally plausible for both examples (i.e. both books and TV miniseries have authors and characters). We introduce a method for automatically identifying many such ambiguities in the data, for both the entities and relations, and show that performance is upperbounded at 83.4%.

Our second main contribution is a baseline that sets a new state-of-the-art performance level, despite using standard methods. Our approach includes (1) a CRF used to tag the mention of the subject in a question and (2) a BiLSTM used to classify the Freebase relation. Despite its simplicity, this approach achieves 78.1% accuracy for predicting Freebase subject-relation queries, surpassing all previous models.

Finally, we present an empirical error analysis of this model which shows the upperbound is loose and that there is likely not much more than 4% of performance to be gained with future work on the data. Together, these results suggest that the SimpleQuestions dataset is nearly solved. Our code and pretrained models are available at `github.com/PetrochukM/Simple-Question-Answering`.

2 Background

Single-relation factoid questions (simple questions) are common in many settings (e.g. Microsoft's search query logs (Yih et al., 2014) and WikiAnswers web questions (Fader et al., 2013)). The SimpleQuestions dataset is one of the most

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 554–558
Brussels, Belgium, October 31 - November 4, 2018.

commonly used benchmarks for studying such questions.

The Freebase knowledge graph (KG) provides the facts for answering the questions in the SimpleQuestions dataset. It includes 3 billion triples of the form (subject, relation, object) (e.g. [04b5zb_ (Fires Creek), location/location/containedby, 0f80hy (Nantahala National Forest)]). We denote such triples as (s, r, o).

The SimpleQuestions task is to rewrite questions into subject-relation pairs of the form (subject, relation), denoted in this paper as (s, r). Each pair defines a graph query that can be used to answer the corresponding natural language question. The subject is a Freebase object with a identifier called an MID (e.g. 04b5zb_). Freebase objects also typically include one or more string aliases (e.g. MID 04b5zb_ is named "Fires Creek"), which we will use later when computing our upper bounds. The relation is an object property (e.g. location/location/containedby) defined by the Freebase ontology. For example, the question "which forest is fires creek in" corresponds with the subject-relation pair (04b5zb_ [Fires Creek], location/location/containedby). Finally, the SimpleQuestions task is evaluated on subject-relation pair accuracy.

The SimpleQuestions dataset provides a set of 108,442 simple questions; each question is accompanied by a ground truth triple (s, r, o). This dataset also provides two subsets of Freebase: FB2M and FB5M.[1]

3 Dataset Ambiguity and Upperbound

The ambiguity in the SimpleQuestions dataset likely comes from the way the data was created. Annotators were shown a single Freebase triple and asked to write a question. For example, given any of the following triples:

- (0btc7 [Gulliver's Travels, Book], book/written_work/author, o3_dj [Dean Swift])
- (06znpjr [Gulliver's Travels, American film], film/film/written_by, 03whnyn [Nicholas Stroller])
- (06znpjr [Gulliver's Travels, American film], film/film/story_by, o3_dj [Dean Swift])

Subject	Description
0btc7	Gulliver's Travels (Book)
090s_0	**Gulliver's Travels (TV miniseries)**
06znpjr	Gulliver's Travels (American film)
02py9bj	Gulliver's Travels (French film)

Table 1: FB2M entities with the alias "gulliver's travels"

Relation	Count
book/written_work/author	132
film/film/written_by	67
film/film/story_by	**9**
...	...

Table 2: SimpleQuestions dataset abstract predicate "who wrote e?" relation count

The annotator might reasonably contribute the question "who wrote gulliver's travels?" However, adding all of these pairs to the data is problematic. Systems are evaluated on producing the correct subject-relation pair, and cannot learn a deterministic mapping that would get these three examples correct. In this section, we present a simple heuristic method for finding many such instances of ambiguity, and use it to upper bound performance on this benchmark.

3.1 Approach

Given an example question q with the ground truth (s, r, o), our goal is to determine the set of all other subject-relation pairs that are equally supported by the text in q.

We first determine a string alias a for the subject by matching a phrase in q with a Freebase alias for s, in our example yielding "gulliver's travels". For 97% of questions q, some string alias a exactly matched a question q phrase. We then find all other Freebase entities that share this alias a and add them to a set S, in our example S is the subject column of Table 1.

We define an abstract predicate p (e.g. "who wrote e?") as q with alias a abstracted. We determine the set of potential relations R as the relations p co-occurs with in the SimpleQuestions dataset, in our example R is the relation column of Table 2.

Finally, if there exists a subject-relation pair $(s, r) \in KG$ such that $r \in R \wedge s \in S$ we de-

---

[1] The FB2M and FB5M subsets of Freebase KG can complete 7,188,636 and 7,688,234 graph queries respectively; therefore, the FB5M subset is 6.9% larger than the FB2M subset. More previous research has cited FB2M numbers than FB5M; therefore, we report our numbers on FB2M.

fine that as an accurate semantic interpretation of q. q is unanswerable if there exists multiple valid subject-relation pairs (s, r). In our example above, the question is unanswerable because of the many different subject, relation pairs that co-occur with "gulliver's travels" and "who wrote e?"

3.2 Results

We find that 33.9% of examples in the SimpleQuestions dataset are unanswerable. In these cases, we can predict a majority baseline (i.e. always guess the most commonly seen Freebase entity or relation), yielding an upperbound of 85.2%.

Finally, we also found that 1.8% of example questions were noisy. For example, "Which book is written about?" does not reference the corresponding ground truth subject 01n7q (california). We also consider these examples unanswerable, yielding a final upperbound of 83.4%.

4 Baseline Model

Our second main contribution is a baseline that sets a new state-of-the-art performance level, despite using standard methods. Our approach includes (1) a CRF tagger to determine the subject alias, and (2) a BiLSTM to classify the relation.

4.1 Approach

Given a question q (e.g. "who wrote gulliver's travels?") our model must predict the corresponding subject-relation pair (s, r). We predict (s, r) with a pipeline that first runs top-k subject recognition and then relation classification.

We make use of two learned distributions. The subject recognition model $P(a|q)$ ranges over text spans A within the question q, in our example A includes the correct subject "gulliver's travels". This distribution is modeled with a CRF, as defined in more detail below. The relation classification model $P(r|q, a)$ will be used to select a Freebase relation r that matches q. The distribution ranges over all relations in Freebase that co-occur with a subject that is named a. It is modeled with an LSTM, that encodes q, again as defined in more detail below.

Given these distributions, we predict the final subject-relation pair (s, r) as follows. First, we determine the most likely subject alias a according to $P(a|q)$ that also matches a subject alias in the KG. We define set S as all Freebase entities named a, in our example S is the subject column of Table 1. Second, we define all potential relations R such that $\forall(s, r) \in KG\{r \in R \wedge s \in S\}$. Using the relation classification model $p(r|q, a)$, we predict the most likely relation $r_{max} \in R$.

Now, the answer candidates are subject-relation pairs such that $(s, r_{max}) \in KG\{r \in R \wedge s \in S\}$. In our example question, if r_{max} is film/film/story_by then S includes both subjects 06znpjr (Gullivers Travels, American film) and 02py9bj (Gullivers Travels, French film). Because there is no explicit linguistic signal to disambiguate this choice, we pick the subject that co-occurs most often with r_{max} in Freebase.

4.2 Model Details

Our approach requires two models, in this section we cover training and configuring these models.

Top-K Subject Recognition We model top-k subject recognition $P(a|q)$ using a linear-chain conditional random field tagger (CRF) with a conditional log likelihood loss objective. k candidates are inferred with the top-k Viterbi algorithm.

Our model is trained on a dataset of questions each with their corresponding subject alias span delimited with IO tagging. The gold standard subject alias spans are determined by heuristically matching a phrase in the question with a Freebase alias for the subject.

All hyperparameters are hand tuned and then a limited set are further tuned with grid search to increase validation accuracy. In total we evaluated at most 100 hyperparameter configurations. The word embeddings are initialized with GloVe (Pennington et al., 2014) and frozen. Adam (Kingma and Ba, 2014), initialized with a learning rate of 0.001, is employed to optimize the model weights. Finally, we halve the learning rate if the validation accuracy has not improved in 3 epochs.

Relation Classification The relation classification distribution $P(r|q, a)$ is modeled with a one layer BiLSTM batchnorm softmax classifier. The BiLSTM encodes an abstract predicate string (e.g. "who wrote e?"), as described in Section 4.1. The last LSTM output vector is provided as input to an output block consisting of batch normalization, ReLU, and softmax.

All hyperparameters are hand tuned and then a limited set are further tuned with Hyperband (Li et al., 2017) to increase validation accuracy. Hyperband is allowed at most 30 epochs per model

Previous Work	**Acc.**
Random guess (Bordes et al., 2015)	4.9
Memory NN (Bordes et al., 2015)	61.6
Attn. LSTM (He and Golub, 2016)	70.9
GRU (Lukovnikov et al., 2017)	71.2
BiGRU-CRF & BiGRU (Mohammed et al., 2017)	73.7
BiLSTM & BiGRU (Mohammed et al., 2017)	74.9
BiGRU & BiGRU (Dai et al., 2016)	75.7
CNN & Attn. CNN & BiLSTM-CRF (Yin et al., 2016)	76.4
HR-BiLSTM & CNN & BiLSTM-CRF (Yu et al., 2017)	77.0
BiLSTM-CRF & BiLSTM (Ours)	**78.1**

Table 3: Summary of past results on the SimpleQuestions benchmark along with the neural models employed. Note that an "&" indicates multiple neural models.

and a total of 1000 epochs. In total we evaluated at most 500 hyperparameter configurations. The word embeddings are initialized with FastText (Bojanowski et al., 2017) and frozen. We use the AMSGrad variant of Adam (Reddi et al., 2018), initialized with an learning rate of 0.001. Finally, we double the batch size (Smith et al., 2017) if the validation accuracy has not improved in 3 epochs.

4.3 Results

Finally, we present our results on the SimpleQuestions test set.

SimpleQuestions Task Our model achieves 78.1% accuracy on the SimpleQuestions test set, a new state-of-the-art without ensembling or data augmentation (Table 3). These results suggest that relatively standard architectures work well when carefully tuned, and approach the level set by our upper bound earlier in the paper. This further confirms the results of Mohammed et al. 2017.

Further Qualitative Analysis We also analyze the remaining errors, to point toward directions for future work.

In Section 3, we showed that questions can provide equal evidence for multiple subject-relation pairs. To remove this ambiguity, we count any of these options as correct, and our performance jumps to 91.5%.

The remaining 8.5% error comes from a number of sources. First, we find that 1.9% of examples were incorrect due to noise, as described in Section 3. To better understand the remaining 6.5% gap, we do an empirical error analysis on a sample of 50 negative examples.

First we found that for 14 of 50 cases the question provided equal linguistic evidence for both the ground truth options and the predicted subject-relation pair, similar to the dataset ambiguity found in Section 3, suggesting that our upper bound is loose. We note that Section 3 did not cover all possible question-subject-relation pair ambiguities. The approach relied on exact string matching to discover ambiguity; therefore, missing other paraphrases. For example, the abstract predicate "what classification is e" had more examples than "what classification is **the** e" allowing our approach to programmatically define more subject-relation pair ambiguities for the former predicate than the latter.

The remaining 36 of 50 cases were linguistic mistakes by our model. Among the 36 cases, we identified these error cases:

- **Low Shot** (16 of 36) The relation label was seen in the training data less than 10 times.
- **Span Identification** (14 of 36) The subject span was incorrectly labeled.
- **Noise** (2 of 36) The question did not make grammatical sense.

Together, this error analysis shows that the upperbound is loose. There is likely not much more than 4% of performance to be gained with future work on the data.

5 Conclusions and Future Work

The SimpleQuestions dataset is one of the most commonly used benchmarks for studying single-relation factoid questions. In this paper, we presented new evidence to suggest that this benchmark can be nearly solved by standard methods. These results suggest there is likely not much more than 4% to be gained with future work on the data.

Finally, other KG (e.g. Freebase) query datasets should consider providing a set of correct subject-relation pairs when there is ambiguity in the linguistic input.

---

[1] Türe and Jojic 2017 reported a 86.8% accuracy but we and Mohammed et al. 2017 have not been able to replicate their results. Wang et al. 2017 scored 77.5% but removed 0.5% of the test examples.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Zihang Dai, Lei Li, and Wei Xu. 2016. Cfo: Conditional focused neural question answering with large-scale knowledge bases. *CoRR*, abs/1606.01994.

Anthony Fader, Luke S. Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*.

Xiaodong He and David Golub. 2016. Character-level question answering with attention. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet S. Talwalkar. 2017. Hyperband: Bandit-based configuration eval- uation for hyperparameter optimization.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *WWW*.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2017. Strong baselines for simple question answering over knowledge graphs with and without neural networks. *CoRR*, abs/1712.01969.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *ICLR*.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. 2017. Don't decay the learning rate, increase the batch size. *CoRR*, abs/1711.00489.

Ferhan Türe and Oliver Jojic. 2017. No need to pay attention: Simple recurrent neural networks work! In *EMNLP*.

Runze Wang, Chen-Di Zhan, and Zhen-Hua Ling. 2017. Question answering with character-level lstm encoders and model-based data augmentation. In *CCL*.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *ACL*.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *COLING*.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *ACL*.

Phrase-Indexed Question Answering:
A New Challenge for Scalable Document Comprehension

Minjoon Seo[2,3]* **Tom Kwiatkowski**[1] **Ankur P. Parikh**[1]
Ali Farhadi[2,4,5] **Hannaneh Hajishirzi**[2]
Google AI Language[1] University of Washington[2] Clova AI, NAVER[3]
Allen Institute for AI[4] XNOR.AI[5]
`{minjoon,ali,hannaneh}@cs.uw.edu`
`{tomkwiat,aparikh}@google.com`

Abstract

We formalize a new modular variant of current question answering tasks by enforcing complete independence of the document encoder from the question encoder. This formulation addresses a key challenge in machine comprehension by requiring a standalone representation of the document discourse. It additionally leads to a significant scalability advantage since the encoding of the answer candidate *phrases* in the document can be pre-computed and *indexed* offline for efficient retrieval. We experiment with baseline models for the new task, which achieve a reasonable accuracy but significantly underperform unconstrained QA models. We invite the QA research community to engage in Phrase-Indexed Question Answering (PIQA, *pika*) for closing the gap. The leaderboard is at: `nlp.cs.washington.edu/piqa`

1 Introduction

Extractive question answering (QA) is the task of selecting an answer phrase (span) to a question given an evidence document. Due to the easiness of evaluation (compared to generative QA) and the fine-grainess of the answer (compared to sentence-level QA), it has become one of the most popular QA tasks, driven by massive new datasets such as SQuAD (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017). Current QA models heavily rely on explicitly learning the interaction between the evidence document and the question using neural attention mechanisms (Wang and Jiang, 2017; Xiong et al., 2017; Seo et al., 2017; Lee et al., 2016, *inter alia*), in which the model is fully aware of the question before or as it reads the document. As a result, despite significant advances, they have not led to the standalone representation of document discourse which is nevertheless a key goal of research in reading comprehension. Furthermore, QA models that condition the document representation on a question have the practical scalability downside that the entire model should be re-applied on the same document for every question.

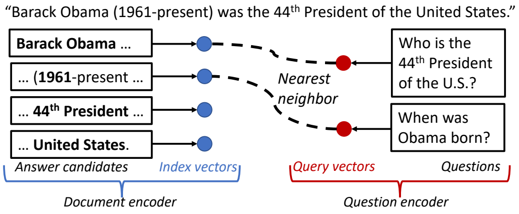

Figure 1: PIQA task for a short context sentence.

In this paper, we formalize a modular variant of the QA task, Phrase Indexed Question Answering (PIQA), that enforces complete independence between document encoder and question encoder (Figure 1). In PIQA, all documents are processed independently of any question to generate *phrase index vectors* (blue nodes in the figure) for each answer candidate (left boxes in the figure). Similarly, the questions are independently mapped to *query vectors* (red nodes in figure). Then, at inference time, the answer is obtained by retrieving the nearest indexed phrase vector to the query vector. Hence the algorithms aimed at tackling PIQA have the inherent benefit of modularity and scalability compared to current QA systems.

The task setup is analogous to how documents or sentences are retrieved in modern search engines via similarity search algorithms (Shrivastava and Li, 2015). Nevertheless, there is a key distinction that search engines index each *document* by its *content*, while PIQA requires one to index each *phrase* in documents by its *context*.

We formally define the PIQA problem and provide baseline models for the new task. Our experiments show that the constraint introduced

---

*Most work done during internship with Google AI.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 559–564
Brussels, Belgium, October 31 - November 4, 2018.

by PIQA leads to meaningful standalone document representations and practical scalability advantage, demonstrating the significance of the new task. Moreover, there is still a large gap between the baselines and the unconstrained state of the art, showing that the task is yet far from being solved. We have set up a leaderboard[1] for PIQA challenge and invite the research community to participate. We currently support SQuAD and plan to expand to other datasets as well.

2 Related Work

Reading comprehension. Massive reading comprehension question answering datasets (Hermann et al., 2015; Hill et al., 2016; Dhingra et al., 2017; Dunn et al., 2017) have driven a large number of successful neural approaches (Kadlec et al., 2016; Hu et al., 2017, *inter alia*). Choi et al. (2017); Chen et al. (2017); Clark and Gardner (2017); Min et al. (2018) tackled large-scale QA by using a fast, coarse model (e.g. TF-IDF) to retrieve few documents or sentences and then using a slower, accurate model to obtain the answer. Salant and Berant (2018) proposed to minimize (but not prohibit) the influence of question when modeling the document. Similarly to ours, Lee et al. (2016) proposed to explicitly learn the representation for each answer candidate (phrase) in the document, but it was conditioned (dependent) on the question.

Sentence retrieval. A closely related task to ours is that of retrieving a sentence/paragraph in a corpus that answers the question (Tay et al., 2017). A comprehensive survey for neural approaches in information retrieval literature is discussed in Mitra and Craswell (2017). We note that our problem is focused on phrasal answer extraction, which presents a unique challenge over sentence retrieval—the need for *context-based* representation as opposed to the *content-based* representation in the sentence-retrieval literature.

Language representation. Recently there has been a growing interest in developing natural language representations that can be transferred across tasks (Vendrov et al., 2016; Wieting et al., 2016; Conneau et al., 2017, *inter alia*). In particular, SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2017) encourage architectures that first encode the hypothesis and the premise independently before a comparator neural network is applied. Our proposed problem shares similar traits but has a stronger constraint that only inner product comparison is allowed and one needs to model phrases instead of complete sentences.

3 Phrase-Indexed Question Answering

Extractive question answering is the task of obtaining the answer $\hat{a}$ to a question $Q = \{q_1 \dots q_n\}$ given an evidence document $D = \{d_1 \dots d_m\}$, where the answer $\hat{a} = (s, e)$ indicates the start and end of a span in the document. The task is often formulated as learning the probabilistic distribution of the answer given the question and the document. In existing literature (Section 2), the distribution is mainly featurized by $\Pr(a|Q, D) \propto \exp(F_\theta(Q, D, a))$ where F_θ could be any real-valued scoring function parameterized by θ. Once θ is learned, the prediction $\hat{a}$ is obtained by

$$\hat{a} = \operatorname*{argmax}_a F_\theta(Q, D, a). \tag{1}$$

So far, most competitive designs of $F_\theta(Q, D, a)$ make use of attention connections between the words in Q and D. As a result, these models cannot yield a query independent representation of the document D. It is subsequently not possible to independently assess the document understanding capability of the model. Furthermore, $F_\theta(Q, D, a)$ needs to be re-computed for the entire document for every new question. We believe that this inefficiency precludes all current models as the candidates for end-to-end QA systems.

We propose a new task—Phrase-Indexed Question Answering (PIQA)—that addresses these issues. We enforce the *decomposability* of F_θ into two exclusive functions $G_\theta(Q), H_\theta(D, a) \in \mathbb{R}^k$. The answer distribution is then modeled by $\Pr(a|Q, D) \propto \exp(G_\theta(Q) \bullet H_\theta(D, a))$, where $\bullet$ is the inner product. The prediction is obtained by

$$\hat{a} = \operatorname*{argmax}_a G_\theta(Q) \bullet H_\theta(D, a). \tag{2}$$

In this setting, the document encoder H_θ learns models the document independently of the question. Successful question answering models that follow the structure of PIQA will have two important advantages over current QA models: full document comprehension and scalablity.

Full document comprehension. Language understanding ability is widely associated with learning a good standalone representation of text (or its

---

[1] `nlp.cs.washington.edu/piqa`

components such as phrases) independent of the end task (Bowman et al., 2015). Under PIQA constraints, the document encoder H_θ learns the representation of the answer candidate phrases a in the document D independent of the question. In order to correctly answer questions, these phrase representations (index vectors) need to correctly encode their meaning with respect to their context. Therefore, PIQA constraint enforces evaluating research in document comprehension and phrase representation learning.

Scalability. Models that adhere to the PIQA constraint only need to be run once for each document, regardless of the number of questions asked. To answer a question, the model then just needs to encode the question and compare it to each of the answer candidates via the inner product in Equation 2. Implemented naively, computing a single inner product for each answer candidate is more efficient than building a new document encoding; after the documents are pre-encoded, Equation 2 is $O(k)$ time per word where k is the vector size (most neural models require $O(k^2)$ per word for matrix multiplications).

More importantly, PIQA also permits an approximate solution in sublinear time using asymmetric locality-sensitive hashing (aLSH) (Shrivastava and Li, 2014, 2015), through which Equation 2 can be approximated for N answer candidates with $O(kN^\rho \log N)$ time, where $\rho < 1$ is a function of the approximation factor and the properties of the hash functions. We argue that this type of approach will be essential for the development of real world QA systems, where the number of potential answers N is extremely large.

4 Baseline Models

We introduce several baselines for PIQA that are motivated by related literature.

For all (neural) baselines, we represent the words in D and Q with one of three embedding mechanisms: CharCNN (Kim, 2014) + GloVe (Pennington et al., 2014), and ELMo (Peters et al., 2018). We follow the majority of the related literature and apply bidirectional LSTMs (Hochreiter and Schmidhuber, 1997) to these embeddings to build the context-aware representations of the document $\mathbf{D} = \{\mathbf{d}_1 \ldots \mathbf{d}_m\}$ and question $\mathbf{Q} = \{\mathbf{q}_1 \ldots \mathbf{q}_n\}$, where the forward & backward LSTM outputs are concatenated to get a single word representation, i.e. $\mathbf{d}_i, \mathbf{q}_i \in \mathbb{R}^{2k}$ where k is the hidden state size of LSTMs.

PIQA disallows cross-attention between document and question. However, we can still benefit from self-attention, which has become crucial for machine translation (Vaswani et al., 2017) and QA (Huang et al., 2018; Yu et al., 2018). In all of our baselines, each variable-length question is collapsed into a fixed length vector via the sum $\mathbf{q}^{\mathrm{SA}} = \sum_i u_i \mathbf{q}_i$ where $\mathbf{u} = \{u_1 \ldots u_n\}$ is a vector containing a single weight for each word in the question. Similarly, we experiment with document side self attention to represent each document word $\mathbf{d}_j$ as a weighted sum of itself and all neighboring words $\mathbf{d}_j^{\mathrm{SA}} = \sum_i h_i^j \mathbf{d}_j$. The weight vectors $\mathbf{u}$ and $\mathbf{h}^j$ are calculated as

$$\mathbf{u} = \mathrm{softmax}_i(\mathbf{w}^\top \mathbf{q}_i)$$
$$\mathbf{h}^j = \mathrm{softmax}_i(R_\theta(\mathbf{D}, j)^\top K_\theta(\mathbf{D}, i))$$

where R_θ, and K_θ are trainable neural networks with the same ouptut size, and $\mathbf{w} \in \mathbb{R}^{2k}$ is a trainable weight vector. We use independent BiLSTMs with hidden state size k (i.e. the output size is $2k$) to model both R_θ and K_θ. That is, $R_\theta(\mathbf{D}, j)$ is the j-th output of BiLSTM on top of $\mathbf{D}$, and we similarly define K_θ with unshared parameters.

For all (neural) baselines, the question is represented using the concatenation of two copies of $\mathbf{q}^{\mathrm{SA}}$, one that should have high inner product with the vector for the answer's start span and another that should have high inner product with the vector for the answer's end. Thus, Equation 2's $G_\theta(Q) = [\mathbf{q}_s^{\mathrm{SA}}, \mathbf{q}_e^{\mathrm{SA}}]$ where the subscripts s (start) and e (end) imply that different sets of parameters were used. Now we define several baselines.

LSTM baseline. An answer candidate $a = (s, e)$ is represented using the LSTM outputs at its endpoints: from Equation 2, $H_\theta(D, (s, e)) = [\mathbf{d}_s, \mathbf{d}_e] \in \mathbb{R}^{4k}$ and $G_\theta(Q) = [\mathbf{q}_s^{\mathrm{SA}}, \mathbf{q}_e^{\mathrm{SA}}] \in \mathbb{R}^{4k}$.

LSTM+SA baseline. The LSTM outputs are augmented with the endpoint representations that come out of the document's self-attention (SA): $H_\theta(D, (s, e)) = [\mathbf{d}_s, \mathbf{d}_s^{\mathrm{SA}}, \mathbf{d}_e, \mathbf{d}_e^{\mathrm{SA}}] \in \mathbb{R}^{8k}$ and $G_\theta(Q) = [\mathbf{q}_{s1}^{\mathrm{SA}}, \mathbf{q}_{s2}^{\mathrm{SA}}, \mathbf{q}_{e1}^{\mathrm{SA}}, \mathbf{q}_{e2}^{\mathrm{SA}}] \in \mathbb{R}^{8k}$.

TF-IDF. We lastly include a purely TF-IDF-based model, where each answer candidate phrase is associated with a bag of neighbor words within a distance of 7. Then the BOW vector is normalized via TF-IDF and indexed. When the query comes in, its TF-IDF vector is queried on the indexed phrases to yield the answer.

Constraint	Model	F1 (%)	EM (%)
PI	TF-IDF	15.0	3.9
	LSTM	57.2	46.8
	LSTM+SA	59.8	49.0
	LSTM+ELMo	60.9	50.9
	LSTM+SA+ELMo	62.7	52.7
None	Rajpurkar et al. (2016)	51.0	40.0
	Yu et al. (2018)	89.3	82.5

Table 1: Performance on SQuAD dev set with the PIQA constraint (top), and without the constraint (bottom). See Section 4 for the description of the terms.

For training the (neural) models, we minimize the negative log probability of getting the correct answer: the loss function for each example (D, Q, a^*) is $L(\theta) = -\log \Pr(a^*|D, Q)$ where a^* is the correct answer.

5 Experiments

We impose the independence restrictions from PIQA on the Stanford Question Answering Dataset[2]. We only consider answer spans with length ≤ 7. We use the hidden state size (k) of 128, which results in a 512D ($4k$) and 1024D ($8k$) vector for each phrase in LSTM and LSTM+SA, respectively. The default embedding model is CharCNN concatenated with 200D GloVe, with an option to append ELMo vectors following the same setup for SQuAD experiments discussed in Peters et al. (2018). We use a batch size of 64 and train for 20 epochs with the default Adam optimizer (Kingma and Ba, 2015), and take the best model on the validation set during training.

Results. Table 1 shows the results for the PIQA baselines (top) and the unconstrained state of the art (bottom). First, the TF-IDF model performs poorly, which signifies the limitations of traditional document retrieval models for the task. Second, we note that the addition of self-attention makes a significant impact on results, improving F1 by 2.6%. Next, we see that adding ELMo gives 3.7% and 2.9% improvement on F1 for LSTM and LSTM+SA models, respectively. Lastly, the best PIQA baseline model is 11.7% higher than the first (unconstrained) baseline model (Rajpurkar et al., 2016) and 26.6% lower than the state of the art (Yu et al., 2018). This gives us a reasonable starting point of the new task and a significant gap to close for future work.

- According to the **American Library Association**, this makes... - ...tasked with drafing a **European Charter of Human Rights**,...
- **The LM engines** were successfully test-fired and restarted, ... - **Steam turbines** were extensively applied...
- ...primarily accomplished through the **ductile stretching and thinning**. - ...directly derived from **the homogeneity or symmetry of space**...

Table 2: Most similar phrase pairs from disjoint sets of documents. Bold print is the phrase, and non-bold is its context.

Phrase representations. Since PIQA models encode all answer candidates into the same space, we expect similar answer candidates to have high inner products with one another. Table 2 shows pairs of answer candidates that come from different documents in SQuAD, but that have similar encodings (high inner product). We observe that phrase representations learned through the PIQA task capture different interesting characteristics of the phrases. In all three rows, we can see that the phrase pairs seem to fit into natural categories: national, or multi-national organizational constructs; mechanical engines; and mechanical properties, respectively. This suggests that the model has learned interesting typing information above the word level. The second and third rows also indicate that the model has learned a rich representation of context. This is particularly obvious in the third row where the two phrases are lexically dissimilar, but preceded by the similar contexts *'primarily accomplished through'* and *'directly derived from'*. We believe that this analysis, while not complete, points toward exciting future lines of work in learning highly contextualized phrase representations through question answering.

Scalability. PIQA can also gain massive execution time speedups once the documents are pre-encoded: in our simple benchmark on a consumer-grade CPU and NumPy (for LSTM+SA model, 1024D vectors), one can easily perform exact search over 1 million document words per second. BiDAF (Seo et al., 2017), an open-sourced and relatively light QA model reaching 77.5% F1 (66.5% EM), can process less than 1k document words per second with an equivalent computing power (after pre-encoding the document as much as possible), which is more than 1,000x slower.[3]

---

[2] PIQA paradigm can be also extended to other extractive QA datasets.

[3] The difference will be even higher with a dedicated similarity search package such as Faiss (Johnson et al., 2017) or approximate search (Section 3).

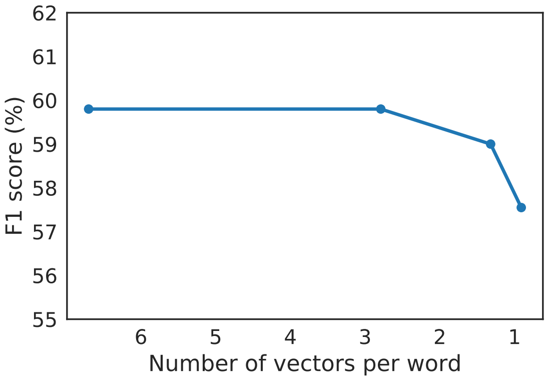

Figure 2: F1 score versus number of vectors per word for LSTM+SA. Answer candidates have been filtered with varying threshold on an independent classifier learned on the candidate representations.

It is also important to consider the memory cost for storing a vector representation of each of the answer candidates. We train an independent single-layer perceptron classifier that predicts whether the phrase encoding is likely to be a good one. By varying a threshold on the score assigned by this classifier, we can filter answer candidates prior to storage. Figure 2 illustrates the trade-off between accuracy and memory (measured in mean number of vectors per document word) resulting from this filtering procedure for the LSTM+SA model. We observe that 1.3 vectors (candidates) per word on average reaches $> 98\%$ of the model's F1 accuracy. This is equivalent to 5.2 KB per word with 1024D (4 KB) float vectors, or around 15 TB for the entire English Wikipedia (3 billion words). Future work will also involve creating a better classifier (i.e. improving the trade-off curve in Figure 2) for determining which phrase vectors to store.

6 Conclusion and Future Work

We introduced Phrase-Indexed Question Answering (PIQA), a new variant of the extractive question answering task that requires documents and question encoded completely independently and that they only interact each other via inner product. We argued that building a question-agnostic document encoder for question answering should be an important consideration for those in the QA community with the research goal of learning a model that reads and comprehends documents. Furthermore, the imposed constraint of the task implies a sublinear scalability benefit. Given that SQuAD models have recently outperformed humans, PIQA formulation motivates a new challenge for which we hope that the community's effort gradually closes the gap between our constrained baselines and the unconstrained models.

Acknowledgments

This research was supported by ONR (N00014-18-1-2826), NSF (IIS 1616112), Allen Distinguished Investigator Award, and gifts from Google, Allen Institute for AI, Amazon, and Bloomberg. We thank the anonymous reviewers for their helpful comments.

References

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *ACL*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*.

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

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *arXiv preprint arXiv:1705.02798*.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2018. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *ICLR*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *ACL*.

Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Shimi Salant and Jonathan Berant. 2018. Contextualized word representations for reading comprehension. In *NAACL*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Anshumali Shrivastava and Ping Li. 2014. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *NIPS*.

Anshumali Shrivastava and Ping Li. 2015. Improved asymmetric locality sensitive hashing (alsh) for maximum inner product search (mips). In *UAI*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. Hyperqa: Hyperbolic embeddings for fast and efficient ranking of question answer pairs. *arXiv preprint arXiv:1707.07847*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *ICLR*.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *ICLR*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *ICLR*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *ICLR*.

Ranking Paragraphs for Improving Answer Recall in Open-Domain Question Answering

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko and **Jaewoo Kang***
Department of Computer Science and Engineering, Korea University
{jinhyuk_lee, ysj5419, hyunjae-kim}@korea.ac.kr
{gomi1503, kangj}@korea.ac.kr

Abstract

Recently, open-domain question answering (QA) has been combined with machine comprehension models to find answers in a large knowledge source. As open-domain QA requires retrieving relevant documents from text corpora to answer questions, its performance largely depends on the performance of document retrievers. However, since traditional information retrieval systems are not effective in obtaining documents with a high probability of containing answers, they lower the performance of QA systems. Simply extracting more documents increases the number of irrelevant documents, which also degrades the performance of QA systems. In this paper, we introduce Paragraph Ranker which ranks paragraphs of retrieved documents for a higher answer recall with less noise. We show that ranking paragraphs and aggregating answers using Paragraph Ranker improves performance of open-domain QA pipeline on the four open-domain QA datasets by 7.8% on average.

1 Introduction

With the introduction of large scale machine comprehension datasets, machine comprehension models that are highly accurate and efficient in answering questions given raw texts have been proposed recently (Seo et al., 2016; Xiong et al., 2016; Wang et al., 2017c). While conventional machine comprehension models were given a paragraph that always contains an answer to a question, some researchers have extended the models to an open-domain setting where relevant documents have to be searched from an extremely large knowledge source such as Wikipedia (Chen et al., 2017; Wang et al., 2017a). However, most of the open-domain QA pipelines depend on traditional information retrieval systems which use TF-IDF rankings (Chen et al., 2017; Wang et al., 2017b). Despite the efficiency of the traditional retrieval systems, the documents retrieved and ranked at the top by such systems often do not contain answers to questions. However, simply increasing the number of top ranked documents to find answers also increases the number of irrelevant documents. The tradeoff between reading more documents and minimizing noise is frequently observed in previous works that defined the N number of top documents as a hyper-parameter to find (Wang et al., 2017a).

In this paper, we tackle the problem of ranking the paragraphs of retrieved documents for improving the answer recall of the paragraphs while filtering irrelevant paragraphs. By using our simple but efficient Paragraph Ranker, our QA pipeline considers more documents for a high answer recall, and ranks paragraphs to read only the most relevant ones. The work closest to ours is that of Wang et al. (2017a). However, whereas their main focus is on re-ranking retrieved sentences to maximize the rewards of correctly answering the questions, our focus is to increase the answer recall of paragraphs with less noise. Thus, our work is complementary to the work of Wang et al. (2017a).

Our work is largely inspired by the field of information retrieval called *Learning to Rank* (Liu et al., 2009; Severyn and Moschitti, 2015). Most learning to rank models consist of two parts: encoding networks and ranking functions. We use bidirectional long short term memory (Bi-LSTM) as our encoding network, and apply various ranking functions proposed by previous works (Severyn and Moschitti, 2015; Tu et al., 2017). Also, as the time and space complexities of ranking paragraphs are much larger than those of ranking sentences (Severyn and Moschitti, 2015), we resort to negative sampling (Mikolov et al., 2013) for an efficient training of our Paragraph Ranker.

---

* Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 565–569
Brussels, Belgium, October 31 - November 4, 2018.

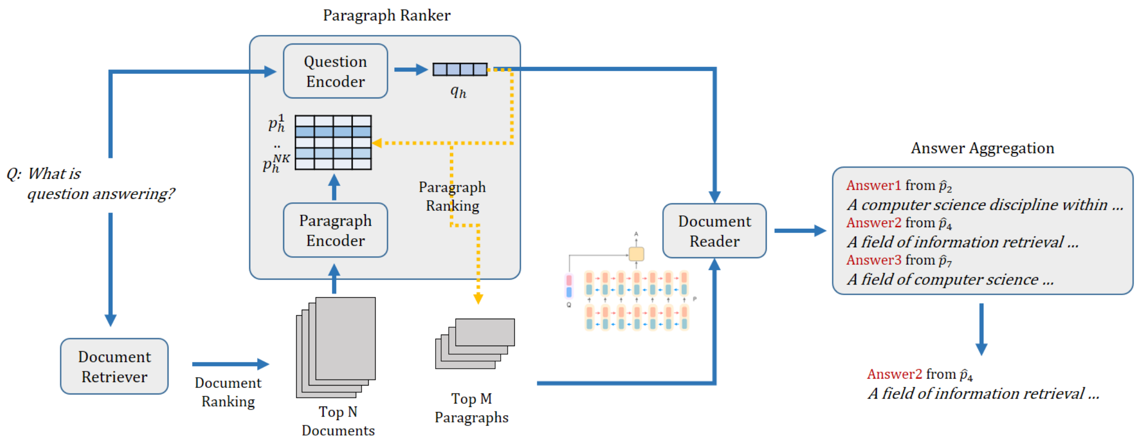

Figure 1: Our proposed open-domain QA pipeline with Paragraph Ranker

Our pipeline with Paragraph Ranker improves the exact match scores on the four open-domain QA datasets by 7.8% on average. Even though we did not further customize Document Reader of DrQA (Chen et al., 2017), the large improvement in the exact match scores shows that future researches would benefit from ranking and reading the more relevant paragraphs. By a qualitative analysis of ranked paragraphs, we provide additional evidence supporting our findings.

2 Open-Domain QA Pipeline

Most open-domain QA systems are constructed as pipelines that include a retrieval system and a reader model. We additionally built Paragraph Ranker that assists our QA pipeline for a better paragraph selection. For the retrieval system and the reader model, we used Document Retriever and Document Reader of Chen et al. (2017).[1] The overview of our pipeline is illustrated in Figure 1.

2.1 Paragraph Ranker

Given N number of documents retrieved from Document Retriever, we assume that each document contains K number of paragraphs on average. Instead of feeding all NK number of paragraphs to Document Reader, we select only M number of paragraphs using Paragraph Ranker. Utilizing Paragraph Ranker, we safely increase N for a higher answer recall, and reduce the number of paragraphs to read by selecting only top ranked paragraphs.

Given the retrieved paragraphs P_i where i ranges from 1 to NK, and a question Q, we encode each paragraph and the question using two separate RNNs such as Bi-LSTM. Representations of each paragraph and the question are calculated as follows:

$$p_h^i = \text{BiLSTM}_p(\text{E}(P_i)) \quad q_h = \text{BiLSTM}_q(\text{E}(Q))$$

where BiLSTM$(\cdot)$ returns the concatenation of the last hidden state of forward LSTM and the first hidden state of backward LSTM. E$(\cdot)$ converts tokens in a paragraph or a question into pretrained word embeddings. We use GloVe (Pennington et al., 2014) for the pretrained word embeddings.

Once each paragraph and the question are represented as p_h^i and q_h, we calculate the probability of each paragraph to contain an answer of the question as follows:

$$p(P_i|Q) = \frac{1}{1 + e^{-s(p_h^i, q_h)}}$$

where we have used similarity function $s(\cdot,\cdot)$ to measure the probability of containing answer to the question Q in the paragraph P_i. While Wang and Jiang (2015) adopted high capacity models such as Match-LSTM for measuring the similarity between paragraphs and questions, we use much simpler scoring functions to calculate the similarity more efficiently. We tested three different scoring functions: 1) the dot product of p_h^i and q_h, 2) the bilinear form ${p_h^i}^T W q_h$, and 3) a multilayer perceptron (MLP) (Severyn and Moschitti, 2015). While utilizing MLP takes much more time than the other two functions, recall of MLP was similar to that of the dot product. Also, as recall of the bilinear form was worse than that of the dot product, we use the dot product as our scoring function.

---

[1] https://github.com/facebookresearch/DrQA

Due to the large size of NK, it is difficult to train Paragraph Ranker on all the retrieved paragraphs.[2] To efficiently train our model, we use a negative sampling of irrelevant paragraphs (Mikolov et al., 2013). Hence, the loss function of our model is as follows:

$$\begin{aligned} J(\Theta) = &- \log p(P_i|Q) \\ &- E_{k \sim p_n}[\log(1 - p(P_k|Q))] \end{aligned}$$

where k indicates indexes of negative samples that do not contain the answer, and Θ denotes trainable parameters of Paragraph Ranker. The distribution of negative samples are defined as p_n. We use the distribution of all the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) training paragraphs as p_n .

Based on the rank of each paragraph from Paragraph Ranker and the rank of source document from Document Retriever, we collect top M paragraphs to read. We combine the ranks by the multiplication of probabilities $p(P_i|Q)$ and $\tilde{p}(D_i|Q)$ to find most relevant paragraphs where $\tilde{p}(D_i|Q)$ denotes TF-IDF score of a source document D_i.

2.2 Answer Aggregation

We feed M paragraphs to Document Reader to extract M answers. While Paragraph Ranker increases the probability of including answers in the top M ranked paragraphs, aggregation step should determine the most probable answer among the M extracted answers. Chen et al. (2017) and Clark et al. (2017) used the unnormalized answer probability from the reader. However, as the unnormalized answer probability is very sensitive to noisy answers, Wang et al. (2017b) proposed a more sophisticated aggregation methods such as coverage-based and strength-based re-rankings.

In our QA pipeline, we incorporate the coverage-based method by Wang et al. (2017b) with paragraph scores from Paragraph Ranker. Although strength-based answer re-ranking showed good performances on some datasets, it is too complex to efficiently re-rank M answers. Given the M candidate answers $[A_1, ..., A_M]$ from each paragraph, we aggregate answers as follows:

$$\begin{aligned} \hat{A} &= \arg\max_{A_i} p(A_i|Q) \\ &= \arg\max_{A_i} \tilde{p}(A_i|P_i, Q)^{\alpha} p(P_i|Q)^{\beta} \tilde{p}(D_i|Q)^{\gamma} \end{aligned} \tag{1}$$

where $\tilde{p}(A_i|P_i, Q)$ denotes the unnormalized answer probability from a reader given the paragraph P_i and the question Q. Importance of each score is determined by the hyperparamters α, β, and γ. Also, we add up all the probabilities of the duplicate candidate answers for the coverage-based aggregation.

3 Experiments

3.1 Datasets

We evaluate our pipeline with Paragraph Ranker on the four open-domain QA datasets. Wang et al. (2017a) termed SQuAD without relevant paragraphs for the open-domain QA as **SQuAD$_{\text{OPEN}}$**, and we use the same term to denote the open-domain setting SQuAD. **CuratedTrec** (Baudiš and Šedivỳ, 2015) was created for TREC open-domain QA tasks. **WebQuestions** (Berant et al., 2013) contains questions from Google Suggest API. **WikiMovies** (Miller et al., 2016) contains questions regarding movies collected from OMDb and the MovieLens database. We pretrain Document Reader and Paragraph Ranker on the SQuAD training set.[3]

3.2 Implementation Details

Paragraph Ranker uses 3-layer Bi-LSTM networks with 128 hidden units. On SQuAD$_{\text{OPEN}}$ and CuratedTrec, we set α, β, and γ of Paragraph Ranker to 1. Due to the different characteristics of questions in WebQuestion and WikiMovies, we find α, β, and γ based on the validation QA pairs of the two datasets. We use $N = 20$ for the number of documents to retrieve and $M = 200$ for the number of paragraphs to read for all the four datasets. We use Adamax (Kingma and Ba, 2014) as the optimization algorithm. Dropout is applied to LSTMs and embeddings with $p = 0.4$.

3.3 Results

In our experiments, **Paragraph Ranker** ranks only paragraphs, and answers are determined by unnormalized scores of the answers. **Paragraph Ranker + Answer Agg.** sums up the unnormalized probabilities of duplicate answers (i.e., $\beta = \gamma = 0$). **Paragraph Ranker + Full Agg.** aggregates answers using Equation 1 with the coverage-based aggregation.

---

[2] $NK \approx 350$ when $N = 5$ in SQuAD QA pairs.

[3] On SQuAD development set, pretrained Document Reader achieves 69.1% EM, and pretrained Paragraph Ranker achieves 96.7% recall on the top 5 paragraph .

	SQuAD$_{\text{OPEN}}$		CuratedTrec		WebQuestions		WikiMovies	
Model	EM	Recall	EM	Recall	EM	Recall	EM	Recall
DrQA (Chen et al., 2017)	27.1	77.8	19.7	86.0	11.8	74.4	24.5	70.3
DrQA + Fine-tune	28.4	-	25.7	-	19.5	-	34.3	-
DrQA + Multitask	29.8	-	25.4	-	20.7	-	36.5	-
R^3 (Wang et al., 2017a)	29.1	-	28.4	-	17.1	-	38.8	-
Par. Ranker	28.5	83.1	26.8	91.4	18.0	70.7	33.4	79.7
Par. Ranker + Answer Agg.	28.9	-	28.2	-	18.4	-	33.9	-
Par. Ranker + Full Agg.	**30.2**	-	**35.4**	-	**19.9**	-	**39.1**	-

Table 1: Open-domain QA results on four QA datasets. Best scores including those of the Multitask model are underlined. Bold texts denote best scores excluding those of the Multitask model.

Question #1	*What position does Von Miller play?* (SQuAD$_{\text{OPEN}}$)
Answer	*linebacker*, *linebacker*, *linebacker*
Doc. Retriever	(Top-1 document) *Ferdinand Miller, from 1875 von Miller ... was an ore caster, ...* *Miller was born and died in Munich. He was the son of the artisan and First ...* *Ferdinand was simultaneously ennobled. Ferdinand's younger brother was the ...*
Par. Ranker	(Top-1 paragraph) *The two teams exchanged field goals ... with a 48-yarder by ...* (Top-2 paragraph) *Luck was strip-sacked by Broncos'* ***linebacker*** *Von Miller ...* (Top-3 paragraph) *Broncos'* ***linebacker*** *Von Miller forced a fumble off RGIII ...*

Table 2: Top ranked paragraphs by Paragraph Ranker based on SQuAD$_{\text{OPEN}}$

In Table 1, we summarize the performance and recall of each model on open-domain QA datasets. We define recall as the probability of read paragraphs containing answers. While Reinforced Reader-Ranker (R^3) (Wang et al., 2017a) performs better than DrQA on the three datasets (SQuAD$_{\text{OPEN}}$, CuratedTrec, WikiMovies), Paragraph Ranker + Full Agg. outperforms both DrQA and R^3. Paragraph Ranker + Full Agg. achieved 3.78%, 24.65%, 2.05%, 0.77% relative improvements in terms of EM on SQuAD$_{\text{OPEN}}$, CuratedTrec, WebQuestion, and WikiMovies, respectively (7.8% on average). It is noticeable that our pipeline with Paragraph Ranker + Full Agg. greatly outperforms DrQA + Multitask in SQuAD$_{\text{OPEN}}$ and CuratedTrec.

3.4 Analysis

In Table 2, we show 3 random paragraphs of the top document returned by Document Retriever, and the top 3 paragraphs ranked by Paragraph Ranker from the top 40 documents. As Document Retriever largely depends on matching of query tokens with document tokens, the top ranked document is usually the document with most tokens matching the query. However, Question 1 includes the polysemy of the word "play" which makes it more difficult for Document Retriever to perform effectively. Our Paragraph Ranker well understands that the question is about a sports player not a musician. The top 1-3 paragraphs for the second question came from the 30th, 7th, and 6th documents, respectively, ranked by Document Retriever. This shows that increasing number of documents to rank helps Paragraph Ranker find more relevant paragraphs.

4 Conclusion

In this paper, we present an open-domain question answering pipeline and proposed Paragraph Ranker. By using Paragraph Ranker, the QA pipeline benefits from increased answer recall from paragraphs to read, and filters irrelevant documents or paragraphs. With our simple Paragraph Ranker, we achieve state-of-the-art performances on the four open-domain QA datasets with large margins. As future works, we plan to further improve Paragraph Ranker based on the researches on learning to rank.

Acknowledgement

This research was supported by National Research Foundation of Korea (NRF-2017R1A2A1A17069645, NRF-2017M3C4A7065887), and the Korean MSIT (Ministry of Science and ICT) under the National Program for Excellence in SW (2015-0-00936) supervised by the IITP (Institute for Information & communications Technology Promotion)

References

Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the yodaqa system. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 222–228. Springer.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

Zhucheng Tu, Matt Crane, Royal Sequiera, Junchen Zhang, and Jimmy Lin. 2017. An exploration of approaches to integrating neural reranking models in multi-stage ranking architectures. *arXiv preprint arXiv:1707.08275*.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2017a. R3: Reinforced reader-ranker for open-domain question answering. *arXiv preprint arXiv:1709.00023*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017b. Evidence aggregation for answer re-ranking in open-domain question answering. *arXiv preprint arXiv:1711.05116*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017c. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Cut to the Chase: A Context Zoom-in Network for Reading Comprehension

Sathish Indurthi[1] **Seunghak Yu**[1,*] **Seohyun Back**[1] **Heriberto Cuayáhuitl**[1,2]

[1] Samsung Research, Seoul, Korea
[2] School of Computer Science, University of Lincoln, Lincoln, United Kingdom
{s.indurthi, seunghak.yu, scv.back}@samsung.com
HCuayahuitl@lincoln.ac.uk

Abstract

In recent years many deep neural networks have been proposed to solve Reading Comprehension (RC) tasks. Most of these models suffer from reasoning over long documents and do not trivially generalize to cases where the answer is not present as a span in a given document. We present a novel neural-based architecture that is capable of extracting relevant regions based on a given question-document pair and generating a well-formed answer. To show the effectiveness of our architecture, we conducted several experiments on the recently proposed and challenging RC dataset 'NarrativeQA'. The proposed architecture outperforms state-of-the-art results (Tay et al., 2018) by 12.62% (ROUGE-L) relative improvement.

1 Introduction

Building Artificial Intelligence (AI) algorithms to teach machines to read and to comprehend text is a long-standing challenge in Natural Language Processing (NLP). A common strategy for assessing these AI algorithms is by treating them as RC tasks. This can be formulated as finding an answer to a question given the document(s) as evidence. Recently, many deep-learning based models (Seo et al., 2017; Xiong et al., 2017; Wang et al., 2017; Shen et al., 2017; Clark and Gardner, 2017) have been proposed to solve RC tasks based on the SQuAD (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017) datasets, reaching human level performance. A common approach in these models is to score and/or extract candidate spans conditioned on a given question-document pair.

Most of these models have limited applicability to real problems for the following reasons. They do not generalize well to scenarios where the answer is not present as a span, or where several discontinuous parts of the document are required to form the answer. In addition, unlike humans, they can not easily skip through irrelevant parts to comprehend long documents (Masson, 1983).

To address the issues above we develop a novel context zoom-in network (ConZNet) for RC tasks, which can skip through irrelevant parts of a document and generate an answer using only the relevant regions of text. The ConZNet architecture consists of two phases. In the first phase we identify the relevant regions of text by employing a reinforcement learning algorithm. These relevant regions are not only useful to generate the answer, but can also be presented to the user as supporting information along with the answer. The second phase is based on an encoder-decoder architecture, which comprehends the identified regions of text and generates the answer by using a residual self-attention network as encoder and a RNN-based sequence generator along with a pointer network (Vinyals et al., 2015) as the decoder. It has the ability to generate better well-formed answers not verbatim present in the document than span prediction models.

Recently, there have been several attempts to adopt condensing documents in RC tasks. Wang et al. (2018) retrieve a relevant paragraph based on the question and predict the answer span. Choi et al. (2017) select sentence(s) to make a summary of the entire document with a feed-forward network and generate an answer based on the summary. Unlike existing approaches, our method has the ability to select relevant regions of text not just based on the question but also on how well regions are related to each other. Moreover, our decoder combines span prediction and sequence generation. This allows the decoder to copy words from the relevant regions of text as well as to generate words from a fixed vocabulary.

We evaluate our model using one of the challenging RC datasets, called 'NarrativeQA', which

---

[*] To whom correspondence should be addressed.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 570–575
Brussels, Belgium, October 31 - November 4, 2018.

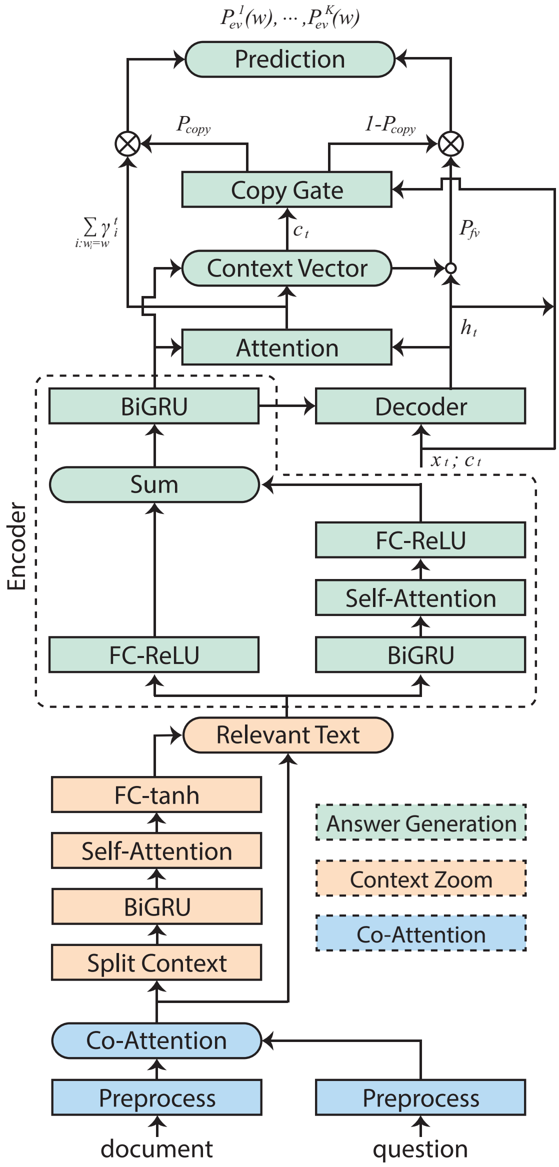

Figure 1: The proposed ConZNet architecture

was released recently by Kočiskỳ et al. (2017). Experimental results show the usefulness of our framework for RC tasks and we outperform state-of-the-art results on this dataset.

2 Proposed Architecture

An overview of our architecture is shown in Figure 1, which consists of two phases. First, the identification of relevant regions of text is computed by the **Co-attention** and **Context Zoom** layers as explained in Sections 2.1 and 2.2. Second, the comprehension of identified regions of text and output generation is computed by **Answer Generation** block as explained in Section 2.3.

2.1 Co-attention layer

The words in the document, question and answer are represented using pre-trained word embeddings (Pennington et al., 2014). These word-based embeddings are concatenated with their corresponding char embeddings. The char embeddings are learned by feeding all the characters of a word into a Convolutional Neural Network (CNN) (Kim, 2014). We further encode the document and question embeddings using a shared bi-directional GRU (Cho et al., 2014) to get context-aware representations.

We compute the co-attention between document and question to get question-aware representations for the document by using tri-linear attention as proposed by Seo et al. (2017). Let d_i be the vector representation for the document word i, q_j be the vector for the question word j, and l_d and l_q be the lengths of the document and question respectively. The tri-linear attention is calculated as

$$a_{ij} = w_d d_i + w_q q_j + w_{dq}(d_i \odot q_j), \qquad (1)$$

where w_d, w_q, and w_{dq} are learnable parameters and $\odot$ denotes the element-wise multiplication.

We compute the attended document word $\tilde{d}_i$ by first computing $\lambda_i = softmax(a_{i:})$ and followed by $\tilde{d}_i = \sum_{j=1}^{l_q} \lambda_{ij} q_j$. Similarly, we compute a question to document attention vector $\tilde{q}$ by first computing $b = softmax(\max(a_{i:}))$ and followed by $\tilde{q} = \sum_{i=i}^{l_d} d_i b_i$. Finally, d_i, $\tilde{d}_i$, $d_i \odot \tilde{d}_i$, $\tilde{d}_i \odot \tilde{q}$ are concatenated to yield a query-aware contextual representation for each word in the document.

2.2 Context Zoom Layer

This layer finds relevant regions of text. We use reinforcement learning to do that, with the goal of improving answer generation accuracy – see Section 2.4.

The *Split Context* operation splits the attended document vectors into sentences or fixed size chunks (useful when sentence tokenization is not available for a particular language). This results in n text regions with each having length l_k, where $l_d = \sum_{k=1}^{n} l_k$. We then get the representations, denoted as z_k, for each text region by running a *BiGRU* and concatenating the last states of the forward and backward GRUs.

The text region representations, z_k, encode how well they are related to the question, and their surrounding context. Generating an answer may depend on multiple regions, and it is important for each text region to collect cues from other regions which are outside of their surroundings. We can compute this by using a *Self-Attention* layer. It is a special case of co-attention where both operands (d_i and q_j) are the text fragment itself, computed by setting $a_{ij} = -\infty$ when $i = j$ in Eq. 1.

These further self-attended text region representations, $\tilde{z}_k$, are passed through a linear layer with *tanh* activation and *softmax* layer as follows:

$$u = tanh(W_c[\tilde{z}_1, \cdots, \tilde{z}_n] + b_c), \tag{2}$$

$$\psi = softmax(u), \tag{3}$$

where ψ is the probability distribution of text regions, which is the evidence used to generate the answer. The policy of the reinforcement learner is defined as $\pi(r|u; \theta_z) = \psi_r$, where ψ_r is the probability of a text region r (agent's action) being selected, u is the environment state as defined in Eq. 2, and θ_z are the learnable parameters. During the training time we sample text regions using ψ, in inference time we follow greedy evaluation by selecting most probable region(s).

2.3 Answer Generation

This component is implemented based on the encoder-decoder architecture of (Sutskever et al., 2014). The selected text regions from the *Context Zoom* layer are given as input to the encoder, where its output is given to the decoder in order to generate the answer.

The encoder block uses residual connected self-attention layer followed by a BiGRU. The selected relevant text regions ($\in \psi_r$) are first passed through a separate BiGRU, then we apply a self-attention mechanism similar to the *Context Zoom* layer followed by a linear layer with ReLU activations. The encoder's output consists of representations of the relevant text regions, denoted by e_i.

The decoder block is based on an attention mechanism (Bahdanau et al., 2015) and a copy mechanism by using a pointer network similar to (See et al., 2017). This allows the decoder to predict words from the relevant regions as well as from the fixed vocabulary. At time step t, the decoder predicts the next word in the answer using the attention distribution, context vector and current word embedding. The attention distribution and context vector are obtained as follows:

$$o_i^t = v^T tanh(W_e e_i + W_h h_t + b_o), \tag{4}$$

$$\gamma^t = softmax(o_i^t), \tag{5}$$

where h_t is hidden state of the decoder, v, W_e, W_h, b_o are learnable parameters. The γ^t represents a probability distribution over words of relevant regions e_i. The context vector is given by $c_t = \sum_i \gamma_i^t e_i$.

	AW	AS/MS
document	659	28/66
question	10	n/a
answer	5	n/a

	NE
Train	32,747
Dev	3,461
Test	10,557

Table 1: NarrativeQA statistics. **AW, AS/MS** are defined as avg. words, avg./max. sentences in the document/question/answer. **NE**: Number of examples.

The probability distribution to predict word w_t from the fixed vocabulary (P_{fv}) is computed by passing state h_t and context vector c_t to a linear layer followed by a softmax function denoted as

$$P_{fv} = softmax(W_v(X_v[h_t, c_t] + b_p) + b_q). \tag{6}$$

To allow decoder to copy words from the encoder sequence, we compute a soft gate (P_{copy}), which helps the decoder to generate a word by sampling from the fixed vocabulary or by copying from a selected text regions (ψ_r). The soft gate is calculated as

$$P_{copy} = \sigma(w_p^T c_t + v_h^T h_t + w_x^T x_t + b_c), \tag{7}$$

where x_t is current word embedding, h_t is hidden state of the decoder, c_t is the context vector, and w_p, v_h, w_x, and b_c are learnable parameters. We maintain a list of out-of-vocabulary (OOV) words for each document. The fixed vocabulary along with this OOV list acts as an extended vocabulary for each document. The final probability distribution (unnormalized) over this extended vocabulary (P_{ev}) is given by

$$P_{ev}(w_t) = (1 - P_{copy})P_{fv}(w_t) + P_{copy} \sum_{i:w_i = w_t} \gamma_i^t. \tag{8}$$

2.4 Training

We jointly estimate the parameters of our model coming from the *Co-attention*, *Context Zoom*, and *Answer Generation* layers, which are denoted as θ_a, θ_z, and θ_g respectively. Estimating θ_a and θ_g is straight-forward by using the cross-entropy objective $J_1(\{\theta_a, \theta_g\})$ and the backpropagation algorithm. However, selecting text regions in the *Context Zoom* layer makes it difficult to estimate θ_z

given their discrete nature. We therefore formulate the estimation of θ_z as a reinforcement learning problem via a policy gradient method. Specifically, we design a reward function over θ_z.

We use mean F-score of ROUGE-1, ROUGE-2, and ROUGE-L (Lin and Hovy, 2003) as our reward function R. The objective function to maximize is the expected reward under the probability distribution of current text regions ψ_r, i.e., $J_2(\theta_z) = E_{p(r|\theta_z)}[R]$. We approximate the gradient $\nabla_{\theta_z} J_2(\theta_z)$ by following the REINFORCE (Williams, 1992) algorithm. To reduce the high variance in estimating $\nabla_{\theta_z} J_2(\theta_z)$ one widely used mechanism is to subtract a baseline value from the reward. It is shown that any number will reduce the variance (Williams, 1992; Zaremba and Sutskever, 2015), here we used the mean of the mini-batch reward b as our baseline. The final objective is to minimize the following equation:

$$J(\theta) = J_1(\{\theta_a, \theta_g\}) - J_2(\theta_z) + \sum_{i=1}^{B} (R^i - b), \quad (9)$$

where, B is the size of mini-batch, and R^i is the reward of example $i \in B$. $J(\theta)$ is now fully differentiable and we use backpropagation to estimate θ.

3 Experimental Results

3.1 Dataset

The NarrativeQA dataset (Kočiskỳ et al., 2017) consists of fictional stories gathered from books and movie scripts, where corresponding summaries and question-answer pairs are generated with the help of human experts and Wikipedia articles. The summaries in NarrativeQA are 4-5 times longer than documents in the SQuAD dataset. Moreover, answers are well-formed by human experts and are not verbatim in the story, thus making this dataset ideal for testing our model. The statistics of NarrativeQA are available in Table 1[1].

3.2 Baselines

We compare our model against reported models in Kočiskỳ et al. (2017) (Seq2Seq, ASR, BiDAF) and the Multi-range Reasoning Unit (MRU) in Tay et al. (2018). We implemented two baseline models (Baseline 1, Baseline 2) with *Context Zoom* layer similar to Wang et al. (2018). In both baselines we replace the span prediction layer with an answer generation layer. In Baseline 1 we use an attention based seq2seq layer without using copy mechanism in the answer generation unit similar to Choi et al. (2017). In Baseline 2 the answer generation unit is similar to our ConZNet architecture.

3.3 Implementation Details

We split each document into sentences using the sentence tokenizer of the NLTK toolkit (Bird and Loper, 2004). Similarly, we further tokenize each sentence, corresponding question and answer using the word tokenizer of NLTK. The model is implemented using Python and Tensorflow (Abadi et al., 2015). All the weights of the model are initialized by Glorot Initialization (Glorot et al., 2011) and biases are initialized with zeros. We use a 300 dimensional word vectors from GloVe (Pennington et al., 2014) (with 840 billion pre-trained vectors) to initialize the word embeddings, which we kept constant during training. All the words that do not appear in Glove are initialized by sampling from a uniform random distribution between [-0.05, 0.05]. We apply dropout (Srivastava et al., 2014) between the layers with keep probability of 0.8 (i.e dropout=0.2). The number of hidden units are set to 100. We trained our model with the AdaDelta (Zeiler, 2012) optimizer for 50 epochs, an initial learning rate of 0.1, and a minibatch size of 32. The hyperparameter 'sample size' (number of relevant sentences) is chosen based on the model performance on the devset.

3.4 Results

Table 2 shows the performance of various models on NarrativeQA. It can be noted that our model with sample size 5 (choosing 5 relevant sentences) outperforms the best ROUGE-L score available so far by 12.62% compared to Tay et al. (2018). The low performance of Baseline 1 shows that the hybrid approach (ConZNet) for generating words from a fixed vocabulary as well as copying words from the document is better suited than span prediction models (Seq2Seq, ASR, BiDAF, MRU).

To validate the importance of finding relevant sentences in contrast to using an entire document for answer generation, we experimented with sample sizes beyond 5. The performance of our model gradually dropped from sample size 7 onwards. This result shows evidence that only a few relevant sentences are sufficient to answer a question.

We also experimented with various sample sizes to see the effect of intra sentence relations for an-

---

[1] please refer Kočiskỳ et al. (2017) for more details

Model	BLEU-1	BLEU-4	ROUGE-L	METEOR
Seq2Seq [*]	15.89	1.26	13.15	4.08
ASR [*]	23.20	6.39	22.26	7.77
BiDAF [*]	33.72	15.53	36.30	15.38
MRU (Tay et al., 2018)	36.55	19.79	41.44	17.87
Baseline 1 (SS=5)	30.22	14.43	34.40	13.36
Baseline 2 (SS=5)	39.35	20.17	43.36	18.01
ConZNet (SS=1)	28.97	16.70	36.02	12.39
ConZNet (SS=3)	36.21	19.33	41.23	17.77
ConZNet (SS=5)	**42.76**	**22.49**	**46.67**	**19.24**
ConZNet (SS=7)	40.80	21.14	44.01	18.67

[*] These results are reported in Kočiskỳ et al. (2017)

Table 2: Performance of various models on NarrativeQA dataset (SS=sample size ≡ number of relevant sentences)

swer generation. The performance of the model improved dramatically with sample sizes 3 and 5 compared to the sample size of 1. These results show that the importance of selecting multiple relevant sentences for generating an answer. In addition, the low performance of Baseline 2 indicates that just selecting multiple sentences is not enough, they should also be related to each other. This result points out that the self-attention mechanism in the *Context zoom* layer is an important component to identify related relevant sentences.

4 Conclusion

We have proposed a new neural-based architecture which condenses an original document to facilitate fast comprehension in order to generate better well-formed answers than span based prediction models. Our model achieved the best performance on the challenging NarrativeQA dataset. Future work can focus for example on designing an inexpensive preprocess layer, and other strategies for improved performance on answer generation.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In Proceedings of the International Conference on Learning Representations.*

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Alexandre Lacoste, Illia Polosukhin, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the ACL*. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *CoRR*, abs/1710.10723.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1601–1611.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *arXiv preprint arXiv:1712.07040*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 71–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael E. J. Masson. 1983. Conceptual processing of text during skimming and rapid sequential reading. *Memory & Cognition*, 11(3):262–274.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *In Proceedings of the International Conference on Learning Representations*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-range reasoning for machine comprehension. *CoRR*, abs/1803.09074.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R^3: Reinforced reader-ranker for open-domain question answering.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. *In Proceedings of the International Conference on Learning Representations*.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines. *CoRR*, abs/1505.00521.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Adaptive Document Retrieval for Deep Question Answering

Bernhard Kratzwald and **Stefan Feuerriegel**
Chair of Management Information Systems
ETH Zurich
Zurich, Switzerland
`{bkratzwald, sfeuerriegel}@ethz.ch`

Abstract

State-of-the-art systems in deep question answering proceed as follows: (1) an initial document retrieval selects relevant documents, which (2) are then processed by a neural network in order to extract the final answer. Yet the exact interplay between both components is poorly understood, especially concerning the number of candidate documents that should be retrieved. We show that choosing a static number of documents – as used in prior research – suffers from a noise-information trade-off and yields suboptimal results. As a remedy, we propose an adaptive document retrieval model. This learns the optimal candidate number for document retrieval, conditional on the size of the corpus and the query. We report extensive experimental results showing that our adaptive approach outperforms state-of-the-art methods on multiple benchmark datasets, as well as in the context of corpora with variable sizes.

1 Introduction

Question-answering (QA) systems proceed by following a two-staged process (Belkin, 1993): in a first step, a module for document retrieval selects n potentially relevant documents from a given corpus. Subsequently, a machine comprehension module extracts the final answer from the previously-selected documents. The latter step often involves hand-written rules or machine learning classifiers (c. f. Shen and Klakow, 2006; Kaisser and Becker, 2004), and recently also deep neural networks (e. g. Chen et al., 2017; Wang et al., 2018)

The number of candidate documents n affects the interplay between both document retrieval and machine comprehension component. A larger n improves the recall of document retrieval and thus the chance of including the relevant information. However, this also increases the noise and might adversely reduce the accuracy of answer extraction. It was recently shown that a top-1 system can potentially outperform a system selecting more than one document (Kratzwald and Feuerriegel, 2018). This finding suggests that a static choice of n can result a suboptimal performance.

Contributions. This work analyzes the interplay between document retrieval and machine comprehension inside neural QA systems. We first reason numerically why a fixed choice of n in document retrieval can negatively affect the performance of question answering. We thus propose a novel machine learning model that adaptively selects the optimal n_i for each document retrieval. The resulting system outperforms state-of-the-art neural question answering on multiple benchmark datasets. Notably, the overall size of the corpus affects the optimal n considerably and, as a result, our system evinces as especially superior over a fixed n in settings where the corpus size is unknown or grows dynamically.

2 Related Work

Taxonomy of QA systems. Question answering systems are frequently categorized into two main paradigms. On the one hand, knowledge-based systems draw upon manual rules, ontologies and large-scale knowledge graphs in order to deduce answers (e. g. Berant et al., 2013; Lopez et al., 2007; Unger et al., 2012). On the other hand, QA system incorporate a document retrieval module which selects candidate documents based on a chosen similarity metric, while a subsequent module then processes these in order to extract the answer (e. g. Cao et al., 2011; Harabagiu et al., 2000).

Deep QA. Recently, Chen et al. (2017) developed a state-of-the-art deep QA system, where the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 576–581
Brussels, Belgium, October 31 - November 4, 2018.

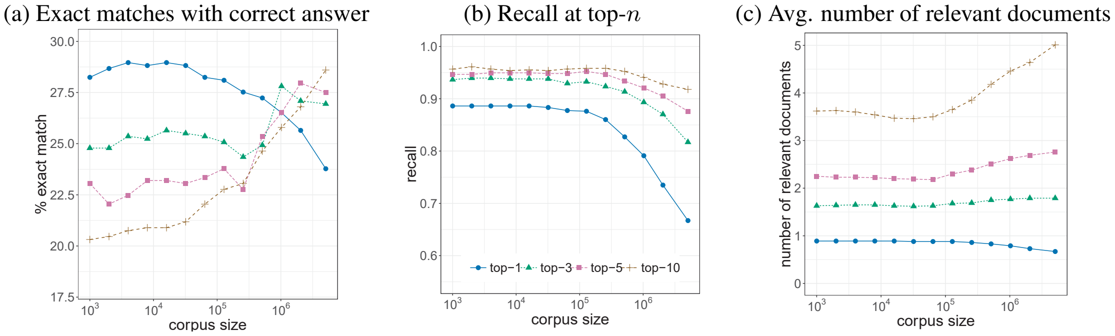

Figure 1: Comparison of how top-n document retrieval affects deep QA. Plot (a) shows the percentage of exact matches with the correct answering, thereby measuring the end-to-end performance of the complete system. Plot (b) gives the recall at top-n, i. e. the fraction of samples where *at least once* the correct answer is returned. Plot (c) depicts the *average number* of documents that contain the ground-truth answer. As a result, the recall lowers with increasing corpus size, yet this not necessarily compromises a top-n system, as it often contains the correct answer more than once.

answer is extracted from the top $n = 5$ documents. This choice stems from computing the dot product between documents and a query vector; with tf-idf weighting of hashed bi-gram counts. Wang et al. (2018) extended this approach by implementing a neural re-ranking of the candidate document, yet keeping the fixed number of n selected documents unchanged. In particular, the interplay between both modules for document retrieval and machine comprehension has not yet been studied. This especially pertains to the number of candidate documents, n, that should be selected during document retrieval.

Component interactions. Extensive research has analyzed the interplay of both document retrieval and machine comprehension in the context of knowledge-based systems (c. f. Moldovan et al., 2003) and even retrieval-based systems with machine learning (c. f. Brill et al., 2002). However, these findings do not translate to machine comprehension with deep learning. Deep neural networks consist of a complex attention mechanism for selecting the context-specific answer (Hermann et al., 2015) that has not been available to traditional machine learning and, moreover, deep learning is highly sensitive to settings involving multiple input paragraphs, often struggling with selecting the correct answer (Clark and Gardner, 2017).

3 Noise-Information Trade-Off in Document Retrieval

In the following, we provide empirical evidence why a one-fits-all n can be suboptimal. For this purpose, we run a series of experiments in order to obtain a better understanding of the interplay between document retrieval and machine comprehension modules. That is, we specifically compare the recall of document retrieval to the end-to-end performance of the complete QA system; see Fig. 1. Our experiments study the sensitivity along two dimensions: on the one hand, we change the number of top-n documents that are returned during document retrieval and, on the other hand, we vary the corpus size.

Our experiments utilize the TREC QA dataset as a well-established benchmark for open-domain question answering. It contains 694 question-answer pairs that are answered with the help of Wikipedia. We vary the corpus between a small case (where each question-answer pair contains only one Wikipedia article with the correct answer plus 50 % articles as noise) and the complete Wikipedia dump containing more than five million documents. Our experiments further draw upon the DrQA system (Chen et al., 2017) for question answering that currently stands as a baseline in deep question answering. We further modified it to return different numbers of candidate documents.

Fig. 1 (a) shows the end-to-end performance across different top-n document retrievals as measured by the exact matches with ground truth. For a small corpus, we clearly register a superior performance for the top-1 system. However, we observe a different pattern with increasing corpus size. Fig. 1 (b) and (c) shed light into the underlying reason by reporting how frequently the correct answer is returned and, as the correct an-

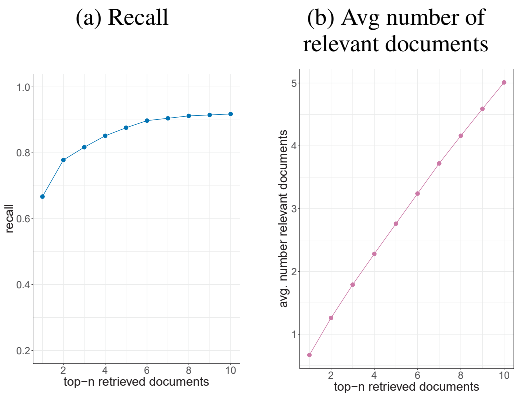
(a) Recall

(b) Avg number of relevant documents

Figure 2: Recall (a) and average number of relevant documents (b) for growing top-n configurations and a static corpus size (full Wikipedia dump). While the recall is converging the number of relevant documents keeps growing resulting in a higher density of relevant information.

swer might appear multiple times, how often it is included in the top-n. Evidently, the recall in (b) drops quickly for a top-1 system when augmenting the corpus. Yet it remains fairly stable for a top-n system, due to the fact that it is sufficient to have the correct answer in any of the n documents. According to (c), the correct answer is often more than once returned by a top-n system, increasing the chance of answer extraction.

The above findings result in a noise-information trade-off. A top-1 system often identifies the correct answer for a small corpus, whereas a larger corpus introduces additional noise and thus impedes the overall performance. Conversely, a top-n system accomplishes a higher density of relevant information for a large corpus as the answer is often contained multiple times. This effect is visualized in an additional experiment shown in Fig. 2. We keep the corpus size fixed and vary only n, i.e. the number of retrieved documents. We see the recall converging fast, while the average number of relevant documents keeps growing, leading to a higher density of relevant information. As a result, a top-n system might not be compromised by a declining recall, since it contains the correct answer over-proportionally often. This logic motivates us in the following to introduce an adaptive n_i that optimizes the number of documents retrievals in a top-n system independently for every query q_i.

4 Adaptive Document Retrieval

This section advances deep question answering by developing adaptive methods for document retrieval. Our methods differ from conventional document retrieval in which the number of returned documents is set to a fixed n. Conversely, we actively optimize the choice of n_i for each document retrieval i. Formally, we select n_i between 1 and a maximum τ (e.g. $\tau = 20$), given documents $[d_i^{(1)}, \ldots, d_i^{(\tau)}]$. These entail further scores denoting the relevance, i.e. $s_i = [s_i^{(1)}, \ldots, s_i^{(\tau)}]^T$ with normalization s.t. $\sum_j s_i^{(j)} = 1$. The scoring function is treated as a black-box and thus can be based on simple tf-idf similarity but also complex probabilistic models.

4.1 Threshold-Based Retrieval

As a naïve baseline, we propose a simple threshold-based heuristic. That is, n_i is determined such that the cumulative confidence score reaches a fixed threshold $\theta \in (0, 1]$. Formally, the number n_i of retrieved documents is given by

$$n_i = \max_k \sum_{j=1}^{k} s_i^{(j)} < \theta. \qquad (1)$$

In other words, the heuristic fills up documents until surpassing a certain confidence threshold. For instance, if the document retrieval is certain that the correct answer must be located within a specific document, it automatically selects fewer documents.

4.2 Ordinal Regression

We further implement a trainable classifier in the form of an ordinal ridge regression which is tailored to ranking tasks. We further expect the cumulative confidence likely to be linear. The classifier then approximates n_i with a prediction y_i that denotes the position of the first relevant document containing the desired answer. As such, we learn a function

$$y_i = f([s_i^{(1)}, \ldots, s_i^{(\tau)}]) = \lceil s_i^T \beta \rceil, \qquad (2)$$

where $\lceil \ldots \rceil$ denotes the ceiling function.

The ridge coefficients are learned through a custom loss function

$$\mathcal{L} = \| \lceil X\beta \rceil - y \|_1 + \lambda \, \| \beta \|_2 \, , \qquad (3)$$

where X is a matrix containing scores of our training samples. In contrast to the classical ridge regression, we introduce a ceiling function and replace the mean squared error by a mean absolute error in order to penalize the difference from the optimal rank. The predicted cut-off $\hat{n}_i$ for document retrieval is then computed for new observations ${s'}_i$ via $\hat{n}_i = \lceil {s'}_i^T \hat{\beta} \rceil + b$. The linear offset b is added in order to ensures that $n_i \leq \hat{n}_i$ holds, i. e. reducing the risk that the first relevant document is not included.

We additionally experimented with non-linear predictors, including random forests and feed-forward neural networks; however; we found no significant improvement that justified the additional model complexity over the linear relationship.

5 Experiments

We first compare our QA system with adaptive document retrieval against benchmarks from the literature. Second, we specifically study the sensitivity of our adaptive approach to variations in the corpus size. All our experiments draw upon the DrQA implementation (Chen et al., 2017), a state-of-the-art system for question answering in which we replaced the default module for document retrieval with our adaptive scheme (but leaving all remaining components unchanged, specifically without altering the document scoring or answer extraction).

For the threshold-based model, we set $\tau = 15$ and the confidence threshold to $\theta = 0.75$. For the ordinal regression approach, we choose $\tau = 20$ and use the original SQuAD train-dev split from the full corpus also as the basis for training across all experiments.

5.1 Overall Performance

In a first series of experiments, we refer to an extensive set of prevalent benchmarks for evaluating QA systems, namely, SQuAD (Rajpurkar et al., 2016), Curated TREC (Baudiš and Šedivý, 2015), WikiMovies (Miller et al., 2016) and WebQuestions (Berant et al., 2013) in order to validate the robustness of our findings. Based on these, we then evaluate our adaptive QA systems against the naïve DrQA system in order to evaluate the relative performance. We included the deep QA system R^3 as an additional, top-scoring benchmark from recent literature (Wang et al., 2018) for better comparability.

Tbl. 1 reports the ratio of exact matches for the different QA systems. The results demonstrate the effectiveness of our adaptive scheme: it yields the best-performing system for three out of four datasets. On top of that, it outperforms the naïve DrQA system consistently across all datasets.

5.2 Sensitivity: Adaptive QA to Corpus Size

We earlier observed that the corpus size affects the best choice of n and we thus study the sensitivity with regard to the size. For this purpose, we repeat the experiments from Section 3 in order to evaluate the performance gain from our adaptive scheme. More precisely, we compare the ordinal regression ($b = 1$) against document retrieval with a fixed document count n.

Fig. 3 shows the end-to-end performance, confirming the overall superiority of our adaptive document retrieval. For instance, the top-1 system reaches a slightly higher rate of exact matches for small corpus sizes, but is ranked last when considering the complete corpus. The high performance of the top-1 system partially originates from the design of the experiment itself, where we initially added *one* correct document per question, which is easy to dissect by adding little additional noise. On the other hand, the top-10 system accomplishes the best performance on the complete corpus, whereas it fails to obtain an acceptable performance for smaller corpus sizes.

To quantify our observations, we use a notation of regret. Formally, let μ_{nm} denote the performance of the top-n system on a corpus of size m. Then the regret of choosing system n at evaluation point m is the difference between the best performing system μ_m^* and the chosen system $r_{nm} = \mu_m^* - \mu_{nm}$. The total regret of system n is computed by averaging the regret over all observations of system n, weighted with the span in-between observations in order to account for the logarithmic intervals. The best top-n system yields a regret of 0.83 and 1.12 respectively, whereas our adaptive control improves it down to 0.70.

5.3 Robustness Check

Experiments so far have been conducted on the DrQA system. To show the robustness of our approach, we repeat all experiments on a different QA system. Different from DrQA, this system operates on paragraph-level information retrieval and

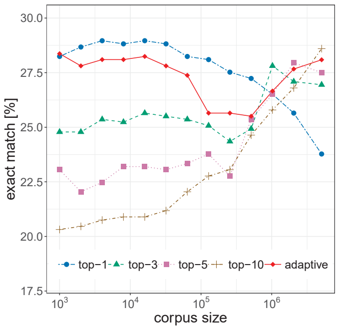

Figure 3: End-to-end performance of adaptive information retrieval over static top-n configurations and a growing corpus.

	SQuAD	TREC	WebQuestions	WikiMovies
DrQA (Chen et al., 2017)†	29.3	27.5	18.5	36.6
Threshold-based ($\theta = 0.75$)	**29.8**	28.7	19.2	38.6
Ordinal regression ($b = 1$)	29.7	28.1	19.4	38.0
Ordinal regression ($b = 3$)	29.6	**29.3**	**19.6**	38.4
R^3 (Wang et al., 2018)	29.1	28.4	17.1	**38.8**

†: Numbers vary slightly from those reported in the original paper, as the public repository was optimized for runtime performance.

Table 1: End-to-end performance of the plain DrQA system measured in exact matches. Performance of two threshold based and two regression based adaptive retreival improvements as well as other state-of-the art systems. Experiments are based on the full Wikipedia dump containing more than 5 million documents.

	SQuAD	TREC	WebQuestions	WikiMovies
Top-50 System	27.0	23.5	15.1	24.4
Top-80 System	27.2	25.9	14.9	26.0
Threshold-based ($\theta = 0.75, \tau = 100$)	27.2	**27.1**	15.4	26.3
Ordinal regression ($b = 3, \tau = 250$)	**27.3**	**27.1**	**16.7**	**26.5**

Table 2: End-to-end performance measured in percentages of exact matching answers of a second QA system that operates on paragraph-level information retrieval. We compare two configurations of the system using the top-50 and top-80 ranked paragraphs to extract the answer against our threshold-based approach and regression approach that selects the cutoff within the first 250 paragraphs.

uses cosine similarity to score tf-idf-weighted bag-of-word (unigram) vectors. The reader is a modified version of the DrQA document reader with an additional bi-directional attention layer (Seo et al., 2017). We are testing two different configurations[1] of this system: one that selects the top-50 paragraphs and one that selects the top-80 paragraphs against our approach as shown in Tab. 2. We see that, owed to the paragraph-level information retrieval, the number of top-n passages gains even more importance. Both variations of the system outperform a system without adaptive retrieval, which confirms our findings.

6 Conclusion

Our contribution is three-fold. First, we establish that deep question answering is subject to a noise-information trade-off. As a consequence, the number of selected documents in deep QA should not be treated as fixed, rather it must be carefully tailored to the QA task. Second, we propose adaptive schemes that determine the optimal document count. This can considerably bolster the performance of deep QA systems across multiple benchmarks. Third, we further demonstrate how crucial an adaptive document retrieval is in the context of different corpus sizes. Here our adaptive strategy presents a flexible strategy that can successfully adapt to it and, compared to a fixed document count, accomplishes the best performance in terms of regret.

Reproducibility

Code to integrate adaptive document retrieval in custom QA system and future research is freely available at `https://github.com/bernhard2202/adaptive-ir-for-qa`

Acknowledgments

We thank the anonymous reviewers for their helpful comments. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. Cloud computing resources were provided by a Microsoft Azure for Research award.

---

[1] Best configurations out of $\{30, 40, 50, 60, 70, 80, 90, \text{and } 100\}$ on SQuAD train split.

References

Petr Baudiš and Jan Šedivý. 2015. Modeling of the question answering task in the yodaqa system. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 222–228, Cham. Springer.

Nicholas Belkin. 1993. Interaction with texts: Information retrieval as information-seeking behavior. *Information Retrieval*, 93:55–66.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing*, pages 1533–1544.

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the AskMSR question-answering system. In *Empirical Methods in Natural Language Processing*, pages 257–264.

YongGang Cao, Feifan Liu, Pippa Simpson, Lamont Antieau, Andrew Bennett, James J. Cimino, John Ely, and Hong Yu. 2011. Askhermes: An online question answering system for complex clinical questions. *Journal of Biomedical Informatics*, 44(2):277–288.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Annual Meeting of the Association for Computational Linguistics*, pages 1870–1879.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2000. Falcon: Boosting knowledge for answer engines. In *Text REtrieval Conference*, pages 479–488.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems*, 1:1693–1701.

Michael Kaisser and Tilman Becker. 2004. Question answering by searching large corpora with linguistic methods. In *Text REtrieval Conference*.

Bernhard Kratzwald and Stefan Feuerriegel. 2018. Putting question-answering systems into practice: Transfer learning for efficient domain customization. *arXiv preprint arXiv:1804.07097*.

Vanessa Lopez, Victoria Uren, Enrico Motta, and Michele Pasin. 2007. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):72–105.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Empirical Methods in Natural Language Processing*, pages 1400–1409.

Dan Moldovan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. 2003. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21(2):133–154.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing*, pages 2383–2392.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Dan Shen and Dietrich Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *Annual Meeting of the Association for Computational Linguistics*, pages 889–896.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *Conference on World Wide Web*, page 639.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced ranker-reader for open-domain question answering. In *Conference on Artificial Intelligence*.

Why is unsupervised alignment of English embeddings from different algorithms so hard?

Mareike Hartmann
Dep. of Computer Science
University of Copenhagen
Denmark
hartmann@di.ku.dk

Yova Kementchedjhieva
Dep. of Computer Science
University of Copenhagen
Denmark
yova@di.ku.dk

Anders Søgaard
Dep. of Computer Science
University of Copenhagen
Denmark
soegaard@di.ku.dk

Abstract

This paper presents a challenge to the community: Generative adversarial networks (GANs) can perfectly align independent English word embeddings induced using *the same* algorithm, based on distributional information alone; but fails to do so, for two different embeddings algorithms. *Why is that?* We believe understanding why, is key to understand *both* modern word embedding algorithms *and* the limitations and instability dynamics of GANs. This paper shows that (a) in all these cases, where alignment fails, there exists a linear transform between the two embeddings (so algorithm biases do not lead to non-linear differences), and (b) similar effects can not easily be obtained by varying hyper-parameters. One plausible suggestion based on our initial experiments is that the differences in the inductive biases of the embedding algorithms lead to an optimization landscape that is riddled with local optima, leading to a very small basin of convergence, but we present this more as a challenge paper than a technical contribution.

1 Introduction

This paper brings together two fascinating research topics in natural language processing (NLP), namely *understanding the properties of word embeddings* (Mikolov et al., 2013; Mitchell and Steedman, 2015; Mimno and Thompson, 2017) and *unsupervised bilingual dictionary induction* (Conneau et al., 2018; Zhang et al., 2017; Søgaard et al., 2018). In an effort to better understand when unsupervised bilingual dictionary induction is possible, we factored out linguistic differences between languages, and studied English-English alignability (by learning to align English embeddings trained on different samples of the English Wikipedia), when we came across a puzzling phenomena: *English-English can be aligned with almost 100% precision, if you use the same embedding algorithms for the two samples, but not at all (0% precision), if you use different embedding algorithms*. This results suggest that the properties of word embeddings induced by different algorithms challenge unsupervised bilingual dictionary algorithms. Understanding why will enable us to develop more stable adversarial learning algorithms and give us a better understanding of how embedding algorithms differ.

Contributions We are, to the best of our knowledge, the first to study unsupervised alignability of pairs of English word embeddings. We show that unsupervised alignment – specifically the MUSE system (Conneau et al., 2018) – fails when the algorithms used to induce the two embeddings differ, and that this is *not* because there is no linear transformation between the two spaces. We further show that poor initialization, as a result of MUSE initially applying an identity transform to two word embeddings far apart in space, is not the sole reason the discriminator suffers from local optima. Finally, we present an experiment showing what the minimal corpus size is for unsupervised alignment to succeed, in the absence of linguistic differences.

2 Aligning embeddings

2.1 Unsupervised alignment using generative adversarial networks

MUSE (Conneau et al., 2018) uses a vanilla generative adversarial network (GAN) with a linear generator to learn alignments between embedding spaces without supervision. In a two-player game, a discriminator D aims to tell the two language spaces apart, while a generator G aims to map the source language into the target language space, fooling the discriminator. While MUSE achieves impressive results at times, MUSE is highly unstable, e.g., with different initializations precision

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 582–586
Brussels, Belgium, October 31 - November 4, 2018.

scores vary between 0% and 45% for English-Greek (Søgaard et al., 2018).

The parameters of a GAN with a linear generator are (Ω, w). They are obtained by solving the following min-max problem:

$$\min_{\Omega} \max_{w} \mathrm{E}[\log(D_w(X)) + \log(1 - D_w(g_{\Omega}(Z)))] \tag{1}$$

which reduces to

$$\min_{\Omega} JS(P_X \mid P_{\Omega}) \tag{2}$$

Ω is initialized as the identity matrix I.

If G wins the game against an ideal discriminator on a very large number of samples, then F (the source vector space) and ΩE (with E being the target vector space) can be shown to be close in Jensen-Shannon divergence, and thus the model has learned the true distribution. This result, referring to the distributions of the data, p_{data}, and the distribution, p_g, G is sampling from, is from Goodfellow et al. (2014): *If G and D have enough capacity, and at each step of training, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$E_{\mathbf{x}\sim p_{data}}[\log D_G^*(\mathbf{x})] + E_{\mathbf{x}\sim p_g}[\log(1 - D_G^*(\mathbf{x}))]$$

then p_g converges to p_{data}.

This result relies on a number of assumptions that do not hold in practice. Our generator, which learns a linear transform Ω, has very limited capacity, for example, and we are updating Ω rather than p_g. In practice, therefore, during training, we alternate between k steps of optimizing the discriminator and one step of optimizing the generator. If the GAN-based alignment is not successful, this can thus be a result of two things: Either that G does not have enough capacity, or that D is stuck in a local optimum. Our results in §3 show that the inability to align English-English in the case of different word embedding algorithms is *not* a result of limited capacity, but a result of the GAN being trapped in one of the many local optima of the loss function.

2.2 Supervised alignment using Procrustes Analysis

Procrustes Analysis (Schönemann, 1966) has been commonly used for supervised alignment of word embeddings (Smith et al., 2017; Artetxe et al., 2018). Here, the optimal alignment between two embedding spaces is computed using singular value decomposition of the aligned embeddings in a seed dictionary. Conneau et al. (2018) use Procrustes Analysis to refine an initial seed dictionary learned by the generative adversarial network without supervision. In our supervised experiments, we use 5000 seed words as supervision for learning the alignment between embeddings.

2.3 Geometry of embeddings

Below we summarize some previous findings about the geometry of monolingual embeddings (Mimno and Thompson, 2017), and add some new observations. We discuss five embedding algorithms: SVD on positive PMI matrices (Hyperwords-SVD) (Levy et al., 2015), skip-gram negative sampling applied to co-occurrence matrices (Hyperwords-SGNS) (Levy et al., 2015), continuous bag-of-words (CBOW) (Mikolov et al., 2013a), GloVe (Pennington et al., 2014), and FastText (Bojanowski et al., 2017). To analyze the geometry of our monolingual embeddings in space, we report average inner product to mean vector; see Mimno and Thompson (2017) for details.

Hyperwords-SVD have a small average inner product (0.0032), suggesting they are well-dispersed through space; like Hyperwords-SGNS and standard SGNS (Mimno and Thompson, 2017), they do not exhibit a clear word frequency bias. **Hyperwords-SGNS** vectors also have a small average inner product (0.0002), in contrast with standard SGNS vectors, which are narrowly clustered in a single orthant (Mimno and Thompson, 2017). In line with standard SGNS vectors, the frequency of words has relatively little effect on their inner product, with the exception of the rare words, which have slightly less positive inner products. **CBOW** vectors have a relatively large average inner product (4.2985). The vectors trained by **GloVe** show a clear relationship with word frequency, with low-frequency words opposing the frequency-balanced mean vector. The embeddings are well-dispersed, with an average inner product of 0.0002. Finally, **FastText** vectors have a large, positive inner product with the mean (0.2988), indicating that they are not evenly dispersed through the space, but pointing in roughly the same direction. The FastText vectors exhibit a frequency bias, much like what has been previously observed with GloVe vectors. The differences are the results of the inductive biases of the different embedding algorithms.

3 Experiments

This section presents our data, the hyper-parameters of our embeddings, our experimental protocols, and our results.

3.1 Data

In the following experiments we learn word embeddings on samples of a publicly available Wikipedia dump from March 2018.[1] The data is preprocessed using a publicly available preprocessing script[2], extracting text, removing non-alphanumeric characters, converting digits to text, and lowercasing the text.

3.2 Hyper-parameters

We train 300-dimensional word embeddings using the algorithms' recommended hyperparameter settings, listed in the following:[3] For **Hyperwords-SGNS**, the window size is set to 2 and the subsampling of frequent words and smoothing of the context distribution are disabled. The minimal word count for being in the vocabulary is 100. The same applies for **Hyperwords-SVD**, and the exponent for weighting the eigenvalue matrix is 0.5. For **CBOW**, the window size is set to 8, the number of negative samples is 25, and the subsampling threshold for frequent words is 1e-4. For **GloVe**, the window size is set to 15 and the cutoff parameter x_{max} to 10. Finally, for **FastText**, the window size is 5, the number of negatives samples is 5 and the sampling threshold is 0.0001.

3.3 Main experiments

We train word embeddings using the different embedding algorithms listed in §3.2 on two non-overlapping 10% samples of the English Wikipedia dump (the samples contain 463,576 and 528,556 distinct words, with an overlap in vocabulary of 351,858 words). We learn unsupervised and supervised alignments for embeddings (as described in §2) trained by different algorithms on the same datasplits, and for embeddings trained by the same algorithm on the two different datasplits. For the unsupervised alignments, we use the default parameters of the MUSE system for the adversarial training, i.e. a discriminator with 2 fully connected layers of 2048 units trained over 5 epochs, 1,000,000 iterations per epoch with 5 discriminator steps per iteration and a batch size of 32.

We evaluate the alignments in terms of Precision@1 in the word translation retrieval task for the 1500 test words used by Bojanowski et al. (2017). The results are shown in Table 1[4]. Our main observations are: (a) MUSE learns perfect alignments for embeddings learned by the same algorithm on different data splits. (b) MUSE cannot learn alignments for embeddings learned by different algorithms on the same data splits, even if there exists a linear transformation aligning both sets of embeddings (the supervised algorithm learns perfect alignments). We also verify that MUSE cannot learn to align embeddings from different algorithms *even when induced from the same sample*. As already mentioned, we also ran experiments to check that the failure of MUSE to learn good alignments was not a result of the differences in hyper-parameter settings. §3.4 presents additional experiments with normalization, for control; §3.5 addresses how much data is needed to align independently induced embeddings from the same algorithm. §4 discusses potential answers to why MUSE fails when embeddings are induced using different algorithms.

3.4 Experiments with normalization

The embeddings in the main experiments differ in several ways; see §2. One possible explanation for the inability of MUSE to align embeddings from different algorithms could be that the two embeddings are so far apart in space that the discriminator learns to discriminate between them too quickly. Recall that Ω is initialized as the identity matrix I, which means that the generator initially presents the discriminator with the source embedding as is. This is an effect that has often been observed with GANs (Arjovsky and Bottou, 2017); could this also be the explanation for our results? At a first glance, this seems a possible explanation. The inner products with the mean differ significantly for the five embedding

---

[1] `https://dumps.wikimedia.org/enwiki/`

[2] `http://mattmahoney.net/dc/textdata.html`

[3] We also ran experiments with one of the embedding algorithms (FastText) to check if our results were robust across hyper-parameter settings

[4] We report Precision at 1 scores but find that the pattern is the same for Precision at 10, with perfect alignments for embeddings from the same algorithm and 0 scores for alignments between embeddings from different algorithms in the unsupervised experiments.

	Hyperwords-SGNS	Hyperwords-SVD	CBOW	GloVe	FastText
		UNSUPERVISED			
Hyperwords-SGNS	**0.997**				
Hyperwords-SVD	0.000	**0.992**			
CBOW	0.000	0.000	**0.997**		
GloVe	0.000	0.000	0.000	**0.997**	
FastText	0.000	0.000	0.000	0.000	**0.997**
		SUPERVISED			
Hyperwords-SVD	0.967				
CBOW	0.990	0.989			
GloVe	0.985	0.992	0.999		
FastText	0.994	0.994	0.999	0.997	

Table 1: Precision at 1 (P@1) for unsupervised GAN alignment with Procrustes refinement (top) and supervised Procrustes analysis for the cases in which unsupervised alignment fails (bottom). Results clearly show that GANs can align two independent embeddings induced by the same algorithm; but not embeddings aligned by different ones. Supervised Procrustes analysis, on the other hand, perfectly aligns the embeddings in both cases.

algorithms (see §2). The only embeddings that have roughly the same directionality are Hyperwords and GloVe, and their centroids are very far apart in cosine space. The cosine similarity of the centroids of the two versions of Hyperwords is -0.006, and the cosine similarity for Hyperwords-SVD and GloVe is 0.019. However, poor initialization as a result of applying the identity transform to very distant word embeddings is not the explanation for the poor performance of MUSE in this set-up: Both sets of Hyperwords embeddings were normalized, but alignment still failed. To verify this holds in general, i.e., that results are not affected by normalization in general, we also ran experiments with the remaining 14 embedding pairs, normalizing and/or centering both embeddings. Results stayed the same: Precision at 1 scores of 0.

3.5 Learning curve

MUSE perfectly aligns independently induced word embeddings induced by the same algorithm. For FastText, it correctly aligns 99.7% of all words in the evaluation lexicon with itself. Our samples are 10% of a publicly available Wikipedia dump, amounting to more than 400 million tokens per sample. English-English alignment is an interesting control experiment for unsupervised bilingual dictionary induction, abstracting away from linguistic differences, and we ran a series of experiments to see how small samples MUSE can align in the absence of linguistic differences. The learning curve is presented in Figure 1.

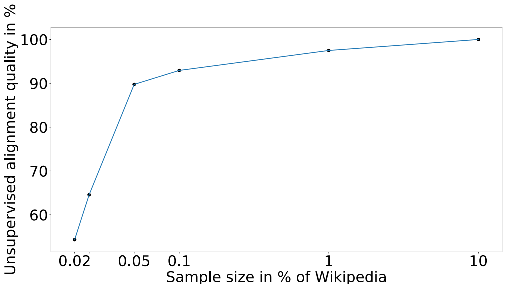

Figure 1: Unsupervised alignment quality for FastText embeddings trained on samples of different sizes, evaluated on 878 words covered by all of the embeddings. The x-axis is log-scaled.

4 Discussion

We have shown that the fact that MUSE cannot align two embedding spaces for English induced by different algorithms (even if using the same corpus), is *not* a result of there not being a linear transformation, and not a result of (lack of) normalization or trivial differences in model hyperparameters. The only explanation left seems to be that the inductive biases of the different algorithms lead to a loss landscape so riddled with local optima that MUSE cannot possible escape them.

To support this hypothesis, compare the loss curves for the MUSE runs aligning embeddings induced with the *same* algorithms (black curves) to the runs aligning embeddings induced with dif-

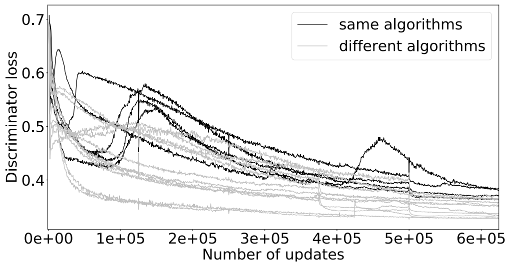

Figure 2: Discriminator losses using the same algorithm for source and target (black curves) or using different algorithms (grey curves).

ferent algorithms, in Figure 2. When the embeddings are induced by the same algorithm, we clearly see the contours of a min-max game, suggesting that the generator and discriminator challenge each other, both contributing to a good alignment. When the embeddings are induced by different algorithms, however, the discriminator quickly drops, with the generator unable to push the discriminator out of a local optimum. *Understanding when biases induce highly non-convex landscapes, and how to make adversarial training less sensitive to such scenarios, remains an open problem, which we think will be key to the success of unsupervised machine translation and related tasks.*

References

Martin Arjovsky and Leon Bottou. 2017. Towards principled methods for training generative adversarial networks. In *ICLR*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of AAAI*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word Translation Without Parallel Data. In *Proceedings of ICLR*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David WardeFarley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. In *Proceedings of NIPS*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL* 3:211–225.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS*.

Tomas Mikolov, Kai Chen, Gregroy S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*.

David Mimno and Laure Thompson. 2017. The strange geometry of skip-gram with negative sampling. In *Proceedings of EMNLP*.

Jeff Mitchell and Mark Steedman. 2015. Orthogonality of syntax and semantics within distributional spaces. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Peter Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika* 31:1–10.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of ICLR (Conference Track)*.

Anders Søgaard, Sebastian Ruder, and Ivan Vulic. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of ACL*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of ACL*.

Quantifying Context Overlap for Training Word Embeddings

Yimeng Zhuang, Jinghui Xie, Yinhe Zheng and Xuan Zhu
Samsung Research Institute China - Beijing (SRC-B)
{ym.zhuang, jh.xie, yh.zheng, xuan.zhu}@samsung.com

Abstract

Most models for learning word embeddings are trained based on the context information of words, more precisely first order co-occurrence relations. In this paper, a metric is designed to estimate second order co-occurrence relations based on context overlap. The estimated values are further used as the augmented data to enhance the learning of word embeddings by joint training with existing neural word embedding models. Experimental results show that better word vectors can be obtained for word similarity tasks and some downstream NLP tasks by the enhanced approach.

1 Introduction

In the last decade, the distributed word representation (a.k.a word embedding) has attracted tremendous attention in the field of natural language processing (NLP). Instead of large vectors, such as the one-hot representation, the distributed word representation embeds semantic and syntactic characteristics of words into a low-dimensional space, which makes it popular in NLP applications.

The main idea of most word embedding models follows the distributional hypothesis (Harris, 1954), i.e., the embedding of each word may be inferred using its context. An important model family for distributional word representation learning is built based on the global matrix factorization approach (Deerwester et al., 1990; Lee and Seung, 2001; Srebro et al., 2005; Mnih and Hinton, 2007; Li et al., 2015; Wang and Cohen, 2016), in which a dimensionality reduction over a sparse matrix is performed to capture the statistical information about a corpus in low-dimensional vectors. Another model family is neural word embeddings (Levy and Goldberg, 2014b), some attempts include the famous Neural Probabilistic Language Model (Bengio et al., 2003), SGNS and CBOW (Mikolov et al., 2013a,b), GloVe (Pennington et al., 2014) and their variants (Shazeer et al., 2016; Kenter et al., 2016; Ling et al., 2017; Patel et al., 2017).

Most of these models capture the context information of each word using the co-occurrence matrix. However, the co-occurrence matrix only represents relatively local information, i.e., it describes context associations based on word pairs' co-occurrence counts without considering global context perspective. Besides, the co-occurrence matrix is only an estimation of a corpus, which is only a sample of a language. A mass of related word pairs may not be observed in the corpus, and the latent relations between unobserved word pairs may not be modeled well due to the missing knowledge.

Few attempts are carried out to indirectly deal with unobserved co-occurrence for dense neural word embeddings. SGNS (Mikolov et al., 2013a,b) indirectly addresses this problem through negative sampling. Swivel (Shazeer et al., 2016) improves GloVe by using a "soft hinge" loss to prevent from over-estimating zero co-occurrences. However, the latent relations between unobserved word pairs are not explicitly represented. There are also some works around semantic composition and distributional inference (Mitchell and Lapata, 2008; Erk and Padó, 2008, 2010; Reisinger and Mooney, 2010; Thater et al., 2011; Kartsaklis et al., 2013; Kober et al., 2016) that are explored to address the sparseness problem, but they are not designed for training neural word embeddings.

In this paper, we explore an approach that utilizes context overlap information to dig up more effective co-occurrence relations and propose extensions for GloVe and Swivel to validate the positive impact of introducing context overlap.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 587–593
Brussels, Belgium, October 31 - November 4, 2018.

2 Quantify Context Overlap

In this work, we explore quantifying context overlap based on the observation that to a certain extent the overlap of Point-wise Mutual Information (PMI) (Church and Hanks, 1990) reflects context overlap.

As shown in Figure 1, two separate words may exhibit a particular aspect of interest or be semantically related when the overlap area between their PMI is relatively large.

The calculation of complete PMI-weighted context overlap may be time-consuming when the number of words is large. To make the time complexity affordable, only the context words that have strong lexical association with a target word i are considered:

$$S_i = \{k \in V | PMI(i,k) > h_{PMI}\} \qquad (1)$$

in which V is the vocabulary, h_{PMI} is a threshold which acts as a magnitude to shift PMI, and S_i denotes the set that consists of the context words that have enough large PMI values with the target word i. It is expected that most context information associated with the word i can be captured by its PMI values over S_i. Then, we measure the degree of context overlap (CO) between two target words i, j as a function of their PMI values over the intersection of S_i and S_j, i.e.,

$$CO(i,j) = \sum_{k \in S_i \cap S_j} \min(f(PMI(i,k)), f(PMI(j,k))) \qquad (2)$$

where f is a monotonic mapping function to rectify the data characteristics for certain objective function in word embedding training.

Compared to identity function $f(x) = x$, we find exponential function $f(x) = exp(x)$ works much better in our experiments. For the quantized context overlap, the exponential mapping function results in a similar data distribution as the co-occurrence counts, i.e., few word pairs have extremely large values while most word pairs' values are distributed in a relatively small range.

3 Extend to Existing Models

We consider the original co-occurrence matrix as a description of first order co-occurrence relations, while the quantized context overlap as a description of second order co-occurrence relations (Schütze, 1998), i.e., co-co-occurrences, which is represented by "non-logarithmic PMI-weighted context overlap" in this work. The context overlap between two words can be inferred even when they never co-occur in the corpus. According to our statistics, more than 84% word pairs in the second order co-occurrence matrix are not included in the first order co-occurrence matrix. We expect introducing second order co-occurrence relations may enhance the quality of the word embedding that is originally trained on first order co-occurrence relations. GloVe (Pennington et al., 2014) and Swivel (Shazeer et al., 2016) are extended by joint training with context overlap information in this paper.

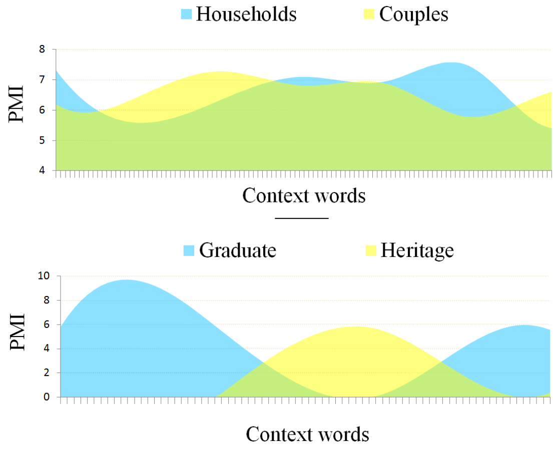

Figure 1: PMI of different words. The x-axis represents a series of context words in a subset of the whole vocabulary, the y-axis denotes PMI values between the target word and context words. (a) The upper part of this figure illustrates the large overlap between semantics related words. (b) The lower part, on the contrary, is an example of relatively unrelated word pair, in which the overlap is relatively small.

GloVe The logarithmic co-occurrence matrix is factorized in GloVe with bias terms, and a weighted least squares loss function is optimized:

$$\mathcal{J}_{GloVe} = \sum_{i,j} \lambda_{ij} (w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 \qquad (3)$$

where X_{ij} denotes the word-context co-occurrence count between a target word i and a context word j. The model parameters to be learned include $w_i \in \mathbb{R}^d$, $\tilde{w}_j \in \mathbb{R}^d$, b_i and $\tilde{b}_j$, which correspond to target word vector, context word vector, bias terms associated with the target word and the context word, respectively. λ_{ij} is a weight whose value equals to $(\min(X_{ij}, x_{max})/x_{max})^\alpha$.

To extend GloVe, two tasks are trained in parallel during the training process: One is the main task that follows the original GloVe training process as above; Another one is an auxiliary task that tunes word embeddings using context overlap. The parameters of word embeddings are shared in both tasks.

Following GloVe-style loss function, in the auxiliary task, the dot products of word vectors are pushed to estimate logarithmic second order co-occurrence.

$$\mathcal{J}^{(2)}_{GloVe} = \sum_{i,j} \lambda^{(2)}_{ij} (Aw_i^T w_j + b^{(2)}_i + b^{(2)}_j - \log X^{(2)}_{ij})^2 \quad (4)$$

where the superscripts $^{(2)}$ are used to differentiate with the terms in the original GloVe. $X^{(2)}_{ij} = CO(i,j)$ represents context overlap, a word independent learnable scale A is adopted to relieve the potential inconformity between first order and second order co-occurrences. The weight $\lambda^{(2)}_{ij}$ is similar to the original λ_{ij}, but using a different hyperparameter $x^{(2)}_{max}$.

The multi-task (Ruder, 2017) loss function is the weighted sum of the two tasks, i.e., $\mathcal{J} = \mathcal{J}_{GloVe} + \beta \cdot \mathcal{J}^{(2)}_{GloVe}$, where the weight β is a hyperparameter.

Swivel As pointed out by (Levy et al., 2015) , if the bias terms in GloVe are fixed to the logarithmic count of the corresponding word, the dot products of target word vectors and context word vectors are almost equivalent to the approximation of logarithmic PMI matrix with a shift of $\log \sum_{i,j} X_{ij}$. Submatrix-wise Vector Embedding Learner (Swivel) directly reconstructs the PMI matrix by dot product between target vectors and context vectors and deals with unobserved co-occurrences using a "soft hinge" loss function. (Shazeer et al., 2016) details its loss functions and training process. In our extended version, we add a supplementary loss function to handle second order co-occurrences. When the second order co-occurrence $X^{(2)}_{ij}$ is more than zero, the PMI of context overlap is approximated.

$$\frac{1}{2} \lambda^{(2)}_{ij} (Aw_i^T w_j + B - PMI^{(2)}(i,j))^2 \quad (5)$$

in which A, B are word independent learnable scale parameters, and $PMI^{(2)}(i,j)$ is the Pointwise Mutual Information computed on the second order co-occurrence matrix $[X^{(2)}_{ij}]$.

4 Experiments

4.1 Setup

Corpus The training dataset contains 6 billion tokens collected from diversified corpora, including the News Crawl corpus (Chelba et al., 2013), the April 2010 Wikipedia dump (Shaoul, 2010; Lee and Chen, 2017), and a year-2012 subset of the Reddit comment datasets [1].

Preprocessing Following (Lee and Chen, 2017), the Stanford tokenizer is used to process the training corpus, which are split into sentences with characters converted to lower cases. Punctuations are removed.

Parameter Configuration The vocabularies are limited to the 200K most frequent words. Following (Pennington et al., 2014), a decreasing weighting function is adopted to construct the co-occurrence matrix. We use symmetric context window of five words to the left and five words to the right.

For GloVe, recommended parameters in (Pennington et al., 2014) are used. Specifically, we set $\alpha = \frac{3}{4}$, $x_{max} = 100$, initial learning rate as 0.05, 100 iterations. For Swivel, recommended parameters in (Shazeer et al., 2016) are used. The weighting function is $0.1 + 0.25x_{ij}^{0.5}$, each shard is sampled about 100 times. But we set the block size as 4000 so that the vocabulary size can be divided exactly.

For the auxiliary tasks, we tune the hyperparameters on the small News Crawl corpus. And we find that in an appropriate range, the threshold h_{PMI} is not sensitive to the performance. In this paper, h_{PMI}, $x^{(2)}_{max}$ and β are set to $\log 100$, 10000 and 0.2 respectively. Since there is no difference between target vectors and context vectors (except random initialization), in order to keep symmetry, we not only approximate context overlap between target vectors, but also approximate context overlap between context vectors simultaneously. Final vectors are the sum of w and $\tilde{w}$ in both GloVe and Swivel.

4.2 Intrinsic Evaluation

Table 1 shows the evaluation results of word similarity tasks and word analogy tasks. Word similarity is measured as the Spearman's rank correlation ρ between human-judged similarity and cosine distance of word vectors. In word analogy

---

[1] Available at `https://files.pushshift.io/reddit/comments/`

Method	WS353	SL999	SCWS	RW	MEN	MT771	Analogy	
							Sem	Syn
GloVe	66.8	35.0	59.3	44.1	74.7	69.9	76.0	75.3
GloVe + CO	69.7	38.0	63.8	45.1	77.6	71.3	78.6	75.0
SGNS	71.1	40.7	**67.1**	52.8	78.1	70.4	67.2	77.3
Swivel	73.1	39.9	66.4	53.4	79.1	71.7	78.6	78.0
Swivel + CO	**74.0**	**41.2**	66.3	**53.6**	**79.8**	**72.5**	**79.4**	**78.1**

Table 1: Word similarity and analogy results ($\rho \times 100$ and analogy accuracy). We denote context overlap enhanced method with "+ CO". 300-dimensional embeddings are used. The datasets used include WS353 (Finkelstein et al., 2001), SL999 (Hill et al., 2016), SCWS (Huang et al., 2012), RW (Luong et al., 2013), MEN (Bruni et al., 2014), MT771 (Halawi et al., 2012), and Mikolov's analogy dataset (Mikolov et al., 2013a).

task, the questions are answered over the whole vocabulary through 3CosMul (Levy and Goldberg, 2014a). In addition to GloVe and Swivel, the evaluations of SGNS are also reported for reference. We train SGNS with the word2vec tool, using symmetric context window of five words to the left and five words to the right, and 5 negative samples.

As can be seen from the table, the context overlap information enhanced word embeddings perform better in most word similarity tasks and get higher analogy accuracy in semantic aspect at the cost of syntactic score. The improved semantics performance, to a certain extent, reflects second order co-occurrence relations are more semantic.

4.3 Text Classification

Text classification tasks are conducted on five shared benchmark datasets from (Kim, 2014) including binary classification tasks CR (Hu and Liu, 2004), MR (Pang and Lee, 2005), Subj (Pang and Lee, 2004) and multiple classification tasks TREC (Li and Roth, 2002), SST1 (Socher et al., 2013). Texts are preprocessed following the description of Section 4.1. We train Convolutional Neural Networks (CNN) on top of our static pretrained word vectors following (Kim, 2014). To avoid the high-risk of single-run estimate being false (Melis et al., 2017; Reimers and Gurevych, 2017), average classification accuracies of 20 runs are reported as the final scores. The results are shown in Table 2. As can be seen from the results that the enhanced word embeddings outperform the baselines.

5 Model Analysis

As it is known to all, word frequency plays an important role in the computation of word embeddings (Gittens et al., 2017). Inspired from the graph in (Shazeer et al., 2016), relations between word analogy accuracy and the log mean frequency of the words in analogy questions and answers are plotted on Figure 2. The word embeddings trained by GloVe with or without context overlap information are used here.

Method	CR	MR	SST1	Subj	TREC
GloVe	80.9	76.5	46.9	90.9	89.7
+ CO	$81.7^{\dagger}$	76.4	$47.6^{\dagger}$	$91.4^{\dagger}$	$90.2^{\dagger}$
Swivel	81.7	76.7	47.9	91.4	90.4
+ CO	$82.4^{\dagger}$	76.7	$48.3^{\dagger}$	$91.7^{\dagger}$	90.5
CBOW	80.6	75.3	46.5	89.8	89.6
SGNS	81.6	77.0	48.0	91.2	90.6

Table 2: Text classification results (Acc.%). Pretrained word vectors with 300 dimensions are reported here. Enhanced runs statistically significantly (t-test, p-value < 0.05) different from the GloVe/Swivel baseline runs are marked with a †. The results of CBOW and SGNS are also given for reference.

An obvious semantic performance improvement is observed in the range of low frequency. Our observation of second order co-occurrences may explain this fact. We randomly sample 1 million word pairs, and rank these word pairs in descending order by their quantized context overlap. In all the word pairs, average word frequency is 13934.4. However, it is only 1676.1 in the top 0.1% word pairs, it is 3984.8 in the top 1%, and it is 7904.9 in the top 10%. This may be caused by PMI's bias towards infrequent words, but it illustrates infrequent words carry more information in second order co-occurrence relations.

6 Conclusion

In this paper, we propose an empirical metric to enhance the word embeddings through estimating second order co-occurrence relations using con-

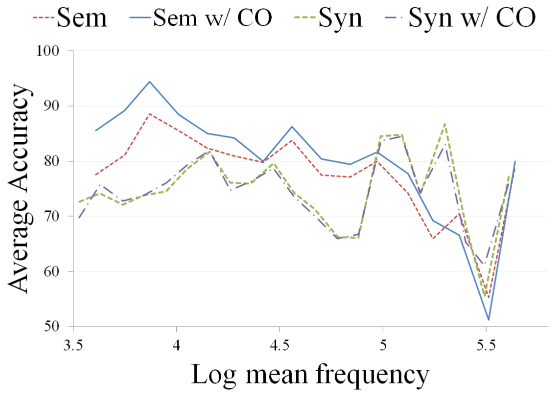

Figure 2: Relations between word analogy accuracy and the log mean frequency.

text overlap. Instead of only local statistical information, context overlap leverages global association distribution to measure word pairs correlation.

The proposed method is easy to extend to existing models, such as GloVe and Swivel, by an auxiliary objective function. The improvement in experimental results helps to validate the positive impact of introducing quantized context overlap.

We have considered the feasibility of enriching SGNS and CBOW with information from context-overlap. However, because of their training mode, we can't remake them in a straightforward way following their "original spirit". When training SGNS and CBOW, the program scans the training text. The target and context words are chosen using a slide window and negative sampling is used. In this process, no co-occurrence matrix is explicitly computed, and we fail to extend it in a united form as we extend GloVe and Swivel. The extensions for GloVe and Swivel can also be used for reference for extending other word embedding approaches that are trained on co-occurrence matrix. The exploration for second order co-occurrence can be traced back to 1990s. We think it is helpful to revive the classical method in a modern, embedding driven way. How to integrate second order co-occurrence information for approaches like SGNS, CBOW should be an interesting future work.

As future works, we suggest further investigating the characteristics of context overlap in diversified ways.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, 49(2014):1–47.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906. Association for Computational Linguistics.

Katrin Erk and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *Proceedings of the acl 2010 conference short papers*, pages 92–97. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Alex Gittens, Dimitris Achlioptas, and Michael W Mahoney. 2017. Skip-gram-zipf+ uniform= vector additivity. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 69–76.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM.

Zellig Harris. 1954. Distributional structure. In *Word*, pages 10(23):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating disambiguation from composition in distributional semantics. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 114–123.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Thomas Kober, Julie Weeds, Jeremy Reffin, and David Weir. 2016. Improving sparse word representations with distributional inference for semantic composition. *arXiv preprint arXiv:1608.06794*.

Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562.

Guang-He Lee and Yun-Nung Chen. 2017. Muse: Modularizing unsupervised sense embeddings. *arXiv preprint arXiv:1704.04601*.

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*, pages 171–180.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Yitan Li, Linli Xu, Fei Tian, Liang Jiang, Xiaowei Zhong, and Enhong Chen. 2015. Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *IJCAI*, pages 3650–3656.

Yuan Ling, Yuan An, Mengwen Liu, Sadid A Hasan, Yetian Fan, and Xiaohua Hu. 2017. Integrating extra knowledge into word embedding models for biomedical nlp tasks. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 968–975. IEEE.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. *proceedings of ACL-08: HLT*, pages 236–244.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Kevin Patel, Divya Patel, Mansi Golakiya, Pushpak Bhattacharyya, and Nilesh Birari. 2017. Adapting pre-trained word embeddings for use in medical coding. *BioNLP 2017*, pages 302–306.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. *arXiv preprint arXiv:1707.09861*.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational linguistics*, 24(1):97–123.

Cyrus Shaoul. 2010. The westbury lab wikipedia corpus. *Edmonton, AB: University of Alberta*.

Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. 2016. Swivel: Improving embeddings by noticing what's missing. *arXiv preprint arXiv:1602.02215*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. 2005. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1134–1143.

William Yang Wang and William W Cohen. 2016. Learning first-order logic embeddings via matrix factorization. In *IJCAI*, pages 2132–2138.

Neural Latent Relational Analysis to Capture Lexical Semantic Relations in a Vector Space

Koki Washio$^{*\dagger 1}$ and **Tsuneaki Kato**^{*2}
*Department of Language and Information Sciences, The University of Tokyo
†RIKEN Center for Advanced Intelligence Project
{^{1}`kokiwashio@g.ecc`, ^{2}`kato@boz.c`}`.u-tokyo.ac.jp`

Abstract

Capturing the semantic relations of words in a vector space contributes to many natural language processing tasks. One promising approach exploits lexico-syntactic patterns as features of word pairs. In this paper, we propose a novel model of this pattern-based approach, neural latent relational analysis (NLRA). NLRA can generalize co-occurrences of word pairs and lexico-syntactic patterns, and obtain embeddings of the word pairs that do not co-occur. This overcomes the critical data sparseness problem encountered in previous pattern-based models. Our experimental results on measuring relational similarity demonstrate that NLRA outperforms the previous pattern-based models. In addition, when combined with a vector offset model, NLRA achieves a performance comparable to that of the state-of-the-art model that exploits additional semantic relational data.

1 Introduction

Information on the semantic relations of words is important for many natural language processing tasks, such as recognizing textual entailment, discourse classification, and question answering. There are two main approaches to obtain the distributed relational representations of word pairs.

One is the vector offset method (Mikolov et al., 2013a,b). This approach represents word pairs as the vector offsets of their word embeddings. Another approach exploits lexico-syntactic patterns to obtain word pair representations. As a pioneer work, Turney (2005) introduced latent relational analysis (LRA), based on the *latent relation hypothesis*. It states that word pairs that co-occur in similar lexico-syntactic patterns tend to have similar semantic relations (Turney, 2008b; Turney and Pantel, 2010). LRA is expected to complement the vector offset model because word embeddings do not contain information on lexico-syntactic patterns that connect word pairs in a corpus (Shwartz et al., 2016).

However, LRA cannot obtain the representations of word pairs that do not co-occur in a corpus. Even with a large corpus, observing a co-occurrence of all semantically related word pairs is nearly impossible because of Zipf's law, which states that most content words rarely occur. This data sparseness problem is a major bottleneck of pattern-based models such as LRA.

In this paper, we propose neural latent relational analysis (NLRA) to solve that data sparseness problem. NLRA unsupervisedly learns the embeddings of target word pairs and co-occurring patterns from a corpus. In addition, it jointly learns the mapping from the word embedding space to the word-pair embedding space. By this mapping, NLRA can generalize the co-occurrences of word pairs and patterns, and obtain the relational embeddings for arbitrary word pairs even if they do not co-occur in the corpus.

Our experimental results on the task of measuring relational similarity show that NLRA significantly outperforms LRA, and it can also capture semantic relations of word pairs without co-occurrences. Moreover, we show that combining NLRA and the vector offset model improves the performance and leads to competitive results to those of the state-of-the-art method that exploits additional semantic relational data.

2 Background

2.1 Vector Offset Model

The vector offset model (Mikolov et al., 2013a,b; Levy and Goldberg, 2014) obtains word embeddings from a corpus and represents each word pair (a, b) as the vector offset of their embedding as

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 594–600
Brussels, Belgium, October 31 - November 4, 2018.

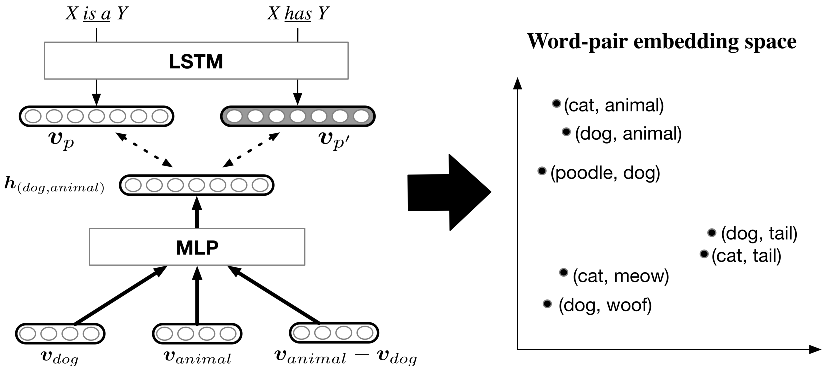

Figure 1: An illustration of NLRA

follows:

$$\boldsymbol{v}_{(a,b)} = \boldsymbol{v}_b - \boldsymbol{v}_a \quad (1)$$

where $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$ are the word embeddings of a and b respectively.

This method regards relational information as the change in multiple topicality dimensions from one word to the other in the word embedding space (Zhila et al., 2013). Meanwhile, it does not contain the information of lexico-syntactic patterns that were shown to capture complementary information with word embeddings in previous studies on the lexical semantic relation detection (Levy et al., 2015; Shwartz et al., 2016).

2.2 Latent Relational Analysis

LRA takes a set of word pairs as input and generates the distributed representations of those word pairs based on their co-occurring patterns.

Given target word pairs $W = \{(a_1, b_1), \ldots, (a_n, b_n)\}$, LRA constructs a list of lexico-syntactic patterns that connect those pairs, such as *is a* or *in the*, from the corpus for each word pair. Then, those patterns are generalized by replacing any or all or none of the intervening words with wildcards. As a feature selection, the generalized patterns generated from many word pairs are used as features. We define the set of these target feature patterns as $C = \{p_1, \ldots, p_m\}$. Then, the $2n \times 2m$ matrix M is constructed. The rows of M correspond to pairs (a_i, b_i) and reversed pairs (b_i, a_i). The columns of M correspond to patterns Xp_iY and swapped patterns Yp_iX, where X and Y are the slots for the words of the word pairs. The value of M_{ij} represents the strength of the association between the corresponding word pair and pattern, which is calculated using weighting methods such as positive pointwise mutual information (PPMI). After these processes, the singular value decomposition (SVD) is applied to M, and the vector $\boldsymbol{v}_{(a,b)}$ is assigned to each word pair (a, b).

Although pattern-based approaches such as LRA have achieved promising results in some semantic relational tasks (Turney, 2008a,b), they have a crucial problem that a co-occurrence of all semantically related word pairs cannot be observed because of Zipf's law, which states that the frequency distribution of words has a long tail. In other words, most words occur very rarely (Hanks, 2009). For the word pairs without co-occurrences, LRA cannot obtain their vector representations.

3 Neural Latent Relational Analysis

We introduce NLRA, based on the latent relation hypothesis. NLRA represents the target word pairs and lexico-syntactic patterns as embeddings. Similar to the skip-gram model (Mikolov et al., 2013a), NLRA updates those representations unsupervisedly, such that the inner products of the word pairs and patterns in which they co-occur in a corpus have high values. Through this learning, the word pairs that co-occur in similar patterns have similar embeddings. Moreover, NLRA can generalize the co-occurrences of the word pairs and patterns by constructing the embeddings of the word pairs from their word embeddings, thus solving the data sparseness problem of word co-occurrences. Therefore, NLRA can provide representations that capture the information of lexico-syntactic patterns even for the word pairs that do not co-occur in a sentence.

Figure 1 is an illustration of our model. NLRA encodes a word pair (a, b) into a dense vector as follows:

$$\boldsymbol{h}_{(a,b)} = MLP([\boldsymbol{v}_a; \boldsymbol{v}_b; \boldsymbol{v}_b - \boldsymbol{v}_a]) \quad (2)$$

where $[\boldsymbol{v}_a; \boldsymbol{v}_b; \boldsymbol{v}_b - \boldsymbol{v}_a]$ is the concatenation of the word embeddings of a and b and their vector offsets; MLP is a multilayer perceptron with nonlinear activation functions.

A pattern p is a sequence of the words $w_1, \ldots, w_k$. The sequence of the corresponding word embeddings $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ are encoded using long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). Then, the final output vector $\boldsymbol{v}_p$ is used as the pattern embedding.

For unsupervised learning, we use the negative sampling objective (Mikolov et al., 2013a). Given a set of observed triples $(a, b, p) \in D$, where a and b are words such that $(a, b) \in W$, or $(b, a) \in W$ and p is a co-occurring pattern from a corpus, the objective is as follows:

$$\begin{aligned} L &= \sum_{(a,b,p) \in D} \log \sigma(\boldsymbol{v}_p \cdot \boldsymbol{h}_{(a,b)}) \\ &+ \sum_{(a,b,p') \in D'} \log \sigma(-\boldsymbol{v}_{p'} \cdot \boldsymbol{h}_{(a,b)}) \end{aligned} \quad (3)$$

where D' is a set of randomly generated negative samples and σ is the sigmoid function. We sampled 10 negative patterns for each word pair. This objective is maximized using the stochastic gradient descent.

After unsupervised learning, we can obtain word pair representations $\boldsymbol{v}_{(a,b)}$ as follows:

$$\boldsymbol{v}_{(a,b)} = [\boldsymbol{h}_{(a,b)}; \boldsymbol{h}_{(b,a)}] \quad (4)$$

4 Evaluation

4.1 Dataset

In our evaluation, we used the SemEval-2012 Task 2 dataset (Jurgens et al., 2012) for the task of measuring relational similarity. This dataset contains a collection of 79 fine-grained semantic relations. For each relation, there are a few prototypical word pairs and a set of several dozen target word pairs. The task is to rank the target pairs based on the extent to which they exhibit the relation. In our experiment, we calculated the score of a target word pair with the average cosine similarity between it and each prototypical word pair. The models are evaluated in terms of the MaxDiff accuracy and Spearman's correlation. Following previous works (Rink and Harabagiu, 2012; Zhila et al., 2013), we used the test set that includes 69 semantic relations to evaluate the performance.

4.2 Baselines

VecOff. We used the 300-dimensional pre-trained GloVe (Pennington et al., 2014)[1] and represented word pairs as described in Section 2.1.

LRA. We implemented LRA as described in Section 2.2. We set W as the lemmatized word pairs of the dataset. We used the English Wikipedia as a corpus. For each word pair, we searched for patterns of from one to three words. When searching for patterns, the left word and right word adjacent to the patterns were lemmatized to ignore their inflections. Following (Turney, 2008b), we selected C as the top $20|W|$ generalized patterns. Then, M was constructed using PPMI weighting, and its dimensionality was reduced to 300 using SVD.

4.3 Our methods

NLRA. For each word pair in the dataset, co-occurring patterns were extracted from the same corpus in the same manner as with LRA, resulting in D. For word embeddings, we used the same pre-trained GloVe as VecOff. These embeddings were updated during the training. For MLP, we used three affine transformations followed by the batch normalization (Ioffe and Szegedy, 2015) and tanh activation. The size of each hidden layer of the MLP was 300. To encode the patterns, we used LSTM with the 300-dimensional hidden state. The objective was optimized by AdaGrad (Duchi et al., 2011) (whose learning rate was 0.01). We trained the model for 50 epochs.

NLRA+VecOff. This method combines NLRA and VecOff by averaging their score for a target word pair.

4.4 Result and Analysis

Table 1 displays the overall result.

NLRA vs. LRA

First, NLRA outperformed LRA in terms of both the average accuracy and correlation. These differences were statistically significant ($p < 0.01$) with the paired t-test. These results indicate that generalizing patterns with LSTM is better than by using wildcards. Moreover, NLRA can successfully calculate the relational similarity for the word pairs that do not co-occur in the corpus. Table 2 shows an example of the Reference–Express relation, where the middle-score pair *handshake:cordiality*

---

[1] https://nlp.stanford.edu/projects/glove/

	Accuracy				Correlation			
Relation	VecOff	LRA	NLRA	NLRA+VecOff	VecOff	LRA	NLRA	NLRA+VecOff
Class-Inclusion	0.543	0.485	0.533	**0.56**	0.487	0.427	**0.622**	0.611
Part-Whole	0.45	0.427	0.465	**0.488**	0.304	0.282	0.38	**0.395**
Similar	0.414	0.346	0.412	**0.436**	0.267	0.123	0.271	**0.315**
Contrast	0.343	0.349	**0.377**	0.374	0.108	0.065	0.092	**0.124**
Attribute	0.462	0.414	0.447	**0.486**	0.406	0.299	0.367	**0.456**
Non-Attribute	**0.39**	0.366	0.369	0.381	**0.217**	0.16	0.125	0.174
Case Relations	0.468	0.438	0.536	**0.558**	0.391	0.291	**0.553**	0.544
Cause Purpose	0.444	0.471	0.448	**0.485**	0.345	0.387	0.397	**0.454**
Space-Time	0.5	0.428	0.516	**0.525**	0.424	0.31	0.489	**0.493**
Reference	0.441	0.447	0.449	**0.465**	0.297	0.346	**0.404**	0.378
Average	0.443	0.415	0.453	**0.475**	0.321	0.246	0.36	**0.391**

Table 1: Average MaxDiff accuracy and Spearman's correlation of each major relation group.

Pair	Human	LRA	NLRA
laugh:happiness	50	0.217	0.578
nod:agreement	46	0.245	0.347
tears:sadness	44	0.381	0.483
...		...	
scream:terror	26	0.396	0.417
handshake:cordiality	24	**0 (no pattern)**	0.34
lie:dishonesty	16	0.206	0.394
...		...	
discourse:relationship	-60	0.331	0.275
friendliness:wink	-68	**0 (no pattern)**	0.26

Table 2: The scores assigned by humans, LRA, and NLRA for the Reference-Express relation. The pairs are sorted in descending order according to the human score.

Model	Accuracy	Correlation
Rink and Harabagiu (2012)	0.394	0.229
Mikolov et al. (2013b)	0.418	0.275
Levy and Goldberg (2014)	0.452	–
Zhila et al. (2013)	0.452	0.353
Iacobacci et al. (2015)	0.459	0.358
Turney (2013)	0.472	**0.408**
VecOff	0.443	0.321
LRA	0.415	0.264
NLRA	0.453	0.36
NLRA+VecOff	**0.475**	0.391

Table 3: Published results of other models on the SemEval2012 Task 2 dataset.

and the low-score pair *friendliness:wink* have no co-occurring pattern. In these cases, LRA could not obtain the representations of those word pairs nor correctly assign the score. By contrast, NLRA could accomplish both because it could generalize the co-occurrences of word pairs and patterns.

NLRA+VecOff vs. Other Models

Second, NLRA+VecOff outperformed the other models. These differences were statistically significant (the correlation difference between NLRA+Vecoff and NLRA: $p < 0.05$; the other differences: $p < 0.01$). These results indicate that lexico-syntactic patterns and the vector offset of word embeddings capture complementary information for measuring relational similarity. This is inconsistent with the findings of Zhila et al. (2013). That work combined heterogeneous models, such as the vector offset model, pattern-based model, etc., and stated that the pattern-based model was less significant than the vector offset model, based on their ablation study. We believe that this was because their pattern-based model did not generalize patterns with wildcards nor select useful features. Their pattern-based model seemed to suffer from sparse feature space. In our experiment, NLRA helped VecOff, for example, for the Part-Whole relation, Cause Purpose relation, and Space-Time relation, where there seemed to be prototypical patterns indicating those relations. Meanwhile, VecOff helped NLRA for the Attribute relation, where the relational patterns seemed to be diverse. These results showed that the combined model is robust.

4.5 Comparison to other systems

We compared the results of our models to other published results. Table 3 displays those results. Rink and Harabagiu (2012) is the pattern-based model with naive Bayes. Mikolov et al. (2013b), Levy and Goldberg (2014), and Iacobacci et al. (2015) are the vector offset models. Zhila et al. (2013) is the model composed of various features. Turney (2013) extracts the statistical features of two word pairs from a word-context co-occurrence matrix and trains the classifier with additional semantic relational data to assign a relational similarity for two word pairs.

NLRA+VecOff achieved a competitive performance to the state-of-the-art method of Turney (2013). Note that our method learns unsupervisedly and does not exploit additional resources, and the method of Turney (2013) cannot obtain the distributed representation of word pairs.

A work similar to ours, Bollegala et al. (2015), represented lexico-syntactic patterns as the vector offset of co-occurring word pairs and updated the vector offsets of word pairs such that word pairs that co-occur in similar patterns have similar offsets. They evaluated their model on all 79 semantic relations of the dataset and achieved 0.449 accuracy. In their setting, NLRA+VecOff achieved 0.47 accuracy, outperforming their model.

5 Related Work

5.1 Word Pairs and Co-occurring Patterns

Hearst (1992) detected the hypernymy relation of word pairs from a corpus using several handcrafted lexico-syntactic patterns. Turney and Littman (2005) used 64 handcrafted lexico-syntactic patterns as features of word pairs to represent word pairs as vectors. To obtain word-pair embeddings, Turney (2005) extended the method of Turney and Littman (2005) as LRA. Our work is a neural extension of LRA.

Washio and Kato (2018) proposed the method similar to ours in lexical semantic relation detection. Their neural method modeled the co-occurrences of word pairs and dependency paths connecting two words to alleviate the data sparseness problem of pattern-based lexical semantic relation detection. While they assigned randomly initialized embeddings to each dependency path, our work encodes co-occurring patterns with LSTM for better generalization. Jameel et al. (2018) embedded word pairs with the context words occurring around word pairs instead of lexico-syntactic patterns. Their method cannot obtain embeddings of word pairs that do not co-occur in a corpus because they directly assigned embeddings to word pairs. By contrast, NLRA can obtain embeddings for those word pairs.

In another research area, relation extraction, several works have explored an idea similar to the latent relation hypothesis (Riedel et al., 2013; Toutanova et al., 2015; Verga et al., 2017). They factorized a matrix of entity pairs and co-occurring patterns, while they focused on named entity pairs instead of word pairs and did not consider co-occurrence frequencies.

5.2 Relation to Knowledge Graph Embedding

Knowledge graph embedding (KGE) embeds entities and relations in knowledge graph (KG), where entities and relations corresponds to nodes and edges respectively (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013; Wang et al., 2014; Lin et al., 2015; Yang et al., 2015; Nickel et al., 2016; Trouillon et al., 2016; Liu et al., 2017; Wang et al., 2017; Ishihara et al., 2018). By considering words and lexico-syntactic patterns as nodes and edges, respectively, a corpus can be viewed as a graph, i.e., corpus graph (CG). Thus, NLRA can be regarded as corpus graph embedding (CGE) models based on the latent relation hypothesis.

Although KGE models can be easily applied to CG, several differences exist between KG and CG. First, the nodes and edges of CG are (sequences of) linguistic expressions, such as tokens, lemmas, phrases, etc. Thus, the nodes and edges of CG might exhibit compositionality and ambiguity, while KG does not have those properties. Second, the edges of CG have weights based on co-occurrence frequencies unlike the edges of KG. Finally, CG might have a large number of edges types while the number of KG edges is at most several thousands. An interesting research direction is exploring models suitable for CGE to capture the property of linguistic expressions and their relations in the embedding space.

6 Conclusion

We presented NLRA, which learns the distributed representation of word pairs capturing semantic relational information through co-occurring patterns encoded by LSTM. This model jointly learns the mapping from the word embedding space into the word-pair embedding space to generalize co-occurrences of word pairs and patterns. Our experiment on measuring relational similarity demonstrated that NLRA outperforms LRA and can successfully solve the data sparseness problem of word co-occurrences, which is a major bottleneck in pattern-based approaches. Moreover, combining the vector offset model and NLRA yielded competitive performance to the state-of-the-art method, though our method relied only on unsupervised learning. This combined model exploits the complementary information of lexico-syntactic patterns and word embeddings.

In our future work, we will apply word-pair embeddings from NLRA to various downstream tasks related to lexical relational information.

Acknowledgments

This work was supported by JSPS KAKENHI Grant numbers JP17H01831. We thank Satoshi Sekine and Kentaro Inui for helpful discussion.

References

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Embedding semantic relations into word representations. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1222–1228.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Patrick Hanks. 2009. The impact of corpora on dictionaries. In Paul Baker, editor, *Contemporary Corpus Linguistics*, pages 214–236. Continuum, London, Great Britain.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.

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

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 95–105. Association for Computational Linguistics.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, pages 448–456.

Takahiro Ishihara, Katsuhiko Hayashi, Hitoshi Manabe, Masashi Shimbo, and Masaaki Nagata. 2018. Neural tensor networks with diagonal slice matrices. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 506–515. Association for Computational Linguistics.

Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. 2018. Unsupervised learning of distributional relation vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33. Association for Computational Linguistics.

David Jurgens, Saif Mohammad, Peter Turney, and Keith Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 356–364. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180. Association for Computational Linguistics.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2168–2178, International Convention Centre, Sydney, Australia. PMLR.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751. Association for Computational Linguistics.

Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016. Holographic embeddings of knowledge graphs. In *AAAI*, volume 2, pages 3–2.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 809–816, New York, NY, USA. ACM.

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

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84. Association for Computational Linguistics.

Bryan Rink and Sanda Harabagiu. 2012. Utd: Determining relational similarity using lexical patterns. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 413–418. Association for Computational Linguistics.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2389–2398, Berlin, Germany. Association for Computational Linguistics.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509. Association for Computational Linguistics.

Tho Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2071–2080, New York, New York, USA. PMLR.

Peter Turney. 2008a. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 905–912. Coling 2008 Organizing Committee.

Peter D. Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1136–1141.

Peter D Turney. 2008b. The latent relation mapping engine: Algorithm and experiments. *Journal of Artificial Intelligence Research*, 33:615–655.

Peter D. Turney. 2013. Distributional semantics beyond words: Supervised learning of analogy and paraphrase. *Transactions of the Association for Computational Linguistics*, 1:353–366.

Peter D Turney and Michael L Littman. 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60(1-3):251–278.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Patrick Verga, Arvind Neelakantan, and Andrew McCallum. 2017. Generalizing to unseen entities and entity pairs with row-less universal schema. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 613–622. Association for Computational Linguistics.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, volume 14, pages 1112–1119.

Koki Washio and Tsuneaki Kato. 2018. Filling missing paths: Modeling co-occurrences of word pairs and dependency paths for recognizing lexical semantic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1123–1133. Association for Computational Linguistics.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR)*, pages 809–816.

Alisa Zhila, Wen-tau Yih, Christopher Meek, Geoffrey Zweig, and Tomas Mikolov. 2013. Combining heterogeneous models for measuring relational similarity. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1000–1009. Association for Computational Linguistics.

Generalizing Word Embeddings using Bag of Subwords

Jinman Zhao, Sidharth Mudgal, Yingyu Liang
Department of Computer Sciences
University of Wisconsin-Madison
`{jz,sidharth,yliang}@cs.wisc.edu`

Abstract

We approach the problem of generalizing pre-trained word embeddings beyond fixed-size vocabularies without using additional contextual information. We propose a subword-level word vector generation model that views words as bags of character n-grams. The model is simple, fast to train and provides good vectors for rare or unseen words. Experiments show that our model achieves state-of-the-art performances in English word similarity task and in joint prediction of part-of-speech tag and morphosyntactic attributes in 23 languages, suggesting our model's ability in capturing the relationship between words' textual representations and their embeddings.

1 Introduction

Word embeddings have been an essential part of neural-network based approaches for natural language processing tasks (Goldberg, 2016). However, many popular word embeddings techniques have a fixed vocabulary (Mikolov et al., 2013; Pennington et al., 2014), i.e., they can only provide vectors over a finite set of common words that appear frequently in a given corpus. Such methods fail to generate vectors for rare words and words not present in the training corpus, but appearing in the test corpus or downstream task texts, raising difficulty for any methods relying on word vectors to efficiently extract useful features from text. This is often referred to as the out-of-vocabulary (OOV) word problem. We aim to address this problem by inferring vectors for OOV words with only access to pre-trained vectors over a fixed vocabulary of common words and the OOV word itself without context.

The motivations come from both linguistics and natural language processing applications. First, from a linguistic view a word can be decomposed into multiple morphemes: stems, affixes, modifiers and etc. This is more often the case for rare words. In some field such as chemistry and agglutinative languages such as Turkish, there exists a systematic way of composing words from morphemes. Some can even be arbitrarily long.

Apart from the explicit and systematic way of making words, we can also observe the ability of a language speaker to infer the meaning of an unseen word. For instance, one can guess that "pre-EMNLP" means "before EMNLP", even without the presence of any context, suggesting that it is part of our implicit linguistic knowledge to infer meaning of an unseen word solely from its lexical form. This observation, together with the morpheme decomposition of many rare words, implies the feasibility of inferring their vectors from those for common words, and also raises the algorithmic question of how to compute them efficiently.

Second, there are many NLP applications where estimating word embeddings of OOV is critical. For instance, in the case of analyzing Twitter data, while there exists pre-trained word embeddings with giant vocabularies trained on massive number of tweets, such as GloVe vectors (Pennington et al., 2014), this would still not cover new words coined by users everyday. In such cases, it is more prudent to extend the available pre-trained vectors trained on very large corpora, so that we can estimate embeddings for OOV words, instead of retraining a new word / subword level embedding model on the new extended data corpus.

OOV words have always been a problem for methods that assume fixed vocabularies. A common workaround is to view all OOV words as a special UNK token and use the same vector for all of them. This would restrict any downstream models from accessing distinct features of those words. Thus, we would like a method to provide vectors that capture semantic and grammatical fea-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 601–606
Brussels, Belgium, October 31 - November 4, 2018.

tures even for OOV words. We also would like such method to maximally rely on the word itself, instead of its context, as contextual information is already used later with sentence level models stacking over word vectors.

To achieve this, we aim to build a word embedding model that generalizes pre-trained word embeddings to OOV words. First, given word embeddings for a fixed vocabulary, our model learns the relationship between the subwords present in each word and its corresponding pre-trained word vector. Then, using the learned subword information, our model can generate word embeddings for any word, regardless if it is OOV or not.

Contribution We propose a simple yet effective subword-level word embedding method that can be efficiently trained given pre-trained word vectors for a limited number of words. Once trained, our embedding model takes the characters n-grams in a word as input and gives its word vector as output.[1]

Our experiments on word similarity tasks in English and POS tagging in a variety of languages suggests that the proposed word embedder is able to mimic and generalize consistently the word vectors from in-vocabulary words to out-of-vocabulary words, and achieves state-of-the-art scores for the tasks compared to previous subword-level word embedders trained under the same setting. This gives evidence that such a simple model is capable of capturing language speaker's morphological knowledge, and also provides an easy way to generate word vectors for rare or unseen (OOV) words with potential application to various natural language processing tasks.

Related work There exist a large body of works that try to incorporate morphological information into word representations, e.g., (Alexandrescu and Kirchhoff, 2006; Luong et al., 2013a; Qiu et al., 2014; Botha and Blunsom, 2014; Cotterell and Schütze, 2015; Soricut and Och, 2015). These approaches typically rely on the morphological decomposition of words. Some other approaches using subword information do not rely on morphological decomposition but requires context information from large text corpus (Schütze, 1993; Santos and Zadrozny, 2014; Ling et al., 2015; Wieting et al., 2016).

In particular, Bojanowski et al. (2017) introduced fastText, a word embedding method enhanced with subword (character n-gram) embeddings. They are able to generate vectors for OOV words, which has been shown useful for text classification (Joulin et al., 2016), but the model is to be trained over large text corpus.

Pinter et al. (2017) use a character-level bidirectional LSTM model called MIMICK, mapping from word strings to word vectors. The idea of using character-level recurrent neural networks (RNNs) for word vectors is not new (Ling et al., 2015; Plank et al., 2016), but as per authors' knowledge, they are by far the only attempt to the exact task of generalizing word vectors from only pre-trained vectors with a fixed vocabulary, i.e. with no access to contextual information.

2 Bag-of-Substring Model

Our Bag-of-Substring (BoS) word vector generation model views a word as a bag of its substrings, or character n-grams. Specifically, we maintain a vector lookup table for each possible substrings (or character n-grams) of length between l_{min} and l_{max}. A word vector is then formed as the average of vectors of all its substrings with lengths in the range. Let Σ be the finite set of characters in the language, $subs_a^b(s) = \{t \text{ is substring of } s : a \leq |t| \leq b\}$ for string $s \in \Sigma^*$ be the set of substrings of s whose length is between a and b inclusive, and $<s>$ be the concatenation of character $<$, string s and character $>$ where $<, > \notin \Sigma$. The BoS embedding for a string/word s can be expressed as

$$\text{BoS}(s; V) = \frac{1}{|\mathcal{S}_{<s>}|} \sum_{t \in \mathcal{S}_{<s>}} v_t, \qquad (1)$$

where $V \in \mathbb{R}^{d \times (|\Sigma|^{l_{min}} + \cdots + |\Sigma|^{l_{max}})}$ are the parameters which stores the embeddings of dimension d for each possible substring of length between l_{min} and l_{max}, v_t is the vector in V indexed by t, $\mathcal{S}_{<s>}$ is a shorthand for $subs_{l_{min}}^{l_{max}}(<s>)$. Special characters $<, > \notin \Sigma$ are used to mark the start and the end of the word and thus help the model to distinguish homographic morphemes that occur at different word parts, e.g. prefixes or suffixes. An example BoS representation for word `infix` is $subs_3^4$(`<infix>`) = {`<in`, `<inf`, `inf`, `infi`, `nfi`, `nfix`, `fix`, `fix>`, `ix>`}.

fastText (Bojanowski et al., 2017) uses the same idea for their word vector generation part. How-

---

[1] The code is available at `https://github.com/jmzhao/bag-of-substring-embedder`.

ever, unlike them, we train the model directly towards pre-trained vectors, instead of via context prediction over text corpora.

Training Given pre-trained vectors for a set of common words, our model views them as targets and is trained to fit these targets. Once the parameters (the vectors v_t for the substrings) are learned, the model can then be used to infer vectors for rare words. Let $U \in \mathbb{R}^{d\times|W|}$ be the target vectors of the same dimension d over finite vocabulary $W \subset \Sigma^*$. Our model is trained by minimizing the overall loss between the generated and the given vectors for each word:

$$\underset{V}{\text{minimize}} \frac{1}{|W|} \sum_{w\in W} l(BoS(w;V), u_w) \qquad (2)$$

where the loss function $l(v,u) = \frac{1}{2}\|v-u\|_2^2$, namely the mean squared loss.

After training, one can use the learned V and Eqn (1) to compute the vector for any given word, even if it is OOV.

Hyperparameters We set the following hyperparameters for all the experiments. For BoS model, $l_{min} = 3$ and $l_{max} = 6$ following Bojanowski et al. (2017). Note that under this setting, $\mathcal{S}_{<s>}$ can never be empty for non-empty string s. For optimization, stochastic gradient descent with learning rate 1 for 100 epochs. The dimension of the word vectors is not a hyperparameters here as it needs to agree with the target vector.

3 Word Similarity

We run experiments to quantitatively evaluate the our model's generalizability towards OOV words.

The word similarity task asks to predict word similarity between a pair of two words. Given a set of pairs of words and gold labels for their similarities, the performance of word embeddings is measured by the correlation between the gold similarities and the similarities induced by the generated embeddings. And we can thus imply how good our model is at generating word vectors. The word similarity here is computed using the cosine distance between the two word vectors, and the correlation is computed using Spearman's ρ.

Datasets We evaluate over Stanford RareWord (RW) introduced by Luong et al. (2013b) and WordSim353 (WS) introduced by Finkelstein et al. (2001). RW consists of less common words so we use it to access our model's ability to generalize word embeddings to OOV words. WS is composed of mostly common words and we use it to test if our subword-level models successfully mimic the target vectors.

	Dim.	# Tokens	RW	WS
Polyglot	64	100k	41(58%)	45(5%)
Google	300	160k	53(11%)	69(1%)

Table 1: Target vectors statistics and word similarity task scores in Spearman's $\rho \times 100$. In parentheses are OOV rates.

Model	Size	Target	RW	WS
EditDist	-	-	18	-2
MIMICK	649KB	Polyglot	14	12
BoS	238MB	Polyglot	36	36
BoS	1.3GB	Google	46	56
fastText	8.0GB	-	48	74

Table 2: Word similarity task results measured in Spearman's $\rho \times 100$.

Target vectors We train our BoS model over the English Polyglot vectors [2] to establish a direct comparison with results from MIMICK (Pinter et al., 2017), and as well as the Google word2vec vectors [3] which are popularly used in NLP tasks. Polyglot (Al-Rfou et al., 2013) is a multilingual NLP dataset, which also provides pre-trained word vectors over each language's corpus with a vocabulary of 100,000 most frequent words. For Google vectors, most of their vocabulary consists of non-words such as URLs and phrases, so we normalize tokens into ASCII characters by taking off all the diacritics and take only tokens consisting of a single word with all lower letters. Statistics of the processed vectors are summarized in Table 1, along with their word similarity task scores (for in-vocabulary words only) and OOV rate over the aforementioned evaluation sets.

Baselines We compare the scores with other subword-level models (fastText and MIMICK) and word similarity induced by non-parametric edit distance (EditDist).

fastText (Bojanowski et al., 2017) uses the same subword-level character n-gram model but is to be trained via context prediction over large text corpora (here English Wikipedia dump [4]). MIM-

---

[2] `http://polyglot.readthedocs.io/en/latest/Download.html`

[3] `https://code.google.com/archive/p/word2vec/`

[4] `https://fasttext.cc/docs/en/pretrained-vectors.html`

ICK (Pinter et al., 2017) is a character-level bidirectional LSTM word embedder trained against pre-trained word vectors (here Polyglot vectors [5]).

Edit distance is defined between two strings as the smallest number of modifications: adding, deleting and changing one character, needed to turn one string into the other. It can be computed using dynamic programming in $O(|s_1| \times |s_2|)$ time. The word similarity between w_1 and w_2 here is the edit distance normalized by the length of the longer word:

$$s_{EditDist}(w_1, w_2) = -\frac{d_{edit}(w_1, w_2)}{\max(|w_1|, |w_2|)} \quad (3)$$

where d_{edit} is edit distance.

Results Results are summarized in Table 2. When trained over Polyglot vectors, our BoS model works better than EditDist and MIMICK. When trained on Google vectors, the correlation scores are almost as good as those of fastText, the state-of-the-art subword level word embedder. However, unlike fastText, our model does not have access to word contexts in a large text corpus for training. In both cases, the significant differences of scores compared to those of EditDist, suggest that our model indeed learns to capture semantic similarities between words, rather than superficial similarities in spelling.

Comparing to MIMICK, our model is able to fill up 81% (14 to 36 against 41) and 73% (12 to 36 against 45) of the gaps in scores over RW and WS respectively. This improvement is more significant on RW with most (58%) of its words are OOV for the PloyGlot vectors, suggesting our model's power in generating consistent word vectors for OOV words. Surprisingly MIMICK performs no better than the edit distance baseline when evaluated on RW. Combined with the fact that it does no better for WS which has a near-zero OOV rate, it suggests MIMICK's limited power of generalizing word vectors towards OOV words, or even reproduce consistent word vector for in-vocabulary words. As a sanity check, we see that all of the embedder models scores obviously better than EditDist when evaluated over common words (WS), showing that all of them are able to at least remember or mimic the word vectors for in-vocabulary words.

Also note that our model is fast to train. With a naive single-thread CPU-only Python implementation, it can finish 100 epochs of training over English PolyGlot vectors within 352 seconds on a machine with an Intel Core i7-6700 (3.4 GHz) CPU, 32GB memory and 1TB SSD. Compared to fastText which, with a fast multithread C++ implementation, takes hours to be trained over giga bytes of text corpus, our method provides a cheap way to generalize reasonably good word vectors for OOV words.

4 Joint Prediction of Part-of-Speech Tags and Morphosyntactic Attributes

Besides word similarity, we try to access our embedders' ability of capturing words' syntactic and semantic features by evaluating with the task of predicting part-of-speech (POS) tags and morphosyntactic attributes for words in a sentence. For each word in a given sentence, the task asks for a POS tag and a label for each applicable morphosyntactic category, such as gender, case or tense.

Dataset We use Universal Dependencies (UD) dataset (Petrov et al., 2012) for this task. UD is an open-community effort to build consistent annotated treebank cross many languages. We pick the specific version 1.4 to enable a direct comparison with Pinter et al. (2017). Since we use PolyGlot vectors to train our word embedders, we conduct experiments on the 23 languages that appear in both Polyglot and UD 1.4.

Model We adopt the same sentence-level bidirectional LSTM model from Pinter et al. (2017) for the joint prediction of both labels. Given a sentence as a sequence of words, we first embed each word using the word embedder we choose and then fed the embeddings into the LSTM. The output of LSTM is then used to predict POS and morphosyntactic tags.

We emphasize the difference in the setting that we *fix* the word embeddings during the training, as to better evaluate the ability and consistency of the embeddings in capturing words' semantics and syntactics, rather than LSTM's ability to memorize words and infer the role of words from their context.

We use the same set of hyperparameters for the LSTM model as Sec. 5.3 in Pinter et al. (2017) and train the model for 20 epochs for each language. The BoS and MIMICK word embedders

---

[5] `https://github.com/yuvalpinter/Mimick`

	N_{train}	POS tagging			Morphosyntactic attributes		
		random	MIMKCK	BoS	random	MIMICK	BoS
Kazakh	4,949	0.589	0.681	**0.758**(0.077)	0.021	0.032	**0.240**(0.208)
Tamil	6,329	0.480	0.678	**0.774**(0.097)	0.568	0.673	**0.762**(0.089)
Latvian	13,781	0.589	0.757	**0.872**(0.115)	0.374	0.572	**0.676**(0.104)
Vietnamese	31,800	0.749	0.564	**0.846**(0.282)	-	-	-
Hungarian	33,017	0.594	0.858	**0.922**(0.065)	0.569	0.775	**0.836**(0.061)
Turkish	41,748	0.636	0.767	**0.890**(0.123)	0.543	0.776	**0.826**(0.050)
Greek	47,449	0.819	0.907	**0.965**(0.058)	0.783	0.903	**0.934**(0.031)
Bulgarian	50,000	0.804	0.903	**0.971**(0.068)	0.649	0.851	**0.915**(0.064)
Swedish	66,645	0.748	0.813	**0.945**(0.132)	0.707	0.812	**0.930**(0.118)
Basque	72,974	0.662	0.823	**0.913**(0.091)	0.564	0.778	**0.820**(0.042)
Russian	79,772	0.665	0.897	**0.948**(0.051)	0.592	0.855	**0.915**(0.060)
Danish	88,980	0.788	0.834	**0.947**(0.114)	0.745	0.813	**0.927**(0.114)
Indonesian	97,531	0.724	0.788	**0.915**(0.127)	-	-	-
Chinese	98,608	0.721	0.793	**0.835**(0.042)	0.699	0.767	**0.790**(0.022)
Persian	121,064	0.843	0.866	**0.957**(0.091)	0.745	0.792	**0.918**(0.125)
Hebrew	135,496	0.814	0.858	**0.957**(0.099)	0.648	0.837	**0.903**(0.066)
Romanian	163,262	0.796	0.874	**0.956**(0.082)	0.718	0.876	**0.942**(0.066)
English	204,587	0.770	0.826	**0.932**(0.106)	0.822	0.859	**0.947**(0.089)
Arabic	225,853	0.780	0.901	**0.950**(0.049)	0.711	0.901	**0.942**(0.041)
Hindi	281,057	0.824	0.848	**0.939**(0.091)	0.863	0.888	**0.951**(0.063)
Italian	289,440	0.810	0.909	**0.964**(0.056)	0.839	0.927	**0.964**(0.037)
Spanish	382,436	0.819	0.914	**0.959**(0.045)	0.793	0.915	**0.954**(0.038)
Czech	1,173,282	0.695	0.908	**0.966**(0.058)	0.622	0.845	**0.905**(0.061)

Table 3: POS tagging accuracy and morphosyntactic attributes micro F1 over 23 languages (UD 1.4). In parentheses are the gains to MIMICK.

are trained beforehand with PolyGlot dataset using the same way described earlier.

Results The POS tagging accuracies and micro F1 scores for morphosyntactic attributes are reported in Table 3 with word vectors generated by different models. The BoS and MIMICK model here are trained against Polyglot vectors. As a comparison, we include the results using random word vectors of the same dimension (64).

Our BoS model shows steady and significant gain compared to MIMICK embeddings for both tasks in *all* languages. We especially observe the greatest margins for agglutinative languages such as Turkish and Indonesian, and in Germanic languages English, Swedish and Danish, suggesting that our model learns stable representations for morphemes to consistent word type signal.

5 Conclusion

We proposed a subword-level word embedding model and a word vector generalization method that enables extending pre-trained word embeddings with fixed size vocabularies to estimate word embeddings for out-of-vocabulary words. Intrinsic evaluation on word similarity tasks and extrinsic evaluation on POS tagging task demonstrate that our model captures morphological knowledge and generates good estimates of word vectors for OOV words.

Acknowledgements

We would like to thank the anonymous reviewers for helpful comments. This work was supported in part by FA9550-18-1-0166. Y. L. would also like to acknowledge that support for this research was provided by the Office of the Vice Chancellor for Research and Graduate Education at the University of Wisconsin-Madison with funding from the Wisconsin Alumni Research Foundation.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *CoNLL*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

Andrei Alexandrescu and Katrin Kirchhoff. 2006. Factored neural language models. In *NAACL-HLT*, pages 1–4.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *ICML*, pages 1899–1907.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *NAACL-HLT*, pages 1287–1292.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*, pages 406–414. ACM.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*, pages 1520–1530.

Thang Luong, Richard Socher, and Christopher Manning. 2013a. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013b. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113. ACL.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC*.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *EMNLP*, pages 102–112. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *ACL*, page 412.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *COLING*, pages 141–150.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.

Hinrich Schütze. 1993. Word space. In *NIPS*, pages 895–902.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *NAACL*, pages 1627–1637.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *EMNLP*, pages 1504–1515.

Neural Metaphor Detection in Context

Ge Gao,$^\diamondsuit$ **Eunsol Choi,**$^\diamondsuit$ **Yejin Choi,**$^{\diamondsuit\dagger}$ **and Luke Zettlemoyer**$^\diamondsuit$
University of Washington$^\diamondsuit$
Allen Institute for Artificial Intelligence†
`{ggao,eunsol,yejin,lsz}@cs.washington.edu`

Abstract

We present end-to-end neural models for detecting metaphorical word use in context. We show that relatively standard BiLSTM models which operate on complete sentences work well in this setting, in comparison to previous work that used more restricted forms of linguistic context. These models establish a new state-of-the-art on existing verb metaphor detection benchmarks, and show strong performance on jointly predicting the metaphoricity of all words in a running text.

1 Introduction

Metaphors are pervasive in natural language, and detecting them requires challenging contextual reasoning about whether specific situations can actually happen. (Lakoff and Johnson, 1980). For example, in Table 1, “examining” is metaphorical because it is impossible to literally use a “microscope” to examine an entire country. In this paper, we present end-to-end neural models for metaphor detection, which can learn rich contextual word representations that are crucial for accurate interpretation of figurative language.

In contrast, most previous approaches focused on limited forms of linguistic context, for example by only providing SVO triples such as (car, **drink**, gasoline) to the model (Shutova et al., 2016; Tsvetkov et al., 2013; Rei et al., 2017; Bulat et al., 2017). While the verbal arguments provide strong cues, providing the full sentential context supports more accurate prediction, as seen in Table 1. Even in the few cases when the full sentence is used (Köper and im Walde, 2017; Turney et al., 2011; Jang et al., 2016) existing models have used unigram-based features with limited expressivity.

We investigate two common task formulations: (1) given a target verb in a sentence, classifying whether it is metaphorical or not, and (2) given a sentence, detecting all of the metaphorical words (independent of their POS tags). We find that relatively standard architectures based on bi-directional LSTMs (Hochreiter and Schmidhuber, 1997) augmented with contextualized word embeddings (Peters et al., 2018) perform surprisingly well on both tasks, even with modest amount of training data. We improve the previous state-of-the-art by 7.5 F1 on the VU Amsterdam Metaphor Corpus (VUA) for the sequence labeling task (Steen et al., 2010), by 2.5 F1 on the VUA verb clasificiation dataset, and by 4.9 F1 on the MOH-X dataset (Mohammad et al., 2016). Our code is publicly available at `https://github.com/gao-g/metaphor-in-context`.

The experts started **examining** the Soviet Union with a microscope to study perceived changes.
Rockford teachers are honored for saving a *drowning* student.
You’re **drowning** in student loan debt.

Table 1: Metaphorical usages of the target word are bold faced, and literal usages are italicized. Full sentence context is crucial for metaphor detection.

2 Task

We study two task formulations.

Sequence Labeling: Given a sentence $x_1,\ldots,x_n$, predict a sequence of binary labels $l_1,\ldots,l_n$ to indicate the metaphoricity of each word.

Classification: Given a sentence $x_1,\ldots,x_n$ and a target verb index i, predict a binary label l to indicate the metaphoricity of the target x_i.

While both formulations have been studied in previous work, it is worth noting that the sequence labeling task generalizes the classification task in

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 607–613
Brussels, Belgium, October 31 - November 4, 2018.

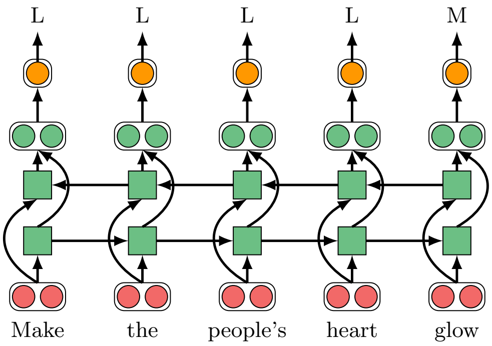

Figure 1: A sequence labeling model for metaphor detection. Every word in a sentence is classified.

that the prediction for the target verb can be extracted from the full sentence predictions. In addition, as will be shown in Section 5, we find that given accurate annotations for all words in a sentence, the sequence labeling model outperforms the classification model even when the evaluation is set up as a classification task.

3 Model

Our models use a bidirectional LSTM to encode a sentence, and a feedforward neural network for classification, optimized for the log-likelihood of gold labels.

Sentence encoding For both sequence labeling and classification, we represent each token x_i in the input sentence with a pre-trained word embedding w_i. To further encode contextual information, we also concatenate ELMo (Embeddings from Language Models) vectors e_i from Peters et al. (2018). These vectors have been shown to be useful for word sense disambiguation, a task closely related to metaphor detection (Birke and Sarkar, 2006).

3.1 Sequence Labeling Model

Figure 1 shows the model architecture. We input the word representation $[w_i; e_i]$ to a bidirectional LSTM, producing a contextualized representation h_i for each token. Then we use a feedforward neural network that takes h_i to predict a label l_i for each word x_i.

When the dataset does not contain annotations for every word, we make the simplifying assumption that every unannotated word is used literally.

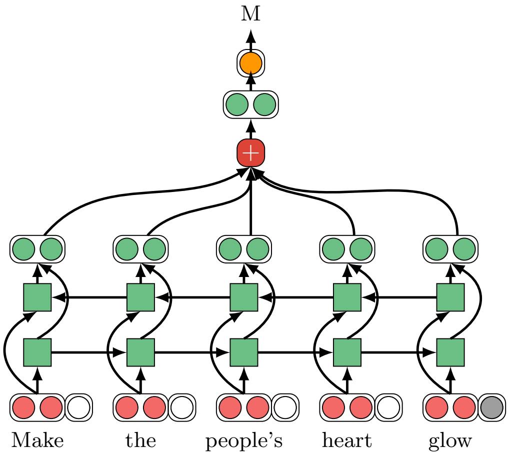

Figure 2: A classification model for metaphor detection. Only a single word per sentence is labeled as metaphorical or literal.

3.2 Classification Model

Figure 2 shows the model architecture. We concatenate an index embedding n_i, which indicates whether x_i is the target verb. We use $[w_i; e_i; n_i]$ as an input to a bidirectional LSTM, producing a contextualized representation h_i.

We add an attention layer by computing the attention weight a_i for token x_i, and compute the representation c as a weighted sum of LSTM output states where W_a and b_a are learned parameters.

$$a_i = \text{SoftMax}_i(W_a h_i + b_a)$$

$$c = \sum_{i=1}^{n} a_i h_i$$

Finally, we feed c to a feedforward network to compute the label scores for target verb.

4 Dataset

We evaluate performance on a number of benchmark datasets, including two for classification (TroFi and MOH\MOH-X) and one for tagging (VUA).[1] Table 2 shows statistics for the verb classification datasets. Despite being two times larger than the MOH dataset, the TroFi dataset contains only 50 unique verbs, and the larger VUA dataset contains over 2K unique verbs. The MOH dataset contains shorter and simpler sentences (example sentences in WordNet), compared to sentences in other datasets which come from resources such as

---

[1] For detailed information about each dataset, please refer to original papers: TroFi (Birke and Sarkar, 2006), MOH (Mohammad et al., 2016), VUA (Steen et al., 2010). MOH-X refers to a subset of MOH dataset used in previous work (Shutova et al., 2016) where verb and its argument are extracted from each sentence.

	# Expl.	% Metaphor	# Uniq. Verb	Avg # Sent. Len
MOH-X	647	49%	214	8.0
MOH	1,639	25%	440	7.4
TroFi	3,737	43%	50	28.3
VUA	23,113	28%	2047	24.5

Table 2: Verb classification dataset statistics. % Metaphor refers to sentence-level percentage.

	Train	Dev	Test
# Unique tokens	13,843	7,458	7,200
# Tokens	116,622	38,628	50,175
# Unique sent.	6,323	1,550	2,694
% Metaphor	11.2	11.6	12.4

Table 3: VUA sequence labeling dataset statistics. % Metaphor refers to token-level percentage.

Model	P	R	F1	Acc.
Lexical Baseline	68.6	45.2	54.5	90.6
Wu (2018) ensemble	60.8	70.0	65.1	-
Ours (SEQ)	**71.6**	**73.6**	**72.6**	**93.1**

Table 4: Performance on the VUA sequence labeling test set for all POS tags.

POS	#	% metaphor	P	R	F1.
VERB	**20K**	18.1	68.1	71.9	69.9
NOUN	20K	13.6	59.9	60.8	60.4
ADP	13K	**28.0**	**86.8**	**89.0**	**87.9**
ADJ	9K	11.5	56.1	60.6	58.3
PART	3K	10.1	57.1	59.1	58.1

Table 5: The breakdown of performance on the VUA sequence labeling test set by POS tags. We show data statistics (count, % metaphor) on the training set. We only show POS tags whose % metaphor >10.

news articles. The TroFi and MOH-X datasets are constructed to have higher percentages of metaphor, compared to the natural likelihood of metaphor in a running text, as seen in the VUA dataset.

Classification Experiment Setup We perform 10 fold cross-validation on the MOH-X and TroFi datasets, following prior work. For the VUA dataset, we use the original training and test split (Klebanov et al., 2016), and set aside 10% of the training set as a development set.

Sequence Labeling Experiment Setup The VUA dataset contains annotations for all words in each sentence. We divide the data into training, development, and test set following the same split for the VUA verb classification task. While the label classes are less balanced (only 11% metaphors at the token level), this dataset is much bigger. Table 3 shows the data statistics.

5 Experiments

Evaluation Metric We report precision, recall and F1 measure for the metaphor class as well as the overall accuracy. For the VUA dataset, we also report macro-averaged F1 score across four genres (conversation, academic writing, fiction and news).

Comparison Systems We propose a simple yet effective lexical baseline. It assigns the metaphor label if the word is annotated metaphorically more frequently than as literally in the training set, and the literal label otherwise. We also compare our models to previously published work, including: (1) a logistic regression classifier with features that indicate verb lemmas and the verbs' semantic class from WordNet (Klebanov et al., 2016), (2) a neural similarity network with skip-gram word embeddings (Rei et al., 2017), (3) a balanced logistic regression classifier on target verb lemma that uses a set of features based on multi-sense abstractness rating (Köper and im Walde, 2017), and (4) a CNN-LSTM ensemble model with weighted-softmax classifier which incorporates pre-trained word2vec, POS tags, and word cluster features (Wu et al., 2018).[2]

We experiment with both sequence labeling model (SEQ) and classification model (CLS) for the verb classification task, and the sequence labeling model (SEQ) for the sequence labeling task.

Implementation Details We used 300d GloVe vectors (Pennington et al., 2014) and 1024d ELMo vectors. We used additional 50d index embedding for the classification task. The LSTM module has a 300d hidden state. We applied dropout on the input to LSTM and on the input to the feedforward layer. We fine-tuned learning rate and dropout rate for each model on each dataset. We used SGD to optimize the CLS model and Adam (Kingma and Ba, 2013) for the SEQ model. We used spaCy (Honnibal and Montani, 2017) for lemmatization, tokenization, and part-of-speech tagging.

---

[2] The best performing model on the VUA Metaphor Detection Shared Task at the NAACL 2018 workshop on Figurative Language Processing.

Model	MOH-X (10 fold)				TroFi (10 fold)				VUA - Test				
	P	R	F1	Acc.	P	R	F1	Acc.	P	R	F1	Acc.	MaF1
Lexical Baseline	39.1	26.7	31.3	43.6	**72.4**	55.7	62.9	71.4	67.9	40.7	50.9	76.4	48.9
Klebanov (2016)	-	-	-	-	-	-	-	-	-	-	-	-	60.0
Rei (2017)	73.6	76.1	74.2	74.8	-	-	-	-	-	-	-	-	-
Köper (2017)	-	-	-	-	-		**75.0**	-	-	-	62.0	-	-
Wu (2018) ensemble	-	-	-	-	-	-	-	-	60.0	**76.3**	67.2	-	-
CLS	75.3	84.3	**79.1**	**78.5**	68.7	**74.6**	72.0	73.7	53.4	65.6	58.9	69.1	53.4
SEQ	**79.1**	73.5	75.6	77.2	70.7	71.6	71.1	**74.6**	**68.2**	71.3	**69.7**	**81.4**	**66.4**

Table 6: Model performances for the verb classification task. Our models achieve strong performance on all datasets. The CLS model performs better than the SEQ model when only one word per sentence is annotated by human (TroFi and MOH-X). When all words in the sentence are accurately annotated (VUA), the SEQ model outperforms the CLS model.

Model	P	R	F1.	Acc.
SEQ	**68.3**	**72.0**	**70.4**	**83.5**
-ELMo	59.4	64.3	61.7	78.2
CLS	52.4	63.0	57.3	74.3
-ELMo	52.0	48.7	50.8	74.1

Table 7: Ablation study on VUA development set for the verb classification task.

Sequence Labeling Results Performance on the sequence labeling task is reported in Table 4. While prior work (Klebanov et al., 2014; Özbal et al., 2016) reported on the same dataset, the experiment setting is not comparable (they did cross validation on a smaller training set).[3] Our lexical baseline performs strongly in terms of precision, as some words and POS tags are almost exclusively annotated as literal. Our sequence labeling model mainly improves recall.

Table 5 reports the breakdown of performance by POS tags. Not surprisingly, tags with more data are easier to classify. Adposition is the easiest to identify as metaphorical and is also the most frequently metaphorical class (28%). On the other hand, particles are challenging to identify, since they are often associated with multi-word expressions, such as "put **down** the disturbances".

Verb Classification Results Table 6 shows performance on the verb classification task for three datasets (MOH-X , TroFi and VUA).[4]

Our models achieve strong performance on all datasets, outperforming existing models on the MOH-X and VUA datasets. On the MOH-X dataset, the CLS model outperforms the SEQ model, likely due to the simpler overall sentence structure and the fact that the target verbs are the only words annotated for metaphoricity. For the VUA dataset, where we have annotations for all words in a sentence, the SEQ model significantly outperforms the CLS model. This result shows that predicting metaphor labels of context words helps to predict the target verb. We hypothesize that Köper et al. (2017) outperforms our models on the TroFi dataset for a similar reason: their work uses concreteness labels, which highly correlate to metaphor labels of neighboring words in the sentence. Also, their best model uses the verb lemma as a feature, which itself provides a strong clue in the dataset of 50 verbs (see lexical baseline).

Table 7 shows an ablation study on input representations (with or without ELMo vectors). Contextualized word vectors improve the performance of both models by a large margin.

Error Analysis We sampled 100 errors of our best model from the VUA verb classification development set for analysis. Table 8 shows examples. Following the original annotation guideline,[5] we classify metaphors into five categories: direct metaphor, indirect metaphor, implicit metaphor, personification, and borderline case. Indirect metaphor, the most common type for verbs, means that the basic meaning of a word is different from its contextual meaning. Implicit metaphor occurs due to an underlying link which points to a recoverable metaphorical concept.

About half of the errors were false positives, and the other half were false negatives. Among the false negatives, 33% are indirect metaphors, 18% are personifications, and 2% are direct metaphors. Among 55 false positives, 31% of verbs have im-

---

[3] As a point of reference, their macro-averaged F1 scores were 33.25 / 50.6 respectively.

[4] We did not compare to Shutova et al. (2016) as their experiment setting is not comparable.

[5] http://www.vismet.org/metcor/documentation/home.html

CLS	SEQ	Sentence	Metaphor Type
✗	✗	To *throw* up an impenetrable Berlin Wall between you and them could be tactless.	-
✗	✗	In reality you just invent a tale, as if you were **sitting** round a fire in a cave.	direct metaphor
✗	✗	So they **bought** immunity.	indirect metaphor
✗	✗	During the early states of the phased evacuation the logistical problem **facing** the police was the street-by-street warning of the population to make ready for evacuation.	indirect metaphor
✗	✔	There are few things worse than being **bludgeoned** into reading a book you hate.	indirect metaphor
✗	✔	He thought of thick, fat, hot motorways **carving** up that land.	personification
✗	✔	One might *ask* whether motorists are ever justified in knowingly taking risks with other people's lives.	-
✗	✔	The abstract talk of *commuting* by rail or road being replaced by information technology finds a concrete expression in the idea of telecottages.	-
✗	✔	A fly landed on the empty, staring vizor, and *crawled* across it.	-

Table 8: Some examples from the VUA verb classification development set. Metaphorical usages of the target word are bold faced, and literal usages are italicized. Leftmost columns show the correctness of prediction.

plicit arguments that are not explicitly mentioned in the context, 15% have long range dependencies (at least five words away) from core arguments, 10% have arguments with rare word senses, and 5% have anthropomorphic arguments. Finally, we found about half of false negatives and 20% of false positives to be borderline cases, showing the subjective nature of the task.

We sampled 257 dev examples that the CLS model gets wrong but the SEQ model gets correct. We found that the SEQ model outperforms the CLS model on detecting personifications, indirect metaphors, and direct metaphors involving uncommon verbs.

6 Related Work

There has been significant work on studying different features for metaphor detection, including concretenesss and abstractness (Turney et al., 2011; Tsvetkov et al., 2014; Köper and im Walde, 2017), imaginability (Broadwell et al., 2013; Strzalkowski et al., 2013), feature norms (Bulat et al., 2017), sensory features (Tekiroglu et al., 2015; Shutova et al., 2016), bag-of-words features (Köper and im Walde, 2016), and semantic class using WordNet (Hovy et al., 2013; Tsvetkov et al., 2014). More recently, embedding-based approaches (Köper and im Walde, 2017; Rei et al., 2017) showed gains on various benchmarks.

Many neural models with various features and architectures were introduced in the 2018 VUA Metaphor Detection Shared Task. They include LSTM-based models and CRFs augmented by linguistic features, such as WordNet, POS tags, concreteness score, unigrams, lemmas, verb clusters, and sentence-length manipulation (Swarnkar and Singh, 2018; Pramanick et al., 2018; Mosolova et al., 2018; Bizzoni and Ghanimifard, 2018; Wu et al., 2018). Researchers also studied different word embeddings, such as embeddings trained from corpora representing different levels of language mastery (Stemle and Onysko, 2018) and binarized vectors that reflect the General Inquirer dictionary category of a word (Mykowiecka et al., 2018). We show that contextualized word embedding significantly improves metaphor detection. We also study both sequence labeling and classification approaches, suggesting that sequence labeling approach enhances performance when used to jointly predict the metaphoricity of all words in a sentence.

7 Conclusion

In this paper, we present simple biLSTM models augmented with contextualized word representation for metaphor detection. Our models establish new state-of-the-arts across multiple existing benchmarks, and our error analysis shows remaining challenges for metaphor detection.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work was supported in part by the NSF (IIS-1714566 and IIS-1252835), the ARO (W911NF-16-1-0121), the DARPA CwC program through ARO (W911NF-15-1-0543), and gifts from Google and Facebook.

References

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *EACL*.

Yuri Bizzoni and Mehdi Ghanimifard. 2018. Bigrams and bilstms two neural networks for sequential metaphor detection. In *Proceedings of the Workshop on Figurative Language Processing, NAACL*, pages 91–101.

George Aaron Broadwell, Umit Boz, Ignacio Cases, Tomek Strzalkowski, Laurie Feldman, Sarah M. Taylor, Samira Shaikh, Ting Liu, Kit Cho, and Nick Webb. 2013. Using imageability and topic chaining to locate metaphors in linguistic corpora. In *SBP*.

Luana Bulat, Stephen Clark, and Ekaterina Shutova. 2017. Modelling metaphor with attribute-based semantics. In *EACL*.

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

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing.

Dirk Hovy, Shashank Shrivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huying Li, Whitney Sanders, and Eduard H. Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proceedings of the First Workshop on Metaphor in NLP, ACL*, pages 52–57.

Hyeju Jang, Yohan Jo, Qinlan Shen, Michael Miller, Seungwhan Moon, and Carolyn Penstein Rosé. 2016. Metaphor detection with topic transition, emotion and cognition in context. In *ACL*.

Diederik P Kingma and Jimmy Ba. 2013. Adam: A method for stochastic optimization.

Beata Beigman Klebanov, Chee Wee Leong, E. Dario Gutiérrez, Ekaterina Shutova, and Michael Flor. 2016. Semantic classifications for detection of verb metaphors. In *ACL*.

Beata Beigman Klebanov, Chee Wee Leong, Michael Heilman, and Michael Flor. 2014. Different texts, same metaphors: Unigrams and beyond. In *Proceedings of the Second Workshop on Metaphor in NLP, ACL*.

Maximilian Köper and Sabine Schulte im Walde. 2016. Distinguishing literal and non-literal usage of german particle verbs. In *HLT-NAACL*.

Maximilian Köper and Sabine Schulte im Walde. 2017. Improving verb metaphor detection by propagating abstractness to words, phrases and individual senses. In *Proceedings of the First Workshop on Sense, Concept and Entity Representations and their Applications*, pages 24–30.

George Lakoff and Mark Johnson. 1980. Metaphors we live by. The University of Chicago Press.

Saif Mohammad, Ekaterina Shutova, and Peter D. Turney. 2016. Metaphor as a medium for emotion: An empirical study. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics (*Sem)*.

Anna Mosolova, Ivan Bondarenko, and Vadim Fomin. 2018. Conditional random fields for metaphor detection. In *Proceedings of the Workshop on Figurative Language Processing, NAACL*, pages 121–123.

Agnieszka Mykowiecka, Aleksander Wawer, and Magorzata Aneta Marciniak. 2018. Detecting figurative word occurrences using recurrent neural networks. In *Proceedings of the Workshop on Figurative Language Processing, NAACL*, pages 124–127.

Gözde Özbal, Carlo Strapparava, Serra Sinem Tekiroglu, and Daniele Pighin. 2016. Learning to identify metaphors from a corpus of proverbs. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matthew Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. abs/1802.05365.

Malay Pramanick, Ashim Gupta, and Pabitra Mitra. 2018. An lstm-crf based approach to token-level metaphor detection. In *Proceedings of the Workshop on Figurative Language Processing, NAACL*, pages 67–75.

Marek Rei, Luana Bulat, Douwe Kiela, and Ekaterina Shutova. 2017. Grasping the finer point: A supervised similarity network for metaphor detection. In *EMNLP*.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *HLT-NAACL*.

Gerard J Steen, Aletta G Dorst, J Berenike Herrmann, Anna A Kaal, and Tina Krennmayr. 2010. Metaphor in usage. 21(4):765796.

Egon W. Stemle and Alexander Onysko. 2018. Using language learner data for metaphor detection. In *Proceedings of the Workshop on Figurative Language Processing, NAACL*, pages 133–138.

Tomek Strzalkowski, George Aaron Broadwell, Sarah Taylor, Laurie Feldman, Samira Shaikh, Ting Liu, Boris Yamrom, Kit Cho, Umit Boz, Ignacio Cases, and Kyle Elliot. 2013. Robust extraction of metaphor from novel data. In *Proceedings of the First Workshop on Metaphor in NLP, ACL*.

Krishnkant Swarnkar and Anil Kumar Singh. 2018. Di-lstm contrast : A deep neural network for metaphor detection. In *Proceedings of the Workshop on Figurative Language Processing, NAACL*, pages 115–120.

Serra Sinem Tekiroglu, Gözde Özbal, and Carlo Strapparava. 2015. Exploring sensorial features for metaphor identification. In *Proceedings of the Third Workshop on Metaphor in NLP, ACL.*

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *ACL*.

Yulia Tsvetkov, Elena Mukomel, and Anatole Gershman. 2013. Cross-lingual metaphor detection using common semantic features. In *Proceedings of the First Workshop on Metaphor in NLP, ACL.*

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *EMNLP*.

Chuhan Wu, Fangzhao Wu, Yubo Chen, Sixing Wu, Zhigang Yuan, and Yongfeng Huang. 2018. Neural metaphor detecting with cnn-lstm model. In *Proceedings of the Workshop on Figurative Language Processing, NAACL*, pages 110–114.

Distant Supervision from Disparate Sources for Low-Resource Part-of-Speech Tagging

Barbara Plank and **Željko Agić**
Department of Computer Science
IT University of Copenhagen
Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark
`bplank@itu.dk, zeag@itu.dk`

Abstract

We introduce DsDs: a cross-lingual neural part-of-speech tagger that learns from disparate sources of distant supervision, and realistically scales to hundreds of low-resource languages. The model exploits annotation projection, instance selection, tag dictionaries, morphological lexicons, and distributed representations, all in a uniform framework. The approach is simple, yet surprisingly effective, resulting in a new state of the art *without* access to any gold annotated data.

1 Introduction

Low-resource languages lack manually annotated data to learn even the most basic models such as part-of-speech (POS) taggers. To compensate for the absence of direct supervision, work in cross-lingual learning and distant supervision has discovered creative use for a number of alternative data sources to learn feasible models:

- aligned *parallel corpora* to project POS annotations to target languages (Yarowsky et al., 2001; Agić et al., 2015; Fang and Cohn, 2016),
- noisy *tag dictionaries* for type-level approximation of full supervision (Li et al., 2012),
- combination of projection and *type constraints* (Das and Petrov, 2011; Täckström et al., 2013),
- rapid annotation of *seed training data* (Garrette and Baldridge, 2013; Garrette et al., 2013).

However, only one or two compatible sources of distant supervision are typically employed. In reality severely under-resourced languages may require a more pragmatic "take what you can get" viewpoint. Our results suggest that combining supervision sources is the way to go about creating viable low-resource taggers.

We propose a method to strike a balance between model simplicity and the capacity to easily integrate heterogeneous learning signals. Our system is a uniform neural model for POS tagging that learns from *disparate sources of distant supervision* (DsDs). We use it to combine: i) multi-source annotation projection, ii) instance selection, iii) noisy tag dictionaries, and iv) distributed word and sub-word representations. We examine how far we can get by exploiting only the wide-coverage resources that are currently readily available for more than 300 languages, which is the breadth of the parallel corpus we employ.

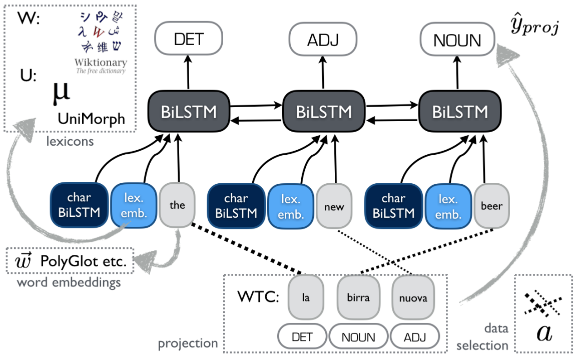

Figure 1: Illustration of DsDs (Distant Supervision from Disparate Sources).

DsDs yields a new state of the art by jointly leveraging disparate sources of distant supervision in an experiment with 25 languages. We demonstrate: i) substantial gains in carefully selecting high-quality instances in annotation projection, ii) the usefulness of lexicon features for neural tagging, and iii) the importance of word embeddings initialization for faster convergence.

2 Method

DsDs is illustrated in Figure 1. The base model is a bidirectional long short-term memory network (bi-LSTM) (Graves and Schmidhuber, 2005; Hochreiter and Schmidhuber, 1997; Plank et al., 2016; Kiperwasser and Goldberg, 2016). Let $x_{1:n}$

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 614–620
Brussels, Belgium, October 31 - November 4, 2018.

be a given sequence of input vectors. In our base model, the input sequence consists of word embeddings $\vec{w}$ and the two output states of a character-level bi-LSTM $\vec{c}$. Given $x_{1:n}$ and a desired index i, the function $BiRNN_\theta(x_{1:n}, i)$ (here instantiated as LSTM) reads the input sequence in forward and reverse order, respectively, and uses the concatenated ($\circ$) output states as input for tag prediction at position i.[1] Our model differs from prior work on the type of input vectors $x_{1:n}$ and distant data sources, in particular, we extend the input with lexicon embeddings, all described next.

Annotation projection. Ever since the seminal work of Yarowsky et al. (2001), projecting sequential labels from source to target languages has been one of the most prevalent approaches to cross-lingual learning. Its only requirement is that parallel texts are available between the languages, and that the source side is annotated for POS.

We apply the approach by Agić et al. (2016), where labels are projected from multiple sources and then decoded through weighted majority voting with word alignment probabilities and source POS tagger confidences. We exploit their wide-coverage Watchtower corpus (WTC), in contrast to the typically used Europarl data. Europarl covers 21 languages of the EU with 400k-2M sentence pairs, while WTC spans 300+ widely diverse languages with only 10-100k pairs, in effect sacrificing depth for breadth, and introducing a more radical domain shift. However, as our results show little projected data turns out to be the most beneficial, reinforcing breadth for depth.

While Agić et al. (2016) selected 20k projected sentences at random to train taggers, we propose a novel alternative: selection by *coverage*. We rank the target sentences by percentage of words covered by word alignment from 21 sources of Agić et al. (2016), and select the top k covered instances for training. In specific, we employ the mean coverage ranking of target sentences, whereby each target sentence is coupled with the arithmetic mean of the 21 individual word alignment coverages for each of the 21 source-language sentences. We show that this simple approach to instance selection offers substantial improvements: across all languages, we learn better taggers with significantly fewer training instances.

Dictionaries. Dictionaries are a useful source for distant supervision (Li et al., 2012; Täckström et al., 2013). There are several ways to exploit such information: i) as type constraints during encoding (Täckström et al., 2013), ii) to guide unsupervised learning (Li et al., 2012), or iii) as additional signal at training. We focus on the latter and evaluate two ways to integrate lexical knowledge into neural models, while comparing to the former two: a) by representing lexicon properties as n-hot vector (e.g., if a word has two properties according to lexicon src, it results in a 2-hot vector, if the word is not present in src, a zero vector), with m the number of lexicon properties; b) by *embedding* the lexical features, i.e., $\vec{e}_{src}$ is a lexicon src embedded into an l-dimensional space. We represent $\vec{e}_{src}$ as concatenation of all embedded m properties of length l, and a zero vector otherwise. Tuning on the dev set, we found the second embedding approach to perform best, and simple concatenation outperformed mean vector representations.

We evaluate two dictionary sources, motivated by ease of accessibility to many languages: WIKTIONARY, a word type dictionary that maps tokens to one of the 12 Universal POS tags (Li et al., 2012; Petrov et al., 2012); and UNIMORPH, a morphological dictionary that provides inflectional paradigms across 350 languages (Kirov et al., 2016). For Wiktionary, we use the freely available dictionaries from Li et al. (2012) and Agić et al. (2017). The size of the dictionaries ranges from a few thousands (e.g., Hindi and Bulgarian) to 2M (Finnish UniMorph). Sizes are provided in Table 1, first columns. UniMorph covers between 8-38 morphological properties (for English and Finnish, respectively).

Word embeddings. Embeddings are available for many languages. Pre-initialization of $\vec{w}$ offers consistent and considerable performance improvements in our distant supervision setup (Section 4). We use off-the-shelf Polyglot embeddings (Al-Rfou et al., 2013), which performed consistently better than FastText (Bojanowski et al., 2016).

3 Experiments

Baselines. We compare to the following weakly-supervised POS taggers:

- **AGIC:** Multi-source annotation projection with Bible parallel data by Agić et al. (2015).
- **DAS:** The label propagation approach by Das and Petrov (2011) over Europarl data.

---

[1] CRF decoding did not consistently improve POS accuracy, as recently also independently found (Yang et al., 2018).

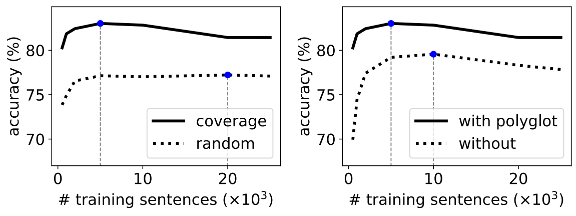

(a) sentence selection (b) pre-trained embeddings

Figure 2: Learning curves for: a) random vs. coverage-based sentence selection in annotation projection, both with Polyglot embeddings, and b) pre-trained embeddings on top of coverage-based selection. Means over 21 languages.

- **Garrette:** The approach by Garrette and Baldridge (2013) that works with projections, dictionaries, and unlabeled target text.
- **Li:** Wiktionary supervision (Li et al., 2012).

Data. Our set of 25 languages is motivated by accessibility to embeddings and dictionaries. In all experiments we work with the 12 Universal POS tags (Petrov et al., 2012). For development, we use 21 dev sets of the Universal Dependencies 2.1 (Nivre et al., 2017). We employ UD test sets on additional languages as well as the test sets of Agić et al. (2015) to facilitate comparisons. Their test sets are a mixture of CoNLL (Buchholz and Marsi, 2006; Nivre et al., 2007) and HamleDT test data (Zeman et al., 2014), and are more distant from the training and development data.

Model and parameters. We extend an off-the-shelf state-of-the-art bi-LSTM tagger with lexicon information. The code is available at: `https://github.com/bplank/bilstm-aux`. The parameter l=40 was set on dev data across all languages. Besides using 10 epochs, word dropout rate (p=.25) and 40-dimensional lexicon embeddings, we use the parameters from Plank et al. (2016). For all experiments, we average over 3 randomly seeded runs, and provide mean accuracy. For the learning curve, we average over 5 random samples with 3 runs each.

4 Results

Table 1 shows the tagging accuracy for individual languages, while the means over all languages are given in Figure 2. There are several take-aways.

Data selection. The first take-away is that coverage-based instance selection yields substantially better training data. Most prior work on annotation projection resorts to arbitrary selection; informed selection clearly helps in this noisy data setup, as shown in Figure 2 (a). Training on 5k instances results in a sweet spot; more data (10k) starts to decrease performance, at a cost of runtime. Training on all WTC data (around 120k) is worse for most languages. From now on we consider the 5k model trained with Polyglot as our baseline (Table 1, column "5k"), obtaining a mean accuracy of 83.0 over 21 languages.

Embeddings initialization. Polyglot initialization offers a large boost; on average +3.8% absolute improvement in accuracy for our 5k training scheme, as shown in Figure 2 (b). The big gap in low-resource setups further shows their effectiveness, with up to 10% absolute increase in accuracy when training on only 500 instances.

Lexical information. The main take-away is that lexical information helps neural tagging, and *embedding* it proves the most helpful. Embedding Wiktionary tags reaches 83.7 accuracy on average, versus 83.4 for n-hot encoding, and 83.2 for type constraints. Only on 4 out of 21 languages are type constraints better. This is the case for only one language for n-hot encoding (French). The best approach is to embed both Wiktionary and Unimorph, boosting performance further to 84.0, and resulting in our final model. It helps the most on morphological rich languages such as Uralic.

On the test sets (Table 4, right) DsDs reaches 87.2 over 8 test languages intersecting Li et al. (2012) and Agić et al. (2016). It reaches 86.2 over the more commonly used 8 languages of Das and Petrov (2011), compared to their 83.4. This shows that our novel "soft" inclusion of noisy dictionaries is superior to a hard decoding restriction, and including lexicons in neural taggers helps. We did not assume any gold data to further enrich the lexicons, nor fix possible tagset divergences.

5 Discussion

Analysis. The inclusion of lexicons results in higher coverage and is part of the explanation for the improvement of DsDs; see correlation in Figure 3 (a). What is more interesting is that our model benefits from the lexicon beyond its content: OOV accuracy for words *not* present in the lexicon overall improves, besides the expected improvement on known OOV, see Figure 3 (b).

	Lex (10^3)		Dev sets (UD2.1)						Test sets				
Language	W	U	5k	TC$_W$	n-hot$_W$	$\vec{e}_W$	DsDs		Das	Li	Garrette	Agic	DsDs
Bulgarian (bg)	3	47	88.6	88.6	88.9	89.6	**89.7**		–	–	83.1	77.7	**83.9**
Croatian (hr)	20	–	84.9	**85.4**	84.9	84.8	†84.8		–	–	–	67.1	†**78.0**
Czech (cs)	14	72	86.6	86.6	86.9	**87.6**	87.2		–	–	–	73.3	**86.8**
Danish (da)	22	24	89.6	89.0	89.8	**90.2**	90.0		83.2	83.3	78.8	79.0	**84.5**
Dutch (nl)	52	26	88.3	88.9	89.0	89.7	**89.8**		79.5	**86.3**	–	–	83.9
English (en)	358	91	86.5	**87.4**	86.8	87.3	87.3		–	**87.1**	80.8	73.0	85.7
Finnish (fi)	104	2,345	81.5	81.2	81.8	82.4	**82.4**		–	–	–	–	–
French (fr)	17	274	91.0	89.6	**91.7**	91.2	91.4		–	–	85.5	76.6	**88.7**
German (de)	62	71	85.0	86.4	85.5	86.0	**86.7**		82.8	85.8	**87.1**	80.2	84.1
Greek (el)	21	–	80.6	**85.7**	80.2	80.5	†80.5		**82.5**	79.2	64.4	52.3	†81.1
Hebrew (he)	3	12	76.0	**76.1**	75.5	74.9	75.3		–	–	–	–	–
Hindi (hi)	2	26	64.6	64.6	64.8	65.4	**66.2**		–	–	–	**67.6**	63.1
Hungarian (hu)	13	13	75.6	75.6	75.3	75.7	**77.9**		–	–	**77.9**	72.0	77.3
Italian (it)	478	410	91.9	91.7	93.4	93.5	**93.7**		86.8	86.5	83.5	76.9	**92.1**
Norwegian (no)	47	18	90.9	90.9	90.9	91.0	**91.5**		–	–	84.3	76.7	**86.2**
Persian (fa)	4	26	42.8	43.0	43.7	43.5	**43.8**		–	–	–	**59.6**	43.6
Polish (pl)	6	132	84.7	84.6	84.2	84.8	**86.0**		–	–	–	75.1	**84.4**
Portuguese	41	211	91.4	91.5	92.3	**92.9**	92.2		87.9	84.5	87.3	83.8	**89.4**
Romanian (ro)	7	4	83.9	83.9	84.8	85.3	**86.3**		–	–	–	–	–
Spanish (es)	234	324	90.4	88.6	91.0	91.5	**92.0**		84.2	86.4	88.7	81.4	**91.7**
Swedish (sv)	89	67	88.9	88.9	89.6	**89.9**	**89.9**		80.5	**86.1**	76.1	75.2	83.1
Avg(21)			83.0	83.2	83.4	83.7	**84.0**	Avg(8: Das)	83.4	84.8	80.8	75.5	**86.2**
								Avg(8: Li$\cap$Agic)	–	84.9	80.8	75.2	**87.2**
Germanic (6)			88.2	88.6	88.6	89.0	**89.2**	Germanic (4: Das)	81.5	**85.4**	–	–	83.9
Romance (5)			89.7	89.0	90.6	90.9	**91.1**	Romance (3: Das)	86.3	85.8	86.5	80.7	**91.1**
Slavic (4)			86.2	86.3	86.2	86.7	**86.9**						
Indo-Iranian (2)			53.7	53.8	54.3	54.4	**55.0**						
Uralic (2)			78.5	78.4	78.6	79.0	**80.1**						

Table 1: Results on the development sets and comparison of our best model to prior work. Lex: Size (word types) of dictionaries (W: Wiktionary, U: UniMorph). TC$_W$: type-constraints using Wiktionary; $\vec{e}_W$ (embedded Wiktionary tags), DsDs: our model with $\vec{e}_{W\cup U}$. Results indicated by † use W only. Best result in boldface; in case of equal means, the one with lower std is boldfaced. Averages over language families (with two or more languages in the sample, number of languages in parenthesis).

More languages. All data sources employed in our experiment are very high-coverage. However, for true low-resource languages, we cannot safely assume the availability of all disparate information sources. Table 2 presents results for four additional languages where some supervision sources are missing. We observe that adding lexicon information always helps, even in cases where only 1k entries are available, and embedding it is usually the most beneficial way. For closely-related languages such as Serbian and Croatian, using resources for one aids tagging the other, and modern resources are a better fit. For example, using the Croatian WTC projections to train a model for Serbian is preferable over in-language Serbian Bible data where the OOV rate is much higher.

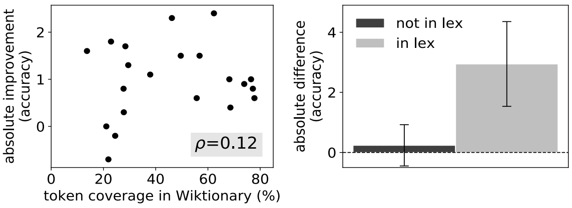

(a) coverage vs. accuracy

(b) OOV accuracy

Figure 3: Analysis of DsDs accuracy improvements over the baseline on all development languages with respect to a) token coverage by the lexicon, including Pearson's ρ; b) OOV accuracy for tokens in/not in the lexicon, with 95% confidence intervals of the mean. Here, a token is covered if we can find it in at least one lexicon.

How much gold data? We assume not having access to *any* gold annotated data. It is thus interesting to ask how much gold data is needed to reach our performance. This is a tricky question, as training within the same corpus naturally favors the same corpus data. We test both in-corpus (UD) and out-of-corpus data (our test sets) and notice an important gap: while in-corpus only 50 sentences are sufficient, outside the corpus one would need over 200 sentences. This experiment was done for a subset of 18 languages with both in- and out-of-corpus test data.

Language	Test	Proj	Emb	Lex (10^3) W	U	TnT	5k	TC$_W$	n-hot$_W$	$\vec{e}_W$	DsDs
Basque (eu)	UD	Bible	eu	1	–	57.5	61.8	61.8	61.4	**62.7**	†**62.7**
Basque (eu)	CoNLL	Bible	eu	1	–	57.0	60.3	60.3	60.3	**61.3**	†**61.3**
Estonian (et)	UD	WTC	et	–	10	79.5	80.6	–	–	–	**81.5**
Serbian (sr)	UD	WTC (hr)	hr	(hr) 20	–	84.0	84.7	**85.5**	85.1	85.2	†85.2
Serbian (sr)	UD	Bible (sr)	hr	(hr) 20	–	77.1	78.9	79.4	80.5	**80.7**	†**80.7**
Tamil (ta)	UD	WTC	ta	–	–	58.2	**61.2**	–	–	–	–

Table 2: Results for languages with missing data sources: WTC projections, Wiktionary (W), or UniMorph (U). Test sets (Test), projection sources (Proj), and embeddings languages (Emb) are indicated. Comparison to TnT (Brants, 2000) trained on Proj. Results indicated by † use W only.

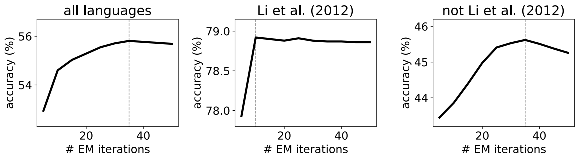

Figure 4: The performance of Li with our dictionary data over EM iterations, separate for the languages from Li et al. (2012) and all the remaining languages in Table 1.

Further comparison. In Table 1 we directly report the accuracies from the original contributions by Das, Li, Garrette, and Agic over the same test data. We additionally attempted to reach the scores of Li by running their tagger over the Table 1 data setup. The results are depicted in Figure 4 as mean accuracies over EM iterations until convergence. We show: i) Li peaks at 10 iterations for their test languages, and at 35 iterations for all the rest. This is in slight contrast to 50 iterations that Li et al. (2012) recommend, although selecting 50 does not dramatically hurt the scores; ii) Our replication falls $\sim$5 points short of their 84.9 accuracy. There is a large 33-point accuracy gap between the scores of Li et al. (2012), where the dictionaries are large, and the other languages in Figure 4, with smaller dictionaries.

Compared to Das, our tagger clearly benefits from pre-trained word embeddings, while theirs relies on label propagation through Europarl, a much cleaner corpus that lacks the coverage of the noisier WTC. Similar applies to Täckström et al. (2013), as they use 1-5M near-perfect parallel sentences. Even if we use much smaller and noisier data sources, DsDs is almost on par: 86.2 vs. 87.3 for the 8 languages from Das and Petrov (2011), and we even outperform theirs on four languages: Czech, French, Italian, and Spanish.

6 Related Work

Most successful work on low-resource POS tagging is based on projection (Yarowsky et al., 2001), tag dictionaries (Li et al., 2012), annotation of seed training data (Garrette and Baldridge, 2013) or even more recently some combination of these, e.g., via multi-task learning (Fang and Cohn, 2016; Kann et al., 2018). Our paper contributes to this literature by leveraging a range of prior directions in a unified, neural test bed.

Most prior work on neural sequence prediction follows the commonly perceived wisdom that hand-crafted features are unnecessary for deep learning methods. They rely on end-to-end training without resorting to additional linguistic resources. Our study shows that this is not the case. Only few prior studies investigate such sources, e.g., for MT (Sennrich and Haddow, 2016; Chen et al., 2017; Li et al., 2017; Passban et al., 2018) and Sagot and Martínez Alonso (2017) for POS tagging use lexicons, but only as n-hot features and without examining the cross-lingual aspect.

7 Conclusions

We show that our approach of distant supervision from disparate sources (DsDs) is simple yet surprisingly effective for low-resource POS tagging. Only 5k instances of projected data paired with off-the-shelf embeddings and lexical information integrated into a neural tagger are sufficient to reach a new state of the art, and both data selection and embeddings are essential components to boost neural tagging performance.

References

Željko Agić, Dirk Hovy, and Anders Søgaard. 2015. If all you have is a bit of the bible: Learning pos taggers for truly low-resource languages. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 268–272, Beijing, China. Association for Computational Linguistics.

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Alonso Martínez, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Željko Agić, Barbara Plank, and Anders Søgaard. 2017. Cross-lingual tagger evaluation without test data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 248–253. Association for Computational Linguistics.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. *arXiv preprint arXiv:1307.1662*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Thorsten Brants. 2000. Tnt - a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing (ANLP-2000)*, Seattle, WA.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City. Association for Computational Linguistics.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945. Association for Computational Linguistics.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, Oregon, USA. Association for Computational Linguistics.

Meng Fang and Trevor Cohn. 2016. Learning when to trust distant supervision: An application to low-resource pos tagging using cross-lingual projection. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 178–186, Berlin, Germany. Association for Computational Linguistics.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147, Atlanta, Georgia. Association for Computational Linguistics.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of pos-taggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 583–592, Sofia, Bulgaria. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

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

Katharina Kann, Johannes Bjerva, Isabelle Augenstein, Barbara Plank, and Anders Sgaard. 2018. Character-level supervision for low-resource pos tagging. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 1–11, Melbourne. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv:1603.04351*.

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large scale parsing and normalization of wiktionary morphological paradigms. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697, Vancouver, Canada. Association for Computational Linguistics.

Shen Li, João Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of the 2012 Joint Conference on Empirical Methods* *in Natural Language Processing and Computational Natural Language Learning*, pages 1389–1398, Jeju Island, Korea. Association for Computational Linguistics.

Joakim Nivre, Željko Agić, and Lars Ahrenberg et al. 2017. Universal dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic. Association for Computational Linguistics.

Peyman Passban, Qun Liu, and Andy Way. 2018. Improving character-based decoding using target-side morphological information for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 58–68. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418. Association for Computational Linguistics.

Benoît Sagot and Héctor Martínez Alonso. 2017. Improving neural tagging with lexical information. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 25–31, Pisa, Italy. Association for Computational Linguistics.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*, pages 1–8. Association for Computational Linguistics.

Daniel Zeman, Ondřej Dušek, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtskỳ, and Jan Hajič. 2014. Hamledt: Harmonized multi-language dependency treebank. *Language Resources and Evaluation*, 48(4):601–637.

Unsupervised Bilingual Lexicon Induction via Latent Variable Models

Zi-Yi Dou **Zhi-Hao Zhou** **Shujian Huang**
State Key Laboratory for Novel Software Technology, Nanjing University, China
{douzy@smail, zhouzh@nlp, huangsj@nlp}.nju.edu.cn

Abstract

Bilingual lexicon extraction has been studied for decades and most previous methods have relied on parallel corpora or bilingual dictionaries. Recent studies have shown that it is possible to build a bilingual dictionary by aligning monolingual word embedding spaces in an unsupervised way. With the recent advances in generative models, we propose a novel approach which builds cross-lingual dictionaries via latent variable models and adversarial training with no parallel corpora. To demonstrate the effectiveness of our approach, we evaluate our approach on several language pairs and the experimental results show that our model could achieve competitive and even superior performance compared with several state-of-the-art models.

1 Introduction

Learning the representations of languages is a fundamental problem in natural language processing and most existing methods exploit the hypothesis that words occurring in similar contexts tend to have similar meanings (Pennington et al., 2014; Bojanowski et al., 2017), which could lead word vectors to capture semantic information. Mikolov et al. (2013) first point out that word embeddings learned on separate monolingual corpora exhibit similar structures. Based on this finding, they suggest it is possible to learn a linear mapping from a source to a target embedding space and then generate bilingual dictionaries. This simple yet effective approach has led researchers to investigate on improving cross-lingual word embeddings with the help of bilingual word lexicons (Faruqui and Dyer, 2014; Xing et al., 2015).

For low-resource languages and domains, cross-lingual signal would be hard and expensive to obtain, and thus it is necessary to reduce the need for bilingual supervision. Artetxe et al. (2017) successfully learn bilingual word embeddings with only a parallel vocabulary of aligned digits. Zhang et al. (2017) utilize adversarial training to obtain cross-lingual word embeddings without any parallel data. However, their performance is still significantly worse than supervised methods. By combining the merits of several previous works, Conneau et al. (2018) introduce a model that reaches and even outperforms supervised state-of-the-art methods with no parallel data.

In recent years, generative models have become more and more powerful. Both Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) and Variational Autoencoders (VAEs) (Kingma and Welling, 2014) are prominent ones. In this work, we borrow the ideas from both GANs and VAEs to tackle the problem of bilingual lexicon induction. The basic idea is to learn latent variables that could capture semantic meaning of words, which would be helpful for bilingual lexicon induction. We also utilize adversarial training for our model and require no form of supervision. We evaluate our approach on several language pairs and experimental results demonstrate that our model could achieve promising performance. We further combine our model with several helpful techniques and show our model could perform competitively and even superiorly compared with several state-of-the-art methods.

2 Related Work

2.1 Bilingual Lexicon Induction

Extracting bilingual lexica has been studied by researchers for a long time. Mikolov et al. (2013) first observe there is isomorphic structure among word embeddings trained separately on monolingual corpora and they learn the linear transformation between languages. Zhang et al. (2016b) improve the method by constraining the transformation matrix to be orthogonal. Xing et al. (2015) incorporate length normalization during training and

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 621–626
Brussels, Belgium, October 31 - November 4, 2018.

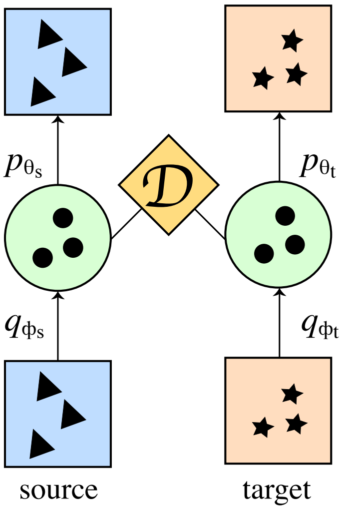

Figure 1: Illustration of our model. ϕ_s and ϕ_t map the source and target word embeddings into latent variables. Discriminator D guides the two latent distributions to be the same.

maximize the cosine similarity instead. They point out that adding an orthogonality constraint can improve performance and has a closed-form solution, which was referred to as Procrustes approach in Smith et al. (2017). Canonical correlation analysis has also been used to map both languages to a shared vector space (Faruqui and Dyer, 2014; Lu et al., 2015).

To reduce the need for supervision signals, Artetxe et al. (2017) use identical digits and numbers to form an initial seed dictionary and then iteratively refine their results until convergence. Zhang et al. (2017) apply adversarial training to align monolingual word vector spaces with no supervision. Conneau et al. (2018) improve the model by combining adversarial training and Procrustes approach, and their unsupervised approach could reach and even outperform state-of-the-art supervised approaches. In this work, we make further improvements and enhance the model proposed in (Conneau et al., 2018) with latent variable model and iterative training procedure.

2.2 Generative Models

VAEs (Kingma and Welling, 2014) represent one of the most successful deep generative models. Standard VAEs assume observed variables are generated from latent variables and the latent variables are sampled from a simple Gaussian distribution. Typically, VAEs utilize an neural inference model to approximate the intractable posterior, and optimize model parameters jointly with a reparameterized variational lower bound. VAEs have been successfully applied in several natural language processing tasks before (Zhang et al., 2016a; Bowman et al., 2016).

GANs (Goodfellow et al., 2014) are another framework for estimating generative models via an adversarial process and have attracted huge attention. The basic strategy is to train a generative model and a discriminative model simultaneously via an adversarial process. Adversarial training technique for matching distribution has proven to be powerful in a variety of tasks (Bowman et al., 2016). Adversarial Autoencoder (Makhzani et al., 2015) is a probabilistic autoencoder that uses the GANs to perform variational inference. By combining a VAE with a GAN, Larsen et al. (2016) use learned feature representations in the GAN discriminator as the basis for the VAE reconstruction objective. GANs have been applied in machine translation before (Yang et al., 2018; Lample et al., 2018).

3 Proposed Approach

In this section, we first briefly introduce VAEs, and then we illustrate the details and training techniques of our proposed model.

3.1 Variational Autoencoder

Variational Autoencoders (VAEs) are deep generative model which are capable of learning complex density models for data via latent variables. Given a nonlinear generative model $p_\theta(x|z)$ with input $x \in \mathbb{R}^D$ and associated latent variable $z \in \mathbb{R}^L$ drawn from a prior distribution $p_0(z)$, the goal of VAEs is to use a recognition model, $q_\phi(z|x)$ to approximate the posterior distribution of the latent variables by maximizing the following variational lower bound

$$\mathcal{L}_{\theta,\phi} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x)||p_0(z)), \tag{1}$$

where KL refers to Kullback-Leibler divergence.

3.2 Our Model

Basically, our model assumes that the source word embedding $\{x_n\}$ and the target word embedding $\{y_n\}$ could be drawn from a same latent variable space $\{z_n\}$, where $\{z_n\}$ is capable of capturing semantic meaning of words.

In contrast to the standard VAE prior that assumes each latent embedding z_n to be drawn from the same latent Gaussian, our model just requires the distribution of latent variables for source and target word embeddings to be equal. To achieve such a goal, we utilize adversarial training to guide the two latent distributions to match with each other.

As in adversarial training, we have networks ϕ_s and ϕ_t for both source and target space, striving to map words into the same latent space, while the discriminator D is a binary classifier which tries to distinguish between the two languages. We also have reconstruction networks θ_s and θ_t as in VAEs.

The objective function for the discriminator D could be formulated as

$$\begin{aligned}\mathcal{L}_D = {} & \mathbb{E}_{z_y \sim q_{\phi_t}(z|y)}[\log D(z_y)] \\ & + \mathbb{E}_{z_x \sim q_{\phi_s}(z|x)}[\log(1 - D(z_x))].\end{aligned} \tag{2}$$

For the source side, the objective is to minimize

$$\begin{aligned}\mathcal{L}_{\phi_s,\theta_s} = {} & \mathbb{E}_{z_x \sim q_{\phi_s}(z|x)}[\log p_{\theta_s}(x|z_x)] \\ & - \mathbb{E}_{z_x \sim q_{\phi_s}(z|x)}[\log D(z_x)].\end{aligned} \tag{3}$$

Here we define $q_{\phi_s}(z|x) = \mathcal{N}(\mu_s(x), \Sigma_s(x))$, where $\mu_s(x) = W_{\mu_s} x$ and $\Sigma_s(x) = \exp(W_{\sigma_s} x)$; W_{μ_s} and W_{σ_s} are learned parameters. We also define the mean of $p_{\theta_s}(x|z)$ to be $W_{\mu_s}^{\mathrm{T}} z$. The objective function and structure for ϕ_t are similar.

The basic framework of our model is shown in Figure 1. As we could see from the figure, our model tries to map the source and target word embedding into the same latent space which could capture the semantic meaning of words.

Theoretical analysis has revealed that adversarial training tries to minimize the Jensen-Shannon (JS) divergence between the real and fake distribution. Therefore, one can view our model as replace KL divergence in Equation 1 with JS divergence and change the Gaussian prior to the target distribution.

3.3 Training Strategy

Our model has two generators ϕ_s and ϕ_t, and we have found that training them jointly would be extremely unstable. In this paper, we propose an iterative method to train our models. Basically, we first initialize W_{μ_t} to be identity matrix and train ϕ_s and θ_s on the source side. After convergence, we freeze W_{μ_s}, and then train ϕ_t and θ_t in the target side. The pseudo-code for this process is shown in Algorithm 1. It should be noted that there is no variance once completing training.

Algorithm 1 Training Strategy

```
$W_{\mu_t} = I$
for $i = 1, \cdots, n_{iter}$ do
    while $\phi_s$ and $\theta_s$ have not converged do
        Update discriminator $D$
        Update $\phi_s$ and $\theta_s$
    end while
    while $\phi_t$ and $\theta_t$ have not converged do
        Update discriminator $D$
        Update $\phi_t$ and $\theta_t$
    end while
end for
```

4 Experiment

Our experiments could be divided into two parts. In the first part, we conduct experiments on small-scale datasets and our main baseline is Zhang et al. (2017). In the second part, we combine our model with several advanced techniques and we compare our model with Conneau et al. (2018) on large-scale datasets.

		#tokens	vocab. size
zh-en	zh	21m	3,349
	en	53m	5,154
es-en	es	61m	4,774
	en	95m	6,637
it-en	it	73m	8,490
	en	93m	6,597

Table 1: Statistics of the non-parallel corpora. Language codes: zh = Chinese, en = English, es = Spanish, it = Italian.

4.1 Small-scale Datasets

In this section, our experiments focus on small-scale datasets and our main baseline model is adversarial autoencoder (Zhang et al., 2017). For justice, we use the same model selection strategy with Zhang et al. (2017), i.e. we choose the model whose sum of reconstruction loss and classification accuracy is the least. The source and target word embeddings would be first mapped into the latent space. For each source word embedding x, it would be first transformed into z_x. The the its k nearest target embeddings would be retrieved and be compared against the entry in a ground truth bilingual lexicon. Performance is measured by top-1 accuracy.

4.1.1 Experiments on Chinese-English Dataset

For this set of experiments, we use the same data as Zhang et al. (2017). The statistics of the final training data is given in Table 1. We use Chinese-English Translation Lexicon Version 3.0 (LDC2002L27) as our ground truth bilingual lexicon for evaluation.

The baseline models are MonoGiza system (Dou et al., 2015), translation matrix (TM) (Mikolov et al., 2013), isometric alignment (IA) (Zhang et al., 2016b) and adversarial training approach (Zhang et al., 2017).

Table 2 summarizes the performance of baseline models and our approach. The results of baseline models are cited from Zhang et al. (2017). As we can see from the table, our model could achieve superior performance compared with other baseline models. Table 3 lists some word translation examples given by our model.

Model	#seeds	Accuracy (%)
MonoGiza w/o emb.	0	0.05
MonoGiza w/ emb.	0	0.09
TM	50	0.29
IA	100	21.79
Zhang et al. (2017)	0	43.31
Ours	0	**51.37**

Table 2: Experimental results on Chinese-English dataset. The results of baseline models are cited from Zhang et al. (2017).

航空	铁路	时代	学校
airline	rail	antiquity	**school**
aviation	**railway**	**era**	education
airliner	**railroad**	century	college
service	freight	medieval	student
flight	metro	historian	teacher

Table 3: Word translation examples for Chinese-English dataset. Ground truths are marked in bold.

4.1.2 Experiments on Other Language Pairs Datasets

We also conduct experiments on Spanish-English and Italian-English language pairs. Again, we use the same dataset with Zhang et al. (2017). and the statistics are shown in Table 1. The ground truth bilingual lexica for Spanish-English and Italian-English are obtained from Multilingual Unsupervised and Supervised Embeddings (MUSE) [1].

	Model	Accuracy (%)
es-en	Zhang et al. (2017)	69.22
	Ours	**75.21**
it-en	Zhang et al. (2017)	55.31
	Ours	**61.08**

Table 4: Experimental results on Spanish-English and Italian-English datasets.

The experimental results are shown in Table 4. Because Spanish, Italian and English are closely related languages, the accuracy would be higher than the Chinese-English dataset. Our model is able to outperform baseline model in this setting.

Model	Accuracy (%)		
	en-es	en-ru	en-zh
Methods without refinement			
Adv-NN	69.8	29.1	18.5
Adv-CSLS	75.7	37.2	23.4
Ours-NN	71.8	32.8	22.9
Ours-CSLS	**76.6**	**39.3**	**26.0**
Methods with refinement			
Adv-Refine-NN	79.1	37.3	30.9
Adv-Refine-CSLS	81.7	44.0	32.5
Ours-Refine-NN	79.1	42.7	32.5
Ours-Refine-CSLS	**82.1**	**48.7**	**33.3**

Table 5: Experimental results on large-scale datasets. Language codes: en=English, es = Spanish, ru = Russian, zh = Chinese.

4.2 Large-scale Datasets

In this section, we integrate our method with Conneau et al. (2018), whose method improves Zhang et al. (2017) by more sophiscated refinement procedure and validation criterion. We replace their first step, namely the adversarial training step, with our model. Basically, we first map the source and target embeddings into the latent space using our algorithm, and then fine-tune the identity mapping in the latent space with the closed-form Procrustes solution. We use their similarity measure, namely cross-domain similarity local scaling (CSLS), to produce reliable matching pairs and validation criterion for unsupervised model selection.

---

[1] https://github.com/facebookresearch/MUSE

We conduct experiments on English-Spanish, English-Russian and English-Chinese datasets, which are the same as Conneau et al. (2018). The results are shown in Table 5. As seen, our model could consistently achieve better performance compared with adversarial training. After refinement, our model could further achieve competitive and even superior results compared with state-of-the-art unsupervised methods. This further demonstrates the capacity of our model.

5 Conclusion

Based on the assumption that word vectors in different languages could be drawn from a same latent variable space, we propose a novel approach which builds cross-lingual dictionaries via latent variable models and adversarial training with no parallel corpora. Experimental results on several language pairs have demonstrated the effectiveness and universality of our model. We hope our method could be beneficial to other areas such as unsupervised machine translation (Lample et al., 2018).

Future directions include validate our model on more realistic scenarios (Dinu et al., 2015) as well as combine our algorithms with more sophiscated adversarial networks (Arjovsky et al., 2017; Gulrajani et al., 2017).

Acknowledgments

We thank all the anonymous reviewers for their insightful comments.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations (ICLR)*.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. *International Conference on Learning Representations (ICLR), Workshop Track*.

Qing Dou, Ashish Vaswani, Kevin Knight, and Chris Dyer. 2015. Unifying bayesian inference and vector space models for improved decipherment. In *International Joint Conference on Natural Language Processing (IJCNLP)*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Conference on Neural Information Processing Systems (NIPS)*.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. *International Conference on Learning Representations (ICLR)*.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. 2016. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning (ICML)*.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *International Conference on Learning Representations (ICLR)*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Improving neural machine translation with conditional sequence generative adversarial nets. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Biao Zhang, Deyi Xiong, Hong Duan, Min Zhang, et al. 2016a. Variational neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016b. Ten pairs to tag–multilingual pos tagging via coarse mapping between embeddings. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Learning Unsupervised Word Translations Without Adversaries

Tanmoy Mukherjee[1] and **Makoto Yamada**[2] and **Timothy Hospedales**[1]
[1] The University of Edinburgh [2] Kyoto University/ RIKEN AIP/ JST PRESTO
`{t.mukherjee@sms.,t.hospedales@}ed.ac.uk, myamada@i.kyoto-u.ac.jp`

Abstract

Word translation, or bilingual dictionary induction, is an important capability that impacts many multilingual language processing tasks. Recent research has shown that word translation can be achieved in an unsupervised manner, without parallel seed dictionaries or aligned corpora. However, state of the art methods for unsupervised bilingual dictionary induction are based on generative adversarial models, and as such suffer from their well known problems of instability and hyper-parameter sensitivity. We present a statistical dependency-based approach to bilingual dictionary induction that is unsupervised – no seed dictionary or parallel corpora required; and introduces no adversary – therefore being much easier to train. Our method performs comparably to adversarial alternatives and outperforms prior non-adversarial methods.

1 Introduction

Translating words between languages, or more generally inferring bilingual dictionaries, is a long-studied research direction with applications including machine translation (Lample et al., 2017), multilingual word embeddings (Klementiev et al., 2012), and knowledge transfer to low resource languages (Guo et al., 2016). Research here has a long history under the guise of decipherment (Knight et al., 2006). Current contemporary methods have achieve effective word translation through theme-aligned corpora (Gouws et al., 2015), or seed dictionaries (Mikolov et al., 2013).

Mikolov et al. (2013) showed that monolingual word embeddings exhibit isomorphism across languages, and can be aligned with a simple linear transformation. Given two sets word vectors learned independently from monolingual corpora, and a dictionary of seed pairs to learn a linear transformation for alignment; they were able to estimate a complete bilingual lexicon. Many studies have since followed this approach, proposing various improvements such as orthogonal mappings (Artetxe et al., 2016) and improved objectives (Lazaridou et al., 2015).

Obtaining aligned corpora or bilingual seed dictionaries is nevertheless not straightforward for all language pairs. This has motivated a wave of very recent research into *unsupervised* word translation: inducing bilingual dictionaries given only monolingual word embeddings (Conneau et al., 2018; Zhang et al., 2017b,a; Artetxe et al., 2017). The most successful have leveraged ideas from Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). In this approach the generator provides the cross-modal mapping, taking embeddings of dictionary words in one language and 'generating' their translation in another. The discriminator tries to distinguish between this 'fake' set of translations and the true dictionary of embeddings in the target language. The two play a competitive game, and if the generator learns to fool the discriminator, then its cross-modal mapping should be capable of inducing a complete dictionary, as per Mikolov et al. (2013).

Despite these successes, such adversarial methods have a number of well-known drawbacks (Arjovsky et al., 2017): Due to the nature of their min-max game, adversarial training is very unstable, and they are prone to divergence. It is extremely hyper-parameter sensitive, requiring problem-specific tuning. Convergence is also hard to diagnose and does not correspond well to efficacy of the generator in downstream tasks (Hoshen and Wolf, 2018).

In this paper, we propose an alternative statistical dependency-based approach to unsupervised word translation. Specifically, we propose to search for the cross-lingual word pairing that maximizes statistical dependency in terms of squared

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 627–632
Brussels, Belgium, October 31 - November 4, 2018.

loss mutual information (SMI) (Yamada et al., 2015; Suzuki and Sugiyama, 2010). Compared to prior statistical dependency-based approaches such as Kernelized Sorting (KS) (Quadrianto et al., 2009) we advance: (i) through use of SMI rather than their Hilbert Schmidt Independence Criterion (HSIC) and (ii) through jointly optimising cross-modal pairing with representation learning within each view. In contrast to prior work that uses a fixed representation, by non-linearly projecting monolingual world vectors before matching, we learn a new embedding where statistical dependency is easier to establish. Our method: (i) achieves similar unsupervised translation performance to recent adversarial methods, while being significantly easier to train and (ii) clearly outperforms prior non-adversarial methods.

2 Proposed model

2.1 Deep Distribution Matching

Let dataset $\mathcal{D}$ contain two sets of unpaired monolingual word embeddings from two languages $\mathcal{D} = (\{\boldsymbol{x}_i\}_{i=1}^n, \{\boldsymbol{y}_j\}_{j=1}^n)$ where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. Let π be a permutation function over $\{1, 2, \ldots, n\}$, and $\boldsymbol{\Pi}$ the corresponding permutation indicator matrix: $\boldsymbol{\Pi} \in \{0,1\}^{n\times n}, \boldsymbol{\Pi}\boldsymbol{1}_n = \boldsymbol{1}_n$, and $\boldsymbol{\Pi}^\top\boldsymbol{1}_n = \boldsymbol{1}_n$. Where $\boldsymbol{1}_n$ is the n-dimensional vector with all ones. We aim to optimize for both the permutation $\boldsymbol{\Pi}$ (bilingual dictionary), and non-linear transformations $\boldsymbol{g}_{\boldsymbol{x}}(\cdot)$ and $\boldsymbol{g}_{\boldsymbol{y}}(\cdot)$ of the respective wordvectors, that maximize statistical dependency between the views. While regularising by requiring the original word embedding information is preserved through reconstruction using decoders $\boldsymbol{f}_{\boldsymbol{x}}(\cdot)$ and $\boldsymbol{f}_{\boldsymbol{y}}(\cdot)$. Our overall loss function is:

$$\begin{aligned}
&\underset{\boldsymbol{\Theta}_x,\boldsymbol{\Theta}_y,\boldsymbol{\Pi}}{min} \underbrace{\Omega(\mathcal{D};\boldsymbol{\Theta}_{\boldsymbol{x}},\boldsymbol{\Theta}_{\boldsymbol{y}})}_{Regularizer} - \underbrace{\lambda D_{\boldsymbol{\Pi}}(\mathcal{D};\boldsymbol{\Theta}_{\boldsymbol{x}},\boldsymbol{\Theta}_{\boldsymbol{y}})}_{Dependency},\\
&D_{\boldsymbol{\Pi}}(\mathcal{D};\boldsymbol{\Theta}_{\boldsymbol{x}},\boldsymbol{\Theta}_{\boldsymbol{y}}) = D_{\boldsymbol{\Pi}}(\{\boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x}_i), \boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{y}_{\pi(i)})\}_{i=1}^n),\\
&\Omega(\mathcal{D};\boldsymbol{\Theta}_{\boldsymbol{x}},\boldsymbol{\Theta}_{\boldsymbol{y}}) = \sum_{i=1}^n \|\boldsymbol{x}_i - \boldsymbol{f}_{\boldsymbol{x}}(\boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x}_i))\|_2^2\\
&\qquad + \|\boldsymbol{y}_i - \boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{y}_i))\|_2^2\\
&\qquad + R(\boldsymbol{\Theta}_{\boldsymbol{x}}) + R(\boldsymbol{\Theta}_{\boldsymbol{y}}).
\end{aligned} \tag{1}$$

where $\boldsymbol{\Theta}$s parameterize the encoding and reconstruction transformations, $R(\cdot)$ is a regularizer (e.g., ℓ_2-norm and ℓ_1-norm), and $D_{\boldsymbol{\Pi}}(\cdot,\cdot)$ is a statistical dependency measure. Crucially compared to prior methods such as matching CCA (Haghighi et al., 2008), dependency measures such as SMI do not need comparable representations to get started, making the bootstrapping problem less severe.

2.2 Dependence Estimation

Squared-Loss Mutual Information (SMI) The squared loss mutual information between two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as (Suzuki and Sugiyama, 2010):

$$\mathrm{SMI} = \iint \left(\frac{p(\boldsymbol{x},\boldsymbol{y})}{p(\boldsymbol{x})p(\boldsymbol{y})} - 1\right)^2 p(\boldsymbol{x})p(\boldsymbol{y})\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{y},$$

which is the Pearson divergence (Pearson, 1900) from $p(\boldsymbol{x},\boldsymbol{y})$ to $p(\boldsymbol{x})p(\boldsymbol{y})$. The SMI is an f-divergence (Ali and Silvey, 1966). That is, it is a non-negative measure and is zero only if the random variables are independent.

To measure SMI from a set of samples we take a direct density ratio estimation approach (Suzuki and Sugiyama, 2010), which leads (Yamada et al., 2015) to the estimator:

$$\widehat{\mathrm{SMI}}(\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^n) = \frac{1}{2n}\mathrm{tr}\left(\mathrm{diag}\left(\widehat{\boldsymbol{\alpha}}\right)\boldsymbol{K}\boldsymbol{L}\right) - \frac{1}{2},$$

where $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ are the gram matricies for $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively, and

$$\begin{aligned}
&\widehat{\boldsymbol{H}} = \frac{1}{n^2}(\boldsymbol{K}\boldsymbol{K}^\top) \circ (\boldsymbol{L}\boldsymbol{L}^\top),\\
&\widehat{\boldsymbol{h}} = \frac{1}{n}(\boldsymbol{K}\circ\boldsymbol{L})\boldsymbol{1}_n,\ \ \widehat{\boldsymbol{\alpha}} = \left(\widehat{\boldsymbol{H}} + \lambda\boldsymbol{I}_n\right)^{-1}\widehat{\boldsymbol{h}},
\end{aligned}$$

$\lambda > 0$ is a regularizer and $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix.

SMI for Matching SMI computes the dependency between two sets of variables, under an assumption of known correspondence. In our application this corresponds to a measure of dependency between two *aligned* sets of monolingual wordvectors. To exploit SMI for matching, we introduce a permutation variable $\boldsymbol{\Pi}$ by replacing $\boldsymbol{L} \to \boldsymbol{\Pi}^\top\boldsymbol{L}\boldsymbol{\Pi}$ in the estimator:

$$\widehat{\mathrm{SMI}}(\{(\boldsymbol{x}_i,\boldsymbol{y}_{\pi(i)})\}_1^n) = \frac{1}{2n}\mathrm{tr}\left(\mathrm{diag}\left(\widehat{\boldsymbol{\alpha}}_{\boldsymbol{\Pi}}\right)\boldsymbol{K}\boldsymbol{\Pi}^\top\boldsymbol{L}\boldsymbol{\Pi}\right) - \frac{1}{2},$$

that will enable optimizing $\boldsymbol{\Pi}$ to maximize SMI.

2.3 Optimization of parameters

To initialize $\boldsymbol{\Theta}_{\boldsymbol{x}}$ and $\boldsymbol{\Theta}_{\boldsymbol{y}}$, we first independently estimate them using autoencoders. Then we employ an alternative optimization on Eq. (1) for

$(\boldsymbol{\Theta}_{\boldsymbol{x}}, \boldsymbol{\Theta}_{\boldsymbol{y}})$ and $\boldsymbol{\Pi}$ until convergence. We use 3 layer MLP neural networks for both $\boldsymbol{f}$ and $\boldsymbol{g}$. Algorithm 1 summarises the steps.

Optimization for $\boldsymbol{\Theta}_{\boldsymbol{x}}$ and $\boldsymbol{\Theta}_{\boldsymbol{y}}$ With fixed permutation matrix $\boldsymbol{\Pi}$ (or π), the objective function

$$\min_{\boldsymbol{\Theta}_{\boldsymbol{x}}, \boldsymbol{\Theta}_{\boldsymbol{y}}} \Omega(\mathcal{D}; \boldsymbol{\Theta}_{\boldsymbol{x}}, \boldsymbol{\Theta}_{\boldsymbol{y}}) - \lambda D_{\boldsymbol{\Pi}}(\mathcal{D}; \boldsymbol{\Theta}_{\boldsymbol{x}}, \boldsymbol{\Theta}_{\boldsymbol{y}}) \quad (2)$$

is an autoencoder optimization with regularizer $D_{\boldsymbol{\Pi}}(\cdot)$, and can be solved with backpropagation.

Optimization for $\boldsymbol{\Pi}$ To find the permutation (word matching) $\boldsymbol{\Pi}$ that maximizes SMI given fixed encoding parameters $\boldsymbol{\Theta}_{\boldsymbol{x}}, \boldsymbol{\Theta}_{\boldsymbol{y}}$, we only need to optimize the dependency term $D_{\boldsymbol{\Pi}}$ in Eq. (1). We employ the LSOM algorithm (Yamada et al., 2015). The estimator of SMI for samples $\{\boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x}_i), \boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{y}_{\pi(i)})\}_{i=1}^{n}$ encoded with $\boldsymbol{g}_x, \boldsymbol{g}_y$ is:

$$\widehat{\mathrm{SMI}} = \frac{1}{2n}\mathrm{tr}\left(\mathrm{diag}\left(\widehat{\boldsymbol{\alpha}}_{\boldsymbol{\Theta},\boldsymbol{\Pi}}\right) \boldsymbol{K}_{\boldsymbol{\Theta}_{\boldsymbol{x}}}\boldsymbol{\Pi}^\top \boldsymbol{L}_{\boldsymbol{\Theta}_{\boldsymbol{y}}}\boldsymbol{\Pi}\right) - \frac{1}{2}.$$

Which leads to the optimization problem:

$$\max_{\boldsymbol{\Pi}\in\{0,1\}^{n\times n}} \mathrm{tr}\left(\mathrm{diag}\left(\widehat{\boldsymbol{\alpha}}_{\boldsymbol{\Theta},\boldsymbol{\Pi}}\right) \boldsymbol{K}_{\boldsymbol{\Theta}_{\boldsymbol{x}}}\boldsymbol{\Pi}^\top \boldsymbol{L}_{\boldsymbol{\Theta}_{\boldsymbol{y}}}\boldsymbol{\Pi}\right)$$
$$\text{s.t.} \quad \boldsymbol{\Pi}\mathbf{1}_n = \mathbf{1}_n, \boldsymbol{\Pi}^\top\mathbf{1}_n = \mathbf{1}_n. \quad (3)$$

Since the optimization problem is NP-hard, we iteratively solve the relaxed problem (Yamada et al., 2015):

$$\boldsymbol{\Pi}^{\mathrm{new}} = (1-\eta)\boldsymbol{\Pi}^{\mathrm{old}} +$$
$$\eta \operatorname*{argmax}_{\boldsymbol{\Pi}} \mathrm{tr}\left(\mathrm{diag}\left(\widehat{\boldsymbol{\alpha}}_{\boldsymbol{\Theta},\boldsymbol{\Pi}^{\mathrm{old}}}\right) \boldsymbol{K}_{\boldsymbol{\Theta}_{\boldsymbol{x}}}\boldsymbol{\Pi}^\top \boldsymbol{L}_{\boldsymbol{\Theta}_{\boldsymbol{y}}}\boldsymbol{\Pi}^{\mathrm{old}}\right),$$

where $0 < \eta \leq 1$ is a step size. The optimization problem is a *linear assignment problem* (LAP). Thus, we can efficiently solve the algorithm by using the *Hungarian method* (Kuhn, 1955). To get discrete $\boldsymbol{\Pi}$, we solve the last step by setting $\eta = 1$.

Intuitively, this can be seen as searching for the permutation $\boldsymbol{\Pi}$ for which the data in the two (initially unsorted views) have a matching within-view affinity (gram) matrix, where matching is defined by maximum SMI.

3 Experiments

In this section, we evaluate the efficacy of our proposed method against various state of the art methods for word translation.

Implementation Details Our autoencoder consists of two layers with dropout and a *tanh* nonlinearity. We use polynomial kernel to compute the gram matrices $\boldsymbol{K}$ and $\boldsymbol{L}$. For all pairs of languages, we fix the number of training epochs to 20. All the word vectors are ℓ_2 unit normalized. For CSLS we set the number of neighbors to 10. For optimizing $\boldsymbol{\Pi}$ at each epoch, we set the step size $\eta = 0.75$ and use 20 iterations. For the regularization $R(\boldsymbol{\Theta})$, we use the sum of the Frobenius norms of weight matrices. We train $\boldsymbol{\Theta}$ using full batch gradient-descent, with learning rate 0.05.

Algorithm 1 SMI-based unsupervised word translation

```
Input: Unpaired word embeddings $\mathcal{D} = (\{\boldsymbol{x}_i\}_{i=1}^{n}, \{\boldsymbol{y}_j\}_{j=1}^{n})$.
Init: weights $\boldsymbol{\Theta}_{\boldsymbol{x}}, \boldsymbol{\Theta}_{\boldsymbol{y}}$, permutation matrix $\boldsymbol{\Pi}$.
while not converged do
    Update $\boldsymbol{\Theta}_{\boldsymbol{x}}, \boldsymbol{\Theta}_{\boldsymbol{y}}$ given $\boldsymbol{\Pi}$: Backprop (2).
    Update $\boldsymbol{\Pi}$ given $\boldsymbol{\Theta}_{\boldsymbol{x}}, \boldsymbol{\Theta}_{\boldsymbol{y}}$: LSOM (3).
end while
Output: Permutation Matrix $\boldsymbol{\Pi}$. Params $\boldsymbol{\Theta}_{\boldsymbol{x}}, \boldsymbol{\Theta}_{\boldsymbol{y}}$.
```

Datasets We performed experiments on the publicly available English-Italian, English-Spanish and English-Chinese datasets released by (Dinu and Baroni, 2015; Zhang et al., 2017b; Vulic and Moens, 2013). We name this collective set of benchmarks BLI. We also conduct further experiments on a much larger recent public benchmark, MUSE (Conneau et al., 2018)[1].

Setting and Metrics We evaluate all methods in terms of Precision@1, following standard practice. We note that while various methods in the literature were initially presented as fully supervised (Mikolov et al., 2013), semi-supervised (using a seed dictionary) (Haghighi et al., 2008), or unsupervised (Zhang et al., 2017b), most of them can be straightforwardly adapted to run in any of these settings. Therefore we evaluate all methods both in the unsupervised setting in which we are primarily interested, and also the commonly evaluated semi-supervised setting with 500 seed pairs.

Competitors: Non-Adversarial In terms of competitors that, like us, do not make use of GANs, we evaluate: **Translation Matrix** (Mikolov et al., 2013), which alternates between estimating a linear transformation by least squares and matching by nearest neighbour (NN). **Multilingual Correlation** (Faruqui and Dyer, 2014), and **Matching CCA** (Haghighi et al., 2008), which alternates between matching and estimat-

---

[1] https://github.com/facebookresearch/MUSE/

	MUSE Dataset						BLI Datasets					
Methods	es-en	en-es	it-en	en-it	zh-en	en-zh	es-en	en-es	it-en	en-it	zh-en	en-zh
TM (Mikolov et al., 2013)	5.6	4.8	5.2	4.8	2.6	1.8	3.2	2.9	4.6	4.2	3.2	2.0
CCA (Faruqui and Dyer, 2014)	6.1	5.6	5.8	5.2	3.1	2.3	5.3	5.0	4.6	4.1	3.2	2.9
MCCA (Haghighi et al., 2008)	5.7	5.1	5.4	4.8	3.0	2.2	2.9	2.5	4.2	4.1	2.8	1.9
KS (Quadrianto et al., 2009)	8.3	7.4	6.3	5.7	4.8	3.2	9.6	8.9	8.2	7.3	3.7	3.5
Self-Training (Artetxe et al., 2017)	12.4	12.2	10.7	10.2	5.8	5.6	15.8	14.5	13.7	12.7	14.8	13.4
EMDOT (Zhang et al., 2017b)	72.4	71.8	72.8	72.6	32.8	31.7	29.3	31.2	25.6	28.4	24.2	27.8
W-GAN (Zhang et al., 2017b)	78.2	77.4	75.3	74.8	38.6	37.5	23.4	26.7	24.0	25.3	21.2	22.8
GAN-NN (Conneau et al., 2018)	69.8	71.3	72.1	71.5	41.3	40.2	21.4	24.3	22.7	23.2	21.3	21.8
Deep-SMI (Ours)	75.9	80.6	75.7	75.2	38.5	38.1	27.3	28.2	25.7	26.4	22.5	22.3
Deep-SMI-CSLS	79.2	84.5	78.8	78.5	43.7	42.8	28.6	29.3	26.7	28.2	23.2	24.7

Table 1: Unsupervised word translation on MUSE and BLI datasets. Precision @ 1 metric. Top group: Conventional methods. Middle group: Adversarial methods. Bottom group: Our methods. Language codes zh=Chinese,en=English,es=Spanish,it=Italian

	MUSE Dataset						BLI Datasets					
Methods	es-en	en-es	it-en	en-it	zh-en	en-zh	es-en	en-es	it-en	en-it	zh-en	en-zh
TM (Mikolov et al., 2013)	32.6	30.1	34.3	33.6	32.4	31.2	28.2	32.1	29.2	32.1	28.5	27.4
CCA (Faruqui and Dyer, 2014)	27.3	27.1	25.4	24.2	23.1	20.2	25.8	28.3	24.3	25.1	19.2	22.8
MCCA (Haghighi et al., 2008)	26.3	25.8	22.7	21.3	24.5	23.8	24.2	26.1	17.6	19.2	18.4	21.6
KS (Quadrianto et al., 2009)	34.5	32.6	35.2	33.8	34.3	33.2	27.5	29.1	34.3	32.1	20.0	23.2
Self-Training (Artetxe et al., 2017)	35.8	31.4	36.0	34.6	34.3	33.0	27.8	29.8	39.7	33.8	23.6	21.4
EMDOT (Zhang et al., 2017b)	78.2	76.3	75.0	74.6	33.2	32.0	30.2	28.4	31.7	30.3	29.3	28.7
W-GAN (Zhang et al., 2017b)	81.2	80.5	77.2	75.1	39.0	38.2	28.6	27.9	33.7	29.5	36.7	34.4
GAN-NN (Conneau et al., 2018)	74.8	72.3	74.3	72.5	43.2	42.7	22.8	26.1	27.9	27.1	24.2	23.6
Deep-SMI (Ours)	80.6	75.9	78.2	76.7	45.7	44.6	38.5	37.6	42.3	38.2	29.2	27.4
Deep-SMI-CSLS	84.5	79.2	79.7	78.7	42.3	44.4	28.6	29.3	26.7	28.2	23.2	24.7

Table 2: Semi-supervised word translation on MUSE and BLI using 500 seed pair initial dictionary. Precision @ 1 metric. Top group: Conventional methods. Middle group: Adversarial methods. Bottom group: Our methods.

ing a joint linear subspace. **Kernelized Sorting** (Quadrianto et al., 2009), which directly uses HSIC-based statistical dependency to match heterogeneous data points. **Self Training** (Artetxe et al., 2017) A recent state of the art method that alternate between estimating an orthonormal transformation, and NN matching.

Competitors: Adversarial In terms of competitors that do make use of adversarial training, we compare: **W-GAN** and **EMDOT** (Zhang et al., 2017b) make use of adversarial learning using Wasserstein GAN and Earth Movers Distance respectively. **GAN-NN** (Conneau et al., 2018) uses adversarial learning to train an orthogonal transformation, along with some refinement steps and an improvement to the conventional NN matching procedure called ‘cross-domain similarity local scaling’ (CSLS). Since this is a distinct step, we also evaluate our method with CSLS.

We use the provided code for GAN-NN and Self-Train, while re-implementing EDOT/W-GAN to avoid dependency on theano.

3.1 Results

Fully Unsupervised Table 1 presents comparative results for unsupervised word translation on BLI and MUSE. From these we observe: (i) Our method (bottom) is consistently and significantly better than non-adversarial alternatives (top). (ii) Compared to adversarial alternatives Deep-SMI performs comparably.

All methods generally perform better on the MUSE dataset than BLI. These differences are due to a few factors: MUSE is a significantly

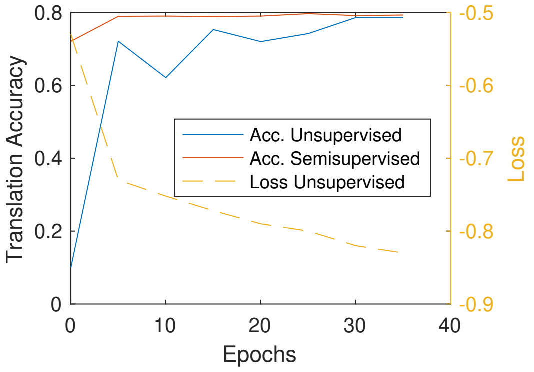

Figure 1: Training process of Deep-SMI

larger dataset than BLI, benefitting methods that can exploit a large amount of training data. In the ground-truth annotation, BLI contains 1-1 translations while MUSE contains more realistic 1-many translations (if any correct translation is picked, a success is counted), making it easier to reach a higher score.

Semi-supervised Results using a 500-word bilingual seed dictionary are presented in Table 2. From these we observe: (i) The conventional methods' performances (top) jump up, showing that they are more competitive if at least some sparse data is available. (ii) Deep-SMI performance also improves, and still outperforms the classic methods significantly overall. (iii) Again, we perform comparably to the GAN methods.

3.2 Discussion

Figure 1 shows the convergence process of Deep-SMI. From this we see that: (i) Unlike the adversarial methods, our objective (Eq. (1)) improves smoothly over time, making convergence much easier to assess. (ii) Unlike the adversarial methods, our accuracy generally mirrors the model's loss. In contrast, the various losses of the adversarial approaches do not well reflect translation accuracy, making model selection or early stopping a challenge in itself. Please compare our Figure 1 with Fig 3 in Zhang et al. (2017b), and Fig 2 in Conneau et al. (2018).

There are two steps in our optimization: matching permutation $\mathbf{\Pi}$ and representation weights $\mathbf{\Theta}$. Although this is an alternating optimization, it is analogous to an EM-type algorithm optimizing latent variables ($\mathbf{\Pi}$) and parameters ($\mathbf{\Theta}$). While local minima are a risk, every optimisation step for either variable reduces our objective Eq. (1). There is no min-max game, so no risk of divergence as in the case of adversarial GAN-type methods.

Our method can also be understood as providing an unsupervised *Deep*-CCA type model for relating heterogeneous data across two views. This is in contrast to the recently proposed unsupervised shallow CCA (Hoshen and Wolf, 2018), and conventional supervised Deep-CCA (Chang et al., 2018) that requires paired data for training; and using SMI rather than correlation as the optimisation objective.

4 Conclusion

We have presented an effective approach to unsupervised word translation that performs comparably to adversarial approaches while being significantly easier to train and diagnose; as well as outperforming prior non-adversarial approaches.

References

Syed M. Ali and Samuel. D. Silvey. 1966. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)*, 28(1):131–142.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *ICML*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *EMNLP*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *ACL*.

Xiaobin Chang, Tao Xiang, and Timothy M. Hospedales. 2018. Scalable and effective deep CCA via soft decorrelation. In *CVPR*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *ICLR*.

Georgiana Dinu and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. *ICLR Workshops*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *EACL*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *ICML*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A distributed representation-based framework for cross-lingual transfer parsing. *JAIR*, 55(1):995–1023.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*.

Yedid Hoshen and Lior Wolf. 2018. Unsupervised correlation analysis. In *CVPR*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *COLING*.

K. Knight, A. Nair, N. Rathod, and K. Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proc. ACL-COLING*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *ACL*.

Tomas Mikolov, Google Inc, Mountain View, Quoc V. Le, Google Inc, Ilya Sutskever, and Google Inc. 2013. Exploiting similarities among languages for machine translation.

Karl Pearson. 1900. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175.

Novi Quadrianto, Le Song, and Alex J Smola. 2009. Kernelized sorting. In *NIPS*.

Taiji Suzuki and Masashi Sugiyama. 2010. Sufficient dimension reduction via squared-loss mutual information estimation. In *AISTATS*.

Ivan Vulic and Marie-Francine Moens. 2013. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *HLT-NAACL*.

Makoto Yamada, Leonid Sigal, Michalis Raptis, Machiko Toyoda, Yi Chang, and Masashi Sugiyama. 2015. Cross-domain matching with squared-loss mutual information. *IEEE TPAMI*, 37(9):1764–1776.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017a. Adversarial training for unsupervised bilingual lexicon induction. In *ACL*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *EMNLP*.

Adversarial Training for Multi-task and Multi-lingual Joint Modeling of Utterance Intent Classification

Ryo Masumura and **Yusuke Shinohara** and **Ryuichiro Higashinaka** and **Yushi Aono**
NTT Media Intelligence Laboratories, NTT Corporation
1-1, Hikarinooka, Yokosuka-shi, Kanagawa, 239-0847, Japan
`ryou.masumura.ba@hco.ntt.co.jp`

Abstract

This paper proposes an adversarial training method for the multi-task and multi-lingual joint modeling needed for utterance intent classification. In joint modeling, common knowledge can be efficiently utilized among multiple tasks or multiple languages. This is achieved by introducing both language-specific networks shared among different tasks and task-specific networks shared among different languages. However, the shared networks are often specialized in majority tasks or languages, so performance degradation must be expected for some minor data sets. In order to improve the invariance of shared networks, the proposed method introduces both language-specific task adversarial networks and task-specific language adversarial networks; both are leveraged for purging the task or language dependencies of the shared networks. The effectiveness of the adversarial training proposal is demonstrated using Japanese and English data sets for three different utterance intent classification tasks.

1 Introduction

In natural language processing fields, full neural network based methods are suitable for joint modeling as they can simultaneously utilize multiple task data sets or multiple language data sets to improve the performance achieved for individual tasks or languages (Collobert and Weston, 2008). It is known that joint modeling can address the data scarcity problem.

Key natural language processing technologies for spoken dialogue systems include utterance intent classification, which is needed to detect intent labels such as dialogue act (Stolcke et al., 2000; Khanpour et al., 2016), domain (Xu and Sarikaya, 2014), and question type (Wu et al., 2005) from input utterances (Ravuri and Stolcke, 2015a,b, 2016). One problem is that the training data are often limited or unbalanced among different tasks or different languages. Therefore, our motivation is to leverage both multi-task joint modeling and multi-lingual joint modeling to enhance utterance intent classification.

The multi-task and multi-lingual joint modeling can be composed by introducing both task-specific networks, which are shared among different languages, and language-specific networks, which are shared among different tasks (Masumura et al., 2018; Lin et al., 2018). Although joint modeling is mainly intended to improve classification performance in resource-poor tasks or languages, its classification performance is degraded in some minor data sets. This is because the language-specific networks often depend on majority tasks, while the task-specific networks often depend on majority languages. What are needed are task-specific networks that are invariant to languages, and language-specific networks that are invariant to tasks.

In order to explicitly improve the invariance of language and task-specific networks, this paper introduces adversarial training (Goodfellow et al., 2014). Our idea is to train language-specific networks so as to be insensitive to the target task, while training task-specific networks to be insensitive to language. To this end, we introduce multiple domain adversarial networks (Ganin et al., 2016), language-specific task adversarial networks, and task-specific language adversarial networks, into a state-of-the-art fully neural network based joint modeling; we adopt the bidirectional long short-term memory recurrent neural networks (BLSTM-RNNs) with attention mechanism (Yang et al., 2016; Zhou et al., 2016). To the best of our knowledge, this paper is the first study to employ adversarial training for multi-input and multi-output joint modeling.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 633–639
Brussels, Belgium, October 31 - November 4, 2018.

Experiments on Japanese and English data sets demonstrate the effectiveness of the adversarial training proposal. To support spoken dialogue systems, three different utterance intent classification tasks are examined: dialogue act, topic type, and question type classification.

2 Related Work

Joint Modeling: In natural language processing research, joint modeling is usually split into multi-task joint modeling and multi-lingual joint modeling. Multi-task joint modeling has been shown to effectively improve individual tasks (Collobert and Weston, 2008; Liu et al., 2016a,b; Zhang and Weng, 2016; Liu et al., 2016c). In addition, multi-lingual joint modeling is achieved by learning common semantic representations among different languages (Guo et al., 2016; Duong et al., 2016; Zhang et al., 2016, 2017b). In addition, a few work have examined multi-task and multi-lingual joint modeling (Masumura et al., 2018; Lin et al., 2018). Different from the previous work, our novelty is to introduce adversarial training for multi-task and multi-lingual joint modeling.

Adversarial Training: The concept of adversarial training was first proposed by Goodfellow et al. (2014), and many studies in the machine learning field have focused on adversarial training. Adversarial training has been well utilized in text classification (Ganin et al., 2016; Chen et al., 2016; Liu et al., 2017; Miyato et al., 2017; Chen and Cardie, 2018). Most natural language processing papers adopted either the language invariant approach (Chen et al., 2016; Zhang et al., 2017a) or the task invariant approach (Ganin et al., 2016; Liu et al., 2017; Chen and Cardie, 2018). This paper aims to fully utilize both task adversarial training and language adversarial training. To this end, we simultaneously introduce language-specific task adversarial networks and task-specific language adversarial networks.

3 Proposed Method

This section details our adversarial training method for multi-task and multi-lingual joint modeling of utterance intent classification.

In the j-th task utterance intent classification for the i-th language input utterance, intent label $l^{(j)} \in \{1, \cdots, K^{(j)}\}$ is estimated from input utterance $\mathcal{W}^{(i)} = \{w_1^{(i)}, \cdots, w_T^{(i)}\}$ where $i \in \{1, \cdots, I\}$ and $j \in \{1, \cdots, J\}$. Utterance intent classification is followed by estimation of the probabilities of each intent label given input utterance, $P(l^{(j)}|\mathcal{W}^{(i)}, \boldsymbol{\Theta}^{(i,j)})$ where $\boldsymbol{\Theta}^{(i,j)}$ is the trainable model parameter for the combination of the i-th language and the j-th task. In multi-task and multi-lingual joint modeling, $\{\boldsymbol{\Theta}^{(1,1)}, \cdots, \boldsymbol{\Theta}^{(I,J)}\}$ are jointly trained from I language and J task data sets.

3.1 Main Joint Network

The proposed method is founded on a fully neural network that employs I language-specific networks, J task-specific networks, and J classification networks as well as Masumura et al. (2018).

The language-specific network can be shared between multiple tasks, where words in the input utterance are converted into language-specific hidden representations. Each word in the i-th language input utterance $\mathcal{W}^{(i)}$ is first converted into a continuous representation. Next, each word representation is converted into a hidden representation that uses BLSTM-RNNs to take neighboring word context information into account. The t-th language-specific hidden representation for the i-th language is given by:

$$\boldsymbol{w}_t^{(i)} = \texttt{EMBED}(w_t^{(i)}; \boldsymbol{\theta}_{\mathrm{h}}^{(i)}), \tag{1}$$

$$\boldsymbol{h}_t^{(i)} = \texttt{BLSTM}(\{\boldsymbol{w}_1^{(i)}, \cdots, \boldsymbol{w}_T^{(i)}\}, t; \boldsymbol{\theta}_{\mathrm{h}}^{(i)}), \tag{2}$$

where $\texttt{EMBED}()$ is a linear transformational function for word embedding, $\texttt{BLSTM}()$ is a function of the BLSTM-RNN layer, and $\boldsymbol{\theta}_{\mathrm{h}}^{(i)}$ is the trainable parameter for the i-th language-specific network.

In addition, task-specific networks can be shared between multiple languages, where the language-specific hidden representations are converted into task-specific hidden representations. The t-th language-specific hidden representation for the j-th task is given by:

$$\boldsymbol{u}_t^{(j)} = \texttt{BLSTM}(\{\boldsymbol{h}_1^{(i)}, \cdots, \boldsymbol{h}_T^{(i)}\}, t; \boldsymbol{\theta}_{\mathrm{u}}^{(j)}), \tag{3}$$

where $\boldsymbol{\theta}_{\mathrm{u}}^{(j)}$ is the trainable parameter for the j-th task-specific network.

In classification networks for each task, the task-specific hidden representations are summarized as sentence representation $\boldsymbol{s}^{(j)}$ by using a self-attention mechanism that can consider the importance of individual hidden representations (Yang et al., 2016; Zhou et al., 2016; Sawada et al., 2017). Next, predicted probabilities of the j-th task intent labels, $\boldsymbol{o}^{(j)} \in \mathbb{R}^{K^{(j)}}$, are given by:

$$\boldsymbol{s}^{(j)} = \mathtt{ATTENSUM}(\{\boldsymbol{h}_1^{(i)}, \cdots, \boldsymbol{h}_T^{(i)}\}; \boldsymbol{\theta}_{\rm o}^{(j)}), \quad (4)$$

$$\boldsymbol{o}^{(j)} = \mathtt{SOFTMAX}(\boldsymbol{s}^{(j)}; \boldsymbol{\theta}_{\rm o}^{(j)}), \quad (5)$$

where $\mathtt{ATTENSUM}()$ is a weighted sum function with self-attention, $\mathtt{SOFTMAX}()$ is a transformational function with softmax activation, and $\boldsymbol{\theta}_{\rm o}^{(j)}$ is the trainable parameter for the j-th classification network. In the main joint networks of the proposal, $\boldsymbol{\Theta}^{(i,j)}$ corresponds to $\{\boldsymbol{\theta}_{\rm h}^{(i)}, \boldsymbol{\theta}_{\rm u}^{(j)}, \boldsymbol{\theta}_{\rm o}^{(j)}\}$.

3.2 Adversarial Networks

The proposed method combines a language-specific task adversarial network with a task-specific language adversarial network. The task adversarial network is used for training the language-specific networks to be insensitive to target task labels, and the language adversarial network is used for training the task-specific networks to be insensitive to target language labels. In order to efficiently use stochastic gradient descent based training for optimizing the adversarial networks, we use gradient reversal layers, which allow the input vectors during forward propagation, and sign inversion of the gradients during back propagation, to be utilized (Ganin et al., 2016).

The i-th language-specific task adversarial network estimates task labels from the i-th language-specific hidden representations. The predicted probabilities of task labels, $\boldsymbol{x}^{(i)} \in \mathbb{R}^J$, are given by:

$$\tilde{\boldsymbol{h}}_t^{(i)} = \mathtt{GRL}(\boldsymbol{h}_t^{(i)}), \quad (6)$$

$$\tilde{\boldsymbol{h}}^{(i)} = \mathtt{ATTENSUM}(\{\tilde{\boldsymbol{h}}_1^{(i)}, \cdots, \tilde{\boldsymbol{h}}_T^{(i)}\}; \boldsymbol{\theta}_{\rm x}^{(i)}), \quad (7)$$

$$\boldsymbol{x}^{(i)} = \mathtt{SOFTMAX}(\tilde{\boldsymbol{h}}^{(i)}, \boldsymbol{\theta}_{\rm x}^{(i)}), \quad (8)$$

where $\mathtt{GRL}()$ represents the gradient reversal layer, and $\boldsymbol{\theta}_{\rm x}^{(i)}$ is the trainable parameter. The j-th task-specific language adversarial network estimates language labels from the j-th task-specific hidden representations. The predicted probabilities of language labels, $\boldsymbol{y}^{(j)} \in \mathbb{R}^I$, are given by:

$$\tilde{\boldsymbol{u}}_t^{(j)} = \mathtt{GRL}(\boldsymbol{u}_t^{(j)}), \quad (9)$$

$$\tilde{\boldsymbol{u}}^{(j)} = \mathtt{ATTENSUM}(\{\tilde{\boldsymbol{u}}_1^{(j)}, \cdots, \tilde{\boldsymbol{u}}_T^{(j)}\}; \boldsymbol{\theta}_{\rm y}^{(j)}), \quad (10)$$

$$\boldsymbol{y}^{(j)} = \mathtt{SOFTMAX}(\tilde{\boldsymbol{u}}^{(j)}, \boldsymbol{\theta}_{\rm y}^{(j)}), \quad (11)$$

where $\boldsymbol{\theta}_{\rm y}$ is the trainable parameter.

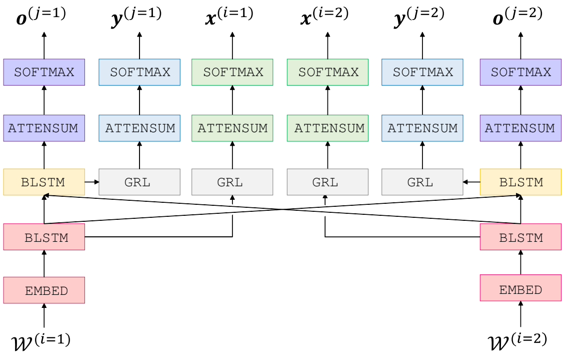

Figure 1: Proposed network structure for two tasks and two languages.

The proposed network structure shown in Figure 1 includes both joint networks and adversarial networks for two tasks and two languages. The red components are language-specific networks, the orange components are task-specific networks, and the purple components are classification networks. In addition, green components are language-specific task adversarial networks, and blue components are task-specific language adversarial networks.

3.3 Training

Our adversarial training proposal jointly optimizes all parameters in both the main joint networks and the adversarial networks by using all training data sets $\{\mathcal{D}^{(1,1)}, \cdots, \mathcal{D}^{(I,J)}\}$ where $\mathcal{D}^{(i,j)}$ represents the sets of the input utterances and the reference. The cross-entropy loss functions of each network are defined as:

$$L_{\boldsymbol{o}} = -\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{|\mathcal{D}^{(i,j)}|}\sum_{k=1}^{K^{(j)}} \hat{\boldsymbol{o}}_{n,k}^{(j)} \log \boldsymbol{o}_{n,k}^{(j)}, \quad (12)$$

$$L_{\boldsymbol{x}} = -\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{|\mathcal{D}^{(i,j)}|}\sum_{j'=1}^{J} \hat{\boldsymbol{x}}_{n,j'}^{(i)} \log \boldsymbol{x}_{n,j'}^{(i)}, \quad (13)$$

$$L_{\boldsymbol{y}} = -\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{|\mathcal{D}^{(i,j)}|}\sum_{i'=1}^{I} \hat{\boldsymbol{y}}_{n,i'}^{(j)} \log \boldsymbol{y}_{n,i'}^{(j)}, \quad (14)$$

where $L_{\boldsymbol{o}}$, $L_{\boldsymbol{x}}$, and $L_{\boldsymbol{y}}$ are the cross entropy loss terms for the classification networks, the task adversarial networks, and the language adversarial networks. $\hat{\boldsymbol{o}}_{n,k}^{(j)}$, $\hat{\boldsymbol{x}}_{n,j'}^{(i)}$, and $\hat{\boldsymbol{y}}_{n,i'}^{(j)}$ are the reference probabilities, and $\boldsymbol{o}_{n,k}$, $\boldsymbol{x}_{n,j'}$, and $\boldsymbol{y}_{n,i'}$ are the estimated probabilities of the k-th label in the j-th task classification network, the j'-th task in the i-th language-specific task adversarial network, and

Task	Utterance	Label
DA	Hello, how are you today?	GREETING
	I am so sorry to hear of your son's accident.	SYMPATHY/AGREE
	Lets go to school an hour early today.	PROPOSAL
TT	What is the highest mountain in the world?	MOUNTAIN
	Who is president of the united states?	PERSON
	What is the name of the most recent Star Wars movie?	MOVIE
QT	Do you like egg salad?	TRUE/FALSE
	How do you correct a hook in a golf swing?	EXPLANATION:METHOD
	Why is blood red?	EXPLANATION:CAUSE

Table 2: Examples of English data sets.

the i'-th language in the j-th task-specific language adversarial network for $\mathcal{W}_n$, respectively.

Due to use of gradient reversal layers, individual parameters are gradually updated as follows:

$$\boldsymbol{\theta}_{\mathtt{o}}^{(j)} \leftarrow \boldsymbol{\theta}_{\mathtt{o}}^{(j)} - \epsilon \frac{\partial L_{\boldsymbol{o}}}{\partial \boldsymbol{\theta}_{\mathtt{o}}^{(j)}}, \tag{15}$$

$$\boldsymbol{\theta}_{\mathtt{y}}^{(j)} \leftarrow \boldsymbol{\theta}_{\mathtt{y}}^{(j)} - \epsilon \beta \frac{\partial L_{\boldsymbol{y}}}{\partial \boldsymbol{\theta}_{\mathtt{y}}^{(j)}}, \tag{16}$$

$$\boldsymbol{\theta}_{\mathtt{u}}^{(j)} \leftarrow \boldsymbol{\theta}_{\mathtt{u}}^{(j)} - \epsilon \Big(\frac{\partial L_{\boldsymbol{o}}}{\partial \boldsymbol{\theta}_{\mathtt{u}}^{(j)}} - \beta \frac{\partial L_{\boldsymbol{y}}}{\partial \boldsymbol{\theta}_{\mathtt{u}}^{(j)}} \Big), \tag{17}$$

$$\boldsymbol{\theta}_{\mathtt{x}}^{(i)} \leftarrow \boldsymbol{\theta}_{\mathtt{x}}^{(i)} - \epsilon \alpha \frac{\partial L_{\boldsymbol{x}}}{\partial \boldsymbol{\theta}_{\mathtt{x}}^{(i)}}, \tag{18}$$

$$\boldsymbol{\theta}_{\mathtt{h}}^{(i)} \leftarrow \boldsymbol{\theta}_{\mathtt{h}}^{(i)} - \epsilon \Big(\frac{\partial L_{\boldsymbol{o}}}{\partial \boldsymbol{\theta}_{\mathtt{h}}^{(i)}} - \alpha \frac{\partial L_{\boldsymbol{x}}}{\partial \boldsymbol{\theta}_{\mathtt{h}}^{(i)}} - \beta \frac{\partial L_{\boldsymbol{y}}}{\partial \boldsymbol{\theta}_{\mathtt{h}}^{(i)}} \Big), \tag{19}$$

where α and β are hyper parameters of the parameter update, and ϵ is the learning rate. Note that adversarial training is suppressed by setting α and β to 0.0. In training, we prepared optimizers for individual data sets. The individual learning rates fall when the validation loss of the target classification network increases.

4 Experiments

Our experiments employed Japanese and English data sets created for three different utterance intent classification tasks. The tasks, dialogue act (DA) classification, topic type (TT) classification, and question type (QT) classification, are intended to support spoken dialogue systems. For example, the task of English DA classification is to obtain a DA label from an input utterance. We used natural language texts as the input utterances and individual label sets were unified between Japanese and English. Data sets employed in experiments were corpora that were made for constructing spoken dialogue systems (Masumura et al., 2018). Each of the data sets were divided into training (Train), validation (Valid), and test (Test) sets. Table 1 shows the number of utterances in individual data sets where #labels represents the number of labels. Table 2 shows English utterances and label examples for individual tasks.

Language	Task	#labels	Train	Valid	Test
Japanese	DA	28	201 K	4 K	4 K
	TT	168	40 K	4 K	4 K
	QT	15	55 K	4 K	4 K
English	DA	28	25 K	3 K	3 K
	TT	168	25 K	3 K	3 K
	QT	15	22 K	2 K	2 K

Table 1: Number of utterances in individual data sets.

4.1 Setups

We examined single-task and mono-lingual modeling, multi-task joint modeling, multi-lingual join modeling, and multi-task and multi-lingual joint modeling with or without adversarial training.

We unified network configurations as follows. Word representation size was set to 128, BLSTM-RNN unit size was set to 400, and sentence representation was set to 400. Dropout was used for `EMBED()` and `BLSTM()`, and the dropout rate was set to 0.5. Words that appeared only once in the training data sets were treated as unknown words. We used mini-batch stochastic gradient descent, in which initial learning rate was set to 0.1. We optimized hyper-parameters of adversarial training (α and β) for the validation sets by varying them from 0.001 to 1.0. Other hyper parameters were also optimized for the validation sets.

4.2 Results

Table 3 shows the results in terms of utterance classification accuracy. For each setup, we constructed five models by varying the initial parameters and evaluated the average accuracy. Line (1) shows baseline results: single-task and mono-lingual modeling. Lines (2) and (3) show results

	Joint modeling		Adversarial Training		Japanese			English		
	Multi-task	Multi-lingual	Task-invariant	Language-invariant	DA	TT	QT	DA	TT	QT
(1).	-	-	-	-	66.6	79.1	87.7	61.8	64.5	83.4
(2)	√	-	-	-	66.5	79.6	89.3	60.6	64.4	**83.7**
(3)	√	-	√	-	66.5	80.6	89.5	61.6	65.7	**83.7**
(4)	-	√	-	-	66.7	78.7	87.2	61.4	64.3	83.0
(5)	-	√	-	√	66.9	79.8	88.2	61.8	64.8	83.3
(6).	√	√	-	-	66.6	79.7	89.3	60.5	65.4	82.6
(7).	√	√	√	-	67.3	81.1	89.6	61.5	66.1	83.5
(8).	√	√	-	√	66.7	80.7	89.5	60.9	**66.7**	83.0
(9).	√	√	√	√	**67.6**	**81.3**	**90.0**	**61.9**	**66.7**	**83.7**

Table 3: Experimental results: utterance classification accuracy (%) for individual test sets.

with only performing multi-task joint modeling, and lines (4) and (5) show results with only performing multi-lingual joint modeling. Note that lines (3) and (5) show the results achieved with adversarial training. Line (6) shows multi-task and multi-lingual joint modeling results: adversarial training was suppressed by setting both α and β to 0.0. Lines (7)–(9) shows the results achieved with adversarial training. Note that setting with bold values achieved the highest performance in our evaluation.

First, in lines (2) and (4), the classification performance deteriorated in some cases, while performance improvements were achieved in other cases. On the other hand, in lines (3) and (5), classification performance in each data sets was improved by introducing adversarial training. This indicates that adversarial training was effective in improving the performance of joint modeling.

Next, line (6) shows that, relative to line 1, multi-task and multi-lingual joint modeling can improve the classification performance for Japanese TT, Japanese QT, and English TT, but classification performance was degraded for English DA and English QT. This indicates that it is difficult to simultaneously improve the classification performance for all data sets because joint modeling often depends on majority tasks or majority languages. In addition, lines (7) and (8) show the introduction of either task adversarial networks or language adversarial networks yielded better performance than line (6) for all data sets. This indicates that adversarial training was effective in improving the performance of multi-task and multi-lingual joint modeling. The best results were achieved by using both language-specific task adversarial networks and task-specific language adversarial networks, line (9). These results confirm that task adversarial networks and language adversarial networks well complement each other. Of particular benefit, the proposed method demonstrated greater classification performance improvements when the number of training utterances per label was small.

5 Conclusions

We have proposed an adversarial training method for the multi-task and multi-lingual joint modeling needed to enhance utterance intent classification. Our adversarial training proposal utilizes both task adversarial networks and language adversarial networks for improving task-invariance in language-specific networks and language-invariance in task-specific networks. Experiments showed that the adversarial training proposal could well realize the benefits of joint modeling in all data sets.

References

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. *arXiv preprint arXiv:1802.05694*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *In Proc. International Conference on Machine Learning (ICML)*.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. *In Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1285–1295.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Francois Laviolette, Mario Marchand, and Victor Lempitsky.

2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17:1–35.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *In Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. *In Proc. AAAI Conference on Artificial Intelligence (AAAI)*, pages 2734–2740.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. *In Proc. International Conference on Computational Linguistics (COLING)*, pages 2012–2021.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. *In Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages pp.799–809.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016a. Deep multi-task learning with shared memory. *In Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 118–127.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016b. Recurrent neural network for text classification with multi-task learning. *In Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2873–2879.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *In Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–10.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016c. Implicit discourse relation classification via multi-task neural networks. *In Proc. AAAI Conference on Artificial Intelligence (AAAI)*, pages 2750–2756.

Ryo Masumura, Tomohiro Tanaka, Ryuichiro Higashinaka, Hirokazu Masataki, and Yushi Aono. 2018. Multi-task and multi-lingual joint learning of neural lexical utterance classification based on partially-shared modeling. *In Proc. International Conference on Computational Linguistics (COLING)*, pages pp.3586–3596.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. *In Proc. International Conference on Learning Representation (ICLR)*.

Suman Ravuri and Andreas Stolcke. 2015a. A comparative study of neural network models for lexical intent classification. *In Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 368–374.

Suman Ravuri and Andreas Stolcke. 2015b. Recurrent neural network and LSTM models for lexical utterance classification. *In Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 135–139.

Suman Ravuri and Andreas Stolcke. 2016. A comparative study of recurrent neural network models for lexical domain classification. *In Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6075–6079.

Naoki Sawada, Ryo Masumura, and Hiromitsu Nishizaki. 2017. Parallel hierarchical attention networks with shared memory reader for multi-stream conversational document classification. *In Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 3311–3315.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martion, Carol Van Ess-Dykema, and Marie Metter. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Chung-Hsien Wu, Jui-Feng Yeh, and Ming-Jun Chen. 2005. Domain-specific FAQ retrieval using independent aspects. *ACM Transactions on Asian Language Information Processing*, 4(1):1–17.

Puyang Xu and Ruhi Sarikaya. 2014. Contextual domain classification in spoken language understanding systems using recurrent neural network. *In Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 136–140.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. *In Proc. Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1480–1489.

Meng Zhang, Yang Liu, Huanbo Luan, Yiqun Liu, and Maosong Sun. 2016. Inducing bilingual lexica from non-parallel data with earth mover's distance regularization. *In Proc. International Conference on Computational Linguistics (COLING)*, pages 3188–3198.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017a. Adversarial training for unsupervised bilingual lexicon induction. *In Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1959–1970.

Meng Zhang, Haoruo Peng, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Bilingual lexicon induction from non-parallel data with minimum supervision. *In Proc. AAAI Conference on Artificial Intelligence (AAAI)*, pages 3379–3384.

Xiaodong Zhang and Houfeng Weng. 2016. A joint model of intent determination and slot filling for spoken language understanding. *In Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2993–2999.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. *In Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 207–212.

Surprisingly Easy Hard-Attention for Sequence to Sequence Learning

Shiv Shankar[*]
IIT Bombay
`shiv_shankar@iitb.ac.in`

Siddhant Garg[*]
IIT Bombay
`sidgarg@cs.wisc.edu`

Sunita Sarawagi
IIT Bombay
`sunita@iitb.ac.in`

Abstract

In this paper we show that a simple beam approximation of the joint distribution between attention and output is an easy, accurate, and efficient attention mechanism for sequence to sequence learning. The method combines the advantage of sharp focus in hard attention and the implementation ease of soft attention. On five translation and two morphological inflection tasks we show effortless and consistent gains in BLEU compared to existing attention mechanisms.

1 Introduction

In structured input-output models as used in tasks like translation and image captioning, the attention variable decides which part of the input aligns to the current output. Many attention mechanisms have been proposed (Xu et al., 2015; Bahdanau et al., 2014; Luong et al., 2015; Martins and Astudillo, 2016) but the de facto standard is a soft attention mechanism that first assigns attention weights to input encoder states, then computes an attention weighted 'soft' aligned input state, which finally derives the output distribution. This method is end to end differentiable and easy to implement.

Another less popular variant is hard attention that aligns each output to exactly one input state but requires intricate training to teach the network to choose that state. When successfully trained, hard attention is often found to be more accurate (Xu et al., 2015; Zaremba and Sutskever, 2015). In NLP, a recent success has been in a monotonic hard attention setting in morphological inflection tasks (Yu et al., 2016; Aharoni and Goldberg, 2017). For general seq2seq learning, methods like SparseMax (Martins and Astudillo, 2016) and local attention (Luong et al., 2015) were proposed to bridge the gap between soft and hard attention.

In this paper we propose a surprisingly simpler alternative based on the original joint distribution between output and attention, of which existing soft and hard attention mechanisms are approximations. The joint model couples input states individually to the output like in hard attention, but it combines the advantage of end-to-end trainability of soft attention. When the number of input states is large, we propose to use a simple approximation of the full joint distribution called Beam-joint. This approximation is also easily trainable and does not suffer from the high variance of Monte-Carlo sampling gradients of hard attention.

We evaluated our model on five translation tasks and increased BLEU by 0.8 to 1.7 over soft attention, which in turn was better than hard and the recent Sparsemax (Martins and Astudillo, 2016) attention. More importantly, the training process was as easy as soft attention. For further support, we also evaluate on two morphological inflection tasks and got gains over soft and hard attention.

2 Background and Related Work

For sequence to sequence (seq2seq) learning the encoder-decoder model is the standard and we review it here. We then review related work on attention mechanisms on these models.

2.1 Attention-based Encoder Decoder Model

Let $x_1, \ldots, x_m$ denote the tokens in the input sequence that have been transformed by an encoder network to state vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$, which we jointly denote as $\mathbf{x}_{1\ldots m}$. Let $y_1, \ldots, y_n$ denote the output tokens in the target sequence. The Encoder-Decoder (ED) network factorizes $\Pr(y_1, \ldots, y_n|\mathbf{x}_{1\ldots m})$ as $\prod_{t=1}^{n} \Pr(y_t|\mathbf{x}_{1\ldots m}, \mathbf{s}_t)$ where $\mathbf{s}_t$ is a decoder state summarizing $y_1, \ldots y_{t-1}$. For each t, a hidden attention variable a_t is used to denote which part of $\mathbf{x}_{1\ldots m}$ aligns with y_t. Let $P(a_t = j|\mathbf{x}_{1\ldots m}, \mathbf{s}_t)$ denote the

---

[*] Both authors contributed equally to this work

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 640–645
Brussels, Belgium, October 31 - November 4, 2018.

probability that encoder state $\mathbf{x}_j$ is relevant for output y_t. Typically this is estimated using a softmax function over attention scores computed from $\mathbf{x}_j$ and decoder state $\mathbf{s}_t$ as follows.

$$P(a_t = j|\mathbf{x}_{1\ldots m}, \mathbf{s}_t) = \frac{e^{A_\theta(\mathbf{x}_j,\mathbf{s}_t)}}{\sum_{r=1}^{m} e^{A_\theta(\mathbf{x}_r,\mathbf{s}_t)}} \quad (1)$$

where $A_\theta(.,.)$ is the attention unit that scores each input state $\mathbf{x}_j$ as per the decoder state $\mathbf{s}_t$. Thereafter, in the popular soft-attention mechanism, the attention weighted sum of the input states is used to model log likelihood for each y_t as

$$\log \Pr(y_t|\mathbf{x}_{1\ldots m}) = \log \Pr(y_t|\sum_a P_t(a)\mathbf{x}_a) \quad (2)$$

where $P_t(a_t = j)$ is the short form for $P(a_t = j|\mathbf{x}_{1\ldots m}, \mathbf{s}_t)$. Also, here and in the rest of the paper we drop $\mathbf{s}_t$ from $P(y_t)$ and $P_t(a)$ for ease of notation. The weighted sum $\sum_a P_t(a)\mathbf{x}_a$ is called an input context $\mathbf{c}_t$ which is fed to the decoder RNN along with y_t for computing the next state $\mathbf{s}_{t+1}$.

2.2 Related Work

We next review existing attention types.

Soft Attention is the attention method described in the previous section and is the current standard for seq2seq learning (Xu Chen, 2018; Koehn, 2017). It was proposed for translation in (Bahdanau et al., 2014) and refined further in (Luong et al., 2015). As shown in Eq 2, here each output is derived from an attention averaged input. This diffuses the coupling between the input and output. The advantage of soft attention is end to end differentiability, and fast training and inference.

Hard Attention was proposed in its current form in (Xu et al., 2015) and attends to exactly one input state for an output[1]. During training, log-likelihood is an expectation over sampled attentions:

$$\log P_t(y_t|\mathbf{x}_{1\ldots m}) = \sum_{l=1}^{M} \log P_t(y_t|\mathbf{x}_{\tilde{a_l}}) \quad (3)$$

where $\tilde{a_1}, \ldots, \tilde{a_M}$ are sampled from the multinomial $P_t(a)$. Because of the sampling, the gradient has to be computed by Monte Carlo gradient/REINFORCE (Williams, 1992) and is subject to high variance. Many tricks are required to train hard attention and there is little standardization across implementations. Xu et al (2015) use a combination of REINFORCE and soft attention. Zaremba et al(2015) uses curriculum learning that starts as soft-attention and gradually becomes discrete. Ling& Rush (2017) aggregates multiple samples during training, and a single sampled attention while testing. However, once trained well the sharp focus on memory provided by hard-attention has been found to yield superior performance (Xu et al., 2015; Shankar and Sarawagi, 2018).

Sparse/Local Attention Many attempts have been made to bridge the gap between soft and hard attention. Luong et al (2015) proposes local attention that averages a window of input. This has been refined later to include syntax (Chen et al., 2017; Sennrich and Haddow, 2016; Chen et al., 2018). Another idea is to replace the softmax in soft attention with sparsity inducing operators (Martins and Astudillo, 2016; Niculae and Blondel, 2017). However, all sparse/local attention methods continue to compute $P(y)$ from an attention weighted sum of inputs (Eq: 2) unlike hard attention.

3 Joint Attention-Output Models

We start from an explicit joint representation of the uncertainty of the attention and output variables.

$$\log P_t(y_t|\mathbf{x}_{1\ldots m}) = \log \sum_a P_t(a)P_t(y_t|\mathbf{x}_a) \quad (4)$$

The joint model directly couples individual input states to the output, and thus is a type of hard attention. Also, by taking an expectation, instead of a single hard attention, it enjoys differentiability as in soft-attention. We call this the **full-joint** method.

Unfortunately, either when the vocabulary or the number of encoder states (m) is large, full-joint is not practical. Existing hard and soft attentions can be viewed as its approximations that either marginalize early or hard select attention. We show a surprisingly simple alternative approximation that provides hard attention without its training complexity. Our method called **Beam-joint** deterministically selects the top-k highest attention values and approximates the full joint log probability as

$$\log P_t(y_t|\mathbf{x}_{1\ldots m}) \approx \log \sum_{a \in \text{TopK}(P_t(a))} P_t(a)P_t(y_t|\mathbf{x}_a) \quad (5)$$

Thus, in beam-joint, we first compute the multinomial attention distribution in $O(m)$ time using

---

[1] Note, attention on a single input encoder state does not imply attention on a single input token because RNNs or self-attention capture the context around the token.

Eq 1, select the Top-K input positions from the multinomial, next with hard attention on each position compute K output softmax, and finally compute the attention weighted output mixture distribution. The number of output softmax is K times in normal soft-attention but the actual running time overhead is only 20–30% for translation tasks. We used the default pass-through TopK operator (which is not differentiable) and optimize the beam-approximation directly. We also experimented with a version which smoothly shifts from soft-attention to beam-attention, but found that training the beam-approximation directly leads to best results.

We show empirically that this very simple scheme is surprisingly effective compared to existing hard and soft attention over several translation tasks. Unlike sampling and variational methods that require careful tuning and exotic tricks during training, this simple scheme trains as easily as soft-attention, without significant increase in training time because even $K = 6$ works well enough.

Another reason why our 'sum of probabilities' form performs better could be the softmax barrier effect highlighted in (Yang et al., 2018). The authors argue that the richness of natural language cannot be captured in normal softmax due to the low rank constraint it imposes on input-to-output matrix. They improve performance using a Mixture of Softmax model. Our beam-joint also is a mixture of softmax and possibly achieves higher rank than a single softmax. However their mixture requires learning multiple softmax matrices, whereas ours are due to varying attention and we do not learn any extra parameters than soft attention.

4 Experiments

We compare attention models on two NLP tasks: machine translation and morphological inflection.

4.1 Machine translation

We experiment on five language pairs from three datasets: **IWSLT15 English↔Vietnamese** (Cettolo et al., 2015) which contains 133k train, 1.5k validation(tst2012) and 1.2k test(tst2013) sentence pairs respectively; **IWSLT14 German↔English** (Cettolo et al., 2014) which contains 160k train, 7.2k validation and 6.7k test sentence pairs respectively ; **Workshop on Asian Translation 2017 Japanese→English** (Nakazawa et al., 2016) which contains 2M train, 1.8k validation and 1.8k test sentence pairs respectively. We use a 2 layer bidirectional encoder and a 2 layer unidirectional decoder with 512 hidden LSTM units and 0.2 dropout rate with vanilla SGD optimizer. We base our implementation[2] on the NMT code[3] in Tensorflow. We did no special hyper-parameter tuning and used standard-softmax tuned parameters on a batch size of 64.

Comparing attention models We compare beam-joint (default $K = 6$) with standard soft and hard attention. To further dissect the reasons behind beam-joint's gains, we compare beam-joint with a sampling based approximation of full-joint called Sample-Joint that replaces the TopK in Eq 5 with K attention weighted samples. We train sample-joint as well as hard-attention with REINFORCE with 6-samples. Also to ascertain that our gains are not explained by sparsity alone, we compare with Sparsemax (Martins and Astudillo, 2016).

In Table 1 we show perplexity and BLEU with three beam sizes (B). Beam-joint significantly outperforms all other variants, including the standard soft attention by 0.8 to 1.7 BLEU points. The perplexity shows even a more impressive drop in all five datasets. Also we observe training times for beam-joint to be only 20–30% higher than soft-attention, establishing that beam-joint is both practical and more accurate.

Sample-joint is much worse than beam-joint. Apart from the problem of high variance of gradients in the reinforce step, another problem is that sampling repeats states whereas TopK in beam-joint gets distinct states. Hard attention too faces training issues and performs worse than soft attention, explaining why it is not commonly used in NMT. Sample-joint is better than Hard attention, further highlighting the merits of the joint distribution. Sparsemax is competitive but marginally worse than soft attention. This is concordant with the recent experiments of (Niculae and Blondel, 2017).

Comparison with Full Joint Next we evaluate the impact of our beam-joint approximation against full-joint and soft attention. Full-joint cannot scale to large vocabularies, therefore we only compare on En-Vi with a batch size of 32. Figure 1a shows final BLEU of these methods as well as BLEU against increasing training steps. Beam-joint both converges faster and to a higher score than soft-

---

[2] https://github.com/sid7954/beam-joint-attention

[3] https://github.com/tensorflow/nmt

Dataset	Attention	PPL	BLEU		
			B=1	B=4	B=10
IWSLT14 DE-EN	Soft	9.61	27.7	28.6	28.5
	Hard	9.50	25.3	25.6	25.5
	Sparse	9.85	27.2	28.4	28.0
	Sample-Joint	9.96	26.3	27.8	27.8
	Beam-Joint	**8.47**	29.0	**29.7**	29.6
IWSLT14 EN-DE	Soft	10.68	23.1	24.2	24.2
	Hard	10.15	21.4	21.8	21.7
	Sparse	10.89	22.5	23.4	23.3
	Sample-Joint	10.05	22.8	23.8	23.6
	Beam-Joint	**8.72**	24.7	**25.4**	25.3
IWSLT15 EN-VI	Soft	10.27	26.0	26.6	26.4
	Hard	10.53	24.1	24.3	24.0
	Sparse	10.13	25.9	26.6	26.1
	Sample-Joint	11.00	25.8	26.3	25.9
	Beam-Joint	**9.67**	27.0	**27.4**	27.3
IWSLT14 VI-EN	Soft	8.30	23.6	24.7	24.6
	Hard	8.28	21.1	21.9	21.5
	Sparse	8.48	22.8	24.2	23.9
	Sample-Joint	8.28	22.7	24.0	23.9
	Beam-Joint	**7.57**	24.5	**25.8**	25.7
WAT17 JA-EN	Soft	12.46	17.6	18.9	18.5
	Hard	12.78	13.2	13.1	12.7
	Sparse	14.18	16.7	17.5	16.8
	Sample-Joint	13.21	16.2	18.1	17.9
	Beam-Joint	**10.00**	19.6	**20.6**	20.2

Table 1: Perplexity and test BLEU with three inference beam widths (B) on five translation tasks. Beam-joint consistently and substantially outperforms soft-attention.

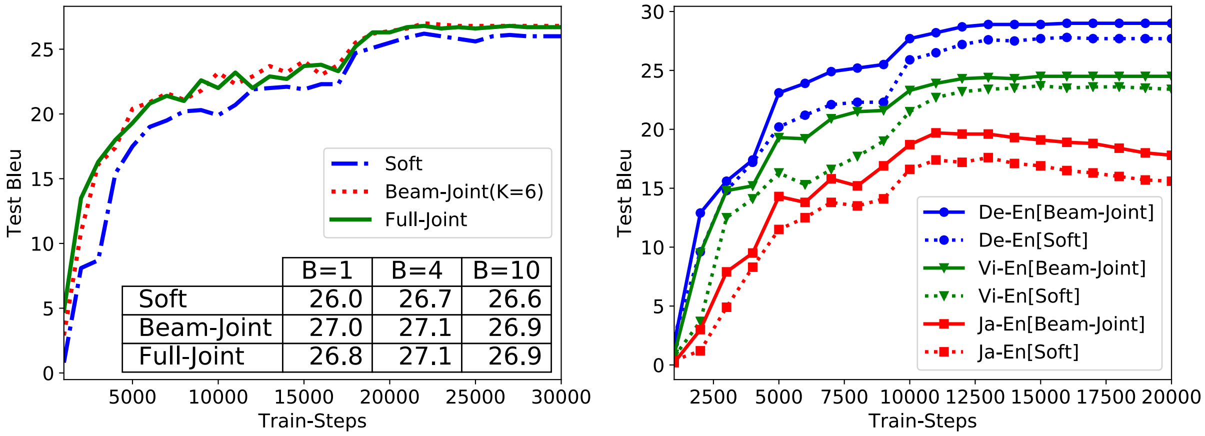

	B=1	B=4	B=10
Soft	26.0	26.7	26.6
Beam-Joint	27.0	27.1	26.9
Full-Joint	26.8	27.1	26.9

(a) BLEU Vs Training steps on En-Vi

(b) BLEU Vs Training steps (Beam=1)

Figure 1: Test BLEU in various settings (Beam=1). Best viewed in color

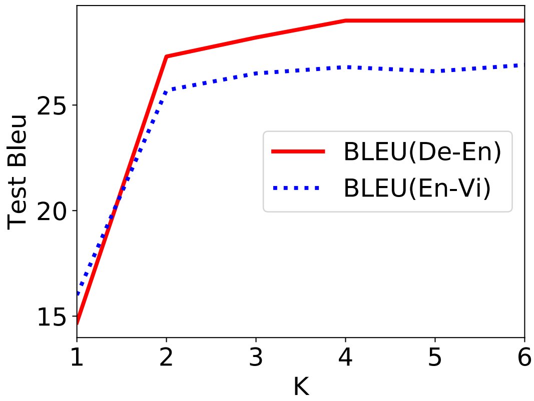

Figure 2: BLEU for Beam-Joint with increasing K. BLEU increases with K and saturates at K=5

attention. For example by 10000 steps (5 epochs), beam-joint has surpassed soft-attention by almost 2 BLEU points (20 vs 22). Moreover beam-joint tracks full-joint well, and both converge finally to similar BLEUs near 27 against 26 for soft attention. This shows that an attention-beam of size 6 suffices to approximate full joint almost perfectly.

Next, in Figure 1b, we compare beam-joint (solid lines) and soft attention (dotted lines) for convergence rates on three other datasets. For each dataset beam-joint trains faster with a consistent improvement of more than 1 BLEU.

Effect of K in Beam-joint We show the effect of K used in TopK of beam-joint in Figure 2 on the En-Vi and De-En tasks. On En-Vi BLEU increases from 16.0 to 25.7 to 26.5 as K increases from 1 to 2 to 3; and then saturates quickly. Similar behavior is observed in the other dataset. This shows that small K values like 6 suffice for translation.

We further evaluate whether the performance gain of beam-joint is due to the softmax barrier alone in Table 2. We used our models trained with K=6, and deployed them for test-time greedy decoding with K set to 1. Since the output now has only a single softmax component, this model faces the same bottleneck as soft-attention. One can observe that as expected these results are worse than beam-joint with K=6, however they still exceed soft-attention by a significant margin, demonstrating that the performance gain is not solely due to the effect of ensembling or softmax-barrier.

4.2 Morphological Inflection

To demonstrate the use of this approach beyond translation, we next consider two morphological inflection tasks. We use (Durrett and DeNero, 2013)'s dataset containing 8 inflection forms for German Nouns (de-N) and 27 forms for German Verbs (de-V). The number of training words is 2364 and 1627 respectively while the validation and test words are 200 each. We train a one layer encoder and decoder with 128 hidden LSTM units each with a dropout rate of 0.2 using Adam(Kingma and Ba, 2014) and measure 0/1 accuracy for soft, hard and full-joint attention models. Due to limited input length and vocabulary, we were able to run directly the full-joint model. We also ran the 100 units wide two layer LSTM with hard-monotonic attention provided by (Aharoni and Goldberg, 2017) labeled Hard-Mono[4]. The table below shows that even for this task full-joint scores over existing attention models[5]. The generic full-joint attention provides slight gains even over the task specific hard-monotonic attention.

Dataset	Soft	Beam-Joint (K=1)	Beam-Joint (K=6)
En-De	23.1	24.5	**24.7**
De-En	27.7	28.4	**29.0**
En-Vi	26.0	26.5	**27.0**

Table 2: Comparing soft attention with Beam-Joint using different values of K during inference. During training $K = 6$ for both Beam-Joint models.

Dataset	Soft	Hard	Hard-Mono	Full-Joint
de-N	85.50	85.13	85.65	**85.81**
de-V	94.91	95.04	95.31	**95.52**

Conclusion

In this paper we showed a simple yet effective approximation of the joint attention-output distribution in sequence to sequence learning. Our joint model consistently provides higher accuracy without significant running time overheads in five translation and two morphological inflection tasks. An interesting direction for future work is to extend beam-joint to multi-head attention architectures as in (Vaswani et al., 2017; Xu Chen, 2018).

Acknowledgements We thank NVIDIA Corporation for supporting this research by the donation of Titan X GPU.

---

[4] https://github.com/roeeaharoni/morphological-reinflection

[5] Our numbers are lower than earlier reported because ours use a single model whereas (Aharoni and Goldberg, 2017) and others report from an ensemble of five models.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2004–2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The iwslt 2015 evaluation campaign. In *IWSLT 2015, International Workshop on Spoken Language Translation*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *ACL*.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. *CoRR*, abs/1711.04231.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Philipp Koehn. 2017. Neural machine translation. *CoRR*, abs/1709.07809.

Jeffrey Ling and Alexander M. Rush. 2017. Coarse-to-fine attention models for document summarization. In *Proceedings of the Workshop on New Frontiers in Summarization, NFiS at EMNLP 2017, Copenhagen, Denmark*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *EMNLP*.

André F. T. Martins and Ramón Fernández Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In *LREC*.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In *NIPS*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *WMT*.

Shiv Shankar and Sunita Sarawagi. 2018. Labeled memory networks for online model adaptation. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*.

Mia et al Xu Chen. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *EMNLP*, pages 1307–1316.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines. *CoRR*, abs/1505.00521.

Joint Learning for Emotion Classification and Emotion Cause Detection

Ying Chen[1], Wenjun Hou[1], Xiyao Cheng[1], Shoushan Li[2]
[1] College of Information and Electrical Engineering, China Agricultural University, China
[2] Natural Language Processing Lab, Soochow University, China
`{chenying,houwenjun,chengxiyao}@cau.edu.cn`
`lishoushan@suda.edu.cn`

Abstract

We present a neural network-based joint approach for emotion classification and emotion cause detection, which attempts to capture mutual benefits across the two sub-tasks of emotion analysis. Considering that emotion classification and emotion cause detection need different kinds of features (affective and event-based separately), we propose a joint encoder which uses a unified framework to extract features for both sub-tasks and a joint model trainer which simultaneously learns two models for the two sub-tasks separately. Our experiments on Chinese microblogs show that the joint approach is very promising.

1 Introduction

The analysis of emotions in texts is an important task in NLP. Traditional studies treat this task as a pipeline of two separated sub-tasks: emotion classification and emotion cause detection. The former identifies the category of an emotion and the latter detects the cause of an emotion. This separated framework makes each sub-task more flexible to deal with, but it neglects the relevance between the two sub-tasks. In this paper, we explore joint approaches which can capture mutual benefits across the relevant two sub-tasks. To the best of our knowledge, this work is the first attempt to incorporate both emotion classification and emotion cause detection into a unified framework.

Although emotion classification relies on affective features and emotion cause detection needs event-based features, we propose a joint encoder which uses a unified framework to extract features for both emotion classification instances and emotion cause detection instances. Then, we propose a joint model trainer which simultaneously learns two models for the two sub-tasks separately. The experiments on Chinese microblogs show that our joint approach can effectively learn models for both sub-tasks.

2 Our Approach

2.1 Corpus

In this paper, we use the human-labeled emotion corpus provided by Cheng *et al.* (2017) as our experimental data (namely Cheng emotion corpus). To better explain our work, we adopt twitter's terminology used in Cheng *et al.* (2017). Cheng emotion corpus can be considered as a collection of subtweets. For each emotion in a subtweet, all emotion keywords expressing the emotion are selected, and then the class and the cause of the emotion are annotated. The emotion categorization used in Huang et al. (2016) is adopted, which includes four basic emotions (i.e., joy, angry, sad and fearful) and three complex emotions (i.e., positive, neutral and negative). E.g. in the following example, the class of the emotion keyword ("😭") is *sad*, and the cause of the emotion is "only I was at home again".

Chinese : 兴冲冲勒跑回家~~发现又是我一个人再家。。😭 早知道就去蹭饭了

English Translation: I was very excited to run back home. I found that only I was at home again.😭 If I knew it earlier, I would have a meal for free.

Figure 1: An example of a subtweet

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 646–651
Brussels, Belgium, October 31 - November 4, 2018.

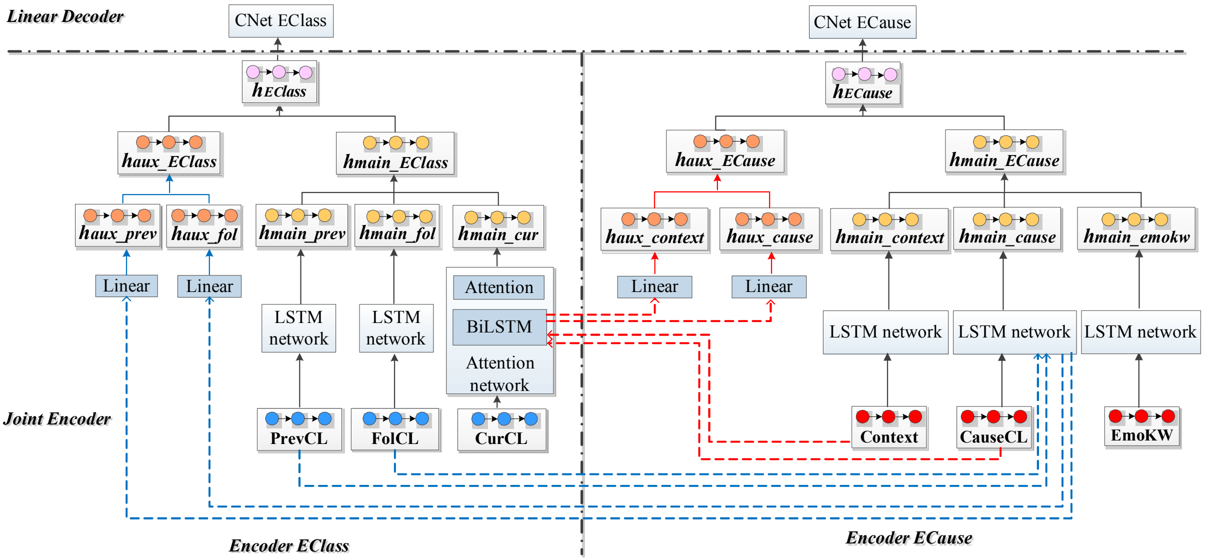

Figure 2: The framework of our joint approach

2.2 Problem Formulation

In this paper, both the emotion classification sub-task (namely *EClass*) and the emotion cause detection sub-task (namely *ECause*) are clause-level. Given an instance which is a clause in a subtweet, *EClass* assigns one of seven labels (i.e. six emotion classes and label ‘*non-emotion*’ which indicates the absence of an emotion) to the instance. Notice, because of the extremely low percentage of emotion ‘*fearful*’ (~0.6% in §3.1 Table 1), we ignore this emotion class in *EClass*. Given an instance which is a pair of <an emotion keyword, a clause in the subtweet>, *ECause* assigns a binary label to the instance to indicates the presence of a causal relation. Moreover, the clause-level *EClass* can effectively avoid the problem of multiple emotions (Li et al., 2015) because clauses are a kind of fine-grained texts.

Furthermore, the input text of an *EClass* instance contains three sequences of words: the previous clause (i.e. *PrevCL*), the current clause (i.e. *CurCL*), and the following clause (i.e. *FolCL*). The previous clause and the following clause provide contextual information for the current clause. The input text of an *ECause* instance also has three sequences of words: the emotion keyword (i.e. *EmoKW*), the current clause (i.e. *CauseCL*) and the context between *EmoKW* and *CauseCL*. The emotion keyword serves as an anchor, the current clause gives the description of an event which may cause the emotion, and the context provides complemental information for the event. Moreover, each word is represented with a vector from our word embedding model which is trained with word2vec[1] and the tweet corpus of Cheng *et al*. (2017).

2.3 The Joint Approach

As shown in Fig. 2, there are two parts in our joint approach which is based on neural networks: a joint encoder (the lower part) which extracts feature representations for both *EClass* instances and *ECause* instances, and a linear decoder (the upper part) which assigns labels to instances according to their representations.

Neural Networks

In the joint encoder, there are two neural networks (the attention network and the LSTM network), and each neural network is composed of several layers: bidirectional LSTM (i.e. BiLSTM) and attention. The BiLSTM layer focuses on the extraction of sequence features, and the attention layer focuses on the learning of word importance (weights). Because of the feature sparse problem in our small-scaled experimental data, the attention network often cannot effectively extract features to represent an event (see §3.2). Thus, in our joint encoder, we use the attention network to extract affective features (e.g. “😱” in Fig. 1) and the LSTM network to extract event-based features (e.g. “I found that only I was at home again” in Fig. 1).

---

[1] https://code.google.com/p/word2vec/

The attention network: we implement the attention network used in Felbo *et al.* (2017), which includes two layers: a BiLSTM layer which extracts a sequence feature for each input word, and an attention layer which represents the input sequence using weighted words.

The LSTM network: the network uses a BiLSTM layer to capture a sequence feature for each input word, and then uses the average of those features as the representation of the input sequence.

In the linear decoder, there are two classification networks (CNet *EClass* and CNet *ECause*) for *EClass* and *ECause* separately. Each classification network uses a linear layer to build a probabilistic classification model.

The Joint Encoder

As shown in Fig. 2, there are two sub-encoders in our joint encoder: Encoder *EClass* (the left part) which provides a representation for an *EClass* instance, and Encoder *ECause* (the right part) which extracts a representation for an *ECause* instance. Given an instance, one sub-encoder extracts a main representation (through the black lines in Fig.2) and the other sub-encoder provides an auxiliary representation (through the blue or red lines in Fig.2). Then, the concatenation of the two representations serves as the final representation for the instance (i.e. h_{EClass} or h_{ECause} in Fig.2). In order to deal with the case that a main representation may be overwhelmed by its corresponding auxiliary representation, linear layers are used to reduce the dimensions of auxiliary representations. Moreover, there are three sequences of words either in the input text of an *EClass* instance or in the input text of an *ECause* instance. In order to effectively use these input sequences, a multi-channel structure is chosen, which encodes the input sequences one by one.

Encoder *EClass*: given the three sequences of words in an *EClass* instance (*PrevCL*, *CurCL* and *FolCL*), the attention network is applied to *CurCL* to extract an affective representation, and the LSTM network is applied to *PrevCL* and *FolCL* separately to extract two event-based representations. Then, the concatenation of the three representations is used as the main representation (i.e. $h_{\text{main\_EClass}}$). Furthermore, in order to extract more contextual information, the LSTM network of Encoder *ECause* is applied to *PrevCL* and *FolCL* (through the blue lines in Fig. 2) to extract the auxiliary representation (i.e. $h_{\text{aux\_EClass}}$), which provides another event-based view for our emotion classification.

Encoder *ECause*: in order to separately deal with the three sequences of words (*EmoKW*, *CauseCL* and *Context*) in an *ECause* instance, the LSTM network is applied to each input sequence and then the concatenation of the three representations is used as the main representation (i.e. $h_{\text{main\_ECause}}$). Furthermore, for each input sequence (*CauseCL* or *Context*), the BiLSTM layer in the attention network is used to extract more event-based features (through the red lines in Fig. 2), and those features serve as an auxiliary representation (i.e. h_{aux_ECause}) which provides another event-based view for our emotion cause detection.

The Joint Model Trainer

During training, two models (*JMEClass* and *JMECause*) are learned simultaneously for the two sub-tasks (*EClass* and *ECause*) separately. Model *JMEClass* contains Encoder *EClass* and CNet *EClass*, and Model *JMECause* contains Encoder *ECause* and CNet *ECause*. Although each model uses auxiliary representations from the other model, but the learning of the model focuses on its own parameters. In other words, gradient calculation is disabled along the dashed lines in Fig. 2.

In each episode, the batch of input data is composed of two sets of instances: *EClass* sub-batch containing only *EClass* instances and *ECause* sub-batch containing only *ECause* instances. Given the batch of data, the parameters of each model are updated according its corresponding loss function. E.g., Model *JMEClass* uses only the *EClass* sub-batch, and its loss function is the mean squared errors of the instances in the sub-batch. In our joint model trainer, the two models are optimized using their own loss functions as pipeline model training does, but they use up-to-date auxiliary representations from each other to help optimization.

3 Experiments

3.1 Experimental Setup

In Cheng emotion corpus, there are ~3,000 subtweets, ~11,000 instances for *EClass*, and ~13,000 instances for *ECause*. Moreover, Table 1 lists the class distribution in Cheng emotion corpus for *EClass*. All experiments in this paper are trained and tested by 5-fold cross-validation on Cheng emotion corpus, and all the results reported are the average ones of 5-fold cross-validation performances. We use the precision, recall and F-score as our evaluation metrics. However, because of the high percentage of label '*non-emotion*' in *EClass* (see Table 1) and label '*0*' in *ECause*, similar to previous work (Li et al. 2015; Felbo et al., 2017; Cheng *et al*., 2017; Gui *et al*., 2017), we report only the evaluation metrics of the six emotion classes for *EClass* and the evaluation metrics of label '*1*' for *ECause*.

Class	%	Class	%
Joy	11.3	Angry	3.5
Sad	2.6	Fearful*	0.6
Positive	8.2	Neutral	4.4
Negative	9.9	Non-emotion	59.5

Table 1: The class distribution in Cheng emotion corpus for *EClass*. (*: ignored)

During our joint training process, the dimension of the word embeddings is 20; the output dimension of the BiLSTM layer used in both the LSTM network and the attention network is 128; the output dimension of the linear network is 8; the batch size is 32.

The two models (*JMEClass* and *JMECause*) which are learned by our joint approach are compared with several pipeline models which are learned in a pipeline manner (i.e. either for *EClass* or for *ECause*) using one of the following state-of-the-art encoders.

- **ATT:** the attention network in Fig.2 .
- **LSTM:** the LSTM network in Fig.2.
- **ATT+LSTM:** an hybrid encoder for emotion classification, which applies *ATT* to *CurCL* and *LSTM* to *PrevCL* and *FolCL*.
- **ConvMSMemnet:** the encoder proposed by Gui et al. (2017) for emotion cause detection, which applies a convolutional multiple-slot deep memory network to *CauseCL*.

3.2 Method Analysis

Table 2 shows the performances of different emotion classification models, where "Sequence" lists the sequences of input words used by each model and each metric is the average performances of six emotion classes. Moreover, Table 3 lists the detailed performances of each emotion class in Model *JMEClass*.

Encoder	Sequence	Prec	Rec	F1
LSTM	*CurCL*	65.5	53.0	58.2
ATT	*CurCL*	67.6	56.5	61.0
	all	67.5	56.7	61.2
ATT+LSTM	all	67.0	57.8	61.7
JMEClass	all	***67.7***	***58.5***	***62.4***

Table 2: The performances of emotion classification models. (all: *PrevCL*, *CurCL* plus *FolCL*)

Class	Prec	Rec	F1
Joy	85.5	83.6	84.5
Angry	62.8	45.0	52.4
Sad	72.6	72.9	72.8
Positive	63.0	54.7	58.6
Neutral	62.5	41.2	49.7
Negative	59.9	53.9	56.7

Table 3: The performances of the six emotions in JMEClass

In Table 2, Model *ATT* + *CurCL* out-performs *LSTM* + *CurCL* by 2.8% in F-scores, where *ATT* is a state-of-the-art encoder for emotion classification (Felbo et al., 2017). The significant performance improvement means that *ATT* can effectively extract affective features in *CurCL*. In fact, the emotion classification on Chinese microblogs can rely much on emotion keywords occurring in *CurCL*. E.g. ~50% emotional instances in our experimental data contains emoticons (e.g. "😭" in Fig. 1) in *CurCL* and those emoticons themselves are strong emotion indicators. Secondly, when different kinds of contextual information are incorporated to Model *ATT* + *CurCL*, different performance improvements obtain (0.2% for *ATT* + *all* and 0.7% for *ATT+LSTM* in F-scores). This indicates that for the emotion classification, the event-based features extracted by *LSTM* are more helpful than the affective features extracted by *ATT*, because contexts often provide the cause event of an emotion. E.g. in Fig. 1, the previous clause of "😭" contains its cause "only I was at home again". Finally, taking the advantage of the event-based features extracted by *JMECause*, *JMEClass* out-performs the best pipeline model (*ATT+LSTM*) by 0.7% in F-scores. This shows that it is important for the emotion classification to have an encoder which can effectively extract event-based features from contexts.

In Table 3, the performance of a basic emotion (i.e., joy, angry or sad) is often better than the one of a complex emotion (i.e., positive, neutral or negative). However, in Table 1, the data size of a basic emotion is often smaller than the one of a complex emotion. This indicates that difference in performance is likely linked to differences in the emotional contents of labels rather than differences in data sizes. E.g. the complex emotion 'negative' (i.e. a collection of complex emotions with negativity, such as 'hate', 'anxious', and so on) is more diverse than the basic emotion 'sad', and this diversity in emotional contents brings more challenges to the detection of this complex emotion. Furthermore, even if both 'sad' and 'angry' are basic emotions and have similar data sizes in our experimental data, it seems much easier to detect 'sad' instances than to detect 'angry' instances. This is maybe because 'angry' is caused by more various events and it is more difficult to capture and utilize those cause events. Thus, it is necessary for the emotion classification to have an encoder which can extract the event-based information of emotion cause from texts.

Encoder	Sequence	Prec	Rec	F1
ConvMS-Memnet	*CauseCL*	34.3	***77.5***	47.5
ATT	all	55.4	60.9	58.0
LSTM	all	***55.6***	61.3	58.3
JMECause	all	53.1	66.7	***59.1***

Table 4: The performances of emotion cause detection models. (all: *EmoKW*, *CauseCL* plus *Context*)

Table 4 shows the performances of different emotion cause detection models, where "Sequence" lists the sequences of input words used by each model. In Table 4, *JMECause* outperforms the best pipeline model (*LSTM*) by 0.8% in F-scores. The *LSTM* encoder is a state-of-the-art approach used for emotion cause detection (Cheng et al., 2017). Furthermore, the performance improvement of *JMECause* is from the significant increasing in recalls (5.4%). This indicates that more emotion causes are correctly detected when the event-based features extracted by Model *JMEClass* are incorporated. Moreover, among all models, the two models (*ATT* and *LSTM*) achieve relatively high precision and relatively low recall, and ConvMS-Memnet obtains the lowest precision and highest recall. This means that both *ATT* and *LSTM* suffer from the feature coverage problem because some useful features cannot be extracted through their encoders, and ConvMS-Memn suffers from the feature quality problem maybe because its encoder cannot handle the informal writing style used in Chinese microblogs.

4 Related Work

In recent years, intensive studies have explored supervised machine learning approaches using various types of features for different-level emotion classification, such as document level (Alm et al. 2005; Li et al. 2014; Huang et al. 2016), sentence level or short text level (Tokushisa et al. 2008; Bhowmick et al. 2009; Xu et al. 2012; Wen and Wan, 2014; Li et al. 2015; Felbo et al., 2017), and so on. Moreover, since both emotion and sentiment belong to affective feeling, some studies have explored the join learning of sentiment classification and emotion classification (Gao et al., 2013; Wang et al., 2015).

In the other hand, most of previous emotion cause detection studies is clause-based, which examine whether a clause around a given emotion keyword is a cause or not. Moreover, these studies (Chen *et al*., 2010; Xu *et al*., 2017; Ghazi *et al*., 2015; Gui *et al*., 2017; Cheng *et al*., 2017) focus on how to extract two kinds of features for supervised model learning: explicit expression patterns (e.g. "to cause", "for"), and implicit features which can reflect the causal relation.

5 Conclusion

In this paper, we focus on a joint learning approach to emotion classification and emotion cause detection on Chinese microblogs, and the experiments show such a joint approach is very promising.

Acknowledgments

This research work was partially supported by four the National Science Foundation of China (No. 61503386).

References

Cecilia Ovesdotter Alm, Dan Roth and Richard Sproat. 2005. Emotions from Text: Machine Learning for Text-based Emotion Prediction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 579–586.

Plaban Kr. Bhowmick, Anupam Basu, Pabitra Mitra and Abhishek Prasad. 2009. Multi-label Text Classification Approach for Sentence Level News Emotion Analysis. In *Proceedings of the International Conference on Pattern Recognition and Machine Intelligence*, pages 261–266.

Ying Chen, Sophia Yat Mei Lee, Shoushan Li and Chu-Ren Huang. 2010. Emotion cause detection with linguistic constructions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 179–187.

Xiyao Cheng, Ying Chen, Bixiao Cheng, Shoushan Li and Guodong Zhou. 2017. An Emotion Cause Corpus for Chinese Microblogs with Multiple-User Structures. In *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, Vol. 17, No. 1, Article 6.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625.

Wei Gao, Shoushan Li, Sophia Yat Mei Lee, Guodong Zhou and Chu-Ren Huang. 2013. Joint Learning on Sentiment and Emotion Classification. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1505-1508.

Diman Ghazi, Diana Inkpen and Stan Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In *Proceedings of the 16th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 152–165.

Lin Gui, Jiannan Hu, Yulan He, Ruifeng Xu, Qin Lu and Jiachen Du. 2017. A Question Answering Approach to Emotion Cause Extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1593–1602.

Lei Huang, Shoushan Li and Guodong Zhou. 2016. Emotion Corpus Construction on Microblog Text. *The series Lecture Notes in Computer Science*, Volume 9332, pages 204-212.

Chengxin Li, Huimin Wu and Qin Jin. 2014. Emotion Classification of Chinese Miroblog Text via Fusion of Bow and Evector Feature Representations. In *Proceedings of the 3rd CCF Conference on Natural Language Processing & Chinese Computing*, pages 217-228.

Shoushan Li, Lei Huang, Rong Wang and Guodong Zhou. 2015. Sentence-level Emotion Classification with Label and Context Dependence. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1045–1053.

Ryoko Tokuhisa, Kentaro Inui and Yuji Matsumoto. 2008. Emotion Classification Using Massive Examples Extracted from the Web. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 881-888.

Jun Xu, Ruifeng Xu, Qin Lu and Xiaolong Wang. 2012. Coarse-to-fine Sentence-level Emotion Classification based on the Intra-sentence Features and Sentential Context. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 2455-2458.

Ruifeng Xu, Jiannan Hu, Qin Lu, Dongyin Wu and Lin Gui. 2017. An Ensemble Approach for Emotion Cause Detection with Event Extraction and Multi-Kernel SVMs. *TSINGHUA SCIENCE AND TECHNOLOGY*, Volume 22, Number 6.

Rong Wang, Shoushan Li, Guodong Zhou and Hanxiao Shi. 2015. Joint Sentiment and Emotion Classification with Integer Linear Programming. In *Liu A., Ishikawa Y., Qian T., Nutanong S., Cheema M. (eds) Database Systems for Advanced Applications, Lecture Notes in Computer Science*, Vol 9052.

Shiyang Wen and Xiaojun Wan. 2014. Emotion Classification in Microblog Texts Using Class Sequential Rules. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 187-193.

Exploring Optimism and Pessimism in Twitter Using Deep Learning

Cornelia Caragea[1], **Liviu P. Dinu**[2], **and Bogdan Dumitru**[2]
[1]Computer Science, Kansas State University
Manhattan, Kansas
[2]Faculty of Mathematics and Computer Science, University of Bucharest
Bucharest, Romania
ccaragea@ksu.edu, ldinu@fmi.unibuc.ro, bogdan27182@gmail.com

Abstract

Identifying optimistic and pessimistic viewpoints and users from Twitter is useful for providing better social support to those who need such support, and for minimizing the negative influence among users and maximizing the spread of positive attitudes and ideas. In this paper, we explore a range of deep learning models to predict optimism and pessimism in Twitter at both tweet and user level and show that these models substantially outperform traditional machine learning classifiers used in prior work. In addition, we show evidence that a sentiment classifier would not be sufficient for accurately predicting optimism and pessimism in Twitter. Last, we study the verb tense usage as well as the presence of polarity words in optimistic and pessimistic tweets.

1 Introduction

"You know, ever since we were little, I would get this feeling like... Like I'm floating outside of my body, looking down at myself... And I hate what I see... How I'm acting, the way I sound. And I don't know how to change it. And I'm so scared... That the feeling is never gonna go away."

The Edge of Seventeen[1]

Much has been written about optimism and pessimism in psychological studies for decades (Scheier and Carver, 1992). These feelings are affected by one's personality from an early age (as pinpointed above) and can strongly impact one's psychological and physical health. For example, pessimism and negative attitudes impact negatively one's mental health, can induce suicidal thoughts, and affect negatively not only the person in question, but also their family and friends (Peterson and Bossio, 2001; Achat et al., 2000; Scheier et al., 2001). On the other hand, optimism reduces stress and promotes better physical health and overall well-being (Carver et al., 2010).

Despite that optimism and pessimism are under the scrutiny of many researchers (Rasmussen et al., 2009; Kumar et al., 2017), large scale analyses that explore optimism and pessimism in social media have just started to emerge (Ruan et al., 2016). However, Ruan et al. (2016) focused on identifying optimism and pessimism in Twitter using a simple "bag of words" representation with no emphasis on incorporating semantic information hidden in text. Often, a deeper understanding of the text that accounts for textual semantic similarities and the writer's intention are required in order to correctly detect the characteristics of optimistic and pessimistic feelings in tweets. Towards this end, our contributions in this paper are as follows. First, we focus on the question: "Would a deep learning approach help to discover these characteristics better than traditional machine learning classifiers used in prior work?" To our knowledge, we take the first step towards exploring the performance of deep learning models for optimism/pessimism prediction in Twitter and identify the most promising deep learning models for this task. Identifying optimism and pessimism in Twitter has many applications including identifying suicidal/depressive people and providing better social support (e.g., emotional/empathetic support) that can improve people's moods and attitudes (Yan and Tan, 2014; Biyani et al., 2014; Khanpour et al., 2018, 2017; Qiu et al., 2011).

Second, since it may seem intuitive that a positive sentiment is associated with optimism and a negative sentiment with pessimism, we address the question: "Would a sentiment classifier be sufficient to correctly identify optimism and pessimism in social media?" Figure 1 shows evidence that a sentiment tool would not suffice on accurately

---

[1] https://www.imdb.com/title/tt1878870/

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 652–658
Brussels, Belgium, October 31 - November 4, 2018.

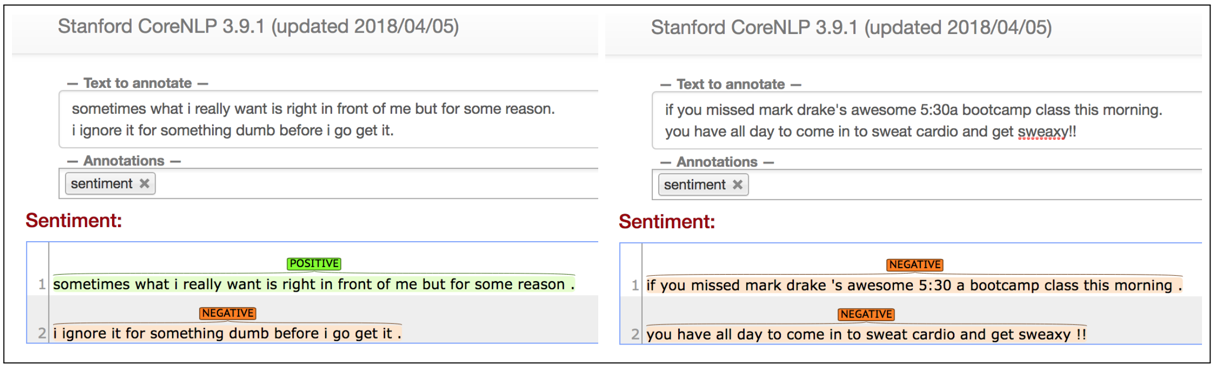

sometimes what i really want is right in front of me but for some reason.
i ignore it for something dumb before i go get it.

1 sometimes what i really want is right in front of me but for some reason .
2 i ignore it for something dumb before i go get it .

if you missed mark drake's awesome 5:30a bootcamp class this morning.
you have all day to come in to sweat cardio and get sweaxy!!

1 if you missed mark drake 's awesome 5:30 a bootcamp class this morning .
2 you have all day to come in to sweat cardio and get sweaxy !!

Figure 1: Sentiment prediction on tweets with pessimist (left) and optimist (right) connotations.

predicting tweets with pessimistic and optimistic connotations (left and right side of the figure, respectively). We answer the above question by investigating a spectrum of sentiment analysis tools and datasets for optimism/pessimism prediction.

Third, we perform a linguistic analysis, first of its kind, and study the usage of verb tenses (past, present, future) in optimistic and pessimistic tweets, as well as the presence of polarity words associated with both types of tweets.

2 Datasets

In this section, we first describe the optimism/pessimism Twitter dataset and then present two datasets used for sentiment analysis.

The Optimism/Pessimism Twitter dataset (**OPT**) was made available to us by Ruan et al. (2016). The total number of tweets in the dataset is 7,475. These tweets were sampled from data corresponding to 500 optimist and 500 pessimist users, and were manually annotated using Amazon Mechanical Turk. Precisely, each tweet was manually annotated by five independent annotators using a score between -3 (very pessimistic) and 3 (very optimistic). For our evaluation, we consider two different thresholds (0 and 1/-1) on the above score and create two settings as follows. In the first evaluation setting, a tweet is labeled as pessimistic if its score is smaller than or equal to 0, and optimistic, otherwise. In the second evaluation setting, a tweet is labeled as pessimistic if its score is smaller than or equal to -1, and optimistic if its score is greater than or equal to 1. A summary of this dataset is given in Table 1.

	Threshold 0	Threshold 1/-1
Number of tweets	7,475 (O: 4,679)	3,847 (O: 2,507)
Number of users	1,000	1,000

Table 1: Dataset statistics. 'O' stands for optimistic tweets.

The Stanford Sentiment Treebank (**SST**) (Socher et al., 2013) is a corpus for sentiment analysis that capture complex linguistic patterns. This dataset[2] is based on a dataset originally introduced by Pang and Lee (2005) and consists of 10,662 sentences from movie reviews downloaded from rottentomatoes.com. From these sentences, 215,154 phrases were extracted using the Stanford Parser (Klein and Manning, 2003) and labeled using Amazon Mechanical Turk such that each phrase was annotated by 3 human judges.

The Twitter Sentiment Analysis (**TSA**) dataset,[3] available online for download, contains 1,578,627 tweets that are classified as 1 for positive sentiment and 0 for negative sentiment.

3 Experiments and Results

In experiments, we explore several deep learning models for optimism/pessimism prediction. The general training strategy is as follows: sentence embeddings are fed into a sentence encoder to obtain the sentence representation. The classifier consists of three fully connected layers topped by a softmax layer. Dropout was applied to the first layer only. We used several encoders as follows, based on: (1) Bidirectional Long Short Term Memory networks (BiLSTMs), which are a special type of Recurrent Neural Networks (RNNs) (Hochreiter and Schmidhuber, 1997); (2) Convolutional Neural Networks (CNNs), which consist of convolution and max pooling (Kim, 2014); and (3) Stacked Gated RNNs (Chung et al., 2015).

We used SGD optimizer (Goodfellow et al., 2016) with a learning rate of 0.1 and no weight decay. At every tenth epoch we decreased the learn-

---

[2] https://nlp.stanford.edu/sentiment/

[3] http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/

Threshold	Tweet Level		User Level	
	0	$1/-1$	0	$1/-1$
BiLSTM	79.65	87.24	**76.65**	90.52
GRUStack	**80.19**	87.76	76.38	**92.24**
CNN	77.78	**90.32**	73.55	91.68
NB	74.20	84.10	71.30	80.10
SVM	67.80	83.30	64.70	81.80

Table 2: Accuracy of deep learning vs. traditional classifiers on the **OPT** dataset (shown as percentage).

ing rate by half. We used mini-batches of 40 samples. Dropout rate was set to 0.5 and the classifier's last three layers have 300, 200, and 100 neurons. We used GloVe vectors (Pennington et al., 2014) trained on Common Crawl 840B[4] with 300 dimensions as fixed word embeddings.

For sentence embedding, after a cleanup process, sentences were transformed into a list of words, then words were replaced with word embeddings (GloVe) and padding was used to align batch sentences to the same size.

3.1 Optimism/Pessimism Prediction

In our first experiment, we explore the above deep learning models on the **OPT** dataset and compare their performance with that of two traditional machine learning classifiers, Naïve Bayes (NB) and Support Vector Machines (SVM), which were used in the previous work for this task by Ruan et al. (2016). In this experiment, the **OPT** dataset is split in train-dev-test as 80-10-10(%), respectively. We repeated each experiment 5 times and averaged the results. Our deep learning implementation is built on top of TensorFlow (Abadi et al., 2015). For NB and SVM, we used their implementation available in scikit-learn (Pedregosa et al., 2011). Table 2 shows the accuracy of all these models at tweet and user level for the two thresholds 0 and 1/-1 (as discussed in Section 2).

We can see that overall, the deep learning models achieve a much higher performance compared with the work by Ruan et al. (2016), i.e., the NB and SVM classifiers on "bag of words," for both tweet and user level with both thresholds, yielding an improvement in performance between 5%-10%. For example, at tweet level and 1/-1 threshold, CNN yields an accuracy of 90.32% as compared with NB, which achieves an accuracy of 84.10%. At user level and 1/-1 threshold, GRUStack yields an accuracy of 92.24%, as compared with 81.80% achieved by SVM. Not surprising, for both tweet and user level, when we use a threshold of 0, the performance of all models is smaller compared with that of models obtained when we use a 1/-1 threshold. Intuitively, this is true since most of the tweets with a human-annotated score between -1 and 1 are in the "gray" area that is harder to classify. Note that Ruan et al. (2016) considered the tweets with a score between -1 and 1 as being neutral.

3.2 Sentiment vs. Optimism/Pessimism

In our second experiment, we investigate the correlation between sentiment and optimism / pessimism, and argue that sentiment analyzers, that are trained to predict sentiment (Liu, 2012; Pang and Lee, 2008), fail to detect optimism and pessimism. Specifically, we train several sentiment classifiers on the large **SST** and **TSA** sentiment datasets (described in Section 2) and evaluate the performance of these classifiers on the optimism/pessimism categories from the **OPT** dataset.

Model	Train	Dev	Test	Acc%
LSTM	SST	SST	OPT	63.20
BiLSTM	SST	SST	OPT	63.60
CNN	SST	SST	OPT	59.60
CNN	TSA	TSA	OPT	67.60
RNN(char)	TSA	OPT	OPT	55.20
GRUStack	OPT	OPT	OPT	**80.19**

Table 3: Performance of sentiment classifiers on **OPT**.

Table 3 shows the performance of several deep learning models trained on either **SST** or **TSA** datasets and evaluated on the **OPT** dataset. Note that the Dev set was used for model selection. As can be seen from the table, the models trained on the sentiment datasets perform poorly on the optimism/pessimism dataset. For example, there is a drop in performance from 80.19% to 67.60% when training on **TSA** (with an even larger decrease when we train on **SST**).

The **SST/TSA** sentiment classifiers are trained to predict the sentiment as negative, neutral, or positive. To calculate the accuracy in Table 3, an optimistic tweet predicted as positive by the sentiment classifier counts as a correct prediction, whereas an optimistic tweet predicted as either neutral or negative by the sentiment classifier counts as an incorrect prediction (similarly for pessimistic tweets). This analysis is done at tweet level for the threshold of 0.

---

[4] https://nlp.stanford.edu/projects/glove/

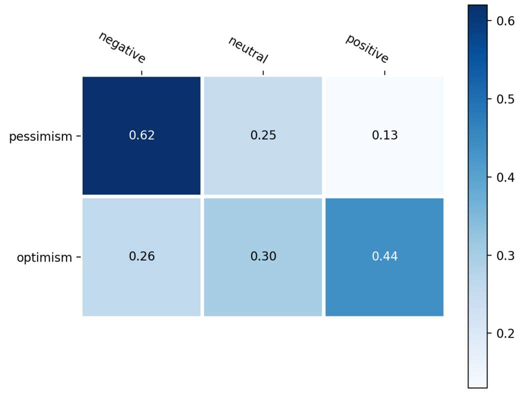

Figure 2: Optimism and pessimism classified as positive, negative, and neutral sentiment.

Figure 2 shows the normalized number of examples from optimism and pessimism categories classified as positive, negative and neutral, using the CNN model trained on **TSA**. Precisely, we show how many tweets from the set of optimistic (or pessimistic) tweets in the **OPT** dataset are predicted as negative, neutral or positive by the **TSA** sentiment classifier. The numbers on each row sum up to 1. As we can see from the figure, although pessimism is more correlated with a negative sentiment, 13% of the pessimistic tweets are classified as positive (with similar results on the optimism category).

3.3 Linguistic Analysis

In this section, we perform a linguistic analysis and study the usage of verb tenses in optimistic and pessimistic tweets, as well as the presence of polarity words associated with both types of tweets. This analysis is done at tweet level with 1/-1 threshold. The reason for using the 1/-1 threshold is that we wanted to study the usage of verb tenses and polarity words in tweets that are clear optimistic or clear pessimistic (far from the decision boundary).

3.3.1 Verb Tenses in Optimism/Pessimism

For this analysis, we used the part of speech tagger spaCy[5] and assigned the verbs to their corresponding tenses according to the Penn Treebank Project; that is, the tags VBD and VBN correspond to past tense, VBG, VBZ , VBP correspond to present tense, whereas an MD tag followed by VB (possibly with a negation between them) corresponds to the future tense.

As mentioned, a tweet was considered optimist if its manually annotated score was above 1 and pessimist if the score was below -1. The numbers of tweets with past, present, and future tenses in the optimistic category are: 1,474, 7,444, and 561, respectively, whereas these numbers in the pessimistic category are: 1,276, 5,311, and 325, respectively.

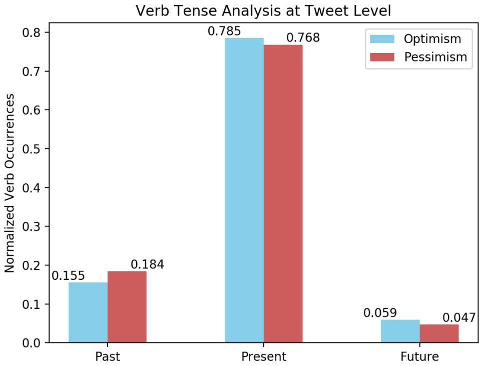

Figure 3: Verb tenses in optimist/pessimist tweets.

Figure 3 shows the normalized verb occurrences at past, present and future tenses in optimistic and pessimistic tweets. As can be seen from the figure, the present tense is the most prevalent for both categories, although there are more present tense verbs in the optimistic category compared with the pessimistic one. We can also observe that more past tense verbs occur in the pessimistic category and less future tense verbs in the pessimistic one.

While there are some common verbs such as "be," "have," and "do," that appear most frequently in both optimistic and pessimistic categories at all three tenses, there are some verbs that are more specific to one category than the other. Examples of such verbs and their frequencies from both categories at the present tense are shown in Table 4. As we can see, optimism is characterized more by verbs with a positive connotation.

[optimism]	*[pessimism]*
('be', 1662) ('have', 738) ('do', 316)	('be', 1198) ('have', 549) ('do', 342)
('love', 266) ('thank', 105) ('look', 98) ('want', 94) ('hope', 70)	('hate', 122) ('f–k', 88) ('kill', 35) ('try', 34) ('cry', 33)

Table 4: Verbs in tweets occurring at present tense.

---

[5] http://textanalysisonline.com/spacy-pos-tagging

3.3.2 Polarity Words in Optimism/Pessimism

Next, we analyze the association of polarity words from the positive and negative lexicons constructed by Hu and Liu (2004), in both tweet categories: optimism and pessimism. Instead of using the presence or absence of the words from the two lexicons in tweets, we calculated the cosine similarity between the word embeddings of the words in the two lexicons with the words in the tweets. If the similarity is above 0.8, then we consider the word from the corresponding lexicon to be present in the tweet (or synonym with a word in tweet). Using the cosine similarity between the word embeddings of words in lexicons with words in tweets captures not only the exact match between the words (a cosine similarity of 1 for exact match), but also incorporates the semantic information that exists in the text.

Although this word similarity computation relaxes the exact match/presence of a word in a tweet and aims at incorporating semantic similarity, a high similarity between antonyms may occur since word embeddings are known to not differentiate well between synonyms and antonyms, which tend to appear in similar contexts.

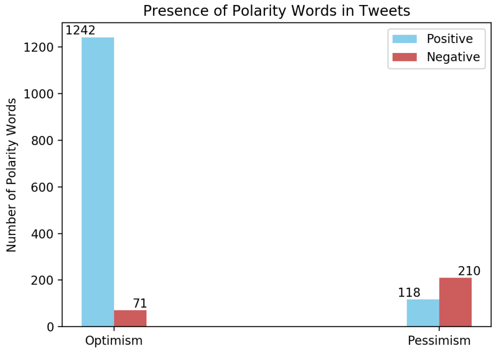

Figure 4: Polarity words in optimist/pessimist tweets.

Figure 4 shows the number of polarity words in optimistic and pessimistic tweets. As shown in the figure, more positive words appear in optimistic tweets compared with negative words (1,242 vs. 71), while there is not a substantial difference between the numbers of positive and negative words in pessimistic tweets (118 vs. 210).

Table 5 shows the top most frequent polarity words associated with optimism and pessimism. As we can see, words with a negative polarity (e.g., bad) although not very frequent, still appear in optimistic tweets. This supports our intuition that a sentiment model is not enough to accurately predict pessimism and optimism in Twitter.

Optimism	
[positive]	*[negative]*
('great', 289)	('bad', 13)
('loved', 279)	('worried', 11)
('wonderful', 155)	('lost', 6)
('glad', 76)	('scared', 5)
('kind', 45)	('terrible', 4)
('thrilled', 29)	('disappointed', 3)
Pessimism	
[positive]	*[negative]*
('great', 29)	('bad', 36)
('kind', 27)	('lost', 18)
('loved', 13)	('scared', 16)
('wonderful', 9)	('alone', 13)
('surprised', 8)	('terrible', 12)
('glad', 6)	('terrified', 12)

Table 5: Top frequent polarity words.

4 Concluding Remarks

In this paper, we explored deep learning models for optimism and pessimism prediction in Twitter and showed that these models substantially outperform traditional classifiers such as Naïve Bayes and Support Vector Machines. To our knowledge, this work is the first computational study that explores optimism and pessimism using deep learning. We also showed that a sentiment classifier would not be sufficient for accurately predicting optimism and pessimism. This topic is less explored despite its importance in many applications such as identifying suicidal/depressive people.

Interesting future directions are: understanding how one's age is correlated with optimism / pessimism; if one user is characterized by a mixture of topics, is that user optimist (pessimist) across all these topics? Thus, decomposing a user's textual data into topic and correlating this with optimism and pessimism may be interesting to explore; last, studying how optimism and pessimism are affected by sarcasm.

As we started our study with a pessimistic quote from the movie "The Edge of Seventeen," we end our study with a quote from the same movie, with a positive sentiment and full of optimism:

"Life's about taking risks. Don't be afraid to put yourself out there."

Acknowledgments

All authors contributed equally. LP Dinu was supported by UEFISCDI, project #53BG/2016. We thank our reviewers for their constructive comments and feedback.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Helen Achat, Ichiro Kawachi, Avron Spiro, Deborah A DeMolles, and David Sparrow. 2000. Optimism and depression as predictors of physical and mental health functioning: the normative aging study. *Annals of Behavioral Medicine*, 22(2):127–130.

Prakhar Biyani, Cornelia Caragea, Prasenjit Mitra, and John Yen. 2014. Identifying emotional and informational support in online health communities. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 827–836, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Charles S. Carver, Michael F. Scheier, and Suzanne C. Segerstrom. 2010. Optimism. *Clinical psychology review*, 30(7):879–889.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2067–2075.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Hamed Khanpour, Cornelia Caragea, and Prakhar Biyani. 2017. Identifying empathetic messages in online health communities. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017, Volume 2: Short Papers*, pages 246–251.

Hamed Khanpour, Cornelia Caragea, and Prakhar Biyani. 2018. Identifying emotional support in online health communities. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1746?1751.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Upendra Kumar, Vishal Singh Rana, Srinivas Pykl, and Amitava Das. 2017. "A pessimist sees the difficulty in every opportunity; an optimist sees the opportunity in every difficulty" - understanding the psycho-sociological influences to it. In *Proceedings of the 14th International Conference on Natural Language Processing*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Journal of Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Christopher Peterson and Lisa M Bossio. 2001. Optimism and physical well-being. *Optimism and pessimism: Implications for theory, research, and practice*, pages 127–145.

Baojun Qiu, Kang Zhao, Prasenjit Mitra, Dinghao Wu, Cornelia Caragea, John Yen, Greta E Greer, and

Kenneth Portier. 2011. Get online support, feel better–sentiment analysis and dynamics in an online cancer survivor community. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 274–281. IEEE.

Heather N Rasmussen, Michael F Scheier, and Joel B Greenhouse. 2009. Optimism and physical health: A meta-analytic review. *Annals of behavioral medicine*, 37(3):239–256.

Xianzhi Ruan, Steven Wilson, and Rada Mihalcea. 2016. Finding optimists and pessimists on twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 320–325, Berlin, Germany. Association for Computational Linguistics.

Michael F Scheier and Charles S Carver. 1992. Effects of optimism on psychological and physical well-being: Theoretical overview and empirical update. *Cognitive therapy and research*, 16(2):201–228.

Michael F Scheier, Charles S Carver, and Michael W Bridges. 2001. Optimism, pessimism, and psychological well-being. *Optimism and pessimism: Implications for theory, research, and practice*, 1:189–216.

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

Lu Yan and Yong Tan. 2014. Feeling blue? go online: an empirical study of social support among patients. *Information Systems Research*, 25(4):690–709.

Predicting News Headline Popularity with Syntactic and Semantic Knowledge Using Multi-Task Learning

Daniel Hardt
Copenhagen Business School
dh.msc@cbs.dk

Dirk Hovy
Bocconi University
dirk.hovy@unibocconi.it

Sotiris Lamprinidis
Copenhagen University
sot.lampr@gmail.com

Abstract

Newspapers need to attract readers with headlines, anticipating their readers' preferences. These preferences rely on topical, structural, and lexical factors. We model each of these factors in a multi-task GRU network to predict headline popularity. We find that pre-trained word embeddings provide significant improvements over untrained embeddings, as do the combination of two auxiliary tasks, news-section prediction and part-of-speech tagging. However, we also find that performance is very similar to that of a simple Logistic Regression model over character n-grams. Feature analysis reveals structural patterns of headline popularity, including the use of forward-looking deictic expressions and second person pronouns.

1 Introduction

The data generated from online news consumption constitutes a rich resource, which allows us to explore the relation between news content and user opinions and behaviors. In order to stay in business, newspapers need to pay attention to this information. For example, what headlines do users click on, and why? With the volume of news being consumed online today, there is great interest in addressing this problem algorithmically. We collaborate with a large Danish newspaper, who gave us access to several years' worth of headlines, and the number of clicks generated by readers.

We aggregate the viewing logs to classify headlines as popular or unpopular, and build models to predict those classifications. We use an expanded version of the dataset investigated by Hardt and Rambow (2017). That work found that bag-of-word models based on headlines did indeed have predictive value concerning viewing behavior, although models based on the article body were more accurate. As Hardt and Rambow noted, this is somewhat paradoxical: how can a model based on the article text be better at predicting clicks? After all, the choice to click on an article must be based on the headline alone – the article is only seen *after* the clicking decision is made. Hardt and Rambow speculate that "it is possible that the headline on its own gives readers a lot of semantic information which we are not capturing with our features, but which the whole article does provide. So human readers can "imagine" the article before they read it and implicitly base their behavior on their expectation." (Hardt and Rambow, 2017)

In other words, readers are able to *anticipate* the contents of an article in advance from a headline, because of the linguistic and world knowledge that they bring to bear when assessing the headline. If we can incorporate this "future" knowledge into a prediction model, we are likely to improve performance.

We test this hypothesis by defining ways to model aspects of the lexical, structural, and topical knowledge of human news readers:

- **Lexical – Word Embeddings:** we provide our models with pretrained word embeddings from large datasets. This models aspects of the rich lexical information and association that human readers bring to bear in reading a headline.
- **Structural – POS Tagging:** part of speech information is a basic component of structural linguistic knowledge, reflected in the structure of common headline templates such as "Can X do Y?" or "You will not believe what happened when X".
- **Topical – Section Prediction:** Each article is labeled with a section (sports, politics, etc). We include a task which predicts the section of a headline. This models the ability of a news reader to understand the most salient

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 659–664
Brussels, Belgium, October 31 - November 4, 2018.

and interesting topical material in a headline text.

We use a multi-task learning (MTL) setup (Caruana, 1993), which provides a natural framework to test the above hypotheses: one of the first uses of MTL was to include the outcome of future diagnostic tests into a prediction task (Caruana et al., 1996).

We explore the effect of pretrained word embeddings, and the effects of auxiliary tasks involving POS tagging and section prediction. We find that the combination of all of these factors results in substantial improvements over the baseline and the previous work, which used a single-task system. We also build logistic regression models, both for word and character n-grams. The word-based models have the advantage that the predictiveness of individual words can be examined.

While the word n-gram models have performance comparable to the baseline neural net, the character n-gram model has higher performance, competing with the best MTL result. This finding is in line with the results from Zhang et al. (2015).

Our results indicate that MTL can indeed provide the tools to implement prediction processes that involve expectations about the future. Given the successful integration of two auxiliary tasks, we see this as a promising starting point for future research. However, the performance parity with the character model underscores the fact that simple model architectures still have a place. Our findings, in line with other current work (Benton et al., 2017), shine light on the question of auxiliary task selection and their interaction, and highlight that MTL results should be rigorously tested.

A good predictive model is a powerful diagnostic tool for editors, allowing them to select proposed headlines. However, journalism is a creative production process, so detection is only part of the application. We also want to be able to give strategic advice to headline *writers*. To this end, we report an analysis of common n-gram features in the word-based logistic regression model, that provide some insights into successful headline patterns.

Contributions We explore an MTL architecture with two auxiliary tasks for headline popularity prediction. We show how aspects of lexical, structural, and topical knowledge are all relevant for headline popularity. The positive results reported here provide a fruitful basis for further development of MTL models for news data. We also analyze lexical features that are predictive of headline popularity.

Kære læser. Få fuld adgang til e-avisen og jyllands-posten.dk 1 kr. for første måned.

International gruppe læger: Vi gør alt for mange til syge

- Dansk studie afslører: Sådan påvirker elektrochok hjernen

Figure 1: Example of Jyllands-Posten headline as seen by audience

2 Data

News Data The present work is based on a significantly expanded and cleaned version of the dataset used by Hardt and Rambow (2017). This dataset includes Jyllands-Posten articles and logs. Jyllands-Posten is a major Danish newspaper (and became known to an international audience over the cartoon controversy). The data covers a period from July 2015 through July 2017. We removed any articles from before July 2015, when the viewing logs began, since these older articles have unreliable numbers of clicks. The resulting dataset consists of 82,532 articles and a total of 281,005,390 user views. We furthermore extracted the news section each article belongs to (sports, politics, etc.) from the URL.

We bin the articles by numbers of clicks into 2 bins, thus defining a classification task: is the article in the top 50% of clicks or not? The data is divided into 80% training, and 10% each development and test data.

Figure 1 shows the top headline on the Jyllands-Posten web site for August 27, 2018. Our data does not include information such as the position of a headline on the page, and possible associated graphical material.

Additional Data In addition to the news data from JP, we obtained a corpus of 100 million words of Danish text from the Society for Danish Language and Literature, or DSL (Jørg Asmussen, 2018). This corpus was collected from diverse sources over a period from 1990 to 2010. The corpus has been automatically annotated for part of speech and lemmatization, and we use this for our POS tagging task. We also downloaded the Danish Wikipedia, which consists of approximately 49 million words of Danish text. We use these corpora in conjunction with the JP article texts to induce pre-trained Danish word-embeddings.

Data Statement A. CURATION RATIONALE The dataset is collected by Jyllands-Posten as part of a general strategy to understand user behavior and preferences with respect to the news content on the site.

B. LANGUAGE VARIETY The data is Danish (da-DK).

C. SPEAKER DEMOGRAPHIC The text is produced by professional journalists.

D. ANNOTATOR DEMOGRAPHIC There is no manual annotation of the text.

E. SPEECH SITUATION The texts were produced from July 2015 until July 2017; the intended audience is Danish news consumers.

F. TEXT CHARACTERISTICS The text is standard, mainstream Danish journalism.

3 Models

Our task is to predict which articles get the most user clicks, based on the headline alone. We report results using logistic regression and a neural network, using MTL.

Logistic Regression We define the following features for logistic regression models:

1. *n-chars*: sequences of n characters, with n ranging from 2 to 6 in all experiments.

2. *word unigrams*: *tfidf* scores for all word unigrams

3. *word bigrams*: *tfidf* scores for all word bigrams

GRU Neural Network While the task is classification, which could be done with a feed-forward model, we want a sequential architecture, so that we can incorporate POS tagging as an auxiliary task, adding POS output at each time step.

Based on good results in recent work (Lee and Dernoncourt, 2016), (Liu et al., 2016), we choose a Recurrent Neural Network architecture and after a series of experiments on the training and validation set, we obtained the best results using GRU (Gated Recurrent Unit) units.

Each layer k consists of two sets of units, labeled fw and bw that process the sequence forwards and backwards respectively, so that information from the whole sequence is available on every timestep t. The two directions' activations are concatenated and fed to a fully-connected softmax (for multi-class classification) or sigmoid layer (for binary classification) to get the output probability y_t^k of the task associated with layer k. So that higher level tasks can benefit, we embed the output probabilities using the fully connected label embedding LE layer, a technique used on similar scenarios (Rønning et al., 2018). The embedded label gets concatenated with the GRU output to get the activation a_t^k that gets fed in the next layer, or the final fully connected prediction layer, as presented in figure 2.

In the sequential auxiliary task, i.e. POS tagging, this is done for every timestep, while for the classification tasks the prediction is made on the final timestep.

For regularization, we apply dropout on every layer of our network.

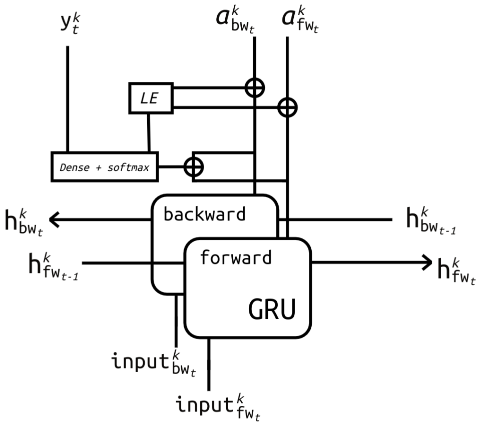

Figure 2: Representation of a single timestep t for a pair of forward-backward units on layer k where h_{t-1}^k is the previous hidden state.

Auxiliary Tasks In our setup, we use two auxiliary tasks:

1. *POS tagging*: we include POS tagging using the DSL dataset on the first recurrent layer of the GRU.

2. *Section prediction*: we include classification into one of the 227 sections of the Jyllands-

Posten website. The output for this task is based on the penultimate recurrent layer.

Hyper-parameters and Training We perform a grid search to find the best hyper-parameters for a single-task model (i.e., without any auxiliary tasks) and then keep those settings for all our experiments. We settle on a model with hidden size $H = 112$ and $N_k = 3$ layers, respectively. The dropout probability $p = 0.3$ gave best results for both models.

We train the model for 10 epochs using Adam optimizer with the default parameters, clipping the gradient updates so that their norm is not higher than 5. We train the different tasks sequentially for each epoch, with the lower level (POS tagging) first and the popularity prediction last. Additionally, we decay the learning rate by a factor of 0.9 after each epoch. While this is not common with adaptive methods such as Adam, it performed better. We stop training if the accuracy on the development set stops improving.

4 Results

Tables 1 and 2 report accuracy for logistic regression and neural classifiers. We also give the best score from Hardt and Rambow (2017) for comparison purposes (note, though, that the data sets are not identical and can therefore not be directly compared). We observe a substantial improvement over the baseline GRU when incorporating the pre-trained embeddings and both auxiliary tasks. It seems that pretrained embeddings and MTL act at least partly as regularizers, as these models trained for more epochs without overfitting. Interestingly, we observe a similar improvement over the word-based logistic regression models with a character n-gram model.

5 Analysis and Discussion

Our main focus in this paper is on MTL as a framework to explore the lexical, structural and topical knowledge involved in users' selection of headlines. However, recognizing a popular headline and giving advice on how to write one are not the same: we want to provide editors and journalists with insights as to what constructions are likely to attract more eyeballs.

One way to explore this is to examine individual words and their contribution to predictiveness. Table 3 displays the top 20 unigrams based on their coefficients in the logistic regression model. For each unigram we provide a translation (if needed) and a comment. We classify several unigrams as Deictic-reference. This follows Blom and Hansen (2015), who suggest that headline ”clickbait” often relies on forward-looking expressions, such as ”This”, as in, e.g., ”This is how you should eat an avocado”. Here, ”this” is a referring expression, but the reader understands that the antecedent will be found in the article body. Several of these top unigrams are names that are of specific topical interest in areas such as sports and politics. Others mention topics of more general interest (Researchers, dead, found). The second person pronoun is also on the list – in general, it was found that second person pronouns are far more predictive of popularity than first or third person pronouns. Finally, several unigrams identify sections of the newspaper of particular interest (car, weather, analysis, and satire).

6 Related Work

Prediction of news headline popularity is an increasingly important problem, as news consumption has moved online. The insights and models described here might well be applicable to related problems of interest: for example, Balakrishnan and Parekh (2014) and Jaidka et al. (2018) study the problem of predicting clicks on email subject lines.

Subramanian et al. (2018) show that a regression-based multitask approach can increase performance for the classification prediction of popularity. Their work looks at the popularity of online petitions, but the methodology applies to our subject as well, and ties in with the approaches taken in this project.

Benton et al. (2017) caution that in order to evaluate MTL results properly, we need to take the number of parameters into account. Our results to some extent support this finding, by showing that a simpler linear model can fare equally well on the task.

The choice of auxiliary tasks greatly influences the performance of MTL architectures, prompting several recent investigations into the selection process (Alonso and Plank, 2017; Bingel and Søgaard, 2017). However, it is still unclear whether these tasks serve as mere regularizers, or whether they can also impart some additional information.

model	input	accuracy
Hardt and Rambow (2017)	word unigrams	61.2
logistic regression	word unigrams	65.6
logistic regression	word bigrams	65.7
logistic regression	character 2-6grams	**67.4**

Table 1: Accuracy results for various Logistic Regression models

model	input	auxiliary tasks	epoch	accuracy
GRU 3 layers w/ 112 hidden	random embeddings	—	3	65.2
GRU 3 layers w/ 112 hidden	pretrained embeddings	—	5	66.8
GRU 3 layers w/ 112 hidden	pretrained embeddings	POS	5	65.7
GRU 3 layers w/ 112 hidden	pretrained embeddings	section	4	66.8
GRU 3 layers w/ 112 hidden	pretrained embeddings	POS+section	7	**67.4**

Table 2: Accuracy results for various GRU model implementations

Unigram	Translation	Comment
Magnussen		Name (Sports)
Trump		Name (Politics)
AGF		Name (Sports)
Test		
Her	Here	Deictic-reference
død	dead	topical
Wozniac		Name (Tech)
Trumps		Name (Politics)
Forskere	Researchers	topical
fundet	found	topical
du	you	pronoun
AGF-træner	AGF coach	Name (sports)
Se	Watch	Deictic-reference
Kevin		Name (Sports)
Islamisk	Islamic	Name (Politics)
Analyse	Analysis	Section
Sådan	This	Deictic-reference
Satire	Satire	Section
bil	car	Section
DMI	Weather	Section

Table 3: Top twenty Unigrams (Logistic Regression)

7 Conclusion

We presented an exploratory approach to predicting newspaper article popularity from headlines alone. Using pre-trained embeddings and a MTL setup, we are able to incorporate rich structural and semantic knowledge into the task and substantially improve performance. While the results are encouraging and allow the exploration of further auxiliary tasks (for example article word prediction), we find that a simple character-based n-gram model performs competitively. These findings highlight two aspects: 1) For any application of MTL, this is a strong case for comparing the results to non-deep models. While it is comparatively easy to show an improvement over the basic STL model, there might be other simple models that are competitive. 2) The selection of auxiliary tasks greatly influences the performance, even beyond simple regularization, and in a non-linear way. It does, however, provide us with a tool to test human intuitions about task interactions and the importance of certain problem aspects.

Acknowledgments

Thanks to A. Michele Colombo for help with the data and experiments. We also thank Jyllands-Posten for giving us access to the data, and to DSL for data for embeddings and POS annotations.

References

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *15th Conference of the European Chapter of the Association for Computational Linguistics*.

Raju Balakrishnan and Rajesh Parekh. 2014. Learning to predict subject-line opens for large-scale email marketing. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 579–584. IEEE.

Adrian Benton, Margaret Mitchell, and Dirk Hovy. 2017. Multitask Learning for Mental Health Conditions with Limited Social Media Data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 152–162.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 164–169.

Jonas Nygaard Blom and Kenneth Reinecke Hansen. 2015. Click bait: Forward-reference as lure in online news headlines. *Journal of Pragmatics*, 76:87 – 100.

Rich Caruana. 1993. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48.

Rich Caruana, Shumeet Baluja, Tom Mitchell, et al. 1996. Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. *Advances in neural information processing systems*, pages 959–965.

Daniel Hardt and Owen Rambow. 2017. Predicting user views in online news. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 7–12.

Kokil Jaidka, Tanya Goyal, and Niyati Chhaya. 2018. Predicting Email and Article Clickthroughs with Domain-adaptive Language Models. In *Proceedings of the 10th ACM Conference on Web Science*, pages 177–184. ACM.

Jørg Asmussen. 2018. Society for Danish Language and Literature. `http://dsl.dk`. [Online; accessed 28-April-2018].

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Ola Rønning, Daniel Hardt, and Anders Søgaard. 2018. Sluice Resolution without Hand-crafted Features over Brittle Syntax Trees. In *NAACL*.

Shivashankar Subramanian, Timothy Baldwin, and Trevor Cohn. 2018. Content-based Popularity Prediction of Online Petitions Using a Deep Regression Model. *Transactions of the Association for Computational Linguistics*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Hybrid Neural Attention for Agreement/Disagreement Inference in Online Debates

Di Chen[1], Jiachen Du[1], Lidong Bing[2], Ruifeng Xu[1]*
[1]Department of Computer Science, Harbin Institute of Technology (Shenzhen), China
[2]Tencent AI Lab, Shenzhen, China
`chandichn@gmail.com, dujiachen@stmail.hitsz.edu.cn`
`lyndonbing@tencent.com, xuruifeng@hit.edu.cn`

Abstract

Inferring the agreement/disagreement relation in debates, especially in online debates, is one of the fundamental tasks in argumentation mining. The expressions of agreement/disagreement usually rely on argumentative expressions in text as well as interactions between participants in debates. Previous works usually lack the capability of jointly modeling these two factors. To alleviate this problem, this paper proposes a hybrid neural attention model which combines self and cross attention mechanism to locate salient part from textual context and interaction between users. Experimental results on three (dis)agreement inference datasets show that our model outperforms the state-of-the-art models.

1 Introduction

The rise of various discussion forums and online debate platforms, has given users a lot of opportunities to express themselves and argue with each other. The online argumentation and discussion are always initiated and evolved by expressions of agreement or disagreement of participants. Inferring the agreement/disagreement in online debates is crucial for many other tasks in broader analysis of social media and argumentation mining, such as stance identification (Somasundaran and Wiebe, 2010), claim/argument extraction (Hidey et al., 2017) and persuasion analysis (Tan et al., 2016).

It is observed that the expression of agreement/disagreement in debates can be decomposed into two factors: 1) the self-expression of claims and 2) argumentative expressions to interact with other participants. To illustrate this observation, we show some examples in Figure 1, which is one of quote-response pair (Q-R pair) in 4forum online debate website. The response expressed disagreement with the claim in quote text. The mark *?* at the end of sentence carries strong emotion of authors, while the phrase *why doesn't He answer* refers to the claims of *God IS GOOD* and express the refutation to it.

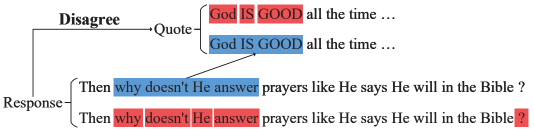

Figure 1: Sampled Q-R Pair with topic of evolution where the words colored red deliver crucial meaning of the text itself, while the words colored blue clarify the interactive relation between users.

Previous works on agreement/disagreement inference mainly focus on exploiting features to model the semantic information which only reveals author's self-expression. (Rosenthal and McKeown, 2015; Menini and Tonelli, 2016). These existing models treat agreement/disagreement inference as a ordinary sentiment classification problem and ignore the interactions between participants in the discussion. In order to jointly leverage the semantic information of the text and interactions between Q-R pairs, we regard the (dis)agreement inference as a special case of Natural Language Inference (NLI) (Rocktäschel et al., 2016), and propose a hybrid neural attention model to this problem. The proposed model consists of two kinds of attention: 1) **self attention** locates salient parts in text of quote and response, and 2) **cross attention** captures the interactive argumentations between Q-R pairs. The fusion of self and cross attention model is capable of jointly modeling the two important factors of inferring (dis)agreement in debates.

The main contributions of this paper are: (1) We propose a neural attention model for (dis)agreement inference which converts this

---

*Corresponding author.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 665–670
Brussels, Belgium, October 31 - November 4, 2018.

problem to a natural language inference task. The proposed model incorporates self and cross attention mechanism, jointly capturing significant part for current context and extracting interactive relations between Q-R pairs. (2) Experimental results on three datasets show that the proposed model significantly improves performance (measured by $F1$ score and accuracy) of state-of-the-art models by 1% on average. The visualization of extracted attention demonstrates different attention mechanism works effectively in different aspect for (dis)agreement inference.

2 Related Work

With the development of social forums, works on (dis)agreement inference have shifted to online debate. Abbott et al. (2011) utilize word-based and dependencies-based features to recognize disagreement in Internet Argument Corpus (IAC) (Walker et al., 2012). Rosenthal and McKeown (2015) present a new corpus derived from participant information, Agreement by Create Debaters (ABCD), and investigate new features for conversational structure. Further, Menini and Tonelli (2016) develop a SVM classifier to detect disagreement, relying on three aspects including sentiment-based, semantic and surface features extracted from both whole text and topic-related part. However, the performances of all these models highly depend on the quality of hand-crafted features. And these representations cannot reflect the interaction between quote and response.

In other NLP tasks, the end-to-end deep learning approaches with attention mechanism have shown impressive results. The attention mechanism is proposed by Bahdanau et al. (2014) in machine translation for selecting alignment between original words and foreign words before translation. For Document Classification, Yang et al. (2016) apply a hierarchical attention from word-level to sentence-level with learnable context vector. In Natural Language Inference (NLI), Liu et al. (2016) construct an inner-attention with mean pooling vector to seize important part from text itself. Hao et al. (2017) propose an cross attention modeling mutual influence between question and answer for Question Answering (QA). But there is no neural attention model incorporating both contextual and interactive information in the scenario of (dis)agreement inference.

3 Model

The overall architecture of our model is shown in Figure 2, comprising two parallel bi-directional LSTM (Hochreiter and Schmidhuber, 1997) networks as quote and response encoder and two attention components that respectively extract self and cross attention.

3.1 Quote and Response Encoder

A quote of length T is denoted as $[q_1, q_2, \cdots, q_T]$, where $q_t \in \mathbb{R}^{d_e}$ is the d_e-dimensional representation of the t-th word in the text sequence. Similarly, the corresponding response can be represented as $[r_1, r_2, \cdots, r_T]$, which shares the same vector space with quote. To model the dependence relation of text sequence, we leverage bi-directional LSTM (BiLSTM) to encode quote and response. The BiLSTM consists of a forward $\overrightarrow{LSTM}$ which reads the text from x_1 to x_T and a backward $\overleftarrow{LSTM}$ which reads from x_T to x_1:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(x_t), t \in [1, T] \quad (1)$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(x_t), t \in [T, 1] \quad (2)$$

Through concatenation, we obtain the representation of each time step $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \in \mathbb{R}^{2d}$ which integrates the information around x_t. The quote and response are encoded as $h_Q = [\overrightarrow{h_Q}; \overleftarrow{h_Q}] \in \mathbb{R}^{T\times 2d}$ and $h_R = [\overrightarrow{h_R}; \overleftarrow{h_R}] \in \mathbb{R}^{T\times 2d}$ respectively.

3.2 Attention Component

After encoding the implicit word semantics, we acquire the representation of both quote and response.

Self Attention

The first source taken into consideration should be the text sequence itself, i.e. the attention from quote to quote itself and that from response to response itself. When issuing an opinion, people tend to center on several keywords which convey the main idea. Thus in some sense, self attention is a kind of dependency parsing that drives the model to focus on salient parts of the context. Here, for quote $h_Q = [h_Q^1, h_Q^2, \cdots, h_Q^T]$, self attention generates signal s_Q^t by:

$$s_Q^t = \frac{\exp[\delta(h_Q^t)]}{\sum_{i=1}^{T} \exp[\delta(h_Q^i)]} \quad (3)$$

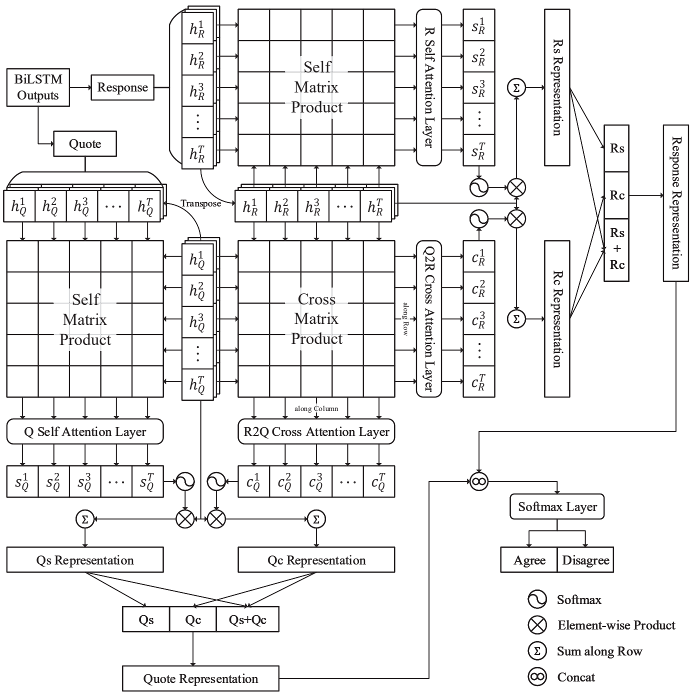

Figure 2: Model Architecture.

where δ is a transformation mapping $2d$-dimensional vector into scalar value, with learnable weight $W_S \in \mathbb{R}^T$ and bias $b_S \in \mathbb{R}$ defined as:

$$\delta(h_Q^t) = \tanh[W_S(h_Q \cdot (h_Q^t)^{\mathrm{T}}) + b_S] \qquad (4)$$

Similarly, with another parallel transformation, the self attention signal of response can be calculated as above. Then, we can obtain a more compact representation of quote and response respectively derived from the weighted sum, where $Q_S, R_S \in \mathbb{R}^{2d}$.

$$Q_S = \textstyle\sum_{t=1}^{T} s_Q^t h_Q^t \qquad (5)$$

$$R_S = \textstyle\sum_{t=1}^{T} s_R^t h_R^t \qquad (6)$$

Cross Attention

Another prominent facet comes from the relation between each Q-R pair, i.e the attention from quote to response and that from response to quote. In whether disagreement or agreement cases, both quote and response provides a precise context for each other. The cross attention integrates interactive influence which produces more specific features for (dis)agreement inference.

As discussed above, cross attention c_Q^t, c_R^t for quote and response can be computed by:

$$c_Q^t = \frac{\exp[\gamma(h_R^t)]}{\sum_{i=1}^{T}\exp[\gamma(h_R^i)]} \qquad (7)$$

$$c_R^t = \frac{\exp[\gamma'(h_Q^t)]}{\sum_{i=1}^{T}\exp[\gamma'(h_Q^i)]} \qquad (8)$$

where γ and γ' are two parallel transformation with learnable weight matrix $W_C, W_C' \in \mathbb{R}^T$ and bias $b_C, b_C' \in \mathbb{R}$ defined as:

$$\gamma(h_R^t) = \tanh[W_C(h_Q \cdot (h_R^t)^{\mathrm{T}}) + b_C] \qquad (9)$$

$$\gamma'(h_Q^t) = \tanh[W_C'(h_R \cdot (h_Q^t)^{\mathrm{T}}) + b_C'] \qquad (10)$$

The representation of whole sequence $Q_C, R_C \in \mathbb{R}^{2d}$ embracing cross attention signal are:

$$Q_C = \textstyle\sum_{t=1}^{T} c_Q^t h_Q^t \qquad (11)$$

$$R_C = \textstyle\sum_{t=1}^{T} c_R^t h_R^t \qquad (12)$$

Hybrid Attention

In order to cooperate the advantage of self attention and cross attention, we design hybrid attention to get a more specific representation for quote and response:

$$Q = Q_S \oplus Q_C \oplus (Q_S + Q_C) \qquad (13)$$

$$R = R_S \oplus R_C \oplus (Q_S + Q_C) \qquad (14)$$

where $Q, R \in \mathbb{R}^{6d}$ and $\oplus$ is the vector concatenation operation.

3.3 (Dis)agreement Inference

Finally, the quote representation Q and response representation R are concatenated as a vector v. We use a fully-connected network to project $12d$-dimensional representation into n-dimensional vector space, i.e.

$$y = softmax(W_l v + b_l) \tag{15}$$

where $y \in \mathbb{R}^n$ is predicted probability distribution for (dis)agreement inference, W_l and b_l are parameters of softmax layer.

In a supervised learning framework, we train our model in an end-to-end way. Given a set of training data $\{(Q_i, R_i), y_i\}$, let $\widehat{y_i}$ denote the predicted probability distribution, the goal of training is to minimize the cross-entropy loss:

$$loss = -\sum_i \sum_j y_i^j \log \widehat{y}_i^j \tag{16}$$

where i is the index of quote-response pair, j is the index of class and y_i is the ground truth of corresponding pair.

4 Experiment and Results

As prior work, we concentrate on direct disagreement and agreement between quote-response (Q-R) pairs. Specifically, in the proposed model, the size of hidden units is 128 and all word embeddings are initialized by GloVe (Pennington et al., 2014) of 300d. Both length of quote and response are set to 64, padded where necessary. Adam is the optimizer of model whose learning rate is $1e-3$, β is $(0.9, 0.999)$, ϵ is $1e-8$ and weight decay is $1e-5$. All models are trained by mini-batch of 32 instances, with 5-fold cross validation.

4.1 Datasets

We conduct experiments on three most commonly-used (dis)agreement inference datasets. Table 1 shows the detail of these datasets.

Table 1: Detail of Datasets

Dataset	# Disagree	# Agree	# Neutral
IAC	6,157	1,113	-
DP	12,899	11,875	-
ABCD	25,200	13,519	72,683

- *Internet Argument Corpus (IAC)* (Walker et al., 2012) is a corpus crawled from online political debate `4forums.com`. Following prior work, we compute average score for each pair and convert the score into binary labels, with $[-5, -1]$ as disagreement and $[+1, +5]$ as agreement.
- *Debatepedia (DP)* (Menini and Tonelli, 2016). DP corpus is crawled from `debatepedia.org`, which is an online encyclopedia of debates.
- *Agreement by Create Debaters (ABCD)* (Rosenthal and McKeown, 2015) is developed from `createdebate.com` with labels of agreement, disagreement and neutral. As the original settings, the comparison experiments are conducted on a balanced training set by downsampling and the full test set.

4.2 Comparison with Baseline Methods

As shown in Table 2, by accuracy and average $F1$-score in percentage, we compare our model with the best performing model of corresponding dataset to our knowledge. These models are reported in (Abbott et al., 2011; Menini and Tonelli, 2016; Rosenthal and McKeown, 2015) as Naive Bayes (NB), JRipχ^2 (ruled based classifier using χ^2 for feature selection), SVM, Maximum Entropy (ME), exploiting a rich suite of features including n-grams, sentiment lexicon and syntax.

We also analyze the contribution of each component in ablation experiment. BiLSTM-sum and BiLSTM-concat refer only sum or concat operation is applied to both self and cross attention respectively. Results show that BiLSTM-hybrid gives the best performance across all datasets regardless of data sizes. For smaller dataset such as IAC, our model outperforms the previous best methods by 8.8%. This outcome is consistent across other larger datasets with a significant improvement of 19.6% on DP. What' s more important, on DP, the length of text is longer than other datasets, so ordinary BiLSTM suffering from gradient vanishing results in the poor performance. It is the hybrid attention that effects. As for ABCD, compared with ME based on textual features, Our BiLSTM-hybrid also gives superior performance of average $F1$ in 3-way inference. Since ABCD is a corpus annotated by meta-thread rules, the ME attaching conversational structure attains the best

Table 2: Experimental results on three datasets.

2-way Inference				3-way Inference	
IAC		DP		ABCD	
Approaches	Accuracy	Approaches	Accuracy	Approaches	Average $F1$
NB	60.3	SVM	74.0	ME	50.8
JRipχ^2	68.2	-	-	ME+structure*	**77.6**
Liu et al., 2016	74.9	Liu et al., 2016	92.4	Liu et al., 2016	76.5
BiLSTM	72.9	BiLSTM	55.2	BiLSTM	71.1
BiLSTM-self	76.1	BiLSTM-self	93.5	BiLSTM-self	75.8
BiLSTM-cross	76.4	BiLSTM-cross	93.3	BiLSTM-cross	75.9
BiLSTM-sum	76.4	BiLSTM-sum	93.3	BiLSTM-sum	76.4
BiLSTM-concat	76.8	BiLSTM-concat	93.5	BiLSTM-concat	76.4
BiLSTM-hybrid	**77.0**	BiLSTM-hybrid	**93.6**	BiLSTM-hybrid	76.9

(*) Conversational structure is a corpus-specific feature.

performance. We think it a corpus-specific feature with weak generalization ability.

In addition, we adapt a NLI-oriented model proposed by Liu et al. (2016) as a stronger baseline, which comprises inner-attention with mean pooling. The mean pooling of text encoder is set as the summary representation for inner-attention to seize important part from text itself. It is similar to our self attention but with coarse-grained level from text to word. The results imply that our BiLSTM-hybrid modeling additional interaction with fine-grained attention from word to word performs better.

4.3 Qualitative Analysis

To validate that different attention focuses on different part of text sequence, we visualize the outputs of self attention layer and cross attention layer, with a Q-R pair of disagreement from IAC. As show in Figure 3, darker color indicates larger weight in the corresponding attention vector.

In quote, the self attention selects *good* which is exactly the point that quote wants to argue. Similarly, the self attention selects *?* in response, which indicates a rhetorical mood to show disagreement. On the other hand, even though *why doesn't he answer* in response is endowed less weight from the self attention, the cross attention highlights it and *god* in quote. When inspecting the cross matrix product of this pair, Figure 4 demonstrates that our method is able to model the reference between *god is good* and *why doesn't he answer* in the whole interactive context.

5 Conclusion

In this paper, we propose a hybrid attention based neural network for (dis)agreement inference in debate. The main motivation is to jointly leverage self attention for textual context and cross attention for interactions between users to improve the capability of inference on agreement/disagreement relations. Experimental results show that our model outperforms several strong baselines. Visualization of extracted attention of our model illustrates that our models is effective in capturing the main point from different aspects.

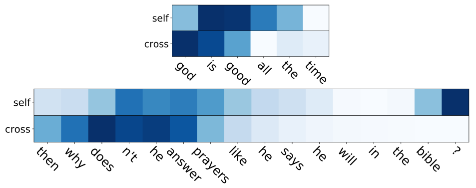

Figure 3: Attention Visualization. The topic is about evolution and the attitude of response is *disagreement*.

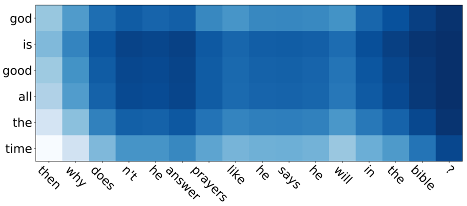

Figure 4: Cross Matrix Product Visualization.

Acknowledgements

This work was supported by the National Natural Science Foundation of China U1636103, 61632011, Key Technologies Research and Development Program of Shenzhen JSGG20170817140856618, Shenzhen Foundational Research Funding 20170307150024907.

References

Rob Abbott, Marilyn Walker, Pranav Anand, Jean E. Fox Tree, Robeson Bowmani, and Joseph King.

2011. How can you say such things?!?: Recognizing disagreement in informal political argument. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 2–11, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 221–231.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathy McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *Proceedings of the 4th Workshop on Argument Mining*, pages 11–21.

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

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Stefano Menini and Sara Tonelli. 2016. Agreement and disagreement: Comparison of points of view in the political domain. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2461–2470.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *The 4th International Conference on Learning Representations (ICLR 2016)*.

Sara Rosenthal and Kathy McKeown. 2015. I couldn't agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic*, pages 168–177.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. pages 116–124.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th international conference on world wide web*, pages 613–624. International World Wide Web Conferences Steering Committee.

Marilyn A Walker, Pranav Anand, Jean E Fox Tree, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *Eighth International Conference on Language Resources & Evaluation*, pages 23–25.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1480–1489.

Increasing In-Class Similarity by Retrofitting Embeddings with Demographic Information

Dirk Hovy
Bocconi University
`dirk.hovy@unibocconi.it`

Tommaso Fornaciari
Bocconi University
`fornaciari@unibocconi.it`

Abstract

Most text-classification approaches represent the input based on textual features, either feature-based or continuous. However, this ignores strong non-linguistic similarities like homophily: people within a demographic group use language more similar to each other than to non-group members. We use homophily cues to retrofit text-based author representations with non-linguistic information, and introduce a trade-off parameter. This approach increases in-class similarity between authors, and improves classification performance by making classes more linearly separable. We evaluate the effect of our method on two author-attribute prediction tasks with various training-set sizes and parameter settings. We find that our method can significantly improve classification performance, especially when the number of labels is large and limited labeled data is available. It is potentially applicable as preprocessing step to any text-classification task.

1 Introduction

Predicting socio-demographic author characteristics is becoming ever more relevant with the pervasive use of user-generated content. Classifying user attributes such as age and gender is useful for a number of applications both in the public sector, where it can support the investigation of crime (in forensic linguistics) or the determination of social policies, and in the private sector, where companies want to profile a potential consumer market, targeting communication strategies and advertising to specific communities. Furthermore, recent work in NLP has shown that incorporating author attributes in various NLP tasks can also improve performance (Volkova et al., 2013; Hovy, 2015; Hovy and Søgaard, 2015; Lynn et al., 2017; Preotiuc-Pietro et al., 2016).

In these tasks, authors are typically represented via their linguistic profiles, i.e., information available in the text. This includes both word-based features as well as continuous representations (embeddings). Generally, linguistic features are divided into content-related and strictly stylistic features. While the first can be effectively represented by (n-grams of) words which capture the topic and meaning of a text, the second ones focus on the use of function words, expressions, pronouns, syntactic structures, etc. There is evidence in the literature that content-related text characteristics are more effective than stylistic features for gender and age prediction (Fatima et al., 2017; Rosenthal and McKeown, 2011). This effect is the consequence of a non-linguistic auto-selection process known as *homophily*: people within in a demographic group tend to be more similar to each other than to other groups, and subjects belonging to different groups are therefore naturally more prone to discuss different topics.

Despite the large amount of available social media data (in April 2018, Facebook had more than two billion active users, YouTube and WhatsApp each one-and-a-half billion, and Twitter 330 million, see statista.com), we often encounter scenarios with limited availability of ground-truth user attributes, leading to remarkable performance differences to, say, blogs. This difference is due to the shortness of social media texts and the wider range of topics (Rangel et al., 2016), which weaken linguistic profile features. In such cases, improving author representations beyond the linguistic profiles can be especially useful.

We implement this intuition of leveraging demographic homophily by using retrofitting (Faruqui et al., 2015), a method introduced to refine word vectors to reflect semantic similarity information from lexicons. In our case, we increase the similarity between the (linguistically-based) continuous authors representations within each class (here: age or gender). Authors who

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 671–677
Brussels, Belgium, October 31 - November 4, 2018.

share the same gender or age therefore get more similar vector representations (see section 3.2). This effectively increases class-separability and can thereby improve classification performance.

We also experiment with a trade-off parameter α, which controls the relative influence of the retrofitting process vs. the original embedding vector on the retrofit representation, allowing us to explore the effect of both factors on the final prediction outcome.

In order to extend the in-class homophily information to unlabeled data, we induce a transformation matrix to translate between the original and retrofitted embedding space. This matrix can be applied to unlabeled data to transform the author representations in the test set.

We use a set of almost 100K authors to predict age and gender. In order to explore limited-resource scenarios, we experiment with a range of training set sizes. Our results indicate that demographic retrofitting of linguistic representations substantially increases classification performance for age and gender prediction, especially in low-resource scenarios.

It is an easy, fast, and efficient preprocessing step that can substantially improve classification performance. We show our method for author-attribute prediction, but believe it can potentially be applied to *any* text-classification task.

Contributions In this paper, we introduce demographic retrofitting based on in-class homophily, and make the following contributions:

1. we present a substantial expansion of the original retrofitting algorithm (Faruqui et al., 2015). In contrast to prior work, which relies on external ontologies, our method relies solely on the information contained within the training data.

2. We show how to generalize the transformation from training data to unlabeled data, using a translation matrix.

3. We publicly release all our data and models on our GitHub page, https://github.com/Bocconi-NLPLab/retrofit_attributes.

2 Data

We use data from Hovy et al. (2015), a collection of reviews of online companies from various countries, including author information. We select all reviews written in English from American and British sources, if they include both age and gender of the author, and if the review is at least 10 tokens long (shorter reviews tend to be mistokenized URLs or replies). We aggregate the reviews by users, so that each instance is a collection of texts' from a unique user. This leaves us with 98,608 individual users, and about 8M words (roughly 80 words per instance). For each user, we use the age (discretized by decade) and gender (self-stated as binary, and augmented by Hovy et al. (2015) based on the users' first name) as target variables. We minimally preprocess the text data, collapsing all numbers into 0s, and tokenizing via spacy (Honnibal and Johnson, 2015).

3 Methodology

In our experiments, we are interested in the effect of homophily-inducing retrofitting on author-attribute prediction. In order to evaluate the effect, we compare the performance of author representations based on linguistic input to the performance of the same representation retrofitted to the author attribute class in question. In this section, we outline the details for the different steps.

3.1 Linguistic author representations

We train Doc2Vec, a paragraph2vec (Le and Mikolov, 2014) implementation, on the corpus, inducing a 98K-by-300 matrix D, where each row represents an author. We follow the parametrization suggested in Lau and Baldwin (2016), setting the window size to 15, minimum word-frequency to 10, negative samples to 5, downsampling rate to 0.00001, and run for 1000 iterations. We use the resulting author embeddings as input to the author-attribute classifier (see 3.4). We induce the author embeddings over the entire corpus of 98K authors, without recurrence to age or gender information.

As comparison, we also create a bag-of-words (BOW) representation with the same dimensionality. We use χ^2 as selection criterion to find the top 300 words in the training data, separately for both age and gender classification.

3.2 Retrofitting

Our goal is to enhance the author representations, which are based on linguistic similarity, with demographic information about the target variable (say, age). In order to introduce this information into the vector space, we rely on *retrofitting*, by in-

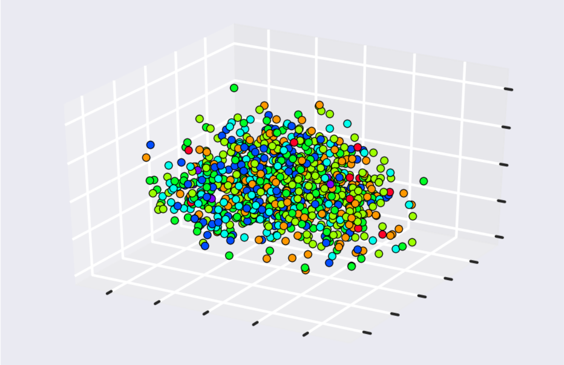

(a) Non-retrofitted

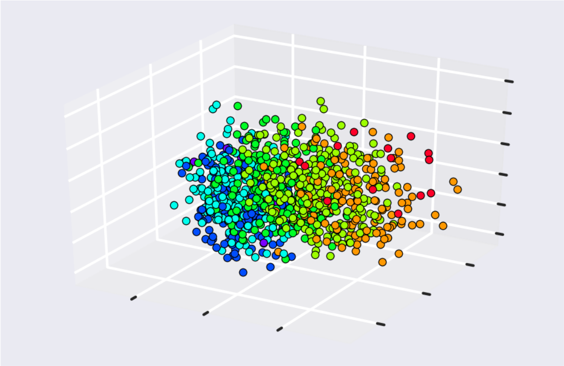

(b) Retrofitted

Figure 1: Schematic representation of 500 authors colored by age group, without (top) and with (bottom) retrofitting

creasing the similarity of authors within the same target group (say, people in their 20s). We thereby separate the target classes in embedding space, making them easier to differentiate by a classifier.

Faruqui et al. (2015) introduced retrofitting of word vectors based on external ontologies, such as WordNet (Miller, 1995) or PPDB (Ganitkevitch et al., 2013). Instead of these resources, we map each labeled author to the list of all other authors with the same label in the training data. Formally, we create a set Ω containing tuples of authors $(d_i, d_j | y_i = y_j)$. We do this separately for each demographic dimension - age and gender.

During retrofitting, we iteratively update the author representation in the training data (initially linguistically-based) to increase the cosine similarity between authors within the same class (as defined in Ω). This creates a retrofitted matrix $\hat{D}_{train}$ of the original author matrix D_{train}. The update for an author representation d_i is a weighted combination of the original embedding and the average over all its current neighbors:

$$\hat{d}_i = \alpha d_i + \beta \frac{\sum_{j:(i,j)\in\Omega} \hat{d}_j}{N}$$

where d_i is the original linguistic representation vector, $N = |\{\forall j : (i,j) \in \Omega\}|$ is the set of all embeddings in the same label group, and α and β are hyper-parameters that control the trade-off between the original representation and the updates from the neighboring embeddings during retrofitting. In Faruqui et al. (2015), $\alpha = \beta$. In contrast, we define

$$\beta = 1 - \alpha$$

By varying α from 0 to 1, we can control the strength of the retrofitting process. $\alpha = 1$ simply reproduces the original matrix, i.e., $\hat{D} = D$, whereas $\alpha = 0$ only relies on the neighborhood updates after the initialization. Figure 1 shows a sample of 500 users in a non-retrofitted (1a) and retrofitted (1b) 3D embedding space, colored by class. The color distribution shows how people belonging to the same group get drawn closer to each other in embeddings space when using retrofitting.

3.3 Translation

We can only retrofit the embeddings of authors in the training set D_{train}, since we need information about the class label in order to construct Ω. However, the retrofitting process changes the configuration of the embedding space (into $\hat{D}_{train}$), so a separating hyperplane learned on $\hat{D}_{train}$ will not be applicable to a test set D_{test} in the original embedding space.

In order to extend the homophily information to authors in the test set, we use a *translation matrix* T (a 300×300 matrix), which approximates the transformation from the original training data matrix D_{train} into the retrofitted matrix $\hat{D}_{train}$. We obtain T by minimizing the least-square difference in $D_{train} \cdot T = \hat{D}_{train}$.

T captures the retrofitting operation, and allows us to modify the test subjects' representations *as if* age and gender were known, despite the absence of class information. In particular, by applying T to the matrix of the author embeddings in the unlabeled test set D_{test}, we obtain a retrofitted version $\hat{D}_{test}$ that preserves the transformation learned on the training data. Since the least-square approximation is not perfect, we find that in practice fitting a classifier on the approximation $D_{train} \cdot T$

works better than using $\hat{D}_{train}$, acting as a regularizer.

3.4 Classification

We retrofit the author embeddings in the training set (see 3.2) and learn a translation matrix to transform the representations of the remaining authors in the test set. We train three separate Logistic Regression classifiers: one on author embeddings, one on the retrofit embeddings, and one on BOW features. It is technically possible to retrofit BOW representations as well, but in practice, the classifier does not converge, as word count-based vectors do not represent a continuous space that captures latent similarities.

We then use the three classifiers to predict the author attributes of the remaining authors in the test data set. We evaluate the results via micro-F1 score (averaged over 100 runs), since our tasks include imbalanced multi-class scenarios: micro-F1 weights the contribution of each class according to their relative size and is therefore more informative than accuracy.

Since we are interested in the effect of the training set size on performance, we vary the number of available training examples from 1000 to 10,000, using the remaining authors as test set. For each training set size, we collect 100 random subsamples and average over them.

4 Results

The learning curves in Figure 2 show that retrofitting outperforms both the original author embeddings and BOW representations for age (top) and gender (bottom) prediction in terms of F1. The effect is stronger when little training data is available. We evaluate the statistical significance of the difference between results with retrofitting and original embeddings via a bootstrap sampling test. We do not test against BOW, since this is consistently lower than embeddings. The resulting p-value are given in the respective figures. For gender classification, there are small, but not significant improvements with retrofitting. By contrast, for age classification, small values of α (0.01, 0.1, and 0.25) result in significantly better classification than when using any other method.

The performance differences between the methods are generally more pronounced for age-prediction, which has 10 possible labels, than for gender prediction (two labels). The difference in optimal α value suggests a relation between α and label space.

In both tasks, the best result is achieved by choosing a low α, i.e., by giving more weight to the demographic association of the users than to their linguistic feature representations. In practice, this value should be determined via cross-validation: here we show different levels of α in order to give some intuition of its on performance. Note that the curves for the original embeddings and BOW are unaffected by α and do not change. We repeat them at each figure for comparison. Increasing α eventually converges with the original embeddings, but we see that even intermediate values can be close to the original embeddings.

5 Related Work

The first studies to apply statistical NLP techniques to author attribute prediction are Koppel et al. (2002); Argamon et al. (2003), using the British National Corpus (BNC). The same authors also introduced the use of blogs as data source (Koppel et al., 2006).

In recent years, predicting socio-demographic variables from text has seen increased interest, with several corpora for the classification of age and gender, covering various languages, such as English (Schler et al., 2006; Rosenthal and McKeown, 2011), Spanish, French, German, Dutch (Company and Wanner, 2015), Greek (Mikros, 2012), Chinese (Zhang et al., 2016), and Vietnamese (Pham et al., 2009).

A big contribution in this field, however, was the shared tasks of the PAN workshops (Rangel et al., 2013, 2014, 2015, 2016).

Research has identified a variety of linguistic features, ranging from "stylistic features with n-grams models, parts-of-speech, collocations, LDA, different readability indexes, vocabulary richness, correctness or verbosity" (Rangel et al., 2016). However, none of these papers used demographic information directly in the author representations.

Closest to our method are Lopez-Monroy et al. (2013), who propose the use of second-order representations. They created specific profiles for the target classes, and exploited them for the creation of the profile of each document. In both cases, the linguistic representation of the documents passes through a class-related profile.

The methods applied in the PAN workshops also reflect the recent research trend towards word embeddings, which we explore in this pa-

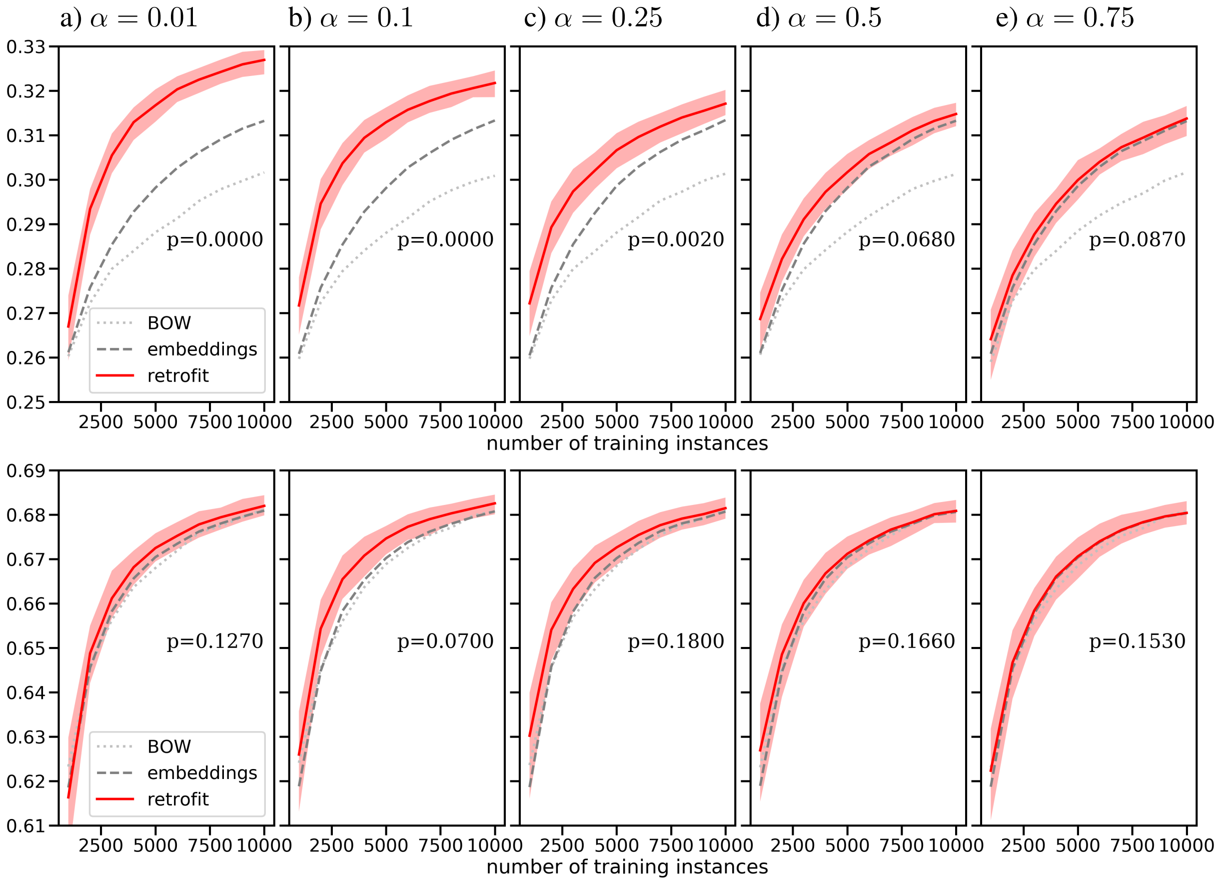

Figure 2: Learning curves (micro-F1) for 3 classifiers on age (top) and gender prediction (bottom) for different values of α. Retrofitting influence decreases from left to right. All data points averaged over 100 runs. Shaded area is 95%-confidence interval for retrofitting. p-values denote statistical difference between original and retrofit embeddings according to bootstrap test.

per. Bayot and Gonçalves (2016) first used `word2vec` embeddings as input features to a SVM classifier, followed by the use of convolutional (CNN) and recurrent neural networks (RNN) by Miura et al. (2017). Markov et al. (2016) also created document representations through word2vec, using a Logistic Regression classifier.

6 Conclusion

We explore retrofitting text-based author embeddings with non-linguistic demographic information to increase in-class similarity. This method increases class-separability to improve classification performance. We use a corpus of almost 100K users, and evaluate the effect of our method on age and gender prediction for various levels of available training data. We find that aggressive retrofitting (prioritizing homophily over linguistic embeddings) is beneficial for prediction performance, especially when the available amount of training data is limited. While the effect diminishes with increased training data size, our approach provides a simple method to incorporate non-linguistic knowledge into author representations. For another application (introducing geographic information into city representations), see Hovy and Purschke (2018). Our method is fast, simple, and applicable to any problem represented in embedding space. It is therefore a viable preprocessing step to any text-classification task.

Acknowledgements

We would like to thank the anonymous reviewers, whose constructive feedback improved the paper, as well as Carlo Baldassi for many enlightening discussions, as well as Barbara Plank, Noah Smith, and Afshin Rahimi, whose feedback helped our conceptualization of the problem.

References

Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. 2003. Gender, genre, and writing style in formal written texts. *TextInterdisciplinary Journal for the Study of Discourse*, 23(3):321–346.

Roy Bayot and Teresa Gonçalves. 2016. Author Profiling using SVMs and Word Embedding Averages-Notebook for PAN at CLEF 2016.

Juan Soler Company and Leo Wanner. 2015. Multiple Language Gender Identification for Blog Posts. In *CogSci*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615.

Mehwish Fatima, Komal Hasan, Saba Anwar, and Rao Muhammad Adeel Nawab. 2017. Multilingual author profiling on Facebook. *Information Processing & Management*, 53(4):886–904.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 752–762.

Dirk Hovy, Anders Johannsen, and Anders Søgaard. 2015. User review sites as a resource for large-scale sociolinguistic studies. In *Proceedings of the 24th International Conference on World Wide Web*, pages 452–461. International World Wide Web Conferences Steering Committee.

Dirk Hovy and Christoph Purschke. 2018. Capturing Regional Variation with Distributed Place Representations and Geographic Retrofitting. In *Proceedings of EMNLP*.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 483–488.

Moshe Koppel, Shlomo Argamon, and Anat Rachel Shimoni. 2002. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4):401–412.

Moshe Koppel, Jonathan Schler, Shlomo Argamon, and James Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs*.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. page 78.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

A Pastor Lopez-Monroy, Manuel Montes-Y-Gomez, Hugo Jair Escalante, Luis Villasenor-Pineda, and Esau Villatoro-Tello. 2013. INAOEs participation at PAN13: Author profiling task. In *CLEF 2013 evaluation labs and workshop*.

Veronica Lynn, Youngseo Son, Vivek Kulkarni, Niranjan Balasubramanian, and H Andrew Schwartz. 2017. Human Centered NLP with User-Factor Adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1146–1155.

Ilia Markov, Helena Gómez-Adorno, Juan-Pablo Posadas-Durán, Grigori Sidorov, and Alexander Gelbukh. 2016. Author profiling with doc2vec neural network-based document embeddings. In *Mexican International Conference on Artificial Intelligence*, pages 117–131. Springer.

George K Mikros. 2012. Authorship attribution and gender identification in Greek blogs. *Methods and Applications of Quantitative Linguistics*, 21:21–32.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Yasuhide Miura, Tomoki Taniguchi, Motoki Taniguchi, and Tomoko Ohkuma. 2017. Author profiling with word+ character neural attention network. *Cappellato et al.[13]*.

Dang Duc Pham, Giang Binh Tran, and Son Bao Pham. 2009. Author profiling for Vietnamese blogs. In *Asian Language Processing, 2009. IALP'09. International Conference on*, pages 190–194. IEEE.

Daniel Preotiuc-Pietro, Wei Xu, and Lyle H Ungar. 2016. Discovering user attribute stylistic differences via paraphrasing. In *AAAI*, pages 3030–3037.

Francisco Rangel, Paolo Rosso, Irina Chugur, Martin Potthast, Martin Trenkmann, Benno Stein, Ben Verhoeven, and Walter Daelemans. 2014. Overview of the 2nd author profiling task at pan 2014. In *CLEF 2014 Evaluation Labs and Workshop Working Notes Papers, Sheffield, UK, 2014*, pages 1–30.

Francisco Rangel, Paolo Rosso, Moshe Koppel, Efstathios Stamatatos, and Giacomo Inches. 2013. Overview of the author profiling task at PAN 2013. In *CLEF Conference on Multilingual and Multimodal Information Access Evaluation*, pages 352–365. CELCT.

Francisco Rangel, Paolo Rosso, Martin Potthast, Benno Stein, and Walter Daelemans. 2015. Overview of the 3rd Author Profiling Task at PAN 2015. In *CLEF*, page 2015. sn.

Francisco Rangel, Paolo Rosso, Ben Verhoeven, Walter Daelemans, Martin Potthast, and Benno Stein. 2016. Overview of the 4th author profiling task at PAN 2016: cross-genre evaluations. In *Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings/Balog, Krisztian [edit.]; et al.*, pages 750–784.

Sara Rosenthal and Kathleen McKeown. 2011. Age prediction in blogs: A study of style, content, and online behavior in pre-and post-social media generations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 763–772. Association for Computational Linguistics.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 6, pages 199–205.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of EMNLP*, pages 1815–1827.

Wanru Zhang, Andrew Caines, Dimitrios Alikaniotis, and Paula Buttery. 2016. Predicting Author Age from Weibo Microblog Posts. In *LREC*.

A Syntactically Constrained Bidirectional-Asynchronous Approach for Emotional Conversation Generation

Jingyuan Li, Xiao Sun[*]
Anhui Province Key Laboratory of Affective Computing
and Advanced Intelligent Machine
Hefei University of Technology, Hefei, China
`lijingyuan@mail.hfut.edu.cn, sunx@hfut.edu.cn`

Abstract

Traditional neural language models tend to generate generic replies with poor logic and no emotion. In this paper, a syntactically constrained bidirectional-asynchronous approach for emotional conversation generation (E-SCBA) is proposed to address this issue. In our model, pre-generated emotion keywords and topic keywords are asynchronously introduced into the process of decoding. It is much different from most existing methods which generate replies from the first word to the last. Through experiments, the results indicate that our approach not only improves the diversity of replies, but gains a boost on both logic and emotion compared with baselines.

1 Introduction

In recent years, as artificial intelligence has developed rapidly, researchers are pursuing technologies with greater similarities to human intelligence. As a subjective factor, emotion performs an elemental difference between humans and machines. In other words, machines that could understand emotion would be more responsive to human needs. For example, in education, positive emotions improve students' learning efficiency (Kort et al., 2002). In healthcare, mood prediction can be used in mental health counseling to help anticipate and prevent suicide or depression (Taylor et al., 2017; Jaques et al., 2017). To make machine more intelligent, we must resolve the conundrum of emotional interactions.

There are tons of researches about conversation, an important channel for communication between humans. And lots of work has recently been carried out in open-domain conversation devoted to generating meaningful replies (Vinyals and Le, 2015; Li et al., 2016; Serban et al., 2016). Unfortunately, the factors considered in these methods only concerns topic, like (Xing et al., 2017), where they failed to take emotion into account. Unlike the former, the work in (Zhou et al., 2017) first addressed the emotional factor in large-scale conversation generation, and it showed that emotional replies obtain superior performances compared to the baselines that did not consider emotion. However, two defects still manifest themselves in the aforementioned models. **First**, all methods above only adopted a single factor (i.e., topic or emotion), because of which the bias of information can not comprehensively summarize the human conversations to achieve favorable results. **Second**, the way that generates replies from the first word to the last can lead to a decline in diversity, limited by the high-frequency generic words in the beginning (e.g., I and you), as argued in (Mou et al., 2016).

The deficiencies above inspire us to introduce a new approach called E-SCBA, studying both emotion and topic. Three main contributions are presented in this paper: (1) It conducts a study of compound information, which constitutes the syntactic constraint in the conversation generation. (2) Different from the work in (Mou et al., 2016), a bidirectional-asynchronous decoder with multi-stage strategy is proposed to utilize the syntactic constraint. It ensures the unobstructed communication between different information and allows a fine-grained control of the reply to address the problem of fluency and grammaticality as argued in (Ghosh et al., 2017; Zhou et al., 2017). (3) Our experiments show that E-SCBA work better on emotion, logic and diversity than the general seq2seq and other models that consider only a single factor during the generation.

2 Model

2.1 Overview

The whole process of emotional conversation generation consists of the following three steps:

---

[*] The corresponding author of this paper.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 678–683
Brussels, Belgium, October 31 - November 4, 2018.

***Step I*:** Given a post, we first use two networks combined with category embeddings to respectively predict emotion keyword and topic keyword that should appear in the final reply (see Section 2.2).

***Step II*:** After the prediction, a newly designed decoder is used to introduce both keywords into the content[1], as shown in Figure 1. It first produces a sequence of hidden states based on the emotion keyword (Step I), and then uses an emotional attention mechanism to affect the generation of middle sequence, which is based on the topic keyword (Step II). The remaining two sides are ultimately generated by the combination of middle part and keywords (Step III). A detailed description is given in Section 2.3.

***Step III*:** Finally, a direction selector is used to arrange the generated reply in a logically correct order by selecting the better one from forward and backward forms of the reply generated in the last step (see Section 2.4).

In this work, we default that the replies contain at least one emotion keyword and one topic keyword, which are expected to appear in the dictionaries we used.

2.2 Keyword Predictor

The keywords to be selected are pre-stored in the prepared dictionaries. The adopted emotion dictionary was proposed by (Xu et al., 2008), which contains 27,466 emotion words divided into 7 categories: *Happy*, *Like*, *Surprise*, *Sad*, *Fear*, *Angry* and *Disgust*. The adopted topic dictionary was obtained by the LDA model (Blei et al., 2003), including 10 categories with 100 words for each category. And to avoid situations in which emotion and topic keywords are predicted to be the same word, all the overlapping words in these two dictionaries default to emotion keywords.

The prediction of emotion and topic keywords follows the similar path. We first derive topic category and emotion category from the post with two classifiers separately. To be more specific, the pretrained LDA model is used for the topic category inference. And the work in (Sun et al., 2018) is applied for emotion. The concrete model is an emotion transfer network. Given a specific external stimuli (e.g., a sentence), the network produce an emotional response, which is specifically an emotion category in this work. After this, combining the sum of hidden states $\tilde{h} = \sum_{i=1}^{T} h_i$ from encoder and the category embeddings $\boldsymbol{k} = \{k^{et}, k^{tp}\}$, keywords are predicted as follows:

$$p(w_{et}^{k}|\boldsymbol{x}, k^{et}) = softmax(\mathbf{W}_{et}^{w}[\tilde{h}; k^{et}]) \quad (1)$$

$$p(w_{tp}^{k}|\boldsymbol{x}, k^{tp}) = softmax(\mathbf{W}_{tp}^{w}[\tilde{h}; k^{tp}]) \quad (2)$$

where w_{et}^{k} and w_{tp}^{k} separately represent the emotion keyword and topic keyword that are expected to appear in the reply.

2.3 Bidirectional-Asynchronous Decoder

Due to the decoder architecture shown in Figure 1, we suppose the reply in this section is $\boldsymbol{y} = (\boldsymbol{y}^{ct}, w_{tp}^{k}, \boldsymbol{y}^{md}, w_{et}^{k}, \boldsymbol{y}^{ce})$[2] where $\boldsymbol{y}^{md}$ is the middle part between two keywords and $\boldsymbol{y}^{ct}$, $\boldsymbol{y}^{ce}$ represent the remaining sides connected to the topic keyword and emotion keyword. The generation of middle part $\boldsymbol{y}^{md} = (y_1^{md}, ..., y_K^{md})$ can be described as follows:

$$c_j^{et} = f_{att}^{et}(s_{j-1}^{tp}, \{s_i^{et}\}_{i=1}^{K'}) \quad (3)$$

$$p(\boldsymbol{y}^{md}|\boldsymbol{x}, \boldsymbol{w}^{k}) = \prod_{j=1}^{K} p(y_j^{md}|y_{j-1}^{md}, s_j^{tp}, c_j^{et}) \quad (4)$$

where $\boldsymbol{w}^{k} = < w_{et}^{k}, w_{tp}^{k} >$ represents the set of keywords, s_i^{et} and s_j^{tp} separately represent the decoding state of the steps that introduce emotion keyword and topic keyword. c_j^{et} is the emotional constrain unit at time j, computing by the emotion control function f_{att}^{et} as follows:

$$c_j^{et} = \sum_{i=1}^{K'} \alpha_{j,i}^{et} s_i^{et} \quad (5)$$

$$\alpha_{j,i}^{et} = \frac{\exp(e_{j,i}^{et})}{\sum_{t=1}^{K'} \exp(e_{j,t}^{et})} \quad (6)$$

$$e_{j,i}^{et} = (\mathbf{v}_{\alpha}^{md})^{T} tanh(\mathbf{W}_{\alpha}^{md} s_{j-1}^{tp} + \mathbf{U}_{\alpha}^{md} s_i^{et}) \quad (7)$$

where $e_{j,i}^{et}$ represents the impact scores of the emotion state s_i^{et} on the topic state s_{j-1}^{tp}.

After generating the middle part, we connect it with the keywords to form a new sequence. Two seq2seq models are used to encode the connected

---

[1] Syntactic constraint starts to work here, and can be intuitively interpreted as relative positions of emotion words and topic words, as well as different combinations between them.

[2] For the training data in opposite direction, we reversed the target replies to meet the requirement.

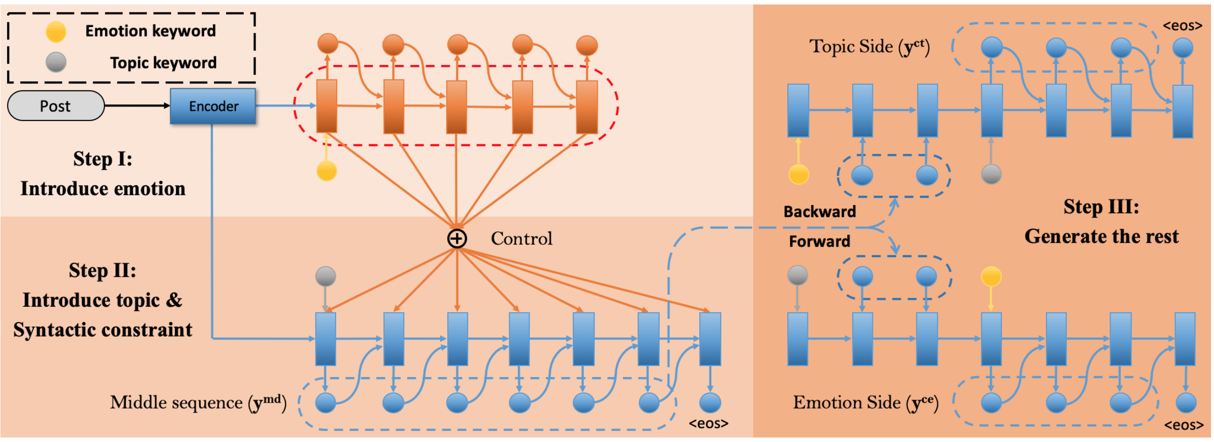

Figure 1: The process of generating replies in the test. The middle part of the reply is generated in Steps I and II, and the remaining two sides are generated in Step III. The RNN networks used in the decoder do not share the parameters with each other.

sequences and decode $\boldsymbol{y}^{ce} = (y^{ce}_1, ..., y^{ce}_M)$ and $\boldsymbol{y}^{ct} = (y^{ct}_1, ..., y^{ct}_N)$, as below:

$$p(\boldsymbol{y}^{ce}|\boldsymbol{w}^k, \boldsymbol{y}^{md}) = p(\boldsymbol{y}^{ce}|[w^k_{tp}, \boldsymbol{y}^{md,f}, w^k_{et}]) \quad (8)$$

$$p(\boldsymbol{y}^{ct}|\boldsymbol{w}^k, \boldsymbol{y}^{md}) = p(\boldsymbol{y}^{ct}|[w^k_{et}, \boldsymbol{y}^{md,b}, w^k_{tp}]) \quad (9)$$

where $\boldsymbol{y}^{md,f}$ and $\boldsymbol{y}^{md,b}$ are the forward and backward situations of the middle part, respectively.

2.4 Direction Selector

To make the samples meet the requirements of decoder, by default we place the topic keyword as the first keyword on the left and the emotion keyword on the right in training. However, in real situations, the topic keyword does not always appear before the emotion keyword, where we must determine correct direction by the machine.

By connecting the results in the preceding section, we get $\boldsymbol{y}^f = (\boldsymbol{y}^{ct,b}, w^k_{tp}, \boldsymbol{y}^{md,f}, w^k_{et}, \boldsymbol{y}^{ce,f})$ as the forward situation and $\boldsymbol{y}^b$ means the backward situation. GRU networks are used as encoders to process sequences in different situations, which do not share parameters. And the direction is predicted by:

$$p(d|\boldsymbol{y}^f, \boldsymbol{y}^b) = sigmoid(\mathbf{W}^d[\tilde{h}^{d,f}, \tilde{h}^{d,b}]) \quad (10)$$

$$\tilde{h}^{d,*} = \sum_{i=1}^{T'} \mathbf{GRU}(y^*_i) \quad (11)$$

where $* \in \{f, b\}$ means forward or backward. After the operation completes, one of the sequences $\boldsymbol{y}^f$ and $\boldsymbol{y}^b$ should conform to our expectations.

3 Experiment

3.1 Data

We evaluated and trained E-SCBA on the emotional conversation dataset NLPCC2017. There are a total of 1,119,201 Chinese post-reply pairs in the set. The dictionaries mentioned in Section 2.2 were used to mark the conversation. The cases whose replies contain both emotion keywords and topic keywords account for 42.6% (476,121) of the total[3], which are suitable data for the bidirectional-asynchronous decoder. We randomly sampled 8,000 for validation, 3,000 for testing and the rest for training. We also sampled another 60,000 pairs from the training set to train the LDA model[4] mentioned in Section 2.2. Besides, an error analysis is presented based on a Chinese movie subtitle dataset which is collected from the Internet.

3.2 Metrics

To evaluate our approach, we use the metrics as below:

Embedding-based Metrics: We measure the similarity computed by cosine distance between a candidate reply and the target reply using sentence-level embedding, following the work in (Liu et al., 2016; Serban et al., 2017).

---

[3] Please note that we did not use the original labels of the dataset, but the emotion categories of the keywords as labels to avoid unnecessary bias. For cases that contain multiple topic keywords or emotion keywords, we chose the keywords that appear less frequently to reduce imbalances.

[4] High frequency words and stop words, which have no benefit to the topics, were removed in advance.

Method	**Overall**			Happy			Like			Surprise		
	C	L	E	C	L	E	C	L	E	C	L	E
S2S	1.301	0.776	0.197	1.368	0.924	0.285	1.341	0.757	0.217	1.186	0.723	0.076
S2S-AW	1.348	1.063	0.231	1.437	1.097	0.237	1.418	1.125	0.276	**1.213**	**0.916**	0.105
E-SCBA	**1.375**	**1.123**	**0.476**	**1.476**	**1.286**	**0.615**	**1.437**	**1.173**	**0.545**	1.197	0.902	**0.245**
Method	Sad			Fear			Angry			Disgust		
	C	L	E	C	L	E	C	L	E	C	L	E
S2S	1.393	0.928	0.237	1.245	0.782	0.215	**1.205**	0.535	0.113	1.368	0.680	0.236
S2S-AW	1.423	1.196	0.293	1.260	1.105	0.272	1.198	**0.860**	0.182	1.488	1.145	0.253
E-SCBA	**1.497**	**1.268**	**0.525**	**1.268**	**1.124**	**0.453**	1.110	0.822	**0.347**	**1.637**	**1.289**	**0.603**

Table 1: The results of human annotations (C = Consistency, L = Logic, E = Emotion).

Method	G–M	E–A	V–E	D–1	D–2
S2S	0.297	0.382	0.284	0.086	0.212
S2S-STW	0.328	0.433	0.327	0.135	0.343
S2S-SEW	0.322	0.421	0.319	0.146	0.364
S2S-AW	0.363	0.485	0.352	0.162	0.417
E-SCBA	**0.405**	**0.553**	**0.395**	**0.218**	**0.582**

Table 2: The results of automatic evaluation (G–E = Greedy Matching, E–A = Embedding Average, V–E = Vector Extrema).

Distinct Metrics: By computing the number of different unigrams (Distinct-1) and bigrams (Distinct-2), we measure information and diversity in the candidate replies, following the work in (Li et al., 2016; Xing et al., 2017).

Human Annotations: We asked four annotators to evaluate the replies[5] generated from our approach and baselines from *Consistency*, *Logic* and *Emotion*. *Consistency* measures fluency and grammaticality of the reply on a three-point scale: 0, 1, 2; *Logic* measures the degree to which the post and the reply logically match on a three-point scale[6] as above; *Emotion* judges whether the reply includes the right emotion. A score of 0 means the emotion is wrong or there is no emotion, and a score of 1 is the opposite.

3.3 Baselines

In the experiments, E-SCBA is compared with the following baselines:

S2S: the general seq2seq model with attention method (Bahdanau et al., 2014).

S2S-STW: the model uses a synchronous method that starts generating its reply solely and directly from the topic keyword.

S2S-SEW: the model uses a synchronous method that starts generating its reply solely and directly from the emotion keyword.

S2S-AW: the model uses an asynchronous method the same as (Mou et al., 2016).

The synchronous method in S2S-STW and S2S-SEW was mentioned in (Mou et al., 2015), acting as the contrast to the asynchronous models.

3.4 Results and Discussion

The results of automatic evaluation are shown in Table 2. Compared with the best model (S2S-AW) that considers only a single factor, E-SCBA makes significant improvement on the distinct metrics (+0.056 and +0.165), which verifies the effectiveness of taking both emotion and topic information into account to improve the diversity. Likewise, our approach also respectively achieves 0.042, 0.068 and 0.043 gains on G-M, E-A and V-E, benefiting from the compound information that captures the thrust of human conversation so that E-SCBA has a better ability to learn the goal distribution. Furthermore, the grades of the asynchronous models are higher than the synchronous models on both kinds of metrics, showing that the asynchronous method is a more suitable way for content-introducing conversation generation.

Table 1 depicts the human annotations (*t-test*: $p < 0.05$ for C and L, $p < 0.01$ for E). Overall, E-SCBA outperforms S2S-AW on all three metrics, where the compound information plays a positive role in the comprehensive promotion. However, in *Surprise* and *Angry*, the grades of *Consistency* and *Logic* are not satisfactory, since the data for them are much less than others (*Surprise* (1.2%)

---

[5] 700 conservations in total, 100 for each emotion category, were sampled randomly from the test set.

[6] If a reply is too short or turns up frequently, it would be annotated as either 0 or 1 (if the annotator thought the reply related to the post), like ”Me too” and ”I think so”.

Post	Emotion	Chinese	English(translated)
受当局追捕，我们只能秘密活动。 Hunted by the authorities, we work in secret.	Disgust	有一种被**嘲讽**的**感觉**。	There is a **sense** of being **mocked**.
我一直在观察你们的制作过程。 I have been observing your production process.	Happy	我**希望**你继续**努力**	I **hope** you keep **working hard**
假如你那天阻止了他，事情就不会这样了。 If you had stopped him that day, things would have been different.	Sad	不要**回忆**过去的**日子**了！	Don’t **look back** on the **past**!
普通民众中也可以拥有某些权利的概念。 The general public can also have some concepts of rights.	Like	可以**作为**他们**生活**的一部分	This can **be** a part of their **life**
我们现在也可以离开这里了。 We can also leave here now.	Sad	**挺遗憾**的。	What a **pity**.
那个人最期待的是像你这样的美女送这个箱子。 The man is looking forward to having a beautiful girl like you to send this box.	Disgust	我**期待**的是像你这样的**流氓**	I am **looking forward to** a **hooligan** like you

Table 3: Sampled conversations with a corresponding emotion from the Chinese movie subtitle data.

and *Angry* (0.7%)). Besides, the score of *Emotion* in *Surprise* has a big difference from others. We think the reason is that the characteristic of *Surprise* overlaps with other categories that have much more data, such as *Happy*, which interferes with the learning efficiency of the approach in *Surprise*. Meanwhile, it is harder for annotators to determine which one is the right emotion.

3.5 Case Study and Error Analysis

In this section, we sampled some typical cases from a Chinese movie subtitle dataset to do a further error analysis. The cases are shown in Table 3. The post of weibo and movie subtitle are applied in different scenes to obey different distributions. The weaker correlation between training sets and test sets can present a more reliable study.

The first three conversations are positive samples and others are negative samples that have content with flaws. For the reply in the antepenultimate line, its problem is the faint emotion. Since the emotion keyword in this sentence is a polysemic word, and it expresses a meaning with no emotion here. Under diverse circumstances, a polysemic word probably have different meanings, emotional or neutral. For example, the word ”like” can be a generic word when it denotes *similar*, but it can also be an emotion word when it denotes *enjoy*. Same situation also occurs in Chinese. Besides, we notice that if the LDA model pick a meaningless topic keyword from the dictionary, our approach may have a difficulty in generating a diverse and long reply, as the reply in the penultimate line. The lack of information causes generic replies which are consisted of few words generated from the networks. The last line presents another limitation. The emotion keyword *hooligan* corresponds to the post and the topic keyword *looking forward to* is meaningful, but the combination of them, *looking forward to* a *hooligan*, does not conform to the normal logic. This situation is caused by the fact that two kinds of keywords are generated independently before decoding, and it may cause a mismatch. In the future, we will try to explore different network architectures to make keywords interact with each other during the generation.

4 Conclusion

In this paper, we proposed a novel conversation generation approach (E-SCBA) to make a more comprehensive optimization for the quality of reply, which introduces both emotion and topic knowledge into the generation. The newly designed decoder makes use of syntactic knowledge to constrain generation and ensures fluency and grammaticality of reply. Experiments show that our approach can generate replies that have rich diversity and feature both emotion and logic.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-lm: A neural language model for customizable affective text generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 634–642.

Natasha Jaques, Sara Taylor, Akane Sano, and Rosalind Picard. 2017. Multimodal autoencoder: A deep learning approach to filling in missing sensor data and enabling better mood prediction. In *Proc. International Conference on Affective Computing and Intelligent Interaction (ACII), San Antonio, Texas*.

B. Kort, R. Reilly, and R. W. Picard. 2002. An affective model of interplay between emotions and learning: reengineering educational pedagogy-building a learning companion. In *Advanced Learning Technologies, 2001. Proceedings. IEEE International Conference on*, pages 43–46.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358.

Lili Mou, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2015. Backward and forward language modeling for constrained sentence generation. *arXiv preprint arXiv:1512.06612*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Xiao Sun, Chen Zhang, Shuai Ding, and Changqin Quan. 2018. Detecting anomalous emotion through big data from social networks based on a deep learning method. *Multimedia Tools and Applications*, pages 1–22.

Sara Ann Taylor, Natasha Jaques, Ehimwenma Nosakhare, Akane Sano, and Rosalind Picard. 2017. Personalized multitask learning for predicting tomorrow's mood, stress, and health. *IEEE Transactions on Affective Computing*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, volume 17, pages 3351–3357.

L. Xu, H. Lin, Y. Pan, H. Ren, and J. Chen. 2008. Constructing the affective lexicon ontology. *Journal of the China Society for Scientific & Technical Information*.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2017. Emotional chatting machine: emotional conversation generation with internal and external memory. *arXiv preprint arXiv:1704.01074*.

Auto-Dialabel: Labeling Dialogue Data with Unsupervised Learning

Chen Shi[1] **Qi Chen**[2] **Lei Sha**[1] **Sujian Li**[1] **Xu Sun**[1]
Houfeng Wang[1] **Lintao Zhang**[2]
[1]MOE Key Lab of Computational Linguistics, Peking University, Beijing, China
`{shichen, shalei, lisujian, xusun, wanghf}@pku.edu.cn`
[2]Microsoft Research Asia, Beijing, China
`{cheqi, lintaoz}@microsoft.com`

Abstract

The lack of labeled data is one of the main challenges when building a task-oriented dialogue system. Existing dialogue datasets usually rely on human labeling, which is expensive, limited in size, and in low coverage. In this paper, we instead propose our framework *auto-dialabel* to automatically cluster the dialogue intents and slots. In this framework, we collect a set of context features, leverage an autoencoder for feature assembly, and adapt a dynamic hierarchical clustering method for intent and slot labeling. Experimental results show that our framework can promote human labeling cost to a great extent, achieve good intent clustering accuracy (84.1%), and provide reasonable and instructive slot labeling results.

1 Introduction

Building a task-oriented dialogue system is challenging. In real world, unlabeled dialogue data is usually available for companies who has interactive platform with users. Based on these unlabeled data, the well-known sequence-to-sequence framework (Sutskever et al., 2014; Cho et al., 2014) is widely used in dialogue response generation (Vinyals and Le, 2015; Sordoni et al., 2015; Shang et al., 2015), but it can not handle task-oriented scenario well since it needs accuracy instead of fluency. Generally, a task-oriented dialogue system needs to realize user's *intent*, which means *user's current goal in a dialogue session*. To fulfill this intent, the system usually needs several key information (*slot*). As shown in Figure 1, a dialogue utterance is labeled with *intent* flight, and several *slots* such as its from_location, arrive_time. Training a task-oriented dialogue system usually needs abundant such labeled data.

Existing well-known dialogue datasets are mostly human-labeled, such as ATIS (Hemphill et al., 1990), DSTC (Williams et al., 2013), Frames (El Asri et al., 2017), and the Stanford dataset (Eric et al., 2017). Human-labeled datasets are expensive to produce, limited in size, and restricted to a specific domain, which make them difficult to extend. Moreover, the intent and slot label sets are usually decided by human experience. Since we usually do not know the exact intents or slots of a new unlabeled data, the assigned label names may be subjective in some extent. To better assist the human labeling process, Wen et al. (2017) proposed an improved version of *Wizard-of-Oz* (Kelley, 1984) data collection methods, which incorporate crowdsourcing to collect domain specific data. Instead of human-labeling, Cohn et al. (1995) proposed a semisupervised framework *active learning*, which can minimize the need for human annotation in a certain extent. However, these approaches are still mostly or selectively human-labeled, and may be distracted by the disadvantages raised above.

Thus, in this paper, we propose an unsupervised labeling method to automatically cluster dialogue intents and corresponding slots. Since the intent of a dialogue utterance may depend on its topic or some frequent key words, utterances in the same intent may share similar context features. Hence, we cluster these utterances into different intents. Given a new dialogue dataset whose number and type of intents are uncharted, the clustering process does not need any prior information and can derive a new set of intents. We modify dynamic clustering methods to automatically decide the number of clustering classes. The clustered intents are labeled as integer indices. Before clustering, for better utilization of extracted features, we leverage an autoencoder to map all features into the same space. For the slot labeling, since the phrases of same slots such as location and time may share same type of features, we leverage a similar way to cluster slots within each intent.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 684–689
Brussels, Belgium, October 31 - November 4, 2018.

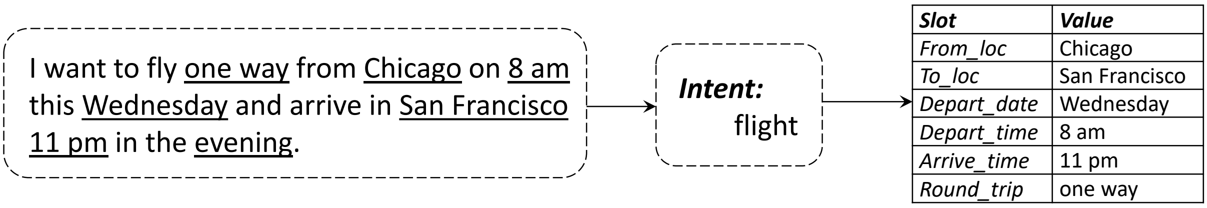

I want to fly one way from Chicago on 8 am this Wednesday and arrive in San Francisco 11 pm in the evening.

Slot	Value
From_loc	Chicago
To_loc	San Francisco
Depart_date	Wednesday
Depart_time	8 am
Arrive_time	11 pm
Round_trip	one way

Figure 1: An example of the dialogue labeling task.

Experimental results on the ATIS dataset (Hemphill et al., 1990) show that our proposed methods can achieve 84.1% intent clustering accuracy, and provide reasonable and instructive slot labeling results. Moreover, since the whole process is unsupervised, it can be much faster and more consistent and objective than human labeling, and extended to other domains. We think the proposed methods can be a good attempt for the automatic dialogue labeling task.

2 Auto-Dialabel Framework

Formally, we treat a multi-turn dialogue session $\mathcal{D} = \{\mathcal{Q}_1, \mathcal{R}_1, \cdots, \mathcal{Q}_\mathcal{N}, \mathcal{R}_\mathcal{N}\}$ as a sequence of $\mathcal{N}$ query-response pairs between two interlocutors, in which *query* $\mathcal{Q}_*$ represents user's utterance and *response* $\mathcal{R}_*$ represents assistance's utterance. Each query utterance $\mathcal{Q}_t$ should be labeled with an intent $\mathcal{I}_t \in [0, \mathcal{K}]$, which represents the interlocutor's purpose in current utterance. $\mathcal{K}$ is the number of intent classes, and should be dynamically decided during the labeling procedure.

Since each intent has its corresponding slots, for intent $\mathcal{I}_t$, we set its slot as $\mathcal{S}_t = \{\mathcal{S}_{t,1}, \cdots, \mathcal{S}_{t,L_t}\}$, where $\mathcal{L}_t$ is number of slots in $\mathcal{I}_t$. $\mathcal{S}_t$ is also learned automatically and dynamically.

In detail, given a set of query utterances, the unsupervised dialogue auto-labeling system labels the intents and slots based on the following steps:

feature extraction and assembly, which extracts a set of context features $\mathcal{F}$ from query utterance, and leverages an autoencoder to compress each extracted feature into same size, then concatenate them as the assembled feature embedding $\mathcal{E}$.

intent clustering, which adopt dynamic hierarchical clustering to get intents based on $\mathcal{E}$.

slot clustering, which leverages the same clustering methods to get slots based on word-level features and labeled intents.

The process of intent labeling is shown in Figure 2. Slot labeling has a similar process to intent labeling. Features are a key to both intent labeling and slot labeling. We first introduce the feature extraction and assembly, which involves all the features used in our model. It is noted that we leverage all the features for intent clustering while we only use word-level features for slot labeling.

2.1 Feature Extraction and Assembly

We design a set of context features $\mathcal{F}_*$ at different levels of granularity to model the query utterance, including *word embedding*, *POS tag*, *frequent key words*, and *topic features*.

Word Embedding Given an n-words $\mathcal{Q} = \{w_1, \cdots, w_n\}$, intuitively, the feature of $\mathcal{Q}$ relied on each words within it. One frequently-used way to model a sentence by words is to use a mean pooling for all word embeddings: $\mathcal{F}_\mathcal{W} = \frac{1}{n}\sum_i^n \textit{embedding of } w_i$

POS Tag Since the distribution of the POS tag may effect the sentence's structure in syntactic level, we use *bag-of-POS* as the POS tag feature $\mathcal{F}_\mathcal{P}$. Given n_p types of POS tags, $\mathcal{F}_\mathcal{P} = \{p_1, \cdots, p_{n_p}\}$ is a discrete vector in which each dimension p_i represents the existence of a POS tag POS_i.

Frequent Key Words In several occasions, the intent of a sentence is decided by some *key words*. So we specially emphasize it by introducing the frequent key words feature $\mathcal{F}_\mathcal{X}$. $\mathcal{F}_\mathcal{X} = \{x_1, \cdots, x_c\}$ represents the information of key words in query utterance. To centralize the word information, we cluster all the noun words into c different classes by its word embedding, then count the occurrence frequency of each class as a discrete vector $\mathcal{F}_\mathcal{X}$.

Topic The topic-level feature denotes the topic information of query utterance. We leverage an unsupervised topic model to get the topic distribution $\mathcal{F}_\mathcal{T} \in \mathbb{R}^t$ as the topic-level feature, t is the number of topics. Since query utterance are short

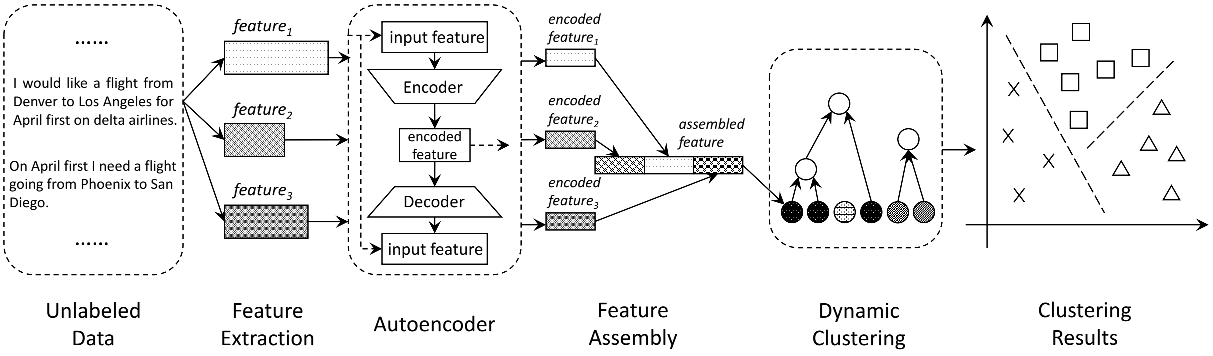

Figure 2: An illustration of our proposed auto-dialabel framework for intent labeling.

texts, while conventional topic models such as LDA and PLSA depends on document-level word co-occurrence patterns to detect topics, which may suffer from data sparsity, thus directly applying those models may not work well. In this paper, we leverage the biterm topic model (BTM) proposed by Yan et al. (2013) for better performance.

The assembled feature embedding $\mathcal{E}$ is the combination of each $\mathcal{F}_i$. Since each $\mathcal{F}_i$ has different dimensions, which may unequally affect the clustering results, we use an autoencoder to encode all $\mathcal{F}_i$ into same dimensions as $\mathcal{E}_i$, then we concatenate all the $\mathcal{E}_i$ as $\mathcal{E}$, which will be used in the clustering procedure, as shown in Figure 2.

2.2 Intent Clustering

Since given a new set of dialogue data, we do not know the number of intents it contains, thus we adapt the hierarchical clustering method to a *dynamic* version which can automatically decide the end of the clustering process by the cohesion of different classes. At the beginning of clustering, each query utterance is considered as a different class. At each steps, the cluster model chooses two classes which are the closest in distance, and cluster them into the same class. We use radial basis function (rbf) as the clustering distance. This process ends when all the distances exceed the threshold value, which is tuned on a labeled dataset, and fixed for future use.

2.3 Slot Clustering

Since slots usually correspond to intents, we do slot labeling based on both the query utterance and the labeled intents. Considering that most slots are composed of noun words, we extract all the noun words in a dialogue, and leverage the same clustering methods as in the intent clustering part to cluster them into different slot classes. Note that the slot clustering are word-level. So in *feature extraction*, it did not extract the topic-level features.

3 Experiments

3.1 Dataset and Baseline Systems

We conduct experiments on the widely used ATIS dataset (Hemphill et al., 1990). The clustering parameters are tuned on the training set of the ATIS dataset (Hemphill et al., 1990), while the experiments are conducted on its test set. During clustering, we tune the clustering distance limit since it may be more general than the number of classes, and can be transferred to other datasets. The radial basis function (rbf) is used with sklearn default settings.

Since there is no existing systems specially designed for unsupervised dialogue labeling, we choose three well-known and widely used sentence representation methods, and leverage the results vector for clustering as our baseline systems. The first one is the BTM topic model (Yan et al., 2013). We use the topic distribution for clustering. The second one is the CDSSM model proposed by Shen et al. (2014). We use the clickthrough data for pre-training and get encoded sentence vector for clustering. The third one is the sentence embedding calculated by the average of word embedding in query utterance. For each baseline, we leverage the k-means for clustering and use the gold intent number as the cluster number of these models.

3.2 Intent Labeling

We leverage glove.6B.300d (Pennington et al., 2014) as the pre-trained word embedding in all the baseline and proposed systems. The POS tag is labeled through NLTK toolkit (Bird et al., 2009).

Models	Intent Labeling Acc (%)
topic model	25.4
CDSSM vector	20.7
glove embedding	25.6
auto-dialabel	**84.1**

Table 1: Intent labeling accuracy.

The topic number of BTM is set to 20 as default. Each encoded feature dimension is set to 30, then concatenated to a 120 dimension assembled vector. We have 17 kinds of gold intents, and our system predicted 18 kinds of intents. Since the clustering results have no label information, we sort the predicted intent classes and gold intent classes by size, and manually map them. We leverage intent labeling accuracy as the evaluation metrics.

3.2.1 Overall Performance

Our auto-dialabel can reduce the tedious work of understanding the intent of each dialogue utterance and finding slot words in it. It clusters all utterances into several intent classes, and words into slot classes, which leaves human labelers only small labors to set label names for each class. Experimental results show that with auto-dialabel, we can label the whole ATIS dataset in less than 1 hour from end to end, compared to days we spend by human labeling. Table 1 shows the performance comparison of our model with other baseline systems. From Table 1, we find that our proposed auto-dialabel achieves high intent labeling accuracy (84.1%) and outperforms other baseline systems by a large margin. This may be because that the baseline systems are not specifically designed for intent scenario so that they can not handle the intent and slot clustering well, or are not capable of capturing complex intent relevant information.

3.2.2 Ablation Tests

Feature Extraction The feature extraction part extracts 4 kinds of features. We conduct ablation tests including and excluding each kind of features to see the performance. The results are shown in Table 2. From Table 2, we find the frequent key information as the most important feature. Since the intent is usually influenced by key words occurring in utterance, these kind of features can better capture key information which contributes most to the intent detection. On the contrary, the topic feature is less useful. Since it is not designed for this task, the information it represents may not directly assist the detection of intents.

Model	Acc	Model	Acc	Model	Acc
$\mathcal{F}_{\mathcal{T}}$	50.8	$-\mathcal{F}_{\mathcal{T}}$	78.9	*no_encode*	32.5
$\mathcal{F}_{\mathcal{X}}$	78.0	$-\mathcal{F}_{\mathcal{X}}$	77.5	*K-means*	28.5
$\mathcal{F}_{\mathcal{W}}$	75.9	$-\mathcal{F}_{\mathcal{W}}$	80.7	*Spectral*	40.7
$\mathcal{F}_{\mathcal{P}}$	70.2	$-\mathcal{F}_{\mathcal{P}}$	78.3	***auto-dialabel***	**84.1**

Table 2: Intent labeling ablation tests. $\mathcal{F}_*$ shows the performance of the system with only feature $\mathcal{F}_*$. $-\mathcal{F}_*$ shows the performance of the system excluded feature $\mathcal{F}_*$. *no_encode* shows the performance of the system excluded the autoencoder parts.

Intent	Slot
flight	*period_of_day* : noon, evening *month_name* : november, april *day_name* : monday, sunday *city_name* : cleveland, houston
ground_service	*period_of_day* : night, morning; *day_name* : monday; *city_name* : denver, washington

Table 3: Slot clustering cases

Feature Assembly We concatenate the originally extracted features for clustering. The results are shown in Table 2 as *no_encode*, which suggests that the encoding part is essential in feature assembly. Since each feature has a different dimension, if we alternatively concatenate them directly, the ”longer” feature may get more ”attention”, which may distract the clustering results. Generally, encoding all the features into the same dimension is an efficient way to balance the information.

Dynamic Clustering To test the performance of our modified dynamic clustering method, we leverage two most common used clustering methods for ablation test, which are k-means and spectral clustering. Both clustering numbers are set to gold intent number. The results are shown in Table 2, and we can find both methods perform worse than our methods, which shows that our methods can handle this task well. Besides, both baseline methods need prior intent number which is unavailable in advance in most cases. Compared with these baseline clustering methods, our method can dynamically determine the intent number and is more practical.

3.3 Slot Labeling

Due to the limitation of space, we just show some slot clustering result cases in Table 3. After manually assigning names, we find that auto-dialabel can extract about 70% of the slots with accuracy, including *city_name*, *period_of_day*, *month_name*, and *day_name*. The labeled slots above are the main slots for this scenario and could cover a large portion of airline ticket reservation demands. Generally, the slots clustered by auto-dialabel are reasonable and constructive.

4 Conclusion

In this paper, we formalize the auto-labeling task for dialogue data, and propose an unsupervised framework *auto-dialabel*. We design a set of linguistics and neural-network based features, leverage an autoencoder for feature assembly, and modify a hierarchical clustering method for dialogue intents and slots labeling. Experimental results show that our framework can achieve 84.1% intent clustering accuracy, and provide reasonable and instructive slot labeling results.

Acknowledgements

Our work is supported by National Natural Science Foundation of China under Grant No.61333018 and the National Key Research and Development Program of China under Grant No.2017YFB1002101. The corresponding author of this paper is Houfeng Wang.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1995. Active learning with statistical models. In *Advances in neural information processing systems*, pages 705–712.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 207–219.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

John F Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, Beijing, China. Association for Computational Linguistics.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NIPS)*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *Computer Science*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 438–449.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456. ACM.

Extending Neural Generative Conversational Model using External Knowledge Sources

Prasanna Parthasarathi
McGill University, Canada
pparth2@cs.mcgill.ca

Joelle Pineau
McGill University, Canada
Facebook AI Research, Canada
jpineau@cs.mcgill.ca

Abstract

The use of connectionist approaches in conversational agents has been progressing rapidly due to the availability of large corpora. However current generative dialogue models often lack coherence and are content poor. This work proposes an architecture to incorporate unstructured knowledge sources to enhance the next utterance prediction in chit-chat type of generative dialogue models. We focus on Sequence-to-Sequence (Seq2Seq) conversational agents trained with the Reddit News dataset, and consider incorporating external knowledge from Wikipedia summaries as well as from the NELL knowledge base. Our experiments show faster training time and improved perplexity when leveraging external knowledge.

1 Introduction

Much of the research in dialogue systems from the last few years has focused on replacing all (or some) of its components with Deep Neural Network (DNN) architectures (Levin and Pieraccini, 1997; Dahl et al., 2012; Li et al., 2017; Serban et al., 2016a,b; Vinyals and Le, 2015). These DNN models are trained end-to-end with large corpora of human-to-human dialogues, and essentially learn to mimic human conversations.

Although these models can represent the input context, the need for a dedicated external memory to remember information in context was pointed out and mechanisms were introduced in models like Memory Networks (Sukhbaatar et al., 2015; Bordes et al., 2016; Gulcehre et al., 2018), and the Neural Turing Machine (Graves et al., 2014). Although these models, in theory, are better at maintaining the *state* using their memory component, they require longer training time and excessive search for hyperparameters.

In this paper we explore the possibility of incorporating external information in dialogue systems as a mechanism to supplement the standard context encoding and facilitate the generation to be more specific with faster learning time. Furthermore, especially in the case of chit-chat systems, knowledge can be leveraged from different topics (education, sports, news, travel, etc.). Current memory-based architectures cannot efficiently handle access to large unstructured external knowledge sources.

In this work, we build on the popular Encoder-Decoder model, and incorporate external knowledge as an additional continuous vector. The proposed Extended Encoder-Decoder (Ext-ED) architecture learns to predict the embedding of the relevant external context during training and, during testing, uses an augmented *state* of external context and encoder final state to generate the next utterance. We evaluate our model with experiments on Reddit News dataset, and consider using either the Wikipedia summary (Scheepers, 2017) or the NELL KB (Carlson et al., 2010) as a source of external context.

2 Related Work

Incorporating supplemental knowledge in neural conversational agents has been addressed in a few recent works on dialogue systems. Previous research however was mostly in the context of goal-driven dialogue systems, where the knowledge-base is highly structured, and queried to obtain very specific information (e.g. bus schedules, restaurant information, more broad tourist information).

Few goal-oriented dialogues research use external information directly from the web or relevant data sources. An exception is (Long et al., 2017), which searches the web for relevant infor-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 690–695
Brussels, Belgium, October 31 - November 4, 2018.

mation that pertains to the input context and provides them as suggestions (like advising on places to visit while the user intends to visit a city). Similarly, instead of dynamically querying the web, (Ghazvininejad et al., 2017) pre-trains the model from facts in Foursquare (Yang et al., 2013) to select relevant facts as suggestions.

Moving closer to unstructured domains, but still within task-driven conversations, (Lowe et al., 2015) proposes a way of retrieving relevant responses based on external knowledge sources. The model selects relevant knowledge from Ubuntu man pages and uses it to retrieve a relevant context-response pair that is inline with the knowledge extracted. Similarly, (Young et al., 2017) extracts relations within the context and parses over it to score the message response pairs. The relational knowledge provides a way of incorporating useful knowledge or *common sense* as termed by the authors.

Though (Guu et al., 2017) did not directly make use of an external knowledge source, they used an *edit vector* that aids in editing a sampled prototype sentence. This is relevant to our proposed model as the generated response is conditioned on a supplementary vector similar to the external context vector discussed later in this paper.

3 Technical Background

3.1 Recurrent Neural Networks

The Recurrent Neural Network (RNN) is a variant of neural network used for learning representations of inputs, $\boldsymbol{x_{1:T}}$, that have an inherent sequential structure (speech, video, dialogue etc.). In natural language processing, RNNs are used to learn language models that generalize over n-gram models (Katz, 1987). The RNN maintains a hidden state, h_t, that is an abstraction of inputs observed until time-step t of the input sequence, and uses x_t to operate on them. RNN uses two projection functions, U and W, for computing operations on input and hidden states respectively. A third function, V, to map h_t to the output, y_t, the output of the RNN at every time step t. y_t, is a distribution over the next token given the previous tokens, and is computed as a function of h_t. The functions of the RNN can be explained as shown in Equations 1 and 2,

$$h_t = g\left(U \cdot x_t + W \cdot h_{t-1} + b\right), \tag{1}$$

$$y_t = V \cdot h_t + d, \tag{2}$$

where y_t is the output, x_t is the vector representation of input token, h_t is the internal state of the RNN at time t and g is a non-linear function (like *tanh* or *sigmoid*). RNNs are trained with Back Propagation Through Time (BPTT) (Rumelhart et al., 1988) to compute weight updates using the derivative of a loss function with respect to the parameters over all previous time-steps.

3.2 Seq2Seq Dialogue Architecture

Generative dialogue models (Sutskever et al., 2014; Serban et al., 2015) extends the language model learned by RNNs to generate natural language that are conditioned not only on the previous words generated in the response but also on a representation of the input context. The ability of such a learning module to *understand* an input sequence of words (that we call context ($\boldsymbol{c^i_{1:T}}$)) and *generate* a response $\boldsymbol{r^i_{1:T}}$ tantamount to solving the *dialogue task*.

(Vinyals and Le, 2015) first proposed a vanilla LSTM (Hochreiter and Schmidhuber, 1997) dialogue model that encodes a given context with an LSTM (*Encoder*) and feeds it to another LSTM (*Decoder*) that generates a response token-by-token. Here the choice of encoder and decoder modules can be any recurrent architectures like GRU (Cho et al., 2014), RNN, Bi-directional LSTM (Schuster and Paliwal, 1997), etc. The model is then trained to learn a conditional distribution over the vocabulary for generating the next token in response to the i^{th} context as shown in Equation 3:

$$P\left(\boldsymbol{r^i_{1:T}} \mid \boldsymbol{c^i_{1:T}}\right) = \Pi_{k=1}^{T} P\left(r^i_k \mid \boldsymbol{r^i_{1:k-1}}, \boldsymbol{c^i_{1:T}}\right). \tag{3}$$

With neural language models, this form can aid in maintaining long term dependencies and the next word distribution can be made to conditionally depend on an arbitrarily long context. Sophisticated models have made significant architectural improvements to aid better modelling of the contexts (Serban et al., 2016b).

4 Extended Encoder Decoder Architecture

The primary objective of the proposed architecture is to supplement response generation with external knowledge relevant to the context. Most of the knowledge sources that are available are free-form and lack suitable structure for easy querying of relevant information. In this work, we attempt to incorporate such unstructured knowledge corpora for dialogue generation in Seq2Seq models.

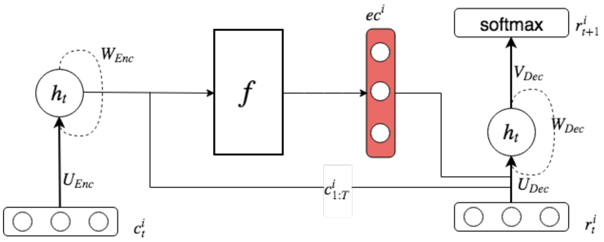

Figure 1: Architecture Diagram of Extended Encoder Decoder Model.

4.1 The Model

The Extended Encoder Decoder (Ext-ED) model, shown in Figure 1, uses an encoder LSTM (parameterized as $\boldsymbol{\Theta}_{Enc}$) to encode the i^{th} context, $\boldsymbol{c^i_{1:T}}$, and a fully connected layer (f) to predict an external context vector ($\boldsymbol{ec^i}$) conditioned on the encoded context. The predicted external context vector, $\boldsymbol{ec^i}$, is provided to the decoder LSTM ($\boldsymbol{\Theta}_{Dec}$) at every step, augmented with the encoder final state and previous predicted token, to generate the next token in the response:

$$P\left(\boldsymbol{ec^i} \mid \boldsymbol{c^i_{1:T}}\right) = f\left(\boldsymbol{\Theta_{Enc}}\left(\boldsymbol{c^i_{1:T}}\right)\right) \quad (4)$$

$$P\left(\boldsymbol{r^i_{1:T}} \mid \boldsymbol{c^i_{1:T}}\right) = \Pi_{k=1}^{T} P\left(r^i_k \mid \boldsymbol{r^i_{1:k-1}}, \boldsymbol{c^i_{1:T}}, \boldsymbol{ec^i}\right). \quad (5)$$

The decoder is provided with an encoding of the context along with the external knowledge encoding, as $\boldsymbol{ec^i}$ acts as information supplement to the knowledge available in the context as shown in Equations 4 and 5.

During training, the gradients for Ext-ED parameters ($f, \Theta_{Enc}, \Theta_{Dec}$) are computed by backpropagating the gradients for parameters with respect to losses in Equations 6, 7 and 8:

$$\mathcal{L}_1 = \sum_{k=1}^{T} Q\left(r^i \mid \hat{ec}^i, c^i\right) \log P\left(r^i \mid c^i\right), \quad (6)$$

$$\mathcal{L}_2 = \|\hat{\boldsymbol{ec}}^i - \boldsymbol{ec^i}\|_2, \quad (7)$$

$$\mathcal{L}_3 = -\sum_{k=1}^{T} Q\left(r^i \mid c^i, \mathbf{0}\right) \log P\left(r^i \mid c^i\right). \quad (8)$$

Here $\mathcal{L}_1$ is the log-likelihood that is used to make the model distribution mimic the data distribution. $\mathcal{L}_2$ trains f to correctly predict ec^i, and $\mathcal{L}_3$ trains Θ_{Dec} to make use of the external context by forcing it the model distribution to diverge when not provided with the external context (ec^i is set to $\mathbf{0}$ vector). In the loss equations, P and Q represent the data and the model learned distributions respectively, and $\boldsymbol{ec^i}$ and $\hat{\boldsymbol{ec}}^{\boldsymbol{i}}$ represent true and model(f) predicted external knowledge encoding.

4.2 External Context Vector

We use Wikipedia summary (Scheepers, 2017) and NELL knowledge base (KB) (Carlson et al., 2010) to compute the external knowledge encoding for every context in the context-response pairs. Algorithm 1 oultines the pseudocode for computing the external context vector ($\boldsymbol{ec^i}$). For i^{th} input context, the methods *Return_All_Values_for_Entity* or *Wiki_Summary_Query* is used to extract the external knowledge vector, ec^i, from NELL KB or Wikipedia summary sources.

Algorithm 1: Get_External_Context _Vector ($\boldsymbol{c^i_{1:T}}$)

```
$\boldsymbol{ec^i} \leftarrow$ zero_vector
$\#\_Ext\_Tokens \leftarrow 0$
for t in range (1,T) do
    if $c^i_t$ is not a stop word then
        External_Tokens$_{List}$ ←
          Wiki_Summary_Query($c^i_t$)
        % Return_All_Values_for_Entity($c^i_t$)
        for token in External_Tokens$_{List}$ do
            $\boldsymbol{ec^i} \leftarrow \boldsymbol{ec^i}$ +
              GloVe_Embedding(token)
            $\#\_Ext\_Tokens \leftarrow \#\_Ext\_Tokens + 1$
return $\frac{\boldsymbol{ec^i}}{\#\_Ext\_Tokens}$
```

The external context encoding, $\boldsymbol{ec^i}$, is a fixed length continuous embedding of the knowledge from external sources, as having all the words sampled (represented as a Bag of Words) proved to be a severe computational overhead because of sparsity.

The continuous embedding of external context provides an additional conditioning with relevant external knowledge that is used to generate the next utterance. Although this is the intended hypothesis, there are certain expected characteristics that are desirable of external knowledge sources for them to be useful:

- The knowledge vectors being away from the

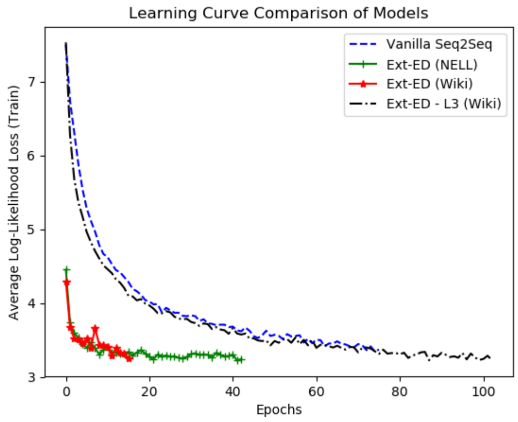

Figure 2: Convergence of Sequence loss (cross-entropy loss in sequential outputs) over different models during training.(-L3 in legend denote exclusion of $\mathcal{L}_3$ loss from gradient computation.)

mean of their distribution.

- The knowledge vectors having high variance.

We analyzed the knowledge vectors constructed using the two knowledge sources and the distribution of distances of $\boldsymbol{ec^i}$ from their mean. The variance of this distribution in NELL KB was *1.44* and from that of Wikipedia summary was *0.73*. Mean distance was very low (*0.77*) in the case of Wikipedia compared to that of NELL (*2.16*). We observed that the vectors not being spread out made them less useful than the encoded context itself in the initial experiments.

5 Experiments and Discussion

We evaluated Ext-ED by training with Reddit News dataset, and incorporated Wikipedia Summary and NELL KB as sources of external knowledge. The objective of the experiments are threefold: (1) to evaluate the ability of the model to make use of the external context to condition the response; (2) to analyze the training time with the additional knowledge provided; and (3) to observe any tangible differences in training with the two knowledge sources.

For the first analysis, we trained Ext-ED with external context ($\mathbb{R}^{100}$) and validated it without providing it (see Ext-ED - $\mathcal{L}_3$ Ablation in Table 1). Without the inclusion of $\mathcal{L}_3$, Ext-ED did not find the external context useful and the performance was not very different from a Vanilla Seq2Seq dialogue model (Figure 2). The encoder context had enough variance to be a viable information source and hence the external context was ignored. This can be observed from similar learning curves of *Vanilla Seq2Seq, Ext-ED - L3 (Wiki)* models in Figure 2.

With propagating back gradients with respect to $\mathcal{L}_3$, we observed that the model learns to use the external context, but, as discussed in Section 4.2, the variance in the external context vectors constructed using the two knowledge sources was too low. To fix this, we scaled the external context vectors with $\mathcal{N}(4, 1)$. This improved the variance in the knowledge that subsequently improved the usefulness of these vectors which was also observed in Figure 2.

Model	PPL	BLEU-4
Vanilla Seq2Seq	38.09	0.437
Ext-ED - $\mathcal{L}_3$ (Wiki)	38.37	0.435
Ext-ED - $\mathcal{L}_3$ Ablation	37.06	0.425
Ext-ED (Wiki)	**30.26**	**0.53**
Ext-ED (NELL KB)	**29.07**	**0.525**
Ext-ED Ablation	601.8	0.274

Table 1: Comparison of BLEU (sentence_bleu) and Perplexity scores on validation set across different models.

The provision of external knowledge improved the Perplexity and BLEU-4 scores as shown in Table 1. Though the improvements are reasonable, the metrics used are not strong indicators for evaluating the influence of external contexts in dialogue. But, they do indicate that the prediction accuracy is improved with the inclusion of external knowledge sources. The poor perplexity for Ext-ED Ablation is because the model is conditioned to predict the next utterance using $\boldsymbol{ec^i}$ and when not provided the context alone is not sufficiently informative. Another way to interpret this would be to see that the external context and and the dialogue context provide complementary information for better predicting the next utterance. Further, the experiments illustrated that the model, when provided with an *informative* source of knowledge (the one that has higher variance), will let the model converge faster. One possible hypothesis is that $\boldsymbol{ec^i}$, which has high variance and is provided as input in every step of decoding, is learned before the RNN parameters converge. The information in $\boldsymbol{ec^i}$ is relevant to the context and subsequently helps in training the decoder faster.

6 Conclusion

The proposed model, Extended Encoder Decoder, offers a framework to incorporate unstructured external knowledge to generate dialogue utterances. The experiments helped in understanding the need for external knowledge sources for improving learning time, and helped characterize the value of external knowledge sources. The experiments showed that external knowledge improved the learning time of the models. In future work, we aim to add more experiments with dialogue tasks that require understanding a supplementary source of knowledge to solve the task. Also, we plan to look at specialized tasks that naturally evaluate the influence of external knowledge, to help the model to generate diverse responses.

Acknowledgements

The authors gratefully acknowledge financial support for portions of this work by the Samsung Advanced Institute of Technology (SAIT) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. In *arXiv preprint*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, and Burr Settles. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *arXiv preprint*.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. In *IEEE Transactions on audio, speech, and language processing*.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A knowledge-grounded neural conversation model. In *arXiv preprint*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. In *arXiv preprint*.

Caglar Gulcehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. 2018. Dynamic neural turing machine with continuous and discrete addressing schemes. In *Neural computation*. MIT Press.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2017. Generating sentences by editing prototypes. In *arXiv preprint*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Neural computation*.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE transactions on acoustics, speech, and signal processing*.

Esther Levin and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *Fifth European Conference on Speech Communication and Technology*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *arXiv preprint*.

Yinong Long, Jianan Wang, Zhen Xu, Zongsheng Wang, Baoxun Wang, and Zhuoran Wang. 2017. A knowledge enhanced generative conversational service agent. In *the 6th Dialog System Technology Challenges (DSTC6) Workshop*.

Ryan Lowe, Nissan Pow, Iulian Serban, Laurent Charlin, and Joelle Pineau. 2015. Incorporating unstructured textual knowledge sources into neural dialogue systems. In *Neural Information Processing Systems Workshop on Machine Learning for Spoken Language Understanding*.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. 1988. Learning representations by back-propagating errors. In *Cognitive modeling*.

Thijs Scheepers. 2017. Improving the compositionality of word embeddings. Master's thesis, Universiteit van Amsterdam.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. In *IEEE Transactions on Signal Processing*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2015. A survey of available corpora for building data-driven dialogue systems. In *arXiv preprint*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016a. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *arXiv preprint*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model.

Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhiwen Yu. 2013. Fine-grained preference-aware location search leveraging crowdsourced digital footprints from lbsns. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*.

Tom Young, Erik Cambria, Iti Chaturvedi, Minlie Huang, Hao Zhou, and Subham Biswas. 2017. Augmenting end-to-end dialog systems with commonsense knowledge. In *arXiv preprint*.

Modeling Temporality of Human Intentions by Domain Adaptation

Xiaolei Huang[1], Lixing Liu[2], Kate B. Carey[3], Joshua Woolley[4],
Stefan Scherer[2], Brian Borsari[4]
[1]Dept. of Information Science, University of Colorado Boulder
[2]Inst. for Creative Technologies, University of Southern California
[3]Dept. of Behavioral and Social Sciences, Brown University
[4] Dept. of Psychiatry, University of California, San Francisco
xiaolei.huang@colorado.edu, {lxliu, scherer}@ict.usc.edu,
kate_carey@brown.edu, {josh.woolley, brian.borsari}@ucsf.edu

Abstract

Categorizing a patient's intentions during clinical interactions in general and within motivational interviewing specifically may improve decision making in clinical treatments. Within this paper, we propose a method that models the temporal flow of a conversation and the transition between topics by using domain adaptation on a clinical dialogue corpus comprising Motivational Interviewing (MI) sessions. We deploy Bi-LSTM and topic models jointly to learn theme shifts across different time stages within these hour-long MI sessions to assess the patient's intent to change their habits or to sustain them respectively. Our experiments show promising results and improvements after considering temporality in the classification task over our baseline. This result confirms and extends related literature that has manually identified that certain phases within MI sessions are more predictive of patient outcomes than others.

1 Introduction

Motivational Interviewing (MI) (Miller and Rollnick, 2012) is a collaborative communication style used to address a variety of health problems such as alcohol and drug use. Accurately understanding the patient's intentions to change from his/her speech during the session could greatly enhance the efficacy of MI. Motivational Interviewing Skill Code (MISC) is a coding system that captures client language, specifically change talk (CT) and sustain talk (ST) (Miller et al., 2008). However, reliable MISC coding is labor-intensive and requires domain expertise. Recent computational annotation methods have been proposed to automatically classify patients' behaviors within MI (Xiao et al., 2016; Pérez-Rosas et al., 2017; Gibson et al., 2017). To this end, Recurrent Neural Networks (RNN) that capture sequential information are applied for the classification of patient behavior.

Recent research shows that themes and words within a conversation change across time (Dufour et al., 2016). Similarly within MI, topics and the patient's attitude towards their willingness to change might shift. Within this work, we investigate how shifts in themes across time affects performance of the intention classifications for the dialogue.

Specifically, we focus on the patient intent classification task and propose a method that adopts the temporal factor by domain adaptation to improve performance of the classifiers. We evaluate our approach on a dataset of college alcoholism (Carey et al., 2009; Borsari et al., 2012), containing transcripts of MI conducted with U.S. college students. Specifically, we first explore the theme shift and give a brief analysis by topic modeling (Blei et al., 2003). We then utilize Bi-directional Long Short-Term Memory (Bi-LSTM) (Graves and Schmidhuber, 2005) to encode utterances from both word and topic embeddings. Next, we concatenate both contextual information with the encoded utterance representations. Finally, we jointly train a unified representation of utterance by domain adversarial training and patient intent classification. We show that this approach can lead to improvements in classification performance.

2 Dataset

We conduct our experiments on a clinical dataset of college student alcoholism (Carey et al., 2009; Borsari et al., 2012), where we obtain 193 MI transcripts with a total of 83677 utterances. Each of the MI session ranged between 60 and 90 minutes. Each client utterance was coded using the MISC. In this paper, we focus on classifying patient behavior on the utterance-level. Specifically, we classify patient behavior on collapsed MISC

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 696–701
Brussels, Belgium, October 31 - November 4, 2018.

Table 1: Examples of utterances in Alcoholism Treatment. "I" stands for interventionist and "P" for patient. MISC codes are provided in the third column. P and I codes are coded following MISC. Change talk (CT), sustain talk (ST), and follow neutral (FN) codes are also provided.

Role	Conversation	MISC
I	Maybe you could tell me a little bit about what you do on the weekends, what your weekends have been like.	quo
P	Well we go out, but before we go out we just drink in the dorm room.	FN
I	Has this sort of changed your thinking, are things different than they were when you came in?	quc
P	I mean, I feel guilty about drinking,	CT (o+3)
I	Yeah. So it feels like, or it sounds like social social drinking is a big part of how you meet other people.	res
P	It's just, like ... and I don't mean to sound mean, but about the kids who don't drink, and people think that, "Oh, the kids who dont drink are losers".	ST (o-3)

annotation codes into with three categories: "CT": *Change talk* indicates utterances that reflect motivating factors related to change; "ST": *Sustain talk* indicate the patient has no intentions to change; "FN": *Follow neutral* means there is no indication of patient inclination. An example conversation snippet, highlighting all three sources of information is provided in Table 1. The intention labels (o+3, o-3) are only available for patients, whose '+' and '-' refer to change vs sustain talk (CT vs ST) and the number measures the "strength of client language," which represents a subjective assessment by human annotators, and the 'quo' and 'quc' refer to "open question" and "closed questions", which are only for interventionist (see (Borsari et al., 2015) for details regarding the coding strategy). While the MISC codes of client utterances within MISC are more complex and comprise other types of annotations, we focus on human intention modeling (i.e., CT vs. ST vs. FN) only.

How the theme of dialogue shift overtime?

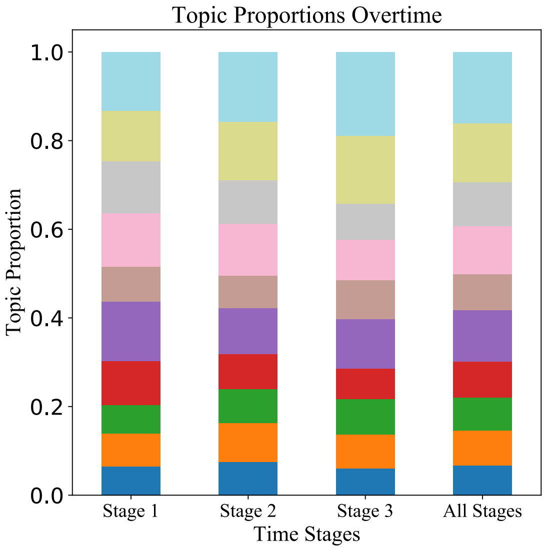

Figure 1: 10 topic proportions of patient's utterance across time stages.

We qualitatively examined how the distribution of content changes across different time stages. To measure the distribution of content, we trained a topic model with 10 topics using Gensim (Řehůřek and Sojka, 2010) with default parameters. The data doesn't have associated timestamps, thus we empirically split each MI transcript by the number of patient utterances equally into three time stages, stage 1, stage 2 and stage 3. We calculated the proportion of each topic within the same time period by take the average of all transcripts. We then normalized the topic distributions and finally visualize the extent to which distributions of the 10 topics varies by time.

We can observe the varied topic distributions across different stages of conversations, where the topic distributions are plotted from the bottom to the top. There are some topics have more variations, such as topic 4, and some topics are very stable such as topic 1 [1]. Recent research shows the performance of classification tasks might be impacted by the temporal character of language (Huang and Paul, 2018). Thus, it might be desirable to model the temporality in the computational classifiers.

---

[1] The 5 top words of topic 4 and 1: yeah, go, friends, know, people; beer, alcohol, games, meeting, playing.

3 Model

The architecture of the proposed model is shown in Figure 2. We feed four types of information to the model: topic- and word-level data of the utterance (*content*), preceding interventionist verbal behavior (*context*) and prior MISC annotations of utterances (*MISC*). Particularly, we empirically extracted previous 5 utterances as context and 10 previous codes as MISC[2], where we set “unk” as the default.

Embeddings. We built two types of embeddings, word embedding and topic embedding. We created word embeddings from Googles pretrained Word2Vec (Mikolov et al., 2013) and created topic embeddings from a trained LDA (Blei et al., 2003) specific to the corpus. We treated each MISC as one document and trained an embedding model.

Unified Representation. We apply Bidirectional LSTM (Bi-LSTM) (Graves and Schmidhuber, 2005) on the inputs. Dropouts (Srivastava et al., 2014) are applied on the outputs of Bi-LSTM. We merge the outputs by concatenation and feed the outputs to the dense layer to learn a unified representation of the utterance.

Joint Learning. We apply domain adversarial training (Ganin et al., 2016) only on the topic inputs from learned topic representations. Our intuition is that the topic distributions across different stages of the MI session could track the variations of patients’ intents. We empirically split the conversation into three time stages: Stage 1-3 (i.e, beginning, middle, and end). The goal of domain adversarial training is converted to a time stage prediction task, which aims to differentiate topic themes both locally and globally. We used one-hot encoding to represent labels of the prediction tasks. We deploy softmax functions for both time stage and intention predictions. We use categorical cross entropy to jointly optimize the training process of the two classification tasks: domain classification and patient intent classification.

4 Experiments

Each utterance is lowercased and tokenized by NLTK (Bird et al., 2009). We filter out the utterances that are shorter than 5 tokens and then remove punctuations. Finally, we obtain 22432 patient utterances. The dataset is stratified and split into training set (80%), validation set (10%) and testing set (10%), as shown in Table 2. We train our models on the training set and run grid search to find the optimal parameters on the validation set by the weighted F1 score.

Table 2: Statistics of the processed dataset.

Datasets	Train	Valid	Test	Total
CT	6246	779	768	7794
ST	3099	406	401	3906
FN	8600	1058	1074	10732
All				22432

The details of optimized parameters are listed as follows. The models were trained for 15 epochs with a batch size of 64. Each utterance and its context are padded to 50 words. The utterance’s previous MISC codes are padded to 10. We pad the sequences with an “unknown”-token. The size of LSTMs was tuned in the range of $[100, 150, 200]$ and the size of dense layer tuned within $[100, 150, 200]$. We select the activation function of the Dense layer within $\{$relu (Hahnloser et al., 2000), tanh, softplus$\}$ (Hahnloser et al., 2000). We tried different flip gradient value within $[0.05, 0.01, 0.005]$ for the domain adversarial training. We tuned the dropout rate between $[0.1, 0.2]$. The optimizer was selected either RMSprop (Hinton et al., 2012) or Adam (Kingma and Ba, 2014) with a fixed learning rate of 0.001. Finally, we empirically set the loss weight of the domain adversarial training to 0.05.

We trained the topic model on the MI corpus using Gensim (Řehůřek and Sojka, 2010). The number of topics was selected by coherence scores among 5, 10, 20 topics. We used Google pre-trained word embedding with 300 dimensions (Mikolov et al., 2013). We obtained 50-dimension code embedding by Word2vec (Mikolov et al., 2013) for the MISC codes, where each sequence of MISC were treated as a document.

We select three different approaches as our baselines with the inputs: content, context, MISC, and topic.

- (Pérez-Rosas et al., 2017) with rich linguistic features (denote as Perez2017_lin): We reproduced their method. We used scikit-learn (Buitinck et al., 2013) to ex-

---

[2] We encode 10 MISC codes prior to the current one as a sequence of 10 “words”, then we treat the sequence as an additional input document.

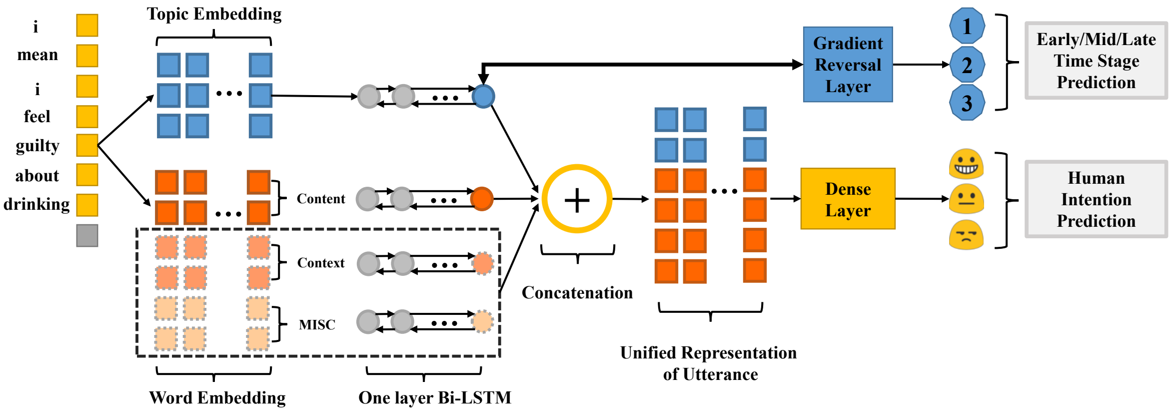

Figure 2: The proposed model learns from tow different channels: word and topic channels. The topic channel captures the theme shifts across stages while the word channel aims to capture the behavior indicators within each utterance.

tract n-gram features, and applied Stanford Parser (Manning et al., 2014) to extract Part-of-Speech (POS). We replaced LIWC (Pennebaker et al., 2001) by a free and open source lexicon (Mohammad and Turney, 2013) to extract semantic features. We use Ngram+ to denote the rich linguistic features.

- (Pérez-Rosas et al., 2017) with embeddings (denote as Perez2017_vec): Word embedding shows superior results over the n-gram features in the classification tasks (Mikolov et al., 2013) in the recent past. We experiment feeding the classifier with word vectors while we keep the same parameter settings as the Perez2017_lin baseline. We deploy the strategy of concatenating word embeddings to build representations of utterances, which is denoted as "Vec-con".
- (Xiao et al., 2016) (denoted as Xiao2016): Their approach applies Bi-directional RNN to encode each utterance by both the utterance itself and its preceding one. There are two major differences between their method and ours: first, they did not consider temporality in their model, second, they did not use the previous MISC sequences as inputs. They used Gated Recurrent Unit (GRU) (Chung et al., 2014) as the RNN cell.

We use the "Co", "Ct", "MISC" to denote the utterance (*content*), preceding interventionist verbal behavior (*context*) and prior MISC annotations of utterances (*MISC*) respectively. And we use "All" to denote all of the inputs[3]. We use the "T" to denote temporal shifts proposed in our paper. We balance training weights for the classification labels. We use metrics from scikit-learn (Buitinck et al., 2013) to evaluate the classification performance by precision, recall and weighted F1 on the intention labels.

4.1 Results and Discussion

Table 3: Results of classification evaluations.

Features	Method	Precision	Recall	F1
	Perez2017_lin			
Ngram+	Co	0.62	0.62	0.62
	Co+Ct	0.60	0.61	0.61
	All	0.61	0.61	0.62
	Perez2017_vec			
Vec-con	Co	0.60	0.58	0.59
	Co+Ct	0.61	0.59	0.60
	All	0.61	0.57	0.58
	Xiao2016			
Vec	Co	0.65	0.63	0.64
	Co+Ct	0.68	0.64	0.65
	All	0.67	0.67	0.67
	Proposed model			
Vec	Co+T	0.65	0.64	0.65
	Co+Ct+T	0.70	0.66	0.68
	All+T	0.74	0.67	0.70

The results of our experiments are summarized in the Table 3. Findings indicate that our proposed approach leads to a small performance boost after using the topic embeddings. Thus, our sim-

---

[3] The baselines did not use one or more inputs (the context and MISC) in the original publications. We used different combinations for fair comparison.

ple feature augmentation approach has the potential to make classifiers more robust. In addition, the contextual information (“Ct”) is quite useful to identify the patients’ current intentions, and the sequential information through time stages has strong indications of human intentions.

Significance Analysis

We conducted significance analysis to compare Xiao2016 and our proposed method. Because Xiao2016 only used content and context inputs, in this analysis, we train our method with the same inputs (Co+Ct). We followed the method of bootstrap samples (Berg-Kirkpatrick et al., 2012) to create 50 pairs of training and test datasets with replacement, where we keep the sizes the same in the Table 2. We keep the same experimental steps and use the parameters that achieved the best performances in the Table 3 to train the models.

To compare the two approaches, we conduct a paired t-test comparing the achieved F1 scores of both models. We used a two-tail test instead of one tail test used in the paper due to its increased rigor and lack of prior assumptions (Dror et al., 2018). The test reveals a significant result with $t(85) = 3.084$ and $p = 0.00275$. The result shows that we can reject the null hypothesis that our proposed method is not better than Xiao2016.

5 Conclusion

In this paper, we focus on the temporal characteristics of the MI corpus and propose a simple method that models the temporal factor within a single MI session. We jointly learn the utterance representation via time stage and intention predictions and the proposed model improves the performance of the classification task. The identified intent of clients could help therapists adjust their treatment strategy. In future work, we will investigate other external sources of knowledge, such as acoustic cues and videos to further improve the performance of the model.

6 Acknowledgements

We thank the anonymous reviewers for their constructive comments. Partial work done while the first author was an summer intern at ICT, USC. The idea of modeling temporal factor was inspired by the paper (Huang and Paul, 2018), co-authored with Michael J. Paul. This work was supported by National Institute on Alcohol Abuse and Alcoholism grants R01 AA015518 and R01 AA017427 to B. Borsari, and R01 AA012518 to K. Carey. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute on Alcohol Abuse and Alcoholism or the National Institutes of Health, or the Department of Veterans Affairs or the United States Government. The authors would like to thank the students and therapists who allowed their audiotapes to be utilized for this study.

References

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in nlp. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O’Reilly Media, Inc.".

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Brian Borsari, Timothy R Apodaca, Kristina M Jackson, Nadine R Mastroleo, Molly Magill, Nancy P Barnett, and Kate B Carey. 2015. In-session processes of brief motivational interventions in two trials with mandated college students. *Journal of consulting and clinical psychology*, 83(1):56.

Brian Borsari, John TP Hustad, Nadine R Mastroleo, Tracy O’Leary Tevyaw, Nancy P Barnett, Christopher W Kahler, Erica Eaton Short, and Peter M Monti. 2012. Addressing alcohol use and problems in mandated college students: a randomized clinical trial using stepped care. *Journal of consulting and clinical psychology*, 80(6):1062.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Kate B Carey, James M Henson, Michael P Carey, and Stephen A Maisto. 2009. Computer versus in-person intervention for students violating campus alcohol policy. *Journal of consulting and clinical psychology*, 77(1):74.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhikers guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1383–1392.

Richard Dufour, Mohamed Morchid, and Titouan Parcollet. 2016. Tracking dialog states using an author-topic based representation. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 544–551. IEEE.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

James Gibson, Dogan Can, Panayiotis Georgiou, David C Atkins, and Shrikanth S Narayanan. 2017. Attention networks for modeling behaviors in addiction counseling. In *Proc. Interspeech*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 4, pages 2047–2052. IEEE.

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Lecture 6a overview of mini-batch gradient descent.

Xiaolei Huang and Michael J Paul. 2018. Examining temporality in document classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 694–699.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The {Stanford} {CoreNLP} Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

William R Miller, Theresa B Moyers, Denise Ernst, and Paul Amrhein. 2008. Manual for the motivational interviewing skill code (misc 2.1). Unpublished manuscript. Retrieved from: `https://casaa.unm.edu/download/misc.pdf`.

William R Miller and Stephen Rollnick. 2012. Motivational interviewing: Helping people change (applications of motivational interviewing).

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

Verónica Pérez-Rosas, Rada Mihalcea, Kenneth Resnicow, Satinder Singh, Lawrence Ann, Kathy J Goggin, and Delwyn Catley. 2017. Predicting counselor behaviors in motivational interviewing encounters. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1128–1137.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Bo Xiao, Dogan Can, James Gibson, Zac E Imel, David C Atkins, Panayiotis G Georgiou, and Shrikanth S Narayanan. 2016. Behavioral coding of therapist language in addiction counseling using recurrent neural networks. In *Interspeech*, pages 908–912.

An Auto-Encoder Matching Model for Learning Utterance-Level Semantic Dependency in Dialogue Generation

Liangchen Luo*, **Jingjing Xu***, **Junyang Lin**, **Qi Zeng**, **Xu Sun**
MOE Key Lab of Computational Linguistics, School of EECS, Peking University
{luolc,jingjingxu,linjunyang,pkuzengqi,xusun}@pku.edu.cn

Abstract

Generating semantically coherent responses is still a major challenge in dialogue generation. Different from conventional text generation tasks, the mapping between inputs and responses in conversations is more complicated, which highly demands the understanding of utterance-level semantic dependency, a relation between the whole meanings of inputs and outputs. To address this problem, we propose an Auto-Encoder Matching (AEM) model to learn such dependency. The model contains two auto-encoders and one mapping module. The auto-encoders learn the semantic representations of inputs and responses, and the mapping module learns to connect the utterance-level representations. Experimental results from automatic and human evaluations demonstrate that our model is capable of generating responses of high coherence and fluency compared to baseline models.[1]

1 Introduction

Automatic dialogue generation task is of great importance to many applications, ranging from open-domain chatbots (Higashinaka et al., 2014; Vinyals and Le, 2015; Li et al., 2016, 2017a; Su et al., 2018) to goal-oriented technical support agents (Bordes and Weston, 2016; Zhou et al., 2017; Asri et al., 2017). Recently there is an increasing amount of studies about purely data-driven dialogue models, which learn from large corpora of human conversations without hand-crafted rules or templates. Most of them are based on the sequence-to-sequence (Seq2Seq) framework (Sutskever et al., 2014) that maximizes the probability of gold responses given the previous dialogue turn. Although such methods offer great promise for generating fluent responses, they still suffer from the poor semantic relevance between inputs and responses (Xu et al., 2018). For example, given "What's your name" as the input, the models generate "I like it" as the output.

Recently, the neural attention mechanism (Luong et al., 2015; Vaswani et al., 2017) has been proved successful in many tasks including neural machine translation (Ma et al., 2018b) and abstractive summarization (Lin et al., 2018), for its ability of capturing word-level dependency by associating a generated word with relevant words in the source-side context. Recent studies (Mei et al., 2017; Serban et al., 2017) have applied the attention mechanism to dialogue generation to improve the dialogue coherence. However, conversation generation is a much more complex and flexible task as there are less "word-to-words" relations between inputs and responses. For example, given *"Try not to take on more than you can handle"* as the input and *"You are right"* as the response, each response word can not find any aligned words from the input. In fact, this task requires the model to understand the utterance-level dependency, a relation between the whole meanings of inputs and outputs. Due to the lack of utterance-level semantic dependency, the conventional attention-based methods that simply capture the word-level dependency achieve less satisfying performance in generating high-quality responses.

To address this problem, we propose a novel **Auto-Encoder Matching** model to learn utterance-level dependency. First, motivated by Ma et al. (2018a), we use two auto-encoders to learn the semantic representations of inputs and responses in an unsupervised style. Second, given the utterance-level representations, the mapping module is taught to learn the utterance-level dependency. The advantage is that by explicitly separating representation learning and dependency

---

*Equal Contribution

[1] The code is available at `https://github.com/lancopku/AMM`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 702–707
Brussels, Belgium, October 31 - November 4, 2018.

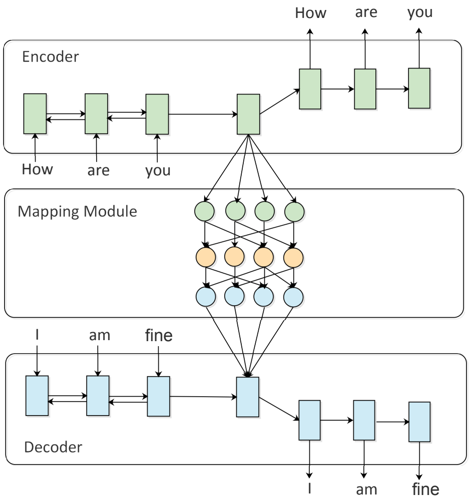

Figure 1: An illustration of the Auto-Encoder Matching model. The encoder and decoder are two auto-encoders that are responsible for learning the semantic representations. The mapping module is responsible for learning the utterance-level dependency.

learning, the model has a stronger modeling ability compared to traditional Seq2Seq models. Experimental results show that our model substantially outperforms baseline methods in generating high-quality responses.

Our contributions are listed as follows:

- To promote coherence in dialogue generation, we propose a novel Auto-Encoder Matching model to learn the utterance-level dependency.
- In our proposed model, we explicitly separate utterance representation learning and dependency learning for a better expressive ability.
- Experimental results on automatic evaluation and human evaluation show that our model can generate much more coherent text compared to baseline models.

2 Approach

In this section, we introduce our proposed model. An overview is presented in Section 2.1. The details of the modules are shown in Sections 2.2, 2.3 and 2.4. The training method is introduced in Section 2.5.

2.1 Overview

The proposed model contains three modules: an encoder, a decoder, and a mapping module, as shown in Figure 1.

In general, our model is different from the conventional sequence-to-sequence models. The encoder and decoder are both implemented as auto-encoders (Baldi, 2012). They learn the internal representations of inputs and target responses, respectively. In addition, a mapping module is built to map the internal representations of the input and the response.

2.2 Encoder

The encoder E_θ is an unsupervised auto-encoder based on Long Short Term Memory Networks (LSTM) (Hochreiter and Schmidhuber, 1996). As it is essentially a LSTM-based Seq2Seq model, we name the encoder and decoder of the auto-encoder "source-encoder" and "source-decoder". To be specific, the encoder E_θ receives the source text $x = \{x_1, x_2, ..., x_n\}$, and encodes it to an internal representation h, and then decodes h to a new sequence $\tilde{x} = \{\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n\}$ for the reconstruction of the input. We extract the hidden state h as the semantic representation. The encoder E_θ is trained to reduce the reconstruction loss, whose loss function is defined as follows:

$$J_1(\theta) = -\log P(\tilde{x}|x;\theta) \qquad (1)$$

where θ refers to the parameters of the encoder E_θ.

2.3 Decoder

Similar to the encoder, our decoder D_ϕ is also a LSTM-based auto-encoder. However, as there is no target text provided in the testing stage, we propose the customized implementation, which is illustrated in Section 2.5. Here in the introduction of the decoder, we do not provide the testing details. Similarly, we name the encoder and decoder of the auto-encoder "target-encoder" and "target-decoder". The target-encoder receives the target $y = \{y_1, y_2, ..., y_n\}$ and encodes it to a utterance-level semantic representation s, and then decodes s to a new sequence to approximate the target text. The loss function is identical to that of the encoder:

$$J_2(\phi) = -\log P(\tilde{y}|y;\phi) \qquad (2)$$

2.4 Mapping Module

As our model is constructed for dialogue generation, we design the mapping module to ensure that the generated response is semantically consistent with the source. There are many matching models that can be used to learn such dependency relations (Hu et al., 2014; Guo et al., 2016; Pang et al., 2016; Chen et al., 2018). For simplicity, we only use a simple feedforward network for implementation. The mapping module M_γ transforms the source semantic representation h to a new representation t. To be specific, we implement a multi-layer perceptron (MLP) $g(\cdot)$ for M_γ and train it by minimizing the L2-norm loss $J_3(\gamma)$ of the transformed representation t and the semantic representation of target response s:

$$\begin{aligned} t &= g(h) \\ J_3(\gamma) &= \frac{1}{2}\|t - s\|_2^2 \end{aligned} \tag{3}$$

2.5 Training and Testing

In the testing stage, given an input utterance, the encoder E_θ, the decoder D_ϕ, and the matching module M_γ work together to produce a dialogue response. The source-encoder first receives the input x and encodes it to a semantic representation h of the source utterance. Then, the mapping module transforms h to t, a target response representation. Finally, t is sent to the target-decoder for response generation.

In the training stage, besides the auto-encoder loss and the mapping loss, we also use an end-to-end loss $J_4(\theta, \phi, \gamma)$:

$$J_4(\theta, \phi, \gamma) = -\log P(y|x; \theta, \phi, \gamma) \tag{4}$$

$$= -\sum_{t=1}^{T} \log P(y_t|x, y_{1..t-1}; \theta, \phi, \gamma) \tag{5}$$

where x is the source input, y is the target response, and T is the length of response sequence. The model learns to generate $\tilde{y}$ to approximate y by minimizing the reconstruction losses $J_1(\theta)$ and $J_2(\phi)$, the mapping loss $J_3(\gamma)$, and the end-to-end loss $J_4(\theta, \phi, \gamma)$. The details are illustrated below:

$$\begin{aligned} J = \lambda_1 \left[J_1(\theta) + J_2(\phi)\right] + \lambda_2 J_3(\gamma) \\ + \lambda_3 J_4(\theta, \phi, \gamma) \end{aligned} \tag{6}$$

where J refers to the total loss, and λ_1, λ_2, and λ_3 are hyperparameters.

3 Experiment

We conduct experiments on a high-quality dialogue dataset called **DailyDialog** built by Li et al. (2017b). The dialogues in the dataset reflect our daily communication and cover various topics about our daily life. We split the dataset into three parts with 36.3K pairs for training, 11.1K pairs for validation, and 11.1K pairs for testing.

3.1 Experimental Details

For dialogue generation, we set the maximum length to 15 words for each generated sentence. Based on the performance on the validation set, we set the hidden size to 512, embedding size to 64 and vocabulary size to 40K for baseline models and the proposed model. The parameters are updated by the Adam algorithm (Kingma and Ba, 2014) and initialized by sampling from the uniform distribution ($[-0.1, 0.1]$). The initial learning rate is 0.002 and the model is trained in minibatches with a batch size of 256. λ_1 and λ_3 are set to 1 and λ_2 is set to 0.01 in Equation (6). It is important to note that for a fair comparison, we re-implement the baseline models with the best settings on the validation set. After fixing the hyperparameters, we combine the training and validation sets together as a larger training set to produce the final model.

3.2 Results

We use BLEU (Papineni et al., 2002), to compare the performance of different models, and use the widely-used BLEU-4 as our main BLEU score. The results are shown in Table 1. The proposed AEM model significantly outperforms the Seq2Seq model. It demonstrates the effectiveness of utterance-level dependency on improving the quality of generated text. Furthermore, we find that the utterance-level dependency also benefits the learning of word-level dependency. The improvement from the AEM model to the AEM+Attention model[2] is 0.68 BLEU-4 point. It is much more obvious than the improvement from the Seq2Seq model to the Seq2Seq+Attention, which is 0.29 BLEU-4 point.

We also report the diversity of the generated responses by calculating the number of distinct unigrams, bigrams, and trigrams. The results are shown in Table 2. We find that the AEM model achieves significant improvement on the diversity of generated text. The number of unique trigram of the AEM model is almost six times more

---

[2] With the additional attention mechanism, the outputs of attention-based decoder and our decoder are combined together to predict the probability of response words.

Models	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Seq2Seq	12.43	4.57	2.69	1.84
AEM	13.55	4.89	3.04	2.16
Seq2Seq+Attention	13.63	4.99	3.05	2.13
AEM+Attention	**14.17**	**5.69**	**3.78**	**2.84**

Table 1: BLEU scores for the AEM model and the Seq2Seq model.

Models	Dist-1	Dist-2	Dist-3
Seq2Seq	0.8K	2.7K	5.5K
AEM	3.1K	14.8K	31.2K
Seq2Seq+Attention	2.5K	13.6K	34.6K
AEM+Attention	**3.3K**	**23.2K**	**53.9K**

Table 2: Diversity scores for the AEM model and the Seq2Seq model. Dist-1, Dist-2, and Dist-3 are respectively the number of distinct unigrams, bigrams, and trigrams in the generated text.

Models	Fluency	Coherence	G-Score
Seq2Seq	6.97	3.51	4.95
AEM	**8.11**	4.18	5.82
Seq2Seq+Attention	5.11	3.30	4.10
AEM+Attention	7.92	**4.97**	**6.27**

Table 3: Human evaluation results of the AEM model and the Seq2Seq model.

Input 1: *My name is David. What is my name?*
Seq2Seq: *Ours, maria.*
AEM: *Pardon me? It is unreasonable!*

Input 2: *I'm going to kill you!*
Seq2Seq: *I know. I'm a good man.*
AEM: *What?!?!!!! No!!*

Input 3: *I hate you.*
Seq2Seq: *I'm not.*
AEM: *You do? you're a sweet girl.*

Input 4: *How are you?*
Seq2Seq: *Fine.*
AEM: *Fine. I'm good.*

Input 5: *Can you say 'Hello'?*
Seq2Seq: *Yeah, I guess.*
AEM: *Sure, I guess.*

Table 4: Examples generated by the AEM model and the Seq2Seq model.

than that of the Seq2Seq model. Also, it should be noticed that the attention mechanism performs almost the same compared to the AEM model (31.2K vs. 34.6K in terms of Dist-3), which indicates that the utterance-level dependency and the word-level dependency are both indispensable for dialogue generation. Therefore, by combining the two dependencies together, the AEM+Attention model achieves the best results. Such improvements are expected. With the increase of the relevance of the generated text, it gets harder for the model to generate repeated responses. In our experimental results, the number of repetitive "I don't know" in the AEM+Attention model is reduced by 50% compared to the Seq2Seq model.

For dialogue generation, human evaluation is more convincing, so we also report human evaluation results on the test set. We randomly choose 100 utterances in daily communication style for the human evaluation, each of which is sent to different models to generate responses. The results are distributed to the annotators who have no knowledge about which model the sentence is from. All annotators have linguistic background. They are asked to score the generated responses in terms of fluency and coherence. Fluency represents whether each sentence is in correct grammar. Coherence evaluates whether the generated response is relevant to the input. The score ranges from 1 to 10 (1 is very bad and 10 is very good). To evaluate the overall performance, we use the geometric mean of fluency and coherence as the final evaluation metric.

Table 3 shows the results of human evaluation. The inter-annotator agreement is satisfactory considering the difficulty of human evaluation. The Pearson's correlation coefficient is 0.69 on coherence and 0.57 on fluency, with $p < 0.0001$. First, it is clear that the AEM model outperforms the Seq2Seq model with a large margin, which proves the effectiveness of the AEM model on generating high quality responses. Second, it is interesting to note that with the attention mechanism, the coherence is decreased slightly in the Seq2Seq model but increased significantly in the AEM model. It suggests that the utterance-level dependency greatly benefits the learning of word-level dependency. Therefore, it is expected that the AEM+Attention model achieves the best G-score.

Table 4 shows the examples generated by the AEM model and the Seq2Seq model. For easy questions (ex. 4 and ex. 5), they both perform well. For hard questions (ex. 1 and ex. 2), the proposed model obviously outperforms the Seq2Seq model. It shows that the utterance-level dependency learned by the proposed model is useful for handling complex inputs.

3.3 Error Analysis

Although our model achieves the best performance, there are still several failure cases. We find that the model performs badly for the inputs with unseen words. For instance, given "Bonjour" as the input, it generates "Stay out of here" as the output. It shows that the proposed model is sensitive to the unseen utterance representations. Therefore, we would like to explore more approaches to address this problem in the future work. For example, the auto-encoders can be replaced by variational auto-encoders to ensure that the distribution of utterance representations is normal, which has a better generalization ability.

4 Conclusion

In this work, we propose an Auto-Encoder Matching model to learn the utterance-level semantic dependency, a critical dependency relation for generating coherent and fluent responses. The model contains two auto-encoders that learn the utterance representations in an unsupervised way, and a mapping module that builds the mapping between the input representation and response representation. Experimental results show that the proposed model significantly improves the quality of generated responses according to automatic evaluation and human evaluation, especially in coherence.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028). We thank all reviewers for providing the constructive suggestions. Xu Sun is the corresponding author of this paper.

References

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pages 207–219.

Pierre Baldi. 2012. Autoencoders, unsupervised learning, and deep architectures. In *Unsupervised and Transfer Learning - Workshop held at ICML 2011, Bellevue, Washington, USA, July 2, 2011*, pages 37–50.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683.

Deli Chen, Shuming Ma, Pengcheng Yang, and Xu Sun. 2018. Identifying high-quality chinese news comments based on multi-target text matching model. *CoRR*, abs/1808.07191.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 55–64.

Ryuichiro Higashinaka, Kenji Imamura, Toyomi Meguro, Chiaki Miyazaki, Nozomi Kobayashi, Hiroaki Sugiyama, Toru Hirano, Toshiro Makino, and Yoshihiro Matsuo. 2014. Towards an open-domain conversational system fully based on natural language processing. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 928–939.

Sepp Hochreiter and Jürgen Schmidhuber. 1996. LSTM can solve hard long time lag problems. In *NIPS, 1996*, pages 473–479.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2042–2050.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016,* *Austin, Texas, USA, November 1-4, 2016*, pages 1192–1202.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017a. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2157–2169.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017b. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 986–995.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. *CoRR*, abs/1805.03989.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Shuming Ma, Xu Sun, Junyang Lin, and Houfeng Wang. 2018a. Autoencoder as assistant supervisor: Improving text representation for chinese social media text summarization. *CoRR*, abs/1805.04869.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018b. Bag-of-words as target for neural machine translation. *CoRR*, abs/1805.04871.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2017. Coherent dialogue with attention-based language models. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3252–3258.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2793–2799.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C. Courville. 2017. Multiresolution recurrent neural networks: An application to dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3288–3294.

Hui Su, Xiaoyu Shen, Pengwei Hu, Wenjie Li, and Yun Chen. 2018. Dialogue generation with GAN. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *EMNLP, 2018*.

Li Zhou, Kevin Small, Oleg Rokhlenko, and Charles Elkan. 2017. End-to-end offline goal-oriented dialog policy learning via policy gradient. *CoRR*, abs/1712.02838.

A Dataset for Document Grounded Conversations

Kangyan Zhou, Shrimai Prabhumoye, Alan W Black
Carnegie Mellon University, Pittsburgh, PA, USA
`{kangyanz, sprabhum, awb}@cs.cmu.edu`

Abstract

This paper introduces a document grounded dataset for conversations. We define "Document Grounded Conversations" as conversations that are about the contents of a specified document. In this dataset the specified documents were Wikipedia articles about popular movies. The dataset contains 4112 conversations with an average of 21.43 turns per conversation. This positions this dataset to not only provide a relevant chat history while generating responses but also provide a source of information that the models could use. We describe two neural architectures that provide benchmark performance on the task of generating the next response. We also evaluate our models for engagement and fluency, and find that the information from the document helps in generating more engaging and fluent responses.

1 Introduction

At present, dialog systems are considered to be either task-oriented, where a specific task is the goal of the conversation (e.g. getting bus information or weather for a particular location); or non-task oriented where conversations are more for the sake of themselves, be it entertainment or passing the time. Ultimately, we want our agents to smoothly interleave between task-related information flow and casual chat for the given situation. There is a dire need of a dataset which caters to both these objectives.

Serban et al. (2015) provide a comprehensive list of available datasets for building end-to-end conversational agents. Datasets based on movie scripts (Lison and Tiedemann, 2016; Danescu-Niculescu-Mizil and Lee, 2011a) contain artificial conversations. The Ubuntu Dialogue Corpus (Lowe et al., 2015) is based on technical support logs from the Ubuntu forum. The Frames dataset (Asri et al., 2017) was collected to solve the problem of frame tracking. These datasets do not provide grounding of the information presented in the conversations. Zhang et al. (2018) focuses on personas in dialogues: each worker has a set of predefined facts about the persona that they can talk about. Most of these datasets lack conversations with large number of on-topic turns.

We introduce a new dataset which addresses the concerns of grounding in conversation responses, context and coherence in responses. We present a dataset which has real human conversations with grounding in a document. Although our examples use Wikipedia articles about movies, we see the same techniques being valid for other external documents such as manuals, instruction booklets, and other informational documents. We build a generative model with and without the document information and find that the responses generated by the model with the document information is more engaging (+7.5% preference) and more fluent (+0.96 MOS). The perplexity also shows a 11.69 point improvement.

2 The Document Grounded Dataset

To create a dataset for document grounded conversations, we seek the following things: (1) A set of documents (2) Two humans chatting about the content of the document for more than 12 turns. We collected conversations about the documents through Amazon Mechanical Turk (AMT). We restrict the topic of the documents to be movie-related articles to facilitate the conversations. We initially experimented with different potential domains. Since movies are engaging and widely known, people actually stay on task when discussing them. In fact in order to make the task interesting, we offered a choice of movies to the participants so that they are invested in the task.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 708–713
Brussels, Belgium, October 31 - November 4, 2018.

2.1 Document Set Creation

We choose Wikipedia (Wiki) [1] articles to create a set of documents $\boldsymbol{D} = \{d_1, \ldots, d_{30}\}$ for grounding of conversations. We randomly select 30 movies, covering various genres like thriller, super-hero, animation, romantic, biopic etc. We extract the key information provided in the Wiki article and divide it into four separate sections. This was done to reduce the load of the users to read, absorb and discuss the information in the document. Hence, each movie document d_i consists of four sections $\{s_1, s_2, s_3, s_4\}$ corresponding to basic information and three key scenes of the movie. The basic information section s_1 contains data from the Wikipedia article in a standard form such as year, genre, director. It also includes a short introduction about the movie, ratings from major review websites, and some critical responses. Each of the key scene sections $\{s_2, s_3, s_4\}$ contains one short paragraph from the plot of the movie. Each paragraph contains on an average 7 sentences and 143 words. These paragraphs were extracted automatically from the original articles, and were then lightly edited by hand to make them of consistent size and detail. An example of the document is attached in Appendix.

2.2 Dataset Creation

To create a dataset of conversations which uses the information from the document, involves the participation of two workers. Hence, we explore two scenarios: (1) Only one worker has access to the document and the other worker does not and (2) Both the workers have access to the document. In both settings, they are given the common instructions of chatting for at least 12 turns.

Scenario 1: One worker has document. In this scenario, only one worker has access to the document. The other worker cannot see the document. The instruction to the worker with the document is: *Tell the other user what the movie is, and try to persuade the other user to watch/not to watch the movie using the information in the document*; and the instruction to the worker without the document is: *After you are told the name of the movie, pretend you are interested in watching the movie, and try to gather all the information you need to make a decision whether to watch the movie in the end*. An example of part of the dialogue for this scenario is shown in Table 1.

user2:	Hey have you seen the inception?
user1:	No, I have not but have heard of it. What is it about
user2:	It's about extractors that perform experiments using military technology on people to retrieve info about their targets.

Table 1: An example conversation for scenario 1. User 1 does not have access to the document, while User 2 does. The full dialogue is attached in the Appendix.

User 2:	I thought The Shape of Water was one of Del Toro's best works. What about you?
User 1:	Did you like the movie?
User 1:	Yes, his style really extended the story.
User 2:	I agree. He has a way with fantasy elements that really helped this story be truly beautiful.

Table 2: An example conversation for scenario 2. Both User 1 and User 2 have access to the Wiki document. The full dialogue is attached in the Appendix.

Scenario 2: Both workers have document. In this scenario, both the workers have access to the same Wiki document. The instruction given to the workers are: *Discuss the content in the document with the other user, and show whether you like/dislike the movie*. An example of the dialogue for this scenario is shown in Table 2.

Workflow: When the two workers enter the chat-room, they are initially given only the first section on basic information s_1 of the document d_i. After the two workers complete 3 turns (for the first section 6 turns is needed due to initial greetings), the users will be shown the next section. The users are encouraged to discuss information in the new section, but are not constrained to do so.

2.3 Dataset Statistics

The dataset consists of total 4112 conversations with an average of 21.43 turns. The number of conversations for scenario 1 is 2128 and for scenario 2 it is 1984. We consider a turn to be an exchange between two workers (say $w1$ and w_2). Hence an exchange of w_1, w_2, w_1 has 2 turns (w_1, w_2) and (w_2, w_1). We show the comparison of our dataset as **CMU Do**cument **G**rounded Conversations (CMU_DoG) with other datasets in Table 3.

---

[1] https://en.wikipedia.org

Dataset	# Utterances	Avg. # of Turns
CMU_DoG	130000	31
Persona-chat (Zhang et al., 2018)	164,356	14
Cornell Movie (Danescu-Niculescu-Mizil and Lee, 2011b)	304,713	1.38
Frames dataset (Asri et al., 2017)	19,986	15

Table 3: Comparison with other datasets. The average number of turns are calculated as the number of utterances divided by the number of conversations for each of the datasets.

	Rating 1	Rating 2	Rating 3	Rating 2& 3
Total # of conversations	1443	2142	527	2669
Total # of utterances	28536	80104	21360	101464
Average # utterances/conversation	19.77(13.68)	35.39(8.48)	40.53(12.92)	38.01(9.607)
Average length of utterance	7.51(50.19)	10.56(8.51)	16.57(15.23)	11.83(10.58)

Table 4: The statistics of the dataset. Standard deviation in parenthesis.

One of the salient features of CMUDoG dataset is that it has mapping of the conversation turns to each section of the document, which can then be used to model conversation responses. Another useful aspect is that we report the quality of the conversations in terms of how much the conversation adheres to the information in the document.

Split Criteria: We automatically measure the quality of the conversations using BLEU (Papineni et al., 2002) score. We use BLEU because we want to measure the overlap of the turns of the conversation with the sections of the document. Hence, a good quality conversation should use more information from the document than a low quality conversation. We divide our dataset into three ratings based on this measure. The BLEU score is calculated between all the utterances $\{x_1, \ldots, x_n\}$ of a conversation C_i and the document d_i corresponding to C_i. We eliminate incomplete conversations that have less than 10 turns. The percentiles for the remaining conversations are shown in Table 5. We split the dataset into three ratings based on BLEU score.

Percentile	20	40	60	80	99
BLEU	0.09	0.20	0.34	0.53	0.82

Table 5: The distribution of BLEU score for conversations with more than 10 turns.

Rating 1: Conversations are given a rating of 1 if their BLEU score is less than or equal to 0.1. We consider these conversations to be of low-quality.

Rating 2: All the conversations that do not fit in rating 1 and 3 are marked with a rating of 2.

Rating 3: Conversations are labeled with a rating of 3, only if the conversation has more than 12 turns and has a BLEU score larger than 0.587. This threshold was calculated by summing the mean (0.385) and the standard deviation (0.202) of BLEU scores of the conversations that do not belong rating 1.

The average BLEU score for workers who have access to the document is 0.22 whereas the average BLEU score for the workers without access to the document is 0.03. This suggests that even if the workers had external knowledge about the movie, they have not extensively used it in the conversation. It also suggests that the workers with the document have not used the information from the document verbatim in the conversation. Table 4 shows the statistics on the total number of conversations, utterances, and average number of utterances per conversation and average length of utterances for all the three ratings.

3 Models

In this section we discuss models which can leverage the information from the document for generating responses. We explore generative models for this purpose. Given a dataset $\boldsymbol{X} = \{x_0, \ldots, x_n\}$ of utterances in a conversation C_i, we consider two settings: (1) to generate a response x_{i+1} when given only the current utterance x_i and (2) to generate a response x_{i+1} when given the corresponding section s_i and the previous utterance x_i.

Without section: We use the sequence-to-sequence model (Sutskever et al., 2014) to build our baseline model. Formally, let $\boldsymbol{\theta}_E$ represent the parameters of the encoder. Then the representation $\boldsymbol{h}_{x_i}$ of the current utterance x_i is given by:

$$\boldsymbol{h}_{x_i} = \text{Encoder}(\boldsymbol{x}_i; \boldsymbol{\theta}_E) \tag{1}$$

Samples of x_{i+1} are generated as follows:

$$p(\hat{\boldsymbol{x}}|\boldsymbol{h}_{x_i}) = \prod_t p(\hat{x}_t|\hat{\boldsymbol{x}}^{<t}, \boldsymbol{h}_{x_i}) \tag{2}$$

where, $\hat{\boldsymbol{x}}^{<t}$ are the tokens generated before $\hat{x}_t$. We also use global attention (Luong et al., 2015) with copy mechanism (See et al., 2017) to guide our generators to replace the unknown (UNK) tokens. We call this model **SEQ**.

With section: We extend the sequence-to-sequence framework to include the section s_i corresponding the current turn. We use the same encoder to encode both the utterance and the section. We get the representation $\boldsymbol{h}_{x_i}$ of the current utterance x_i using Eq. 1. The representation of the section is given by:

$$\boldsymbol{h}_{s_i} = \text{Encoder}(\boldsymbol{s}_i; \boldsymbol{\theta}_E) \tag{3}$$

The input at each time step t to the generative model is given by $h_t = [x_{t-1}; h_s]$, where x_{t-1} is the embedding of the word at the previous time step. We call this model **SEQS**.

Experimental Setup: For both SEQ and SEQS model, we use a two-layer bidirectional LSTM as the encoder and a LSTM as the decoder. The dropout rate of the LSTM output is set to be 0.3. The size of hidden units for both LSTMs is 300. We set the word embedding size to be 100, since the size of vocabulary is relatively small[2]. The models are trained with adam (Kingma and Ba, 2014) optimizer with learning rate 0.001 until they converge on the validation set for the perplexity criteria. We use beam search with size 5 for response generation. We use all the data (i.e all the conversations regardless of the rating and scenario) for training and testing. The proportion of train/validation/test split is 0.8/0.05/0.15.

4 Evaluation

In what follows, we first present an analysis of the dataset, then provide an automatic metric for evaluation of our models–perplexity and finally present the results of human evaluation of the generated responses for engagement and fluency.

scenario	NW	LT
1	0.78	12.85
2	5.84	117.12

Table 6: The results of data analysis. LT refers to the average length of x_i in scenario 1 and $x_i, \ldots, x_{i+k}$ in scenario 2.

Dataset analysis: We perform two kinds of automated evaluation to investigate the usefulness of the document in the conversation. The first one is to investigate if the workers use the information from the document in the conversation. The second analysis is to show that the document adds value to the conversation. Let the set of tokens in the current utterance x_i be N, the set of tokens in the current section s_i be M, the set of tokens in the previous three utterances be H, and the set of stop words be S. In scenario 1, we calculate the set operation (NW) as $|((N \cap M) \setminus H) \setminus S|$. Let the tokens that appear in all the utterances $(x_i, \ldots, x_{i+k})$ corresponding to the current section s_i be K and the tokens that appear in all the utterances $(x_i, \ldots, x_{i+p})$ corresponding to the previous section s_{i-1} be P. In scenario 2, we calculate the set operation (NW) as $|((K \cap M) \setminus P) \setminus S|$. The results in Table 6 show that people use the information in the new sections and are not fixated on old sections. It also shows that they use the information to construct the responses.

Perplexity: To automatically evaluate the fluency of the models, we use perplexity measure. We build a language model on the train set of responses using ngrams up to an order of 3[3]. The generated test responses achieve a perplexity of 21.8 for the SEQ model and **10.11** for the SEQS model. This indicates that including the sections of document helps in the generation process.

4.1 Human Evaluation

We also perform two kinds of human evaluations to evaluate the quality of predicted utterances – engagement and fluency. These experiments are performed on Amazon Mechanical Turk.

Engagement: We set up a pairwise comparison following Bennett (2005) to evaluate the engagement of the generated responses. The test presents the chat history (1 utterance) and then, in random

---

[2] The total number of tokens is 46000, and we limit the vocabulary to be 10000 tokens.

[3] We use the SRILM toolkit (Stolcke, 2002)

order, its corresponding response produced by the SEQ and SEQS models. A third option "No Preference" was given to participants to mark no preference for either of the generated responses. The instruction given to the participants is "Given the above chat history as context, you have to pick the one which can be best used as the response based on the engagingness." We randomly sample 90 responses from each model. Each response was annotated by 3 unique workers and we take majority vote as the final label. The result of the test is that SEQ generated responses were chosen only 36.4% times as opposed to SEQS generated responses which were chosen **43.9%** and the "No Preference" option was chosen 19.6% of times. This result shows the information from the sections improves the engagement of the generated responses.

Fluency: The workers were asked to evaluate the fluency of the generated response on a scale of 1 to 4, where 1 is unreadable and 4 is perfectly readable. We randomly select 120 generated responses from each model and each response was annotated by 3 unique workers. The SEQ model got a low score of 2.88, contrast to the SEQS score of **3.84**. This outcome demonstrates that the information in the section also helps in guiding the generator to produce fluent responses.

5 Conclusion

In this paper we introduce a crowd-sourced conversations dataset that is grounded in a predefined set of documents which is available for download [4]. We perform multiple automatic and human judgment based analysis to understand the value the information from the document provides to the generation of responses. The SEQS model which uses the information from the section to generate responses outperforms the SEQ model in the evaluation tasks of engagement, fluency and perplexity.

Acknowledgments

This work was funded by a fellowship from Robert Bosch, and in part by Facebook Inc. and Microsoft Corporation. This work was performed as a part of The Conversational Intelligence Challenge (ConvAI, NIPS 2017) and we would like to thank the ConvAI team. We are also grateful to the anonymous reviewers for their constructive feedback and to Carolyn Penstein Rose, Shivani Poddar, Sreecharan Sankaranarayanan, Samridhi Shree Choudhary and Zhou Yu for valuable discussions at earlier stages of this work.

References

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv preprint arXiv:1704.00057*.

Christina L Bennett. 2005. Large scale evaluation of corpus-based synthesizers: Results and lessons from the blizzard challenge 2005. In *Ninth European Conference on Speech Communication and Technology*.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011a. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87. Association for Computational Linguistics.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011b. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. ACL*.

---

[4] https://www.github.com/festvox/datasets/CMU_DoG

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2015. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

Out-of-domain Detection based on Generative Adversarial Network

Seonghan Ryu[1,2], **Sangjun Koo**[2], **Hwanjo Yu**[2], **and Gary Geunbae Lee**[2]
[1]Language Understanding Lab, Artificial Intelligence Center, Samsung Research
[2]Computer Science and Engineering Department, Pohang University of Science and Technology
`seonghan.ryu@samsung.com`

Abstract

The main goal of this paper is to develop *out–of–domain* (OOD) detection for dialog systems. We propose to use only in–domain (IND) sentences to build a generative adversarial network (GAN) of which the discriminator generates low scores for OOD sentences. To improve basic GANs, we apply feature matching loss in the discriminator, use domain–category analysis as an additional task in the discriminator, and remove the biases in the generator. Thereby, we reduce the huge effort of collecting OOD sentences for training OOD detection. For evaluation, we experimented OOD detection on a multi–domain dialog system. The experimental results showed the proposed method was most accurate compared to the existing methods.

1 Introduction

Multi–domain dialog systems (Hakkani-Tur et al., 2016; Jiang et al., 2014; Lee et al., 2013; Ryu et al., 2015; Seon et al., 2014) should detect whether an input request is *out–of–domain* (OOD) because users do not know the exact coverages of those systems. One important problem of building OOD detection is the huge effort required to collect OOD sentences. This paper focuses on developing an accurate OOD detection method that requires only in–domain (IND) sentences for training, so this paper can reduce the effort of collecting OOD sentences.

For OOD detection, sentences would be represented in a continuous vector space in which IND cases are distinguished from OOD cases. Therefore, we use the existing *sentence embedding* method for OOD detection (Ryu et al., 2017). The authors train a recurrent neural network (RNN) for *domain–category analysis* task, in which one domain–category is assigned to an input sentence. Due to the similarity between OOD detection and domain–category analysis, the extracted features (i.e., representation) of the RNN contain information about domain–category. In addition, the word representations are pre–trained from a large unlabelled corpus, so the sentence embedding method has the advantage in understanding rare or unknown words that are likely to appear in OOD sentences.

Afterwards, we use the learned representations of IND sentences to train *one–class classifiers* that distinguish IND sentences from OOD sentences. We propose to use a *generative adversarial network* (GAN) (Goodfellow et al., 2014) that consists of a generator G and a discriminator D. We train D that distinguishes the IND sentences from the fake sentences generated by G, so we expect D to reject OOD sentences. We apply three modifications to improve basic GANs. To the best of our knowledge, this is the first study that uses GANs to solve OOD detection.

2 Related Work

Lane et al. (2007) proposed an *in–domain verification* method. The authors first build a basic binary classifier for each domain, and then build a meta classifier that takes the scores by the basic binary classifiers as input. However, in our experiment, many OOD sentences were misclassified into IND because OOD sentences were not in the negative examples of the classifiers. Therefore, the confidence scores of the basic binary classifiers are not sufficiently reliable evidences of OOD. Also, understanding rare or unknown words remains a problem because bag–of–words model is used.

Ryu et al. (2017) proposed an *autoencoder*–based method. The authors use neural sentence embedding that has the advantage in representing rare or unknown words. Based on those distributed sentence representations, an autoencoder

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 714–718
Brussels, Belgium, October 31 - November 4, 2018.

① “*When was Good Day released?*” (Songfinder)

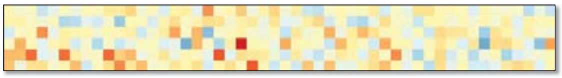

② “*I have an appointment November 20 at 4 pm.*” (Schedule)

③ “*What exercise should I do in the evening?*” (Diet Talk)

④ “*Please delete the recorded Infinite Challenge.*” (TV)

Figure 1: Distributed representations of IND sentences.

① “*Please send this message to my dad.*” (Message)

② “*Show me my hotel reservations.*” (Hotel)

③ “*I don't play with you any more.*” (Smalltalk)

④ “*I heard my friend has divorced.*” (Smalltalk)

Figure 2: Distributed representations of OOD sentences.

is trained on IND sentences. The autencoder will have low reconstruction errors for IND sentences, so an input sentence can be classified into either IND or OOD. However, the autoencoder–based method has a limitation in expandability. When the weights are initialized carefully and regularization techniques are applied, the trained autoencoder reconstructs any input accurately. This result means that the reconstruction errors by the *ideal* autoencoder are not reliable evidence of OOD, so in OOD detection, autoencoders have little room for improvement.

3 Methods

As we discussed in Section 1, we use the sentence embedding to represent sentences in an 300–dimensional continuous vector space. We propose to use a GAN to OOD detection; a GAN consists of two adversarial components: generator G and discriminator D. G generates artificial data to deceive D. D distinguishes real data from the artificial data generated by G. GAN is an unsupervised algorithm because learning G and D does not require labels. Standard GANs are trained based on the objective function $V(D,G)$ as

$$\begin{aligned}\min_G \max_D V(D,G) = {} & \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] \\ & + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log 1 - D(G(\boldsymbol{z}))].\end{aligned} \quad (1)$$

So GAN is a minimax two–player game because G minimizes $V(D,G)$, and D maximizes $V(D,G)$.

We propose to use GANs to obtain a one–class classifier for OOD detection. When we train G to generate sentences similar to IND sentences and D to classify real IND sentences and fake sentences generated by G, we expect D to reject OOD sentences. Therefore, we use the low confidence score by D about an input sentence as the evidence that the sentence is OOD.

Let $p_{\mathbf{z}}(z)$ be a continuous uniform distribution $(-1,1)$. We define G that generates fake data $G(\mathbf{z}) \in \mathrm{R}^m$ from input noise $\mathbf{z} \sim p_{\mathbf{z}}(z)$, $\mathbf{f}$ that extracts features from either real data $\mathbf{x}$ of $(\mathbf{x},\mathbf{y}) \sim p_{\text{data}}(\mathbf{x},\mathbf{y})$ or $G(\mathbf{z})$, and D that measures the probability of either $\mathbf{f}(\mathbf{x})$ or $\mathbf{f}(G(\mathbf{z}))$ from the real data as

$$\begin{aligned} G(\mathbf{z}) &= \sigma_h(\mathbf{W}_g\mathbf{z}), && (2)\\ \mathbf{f}(\mathbf{x}) &= \sigma_h(\mathbf{W}_h\mathbf{x} + \mathbf{b}_f), && (3)\\ \mathbf{f}(G(\mathbf{z})) &= \sigma_h(\mathbf{W}_h G(\mathbf{z}) + \mathbf{b}_f), && (4)\\ D(\mathbf{f}(\mathbf{x})) &= \sigma_g(\mathbf{W}_d\mathbf{f}(\mathbf{x}) + \mathbf{b}_d), && (5)\\ D(\mathbf{f}(G(\mathbf{z}))) &= \sigma_g(\mathbf{W}_d\mathbf{f}(G(\mathbf{z})) + \mathbf{b}_d), && (6)\end{aligned}$$

where $\mathbf{W}_g$, $\mathbf{W}_f$, and $\mathbf{W}_d$ are weight matrices, $\mathbf{b}_f$ and $\mathbf{b}_d$ are bias vectors. So we define two objective functions

$$\begin{aligned} L_{\text{D}} = {} & -\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}[\log D(\mathbf{f}(\mathbf{x}))] && (7)\\ & - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(\mathbf{f}(G(\mathbf{z}))))], \\ L_{\text{G}} = {} & -\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log D(\mathbf{f}(G(\mathbf{z})))]. && (8)\end{aligned}$$

Because $G(\mathbf{z})$ continuously changes during the training, f of traditional GAN also changes. Thus we define $C \in \mathbb{R}^{|D|}$ that computes domain–category of $\mathbf{f}(\mathbf{x})$ as

$$C(\mathbf{f}(\mathbf{x})) = \text{softmax}(\mathbf{W}_c\mathbf{f}(\mathbf{x}) + \mathbf{b}_c), \quad (9)$$

where $\mathbf{W}_c$ is a weight matrix and $\mathbf{b}_c$ is a bias vector. We expect f trained by the losses of both D

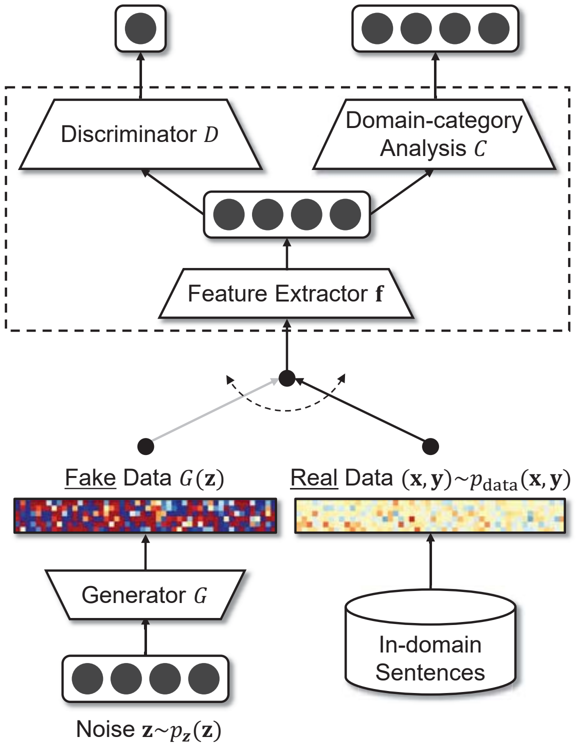

Figure 3: Generative adversarial network for out–of–domain sentence detection.

and C to be more stable than f trained by the loss of only D, so we define an objective function

$$L_{\text{C}} = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}[H(C(\mathbf{f}(\mathbf{x})),\mathbf{y})], \qquad (10)$$

where $H(p,q)$ is the categorical cross entropy and $\mathbf{y}$ is the true domain–category.

In addition, GAN suffers from a *mode collapse* problem in which G generates samples with a low variance. To solve the problem, we remove the biases in the generator because the G was trained to use the biases mainly instead of the weights to generate data.

Second, we apply feature matching (Salimans et al., 2016). The authors say "Instead of directly maximizing the output of the discriminator, the new objective requires the generator to generate data that matches [sic] the statistics of the real data, where we use the discriminator only to specify the statistics that we think are worth matching". So G is trained to generate high variance sentence $G(\mathbf{z})$ by additional objective function

$$L_{\text{f}} = ||\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}\mathbf{f}(\mathbf{x}) - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\mathbf{f}(G(\mathbf{z}))||_2^2. \qquad (11)$$

Based on our design of GAN, we train D, C, $\mathbf{f}$, and G as Algorithm 1. To implement our GAN for one–class classification, we use the Tensorflow library (Abadi et al., 2015). We train our models by using Adam (Kingma and Ba, 2015) optimizer with a mini–batch size of 256 and an initial learning rate of 0.01 that is decreased linearly during 500 epochs. All weights are initialized from a zero–centered Normal distribution with standard deviation 1.0.

Algorithm 1 Training process of GAN for OOD detection.

```
for number of training iterations do
    Sample real data $(\mathbf{x},\mathbf{y}) \sim p_{\text{data}}(\mathbf{x},\mathbf{y})$.
    Sample noise $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$.
    Update $D$, $C$, and $\mathbf{f}$ based on $L_{\text{D}} + L_{\text{C}}$.
    Sample noise $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$.
    Update $G$ based on $L_{\text{G}} + L_{\text{f}}$.
end for
```

4 Experiments

4.1 Data Set

We experimented on a data set of 6,268 Korean sentences. We collected 706 OOD sentences about three domains: hotel, message, and smalltalk; and 5,562 IND sentences about fourteen domains: airplane, alarm, bus, call, car navigation, diet talk, exchange, general, schedule, songfinder, time, train, and TV control. We used eighty percent of the IND sentences to train the models; we used the remaining IND sentences and all OOD sentences for testing.

4.2 Evaluation Metrics

We use *equal error rate* (EER) to represent the accuracy of OOD detection (Lane et al., 2007). EER is the error rate at which false acceptance rate

$$\text{FAR} = \frac{\text{Number of } \textit{accepted} \text{ OOD sentences}}{\text{Number of OOD sentences}} \qquad (12)$$

and false rejection rate

$$\text{FRR} = \frac{\text{Number of } \textit{rejected} \text{ ID sentences}}{\text{Number of ID sentences}} \qquad (13)$$

are equal.

We performed each experiment 20 times, and recorded the average EER of OOD detection.

4.3 Compared Methods

We have three variations of vanilla GAN: to remove the biases, to add domain–category analy-

Table 1: EERs [%] ± s.d. (n = 20) of OOD detection.

Method	EER
Local outlier factor	13.33
One–class SVM	13.76
Autoencoder	9.24 ± 0.43
GAN with biases	15.93 ± 5.82
GAN	9.18 ± 0.30
GAN with DCA task	9.17 ± 0.40
GAN with FM loss	9.04 ± 0.30
GAN with DCA task and FM loss	8.96 ± 0.34

sis (DCA) task, and to add feature matching (FM) loss. So we assessed five settings about GAN.

We compare our method to three one–class classifiers. (1) *Local outlier factor* (Breunig et al., 2000) compares the local density of a point to the local densities of its neighbors, and considers the point that has lower density than their neighbors as an outlier. The local density of a point is defined by the distance to its nearest neighbors. (2) *One–class support vector machines* (One–class SVMs) (Schölkopf et al., 2001) that treats the origin as a negative example to learn a decision function. (3) *Autoencoder* is explained in Section 2.

4.4 Results

In the experiments (Table 1), the best EER (8.96%) was obtained by the GAN in which all three of our variations are applied ($p < 0.05$). This result means that (1) removing the biases and using the feature matching prevented the generator from mode collapse problem and (2) using domain–category analysis as an auxiliary task stabilized the training of feature extractor. Compared to the other one–class classification methods including the autoencoder, the proposed GAN was most accurate ($p < 0.05$), so we can say that the discriminator scores of the GAN are reliable evidence for OOD detection (Table 2).

5 Conclusion

In this paper, we aimed at building OOD detection without OOD sentences for training. We proposed to use the discriminator of a GAN, which is trained on only IND sentences. The proposed method outperformed the existing methods in our data set.

Table 2: Average score [%] by the discriminator of the GAN.

Data	Score
IND training sentences	98.69 (± 3.36)
Fake sentences $G(\mathbf{z})$	0.20 (± 1.70)
IND test sentences	88.59 (± 28.09)
OOD test sentences	8.04 (± 23.10)

To train the GAN, we used the distributed sentence representations computed from the pre–trained sentence embeddings instead of symbolic sentences. However, we think the limitation of the pre–trained sentence embeddings can be overcome by building a GAN that generates symbolic sentences and discriminates them.

Acknowledgements

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2018-2015-0-00742) supervised by the IITP(Institute for Information & communications Technology Promotion)

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, ..., Devin, Moore, Vanhoucke, Warden, 2015. TensorFlow: Large–scale machine learning on heterogeneous distributed systems. Software available from tensorflow.org.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying density–based local outliers. In *Proceedings of ACM SIGMOD*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*.

Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi–domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Proceedings of Interspeech*.

Ridong Jiang, Rafael E. Banchs, Seokhwan Kim, Kheng Hui Yeo, Arthur Niswar, and Haizhou Li. 2014. Web–based multimodal multi-domain spoken dialogue system. In *Proceedings of IWSDS*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Ian Lane, Tatsuya Kawahara, Tomoko Matsui, and Satoshi Nakamura. 2007. Out–of–domain utterance detection using classification confidences of multiple topics. *IEEE/ACM Trans. Audio, Speech, Language Process.*, 15:150–161.

Donghyeon Lee, Minwoo Jeong, Kyungduk Kim, Seonghan Ryu, and Gary Geunbae Lee. 2013. Unsupervised spoken language understanding for a multi–domain dialog system. *IEEE/ACM Trans. Audio, Speech, Language Process.*, 21:2451–2464.

Seonghan Ryu, Seokhwan Kim, Junhwi Choi, Hwanjo Yu, and Gary Geunbae Lee. 2017. Neural sentence embedding using only in-domain sentences for out-of-domain sentence detection in dialog systems. *Pattern Recogn. Lett.*, 88:26–32.

Seonghan Ryu, Jaiyoun Song, Sangjun Koo, Soonchoul Kwon, and Gary Geunbae Lee. 2015. Detecting multiple domains from users utterance in spoken dialog system. In *Proceedings of IWSDS*.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training GANs. In *Proceedings of NIPS*.

Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. 2001. Estimating the support of a high–dimensional distribution. *Neural Comput.*, 13:1443–1471.

Choong-Nyoung Seon, Hyunjung Lee, Harksoo Kim, and Jungyun Seo. 2014. Improving domain action classification in goal–oriented dialogues using a mutual retraining method. *Pattern Recogn. Lett.*, 45.

Listening Comprehension over Argumentative Content

Shachar Mirkin[*1], **Guy Moshkowich**[*2], **Matan Orbach**[*2], **Lili Kotlerman**[2], **Yoav Kantor**[2],
Tamar Lavee[3], **Michal Jacovi**[2], **Yonatan Bilu**[2], **Ranit Aharonov**[2] **and Noam Slonim**[2]
[1]Digimind, [2]IBM Research, [3]IBM Watson

Abstract

This paper presents a task for machine listening comprehension in the argumentation domain and a corresponding dataset in English. We recorded 200 spontaneous speeches arguing for or against 50 controversial topics. For each speech, we formulated a question, aimed at confirming or rejecting the occurrence of potential arguments in the speech. Labels were collected by listening to the speech and marking which arguments were mentioned by the speaker. We applied baseline methods addressing the task, to be used as a benchmark for future work over this dataset. All data used in this work is freely available for research.

1 Introduction

Machine reading comprehension (MRC) is the NLP task equivalent to reading comprehension tests that assess the understanding of written texts by humans. MRC is usually realized as a question answering (QA) task through multiple-choice questions or as a cloze test (Richardson et al., 2013; Hermann et al., 2015; Hill et al., 2015). With the abundance of multimedia content nowadays, this line of work has been extended to speech, by applying QA methods to speech transcripts, i.e. the output of automatic speech recognition (ASR). In such works, the task is consequently referred to as 'spoken question answering' (Li et al., 2018), 'question answering over speech transcripts' (Turmo et al., 2007; Lamel et al., 2008) or *machine listening comprehension* (MLC) (Chung and Glass, 2018).

We continue this line of work, and present a listening comprehension task and associated benchmark data over argumentative content. In the argumentation domain, such as political debates, people are often exposed directly to the audio (or the video), without access to a written version. Human comprehension is then done in real-time through listening. We simulate this scenario in our dataset. Namely, annotation is carried out by *listening* to debate speeches rather than by reading transcripts as done in prior work. The auditory modality is richer than written text in terms of the signal available to listeners, e.g., prosody. Similarly, machine comprehension can make use of the extra-lexical signal. The dataset we construct and release enables utilizing such signals, as done for instance in (Lippi and Torroni, 2016) for detecting claims in debates.

Most often, in both reading and listening comprehension tasks, the answer is explicitly mentioned in the text; frequently, the answer is even an actual segment of the text, as in SQuAD (Rajpurkar et al., 2016), one of the most popular MRC datasets. Conversely, in argumentation, presuppositions are fundamental (Habernal et al., 2018), inferences are more subtle and the answer may rely on common knowledge. Going beyond the factoid level, Tseng et al. (2016) presented a listening comprehension task over TOEFL listening tests.[1] In comparison, our data consists of spontaneous speech and is not adapted for non-native speakers.

We use data from iDebate[2], a high-quality, curated database of controversial topics – referred to as "motions", as in formal parliamentary proposals – with a list of arguments for and against each motion. We selected 50 motions, and recorded experienced debaters making four speeches for each of them (two for and two against the motion). We then asked annotators to listen to a speech and presented them with a list of arguments that were proposed independently in iDebate for the motion. The annotators had to mark which of these argu-

---

[*] This work was done at IBM within Project Debater; the first 3 authors equally contributed to this work.

[1] `https://www.ets.org/toefl`

[2] `https://idebate.org/debatabase`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 719–724
Brussels, Belgium, October 31 - November 4, 2018.

ments were mentioned in the speech (see Section 2 for further details). Example 1 shows one such argument alongside a speech snippet, demonstrating the unique nature of this domain and data. Specifically, the argument against the motion is implied from the speech, but is not explicitly mentioned in it.

Example 1 **(Positively labeled argument)**
Motion: *We should introduce goal line technology [in sports]*
Argument: *Controversy and debate are a part of the game*
Speech stance: *opposing ("con")*
... people also like it to some extent when officials make mistakes, because it adds to some of the like drama, the, oh, what if this happened? ... And I think that one of the biggest things that fans enjoy bonding over is when refs make mistakes that are blatantly wrong.

---

iDebate was chosen since its arguments are good candidates to construct comprehension questions: more than half of the assessed arguments are mentioned in our recorded speeches, and the large majority of the speeches contain at least one of the arguments suggested by iDebate. Furthermore, each argument in iDebate is coupled with a counter-argument. Those, in turn, may be used to rebut each argument that was detected through MLC, suggesting intriguing future directions to explore the released data. In a task related to ours, Boltužić and Šnajder (2014) have searched iDebate arguments in user argumentative comments. Their work, though, consisted of only two motions and included written, rather than audio data.

We release our annotated data and the results of baseline methods applied to it as a benchmark dataset for the MLC task. The dataset includes 200 speeches for 50 motions, in English. For each speech we include the following: (i) the audio version of the speech; (ii) manual and automatic transcripts; (iii) a labeled listening comprehension question, consisting of a set of arguments from iDebate that potentially appear in the speech.

The main contributions of this work are: (i) proposal of the new task of listening comprehension over argumentative content, a domain very different from those previously used for reading or listening comprehension tasks; (ii) a comprehensive and rich labeled dataset of 200 speeches covering 50 topics, transcribed both automatically and manually, and labeled for the listening comprehension task; (iii) establishment of baselines over the dataset.

All the recordings, their transcripts and labels are available for research at `https://www.research.ibm.com/haifa/dept/vst/debating_data.shtml`.

2 Data

iDebate iDebate contains a list of controversial topics, phrased as parliamentary motions. Each motion is associated with arguments (referred to as "points") supporting or contesting it. Each argument may be comprised of several sentences and is briefly summarized in a *title*.

Selecting iDebate motions At the time of this research, iDebate contained 684 motions.[3] We selected 50 clearly-defined motions, and simplified their phrasing when necessary. For example, we rephrased "This House believes that cannabis should be legalised" to "We should legalize cannabis".

Producing debate speeches We recorded argumentative speeches for each motion. First, two speeches supporting each motion were recorded by two experienced debaters. In doing so, we followed the process described in (Mirkin et al., 2018), where a speaker is presented with a motion and its description and is instructed to record a few minutes speech that supports it, with 10 minutes to prepare, but without checking any online materials. Given a speech supporting the motion, we asked another debater to listen to it and then record a speech rebutting it, and in consequence – opposing the motion. These response speeches are of different nature than the initial speeches beyond the opposite stance, as they often contain references and rebuttal to arguments mentioned in this initial speech.

Through this process, we produced in total 200 speeches recorded by 14 different speakers, equally distributed between the motions, and between the "pro" and "con" stances. Each speech was transcribed automatically using Watson ASR.[4] The transcripts were split automatically into sentences using a bi-directional LSTM that was trained on spoken language corpora (Pahuja et al., 2017). For completeness of the dataset,

---

[3] We accessed iDebate on Jan. 28, 2018.

[4] https://www.ibm.com/watson/services/speech-to-text

for each ASR transcript we include a manually-corrected "reference" human transcript, including manually added punctuation. Based on the human transcripts, we computed the word error rate (WER) of the ASR transcripts: 8.03% on average. For comparison, the WER of the ASR transcripts in (Li et al., 2018) is 22.73%.

Labeling Given the recorded speeches, we carried out a labeling task employing experienced annotators, all of whom are highly proficient English-speakers. Given a motion and a speech, the annotators were instructed to listen to it once, preferably without pausing, and select which ones of iDebate argument titles were mentioned in it, or *None* if none of them was. Specifically, they were instructed to answer positively if it would be correct to say that "the speaker argued that arg ..." where arg is the argument's title. A single question contained all the iDebate titles for the motion, which have the suitable stance for the given speech. Each of the 200 questions was answered by five annotators.

On average, the labeled data contains 4.4 argument titles per speech, where a title contains 10.5 words and an argument text includes 6 sentences and 156 words.

Labeling results In 173 (86.5%) of the speeches, at least one iDebate argument was found, and 248 (∼56%) of the iDebate arguments were labeled as positive at least once.

In order to analyze agreement between annotators, we transformed each multiple-choice question to a set of binary questions containing a speech and a single argument title. This amounted to 878 annotated speech–argument pairs, of which 354 (40.3%) are labeled as positive (i.e. an average of ∼1.8 positive arguments per speech). The average pairwise Cohen's Kappa (Cohen, 1960) score over the labels is 0.52 (0.55 for supporting and 0.50 for opposing speeches). The Fleiss' Kappa (Fleiss, 1971) is 0.52. Noteworthy, 78.5% of the labels were of high confidence: four or five annotators agreed on the label. Figure 1 shows the distribution of positive answers over the binary questions.

We analyzed a sample of arguments–speech pairs that had low agreement between annotators, i.e., those that have two or three positive labels. One reason for disagreement that we identified concerns argument titles that contain two claims, of which only one is argued by the speaker. An example is the title "Gambling is associated with other forms of addiction and harmful behaviour" concerning the motion "We should ban gambling". In a speech about this motion, only "addiction" was mentioned but not "harmful behaviour", resulting with two annotators accepting it and three rejecting it. Another possible source of disagreement is argument titles that are semantically similar, but not identical to the arguments presented in the speech. For instance, two of five annotators accepted the argument "on-line gambling affects families" when the speaker argued on the effects of gambling on families, but did not mention on-line gambling specifically.

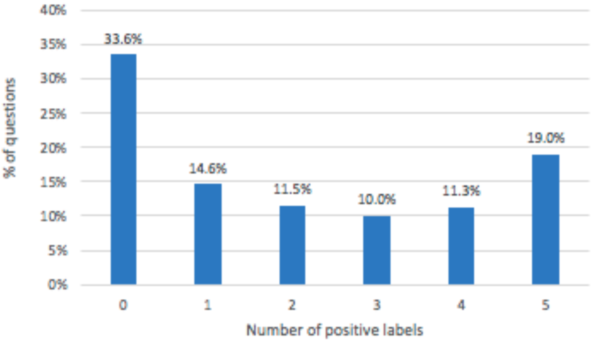

Figure 1: The percentage of binary questions labeled as positive by 0 to 5 annotators.

Listening vs. reading To corroborate the reliability of our labeling through listening, we repeated the task for 40 randomly sampled speeches, replacing only the audio with manual human transcripts of the speeches.[5] The average pairwise Cohen's Kappa score over the labels is 0.59 and the Fleiss' Kappa is 0.60. While these may indicate that the reading task is somewhat easier (e.g. because the annotator can read the text multiple times), it was encouraging to discover that audio-based labeling achieves similar results to text-based labeling: we compared the labels obtained via reading and via listening and found that 87% of them were identical. Labeling by listening is closer to the task of listening comprehension than labeling via reading. Another advantage is that it removes the need to manually transcribe the speeches (in our experience, ASR transcripts are not ideal for labeling). As mentioned, in our

---

[5] We use manual and not ASR transcripts for this analysis, under the assumption that when listening, the annotators are also receiving error-free content.

released dataset we do provide human transcripts for all speeches.

Analysis of potential bias Recent works showed undesired artifacts in natural language inference datasets, namely that in some datasets for inferring relations between two texts, inference can in fact be done by only considering one of them (Schwartz et al., 2017; Gururangan et al., 2018; Tsuchiya, 2018; Poliak et al., 2018). To explore this issue in our dataset, we assessed the correlation between several features of the argument title and the label. Specifically, we computed the Spearman's correlation (Spearman, 1904) between the label and the title's length, occurrences of named entities or negation words in the title, and the correlation between the labels and the titles' 100 most frequent words, stopwords excluded. This resulted in low correlation coefficients, summarized in Table 1. This preliminary analysis suggests that naive features extracted from the title are not sufficient for predicting its label.[6]

Feature	**Spearman ρ**
Title length	-0.07
Named entity	-0.08
Negation	-0.03
Frequent words	0.14

Table 1: Spearman correlation between labels and features of argument titles. For the frequent words, the figure shown is the maximum (absolute) correlation found between positive labels and words from the titles. The words which yielded this correlation were "women" and "environment".

3 Evaluation

Next, we establish baselines for our annotated dataset. All baselines are based on simple unsupervised text similarity methods for selecting which arguments were mentioned in a debate speech. Strictly, this is an entailment task rather than a similarity task, and similarity serves here as an approximation. Below we describe the establishment of the baselines.

Evaluation configurations We considered two ways of representing a speech and two ways of representing an argument, to a total of four different experimental configurations. A speech can be represented as a single text or as the set of its sentences. When a speech is represented by a set of sentences, we considered the maximum similarity score obtained by a sentence in the speech. For the speech text we used the ASR transcripts of the audio speeches, with the automatic split into sentences, as described in Section 2. The sentence-based configurations, therefore, depend on the specific automatic splitting of the texts into sentences. An argument can be represented using only its title or by its extended text. The validity of matching an argument's text against the speech while labels refer to titles stems from entailment transitivity: under the observation that in our data an argument text typically entails its title, if an argument text is mentioned in the speech (i.e. is entailed by it), its title is also entailed by the speech (that is, given that typically $arg \Rightarrow title$, if $speech \Rightarrow arg$ then $speech \Rightarrow title$.)

Evaluation metric The performance of each method is calculated as the average accuracy over speeches in the test set, where the accuracy of a speech is the ratio of correct answers out of the number of choices presented for it. This ensures that each speech contributes equally to the overall accuracy regardless of the number of potential arguments associated with it. Since there is an equal number of speeches per motion, this is also the average accuracy over motions.[7]

Development and test sets We randomly split the data into development and test sets (*dev* and *test* below), such that 60% of the motions (30 motions, 120 speeches, 60 of each of supporting and opposing speeches) are in *dev* and 40% (80 speeches) are in *test*. For each method, we select a threshold that maximizes accuracy over *dev*, and apply it to *test*. In other words, an argument is considered to be mentioned in the speech if its similarity score is above the threshold.

---

[6] We thank the anonymous reviewers for helping us improve this analysis.

[7] We chose accuracy over precision and recall since we wished to give an equal weight to each question; therefore, a micro-average score – where we consider each argument-speech pair as an item in the calculation – was unsuitable. With a small number of pairs per question, one often encounters anomalies: when there are no positive labels, recall is undefined, and when no positive prediction is given for a question, precision is undefined.

3.1 Assessed methods

***All-yes* baseline** As a reference point, we compute the accuracy obtained when all arguments are predicted to be mentioned in the speech, resulting in 39.8% accuracy.

word2vec (*w2v*) We create a w2v (Mikolov et al., 2013) vector representation for each text, removing stopwords; each word is represented by a 200-dimensional word embedding learned over Wikipedia. A *tfidf*-weighted average of the word embeddings represents each text, where *idf* values are counted when considering each Wikipedia sentence as a document. Given a pair of texts, their score is the cosine similarity between their vector representations.

skip-thought (ST) Kiros et al. (2015) presented a general sentence encoder, that has been applied successfully to a variety of tasks such as semantic relatedness and paraphrase detection, often obtaining state of the art results. We use its available implementation to encode the texts as vectors, and compute the cosine similarity between them.

3.2 Results

Table 2 shows the accuracy of all *w2v* configurations. Representing an argument using its more verbose several-sentences-long content outperforms using its short single-sentence title. On the speech side, considering each sentence separately is preferable to using the entire speech. We compared the results of the best *w2v*-based configuration (*arg-sentence*), to the performance of the *skip-thought* auto-encoder. In this setting, encoding individual speech sentences and an argument, the accuracy of *skip-thought* was 60.2%.

Method	Accuracy (%)
all-yes	39.8
w2v title-speech	49.8
w2v arg-speech	57.6
w2v title-sentence	55.8
w2v arg-sentence	64.6
ST arg-sentence	60.2

Table 2: Accuracy results over the test set ASR transcripts, for *w2v* and *skip-thought* (ST).

The highest scoring method, *w2v arg-sentence*, reaches, then, a rather modest accuracy of 64.6%. One weakness of this method, revealed through analysis of its false positive predictions, is its tendency to prefer longer sentences. It is nevertheless substantially superior to the trivial *all-yes* baseline, as well as its *all-no* counterpart.

As explained, we chose accuracy as the main metric for this benchmark as it enables computing macro-average scores over speeches with a variable number of arguments. For reference, the micro-average precision and recall scores of the *w2v arg-sentence* are 57.1% and 43% respectively, with an F_1 score of 49.1%. Optimizing for this metric enables controlling the trade-off between precision and recall with a threshold, depending on the end-application needs.

In sum, we have set a baseline for this task by computing similarity between averaged word embeddings vectors. This simple method can be used as a starting point for future works on this dataset.

4 Conclusions

Machine listening comprehension is a challenging task, whose complexity stems, among other things, from the difficulty to handle spoken language and from errors due to automatic transcription. The argumentation domain, often with complex and elaborate reasoning, relying on presuppositions and world knowledge, adds another dimension to this complexity. In this work, we suggest a task and a corresponding benchmark dataset to assess comprehension in this domain. We focused on the task of confirming the occurrence of arguments in a speech, which – as shown in this work – can be handled to some degree with standard textual inference methods. Other types of questions can be formulated over this data in following work. We release a rich dataset, accompanied with benchmarks, that can drive various studies in listening comprehension and argumentation mining.

5 Acknowledgments

We are thankful to the debaters and annotators who took part in the creation of this dataset.

References

F. Boltužić and J. Šnajder. 2014. Back up your Stance: Recognizing Arguments in Online Discussions. In *Proceedings of the First Workshop on Argumentation Mining, ACL.*

Yu-An Chung and James R. Glass. 2018. Speech2Vec: A Sequence-to-Sequence Framework for Learning Word Embeddings from Speech. *CoRR*, abs/1803.08976.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46.

Joseph L. Fleiss. 1971. Measuring Nominal Scale Agreement among Many Raters. *Psychological Bulletin*, 76(5):378–382.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation Artifacts in Natural Language Inference Data. *Proceedings of NAACL*.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. The Argument Reasoning Comprehension Task: Identification and Reconstruction of Implicit Warrants. In *Proceedings of NAACL*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Proceedings of NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *CoRR*, abs/1511.02301.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *NIPS*.

Lori Lamel, Sophie Rosset, Christelle Ayache, Djamel Mostefa, Jordi Turmo, and Pere Comas. 2008. Question Answering on Speech Transcriptions: the QAST evaluation in CLEF. In *Proceedings of LREC*.

Chia-Hsuan Li, Szu-Lin Wu, Chi-Liang Liu, and Hung-yi Lee. 2018. Spoken SQuAD: A Study of Mitigating the Impact of Speech Recognition Errors on Listening Comprehension. In *Proceedings of Interspeech*.

Marco Lippi and Paolo Torroni. 2016. Argument Mining from Speech: Detecting Claims in Political Debates. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Shachar Mirkin, Michal Jacovi, Tamar Lavee, Hong-Kwang Kuo, Samuel Thomas, Leslie Sager, Lili Kotlerman, Elad Venezian, and Noam Slonim. 2018. A recorded debating dataset. In *Proceedings of LREC*.

Vardaan Pahuja, Anirban Laha, Shachar Mirkin, Vikas Raykar, Lili Kotlerman, and Guy Lev. 2017. Joint Learning of Correlated Sequence Labelling Tasks Using Bidirectional Recurrent Neural Networks.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis Only Baselines in Natural Language Inference.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. In *Proceedings of EMNLP*.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Li Zilles, Yejin Choi, and Noah A Smith. 2017. The Effect of Different Writing Tasks on Linguistic Style: A Case Study of the ROC Story Cloze Task. *Proceedings of CoNLL*.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101.

Bo-Hsiang Tseng, Sheng-Syun Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Towards Machine Comprehension of Spoken Content: Initial TOEFL Listening Comprehension Test by Machine. In *Proceedings of Interspeech*.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *Proceedings of LREC*.

Jordi Turmo, Pere Comas, Christelle Ayache, Djamel Mostefa, Sophie Rosset, and Lori Lamel. 2007. Overview of qast 2007. In *Proceedings of CLEF*.

Using Active Learning to Expand Training Data for Implicit Discourse Relation Recognition

Yang Xu **Yu Hong**[*] **Huibin Ruan** **Jianmin Yao** **Min Zhang** **Guodong Zhou**
Institute of Artificial Intelligence, Soochow University
School of Computer Science and Technology, Soochow University
No.1, Shizi ST, Suzhou, China, 215006
{andreaxu41, tianxianer, hbr416}@gmail.com
{jmyao}@szkj.gov.cn
{minzhang, gdzhou}@suda.edu.cn

Abstract

We tackle discourse-level relation recognition, a problem of determining semantic relations between text spans. Implicit relation recognition is challenging due to the lack of explicit relational clues. The increasingly popular neural network techniques have been proven effective for semantic encoding, whereby widely employed to boost semantic relation discrimination. However, learning to predict semantic relations at a deep level heavily relies on a great deal of training data, but the scale of the publicly available data in this field is limited. In this paper, we follow Rutherford and Xue (2015) to expand the training data set using the corpus of explicitly-related arguments, by arbitrarily dropping the overtly presented discourse connectives. On the basis, we carry out an experiment of sampling, in which a simple active learning approach is used, so as to take the informative instances for data expansion. The goal is to verify whether the selective use of external data not only reduces the time consumption of retraining but also ensures a better system performance. Using the expanded training data, we retrain a convolutional neural network (CNN) based classifer which is a simplified version of Qin et al. (2016)'s stacking gated relation recognizer. Experimental results show that expanding the training set with small-scale carefully-selected external data yields substantial performance gain, with the improvements of about 4% for accuracy and 3.6% for F-score. This allows a weak classifier to achieve a comparable performance against the state-of-the-art systems.

1 Introduction

Since the Penn Discourse Treebank of version 2.0 (PDTB) was released in 2008 (Prasad et al., 2008), there is a significant amount of research has been carried out on discourse-level relation recognition between a variety of text spans (namely, argument-argument relations). From a perspective of inclusion or omission of conjunctions, the study in this field has been directed toward two issues: recognizing explicit relations or implicit. Listed below are two pairs of arguments, where the arguments in 1) hold an explicit causal relation while those in 2) are implicitly related with a causal relation.

> 1) [She left the company]$_{Arg1}$ **because** [she would move to California]$_{Arg2}$.
>
> 2) [We have never seen the kitty since then]$_{Arg1}$. [John told us the kitty has been adopted]$_{Arg2}$.

In general, the explicit relations can be directly signaled by the conjunctions (also called connectives) which inherently exist, such as the conjunction "*because*" in 1). This allows a predictor to speculate relations by the word senses of conjunctions. Using conjunctions as relational predicates, the earlier study has achieved a prediction performance of no less than 93% for accuracy (Pitler and Nenkova, 2009). By contrast, the implicit relations like that in 2) are difficult to automatically recognize due to the lack of conjunctions.

We focus on the implicit relation recognition in this paper, and follow Rutherford and Xue (2015) to strengthen the current neural discourse-level relation classification by expanding the training data set. The explicit-to-implicit (Exp2Imp for short) relation transformation is used. In particular, we propose to introduce active learning into the data expansion process, with the aim to reduce redundancy and reinforce the use of informative instances. In our experiments, we cooperate active learning with a simple version of Exp2Imp transformation. Experimental results show that such a cooperation allows substantial improvements to be achieved for 4 main relation types in PDTB.

---

[*] Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 725–731
Brussels, Belgium, October 31 - November 4, 2018.

2 Related Work

Multi-class implicit relation recognition can be boiled down to a classification problem. This encourages the study of supervised classification at the earlier time (Pitler and Nenkova, 2009; Lin et al., 2009; Louis et al., 2010; Park and Cardie, 2012; Rutherford and Xue, 2014). Recently, the neural network based approaches become increasingly popular due to the capacity of deep semantic learning and understanding (Zhang et al., 2015; Qin et al., 2016; Chen et al., 2016; Qin et al., 2017; Liu and Li, 2016). However, a large amount of labeled data is urgently needed to train the models. (Rutherford and Xue, 2015; Braud and Denis, 2016; Liu et al., 2016; Wu et al., 2017).

The explicitly-related arguments in the corpus of PDTB has been sufficiently proven to be usable for creating implicitly-related arguments (Rutherford and Xue, 2015; Braud and Denis, 2016; Liu et al., 2016; Wu et al., 2017), only if the omission of inherent conjunctions will not distort the original semantic relations (Rutherford and Xue, 2015). Benefiting from the high-accuracy explicit relation recognition, a simple pattern, such as `Argument1`+*because*+`Argument2`, may enable the acquisition of countless explicitly-related arguments from texts. It makes it possible to cooperate with Rutherford and Xue (2015)'s Exp2Imp relation transformation for creating a tremendous number of labeled implicit relation instances.

However, it is inevitable to bring in the redundant information when using such a scale of unrefined artificial instances to directly expand the existing training data. This causes a time-consuming retraining process. By contrast, random sampling for retrenchment may leave informative instances out of the expanded data set.

Active learning (AL) is especially applicable to redundancy reduction. It is able to automatically select the most informative samples for use in a cycle of training. Nowadays, there have been a variety of AL models successfully used in different language processing tasks (Li and Guo, 2013; Guo and Wang, 2015; Yang et al., 2015; Zhang et al., 2017; Ramirez-Loaiza et al., 2017). This allows us to draw lessons from the experiences.

3 Informative Instances

In general, in the field of machine learning, an informative instance is defined as the one which has been classified with less confidence (i.e., higher uncertainty). Assume that $I_{r_j}(x_i, M)$ refers to the the level of uncertainty at which a classifier *M* determines an instance x_i as the member of the class r_j, thus x_i is informative only if it significantly increases the level of uncertainty:

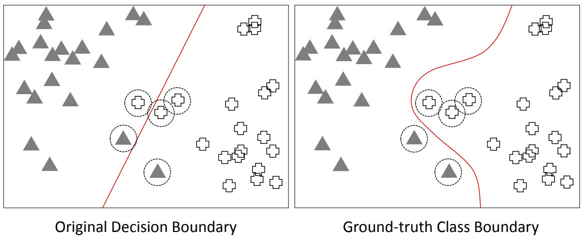

Figure 1: Examples of informative instances

$$x^* = \arg\max \sum_{r_j \in R} I_{r_j}(x_i; M) \qquad (1)$$

The utilization of informative instances in data expansion for training has been proven effective in improving the fully-supervised classification models. For example, the instances marked with a dot-dashed circle in Figure 1 can be regarded to be informative. It is because a classifier may fail to deterministically distinguish between them. If using such instances as additional training data, we may retrain the classifier to revise the original decision boundary, and pursue the ground-truth.

4 Active Learning (AL)

AL is a kind of retraining mechanism by data expansion. Figure 2 shows the workflow, which primarily includes 4 steps:

- **Step 1**: in which a learning model is required to be trained on the previously labeled data.
- **Step 2**: where the well-trained model is used to classify the unlabeled external data.
- **Step 3**: relies on the classification results to evaluate informativeness over the unlabeled data. The informative instances will be eventually adopted for manual annotation.
- **Step 4**: adds the newly annotated data to the existing, and retrains the learning model.

It is noteworthy that AL is not an "once-for-all" deal. On the contrary, it needs to be carried out repeatedly and iteratively, until a predefined condition is met, such as the time at which the classification performance remains almost the same within

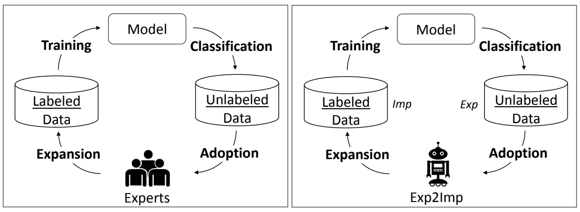

General AL workflow   Cooperating AL with Exp2Imp

Figure 2: Workflows of ALs

a series of successive iterations, or termination after a fixed number of iterations.

5 Cooperating AL with Exp2Imp

We cooperate AL with Exp2Imp transformation. The workflow is also shown in Figure 2, where the labeled data set contains a certain number of implicitly-related argument pairs, while the unlabeled the explicitly-related.

In each iteration of AL, a classifier is trained on the labeled implicitly-related argument pairs, but it is driven to forcibly classify the explicitly-related argument pairs. On the basis, informativeness measurement is performed to sample the informative explicitly-related argument pairs. And then, instead of experts, the Exp2Imp transformation module serves as an annotator to mark the relations of the sampled argument pairs. Along with the automatically-annotated relations, such argument pairs are used as the counterfeit implicit relation instances. They are eventually added to the original labeled data set for expansion.

5.1 Informativeness Measurement

We employ an uncertainty sampling function (Zhu et al., 2008; Settles, 2010; Yang et al., 2015; Ramirez-Loaiza et al., 2017) to measure the informativeness:

$$Inf(x_i) = \sum_{r_j \in R} I_{r_j}(x_i; M) \qquad (2)$$

$$= \sum_{r_j \in R} P(r_j|x_i) \log P(r_j|x_i) \qquad (3)$$

where, for an instance x_i, the entropy of the predicted probabilities over all kinds of PDTB relation classes is used as the score of informativeness.

In order to reduce the computational complexity in practice, we sample informative instances in an iteration-independent batch-by-batch manner. In each iteration, a batch of instances in the unlabeled data set will be taken, only if their informativeness scores are higher than a constant threshold θ:

$$U' = \{x_i \mid Inf(x_i) > \theta, \forall x_i \in U\} \qquad (4)$$

where, U is the unlabeled data set while U' consists of the potentially informative instances.

Learning rate	0.001	Filter size	$(2, 3, 5)$
Number of filter	512	Optimizer	Adam
Batch size	128	threshold θ	0.95

Table 1: Hyperparameter settings of CNN

5.2 Statistical Information based Exp2Imp

We implement a much simpler Exp2Imp transformation model than Rutherford and Xue (2015)'s work. Statistical information is used.

For an explicitly-related argument pair in U, we rely on the conjunction to determine the relation. For example, if the arguments are syntactically connected by the conjunction "*because*", their relation will be identified as `Contingency` (`Causality` plus `Condition`). We previously look up conjunctions in a small subset of U and analysis the ground-truth explicit relations they signal. If a conjunction invariably signals a single type of relation (e.g., because→ `Contingency`), we preserve the one-to-one correspondence between the conjunction and the relation type. Else if a conjunction used to signal multiple-type relations, we align it solely with the relation it most frequently signals ever.

The Exp2Imp module first determines explicit relations based on conjunction-relation alignment, and then omits the conjunctions to create the counterfeit implicitly-related argument pairs.

5.3 CNN based Classification

We follow Qin et al. (2016) to use Siamese CNN for argument modeling and relation classification. The 300-dimensional word embeddings and 50-dimensional POS embeddings are used to represent the arguments. We also follow Mikolov et al. (2013) to pretrain the word embeddings and initialize the POS by random sampling in [-1,1]. Table 1 shows the hyperparameter settings.

The source codes of AL, Exp2Imp and Siamese CNN[1] to fully reproduce the experiments has been made publicly available, to which we attached the set of informative training data.

---

[1] https://github.com/AndreaXu0401/ALIDRC

Metrics		Baseline	Blender (U)	AL (U)
Tem.	P	61.54	66.67	81.82
	R	14.55	14.54	16.36
	F_1	23.53	23.88	**27.27**
Com.	P	38.74	72.72	82.50
	R	29.66	17.94	22.76
	F_1	33.59	28.73	**35.68**
Con.	P	53.15	45.64	59.00
	R	27.84	32.60	34.80
	F_1	36.54	38.03	**43.78**
Exp.	P	60.08	55.98	59.57
	R	83.09	79.92	88.48
	F_1	69.73	65.85	**71.21**
Accuracy		56.78	54.70	**60.63**
Macro F_1		40.85	39.12	**44.48**

Table 2: The four-way classification performance

6 Experimentation

We experiment on the PDTB v2.0 (Prasad et al., 2008). Sections 2-20 are used as the benchmark training set. They are also used as the labeled set in the AL process. Sections 21-22 are taken as the test set and sections 0-1 the development set.

The ground-truth explicitly-related argument pairs in the PDTB corpus are divided into two subsets: one consists of 450 instances which are used for aligning conjunctions and relation types, the other is consisted of 17,000 instances and used as the unlabeled data set U in the AL process.

6.1 Main Results

For the purpose of comparison, we expand the benchmark training data set (**Baseline**) in two different ways: **Blender** and **AL**. Blender combines the benchmark with U. Exp2Imp transformation is performed for the instances in U. Thus Blender mixes the true implicitly-related argument pairs and all the counterfeits in U. By contrast, AL samples informative counterfeits for expansion.

We train CNN on the benchmark and the expanded versions, and test the best-developed models for four-way classification among the relation types of `Expansion` (**Exp**), `Contingency` (**Con**), `Comparison` (**Com**) and `Temporality` (**Tem**). Table 2 shows the performance. It can be observed that simply adding all the counterfeits to the training data set negatively influences the learning process, causing a performance loss of about 1.73% for macro F_1 and 2.08% for accuracy. This may result from the fact that the relations of some counterfeits change to be uncertain due to the omission of inherent conjunctions. Such counterfeits probably mislead CNN during the learning process.

Systems	Accuracy	Macro F_1
IO (2015)	57.10	40.50
MNN (2016)	57.27	44.98
MTN (2016)	-	42.50
DSWE (2017)	58.85	44.84
MANN (2017)	57.39	**47.80**
Ours	**60.63**	44.48

Table 3: Comparison with the state of the art

On the contrary, AL obtains a substantial performance gain. Using about 15 percent of instances in U for expansion, AL improves CNN with 3.85% for accuracy and 3.63% for macro F_1. This may imply that the carefully-selected counterfeits by AL most probably have positive effects on the retraining. The uncertainty caused by arbitrary conjunction omission plays a counter-productive role, prompting CNN to pursue more precise classification boundaries in the AL process.

6.2 Discussion

6.2.1 Comparison to Expansion Methods

We compare the proposed method to the recently popular, all of which more or less expand the training data, so as to introduce comprehensive linguistic knowledge into the learning process:

- Intelligent Omission (**IO**) (Rutherford and Xue, 2015) consciously omits conjunctions in explicitly related argument pairs to create counterfeits. Herein, a model is developed to identify the counterfeits whose semantic relations keep unchanged after conjunction omission. Such samples are used for expansion.

- Multi-task Neural Network (**MNN**) (Liu et al., 2016). The training set is expanded with RST-DT (Carlson et al., 2003) and NYT (Sandhaus, 2008). The other two MNN based models include Wu et al. (2016)'s **MTN** and Lan et al. (2017)'s **MANN**. MANN is more sophisticated due to the use of attention mechanism, while MTN acts as a bilingual discourse analysis system. Both of them introduce external datasets in the training processes, NANT and BiSynData, respectively .

- **DSWE** based CNN model (Wu et al., 2017), where Discourse-Specific Word Embeddings (DSWE) are used. DSWE are pre-trained on the large English Gigaword corpus.

It can be found that our method achieves competitive performance, showing the best accuracy and almost comparable F score to those of most competitors. It is noteworthy that the neural network we use is much weaker than the sophisticated MNN, MTN and MANN. Besides, there are fewer instances used for expansion than that for training DSWE. This may imply that AL is cost-effective.

6.2.2 Informative Instances

We randomly sample a couple of informative instances by tracking the AL process, along with some less informative cases. They helps illustrate the intuition that informativeness verification benefits data cleaning, in the process of using counterfeits for distant supervision.

> 3) [He further said that it would study other alternatives]$_{Arg1}$ **omitted conjunction** [it hasn't yet made any proposals to shareholders]$_{Arg2}$.
>
> Conjunction: **however**
> Relation: `Comparison.Contrast`
>
> 4) [all of a sudden you are relegated to a paltry sum]$_{Arg1}$ **omitted conjunction** [you become a federal judge]$_{Arg2}$.
>
> Conjunction: **when**
> Relation:`Temporality.Synchrony`

The instances listed above are to some extent informative. When the conjunctions inherently connecting the arguments were pruned off, the semantic relations changed to be uncertain. For example, assume the conjunctions "*however*" and "*when*" are respectively replaced by "*therefore*" and "*although*", the arguments in 3) will appear to have a `Contingency.Causality` relation as well, and those in 4) seems to hold a `Contingency.Compromise` relation with a very reasonable possibility. It means that the relations the argument pair hold are ambiguous if there isn't any manually-edited explicit relational signal (e.g., conjunction). Nevertheless, from the other perspective, such instances are frankly useful for training a classifier. It is because:

- Learning to distinguish the relations of such arguments is challenging beyond other cases.
- A challenging task enables the training process to be more strict but effective.
- This makes it possible to learn the subtle differences among argument pairs which belong to different relation classes.

On the contrary, the instances listed below are far from informative. There are less alternative relations can be imaged to replace the original, even if the conjunctions "*although*" and "*because*" have been pruned off. Note that such instances are undoubtedly useful at the very beginning of the training process. At that time, they facilitate the initialization of fuzzy classification boundaries. However, they are most probably useless for calibrating the boundaries at the level of subtle difference.

> 5) [She was dreadful to her war-damaged husband]$_{Arg1}$ **omitted conjunction** [she was kind and playful to her children]$_{Arg2}$.
>
> Conjunction: **although**
> Relation: `Comparison.Contrast`
>
> 6) [The ad was devastating]$_{Arg1}$. **omitted conjunction** [it raised questions about Mr. Courter's credibility]$_{Arg2}$.
>
> Conjunction: **because**
> Relation: `Contingency.Cause`

7 Conclusion

This paper demonstrates the contributions of AL to the enhancement of implicit relation classification. Using an AL model, we successfully sample a batch of informative explicitly-related argument pairs. Following Exp2Imp strategy, we convert the arguments into implicitly-related cases. This helps expand training data for fully-supervised relation modeling. By retraining, we enable a weak CNN model to achieve competitive performance.

Active learning can be systematically cooperated with Rutherford and Xue (2015)'s intelligent omission. This will enable the sampling of both informative and reliable instances for data expansion. Besides, using the carefully expanded training data set, the sophisticated learning model like MNN may be further enhanced significantly.

Acknowledgments

This work was supported by the national Natural Science Foundation of China (NSFC) via Grant Nos. 61672368, 61672367, 61525205.

References

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 571–583. Association for Computational Linguistics.

Chloé Braud and Pascal Denis. 2016. Learning connective-based word representations for implicit discourse relation identification. In *EMNLP*, pages 203–213.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *ACL (1)*.

Fengyu Guo, Ruifang He, Di Jin, Jianwu Dang, Longbiao Wang, and Xiangang Li. 2018. Implicit discourse relation recognition using neural tensor network with interactive attention and sparse learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 547–558. Association for Computational Linguistics.

Husheng Guo and Wenjian Wang. 2015. An active learning-based svm multi-class classification model. *Pattern recognition*, 48(5):1577–1597.

Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu, and Haifeng Wang. 2017. Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1299–1308.

Xin Li and Yuhong Guo. 2013. Active learning with multi-label svm classification. In *IJCAI*, pages 1479–1485.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 343–351. Association for Computational Linguistics.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. *arXiv preprint arXiv:1609.06380*.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *AAAI*, pages 2750–2756.

Annie Louis, Aravind Joshi, Rashmi Prasad, and Ani Nenkova. 2010. Using entity features to classify implicit discourse relations. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 59–62. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 108–112. Association for Computational Linguistics.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 13–16. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2263–2270.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric P Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. *arXiv preprint arXiv:1704.00217*.

Maria E Ramirez-Loaiza, Manali Sharma, Geet Kumar, and Mustafa Bilgic. 2017. Active learning: an empirical study of common baselines. *Data Mining and Knowledge Discovery*, 31(2):287–313.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *EACL*, volume 645, page 2014.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *HLT-NAACL*, pages 799–808.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

Changxing Wu, Yidong Chen, Yanzhou Huang, et al. 2016. Bilingually-constrained synthetic data for implicit discourse relation recognition. In *Proceedings* *of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2306–2312.

Changxing Wu, Xiaodong Shi, Yidong Chen, Jinsong Su, and Boli Wang. 2017. Improving implicit discourse relation recognition with discourse-specific word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 269–274, Vancouver, Canada. Association for Computational Linguistics.

Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. 2015. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *EMNLP*, pages 2230–2235.

Ye Zhang, Matthew Lease, and Byron C Wallace. 2017. Active discriminative text representation learning. In *AAAI*, pages 3386–3392.

Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K Tsou. 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1137–1144. Association for Computational Linguistics.

Learning To Split and Rephrase From Wikipedia Edit History

Jan A. Botha[*] **Manaal Faruqui**[*] **John Alex** **Jason Baldridge** **Dipanjan Das**

{jabot,mfaruqui,jpalex,jasonbaldridge,dipanjand}@google.com

Google AI Language

Abstract

Split and rephrase is the task of breaking down a sentence into shorter ones that together convey the same meaning. We extract a rich new dataset for this task by mining Wikipedia's edit history: WikiSplit contains one million naturally occurring sentence rewrites, providing sixty times more distinct split examples and a ninety times larger vocabulary than the WebSplit corpus introduced by Narayan et al. (2017) as a benchmark for this task. Incorporating WikiSplit as training data produces a model with qualitatively better predictions that score 32 BLEU points above the prior best result on the WebSplit benchmark.

A classic leaf symptom is water-soaked lesions between the veins which appear as angular leaf-spots where the lesion edge and vein meet.

A classic leaf symptom is the appearance of angular, water-soaked lesions between the veins. The angular appearance results where the lesion edge and vein meet.

Figure 1: A split-and-rephrase example extracted from a Wikipedia edit, where the top sentence had been edited into two new sentences by removing some words (yellow) and adding others (blue).

1 Introduction

A complex sentence can typically be rewritten into multiple simpler ones that together retain the same meaning. Performing this *split-and-rephrase* task is one of the main operations in text simplification, alongside paraphrasing and dropping less salient content (Siddharthan, 2006; Zhu et al., 2010; Woodsend and Lapata, 2011, i.a.). The area of automatic text simplification has received a lot of attention (Siddharthan, 2014; Shardlow, 2014), yet still holds many open challenges (Xu et al., 2015). Splitting sentences in this way could also benefit systems where predictive quality degrades with sentence length, as observed in, e.g., relation extraction (Zhang et al., 2017) and translation (Koehn and Knowles, 2017). And the schema-free nature of the task may allow for future supervision in the form of crowd-sourced rather than expensive expert annotation (He et al., 2015).

Narayan et al. (2017) introduce the WebSplit corpus for the split-and-rephrase task and report results for several models on it. Aharoni and Goldberg (2018) improve WebSplit by reducing overlap in the data splits, and demonstrate that neural encoder-decoder models (Bahdanau et al., 2014) perform poorly, even when enhanced with a copy mechanism (Gu et al., 2016; See et al., 2017).

One limitation of the WebSplit examples themselves is that they contain fairly unnatural linguistic expression using a small vocabulary. We introduce new training data mined from Wikipedia edit histories that have some noise, but which have a rich and varied vocabulary over naturally expressed sentences and their extracted splits. Figure 1 gives an example of how a Wikipedia editor rewrote a single sentence into two simpler ones. We create WikiSplit, a set of one million such examples mined from English Wikipedia, and show that models trained with this resource produce dramatically better output for split and rephrase.

Our primary contributions are:

- A scalable, language agnostic method for extracting split-and-rephrase rewrites from Wikipedia edits.
- Public release of the English WikiSplit dataset, containing one million rewrites: http://goo.gl/language/wiki-split
- By incorporating WikiSplit into training, we more than double (30.5 to 62.4) the BLEU score obtained on WebSplit by Aharoni and Goldberg (2018).

---

[*] Both authors contributed equally.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 732–737
Brussels, Belgium, October 31 - November 4, 2018.

Correct
Street Rod is the first in a series of two games released for the PC and Commodore 64 in 1989. Street Rod is the first in a series of two games. It was released for the PC and Commodore 64 in 1989.
He played all 60 minutes in the game and rushed for 114 yards, more yardage than all the Four Horsemen combined. He played all 60 minutes in the game. He rushed for 114 yards, more yardage than all the Four Horsemen combined.
Unsupported
When the police see Torco's injuries, they send Ace to a clinic to be euthanized, but he escapes and the clinic worker covers up his incompetence. When the police see Torco's injuries to his neck, they believe it is a result of Ace biting him. They send Ace to a clinic to be euthanized, but he escapes and the clinic worker covers up his incompetence.
Missing
The avenue was extended to Gyldenløvesgade by Copenhagen Municipality in 1927-28 and its name was changed to Rosenørns Allé after Ernst Emil Rosenørn (1810-1894) . The avenue was extended to Gyldenløvesgade by Copenhagen Municipality in 1927-28. The street was named after Ernst Emil Rosenørn (1810-1894) .

Table 1: Examples of correct and noisy sentence splits extracted from Wikipedia edits. Noise from unsupported or missing statements is visualized with the same coloring as in Figure 1.

2 The WikiSplit Corpus

WebSplit provides a basis for measuring progress on splitting and rephrasing sentences. However, its small size, inherent repetitiveness, and synthetic nature limit its broader applicability. In particular, we see it as a viable benchmark for evaluating models, but not for training them. To that end, we introduce the WikiSplit corpus and detail its construction next.

2.1 Mining Wikipedia Edits

Wikipedia maintains snapshots of entire documents at different timestamps, which makes it possible to reconstruct edit histories for documents. This has been exploited for many NLP tasks, including sentence compression (Yamangil and Nelken, 2008), text simplification (Yatskar et al., 2010; Woodsend and Lapata, 2011; Tonelli et al., 2016) and modeling semantic edit intentions (Yang et al., 2017).

To construct the WikiSplit corpus, we identify edits that involve sentences being split. A list of sentences for each snapshot is obtained by stripping HTML tags and Wikipedia markup and running a sentence break detector (Gillick, 2009). Temporally adjacent snapshots of a Wikipedia page are then compared to check for sentences that have undergone a split like that shown in Figure 1. We search for splits in both temporal directions.

Given all candidate examples extracted this way, we use a high-precision heuristic to retain only high quality splits. To extract a full sentence C and its candidate split into $S = (S_1, S_2)$, we require that C and S_1 have the same trigram prefix, C and S_2 have the same trigram suffix, and S_1 and S_2 have different trigram suffixes. To filter out misaligned pairs, we use BLEU scores (Papineni et al., 2002) to ensure similarity between the original and the split versions.

Thresh. δ	Correct	Unsupp.	Miss.	Size
0.1	161	35	6	1.4m
0.2	168	35	4	1.0m
0.3	169	31	4	0.5m

Table 2: Quality vs corpus size trade-off when setting the similarity threshold. The counts are for a random sample of 100 split-and-rephrase examples extracted using our method (i.e., 200 simple sentences). Keys: **Unsupp**orted; **Miss**ing

Specifically, we discard pairs where BLEU(C, S_1) or BLEU(C, S_2) is less than δ (an empirically chosen threshold). If multiple candidates remain for a given sentence C, we retain $\arg\max_S (\text{BLEU}(C, S_1) + \text{BLEU}(C, S_2))$.[1]

2.2 Corpus Statistics and Quality

Our extraction heuristic is imperfect, so we manually assess corpus quality using the same categorization schema proposed by Aharoni and Goldberg (2018); see Table 1 for examples of *correct*, *unsupported* and *missing* sentences in splits extracted from Wikipedia. We do this for 100 randomly selected examples using three different

---

[1] We attempted to mitigate other noise inherent in Wikipedia by removing items that 1) repeated a token more than three times in a row; 2) contained a token longer than 25 characters; 3) were suggestive of profane vandalism.

	WebSplit		WikiSplit	
	Count	Unique	Count	Unique
C	1.3m	17k	1.0m	1.0m
S'	6.1m	28k	2.0m	1.9m
t	344k	7k	33.1m	633k

Table 3: Training corpus statistics in terms of complex sentences (C), simple sentences ($S' = \cup_i S_i$) and tokens (t, appearing across unique complex sentences). WikiSplit provides much greater diversity and scale.

thresholds of δ. As shown in Table 2, δ=0.2 provides the best trade-off between quality and size.

Out of the 100 complex sentences in the sample, only 4 contained information that was not completely covered by the simple sentences. In our corpus, every complex sentence is split into two simpler sentences, so the sample contains 200 simple sentences. Out of these we found 168 (84%) to be correct, while 35 (18%) contained unsupported facts. Thus, for the overall sample of 100 split-and-rephrase examples, 68% are perfect while 32% contain some noise (either unsupported facts or missing information). We stress that our main goal is to use data extracted this way as training data and accept that its use for evaluation is an imperfect signal with some inherent noise and bias (by construction).

After extraction and filtering, we obtain over one million examples of sentence splits from around 18 million English documents. We randomly reserved 5000 examples each for tuning, validation and testing, producing 989,944 unique complex training sentences, compared to the 16,938 of WebSplit (cf. Table 3).

2.3 Comparison to WebSplit

Narayan et al. (2017) derived the WebSplit corpus by matching up sentences in the WebNLG corpus (Gardent et al., 2017) according to partitions of their underlying meaning representations (RDF triples). The WebNLG corpus itself was created by having crowd workers write sentential realizations of one or more RDF triples. The resulting language is often unnatural, for example, *"Akeem Dent once played for the Houston Texans team which is based in Houston in Texas."*[2]

Repetition arises because the same sentence fragment may appear in many different examples. This is to be expected given that WebSplit's small vocabulary of 7k words must account for the 344k tokens that make up the distinct complex sentences themselves.[3]

This is compounded in that each sentence contains a named entity by construction. In contrast, our large new WikiSplit dataset offers more natural and diverse text (see examples in Table 1), having a vocabulary of 633k items covering the 33m tokens in its distinct complex sentences.

The task represented by our WikiSplit dataset is *a priori* both harder and easier than that of the WebSplit dataset – harder because of the greater diversity and sparsity, but potentially easier due to the uniform use of a single split.

Of the two datasets, WebSplit is better suited for evaluation: its construction method guarantees cleaner data than is achieved by our extraction heuristic, and it provides multiple reference decompositions for each complex sentence, which tends to improve the correlation of automatic metrics with human judgment in related text generation tasks (Toutanova et al., 2016).

3 Experiments

In order to understand how WikiSplit can inform the split-and-rephrase task, we vary the composition of the training set when training a fixed model architecture. We compare three training configurations: WEBSPLIT only, WIKISPLIT only, and BOTH, which is simply their concatenation.

Text-to-text training instances are defined as all the unique pairs of (C, S), where C is a complex sentence and S is its simplification into multiple simple sentences (Narayan et al., 2017; Aharoni and Goldberg, 2018). For training, we delimit the simple sentences with a special symbol. We depart from the prior work by only using a subset of the WebSplit training set: we take a fixed sub-sample such that each distinct C is paired with a single S, randomly selected from the multiple possibilities in the dataset. This scheme produced superior performance in preliminary experiments.

As a quality measure, we report multi-reference *corpus-level* BLEU[4] (Papineni et al., 2002), but

---

[2] Given RDF triple: {(H_Txns, city, Texas), (Akeem_Dent, formerTeam, H_Txns), (H_Txns, city, Houston)}.

[3] We use **WebSplit v1.0** throughout, which is the scaled-up re-release by Narayan et al. (2017) at `http://github.com/shashiongithub/Split-and-Rephrase`, commit `a9a288c`. Preliminary experiments showed the same trends on the smaller v0.1 corpus, as resplit by Aharoni and Goldberg (2018).

[4] Using NLTK v3.2.2, with case sensitive scoring.

↓train/eval→	WebSplit 1.0	WikiSplit
SOURCE	58.0	73.4
SPLITHALF	54.9	71.7
WEBSPLIT	35.3	4.2
WIKISPLIT	59.4	76.0
BOTH	**61.4**	**76.1**

Table 4: Corpus-level BLEU scores on the validation sets for the same model architecture trained on different data.

	BLEU	sBLEU	#S/C	#T/S
Reference		–	2.5	10.9
SOURCE	58.7	56.1	1.0	20.5
SPLITHALF	55.7	53.0	2.0	10.8
AG18	30.5	25.5	2.3	11.8
WEBSPLIT	34.2	30.5	2.0	8.8
WIKISPLIT	60.4	58.0	2.0	11.2
BOTH	**62.4**	**60.1**	2.0	11.0

Table 5: Results on the WebSplit v1.0 test set when varying the training data while holding model architecture fixed: corpus-level BLEU, sentence-level BLEU (to match past work), simple sentences per complex sentence, and tokens per simple sentence (micro-average). AG18 is the previous best model by Aharoni and Goldberg (2018), which used the full WebSplit training set, whereas we downsampled it.

include sentence-level BLEU (sBLEU) for direct comparison to past work.[5] We also report length-based statistics to quantify splitting.

We use the same sequence-to-sequence architecture that produced the top result for Aharoni and Goldberg (2018), "Copy512", which is a one-layer, bi-directional LSTM (cell size 512) with attention (Bahdanau et al., 2014) and a copying mechanism (See et al., 2017) that dynamically interpolates the standard word distribution with a distribution over the words in the input sentence. Training details are as described in the Appendix of Aharoni and Goldberg (2018) using the OpenNMT-py framework (Klein et al., 2017).[6]

3.1 Results

We compare to the SOURCE baseline, which is the previously reported method of taking the unmodified input sentence as prediction, and we add SPLITHALF, the natural baseline of deterministically splitting a complex sentence into two equal-length token sequences and appending a period to the first one.

Table 4 compares our three training configurations on the validation sets of both WebSplit and WikiSplit. The WEBSPLIT model scores 35.3 BLEU on the WebSplit validation set but fails to generalize beyond its narrow domain, as evidenced by reaching only 4.2 BLEU on the WikiSplit validation set.

The example predictions in Table 7 illustrate how this model tends to drop content ("Alfred Warden", "mouth", "Hamburg"), hallucinate common elements from its training set ("food", "ingredient", "publisher") and generally fails to produce coherent sentences.

In contrast, the WIKISPLIT model achieves 59.4 BLEU on the WebSplit validation set, *without observing any in-domain data.* It also outperforms the two deterministic baselines on both validation sets by a non-trivial BLEU margin. This indicates that the WikiSplit training data enable better generalization than when using WebSplit by itself. Reintroducing the downsampled, in-domain training data (BOTH) further improves performance on the WebSplit evaluation.

These gains in BLEU from using WikiSplit carry over to the blind manual evaluation we performed on a random sample of model predictions on the WebSplit validation set. As shown in Table 6, the BOTH model produced the most accurate output (95% correct simple sentences), with the lowest incidence of missed or unsupported statements. Our manual evaluation includes the corresponding outputs from Aharoni and Goldberg (2018) (AG18), which were 22% accurate.

The examples in Table 7 demonstrate that the WIKISPLIT and BOTH models produce much more coherent output which faithfully rephrases the input. In Example 1, the combined model (BOTH) produces three fluent sentences, overcoming the strong bias toward two-sentence output inherent in the majority of its training examples.

We relate our approach to prior work on WebSplit v1.0 by reporting scores on its test set in Table 5. Our best performance in BLEU is again obtained by combining the proposed WikiSplit dataset with the downsampled WebSplit, yielding a 32 point improvement over the prior best result.

---

[5] Past work on WebSplit (Narayan et al., 2017; Aharoni and Goldberg, 2018) reported macro-averaged *sentence-level* BLEU, calculated without smoothing precision values of zero. We found this ill-defined case occurred often for low-quality output.

[6] `github.com/OpenNMT/OpenNMT-py`, 0ecec8b

Training	Unsupported	Missing	Repeated	Correct
AG18	82	45	12	26/119 (22%)
WebSplit	58	47	13	32/100 (32%)
WikiSplit	8	5	0	91/100 (91%)
Both	**4**	**4**	**0**	**95/100 (95%)**

Table 6: Manual evaluation results, as counts over the simple sentences predicted by each model for a random sample of 50 inputs from WebSplit 1.0 validation set.

Example 1 from WebSplit validation set	
Input	*Alan Bean , born in Wheeler , Texas , was a crew member of Apollo 12 where Alfred Worden served as the backup pilot .*
AG18	Alan Bean is a member of the crew of Apollo 12 which was operated by backup pilot . Alan Shepard was born in Texas .
WebSplit	Alan Bean served as a crew member of Apollo 12 . Alan Bean was born in Wheeler .
WikiSplit	Alan Bean , born in Wheeler , Texas . He was a crew member of Apollo 12 where Alfred Worden served as the backup pilot .
Both	Alan Bean was born in Wheeler , Texas . Alan Bean was a crew member of Apollo 12 . Alfred Worden served as the backup pilot .

Example 2 from WikiSplit validation set	
Input	*He is a short , dark grayish - green monster with a beak - like mouth and a cap made out of the upper part of a creature 's skull .*
WebSplit	He is a food found in grayish . grayish monster is an ingredient in grayish .
WikiSplit	He is a short , dark grayish - green monster with a beak - like mouth . A cap made out of the upper part of a creature 's skull .
Both	He is a short , dark grayish - green monster with a beak - like mouth . His cap is made out of the upper part of a creature 's skull .

Example 3 from WikiSplit validation set	
Input	*Der beglückte Florindo is an opera composed by Handel in Hamburg in 1708 as part of a double opera , the other part being Die verwandelte Daphne .*
WebSplit	Handel opera is the publisher of the opera opera . Handel is the capital of 1708 .
WikiSplit	Der beglückte Florindo is an opera composed by Handel in Hamburg in 1708 . It was part of a double opera , the other part being Die verwandelte Daphne .
Both	Der beglückte Florindo is an opera composed by Handel in Hamburg in 1708 as part of a double opera . The other part being Die verwandelte Daphne .

Table 7: Example model predictions for items from each validation set. AG18 gives the output of the Copy512-model of Aharoni and Goldberg (2018), while the other outputs are from our models trained on the corresponding data.

4 Conclusion and Outlook

Our results demonstrate a large, positive impact on the split-and-rephrase task when training on large, diverse data that contains some noise. This suggests that future improvements may come from finding other such sources of data as much as from modeling. The new WikiSplit dataset is intended as training data, but for further progress on the split-and-rephrase task, we ideally need evaluation data also derived from naturally occurring sentences, and an evaluation metric that is more sensitive to the particularities of the task.

5 Acknowledgments

Thanks go to Kristina Toutanova and the anonymous reviewers for helpful feedback on an earlier draft, and to Roee Aharoni for supplying his system's outputs.

References

Roee Aharoni and Yoav Goldberg. 2018. Split and Rephrase: Better Evaluation and a Stronger Baseline. In *Proc. of ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proc. of EMNLP*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating Training Corpora for NLG Micro-Planners. In *Proc. of ACL*.

Dan Gillick. 2009. Sentence Boundary Detection and the Problem with the U.S. In *Proc. of NAACL*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proc. of ACL*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-Answer Driven Semantic Role Labeling: Using Natural Language to Annotate Natural Language. In *Proc. of EMNLP*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proc. of ACL, System Demonstrations*.

Philipp Koehn and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. In *Proc. of the First Workshop on Neural Machine Translation*.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. 2017. Split and Rephrase. In *Proc. of EMNLP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proc. of ACL*.

Matthew Shardlow. 2014. A Survey of Automated Text Simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.

Advaith Siddharthan. 2006. Syntactic Simplification and Text Cohesion. *Research on Language and Computation*, 4(1):77–109.

Advaith Siddharthan. 2014. A survey of research on text simplification. *International Journal of Applied Linguistics*, 165(2):259–298.

Sara Tonelli, Alessio Palmero Aprosio, and Francesca Saltori. 2016. SIMPITIKI: a Simplification corpus for Italian. In *Proc. of CLiC-it*.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A Dataset and Evaluation Metrics for Abstractive Compression of Sentences and Short Paragraphs. In *Proc. of EMNLP*.

Kristian Woodsend and Mirella Lapata. 2011. Learning to Simplify Sentences with Quasi-Synchronous Grammar and Integer Programming. In *Proc. of EMNLP*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Elif Yamangil and Rani Nelken. 2008. Mining Wikipedia Revision Histories for Improving Sentence Compression. In *Proc. of ACL*.

Diyi Yang, Aaron Halfaker, Robert Kraut, and Eduard Hovy. 2017. Identifying Semantic Edit Intentions from Revisions in Wikipedia. In *Proc. of EMNLP*.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from Wikipedia. In *Proc. of NAACL*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware Attention and Supervised Data Improve Slot Filling. In *Proc. of EMNLP*.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A Monolingual Tree-based Translation Model for Sentence Simplification. In *Proc. of Coling*.

BLEU is Not Suitable for the Evaluation of Text Simplification

Elior Sulem, Omri Abend, Ari Rappoport
Department of Computer Science, The Hebrew University of Jerusalem
`{eliors|oabend|arir}@cs.huji.ac.il`

Abstract

BLEU is widely considered to be an informative metric for text-to-text generation, including Text Simplification (TS). TS includes both lexical and structural aspects. In this paper we show that BLEU is not suitable for the evaluation of sentence splitting, the major structural simplification operation. We manually compiled a sentence splitting gold standard corpus containing multiple structural paraphrases, and performed a correlation analysis with human judgments.[1] We find low or no correlation between BLEU and the grammaticality and meaning preservation parameters where sentence splitting is involved. Moreover, BLEU often negatively correlates with simplicity, essentially penalizing simpler sentences.

1 Introduction

BLEU (Papineni et al., 2002) is an n-gram-based evaluation metric, widely used for Machine Translation (MT) evaluation. BLEU has also been applied to monolingual translation tasks, such as grammatical error correction (Park and Levy, 2011), summarization (Graham, 2015) and text simplification (Narayan and Gardent, 2014; Štajner et al., 2015; Xu et al., 2016), i.e. the rewriting of a sentence as one or more simpler sentences.

Along with the application of parallel corpora and MT techniques for TS (e.g., Zhu et al., 2010; Wubben et al., 2012; Narayan and Gardent, 2014), BLEU became the main automatic metric for TS, despite its deficiencies (see §2). Indeed, focusing on lexical simplification, Xu et al. (2016) argued that BLEU gives high scores to sentences that are close or even identical to the input, especially when multiple references are used. In their experiments, BLEU failed to predict simplicity, but obtained a higher correlation with grammaticality and meaning preservation, relative to the SARI metric they proposed.

In this paper, we further explore the applicability of BLEU for TS evaluation, examining BLEU's informativeness where sentence splitting is involved. Sentence splitting, namely the rewriting of a single sentence as multiple sentences while preserving its meaning, is the main structural simplification operation. It has been shown useful for MT preprocessing (Chandrasekar et al., 1996; Mishra et al., 2014; Li and Nenkova, 2015) and human comprehension (Mason and Kendall, 1979; Williams et al., 2003), independently from other lexical and structural simplification operations. Sentence splitting is performed by many TS systems (Zhu et al., 2010; Woodsend and Lapata, 2011; Siddharthan and Angrosh, 2014; Narayan and Gardent, 2014, 2016). For example, 63% and 80% of the test sentences are split by the systems of Woodsend and Lapata (2011) and Zhu et al. (2010), respectively (Narayan and Gardent, 2016). Sentence splitting is also the focus of the recently proposed Split-and Rephrase sub-task (Narayan et al., 2017; Aharoni and Goldberg, 2018), in which the automatic metric used is BLEU.

For exploring the effect of sentence splitting on BLEU scores, we compile a human-generated gold standard sentence splitting corpus – HSplit, which will also be useful for future studies of splitting in TS, and perform correlation analyses with human judgments. We consider two reference sets. First, we experiment with the most common set, proposed by Xu et al. (2016), evaluating a variety of system outputs, as well as HSplit. The references in this setting explicitly emphasize lexical operations, and do not contain splitting or content deletion.[2] Second, we experiment with HSplit as

---

[1] The corpus can be found in `https://github.com/eliorsulem/HSplit-corpus`

[2] Nevertheless, they are also used in contexts where struc-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 738–744
Brussels, Belgium, October 31 - November 4, 2018.

the reference set, evaluating systems that focus on sentence splitting. The first setting allows assessing whether BLEU with the standard reference set is a reliable metric on systems that perform splitting. The second allows assessing whether BLEU can be adapted to evaluate splitting, given a reference set so oriented.

We find that BLEU is often negatively correlated with simplicity, even when evaluating outputs without splitting, and that when evaluating outputs with splitting, it is less reliable than a simple measure of similarity to the source (§4.2). Moreover, we show that BLEU cannot be adapted to assess sentence splitting, even where the reference set focuses on this operation (§4.3). We conclude that BLEU is not informative and is often misleading for TS evaluation and for the related Split and Rephrase task.

2 Related Work

The BLEU Metric. BLEU (Papineni et al., 2002) is reference-based, where the use of multiple references is used to address cross-reference variation. To address changes in word order, BLEU uses n-gram precision, modified to eliminate repetitions across the references. A brevity term penalizes overly short sentences. Formally:

$$\mathrm{BLEU} = \mathrm{BP} \times \exp(\sum_{n=1}^{N} w_n \log(p_n))$$

where BP is the brevity penalty term, p_n are the modified precisions, and w_n are the corresponding weights, which are usually uniform in practice.

The experiments of Papineni et al. (2002) showed that BLEU correlates with human judgments in the ranking of five English-to-Chinese MT systems and that it can distinguish human and machine translations. Although BLEU is widely used in MT, several works have pointed out its shortcomings (e.g., Koehn and Monz, 2006). In particular, Callison-Burch et al. (2006) showed that BLEU may not correlate in some cases with human judgments since a huge number of potential translations have the same BLEU score, and that correlation decreases when translation quality is low. Some of the reported shortcomings are relevant to monolingual translation, such as the impossibility to capture synonyms and paraphrases that are not in the reference set, or the uniform weighting of words.

BLEU in TS. While BLEU is standardly used for TS evaluation (e.g., Xu et al., 2016; Nisioi et al., 2017; Zhang and Lapata, 2017; Ma and Sun, 2017), only few works tested its correlation with human judgments. Using 20 source sentences from the PWKP test corpus (Zhu et al., 2010) with 5 simplified sentences for each of them, Wubben et al. (2012) reported positive correlation of BLEU with simplicity ratings, but no correlation with adequacy. T-BLEU (Štajner et al., 2014), a variant of BLEU which uses lower n-grams when no overlapping 4-grams are found, was tested on outputs that applied only structural modifications to the source. It was found to have moderate positive correlation for meaning preservation, and positive but low correlation for grammaticality. Correlation with simplicity was not considered in this experiment. Xu et al. (2016) focused on lexical simplification, finding that BLEU obtains reasonable correlation for grammaticality and meaning preservation but fails to capture simplicity, even when multiple references are used. To our knowledge, no previous work has examined the behavior of BLEU on sentence splitting, which we investigate here using a manually compiled gold standard.

3 Gold-Standard Splitting Corpus

In order to investigate the effect of correctly splitting sentences on the automatic metric scores, we build a parallel corpus, where each sentence is modified by 4 annotators, according to specific sentence splitting guidelines. We use the complex side of the test corpus of Xu et al. (2016).[3]

While Narayan et al. (2017) recently proposed the semi-automatically compiled WEB-SPLIT dataset for training automatic sentence splitting systems, here we generate a completely manual corpus, without a-priori splitting points nor do we pre-suppose that all sentences should be split. This corpus enriches the set of references focused on lexical operations that were collected by Xu et al. (2016) for the same source sentences and can also be used as an out-of-domain test set for Split-and-Rephrase (Narayan et al., 2017).

We use two sets of guidelines. In Set 1, annotators are required to split the original as much as possible, while preserving the sentence's gram-

---

tural operations are involved (Nisioi et al., 2017; Sulem et al., 2018b).

[3] `https://github.com/cocoxu/simplification` includes the corpus, the SARI metric and the SBMT-SARI system. The corpus comprises 359 sentences.

maticality, fluency and meaning. The guidelines include two sentence splitting examples.[4] In Set 2, annotators are encouraged to split only in cases where it simplifies the original sentence. That is, simplicity is implicit in Set 1 and explicit in Set 2. In both sets, the annotators are instructed to leave the source unchanged if splitting violates grammaticality, fluency or meaning preservation.[5]

Each set of guidelines is used by two annotators, with native or native-like proficiency in English. The obtained corpora are denoted by HSplit1, HSplit2 (for Set 1), and HSplit3 and HSplit4 (for Set 2), each containing 359 sentences. Table 1 presents statistics for the corpora. Both in terms of the number of splits per sentence (# Sents) and in terms of the proportion of input sentences that have been split (SplitSents), we observe that the average difference within each set is significantly greater than the average difference between the sets.[6] This suggests that the number of splits is less affected by the explicit mention of simplicity than by the inter-annotator variability.

	# Sents	SplitSents (%)
HSplit1	1.93	68
HSplit2	2.28	86
HSplit3	1.87	63
HSplit4	1.99	71
HSplitAverage	2.02	72

Table 1: Statistics for the sentence splitting benchmark. #Sents denotes the average number of sentences in the output. SplitSents denotes the proportion of input sentences that have been split. The last row presents the average scores of the 4 HSplit corpora.

4 Experiments

4.1 Experimental Setup

Metrics. In addition to BLEU,[7] we also experiment with (1) iBLEU (Sun and Zhou, 2012) which was recently used for TS (Xu et al., 2016; Zhang et al., 2017) and which takes into account the BLEU scores of the output against the input and against the references; (2) the Flesch-Kincaid Grade Level (FK; Kincaid et al., 1975), computed at the system level, which estimates the readability of the text with a lower value indicating higher readability;[8] (3) SARI (Xu et al., 2016), which compares the n-grams of the system output with those of the input and the human references, separately evaluating the quality of words that are added, deleted and kept by the systems. For completeness, we also experiment with the negative Levenshtein distance to the source (-LD$_{SC}$), which serves as a measure of conservatism.[9]

We explore two settings. In one ("Standard Reference Setting", §4.2), we use two sets of references: the Simple Wikipedia reference (yielding BLEU-1ref and iBLEU-1ref), and 8 references obtained by crowdsourcing by Xu et al. (2016) (yielding BLEU-8ref, iBLEU-8ref and SARI-8ref). In the other ("HSplit as Reference Setting", §4.3), we use HSplit as the reference set.

Systems. For "Standard Reference Setting", we consider both a case where evaluated systems do not perform any splittings on the test set ("Systems/Corpora without Splits"), and one where we evaluate these systems, along with the HSplit corpus, used in the role of system outputs ("All Systems/Corpora"). Systems include six MT-based simplification systems, including outputs of the state-of-the-art neural TS system of Nisioi et al. (2017), in four variants: either default settings or initialization by word2vec, for each both the highest and the fourth ranked hypotheses in the beam are considered.[10] We further include Moses (Koehn et al., 2007) and SBMT-SARI (Xu et al., 2016), a syntax-based MT system tuned against SARI, and the identity function (outputs are same as inputs). The case which evaluates outputs with sentence splitting additionally includes the four HSplit corpora and the HSplit average scores.

For "HSplit as Reference Setting", we consider the outputs of six simplification systems whose main simplification operation is sentence splitting: DSS, DSS^m, SEMoses, SEMoses^m, SEMoses$_{LM}$ and SEMoses$^m_{LM}$, taken from (Sulem et al., 2018b).

Human Evaluation. We use the evaluation benchmark provided by Sulem et al. (2018b),[11] including system outputs and human evaluation scores corresponding to the first 70 sentences of

---

[4] Examples are taken from Siddharthan (2006).

[5] Examples are not provided in the case of Set 2 so as not to give an a-priori notion of simplicity. The complete guidelines are found in the supplementary material.

[6] Wilicoxon's signed rank test, $p = 1.6 \cdot 10^{-5}$ for #Sents and $p = 0.002$ for SplitSents.

[7] System-level BLEU scores are computed using the multi-bleu Moses support tool. Sentence-level BLEU scores are computed using NLTK (Loper and Bird, 2002).

[8] We thus computed the correlation in §4.2 for -FK.

[9] LD$_{SC}$ is computed using NLTK.

[10] Taking the fourth hypothesis rather than the first has been found to yield considerably less conservative TS systems.

[11] `https://github.com/eliorsulem/simplification-acl2018`

	Systems/Corpora without Splits				All Systems/Corpora			
	G	**M**	**S**	**StS**	**G**	**M**	**S**	**StS**
BLEU-1ref	0.43 (0.2)	1.00 (0)	-0.81 (0.01)	-0.43 (0.2)	0.11 (0.4)	0.08 (0.4)	-0.60 (0.02)	-0.67 (0.008)
BLEU-8ref	0.61 (0.07)	0.89 (0.003)	-0.59 (0.08)	-0.11 (0.4)	0.26 (0.2)	0.13 (0.3)	-0.42 (0.08)	-0.50 (0.05)
iBLEU-1ref	0.21 (0.3)	0.93 (0.001)	-0.85 (0.008)	-0.61 (0.07)	0.02 (0.5)	0.07 (0.4)	-0.61 (0.02)	-0.71(0.004)
iBLEU-8ref	0.61 (0.07)	0.89 (0.003)	-0.59 (0.08)	-0.11 (0.4)	0.26 (0.2)	0.13 (0.3)	-0.42 (0.08)	-0.50 (0.05)
-FK	-0.21 (0.3)	-0.57 (0.09)	0.67 (0.05)	0.39 (0.2)	-0.05 (0.4)	-0.03 (0.5)	0.51 (0.05)	0.64 (0.01)
SARI-8ref	-0.64 (0.06)	-0.86 (0.007)	0.52 (0.1)	0.00 (0.5)	-0.64 (0.01)	-0.72 (0.004)	0.26 (0.2)	-0.02 (0.5)
-LD$_{SC}$	0.29 (0.3)	0.86 (0.007)	-0.88 (0.004)	-0.57 (0.09)	0.21 (0.3)	0.51 (0.04)	-0.68 (0.007)	-0.52 (0.04)

Table 2: Spearman correlation (and p-values) at the system level between the rankings of automatic metrics and of human judgments for "Standard Reference Setting". Automatic metrics (rows) include BLEU and iBLEU (each used either with a single reference or with 8 references), the negative Flesh-Kincaid Grade Level (-FK), and SARI, computed with 8 references. We also include the negative Levenshtein distance between the output and the source (-LD$_{SC}$). Human judgments are of the Grammaticality (G), Meaning Preservation (M), Simplicity (S) and Structural Simplicity (StS) of the output. The left-hand side reports correlations where only simplifications that do not include sentence splitting are considered. The right-hand side reports correlations where the HSplit corpora are evaluated as well (see text). BLEU negatively correlates with S and StS in both cases, and shows little to no correlation with G and M where sentence splitting is involved.

the test corpus of Xu et al. (2016), and extend it to apply to HSplit as well.

The evaluation of HSplit is carried out by 3 in-house native English annotators, who rated the different input-output pairs for the different systems according to 4 parameters: Grammaticality (G), Meaning preservation (M), Simplicity (S) and Structural Simplicity (StS). G and M are measured using a 1 to 5 scale. A -2 to +2 scale is used for measuring simplicity and structural simplicity. For computing the inter-annotator agreement of the whole benchmark (including the system outputs and the HSplit corpora), we follow Pavlick and Tetreault (2016) and randomly select, for each sentence, one annotator's rating to be the rating of Annotator 1 and the rounded average rating of the two other annotators to be the rating of Annotator 2. We then compute weighted quadratic κ (Cohen, 1968) between Annotator 1 and 2. Repeating this process 1000 times, the obtained medians and 95% confidence intervals are 0.42 ± 0.002 for G, 0.77 ± 0.001 for M and 0.59 ± 0.002 for S and StS.

4.2 Results with Standard Reference Setting

Description of the Human Evaluation Scores. The human evaluation scores for each parameter are obtained by averaging over the 3 annotators. The scores at the system level are obtained by averaging over the 70 sentences. In the "All systems/corpora" case of the "Standard Reference Setting", where 12 systems/corpora are considered, the range of the average G scores at the system level is from 3.71 to 4.80 (σ = 0.29). For M, this max-min difference between the systems is 1.23 (σ=0.40). For S and StS, the differences are 0.53 (σ = 0.17) and 0.65 (σ = 0.20). At the sentence level, considering 840 sentences (70 for each of the system/corpora), the G and M scores vary from 1 to 5 (σ equals 0.69 and 0.85 respectively), and the S and StS scores from -1 to 2 (σ equals 0.53 and 0.50).

In the "Systems/corpora without Splits" case of the "Standard Reference Setting", where 7 systems/corpora are considered, the max-min difference at the system level are again 1.09 (σ = 0.36) and 1.23 (σ = 0.47) for G and M respectively. For S and StS, the differences are 0.45 and 0.49 (σ = 0.18). At the sentence level, considering 490 sentences (70 for each of the system/corpora), the G and M scores vary from 1 to 5 (σ equals 0.78 and 1.01 respectively), and the S and StS scores from -1 to 2 (σ equals 0.51 and 0.46).

Comparing HSplit to Identity. Comparing the BLEU score on the input (the identity function) and on the HSplit corpora, we observe that the former yields much higher BLEU scores. Indeed, BLEU-1ref obtains 59.85 for the input and 43.90 for the HSplit corpora (averaged over the 4 HSplit corpora). BLEU-8ref obtains 94.63 for the input and 73.03 for HSplit.[12] The high scores obtained for Identity, also observed by Xu et al. (2016), indicate that BLEU is a not a good predictor for relative simplicity to the input. The drop in the BLEU scores for HSplit is not reflected by the human evaluation scores for grammaticality (4.43 for AvgHSplit vs. 4.80 for Identity) and meaning preservation (4.70 vs. 5.00), where the decrease between Identity and HSplit is much more limited. For examining these tendencies in more detail, we compute the correlations between the au-

---

[12] These scores concern the first 70 sentences of the corpus. A similar phenomenon is observed on the whole corpus (359 sentences). BLEU-1ref obtains 59.23 for the input and 45.68 for HSplit. BLEU-8ref obtains 94.93 for the input and 75.68 for HSplit.

tomatic metrics and the human evaluation scores. They are described in the following paragraph.

Correlation with Human Evaluation. The system-level Spearman correlations between the rankings of the automatic metrics and the human judgments (see §4.1) are presented in Table 2. We find that in all cases BLEU and iBLEU negatively correlate with S and StS, indicating that they fail to capture simplicity and structural simplicity. Where gold standard splits are evaluated as well, BLEU's and iBLEU's failure to capture StS is even more pronounced. Moreover, BLEU's correlation with G and M in this case disappears. In fact, BLEU's correlation with M in this case is considerably lower than that of -LD$_{SC}$ and its correlation with G is comparable, suggesting BLEU is inadequate even as a measure of G and M if splitting is involved.

We examine the possibility that BLEU mostly acts as a measure of conservatism, and compute the Spearman correlation between -LD$_{SC}$ and BLEU. The high correlations we obtain between the metrics indicate that this may be the case. Specifically, BLEU-1ref obtains correlations of 0.86 ($p = 7 \times 10^{-3}$) without splits and of 0.52 ($p = 0.04$) where splitting is involved. BLEU-8ref obtains 0.82 ($p = 0.01$) and 0.55 ($p = 0.03$).

SARI obtains positive correlations with S, of 0.52 (without splits) and 0.26 (all systems/corpora), but correlates with StS in neither setting. This may stem from SARI's focus on lexical, rather than structural TS.

Similar trends are observed in the sentence-level correlation for S, StS and M, whereas G sometimes benefits in the sentence level from including HSplit in the evaluation. For G and M, the correlation with BLEU is lower than its correlation with -LD$_{SC}$ in both cases.

	G	**M**	**S**	**StS**
BLEU	0.36^*	0.43^*	$0.17\ (3 \cdot 10^{-4})$	$0.17\ (3 \cdot 10^{-4})$
iBLEU	0.32^*	0.40^*	$0.15\ (8 \cdot 10^{-4})$	$0.15\ (8 \cdot 10^{-4})$
SARI	-0.05 (0.2)	-0.11 (0.02)	$0.18\ (10^{-4})$	$0.19\ (6 \cdot 10^{-5})$
-LD$_{SC}$	0.65^*	0.66^*	0.21^*	$0.20\ (10^{-5})$

Table 3: Sentence-level Spearman correlation (and p-values) between the automatic metrics and the human ratings for "HSplit as Reference Setting". $^*p < 10^{-5}$.

4.3 Results with HSplit as Reference Setting

We turn to examining whether BLEU may be adapted to address sentence splitting, if provided with references that include splittings.

Description of the Human Evaluation Scores. In the "HSplit as Reference Reference Setting", where 6 systems are considered, the max-min difference at the system level is 0.16 (σ = 0.06) for G, 0.37 for M (σ = 0.15), and 0.41 for S and StS (σ equals 0.20 and 0.19 respectively). At the sentence level, considering 420 sentences (70 for each of the systems), the G and M scores vary from 1 to 5 (σ equals 0.99 and 0.88 respectively), and the S and StS scores from -2 to 2 (σ equals 0.63).

Correlation with Human Evaluation. On the system-level Spearman correlation between BLEU and human judgments, we find that while correlation with G is high (0.57, $p = 0.1$), it is low for M (0.11, $p = 0.4$), and negative for S (-0.70, $p = 0.06$) and StS (-0.60, $p = 0.1$). Sentence-level correlations of BLEU and iBLEU are positive, but they are lower than those obtained by LD$_{SC}$. See Table 3.

To recap, results in this section demonstrate that even when evaluated against references that focus on sentence splitting, BLEU fails to capture the simplicity and structural simplicity of the output.

5 Conclusion

In this paper we argued that BLEU is not suitable for TS evaluation, showing that (1) BLEU negatively correlates with simplicity, and that (2) even as a measure of grammaticality or meaning preservation it is comparable to, or worse than -LD$_{SC}$, which requires no references. Our findings suggest that BLEU should not be used for the evaluation of TS in general and sentence splitting in particular, and motivate the development of alternative methods for structural TS evaluation, such as (Sulem et al., 2018a).

Acknowledgments

We would like to thank the annotators for participating in our generation and evaluation experiments. We also thank the anonymous reviewers for their helpful advices. This work was partially supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI) and by the Israel Science Foundation (grant No. 929/17), as well as by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

References

Roee Aharoni and Yoav Goldberg. 2018. Split and rephrase: Better evaluation and a stronger baseline. In *Proc. of ACL'18, Short papers*, pages 719–724. http://aclweb.org/anthology/P18-2114.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation. In *Proc. of EACL'06*, pages 249–256. http://www.aclweb.org/anthology/E06-1032.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for sentence simplification. In *Proc. of COLING'96*, pages 1041–1044. http://aclweb.org/anthology/C/C96/C96-2183.pdf.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proc. of EMNLP'15*, pages 128–137. http://www.aclweb.org/anthology/D15-1013.

J. Peter Kincaid, Robert P. Fishburne Jr., Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and Flesch reading ease formula) for Navy enlisted personnel. Technical report, Defense Technical Information Center (DTIC) Document.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Buch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proc. of ACL'07 on interactive poster and demonstration sessions*, pages 177–180. http://aclweb.org/anthology/P/P07/P07-2045.pdf.

Philipp Koehn and Christof Monz. 2006. Manual and automatic evaluation of machine translation between European languages. In *Proc. of the Workshop on Statistical Machine Translation*. http://www.aclweb.org/anthology/W06-3114.

Junyi Jessi Li and Ani Nenkova. 2015. Detecting content-heavy sentences: A cross-language case study. In *Proc. of EMNLP'15*, pages 1271–1281. http://www.aclweb.org/anthology/D15-1148.

Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. In *Proc. of EMNLP'02*, pages 63–70. http://www.aclweb.org/anthology/W/W02/W02-0109.pdf.

Shuming Ma and Xu Sun. 2017. A semantic relevance based neural network for text summarization and text simplification. ArXiv:1710.02318 [cs.CL]. https://arxiv.org/pdf/1710.02318.pdf.

Jana M. Mason and Janet R. Kendall. 1979. Facilitating reading comprehension through text structure manipulation. *Alberta Journal of Medical Psychology*, 24:68–76.

Kshitij Mishra, Ankush Soni, Rahul Sharma, and Dipti Misra Sharma. 2014. Exploring the effects of sentence simplification on Hindi to English Machine Translation systems. In *Proc. of the Workshop on Automatic Text Simplification: Methods and Applications in the Multilingual Society*, pages 21–29. http://www.aclweb.org/anthology/W14-5603.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proc. of ACL14*, pages 435–445. http://aclweb.org/anthology/P/P14/P14-1041.pdf.

Shashi Narayan and Claire Gardent. 2016. Unsupervised sentence simplification using deep semantics. In *Proc. of INLG'16*, pages 111–120. http://aclweb.org/anthology/W/W16/W16-6620.pdf.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *Proc. of EMNLP'17*, pages 617–627. http://aclweb.org/anthology/D17-1064.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proc. of ACL'17 (Short paper)*, pages 85–91. http://www.aclweb.org/anthology/P17-2014.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL'02*, pages 311–318. http://aclweb.org/anthology/P/P02/P02-1040.pdf.

Y. Albert Park and Roger Levy. 2011. Automated whole sentence grammar correction using a noisy channel model. In *Proc.of ACL-HLT'11*, pages 934–944. http://aclweb.org/anthology/P/P11/P11-1094.pdf.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *TACL*, 4:61–74. http://www.aclweb.org/anthology/Q16-1005.

Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4:77–109.

Advaith Siddharthan and M. A. Angrosh. 2014. Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules. In *Proc. of EACL'14*, pages 722–731. http://aclweb.org/anthology/E/E14/E14-1076.pdf.

Sanja Štajner, Hannah Bechara, and Horacio Saggion. 2015. A deeper exploration of the standard PB-SMT approach to text simplification and its evaluation. In *Proc. of ACL'15, Short papers*, pages 823–828. http://aclweb.org/anthology/P/P15/P15-2135.pdf.

Sanja Štajner, Ruslan Mitkov, and Horacio Saggion. 2014. One step closer to automatic evaluation of text simplification systems. *Proc. of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 1–10. http://www.aclweb.org/anthology/W14-1201.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018a. Semantic structural evaluation for text simplification. In *Proc. of NAACL'18*, pages 685–696. http://aclweb.org/anthology/N18-1063.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018b. Simple and effective text simplification using semantic and neural methods. In *Proc. of ACL'18*, pages 162–173. http://aclweb.org/anthology/P18-1016.

Hong Sun and Ming Zhou. 2012. Joint learning of a dual SMT system for paraphrase generation. In *Proc. of ACL'12*, pages 38–42. http://aclweb.org/anthology/P/P12/P12-2008.pdf.

Sandra Williams, Ehud Reiter, and Liesl Osman. 2003. Experiments with discourse-level choices and readability. In *Proc. of the European Natural Language Workshop (ENLG)*.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proc. of EMNLP'11*, pages 409–420. http://aclweb.org/anthology/D/D11/D11-1038.pdf.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proc. of ACL'12*, pages 1015–1024. http://aclweb.org/anthology/P/P12/P12-1107.pdf.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *TACL*, 4:401–415. http://aclweb.org/anthology/Q/Q16/Q16-1029.pdf.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proc. of EMNLP'17*, pages 595–605. http://aclweb.org/anthology/D17-1062.

Yaoyuan Zhang, Zhenxu Ye, Dongyan Zhao, and Rui Yan. 2017. A constrained sequence-to-sequence neural model for sentence simplification. ArXiv:1704.02312 [cs.CL]. https://arxiv.org/pdf/1704.02312.pdf.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proc. of COLING'10*, pages 1353–1361. http://aclweb.org/anthology/C/C10/C10-1152.pdf.

S2SPMN:A Simple and Effective Framework for Response Generation with Relevant Information

Jiaxin Pei
School of Computer Science
Wuhan University
Hubei, China
pedropei@vip.qq.com

Chenliang Li*
School of Cyber Science and Engineering
Wuhan University
Hubei, China
cllee@whu.edu.cn

Abstract

How to generate relevant and informative responses is one of the core topics in response generation area. Following the task formulation of machine translation, previous works mainly consider response generation task as a mapping from a source sentence to a target sentence. To realize this mapping, existing works tend to design intuitive but complex models. However, the relevant information existed in large dialogue corpus is mainly overlooked. In this paper, we propose Sequence to Sequence with Prototype Memory Network (S2SPMN) to exploit the relevant information provided by the large dialogue corpus to enhance response generation. Specifically, we devise two simple approaches in S2SPMN to select the relevant information (named prototypes) from the dialogue corpus. These prototypes are then saved into prototype memory network (PMN). Furthermore, a hierarchical attention mechanism is devised to extract the semantic information from the PMN to assist the response generation process. Empirical studies indicate the advantage of our model over several classical and strong baselines.

1 Introduction

Dialogue systems, or say, chatbots are usually considered as the future of human-computer interaction and extensive works have been done in this area (Wen et al., 2016; Qiu et al., 2017; Wen et al., 2017; Kreyssig et al., 2018).

As one of the main approaches for dialogue system design, response generation has attracted more and more attention from research community. Neural networks based models like Seq2Seq architecture (Vinyals and Le, 2015; Shang et al., 2015) are proven to be effective to generate valid responses for a dialogue system. However, as revealed in many previous works (Li et al., 2016a; Wu et al., 2018), ”safe reply” is still an open problem and lots of efforts are made to generate more informative responses (Li et al., 2016a; Mou et al., 2016; Li et al., 2016b; Qiu et al., 2017; Li et al., 2017; Zhao et al.; He et al., 2017; Zhou et al., 2017; Liu et al., 2018; Chen et al., 2018).

Note that in this paper when we say response generation, we focus on single turn chit-chat for that other tasks like multi-turn (Zhang et al., 2018) or goal-oriented (Kan et al., 2018)generation could be partly considered as the extensions of single-turn generation.

Though existing works mentioned above are helpful in some ways, they all follow the task formulation proposed by (Ritter et al., 2011), which considers response generation (RG) task as a mapping from a source sentence to a target sentence like machine translation (MT). This task formulation ignores the natural difference between MT and RG: MT deals with sentence pairs of the same meanings while RG needs to realize the meaning transformation from a source post to the target response. In this sense, the meaning transformation is more difficult than machine translation. Hence, many researchers have designed more and more complex models. However, given a target post, the relevant information covered by the dialogue corpus is usually overlooked. It is intuitive that the responses for a similar post would provide more contextual information to guide the response generation. To this end, we are interested in exploiting the relevant responses in the training set as soft prototypes to assist the response generation.

Specifically, in this paper, we propose Sequence to Sequence with Prototype Memory Network (named S2SPMN). We introduce two Prototype Memory Networks (PMNs) to store the relevant responses extracted from the dialogue corpus: static PMN and dynamic PMN. Tested on a widely used benchmark dataset, the proposed

---

*Chenliang Li is the Corresponding Author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 745–750
Brussels, Belgium, October 31 - November 4, 2018.

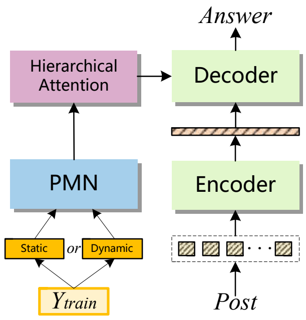

Figure 1: S2SPMN Framework

S2SPMN produces more informative responses than the standard and strong baselines. To the best of our knowledge, it is the first work leveraging prototype information in dialogue corpus in response generation area.

The contributions of this paper could be summarized as follows:

(1) We propose S2SPMN, a simple yet effective response generation model which could leverage relevant information in dialogue corpus to assist response generation.

(2) Empirical studies indicate the superiority of proposed S2SPMN over other methods.

2 Architecture

2.1 Problem Definition

Given a dialogue dataset $\Gamma = \{X_i, Y_i\}_{i=1}^{N}$, where Y_i is the response for a post X_i, we aim to train a model with Γ such that the model can generate an accurate and informative response for a new post X'. Here, we propose to exploit the relevant information provided by Γ. Let $T' = (r_1, r_2, ..., r_m)$ refers to the prototype memory network constructed for post X', where r_i is the i-th relevant response (named prototype) extracted from dialogue dataset Γ. The goal is to derive the model to generate the response Y': $p(Y'|X') = p(Y'|T', X')$.

In following sections, we firstly introduce the generation framework with hierarchical attention mechanism assuming PMN is constructed. Then we will introduce two kinds of PMNs: static PMN and dynamic PMN.

2.2 Sequence-to-Sequence with Prototype Memory Network

S2SPMN is built with a Seq2Seq encoder-decoder framework (Sutskever et al., 2014) with the attention mechanism (Bahdanau et al., 2014). We use LSTM (Hochreiter and Schmidhuber, 1997) to materialize both encoder and decoder. The hidden state at t-th encoding step is generated from previous hidden state h_{t-1} and current input x_t as follows:

$$h_t = lstm(x_t, h_{t-1}) \tag{1}$$

For decoder, at i-th timestep, s_i is the decoder's hidden state and p_i is the probability distribution of candidate words .

$$s_i = lstm(y_{i-1}, s_{i-1}, c_i, o_i) \tag{2}$$

$$p_i = softmax(MLP(s_i, y_{t-1}, o_i, c_i)) \tag{3}$$

where $MLP()$ is a one-layer perception, o_i is the hierarchical attention over entire prototype memory network which will be formalized in following sections. c_i is the summarization for the post regarding to the hidden state s_{i-1}:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j, \quad \alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})} \tag{4}$$

$$e_{ij} = v_1^T MLP(s_{i-1}, h_j) \tag{5}$$

where v_1 is the attention parameter.

2.3 Prototype Memory Network

Given a post X', a set of responses are selected from training set as prototypes and are then saved into the Prototype Memory Network(PMN). We propose two kinds of Prototype Memory Networks.

Static PMN: For static PMN(SPMN), we randomly select m responses before training starts and the entire PMN remains unchanged during the training process. That is, we use the same prototypes for all the post-response pairs.

Dynamic PMN: In dynamic PMN(DPMN), prototypes are selected by retrieving the most relevant posts. We calculate the cosine similarity with TF-IDF weighting scheme between the given post and all the posts in training set. We consider top-m posts and put the associated responses into DPMN. This means that the prototypes are characteristic for each post-response pair.

In both SPMN and DPMN, m is a predefined hyper-parameter controlling the size of the PMN. Each prototype is represented with the concatenation of word embeddings. We perform zero padding for both SPMN and DPMN with a pseudo word[1], making the length for the representation of each prototype be the same. Here we denote the prototype memory network as PMN = $\{r_1, r_2, ..., r_m\}$, in which r_m is the representation of m-th prototype and m is the size of the PMN. And $r_m = \{w_{m,1}, w_{m,2}, ..., w_{m,l}\}$ where $w_{m,i}$ is the embedding of i-th word, and l is the maximum allowable length for a prototype.

For both SPMN and DPMN, we select responses rather than posts although sometimes they have similar vocabularies and syntactic structure. We believe that using responses as prototypes could help with the meaning transformation from post to response. In DPMN, all the retrieved prototypes could be considered as responses to the target post. It is intuitive that the generated response would have similar representation to these prototypes.

2.4 Hierarchical Attention Mechanism

We use a two-stage hierarchical attention mechanism to extract useful information in PMN and integrate it into the decoding process. The first stage is a sentence level attention over entire PMN to generate the abstractive prototype $\hat{r}_i$ at each timestep:

$$\hat{r}_i = \sum_{j=1}^{M} \beta_{ij} r_j, \quad \beta_{ij} = \frac{exp(f_{ij})}{\sum_{k=1}^{M} exp(f_{ik})} \quad (6)$$

$$f_{ij} = v_2^T MLP(s_{i-1}, r_j) \quad (7)$$

where v_2 is the attention parameter.

The second stage is a word level attention o_i over the generated $\hat{r}_i = \{\hat{w}_1, \hat{w}_2, ..., \hat{w}_l\}$ and is calculated as follows:

$$o_i = \sum_{j=1}^{l} \gamma_{ij} \hat{w}_j, \quad \gamma_{ij} = \frac{exp(g_{ij})}{\sum_{k=1}^{l} exp(g_{ik})} \quad (8)$$

$$g_{ij} = v_3^T MLP(s_{i-1}, \hat{w}_j) \quad (9)$$

where v_3 is the attention parameter.

3 Experiment

3.1 Experiment Setup

We use a subset of STC dataset (Shang et al., 2015) crawled from Weibo, the largest social media in China. The vocabulary size is set to be $8,000$ for computational efficiency and words out of vocabulary are replaced by the symbol ”unk”. We remove sentences longer than 25 words or containing more than 2 unk symbols. After preprocessing step, we have $315,980$ post-response pairs in training set, $3,510$ pairs in validation set and 300 in test set.

In our model, we use one-layer LSTM and the hidden size is set to be 600 in both encoder and decoder. For all the words used in our model, the embedding size is 300. Mini-batch learning is used and batch size is set as 64. We use simple SGD for optimization and the initial learning rate is set to be 0.2.

3.2 Evaluation Metrics

We use two automatic evaluation metrics including Perplexity and Distinct. Human evaluation is also conducted as the only gold standard for response generation is human judgement.

Perplexity: Following (Vinyals and Le, 2015) and (Xing et al., 2017), we use perplexity as one of our automatic evaluation metrics. Perplexity could measure the holistic condition of model learning. A lower perplexity score indicates better generalization performance. Perplexity on both validation set (PPL-V) and test set (PPL-T) are presented in table 2.

Distinct-1, Distinct-2: Distinct-1 and distinct-2 calculate the ratios of distinct unigrams and bigrams in the generated responses respectively (Li et al., 2016a; Xing et al., 2017; Wu et al., 2018). The higher score suggests that the generated response is more diverse and informative. Here, we report the distinct-1 and distinct-2 scores on entire test set.

Human Annatation: We further recruit human annotators to judge the quality of the generated answers for all the qa-pairs in test set. Responses generated by all the methods are pooled and randomly shuffled for each annotator. A score between 0 and 2 is assigned to each generated answer based on the following criteria:

+2: the answer is natural and relevant to the question.

+1: the answer can be used as a reply, but is not informative enough (e.g. “我也是” (me too), “不知道” (I don’t know)).

+0: the answer is irrelevant and unclear in meaning (e.g. too many grammatical errors to understand).

---

[1] The embedding of the pseudo word is a zero vector.

Model	PPL-V	PPL-T	distinct-1	distinct-2
S2SA	8.41	9.05	**0.0809**	**0.2110**
TAS2S	7.38	7.84	0.04759	0.1087
SPMN500	7.04	7.93	0.06430	0.1734
SPMN1000	**6.28**	7.72	0.07347	0.1909
DPMN100	6.45	**7.69**	0.04350	0.1048

Table 1: Automatic evaluation

Model	0	1	2	Kappa
S2SA	**76.83**%	16.33%	6.83%	0.6124
TAS2S	69.83%	19.83%	10.33%	0.7425
SPMN500	21.67%	55.00%	23.33%	0.6534
SPMN1000	19.17%	52.50%	28.33%	0.7330
DPMN100	12.08%	**56.67**%	**31.25**%	0.6280

Table 2: Human Annotation

3.3 Results Comparation

We use a standard baseline and a strong baseline for comparison.

S2SA: The standard Seq2Seq model with an attention mechanism (Vinyals and Le, 2015).

TAS2S: One of the existing state-of-the-art neural models based on Seq2Seq architecture. The topical words relevant to the post are considered via an attention mechanism when decoding (Xing et al., 2017).

As for our models, we use SPMN to denote the generating method with static prototype memory networks and DPMN with dynamic prototype memory networks. The numbers following model names are the size of PMN.

Automatic Evaluation: Table 1 shows the automatic evaluation results. We see that both SPMN and DPMN obtain huge improvements over the two baselines in terms of PPL-V and PPL-T. Also, we observe that SPMN1000 outperforms SPMN500 in all the four automatic metrics. Note that each post has the same prototypes provided by SPMN. This is reasonable that the relevant response is more likely to be covered by storing more prototypes in SPMN. As for the DPMN, we can see that DPMN achieves the best performance with only 100 prototypes in terms of PPL-T, compared with the other 4 methods. This suggests that using a retrieval mechanism to incorporate the relevant responses brings more useful information for better response generation. Note that S2SA outperforms the others in terms of distinct-1 and distinct-2. Further human evaluation indicates that many responses generated by S2SA are irrelevant and meaningless, which could inevitably increase the distinct scores.

Human Annotation: Table 2 shows human annotation results. It is clear that our models (SPMN500, SPMN1000, DPMN100) generate much more informative and valid responses and much less meaningless or "safe" responses than baseline models (S2SA, TAS2S). Specifically, SPMN500, SPMN1000 and DPMN100 all outperform S2SA and TAS2S by producing more informative and valid responses. Also, we can find that DPMN still outperforms SPMN500 and SPMN1000 with only 100 relevant responses, which is consistent with the observation made in automatic evaluation (in terms of PPL-V and PPL-T).

Case Study Table 3 shows several cases generated by different models. Note that the size of training set and vocabulary used in our experiments are relatively small compared to millions of qa-pairs used in other works (Xing et al., 2017; Wu et al., 2018), so it's reasonable that bad cases sometimes occur in results of baselines. However, our models, no matter the static one or the dynamic one, could generate amazing responses which are not only grammatical and informative, but also have some emotional expressions like the use of punctuation and repetition.

4 Related Work

4.1 Natural language generation

How to generate grammatical and interesting sentences in different situations is one of the core topics in natural language processing area. Extensive works are proposed to generate poems (Zhang et al., 2017), abstracts (Wang and Ling, 2016), arguments (Hua and Wang, 2018), stories (Peng et al., 2018) and so on. Although existing approaches are useful in some ways, it's still difficult to generate natural sentences from scratch and integrating retrieved results has recently become a new fashion in this area. Hua and Wang (2018) proposed an encoder-decoder style neural network-based argument generation model enriched with externally retrieved evidence from Wikipedia. Li et al. (2018) devised a Retrieve-Rerank-Rewrite model for abstractive summarization which uses retrieved results as soft template to assist the decoding process.

Post 1	**美国加州莫诺湖，奇异的美丽 (Mono Lake in California,the US, fantastically beautiful)**
S2SA	有unk的地方,就是unk (It’s unk if there’s unk)
TAS2S	，，， (speechless)
SPMN500	这是什么地方？我也去看看 (Where’s the place? I’d like to go and see)
SPMN1000	这是在哪？！ (What’s the place?!)
DPMN100	我也想去的地方 (That’s exactly where I want to go)

Post 2	**看看能让妈妈疯掉的baby (Oh! Look at the baby! She’s driving mom mad!)**
S2SA	unk,unk! (speechless)
TAS2S	我也喜欢 (Wow,I like her,too)
SPMN500	好可爱啊！好可爱！ (She’s so cute!)
SPMN1000	我也是这样的 (I was like her when I was at her age)
DPMN100	我也想养一个 (I want a baby like her)

Post 3	**装修以后很快后悔的80件事！还没装修的朋友们，一定要借鉴！ (80 things to regret after decorating your house! Look at this article if you haven’t started decoration!)**
S2SA	有unk的时候，我会有一个unk的unk! (When I have unk, I will have a unk unk!)
TAS2S	，的 (speechless)
SPMN500	我也要去看看，一定要收藏！ (Ok,I will read and collect it!)
SPMN1000	很实用，很实用，很实用 (very very very useful)
DPMN100	很实用，很实用，很实用，很实用！ (very very very very useful!)

Table 3: The answers generated by different models for the sampled questions.

4.2 Response generation

Hand-craft rules, retrieval and generation are three main solutions for conversational AI and generation is the most interesting one in current research community. Li et al. (2016a; 2016b; 2017) proposed a series of works in solving the ”safe reply” problem using different approaches like redefining the objective function or leveraging GAN. Xing et al. (2017) considered topic coherence issue by incorporating topical words. Dynamically restricting the target vocabulary is also an interesting idea and Wu et al. (2018) proposed to filter irrelevant words while achieving better computational efficiency . He et al. (2017) introduced copy mechanism to simulate people’s behaviors in real conversations and the proposed model could copy useful words from source sentences. Zhou et al. (2017) indicated that emotion is quite important in real dialogues thus an emotional chatting machine was devised to generate emotional responses. Liu et al. (2018) proposed a neural knowledge diffusion (NKD) model to introduce knowledge into dialogue generation.

5 Conclusion and Future Work

In this paper, we propose S2SPMN, a simple yet effective response generation model by exploiting relevant information contained in large dialogue dataset. Empirical studies indicate that simply selecting responses from training set as prototypes and integrating them into the generation process could dramatically improve the quality of generated responses. Moreover, our model is very flexible and could be adapted to any other Seq2Seq based generation methods. Most importantly, we claim the intrinsic difference between RG and MT and propose a new way to define response generation.

As the first work trying to help with the meaning transformation between source and target, we have obtained the encouraging progress. However, we know that there are still many directions to enrich the proposed framework. In future work, we would like to devise more sophisticated solutions to bridge the semantic gap in RG and explore linguistic patterns in conversations like what has been done in discourse analysis (Lei et al., 2018) .

Acknowledgments

This research was supported by National Natural Science Foundation of China (No. 61502344, No. 61872278), Natural Scientific Research Program of Wuhan University (No. 2042017kf0225). Chenliang Li is the corresponding author.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, and Dawei Yin. 2018. Hierarchical variational memory network for dialogue generation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1653–1662.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *ACL*, pages 199–208.

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

Xinyu Hua and Lu Wang. 2018. Neural argument generation augmented with externally retrieved evidence. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Min-Yen Kan, Xiangnan He, Wenqiang Lei, Xisen Jin, Zhaochun Ren, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *ACL*.

Florian Kreyssig, Iñigo Casanueva, Pawel Budzianowski, and Milica Gasic. 2018. Neural user simulation for corpus-based policy optimisation for spoken dialogue systems. In *SIGDIAL Conference*.

Wenqiang Lei, Yuanxin Xiang, Yuwei Wang, Qian Zhong, Meichun Liu, and Min-Yen Kan. 2018. Linguistic properties matter for implicit discourse relation recognition: Combining semantic interaction, topic continuity and attribution. In *AAAI*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL*, pages 110–119.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *EMNLP*, pages 1192–1202.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *EMNLP*, pages 2157–2169.

Wenjie Li, Furu Wei, Sujian Li, and Ziqiang Cao. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *ACL*.

Qun Liu, Yang Feng, Hongshen Chen, Zhaochun Ren, Dawei Yin, and Shuman Liu. 2018. Knowledge diffusion for neural dialogue generation. In *ACL*.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *COLING*, pages 3349–3358.

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49.

Minghui Qiu, Feng-Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu. 2017. Alime chat: A sequence to sequence and rerank based chatbot engine. In *ACL*, pages 498–503.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *EMNLP*, pages 583–593.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*, pages 1577–1586.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *HLT-NAACL*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei hao Su, David Vandyke, and Steve J. Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *HLT-NAACL*.

Yu Wu, Wei Wu, Dejian Yang, Can Xu, Zhoujun Li, and Ming Zhou. 2018. Neural response generation with dynamic vocabularies. In *AAAI*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, pages 3351–3357.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative chinese poetry generation using neural memory. In *ACL*.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018. Modeling multi-turn conversation with deep utterance aggregation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 3740–3752.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*, pages 654–664.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2017. Emotional chatting machine: emotional conversation generation with internal and external memory. *arXiv preprint arXiv:1704.01074*.

Improving Reinforcement Learning Based Image Captioning with Natural Language Prior

Tszhang Guo[1], Shiyu Chang[2], Mo Yu[2], Kun Bai[1]
[1]Mobile Internet Group, Tencent
[2]MIT-IBM Watson AI Lab, IBM Research
`{kkzkguo,kunbai}@tencent.com`
`shiyu.chang@ibm.com`, `yum@us.ibm.com`

Abstract

Recently, Reinforcement Learning (RL) approaches have demonstrated advanced performance in image captioning by directly optimizing the metric used for testing. However, this shaped reward introduces learning biases, which reduces the readability of generated text. In addition, the large sample space makes training unstable and slow. To alleviate these issues, we propose a simple coherent solution that constrains the action space using an n-gram language prior. Quantitative and qualitative evaluations on benchmarks show that RL with the simple add-on module performs favorably against its counterpart in terms of both readability and speed of convergence. Human evaluation results show that our model is more human readable and graceful. The implementation will become publicly available upon the acceptance of the paper[1].

1 Introduction

Image captioning (Farhadi et al., 2010; Kulkarni et al., 2011; Yao et al., 2017; Lu et al., 2016; Dai et al., 2017; Li et al., 2017) aims at generating natural language descriptions of images. Advanced by recent developments of deep learning, many captioning models rely on an encoder-decoder based paradigm (Vinyals et al., 2015), where the input image is encoded into hidden representations using a Convolutional Neural Network (CNN) followed by a Recurrent Neural Network (RNN) decoder to generate a word sequence as the caption. Further, the decoder RNN can be equipped with spatial attention mechanisms (Xu et al., 2015) to incorporate precise visual contexts, which often yields performance improvements empirically.

Although the encoder-decoder framework can be effectively trained with maximum likelihood estimation (MLE) (Salakhutdinov, 2010), recent research (Ranzato et al., 2015) have pointed out that the MLE based approaches suffer from the so-called exposure bias problem. To address this problem, (Ranzato et al., 2015) proposed a Reinforcement Learning (RL) based training framework. The method, developed on top of the REINFORCE algorithm (Williams, 1992), directly optimizes the non-differentiable test metric (*e.g.* BLEU (Papineni et al., 2002), CIDEr (Vedantam et al., 2015), METEOR (Banerjee and Lavie, 2005) *etc.*), and achieves promising improvements. However, learning with RL is a notoriously difficult task due to the high-variance of gradient estimation. Actor-critic (Sutton and Barto, 1998) methods are often adopted, which involves training an additional value network to predict the expected reward. On the other hand, (Rennie et al., 2017) designed a self-critical method that utilizes the output of its own test-time inference algorithm as the baseline to normalize the rewards, which leads to further performance gains.

Beside to the high-variance problem, we notice that there are two other drawbacks of RL-based captioning methods that are often overlooked in the literature. First, while these methods can directly optimize the non-differentiable rewards and achieve high test scores, the generated captions contain many repeated trivial patterns, especially at the end of the sequence. Table 1 shows examples of bad-endings generated by a self-critical based RL algorithm (model details refer to Section 4). Specifically, 46.44% generated captions end with phrases as “with a”, “on a”, “of a”, *etc.* (for detailed statistics see Appendix A), on the MSCOCO (Chen et al., 2015) validation set with the standard data splitting by (Karpathy and Li, 2015). The reason is that the shaped reward function biases the learning. In Figure 1, we see these additive patterns at the end of captions, although make no sense to humans, yield to a higher re-

---

[1] https://github.com/tgGuo15/PriorImageCaption

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 751–756
Brussels, Belgium, October 31 - November 4, 2018.

Image ID	Generated sentence	CIDEr
262262	a tall building with a clock tower with a	160.1
262148	a man doing a trick on a skateboard on a	146.5
52413	a person holding a cell phone in a	132.4
393225	a bow of soup with carrots and a	118.5

Table 1: Examples of bad sequences generated by a self-critical based RL baseline. Blue color indicates the bad ending. Sequences with bad endings have high CIDEr scores.

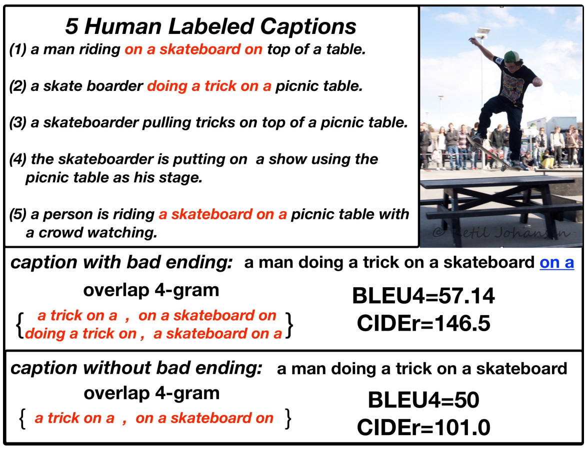

5 Human Labeled Captions

(1) a man riding on a skateboard on top of a table.

(2) a skate boarder doing a trick on a picnic table.

(3) a skateboarder pulling tricks on top of a picnic table.

(4) the skateboarder is putting on a show using the picnic table as his stage.

(5) a person is riding a skateboard on a picnic table with a crowd watching.

caption with bad ending: a man doing a trick on a skateboard on a

overlap 4-gram

{ a trick on a , on a skateboard on
doing a trick on , a skateboard on a }

BLEU4=57.14
CIDEr=146.5

caption without bad ending: a man doing a trick on a skateboard

overlap 4-gram

{ a trick on a , on a skateboard on }

BLEU4=50
CIDEr=101.0

Figure 1: A demonstration of the sequence with bad ending has higher BLEU and CIDEr scores compared to the one without.

ward. Empirically, removing these endings results in a huge performance drop of around 6%. (Paulus et al., 2017) has also reported that in abstractive summarization, using RL only achieves high ROUGE (Lin, 2004) score, yet the human-readability is very poor. The second drawback is that RL-based text generation is sample-inefficient due to the large action space. Specifically, the search space is of size $O(|\mathcal{V}|^T)$, where $\mathcal{V}$ is a set of words, T is the sentence length, and $|\cdot|$ denotes the cardinality of a set. This often makes training unstable and converge slowly.

In this work, to tackle these two issues, we propose a simple yet effective solution by introducing coherent language constraints on local action selections in RL. Specifically, we first obtain word-level n-gram (Kneser and Ney, 1995) model from the training set and then use it as an effective prior. During the action sampling step in RL, we reduce the search space of actions based on the constitution of the previous word contexts as well as our n-gram model. To further promote samples with high rewards, we sample multiple sentences during the training and update the policy based on the best-rewarded one. Such simple treatments prevent the appearance of bad endings and expedite the convergence while maintaining comparable performance to the pure RL counterpart. In addition, the proposed framework is generic, which can be applied to many different kinds of neural structures and applications.

2 Model Architecture

Encoder-Decoder Model: We adopt a similar structure as GNIC (Vinyals et al., 2015), which first encodes an image I to a dense vector h_I by CNN. The vector h_I is then fed as the input to an LSTM-based (Hochreiter and Schmidhuber, 1997) language model decoder. At each step t, the LSTM receives the previous output w_{t-1} as the input; computes the hidden state h_t; and predicts the next word w_t as below:

$$\begin{aligned} & h_t = \text{LSTM}(h_{t-1}, w_{t-1}), \quad l_t = W_l h_t \\ & w_t \sim \text{softmax}(l_t), \end{aligned} \tag{1}$$

where $w_0 = h_I$ and h_0 and c_0 are initialized to zero. The generation ends if a special token *end* is predicted.

Attention Model: Instead of utilizing a static representation of the image, attention mechanism dynamically reweights the spatial features from CNN to focus on the different region of the image at each word generation. We specifically consider the standard architecture used in (Xu et al., 2015), where $\mathcal{A} = \{a_1, a_2, ..., a_L\}$ is the spatial feature set and each $a_i \in R^D$ corresponds to features extracted at different image locations. Then the hidden states of the LSTM is computed as

$$\begin{aligned} & e_{ti} = f_{\text{att}}(a_i, h_{t-1}), \quad \beta_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{L} \exp(e_{tk})}, \\ & z_t = \sum_{k=1}^{L} \beta_{tk} a_k, \quad h_t = \text{LSTM}([h_{t-1}, z_t], w_{t-1}), \end{aligned} \tag{2}$$

where f_{att} is an attention model, which we use a single fully connected layer conditioned on the previous hidden state. Once h_t is obtained, the word generation is same as equation (1).

Sequence Generation with RL: We follow the training procedure of (Rennie et al., 2017). The decoder LSTM can be viewed as a "policy" denoted by p_θ, where θ is the set of parameters of the network. At each time step t, the policy chooses an action by generating a word w_t and obtains a new “state” (*i.e.* hidden states of LSTM, attention weights, *etc.*). Once the end token is generated, a “reward” r is given based on the score (*e.g.* CIDEr or BLEU) of the predicted sentence. The goal is to maximize the expected reward as

$$L(\theta) = \mathbb{E}_{w^s \sim p_\theta}[r(w^s)], \tag{3}$$

where $w^s = \{w_1^s, w_2^s, ..., w_T^s\}$ are sampled words at every time step. The REINFORCE algorithm (Williams, 1992) provides unbiased gradient estimation of θ as

$$\nabla_\theta L(\theta) \approx r(w^s) \nabla_\theta \log p_\theta(w^s), \tag{4}$$

using a single sequence.

Variance Reduction with Self-Critical: We reduce the variance of the gradient estimator by using the self-critical approach as

$$\nabla_\theta L(\theta) \approx (r(w^s) - r(\bar{w})) \nabla_\theta \log p_\theta(w^s), \tag{5}$$

where $\bar{w}_t$ is the baseline reward calculated by the current model under the inference algorithm used at test time defined as

$$\bar{w}_t = \arg\max_{w_t} p_\theta(w_t | h_t). \tag{6}$$

Then, sequences have rewards higher than $\bar{w}$ will be increased in probability, while samples result in lower reward will be suppressed.

3 Prior Language Constraint with N-Gram Model

Method: We collect all n-grams (n=3 or 4 in our experiments) from a corpus of captions. We use the training set from MSCOCO to avoid the usage of the additional resource. Thus, a fair comparison to previous methods is guaranteed. Then, we filter the n-grams with frequencies lower than five. The set of remaining ones is denoted as $\mathcal{F}$. During training, given the previous tokens predicted by the decoder, we constraint the sample space the current prediction by

$$w_t \sim \text{softmax}(p_\theta(w_t^s) \cdot \boldsymbol{\alpha_t}), \tag{7}$$

where $\boldsymbol{\alpha_i}$ is an indicator vector whose length is the vocabulary size $|\mathcal{V}|$ and its elements are non-zero only if the corresponding word and the previous $(n-1)$-gram constitute a valid n-gram in $\mathcal{F}$ as

$$\boldsymbol{\alpha_t}[k] = \begin{cases} 1 & \text{if } \{w_{t-n+1}^s, \cdots, w_{t-1}^s, k\} \in \mathcal{F} \\ 0 & \text{otherwise} \end{cases}. \tag{8}$$

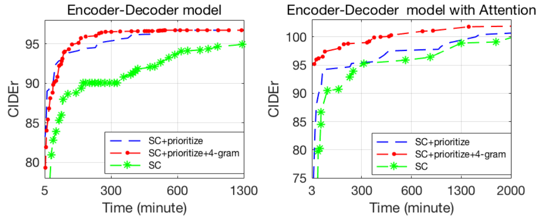

Figure 2: Training time of models with (right) and without (left) spatial attention.

Discussion: The key motivation for applying the above constraint is two-fold: (1) this ensures generated captions always formed by valid n-grams, which provides us a direct way of eliminating the repeated common phrases and bad-endings like the ones in Table 1; and (2) this shrinks the size of action space, which makes the training converges much faster. For MSCOCO, action space is changed from more than 9,000 to 56 on average.

4 Experiments

Dataset: We perform both quantitative and qualitative evaluations on MSCOCO dataset. The dataset contains 123,287 images and each image has at least five human captions. To seek fair comparison to others, we use the publicly available splits, which contains 82,783 training, 5,000 validation and 5,000 testing images.

Implementation Details: Our implementations are based on the publicly project.[2] We use an ImageNet pre-trained 101-layered ResNet[3] (He et al., 2016) to extract visual features. We consider two types (see Section 2) of architectural training with RL: (1) the plain encoder-decoder, and (2) the encoder-decoder with attention. For the former one, we represent each image by a 2,048-dimension vector by extracting the features from the last convolutional layer with average pooling. For the attention model, we apply spatial adaptive max pooling and the output feature map has the size of $14 \times 14 \times 2,048$. At each time step, the attention model produces weights over 196 spatial locations. The size of word embeddings and the hidden dimension of the LSTM are set to 512 for all experiments. More details are in Appendix B.

Compared Methods: We report our results in four different settings, which include the combinations of with/without attention and using tri-/four-

---

[2] https://github.com/ruotianluo/self-critical.pytorch

[3] https://github.com/KaimingHe/deep-residual-networks

	Methods	CIDEr	BLEU4	ROUGE-L	METEOR	BadEnd-Rate
Published	(Karpathy and Li, 2015)	66.0	23.0	- -	19.5	0.0
	(Xu et al., 2015)	- -	25.0	- -	23.0	0.0
	MIXER (Ranzato et al., 2015)	- -	29.1	- -	- -	- -
	(Ren et al., 2017)	93.7	30.4	52.5	25.1	- -
Implemented	ED-XE	89.8	28.0	51.7	24.2	0.0
	Att-XE	95.1	29.2	52.8	24.8	0.0
	ED-SC (Rennie et al., 2017)	101.8 / 96.1	31.2 / 30.3	53.1 / 52.9	24.6 / 23.9	46.4% / 0.0
	Att-SC (Rennie et al., 2017)	105.7 / 100.8	32.3 / 30.8	53.8 / 53.1	25.2 / 24.1	43.7% / 0.0
	Ours-ED-4-gram	96.7	29.1	51.4	23.9	0.0
	Ours-Att-4-gram	**102.0**	30.2	**53.6**	**25.6**	0.0
	Ours-ED-tri-gram	95.1	29.8	52.4	24.1	0.0
	Ours-Att-tri-gram	100.4	28.7	51.8	25.0	0.0

Table 2: Quantitative evaluation of our method compared to baselines on MSCOCO. Blue text indicates the performance after adjustments and **red** text indicates the best performance.

gram. We directly compare with our counterparts that have the same structures but no n-gram modules. Specifically, they are encoder-decoder based self-critical (ED-SC), and the one with attention (Att-SC). In addition, since our experimental setup is almost identical to many existing works, we also include their reported results, which include (Karpathy and Li, 2015; Xu et al., 2015; Ranzato et al., 2015; Ren et al., 2017). At last, we also include the performance of our warm-start models - the models trained by MLE (Vinyals et al., 2015) using cross entropy (ED-XE and Att-XE) - as a reference.

Evaluation Metric and Performance Adjustment: We report performance on FIVE metrics: BLEU4, METEOR, ROUGE-L,CIDEr and Bad Ending Rate. For the self-critical baselines, we report two sets of performances: 1) the captions directly generated by the model; and 2) the sequences of removing bad endings of the generated captions, based on the distribution in Appendix A.

Results: Table 2 summarizes the performances of our models compared with other baselines. We see that without performance adjustments, the self-critical RL with attention performs the best. However, since it contains many bad endings, our method achieves supreme results after these repeated patterns are removed. We also provide some qualitative comparison between our attention model and self-critical in Appendix C.

Efficient Training: We show that constraining the action space leads to a more efficient RL training in Figure 2. CIDEr score is calculated after removing bad endings. We plot three curves using architectures with/without attentions. The Green curve is the self-critical, the blue one is with prioritized sampling, and the red one is our final model with 4-gram constraint. We observe that we can speed up almost twice than its counterpart.

Online Evaluation: We also evaluate our attention model on COCO online server[4] and results are reported in Table 3. Att-SC gets a higher score than ours in the online test, however, with a lot of bad endings where the bad ending ratio is 72.7%.

Human Evaluation: We also implement human evaluation on the results generated by our Att-4-gram compared with Att-SC. We randomly select 200 images from the test set. Each time, one image with two captions generated by two different models are shown to the volunteer and three choices are provided: (1) the first one is better; (2) both are the same level; (3) the second one is better. See more details in Appendix D. In Table 5, our model wins 400 times and performs more closely to human than Att-SC.

Evaluating Captions Diversity: To further evaluate the quality of the caption model, we follow (Shetty et al., 2017) to measure the diversity of the generated captions. We compute the novelty score of our 4-gram model, which is defined as whether a particular caption has been observed in the training set. When two models have the same level predictive performances (*e.g.* CIDEr), a higher novelty score usually indicates more diverse generations. We conduct the experiment five times and report the averaged novelty score of our 4-gram model and the Att-SC, which are 77.83% and 59.28% respectively. As the reference, the METEOR and novelty scores reported in (Shetty et al., 2017) are 23.6, and 79.84%, respectively.

---

[4] https://competitions.codalab.org/competitions/3221

Methods	CIDEr	BLEU4	METEOR	ROUGE-L	BadEnd-Rate
Att-SC	**109.3**	**61.9**	32.9	67.7	72.7%
Att-4-gram	104.7	59.8	33.0	66.2	0
Att-LSTM-LM	104.3	61.0	**33.9**	**68.5**	0

Table 3: Quantitative results on online server (C40 test).**Red** text indicates the best performance.

Methods	CIDEr	BLEU4	ROUGE-L	METEOR	BadEnd-Rate
Att-SC (Rennie et al., 2017)	105.7 / 100.8	32.3 / 30.8	53.8 / 53.1	25.2 / 24.1	43.7% / 0.0
Ours-ED-4-gram	96.7	29.1	51.4	23.9	0.0
Ours-Att-4-gram	102.0	30.2	53.6	**25.6**	0.0
ED-LSTM-LM	99.4	30.9	52.7	24.6	0.0
Att-LSTM-LM	**105.9**	**32.8**	**54.1**	25.4	0.0

Table 4: Quantitative evaluation with our extension methods on MSCOCO. Blue text indicates the performance after adjustments and **red** text indicates the best performance.

Methods	4-gram win	Same level	4-gram lose
4-gram **VS** SC	400	349	251

Table 5: Human evaluation results for attention models

5 Neural Language Models Extension

Inspired by the paper reviews, we extend our model by adopting another language prior to evaluating the effectiveness of constraining action space during REINFORCE training. We train our neural language model based on the MSCOCO caption corpus with an LSTM unit.

LSTM Language Model: Given a word series $\{w_0, w_1, ..., w_T\}$, the target of a neural language model is to maximize the log-likelihood as:

$$\max_{\theta} \log p_\theta(w_0, w_1, ..., w_T). \tag{9}$$

We model $p_\theta(w_0, w_1, ..., w_T)$ by an LSTM unit:

$$\begin{aligned} \log p_\theta(w_0, ..., w_T) &= \sum_{t=1}^{T} \log p_{\theta_{LM}}(w_t|h_{t-1}) \\ h_t &= \text{LSTM}_{LM}(h_{t-1}, w_{t-1}), \end{aligned} \tag{10}$$

where w_0 is set to a *start* token for all sentences. h_0 and c_0 are initialized to zero. After obtaining the optimized θ^*_{LM}, we can use it to constrain the action space similar to the N-gram language model. Specifically, given previous $t-1$ sampled words from current caption model, we compute $p_{\theta^*_{LM}}(w_t|w_0, w_1, ..., w_{t-1})$, which is the probability of the next word over the entire vocabulary. We then apply a simple thresholding rule to form a subset of valid words for the captioning model.

$$\begin{aligned} \boldsymbol{\alpha_t}[k] &= \begin{cases} 1 & \text{if } \{k\} \in \mathcal{F} \\ 0 & \text{otherwise} \end{cases}, \text{where} \\ \mathcal{F} &= \{w_t | p_{\theta^*_{LM}}(w_t|w_0, w_1, ..., w_{t-1}) \geq \eta\}. \end{aligned} \tag{11}$$

η is a hyperparameter.

Additional Experiments The word embedding size and hidden dimension of θ_{LM} are set to 256 for this experiment. We use Adam optimizer for training language model and the learning rate is set to 0.001. The batch size of language model training and REINFORCE training are both set to 20 in the experiments. η is set to 0.00005 for the first word and increases by a factor of two for every timestep. We report our results in two settings, which include the combination of with/without attention for the caption model (termed ED-LSTM-LM and Att-LSTM-LM). We use the same warm-start models as in the N-gram experiments. The performances are summarized in Table 4 and Table 3. We see that the neural language model provides further performance gains compared to the N-gram model without introducing any bad-endings. This is because that the LSTM language model covers a larger context than N-gram, which helps to generate more accurate captions.

6 Conclusion

In this paper, we present a simple but efficient approach to RL-based image caption by considering n-gram language prior to constrain the action space. Our method converges faster and achieves better results than self-critical setting after removing bad endings in the generated captions. In addition, captions generated by our models are more human readable and graceful. We further extend our ideas using neural language model. The results demonstrate that the captioning models are more beneficial from the neural language model than the N-gram model.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL-workshop*, pages 228–231.

Xinlei Chen, Hao Fang, Tsung Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *Computer Science*.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *ICCV*, pages 2989–2998.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: generating sentences from images. *Lecture Notes in Computer Science*, 21(10):15–29.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

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

Andrej Karpathy and Fei Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

G. Kulkarni, V. Premraj, S. Dhar, Siming Li, Yejin Choi, A. C. Berg, and T. L. Berg. 2011. Baby talk: Understanding and generating simple image descriptions. In *CVPR*, pages 1601–1608.

Matt J. Kusner and Jos Miguel Hernndezlobato. 2016. Gans for sequences of discrete elements with the gumbel-softmax distribution.

Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. 2017. Scene graph generation from objects, phrases and region captions. In *ICCV*, pages 1270–1279.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL-workshop*, page 10.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2016. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. pages 3242–3250.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *Computer Science*.

Zhou Ren, Xiaoyu Wang, Ning Zhang, Xutao Lv, and LiJia Li. 2017. Deep reinforcement learning-based image captioning with embedding reward. In *CVPR*, pages 1151–1159.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, pages 1179–1195.

Ruslan Salakhutdinov. 2010. Learning deep generative models. 2(1):361–385.

Rakshith Shetty, Marcus Rohrbach, Lisa Anne Hendricks, Mario Fritz, and Bernt Schiele. 2017. Speaking the same language: Matching machine to human captions by adversarial training. In *ICCV*, pages 4155–4164.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057.

Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. 2017. Boosting image captioning with attributes. In *ICCV*, pages 4904–4912.

Training for Diversity in Image Paragraph Captioning

Luke Melas-Kyriazi **George Han** **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
{lmelaskyriazi@college, hanz@college, srush@seas}.harvard.edu

Abstract

Image paragraph captioning models aim to produce detailed descriptions of a source image. These models use similar techniques as standard image captioning models, but they have encountered issues in text generation, notably a lack of diversity between sentences, that have limited their effectiveness. In this work, we consider applying sequence-level training for this task. We find that standard self-critical training produces poor results, but when combined with an integrated penalty on trigram repetition produces much more diverse paragraphs. This simple training approach improves on the best result on the Visual Genome paragraph captioning dataset from 16.9 to 30.6 CIDEr, with gains on METEOR and BLEU as well, without requiring any architectural changes.

1 Introduction

Image captioning aims to describe the objects, actions, and details present in an image using natural language. Most image captioning research has focused on single-sentence captions, but the descriptive capacity of this form is limited; a single sentence can only describe in detail a small aspect of an image. Recent work has argued instead for image paragraph captioning with the aim of generating a (usually 5-8 sentence) paragraph describing an image.

Compared with single-sentence captioning, paragraph captioning is a relatively new task. The main paragraph captioning dataset is the Visual Genome corpus, introduced by Krause et al. (2016). When strong single-sentence captioning models are trained on this dataset, they produce repetitive paragraphs that are unable to describe diverse aspects of images. The generated paragraphs repeat a slight variant of the same sentence multiple times, even when beam search is used. Prior work, discussed in the following section, tried to address this repetition with architectural changes, such as hierarchical LSTMs, which separate the generation of sentence topics and words.

In this work, we consider an approach for training paragraph captioning models that focuses on increasing the diversity of the output paragraph. In particular, we note that self-critical sequence training (SCST) (Ranzato et al., 2015; Rennie et al., 2016), a technique which uses policy gradient methods to directly optimize a target metric, has been successfully employed in standard captioning, but not in paragraph captioning. We observe that during SCST training the intermediate results of the system lack diversity, which makes it difficult for the model to improve. We address this issue with a simple repetition penalty which downweights trigram overlap.

Experiments show that this technique greatly improves the baseline model. A simple baseline, non-hierarchical model trained with repetition-penalized SCST outperforms complex hierarchical models trained with both cross-entropy and customized adversarial losses. We demonstrate that this strong performance gain comes from the combination of repetition-penalized search and SCST, rather than from either individually, and discuss how this impacts the output paragraphs.

2 Background and Related Work

Nearly all modern image captioning models employ variants of an encoder-decoder architecture. As introduced by Vinyals et al. (2014), the encoder is a CNN pre-trained for classification and the decoder is a LSTM or GRU. Following work in machine translation, Xu et al. (2015) added an attention mechanism over the encoder features. Recently, Anderson et al. (2017) further improved single-sentence captioning performance by incorporating object detection in the encoder (bottom-up attention) and adding an LSTM layer before attending to spatial features in the decoder (top-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 757–761
Brussels, Belgium, October 31 - November 4, 2018.

down attention).

Single-sentence and paragraph captioning models are evaluated with a number of metrics, including some designed specifically for captioning (CIDEr) and some adopted from machine translation (BLEU, METEOR). CIDEr and BLEU measure accuracy with n-gram overlaps, with CIDEr weighting n-grams by TF-IDF (term-frequency inverse-document-frequency), and METEOR uses unigram overlap, incorporating synonym and paraphrase matches. We discuss these metrics in greater detail when analyzing our experiments.

Related Models Krause et al. (2016) introduced the first large-scale paragraph captioning dataset, a subset of the Visual Genome dataset, along with a number of models for paragraph captioning. Empirically, they showed that paragraphs contain significantly more pronouns, verbs, coreferences, and greater overall "diversity" than single-sentence captions. Whereas most single-sentence captions in the MSCOCO dataset describe only the most important object or action in an image, paragraph captions usually touch on multiple objects and actions.

The paragraph captioning models proposed by Krause et al. (2016) included template-based (non-neural) approaches and two encoder-decoder models. In both neural models, the encoder is an object detector pre-trained for dense captioning. In the first model, called the flat model, the decoder is a single LSTM which outputs an entire paragraph word-by-word. In the second model, called the hierarchical model, the decoder is composed of two LSTMs, where the output of one sentence-level LSTM is used as input to the other word-level LSTM.

Recently, Liang et al. (2017) extended this model with a third (paragraph-level) LSTM and added adversarial training. In total, their model (RTT-GAN) incorporates three LSTMs, two attention mechanisms, a phrase copy mechanism, and two adversarial discriminators. To the best of our knowledge, this model achieves state-of-the-art performance of 16.9 CIDEr on the Visual Genome dataset (without external data).

For our experiments, we use the top-down single-sentence captioning model in Anderson et al. (2017). This model is similar to the "flat" model in Krause et al. (2016), except that it incorporates attention with a top-down mechanism.

3 Approach

The primary issue in current paragraph captioning models, especially non-hierarchical ones, is lack of diversity of topic in the output paragraph. For example, for the image of a skateboarder in Figure 1, the flat model outputs "The man is wearing a black shirt and black pants" seven times. This example is not anomalous: it is a typical failure case of the model. Empirically, in validation, ground truth paragraphs contain 0.62 repeated trigrams on average, whereas paragraphs produced by the flat cross-entropy model contain 25.9 repeated trigrams on average.

3.1 Self-Critical Sequence Training

Self-critical sequence training (SCST) is a sequence-level optimization procedure proposed by Rennie et al. (2016), which has been widely adopted in single-sentence captioning but has not yet been applied to paragraph captioning. This method provides an alternative approach to word-level cross-entropy which can incorporate a task specific metric.

Sequence-level training employs a policy gradient method to optimize directly for a non-differentiable metric, such as CIDEr or BLEU. This idea was first applied to machine translation by Ranzato et al. (2015) in a procedure called MIXER, which incrementally transitions from cross-entropy to policy gradient training. To normalize the policy gradient reward and reduce variance during training, MIXER subtracts a baseline estimate of the reward as calculated by a linear regressor.

SCST replaces this baseline reward estimate with the reward obtained by the test-time inference algorithm, namely the CIDEr score of greedy search. This weights the gradient by the difference in reward given to a sampled paragraph compared to the current greedy output (see Eq. 3-9 in (Rennie et al., 2016)). Additionally, SCST uses a hard transition from cross-entropy to policy gradient training. The final gradient is:

$$-\mathbb{E}_{w^s \sim p_\theta}[(r(w^s) - r(w^g))\nabla_\theta \log p_\theta(w^s \mid x)]$$

Where w^s is a sampled paragraph, w^g is a greedy decoded paragraph, r is the reward (e.g CIDEr), p_θ is the captioning model.

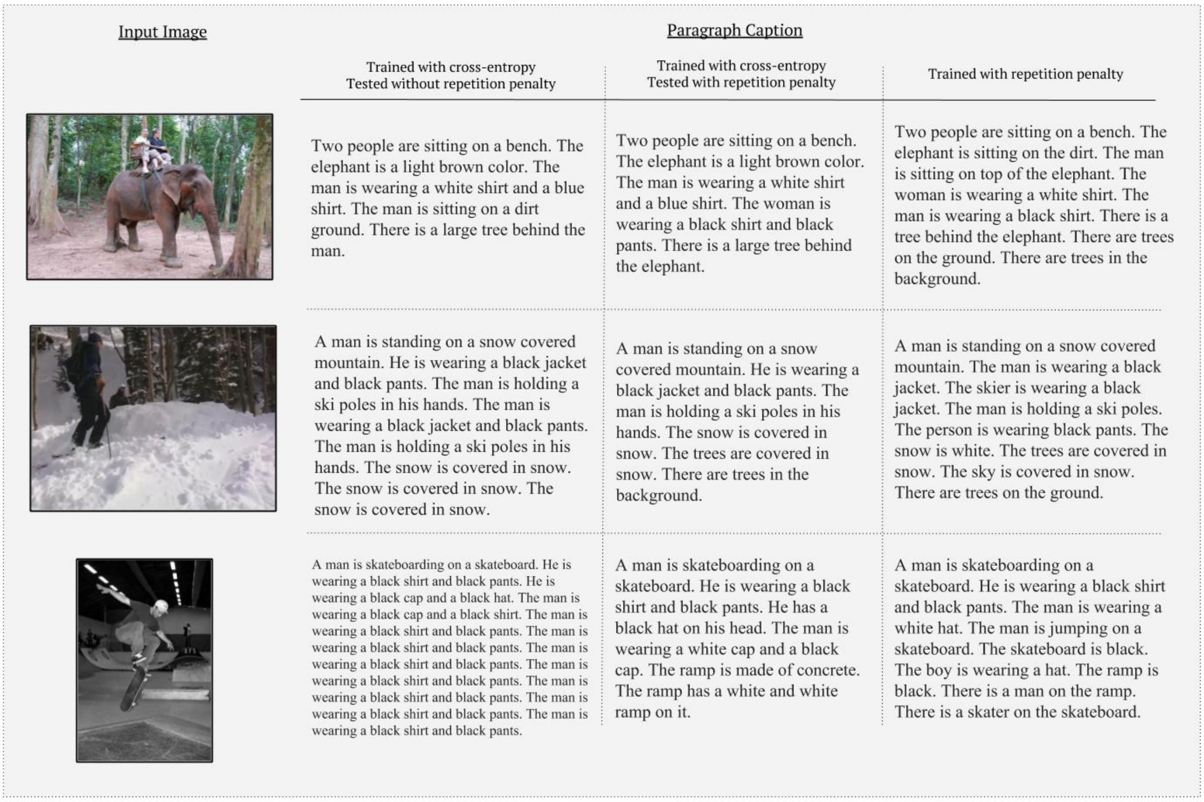

Input Image	Paragraph Caption Trained with cross-entropy Tested without repetition penalty	 Trained with cross-entropy Tested with repetition penalty	 Trained with repetition penalty
	Two people are sitting on a bench. The elephant is a light brown color. The man is wearing a white shirt and a blue shirt. The man is sitting on a dirt ground. There is a large tree behind the man.	Two people are sitting on a bench. The elephant is a light brown color. The man is wearing a white shirt and a blue shirt. The woman is wearing a black shirt and black pants. There is a large tree behind the elephant.	Two people are sitting on a bench. The elephant is sitting on the dirt. The man is sitting on top of the elephant. The woman is wearing a white shirt. The man is wearing a black shirt. There is a tree behind the elephant. There are trees on the ground. There are trees in the background.
	A man is standing on a snow covered mountain. He is wearing a black jacket and black pants. The man is holding a ski poles in his hands. The man is wearing a black jacket and black pants. The man is holding a ski poles in his hands. The snow is covered in snow. The snow is covered in snow. The snow is covered in snow.	A man is standing on a snow covered mountain. He is wearing a black jacket and black pants. The man is holding a ski poles in his hands. The snow is covered in snow. The trees are covered in snow. There are trees in the background.	A man is standing on a snow covered mountain. The man is wearing a black jacket. The skier is wearing a black jacket. The man is holding a ski poles. The person is wearing black pants. The snow is white. The trees are covered in snow. The sky is covered in snow. There are trees on the ground.
	A man is skateboarding on a skateboard. He is wearing a black shirt and black pants. He is wearing a black cap and a black hat. The man is wearing a black cap and a black shirt. The man is wearing a black shirt and black pants. The man is wearing a black shirt and black pants. The man is wearing a black shirt and black pants. The man is wearing a black shirt and black pants. The man is wearing a black shirt and black pants. The man is wearing a black shirt and black pants. The man is wearing a black shirt and black pants.	A man is skateboarding on a skateboard. He is wearing a black shirt and black pants. He has a black hat on his head. The man is wearing a white cap and a black cap. The ramp is made of concrete. The ramp has a white and white ramp on it.	A man is skateboarding on a skateboard. He is wearing a black shirt and black pants. The man is wearing a white hat. The man is jumping on a skateboard. The skateboard is black. The boy is wearing a hat. The ramp is black. There is a man on the ramp. There is a skater on the skateboard.

Figure 1: Example paragraph outputs of our model. The final example is a failure case of both our model and the non-penalized model. Our model does not suffer from the repetition problem of the non-penalized, but it does not produce a great caption because it does not understand that the image is black-and-white.

3.2 Repetition Penalty

In preliminary experiments, we find that directly applying SCST is not effective for paragraph captioning models. Table 1 shows that when training with SCST, the model performs only marginally better than cross-entropy. In further analysis, we see that the greedy baseline in SCST training has very non-diverse output, which leads to poor policy gradients. Unlike in standard image captioning, the cross-entropy model is too weak for SCST to be effective.

To address this problem, we take inspiration from recent work in abstractive text summarization, which encounters the same challenge when producing paragraph-length summaries of documents (Paulus et al., 2017). These models target the repetition problem by simply preventing the model from producing the same trigram more than once during inference. We therefore introduce an inference constraint that penalizes the log-probabilities of words that would result in repeated trigrams. The penalty is proportional to the number of times the trigram has already been generated.

Formally, denote the (pre-softmax) output of the LSTM by o, where the length of o is the size of the target vocabulary and o_w is the log-probability of word w. We modify o_w by $o_w \to o_w - k_w \cdot \alpha$, where k_w is the number of times the trigram completed by word w has previously been generated in the paragraph, and α is a hyperparameter which controls the degree of blocking. When $\alpha = 0$, there is no penalty, so we have standard greedy search. When $\alpha \to \infty$, or in practice when α exceeds about 5, we have full trigram blocking.

We incorporate this penalty into the greedy baseline used to compute policy gradients in SCST. During inference, we employ the same repetition-penalized greedy search.

4 Methods and Results

For our paragraph captioning model we use the top-down model from Anderson et al. (2017). Our encoder is a convolutional network pretrained for object detection (as opposed to dense captioning, as in Krause et al. (2016) and Liang et al. (2017)).

	METEOR	CIDEr	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Krause et al. (Template)	14.31	12.15	37.47	21.02	12.30	7.38
Krause et al. (Flat w/o object detector)	12.82	11.06	34.04	19.95	12.20	7.71
Krause et al. (Flat)	13.54	11.14	37.30	21.70	13.07	8.07
Krause et al. (Hierarchical)	15.95	13.52	41.90	24.11	14.23	8.69
Liang et al. (w/o discriminator)	16.57	15.07	41.86	24.33	14.56	8.99
Liang et al.	17.12	16.87	41.99	24.86	14.89	9.03
Ours (XE training, w/o rep. penalty)	13.66	12.89	32.78	19.00	11.40	6.89
Ours (XE training, w/ rep. penalty)	15.17	22.68	35.68	22.40	14.04	8.70
Ours (SCST training, w/o rep. penalty)	13.63	13.77	29.67	16.45	9.74	5.88
Ours (SCST training, w/ rep. penalty)	**17.86**	**30.63**	**43.54**	**27.44**	**17.33**	**10.58**

Table 1: Results of our model compared with prior published results. Note that Liang et al. (2017) also trains a model on additional data, but here we only compare models trained on Visual Genome. Also note that our models employ greedy search, whereas other models employ beam search.

The encoder extracts between 10 and 100 objects per image and applies spatial max-pooling to yield a single feature vector of dimension 2048 per object. The decoder is a 1-layer LSTM with hidden dimension 512 and top-down attention.

Evaluation is done on the Visual Genome dataset with the splits provided by Krause et al. (2016). We first train for 25 epochs with cross-entropy (XE) loss, using Adam with learning rate $5 \cdot 10^{-4}$. We then train an additional 25 epochs with repetition-penalized SCST targeting a CIDEr-based reward, using Adam with learning rate $5 \cdot 10^{-5}$.

Our PyTorch-based implementation is available at `https://github.com/lukemelas/image-paragraph-captioning`.

Results Table 1 shows the main experimental results. Our baseline cross-entropy captioning model gets similar scores to the original flat model. When the repetition penalty is applied to a model trained with cross-entropy, we see a large improvement on CIDEr and a minor improvement on other metrics.[1] When combining the repetition penalty with SCST, we see a dramatic improvement across all metrics, and particularly on CIDEr. Interestingly, SCST only works when its baseline reward model is strong; for this reason the combination of the repetition penalty and SCST is particularly effective.

Table 2 compares the effect of training with different values of the penalty hyperparameter α, demonstrating that intermediate values of α ($\approx$ 2.0) perform slightly better than large values. An intermediate value of α discourages the model from producing repeat trigrams, but still permits the model to output them when there are no likely alternative phrases.

α	METEOR	CIDEr	BLEU-4
0.0	13.8	13.6	5.9
1.0	17.4	28.9	10.2
2.0	**17.7**	**31.4**	**10.8**
4.0	17.6	30.1	10.4
10.0	17.5	30.6	9.9

Table 2: Varying the repetition penalty α (on the validation set). $\alpha = 10$ is equivalent to trigram blocking.

	XE w/o penalty	XE w/ penalty	SCST w/ penalty
Avg. # of trigram repeats in output	25.9	0.67	3.70
Avg. # unique trigram overlaps btw. output and ground truth	2.23	2.97	3.49

Table 3: Analysis of different model outputs ($\alpha = 2.0$ for models w/ penalty)

Finally, Table 3 shows quantitative changes in trigram repetition and ground truth matches. The cross-entropy model fails to generate enough unique phrases. Blocking these entirely gives some benefit, but the SCST model is able to raise the total number of matched trigrams while reintroducing few repeats.

---

[1] We believe the improvement may be largest on CIDEr because, in the calculation of CIDEr, n-gram counts are clipped to the number of times each n-gram appears in the reference sentence. However, a similar procedure is applied in calculating BLEU, on which we show lesser improvements.

5 Conclusion

This work targets increased diversity in image paragraph captioning. We show that training with SCST combined with a repetition penalty leads to a substantial improvement in the state-of-the-art for this task, without requiring architectural changes or adversarial training. In future work, we hope to further address the language issues of paragraph generation as well as extend this simple approach to other tasks requiring long-form text or paragraph generation.

Acknowledgements

We would like to thank Ruotian Luo, Sang Phan, and all contributors to OpenNMT-py, for their work on open-source implementations of different image captioning, video captioning, and translation models. AMR is supported by NSF-CCF 1704834, Google, Facebook, Bloomberg, and Amazon research awards.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2017. Bottom-up and top-down attention for image captioning and vqa. *arXiv preprint arXiv:1707.07998*.

Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. 2016. Re-evaluating automatic metrics for image captioning. *arXiv preprint arXiv:1612.07600*.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. 2016. A hierarchical approach for generating descriptive image paragraphs. *arXiv preprint arXiv:1611.06607*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations.

Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan, and Eric P Xing. 2017. Recurrent topic-transition gan for visual paragraph generation.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Ruotian Luo, Brian L. Price, Scott Cohen, and Gregory Shakhnarovich. 2018. Discriminability objective for training descriptive captions. *CoRR*, abs/1803.04376.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044.

A Graph-theoretic Summary Evaluation for ROUGE

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, Fang Chen
University of New South Wales, Sydney, Australia
Data61 CSIRO, Sydney, Australia
{elahehs,mohammade,wong,fang}@cse.unsw.edu.au

Abstract

ROUGE is one of the first and most widely used evaluation metrics for text summarization. However, its assessment merely relies on surface similarities between peer and model summaries. Consequently, ROUGE is unable to fairly evaluate summaries including lexical variations and paraphrasing. We propose a graph-based approach adopted into ROUGE to evaluate summaries based on both lexical and semantic similarities. Experiment results over TAC AESOP datasets show that exploiting the lexico-semantic similarity of the words used in summaries would significantly help ROUGE correlate better with human judgments.

1 Introduction

Quantifying the quality of summaries is an important and necessary task in the field of automatic text summarization. Among the metrics proposed for this task (Hovy et al., 2006; Tratz and Hovy, 2008; Giannakopoulos et al., 2008), ROUGE (Lin, 2004) is the first and still most widely used one (Graham et al., 2015). This metric measures the concordance of system-generated (peer) summaries and human-generated reference (model) summaries by determining n-grams, word sequences, and word pair matches. ROUGE assumes that a peer summary is of high quality if it shares many words or phrases with a model summary. However, different terminology may be used to refer to the same concepts and hence relying only on lexical overlaps may underrate content quality scores. For clarity, consider the following two sentences:

(i) *They strolled around the city.*
(ii) *They took a walk to explore the town.*

These sentences are semantically similar, but lexically different. If one of them is included in a model summary, while a peer summary contains another one, ROUGE or other surface based evaluation metrics cannot capture their similarity due to the minimal lexical overlap. We aim to help ROUGE with identifying the semantic similarities of linguistic items, and consequently tackling the main problem of its bias towards lexical similarities.

Considering senses instead of words, we use the Personalized PageRank (PPR) algorithm (Haveliwala, 2002) to leverage repetitive random walks on WordNet 3.0 (Fellbaum, 1998) as a semantic network. We disambiguate each word into its intended sense, and obtain the probability distribution of each sense over all senses in the network. Weights in this distribution denote the relevance of the corresponding senses. At each iteration, we measure the semantic similarity by looking at the path taken by the random walker, and weighting the overlaps between a pair of ranked PPR vectors. Our graph-based approach (ROUGE-G) computes semantic similarity scores between n-grams, along with their match counts, to perform both semantic and lexical comparisons of peer and model summaries. The experiment results indicate that ROUGE-G variants significantly outperform their corresponding variants of ROUGE. Beyond enhancing the evaluation prowess of ROUGE, due to its lexico-semantic analysis of summaries, we believe that ROUGE-G has the potential to expand the applicability of ROUGE to abstractive summarization.

2 Background

In the summarization literature, a couple of ROUGE variants (i.e., ROUGE-1, ROUGE-2, ROUGE-SU4) are reported to have a strong correlation with human assessments, and are frequently used to evaluate summaries (Lin and Och, 2004; Owczarzak and Dang, 2011; Over and Yen, 2004). Although ROUGE is a popular evaluation metric, improving the current evaluation metrics is

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 762–767
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

still an open research area. Many of these efforts are analyzed and gathered in the surveys provided by Steinberger and Ježek (2012). Herein, we try to briefly review the most significant ones. Since DUC 2005, the Pyramid metric (Passonneau et al., 2005) was introduced as one of the principal metrics for evaluating summaries in the TAC conference. However, this metric is semi-automated and requires manual identification of summary content units (SCUs). Soon after, Hovy et al. (2006) proposed a metric based on comparison of basic syntactic units (Basic Elements) between peer and model summaries. This metric, BE-HM, was specified as one of the baselines in the TAC AESOP task. Among systems participated in this task from 2009 to 2011, AutoSummENG (DEMOKRITOSGR) (Giannakopoulos et al., 2008) is one of the top systems which compares the graph representations of peer and model summaries. Recently, some evaluation metrics have studied the effectiveness of word semantic similarity to evaluate summaries including terminology variations and paraphrasing (Baroni et al., 2014; ShafieiBavani et al., 2017, 2018). For instance, an automated variant of the Pyramid metric has used distributional semantics to map text content within peer summaries to SCUs (Passonneau et al., 2013). A more recent metric, ROUGE-WE, (Ng and Abrecht, 2015) has also enhanced ROUGE by incorporating the use of a variant of word embeddings, called word2vec (Mikolov et al., 2013).

3 Graph-Theoretic Summary Evaluation

Given a pair of peer and model summaries, we compute PPR vectors at the following levels: (i) *sense level*, to disambiguate each word (having a set of senses); and (ii) *n-gram level*, to measure the semantic similarity. We compare the PPR vectors of each pair of n-grams using the following measures: (i) *Path-based:* considering the path that the random walker takes at each iteration to get to a particular node; (ii) *Rank and Weight:* weighting the overlaps between a pair of ranked PPR vectors.

3.1 Vector Representation

The WordNet graph has edges of various types, with the main types being hypernymy and meronymy to connect nodes containing senses. However, we do not use these types, and consider an edge as an undirected semantic or lexical relation between two synsets. We have utilized the WordNet graph enriched by connecting a sense - irrespective of its part-of-speech (POS) - with all the other senses that appear in its disambiguated gloss (Pilehvar and Navigli, 2015). Dimension of the vector representation is the number of connected nodes in the graph. For better clarity, we consider the adjacency matrix A for our semantic graph, and perform iterative random walks beginning at a set of senses S on WordNet with the probability mass of $p^{(0)}(S)$, which is uniformly distributed across the senses $s_i \in S$, and the mass for all $s_i \notin S$ set to zero. This provides a frequency or multinomial distribution over all senses in WordNet, with a higher probability assigned to senses that are frequently visited. The PPR vector of S is given by:

$$p^{(k)}(S) = dAp^{(k-1)}(S) + (1-d)p^{(0)}(S) \qquad (1)$$

At each iteration, the random walker may follow one of the edges with probability d or jump back to any node $s_i \in S$ with probability $(1-d)/|S|$. Following the standard convention, the value of damping factor d is set to 0.85. The number of iterations k is also set to 20, which is sufficient for the distribution to converge.

3.2 Comparing Vectors

Conventional measures for comparing PPR vectors calculate the probability that a random walker meets a particular node after a specific number of iterations, which is potentially problematic (Rothe and Schütze, 2014). For example, consider the following connected nodes:

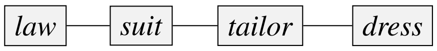

The PPR vectors of *suit* and *dress* have some weight on *tailor*, which is desirable. However, the PPR vector of *law* will also have a non-zero weight for *tailor*. Consequently, *law* and *dress* are spuriously similar because of the node *tailor*. To prevent this type of false similarity, the random walker needs to take into account the walking path to reach a particular node (Rothe and Schütze, 2014). We formalize this by defining the semantic similarity of two sets of nodes I and J as:

$$Sim_{sem}(I,J) = \sum_{x=0}^{k} c^x \times RW(p^{(x)}(I), p^{(x)}(J)) \qquad (2)$$

where damping factor c was optimized on the TAC 2010 (Owczarzak and Dang, 2010) AESOP dataset, and set to 0.7 to ensure that early meetings are more valuable than later meetings. At each iteration x, we compare PPR vectors by ranking their dimensions (senses) based on their values, and weighting the overlaps between them (Equation 3). Hence, we weight the similarity such that differences in the highest ranks (most important senses in a vector) are penalized more than differences in lower ranks. This measure has proven to be superior to cosine similarity, Jensen-Shannon divergence, and Rank-Biased Overlap for comparing vectors (Pilehvar et al., 2013).

$$RW(Y,Z) = \begin{cases} \frac{\sum_{h \in H} (r_h(Y) + r_h(Z))^{-1}}{\sum_{i=1}^{|H|} (2i)^{-1}}, & \text{if } |H| > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where H is the intersection of all senses with non-zero probability in both vectors Y and Z. $r_h(Y)$ denotes the rank of sense h in vector Y, where rank 1 is the highest rank. The denominator is used as a normalization factor that guarantees a maximum value of one. The minimum value is zero and occurs when there is no overlap, i.e., $|H| = 0$.

3.3 Calculating ROUGE-G

We combine lexical and semantic similarities to compute ROUGE-G-N:

$$\text{ROUGE-G-N} = \frac{\sum_{M \in \{ModelSums\}} \sum_{\substack{n\text{-}gram_m \in M, \\ n\text{-}gram_p \in PeerSum}} Sim_{LS}(n\text{-}gram_m, n\text{-}gram_p)}{\sum_{M \in \{ModelSums\}} \sum_{n\text{-}gram_m \in M} Count(n\text{-}gram_m)} \tag{4}$$

where Sim_{LS} is the score of lexico-semantic similarity between a pair of n-grams in model summary ($n\text{-}gram_m$) and peer summary ($n\text{-}gram_p$):

$$\begin{aligned} Sim_{LS}(n\text{-}gram_m, n\text{-}gram_p) = \\ \beta \times Count_{match}(n\text{-}gram_m, n\text{-}gram_p) \\ + (1-\beta) \times Sim_{sem}(n\text{-}gram_m, n\text{-}gram_p) \end{aligned} \tag{5}$$

Scaling factor β was optimized on the TAC 2010 AESOP dataset, and set to 0.5 to reach the best correlation with the manual metrics. $Count_{match}(n\text{-}gram_m, n\text{-}gram_p)$ is the maximum number of the $n\text{-}gram$ co-occurring in a peer summary and a set of model summaries.

3.4 Disambiguation of n-grams

Prior to measuring semantic similarities, each word in n-grams has to be analyzed and disambiguated into its intended sense. However, conventional word sense disambiguations are not applicable due to the lack of contextual information. Hence, we seek the semantic alignment that maximizes the similarity of the senses of the compared words. As an example (Pilehvar et al., 2013), consider two sentences of *"a1. Officers fired."* and *"a2. Several policemen terminated in corruption probe."*, the semantic alignment procedure has been performed as $"P_{a1}.\ officer^3_n, fire^4_v"$, *and* $"P_{a2}.\ policeman^1_n, terminate^4_v, corruption^6_n, probe^1_n"$. t^i_p denotes the i-th sense of a word t in WordNet with POS p. After alignment, among all possible pairings of all senses of $fire_v$ to all senses of all words in *a2*, the sense $fire^4_v$ (employment termination) obtains the maximal similarity value of $Sim_{sem}(fire^4_v, terminate^4_v) = 1$.

3.5 OOV Handling

Out-of-vocabulary (OOV) words are the words that are not defined in the corresponding lexical resource, hence, they will be ignored while generating PPR vectors. The reason is that they do not have an associated node in the WordNet graph for the random walk to be initialized from. To take them into consideration, we add an extra dimension for each OOV term in the resulting PPR vector. Following Pilehvar and Navigli (2015), we set the associated weights of the new dimensions to 0.5 so as to guarantee their placement among the top dimensions in their vectors.

4 Experiments

4.1 Data and Metrics

The only available datasets for the task of Summarization Evaluation are three AESOP datasets[1] provided by TAC 2009, 2010, and 2011. Among them, we optimize scaling factors using the TAC 2010 AESOP dataset, and evaluate ROUGE-G on the TAC 2011 (Owczarzak and Dang, 2011) AESOP dataset for two main reasons: (i) it is the only dataset on which evaluation metrics can be assessed for their ability to measure summary Readability; (ii) To be in line with the most recent work (ROUGE-WE) that has also been evaluated

---

[1] https://tac.nist.gov/data/

only on this dataset for measuring the Readability scores. This dataset consists of 44 topics, and a set of 10 documents for each topic. There are four human-crafted model summaries for each document set. A summary for each topic is generated by each of the 51 summarizers which participated in the main TAC summarization task. The output of participating automatic metrics is tasked to be compared against human judgments using three manual metrics of *Pyramid*, *Readability*, and *Responsiveness*. Hence, the outputs are scored based on their summary content, linguistic quality, and a combination of both, respectively.

Prior to computing correlation of ROUGE-G variants with manual metrics, ROUGE-G scores have reliably been computed (95% confidence intervals) under ROUGE bootstrap resampling with the default number of sampling point (1000). Correlation of ROUGE-G evaluation scores with the human judgments is then assessed with three metrics of correlation: Pearson r; Spearman ρ; and Kendall τ. We compute scores using the default NIST settings for baselines in the TAC 2011 AESOP task (with stemming and keeping stopwords).

4.2 Results

We evaluate ROUGE-G, against the top metrics (C_S_IIITH3, DemokritosGR1, Catolicasc1) among the 23 metrics participated in TAC AESOP 2011, ROUGE, and the most recent related work (ROUGE-WE) (Table 1). Overall results support our proposal to consider semantics besides surface with ROUGE. Since the large/small differences in competing correlations with human assessment are not an acceptable proof of superiority/inferiority in performance of one metric over another, significance tests should be applied. To better clarify the effectiveness of ROUGE-G, we have used pairwise Williams significance test recommended by Graham et al. (2015) for summarization evaluation. Accordingly, evaluation of a given summarization metric, M_{new}, takes the form of quantifying three correlations: $r(M_{new}, H)$, that exists between the evaluation metric scores for summarization systems and corresponding human assessment scores; $r(M_{base}, H)$, that stands for the correlation of baseline metrics with human judges; and the third correlation, between evaluation metric scores themselves, $r(M_{base}, M_{new})$. It can happen for a pair of competing metrics for which the correlation between metric scores is strong, that a small difference in competing correlations with human assessment is significant, while, for a different pair of metrics with a larger difference in correlation, the difference is not significant (Graham et al., 2015). Using this significance test, the results show that all increases in correlations of ROUGE-G compared to ROUGE and ROUGE-WE variants are statistically significant ($p < 0.05$). We analyze the correlation results reported in Table 1 in the following.

ROUGE-G-2 achieves the best correlation with Pyramid, regarding all correlation metrics. Moreover, every ROUGE-G variant outperforms its corresponding ROUGE and ROUGE-WE variants, regardless of the correlation metric used. However, the only exception is ROUGE-SU4, which correlates slightly better with Pyramid when measuring with Pearson correlation. One possible reason is that Pyramid measures content similarity between peer and model summaries, while the variants of ROUGE-G favor semantics behind the content for measuring similarities. Since some of the semantics attached to the skipped words are lost in the construction of skip-bigrams, ROUGE-SU4 shows a better correlation comparing to ROUGE-G-SU4.

For Responsiveness, ROUGE-G-SU4 achieves the best correlation when measuring with Pearson. We also observe that ROUGE-G-2 obtains the best correlation with Responsiveness while measuring with the Spearman and Kendall rank correlations. The reason is that semantic interpretation of bigrams is easier, and that of contiguous bigrams is much more precise. We also see that every variant of ROUGE-G outperforms its corresponding ROUGE and ROUGE-WE variants.

The readability score is based on grammaticality, structure, and coherence. Although our main goal is not to improve the readability, ROUGE-G-SU4 and ROUGE-G-2 are observed to correlate very well with this metric when measured with the Pearson and Spearman/Kendall rank correlations, respectively. Besides, every variant of ROUGE-G represents the best correlation results comparing to its corresponding variants of ROUGE and ROUGE-WE for all correlation metrics. This is likely due to considering word types and POS tagging while aligning and disambiguating n-grams. POS features are shown by Feng et al. (2010) to be helpful in predicting linguistic quality.

We optimize scaling factor β (Equation 5) on the TAC 2010 AESOP dataset. Figure 1 shows

	Pyramid			Responsiveness			Readability		
Metric	Pearson	Spearman	Kendall	Pearson	Spearman	Kendall	Pearson	Spearman	Kendall
C_S_IIITH3	0.965	0.903	0.758	0.933	0.781	0.596	0.731	0.358	0.242
DemokritosGR1	0.974	0.897	0.747	0.947	0.845	0.675	0.794	0.497	0.359
Catolicasc1	0.967	0.902	0.735	0.950	0.837	0.666	0.819	0.494	0.366
ROUGE-1	0.966	0.909	0.747	0.935	0.818	0.633	0.790	0.391	0.285
ROUGE-2	0.961	0.894	0.745	0.942	0.790	0.610	0.752	0.398	0.293
ROUGE-SU4	0.981	0.894	0.737	0.955	0.790	0.602	0.784	0.395	0.293
ROUGE-WE-1	0.949	0.914	0.753	0.916	0.819	0.631	0.785	0.431	0.322
ROUGE-WE-2	0.977	0.898	0.744	0.953	0.797	0.615	0.782	0.414	0.304
ROUGE-WE-SU4	0.978	0.881	0.720	0.954	0.787	0.597	0.793	0.407	0.302
ROUGE-G-1	0.971	0.915	0.758	0.944	0.825	0.638	0.791	0.434	0.330
ROUGE-G-2	**0.983**	**0.926**	**0.774**	0.956	**0.869**	**0.713**	0.790	**0.516**	**0.385**
ROUGE-G-SU4	0.979	0.898	0.741	**0.957**	0.814	0.616	**0.823**	0.445	0.334

Table 1: Correlation results with the manual metrics of Pyramid, Responsiveness, and Readability using the correlation metrics of Pearson r, Spearman ρ, and Kendall τ. The best correlations are specified in bold, and the underlined scores show the top correlations in the TAC AESOP 2011.

the correlation results by the variants of ROUGE-G with Pyramid (Pyr) and Responsiveness (Rsp) metrics measured by Pearson. The best results are observed when $\beta = 0.5$. Performance deteriorates when β approaches 1.0 which indicates the ROUGE scores without any touch of semantic similarity. Decreasing β to zero causes the exclusion of lexical match counts, and consequently inappropriateness of the outcomes. This shows the importance of using both lexical and semantic similarities to fairly judge the quality of summaries.

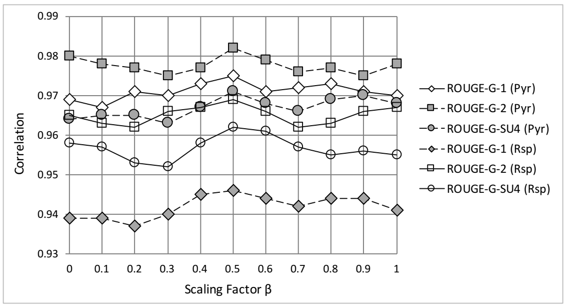

Figure 1: Exploring scaling factor β

It is noteworthy that we have evaluated our approach with the following settings for computing and comparing PPR vectors: (i) *Path-based* with *Rank and Weight* measure (current setting); (ii) *Path-based* with *cosine similarity*; (iii) Excluding *path-based* measure and using *Rank and Weight* measure solely. The results showed that the current setting performs better than the other two.

5 Conclusion

This paper presents ROUGE-G to overcome the limitation of high lexical dependency in ROUGE. Our approach leverages a sense-based representation to calculate PPR vectors for n-grams. The semantic similarity of n-grams are then computed using a formalization of *Path-based* and *Rank and Weight* measures. We finally improve on ROUGE by performing both semantic and lexical analysis of summaries. Experiments over the TAC AESOP datasets demonstrate that ROUGE-G achieves higher correlations with manual judgments in comparison with ROUGE.

In order to demonstrate the effectiveness of ROUGE-G to fairly evaluate abstractive summaries, we need to conduct experiments on a dataset composed of abstractive summaries. However, we evaluated our approach on the TAC 2011 AESOP dataset, which is made of summaries that were generated mostly by extractive systems. Since there is not such dataset at the time of writing this paper, we can continue building on this work by using model summaries, which are abstractive in nature, as a proxy. Thereupon, it is possible to incorporate *jackknifing* procedure in the scoring process in order to see whether our metric can differentiate between peer summaries (naturally extractive) vs. model summaries (naturally abstractive).

Acknowledgements

We thank the anonymous reviewers for their insightful comments and valuable suggestions. The first author was supported by the ”Australian Government Research Training Program Scholarship”.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 238–247.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. 2010. A comparison of features for automatic readability assessment. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters (COLING 2010)*, pages 276–284. ACL.

George Giannakopoulos, Vangelis Karkaletsis, George Vouros, and Panagiotis Stamatopoulos. 2008. Summarization system evaluation revisited: N-gram graphs. *ACM Transactions on Speech and Language Processing (TSLP)*, 5(3):1–5.

Yvette Graham et al. 2015. Re-evaluating automatic summarization with bleu and 192 shades of rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 128–137. ACL.

Taher H Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web (WWW 2002)*, pages 517–526. ACM.

Eduard Hovy, Chin-Yew Lin, Liang Zhou, and Junichi Fukumoto. 2006. Automated summarization evaluation with basic elements. In *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC 2006)*, pages 604–611.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. volume 8. ACL.

Chin-Yew Lin and FJ Och. 2004. Looking for a few good metrics: Rouge and its evaluation. In *Proceedings of the 4th NII Testbeds and Community for Information Access Research: Workshops (NTCIR 2004)*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Annual Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT 2013)*, volume 13, pages 746–751. ACL.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1925–1930. ACL.

Paul Over and J Yen. 2004. An introduction to duc 2004 intrinsic evaluation of generic new text summarization systems, 2004. *National Institute of Standards and Technology (NIST)*.

Karolina Owczarzak and Hoa Trang Dang. 2010. Overview of the tac 2010 summarization track. In *Proceedings of the 3rd Text Analysis Conference (TAC 2010)*.

Karolina Owczarzak and Hoa Trang Dang. 2011. Overview of the tac 2011 summarization track: Guided task and aesop task. In *Proceedings of the 4th Text Analysis Conference (TAC 2011)*.

Rebecca J Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated pyramid scoring of summaries using distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: Short Papers (ACL 2013)*. ACL.

Rebecca J Passonneau, Ani Nenkova, Kathleen McKeown, and Sergey Sigelman. 2005. Applying the pyramid method in duc 2005. In *Proceedings of the 2005 Document Understanding Conference (DUC 2005)*.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1341–1351. ACL.

Mohammad Taher Pilehvar and Roberto Navigli. 2015. From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. *Artificial Intelligence*, 228:95–128.

Sascha Rothe and Hinrich Schütze. 2014. Cosimrank: A flexible & efficient graph-theoretic similarity measure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 1392–1402.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, and Fang Chen. 2017. A semantically motivated approach to compute ROUGE scores. *arXiv preprint arXiv:1710.07441*.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, and Fang Chen. 2018. Summarization evaluation in the absence of human model summaries using the compositionality of word embeddings. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 905–914. ACL.

Josef Steinberger and Karel Ježek. 2012. Evaluation measures for text summarization. *Computing and Informatics*, 28(2):251–275.

Stephen Tratz and Eduard Hovy. 2008. Bewte: basic elements with transformations for evaluation. In *Proceedings of the 1st Text Analysis Conference (TAC 2008)*.

Guided Neural Language Generation for Abstractive Summarization using Abstract Meaning Representation

Hardy
The University of Sheffield
hhardy2@sheffield.ac.uk

Andreas Vlachos
The University of Sheffield
a.vlachos@sheffield.ac.uk

Abstract

Recent work on abstractive summarization has made progress with neural encoder-decoder architectures. However, such models are often challenged due to their lack of explicit semantic modeling of the source document and its summary. In this paper, we extend previous work on abstractive summarization using Abstract Meaning Representation (AMR) with a neural language generation stage which we guide using the source document. We demonstrate that this guidance improves summarization results by 7.4 and 10.5 points in ROUGE-2 using gold standard AMR parses and parses obtained from an off-the-shelf parser respectively. We also find that the summarization performance using the latter is 2 ROUGE-2 points higher than that of a well-established neural encoder-decoder approach trained on a larger dataset. Code is available at `https://github.com/sheffieldnlp/AMR2Text-summ`

1 Introduction

Abstractive summarization is the task of automatically producing the summary of a source document through the process of paraphrasing, aggregating and/or compressing information. Recent work in abstractive summarization has made progress with neural encoder-decoder architectures (See et al., 2017; Chopra et al., 2016; Rush et al., 2015). However, these models are often challenged when they are required to combine semantic information in order to generate a longer summary (Wiseman et al., 2017). To address this shortcoming, several works have explored the use of Abstract Meaning Representation (Banarescu et al., 2013, AMR). These were motivated by AMR's capability to capture the predicate-argument structure which can be utilized in information aggregation during summarization.

However, the use of AMR also has its own shortcomings. While AMR is suitable for information aggregation, it ignores aspects of language such as tense, grammatical number, etc., which are important for the natural language generation (NLG) stage that normally occurs in the end of the summarization process. Due to the lack of such information, approaches for NLG from AMR typically infer it from regularities in the training data (Pourdamghani et al., 2016; Konstas et al., 2017; Song et al., 2016; Flanigan et al., 2016), which however is not suitable in the context of summarization. Consequently, the main previous work on AMR-based abstractive summarization (Liu et al., 2015) only generated bag-of-words from the summary AMR graph.

In this paper, we propose an approach to guide the NLG stage in AMR-based abstractive summarization using information from the source document. Our objective is twofold: (1) to retrieve the information missing from AMR but needed for NLG and (2) improve the quality of the summary. We achieve this in a two-stages process: (1) estimating the probability distribution of the side information, and (2) using it to guide a Luong et al. (2015)'s seq2seq model for NLG.

Our approach is evaluated using the Proxy Report section from the AMR dataset (Knight et al., 2017, LDC2017T10) which contains manually annotated document and summary AMR graphs. Using our proposed guided AMR-to-text NLG, we improve summarization results using both gold standard AMR parses and parses obtained using the RIGA (Barzdins and Gosko, 2016) parser by 7.4 and 10.5 ROUGE-2 points respectively. Our model also outperforms a strong baseline seq2seq model (See et al., 2017) for summarization by 2 ROUGE-2 points.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 768–773
Brussels, Belgium, October 31 - November 4, 2018.

2 Related Work

Abstractive Summarization using AMR: In Liu et al. (2015) work, the source document's sentences were parsed into AMR graphs, which were then combined through merging, collapsing and graph expansion into a single AMR graph representing the source document. Following this, a summary AMR graph was extracted, from which a bag of concept words was obtained without attempting to form fluent text. Vilca and Cabezudo (2017) performed a summary AMR graph extraction augmented with discourse-level information and the PageRank (Page et al., 1998) algorithm. For text generation, Vilca and Cabezudo (2017) used a rule-based syntactic realizer (Gatt and Reiter, 2009) which requires substantial human input to perform adequately.

Seq2seq using Side Information: In Neural Machine Translation (NMT) field, recent work (Zhang et al., 2018) explored modifications to the decoder of seq2seq models to improve translation results. They used a search engine to retrieve sentences and their translation (referred to as translation pieces) that have high similarity with the source sentence. When similar n-grams from a source document were found in the translation pieces, they rewarded the presence of those n-grams during the decoding process through a scoring mechanism calculating the similarity between source sentence and the source side of the translation pieces. Zhang et al. (2018) reported improvements in translation results up to 6 BLEU points over their seq2seq NMT baseline. In this paper we use the same principle and reward n-grams that are found in the source document during the AMR-to-Text generation process. However we use a simpler approach using a probabilistic language model in the scoring mechanism.

3 Guiding NLG for AMR-based summarization

We first briefly describe the AMR-based summarization method of Liu et al. (2015) and then our guided NLG approach.

3.1 AMR-based summarization

In Liu et al. (2015)'s work, each of the sentence of the source document was parsed into an AMR graph, and combined into a source graph, $G = (V, E)$ where $v \in V$ and $e \in E$ are the unique concepts and the relations between pairs of concepts. They then extracted a summary graph, G' using the following sub-graph prediction:

$$G' = \underset{\hat{G}=(\hat{V},\hat{E})}{\arg\max} \sum_{v \in \hat{V}} \boldsymbol{\theta}^{\intercal}\mathbf{f}(v) + \sum_{e \in \hat{E}} \boldsymbol{\psi}^{\intercal}\mathbf{f}(e) \qquad (1)$$

where $\mathbf{f}(v)$ and $\mathbf{f}(e)$ are the feature representations of node v and edge e respectively. The final summary produced was a bag of concept words extracted from G'. This output we will be replacing with our proposed guided NLG.

3.2 Unguided NLG from AMR

Our baseline is a standard (unguided) seq2seq model with attention (Luong et al., 2015) which consists of an encoder and a decoder. The encoder computes the hidden representation of the input, $\{z_1, z_2, \ldots, z_k\}$, which is the linearized summary AMR graph, G' from Liu et al. (2015), following Van Noord and Bos (2017)'s preprocessing steps. Following this, the decoder generates the target words, $\{y_1, y_2, \ldots, y_m\}$, using the conditional probability $P_{s2s}(y_j|y_{<j}, z)$, which is calculated using the equation

$$P_{s2s}(y_j|y_{<j}, z) = \text{softmax}(\mathbf{W}_s\tilde{\mathbf{h}}_t) \qquad (2)$$

, where the attentional hidden state, $\tilde{\mathbf{h}}_t$ is calculated using the equation

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t]) \qquad (3)$$

, where $\mathbf{c}_t$ is the source context vector, and $\mathbf{h}_t$ is the target RNN hidden state. The source context vector is defined as the weighted average over all the source RNN hidden states, $\bar{\mathbf{h}}_s$, given the alignment vector, $\mathbf{a}_t$ where $\mathbf{a}_t$ is defined as

$$\mathbf{a}_t(s) = \frac{\exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s))}{\sum_{s'} \exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_{s'}))} \qquad (4)$$

3.3 Guided NLG from AMR

Our goal is to improve the text generated from the summary AMR graph by the probability distribution of the seq2seq model, P_{s2s} using the source document. Since not all sentences in the source document will be used in generating the summary, we prune the source document to a set of k sentences which have the highest similarity with the summary AMR graph. For graph-to-graph similarity comparison, we use the source document AMR parses and calculate the Longest Common

Subsequence (LCS) between the linearized AMR parses and the summary AMR graph. We keep the top-k sentences sorted by LCS length. To distinguish this pruned document from the source document, we refer to the former as side information.

Our aim is is to combine P_{s2s} with the probability distribution estimated using words in the side information, P_{side}, in order to score each word given its context during decoding. We estimate P_{side} as the linear interpolation of 2-gram to 4-gram probabilities in the form of

$$\begin{aligned} P_{side}(x_j|x_{j-3}^{j-1}) &= \lambda_3 P_{LM}(x_j|x_{j-3}^{j-1}) \\ &+ \lambda_2 P_{LM}(x_j|x_{j-2}^{j-1}) \\ &+ \lambda_1 P_{LM}(x_j|x_{j-1}) \end{aligned} \tag{5}$$

, where x_j is a word occurring in side information document, P_{LM} is an N-gram LM estimated using Maximum Likelihood:

$$P_{LM}(x_j|x_{j-N-1}^{j-1}) = \frac{\text{count}(x_{j-N-1} \ldots x_j)}{\text{count}(x_{j-N-1} \ldots x_{j-1})} \tag{6}$$

and λ_i is defined as

$$\lambda_i = \theta\lambda_{i-1} \text{ where } \theta \in \mathbb{R},\ \lambda_i > 0 \text{ and } \sum_i \lambda_i = 1 \tag{7}$$

where θ is a hyper-parameter that we tune using the dev dataset during the experiments.

Lastly, we combine the probability distribution of the decoder, P_{s2s} with that provided by the side information, P_{side}, as follows:

$$s(y_j|y_{<j}, z) = log\, a + \psi * log(\frac{b}{a} + 1) \tag{8}$$

where ψ is a hyper-parameter determining the influence of the side information on the decoding process, a is $P_{s2s}(y_j|y_{<j}, z)$ and b is $P_{side}(y_j|y_{j-3}^{j-1})$. $s(y_j|y_{<j}, z)$ is used during beam search replacing $P_{s2s}(y_j|y_{<j}, z)$ for all words that occur in the side information. The intuition behind Eq. 8 is that we are rewarding word y_j when it appears in similar context in the side information, i.e. the source document being summarized.

4 Experiments

We conduct experiments in order to answer the following questions about our proposed approach: (1) Is our baseline model comparable with the state-of-the-art AMR-to-text approaches? (2) Does the guidance from the source document improve the result of AMR-to-Text in the context of summarization? (3) Does the improvement in AMR-to-Text hold when we use the generator for abstractive summarization using AMR? We answer each of these in the following paragraphs.

Model	BLEU
Our model (unguided NLG)	21.1
NeuralAMR (Konstas et al., 2017)	22.0
TSP (Song et al., 2016)	22.4
TreeToStr (Flanigan et al., 2016)	23.0

Table 1: Results for AMR-to-text

AMR-to-Text baseline comparison We compare our baseline model (described in §3.2) against previous works in AMR-to-text using the data from the recent SemEval-2016 Task 8 (May, 2016, LDC2015E86). Table 1 reports BLEU scores comparing our model against previous works. Here, we see that our model achieves a BLEU score comparable with the state-of-the-art, and thus we argue that it is sufficient to be used in our subsequent experiments with guidance.

Guided NLG for AMR-to-Text In this experiment we apply our guided NLG mechanism described in §3.3 to our baseline seq2seq model. To isolate the effects of guidance we skip the actual summarization process and proceed to directly generating the summary text from the gold standard summary AMR graphs from the Proxy Report section. To determine the hyper-parameters, we perform a grid search using the dev dataset, where we found the best combination of ψ, θ and k are 0.95, 2.5 and 15 respectively. We have two different settings for this experiment: the oracle and non-oracle settings. In the oracle setting, we directly use the gold standard summary text as the guidance for our model. The intuition is that in this setting, our model knows precisely which words should appear in the summary text, thus providing an upper bound for the performance of our guided NLG approach. In the non-oracle setting, we use the mechanism described in §3.3. We also compare them against the baseline (unguided) model from §3.2. Table 2 reports performance for all models. The difference between the guided and the unguided model is 16.2 points in BLEU and 9.9 points in ROUGE-2, while there is room for improvement as evidenced by the difference between the oracle and non-oracle result.

Guided NLG for full summarization In this experiment we combine our guided NLG model

Model	BLEU	F_1 ROUGE		
		R-1	R-2	R-L
Guided NLG (Oracle)	61.3	79.4	63.7	76.4
Guided NLG	45.8	70.7	49.5	64.9
Unguided NLG	29.6	68.6	39.6	61.3

Table 2: BLEU and ROUGE results for guided and unguided models using test dataset.

with Liu et al. (2015)'s work in order to generate fluent texts from their summary AMR graphs using the hyper-parameters tuned in the previous paragraph. Liu et al. (2015) used parses from both the manual annotation of the Proxy dataset as well as those obtained using the JAMR parser (Flanigan et al., 2014). Instead of JAMR we use the RIGA parser (Barzdins and Gosko, 2016) which had the highest accuracy in the SemEval 2016 Task 8 (May, 2016). We compare our result against Liu et al. (2015)'s bag of words[1], the unguided AMR-to-text model from §3.2, and a seq2seq summarization model (OpenNMT BRNN)[2,3] which summarizes directly from the source document to summary sentence without using AMR as an interlingua and is trained on CNN/DM corpus (Hermann et al., 2015) using the same settings as See et al. (2017).

AMR parses	NLG Model	F_1 ROUGE		
		R-1	R-2	R-L
Gold	Guided	40.4	20.3	31.4
	Unguided	38.9	12.9	27.0
	Liu et al. (2015)	39.6	6.2	22.1
RIGA	Guided	42.3	21.2	33.6
	Unguided	37.8	10.7	26.9
	Liu et al. (2015)	40.9	5.5	21.4
Directly from Text	OpenNMT BRNN 2 layer, emb 256, hidden 1024	36.1	19.2	31.1

Table 3: The F_1 ROUGE scores for guided, unguided, Liu et al. (2015) (BoW) results in Gold and RIGA parses, and seq2seq summarization. All models are run using test dataset.

In Table 3, we can see that our approach results in improvements over both the unguided AMR-to-text and the standard seq2seq summarization. One interesting note is that using the RIGA parses result in higher ROUGE scores than the gold parses for the guided model in our experiment. This phenomenon was also observed in Liu et al. (2015)'s experiment where the summary graphs extracted from automatic parses had higher accuracy than those extracted from manual parses. We hypothesize this can be attributed to how the AMR dataset is annotated as there might be discrepancies in different annotator's choices of AMR concepts and relations for sentences with similar wording. In contrast, the AMR parsers introduce errors, but they are consistent in their choices of AMR concepts and relations. The discrepancies in the manual annotation could have impacted the performance of the AMR summarizer that we use more negatively than the noise introduced due to the AMR parsing errors.

NLG Model	Generated Summary
Gold	on 8 august 2008 russia conducted **airstrikes** on **georgian** targets .
Guided	on 8 august 2008 russia conducted **airstrikes** on **georgian** separatist targets .
Unguided	on 8 august 2008 russia conducted **a softening** of the **georgia 's** separatist target .
Seq2seq	the russian laboratory complex is a 90 - building campus and served as the location for russia 's secret biological weapons program in the soviet era of a moscow regional depository threaten moscow .

Table 4: Result summaries of guided, unguided and seq2seq models compared with gold summary.

In Table 4, we show sample summaries from the different models, where we can see that our guided model improves the unguided model by correcting a wrong word (*a softening*) into a correct one (*airstrikes*) and introducing a better suited word from the source document (*georgian* instead of *georgia 's*).

NLG Model	Fluency
Guided	2.66
Unguided	2.16

Table 5: Fluency scores on test dataset.

We also evaluated manually by asking human evaluators to judge sentences' fluency (grammatical and naturalness) on a scale of 1 (worst) to 6 (best) for the guided and unguided model (see Ta-

---

[1] We were able to obtain comparable AMR summarization subgraph prediction to their reported results using their published software but not to match their bag-of-word generation results.

[2] We use the OpenNMT-pytorch implementation `https://github.com/OpenNMT/OpenNMT-py` and a pre-trained model downloaded from `http://opennmt.net/OpenNMT-py/Summarization.html` which has higher result than See et al. (2017)'s summarizer.

[3] The pre-trained model generates multiple sentences summary, but we use only the first sentence summary for evaluation in accordance with the AMR dataset.

ble 5). While the manual evaluation shows improvement over the unguided model, on the other hand, grammatical mistakes and redundant repetition in the generated text are still major problems (see Table 6) in our AMR generation.

Guided NLG Model	Problems
the soldiers were injured when **a attempt** to defuse the bombs .	grammatical mistake
on 20 october 2002 the state - run radio nepal reported **on 20 october 2002** that at the evening **- run radio nepal reported on 20 october 2002** that the guerrillas were killed **and killed** .	redundant repetition

Table 6: Problems in guided model's summaries.

5 Conclusion and Future Works

In this paper we proposed a guided NLG approach that substantially improves the output of AMR-based summarization. Our approach uses a simple guiding process based on a probabilistic language model. In future work we aim to improve summarization performance by jointly training the guiding process with the AMR-based summarization process.

Acknowledgments

We are thankful for Gerasimos Lampouras for his help with the manual evaluation process and all volunteers who participated in it. We would also like to thank the Indonesian government that has sponsored the first author's studies through the Indonesia Endowment Fund for Education (LPDP). The second author is supported by the EU H2020 SUMMA project (grant agreement number 688139) and the EPSRC grant eNeMILP (EP/R021643/1).

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Guntis Barzdins and Didzis Gosko. 2016. RIGA at SemEval-2016 Task 8: Impact of Smatch Extensions and Character-Level Neural Translation on AMR Parsing Accuracy. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1143–1147.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *Proceedings of the 15th Annual Conference of the NAACL HLT*, pages 93–98.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. Generation from Abstract Meaning Representation using Tree Transducers. In *Proceedings of the 2016 Conference of the NAACL*, pages 731–739.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. *Proceedings of the 52th Annual Meeting of the ACL*, pages 1426–1436.

Albert Gatt and Ehud Reiter. 2009. SimpleNLG : A realisation engine for practical applications. *Proceedings of the 12th European Workshop on NLG*, (March):90–93.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Neural Information Processing Systems*, pages 1–14.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. 2017. Abstract Meaning Representation (AMR) Annotation Release 2.0 LDC2017T10.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *Proceedings of the 2015 Conference of the NAACL HLT*, pages 1077 – 1086.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward Abstractive Summarization Using Semantic Representations. In *Proceedings of the 2015 Conference of the NAACL HLT*, pages 1077–1086.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. The Association for Computational Linguistics.

Jonathan May. 2016. SemEval-2016 Task 8: Meaning Representation Parsing. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1063–1073.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. *World Wide Web Internet And Web Information Systems*, 54(1999-66):1–17.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from Abstract Meaning Representations. In *Proceedings of the 9th International Natural Language Generation*, volume 0, pages 21–25.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. *In Proceedings of the Conference on EMNLP*, (September):379–389.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the ACL*.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. AMR-to-text generation as a Traveling Salesman Problem. In *Proceedings of the 2016 Conference on EMNLP*, pages 2084–2089.

Rik Van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7(2016):93–108.

Gregory Cesar Valderrama Vilca and Marco Antonio Sobrevilla Cabezudo. 2017. A Study of Abstractive Summarization Using Semantic Representations and Discourse Level Information. In *International Conference on Text, Speech, and Dialogue*, pages 482–490. Springer.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in Data-to-Document Generation. In *Proceedings of the 2017 Conference on EMNLP*, pages 2253–2263, Copenhagen, Denmark.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding Neural Machine Translation with Retrieved Translation Pieces. In *Proceedings of the 16th Annual Conference of the NAACL HLT*, New Orleans, Louisiana. The Association for Computational Linguistics.

Evaluating *Multiple* System Summary Lengths: A Case Study

Ori Shapira[1], **David Gabay**[1],
Hadar Ronen[2], **Judit Bar-Ilan**[2], **Yael Amsterdamer**[1],
Ani Nenkova[3], **and Ido Dagan**[1]

[1]Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel
[2]Information Science Department, Bar-Ilan University, Ramat-Gan, Israel
[3]University of Pennsylvania, Philadelphia, PA
`{obspp18, dawid.gabay, hadarg, juditb}@gmail.com`
`amstery@cs.biu.ac.il, nenkova@seas.upenn.edu, dagan@cs.biu.ac.il`

Abstract

Practical summarization systems are expected to produce summaries of varying lengths, per user needs. While a couple of early summarization benchmarks tested systems across multiple summary lengths, this practice was mostly abandoned due to the assumed cost of producing reference summaries of multiple lengths. In this paper, we raise the research question of whether *reference summaries of a single length* can be used to reliably evaluate *system summaries of multiple lengths*. For that, we have analyzed a couple of datasets as a case study, using several variants of the ROUGE metric that are standard in summarization evaluation. Our findings indicate that the evaluation protocol in question is *indeed competitive*. This result paves the way to practically evaluating varying-length summaries with simple, possibly existing, summarization benchmarks.

1 Introduction

Automated summarization systems typically produce a text that mimics a manual summary. In these systems, an important aspect is the output summary length, which may vary according to user needs. Consequently, output length has been a common tunable parameter in pre-neural summarization systems and has been incorporated recently in few neural models as well (Kikuchi et al., 2016; Fan et al., 2017; Ficler and Goldberg, 2017).

It was originally assumed that summarization systems should be assessed across multiple summary lengths. For that, the earliest Document Understand Conference (DUC) (NIST, 2011) benchmarks, in 2001 and 2002, defined several target summary lengths and evaluated each summary against (manually written) reference summaries of the same length.

However, due to the high cost incurred, subsequent DUC and TAC (NIST, 2018) benchmarks (2003-2014), as well as the more recently popular datasets CNN/Daily Mail (Nallapati et al., 2016) and Gigaword (Graff et al., 2003), included references and evaluation for just one summary length per input text. Accordingly, systems were asked to produce a *single* summary, of corresponding length. This decision was partly supported by an observation that system rankings tended to correlate across different summary lengths (Over et al., 2007), even though, as we show in Section 2, this correlation is limited.

In this paper, we propose that the summarization community should consider resuming evaluating summarization systems over *multiple* length outputs, as it would allow better assessment of length-related performance within and across systems (illustrated in Section 3). To avoid the need in multiple-length reference summaries we raise the following research question: *can reference summaries of a single length be used to evaluate system summaries of multiple lengths, as reliably as when using references of multiple lengths, with respect to different standard evaluation metrics?* Recently, Kikuchi et al. (2016) evaluated system summaries of three different lengths against reference summaries of a single length. Yet, their evaluation methodology was not assessed through correlation to human judgment, as has been commonly done for other automatic evaluation protocols. Here, we provide a closer look into this methodology, given its potential value.

As a first accessible case study, we test our research question over the DUC 2001 and 2002 data (Section 2). To the best of our knowledge, these are the only two datasets that include multiple length reference and submitted system summaries, as well as manual assessment of the latter. Our analysis reveals that, for this data and with respect to various highly utilized automatic ROUGE metrics, the answer to our question is affirmative, in

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 774–778
Brussels, Belgium, October 31 - November 4, 2018.

	# refs	ref lengths (# words)	# clusters	# systems
2001	3	50, 100, 200, 400	30	14
2002	2	10, 50, 100, 200	59	10

Table 1: DUC 2001 and 2002. Number of reference summaries per length for each text cluster, reference lengths, number of clusters and number of evaluated systems.

terms of correlation with human judgment.

Our promising results suggest repeating the assessment methodology presented here in future work, to test our question over more recent and broader summarization datasets and human evaluation schemes. This, in turn, would allow the community to feasibly resume proper evaluation and deliberate development of systems that target effective summaries across a range of lengths.

2 Case Study Analysis

Here, we first examine the relevance of our proposal to reinstitute summarization evaluation over multiple summary lengths. Then, we investigate our research question of whether using reference summaries of a single length suffices for evaluating system summaries of multiple lengths. We turn to the DUC 2001 and 2002 multi-document summarization datasets, which, to the best of our knowledge, are the only available datasets that provide the necessary requirements for this analysis (see Table 1).

The importance of evaluating and comparing systems at several lengths is demonstrated with the observation that system rankings can change quite significantly at different summary lengths. In 2001, the Spearman correlation between the available human rankings of systems at the 50-word and 400-word lengths is 0.61. For example, the system ranked first at length 50 ranks sixth at lengths 200 and 400. Even for the human system ranking at the 100-word length, which deviates the least from human rankings at the other lengths, the correlation with system ranking at the 400 length is only 0.73. Generally, the larger the difference between a pair of summary lengths, the greater the fluctuation in system rankings. Similar trends were observed for DUC 2002, and when comparing system rankings by automatic ROUGE scoring (both rankings are elaborated below). Obviously, such performance differences are overlooked when evaluating systems over summaries of a single length.

Next, we turn to investigate our research question. In this paper, we examine it with respect to automatic summary evaluation, which has become most common for system development and evaluation, thanks to its speed and low cost. Specifically, we use several variants of the ROUGE metric (Lin, 2004), which is almost exclusively utilized as an automatic evaluation metric class for summarization. ROUGE variants are based on word sequence overlap between a system summary and a reference summary, where each variant measures a different aspect of text comparison. Despite its pitfalls, ROUGE has shown reasonable correlation of its system scores to those obtained by manual evaluation methods (Lin, 2004; Over and James, 2004; Over et al., 2007; Nenkova et al., 2007; Louis and Nenkova, 2013; Peyrard et al., 2017), such as SEE (Lin, 2001), responsiveness (NIST, 2006) and Pyramid (Nenkova et al., 2007).

We follow the same methodology of assessing the reliability of automatic evaluation scores by measuring their correlation to human evaluation scores. In our case, DUC 2001 and 2002 applied the SEE manual evaluation method. NIST assessors compared systems' summaries to reference summaries, which were all decomposed into a list of elementary discourse units (EDUs). Each reference EDU was marked throughout the system EDUs and was scored for how well it was expressed. The final manually evaluated scores, called the *human mean content coverage scores*, are provided in the DUC datasets. We can then correlate the human-based system ranking, attained from these provided scores, to the system ranking attained from the automatic scores that we calculate using our proposed methodology.

As a baseline, we consider the ROUGE Recall scores obtained by the standard reference summary configuration (Standard, first row in Table 2), that is, when system summaries of each length (table columns) are evaluated against reference summaries of the same length. This is the same configuration used by Lin (2004) when introducing and assessing ROUGE. Then, looking into our research question, we consider reference summary configurations in which system summaries of all lengths are evaluated against reference summaries of a single chosen length (OnlyNNN, subsequent rows of Table 2). In each configuration (each row), we repeat the evaluation twice: once using the complete set of available reference sum-

	System Summary Length									
	50		**100**		**200**		**400**		**Avg. across lengths**	
Reference Set	3refs	1ref	3refs	1ref	3refs	1ref	3refs	1ref	3refs	1ref
Standard	0.72	0.65	0.88	0.85	0.9	0.86	0.95	0.94	0.86	0.83
Only50	0	0	+0.02	0	+0.01	+0.04	+0.01	+0.02	**+0.010**	**+0.015**
Only100	-0.01	+0.04	0	0	+0.01	-0.01	+0.02	0	+0.005	+0.008
Only200	-0.09	-0.09	-0.06	-0.08	0	0	+0.01	-0.01	-0.035	-0.0045
Only400	-0.06	+0.02	-0.09	-0.09	-0.01	+0.03	0	0	-0.040	-0.010

Table 2: Pearson correlations between ROUGE-1 and human scores over DUC 2001 for different system summary lengths (column pairs) and different reference summary configurations (rows), when using one reference or three. The first baseline row presents absolute correlations while successive rows show relative differences to the baseline.

	2001						**2002**					
	R-1		**R-2**		**R-L**		**R-1**		**R-2**		**R-L**	
Reference Set	3refs	1ref	3refs	1ref	3refs	1ref	2refs	1ref	2refs	1ref	2refs	1ref
Standard	0.86	0.83	0.79	0.77	0.87	0.83	0.78	0.75	0.86	0.82	0.82	0.77
Only10	-	-	-	-	-	-	0	-0.015	-0.100	-0.178	-0.003	-0.045
Only50	**+0.010**	**+0.015**	-0.013	-0.038	**+0.008**	+0.010	**+0.035**	**+0.053**	-0.050	-0.038	**+0.020**	**+0.080**
Only100	+0.005	+0.008	**-0.010**	**-0.003**	+0.005	**+0.013**	+0.023	+0.048	**-0.035**	**0**	-0.008	+0.040
Only200	-0.035	-0.045	-0.055	-0.053	-0.033	-0.04	+0.013	+0.023	-0.068	-0.025	-0.028	+0.005
Only400	-0.040	-0.010	-0.075	-0.075	-0.038	0	-	-	-	-	-	-

Table 3: Averaged correlations (across system summary lengths, equivalent to the rightmost columns in Table 2) for different ROUGE variants (column pairs) and reference summary configurations (rows), when using 1 reference or multiple. The first row presents absolute correlations, with relative differences in subsequent rows.

maries of the utilized reference length, and once with just one randomly chosen reference summary from that set (the 3refs and 1ref sub-columns).

For each reference summary configuration, we compute ROUGE Recall system scores[1] for the three common ROUGE variants R-1, R-2 and R-L, which compare unigrams, bigrams and the longest common subsequence, respectively. System scores, per summary length, are obtained by averaging across all summarized texts. We then calculate their Pearson correlation[2] with the available human mean content coverage scores for the systems. The first row of Table 2 shows these correlations, considering the R-1 scores for the DUC 2001 systems, per summary length. The subsequent rows show the corresponding figures for the single-reference-length configurations. For readability, we present in these rows the relative *differences* to the Standard baseline row. Hence, positive values indicate a configuration that is at least as good as the standard configuration.

Table 3 presents correlations averaged over all summary lengths, for the three ROUGE variants over both datasets. We see in the tables that evaluating system summaries of all lengths against references of a single length often performs on par with the standard configuration. In particular, the single fixed set of 50-word reference summaries performs overall as well as the standard approach, and, although not substantially, is the most effective configuration within the data analyzed. In other words, in this dataset, the 50-word reference summaries provide a "test sample" for evaluating the longer system summaries, which is as effective as the same length references used by the standard method.

We note that even when a single reference summary is available, reasonable correlations with human scores are obtained for the 50 word reference. This suggests that it may be possible to compare system summaries of multiple lengths even against a single reference summary, of a relatively short length. This observation seems to deserve further assessment over recent large scale datasets, such as CNN/DailyMail, which provide a single relatively short reference for each summarized text.

In addition to correlation to human assessment, we computed the correlations between system rankings calculated by Standard and those calcu-

---

[1] Omitting stop words.

[2] Following Lin (2004). Spearman ranking correlations provide similar results.

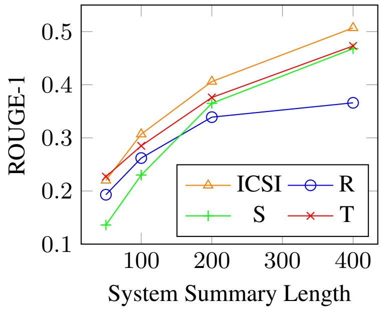

Figure 1: R-1 scores of a few systems, evaluated against the 50-word reference set of DUC 01. Systems R, S and T are from DUC 01; ICSISumm is a later competitive system (Gillick et al., 2008).

lated by Only50, at each system summary length. We find very high correlations (above 0.95 for all system summary lengths, in both datasets) when using multiple references and slightly lower (0.85 to 0.9) with one reference summary. These figures show that the Only50 configuration ranks systems very similarly to Standard.

To further verify our results, we computed correlations in two additional settings. First, we conducted the same analysis, excluding 2-3 of the worst systems, which might artificially boost the correlation (Rankel et al., 2013). Second, we computed score differences between all pairs of systems, for both human and ROUGE scores, and computed the correlation between these two sets of differences (Rankel et al., 2011). In both cases we observed rather consistent results, assessing that a single set of short reference summaries evaluates system summaries of different lengths just as well as the standard configuration.

3 Cross-length Summary Evaluation

This section illustrates how system performances can be measured and compared when evaluating them on outputs of varying lengths against a single reference point. Figure 1 presents the ROUGE scores of the Only50 configuration for three DUC-01 submitted systems, and for ICSISumm (Gillick et al., 2008), a later competitive system.

As expected when measuring ROUGE Recall against a fixed reference length, longer system summaries typically cover more of the reference summaries content than shorter ones, yielding higher scores. Yet, it can be noted, for example, that the value of the 400-word summary of system R in the figure is lower than that of the 200-word summaries of the other systems. Such a comparison is impossible in the standard setup, as each system length is evaluated against different reference summaries. We note that similar comparisons are embedded in the evaluations of Steinberger and Jezek (2004) and Kikuchi et al. (2016), who also evaluated multiple summary lengths.

Further, one can define the marginal value of longer summaries of a given system as the ROUGE score increase per number of additional words, namely the graph slope. This denotation allows measuring the effectiveness of producing longer summaries. For example, deploying system R, we might decide to output only summaries no longer than 200 words, since the marginal value of longer summaries becomes too small. The other systems, on the other hand, seem marginally effective also in 400 word summaries.

4 Discussion

We proposed the potential value of evaluating summarization systems at different summary lengths. Such evaluations would allow proper evaluation of systems' "length knob", tracking how their ranking changes across summary lengths as well as tracking the cross-length behavior of individual systems. Given that reference summaries of a single length are usually available in practice, we analyzed the potential use of *reference summaries of a single length* for evaluating *system summaries of multiple lengths*. We found, on the only two datasets readily available for such analysis, that this configuration is as reliable as the standard configuration, which evaluates each system summary against a reference of a matching length.

To broadly substantiate our findings, we propose future work that would follow our assessment methodology over test samples from current datasets (e.g. CNN/DailyMail), judging performance of current systems and utilizing current manual evaluation protocols. This would require preparing, for limited samples, additional manually crafted summaries of several lengths and manually evaluating system summaries of corresponding lengths. Using such data, it will be possible to repeat our analysis and test the broader validity of the single-reference-length configuration. If broadly assessed, it will be possible to start evaluating system summaries of multiple lengths over most currently available datasets, leveraging the available single-length reference summaries. Future benchmarks could require systems to produce different length outputs, while feasibly evaluating them using the existing, single length, reference summaries. This, in turn, is likely to drive research to better address the need for producing high quality summaries flexibly across a range of summary lengths, a dimension that has been disregarded for long.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. We thank Yinfei Yang for his assistance in producing the ICSISumm summaries that we utilized in our analysis. This work was supported in part by grants from the MAGNET program of the Israel Innovation Authority; by the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1); by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Ministers Office; and by the Israel Science Foundation (grants 1157/16 and 1951/17).

References

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *CoRR*, abs/1711.05217.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104. Association for Computational Linguistics.

Daniel Gillick, Benoit Favre, and Dilek Hakkani-Tür. 2008. The ICSI Summarization System at TAC 2008. In *TAC*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4:1.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338. Association for Computational Linguistics.

Chin-Yew Lin. 2001. Summary evaluation environment user guide. `http://www1.cs.columbia.edu/nlp/tides/SEEManual.pdf`.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Annie Louis and Ani Nenkova. 2013. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2):267–300.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(2):4.

NIST. 2006. Responsiveness assessment instructions. `www-nlpir.nist.gov/projects/duc/duc2006/responsiveness.assessment.instructions`.

NIST. 2011. Document Understanding Conferences. `https://duc.nist.gov/`.

NIST. 2018. Text Analysis Conferences. `https://tac.nist.gov/`.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in context. *Information Processing & Management*, 43(6):1506–1520.

Paul Over and Yen James. 2004. An Introduction to DUC 2004 Intrisic Evaluation of Generic News Text Summarization Systems. `duc.nist.gov/pubs/2004slides/duc2004.intro.pdf`.

Maxime Peyrard, Teresa Botschen, and Iryna Gurevych. 2017. Learning to score system summaries for better content selection evaluation. In *Proceedings of the EMNLP 2017 Workshop on New Frontiers in Summarization*, pages 74–84.

Peter Rankel, John M Conroy, Eric V Slud, and Dianne P O'Leary. 2011. Ranking human and machine summarization systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 467–473. Association for Computational Linguistics.

Peter A Rankel, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2013. A decade of automatic content evaluation of news summaries: Reassessing the state of the art. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 131–136.

Josef Steinberger and Karel Jezek. 2004. Using latent semantic analysis in text summarization and summary evaluation. *Proc. ISIM*, 4:93–100.

Neural Latent Extractive Document Summarization

Xingxing Zhang[†], **Mirella Lapata**[‡], **Furu Wei**[†] and **Ming Zhou**[†]
[†]Microsoft Research Asia, Beijing, China
[‡]Institute for Language, Cognition and Computation,
School of Informatics, University of Edinburgh, UK
{xizhang,fuwei,mingzhou}@microsoft.com,mlap@inf.ed.ac.uk

Abstract

Extractive summarization models require sentence-level labels, which are usually created heuristically (e.g., with rule-based methods) given that most summarization datasets only have document-summary pairs. Since these labels might be suboptimal, we propose a latent variable extractive model where sentences are viewed as latent variables and sentences with activated variables are used to infer gold summaries. During training the loss comes *directly* from gold summaries. Experiments on the CNN/Dailymail dataset show that our model improves over a strong extractive baseline trained on heuristically approximated labels and also performs competitively to several recent models.

1 Introduction

Document summarization aims to automatically rewrite a document into a shorter version while retaining its most important content. Of the many summarization paradigms that have been identified over the years (see Mani 2001 and Nenkova and McKeown 2011 for comprehensive overviews), two have consistently attracted attention: *extractive* approaches generate summaries by copying parts of the source document (usually whole sentences), while *abstractive* methods may generate new words or phrases which are not in the document.

A great deal of previous work has focused on extractive summarization which is usually modeled as a sentence ranking or binary classification problem (i.e., sentences which are top ranked or predicted as `True` are selected as summaries). Early attempts mostly leverage human-engineered features (Filatova and Hatzivassiloglou, 2004) coupled with binary classifiers (Kupiec et al., 1995), hidden Markov models (Conroy and O'leary, 2001), graph based methods (Mihalcea, 2005), and integer linear programming (Woodsend and Lapata, 2010).

The successful application of neural network models to a variety of NLP tasks and the availability of large scale summarization datasets (Hermann et al., 2015; Nallapati et al., 2016) has provided strong impetus to develop data-driven approaches which take advantage of continuous-space representations. Cheng and Lapata (2016) propose a hierarchical long short-term memory network (LSTM; Hochreiter and Schmidhuber 1997) to learn context dependent sentence representations for a document and then use yet another LSTM decoder to predict a binary label for each sentence. Nallapati et al. (2017) adopt a similar approach, they differ in their neural architecture for sentence encoding and the features used during label prediction, while Narayan et al. (2018) equip the same architecture with a training algorithm based on reinforcement learning. Abstractive models (Nallapati et al., 2016; See et al., 2017; Paulus et al., 2017) are based on sequence-to-sequence learning (Sutskever et al., 2014; Bahdanau et al., 2015), however, most of them underperform or are on par with the baseline of simply selecting the leading sentences in the document as summaries (but see Paulus et al. 2017 and Celikyilmaz et al. 2018 for exceptions).

Although seemingly more successful than their abstractive counterparts, extractive models require sentence-level labels, which are not included in most summarization datasets (only document and gold summary pairs are available). Sentence labels are usually obtained by rule-based methods (Cheng and Lapata, 2016) or by maximizing the ROUGE score (Lin, 2004) between a subset of sentences and the human written summaries (Nallapati et al., 2017). These methods do not fully exploit the human summaries, they only create `True`/`False` labels which might be suboptimal.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 779–784
Brussels, Belgium, October 31 - November 4, 2018.

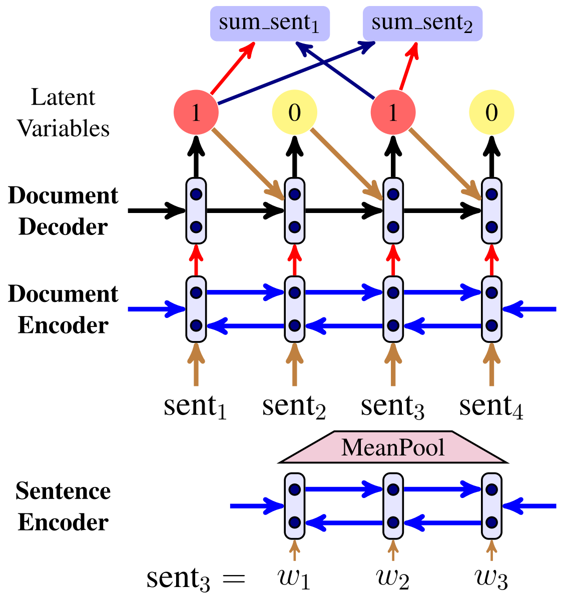

Figure 1: Latent variable extractive summarization model. sent$_i$ is a sentence in a document and sum_sent$_i$ is a sentence in a gold summary of the document.

In this paper we propose a latent variable extractive model and view labels of sentences in a document as binary latent variables (i.e., zeros and ones). Instead of maximizing the likelihood of "gold" standard labels, the latent model directly maximizes the likelihood of human summaries given selected sentences. Experiments on the CNN/Dailymail dataset (Hermann et al., 2015) show that our latent extractive model improves upon a strong extractive baseline trained on rule-based labels and also performs competitively to several recent models.

2 Model

We first introduce the neural extractive summarization model upon which our latent model is based on. We then describe a sentence compression model which is used in our latent model and finally move on to present the latent model itself.

2.1 Neural Extractive Summarization

In extractive summarization, a subset of sentences in a document is selected as its summary. We model this problem as an instance of sequence labeling. Specifically, a document is viewed as a sequence of sentences and the model is expected to predict a `True` or `False` label for each sentence, where `True` indicates that the sentence should be included in the summary. It is assumed that during training sentences and their labels in each document are given (methods for obtaining these labels are discussed in Section 3).

As shown in the lower part of Figure 1, our extractive model has three parts: a *sentence encoder* to convert each sentence into a vector, a *document encoder* to learn sentence representations given surrounding sentences as context, and a *document decoder* to predict sentence labels based on representations learned by the document encoder. Let $\mathcal{D} = (S_1, S_2, \dots, S_{|\mathcal{D}|})$ denote a document and $S_i = (w_1^i, w_2^i, \dots, w_{|S_i|}^i)$ a sentence in $\mathcal{D}$ (where w_j^i is a word in S_i). Let $Y = (y_1, \dots, y_{|\mathcal{D}|})$ denote sentence labels. The sentence encoder first transforms S_i into a list of hidden states $(\mathbf{h}_1^i, \mathbf{h}_2^i, \dots, \mathbf{h}_{|S_i|}^i)$ using a Bidirectional Long Short-Term Memory Network (Bi-LSTM; Hochreiter and Schmidhuber 1997; Schuster and Paliwal 1997). Then, the sentence encoder yields $\mathbf{v}_i$, the representation of S_i, by averaging these hidden states (also see Figure 1):

$$\mathbf{v}_i = \frac{1}{|S_i|} \sum_j \mathbf{h}_j^i \tag{1}$$

In analogy to the sentence encoder, the document encoder is another Bi-LSTM but applies on the sentence level. After running the Bi-LSTM on a sequence of sentence representations $(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{|\mathcal{D}|})$, we obtain context dependent sentence representations $(\mathbf{h}_1^E, \mathbf{h}_2^E, \dots, \mathbf{h}_{|\mathcal{D}|}^E)$.

The document decoder is also an LSTM which predicts sentence labels. At each time step, it takes the context dependent sentence representation of S_i produced by the document encoder as well as the prediction in the previous time step:

$$\mathbf{h}_i^D = \text{LSTM}(\mathbf{h}_{i-1}^D, \begin{bmatrix} \mathbf{W}_e \; e(y_{i-1}) \\ \mathbf{h}_i^E \end{bmatrix}) \tag{2}$$

where $\mathbf{W}_e \in \mathbb{R}^{d \times 2}$ is the label embedding matrix (d is the hidden dimension for the document decoder LSTM) and y_{i-1} is the prediction at time step $i-1$; the predicted label distribution for y_i is:

$$p(y_i | y_{1:i-1}, \mathbf{h}_{i-1}^D) = \text{softmax}(\mathbf{W}_o \, \mathbf{h}_i^D) \tag{3}$$

where $\mathbf{W}_o \in \mathbb{R}^{2 \times d}$.

The model described above is usually trained by minimizing the negative log-likelihood of sentence labels in training documents; it is almost identical to Cheng and Lapata (2016) except that we use a word-level long short-term memory network coupled with mean pooling to learn sentence representations, while they use convolutional neural network coupled with max pooling (Kim et al., 2016).

2.2 Sentence Compression

We train a sentence compression model to map a sentence selected by the extractive model to a sentence in the summary. The model can be used to evaluate the quality of a selected sentence with respect to the summary (i.e., the degree to which it is similar) or rewrite an extracted sentence according to the style of the summary.

For our compression model we adopt a standard attention-based sequence-to-sequence architecture (Bahdanau et al., 2015; Rush et al., 2015). The training set for this model is generated from the same summarization dataset used to train the exractive model. Let $\mathcal{D} = (S_1, S_2, \dots, S_{|\mathcal{D}|})$ denote a document and $\mathcal{H} = (H_1, H_2, \dots, H_{|\mathcal{H}|})$ its summary. We view each sentence H_i in the summary as a target sentence and assume that its corresponding source is a sentence in $\mathcal{D}$ most similar to it. We measure the similarity between source sentences and candidate targets using ROUGE, i.e., $S_j = \text{argmax}_{S_j} \text{ROUGE}(S_j, H_i)$ and $\langle S_j, H_i \rangle$ is a training instance for the compression model. The probability of a sentence $\hat{H}_i$ being the compression of $\hat{S}_j$ (i.e., $p_{s2s}(\hat{H}_i|\hat{S}_j)$) can be estimated with a trained compression model.

2.3 Latent Extractive Summarization

Training the extractive model described in Section 2.1 requires sentence-level labels which are obtained heuristically (Cheng and Lapata, 2016; Nallapati et al., 2017). Our latent variable model views sentences in a document as binary variables (i.e., zeros and ones) and uses sentences with activated latent variables (i.e., ones) to infer gold summaries. The latent variables are predicted with an extractive model and the loss during training comes from gold summaries *directly*.

Let $\mathcal{D} = (S_1, S_2, \dots, S_{|\mathcal{D}|})$ denote a document and $\mathcal{H} = (H_1, H_2, \dots, H_{|\mathcal{H}|})$ its human summary (H_k is a sentence in $\mathcal{H}$). We assume that there is a latent variable $z_i \in \{0, 1\}$ for each sentence S_i indicating whether S_i should be selected, and $z_i = 1$ entails it should. We use the extractive model from Section 2.1 to produce probability distributions for latent variables (see Equation (3)) and obtain them by sampling $z_i \sim p(z_i|z_{1:i-1}, \mathbf{h}^D_{i-1})$ (see Figure 1). $\mathcal{C} = \{S_i|z_i = 1\}$, the set of sentences whose latent variables equal to one, are our current extractive summaries. Without loss of generality, we denote $\mathcal{C} = (C_1, \dots, C_{|\mathcal{C}|})$. Then, we estimate how likely it is to infer the human summary $\mathcal{H}$ from $\mathcal{C}$. We estimate the likelihood of summary sentence H_l given document sentence C_k with the compression model introduced in Section 2.2 and calculate the normalized[1] probability s_{kl}:

$$s_{kl} = \exp\left(\frac{1}{|H_l|} \log p_{s2s}(H_l|C_k)\right) \quad (4)$$

The score R_p measures the extent to which $\mathcal{H}$ can be inferred from $\mathcal{C}$:

$$R_p(\mathcal{C}, \mathcal{H}) = \frac{1}{|\mathcal{C}|} \sum_{k=1}^{|\mathcal{C}|} \max_{l=1}^{|\mathcal{H}|} s_{kl} \quad (5)$$

For simplicity, we assume one document sentence can only find one summary sentence to explain it. Therefore, for all H_l, we only retain the most evident s_{kl}. $R_p(\mathcal{C}, \mathcal{H})$ can be viewed as the "precision" of document sentences with regard to summary sentences. Analogously, we also define R_r, which indicates the extent to which $\mathcal{H}$ can be covered by $\mathcal{C}$:

$$R_r(\mathcal{C}, \mathcal{H}) = \frac{1}{|\mathcal{H}|} \sum_{l=1}^{|\mathcal{H}|} \max_{k=1}^{|\mathcal{C}|} s_{kl} \quad (6)$$

$R_r(\mathcal{C}, \mathcal{H})$ can be viewed as the "recall" of document sentences with regard to summary sentences. The final score $R(\mathcal{C}, \mathcal{H})$ is the weighted sum of the two:

$$R(\mathcal{C}, \mathcal{H}) = \alpha\, R_p(\mathcal{C}, \mathcal{H}) + (1 - \alpha)\, R_r(\mathcal{C}, \mathcal{H}) \quad (7)$$

Our use of the terms "precision" and "recall" is reminiscent of relevance and coverage in other summarization work (Carbonell and Goldstein, 1998; Lin and Bilmes, 2010; See et al., 2017).

We train the model by minimizing the negative expected $R(\mathcal{C}, \mathcal{H})$:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(z_1,\dots,z_{|\mathcal{D}|}) \sim p(\cdot|\mathcal{D})}[R(\mathcal{C}, \mathcal{H})] \quad (8)$$

where $p(\cdot|\mathcal{D})$ is the distribution produced by the neural extractive model (see Equation (3)). Unfortunately, computing the expectation term is prohibitive, since the possible latent variable combinations are exponential. In practice, we approximate this expectation with a single sample from

---

[1] We also experimented with unnormalized probabilities (i.e., excluding the exp in Equation (4)), however we obtained inferior results.

the distribution of $p(\cdot|\mathcal{D})$. We use the REINFORCE algorithm (Williams, 1992) to approximate the gradient of $\mathcal{L}(\theta)$:

$$\nabla\mathcal{L}(\theta) \approx \sum_{i=1}^{|\mathcal{D}|} \nabla \log p(z_i|z_{1:i-1}, \mathbf{h}_{i-1}^{D})[R(\mathcal{C},\mathcal{H}) - b_i]$$

Note that the model described above can be viewed as a reinforcement learning model, where $R(\mathcal{C},\mathcal{H})$ is the reward. To reduce the variance of gradients, we also introduce a baseline linear regression[2] model b_i (Ranzato et al., 2016) to estimate the expected value of $R(\mathcal{C},\mathcal{H})$. To avoid random label sequences during sampling, we use a pre-trained extractive model to initialize our latent model.

3 Experiments

Dataset and Evaluation We conducted experiments on the CNN/Dailymail dataset (Hermann et al., 2015; See et al., 2017). We followed the same pre-processing steps as in See et al. (2017). The resulting dataset contains 287,226 document-summary pairs for training, 13,368 for validation and 11,490 for test. To create sentence level labels, we used a strategy similar to Nallapati et al. (2017). We label the subset of sentences in a document that maximizes ROUGE (against the human summary) as `True` and all other sentences as `False`. Using the method described in Section 2.2, we created a compression dataset with 1,045,492 sentence pairs for training, 53,434 for validation and 43,382 for testing. We evaluated our models using full length F1 ROUGE (Lin, 2004) and the official `ROUGE-1.5.5.pl` script. We report ROUGE-1, ROUGE-2, and ROUGE-L.

Implementation We trained our extractive model on an Nvidia K80 GPU card with a batch size of 32. Model parameters were uniformly initialized to $[-\frac{1}{\sqrt{c}}, \frac{1}{\sqrt{c}}]$ (c is the number of columns in a weight matrix). We used Adam (Kingma and Ba, 2014) to optimize our models with a learning rate of 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We trained our extractive model for 10 epochs and selected the model with the highest ROUGE on the validation set. We rescaled the gradient when its norm exceeded 5 (Pascanu et al., 2013) and regularized all LSTMs with a dropout rate of 0.3 (Srivastava et al., 2014; Zaremba et al., 2014). We also applied word dropout (Iyyer et al., 2015) at rate 0.2. We set the hidden unit size $d = 300$ for both word-level and sentence-level LSTMs and all LSTMs had one layer. We used 300 dimensional pre-trained FastText vectors (Joulin et al., 2017) to initialize our word embeddings. The latent model was initialized from the extractive model (thus both models have the same size) and we set the weight in Equation (7) to $\alpha = 0.5$. The latent model was trained with SGD, with learning rate 0.01 for 5 epochs. During inference, for both extractive and latent models, we rank sentences with $p(y_i = \texttt{True}|y_{1:i-1}, \mathcal{D})$ and select the top three as summary (see also Equation (3)).

Model	R-1	R-2	R-L
LEAD3	40.34	17.70	36.57
LEAD3 (Nallapati et al., 2017)	39.20	15.70	35.50
abstract	35.46	13.30	32.65
pointer+coverage	39.53	17.28	36.38
abstract-RL	**41.16**	15.75	**39.08**
abstract-ML+RL	39.87	15.82	36.90
SummaRuNNer	39.60	16.20	35.30
EXTRACT-CNN	40.11	17.52	36.39
REFRESH (Narayan et al., 2018)	40.00	18.20	36.60
EXTRACT	40.62	18.45	37.14
LATENT	41.05	**18.77**	37.54
LATENT+COMPRESS	36.69	15.43	34.33

Table 1: Results of different models on the CNN/Dailymail test set using full-length F1 ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L).

Comparison Systems We compared our model against LEAD3, which selects the first three leading sentences in a document as the summary and a variety of abstractive and extractive models. Abstractive models include a sequence-to-sequence architecture (Nallapati et al., 2016); *abstract*), its pointer generator variant (See et al. 2017; *pointer+coverage*), and two reinforcement learning-based models (Paulus et al. 2017; *abstract-RL* and *abstract-ML+RL*). We also compared our approach against an extractive model based on hierarchical recurrent neural networks (Nallapati et al. 2017; *SummaRuNNer*), the model described in Section 2.1 (EXTRACT) which encodes sentences using LSTMs, a variant which employs CNNs instead (Cheng and Lapata 2016; EXTRACT-CNN), as well as a similar system based on reinforcement learning (Narayan et al. 2018; REFRESH).

---

[2] The linear regression model b_t is trained by minimizing the mean squared error between the prediction of b_t and $R(\mathcal{C},\mathcal{H})$.

Results As shown in Table 1, EXTRACT, our extractive model outperforms LEAD3 by a wide margin. EXTRACT also outperforms previously published extractive models (i.e., *SummaRuNNer*, EXTRACT-CNN, and REFRESH). However, note that *SummaRuNNer* generates anonymized summaries (Nallapati et al., 2017) while our models generate non-anonymized ones, and therefore the results of EXTRACT and *SummaRuNNer* are not strictly comparable (also note that LEAD3 results are different in Table 1). Nevertheless, EXTRACT exceeds LEAD3 by +0.75 ROUGE-2 points and +0.57 in terms of ROUGE-L, while *SummaRuNNer* exceeds LEAD3 by +0.50 ROUGE-2 points and is worse by −0.20 points in terms of ROUGE-L. We thus conclude that EXTRACT is better when evaluated with ROUGE-2 and ROUGE-L. EXTRACT outperforms all abstractive models except for *abstract-RL*. ROUGE-2 is lower for *abstract-RL* which is more competitive when evaluated against ROUGE-1 and ROUGE-L.

Our latent variable model (LATENT; Section 2.3) outperforms EXTRACT, despite being a strong baseline, which indicates that training with a loss directly based on gold summaries is useful. Differences among LEAD3, EXTRACT, and LATENT are all significant with a 0.95 confidence interval (estimated with the ROUGE script). Interestingly, when applying the compression model from Section 2.2 to the output of our latent model (LATENT+COMPRESS), performance drops considerably. This may be because the compression model is a sentence level model and it removes phrases that are important for creating the document-level summaries.

4 Conclusions

We proposed a latent variable extractive summarization model which leverages human summaries directly with the help of a sentence compression model. Experimental results show that the proposed model can indeed improve over a strong extractive model while application of the compression model to the output of our extractive system leads to inferior output. In the future, we plan to explore ways to train compression models tailored to our summarization task.

Acknowledgments

We thank the EMNLP reviewers for their valuable feedback and Qingyu Zhou for preprocessing the CNN/Dailymail dataset. We gratefully acknowledge the financial support of the European Research Council (award number 681760; Lapata).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *In Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany.

John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 104–111, Barcelona, Spain.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701. Curran Associates, Inc.

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691, Beijing, China.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *In Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2741–2749, Phoenix, Arizona.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73. ACM.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920, Los Angeles, California.

Inderjeet Mani. 2001. *Automatic Summarization*. John Benjamins Pub Co.

Rada Mihalcea. 2005. Language independent extractive summarization. In *Proceedings of the ACL Interactive poster and demonstration sessions*, pages 49–52, Ann Arbor, Michigan.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *In Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3075–3091, San Francisco, California.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023.*

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2–3):103–233.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1310–1318, Atlanta, Georgia.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304.*

MarcAurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112. urran Associates, Inc.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 565–574, Uppsala, Sweden.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329.*

On the Abstractiveness of Neural Document Summarization

Fangfang Zhang[1*] **Jin-ge Yao**[1*] **Rui Yan**[1,2]
[1]Institute of Computer Science and Technology, Peking University, China
[2]Beijing Institute of Big Data Research, Beijing, China
{ffz, yaojinge, ruiyan}@pku.edu.cn

Abstract

Many modern neural document summarization systems based on encoder-decoder networks are designed to produce abstractive summaries. We attempted to verify the degree of abstractiveness of modern neural abstractive summarization systems by calculating overlaps in terms of various types of units. Upon the observation that many abstractive systems tend to be near-extractive in practice, we also implemented a pure copy system, which achieved comparable results as abstractive summarizers while being far more computationally efficient. These findings suggest the possibility for future efforts towards more efficient systems that could better utilize the vocabulary in the original document.

1 Introduction

Document summarization has been a hot research topic in natural language processing for long. When human writers summarize a document, they often edit its constituent sentences in order to succinctly capture the meaning of the document. For instance, Jing and McKeown (2000) observed that summary authors trimmed extraneous content, combined sentences, replaced phrases or clauses with more general or specific variants. The abstractive summaries thus involve sentences which deviate from those of the source document in structure or content.

On the contrary, automated summarization generally produces extractive summaries by selecting complete sentences from the source document (Nenkova et al., 2011) to ensure that the output is grammatical.

Recently, many modern neural summarization systems based on encoder-decoder networks have been proposed, aiming at producing abstractive summaries. These systems highly rely on the attention mechanism (Bahdanau et al., 2015) that focus on different parts of input during the decoding stage. Some also suggested to use a copying mechanism (Gulcehre et al., 2016; Gu et al., 2016) to directly copy words from input. This naturally brings us to a question: in how much degree are current neural document summarizers abstractive? In this work, we conduct such a study on the popularly-used CNN / DailyMail news corpora. By calculating various types of overlaps between summaries generated by neural abstractive summarizers and the original document, we verified that many systems are in fact heavily extracting text spans from input.

Recent studies found that automated methods can generate a wider range of summaries by extracting over sub-sentential units of meaning, such as elementary discourse units (EDUs), from the source documents rather than whole sentences (Li et al., 2016; Durrett et al., 2016). We built on a rather standard pointer-generator system to produce a summarizer that purely copies from input. Limited vocabulary size makes the new summarizer more computationally efficient, without loss of performance. The findings in this paper may hint future studies towards more efficient and more effective near-extractive systems, instead of a less important target of improving abstraction.

To summarize, in this paper we provide:

- A quantitative analysis on how abstractive current neural document summarizers are by calculating various types of content overlap with the input documents.
- A simple modification on the standard pointer-generator document summarizer to produce equally good near-extractive summaries while being more computationally efficient due to largely reduced vocabulary size.

---

[*] The first two authors contributed equally.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 785–790
Brussels, Belgium, October 31 - November 4, 2018.

2 Neural Abstractive Summarization

Recently end-to-end training with encoder-decoder neural networks (Sutskever et al., 2014) have achieved huge success in data sufficient sequence transduction tasks such as machine translation, which brings potential applications for summarization tasks, especially for abstractive settings. Earlier practice is mostly achieved on abstractive sentence summarization (Rush et al., 2015), which is essentially sentence simplification working on short text inputs. These neural sentence abstraction models are able to achieve good ROUGE (Lin and Hovy, 2003) scores on headline generation benchmarks, [1] but have not been proved to be useful for generating summaries with multiple sentences for full documents with longer contexts, which is the main focus of this study on document summarization.

One possible way for document-level neural summarization is to design hierarchical encoding to represent sentences and words at different levels. Related studies treat a document as a sequence of sentences and take sentence embeddings as input for a document-level RNN, while using a convolutional network or recurrent network to generate sentence vectors from original tokens (Cheng and Lapata, 2016). Meanwhile, the attention mechanism will become hierarchical as well (Nallapati et al., 2016). When decoding, sentence-level attention weights will be used as input for calculating word-level attention weights. Experimental results in previous work suggest that such schemes could be useful for extractive summarization when calculating sentence weights, but could only generate rather disappointing results for abstractive summaries.

It has been shown to be useful to incorporate the copying mechanism (Gulcehre et al., 2016; Gu et al., 2016; See et al., 2017) that allows a word to be generated by directly copying an input word rather than producing all words from the hidden state from scratch. Meanwhile, directly optimizing ROUGE via reinforcement learning has been shown to be more effective than optimizing reference likelihood (Paulus et al., 2018).

Recent work has achieved improvements by modeling attention based on more structured inter-sentence relationships such as graphs (Tan et al., 2017; Yasunaga et al., 2017). In practice, a severe issue of repetitive generation has been reported in other related work. It has been shown helpful to encourage diversity and novelty in calculating attention weights (Chen et al., 2016; Nema et al., 2017), or incorporating different modules with mutual communications to encode different paragraphs in the input document (Celikyilmaz et al., 2018). Another perspective is to promote better information coverage, such as pre-estimating term frequencies in the target summary (Suzuki and Nagata, 2017) or directly introducing a coverage loss between encoder states and decoder states (See et al., 2017).

Among the aforementioned related studies, a few proposed systems explicitly targeted at generating *abstractive* summaries for documents. However, these systems highly rely on the attention mechanism and/or copying mechanism that heavily depends on different part of input during the decoding stage. This naturally brings to a question on whether neural summarizers are indeed generating abstractive summaries after reading and digesting the input document, or they are just extracting subparts of the original document to perform near-extractive summaries.

3 Quantitative Analysis

3.1 Approaches

To verify whether current abstractive neural summarizers are just lazy generators that tend to copy original words and text spans, we computed the overlaps between the output summaries and the original article using the overlapping ratio measured by the following units: longest common subsequences (LCS), n-grams, and full sentences.

We studied a few representative systems that explicitly claim to have the ability to output *abstractive* summaries. The systems we compared in this work include:

- A basic sequence-to-sequence model with the attention mechanism.
- The pointer-generator system plus coverage mechanism presented by See et al. (2017).
- The graph-based attention system by Tan et al. (2017), aiming at better capturing salient information.
- The Distraction system by Chen et al. (2016), which attempt to distract attention.

---

[1] One caveat is that achieving high ROUGE scores on datasets with single references only is not an indication that the system is indeed generating good results.

- The deep reinforced model (Paulus et al., 2018) which combines intra-attention mechanism and reinforcement learning to target for better ROUGE scores and summary quality. to traverse between different content of a document so as to better grasp the overall meaning for summarization.

We also attempted to include some other popular abstractive document summarizers such as the SummaRuNNer system (Nallapati et al., 2017), but we failed to reach the authors to get their system outputs.

We collected experimental results from these systems on two large-scale corpora of CNN and DailyMail, which have been almost exclusively used in recent work on neural document summarization. The corpora were originally constructed in (Hermann et al., 2015) by collecting human generated abstractive highlights from the news stories. Just like almost all recent studies on neural summarization, the main conclusions might vary on other domains or even other news datasets.

3.2 Results

Table 1 displays the results of overlaps calculated for various system outputs over the original documents. Note that the authors of the Distraction system (Chen et al., 2016) did not conduct experiments on the Daily Mail subset of data, therefore only the results on CNN dataset are shown. We also include overlap results of manually-written reference summary highlights for comparison.

We can observe that the outputs from the pointer-generator systems (See et al., 2017) have the most amount of overlaps in terms of whole sentences, and most of the words or n-grams are in fact taken from the original document without further modifications or paraphrases. This observation is predictable since the system relies heavily on the pointer network module that directly copy from the input. The deep Reinforced model (Paulus et al., 2018), which relies on an intra-attention mechanism, also have rather high overlaps with the original document, suggesting that it is also an near-extractive system by some degree.

Other abstractive neural summarizers do not tend to directly copy full sentences, but do not generate words beyond the lexical choices used in the input document either, with considerably large overlaps of n-grams in general. A notable exception is the graph-based attention system where the overlap statistics are close to manually-written reference summaries. However, we manually checked a few samples and observed that the produced summaries tend to generate contents that do not conform to the information conveyed in the original documents. This is also verified in manual rating scores described later.

We conclude that currently many neural news summarizers which claimed to be abstractive tend to directly copy large spans of contents from the original documents, at least on the CNN / Daily Mail dataset which is the almost exclusively used benchmark in recent studies.

4 Near-Extractive Summarization

Now that we have observed large long-span overlaps between generated summaries and the original documents, it is natural to think about the following question: Do we really need to generate tokens from decoder states in a neural summarizer rather than just simply copying spans from input?

As previously mentioned, near-extractive summaries containing smaller text units from the input document have been shown sufficient for producing good summaries. On the other hand, generating words from a decoder state is based on time-consuming calculations of a softmax distribution, given that the vocabulary size is relatively large. Therefore, we would like to try abandoning decoder word generation, while just directly copying words from the input document instead.

4.1 Approach

We built a summarizer that only copy input words for outputs with trivial modifications upon the pointer-generator model (See et al., 2017). Specifically, it implements a sequence-to-sequence model that uses the soft attention distribution to produce an output sequence whose elements are all from the input sequence, similar to what a pointer network (Vinyals et al., 2015) does. We simply use the attention distribution as the final copy distribution, while the model does not generate words from the whole vocabulary using a softmax layer as in original recurrent networks. The training objective is to maximize the likelihood of words contained in reference summaries, similar to what has been used in the SummaRuNNer system for abstractive training (Nallapati et al., 2017). We keep using the same hyperparameters as in the original pointer-generator model.

	LCS	unigram	bigram	4-gram	sentence
Reference	75.0%	87.6%	49.0%	34.0%	3.5%
Seq2seq	87.4%	93.2%	76.0%	66.0%	10.8%
Pointer-generator	98.2%	99.8%	92.5%	93.0%	60.1%
Pointer-generator+coverage	98.8%	99.9%	96.1%	94.0%	70.0%
Reinforced (Paulus et al., 2018)	90.6%	95.8%	85.3%	80.2%	19.3%
Graph attention (Tan et al., 2017)	74.3%	82.3%	59.9%	42.1%	3.3%
Reference(CNN)	63.0%	75.2%	39.2%	25.9%	0.8%
Distraction (Chen et al., 2016) (CNN)	74.7%	94.0%	65.4%	38.7%	0.8%

Table 1: The overlap proportions between summaries and the original document

4.2 Experiments

We conducted experiments on the CNN / Daily Mail datasets and adopt the widely used ROUGE metrics (Lin and Hovy, 2003) for evaluation as previous work did.

Table 2 shows the ROUGE scores for the produced summaries. We can see that a pure copy system could produce equally or slightly better results in terms of word-matching metrics. The coverage mechanism introduced by See et al. (2017) is also effective for a pure copy system. As a side note, we verified the observation from See et al. (2017) that current neural summarizers cannot genuinely outperform a properly implemented LEAD baseline that simply takes the first three sentences from the original document, at least for the datasets used here that mainly consist of news describing events or activities.

	R-1	R-2	R-L
PtrGen	36.44	15.66	33.42
PtrGen + cov	39.53	17.28	36.38
Ptr	37.44	16.08	34.25
Ptr + cov	39.74	**17.31**	**36.53**
Lead3	**39.98**	17.25	36.20

Table 2: ROUGE scores on CNN/Daily Mail

We also conducted human evaluation on outputs for a sample of 30 documents in the common subpart of the system inputs. We asked three raters to evaluate on the following metrics in a summary using 1-5 scoring scheme (5 is the best, and rational numbers were allowed if raters felt uncertain over some cases): *informativeness* (INF), *relevance* (REL), *fluency* (FLU) and *coherence* (COH), as used by Grusky et al. (2018). The results are listed in Table 3. We find the pure copy system performs similarly to the pointer-generator. However, we can also observe a rough trend that: the more abstractive a system is, the higher the chance of generating irrelevant or grammatically worse content. Such observation is consistent with manual evaluation results conducted by another study towards more abstraction (Kryściński et al., 2018), in parallel to our work. This is a signal that other than pursuing for heavily abstractiveness, we could also spend more efforts on directly identifying and extracting useful pieces from the input, in order to get more controlled and more useful summaries with better quality beyond the heavily biased ROUGE metrics (Chaganty et al., 2018).

	INF	REL	FLU	COH
Lead3	2.97	3.20	4.10	3.33
Pure copy	3.03	3.37	3.87	3.14
PtrGen	3.01	3.44	3.65	3.11
Reinforced	3.18	3.17	3.57	2.97
Distraction	2.44	2.88	3.37	2.70
GraphAtt	2.75	2.87	2.47	2.14
Reference	3.71	4.32	4.54	4.12

Table 3: Human ratings

A larger merit for implementing a pure copy system is that it is more computationally effective than abstractive summarizers that generate words in a large vocabulary from decoder states. Table 4 lists the speed for decoding as well as the memory costs in training for the pure copy system, compared with the original pointer-generator system. The numbers are averaged results from multiple runs on the same computing environment of Tesla M40 GPU. We can see that the speed doubles from a pure copy summarizer, while the GPU memory cost is reduced to around a quarter.

We visualize the copying probabilities (attention weights) for an example summary along with the original document in Figure 1. We can observe that the pure copy system tends to attend on con-

	Pure copy	PtrGen
Decoding Speed	0.87 step/s	0.44 step/s
GPU Usage	2216MB	8368MB
Memory Cost	2.67GB	3.30GB

Table 4: Comparison of computational costs

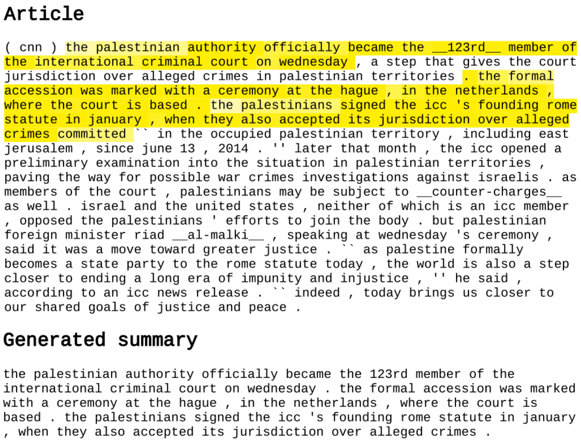

Article

(cnn) the palestinian authority officially became the __123rd__ member of the international criminal court on wednesday , a step that gives the court jurisdiction over alleged crimes in palestinian territories . the formal accession was marked with a ceremony at the hague , in the netherlands , where the court is based . the palestinians signed the icc 's founding rome statute in january , when they also accepted its jurisdiction over alleged crimes committed `` in the occupied palestinian territory , including east jerusalem , since june 13 , 2014 . '' later that month , the icc opened a preliminary examination into the situation in palestinian territories , paving the way for possible war crimes investigations against israelis . as members of the court , palestinians may be subject to __counter-charges__ as well . israel and the united states , neither of which is an icc member , opposed the palestinians ' efforts to join the body . but palestinian foreign minister riad __al-malki__ , speaking at wednesday 's ceremony , said it was a move toward greater justice . `` as palestine formally becomes a state party to the rome statute today , the world is also a step closer to ending a long era of impunity and injustice , '' he said , according to an icc news release . `` indeed , today brings us closer to our shared goals of justice and peace .

Generated summary

the palestinian authority officially became the 123rd member of the international criminal court on wednesday . the formal accession was marked with a ceremony at the hague , in the netherlands , where the court is based . the palestinians signed the icc 's founding rome statute in january , when they also accepted its jurisdiction over alleged crimes .

Figure 1: Visualization of copy probabilities

tinuous spans of input to form sentences.

5 Conclusion

In this work we attempted to quantify the abstractiveness of modern neural abstractive summarization systems by calculating overlaps of various units. Inspired by the observation that many systems tend to be near-extractive, we also implemented a pure copy system and achieved comparable performance while being far more efficient. Giving the observations that the abstractive summaries produced by current systems have lower quality than extractive summaries, our study should give hints for focusing on better extractions from the input, rather than deliberately pursuing for more abstraction but losing real quality beyond automatic metrics.

Acknowledgments

We would like to thank Abigail See for her open-sourced implementation of the pointer-generator summarizer, and Jiwei Tan, Qian Chen, Romain Paulus for providing the outputs from their systems. We also thank the anonymous reviewers for helpful comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61876196 and No. 61672058). Rui Yan was sponsored by CCF-Tencent Open Research Fund and Microsoft Research Asia (MSRA) Collaborative Research Program. Rui Yan is the corresponding author of this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana. Association for Computational Linguistics.

Arun Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debasing automatic metrics in natural language evaluation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, Melbourne, Australia. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2754–2760.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008, Berlin, Germany. Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1693–1701.

Hongyan Jing and Kathleen R McKeown. 2000. Cut and paste based text summarization. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 178–185. Association for Computational Linguistics.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Junyi Jessy Li, Kapil Thadani, and Amanda Stent. 2016. The role of discourse units in near-extractive summarization. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 137–147, Los Angeles. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI Conference on Artificial Intelligence*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Ca glar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Preksha Nema, Mitesh M. Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1063–1072, Vancouver, Canada. Association for Computational Linguistics.

Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 291–297, Valencia, Spain. Association for Computational Linguistics.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181, Vancouver, Canada. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28*, pages 2692–2700.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, Vancouver, Canada. Association for Computational Linguistics.

Automatic Essay Scoring Incorporating Rating Schema via Reinforcement Learning

Yucheng Wang[1], **Zhongyu Wei**[23], **Yaqian Zhou**[13*], **Xuanjing Huang**[13]
[1]School of Computer Science, Fudan University, Shanghai, China
[2]School of Data Science, Fudan University, Shanghai, China
[3]Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai, China
`{yuchengwang14,zywei,zhouyaqian,xjhuang}@fudan.edu.cn`

Abstract

Automatic essay scoring (AES) is the task of assigning grades to essays without human interference. Existing systems for AES are typically trained to predict the score of each single essay at a time without considering the rating schema. In order to address this issue, we propose a reinforcement learning framework for essay scoring that incorporates quadratic weighted kappa as guidance to optimize the scoring system. Experiment results on benchmark datasets show the effectiveness of our framework.

1 Introduction

In recent years, neural networks have been widely used to grade student essays automatically and achieve state-of-the-art performance. In particular, a distributed representation is learned for an essay with variant neural networks and a linear layer is then used to produce the final score. Existing researches focus on learning better essay representation using different neural networks, including long short-term memory (LSTM) network (Taghipour and Ng, 2016), hierarchical convolutional neural networks (CNN) (Dong and Zhang, 2016), hierarchical CNN-LSTM structure with attention mechanism (Dong et al., 2017), and SkipFlow LSTM (Tay et al., 2017).

The major evaluation metric for AES is quadratic weighted kappa (QWK), which is also the official metric of Automated Student Assessment Prize[1] (ASAP). It evaluates the scoring results by taking rating schema into consideration. Because QWK is not differentiable, it is hard to train systems via optimizing this metric directly. Alternatively, existing AES systems are typically trained to predict the score for a single essay and optimized using mean square error (MSE). The gap between training and testing also limits the performance of state-of-the-art AES systems.

Recently, reinforcement learning (RL) has been introduced to optimize models in terms of non-differentiable quality metrics and studies have shown its effectiveness for various tasks including language generation (Ranzato et al., 2015; Rennie et al., 2016; Zhang et al., 2017), machine translation (Bahdanau et al., 2016) and relation classification (Feng et al., 2018).

Inspired by these researches, we propose a novel reinforcement learning framework that incorporates QWK as the guidance to optimize the essay scoring system. In our framework, we score a pack of essays at a time and the scoring of each single essay is treated as an action. The QWK value computed for the pack of essays is then delivered as a reward to update the scoring system. Because the existing regression-based essay scorer is unable to generate a probability distribution in nature, it is non-trivial to be used within the reinforcement learning framework. We therefore propose to use a classification-based scoring system instead. The proposed framework is evaluated in the benchmark datasets from ASAP and experiment results confirm its effectiveness on two different settings of essay representation structures.

2 Model

Typically, an essay scorer contains two components, namely, essay representation and essay scoring. The component of essay representation transforms an input essay into a distributed vector and the component of essay scoring assigns a score to the essay based on the vector. Both components are usually trained jointly. In order to incorporate QWK to guide the process of essay scoring, we introduce a novel essay scoring strategy named *packed evaluation*. At each time, essay scorer grades a pack of essays together with the target essay, and QWK is calculated for the pack.

---

[*] Corresponding author

[1] https://www.kaggle.com/c/asap-aes

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 791–797
Brussels, Belgium, October 31 - November 4, 2018.

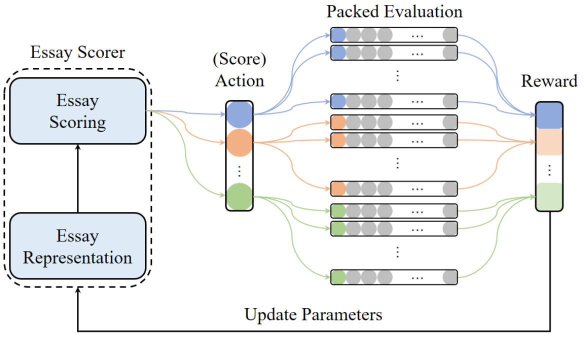

Figure 1: The reinforcement learning framework for automatic essay scoring. Node in color stands for a target essay and nodes in grey are essays randomly chose to form a pack for QWK calculation.

To avoid contingency, for each target essay, we repeat the packed evaluation multiple times by randomly choosing other essays in a pack. And the average QWK it achieves is set to be the reward. The reward is then delivered to the essay scorer as a weak signal to supervise the scorer. Figure 1 illustrates the training process of our model. We will introduce the different parts in detail in the rest of this section.

2.1 Essay Representation

This component converts an input essay into a dense vector as its representation. Recurrent neural networks (Williams and Zipser, 1989) are widely used to learn a representation for a sequence of words for essay scoring. Following existing researches, we also use recurrent neural networks (RNN) and test two different structures.

Bidirectional LSTM We first use a double-layer bi-directional LSTM network (Hochreiter and Schmidhuber, 1997) to process the essay. LSTM is a variant of recurrent neural network which uses gates to control the information flow. Our LSTM processes one word at a timestamp. Given the word embedding sequence $\{x_1, x_2, ..., x_n\}$ for the essay, the hidden states of the LSTM are calculated as follows:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

$$h_t = o_t \circ tanh(c_t)$$

where $W_f, W_i, W_o, W_c, U_f, U_i, U_o$ and U_c are weight matrices, b_f, b_i, b_o and b_c are bias vectors. σ denotes sigmoid function and $\circ$ denotes element-wise multiplication.

In particular, the average value over all hidden states of each LSTM layer are computed, and we concatenate the mean states of the two layers together as the embedding vector of the essay. Given $h_{i,j}$ as the *j-th* hidden state of the *i-th* layer, the layer outputs and the essay embedding vector E are defined as follows:

$$output_i = \frac{1}{n}\sum_{j=1}^{n} h_{i,j}$$

$$E = \begin{bmatrix} output_0 \\ output_1 \end{bmatrix}$$

Dilated LSTM Dilated recurrent neural networks (Chang et al., 2017) are proved to be more effective than traditional RNNs in long sequence processing, by capturing multi-timescale information along the sequence, with the mechanism of dilated skip connections.

Denoting $s^i_t = f(x^i_t, s^i_{t-1})$ as the iteration of cell states in traditional RNNs, states in dilated RNNs are iterated as $s^i_t = f(x^i_t, s^i_{t-k^i})$, where k^i is the skip length in the *i-th* layer. In order to keep the most information active, we simply concatenate the average hidden states of every layer to form the essay embedding.

$$E = concat(output_i), for\ i\ in\ 1, 2, ..., L$$

where L is the number of layers.

2.2 Essay Scoring

Traditionally, a linear layer with sigmoid function is used to score an essay. Given an essay embedding vector E, the essay score is calculated as follows:

$$y = sigmoid(W_l^T E + b_l)$$

where W_l and b_l are weight vector and bias for scoring. By running n examples together, mean square error is used to evaluate the predicted score.

$$loss_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y and $\hat{y}$ are score vectors representing predicted scores and ground truth scores, respectively. As we can see, such objective function is unable to take rating schema into consideration.

The regression-based scorer only outputs a single value without probability distribution. It is thus non-trivial to use it for policy learning in RL framework directly. Therefore, we propose to use a classification-based scorer, in which different score categories and their probabilities constitute an action space.

Classification-based Scoring We first feed the essay vector into a fully connected layer, then softmax function is used to transform the output into a probability distribution. Given an essay embedding vector E, the probability distribution vector c is calculated as follows:

$$c = softmax(W_c E + b_c)$$

where W_c and b_c are weight matrix and bias vector, respectively.

Given the ground truth category, cross entropy loss is applied to evaluate the agreement of the probabilities as follows:

$$loss_{CE} = -\sum_{i=1}^{N} Y_i log(c_i)$$

where N is the number of categories, which is equal to the number of possible ratings. Y is a one-hot vector with the element representing the ground truth category set as one.

Inter-class Penalty Cross entropy loss used in classification-based scorer does not imply the difference between categories, i.e. the rank information that is deemed to be important for essay scoring. Thus we enforce a penalty in addition to the cross entropy loss. Inspired by the definition of QWK, the penalty vector p is defined as follows:

$$p_i = \frac{(i - score)^2}{(N-1)^2}$$

where $score$ is the ground truth score of the essay.

The penalty loss function is defined as:

$$loss_P = \sum_{i=1}^{N} c_i p_i$$

Mixed Scoring In practice, we jointly train both a regression-based scoring layer and a classification-based scoring layer over the same document representation to help the classification-based scorer converge. By combining the two scorers together, the overall loss function can be written as:

$$loss_{pre} = \alpha_0 loss_{MSE} + \beta_0 loss_{CE} + \gamma_0 loss_P$$

where α_0, β_0 and γ_0 are hyper parameters. Mixed scoring is used as a pre-train model for our essay scorer in the phase of reinforcement learning.

2.3 Reinforcement Learning

We define our loss function as the negative expected reward:

$$loss_{RL} = -\mathrm{E}_{\tau \sim p(\tau)} r(\tau)$$

where τ is the set of actions, r denotes the reward, which is the average QWK an essay achieves in the packed evaluation.

By running n examples at a time, according to the REINFORCE algorithm (Williams, 1992), an approximated gradient can be calculated by:

$$\frac{\partial loss_{RL}}{\partial \theta} = \sum_{i=1}^{n} [\partial log(p_{i,y}|E_i;\theta) R_i]$$

where θ denotes all parameters relevant to score calculation, and $\partial log(p_{i,y}|E_i;\theta)$ can be computed by standard back propagation.

Note that only the classification-based scorer is involved in the process of reinforcement learning for essay scoring. The overall loss function for this phase can be written as:

$$loss_{overall} = \alpha_1 loss_{RL} + \beta_1 loss_{CE} + \gamma_1 loss_P$$

where α_1, β_1 and γ_1 are hyper parameters.

2.4 Quadratic Weighted Kappa(QWK)

QWK calculation emphasizes on the overall rating schema. By setting QWK as the reward, our model is trained at a macro aspect taking the grading specialty of different sets of essays into consideration.

An *N-by-N* quadratic weight matrix *W* is first computed to encode the rating information.

$$W_{i,j} = \frac{(i-j)^2}{(N-1)^2}$$

where N is the number of possible ratings. An *N-by-N* matrix *A* is calculated such that $A_{i,j}$ corresponds to the number of essays that receive a score *i* by the human rater, and a score *j* by the scoring system. Another *N-by-N* matrix *B* is constructed as the outer product of the histogram vectors of the two ratings. *A* and *B* are then normalized such that they have the same sum. Finally, from the three matrices, the quadratic weighted kappa is calculated as follows:

$$\kappa = 1 - \frac{\Sigma_{i,j} W_{i,j} A_{i,j}}{\Sigma_{i,j} W_{i,j} B_{i,j}}$$

set	# of essays	avg. of len.	rating range
1	1783	350	2-12
2	1800	350	1-6
3	1726	150	0-3
4	1772	150	0-3
5	1805	150	0-4
6	1800	150	0-4
7	1569	250	0-30
8	723	650	0-60

Table 1: Details of the ASAP dataset.

3 Experiment

3.1 Experiment Setup

The ASAP dataset is used for evaluation. It consists of essays written by middle-school English-speaking students ranging among eight different topics. More details are listed in Table 1. As there are no released labels for the test data, we separate the validation set and test set from the original training data. Following Taghipour and Ng (2016) and Dong et al. (2017), we use 5-fold cross-validation. In each fold, the split is 60%, 20%, 20% for training, validation and testing respectively.

All essays are parsed with the NLTK[2] tokenizer. We pre-train the word embedding via word2vec (Mikolov et al., 2013) on the whole dataset. The number of hidden states in LSTMs is 200. We use a four-layer double-directional dilated LSTM with skip lengths 1,2,4,8 in each layer respectively. During the training and the scoring, scores are scaled to range [0,1] for regression-based scorer. They are restored to integers when calculating QWK values. In the RL phase, the pack size is 64 essays, and packed evaluation is repeated 7 times per essay. The essay scorer for RL is pre-trained by mixed scoring.

We compare the performance of different approaches:

- **B0**: This model uses a double-layer bi-directional LSTM to encode an essay and mean square error as objective function to train the essay scorer;
- **B1**: This is a classification-based scorer and it is trained jointly with a regression-based scorer;
- **P0**: Based on B1, this model incorporates penalty loss function;
- **P1**: This model shares the same settings with *P0*, but uses dilated LSTM instead of a double-layer bi-directional LSTM for essay representation;
- **RL0**: This model uses *P0* as the scorer under our reinforcement learning framework;
- **RL1**: This model uses *P1* as the scorer under our reinforcement learning framework.

set	B0	B1	P0	RL0	P1	RL1
1	.711	.666	.666	.680	.759	**.766**
2	.582	.579	.579	.589	.630	**.659**
3	.627	.653	.662	.670	.673	**.688**
4	.758	.758	.761	.771	.768	**.778**
5	.768	.781	.786	.796	.795	**.805**
6	.776	.786	.787	**.798**	.790	.791
7	**.766**	.711	.726	.737	.748	.760
8	**.604**	.491	.525	.545	.536	.545
Avg	.699	.678	.687	.698	.712	**.724**

Table 2: Experiment results for different models in terms of QWK. **Bolded** number is the best performance in each row.

3.2 Results

The overall results of our models in terms of QWK are shown in Table 2. We have the following findings:

- By incorporating a penalty loss to the classification scorer, the performance of *P0* is equal to or better than *B1* on all the eight sets. This indicates the effectiveness of combining rank information with cross-entropy loss for essay scoring.
- By replacing double-layer bi-directional LSTM with dilated LSTM, *P1* improves the QWK values by a large margin compared with *P0* on all the eight sets. This indicates the effectiveness of using dilated LSTM for document representation for the task of automatic essay scoring. The performance improvement brought by *P1* compared to *P0* is even greater when the length of essays are higher (set 1,2,7,8, see Table 1), indicating that dilated networks are specifically better at long sequence processing.
- By incorporating QWK to guide the optimization of essay scorer, approaches (*RL0* and *RL1*) with reinforcement learning strategy can improve the performance consistently on all the eight sets compared to their counterparts (*P0* and *P1*). We also performed

---

[2] http://www.nltk.org

one-tailed *t*-test, showing that the improvements brought by reinforcement learning are significant with $p < 0.05$ compared to their base scorer models (*RL1* vs. *P1* and *RL2* vs. *P2*).

- The performance of classification-based scorer *B1* can equate or improve the performance on four datasets (set 3,4,5,6) compared with regression-based scorer *B0*. The rating ranges for set 1,2,7,8 are much greater than set 3,4,5,6 (see Table 1). The performance difference between *B1* and *B0* decreases (from positive to negative) when the number of rating categories increases. This is because when the number of categories get larger, it requires much more parameters for the classification-based scorer to be well trained. Given N categories, the classification layer should output N probabilities for each category per essay, costing N times more parameters than regression-based scoring.

4 Related Work

There are two lines of research related to our work including text quality evaluation and reinforcement learning for natural language processing.

4.1 Text Quality Evaluation

Traditionally, AES models are usually divided into three categories: classification, regression and ranking. Naive Bayes models are mostly used in classification tasks. Larkey (1998) use bag-of-word features. Following that, Rudner and Liang (2002) develop a system based on multinomial Bernoulli Naive Bayes, using content and style features. E-rater (Attali and Burstein, 2004) is one of the earliest systems to adopt regression methods. Phandi et al. (2015) use correlated Bayesian Linear Ridge Regression (cBLRR) focusing on domain-adaptation tasks. Ranking models use linguistic features. Yannakoudakis et al. (2011) formulate AES as a pair-wise ranking problem by ranking the order of pair essays. Chen and He (2013) formulate AES into a list-wise ranking problem by considering the order relation among the whole essays.

Argument quality evaluation is a task closely related to AES, which involves evaluation of argumentative texts with various grains (argument-level, post-level, etc.). Tan et al. (2016); Wei et al. (2016a); Wang et al. (2017) make use of linguistic features to evaluate the persuasiveness of arguments in online forums. Wei et al. (2016b); Ji et al. (2018) consider features from the perspectives of argumentation interaction between participants. Persing and Ng (2017) construct their model based on error types for argumentation.

4.2 Reinforcement Learning for Natural Language Processing

Being able to optimize non-differentiable quality metrics, reinforcement learning has been widely used in natural language processing tasks such as machine translation (Bahdanau et al., 2016), image captioning (Rennie et al., 2016; Zhang et al., 2017) and text summarization (Ranzato et al., 2015). To the best of our knowledge, this paper is the first attempt to optimize the scorer by QWK that considers rating schema.

Skip connections in RNNs are capable of capturing long-term dependencies in sequences. Vezhnevets et al. (2017) introduces dilated LSTM to allow its manager to operate at a low temporal resolution. Yu et al. (2017) propose a reinforcement learning method to let the network learn how long to skip.

5 Conclusion and Future Work

In this paper, we propose a reinforcement learning framework incorporating QWK metric as the reward to train the essay scoring system directly. A packed evaluation strategy is used for QWK computation and the scoring of each essay is treated as a single action. In particular, dilated LSTM is used to encode an essay, and a softmax layer is utilized for essay grading. Experiment results on benchmark datasets prove that training the grading system toward QWK is effective.

Further analysis on experiment results indicates the disadvantage of using a classification-based scorer for essays with complex grading schema. One of the future directions can be exploring other kinds of scoring actions than classification under the reinforcement learning framework.

Acknowledgments

Thanks for the constructive comments from anonymous reviewers. This work is partially supported by National Natural Science Foundation of China (Grant No. 61702106), Shanghai Science and Technology Commission (Grant No. 17JC1420200, Grant No. 17YF1427600 and Grant No. 16JC1420401).

References

Yigal Attali and Jill Burstein. 2004. Automated essay scoring with e-rater® v. 2.0. *ETS Research Report Series*, 2004(2).

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. 2017. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 76–86.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1752.

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring–an empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1072–1077.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 153–162.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*.

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

Lu Ji, Zhongyu Wei, Xiangkun Hu, Yang Liu, Qi Zhang, and Xuanjing Huang. 2018. Incorporating argument-level interactions for persuasion comments evaluation using co-attention model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3703–3714.

Leah S Larkey. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 90–95.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Isaac Persing and Vincent Ng. 2017. Why cant you convince me? modeling weaknesses in unpersuasive arguments. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4082–4088. AAAI Press.

Peter Phandi, Kian Ming A Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 431–439.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563*.

Lawrence M Rudner and Tahung Liang. 2002. Automated essay scoring using bayes' theorem. *The Journal of Technology, Learning and Assessment*, 1(2).

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1891.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th international conference on world wide web*, pages 613–624. International World Wide Web Conferences Steering Committee.

Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Skipflow: Incorporating neural coherence features for end-to-end automatic text scoring. *arXiv preprint arXiv:1711.04981*.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*.

Lu Wang, Nick Beauchamp, Sarah Shugars, and Kechen Qin. 2017. Winning on the merits: The joint effects of content and style on debate outcomes. *arXiv preprint arXiv:1705.05040*.

Zhongyu Wei, Yang Liu, and Yi Li. 2016a. Is this post persuasive? ranking argumentative comments in online forum. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 195–200.

Zhongyu Wei, Yandi Xia, Chen Li, Yang Liu, Zachary Stallbohm, Yi Li, and Yang Jin. 2016b. A preliminary study of disputation behavior in online debating forum. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 166–171.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 180–189. Association for Computational Linguistics.

Adams Wei Yu, Hongrae Lee, and Quoc V Le. 2017. Learning to skim text. *arXiv preprint arXiv:1704.06877*.

Li Zhang, Flood Sung, Feng Liu, Tao Xiang, Shaogang Gong, Yongxin Yang, and Timothy M Hospedales. 2017. Actor-critic sequence training for image captioning. *arXiv preprint arXiv:1706.09601*.

Identifying Well-formed Natural Language Questions

Manaal Faruqui **Dipanjan Das**
Google AI Language

Abstract

Understanding search queries is a hard problem as it involves dealing with "word salad" text ubiquitously issued by users. However, if a query resembles a well-formed question, a natural language processing pipeline is able to perform more accurate interpretation, thus reducing downstream compounding errors. Hence, identifying whether or not a query is well formed can enhance query understanding. Here, we introduce a new task of identifying a well-formed natural language question. We construct and release a dataset of 25,100 publicly available questions classified into well-formed and non-wellformed categories and report an accuracy of 70.7% on the test set. We also show that our classifier can be used to improve the performance of neural sequence-to-sequence models for generating questions for reading comprehension.

1 Introduction

User issued search queries often do not follow formal grammatical structure, and require specialized language processing (Bergsma and Wang, 2007; Barr et al., 2008; Manshadi and Li, 2009; Mishra et al., 2011). Traditional natural language processing (NLP) tools trained on formal text (e.g. treebanks) often have difficulty analyzing search queries; the lack of regularity in the structure of queries makes it difficult to train models that can optimally process the query to extract information that can help understand the user intent behind the query (Baeza-Yates et al., 2006).

One clear direction to improve query processing is to annotate a large number of queries with the desired annotation scheme. However, such an annotation can be prohibitively expensive and models trained on such queries might suffer from freshness issues, as the domain and nature of queries evolve frequently (Markatos, 2001; Bawa et al., 2003; Roy et al., 2012). Another direction is to obtain a paraphrase of the given query that is a grammatical natural language question, and then analyze that paraphrase to extract the required information (Nogueira and Cho, 2017; Buck et al., 2018). There are available tools and datasets, such as Quora question paraphrases and the Paralex dataset (Fader et al., 2013) – for identifying query paraphrases (Wang et al., 2017; Tomar et al., 2017), but these datasets do not contain information about whether a query is a natural language question or not. Identifying well-formed natural language questions can also facilitate a more natural interaction between a user and a machine in personal assistants or chatbots (Yang et al., 2014; Mostafazadeh et al., 2016) or while recommending related queries in search-engines.

Identifying a well-formed question should be easy by parsing with a grammar, such as the English resource grammar (Copestake and Flickinger, 2000), but such grammars are highly precise and fail to parse more than half of web queries. Thus, in this paper we present a model to predict whether a given query is a well-formed natural language question. We construct and publicly release a dataset of 25,100 queries annotated with the probability of being a well-formed natural language question (§2.1). We then train a feed-forward neural network classifier that uses the lexical and syntactic features extracted from the query on this data (§2.2). On a test set of 3,850 queries, we report an accuracy of 70.1% on the binary classification task. We also demonstrate that such a query well-formedness classifier can be used to improve the quality of a sequence-to-sequence question generation model (Du et al., 2017) by showing an improvement of 0.2 BLEU score in its performance (§3). Our dataset ise available for download at `http://goo.gl/language/query-wellformedness`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 798–803
Brussels, Belgium, October 31 - November 4, 2018.

Query	Well-formed?	Reasoning
what is the breed of scooby doo?	1	Grammatical and an explicit question
tell me whats the breed of scooby doo?	0	A command but not a question
headache evenings?	0	Ungrammatical and not a question
what causes headaches during evenings	1	Grammatical and an explicit question
what 12.5 as a fraction?	0	An explicit question but ungrammatical

Table 1: Examples of well-formed and non-wellformed queries according to the annotation guideline.

2 Well-formed Natural Language Question Classifier

In this section we describe the data annotation, and the models used for question well-formedness classification.

2.1 Dataset Construction

We use the Paralex corpus (Fader et al., 2013) that contains pairs of noisy paraphrase questions. These questions were issued by users in WikiAnswers (a Question-Answer forum) and consist of both web-search query like constructs (*"5 parts of chloroplast?"*) and well-formed questions (*"What is the punishment for grand theft?"*), and thus is a good resource for constructing the question well-formedness dataset. We select 25,100 queries from the unique list of queries extracted from the corpus such that no two queries in the selected set are paraphrases. The queries are then annotated into well-formed or non-wellformed questions. We define a query to be a well-formed natural language question if it satisfies the following:

1. Query is grammatical.
2. Query is an explicit question.
3. Query does not contain spelling errors.

Query (q)	$p_{wf}(q)$
population of owls just in north america?	0.0
who disscoverd rihanna?	0.2
what countries have genocide happened in?	0.6
what is released when an ion is formed?	1.0

Table 2: Examples of human annotations on query well-formedness.

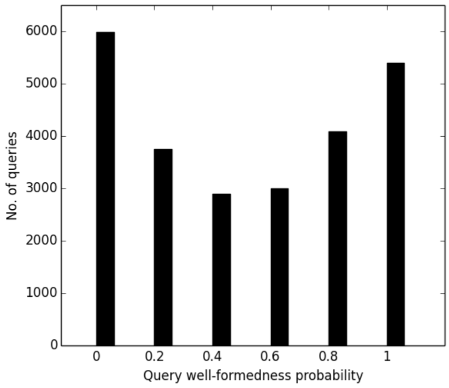

Figure 1: The distribution of the annotated questions according to well-formedness probability.

Table 1 shows some examples that were shown to the annotators to illustrate each of the above conditions. Every query was labeled by five different crowdworkers with a binary label indicating whether a query is well-formed or not. We average the ratings of the five annotators to get the probability of a query being well-formed. Table 2.1 shows some queries with obtained human annotation. Humans are pretty good at identifying an implicit query (*"Population of owls..."*) or a simple well-formed question (*"What is released..."*), but may miss out on subtle spelling mistakes like "disscovered" or disagree on whether the determiner "the" is needed before the word "genocide" (*"What countries have genocide happened in?"*). Similar to other NLP tasks like entailment (Dagan et al., 2006; Bowman et al., 2015), paraphrasing (Wieting et al., 2015) etc. we rely on the wisdom of the crowd to get such annotations in order to make the data collection scalable and language-independent.

Figure 1 is the histogram of query well-formedness probability across the dataset. Interestingly, the number of queries where at least 4 or more annotators agree[1] on well-formedness is large: $|\{q \mid 0.8 \le p_{wf}(q) \le 0.2\}| = 19206$ queries. These constitute 76.5% of all queries in the dataset. The Fleiss' kappa (Fleiss, 1971) for measuring agreement among multiple annotators is computed to be $\kappa = 0.52$ which shows moderate agreement (Landis and Koch, 1977). We

---

[1] At least 4 annotators label the query with 0 or 1.

then randomly divided the dataset in approx. 70%, 15%, 15% ratio into training, development and test sets containing 17500, 3750, and 3850 queries respectively. While testing, we consider every query well-formed where at least 4 out of 5 annotators ($p_{wf} \geq 0.8$) marked it as well-formed.[2]

2.2 Model

We use a feed-forward neural network with 2 hidden layers with ReLU activations (Glorot et al., 2011) on each layer and a softmax at the output layer predicting 0 or 1. We extract a variety of features from the query which can be helpful in the classification. We extract character-3, 4-grams and word-1, 2-grams as they can be helpful in capturing spelling errors. In addition to lexical features, we also extract syntactic features that can inform the model on any anomaly in the structure of the query. Specifically, we annotate the query with POS-tags using SyntaxNet POS tagger (Alberti et al., 2015) and extract POS-1, 2, 3-grams.[3] Every feature in the network is represented as a real-valued embedding. All the n-grams embeddings of every feature type are summed together and concatenated to form the input layer as shown in Figure 2. The model is trained using cross-entropy loss against the gold labels for each query. The hyperparameters are tuned to maximize accuracy on the dev set and results are reported on the test set.

Hyperparameters. We fix the size of the first and second hidden layers to be 128 and 64 respectively. The character n-gram embeddings were of length 16 and all other feature embeddings were of length 25. We use stochastic gradient descent with momentum for optimization with learning rate tuned over $[0.001 - 0.3]$, a batch size of 32 and 50000 training steps.

2.3 Experiments

Baselines. The majority class baseline is 61.5% which corresponds to all queries being classified non-wellformed. The question word baseline that classifies any query starting with a question word as a well-formed question gets 54.9%.[4] Also, we used a single-layer word-level biLSTM encoder with hidden layer of length 50 to encode the question and then use this representation in the softmax layer to predict the label (Lee and Dernoncourt, 2016). This classifier achieved 65.8%.

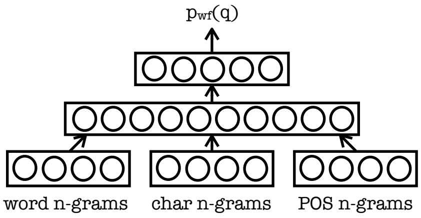

Figure 2: A feed-forward neural network for query well-formedness classification.

Model	Accuracy (%)
majority class baseline	61.5
word bi-LSTM baseline	65.8
question word baseline	54.9
word-1	65.4
word-1, 2	65.5
word-1, 2 char-3, 4	66.9
word-1, 2 POS-1, 2, 3	**70.7**
word-1, 2 char-3, 4 POS-1, 2, 3	70.2
Approx. human upper bound	88.4

Table 3: Performance of well-formedness query classifier on the test set.

Results. The best performance obtained is 70.7% while using word-1, 2-grams and POS-1, 2, 3-grams as features. Using POS n-grams gave a strong boost of 5.2 points over word unigrams and bigrams. Although character-3, 4-grams gave improvement over word unigrams and bigrams, the performance did not sustain when combined with POS tags.[5] A random sample of 1000 queries from the test set were annotated by one of the authors of the paper with proficiency in English, which matched the gold label with 88.4% accuracy providing an approximate upper-bound for model performance.

A major source of error is our model's inability to understand deep semantics and syntax. For example, *"What is the history of dirk bikes?"* is labeled as a non-wellformed question with $p_{wf} =$

---

[2] We randomly selected 100 queries and manually determined if each of those queries were well-formed. We found $p_{wf}(q) = 0.8$ to be the value above which all queries were well-formed.

[3] The use of dependency labels as features and use of pre-trained Glove embeddings did not show improvement and hence omitted here for space constraints.

[4] List of question words: `https://en.wikipedia.org/wiki/Interrogative_word`

[5] We assumed character n-grams to help identify spelling mistakes, but our dataset has relatively few misspelled words–only 6 in 100 random queries.

0 by annotators because of the misspelled word “dirk” (the correct word is “dirt”). However, the POS tagger identifies “dirk” as a noun and as “NN NNS” is a frequent POS-bigram, our model tags it as a well-formed question with $p_{wf} = 0.8$, unable to identify that the word does not fit in the context of the question. Another source of error is the inability to capture long term grammatical dependencies. For example, in *“What sort of work did Edvard Munch made ?”* the verb “made” is incorrectly in the past tense instead of present tense. Our model is unable to capture the relationship between “did” and “made” and thus marks this as a well-formed question.

3 Improving Question Generation

Automatic question generation is the task of generating questions that ask about the information or facts present in either a given sentence or paragraph (Vanderwende, 2008; Heilman and Smith, 2010). Du et al. (2017) present a state-of-the-art neural sequence-to-sequence model to generate questions from a given sentence/paragraph. The model used is an attention-based encoder-decoder network (Bahdanau et al., 2015), where the encoder reads in a given text and the decoder is an LSTM RNN that produces the question by predicting one word at a time.

Du et al. (2017) use the SQuAD question-answering dataset (Rajpurkar et al., 2016) to develop a question generation dataset by pairing sentences from the text with the corresponding questions. The question generation dataset contains approx 70k, 10k, and 12k training, development and test examples. Their current best model selects the top ranked question from the n-best list produced by the decoder as the output. We augment their system by training a discriminative reranker (Collins and Koo, 2005) with the model score of the question generation model and the well-formedness probability of our classifier as features to optimize BLEU score (Papineni et al., 2002) between the selected question from the 10-best list and the reference question on the development set. We then use this reranker to select the best question from the 10-best list of the test set.

We use the evaluation package released by Chen et al. (2015) to compute BLEU-1 and BLEU-4 scores.[6] Table 4 shows that the reranked question selected using our query well-formedness classifier improves the BLEU-4 score of a seq-to-seq question generation model from 12.0 to 12.2. The oracle improvement, by selecting the sentence from the list that maximizes the BLEU-4 score is 15.2. However, its worth noting that an increase in well-formedness doesn’t guarantee an improved BLEU score, as the oracle sentence maximizing the BLEU score might be fairly non-wellformed (Callison-Burch et al., 2006). For example, *“who was elected the president of notre dame in?”* has a higher BLEU score to the reference *“who was the president of notre dame in 1934?”* than our well-formed question *“who was elected the president of notre dame?”*. Figure 3 shows a question generation example with the output of Du et al. (2017) as the baseline result and the reranked question using the wellformed probability.

Model	BLEU-1	BLEU-4
Baseline	41.3	12.0
Reranked	**41.6**	**12.2**

Table 4: Reranking the n-best output of a neural seq2seq question generation model using well-formedness probability.

Sentence: montana is home to the rocky mountain elk foundation and has a historic big game hunting tradition.
Gold question: what is the name of the big game hunting foundation in montana?

seq2seq: what is a historic big game hunting tradition? ($p_{wf} = 0.7$)
Reranked: what is the name of the historic big game tradition? ($p_{wf} = 0.8$)

Figure 3: Example showing question selection from the n-best list using our reranking model.

4 Related Work

We have referenced much of the related work throughout the paper. We now review another orthogonally related field of work. Grammatical error correction (GEC) is the task of correcting the grammatical errors (if any) in a piece of text (Ng et al., 2014). As GEC includes not just identification of ungrammatical text but also correcting the text to produce grammatical text, its a more complex task. However, grammatical error prediction (Schmaltz et al., 2016; Daudaravicius et al., 2016) is the task of classifying whether or not a sentence is grammatical, which is more closely related to

---

[6] BLEU-x uses precision computed over $[1, x]$-grams.

our task as classifying a question as well-formed requires making judgement on both the style and grammar of the text.

5 Conclusion

We proposed a new task of well-formed natural language question identification and established a strong baseline on a new dataset that can be downloaded at: `http://goo.gl/language/query-wellformedness`. We also showed that question well-formedness information can be a helpful signal in improving state-of-the-art question generation systems.

References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proc. of EMNLP*.

Ricardo Baeza-Yates, Liliana Calderón-Benavides, and Cristina González-Caro. 2006. The intention behind web queries. In *Proc. of SPIRE*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of English web-search queries. In *Proc. of EMNLP*.

Mayank Bawa, Roberto J. Bayardo, Jr., Sridhar Rajagopalan, and Eugene J. Shekita. 2003. Make it fresh, make it quick: Searching a network of personal webservers. In *Proc. of WWW*.

Shane Bergsma and Qin Iris Wang. 2007. Learning noun phrase query segmentation. In *Proc. of EMNLP-CoNLL*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proc. of EMNLP*.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *Proc. of ICLR*.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of bleu in machine translation research. In *Proc. of EACL*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *CoRR*, abs/1504.00325.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Comput. Linguist.*, 31(1):25–70.

Ann A Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage english grammar using HPSG. In *Proc. of LREC*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges.*, pages 177–190. Springer.

Vidas Daudaravicius, Rafael E. Banchs, Elena Volodina, and Courtney Napoles. 2016. A report on the automatic evaluation of scientific writing shared task. In *Proc. of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proc. of ACL*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proc. of ACL*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proc. of AISTATS*.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Proc. of NAACL*.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proc. of NAACL*.

Mehdi Manshadi and Xiao Li. 2009. Semantic tagging of web search queries. In *Proc. of ACL-IJCNLP*.

Evangelos P. Markatos. 2001. On caching search engine query results. *Computer Communications*, 24(2):137–143.

Nikita Mishra, Rishiraj Saha Roy, Niloy Ganguly, Srivatsan Laxman, and Monojit Choudhury. 2011. Unsupervised query segmentation using only query logs. In *Proc. of WWW*.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proc. of ACL*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *CoNLL Shared Task*.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *Proc. of EMNLP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. of ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proc. of EMNLP*.

Rishiraj Saha Roy, Monojit Choudhury, and Kalika Bali. 2012. Are web search queries an evolving protolanguage. In *In Proc. of Evolang*.

Allen Schmaltz, Yoon Kim, Alexander M. Rush, and Stuart Shieber. 2016. Sentence-level grammatical error identification as sequence-to-sequence correction. In *Proc. of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural paraphrase identification of questions with noisy pretraining. In *Proc. of First Workshop on Subword and Character Level Models in NLP*.

Lucy Vanderwende. 2008. The importance of being important: Question generation. In *Proc. of the Workshop on the Question Generation Shared Task and Evaluation Challenge*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proc. of IJCAI*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *TACL*, 3:345–358.

Jie Yang, Claudia Hauff, Alessandro Bozzon, and Geert-Jan Houben. 2014. Asking the right question in collaborative QA systems. In *Proc. of Conference on Hypertext and Social Media*.

Self-Governing Neural Networks for On-Device Short Text Classification

Sujith Ravi
Google Research
Mountain View, CA, USA
sravi@google.com

Zornitsa Kozareva
Google
Mountain View, CA, USA
zornitsa@kozareva.com

Abstract

Deep neural networks reach state-of-the-art performance for wide range of natural language processing, computer vision and speech applications. Yet, one of the biggest challenges is running these complex networks on devices such as mobile phones or smart watches with tiny memory footprint and low computational capacity. We propose on-device Self-Governing Neural Networks (SGNNs), which learn compact projection vectors with local sensitive hashing. The key advantage of SGNNs over existing work is that they surmount the need for pre-trained word embeddings and complex networks with huge parameters. We conduct extensive evaluation on dialog act classification and show significant improvement over state-of-the-art results. Our findings show that SGNNs are effective at capturing low-dimensional semantic text representations, while maintaining high accuracy.

1 Introduction

Deep neural networks are one of the most successful machine learning methods outperforming many state-of-the-art machine learning methods in natural language processing (Sutskever et al., 2014), speech (Hinton et al., 2012) and visual recognition tasks (Krizhevsky et al., 2012). The availability of high performance computing has enabled research in deep learning to focus largely on the development of deeper and more complex network architectures for improved accuracy. However, the increased complexity of the deep neural networks has become one of the biggest obstacles to deploy deep neural networks on-device such as mobile phones, smart watches and IoT (Iandola et al., 2016). The main challenges with developing and deploying deep neural network models on-device are (1) the tiny memory footprint, (2) inference latency and (3) significantly low computational capacity compared to high performance computing systems such as CPUs, GPUs and TPUs on the cloud.

There are multiple strategies to build lightweight text classification models for on-device. One can create a small dictionary of common *input* → *category* mapping on the device and use a naive look-up at inference time. However, such an approach does not scale to complex natural language tasks involving rich vocabularies and wide language variability. Another strategy is to employ fast sampling techniques (Ahmed et al., 2012; Ravi, 2013) or incorporate deep learning models with graph learning like (Bui et al., 2017, 2018), which result in large models but have proven to be extremely powerful for complex language understanding tasks like response completion (Pang and Ravi, 2012) and Smart Reply (Kannan et al., 2016).

In this paper, we propose Self-Governing Neural Networks (SGNNs) inspired by projection networks (Ravi, 2017). SGNNs are on-device deep learning models learned via embedding-free projection operations. We employ a modified version of the locality sensitive hashing (LSH) to reduce input dimension from millions of unique words/features to a short, fixed-length sequence of bits. This allows us to compute a projection for an incoming text very fast, on-the-fly, with a small memory footprint on the device since we do not need to store the incoming text and word embeddings. We evaluate the performance of our SGNNs on Dialogue Act classification, because (1) it is an important step towards dialog interpretation and conversational analysis aiming to understand the intent of the speaker at every utterance of the conversation and (2) deep learning methods reached state-of-the-art (Lee and Dernoncourt, 2016; Khanpour et al., 2016; Tran et al., 2017; Ortega and Vu, 2017).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 804–810
Brussels, Belgium, October 31 - November 4, 2018.

The main contributions of the paper are:

- Novel Self-Governing Neural Networks (SGNNs) for on-device deep learning for short text classification.

- Compression technique that effectively captures low-dimensional semantic text representation and produces compact models that save on storage and computational cost.

- On the fly computation of projection vectors that eliminate the need for large pre-trained word embeddings or vocabulary pruning.

- Exhaustive experimental evaluation on dialog act datasets, outperforming state-of-the-art deep CNN (Lee and Dernoncourt, 2016) and RNN variants (Khanpour et al., 2016; Ortega and Vu, 2017).

2 Self-Governing Neural Networks

We model the Self-Governing network using a *projection* model architecture (Ravi, 2017). The *projection* model is a simple network with dynamically-computed layers that encodes a set of efficient-to-compute operations which can be performed directly on device for inference. The model defines a set of efficient "projection" functions $\mathbb{P}(\vec{x}_i)$ that project each input instance $\vec{x}_i$ to a different space $\Omega_{\mathbb{P}}$ and then performs learning in this space to map it to corresponding outputs y_i^p. A very simple projection model comprises just few operations where the inputs $\vec{x}_i$ are transformed using a series of T projection functions $\mathbb{P}^1, ..., \mathbb{P}^T$ followed by a single layer of activations.

2.1 Model Architecture

In this work, we design a Self-Governing Neural Network (SGNN) using multi-layered locality-sensitive projection model. Figure 1 shows the model architecture of the on-device SGNN network. The *self-governing* property of this network stems from its ability to learn a model (e.g., a classifier) without having to initialize, load or store any feature or vocabulary weight matrices. In this sense, our method is a truly **embedding-free** approach unlike majority of the widely-used state-of-the-art deep learning techniques in NLP whose performance depends on embeddings pre-trained on large corpora. Instead, we use the projection functions to dynamically transform each input to a low-dimensional representation. Furthermore, we stack this with additional layers and non-linear activations to achieve deep, non-linear combinations of projections that permit the network to learn complex mappings from inputs x_i to outputs y_i. An SGNN network is shown below:

$$\begin{aligned} i_p &= [\mathbb{P}^1(x_i), ..., \mathbb{P}^T(x_i)] && (1) \\ h_p &= \sigma(W_p \cdot i_p + b_p) && (2) \\ h_t &= \sigma(W_t \cdot h_{t-1} + b_t) && (3) \\ y_i &= \texttt{softmax}(W_o \cdot h_k + b_o) && (4) \end{aligned}$$

where, i_p refers to the output of projection operation applied to input x_i, h_p is applied to projection output, h_t is applied at intermediate layers of the network with depth k followed by a final `softmax` activation layer at the top. In a k-layer SGNN, h_t, where $t = p, p+1, ..., p+k-1$ refers to the k subsequent layers after the projection layer. W_p, W_t, W_o and b_p, b_t, b_o represent trainable weights and biases respectively.

The projection transformations use pre-computed parameterized functions, i.e., they are not trained during the learning process, and their outputs are concatenated to form the hidden units for subsequent operations. Each input text x_i is converted to an intermediate feature vector (via raw text features such as skip-grams) followed by projections.

$$x_i \xrightarrow{\mathbb{F}} \vec{x}_i \xrightarrow{\mathbb{P}} [\mathbb{P}^1(x_i), ..., \mathbb{P}^T(x_i)] \quad (5)$$

On-the-fly Computation. The transformation step $\mathbb{F}$ dynamically extracts features from the raw input. Text features (e.g., skip-grams) are converted into feature-ids f_j (via hashing) to generate a sparse feature representation $\vec{x}_i$ of feature-id, weight pairs (f_j, w_j) . This intermediate feature representation is passed through projection functions $\mathbb{P}$ to construct projection layer i_p in SGNN. For this last step, a projection vector $\mathbb{P}^k$ is first constructed on-the-fly using a hash function with feature ids f_j in $\vec{x}_i$ and fixed seed as input, then dot product of the two vectors $< \vec{x}_i, \mathbb{P}^k >$ is computed and transformed into binary representation $\mathbb{P}^k(\vec{x}_i)$ using $sgn(.)$ of the dot product.

As shown in Figure 1, both $\mathbb{F}$ and $\mathbb{P}$ steps are computed on-the-fly, i.e., no word-embedding or vocabulary/feature matrices need to be stored and looked up during training or inference. Instead feature-ids and projection vectors are dynamically computed via hash functions. For intermediate feature weights w_j, we use observed counts in

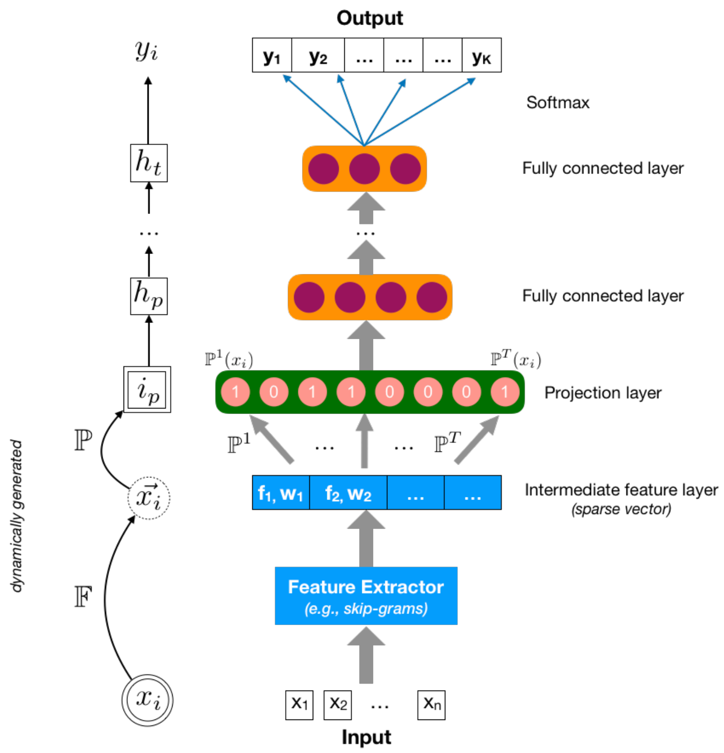

Figure 1: Self-Governing Neural Network (SGNN) architecture.

each input text and do not use pre-computed statistics like idf. Hence the method is embedding-free.

Model Optimization. The SGNN network is trained from scratch on the task data using a supervised loss defined wrt ground truth $\hat{y}_i$:

$$\mathcal{L}(.) = \sum_{i \in N} \texttt{cross} - \texttt{entropy}(y_i, \hat{y}_i) \quad (6)$$

During training, the network learns to choose and apply specific projection operations $\mathbb{P}^j$ (via activations) that are more predictive for a given task. The choice of the type of projection matrix $\mathbb{P}$ as well as representation of the projected space $\Omega_{\mathbb{P}}$ has a direct effect on computation cost and model size. We leverage an efficient randomized projection method and use a binary representation $\{0,1\}^d$ for $\Omega_{\mathbb{P}}$. This yields a drastically lower memory footprint both in terms of number and size of parameters.

Computing Projections. We employ an efficient randomized projection method for the projection step. We use locality sensitive hashing (LSH) (Charikar, 2002) to model the underlying projection operations in SGNN. LSH is typically used as a dimensionality reduction technique for clustering (Manning et al., 2008). LSH allows us to project similar inputs $\vec{x}_i$ or intermediate network layers into hidden unit vectors that are nearby in metric space. We use repeated binary hashing for $\mathbb{P}$ and apply the projection vectors to transform the input $\vec{x}_i$ to a binary hash representation denoted by $\mathbb{P}_k(\vec{x}_i) \in \{0,1\}$, where $[\mathbb{P}_k(\vec{x}_i)] := sgn[\langle \vec{x}_i, \mathbb{P}_k \rangle]$. This results in a d-bit vector representation, one bit corresponding to each projection row $\mathbb{P}_{k=1...d}$.

The same projection matrix $\mathbb{P}$ is used for training and inference. We never need to explicitly store the random projection vector $\mathbb{P}_k$ since we can compute them on the fly using hash functions over feature indices with a fixed row seed rather than invoking a random number generator. This also permits us to perform projection operations that are linear in the *observed* feature size rather than the *overall* feature or vocabulary size which can be prohibitively large for high-dimensional data, thereby saving both memory and computation cost. Thus, SGNN can efficiently model high-dimensional sparse inputs and large vocabulary sizes common for text applications instead of relying on feature pruning or other pre-processing heuristics employed to restrict input sizes in standard neural networks for feasible training. The binary representation is significant since this results in a significantly compact representation for the *projection* network parameters that in turn considerably reduces the model size.

SGNN Parameters. In practice, we employ T different projection functions $\mathbb{P}^{j=1...T}$, each resulting in d-bit vector that is concatenated to form the projected vector i_p in Equation 5. T and d vary depending on the *projection* network parameter configuration specified for $\mathbb{P}$ and can be tuned to trade-off between prediction quality and model size. Note that the choice of whether to use a single projection matrix of size $T \cdot d$ or T separate matrices of d columns depends on the type of projection employed (dense or sparse). For the intermediate feature step $\mathbb{F}$ in Equation 5, we use skip-gram features (3-grams with skip-size=2) extracted from raw text.

2.2 Training and Inference

We use the compact bit units to represent the *projection* in SGNN. During training, the network learns to move the gradients for points that are nearby to each other in the projected bit space $\Omega_{\mathbb{P}}$ in the same direction. SGNN network is trained end-to-end using backpropagation. Training can progress efficiently with stochastic gradient descent with distributed computing on high-performance CPUs or GPUs.

Complexity. The overall complexity for SGNN inference, governed by the projection layer, is $O(n \cdot T \cdot d)$, where n is the observed feature size (*not* overall vocabulary size) which is linear in input size, d is the number of LSH bits specified for each projection vector $\mathbb{P}_k$, and T is the number of projection functions used in $\mathbb{P}$. The model size (in terms of number of parameters) and memory storage required for the *projection* inference step is $O(T \cdot d \cdot C)$, where C is the number of hidden units in h_p in the multi-layer projection network and typically smaller than $T \cdot d$.

3 Datasets and Experimental Setup

3.1 Data Description

We conduct our experimental evaluation on two dialog act benchmark datasets.

- **SWDA:** Switchboard Dialog Act Corpus (Godfrey et al., 1992; Jurafsky et al., 1997) is a popular open domain dialogs corpus between two speakers with 42 dialogs acts.
- **MRDA:** ICSI Meeting Recorder Dialog Act Corpus (Adam et al., 2003; Shriberg et al., 2004) is a dialog corpus of multiparty meetings with 5 tags of dialog acts.

Datasets	Class	Vocab.	Train	Validation	Test
SwDA	42	20K	193K	23K	5K
MRDA	5	12K	78K	16K	15K

Table 1: Dialog Act Datasets Statistics

Table 1 summarizes dataset statistics. We use the train, validation and test splits as defined in (Lee and Dernoncourt, 2016; Ortega and Vu, 2017).

3.2 Experimental Setup

We setup our experimental evaluation, as follows: given a classification task and a dataset, we generate an on-device model. The size of the model can be configured (by adjusting the projection matrix $\mathbb{P}$) to fit in the memory footprint of the device, i.e. a phone has more memory compared to a smart watch. For each classification task, we report $Accuracy$ on the test set.

3.3 Hyperparameter and Training

For both datasets we used the following: 2-layer SGNN ($\mathbb{P}_{T=80,d=14} \times \text{FullyConnected}_{256} \times \text{FullyConnected}_{256}$), mini-batch size of 100, dropout rate of 0.25, learning rate was initialized to 0.025 with cosine annealing decay (Loshchilov and Hutter, 2016). Unlike prior approaches (Lee and Dernoncourt, 2016; Ortega and Vu, 2017) that rely on pre-trained word embeddings, we learn the projection weights on the fly during training, i.e word embeddings (or vocabularies) do not need to be stored. Instead, features are computed on the fly and are dynamically compressed via the projection matrices into projection vectors. These values were chosen via a grid search on development sets, we do not perform any other dataset-specific tuning. Training is performed through stochastic gradient descent over shuffled mini-batches with Nesterov momentum optimizer (Sutskever et al., 2013), run for 1M steps.

4 Results

Tables 2 and 3 show results on the SwDA and MRDA dialog act datasets. Overall, our SGNN model consistently outperforms the baselines and prior state-of-the-art deep learning models.

4.1 Baselines

We compare our model against a majority class baseline and Naive Bayes classifier (Lee and Dernoncourt, 2016). Our model significantly outperforms both baselines by 12 to 35% absolute.

4.2 Comparison against State-of-art Methods

We also compare our performance against prior work using HMMs (Stolcke et al., 2000) and recent deep learning methods like CNN (Lee and Dernoncourt, 2016), RNN (Khanpour et al., 2016) and RNN with gated attention (Tran et al., 2017).

To the best of our knowledge, (Lee and Dernoncourt, 2016; Ortega and Vu, 2017; Tran et al., 2017) are the latest approaches in dialog act classification, which also reported on the same data splits. Therefore, we compare our research against these works. According to (Ortega and Vu, 2017), prior work by (Ji and Bilmes, 2006) achieved promising results on the MRDA dataset, but since the evaluation was conducted on a different data split, it is hard to compare them directly.

For both SwDA and MRDA datasets, our SGNNs obtains the best result of 83.1 and 86.7 accuracy outperforming prior state-of-the-art work. This is very impressive given that we work with very small memory footprint and we do not rely on pre-trained word embeddings. Our study also shows that the proposed method is very effective for such natural language tasks compared to more complex neural network architectures such as deep CNN (Lee and Dernoncourt, 2016) and RNN variants (Khanpour et al., 2016; Ortega and Vu, 2017). We believe that the compression techniques like locality sensitive projections jointly coupled with non-linear functions are effective at capturing low-dimensional semantic text representations that are useful for text classification applications.

4.3 Discussion on Model Size and Inference

LSTMs have millions of parameters, while our on-device architecture has just 300K parameters (order of magnitude lower). Most deep learning methods also use large vocabulary size of 10K or higher. Each word embedding is represented as 100-dimensional vector leading to a storage requirement of $10,000 \times 100$ parameter weights just in the first layer of the deep network. In contrast, SGNNs in all our experiments use a fixed 1120-dimensional vector regardless of the vocabulary or feature size, dynamic computation results in further speed up for high-dimensional feature spaces. This amounts to a huge savings in storage and computation cost wrt FLOPs (floating point operations per second).

Method	Acc.
Majority Class (baseline) (Ortega and Vu, 2017)	33.7
Naive Bayes (baseline) (Khanpour et al., 2016)	47.3
HMM (Stolcke et al., 2000)	71.0
DRLM-conditional training (Ji and Bilmes, 2006)	77.0
DRLM-joint training (Ji and Bilmes, 2006)	74.0
LSTM (Lee and Dernoncourt, 2016)	69.9
CNN (Lee and Dernoncourt, 2016)	73.1
Gated-Attention&HMM (Tran et al., 2017)	74.2
RNN+Attention (Ortega and Vu, 2017)	73.8
RNN (Khanpour et al., 2016)	80.1
SGNN: Self-Governing Neural Network (ours)	**83.1**

Table 2: SwDA Dataset Results

Method	Acc.
Majority Class (baseline)(Ortega and Vu, 2017)	59.1
Naive Bayes (baseline) (Khanpour et al., 2016)	74.6
Graphical Model (Ji and Bilmes, 2006)	81.3
CNN (Lee and Dernoncourt, 2016)	84.6
RNN+Attention(Ortega and Vu, 2017)	84.3
RNN (Khanpour et al., 2016)	**86.8**
SGNN: Self-Governing Neural Network (ours)	**86.7**

Table 3: MRDA Dataset Results

5 Conclusion

We proposed Self-Governing Neural Networks for on-device short text classification. Experiments on multiple dialog act datasets showed that our model outperforms state-of-the-art deep leaning methods (Lee and Dernoncourt, 2016; Khanpour et al., 2016; Ortega and Vu, 2017). We introduced a compression technique that effectively captures low-dimensional semantic representation and produces compact models that significantly save on storage and computational cost. Our approach does not rely on pre-trained embeddings and efficiently computes the projection vectors on the fly. In the future, we are interested in extending this approach to more natural language tasks. For instance, we built a multilingual SGNN model for customer feedback classification (Liu et al., 2017) and obtained 73% on Japanese, close to best performing system on the challenge (Plank, 2017). Unlike their method, we did not use any pre-processing, tagging, parsing, pre-trained embeddings or other resources.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback and suggestions.

References

Janin Adam, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. 2003. The icsi meeting corpus. In *Proceedings of the 5TH SIGdial Workshop on Discourse and Dialogue*, pages 364–367.

Amr Ahmed, Sujith Ravi, Alex J. Smola, and Shravan M. Narayanamurthy. 2012. Fastex: Hash clustering with exponential families. In *Advances in Neural Information Processing Systems 25*, pages 2798–2806.

Thang D. Bui, Sujith Ravi, and Vivek Ramavajjala. 2017. Neural graph machines: Learning neural networks using graphs. *CoRR*, abs/1703.04818.

Thang D. Bui, Sujith Ravi, and Vivek Ramavajjala. 2018. Neural graph learning: Training neural networks using graphs. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 64–71.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, New York, NY, USA. ACM.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ICASSP'92, pages 517–520. IEEE Computer Society.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*.

Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360.

Gang Ji and Jeff Bilmes. 2006. Backoff model training using partially observed data: Application to dialog act tagging. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 280–287.

Daniel Jurafsky, Rebecca Bates, Rachel Martin Noah Coccaro, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Audreas Stolcke, Paul Taylor, and Van Ess-Dykema. 1997. Automatic detection of discourse structure for speech recognition and understanding. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 88–95.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 955–964.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2012–2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520.

Chao-Hong Liu, Yasufumi Moriya, Alberto Poncelas, and Declan Groves. 2017. Ijcnlp-2017 task 4: Customer feedback analysis. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 26–33, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Ilya Loshchilov and Frank Hutter. 2016. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Daniel Ortega and Ngoc Thang Vu. 2017. Neural-based context representation learning for dialog act classification. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 247–252.

Bo Pang and Sujith Ravi. 2012. Revisiting the predictability of language: Response completion in social media. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1489–1499.

Barbara Plank. 2017. All-in-1 at ijcnlp-2017 task 4: Short text classification with one model for all languages. In *Proceedings of the IJCNLP 2017, Shared*

Tasks, pages 143–148, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Sujith Ravi. 2013. Scalable decipherment for machine translation via hash sampling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 362–371. Association for Computational Linguistics.

Sujith Ravi. 2017. Projectionnet: Learning efficient on-device deep networks using neural projections. *CoRR*, abs/1708.00630.

Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The icsi meeting recorder dialog act (mrda) corpus. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*, pages 97–100, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Comput. Linguist.*, 26(3):339–373.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1139–III–1147.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112.

Quan Hung Tran, Gholamreza Haffari, and Ingrid Zukerman. 2017. A generative attentional neural network model for dialogue act classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 524–529.

HFT-CNN: Learning Hierarchical Category Structure for Multi-label Short Text Categorization

Kazuya Shimura[1], **Jiyi Li**[2] **and Fumiyo Fukumoto**[2]
Graduate School of Engineering[1]
Interdisciplinary Graduate School[2]
University of Yamanashi
4-3-11, Takeda, Kofu, 400-8511 Japan
`{g17tk008,jyli,fukumoto}@yamanashi.ac.jp`

Abstract

We focus on the multi-label categorization task for short texts and explore the use of a hierarchical structure (HS) of categories. In contrast to the existing work using non-hierarchical flat model, the method leverages the hierarchical relations between the categories to tackle the data sparsity problem. The lower the HS level, the worse the categorization performance. Because lower categories are fine-grained and the amount of training data per category is much smaller than that in an upper level. We propose an approach which can effectively utilize the data in the upper levels to contribute categorization in the lower levels by applying a Convolutional Neural Network (CNN) with a fine-tuning technique. The results using two benchmark datasets show that the proposed method, Hierarchical Fine-Tuning based CNN (HFT-CNN) is competitive with the state-of-the-art CNN based methods.

1 Introduction

Short text categorization is widely studied since the recent explosive growth of online social networking applications (Song et al., 2014). In contrast with documents, short texts are less topic-focused in texts. Major attempts to tackle the problem is to expand short texts with knowledge extracted from the textual corpus, machine-readable dictionaries, and thesauri (Phan et al., 2008; Wang et al., 2008; Chen et al., 2011; Wu et al., 2012). However, because of domain-independent nature of dictionaries and thesauri, it is often the case that the data distribution of the external knowledge is different from the test data collected from some specific domain, which deteriorates the overall performance of categorization. A methodology which maximizes the impact of pre-defined domains/categories is needed to improve categorization performance.

More recently, many authors have attempted to apply deep learning techniques including CNN (Wang et al., 2015; Zhang and Wallace, 2015; Zhang et al., 2017; Wang et al., 2017), the attention based CNN (Yang et al., 2016), bag-of-words based CNN (Johnson and Zhang, 2015a), and the combination of CNN and recurrent neural network (Lee and Dernoncourt, 2016; Zhang et al., 2016) to text categorization. Most of them demonstrated that neural network models are powerful for learning features from texts, while they focused on single-label or a few labels problem. Several efforts have been made to multi-labels (Johnson and Zhang, 2015b; Liu et al., 2017). Liu *et al.* explored a family of new CNN models which are tailored for extreme multi-label classification (Liu et al., 2017). They used a dynamic max pooling scheme, a binary cross-entropy loss, and a hidden bottleneck layer to improve the overall performance. The results by using six benchmark datasets where the label-set sizes are up to 670K showed that their method attained at the best or second best in comparison with seven state-of-the-art methods including FastText (Joulin et al., 2017) and bag-of-words based CNN (Johnson and Zhang, 2015a). However, all of these attempts aimed at utilizing a large volume of data.

We address the problem of multi-label short text categorization and explore the use of a HS of categories. The lower level of categories are fine-grained compared to the upper level of categories. Moreover, it is often the case that the amount of training data in a lower level is much smaller than that in an upper level which deteriorates the overall performance of categorization. We propose an approach which can effectively utilize the data in the upper levels to contribute categorization in lower levels by applying fine-tuning to the CNN which can learn a HS of categories and incorporate

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 811–816
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

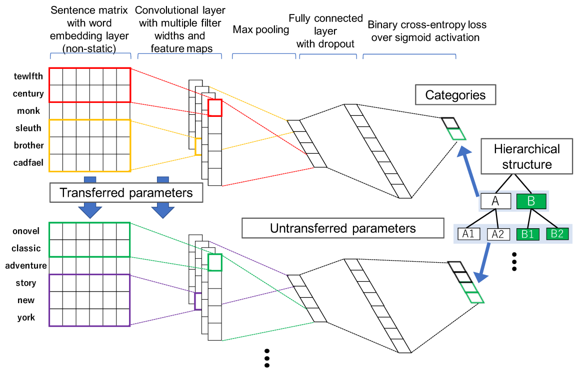

Figure 1: HFT-CNN model

granularity of categories into categorization. We transferred the parameters of CNN trained from upper to lower levels according to the HS, and finely tuned parameters. The main contributions of our work can be summarized: (1) We propose a method that maximizes the impact of pre-defined categories to alleviate data sparsity in multi-label short texts. (2) We empirically examined a fine-tuning with CNN that fits to learn a HS of categories defined by lexicographers, and (3) The results show that our method is competitive to the state-of-the-art CNN based methods by using two benchmark datasets, especially it is effective for categorization of short texts consisting of a few words with a large number of labels.

2 Hierarchical Fine-Tuning based CNN

2.1 CNN architecture

Similar to other CNN (Johnson and Zhang, 2015a; Liu et al., 2017), our HFT-CNN model shown in Figure 1 is based on (Kim, 2014). Let $\mathbf{x}_i \in \mathbb{R}^k$ be the k-dimensional word vector with the i-th word in a sentence obtained by applying skip-gram model provided in fastText[1]. A sentence with length n is represented as $\mathbf{x}_{1:n} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{nk}$. A convolution filter $\mathbf{w} \in \mathbb{R}^{hk}$ is applied to a window size of h words to produce a new feature, $c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$ where $b \in \mathbb{R}$ indicates a bias term and f refers to a non-linear activation function. We applied this convolution filter to each possible window size in the sentence and obtained a feature map, $m \in \mathbb{R}^{n-h+1}$. As shown in Figure 1, we then apply a max pooling operation over the feature map and obtain the maximum value $\hat{m}$ as a feature of this filter. We obtained multiple filters by varying window sizes and multiple features. These features form a pooling layer and are passed to a fully connected layer. In the fully connected layer, we applied dropout (Hinton et al., 2012). The dropout randomly sets values in the layer to 0. Finally, we obtained the probability distribution over categories. The network is trained with the objective that minimizes the binary cross-entropy (BCE) of the predicted distributions and the actual distributions by performing stochastic gradient descent.

2.2 Hierarchical structure learning

Our key idea is to use a fine-tuning technique in CNN to tackle the data sparsity problem, especially a lower level of a HS. Following a HS, we transferred the parameters of CNN trained in the upper levels to the lower levels which are worse trained because of the lack of data, and then finely tuned parameters of CNN for lower levels (Figure 1). This approach can effectively utilize the data in the upper levels to contribute categorization in the lower levels.

Fine-tuning is motivated by the observation that the earlier features of CNN contain more generic features that should be effective for many tasks, but later layers of the CNN becomes progressively more specific to the details of the classes contained in the original dataset. The motivation is identical to a HS of categories as we first learn to distinguish among generic categories at the upper level of a hierarchy, then learns lower level distinctions by using only within the appropriate top level of the HS. We note that fine-tuning the last few layers are usually sufficient for transfer learning as the last few layers become more specific features. However, the HS consisting of deep level needs to fine-tune the early layers as well because the distance between the upper and lower level of categories is significant. For this reason, we transferred two layers shown in Figure 1, i.e., a layer obtained by word embedding and the convolutional layer. We used them as an initial parameter to learn the second level of a hierarchy. We repeated this procedure from the top level to the bottom level of a hierarchy. We note that a HS consists of many levels. We fine-tune between adjacent layers only because they are more correlated with each other compared to distant layers.

2.3 Multi-label categorization

Each test instance is classified into categories with probabilities/scores by applying HFT-CNN. We

---

[1] https://github.com/facebookresearch/fastText

Dataset	#L	Tr	Te	C
RCV1	4	23,149	781,265	103
Amazon670K	9	490,449	153,025	670,091

Table 1: Data Statistics: #L shows the depth of a hierarchy. Tr and Te refer to the # of training and test data, respectively. C indicates the total # of categories.

then utilize a constraint of a HS to obtain final results which differs from the existing work on non-hierarchical flat model (Johnson and Zhang, 2015a; Liu et al., 2017). This is done by using two scoring functions: One is a Boolean Scoring Function (BSF). Another is a Multiplicative Scoring Function (MSF). Both functions set a threshold value and categories whose scores exceed the threshold value are considered for selection. The difference is that BSF has a constraint that a category can only be selected if its ancestor categories are selected. MSF does not have such a constraint, i.e., we extracted all the categories whose scores exceeded the threshold value and sorted them in descending order as the system's assignments.

3 Experiments

3.1 Data and HFT-CNN model setting

We selected two benchmark datasets having a HS from the extreme classification repository[2]: RCV1 (Lewis et al., 2004) and Amazon670K (Leskovec and Krevl, 2015). All the documents in RCV1 and item descriptions in Amazon670K are tagged by using Tree Tagger (Schmid, 1995). We used nouns, verbs, and adjectives. We then applied fastText. Each dataset has an official training and test sets. We used each fold in the experiments. We choose titles from the training and test set on RCV1. The maximum number of words in the title was 13 words. Each text of Amazon670K consists of a product name and its item description. We extracted the first 13 words from each item description and used them in the experiments. Table 1 presents the statistics on the datasets. We divided the training data into two folds; we used 5% to tuning the parameters, and the remains to train the models. Our model setting is shown in Table 2[3]. In the experiments, we run three times for each model and obtained the averaged performance.

3.2 Evaluation Metrics

We used the standard F1 measure. Furthermore, we evaluated our method by two rank-based evaluation metrics: the precision at top k, P@k and the Normalized Discounted Cumulated Gains, NDCG@k which are commonly used for comparing extreme multi-label classification methods (Liu et al., 2017). We calculated P@k and NDCG@k for each test data and then obtained an average over all the test data.

3.3 Basic results

We compared HFT-CNN with a method which has hierarchical-based categorization but without fine-tuning (WoFT-CNN) and Flat model to examine the effect of the fine-tuning. WoFT-CNN shows that we independently trained parameters of CNN for each level and trained parameters are not transferred. Flat means that we simply applied our CNN model. The results are shown in Table 3. The HFT-CNN is better than WoFT-CNN and Flat model except for Micro-F1 obtained by WoFT-CNN(M) in Amazon670K. We also found that the overall results obtained by MSF were better to those obtained by BSF.

3.4 Comparison with state-of-the-art method

We chose XML-CNN as a comparative method because their method attained at the best or second best compared to the seven existing methods in six benchmark datasets (Liu et al., 2017). Original XML-CNN is implemented by using Theano[4], while we implemented HFT-CNN by Chainer[5]. In order to avoid the influence of differences in libraries, we implemented XML-CNN by Chainer and compared it with HFT-CNN. We used the author-provided implementation in Chainer's version of XML-CNN. We recall that we set convolutional filters with the window sizes to (2,3,4) and the stride size to 1 because of short text. To make a fair comparison, we also evaluated XML-CNN with the same window sizes and stride size as HFT-CNN.

Liu et al. evaluated their method by using P@k and NDCG@k. We used their metrics as well as F1 measure. We did not set a threshold value on BSF and MSF when we evaluated by using these metrics, but instead, we used a ranked list of cate-

---

[2] manikvarma.org/downloads/XC/XMLRepository.html

[3] Our source code including Chainer's version of XML-CNN is available at: HTTP://github.com/ShimShim46/HFT-CNN.

[4] https://drive.google.com/file/d/1Wwy1MNkrJRXZM3WNZNywa94c2-iEh_6U/view

[5] https://chainer.org

Description	Values	Description	Values
Input word vectors	fastText	Filter region size	(2,3,4)
Stride size	1	Feature maps (m)	128
Filters	128 × 3	Activation function	ReLu
Pooling	1-max pooling	Dropout	Randomly selected
Dropout rate1	0.25	Dropout rate2	0.5
Hidden layers	1,024	Batch sizes	100
Learning rate	Predicted by Adam	Epoch	40 with early stopping
Loss function	BCE loss over sigmoid activation	Threshold value for BSF and MSF	0.5

Table 2: HFT-CNN model settings: Dropout rate1 shows dropout immediately after embedding layer, and Dropout rate2 refers to dropout in a fully connected layer.

Metric F1	RCV1				
	HFT(B)	WoFT(B)	HFT(M)	WoFT(M)	Flat
Micro	79.87	79.69	**80.29**	80.06	79.51
Macro	50.31	49.59	**51.40**	50.64	47.71
Metrics	Amazon670K				
	HFT(B)	WoFT(B)	HFT(M)	WoFT(M)	Flat
Micro	49.74	50.12	∗50.94	**50.94**	49.10
Macro	6.78	6.37	**9.87**	8.68	5.73

Table 3: Basic results: (B) and (M) refer to a BSF and MSF, respectively. Bold font shows the best result within each line. The method marked with "∗" indicates the score is not statistically significant compared to the best one. We used a t-test, p-value < 0.05.

Metric	RCV1			
	HFT(B)	HFT(M)	XML(1)	XML(2)
P@1	92.60	**93.29**	92.93	92.55
P@3	∗77.56	**77.70**	77.18	76.80
P@5	∗53.96	**54.23**	53.85	53.60
G@1	92.60	**93.29**	92.93	92.55
G@3	∗88.47	**88.79**	88.16	87.75
G@5	89.37	**89.81**	89.19	88.80
Metric	Amazon670K			
	HFT(B)	HFT(M)	XML(1)	XML(2)
P@1	**86.54**	85.39	84.62	84.12
P@3	**66.25**	65.34	64.83	64.48
P@5	**51.09**	∗50.84	49.93	49.75
G@1	**86.54**	85.39	84.62	84.12
G@3	**76.26**	75.05	74.50	74.13
G@5	**72.84**	71.46	70.99	70.74

Table 4: Comparative results: "1" and "2" of XML show the stride size=1 and 2 by XML-CNN, respectively. "G" stands for NDCG.

gories assigned to the test instance. The results are shown in Table 4. HFT-CNN with BSF/MSF has the best scores with statistical significance compared to both of the XML-CNNs. On RCV1, HFT-CNN(B) in P@1 and NDCG@1 were worse than XML-CNN(1), while HFT-CNN(M) with the same metrics were statistically significant compared to XML-CNN(1). This is not surprising because hierarchical fine-tuning does not contribute to the accuracy at the top level as the trained parameters on the top level have not changed in the level.

We also examined the affection on each system performance by the depth of a hierarchical structure. Figure 2 shows Micro-F1 at each hierarchical level. The deeper the hierarchical level, the worse the system's performance. However, HFT-CNN is still better than XML-CNNs. The improvement by MSF was 1.00 $\sim$ 1.34% by Micro-F1 and 3.77 $\sim$ 10.07% by Macro-F1 on RCV1. On Amazon670K, the improvement was 1.10 $\sim$ 9.26% by Micro-F1 and 1.10 $\sim$ 3.60% by Macro-F1. This shows that hierarchical fine-tuning fits to learn the hierarchical category structure.

We recall that we focused on the multi-label problem. Figures 3 illustrates Micro-F1 and Macro-F1 against the number of categories per short text. We can see from RCV1 in Figure 3 that Micro-F1 obtained by HFT-CNN and XML-CNNs were not statistically significant difference in the number of categories, while Macro-F1 by HFT-CNN except for the number of 13 categories was constantly better to XML-CNNs. On Amazon670K data, when the number of categories assigned to the short text is less than 38, HFT-CNN was better than XML-CNNs or HFT-CNN was not statistically significant compared to XML-CNNs by both F1-scores. However, when it exceeds 39, HFT-CNN was worse than XML-CNNs. One possible reason is the use of BSF: a category can only be selected if its ancestor categories are selected. Therefore, once the test data could not be classified into categories correctly, their child categories also cannot be correctly assigned to the test data. In contrast, as shown in Figure 5, HFT-CNN by MSF was better than XML-CNNs in both Micro and Macro F1 even in the deep level of a hierarchy. From the observations, a robust scoring function is needed for further improvement.

It is important to note that how the ratio of training data affects the overall performance as we focused on the data sparsity problem. Figure

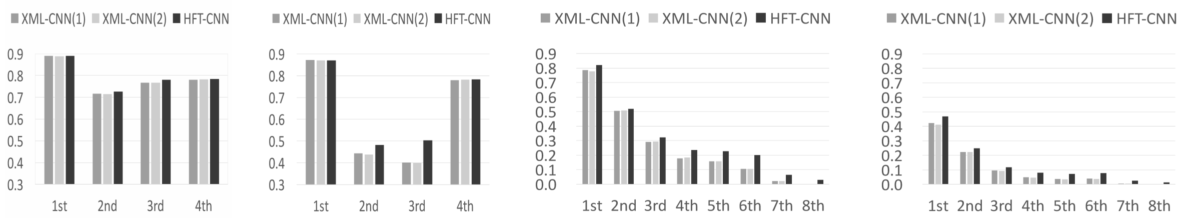

(a) Micro-F1 (RCV1) (b) Macro-F1(RCV1) (c) Micro-F1 (Amazon670K) (d) Macro-F1 (Amazon670K)

Figure 2: Performance in each hierarchical level: HFT-CNN used BSF

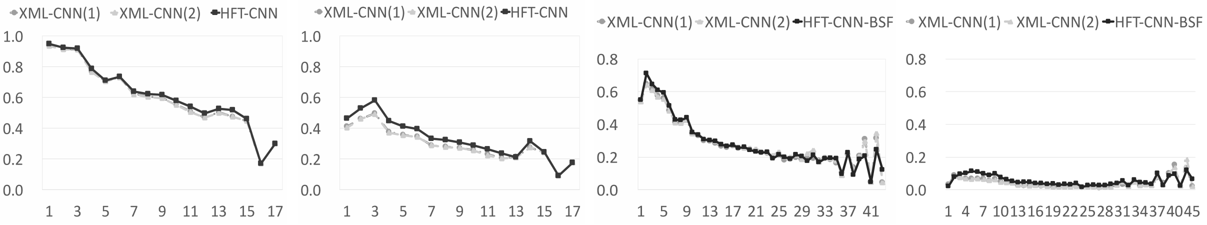

(a) Micro-F1 (RCV1) (b) Macro-F1 (RCV1) (c) Micro-F1 (Amazon670K) (d) Macro-F1 (Amazon670K)

Figure 3: Performance against the # of categories per short text: HFT-CNN used BSF

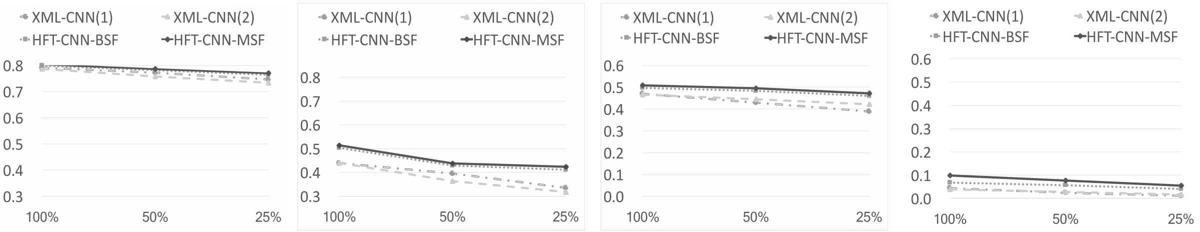

(a) Micro-F1 (RCV1) (b) Macro-F1 (RCV1) (c) Micro-F1 (Amazon670K) (d) Macro-F1 (Amazon670K)

Figure 4: Performance against a ratio of the training data

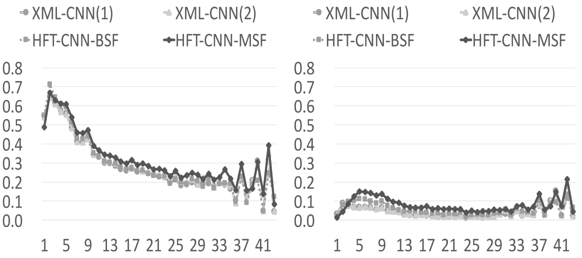

(a) Micro-F1 (Amazon670K) (b) Macro-F1 (Amazon670K)

Figure 5: Performance against the # of categories per short text: Comparison with HFT-CNN by MSF and other methods

4 shows Micro and Macro-F1 against a ratio of the training data. Overall, the curves show that more training helps the performance, while the curves obtained by HFT-CNN drop slowly compared to other methods in both datasets and evaluation metrics. From the observations mentioned in the above, we can conclude that fine-tuning works well, especially in the cases that the number of the training data per category is small.

4 Conclusion

We have presented an approach to multi-label categorization for short text. The comparative results with XML-CNN showed that HFT-CNN is competitive, especially for the cases that there exists only a small amount of training data. Future work will include: (i) incorporating lexical semantics such as named entities and domain-specific senses for further improvement, (ii) extending the method to utilize label dependency constraints (Bi and Kwok, 2011), and (iii) improving the accuracy of the top ranking categories to deal with P@1 and NDCG@1 metrics.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work is supported in part by Support Center for Advanced Telecommunications Technology Research, Foundation and the Grant-in-aid for the Japan Society for the Promotion of Science, No.17K00299.

References

W. Bi and J. T. Kwok. 2011. Multi-Label Classification on Tree- and DAG-Structured Hierarchies. In *Proc. of the 28th Internation Conference on Mathine Learning*, pages 17–24.

M. Chen, X. Jin, and D. Shen. 2011. Short Text Classification Improved by Learning Multi-Granularity Topics. In *Proc. of the 22nd International Joint Conference on Artificial Intelligence*, pages 1776–1781.

G. E. Hinton, N. Srivastava, A. Krizhevsky, H. Sutskever, and R. R. Salakhutdinov. 2012. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. In *arXiv preprint arXiv:1207.0580*.

R. Johnson and T. Zhang. 2015a. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. In *Proc of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112.

R. Johnson and T. Zhang. 2015b. Semi-Supervised Convolutional Neural Networks for Text Categorization vis Region Embedding. *Advances in Neural Information Processing Systems*, 28:919–927.

A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proc. of the 15th Conference of the European Chapter of the Association for Conputational Linguistics*, pages 427–431.

Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

J. Y. Lee and F. Dernoncourt. 2016. Sequential Short-Text classification with Recurrent and Convolutional Neural Networks. In *Proc. of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*, pages 515–520.

J. Leskovec and A. Krevl. 2015. SNAP Datasets: Stanford Large Network Dataset Collection.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5:361–397.

J. Liu, W-C. Chang, Y. Wu, and Y. Yang. 2017. Deep Learning for Extreme Multi-Label Text Classification. In *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124.

X. H. Phan, L. M. Nguyen, and S. Horiguchi. 2008. Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-Scale Data Collections. In *Proc. of the 17th International World Wide Web Conference*, pages 91–100.

H. Schmid. 1995. Improvements in Part-of-Speech Tagging with an Application to German. In *Proc. of the EACL SIGDAT Workshop*, pages 47–50.

G. Song, Y. Ye, X. Du, X. Huang, and S. Bie. 2014. Short Text Classification: A Survey. *Multimedia*, 9(5):635–643.

F. Wang, Z. Wang, Z. Li, and J-R. Wen. 2008. Concept-Based Short Text Classification and Ranking. In *Proc. of the 23rd ACM International Conference on Information and Knowledge Management*, pages 1069–1078.

J. Wang, Z. Wang, D. Zhang, and J. Yan. 2017. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In *Proc. of the 26th International Joint Conference on Artificial Intelligence*, pages 2915–2921.

P. Wang, J. Xu, B. Xu, C-L. Liu, H. Zhang, F. Wang, and H. Hao. 2015. Semantic Clustering and Convolutional Neural Network for Short Text Categorization. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 352–357.

W. Wu, H. Li, H. Wang, and K. Q. Zhu. 2012. A Probabilistic Taxonomy for Text Understanding. In *Proc. of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492.

Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *Proc. of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*, pages 1480–1489.

R. Zhang, H. Lee, and D. Radev. 2016. Dependency Sensitive Convolutional Neural Networks for Modeling Sentences and Documents. In *Proc. of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*, pages 1512–1521.

Y. Zhang, M. Lease, and B. C. Wallace. 2017. Exploiting Domain Knowledge via Grouped Weight Sharing with Application to Text Categorization. In *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 155–160.

Y. Zhang and B. C. Wallace. 2015. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. In *Proc. of the 8th International Joint Conference on Natural Language Processing*, pages 253–263.

A Hierarchical Neural Attention-based Text Classifier

Koustuv Sinha [1,2], Yue Dong [1,2], Jackie C.K. Cheung [1,2] and Derek Ruths [1]
[1] School of Computer Science, McGill University, Canada
[2] Montreal Institute of Learning Algorithms, Canada
{koustuv.sinha, yue.dong2, jcheung, derek.ruths }
@{mail.mcgill.ca, mail.mcgill.ca, cs.mcgill.ca, mcgill.ca}

Abstract

Deep neural networks have been displaying superior performance over traditional supervised classifiers in text classification. They learn to extract useful features automatically when sufficient amount of data is presented. However, along with the growth in the number of documents comes the increase in the number of categories, which often results in poor performance of the multiclass classifiers. In this work, we use external knowledge in the form of topic category taxonomies to aide the classification by introducing a deep hierarchical neural attention-based classifier. Our model performs better than or comparable to state-of-the-art hierarchical models at significantly lower computational cost while maintaining high interpretability.

1 Introduction

A large number of documents are being generated all over the world everyday, and as a result automatic text classification has become an essential tool for searching, retrieving, and managing the text (Allahyari et al., 2017). There has been an increasing trend in developing data-driven neural text classifiers (Collobert et al., 2011; Lai et al., 2015; Zhang et al., 2015; Yogatama et al., 2017; Conneau et al., 2017), due to their ability to handle large-scale corpora and their robustness in automatic feature extraction.

However, text classification has become increasingly challenging as the number of categories grows with continually expanding corpus. To alleviate this problem, one form of the external knowledge – class taxonomy – has been introduced to aid the classification in a hierarchical fashion (Koller and Sahami, 1997). In general, hierarchical classifiers can be categorized into two broad approaches: *local* (*top-down* and *bottom-up*) and *global* (or *big-bang*) (Silla and Freitas, 2011). The local approaches create a unique classifier for each parent node in the taxonomy (Liu et al., 2001; Quinn and Laier, 2006; Vens et al., 2008; Kowsari et al., 2017), while global approaches create a single classifier for the entire taxonomy (Silla Jr and Freitas, 2009).

Kowsari et al. (2017) recently proposed a hierarchical neural-based model called HDLTex, which displayed superior performance over traditional non-neural-based models with a top-down structure. However, HDLTex suffers the inherited disadvantage of the *top-down* approach: the number of sub-models grows exponentially with respect to the number of sub-trees. This is especially problematic in HDLTex, as it uses deep networks with a large number of parameters for the sub-models, and the combined model itself grows exponentially with the depth of taxonomy.

In contrast, we propose a unified *global* deep neural-based classifier that overcomes the problem of exploding models. The backbone of our approach is one encoder-decoder structure that sequentially predicts the class label of the next level, conditioned on a dynamic document representation obtained based on a variant of an attention mechanism (Bahdanau et al., 2015). The contribution of our paper is as follows:

1. We propose an end-to-end *global* neural attention-based model for hierarchical classification, which performs better than the state-of-the-art hierarchical classifier at lower computation cost.

2. We empirically show that the use of hierarchical taxonomy provides a robust classifier, by comparing with state-of-the-art flat classifiers.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 817–823
Brussels, Belgium, October 31 - November 4, 2018.

2 Literature Review

Traditional text classification methods focus on selecting a good set of features (for example,TF-IDF (Salton and Buckley, 1987)) to represent the documents and employing non-linear classifiers such as SVM (Dumais et al., 1998; Joachims, 1999; Tong and Koller, 2001), decision trees (Apté et al., 1994), or Naive Bayes (McCallum et al., 1998; Kim et al., 2006) methods for text classification. More recent work has employed deep neural networks to merge feature extraction and classification into one joint process, where the model parameters can be learned through back-propagation (Xue et al., 2008; Lai et al., 2015; Zhang et al., 2015). A common theme in these convolutional neural networks (CNN)-based or recurrent neural network (RNN)-based approaches is to create a document representation from either the last hidden state of the RNN or via some pooling operations on all hidden states.

Furthermore, the attention mechanism (Bahdanau et al., 2015; Sutskever et al., 2014) has been adapted for these CNN/RNN structures for text classification (Lin et al., 2017), providing high interpretability and allowing us to inspect which parts of the text are discriminative for a particular sample.

In addition, external knowledge has been examined as a way to boost the performance of text classifiers (Collobert and Weston, 2008a; Ngiam et al., 2011; Howard and Ruder, 2018). One form of external knowledge is built on top of the hierarchical relations of the classes (Koller and Sahami, 1997), where a class taxonomy is used to improve the performance of the end-level classification[1]. Most of the hierarchical classifiers[2] perform classification by navigating through the hierarchy in *top-down* approaches (Liu et al., 2001; Quinn and Laier, 2006; Vens et al., 2008), where a local classifier is constructed at each parent node. The state-of-the-art hierarchical classifier HDLTex is proposed by Kowsari et al. (2017). It combines deep neural networks in the *top-down* fashion where a separate neural network (either CNN or RNN) is built at each parent node to classify its children.

3 Model

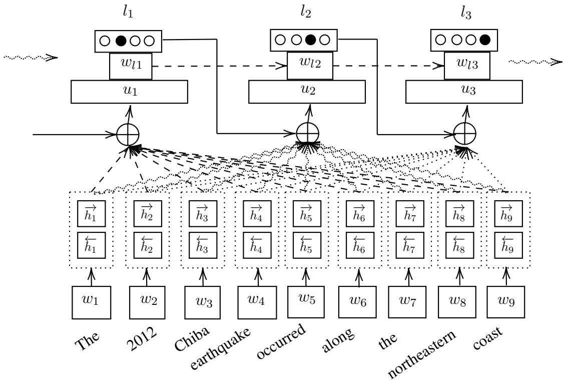

Figure 1: Proposed model architecture

Our proposed model (Figure 1) consists of three parts: 1) a bidirectional LSTM encoder (Hochreiter and Schmidhuber, 1997) that transforms each word into vector representations based on their context. 2) an attention module that helps to generate dynamic document representations across different level of classification, 3) multi-layer perceptron (MLP) classifiers at each level that makes the prediction of classes at that level based on the dynamically generated document representation and the level masking.

Our hierarchical classification model can be viewed as a sequence-to-sequence model, where a sequence of word embeddings is used to generate a sequence of hierarchical class labels. In addition, we employ a modified attention module from the traditional attention mechanism used in sequential generation tasks (Bahdanau et al., 2015; Sutskever et al., 2014). Instead of computing attention weights conditioned on the hidden state of the decoder at time step i, we condition on the parent category embedding c_{k-1}. This is intuitive in our setting as the document representation should depend on the parent class predicted by the model.

Formally, suppose we are given a document with n tokens $D = (w_1, w_2, ..., w_n)$ and its category labels of m levels $C = (c_1, \ldots, c_m), c_k \in \{c_1^{l_k}, \ldots, c_{s_k}^{l_k}\}$ where l_k indicates the k-th level of the class taxonomy and s_k represents the number of classes in level k [3]. A bidirectional LSTM is

---

[1] Classifiers that do not take into account the hierarchy and are only concerned with predicting the leaf nodes are termed *flat* classifiers in this work.

[2] We use the term "hierarchical classifiers" to refer the models that follow the external taxonomy of class labels, which is substantially different from hierarchical attention networks (Yang et al., 2016). In Yang et al. (2016), hierarchical attention networks refer to the hierarchical nature of their attention mechanism; the model attends to the sentences first and then attends to the words.

[3] We suppose w_i and c_i are word embeddings and class embedding respectively.

first used to extract features of the document:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(w_t, \overrightarrow{h_{t-1}}), \\ \overleftarrow{h_t} = \overleftarrow{LSTM}(w_t, \overleftarrow{h_{t+1}}). \tag{1}$$

The encoder's hidden states $H = (h_1, \ldots, h_n)$ are constructed by the concatenation of $(\overrightarrow{h_t})$ and $(\overleftarrow{h_t})$ as $h_i = [\overrightarrow{h_t}, \overleftarrow{h_t}]$.

When classifying the class label at level k, we first form contextual word features $\bar{H}_k$ by concatenating the previously predicted category embedding c_{k-1} (parent) with each of the encoder's outputs $H = (h_1, \ldots, h_n)$:

$$\bar{H}_k = H \oplus c_{k-1}. \tag{2}$$

Then, we transform these n vectors in $\bar{H}_k$ into n attention scores (scalars) through a series of linear and non-linear transformations:

$$a_k = \text{softmax}(w_{s2}\text{tanh}(W_{s_1}\bar{H}_k^T)). \tag{3}$$

As one single attention distribution might only focus on a specific component of the semantics in the document, we follow Lin et al. (2017)'s work to perform m hops of attention and form the multi-head attention matrix A_k $(m \times n)$. To encourage diversity over the multiple hops of the attention distributions, we employ the Frobenius norm penalty (Lin et al., 2017) $P = \left\| A_k A_k^\top - I \right\|_F^2$ to force the attention hops to focus on different aspects of the semantics.

The document representation for level k is obtained by multiplying the multi-head attention matrix and the contextual word features:

$$D_k = W_{s_3} A_k \bar{H}_k. \tag{4}$$

Finally, a two layered multi-layer perceptron (MLP) is employed to classify the category at level k:

$$d_k = \text{RELU}(W_D[D_k, d_k - 1]), \\ y_k = \text{softmax}(W_k d_k) \tag{5}$$

Normally, the softmax in Equation 5 is computed over all class labels across the entire taxonomy levels. This is not desirable when the taxonomy is deep and the number of classes is large. We solve this by employing a *level masking* technique where we mask out all the classes that are not in the current classification level k. The loss is then calculated as the joint cross entropy loss among all levels of the taxonomy: $l = \sum_{i=1}^{m} l_i$.

	DBpedia	WOS
Level 1 Categories	9	7
Level 2 Categories	70	134
Level 3 Categories	219	NA
Number of documents	381,025	46,985
Mean document length	106.9	200.7

Table 1: Dataset Comparison

4 Experimental Setup

Dataset Two datasets are used for our experiments: Web of Science (WOS) and DBpedia. Web of Science (WOS) is a hierarchical two-level taxonomy dataset that contains 46,985 documents collected from *Web of Science* (Reuters, 2012) by Kowsari et al. (2017). Despite its small size, WOS is used as a benchmark dataset for hierarchical classification as it provides the raw text for deep neural models to train on[4].

As deep learning models usually contain a large number of parameters that need to be learned, to prevent over-fitting (Lawrence et al., 1997; Srivastava et al., 2014) we usually need a large dataset to train upon. Thus, we curated a bigger dataset with hierarchical labels from Wikipedia meta information provider DBpedia[5]. Compared to WOS, our DBpedia dataset is larger in two aspects: the number of data instances and the number of hierarchical levels (Table 1). The DBpedia ontology was first used in Zhang et al. (2015) for flat text classification. We instead use the DBpedia ontology to construct a dataset with a three-level taxonomy of classes. In order to ensure enough documents per-class, we only extract leaf-classes with more than 200 documents. We also randomly subsample 3,000 documents per category to balance the number of leaf-level categories. This results in 381,025 documents in total, which we split into 90% for training (from which 10% were kept aside for validation) and 10% on testing, on which we report our classification metrics[6].

Baselines State-of-the-art *flat* classifiers such as FastText (Joulin et al., 2017), Bi-directional

---

[4] The LSHTC dataset (Partalas et al., 2015) has been widely used as a benchmark for hierarchical text classification. However, the raw texts are not available which makes it difficult to extract features for modern neural approaches. Instead, only the tf-idf vectors are provided as inputs with no option to retrieve the original text (even after consulting with the original authors we were unable to procure it).

[5] `http://wiki.dbpedia.org/`

[6] Our code and data will be released at `https://github.com/koustuvsinha/hier-class`

	DBpedia				WOS		
Flat Baselines				Overall			Overall
FastText				86.2			61.3
BiLSTM + MLP + Maxpool				94.20			**77.69**
BiLSTM + MLP + Meanpool				**94.68**			73.08
Structured Self Attention (m=1)				94.04			77.40
Hierarchical Models	l_1	l_2	l_3	Overall	l_1	l_2	Overall
HDLTex (5B params)	99.26	97.18	95.5	92.10	90.45	84.66	76.58
Our model (34M params)	99.21	96.03	95.32	**93.72**	89.32	82.42	**77.46**

Table 2: Test accuracy on the WOS and DBpedia datasets. The flat baseline models are trained without the hierarchical taxonomy of classes and therefore only have results on the leaf-node classification.

LSTM with max/mean pooling (Collobert and Weston, 2008b; Lee and Dernoncourt, 2016) and the Structured Self-attentive classifier (Lin et al., 2017) are used for the comparison. We noticed that using the default hyperparameters of the Structured Self-attentive classifier with high attention hops ($m >= 8$) performed poorly compared to use just one attention hop ($m = 1$). Therefore, we reported the results of using one attention hop ($m = 1$) as our baselines for fair comparison. We also compare our classifier to the state-of-the-art hierarchical classifier HDLTex (Kowsari et al., 2017).

Hyperparameters We use 300-dimensional word embeddings which are randomly initialized and fine-tuned during training. Two-layer Bidirectional LSTM with 300 hidden units in each layer are employed. In the multi-head attention mechanism, we use 4 heads (hops) with 0.1 Frobenius norm penalty because it gives the best validation performance. The final fully-connected MLP layer W_D has 1200 hidden units. In addition, we add 0.4 dropout on BiLSTM layers and MLP layers to prevent over-fitting.

For optimization, we use the standard Adam optimizer (Kingma and Ba, 2014) with the learning rate of 0.001, weight decay of 10^{-4} and 10^{-6} for WOS and DBpedia, respectively. The gradients are clipped to 0.5 in order to prevent exploding gradients. All the results are obtained after 25 epochs of training. After every 10 epochs, we reduce the learning rate by half if the validation accuracy is not improving. We employ early-stopping to select the best model. In addition, a weighted loss function is utilized to balance the performance on under-represented classes.

Hierarchical Evaluation For evaluating hierarchical models, we present the *teacher-forcing* result on each level, such as l_1, l_2 and l_3. This indicates the per-level classification performance when we provide the true parent class to the classifier while predicting the next class. However, this is not desirable as during inference we should not have access to the correct parent class. Hence we also present the *Overall* score in Table 2, where the classifier uses its own prediction as the parent class.

5 Results

Our model is significantly better than the existing state-of-the-art hierarchical baseline (Table 2). Although, we also see that both hierarchical classifiers (ours and HDLTex) perform comparably with or slightly worse than the state-of-the-art flat classifiers in terms of accuracy. However, the robustness analysis we performed in Table 3 indicates that hierarchical models are more robust in their errors since most of the errors generated by hierarchical classifiers remain within the correct tree of the parent class, while flat classifiers do worse. For example, on WOS, 88.57% of all classified data by our hierarchical model is within the correct subtree compared to 85.56% for the flat classifier.

Classifier	Correct parent	Predicted parent
Flat classifier - BiLSTM Max Pooling	90.74	85.56
Hierarchical approach - Our model	93.03	88.57

Table 3: Robustness analysis of taxonomy on the WOS dataset. We compare the success rate of our model and the BiLSTM flat classifier. The success rate is defined as the number of times the predicted class is within the same subtree as the correct parent. We calculate this in two scenarios: 1. when the true parent class is manually provided, or *teacher-forced* (Correct parent), and 2. when the true parent class is predicted by our model (Predicted parent)

Interestingly, the class taxonomy seems to be more beneficial in boosting the performance of hierarchical classifiers on WOS than DBpedia. The hierarchical classifiers perform better on the

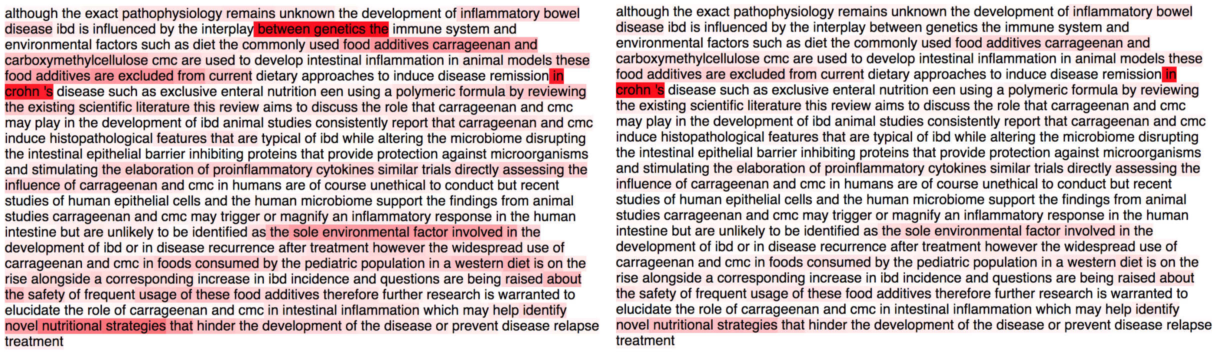

although the exact pathophysiology remains unknown the development of inflammatory bowel disease ibd is influenced by the interplay between genetics the immune system and environmental factors such as diet the commonly used food additives carrageenan and carboxymethylcellulose cmc are used to develop intestinal inflammation in animal models these food additives are excluded from current dietary approaches to induce disease remission in crohn 's disease such as exclusive enteral nutrition een using a polymeric formula by reviewing the existing scientific literature this review aims to discuss the role that carrageenan and cmc may play in the development of ibd animal studies consistently report that carrageenan and cmc induce histopathological features that are typical of ibd while altering the microbiome disrupting the intestinal epithelial barrier inhibiting proteins that provide protection against microorganisms and stimulating the elaboration of proinflammatory cytokines similar trials directly assessing the influence of carrageenan and cmc in humans are of course unethical to conduct but recent studies of human epithelial cells and the human microbiome support the findings from animal studies carrageenan and cmc may trigger or magnify an inflammatory response in the human intestine but are unlikely to be identified as the sole environmental factor involved in the development of ibd or in disease recurrence after treatment however the widespread use of carrageenan and cmc in foods consumed by the pediatric population in a western diet is on the rise alongside a corresponding increase in ibd incidence and questions are being raised about the safety of frequent usage of these food additives therefore further research is warranted to elucidate the role of carrageenan and cmc in intestinal inflammation which may help identify novel nutritional strategies that hinder the development of the disease or prevent disease relapse treatment

(a) Level 1 - correct class : Medical

although the exact pathophysiology remains unknown the development of inflammatory bowel disease ibd is influenced by the interplay between genetics the immune system and environmental factors such as diet the commonly used food additives carrageenan and carboxymethylcellulose cmc are used to develop intestinal inflammation in animal models these food additives are excluded from current dietary approaches to induce disease remission in crohn 's disease such as exclusive enteral nutrition een using a polymeric formula by reviewing the existing scientific literature this review aims to discuss the role that carrageenan and cmc may play in the development of ibd animal studies consistently report that carrageenan and cmc induce histopathological features that are typical of ibd while altering the microbiome disrupting the intestinal epithelial barrier inhibiting proteins that provide protection against microorganisms and stimulating the elaboration of proinflammatory cytokines similar trials directly assessing the influence of carrageenan and cmc in humans are of course unethical to conduct but recent studies of human epithelial cells and the human microbiome support the findings from animal studies carrageenan and cmc may trigger or magnify an inflammatory response in the human intestine but are unlikely to be identified as the sole environmental factor involved in the development of ibd or in disease recurrence after treatment however the widespread use of carrageenan and cmc in foods consumed by the pediatric population in a western diet is on the rise alongside a corresponding increase in ibd incidence and questions are being raised about the safety of frequent usage of these food additives therefore further research is warranted to elucidate the role of carrageenan and cmc in intestinal inflammation which may help identify novel nutritional strategies that hinder the development of the disease or prevent disease relapse treatment

(b) Level 2 - correct class : Crohn's disease

Figure 2: WOS dataset attention rereading per level. Highlighted words indicate the attented words. Stronger color denote higher focus of attention. We note that the attention spread becomes much more focused in Level 2 compared to its parent Level 1.

leaf-node level classification of WOS than that on DBpedia. We observe this behaviour due to the dataset of DBpedia being shorter in average length making it easier to classify for *flat* classifiers, hence hierarchical classifiers overfit on the training data.

In addition to the performance improvement on both datasets over HDLTex, our model takes significantly less time and resources to train, especially when the dataset is large in terms of the intermediate non-leaf nodes in the output taxonomy. As HDLTex needs to build one sub-classifier for each parent nodes, the number of sub-classifiers grows quickly. For example, there are 80 parent nodes in the taxonomy of the DBpedia dataset and HDLTex needs to build 80 RNNs, where each sub-classifier contains around 67 million parameters. As a consequence, we can barely fit the whole model of HDLTex on our CPU [7] because it requires 60 GB RAM to build these 80 deep neural networks.

6 Discussion

Analysis of Attention The intuition behind building dynamic document representations, using multiple attentions across different hierarchical levels, is to have a re-reading effect over the taxonomy. When we first encounter an article as humans, we tend to read it carefully, but on subsequent reads we can easily identify the key aspects of the article. We find in our exploratory experiments the attention vectors behave exactly the same. For the first level, the attention values are more spread out to help our classifier pick various important aspects of the article, but on the subsequent levels, the attention is more focused towards specific keywords for that subclass, as the example shown in Figure 2 [8]. We perform additional qualitative analysis of attention spread which is provided in Appendix.

7 Conclusion

In this work, we propose a light-weight neural-based hierarchical classifier that performs better than or comparable to the state-of-the-art hierarchical model at lower computation cost. Our model employs an adapted version of attention to represent documents dynamically through the hierarchy, which provides additional interpretability of the dynamic document representations. In addition, we demonstrate that the robustness of flat text classification can be improved by using external knowledge such as a hierarchical taxonomy. As a future direction, we will advance our model to automatically construct the hierarchical taxonomy in order to improve text classification with a large number of classes.

Acknowledgements

We thank Compute Canada and Calcul Quebec for providing the GPU compute resources. We would also express our appreciation to our sponsors, Pierre Arbour Foundation and FRQNT for supporting Koustuv Sinha and NSERC for supporting Yue Dong for this research.

---

[7] It is not possible to fit the entire model in one GPU as our best GPU has the RAM capacity of 12GB, one needs to have multiple GPU's and parallel execution for this task.

[8] We use the same visualization script as of Lin et al. (2017).

References

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. A brief survey of text mining: Classification, clustering and extraction techniques. In *Proceedings of KDD Bigdas, Halifax, Canada*.

Chidanand Apté, Fred Damerau, and Sholom M Weiss. 1994. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)*, 12(3):233–251.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Ronan Collobert and Jason Weston. 2008a. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Ronan Collobert and Jason Weston. 2008b. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116.

Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155. ACM.

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

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. 2006. Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering*, 18(11):1457–1466.

Diederik P Kingma and Jimmy Lei Ba. 2014. Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*.

Daphne Koller and Mehran Sahami. 1997. Hierarchically classifying documents using very few words. Technical report, Stanford InfoLab.

Kamran Kowsari, Donald E Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S Gerber, and Laura E Barnes. 2017. HDLTex: Hierarchical deep learning for text classification. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 364–371.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Steve Lawrence, C Lee Giles, and Ah Chung Tsoi. 1997. Lessons in neural network training: Overfitting may be harder than expected. In *AAAI/IAAI*, pages 540–545. Citeseer.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 515–520.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations (ICLR 2017)*.

Shaohui Liu, Mingkai Dong, Haijun Zhang, Rong Li, and Zhongzhi Shi. 2001. An approach of multihierarchy text classification. In *Info-tech and Info-net, 2001. Proceedings. ICII 2001-Beijing. 2001 International Conferences on*, volume 3, pages 95–100. IEEE.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*. Citeseer.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696.

Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Paliouras, Eric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Galinari. 2015. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*.

Michael J Quinn and Mary L Laier. 2006. Method and apparatus for fast lookup of related classification entities in a tree-ordered classification hierarchy. US Patent 7,032,072.

Thomson Reuters. 2012. Web of science.

Gerard Salton and Chris Buckley. 1987. Term weighting approaches in automatic text retrieval. Technical report, Cornell University.

Carlos N Silla and Alex A Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72.

Carlos N Silla Jr and Alex A Freitas. 2009. A global-model naive bayes approach to the hierarchical prediction of protein functions. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 992–997. IEEE.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66.

Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185.

Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. 2008. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 619–626. ACM.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Labeled Anchors and a Scalable, Transparent, and Interactive Classifier

Jeffrey Lund, Stephen Cowley, Wilson Fearn, Emily Hales, Kevin Seppi
Computer Science Department
Brigham Young University
`{jefflund, scowley4, wfearn, emilyhales, kseppi}@byu.edu`

Abstract

We propose Labeled Anchors, an interactive and supervised topic model based on the anchor words algorithm (Arora et al., 2013). Labeled Anchors is similar to Supervised Anchors (Nguyen et al., 2014) in that it extends the vector-space representation of words to include document labels. However, our formulation also admits a classifier which requires no training beyond inferring topics, which means our approach is also fast enough to be interactive. We run a small user study that demonstrates that untrained users can interactively update topics in order to improve classification accuracy.

In this paper, we concern ourselves with the problem of interactive and transparent text classification. The value of such a classifier can be seen in the events shortly before the 2016 US presidential election when FBI Director James Comey notified Congress that the FBI had obtained emails from candidate Hillary Clinton's private email server which potentially contained state secrets. Nearly a week later, just two days before the election, Comey announced that nothing had been found in the emails that warranted prosecution. Many speculate that the timing of these announcements may have influenced the election.

There are times when the ability to quickly analyze large quantities of text is of critical importance. In the case of the Clinton emails, manual inspection appears to have been possible in one week's time, but there could have been less controversy if the emails had been categorized in a shorter period of time. Furthermore, in future cases the data may be too large for manual analysis.

While there are many text classification algorithms, none are both interactive and transparent at scale. We require interactivity because we would like to leverage human intuition to improve classification accuracy for a specific task. Transparency not only enables interactivity, but also allows users to inspect the classifier and gain confidence in the results.

Topic models such as Latent Dirichlet Allocation (or LDA) (Blei et al., 2003) aim to automatically distill large collections of documents into topics. These topics can be used to perform document classification (Rubin et al., 2012). Furthermore, work has been done to increase the human interpretability of topics (Mimno et al., 2011). Traditionally, topic models are graphical models which typically scale poorly to large data. A faster alternative is the Anchor Words algorithm, which relies on non-negative matrix factorization to infer topics (Arora et al., 2013). Ordinarily, this factorization is NP-Hard (Arora et al., 2012), but with certain separability assumptions related to "anchor" words which uniquely identify topics, the factorization is scalable.

The Interactive Topic Model (Hu et al., 2011) allows human knowledge to be injected into the model in order to shape the topics in some meaningful way. While this model does incorporate user feedback, it is not fast enough to be truly interactive. A more scalable alternative is Tandem Anchors, which allows users to specify anchor words in order to influence the resulting topics (Lund et al., 2017).

A separate line of topic modeling research deals with supervised topic modeling, which allows document labels to influence topic inference (Mcauliffe and Blei, 2008). The most recent work on supervised topic modeling is Supervised Anchors (Nguyen et al., 2014). This approach uses document labels to influence the selection of anchor words, which in turn affects the resulting topics. However, Supervised Anchors requires a

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 824–829
Brussels, Belgium, October 31 - November 4, 2018.

downstream classifier to be trained using topics as features.

Our main contribution combines the idea of Tandem Anchors with Supervised Anchors to produce text classification which is both interactive and transparent. Additionally, the mathematical approach we take to build this classification requires no training beyond inferring topics, unlike Supervised Anchors which requires both topic inference and significant additional time for training a downstream classifier. While Supervised Anchors requires the construction of an external classifier, our approach generates the classifier as part of topic inference. Consequently, our model is extremely fast and scalable compared to Supervised Anchors. We demonstrate that users are able to use our model to interactively improve document classification accuracy by manipulating topics.

1 Labeled Anchors

In this section we describe our approach, combining interactive and supervised topic modeling, which we call Labeled Anchors. We extend the Anchor Words algorithm (Arora et al., 2013) which takes as input a $V \times D$ matrix M of document-word counts and recovers a $V \times K$ matrix A of word probabilities conditioned by topic, where there are V word types, D documents, and K topics. Our approach extends this algorithm to incorporate L possible document labels.

1.1 Vanilla Anchor Words

In order to compute the topic-word matrix A, the Anchor Words algorithm uses a $V \times V$ cooccurrence matrix $\bar{Q}$. Each entry $\bar{Q}_{i,j}$ gives the conditional probability of word j occurring after observing word i in a document. Following Appendix D.1 of Arora et al. (2013), $\bar{Q}$ is obtained by row-normalizing Q, which in turn is constructed using

$$Q = \bar{M}\bar{M}^T - \hat{M} \qquad (1)$$

where $\bar{M}$ is a normalized version of the document-word matrix M giving equal weight to each document regardless of document length, and $\hat{M}$ accounts for words not cooccurring with themselves.

$\bar{Q}$ is a V-dimensional vector-space representation of each word and is used to compute a set of anchor words S. Each anchor word uniquely identifies a topic by having non-zero probability in one topic only. These anchors are computed using an adaptation of the Gram-Schmidt process from Arora et al. (2013). Once the set of anchor words S has been computed, we reconstruct the non-anchor words as a convex combination of the anchor word vectors. The coefficients of these combinations C are computed using exponentiated gradient descent to optimize

$$C_i = \underset{C_i}{argmin}\, D_{KL}(\bar{Q}_i || \sum_{k \in S} C_{i,k}\bar{Q}_k) \qquad (2)$$

where i is the ith word of the vocabulary, $\bar{Q}_i$ is the vector-space representation of word i, and $D_{KL}(\cdot||\cdot)$ is Kullback-Leibler divergence.[1]

Because the occurrence pattern of each anchor word throughout the documents must mirror that of the topic it anchors, each coefficient $C_{i,j}$ gives the conditional probability of topic j occurring given word i. This is the inverse conditioning we desire in the topic-word matrix A. We can therefore compute A using Bayes' Rule by multiplying the coefficient matrix C with the empirical probability of each word to get the probability of a word given a particular topic.

1.2 Vector-Space Representations

Supervised Anchors (Nguyen et al., 2014) augments $\bar{Q}$ by appending L additional columns to $\bar{Q}$ corresponding to the probability of words cooccurring with the L possible document labels. Because this augmented vector-space representation includes dimensions corresponding to document labels, both the anchor words and the resulting topics will reflect the document labels.

Our algorithm, called Labeled Anchors, also augments the vector-space representation to include the L document labels. However, we do not directly modify $\bar{Q}$. Instead, we treat the L possible document labels as words and pretend that we observe these label pseudowords directly in each labeled document; a graphical representation of this is shown below in Figure 1.[2]

Consequently, our document-word matrix M is a $(V + L) \times D$ matrix. The first V entries of each column of M give the word counts for a particular document. The last L entries are zero, except for the entry corresponding to the label of that document.

We then construct $\bar{Q}$ using Equation 1, obtaining an order $V + L$ square matrix. As with Su-

---

[1] Alternatively, we can use l^2-norm in place of KL-divergence.

[2] We could add multiple such words per label, but our preliminary experiments indicate that one per label is sufficient.

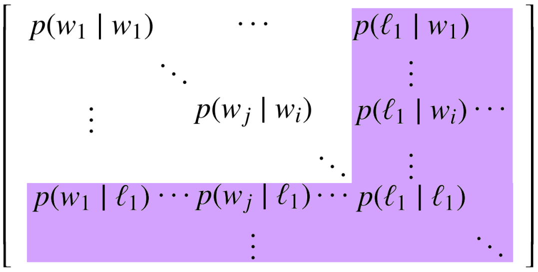

Figure 1: Labeled Anchors treats labels as observed words in each labeled documents and updates $\bar{Q}$ under this assumption, creating the additional rows and columns highlighted here.

pervised Anchors, these additional L dimensions guide anchor selection to include anchors which reflect the underlying document labels. When we use Equation 2 to compute C, we also obtain an additional L rows of coefficients which each correspond to the conditional probability of a topic given a label. Finally, the first V rows of A are computed using Bayes' Rule to give us the probability of words given topics.

Labeled Anchors inherits the run time characteristics of the original Anchor Words algorithm. As shown in Arora et al. (2013), topic recovery requires $O(KV^2 + K^2VT)$, where V is the size of the vocabulary, K is the number of anchors/topics, and T is the average number of iterations (typically around 100 in our experiments). Since $V \gg K$, adding any modest number of topics (less than 200) does not noticeably increase the runtime. Furthermore, since vocabulary size tends to grow logarithmically with respect to the size of the data (Heaps, 1978), this approach is scalable even for very large datasets.

1.3 Free Classifier

Note that once the cooccurrence matrix $\bar{Q}$ has been computed, the recovery of the topic-word matrix A scales with the size of the vocabulary, not the size of the data. However, Supervised Anchors requires topic assignments for each training document[3] for use as features for some downstream classifier. Therefore, the process of building a classifier scales linearly with the number of documents and can be time consuming compared to topic recovery.

In contrast, the formulation of Labeled Anchors allows us to construct a classifier with no additional training. To do so, rather than using LDA with fixed topics, we employ a simple model similar to Labeled LDA (Ramage et al., 2009) with the following generative story for an individual document containing N words:

1. Draw label $\ell \sim Cat(\lambda)$
2. For each $i \in [1, ..., N]$:
   (a) Draw topic assignment $z_i|\ell \sim Cat(\psi_l)$
   (b) Draw word $w_i|z_i \sim Cat(\phi_{z_i})$

The prior over document labels λ is simply the proportion of each label in the training data. We can estimate topic-label probabilities ψ using the last L rows of the coefficient matrix C, while the word-topic probabilities ϕ are the first V rows of A. Using these hyperparameters, we make predictions using the following:

$$\ell^* = \underset{\ell}{argmax}\ p(\ell|\mathbf{w}) = \underset{\ell}{argmax}\ p(\ell, \mathbf{w}) \quad (3)$$

$$= \underset{\ell}{argmax} \sum_{z_1=1}^{K} ... \sum_{z_N=1}^{K} p(\ell, \mathbf{z}, \mathbf{w}) \quad (4)$$

$$= \underset{\ell}{argmax}\ p(\ell) \prod_{i=1}^{N} \sum_{z_i=1}^{K} p(z_i|\ell) p(w_i|z_i) \quad (5)$$

$$= \underset{\ell}{argmax}\ \lambda_\ell \prod_{i=1}^{N} \sum_{z_i=1}^{K} \psi_{\ell,z_i} \phi_{z_i,w_i} \quad (6)$$

$$= \underset{\ell}{argmax}\ log\lambda_\ell + \sum_{i=1}^{N} log \left(\sum_{z_i=1}^{K} C_{\ell,z_i} A_{z_i,w_i} \right) \quad (7)$$

where Equation 4 unmarginalizes the probabilities across the word-topic assignments, Equation 5 uses the model's conditional independencies to expand and simplify the probabilities, Equation 6 explicitly uses the parameters from the generative model, and Equation 7 transitions to the matrix representations for these probabilities as found in Section 1.2. In Equation 7 we also switch to log space to mitigate numeric precision issues.

1.4 User Interaction

Assuming that $\bar{Q}$ is precomputed and fixed, Labeled Anchors is fast enough to allow interactive modification of the topics as well as interactive display of classification accuracy, even on large datasets. The final step to solving the problem of creating an interactive and transparent classifier is to allow users to inject domain specific knowledge

---

[3] This is typically done using LDA with fixed topics.

#Docs	#Vocab	Labeled	Supervised
39388	3406	.532s	17.1s
99955	4829	.886s	28.6s
990820	6648	1.10s	282s

Table 1: Runtime for Labeled Anchors and Supervised Anchors on various subsets of Amazon product reviews. Labeled Anchors is dramatically faster than Supervised Anchors and scales to much larger datasets.

into the topic model. To do so, we use the idea of Tandem Anchors (Lund et al., 2017), which allows users to manually select sets of words to form anchors.

Ordinarily, anchor words can be somewhat inscrutable to human users. Because anchor words must uniquely identify topics, good anchors are typically esoteric low-to-mid frequency words. Intuitive, high frequency words usually appear in multiple topics. However, if we examine Equation 2, we can see that the anchor words are just points in V-dimension space; they do not actually have to correspond to any particular word so long as that point in space uniquely identifies a topic.

Tandem Anchors allows multiple words to form a single anchor pseudoword by computing the element-wise harmonic mean of a set of words. Since the harmonic mean tends towards the lowest values, the resulting pseudoword anchor largely ignores superfluous cooccurrence patterns in the constituent words. Consequently, while individual words forming the anchor may be ambiguous, users can combine multiple ambiguous words to intuitively express a single coherent idea.

2 Experimental Results

Before running a user study to validate that Labeled Anchors works as an interactive and transparent classifier, we first run a synthetic experiment to determine the runtime characteristics of our algorithm. We take subsets of a large collection of Amazon reviews[4] to produce datasets of various sizes. Using this data, we compare the runtime of Labeled Anchors with that of Supervised Anchors. All results are obtained using a single core of an Intel Core i7-4770K.[5]

As shown in Table 1, Labeled Anchors is orders of magnitude faster than Supervised Anchors, even for moderately sized datasets. Both Labeled Anchors and Supervised Anchors require us to recover topic-word distributions, an operation which scales with the size of the vocabulary. However, Supervised Anchors also requires us to infer document-topic distributions in order to train an external classifier, an operation which scales linearly with the number of documents. Since vocabulary size typically grows logarithmically with respect to the number of documents (Heaps, 1978), Labeled Anchors scales much better than Supervised Anchors.

When a user updates the anchors, the system must reinfer the topics, create the classifier, and evaluate the development dataset, all within a few seconds. If the update is too slow, the interaction will suffer due to increased cognitive load on users (Cook and Thomas, 2005). Results from an exploratory user study confirm this: when participants are faced with update times around 10 seconds, they are not successful in their topic-based tasks.[6]

Having established that Labeled Anchors is fast enough to be interactive, we now demonstrate that participants can use our system to improve topics for classification. In order demonstrate the role of human knowledge in interactive topic modeling, we ask the users to identify sentiment (i.e. product rating) rather than product category, since the natural topics which arise from the Anchor Words algorithm tend to reflect product category instead of rating. We preprocess a set of Amazon product reviews with standard tokenization, stopword removal, and by removing words which appear in fewer than 100 documents. After preprocessing, empty documents are discarded, resulting in 39,388 documents. We use an 80/20 train/test split, with 1,500 training documents reserved as development data. We recruit five participants drawn from a university student body. The median age is 21. Three are male and two are female. None of the students have any prior familiarity with topic modeling.

We present the participants with a user interface similar to that of Lund et al. (2017). Users can view and edit a list of anchor words (or rather, sets of words which form each anchor), and they can view the top ten most probable words for each topic. We display the classification accuracy on the development data to give users an indication of

---

[4] http://jmcauley.ucsd.edu/data/amazon

[5] Our Python implementation is available at `https://github.com/byu-aml-lab/ankura`.

[6] For this reason, we do not report interactive results with Supervised Anchors.

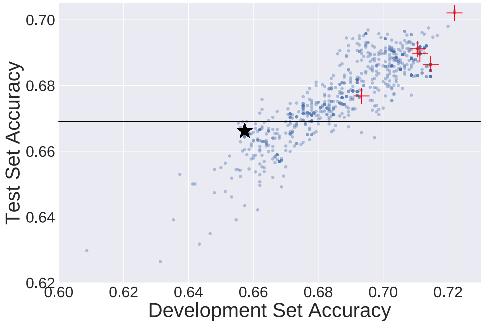

Figure 2: User study accuracy results comparing accuracy on the development set to the accuracy on the test set. The black horizontal line indicates the baseline accuracy from Supervised Anchors. The black star indicates the initial accuracy using Gram-Schmidt anchors with Labeled Anchors. The blue dots indicate various intermediate steps while editing the anchors. The red pluses are the final states after each user completes the task.

how they are doing. After a brief training on the interface, users are asked to modify the anchors to produce topics which reflect the underlying product ratings and improve the classification accuracy on the development dataset as much as possible. Participants are given forty minutes to perform this task.

Figure 2 summarizes the results of our user study. With just baseline anchors from Gram-Schmidt, the classification accuracy of Labeled Anchors is on par with that of Supervised Anchors using logistic regression as the downstream classifier. However, because Labeled Anchors is fast enough to allow interaction, participants are able to improve classification accuracy on the development set by an average of 5.31%. This corresponds to a 2.31% increase in accuracy on the test set.

We record each step of the user interactions and find a Pearson correlation coefficient of .88 between development accuracy and test accuracy. Thus, Labeled Anchors allows participants to interactively see updated classification accuracy and have confidence that held-out test accuracy will also improve.

With regard to the interaction that users had with the dataset, we observe several common strategies. Firstly, we notice that users who made more edits tend to have more success in terms of accuracy; this validates our assertion that slower update times hurt performance. Secondly, users end with a median of 21 topics, which is close to the 20 topics they start with, suggesting that either the users felt like this was an appropriate number of topics, or that they felt uneasy drastically changing the total number of topics from what they started with. Lastly, we find that users chose more single word anchors than we expected, with about 88% of anchors being single word anchors. Most of the multiword anchors users used were short 2-3 word phrases which did not have an obvious single word counterpart.

3 Conclusion

Our results demonstrate that Labeled Anchors yields a classifier that is both human-interpretable and fast. Our approach not only combines the strengths of Supervised Anchors and Tandem Anchors, but introduces a mathematical construct for producing a classifier as a by-product of topic inference. Compared to Supervised Anchors, which requires costly training of a downstream classifier in addition to topic inference, our approach is much more scalable. Using Labeled Anchors, our participants are able to adjust the classifier so as to obtain superior classification results than those produced by Supervised Anchors alone.

Returning to our original motivating problem of quickly annotating a large collection of unlabeled emails, we assert that our approach could aid in quickly labeling the entire collection. With a modest investment of manual annotation, the initial training set could be labeled, and then with the help of our system the remaining documents could be automatically labeled in a transparent and explainable fashion.

Acknowledgements

This work was supported by the NSF Grant IIS-1409739.

Thanks to Piper Armstrong, Naomi Johnson, Connor Cook and Nozomu Okuda for their invaluable help with this work.

References

Sanjeev Arora, Rong Ge, Yoni Halpern, David M. Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Zhu Michael. 2013. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of the International Conference of Machine Learning*.

Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. 2012. Computing a nonnegative matrix factorization–provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Kristin A. Cook and James J. Thomas. 2005. Illuminating the path: The research and development agend for visual analytics.

Harold Stanley Heaps. 1978. *Information retrieval: Computational and theoretical aspects*. Academic Press, Inc.

Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2011. Interactive topic modeling. In *Proceedings of the Association for Computational Linguistics*.

Jeffrey Lund, Connor Cook, Kevin Seppi, and Jordan Boyd-Graber. 2017. Tandem anchoring: A multiword anchor approach for interactive topic modeling. In *Proceedings of the Association for Computational Linguistics*.

Jon D Mcauliffe and David M Blei. 2008. Supervised topic models. *Proceedings of Advances in Neural Information Processing Systems*.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of Empirical Methods in Natural Language Processing*.

Thang Nguyen, Jordan Boyd-Graber, Jeff Lund, Kevin Seppi, and Eric Ringger. 2014. Is your anchor going up or down? Fast and accurate supervised topic models. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. 2009. Labeled lda: A supervised topic model for credit attribution in multilabeled corpora. In *Proceedings of Empirical Methods in Natural Language Processing*.

Timothy Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. 2012. Statistical topic models for multi-label document classification. *Machine Learning*, 1(88):157–208.

Coherence-Aware Neural Topic Modeling

Ran Ding, Ramesh Nallapati, Bing Xiang
Amazon Web Services
`{rding, rnallapa, bxiang}@amazon.com`

Abstract

Topic models are evaluated based on their ability to describe documents well (i.e. low perplexity) and to produce topics that carry coherent semantic meaning. In topic modeling so far, perplexity is a direct optimization target. However, topic coherence, owing to its challenging computation, is not optimized for and is only evaluated after training. In this work, under a neural variational inference framework, we propose methods to incorporate a topic coherence objective into the training process. We demonstrate that such a coherence-aware topic model exhibits a similar level of perplexity as baseline models but achieves substantially higher topic coherence.

1 Introduction

In the setting of a topic model (Blei, 2012), perplexity measures the model's capability to describe documents according to a generative process based on the learned set of topics. In addition to describing documents well (i.e. achieving low perplexity), it is desirable to have topics (represented by top-N most probable words) that are human-interpretable. Topic interpretability or coherence can be measured by *normalized point-wise mutual information* (NPMI) (Aletras and Stevenson, 2013; Lau et al., 2014). The calculation of NPMI however is based on look-up operations in a large reference corpus and therefore is non-differentiable and computationally intensive. Likely due to these reasons, topic models so far have been solely optimizing for perplexity, and topic coherence is only evaluated after training. As has been noted in several publications (Chang et al., 2009), optimization for perplexity alone tends to negatively impact topic coherence. Thus, without introducing topic coherence as a training objective, topic modeling likely produces sub-optimal results.

Compared to classical methods, such as mean-field approximation (Hoffman et al., 2010) and collapsed Gibbs sampling (Griffiths and Steyvers, 2004) for the latent Dirichlet allocation (LDA) (Blei et al., 2003) model, neural variational inference (Kingma and Welling, 2013; Rezende et al., 2014) offers a flexible framework to accommodate more expressive topic models. We build upon the line of work on topic modeling using neural variational inference (Miao et al., 2016, 2017; Srivastava and Sutton, 2017) and incorporate topic coherence awareness into topic modeling.

Our approaches of constructing topic coherence training objective leverage pre-trained word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Salle et al., 2016; Joulin et al., 2016). The main motivation is that word embeddings carry contextual similarity information that is highly related to the mutual information terms involved in the calculation of NPMI. In this paper, we explore two methods: (1) we explicitly construct a differentiable surrogate topic coherence regularization term; (2) we use word embedding matrix as a factorization constraint on the topical word distribution matrix that implicitly encourages topic coherence.

2 Models

2.1 Baseline: Neural Topic Model (NTM)

The model architecture shown in Figure 1 is a variant of the Neural Variational Document Model (NVDM) (Miao et al., 2016). Let $x \in \mathbb{R}^{|V|\times 1}$ be the bag-of-words (BOW) representation of a document, where $|V|$ is the size of the vocabulary and let $z \in \mathbb{R}^{K\times 1}$ be the latent topic variable, where K is the number of topics. In the encoder $q_\phi(z|x)$, we have $\pi = f_{MLP}(x)$, $\mu(x) = l_1(\pi)$, $\log \sigma(x) = l_2(\pi)$, $h(x,\epsilon) = \mu + \sigma \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$, and finally $z = f(h) = \mathrm{ReLU}(h)$. The functions l_1 and l_2 are linear transformations

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 830–836
Brussels, Belgium, October 31 - November 4, 2018.

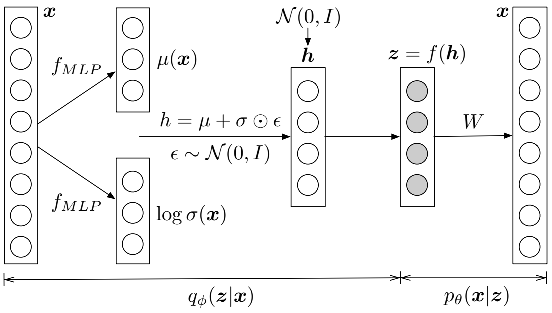

Figure 1: Model architecture

with bias. We choose the multi-layer perceptron (MLP) in the encoder to have two hidden layers with $3 \times K$ and $2 \times K$ hidden units respectively, and we use the sigmoid activation function. The decoder network $p_\theta(x|z)$ first maps z to the predicted probability of each of the word in the vocabulary $y \in \mathbb{R}^{|V|\times 1}$ through $y = \text{softmax}(Wz + b)$, where $W \in \mathbb{R}^{|V|\times K}$. The log-likelihood of the document can be written as $\log p_\theta(x|z) = \sum_{i=1}^{|V|}\{x \odot \log y\}$. We name this model Neural Topic Model (NTM) and use it as our baseline. We use the same encoder MLP configuration for our NVDM implementation and all variants of NTM models used in Section 3. In NTM, the objective function to maximize is the usual *evidence lower bound* (ELBO) which can be expressed as

$$\mathcal{L}_{ELBO}(x^i)$$
$$\approx \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^i|z^{i,l}) - D_{KL}(q_\phi(h|x)||p_\theta(h))$$

where $z^{i,l} = \text{ReLU}(h(x^i, \epsilon^l))$, $\epsilon^l \sim \mathcal{N}(0, I)$. We approximate $\mathbb{E}_{z\sim q(z|x)}[\log p_\theta(x|z)]$ with Monte Carlo integration and calculate the Kullback-Liebler (KL) divergence analytically using the fact $D_{KL}(q_\phi(z|x)||p_\theta(z)) = D_{KL}(q_\phi(h|x)||p_\theta(h))$ due to the invariance of KL divergence under deterministic mapping between h and z.

Compared to NTM, NVDM uses different activation functions and has $z = h$. Miao (2017) proposed a modification to NVDM called Gaussian Softmax Model (GSM) corresponding to having $z = \text{softmax}(h)$. Srivastava (2017) proposed a model called ProdLDA, which uses a Dirichlet prior instead of Gaussian prior for the latent variable h. Given a learned W, the practice to extract top-N most probable words for each topic is to take the most positive entries in each column of W (Miao et al., 2016, 2017; Srivastava and Sutton, 2017). This is an intuitive choice, provided that z is non-negative, which is indeed the case for NTM, GSM and ProdLDA. NVDM, GSM, and ProdLDA are state-of-the-art neural topic models which we will use for comparison in Section 3.

2.2 Topic Coherence Regularization: NTM-R

The topic coherence metric NPMI (Aletras and Stevenson, 2013; Lau et al., 2014) is defined as

$$\text{NPMI}(\boldsymbol{w})$$
$$= \frac{1}{N(N-1)} \sum_{j=2}^{N}\sum_{i=1}^{j-1} \frac{\log \frac{P(w_i,w_j)}{P(w_i)P(w_j)}}{-\log P(w_i, w_j)}$$

where $\boldsymbol{w}$ is the list of top-N words for a topic. N is usually set to 10. For a model generating K topics, the overall NPMI score is an average over all topics. The computational overhead and non-differentiability originate from extracting the co-occurrence frequency from a large corpus[1].

From the NPMI formula, it is clear that word-pairs that co-occur often would score high, unless they are rare word-pairs – which would be normalized out by the denominator. The NPMI scoring bears remarkable resemblance to the contextual similarity produced by the inner product of word embedding vectors. Along this line of reasoning, we construct a differentiable, computation-efficient word embedding based topic coherence (WETC).

Let E be the row-normalized word embedding matrix for a list of N words, such that $E \in \mathbb{R}^{N\times D}$ and $\|E_{i,:}\| = 1$, where D is the dimension of the embedding space. We can define *pair-wise* word embedding topic coherence in a similar spirit as NPMI:

$$\text{WETC}_{PW}(E) = \frac{1}{N(N-1)} \sum_{j=2}^{N}\sum_{i=1}^{j-1} \langle E_{i,:}, E_{j,:}\rangle$$
$$= \frac{\sum\{E^T E\} - N}{2N(N-1)}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. Alternatively, we can define *centroid* word embedding topic coherence

$$\text{WETC}_C(E) = \frac{1}{N}\sum\{Et^T\}$$

where vector $t \in \mathbb{R}^{1\times D}$ is the centroid of E, normalized to have $\|t\| = 1$. Empirically, we found

---

[1] A typical calculation of NPMI over 50 topics based on the Wikipedia corpus takes ∼20 minutes, using code provided by (Lau et al., 2014) at `https://github.com/jhlau/topic_interpretability`.

that the two WETC formulations behave very similarly. In addition, both WETC$_{PW}$ and WETC$_{C}$ correlate to human judgement almost equally well as NPMI when using `GloVe` (Pennington et al., 2014) vectors[2].

With the above observations, we propose the following procedure to construct a WETC-based surrogate topic coherence regularization term: (1) let $E \in \mathbb{R}^{|V| \times D}$ be the pre-trained word embedding matrix for the vocabulary, rows aligned with W; (2) form the W-weighted centroid (topic) vectors $T \in \mathbb{R}^{D \times K}$ by $T = E^T W$; (3) calculate the cosine similarity matrix $S \in \mathbb{R}^{|V| \times K}$ between word vectors and topic vectors by $S = ET$; (4) calculate the W-weighted sum of word-to-topic cosine similarities for each topic $C \in \mathbb{R}^{1 \times K}$ as $C = \sum_i (S \odot W)_{i,:}$. Compared to WETC$_C$, in calculating C we do not perform top-N operation in W, but directly use W for weighted sum. Specifically, we use W-weighted topic vector construction in Step-2 and W-weighted sum of the cosine similarities between word vectors and topic vectors in Step-4. To avoid unbounded optimization, we normalize the rows of E and the columns of W before Step-2, and normalize the columns of T after Step-2. The overall maximization objective function becomes $\mathcal{L}_R(x; \theta, \phi) = \mathcal{L}_{ELBO} + \lambda \sum_i C_i$, where λ is a hyper-parameter with positive values controlling the strength of topic coherence regularization. We name this model NTM-R.

2.3 Word Embedding as a Factorization Constraint: NTM-F and NTM-FR

Instead of allowing all the elements in W to be freely optimized, we can impose a factorization constraint of $W = E\hat{T}$, where E is the pre-trained word embedding matrix that is *fixed*, and only $\hat{T}$ is allowed to be learned through training. Under this configuration, $\hat{T}$ lives in the embedding space, and each entry in W is the dot product similarity between a topic vector $\hat{T}_i$ and a word vector E_j. As one can imagine, similar words would have similar vector representations in E and would have similar weights in each column of W. Therefore the factorization constraint encourages words with similar meaning to be selected or de-selected together thus potentially improving topic coherence.

We name the NTM model with factorization constraint enabled as NTM-F. In addition, we can apply the regularization discussed in the previous section on the resulting matrix W and we name the resulting model NTM-FR.

Metric	Perplexity		NPMI	
Number of topics	50	200	50	200
LDA				
LDA, mean-field	1046	1195	0.11	0.06
LDA, collapsed Gibbs	**728**	**688**	0.17	0.14
Neural Models				
NVDM	750	743	0.14	0.13
GSM	787	829	0.22	0.19
ProdLDA	1172	1168	0.28	0.24
NTM	780	768	0.18	0.18
NTM-R	**775**	**763**	**0.28**	**0.23**
NTM-F	898	1086	**0.29**	0.24
NTM-FR	924	1225	0.27	**0.26**

Table 1: Comparison to LDA and neural variational models on the *20NewsGroup* dataset. Best numbers are bolded. The blue underlined row highlights the best NPMI and perplexity tradeoff as discussed in text.

3 Experiments and Discussions

3.1 Results on *20NewsGroup*

First, we compare the proposed models to state-of-the-art neural variational inference based topic models in the literature (NVDM, GSM, and ProdLDA) as well as LDA benchmarks, on the *20NewsGroup* dataset[3]. In training NVDM and all NTM models, we used `Adadelta` optimizer (Zeiler, 2012). We set the learning rate to 0.01 and train with a batch size of 256. For NTM-R, NTM-F and NTM-FR, the word embedding we used is `GloVe` (Pennington et al., 2014) vectors pre-trained on Wikipedia and Gigaword with 400,000 vocabulary size and 50 embedding dimensions[4]. The topic coherence regularization coefficient λ is set to 50. The results are presented in Table 1.

Overall we can see that LDA trained with collapsed Gibbs sampling achieves the best perplexity, while NTM-F and NTM-FR models achieve the best topic coherence (in NPMI). Clearly, there is a trade-off between perplexity and NPMI as identified by other papers. So we constructed Figure 2, which shows the two metrics from various models. For the models we implemented, we additionally

---

[2] See Appendix A for details on an empirical study of human judgement of topic coherence, NPMI and WETC with various types of word embeddings.

[3] We use the exact dataset from (Srivastava and Sutton, 2017) to avoid subtle differences in pre-processing

[4] Obtained from `https://nlp.stanford.edu/projects/glove/`

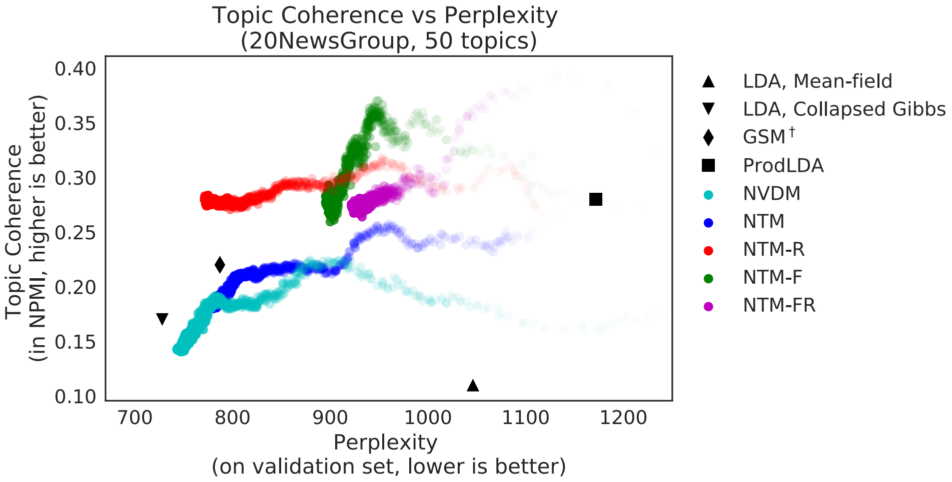

Figure 2: NPMI vs. perplexity for various models at 50 topics. For NVDM and NTM models the traces correspond to the evolution over training epochs. High transparency is the beginning of the training.

show the full evolution of these two metrics over training epochs.

From Figure 2, it becomes clear that although ProdLDA exhibits good performance on NPMI, it is achieved at a steep cost of perplexity, while NTM-R achieves similar or better NPMI at much lower perplexity levels. At the other end of the spectrum, if we look for low perplexity, the best numbers among neural variational models are between 750 and 800. In this neighborhood, NTM-R substantially outperforms the GSM, NVDM and NTM baseline models. Therefore, we consider NTM-R the best model overall. Different downstream applications may require different tradeoff points between NPMI and perplexity. However, the proposed NTM-R model does appear to provide tradeoff points on a Pareto front compared to other models across most of the range of perplexity.

3.2 Comments on NTM-F and NTM-FR

It is worth noting that although NTM-F and NTM-FR exhibit high NPMI early on, they fail to maintain it during the training process. In addition, both models converged to fairly high perplexity levels. Our hypothesis is that this is caused by NTM-F and NTM-FR's substantially reduced parameter space - from $|V| \times K$ to $D \times K$, where $|V|$ ranges from 1,000 to 150,000 in a typical dataset, while D is on the order of 100.

Some form of relaxation could alleviate this problem. For example, we can let $W = E\hat{T} + A$, where A is of size $|V| \times K$ but is heavily regularized, or let $W = EQ\hat{T}$ where Q is allowed as additional free parameters. We leave fully addressing this to future work.

3.3 Validation on other Datasets

To further validate the performance improvement from using WETC-based regularization in NTM-R, we compare NTM-R with the NTM baseline model on a few more datasets: DailyKOS, NIPS, and NYTimes[5] (Asuncion and Newman, 2007). These datasets offer a wide range of document length (ranging from ∼100 to ∼1000 words), vocabulary size (ranging from ∼7,000 to ∼140,000), and type of documents (from news articles to long-form scientific writing). In this set of experiments, we used the same settings and hyperparameter λ as before and did not fine-tune for each dataset. The results are presented in Figure 3. In a similar style as Figure 2, we show the evolution of NPMI and WETC versus perplexity over epochs until convergence.

Among all datasets, we observed improved NPMI at the same perplexity level, validating the effectiveness of the topic coherence regularization. However, on the NYTimes dataset, the improvement is quite marginal even though WETC improvements are very noticeable. One particularity about the NYTimes dataset is that approximately 58,000 words in the 140,000-word vocabulary are named entities. It appears that the large number of named entities resulted in a divergence between NPMI and WETC scoring, which is an issue to address in the future.

---

[5] `https://archive.ics.uci.edu/ml/datasets/Bag+of+Words`

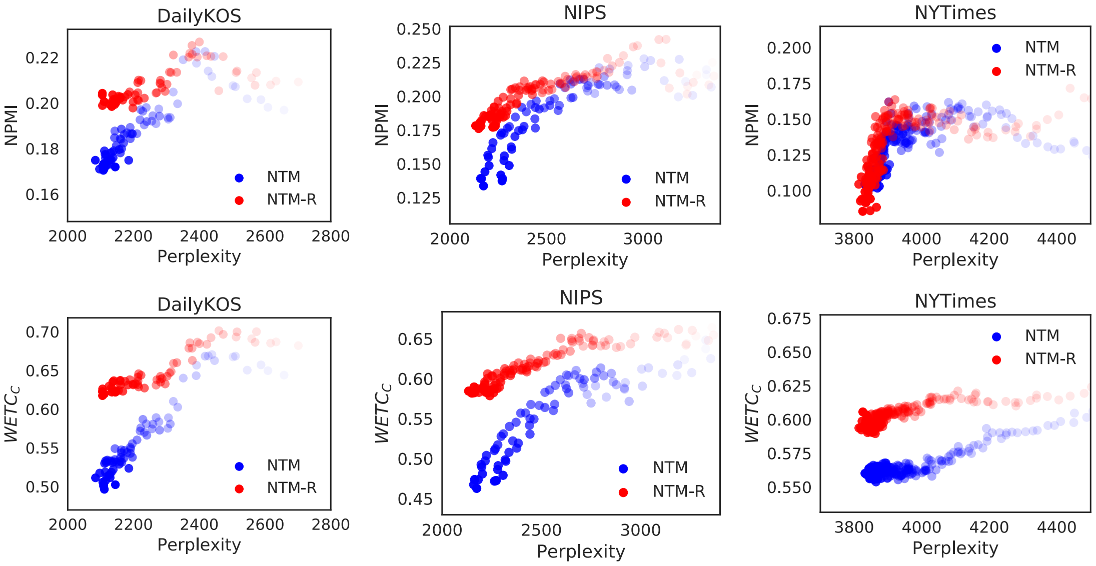

Figure 3: Performance comparison between NTM-R and NTM on multiple datasets, with 50 topics. Top row is NPMI versus perplexity, bottom row is WETC_C versus perplexity. From left to right: DailyKOS, NIPS, and NYTimes. See text for details about the datasets.

4 Conclusions

In this work, we proposed regularization and factorization constraints based approaches to incorporate awareness of topic coherence into the formulation of topic models: NTM-R and NTM-F respectively. We observed that NTM-R substantially improves topic coherence with minimal sacrifice in perplexity. To our best knowledge, NTM-R is the first topic model that is trained with an objective towards topic coherence – a feature directly contributing to its superior performance. We further showed that the proposed WETC-based regularization method is applicable to a wide range of text datasets.

References

Nikolaos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*, pages 13–22.

Arthur Asuncion and David Newman. 2007. UCI Machine Learning Repository.

David M Blei. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, pages 288–296.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 856–864.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. In *International Conference on Machine Learning*, pages 2410–2419.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286.

Alexandre Salle, Marco Idiart, and Aline Villavicencio. 2016. Matrix factorization using window sampling and negative sampling for improved word representations. *arXiv preprint arXiv:1606.00819*.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

A Word Embedding Topic Coherence

As studied in (Aletras and Stevenson, 2013) and (Lau et al., 2014), the NPMI metric for assessing topic coherence over a list of words $\boldsymbol{w}$ is defined in Eq. 1.

$$\text{NPMI}(\boldsymbol{w}) = \frac{1}{N(N-1)} \sum_{j=2}^{N} \sum_{i=1}^{j-1} \frac{\log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}}{-\log P(w_i, w_j)} \quad (1)$$

where $P(w_i)$ and $P(w_i, w_j)$ are the probability of words and word pairs, calculated based on a reference corpus. N is usually set to 10, by convention, so that NPMI is evaluated over the topic-10 words for each topic. For a model generating K topics, the overall NPMI score is an average over all the topics. The computational overhead comes from extracting the relevant co-occurrence frequency from a large corpus. This problem is exacerbated when the look-up also requires a small sliding window as the authors of (Lau et al., 2014) suggested. A typical calculation of 50 topics based on a few million documents from the Wikipedia corpus takes ∼20 minutes[6].

For a list of words $\boldsymbol{w}$ of length N, we can assemble a corresponding word embedding matrix $E \in \mathbb{R}^{N \times D}$ with each row corresponding to a word in the list. D is the dimension of the embedding space. Averaging across the rows, we can obtain vector $t \in \mathbb{R}^{1 \times D}$ as the centroid of all the word vectors. It may be regarded as a "topic" vector. In addition, we assume that each row of E and t is normalized, i.e. $\|t\| = 1$ and $\|E_{i,:}\| = 1$. With these, we define *pair-wise* and *centroid* word embedding topic coherence WETC_{PW} and WETC_{C} as follows:

$$\begin{aligned}\text{WETC}_{PW}(E) &= \frac{1}{N(N-1)} \sum_{j=2}^{N} \sum_{i=1}^{j-1} \langle E_{i,:}, E_{j,:} \rangle \\ &= \frac{\sum\{E^T E\} - N}{2N(N-1)}\end{aligned} \quad (2)$$

$$\text{WETC}_C(E) = \frac{1}{N} \sum \{E t^T\} \quad (3)$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. The simplification in Eq. 2 is due to the row normalization of E.

In this setting, we have the flexibility to use any pre-trained word embeddings to construct E. To experiment, we compared several recently developed variants [7]. The dataset from (Aletras and Stevenson, 2013) provides human ratings for 300 topics coming from 3 corpora: 20NewsGroup (20NG), New York Times (NYT) and genomics scientific articles (Genomics), which we use as the human gold standard. We use Pearson and Spearman correlations to compare NPMI and WETC scores against human ratings. The results are shown in Table 2.

---

[6] Using code provided by (Lau et al., 2014) at `https://github.com/jhlau/topic_interpretability`. Running parallel processes on 8 Intel Xeon E5-2686 CPUs.

[7] Details of pre-trained word embeddings used in Table 2

- `Word2Vec` (Mikolov et al., 2013): pre-trained on GoogleNews, with 3 million vocabulary size and 300 embedding dimension. Obtained from `https://code.google.com/archive/p/word2vec/`.
- `GloVe` (Pennington et al., 2014): pre-trained on Wikipedia and Gigaword, with 400,000 vocabulary size and 50 and 300 embedding dimension. Obtained from `https://nlp.stanford.edu/projects/glove/`.
- `FastText` (Joulin et al., 2016): pre-trained on Wikipedia with 2.5 million vocabulary size and 300 embedding dimension. Obtained from `https://github.com/facebookresearch/fastText`.
- `LexVec` (Salle et al., 2016): pre-trained on Wikipedia with 370,000 vocabulary size and 300 embedding dimension. Obtained from `https://github.com/alexandres/lexvec`.

Dataset	20NG		NYT		Genomics	
Correlation	P	S	P	S	P	S
NPMI	0.74	0.74	0.72	0.71	0.62	0.65
GloVe-50d						
WETC$_{PW}$	0.82	0.77	0.73	0.71	0.65	0.65
WETC$_{C}$	0.81	0.77	0.73	0.71	0.65	0.65
GloVe-300d						
WETC$_{PW}$	0.77	0.75	0.78	0.76	0.68	0.70
WETC$_{C}$	0.80	0.75	0.78	0.76	0.68	0.70
Word2Vec						
WETC$_{PW}$	0.29	0.23	0.53	0.59	0.56	0.55
WETC$_{C}$	0.31	0.23	0.55	0.59	0.56	0.55
FastText						
WETC$_{PW}$	0.40	0.61	0.63	0.67	0.62	0.62
WETC$_{C}$	0.48	0.61	0.64	0.67	0.63	0.62
LexVec						
WETC$_{PW}$	0.37	0.57	0.79	0.80	0.65	0.64
WETC$_{C}$	0.47	0.57	0.81	0.80	0.65	0.64

Table 2: NPMI and WETC correlation with human gold standard (P: Pearson, S: Spearman)

From Table 2 we observed a minimal difference between pair-wise and centroid based WETC in general. Overall, `GloVe` appears to perform the best across different types of corpora and its correlation with human ratings is very comparable to NPMI-based scores. Our NPMI calculation is based on the Wikipedia corpus and should serve as a fair comparison. In addition to the good correlation exhibited by WETC, the evaluation of WETC only involves matrix multiplications and summations and thus is fully differentiable and several orders of magnitude faster than NPMI calculations. WETC opens the door of incorporating topic coherence as a training objective, which is the key idea we will investigate in the subsequent sections. It is worth mentioning that, for `GloVe`, the low dimensional embedding (50d) appears to perform almost equally well as high dimensional embedding (300d). Therefore, we will use Glove-400k-50d in all subsequent experiments.

While the WETC metric on its own might be of interest to the topic modeling research community, we leave the task of formally establishing it as a standard metric in place of NPMI to future work. In this work, we still use the widely accepted NPMI as the objective topic coherence metric for model comparisons.

Utilizing Character and Word Embeddings for Text Normalization with Sequence-to-Sequence Models

Daniel Watson, Nasser Zalmout and Nizar Habash
Computational Approaches to Modeling Language Lab
New York University Abu Dhabi
{daniel.watson,nasser.zalmout,nizar.habash}@nyu.edu

Abstract

Text normalization is an important enabling technology for several NLP tasks. Recently, neural-network-based approaches have outperformed well-established models in this task. However, in languages other than English, there has been little exploration in this direction. Both the scarcity of annotated data and the complexity of the language increase the difficulty of the problem. To address these challenges, we use a sequence-to-sequence model with character-based attention, which in addition to its self-learned character embeddings, uses word embeddings pre-trained with an approach that also models subword information. This provides the neural model with access to more linguistic information especially suitable for text normalization, without large parallel corpora. We show that providing the model with word-level features bridges the gap for the neural network approach to achieve a state-of-the-art F_1 score on a standard Arabic language correction shared task dataset.

1 Introduction

Text normalization systems have many potential applications – from assisting native speakers and language learners with their writing, to supporting NLP applications with sparsity reduction by cleaning large textual corpora. This can help improve benchmarks across many NLP tasks.

In recent years, neural encoder-decoder models have shown promising results in language tasks like translation, part-of-speech tagging, and text normalization, especially with the use of an attention mechanism. In text normalization, however, previous state-of-the-art results rely on developing many other pipelines on top of the neural model. Furthermore, such neural approaches have barely been explored for this task in Arabic, where previous state-of-the-art systems rely on combining various statistical and rule-based approaches.

We experiment with both character embeddings and pre-trained word embeddings, using several embedding models, and we achieve a state-of-the-art F_1 score on an Arabic spelling correction task.

2 Related Work

The encoder-decoder neural architecture (Sutskever et al., 2014; Cho et al., 2014) has shown promising results in text normalization tasks, particularly in character-level models (Xie et al., 2016; Ikeda et al., 2016). More recently, augmenting this neural architecture with the attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) has dramatically increased the quality of results across most NLP tasks. However, in text normalization, state-of-the-art results involving attention (e.g., Xie et al. 2016) also rely on several other models during inference, such as language models and classifiers to filter suggested edits. Neural architectures at the word level inherently rely on multiple models to align and separately handle out-of-vocabulary (OOV) words (Yuan and Briscoe, 2016).

In the context of Arabic, we are only aware of one attempt to use a neural model for end-to-end text normalization (Ahmadi, 2017), but it fails to beat all baselines reported later in this paper. Arabic diacritization, which can be considered forms of text normalization, has received a number of neural efforts (Belinkov and Glass, 2015; Abandah et al., 2015). However, state-of-the-art approaches for end-to-end text normalization rely on several additional models and rule-based approaches as hybrid models (Pasha et al., 2014; Rozovskaya et al., 2014; Nawar, 2015; Zalmout and Habash, 2017), which introduce direct human knowledge into the system, but are limited to correcting specific mistakes and rely on expert knowledge to be developed.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 837–843
Brussels, Belgium, October 31 - November 4, 2018.

3 Approach

Many common mistakes addressed by text normalization occur fundamentally at the character level. Moreover, the input data tends to be too noisy for a word-level neural model to be an end-to-end solution due to the high number of OOV words. In Arabic, particularly, mistakes may range from simple orthographic errors (e.g., positioning of Hamzas) and keyboard errors to dialectal code switching and spelling variations, making the task more challenging than a generic language correction task. We opt for a character-level neural approach to capture these highly diverse mistakes. While this method is less parallelizable due to the long sequence lengths, it is still more efficient due to the small vocabulary size, making inference and beam search computationally feasible.

3.1 Neural Network Architecture

Given an input sentence $\mathbf{x}$ and its corrected version $\mathbf{y}$, the objective is to model $P(\mathbf{y}|\mathbf{x})$. The vocabulary can consist of any number of unique tokens, as long as the following are included: a padding token to make input batches have equal length, the two canonical start-of-sentence and end-of-sentence tokens of the encoder-decoder architecture, and an OOV token to replace any character outside the training data during inference. Each character x_i in the source sentence $\mathbf{x}$ is mapped to the corresponding d_{ce}-dimensional row vector $\mathbf{c_i}$ of a learnable $d_{voc} \times d_{ce}$ embedding matrix, initialized with a random uniform distribution with mean 0 and variance 1. For the encoder, we learn d-dimensional representations for the sentence with two gated recurrent unit (GRU) layers (Cho et al., 2014), making only the first layer bidirectional following Wu et al. (2016). Like long short-term memory (Hochreiter and Schmidhuber, 1997), GRU layers are well-known to improve the performance of recurrent neural networks (RNN), but are slightly more computationally efficient than the former.

For the decoder, we use two GRU layers along with the attention mechanism proposed by Luong et al. (2015) over the encoder outputs h_i. The initial states for the decoder layers are learned with a fully-connected $\tanh$ layer in a similar fashion to Cho et al. (2014), but we do so from the first encoder output. During training, we use scheduled sampling (Bengio et al., 2015) and feed the d_{ce}-dimensional character embeddings at every time step, but using a constant sampling probability. While tuning scheduled sampling, we found that introducing a sampling probability provided better results than relying on the ground truth, i.e., teacher forcing (Williams and Zipser, 1989). However, introducing a schedule did not yield any improvement as opposed to keeping the sampling probability constant and unnecessarily complicates hyperparameter search. For both the encoder and decoder RNNs, we also use dropout (Srivastava et al., 2014) on the non-recurrent connections of both the encoder and decoder layers during training.

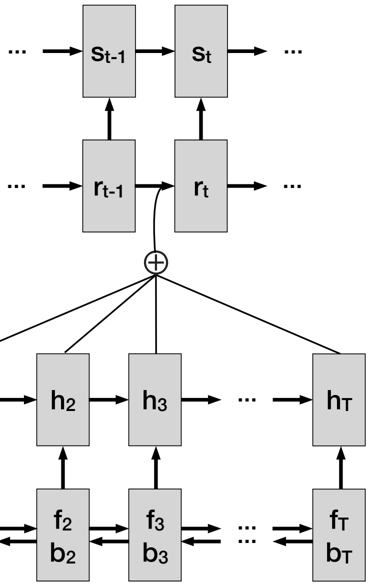

Figure 1: Illustration of the encoder and decoder recurrent layers.

The decoder outputs are fed to a final softmax layer that reshapes the vectors to dimension d_{voc} to yield an output sequence $\mathbf{y}$. The loss function is the canonical cross-entropy loss per time step averaged over the y_i.

3.2 Word Embeddings

To address the challenge posed by the small amount of training data, we propose adding pre-trained word-level information to each character embedding. To learn these word representations, we use FastText (Bojanowski et al., 2016), which extends Word2Vec (Mikolov et al., 2013) by adding subword information to the word vector. This is very suitable for this task, not only because many mistakes occur at the character level,

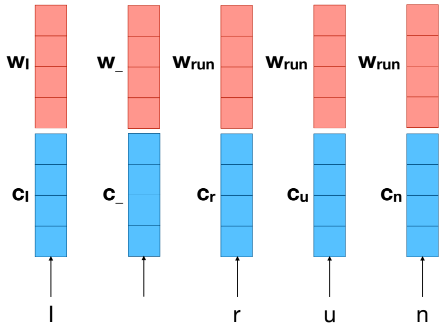

Figure 2: Illustration showing how the character embeddings are enriched with word-level features.

Baseline	P	R	F_1
MLE	77.08	41.56	54.00
MADAMIRA	77.47	32.10	45.39
MLE then MADAMIRA	64.38	38.42	48.12
MADAMIRA then MLE	73.55	44.61	**55.54**

Table 1: Baselines scores on the QALB 2014 shared task development set.

but also because FastText handles almost all OOVs by omitting the Word2Vec representation and simply using the subword-based representation. It is possible, yet extremely rare that FastText cannot handle a word– this can occur if the word contains an OOV n-gram or character that did not appear in the data used to train the embeddings. It should also be noted that these features are only fed to the encoder layer; the decoder layer only receives d_{ce}-dimensional character embeddings as inputs, and the softmax layer has a d_{voc}-dimensional output.

Each character embedding $\mathbf{c_i}$ is replaced by the concatenation $[\mathbf{c_i}; \mathbf{w_j}]$ before being fed to the encoder-decoder model, where $\mathbf{w_j}$ is the d_{we}-dimensional word embedding for the word in which $\mathbf{c_i}$ appears in. This effectively handles almost all cases except white spaces, in which we just always append a d_{we}-dimensional vector $\mathbf{w\_}$ initialized with a random uniform distribution of mean 0 and variance 1. For OOVs, we just append the whitespace embedding $\mathbf{w\_}$ to the word's characters.

3.3 Inference

During inference, the decoder layer uses a beam search, keeping a fixed number (i.e., beam width) of prediction candidates with the highest log-likelihood at each step. Whenever an "end-of-sentence" token is produced in a beam, the decoder stops predicting further tokens for it. We then pick the individual beam with the highest overall log-likelihood as our prediction. As a final step, we reduce text sequences that are repeated six or more times to a threshold of 5 repetitions (e.g., `"abababababab"` → `"ababababab"`). This helps address rare cases where the decoder misbehaves and produces non-stop repetitions of text, and also helps avoid extreme running times for the NUS MaxMatch scorer (Dahlmeier and Ng, 2012), which we use for evaluation and comparison purposes.

4 Evaluation

4.1 Data

We tested the proposed approach on the QALB dataset, a corpus for Arabic language correction and subject of two shared tasks (Zaghouani et al., 2014; Mohit et al., 2014; Rozovskaya et al., 2015). Following the guidelines of both shared tasks, we only used the training data of the QALB 2014 shared task corpus (19,411 sentences). Similarly, the validation dataset used is only that of the QALB 2014 shared task, consisting of 1,017 sentences. We use two blind tests, one from each year. During training, we only kept sentences of up to 400 characters in length, resulting in the loss of 172 sentences.

4.2 Metric

Like in the QALB shared tasks, we use the *MaxMatch* scorer to compute the optimal word-level edits that map each source sentence to its respective corrected sentence. We report the F_1 score of these edits against those provided in the gold data by the same tool. We compare against the best reported system in the QALB 2014 shared task test set (CLMB) (Rozovskaya et al., 2014), as well as the best in the QALB 2014 shared task development and the QALB 2015 shared task test sets (CUFE) (Nawar, 2015).

4.3 Baselines

We consider two different baselines. The first is the output from MADAMIRA (Pasha et al., 2014), a tool for morphological analysis and disambiguation of Arabic. The second is using maximum likelihood estimation (MLE) based on the counts of the MaxMatch gold edits from the training data; that is, each word or phrase gets either replaced

Model	P	R	F_1
Wide embeds	80.80	59.80	68.73
+ preprocessing	79.63	58.81	67.57
Narrow embeds	80.00	62.46	**70.15**
+ preprocessing	80.25	57.80	67.20
Concat embeds	80.74	61.10	69.56
+ preprocessing	79.81	58.28	67.37
CUFE (Nawar, 2015)	–	–	68.72

Table 2: System scores on the QALB 2014 shared task development dataset for the different FastText embeddings.

or kept as is, depending on the most common action in the training data. We found that, unlike Eskander et al. (2013) suggested, first using MADAMIRA and then MLE yields better results than composing these in the reverse order. The results are presented in Table 1.

4.4 Model Settings

In all experiments, we used batch and character embedding sizes of $b = d_{ce} = 128$, hidden layer size of $d = 256$, dropout probability of 0.1, decoder sampling probability of 0.35, and gradient clipping with a maximum norm of 10. When running all the trained models during inference, we used a beam width of 5. We used the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0005, $\epsilon = 1 \cdot 10^{-8}$, β_1=0.9, and β_2=0.999, and trained the model for 30 epochs. We report three different setups with FastText word embeddings: narrow, wide, and the concatenation of both. For each of these, we report results on two separately trained models: one without preprocessing, and one with MADAMIRA and then MLE preprocessing to the inputs. We also report an ablation study where we choose the best of these six trained models and compare against two separately trained models with identical setups, but using Word2Vec and no word-level features, respectively.

All the word embeddings used are of dimension $d_{we} = 300$, and were trained with a single epoch over the Arabic Gigaword corpus (Parker et al., 2011). In the experiments including preprocessing, the respective word vectors were obtained from Gigaword preprocessed with MADAMIRA. The narrow and wide word embeddings were trained with context windows of sizes 2 and 5, respectively. All other hyperparameters were kept to the default FastText values, except the minimum n-gram size, which was reduced from 3 to 2 to compensate for single-character prefixes and suffixes that appear in Arabic when omitting the short vowels (Erdmann et al., 2018).

Model	P	R	F_1
No word embeds	81.55	56.13	66.49
Word2Vec	**82.16**	51.53	63.33
FastText	80.00	**62.46**	**70.15**

Table 3: Ablation tests on the QALB 2014 shared task development dataset. All settings used no preprocessing and narrow word embeddings.

4.5 Results

Development set results are shown in Tables 2 and 3, test set results in Table 4. In all models, training without preprocessing consistently yielded better results than their analogues with the inputs pre-fed to MADAMIRA and then MLE. All the FastText embeddings setups with no preprocessing outperformed the previous state-of-the-art results in the development dataset. We hypothesize that this is occurs because the model has access to more examples of errors to normalize during training, thereby increasing performance. The best performing model was that with the narrow word embeddings; consistent with the results of Zalmout et al. (2018) showing the superior performance of narrow word embeddings over both wide embeddings and the concatenation of both. This is justified by Goldberg (2015) and Trask et al. (2015), who illustrate that wider word embeddings tend to capture more semantic information, while narrower word embeddings model more syntactic phenomena.

In our ablation study, we compared the performance of the narrow FastText embeddings against narrow Word2Vec embeddings trained over the same Arabic Gigaword corpus with the same hyperparameters, as well as to no word-level embeddings at all. The results, displayed in Table 3, show that using only Word2Vec slightly increases precision but significantly hurts recall. This highlights the effectiveness of using FastText for text normalization, as well as the need to handle OOVs in a noisy context for word-level representations to help in this particular problem. Despite that having OOV cases can help the model by indicating that a word is likely erroneous, this does not provide linguistic information the way FastText does. The narrow FastText embeddings with no prepro-

	(2014)	(2015)
Model	F_1	F_1
MADAMIRA then MLE	55.56	60.98
CLMB (Rozovskaya et al., 2014)	67.91	–
CUFE (Nawar, 2015)	66.68	72.87
Narrow embeds	**70.39**	**73.19**

Table 4: System score on the QALB 2014 and QALB 2015 shared task test datasets.

cessing setup achieved state-of-the-art results in all three datasets, beating all systems in both the 2014 and 2015 QALB shared tasks in F_1 score.

4.6 Error Analysis

We conducted a detailed error analysis of 101 sentences from the development set (6,370 words). The sample contained 1,594 erroneous words (25%). The errors were manually classified in a number of categories, which are presented in Table 5. The Table indicates the percentage of the error type in the whole set of errors as well as the error-specific F_1 and an example. Some common problems, Hamza (glottal stop) and Ta Marbuta (feminine ending), are well handled in our best system. This is due to their commonality in the training data. Other types are less common – dialectal constructions, consonantal switches and Mood/Case. Punctuation is very common, however it is also very idiosyncratic. We also note the presence of a small percentage (0.5%) of gold errors. For more information on Arabic language orthography issues from a computational perspective, see (Buckwalter, 2007; Habash, 2010; Habash et al., 2012).

Gold%	**Error Type**	F_1	**Example**
4.8	Ta Marbuta	95.4	كتابه → كتابة
29.8	Hamza	92.8	تاييد → تأييد
10.5	Space	87.5	ماسبب → ما سبب
0.8	Alif Maqsura	83.3	التى → التي
0.7	Repeated Letter	81.8	الرجاااال → الرجال
0.6	Wa of Plurality	66.7	قالو → قالوا
39.3	Punctuation	56.4	$NIL \rightarrow .$
2.2	Multiple Types	43.1	ألقيامه → القيامة
1.7	Consonant Switch	41.0	شحص → شخص
1.6	Other Types	38.3	التقل → القتل
2.3	Mood & Case	33.3	مصريون → مصريين
2.8	Dialect	32.8	هيكتب → سيكتب
1.3	Deleted Letter	n/a	سيتصر → سينتصر
1.1	Grammar	n/a	يتجاوز → يتجاوزون
0.5	Gold Error	n/a	اللتى → التى

Table 5: Error analysis on a sample from the QALB 2014 shared task development set, ordered by F_1 score.

5 Conclusion and Future Work

We propose a novel approach to text normalization by enhancing character embeddings with word-level features that model subword information and model syntactic phenomena. This significantly improves the neural model's recall, allowing the correction of more complex and diverse errors. Our approach achieves state-of-the-art results in the QALB dataset, despite it being small and seemingly unsuited for a neural model. Moroever, our neural model is sophisticated enough to not benefit from preprocessing techniques that reduce the number of errors in the data. Our approach is general enough to be implemented for any other text normalization task and does not rely on domain-specific knowledge to develop.

Future directions include expanding the number of training pairs via synthetic data generation, where generative models can potentially add human-like errors to a large, unannotated corpus. Different sequence-to-sequence architectures, such as the Transformer module (Vaswani et al., 2017), could also be explored and researched more exhaustively. The word-level features provided by FastText could also be replaced by separately trained neural approaches that generate word embeddings from a word's characters (e.g., ELMo embeddings, Peters et al. 2018), and could also be fine-tuned towards specific applications. Another interesting direction includes hybrid word-character architectures, where the encoder receives word-level features, while the decoder operates at the character level. We are also interested in applying our approach to other languages and dialects.

Acknowledgment

The second author was supported by the New York University Abu Dhabi Global PhD Student Fellowship program. The support and resources from the High Performance Computing Center at New York University Abu Dhabi are also gratefully acknowledged.

References

Gheith A. Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of Arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):183–197.

Sina Ahmadi. 2017. Attention-based encoder-decoder networks for spelling and grammatical error correction. Master's thesis, Paris Descartes University, 9.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2281–2285.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Tim Buckwalter. 2007. Issues in Arabic Morphological Analysis. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012. A beam-search decoder for grammatical error correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 568–578. Association for Computational Linguistics.

Alexander Erdmann, Nasser Zalmout, and Nizar Habash. 2018. Addressing Noise in Multidialectal Word Embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing Spontaneous Orthography. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726.

Nizar Habash, Mona Diab, and Owen Rambow. 2012. Conventional Orthography for Dialectal Arabic. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 711–718, Istanbul, Turkey.

Nizar Y Habash. 2010. *Introduction to Arabic natural language processing*, volume 3. Morgan & Claypool Publishers.

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

Taishi Ikeda, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Japanese text normalization with encoder-decoder model. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 129–137.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. 2014. The first QALB shared task on automatic text correction for Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 39–47.

Michael Nawar. 2015. CUFE@QALB-2015 shared task: Arabic error correction system. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, pages 133–137, Beijing, China.

Robert Parker, David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2011. Arabic Gigaword Fifth Edition. LDC catalog number No. LDC2011T11, ISBN 1-58563-595-2.

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alla Rozovskaya, Houda Bouamor, Nizar Habash, Wajdi Zaghouani, Ossama Obeid, and Behrang Mohit. 2015. The second QALB shared task on automatic text correction for arabic. In *Proceedings of the*

Second Workshop on Arabic Natural Language Processing, pages 26–35.

Alla Rozovskaya, Nizar Habash, Ramy Eskander, Noura Farra, and Wael Salloum. 2014. The Columbia system in the QALB-2014 shared task on Arabic error correction. In *ANLP@EMNLP*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Andrew Trask, David Gilmore, and Matthew Russell. 2015. Modeling order in neural word embeddings at scale. *arXiv preprint arXiv:1506.02338*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ÅĄukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Daniel Jurafsky, and Andrew Y. Ng. 2016. Neural language correction with character-based attention. *CoRR*, abs/1603.09727.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386.

Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Ossama Obeid, Nadi Tomeh, Alla Rozovskaya, Noura Farra, Sarah Alkuhlani, and Kemal Oflazer. 2014. Large scale Arabic error annotation: Guidelines and framework. In *International Conference on Language Resources and Evaluation (LREC 2014)*.

Nasser Zalmout, Alexander Erdmann, and Nizar Habash. 2018. Noise-Robust Morphological Disambiguation for Dialectal Arabic. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Nasser Zalmout and Nizar Habash. 2017. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 704–713, Copenhagen, Denmark. Association for Computational Linguistics.

Topic Intrusion for Automatic Topic Model Evaluation

Shraey Bhatia[1] **Jey Han Lau**[1,2] **Timothy Baldwin**[1]

[1] School of Computing and Information Systems,
The University of Melbourne
[2] IBM Research Australia

shraeybhatia@gmail.com, jeyhan.lau@gmail.com, tb@ldwin.net

Abstract

Topic coherence is increasingly being used to evaluate topic models and filter topics for end-user applications. Topic coherence measures how well topic words relate to each other, but offers little insight into the utility of the topics in describing the documents. In this paper, we explore the topic intrusion task — the task of guessing an outlier topic given a document and a set of topics — and propose a method to automate it. We improve upon the state-of-the-art substantially, demonstrating its viability as an alternative method for topic model evaluation.

1 Introduction

Topic models have traditionally been evaluated using model perplexity, but there is an increasing trend to use topic coherence as a task-independent evaluation (Newman et al., 2010; Mimno et al., 2011; Aletras and Stevenson, 2013; Lau et al., 2014; Röder et al., 2015). In earlier work (Bhatia et al., 2017), we showed that topic coherence as a standalone evaluation can be misleading, which we illustrated with an adversarial topic model that produces highly coherent topics that collectively tell us little about the content of the document collection.

We went on to explore an alternative approach to assessing topics using topic intrusion, based on the manual task of Chang et al. (2009). In the original topic intrusion setup, users are presented with a document, a set of associated topics (from a topic model) and an intruder topic, and are tasked to find the intruder. Success in the task demonstrates that the topics learnt by the topic model are relevant to the document. We proposed a method to automate this (Bhatia et al., 2017), by training a support vector regression model based on information retrieval (IR) and word co-occurrence features to predict the intruder topic.

Although our earlier method is able to distinguish between good and bad topic models (at the model-level), we provided no evaluation at the document level other than the observation that "there are still slight disparities between human annotators and the automated method in intruder topic selection". Additionally, the method involves a number of dependencies on complex external systems such as Indri, and no implementation of the method was ever released. In this paper, we extend our earlier work (Bhatia et al., 2017) as follows: (1) we improve the results based on a novel neural model and provide additional analysis of document-level evaluation via mean-absolute-error; (2) we propose a new metric to measure the performance of the system; and (3) we release an open source implementation of our system.[1]

2 Related Work

Chang et al. (2009) introduced the word and topic intrusion tasks to assess topic models. Since then, various automatic measures to assess topics have been proposed (Newman et al., 2010; Mimno et al., 2011; Aletras and Stevenson, 2013). Lau et al. (2014) compared and contrasted these approaches, and proposed a variant method based on normalised pointwise mutual information. Röder et al. (2015) conducted a systematic search using a framework that combines various existing measures.

In Bhatia et al. (2017), we revisited the topic intrusion task of Chang et al. (2009), and explored its viability as an alternative task-independent approach for topic model evaluation. We tested a number of topic models and found that there can be large discrepancies between conventional topic coherence measures and topic intrusion results, suggesting that topics can be individually coherent but

---

[1] Source code and dataset can be downloaded at: `https://github.com/sb1992/Topic-Intrusion-for-Automatic-Topic-Model-Evaluation`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 844–849
Brussels, Belgium, October 31 - November 4, 2018.

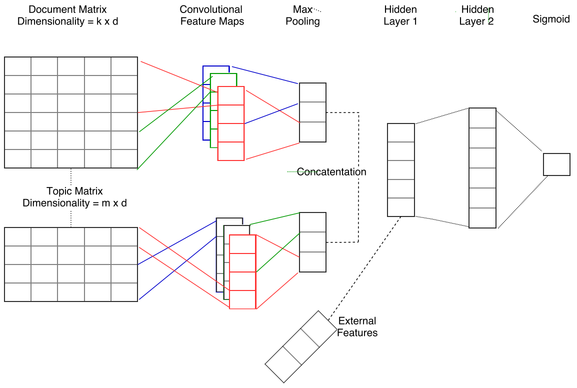

Figure 1: Architecture diagram of our method

poor descriptors of the documents. In addition, we proposed a method to automate the topic intrusion task and reported encouraging correlation levels with human judgements for model-level evaluation.

3 Datasets and Topic Models

We conduct our experiments using the datasets of Bhatia et al. (2017): (1) APNEWS, a collection of Associated Press news articles; and (2) the British National Corpus ("BNC": Burnard (1995)), made up of excerpts from diverse sources such as journals, books, letters, and articles. For the topic models we experiment with the following: standard LDA (`lda`: Blei et al. (2003)), correlated topic model (`ctm`: Blei and Lafferty (2006)), non-parametric topic model (`hca`: Buntine and Mishra (2014)), neural topic model (`ntm`: Cao et al. (2015)), and an adversarial topic model (`cluster`: Bhatia et al. (2017)). `cluster` is adversarial in the sense that it is designed to produce topics that are coherent but poor descriptors of documents.

4 Methodology

In this section, we briefly describe the topic intrusion task and propose an improved methodology to automate it.

4.1 Task

Chang et al. (2009) first proposed the topic intrusion task with the aim of assessing whether topics associated with a document capture its content. In this task, an annotator is presented with a document along with its top-3 highest probability topics and a low probability intruder topic, and are asked to identify the outlier intruder topic. Bhatia et al. (2017) incorporate an additional constraint: the intruder topic has to have high probability for at least one other document. Their rationale is to ensure that the intruder is interpretable. We follow the approach of our earlier work (Bhatia et al., 2017) when generating intruder topics.

4.2 Human Judgements

To assess our methodology, we need human annotations for the topic intrusion task. We collect human judgements using Amazon Mechanical Turk. Each HIT is comprised of 5 documents, and each document is paired with 4 topics (3 real and 1 intruder). To control for annotation quality, an additional document–topics pair is inserted as part of the HIT. The control item's intruder topic is generated by randomly sampling words from the corpus vocabulary. To pass the quality control, an annotator has to select the correct intruder topic; they are filtered out if their pass rate over all controls is < 0.6.[2]

Each HIT is judged by 10 workers. We collect additional annotations by releasing the task internally to a small number of local workers. We needed to carry out some annotations internally to make sure that each HIT had at least 4 annotations. The average number of internal annotations was approximately 1.6. For each topic model, we collected annotations for 100 documents on 2 corpora (5 topic models $\times$ 100 documents $\times$ 2 collections $=$ 1000). After filtering and including internal judgements we have an average of 6.7 and 6.9 annotations for APNEWS and BNC, respectively.

4.3 Intruder Topic Detection

We propose a neural network model to automatically predict intruder topics. Our model is inspired by Severyn and Moschitti (2015), where they combine a learn-to-rank deep learning architecture in an IR setting to rank the documents for a given query. We adapt it to our topic intrusion task by ranking topics for a given document. Our task takes the form of a document d_i with corresponding topics $T_i = \{t_i^1, t_i^2, t_i^3, t_i^4\}$, where 3 topics are real and 1 is the intruder. The topic set T_i has labels $Y_i = \{y_i^1, y_i^2, y_i^3, y_i^4\}$ ("1" denotes the intruder topic, or "0" otherwise). We train using a pointwise ranking approach, where training examples

---

[2] We fixed the threshold to 0.6 based on preliminary experiments. We found that it was a challenging task, and this value provides quality without filtering out most of the workers.

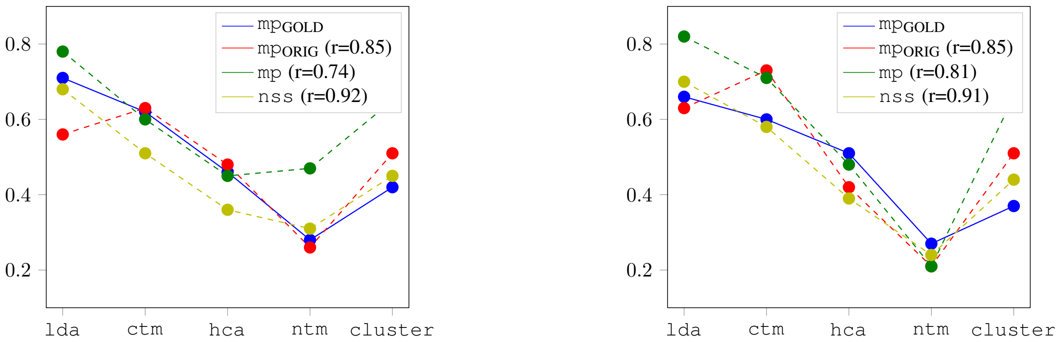

(a) APNEWS

(b) BNC

Figure 2: $\texttt{mp}_{\text{GOLD}}$ vs. System Scores at the model level

are triples of (d_i, t_i^j, y_i^j) — essentially the task is formulated as a binary classification problem.

The architecture of our network is given in Figure 1. The input to our model is a document–topic pair, with each represented as a sequence of words. These words are mapped to embeddings, via embedding matrix $W \in \mathbb{R}^{|V| \times d}$, where V is the vocabulary and d the dimensionality of the embeddings. The document embeddings $E_d \in \mathbb{R}^{k \times d}$ (k = document length) and topic embeddings $E_t \in \mathbb{R}^{m \times d}$ (m = number of topic words) are processed via convolutional layers (Kim, 2014; Severyn and Moschitti, 2015) to produce two hidden representations for the document and topic. The convolution operation is performed using feature maps of varying size followed by a max-pooling operation to produce a constant-length vector. The document and topic hidden representations are concatenated and fed to 2 dense layers and ultimately reduced to a sigmoid-activated score.

4.3.1 External IR Feature

A good topic model learns common themes in the document collection. A limitation of our network is the lack of global- or collection-level information (as the input consists of only a document and topic). To incorporate collection-level information, we include an IR feature where we query document d_i using the topic words of t_i^j. We use Okapi BM25 (Robertson and Walker, 1994) to compute the relevance score of the document with respect to its N topic words independently, thereby constructing an N-dimensional feature vector.[3] This external feature vector is incorporated into the network after the convolutional layers (see Figure 1).

Model	BNC → APNEWS			APNEWS → BNC		
	$\texttt{mp}_{\text{ORIG}}$	mp	nss	$\texttt{mp}_{\text{ORIG}}$	mp	nss
lda	0.47	0.31	**0.21**	0.40	0.32	**0.22**
ctm	0.44	0.34	**0.20**	0.41	0.31	**0.19**
hca	0.48	0.37	**0.21**	0.42	0.35	**0.20**
ntm	0.40	0.43	**0.19**	0.37	0.32	**0.18**
cluster	0.48	0.42	**0.19**	0.51	0.47	**0.22**
Overall	0.46	0.37	**0.20**	0.42	0.36	**0.21**

Table 1: mae between $\texttt{mp}_{\text{GOLD}}$ and nss/mp. "BNC → APNEWS" means the model is trained on BNC and tested on APNEWS. Boldface indicates optimal performance for each dataset.

4.4 Aggregating Human and System Scores for a Document

For each document we have a number of workers identifying the intruder topic. To aggregate the results, Chang et al. (2009) define model precision ($\texttt{mp}_{\text{GOLD}}$), which is the proportion of workers who correctly identified the intruder, as a proxy for how clearly the intruder topic is inappropriate for the document.

Our system and that of Bhatia et al. (2017) compute several scores for a document (one for each topic). Bhatia et al. (2017) select the topic with the maximum score as the intruder, and compute model precision (mp) based on that. This yields binary precision scores (i.e. the model either predicts the intruder correctly or not) and ignores the relative magnitude of the system score. We additionally propose using the normalised sigmoid score (nss) as a means of scoring the intruder topic for a given document, which is computed by normalising the raw sigmoid scores over all topics.

---

[3] $N = 5$ in our experiments.

Model	Best/Worst Topics	`nss`
`lda`	share revenue cents billion quarter earnings analysts net rose income	0.001
	european greece europe billion debt country crisis minister french france	0.002
`ctm`	building lodge bauer buildings fee part stephens hall property council	0.007
	military army afghanistan killed soldiers forces troops iraq war attacks	0.013
`hca`	shares earnings keywords insights profit thomson cents reuters premarket net	0.011
	upheld ruling appeals justices appellate supreme injunction plaintiffs unconstitutional rulings	0.051
`ntm`	rose shrank pct decliners quadrupled exhibitors parade spectrum index outperform	0.110
	arraigned burglarizing arrested bigamy detectives motorcyclist arraignment coroner accomplice fondled	0.141
`cluster`	soared plummeted climbed surged dipped tumbled dropped fell slipped rose	0.005
	students teachers kindergarten tutors elementary coursework curriculum teaching tutoring education	0.013
`lda`	lot good things long put start number making kind place	0.291
	political issue called issues policy decision long change statement support	0.271
`ctm`	online information internet book video media facebook phone computer technology	0.263
	show music film movie won festival tickets game band play	0.233
`hca`	richter riverboat sheppard lander plazas tam mandarin amarillo colosseum nassau	0.376
	deplorable interaction foresee envelope handwriting knot quickest scrambled alarmed mum	0.368
`ntm`	aboard spacewalks bushels budget lifeboats flotilla lifeboat spacewalk millage spaceflight	0.364
	evacuated evacuations evacuate evacuating airlifted twisters aftershocks evacuation driest barricaded	0.323
`cluster`	accord delegations accords cooperation consultations negotiators negotiation committees intergovernmental negotiations	0.323
	summaries summary critiques excerpts articles responses quotes references descriptions critique	0.309

Table 2: Examples of best topics (top-half) and worst topics (bottom-half) based on `nss`.

4.4.1 Implementation Details

For our experiments, we train the model on outputs from all topics models over one dataset, and test it on the other (cross-domain training). We use a single channel for the convolutional networks, pad the documents as necessary ($k = 200$), and use the top-10 words to represent a topic (i.e. $m = 10$). Word embeddings are initialised using pre-trained GloVe (Pennington et al., 2014) vectors ($d = 100$), and their weights are fixed during training. We use kernel windows of width $= \{3, 5, 7\}$ with 100 feature maps each and two (fully-connected) hidden layers, with dimensionality of 50 and 10. We use a dropout rate of 0.5, 0.5 and 0.25 after the document, topic and first hidden layer, respectively. We set the batch size to 100, and use Adam as the optimizer with a learning rate of 0.001. For activation functions, we use ReLU for the fully-connected layers and sigmoid for the final layer. To reduce variance, we run the models with 8 different seeds for initialisation and take the average for a topic's sigmoid score.

5 Results

By taking the mean of $\texttt{mp}_{\text{GOLD}}$ and `mp` over documents, Bhatia et al. (2017) compute a single human/system score for each topic model. Although this resulted in a strong correlation between $\texttt{mp}_{\text{GOLD}}$ and `mp`, the evaluation is limited to model-level comparison: it separates good topic models from bad topic models, but does not provide any insights into the performance of each top model over individual documents. We aim to improve model-level correlation in this work, in addition to analysing document-level evaluation, i.e. investigating how well the system predicts $\texttt{mp}_{\text{GOLD}}$ for each individual document.

We present plots of human and system scores in Figure 2. There are 3 system scores: `mp` of Bhatia et al. (2017) ($\texttt{mp}_{\text{ORIG}}$), and `mp` and `nss` of our proposed system. In general, we found strong correlation for all systems, but `nss` of our proposed system performs substantially better than $\texttt{mp}_{\text{ORIG}}$, though our `mp` is lower than $\texttt{mp}_{\text{ORIG}}$.

To compare the performance of our system with human judgements at the document level, we compute mean absolute error (`mae`) between $\texttt{mp}_{\text{GOLD}}$ and `nss`/`mp`, as summarised in Table 1. We find for both datasets `nss` consistently outperforms $\texttt{mp}_{\text{ORIG}}$ and `mp` by a substantial margin, and also has a score close to human judgements. We can attribute this to the fact that `nss` provides more nuanced system predictions (over the full range $[0, 1]$), whereas `mp` tends to be binary.[4]

6 Discussion

One motivation we have in this paper is to test whether topic intrusion can be used as an alternative means for assessing topics. Given the encouraging

---

[4] Strictly speaking, it is continuous as it is averaged over the runs for the multiple random seeds, but in general, it tends to be (close to) 0 or 1.

`mae` results, we attempt to use `nss` to rank topics produced by a topic model.

To accomplish this, we first filter out the topics that occur in less than 5 documents as top 1-topic: these topics tend to be noisy, and as such do not appear with significant weight in any documents. For each of the filtered topics we randomly select 5–10 documents for which it is a top topic and calculate its mean `nss` over these documents. We then use the topics' mean `nss` to rank them; in Table 2 we show some selected best and worst topics for different topic models. Overall, the top-ranked topics appear to be more descriptive than the bottom-ranked topics. Having said that, we found instances where coherent topics have low `nss` ranking (e.g. `ctm` topics in the bottom half of Table 2), but stress that ultimately the topic intrusion approach to assessing topics is very different to topic coherence. We include a more comprehensive list of best/worst topics in the supplementary material.

7 Conclusion

We explore an alternative approach to evaluate topic models based on topic allocations in documents, i.e. via topic intrusion. We propose an automated method that improves upon the state-of-the-art substantially at the model- and document-levels, and demonstrate that it can be used to rank/filter topics. As future work we intend to explore ways that combine both the topic coherence and topic intrusion for topic model evaluation.

References

Nikos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In *Proceedings of the Tenth International Workshop on Computational Semantics (IWCS-10)*, pages 13–22, Potsdam, Germany.

Shraey Bhatia, Jey Han Lau, and Timothy Baldwin. 2017. An automatic approach for document-level topic model evaluation. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 206–215. Association for Computational Linguistics.

David Blei and John Lafferty. 2006. Correlated topic models. *Advances in Neural Information Processing Systems*, 18.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Wray L Buntine and Swapnil Mishra. 2014. Experiments with non-parametric topic models. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 881–890.

Lou Burnard. 1995. User guide for the British National Corpus.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Proceedings of AAAI 2015*, pages 2210–2216.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems 21 (NIPS-09)*, pages 288–296, Vancouver, Canada.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of EACL 2014*, pages 530–539.

David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 262–272, Edinburgh, UK.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

S.E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-Poisson Model for probabilistic weighted retrieval. In *Proceedings of 17th International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 232–241.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM 2015)*, pages 399–408, Shanghai, China.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research* *and Development in Information Retrieval*, pages 373–382. ACM.

Supervised and Unsupervised Methods for Robust Separation of Section Titles and Prose Text in Web Documents

Abhijith Athreya Mysore Gopinath[1], **Shomir Wilson**[1] **and Norman Sadeh**[2]
[1]College of IST, Pennsylvania State University, University Park, PA 16802, USA
[2]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
abhijith@psu.edu, shomir@psu.edu, sadeh@cs.cmu.edu

Abstract

The text in many web documents is organized into a hierarchy of section titles and corresponding prose content, a structure which provides potentially exploitable information on discourse structure and topicality. However, this organization is generally discarded during text collection, and collecting it is not straightforward: the same visual organization can be implemented in a myriad of different ways in the underlying HTML. To remedy this, we present a flexible system for automatically extracting the hierarchical section titles and prose organization of web documents irrespective of differences in HTML representation. This system uses features from syntax, semantics, discourse and markup to build two models which classify HTML text into section titles and prose text. When tested on three different domains of web text, our domain-independent system achieves an overall precision of 0.82 and a recall of 0.98. The domain-dependent variation produces very high precision (0.99) at the expense of recall (0.75). These results exhibit a robust level of accuracy suitable for enhancing question answering, information extraction, and summarization.[1]

1 Introduction

Web text continues to be an immense resource for researchers working in NLP and related areas, but its typographic structure (i.e., its visual organization) remains underutilized. Many texts on the web are organized into sections based on the topics presented, and each section has a title followed by prose text. The title tends to be visually distinct to separate it from the prose that succeeds it. Apart from improving the readability of the page, this explicit organization makes titles act as intuitive indexes for the prose text(s) that follow them. In other words, titles are concise summaries of the following prose text.

Most current methods of web text extraction do not separate titles from prose text, instead treating the entire document as a single unit without internal structure. However, this internal organization has the potential to provide input to a variety of NLP tasks that can make use of information on topicality or discourse structure. In the case of question answering, this information can facilitate identifying maximally relevant sections to search for answers. It can also increase accuracy by filtering out false positives and minor references related to the topic of the question, which might be present in other sections. For information retrieval tasks, semantic matching of the search terms can be performed on the titles first, followed by a search on the prose texts associated with closely matching headers.

However, detecting the titles and prose segments in an HTML document is difficult for two reasons. One of them is the flexibility of HTML, which allows the same typographic layout to be represented in code in multiple ways. Tags are also nested with varying depths. Figure 1 illustrates this problem: similar title and prose text segments from four website privacy policies[2] have altogether different HTML tag structures. The second problem is that it is not straightforward to distinguish the information (encoded in HTML) that is necessary for title-prose detection from the rest of the HTML structure, including unrelated links, multiple tags with little or no content and page headers and footers. Sieving only useful information from these pages requires a flexible approach.

---

[1] The source code and corpora generated by this research are available at https://github.com/abhijith-athreya/ASDUS.

[2] All the policies were retrieved on 2018-01-20, from the below URLs:
https://rule.alibaba.com/rule/detail/2034.htm
https://www.apple.com/legal/privacy/en-ww/
https://www.cbsinteractive.com/legal/cbsi/privacy-policy
https://help.bet365.com/en/privacy-policy

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 850–855
Brussels, Belgium, October 31 - November 4, 2018.

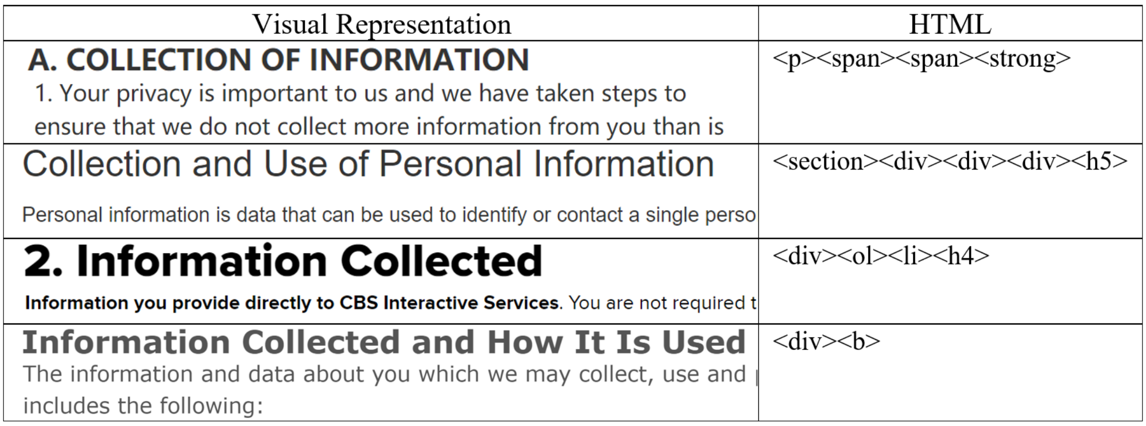

Visual Representation	HTML
A. COLLECTION OF INFORMATION 1. Your privacy is important to us and we have taken steps to ensure that we do not collect more information from you than is	<p><span><span><strong>
Collection and Use of Personal Information Personal information is data that can be used to identify or contact a single perso	<section><div><div><div><h5>
2. Information Collected **Information you provide directly to CBS Interactive Services**. You are not required t	<div> <h4>
Information Collected and How It Is Used The information and data about you which we may collect, use and includes the following:	<div><b>

Figure 1: Excerpts from the website privacy policies of Alibaba.com, Apple, CBS Interactive and bet365. At left are browser renderings, and at right are the XPaths of the section titles.

We present ASDUS (Automatic Segment Detection using Unsupervised and Supervised learning), a system that uses a variety of features of text and markup structure to identify the title and prose organization of HTML documents automatically. Our approach effectively strips away all unrelated HTML and produces simplified HTML with a predictable tag structure, thus making the extraction of section titles and prose text straightforward for downstream applications. We present two approaches, a domain-independent approach that requires no prior training and a domain-dependent approach that takes advantage of a labeled corpus. The domain-independent approach (abbreviated *DI*) yields an overall precision of 0.82, recall of 0.98 and coverage of 97.21%. The domain dependent approach (*DD*) produces an overall precision of 0.99, recall of 0.75 and coverage of 93.10%. We are releasing ASDUS and associated datasets to the research community for use and improvement.

2 Related Work

The detection of titles and prose in web documents has received little (if any) prior attention, but we briefly survey literature in some related areas.

Titles can be thought of as metalinguistic descriptions for the prose text they are associated with. Wilson (2013) attempted to identify a core metalinguistic vocabulary for the English language and to automatically identify instances of metalanguage usage. Deixis present in a text can also be considered metadata and detection of deixis helps in structuring the flow of information. Wilson and Oberlander (2014) attempted to capture word senses related to deixis.

Topic classification is the problem of segregating a document into different topics, and argumentative zoning (Teufel et al., 1999) was an early effort that shares some goals with the present work, as it addressed the detection of the main thematic areas in research articles. Teufel and Kan (2011) built a robust argumentative zoning system which used maximum entropy modeling to go with morphological features. Conditional random fields were adopted for categorization of sentences of a scientific abstract into different sections by Hirohata et al. (2008). Using posterior discourse and lexical constraints as features, Guo et al. (2013) improved upon existing information structure analysis of scientific documents through unsupervised and minimal supervised learning.

HTML structure analysis is the process of extracting useful information by utilizing the underlying HTML document structure. Information extraction from HTML using machine learning was introduced in SRV (Freitag, 1998), a top-down relational algorithm for information extraction. This system aimed at filling pre-defined slots for a web page in a particular domain. A set of extraction rules suitable to extract information from a website is called a wrapper (Flesca et al., 2004). Dalvi et al. (2011) worked on enhancing wrapper induction techniques by introducing a generic framework which allows for training on noisy data. Liao et al. (2015) used web block segmentation and machine learning algorithms to retrieve business event data, such as coupons, tickets, and sales campaigns. García-Plaza et al. (2017) worked on using fuzzy logic to leverage HTML markup for web page clustering. Using four essential features viz., text frequency, title, emphasis and position, they define 31 independent rules to arrive at the importance of a text segment. Unlike our approach, which is independent of the tag structures and learns patterns on its own, these methods depend on handcrafted rules and similar tag structures to identify various sections of the document.

3 Approach

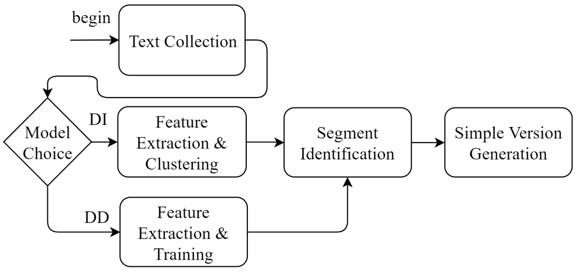

Figure 2: Stages of the DI and DD approaches.

3.1 Domain-Independent Approach (DI)

Figure 2 depicts the stages of this approach, which are explained in detail below.

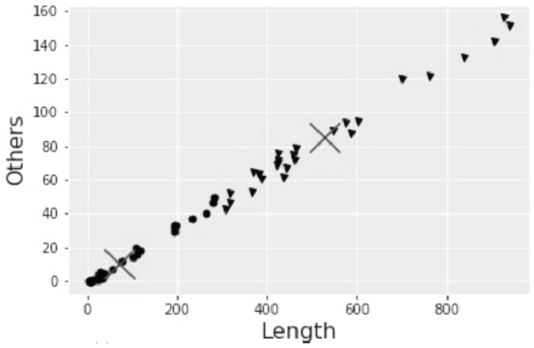

Figure 3: The graph shows two clusters. The one nearer to the origin is the title cluster and the farther one is the prose cluster. Cluster medians are marked with Xs.

Text Collection: Using jsoup (Hedley, 2017), we parse the HTML file and for each non-empty tag encountered we extract a tuple consisting of the text and its XPath.

Feature Extraction and Clustering: For each tuple, we extract a set of features which aids in differentiating titles from prose text. The features are own text length, next text length, number of punctuation symbols, number of sentences, number of stop words (stop words were derived from Weka (Hall et al., 2009)), number of discourse cues, number of named entity slots and number of words with capitalized initial letter. We calculated the number of discourse cues as the sum of explicit discourse markers provided by Denver (2018) and the number of coreference chains. We used Stanford CoreNLP (Manning et al., 2014) for identifying coreference chains and named entities.

Titles tend to contain less text than prose segments, as well as fewer punctuation symbols, stop words, discourse markers, sentences and named entities. Since titles are not followed immediately by titles (in most cases), the next text length feature helps to remove false positives for title identification. All these features are collapsed into two dimensions, one being the text length and the other being the linear combination of remaining features.

K-means clustering (using scikit-learn (Pedregosa et al., 2011)) is performed on the feature set to group it into two mutually exclusive subsets. Figure 3 shows an example plot of the clustering. We obtain two distinct clusters with the title cluster closer to the origin and the prose cluster away from the origin. We leverage this property to obtain the label of the title cluster by identifying the label for the origin ([0,0]). Each text segment (and thus each tuple) is classified into title or prose using the k-means model.

Segment Identification: For each potential title identified in the previous phase, an overlap score is calculated by measuring the overlap between the lemmatized form of words belonging to the title and the lemmatized form of words from the next text segment. Based on experiments on a development set, the overlap threshold was set to 75%. Titles with overlap scores exceeding the threshold and their corresponding XPaths are added to the list of probable title candidates.

Simple Version Generation: Each element in the final title list is marked with a custom attribute in the original HTML. To generate the simple version, a top down parse of the HTML is performed, wherein nodes with custom attribute (titles) are enclosed in <*h2*> tags and the text between two consecutive titles (prose) is enclosed in <*p*> tags. For the last segment, the prose text immediately following the title is added. For subsequent prose sections, a unigram overlap score similar to the one in the previous step is performed to avoid the addition of unrelated text, such as a page footer. Legitimate textual content appearing before the first title is included in the final output under an uncategorized title.

Type of Doc.	# Docs.	# Titles	# Prose
Privacy	152	3611	14506
TOS	100	2299	7818
Misc.	51	685	3676

Table 1: Test set details of the DI approach.

3.2 Domain-Dependent Approach (DD)

The DD approach differs from the DI approach by feature selection: the DD approach trains a neural network classifier on previously labeled examples.

Feature Extraction and Training: To construct labeled data for training, we chose to use web privacy policies due to their lengthy nature, the presence of a reasonable number of segments, the presence of relatively similar content with varying HTML structure across websites and their importance to the general public. We annotated each HTML tag of 100 web privacy policies with one of three labels: title, prose and unrelated. We built two word embedding models (using gensim (Řehůřek and Sojka, 2010)), one using text from the titles and the other using prose texts. Then for each HTML tag, we calculated two semantic relatedness scores: one between the title embedding and the text in the tag (*t-score*), and the other between the prose embedding and the text in the tag (*p-score*). The t-score is the log probability for a text segment with respect to the title embedding. The p-score is the log probability for a text segment with respect to the prose embedding. The intuition behind using these scores is that the t-score will be higher for titles and lower for prose text. Similarly, the p-score will be higher for prose text than titles. The t-score, p-score and length of the text formed the feature set. Using Tensor Flow (Abadi et al., 2015), we trained a feed forward neural network (h1=24, h2=48) to classify text between tags as title, prose or unrelated.

4 Dataset and Results

The problem of automatic detection of titles and prose text in HTML documents has received scant prior attention. Due to this, a corpus containing HTML documents along with their respective annotated versions (titles and prose sections annotated) was unavailable. This lack of data prompted the creation of a new corpus consisting of web documents and their respective annotated versions.

The dataset consisted of three sets of web documents to achieve an exhaustive evaluation of the system. The first set consisted of 152 website privacy policies. We collected 80 of them from Amazon Alexa's top 100 websites list for 2016 and 72 from the top two websites of each top Google Trends entry of 2017. Privacy policies of various companies have different HTML structure. They tend to contain many sections with each section having a title and corresponding prose text. They are also lengthy and include essential information which is often neglected by most users (Wilson et al., 2016). These factors make privacy policies ideal candidates for testing ASDUS. For the second set, we used www.randomlists.com to generate a list of 200 random sites. We selected the websites which had terms of service in English, and this left us with 100 documents. Similar to privacy policies, terms of service documents offer a reliable set of testing opportunities. For the third set, we wanted greater diversity in content and domain, leading us to chose web pages by collecting the top two to four Google search results for the following topics: *news, sports, botany, web design, photography, data science, cookie policies, HTML, history, migraine, dataset, technical documentation, shoes, grammar, kids stories and cricket*. We skipped web pages which did not have sectional demarcations. Table 1 has more details on the size of the dataset. The dataset for the DD approach consisted of the same 152 web privacy policies which were used in the DI approach. Privacy policies of different websites have content related to similar topics which makes them specific and suitable for the DD approach. Out of 152, 122 were used for training and development and the remaining 30 were used for testing. We manually annotated the entire data set and created sanitized versions out of them for evaluation.

We evaluated our models in two ways. One was the ability to detect all title and prose segments, measured via precision, recall and F-1 scores. This metric is evaluated by comparing the simplified output of ASDUS with the sanitized version. The output was deemed correct only when the system detected and produced both the title and its corresponding prose section. The second facet of evaluation was to determine the percentage of legitimate original text reproduced in the output. We name this the *coverage* of the output. A higher coverage indicates lower loss of text, which is desirable.

The results of the DI approach are presented in Table 2. The near-perfect recall is due to the ro-

Type	P	R	F1	S	C
PP	0.75	0.97	0.85	3611	96.92%
TOS	0.92	0.99	0.95	2299	98.85%
Misc	0.89	0.98	0.93	685	95.86%
Avg	0.82	0.98	0.89	6595	97.21%

Table 2: Results of the DI model. Columns represent type of web document, precision, recall, F-1 score, support and coverage respectively. Under *Type*, PP=privacy policies, TOS=terms of service, Misc=miscellaneous and Avg=weighted average.

bust method of learning XPaths of all titles coupled with the clustering method, which ensured the detection of nearly all the segments. Precision is slightly lower because some prose texts with relatively short lengths were wrongly classified as titles. The feature set enables the creation of two distinct clusters, which in turn results in all titles being detected during the clustering phase. To test the effect of each feature on the final results, an ablation study was conducted by removing one different feature from each run. This resulted in a total of seven runs, whose results are depicted in Table 3. The dropped feature is listed in the first column. The biggest drop in performance occurs when the own text length feature is dropped. The drop in F-1 scores in all runs suggests the contribution of every feature towards the result. The removal of discourse markers and named entity slots resulted in the least decrease in performance, and the greatest decrease came from removing text length.

Dropped Feature	P	R	F1
text length	0.69	0.73	0.71
stop words	0.75	0.75	0.75
punctuation symbols	0.76	0.79	0.77
sentences	0.76	0.79	0.77
next text length	0.76	0.81	0.78
capitalized first letter	0.78	0.80	0.79
discourse markers	0.79	0.85	0.81
named entity slots	0.79	0.85	0.81
None	**0.82**	**0.98**	**0.85**

Table 3: Results of the ablation study for the DI approach. The columns indicate dropped feature, precision, recall and F-1 score respectively.

In sharp contrast to the DI approach, the DD approach (Table 4) has high precision owing to the similarity of title texts across documents in the same domain and differences between vocabularies of title and prose text. Training on word embeddings of titles has rendered the system sensitive to variations, resulting in the rejection of many legitimate titles, which in turn led to the slightly lower recall and coverage values. This over-fitting can be mitigated by training on a larger corpus and by increasing the context window while generating word embeddings of prose text.

Precision	Recall	F-1	Supp.	Cover
0.99	0.75	0.86	754	93.10%

Table 4: Results of the DD model.

The complementary nature of the two models is identifiable from the results. The word embeddings of the DD approach contribute towards precision, and the lexical and morphological features of the DI approach contribute towards recall. We can treat the word embeddings as the semantic aspect of the underlying text and the lexical and morphological characteristics of the DI model as the syntactic aspect. Both of these aspects of language thus inform the orthographic structure of documents. However, the domain-independence or the ability of the DI approach to work on different domains without the need for prior labeled data is a substantial advantage over the DD approach, which requires pre-labeled data. Annotation of HTML documents and generation of the sanitized version is a tedious process and requires human effort, lending value to both methods.

5 Future Work and Conclusion

The two methods presented in this paper were effective in identifying the title and prose texts of segments of HTML pages. The DI approach, with its high coverage, is desirable when the penalty for losing text is high. The DD approach can be used in situations where precision is prioritized over recall. In the future, we intend to build upon our methods and enable automatic detection of subheaders. This would lead to the identification of hierarchical organization of the text, which provides a novel approach to generate web ontologies. Further, we have planned to generate titles for prose text essentially creating micro-summaries of text, a relatively unexplored area.

Acknowledgment

The work reported herein was conducted as part of the Usable Privacy Policy Project (Sadeh et al., 2013) and funded by the National Science Foundation under grant CNS-1330596.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Nilesh Dalvi, Ravi Kumar, and Mohamed Soliman. 2011. Automatic wrappers for large scale web extraction. *Proceedings of the VLDB Endowment*, 4(4):219–230.

Tanya Denver. 2018. Discourse Markers – Connectors a list of discourse markers with examples. `https://www.academia.edu/6888756/discourse_markers_connectors_a_list_of_discourse_markers_with_examples`.

Sergio Flesca, Giuseppe Manco, Elio Masciari, Eugenio Rende, and Andrea Tagarelli. 2004. Web wrapper induction: A brief survey. *AI Communications*, 17(2):57–61.

Dayne Freitag. 1998. Information extraction from HTML: Application of a general machine learning approach. In *AAAI/IAAI*, pages 517–523.

Alberto P García-Plaza, Víctor Fresno, Raquel Martínez Unanue, and Arkaitz Zubiaga. 2017. Using fuzzy logic to leverage HTML markup for web page representation. *IEEE Transactions on Fuzzy Systems*, 25(4):919–933.

Yufan Guo, Roi Reichart, and Anna Korhonen. 2013. Improved information structure analysis of scientific documents through discourse and lexical constraints. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 928–937.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: An update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Jonathan Hedley. 2017. jsoup (1.11.3). `https://jsoup.org/`.

Kenji Hirohata, Naoaki Okazaki, Sophia Ananiadou, and Mitsuru Ishizuka. 2008. Identifying sections in scientific abstracts using conditional random fields. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

Chenyi Liao, Kei Hiroi, Katsuhiko Kaji, and Nobuo Kawaguchi. 2015. An event data extraction method based on HTML structure analysis and machine learning. In *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, volume 3, pages 217–222. IEEE.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Norman Sadeh, Alessandro Acquisti, Travis D. Breaux, Lorrie Faith Cranor, Aleecia M. McDonald, Joel R. Reidenberg, Noah A. Smith, Fei Liu, N Cameron Russell, Florian Schaub, et al. 2013. The Usable Privacy Policy Project. Technical report, Technical report, Technical Report, CMU-ISR-13-119, Carnegie Mellon University.

Simone Teufel and Min-Yen Kan. 2011. Robust argumentative zoning for sensemaking in scholarly documents. In *Advanced Language Technologies for Digital Libraries*, pages 154–170. Springer.

Simone Teufel et al. 1999. *Argumentative zoning: Information extraction from scientific text*. Ph.D. thesis, University of Edinburgh.

Shomir Wilson. 2013. Toward automatic processing of english metalanguage. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 760–766.

Shomir Wilson and Jon Oberlander. 2014. Determiner-established deixis to communicative artifacts in pedagogical text. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 409–414.

Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N. Cameron Russell, et al. 2016. The creation and analysis of a website privacy policy corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1330–1340.

SwitchOut: an Efficient Data Augmentation Algorithm for Neural Machine Translation

Xinyi Wang^{*,1}, **Hieu Pham**^{*,1,2}, **Zihang Dai**^{1}, **Graham Neubig**^{1}

`{xinyiw1,hyhieu,dzihang,gneubig}@cs.cmu.edu`
^{1}Language Technology Institute, Carnegie Mellon University, Pittsburgh, PA 15213
^{2}Google Brain, Mountain View, CA 94043

Abstract

In this work, we examine methods for data augmentation for text-based tasks such as neural machine translation (NMT). We formulate the design of a data augmentation policy with desirable properties as an optimization problem, and derive a generic analytic solution. This solution not only subsumes some existing augmentation schemes, but also leads to an extremely simple data augmentation strategy for NMT: randomly replacing words in *both the source sentence and the target sentence* with other random words from their corresponding vocabularies. We name this method SwitchOut. Experiments on three translation datasets of different scales show that SwitchOut yields consistent improvements of about 0.5 BLEU, achieving better or comparable performances to strong alternatives such as word dropout (Sennrich et al., 2016a). Code to implement this method is included in the appendix.

1 Introduction and Related Work

Data augmentation algorithms generate extra data points from the empirically observed training set to train subsequent machine learning algorithms. While these extra data points may be of lower quality than those in the training set, their quantity and diversity have proven to benefit various learning algorithms (DeVries and Taylor, 2017; Amodei et al., 2016). In image processing, simple augmentation techniques such as flipping, cropping, or increasing and decreasing the contrast of the image are both widely utilized and highly effective (Huang et al., 2016; Zagoruyko and Komodakis, 2016).

However, it is nontrivial to find simple equivalences for NLP tasks like machine translation, because even slight modifications of sentences can result in significant changes in their semantics, or require corresponding changes in the translations in order to keep the data consistent. In fact, indiscriminate modifications of data in NMT can introduce noise that makes NMT systems brittle (Belinkov and Bisk, 2018).

Due to such difficulties, the literature in data augmentation for NMT is relatively scarce. To our knowledge, data augmentation techniques for NMT fall into two categories. The first category is based on back-translation (Sennrich et al., 2016b; Poncelas et al., 2018), which utilizes monolingual data to augment a parallel training corpus. While effective, back-translation is often vulnerable to errors in initial models, a common problem of self-training algorithms (Chapelle et al., 2009). The second category is based on word replacements. For instance, Fadaee et al. (2017) propose to replace words in the target sentences with rare words in the target vocabulary according to a language model, and then modify the aligned source words accordingly. While this method generates augmented data with relatively high quality, it requires several complicated preprocessing steps, and is only shown to be effective for low-resource datasets. Other generic word replacement methods include word dropout (Sennrich et al., 2016a; Gal and Ghahramani, 2016), which uniformly set some word embeddings to 0 at random, and Reward Augmented Maximum Likelihood (RAML; Norouzi et al. (2016)), whose implementation essentially replaces some words in the target sentences with other words from the target vocabulary.

In this paper, we derive an extremely simple and efficient data augmentation technique for NMT. **First,** we formulate the design of a data augmentation algorithm as an optimization problem, where we seek the data augmentation policy that maximizes an objective that encourages two desired properties: smoothness and diversity. This optimization problem has a tractable analytic solution,

---

*: Equal contributions.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 856–861
Brussels, Belgium, October 31 - November 4, 2018.

which describes a generic framework of which both word dropout and RAML are instances. **Second,** we interpret the aforementioned solution and propose a novel method: independently replacing words in *both the source sentence and the target sentence* by other words uniformly sampled from the source and the target vocabularies, respectively. Experiments show that this method, which we name SwitchOut, consistently improves over strong baselines on datasets of different scales, including the large-scale WMT 15 English-German dataset, and two medium-scale datasets: IWSLT 2016 German-English and IWSLT 2015 English-Vietnamese.

2 Method

2.1 Notations

We use uppercase letters, such as X, Y, etc., to denote random variables and lowercase letters such as x, y, etc., to denote the corresponding actual values. Additionally, since we will discuss a data augmentation algorithm, we will use a hat to denote augmented variables and their values, *e.g.* $\widehat{X}$, $\widehat{Y}$, $\widehat{x}$, $\widehat{y}$, etc. We will also use boldfaced characters, such as $\mathbf{p}$, $\mathbf{q}$, etc., to denote probability distributions.

2.2 Data Augmentation

We facilitate our discussion with a probabilistic framework that motivates data augmentation algorithms. With X, Y being the sequences of words in the source and target languages (*e.g.* in machine translation), the canonical MLE framework maximizes the objective

$$J_{\text{MLE}}(\theta) = \mathbb{E}_{x,y \sim \widehat{\mathbf{p}}(X,Y)} \left[\log \mathbf{p}_\theta(y|x)\right].$$

Here $\widehat{\mathbf{p}}(X,Y)$ is the empirical distribution over all training data pairs (x,y) and $\mathbf{p}_\theta(y|x)$ is a parameterized distribution that we aim to learn, *e.g.* a neural network. A potential weakness of MLE is the mismatch between $\widehat{\mathbf{p}}(X,Y)$ and the true data distribution $\mathbf{p}(X,Y)$. Specifically, $\widehat{\mathbf{p}}(X,Y)$ is usually a bootstrap distribution defined only on the observed training pairs, while $\mathbf{p}(X,Y)$ has a much larger support, *i.e.* the entire space of valid pairs. This issue can be dramatic when the empirical observations are insufficient to cover the data space.

In practice, data augmentation is often used to remedy this support discrepancy by supplying additional training pairs. Formally, let $\mathbf{q}(\widehat{X},\widehat{Y})$ be the augmented distribution defined on a larger support than the empirical distribution $\widehat{\mathbf{p}}(X,Y)$. Then, MLE training with data augmentation maximizes

$$J_{\text{AUG}}(\theta) = \mathbb{E}_{\widehat{x},\widehat{y} \sim \mathbf{q}(\widehat{X},\widehat{Y})} \left[\log \mathbf{p}_\theta(\widehat{y}|\widehat{x})\right].$$

In this work, we focus on a specific family of $\mathbf{q}$, which depends on the empirical observations by

$$\mathbf{q}(\widehat{X},\widehat{Y}) = \mathbb{E}_{x,y \sim \widehat{\mathbf{p}}(x,y)} \left[\mathbf{q}(\widehat{X},\widehat{Y}|x,y)\right].$$

This particular choice follows the intuition that an augmented pair $(\widehat{x},\widehat{y})$ that diverges too far from any observed data is more likely to be invalid and thus harmful for training. The reason will be more evident later.

2.3 Diverse and Smooth Augmentation

Certainly, not all $\mathbf{q}$ are equally good, and the more similar $\mathbf{q}$ is to $\mathbf{p}$, the more desirable $\mathbf{q}$ will be. Unfortunately, we only have access to limited observations captured by $\widehat{\mathbf{p}}$. Hence, in order to use $\mathbf{q}$ to bridge the gap between $\widehat{\mathbf{p}}$ and $\mathbf{p}$, it is necessary to utilize some assumptions about $\mathbf{p}$. Here, we exploit two highly generic assumptions, namely:

- **Diversity:** $\mathbf{p}(X,Y)$ has a wider support set, which includes samples that are more diverse than those in the empirical observation set.

- **Smoothness:** $\mathbf{p}(X,Y)$ is smooth, and similar (x,y) pairs will have similar probabilities.

To formalize both assumptions, let $s(\widehat{x},\widehat{y};x,y)$ be a similarity function that measures how similar an augmented pair $(\widehat{x},\widehat{y})$ is to an observed data pair (x,y). Then, an ideal augmentation policy $\mathbf{q}(\widehat{X},\widehat{Y}|x,y)$ should have two properties. First, based on the smoothness assumption, if an augmented pair $(\widehat{x},\widehat{y})$ is more similar to an empirical pair (x,y), it is more likely that $(\widehat{x},\widehat{y})$ is sampled under the true data distribution $\mathbf{p}(X,Y)$, and thus $\mathbf{q}(\widehat{X},\widehat{Y}|x,y)$ should assign a significant amount of probability mass to $(\widehat{x},\widehat{y})$. Second, to quantify the diversity assumption, we propose that the entropy $\mathbb{H}[\mathbf{q}(\widehat{X},\widehat{Y}|x,y)]$ should be large, so that the support of $\mathbf{q}(\widehat{X},\widehat{Y})$ is larger than the support of $\widehat{\mathbf{p}}$ and thus is closer to the support $\mathbf{p}(X,Y)$. Combining these assumptions implies that $\mathbf{q}(\widehat{X},\widehat{Y}|x,y)$ should maximize the objective

$$\begin{aligned} J(\mathbf{q};x,y) = &\ \mathbb{E}_{\widehat{x},\widehat{y} \sim \mathbf{q}(\widehat{X},\widehat{Y}|x,y)} \left[s(\widehat{x},\widehat{y};x,y)\right] \\ &+ \tau \mathbb{H}(\mathbf{q}(\widehat{X},\widehat{Y}|x,y)), \end{aligned} \tag{1}$$

where τ controls the strength of the diversity objective. The first term in (1) instantiates the smoothness assumption, which encourages $\mathbf{q}$ to draw samples that are similar to (x, y). Meanwhile, the second term in (1) encourages more diverse samples from $\mathbf{q}$. Together, the objective $J(\mathbf{q}; x, y)$ extends the information in the "pivotal" empirical sample (x, y) to a diverse set of similar cases. This echoes our particular parameterization of $\mathbf{q}$ in Section 2.2.

The objective $J(\mathbf{q}; x, y)$ in (1) is the canonical maximum entropy problem that one often encounters in deriving a max-ent model (Berger et al., 1996), which has the analytic solution:

$$\mathbf{q}^*(\widehat{x}, \widehat{y}|x, y) = \frac{\exp\left\{s(\widehat{x}, \widehat{y}; x, y)/\tau\right\}}{\sum_{\widehat{x}', \widehat{y}'} \exp\left\{s(\widehat{x}', \widehat{y}'; x, y)/\tau\right\}} \tag{2}$$

Note that (2) is a fairly generic solution which is agnostic to the choice of the similarity measure s. Obviously, not all similarity measures are equally good. Next, we will show that some existing algorithms can be seen as specific instantiations under our framework. Moreover, this leads us to propose a novel and effective data augmentation algorithm.

2.4 Existing and New Algorithms

Word Dropout. In the context of machine translation, Sennrich et al. (2016a) propose to randomly choose some words in the source and/or target sentence, and set their embeddings to 0 vectors. Intuitively, it regards every new data pair generated by this procedure as similar enough and then includes them in the augmented training set. Formally, word dropout can be seen as an instantiation of our framework with a particular similarity function $s(\hat{x}, \hat{y}; x, y)$ (see Appendix A.1).

RAML. From the perspective of reinforcement learning, Norouzi et al. (2016) propose to train the model distribution to match a target distribution proportional to an exponentiated reward. Despite the difference in motivation, it can be shown (*c.f.* Appendix A.2) that RAML can be viewed as an instantiation of our generic framework, where the similarity measure is $s(\widehat{x}, \widehat{y}; x, y) = r(\widehat{y}; y)$ if $\widehat{x} = x$ and $-\infty$ otherwise. Here, r is a task-specific reward function which measures the similarity between $\widehat{y}$ and y. Intuitively, this means that RAML only exploits the smoothness property on the target side while keeping the source side intact.

SwitchOut. After reviewing the two existing augmentation schemes, there are two immediate insights. Firstly, augmentation should not be restricted to only the source side or the target side. Secondly, being able to incorporate prior knowledge, such as the task-specific reward function r in RAML, can lead to a better similarity measure.

Motivated by these observations, we propose to perform augmentation in both source and target domains. For simplicity, we separately measure the similarity between the pair $(\widehat{x}, x)$ and the pair $(\widehat{y}, y)$ and then sum them together, *i.e.*

$$s(\widehat{x}, \widehat{y}; x, y)/\tau \approx r_x(\widehat{x}, x)/\tau_x + r_y(\widehat{y}, y)/\tau_y, \tag{3}$$

where r_x and r_y are domain specific similarity functions and τ_x, τ_y are hyper-parameters that absorb the temperature parameter τ. This allows us to factor $\mathbf{q}^*(\widehat{x}, \widehat{y}|x, y)$ into:

$$\begin{aligned} \mathbf{q}^*(\widehat{x}, \widehat{y}|x, y) = & \frac{\exp\left\{r_x(\widehat{x}, x)/\tau_x\right\}}{\sum_{\widehat{x}'} \exp\left\{r_x(\widehat{x}', x)/\tau_x\right\}} \\ & \times \frac{\exp\left\{r_y(\widehat{y}, y)/\tau_y\right\}}{\sum_{\widehat{y}'} \exp\left\{r_y(\widehat{y}', y)/\tau_y\right\}} \end{aligned} \tag{4}$$

In addition, notice that this factored formulation allows $\widehat{x}$ and $\widehat{y}$ to be sampled independently.

Sampling Procedure. To complete our method, we still need to define r_x and r_y, and then design a practical sampling scheme from each factor in (4). Though non-trivial, both problems have been (partially) encountered in RAML (Norouzi et al., 2016; Ma et al., 2017). For simplicity, we follow previous work to use the *negative* Hamming distance for both r_x and r_y. For a more parallelized implementation, we sample an augmented sentence $\widehat{s}$ from a true sentence s as follows:

1. Sample $\widehat{n} \in \{0, 1, ..., |s|\}$ by $\mathbf{p}(\widehat{n}) \propto e^{-\widehat{n}/\tau}$.
2. For each $i \in \{1, 2, ..., |s|\}$, with probability $\widehat{n}/|s|$, we can replace s_i by a uniform $\widehat{s_i} \neq s_i$.

This procedure guarantees that any two sentences $\widehat{s}_1$ and $\widehat{s}_2$ with the same Hamming distance to s have the same probability, but slightly changes the relative odds of sentences with different Hamming distances to s from the true distribution by negative Hamming distance, and thus is an approximation of the actual distribution. However, this efficient sampling procedure is much easier to implement while achieving good performance.

Algorithm 1 illustrates this sampling procedure, which can be applied independently and in parallel for each batch of source sentences and target sentences. Additionally, we open source our implementation in TensorFlow and in PyTorch (respectively in Appendix A.5 and A.6).

Algorithm 1: Sampling with SwitchOut.

```
Input   : $s$: a sentence represented by vocab integral ids,
          $\tau$: the temperature, $V$: the vocabulary
Output  : $\widehat{s}$: a sentence with words replaced
Function HammingDistanceSample($s$, $\tau$, $|V|$):
    Let $Z(\tau) \leftarrow \sum_{n=0}^{|s|} e^{-n/\tau}$ be the partition function.
    Let $p(n) \leftarrow e^{-n/\tau}/Z(\tau)$ for $n = 0, 1, ..., |s|$.
    Sample $\widehat{n} \sim p(n)$.
    In parallel, do:
        Sample $a_i \sim$ Bernoulli($\widehat{n}/|s|$).
        if $a_i = 1$ then
            $\widehat{s}_i \leftarrow$ Uniform($V \backslash \{s_i\}$).
        else
            $\widehat{s}_i \leftarrow s_i$.
        end
    return $\widehat{s}$
```

3 Experiments

Datasets. We benchmark SwitchOut on three translation tasks of different scales: 1) IWSLT 2015 English-Vietnamese (en-vi); 2) IWSLT 2016 German-English (de-en); and 3) WMT 2015 English-German (en-de). All translations are word-based. These tasks and pre-processing steps are standard, used in several previous works. Detailed statistics and pre-processing schemes are in Appendix A.3.

Models and Experimental Procedures. Our translation model, *i.e.* $\mathbf{p}_\theta(y|x)$, is a Transformer network (Vaswani et al., 2017). For each dataset, we first train a standard Transformer model without SwitchOut and tune the hyper-parameters on the dev set to achieve competitive results. (w.r.t. Luong and Manning (2015); Gu et al. (2018); Vaswani et al. (2017)). Then, fixing all hyper-parameters, and fixing $\tau_y = 0$, we tune the τ_x rate, which controls how far we are willing to let $\widehat{x}$ deviate from x. Our hyper-parameters are listed in Appendix A.4.

Baselines. While the Transformer network without SwitchOut is already a strong baseline, we also compare SwitchOut against two other baselines that further use existing varieties of data augmentation: 1) word dropout on the source side with the dropping probability of $\lambda_{\text{word}} = 0.1$; and 2) RAML on the target side, as in Section 2.4. Additionally, on the en-de task, we compare SwitchOut against back-translation (Sennrich et al., 2016b).

Method	**en-de**	**de-en**	**en-vi**
Transformer	21.73	29.81	27.97
+WordDropout	20.63	**29.97**	28.56
+SwitchOut	**22.78**†	29.94	**28.67**†
+RAML	22.83	30.66	28.88
+RAML +WordDropout	20.69	30.79	28.86
+RAML +SwitchOut	**23.13**†	**30.98**†	**29.09**

Table 1: Test BLEU scores of SwitchOut and other baselines (median of multiple runs). Results marked with † are statistically significant compared to the best result without SwitchOut. For example, for en-de results in the first column, +SwitchOut has significant gain over Transformer; +RAML +SwitchOut has significant gain over +RAML.

SwitchOut vs. Word Dropout and RAML. We report the BLEU scores of SwitchOut, word dropout, and RAML on the test sets of the tasks in Table 1. To account for variance, we run each experiment multiple times and report the median BLEU. Specifically, each experiment without SwitchOut is run for 4 times, while each experiment with SwitchOut is run for 9 times due to its inherently higher variance. We also conduct pairwise statistical significance tests using paired bootstrap (Clark et al., 2011), and record the results in Table 1. For 4 of the 6 settings, SwitchOut delivers significant improvements over the best baseline without SwitchOut. For the remaining two settings, the differences are not statistically significant. The gains in BLEU with SwitchOut over the best baseline on WMT 15 en-de are all significant ($p < 0.0002$). Notably, SwitchOut on the source demonstrates as large gains as these obtained by RAML on the target side, and SwitchOut delivers further improvements when combined with RAML.

SwitchOut vs. Back Translation. Traditionally, data-augmentation is viewed as a method to enlarge the training datasets (Krizhevsky et al., 2012; Szegedy et al., 2014). In the context of neural MT, Sennrich et al. (2016b) propose to use artificial data generated from a weak back-translation model, effectively utilizing monolingual data to enlarge the bilingual training datasets. In connection, we compare SwitchOut against back translation. We only compare SwitchOut against back translation on the en-de task, where the amount of bilingual training data is already sufficiently large[2]. The

---

[2] We add the extra monolingual data from `http://data.statmt.org/rsennrich/wmt16_backtranslations/en-de/`

Method	en-de
Transformer	21.73
+SwitchOut	22.78
+BT	21.82
+BT +RAML	21.53
+BT +SwitchOut	22.93
+BT +RAML +SwitchOut	**23.76**

Table 2: Test BLEU scores of back translation (BT) compared to and combined with SwitchOut (median of 4 runs).

BLEU scores with back-translation are reported in Table 2. These results provide two insights. First, the gain delivered by back translation is less significant than the gain delivered by SwitchOut. Second, SwitchOut and back translation are not mutually exclusive, as one can additionally apply SwitchOut on the additional data obtained from back translation to further improve BLEU scores.

Effects of τ_x and τ_y. We empirically study the effect of these temperature parameters. During the tuning process, we translate the dev set of the tasks and report the BLEU scores in Figure 1. We observe that when fixing τ_y, the best performance is always achieved with a non-zero τ_x.

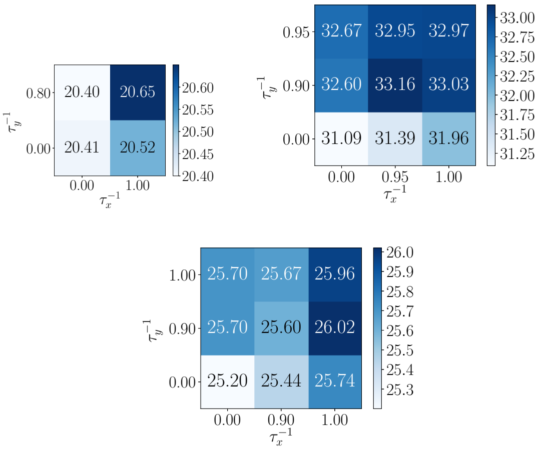

Figure 1: Dev BLEU scores with different τ_x and τ_y. *Top left*: WMT 15 en-de. *Top right*: IWSLT 16 de-en. *Bottom*: IWSLT 15 en-vi.

Where does SwitchOut Help the Most? Intuitively, because SwitchOut is expanding the support of the training distribution, we would expect that it would help the most on test sentences that are *far from those in the training set* and would thus benefit most from this expanded support. To test this hypothesis, for each test sentence we find its most similar training sample (*i.e.* nearest neighbor), then bucket the instances by the distance to their nearest neighbor and measure the gain in BLEU afforded by SwitchOut for each bucket. Specifically, we use (negative) word error rate (WER) as the similarity measure, and plot the bucket-by-bucket performance gain for each group in Figure 2. As we can see, SwitchOut improves increasingly more as the WER increases, indicating that SwitchOut is indeed helping on examples that are far from the sentences that the model sees during training. This is the desirable effect of data augmentation techniques.

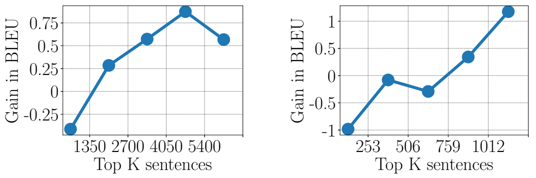

Figure 2: Gains in BLEU of RAML+SwitchOut over RAML. x-axis is ordered by the WER between a test sentence and its nearest neighbor in the training set. *Left*: IWSLT 16 de-en. *Right*: IWSLT 15 en-vi.

4 Conclusion

In this paper, we propose a method to design data augmentation algorithms by solving an optimization problem. These solutions subsume a few existing augmentation schemes and inspire a novel augmentation method, SwitchOut. SwitchOut delivers improvements over translation tasks at different scales. Additionally, SwitchOut is efficient and easy to implement, and thus has the potential for wide application.

Acknowledgements

We thank Quoc Le, Minh-Thang Luong, Qizhe Xie, and the anonymous EMNLP reviewers, for their suggestions to improve the paper.

This material is based upon work supported in part by the Defense Advanced Research Projects Agency Information Innovation Office (I2O) Low Resource Languages for Emergent Incidents (LORELEI) program under Contract No. HR0011-15-C0114. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, and more authors. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML*.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *ICLR*.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Jonathan Clark, Chris Dyer, Alon Lavie, and Noah Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *ACL*.

Terrance DeVries and Graham W. Taylor. 2017. Improved regularization of convolutional neural networks with cutout. *Arxiv, 1708.04552*.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *ACL*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *ICLR*.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2016. Densely connected convolutional networks. In *CVPR*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *IWLST*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Xuezhe Ma, Pengcheng Yin, Jingzhou Liu, Graham Neubig, and Eduard Hovy. 2017. Softmax q-distribution estimation for structured prediction: A theoretical interpretation for raml. *Arxiv, 1705.07136*.

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In *NIPS*.

Alberto Poncelas, Dimitar Shterionov, Andy Way, Gideon Maillette de Buy Wenniger, and Peyman Passban. 2018. Investigating backtranslation in neural machine translation. *Arxiv, 1804.06189*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16. In *WMT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *ACL*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going deeper with convolutions. In *CVPR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. In *BMVC*.

Improving Unsupervised Word-by-Word Translation with Language Model and Denoising Autoencoder

Yunsu Kim **Jiahui Geng** **Hermann Ney**
Human Language Technology and Pattern Recognition Group
RWTH Aachen University
Aachen, Germany
`{surname}@cs.rwth-aachen.de`

Abstract

Unsupervised learning of cross-lingual word embedding offers elegant matching of words across languages, but has fundamental limitations in translating sentences. In this paper, we propose simple yet effective methods to improve word-by-word translation of cross-lingual embeddings, using only monolingual corpora but without any back-translation. We integrate a language model for context-aware search, and use a novel denoising autoencoder to handle reordering. Our system surpasses state-of-the-art unsupervised neural translation systems without costly iterative training. We also analyze the effect of vocabulary size and denoising type on the translation performance, which provides better understanding of learning the cross-lingual word embedding and its usage in translation.

1 Introduction

Building a machine translation (MT) system requires lots of bilingual data. Neural MT models (Bahdanau et al., 2015), which become the current standard, are even more difficult to train without huge bilingual supervision (Koehn and Knowles, 2017). However, bilingual resources are still limited to some of the selected language pairs—mostly from or to English.

A workaround for zero-resource language pairs is translating via an intermediate (pivot) language. To do so, we need to collect parallel data and train MT models for source-to-pivot and pivot-to-target individually; it takes a double effort and the decoding is twice as slow.

Unsupervised learning is another alternative, where we can train an MT system with only monolingual corpora. Decipherment methods (Ravi and Knight, 2011; Nuhn et al., 2013) are the first work in this direction, but they often suffer from a huge latent hypothesis space (Kim et al., 2017).

Recent work by Artetxe et al. (2018) and Lample et al. (2018) train sequence-to-sequence MT models of both translation directions together in an unsupervised way. They do back-translation (Sennrich et al., 2016a) back and forth for every iteration or batch, which needs an immensely long time and careful tuning of hyperparameters for massive monolingual data.

Here we suggest rather simple methods to build an unsupervised MT system quickly, based on word translation using cross-lingual word embeddings. The contributions of this paper are:

- We formulate a straightforward way to combine a language model with cross-lingual word similarities, effectively considering context in lexical choices.
- We develop a postprocessing method for word-by-word translation outputs using a denoising autoencoder, handling local reordering and multi-aligned words.
- We analyze the effect of different artificial noises for the denoising model and propose a novel noise type.
- We verify that cross-lingual embedding on subword units performs poorly in translation.
- We empirically show that cross-lingual mapping can be learned using a small vocabulary without losing the translation performance.

The proposed models can be efficiently trained with off-the-shelf softwares with little or no changes in the implementation, using only monolingual data. The provided analyses help for better learning of cross-lingual word embeddings for translation purpose. Altogether, our unsupervised MT system outperforms the sequence-to-sequence neural models even without training signals from the opposite translation direction, i.e. via back-translation.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 862–868
Brussels, Belgium, October 31 - November 4, 2018.

2 Cross-lingual Word Embedding

As a basic step for unsupervised MT, we learn a word translation model from monolingual corpora of each language. In this work, we exploit cross-lingual word embedding for word-by-word translation, which is state-of-the-art in terms of type translation quality (Artetxe et al., 2017; Conneau et al., 2018).

Cross-lingual word embedding is a continuous representation of words whose vector space is shared across multiple languages. This enables distance calculation between word embeddings across languages, which is actually finding translation candidates.

We train cross-lingual word embedding in a fully unsupervised manner:

1. Learn monolingual source and target embeddings independently. For this, we run skip-gram algorithm augmented with character n-gram (Bojanowski et al., 2017).
2. Find a linear mapping from source embedding space to target embedding space by adversarial training (Conneau et al., 2018). We do not pre-train the discriminator with a seed dictionary, and consider only the top $V_{\text{cross-train}}$ words of each language as input to the discriminator.

Once we have the cross-lingual mapping, we can transform the embedding of a given source word and find a target word with the closest embedding, i.e. nearest neighbor search. Here, we apply cross-domain similarity local scaling (Conneau et al., 2018) to penalize the word similarities in dense areas of the embedding distribution.

We further refine the mapping obtained from Step 2 as follows (Artetxe et al., 2017):

3. Build a synthetic dictionary by finding mutual nearest neighbors for both translation directions in vocabularies of $V_{\text{cross-train}}$ words.
4. Run a Procrustes problem solver with the dictionary from Step 3 to re-train the mapping (Smith et al., 2017).
5. Repeat Step 3 and 4 for a fixed number of iterations to update the mapping further.

3 Sentence Translation

In translating sentences, cross-lingual word embedding has several drawbacks. We describe each of them and our corresponding solutions.

3.1 Context-aware Beam Search

The word translation using nearest neighbor search does not consider context around the current word. In many cases, the correct translation is not the nearest target word but other close words with morphological variations or synonyms, depending on the context.

The reasons are in two-fold: 1) Word embedding is trained to place semantically related words nearby, even though they have opposite meanings. 2) A hubness problem of high-dimensional embedding space hinders a correct search, where lots of different words happen to be close to each other (Radovanović et al., 2010).

In this paper, we integrate context information into word-by-word translation by combining a language model (LM) with cross-lingual word embedding. Let f be a source word in the current position and e a possible target word. Given a history h of target words before e, the score of e to be the translation of f would be:

$$L(e; f, h) = \lambda_{\text{emb}} \log q(f, e) + \lambda_{\text{LM}} \log p(e|h)$$

Here, $q(f, e)$ is a lexical score defined as:

$$q(f, e) = \frac{d(f, e) + 1}{2}$$

where $d(f, e) \in [-1, 1]$ is a cosine similarity between f and e. It is transformed to the range $[0, 1]$ to make it similar in scale with the LM probability. In our experiments, we found that this simple linear scaling is better than sigmoid or softmax functions in the final translation performance.

Accumulating the scores per position, we perform a beam search to allow only reasonable translation hypotheses.

3.2 Denoising

Even when we have correctly translated words for each position, the output is still far from an acceptable translation. We adopt sequence denoising autoencoder (Hill et al., 2016) to improve the translation output of Section 3.1. The main idea is to train a sequence-to-sequence neural network model that takes a noisy sentence as input and produces a (denoised) clean sentence as output, both of which are of the same (target) language. The model was originally proposed to learn sentence embeddings, but here we use it directly to actually remove noise in a sentence.

Training label sequences for the denoising network would be target monolingual sentences, but we do not have their noisy versions at hand. Given a clean target sentence, the noisy input should be ideally word-by-word translation of the corresponding source sentence. However, such bilingual sentence alignment is not available in our unsupervised setup.

Instead, we inject artificial noise into a clean sentence to simulate the noise of word-by-word translation. We design different noise types after the following aspects of word-by-word translation.

3.2.1 Insertion

Word-by-word translation always outputs a target word for every position. However, there are a plenty of cases that multiple source words should be translated to a single target word, or that some source words are rather not translated to any word to make a fluent output. For example, a German sentence "*Ich höre zu.*" would be translated to "*I'm listening to.*" by a word-by-word translator, but "*I'm listening.*" is more natural in English (Figure 1).

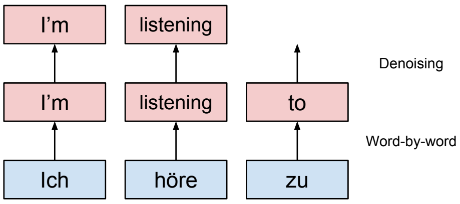

Figure 1: Example of denoising an insertion noise.

We pretend to have extra target words which might be translation of redundant source words, by inserting random target words to a clean sentence:

1. For each position i, sample a probability $p_i \sim \text{Uniform}(0, 1)$.
2. If $p_i < p_{\text{ins}}$, sample a word e from the most frequent V_{ins} target words and insert it before position i.

We limit the inserted words by V_{ins} because target insertion occurs mostly with common words, e.g. prepositions or articles, as the example above. We insert words only before—not after—a position, since an extra word after the ending word (usually a punctuation) is not probable.

3.2.2 Deletion

Similarly, word-by-word translation cannot handle the contrary case: when a source word should be translated into more than one target words, or a target word should be generated from no source words for fluency. For example, a German word "*im*" must be "*in the*" in English, but word translation generates only one of the two English words. Another example is shown in Figure 2.

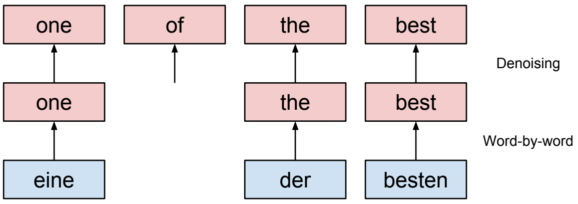

Figure 2: Example of denoising a deletion noise.

To simulate such situations, we drop some words randomly from a clean target sentence (Hill et al., 2016):

1. For each position i, sample a probability $p_i \sim \text{Uniform}(0, 1)$.
2. If $p_i < p_{\text{del}}$, drop the word in the position i.

3.2.3 Reordering

Also, translations generated word-by-word are not in an order of the target language. In our beam search, LM only assists in choosing the right word in context but does not modify the word order. A common reordering problem of German→English is illustrated in Figure 3.

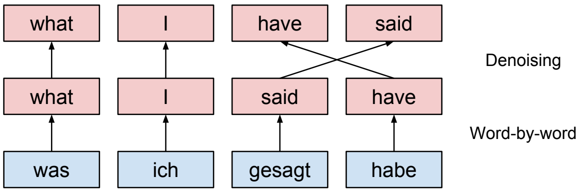

Figure 3: Example of denoising the reordering noise.

From a clean target sentence, we corrupt its word order by random permutations. We limit the maximum distance between an original position and its new position like Lample et al. (2018):

1. For each position i, sample an integer δ_i from $[0, d_{\text{per}}]$.
2. Add δ_i to index i and sort the incremented indices $i + \delta_i$ in an increasing order.
3. Rearrange the words to be in the new positions, to which their original indices have moved by Step 2.

System	de-en BLEU [%]	en-de BLEU [%]	fr-en BLEU [%]	en-fr BLEU [%]
Word-by-Word	11.1	6.7	10.6	7.8
+ LM	14.5	9.9	13.6	10.9
+ Denoising	**17.2**	**11.0**	**16.5**	13.9
(Lample et al., 2018)	13.3	9.6	14.3	15.1
(Artetxe et al., 2018)	-	-	15.6	15.1

Table 1: Translation results on German↔English `newstest2016` and French↔English `newstest2014`. Beam size is 10 and top 100 words are considered in the nearest neighbor search.

This is a generalized version of swapping two neighboring words (Hill et al., 2016). Reordering is highly dependent of each language, but we found that this noise is generally close to word-by-word translation outputs.

Insertion, deletion, and reordering noises were applied to each mini-batch with different random seeds, allowing the model to see various noisy versions of the same clean sentence over the epochs.

Note that the deletion and permutation noises are integrated in the neural MT training of Artetxe et al. (2018) and Lample et al. (2018) as additional training objectives. Whereas we optimize an independent model solely for denoising without architecture change. It allows us to easily train a larger network with a larger data. Insertion noise is of our original design, which we found to be the most effective (Section 4.1).

4 Experiments

We applied the proposed methods on WMT 2016 German↔English task and WMT 2014 French↔English task. For German/English, we trained word embeddings with 100M sentences sampled from News Crawl 2014-2017 monolingual corpora. For French, we used News Crawl 2007-2014 (around 42M sentences). The data was lowercased and filtered to have a maximum sentence length 100. German compound words were splitted beforehand. Numbers were replaced with category labels and recovered back after decoding by looking at the source sentence. Also, frequent casing was applied to the translation output.

fasttext (Bojanowski et al., 2017) was used to learn monolingual embeddings for only the words with minimum count 10. MUSE (Conneau et al., 2018) was used for cross-lingual mappings with $V_{\text{cross-train}}$ = 100k and 10 refinement iterations (Step 3-5 in Section 2). Other parameters follow the values in Conneau et al. (2018). With the same data, we trained 5-gram count-based LMs using KenLM (Heafield, 2011) with its default setting.

Denoising autoencoders were trained using Sockeye (Hieber et al., 2017) on News Crawl 2016 for German/English and News Crawl 2014 for French. We considered only top 50k frequent words for each language and mapped other words to `<unk>`. The unknowns in the denoised output were replaced with missing words from the noisy input by a simple line search.

We used 6-layer Transformer encoder/decoder (Vaswani et al., 2017) for denoisers, with embedding/hidden layer size 512, feedforward sublayer size 2048 and 8 attention heads.

As a validation set for the denoiser training, we used `newstest2015` (German ↔ English) or `newstest2013` (French ↔ English), where the input/output sides both have the same clean target sentences, encouraging a denoiser to keep at least clean part of word-by-word translations. Here, the noisy input showed a slight degradation of performance; the model seemed to overfit to specific noises in the small validation set.

Optimization of the denoising models was done with Adam (Kingma and Ba, 2015): initial learning rate 0.0001, checkpoint frequency 4000, no learning rate warmup, multiplying 0.7 to the learning rate when the perplexity on the validation set did not improve for 3 checkpoints. We stopped the training if it was not improved for 8 checkpoints.

Table 1 shows the results. LM improves word-by-word baselines consistently in all four tasks, giving at least +3% BLEU. When our denoising model is applied on top of it, we have additional gain around +3% BLEU. Note that our methods do not involve any decoding steps to generate pseudo-parallel training data, but still perform better than unsupervised MT systems that rely on repetitive back-translations (Artetxe et al., 2018; Lample et al., 2018) by up to +3.9% BLEU. The total training time of our method is only 1-2 days with a single GPU.

4.1 Ablation Study: Denoising

d_{per}	p_{del}	V_{ins}	BLEU [%]
2			14.7
3			**14.9**
5			14.9
3	0.1		**15.7**
	0.3		15.1
3	0.1	10	16.8
		50	**17.2**
		500	16.8
		5000	16.5

Table 2: Translation results with different values of denoising parameters for German→English.

To examine the effect of each noise type in denoising autoencoder, we tuned each parameter of the noise and combined them incrementally (Table 2). Firstly, for permutations, a significant improvement is achieved from $d_{\text{per}} = 3$, since a local reordering usually involves a sequence of 3 to 4 words. With $d_{\text{per}} > 5$, it shuffles too many consecutive words together, yielding no further improvement. This noise cannot handle long-range reordering, which is usually a swap of words that are far from each other, keeping the words in the middle as they are.

Secondly, we applied the deletion noise with different values of p_{del}. 0.1 gives +0.8% BLEU, but we immediately see a degradation with a larger value; it is hard to observe one-to-many translations more than once in each sentence pair.

Finally, we optimized V_{ins} for the insertion noise, fixing $p_{\text{ins}} = 0.1$. Increasing V_{ins} is generally not beneficial, since it provides too much variations in the inserted word; it might not be related to its neighboring words. Overall, we observe the best result (+1.5% BLEU) with $V_{\text{ins}} = 50$.

4.2 Ablation Study: Vocabulary

We also examined how the translation performance varies with different vocabularies of cross-lingual word embedding in Table 3. The first three rows show that BPE embeddings performs worse than word embeddings, especially with smaller vocabulary size. For small BPE tokens (1-3 characters), the context they meet during the embedding training is much more various than a complete word, and a direct translation of such small token to a BPE token of another language would be very ambiguous.

Vocabulary		BLEU [%]
	Merges	
BPE	20k	10.4
	50k	12.5
	100k	13.0
	$V_{\text{cross-train}}$	
Word	20k	14.4
	50k	14.4
	100k	**14.5**
	200k	14.4

Table 3: Translation results with different vocabularies for German→English.

For word level embeddings, we compared different vocabulary sizes used for training the cross-lingual mapping (the second step in Section 2). Surprisingly, cross-lingual word embedding learned only on top 20k words is comparable to that of 200k words in the translation quality. We also increased the search vocabulary to more than 200k but the performance only degrades. This means that word-by-word translation with cross-lingual embedding depends highly on the frequent word mappings, and learning the mapping between rare words does not have a positive effect.

5 Conclusion

In this paper, we proposed a simple pipeline to greatly improve sentence translation based on cross-lingual word embedding. We achieved context-aware lexical choices using beam search with LM, and solved insertion/deletion/reordering problems using denoising autoencoder. Our novel insertion noise shows a promising performance even combined with other noise types. Our methods do not need back-translation steps but still outperforms costly unsupervised neural MT systems. In addition, we proved that for general translation purpose, an effective cross-lingual mapping can be learned using only a small set of frequent words, not on subword units.

Acknowledgments

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement No. 694537 (SEQCLAS). The GPU computing cluster was partially funded by Deutsche Forschungsgemeinschaft (DFG) under grant INST 222/1168-1 FUGG. The work reflects only the authors' views and neither ERC nor DFG is responsible for any use that may be made of the information it contains.

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 937–947, Valencia, Spain.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 451–462, Vancouver, Canada.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *6th International Conference on Learning Representations (ICLR 2018)*, Vancouver, Canada.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, CA, USA.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herve Jegou. 2018. Word translation without parallel data. In *6th International Conference on Learning Representations (ICLR 2018)*, Vancouver, Canada.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies (ACL-HLT 2008)*, pages 771–779, Columbus, OH, USA.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *EMNLP 6th Workshop on Statistical Machine Translation (WMT 2011)*, pages 187–197, Edinburgh, Scotland.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv Preprint*. 1712.05690.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 1367–1377, San Diego, CA, USA.

Yunsu Kim, Julian Schamper, and Hermann Ney. 2017. Unsupervised training for large vocabulary translation using sparse lexicon and word classes. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 650–656, Valencia, Spain.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, CA, USA.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *ACL 2017 1st Workshop on Neural Machine Translation (NMT 2017)*, pages 28–39, Vancouver, Canada.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT 2003)*, pages 48–54, Edmonton, Canada.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *6th International Conference on Learning Representations (ICLR 2018)*, Vancouver, Canada.

Malte Nuhn, Julian Schamper, and Hermann Ney. 2013. Beam search for solving substitution ciphers. In *51th Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1569–1576, Sofia, Bulgaria.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. On the existence of obstinate results in vector space models. In *33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2010)*, pages 186–193, Geneva, Switzerland.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pages 12–21, Portland, OR, USA.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *54th Annual Meeting of the Association of Computational Linguistics (ACL 2016)*, pages 86–96, Berlin, Germany.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *54th Annual Meeting of the Association of Computational Linguistics (ACL 2016)*, pages 1715–1725, Berlin, Germany.

Samuel L. Smith, David H.P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *5rd International Conference on Learning Representations (ICLR 2017)*, Toulon, France.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008, Long Beach, CA, USA.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1959–1970, Vancouver, Canada.

Decipherment of Substitution Ciphers with Neural Language Models

Nishant Kambhatla, Anahita Mansouri Bigvand, Anoop Sarkar
School of Computing Science
Simon Fraser University
Burnaby, BC , Canada
`{nkambhat,amansour,anoop}@sfu.ca`

Abstract

Decipherment of homophonic substitution ciphers using language models (LMs) is a well-studied task in NLP. Previous work in this topic scores short local spans of possible plaintext decipherments using n-gram LMs. The most widely used technique is the use of beam search with n-gram LMs proposed by Nuhn et al. (2013). We propose a beam search algorithm that scores the entire candidate plaintext at each step of the decipherment using a neural LM. We augment beam search with a novel rest cost estimation that exploits the prediction power of a neural LM. We compare against the state of the art n-gram based methods on many different decipherment tasks. On challenging ciphers such as the Beale cipher we provide significantly better error rates with much smaller beam sizes.

1 Introduction

Breaking substitution ciphers recovers the *plaintext* from a *ciphertext* that uses a 1:1 or homophonic cipher key. Previous work using pretrained language models (LMs) for decipherment use n-gram LMs (Ravi and Knight, 2011; Nuhn et al., 2013). Some methods use the Expectation-Maximization (EM) algorithm (Knight et al., 2006) while most state-of-the-art approaches for decipherment of 1:1 and homophonic substitution ciphers use beam search and rely on the clever use of n-gram LMs (Nuhn et al., 2014; Hauer et al., 2014). Neural LMs globally score the entire candidate plaintext sequence (Mikolov et al., 2010). However, using a neural LM for decipherment is not trivial because scoring the entire candidate partially deciphered plaintext is computationally challenging. We solve both of these problems in this paper and provide an improved beam search based decipherment algorithm for homophonic ciphers that exploits pre-trained neural LMs for the first time.

2 Decipherment Model

We use the notation from Nuhn et al. (2013). Ciphertext $f_1^N = f_1..f_i..f_N$ and plaintext $e_1^N = e_1..e_i..e_N$ consist of vocabularies $f_i \in V_f$ and $e_i \in V_e$ respectively. The beginning tokens in the ciphertext (f_0) and plaintext (e_0) are set to "\$" denoting the beginning of a sentence. The substitutions are represented by a function $\phi : V_f \rightarrow V_e$ such that 1:1 substitutions are *bijective* while homophonic substitutions are *general*. A cipher function ϕ which does not have every $\phi(f)$ fixed is called a partial cipher function (Corlett and Penn, 2010). The number of fs that are fixed in ϕ is given by its cardinality. ϕ' is called an extension of ϕ, if f is fixed in ϕ' such that $\delta(\phi'(f), \phi(f))$ yields true $\forall f \in V_f$ which are already fixed in ϕ where δ is Kronecker delta. **Decipherment** is then the task of finding the ϕ for which the probability of the deciphered text is maximized.

$$\hat{\phi} = \arg\max_{\phi} p(\phi(f_1)...\phi(f_N)) \tag{1}$$

where $p(.)$ is the language model (**LM**). Finding this argmax is solved using a beam search algorithm (Nuhn et al., 2013) which incrementally finds the most likely substitutions using the language model scores as the ranking.

2.1 Neural Language Model

The advantage of a neural LM is that it can be used to score the entire candidate plaintext for a hypothesized partial decipherment. In this work, we use a state of the art byte (character) level neural LM using a multiplicative LSTM (Radford et al., 2017).

Consider a sequence $S = w_1, w_2, w_3, ..., w_N$. The LM score of S is $\texttt{SCORE}(S)$:

$$\begin{aligned} P(S) &= P(w_1, w_2, w_3, ..., w_N) \\ P(S) &= \prod_{i=1}^{N} P(w_i \mid w_1, w_2, ..., w_{i-1})) \\ \texttt{SCORE}(S) &= -\sum_{i=1}^{N} log(P(w_i \mid w_{<i})) \end{aligned} \tag{2}$$

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 869–874
Brussels, Belgium, October 31 - November 4, 2018.

2.2 Beam Search

Algorithm 1 is the beam search algorithm (Nuhn et al., 2013, 2014) for solving substitution ciphers. It monitors all partial hypotheses in lists H_s and H_t based on their quality. As the search progresses, the partial hypotheses are extended, scored with `SCORE` and appended to H_t. `EXT_LIMITS` determines which extensions should be allowed and `EXT_ORDER` picks the next cipher symbol for extension. The search continues after pruning: $H_s \leftarrow \texttt{HISTOGRAM\_PRUNE}(H_t)$. We augment this algorithm by updating the `SCORE` function with a neural LM.

Algorithm 1 Beam Search for Decipherment

```
function (BEAM_SEARCH (EXT_ORDER, EXT_LIMITS))
    initialize sets $H_s$, $H_t$
    CARDINALITY = 0
    $H_s$.ADD(($\emptyset$,0))
    while CARDINALITY < $|V_f|$ do
        $f$ = EXT_ORDER[CARDINALITY]
        for all $\phi \in H_s$ do
            for all $e \in V_e$ do
                $\phi$' := $\phi \cup \{(e, f)\}$
                if EXT_LIMITS($\phi$') then
                    $H_t$.ADD($\phi$',SCORE($\phi$'))
        HISTOGRAM_PRUNE($H_t$)
        CARDINALITY = CARDINALITY + 1
        $H_s = H_t$
        $H_t$.CLEAR()
    return WINNER($H_s$)
```

3 Score Estimation (`SCORE`)

Score estimation evaluates the quality of the partial hypotheses ϕ. Using the example from Nuhn et al. (2014), consider the vocabularies $V_e = \{a, b, c, d\}$ and $V_f = \{A, B, C, D\}$, extension order (B, A, C, D), and ciphertext `$ ABDDCABCDADCABDC $`. Let $\phi = \{(a, A), (b, B))\}$ be the partial hypothesis. Then $\texttt{SCORE}(\phi)$ scores this hypothesized partial decipherment (only A and B are converted to plaintext) using a pre-trained language model in the hypothesized plaintext language.

3.1 Baseline

The initial rest cost estimator introduced by Nuhn et al. nuhnbeam computes the score of hypotheses only based on partially deciphered text that builds a shard of n adjacent solved symbols. As a heuristic, n-grams which still consist of unsolved cipher-symbols are assigned a trivial estimate of probability 1. An improved version of rest cost estimation (Nuhn et al., 2014) consults lower order n-grams to score each position.

3.2 Global Rest Cost Estimation

The baseline scoring method greatly relies on local context, *i.e.* the estimation is strictly based on partial character sequences. Since this depends solely on the n-gram LM, the true conditional probability under Markov assumption is not modeled and, therefore, context dependency beyond the window of $(n - 1)$ is ignored. Thus, attempting to utilize a higher amount of context can lower the probability of some tokens resulting in poor scores.

We address this issue with a new improved version of the rest cost estimator by supplementing the partial decipherment $\phi(f_1^N)$ with predicted plaintext text symbols using our neural language model (NLM). Applying $\phi = \{(a, A), (b, B))\}$ to the ciphertext above, we get the following partial hypothesis:

$$\phi(f_1^N) = \texttt{\$a}_1\texttt{b}_2\texttt{...a}_6\texttt{b}_7\texttt{..a}_{10}\texttt{..a}_{13}\texttt{b}_{14}\texttt{..\$}$$

We introduce a scoring function that is able to score the entire plaintext including the missing plaintext symbols. First, we sample[1] the plaintext symbols from the NLM at all locations depending on the deciphered tokens from the partial hypothesis ϕ such that these tokens maintain their respective positions in the sequence, and at the same time are sampled from the neural LM to fit (probabilistically) in this context. Next, we determine the probability of the entire sequence including the scores of sampled plaintext as our rest cost estimate.

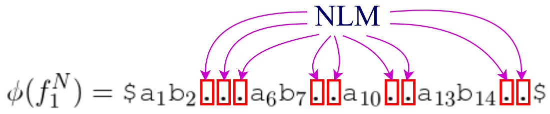

In our running example, this would yield a score estimation of the partial decipherment, $\phi(f_1^N)$:

$$\phi(f_1^N) = \$\, \mathrm{a}_1\mathrm{b}_2\mathrm{d}_3\mathrm{c}_4\mathrm{c}_5\mathrm{a}_6\mathrm{b}_7\mathrm{c}_8\mathrm{d}_9\mathrm{a}_{10}\mathrm{d}_{11}\mathrm{d}_{12}\mathrm{a}_{13}\mathrm{b}_{14}\mathrm{d}_{15}\mathrm{c}_{16}\ \$$$

Thus, the neural LM is used to predict the score of the full sequence. This method of global scoring evaluates each candidate partial decipherment by scoring the entire message, augmented by the sam-

---

[1] The char-level sampling is done incrementally from left to right to generate a sequence that contains the deciphered tokens from ϕ at the exact locations they occur in the above $\phi(f_1^N)$. If the LM prediction contradicts the hypothesized decipherment we stop sampling and start from the next character.

Cipher	Length	Unique Symbols	Obs/ symbol
Zodiac-408	408	54	7.55
Beale Pt. 2	763	180	4.23

Table 1: Homophonic ciphers used in our experiments.

pled plaintext symbols from the NLM. Since more terms participate in the rest cost estimation with global context, we use the plaintext LM to provide us with a better rest cost in the beam search.

3.3 Frequency Matching Heuristic

Alignment by frequency similarity (Yarowsky and Wicentowski, 2000) assumes that two forms belong to the same lemma when their relative frequency fits the expected distribution. We use this heuristic to augment the score estimation (`SCORE`):

$$\texttt{FMH}(\phi') = \left| log \left(\frac{\nu(f)}{\nu(e)} \right) \right| \qquad f \in V_f, \quad e \in V_e \tag{3}$$

$\nu(f)$ is the percentage relative frequency of the ciphertext symbol f, while $\nu(e)$ is the percentage relative frequency of the plaintext token e in the plaintext language model. The closer this value to 0, the more likely it is that f is mapped to e.

Thus given a ϕ with the $\texttt{SCORE}(\phi)$, the extension ϕ' (Algo. 1) is scored as:

$$\texttt{SCORE}(\phi') = \texttt{SCORE}(\phi) + \texttt{NEW}(\phi') - \texttt{FMH}(\phi') \tag{4}$$

where `NEW` is the score for symbols that have been newly fixed in ϕ' while extending ϕ to ϕ'. Our experimental evaluations show that the global rest cost estimator and the frequency matching heuristic contribute positively towards the accuracy of different ciphertexts.

4 Experimental Evaluation

We carry out 2 sets of experiments: one on letter based 1:1, and another on homophonic substitution ciphers. We report Symbol Error Rate (SER) which is the fraction of characters in the deciphered text that are incorrect.

The character NLM uses a single layer multiplicative LSTM (mLSTM) (Radford et al., 2017) with 4096 units. The model was trained for a single epoch on mini-batches of 128 subsequences of length 256 for a total of 1 million weight updates. States were initialized to zero at the beginning of each data shard and persisted across updates to simulate full-backprop and allow for the forward propagation of information outside of a given subsequence. In all the experiments we use a character NLM trained on English Gigaword corpus augmented with a short corpus of plaintext letters of about 2000 words authored by the Zodiac killer[2].

4.1 1:1 Substitution Ciphers

In this experiment we use a synthetic 1:1 letter substitution cipher dataset following Ravi and Knight (2008), Nuhn et al. (2013) and Hauer et al. (2014). The text is from English Wikipedia articles about history[3], preprocessed by stripping the text of all images, tables, then lower-casing all characters, and removing all non-alphabetic and non-space characters. We create 50 cryptograms for each length 16, 32, 64, 128 and 256 using a random Caesar-cipher 1:1 substitution.

Length	Beam	SER(%) 1	SER(%) 2
64	100	4.14	4.14
	1,000	1.09	1.04
	10,000	0.08	0.12
	100,000	0.07	0.07
128	100	7.31	7.29
	1,000	1.68	1.55
	10,000	0.15	0.09
	100,000	0.01	0.02

Table 2: Symbol Error Rates (SER) based on Neural Language Model and beam size (Beam) for solving 1:1 substitution ciphers of lengths 64 and 128, respectively. SER 1 shows beam search with global scoring, and SER 2 shows beam search with global scoring with frequency matching heuristic.

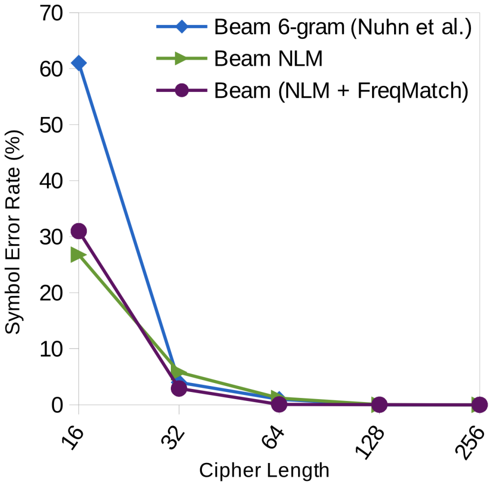

Figure 1: Symbol error rates for decipherment of 1:1 substitution ciphers of different lengths. The beam size is 100k. Beam 6-gram model uses the beam search from Nunh et al. (2013).

---

[2] https://en.wikisource.org/wiki/Zodiac_Killer_letters

[3] http://en.wikipedia.org/wiki/History

I	H	A	V	E	D	E	P	O	S	I	T	E	D	I	N	T	H	E	C	O	U	N	T	Y	O
115	73	24	807	37	52	49	17	31	62	647	22	7	15	140	47	29	107	79	84	56	239	10	26	811	5
F	B	E	D	F	O	R	D	A	B	O	U	T	F	O	U	R	M	I	L	E	S	F	R	O	M
196	308	85	52	160	136	59	211	36	9	46	316	554	122	106	95	53	58	2	42	7	35	122	53	31	82
B	U	F	O	R	D	S	I	N	A	N	E	X	C	A	V	A	T	I	O	N	O	R	V	A	U
77	250	196	56	96	118	71	140	287	28	353	37	1005	65	147	807	24	3	8	12	47	43	59	807	45	316
L	T	S	I	X	F	E	E	T	B	E	L	O	W	T	H	E	S	U	R	F	A	C	E	O	F
101	41	78	154	1005	122	138	191	16	77	49	102	57	72	34	73	85	35	371	59	196	81	92	191	106	273
T	H	E	G	R	O	U	N	D	T	H	E	F	O	L	L	O	W	I	N	G	A	R	T	I	C
60	394	620	270	220	106	388	287	63	3	6	191	122	43	234	400	106	290	314	47	48	81	96	26	115	92
L	E	S	B	E	L	O	N	G	I	N	G	J	O	I	N	T	L	Y	T	O	T	H	E	P	A
158	191	110	77	85	197	46	10	113	140	353	48	120	106	2	607	61	420	811	29	125	14	20	37	105	28

Figure 2: First few lines from part two of the Beale cipher. The letters have been capitalized.

Fig 1 plots the results of our method for cipher lengths of 16, 32, 64, 128 and 256 alongside Beam 6-gram (the best performing model) model (Nuhn et al., 2013)

4.2 An Easy Cipher: Zodiac-408

Zodiac-408, a homophonic cipher, is commonly used to evaluate decipherment algorithms.

Beam	SER (%) 1	SER (%) 2
10k	3.92	3.18
100k	2.40	1.91
1M	1.47	**1.22**

Table 3: Symbol Error Rates (SER) based on Neural Language Model and beam size (Beam) for deciphering Zodiac-408, respectively. SER 1 shows beam search with global scoring, and SER 2 shows beam search with global scoring with the frequency matching heuristic.

Our neural LM model with global rest cost estimation and frequency matching heuristic with a beam size of 1M has SER of 1.2% compared to the beam search algorithm (Nuhn et al., 2013) with beam size of 10M with a 6-gram LM which gives an SER of 2%. The improved beam search (Nuhn et al., 2014) with an 8-gram LM, however, gets 52 out of 54 mappings correct on the Zodiac-408 cipher.

4.3 A Hard Cipher: Beale Pt 2

Part 2 of the Beale Cipher is a more challenging homophonic cipher because of a much larger search space of solutions. Nunh et al. (2014) were the first to automatically decipher this Beale Cipher.

With an error of 5% with beam size of 1M vs 5.4% with 8-gram LM and a pruning size of 10M, our system outperforms the state of the art (Nuhn et al., 2014) on this task.

Figure 3: First 119 characters from deciphered Zodiac-408 text. The letters have been capitalized.

Beam	SER (%) 1	SER (%) 2
10k	41.67	48.33
100k	7.20	10.09
1M	**4.98**	5.50

Table 4: Symbol Error Rates (SER) based on Neural Language Model and beam size (Beam) for deciphering Part 2 of the Beale Cipher. SER 1 shows beam search with global scoring, and SER 2 shows beam search with global scoring with the frequency matching heuristic.

5 Related Work

Automatic decipherment for substitution ciphers started with dictionary attacks (Hart, 1994; Jakobsen, 1995; Olson, 2007). Ravi and Knight (2008) frame the decipherment problem as an integer linear programming (ILP) problem. Knight et al. (2006) use an HMM-based EM algorithm for solving a variety of decipherment problems. Ravi and Knight (2011) extend the HMM-based EM approach with a Bayesian approach, and report the first automatic decipherment of the Zodiac-408 cipher.

Berg-Kirkpatrick and Klein (2013) show that a large number of random restarts can help the EM approach.Corlett and Penn (2010) presented an efficient A* search algorithm to solve letter substitution ciphers. Nuhn et al. (2013) produce better results in faster time compared to ILP and EM-based decipherment methods by employing a higher order language model and an iterative beam search algorithm. Nuhn et al. (2014) present various improvements to the beam search algorithm in Nuhn et al. (2013) including improved rest cost estimation and an optimized strategy for ordering decipherment of the cipher symbols. Hauer et al. (2014) propose a novel approach for solving mono-alphabetic substitution ciphers which combines character-level and word-level language model. They formulate decipherment as a tree search problem, and use Monte Carlo Tree Search (MCTS) as an alternative to beam search. Their approach is the best for short ciphers.

Greydanus (2017) frames the decryption process as a sequence-to-sequence translation task and uses a deep LSTM-based model to learn the decryption algorithms for three polyalphabetic ciphers including the Enigma cipher. However, this approach needs supervision compared to our approach which uses a pre-trained neural LM. Gomez et al. (2018) (CipherGAN) use a generative adversarial network to learn the mapping between the learned letter embedding distributions in the ciphertext and plaintext. They apply this approach to shift ciphers (including Vigenère ciphers). Their approach cannot be extended to homophonic ciphers and full message neural LMs as in our work.

6 Conclusion

This paper presents, to our knowledge, the first application of large pre-trained neural LMs to the decipherment problem. We modify the beam search algorithm for decipherment from Nuhn et al. (2013; 2014) and extend it to use global scoring of the plaintext message using neural LMs. To enable full plaintext scoring we use the neural LM to sample plaintext characters which reduces the beam size required. For challenging ciphers such as Beale Pt 2 we obtain lower error rates with smaller beam sizes when compared to the state of the art in decipherment for such ciphers.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful remarks. The research was also partially supported by the Natural Sciences and Engineering Research Council of Canada grants NSERC RGPIN-2018-06437 and RGPAS-2018-522574 and a Department of National Defence (DND) and NSERC grant DGDND-2018-00025 to the third author.

References

Taylor Berg-Kirkpatrick and Dan Klein. 2013. Decipherment with a million random restarts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 874–878.

Eric Corlett and Gerald Penn. 2010. An exact A* method for deciphering letter-substitution ciphers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1040–1047. Association for Computational Linguistics.

Aidan N. Gomez, Scng Huang, Ivan Zhang, Bryan M. Li, Muhammad Osama, and ukasz Kaiser. 2018. Unsupervised cipher cracking using discrete gans. *arXiv preprint arXiv:1801.04883*.

Sam Greydanus. 2017. Learning the enigma with recurrent neural networks. *arXiv preprint arXiv:1708.07576*.

George W Hart. 1994. To decode short cryptograms. *Communications of the ACM*, 37(9):102–108.

Bradley Hauer, Ryan Hayward, and Grzegorz Kondrak. 2014. Solving substitution ciphers with combined language models. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2314–2325.

Thomas Jakobsen. 1995. A fast method for cryptanalysis of substitution ciphers. *Cryptologia*, 19(3):265–274.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 499–506. Association for Computational Linguistics.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Malte Nuhn, Julian Schamper, and Hermann Ney. 2013. Beam search for solving substitution ciphers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1568–1576.

Malte Nuhn, Julian Schamper, and Hermann Ney. 2014. Improved decipherment of homophonic ciphers. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1764–1768.

Edwin Olson. 2007. Robust dictionary attack of short simple substitution ciphers. *Cryptologia*, 31(4):332–342.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Sujith Ravi and Kevin Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *proceedings of the conference on Empirical Methods in Natural Language Processing*, pages 812–819. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. 2011. Bayesian inference for zodiac and other homophonic ciphers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 239–247. Association for Computational Linguistics.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216. Association for Computational Linguistics.

Rapid Adaptation of Neural Machine Translation to New Languages

Graham Neubig, Junjie Hu
Language Technologies Institute, Carnegie Mellon University
`{gneubig,junjieh}@cs.cmu.edu`

Abstract

This paper examines the problem of adapting neural machine translation systems to new, low-resourced languages (LRLs) as *effectively* and *rapidly* as possible. We propose methods based on starting with massively multilingual "seed models", which can be trained ahead-of-time, and then continuing training on data related to the LRL. We contrast a number of strategies, leading to a novel, simple, yet effective method of "similar-language regularization", where we jointly train on both a LRL of interest and a similar high-resourced language to prevent over-fitting to small LRL data. Experiments demonstrate that massively multilingual models, even without any explicit adaptation, are surprisingly effective, achieving BLEU scores of up to 15.5 with *no* data from the LRL, and that the proposed similar-language regularization method improves over other adaptation methods by 1.7 BLEU points average over 4 LRL settings.[1]

1 Introduction

When disaster strikes, news and social media are invaluable sources of information, allowing humanitarian organizations to rapidly mitigate crisis situations and save lives (Vieweg et al., 2010; Neubig et al., 2011; Starbird et al., 2012). However, language barriers looms large over these efforts, especially when disasters occur in parts of the world that use less common languages. In these cases, machine translation (MT) technology can be a valuable tool, with one widely-heralded success story being the deployment of Haitian Creole-to-English translation systems during the earthquakes in Haiti (Lewis, 2010; Munro, 2010).

However, data-driven MT systems, particularly neural machine translation (NMT; Kalchbrenner and Blunsom (2013); Bahdanau et al. (2015)), require large amounts of training data, and creating high-quality systems in low-resource languages (LRLs) is a difficult challenge where research efforts have just begun (Gu et al., 2018). Another hurdle, which to our knowledge has not been covered in previous research, is the *time* it takes to create such a system. In a crisis situation, time is of the essence, and systems that require days or weeks of training will not be desirable or even feasible.

In this paper we focus on the question: *how can we create MT systems for new language pairs as accurately as possible, and as quickly as possible?* To examine this question we propose NMT methods at the intersection of cross-lingual transfer learning (Zoph et al., 2016) and multilingual training (Johnson et al., 2016), two paradigms that, to our knowledge, have not been used together in previous work. Our methods, laid out in §2 follow the process of training a seed model on a large number of languages, then fine-tuning the model to improve its performance on the language of interest. We propose a novel method of *similar-language regularization* (SLR) where training data from a second similar languages is used to help prevent over-fitting to the small LRL dataset.

In the experiments in §3, we attempt to answer two questions: (1) Which method of creating multilingual systems and adapting them to an LRL is the most effective way to increase accuracy? (2) How can we create the strongest system possible with a bare minimum of training time? The results are sometimes surprising – we first find that a single monolithic model trained on 57 languages can achieve BLEU scores as high as 15.5 with *no* training data in the new source language whatsoever. In addition, the proposed method starting with a universal model then fine-tuning with the SLR proves most effective, achieving gains of 1.7

---

[1] Code to reproduce experiments at `https://github.com/neubig/rapid-adaptation`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 875–880
Brussels, Belgium, October 31 - November 4, 2018.

BLEU points averaged over several language pairs compared to previous methods adapting to only the LRL.

2 Training Paradigms

In this paper, we consider the setting where we have a *source LRL* of interest, and we want to translate into English.[2] All of our adaptation methods are based on first training on larger data including other languages, then fine-tuning the model to be specifically tailored to the LRL. We first discuss a few multilingual training paradigms from previous literature (§2.1), then discuss our proposed adaptation methods (§2.2).

2.1 Multilingual Modeling Methods

We use three varieties of multilingual training:

Single-source modeling ("Sing.") is the first method, using only parallel data between the LRL of interest and English. This method is straightforward and the resulting model will be most highly tailored to the final test language pair, but the method also has the obvious disadvantage that training data is very sparse.

Bi-source modeling ("Bi") trains an MT system with two source languages: one LRL that we would like to translate from, and a second highly related high-resource language (HRL): the *helper* source language.[3] This method is inspired by Johnson et al. (2016), who examine multilingual translation models to/from English and two highly related languages such as Spanish/Portuguese or Japanese/Korean. The advantage of this method is that it allows the LRL to learn from a highly similar helper, potentially increasing accuracy.

All-source modeling ("All") trains not only on a couple source languages, but instead creates a universal model on all of the languages that we have at our disposal. In our experiments (§3.1) this entails training systems on 58 source languages, to our knowledge the largest reported in NMT experiments.[4] This paradigm allows us to train a single model that has wide coverage of vocabulary and syntax of a large number of languages, but also has the drawback in that a single model must be able to express information about all the languages in the training set within its limited parameter budget. Thus, it is reasonable to expect that this model may achieve worse accuracy than a model created specifically to handle a particular source language.

In the following, we will consider adaptation methods that focus on tailoring a more general model (i.e. bi-source or universal) to a more specific model (i.e. single-source or bi-source).

2.2 Adaptation to New Languages

As noted in the introduction, there are two major requirements: the *accuracy* of the system is important and the *training time required* from when we learn of a need for translation to when we can first start producing adequate results. Throughout the discussion, we will compare various adaptation paradigms with respect to these two aspects.

2.2.1 Adaptation by Fine-tuning

Our first adaptation method, inspired by Zoph et al. (2016) is based on fine-tuning to the source language of interest. Within our experiments, we will test this setting, but also make two distinctions between the types of adaptation:

Seed Model Variety: Zoph et al. (2016) performed experiments taking a bilingual system trained on a different language (e.g. French) and adapting it to a new LRL (e.g. Uzbek). We can also take universal model and adapt it to the new language, a setting that we examine (to our knowledge, for the first time) in this work.

Warm vs. Cold Start: Another contrast is whether we have training data for the LRL of interest while training the original system, or whether we only receive training data *after* the original model has already been trained. We call the former *warm start*, and the latter *cold start*. Intuitively, we expect warm-start training to perform better, as having access to the LRL of interest during the training of the original model will ensure that it can handle the LRL to some extent. However, the cold-start scenario is also of interest: we may want to spend large amounts of time training a strong model, then quickly adapt to a new language that we have never seen before in our training data as data becomes available. For the cold-start models, we start with a model that is only trained on the HRL similar to the LRL (Bi$^-$), or a model trained

---

[2] Translation into LRLs, is a challenging and interesting problem in it's own right, but beyond the scope of the paper.

[3] "Related" could mean different things: typologically related or having high lexical overlap. In our experiments our LRLs are all selected to have an helper that is highly similar in both aspects, but choosing an appropriate helper when this is not the case is an interesting problem for future work.

[4] In contrast to Gu et al. (2018), who train on 10 languages. Malaviya et al. (2017); Tiedemann (2018) train NMT on over 1,000 languages, but only as a feature extractor for downstream tasks; MT accuracy itself is not evaluated.

LRL	train	dev	test	HRL	train
aze	5.94k	671	903	tur	182k
bel	4.51k	248	664	rus	208k
glg	10.0k	682	1,007	por	185k
slk	61.5k	2,271	2,445	ces	103k

Table 1: Data sizes in sentences for LRL/HRL pairs

on all languages *but* the LRL (All^{-}).

2.2.2 Similar-Language Regularization

One problem with adapting to a small amount of data in the target language is that it will be very easy for the model to over-fit to the small training set. To alleviate this problem, we propose a method of *similar language regularization*: while training to adapt to the language of interest, we also add some data from another similar HRL that has sufficient resources to help prevent overfitting. We do this in two ways:

Corpus Concatenation: Simply concatenate the data from the two corpora, so that we have a small amount of data in the LRL, and a large amount of data in the similar HRL.

Balanced Sampling: Every time we select a minibatch to do training, we either sample it from the LRL, or from the HRL according to a fixed ratio. We try different sampling strategies, including sampling with a 1-to-1 ratio, 1-to-2 ratio, and 1-to-4 ratio for the LRL and HRL respectively.

3 Experiments

3.1 Experimental Setup

We perform experiments on the 58-language-to-English TED corpus (Qi et al., 2018), which is ideal for our purposes because it has a wide variety of languages over several language families, some high-resourced and some low-resourced. Like Qi et al. (2018), we experiment with Azerbaijani (aze), Belarusian (bel), and Galician (glg) to English, and also additionally add Slovak (slk), a slightly higher resourced language, for contrast. These languages are all paired with a similar HRL: Turkish (tur), Russian (rus), Portuguese (por), and Czech (ces) respectively. Data sizes are shown in Table 1.

Models are implemented using `xnmt` (Neubig et al., 2018), commit `8173b1f`, and start with the recipe for training on IWSLT TED[5]. The model consists of an attentional neural machine translation model (Bahdanau et al., 2015), using bi-directional LSTM encoders, 128-dimensional word embeddings, 512-dimensional hidden states, and a standard LSTM-based decoder.

Following standard practice (Sennrich et al., 2016; Denkowski and Neubig, 2017), we break low-frequency words into subwords using the `sentencepiece` toolkit.[6] There are two alternatives for creating subword units: *jointly* learning subwords over all source language, or *separately* learning subwords for each source language, then taking the union of all the subword vocabularies as the vocabulary for the multilingual model. Previous work on multilingual training has preferred the former (Nguyen and Chiang, 2017), but in this paper we use the latter for two reasons: (1) because data in the LRL will not affect the subword units from the other languages, in the cold-start scenario we can postpone creation of subword units for the LRL until directly before we start training on the LRL itself, and (2) we need not be concerned with the LRL being "overwhelmed" by the higher-resourced languages when calculating statistics used in the creation of subword units, because all languages get an equal share.[7] In the experiments, we use a subword vocabulary of 8,000 for each language.

We also compare with two additional baselines: phrase-based MT implemented in `Moses`,[8] and unsupervised NMT implemented in `undreamt`.[9] `Moses` is trained on the bilingual data only (training multilingually reduced average accuracy), and `undreamt` is trained on all monolingual data available for the LRL and English.

3.2 Experimental Results

Table 2 shows our main translation results, with warm-start scenarios in the upper half and cold-start scenarios in the lower half.

Does Multilingual Training Help? To answer this question, we can compare the warm-start Sing., Bi, and All settings, and find that the answer is a resounding yes, gains of 7-13 BLEU points are obtained by going from single-source to bi-source or all-source training, corroborating previous work (Gu et al., 2018). Bi-source models tend to perform slightly better than all-source models, indicating that given identical parameter

---

[5] Found in `examples/stanford-iwslt/`

[6] `https://github.com/google/sentencepiece`, using the `unigram` training setting.

[7] Preliminary experiments found both comparable: with scores of 20.1 and 19.4 for separate and joint respectively.

[8] `http://statmt.org/moses`

[9] `https://github.com/artetxem/undreamt`

	Strategy	aze/tur	bel/rus	glg/por	slk/ces	Avg.
	Phrase-based	5.9	10.5	22.3	23.0	15.4
	Unsupervised NMT	0.0	0.3	0.4	0.0	0.2
Warm Start	Sing.	2.7	2.8	16.2	24.0	11.4
	Bi	10.9	15.8	27.3	26.5	20.1
	All	9.7	16.7	26.5	25.0	19.5
	Bi →Sing.	11.4	16.3	27.5	27.1	20.6
	All→Sing.	10.1	17.5	28.2	27.4	20.8
	All→Bi	**11.7**	**18.3**	28.8	28.2	**21.8**
	All→Bi 1-1	10.2	18.3	28.8	**28.3**	21.4
	All→Bi 1-2	11.0	17.5	**29.1**	28.2	21.4
	All→Bi 1-4	11.1	17.9	28.5	27.9	21.3
Cold Start	Bi$^-$	3.8	2.5	8.6	5.4	5.1
	All$^-$	3.7	3.5	15.5	7.3	7.5
	Bi$^-$ →Sing.	8.7	11.8	25.4	26.8	18.2
	All$^-$→Sing.	8.8	15.3	26.5	27.6	19.5
	All$^-$→Bi	10.7	**17.4**	28.4	28.0	**21.2**
	All$^-$→Bi 1-1	10.5	16.0	28.0	**28.2**	20.7
	All$^-$→Bi 1-2	10.7	17.1	28.3	27.9	21.0
	All$^-$→Bi 1-4	**11.0**	17.4	**28.4**	27.6	21.1

Table 2: BLEU for single-source (Sing.), bi-source (Bi), and all-source universal (All) models, with adapted counterparts. 1-1, 1-2, 1-4 indicate balanced sampling from §2.2. Bold indicates highest score.

capacity, training on a highly resourced language is effective. Comparing with the phrase-based baseline, as noted by Koehn and Knowles (2017) NMT tends to underperform on low-resource settings when trained only on the data available for these languages. However, multilingual training of any variety quickly remedies this issue; all outperform phrase-based handily.

More interestingly, examining the cold-start results, we can see that even systems with *no* data in the target language are able to achieve non-trivial accuracies, up to 15.5 BLEU on glg-eng. Interestingly, in the cold-start scenario, the All$^-$ model bests the Bi$^-$ model, indicating that massively multilingual training is more useful in this setting. In contrast, the unsupervised NMT model struggles, achieving a BLEU score of around 0 for all language pairs – this is because unsupervised NMT requires high-quality monolingual embeddings from the same distribution, which can be trained easily in English, but are not available in the low-resource languages we are considering.

Does Adaptation Help? Regarding adaptation, we can first observe that regardless of the original model and method for adaptation, adaptation is helpful, particularly (and unsurprisingly) in the cold-start case. When adapting directly to only the target language ("→Sing."), adapting from the massively multilingual model performs better, indicating that information about all input languages is better than just a single language. Next, comparing with our proposed method of adding similar language regularization ("→Bi"), we can see that this helps significantly over adapting directly to the LRL, particularly in the cold-start case where we can observe gains of up to 1.7 BLEU points. Finally, in our data setting, corpus concatenation outperforms balanced sampling in both the cold-start and warm-start scenarios.

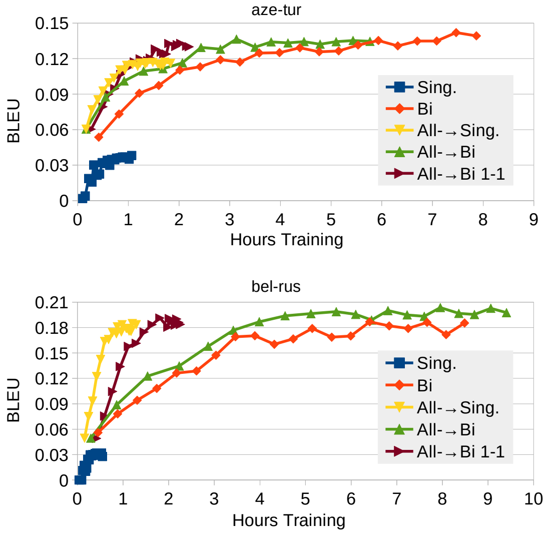

Figure 1: Example of adaptation on the aze-eng and bel-eng development sets

How Can We Adapt Most Efficiently? Finally, we revisit adapting to new languages efficiently, with Figure 1 showing BLEU vs. hours training for the aze/tur and bel/rus source language pairs (others were similar). We can see that in all cases the cold-start models (All$^-$ →) either outperform or are comparable in final accuracy to the from-scratch single-source and bi-source models. In addition, all of the adapted models converge faster than the bi-source from-scratch trained models, indicating that adapting from seed models is a good strategy for rapid construction of MT systems in new languages. Comparing the cold-start adaptation strategies, we can see that in general, the higher the density of target language training data, the faster the training converges to a solution, but the worse the final solution is. This suggests that there is a speed/accuracy tradeoff in the amount of similar language regularization we apply during fine-tuning.

4 Related Work

While adapting MT systems to new languages is a long-standing challenge (Schultz and Black, 2006; Jabaian et al., 2013), multilingual NMT is highly promising in its ability to abstract across language boundaries (Firat et al., 2016; Ha et al., 2016; Johnson et al., 2016). Results on multilingual training for low-resource translation (Gu et al., 2018; Qi et al., 2018) further demonstrates this potential, although these works do not consider adaptation to new languages, the main focus of our work. Notably, we did not examine partial freezing of parameters, another method proven useful for cross-lingual adaptation (Zoph et al., 2016); this is orthogonal to our multi-lingual training approach but the two methods could potentially be combined. Finally, unsupervised NMT approaches (Artetxe et al., 2017; Lample et al., 2018, 2017) require no parallel data, but rest on strong assumptions about high-quality comparable monolingual data. As we show, when this assumption breaks down these methods fail to function, while our cold-start methods achieve non-trivial accuracies even with no monolingual data.

5 Conclusion

This paper examined methods to rapidly adapt MT systems to new languages by fine-tuning. In both warm-start and cold-start scenarios, the best results were obtained by adapting a pre-trained universal model to the low-resource language while regularizing with similar languages.

Acknowledgements

The authors thank Jaime Carbonell, Xinyi Wang, Rebecca Knowles, Arya McCarthy, and anonymous reviewers for their constructive comments on this paper.

This work is sponsored by Defense Advanced Research Projects Agency Information Innovation Office (I2O). Program: Low Resource Languages for Emergent Incidents (LORELEI). Issued by DARPA/I2O under Contract No. HR0011-15-C0114. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *Proc. WNMT*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proc. NAACL*, pages 866–875.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. 2018. Universal neural machine translation for extremely low resource languages. *Proc. NAACL*.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*.

Bassam Jabaian, Laurent Besacier, and Fabrice Lefevre. 2013. Comparison and combination of lightly supervised approaches for language portability of a spoken language understanding system. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(3):636–648.

Melvin Johnson et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proc. EMNLP*, pages 1700–1709.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *Proc. WNMT*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

William D. Lewis. 2010. Haitian Creole: how to build and ship an MT engine from scratch in 4 days, 17 hours, & 30 minutes. In *Proc. EAMT*.

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *Proc. EMNLP*.

Robert Munro. 2010. Crowdsourced translation for emergency response in Haiti: the global collaboration of local knowledge. In *Proc. AMTA Workshop on Collaborative Crowdsourcing for Translation*.

Graham Neubig, Yuichiroh Matsubayashi, Masato Hagiwara, and Koji Murakami. 2011. Safety information mining - what can NLP do in a disaster -. In *Proc. IJCNLP*, pages 965–973.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, John Hewitt, Rachid Riad, and Liming Wang. 2018. XNMT: The extensible neural machine translation toolkit. In *Proc. AMTA*, Boston.

Toan Q. Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for neural machine translation. pages 296–301.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proc. NAACL*, New Orleans, USA.

Tanja Schultz and Alan W Black. 2006. Challenges with rapid adaptation of speech translation systems to new language pairs. In *Proc. ICASSP*, volume 5. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. ACL*, pages 1715–1725.

Kate Starbird, Grace Muzny, and Leysia Palen. 2012. Learning from the crowd: Collaborative filtering techniques for identifying on-the-ground Twitterers during mass disruptions. In *Proc. ISCRAM*.

Jörg Tiedemann. 2018. Emerging language spaces learned from massively multilingual corpora. *arXiv preprint arXiv:1802.00273*.

Sarah Vieweg, Amanda L Hughes, Kate Starbird, and Leysia Palen. 2010. Microblogging during two natural hazards events: what Twitter may contribute to situational awareness. In *Proc. CHI*, pages 1079–1088.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proc. EMNLP*, pages 1568–1575.

Compact Personalized Models for Neural Machine Translation

Joern Wuebker, Patrick Simianer, John DeNero
Lilt, Inc.
`first_name@lilt.com`

Abstract

We propose and compare methods for gradient-based domain adaptation of self-attentive neural machine translation models. We demonstrate that a large proportion of model parameters can be frozen during adaptation with minimal or no reduction in translation quality by encouraging structured sparsity in the set of offset tensors during learning via group lasso regularization. We evaluate this technique for both batch and incremental adaptation across multiple data sets and language pairs. Our system architecture—combining a state-of-the-art self-attentive model with compact domain adaptation—provides high quality personalized machine translation that is both space and time efficient.

1 Introduction

Professional translators typically translate a collection of related documents drawn from a domain for which they have a set of previously translated examples. Domain adaptation is critical to providing high quality suggestions for interactive machine translation and post-editing interfaces. When many translators use the same shared service, the system must train and apply a *personalized* adapted model for each user. We describe a system architecture and training method that achieve high space efficiency, time efficiency, and translation performance by encouraging structured sparsity in the set of offset tensors stored for each user.

Effective model personalization requires both *batch* adaptation to an in-domain training set, as well as *incremental* adaptation to the test set. Batch adaptation is applied when a user uploads relevant translated documents before starting to work. Incremental adaptation is applied when a user provides a correct translation of each segment just after receiving machine translation suggestions, and the system is able to train on that correction before generating suggestions for the next segment. This is referred to as *a posteriori* adaptation by Turchi et al. (2017). Our experiments compare both types of adaptation. There are cases for which incremental adaptation achieves better performance using fewer examples, as examples drawn directly from the test set are often highly relevant to subsequent parts of that test set. There are also cases for which the gains from both types of domain adaptation are additive.

The time required to translate and to adapt both must be minimal in a personalized translation service. Interactive translation requires suggestions to be generated at typing speed, and incremental adaptation must occur within a few hundred milliseconds to keep up with a translator's typical workflow. The service can be expected to store models for a large number of users and dynamically load and adapt models for many active users concurrently. Therefore, minimizing the number of parameters stored for each user's personalized model is important both for reducing storage requirements and latency. We achieve space and time efficiency by representing each user's model as an offset from the unadapted baseline parameters and encouraging most offset tensors to be zero during adaptation.

We show that group lasso regularization can be applied to a self-attentive Transformer model to freeze up to 75% of the parameters with minimal or no loss of adapted translation quality across experiments on four English→German data sets. We confirm these findings for six additional language pairs.

2 Related Work

There is extensive work on incremental adaptation from human post edits or simulated post edits, both for statistical machine translation (Green et al., 2013; Denkowski et al., 2014a,b; Wuebker et al., 2015) and neural machine translation (Peris et al.,

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 881–886
Brussels, Belgium, October 31 - November 4, 2018.

2017; Turchi et al., 2017; Karimova et al., 2017). Both Turchi et al. (2017) and Karimova et al. (2017) apply vanilla fine-tuning algorithms. In addition to fine-tuning towards user corrections, the former applies *a priori* adaptation to retrieved data that is similar to the incoming source sentences. Peris et al. (2017) propose a variant of fine-tuning with passive-aggressive learning algorithms. In contrast to these papers, where all model parameters are possibly altered during training, this work focuses on space efficiency of the adapted models.

Regularization methods that promote or enforce sparsity have been previously used in the context of sparse feature models for SMT: Duh et al. (2010) presented an application of multi-task learning via ℓ_1/ℓ_2 regularization for feature selection in an N-best reranking task. A similar approach, employing ℓ_1/ℓ_2 regularization for feature selection and multi-task learning, was developed by Simianer et al. (2012) and Simianer and Riezler (2013) for tuning of SMT systems. Both works report improvements from regularization.

Techniques for enforcing sparse models using ℓ_1 regularization during stochastic gradient descent optimization were previously developed for linear models (Tsuruoka et al., 2009).

An extremely space efficient method for personalized model adaptation is presented by Michel and Neubig (2018). Here, adaptation is performed solely on the output vocabulary bias vector. Another notable approach for creating compact models is student-teacher-training or knowledge distillation (Kim and Rush, 2016). To the best of our knowledge, this has not been applied in a domain adaptation setting.

3 Self-Attentive Translation Model

The neural machine translation systems used in this work are based on the Transformer model introduced by Vaswani et al. (2017), which uses self-attention rather than recurrent or convolutional layers to aggregate information across words. In addition to its superior performance, its main practical advantage over recurrent models is faster training.

The Transformer follows the encoder-decoder paradigm. Source word vectors $x_1, \dots, x_m$ are chosen from an embedding matrix X_e. A series of stacked encoder layers generate intermediate representations $z_1, \dots, z_m$. Each layer of the encoder consists of two sub-layers: a multi-head *self-attention* layer that uses scaled dot-product attention over all source positions, followed by a feed-forward *filter* layer. Layer normalization (Ba et al., 2016), dropout (Srivastava et al., 2014), and residual connections (He et al., 2016) are applied to each sub-layer.

A series of stacked decoder layers produces a sequence of target word vectors $y_1, \dots, y_n$. Each decoder layer has three sub-layers: self-attention, encoder-attention, and a filter. For target position j, the *self-attention* layer can attend to any previous target position $j' \in [1, j]$, with target words offset by one so that representations at j can observe word $j-1$, but not word j. The *encoder-attention* layer can attend to the final encoder state z_i for any source position $i \in [1, m]$. Observed target word vectors are chosen from an embedding matrix Y_e, and target word j is predicted from y_j via a soft-max layer parameterized by an output projection matrix Y_o.

The encoders in this work have six layers that have a *self-attention* sub-layer size of 256 and a *filter* sub-layer size of 512. Each filter performs two linear transformations and a ReLU activation:

$$f(x) = \max(0, xW_1 + b_1)W_2 + b_2.$$

The decoders in this work have three layers, and all sub-layer sizes are 256. The decoder sub-layers are simplified versions of those described in Vaswani et al. (2017): The *filter* sub-layers perform only a single linear transformation, and layer normalization is only applied once per decoder layer after the *filter* sub-layer.

Unlike in Vaswani et al. (2017), none of X_e, Y_e, or Y_o share parameters in our TensorFlow[1] implementation. Baseline models are optimized with Adam (Kingma and Ba, 2015).

4 Compact Adaptation

4.1 Fine Tuning

Personalized machine translation requires batch adaptation to a domain-relevant bitext (such as a user provided translation memory) as well as incremental adaptation to the test set. We apply fine-tuning, which involves continuing to train model parameters with a gradient-based method on domain-relevant data, as a simple and effective method for neural translation domain adaptation (Luong and Manning, 2015). The fine-tuned model without regularization and clipping is denoted as the *Full Model*. Confirming previous work, we found that

---

[1] `https://www.tensorflow.org/`

stochastic gradient descent (SGD) is the most effective optimizer for fine tuning (Turchi et al., 2017). In our experiments, batch adaptation uses a batch size of 7000 words for 10 Epochs and a fixed learning rate of 0.1, dropout of 0.1, and label smoothing with $\epsilon_{ls} = 0.1$ (Szegedy et al., 2016).

Incremental adaptation uses a batch size of one and a learning rate of 0.01. To ensure a strong adaptation effect within a single document, we set dropout and label smoothing to zero and perform up to three SGD updates on each segment. After each update, we measure the model perplexity on the current training example and continue with another update if the perplexity is still above 1.5.

4.2 Offset Tensors

In a personalized translation service, adapted models need to be loaded quickly, so a space-efficient representation is critical for time efficiency as well. Production speed requirements using contemporary cloud hardware limit model sizes to roughly 10M parameters per user, while a high-quality baseline Transformer model typically requires 35M parameters or more. We propose to store the parameters of an adapted model as an offset from the baseline model. Each tensor is a sum $W = W_b + W_u$, where W_b is from the baseline model and is shared across all adapted models, while the offset W_u is specific to an individual user domain. Space efficiency is achieved by only storing W_u for a subset of tensors and setting the rest of the offset tensors to zero.

One approach to achieving model sparsity is to manually partition the network into a small number of regions and systematically evaluate translation performance when storing offsets for only one region. We define five distinct regions, which are evaluated in isolation: Outer layers (the first and last layers of both encoder and decoder), inner layers (all the remaining layers), the two embedding matrices X_e and Y_e, and the output projection matrix Y_o. The latter three are each stored as a single matrix and each contributes 10.3M parameters to the full model size in English→German. During adaptation, the embedding matrices are only updated for vocabulary present in the training examples, and so the offsets can be stored efficiently as a sparse collection of columns. The same principle can be applied to the output projection matrix by only updating parameters corresponding to vocabulary items that appears in the adaptation examples (denoted *Sparse Output Proj.* in Table 1).

A second approach to achieving model sparsity is to use a procedure to select the subset of offset tensors that are stored. For example, we evaluate a simple policy that stores an offset for all tensors whose average change in parameter values is higher than a threshold. This set is selected on a development domain and held fixed for all other domains. We refer to this method as *fixed* adaptation.

4.3 Tensor Selection via Group Lasso

A group sparse regularization penalty such as group lasso can be applied to the offset tensors for simultaneous regularization and tensor selection. This penalty drives entire offset tensors to zero, so that they do not need to be stored or loaded. We add the following regularization term to the loss function (Scardapane et al., 2017):

$$R_{\ell_{1,2}}(\mathcal{T}) = \sum_{T \in \mathcal{T}} \sqrt{|T|} \|\Delta T\|_2 \qquad (1)$$

$$\|\Delta T\|_2 = \sum_{\tau \in T} \Delta\tau^2 \qquad (2)$$

Here, each tensor corresponds to one group. $\mathcal{T}$ denotes the set of all tensors in the model, $\tau \in T$ the set of all weights within a single tensor and $\Delta\tau$ the size of the offset for τ. Note that we are regularizing the *difference* between the parameters of the adapted model and the baseline model, rather than regularizing the full network parameters directly. In this way, we maintain the expressive power of the full network while minimizing the size of the adapted models. Group lasso regularization is equivalent to ℓ_1 regularization when the group size is 1. Sparsity among groups is encouraged because the ℓ_1 norm serves as a convex proxy for the ℓ_0 norm, which would explicitly penalize the number of non-zero elements (Yuan and Lin, 2006). To facilitate tensor selection, we define a threshold ϑ to clip offset tensors ΔT with average weight $\frac{1}{|T|} \sum_{\tau \in T} \Delta\tau < \vartheta$ to zero. Both the threshold ϑ and the regularization weight λ were manually tuned on a development domain and set to $\vartheta = 10^{-4}$ and $\lambda = 10^{-6}$. We apply clipping to all tensors except the embedding and output projection matrices X_e, Y_e and Y_o. As our production constraints allow us to retain only one of the three, we pre-select the sparse output projection as part of the model and exclude the embedding matrices from adaptation. This method will be denoted as *Lasso*.

		User1		User2		Autodesk		IWSLT	
	# Param.	Batch	Incr.	Batch	Incr.	Batch	Incr.	Batch	Incr.
Baseline	36.2M	35.7		32.7		40.3		25.9	
Full Model	25.8M	47.5	48.2	44.2	34.8	47.7	46.6	27.5	26.3
Outer Layers	2.2M	45.0	47.9	36.4	33.1	45.5	44.7	27.3	26.1
Inner Layers	2.7M	45.4	47.2	36.7	33.6	45.5	43.9	27.8	26.5
Encoder Embed.	5.0M	41.7	41.8	33.6	32.9	42.5	41.6	27.4	26.4
Decoder Embed.	5.5M	36.6	37.6	33.0	33.0	40.8	40.5	26.2	25.9
Output Proj.	10.3M	44.2	46.0	38.1	34.5	45.2	42.9	27.1	26.5
Sparse Output Proj. (*)	5.5M	43.5	46.7	39.7	34.7	45.5	43.3	27.1	26.7
(*) + Fixed	6.9M	46.4	47.8	42.3	30.9	47.6	43.7	27.3	26.0
(*) + Lasso	6.7M	47.6	46.6	43.1	33.2	47.9	41.5	27.5	27.0
Full Model Batch+Incr.	25.9M	50.6		41.8		52.6		27.0	
(*) + Lasso Batch+Incr.	9.2M	51.3		39.1		51.1		27.6	
Repetition Rate Source		11.0		8.8		18.3		9.2	
Repetition Rate Target		9.4		8.7		17.5		7.5	

Table 1: Experimental results in Bleu (%) on the English→German data. We evaluate changes to each region of the network separately. In combination with sparse output projection, we also evaluate a *fixed* selection of parameters chosen by thresholding and a set selected dynamically for each data set using group *lasso*. The two bottom rows show repetition rates in % for the source and target sides of the test data.

5 Experiments

5.1 Data

We first evaluate all techniques on an English→German Transformer network trained on 98M parallel sentence pairs. We apply byte pair encoding (Sennrich et al., 2016) separately to each language and obtain vocabularies with 40K unique tokens each. We refer to the unadapted model as *Baseline*. We evaluate on four domains. For development, we use a data set labeled *User1* that was gathered from a user of the browser-based CAT (computer-aided translation) tool Lilt[2] and contains documents from the financial domain with 48K segments for batch adaptation and 1790 segments for testing and incremental adaptation. We further evaluate on a second user test set *User2* (technical support, 31k batch adaptation, 1000 test segments); the public Autodesk corpus[3], where we select the first 20k segments for batch adaptation and the next 1000 segments for testing; and the IWSLT corpus[4] (semi-technical talks), where we use all provided 206K sentences for batch adaptation and the *dev2010* set (888 sentences) for testing. The overall best performing compact adaptation technique, group lasso regularization, is further evaluated on six other language pairs trained using production data sets collected from Lilt's user base: English↔French, English↔Russian and English↔Chinese. Adaptation is performed on user data from various domains (technical manuals, finance, legal), each with 8k-10k segments for batch adaptation and 2000 segments for testing and incremental adaptation. Translation quality is evaluated using the cased Bleu (Papineni et al., 2002) measure.

5.2 Results

Table 1 shows English→German results. Full model adaptation, where all offsets are stored, improves over the baseline in all cases to various degrees. This full model contains only 25.8M parameters, as offsets for both embedding matrices are stored as sparse collections of columns for the vocabulary present in the adaptation data. Next, we evaluate the impact of storing offsets only for one region at a time. We observe that among the three vocabulary matrices, the output projection Y_o has the strongest impact on quality, which is not dimin-

---

[2] https://lilt.com

[3] https://autodesk.app.box.com/Autodesk-PostEditing

[4] http://workshop2017.iwslt.org/

	en→fr	fr→en	en→ru	ru→en	en→zh	zh→en	Avg.
Baseline	28.8	35.8	10.7	29.2	19.9	18.9	23.9
Full Model Batch+Incr.	36.6	49.6	21.0	42.1	40.6	46.6	39.4
(*) + Lasso Batch+Incr.	36.0	46.3	19.8	42.7	39.3	45.0	38.2

Table 2: Experimental results in BLEU (%) on six production language pairs. We compare the unadapted baseline model with a full model and the model with sparse output projection and group lasso, both with application of batch and incremental adaptation.

ished by storing a sparse variant that is restricted only to observed vocabulary.

In addition, we evaluate two methods of choosing a subset of tensors procedurally. We first experiment with a fixed subset of tensor offsets that was chosen by selecting all tensors for which parameters were offset by more than 0.002 on average after batch adaptation on the *User1* data set. This simple procedure approaches the performance of full model adaptation, but stores only 27% of its parameters. Dynamically selecting tensor offsets for each data set using group lasso regularization improves performance on 6 out of 8 data conditions.

The combination of batch and incremental adaptation yields further improvements, with the exception of the *User2* and *IWSLT* tasks, where incremental adaptation overall performs not as well as batch adaptation. For these tasks, both tests sets exhibit lower repetition rates[5] (Cettolo et al., 2014) than the test sets for the two other tasks (see the two bottom lines in Table 1). The *User2* test set is furthermore a random sample of non-consecutive text from a translation memory, which is suboptimal for incremental learning.

Altogether, we are able to achieve translation performance similar to full model adaptation with 25% of the total network parameters. Note that due to the selection of entire tensors with groupwise regularization, there is nearly zero space overhead incurred by storing a sparse set of offset tensors.

Table 2 confirms our main findings on six other language pairs. We observe average improvements of 14.3 BLEU with our final compact model, which compares to 15.5 BLEU for full model adaptation.

6 Conclusion

We describe an efficient approach to personalized machine translation that stores a sparse set of tensor offsets for each user domain. Group lasso regularization applied to the offsets during adaptation achieves high space and time efficiency while yielding translation performance close to a full adapted model, for both batch and incremental adaptation and their combination.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450.*

Nicola Bertoldi, Patrick Simianer, Mauro Cettolo, Katharina Wäschle, Marcello Federico, and Stefan Riezler. 2014. Online adaptation to post-edits for phrase-based statistical machine translation. *Machine Translation*, 28(3-4):309–339.

Mauro Cettolo, Nicola Bertoldi, and Marcello Federico. 2014. The repetition rate of text as a predictor of the effectiveness of machine translation adaptation. In *Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 166–179, Vancouver, Canada.

Michael Denkowski, Chris Dyer, and Alon Lavie. 2014a. Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 395–404, Gothenburg, Sweden. Association for Computational Linguistics.

Michael Denkowski, Alon Lavie, Isabel Lacruz, and Chris Dyer. 2014b. Real time adaptive machine translation for post-editing with cdec and transcenter. In *Proceedings of the EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 72–77, Gothenburg, Sweden. Association for Computational Linguistics.

Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, Hideki Isozaki, and Masaaki Nagata. 2010. N-best reranking by multitask learning. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 375–383, Uppsala, Sweden. Association for Computational Linguistics.

---

[5] Repetition rates have been confirmed to be a suitable indicator for gains through incremental adaptation in numerous works (Wuebker et al., 2015; Bertoldi et al., 2014).

Spence Green, Sida Wang, Daniel Cer, and Christopher D. Manning. 2013. The efficacy of human post-editing for language translation. In *ACM CHI Conference on Human Factors in Computing Systems*, Paris, France.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sariya Karimova, Patrick Simianer, and Stefan Riezler. 2017. A user-study on online adaptation of neural machine translation to human post-edits. *CoRR*, abs/1712.04853.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Diederik Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 312–318. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

Álvaro Peris, Luis Cebrián, and Francisco Casacuberta. 2017. Online learning for neural machine translation post-editing. *CoRR*, abs/1706.03196.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. 2017. Group sparse regularization for deep neural networks. *Neurocomput.*, 241(C):81–89.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Patrick Simianer and Stefan Riezler. 2013. Multi-task learning for improved discriminative training in SMT. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 292–300, Sofia, Bulgaria.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 11–21, Jeju Island, South Korea. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 477–485. Association for Computational Linguistics.

Marco Turchi, Matteo Negri, M Amin Farajian, and Marcello Federico. 2017. Continuous learning from human post-edits for neural machine translation. *The Prague Bulletin of Mathematical Linguistics*, 108(1):233–244.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Joern Wuebker, Spence Green, and John DeNero. 2015. Hierarchical incremental adaptation for statistical machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1059–1065.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Self-Governing Neural Networks for On-Device Short Text Classification

Sujith Ravi
Google Research
Mountain View, CA, USA
sravi@google.com

Zornitsa Kozareva
Google
Mountain View, CA, USA
zornitsa@kozareva.com

Abstract

Deep neural networks reach state-of-the-art performance for wide range of natural language processing, computer vision and speech applications. Yet, one of the biggest challenges is running these complex networks on devices such as mobile phones or smart watches with tiny memory footprint and low computational capacity. We propose on-device Self-Governing Neural Networks (SGNNs), which learn compact projection vectors with local sensitive hashing. The key advantage of SGNNs over existing work is that they surmount the need for pre-trained word embeddings and complex networks with huge parameters. We conduct extensive evaluation on dialog act classification and show significant improvement over state-of-the-art results. Our findings show that SGNNs are effective at capturing low-dimensional semantic text representations, while maintaining high accuracy.

1 Introduction

Deep neural networks are one of the most successful machine learning methods outperforming many state-of-the-art machine learning methods in natural language processing (Sutskever et al., 2014), speech (Hinton et al., 2012) and visual recognition tasks (Krizhevsky et al., 2012). The availability of high performance computing has enabled research in deep learning to focus largely on the development of deeper and more complex network architectures for improved accuracy. However, the increased complexity of the deep neural networks has become one of the biggest obstacles to deploy deep neural networks on-device such as mobile phones, smart watches and IoT (Iandola et al., 2016). The main challenges with developing and deploying deep neural network models on-device are (1) the tiny memory footprint, (2) inference latency and (3) significantly low computational capacity compared to high performance computing systems such as CPUs, GPUs and TPUs on the cloud.

There are multiple strategies to build lightweight text classification models for on-device. One can create a small dictionary of common *input* → *category* mapping on the device and use a naive look-up at inference time. However, such an approach does not scale to complex natural language tasks involving rich vocabularies and wide language variability. Another strategy is to employ fast sampling techniques (Ahmed et al., 2012; Ravi, 2013) or incorporate deep learning models with graph learning like (Bui et al., 2017, 2018), which result in large models but have proven to be extremely powerful for complex language understanding tasks like response completion (Pang and Ravi, 2012) and Smart Reply (Kannan et al., 2016).

In this paper, we propose Self-Governing Neural Networks (SGNNs) inspired by projection networks (Ravi, 2017). SGNNs are on-device deep learning models learned via embedding-free projection operations. We employ a modified version of the locality sensitive hashing (LSH) to reduce input dimension from millions of unique words/features to a short, fixed-length sequence of bits. This allows us to compute a projection for an incoming text very fast, on-the-fly, with a small memory footprint on the device since we do not need to store the incoming text and word embeddings. We evaluate the performance of our SGNNs on Dialogue Act classification, because (1) it is an important step towards dialog interpretation and conversational analysis aiming to understand the intent of the speaker at every utterance of the conversation and (2) deep learning methods reached state-of-the-art (Lee and Dernoncourt, 2016; Khanpour et al., 2016; Tran et al., 2017; Ortega and Vu, 2017).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 887–893
Brussels, Belgium, October 31 - November 4, 2018.

The main contributions of the paper are:

- Novel Self-Governing Neural Networks (SGNNs) for on-device deep learning for short text classification.
- Compression technique that effectively captures low-dimensional semantic text representation and produces compact models that save on storage and computational cost.
- On the fly computation of projection vectors that eliminate the need for large pre-trained word embeddings or vocabulary pruning.
- Exhaustive experimental evaluation on dialog act datasets, outperforming state-of-the-art deep CNN (Lee and Dernoncourt, 2016) and RNN variants (Khanpour et al., 2016; Ortega and Vu, 2017).

2 Self-Governing Neural Networks

We model the Self-Governing network using a *projection* model architecture (Ravi, 2017). The *projection* model is a simple network with dynamically-computed layers that encodes a set of efficient-to-compute operations which can be performed directly on device for inference. The model defines a set of efficient "projection" functions $\mathbb{P}(\vec{x}_i)$ that project each input instance $\vec{x}_i$ to a different space $\Omega_{\mathbb{P}}$ and then performs learning in this space to map it to corresponding outputs y_i^p. A very simple projection model comprises just few operations where the inputs $\vec{x}_i$ are transformed using a series of T projection functions $\mathbb{P}^1, ..., \mathbb{P}^T$ followed by a single layer of activations.

2.1 Model Architecture

In this work, we design a Self-Governing Neural Network (SGNN) using multi-layered locality-sensitive projection model. Figure 1 shows the model architecture of the on-device SGNN network. The *self-governing* property of this network stems from its ability to learn a model (e.g., a classifier) without having to initialize, load or store any feature or vocabulary weight matrices. In this sense, our method is a truly **embedding-free** approach unlike majority of the widely-used state-of-the-art deep learning techniques in NLP whose performance depends on embeddings pre-trained on large corpora. Instead, we use the projection functions to dynamically transform each input to a low-dimensional representation. Furthermore, we stack this with additional layers and non-linear activations to achieve deep, non-linear combinations of projections that permit the network to learn complex mappings from inputs x_i to outputs y_i. An SGNN network is shown below:

$$
\begin{aligned}
i_p &= [\mathbb{P}^1(x_i), ..., \mathbb{P}^T(x_i)] && (1)\\
h_p &= \sigma(W_p \cdot i_p + b_p) && (2)\\
h_t &= \sigma(W_t \cdot h_{t-1} + b_t) && (3)\\
y_i &= \texttt{softmax}(W_o \cdot h_k + b_o) && (4)
\end{aligned}
$$

where, i_p refers to the output of projection operation applied to input x_i, h_p is applied to projection output, h_t is applied at intermediate layers of the network with depth k followed by a final `softmax` activation layer at the top. In a k-layer SGNN, h_t, where $t = p, p+1, ..., p+k-1$ refers to the k subsequent layers after the projection layer. W_p, W_t, W_o and b_p, b_t, b_o represent trainable weights and biases respectively.

The projection transformations use pre-computed parameterized functions, i.e., they are not trained during the learning process, and their outputs are concatenated to form the hidden units for subsequent operations. Each input text x_i is converted to an intermediate feature vector (via raw text features such as skip-grams) followed by projections.

$$x_i \xrightarrow{\mathbb{F}} \vec{x_i} \xrightarrow{\mathbb{P}} [\mathbb{P}^1(x_i), ..., \mathbb{P}^T(x_i)] \qquad (5)$$

On-the-fly Computation. The transformation step $\mathbb{F}$ dynamically extracts features from the raw input. Text features (e.g., skip-grams) are converted into feature-ids f_j (via hashing) to generate a sparse feature representation $\vec{x_i}$ of feature-id, weight pairs (f_j, w_j) . This intermediate feature representation is passed through projection functions $\mathbb{P}$ to construct projection layer i_p in SGNN. For this last step, a projection vector $\mathbb{P}^k$ is first constructed on-the-fly using a hash function with feature ids f_j in $\vec{x_i}$ and fixed seed as input, then dot product of the two vectors $< \vec{x_i}, \mathbb{P}^k >$ is computed and transformed into binary representation $\mathbb{P}^k(\vec{x_i})$ using $sgn(.)$ of the dot product.

As shown in Figure 1, both $\mathbb{F}$ and $\mathbb{P}$ steps are computed on-the-fly, i.e., no word-embedding or vocabulary/feature matrices need to be stored and looked up during training or inference. Instead feature-ids and projection vectors are dynamically computed via hash functions. For intermediate feature weights w_j, we use observed counts in

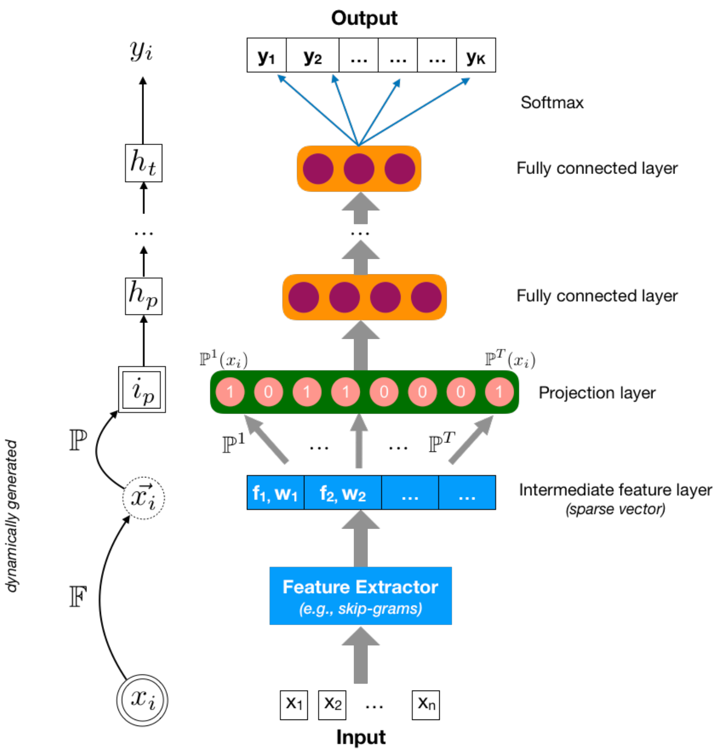

Figure 1: Self-Governing Neural Network (SGNN) architecture.

each input text and do not use pre-computed statistics like idf. Hence the method is embedding-free.

Model Optimization. The SGNN network is trained from scratch on the task data using a supervised loss defined wrt ground truth $\hat{y}_i$:

$$\mathcal{L}(.) = \sum_{i \in N} \texttt{cross}-\texttt{entropy}(y_i, \hat{y}_i) \quad (6)$$

During training, the network learns to choose and apply specific projection operations $\mathbb{P}^j$ (via activations) that are more predictive for a given task. The choice of the type of projection matrix $\mathbb{P}$ as well as representation of the projected space $\Omega_{\mathbb{P}}$ has a direct effect on computation cost and model size. We leverage an efficient randomized projection method and use a binary representation $\{0, 1\}^d$ for $\Omega_{\mathbb{P}}$. This yields a drastically lower memory footprint both in terms of number and size of parameters.

Computing Projections. We employ an efficient randomized projection method for the projection step. We use locality sensitive hashing (LSH) (Charikar, 2002) to model the underlying projection operations in SGNN. LSH is typically used as a dimensionality reduction technique for clustering (Manning et al., 2008). LSH allows us to project similar inputs $\vec{x}_i$ or intermediate network layers into hidden unit vectors that are nearby in metric space. We use repeated binary hashing for $\mathbb{P}$ and apply the projection vectors to transform the input $\vec{x}_i$ to a binary hash representation denoted by $\mathbb{P}_k(\vec{x}_i) \in \{0, 1\}$, where $[\mathbb{P}_k(\vec{x}_i)] := sgn[\langle \vec{x}_i, \mathbb{P}_k \rangle]$. This results in a d-bit vector representation, one bit corresponding to each projection row $\mathbb{P}_{k=1 \ldots d}$.

The same projection matrix $\mathbb{P}$ is used for training and inference. We never need to explicitly store the random projection vector $\mathbb{P}_k$ since we can compute them on the fly using hash functions over feature indices with a fixed row seed rather than invoking a random number generator. This also permits us to perform projection operations that are linear in the *observed* feature size rather than the *overall* feature or vocabulary size which can be prohibitively large for high-dimensional data, thereby saving both memory and computation cost. Thus, SGNN can efficiently model high-dimensional sparse inputs and large vocabulary sizes common for text applications instead of relying on feature pruning or other pre-processing heuristics employed to restrict input sizes in standard neural networks for feasible training. The binary representation is significant since this results in a significantly compact representation for the *projection* network parameters that in turn considerably reduces the model size.

SGNN Parameters. In practice, we employ T different projection functions $\mathbb{P}^{j=1\dots T}$, each resulting in d-bit vector that is concatenated to form the projected vector i_p in Equation 5. T and d vary depending on the *projection* network parameter configuration specified for $\mathbb{P}$ and can be tuned to trade-off between prediction quality and model size. Note that the choice of whether to use a single projection matrix of size $T \cdot d$ or T separate matrices of d columns depends on the type of projection employed (dense or sparse). For the intermediate feature step $\mathbb{F}$ in Equation 5, we use skip-gram features (3-grams with skip-size=2) extracted from raw text.

2.2 Training and Inference

We use the compact bit units to represent the *projection* in SGNN. During training, the network learns to move the gradients for points that are nearby to each other in the projected bit space $\Omega_{\mathbb{P}}$ in the same direction. SGNN network is trained end-to-end using backpropagation. Training can progress efficiently with stochastic gradient descent with distributed computing on high-performance CPUs or GPUs.

Complexity. The overall complexity for SGNN inference, governed by the projection layer, is $O(n \cdot T \cdot d)$, where n is the observed feature size (*not* overall vocabulary size) which is linear in input size, d is the number of LSH bits specified for each projection vector $\mathbb{P}_k$, and T is the number of projection functions used in $\mathbb{P}$. The model size (in terms of number of parameters) and memory storage required for the *projection* inference step is $O(T \cdot d \cdot C)$, where C is the number of hidden units in h_p in the multi-layer projection network and typically smaller than $T \cdot d$.

3 Datasets and Experimental Setup

3.1 Data Description

We conduct our experimental evaluation on two dialog act benchmark datasets.

- **SWDA:** Switchboard Dialog Act Corpus (Godfrey et al., 1992; Jurafsky et al., 1997) is a popular open domain dialogs corpus between two speakers with 42 dialogs acts.
- **MRDA:** ICSI Meeting Recorder Dialog Act Corpus (Adam et al., 2003; Shriberg et al., 2004) is a dialog corpus of multiparty meetings with 5 tags of dialog acts.

Datasets	Class	Vocab.	Train	Validation	Test
SwDA	42	20K	193K	23K	5K
MRDA	5	12K	78K	16K	15K

Table 1: Dialog Act Datasets Statistics

Table 1 summarizes dataset statistics. We use the train, validation and test splits as defined in (Lee and Dernoncourt, 2016; Ortega and Vu, 2017).

3.2 Experimental Setup

We setup our experimental evaluation, as follows: given a classification task and a dataset, we generate an on-device model. The size of the model can be configured (by adjusting the projection matrix $\mathbb{P}$) to fit in the memory footprint of the device, i.e. a phone has more memory compared to a smart watch. For each classification task, we report $Accuracy$ on the test set.

3.3 Hyperparameter and Training

For both datasets we used the following: 2-layer SGNN ($\mathbb{P}_{T=80,d=14}$ $\times$ FullyConnected$_{256}$ $\times$ FullyConnected$_{256}$), mini-batch size of 100, dropout rate of 0.25, learning rate was initialized to 0.025 with cosine annealing decay (Loshchilov and Hutter, 2016). Unlike prior approaches (Lee and Dernoncourt, 2016; Ortega and Vu, 2017) that rely on pre-trained word embeddings, we learn the projection weights on the fly during training, i.e word embeddings (or vocabularies) do not need to be stored. Instead, features are computed on the fly and are dynamically compressed via the projection matrices into projection vectors. These values were chosen via a grid search on development sets, we do not perform any other dataset-specific tuning. Training is performed through stochastic gradient descent over shuffled mini-batches with Nesterov momentum optimizer (Sutskever et al., 2013), run for 1M steps.

4 Results

Tables 2 and 3 show results on the SwDA and MRDA dialog act datasets. Overall, our SGNN model consistently outperforms the baselines and prior state-of-the-art deep learning models.

4.1 Baselines

We compare our model against a majority class baseline and Naive Bayes classifier (Lee and Dernoncourt, 2016). Our model significantly outperforms both baselines by 12 to 35% absolute.

4.2 Comparison against State-of-art Methods

We also compare our performance against prior work using HMMs (Stolcke et al., 2000) and recent deep learning methods like CNN (Lee and Dernoncourt, 2016), RNN (Khanpour et al., 2016) and RNN with gated attention (Tran et al., 2017).

To the best of our knowledge, (Lee and Dernoncourt, 2016; Ortega and Vu, 2017; Tran et al., 2017) are the latest approaches in dialog act classification, which also reported on the same data splits. Therefore, we compare our research against these works. According to (Ortega and Vu, 2017), prior work by (Ji and Bilmes, 2006) achieved promising results on the MRDA dataset, but since the evaluation was conducted on a different data split, it is hard to compare them directly.

For both SwDA and MRDA datasets, our SGNNs obtains the best result of 83.1 and 86.7 accuracy outperforming prior state-of-the-art work. This is very impressive given that we work with very small memory footprint and we do not rely on pre-trained word embeddings. Our study also shows that the proposed method is very effective for such natural language tasks compared to more complex neural network architectures such as deep CNN (Lee and Dernoncourt, 2016) and RNN variants (Khanpour et al., 2016; Ortega and Vu, 2017). We believe that the compression techniques like locality sensitive projections jointly coupled with non-linear functions are effective at capturing low-dimensional semantic text representations that are useful for text classification applications.

4.3 Discussion on Model Size and Inference

LSTMs have millions of parameters, while our on-device architecture has just 300K parameters (order of magnitude lower). Most deep learning methods also use large vocabulary size of 10K or higher. Each word embedding is represented as 100-dimensional vector leading to a storage requirement of $10,000 \times 100$ parameter weights just in the first layer of the deep network. In contrast, SGNNs in all our experiments use a fixed 1120-dimensional vector regardless of the vocabulary or feature size, dynamic computation results in further speed up for high-dimensional feature spaces. This amounts to a huge savings in storage and computation cost wrt FLOPs (floating point operations per second).

Method	Acc.
Majority Class (baseline) (Ortega and Vu, 2017)	33.7
Naive Bayes (baseline) (Khanpour et al., 2016)	47.3
HMM (Stolcke et al., 2000)	71.0
DRLM-conditional training (Ji and Bilmes, 2006)	77.0
DRLM-joint training (Ji and Bilmes, 2006)	74.0
LSTM (Lee and Dernoncourt, 2016)	69.9
CNN (Lee and Dernoncourt, 2016)	73.1
Gated-Attention&HMM (Tran et al., 2017)	74.2
RNN+Attention (Ortega and Vu, 2017)	73.8
RNN (Khanpour et al., 2016)	80.1
SGNN: Self-Governing Neural Network (ours)	**83.1**

Table 2: SwDA Dataset Results

Method	Acc.
Majority Class (baseline)(Ortega and Vu, 2017)	59.1
Naive Bayes (baseline) (Khanpour et al., 2016)	74.6
Graphical Model (Ji and Bilmes, 2006)	81.3
CNN (Lee and Dernoncourt, 2016)	84.6
RNN+Attention(Ortega and Vu, 2017)	84.3
RNN (Khanpour et al., 2016)	**86.8**
SGNN: Self-Governing Neural Network (ours)	**86.7**

Table 3: MRDA Dataset Results

5 Conclusion

We proposed Self-Governing Neural Networks for on-device short text classification. Experiments on multiple dialog act datasets showed that our model outperforms state-of-the-art deep leaning methods (Lee and Dernoncourt, 2016; Khanpour et al., 2016; Ortega and Vu, 2017). We introduced a compression technique that effectively captures low-dimensional semantic representation and produces compact models that significantly save on storage and computational cost. Our approach does not rely on pre-trained embeddings and efficiently computes the projection vectors on the fly. In the future, we are interested in extending this approach to more natural language tasks. For instance, we built a multilingual SGNN model for customer feedback classification (Liu et al., 2017) and obtained 73% on Japanese, close to best performing system on the challenge (Plank, 2017). Unlike their method, we did not use any pre-processing, tagging, parsing, pre-trained embeddings or other resources.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback and suggestions.

References

Janin Adam, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. 2003. The icsi meeting corpus. In *Proceedings of the 5TH SIGdial Workshop on Discourse and Dialogue*, pages 364–367.

Amr Ahmed, Sujith Ravi, Alex J. Smola, and Shravan M. Narayanamurthy. 2012. Fastex: Hash clustering with exponential families. In *Advances in Neural Information Processing Systems 25*, pages 2798–2806.

Thang D. Bui, Sujith Ravi, and Vivek Ramavajjala. 2017. Neural graph machines: Learning neural networks using graphs. *CoRR*, abs/1703.04818.

Thang D. Bui, Sujith Ravi, and Vivek Ramavajjala. 2018. Neural graph learning: Training neural networks using graphs. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 64–71.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, New York, NY, USA. ACM.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ICASSP'92, pages 517–520. IEEE Computer Society.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*.

Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360.

Gang Ji and Jeff Bilmes. 2006. Backoff model training using partially observed data: Application to dialog act tagging. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 280–287.

Daniel Jurafsky, Rebecca Bates, Rachel Martin Noah Coccaro, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Audreas Stolcke, Paul Taylor, and Van Ess-Dykema. 1997. Automatic detection of discourse structure for speech recognition and understanding. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 88–95.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 955–964.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2012–2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520.

Chao-Hong Liu, Yasufumi Moriya, Alberto Poncelas, and Declan Groves. 2017. Ijcnlp-2017 task 4: Customer feedback analysis. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 26–33, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Ilya Loshchilov and Frank Hutter. 2016. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Daniel Ortega and Ngoc Thang Vu. 2017. Neural-based context representation learning for dialog act classification. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 247–252.

Bo Pang and Sujith Ravi. 2012. Revisiting the predictability of language: Response completion in social media. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1489–1499.

Barbara Plank. 2017. All-in-1 at ijcnlp-2017 task 4: Short text classification with one model for all languages. In *Proceedings of the IJCNLP 2017, Shared*

Tasks, pages 143–148, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Sujith Ravi. 2013. Scalable decipherment for machine translation via hash sampling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 362–371. Association for Computational Linguistics.

Sujith Ravi. 2017. Projectionnet: Learning efficient on-device deep networks using neural projections. *CoRR*, abs/1708.00630.

Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The icsi meeting recorder dialog act (mrda) corpus. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*, pages 97–100, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Comput. Linguist.*, 26(3):339–373.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1139–III–1147.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112.

Quan Hung Tran, Gholamreza Haffari, and Ingrid Zukerman. 2017. A generative attentional neural network model for dialogue act classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 524–529.

Supervised Domain Enablement Attention for Personalized Domain Classification

Joo-Kyung Kim **Young-Bum Kim**
Amazon Alexa
`{jookyk,youngbum}@amazon.com`

Abstract

In large-scale domain classification for natural language understanding, leveraging each user's domain enablement information, which refers to the preferred or authenticated domains by the user, with attention mechanism has been shown to improve the overall domain classification performance. In this paper, we propose a supervised enablement attention mechanism, which utilizes sigmoid activation for the attention weighting so that the attention can be computed with more expressive power without the weight sum constraint of softmax attention. The attention weights are explicitly encouraged to be similar to the corresponding elements of the ground-truth's one-hot vector by supervised attention, and the attention information of the other enabled domains is leveraged through self-distillation. By evaluating on the actual utterances from a large-scale IPDA, we show that our approach significantly improves domain classification performance.

1 Introduction

Due to recent advances in deep learning techniques, intelligent personal digital assistants (IPDAs) such as Amazon Alexa, Google Assistant, Microsoft Cortana, and Apple Siri have been widely used as real-life applications of natural language understanding (Sarikaya et al., 2016; Sarikaya, 2017).

In natural language understanding, domain classification is a task that finds the most relevant domain given an input utterance (Tur and de Mori, 2011). For example, "make a lion sound" and "find me an apple pie recipe" should be classified as `ZooKeeper` and `AllRecipe`, respectively. Recent IPDAs cover more than several thousands of diverse domains by including third-party developed domains such as Alexa Skills (Kumar et al., 2017; Kim et al., 2018a; Kim and Kim, 2018), Google Actions, and Cortana Skills, which makes domain classification to be a more challenging task.

Given a large number of domains, leveraging user's enabled domain information[1] has been shown to improve the domain classification performance since enabled domains reflect the user's context in terms of domain usage (Kim et al., 2018b). For an input utterance, Kim et al. (2018b) use attention mechanism so that a weighted sum of the enabled domain vectors are used as an input signal as well as the utterance vector. The enabled domain vectors and the attention weights are automatically trained in an end-to-end fashion to be helpful for the domain classification.

In this paper, we propose a supervised enablement attention mechanism for more effective attention on the enabled domains. First, we use logistic sigmoid instead of softmax as the attention activation function to relax the constraint that the weight sum over all the enabled domains is 1 to the constraint that each attention weight is between 0 and 1 regardless of the other weights (Martins and Astudillo, 2016; Kim et al., 2017). Therefore, all the attention weights can be very low if there are no enabled domains relevant to a ground-truth so that we can disregard the irrelevant enabled domains, and multiple attention weights can have high values when multiple enabled domains are helpful for disambiguating an input utterance. Second, we encourage each attention weight to be high if the corresponding enabled domain is a ground-truth domain and if otherwise, to be low, by a supervised attention method (Mi et al., 2016) so that the attention weights can be directly tuned for the downstream classification task. Third, we

---

[1] Enabled domain information specifically refers to preferred or authenticated domains in Amazon Alexa, but it can be extended to other information such as the list of recently used domains.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 894–899
Brussels, Belgium, October 31 - November 4, 2018.

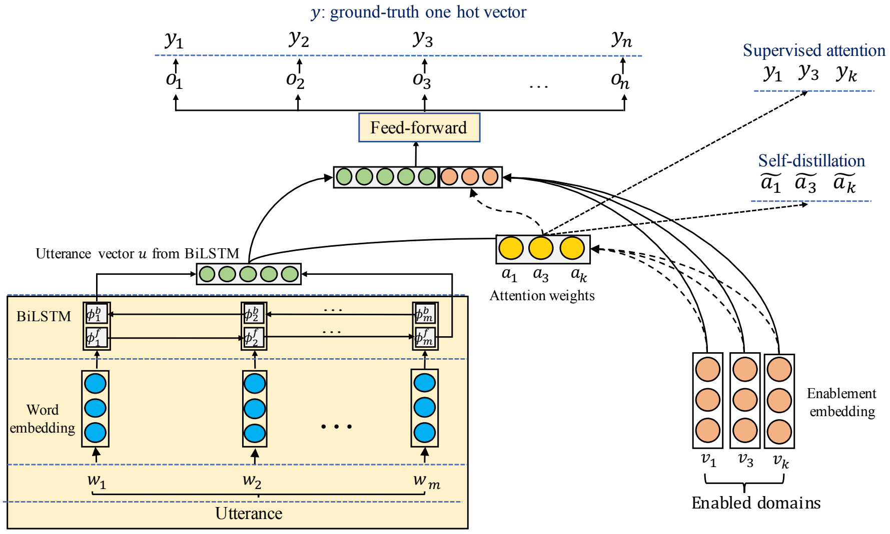

Figure 1: Model architecture: the input utterance is represented as a dense vector through word embedding and BiLSTM. Domain enablement vector is computed as a weighted sum of enabled domain vectors through the proposed attention mechanism. The two vectors are concatenated for the final domain prediction thorough a feed-forward neural network.

apply self-distillation (Furlanello et al., 2018) on top of the enablement attention weights so that we can better utilize the enabled domains that are not ground-truth domains but still relevant.

Evaluating on datasets obtained from real usage in a large-scale IPDA, we show that our approach significantly improves domain classification performance by utilizing the domain enablement information effectively.

2 Model

Figure 1 shows the overall architecture of the proposed model.

Given an input utterance, each word of the utterance is represented as a dense vector through word embedding followed by bidirectional long short-term memory (BiLSTM) (Graves and Schmidhuber, 2005). Then, an utterance vector is composed by concatenating the last outputs of the forward LSTM and the backward LSTM.[2]

To represent the domain enablement information, we obtain a weighted sum of domain enablement vector where the weights are calculated by logistic sigmoid function on top of the multiplicative attention (Luong et al., 2015) for the utterance vector and the domain enablement vectors. The attention weight of an enabled domain e is formulated as follows:

$$a_e = \sigma\left(u \cdot v_e\right),$$

where u is the utterance vector, v_e is the enablement vector of enabled domain e, and σ is sigmoid function. Compared to conventional attention mechanism using softmax function, which constraints the sum of the attention weights to be 1, sigmoid attention has more expressive power, where each attention weight can be between 0 and 1 regardless of the other weights. We show that using sigmoid attention is actually more effective for improving prediction performance in Section 3.

The utterance vector and the weighted sum of the domain enablement vectors are concatenated to represent the utterance and the domain enablement as a single vector. Given the concatenated vector, a feed-forward neural network with a single hidden layer[3] is used to predict the confidence score by logistic sigmoid function for each domain.

One issue of the proposed architecture is that the domain enablement can be trained to be a very strong signal, where one of the enabled domains would be the predicted domains regardless of the relevancy of the utterances to the predicted domains in many cases. To reduce this prediction bias, we use randomly sampled enabled domains

---

[2] We have also evaluated word vector summation, CNN (Kim, 2014), BiLSTM mean-pooling, and BiLSTM max-pooling (Conneau et al., 2017) as alternative utterance representation methods, but they did not show better performance on our task.

[3] We utilize scaled exponential linear units (SeLU) as the activation function for the hidden layer(Klambauer et al., 2017).

instead of the correct enabled domains of an input utterance with 50% probability during training so that the domain enablement is used as an auxiliary signal rather than determining signal. During inference, we always use the correct domain enablements of the given utterances.

The main loss function of our model is formulated as binary log loss between the confidence score and the ground-truth vector as follows:

$$\mathcal{L}_m = -\sum_{i=1}^{n} y_i \log o_i + (1 - y_i) \log (1 - o_i) ,$$

where n is the number of all domains, o is an n-dimensional confidence score vector from the model, and y is an n-dimensional one-hot vector whose element corresponding to the position of the ground-truth domain is set to 1.

2.1 Supervised Enablement Attention

Attention weights are originally intended to be automatically trained in an end-to-end fashion (Bahdanau et al., 2015), but it has been shown that applying proper explicit supervision to the attention improves the downstream tasks such as machine translation given the word alignment and constituent parsing given annotations between surface words and nonterminals (Mi et al., 2016; Liu et al., 2016; Kamigaito et al., 2017).

We hypothesize that if the ground-truth domain is one of the enabled domains, the attention weight for the ground-truth domain should be high and vice versa. To apply this hypothesis in the model training as a supervised attention method, we formulate an auxiliary loss function as follows:

$$\mathcal{L}_a = -\sum_{e \in E} y_e \log a_e + (1 - y_e) \log (1 - a_e) ,$$

where E is a set of enabled domains and a_e is the attention weight for the enabled domain e.

2.2 Self-Distilled Attention

One issue of supervised attention in Section 2.1 is that enabled domains that are not ground-truth domains are encouraged to have lower attention weights regardless of their relevancies to the input utterances and the ground-truth domains. Distillation methods utilize not only the ground-truth but also all the output activations of a source model so that all the prediction information from the source model can be utilized for more effective knowledge transfer between the source model and the target model (Hinton et al., 2014). Self-distillation, which trains a model leveraging the outputs of the source model with the same architecture or capacity, has been shown to improve the target model's performance with a distillation method (Furlanello et al., 2018).

We use a variant of self-distillation methods, where the model outputs at the previous epoch with the best dev set performance are used as the soft targets for the distillation,[4] so that the enabled domains that are not ground-truths can also be used for the supervised attention. While conventional distillation methods utilize softmax activations as the target values, we show that distillation on top of sigmoid activations is also effective without loss of generality. The loss function for the self-distillation on the attention weights is formulated as follows:

$$\mathcal{L}_d = -\sum_{e \in E} \tilde{a_e} \log a_e + (1 - \tilde{a_e}) \log (1 - a_e) ,$$

where $\tilde{a_e}$ is the attention weight of the model showing the dev set performance in the previous epochs. It is formulated as:

$$\tilde{a_e} = \sigma \left(\frac{u \cdot v_e}{T} \right) ,$$

where T is the temperature for sufficient usage of all the attention weights as the soft target. In this work, we set T to be 16, which shows the best dev set performance.

We have also evaluated soft-target regularization (Aghajanyan, 2017), where a weighted sum of the hard ground-truth target vector and the soft target vector is used as a single target vector, but it did not show better performance than self-distillation.

All the described loss functions are added to compose a single loss function as follows:

$$\mathcal{L} = \mathcal{L}_m + \alpha \left\{ (1 - \beta) \mathcal{L}_a + \beta^t \mathcal{L}_d \right\} ,$$

where α is a coefficient representing the degree of supervised enablement attention and β^t denotes the degree of the self-distillation. We set α to be 0.01 in this work. Following Hu et al. (2016), $\beta^t = 1 - 0.95^t$, where t denotes the current training epoch starting from 0 so that the hard ground-truth targets are more influential in the early epochs and the self-distillation is more utilized in the late epochs.

---

[4] This approach is closely related to Temporal Ensembling

Model no	Attention method	Biased ground-truth inclusion			Unbiased ground-truth inclusion		
		Top1	MRR	Top3	Top1	MRR	Top3
(1)	sfm	95.81	97.27	99.08	90.65	93.60	97.31
(2)	sgmd	95.98	97.43	99.19	91.03	93.92	97.49
(3)	sgmd, spvs	96.10	97.50	99.21	91.11	93.98	97.53
(4)	sgmd, spvs, sdst	96.29	97.65	99.32	**91.33**	**94.14**	**97.62**
(5)	sfm, bias	97.01	98.26	99.75	90.07	93.03	96.84
(6)	sgmd, spvs, sdst, bias	**97.48**	**98.51**	**99.76**	90.58	93.30	96.73

Table 1: Accuracies (%) on a test set with biased ground-truth inclusion in the enabled domains (90%) (left) and a test set with unbiased inclusion (70%) (right) with various enablement attention methods. **sftm**, **sgmd**, **spvs**, **sdst**, and **bias** denote softmax, sigmoid, supervised, self-distilled, and domain enablement bias, respectively.

Utterance	Ground-truth	Enabled domain: [attention weights for model (1), (2), and (4)], ...
what is the price of bitcoin	Crypto Price	Sleep and Relaxation Sounds: [0.9998, 0.0004, 0.2029], Crypto Price: [0.0001, 9.21e-0.6, 0.9977]
find me a round trip ticket flight	Expedia	Expedia: [0.0048, 5.37e-08, 0.6205], KAYAK: [0.9952, 0.0004, 0.461]
find my phone	Find My Phone	The Name Game: [1.0, 0.0001, 0.1677]

Table 2: Sample utterances correctly predicted with model (4) but not with model (1) and (2).

3 Experiments

We evaluate our proposed model on domain classification leveraging enabled domains. The enabled domains can be a crucial disambiguating signal especially when there are multiple similar domains. For example, assume that the input utterance is *"what's the weather"* and there are multiple weather-related domains such as `NewYorkWeather`, `AccuWeather`, and `WeatherChannel`. In this case, if `WeatherChannel` is included as an enabled domain of the current user, it is likely to be the most relevant domain to the user.

3.1 Datasets

Following the data collection methods used in Kim et al. (2018b), our models are trained using utterances with explicit invocation patterns. For example, given a user's utterance, *"Ask* {`ZooKeeper`} *to* {*play peacock sound*}*," "play peacock sound"* and `ZooKeeper` are extracted to compose a pair of the utterance and the ground-truth, respectively. In this way, we have generated train, development, and test sets containing 4.4M, 500K, and 500K utterances, respectively. All the utterances are from the usage log of Amazon Alexa and the ground-truth of each utterance is one of 1K frequently used domains. The average number of enabled domains per utterance in the test sets is 8.47.

One issue of this collected data sets is that the ground-truth is included in the enabled domains for more than 90% of the utterances, where the ground-truths are biased to enabled domains.[5] For more correct and unbiased evaluation of the models on the input utterances from real live traffic, we also evaluate the models on the same sized train, development, and test sets where the utterances are sampled to set the ratio of ground-truth inclusion in enabled domains to be 70%, which is closer to the ratio for actual input traffic.

3.2 Results

Table 1 shows the accuracies of our proposed models on the two test sets. We also show mean reciprocal rank (MRR) and top-3, accuracy[6] which is meaningful when utilizing post reranker, but we do not cover reranking issues in this paper (Robichaud et al., 2014; Kim et al., 2018a).

From Table 1, we can first see that changing softmax attention to sigmoid attention significantly improves the performance. This means that having more expressive power for the domain enablement information by relaxing the softmax constraint is effective in terms of leveraging the domain enablement information for domain classification. Along with sigmoid attention, supervised attention leveraging ground-truth slightly improves the performance, and supervised attention combined with self-distillation shows significant performance improvement. It demon-

---

(Laine and Aila, 2017), but we just leverage the model outputs at the previous epoch rather than accumulating the outputs over multiple epochs.

[5] Since the data collection method leverages utterances where users already know the exact domain names, such domains are likely to be the enabled domains of the users.

[6] Top-3 accuracy is calculated as # (utterances one of whose top three predictions is a ground-truth) / # (total utterances).

strates that supervised domain enablement attention leveraging ground-truth enabled domains is helpful, and utilizing attention information from other enabled domains is synergistic.

Kim et al. (2018b)'s model also adds a domain enablement bias vector to the final output, which is helpful when the ground-truth domain is one of the enabled domains. Such models (5) and (6) also show good performance for the test set where the ground-truth is one of the enabled domains with more than 90% probability. However, for the unbiased test set where the ground-truth is included in the enabled domains with a smaller probability, not adding the bias vector is shown to be better overall.

Table 2 shows sample utterances correctly predicted with model (4) but not with model (1) and (2). For the first two utterances, the ground-truths are included in the enabled domains, but there were only hundreds or fewer training instances whose ground-truths are `CryptoPrice` or `Expedia`. In these cases, we can see that model (1) attends to unrelated domains, model (2) attends to none of the enabled domains, but model (4), which uses supervised attention, is shown to attend to the ground-truth even without many training examples. "*find my phone*" has a single enabled domain which is not a ground-truth. In this case, model (1) still fully attends to the unrelated domain because of softmax attention while model (2) and (4) do not highly attend to it so that the unrelated enabled domain is not impactive.

3.3 Implementation Details

The word vectors are initialized with off-the-shelf GloVe vectors (Pennington et al., 2014), and all the other model parameters are initialized with Xavier initialization (Glorot and Bengio, 2010). Each model is trained for 25 epochs and the parameters showing the best performance on the development set are chosen as the model parameters. We use ADAM (Kingma and Ba, 2015) for the optimization with the initial learning rate 0.0002 and the mini-batch size 128. We use gradient clipping, where the threshold is set to 5. We use a variant of LSTM, where the input gate and the forget gate are coupled and peephole connections are used (Gers and Schmidhuber, 2000; Greff et al., 2017). We also use variational dropout for the LSTM regularization (Gal and Ghahramani, 2016). All the models are implemented with DyNet (Neubig et al., 2017).

4 Conclusion

We have introduced a novel domain enablement attention mechanism improving domain classification performance utilizing domain enablement information more effectively. The proposed attention mechanism uses sigmoid attentions for more expressive power of the attention weights, supervised attention leveraging ground-truth information for explicit guidance of the attention weight training, and self-distillation for the attention supervision leveraging enabled domains that are not ground truth domains. Evaluating on utterances from real usage in a large-scale IPDA, we have demonstrated that our proposed model significantly improves domain classification performance by better utilizing domain enablement information.

References

Armen Aghajanyan. 2017. Softtarget regularization: An effective technique to reduce over-fitting in neural networks. In *IEEE Conference on Cybernetics (CYBCONF)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *EMNLP*, pages 670–680.

Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born Again Neural Networks. In *International Conference on Machine Learning (ICML)*, pages 1602–1611.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 1019–1027.

Felix A. Gers and Jürgen Schmidhuber. 2000. Recurrent Nets that Time and Count. In *IJCNN*, volume 3, pages 189–194.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2017. LSTM: A search space odyssey. *Transactions on Neural Network Learning and Systems (TNNLS)*, 28(10):2222–2232.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing Deep Neural Networks with Logic Rules. In *ACL*, pages 2410–2420.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2017. Supervised attention for sequence-to-sequence constituency parsing. In *International Joint Conference on Natural Language Processing (IJCNLP)*.

Joo-Kyung Kim and Young-Bum Kim. 2018. Joint learning of domain classification and out-of-domain detection with dynamic class weighting for satisficing false acceptance rates. In *Interspeech*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1292–1302.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured Attention Networks. In *International Conference on Learning Representations (ICLR)*.

Young-Bum Kim, Dongchan Kim, Joo-Kyung Kim, and Ruhi Sarikaya. 2018a. A scalable neural shortlisting-reranking approach for large-scale domain classification in natural language understanding. In *NAACL*, pages 16–24.

Young-Bum Kim, Dongchan Kim, Anjishnu Kumar, and Ruhi Sarikaya. 2018b. Efficient Large-Scale Neural Domain Classification with Personalized Attention. In *ACL*, pages 2214–2224.

Diederik P. Kingma and Jimmy Lei Ba. 2015. ADAM: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 972–981.

Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Bjorn Hoffmeister, Markus Dreyer, Stanislav Peshterliev, Ankur Gandhe, Denis Filiminov, Ariya Rastrow, Christian Monson, and Agnika Kumar. 2017. Just ASK: Building an Architecture for Extensible Self-Service Spoken Language Understanding. In *NIPS Workshop on Conversational AI*.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations (ICLR)*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *COLING*, pages 3093––3102.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*, pages 1412–1421.

André F. T. Martins and Ramón Fernandez Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *International Conference on Machine Learning (ICML)*, pages 1614–1623.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised Attentions for Neural Machine Translation. In *EMNLP*, pages 2283–2288.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*, pages 1532–1543.

Jean-Philippe Robichaud, Paul A. Crook, Puyang Xu, Omar Zia Khan, and Ruhi Sarikaya. 2014. Hypotheses ranking for robust domain classification and tracking in dialogue systems. In *Interspeech*, pages 145–149.

Ruhi Sarikaya. 2017. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine*, 34(1):67–81.

Ruhi Sarikaya, Paul A Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, and Xiaohu Liu. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *Spoken Language Technology Workshop (SLT)*, page 391–397.

Gokhan Tur and Renato de Mori. 2011. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. New York, NY: John Wiley and Sons.

A Deep Neural Network Sentence Level Classification Method with Context Information

Xingyi Song and **Johann Petrak**
Department of Computer Science
University of Sheffield
Sheffield, UK
{x.song, johann.petrak}@sheffield.ac.uk

Angus Roberts
NIHR Biomedical Research Centre
Institute of Psychiatry, Psychology and Neuroscience
Kings College London
London, UK
angus.roberts@kcl.ac.uk

Abstract

In the sentence classification task, context formed from sentences adjacent to the sentence being classified can provide important information for classification. This context is, however, often ignored. Where methods do make use of context, only small amounts are considered, making it difficult to scale. We present a new method for sentence classification, Context-LSTM-CNN, that makes use of potentially large contexts. The method also utilizes long-range dependencies within the sentence being classified, using an LSTM, and short-span features, using a stacked CNN. Our experiments demonstrate that this approach consistently improves over previous methods on two different datasets.

1 Introduction

Artificial neural networks (ANN) and especially Deep Neural Networks (DNN) give state-of-the art results for sentence classification tasks. Usually, sentences are treated as separate instances for the task. However, in many situations the sentence that is the focus of classification appears in a context that can provide additional information. For example, in the below sentences from the IEMOCAP dataset, it is difficult to classify M02 as showing excitement, without the prior context:

- M01: I got it. I got accepted to U.S.C..
- F01: Oh, for real?
- M02: Yes! I just found out today. I just got the letter.

Our work is motivated by sentence classification in the text of medical records, in which complex judgements may be made across several sentences, each adding weight and nuance to a point. We believe, however, that the techniqe is more widely applicable. In order to test generalisability and to allow reproducibility, we therefore present an evaluation of the method with publicy available, non-medical corpora.

Previous work on using context for sentence classification used LSTM and CNN network layers to encode the surrounding context, giving an improvement in classification accuracy (Lee and Dernoncourt, 2016). However, the use of CNN and LSTM layers imposes a significant computational cost when training the network, especially if the size of the context is large. For this reason, the approach presented in (Lee and Dernoncourt, 2016) is explicitly intended for sequential, short-text classification.

In many cases, however, the context available is of significant size. We therefore introduce a new method, **Context-LSTM-CNN**[1], which is based on the computationally efficient FOFE (Fixed Size Ordinally Forgetting) method (Zhang et al., 2015), and an architecture that combines an LSTM and CNN for the focus sentence. The method consistently improves over results obtained from either LSTM alone, CNN alone, or these two combined, with little increase in training time.

This paper makes three contributions: 1) a demonstration of the importance of context in some sentence classification tasks; 2) an adaptation of existing datasets for such sentence classification tasks, in order to support reproducibility of evaluations; 3) a neural architecture for sentence classification that outperforms previous methods, and can include context of arbitrary size without incurring a large computational cost.

2 Related work

Since their introduction (Collobert et al., 2011), CNNs with word embedding language models have become common for text classification tasks (Kim, 2014; Conneau et al., 2017). One limitation of the original CNN approach is the loss

---

[1] The code is publicly available at https://github.com/deansong/contextLSTMCNN

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 900–904
Brussels, Belgium, October 31 - November 4, 2018.

of long distance dependencies. In order to deal with this in image and speech recognition tasks, Xu et al. (2015); Sainath et al. (2015) combined CNNs with a Recurrent Neural Network (RNN) layer. Zhou et al. (2015) subsequently applied this to text classification. However, the CNN-RNN approach was originally devised for sequence labelling, is biased towards later words in the sequence, and does not perform better than CNN alone. Huynh et al. (2016) suggested reversing the architecture to first apply the RNN followed by a CNN with pooling to obtain global features. This gave results that improved over CNN-RNN, but not over CNN alone. In this paper, we build on Huynh et al. (2016)'s approach by replacing the GRU-based RNN (Cho et al., 2014) with an LSTM (Hochreiter and Schmidhuber, 1997) and by using multiple kernel sizes and more features in the subsequent CNN layer.

Lee and Dernoncourt (2016) showed that when classifying short texts, accuracy can be boosted by adding a CNN or LSTM derived vector representation of the surrounding context. For long contexts (such as patient records which may include well over 100 sentences), however, this will incur a significant additional computational cost. In this paper, we therefore apply an adaptation of the FOFE encoding (Zhang et al., 2015) to encode context.

3 Model

The Context-LSTM-CNN model is shown in Figure 1. It is based on the following components:

1. Input layer using word embeddings to encode the words of the focus sentence.
2. Bi-directional LSTM applied to the word embeddings of the focus sentence.
3. CNN on the outputs of the LSTM.
4. FOFE applied to word embeddings of both left and right context.
5. A final output layer.

In brief, an LSTM layer is used to encode the focus sentence. This is followed by convolutional layers with small-size kernels and max-pooling to extract local features at specific points from the LSTM outputs. In addition to processing the focus sentence, we also encode the full left and right contexts using an adaptation of FOFE applied to our embeddings. This encodes any variable length context into a fixed length embedding, thus allowing us to include large contexts without rapidly increasing the computational cost. The output of the FOFE layers are then each passed through separate fully connected layers, before being concatenated and connected to output layer.

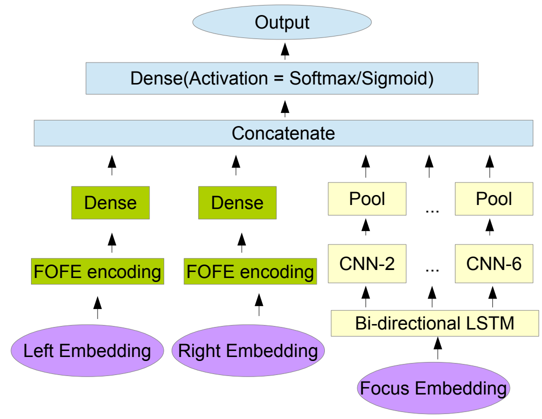

Figure 1: Structure of the C-LSTM-CNN model

In detail, the full network takes three inputs. The first is the sequence of words $X = (x_1, x_2, ...x_T)$, where T is the length of the sentence to be classified, and where each x_i is a word embedding for the respective word in this sentence. Embeddings are pre-trained by Word2Vec (Mikolov et al., 2013) on the corpus used for the respective experiment. The embeddings are not updated during the training of our network.

The second and third inputs are the left and right context, which will connect to the FOFE encoders. Each context is a sequence of sentences $X_C = (s_1, s_2, ...s_N)$, where each sentence is a sequence of word embeddings $s_n = (x_1, x_2, ...x_U)$ from the same embedding space as X.

The first component of the inputs, derived from the focus sentence, is processed by a bi-directional LSTM with one layer, in order to capture long-distance dependencies within the sentence. Since LSTMs impose a significant computational cost for very long sequences we only use this layer for the input representing the focus sentence, and not for the left and right contexts.

The LSTM generates outputs $h_{lstm} = (h_1, h_2..., h_T)$ which are passed on to the convolutional layer (CNN) in order to learn local features for different kernel sizes l from the history-aware outputs of the LSTM. For each of several kernel sizes, we generate f different features, to give CNN outputs $c^l_{cnn} = (c_1, c_2, \cdots, c_{T-l+1})$. For each CNN output c^l_{cnn}, we use max-overtime pooling to extract the most significant feature, and dropout to make the learned features more robust.

We use an adapted version of FOFE to provide information about the left and right contexts of the focus. Instead of the original 1 of k FOFE representation, we apply FOFE encoding to word2vec embeddings. This gives a weighted sum of the context word embeddings, with weights decreasing exponentially with distance from the focus.

The embedding z for a sentence $(x_1, x_2, ...x_U)$ is initialised to $z_1 = x_1$, and then calculated recursively for $u \in 2 \cdots U$ as $z_u = \alpha \cdot z_{u-1} + x_u$. The parameter α is the forgetting factor, which controls how fast the weights used for words farther away from the start of the sentence diminish. This method is fast and compactly encodes the words of a sentence in a single embedding vector.

For our use of FOFE, we encode all sentences in the document to left and right of the focus sentence, in two hierarchical steps. First we encode each context sentence into a FOFE embedding z^{sent}, with a slowly-decreasing α_{sent}. Following this, the left context FOFE encodings are themselves encoded into a single context embedding using a rapidly decreasing α_{cont}. This is calculated starting with $z_1^{cont} = z_1^{sent}$ and is calculated for $m \in 2 \cdots |C_{left}|$ as $z_m^{cont} = \alpha_{cont} \cdot z_{m-1}^{cont} + z_m^{sent}$. The right context FOFE encodings are encoded in the same way, starting with $z_{|C_{right}|}^{cont} = z_{|C_{right}|}^{sent}$ and recursively applying the same formula for $m \in |C_{right}| \cdots 2$. This gives a heavy bias towards sentences more local to the focus sentence, but only slightly decreases the importance of words within each sentence. The final FOFE embeddings for the left and right contexts are then put through a dense linear layer to obtain the hidden layer outputs, which are combined with the LSTM-CNN outputs. The concatenated outputs from the dense FOFE layers and from the CNN layer for all kernel sizes are then used as input to a final softmax output layer.

4 Experiments

We compare the performance of four different network architectures: 1) CNN only; 2) LSTM only; 3) LSTM-CNN; 4) LSTM context encoded LSTM-CNN (L-LSTM-CNN), in which the one left and right context sentence are encoded by LSTM; and 5) Context-LSTM-CNN (C-LSTM-CNN). We use the following two datasets for evaluation:

Interactive Emotional Dyadic Motion Capture Database (Busso et al., 2008)[2] (IEMOCAP). Originally created for the analysis of human emotions based on speech and video, a transcript of the speech component is available for NLP research. Each sentence in the dialogue is annotated with one of 10 types of emotion. There is a class imbalance in the labelled data, and so we follow the approach of (Chernykh et al., 2017), and only use sentences classified with one of four labels ('Anger', 'Excitement', 'Neutral' and 'Sadness'). For this dataset, instead of using left and right contexts, we assign all sentences from one person to one context and all sentences from the other person to the other context. While only the sentences with the four classes of interest are used for classification, all sentences of the dialog are used as the context. This results in a set of 4936 labelled sentences with average sentence length 14, and average document length is 986.

Drug-related Adverse Effects (Gurulingappa et al., 2012)[3] (ADE). This dataset contains sentences sampled from the abstracts of medical case reports. For each sentence, the annotation indicates whether adverse effects of a drug are being described ('Positive') or not ('Negative'). The original release of the data does not contain the document context, which we reconstructed from PubMed[4]. Sentences for which the full abstract could not be found were removed, resulting in 20,040 labelled sentences, with average sentence length 21 and average document length 129.

Model	IEMOCAP	ADE	time(s)
CNN only	58.16 (0.78)	89.49 (0.75)	218
LSTM only	56.30 (2.16)	89.04 (0.75)	648
LSTM-CNN	59.43 (1.60)	89.86 (1.06)	1239
L-LSTM-CNN	63.84 (2.03)	90.22 (0.75)	1800
C-LSTM-CNN	**71.49** (2.32)	**90.85** (0.37)	1243

Table 1: Average test accuracy and training time. Best values are marked as bold, standard deviations in parentheses

In all experiments, five-fold cross validation was used for evaluation (for comparison with (Huynh et al., 2016)). For each fold, 50 epochs were run for training using a minibatch size of 64 for each fold, and the Adamax optimization algo-

---

[2] `http://sail.usc.edu/iemocap/iemocap_release.htm`

[3] `https://sites.google.com/site/adecorpus/home/document`

[4] `https://www.ncbi.nlm.nih.gov/pubmed/`

F-measure	Anger (1,103)	Sadness (1,084)	Neutral (1,708)	Excitement (1,041)	Negative(14,854)	Positive(5,186)
CNN only	67.44 (1.02)	56.92 (3.25)	54.93 (3.70)	53.93 (2.50)	80.59 (1.08)	92.28 (0.59)
LSTM only	65.07 (2.49)	54.21 (4.03)	55.12 (2.95)	49.75 (1.80)	80.25 (1.23)	92.24 (0.54)
LSTM-CNN	67.74 (1.11)	55.86 (6.56)	57.17 (3.27)	56.95 (4.06)	81.55 (0.99)	93.00 (0.87)
L-LSTM-CNN	72.83 (1.81)	60.35 (4.65)	61.67 (3.18)	61.30 (2.64)	82.29 (0.80)	93.24 (0.60)
C-LSTM-CNN	**79.54**(1.70)	**66.07**(4.65)	**67.54**(4.72)	**73.11**(4.09)	**83.11**(0.24)	**93.72** (0.33)

Table 2: Average test F-measure for each class. Instance numbers in parentheses after class name. Best values are marked as bold, standard deviations in parentheses

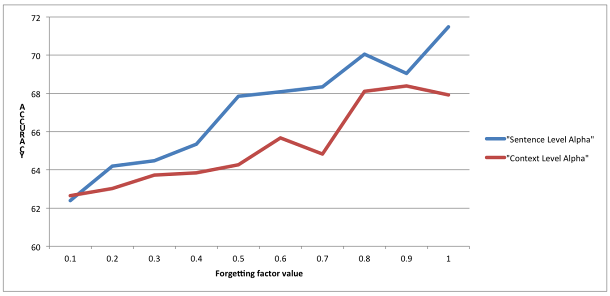

Figure 2: Context level (red line) and sentence level (blue line) forgetting factor test

rithm. To deal with label imbalance in the data, class weights w_i for class i were set proportional to $\max(f_i)/f_i$ where f_i is the frequency of class i.

We used word2vec embeddings with 50 dimensions (suggesed as sufficient by (Lai et al., 2016)). For the LSTM, 64 hidden units were used. For the CNN, layers for kernel sizes 2 to 6 were included in the network, and 64 features were used for each.

4.1 Effect of Forgetting Factors

We examined the effect of the two context encoder hyperparameters: α_{cont} (context level forgetting factor) and α_w (sentence level forgetting factor) on classification performance over the IEMOCAP dataset. We tested both in the range of 0.1 to 1 with an incremental step of 0.1. Results are shown in Figure 2. Accuracy improves as α_{cont} increases, but drops at $\alpha_{cont} = 1$, at which point all context sentence are given equal weight. This may be because context closest to the focus sentence is more important than distant context. Therefore, we select $\alpha_{cont} = 0.9$ in all experiments.

For α_{sent}, performance always increases as α_{sent} increases, with best results at $\alpha_{sent} = 1$, at which point all words in the sentence contribute equally in the context code. This implies that for individual sentences in the context, it is more preferable to lose word order, than to down weight any individual word. In all experiments, we therefore set the sentence level forgetting factor to $\alpha_{sent} = 1$

4.2 Evaluation Results

Table 1 shows the mean and standard deviations for accuracy over the cross validation folds, and training time, for both data sets. CNN alone performs better than LSTM alone in both tasks. The combined LSTM-CNN network consistently improves performance beyond both CNN alone and LSTM alone. Both context based models (L-LSTM-CNN and C-LSTM-CNN) perform better than non context based models, but note that L-LSTM-CNN increases training time by approximately 1.5x, whereas C-LSTM-CNN shows only a marginal increase in training time, with a large increase in accuracy on the IEMOCAP corpus.

Table 2 shows the F1-measure for each class in the two datasets. Again, Context-LSTM-CNN outperforms the other models on all classes for all data sets. C-LSTM-CNN improves on average by 6.28 over L-LSTM-CNN, 10.16 over LSTM-CNN, 11.4 over CNN and 13.29 over LSTM.

We conducted a t-test between L-LSTM-CNN and C-LSTM-CNN. On IEMOCAP, C-LSTM-CNN is significantly better than L-LSTM-CNN ($p = 0.002$). On ADE, C-LSTM-CNN is not significantly better than L-LSTM-CNN ($p = 0.128$). This may because ADE sentences are less context dependent. Alternatively, as the ADE task is relatively easy, with all models able to achieve about 90% accuracy, a context based approach might not be able to further improve the accuracy.

5 Conclusion

In this paper we introduced a new ANN model, **Context-LSTM-CNN**, that combines the strength of LSTM and CNN with the lightweight context encoding algorithm, FOFE. Our model shows a consistent improvement over either a non-context based model and a LSTM context encoded model, for the sentence classification task.

Acknowledgements

This work was partially supported by the European Union under grant agreement No. 654024 SoBigData.

References

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335.

Vladimir Chernykh, Grigoriy Sterling, and Pavel Prihodko. 2017. Emotion recognition from speech with recurrent neural networks. *arXiv preprint arXiv:1701.08071*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Dzmitry Bahdanau, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *The 2014 Conference on Empirical Methods in Natural Language Processing*.

Ronan Collobert, Jason Weston, Lon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*.

Alexis Conneau, Holger Schwenk, Loc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of biomedical informatics*, 45(5):885–892.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Trung Huynh, Yulan He, Allistair Willis, and Stefan Rueger. 2016. Adverse drug reaction classification with deep neural networks. In *Proceedings of The 26th International Conference on Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *The 2014 Conference on Empirical Methods in Natural Language Processing*.

S. Lai, K. Liu, S. He, and J. Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Tara Sainath, Oriol Vinyals, Andrew Senior, and Hasim Sak. 2015. Convolutional, long short-term memory, fully connected deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *Proceedings of 2015th International Conference on Machine Learning*.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. 2015. A c-lstm neural network for text classification. *arXiv151108630Z*.

Towards Dynamic Computation Graphs via Sparse Latent Structure

Vlad Niculae[♮] **André F. T. Martins**[♮♭] and **Claire Cardie**[♯]
[♮]Instituto de Telecomunicações / [♭]Unbabel, Lisbon, Portugal
[♯]Cornell University, Ithaca, NY, USA
`vlad@vene.ro`, `andre.martins@unbabel.com`, `cardie@cs.cornell.edu`.

Abstract

Deep NLP models benefit from underlying structures in the data—*e.g.*, parse trees—typically extracted using off-the-shelf parsers. Recent attempts to jointly learn the latent structure encounter a tradeoff: either make factorization assumptions that limit expressiveness, or sacrifice end-to-end differentiability. Using the recently proposed SparseMAP inference, which retrieves a sparse distribution over latent structures, we propose a novel approach for end-to-end learning of latent structure predictors jointly with a downstream predictor. To the best of our knowledge, our method is the first to enable unrestricted dynamic computation graph construction from the *global* latent structure, while maintaining differentiability.

1 Introduction

Latent structure models are a powerful tool for modeling compositional data and building NLP pipelines (Smith, 2011). An interesting emerging direction is to **dynamically** adapt a network's computation graph, based on structure inferred from the input; notable applications include learning to write programs (Bosnjak et al., 2017), answering visual questions by composing specialized modules (Hu et al., 2017; Johnson et al., 2017), and composing sentence representations using latent syntactic parse trees (Yogatama et al., 2017).

But how to learn a model that is able to condition on such combinatorial variables? The question then becomes: how to marginalize over *all* possible latent structures? For tractability, existing approaches have to make a choice. Some of them eschew *global* latent structure, resorting to computation graphs built from smaller local decisions: *e.g.*, structured attention networks use local posterior marginals as attention weights (Kim et al., 2017; Liu and Lapata, 2018), and Maillard et al. (2017) construct sentence representations from parser chart entries. Others allow more flexibility at the cost of losing end-to-end differentiability, ending up with reinforcement learning problems (Yogatama et al., 2017; Hu et al., 2017; Johnson et al., 2017; Williams et al., 2018). More traditional approaches employ an off-line structure predictor (*e.g.*, a parser) to define the computation graph (Tai et al., 2015; Chen et al., 2017), sometimes with some parameter sharing (Bowman et al., 2016). However, these off-line methods are unable to *jointly* train the latent model and the downstream classifier via error gradient information.

We propose here a new strategy for building **dynamic computation graphs** with latent structure, through **sparse structure prediction**. Sparsity allows selecting and conditioning on a tractable number of global structures, eliminating the limitations stated above. Namely, our approach is the first that:

A) is **fully differentiable**;

B) supports **latent structured variables**;

C) can marginalize over full **global structures**.

This contrasts with off-line and with reinforcement learning-based approaches, which satisfy **B** and **C** but not **A**; and with local marginal-based methods such as structured attention networks, which satisfy **A** and **B**, but not **C**. Key to our approach is the recently proposed **SparseMAP inference** (Niculae et al., 2018), which induces, for each data example, a very sparse posterior distribution over the possible structures, allowing us to compute the expected network output efficiently and explicitly in terms of a small, interpretable set of latent structures. Our model can be trained end-to-end with gradient-based methods, without the need for policy exploration or sampling.

We demonstrate our strategy on inducing latent dependency TreeLSTMs, achieving competitive results on sentence classification, natural language inference, and reverse dictionary lookup.

2 Sparse Latent Structure Prediction

We describe our proposed approach for learning with combinatorial structures (in particular, non-projective dependency trees) as latent variables.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 905–911
Brussels, Belgium, October 31 - November 4, 2018.

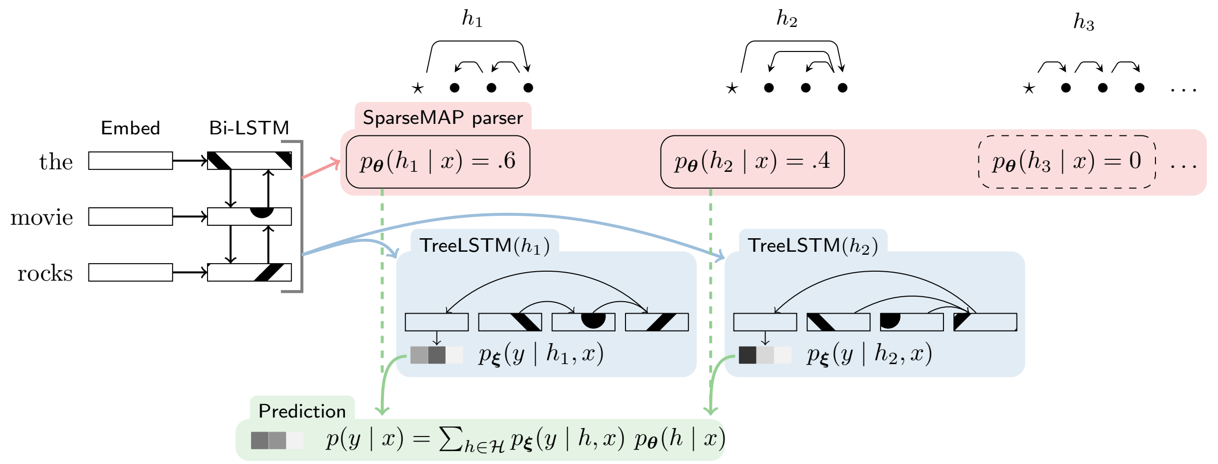

Figure 1: Our method computes a sparse probability distribution over all possible latent structures: here, only two have nonzero probability. For each selected tree h, we evaluate $p_{\boldsymbol\xi}(y \mid h, x)$ by dynamically building the corresponding computation graph (*e.g.*, a TreeLSTM). The final, posterior prediction is a sparse weighted average.

2.1 Latent Structure Models

Let x and y denote classifier inputs and outputs, and $h \in \mathcal{H}(x)$ a latent variable; for example, $\mathcal{H}(x)$ can be the set of possible dependency trees for x. We would like to train a neural network to model

$$p(y \mid x) := \sum_{h \in \mathcal{H}(x)} p_{\boldsymbol\theta}(h \mid x)\, p_{\boldsymbol\xi}(y \mid h, x), \qquad (1)$$

where $p_{\boldsymbol\theta}(h \mid x)$ is a structured-output parsing model that defines a distribution over trees, and $p_{\boldsymbol\xi}(y \mid h, x)$ is a classifier whose computation graph may depend **freely and globally** on the structure h (*e.g.*, a TreeLSTM). The rest of this section focuses on the challenge of defining $p_{\boldsymbol\theta}(h \mid x)$ such that Eqn. 1 remains tractable and differentiable.

2.2 Global Inference

Denote by $f_{\boldsymbol\theta}(h;\, x)$ a scoring function, assigning each tree a non-normalized score. For instance, we may have an *arc-factored* score $f_{\boldsymbol\theta}(h; x) := \sum_{a \in h} s_{\boldsymbol\theta}(a; x)$, where we interpret a tree h as a set of directed arcs a, each receiving an atomic score $s_{\boldsymbol\theta}(a; x)$. Deriving $p_{\boldsymbol\theta}$ given $f_{\boldsymbol\theta}$ is known as *structured inference*. This can be written as a Ω-regularized optimization problem of the form

$$p_{\boldsymbol\theta}(\cdot \mid x) := \operatorname*{argmax}_{q \in \triangle^{|\mathcal{H}(x)|}} \sum_{h \in \mathcal{H}(x)} q(h) f_{\boldsymbol\theta}(h; x) - \Omega(q),$$

where $\triangle^{|\mathcal{H}(x)|}$ is the set of all possible probability distributions over $\mathcal{H}(x)$. Examples follow.

Marginal inference. With negative entropy regularization, *i.e.*, $\Omega(q) := \sum_{h \in \mathcal{H}(x)} q(h) \log q(h)$, we recover marginal inference, and the probability of a tree becomes (Wainwright and Jordan, 2008)

$$p_{\boldsymbol\theta}(h \mid x) \propto \exp(f_{\boldsymbol\theta}(h; x)).$$

This closed-form derivation, detailed in Appendix A, provides a differentiable expression for $p_{\boldsymbol\theta}$. However, crucially, since $\exp(\cdot) > 0$, every tree is assigned strictly nonzero probability. Therefore—unless the downstream $p_{\boldsymbol\xi}$ is constrained to also factor over arcs, as in Kim et al. (2017); Liu and Lapata (2018)—the sum in Eqn. 1 requires enumerating the exponentially large $\mathcal{H}(x)$. This is generally intractable, and even hard to approximate via sampling, even when $p_{\boldsymbol\theta}$ is tractable.

MAP inference. At the polar opposite, setting $\Omega(q) := 0$ yields **maximum a posteriori (MAP) inference** (see Appendix A). MAP assigns a probability of 1 to the highest-scoring tree, and 0 to all others, yielding a very sparse $p_{\boldsymbol\theta}$. However, since the top-scoring tree (or top-k, for fixed k) does not vary with small changes in $\boldsymbol\theta$, error gradients cannot propagate through MAP. This prevents end-to-end gradient-based training for MAP-based latent variables, which makes them more difficult to use. Related reinforcement learning approaches also yield only one structure, but sidestep non-differentiability by instead introducing more challenging search problems.

2.3 Sparse Inference

In this work, we propose using **SparseMAP inference** (Niculae et al., 2018) to sparsify the set $\mathcal{H}$ while preserving differentiability. SparseMAP uses a quadratic penalty on the posterior marginals

$$\Omega(q) := \|\boldsymbol{u}(q)\|_2^2, \quad \text{where} \quad [\boldsymbol{u}(q)]_a := \sum_{h:a \in h} q(h).$$

Situated between marginal inference and MAP inference, SparseMAP assigns nonzero probability to only a small set of plausible trees $\bar{\mathcal{H}} \subset \mathcal{H}$, of size at most equal to the number of arcs (Martins et al., 2015, Proposition 11). This guarantees that the summation in Eqn. 1 can be computed efficiently by iterating over $\bar{\mathcal{H}}$: this is depicted in Figure 1 and described in the next paragraphs.

Forward pass. To compute $p(y \mid x)$ (Eqn. 1), we observe that the SparseMAP posterior $p_{\boldsymbol{\theta}}$ is nonzero only on a small set of trees $\bar{\mathcal{H}}$, and thus we only need to compute $p_{\boldsymbol{\xi}}(y \mid h, x)$ for $h \in \bar{\mathcal{H}}$. The support and values of $p_{\boldsymbol{\theta}}$ are obtained by solving the SparseMAP inference problem, as we describe in Niculae et al. (2018). The strategy, based on the active set algorithm (Nocedal and Wright, 1999, chapter 16), involves a sequence of MAP calls (here: maximum spanning tree problems.)

Backward pass. We next show how to compute end-to-end gradients efficiently. Recall from Eqn. 1 $p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\theta}}(h \mid x)\, p_{\boldsymbol{\xi}}(y \mid h, x)$, where h is a discrete index of a tree. To train the classifier, we have $\partial p(y|x)/\partial \boldsymbol{\xi} = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\theta}}(h \mid x) \partial p_{\boldsymbol{\xi}}(y|h,x)/\partial \boldsymbol{\xi}$, therefore only the terms with nonzero probability (*i.e.*, $h \in \bar{\mathcal{H}}$) contribute to the gradient. $\partial p_{\boldsymbol{\xi}}(y|h,x)/\partial \boldsymbol{\xi}$ is readily available by implementing $p_{\boldsymbol{\xi}}$ in an automatic differentiation library.[1] To train the latent parser, the total gradient $\partial p(y|x)/\boldsymbol{\theta}$ is the sum $\sum_{h \in \bar{\mathcal{H}}} p_{\boldsymbol{\xi}}(y \mid h, x)\ \partial p_{\boldsymbol{\theta}}(h|x)/\partial \boldsymbol{\theta}$. We derive the expression of $\partial p_{\boldsymbol{\theta}}(h|x)/\partial \boldsymbol{\theta}$ in Appendix B. **Crucially, the gradient sum is also sparse, like $p_{\boldsymbol{\theta}}$, and efficient to compute,** amounting to multiplying by a $|\bar{\mathcal{H}}(x)|$-by-$|\bar{\mathcal{H}}(x)|$ matrix. The proof, given in Appendix B, is a novel extension of the SparseMAP backward pass (Niculae et al., 2018).

Generality. Our description focuses on probabilistic classifiers, but our method can be readily applied to networks that output any representation, not necessarily a probability. For this, we define a function $\boldsymbol{r}_{\boldsymbol{\xi}}(h, x)$, consisting of any auto-differentiable computation w.r.t. x, conditioned on the discrete latent structure h in arbitrary, non-differentiable ways. We then compute

$$\bar{\boldsymbol{r}}(x) := \sum_{h \in \mathcal{H}(x)} p_{\boldsymbol{\theta}}(h \mid x) \boldsymbol{r}_{\boldsymbol{\xi}}(h, x) = \mathbb{E}_{h \sim p_{\boldsymbol{\theta}}} \boldsymbol{r}_{\boldsymbol{\xi}}(h, x).$$

	subj.	SST	SNLI
left-to-right	**92.71**	82.10	80.98
flat	92.56	83.96	81.74
off-line	92.15	83.25	81.37
latent	92.25	**84.73**	**81.87**

Table 1: Accuracy scores for classification and NLI.

This strategy is demonstrated in our reverse-dictionary experiments in §3.4. In addition, our approach is not limited to trees: any structured model with tractable MAP inference may be used.

3 Experiments

We evaluate our approach on three natural language processing tasks: sentence classification, natural language inference, and reverse dictionary lookup.

3.1 Common aspects

Word vectors. Unless otherwise mentioned, we initialize with 300-dimensional GloVe word embeddings (Pennington et al., 2014) We transform every sentence via a bidirectional LSTM encoder, to produce a context-aware vector $\boldsymbol{v}_i$ encoding word i.

Dependency TreeLSTM. We combine the word vectors $\boldsymbol{v}_i$ in a sentence into a single vector using a tree-structured Child-Sum LSTM, which allows an arbitrary number of children at any node (Tai et al., 2015). Our baselines consist in extreme cases of dependency trees: where the parent of word i is word $i+1$ (resulting in a **left-to-right** sequential LSTM), and where all words are direct children of the root node (resulting in a **flat** additive model). We also consider **off-line** dependency trees precomputed by Stanford CoreNLP (Manning et al., 2014).

Neural arc-factored dependency parsing. We compute arc scores $s_{\boldsymbol{\theta}}(a; x)$ with one-hidden-layer perceptrons (Kiperwasser and Goldberg, 2016).

Experimental setup. All networks are trained via stochastic gradient with 16 samples per batch. We tune the learning rate on a log-grid, using a decay factor of 0.9 after every epoch at which the validation performance is not the best seen, and stop after five epochs without improvement. At test time, we scale the arc scores $s_{\boldsymbol{\theta}}$ by a temperature t

---

[1] Here we assume $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$ to be disjoint, but weight sharing is easily handled by automatic differentiation via the product rule. Differentiation w.r.t. the summation index h is not necessary: $p_{\boldsymbol{\xi}}$ may use the discrete structure h freely and globally.

	seen			unseen			concepts		
	rank	acc^{10}	acc^{100}	rank	acc^{10}	acc^{100}	rank	acc^{10}	acc^{100}
left-to-right	17	42.6	73.8	43	33.2	61.8	28	35.9	66.7
flat	18	45.1	71.1	**31**	**38.2**	**65.6**	29	34.3	68.2
latent	**12**	**47.5**	**74.6**	40	35.6	60.1	**20**	**38.4**	**70.7**
Maillard et al. (2017)	58	30.9	56.1	40	33.4	57.1	40	57.1	62.6
Hill et al. (2016)	12	48	28	22	41	70	69	28	54

Table 2: Results on the reverse dictionary lookup task (Hill et al., 2016). Following the authors, for an input definition, we rank a shortlist of approximately 50k candidate words according to the cosine similarity to the output vector, and report median rank of the expected word, accuracy at 10, and at 100.

chosen on the validation set, controlling the sparsity of the SparseMAP distribution. All hidden layers are 300-dimensional.[2]

3.2 Sentence classification

We evaluate our models for sentence-level subjectivity classification (Pang and Lee, 2004) and for binary sentiment classification on the Stanford Sentiment Treebank (Socher et al., 2013). In both cases, we use a softmax output layer on top of the Dependency TreeLSTM output representation.

3.3 Natural language inference (NLI)

We apply our strategy to the SNLI corpus (Bowman et al., 2015), which consists of classifying premise-hypothesis sentence pairs into entailment, contradiction or neutral relations. In this case, for each pair (x_P, x_H), the running sum is over *two* latent distributions over parse trees, *i.e.*, $\sum_{h_P \in \mathcal{H}(x_P)} \sum_{h_H \in \mathcal{H}(x_H)} p_{\boldsymbol{\xi}}(y \mid x_{\{P,H\}}, h_{\{P,H\}}) p_{\boldsymbol{\theta}}(h_P \mid x_P) p_{\boldsymbol{\theta}}(h_H \mid x_H)$. For each pair of trees, we independently encode the premise and hypothesis using a TreeLSTM. We then concatenate the two vectors, their difference, and their element-wise product (Mou et al., 2016). The result is passed through one *tanh* hidden layer, followed by the softmax output layer.[3]

3.4 Reverse dictionary lookup

The reverse dictionary task aims to compose a dictionary definition into an embedding that is close to the defined word. We therefore used *fixed* input and output embeddings, set to unit-norm 500-dimensional vectors provided, together with training and evaluation data, by Hill et al. (2016). The network output is a projection of the TreeLSTM encoding back to the dimension of the word embeddings, normalized to unit ℓ_2 norm. We maximize the cosine similarity of the predicted vector with the embedding of the defined word.

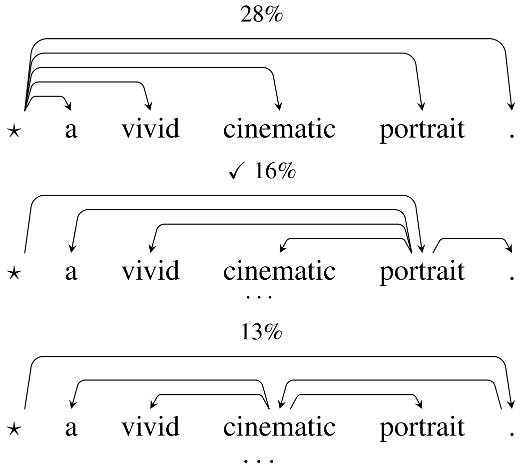

Figure 2: Three of the sixteen trees with nonzero probability for an SST test example. Flat representations, such as the first tree, perform well on this task, as reflected by the baselines. The second tree, marked with ✓, agrees with the off-line parser.

4 Discussion

Experimental performance. Classification and NLI results are reported in Table 1. Compared to the latent structure model of Yogatama et al. (2017), our model performs better on SNLI (80.5%) but worse on SST (86.5%). On SNLI, our model also outperforms Maillard et al. (2017) (81.6%). To our knowledge, latent structure models have not been tested on subjectivity classification. Surprisingly, the simple flat and left-to-right baselines are very strong, outperforming the off-line dependency tree models on all three datasets. The latent TreeLSTM model reaches the best accuracy on two out of the three datasets. On reverse dictionary lookup (Ta-

---

[2] Our `dynet` (Neubig et al., 2017) implementation is available at `https://github.com/vene/sparsemap`.

[3] For NLI, our architecture is motivated by our goal of evaluating the impact of latent structure for learning compositional sentence representations. State-of-the-art models conditionally transform the sentences to achieve better performance, *e.g.*, 88.6% accuracy in Chen et al. (2017).

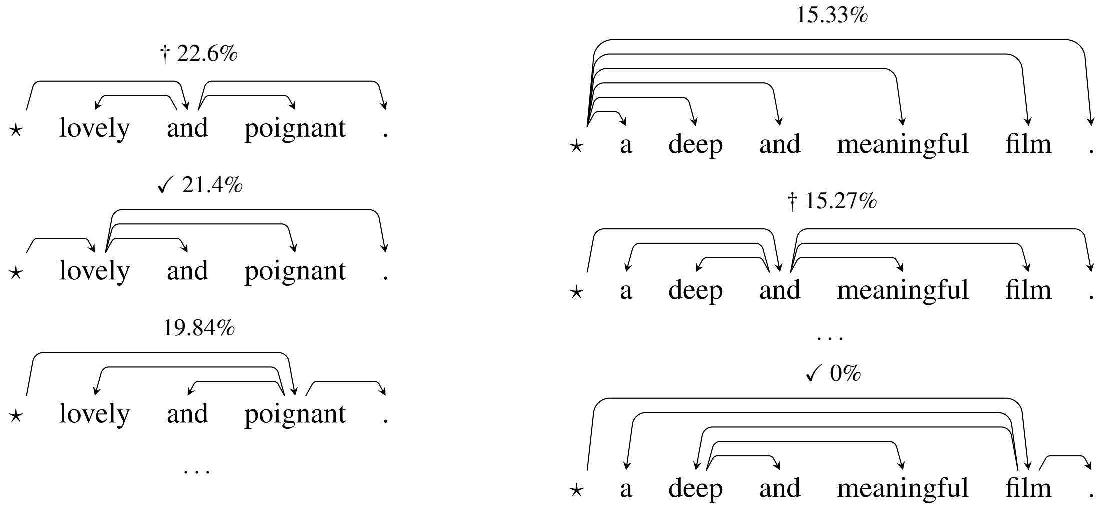

Figure 3: Examples of coordinate structures where our model assigns high probability to a symmetric parse (marked †). While not consistent with the standard asymmetrical parse produced by CoreNLP (marked with ✓), the symmetric analysis may be more appropriate for TreeLSTM composition.

ble 2), our model also performs well, especially on concept classification, where the input definitions are more different from the ones seen during training. For context, we repeat the scores of the CKY-based latent TreeLSTM model of Maillard et al. (2017), as well as of the LSTM from Hill et al. (2016); these different-sized models are not entirely comparable. We attribute our model's performance to the latent parser's flexibility, investigated below.

Selected latent structures. We analyze the latent structures selected by our model on SST, where the flat composition baseline is remarkably strong. We find that our model, to maximize accuracy, prefers flat or nearly-flat trees, but not exclusively: the average posterior probability of the flat tree is 28.9%. In Figure 2, the highest-ranked tree is flat, but deeper trees are also selected, including the projective CoreNLP parser output. Syntax is not necessarily an optimal composition order for a latent TreeLSTM, as illustrated by the poor performance of the off-line parser (Table 1). Consequently, our (fully unsupervised) latent structures tend to disagree with CoreNLP: the average probability of CoreNLP arcs is 5.8%; Williams et al. (2018) make related observations. Indeed, some syntactic conventions may be questionable for recursive composition. Figure 3 shows two examples where our model identifies a plausible symmetric composition order for coordinate structures: this analysis disagrees with CoreNLP, which uses the asymmetrical Stanford / UD convention of assigning the left-most conjunct as head (Nivre et al., 2016). Assigning the conjunction as head instead seems preferable in a Child-Sum TreeLSTM.

Training efficiency. Our model must evaluate at least one TreeLSTM for each sentence, making it necessarily slower than the baselines, which evaluate exactly one. Thanks to sparsity and autobatching, the actual slow-down is not problematic; moreover, as the model trains, the latent parser gets more confident, and for many unambiguous sentences there may be only one latent tree with nonzero probability. On SST, our average training epoch is only 4.7× slower than the off-line parser and 6× slower than the flat baseline.

5 Conclusions and future work

We presented a novel approach for training latent structure neural models, based on the key idea of sparsifying the set of possible structures, and demonstrated our method with competitive latent dependency TreeLSTM models. Our method's generality opens up several avenues for future work: since it supports any structure for which MAP inference is available (*e.g.*, matchings, alignments), and we have no restrictions on the downstream $p_{\boldsymbol{\xi}}(y \mid h, x)$, we may design latent versions of more complicated state-of-the-art models, such as ESIM for NLI (Chen et al., 2017). In concurrent work, Peng et al. (2018) proposed an approximate MAP backward pass, relying on a relaxation and a gradient projection. Unlike our method, theirs does not support multiple latent structures; we intend to further study the relationship between the methods.

Acknowledgments

This work was supported by the European Research Council (ERC StG DeepSPIN 758969) and by the Fundação para a Ciência e Tecnologia through contract UID/EEA/50008/2013. We thank Annabelle Carrell, Chris Dyer, Jack Hessel, Tim Vieira, Justine Zhang, Sydney Zink, and the anonymous reviewers, for helpful and well-structured feedback.

References

Matko Bosnjak, Tim Rocktäschel, Jason Naradowsky, and Sebastian Riedel. 2017. Programming with a differentiable Forth interpreter. In *Proc. ICML*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proc. EMNLP*.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proc. ACL*.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge University Press.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proc. ACL*.

George B Dantzig, Alex Orden, Philip Wolfe, et al. 1955. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *TACL*, 4(1):17–30.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proc. ICCV*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Inferring and executing programs for visual reasoning. In *Proc. ICCV*.

Yoon Kim, Carl Denton, Loung Hoang, and Alexander M Rush. 2017. Structured attention networks. In *Proc. ICLR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *TACL*, 6:63–75.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised tree-LSTMs. *preprint arXiv:1705.09189*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. ACL (demonstrations)*.

André FT Martins, Mário AT Figueiredo, Pedro MQ Aguiar, Noah A Smith, and Eric P Xing. 2015. AD3: Alternating directions dual decomposition for MAP inference in graphical models. *JMLR*, 16(1):495–545.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications? In *Proc. EMNLP*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *preprint arXiv:1701.03980*.

Vlad Niculae, André FT Martins, Mathieu Blondel, and Claire Cardie. 2018. SparseMAP: Differentiable sparse structured inference. In *Proc. ICML*.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proc. LREC*.

Jorge Nocedal and Stephen Wright. 1999. *Numerical optimization*. Springer New York.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity. In *Proc. ACL*.

Hao Peng, Sam Thomson, and Noah A Smith. 2018. Backpropagating through structured argmax using a SPIGOT. In *Proc. ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proc. EMNLP*.

Noah A Smith. 2011. Linguistic structure prediction. *Synth. Lect. Human Lang. Technol.*, 4(2):1–274.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured Long Short-Term Memory networks. In *Proc. ACL-IJCNLP*.

Martin J Wainwright and Michael I Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Adina Williams, Andrew Drozdov, and Samuel R Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *TACL*, 6:253–267.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *Proc. ICLR*.

Convolutional Neural Networks with Recurrent Neural Filters

Yi Yang
Bloomberg
New York, NY 10022
`yyang464@bloomberg.net`

Abstract

We introduce a class of convolutional neural networks (CNNs) that utilize recurrent neural networks (RNNs) as convolution filters. A convolution filter is typically implemented as a linear affine transformation followed by a non-linear function, which fails to account for language compositionality. As a result, it limits the use of high-order filters that are often warranted for natural language processing tasks. In this work, we model convolution filters with RNNs that naturally capture compositionality and long-term dependencies in language. We show that simple CNN architectures equipped with recurrent neural filters (RNFs) achieve results that are on par with the best published ones on the Stanford Sentiment Treebank and two answer sentence selection datasets.[1]

1 Introduction

Convolutional neural networks (CNNs) have been shown to achieve state-of-the-art results on various natural language processing (NLP) tasks, such as sentence classification (Kim, 2014), question answering (Dong et al., 2015), and machine translation (Gehring et al., 2017). In an NLP system, a convolution operation is typically a sliding window function that applies a convolution filter to every possible window of words in a sentence. Hence, the key components of CNNs are a set of convolution filters that compose low-level word features into higher-level representations.

Convolution filters are usually realized as linear systems, as their outputs are affine transformations of the inputs followed by some non-linear activation functions. Prior work directly adopts a linear convolution filter to NLP problems by utilizing a concatenation of embeddings of a window of words as the input, which leverages word order information in a shallow additive way. As early CNN architectures have mainly drawn inspiration from models of the primate visual system, the necessity of coping with language compositionality is largely overlooked in these systems. Due to the linear nature of the convolution filters, they lack the ability to capture complex language phenomena, such as compositionality and long-term dependencies.

To overcome this, we propose to employ recurrent neural networks (RNNs) as convolution filters of CNN systems for various NLP tasks. Our recurrent neural filters (RNFs) can naturally deal with language compositionality with a recurrent function that models word relations, and they are also able to implicitly model long-term dependencies. RNFs are typically applied to word sequences of moderate lengths, which alleviates some well-known drawbacks of RNNs, including their vulnerability to the gradient vanishing and exploding problems (Pascanu et al., 2013). As in conventional CNNs, the computation of the convolution operation with RNFs can be easily parallelized. As a result, RNF-based CNN models can be 3-8x faster than their RNN counterparts.

We present two RNF-based CNN architectures for sentence classification and answer sentence selection problems. Experimental results on the Stanford Sentiment Treebank and the QASent and WikiQA datasets demonstrate that RNFs significantly improve CNN performance over linear filters by 4-5% accuracies and 3-6% MAP scores respectively. Analysis results suggest that RNFs perform much better than linear filters in detecting longer key phrases, which provide stronger cues for categorizing the sentences.

2 Approach

The aim of a convolution filter is to produce a local feature for a window of words. We describe a novel approach to learning filters using RNNs, which is especially suitable for NLP problems.

---

[1] The code is available at `https://github.com/bloomberg/cnn-rnf`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 912–917
Brussels, Belgium, October 31 - November 4, 2018.

We then present two CNN architectures equipped with RNFs for sentence classification and sentence matching tasks respectively.

2.1 Recurrent neural filters

We briefly review the linear convolution filter implementation by Kim (2014), which has been widely adopted in later works. Consider an m-gram word sequence $[\mathbf{x}_i, \cdots, \mathbf{x}_{i+m-1}]$, where $\mathbf{x}_i \in \mathbb{R}^k$ is a k-dimensional word vector. A linear convolution filter is a function applied to the m-gram to produce a feature $c_{i,j}$:

$$\begin{aligned} c_{i,j} &= f(\mathbf{w}_j^\top \mathbf{x}_{i:i+m-1} + b_j), \\ \mathbf{x}_{i:i+m-1} &= \mathbf{x}_i \oplus \mathbf{x}_{i+1} \oplus \cdots \oplus \mathbf{x}_{i+m-1}, \end{aligned} \quad (1)$$

where $\oplus$ is the concatenation operator, b_j is a bias term, and f is a non-linear activation function. We typically use multiple independent linear filters to yield a feature vector $\mathbf{c}_i$ for the word sequence. Linear convolution filters make strong assumptions about language that could harm the performance of NLP systems. First, linear filters assume local compositionality and ignore long-term dependencies in language. Second, they use separate parameters for each value of the time index, which hinders parameter sharing for the same word type (Goodfellow et al., 2016). The assumptions become more problematic if we increase the window size m.

We propose to address the limitations by employing RNNs to realize convolution filters, which we term *recurrent neural filters* (RNFs). RNFs compose the words of the m-gram from left to right using the same recurrent unit:

$$\mathbf{h}_t = \mathrm{RNN}(\mathbf{h}_{t-1}, \mathbf{x}_t), \quad (2)$$

where $\mathbf{h}_t$ is a hidden state vector that encoded information about previously processed words, and the function RNN is a recurrent unit such as Long Short-Term Memory (LSTM) unit (Hochreiter and Schmidhuber, 1997) or Gated Recurrent Unit (GRU) (Cho et al., 2014). We use the last hidden state $\mathbf{h}_{i+m-1}$ as the RNF output feature vector $\mathbf{c}_i$. Features learned by RNFs are interdependent of each other, which permits the learning of complementary information about the word sequence. The left-to-right word composing procedure in RNFs preserves word order information and implicitly models long-term dependencies in language. RNFs can be treated as simple drop-in replacements of linear filters and potentially adopted in numerous CNN architectures.

2.2 CNN architectures

Sentence encoder We use a CNN architecture with one convolution layer followed by one max pooling layer to encode a sentence. In particular, the RNFs are applied to every possible window of m words in the sentence $\{\mathbf{x}_{1:m}, \mathbf{x}_{2:m+1}, \cdots, \mathbf{x}_{n-m+1:n}\}$ to generate feature maps $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_{n-m+1}]$. Then a max-over-time pooling operation (Collobert et al., 2011) is used to produce a fixed size sentence representation: $\mathbf{v} = \max\{\mathbf{C}\}$.

Sentence classification We use a CNN architecture that is similar to the CNN-static model described in Kim (2014) for sentence classification. After obtaining the sentence representation $\mathbf{v}$, a fully connected softmax layer is used to map $\mathbf{v}$ to an output probability distribution. The network is optimized against a cross-entropy loss function.

Sentence matching We exploit a CNN architecture that is nearly identical to the CNN-Cnt model introduced by Yang et al. (2015). Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be the vector representations of the two sentences. A bilinear function is applied to $\mathbf{v}_1$ and $\mathbf{v}_2$ to produce a sentence matching score. The score is combined with two word matching count features and fed into a sigmoid layer. The output of the sigmoid layer is used by binary cross-entropy loss to optimize the model.

3 Experiments

We evaluate RNFs on some of the most popular datasets for the sentence classification and sentence matching tasks. After describing the experimental setup, we compare RNFs against both linear filters and conventional RNN models, and report our findings in § 3.2.

3.1 Experimental settings

Data We use the Stanford Sentiment Treebank (Socher et al., 2013) in our sentence classification experiments. We report accuracy results for both binary classification and fine-grained classification settings. Two answer sentence selection datasets, QASent (Wang et al., 2007) and WikiQA (Yang et al., 2015), are adopted in our sentence matching experiments. We use MAP and MRR to evaluate the performance of answer sentence selection models.

Competitive systems We consider CNN variants with linear filters and RNFs. For RNFs, we adopt two implementations based on GRUs and LSTMs respectively. We also compare against the following RNN variants: GRU, LSTM, GRU with max pooling, and LSTM with max pooling.[2] We use the publicly available 300-dimensional GloVe vectors (Pennington et al., 2014) pre-trained with 840 billion tokens to initialize the word embeddings for all the models. The word vectors are fixed during downstream training. Finally, we report best published results for each dataset.[3]

Parameter tuning For all the experiments, we tune hyperparameters on the development sets and report results obtained with the selected hyperparameters on the test sets. After the preliminary search, we choose the number of hidden units (feature maps) from $\{200, 300, 400\}$, and the filter window width from $\{2, 3, 4, 5\}$ for linear filters and from $\{5, 6, 7, 8\}$ for RNFs. Linear filters tend to perform well with small window widths, while RNFs work better with larger window widths. We apply dropout to embedding layers, pooling layers, RNN input layers, and RNN recurrent layers. The dropout rates are selected from $\{0, 0.2, 0.4\}$, where 0 indicates that dropout is not used for the specific layer. Optimization is performed by Adam (Kingma and Ba, 2015) with early stopping. We conduct random search with a budget of 100 runs to seek the best hyperparameter configuration for each system.

3.2 Results

The evaluation results on sentiment classification and answer sentence selection are shown in Table 1 and Table 2 respectively. RNFs significantly outperform linear filters on both tasks. In fact, we find that CNN-RNF variants significantly outperform CNN-linear-filter on nearly every hyperparameter configuration in our experiments. On the Stanford Sentiment Treebank, CNN-RNF-LSTM outperforms CNN-linear-filter by 5.4% and 3.9% accuracies on the fine-grained and binary classification settings respectively. We observe similar performance boosts by adopting RNFs for CNNs on the QASent and WikiQA test sets. CNN-RNF-GRU improves upon CNN-linear-filter by 3.7% MRR score on QASent and CNN-RNF-LSTM performs better than CNN-linear-filter by 6.1% MAP score on WikiQA. In particular, CNN-RNF-LSTM achieves 53.4% and 90.0% accuracies on the fine-grained and binary sentiment classification tasks respectively, which match the state-of-the-art results on the Stanford Sentiment Treebank. CNN-RNF-LSTM also obtains competitive results on answer sentence selection datasets, despite the simple model architecture compared to state-of-the-art systems.

System	Fine-grained	Binary
CNN variants		
CNN-linear-filter	48.0	86.1
CNN-RNF-GRU	53.0	**90.0**
CNN-RNF-LSTM	**53.4**	**90.0**
RNN variants		
GRU	50.5	88.7
LSTM	50.3	89.3
GRU-maxpool	51.7	89.7
LSTM-maxpool	51.6	89.8
Best published results		
(Lei et al., 2017)	53.2	89.9

Table 1: Accuracy results on the Stanford Sentiment Treebank. The best results obtained from our implementations are in **bold**. The best published results are underlined.

Conventional RNN models clearly benefit from max pooling, especially on the task of answer sentence selection. Like RNF-based CNN models, max-pooled RNNs also consist of two essential layers. The recurrent layer learns a set of hidden states corresponding to different time steps, and the max pooling layer extracts the most salient information from the hidden states. However, a hidden state in RNNs encodes information about all the previous time steps, while RNFs focus on detecting local features from a window of words that can be more relevant to specific tasks. As a result, RNF-based CNN models perform consistently better than max-pooled RNN models.

CNN-RNF models are much faster to train than their corresponding RNN counterparts, despite they have the same numbers of parameters, as RNFs are applied to word sequences of shorter lengths and the computation is parallelizable. The training of CNN-RNF-LSTM models takes 10-20 mins on the Stanford Sentiment Treebank, which is 3-8x faster than the training of LSTM models, on an NVIDIA Tesla P100 GPU accelerator.

---

[2] Max pooling performed better than mean pooling in our preliminary experiments.

[3] We exclude results obtained from systems using external resources beyond word embeddings.

System	QASent		WikiQA	
	MAP	MRR	MAP	MRR
CNN variants				
CNN-linear-filter	0.750	0.813	0.668	0.687
CNN-RNF-GRU	0.773	**0.850**	0.726	0.738
CNN-RNF-LSTM	**0.780**	0.841	**0.729**	**0.747**
RNN variants				
GRU	0.739	0.815	0.658	0.678
LSTM	0.745	0.806	0.651	0.671
GRU-maxpool	0.744	0.814	0.712	0.724
LSTM-maxpool	0.762	0.825	0.701	0.711
Best published results				
(Rao et al., 2016)	0.801	0.877	0.701	0.718
(Wang and Jiang, 2017)	-	-	0.743	0.755

Table 2: Evaluation results on answer sentence selection datasets. The best results obtained from our implementations are in **bold**. The best published results are underlined.

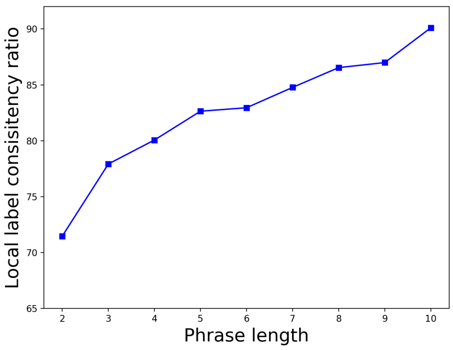

Figure 1: Local label consistency ratio results computed on SST dev set.

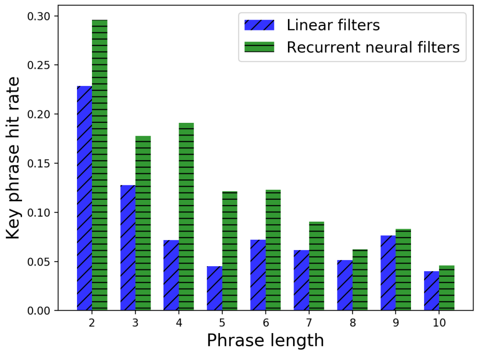

Figure 2: Key phrase hit rate results computed on SST dev set. The RNFs are implemented with LSTMs.

4 Analysis

We now investigate why RNFs are more effective than linear convolution filters on the binary sentiment classification task. We perform quantitative analysis on the development set of the Stanford Sentiment Treebank (SST), in which sentiment labels for some phrases are also available.

Local label consistency (LLC) ratio We first inspect whether longer phrases have a higher tendency to express the same sentiment as the entire sentence. We define the *local label consistency (LLC) ratio* as the ratio of m-grams that share the same sentiment labels as the original sentences. The LLC ratios with respect to different phrase lengths are shown in Figure 1. Longer phrases are more likely to convey the same sentiments as the original sentences. Therefore, the ability to model long phrases is crucial to convolution filters.

Key phrase hit rate We examine linear filters and RNFs on the ability to detect a key phrase in a sentence. Specifically, we define the key phrases for a sentence to be the set of phrases that are labeled with the same sentiment as the original sentence. The *key phrase hit rate* is then defined as the ratio of filter-detected m-grams that fall into the corresponding key phrase sets. The filter-detected m-gram of a sentence is the one whose convolution feature vector has the shortest Euclidean distance to the max-pooled vector. The hit rate results computed on SST dev set are presented in Figure 2. As shown, RNFs consistently perform better than linear filters on identifying key phrases across different phrase lengths, especially on phrases of moderate lengths (4-7). The results suggest that RNFs are superior to linear filters, as they can better model longer phrases whose labels are more consistent with the sentences.[4]

5 Related work

Linear convolution filters are dominated in CNN-based systems for both computer vision and natural language processing tasks. One exception is the work of Zoumpourlis et al. (2017), which proposes a convolution filter that is a function with quadratic forms through the Volterra kernels. However, this non-linear convolution filter is developed in the context of a computational model of the visual cortex, which is not suitable for NLP problems. In contrast, RNFs are specifically derived for NLP tasks, in which a different form of non-linearity, language compositionality, often plays a critical role.

Several works have employed neural network architectures that contain both CNN and RNN components to tackle NLP problems. Tan et al. (2016) present a deep neural network for answer sentence selection, in which a convolution layer is applied to the output of a BiLSTM layer for extracting sentence representations. Ma and Hovy (2016) propose to compose character representations of a word using a CNN, whose output is then fed into a BiLSTM for sequence tagging. Kalchbrenner and Blunsom (2013) introduce a neural architecture that uses a sentence model based on CNNs and a discourse model based on RNNs. Their system achieves state-of-the-art results on the task of dialogue act classification. Instead of treating an RNN and a CNN as isolated components, our work directly marries RNNs with the convolution operation, which illustrates a new direction in mixing RNNs with CNNs.

6 Conclusion and future work

We present RNFs, a new class of convolution filters based on recurrent neural networks. RNFs sequentially apply the same recurrent unit to words of a phrase, which naturally capture language compositionality and implicitly model long-term dependencies. Experiments on sentiment classification and answer sentence selection tasks demonstrate that RNFs give a significant boost in performance compared to linear convolution filters. RNF-based CNNs also outperform a variety of RNN-based models, as they focus on extracting local information that could be more relevant to particular problems. The quantitive analysis reveals that RNFs can handle long phrases much better than linear filters, which explains their superiority over the linear counterparts. In the future, we would like to investigate the effectiveness of RNFs on a wider range of NLP tasks, such as natural language inference and machine translation.

7 Acknowledgments

We thank Chunyang Xiao for helping us to run the analysis experiments. We thank Kazi Shefaet Rahman, Ozan Irsoy, Chen-Tse Tsai, and Lingjia Deng for their valuable comments on earlier versions of this paper. We also thank the EMNLP reviewers for their helpful feedback.

References

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

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

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

---

[4] Phrases of very long lengths (8-10) are rarely annotated in SST dev data, which could explain why linear filters and RNFs achieve similar hit rates, as small data sample often leads to high variance.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2017. Deriving neural architectures from sequence and graph kernels. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the ACM International on Conference on Information and Knowledge Management (CIKM)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Lstm-based deep learning models for non-factoid answer selection. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Georgios Zoumpourlis, Alexandros Doumanoglou, Nicholas Vretos, and Petros Daras. 2017. Non-linear convolution filters for cnn-based learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Exploiting Rich Syntactic Information for Semantic Parsing with Graph-to-Sequence Model

Kun Xu[1]*, **Lingfei Wu**[2], **Zhiguo Wang**[2], **Mo Yu**[2], **Liwei Chen**[3], **Vadim Sheinin**[2]
[1]Tencent AI Lab
[2]IBM Research
[3]Peking University, Beijing, China
`{syxu828,zgw.tomorrow,gflfof}@gmail.com`, `lwu@email.wm.edu`
`chenliwei@pku.edu.cn`,`vadims@us.ibm.com`

Abstract

Existing neural semantic parsers mainly utilize a sequence encoder, i.e., a sequential LSTM, to extract word order features while neglecting other valuable syntactic information such as dependency or constituent trees. In this paper, we first propose to use the *syntactic graph* to represent three types of syntactic information, i.e., word order, dependency and constituency features; then employ a graph-to-sequence model to encode the syntactic graph and decode a logical form. Experimental results on benchmark datasets show that our model is comparable to the state-of-the-art on Jobs640, ATIS, and Geo880. Experimental results on adversarial examples demonstrate the robustness of the model is also improved by encoding more syntactic information.

1 Introduction

The task of *semantic parsing* is to translate text to its formal meaning representations, such as logical forms or structured queries. Recent neural semantic parsers approach this problem by learning soft alignments between natural language and logical forms from (text, logic) pairs (Jia and Liang, 2016; Dong and Lapata, 2016; Krishnamurthy et al., 2017). All these parsers follow the conventional encoder-decoder architecture that first encodes the text into a distributional representation and then decodes it to a logical form. These parsers may differ in the choice of the decoders, such as sequence or tree decoders, but they utilize the same encoder which is essentially a sequential Long Short-Term Memory network (SeqLSTM). This encoder only extracts word order features while neglecting useful syntactic information, such as dependency parse and constituency parse.

However, the syntactic features capture important structural information of the natural language input, which complements the simple word sequence. For example, a dependency graph presents grammatical relations that hold among the words; and a constituent tree provides a phrase structure representation. Intuitively, by incorporating such additional information, the encoder could produce a more meaningful and robust sentence representation. The combination of these features (i.e., sequence + trees) forms a general graph structure (see Figure 1). This inspires us to apply a graph encoder to produce a representation of a graph-structured input. The graph encoder also has the advantages that it could simultaneously encode all types of syntactic contexts, and incorporate multiple types of syntactic structures in a unified way.

In this paper, we first introduce a structure, namely *syntactic graph*, to represent three types of syntactic information, i.e., word order, dependency and constituency features (see §2). We then employ a novel graph-to-sequence (Graph2Seq) model (Xu et al., 2018), which consists of a graph encoder and a sequence decoder, to learn the representation of the syntactic graph (see §3). Specifically, the graph encoder learns the representation of each node by aggregating information from its K-hop neighbors. Given the learned node embeddings, the graph encoder uses a pooling-based method to generate the graph embedding. On the decoder side, a Recurrent Neural Network (RNN) decoder takes the graph embedding as its initial hidden state to generate the logical form while employing an attention mechanism over the node embeddings. Experimental results show that our model achieves the competitive performance on Jobs640, ATIS, and Geo880 datasets.

Different from existing works, we also investigate the robustness of our model by evaluating the model on two types of adversarial examples (Belinkov and Bisk, 2017; Cheng et al., 2018).

---

[*] Work done when the author was at IBM Research.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 918–924
Brussels, Belgium, October 31 - November 4, 2018.

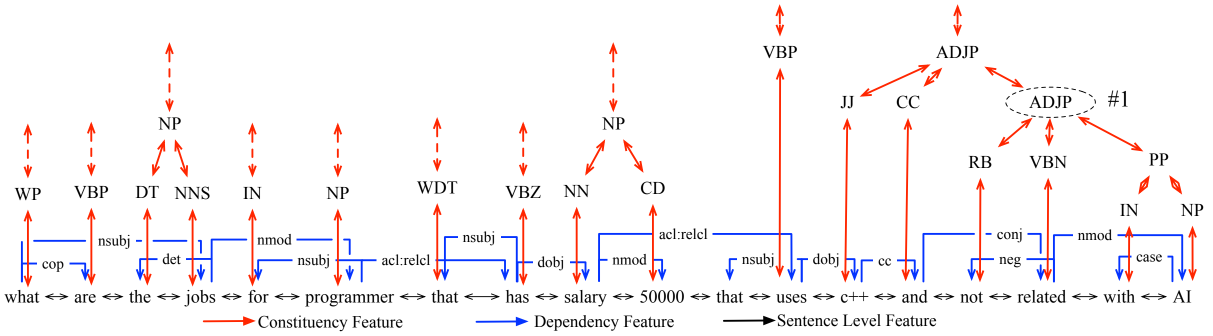

Figure 1: The syntactic graph for the Jobs640 question *what are the jobs for programmer that has salary 50000 that uses c++ and not related with AI*. Due to the space limitation, the constituent tree is partially shown here.

Experimental results show that the model coupling all syntactic features has the best robustness, achieving the best performance. Our code and data is available at `https://github.com/IBM/Text-to-LogicForm`.

2 Syntactic Graph

We represent three types of syntactic features, i.e., word order, dependency parse and constituency parse, as the syntactic graph (see Figure 1).

- **Word Order Features.** Previous neural semantic parsers mainly use these features by building a SeqLSTM that works on the word sequence. Our syntactic graph also incorporates this information by generating a node for each word and connecting them in the chain form. In order to capture the forward and backward contextual information, we link these nodes in two directions, that is, from left to right and from right to left.
- **Dependency Features.** A dependency parse describes the grammatical relations that hold among words. Reddy et al. (2016, 2017) have demonstrated that the dependency parse tree could be directly transformed to a logical form, which indicates that the dependency information (i.e., tree structure and dependency labels) is critical to the semantic parsing task. We incorporate this information into the syntactic graph by adding directed edges between the word nodes and assign them with dependency labels.
- **Constituency Features.** Similar to the dependency parse, the constituency parse represents the phrase structure, which is also important to the semantic parsing task. Take Figure 1 as an example: given the constituent tree that explicitly annotates "*not related with AI*" (node #1) is a proposition phrase, the model could learn a meaningful embedding for this phrase by encoding this structure into the model. Motivated by this observation, we add the non-terminal nodes of the constituent tree and the edges describing their parent-child relationships into the syntactic graph.

3 Graph-to-sequence Model for Semantic Parsing

After building the syntactic graph for the input text, we employ a novel graph-to-sequence model (Xu et al., 2018), which includes a graph encoder and a sequence decoder with attention mechanism, to map the syntactic graph to the logical form. Conceptually, the graph encoder generates node embeddings for each node by accumulating information from its K-hop neighbors, and then produces a graph embedding for the entire graph by abstracting all these node embeddings. Next, the sequence decoder takes the graph embedding as the initial hidden state, and calculates the attention over all node embeddings on the encoder side to generate logical forms. Note that this graph encoder does not explicitly encode the edge label information, therefore, for each labeled edge, we add a node whose text attribute is the edge's label.

Node Embedding. Given the syntactic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we take the embedding generation process for node $v \in \mathcal{V}$ as an example to explain the node embedding generation algorithm[1]:

(1) We first transform node v's text attribute to a feature vector, $\mathbf{a}_v$, by looking up the embedding matrix W_e;

(2) The neighbors of v are categorized into forward neighbors $\mathcal{N}_{\vdash}(v)$ and backward neighbors $\mathcal{N}_{\dashv}(v)$ according to the edge direction. In particular, $\mathcal{N}_{\vdash}(v)$ returns the nodes that v directs to and $\mathcal{N}_{\dashv}(v)$ returns the nodes that direct to v;

(3) We aggregate the **forward** representations of v's forward neighbors $\{\mathbf{h}_{u\vdash}^{k-1}, \forall u \in \mathcal{N}_{\vdash}(v)\}$ into

---

[1] Interested readers may refer to (Xu et al., 2018) for more implementation details.

a single vector, $\mathbf{h}^{k}_{\mathcal{N}_{\vdash}(v)}$, where $k\in\{1, ..., K\}$ is the iteration index. Specifically, this aggregator feeds each neighbor's vector to a fully-connected neural network and applies an element-wise max-pooling operation to capture different aspects of the neighbor set. Notice that, at iteration k, this aggregator only uses the representations generated at $k-1$. The initial forward representation of each node is its feature vector calculated in step (1);

(4) We concatenate v's current **forward** representation $\mathbf{h}^{k-1}_{v\vdash}$ with the newly generated neighborhood vector $\mathbf{h}^{k}_{\mathcal{N}_{\vdash}(v)}$. The resulted vector is fed into a fully connected layer with nonlinear activation function σ, which updates the **forward** representation of v, $\mathbf{h}^{k}_{v\vdash}$, to be used at the next iteration;

(5) We update the **backward** representation of v, $\mathbf{h}^{k}_{v\dashv}$ using the similar procedure as introduced in step (3) and (4) except that operating on the backward representations;

(6) We repeat steps (3)$\sim$(5) K times, and the concatenation of the final forward and backward representations is used as the final representation of v. Since the neighbor information from different hops may have different impacts on the node embedding, we learn a distinct aggregator at each iteration.

Graph Embedding. We feed the obtained node embeddings into a fully-connected neural network, and apply the element-wise *max*-pooling operation on all node embeddings. We did not find substantial performance improvement using *mean*-pooling.

Sequence Decoding. The decoder is an RNN which predicts the next token y_i given all the previous words $y_{<i} = y_1, ..., y_{i-1}$, the RNN hidden state s_i for time-step i and the context vector c_i that captures the attention of the encoder side. In particular, the context vector c_i depends on a set of node representations ($\mathbf{h}_1$,...,$\mathbf{h}_{\mathcal{V}}$) to which the encoder maps the input graph. The context vector c_i is dynamically computed using an attention mechanism over the node representations. The whole model is jointly trained to maximize the conditional log-probability of the correct description given a source graph. In the inference phase, we use the beam search algorithm to generate a description with beam size = 3.

4 Experiments

We evaluate our model on three datasets: Jobs640, a set of 640 queries to a database of job listings; Geo880, a set of 880 queries to a database of U.S. geography; and ATIS, a set of 5,410 queries to a flight booking system. We use the standard train/development/test split as previous works, and the logical form accuracy as our evaluation metric.

Method	Jobs	Geo	ATIS
Zettlemoyer and Collins (2007)	79.3	86.1	84.6
Kwiatkowski et al. (2011)	-	88.6	82.8
Liang et al. (2011)	90.7	87.9	-
Kwiatkowski et al. (2013)	-	**89.0**	-
Wang et al. (2014)	-	90.4	**91.3**
Zhao and Huang (2015)	85.0	88.9	84.2
Jia and Liang (2016)	-	85.0	76.3
Dong and Lapata (2016)-Seq2Seq	87.1	85.0	84.2
Dong and Lapata (2016)-Seq2Tree	90.0	87.1	84.6
Rabinovich et al. (2017)	**92.9**	85.7	85.3
Graph2Seq	91.2	88.1	85.9
w/o word order features	86.7	84.4	82.9
w/o dependency features	89.3	85.8	83.8
w/o constituency features	88.9	84.7	84.6
w/ ONLY word order features	88.0	84.8	83.1
BASELINE	88.1	84.9	83.0

Table 1: Exact-match accuracy on Jobs640, Geo880 and ATIS.

The model is trained using the Adam optimizer (Kingma and Ba, 2014), with mini-batch size 30. Our hyper-parameters are cross-validated on the training set for Jobs640 and Geo880, and tuned on the development set for ATIS. The learning rate is set to 0.001. The decoder has 1 layer, and its hidden state size is 300. The dropout strategy (Srivastava et al., 2014) with the ratio of 0.5 is applied at the decoder layer to avoid overfitting. W_e is initialized using GloVe word vectors from Pennington et al. (2014) and the dimension of word embedding is 300. For the graph encoder, the hop size K is set to 10, the non-linearity function σ is implemented as ReLU (Glorot et al., 2011), the parameters of the aggregators are randomly initialized. We use the Stanford CoreNLP tool (Manning et al., 2014) to generate the dependency and constituent trees.

Results and Discussion. Table 1 summarizes the results of our model and existing semantic parsers on three datasets. Our model achieves competitive performance on Jobs640, ATIS and Geo880. Our work is the first to use both multiple trees and the word sequence for semantic parsing, and it outperforms the Seq2Seq model reported in Dong and Lapata (2016), which only uses limited syntactic information.

Comparison with Baseline. To better demonstrate that our work is an effective way to utilize both multiple trees and the word sequence for semantic parsing, we compare with an additional straightforward baseline method (referred as BASELINE in Table 1). To deal with the graph input, the BASELINE decomposes the graph embedding to two steps and applies different types of encoders sequentially: (1) a SeqLSTM to extract word order features, which results in word embeddings, W_{seq}; (2) two TreeLSTMs (Tai et al., 2015) to extract the dependency tree and constituency features while taking W_{seq} as initial word embeddings. The resulted word embeddings and nonterminal node embeddings (from TreeLSTMs) are then fed into a sequence decoder.

We can see that our model significantly outperforms the BASELINE. One possible reason is that our graph encoder jointly extracts these features in a unified model by propagating the dependency and constituency information to all nodes in the syntactic graph. However, BASELINE separately models these features using two distinct TreeLSTMs. As a result, the non-terminal tree nodes only retain only one type of syntactic information propagated from their descendants in the tree.

Ablation Study. In Table 1, we also report the results of three ablation variants of our model, i.e., without word order features/dependency features/constituency features. We find that Graph2Seq is superior to Seq2Seq (Dong and Lapata, 2016) which is expected since Graph2Seq exploits more syntactic information. Among these features, the word order feature have more impact on the performance than other two syntactic features. By incorporating either the dependency or the constituency features, the model could gain further performance improvement, which underlines the importance of utilizing more aspects of syntactic information. Finally, removing both syntactic features (w/ ONLY word order) performs slightly worse compared to the Seq2Seq baseline. This shows that using K=10 hops is good enough for memorizing the sentences in our benchmarks, although still weaker compared to a bidirectional LSTM encoder.

A natural question here is on which type of queries our model could benefit from incorporating these parse features. By analyzing the queries and our predicted logical forms, we find that the parse features mainly improve the prediction accuracy for the queries with complex logical forms. Table 2 gives some running examples of complicated queries in three datasets. We find that the model that exploits three syntactic information could correctly predict these logical forms while the model that only uses word order features may fail.

Complicated Query & Predicted Logical Forms
Jobs Q: *what are the jobs for programmer that has salary 50000 that uses c++ and not related with AI*
Pred: answer(J,(job(J),-((area(J,R),const(R,'ai'))), language(J,L),const(L,'c++'), title(J,P), const(P,'Programmer'),salary_greater_than(J, 50000,year)))).
Geo Q: *which is the density of the state that the largest river in the united states run through*
Pred: answer(A,(density(B,A),state(B), longest(C,(river(C),loc(C,D),const(D,id(usa)))), traverse(C,B))))
ATIS Q: *please find a flight round trip from los angeles to tacoma washington with a stopover in san francisco not exceeding the price of 300 dollars for june tenth 1993*
Pred: (lambda \$0 e (and (flight \$0) (round_trip \$0) (from \$0 *los angeles*) (to \$0 *tacoma washington*) (stop \$0 *san francisco*) (< (cost \$0) 300) (day_number \$0 *tenth*) (month \$0 *june*) (year \$0 1993)))

Table 2: Examples of complicated query and predicted logical forms.

Robustness Study. Different from previous works, we evaluate the robustness of our model by creating adversarial examples with the hope to investigate the impact of introducing more syntactic information on robustness. Specifically, we create two types of adversarial examples and conduct experiments on the ATIS dataset. Following Belinkov and Bisk (2017), we first experiment with the synthetic noise, **SWAP**, which swaps two letters (e.g. noise→nosie). It is common to see such noisy information when typing quickly. Given a text, we randomly perform swap on $m \in \{1, 2, 3, 4, 5\}$ words that **not** correspond to the operators or arguments in logical forms, ensuring the meaning of the text is not changed. We train Graph2Seq on the training data and first evaluate it on the original development data, Dev_{ori}. Then we use the same model but evaluate it on a variant of Dev_{ori}, whose queries contain m swapped words.

Figure 2 summarizes the results of our model on the first type of adversarial examples, i.e., the ATIS development set with the SWAP noise. From Figure 2, we can see that (1) the performance of our model on all combinations of features degrade significantly when increasing the number of swapped words; (2) the model that uses three syntactic features (our default model) always achieves the best performance, and the performance gap

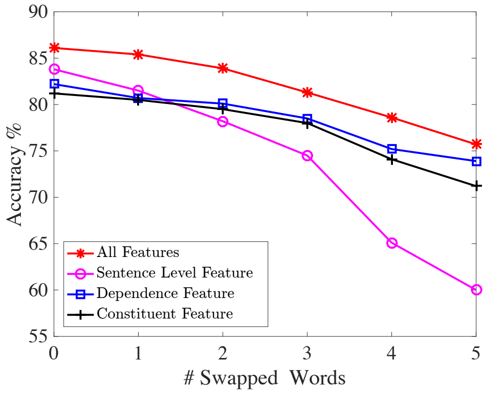

Figure 2: The logical form accuracy on the development set of ATIS with various swapped words number.

compared to others increases when rising the number of swapped words; (3) word order features are the most sensitive to the word sequence while the dependency and constituency features seem more robust to such noisy information; (4) thanks to the robustness of the dependency and constituency features, the default model performs significantly better than the one that only uses word order features on the noisy sentences. These findings demonstrate that incorporating more aspects of syntactic information could enhance the robustness of the model.

We also experiment with the paraphrase of the input text as the second type of adversarial examples. More specifically, we collect the paraphrase of a text by first translating it to the other language such as Chinese and then translating it back to English, using the Google Translate service. We use this method to collect a new variant of Dev$_{ori}$ whose queries are the paraphrases of the original ones. By manually reading these queries, we find 94% queries convey the same meaning as original ones. Similar to the first experiment, we still train the model on Dev$_{ori}$ and evaluate it on the newly created dataset.

Feature	Acc$_{ori}$	Acc$_{para}$	Diff.
Word Order	84.8	78.7	-6.1
Dep	83.5	80.1	-3.4
Cons	82.9	77.3	-5.6
Dep + Cons	84.0	80.7	-3.3
Word Order + Dep	85.2	82.3	-2.9
Word Order + Cons	84.9	79.9	-5.0
Word Order + Dep + Cons	**86.0**	**83.5**	**-2.5**

Table 3: Evaluation results on ATIS where Acc$_{ori}$ and Acc$_{para}$ denote the accuracy on the original and paraphrased development set of ATIS, respectively.

Table 3 shows the results of our model on the second type of adversarial examples, i.e., the paraphrased ATIS development set. We also report the result of our model on the original ATIS development set. We can see that (1) no matter which feature our model uses, the performance degrades at least 2.5% on the paraphrased dataset; (2) the model that only uses word order features achieves the worst robustness to the paraphrased queries while the dependency feature seems more robust than other two features. (3) simultaneously utilizing three syntactic features could greatly enhance the robustness of our model. These results again demonstrate that our model could benefit from incorporating more aspects of syntactic information.

5 Related Work

Existing works of generating text representation has evolved into two main streams. The first one is based on the word order, that is, either generating general purpose and domain independent embeddings of word sequences (Wu et al., 2018a; Arora et al., 2017), or building Bi-directional LSTMs over the text (Zhang et al., 2018). These methods neglect other syntactic information, which, however, has been proved to be useful in shallow semantic parsing, e.g., semantic role labeling (Punyakanok et al., 2008). To address this, recent works attempt to incorporate these syntactic information into the text representation. For example, Xu et al. (2016) builds separated neural networks for different types of syntactic annotation. Gormley et al. (2015); Wu et al. (2018b) decompose a graph to simpler sub-graphs and embed these subgraphs independently. Our approach, compared to the above methods, provided a unified solution to arbitrary combinations of syntactic graphs. In parallel to syntactic features, other works leverage additional information such as dialogue and paraphrasing for semantic parsing (Su and Yan, 2017; Gur et al., 2018).

6 Conclusions

Existing neural semantic parsers mainly leverage word order features while neglecting other valuable syntactic information. To address this, we propose to build a syntactic graph which represents three types of syntactic information, and further apply a novel graph-to-sequence model to map the syntactic graph to a logical form. Experimental results show that the robustness of our model is improved due to the incorporating more aspects of syntactic information, and our model outperforms previous semantic parsing systems.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *CoRR*, abs/1601.01280.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 315–323.

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1774–1784.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. 2018. Dialsql: Dialogue based structured query generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1339–1349.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *CoRR*, abs/1606.03622.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1512–1523.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. *Computational Linguistics*, 39:389–446.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. *arXiv preprint arXiv:1702.03196*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. *arXiv preprint arXiv:1704.05974*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Adrienne Wang, Tom Kwiatkowski, and Luke S. Zettlemoyer. 2014. Morpho-syntactic lexical generalization for ccg semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1284–1295.

Lingfei Wu, Ian E.H. Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J. Witbrock. 2018a. Word mover's embedding: From word2vec to document embedding. In *EMNLP*.

Lingfei Wu, Ian En-Hsu Yen, Fangli Xu, Pradeep Ravikuma, and Michael Witbrock. 2018b. D2ke: From distance to kernel and embedding. *arXiv preprint arXiv:1802.04956*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *ACL 2016*.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state lstm for text representation. *arXiv preprint arXiv:1805.02474*.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *HLT-NAACL*.

Retrieval-Based Neural Code Generation

Shirley Anugrah Hayati **Raphael Olivier** **Pravalika Avvaru**
Pengcheng Yin **Anthony Tomasic** **Graham Neubig**

Language Technologies Institute, Carnegie Mellon University
`{shayati,rolivier,pavvaru,pcyin,tomasic,gneubig}@cs.cmu.edu`

Abstract

In models to generate program source code from natural language, representing this code in a tree structure has been a common approach. However, existing methods often fail to generate complex code correctly due to a lack of ability to memorize large and complex structures. We introduce RECODE, a method based on subtree retrieval that makes it possible to explicitly reference existing code examples within a neural code generation model. First, we retrieve sentences that are similar to input sentences using a dynamic-programming-based sentence similarity scoring method. Next, we extract n-grams of action sequences that build the associated abstract syntax tree. Finally, we increase the probability of actions that cause the retrieved n-gram action subtree to be in the predicted code. We show that our approach improves the performance on two code generation tasks by up to +2.6 BLEU.[1]

1 Introduction

Natural language to code generation, a subtask of semantic parsing, is the problem of converting natural language (NL) descriptions to code (Ling et al., 2016; Yin and Neubig, 2017; Rabinovich et al., 2017). This task is challenging because it has a well-defined structured output and the input structure and output structure are in different forms.

A number of neural network approaches have been proposed to solve this task. Sequential approaches (Ling et al., 2016; Jia and Liang, 2016; Locascio et al., 2016) convert the target code into a sequence of symbols and apply a sequence-to-sequence model, but this approach does not ensure that the output will be syntactically correct. Tree-based approaches (Yin and Neubig, 2017; Rabinovich et al., 2017) represent code as Abstract Syntax Trees (ASTs), which has proven effective in improving accuracy as it enforces the well-formedness of the output code. However, representing code as a tree is not a trivial task, as the number of nodes in the tree often greatly exceeds the length of the NL description. As a result, tree-based approaches are often incapable of generating correct code for phrases in the corresponding NL description that have low frequency in the training data.

In machine translation (MT) problems (Zhang et al., 2018; Gu et al., 2018; Amin Farajian et al., 2017; Li et al., 2018), hybrid methods combining retrieval of salient examples and neural models have proven successful in dealing with rare words. Following the intuition of these models, we hypothesize that our model can benefit from querying pairs of NL descriptions and AST structures from training data.

In this paper, we propose RECODE, and adaptation of Zhang et al. (2018)'s retrieval-based approach neural MT method to the code generation problem by expanding it to apply to generation of tree structures. Our main contribution is to introduce the use of retrieval methods in neural code generation models. We also propose a dynamic programming-based sentence-to-sentence alignment method that can be applied to similar sentences to perform word substitution and enable retrieval of imperfect matches. These contributions allow us to improve on previous state-of-the-art results.

2 Syntactic Code Generation

Given an NL description q, our purpose is to generate code (e.g. Python) represented as an AST a. In this work, we start with the syntactic code gen-

---

[1] Code available at `https://github.com/sweetpeach/ReCode`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 925–930
Brussels, Belgium, October 31 - November 4, 2018.

eration model by Yin and Neubig (2017), which uses sequences of actions to generate the AST before converting it to surface code. Formally, we want to find the best generated AST $\hat{a}$ given by:

$$\hat{a} = \arg\max_{a} p(a|q)$$

$$p(a|q) = \prod_{t=1}^{T} p(y_t|y_{<t}, q)$$

where y_t is the action taken at time step t and $y_{<t} = y_1 ... y_{t-1}$ and T is the number of total time steps of the whole action sequence resulting in AST a.

We have two types of actions to build an AST: APPLYRULE and GENTOKEN. APPLYRULE(r) expands the current node in the tree by applying production rule r from the abstract syntax grammar[2] to the current node. GENTOKEN(v) populates terminal nodes with the variable v which can be generated from vocabulary or by COPYing variable names or values from the NL description. The generation process follows a preorder traversal starting with the `root` node. Figure 1 shows an action tree for the example code: the nodes correspond to actions per time step in the construction of the AST.

Interested readers can reference Yin and Neubig (2017) for more detail of the neural model, which consists of a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) encoder-decoder with action embeddings, context vectors, parent feeding, and a copy mechanism using pointer networks.

3 RECODE: Retrieval-Based Neural Code Generation

We propose RECODE, a method for retrieval-based neural syntactic code generation, using retrieved action subtrees. Following Zhang et al. (2018)'s method for neural machine translation, these retrieved subtrees act as templates that bias the generation of output code. Our pipeline at test time is as follows:

- retrieve from the training set NL descriptions that are most similar with our input sentence (§3.1),
- extract **n-gram action subtrees** from these retrieved sentences' corresponding target ASTs (§3.2),
- alter the copying actions in these subtrees, by substituting words of the retrieved sentence with corresponding words in the input sentence (§3.3), and
- at every decoding step, increase the probability of actions that would lead to having these subtrees in the produced tree (§3.4).

3.1 Retrieval of Training Instances

For every retrieved NL description q_m from training set (or *retrieved sentence* for short), we compute its similarity with input q, using a sentence similarity formula (Gu et al., 2016; Zhang et al., 2018):

$$\text{sim}(q, q_m) = 1 - \frac{d(q, q_m)}{\max(|q|, |q_m|)}$$

where d is the edit distance. We retrieve only the top M sentences according to this metric where M is a hyperparameter. These scores will later be used to increase action probabilities accordingly.

3.2 Extracting N-gram Action Subtrees

In Zhang et al. (2018), they collect n-grams from the output side of the retrieved sentences and encourage the model to generate these n-grams. Word n-grams are obvious candidates when generating a sequence of words as output, as in NMT. However, in syntax-based code generation, the generation target is ASTs with no obvious linear structure. To resolve this problem, we instead use *retrieved pieces* of n-gram subtrees from the target code corresponding to the retrieved NL descriptions. Though we could select successive nodes in the AST as retrieved pieces, such as `[assign; expr*(targets); expr]` from Figure 1, we would miss important structural information from the rules that are used. Thus, we choose to exploit actions in the generation model rather than AST nodes themselves to be candidates for our retrieved pieces.

In the action tree (Figure 1), we considered only successive actions, such as subtrees where each node has one or no children, to avoid overly rigid structures or combinatorial explosion of the number of retrieved pieces the model has to consider. For example, such an action subtree would be given by `[assign → expr*(targets), expr(value) ; expr(value) → List; List → epsilon]`.

As the node in the action tree holds structural information about its children, we set the subtrees

---

[2] `https://docs.python.org/2/library/ast.html`

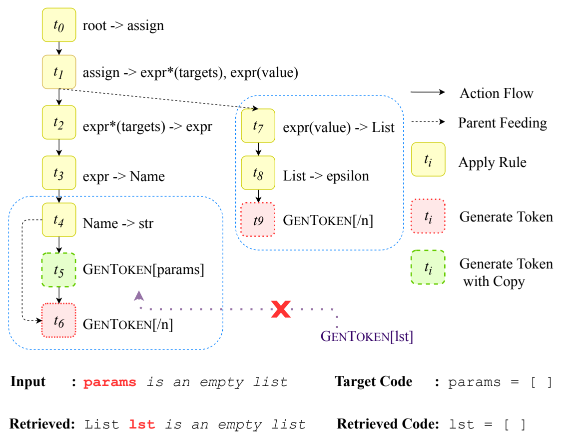

Figure 1: The action sequence used to generate AST for the target code given the input example. Dashed nodes represent terminals. Each node is labeled with time steps. APPLYRULE action is represented as rule in this figure. Blue dotted boxes denote 3-gram action subtrees. Italic words are unedited words. Red bold words are different object names.

to have a fixed depth, linear in the size of the tree. These can be considered "n-grams of actions", emphasizing the comparison with machine translation which uses n-grams of words. n is a hyperparameter to be tuned.

3.3 Word Substitution in Copy Actions

Using the retrieved subtree without modification is problematic if it contains at least one node corresponding to a COPY action because copied tokens from the retrieved sentence may be different from those in the input. Figure 1 shows an example when the input and retrieved sentence have four common words, but the object names are different. The extracted action n-gram would contain the rule that copies the second word ("lst") of the retrieved sentence while we want to copy the first word ("params") from the input.

By computing word-based edit distance between the input description and the retrieved sentence, we implement a one-to-one sentence alignment method that infers correspondences between uncommon words. For unaligned words, we alter all COPY rules in the extracted n-grams to copy tokens by their aligned counterpart, such as replace "params" with "lst", and delete the n-gram subtree, as it is not likely to be relevant in the predicted tree. Thus, in the example in Figure 1, the GENTOKEN(LST) action in t_5 will not be executed.

3.4 Retrieval-Guided Code Generation

N-gram subtrees from all retrieved sentences are assigned a score, based on the best similarity score of all instances where they appeared. We normalize the scores for each input sentence by subtracting the average over the training dataset.

At decoding time, incorporate these retrieval-derived scores into beam search: for a given time step, all actions that would result in one of the retrieved n-grams u to be in the prediction tree has its log probability $\log(p(y_t \mid y_1^{t-1}))$ increased by $\lambda * \text{score}(u)$ where λ is a hyperparameter, and $\text{score}(u)$ is the maximal $\text{sim}(q, q_m)$ from which u is extracted. The probability distribution is then renormalized.

Dataset	HS	Django
Train	533	16,000
Dev	66	1,000
Test	66	1,805
Avg. tokens in description	39.1	14.3
Avg. number of nodes of AST	136.6	17.2

Table 1: Dataset statistics as reported Yin and Neubig (2017)

4 Datasets and Evaluation Metrics

We evaluate RECODE with the Hearthstone (HS) (Ling et al., 2016) and Django (Oda et al., 2015) datasets, as preprocessed by Yin and Neubig (2017). HS consists of Python classes that implement Hearthstone card descriptions while Django contains pairs of Python source code and English pseudo-code from Django web framework. Table 1 summarizes dataset statistics.

For evaluation metrics, we use accuracy of exact match and the BLEU score following Yin and Neubig (2017).

5 Experiments

For the neural code generation model, we use the settings explained in Yin and Neubig (2017). For the retrieval method, we tuned hyperparameters and achieved best result when we set $n_{max} = 4$ and $\lambda = 3$ for both datasets[3]. For HS, we set $M = 3$ and $M = 10$ for Django.

We compare our model with Yin and Neubig (2017)'s model that we call YN17 for brevity, and a sequence-to-sequence (SEQ2SEQ) model that we implemented. SEQ2SEQ is an attention-enabled encoder-decoder model (Bahdanau et al., 2015). The encoder is a bidirectional LSTM and the decoder is an LSTM.

5.1 Results

Table 2 shows that RECODE outperforms the baselines in both BLEU and accuracy, providing ev-

---

[3] n-gram subtrees are collected up to n_{max}-gram

idence for the effectiveness of incorporating retrieval methods into tree-based approaches.

	HS		Django	
	Acc	**BLEU**	**Acc**	**BLEU**
SEQ2SEQ	0.0	55.0	13.9	67.3
YN17	16.2	75.8	71.6	84.5
ASN[†]	18.2	77.6	-	-
ASN + SUPATT[†]	*22.7*	*79.2*	-	-
RECODE	**19.6**	**78.4**	**72.8**	**84.7**

Table 2: Results compared to baselines. YN17 result is taken from Yin and Neubig (2017). ASN result is taken from Rabinovich et al. (2017)

.

We ran statistical significance tests for RECODE and YN17, using bootstrap resampling with N = 10,000. For the BLEU scores of both datasets, $p < 0.001$. For the exact match accuracy, $p < 0.001$ for Django dataset, but for Hearthstone, $p > 0.3$, showing that the retrieval-based model is on par with YN17. It is worth noting, though, that HS consists of long and complex code, and that generating exact matches is very difficult, making exact match accuracy a less reliable metric.

We also compare RECODE with Rabinovich et al. (2017)'s Abstract Syntax Networks with supervision (ASN+SUPATT) which is the state-of-the-art system for HS. RECODE exceeds ASN without extra supervision though ASN+SUPATT has a slightly better result. However, ASN+SUPATT is trained with supervised attention extracted through heuristic exact word matches while our attention is unsupervised.

5.2 Discussion and Analysis

From our observation and as mentioned in Rabinovich et al. (2017), HS contains classes with similar structure, so the code generation task could be simply matching the tree structure and filling the terminal tokens with correct variables and values. However, when the code consists of complex logic, partial implementation errors occur, leading to low exact match accuracy (Yin and Neubig, 2017). Analyzing our result, we find this intuition to be true not only for HS but also for Django.

Example 1	
"if offset is lesser than integer 0, sign is set to '-', otherwise sign is '+' "	**Input**
`sign = offset < 0 or '-'`	**YN17**
`sign = '-' if offset < 0 else '+'`	**RECODE**
`sign = '-' if offset < 0 else '+'`	**Gold**
Example 2	
"evaluate the function timesince with d, now and reversed set to boolean true as arguments, return the result."	**Input**
`return reversed(d, reversed=now)`	**YN17**
`return timesince(d, now, reversed=now)`	**RECODE**
`return timesince(d, now, reversed=True)`	**Gold**
Example 3	
"return an instance of SafeText , created with an argument s converted into a string ."	**Input**
`return SafeText(bool(s))`	**YN17**
`return SafeText(s)`	**RECODE**
`return SafeString(str(s))`	**Gold**

Table 3: Django examples on correct code and predicted code with retrieval (RECODE) and without retrieval (YN17).

NAME_BEGIN Earth Elemental NAME_END ATK_BEGIN 7 ATK_END DEF_BEGIN 8 DEF_END COST_BEGIN 5 COST_END DUR_BEGIN -1 DUR_END TYPE_BEGIN Minion TYPE_END PLAYER_CLS_BEGIN Shaman PLAYER_CLS_END RACE_BEGIN NIL RACE_END RARITY_BEGIN Epic RARITY_END DESC_BEGIN Taunt . Overload : (3) DESC_END. **Input**

YN17

```
class EarthElemental (MinionCard) :
    def __init__ (self) :
         super ( ).__init__ ("Earth Elemental", 5,
           CHARACTER_CLASS.SHAMAN, CARD_RARITY.EPIC,
           buffs=[Buff(ManaChange(Count
         (MinionSelector(None, BothPlayer())), -1))])
    def create_minion (self, player) :
         return Minion(7, 8, taunt=True)
```

RECODE

```
class EarthElemental (MinionCard) :
    def __init__ (self) :
         super ( ).__init__ ("Earth Elemental", 5,
           CHARACTER_CLASS.SHAMAN, CARD_RARITY.EPIC,
           overload=3)
    def create_minion (self, player) :
         return Minion(7, 8, taunt=True)
```

Gold

```
class EarthElemental (MinionCard) :
    def __init__ (self) :
         super ( ).__init__ ("Earth Elemental", 5,
           CHARACTER_CLASS.SHAMAN, CARD_RARITY.EPIC,
           overload=1)
    def create_minion (self, player) :
         return Minion(7, 8, taunt=True)
```

Table 4: HS examples on correct code and predicted code with retrieval (RECODE) and without retrieval (YN17).

Examining the generated output for the Django dataset in Table 3, we can see that in the first example, our retrieval model can successfully generate the correct code when YN17 fails. This difference suggests that our retrieval model benefits from the action subtrees from the retrieved sentences. In the second example, although our generated code does not perfectly match the reference code, it has a higher BLEU score compared to the output of YN17 because our model can predict part of the code (`timesince(d, now, reversed)`) correctly. The third example shows where our method fails to apply the correct action as it cannot cast `s` to `str` type while YN17 can at least cast `s` into a type (`bool`). Another common type of error that we found RECODE's generated outputs is incorrect variable copying, similarly to what is discussed in Yin and Neubig (2017) and Rabinovich et al. (2017).

Table 4 presents a result on the HS dataset[4]. We can see that our retrieval model can handle complex code more effectively.

6 Related Work

Several works on code generation focus on domain specific languages (Raza et al., 2015; Kushman and Barzilay, 2013). For general purpose code generation, some data-driven work has been

---

[4] More example of HS code is provided in the supplementary material.

done for predicting input parsers (Lei et al., 2013) or a set of relevant methods (Raghothaman et al., 2016). Some attempts using neural networks have used sequence-to-sequence models (Ling et al., 2016) or tree-based architectures (Dong and Lapata, 2016; Alvarez-Melis and Jaakkola, 2017). Ling et al. (2016); Jia and Liang (2016); Locascio et al. (2016) treat semantic parsing as a sequence generation task by linearizing trees. The closest work to ours are Yin and Neubig (2017) and Rabinovich et al. (2017) which represent code as an AST. Another close work is Dong and Lapata (2018), which uses a two-staged structure-aware neural architecture. They initially generate a low-level sketch and then fill in the missing information using the NL and the sketch.

Recent works on retrieval-guided neural machine translation have been presented by Gu et al. (2018); Amin Farajian et al. (2017); Li et al. (2018); Zhang et al. (2018). Gu et al. (2018) use the retrieved sentence pairs as extra inputs to the NMT model. Zhang et al. (2018) employ a simpler and faster retrieval method to guide neural MT where translation pieces are n-grams from retrieved target sentences. We modify Zhang et al. (2018)'s method from textual n-grams to n-grams over subtrees to exploit the code structural similarity, and propose methods to deal with complex statements and rare words.

In addition, some previous works have used subtrees in structured prediction tasks. For example, Galley et al. (2006) used them in syntax-based translation models. In Galley et al. (2006), subtrees of the input sentence's parse tree are associated with corresponding words in the output sentence.

7 Conclusion

We proposed an action subtree retrieval method at test time on top of an AST-driven neural model for generating general-purpose code. The predicted surface code is syntactically correct, and the retrieval component improves the performance of a previously state-of-the-art model. Our successful result suggests that our idea of retrieval-based generation can be potentially applied to other tree-structured prediction tasks.

Acknowledgements

We are grateful to Lucile Callebert for insightful discussions, Aldrian Obaja Muis for helpful input on early version writing, and anonymous reviewers for useful feedback. This material is based upon work supported by the National Science Foundation under Grant No. 1815287.

References

D. Alvarez-Melis and T. Jaakkola. 2017. Tree structured decoding with doubly recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

M Amin Farajian, Marco Turchi, Matteo Negri, Marcello Federico, and Fondazione Bruno Kessler. 2017. Multi-Domain Neural Machine Translation through Unsupervised Adaptation. In *Proceedings of the Second Conference on Machine Translation (WMT), Volume 1: Research Papers*, pages 127–137.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 33–34.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 731–742.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 961–968.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1631–1640.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. 2018. Search engine guided neural machine translation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*.

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

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *The 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 12–22.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings*, pages 826–836.

Tao Lei, Fan Long, Regina Barzilay, and Martin C. Rinard. 2013. From natural language specifications to program input parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1294–1303.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2018. One Sentence One Model for Neural Machine Translation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 599–609.

Nicholas Locascio, Karthik Narasimhan, Eduardo De Leon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1918–1923.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (t). In *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 574–584.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1139–1149.

Mukund Raghothaman, Yi Wei, and Youssef Hamadi. 2016. Swim: Synthesizing what i mean: Code search and idiomatic snippet synthesis. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, pages 357–367.

Mohammad Raza, Sumit Gulwani, and Natasa Milic-Frayling. 2015. Compositional program synthesis from natural language and examples. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 792–800.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 440–450.

Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding neural machine translation with retrieved translation pieces. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

SQL-to-Text Generation with Graph-to-Sequence Model

Kun Xu[1]*, **Lingfei Wu**[2], **Zhiguo Wang**[2], **Yansong Feng**[3], **Vadim Sheinin**[2]
[1]Tencent AI Lab
[2]IBM Research
[3]Peking University, Beijing, China
`{syxu828,zgw.tomorrow}@gmail.com`, `lwu@email.wm.edu`
`fengyansong@pku.edu.cn`, `vadims@us.ibm.com`

Abstract

Previous work approaches the SQL-to-text generation task using vanilla Seq2Seq models, which may not fully capture the inherent graph-structured information in SQL query. In this paper, we first introduce a strategy to represent the SQL query as a directed graph and then employ a graph-to-sequence model to encode the global structure information into node embeddings. This model can effectively learn the correlation between the SQL query pattern and its interpretation. Experimental results on the WikiSQL dataset and Stackoverflow dataset show that our model significantly outperforms the Seq2Seq and Tree2Seq baselines, achieving the state-of-the-art performance.

1 Introduction

The goal of the SQL-to-text task is to automatically generate human-like descriptions interpreting the meaning of a given structured query language (SQL) query (Figure 1 gives an example). This task is critical to the natural language interface to a database since it helps non-expert users to understand the esoteric SQL queries that are used to retrieve the answers through the question-answering process (Simitsis and Ioannidis, 2009) using varous text embeddings techniques (Kim, 2014; Arora et al., 2017; Wu et al., 2018a).

Earlier attempts for SQL-to-text task are rule-based and template-based (Koutrika et al., 2010; Ngonga Ngomo et al., 2013). Despite requiring intensive human efforts to design temples or rules, these approaches still tend to generate rigid and stylized language that lacks the natural text of the human language. To address this, Iyer et al. (2016) proposes a sequence-to-sequence (Seq2Seq) network to model the SQL query and natural language jointly. However, since the SQL is designed to express graph-structured query intent, the sequence encoder may need an elaborate design to fully capture the global structure information. Intuitively, varous graph encoding techniques base on deep neural network (Kipf and Welling, 2016; Hamilton et al., 2017; Song et al., 2018) or based on Graph Kernels (Vishwanathan et al., 2010; Wu et al., 2018b), whose goal is to learn the node-level or graph-level representations for a given graph, are more proper to tackle this problem.

(SQL): **SELECT** company **WHERE** assets > val$_0$ **AND** sales > val$_0$ **AND** industry_rank <= val$_2$ **AND** revenue = val$_3$

Interpretation:
which company has both the market value and assets higher than val$_{0,}$ ranking in top val$_2$ and revenue of val$_3$

Figure 1: An example of SQL query and its interpretation.

In this paper, we first introduce a strategy to represent the SQL query as a directed graph (see §2) and further make full use of a novel graph-to-sequence (Graph2Seq) model (Xu et al., 2018) that encodes this graph-structured SQL query, and then decodes its interpretation (see §3). On the encoder side, we extend the graph encoding work of Hamilton et al. (2017) by encoding the edge direction information into the node embedding. Our encoder learns the representation of each node by aggregating information from its K-hop neighbors. Different from Hamilton et al. (2017) which neglects the edge direction, we classify the neighbors of a node according to the edge direction, say v, into two classes, i.e., forward nodes (v directs to) and backward nodes (direct to v). We apply two distinct aggregators to aggregate the information of these two types of nodes, resulting two representations. The node embedding of v is the concatenation of these two representations. Given the learned node embeddings, we further introduce a pooling-based and an aggregation-based method to generate the graph embedding.

---

[*] Work done when the author was at IBM Research.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 931–936
Brussels, Belgium, October 31 - November 4, 2018.

On the decoder side, we develop an RNN-based decoder which takes the graph vector representation as the initial hidden state to generate the sequences while employing an attention mechanism over all node embeddings. Experimental results show that our model achieves the state-of-the-art performance on the WikiSQL dataset and Stackoverflow dataset. Our code and data is available at https://github.com/IBM/SQL-to-Text.

2 Graph Representation of SQL Query

Representing the SQL query as a graph instead of a sequence could better preserve the inherent structure information in the query. An example is illustrated in the blue dashed frame in Figure 2. One can see that representing them as a graph instead of a sequence could help the model to better learn the correlation between this graph pattern and the interpretation "*...both X and Y higher than Z...*". This observation motivates us to represent the SQL query as a graph. In particular, we use the following method to transform the SQL query to a graph:[1]

SELECT Clause. For the SELECT clause such as "SELECT company", we first create a node assigned with text attribute *select*. This SELECT node connects with column nodes whose text attributes are the selected column names such as *company*. For SQL queries that contain aggregation functions such as `count` or `max`, we add one aggregation node which is connected with column nodes. Similarly, their text attributes are the aggregation function names.

WHERE Clause. The WHERE clause usually contains more than one condition. For each condition, we use the same process as for the SELECT clause to create nodes. For example, in Figure 2, we create node *assets* and $>val_0$ for the first condition, the node *sales* and $>val_0$ for the second condition. We then integrate the constraint nodes that have the same text attribute (e.g., $>val_0$ in Figure 2). For a logical operator such as AND, OR and NOT, we create a node that connects with all column nodes that the operator works on. Finally, these logical operator nodes connect with the SELECT node.

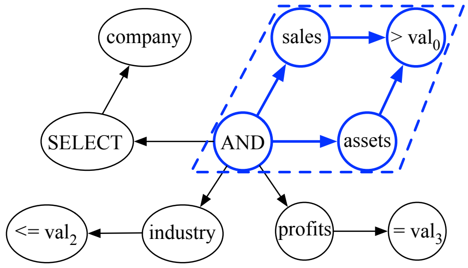

Figure 2: The graph representation of the SQL query in Figure 1.

3 Graph-to-sequence Model

Based on the constructed graphs for the SQL queries, we make full use of a novel graph-to-sequence model (Xu et al., 2018), which consists of a graph encoder to learn the embedding for the graph-structured SQL query, and a sequence decoder with attention mechanism to generate sentences. Conceptually, the graph encoder generates the node embedding for each node by accumulating information from its K-hop neighbors, and produces a graph embedding for the entire graph by abstracting all node embeddings. Our decoder takes the graph embedding as the initial hidden state and calculates the attention over all node embeddings on the encoder side to generate natural language interpretations.

Node Embedding. Given the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, since the text attribute of a node may include a list of words, we first use a Long Short Term Memory (LSTM) to generate the feature vector $\mathbf{a}_v$ for all nodes $\forall v \in \mathcal{V}$ from v's text attribute. We use these feature vectors as initial node embeddings. Then, our model incorporates information from a node's neighbors within K hop into its representation by repeating the following process K times:

$$\mathbf{h}^0_{v\vdash} = \mathbf{a}_v, \mathbf{h}^0_{\vdash v} = \mathbf{a}_v, \forall v \in \mathcal{V} \tag{1}$$

$$\mathbf{h}^k_{\mathcal{N}_\vdash(v)} = \mathbf{M}^k_\vdash(\{\mathbf{h}^{k-1}_{u\vdash}, \forall u \in \mathcal{N}_\vdash(v)\}) \tag{2}$$

$$\mathbf{h}^k_{v\vdash} = \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_{v\vdash}, \mathbf{h}^k_{\mathcal{N}_\vdash(v)})) \tag{3}$$

$$\mathbf{h}^k_{\mathcal{N}_\dashv(v)} = \mathbf{M}^k_\dashv(\{\mathbf{h}^{k-1}_{u\dashv}, \forall u \in \mathcal{N}_\dashv(v)\}) \tag{4}$$

$$\mathbf{h}^k_{v\dashv} = \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_{v\dashv}, \mathbf{h}^k_{\mathcal{N}_\dashv(v)})) \tag{5}$$

where $k \in \{1, ..., K\}$ is the iteration index, $\mathcal{N}$ is the neighborhood function[2], $\mathbf{h}^k_{v\vdash}$ ($\mathbf{h}^k_{v\dashv}$) is node v's forward (backward) representation which aggregates the information of nodes in $\mathcal{N}_\vdash(v)$ ($\mathcal{N}_\dashv(v)$),

---

[1] This method could be simply extended to cope with more general SQL queries that have complex syntaxes such as JOIN and ORDER BY.

[2] $\mathcal{N}_\vdash(v)$ returns the nodes that v directs to and $\mathcal{N}_\dashv(v)$ returns the nodes that direct to v.

$\mathbf{M}_{\vdash}^{k}$ and $\mathbf{M}_{\dashv}^{k}$ are the forward and backward aggregator functions, $\mathbf{W}^{k}$ denotes weight matrices, σ is a non-linearity function.

For example, for node $v \in \mathcal{V}$, we first aggregate the forward representations of its immediate neighbors $\{\mathbf{h}_{u\vdash}^{k-1}, \forall u \in \mathcal{N}_{\vdash}(v)\}$ into a single vector $\mathbf{h}_{\mathcal{N}_{\vdash}(v)}^{k}$ (equation 2). Note that this aggregation step only uses the representations generated at previous iteration and its initial representation is $\mathbf{a}_v$. Then we concatenate v's current forward representation $\mathbf{h}_{v\vdash}^{k-1}$ with the newly generated neighborhood vector $\mathbf{h}_{\mathcal{N}_{\vdash}(v)}^{k}$. This concatenated vector is fed into a fully connected layer with nonlinear activation function σ, which updates the forward representation of v to be used at the next iteration (equation 3). Next, we update the backward representation of v in the similar fashion (equation 4$\sim$5). Finally, the concatenation of the forward and backward representation at last iteration K, is used as the resulting representation of v. Since the neighbor information from different hops may have the different impact on the node embedding, we learn a distinct aggregator function at each step. This aggregator feeds each neighbor's vector to a fully-connected neural network and an element-wise max-pooling operation is applied to capture different aspects of the neighbor set.

Graph Embedding. Most existing works of graph convolution neural networks focus more on node embeddings rather than graph embeddings (GE) since their focus is on the node-wise classification task. However, graph embeddings that convey the entire graph information are essential to the downstream decoder, which is crucial to our task. For this purpose, we propose two ways to generate graph embeddings, namely, the Pooling-based and Node-based methods.

Pooling-based GE. This method feeds the obtained node embeddings into a fully-connected neural network and applies the element-wise *max*-pooling operation on all node embeddings. In experiments, we did not observe significant performance improvement using min-pooling and average-pooling.

Node-based GE. Following (Scarselli et al., 2009), this method adds a **super** node v_s that is connected to all other nodes by a special type of edge. The embedding of v_s, which is treated as graph embedding, is produced using node embedding generation algorithm mentioned above.

Sequence Decoding. The decoder is an RNN which predicts the next token y_i given all the previous words $y_{<i} = y_1, ..., y_{i-1}$, the RNN hidden state s_i for time-step i and the context vector c_i that captures the attention of the encoder side. In particular, the context vector c_i depends on a set of node representations ($\mathbf{h}_1$,...,$\mathbf{h}_{\mathcal{V}}$) to which the encoder maps the input graph. The context vector c_i is dynamically computed using attention mechanism over the node representations. Our model is jointly trained to maximize the conditional log-probability of the correct description given a source graph with respect to the parameters θ of the model:

$$\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(y_t^n | y_{<t}^n, x^n)$$

where (x^n, y^n) is the n-th SQL-interpretation pair in the training set, and T_n is the length of the n-th target sentence y^n. In the inference phase, we use the beam search algorithm with beam size = 5.

4 Experiments

We evaluate our model on two datasets, WikiSQL (Zhong et al., 2017) and Stackoverflow (Iyer et al., 2016). WikiSQL consists of a corpus of 87,726 hand-annotated SQL query and natural language question pairs. These SQL queries are further split into training (61,297 examples), development (9,145 examples) and test sets (17,284 examples). StackOverflow consists of 32,337 SQL query and natural language question pairs, and we use the same train/development/test split as (Iyer et al., 2016). We use the BLEU-4 score (Papineni et al., 2002) as our automatic evaluation metric and also perform a human study. For human evaluation, we randomly sampled 1,000 predicted results and asked three native English speakers to rate each interpretation against both the correctness conforming to the input SQL and grammaticality on a scale between 1 and 5. We compare some variants of our model against the template, Seq2Seq, and Tree2Seq baselines.

***Graph2Seq*-PGE.** This method uses the **P**ooling method for generating **G**raph **E**mbedding.

***Graph2Seq*-NGE.** This method uses the **N**ode based **G**raph **E**mbedding.

Template. We implement a template-based method which first maps each element of a SQL query to an utterance and then uses simple rules to assemble these utterances. For example, we

	BLEU-4	Grammar.	Correct.
Template	15.71	1.50	-
Seq2Seq	20.91	2.54	62.1%
Seq2Seq + Copy	24.12	2.65	64.5%
Tree2Seq	26.67	2.70	66.8%
Graph2Seq-PGE	**38.97**	**3.81**	**79.2%**
Graph2Seq-NGE	34.28	3.26	75.3%
(Iyer et al., 2016)	18.4	3.16	64.2%
Graph2Seq-PGE	**23.3**	**3.23**	**70.2%**
Graph2Seq-NGE	21.9	2.97	65.1%

Table 1: Results on the WikiSQL (above) and Stackoverflow (below).

SQL Query & Interpretations
1. COUNT Player WHERE starter = val$_0$ AND touchdowns = val$_1$ AND position = val$_2$
S: How many players played in position val$_2$
G: number of players with starter val$_0$ and get touchdowns val$_1$ for val$_2$
2. SELECT Tires WHERE engine = val$_0$ AND chassis = val$_1$ AND team = val$_2$
S: which tire has engine val$_0$ and chassis val$_1$ and val$_2$
G: which tire does val$_2$ run with val$_0$ engine and val$_1$ chassis

Table 2: Example of SQL queries and predicted interpretations where S and G denotes Seq2Seq and Graph2Seq models, respectively.

map SELECT to *which*, WHERE to *where*, $>$ to *more than*. This method translates the SQL query of Figure 1 to *which company where assets more than val$_0$ and sales more than val$_0$ and industry less than or equal to val$_1$ and profits equals val$_2$*.

Seq2Seq. We choose two Seq2Seq models as our baselines. The first one is the attention-based Seq2Seq model proposed by Bahdanau et al. (2014), and the second one additionally introduces the copy mechanism in the decoder side (Gu et al., 2016). To evaluate these models, we employ a template to convert the SQL query into a sequence: "SELECT + <*aggregation function*> + <Split Symbol> + <*selected column*> + WHERE + <*condition*$_0$> + <Split Symbol> + <*condition*$_1$> + ... ".

Tree2Seq. We also choose a tree-to-sequence model proposed by (Eriguchi et al., 2016) as our baseline. We use the SQL Parser tool[3] to convert a SQL query into the tree structure[4] which is fed to the Tree2Seq model.

Our proposed models are trained using the Adam optimizer (Kingma and Ba, 2014), with mini-batch size 30. Our hyper-parameters are set based on performance on the validation set. The learning rate is set to 0.001. We apply the dropout strategy (Srivastava et al., 2014) with the ratio of 0.5 at the decoder layer to avoid overfitting. Gradients are clipped when their norm is bigger than 20. We initialize word embeddings using GloVe word vectors from Pennington et al. (2014), and the word embedding dimension is 300. For the graph encoder, the hop size K is set to 6, the non-linearity function σ is implemented as ReLU (Glorot et al., 2011), the parameters of weight matrices $\mathbf{W}^k$ are randomly initialized. The decoder has one layer, and its hidden state size is 300.

Results and Discussion Table 1 summarizes the results of our models and baselines. Although the template-based method achieves decent BLEU scores, its grammaticality score is substantially worse than other baselines. We can see that on both two datasets, our Graph2Seq models perform significantly better than the Seq2Seq and Tree2Seq baselines. One possible reason is that in our graph encoder, the node embedding retains the information of neighbor nodes within K hops. However, in the tree encoder, the node embedding only aggregates the information of descendants while losing the knowledge of ancestors. The pooling-based graph embedding is found to be more useful than the node-based graph embedding because *Graph2Seq*-NGE adds a nonexistent node into the graph, which introduces the noisy information in calculating the embeddings of other nodes. We also conducted an experiment that treats the SQL query graph as an undirected graph and found the performance degrades.

By manually analyzing the cases in which the Graph2Seq model performs better than Seq2Seq, we find the Graph2Seq model is better at interpreting two classes of queries: (1) the complicated queries that have more than two conditions (Query 1); (2) the queries whose columns have implicit relationships (Query 2). Table 2 lists some such SQL queries and their interpretations. One possible reason is that the Graph2Seq model can better learn the correlation between the graph pattern and natural language by utilizing the global structure information.

We find the hop size has a significant impact on our model since it determines how many neighbor nodes to be considered during the node embedding generation. As the hop size increasing, the performance is found to be significantly improved. However, after the hop size reaches 6,

---

[3] `http://www.sqlparser.com`

[4] See Appendix for details.

increasing the hop size can not boost the performance on WikiSQL anymore. By analyzing the most complicated queries (around 6.2%) in WikiSQL, we find there are average six hops between a node and its most distant neighbor. This result indicates that the selected hop size should guarantee each node can receive the information from others nodes in the graph.

5 Conclusions

Previous work approaches the SQL-to-text task using an Seq2Seq model which does not fully capture the global structure information of the SQL query. To address this, we proposed a Graph2Seq model which includes a graph encoder, an attention based sequence decoder. Experimental results show that our model significantly outperforms the Seq2Seq and Tree2Seq models on the WikiSQL and Stackoverflow datasets.

Appendix

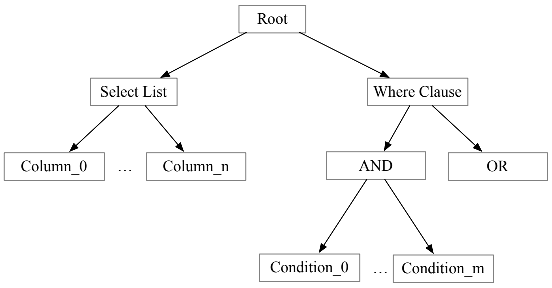

Figure 3: Tree representation of the SQL query.

We apply the SQL Parser tool[5] to convert an SQL query to a tree whose structure is illustrated in Figure 3. More specifically, the root has two child nodes, namely *Select List* and *Where Clause*. The child nodes of *Select List* represent the selected columns in the SQL query. The *Where Clause* has the logical operators occurred in the SQL query as its children. The children of a logical operator node are the conditions on which this operator works.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. *arXiv preprint arXiv:1603.06075*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 315–323.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1025–1035.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2073–2083.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Georgia Koutrika, Alkis Simitsis, and Yannis E Ioannidis. 2010. Explaining structured queries in natural language. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 333–344. IEEE.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak sparql: translating sparql queries into natural language. In *Proceedings of the 22nd international conference on World Wide Web*, pages 977–988. ACM.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

---

[5] `http://www.sqlparser.com`

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Alkis Simitsis and Yannis Ioannidis. 2009. Dbmss should talk back too. *arXiv preprint arXiv:0909.1786*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. *arXiv preprint arXiv:1805.02473*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242.

Lingfei Wu, Ian E.H. Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J. Witbrock. 2018a. Word mover's embedding: From word2vec to document embedding. In *EMNLP*.

Lingfei Wu, Ian En-Hsu Yen, Fangli Xu, Pradeep Ravikuma, and Michael Witbrock. 2018b. D2ke: From distance to kernel and embedding. *arXiv preprint arXiv:1802.04956*.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Generating Syntactic Paraphrases

Emilie Colin
Lorraine University/LORIA
Nancy, France
emilie.colin@loria.fr

Claire Gardent
CNRS/LORIA
Nancy, France
claire.gardent@loria.fr

Abstract

We study the automatic generation of syntactic paraphrases using four different models for generation: data-to-text generation, text-to-text generation, text reduction and text expansion, We derive training data for each of these tasks from the WebNLG dataset and we show (i) that conditioning generation on syntactic constraints effectively permits the generation of syntactically distinct paraphrases for the same input and (ii) that exploiting different types of input (data, text or data+text) further increases the number of distinct paraphrases that can be generated for a given input.

1 Introduction

The ability to automatically generate paraphrases (alternative phrasings of the same content) has been shown to be useful in many areas of Natural Language Processing such as question answering (Riezler et al., 2007), semantic parsing (Berant and Liang, 2014)), machine translation (Kauchak and Barzilay, 2006; Zhou et al., 2006), sentence compression (Napoles et al., 2011) and sentence representation (Wieting et al., 2015). From a linguistic standpoint, the automatic generation of paraphrases is an important task in its own right as it demonstrates the capacity of NLP techniques to handle a key feature of natural language.

In this paper, we focus on the automatic generation of *syntactic paraphrases* that is, texts which share the same meaning but differ in their syntax. Our work makes the following contributions. We show that conditioning text generation on syntactic information permits generating distinct syntactic paraphrases for the same input. We provide a systematic exploration of how different types of generation tasks impact paraphrasing and show that exploiting different types of input permits increasing the number of paraphrases produced for a given input. We make available four training corpora for syntactically constrained, data- and text-to-text generation, text expansion and text reduction.

2 Related Work

Previous work on paraphrase generation falls into three main groups. Based mainly on monolingual data, earlier approaches use data-driven, (Lin and Pantel, 2001), grammar- or thesaurus-based methods (Madnani et al., 2007; McKeown, 1983; Hassan et al., 2007; Kozlowski et al., 2003; Quirk et al., 2004; Zhao et al., 2008). In contrast, the pivot-based approach exploits bilingual data and machine translation methods to extract and generate paraphrases (Callison-Burch, 2008; Ganitkevitch and Callison-Burch, 2014; Ganitkevitch et al., 2011). Finally, neural approaches build upon the encoder-decoder architecture to learn paraphrase generation models (Mallinson et al., 2017; Prakash et al., 2016).

(Prakash et al., 2016) uses a stacked residual LSTM network with residual connections between LSTM layers and show that their model outperforms sequence to sequence, attention-based, and bi- directional LSTM model on three datasets (PPDB, WikiAnswers, and MSCOCO).

(Mallinson et al., 2017) introduces a neural model for multi-lingual, multi-pivot backtranslation and show that it outperforms a paraphrase model trained with a commonly used Statistical Machine Translation system (SMT) on three tasks, namely, correlation with human judgments of paraphrase quality; paraphrase and similarity detection; and sentence-level paraphrase generation.

(Iyyer et al., 2018) also use backtranslation as a mean to provide training data. In addition, it uses syntax to control paraphrase generation. Given

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 937–943
Brussels, Belgium, October 31 - November 4, 2018.

D2T$_{5best}$
Aktieselskab is the operating organisation for Aarhus Airport which has a runway name of ”10R/28L” with a length of 2777.
The Aktieselskab is the operating organisation for Aarhus Airport which has a runway name of ”10R/28L” with a length of 2777.
Aktieselskab is the operating organisation for Aarhus Airport which has a runway length of 2777 and is named ”10R/28L”.

T2T$_{5best}$
Operated by Aktieselskab, Aarhus Airport has a runway length of 2777 metres and a runway name ”10R/28L”.
Operated by the Aktieselskab organisation, Aarhus Airport, has a runway length of 2777 metres and a runway named ”10R/28L”.
Aarhus Airport is operated by Aktieselskab. Its runway name is ”10R/28L” and its length is 2777 metres.
Aarhus Airport, operated by Aktieselskab, has a runway length of 2777 with a name of ”10R/28L”.
Aktieselskab is the operation organisation of Aarhus Airport where the runway ”10R/28L” with a length of 2777.
The Aktieselskab is the operating organisation of Aarhus Airport., This airport has a runway length of 2777 metres and a runway named ”10R/28L”.
The ”10R/28L” runway at Aarhus Airport is 2777 in length, which is run by the operating organization of Aktieselskab.
The ”10R/28L” runway at Aarhus Airport, operated by the Aktieselskab, is 2777 in length.

ALL$_{syn}$
Aarhus Airport has a runway length of 2777.0 metres and is operated by Aktieselskab. The name of the runway is ”10L/28R”.
Aarhus Airport is operated by Aktieselskab; its runway name is ”10L/28R” and its runway length is 2777.0.
Aarhus Airport is operated by Aktieselskab, its runway name is ”10R/28L” and has a length of 2777.
Aarhus Airport is operated by the Aktieselskab organisation. Its runway name is ”10R/28L” and has a length of 2777.
Aarhus Airport, operated by Aktieselskab, has a runway length of 2777 and the runway name is ”10R/28L”.
Aarhus Airport, operated by Aktieselskab, has a runway length of 2777 with a name of ”10R/28L”.
Aarhus Airport, which is operated by the Aktieselskab organisation, has a runway that ’s 2777.0 long and is named ”10L/28R”.
Aktieselskab is the operation organisation of Aarhus Airport where the runway ”10R/28L” with a length of 2777.
Aktieselskab is the operation organisation of Aarhus Airport, where the runway is named ”10R/28L”, with a length of 2777.
Aktieselskab is the operation organization for Aarhus Airport, where the runway is named ”10L/28R”, with a length of 2777.0.
Aktieselskab operates Aarhus Airport which has a runway that is 2777 meters long and the runway name ”10R/28L”.
Operated by Aktieselskab, Aarhus Airport, has a runway length of 2777 metres and a runway named ”10R/28L”.
Operated by Aktieselskab, Aarhus Airport, has a runway length of 2777 metres and is named ”10R/28L”.
Operated by the Aktieselskab organisation, Aarhus Airport has a runway that is 2777.0 metres long. It also has a runway with the name ”10L/28R”.
The ”10R/28L” runway which is 2777 meters long is located in Aarhus Airport which is operated by the Aktieselskab organisation.

Table 1: Some Example Output

TX$_{syn}$		$T_i, k, t \Rightarrow T_o$ with $mg(T_o) = mg(T_i) \cup \{t\}, k \in K(T)$
	T_i	*Madrid is part of Community of Madrid whose leader party is the Ahora Madrid. The Adolfo Suarez Madrid-Barajas Airport is located there.*
	k	possessive
	t	{ (Madrid country Spain) }
	T_o	*Adolfo Suarez Madrid-Barajas Airport is located in Madrid, part of the Community of Madrid in Spain where the leader party is Ahora Madrid.*
TR$_{syn}$		$T_i, k, t \Rightarrow T_o$ with $mg(T_i) = mg(T_o) \cup \{t\}, k \in K(T)$
	T_i	*Al Asad Airbase is located at ”Al Anbar Province, Iraq” and operated by the United States Air Force. The base ’s runway called ”08/26” and 3990 meters long.*
	k	Subject Relative
	t	{ (Al Asad Airbase operatingOrganisation United States Air Force) }
	T_o	*Al Asad Airbase (in ”Al Anbar Province, Iraq”), has a runway named ”08/26” and a runway that is 3990 metres long.*

Table 2: TR: Text Reduction, T_i, T_o: input/output text, k: syntactic constraint, M: meaning representation, a set of RDF triples, $mg(T)$: meaning of text T, $K(T)$: syntactic constraints realised by text T).

a syntactic template T and an input sentence S, the model first generates a full syntactic parse P_T. Next this syntactic parse is used together with the input sentence to predict a syntactic paraphrase of S which realises the input syntactic template T.

Our approach is closest to (Iyyer et al., 2018) but differs from it in that instead of restricting paraphrase generation to a text rewriting problem, we explore how various sources of input impacts the number and the type of generated paraphrases. It also differs from the former two approaches (Prakash et al., 2016; Mallinson et al., 2017) in that we focus on syntactic paraphrases and condition generation on syntax. In that sense, our approach also shares similarities with recent models for controllable text generation (Hu et al., 2017; Semeniuta et al., 2017), which use variational autoencoders to model holistic properties of sentences such as style, topic and various other syntactic features. Our work is arguably conceptually simpler, focuses on syntactic paraphrases and introduces a new text production mode based on hybrid "data and text" input.

3 Generating Syntactic Paraphrases

In order to generate syntactically distinct paraphrases, we formulate the generation task as a structured prediction task conditioned on both some input I and some syntactic constraint k. In this way, the same input I can be mapped to several output T_i each satisfying a different syntactic constraint k_i. Table 1 shows some examples.

In addition, we consider different, semantically equivalent, sources of information. That is, we compare the paraphrases obtained when generating text from data, from text or from text and data. For the later, we consider two subtasks namely text expansion and text reduction. For each of these two tasks, the input is a text and a data unit. For text expansion, the output is a text verbalising both the input text and the input data. Conversely, for text reduction, the output is a text verbalising the input text minus the text verbalising the input data. Table 2 shows some example input and output for text expansion and text reduction.

4 Training and Test Data

Training data. The WEBNLG dataset (Gardent et al., 2017) associates sets of RDF triples with one or more texts verbalising these sets of triples. We derive training corpora for syntactically constrained generation from this dataset as follows.

We enrich the WEBNLG texts with labels indicating syntactic structures that are realised by these texts by first, parsing[1] these texts and then using syntactic templates to identify the target structures occurring in those texts. We use the following list of syntactic labels: subject relative, object relative, sentence coordination, VP coordination, passive voice, apposition, possessive relative, pied piping, transitive clause, prepositional object, ditransitive clause, predicative clause.

Based on the resulting, syntactically enriched, WEBNLG corpus, we then build four training corpora (T2T_{syn}, TX_{syn}, D2T_{syn}, TR_{syn}) using the sets of RDF triples as pivots to relate paraphrases. For data-to-text generation (D2T_{syn}), the input is a linearised and delexicalised version of the set of RDF triples representing the meaning of the output text, for text-to-text generation (T2T_{syn}), the input is a text and for hybrid data-and-text-to-text generation (TX_{syn} and TR_{syn}), the input is a text and a linearised RDF triple.

For the text-to-text datasets, we additionally require that, for any corpus instance $\langle k, T_i, T_o \rangle$, T_o differs from T_i on exactly one syntactic label[2].

Test data. For any input $\langle k, I \rangle$ occuring in the test data, we ensure that $\langle k, I \rangle$ does not occur in the training data. (where I is either a set of RDF triples, a text or a text and an RDF triple).

5 Experimental Setup

Models and Baselines D2T_{5best} and T2T_{5best} For each generation task, we aim to learn a model that maximises the likelihood $P(T|I;k;\theta)$ of a text given some input I, some model parameters θ and some syntactic constraint k. We use a simple encoder-decoder model where both encoder and decoder are bidirectional LSTMs and the encoder receives as input a sequence including both the input I and the syntactic constraint k.

We compare our models with the output produced by beam search when no syntactic constraint applies. For D2T_{5best}, we take the 5 best output generated from data. For T2T_{5best}, there may be several input sentences associated with the same meaning: we take the 5 best output for each

---

[1] We used the Stanford CoreNLP dependency parser version 3.8, 2018-06-09

[2] $K(T_i) = (K(T_i) \cap K(T_o)) \cup \{k\}$ and $K(T_o) = (K(T_i) \cap K(T_o)) \cup \{k'\}$ for some $k \neq k'$.

of these sentences hence T2T$_{5best}$ may (and does) in fact yield more than 5 output per input meaning. Finally, ALL$_{syn}$ groups together all output generated by the four syntactically constrained models for a given meaning.

Implementation Details We use the OpenNMT$_{py}$ sequence-to-sequence model (Klein et al., 2017) with attention and a bidirectional LSTM encoder. The encoder and decoder have two layers. Models were trained for 13 epochs, with a mini-batch size of 64, a dropout rate of 0.3, and a word embedding size of 500. They were optimised with SGD with a starting learning rate of 1.0.

Evaluation. We assess both the linguistic/syntactic adequacy of the generated texts and the diversity of the paraphrases being generated.

Syntactic and Linguistic Adequacy (BLEU, Synt, BLEU$_{syn}$). For the syntactically constrained models, given an input syntactic constraint k, the BLEU score[3] is computed with respect to those references which satisfy k. In that way, the BLEU score indicates how close to the syntactic target the generated sentence is and therefore how well the model succeeds in generating the required syntactic constructs – as the number of references varies across inputs, we use BLEU at the sentence level (Papineni et al., 2002). In addition, we compute the proportion of output satisfying the input syntactic constraint (Synt) and the BLEU score for these output which satisfy the input syntactic constraint (BLEU$_{syn}$). The number of output satisfying the input syntactic constraint is computed by first parsing the generated output and then applying the templates used for the automatic annotation of the training data.

Diversity (Sim, #Txt/Mg). To measure the level of paraphrasing obtained, we group together inputs which share the same meaning (i.e., inputs that are linked in the WEBNLG dataset to the same set of RDF triples) and we compute the number of distinct texts generated per meaning (# Txt/Mg). We further analyse these sets by computing the average pairwise similarity (Sim) of the texts present in these sets. We use the Ratcliff/Obershelp algorithm (Black, 2004) to compute similarity[4]. A low similarity indicates more diversity across the set of outputs sharing the same meaning.

Human Evaluation (% SPar). For each model, we manually examined for 50 meanings, a maximum of 10 randomly chosen output and recorded the average number (# SPar) of syntactically correct paraphrases per input.

6 Results

Table 3 summarises the results.

Diversity. The results for ALL$_{syn}$ (aggregating all output texts generated for a given meaning) shows that combining different generation models increases diversity (# Txt/Mg:13.25, Sim:0.61)) while maintaining a good level of linguistic (BLEU:62.87) and syntactic adequacy (Synt:0.91).

The human evaluation further shows that the distinct outputs generated by the ALL$_{syn}$ model are indeed syntactic, not purely lexical, variants. Table 1 shows some example output for ALL$_{syn}$.

Expansion, Reduction and Generation. Interestingly, the text expansion and reduction models markedly improve on traditional T2T and D2T models both in terms of linguistic adequacy (higher BLEU score) and in terms of diversity (higher number of distinct output per meaning, lower similarity between texts generated from the same meaning). The comparison with T2T generation is particularly striking as the training data is 3 to 5 times larger for the T2T$_{syn}$ model than for the TX$_{syn}$ and the TR$_{syn}$ model respectively. Similarly, it is noticable that although the T2T$_{syn}$ training corpus is 3 times larger than the D2T$_{syn}$ corpus, the T2T$_{syn}$ and the D2T$_{syn}$ models show similar results. This is in line with results from (Aharoni and Goldberg, 2018) which shows that rephrasing is a difficult task.

Linguistic Adequacy. Overall the linguistic adequacy of the syntactically constrained models is high with a BLEU score with respect to a single reference ranging from 46.20 (D2T$_{syn}$) to 83.87 (TX$_{syn}$). Moreover, the generated sentences show close similarity with the reference sentence realising the input constraint (BLEU$_{syn}$: from 48.16 to 89.32).

---

[3] We use the sacrebleu script with BLEU-4.

[4] The Ratcliff/Obershelp similarity score varies between O and 1 where 1 is a complete match. It is expressed by the formula $sim(S1, S2) = \frac{2*match(S1,S2)}{len(S1)+len(S2)}$ where a match is defined as the sum of the length of the matching segments ($match(S1, S2) = \sum_{m \in overlap(S1,S2)} len(m)$.

Model	BLEU	Synt	BLEU$_{syn}$	Sim test	# Txt/Mg	# Corpora	# SPar/Mg
	-	-	-	(vs ref)	(# txt/Mg ref)		(# outputs)
T2T$_{5best}$	6.21	N/A	N/A	0.76 (0.61)	5.85 (44.98)	715910	3.98 (5.5)
D2T$_{5best}$	4.71	N/A	N/A	0.81 (0.63)	4.71 (5.98)	27156	1.86 (4.42)
TR$_{syn}$	66.63	0.70	80.45	0.62 (0.63)	2.92 (6.10)	40936	2.56 (3.9)
TX$_{syn}$	83.87	0.88	89.32	0.68 (0.68)	2.56 (6.55)	74844	0.92 (1.16)
T2T$_{syn}$	49.95	0.98	50.23	0.72 (0.68)	1.92 (18.49)	202218	1.58 (1.70)
D2T$_{syn}$	46.20	0.84	48.16	0.81 (0.61)	1.04 (3.87)	66595	1 (1.1)
ALL$_{syn}$	62.87	0.91	65.11	0.61 (0.62)	13.25 (68.95)	NA	6.3 (15)

Table 3: Results and Corpora Size (BLEU score wrt reference satisfying the input constraint k, Synt: proportion of output satisfying k; BLEU$_{syn}$: BLEU score for output satisfying k; Sim: average pair-wise similarity between sentences output for a given (data or text) input (in brackets: the similarity score calculated on the reference corpus); # Txt/Mg: avg nb of distinct text generated per meaning (in brackets: the average number of texts per meaning occurring in the reference data); # Corpora : size of the corpora; # SPar/Mg (%): Avg number of syntactic paraphrases per meaning found by human evaluation (in brackets: the average total number of output considered)

While the baseline models underperform in terms of BLEU scores, the manual evaluation (# SPar/Mg) indicates that they, in fact, produce acceptable output. The low BLEU scores for these models are probably due to the fact that each output is evaluated against a single reference while the dataset is constructed to maximise the number of paraphrases available for a given input.

7 Some examples

Table 1 shows some example outputs illustrating the main differences between the D2T$_{5best}$, T2T$_{5best}$ and the ALL$_{syn}$ model. As these examples show, syntactically constrained generation (ALL$_{syn}$) outputs a much larger number of paraphrases. The difference is due both to the fact that ALL$_{syn}$ groups together the output of 4 (syntactically driven) generation models and to the input syntactic constraint, which ensures greater diversity. Thus in the example shown, ALL$_{syn}$ yields 15 paraphrases each with strong syntactic differences as summarised below.

Sentence Segmentation. The number of verb phrases, clauses and sentences used to verbalise the same input varies. One output text is made of 2 sentences and one VP coordination, another of 3 coordinated clauses and a third of two coordinated clauses and a VP coordination.

Syntax. The same input property is realised by different syntactic structures. For instance, the property *operatingOrganisation* is alternatively realised by an active verb (*operates*), a passive verb (*is operated by*), a participial apposition (*,operated by ..,*), a subject relative (*which is operated by*), a nominal predicative construction (*is the operation organization*) and a preposed participial (*Operated by .., *).

Word Order. The same content is verbalised using varying word order and clause ordering. Thus the ALL$_{syn}$ output shows four different ways of ordering the realisation of the three properties *operatinOrganization (oO), runwayLength (rL), runwayName (rN)* contained in the input namely, *rL-oO-rN* (once), *oO-rN-rL* (6 times), *oO-rL-rN* (6 times) and *rN-rL-oO* (once).

By constrast, the baseline models output a much smaller range of syntactic paraphrases. The D2T$_{5best}$ model is particularly weak as among the five best outputs it produces, only three are distinct and all have almost identical syntax. The T2T$_{5best}$ model produces more outputs (8 against 3 for the D2T$_{5best}$ model and 15 for the ALL$_{syn}$ model). One reason for this is that, contrary to the D2T$_{5best}$ model which has a single input (namely a set of RDF triples), this model can have several inputs for the same set of RDF triples.

8 Conclusion

We have proposed new syntactically constrained models for text generation and shown that their use effectively supports the generation of syntactic paraphrases. In future work, we plan to investigate to what extent these methods can be used to support the automatic generation of grammar exercises.

References

Roee Aharoni and Yoav Goldberg. 2018. Split and rephrase: Better evaluation and a stronger baseline. In *ACL*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1415–1425.

Paul E Black. 2004. Ratcliff/obershelp pattern recognition. *Dictionary of Algorithms and Data Structures*, 17.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 196–205. Association for Computational Linguistics.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *LREC*, pages 4276–4283.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1168–1179. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planning. In *55th annual meeting of the Association for Computational Linguistics (ACL)*.

Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. Unt: Subfinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 410–413. Association for Computational Linguistics.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Controllable text generation. *arXiv preprint arXiv:1703.00955*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885. Association for Computational Linguistics.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 455–462. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Raymond Kozlowski, Kathleen F McCoy, and K Vijay-Shanker. 2003. Generation of single-sentence paraphrases from predicate/argument structure using lexico-grammatical resources. In *Proceedings of the second international workshop on Paraphrasing-Volume 16*, pages 1–8. Association for Computational Linguistics.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360.

Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie J Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 120–127. Association for Computational Linguistics.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 881–893.

Kathleen R McKeown. 1983. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10.

Courtney Napoles, Chris Callison-Burch, Juri Ganitkevitch, and Benjamin Van Durme. 2011. Paraphrastic sentence compression with a character-based metric: Tightening without deletion. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 84–90. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Laura Perez-Beltrachini, Claire Gardent, and German Kruszewski. 2012. Generating grammar exercises. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 147–156. Association for Computational Linguistics.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*.

Chris Quirk, Chris Brockett, and Bill Dolan. 2004. Monolingual machine translation for paraphrase generation.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 464–471.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. *arXiv preprint arXiv:1702.02390*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *arXiv preprint arXiv:1506.03487*.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008. Combining multiple resources to improve smt-based paraphrasing model. *Proceedings of ACL-08: HLT*, pages 1021–1029.

Liang Zhou, Chin-Yew Lin, and Eduard Hovy. 2006. Re-evaluating machine translation results with paraphrase support. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 77–84. Association for Computational Linguistics.

Neural-Davidsonian Semantic Proto-role Labeling

Rachel Rudinger
Johns Hopkins University

Adam Teichert
Johns Hopkins Univeristy

Ryan Culkin
Johns Hopkins University

Sheng Zhang
Johns Hopkins University

Benjamin Van Durme
Johns Hopkins University

Abstract

We present a model for semantic proto-role labeling (SPRL) using an adapted bidirectional LSTM encoding strategy that we call *Neural-Davidsonian*: predicate-argument structure is represented as pairs of hidden states corresponding to predicate and argument head tokens of the input sequence. We demonstrate: (1) state-of-the-art results in SPRL, and (2) that our network naturally shares parameters between attributes, allowing for learning new attribute types with limited added supervision.

1 Introduction

Universal Decompositional Semantics (UDS) (White et al., 2016) is a contemporary semantic representation of text (Abend and Rappoport, 2017) that forgoes traditional inventories of semantic categories in favor of bundles of simple, interpretable properties. In particular, UDS includes a practical implementation of Dowty's theory of *thematic proto-roles* (Dowty, 1991): arguments are labeled with properties typical of Dowty's *proto-agent* (AWARENESS, VOLITION ...) and *proto-patient* (CHANGED STATE ...).

Annotated corpora have allowed the exploration of *Semantic Proto-role Labeling* (SPRL) [1] as a natural language processing task (Reisinger et al., 2015; White et al., 2016; Teichert et al., 2017). For example, consider the following sentence, in which a particular pair of predicate and argument heads have been emphasized: "The cat *ate* the rat." An SPRL system must infer from the context of the sentence whether the rat had VOLITION, CHANGED-STATE, and EXISTED-AFTER the *eating* event (see Table 2 for more properties).

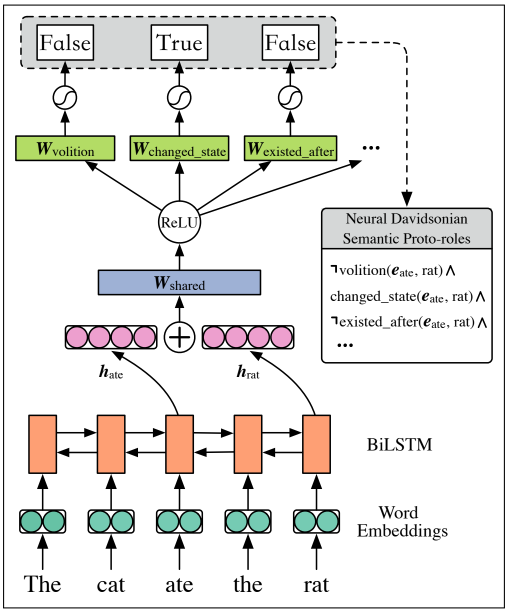

Figure 1: BiLSTM sentence encoder with SPR decoder. Semantic proto-role labeling is with respect to a specific predicate and argument within a sentence, so the decoder receives the two corresponding hidden states.

We present an intuitive neural model that achieves state-of-the-art performance for SPRL.[2] As depicted in Figure 1, our model's architecture is an extension of the bidirectional LSTM, capturing a Neo-Davidsonian like intuition, wherein select pairs of hidden states are concatenated to yield a dense representation of predicate-argument structure and fed to a prediction layer for end-to-end training. We include a thorough quantitative analysis highlighting the contrasting errors between the proposed model and previous (non-neural) state-of-the-art.

In addition, our network naturally shares a subset of parameters between attributes. We demonstrate how this allows learning to predict new at-

---

[1] SPRL and SPR refer to the labeling task and the underlying semantic representation, respectively.

[2] Implementation available at `https://github.com/decomp-sem/neural-sprl`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 944–955
Brussels, Belgium, October 31 - November 4, 2018.

SPR Property	Explanation of Property	🐈	🐀
INSTIGATION	`Arg` caused the `Pred` to happen?	✓	✗
VOLITIONAL	`Arg` chose to be involved in the `Pred`?	✓	✗
AWARE	`Arg` was/were aware of being involved in the `Pred`?	✓	✓
PHYSICALLY EXISTED	`Arg` existed as a physical object?	✓	✓
EXISTED AFTER	`Arg` existed after the `Pred` stopped?	✓	✗
CHANGED STATE	The `Arg` was/were altered or somehow changed during or by the end of the `Pred`?	✓	✓

Table 1: Example SPR annotations for the toy example "The cat ate the rat," where the `Predicate` in question is "*ate*" and the `Argument` in question is either "cat" or "rat." Note that not all SPR properties are listed, and the binary labels (✓, ✗) are coarsened from a 5-point Likert scale.

tributes with limited supervision: a key finding that could support efficient expansion of new SPR attribute types in the future.

2 Background

Davidson (1967) is credited for representations of meaning involving propositions composed of a fixed arity predicate, all of its core arguments arising from the natural language syntax, and a distinguished event variable. The earlier example could thus be denoted (modulo tense) as $(\exists e)\mathbf{eat}[(e, \textsc{cat}, \textsc{rat})]$, where the variable e is a *reification* of the eating event. The order of the arguments in the predication implies their role, where leaving arguments unspecified (as in "The cat eats") can be handled either by introducing variables for unstated arguments, e.g., $(\exists e)(\exists x)[\mathbf{eat}(e, \textsc{cat}, x)]$, or by creating new predicates that correspond to different arities, e.g., $(\exists e)\mathbf{eat\_intransitive}[(e, \textsc{cat})]$.[3] The Neo-Davidsonian approach (Castañeda, 1967; Parsons, 1995), which we follow in this work, allows for variable arity by mapping the argument positions of individual predicates to generalized *semantic roles*, shared across predicates,[4] e.g., AGENT, PATIENT and THEME, in: $(\exists e)[\mathbf{eat}(e) \wedge \mathbf{Agent}(e, \textsc{cat}) \wedge \mathbf{Patient}(e, \textsc{rat})]$.

Dowty (1991) conjectured that the distinction between the role of a prototypical **Agent** and prototypical **Patient** could be decomposed into a number of semantic properties such as *"Did the argument change state?"*. Here we formulate this as a Neo-Davidsonian representation employing *semantic proto-role* (SPR) attributes:

$$\begin{aligned}&(\exists e)\,[\mathbf{eat}(e)\\ &\qquad \wedge\, \mathbf{volition}(e, \textsc{cat}) \wedge \mathbf{instigation}(e, \textsc{cat})\ldots\\ &\qquad \wedge\, \neg\mathbf{volition}(e, \textsc{rat}) \wedge \mathbf{destroyed}(e, \textsc{rat})\ldots\,]\end{aligned}$$

Dowty's theory was empirically verified by Kako (2006), followed by pilot (Madnani et al., 2010) and large-scale (Reisinger et al., 2015) corpus annotation efforts, the latter introducing a logistic regression baseline for SPRL. Teichert et al. (2017) refined the evaluation protocol,[5] and developed a CRF (Lafferty et al., 2001) for the task, representing existing state-of-the-art.

Full details about the SPR datasets introduced by Reisinger et al. (2015) and White et al. (2016), which we use in this work, are provided in Appendix B. For clarity, Table 1 shows a toy SPRL example, including a few sample SPR properties and explanations.

3 "Neural-Davidsonian" Model

Our proposed SPRL model (Fig. 1) determines the value of each attribute (e.g., VOLITION) on an *argument* (a) with respect to a particular *predication* (e) as a function on the latent states associated with the pair, (e, a), in the context of a full sentence. Our architecture encodes the sentence using a shared, one-layer, bidirectional LSTM (Hochreiter and Schmidhuber, 1997; Graves et al., 2013). We then obtain a continuous, vector representation $\boldsymbol{h}_{ea} = [\boldsymbol{h}_e; \boldsymbol{h}_a]$, for each predicate-argument pair as the concatenation of the hidden BiLSTM

---

[3] This formalism aligns with that used in PropBank (Palmer et al., 2005), which associated numbered, core arguments with each sense of a verb in their corpus annotation.

[4] For example, as seen in FrameNet (Baker et al., 1998).

[5] Splitting train/dev/test along Penn Treebank boundaries and casting the SPRL task as multi-label binary classification.

states $\boldsymbol{h}_e$ and $\boldsymbol{h}_a$ corresponding to the syntactic head of the predicate of e and argument a respectively. These heads are obtained over gold syntactic parses using the predicate-argument detection tool, PredPatt (White et al., 2016).[6]

For each SPR attribute, a score is predicted by passing $\boldsymbol{h}_{ea}$ through a separate two-layer perceptron, with the weights of the first layer shared across all attributes:

$$\text{Score}(\text{attr}, \boldsymbol{h}_{ea}) = \boldsymbol{W}_{\text{attr}} \left[g \left(\boldsymbol{W}_{\text{shared}} \left[\boldsymbol{h}_{ea} \right] \right) \right]$$

This architecture accomodates the definition of SPRL as multi-label binary classification given by Teichert et al. (2017) by treating the score as the log-odds of the attribute being present (i.e. $\text{P}(\text{attr}|\boldsymbol{h}_{ea}) = \frac{1}{1+\exp[-\text{Score}(\text{attr},\boldsymbol{h}_{ea})]}$). This architecture also supports SPRL as a *scalar* regression task where the parameters of the network are tuned to directly minimize the discrepancy between the predicted score and a reference scalar label. The loss for the binary and scalar models are negative log-probability and squared error, respectively; the losses are summed over all SPR attributes.

Training with Auxiliary Tasks A benefit of the shared neural-Davidsonian representation is that it offers many levels at which multi-task learning may be leveraged to improve parameter estimation so as to produce semantically rich representations $\boldsymbol{h}_{ea}$, $\boldsymbol{h}_e$, and $\boldsymbol{h}_a$. For example, the sentence encoder might be pre-trained as an encoder for machine translation, the argument representation $\boldsymbol{h}_a$ can be jointly trained to predict word-sense, the predicate representation, $\boldsymbol{h}_e$, could be jointly trained to predict factuality (Saurí and Pustejovsky, 2009; Rudinger et al., 2018), and the predicate-argument representation, $\boldsymbol{h}_{ea}$, could be jointly trained to predict other semantic role formalisms (e.g. PropBank SRL—suggesting a neural-Davidsonian *SRL* model in contrast to recent BIO-style neural models of SRL (He et al., 2017)).

To evaluate this idea empirically, we experimented with a number of multi-task training strategies for SPRL. While all settings outperformed prior work in aggregate, simply initializing the BiLSTM parameters with a pretrained English-to-French machine translation encoder[7] produced the best results,[8] so we simplify discussion by focusing on that model. The efficacy of MT pretraining that we observe here comes as no surprise given prior work demonstrating, e.g., the utility of bitext for paraphrase (Ganitkevitch et al., 2013), that NMT pretraining yields improved contextualized word embeddings[9] (McCann et al., 2017), and that NMT encoders specifically capture useful features for SPRL (Poliak et al., 2018).

Full details about each multi-task experiment, including a full set of ablation results, are reported in Appendix A; details about the corresponding datasets are in Appendix B.

Except in the ablation experiment of Figure 2, our model was trained on only the SPRL data and splits used by Teichert et al. (2017) (learning all properties jointly), using GloVe[10] embeddings and with the MT-initialized BiLSTM. Models were implemented in PyTorch and trained end-to-end with Adam optimization (Kingma and Ba, 2014) and a default learning rate of 10^{-3}. Each model was trained for ten epochs, selecting the best-performing epoch on dev.

Prior Work in SPRL We additionally include results from prior work: "LR" is the logistic-regression model introduced by Reisinger et al. (2015) and "CRF" is the CRF model (specifically `SPRL`*) from Teichert et al. (2017). Although White et al. (2016) released additional SPR annotations, we are unaware of any benchmark results on that data; however, our multi-task results in Appendix A do use the data and we find (unsurprisingly) that concurrent training on the two SPR datasets can be helpful. Using only data and splits from White et al. (2016), the scalar regression architecture of Table 6 achieves a Pearson's ρ of 0.577 on test.

There are a few noteworthy differences between our neural model and the CRF of prior work. As an adapted BiLSTM, our model easily ex-

---

[6] Observed to be state-of-the-art by Zhang et al. (2017).

[7] using a modified version of `OpenNMT-py` (Klein et al., 2017) trained on the 10^9 Fr-En corpus (Callison-Burch et al., 2009) (Appendix A).

[8] e.g. this initialization resulted in raising micro-averaged F1 from 82.2 to 83.3

[9] More recent discoveries on the usefulness of language model pretraining (Peters et al., 2018; Howard and Ruder, 2018) for RNN encoders suggest a promising direction for future SPRL experiments.

[10] 300-dimensional, uncased; `glove.42B.300d` from `https://nlp.stanford.edu/projects/glove/`; 15,533 out-of-vocabulary words across all datasets were assigned a random embedding (uniformly from $[-.01, .01]$). Embeddings remained fixed during training.

	previous work		this work	
	LR	CRF	binary	scalar
instigation	76.7	85.6	**88.6**	0.858
volition	69.8	86.4	**88.1**	0.882
awareness	68.8	87.3	**89.9**	0.897
sentient	42.0	85.6	**90.6**	0.925
physically existed	50.0	76.4	**82.7**	0.834
existed before	79.5	84.8	**85.1**	0.710
existed during	93.1	**95.1**	95.0	0.673
existed after	82.3	**87.5**	85.9	0.619
created	0.0	**44.4**	39.7	0.549
destroyed	17.1	0.0	**24.2**	0.346
changed	54.0	67.8	**70.7**	0.592
changed state	54.6	66.1	**71.0**	0.604
changed possession	0.0	38.8	**58.0**	0.640
changed location	6.6	35.6	**45.7**	0.702
stationary	13.3	21.4	**47.4**	0.711
location	0.0	18.5	**53.8**	0.619
physical contact	21.5	40.7	**47.2**	0.741
manipulated	72.1	86.0	**86.8**	0.737
micro f1	71.0	81.7	**83.3**	
macro f1	$55.4^{\star}$	$65.9^{\star}$	**71.1**	
macro-avg pearson				0.753

Table 2: SPR comparison to Teichert et al. (2017). Bold number indicate best F1 results in each row. Right-most column is pearson correlation coeficient for a model trained and tested on the scalar regression formulation of the same data.

ploits the benefits of large-scale pretraining, in the form of GloVe embeddings and MT pretraining, both absent in the CRF. Ablation experiments (Appendix A) show the advantages conferred by these features. In contrast, the discrete-featured CRF model makes use of gold dependency labels, as well as joint modeling of SPR attribute pairs with explicit joint factors, both absent in our neural model. Future SPRL work could explore the use of models like the LSTM-CRF (Lample et al., 2016; Ma and Hovy, 2016) to combine the advantages of both paradigms.

4 Experiments

Table 2 shows a side-by-side comparison of our model with prior work. The full breakdown of F1 scores over each individual property is provided. For every property except EXISTED DURING, EXISTED AFTER, and CREATED we are able to exceed prior performance. For some properties, the absolute F1 gains are quite large: DESTROYED (+24.2), CHANGED POSSESSION (+19.2.0), CHANGED LOCATION (+10.1), STATIONARY (+26.0) and LOCATION (+35.3). We also report performance with a scalar regression version of the model, evaluated with Pearson correlation. The scalar model is with respect to the original SPR annotations on a 5-point Likert scale, instead of a binary cut-point along that scale (> 3).

		phys. contact			volition		
		# DIFFER	Δ FALSE–	Δ FALSE+	# DIFFER	Δ FALSE–	Δ FALSE+
1	ALL	80	−14	6	80	−14	−10
2	PROPERNOUN	18	−2	−2	21	4	−5
3	ORG.	15	−9	2	31	−6	−1
4	PRONOUN	10	0	8	12	0	0
5	PHRASEVERB	14	−6	0	9	−4	1
6	METAPHOR	11	−5	−2	6	−2	0
7	LIGHTVERB	5	−2	1	5	−1	2

Table 3: Manual error analysis on a sample of instances (80 for each property) where outputs of CRF and the binary model from Table 2 differ. **Negative** Δ FALSE+ and Δ FALSE– indicate the neural model represents a **net *reduction* in type I and type II errors** respectively over CRF. Positive values indicate a net increase in errors. Each row corresponds to one of several (overlapping) subsets of the 80 instances in disagreement: (1) all (sampled) instances; (2) argument is a proper noun; (3) argument is an organization or institution; (4) argument is a pronoun; (5) predicate is phrasal or a particle verb construction; (6) predicate is used metaphorically; (7) predicate is a light-verb construction. #DIFFER is the size of the respective subset.

Manual Analysis We select two properties (VOLITION and MAKES PHYSICAL CONTACT) to perform a manual error analysis with respect to CRF [11] and our binary model from Table 2. For each property, we sample 40 dev instances with gold labels of “True” (> 3) and 40 instances of “False” (≤ 3), restricted to cases where the two system predictions disagree.[12] We manually label each of these instances for the six features shown in Table 3. For example, given the input “He *sits* down at the piano and plays,” our neural model correctly predicts that He makes physical contact during the *sitting*, while CRF does not. Since He is a pronoun, and *sits down* is phrasal, this example contributes −1 to Δ FALSE– in rows 1, 4 and 5.

---

[11] We obtained the CRF dev system predictions of Teichert et al. (2017) via personal communication with the authors.

[12] According to the reference, of the 1071 dev examples, 150 have physical contact and 350 have volition. The two models compared here differed in phy. contact on 62 positive and 44 negative instances and for volition on 43 positive and 54 negative instances.

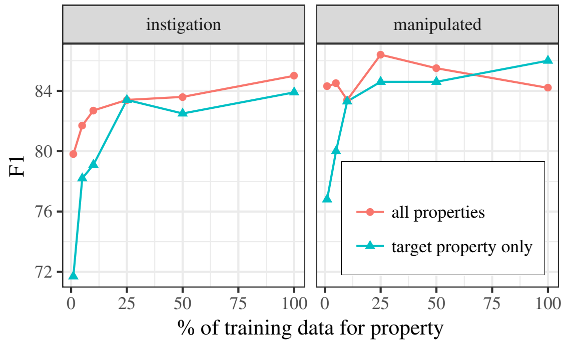

Figure 2: Effect of using only a fraction of the training data for a property while either ignoring or co-training with the full training data for the other SPR1 properties. Measurements at 1%, 5%, 10%, 25%, 50%, and 100%.

For both properties our model appears more likely to correctly classify the argument in cases where the predicate is a phrasal verb. This is likely a result of the fact that the BiLSTM has stronger language-modeling capabilities than the CRF, particularly with MT pretraining. In general, our model increases the false-positive rate for MAKES PHYSICAL CONTACT, but especially when the argument is pronominal.

Learning New SPR Properties One motivation for the decompositional approach adopted by SPRL is the ability to incrementally build up an inventory of annotated properties according to need and budget. Here we investigate (1) the degree to which having less training data for a single property degrades our F1 for that property on held-out data and (2) the effect on degradation of concurrent training with the other properties. We focus on two properties only: INSTIGATION, a canonical example of a proto-agent property, and MANIPULATED, which is a proto-patient property. For each we consider six training set sizes (1, 5, 10, 25, 50 and 100 percent of the instances). Starting with the same randomly initialized BiLSTM[13], we consider two training scenarios: (1) ignoring the remaining properties or (2) including the model's loss on other properties with a weight of $\lambda = 0.1$ in the training objective.

Results are presented in Figure 2. We see that, in every case, most of the performance is achieved with only 25% of the training data. The curves also suggest that training simultaneously on all SPR properties allows the model to learn the target property more quickly (i.e., with fewer training samples) than if trained on that property in isolation. For example, at 5% of the training training data, the "all properties" models are achieving roughly the same F1 on their respective target property as the "target property only" models achieves at 50% of the data.[14] As the SPR properties currently annotated are by no means semantically exhaustive,[15] this experiment indicates that future annotation efforts may be well served by favoring breadth over depth, collecting smaller numbers of examples for a larger set of attributes.

5 Conclusion

Inspired by: (1) the SPR decomposition of predicate-argument relations into overlapping feature bundles and (2) the neo-Davidsonian formalism for variable-arity predicates, we have proposed a straightforward extension to a BiLSTM classification framework in which the states of pre-identified predicate and argument tokens are pairwise concatenated and used as the target for SPR prediction. We have shown that our *Neural-Davidsonian* model outperforms the prior state of the art in aggregate and showed especially large gains for properties of CHANGED-POSSESSION, STATIONARY, and LOCATION. Our architecture naturally supports discrete or continuous label paradigms, lends itself to multi-task initialization or concurrent training, and allows for parameter sharing across properties. We demonstrated this sharing may be useful when some properties are only sparsely annotated in the training data, which is suggestive of future work in efficiently increasing the range of annotated SPR property types.

Acknowledgments

This research was supported by the JHU HLTCOE, DARPA AIDA, and NSF GRFP (Grant No. DGE-1232825). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA, NSF, or the U.S. Government.

---

[13] Note that this experiment does not make use of MT pretraining as was used for Table 2, to best highlight the impact of parameter sharing across attributes.

[14] As we observed the same trend more clearly on the dev set, we suspect some over-fitting to the development data which was used for independently select a stopping epoch for each of the plotted points.

[15] E.g., annotations do not include any questions relating to the *origin* or *destination* of an event.

References

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 77–89.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English web treebank. *Linguistic Data Consortium, Philadelphia, PA*.

Claire Bonial, Julia Bonn, Kathryn Conger, Jena D. Hwang, and Martha Palmer. 2014. Propbank: Semantics of new predicate types. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece. Association for Computational Linguistics.

Hector Neri Castañeda. 1967. Comment on d. davidsons "the logical forms of action sentences". In N. Rescher, editor, *The Logic of Decision and Action*. University of Pittsburgh Press, Pittsburgh.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Donald Davidson. 1967. The logical forms of action sentences. In N. Rescher, editor, *The Logic of Decision and Action*. University of Pittsburgh Press, Pittsburgh.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language*, 67(3):547–619.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

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

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

N. Ide and J. Pustejovsky. 2017. *Handbook of Linguistic Annotation*. Springer Netherlands.

Edward Kako. 2006. Thematic role properties of subjects and objects. *Cognition*, 101(1):1–42.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1528–1533, San Diego, California. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML

'01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *International Conference on Learning Representations*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Nitin Madnani, Jordan Boyd-Graber, and Philip Resnik. 2010. Measuring transitivity using untrained annotators. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazons Mechanical Turk*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6294–6305. Curran Associates, Inc.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 479–489.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Terence Parsons. 1995. Thematic relations and arguments. *Linguistic Inquiry*, pages 635–662.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Adam Poliak, Yonatan Belinkov, James Glass, and Benjamin Van Durme. 2018. On the evaluation of semantic phenomena in neural machine translation using natural language inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 513–523.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488.

Rachel Rudinger, Aaron Steven White, and Benjamin Van Durme. 2018. Neural models of factuality. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, Louisiana. Association for Computational Linguistics.

Roser Saurí and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. *Language Resources and Evaluation*, 43(3):227.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Adam Teichert, Adam Poliak, Benjamin Van Durme, and Matthew R Gormley. 2017. Semantic proto-role labeling. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. 2017. An Evaluation of PredPatt and Open IE via Stage 1 Semantic Role Labeling. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*.

Name	#	Description
LR		Logistic Regr. model, Reisinger et al. (2015)
CRF		CRF model, Teichert et al. (2017)
SPR1	0	SPR1 basic model
SPR1-RAND	0	SPR1, random word embeddings
MT:SPR1	1a	SPR1 after MT pretraining
PB:SPR1	1a	SPR1 after PB pretraining
MT:PB:SPR1	1a	SPR1 after MT+PB pretraining
SPR1+2	1b	SPR1 and SPR2 concurrently
SPR1+WSD	1b	SPR1 and WSD concurrently
MT:SPR1+2	1b	SPR1+2 after MT pretraining
MT:SPR1+WSD	1b	SPR1+WSD after MT pretraining
MT:SPR1S	1c	SPR1 scalar after MT pretraining
PB:SPR1S	1c	SPR1 scalar after PB pretraining
PS-MS	1d	SPR1 propty-specific model sel.
SPR2	3	SPR2 basic scalar model
MT:SPR2	3	SPR2 after MT pretraining
PB:SPR2	3	SPR2 after PB pretraining
MT:PB:SPR2	3	SPR2 after MT+PB pretraining

Table 4: Name and short description of each experimental condition reported. MT: indicates pretraining with machine translation; PB: indicates pretraining with PropBank SRL.

A Mult-Task Investigation

Multi-task learning has been found to improve performance on many NLP tasks, particularly for neural models, and is rapidly becoming *de rigueur* in the field. The strategy involves optimizing for multiple training objectives corresponding to different (but usually related) tasks. Collobert and Weston (2008) use multi-task learning to train a convolutional neural network to perform multiple core NLP tasks (POS tagging, named entity recognition, etc.). Multi-task learning has also been used to improve sentence compression (Klerke et al., 2016), chunking and dependency parsing (Hashimoto et al., 2017). Related work on UDS (White et al., 2016) shows improvements on event factuality prediction with multi-task learning on BiLSTM models (Rudinger et al., 2018). To complete the basic experiments reported in the main text, here we include an investigation of the impact of multi-task learning for SPRL.

We borrow insights from Mou et al. (2016) who explore different multi-task strategies for NLP including approach of initializing a network by training it on a related task ("INIT") versus interspersing tasks during training ("MULT"). Here we employ both of these strategies, referring to them as *pretraining* and *concurrent* training. We also use the terminology *target task* and *auxiliary task* to differentiate the primary task(s) we are interested in from those that play only a supporting role in training. In order to tune the impact of auxiliary tasks on the learned representation, Luong et al. (2016) use a *mixing parameter*, α_i, for each task i. Each parameter update consists of selecting a task with probability proportional to its α_i and then performing one update with respect to that task alone. They show that the choice of α has a large impact on the effect of multi-task training, which influences our experiments here.

Please refer to Appendix B for details on the datasets used in this section. In particular, with a few exceptions, White et al. (2016) annotates for the same set of properties as Reisinger et al. (2015), but with slightly different protocol and on a different genre. However, in this section we treat the two datasets as if they were separate tasks. To avoid cluttering the results in the main text, we exclusively present results there on what we call *SPR1* which consists of the data from Reisinger et al. (2015) and the train/dev/test splits of Teichert et al. (2017). We refer to the analogous tasks built on the data and splits of White et al. (2016) using the term SPR2. (We are not aware of any prior published results on property prediction for the SPR2.)

In addition to the binary and scalar SPR architectures outlined in Section 3 of the main paper, we also considered concurrently training the BiLSTM on a fine-grained word-sense disambiguation task or on joint SPR1 and SPR2 prediction. We also experimented with using machine translation and PropBank SRL to initialize the parameters of the BiLSTM. Preliminary experimentation on dev data with other combinations helped prune down the set of interesting experiments to those listed in Table 4 which assigns names to the models explored here. Our ablation study in Section 4 of the main paper uses the model named SPR1 while the other results in the main paper correspond to MT:SPR1 in the case of binary prediction and MT:SPR1S in the case of scalar prediction. After detailing the additional components used for pretraining or concurrent training, we present aggregate results and for the best performing models (according to dev) we present property-level aggregate results.

A.1 Auxiliary Tasks

Each auxiliary task is implemented in the form of a task-specific decoder with access to the hidden states computed by the shared BiLSTM encoder. In this way, the losses from these tasks backpropagate through the BiLSTM. Here we describe each task-specific decoder.

PropBank Decoder The network architecture for the auxiliary task of predicting abstract role types in PropBank is nearly identical to the architecture for SPRL described in Section 3 of the main paper. The main difference is that the PropBank task is a single-label, categorical classification task.

$$\mathrm{P}(\mathrm{role}_i|\boldsymbol{h}_{ea}) = \mathrm{softmax}_i\left(\boldsymbol{W}_{\mathrm{propbank}}\left[\boldsymbol{h}_{ea}\right]\right)$$

The loss from this decoder is the negative log of the probability assigned to the correct label.

Supersense Decoder The word sense disambiguation decoder computes a probability distribution over 26 WordNet supersenses with a simple single-layer feedforward network:

$$\mathrm{P}(\mathrm{supersense}_i|\boldsymbol{h}_a) = \mathrm{softmax}_i(\boldsymbol{W}\left[\boldsymbol{h}_a\right])$$

where $\boldsymbol{W} \in \mathbb{R}^{1200\times 26}$ and $\boldsymbol{h}_a$ is the RNN hidden state corresponding to the argument head token we wish to disambiguate. Since the gold label in the supersense prediction task is a *distribution* over supersenses, the loss from this decoder is the cross-entropy between its predicted distribution and the gold distribution.

French Translation Decoder Given the encoder hidden states, the goal of translation is to generate the reference sequence of tokens $Y = y_1, \cdots, y_n$ in the target language, i.e., French. We employ the standard decoder architecture for neural machine translation. At each time step i, the probability distribution of the decoded token y_i is defined as:

$$P(y_i) = \mathrm{softmax}\big(\tanh(\boldsymbol{W}_{\mathrm{fr}}\left[\boldsymbol{s_i}; \boldsymbol{c_i}\right] + \boldsymbol{b}_{\mathrm{fr}})\big)$$

where $\boldsymbol{W}_{\mathrm{fr}}$ is a transform matrix, and $\boldsymbol{b}_{\mathrm{fr}}$ is a bias. The inputs are the decoder hidden state $\boldsymbol{s_i}$ and the context vector $\boldsymbol{c_i}$. The decoder hidden state $\boldsymbol{s_i}$ is computed by:

$$\boldsymbol{s_i} = \mathrm{RNN}(\boldsymbol{y_{i-1}}, \boldsymbol{s_{i-1}})$$

where RNN is a recurrent neural network using L-layer stacked LSTM, $\boldsymbol{y_{i-1}}$ is the word embedding of token y_{i-1}, and $\boldsymbol{s_0}$ is initialized by the last encoder left-to-right hidden state.

	micro-F1	macro-F1
LR	71.0	55.4*
CRF	81.7	65.9*
SPR1-RAND	77.7	57.3
SPR1	**82.2**	69.3
MT:SPR1	**83.3**	71.1
PB:SPR1	**82.3**	67.9
MT:PB:SPR1	**82.8**	70.9
SPR1+2	**83.3**	70.4
SPR1+WSD	**81.9**	67.9
MT:SPR1+2	**83.2**	70.0
MT:SPR1+WSD	**81.8**	67.4
PS-MS	**82.9**	69.5

Table 5: Overall test performance for all settings described in Experiments 1 and 1a-d. The target task is SPR1 as binary classification. Micro- and macro-F1 are computed over all properties. (*Baseline macro-F1 scores are computed from property-specific precision and recall values in Teichert et al. (2017) and may introduce rounding errors.)

The context vector $\boldsymbol{c_i}$ is computed by an attention mechanism (Bahdanau et al., 2014; Luong et al., 2015),

$$\boldsymbol{c_i} = \sum_t \alpha_{i,t}\boldsymbol{h_t},$$

$$\alpha_{i,t} = \frac{\exp\left(\boldsymbol{s_i}^\top(\boldsymbol{W_\alpha h_t} + \boldsymbol{b_\alpha})\right)}{\sum_k \exp\left(\boldsymbol{s_i}^\top(\boldsymbol{W_\alpha h_k} + \boldsymbol{b_\alpha})\right)},$$

where $\boldsymbol{W_\alpha}$ is a transform matrix and $\boldsymbol{b_\alpha}$ is a bias. The loss is the negative log-probability of the decoded sequence.

A.2 Results

In this section, we present a series of experiments using different components of the neural architecture described in Section 3, with various training regimes. Each experimental setting is given a name (in SMALLCAPS) and summarized in Table 4. Unless otherwise stated, the target task is SPR1 (classification). To ease comparison, we include results from the main paper as well as additional results.

Experiment 0: Embeddings By default, all models reported in this paper employ pretrained word embeddings (GloVe). In this experiment we replaced the pretrained embeddings in the vanilla

	CRF	SPR1	MT:SPR1	SPR1+2
instigation	85.6	84.6	**88.6**	85.6
volition	86.4	**87.9**	**88.1**	**88.0**
awareness	87.3	**88.3**	**89.9**	**88.4**
sentient	85.6	**89.6**	**90.6**	**90.0**
physically existed	76.4	**82.3**	**82.7**	**80.2**
existed before	84.8	**86.0**	**85.1**	**86.8**
existed during	95.1	94.2	95.0	94.8
existed after	87.5	86.9	85.9	87.5
created	44.4	**46.6**	39.7	**51.6**
destroyed	0.0	**11.1**	**24.2**	**6.1**
changed	67.8	67.4	**70.7**	**68.1**
changed state	66.1	**66.8**	**71.0**	**67.1**
changed possession	38.8	**57.1**	**58.0**	**63.7**
changed location	35.6	**60.0**	**45.7**	**52.9**
stationary	21.4	**43.2**	**47.4**	**53.1**
location	18.5	**46.9**	**53.8**	**53.6**
physical contact	40.7	**52.7**	**47.2**	**54.7**
manipulated	86.0	82.2	**86.8**	**86.7**
micro f1	81.7	**82.2**	**83.3**	**83.3**
macro f1	65.9	**69.3**	**71.1**	**70.4**

Table 6: Breakdown by property of binary classification F1 on SPR1. All new results outperforming prior work (CRF) in bold.

SPR1 model (SPR1) with randomly initialized word embeddings (SPR1-RAND). The results (Table 5) reveal substantial gains from the use of pretrained embeddings; this is likely due to the comparatively small size of the SPR1 training data.

Experiment 1a: Multi-task Pretraining We pretrained the BiLSTM encoder with two separate auxiliary tasks: **French Translation** and **PropBank Role Labeling**. There are three settings: (1) Translation pretraining only (MT:SPR1), (2) PropBank pretraining only (PB:SPR1), and (3) Translation pretraining followed by PropBank pretraining (MT:PB:SPR1). In each case, after pretraining, the SPRL decoder is trained end-to-end, as in Experiment 0 (on SPR1 data).

Experiment 1b: Multi-task Concurrent One auxiliary task (**Supersense** or **SPR2**) is trained concurrently with SPR1 training. In one epoch of training, a training example is sampled at random (without replacement) from either task until all training instances have been sampled. The loss from the auxiliary task (which, in both cases, has more training instances than the target SPRL task) is down-weighted in proportion to ratio of the dataset sizes:

$$\alpha = \frac{|\text{target task}|}{|\text{auxiliary task}|}$$

SPR property	SPR1S	MT:SPR1S	SPR2
instigation	0.835	0.858	0.590
volition	0.869	0.882	0.837
awareness	0.873	0.897	0.879
sentient	0.917	0.925	0.880
physically existed	0.820	0.834	-
existed before	0.696	0.710	0.618
existed during	0.666	0.673	0.358
existed after	0.612	0.619	0.478
created	0.540	0.549	-
destroyed	0.268	0.346	-
changed	0.619	0.592	-
changed state	0.616	0.604	0.352
changed possession	0.359	0.640	0.488
change of location	0.778	0.702	0.492
changed state continuous	-	-	0.373
was for benefit	-	-	0.578
stationary	0.705	0.711	-
location	0.627	0.619	-
physical contact	0.731	0.741	-
manipulated	0.715	0.737	-
was used	-	-	0.203
partitive	-	-	0.359
macro-avg pearson	0.743	0.753	0.591

Table 7: SPR1 and SPR2 as scalar prediction tasks. Pearson correlation between predicted and gold values.

The auxiliary task loss is further down-weighted by a hyperparameter $\lambda \in \{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ which is chosen based on dev results. We apply this training regime with the auxiliary task of Supersense prediction (SPR1+WSD) and the scalar SPR2 prediction task (SPR1+SPR2), described in Experiment 2.

Experiment 1c: Multi-task Combination This setting is identical to Experiment 1b, but includes MT pretraining (the best-performing pretraining setting on dev), as described in 1a. Accordingly, the two experiments are MT:SPR1+WSD and MT:SPR1+SPR2.

Experiment 1d: Property-Specific Model Selection (PS-MS) Experiments 1a–1c consider a variety of pretraining tasks, co-training tasks, and weight values, λ, in an effort to improve aggregate F1 for SPR1. However, the SPR properties are diverse, and we expect to find gains by choosing training settings on a property-specific basis. Here, for each property, we select from the set of models considered in experiments 1a–1c the one that achieves the highest dev F1 for the target property. We report the results of applying those property-specific models to the test data.

SPR1S	0.743	SPR2	**0.591**
MT:SPR1S	**0.753**	MT:SPR2	0.577
PB:SPR1S	0.731	PB:SPR2	0.568
MT:PB:SPR1S	0.720	MT:PB:SPR2	0.564

Table 8: SPR1 and SPR2 as scalar prediction tasks. The overall performance for each experimental setting is reported as the average Pearson correlation over all properties. Highest SPR1 and SPR2 results are in bold.

Experiment 2: SPR as a scalar task In Experiment 2, we trained the SPR decoder to predict properties as scalar instead of binary values. Performance is measured by Pearson correlation and reported in Tables 8 and 7. In this case, we treat SPR1 and SPR2 both as target tasks (separately). By including SPR1 as a target task, we are able to compare (1) SPR as a binary task and a scalar task, as well as (2) SPR1 and SPR2 as scalar tasks. These results constitute the first reported numbers on SPR2.

We observe a few trends. First, it is generally the case that properties with high F1 on the SPR1 binary task also have high Pearson correlation on the SPR1 scalar task. The higher scoring properties in SPR1 scalar are also generally the higher scoring properties in SPR2 (where the SPR1 and SPR2 properties overlap), with a few notable exceptions, like INSTIGATION. Overall, correlation values are lower in SPR2 than SPR1. This may be the case for a few reasons. (1) The underlying data in SPR1 and SPR2 are quite different. The former consists of sentences from the Wall Street Journal via PropBank (Palmer et al., 2005), while the latter consists of sentences from the English Web Treebank (Bies et al., 2012) via the Universal Dependencies; (2) certain filters were applied in the construction of the SPR1 dataset to remove instances where, e.g., predicates were embedded in a clause, possibly resulting in an easier task; (3) SPR1 labels came from a single annotator (after determining in pilot studies that annotations from this annotator correlated well with other annotators), where SPR2 labels came from 24 different annotators with scalar labels averaged over two-way redundancy.

Discussion With SPR1 binary classification as the target task, we see overall improvements from various multi-task training regimes (Experiments 1a-d, Tables 5 and 6), using four different auxiliary tasks: machine translation into French, PropBank abstract role prediction, word sense disambiguation (WordNet supersenses), and SPR2.[16] These auxiliary tasks exhibit a loose trade-off in terms of the quantity of available data and the semantic relatedness of the task: MT is the least related task with the most available (parallel) data, while SPR2 is the most related task with the smallest quantity of data. While we hypothesized that the relatedness of PropBank role labeling and word sense disambiguation tasks might lead to gains in SPR performance, we did not see substantial gains in our experiments (PB:SPR1, SPR1+WSD). We did, however, see improvements over the target-task only model (SPR1) in the cases where we added MT pretraining (MT:SPR1) or SPR2 concurrent training (SPR1+2). Interestingly, combining MT pretraining with SPR2 concurrent training yielded no further gains (MT:SPR1+2).

B Data

SPR1 The SPR1.0 ("SPR1") dataset introduced by Reisinger et al. (2015) contains proto-role annotations on 4,912 Wall Street Journal sentences from PropBank (Palmer et al., 2005) corresponding to 9,738 predicate-argument pairs with 18 properties each, in total 175,284 property annotations. All annotations were performed by a single, trusted annotator. Each annotation is a rating from 1 to 5 indicating the likelihood that the property applies, with an additional "N/A" option if the question of whether the property holds is nonsensical in the context.

To compare with prior work (Teichert et al., 2017), we treat the SPR1 data as a binary prediction task: the values 4 and 5 are mapped to **True** (property holds), while the values 1, 2, 3, and "N/A" are mapped to **False** (property does not hold). In additional experiments, we move to treating SPR1 as a scalar prediction task; in this case, "N/A" is mapped to 1, and all other annotation values remain unchanged.

SPR2 The second SPR release (White et al., 2016) contains annotations on 2,758 sentences from the English Web Treebank (EWT) (Bies et al., 2012) portion of the Universal Dependencies (v1.2) (Silveira et al., 2014)[17], corresponding

---

[16] Note that in some cases we treat SPR2 as an auxiliary task, and in others, the target task.

[17] We exclude the SPR2 pilot data; if included, the SPR2

to 6,091 predicate-argument pairs. With 14 proto-role properties each, there are a total of 85,274 annotations, with two-way redundancy. As in SPR1, the value of each annotation is an integral value 1-5 or "N/A." We treat SPR2 as a scalar prediction task, first mapping "N/A" to 1, and then averaging the two-way redundant annotation values to a single value.

Word Sense Disambiguation Aligned with proto-role property annotations in the SPR2 release are word sense disambiguation judgments for the head tokens of arguments. Candidate word senses (fine-grained) from WordNet (Fellbaum, 1998) were presented to Mechanical Turk workers (at least three annotators per instance), who selected every applicable sense of the word in the given context. In this work, we map the fine-grained word senses to one of 26 coarse-grained WordNet noun supersenses (e.g., `noun.animal`, `noun.event`, `noun.quantity`, etc.). In many cases, a word may be mapped to more than one supersense. We treat the supersense label on a word as a distribution over supersenses, where the probability assigned to one supersense is proportional to the number of annotators that (indirectly) selected that supersense. In practice, the entropy of these resulting supersense distributions is low, with an average perplexity of 1.42.

PropBank The PropBank project consists of predicate-argument annotations over corpora for which gold Penn TreeBank-style constituency parses are available. We use the Unified PropBank release (Bonial et al., 2014; Ide and Pustejovsky, 2017), which contains annotations over OntoNotes as well as the English Web TreeBank (EWT). Each predicate in each corpus is annotated for word sense, and each argument of each predicate is given a label such as ARG0, ARG1, etc., where the interpretation of the label is defined relative to the word sense. We use PropBank Frames to map these sense-specific labels to 16 sense-independent labels such as PAG (proto-agent), PPT (proto-patient), etc., and then formulate a task to predict the abstracted labels. Because our model requires knowledge of predicate and argument head words, we ran the Stanford Universal Dependencies converter (Schuster and Manning, 2016) over the gold constituency parses to obtain Universal Dependency parses, which were then processed by the PredPatt framework (Zhang et al., 2017; White et al., 2016) to identify head words.

English-French Data The 10^9 French-English parallel corpus (Callison-Burch et al., 2009) contains 22,520,376 French-English sentence pairs, made up of 811,203,407 French words and 668,412,817 English words. The corpus was constructed by crawling the websites of international organizations such as the Canadian government, the European Union, and the United Nations.

---

release contains annotations for 2,793 sentences.

Conversational Decision-Making Model for Predicting the King's Decision in the Annals of the Joseon Dynasty

JinYeong Bak
School of Computing
KAIST
Republic of Korea
`jy.bak@kaist.ac.kr`

Alice Oh
School of Computing
KAIST
Republic of Korea
`alice.oh@kaist.edu`

Abstract

Styles of leaders when they make decisions in groups vary, and the different styles affect the performance of the group. To understand the key words and speakers associated with decisions, we initially formalize the problem as one of predicting leaders' decisions from discussion with group members. As a dataset, we introduce conversational meeting records from a historical corpus, and develop a hierarchical RNN structure with attention and pre-trained speaker embedding in the form of a, Conversational Decision Making Model (CDMM). The CDMM outperforms other baselines to predict leaders' final decisions from the data. We explain why CDMM works better than other methods by showing the key words and speakers discovered from the attentions as evidence.

1 Introduction

Decision making in groups refers to the process of making choices to resolve issues by discussing the issues with group members (Lunenburg, 2011). It has various styles based on the balance of the participation between the leader and members from autocratic, democratic, *laissez-faire* (let go) to delegation types of groups (Lewin et al., 1939; Vroom and Jago, 1988). Social psychologists note that decision making affects the group performance and the satisfaction of its members (Yang, 2010), and that leadership plays a role (Larson Jr et al., 1998). In this paper, we study the key factors that are closely related to the decision making process used by leaders.

First, we build conversational meeting records from *The Annals of the Joseon Dynasty* (henceforth referred to as the AJD), after which, we formalize our research problem as predicting leaders' decisions in conversational discussions from the data (Sec 2). The AJD consists of the records of kings who governed the Korean peninsula from 1392 to 1910. In the AJD, the kings discuss the issues with government officials and decide upon a course of action. Many discussion corpora are available such as Augmented Multi-party Interaction (AMI) (Carletta et al., 2005) which is meeting recordings as video, and are used to identify and summarize decisions in the conversation (Hsueh and Moore, 2007; Fernández et al., 2008; Bui et al., 2009). However, the AJD has more speakers than AMI, and it is a longitudinal corpus spanning over 400 years.

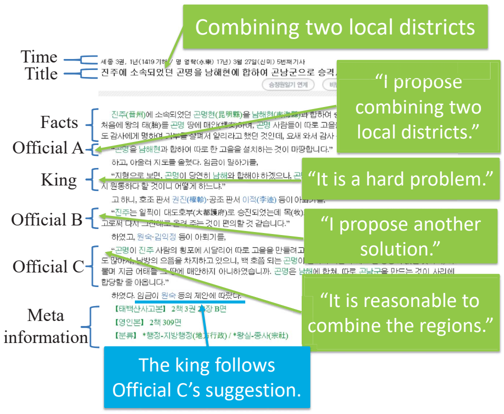

Figure 1: Screenshot and structure of an article in the annals of the Joseon dynasty

To predict the decisions in the corpus, we develop a model which we term the Conversational Decision-Making Model (CDMM) (Sec 3). CDMM is based on the hierarchical RNN structure with attention (Yang et al., 2016), but we add speaker information with pre-trained embedding. We also devise a way to make the speaker embedding using co-occurrence document network (Sec 3.3). In comparison with several other methods, CDMM shows the highest macro-averaged F1 score (Sec 4). We also show why CDMM works better with key words and speakers by examining the attention values (Sec 5).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 956–961
Brussels, Belgium, October 31 - November 4, 2018.

Kings	Articles	Utterances	Participants
15	13,216	95,615	4,502

(a) Basic statistics of the corpus

Order	1,996	*Accept*	1,457
Approve	2,245	*Reject*	818
Disapprove	468	*Discuss*	6,214

(b) Distribution of articles over decisions

Table 1: Statistics of a conversational meeting records and king's decisions from the AJD

2 Meeting Records from the AJD

Historiographers recorded the behaviors of kings and events in the country, and compiled these records as books when the king died or abdicated the throne. Each article of the AJD consists of the time, title, body and meta-information such as categories.

Meeting articles in the AJD consist of who said what on an issue in dialogue form, and the king's decision. Figure 1 shows an example of a meeting record article[1]. In the article, the king and government officials discuss the issue of combining two local regions. The king asks for a solution to the issue from the officials, and they state their opinions. At the end of the article, the king decided to follow official C's suggestion to solve this issue.

We build a corpus from the AJD using the following process. We crawl the AJD website to retrieve the documents and select articles that have three or more speakers per document. We identify the king's final decision in each article by examining the final sentence and the title as summarized by historians. We initially determine whether or not the final sentence of the subject is that by a king, as some issues are dealt with by others, such as the king's mother. We also extract the verbs in the final sentence and the title that indicates the decisions. From these, we categorize each king's decisions into six types: *Order*, *Approve*, *Disapprove*, *Accept* and *Reject*. Some articles include a discussion of an issue, but the king's final decision is not explicitly recorded or the king postpones the decision. We treat this type of decision as *Discuss*, i.e., the sixth category. Finally, we choose fifteen kings with more than 200 articles that have his final decisions. Table 1 shows the basic statistics of our meeting records data from the AJD.

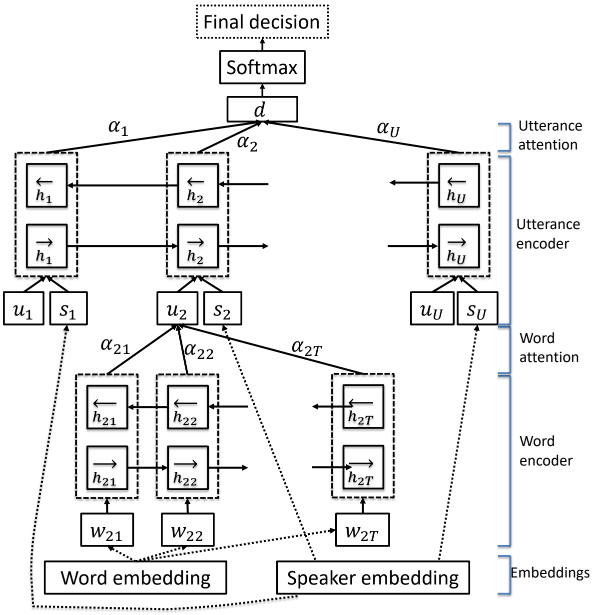

Figure 2: Conversational Decision-Making Model

3 Conversational Decision Making Model

This section describes our model, the Conversational Decision-Making Model (CDMM), for identifying leaders' decisions from meeting records. CDMM is based on the Hierarchical Attention Network (HAN) (Yang et al., 2016), but we change the sentence level to the utterance level and use speaker information (described in Section 3.2). To encode the speaker information, we build the speaker embedding from co-occurrence document network (described in Section 3.3).

3.1 Word Encoder

To encode the t-th word of i-th utterance x_{it}, $t \in \{1, \dots, T\}$, we initially change the word x_{it} to word vector w_{it} using the word embedding matrix W_w, $w_{it} = W_w x_{it}$. We use a bi-directional GRU (Bahdanau et al., 2014), and concatenate the hidden states $h_{it} = [\overrightarrow{h_{it}}; \overleftarrow{h_{it}}]$. Then, we use the attention mechanism in HAN to find important words to classify the decision. Each word has an attention value α_{it}, and we compute the utterance word vector, $u_i = \sum_{t=1}^{T} \alpha_{it} h_{it}$.

3.2 Utterance Encoder with Speaker

In CDMM, the i-th utterance has word sequence representation vector u_i and speaker vector s_i. First, we change the speaker z_i to vector s_i using

---

[1] `http://sillok.history.go.kr/id/kda_10103027_005`

the speaker embedding matrix $W_s, s_i = W_s z_i$. To encode a length U of the utterances $(u_i, s_i), i \in \{1, \ldots, U\}$, we suggest encoders based on GRU (Bahdanau et al., 2014), which can learn u_i and s_i simultaneously, as follows:

$$\begin{aligned}
h_i &= (1 - z_i) \odot h_{i-1} + z_i \odot \tilde{h}_i \\
z_i &= \sigma(W_{zu} u_i + W_{zs} s_i + U_z h_{i-1} + b_z) \\
r_i &= \sigma(W_{ru} u_i + W_{rs} s_i + U_r h_{i-1} + b_r) \\
\tilde{h}_i &= \tanh(W_{hu} u_i + W_{hs} s_i + r_i \odot (U_h h_{i-1}) + b_h)
\end{aligned}$$

Here, h_i is the i-th utterance hidden state, and z_i and r_i denote the update and reset gate, respectively. This is similar to earlier work (Li et al., 2016), but we add the speaker vector to the utterance level, not the word level.

As in the word encoder, we use the bidirectional GRU with the utterance encoder and concatenate the hidden states $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$. We use the same attention mechanism to find important utterances. Each utterance has an attention value of α_i, and for the conversation vector we use $d = \sum_{i=1}^{U} \alpha_i h_i$.

With vector d, CDMM predicts the decision using softmax $p = softmax(W_c d + b_c)$, and a dropout scheme (Srivastava et al., 2014) to avoid over-fitting.

3.3 Pre-trained Speaker Embedding

Unlike word embedding which is pre-trained from news or Wikipedia articles (Mikolov et al., 2013; Bojanowski et al., 2017), pre-trained speaker embedding for the AJD does not exist. To overcome this limitation, we suggest the building of speaker embedding from the co-occurrence document network in the AJD. The AJD contains not only meeting records but also personnel management reports and explanations of the officials. We therefore build a co-occurrence network. The vertices are people, and two individuals are connected if they appear in the same article. The weight of the edge is the number of co-occurrences in the same article. With this network, we realize speaker embedding using the node2vec algorithm (Grover and Leskovec, 2016), which generates node vector representation.

4 Experiments

This section describes the experiments and results of CDMM as well as other methods for classifying the king's decisions in the AJD.

4.1 Experiment Setting

We split the data as 80/10/10 for training/validation/test. Because the meeting records contain fifteen kings, we split the data randomly for each king and merge each part into the entire training, validation and test set.

We compare CDMM with the following methods. The majority of classes predicts all test examples as the major class, *Discuss*. We apply Naive Bayes and the SVM with the linear kernel. To use these methods, we remove words whose document frequency is smaller than twenty. To see the power of the speaker information, we run these baselines on words and speaker features together. We also run fastText (Joulin et al., 2017), which is a classifier with n-gram features and hierarchical softmax, and is similar to CBOW (Mikolov et al., 2013). We use pre-trained Korean word vectors[2] (Grave et al., 2018) to fastText and CDMM. We create the speaker embedding from the AJD. For a fair comparison, we exclude the valid and test articles to construct the co-occurrence network. We use node2vec implementation[3] for speaker embedding. We set the GRU hidden state size to 200, the dimension of the speaker embedding to 200 and the dropout probability to 0.5 for CDMM.

4.2 Predictions of the King's Decision Results

Table 2 shows the results. CDMM performs better than all other methods for macro-average and weighted-averaged metrics. The majority of classes shows the lowest performance. Naive Bayes and SVM outperform the baseline. fastText with pre-trained word vectors outperforms its counterpart, in accordance with an earlier result (Lample et al., 2016). CDMM without a speaker performs equally to HAN, the only difference being that HAN encodes sentences and CDMM encodes utterances. It does not show good performance as it models only the hierarchical structure of the conversation. However, when we add speaker information, the performance increases even with random initialization of speaker embedding. The performances of Naive Bayes and SVM also increase when they are assigned speakers as features. These observations signal that speaker information is helpful for predicting the king's decisions. Finally, CDMM with pre-trained speaker

---

[2] `https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md`

[3] `http://snap.stanford.edu/node2vec`

Method	*Micro* F_1	*Macro* $Prec$	*Macro* Rec	*Macro* F_1	*W-avg* F_1
Majority of classes	0.472	0.079	0.167	0.107	0.303
Naive Bayes	0.479	0.173	0.176	0.126	0.321
SVM linear	0.381	0.249	0.246	0.246	0.383
SVM RBF	0.487	0.236	0.186	0.142	0.337
Naive Bayes with speaker	0.466	0.268	0.177	0.135	0.323
SVM linear with speaker	0.423	0.292	0.259	0.243	0.403
SVM RBF with speaker	0.472	0.079	0.167	0.107	0.303
fastText w/o word vector	0.487	0.158	0.193	0.150	0.349
fastText	0.499	0.315	0.225	0.215	0.402
CDMM w/o speaker	0.481	0.176	0.214	0.178	0.379
CDMM with speaker (random init)	**0.504**	0.258	0.227	0.208	0.401
CDMM with speaker (pre-trained)	0.476	**0.329**	**0.307**	**0.313**	**0.456**

Table 2: King's decision classification precision, recall and F-measures. *Micro* $F1$ is the micro-averaged value of F-measure, and *Macro* $Prec$, Rec and F_1 are the macro-averaged values of precision, recall and F-measure respectively. *W-avg* F_1 is the weighted average according to the number of true examples in each class. CDMM outperforms all other methods compared.

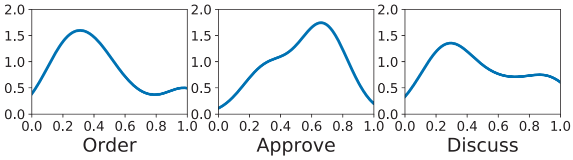

(a) Word "Wish to do"

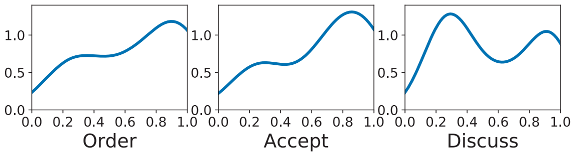

(b) Word "Okay"

Figure 3: Attention weight distribution of words for each class

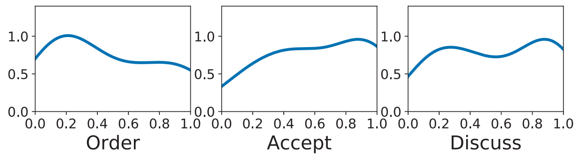

(a) Word "Okay" from kings

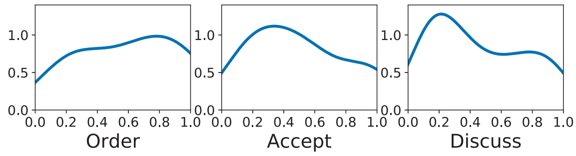

(b) Word "Okay" from officials

Figure 4: Attention weight distribution of word for each class from kings and officials

embedding shows better results compared to all other methods.

5 Discussion

Here, we investigate the attention values to determine the important words and speakers for predicting the king's decisions. We also obtain evidence showing why CDMM with pre-trained speaker embedding outperforms the others.

5.1 Key Words and Speakers

We investigate the important words using word attention values. To find the important words, we compute the mutual information (Christopher et al., 2008) of words that have the top 10% of attention values in the utterances among the classes.

Figure 3 shows the attention weight distributions of the two examples of the top words "Wish to do" and "Okay". The word "Wish to do" is usually used to make a request to the king. The peak of the attention weight distribution of "Wish to do" for the *Approve* class is around 0.7, whereas it is around 0.3 for *Order* and *Discuss*. We can interpret this to mean that CDMM assigns greater attention to that word to predict *Approve* compared to *Order* and *Discuss*. The word "Okay" is used to consent to the opinions of others. CDMM assigns a high attention value to the word to predict *Order* and *Accept* compared to *Discuss*.

However, the attention values differ according to the speaker. As shown in Figure 4, CDMM gives a high attention score to the word "Okay" for

Name (Eng)	Position	Class
Sin Sukju	Secretary	*Order*
Jeong Changson	Secretary	*Order*
Kim Jonkyung	Local gov	*Approve*
Kim Neuk	Local gov	*Approve*
Gwon Jin	Local gov	*Disapprove*
Kim Seup	Remonstrator	*Disapprove*
Hwang Hui	Central gov	*Accept*
Han Myeonghoe	Central gov	*Accept*
Kim Jikyung	Remonstrator	*Reject*
Sung Damnyeon	Remonstrator	*Reject*

Table 3: Name (translated in English) and position of the speakers who have high mutual information scores for the classes. Local gov is the local government official and Central gov is the central government official. Remonstrator is the official who remonstrates to the king. The position of the speaker is important to predict the king's decision.

Accept as compared to the other classes when the speaker is king. However, when officials use this word, CDMM assigns a high attention value to the word in the *Order* class. Despite the fact that the same word is used, the king's decision is changed based on the speaker. This is additional evidence showing why the speaker information is useful to predict the decision.

5.2 Position of the Speaker

We investigate the key speakers from utterance attention values. To determine the important person, we use the same technique of finding important words.

We find that high ranking person's positions are shared for each class. Table 3 shows the top ranked speakers and their positions for each class. The chief secretary who takes orders from the king has a high rank in the *Order* class. For *Approve* and *Disapprove*, local authorities are highly ranked. For *Accept*, central government officials have high MI values. Interestingly, officials who remonstrate to the king have high scores in the *Disapprove* and *Reject* class. We can thus say that the kings refuse admonitions commonly from officials.

From these results, we can gain insight into why pre-trained speaker embedding is helpful to predict the king's decisions. People in the same organization are in the same community of co-occurrence news article network (Özgür et al., 2008). Therefore, the AJD network contains the community information, and node2vec generates the node's closeness via embedding. CDMM can have this knowledge in the model therefore outperforms the other methods.

6 Conclusion

In this paper, we created conversational meeting data from the Annals of the Joseon Dynasty (AJD). We presented Conversational Decision-Making Model (CDMM) to predict leaders' decisions from the data. We also suggested the use of speaker embedding from co-occurrence document network with node2vec. With this data, we showed that CDMM outperforms other methods in terms of most metrics. We implemented CDMM using tensorflow (Abadi et al., 2016), and published the code and data in public[4]. We also analyzed the reasoning behind the success of CDMM and the key words and speakers by investigating the concept of attention.

Studies of small group dynamics can be helpful when attempting to understand group decision making behavior (Backstrom et al., 2006). Prior work which analyzed small group dynamics relied on a hidden Markov model (Magdon-Ismail et al., 2003), a dynamic Bayesian network (Mathur et al., 2012) or a layered probabilistic model (Cheng et al., 2014) for various datasets such as networks or recorded video. We suggest CDMM, which combine two types of data to predict leaders' decision. We can also apply this idea to other group dynamics analyses.

Acknowledgments

We would like to thank Jooyeon Kim for the helpful discussion and the anonymous reviewers for the inspiring questions and comments. This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-01778, Development of Explainable Human-level Deep Machine Learning Inference Framework).

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard,

---

[4] `https://github.com/NoSyu/CDMM`

et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283.

Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. 2006. Group formation in large social networks: Membership, growth, and evolution. In *Proceedings of the SIGKDD*, pages 44–54. ACM.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Trung H Bui, Matthew Frampton, John Dowding, and Stanley Peters. 2009. Extracting decisions from multi-party dialogue using directed graphical models and semantic similarity. In *Proceedings of the SIGDIAL*, pages 235–243.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. 2005. The ami meeting corpus: A pre-announcement. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 28–39. Springer.

Zhongwei Cheng, Lei Qin, Qingming Huang, Shuicheng Yan, and Qi Tian. 2014. Recognizing human group action by layered model with multiple cues. *Neurocomputing*, 136:124–135.

D Manning Christopher, Raghavan Prabhakar, and Schacetzel Hinrich. 2008. Introduction to information retrieval. *An Introduction To Information Retrieval*, 151(177):5.

Raquel Fernández, Matthew Frampton, Patrick Ehlen, Matthew Purver, and Stanley Peters. 2008. Modelling and detecting decisions in multi-party dialogue. In *Proceedings of the SIGDIAL*, pages 156–163.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the LREC*.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the SIGKDD*, pages 855–864. ACM.

Pei-Yun Hsueh and Johanna D Moore. 2007. What decisions have you made?: Automatic decision detection in meeting conversations. In *Proceedings of the NAACL HLT*, pages 25–32.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the EACL*, pages 427–431.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the NAACL HLT*, pages 260–270.

James R Larson Jr, Pennie G Foster-Fishman, and Timothy M Franz. 1998. Leadership style and the discussion of shared and unshared information in decision-making groups. *Personality and Social Psychology Bulletin*, 24(5):482–495.

Kurt Lewin, Ronald Lippitt, and Ralph K White. 1939. Patterns of aggressive behavior in experimentally created "social climates". *The Journal of social psychology*, 10(2):269–299.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the ACL*, pages 994–1003.

Frank C Lunenburg. 2011. Decision making in organizations. *International journal of management, business, and administration*, 15(1):1–9.

Malik Magdon-Ismail, Mark Goldberg, William Wallace, and David Siebecker. 2003. Locating hidden groups in communication networks using hidden markov models. In *International Conference on Intelligence and Security Informatics*, pages 126–137. Springer.

Shobhit Mathur, Marshall Scott Poole, Feniosky Pena-Mora, Mark Hasegawa-Johnson, and Noshir Contractor. 2012. Detecting interaction links in a collaborating group using manually annotated data. *Social Networks*, 34(4):515–526.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Arzucan Özgür, Burak Cetin, and Haluk Bingol. 2008. Co-occurrence network of reuters news. *International Journal of Modern Physics C*, 19(05):689–702.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Victor H Vroom and Arthur G Jago. 1988. *The new leadership: Managing participation in organizations.* Prentice-Hall, Inc.

Maria C Yang. 2010. Consensus and single leader decision-making in teams using structured design methods. *Design Studies*, 31(4):345–362.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the NAACL: HLT*, pages 1480–1489.

Toward Fast and Accurate Neural Discourse Segmentation

Yizhong Wang **Sujian Li** **Jingfeng Yang**
MOE Key Lab of Computational Linguistics, School of EECS, Peking University
`{yizhong, lisujian, yjfllpyym}@pku.edu.cn`

Abstract

Discourse segmentation, which segments texts into Elementary Discourse Units, is a fundamental step in discourse analysis. Previous discourse segmenters rely on complicated hand-crafted features and are not practical in actual use. In this paper, we propose an end-to-end neural segmenter based on BiLSTM-CRF framework. To improve its accuracy, we address the problem of data insufficiency by transferring a word representation model that is trained on a large corpus. We also propose a restricted self-attention mechanism in order to capture useful information within a neighborhood. Experiments on the RST-DT corpus show that our model is significantly faster than previous methods, while achieving new state-of-the-art performance. [1]

1 Introduction

Discourse segmentation, which divides text into proper discourse units, is one of the fundamental tasks in natural language processing. According to Rhetorical Structure Theory (RST) (Mann and Thompson, 1988), a complex text is composed of non-overlapping Elementary Discourse Units (EDUs), as shown in Table 1. Segmenting text into such discourse units is a key step in discourse analysis (Marcu, 2000) and can benefit many downstream tasks, such as sentence compression (Sporleder and Lapata, 2005) or document summarization (Li et al., 2016).

Since EDUs are initially designed to be determined with lexical and syntactic clues (Carlson et al., 2001), existing methods for discourse segmentation usually design hand-crafted features to capture these clues (Feng and Hirst, 2014). Especially, nearly all previous methods rely on syntactic parse trees to achieve good performance. But extracting such features usually takes a long time, which contradicts the fundamental role of discourse segmentation and hinders its actual use. Considering the great success of deep learning on many NLP tasks (Lu and Li, 2016), it's a natural idea for us to design an end-to-end neural model that can segment texts fast and accurately.

[Mr. Rambo says]$_{e_1}$ [that a 3.2-acre property]$_{e_2}$ [overlooking the San Fernando Valley]$_{e_3}$ [is priced at \$4 million]$_{e_4}$ [because the late actor Erroll Flynn once lived there.]$_{e_5}$

Table 1: A sentence that is segmented into five EDUs

The first challenge of applying neural methods to discourse segmentation is data insufficiency. Due to the limited size of labeled data in existing corpus (Carlson et al., 2001), it's quite hard to train a data-hungry neural model without any prior knowledge. In fact, some traditional features, such as the POS tags or parse trees, naturally provide strong signals for identifying EDUs. Removing them definitely increases the difficulty of learning an accurate model. Secondly, many EDU boundaries are actually not determined locally. For example, to recognize the boundary between e_3 and e_4 in Table 1, our model has to be aware that e_3 is an embedded clauses starting from "overlooking", otherwise it could regard "San Fernando Valley" as the subject of e_4. Such kind of long-distance dependency can be precisely extracted from parse trees but is difficult for neural models to capture.

To address these challenges, in this paper, we propose a neural discourse segmenter based on the BiLSTM-CRF (Huang et al., 2015) framework and further improve it from two aspects. Firstly, since the discourse segmentation corpus is too small to learn precise word representations, we transfer a word representation model trained on a large corpus into our task, and show that this trans-

---

[1] Our code is available at `https://github.com/PKU-TANGENT/NeuralEDUSeg`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 962–967
Brussels, Belgium, October 31 - November 4, 2018.

ferred model can provide very useful information for our task. Secondly, in order to model long-distance dependency, we employ the self-attention mechanism (Vaswani et al., 2017) when encoding the text. Different from previous self-attention, we restrict the attention area to a neighborhood of fixed size. The motivation is that effective information for determining the boundaries is usually collected from adjacent EDUs, while the whole text may contain many disturbing words, which could mislead the model into incorrect decisions. In summary, the contributions of this work are as follows:

- Our neural discourse segmentation model doesn't rely on any syntactic features, while it can outperform other state-of-the-art systems and achieve significant speedup.
- To our knowledge, we are the first to transfer word representations learned from large corpus into discourse segmentation task and show that they can significantly alleviate the data insufficiency problem.
- Based on the nature of discourse segmentation, we propose a restricted attention mechanism , which enables the model to capture useful information within a neighborhood but ignore unnecessary faraway noises.

2 Neural Discourse Segmentation Model

We model discourse segmentation as a sequence labeling task, where the start word of each EDU (except the first EDU) is supposed to be labeled as 1 and other words are labeled as 0. Figure 1 gives an overview of our segmentation model. We will introduce the BiLSTM-CRF framework in Section 2.1, and describe the two key components of our model in Section 2.2, 2.3.

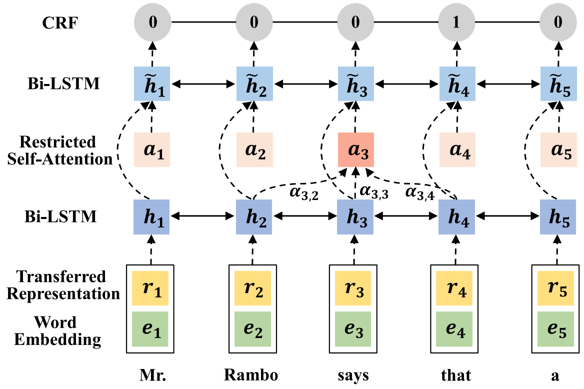

Figure 1: Overview of our model for discourse segmentation

2.1 BiLSTM-CRF for Sequence Labeling

Conditional Random Fields (CRF) (Lafferty et al., 2001) is an effective method to sequence labeling problem and has been widely used in many NLP tasks (Sutton and McCallum, 2012). To approach our discourse segmentation task in a neural way, we adopt the BiLSTM-CRF model (Huang et al., 2015) as the framework of our system. Formally, given an input sentence $\mathbf{x} = \{x_t\}_{t=1}^{n}$, we first embed each word into a vector $\mathbf{e}_t$. Then these word embeddings are fed into a bi-directional LSTM layer to model the sequential information:

$$\mathbf{h}_t = \text{BiLSTM}(\mathbf{h}_{t-1}, \mathbf{e}_t) \qquad (1)$$

where $\mathbf{h}_t$ is the concatenation of the hidden states from both forward and backward LSTMs. After encoding this sentence, we make labeling decisions for each word. Instead of modeling the decisions independently, the CRF layer computes the conditional probability $p(\mathbf{y}|\mathbf{h}; \mathbf{W}, \mathbf{b})$ over all possible label sequences $\mathbf{y}$ given $\mathbf{h}$ as follows:

$$p(\mathbf{y}|\mathbf{h}; \mathbf{W}, \mathbf{b}) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, \mathbf{h})}{\sum\limits_{y' \in \mathcal{Y}} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, \mathbf{h})} \qquad (2)$$

where $\psi_i(y_{i-1}, y_i, \mathbf{h}) = \exp(\mathbf{w}^T \mathbf{h}_i + b)$ is the potential function and $\mathcal{Y}$ is the set of possible label sequences. The training objective is to maximize the conditional likelihood of the golden label sequence. During testing, we search for the label sequence with the highest conditional probability.

2.2 Transferring Representations Learned from Large Corpus

Due to the large parameter space, neural models usually require much more training data in order to achieve good performance. However, to the best of our knowledge, nearly all existing discourse segmentation corpora are limited in size. After we remove all the syntactic features, which has been proven useful in many previous work (Bach et al., 2012; Feng and Hirst, 2014; Joty et al., 2015), it's expected that our neural model will not achieve very satisfying results.

To tackle this issue, we propose to leverage model learned from other large datasets, aiming that this transferred model has been well trained to encode text and capture useful signals. Instead of training the transferred model by ourselves, in this paper, we adopt the ELMo word representations (Peters et al., 2018), which are derived from a bidirectional language model (BiLM) trained on one billion word benchmark corpus (Chelba et al., 2014). Specifically, this BiLM has one character convolution layer and two biLSTM layers, and correspondingly there are three internal representations for each word x_t, which are denoted as $\{\mathbf{h}_{t,l}^{\text{LM}}\}_{l=1}^{3}$. Following (Peters et al., 2018), we compute the ELMo representation $\mathbf{r}_t$ for word x_t as follows:

$$\mathbf{r}_t = \gamma^{\text{LM}} \sum_{l=0}^{3} s_l^{\text{LM}} \mathbf{h}_{t,l}^{\text{LM}} \tag{3}$$

where $\mathbf{s}^{\text{LM}}$ are normalized weights and γ^{LM} controls the scaling of the entire ELMo vector. Then we concatenate $\mathbf{r}_t$ with the word embedding $\mathbf{e}_t$, and take them as the input of Equation (1).

2.3 Restricted Self-Attention

As we have introduced in Section 1, some EDU boundaries rely on relatively long-distance signals to recognize, while normal LSTM model is still weak at this. Recently, self-attention mechanism, which relates different positions of a single sequence, has been successfully applied to many NLP tasks (Vaswani et al., 2017; Wang et al., 2017) and shows its superiority in capturing long dependency. However, we found that most boundaries are actually only influenced by nearby EDUs, thereby forcing the model to attend to the whole sequence will bring in unnecessary noises. Therefore, we propose a restricted self-attention mechanism, which only collects information from a fixed neighborhood. To do this, we first compute the similarity between current word x_i and each nearby word x_j within a window:

$$s_{i,j} = \mathbf{w}_{attn}^{T}[\mathbf{h}_i, \mathbf{h}_j, \mathbf{h}_i \odot \mathbf{h}_j] \tag{4}$$

Then the attention vector $\mathbf{a}_i$ is computed as a weighted sum of nearby words:

$$\alpha_{i,j} = \frac{e^{s_{i,j}}}{\sum_{k=-K}^{K} e^{s_{i,i+k}}} \tag{5}$$

$$\mathbf{a}_i = \sum_{j=-K}^{K} \alpha_{i,i+k} \mathbf{h}_{i+k} \tag{6}$$

where hyper-parameter K is the window size. This attention vector $\mathbf{a}_i$ is then put into another BiLSTM layer together with $\mathbf{h}_i$ in order to fuse the information:

$$\tilde{\mathbf{h}}_t = \text{BiLSTM}(\tilde{\mathbf{h}}_{t-1}, [\mathbf{h}_t, \mathbf{a}_t]) \tag{7}$$

We use $\tilde{\mathbf{h}}_t$ as the new input to the CRF layer.

3 Experiments and Results

3.1 Dataset and Metrics

We conduct experiments on the RST Discourse Treebank (RST-DT) (Carlson et al., 2001). The original corpus contains 385 Wall Street Journal articles from the Penn Treebank, which are divided in to training set (347 articles, 6132 sentences) and test set (38 articles, 991 sentences). We randomly sample 34 (10%) articles from the train set as validation set in order to tune the hyper-parameters and only train our model on the remained train set. We follow mainstream studies (Soricut and Marcu, 2003; Joty et al., 2015) to measure segmentation accuracy only with respect to the intra-sentential segment boundaries, and we report Precision (P), Recall (R) and F1-score (F1) for segmentation performance.

3.2 Implementation Details

We tune all the hyper-parameters according to the model performance on the separated validation set. The 300-D Glove embeddings (Pennington et al., 2014) are employed and kept fixed during training. We use the AllenNLP toolkit (Gardner et al., 2018) to compute the ELMo word representations. The hidden size of our model is set to be 200 and the batch size is 32. L2 regularization is applied to trainable variables with its weight as 0.0001 and we use dropout between every two layers, where the dropout rate is 0.1. For model training, we employ the Adam algorithm (Kingma and Ba, 2014) with its initial learning rate as 0.0001 and we clip the gradients to a maximal norm 5.0. Exponential moving average is applied to all trainable variables with a decay rate 0.9999. The window size K for restricted attention is set to be 5.

3.3 Performance

The results of our model and other competing systems on the test set of RST-DT are shown in Table 2. We compare our results against the following systems: (1) **SPADE** (Soricut and Marcu, 2003) is an early system using simple lexical and syntactic features; (2) **NNDS** (Subba and Di Eugenio, 2007) uses a neural network classifier to do the

Model	Tree	P(%)	R(%)	F1(%)
SPADE	Gold	84.1	85.4	84.7
NNDS	Gold	85.5	86.6	86.0
CRFSeg	Gold	92.7	89.7	91.2
Reranking	Gold	**93.1**	94.2	93.7
CRFSeg	Stanford	91.0	87.2	89.0
CODRA	BLLIP	88.0	92.3	90.1
Reranking	Stanford	91.5	90.4	91.0
Two-Pass	BLLIP	92.8	92.3	92.6
Our Model	No	92.9	**95.7**	**94.3**
- Attention	No	92.4	94.8	93.6
- ELMo	No	87.9	84.5	86.2
- Both	No	87.0	82.8	84.8
Human	No	98.5	98.2	98.3

Table 2: Performance of our model and other systems on the RST-DT test set [3]

segmentation after extracting features; (3) **CRFSeg** (Hernault et al., 2010) is the first discourse segmenter using CRF model; (4) **CODRA** (Joty et al., 2015) uses fewer features and a simple logistic regression model to achieve impressive results; (5) **Reranking** (Bach et al., 2012) reranks the N-best outputs of a base CRF segmenter; (6) **Two-Pass** (Feng and Hirst, 2014) conducts a second segmentation after extracting global features from the first segmentation result. All these methods rely on tree features and we list their performance given different parse trees, where **Gold** are the trees extracted from the Penn Treebank (Prasad et al., 2005), **Stanford** represents trees from the Stanford parser (Klein and Manning, 2003) and **BLLIP** represents those from the BLLIP parser (Charniak and Johnson, 2005). It should be noted that the results of SPADE and CRFSeg are taken from Bach et al. (2012) since the original papers adopt different evaluation metrics. All the other results are taken from the corresponding original papers.

From Table 2, we can see that our model achieves state-of-the-art performance without extra parse trees. Especially, if no gold parse trees are provided, our system outperforms other methods by more than 1.7 points in F1 score. Since the gold parse trees are not available when processing new sentences, this improvement becomes more valuable when the system is put into use.

System	Speed (Sents/s)	Speedup
Two-Pass	1.39	1.0x
SPADE	3.78	2.7x
Ours (Batch=1)	9.09	6.5x
Ours (Batch=32)	76.23	54.8x

Table 3: Speed comparison with two open-sourced discourse segmenter

To further explore the influence of different components in our model, we also report the results of ablation experiments in Table 2. We can see that the transferred ELMo representations provide the most significant improvement. This accords with our assumption that the RST-DT corpus itself is not large enough to train an expressive neural model sufficiently. With the help of the transferred representations, we are capable of capturing more semantic and syntactic signals. Also, comparing the models with and without the restricted self-attention, we find that this attention mechanism can further boost the performance. Especially, if there are no ELMo vectors, the improvement provided by the attention mechanism is more noticeable.

3.4 Speed Comparison

We also measure the speedup of our model against traditional systems in Table 3. The **Two-Pass** system has the best performance among all existing methods, while **SPADE** is much simpler with less features. We test these systems on the same machine (CPU: Intel Xeon E5-2690, GPU: NVIDIA Tesla P100). The results show that our system is 2.4-6.5 times faster than the compared systems if the batch size is 1. Moreover, if we process the test sentences in parallel, we can achieve 20.2-54.8 times speedup with the batch size as 32. This makes our system more practical in actually use.

3.5 Effect of Restricted Self-Attention

We propose to restrict the self-attention within a neighborhood instead of the whole sequence. Table 4 demonstrates the performance of our model over different window size K. We can see that all these results is better than the performance our model without attention mechanism. However, a proper restriction window is helpful for the attention mechanism to take better effect.

---

[3] In parallel with our work, Li et al. (2018) proposes another neural model with its performance as: P-91.6, R-92.8, F1-92.2. We didn't see their paper at the time of submission, but it's worth mentioning here for the readers' reference.

Window Size	1	5	10	∞
F1-score	94.0	94.3	94.2	93.8

Table 4: Performance of our model over different attention window size

4 Conclusion

In this paper, we propose a neural discourse segmenter that can segment text fast and accurately. Different from previous methods, our segmenter doesn't rely on any hand-crafted features, especially the syntactic parse tree. To achieve our goal, we propose to leverage the word representations learned from large corpus and we also propose a restricted self-attention mechanism. Experimental results on RST-DT show that our system can achieve state-of-the-art performance together with significant speedup.

Acknowledgments

We thank the anonymous reviewers for their insightful comments on this paper. This work was partially supported by National Natural Science Foundation of China (61572049 and 6187022165). The corresponding author of this paper is Sujian Li.

References

Ngo Xuan Bach, Minh Le Nguyen, and Akira Shimazu. 2012. A reranking model for discourse segmentation using subtree features. In *Proceedings of the SIGDIAL 2012 Conference, The 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 5-6 July 2012, Seoul National University, Seoul, South Korea*, pages 160–168.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurovsky. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the SIGDIAL 2001 Workshop, The 2nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, Saturday, September 1, 2001 to Sunday, September 2, 2001, Aalborg, Denmark.*

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 173–180.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639.

Vanessa Wei Feng and Graeme Hirst. 2014. Two-pass discourse segmentation with pairing and global features. *CoRR*, abs/1407.8215.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640.*

Hugo Hernault, Helmut Prendinger, David A DuVerle, Mitsuru Ishizuka, and Tim Paek. 2010. Hilda: a discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Shafiq R. Joty, Giuseppe Carenini, and Raymond T. Ng. 2015. CODRA: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan.*, pages 423–430.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289.

Jing Li, Aixin Sun, and Shafiq Joty. 2018. Segbot: A generic neural text segmentation model with pointer network. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4166–4172.

Junyi Jessy Li, Kapil Thadani, and Amanda Stent. 2016. The role of discourse units in near-extractive summarization. In *Proceedings of the SIGDIAL 2016 Conference, The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 13-15 September 2016, Los Angeles, CA, USA*, pages 137–147.

Zhengdong Lu and Hang Li. 2016. Recent progress in deep learning for NLP. In *Tutorial Abstracts, NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016, pages 11–13.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT press.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Rashmi Prasad, Aravind Joshi, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, and Bonnie Webber. 2005. The penn discourse treebank as a resource for natural language generation. In *Proceedings of the Corpus Linguistics Workshop on Using Corpora for Natural Language Generation*, pages 25–32.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics.

Caroline Sporleder and Mirella Lapata. 2005. Discourse chunking and its application to sentence compression. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 257–264. Association for Computational Linguistics.

Rajen Subba and Barbara Di Eugenio. 2007. Automatic discourse segmentation using neural networks. In *Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue*, pages 189–190.

Charles A. Sutton and Andrew McCallum. 2012. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*.

A Dataset for Telling the Stories of Social Media Videos

Spandana Gella[1*] **Mike Lewis**[2] **Marcus Rohrbach**[2]
[1]University of Edinburgh, [2]Facebook AI Research
`spandana.gella@ed.ac.uk`, {`mikelewis,mrf`}`@fb.com`

Abstract

Video content on social media platforms constitutes a major part of the communication between people, as it allows everyone to share their stories. However, if someone is unable to consume video, either due to a disability or network bandwidth, this severely limits their participation and communication. Automatically telling the stories using multi-sentence descriptions of videos would allow bridging this gap. To learn and evaluate such models, we introduce VideoStory, a new large-scale dataset for video description as a new challenge for multi-sentence video description. Our VideoStory captions dataset is complementary to prior work and contains 20k videos posted publicly on a social media platform amounting to 396 hours of video with 123k sentences, temporally aligned to the video.

1 Introduction

Telling stories about what we experience is a central part of human communication (Mateas and Sengers, 2003). Increasingly, stories about our experiences are captured in the form of videos and then shared on social media platforms. One goal of automatically understanding and describing such videos with natural language is to generate multi-sentence descriptions which convey the story, making them accessible to situationally (e.g. bandwidth) or physically ("blind") disabled people. However, it is still a challenge for vision and language models to automatically encode and describe temporal content in videos with multi-sentence descriptions (Rohrbach et al., 2014; Zhou et al., 2018b). To better understand the stories shared on social media we collect and annotate a novel dataset consisting of videos from a social media platform. Importantly, we collect descriptions containing multiple sentences, as single sentences would typically not be able to capture the narration and plot of the video.

We introduce a large-scale multi-sentence description dataset for videos. To build a dataset of high quality, diverse and narratively interesting videos, we choose videos that had high engagement on a social media platform. Existing video captioning datasets, such as ActivityNet Captions (Krishna et al., 2017) or cooking video datasets (Regneri et al., 2013; Zhou et al., 2018a), have focused on sets of pre-selected human activities, whereas social media videos contain a great diversity of topics. Videos with high engagement tend to be narratively interesting, because humans find very predictable videos less enjoyable, meaning that captioning of the videos accurately requires integrating information from the entire video to describe a sequence of events (see Figure 1). Together, this creates a diverse and challenging new benchmark for video and language understanding.

We present a thorough analysis of the new benchmark, demonstrating that linguistic and video context is crucial to accurate captioning and that the captions have a temporal consistency. We also show baseline results using state-of-the-art models.

2 Multi-Sentence VideoStory Dataset

In Table 1 we summarize existing video description datasets; most provide only single-sentence descriptions or are restricted to narrow domains. Other multi-sentence description datasets are proposed for story narration of sets of images taken from a Flickr album (Huang et al., 2016; Krause et al., 2017). Other related work includes visual summarization of Flickr photo albums (Sigurdsson et al., 2016a) or videos (De Avila et al., 2011; Zhang et al., 2016) where the idea is to pick the key images or frames that summarize the visual content.

---

[*] Work done while SG was intern at Facebook AI Research.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 968–974
Brussels, Belgium, October 31 - November 4, 2018.

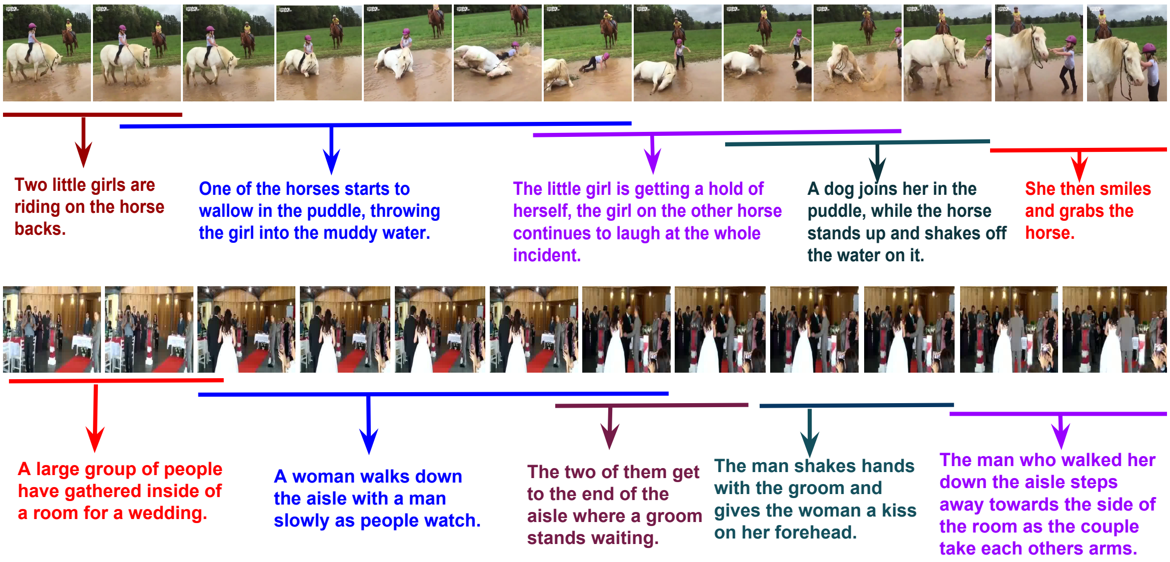

Figure 1: Example videos and multi-sentence description in our VideoStory Dataset showing temporally, overlapping time alignments. Each segment has time boundaries annotated and is described by a sentence.

Dataset	Domain	# videos:clips	Avg.D	#ActL	#sent	Loc	multi-sent.	overlap
MSVD (Chen and Dolan, 2011)	human	:2k	10s	-	70k	✓	-	-
MSR-VTT (Xu et al., 2016)	open	7k:10k	20s	-	200k	✓	-	-
Charades (Sigurdsson et al., 2016b)	human	10k:	30s	157	16.1k	-	✓	-
YouCook (Das et al., 2013)	cooking	88:-	95s		2.7k	✓	-	-
VTW (Zeng et al., 2016)	open	18k:-	90s	-	45k	-	✓	-
TGIF (Li et al., 2016)	open	:100k	3s	-	128k	✓	-	-
MPII MD (Rohrbach et al., 2015)	movie	94:68k	4s	-	68.3k	✓	(✓)	(✓)
M-VAD (Torabi et al., 2015)	movie	92:46k	6s	-	55.9k	✓	(✓)	(✓)
LSMDC (Rohrbach et al., 2017)	movie	200:128k	4s	-	128.1k	✓	(✓)	(✓)
TACoS (Regneri et al., 2013)	cooking	127:3.5k	286s:		11.8k	✓	✓	-
TACos multi-level (Rohrbach et al., 2014)	cooking	185:25k	307s:	67	75k	✓	✓	-
Youcook II (Zhou et al., 2018a)	cooking	2k:15.4	316:19.6s	-	15.4k	✓	✓	-
ActivityNet Captions (Krishna et al., 2017)	human activity	20k:100k	180:36s	203	100k	✓	✓	✓
VideoStory (Ours)	social media	20k:123k	70:18s	-	123k	✓	✓	✓

Table 1: Comparison of our dataset with other video description datasets. Avg.D: Average duration of the video/clip. #ActL: No. of action labels. Loc: temporally localized language descriptions; multi-sent: multi-sentence descriptions; overlap: allows overlap among segments. (✓) indicates datasets with multiple sentences, however they are mainly used to generate individual clip descriptions.

We select videos posted on a social media platform to create our dataset because of the variability in topics, length, viewpoints, and quality. They also tend to represent a good distribution of stories communicated by humans. We select videos from social media that are public and popular with a large number of comments and shares that triggered interactions between people. In total, our dataset consists of 20k videos with duration ranging from 20s-180s and spanning across diverse topics that are observed on social media platforms. We follow Krishna et al. (2017) to create temporally annotated sentences where each task is divided into two steps: (i) describing the video in multiple sentences, covering objects, situations and important details of the video; (ii) aligning each sentence in the paragraph with the corresponding timestamps in the video. We refer to these as video segments. In Figure 1, we present two example annotated videos describing (i) a scene where two girls are playing with horses; (ii) a wedding with a bride walking down the aisle.

We summarize the statistics of our dataset in Table 2 and compare it to prior work in Table 1. Each of the 20k videos in our VideoStory dataset is annotated with a paragraph which has on average 4.67 temporally localized sentences. As we have three paragraphs per video for validation and test set, we have a total of 26,245 paragraphs with a total of 123k sentences. Each sentence in the dataset

Split	#Videos	#Clips	#Para	#W/P	Original	shuffled
train	17,098	80,598	17,098	61.76	-	-
val	999	13,796	2,997	59.88	20.95	24.82
test	1,011	14,093	3,033	59.77	21.12	24.95
test_blind	1,039	14,139	3,117	69.45	23.81	27.99
total	20,147	122,626	26,245	62.23	-	-

Table 2: VideoStory dataset: Dataset statistics (#V: No.of unique videos. #Para: No.of unique paragraphs. #W/P: Average number of words per paragraph.) and perplexity scores for original and shuffled sentences.

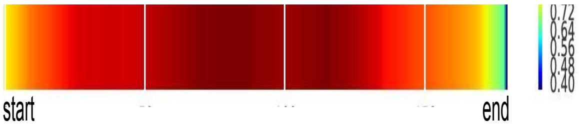

Figure 2: Distribution of annotations in time in VideoStory dataset. Most of the videos have majority of it annotated except the first few and last few seconds—which, in our analysis, correlated with the page/logo information.

has an average length of 13.32 words, and each video has the average paragraph length of 62.23 words. Each sentence is aligned to a clip of on average 18.33 seconds which covers on average 26.04% of the full video. However, the entire paragraph for each video on average describes 96.7% of the whole video, demonstrating that each paragraph annotation covers the majority of the video. Furthermore, we found that 22% of the temporal descriptions overlap, showing that our annotation allows co-occurring or simultaneous events. We divide our dataset in training (17098 videos), validation (999), test (1011) and blind test splits (1039). Each video in the training set has a single annotation, but videos in validation, test, and blind test splits have three temporally localized paragraph annotations, for evaluation. While the test set can be used to compare model variants in a paper, only the best model per paper should be evaluated on the blind test set annotations, which will only be possible on an evaluation server. Annotations for the blind test set will not be released.

To explore the different domains in our dataset vs. ActivityNet captions we use the normalized pointwise mutual information to identify the words most closely associated with each dataset. Highest ranked words for ActivityNet are almost exclusively sports related, whereas in our dataset they include animals, baby, and words related to social events such as weddings. Most dominant actions in ActivityNet are either sports or household activity related whereas actions in our dataset are related to social activities such as laughing, waving, cheering etc. Our analysis of the distribution of POS categories show that nouns are the most dominant category observed in the VideoStory captions dataset with 24% of the total tokens followed by verbs (18.5%), determiners (15.9%), adjectives (4.36%), adverbs (5.16%) and propositions (5.04%). We also observe the similar distribution of POS categories in ActivityNet captions.

We also find that ActivityNet has 50% of the videos where at least one segment in the video describes more than half of the video duration whereas in our dataset only 30% of videos have that phenomenon. In Figure 2, we show the distribution of sentence/segment annotations in time. The average number of (temporally localized) sentences is 4.67 compared to 3.65 in ActivityNet, despite having shorter videos, indicating the high information content of our videos.

In Table 3 we present all three paragraph annotations for a video showing a wedding ceremony. Out of 3 annotations, Annotation 2 is more descriptive compared to 1 and 3. However, it misses details about the presence of the photographer and taking the pictures.

Temporal Analysis. High quality video descriptions are more than bags of single-sentence captions; they should tell a coherent story. To identify the importance of sentence ordering or temporal coherence in our video paragraphs, we train a neural language model (Merity et al., 2017) on the training paragraphs of the VideoStory dataset and report perplexity on the correct order of sentences vs. randomly shuffled order of sentences in the descriptions created to understand the importance of temporal coherence in the video descriptions of our dataset. Results in Table 2 show that shuffled sentences have higher perplexity scores, demonstrating that order of sentences in the paragraphs are important for the coherence in the story.

3 Baseline Captioning Models

We explore learning to caption the videos using ground truth video segments.

Image Captioning Models. To understand if the temporal component of the video is contributing

Annotation 1: A bride walks down the aisle to her waiting bridegroom. As the bride walks, a photographer captures photos. At the end of the aisle the man giving the bride away shakes hands and hugs the bridegroom. The bride and bridegroom then interlock arms and face forward together.

Annotation 2: A large group of people have gathered inside of a room for a wedding. A woman walks down the aisle with a man slowly as people watch. The two of them get to the end of the aisle where a groom stands waiting.The man shakes hands with the groom and gives the woman a kiss on her forehead. The man who walked her down the aisle steps away towards the side of the room as the couple take each others arms.

Annotation 3: A groom is standing at the end of an aisle as a photographer takes a photo. The bride and father then come into view and walk down the aisle to the waiting groom. They stop at the grooms spot and the bride's father then shakes the grooms hand and gives a hug and walks to his spot. The groom then holds arms with the bride to begin the wedding ceremony.

Table 3: Example video description annotations in our VideoStory set. Each video has multiple paragraphs and localized time-interval annotations for every sentence in the paragraph.

to the description, we trained image captioning models on a frame sampled from the middle of the each segment of a video. We use the Show and Tell (Vinyals et al., 2015) image captioning architecture to generate captions.

Video Captioning Models. We study various video captioning models. First, we use sequence to sequence (seq-seq) recurrent neural network (RNN) model which has a two-layer encoder RNN to encode video features and a decoder RNN to generate descriptions. In the seq-seq approach we treat each description/segment individually and use an RNN decoder to describe each segment of the video, similar to Venugopalan et al. (2015), but using Gated Recurrent Units, GRUs, (Cho et al., 2014) for both the encoder and decoder.

In most videos, events are correlated with previous and future events. For example, for the first video description shown in Figure 1 once the girl is thrown into the water, she gets hold of herself, and the horse shakes off water on her. To capture such contextual correlations, we incorporate context from previous segment description into the captioning module. We build a model (seq-seq + context) which takes current segment video features and hidden representation of previous segment's sentence generation RNN at every timestamp in the decoder. For a given video segment, with hidden encoded video representation h_i^v and hidden representation of previous segment h_{i-1}^s, the concatenation of (h_i^v, h_{i-1}^s) is fed as input to the decoder that describes the segment (shown in Figure 3). Prior work has shown using previous video context has improved generated captions (Krishna et al., 2017).

Visual representation. For the image captioning models, we used features extracted from pre-trained ResNet-152 on ImageNet (He et al., 2016). For video captioning models we extract features from pre-trained 3D convolution ResNext-101 architecture trained on Kinetics (Kay et al., 2017), denoted as R3D, which achieved state-of-the-art results on various activity recognition tasks (Hara et al., 2018). Since a significant percentage of our videos has objects other than humans (e.g., animals) we also experiment with image-video fusion features(denoted by RNEXT, R3D) i.e., concatenation of ResNext-101 features extracted from pre-trained ImageNet with R3D features described above. We extract image features from the same frames which were used to extract R3D features.

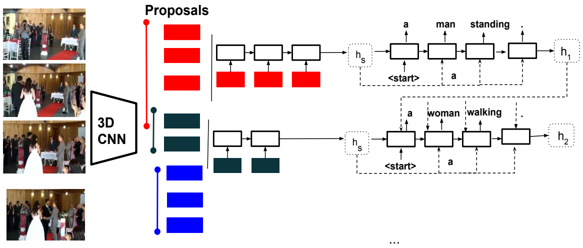

Figure 3: Our seq-seq+context model

4 Experiments and Results

For every segment, we set the maximum number of the sequence of features to 120 (i.e., 16X120 frames from the video) and maximum sentence length to 30. We trained using Adam optimizer with learning rate 0.0001. We use GRU as recurrent architecture to encode frames and decode captions with 512 dimensional hidden representation. We measure the captioning performance with most

GT (Ground Truth): A baby is playing outside with two dogs. The baby rolls the ball and the dog brings the ball back to the baby. The baby tosses the ball again and again for the dogs. One of the dogs walk away but the other stays and plays with the baby.	**I** (Image): The dog is standing on the bed . The dog is looking at the dog . The dog is walking on the ground . The dog is walking around the room .
seq-seq (RNEXT,R3D): A dog is walking in the water with a baby . The dog runs up and down the water . The dog runs up and down the slide . A baby is walking around the house with a baby .	**seq-seq+context** (RNEXT,R3D): A dog is standing in the middle of a house . The dog runs around the room and the dog jumps up and down . The dog is walking on the floor and the dog walks away . the girl runs around the house and the other dog runs away .

Table 4: Qualitative results: Descriptions generated by all variations of our baseline models.

Model	visual feat frame,video	B-3	B-4	M	R	C
I (single-frame)	RN-152, –	1.99	0.52	7.87	18.99	23.00
seq-seq	–,R3D	2.33	0.60	8.33	19.59	26.48
seq-seq + context	– ,R3D	2.78	0.78	9.20	21.24	30.80
seq-seq	RNEXT,R3D	2.63	0.79	8.44	19.89	27.64
seq-seq + context	RNEXT,R3D	**3.37**	**1.20**	**9.37**	**21.52**	**33.88**
trained on ActivityNet Captions						
seq-seq + context	RNEXT,R3D	1.68	0.49	8.48	19.40	22.12

Table 5: Captioning results from VideoStory Dataset using ground-truth test video segments. We report BLEU (B) and METEOR (M), ROUGE-L(R) and CIDEr (C). Best scores are in bold.

commonly-used evaluation metrics: BLEU{3,4}, METEOR, ROUGE-L, and CIDEr following previous works of image and video captioning (Papineni et al., 2002; Lin, 2004; Banerjee and Lavie, 2005; Vedantam et al., 2015).

In Table 5, we present the performance of our baseline models on VideoStory test dataset. We observe that models that consider context (seq-seq+context) from the previously generated sentence have better performance than the corresponding models without context (seq-seq), with both 3D convolution based features (R3D) as well as image-video fusion features (RNEXT,R3D). This indicates that our model benefited from contextual information, and that sentences in our stories are contextual, rather than independent.

To validate the strength of our baseline model, we train our best performing model on ActivityNet Captions. It achieves 10.92 (METEOR) and 43.42 (CIDEr) on the val set, close to state-of-the-art results of 11.06 and 44.71 by Zhou et al. (2018b), indicating that it is a strong baseline. However, when evaluating our ActivityNet model on our VideoStory dataset (Table 5, last row), we see significantly lower performance compared to a model trained on our dataset, highlighting the complementary nature of our dataset.

Our image only (single frame) model has the lowest scores across all metrics suggesting that a single image is not enough to generate contextual descriptions. We observed that our fusion models consistently outperform models with video-only R3D features, indicating features extracted using pre-trained ImageNet complement activity based R3D features. We show qualitative results from the variants of our models in Table 4. We observe that single frame models tend to repeat same captions and seq-seq model without context repeats phrases in the descriptions.

5 Conclusions

This paper introduces a dataset which we sourced from videos on social media and annotated with multi-sentence descriptions. We benchmark strong baseline approaches on the dataset, and our evaluations show that our dataset is complementary from prior work due to more diverse topics and the selection of engaging videos which tell a story. Our VideoStory dataset can serve as a good benchmark to build models for story understanding and multi-sentence video description.

Acknowledgements

We would like to thank Ranjay Krishna for providing the annotation interface used in Krishna et al. (2017) which we adapted to collect our dataset. We would also like to thank Haoqi Fan, Boris Vassilev, Jamie Ray, Sasha Sheng, Nikhila Ravi, and Evan Numbers for their help collecting the dataset, Devi Parikh for feedback on the annotation interface and Anna Rohrbach for useful feedback on drafts of this paper.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

David Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 190–200.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111.

Pradipto Das, Chenliang Xu, Richard F. Doell, and Jason J. Corso. 2013. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 2634–2641.

Sandra Eliza Fontes De Avila, Ana Paula Brandão Lopes, Antonio da Luz Jr, and Arnaldo de Albuquerque Araújo. 2011. Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters*, 32(1):56–68.

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. 2018. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA*, pages 18–22.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1233–1239.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. 2017. A hierarchical approach for generating descriptive image paragraphs. In *Computer Vision and Patterm Recognition (CVPR)*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *International Conference on Computer Vision (ICCV)*.

Yuncheng Li, Yale Song, Liangliang Cao, Joel R. Tetreault, Larry Goldberg, Alejandro Jaimes, and Jiebo Luo. 2016. TGIF: A new dataset and benchmark on animated GIF description. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4641–4650.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Michael Mateas and Phoebe Sengers. 2003. *Narrative intelligence*. J. Benjamins Pub.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *TACL*, 1:25–36.

Anna Rohrbach, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. In *Pattern Recognition - 36th German Conference, GCPR 2014, Münster, Germany, September 2-5, 2014, Proceedings*, pages 184–195.

Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. 2015. A dataset for movie description. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3202–3212.

Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele. 2017. Movie description. *International Journal of Computer Vision*, 123(1):94–120.

Gunnar A Sigurdsson, Xinlei Chen, and Abhinav Gupta. 2016a. Learning visual storylines with skipping recurrent neural networks. In *European Conference on Computer Vision*, pages 71–88. Springer.

Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. 2016b. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, pages 510–526.

Atousa Torabi, Christopher J. Pal, Hugo Larochelle, and Aaron C. Courville. 2015. Using descriptive video services to create a large data source for video annotation research. *CoRR*, abs/1503.01070.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence-video to text. In *Proceedings of the IEEE international conference on computer vision*, pages 4534–4542.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3156–3164. IEEE.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. MSR-VTT: A large video description dataset for bridging video and language. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 5288–5296.

Kuo-Hao Zeng, Tseng-Hung Chen, Juan Carlos Niebles, and Min Sun. 2016. Generation for user generated videos. In *European conference on computer vision*, pages 609–625. Springer.

Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. 2016. Video summarization with long short-term memory. In *European conference on computer vision*, pages 766–782. Springer.

Luowei Zhou, Chenliang Xu, and Jason J. Corso. 2018a. Towards automatic learning of procedures from web instructional videos. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.*

Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. 2018b. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8739–8748.

Cascaded Mutual Modulation for Visual Reasoning

Yiqun Yao[123], Jiaming Xu[12*], Feng Wang[1] and Bo Xu[1234]
[1]Institute of Automation, Chinese Academy of Sciences (CASIA). Beijing, China
[2]Research Center for Brain-inspired Intelligence, CASIA
[3]University of Chinese Academy of Sciences
[4]Center for Excellence in Brain Science and Intelligence Technology, CAS. China
`{yaoyiqun2014,jiaming.xu,feng.wang,xubo}@ia.ac.cn`

Abstract

Visual reasoning is a special visual question answering problem that is multi-step and compositional by nature, and also requires intensive text-vision interactions. We propose CMM: Cascaded Mutual Modulation as a novel end-to-end visual reasoning model. CMM includes a multi-step comprehension process for both question and image. In each step, we use a Feature-wise Linear Modulation (FiLM) technique to enable textual/visual pipeline to mutually control each other. Experiments show that CMM significantly outperforms most related models, and reach state-of-the-arts on two visual reasoning benchmarks: CLEVR and NLVR, collected from both synthetic and natural languages. Ablation studies confirm that both our multi-step framework and our visual-guided language modulation are critical to the task. Our code is available at `https://github.com/FlamingHorizon/CMM-VR`.

1 Introduction

It is a challenging task in artificial intelligence to perform reasoning with both textual and visual inputs. Visual reasoning task is designed for researches in this field. It is a special visual question answering (VQA) (Antol et al., 2015) problem, requiring a model to infer the relations between entities in both image and text, and generate a textual answer to the question correctly. Unlike other VQA tasks, questions in visual reasoning often contain extensive logical phenomena, and refer to multiple entities, specific attributes and complex relations. Visual reasoning datasets such as CLEVR (Johnson et al., 2017a) and NLVR (Suhr et al., 2017) are built on unbiased, synthetic images, with either complex synthetic questions or natural-language descriptions, facilitating in-depth analyses on reasoning ability itself.

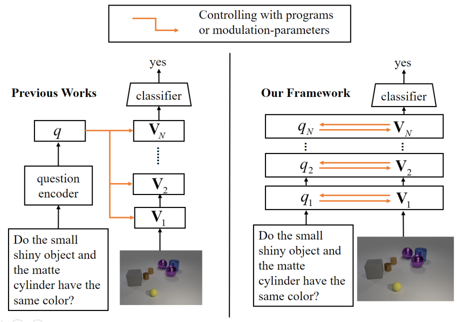

Figure 1: Connections and differences between previous "program-generating" works and our model: other models generate/control multi-step image-comprehension processes with single question representation, while we put more attention on language logics and let multi-modal information modulate each other in each step. The question and image are taken as a visual-reasoning example from CLEVR dataset.

Most previous visual reasoning models focus on using the question to guide the multi-step computing on visual features (which can be defined as a image-comprehension "program"). Neural Module Networks (NMN) (Andreas et al., 2016a,b) and Program Generator + Execution Engine (PG+EE) (Johnson et al., 2017b) learn to combine specific image-processing modules, guided by question semantics. Feature-modulating methods like FiLM (De Vries et al., 2017; Perez et al., 2018) control image-comprehension process using modulation-parameters generated from the question, allowing models to be trained end-to-end. However, the image-comprehension program in visual reasoning tasks can be extremely long and sophisticated. Using a single question representation to generate or control the whole image-comprehension process raises difficulties in learning. Moreover, since information comes from multiple modalities, it is not intuitive to assume

---

* Corresponding Author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 975–980
Brussels, Belgium, October 31 - November 4, 2018.

that one (language) is the "program generator", and the other (image) is the "executor". One way to avoid making this assumption is to perform multiple steps of reasoning with each modality being generator and executor alternately in each step. For these two reasons, we propose Cascaded Mutual Modulation (Figure 1), a novel visual reasoning model to solve the problem that previous "program-generating" models lack a method to use visual features to guide multi-step reasoning on language logics. CMM reaches state-of-the-arts on two benchmarks: CLEVR (complex synthetic questions) and NLVR (natural-language).

2 Related Work

Perez et al. (2018) proposed FiLM as an end-to-end feature-modulating method. The original ResBlock+GRU+FiLM structure uses single question representation, and conditions all image-modulation-parameters on it, without sufficiently handling multi-step language logics. In contrast, we modulate both image and language features alternately in each step, and condition the modulation-parameters on the representations from previous step. We design an image-guided language attention pipeline and use it in combination with FiLM in our CMM framework, and significantly outperform the original structure.

Other widely-cited works on CLEVR/NVLR include Stacked Attention Networks (SAN) (Yang et al., 2016), NMN (Andreas et al., 2016b), N2NMN (Hu et al., 2017), PG+EE (Johnson et al., 2017b) and Relation Networks (RN) (Santoro et al., 2017). The recent CAN model (Hudson and Manning, 2018) also uses multiple question representations and has strong performances on CLEVR. However, these representations are not modulated by the visual part as in our model.

In other VQA tasks, DAN (Nam et al., 2017) is the only multi-step dual framework related to ours. For comparison, in every time step, DAN computes textual and visual attention *in parallel* with the same key-vector, while we perform textual attention and visual modulation (instead of attention) in a *cascaded* manner.

3 Model

We review and extend FiLM in Section 3.1-3.2, and introduce CMM model in Section 3.3-3.4.

3.1 Visual Modulation

Perez et al. (2018) proposed **F**eature-w**i**se **L**inear **M**odulation (FiLM), an affine transformation on intermediate outputs of a neural network (v stands for visual):

$$FiLM^v(\mathbf{F}_{i,c}|\gamma_{i,c},\beta_{i,c}) = \gamma_{i,c}\mathbf{F}_{i,c} + \beta_{i,c}, \quad (1)$$

where $\mathbf{F}_{i,c}$ is the c-th feature map (C in total) generated by Convolutional Neural Networks (CNN) in the i-th image-comprehension step. Modulation-parameters $\gamma_{i,c}$ and $\beta_{i,c}$ can be conditioned on any other part of network (in their work the *single* question representation q). If the output tensor $\mathbf{V}_i$ of a CNN block is of size $C \times H \times W$, then $\mathbf{F}_{i,c}$ is a single slice of size $1 \times H \times W$. H and W are the height and width of each feature map.

Unlike (Perez et al., 2018), in each step i, we compute a new question vector q_i. Modulation-parameters γ_i and β_i ($C \times 1$ vectors, $\gamma_i = [\gamma_{i,1}, ..., \gamma_{i,C}]$, etc.) are conditioned on the previous question vector q_{i-1} instead of a single q:

$$\gamma_i, \beta_i = MLP^i(q_{i-1}). \quad (2)$$

MLP stands for fully connected layers with linear activations. The weights and biases are not shared among all steps.

3.2 Language Modulation

In each step i, we also apply FiLM to modulate every language "feature map". If the full question representation is a $D \times T$ matrix, a question "feature map" $\mathbf{f}_{i,d}$ is defined as a $1 \times T$ vector gathering T features along a single dimension. D is the hidden-state dimension of language encoder, and T is a fixed maximum length of word sequences.

$$FiLM^l(\mathbf{f}_{i,d}|\gamma_{i,d},\beta_{i,d}) = \gamma_{i,d}\mathbf{f}_{i,d} + \beta_{i,d}, \quad (3)$$

where l stands for language. Concatenated modulation-parameters γ_i and β_i ($D \times 1$) are conditioned on the visual features $\mathbf{V}_i$ computed in the same step:

$$\gamma_i, \beta_i = g_m(\mathbf{V}_i), \quad (4)$$

where g_m (Section 3.4) is an interaction function that converts 3-d visual features to language-modulation-parameters. The weights in g_m are shared among all N steps.

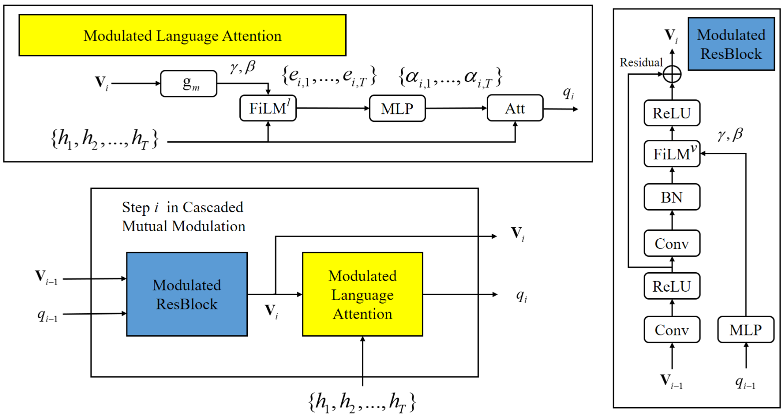

Figure 2: Details in CMM step i (middle), with a modulated ResBlock (right) and a modulated textual attention pipeline (top). Visual and textual features modulate each other in each step to compute new representations.

3.3 Cascaded Mutual Modulation

The whole pipeline of our model is built up with multiple *steps*. In each step i (N in total), previous question vector q_{i-1} and the visual features $\mathbf{V}_{i-1}$ are taken as input; q_i and $\mathbf{V}_i$ are computed as output. Preprocessed questions/images are encoded by language/visual encoders to form q_0 and $\mathbf{V}_0$.

In each step, we cascade a FiLM-ed ResBlock with a modulated textual-attention. We feed $\mathbf{V}_{i-1}$ into the ResBlock modulated by parameters from q_{i-1} to compute $\mathbf{V}_i$, and then control the textual attention process with modulation-parameters from $\mathbf{V}_i$, to compute the new question vector q_i. (Figure 2, middle).

Each ResBlock contains a 1×1 convolution, a 3×3 convolution, a batch-normalization (Ioffe and Szegedy, 2015) layer before FiLM modulation, followed by a residual connection (He et al., 2016). (Figure 2, right. We keep the same ResBlock structure as (Perez et al., 2018)). To be consistent with (Johnson et al., 2017b; Perez et al., 2018), we concatenate the input visual features $\mathbf{V}_{i-1}$ of each ResBlock i with two "coordinate feature maps" scaled from -1 to 1, to enrich representations of spatial relations. All CNNs in our model use ReLU as activation functions; batch-normalization is applied before ReLU.

After the ResBlock pipeline, we apply language modulation on the full language features $\{h_1, ..., h_T\}$ ($D \times T$) conditioned on $\mathbf{V}_i$ and rewrite along the time dimension, yielding:

$$e_{i,t} = FiLM^l(h_t | g_m(\mathbf{V}_i)), \qquad (5)$$

and compute visual-guided attention weights:

$$\alpha_{i,t} = \mathrm{softmax}_t(\mathbf{W}_i^{att} e_{i,t} + b_i^{att}), \qquad (6)$$

and weighted summation over time:

$$q_i = \sum_{t=1}^{T} \alpha_{i,t} h_t. \qquad (7)$$

In equation (6), $W_i^{att} \in \mathbb{R}^{1 \times D}$ and $b_i^{att} \in \mathbb{R}^{1 \times 1}$ are network weights and bias; h_t is the t-th language state vector ($D \times 1$), computed using a bi-directional GRU (Chung et al., 2014) from word embeddings $\{w_1, ..., w_T\}$. In each step i, the language pipeline does not re-compute h_t, but re-modulate it as $e_{i,t}$ instead. (Figure 2, top.)

3.4 Feature Projections

We use a function g_p to project the last visual features $\mathbf{V}_N$ into a final representation:

$$u_{final} = g_p(\mathbf{V}_N). \qquad (8)$$

g_p includes a convolution with K 1×1 kernels, a batch-normalization afterwards, followed by global max pooling over all pixels ($K = 512$).

We also need a module g_m (equation (4)) to compute language-modulations with $\mathbf{V}_i$, since $\mathbf{V}_i$ is 3-d features (not a weighted-summed vector as in traditional visual-attention). We choose g_m to have the same structure as g_p, except that K equals to the total number of modulation-parameters in each step. This design is critical (Section 4.3).

We use a fully connected layer with 1024 ReLU hidden units as our answer generator. It takes u_{final} as input, and predicts the most probable answer in the answer vocabulary.

Model	Overall	Count	Exist	Compare Numbers	Query Attribute	Compare Attribute
Human	92.6	86.7	96.6	86.5	95.0	96.0
SAN (Yang et al., 2016)	76.7	64.4	82.7	77.4	82.6	75.4
N2NMN (Andreas et al., 2016a)	83.7	68.5	85.7	84.9	90.0	88.7
PG+EE-9K	88.6	79.7	89.7	79.1	92.6	96.0
PG+EE-700K (Johnson et al., 2017b)	96.9	92.7	97.1	98.7	98.1	98.9
RN (Santoro et al., 2017)	95.5	90.1	97.8	93.6	97.9	97.1
COG-model (Yang et al., 2018)	96.8	91.7	99.0	95.5	98.5	98.8
FiLM	97.7	94.3	99.1	96.8	99.1	99.1
FiLM-raw (Perez et al., 2018)	97.6	94.3	99.3	93.4	99.3	99.3
DDRprog (Suarez et al., 2018)	98.3	96.5	98.8	98.4	99.1	99.0
CAN (Hudson and Manning, 2018)	98.9	97.1	**99.5**	**99.1**	99.5	**99.5**
CMM-single (ours)	98.6	96.8	99.2	97.7	99.4	99.1
CMM-ensemble (ours)	**99.0**	**97.6**	**99.5**	98.5	**99.6**	99.4

Table 1: Accuracies on CLEVR test set. N2NMN and PG+EE need extra supervision to train with reinforcement learning. FiLM-raw uses raw image as input (others use pre-extracted features). Another work (Mascharka et al., 2018) gets 99.1% accuracy but uses strong program supervision, which is a totally different setting.

4 Experiments

We are the first to achieve top results on both datasets (CLEVR, NLVR) with one structure. See Appendix for more ablation and visualization results.

4.1 CLEVR

CLEVR (Johnson et al., 2017a) is a commonly-used visual reasoning benchmark containing 700,000 training samples, 150,000 for validation and test. Questions in CLEVR cover several typical elements of reasoning: counting, comparing, querying the memory, etc. Many well-designed models on VQA have failed on CLEVR, revealing the difficulty to handle the multi-step and compositional nature of logical questions.

On CLEVR dataset, we embed the question words into a 200-dim continuous space, and use a bi-directional GRU with 512 hidden units to generate 1024-dim question representations. Questions are padded with NULL token to a maximum length $T = 46$. As the first-step question vector in CMM, q_0 can be arbitrary RNN hidden state in the set $\{h_1, ..., h_T\}$ (Section 3.3). We choose the one at the end of the unpadded question.

In each ResBlock, the feature map number C is set to 128. Images are pre-processed with a ResNet101 network pre-trained on ImageNet (Russakovsky et al., 2015) to extract $1024 \times 14 \times 14$ visual features (this is also common practice on CLEVR). We use a trainable one-layer CNN with 128 kernels (3×3) to encode the extracted features into $\mathbf{V}_0$ ($128 \times 14 \times 14$). Convolutional paddings are used to keep the feature map size to be 14×14 through the visual pipeline.

We train the model with an ADAM (Kingma and Ba, 2014) optimizer using a learning rate of 2.5e-4 and a batch-size of 64 for about 90 epoches, and switch to an SGD with the same learning rate and 0.9 momentum, fine-tuning for another 20 epoches. SGD generally brings around 0.3 points gains to CMM on CLEVR.

We achieve 98.6% accuracy with single model (4-step), significantly better than FiLM and other related work, only slightly lower than CAN, but CAN needs at least 8 model-blocks for >98% (and 12 for best). We achieve state-of-the-art of 99.0% with ensemble of 4/5/6 step CMM models. Table 1 shows test accuracies on all types of questions. The main improvements over program-generating models come from "Counting" and "Compare Numbers", indicating that CMM framework significantly enhances language (especially numeric) reasoning without sophisticated memory design like CAN.

4.2 NLVR

NLVR (Suhr et al., 2017) is a visual reasoning dataset proposed by researchers in NLP field. NLVR has 74,460 samples for training, 5,940 for validation and 5,934 for public test. In each sample, there is a human-posed natural language description on an image with 3 sub-images, and requires a false/true response.

We use different preprocessing methods on NLVR. Before training, we reshape NLVR images into 14×56 raw pixels and use them directly as visual inputs $\mathbf{V}_0$. For language part, we correct some obvious typos among the rare words (frequency < 5) in the training set, and pad the sentences to a maximum length of 26. Different from

CLEVR, LSTM works better than GRU on the real-world questions. For training, we use ADAM with a learning rate of 3.5e-4 and a batch-size of 128 for about 200 epoches, without SGD fine-tuning.

Our model (3-step, 69.9%) outperforms all proposed models on both validation and public test set, showing that CMM is also suitable for real-world languages (Table 2).

Model	Dev	Test-P
Text only	56.6	57.2
Image only	55.4	56.1
CNN+RNN (Suhr et al., 2017)	56.6	58.0
NMN (Andreas et al., 2016b)	63.1	66.1
FiLM (our run)	59.0	61.3
CNN-BiAtt (Tan and Bansal, 2018)	66.9	69.7
CMM-3-steps (ours)	**68.0**	**69.9**

Table 2: Accuracies on valid and test set of NLVR.

4.3 Ablation Studies

We list CMM ablation results in Table 3. Ablations on CLEVR show that CMM is robust to step number but sensitive to g_m structure because it's the key to multi-modal interaction. Section 3.4 is temporarily a best choice. CMM performs over 7-point higher than FiLM on NLVR in a setting of same hyper-parameters and ResBlocks, showing the importance of handling *language* logics (see also difficult CLEVR subtasks in Table 1).

CLEVR		NLVR		
Model	Val	Model	Dev	Test-P
5-step	98.4	FiLM-hyp	59.0	61.3
6-step	98.4	1-step	65.3	66.9
g_m-CNN	93.3	2-step	67.7	66.8
g_m w/o BN	94.1	3-step	68.5	68.4
g_m-NS	97.0	4-step	67.2	66.5
4-step	98.6	3-step-LSTM	68.0	69.9

Table 3: Ablation studies on CLEVR/NLVR. g_m-CNN means using 2-layer-CNN with 3×3 kernels, followed by concatenation and MLP, as g_m. BN means batch-normalization in g_m. NS means not sharing weights. "FiLM-hyp" uses all the same hyper-parameters as the 3-step CMM (both use GRU as question encoder).

4.4 A Case Study

We select an image-question pair from the validation set of CLEVR for visualization. In Table 4, we visualize the multi-step attention weights on question words, and the distribution of argmax position in the global max-pooling layer of g_p (equivalent to the last visual "attention map" although there isn't explicit visual attention in our image-comprehension pipeline). On the bottom right is the original image, and on the top right is the distribution of argmax positions in the global max-pooling, multiplied by the original image.

Words	Block 1	Block 2	Block 3	Visual Attention Map
<START>	0.017	0.030	0.052	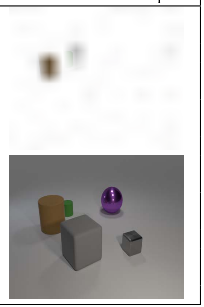
Is	0.006	0.048	0.501	
there	0.010	0.041	0.299	
a	0.006	0.031	0.015	
big	0.008	0.031	0.001	
brown	0.049	0.040	0.002	
object	0.149	0.265	0.003	
of	0.032	0.063	0.003	
the	0.030	0.029	0.004	
same	0.062	0.032	0.002	
shape	0.345	0.152	0.007	
as	0.233	0.031	0.001	
the	0.000	0.005	0.004	
green	0.003	0.014	0.004	
thing	0.012	0.075	0.011	
<END>	0.006	0.036	0.025	

Table 4: Visualization of CMM intermediate outputs on a sample from CLEVR validation set. We colour the largest attention weight with dark gray, and top four attention weights in the rest with light gray.

Our model attends to phrases "same shape as" and "brown object" in the first two reasoning steps. These phrases are meaningful because "same shape as" is the core logic in the question, and "brown object" is the key entity to generating the correct answer. In the last step, the model attends to the phrase "is there". This implicitly classifies the question into question-type "exist", and directs the answer generator to answer "no" or "yes". The visual map, guided by question-based modulation parameters, concentrates on the green and brown object correctly.

The result shows that visual features can guide the comprehension of question logics with textual modulation. On the other hand, question-based modulation parameters enable the ResBlocks to filter out irrelative objects.

5 Conclusion

We propose CMM as a novel visual reasoning model cascading visual and textual modulation in each step. CMM reaches state-of-the-arts on visual reasoning benchmarks with both synthetic and real-world languages.

Acknowledgements

We thank the reviewers for their insightful comments. This work is supported by the National Natural Science Foundation of China (61602479) and the Strategic Priority Research Program of the Chinese Academy of Sciences (XDBS01070000).

According to NLVR rules, we will run on the unreleased test set (Test-U) in the near future.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural networks for question answering. In *Proceedings of NAACL-HLT*, pages 1545–1554.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. 2017. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6597–6607.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. *CoRR, abs/1704.05526*, 3.

Drew A Hudson and Christopher D Manning. 2018. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017a. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017*, pages 1988–1997. IEEE.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017b. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. 2018. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. 2017. Dual attention networks for multimodal reasoning and matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 299–307.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4974–4983.

Joseph Suarez, Justin Johnson, and Fei-Fei Li. 2018. Ddrprog: A clevr differentiable dynamic reasoning programmer. *arXiv preprint arXiv:1803.11361*.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 217–223.

Hao Tan and Mohit Bansal. 2018. Object ordering with bidirectional matchings for visual reasoning. In *Proceedings of NAACL-HLT*.

Guangyu Robert Yang, Igor Ganichev, Xiao-Jing Wang, Jonathon Shlens, and David Sussillo. 2018. A dataset and architecture for visual reasoning with a working memory. *arXiv preprint arXiv:1803.06092*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29.

How agents see things:
On visual representations in an emergent language game

Diane Bouchacourt
Facebook AI Research
dianeb@fb.com

Marco Baroni
Facebook AI Research
mbaroni@fb.com

Abstract

There is growing interest in the language developed by agents interacting in emergent-communication settings. Earlier studies have focused on the agents' symbol usage, rather than on their representation of visual input. In this paper, we consider the referential games of Lazaridou et al. (2017), and investigate the representations the agents develop during their evolving interaction. We find that the agents establish successful communication by inducing visual representations that almost perfectly align with each other, but, surprisingly, do not capture the conceptual properties of the objects depicted in the input images. We conclude that, if we are interested in developing language-like communication systems, we must pay more attention to the visual semantics agents associate to the symbols they use.

1 Introduction

There has recently been a revival of interests in language emergence simulations involving agents interacting in visually-grounded games. Unlike earlier work (e.g., Briscoe, 2002; Cangelosi and Parisi, 2002; Steels, 2012), many recent simulations consider realistic visual input, for example, by playing referential games with real-life pictures (e.g., Jorge et al., 2016; Lazaridou et al., 2017; Havrylov and Titov, 2017; Lee et al., 2018; Evtimova et al., 2018). This setup allows us to address the exciting issue of whether the needs of goal-directed communication will lead agents to associate visually-grounded conceptual representations to discrete symbols, developing natural-language-like word meanings. However, while most studies present some analysis of the agents' symbol usage, they pay little or no attention to the representation of the visual input that the agents develop as part of their evolving interaction.

We study here agent representations following the model and setup of Lazaridou et al. (2017). This is an ideal starting point, since it involves an extremely simple signaling game (Lewis, 1969), that is however played with naturalistic images, thus allowing us to focus on the question of how the agents represent these images, and whether such representations meet our expectations for natural word meanings. In their first game, Lazaridou's Sender and Receiver are exposed to the same pair of images, one of them being randomly marked as the "target". The Sender always sees the target in the left position, and it must pick one discrete symbol from a fixed vocabulary to send to the Receiver. The Receiver sees the images in random order, together with the sent symbol, and it tries to guess which image is the target. In case of success, both players get a payoff of 1. Since an analysis of vocabulary usage brings inconclusive evidence that the agents are using the symbols to represent natural concepts (such as *beaver* or *bayonet*), Lazaridou and colleagues next modify the game, by presenting to the Sender and the Receiver different images for each of the two concepts (e.g., the Sender must now signal that the target is a *beaver*, while seeing a different beaver from the one shown to the Receiver). This setup should encourage concept-level thinking, since the two agents should not be able to communicate about low-level perceptual characteristics of images they do not share. Lazaridou and colleagues present preliminary evidence suggesting that, indeed, agents are now developing conceptual symbol meanings. We replicate Lazaridou's games, and we find that, in both, the agents develop successfully aligned representations that, however, are not capturing conceptual properties at all. In what is perhaps our most striking result, agents trained in either version of the game succeed at communicating about pseudo-images generated from random noise (Fig. 2). We conclude that, if we want interactive agents to

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 981–985
Brussels, Belgium, October 31 - November 4, 2018.

develop a vocabulary of words denoting natural meanings, more attention must be paid to the way in which they are representing their perceptual input.

2 Experimental setup

Architecture We re-implement Lazaridou's Sender and Receiver architectures (using their better-behaved "informed" Sender). Both agents are feed-forward networks. The **Sender** takes image representations as input, it projects them into its own representational space, compares them, and finally outputs a probability distribution over vocabulary symbols, from which a single discrete symbol is then sampled. We report here results obtained with an output vocabulary of 100 symbols, but the same patterns were observed using a range of sizes from 2 to $1,000$. The **Receiver** takes as input the target and distractor input image representations in random order, as well as the symbol produced by the sender (as a vocabulary-sized one-hot vector). It embeds the images and the symbol into its own representational space, where it performs a symbol-to-image comparison, producing a probability distribution over the two images, one of which is selected by sampling from this distribution. If the Receiver selected the target image, a reward of 1 is assigned to both agents. The whole architecture is jointly trained by letting the agents play, and updating their parameters with Reinforce (Williams, 1992). See Lazaridou et al. (2017) for details.

Data Following Lazaridou et al. (2017), for each of the 463 concepts they used, we randomly sample 100 images from ImageNet (Deng et al., 2009). We construct $50,000$ mini-batches of 32 image pairs during training and $1,024$ pairs for validation. We construct a held-out test set in the same way by sampling 10 images per concept from ImageNet (for 2 concepts, we were not able to assemble enough further images), for a total of $4,610$. We compute RSA scores (see below) on the cross-product of these images. We also use the held-out set to construct mini-batches of images pairs to compute test performance. Following Lazaridou, the images are passed through a pre-trained VGG ConvNet (Simonyan and Zisserman, 2015). The input vector fed to the agents is the second-to-last 4096-D fully connected layer[1].

Games We re-implement both Lazaridou's **same-image** game, where Sender and Receiver are shown the same two images (always of different concepts), and their **different-image** game, where the Receiver sees different images than the Sender's. We repeat all experiments using 100 random initialization seeds. As we faithfully reproduced the setup of Lazaridou et al. (2017), we refer the reader there for hyper-parameters and training details.

3 Experiments

We first asked in which way playing the game affects the way agents "see" the input data, that is, in which way their image embeddings differ from the input image representations, and from each other. Concerning Sender and Receiver, a reasonable expectation is that successful communication implies a convergence of representations. How should these representations relate to the input? Recall that input representations are from one of the top layers of a state-of-the-art ConvNet trained on ImageNet concept categorization, and the top layers of such networks are known to capture high-level concept semantics (Zeiler and Fergus, 2014). The game image pairs are always sampled from different concepts. So, it would make sense for the agents to simply learn to carry through the similarity structure of the input space, in order to communicate about distinct concepts. Consequently, we predicted that, as training proceeds, Sender and Receiver representations will become closer to each other, and to the input ones.

In order to compare the similarity structure of input, Sender and Receiver spaces, we borrow *representational similarity analysis* (RSA) from computational neuroscience (Kriegeskorte et al., 2008). Given two sets r_1 and r_2 of representations of the same item collection (e.g., r_1 is the collection of input images mapped in Sender embedding space and r_2 is the same collection represented by Receiver), we first compute s_1 as all possible pairwise (cosine) similarities between the representations in r_1, and s_2 as those in r_2. We then compute the (Spearman) correlation between the similarity vectors s_1 and s_2. This latter value, which we will call *RSA score*, measures the global agreement between s_1 and s_2, relative to the chosen input collection. If N is the number of items in the collection that we compute representations for, both similarity vectors s_1 and s_2 are of length

---

[1] We found very similar results with the top 1000-D softmax layer.

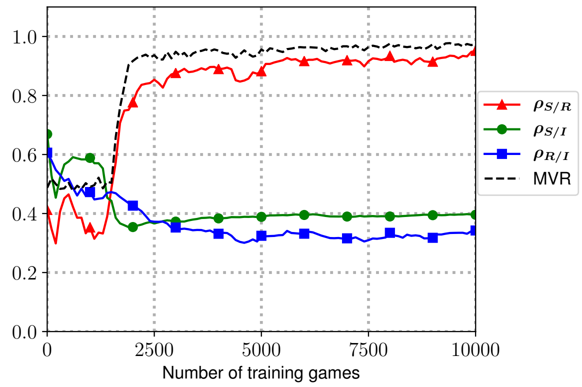

Figure 1: RSA scores of the two agents ($\rho_{S/R}$), and of each agent with the input ($\rho_{S/I}$ and $\rho_{R/I}$), during the first $10,000$ training games. S refers to Sender, R to Receiver, I to input. Values remain stable until end of training. Best viewed in color.

$N(N-1)$. Therefore, it is not necessary for representations r_1 and r_2 to belong to the same space (for example, in our case, input and agent vectors have different dimensionality).

Figure 1 shows RSA and mean validation reward (MVR) development curves for the cross-validated best seed in the same-image game. At the beginning of training, the RSA scores are non-zeros, which is expected as the two agents architectures are similar and randomly initialized the same way. They are also somewhat correlated with the input, which we attribute to the fact that untrained neural networks can already extract relevant image features (Jarrett et al., 2009). As training converges, Sender and Receiver similarity spaces also converge. However, contrary to our prediction, the agent similarity spaces are not strongly correlated with the input visual space. We note that, during the first few hundred games, the Sender (green curve) aligns with the input, but the Receiver (blue curve) does not. Therefore, it seems that, in order to establish communication, the two agents have to drift from the input. Indeed, when communication is successfully established at the end of training,[2] the two agents have a RSA score of $\rho_{S/R} = 0.98$. However either agent's score with the input is a much lower $\rho_{S/I} = \rho_{R/I} = 0.33$.[3] On the contrary, when the agents fail to establish communication, by the end of training their RSA score is just $\rho_{S/R} = 0.39$, but they stay closer to the input ($\rho_{S/I} = 0.58$ and $\rho_{R/I} = 0.42$).[4]

The drift of the agents from input similarity could be attributed to the characteristics of the game they are playing. Since they are only asked to distinguish between pictures of different concepts, they have no incentive to keep different instances of a concept distinct (if the agents are never asked to distinguish one dog from another, they might eventually become unable to tell dogs apart). That is, we might be assisting to the inception of a form of categorical perception (Goldstone and Hendrickson, 2010), whereby the agents lose sensitivity to within-category differences. If this is the case, we should observe that same-concept image similarity is *higher* in Sender (or Receiver) space with respect to input space. However, this turns out not to be the case. To the contrary, average pairwise same-concept similarity is consistently *lower* in Sender space than in the input (mean z-normalized same-concept similarity in input space is at 1.94 vs. 0.57 in Sender space, averaged across successful seeds). A similar effect is observed by looking at higher-class (*mammal*, *furniture*, etc.) similarities: images from the same classes become less similar in Sender space (0.61 z-normalized within-class input similarity vs. 0.30 in Sender space). This suggests that the agents are becoming less proficient at capturing the similarity among instances of the same concept or of the same class. The same conclusion is qualitatively supported by the pairs of images that underwent the largest shift between input and Sender space. For example, for two test images of avocados which have an input similarity of 0.82 (and are reasonably similar to the human eye), the Sender similarity is at the low value of -0.27 (Receiver similarity is -0.59). Contrarily, for an image of a cabin in a field and an image of a telephone that have an intuitively correct very low input similarity of 0.02, the Sender similarity for these images is 0.94 (Receiver similarity is 0.95).

Lazaridou et al. (2017) designed their second game to encourage more general, concept-like referents. Unfortunately, we replicate the anomalies above in the different-image setup, although to a less marked extent. When successful communication is established at the end of training, the agents have $\rho_{S/R} = 0.90$. But again, the agents' representation do not align with the input space: their scores with the input are at lower values of $\rho_{S/I} =$

---

[2] We consider training successful if MVR $\geq$ 80%.

[3] Values averaged over the 96 successful seeds.

[4] Values averaged over the 4 failing seeds.

		Test		
		Same im.	Diff. im.	Noise
Train	Same im.	100	72	95
	Diff. im.	98	83	87

Table 1: Percentage average rewards on the same-image, different-image and noise test sets for agents trained in the same- and different-image games (chance level at 50%). For each game, values are averaged on 10 test runs consisting of $1,000$ games of mini-batches of 32 image pairs, using the cross-validated best seed.

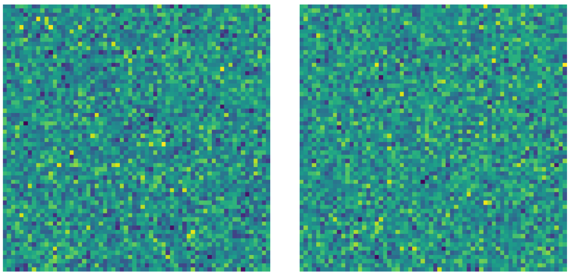

Figure 2: Noise vectors agents trained on the same-image game successfully communicate about.

0.40 and $\rho_{R/I} = 0.37$.[5] In case of communication failure, by the end of training their RSA score is at the lower value of $\rho_{S/R} = 0.74$, and their values with respect to the input are $\rho_{S/I} = 0.36$ and $\rho_{R/I} = 0.34$.[6] Again, same-concept images drift apart in agent space, although now to a lesser extent (1.94 z-normalized mean similarity in input space vs. 1.07 in Sender space). More encouragingly, we don't find the same pattern for within-class mean similarities (0.61 input space vs. 0.75 Sender space).

We must conjecture that the agents are comparing low-level properties of the image pairs, independently of the game they play. As an extreme way to test this, we look at how agents trained to play the two games behave when tested with input pairs that are just random noise vectors drawn from a standard Normal distribution.[7] If the agents are indeed indifferent to the objects represented by images, the radical shift in the nature of the input to the game should not affect them much.

Results are shown in Table 1. We confirm that the same-image game is the easiest, and we observe that agents trained in one game perform reasonably well on the other. More importantly, no matter which game they are trained on, the agents perform very well on noise input! This confirms our hypothesis that the Sender and Receiver are able to communicate about input data that contain no conceptual content at all, which in turn suggests that they haven't extracted any concept-level information (e.g., features that would allow them to recognize instances of the *dog* or *chair* category) during training. To get a sense of the sort of noise pairs agents succeed to communicate about, Figure 2 provides an example.

Finally, we draw $1,000$ noise pairs (z_1, z_2), and present each to the Sender with either z_1 or z_2 as target. We then compare, pair by pair, whether the highest probability symbol changes when the target is swapped. We average across 10 random runs using the best cross-validated seed. In both versions of the game, for more than 99% of the pairs, the symbol with highest probability changes when the target is swapped. This suggests that the agents perform a *relative* comparison of the two inputs, rather than an absolute one, in line with the general conclusion that they are not using the vocabulary to denote stable conceptual properties of the objects depicted in the images.

4 Discussion

Existing literature in game theory already showed that convergence towards successful communication is ensured under specific conditions (see Skyrms (2010) and references therein). However, the important contribution of Lazaridou et al. (2017) is to play a signaling game with real-life images instead of artificial symbols. This raises new empirical questions that are not answered by the general mathematical results, such as: When the agents do succeed at communicating, what are the input features they rely upon? Do the internal representations they develop relate to the conceptual properties of the input? Our study suggests that the agents' representations are not capturing general conceptual properties of different objects, but they are rather specifically tuned to successfully distinguish images based on inscrutable low-level relational properties.

Interestingly, our conclusions can be aligned with findings in psycholinguistic experimental literature on dialogue. In order to achieve communication, the agents develop a form of ''conceptual

---

[5] Values averaged over the 19 successful seeds.

[6] Values averaged over the 81 failing seeds.

[7] As during training inputs are divided by their norm, we also normalize each noise vector, so multiple noise variances would have no effect.

pact" (Brennan and Clark, 1996): Their internal representations align while at the same time drifting away from human-level properties of the input. The agents agree on a shared use of the vocabulary, that does not correspond to concepts in the input data.

In future work, we would like to encourage the development of more natural word meanings by enforcing the agent representations to stay more faithful to the perceptual input they receive. Moving ahead, it is fundamental to design setups where agents would have stronger reasons to develop human-like communication strategies.

Acknowledgments

We thank Angeliki Lazaridou, Douwe Kiela and Calvin Lee for their useful discussions and insights. We also thank Francisco Massa for his help on setting up the experiments.

References

Susan E. Brennan and Herbert H. Clark. 1996. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22:1482–1493.

Ted Briscoe, editor. 2002. *Linguistic evolution through language acquisition*. Cambridge University Press, Cambridge, UK.

Angelo Cangelosi and Domenico Parisi, editors. 2002. *Simulating the evolution of language*. Springer, New York.

Jia Deng, Wei Dong, Richard Socher, Lia-Ji Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of CVPR*, pages 248–255, Miami Beach, FL.

Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. 2018. Emergent communication in a multi-modal, multi-step referential game. In *Proceedings of ICLR Conference Track*, Vancouver, Canada. Published online: `https://openreview.net/group?id=ICLR.cc/2018/Conference`.

Robert Goldstone and Andrew Hendrickson. 2010. Categorical perception. *Wiley Interdisciplinary Reviews: Cognitive Science*, 1(1):69–78.

Serhii Havrylov and Ivan Titov. 2017. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Proceedings of NIPS*, pages 2149–2159, Long Beach, CA, USA.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. 2009. What is the best multi-stage architecture for object recognition? In *Proceedings of the 12th International Conference on Computer Vision*, pages 2146–2153.

Emilio Jorge, Mikael Kågebäck, and Emil Gustavsson. 2016. Learning to play Guess Who? and inventing a grounded language as a consequence. In *Proceedings of the NIPS Deep Reinforcement Learning Workshop*, Barcelona, Spain. Published online: `https://sites.google.com/site/deeprlnips2016/`.

Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. 2008. Representational similarity analysis: Connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2(4):1–28.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2017. Multi-agent cooperation and the emergence of (natural) language. In *Proceedings of ICLR Conference Track*, Toulon, France. Published online: `https://openreview.net/group?id=ICLR.cc/2017/conference`.

Jason Lee, Kyunghyun Cho, Jason Weston, and Douwe Kiela. 2018. Emergent translation in multi-agent communication. In *Proceedings of ICLR Conference Track*, Vancouver, Canada. Published online: `https://openreview.net/group?id=ICLR.cc/2018/Conference`.

David Lewis. 1969. *Convention*. Harvard University Press, Cambridge, MA.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of ICLR Conference Track*, San Diego, CA. Published online: `http://www.iclr.cc/doku.php?id=iclr2015:main`.

Brian Skyrms. 2010. *Signals: Evolution, learning, and information*. Oxford University Press, Oxford, UK.

Luc Steels, editor. 2012. *Experiments in Cultural Language Evolution*. John Benjamins, Amsterdam.

Ronald Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Matthew Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of ECCV (Part 1)*, pages 818–833, Zurich, Switzerland.

Attention-Based Capsule Networks with Dynamic Routing for Relation Extraction

Ningyu Zhang[1] **Shumin Deng**[2,3] **Zhanlin Sun**[2] **Xi Chen**[2]
Wei Zhang[3,4] **Huajun Chen**[2*]
1. Artificial Intelligence Research Institute, Zhejiang Lab, China
2. College of Computer Science and Technology, Zhejiang University, China
3. Alibaba-Zhejiang University Frontier Technology Research Center, China
4. Alibaba Group, China
zhangningyu@zju.edu.cn

Abstract

A capsule is a group of neurons, whose activity vector represents the instantiation parameters of a specific type of entity. In this paper, we explore the capsule networks used for relation extraction in a multi-instance multi-label learning framework and propose a novel neural approach based on capsule networks with attention mechanisms. We evaluate our method with different benchmarks, and it is demonstrated that our method improves the precision of the predicted relations. Particularly, we show that capsule networks improve multiple entity pairs relation extraction[1].

1 Introduction

This paper focus on the task of relation extraction. One popular method for relation extraction is the multi-instance multi-label learning framework (MIML) (Surdeanu et al., 2012) with distant supervision, where the mentions for an entity pair are aligned with the relations in Freebase (Bollacker et al., 2008). The recently proposed neural network (NN) models (Zeng et al., 2014; Ye et al., 2017; Yang et al., 2018; Wang et al., 2018a) achieve state-of-the-art performance. However, despite the great success of these NNs, some disadvantages remain. First, the existing models focus on, and heavily rely on, the quality of instance representation. Using a vector to represent a sentence is limited because languages are delicate and complex. Second, CNN subsampling fails to retain the precise spatial relationships between higher-level parts. The structural relationships such as the positions in sentences are valuable. Besides, existing aggregation operations summarizing the sentence meaning into a fixed-size vector such as max or average pooling are lack of guidance by task information. Self-attention (Lin et al., 2017) can select task-dependent information. However, the context vectors are fixed once learned (Gong et al., 2018).

More importantly, most state-of-the-art systems can only predict one most likely relation for a single entity pair. However, it is very common that one sentence may contain multiple entity pairs and describe multiple relations. It is beneficial to consider other relations in the context while predicting the relations (Sorokin and Gurevych, 2017). For example, given the sentence "*[**Swag It Out**] is the official debut single by American [**singer**] [**Zendaya**]*", our model can predict multi-relations for these multiple entity pairs simultaneously.

In our work, we present a novel architecture based on capsule networks (Sabour et al., 2017) for relation extraction. We regard the aggregation as a routing problem of how to deliver the messages from source nodes to target nodes. This process enables the capsule networks to decide what and how much information need to be transferred as well as identify complex and interleaved features. Furthermore, the capsule networks convert the multi-label classification problem into a multiple binary classification problem. We also import word-level attention by considering the different contribution of the words. The experimental results show that our model achieves improvements in both single and multiple relation extraction.

2 Related Work

Neural Relation Extraction: In the recent years, NN models have shown superior performance over approaches using hand-crafted features in various tasks. CNN is the first one of the deep learning models that have been applied to relation ex-

---

[*] Corresponding author.

[1] In this paper, multiple entity pairs relation extraction refers to multiple entity pairs in a single sentence and each pair of entities contains only one relation label.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 986–992
Brussels, Belgium, October 31 - November 4, 2018.

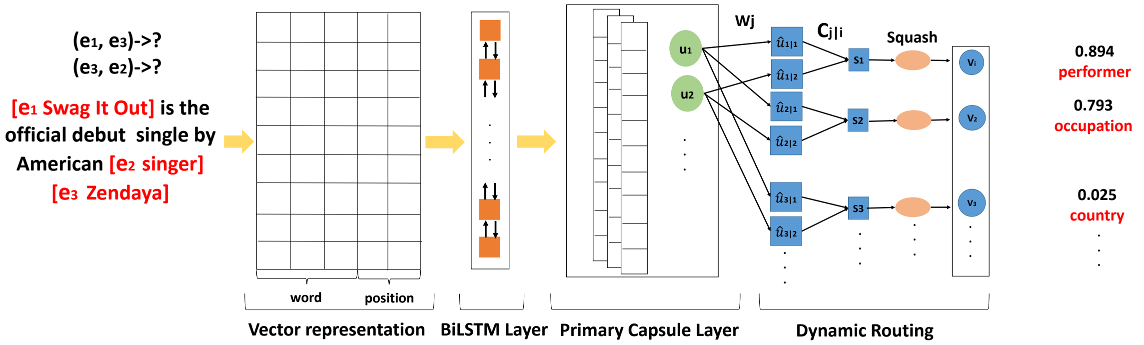

Figure 1: Architecture of capsule networks for relation extraction

traction (Santos et al., 2015). Variants of convolutional networks include piecewise-CNN (PCNN) (Zeng et al., 2015), instance-level selective attention CNN (Lin et al., 2016), rank CNN (Ye et al., 2017), attention and memory CNN (Feng et al., 2017) and syntax-aware CNN (He et al., 2018). Recurrent neural networks (RNN) are another popular choice, and have been used in recent works in the form of attention RNNs (Zhou et al., 2016), context-aware long short-term memory units (LSTMs) (Sorokin and Gurevych, 2017), graph-LSTMs (Peng et al., 2017) and ensemble LSTMs (Yang et al., 2018).

Capsule Network: Recently, the capsule network has been proposed to improve the representation limitations of CNNs and RNNs. (Sabour et al., 2017) replaced the scalar-output feature detectors of CNNs with vector-output capsules and max-pooling with routing-by-agreement. (Hinton et al., 2018)) proposed a new iterative routing procedure among capsule layers, based on the EM algorithm. For natural language processing tasks, (Zhao et al., 2018) explored capsule networks for text classification. (Gong et al., 2018) designed two dynamic routing policies to aggregate the outputs of RNN/CNN encoding layer into a final encoding vector. (Wang et al., 2018b) proposed a capsule model based on RNN for sentiment analysis. To the best of our knowledge, there has been no work that investigates the performance of capsule networks in relation extraction tasks at present.

3 Methodology

In this section, we first introduce the MIML framework, and then describe the model architecture we propose for relation extraction, which is shown in Figure 1.

3.1 Preliminaries

In MIML, the set of text sentences for the single entity pair or multiple entity pairs[2] (maximum two entity pairs in this paper) is denoted by $X = \{x_1, x_2, ..., x_n\}$. Assumed that there are E predefined relations (including NA) to extract. Formally, for each relation r, the prediction target is denoted by $P(r|x_1, ..., x_n)$.

3.2 Neural Architectures

Input Representation: For each sentence x_i, we use pretrained word embeddings to project each word token onto the d_w-dimensional space. We adopt the position features as the combinations of the relative distances from the current word to M entities and encode these distances in M d_p-dimensional vectors[3]. For single entity pair relation extraction, $M = 2$; for multiple entity pairs relation extraction, we limit the maximum number of entities in a sentence to four (i.e. two entity pairs). As three entities in one instance is possible when two tuples have a common entity, we set the relative distance to the missing entity to a very large number. Finally, each sentence is transformed into a matrix $x_i = \{w_1, w_2, ..., w_L\} \in R^{L \times V}$, where L is the sentence length with padding and $V = d_w + d_p * M$.

Bi-LSTM Layer: We make use of LSTMs to deeply learn the semantic meaning of a sentence. We concatenate the current memory cell hidden state vector h_t of LSTM from two directions as the output vector $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t] \in R^{2B}$ at time t, where B denotes the dimensionality of LSTM.

---

[2] Since the number of sentences with multiple entity pairs is relatively small, we make use of all the sentences as training samples.

[3] We also adopt an attention mechanism similar to word-level attention in Bi-LSTM layer by considering the different contribution of the M position embeddings.

We import word-level attention mechanism as only a few words in a sentence that are relevant to the relation expressed (Jat et al., 2018). The scoring function is $g_t = h_t \times A \times r$, where $A \in R^{E \times E}$ is a square matrix and $r \in R^{E \times 1}$ is a relation vector. Both A and r are learned. After obtaining g_t, we feed them to a softmax function to calculate the final importance $\alpha_t = softmax(g_t)$. Then, we get the representation $\tilde{x}_t = \alpha_t h_t$.

For a given bag of sentences, the learning is done using the setting proposed by (Zeng et al., 2015), where the sentence with highest probability of expressing the relation in a bag is selected to train the model in each iteration.

Primary Capsule Layer: Suppose $u_i \in R^d$ denotes the instantiated parameters set of a capsule, where d is the dimension of the capsule. Let $W^b \in R^{2 \times 2B}$ be the filter shared across different windows. We have a window sliding each 2-gram vector in the sequence $\tilde{x} \in R^{L \times 2B}$ with stride 1 to produce a list of capsules $U \in R^{(L+1) \times C \times d}$, totally $C \times d$ filters.

$$u_{ij} = squash(W_i^b \otimes S_{j-1:j} + b_1) \qquad (1)$$

where $0 \leq i \leq C \times d$, $0 \leq j \leq L+1$, $squash(x) = \frac{||x||^2}{0.5+||x||^2}\frac{x}{||x||}$, b_1 is the bias term. For all $C \times d$ filters, the generated capsule feature maps can be rearranged as $U = \{u_1, u_2, ..., u_{(L+1) \times C}\}$, where totally $(L+1) \times C$ d-dimensional vectors are collected as capsules.

Algorithm 1 Dynamic Routing Algorithm

```
procedure ROUTING($\hat{u}_{j|i}, \hat{a}_{j|i}, r, l$)
Initialize the logits of coupling coefficients $b_{j|i} = 0$
for $r$ iterations do
    for all capsule $i$ in layer $l$ and capsule $j$ in layer $l+1$ do
        $c_{j|i} = \hat{a}_{j|i} \cdot softmax(b_{j|i})$
    for all capsule $j$ in layer $l+1$ do
        $v_j = squash(\sum_i c_{j|i}\hat{u}_{j|i}), a_j = ||v_j||$
    for all capsule $i$ in layer $l$ and capsule $j$ in layer $l+1$ do
        $b_{j|i} = b_{j|i} + \hat{u}_{j|i} \cdot v_j$
return $v_j, a_j$
```

Dynamic Routing: We explore the transformation matrices to generate the prediction vector $u_{j|i} \in R^d$ from a child capsule i to its parent capsule j. The transformation matrices share weights $W^c \in R^{E \times d \times d}$ across the child capsules, where E is the number relations (parent capsules) in the layer above. Formally, each corresponding vote can be computed by:

$$\hat{u}_{j|i} = W_j^c u_i + \hat{b}_{j|i} \in R^d \qquad (2)$$

The basic idea of dynamic routing is to design a nonlinear map:

$$\{\hat{u}_{j|i} \in R^d\}_{i=1,...,H, j=1,...,E} \mapsto \{v_j \in R^d\}_{j=1}^{E}$$

where $H = (L+1) \times C$.

Inspired by (Zhao et al., 2018), we attempt to use the probability of existence of parent capsules to iteratively amend the connection strength, which is summarized in Algorithm 1. The length of the vector v_j represents the probability of each relation. We use a separate margin loss L_k for each relation capsule k:

$$\begin{aligned} L_k &= Y_k max(0, m^+ - ||v_k||)^2 \\ &+ \lambda(1 - Y_k) max(0, ||v_k|| - m^-)^2 \end{aligned} \qquad (3)$$

where $Y_k = 1$ if the relation k is present, $m^+ = 0.9$, $m^- = 0.1$ and $\lambda = 0.5$. The total loss can be formulated as: $L_{total} = \sum_{k=1}^{E} L_k$

3.3 Prediction

For single entity pair relation extraction, we calculate the length of the vector v_j which represents the probability of each relation. For multiple entity pairs relation extraction, we choose relations with top two probability meanwhile bigger than the threshold (We empirically set the threshold 0.7). Finally, we may get one or two predicted relations r. Given entity pair (e_1, e_2), in order to choose which relationship the tuple belongs to, we adopt the pretrained embeddings of entities and relations[4] and calculate $r_k = \arg\min_k |t - h - r_k|$, where t, h are the embeddings of entities e_1, e_2 respectively and r_k is the relation embedding. The relation with the closest embedding to the entity embedding difference is the predicted category.

4 Experiments

We test our model on the NYT dataset (NYT) developed by (Riedel et al., 2010) for single entity pair relation extraction and the Wikidata dataset (Sorokin and Gurevych, 2017) for multiple entity pairs relation extraction. We exclude sentences longer than L . All code is implemented in Tensorflow (Abadi et al., 2016). We adopt the Adam

---

[4] http://openke.thunlp.org

optimizer (Kingma and Ba, 2014) with learning rate 0.001, batch size 128, LSTMs' unit size 300, $L = 120$, $d_p = 5$, $d = 8$, $C = 32$, dropout rate 0.5, routing iteration 3.

4.1 Practical Performance

NYT dataset (Single entity pair):

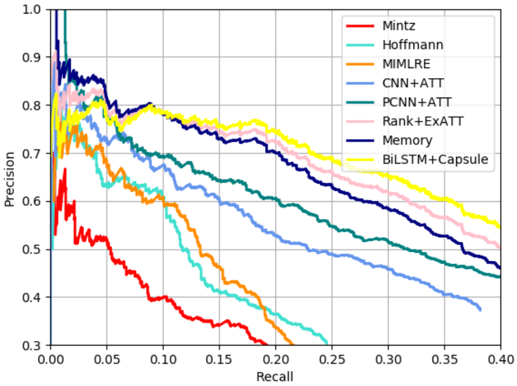

Figure 2: Performance comparison on the NYT dataset.

We utilize the word embeddings released by (Lin et al., 2016)[5]. The precision-recall curves for different models on the test set are shown in Figure 2. Our model BiLSTM+Capsule achieves comparable results compared with all baselines, where Mintz refers to (Mintz et al., 2009), Hoffmann refers to (Hoffmann et al., 2011), MIMLRE refers to (Surdeanu et al., 2012), CNN+ATT refers to (Zeng et al., 2015), PCNN+ATT refers to (Lin et al., 2016), Rank+ExATT refers to (Ye et al., 2017) and Memory refers to (Feng et al., 2017). We also show the precision numbers for some particular recalls as well as the AUC in Table 1, where our model generally leads to better precision. Interestingly, we observe our model achieve comparable results to predict multi-relation compared with Rank+ExATT in Figure 3. Given an entity tuple (South Korea, Seoul) which has two relations: /location/./administrative_divisions and /location/./capital. We observe these two relations have the highest scores among the other relations in our model which demonstrate the ability of multi-relations prediction.

Wikidata dataset (Multiple entity pairs):

We train word embeddings using Glove (Pennington et al., 2014)[6] on the Wikipedia Corpus.

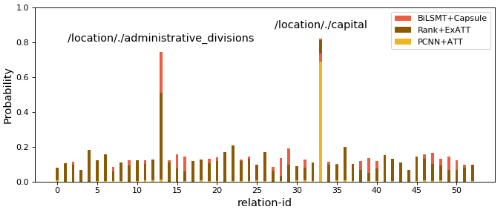

Figure 3: Normalized output relation scores.

Table 1: Precisions on the NYT dataset.

Recall	0.1	0.2	0.3	0.4	AUC
PCNN+ATT	0.698	0.606	0.518	0.446	0.323
Rank+ExATT	0.789	0.726	0.620	0.514	0.395
Our Model	0.788	**0.743**	**0.654**	**0.546**	**0.397**

We show the precision numbers for some particular recalls as well as the AUC in Table 2, where PCNN+ATT (1) refers to train sentences with two entities and one relation label, PCNN+ATT (m) refers to train sentences with four entities[7] and two relation labels. We observe that our model exhibits the best performances. Moreover, in the process of predicting the existence of relations for a sentence, our approach is more convenient , as the PCNN-ATT (1) has to predict all possible pairs of entities in the sentence while our approach can predict multiple relations simultaneously.

Table 2: Precisions on the Wikidata dataset.

Recall	0.1	0.2	0.3	AUC
Rank+ExATT	0.584	0.535	0.487	0.392
PCNN+ATT (m)	0.365	0.317	0.213	0.204
PCNN+ATT (1)	0.665	0.517	0.413	0.396
Our Model	0.650	0.519	0.422	**0.405**

Ablation study: To better demonstrate the performance of capsule net and attention mechanism, we remove the primary capsule layer and dynamic routing to make Bi-LSTM layer followed by a fully connected layer instead. We also remove the word-level attention separately. The experimental results on Wikidata dataset are summarized in Table 3. The results of "-Word-ATT" row refers to the results without word-level attention. According to the table, the drop of precision demonstrates that the word-level attention is quite useful. Generally, all two proposed strategies contribute to the effectiveness of our model.

---

[5] $d_w = 50$

[6] $d_w = 200$

[7] Two additional position embeddings.

Table 3: Ablation study of capsule net and word-level attention on Wikidata dataset.

Recall	0.1	0.2	0.3	AUC
-Word-ATT	0.648	0.515	0.395	0.389
-Capsule	0.635	0.507	0.413	0.386
Our Model	0.650	0.519	0.422	0.405

4.2 Discussion

CNN vs Capsule: Capsule networks achieve comparable results compared with baselines. In fact, the capsule combines features by clustering. A nonlinear map is constructed in an iterative manner, ensuring the output of each capsule to be sent to an appropriate parent in the subsequent layer. Dynamic routing may be more effective than the strategies such as max-pooling in CNN, which essentially detects whether a feature is present in any position of the text or not, but loses spatial information of the feature. Additionally, capsule achieves comparable results to predict multi-relations in the case of single entity pair, and performs better in the case of multiple entity pairs relation extraction.

Choice of d: In the experiments, the larger the dimension of the capsule, the more the capabilities of the feature information it contains. However, larger dimension increases the computational complexity. We test different levels of dimensions of capsules. The model is trained on two Nvidia GTX1080ti GPUs with 64G RAM and six Intel(R) Core(TM) i7-6850K CPU 3.60GHz. As the table 4 depicts, the training time increases with the growth of d. When $d = 32$, we observe that the loss decreases very slowly and the model is difficult to converge. So we only train 2 epochs and stop training. We set the parameter $d = 8$ empirically to balance the precision and training time cost.

Table 4: Precisions on the Wikidata dataset with different choice of d.

Recall	0.1	0.2	0.3	AUC	Time
$d = 1$	0.602	0.487	0.403	0.367	4h
$d = 32$	0.645	0.501	0.393	0.370	-
$d = 16$	0.655	0.518	0.413	0.413	20h
$d = 8$	0.650	0.519	0.422	0.405	8h

Effects of Iterative Routing:

We also study how the iteration number affect the performance on the Wikidata dataset. Table 5 shows the comparison of 1 - 5 iterations. We find that the performance reach the best when iteration is set to 3. The results indicate the dynamic routing is contributing to improve the performance. Specifically, in the iteration algorithm, the $b_{j|i} = b_{j|i} + \hat{u}_{j|i} \cdot v_j$. When the number of iteration is very large, v_j becomes either 0 or 1, which means each underlying capsule is only linked to a single upper capsule. Therefore, the iteration times should not be too large.

Table 5: Precisions on the Wikidata dataset with different number of dynamic routing iterations.

Recall	0.1	0.2	0.3	AUC
Iteration=1	0.531	0.455	0.353	0.201
Iteration=2	0.592	0.498	0.385	0.375
Iteration=3	0.650	0.519	0.422	0.405
Iteration=4	0.601	0.505	0.422	0.385
Iteration=5	0.575	0.495	0.394	0.376

4.3 Conclusion

We propose a relation extraction approach based on capsule networks with attention mechanism. Although we use Bi-LSTM as sentence encoding in this paper, the other encoding method, such as convolved n-gram, could be alternatively used. Experimental results of two benchmarks show that the model improves the precision of the predicted relations.

In the future, we tend to resolve the situation of how to assign predicted relationship to multi entity pairs when two entities have multi-relations by utilizing prior knowledge such as entity type and joint training with named entity recognition. We will also try to optimize the model in terms of speed and focus on other problems by leveraging class ties between labels, specially on multi-label learning problems. Besides, dynamic routing could also be useful to improve other natural language processing tasks such as the sequence-to-sequence task and so on.

Acknowledgments

We want to express gratitude to the anonymous reviewers for their hard work and kind comments, which will further improve our work in the future. This work is funded by NSFC 61673338/61473260, and partly supported by Alibaba-Zhejiang University Joint Institute of Frontier Technologies.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Xiaocheng Feng, Jiang Guo, Bing Qin, Ting Liu, and Yongjie Liu. 2017. Effective deep memory networks for distant supervised relation extraction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, pages 19–25.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. *arXiv preprint arXiv:1806.01501*.

Zhengqiu He, Wenliang Chen, Zhenghua Li, Meishan Zhang, Wei Zhang, and Min Zhang. 2018. See: Syntax-aware entity embedding for neural relation extraction. *arXiv preprint arXiv:1801.03603*.

Geoffrey Hinton, Nicholas Frosst, and Sara Sabour. 2018. Matrix capsules with em routing.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2018. Improving distantly supervised relation extraction using word and entity based attention. *arXiv preprint arXiv:1804.06987*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *arXiv preprint arXiv:1708.03743*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580*.

Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1784–1789.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Guanying Wang, Ruoxu Wang, and Huajun Chen. 2018a. Label-free distant supervision for relation extraction via knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. 2018b. Sentiment analysis by capsules. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1165–1174. International World Wide Web Conferences Steering Committee.

Dongdong Yang, Senzhang Wang, and Zhoujun Li. 2018. Ensemble neural relation extraction with adaptive boosting. *arXiv preprint arXiv:1801.09334*.

Hai Ye, Wenhan Chao, Zhunchen Luo, and Zhoujun Li. 2017. Jointly extracting relations with class ties via effective deep ranking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1810–1820.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. *arXiv preprint arXiv:1804.00538*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

Put It Back: Entity Typing with Language Model Enhancement

Ji Xin[1,2], **Hao Zhu**[1], **Xu Han**[1], **Zhiyuan Liu**[1*], **Maosong Sun**[1]
[1]State Key Laboratory on Intelligent Technology and System
Beijing National Research Center for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, China
[2]David R. Cheriton School of Computer Science, University of Waterloo, Canada
`ji.xin@uwaterloo.ca`
`{zhuhao15,hanxu17}@mails.tsinghua.edu.cn`
`{liuzy,sms}@tsinghua.edu.cn`

Abstract

Entity typing aims to classify semantic types of an entity mention in a specific context. Most existing models obtain training data using distant supervision, and inevitably suffer from the problem of noisy labels. To address this issue, we propose entity typing with language model enhancement. It utilizes a language model to measure the compatibility between context sentences and labels, and thereby automatically focuses more on context-dependent labels. Experiments on benchmark datasets demonstrate that our method is capable of enhancing the entity typing model with information from the language model, and significantly outperforms the state-of-the-art baseline. Code and data for this paper can be found from `https://github.com/thunlp/LME`.

1 Introduction

Entity typing classifies semantic types of an entity mention in a context sentence, and can be beneficial for a large number of natural language processing tasks (Neelakantan and Chang, 2015), such as entity linking (Chabchoub et al., 2016), relation extraction (Miwa and Sasaki, 2014), and question answering (Yahya et al., 2013). Fine-grained entity typing (FET) (Ling and Weld, 2012; Yosef et al., 2012; Yao et al., 2013; Gillick et al., 2014; Del Corro et al., 2015; Yogatama et al., 2015; Yaghoobzadeh and Schütze, 2015; Ren et al., 2016a; Yuan and Downey, 2018) is based on a large set of fine-grained types and is therefore more challenging. So far, neural models (Dong et al., 2015; Shimaoka et al., 2017; Xin et al., 2018) have achieved state-of-the-art results on FET.

All current FET models rely on distant supervision (DS) (Mintz et al., 2009) to obtain training data, due to the lack of large-scale human-labeled data. Such reliance on DS has been a significant problem for entity typing. In DS, an entity mention in the context sentence is first linked to a named entity in the knowledge base (KB). The entity has type labels[1] stored in the KB, and all labels will be assigned to this entity mention. In other words, these are noisy global labels without considering the specific context of the entity mention. On the other hand, entity typing aims to predict context-dependent types of the entity mention, and test datasets are all human-labeled. The difference between DS and human annotation leads to a huge gap in performances between training/development and test dataset.[2]

Raw	**Schwarzenegger** was elected to be the governor. **Schwarzenegger** acted in the film *Terminator*.
Good	(A) `politician` was elected to be the governor. (An) `actor` acted in the film *Terminator*.
Bad	(An) `actor` was elected to be the governor. (A) `politician` acted in the film *Terminator*.

Table 1: Examples of entity mention—type name replacement.

To address this problem, we propose Entity Typing with **L**anguage **M**odel **E**nhancement (LME). It is able to measure the compatibility between the context sentence and each distantly supervised label, in an *unsupervised* manner using meaning of the label.

In previous works, the hierarchical structure of labels has been considered (Ren et al., 2016b; Karn et al., 2017; Xu and Barbosa, 2018). However, to the best of our knowledge, precious

---

* Corresponding author: Zhiyuan Liu.

[1] Since entities are classified into labels of types, *type* and *label* have the same meaning in this paper.

[2] In the WIKI dataset, strict accuracies and macro-F1 scores are respectively 72.3%/89.2% on the development set and 59.7%/79.0% on the test set, using the model NFGEC from (Shimaoka et al., 2017).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 993–998
Brussels, Belgium, October 31 - November 4, 2018.

information inside names of labels has never been used. For example, whether the label is `/person/actor` or `/foo/bar` makes no difference. We argue that, the meaning of entity mention words can also be expressed by the name of its **context-dependent type**, to some extent. Based on this argument, replacements with context-dependent types make more sense than those with global-but-context-irrelevant ones. We provide an example in Table 1. The entity *Schwarzenegger* has types `/actor` and `/politician`, and we can see that replacements with context-dependent types produce better sentences.

The natural way to evaluate the soundness of sentences is language modeling (Bengio et al., 2003; Mikolov et al., 2010). Our method employs a language model to evaluate the soundness of each synthetic sentence generated by replacing the entity mention with its type's name. It is able to focus more on context-dependent types.

We conduct experiments to compare our model with the state-of-the-art baseline on two widely used datasets. The results demonstrate that LME is capable of improving entity typing systems by considering the meaning of labels, and alleviating the problem of noise in distantly-supervised entity typing.

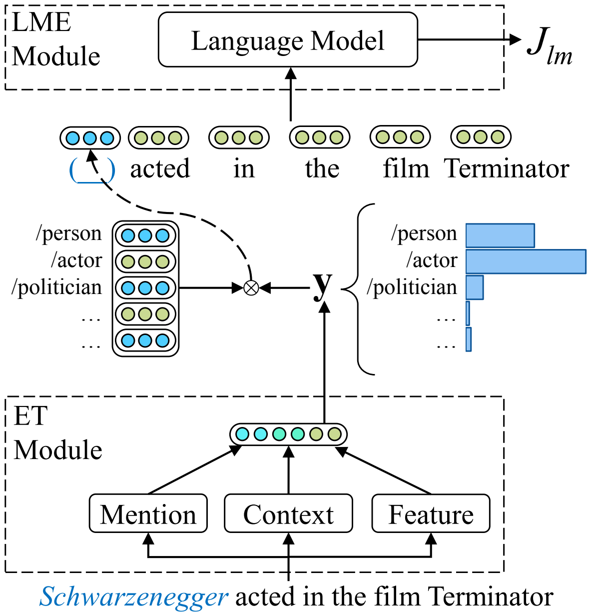

Figure 1: Our model: an entity typing (ET) module and a language model enhancement (LME) module.

2 Model

Our model (Figure 1) consists of two parts: an entity typing (ET) module, and a language model enhancement (LME) module.

ET predicts a probability distribution vector $\mathbf{y}$ for an entity mention, where each entry $\mathbf{y}_i$ represents the predicted probability for each type label.

In the training phase, LME optimizes a language model whose input includes $\mathbf{y}$, and also back-propagates gradients through $\mathbf{y}$ to parameters inside ET. In the testing phase, LME is **not** involved and $\mathbf{y}$ is directly used for inference: if $\mathbf{y}_i$ is greater than a threshold 0.5, the i$^{\text{th}}$ type is considered true; if all entries are below the threshold, the type with the greatest entry is considered true.

2.1 Entity Typing Module

Entity typing is defined on an ontology $\mathcal{T}$ (the set of all labels). Given an entity mention e and its context sentence $s = \{l_1, l_2, ..., e, r_1, r_2, ...\}$ (l_i and r_i are left and right context words), the typing model predicts a vector $\mathbf{y}$ indicating the probability distribution over all labels in the ontology:

$$\mathbf{y} = \sigma(\mathbf{W}_y \; [\mathbf{v}_M; \mathbf{v}_C; \mathbf{v}_F]), \qquad (1)$$

where σ is the sigmoid function, $\mathbf{W}_y$ is a parameter matrix, and $[;;]$ denotes concatenation. Three vectors: **M**ention, **C**ontext and **F**eature, are built from e and s as follows:

Entity mention vector There may be multiple words $e_1, e_2, ...$ in the entity mention, and $\mathbf{v}_M$ is the average of word embeddings of these words.

Context vector Two bi-directional LSTMs (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) are used for left and right context words. The outputs of BiLSTMs further go through a self-attention layer. $\mathbf{v}_C$ is the concatenation of the attention-layer outputs.

Hand-crafted feature vector A sparse feature vector $\mathbf{f}$ is built from the entity mention e. The features are adopted from those used by Gillick et al. (2014) and Yogatama et al. (2015). $\mathbf{v}_F$ is a dense projection of $\mathbf{f}$:

$$\mathbf{v}_F = \mathbf{W}_f \mathbf{f}, \qquad (2)$$

where $\mathbf{W}_f$ is the projection matrix.

After $\mathbf{y}$ is calculated, DS provides a label vector $\mathbf{y}^* \in \{0,1\}^{|\mathcal{T}|}$, where $|\mathcal{T}|$ is the number of labels. The loss function for typing is the cross-entropy between $\mathbf{y}$ and $\mathbf{y}^*$:

$$J_{type} = H(\mathbf{y}^*, \mathbf{y}) \\ = -\sum_i y_i^* \log(y_i) + (1 - y_i^*) \log(1 - y_i), \tag{3}$$

2.2 Language Model Enhancement Module

The core part of the LME module is an LSTM language model (Sundermeyer et al., 2012). The language model takes a sentence $\{w_1, w_2, ..., w_n\}$ as input, and assigns a probability to this sentence. Concretely, at step i, the LSTM reads the word sub-sequence $\{w_1, ..., w_i\}$, and predicts the probability of w_{i+1} succeeding the sub-sequence. A well trained language model predicts high probability for a reasonable sentence.

Before applying the LME module to enhance the ET module, the language model is pre-trained with sentences from the training set. The loss function for s in the pre-train phase is:

$$J_{pre} = \texttt{LM}(\{\mathbf{l}_1, \mathbf{l}_2, ..., \mathbf{e}, \mathbf{r}_1, \mathbf{r}_2, ...\}), \tag{4}$$

where bold face letters are word embeddings for corresponding words. $\texttt{LM}(\cdot)$ is the language model loss function: accumulative step-wise log-probability of each word of the input sequence. A well-trained language model calculates smaller loss for a more reasonable sentence.

After pre-training the language model, the LME module is combined with the ET module. Concretely, we assign an embedding vector $\mathbf{L}_i$ for each label, and take the sum of label embeddings weighted by $\mathbf{y}$. The sum $\mathbf{h}$ replaces $\mathbf{e}$ in the input sequence of the language model:

$$\mathbf{h} = \sum_{i=1}^{T} \mathbf{y}_i \mathbf{L}_i, \tag{5}$$

$$J_{lm} = \texttt{LM}(\{\mathbf{l}_1, \mathbf{l}_2, ..., \mathbf{h}, \mathbf{r}_1, \mathbf{r}_2, ...\}), \tag{6}$$

where $\mathbf{L}$ is the matrix of all label embeddings, and J_{lm} is loss function of the language model used in the training phase. In order to ensure that label embeddings are in the same semantic space with word embeddings, $\mathbf{L}$ is initialized with word embeddings of the labels' names.

In the training phase, parameters of the ET module are updated w.r.t.

$$J_{train} = J_{type} + \lambda J_{lm}, \tag{7}$$

where λ is the weight to balance the loss.

The ET module has a much smaller parameter space than the language model. In order to make full use of the gradients, we only update parameters of the ET module and **fix** the language model in the training phase. Now that the language model is fixed, when J_{lm} is being minimized, it adjusts the probability distribution in $\mathbf{y}$. If a label i is compatible with the context sentence, its corresponding entry $\mathbf{y}_i$ is expected to have a high value. Gradients are back-propagated through $\mathbf{y}$ and update parameters of the ET module. In this way, $\mathbf{y}$ can learn to be more context-dependent.

3 Experiments

3.1 Dataset

We employ two well-established and widely-used dataset for evaluating our model: WIKI (Ling and Weld, 2012) and ONTONOTES (Gillick et al., 2014).

Training parts of both datasets are labeled with DS, and testing parts are annotated by human. Therefore they are suitable for evaluating how our model can narrow the gap between DS and ground-truth context-dependent labels. Statistics of the two datasets are provided in Table 2.

Dataset	Train	Development	Test
WIKI	2,000,000	10,000	563
ONTONOTES	251,039	2,202	8,963

Table 2: Number of instances in each part of datasets.

3.2 Experiment Settings

The baseline for comparison is the hybrid model NFGEC proposed by Shimaoka et al. (2017). It is described as the ET module of our model. Our own model is referred to as NFGEC+LME.

We implement our model based on the source code of NFGEC.[3] For a fair comparison, the ET module is unchanged, including all hyperparameters and methods of parameter random initialization. Word embeddings are initialized with pre-trained embeddings provided by Pennington et al. (2014).

There are a few additional hyperparameters in our model. The most important one is λ, the weight between two parts of the loss function. Other ones include the learning rate r for pretraining the language model and the hidden size h of LSTM used in the language model. We perform

---

[3] `https://github.com/shimaokasonse/NFGEC`

a grid-search based on performances on the development set, and set $r = 0.005$ and $h = 500$. Details of λ will be discussed in Section 3.4.

3.3 Overall Results

We compare vanilla NFGEC and NFGEC+LME in Table 3. The results of NFGEC come from the paper by Shimaoka et al. (2017). For running NFGEC+LME, λ is set to 0.005 in WIKI and 0.001 in ONTONOTES.

Evaluation metrics include strict accuracy, macro-F1 score and micro-F1 score (Ling and Weld, 2012).

Dataset	WIKI			ONTONOTES		
Metric	**Strict**	Macro	Micro	**Strict**	Macro	Micro
NFGEC	59.68	78.97	75.36	50.89	70.80	64.93
+LME	62.88	80.61	76.95	52.90	72.41	65.17

Table 3: Performance of entity typing, evaluated by strict accuracy, macro-F1 and micro-F1 score. (%)

From the results we see that:

(1) In both datasets, LME consistently helps NFGEC to better classify entity mentions into their context-dependent types. We can see improvements in all metrics. This is because LME is capable of evaluating the appropriateness of each label and distinguishing context-dependent ones from global-but-context-irrelevant ones. Therefore LME helps the system to focus on more reasonable types.

(2) Among all metrics, the improvement on strict accuracy is the most significant. Strict accuracy is the proportion of entity mentions whose predicted types are completely identical with human annotation. It is therefore the most important metric for evaluating how robust the system is against noisy labels. The ability of LME alleviating noises from DS contributes to improving strict accuracy most.

3.4 Analysis of λ

We choose the optimal λ values for results in Table 3 according to their performances on the *development* set. After they are chosen, we compare the results on the *test* set under different values of λ in Figure 2.

Conclusions from the previous subsection can be seen again: when λ is set to a proper value, our model can consistently outperform the baseline over all metrics; strict accuracy is the metric with the most significant improvements.

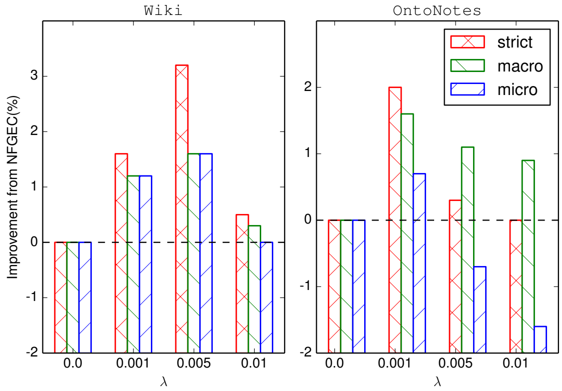

Figure 2: The performance under different λ values. $\lambda = 0$ is the vanilla NFGEC. Note that values of the vertical axis are improvements compared to vanilla NFGEC.

Also, we notice that the performances deteriorate when λ grows too large, and may even be worse than the baseline. The reason is that LME is a kind of *regularization*: its role is only in the training phase, exchanging the performance on training set with that on test set. So λ, as a regularization coefficient, must be carefully chosen.

3.5 Qualitative Analysis

In order to have an intuitive feeling of the model, we provide an example of LME's effect.

In the following sentence (from the test set of WIKI), both models try to predict the type of **Lake Placid** which, in this very context, is a town in New York. We show all labels with at least one score over the threshold 0.5, or is annotated true by human in Table 4.

> *Scaringe dismissed Brian Barrett of **Lake Placid** as one of his defense attorneys.*

Type	NFGEC	+LME	Human
`/person`	0.622	0.150	False
`/location`	0.323	0.627	True
`/location/city`	0.024	0.208	True

Table 4: An example, showing the scores by two models as well as human annotation.

NFGEC predicts a high score for `/person` and a low score for `/location`, probably because both words of the entity mention are first-letter capitalized and thus look like a person's name. LME, however, may consider the sentence structure *person of location* to be more reasonable than *person of person*, and makes the correct judgment between these two labels. As for `/location/city`, LME also shows higher confidence than NFGEC, but it is still regretfully below the threshold. This also demonstrates a weakness of LME: limited by the performance of the ET module. Addressing this limitation can be considered as a future direction for improvement.

4 Conclusion

In this paper, we propose a novel architecture LME to improve entity typing systems. It utilizes a language model and a set of label embeddings to judge the compatibility between labels and context sentences, and reduces noises introduced by DS. Experiments demonstrate that LME is capable of helping NFGEC, a state-of-the-art entity typing model, to alleviate the problem of noisy labels, and reaching a new state-of-the-art performance. Since the LME module does not depend on the ET module, we are confident that LME can be adapted to other entity typing systems as well.

Future Work Utilizing meaning of labels to alleviate the problem of noises from DS is an interesting direction. We make the first attempt in this paper, and we believe the direction is worth further exploring. For example, (1) how to train a language model that is sensitive with incorrect labels; (2) how to combine meaning of labels with the hierarchical structure of types; (3) how to find the optimal λ easily for a new dataset. LME may also be extended to other tasks that also suffer from noises and incompleteness of DS, such as relation extraction (Takamatsu et al., 2012; Ritter et al., 2013; Lin et al., 2016). However, since a relation does not have a specific location in the sentence, it needs more effort than a simple replacement.

Acknowledgment

This work is supported by the 973 Program (No. 2014CB340501), the National Natural Science Foundation of China (NSFC No. 61572273, 61661146007) and Tsinghua University Initiative Scientific Research Program (20151080406). This work is also funded by China Association for Science and Technology (2016QNRC001). We also thank anonymous reviewers for their insightful suggestions.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JMLR*.

Mohamed Chabchoub, Michel Gagnon, and Amal Zouaq. 2016. Collective disambiguation and semantic annotation for entity linking and typing. In *Proceedings of SWEC*.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of EMNLP*.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of IJCAI*.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Sanjeev Karn, Ulli Waltinger, and Hinrich Schütze. 2017. End-to-end trainable attentive decoder for hierarchical entity classification. In *Proceedings of EACL*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of AAAI*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of EMNLP*.

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. *Proceedings of NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of EMNLP*.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of KDD*.

Alan Ritter, Mausam, Luke Zettlemoyer, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *TACL*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of EACL*.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Proceedings of INTERSPEECH*.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of ACL*.

Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Improving neural fine-grained entity typing with knowledge attention. In *Proceedings of AAAI*.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. In *Proceedings of NAACL*.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of EMNLP*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *Proceedings of CIKM*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Proceedings of Workshop on AKBC*.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of ACL*.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical type classification for entity names. In *Proceedings of COLING*.

Zheng Yuan and Doug Downey. 2018. Otyper: A neural architecture for open named entity typing. In *Proceedings of AAAI*.

Event Detection with Neural Networks: A Rigorous Empirical Evaluation

J. Walker Orr *
George Fox University
414 N Meridian St.
Newberg, OR 97132
`jorr@georgefox.edu`

Prasad Tadepalli and **Xiaoli Fern**
Oregon State University
Corvallis OR 97331
`tadepall@eecs.oregonstate.edu`,
`xfern@eecs.oregonstate.edu`

Abstract

Detecting events and classifying them into predefined types is an important step in knowledge extraction from natural language texts. While the neural network models have generally led the state-of-the-art, the differences in performance between different architectures have not been rigorously studied. In this paper we present a novel GRU-based model that combines syntactic information along with temporal structure through an attention mechanism. We show that it is competitive with other neural network architectures through empirical evaluations under different random initializations and training-validation-test splits of ACE2005 dataset.

1 Introduction

Events are the lingua franca of news stories and narratives and describe important changes of state in the world. Identifying events and classifying them into different types is a challenging aspect of understanding text. This paper focuses on the task of *event detection*, which includes identifying the "trigger" words that indicate events and classifying the events into refined types. Event detection is the necessary first step in inferring more semantic information about the events including extracting the arguments of events and recognizing temporal and causal relationships between different events.

Neural network models have been the most successful methods for event detection. However, most current models ignore the syntactic relationships in the text. One of the main contributions of our work is a new DAG-GRU architecture (Chung et al., 2014) that captures the context and syntactic information through a bidirectional reading of the text with dependency parse relationships. This generalizes the GRU model to operate on a graph by novel use of an attention mechanism.

Following the long history of prior work on event detection, ACE2005 is used for the precise definition of the task and the data for the purposes of evaluation. One of the challenges of the task is the size and sparsity of this dataset. It consists of 599 documents, which are broken into a training, development, testing split of 529, 30, and 40 respectively. This split has become a de-facto evaluation standard since (Li et al., 2013). Furthermore, the test set is small and consists only of newswire documents, when there are multiple domains within ACE2005. These two factors lead to a significant difference between the training and testing event type distribution. Though some work had been done comparing method across domains (Nguyen and Grishman, 2015), variations in the training/test split including all the domains has not been studied. We evaluate the sensitivity of model accuracy to changes in training and test set splits through a randomized study.

Given the limited amount of training data in comparison to other datasets used by neural network models, and the narrow margin between many high performance methods, the effect of the initialization of these methods needs to be considered. In this paper, we conduct an empirical study of the sensitivity of the system performance to the model initialization.

Results show that our DAG-GRU method is competitive with other state-of-the-art methods. However, the performance of all methods is more sensitive to the random model initialization than expected. Importantly, the ranking of different methods based on the performance on the standard training-validation-test split is sometimes different from the ranking based on the average over multiple splits, suggesting that the community should move away from single split evaluations.

---

[1] Also associated with Oregon State University.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 999–1004
Brussels, Belgium, October 31 - November 4, 2018.

2 Related Work

Event detection and extraction are well-studied tasks with a long history of research.

Nguyen and Grishman (2015) used CNNs to represent windows around candidate triggers. Each word is represented by a concatenation of its word and entity type embeddings with the distance to candidate trigger. Global max-pooling summarizes the CNN filter and the result is passed to a linear classifier.

Nguyen and Grishman (2016) followed up with a skip-gram based CNN model which allows the filter to skip non-salient or otherwise unnecessary words in the middle of word sequences.

Feng et al. (2016) combined a CNN, similar to (Nguyen and Grishman, 2015), with a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to create a hybrid network. The output of both networks was concatenated together and fed to a linear model for final predictions.

Nguyen et al. (2016) uses a bidirectional gated recurrent units (GRUs) for sentence level encoding, and in conjunction with a memory network, to jointly predict events and their arguments.

Liu et al. (2016) created a probablistic soft logic model incorporating the semantic frames from Framenet (Baker et al., 1998) in the form of extra training examples. Based on the intuition that entity and argument information is important for event detection, Liu et al. (2017) built an attention model over annotated arguments and the context surrounding event trigger candidates.

Liu et al. (2018) created a cross language attention model for event detection and used it for event detection in both the English and Chinese portions of the ACE2005 data.

Recently, Nguyen and Grishman (2018) used graph-CNN (GCCN) where the convolutional filters are applied to syntactically dependent words in addition to consecutive words. The addition of the entity information into the network structure produced the state-of-the-art CNN model.

Another neural network model that includes syntactic dependency relationships is DAG-based LSTM (Qian et al., 2018). It combines the syntactic hidden vectors by weighted average and adds them through a dependency gate to the output gate of the LSTM model. To the best of our knowledge, none of the neural models combine syntactic information with attention, which motivates our research.

3 DAG GRU Model

Event detection is often framed as a multi-class classification problem (Chen et al., 2015; Ghaeini et al., 2016). The task is to predict the event label for each word in the test documents, and *NIL* if the word is not an event. A sentence is a sequence of words $x_1 \ldots x_n$, where each word is represented by a k-length vector. The standard GRU model works as follows:

$$
\begin{aligned}
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\
\tilde{h}_t &= \tanh(W_h x_t + r_t \odot U_h h_{t-1} + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{aligned}
$$

The GRU model produces a hidden vector h_t for each word x_t by combining its representation with the previous hidden vector. Thus h_t summarizes both the word and its prior context.

Our proposed DAG-GRU model incorporates syntactic information through dependency parse relationships and is similar in spirit to (Nguyen and Grishman, 2018) and (Qian et al., 2018). However, unlike those methods, DAG-GRU uses attention to combine syntactic and temporal information. Rather than using an additional gate as in (Qian et al., 2018), DAG-GRU creates a single combined representation over previous hidden vectors and then applies the standard GRU model. Each relationship is represented as an edge, (t, t', e), between words at index t and t' with an edge type e. The standard GRU edges are included as $(t, t-1, temporal)$.

Each dependency relationship may be between any two words, which could produce a graph with cycles. However, back-propagation through time (Mozer, 1995) requires a directed acyclic graph (DAG), Hence the sentence graph, consisting of temporal and dependency edges E, is split into two DAGs: a “forward” DAG G_f that consists of only of edges (t, t', e) where $t' < t$ and a corresponding “backward” DAG G_b where $t' > t$. The dependency relation between t and t' also includes the parent-child orientation, e.g., $nsubj\text{-}parent$ or $nsubj\text{-}child$ for a $nsubj$ (subject) relation.

An attention mechanism is used to combine the multiple hidden vectors. The matrix D_t is formed at each word x_t by collecting and transforming all the previous hidden vectors coming into node t, one per each edge type e. α gives the attention, a distribution weighting importance over the edges.

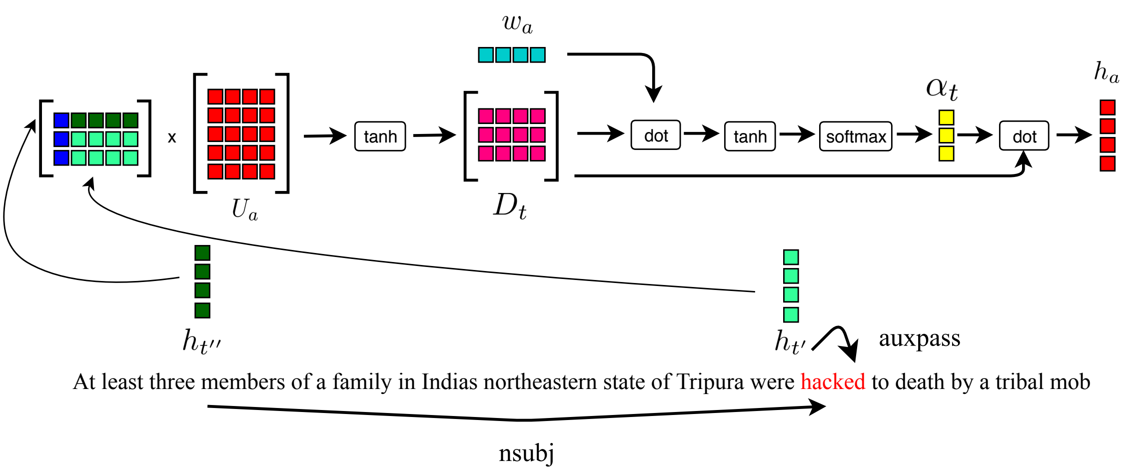

Figure 1: The hidden state of "hacked" is a combination of previous output vectors. In this case, three vectors are aggregated with DAG-GRU's attention model. $h_{t''}$, is included in the input for the attention model since it is accessible through the "subj" dependency edge. $h_{t'}$ is included twice because it is connected through a narrative edge and a dependency edge with type "auxpass." The input matrix is non-linearly transformed by U_a and tanh. Next, w_a determines the importance of each vector in D_t. Finally, the attention a_t is produced by tanh followed by softmax then applied to D_t. The subject "members" would be distant under a standard RNN model, however the DAG-GRU model can focus on this important connection via dependency edges and attention.

Finally, the combined hidden vector h_a is created by summing D_t weighted by attention.

$$D_t^T = [\tanh(U_e h_{t'}) | (t, t', e) \in E]$$
$$\alpha = \text{softmax}(\tanh(D_t w_a))$$
$$h_a = D_t^T \alpha$$

However, having a set of parameters U_e for each edge type e is over-specific for small datasets. Instead a shared set of parameters U_a is used in conjunction with an edge embedding v_e.

$$D_t^T = [\tanh(U_a h_{t'} \circ v_e) | (t, t', e) \in E]$$

The edge type embedding v_e is concatenated with the hidden vector $h_{t'}$ and then transformed by the shared weights U_a. This limits the number of parameters while flexibly weighting the different edge types. The new combined hidden vector h_a is used instead of h_{t-1} in the GRU equations.

The model is run forward and backward with the output concatenated, $h_{c,t} = h_{f,t} \odot h_{b,t}$, for a representation that includes the entire sentence's context and dependency relations. After applying dropout (Srivastava et al., 2014) with 0.5 rate to $h_{c,t}$, a linear model with softmax is used to make predictions for each word at index t.

4 Experiments

We use the ACE2005 dataset for evaluation. Each word in each document is marked with one of the thirty-three event types or *Nil* for non-triggers. Several high-performance models were reproduced for comparison. Each is a good faith reproduction of the original with some adjustments to level the playing field.

For word embeddings, Elmo was used to generate a fixed representation for every word in ACE2005 (Peters et al., 2018). The three vectors produced per word were concatenated together for a single representation. We did not use entity type embeddings for any method. The models were trained to minimize the cross entropy loss with Adam (Kingma and Ba, 2014) with L2 regularization set at 0.0001. The learning rate was halved every five epochs starting from 0.0005 for a maximum of 30 epochs or until convergence as determined by F1 score on the development set.

The same training method and word embeddings were used across all the methods. Based on preliminary experiments, these settings resulted in better performance than those originally specified. However, notably both GRU (Nguyen et al., 2016) and DAG-LSTM (Qian et al., 2018) were not used as joint models. Further, the GRU implementation did not use a memory network, instead we used the final vectors from the forward and backward pass concatenated to each timestep's output for additional context. For CNNs (Nguyen and Grishman, 2015) the number of filters was reduced to 50 per filter size. The CNN+LSTM (Feng et al.,

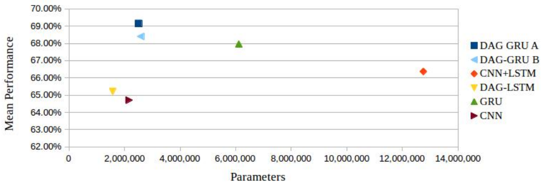

Figure 2: A comparison of mean performance versus number of parameters.

2016) model had no other modifications. DAG-GRU used a hidden layer size of 128. Variant A of the DAG-GRU model utilized the attention mechanism, while variant B used averaging, that is:

$$D_t = \frac{1}{|E(t)|} \sum_{(t',e) \in E(t)} \tanh(U_a h_{t'} \circ v_e)$$

4.1 Effects of Random Initialization

Given that ACE2005 is small as far as neural network models are concerned, the effect of the random initialization of these models needs to be studied. Although some methods include tests of significance, the type of statistical test is often not reported. Simple statistical significance tests, such as the t-test, are not compatible with a single F1 score, instead the average of F1 scores should be tested (Wang et al., 2015).

We reproduced and evaluated five different systems with different initializations to empirically assess the effect of initialization. The experiments were done on the standard ACE2005 split and the aggregated results over 20 random seeds were given in Table 1. The random initializations of the models had a significant impact on their performance. The variation was large enough that the observed range of the F1 scores overlapped across almost all the models. However the differences in average performances of different methods, except for CNN and DAG-LSTM, were significant at $p < 0.05$ according to the t-test, not controlling for multiple hypotheses.

Both the GRU (Nguyen et al., 2016) and CNN (Nguyen and Grishman, 2015) models perform well with their best scores being close to the reported values. The CNN+LSTM model's results were significantly lower than the published values, though this method has the highest variation. It is possible that there is some unknown factor such as the preprocessing of the data that significantly impacted the results or that the value is an outlier. Likewise, the DAG-LSTM model underperformed. However, the published results were based on a joint event and argument extraction model and probably benefited from the additional entity and argument information.

DAG-GRU A consistently and significantly outperforms the other methods in this comparison. The best observed F1 score, 71.1%, for DAG-GRU is close to the published state-of-the-art scores of DAG-LSTM and GCNN at 71.9% and 71.4% respectively. With additional entity information, GCNN achieves a score of 73.1%. Also, the attention mechanism used in DAG-GRU A shows a significant improvement over the averaging method of DAG-GRU B. This indicates that some syntactic links are more useful than others and that the weighting attention applies is necessary to utilize that syntactic information.

Another source of variation was the distributional differences between the development and testing sets. Further, the testing set only include newswire articles whereas the training and dev. sets contain informal writing such as web log (WL) documents. The two sets have different proportions of event types and each model saw at least a 2% drop in performance between dev. and test on average. At worst, the DAG-LSTM model's drop was 5.26%. This is a problem for model selection, since the dev. score is used to choose the best model, hyperparameters, or random initialization. The distributional differences mean that methods which outperform others on the dev. set do not necessarily perform as well on the test set. For example, DAG-GRU A performs worse that DAG-GRU B on the dev. set, however it achieves a higher mean score on the testing set.

One method of model selection over random initializations is to train the model k times and pick the best one based on the dev. score. Repeating this model selection procedure many times for each model is prohibitively expensive, so the experiment was approximated by bootstrapping the twenty samples per model (Efron, 1992). For each model, 5 dev. & test score pairs were sampled with replacement from the twenty available pairs. The initialization with the best dev. score was selected and the corresponding test score was taken. This model selection process of picking the best of 5 random samples was repeated 1000 times and the results are shown in Table 2. This process did not substantially increase the average performance beyond the results in Table 1, although it did reduce the variance, except for the CNN model. It ap-

Model	Dev Mean	Mean	Min	Max	Std. Dev.	Published
DAG-GRU A	70.3%	69.2% ± 0.42	67.8%	71.1%	0.91%	-
DAG-GRU B	71.2%	68.4% ± 0.45	67.39%	70.53%	0.96%	-
GRU	70.3%	68.0% ± 0.42	66.4%	69.4%	0.86%	69.3%†
CNN+LSTM	69.6%	66.4% ± 0.62	63.6%	68.1%	1.32%	73.4%
DAG-LSTM	70.5%	65.2% ± 0.44	63.0%	66.8%	0.94%	71.9%†
CNN	68.5%	64.7% ± 0.65	61.6%	67.2%	1.38%	67.6%

Table 1: The statistics of the 20 random initializations experiment. † denotes results are from a joint event and argument extraction model.

Model	Dev Mean	Mean	Std. Dev.
DAG-GRU A	72.0%	69.2% ± 0.04%	0.68%
DAG-GRU B	72.0%	67.9% ± 0.04%	0.60%
GRU	71.5%	68.4% ± 0.05%	0.80%
CNN+LSTM	70.8%	66.8% ± 0.07%	1.08%
DAG-LSTM	70.4%	65.5% ± 0.02%	0.40%
CNN	69.6%	65.4% ± 0.09%	1.49%

Table 2: Bootstrap estimates on 1000 samples for each model after model selection based on dev set scores.

pears that using the dev. score for model selection is only marginally helpful.

4.2 Randomized Splits

In order to explore the effect of the training/testing split popularized by (Li et al., 2013), a randomized cross validation experiment was conducted. From the set of 599 documents in ACE2005, 10 random splits were created maintaining the same 529, 30, 40 document counts per split, training, development, testing, respectively. This method was used to evaluate the effect of the standard split, since it maintains the same data proportions while only varying the split. The results of the experiment are found in Table 3.

The effect of the split is substantial. Almost all models' performance dropped except for DAG-LSTM, however the variance increased across all models. In the worst case, the standard deviation increased threefold from 0.86% to 2.60% for the GRU model. In fact, the increased variation of the splits means that the confidence intervals for all the models overlap. This aligns with cross domain analysis, some domains such as WL are known to be much more difficult than the newswire domain which comprises all of the test data under the standard split (Nguyen and Grishman, 2015). Further, the effect of the difference in splits also negates the benefits of the attention mechanism of DAG-GRU A. This is likely due to the test partitions' inclusion of WL and other kinds of informal writing. The syntactic links are much more likely to be noisy for informal writing, reducing the syntactic information's usefulness and reliability.

Method	Dev Mean	Mean	Min	Max	Std. Dev.
DAG-GRU A	71.4%	68.4% ± 1.85%	65.7%	74.1%	2.59%
DAG-GRU B	70.9%	68.4% ± 1.88%	64.19%	73.59	2.63%
DAG-LSTM	68.9%	67.3% ± 1.43%	63.5%	70.7%	2.00%
GRU	69.8%	66.6% ± 1.86%	62.5%	71.1%	2.60%
CNN+LSTM	69.8%	66.3% ± 2.03%	60.1%	70.3%	2.83%
CNN	68.0%	65.4% ± 1.59%	60.7%	69.2%	2.22%

Table 3: Average results on 10 randomized splits.

All these sources of variation are greater than most advances in event detection, so quantifying and reporting this variation is essential when assessing model performance. Further, understanding this variation is important for reproducibility and is necessary for making any valid claims about a model's relative effectiveness.

5 Conclusions

We introduced and evaluated a DAG-GRU model along with four previous models in two different settings, the standard ACE2005 split with multiple random initializations and the same dataset with multiple random splits. These experiments demonstrate that our model, which utilizes syntactic information through an attention mechanism, is competitive with the state-of-the-art. Further, they show that there are several significant sources of variation which had not been previously studied and quantified. Studying and mitigating this variation could be of significant value by itself. At a minimum, it suggests that the community should move away from evaluations based on single random initializations and single training-test splits.

Acknowledgments

This work was supported by grants from DARPA XAI (N66001-17-2-4030), DEFT (FA8750-13-2-0033), and NSF (IIS-1619433 and IIS-1406049).

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL (1)*, pages 167–176.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Bradley Efron. 1992. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 66–71.

Reza Ghaeini, Xiaoli Z Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 369.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL (1)*, pages 73–82.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018. Event detection via gated multilingual attention mechanism.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2134–2143.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1789–1798.

Michael C Mozer. 1995. A focused backpropagation algorithm for temporal. *Backpropagation: Theory, architectures, and applications*, page 137.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL-HLT*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *ACL (2)*, pages 365–371.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Feng Qian, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Yu Wang, Jihong Li, Ruibo Wang, and Wingli Yang. 2015. Confidence interval for f1 measure of algorithm performance based on blocked 32 cross-validation. volume 27, pages 651–659.

PubSE: A Hierarchical Model for Publication Extraction from Academic Homepages

Yiqing Zhang **Jianzhong Qi** **Rui Zhang**[*] **Chuandong Yin**
School of Computing and Information Systems
The University of Melbourne
`{yiqingz2,chuandongy}@student.unimelb.edu.au`
`{jianzhong.qi,rui.zhang}@unimelb.edu.au`

Abstract

Publication information in a researcher's academic homepage provides insights about the researcher's expertise, research interests, and collaboration networks. We aim to extract all the publication strings from a given academic homepage. This is a challenging task because the publication strings in different academic homepages may be located at different positions with different structures. To capture the positional and structural diversity, we propose an end-to-end hierarchical model named PubSE based on Bi-LSTM-CRF. We further propose an alternating training method for training the model. Experiments on real data show that PubSE outperforms the state-of-the-art models by up to 11.8% in F1-score.

1 Introduction

Researchers often list their publications in their academic homepages. These publications provide insights about the researchers' expertise, research interests, and collaboration networks. Extracting publications from a researcher's homepage is an essential step in extracting the researcher's profile (Tang et al., 2010). In this study, we aim to extract every publication from a researcher's homepage. For ease of discussion, we call such a publication item a *publication string*. Figure 1 illustrates the studied problem. There are two publications on the homepage shown in the figure, a journal article and a conference paper. We aim to extract them as two separate publication strings.

Extracting publication strings from academic homepages helps bypass the problem of name ambiguity (*i.e.*, different authors with the same name) (Zhu et al., 2018) in extracting publication strings from indexing sites such as DBLP and PubMed. However, extracting publication strings from academic homepages directly has its own challenges: (i) The list of publications may be located anywhere in a homepage with varying contexts. The structure of the list and the formatting styles of a publication string can vary vastly across different homepages. For example, some researchers like to list some of their publications with more details, such as full venue names, volume and page information, while listing the other publication in a concise way. Also, some researchers like to group their publication by year or by topic. (ii) A publication string may contain multiple lines of text (cf. Figure 1), and there may not be a clear boundary between two publications strings. (iii) There may be strings in an academic homepage that share very similar structures and styles with publication strings, such as records of conference presentations (cf. Figure 1). Previous work (Hong et al., 2009; Chung et al., 2012) focuses on feature and rule engineering and cannot accommodate the above challenges.

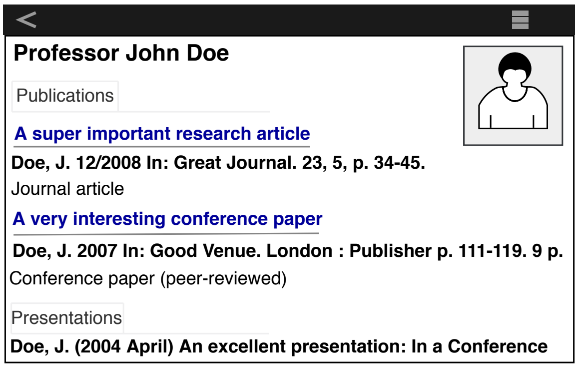

Professor John Doe

Publications

A super important research article

Doe, J. 12/2008 In: Great Journal. 23, 5, p. 34-45.
Journal article

A very interesting conference paper

Doe, J. 2007 In: Good Venue. London : Publisher p. 111-119. 9 p.
Conference paper (peer-reviewed)

Presentations

Doe, J. (2004 April) An excellent presentation: In a Conference

Figure 1: Part of a homepage with publication strings

To address these challenges, we propose a model named *PubSE* to extract every publication string from an academic homepage. PubSE has two characteristics: (i) The model structure reflects the structure of a list of publications, by its loss-functions at both text line-level and webpage-level. (ii) The training process of the

---

[*] Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1005–1010
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

model utilizes both text line-level and webpage-level information via an alternating training procedure to reduce overfitting. Our PubSE model can extract publication strings in non-trivial cases, such as multi-line publication strings from a non-continuous publication list. We make the following contributions: (i) We create a dataset of 2,500 homepages[1], in which each publication string is labeled. (ii) We address the problem of publication string extraction by end-to-end learning, without feature engineering. (iii) We propose a model that can learn the structures of publication lists and the styles of publication strings. We also propose an alternating training method that can reduce overfitting and further improve the prediction accuracy.

2 Related Work

Earlier studies extract publication strings from research papers (Peng and McCallum, 2006; Councill et al., 2008; Tkaczyk et al., 2015). Such a problem is simpler for two reasons: (i) The reference list of a research paper usually appears at a fixed position (e.g., end of paper) and is continuous. (ii) The references are usually well-formatted and have few format variations since they may be generated by software such as LaTeX.

For extracting publication strings from academic homepages, previous studies use either rule-based (Hong et al., 2009; Yang and Ho, 2010) or a hybrid of machine learning and rule-based methods (Chung et al., 2012).

For example, Chung et al. (2012) develop a system named PRM that first segments an HTML DOM tree and its contents based on HTML tags. Then, PRM uses a linear chain CRF model to label the different parts of the tree. Based on the labels, it refines the publication string boundaries by heuristic rules to produce final predictions.

Relying on the HTML DOM tree structure makes it difficult to train a machine learning based model for publication extraction because: (i) Text in a publication string may be separated in many different DOM tree nodes. (ii) The DOM tree structure, which previous web data record extraction systems (Liu et al., 2003; Furche et al., 2014; Omari et al., 2016) rely on, may vary given the same webpage content.

As a result, we do not use HTML tags in our model. Instead, we work on the visible text contents. To the best of our knowledge, no existing studies for publication string extraction can effectively model the structure of publication lists.

3 Dataset

To the best of our knowledge, there is no public dataset of academic homepages that has labeled publication strings.

We downloaded 2,500 academic homepages from 100 universities around the world. We use Selenium, an open-source automated rendering software, to render the webpages. We collect visible texts from the webpages and then manually tag all the publication strings in them. During tagging, we mark the beginning and ending byte offsets of each publication string. Among the 2,500 academic homepages, 723 homepages (28.9%) contain publication lists, which consist of a total of 13,237 publication strings. Among the 723 homepages that contain publication strings, there are 117 homepages (16.2%) that contain multi-line publication strings. On average, there are 732.1 (std=1583.3) tokens, 89.9 (std=141.6) lines, and 18.3 (std=35.4) publication strings per homepage. We call this dataset *HomePub*.

Each publication string in HomePub dataset is annotated by two annotators. Disagreement is resolved by a third annotator. On publication string level, annotators agree on 83.76% publications, and the Cohen's kappa is 0.2084.

We have developed a program *PageTagger*[2] to assist the annotation. On average, it takes about 2.5 minutes to annotate one academic homepage when using the PageTagger tool.

Note that our annotation does not consider the following as publication strings: (i) Master or PhD theses; (ii) working papers; (iii) seminars, invited talks, or presentations; and (iv) patents. Our annotation also excludes the numbers (e.g., [1] or [i]) if the publication strings are in a numbered list.

4 Methods

We summarize the baseline models in Section 4.1 and present our PubSE model in Section 4.2.

4.1 Baselines

PRM: Chung et al. (2012) develop a publication string extraction system based on region boundary

---

[1] Dataset available at `http://www.ruizhang.info/data/homepub.html`

[2] `https://github.com/yiqingzhang/page-tagger`

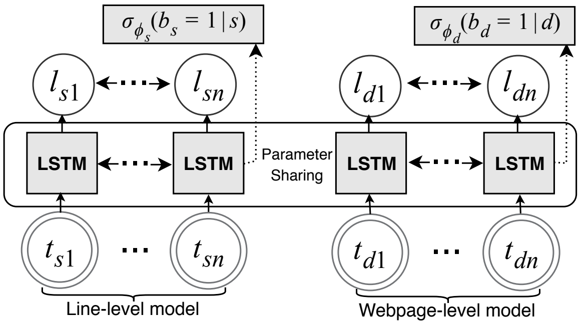

Figure 2: PubSE model

analysis and CRF. We train their system on 60% of the HomePub dataset.

ParsCit: ParsCit (Councill et al., 2008) is a CRF-based document structure analysis and reference parsing tool. We use ParsCit as an off-the-shelf tool on the HomePub dataset.

CNN-Sentence: In the HomePub dataset, over 80% of the publication strings are in a single line. The problem of extracting publication strings can be viewed as a single-line text classification problem (Kim, 2014; Yang et al., 2016; Joulin et al., 2017). Following Kim (2014), we implement a line-level classification model. We use the GloVe (300 dimensions) pre-trained embedding on this model (the same embedding is used across all the models that require word embeddings as the input). This model predicts whether each line in the webpage is a publication string.

Bi-LSTM-CRF: The problem of extracting publication strings can also be viewed as a sequence labeling task (Lample et al., 2016; Gui et al., 2017; Liu et al., 2017), where there are two possible labels for each token, publication (`I`) or non-publication (`O`). A consecutive sequence of `I` tokens forms a publications string. Sequence labeling approaches can capture correlations of labels and words, as well as words themselves. Following Ma and Hovy (2016), we implement a Bi-LSTM-CRF model. We concatenate token-level with character-level embeddings as the model input, to better deal with out-of-vocabulary tokens.

4.2 PubSE Model

To capture the hierarchical structure of a homepage and address the issue of multi-line publication strings, we propose a hierarchical model named *PubSE* with an alternating training method, which alternates between training a line-level and a webpage-level model to reduce overfitting. Figure 2 illustrates the model structure.

We incorporate both line-level and webpage-level information in the model since the predictions depend on both local (line-level) and global (webpage-level) context. On one hand, local information such as word embeddings and morphology information is crucial for the predictions over individual lines. On the other hand, global information is also necessary, e.g., to help identify the starting and ending positions of a publication list and the boundaries between multi-line publications strings.

Line-level model: As shown in Figure 2, the left Bi-LSTM network $\pi_\theta(\ell|s)$ specializes in line-level inputs, where each mini batch of input composes of lines in a webpage. On top of this sub-module, we add another layer $\sigma_{\phi_s}(b_s|s)$ to model whether each line contains a publication string:

$$\mathcal{L}_{line} = \lambda_{\theta_s}\mathcal{L}(\pi_\theta(\ell|s), y_t) + \lambda_{\phi_s}\mathcal{L}(\sigma_{\phi_s}(b_s|s), y_l),$$

where ℓ denotes the predicted labels for tokens t_{si} in a line s, and b_s denotes the predicted label for a line, *i.e.*, whether the line contains a publication string or not; y_t and y_l denote the ground truth label for each token and line. Hyperparameters λ_{θ_s} and λ_{ϕ_s} are the coefficients for token-level and line-level, while θ and ϕ_s are the parameters of the two networks; $\mathcal{L}$ is the cross-entropy loss:

$$\mathcal{L}(\hat{y}, y) = -(1-y)log(1-\hat{y}) - ylog(\hat{y}),$$

where $\hat{y}$ denotes the predicted label, and y denotes the ground truth label.

The inputs to the line-level model are each line in the form of $\langle(e_1, c_1), (e_2, c_2), ..., (e_n, c_n)\rangle$, where e_k and c_k are the word embedding and character embedding of the token t_{sk}. Network $\pi_\theta(\ell|s)$ outputs a label for each token, while network $\sigma_{\phi_s}(b_s|s)$ gives binary output for each line, indicating whether it contains a publication string. Extraction result is based on the output of $\pi_\theta(\ell|s)$.

Webpage-level model: Similarly, the right Bi-LSTM sub-module $\pi_\theta(\ell|d)$ in Figure 2 specializes in webpage-level inputs, where the whole homepage d is supplied to the model as a long sequence of token embeddings. We add another layer $\sigma_{\phi_d}(b_d|d)$ to reflect whether the homepage contains publication strings:

$$\mathcal{L}_{page} = \lambda_{\theta_d}\mathcal{L}(\pi_\theta(\ell|d), y_t) + \lambda_{\phi_d}\mathcal{L}(\sigma_{\phi_d}(b_d|d), y_w),$$

where b_d denotes whether the document contains publication or not; λ_{θ_d} and λ_{ϕ_d} are token-level and webpage-level coefficients; y_w denotes the ground truth label for a webpage; θ and ϕ_d are network parameters. Note that the left and right sub-modules collaborate by sharing network weights θ.

Methods	Exact Match			85% Match			Multi-line Only (Exact)		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
PRM (Chung et al., 2012)	15.39	37.54	21.83	18.25	44.52	25.89	5.70	10.83	7.47
ParsCit (Councill et al., 2008)	70.34	18.22	28.94	72.46	18.76	29.81	12.94	1.22	2.22
CNN-Sentence (Kim, 2014)	73.39	76.69	75.00	76.57	80.01	78.25	16.03	18.01	16.96
Bi-LSTM-CRF (Ma and Hovy, 2016)	74.15	77.22	75.65	75.76	78.90	77.30	21.11	22.21	21.65
PubSE (Proposed model)	**84.12**	**91.12**	**87.48**	**87.28**	**94.53**	**90.76**	**70.70**	**76.80**	**73.62**

Table 1: Performance of different models on the HomePub dataset. The first and second groups show the performance on the whole test dataset. The third group shows the exact matching performance on the subset that contains multi-line publications.

Alternating training method: Inspired by the training procedure of curriculum learning (Bengio et al., 2009) and soft-landing (He et al., 2016), we adopt an alternating training procedure controlled by the following function:

$$f(k) = H(cos(k/T)),$$

where T controls the period of the function, and k is the number of epochs. The H denotes the Heaviside step function. In the k^{th} epoch, we will train only one of the submodules, given by

$$\mathcal{L} = f(k)\mathcal{L}_{line} + f(k+1)\mathcal{L}_{page}.$$

Our intuition is that the training of line-level and webpage-level networks can reinforce each other and reduce overfitting. If we only train the model on the line-level input, the model will lose all the long-term dependency information. For example, a string that describes a thesis resembles that of a conference paper. To filter such strings, we need to rely on indicators that may reside in a different line such as a heading “Dissertations supervised”. On the other hand, if we only train the model on webpage-level input, the model may be dominated by the longest line on the homepage, such as biography information. Our alternating training procedure balances the two factors and can better model the hierarchical structure of a publication list.

The PubSE model can capture and exploit information on the webpage from four different perspectives: (i) character-level information such as word morphology; (ii) token-level information such as word context; (iii) line-level information such as whether a line is a publication string; and (iv) webpage-level information such as whether a webpage contains publication strings.

5 Experiments

Settings and Evaluation: We divide the HomePub dataset by a 60-40 split and train our model on 60% of the total data. We use 20% of the training set as a validation set for early stopping and hyperparameter tuning. The optimal hyperparameters are obtained with a standard grid search procedure on the validation set. We use 40 as the batch size for the line-level model and 1 for the webpage-level model. We set $\lambda_{\theta_s}, \lambda_{\phi_s}, \lambda_{\theta_d}, \lambda_{\phi_d}, T$ as 1, 0.05, 1, 0.3, $1/\pi$, respectively.

We use precision, recall, and F1-score to measure the performance, and we report both exact and relaxed matching performance. In exact matching, a publication string is considered to be correctly extracted only if it exactly matches a publication string in the ground truth. In relax matching, we allow mismatching 15% of publication strings. (*i.e.*, a publication string is considered correct if it contains at least 85% of the tokens of a publication string in the ground truth.) We also list the model performance on webpages in the test set that contain multi-line publication strings.

Results: We report the experiment results in Table 1. The result shows that the proposed PubSE model consistently outperforms all the baselines with a statistically significant margin, and the advantage is up to 11.8%. In particular, the use of the webpage-level sub-module helps PubSE to handle multi-line publication strings, which yields a significant performance gain.

In comparison, PRM struggles in determining which part of a page contains publications. ParsCit requires well-formatted inputs. For example, if publication strings do not contain page numbers, ParsCit will be reluctant to separate the list of publication strings into individual records. CNN-Sentence and Bi-LSTM-CRF give poor results in pages that contain multi-line publication strings.

Methods	Exact Match	85% Match	Multi-line
`L + LP`	76.65	79.38	18.99
`W`	83.04	86.27	62.04
`W + WP`	84.18	87.22	63.77
`L + W`	85.98	89.1	63.36
PubSE (`L+LP+W+WP`)	87.48	90.76	73.62

Table 2: F1-score of different variations of the PubSE model. `L` means only the line-level sub-module $\pi_\theta(\ell|s)$; `LP` means the extra layer $\sigma_{\phi_s}(b_s|s)$; `W` means only the webpage-level sub-module $\pi_\theta(\ell|d)$; `WP` means the extra layer $\sigma_{\phi_d}(b_d|d)$.

Ablation Study: We also test different variations of our proposed model PubSE, and the results are shown in Table 2.

About 50% improvement over the best baseline

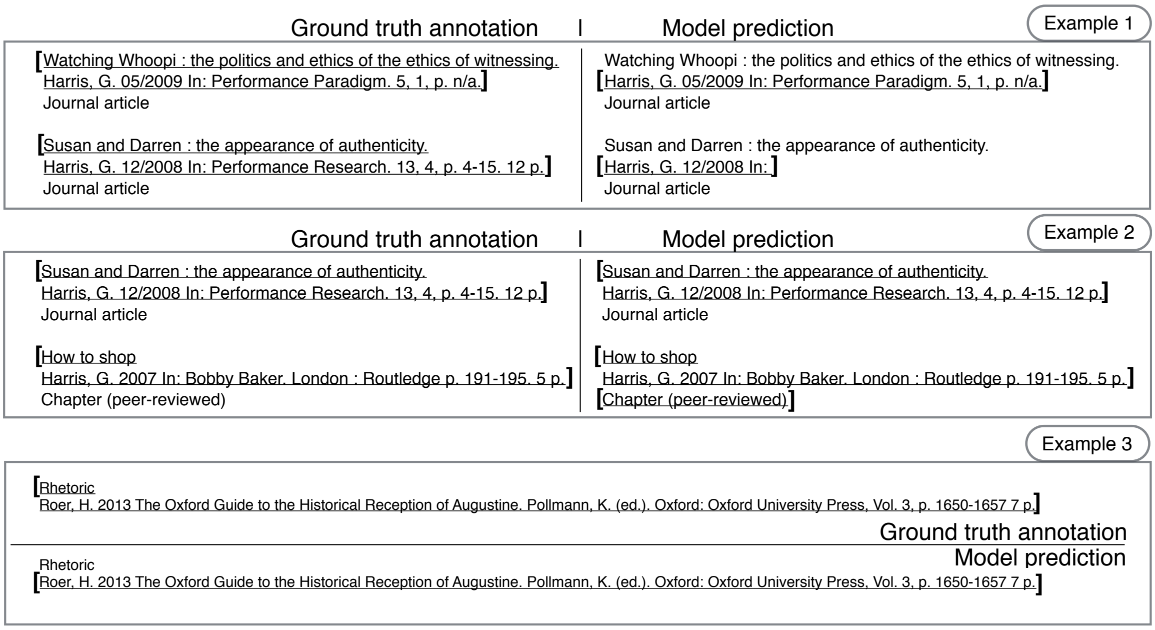
Ground truth annotation | Model prediction

Example 1

[Watching Whoopi : the politics and ethics of the ethics of witnessing.
Harris, G. 05/2009 In: Performance Paradigm. 5, 1, p. n/a.]
Journal article

[Susan and Darren : the appearance of authenticity.
Harris, G. 12/2008 In: Performance Research. 13, 4, p. 4-15. 12 p.]
Journal article

Watching Whoopi : the politics and ethics of the ethics of witnessing.
[Harris, G. 05/2009 In: Performance Paradigm. 5, 1, p. n/a.]
Journal article

Susan and Darren : the appearance of authenticity.
[Harris, G. 12/2008 In:]
Journal article

Ground truth annotation | Model prediction

Example 2

[Susan and Darren : the appearance of authenticity.
Harris, G. 12/2008 In: Performance Research. 13, 4, p. 4-15. 12 p.]
Journal article

[How to shop
Harris, G. 2007 In: Bobby Baker. London : Routledge p. 191-195. 5 p.]
Chapter (peer-reviewed)

[Susan and Darren : the appearance of authenticity.
Harris, G. 12/2008 In: Performance Research. 13, 4, p. 4-15. 12 p.]
Journal article

[How to shop
Harris, G. 2007 In: Bobby Baker. London : Routledge p. 191-195. 5 p.]
[Chapter (peer-reviewed)]

Example 3

[Rhetoric
Roer, H. 2013 The Oxford Guide to the Historical Reception of Augustine. Pollmann, K. (ed.). Oxford: Oxford University Press, Vol. 3, p. 1650-1657 7 p.]

Ground truth annotation

Model prediction

Rhetoric
[Roer, H. 2013 The Oxford Guide to the Historical Reception of Augustine. Pollmann, K. (ed.). Oxford: Oxford University Press, Vol. 3, p. 1650-1657 7 p.]

Figure 3: Typical errors of various models. Underlined tokens are labeled as `I` (publication). "[" and "]" are not part of the text. They are used to highlight the boundary of publication strings.

is made by training the model with webpage-level input (`W`) since it is difficult to extract multi-line publication strings without a global view of the whole webpage.

The effect of the alternating training method (`L`+`W`) is also significant. The webpage-level model (`W`) may not handle short lines too well, e.g., a line with text "Conference paper (peer-reviewed)" as shown in Figure 1. This problem is solved by combining the line-level model with the webpage-level model (`L`+`W`).

Error analysis: Figure 3 shows typical errors made by various models. Examples 1 shows errors occurred in line-level model prediction results. The line-level model does not handle multi-line publication strings well since the predictions of different lines are independent, so the model fails to capture dependency relationships in different lines.

Example 2 shows prediction results given by the webpage-level model. We see that the webpage-level model can make a more accurate prediction for multi-line publications. However, it may make false positive predictions for short lines (e.g., "Chapter (peer-reviewed)"), while the line-level model seldom makes such mistakes. This is the motivation for us to integrate both the line-level and the webpage-level models.

PubSE can avoid most of the errors shown in Examples 1 and 2. Nevertheless, PubSE still makes mistakes in some challenging cases. Example 3 shows such a case, where PubSE does not recognize that "Rhetoric" is a publication title. A possible explanation is that such a short publication title is less common.

6 Conclusions and Future work

We studied publication string extraction and proposed a model named PubSE for the problem. PubSE models the publication list structure with its hierarchical structure and loss functions. We proposed an alternating training scheme that combines both line-level and webpage-level information, which are crucial for predicting multi-line publication strings. Experiments show that the proposed PubSE model outperforms the state-of-the-art models by up to 11.8% in F1-score.

For future work, we aim to expand our experimental study to a larger scale. We further consider extracting publication strings from academic homepages of the same organization. Such homepages may share similar templates, which may help improve the extraction accuracy. We also plan to investigate adaptive alternating model training schemes as well as external memory mechanism such as memory networks.

Acknowledgments

We thank Afshin Rahimi and the anonymous reviewers for their valuable suggestions, and we gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research. We also sincerely thank Han Liu and other annotators for their help in creating the HomePub dataset.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the Annual International Conference on Machine Learning (ICML)*, pages 41–48.

Jen-Ming Chung, Ya-Huei Lin, Hahn-Ming Lee, and Jan-Ming Ho. 2012. Mining publication records on personal publication web pages based on conditional random fields. In *Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 319–326.

Isaac G. Councill, C. Lee Giles, and Min-Yen Kan. 2008. ParsCit: an open-source CRF reference string parsing package. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 661–667.

Tim Furche, Georg Gottlob, Giovanni Grasso, Xiaonan Guo, Giorgio Orsi, Christian Schallhart, and Cheng Wang. 2014. DIADEM: thousands of websites to a single database. *Proceedings of the VLDB Endowment (PVLDB)*, 7(14):1845–1856.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2411–2420.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 820–828.

Ching Hoi Andy Hong, Jesse Prabawa Gozali, and Min-Yen Kan. 2009. FireCite: Lightweight real-time reference string extraction from webpages. In *Proceedings of the Workshop on Text and Citation Analysis for Scholarly Digital Libraries (NLPIR4DL)*, pages 71–79.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 427–431.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

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

Bing Liu, Robert Grossman, and Yanhong Zhai. 2003. Mining data records in web pages. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 601–606.

Fei Liu, Timothy Baldwin, and Trevor Cohn. 2017. Capturing long-range contextual dependencies with memory-enhanced conditional random fields. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 555–565.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1064–1074.

Adi Omari, Benny Kimelfeld, Eran Yahav, and Sharon Shoham. 2016. Lossless separation of web pages into layout code and data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1805–1814.

Fuchun Peng and Andrew McCallum. 2006. Information extraction from research papers using conditional random fields. *Information Processing and Management*, 42(4):963–979.

Jie Tang, Limin Yao, Duo Zhang, and Jing Zhang. 2010. A combination approach to web user profiling. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(1):1–38.

Dominika Tkaczyk, Paweł Szostek, Mateusz Fedoryszak, Piotr Jan Dendek, and Łukasz Bolikowski. 2015. CERMINE: automatic extraction of structured metadata from scientific literature. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(4):317–335.

Kai-Hsiang Yang and Jan-Ming Ho. 2010. Parsing publication lists on the web. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 444–447.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1480–1489.

Jia Zhu, Xingcheng Wu, Xueqin Lin, Changqin Huang, Gabriel Pui Cheong Fung, and Yong Tang. 2018. A novel multiple layers name disambiguation framework for digital libraries using dynamic clustering. *Scientometrics*, 114(3):781–794.

A Neural Transition-based Model for Nested Mention Recognition

Bailin Wang
University of Massachusetts
Amherst
`bailinwang@cs.umass.edu`

Wei Lu
Singapore University of Technology
and Design
`luwei@sutd.edu.sg`

Yu Wang and **Hongxia Jin**
Samsung Research America
`{yu.wang1, hongxia.jin}@samsung.com`

Abstract

It is common that entity mentions can contain other mentions recursively. This paper introduces a scalable transition-based method to model the nested structure of mentions. We first map a sentence with nested mentions to a designated forest where each mention corresponds to a constituent of the forest. Our shift-reduce based system then learns to construct the forest structure in a bottom-up manner through an action sequence whose maximal length is guaranteed to be three times of the sentence length. Based on Stack-LSTM which is employed to efficiently and effectively represent the states of the system in a continuous space, our system is further incorporated with a character-based component to capture letter-level patterns. Our model achieves the state-of-the-art results on ACE datasets, showing its effectiveness in detecting nested mentions.[1]

1 Introduction

There has been an increasing interest in named entity recognition or more generally recognizing entity mentions[2] (Alex et al., 2007; Finkel and Manning, 2009; Lu and Roth, 2015; Muis and Lu, 2017) that the nested hierarchical structure of entity mentions should be taken into account to better facilitate downstream tasks like question answering (Abney et al., 2000), relation extraction (Mintz et al., 2009; Liu et al., 2017), event extraction (Riedel and McCallum, 2011; Li et al., 2013), and coreference resolution (Soon et al., 2001; Ng and Cardie, 2002; Chang et al., 2013). Practically, the mentions with nested structures frequently exist in news (Doddington et al., 2004) and biomedical documents (Kim et al., 2003). For example in Figure 1, "UN Secretary General" of type Person also contains "UN" of type Organization.

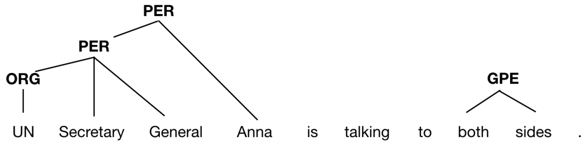

Figure 1: An example sentence of nested mentions represented in the structure of forest. PER:Person, ORG:Organization, GPE:Geo-Political Entity.

Traditional sequence labeling models such as conditional random fields (CRF) (Lafferty et al., 2001) do not allow hierarchical structures between segments, making them incapable to handle such problems. Finkel and Manning (2009) presented a chart-based parsing approach where each sentence with nested mentions is mapped to a rooted constituent tree. The issue of using a chart-based parser is its cubic time complexity in the number of words in the sentence.

To achieve a scalable and effective solution for recognizing nested mentions, we design a transition-based system which is inspired by the recent success of employing transition-based methods for constituent parsing (Zhang and Clark, 2009) and named entity recognition (Lou et al., 2017), especially when they are paired with neural networks (Watanabe and Sumita, 2015). Generally, each sentence with nested mentions is mapped to a forest where each outermost mention forms a tree consisting of its inner mentions. Then our transition-based system learns to construct this forest through a sequence of shift-reduce actions. Figure 1 shows an example of such a forest. In contrast, the tree structure by Finkel and Manning (2009) further uses a root node to connect all tree elements. Our forest representation eliminates the root node so that the number of actions required to

---

[1] We make our implementation available at `https://github.com/berlino/nest-trans-em18`.

[2] Mentions are defined as references to entities that could be named, nominal or pronominal (Florian et al., 2004).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1011–1017
Brussels, Belgium, October 31 - November 4, 2018.

construct it can be reduced significantly.

Following (Dyer et al., 2015), we employ Stack-LSTM to represent the system's state, which consists of the states of input, stack and action history, in a continuous space incrementally. The (partially) processed nested mentions in the stack are encoded with recursive neural networks (Socher et al., 2013) where composition functions are used to capture dependencies between nested mentions. Based on the observation that letter-level patterns such as capitalization and prefix can be beneficial in detecting mentions, we incorporate a character-level LSTM to capture such morphological information. Meanwhile, this character-level component can also help deal with the out-of-vocabulary problem of neural models. We conduct experiments in three standard datasets. Our system achieves the state-of-the-art performance on ACE datasets and comparable performance in GENIA dataset.

2 Related Work

Entity mention recognition with nested structures has been explored first with rule-based approaches (Zhang et al., 2004; Zhou et al., 2004; Zhou, 2006) where the authors first detected the innermost mentions and then relied on rule-based post-processing methods to identify outer mentions. McDonald et al. (2005) proposed a structured multi-label model to represent overlapping segments in a sentence. but it came with a cubic time complexity in the number of words. Alex et al. (2007) proposed several ways to combine multiple conditional random fields (CRF) (Lafferty et al., 2001) for such tasks. Their best results were obtained by cascading several CRF models in a specific order while each model is responsible for detecting mentions of a particular type. However, such an approach cannot model nested mentions of the same type, which frequently appear.

Lu and Roth (2015) and Muis and Lu (2017) proposed new representations of mention hypergraph and mention separator to model *overlapping mentions*. However, the nested structure is not guaranteed in such approaches since overlapping structures additionally include the *crossing structures*[3], which rarely exist in practice (Lu and Roth, 2015). Also, their representations did not model the dependencies between nested mentions explicitly, which may limit their performance. In contrast, the chart-based parsing method (Finkel and Manning, 2009) can capture the dependencies between nested mentions with composition rules which allow an outer entity to be influenced by its contained entities. However, their cubic time complexity makes them not scalable to large datasets.

$$\begin{array}{cc}
\text{Initial State} & [\phi, 0, \phi] \\
\text{Final State} & [S, n, A] \\
\\
\textsc{Shift} & \dfrac{[S,\ i,\ A]}{[S|w,\ i{+}1,\ A|\textsc{Shift}]} \\
\\
\textsc{Reduce-X} & \dfrac{[S|t_1 t_0,\ i,\ A]}{[S|X,\ i,\ A|\textsc{Reduce-X}]} \\
\\
\textsc{Unary-X} & \dfrac{[S|t_0,\ i,\ A]}{[S|X,\ i,\ A|\textsc{Unary-X}]}
\end{array}$$

Figure 2: Deduction rules. $[S, i, A]$ denotes stack, buffer front index and action history respectively.

As neural network based approaches are proven effective in entity or mention recognition (Collobert et al., 2011; Lample et al., 2016; Huang et al., 2015; Chiu and Nichols, 2016; Ma and Hovy, 2016), recent efforts focus on incorporating neural components for recognizing nested mentions. Ju et al. (2018) dynamically stacked multiple LSTM-CRF layers (Lample et al., 2016), detecting mentions in an inside-out manner until no outer entities are extracted. Katiyar and Cardie (2018) used recurrent neural networks to extract features for a hypergraph which encodes all nested mentions based on the BILOU tagging scheme.

3 Model

Specifically, given a sequence of words $\{x_0, x_1, \ldots, x_n\}$, the goal of our system is to output a set of mentions where nested structures are allowed. We use the forest structure to model the nested mentions scattered in a sentence, as shown in Figure 1. The mapping is straightforward: each outermost mention forms a tree where the mention is the root and its contained mentions correspond to constituents of the tree.[4]

3.1 Shift-Reduce System

Our transition-based model is based on the shift-reduce parser for constituency parsing (Watan-

---

[3] For example, in a four-word sentence ABCD, the phrase ABC and BCD together form a crossing structure.

[4] Note that words that are not contained in any mention each forms a single-node tree.

abe and Sumita, 2015), which adopts (Zhang and Clark, 2009; Sagae and Lavie, 2005). Generally, our system employs a stack to store (partially) processed nested elements. The system's state is defined as $[S, i, A]$ which denotes stack, buffer front index and action history respectively. In each step. an action is applied to change the system's state.

Our system consists of three types of transition actions, which are also summarized in Figure 2:

- SHIFT pushes the next word from buffer to the stack.
- REDUCE-X pops the top two items t_0 and t_1 from the tack and combines them as a new tree element $\{\mathrm{X} \rightarrow t_0 t_1\}$ which is then pushed onto the stack.
- UNARY-X pops the top item t_0 from the stack and constructs a new tree element $\{\mathrm{X} \rightarrow t_0\}$ which is pushed back to the stack.

Since the shift-reduce system assumes unary and binary branching, we binarize the trees in each forest in a left-branching manner. For example, if three consecutive words A, B, C are annotated as Person, we convert it into a binary tree $\{Person \rightarrow \{Person{*} \rightarrow A, B\}, C\}$ where $Person{*}$ is a temporary label for $Person$. Hence, the X in reduce- actions will also include such temporary labels.

Note that since most words are not contained in any mention, they are only shifted to the stack and not involved in any reduce- or unary- actions. An example sequence of transitions can be found in Figure 3. Our shift-reduce system is different from previous parsers in terms of the terminal state. 1) It does not require the terminal stack to be a rooted tree. Instead, the final stack should be a forest consisting of multiple nested elements with tree structures. 2) To conveniently determine the ending of our transition process, we add an auxiliary symbol \$ to each sentence. Once it is pushed to the stack, it implies that all deductions of actual words are finished. Since we do not allow unary rules between labels like X1 $\rightarrow$ X2, the length of maximal action sequence is $3n$.[5]

3.2 Action Constraints

To make sure that each action sequence is valid, we need to make some hard constraints on the action to take. For example, reduce- action can only be conducted when there are at least two elements in the stack. Please see the Appendix for the full list of restrictions. Formally, we use $\mathcal{V}(S, i, A)$ to denote the valid actions given the parser state. Let us denote the feature vector for the parser state at time step k as $\mathbf{p}_k$. The distribution of actions is computed as follows:

$$p(z_k \mid \mathbf{p}_k) = \frac{\exp\left(\mathbf{w}_{z_k}^{\top} \mathbf{p}_k + b_{z_k}\right)}{\sum_{z' \in \mathcal{V}(S,i,A)} \exp\left(\mathbf{w}_{z'}^{\top} \mathbf{p}_k + b_{z'}\right)} \tag{1}$$

where $\mathbf{w}_z$ is a column weight vector for action z, and b_z is a bias term.

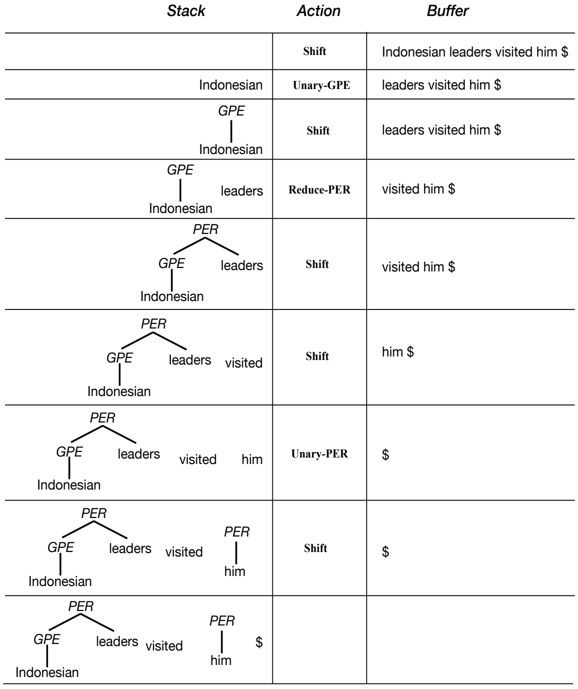

Figure 3: An example sequence of transition actions for the sentence "Indonesian leaders visited him". \$ is the special symbol indicating the termination of transitions. PER:Person, GPE:Geo-Political Entity.

3.3 Neural Transition-based Model

We use neural networks to learn the representation of the parser state, which is $\mathbf{p}_k$ in (1).

Representation of Words

Words are represented by concatenating three vectors:

$$\mathbf{e}_{x_i} = [\mathbf{e}_{w_i}, \mathbf{e}_{p_i}, \mathbf{c}_{w_i}] \tag{2}$$

where $\mathbf{e}_{w_i}$ and $\mathbf{e}_{p_i}$ denote the embeddings for i-th word and its POS tag respectively. $\mathbf{c}_{w_i}$ denotes the representation learned by a character-level model

---

[5] In this case, each word is shifted (n) and involved in a unary action (n). Then all elements are reduced to a single node ($n - 1$). The last action is to shift the symbol \$.

using a bidirectional LSTM. Specifically, for character sequence $s_0, s_1, \dots, s_n$ in the i-th word, we use the last hidden states of forward and backward LSTM as the character-based representation of this word, as shown below:

$$\mathbf{c}_{w_i} = [\overrightarrow{\text{LSTM}_\text{c}}(s_0, \dots, s_n), \overleftarrow{\text{LSTM}_\text{c}}(s_0, \dots, s_n)] \tag{3}$$

Representation of Parser States

Generally, the buffer and action history are encoded using two vanilla LSTMs (Graves and Schmidhuber, 2005). For the stack that involves popping out top elements, we use the Stack-LSTM (Dyer et al., 2015) to efficiently encode it.

Formally, if the unprocessed word sequence in the buffer is $x_i, x_{i+1}, \dots, x_n$ and action history sequence is $a_0, a_1, \dots, a_{k-1}$, then we can compute buffer representation $\mathbf{b}_k$ and action history representation $\mathbf{a}_k$ at time step k as follows:

$$\mathbf{b}_k = \overleftarrow{\text{LSTM}_\text{b}}[\mathbf{e}_{x_i}, \dots, \mathbf{e}_{x_n}] \tag{4}$$

$$\mathbf{a}_k = \overrightarrow{\text{LSTM}_\text{a}}[\mathbf{e}_{a_0}, \dots, \mathbf{e}_{a_{k-1}}] \tag{5}$$

where each action is also mapped to a distributed representation $\mathbf{e}_a$.[6] For the state of the stack, we also use an LSTM to encode a sequence of tree elements. However, the top elements of the stack are updated frequently. Stack-LSTM provides an efficient implementation that incorporates a stack-pointer.[7] Formally, the state of the stack $\mathbf{b}_k$ at time step k is computed as:

$$\mathbf{s}_k = \text{Stack-LSTM}[\mathbf{h}_{t_m}, \dots, \mathbf{h}_{t_0}] \tag{6}$$

where $\mathbf{h}_{t_i}$ denotes the representation of the i-th tree element from the top, which can be computed recursively similar to Recursive Neural Network (Socher et al., 2013) as follows:

$$\mathbf{h}_{parent} = \mathbf{W}_{u,l}^{\top} \mathbf{h}_{child} + \mathbf{b}_{u,l} \tag{7}$$

$$\mathbf{h}_{parent} = \mathbf{W}_{b,l}^{\top} [\mathbf{h}_{lchild}, \mathbf{h}_{rchild}] + \mathbf{b}_{u,l} \tag{8}$$

where $\mathbf{W}_{u,l}$ and $\mathbf{W}_{b,l}$ denote the weight matrices for unary(u) and binary(b) composition with parent node being label(l). Note that the composition function is distinct for each label l. Recall that the leaf nodes of each tree element are raw words. Instead of representing them with their original embeddings introduced in Section 3.3, we found that concatenating the buffer state in (5) are beneficial during our initial experiments. Formally, when a word x_i is shifted to the stack at time step k, its representation is computed as:

$$\mathbf{h}_{leaf} = \mathbf{W}_{leaf}^{\top} [\mathbf{e}_{x_i}, \mathbf{b}_k] + \mathbf{b}_{leaf} \tag{9}$$

Models	ACE04	ACE05	GENIA	w/s
Finkel and Manning (2009)	-	-	70.3	$38^{\dagger}$
Lu and Roth (2015)	62.8	62.5	70.3	454
Muis and Lu (2017)	64.5	63.1	70.8	263
Katiyar and Cardie (2018)	72.7	70.5	73.6	-
Ju et al. (2018) [8]	-	72.2	**74.7**	-
Ours	**73.3**	**73.0**	73.9	1445
- char-level LSTM	72.3	71.9	72.1	1546
- pre-trained embeddings	71.3	71.5	72.0	1452
- dropout layer	71.7	72.0	72.7	1440

Table 1: Main results in terms of F_1 score (%). w/s: # of words decoded per second, number with † is retrieved from the original paper.

Finally, the state of the system $\mathbf{p}_k$ is the concatenation of the states of buffer, stack and action history:

$$\mathbf{p}_k = [\mathbf{b}_k, \mathbf{s}_k, \mathbf{a}_k] \tag{10}$$

Training

We employ the greedy strategy to maximize the log-likelihood of the local action classifier in (1). Specifically, let z_{ik} denote the k-th action for the i-th sentence, the loss function with ℓ_2 norm is:

$$\mathcal{L}(\theta) = -\sum_i \sum_k \log p(z_{ik}) + \frac{\lambda}{2} \|\theta\|^2 \tag{11}$$

where λ is the ℓ_2 coefficient.

4 Experiments

We mainly evaluate our models on the standard ACE-04, ACE-05 (Doddington et al., 2004), and GENIA (Kim et al., 2003) datasets with the same splits used by previous research efforts (Lu and Roth, 2015; Muis and Lu, 2017). In ACE datasets, more than 40% of the mentions form nested structures with some other mention. In GENIA, this number is 18%. Please see Lu and Roth (2015) for the full statistics.

4.1 Setup

Pre-trained embeddings GloVe (Pennington et al., 2014) of dimension 100 are used to initialize the

---

[6] Note that LSTM_b runs in a right-to-left order such that the output can represent the contextual information of x_i.

[7] Please refer to Dyer et al. (2015) for details.

[8] Note that in ACE2005, Ju et al. (2018) did their experiments with a different split from Lu and Roth (2015) and Muis and Lu (2017) which we follow as our split.

word vectors for all three datasets.[9] The embeddings of POS tags are initialized randomly with dimension 32. The model is trained using Adam (Kingma and Ba, 2014) and a gradient clipping of 3.0. Early stopping is used based on the performance of development sets. Dropout (Srivastava et al., 2014) is used after the input layer. The ℓ_2 coefficient λ is also tuned during development process.

4.2 Results

The main results are reported in Table 1. Our neural transition-based model achieves the best results in ACE datasets and comparable results in GENIA dataset in terms of F_1 measure. We hypothesize that the performance gain of our model compared with other methods is largely due to improved performance on the portions of nested mentions in our datasets. To verify this, we design an experiment to evaluate how well a system can recognize nested mentions.

Handling Nested Mentions

The idea is that we split the test data into two portions: sentences with and without nested mentions. The results of GENIA are listed in Table 2. We can observe that the margin of improvement is more significant in the portion of nested mentions, revealing our model's effectiveness in handling nested mentions. This observation helps explain why our model achieves greater improvement in ACE than in GENIA in Table 1 since the former has much more nested structures than the latter. Moreover, Ju et al. (2018) performs better when it comes to non-nested mentions possibly due to the CRF they used, which globally normalizes each stacked layer.

Decoding Speed

Note that Lu and Roth (2015) and Muis and Lu (2017) also feature linear-time complexity, but with a greater constant factor. To compare the decoding speed, we re-implemented their model with the same platform (PyTorch) and run them on the same machine (CPU: Intel i5 2.7GHz). Our model turns out to be around 3-5 times faster than theirs, showing its scalability.

	GENIA					
	Nested			Non-Nested		
	P	R	F_1	P	R	F_1
Lu and Roth (2015)	76.3	60.8	67.7	73.1	70.7	71.9
Muis and Lu (2017)	76.5	60.3	67.4	74.8	71.3	73.0
Ju et al. (2018)	79.4	63.6	70.6	78.5	77.5	78.0
Ours	80.3	64.6	71.6	76.8	73.9	75.3

Table 2: Results (%) on different types of sentences on the GENIA dataset.

Ablation Study

To evaluate the contribution of neural components including pre-trained embeddings, the character-level LSTM and dropout layers, we test the performances of ablated models. The results are listed in Table 1. From the performance gap, we can conclude that these components contribute significantly to the effectiveness of our model in all three datasets.

5 Conclusion and Future Work

In this paper, we present a transition-based model for nested mention recognition using a forest representation. Coupled with Stack-LSTM for representing the system's state, our neural model can capture dependencies between nested mentions efficiently. Moreover, the character-based component helps capture letter-level patterns in words. The system achieves the state-of-the-art performance in ACE datasets.

One potential drawback of the system is the greedy training and decoding. We believe that alternatives like beam search and training with exploration (Goldberg and Nivre, 2012) could further boost the performance. Another direction that we plan to work on is to apply this model to recognizing overlapping and entities that involve discontinuous spans (Muis and Lu, 2016) which frequently exist in the biomedical domain.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. We also thank Meizhi Ju for providing raw predictions and helpful discussions. This work was done after the first author visited Singapore University of Technology and Design. This work is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156.

---

[9] We also additionally tried using embeddings trained on PubMed for GENIA but the performance was comparable.

References

Steven Abney, Michael Collins, and Amit Singhal. 2000. Answer extraction. In *Proc. of the sixth conference on applied natural language processing*.

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Proc. of BioNLP*.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proc. of EMNLP*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Proc. of LREC*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proc. of EMNLP*.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proc. of HLT-NAACL*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. *Proceedings of COLING 2012*, pages 959–976.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*.

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

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proc. of NAACL-HLT*.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proc. of NAACL-HLT*.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL-HLT*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. of ACL*.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. In *Proc. of EMNLP*.

Yinxia Lou, Yue Zhang, Tao Qian, Fei Li, Shufeng Xiong, and Donghong Ji. 2017. A transition-based joint model for disease named entity recognition and normalization. *Bioinformatics*.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proc. of EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of ACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Flexible text segmentation with structured multilabel classification. In *Proc. of HLT-EMNLP*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proc. of ACL-IJCNLP*.

Aldrian Obaja Muis and Wei Lu. 2016. Learning to recognize discontiguous entities. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 75–84.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proc. of EMNLP*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

S. Riedel and A. McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proc. of EMNLP*.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *IWPT*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proc. of ACL*.

Jie Zhang, Dan Shen, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2004. Enhancing hmm-based biomedical named entity recognition by studying special phenomena. *Journal of Biomedical Informatics*, 37(6):411–422.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proc. of IWPT*.

Guodong Zhou. 2006. Recognizing names in biomedical texts using mutual information independence model and svm plus sigmoid. *International Journal of Medical Informatics*, 75(6):456–467.

Guodong Zhou, Jie Zhang, Jian Su, Dan Shen, and Chewlim Tan. 2004. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics*, 20(7):1178–1190.

Appendix

The action constraints are listed as follows:

- The SHIFT action is valid only when the buffer is not empty.
- The UNARY-X actions are valid only when the stack is not empty.
- The REDUCE-X actions are valid only when the stack has two or more elements.
- If the top element of the stack is labeled, then unary actions are not valid. That is, $\{X1 \rightarrow X2\}$ is not allowed.
- If only one of the top two elements of the stack is temporary, say X*, then among all reduce actions, only REDUCE-X* and REDUCE-X are valid.
- If the top two elements of the stack are both temporary, then all reduce actions are not allowed.
- If one of the elements in the stack is temporary, say X*, which means it is not finished, then last terminal symbol $ cannot be shifted until it is reduced to X.

Genre Separation Network with Adversarial Training for Cross-genre Relation Extraction

Ge Shi[1], **Chong Feng**[1]*, **Lifu Huang**[2], **Boliang Zhang**[2],
Heng Ji[2], **Lejian Liao**[1], **Heyan Huang**[1]
[1] Beijing Engineering Research Center of High Volume Language Information Processing
and Cloud Computing Applications, School of Computer,
Beijing Institute of Technology, Beijing, 100081, China
`{shige, fengchong, liaolj, hhy63}@bit.edu`
[2] Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
`{huangl7, zhangb8, jih}@rpi.edu`

Abstract

Relation Extraction suffers from dramatical performance decrease when training a model on one genre and directly applying it to a new genre, due to the distinct feature distributions. Previous studies address this problem by discovering a shared space across genres using manually crafted features, which requires great human effort. To effectively automate this process, we design a genre-separation network, which applies two encoders, one genre-independent and one genre-shared, to explicitly extract genre-specific and genre-agnostic features. Then we train a relation classifier using the genre-agnostic features on the source genre and directly apply to the target genre. Experiment results on three distinct genres of the ACE dataset show that our approach achieves up to 6.1% absolute F1-score gain compared to previous methods. By incorporating a set of external linguistic features, our approach outperforms the state-of-the-art by 1.7% absolute F1 gain. We make all programs of our model publicly available for research purpose [1] .

1 Introduction

Relation extraction aims to identify and categorize the semantic relation between two entity mentions based on the contexts within the sentence. Supervised learning approaches have shown to be effective on this task. However, as relation extraction highly depends on information about entities and their contexts, a supervised model trained in one genre suffers from dramatical performance decrease when applied to a new genre, due to the distinct contexts among various genres.

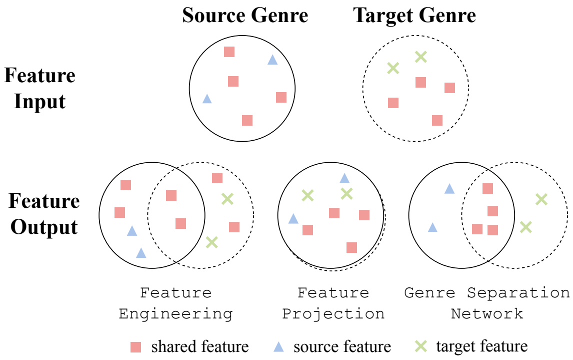

Figure 1: Comparison of Genre Separation Methods.

Previous studies (Plank and Moschitti, 2013; Nguyen and Grishman, 2014, 2015; Yu et al., 2015; Gormley et al., 2015) tackle this problem by manually crafting genre-agnostic features such as word clusters and word embeddings, to train a genre-shared relation extractor. These methods suffer from information loss due to the limited human knowledge to capture all genre-agnostic features. As depicted in Figure 1, where red rectangles are features shared by two genres, and blue and green triangles are source and target genre features respectively, *Feature Engineering* only captures a portion of the genre-agnostic features. Fu et al. (2017), depicted as *Feature Projection*, applies a domain adversarial neural network to automatically project the source and target genre features into one unified feature space. However, it unnecessarily introduces genre-specific features which undermine the overall performance.

To address these problems, we propose a genre-separation network, which consists of two separate Convolutional Neural Networks (CNNs) to automatically separates genre-specific and genre-agnostic features for each genre, which is depicted as *Genre Separation Network* in Figure 1. To avoid information loss during feature encoding, we reconstruct the original input from the two separate feature spaces via a novel reconstruction loss. Then we use an adversarial similarity loss to limit the genre-agnostic features into one fea-

---

*Corresponding author

[1] We make all cleaned codes and resources publicly available at https://github.com/Garym713/Genre-Separation-Network-for-Relation-Extraction

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1018–1023
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

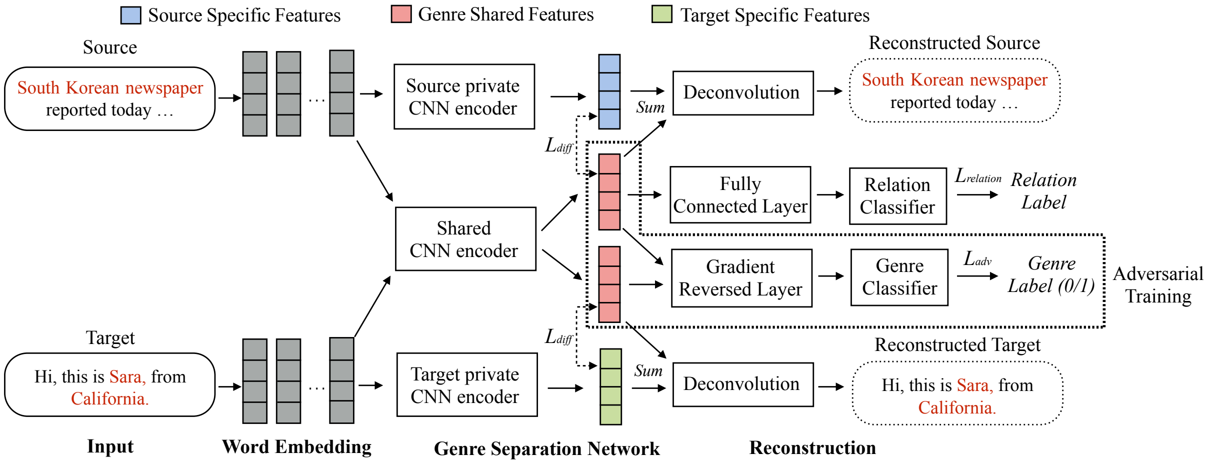

Figure 2: Overall genre separation framework for cross-genre relation extraction

ture space. The genre-agnostic features are finally used to predict entity relations in the source and target genres.

2 Approach

2.1 Overview

An overview of our framework is presented in Figure 2. We formulate the task as follows: given a labeled source genre corpus $S = \{(s_1, e_{11}, e_{12}, r_1), ..., (s_n, e_{n1}, e_{n2}, r_n)\}$, where $s_i = [w_{i1}, ..., w_{im}]$ denotes a sentence. e_{i1} and e_{i2} denote two entity mentions, and r_i denotes the relation between e_{i1} and e_{i2}, we build a relation extraction model on S and apply it to a different target genre corpus $T = \{(\hat{s_1}, \hat{e_{11}}, \hat{e_{12}}), ..., (\hat{s_n}, \hat{e_{n1}}, \hat{e_{n2}})\}$.

2.2 Genre Separation Network (GSN)

As shown in Figure 1, our goal is to distinguish the genre-agnostic features (red rectangles) and genre-specific features (blue triangles and green crosses). Using source genre as an example, we apply a source private CNN encoder on the source sentence to generate the source-specific feature representation f_s^p, and a shared CNN encoder to generate genre-agnostic feature f_s^c. Similarly, we get f_t^p and f_t^c from the target private CNN encoder and the shared CNN encoder respectively. To separate f_s^p from f_s^c and separate f_t^p from f_t^c, we introduce a *difference loss* following previous studies (Bousmalis et al., 2016; Liu et al., 2017). More details will be elaborated below.

Formally, given a source sentence (s, e_1, e_2, r) where $s = [w_1, ..., w_m]$, for each word w_{ik}, we generate a multi-type embedding: $\tilde{v}_i = [v_i, p_i, \tilde{p}_i, t_i, \tilde{t}_i, c_i, \eta_i]$ where v_i denotes a pre-trained word embedding. p_i and $\tilde{p}_i$ are position embeddings (Al-Badrashiny et al., 2017) indicating the distance from w_i to e_1 and e_2 respectively. t_i and $\tilde{t}_i$ are entity type embedding (Ren et al., 2016; Huang et al., 2016) of e_1 and e_2. c_i is the chunking embedding, and η_i is a binary digit indicating whether the word is within the shortest dependency path between e_1 and e_2 (Bunescu and Mooney, 2005; Liu et al., 2015; Huang et al., 2017). All these embeddings except pre-trained word embedding are randomly initialized and optimized during training. Thus the input layer is a sequence of word representations $V = \{\tilde{v}_1, \tilde{v}_2, ..., \tilde{v}_n\}$. We then apply the convolution weights W to each sliding n-gram phrase g_j with a biased vector b, i.e., $g_j^{'} = \tanh(W \cdot V) + b$. All n-gram representations $g_j^{'}$ are further used to get an overall vector representation f by max-pooling.

Once we obtain f_s^p, f_s^c, f_s^p and f_s^c, we compute the *difference loss*:

$$L_{diff} = ||f_s^{p\top} \cdot f_s^c + f_t^{p\top} \cdot f_t^c||_2^F$$

where $||.||_2^F$ represents the squared Frobenius norm.

To limit the genre-agnostic features from various genres into a shared feature space, we further design a genre adversarial training component. We take the genre-agnostic features from both source genre and target genre as input to a Gradient Reversed Layer(GRL) (Ganin et al., 2016), which acts as a general hidden layer in forward process and reverses the gradient in loss backward phase to confuse the genre classifier, so that it cannot distinguish the input features from the source genre to the target genre:

$$L_{adv} = \sum_{i=0}^{N_s+N_t} d_i l \log(\hat{d_i}) + (1-d_i)\log(1-\hat{d_i})$$

where $d_i \in \{0,1\}$ indicates the samples from the source genre or the target genre, and N_s, N_t refer to the number of examples in the source genre and the target genre respectively. The term $\hat{d_i}$ represents the probability of the sample from the source genre, which is acquired by a linear function of the genre classifier.

2.3 Genre Reconstruction

Till now, we can separate the features of each genre into two separated feature spaces by optimizing L_{diff} and L_{adv}. However, there is no guarantee that the separated feature spaces are actually meaningful. From equation L_{diff}, we can see that the f_s^p, f_t^p would be easily optimized to zero if we did not place a constraint, in which case the model would fail to train. Therefore, we further reconstruct the input sentence from both genre-specific features and genre-agnostic features.

For each genre, e.g., the source genre, we first sum the genre-specific feature vector f_s^p and genre-agnostic feature vector f_s^c, i.e., $f_s = f_s^p + f_s^c$. We take f_s as input to an unpooling layer (Zeiler and Fergus, 2014) followed by a deconvolutional neural netowrk (DcNN) (Xu et al., 2014). The output of DcNN will include the same number of decoded vectors $V^* = \{\tilde{v}_1^*, \tilde{v}_2^*, ..., \tilde{v}_n^*\}$ as input $V = \{\tilde{v}_1, \tilde{v}_2, ..., \tilde{v}_n\}$. We optimize the DcNN with the following reconstruction loss:

$$L_{rec} = 1 - \sum_{i=0}^{n} |cos(\tilde{v}_i, \tilde{v}_i^*)|$$

where n indicates the total number of words in the input sentence, $\tilde{v}_i$ represents the input word representation described in Section 2.2, and $\tilde{v}_i^*$ is the corresponding reconstructed vector from DcNN.

2.4 Cross Genre Relation Extraction

We next utilize the genre-agnostic features from the source genre f_s^c to train a relation classifier. We first feed f_s^c into a fully connected layer and obtain a dense vector, then we use a linear projection function with a softmax as the relation classifier to determine the relation type

$$L_{relation} = \sum_{i=0}^{N_s}\sum_{k=0}^{K} -x_k \log(x_k)$$

where K is the total number of relation types. x_k represents the probability of entities being classified to category k.

We finally linearly combine all the loss functions and jointly optimize the model using SGD (Bottou, 2010).

$$L = L_{relation} + \alpha L_{diff} + \beta L_{rec} + \gamma L_{adv}$$

where α, β, γ denote the weights of various losses.[2]

3 Experiments

3.1 Data and Parameters

We evaluate our approach on the English portion of ACE2005 dataset (Walker et al., 2006; Ji et al., 2010; Hong et al., 2015; Yu et al., 2016). It covers 6 genres: Newswire (nw), Broadcast Conversation (bc), Broadcast News (bn), Telephone Speech (cts), Usenet Newsgroups (un), and Weblogs (wl) and 11 relation types. Following previous work (Yu et al., 2015; Nguyen and Grishman, 2015; Gormley et al., 2015), we use newswire and broadcast news (nw&bn) as training data, half of bc as development set, and test the model on the remaining half of bc, cts, wl. We conduct the same preprocessing steps as previous work and yield 43,497 entity pairs in total for training.

Table 3.1 shows the hyper-parameters that we use to train our model.

Hyper-parameters	Value
# of Filters in Shared/Private CNN encoder	800
Filter Width	3
Hidden Size of Fully Connected Layer	300
Position Embedding Size	25
Entity Type/Chunking Embedding Size	25
Optimizer	SGD
Learning rate	0.001
Pre-trained Word Embedding	Glove-100[3]

Table 1: Hyper-parameters

3.2 Baseline Models

We compare our approach with the following methods:

FCM (Gormley et al., 2015) is a feature combination model which composes word embeddings with traditional linguistic features.

Hybrid FCM (Gormley et al., 2015) incorporates many more selected linguistic features compared to FCM.

---

[2] We set $\alpha = 0.075, \beta = 0.01, \gamma = 0.25$ when the model performs the best on the development set.

LRFCM (Yu et al., 2015) is a feature compositional model which scales to more features and more labels.

Log-linear & DNN (Nguyen and Grishman, 2015) explores CNN, Bi-GRU, Forward GRU, Backward GRU, and log-linear model for relation extraction. We compare against the performances of individual models instead of assembled models.

CNN+DANN (Fu et al., 2017) utilizes domain adversarial training to automatically extract genre-agnostic features for source and target genre within one feature space.

3.3 Comparison and Analysis

Table 2 shows the cross-genre relation extraction performance among various methods. Our approach significantly outperforms all previous baselines by 1.2%-1.7% (F1). Table 3 presents the results without using extra linguistic features (only embedding based features), our approach achieves 2.9%-6.1% absolute gain over baselines. The ablation test by removing each component at a time justifies the contribution of each method. The difference and reconstruction components ensure the features to be separated into shared and private spaces, and they can remove redundant genre-specific features to some extent. That's why we got a significant F-score improvement when only utilizing these two components. The adversarial training component can further encourage the features of each genre from the shared encoder to be close to each other, thus the performance is further improved. We also conduct ablation experiments on each feature components. Among the linguistic features we used, the entity type and position features contribute the most to the performance. For example, the relation extraction performance decreases by about 8% if removing the entity type feature. We analyze the reasons and find that the entity type feature is vital to ensure the types of two entity mentions to be consistent with the hard entity type constraint of each relation type defined in ACE schema.

For the remaining errors, we notice that our model easily fails to predict relations between nested entity mentions. For example, in "*Our president has put homeland security in the hands of failed Republican hacks.*", our model mistakenly predicts the relationship between *Republican* and *failed Republican hacks* as *None* instead of *organization-affiliation*, due to the lack of context information. Besides, we also observe some failed cases where the two entities are separated with a extreme wide context, which suggests us to incorporate dependency path based deep neural networks into the framework.

System	bc	cts	wl
FCM	61.9	52.93	50.36
Hybrid FCM	63.48	56.12	55.17
LRFCM	59.4	-	-
Log-linear	57.83	53.14	53.06
CNN	63.26	55.63	53.91
Bi-GRU	63.07	56.47	53.65
Forward GRU	61.44	54.93	55.10
Backward GRU	60.82	56.03	51.78
CNN+DANN	65.16	-	-
w/o Difference	59.87	54.10	52.73
w/o Adversarial	64.42	57.32	55.63
w/o Reconstruction	59.12	53.48	53.17
Our Approach	**66.38**	**57.92**	**56.84**

Table 2: Cross Genre Relation Extraction Performances (Macro F-score %) on Various Genres. *w/o Difference* means to ablate the L_{diff} loss. *w/o Adversarial* means to ablate the adversarial training component. *w/o Reconstruction* means to ablate the genre reconstruction component.

System	bc	cts	wl
CNN	46.3	40.8	35.8
GRU	45.2	40.2	35.1
Bi-GRU	46.7	41.2	36.5
Our Approach	**52.8**	**45.3**	**39.4**

Table 3: Cross Genre Relation Extraction Performances (Macro F-score %) on Various Genres (without linguistic features)

4 Related Work

Previous studies on cross-genre relation extraction either manually or automatically extract genre-agnostic features (Plank and Moschitti, 2013; Nguyen and Grishman, 2014; Yu et al., 2015; Gormley et al., 2015; Nguyen and Grishman, 2015), suffering from human labor and limited coverage of effective features, or automatically project source and target genres into one unified feature space and learn genre shared features (Fu et al., 2017), which inevitably introduces noise from genre specific features. Compared with these methods, our approach separates genre-specific features from genre-agnostic features first, and then automatically extracts meaningful features for cross-genre relation extraction.

Our work is also related to studies on domain separation networks (Bousmalis et al., 2016; Liu et al., 2017; Chen et al., 2017), which explicitly extracts features from two separate subspaces: domain-specific and domain-agnostic. We adopt a similar framework for cross-genre relation extraction and introduce a novel reconstruction component which is proved to be suitable to relation extraction.

5 Conclusions

We propose a genre separation framework for cross-genre relation extraction. Without requiring human crafted features, this framework can effectively separate genre-specific features from genre-agnostic ones, and automatically extract meaningful features for the task. To ensure the separation of features within each genre and enforce the genre agnostic features from source genre and target genre to be in the same feature space, we design a difference loss and an adversarial training component. Experiments on various genres demonstrate the effectiveness of our framework. In the future, we will extend our framework to cross-lingual and cross-domain information extraction tasks.

6 Acknowledgements

We thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the National Key Research and Development Program of China No. 2017YFB1002101, the National Natural Science Foundation of China No. U1636203, China Scholarship Council CSC, No. 201706030131. RPI co-authors were supported by the U.S. DARPA AIDA Program No. FA8750-18-2-0014 and U.S. ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Mohamed Al-Badrashiny, Jason Bolton, Arun Tejasvi Chaganty, Kevin Clark, Craig Harman, Lifu Huang, Matthew Lamm, Jinhao Lei, Di Lu, Xiaoman Pan, et al. 2017. Tinkerbell: Cross-lingual cold-start knowledge base construction. In *TAC*.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 724–731. Association for Computational Linguistics.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. *arXiv preprint arXiv:1704.07556*.

Lisheng Fu, Thien Huu Nguyen, Bonan Min, and Ralph Grishman. 2017. Domain adaptation for relation extraction with domain adversarial neural network. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 425–429.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. *arXiv preprint arXiv:1505.02419*.

Yu Hong, Di Lu, Dian Yu, Xiaoman Pan, Xiaobin Wang, Yadong Chen, Lifu Huang, and Heng Ji. 2015. Rpi blender tac-kbp2015 system description. In *Proc. Text Analysis Conference (TAC2015)*.

Lifu Huang, Jonathan May, Xiaoman Pan, and Heng Ji. 2016. Building a fine-grained entity typing system overnight for a new x (x= language, domain, genre). *arXiv preprint arXiv:1603.03112*.

Lifu Huang, Avirup Sil, Heng Ji, and Radu Florian. 2017. Improving slot filling performance with attentive neural networks on dependency structures. *arXiv preprint arXiv:1707.01075*.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*, volume 3, pages 3–3.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. *arXiv preprint arXiv:1507.04646*.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 68–74.

Thien Huu Nguyen and Ralph Grishman. 2015. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1498–1507.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. 2014. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems*, pages 1790–1798.

Dian Yu, Xiaoman Pan, Boliang Zhang, Lifu Huang, Di Lu, Spencer Whitehead, and Heng Ji. 2016. Rpi blender tac-kbp2016 system description. In *Proceedings of the 2016 Text Analysis Conference (TAC2016)*.

Mo Yu, Matthew R Gormley, and Mark Dredze. 2015. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1374–1379.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Effective Use of Context in Noisy Entity Linking

David Mueller
Department of Computer Science
Johns Hopkins University*
dam@jhu.edu

Greg Durrett
Department of Computer Science
The University of Texas at Austin
gdurrett@cs.utexas.edu

Abstract

To disambiguate between closely related concepts, entity linking systems need to effectively distill cues from a mention's textual context. We investigate several techniques for using these cues in the task of noisy entity linking on short texts. Our starting point is a state-of-the-art attention-based model from prior work; while this model's attention typically identifies context that is topically relevant, it fails to identify some of the most indicative context words, especially those exhibiting lexical overlap with the true title. Augmenting the model with convolutional networks over characters still leaves it largely unable to pick up on these cues compared to sparse features that target them directly, indicating that automatically learning how to identify relevant character-level context features is a hard problem. Armed with these sparse features, our final system[1] outperforms past work on the WikilinksNED test set by 2.8% absolute.

1 Introduction

Effectively using an entity mention's context to disambiguate it is the crux of the entity linking task: in isolation, the mention *Richard Wright* could refer to three possible entities in Wikipedia's knowledge base corresponding to an artist, a musician, or an author. Previous work in this area has distilled context information by exploiting tf-idf features (Cucerzan, 2007; Milne and Witten, 2008; Ratinov et al., 2011), global link coherence (Hoffart et al.; Sil and Florian, 2016), cues from coreference (Cheng and Roth, 2013; Hajishirzi et al., 2013; Durrett and Klein, 2014), convolutional neural networks (Sun et al.; Francis-Landau et al., 2016), or more sophisticated neural architectures (Gupta et al., 2017; Sil et al., 2018). These approaches typically focus on aggregating information from a mix of sources, including long-range information from the textual context or other linked entities. While this approach is suitable for entity linking settings such as newswire (Bentivogli, 2010) and Wikipedia (Ratinov et al., 2011), we cannot always rely on this information in other settings like Twitter (Guo et al., 2013; Fang and Chang, 2014; Huang et al., 2014; Dredze et al., 2016), Snapchat (Moon et al., 2018), other web platforms (Eshel et al., 2017), or dialogue systems (Bowden et al., 2018). We need models that can make effective use of limited context windows in noisy settings.

In this work, we investigate this problem of effectively using context in the setting of the WikilinksNED dataset from Eshel et al. (2017). The examples in this dataset, which consists of 3.2 million entity disambiguation examples derived from Wikilinks (Singh et al., 2012), have at most 20 words of context on either side and usually no other mentions of the entity being disambiguated. We build off a state-of-the-art attentive LSTM model from prior work (Eshel et al., 2017) and show that despite its good performance, it fails to resolve some examples that human readers would find trivial. For example, disambiguating the identity of the song *Down* in Figure 1 is easy if we can recognize the nearby string *Jay Sean* in the context, but the model sometimes fails to do this.

We explore the performance of a standard attention mechanism as well as two modifications. First, we inject character information into the model through character-level CNNs; these give the model a deeper ability to recognize character correspondences between the context and entity title. However, these convolutional filters struggle to learn useful features in this noisy context and ultimately do not help performance. By contrast, sparse features explicitly targeting these overlaps

---

*Work done while at UT Austin.

[1] Code available at
`https://github.com/davidandym/wikilinks-ned`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1024–1029
Brussels, Belgium, October 31 - November 4, 2018.

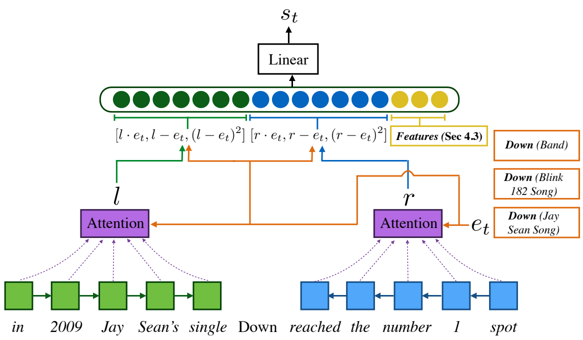

Figure 1: Neural entity linking model. *Down* has three possible link targets: in the attentive variants of our model, each target computes attention weights over GRUs that consume the left and right context. These representations are passed into a layer that compares them to the entity's embedding, yielding a final score which is normalized over all possible link targets.

appear to be more successful. We investigate the relation between our model's attention and what the sparse features learn. Our final model, using these features, achieves an accuracy of 75.8% on this dataset, substantially outperforming our baseline model as well as results from prior work.

2 Basic Model

The WikilinksNED dataset consists of entity mentions in context scraped from the web, with gold annotation derived from the fact that those mentions originally appeared with hyperlinks to Wikipedia. We denote the mention text (i.e., anchor text of the hyperlink) by m, and denote the left and right context of the mention by c_l and c_r respectively; these are at most 20 words. For this dataset, we can assume that the possible linked titles for a mention have been seen in training, and the main task is instead to disambiguate between them and identify the gold title t^*. We therefore follow prior work (Eshel et al., 2017) and take as candidates all gold entities in the training set whose mention was m rather than relying on a separate candidate generation scheme.

Our model places a distribution over titles $P(t|m, c_l, c_r)$, where t takes values in the set of candidate Wikipedia titles for that mention. This model, depicted in Figure 1, roughly follows that of Eshel et al. (2017), with some key differences, as we discuss in the rest of this section.

Embedding contexts Given an example of the form (m, c_l, c_r), our model first uses a GRU layer (Cho et al., 2014) over each context to convert c_l and c_r into continuous vector representations l and r, respectively. Our word embeddings are trained over Wikipedia as described in the following paragraph.

Embedding entities We follow the method of Eshel et al. (2017) for generating entity embeddings, using `word2vecf` (Levy and Goldberg, 2014) to jointly train word and entity embeddings simultaneously using Wikipedia article text. Each title t is associated in turn with each content word w in the article, yielding a set of (w, t) pairs that are consumed by the training procedure. This yields a set of title embeddings e_t which we can treat as distributed representations of entities.

Entity-context comparison We systematically compare the representations for l, r, and e_t as follows:

$$[l \cdot e_t, r \cdot e_t, l - e_t, r - e_t, (l - e_t)^2, (r - e_t)^2]$$

where $\cdot$ denotes the conventional dot product and the other comparisons are elementwise. These features form the input to a final feedforward layer which produces a real-valued score s_t for the given title. Repeating this computation for each title, our model's distribution is $P(t|m, c_l, c_r) = \text{softmax}_t(s_t)$.

Training Because our model involves substantial computation for each possible title, we want to limit the set of titles considered during training. For each example we consider, we construct a set T containing the gold title and 4 negative "distractor" titles from the candidate set. Unlike Eshel et al. (2017), we structure training as a multiclass decision among these titles rather than a binary prediction problem over each title as gold or not. We run our model over the candidates $t \in T$ to produce the distribution $P(t|m, c_l, c_r)$ and train to maximize the log probablity $\log P(t^*|m, c_l, c_r)$ of the gold title.

Results The model set forth in this section is the basis for the remaining models in this paper; we call it the GRU model as that is the only context encoding mechanism it uses. As shown in Table 1, this GRU model gets a score of 73.4 on the WikilinksNED development set. In the next section, we explore techniques for using the context in a more sophisticated way to improve further on this result.

Model	Accuracy on Test (%)
Eshel et al. (2017)	73.0
Eshel system release	72.2
GRU+ATTN	74.5
GRU+ATTN+FEATS	**75.8**
Model	**Accuracy on Dev (%)**
GRU	73.4
GRU+ATTN	74.4
GRU+ATTN+FEATS	**74.9**
GRU+ATTN+CNN	73.8

Table 1: Results on the WikilinksNED dev and test sets. Our model including features achieves state-of-the-art performance on the test set, compared to both the reported numbers from Eshel et al. (2017) as well as their released software. Incorporating character CNNs surprisingly leads to lower performance compared to these simple features.

3 Exploiting Context Cues

3.1 Attention

One way to improve over the basic GRU model is to use attention over the context based on the title under consideration. The attention we use is a modified version of the dot product attention (Luong et al., 2015) used by Eshel et al. (2017), allowing the model to weight the importance of the outputs of the GRU at each time step. Each context (left and right) has its own attention weights. For a given side of context and candidate t, the attention first computes a transformation of the entity embedding e_t as follows: $q_t = \tanh(We_t)$. This allows the model to learn an attention query q_t distinct from the candidate embedding e_t. The model then computes attention probabilities α_i for each GRU output o_i, normalized over the entire sequence of GRU outputs (of length n):

$$\alpha_i = \text{softmax}_i(q_t \cdot o_i)$$

The resulting probability distribution is used to take a weighted sum of GRU outputs to get a representation a:

$$a = \sum_i^n \alpha_i o_i$$

We compute a_l and a_r independently and symmetrically for the left and right context. These vectors are then fed forward through the model as the final continuous representation of the left or right context, l or r respectively.

Results In Table 1, we see that our model with attention (GRU+ATTN) outperforms our basic GRU model by around 1% absolute. It also outperforms the roughly similar model of Eshel et al. (2017) on the test set: this gain is due to a combination of factors including the improved training procedure and some small modeling changes.[2] However, our attention scheme is not without its shortcomings, as we now discuss.

3.2 Shortcomings of Attention

One common and frustrating error our model makes is failing to correctly disambiguate mentions whose contexts share similar words or character overlap with the gold entity's actual Wikipedia title. In these instances, the model fails to attend correctly to words that we, as human readers, would most likely see as disambiguating terms. For instance, in this example's left context:

> ...known also for the B.P. Koirala Institute of Health Sciences, one of the biggest government hospital. The indigenous **people** of Dharan are *Limbu* ...

the model fails to identify **people** as a critical term for disambiguation. This failure is partially due to the model's sole reliance on distributed representations: the embedding for **people** and the title embedding for *Limbu People* need to somehow contain enough common information for the model to associate these, identify **people** as an important token, and use it to disambiguate between candidate titles such as *Limbu People*, *Limbu Language*, and *Limbu Alphabet*. Moreover, with such noisy, unstructured context, it is difficult for the model to learn to rely on other grammatical or semantic cues (such as *are* indicating that the title is probably a plural noun, which *alphabet* and *language* are not).

3.3 Character CNNs

One way to address these issues in the model is to exploit more fine-grained character-level information. This circumvents the need to separately learn a distributed correspondence between terms with lexical overlap, and is especially useful when these terms may be unknown words; for example, a year mentioned within a context is often

---

[2] Note that in Eshel et al. (2017), the authors point out that their dataset has a high percentage of errors (35% of the errors made by their model are spurious), meaning that the skyline on this task is likely not higher than 90%.

unknown and therefore assigned an UNK embedding, even if that year matches exactly with a year in the gold candidate's title.

One solution to this is to allow our model to consult character-level information, which past models have done successfully for named entity recognition (Chiu and Nichols, 2015; Lample et al., 2016; Ma and Hovy, 2016), text classification (Zhang et al., 2015), and POS tagging (Santos and Zadrozny, 2014). We use convolutional neural networks (CNNs) to distill character representations of words into vectors that we concatenate with our word representations. We additionally use character CNNs over entity titles and concatenate these representations with the title embeddings e_t, to allow the model to learn to characterize similarities between contexts and entity titles. Our CNNs use window sizes of 6 and 100 filters each; these values were selected through hyperparameter tuning on the development set.

Table 1 shows the impact of incorporating character CNNs (GRU+ATTN+CNN). Surprisingly, these have a mild negative impact on performance. One possible explanation of this is that it causes the model to split its attention between semantically important and lexically similar context terms. Consider the following example:

> really think Final Fight could be a lot of a fun as a vigilante justice **movie** with a high quotient of hand-to-hand fight sequences. Think *The Warriors*

The gold title is *The Warrior (film)* and the base model correctly places 90% of its attention weight on the word **movie** when calculating attention for this title. However, the character-level CNN model only places 60% of its attention weight on it, distributing its attention values more evenly across the rest of the words. Such cases are frequent: the average highest weight given by attention in GRU+ATTN+CNN is about 6% lower than the average highest attention weight given by GRU+ATTN. The CNNs seem to have generally decreased the model's confidence in what context clues are key for disambiguation, leading to lower performance. We will return to more analysis of this in Section 4.

3.4 Lexical Feature Set

To determine whether character level overlap between the entity title and context is useful, we take a more direct approach to incorporating that information into our model and build a set of sparse features that directly target it.

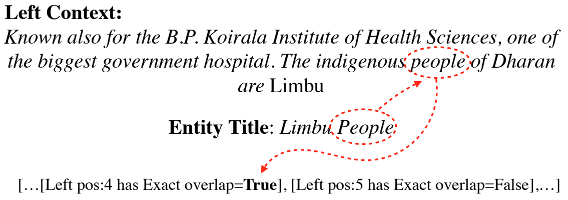

Left Context:
Known also for the B.P. Koirala Institute of Health Sciences, one of the biggest government hospital. The indigenous people of Dharan are Limbu

Entity Title: *Limbu People*

[…[Left pos:4 has Exact overlap=**True**], [Left pos:5 has Exact overlap=False],…]

Figure 2: An example of feature generation from an example. Here, because the word *people* occurs in the title and in the left context 4 words away from the mention, the indicator feature [Pos=4,Match=ExactWord,Context=Left] fires in the feature set.

Figure 2 shows an example of how our features are computed. We fire features on each word in the context that is either an exact match or a substring of a word in the candidate title; *people* is the relevant token here. We conjoin that match information with whether the word is in the left or right context along with the bucketed offset of the word from the mention. This feature set is then appended to the vector comparison features to form the input to the model's feedforward layer (see Figure 1).

Table 1 shows the results of stacking these features on top of our model with attention (GRU+ATTN+FEATS). We see our highest development set performance and correspondingly high test performance from this model. This indicates that character-level information is useful for disambiguation, but character CNNs as we incorporated them are not able to distill it as effectively as sparse features can. Our model augmented with these sparse features achieves state-of-the-art results on the test set.

4 Attention and "Obvious" Terms

Now that we have identified features which seem useful for this entity linking problem, we can ask how the tokens attended by our attention mechanism compare to those singled out by the features.

Table 2 contains statistics regarding the attention values of our GRU+ATTN and GRU+ATTN+CNN model on a subset of examples that both models got wrong. We define accuracy as the percentage of examples in which the model gives the highest attention to a word

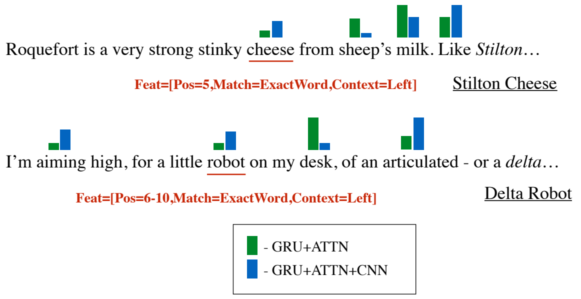

Figure 3: Examples of our models putting high attention weight into irrelevant context words, not acknowledging the relevance of disambiguating terms that share lexical overlap with the correct title. We display the weight given to the top 4 attended words above each word for two of our models.

that contains one of our lexical features, out of all examples where such a feature exists anywhere. The reported probability mass is the total attention mass that the model puts into words that associated with lexical features, averaged over all examples where such features exist. We see that the model frequently fails to exploit this information, and moreover the addition of CNNs does not strongly improve this.

Figure 3 shows examples of this behavior. In the first example, rather than identifying *cheese* as a salient term, both models instead focus more heavily on *milk* and *like*. Similarly, in the second example, the model fails to recognize the importance of *robot* in the context.

One possible reason that CNNs don't help more is that the sparse features only trigger on a subset of examples. Because the CNNs process every example, they may not see enough examples of lexical overlap to pick up on it, and instead try to augment what the word embedding model is already doing with subword information, which ends up being unstable for this task. Naturally, words with these overlap characteristics are not always the most disambiguating term. However, in light of noisy contexts, when the standard representation of context fails to be sufficient for allowing the model to disambiguate, we want the model to be able to leverage this character level information to help it make intuitive decisions, which the CNN fails to do.

Context	**Acc (%)**	**Prob Mass (%)**
GRU+ATTN L	0.41	0.32
GRU+ATTN R	0.36	0.30
GRU+ATTN+CNN L	0.46	0.32
GRU+ATTN+CNN R	0.36	0.28

Table 2: Our models' attention "accuracy": how often each model's maximally-attended word also triggered a feature to fire. Prob Mass indicates the average sum of attention scores over feature-triggering words. All values are computed over a sample of 10,000 examples that each model got wrong.

5 Conclusion

In this paper, we observed that in noisy entity linking settings on short texts, neural models relying on attention do not always pick up on the correct context clues, even when those clues exhibit very obvious surface overlap with the correct entity title. These models can perform better when augmented with sparse features explicitly targeting this kind of lexical overlap: our system using these features achieves state-of-the-art disambiguation accuracy on the WikilinksNED dataset. By contrast, automatically learning learning fine-grained character-level features with CNNs in this context is hard. More exploration is needed to better understand what inductive biases are necessary for an entity linking system to make maximally effective use of the information available to it.

Acknowledgments

This work was partially supported by NSF Grant IIS-1814522, a Bloomberg Data Science Grant, and an equipment grant from NVIDIA. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources used to conduct this research. Thanks as well to the anonymous reviewers for their helpful comments.

References

Luisa Bentivogli. 2010. Extending English ACE 2005 Corpus Annotation with Ground-truth Links to Wikipedia. In *Proceedings of COLING*.

Kevin K. Bowden, Jiaqi Wu, Shereen Oraby, Amita Misra, and Marilyn Walker. 2018. SlugNERDS: A Named Entity Recognition Tool for Open Domain Dialogue Systems. In *Proceedings of LREC*.

Xiao Cheng and Dan Roth. 2013. Relational Inference for Wikification. In *Proceedings of EMNLP*.

Jason P. C. Chiu and Eric Nichols. 2015. Named Entity Recognition with Bidirectional LSTM-CNNs. In *Proceedings of ACL*.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of EMNLP*.

Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of EMNLP-CoNLL*.

Mark Dredze, Nicholas Andrews, and Jay DeYoung. 2016. Twitter at the Grammys: A Social Media Corpus for Entity Linking and Disambiguation. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*.

Greg Durrett and Dan Klein. 2014. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. In *TACL*.

Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. 2017. Named Entity Disambiguation for Noisy Text. In *Proceedings of CoNLL*.

Yuan Fang and Ming-Wei Chang. 2014. Entity Linking on Microblogs with Spatial and Temporal Signals. In *TACL*.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing Semantic Similarity for Entity Linking with Convolutional Neural Networks. In *Proceedings of NAACL*.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To Link or Not to Link? A Study on End-to-End Tweet Entity Linking. In *Proceedings of NAACL*.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity Linking via Joint Encoding of Types, Descriptions, and Context. In *Proceedings of EMNLP*.

Hannaneh Hajishirzi, Leila Zilles, Daniel S. Weld, and Luke Zettlemoyer. 2013. Joint Coreference Resolution and Named-Entity Linking with Multi-Pass Sieves. In *Proceedings of EMNLP*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust Disambiguation of Named Entities in Text. In *Proceedings of EMNLP*.

Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji, and Chin-Yew Lin. 2014. Collective Tweet Wikification based on Semi-supervised Graph Regularization. In *Proceedings of ACL*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL*.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of ACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of ACL*.

David Milne and Ian H. Witten. 2008. Learning to Link with Wikipedia. In *Proceedings of CIKM*.

Seungwhan Moon, Leonardo Neves, and Vitor Carvalho. 2018. Multimodal Named Entity Disambiguation for Noisy Social Media Posts. In *Proceedings of the ACL*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proceedings of NAACL*.

Cicero Dos Santos and Bianca Zadrozny. 2014. Learning Character-level Representations for Part-of-Speech Tagging. In *Proceedings of ICML*.

Avirup Sil and Radu Florian. 2016. One for all: Towards language independent named entity linking. In *Proceedings of ACL*.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. In *Proceedings of AAAI*.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A Large-scale Cross-Document Coreference Corpus Labeled via Links to Wikipedia. Technical report.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation. In *Proceedings of IJCAI*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Proceedings of NIPS*.

Exploiting Contextual Information via Dynamic Memory Network for Event Detection

Shaobo Liu[†§], **Rui Cheng**[†§], **Xiaoming Yu**[†], **Xueqi Cheng**[†]
[†]CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Science (CAS);
[§]School of Computer and Control Engineering, The University of CAS
{liushaobo, chengrui}@software.ict.ac.cn;
{yuxiaoming, cxq}@ict.ac.cn

Abstract

The task of event detection involves identifying and categorizing event triggers. Contextual information has been shown effective on the task. However, existing methods which utilize contextual information only process the context once. We argue that the context can be better exploited by processing the context multiple times, allowing the model to perform complex reasoning and to generate better context representation, thus improving the overall performance. Meanwhile, dynamic memory network (DMN) has demonstrated promising capability in capturing contextual information and has been applied successfully to various tasks. In light of the multi-hop mechanism of the DMN to model the context, we propose the trigger detection dynamic memory network (TD-DMN) to tackle the event detection problem. We performed a five-fold cross-validation on the ACE-2005 dataset and experimental results show that the multi-hop mechanism does improve the performance and the proposed model achieves best F_1 score compared to the state-of-the-art methods.

1 Introduction

According to ACE (Automatic Content Extraction) event extraction program, an event is identified by a word or a phrase called event trigger which most represents that event. For example, in the sentence "No major *explosion* we are aware of", an event trigger detection model is able to identify the word "*explosion*" as the event trigger word and further categorize it as an *Attack* event. The ACE-2005 dataset also includes annotations for event arguments, which are a set of words or phrases that describe the event. However, in this work, we do not tackle the event argument classification and focus on event trigger detection.

The difficulty of the event trigger detection task lies in the complicated interaction between the event trigger candidate and its context. For instance, given a sentence at the end of a passage:

they are going over there to do a mission they believe in and as we said, 250 *left* yesterday.

It's hard to directly classify the trigger word "*left*" as an "*End-Position*" event or a "*Transport*" event because we are not certain about what the number "250" and the pronoun "they" are referring to. But if we see the sentence:

we are seeing these soldiers head out.

which is several sentences away from the former one, we now know the "250" and "they" refer to "the soldiers", and from the clue "these soldiers head out" we are more confident to classify the trigger word "*left*" as the "*Transport*" event.

From the above, we can see that the event trigger detection task involves complex reasoning across the given context. Exisiting methods (Liu et al., 2017; Chen et al., 2015; Li et al., 2013; Nguyen et al., 2016; Venugopal et al., 2014) mainly exploited sentence-level features and (Liao and Grishman, 2010; Zhao et al., 2018) proposed document-level models to utilize the context.

The methods mentioned above either not directly utilize the context or only process the context once while classifying an event trigger candidate. We argue that processing the context multiple times with later steps re-evaluating the context with information acquired from the previous steps improves the model performance. Such a mechanism allows the model to perform complicated reasoning across the context. As in the example, we are more confident to classify "*left*" as a "*Tranport*" event if we know "250" and "they" refer to "soldiers" in previous steps.

We utilize the dynamic memory network (DMN) (Xiong et al., 2016; Kumar et al., 2016) to capture the contextual information of the given trigger word. It contains four modules: the input module for encoding reference text where the an-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1030–1035
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

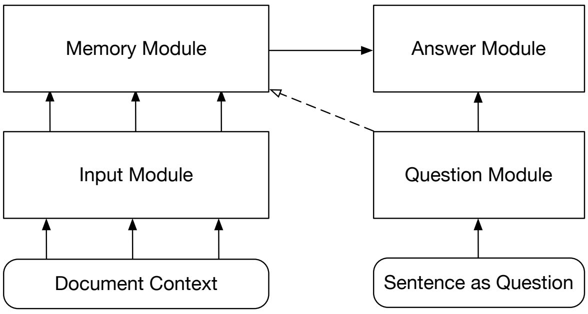

Figure 1: Overview of the TD-DMN architecture

swer clues reside, the memory module for storing knowledge acquired from previous steps, the question module for encoding the questions, and the answer module for generating answers given the output from memory and question modules.

DMN is proposed for the question answering task, however, the event trigger detection problem does not have an explicit question. The original DMN handles such case by initializing the question vector produced by the question module with a zero or a bias vector, while we argue that each sentence in the document could be deemed as a question. We propose the trigger detection dynamic memory network (TD-DMN) to incorporate this intuition, the question module of TD-DMN treats each sentence in the document as implicitly asking a question "What are the event types for the words in the sentence given the document context". The high-level architecture of the TD-DMN model is illustrated in Figure 1.

We compared our results with two models: DMCNN(Chen et al., 2015) and DEEB-RNN(Zhao et al., 2018) through 5-fold cross-validation on the ACE-2005 dataset. Our model achieves best F_1 score and experimental results further show that processing the context multiple times and adding implicit questions do improve the model performance. The code of our model is available online.[1]

2 The Proposed Approach

We model the event trigger detection task as a multi-class classification problem following existing work. In the rest of this section, we describe four different modules of the TD-DMN separately along with how data is propagated through these modules. For simplicity, we discuss a single document case. The detailed architecture of our model is illustrated in Figure 2.

2.1 Input Module

The input module further contains two layers: the sentence encoder layer and the fusion layer. The sentence encoder layer encodes each sentence into a vector independently, while the fusion layer gives these encoded vectors a chance to exchange information between sentences.

Sentence encoder layer Given document d with l sentences $(s_1, \dots, s_l)$, let s_i denotes the i-th sentence in d with n words $(w_{i1}, \dots, w_{in})$. For the j-th word w_{ij} in s_i, we concatenate its word embedding $\boldsymbol{w_{ij}}$ with its entity type embedding[2] $\boldsymbol{e_{ij}}$ to form the vector $\boldsymbol{W_{ij}}$ as the input to the sentence encoder Bi-GRU(Cho et al., 2014) of size H_s. We obtain the hidden state $\boldsymbol{h_{ij}}$ by merging the forward and backward hidden states from the Bi-GRU:

$$\boldsymbol{h_{ij}} = \overrightarrow{\mathrm{GRU}_s}(\boldsymbol{W_{ij}}) + \overleftarrow{\mathrm{GRU}_s}(\boldsymbol{W_{ij}}) \quad (1)$$

where + denotes element-wise addition.

We feed $\boldsymbol{h_{ij}}$ into a two-layer perceptron to generate the unnormalized attention scalar u_{ij}:

$$u_{ij} = \tanh(\boldsymbol{h_{ij}} \cdot W_{s_1}) \cdot W_{s_2} \quad (2)$$

where W_{s_1} and W_{s_2} are weight parameters of the perceptron and we omitted bias terms. u_{ij} is then normalized to obtain scalar attention weight α_{ij}:

$$\alpha_{ij} = \frac{\exp(u_{ij})}{\sum_{k=1}^{n} \exp(u_{ik})} \quad (3)$$

The sentence representation $\boldsymbol{s_i}$ is obtained by:

$$\boldsymbol{s_i} = \sum_{j=1}^{n} \alpha_{ij} \boldsymbol{h_{ij}} \quad (4)$$

Fusion layer The fusion layer processes the encoded sentences and outputs fact vectors which contain exchanged information among sentences. Let $\boldsymbol{s_i}$ denotes the i-th sentence representation obtained from the sentence encoder layer. We generate fact vector $\boldsymbol{f_i}$ by merging the forward and backward states from fusion GRU:

$$\boldsymbol{f_i} = \overrightarrow{\mathrm{GRU}_f}(\boldsymbol{s_i}) + \overleftarrow{\mathrm{GRU}_f}(\boldsymbol{s_i}) \quad (5)$$

Let H_f denotes the hidden size of the fusion GRU, we concatenate fact vectors $\boldsymbol{f_1}$ to $\boldsymbol{f_l}$ to obtain the matrix $\boldsymbol{F}$ of size l by H_f, where the i-th row in $\boldsymbol{F}$ stores the i-th fact vector $\boldsymbol{f_i}$.

---

[1] https://github.com/AveryLiu/TD-DMN

[2] The ACE-2005 includes entity type (including type "NA" for none-entity) annotations for each word, the entity type embedding is a vector associated with each entity type.

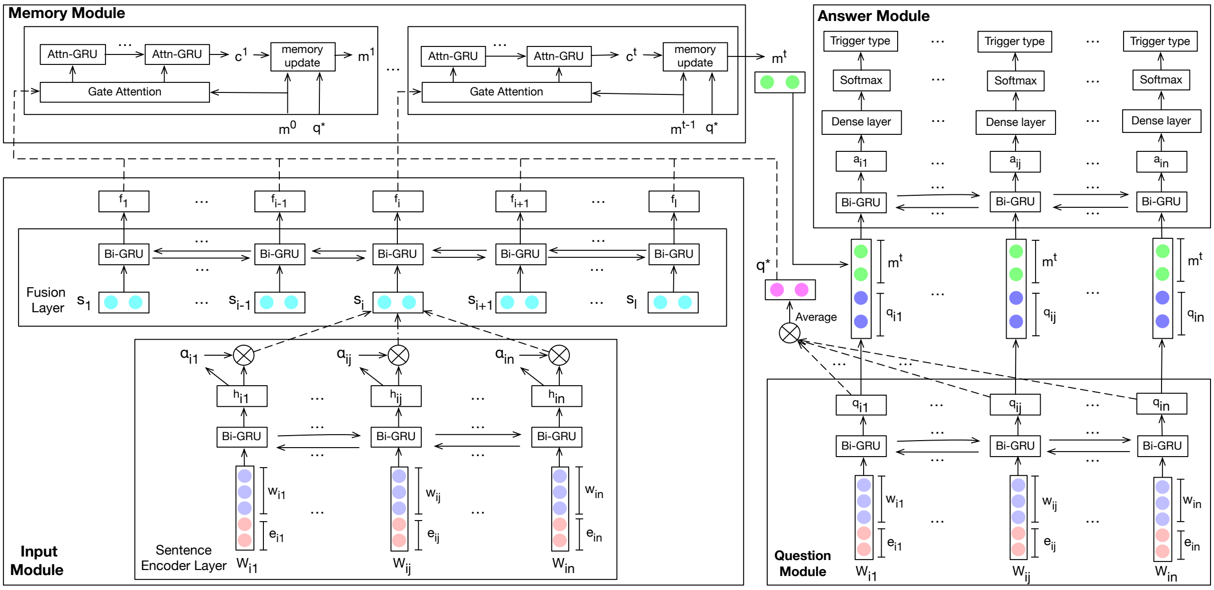

Figure 2: The detailed architecture of the TD-DMN model. The figure depicts a simplified case where a single document d with l sentences is the input to the input module and a sentence s_i of d with n words is the input to the question module. The input module encodes document d into a fact matrix $\{\boldsymbol{F}|\boldsymbol{f_1}, \ldots, \boldsymbol{f_l}\}$. The question module encodes sentence s_i into the question vector $\boldsymbol{q^*}$. The memory module initializes $\boldsymbol{m_0}$ with $\boldsymbol{q^*}$ and iteratively processes for t times, at each time k it produces a memory vector $\boldsymbol{m_k}$ using fact matrix $\boldsymbol{F}$, question vector $\boldsymbol{q^*}$ and previous memory state $\boldsymbol{m_{k-1}}$. The answer module outputs the predicted trigger type for each word in s_i using the concatenated tensor of the hidden states of the question module and the last memory state $\boldsymbol{m_t}$.

2.2 Question Module

The question module treats each sentence s in d as implicitly asking a question: What are the event types for each word in the sentence s given the document d as context? For simplicity, we only discuss the single sentence case. Iteratively processing from s_1 to s_l will give us all encoded questions in document d.

Let $\boldsymbol{W_{ij}}$ be the vector representation of j-th word in s_i, the question GRU generates hidden state $\boldsymbol{q_{ij}}$ by:

$$\boldsymbol{q_{ij}} = \overrightarrow{\mathrm{GRU}}_q(\boldsymbol{W_{ij}}) + \overleftarrow{\mathrm{GRU}}_q(\boldsymbol{W_{ij}}) \quad (6)$$

The question vector $\boldsymbol{q^*}$ is obtained by averaging all hidden states of the question GRU:

$$\boldsymbol{q^*} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{q_{ij}} \quad (7)$$

Let H_q denotes the hidden size of the question GRU, $\boldsymbol{q^*}$ is a vector of size H_q, the intuition here is to obtain a vector that represents the question sentence. $\boldsymbol{q^*}$ will be used for the memory module.

2.3 Memory Module

The memory module has three components: the attention gate, the attentional GRU(Xiong et al., 2016) and the memory update gate. The attention gate determines how much the memory module should attend to each fact given the facts $\boldsymbol{F}$, the question $\boldsymbol{q^*}$, and the acquired knowledge stored in the memory vector $\boldsymbol{m_{t-1}}$ from the previous step.

The three inputs are transformed by:

$$\boldsymbol{u} = [\boldsymbol{F} * \boldsymbol{q^*}; |\boldsymbol{F} - \boldsymbol{q^*}|; \boldsymbol{F} * \boldsymbol{m_{t-1}}; |\boldsymbol{F} - \boldsymbol{m_{t-1}}|] \quad (8)$$

where ; is concatenation. $*$, $-$ and $|.|$ are element-wise product, subtraction and absolute value respectively. $\boldsymbol{F}$ is a matrix of size (m, H_f), while $\boldsymbol{q^*}$ and $\boldsymbol{m_{t-1}}$ are vectors of size $(1, H_q)$ and $(1, H_m)$, where H_m is the output size of the memory update gate. To allow element-wise operation, H_f, H_q and H_m are set to a same value H. Meanwhile, $\boldsymbol{q^*}$ and $\boldsymbol{m_{t-1}}$ are broadcasted to the size of (m, H). In equation 8, the first two terms measure the similarity and difference between facts and the question. The last two terms have the same functionality for facts and the last memory state.

Let $\boldsymbol{\beta}$ of size l denotes the generated attention vector. The i-th element in $\boldsymbol{\beta}$ is the attention weight for fact $\boldsymbol{f_i}$. $\boldsymbol{\beta}$ is obtained by transforming $\boldsymbol{u}$ using a two-layer perceptron:

$$\boldsymbol{\beta} = \mathrm{softmax}(\tanh(\boldsymbol{u} \cdot W_{m_1}) \cdot W_{m_2}) \quad (9)$$

Methods	Fold 1			Fold 2			Fold 3			Fold 4			Fold 5			Avg
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	F_1
DMCNN	67.6	60.5	63.9	62.6	63.1	62.9	68.9	62.1	65.3	68.9	65.0	66.9	66.0	65.5	65.8	64.9
DEEB-RNN	64.9	64.1	64.5	63.4	64.7	**64.0**	66.1	64.3	65.2	66.0	67.3	66.6	65.5	67.2	66.3	65.3
TD-DMN 1-hop	67.3	62.1	64.6	65.4	61.7	63.5	72.0	60.0	65.4	66.6	68.0	**67.3**	68.3	65.0	66.6	65.5
TD-DMN 2-hop	69.2	61.0	64.8	64.6	63.4	**64.0**	64.3	66.4	65.3	68.7	65.9	**67.3**	68.5	65.7	**67.1**	65.7
TD-DMN 3-hop	66.3	63.7	64.9	66.9	60.6	63.6	68.3	64.0	66.1	67.9	66.3	67.1	70.2	64.3	**67.1**	**65.8**
TD-DMN 4-hop	66.7	63.4	**65.0**	61.4	65.5	63.4	66.4	66.0	**66.2**	64.7	69.1	66.8	70.0	63.4	66.5	65.6

Table 1: 5-fold cross-validation results on the ACE-2005 dataset. The results are rounded to a single digit. The F_1 of the last column are calculated by averaging F_1 scores of all folds.

where W_{m_1} and W_{m_2} are parameters of the perceptron and we omitted bias terms.

The attentional GRU takes facts $\boldsymbol{F}$, fact attention $\boldsymbol{\beta}$ as input and produces context vector $\boldsymbol{c}$ of size H_c. At each time step t, the attentional GRU picks the $\boldsymbol{f_t}$ as input and uses β_t as its update gate weight. For space limitation, we refer reader to (Xiong et al., 2016) for the detailed computation.

The memory update gate outputs the updated memory $\boldsymbol{m_t}$ using question $\boldsymbol{q^*}$, previous memory state $\boldsymbol{m_{t-1}}$ and context $\boldsymbol{c}$:

$$\boldsymbol{m_t} = \mathrm{relu}([\boldsymbol{q^*}; \boldsymbol{m_{t-1}}; \boldsymbol{c}] \cdot W_u) \qquad (10)$$

where W_u is the parameter of the linear layer.

The memory module could be iterated several times with a new $\boldsymbol{\beta}$ generated for each time. This allows the model to attend to different parts of the facts in different iterations, which enables the model to perform complicated reasoning across sentences. The memory module produces $\boldsymbol{m_t}$ as the output at the last iteration.

2.4 Answer Module

Answer module predicts the event type for each word in a sentence. For each question GRU hidden state $\boldsymbol{q_{ij}}$, the answer module concatenates it with the memory vector $\boldsymbol{m_t}$ as the input to the answer GRU with hidden size H_a. The answer GRU outputs $\boldsymbol{a_{ij}}$ by merging its forward and backward hidden states. The fully connected dense layer then transforms $\boldsymbol{a_{ij}}$ to the size of the number of event labels O and the softmax layer is applied to output the probability vector $\boldsymbol{p_{ij}}$. The k-th element in $\boldsymbol{p_{ij}}$ is the probability for the word w_{ij} being the k-th event type. Let y_{ij} be the true event type label for word w_{ij}. Assuming all sentences are padded to the same length n, the cross-entropy loss for the single document d is applied as:

$$J(\boldsymbol{y}, \boldsymbol{p}) = -\sum_{i=1}^{l}\sum_{j=1}^{n}\sum_{k=1}^{O} \mathrm{I}(y_{ij} = k) \log p_{ij}^{(k)} \qquad (11)$$

where $I(\cdot)$ is the indicator function.

3 Experiments

3.1 Dataset and Experimental Setup

Dateset

Different from prior work, we performed a 5-fold cross-validation on the ACE 2005 dataset. We partitioned 599 files into 5 parts. The file names of each fold can be found online[3]. We chose a different fold each time as the testing set and used the remaining four folds as the training set.

Baselines

We compared our model with two other models: DMCNN (Chen et al., 2015) and DEEB-RNN (Zhao et al., 2018). DMCNN is a sentence-level event detection model which enhances traditional convolutional networks with dynamic multiple pooling mechanism customized for the task. The DEEB-RNN is a state-of-the-art document-level event detection model which firstly generate a document embedding and then use it to aid the event detection task.

Evaluation

We report precision, recall and F_1 score of each fold along with the averaged F_1 score of all folds. We evaluated all the candidate trigger words in each testing set. A candidate trigger word is correctly classified if its event subtype and offsets match its human annotated label.

Implementation Details

To avoid overfitting, we fixed the word embedding and added a 1 by 1 convolution after the embedding layer to serve as a way of fine tuning but with a much smaller number of parameters. We removed punctuations, stop words and sentences which have length less equal than 2. We used the Stanford corenlp toolkit(Manning et al., 2014) to

---

[3] https://github.com/AveryLiu/TD-DMN/data/splits

separate sentences. We down-sampled negative samples to ease the unbalanced classes problem.

The setting of the hyperparameters is the same for different hops of the TD-DMN model. We set H, H_s, H_c, and H_a to 300, the entity type embedding size to 50, W_{s_1} to 300 by 600, W_{s_2} to 600 by 1, W_{m_1} to 1200 by 600, W_{m_2} to 600 by 1, W_u to 900 by 300, the batch size[4] to 10. We set the down-sampling ratio to 9.5 and we used Adam optimizer (Kingma and Ba, 2014) with weight decay set to $1e^{-5}$. We set the dropout(Srivastava et al., 2014) rate before the answer GRU to 0.4 and we set all other dropout rates to 0.2. We used the pre-trained word embedding from (Le and Mikolov, 2014).

3.2 Results on the ACE 2005 Corpus

The performance of each model is listed in table 1. The first observation is that models using document context drastically outperform the model that only focuses on the sentence level feature, which indicates document context is helpful in event detection task. The second observation is that increasing number of hops improves model performance, this further implies that processing the context for multiple times does better exploit the context. The model may have exhausted the context and started to overfit, causing the performance drop at the fourth hop.

The performance of reference models is much lower than that reported in their original papers. Possible reasons are that we partitioned the dataset randomly, while the testing set of the original partition mainly contains similar types of documents and we performed a five-fold cross-validation.

3.3 The Impact of the Question Module

To reveal the effects of the question module, we ran the model in two different settings. In the first setting, we initialized the memory vector $\boldsymbol{m_0}$ and question vector $\boldsymbol{q^*}$ with a zero vector, while in the second setting, we ran the model untouched. The results are listed in the table 2. The two models perform comparably under the 1-hop setting, this implies that the model is unable to distinguish the initialization values of the question vector well in the 1-hop setting. For higher number of hops, the untouched model outperforms the modified one. This indicates that with a higher number of memory iterations, the question vector $\boldsymbol{q^*}$ helps the model to better exploit the context information.

Methods	F_1	F_1^*
TD-DMN 1-hop	65.48	65.52
TD-DMN 2-hop	65.69	65.46
TD-DMN 3-hop	**65.78**	65.51
TD-DMN 4-hop	65.57	65.40

Table 2: The impact of the question module, F_1^* indicates results with empty questions.

We still observe the increase and drop pattern of the F_1 for the untouched model. However, such a pattern is not obvious with empty questions. This implies that we are unable to have a steady gain without the question module in this specific task.

4 Future Work

In this work, we explored the TD-DMN architecture to exploit document context. Extending the model to include wider contexts across several similar documents may also be of interest. The detected event trigger information can be incorporated into question module when extending the TD-DMN to the argument classification problem. Other tasks with document context but without explicit questions may also benefit from this work.

5 Conclusion

In this paper, we proposed the TD-DMN model which utilizes the multi-hop mechanism of the dynamic memory network to better capture the contextual information for the event trigger detection task. We cast the event trigger detection as a question answering problem. We carried five-fold cross-validation experiments on the ACE-2005 dataset and results show that such multi-hop mechanism does improve the model performance and we achieved the best F_1 score compared to the state-of-the-art models.

Acknowledgments

We thank Prof. Huawei Shen for providing mentorship in the rebuttal phase. We thank Jinhua Gao for discussion on the paper presentation. We thank Yixing Fan and Xiaomin Zhuang for providing advice regarding hyper-parameter tuning. We thank Yue Zhao for the initial discussion on event extraction. We thank Yaojie Lu for providing preprocessing scripts and the results of DMCNN. We thank anonymous reviewers for their advice. The first author personally thank Wei Qi for being supportive when he was about to give up.

---

[4] In each batch, there are 10 documents.

References

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 167–176.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1789–1798.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 831–843.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*, pages 2397–2406.

Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2018. Document embedding enhanced event detection with hierarchical and supervised attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 414–419.

Do explanations make VQA models more predictable to a human?

Arjun Chandrasekaran[*,1] **Viraj Prabhu**[*,1] **Deshraj Yadav** [*,1]
Prithvijit Chattopadhyay[*,1] **Devi Parikh**[1,2]
[1]Georgia Institute of Technology [2]Facebook AI Research
`{carjun, virajp, deshraj, prithvijit3, parikh}@gatech.edu`

Abstract

A rich line of research attempts to make deep neural networks more transparent by generating human-interpretable ‘explanations’ of their decision process, especially for interactive tasks like Visual Question Answering (VQA). In this work, we analyze if existing explanations indeed make a VQA model – its responses as well as failures – more predictable to a human. Surprisingly, we find that they do not. On the other hand, we find that human-in-the-loop approaches that treat the model as a black-box do.

1 Introduction

As technology progresses, we are increasingly collaborating with AI agents in *interactive* scenarios where humans and AI work together as a team, e.g., in AI-assisted diagnosis, autonomous driving, etc. Thus far, AI research has typically only focused on the AI in such an interaction – for it to be more accurate, be more human-like, understand our intentions, beliefs, contexts, and mental states.

In this work, we argue that for human-AI interactions to be more effective, humans must also understand the AI’s beliefs, knowledge, and quirks.

Many recent works generate human-interpretable ‘explanations’ regarding a model’s decisions. These are usually evaluated offline based on whether human judges found them to be ‘good’ or to improve trust in the model. However, their contribution in an interactive setting remains unclear. In this work, we evaluate the role of explanations towards making a model predictable to a human.

We consider an AI trained to perform the multi-modal task of Visual Question Answering (VQA) (Malinowski and Fritz, 2014; Antol et al., 2015), i.e., answering free-form natural language questions about images. VQA is applicable to scenarios where humans actively elicit information from visual data, and naturally lends itself to human-AI interactions. We consider two tasks that demonstrate the degree to which a human understands their AI teammate (we call Vicki) – Failure Prediction (FP) and Knowledge Prediction (KP). In FP, we ask subjects on Amazon Mechanical Turk to predict if Vicki will correctly answer a given question about an image. In KP, subjects predict Vicki’s exact response.

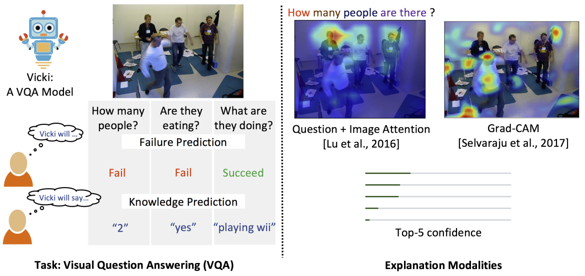

Figure 1: We evaluate the extent to which explanation modalities (right) and familiarization with a VQA model help humans predict its behavior – its responses, successes, and failures (left).

We aid humans in forming a mental model of Vicki by (1) familiarizing them with its behavior in a ‘training’ phase and (2) exposing them to its internal states via various explanation modalities. We then measure their FP and KP performance.

Our key findings are that (1) humans are indeed capable of predicting successes, failures, and outputs of the VQA model better than chance, (2) explicitly training humans to familiarize themselves with the model improves their performance, and (3) existing explanation modalities do not enhance human performance.

2 Related Work

Explanations in deep neural networks. Several works generate explanations based on inter-

---

[*] Denotes equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1036–1042
Brussels, Belgium, October 31 - November 4, 2018.

nal states of a decision process (Zeiler and Fergus, 2014; Goyal et al., 2016b), while others generate justifications that are consistent with model outputs (Ribeiro et al., 2016; Hendricks et al., 2016). Another popular form of providing explanations is to visualize regions in the input that contribute to a decision – either by explicitly attending to relevant input regions (Bahdanau et al., 2014; Xu et al., 2015), or exposing implicit attention for predictions (Selvaraju et al., 2017; Zhou et al., 2016).

Evaluating explanations. Several works evaluate the role of explanations in developing trust with users (Cosley et al., 2003; Ribeiro et al., 2016) or helping them achieve an end goal (Narayanan et al., 2018; Kulesza et al., 2012). Our work, however, investigates the role of machine-generated explanations in improving the *predictability* of a VQA model.

Failure prediction. While Bansal et al. (2014) and Zhang et al. (2014) predict failures of a model using simpler statistical models, we explicitly train a person to do this.

Legibility. Dragan et al. (2013) describe the intent-expressiveness of a robot as its trajectory being expressive of its goal. Analogously, we evaluate if explanations of the intermediate states of a VQA model are expressive of its output.

Humans adapting to technology. Wang et al. (2016) and Pelikan and Broth (2016) observe humans' strategies while adapting to the limited capabilities of an AI in interactive language games. In our work we explicitly measure to what extent humans can form an accurate model of an AI, and the role of familiarization and explanations.

3 Setup

Agent. We use the VQA model by Lu et al. (2016) as our AI agent (that we call Vicki). The model processes the question at multiple levels of granularity (words, phrases, entire question) and at each level, has explicit attention mechanisms on both the image and the question[1]. It is trained on the train split of the VQA-1.0 dataset (Antol et al., 2015). Given an image and a question about the image, it outputs a probability distribution over 1000 answers. Importantly, the model's image and question attention maps provide access to its 'internal states' while making a prediction.

Vicky is *quirky* at times, i.e., has biases, albeit in a predictable way. Agrawal et al. (2016) outlines several such quirks. For instance, Vicki has a limited capability to understand the image – when asked the color of a small object in the scene, say a soda can, it may simply respond with the most dominant color in the scene. Indeed, it may answer similarly even if no soda can is present, i.e. if the question is irrelevant.

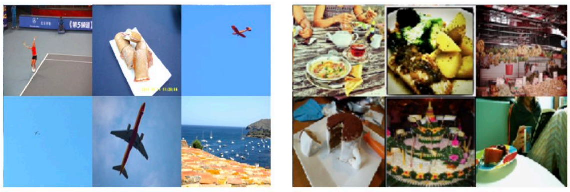

What color is the grass? Blue

What are the people doing? Eating

How many people are there? 4

What is the man holding? Fire Hydrant

Figure 2: These montages highlight some of Vicki's quirks. For a given question, Vicki has the same response to each image in a montage. Common visual patterns (that Vicki presumably picks up on) within each montage are evident.

Further, Vicki has a limited capability to understand free-form natural language, and in many cases, answers questions based only on the first few words of the question. It is also generally poor at answering questions requiring "common sense" reasoning. Moreover, being a discriminative model, Vicki has a limited vocabulary (1k) of answers. Additionally, the VQA 1.0 dataset contains label biases; therefore, the model is very likely to answer "white' to a "what color" question (Goyal et al., 2016a).

To get a sense for this, see Fig. 2 which depicts a clear pattern. In top-left, even when there is no grass, Vicki tends to latch on to one of the dominant colors in the image. For top-right, even when there are no people in the image, it seems to respond with what people could *plausibly* do in the scene if they were present. In this work, we measure to what extent lay people can pick up on these quirks by interacting with the agent, and whether existing explanation modalities help do so.

Tasks: Failure Prediction (FP). Given an image and a question about the image, we measure how well a person can predict if Vicki will successfully answer the question. A person can presumably predict the failure modes of Vicki well if they have a good sense of its strengths and weaknesses.

---

[1] We use question-level attention maps in our experiments.

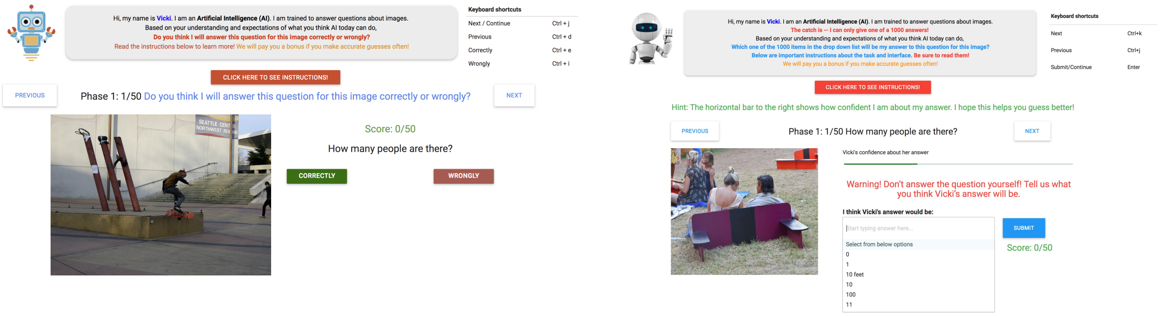
Hi, my name is Vicki. I am an **Artificial Intelligence (AI)**. I am trained to answer questions about images.
Based on your understanding and expectations of what you think AI today can do,
Do you think I will answer this question for this image correctly or wrongly?
Read the instructions below to learn more! We will pay you a bonus if you make accurate guesses often!

Phase 1: 1/50 Do you think I will answer this question for this image correctly or wrongly?

Score: 0/50

How many people are there?

(a) The Failure Prediction (FP) interface.

Hi, my name is Vicki. I am an **Artificial Intelligence (AI)**. I am trained to answer questions about images.
The catch is -- I can only give one of a 1000 answers!
Based on your understanding and expectations of what you think AI today can do,
Which one of the 1000 items in the drop down list will be my answer to this question for this image?
Below are important instructions about the task and interface. Be sure to read them!
We will pay you a bonus if you make accurate guesses often!

Hint: The horizontal bar to the right shows how confident I am about my answer. I hope this helps you guess better!

Phase 1: 1/50 How many people are there?

Vicki's confidence about her answer

Warning! Don't answer the question yourself! Tell us what you think Vicki's answer will be.

I think Vicki's answer would be:

Score: 0/50

(b) The Knowledge Prediction (KP) interface.

Figure 3: (a) A person guesses if a VQA model (Vicki) will answer this question for this image correctly or wrongly. (b) A person guesses what Vicki's exact answer will be for this QI–pair.

Knowledge Prediction (KP). In this task, we aim to obtain a fine-grained measure of a person's understanding of Vicki's behavior. Given a QI–pair, a subject guesses Vicki's exact response from a set of its output labels. Snapshots of our interfaces can be seen in Fig. 3.

4 Experimental Setup

In this section we investigate ways to make Vicki's behavior more predictable to a subject. We approach this by – providing instant feedback about Vicki's actual behavior on each QI pair once the subject responds, and exposing subjects to various explanation modalities that reveal Vicki's internal states before they respond.

Data. We identify a subset of questions in the VQA-1.0 (Antol et al., 2015) validation split that occur more than 100 times. We select 7 diverse questions[2] from this subset that are representative of the different types of questions (counting, yes/no, color, scene layout, activity, etc.) in the dataset. For each of the 7 questions, we sample a set of 100 images. For FP, the 100 images are random samples from the set of images on which the question was asked in VQA-1.0 val. For the KP task, these 100 images are random images from VQA-1.0 val. Ray et al. (2016) found that randomly pairing an image with a question in the VQA-1.0 dataset results in about 79% of pairs being irrelevant. This combination of relevant and irrelevant QI-pairs allows us to test subjects' ability to develop a robust understanding of Vicki's behavior across a wide variety of inputs.

Study setup. We conduct our studies on Amazon Mechanical Turk. Each task (HIT) comprises of 100 QI-pairs where for simplicity (for the subject), a single question is asked across all 100 images. The annotation task is broken down into a train and test phase of 50 QI-pairs each. Over all settings, 280 workers took part in our study (1 unique worker per HIT), resulting in 28k human responses. Subjects were paid an average of \$3 base plus \$0.44 performance bonus, per HIT.

There are some challenges involved in scaling data-collection in this setting: (1) Due to the presence of separate train and test phases, our AMT tasks tend to be unusually long (mean HIT durations across the tasks of FP and KP $= 10.11 \pm 1.09$ and 24.49 ± 1.85 min., respectively). Crucially, this also reduces the subject pool to only those willing to participate in long tasks. (2) Once a subject participates in a task, they cannot do another because their familiarity with Vicki would leak over. This constraint causes our analyses to require as many subjects as tasks. Since work division in crowdsourcing tasks follows a Pareto principle (Little, 2009), this makes data collection very slow.

In light of these challenges, we focus on a small set of questions to systematically evaluate the role of training and exposure to Vicki's internal states.

4.1 Evaluating the role of familiarization

To familiarize subjects with Vicki, we provide them with instant feedback during the train phase. Immediately after a subject responds to a QI–pair, we show them whether Vicki actually answered the question correctly or not (in FP) or what Vicki's response was (in KP), along with a running score of how well they are doing. Once training is complete, no further feedback is provided and subjects are asked to make predictions for the test phase. At the end, they are shown their

---

[2] What kind of animal is this? What time is it? What are the people doing? Is it raining? What room is this? How many people are there? What color is the umbrella?

score and paid a bonus proportional to the score.

Failure Prediction. In FP, always guessing that Vicki answers 'correctly' results in 58.29% accuracy, while subjects do slightly better and achieve 62.66% accuracy, even without prior familiarity with Vicki (No Instant Feedback (IF)). Further, we find that subjects that receive training via instant feedback (IF) achieve 13.09% higher mean accuracies than those who do not (see Fig 4; IF vs No IF for FP (left)).

Knowledge Prediction. In KP, answering each question with Vicki's most popular answer overall ('no') would lead to an accuracy of 13.4%. Additionally, answering each question with its most popular answer *for that question* leads to an accuracy of 31.43%. Interestingly, subjects who are unfamiliar with Vicki (No IF) achieve 21.27% accuracy – better than the most popular answer overall, but worse than the question-specific prior over its answers. The latter is understandable as subjects unfamiliar with Vicki do not know which of its 1000 possible answers the model is most likely to predict for each question.

We find that mean performance in KP with IF is 51.11%, 29.84% higher than KP without IF (see Fig 4; IF vs No IF for KP (right)). It is apparent that just from a few (50) training examples, subjects succeed in building a mental model of Vicki's behavior that generalizes to new images. Additionally, the 29.84% improvement over No IF for KP is significantly larger than that for FP (13.09%). This is understandable because a priori (No IF), KP is a much harder task as compared to FP due to the increased space of possible subject responses given a QI-pair, and the combination of relevant and irrelevant QI-pairs in the test phase.

Questions such as 'Is it raining?' have strong language priors – to these Vicki often defaults to the most popular answer ('no'), irrespective of image. On such questions, subjects perform considerably better in KP once they develop a sense for Vicki's inherent biases via instant feedback. For open-ended questions like 'What time is it?', feedback helps subjects (1) narrow down the 1000 potential options to the subset that Vicki typically answers with – in this case time periods such as 'daytime' rather than actual clock times and (2) identify correlations between visual patterns and Vicki's answer. In other cases like 'How many people are in the image?' the space of possible answers is clear a priori, but after IF subjects realize that Vicki is bad at detailed counting and bases its predictions on coarse signals of the scene layout.

4.2 Evaluating the role of explanations

In this setting, we show subjects an image, a question, and one of the explanation modalities described below. We experiment with 3 qualitatively different modalities (see Fig.1, right):

Confidence of top-5 predictions. We show subjects Vicki's confidence in its top-5 answer predictions from its vocabulary as a bar plot (of course, we do not show the actual top-5 predictions). **Attention maps.** Along with the image we show subjects the spatial attention map over the image and words of the question which indicate the regions that Vicki is looking at and listening to, respectively. **Grad-CAM.** We use the CNN visualization technique by Selvaraju et al. (2017), using the (implicit) attention maps corresponding to Vicki's most confident answer.

Automatic approaches. We also evaluate automatic approaches to detect Vicki's failure from its internal states. We find that both, a decision stump on Vicki's confidence in its top answer, and on the entropy of its softmax output, result in an FP accuracy of 60% on our test set. A Multi-layer Perceptron (MLP) trained on Vicki's output 1000-way softmax to predict success vs failure, achieves an FP accuracy of 81%. Training on just top-5 softmax outputs achieves an FP accuracy of 61.43%. Training an MLP which takes as input question features (average word2vec embeddings (Mikolov et al., 2013) of words in the question) concatenated with image features (fc7 from VGG-19) to predict success vs failure (which we call ALERT following (Zhang et al., 2014)) achieves an FP accuracy of 65%. Training an MLP on identical question features as above but concatenated with Grad-CAM saliency maps leads to FP accuracy of 73.14%. [3] Note that we only report machine results to put human accuracies in perspective. We do not draw any inferences about the relative capabilities of both.

Results. Average performance of subjects in the test phases of FP and KP, for different experimental settings are summarized in Fig. 4. In the first setting, we show subjects an explanation modality with instant feedback (IF+Explanation). For reference, also see performance of subjects provided with IF and no explanation modality (IF).

---

[3] These methods are trained on 66% of VQA-1.0 val. The remaining data is used for validation.

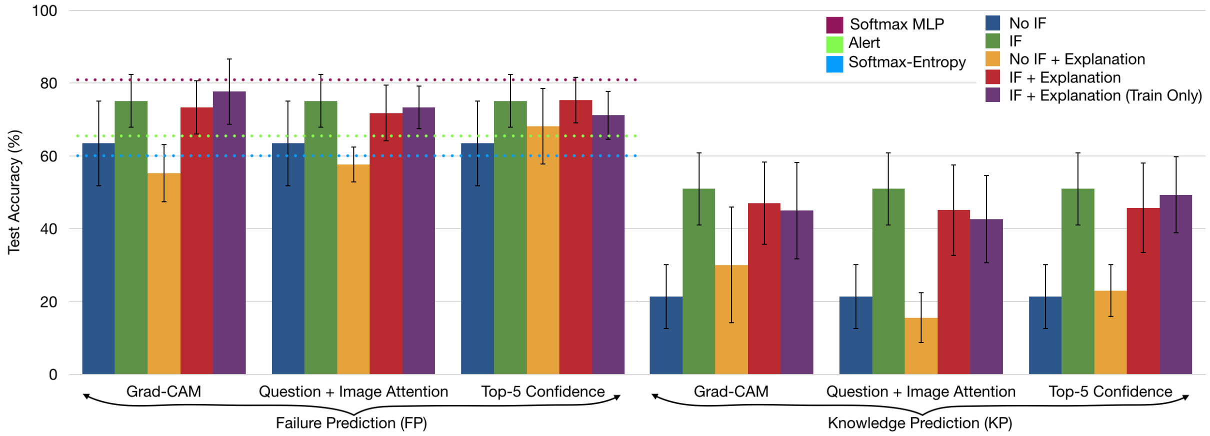

Figure 4: Average performance across subjects for Failure Prediction and Knowledge Prediction, across different settings: with or without (1) Instant feedback (IF) in the train phase, and (2) an explanation modality. Explanation modalities are shown in both train and test phases unless stated otherwise. Error bars are 95% confidence intervals from 1000 bootstrap samples. Note that the dotted lines are various machine approaches applied to FP.

We observe that on both FP and KP, subjects who received an explanation along with IF show no statistically significant difference in performance compared to those who did not. We see in Fig. 4, that both bootstrap based standard error (95% confidence intervals) overlap significantly.

Seeing that explanations in addition to IF does not outperform an IF baseline, we next measure whether explanations help a user not already familiar with Vicki via IF. That is, we evaluate if explanations help against a No IF baseline by providing an explanation only in the *test* phase, and no IF (see Fig 4; No IF + Explanation). Additionally, we also experiment with providing IF and an explanation *only* during the train phase (see Fig 4; IF + Explanation (Train Only)), to measure whether access to internal states during training can help subjects build better intuitions for model behavior without needing access to internal states at test time. In both settings however, we observe no statistically significant difference in performance over the No IF and IF baselines, respectively. [4]

5 Conclusion

As technology progresses, human-AI teams are inevitable. We argue that for these teams to be more effective, we should also be pursuing research directions to help humans understand the strengths, weaknesses, quirks, and tendencies of AI. We instantiate these ideas in the domain of Visual Question Answering (VQA), by proposing two tasks that help measure how well a human 'understands' a VQA model (we call Vicki) – Failure Prediction (FP) and Knowledge Prediction (KP). We find that lay people indeed get better at predicting Vicki's behavior using just a few 'training' examples, but surprisingly, existing popular explanation modalities do not help make its failures or responses more predictable. While previous works have typically assessed their interpretability or their role in improving human trust, our preliminary hypothesis is that these modalities may not yet help performance of human-AI teams in a goal-driven setting. Clearly, much work remains to be done in developing improved explanation modalities that can improve human-AI teams.

Future work involves closing the loop and evaluating the extent to which improved human performance at FP and KP translates to improved success of human-AI teams at accomplishing a shared goal. Co-operative human-AI games may be a natural fit for such an evaluation.

Acknowledgements. We thank Satwik Kottur for his help with data analysis, and for many fruitful discussions. We thank Aishwarya Agrawal and Akrit Mohapatra for their help with experiments. We would also like to acknowledge the workers on Amazon Mechanical Turk for their effort. This work was supported in part by NSF, AFRL, DARPA, Siemens, Google, Amazon, ONR YIP and ONR Grants N00014-16-1-2713.

---

[4] When piloting the tasks ourselves, we found it easy to 'overfit' to the explanations and hallucinate patterns.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Aayush Bansal, Ali Farhadi, and Devi Parikh. 2014. Towards transparent systems: Semantic characterization of failure modes. In *European Conference on Computer Vision*, pages 366–381. Springer.

Dan Cosley, Shyong K Lam, Istvan Albert, Joseph A Konstan, and John Riedl. 2003. Is seeing believing?: how recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 585–592. ACM.

Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. 2013. Legibility and predictability of robot motion. In *Human-Robot Interaction (HRI), 2013 8th ACM/IEEE International Conference on*, pages 301–308. IEEE.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016a. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *arXiv preprint arXiv:1612.00837*.

Yash Goyal, Akrit Mohapatra, Devi Parikh, and Dhruv Batra. 2016b. Towards transparent ai systems: Interpreting visual question answering models. *arXiv preprint arXiv:1608.08974*.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer.

Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. 2012. Tell me more?: the effects of mental model soundness on personalizing an intelligent agent. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1–10. ACM.

Greg Little. 2009. How many turkers are there.(dec 2009).

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*, pages 1682–1690.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Menaka Narayanan, Emily Chen, Jeffrey He, Been Kim, Sam Gershman, and Finale Doshi-Velez. 2018. How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1802.00682*.

Hannah RM Pelikan and Mathias Broth. 2016. Why that nao?: How humans adapt to a conventional humanoid robot in taking turns-at-talk. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4921–4932. ACM.

Arijit Ray, Gordon Christie, Mohit Bansal, Dhruv Batra, and Devi Parikh. 2016. Question relevance in vqa: Identifying non-visual and false-premise questions. *arXiv preprint arXiv:1606.06622*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*.

Sida I Wang, Percy Liang, and Christopher D Manning. 2016. Learning language games through interaction. *arXiv preprint arXiv:1606.02447*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Peng Zhang, Jiuling Wang, Ali Farhadi, Martial Hebert, and Devi Parikh. 2014. Predicting failures of vision systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3573.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929.

Facts That Matter

Marco Ponza[§], **Luciano Del Corro**[†], **Gerhard Weikum**[†]
[§]Department of Computer Science, University of Pisa, Italy
[†]Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
`marco.ponza@di.unipi.it`, `{corrogg,weikum}@mpi-inf.mpg.de`

Abstract

This work introduces fact salience: The task of generating a machine-readable representation of the most prominent information in a text document as a set of facts. We also present SALIE, the first fact salience system. SALIE is unsupervised and knowledge agnostic, based on open information extraction to detect facts in natural language text, PageRank to determine their relevance, and clustering to promote diversity. We compare SALIE with several baselines (including positional, standard for saliency tasks), and in an extrinsic evaluation, with state-of-the-art automatic text summarizers. SALIE outperforms baselines and text summarizers showing that facts are an effective way to compress information.

1 Introduction

Automatic knowledge acquisition at large scale requires the transformation of human-readable knowledge into a machine-understandable format. Machine-readable information is usually structured in the form of facts, in which a given relation links a set of arguments [e.g., *("US", "withdraws from", "Iran nuclear deal")*]. Facts are at the core of several natural language understanding applications such as knowledge-base (KB) construction (Nguyen et al., 2017), question answering (Abujabal et al., 2018), structured search (Bast et al., 2014), or entity-linking (Cheng and Roth, 2013).

Different approaches aim to discover facts from natural language text. In the extremes of the spectrum, relation extraction (Mintz et al., 2009) looks for all facts linkable to a KB, whereas open information extraction (Banko et al., 2007) extracts facts over an unconstrained set of arguments and relations. In this paper, we aim to additionally score facts according to their prominence.

We define *fact salience* as the task of discovering the most prominent facts in a text document. A fact is salient if it carries the essential information that the text conveys. A higher salient score denotes higher prominence, determining a ranking across all facts in the document. This ranking must reflect relevance and diversity: We want the top-k facts to compress the most relevant information in the smallest number of facts.

Fact salience is closely related to automatic text summarization (Erkan and Radev, 2004) as both try to capture the essential information in a document. However, fact salience output is required to be interpreted by machines to a certain extent. Text summarization, on the contrary, is meant to be understood by humans alone; it is often composed by ungrammatical text and isolated keywords, making it difficult to structure in a machine-readable form ex-post.

Here we present SALIE (Salient Information Extraction), the first fact salience system able to output a ranking of salient open facts from a text document. SALIE is unsupervised and knowledge agnostic. It uses facts as atomic units and PageRank to detect their relevance. It also exploits the fact structure to promote diversity via clustering.

We evaluated SALIE on a real-world dataset and compared it with the strong positional baseline (facts appearing first are more relevant) and, in an extrinsic evaluation, with two top text summarizers (one reimplemented to work at fact level). SALIE outperforms baselines and text summarization competitors particularly when the size of the output is restricted, suggesting that facts, as atomic units expressing a single proposition (Del Corro and Gemulla, 2013), are an effective way to compress information.

The source code and the processed datasets are

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1043–1048
Brussels, Belgium, October 31 - November 4, 2018.

publicly available[1] to encourage further developments of the fact salience task.

2 Fact Salience

Fact salience is the task of extracting *salient* facts from a text document. *Salient* facts must fulfil two requirements: (i) *relevance* and (ii) *diversity*.

A fact is relevant if it carries the essential information that the text conveys. A fact is not relevant per se but in a specific context. In an article about the US-Iran nuclear deal the fact *("US", "withdraws from", "Iran Nuclear Deal")* is more relevant that *("Washington", "is", "US capital")*.

The output of a fact salience system must ensure that the top-k facts contain the maximum information in the smallest number of facts. This implies a dependency between facts as less relevant facts should be penalized when they carry information already contained in more relevant ones.

3 Related Work

Fact salience is close to automatic text summarization (Erkan and Radev, 2004); both must detect the most prominent information in the text. However, while text summarization generates summaries for humans, fact salience output must be interpretable by machines. Fluency and language cohesion are not requirements for fact salience.

Triple scoring in KBs (Bast et al., 2017) is also related. However, while in fact salience a fact is not relevant per se but locally in a textual context, triple KB scoring asses the global relevance of a KB fact for a specific entity [*("T. Burton", "profession", "actor")* vs. *("T. Burton", "profession", "director")*] .

Typically, a text summarization system splits the text into atomic units (usually sentences) that are scored and ranked (Allahyari et al., 2017). Diversity is generally guaranteed by clustering them in topics and selecting the most representative members from each cluster. Once selected, the atomic units are compressed to ensure minimality.

Generating a machine-readable representation from text summarization output is difficult. This output can be incomplete or ungrammatical, given the use of compression techniques (Zajic et al., 2007), or the inclusion of keywords or short unconnected phrases with topical information (Hasan and Ng, 2014). Open information extractors will most likely fail to generate meaningful facts in these circumstances. However, text summarization techniques to score the atomic units can be exploited for fact salience.

Open facts have been already used in text summarization for redundancy, using synonymity (Christensen et al., 2013) or as input for a classifier (Christensen et al., 2014). In this case, we use facts as atomic units. Working at the fact level provides a natural framework to detect essential information in a text document, since facts are minimal comprehensive atomic units expressing a single proposition (Del Corro and Gemulla, 2013). This helps to avoid working with sentences that might express more than one proposition or arbitrary chunking the input text. Compression is also more principled at a fact level as the fact hierarchical structure is clearly defined (Gashteovski et al., 2017). Additionally, we exploit the fact structure to promote diversity.

Several supervised and unsupervised methods have been used in text summarization to determine the relative prominence of the atomic units. For instance, two of the top performer systems Durrett et al. (2016) and Mihalcea and Tarau (2004), which we include in our extrinsic evaluation, are based on ILP and an unsupervised graph algorithm respectively. Other approaches include LDA (Pouriyeh et al., 2017), ontology-based (Baralis et al., 2013) or clustering (Yang et al., 2014), and more recently neural-based methods (See et al., 2017). As in Mihalcea and Tarau (2004) or Erkan and Radev (2004) we use PageRank to establish the relative prominence of the atomic units (Sec. 4.1). However, we weight the graph edges using word vectors to allow more expressive semantics, avoiding the sparsity of frequency-based methods.

Different approaches have also been explored to promote diversity. Xiong and Luo (2014), for example, use LSA, and Chien and Chang (2013) rely on topic models. In our case, we generate diversity by exploiting the fact structure (Sec. 4.2). We cluster facts in terms of their subjects as a way to have the most relevant information about the different entities appearing in the text. The subject is typically the topic of the clause or proposition (Quirk et al., 1985).

---

[1] `https://github.com/mponza/SalIE`

4 SALIE: Salient Information Extraction

SALIE is a graph-based method for the extraction of *salient* open facts in text documents. Open facts are a structured machine-readable representation of the information in text. Its arguments are not linked to an existing KB. SALIE takes as input all open facts detected by an open information extraction system (in our implementation we use MINIE (Gashteovski et al., 2017)).

SALIE works in two stages: (i) relevance and (ii) diversification. First, a graph with open facts as nodes is instantiated so that PageRank assesses their relative relevance. Later, a clustering algorithm selects a diversified set of facts.

4.1 Fact Relevance

SALIE computes fact *relevance* by growing a complete graph of open facts $G_{OF} = (V, E)$ extracted from the input text. *Coherence* is induced by weighting the edges E between nodes V, whereas a *relevance* prior is induced via the instantiation of the PageRank's teleport vector.

Step 1 – Facts as Nodes. Each node is a fact extracted by MINIE. Undefined facts (with no clear co-reference) [*("He", "plays", "softball")*] or facts with constituents composed by single words (generally uninformative or noisy) [*("doorman", "has", "age")*] are removed.

Step 2 – Coherence: Edge Weighting. We want related facts to get a higher weight assuming that the most relevant facts will be those more central. We weight each edge (u, v) with the *semantic similarity* between u and v as the cosine between the centroid of the word embeddings in the facts. Stanovsky et al. (2015) have shown that learning word embeddings with open facts allows the generation of higher quality vectors . The assumption is that the relatedness of words within a fact is stronger than with words outside. This provides the basis for more accurate contextualization. Accordingly, in our implementation we use GloVe (Pennington et al., 2014) trained on the Wikipedia corpus using open facts extracted by MINIE for co-occurence context.

Step 3 – Relevance Prior. We introduce a prior for each fact by computing a score used to instantiate the PageRank's teleport vector. The assumption is that authors tend to express the most relevant facts at the beginning. We instantiated each fact teleport as $factPrior(i) = \frac{x_i}{\|X\|}$, where $x_i = |V| - i$ and i is the fact index. This is important especially for news where the lead paragraph is the most important part of the article. That's why the positional baseline is so strong in tasks as text summarization or entity salience (Ponza et al., 2018).

Step 4 – Relevance Computation. This stage runs PageRank on the graph. The stationary distribution will capture the *relevance* of each open fact. This distribution reflects the semantic centrality of each fact weighted by its relevance prior.

4.2 Fact Diversification

In this stage SALIE diversifies the set of *relevant* facts computed in the previous stage. Facts are clustered exploiting the fact structure, and the most relevant facts in each cluster are selected according to the relevance scores.

Facts have clear semantics regarding the role of each of its constituents (i.e., subject, relation, and object) in the proposition. SALIE exploits this by clustering together those facts that have the same head in the subject's constituent. As the subject is typically the theme (or topic) of a the clause (Quirk et al., 1985), the intuition here is that facts with the same subject express information about the same entity. Therefore, each cluster will contain a ranked set of facts about each entity in the document.

After the facts have been clustered, we iteratively select facts from each cluster according to its relevance until we reach the desired number of facts as output. The number of facts in the output is a parameter of the system.

5 Experiments

Methodology. Given a document we want to evaluate how salient the top-k facts are. The number of facts in the ranking is a parameter of the model so we evaluate 5 configurations: top-1 to top-5 facts.
Dataset. As there is no dataset to directly asses the saliency of facts, we compare the extracted facts in each ranking with a manually generated summary. We use the New York Times (Sandhaus, 2008) corpus, consisting of 3956 news articles and summaries from 2007 (with summaries larger than 50 tokens) as described by Durrett et al. (2016).
Metrics. To measure how close is the ranking to the summary, we use the ROUGE package[2], standard for document summarization (Lin,

---

[2] `pypi.org/project/pyrouge/0.1.3`

Human Summary		
Body of Toni Grossi Abrams, widow and Staten Island socialite, is found in warehouse on outskirts of Panama City, Panama, where she had moved to begin career in real estate; Debra Ann Ridgley, one of her tenants, is charged with stabbing Abrams to death in her apartment on April 9.		
Method		**Salient Facts / Summary**
Position	1	*("Surgery patients", "lie low in", "style retreat")*
	2	*("Remains", "were discovered beside warehouse at", "edge of cinder-topped soccer field on outskirts of Panama City")*
	3	*("Abrams", "had been stabbed to death in", "apartment")*
TextRank	1	*("Ridgley", "was in Abrams's apartment", "Garcia and friend")*
	2	*("Ridgley", "was in Abrams's apartment that", "night")*
	3	*("Abrams's body", "remains in", "Panama City morgue")*
Berkeley		*The widow of a mortgage executive, Ms. Abrams was something of a force of nature in Staten Island society. The suspect, Debra Ann Ridgley, is.*
SalIE	1	*("Abrams", "had been stabbed to death in", "apartment")*
	2	*("Remains", "were discovered beside warehouse at", "edge of cinder-topped soccer field on outskirts of Panama City")*
	3	*("Apartment", "tending wounds at time of", "murder")*

(a) MinIE *safe* mode.

Human Summary		
Russian state oil company Rosneft has lined up $22 billion in financing from consortium of Western banks to buy assets from bankrupt rival Yukos; Rosneft says it will bid for refineries owned by Yukos as outlet for production from its Yugansk subsidiary in western Siberia; some of banks listed.		
Method		**Salient Facts / Summary**
Position	1	*("State oil company", "lined up $ from consortium of banks buy assets from", "rival")*
	2	*("Rosneft", "increase footprint in", "oil and gas business")*
	3	*("Bids", "are successful as", "expected")*
TextRank	1	*("Banks", "made loans to", "Rosneft and state company")*
	2	*("Banks", "lent company related to Rosneft", "$ increase share")*
	3	*("State oil company", "lined up $ from consortium of banks buy assets from", "rival")*
Berkeley		*The Russian state oil company Rosneft has lined up $22 billion from a consortium of Western banks.*
SalIE	1	*("State oil company", "lined up $ from consortium of banks buy assets from", "rival")*
	2	*("Banks", "made loans to", "Rosneft and state company")*
	3	*("Rosneft", "increase footprint in", "oil and gas business")*

(b) MinIE *aggressive* mode.

Table 1: Top-3 salient facts automatically extracted from a sample of two NYT documents with two different MinIE modes. For Berkeley (which does not return facts) we show its produced summary. On the top of each table we show the summary written by a human for the input document.

2004). ROUGE-1 measures the presence of single words between the salient facts and the summary; ROUGE-L identifies the longest common subsequence (LCS) with maximum length between facts and summary; ROUGE-1.2W measures the weighted LCS by taking into account spatial relations and giving higher values to consecutive matches; ROUGE-SU is the number of occurring bigrams between the facts and summary with arbitrary gaps. For each metric we report the F_1 performance, all computed with a 95% confidence interval, run with stemming and stopword removal[3].

To compute the ROUGE score, the facts were flattened and concatenated into a sequence of tokens respecting the ranking order. For the computation, this sequence is considered equivalent to a summary, so the same conditions apply: If all the extracted tokens fully cover the gold standard summary, the ROUGE score reaches its highest value.

Note that we do not take into account the correctness of the facts (i.e., if they are well-structured). All systems implemented, except the Berkeley summarizer (Durrett et al., 2016), use the same open facts extracted by MinIE. Also for the Berkeley summarizer, we do not evaluate the structure or fluency of the summary.

SalIE. Outputs top-k facts per article. We show results for two MinIE configurations: *safe* and *aggressive*, which differ in the fact average size.

Intrinsic Evaluation. As there is no direct fact salience competitor, we designed three baselines: The standard *Position* baseline which ranks facts with respect to their order of appearance, *Tf-Idf* which ranks them with respect to the subject's head tf-idf and the *Context* baseline which ranks facts with respect to the cosine-similarity between the document and the fact embedding's centroid.

Extrinsic Evaluation. We used two state-of-the-art document summarizers, i.e., the unsupervised graph-based TextRank (Mihalcea and Tarau, 2004) and the supervised Berkeley summa-

---

[3] Package arguments: `-c 95 -m -s -U -w 1.2`.

Method	**ROUGE-1**					**ROUGE-L**					**ROUGE-1.2W**					**ROUGE-SU**				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
Position	13.9	20.4	24.8	27.8	29.7	12.8	18.1	21.8	24.4	26.0	6.20	8.80	10.8	12.2	13.2	2.70	5.30	7.50	9.00	10.0
Tf-Idf	10.5	15.8	19.0	21.0	22.3	9.60	13.2	15.7	17.5	18.5	4.60	6.30	7.50	8.50	9.20	1.40	2.80	4.00	4.80	5.30
Context	13.6	19.6	22.8	24.6	25.7	11.5	16.3	18.9	20.4	21.3	5.60	8.00	9.40	10.3	11.0	2.50	4.40	5.60	6.30	6.70
TextRank	15.2	21.5	24.5	26.1	26.8	13.0	17.5	19.8	21.3	22.0	6.20	8.40	9.70	10.6	11.2	2.60	4.90	6.40	7.20	7.50
Berkeley	8.50	18.0	25.4	30.4	34.1	8.00	16.3	22.5	26.7	29.7	3.80	7.70	11.0	13.2	14.9	0.80	3.40	6.90	10.1	12.7
SALIE	17.1	24.2	28.0	30.0	30.9	15.3	21.2	24.3	26.0	26.8	7.40	10.3	12.0	13.1	13.6	3.60	6.50	8.30	9.20	9.50
Diff.	**+1.9**	**+2.7**	**+2.6**	-0.4	-3.2	**+2.3**	**+3.1**	**+1.8**	-0.7	-2.9	**+1.2**	**+1.5**	**+1.0**	-0.1	-1.3	**+0.9**	**+1.2**	**+0.8**	-0.9	-3.2

(a) MINIE *safe* mode.

Method	**ROUGE-1**					**ROUGE-L**					**ROUGE-1.2W**					**ROUGE-SU**				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
Position	10.1	15.3	19.3	22.3	24.5	9.60	13.8	17.2	19.8	21.7	4.40	6.30	7.80	9.10	10.1	1.40	2.90	4.50	5.80	6.90
Tf-Idf	8.90	13.6	16.4	18.3	19.6	8.30	11.3	13.4	15.0	16.1	3.80	5.20	6.20	6.90	7.60	0.90	2.00	2.90	3.50	4.10
Context	9.50	14.5	17.9	20.0	21.4	8.40	12.4	15.1	16.8	18.0	3.90	5.70	6.90	7.80	8.50	1.20	2.40	3.40	4.20	4.80
TextRank	11.3	17.2	20.3	22.2	23.3	10.1	14.3	16.7	18.2	19.2	4.60	6.40	7.60	8.40	9.00	1.30	2.90	4.20	5.00	5.60
Berkeley	3.60	10.6	16.2	21.2	25.4	3.50	9.90	14.8	19.0	22.5	1.60	4.60	7.00	9.10	11.0	0.20	1.20	2.80	4.80	6.90
SALIE	11.6	17.9	21.6	24.2	25.9	10.5	15.9	19.1	21.3	22.8	4.80	7.20	8.60	9.70	10.5	1.60	3.30	4.60	5.70	6.50
Diff.	**+0.3**	**+0.7**	**+1.3**	**+1.9**	**+0.5**	**+0.4**	**+1.6**	**+1.9**	**+1.5**	**+0.3**	**+0.2**	**+0.8**	**+0.8**	**+0.6**	-0.5	**+0.2**	**+0.4**	**+0.1**	-0.1	-0.4

(b) MINIE *aggressive* mode.

Table 2: Results on the NYT dataset with two different MINIE modes.

rizer (Durrett et al., 2016). We adapted TextRank to work with facts instead of sentences. For the Berkeley summarizer, we used the model online[4]. As the size of the summaries is a parameter of the summarizer, we set it to match the average size of MINIE facts (*safe* is 10 and *aggressive* is 6), For example, for the top-5 configuration in the aggressive mode, the summary length is set to 30.

Tab. 1 shows example outputs for the position baseline, the text summarizers and SALIE.

5.1 Results

Tabs. 2a and 2b show the results for all the systems and baselines. We use colors **black**, **gray** and **light gray** for the first, second and third best performing methods. In each ROUGE configuration, we show results for five rankings: top-1 to top-5. The difference between SALIE and the best competitor is reported in the last line of the tables.

Tab. 2a shows the results where facts have been extracted with MINIE's *safe* mode. SALIE outperforms all other methods and baselines for the first three rankings (top-1 to top-3), although Berkeley summarizer comes first in top-4 and 5 facts as a higher budget takes the system closer to the gold standard human-readable summaries. TextRank has an opposite behavior compared to Berkeley, performing well in top-1 and 2 but lagging behind as more facts are added probably due to the lack of a diversification stage. It is interesting to note that systems working at the fact level do well in constrained settings, suggesting that facts may be an effective way to compress information.

Tab. 2b shows the results when MINIE is used in *aggressive* mode. In this experiment, we aim to analyze the behavior of the systems in a restricted scenario with a very small budget size (6 tokens per fact). SALIE achieves the highest performance overall metrics independently the number of facts used, with the only exception on the ROUGE-1.2W and ROUGE-SU score when 4 or 5 facts are used. The second and third best performing methods are Position and TextRank. Again, in this case, it is suggested that facts are an appropriate mechanism to compress information.

Overall SALIE shows a more stable balance across all rankings in both settings. It always ranks first or second (except in ROUGE-SU top-5 where it comes third). Compared to TextRank it seems to significantly better manage redundancy, while compared to the Berkeley it does better at detecting relevant information in constrained settings. This is due to the use of facts as a mean to compress information.

6 Conclusions

We introduced the fact salience task. We also presented SALIE, the first fact salience system. SALIE outperformed standard baselines but also state-of-the-art automatic text summarizer. We showed that working at the fact level allows to more effectively compress information.

---

[4] `nlp.cs.berkeley.edu/projects/summarizer.shtml`

References

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2018. Never-ending learning for open-domain question answering over knowledge bases. In *Proceedings of the WWW*.

Mehdi Allahyari, Seyed Amin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text summarization techniques: A brief survey. *CoRR*.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of IJCAI*.

Elena Baralis, Luca Cagliero, Saima Jabeen, Alessandro Fiori, and Sajid Shah. 2013. Multi-document summarization based on the Yago ontology. *Expert Systems with Applications*.

Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haussmann. 2014. Semantic full-text search with Broccoli. In *Proceedings of SIGIR*.

Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2017. Overview of the triple scoring task at the WSDM cup 2017. *Proceedings of WSDM*.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proceedings of EMNLP*.

Jen-Tzung Chien and Ying-Lan Chang. 2013. Hierarchical theme and topic model for summarization. In *International Workshop on MLSP*.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *Proceedings of NAACL-HLT*.

Janara Christensen, Stephen Soderland, Gagan Bansal, and Mausam. 2014. Hierarchical summarization: Scaling up multi-document summarization. In *Proceedings of ACL*.

Luciano Del Corro and Rainer Gemulla. 2013. ClausIE: Clause-based open information extraction. In *Proceedings of WWW*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of ACL*.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*.

Kiril Gashteovski, Rainer Gemulla, and Luciano Del Corro. 2017. MinIE: minimizing facts in open information extraction. In *Proceedings of EMNLP*.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of ACL*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of EMNLP*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of IJCNLP*.

Dat Ba Nguyen, Abdalghani Abujabal, Nam Khanh Tran, Martin Theobald, and Gerhard Weikum. 2017. Query-driven on-the-fly knowledge base construction. *Proceedings of VLDB*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*.

Marco Ponza, Paolo Ferragina, and Francesco Piccinno. 2018. Swat: A system for detecting salient wikipedia entities in texts. *CoRR*.

Seyedamin Pouriyeh, Mehdi Allahyari, Krzysztof Kochut, Gong Cheng, and Hamid Reza Arabnia. 2017. Es-lda: Entity summarization using knowledge-based topic modeling. In *Proceedings of IJCNLP*.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *Proceedings of ACL*.

Gabriel Stanovsky, Ido Dagan, et al. 2015. Open ie as an intermediate structure for semantic tasks. In *Proceedings of ACL*.

Shuchu Xiong and Yihui Luo. 2014. A new approach for multi-document summarization based on latent semantic analysis. In *Proceedings of ISCID*.

Libin Yang, Xiaoyan Cai, Yang Zhang, and Peng Shi. 2014. Enhancing sentence-level clustering with ranking-based clustering framework for theme-based summarization. *Information Sciences*.

David Zajic, Bonnie J Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing & Management*.

Entity Tracking Improves Cloze-style Reading Comprehension

Luong Hoang **Sam Wiseman** **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA
lhoanger@gmail.com, {swiseman,srush}@seas.harvard.edu

Abstract

Reading comprehension tasks test the ability of models to process long-term context and remember salient information. Recent work has shown that relatively simple neural methods such as the Attention Sum-Reader can perform well on these tasks; however, these systems still significantly trail human performance. Analysis suggests that many of the remaining hard instances are related to the inability to track entity-references throughout documents. This work focuses on these hard entity tracking cases with two extensions: (1) additional entity features, and (2) training with a multi-task tracking objective. We show that these simple modifications improve performance both independently and in combination, and we outperform the previous state of the art on the LAMBADA dataset, particularly on difficult entity examples.

1 Introduction

There has been tremendous interest over the past several years in Cloze-style (Taylor, 1953) reading comprehension tasks, datasets, and models (Hermann et al., 2015; Hill et al., 2016; Kadlec et al., 2016; Dhingra et al., 2016; Cui et al., 2016). Many of these systems apply neural models to learn to predict answers based on contextual matching, and have inspired other work in long-form generation and question answering. The extent and limits of these successes have also been a topic of interest (Chen et al., 2016; Chu et al., 2017). Recent analysis by Chu et al. (2017) suggests that a significant portion of the errors made by standard models, especially on the LAMBADA dataset (Paperno et al., 2016), derive from the inability to correctly track entities or speakers, or a failure to handle various forms of reference.

This work targets these shortcomings by designing a model and training scheme targeted towards *entity tracking*. Specifically we introduce two simple changes to a stripped down model: (1) simple, entity-focused features, and (2) two multi-task objectives that target entity tracking. Our ablation analysis shows that both independently improve entity tracking, which is the primary source of overall model's improvement. Together they lead to state-of-the-art performance on LAMBADA dataset and near state-of-the-art on CBT dataset (Hill et al., 2016), even with a relatively simple model.

usually , that teen behavior was directed toward a parent , but since [julie] had somewhat filled the role of both parents for amy , [she] knew that **amy** was now directing that rebellion at [her] . `` i 'm sorry , marsh , '' [**julie**] glanced at [her] friend out of the corner of [her] eye as [she] drove . `` do n't be . '' `` do n't apologize for me , [**julie***]

Figure 1: A LAMBADA example where the final word "julie" (with reference chain in brackets) is the answer, y, to be predicted from the preceding context x. A system must know the two speakers and the current dialogue turn, simple context matching is not sufficient. Here, our model's predictions before and after adding multi-task objective are shown.

2 Background and Related Work

Cloze-style reading comprehension uses a passage of word tokens $x = x_{1:n}$ (the *context*), with one token x_j masked; the task is to fill in the masked word y, which was originally at position j. These datasets aim to present a benchmark challenge requiring some understanding of the context to select the correct word. This task is a prerequisite for problems like long-form generation and document-based question answering.

A number of datasets in this style exist with dif-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1049–1055
Brussels, Belgium, October 31 - November 4, 2018.

ferent focus. Here we considered the LAMBADA dataset and the named entity portion of the Children's Book Test dataset (CBT-NE). LAMBADA uses novels where examples consist of 4-5 sentences and the last word to be predicted is masked, x_n. The dataset is constructed carefully to focus on examples where humans needed the context to predict the masked word. CBT-NE examples, on the other hand, include 21 sentences where the masked word is a named entity extracted from the last sentence, with $j \leq n$, and is constructed in a more automated way. We show an example from LAMBADA in Figure 1. In CBT, as well as the similar CNN/Daily Mail dataset (Hermann et al., 2015), the answer y is always contained in x whereas in LAMBADA it may not be. Chu et al. (2017) showed, however, that training only on examples where y *is* in x leads to improved overall performance, and we adopt this approach as well.

Related Work The first popular neural network reading comprehension models were the Attentive Reader and its variant Impatient Reader (Hermann et al., 2015). Both were the first to use bidirectional LSTMs to encode the context paragraph and the query separately. The Stanford Reader (Chen et al., 2016) is a simpler version with fewer layers for inference. These models use an encoder to map each context token x_i to a vector $\boldsymbol{u}_i$. Following the terminology of Wang et al. (2017), *explicit reference models* calculate a similarity measure $s_i = s(\boldsymbol{u}_i, \boldsymbol{q})$ between each context vector $\boldsymbol{u}_i$ and a query vector $\boldsymbol{q}$ derived for the masked word. These similarity scores are projected to an attention distribution $\boldsymbol{\alpha} = \mathrm{softmax}(\{s_i\})$ over the context positions in $1, \ldots, n$, which are taken to be candidate answers.

The Attention Sum Reader (Kadlec et al., 2016) is a further simplified version. It computes $\boldsymbol{u}_i$ and $\boldsymbol{q}$ with separate bidirectional GRU (Chung et al., 2014) networks, and s_i with a dot-product. It is trained to minimize:

$$\begin{aligned}\mathcal{L}^0(\boldsymbol{\theta}) &= -\ln p(y \,|\, x, \boldsymbol{q}) \\ &= -\ln \sum_{i: x_i = y} p(x_i \,|\, \boldsymbol{q}) = -\ln \sum_{i: x_i = y} \alpha_i,\end{aligned}$$

where $\boldsymbol{\theta}$ is the set of all parameters associated with the model, and y is the correct answer. At test time, a *pointer sum attention* mechanism is used to predict the word type with the highest aggregate attention as the answer. The Gated Attention Reader (Dhingra et al., 2016) leverages the same mechanism for prediction and introduces an attention gate to modulate the joint context-query information over multiple hops.

The Recurrent Entity Networks (Henaff et al., 2016) uses a custom gated recurrent module, Dynamic Memory, to learn and update entity representations as new examples are received. Their gate function is combined of (1) a similarity measure between the input and the hidden states, and (2) a set of trainable "key" vectors which could learn any attribute of an entity such as its location or other entities it is interacting with in the current context. The Query Reduction Networks (Seo et al., 2016) is also a gated recurrent network which tracks state in a paragraph and uses a hidden query vector to keep pointing to the answer at each step. The query is successively transformed with each new sentence to a reduced state that's easier to answer given the new information.

Model In this work, we were particularly interested in the shortcomings of simple models and exploring whether or how much entity tracking could help, since Chu et al. (2017) has pointed out this weakness. As a result, we adapt a simplified Attention Sum (AttSum) reader throughout all experiments. Our version uses only a single bidirectional GRU for both $\boldsymbol{u}_i$ and $\boldsymbol{q}$. This GRU is of size $2d$, using the first d states for the context and second d for the query. Formally, let $\overrightarrow{\boldsymbol{h}}_i$ and $\overleftarrow{\boldsymbol{h}}_i$ (both in $\mathbb{R}^{2d}$) represent the forward and backward states of a bidirectional GRU run over x, and let $\overrightarrow{\boldsymbol{h}}_{i,\uparrow}$ and $\overrightarrow{\boldsymbol{h}}_{i,\downarrow}$ be the first and second d states respectively, and define $||$ as the concatenation operator. The context vectors are constructed as $\boldsymbol{u}_i = \overrightarrow{\boldsymbol{h}}_{i,\uparrow} || \overleftarrow{\boldsymbol{h}}_{i,\uparrow}$. For datasets using the last word, the query is constructed as $\boldsymbol{q} = \overrightarrow{\boldsymbol{h}}_{n,\downarrow} || \overleftarrow{\boldsymbol{h}}_{1,\downarrow}$. When the masked word can be anywhere, the query is constructed as $\boldsymbol{q} = \overrightarrow{\boldsymbol{h}}_{j-1,\downarrow} || \overleftarrow{\boldsymbol{h}}_{j+1,\downarrow}$.

Our main contribution is the extension of this simple model to incorporate entity tracking. Other authors have explored extending neural reading comprehension models with linguistic features, particularly Dhingra et al. (2017) who use a modified GRU with knowledge such as coreference relations and hypernymy. In Dhingra et al. (2018), the most recent coreferent antecedent for each token is incorporated into the update equations of the GRU unit to bias the reader towards coreferent recency. In this work, we instead use a much sim-

1	Sentence Index, POS Tag, NER Tag
2	Is among last 3 PERSON words in $\boldsymbol{x}$
3	Is a PERSON word in the last sentence
4	Is a PERSON word identical to previous PERSON word
5	Is a PERSON word identical to next PERSON word
6	Quoted-speech Index
7	Speaker

Table 1: Word-level features used in AttSum-Feat model.

pler set of features and compare to this and several other models as baseline approaches.

3 Learning to Track Entities

Analysis on reading comprehension has indicated that neural models are strong at matching local context information but weaker at following entities through the discourse (Chen et al., 2016; Chu et al., 2017). We consider two straightforward ways for extending the Attention Sum baseline to better track entities.

Method 1: Features We introduce a short-list of features in Table 1 to augment the representation of each word in x. These features are meant to help the system to identify and use the relationships between words in the passage.[1] Features 2-5 apply only to words tagged PERSON by the NER tagger. Features 6-7 apply only to words between opening and closing quotation marks. Feature 6 indicates the index of the quote in the document, and Feature 7 gives the assumed speaker of the quote using some simple rules; we provide the rules in the Supplementary Material. Though most of these features are novel, they are motivated by recent analysis (Wang et al., 2015; Chen et al., 2016; Wang et al., 2017).

All features are incorporated into a word's representation by embedding each discrete feature into a vector of the same size as the original word embedding, adding the vectors as well as a bias, and applying a $\tanh$ nonlinearity.

Method 2: Multitasking We additionally encourage the neural model to keep track of entities by multitasking with simple auxiliary entity-tracking tasks. Examples such as Figure 1 suggest that keeping track of which entities are currently in scope is useful for answering reading comprehension questions. There, *amy* and *julie* are conversing, and being able to track that *amy* is the speaker of the final quote helps to rule her out as a candidate answer. We consider two tasks:

For Task 1 ($\mathcal{L}^1$) we train the same model to predict repeated named entities. For all named entities x_j such that there is a $x_i = x_j$ with $i < j$, we attempt to mask and predict the word type x_j. This is done by introducing another Cloze prediction, but now setting the target $y = x_j$, reducing the context to preceding words $x_{1:j-1}$ with $\mathbf{u_i} = \vec{\boldsymbol{h}}_i$, and the query $\boldsymbol{q} = \vec{\boldsymbol{h}}_{j-1}$. (Note that unlike above, both of these only use the forward states of the GRU). We use a bilinear similarity score $s_i = \boldsymbol{q}^\top \boldsymbol{Q}\, \boldsymbol{u}_i$, for this prediction where $\boldsymbol{Q}$ is a learned transformation in $\mathbb{R}^{2d\times 2d}$. This task is inspired by the antecedent ranking task in coreference (Wiseman et al., 2015, 2016).

For Task 2 ($\mathcal{L}^2$) we train to predict the order index in which a named entity has been introduced. For example, in Figure 1, *julie* would be 1, *amy* would be 2, *marsh* would be 3, etc. The hope here is that learning to predict when entities reappear will help the model track their reoccurences. For the blue labeled *julie*, the model would aim to predict 1, even though it appears later in the context. This task is inspired by the One-Hot Pointer Reader of Wang et al. (2017) on the Who-did-What dataset (Onishi et al., 2016). Formally, letting $\widehat{\mathrm{idx}}(x_j)$ be the predicted index for x_j, we minimize:

$$\begin{aligned}\mathcal{L}^2(\boldsymbol{\theta}) &= -\ln p(\widehat{\mathrm{idx}}(x_j) = \mathrm{idx}(x_j) \mid x_{1:j-1}) \\ &= -\ln \mathrm{softmax}(\boldsymbol{W}\vec{\boldsymbol{h}}_j)_{\mathrm{idx}(x_j)},\end{aligned}$$

where $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{E}|\times 2d}$ and $\mathcal{E}$ is the set of entity word types in the document. Note that this is a simpler computation, requiring only $O(|\mathcal{E}| \times n)$ predictions per $\boldsymbol{x}$, whereas $\mathcal{L}^1$ requires $O(n^2)$.

The full model minimizes a multi-task loss: $\mathcal{L}^0(\boldsymbol{\theta}) + \gamma_1\mathcal{L}^1(\boldsymbol{\theta}) + \gamma_2\mathcal{L}^2(\boldsymbol{\theta})$. Using $\mathcal{L}^1$ and $\mathcal{L}^2$ simultaneously did not lead to improved performance however, and so either γ_1, γ_2 is always 0. We believe that this is because, while the learning objectives for $\mathcal{L}^1$ and $\mathcal{L}^2$ are mathematically different, they are both designed to similarly track the entities mentioned so far in the document and thus do not provide complementary information to each other.

We found it useful to have two hyperparameters per auxiliary task governing the number of distinct

---

[1] POS tags are produced with the NLTK library (Bird et al., 2009), and NER tags with the Stanford NER tagger (Finkel et al., 2005). We additionally found it useful to tag animate words as PERSONs on the CBT-NE data, using the animate word list of Bergsma and Lin (2006).

named entity word *types* and *tokens* used in defining the losses $\mathcal{L}^1$ and $\mathcal{L}^2$. In particular, per document these hyperparameters control in a top-to-bottom order the number of distinct named entity word types we attempt to predict, as well as the number of tokens of each type considered.

4 Experiments

Methods This section highlights several aspects of our methodology; full hyperparameters are given in the Supplementary Material. For the training sets, we exclude examples where the answer is not in the context. The validation and test sets are not modified however and the model with the highest accuracy on the validation set is chosen for testing. For both tasks, the context words are mapped to learned embeddings; importantly, we initialize the first 100 dimensions with the 100-dimensional GLOVE embeddings (Pennington et al., 2014). Named entity words are anonymized, as is done in the CNN/Daily Mail corpus (Hermann et al., 2015) and in some of the experiments of Wang et al. (2017). The model is regularized with dropout (Srivastava et al., 2014) and optimized with ADAM (Kingma and Ba, 2014). For all experiments we performed a random search over hyperparameter values (Bergstra and Bengio, 2012), and report the results of the models that performed best on the validation set. Our implementation is available at `https://github.com/harvardnlp/readcomp`.

Results and Discussion Table 2 shows the full results of our best models on the LAMBADA and CBT-NE datasets, and compares them to recent, best-performing results in the literature.

For both tasks the inclusion of either entity features or multi-task objectives leads to large statistically significant increases in validation and test score, according to the McNemar test ($\alpha = 0.05$) with continuity correction (Dietterich, 1998). Without features, AttSum + $\mathcal{L}^2$ achieves the best test results, whereas with features AttSum-Feat + $\mathcal{L}^1$ performs best on CBT-NE. The results on LAMBADA indicate that entity tracking is a very important overlooked aspect of the task. Interestingly, with features included, AttSum-Feat + $\mathcal{L}^2$ appears to hurt test performance on LAMBADA and leaves CBT-NE performance essentially unchanged, amounting to a negative result for $\mathcal{L}^2$. On the other hand, the effect of AttSum-Feat + $\mathcal{L}^1$ is pronounced on CBT-NE, and while our simple models do not increase the state-of-the-art test performance on CBT-NE, they outperform "attention-over-attention" in addition to reranking (Cui et al., 2016), and is outperformed only by architectures supporting "multiple-hop" inference over the document (Dhingra et al., 2016). Our best model on CBT-NE test set, AttSum-Feat + $\mathcal{L}^1$, is very close to the current state-of-the-art result. On the validation sets for both LAMBADA and CBT-NE, the improvements from adding features to AttSum + $\mathcal{L}^i$ are statistically significant (for full results refer to our supplementary material). On LAMBADA, the $\mathcal{L}^1$ multi-tasked model is a 3.5-point increase on the state of the art.

Our method also employs fewer parameters than other richer models such as the GA Reader in (Dhingra et al., 2016). More specifically, in terms of number of parameters, our models are very similar to a 1-hop GA Reader. In contrast, all published experiments of the latter use 3 hops where each hop requires 2 separate Bi-GRUs, one

LAMBADA	Val	Test
GA Reader (Chu et al., 2017)	-	49.00
MAGE (48) (Dhingra et al., 2017)	51.10	51.60
MAGE (64) (Dhingra et al., 2017)	52.10	51.10
GA + C-GRU (Dhingra et al., 2018)	-	55.69
AttSum	56.03	55.60
AttSum + $\mathcal{L}^1$	58.35	56.86
AttSum + $\mathcal{L}^2$	58.08	57.29
AttSum-Feat	59.62	59.05
AttSum-Feat + $\mathcal{L}^1$	**60.22**	**59.23**
AttSum-Feat + $\mathcal{L}^2$	60.13	58.47
CBT-NE		
GA Reader (Dhingra et al., 2016)	78.50	**74.90**
EpiReader (Trischler et al., 2016)	75.30	69.70
DIM Reader (Liu et al., 2017)	77.10	72.20
AoA (Cui et al., 2016)	77.80	72.0
AoA + Reranker (Cui et al., 2016)	**79.60**	74.0
AttSum	74.35	69.96
AttSum + $\mathcal{L}^1$	76.20	72.16
AttSum + $\mathcal{L}^2$	76.80	72.60
AttSum-Feat	77.80	72.36
AttSum-Feat + $\mathcal{L}^1$	78.40	74.36
AttSum-Feat + $\mathcal{L}^2$	79.40	72.40

Table 2: Validation & Test results on all datasets. AttSum* are our models, including variants with features and multi-task loss. Others indicate previous best published results. All improvements over AttSum are statistically significant ($\alpha = 0.05$) according to the McNemar test with continuity correction (Dietterich, 1998).

LAMBADA	All	Entity	Speaker	Quote
AttSum	56.03	75.17	74.81	73.31
AttSum + $\mathcal{L}^1$	58.35	78.51	78.38	79.42
AttSum + $\mathcal{L}^2$	58.08	78.17	77.96	76.76
AttSum-Feat	59.62	79.40	80.34	79.68
AttSum-Feat + $\mathcal{L}^1$	60.22	82.00	82.98	81.67
AttSum-Feat + $\mathcal{L}^2$	60.14	82.06	83.06	82.60
CBT-NE				
AttSum	74.35	76.28	75.08	74.96
AttSum + $\mathcal{L}^1$	76.20	78.03	76.98	77.33
AttSum + $\mathcal{L}^2$	76.80	77.45	76.27	76.48
AttSum-Feat	77.80	80.58	79.84	79.61
AttSum-Feat + $\mathcal{L}^1$	78.40	80.44	79.68	79.78
AttSum-Feat + $\mathcal{L}^2$	79.40	82.41	81.51	81.39

Table 3: Ablation results on validation sets, see text for definitions of the numeric columns and models.

to model the document and one for the query. This constitutes the largest difference in model size between the two approaches.

Table 3 considers the performance of the different models based on a segmentation of the data. Here we consider examples where: (1) Entity - if the answer is a named entity; (2) Speaker - if the answer is a named entity and the speaker of quote; (3) Quote - if the answer is found within a quoted speech. Note that Speaker and Quote categories, while mutually exclusive, are subsets of the overall Entity category. We see that both the additional features and multi-task objectives independently result in a clear improvement in all categories, but that the gains are particularly pronounced for named entities and specifically for Speaker and Quote examples. Here we see sizable increases in performance, particularly in the Speaker category. We see larger increases in the more dialog heavy LAMBADA task.

As a qualitative example of the improvement afforded by multi-task training, in Figure 1 we show the different predictions made by our model with and without $\mathcal{L}^1$ (colored as blue and red, respectively). Note that *amy* and *julie* are both entities that have been repeated twice in the passage. In addition to the final answer, our model with the $\mathcal{L}^1$ loss was also able to predict these entities (at the colored locations) given preceding words. Further qualitative analysis reveals that these augmentations improved the model's ability to eliminate non-entity choices from predictions. Some examples are shown in Figure 2.

after a few more minutes axel left patrick at the wall and went for a walk around the **village** . he stopped wherever he saw people and chatted . lifting spirits , cracking jokes , giving advice . the **vicar** was holding a service in the village square . a short and simple one . ` when you go out to war against your enemies , ' said the [vicar]

as the piece of rock fell off and was accelerating its way to the ground , he kicked it toward the robot and destroyed it upon contact . **lance** watched on the screen through the point of **view** of the last robot as the rock flew speedily toward it and then his screen cut into black . a soldier quickly walked in and went up to [lance]

Figure 2: LAMBADA examples where AttSum incorrectly predicts a non-entity answer whereas AttSum-Feat and AttSum + $\mathcal{L}^i$ choose correctly.

5 Conclusion

This work demonstrates that learning to track entities with features and multi-task learning significantly increases the performance of a baseline reading comprehension system, particularly on the difficult LAMBADA dataset. This result indicates that higher-level word relationships may not be modeled by simple neural systems, but can be incorporated with minor additional extensions. This work hints that it is difficult for vanilla models to learn long-distance entity relations, and that these may need to be encoded directly through features or possibly with better pre-trained representations.

Acknowledgments

SW gratefully acknowledges the support of a Siebel Scholars award. AMR gratefully acknowledges the support of NSF CCF-1704834, Intel Research, and Amazon AWS Research grants.

References

Shane Bergsma and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 33–40. Association for Computational Linguistics.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyz-*

ing text with the natural language toolkit. " O'Reilly Media, Inc.".

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *ACL*.

Zewei Chu, Hai Wang, Kevin Gimpel, and David A. McAllester. 2017. Broad context language modeling as reading comprehension. In *EACL*, pages 52–57.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. *arXiv preprint arXiv:1804.05922*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Bhuwan Dhingra, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2017. Linguistic knowledge as memory for recurrent neural networks. *arXiv preprint arXiv:1703.02620*.

Thomas G Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zhuang Liu, Degen Huang, Kaiyu Huang, and Jing Zhang. 2017. *DIM Reader: Dual Interaction Model for Machine Comprehension*. Springer International Publishing, Cham.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. *arXiv preprint arXiv:1608.05457*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Query-reduction networks for question answering. *arXiv preprint arXiv:1606.04582*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Wilson L Taylor. 1953. cloze procedure: a new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. 2016. Natural language comprehension with the epireader. *arXiv preprint arXiv:1606.02270*.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 700–706.

Hai Wang, Takeshi Onishi, Kevin Gimpel, and David McAllester. 2017. Emergent predication structure in hidden state vectors of neural readers. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 26–36.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *NAACL HLT*, pages 994–1004.

Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1416–1426.

Adversarial Domain Adaptation for Duplicate Question Detection

Darsh J Shah[1], **Tao Lei**[2], **Alessandro Moschitti**[3]*, **Salvatore Romeo**[4], **Preslav Nakov**[4]
[1]MIT CSAIL, Cambridge, MA, USA
[2]ASAPP Inc., New York, NY, USA
[3]Amazon, Manhattan Beach, CA, USA
[4]Qatar Computing Research Institute, HBKU, Doha, Qatar
`darsh@csail.mit.edu, tao@asapp.com, amosch@amazon.com`
`sromeo@qf.org.qa, pnakov@qf.org.qa`

Abstract

We address the problem of detecting duplicate questions in forums, which is an important step towards automating the process of answering new questions. As finding and annotating such potential duplicates manually is very tedious and costly, automatic methods based on machine learning are a viable alternative. However, many forums do not have annotated data, i.e., questions labeled by experts as duplicates, and thus a promising solution is to use domain adaptation from another forum that has such annotations. Here we focus on adversarial domain adaptation, deriving important findings about when it performs well and what properties of the domains are important in this regard. Our experiments with *StackExchange* data show an average improvement of 5.6% over the best baseline across multiple pairs of domains.

1 Introduction

Recent years have seen the rise of community question answering forums, which allow users to ask questions and to get answers in a collaborative fashion. One issue with such forums is that duplicate questions easily become ubiquitous as users often ask the same question, possibly in a slightly different formulation, making it difficult to find the best (or one correct) answer (Hoogeveen et al., 2018; Lai et al., 2018). Many forums allow users to signal such duplicates, but this can only be done after the duplicate question has already been posted and has possibly received some answers, which complicates merging the question threads. Discovering possible duplicates at the time of posting is much more valuable from the perspective of both (*i*) the forum, as it could prevent a duplicate from being posted, and (*ii*) the users, as they could get an answer immediately.

Duplicate question detection is a special case of the more general problem of question-question similarity. The latter was addressed using a variety of textual similarity measures, topic modeling (Cao et al., 2008; Zhang et al., 2014), and syntactic structure (Wang et al., 2009; Filice et al., 2016; Da San Martino et al., 2016; Barrón-Cedeño et al., 2016; Filice et al., 2017). Another approach is to use neural networks such as feed-forward (Nakov et al., 2016a), convolutional (dos Santos et al., 2015; Bonadiman et al., 2017; Wang et al., 2018), long short-term memory (Romeo et al., 2016), and more complex models (Lei et al., 2016; Nicosia and Moschitti, 2017; Uva et al., 2018; Joty et al., 2018; Zhang and Wu, 2018). Translation models have also been popular (Zhou et al., 2011; Jeon et al., 2005; Guzmán et al., 2016a,b).

The above work assumes labeled training data, which exists for question-question similarity, e.g., from SemEval-2016/2017 (Agirre et al., 2016; Nakov et al., 2016b, 2017), and for duplicate question detection, e.g., SemEval-2017 task 3 featured four StackExchange forums, *Android*, *English*, *Gaming*, and *Wordpress*, from CQADupStack (Hoogeveen et al., 2015, 2016). Yet, such annotation is not available for many other forums, e.g., the *Apple* community on StackExchange.

In this paper, we address this lack of annotation using adversarial domain adaptation (ADA) to effectively use labeled data from another forum. Our contributions can be summarized as follows:

- we are the first to apply adversarial domain adaptation to the problem of duplicate question detection across different domains;[1]
- on the StackExchange family of forums, our model outperforms the best baseline with an average relative improvement of 5.6% (up to 14%) across all domain pairs.

---

[*] Work conducted while the author was at QCRI.

[1] The code and the data are available at the following link: `http://github.com/darsh10/qra_code`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1056–1063
Brussels, Belgium, October 31 - November 4, 2018.

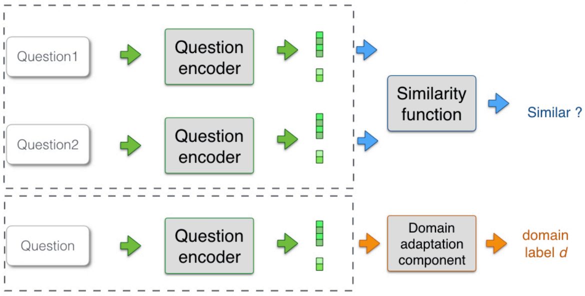

Figure 1: Our cross-domain question-question similarity model. The Question encoder is common for the questions from the source domain and from the target domain. The model and its training procedure are described in Section 2.

- we study when transfer learning performs well and what properties of the domains are important in this regard; and
- we show that adversarial domain adaptation can be efficient even for unseen target domains, given some similarity of the target domain with the source one and with the regularizing adversarial domain.

Adversarial domain adaptation (ADA) was proposed by Ganin and Lempitsky (2015), and was then used for NLP tasks such as sentiment analysis and retrieval-based question answering (Chen et al., 2016; Ganin et al., 2016; Li et al., 2017; Liu et al., 2017; Yu et al., 2018; Zhang et al., 2017), including cross-language adaptation (Joty et al., 2017) for question-question similarity.[2]

The rest of this paper is organized as follows: Section 2 presents our model, its components, and the training procedure. Section 3 describes the datasets we used for our experiments, stressing upon their nature and diversity. Section 4 describes our adaptation experiments and discusses the results. Finally, Section 5 concludes with possible directions for future work.

2 Method

Our ADA model has three components: (*i*) question encoder, (*ii*) similarity function, and (*iii*) domain adaptation component, as shown in Figure 1.

The encoder E maps a sequence of word tokens $x = (x_1, .., x_n)$ to a dense vector $\mathbf{v} = E(x)$. The similarity function f takes two question vectors, $\mathbf{v}_1$ and $\mathbf{v}_2$, and predicts whether the corresponding questions are duplicates.

The domain classifier g takes a question vector $\mathbf{v}$ and predicts whether the question is from the source or from the target domain. We train the encoder not only to do well on the task for the source data, but also to fool the domain classifier, as shown in Algorithm 1. We describe the design choices considered for our domain adaptation model in the following two subsections.

2.1 Question Similarity Function

We consider two options for our similarity function $f(\mathbf{v}_1, \mathbf{v}_2)$:
(*i*) a logistic function that computes the probability that two questions are similar/duplicates, which is trained with the cross-entropy loss:

$$\text{sigmoid}\left(\mathbf{W}^\top(\mathbf{v}_1 \odot \mathbf{v}_2) + \mathbf{b}\right)$$

where $\odot$ is an element-wise vector product between unit encodings of questions;

(*ii*) a simple cosine similarity function, i.e., $cosine(\mathbf{v}_1, \mathbf{v}_2)$, trained using the pairwise hinge loss with a margin m:

$$\sum_i max(\{(1-y^i)f(\mathbf{v}_1^i, \mathbf{v}_2^i) + m - y^i f(\mathbf{v}_1^i, \mathbf{v}_2^i)\}, 0)$$

Our experiments reported in Table 3 show that the $cosine$ similarity function performs far better.

2.2 Domain Adaptation Components

The adversarial component is responsible for reducing the difference between the source and the target domain distributions. There are two common approaches to achieve this: (*i*) classification-based (Ganin and Lempitsky, 2015) and (*ii*) Wasserstein (Arjovsky et al., 2017).

The main difference between them is in the way the domain discrepancy loss is computed. In the classification-based approach, the adversarial component is a classifier trained to correctly predict the domain (source vs. target) of the input question. In contrast, the question encoder is optimized to confuse the domain classifier, which, as a result, encourages domain invariance. Arjovsky and Bottou (2017) showed that this adversarial optimization process resembles minimizing the Jenson-Shannon (JS) divergence between the source P_s and the target distribution P_t:

$$JS(P_s, P_t) = KL(P_s, P_m) + KL(P_g, P_m)$$

where $P_m = (P_s + P_t)/2$ and KL is the Kullback-Leibler divergence.

---

[2] Prior work on cross-language adaptation for question-question similarity used cross-language tree kernels (Da San Martino et al., 2017).

Algorithm 1: Training Procedure

```
Input: source data $X^s$; target data $X^t$
Hyper-parameters: learning rates $\alpha_1, \alpha_2$; batch size $m$; adversarial importance $\lambda$
Parameters to be trained: question encoder $\theta_e$, question similarity classifier $\theta_s$ and domain classifier $\theta_d$
Similarity classification loss $L_c$ is either the cross-entropy loss or hinge loss, described in Section 2.1
Adversarial loss $L_d$, described in Section 2.2
repeat
    for each batch do
        Construct a sub-batch of similar and dissimilar question pairs from the annotated source data $\{(x^s_{i_1}, x^s_{i_2}), y^s_i\}^m_{i=1}$
        Calculate the classification loss $L_c$ using $\theta_e$ and $\theta_s$ for this sub-batch
        Construct a sub-batch of questions $\{x^s_i, x^t_j\}^m_{i=1}$ from the corpora of source and target domains
        Calculate the domain discrepancy loss $L_d$ using $\theta_e$ and $\theta_d$ for this sub-batch
        Total loss $L = L_c - \lambda L_d$
        $\theta_e = \theta_e - \alpha_1 \nabla_{\theta_e} L$
        $\theta_s = \theta_s - \alpha_1 \nabla_{\theta_s} L$
        $\theta_d = \theta_d + \alpha_2 \nabla_{\theta_d} L$
    end for
until $\theta_e$, $\theta_s$ and $\theta_d$ converge
```

In contrast, the Wasserstein method attempts to reduce the approximated Wasserstein distance (also known as *Earth Mover's Distance*) between the distributions for the source and for the target domain as follows:

$$W(P_s, P_t) = \underset{||f||_L \leq 1}{sup} \; \mathbb{E}_{x\sim P_s}[f(x)] - \mathbb{E}_{x\sim P_t}[f(x)]$$

where f is a Lipchitz-1 continuous function realized by a neural network.

Arjovsky et al. (2017) have shown that the Wasserstein method yields more stable training for computer vision tasks.

2.3 Training Procedure

Algorithm 1 describes the procedure to train the three components of our model. Adversarial training needs two kinds of training data: (*i*) annotated question pairs from the source domain, and (*ii*) unlabeled questions from the source and the target domains.

The question encoder is trained to perform well on the source domain using the similarity classification loss L_c. In order to enforce good performance on the target domain, the question encoder is simultaneously trained to be incapable in discriminating between question pairs from the source vs. the target domain. This is done through the domain classification loss L_d.

Dataset	Questions	Duplicates	Train	Dev	Test
AskUbuntu	257,173	27,289	9,106	1,000	1,000
SuperUser	343,033	11,407	9,106	1,000	1,000
Apple	80,466	2,267	–	1,000	1,000
Android	42,970	2,371	–	1,000	1,000
Sprint	31,768	23,826	9,100	1,000	1,000
Quora	537,211	149,306	9,100	–	–

Table 1: Statistics about the datasets. The table shows the number of question pairs that have been manually marked as similar/duplicates by the forum users (i.e., positive pairs). We further add 100 negative question pairs per duplicate question by randomly sampling from the full corpus of questions.

3 Datasets

The datasets we use can be grouped as follows:

- **Stack Exchange** is a family of technical community support forums. We collect questions (composed by title and body) from the XML dumps of four forums: *AskUbuntu*, *SuperUser*, *Apple*, and *Android*. Some pairs of similar/duplicate questions in these forums are marked by community users.
- **Sprint FAQ** is a newly crawled dataset from the Sprint technical forum website. It contains a set of frequently asked questions and their paraphrases, i.e., three similar questions, paraphrased by annotators.
- **Quora** is a dataset of pairs of similar questions asked by people on the Quora website. They cover a broad set of topics touching upon philosophy, entertainment and politics.

Note that these datasets are quite heterogeneous: the *StackExchange* forums focus on specific technologies, where questions are informal and users tend to ramble on about their issues, the *Sprint FAQ* forum is technical, but its questions are concise and shorter, and the *Quora* forum covers many different topics, including non-technical.

Statistics about the datasets are shown in Table 1. Moreover, in order to quantify the differences and the similarities, we calculated the fraction of unigrams, bigrams and trigrams that are shared by pairs of domains. Table 2 shows statistics about the n-gram overlap between *AskUbuntu* or *Quora* as the source and all other domains as the target. As one might expect, there is a larger overlap within the *StackExchange* family.

Source	Target	Unigrams	Bigrams	Trigrams
AskUbuntu	Android	0.989	0.926	0.842
	Apple	0.991	0.926	0.853
	SuperUser	0.990	0.921	0.822
	Sprint	0.959	0.724	0.407
	Quora	0.922	0.696	0.488
Quora	AskUbuntu	0.949	0.647	0.326
	Apple	0.969	0.721	0.426
	Android	0.973	0.762	0.473
	SuperUser	0.958	0.663	0.338
	Sprint	0.942	0.647	0.310

Table 2: Proportion of n-grams that are shared between the source and the target domains.

4 Experiments and Evaluation

4.1 Experimental Setup

Baselines We compare our ADA model to the following baselines: (*a*) *direct transfer*, which directly applies models learned from the source to the target domain without any adaptation; and (*b*) the standard unsupervised *BM25* (Robertson and Zaragoza, 2009) scoring provided in search engines such as Apache Lucene (McCandless et al., 2010).

Models We use a bi-LSTM (Hochreiter and Schmidhuber, 1997) encoder that operates on 300-dimensional GloVe word embeddings (Pennington et al., 2014), which we train on the combined data from all domains. We keep word embeddings fixed in our experiments. For the adversarial component, we use a multi-layer perceptron.

Evaluation Metrics As our datasets may contain some duplicate question pairs, which were not discovered and thus not annotated, we end up having false negatives. Metrics such as MAP and MRR are not suitable in this situation. Instead, we use AUC (area under the curve) to evaluate how well the model ranks positive pairs vs. negative ones. AUC quantifies how well the true positive rate (tpr) grows at various false positive rates (fpr) by calculating the area under the curve starting from $fpr = 0$ to $fpr = 1$. We compute the area integrating the false positive rate (x-axis) from 0 up to a threshold t, and we normalize the area to $[0, 1]$. This score is known as AUC(t). It is more stable than MRR and MAP in our case when there could be several false negatives.[3]

Adaptation	Similarity	AUC(0.05)	AUC(0.1)
—	Sigmoid	0.431	0.557
—	Cosine	0.692	0.782
Classification	Cosine	0.791	0.862
Wasserstein	Cosine	0.795	0.869

Table 3: Duplicate question detection: direct transfer vs. adversarial domain adaptation from *AskUbuntu* to *Android*.

Source	Target	Direct	BM25	Adv.
AskUbuntu	Android	0.692	0.681	**0.790**
	Apple	0.828	0.747	**0.855**
	SuperUser	0.908	0.765	**0.911**
	Sprint	0.917	**0.956**	0.937
SuperUser	AskUbuntu	0.730	0.644	**0.796**
	Apple	0.828	0.747	**0.861**
	Android	0.770	0.681	**0.790**
	Sprint	0.928	**0.956**	0.932

Table 4: Domain adaptation for the *StackExchange* source-target domain pairs when using the *Direct* approach, *BM25*, and our adaptation model, measured with AUC(0.05).

4.2 Choosing the Model Components

Model Selection We select the best components for our domain adaptation model via experimentation on the *AskUbuntu–Android* domain pair. Then, we apply the model with the best-performing components across all domain pairs.

Hyperparameters We fine-tune the hyperparameters of all models on the development set for the target domain.

Similarity Function Table 3 shows the AUC at 0.05 and 0.1 for different models of question similarity, training on *AskUbuntu* and testing on *Android*. The first row shows that using cosine similarity with a hinge loss yields much better results than using a cross-entropy loss. This is likely because (*i*) there are some duplicate question pairs that were not tagged as such and that have come up as negative pairs in our training set, and the hinge loss deals with such outliers better. (*ii*) The cosine similarity is domain-invariant, while the weights of the feed-forward network of the softmax layers capture source-domain features.

Domain Adaptation Component We can see that the Wasserstein and the classification-based methods perform very similarly, after proper hyper-parameter tuning. However, Wasserstein yields better stability, achieving an AUC variance 17 times lower than the one for classification across hyper-parameter settings. Thus, we chose it for all experiments in the following subsections.

---

[3] For illustration, say of 100 candidates, 2 false negatives are ranked higher than the correct pair, the AUC score drops by 3 points (linear drop), as compared to the 66.67 point drop for MRR. We can avoid the expensive manually tagging of the negative pairs for experiments by using the AUC score.

Source	**Target**	**Direct**	**BM25**	**Adv.**
Sprint	AskUbuntu	0.615	**0.644**	0.615
	Apple	0.719	**0.747**	0.728
	Android	0.627	**0.681**	0.648
	Sprint	0.977	0.956	–
	SuperUser	0.795	0.765	**0.795**
Quora	AskUbuntu	0.446	**0.644**	0.446
	Apple	0.543	**0.747**	0.543
	Android	0.443	**0.681**	0.460
	Sprint	0.786	**0.956**	0.794
	SuperUser	0.624	**0.765**	0.649

Table 5: Domain adaptation results when using *Sprint* and *Quora* as the source domains with the *Direct* approach, *BM25*, and our adaptation model, measured with AUC(0.05).

4.3 When Does Adaptation Work Well?

Tables 4 and 5 study the impact of domain adaptation when applied to various source-target domain pairs, using the *Direct* approach, *BM25*, and our adaptation model. We can make the following observations:

- For almost all source–target domain pairs from the *StackExchange* family, domain adaptation improves over both baselines, with an average relative improvement of 5.6%. This improvement goes up to 14% for the *AskUbuntu–Android* source–target domain pair.
- Domain adaptation on the *Sprint* dataset performs better than direct transfer, but it is still worse than *BM25*.
- Domain adaptation from *Quora* performs the worst, with almost no improvement over direct transfer, which is far behind *BM25*.
- The more similar the source and the target domains, the better our adaptation model performs.

Table 2 shows that *AskUbuntu* has high similarity to other *StackExchange* domains, lower similarity to *Sprint*, and even lower similarity to *Quora*. The Pearson coefficient (Myers et al., 2010) between the n-gram fractions and the domain adaptation effectiveness for unigrams, bigrams and trigrams is 0.57, 0.80 and 0.94, respectively, which corresponds to moderate-to-strong positive correlation. This gives insight into how simple statistics can predict the overall effectiveness of domain adaptation.

Pivot\Target	SuperUser	Apple	Android
SuperUser	0.911	0.827	0.678
Apple	0.900	0.855	**0.711**
Android	0.904	**0.843**	0.790
Quora	0.906	0.815	0.673
Direct	0.908	0.828	0.692

Table 6: AUC(0.05) of ADA to unseen domains, with *AskUbuntu* as a source.

4.4 Adapting to Unseen Domains

We also experiment with domain adaptation to a target domain that was not seen during training (even adversarial training). We do so by training to adapt to a pivot domain different from the target. Table 6 shows that this yields better AUC compared to *direct transfer* when using *Apple* and *Android* as the pivot/target domains. We hypothesize that this is due to *Apple* and *Android* being closely related technical forums for iOS and Android devices. This sheds some light on the generality of adversarial regularization.

5 Conclusion and Future Work

We have applied and analyzed adversarial methods for domain transfer for the task of duplicate question detection; to the best of our knowledge, this is the first such work. Our experiments suggest that (*i*) adversarial adaptation is rather effective between domains that are similar, and (*ii*) the effectiveness of adaptation is positively correlated with the n-gram similarity between the domains.

In future work, we plan to develop better methods for adversarial adaptation based on these observations. One idea is to try source-pivot-target transfer, similarly to the way this is done for machine translation (Wu and Wang, 2007). Another promising direction is to have an attention mechanism (Luong et al., 2015) for question similarity which can be adapted across domains.[4]

Acknowledgments

This research was carried out in collaboration between the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) and the Qatar Computing Research Institute (QCRI), HBKU.

---

[4] In our experiments, we found that using attention was lowering the adaptation performance. Upon analysis, we found that adversarial techniques alone were not enough to make the attention weights domain-invariant.

References

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, pages 497–511, San Diego, CA, USA.

Martín Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, Toulon, France.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, ICML '17, pages 214–223, Sydney, NSW, Australia.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad A. Al Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and question selection for question answering on Arabic and English fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 896–903, San Diego, CA, USA.

Daniele Bonadiman, Antonio Uva, and Alessandro Moschitti. 2017. Effective shared representations with multitask learning for community question answering. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '17, pages 726–732, Valencia, Spain.

Yunbo Cao, Huizhong Duan, Chin-Yew Lin, Yong Yu, and Hsiao-Wuen Hon. 2008. Recommending questions using the MDL-based tree cut model. In *Proceedings of the International Conference on World Wide Web*, WWW '08, pages 81–90, Beijing, China.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Giovanni Da San Martino, Alberto Barrón-Cedeño, Salvatore Romeo, Antonio Uva, and Alessandro Moschitti. 2016. Learning to re-rank questions in community question answering using advanced features. In *Proceedings of the ACM Conference on Information and Knowledge Management*, CIKM '16, pages 1997–2000, Indianapolis, IN, USA.

Giovanni Da San Martino, Salvatore Romeo, Alberto Barroón-Cedeño, Shafiq Joty, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2017. Cross-language question re-ranking. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1145–1148, Tokyo, Japan.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 task 3: Learning semantic relations between questions and answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 1116–1123, San Diego, CA, USA.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2017. KeLP at SemEval-2017 task 3: Learning pairwise patterns in community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, pages 327–334, Vancouver, Canada.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the International Conference on Machine Learning*, ICML '15, pages 1180–1189, Lille, France.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030.

Francisco Guzmán, Lluís Màrquez, and Preslav Nakov. 2016a. Machine translation evaluation meets community question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 460–466, Berlin, Germany.

Francisco Guzmán, Lluís Màrquez, and Preslav Nakov. 2016b. MTE-NN at SemEval-2016 Task 3: Can machine translation evaluation help community question answering? In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 887–895, San Diego, CA, USA.

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

Doris Hoogeveen, Karin Verspoor, and Timothy Baldwin. 2016. CQADupStack: Gold or silver? In *Proceedings of the SIGIR Workshop on Web Question Answering Beyond Factoids*, WebQA '16, Pisa, Italy.

Doris Hoogeveen, Karin M. Verspoor, and Timothy Baldwin. 2015. CQADupStack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian Document Computing Symposium*, ADCS '15, pages 3:1–3:8, Parramatta, NSW, Australia.

Doris Hoogeveen, Li Wang, Timothy Baldwin, and Karin M. Verspoor. 2018. Web forum retrieval and text analytics: A survey. *Foundations and Trends in Information Retrieval*, 12(1):1–163.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the ACM Conference on Information and Knowledge Management*, CIKM '05, pages 84–90, Bremen, Germany.

Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2018. Joint multitask learning for community question answering using task-specific embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, Brussels, Belgium.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language learning with adversarial neural networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, CoNLL '17, pages 226–237, Vancouver, Canada.

Tuan Manh Lai, Trung Bui, and Sheng Li. 2018. A review on deep learning techniques applied to answer selection. In *Proceedings of the International Conference on Computational Linguistics*, COLING '18, pages 2132–2144, Santa Fe, NM, USA.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi S. Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez. 2016. Semi-supervised question retrieval with gated convolutions. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '16, pages 1279–1289, San Diego, CA, USA.

Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 2237–2243, Melbourne, Australia.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL '17, pages 1–10, Vancouver, Canada.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP '15, pages 1412–1421, Lisbon, Portugal.

Michael McCandless, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in Action: Covers Apache Lucene 3.0*. Manning Publications Co.

Jerome L Myers, Arnold Well, and Robert Frederick Lorch. 2010. *Research design and statistical analysis*. Routledge.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 task 3: Community question answering. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '17, pages 525–545, Vancouver, Canada.

Preslav Nakov, Lluís Màrquez, and Francisco Guzmán. 2016a. It takes three to tango: Triangulation approach to answer ranking in community question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '16, pages 1586–1597, Austin, TX, USA.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016b. SemEval-2016 task 3: Community question answering. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, pages 525–545, San Diego, CA, USA.

Massimo Nicosia and Alessandro Moschitti. 2017. Accurate sentence matching with hybrid Siamese networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 2235–2238, Singapore.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1532–1543, Doha, Qatar.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Salvatore Romeo, Giovanni Da San Martino, Alberto Barrón-Cedeño, Alessandro Moschitti, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Mitra Mohtarami, and James Glass. 2016. Neural attention for learning to rank questions in community question answering. In *Proceedings of the 26th International Conference on Computational Linguistics*, COLING '16, pages 1734–1745, Osaka, Japan.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 694–699, Beijing, China.

Antonio Uva, Daniele Bonadiman, and Alessandro Moschitti. 2018. Injecting relational structural representation in neural networks for question similarity. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 285–291, Melbourne, Australia.

Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based QA services. In *Proceedings of the 32nd International ACM SIGIR*

Conference on Research and Development in Information Retrieval, SIGIR '09, pages 187–194, Boston, MA, USA.

Pengwei Wang, Lei Ji, Jun Yan, Dejing Dou, Nisansa De Silva, Yong Zhang, and Lianwen Jin. 2018. Concept and attention-based CNN for question retrieval in multi-view learning. *ACM Trans. Intell. Syst. Technol.*, 9(4):41:1–41:24.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181.

Jianfei Yu, Minghui Qiu, Jing Jiang, Jun Huang, Shuangyong Song, Wei Chu, and Haiqing Chen. 2018. Modelling domain relationships for transfer learning on retrieval-based question answering systems in e-commerce. In *Proceedings of the Conference on Web Search and Data Mining*, WSDM '18, pages 682–690, Marina Del Rey, CA, USA.

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question retrieval with high quality answers in community question answering. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, CIKM '14, pages 371–380, Shanghai, China.

Minghua Zhang and Yunfang Wu. 2018. An unsupervised model with attention autoencoders for question retrieval. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI '18, New Orleans, LA, USA.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics*, 5:515–528.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, ACL '11, pages 653–662, Portland, OR, USA.

Translating a Math Word Problem to a Expression Tree

Lei Wang[1]*, **Yan Wang**[2], **Deng Cai**[23]*, **Dongxiang Zhang**[1], **Xiaojiang Liu**[2]
[1]Center for Future Media and School of Computer Science & Engineering, UESTC, [2]Tencent AI Lab
[3]Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong
demolei@outlook.com, {bradenwang, kieranliu}@tencent.com,
thisisjcykcd@gmail.com, zhangdo@uestc.edu.cn

Abstract

Sequence-to-sequence (SEQ2SEQ) models have been successfully applied to automatic math word problem solving. Despite its simplicity, a drawback still remains: a math word problem can be correctly solved by more than one equations. This non-deterministic transduction harms the performance of maximum likelihood estimation. In this paper, by considering the uniqueness of expression tree, we propose an equation normalization method to normalize the duplicated equations. Moreover, we analyze the performance of three popular SEQ2SEQ models on the math word problem solving. We find that each model has its own specialty in solving problems, consequently an ensemble model is then proposed to combine their advantages. Experiments on dataset Math23K show that the ensemble model with equation normalization significantly outperforms the previous state-of-the-art methods.

1 Introduction

Developing computer systems to automatically solve math word problems (MWPs) has been an interest of NLP researchers since 1963 (Feigenbaum et al., 1963; Bobrow, 1964). A typical MWP is shown in Table 1. Readers are asked to infer how many pens and pencils Jessica have in total, based on the textual problem description provided. Statistical machine learning-based methods (Kushman et al., 2014; Amnueypornsakul and Bhat, 2014; Zhou et al., 2015; Mitra and Baral, 2016; Roy and Roth, 2018) and semantic parsing-based methods (Shi et al., 2015; Koncel-Kedziorski et al., 2015; Roy and Roth, 2015; Huang et al., 2017) are proposed to tackle this problem, yet they still require considerable manual efforts on feature or template designing. For more literatures about solving math word problems automatically, refer to a recent survey paper Zhang et al. (2018).

Problem: Dan has 5 pens and 3 pencils, Jessica has 4 more pens and 2 less pencils than him. How many pens and pencils does Jessica have in total?
Equation: $x = 5 + 4 + 3 - 2$; **Solution:** 10

Table 1: A math word problem

Recently, the Deep Neural Networks (DNNs) have opened a new direction towards automatic MWP solving. Ling et al. (2017) take multiple-choice problems as input and automatically generate rationale text and the final choice. Wang et al. (2018) then make the first attempt of applying deep reinforcement learning to arithmetic word problem solving. Wang et al. (2017) train a deep neural solver (DNS) that needs no hand-crafted features, using the SEQ2SEQ model to automatically learn the problem-to-equation mapping.

Although promising results have been reported, the model in (Wang et al., 2017) still suffers from an equation duplication problem: a MWP can be solved by multiple equations. Taking the problem in Table 1 as an example, it can be solved by various equations such as $x = 5 + 4 + 3 - 2$, $x = 4 + (5 - 2) + 3$ and $x = 5 - 2 + 3 + 4$. This duplication problem results in a non-deterministic output space, which has a negative impact on the performance of most data-driven methods. In this paper, by considering the uniqueness of expression tree, we propose an equation normalization method to solve this problem.

Given the success of different SEQ2SEQ models on machine translation (such as recurrent encoder-decoder (Wu et al., 2016), Convolutional SEQ2SEQ model (Gehring et al., 2017) and Trans-

---

[*] The work was done when Lei Wang and Deng Cai were interns at Tencent AI Lab.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1064–1069
Brussels, Belgium, October 31 - November 4, 2018.

former (Vaswani et al., 2017)), it is promising to adapt them to MWP solving. In this paper, we compare the performance of three state-of-the-art SEQ2SEQ models on MWP solving. We observe that different models are able to correctly solve different MWPs, therefore, as a matter of course, an ensemble model is proposed to achieve higher performance. Experiments on dataset Math23K show that by adopting the equation normalization and model ensemble techniques, the accuracy boosts from 60.7% to 68.4%.

The remaining part of this paper is organized as follows: we first introduce the SEQ2SEQ Framework in Section 2. Then the equation normalization process is presented in Section 3, following which three SEQ2SEQ models and an ensemble model are applied to MWP solving in Section 4. The experimental results are presented in Section 5. Finally we conclude this paper in Section 6.

2 SEQ2SEQ Framework

The process of using SEQ2SEQ model to solve MWPs can be divided into two stages (Wang et al., 2017). In the first stage (number mapping stage), significant numbers (numbers that will be used in real calculation) in problem P are mapped to a list of number tokens $\{n_1, \ldots, n_m\}$ by their natural order in the problem text. Throughout this paper, we use the significant number identification (SNI) module proposed in (Wang et al., 2017) to identify whether a number is significant. In the second stage, SEQ2SEQ models can be trained by taking the problem text as the source sequence and equation templates (equations after number mapping) as the target sequence.

Taking the problem P in Table 1 as an example, first we can obtain a number mapping $M : \{n_1 = 5; \quad n_2 = 3; \quad n_3 = 4; \quad n_4 = 2;\}$, and transform the given equation $E_P : x = 5+4+3-2$ to an equation template $T_P : x = n_1+n_3+n_2-n_4$. During training, the objective of our SEQ2SEQ model is to maximize the conditional probability $P(T_p|P)$, which will be decomposed to token-wise probabilities. During decoding, we use beam search to approximate the most likely equation template. After that, we replace the number tokens with actual numbers and calculate the solution S with a math solver.

3 Equation Normalization

In the number mapping stage, the equations E_P have been successfully transformed to equation templates T_P. However, due to the equation duplication problem introduced in Section 1, this problem-equation templates formalization is a non-deterministic transduction that will have adverse effects on the performance of maximum likelihood estimation. There are two types of equation duplication: 1) order duplication such as "$n1 + n3 + n2$" and "$n1 + n2 + n3$", 2) bracket duplication such as "$n_1 + n_3 - n_2$" and "$n1 + (n_3 - n_2)$".

To normalize the order-duplicated templates, we define two normalization rules:

- Rule 1: Two duplicated equation templates with unequal length should be normalized to the shorter one. For example, two equation templates "$n_1 + n_2 + n_3 + n_3 - n_3$", "$n_1 + n_2 + n_3$" should be normalized to the latter one.

- Rule 2: The number tokens in equation templates should be ordered as close as possible to their order in number mapping. For example, three equation templates "$n_1+n_3+n_2$", "$n_1 + n_2 + n_3$" and "$n_3 + n_1 + n_2$" should be normalized to "$n_1 + n_2 + n_3$".

To solve the bracket duplication problem, we further normalize the equation templates to an expression tree. Every inner node in the expression tree is an operator with two children, while each leaf node is expected to be a number token. An example of expressing equation template $n_1 + n_2 + n_3 - n_4$ as the unique expression tree is shown in Figure 1.

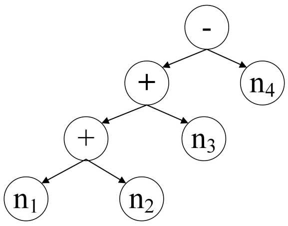

Figure 1: A Unique Expression Tree

After equation normalization, the SEQ2SEQ models can solve MWPs by taking problem text as source sequence and the postorder traversal of an unique expression tree as target sequence, as shown in Figure 2.

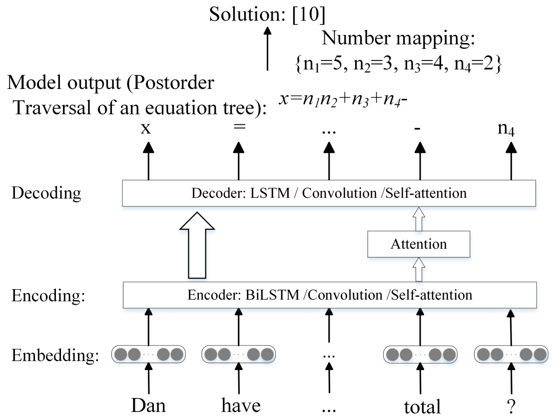

Model input (Problem text after number mapping): Dan has n_1 pens and n_2 pencils, Jessica has n_3 more pens and n_4 less pencils than him. How many pens and pencils does Jessica have in total?

Figure 2: Framework of SEQ2SEQ models

4 Models

In this section, we present three types of SEQ2SEQ models to solve MWPs: bidirectional Long Short Term Memory network (BiLSTM) (Wu et al., 2016), Convolutional SEQ2SEQ model (Gehring et al., 2017), and Transformer (Vaswani et al., 2017). To benefit the output accuracy with all three architectures, we propose to use a simple ensemble method.

4.1 BiLSTM

The BiLSTM model uses two LSTMs (forward and backward) to learn the representation of each token in the sequence based on both the past and the future context of the token. At each time step of decoding, the deocder uses a global attention mechanism to read those representations.

In more detail, we use two-layer Bi-LSTM cells with 256 hidden units as encoder, and two layers LSTM cells with 512 hidden units as decoder. In addition, we use Adam optimizer with learning rate $1e^{-3}$, $\beta_1 = 0.9$, and $\beta_2 = 0.99$. The epochs, minibatch size, and dropout rate are set to 100, 64, and 0.5, respectively.

4.2 ConvS2S

ConvS2S (Gehring et al., 2017) uses a convolutional architecture instead of RNNs. Both encoder and decoder share the same convolutional structure that uses n kernels striding from one side to the other, and uses gate linear units as nonlinearity activations over the output of convolution.

Our ConvS2S model adopts a four layers encoder and a three layers decoder, both using kernels of width 3 and hidden size 256. We adopt early stopping and learning rate annealing and set max-epochs equals to 100.

4.3 Transformer

Vaswani et al. (2017) proposed the Transformer based on an attention mechanism without relying on any convolutional or recurrent architecture. Both encoder and decoder are composed of a stack of identical layers. Each layer contains two parts: a multi-head self-attention module and a position-wise fully-connected feed-forward network.

Our transformer is four layers deep, with $n_{head} = 16$, $d_k = 12$, $d_v = 32$, and $d_{model} = 512$, where n_{head} is the number of heads of its self-attention, d_k is the dimension of keys, d_v is the dimension of values, and d_{model} is the output dimension of each sub-layer. In addition, we use Adam optimizer with learning rate $1e^{-9}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and dropout rate of 0.3.

4.4 Ensemble Model

Through careful observation (detailed in Section 5.2), we find that each model has a speciality in solving problems. Therefore, we propose an ensemble model which selects the result according to models' generation probability:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t | y_{<t}, \mathbf{x})$$

where $\mathbf{y} = \{y_1, ..., y_T\}$ is the target sequence, and $\mathbf{x} = \{x_1, ..., x_S\}$ is the source sequence. Finally, the output of the model with the highest generation probability is selected as the final output.

5 Experiment

In this section, we conduct experiments on dataset Math23K to examine the performance of different SEQ2SEQ models. Our main experimental result is to show a significant improvement over the baseline methods. We further conduct a case study to analyze why different SEQ2SEQ models can solve different kinds of MWPs.

Dataset: Math23K[1] collected by Wang et al. (2017) contains 23,162 labeled MWPs. All these

---

[1] `https://ai.tencent.com/ailab/Deep_Neural_Solver_for_Math_Word_Problems.html`

	Acc w/o EN (%)	Acc w/ EN (%)
DNS	58.1	60.7
Bi-LSTM	59.6	66.7
ConvS2S	61.5	64.2
Transformer	59.0	62.3
Ensemble	66.4	**68.4**

Table 2: Model comparison. EN is short for equation normalization

	Bi-LSTM	Transformer	ConvS2S
w/o EN	59.6	59.0	61.5
+ SE	63.1	59.9	62.2
+ OE	63.7	60.7	62.9
+ EB	65.3	61.2	62.9

Table 3: The ablation study of three equation normalization methods. SE is the first Rule mentioned in Section 3. OE is the second rule mentioned in Section 3. EB means eliminating the brackets.

problems are linear algebra questions with only one unknown variable.

Baselines: We compare our methods with two baselines: DNS and DNS-Hybrid. Both of them are proposed in (Wang et al., 2017), with state-of-the-art performance on dataset Math23K. The DNS is a vanilla SEQ2SEQ model that adopts GRU (Chung et al., 2014) as encoder and LSTM as decoder. The DNS-Hybrid is a hybrid model that combines DNS and a retrieval-based solver to achieve better performance.

5.1 Results

In experiments, we use the testing set in Math23K as the test set, and randomly split 1, 000 problems from the training set as validation set. Evaluation results are summarized in Table 2. First, to examine the effectiveness of equation normalization, model performance with and without equation normalization are compared. Then the performance of DNS, DNS-Hybrid, Bi-LSTM, ConvS2S, Transformer, and Ensemble model are examined on the dataset.

Several observations can be made from the results. First, the equation normalization process significantly improves the performance of each model. The accuracy of different models gain increases from 2.7% to 7.1% after equation normalization. Second, Bi-LSTM, ConvS2S, Transformer can achieve much higher performance than DNS, which means that popular machine translation models are also efficient in automatic MWP solving. Third, by combining the SEQ2SEQ models, our ensemble model gains additional 1.7% increase on accuracy.

In addition, we have further conducted three extra experiments to disentangle the benefits of three different EN techniques. Table 3 gives the details of the ablation study of the three SEQ2SEQ models. Taking Bi-LSTM as an example, accuracies of rule 1 (SE), rule 2 (OE) and eliminating brackets (EB) are 63.1%, 63.7% and 65.3%, respectively. Obviously, the performance of SEQ2ESQ models benefits from the equation normalization technologies.

5.2 Case Study

Further, we conduct a case analysis on the capability of different SEQ2SEQ models and provide three examples in Table 4. Our analysis is summarized as follows: 1) Transformer occasionally generates mathematically incorrect templates, while Bi-LSTM and ConvS2S almost do not, as shown in Example 1. This is probably because the size of training data is still not enough to train the multi-head self-attention structures; 2) In Example 2, the Transformer is adapted to solve problems that require complex inference. It is mainly because different heads in a self-attention structure can model various types of relationships between number tokens; 3) The multi-layer convolutional block structure in ConvS2S can properly process the context information of number tokens. In Example 3, it is the only one that captures the relationship between stamp A and stamp B.

6 Conclusion

In this paper, we first propose an equation normalization method that normalizes duplicated equation templates to an expression tree. We test different SEQ2SEQ models on MWP solving and propose an ensemble model to achieve higher performance. Experimental results demonstrate that the proposed equation normalization method and the ensemble model can significantly improve the state-of-the-art methods.

Acknowledgement

This work is supported in part by the National Nature Science Foundation of China under grants No. 61602087, and the Fundamental Research Funds for the Central Universities under grants No. ZYGX2016J080.

Example 1: Two biological groups have produced 690 (n_1) butterfly specimens in 15 (n_2) days. The first group produced 20 (n_3) each day. How many did the second group produced each day?
Bi-LSTM: $n_1 n_2 / n_3 -$; (correct) **ConvS2S:** $n_2 n_3 + n_1 *$; (error) **Transformer:** $n_2 n_1 n_3 n_3 +$; (error)

Example 2: A plane, in a speed of 500 (n_1) km/h, costs 3 (n_2) hours traveling from city A to city B. It only costs 2 (n_3) hours for return. How much is the average speed of the plane during this round-trip?
Bi-LSTM: $n_1 n_2 * n_3 *$; (error) **ConvS2S:** $11 + 1 n_1 / 1 n_2 / + /$; (error) **Transformer:** $n_1 n_2 * n_3 * n_2 n_3 + /$; (correct)

Example 3: Stamp A is 2 (n_1) paise denomination, and stamp B is 7 (n_2) paise denomination. If we are asked to buy 10 (n_3) of each, how much more does it cost to buy stamps A than to buy stamps B.
Bi-LSTM: $n_1 n_2 n_3 * -$; (error) **ConvS2S:** $n_1 n_3 * n_2 n_3 * -$; (correct) **Transformer:** $n_2 n_2 * n_2 n_3 * -$; (error)

Table 4: Three examples of solving MWP with SEQ2SEQ model. Note that the results are postorder traversal of expression trees, and the problems are translated to English for brevity.

References

Bussaba Amnueypornsakul and Suma Bhat. 2014. Machine-guided solution to mathematical word problems. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation, PACLIC 28, Cape Panwa Hotel, Phuket, Thailand, December 12-14, 2014*, pages 111–119.

Daniel G Bobrow. 1964. Natural language input for a computer problem solving system.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Edward A Feigenbaum, Julian Feldman, et al. 1963. *Computers and thought*. New York.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 805–814. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *TACL*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167. Association for Computational Linguistics.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1743–1752.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. *TACL*, 6:159–172.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1132–1142.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. Mathdqn: Solving arithmeticword problems via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Dongxiang Zhang, Lei Wang, Nuo Xu, Bing Tian Dai, and Heng Tao Shen. 2018. The gap of semantic parsing: A survey on automatic math word problem solvers. *arXiv preprint arXiv:1808.07290*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 817–822.

Semantic Linking in Convolutional Neural Networks for Answer Sentence Selection

Massimo Nicosia[*] and **Alessandro Moschitti**[†]
[*]DISI, University of Trento, 38123 Povo (TN), Italy
Amazon, Manhattan Beach, CA, USA, 90266
`m.nicosia@gmail.com` `amosch@amazon.com`

Abstract

State-of-the-art networks that model relations between two pieces of text often use complex architectures and attention. In this paper, instead of focusing on architecture engineering, we take advantage of small amounts of labelled data that model semantic phenomena in text to encode matching features directly in the word representations. This greatly boosts the accuracy of our reference network, while keeping the model simple and fast to train. Our approach also beats a tree kernel model that uses similar input encodings, and neural models which use advanced attention and compare-aggregate mechanisms.

1 Introduction

Modeling a match between pieces of text is at the core of many NLP tasks. Recently, manual feature engineering methods have been shadowed by neural network approaches. These networks model the interaction of two pieces of text, or word-to-word interactions across sentences, using sophisticated attention mechanisms (Wang et al., 2016a; Santos et al., 2016) and compare-aggregate frameworks (He and Lin, 2016; Wang et al., 2017).

Architectural complexity is tied to longer training times [1]. Meaningful features may take long time to emerge by only leveraging word representations and the training data of the task at hand. This is especially problematic with little data, as it often happens in question answering (QA) tasks, e.g., answer sentence selection (Wang et al., 2007; Yang et al., 2015). Thus, effective word representations are crucial in neural network models to get state-of-the-art performance.

In this work, we try to answer the following research questions: (i) in addition to lexical links, can we incorporate higher-level semantic links between the words in a question and a candidate answer passage, and (ii) can we show that such information has an impact on the quality of our model, and also allows us to keep the architecture simple?

We show that modeling semantic relations improves the performance of a neural network for answer sentence selection with (i) a little number of semantic annotations, and (ii) a little increase in training time w.r.t. more complex architecture.

2 Related Work

Traditional work on QA makes heavily use of syntactic and semantic features (Hickl et al., 2007; Ferrucci et al., 2010). A different direction consists in using structural kernels on text encoded as trees (Severyn and Moschitti, 2012; Severyn et al., 2013a,b; Tymoshenko et al., 2014; Tymoshenko and Moschitti, 2015). Recently, deep learning methods have been very successful in NLP tasks. Words and sentences are mapped into low dimensional representations using convolutional (Krizhevsky et al., 2012) and recurrent networks (Schuster and Paliwal, 1997), and then adoperated for classification. Complex networks for such a task include attentive networks and compare-aggregate networks.

Attentive Networks (Bahdanau et al., 2015; Parikh et al., 2016; Yin et al., 2016) build a sentence representation by also considering the other sentence, weighting the contribution of its parts with the so-called attention mechanism.

Compare-Aggregate Networks (Wang and Jiang, 2017) apply several decompositions to each sentence in a pair. The resulting vectors are compared or composed with multiple functions, and possibly some attention mechanisms. All the

---

[*] Now at Google

[†] This work was partially carried out when the author was at the University of Trento

[1] `http://dawn.cs.stanford.edu/benchmark/`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1070–1076
Brussels, Belgium, October 31 - November 4, 2018.

intermediate results are then aggregated into a fixed size vector to quantify the final match.

In this work, we take some elements of the traditional QA research, i.e., semantic features, and use them to model relationships between sentence pairs, in the context of a neural network, which is less complex than attentive and compare-aggregate counterparts.

3 Question Analysis

Question Analysis is an important part of a QA system (Lally et al., 2012) and can give us syntactic and semantic clues that greatly help in scoring answer passages, and in identifying the final answer. Leveraging a relatively small number of annotated examples, we can automatically extract question properties that may be exploited by a QA model to increase the accuracy of its answers. We use classifiers to extract the *question category* and the *question focus*.

Question Category. Questions can be broadly classified into categories according to a given taxonomy. When the category is indicative of the answer type, the latter can be furtherly characterized by the Lexical Answer Type (LAT), which according to Lally et al. (2012) is a word or noun phrase in the question that specifies the type of the answer without any attempt to understand its semantics.

Question Focus. In the literature there are multiple definitions of question focus. According to Ferrucci et al. (2010), the focus is the question part that substituted with the answer, renders the question a stand-alone statement. According to Bunescu and Huang (2010), the focus is the "set of all maximal noun phrases in the question that corefer with the answer". Their definition allows a question to have multiple focuses or an implicit focus. Additionally, it is more tied to the LAT and indeed the focus can be used to infer the answer type. We adopt such definition since we build our question focus identifier using the annotated data they provide. Note that we do not consider multi-word or implicit focus.

4 Answer Sentence Selection with CNNs

Given a query or question q and a candidate answer passage a, the task of answer selection can be defined as learning a function $f(q, a)$ that outputs a relevancy probability $s \in [0, 1]$. Multiple answers associated with a question are sorted in descending order by the score s. A good answer selection system places the highest number of correct answers at the top of a candidate answer list. In this paper, we use convolutional neural networks, referred to as CNNs (Kim, 2014; Kalchbrenner et al., 2014), to (i) classify a question into a category, (ii) identify the focus word in a question, and (iii) build a question and answer representations for QA.

4.1 Sentence Matrix Encoding

A sentence s of length n is a sequence of words (w_1, ..., w_n), which are drawn from a vocabulary V. Each word is encoded with an integer id from 1 to $|V|$, and then represented as a vector, $\mathbf{w} \in \mathbb{R}^d$, looked up into an embedding matrix, $\mathbf{E} \in \mathbb{R}^{d \times |V|}$. The matrix $\mathbf{E}$ is obtained by concatenating all the embeddings of the words in V. The id 0 is used for padding and it is mapped to the zero vector. The i^{th} column in $\mathbf{E}$ corresponds to the word with integer id i to facilitate the lookup.

4.2 Question Analysis Networks

We use CNNs for question analysis. The **question category network** applies convolutions of different width and then pooling on the question. The results are concatenated and fed to a multilayer perceptron (MLP) that outputs a probability distribution over the possible categories seen during training. The **question focus network** applies convolutions that operate on windows centered on each question word. Therefore, the input and output resolutions are the same. We stack a number of convolutions to increase the receptive field. Every output vector from the last convolution of the stack is passed through an MLP, which produces a scalar value. All those values are normalized across each sentence with a softmax, to form a probability distribution over the sentence tokens.

4.3 Answer Sentence Selection Network

Our neural model is based on the Severyn and Moschitti (2015, 2016) model (S&M from now on), showed in Figure 1. This model is simple, fast and well studied. It has also been reproduced in other work (Rao et al., 2017; Chen et al., 2017; Sequiera et al., 2017).

The S&M model embeds the question and answer passage and operates independent convolutional and max-pooling layers on each. A bilinear transformation (Bordes et al., 2014) produces a similarity value x_{sim} for the pair. The similarity, the encoded question and passage, and a vec-

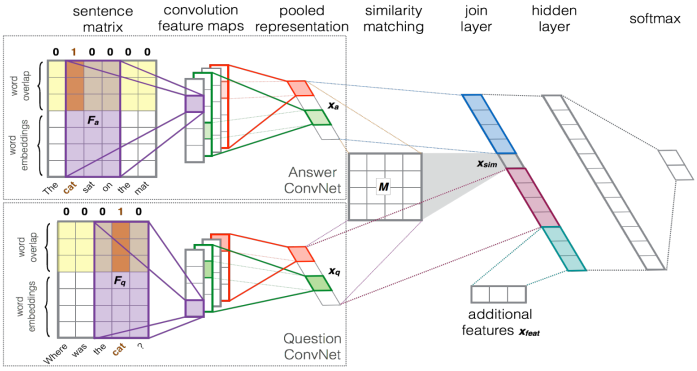

Figure 1: The S&M CNN model. Diagram from Severyn and Moschitti (2016).

Cat.	Named Entity Type
HUM	Person
LOC	Loc, Gpe
NUM	Date, Time, Percent, Quantity, Ordinal, Cardinal
ENTY	Norp, Org, Facility, Product, Event, Work of art, Law, Language
DESC	Norp, Org, Facility, Product, Event, Work of art, Law, Language, Date, Time, Percent, Quantity, Ordinal, Cardinal
ABBR	Norp, Org, Facility, Product, Event, Work of art, Law, Language

Table 1: Mapping between question categories and OntoNotes entity types.

tor of real valued features x_{feat} are concatenated in the join layer. The latter is fed to a hidden layer with a non-linearity, and the final softmax layer outputs the matching probability. The word vectors of the question and the answer are augmented with an additional feature, which is embedded in a small dimensional space. This feature signals if a word appears in both the question and answer. We found that the real valued features and the similarity matrix do not increase the network accuracy and we removed them from our model. This finding is consistent with recent reproduction papers by Rao et al. (2017); Sequiera et al. (2017).

4.4 Our QA Network with Semantic Overlap

We propose to add semantic features to the sentence matrix to establish links between words that go beyond lexical matching. Figure 2 describes our network. The key addition to the S&M model is the semantic overlap vector. Each word is therefore represented by concatenating three vectors: the word embedding vector, a feature embedding vector which can represent two values – if a word is contained or not in both question and answer – and the semantic overlap embedding vector. The semantic vector $\mathbf{w}_{so}$, with dimensionality s, embeds a feature so which can assume $C+1$ values, if we consider the C question classes plus a no-match value. Each feature value is looked up into an embedding matrix $\mathbf{W}_{so} \in \mathbb{R}^{s \times |C|+1}$. Analogously, the word overlap binary feature is looked up into an embedding matrix $\mathbf{W}_{wo} \in \mathbb{R}^{r \times |2|}$. The final word representation will be the concatenation of all these vectors: $\mathbf{w}^{'} = [\mathbf{w}; \mathbf{w}_{wo}; \mathbf{w}_{so}]$.

Here we describe how the semantic word overlap feature is computed. For each question we collect the output of our question analysis CNNs. The question focus CNN determines which word in the question is the focus. The question category CNN assigns a class to the question. After that, each word is associated with a semantic overlap feature so (which will eventually be embedded using $\mathbf{W}_{wo}$) according to the following strategy:

1. for each word in the question which is not the question focus so is equal to 0. For the question focus word so is equal to the id of the question category (the question focus and category are output by our CNN classifiers);
2. for each answer word so is equal to 0, with the exeception of words covered by named entities (NEs), for which so is equal to the id of the question category that is compatible with their entity type, according to the mapping in Table 1.

The $\mathbf{W}_{wo}$ and $\mathbf{W}_{so}$ matrices are parameters of the model, and they are learned during training. The question category and question focus annotations for the QA datasets are produced by our neural network classifiers. The NEs are obtained with an off-the-shelf processor [2], trained on OntoNotes (Weischedel et al., 2012).

5 Experimental Results

Here we describe how we train our networks for question analysis and then we present the answer sentence selection experiments. More details about preprocessing, training and hyperparameter choice can be found in the appendix.

5.1 Question Classification

Dataset. The CNN question classifier is trained on the UIUC dataset (Li and Roth, 2006). We use the 6 coarse classes to train the classifier.

---

[2] In the future, we will also train the NE recognizer.

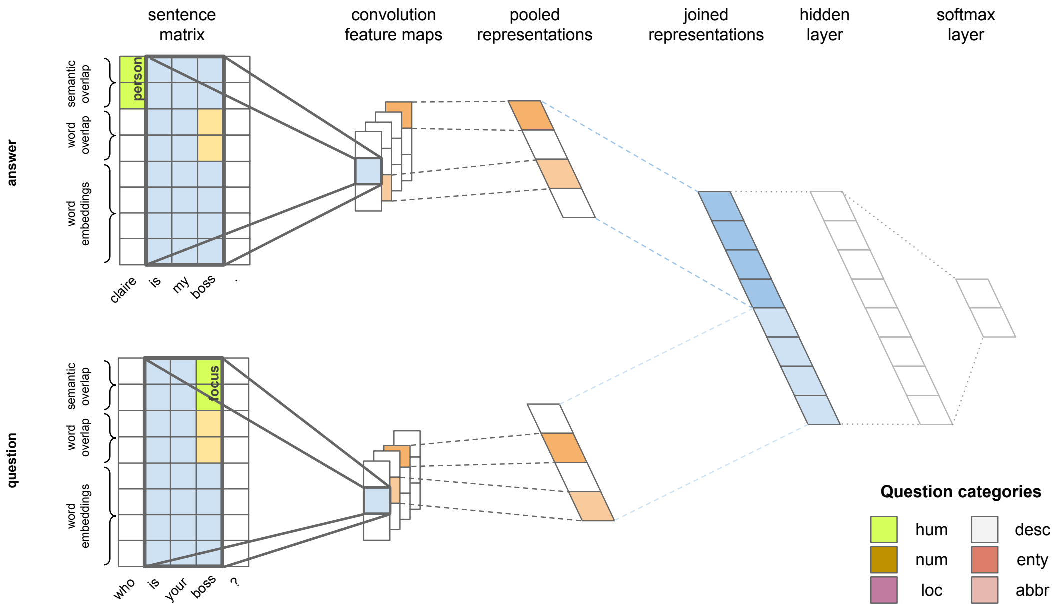

Figure 2: Our model with word and semantic overlap vectors. The convolution of size 3 is not padded, and the filters are 4. The semantic overlap vectors of the question focus word *boss*, and the answer word *claire* are the same, because the latter is an entity of type *Person*. The question has HUM category. Ignoring stopwords, the word *boss* appears in the question and the answer, and this is reflected in the word overlap embedding space.

Results. The classifier has accuracy of 91.2% on the UIUC test set. Our goal is to annotate new questions with reasonable accuracy. Since the model convergences well, we annotate the questions in the QA datasets after training on the UIUC data, and select the best model on the test data.

5.2 Question Focus Identification

Dataset. The CNN focus identifier is trained on the dataset from Bunescu and Huang (2010), which contains the first 2,000 UIUC questions annotated with focus information. After removing the questions with implicit and multi-focus, we end up with 1,030 questions.

Results. The cross-validation accuracy of the classifier is 92.3%. After convergence, we annotate the focus words in the QA datasets.

5.3 TrecQA

Dataset. We test our model on TrecQA (Wang et al., 2007), one of the most popular benchmarks for answer selection. The dataset contains factoid questions and candidate answer sentences.

System	MAP	MRR
Santos et al. (2016)	75.30	85.11
He and Lin (2016)	75.88	82.19
Severyn and Moschitti (2016)	76.54	81.86
Wang et al. (2016b)	77.14	84.47
Rao et al. (2016)	80.10	87.70
Wang et al. (2017)	80.20	87.50
Shen et al. (2017)	82.20	88.90
CNN$_{WO}$ TRAIN-ALL	76.49 (0.4)	84.22 (0.5)
CNN$_{WO+SO}$ TRAIN-ALL	77.93 (0.7)	84.89 (0.9)

Table 2: MAP and MRR (%) on the TrecQA Clean dataset.

We use the same splits of the original data, but we run our experiments using the larger provided training set (TRAIN-ALL). This is noisier data, which, on the other hand, gives us more examples for training. We remove from the dev. and test sets questions without answers, and questions with only correct or incorrect answer sentence candidates. The resulting dev. and test sets contain respectively 65 and 68 questions. This setting follows a standard in recent work on TREC-QA which is referred to as TrecQA Clean (See Shen et al. (2017)).

Results. Table 2 contains results from previous work, and the performance of our models. CNN$_{WO}$ is our variant of the S&M model. It has comparable performance in terms of MAP, but it is 2.4% points higher in MRR. Our model CNN$_{WO+SO}$ that integrates the semantic overlap improves over CNN$_{WO}$ by 1.44% points in MAP, and 0.67% points in MRR. It approaches the model by Rao et al. (2016), which uses a triplet ranking loss, and several strategies to build

System	MAP	MRR
Miao et al. (2016)	68.86	70.69
Yin et al. (2016)	69.21	71.08
Severyn and Moschitti (2016)	69.51	71.07
Chen et al. (2017)	70.10	71.80
Rao et al. (2016)	70.90	72.30
Tymoshenko et al. (2016)	71.25	72.30
Guo et al. (2017)	71.71	73.36
Wang et al. (2017)	71.80	73.10
Shen et al. (2017)	73.30	75.00
Wang et al. (2016a)	73.41	74.18
Wang and Jiang (2017)	74.33	75.40
CNN$_{WO}$	69.53 (0.5)	71.35 (0.5)
CNN$_{WO+SO}$	72.24 (0.5)	73.91 (0.5)

Table 3: MAP and MRR (%) on the WikiQA dataset.

training instances with difficult negative examples. Our system beats several others that use word alignments and attention mechanisms. The better systems employ expensive bidirectional networks, sophisticated attention mechanisms, and extract multiple views of questions and answers for comparing and aggregating them.

5.4 WikiQA

Dataset. TrecQA and its test set are small, so results may be unstable. In addition, lexical overlap between questions and answer candidates is high (Yih et al., 2013). This means that simple lexical similarity features are highly discriminative. Therefore, we also experiment with WikiQA (Yang et al., 2015), which is an order of magnitude larger than TrecQA. We use the Yin et al. (2016) experimental setting.

Results. Table 3 contains the results on WikiQA. Again the MAP score is comparable with the S&M model, while the MRR is slightly higher. Our model CNN$_{WO+SO}$ improves over CNN$_{WO}$ by 2.71% points in MAP, and 2.56% points in MRR, with a higher margin with respect to TrecQA. Interestingly, our approach improves upon the Tymoshenko et al. (2016) tree-kernel model by 1 MAP point. This model includes relational information in terms of question focus, entities and question categories too, but uses additional syntactic information (i.e., syntactic trees). Our network is able to make better use of the provided semantic clues. Surprisingly, CNN$_{WO+SO}$ also achieves higher MAP than the model by Wang et al. (2017), which is a state-of-the-art complex approach mixing attention and interaction factors of multiple sentence perspectives.

5.5 Discussion

The results with the CNN$_{WO+SO}$ model suggest that the semantic overlap vectors are an effective way of linking questions and answers. This is especially true, given the results on WikiQA, where the questions and answers have little lexical overlap. With the additional semantic information, the CNN is able to better model the relevancy of candidate passages. It also surpasses the accuracy of more complex systems, which have higher training time. The annotation networks (which can be trained only once) and the answer selection networks take little time to train: from 10 to 20 minutes in total, depending on the number of question/answer pairs. CNNs are faster at training and inference time with respect to RNNs, especially when the latter incorporate attention mechanisms, which increase the number of computations. We argue that annotating a relatively small number of examples with semantic information, could be time well spent to increase model accuracy, without increasing its architectural complexity. We would like to add that we also experimented with RNNs (LSTM and GRU) in place of the CNN sentence model. Such encoders easily overfitted, requiring careful regularization, and did not yield better results for us.

6 Conclusion and Future Work

In this paper, we presented a neural network that models semantic links between questions and answers, in addition to lexical links. The annotations for establishing such links are produced by a set of fast neural components for question analysis, trained on publicly available datasets. The evaluation on two QA datasets shows that our approach can achieve state-of-the-art performance using a simple CNN, leading to a low complexity and training time. Our approach is an interesting first step towards a future architecture, in which we will jointly optimize the semantic annotators and the answer sentence selection model, in an end-to-end fashion.

Acknowledgments

The first author was supported by the Google Europe Doctoral Fellowship Award 2015. Many thanks to the anonymous reviewers, the Area and PC Chairs for their valuable work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Volume 8724*, ECML PKDD 2014, pages 165–180, New York, NY, USA. Springer-Verlag New York, Inc.

Razvan Bunescu and Yunfeng Huang. 2010. Towards a general model of answer typing: Question focus identification. In *CICLing*.

Ruey-Cheng Chen, Evi Yulianti, Mark Sanderson, and W. Bruce Croft. 2017. On the benefit of incorporating external features in a neural architecture for answer sentence selection. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1017–1020, New York, NY, USA. ACM.

David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79.

Jiahui Guo, Bin Yue, Guandong Xu, Zhenglu Yang, and Jin-Mao Wei. 2017. An enhanced convolutional neural network model for answer selection. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 789–790, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, San Diego, California. Association for Computational Linguistics.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Ying Shi, and Bryan Rink. 2007. Question answering with lcc's chaucer at trec 2006. In *Proceedings of the Text REtrieval Conference*, pages 283–292.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*.

A. Lally, JM Prager, MC McCord, BK Boguraev, S. Patwardhan, J. Fan, P. Fodor, and J. Chu-Carroll. 2012. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*, 56(3.4).

Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proc. of EMNLP*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1913–1916, New York, NY, USA. ACM.

Jinfeng Rao, Hua He, and Jimmy Lin. 2017. Experiments with convolutional neural network models for answer selection. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1217–1220. ACM.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11).

Royal Sequiera, Gaurav Baruah, Zhucheng Tu, Salman Mohammed, Jinfeng Rao, Haotian Zhang, and Jimmy J. Lin. 2017. Exploring the effectiveness of convolutional neural networks for answer selection in end-to-end question answering. In *SIGIR 2017 Workshop on Neural Information Retrieval (Neu-IR'17)*.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 741–750, New York, NY, USA. ACM.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. *CoRR*, abs/1604.01178.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013a. Building structures from classifiers for passage reranking. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, CIKM '13, pages 969–978, New York, NY, USA. ACM.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013b. Learning adaptable patterns for passage reranking. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 75–83, Sofia, Bulgaria. Association for Computational Linguistics.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189, Copenhagen, Denmark. Association for Computational Linguistics.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1268–1278, San Diego, California. Association for Computational Linguistics.

Kateryna Tymoshenko and Alessandro Moschitti. 2015. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1451–1460, New York, NY, USA. ACM.

Kateryna Tymoshenko, Alessandro Moschitti, and Aliaksei Severyn. 2014. Encoding semantic resources in syntactic structures for passage reranking. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–672, Gothenburg, Sweden. Association for Computational Linguistics.

Bingning Wang, Kang Liu, and Jun Zhao. 2016a. Inner attention based recurrent neural networks for answer selection. In *ACL*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic. Association for Computational Linguistics.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *Proc. of ICLR*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of IJCAI*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1340–1349. The COLING 2016 Organizing Committee.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, et al. 2012. Ontonotes release 4.0.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1744–1753, Sofia, Bulgaria. Association for Computational Linguistics.

Wenpeng Yin, Hinrich Schutze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

A dataset and baselines for sequential open-domain question answering

Ahmed Elgohary[*], Chen Zhao[*], Jordan Boyd-Graber
Department of Computer Science
University of Maryland, College Park
`{elgohary, chenz, jbg}@cs.umd.edu`

Abstract

Previous work on question-answering systems mainly focuses on answering individual questions, assuming they are independent and devoid of context. Instead, we investigate sequential question answering, asking multiple related questions. We present QBLink, a new dataset of fully human-authored questions. We extend existing strong question answering frameworks to include previous questions to improve the overall question-answering accuracy in open-domain question answering. The dataset is publicly available at `http://sequential.qanta.org`.

1 Introduction

The framework of combining information retrieval and neural reading comprehension has been the basis of several systems for answering open-domain questions over unstructured text (Chen et al., 2017; Wang et al., 2018; Clark and Gardner, 2018; Htut et al., 2018). Typically, such systems take one input question at a time, retrieving and ranking multiple paragraphs that potentially contain the answer. A reading comprehension model then produces a ranked list of candidate answer spans from each paragraph. The final answer is then selected from the produced spans.

In information-seeking dialogs, e.g., personal assistants, users interact with a question answering system by asking a sequence of related questions, where questions share the same predicate, entities, or at least a topic. Answering each question in isolation is sub-optimal as information from previously asked questions and previously obtained answers can help better answer the current question.

Lead-in: Only twenty-one million units in this system will ever be created. For 10 points each:
Question 1: Name this digital payment system whose transactions are recorded on a "block chain".
Answer: Bitcoin
Question 2: Bitcoin was invented by this person, who, according to a dubious Newsweek cover story, is a 64-year-old Japanese-American man who lives in California.
Answer: Satoshi Nakamoto
Question 3: This online drugs marketplace, Chris Borglum's one-time favorite, used bitcoins to conduct all of its transactions. It was started in 2011 by Ross Ulbricht using the pseudonym Dread Pirate Roberts.
Answer: Silk Road

Figure 1: An example sequence of questions from QBLink. The lead-in and question 1 are asking about the same object/answer. The subject of question 2 is the same as the object of question 1. All questions are about a narrow topic, Bitcoin.

We study the task of sequential open-domain question answering. We ask how a standard open-domain question answering system can incorporate connections between question-answer pairs in the same sequence. We introduce QBLink, a new dataset of about 18,000 question sequences (Figure 1), each sequence consists of three naturally occurring human-authored questions (totaling around 56,000 unique questions). The sequences themselves are also naturally occurring (i.e., we do not artificially combine individually-authored questions to form sequences), which allows us to focus more on the important connections between questions that should be incorporated to improve the end-to-end question answering accuracy.

We compare sequence-aware models to baselines that process each question separately. For our sequence-aware models, we tweak the retrieval component by incorporating previous questions and their answers together with the current question to better rank the retrieved paragraphs. For the reader, we use the semantic relations between entities in previous questions (or their corresponding answers) and entities mentioned in the paragraph

---

[*] The first two authors contributed equally.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1077–1083
Brussels, Belgium, October 31 - November 4, 2018.

being read (candidate answers) to better choose the answer entity. Both the retrieval and reading steps can be slightly improved by incorporating sequence information.

Our contributions are two-fold: first, we present a new dataset for sequential question answering. Our dataset is composed of complex questions about a variety of topics. We make the dataset publicly available to encourage future research. Second, we use our dataset to compare baselines in the open-domain question answering setup with the goal of showing that incorporating sequential connections between questions is advantageous.

2 Sequential Question Answering Task

We define the task of open-domain sequential question answering: given a document collection D and questions grouped into disjoint sequences $\{S_i \,|\, i = 1 \ldots n\}$ where each S_i is an ordered sequence of question, answer pairs, and a subset of documents $S_i = ((q_i^j,\ a_i^j,\ D_i^j) \,|\, j = 1 \ldots m)$, the task is to answer questions $q_i^{\hat{j}}$ with document evidences $D_i^{\hat{j}}$ given access to previously asked questions in the same sequence and their corresponding answers $\{(q_i^j,\ a_i^j) \,|\, j < \hat{j}\}$.

Following Chen et al. (2017), we split the task into two steps—a retrieval step and a reading step. In the retrieval step the current question $q_i^{\hat{j}}$ and previous questions and answers $\{(q_i^j,\ a_i^j) \,|\, j < \hat{j}\}$ are used to retrieve a ranked list of paragraphs $D_i^{\hat{j}}$ from D that are likely to contain the correct answer to the current question $q_i^{\hat{j}}$. The retrieved paragraphs $D_i^{\hat{j}}$ are the input to the reading step that selects a span from $D_i^{\hat{j}}$ as the answer to $q_i^{\hat{j}}$. The reading step has access to previous questions and answers $\{(q_i^j,\ a_i^j) \,|\, j < \hat{j}\}$ as well.

3 Dataset Construction

This section describes QBLink's construction. QBLink is based on the *bonus questions* of **Q**uiz **B**owl tournaments. Unlike previous work that only uses the starter (or tossup) questions (Boyd-Graber et al., 2012), bonus questions are not interruptable (players always hear the complete question) and have greater variability in difficulty. Bonus questions start with a lead-in text, which sets the stage for the rest of the question, followed by a sequence of related questions. Figure 1 shows an example of a sequence of three questions.

Num. Questions (Num. Sequences) × 3	
Training	45,747 (15,249)
Developme	3,630 (1,210)
Testing	6,555 (2,185)
Num. Sequences per Domain	
Current Events	240
Fine Arts	2,588
Geography	472
History	3,961
Literature	3,879
Mythology	758
Philosophy	692
Religion	746
Science	4,028
Social Science	827
Trash	453
Num. Questions Tokens	
Training	32.6 ± 9.6
Developme	33.5 ± 9.9
Testing	32.1 ± 10.24
Num. TagMe Entities	
Training	2.46 ± 1.68
Developme	2.49 ± 1.68
Testing	2.48 ± 1.77
Num. Unique Answers	43,597
Num. Unique Answer Pages	18,529

Table 1: Statistics about QBLink. Most questions are fairly long and contain 2.5 entity mentions in average which demonstrates the complexity of the questions.

Specifically, we collect bonus questions from `http://quizdb.org` for the tournaments in 2008–2018. Each question is categorized by topic as history, literature, science, geography, fine arts, philosophy, religion, mythology, social sciences, current events or trash. We filter out too short questions (fewer than ten tokens), and only keep questions with exactly three sub-questions.

We map the answers to unambiguous Wikipedia pages using combination of rule based matching and fuzzy string matching, then filter out the questions whose answers are not mapped to any Wikipedia page (12.5% of the questions).

To keep our development and test set intact and and of a reasonable percentage of questions, we use the questions in 2014 tournament (the year with the largest number of questions) for development and testing, and the rest of the questions are used for training. Table 1 shows the number of question sequences and questions per split as well as tokens and linked Wikipedia mentions per question. We use TagMe (Ferragina and Scaiella, 2010) for mention detection and linking question text to Wikipedia.

4 Baselines

We build our baselines on the DrQA framework of Chen et al. (2017) for open-domain question answering over Wikipedia.[1] The framework operates in a retrieval phase followed by a reading phase.

The retrieval phase uses a simple tf-idf (Salton and Buckley, 1987) ranking of Wikipedia articles given a question as the query.

The reading phase is a multi-layer recurrent neural network model that extracts an answer span from the top d retrieved paragraphs. The reader model computes a contextualized representation of each token $\mathbf{t_i}$ by running the token sequence through a multi-layer bidirectional long short-term memory network (BiLSTM) (Hochreiter and Schmidhuber, 1997) and taking the corresponding hidden state to each token at the top layer. The question itself is encoded in a vector $\mathbf{q}$ as a weighted average of the hidden states of a BiLSTM over the word embeddings of its individual tokens. The model then computes an unnormalized probability score of t_i as the start and end token of the answer span,

$$\begin{aligned} Start(i) &= \exp(\mathbf{t_i}^T \mathbf{W_{start}} \mathbf{q}); \\ End(i) &= \exp(\mathbf{t_i}^T \mathbf{W_{end}} \mathbf{q}). \end{aligned} \tag{1}$$

To find the answer in multiple paragraphs at test time, we merge all paragraphs before feeding them to the reader (Clark and Gardner, 2018).

4.1 Answering Question in Isolation

We experiment with three models that ignore the sequential connections between questions and answer each question in isolation. Our first model is a simple information retrieval (IR) baseline that only uses the retrieval component: the title of the top-1 Wikipedia article is predicted as the answer.

Our second baseline is the full DrQA baseline whose reader is trained/tuned on the training/development questions of our dataset. To assign paragraphs to each of the training questions, we follow a similar distant-supervision approach to Chen et al. (2017). We retrieve the top twenty Wikipedia articles for each question, exclude the paragraphs that do not contain the gold answer, and then rank the remaining paragraphs using tf-idf. Each of the top ten paragraphs is paired with the question to form a data instance for training the reader.

Finally, we tweak the DrQA reader to limit the candidate answer spans to entity mentions that are linked to Wikipedia. We set the pre-normalization start and end scores of spans that are not detected mentions to zero.

4.2 Incorporating Context in Retrieval

To incorporate the sequential connections between questions in the retrieval phase, we append the previously asked questions to the current question. We also compare appending the predicted answers (top-1 span) to each of the previous questions as well as the gold answers to the current question.

4.3 Incorporating Context in Reader

In addition to encoding which entities have appeared in previous questions, we also want to provide our models with *relationship* information. However, pre-defined relationships from knowledge bases tend to be brittle. Instead, we use a continuous representation of relationships (Iyyer et al., 2016). For example, suppose we want to encode the relationships for an entity (answer candidate) that starts at i and ends at j. We summarize that entities relationships from each of possible k relation-spans. A relation-span is a sequence of tokens from Wikipedia that contains both the answer candidate and an answer to a previous question (For example, the correct answer in Figure 2 has a relation-span *"He is best known for defending President Ronald Reagan during the assassination attempt by John Hinckley Jr."* with the previous answer *"Ronald Reagan"*). This is summarized in a vector $\mathbf{r}_{ij}$ by merging all k relation-spans in a single span that is then fed through a BiLSTM whose hidden states are combined as a weighted sum where the weights are computed with self-attention (Lin et al., 2017).

The stronger the similarity between the relation that the question is asking about and the relation-spans, the higher the score of the candidate answer should be. We estimate that similarity r by concatenating the elementwise absolute difference and hadamard product between $\mathbf{r}_{ij}$ and the question embedding $\mathbf{q}$. Then, we use a trainable weight vector $\mathbf{w_{rel}}$ to combine the components of the concatenation output and produce a single similarity score as

$$r = \mathbf{w_{rel}}^T [|\mathbf{q} - \mathbf{r_{ij}}|; \mathbf{q} \circ \mathbf{r_{ij}}].$$

This can then influence the final selection of the answer span by adding the relation similarity

---

[1] We use the Wikipedia dump of 2017–09–20.

Method	**EM**
Baselines: Questions in Isolation	
IR	15.6
DrQA	39.3
DrQA + limiting to entities	39.7
DrQA + Retrieval with context	
Previous questions	36.4
Previous predicted answers	39.8
Previous gold answers	40.1
DrQA + Reading with context	
Append relation descriptions w/ predicted answers	40.2
Append relation descriptions w/ gold answers	40.7
Explicit relation embedding w/ predicted answers	38.3
Explicit relation embedding w/ gold answers	39.5
IR w/ Previous gold answers + **Reading** w/ Append relation descriptions w/ gold answers	40.7

Table 2: Incorporating sequence information in the retrieval and the reading step slightly improves overall accuracy compared to answering questions in isolation.

score r to the start and end scores of the candidate answer (Equation 1) as

$$\begin{aligned} Start(i) &= \exp(\mathbf{t_i}^T W_{start}\mathbf{q} + r) \\ End(j) &= \exp(\mathbf{t_j}^T W_{end}\mathbf{q} + r). \end{aligned} \qquad (2)$$

The relation embedding module is trained jointly with the reader.

5 Baseline Results

We use QBLink to compare the baselines' question answering accuracy. Incorporating previous questions and answers slightly improves the accuracy (Table 2).

We set the maximum number of retrieved documents to ten, and each document is divided into paragraphs each of 400 tokens. At test time, we merge the top ten ranked such paragraphs and feed them to the reader. We use the reader network of Chen et al. (2017). We limit the number of relation description spans for each entity pair to five. We used an LSTM of one hidden layer and 128 hidden units for the paragraph, question, and relation description encoders. Each reader was trained for twenty epochs.

Table 2 summarizes the results of the baselines (Section 4). Question-answering accuracy is exact-match accuracy since we limit the answer spans to entity mentions whose boundaries are fixed for all models.

Question: This man attempted to impress Jodie Foster by shooting Ronald Reagan, but he failed to kill the President. At trial, he was found not guilty by reason of insanity.
Gold answer to previous question: Ronald Reagan
Predict without relation span: George H. W. Bush
Correct answer: John Hinckley Jr.
Relation span: He is best known for defending President Ronald Reagan during the assassination attempt by John Hinckley Jr.

Figure 2: Modeling the relation between *President Ronald Reagan* and *John Hinckley Jr.* expressed by relation span helps the reader select the correct answer entity.

Incorporating the previous answer in the retrieval and the reading components slightly improves the overall question answering accuracy (Table 2). The accuracy drops by more than 3% when using the entire text of previous questions in the retrieval phase. Modeling relations reduces the accuracy slightly compared to augmenting paragraphs with relation spans. One possible explanation is that our relation embedding model ends up being under-trained since we could not retrieve any relation-spans for many questions. Replacing Wikipedia with a larger corpus (e.g., ClueWeb) might help improve the training of the relation embedding model. Unsurprisingly, the gold answers to previous questions are more useful than the predicted answers, which highlights a need for models that take into account the uncertainty about previous answers when gold previous answers are not available. However, providing answers to previous questions is consistent for most Quiz Bowl tournament play.

Figure 2 gives an example of how explicit relation embedding helps reader get a correct prediction. Without the relation span, the model predicts George H. W. Bush (vice president at that time) as correct answer. Including the direct relation span between Reagan and John Hinckley Jr., the model gets the correct answer.

6 Related Work and Discussion

We adopt the open-domain question answering framework (Wang et al., 2018; Chen et al., 2017). Previous work considers improving that base framework itself (Clark and Gardner, 2018; Swayamdipta et al., 2018, inter alia). But retains the assumption of answering individual questions.

Aside from the open-domain setup, much of the recent work on question answering has focused on the sub-problem of reading-comprehension, where the gold answer to each question is assumed to exist in a given single paragraph for the model to read (Hermann et al., 2015; Rajpurkar et al., 2016; Seo et al., 2017). Another line of work on question answering is question answering over structured knowledge-bases (Berant et al., 2013; Berant and Liang, 2014; Yao and Van Durme, 2014; Gardner and Krishnamurthy, 2017). Although we focus on the more general open-domain setup, QBLink can be adapted to be usable in the reading-comprehension setup as well as the question answering over knowledge-bases setup.

Several question answering datasets have been proposed (Berant et al., 2013; Joshi et al., 2017; Trischler et al., 2017; Rajpurkar et al., 2018, inter alia). However, all of them were limited to answering individual questions.

Saha et al. (2018) study the problem of sequential question answering, and introduce a dataset for the task. However, we differ from them in two aspects: 1) They consider question-answering over structured knowledge-bases. 2) Their dataset construction was overly synthetic: templates were collected by human annotators given knowledge-base predicates. Further, sequences were constructed synthetically as well by grouping individual questions by predicate or subjects.

Both Iyyer et al. (2017) and Talmor and Berant (2018) answer complex questions by decomposing each into a sequence of simple questions. Iyyer et al. (2017) adopt a semantic parsing approach to answer questions over semi-structured tables. They construct a dataset of around 6,000 question sequences by asking humans to rewrite a set of 2,000 complex questions into simple sequences. Talmor and Berant (2018) consider the setup of open-domain question answering over unstructured text, but their dataset is constructed synthetically (with human paraphrasing) by combining simple questions with a few rules.

In parallel to our work, Choi et al. (2018) and Reddy et al. (2018) introduce sequential question answering datasets (QuAC and CoQA) that focus on the reading comprehension setup (i.e., a single text snippet is pre-specified for answering the given questions). QBLink is entirely naturally occurring (all questions and answers were authored independently from any knowledge sources) and is primarily designed to challenge human players.

The idea of our baseline to improving the reading step by incorporating additional relation description spans is similar as Weissenborn et al. (2017) and Mihaylov and Frank (2018), who integrate background commonsense knowledge into reading-comprehension systems. Both rely on structured knowledge bases to extract information about semantic relations that hold between entities. On the other hand, we extract text spans that mention each pair of entities and encoded them into vector representations of the relations between entities.

7 Conclusions and Future Work

We introduce QBLink, a dataset of 56,000 naturally occurring sequential question, answer pairs. The questions are designed primarily to challenge human players in Quiz Bowl tournaments. We use QBLink to evaluate baselines for sequential open-domain question answering. We show that incorporating sequential information helps slightly improve question answering accuracy.

In the future, we would like to invest in building better sequential question answering models that push the accuracy beyond the presented baselines. Specifically, we will look at how to better model the interaction between the reader and the relation embedding model and how to improve the relation embedding model itself by adopting ideas from the relation extraction literature (Miwa and Bansal, 2016; Peng et al., 2017; Ammar et al., 2017).

Acknowledgments

We thank the anonymous reviewers and members of the UMD CLIP lab for their comments and suggestions. Jordan Boyd-Graber is supported by NSF Grant IIS-1652666. Ahmed Elgohary is supported by an IBM PhD fellowship. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

References

Waleed Ammar, Matthew Peters, Chandra Bhagavatula, and Russell Power. 2017. The AI2 system at SemEval-2017 task 10 (ScienceIE): semi-supervised end-to-end entity and relation extraction. In *The International Workshop on Semantic Evaluation*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics*.

Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daumé III. 2012. Besting the quiz master: Crowdsourcing incremental classification games. In *Empirical Methods in Natural Language Processing*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. In *Empirical Methods in Natural Language Processing*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Association for Computational Linguistics*.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *ACM International Conference on Information and Knowledge Management*.

Matt Gardner and Jayant Krishnamurthy. 2017. Open-vocabulary semantic parsing with both distributional statistics and formal knowledge. In *AAAI Conference on Artificial Intelligence*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*.

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

Phu Mon Htut, Samuel R Bowman, and Kyunghyun Cho. 2018. Training a ranking function for open-domain question answering. In *North American Chapter of the Association for Computational Linguistics: Student Research Workshop*.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *North American Association for Computational Linguistics*.

Mohit Iyyer, Wen tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Association for Computational Linguistics*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Association for Computational Linguistics*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *International Conference on Learning Representations*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Association for Computational Linguistics*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Association for Computational Linguistics*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics* 5.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. CoQA: A conversational question answering challenge. In *Empirical Methods in Natural Language Processing*.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *AAAI Conference on Artificial Intelligence*.

Gerard Salton and Chris Buckley. 1987. Term weighting approaches in automatic text retrieval. Technical report, Cornell University.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Swabha Swayamdipta, Ankur P Parikh, and Tom Kwiatkowski. 2018. Multi-mention learning for reading comprehension with neural cascades. In *International Conference on Learning Representations*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *North American Association for Computational Linguistics*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *The 2nd Workshop on Representation Learning for NLP*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018. Evidence aggregation for answer re-ranking in open-domain question answering. In *International Conference on Learning Representations*.

Dirk Weissenborn, Tom Koisk, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural NLU systems. *arXiv preprint arXiv:1706.02596* .

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Association for Computational Linguistics*.

Improving the results of string kernels in sentiment analysis and Arabic dialect identification by adapting them to your test set

Radu Tudor Ionescu[1,2] and **Andrei M. Butnaru**[1]

[1]University of Bucharest,
Department of Computer Science
14 Academiei, Bucharest, Romania
`butnaruandreimadalin@gmail.com`

[2]Inception Institute
of Artificial Intelligence (IIAI)
Al Maryah Island, Abu Dhabi, UAE
`raducu.ionescu@gmail.com`

Abstract

Recently, string kernels have obtained state-of-the-art results in various text classification tasks such as Arabic dialect identification or native language identification. In this paper, we apply two simple yet effective transductive learning approaches to further improve the results of string kernels. The first approach is based on interpreting the pairwise string kernel similarities between samples in the training set and samples in the test set as features. Our second approach is a simple self-training method based on two learning iterations. In the first iteration, a classifier is trained on the training set and tested on the test set, as usual. In the second iteration, a number of test samples (to which the classifier associated higher confidence scores) are added to the training set for another round of training. However, the ground-truth labels of the added test samples are not necessary. Instead, we use the labels predicted by the classifier in the first training iteration. By adapting string kernels to the test set, we report significantly better accuracy rates in English polarity classification and Arabic dialect identification.

1 Introduction

In recent years, methods based on string kernels have demonstrated remarkable performance in various text classification tasks ranging from authorship identification (Popescu and Grozea, 2012) and sentiment analysis (Giménez-Pérez et al., 2017; Popescu et al., 2017) to native language identification (Popescu and Ionescu, 2013; Ionescu et al., 2014, 2016; Ionescu and Popescu, 2017), dialect identification (Ionescu and Popescu, 2016b; Ionescu and Butnaru, 2017; Butnaru and Ionescu, 2018) and automatic essay scoring (Cozma et al., 2018). As long as a labeled training set is available, string kernels can reach state-of-the-art results in various languages including English (Ionescu et al., 2014; Giménez-Pérez et al., 2017; Cozma et al., 2018), Arabic (Ionescu, 2015; Ionescu et al., 2016; Ionescu and Butnaru, 2017; Butnaru and Ionescu, 2018), Chinese (Popescu et al., 2017) and Norwegian (Ionescu et al., 2016). Different from all these recent approaches, we use unlabeled data from the test set to significantly increase the performance of string kernels. More precisely, we propose two transductive learning approaches combined into a unified framework. We show that the proposed framework improves the results of string kernels in two different tasks (cross-domain sentiment classification and Arabic dialect identification) and two different languages (English and Arabic). To the best of our knowledge, transductive learning frameworks based on string kernels have not been studied in previous works.

2 Transductive String Kernels

String kernels. Kernel functions (Shawe-Taylor and Cristianini, 2004) capture the intuitive notion of similarity between objects in a specific domain. For example, in text mining, string kernels can be used to measure the pairwise similarity between text samples, simply based on character n-grams. Various string kernel functions have been proposed to date (Lodhi et al., 2002; Shawe-Taylor and Cristianini, 2004; Ionescu et al., 2014). Perhaps one of the most recently introduced string kernels is the histogram intersection string kernel (Ionescu et al., 2014). For two strings over an alphabet Σ, $x, y \in \Sigma^*$, the intersection string kernel is formally defined as follows:

$$k^{\cap}(x,y) = \sum_{v \in \Sigma^p} \min\{\text{num}_v(x), \text{num}_v(y)\}, \quad (1)$$

where $\text{num}_v(x)$ is the number of occurrences of n-gram v as a substring in x, and p is the length of v. The spectrum string kernel or the presence bits string kernel can be defined in a similar fashion

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1084–1090
Brussels, Belgium, October 31 - November 4, 2018.

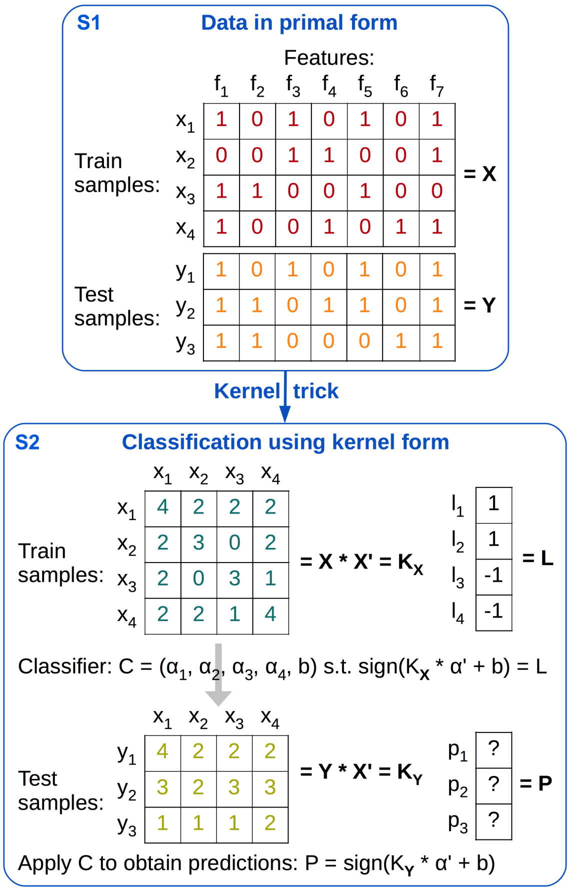

Figure 1: The standard kernel learning pipeline based on the linear kernel. Kernel normalization is not illustrated for simplicity. Best viewed in color.

(Ionescu et al., 2014). The standard kernel learning pipeline is presented in Figure 1. String kernels help to efficiently (Popescu et al., 2017) compute the dual representation directly, thus skipping the first step in the pipeline illustrated in Figure 1.

Transductive string kernels. We propose a simple and straightforward approach to produce a transductive similarity measure suitable for strings, as illustrated in Figure 2. We take the following steps to derive transductive string kernels. For a given kernel (similarity) function k, we first build the full kernel matrix K, by including the pairwise similarities of samples from both the train and the test sets (step $S1$ in Figure 2) . For a training set $X = \{x_1, x_2, ..., x_m\}$ of m samples and a test set $Y = \{y_1, y_2, ..., y_n\}$ of n samples, such that $X \cap Y = \emptyset$, each component in the full kernel matrix is defined as follows (step $S2$ in Figure 2):

$$K_{ij} = k(z_i, z_j), \tag{2}$$

where z_i and z_j are samples from the set $Z = X \cup Y = \{x_1, x_2, ..., x_m, y_1, y_2, ..., y_n\}$, for all $1 \le i, j \le m + n$. We then normalize the kernel matrix by dividing each component by the square

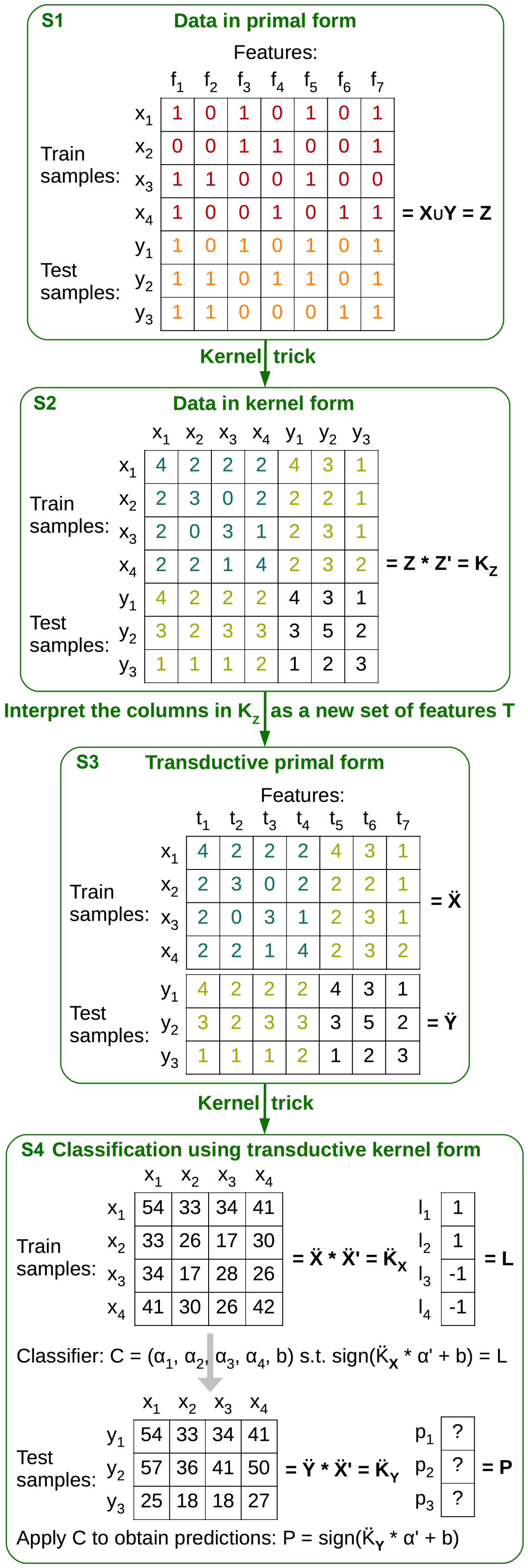

Figure 2: The transductive kernel learning pipeline based on the linear kernel. Kernel normalization and RBF kernel transformation are not illustrated for simplicity. Best viewed in color.

root of the product of the two corresponding diagonal components:

$$\hat{K}_{ij} = \frac{K_{ij}}{\sqrt{K_{ii} \cdot K_{jj}}}. \quad (3)$$

We transform the normalized kernel matrix into a radial basis function (RBF) kernel matrix as follows:

$$\tilde{K}_{ij} = exp\left(-\frac{1 - \hat{K}_{ij}}{2\sigma^2}\right). \quad (4)$$

As the kernel matrix is already normalized, we can choose $\sigma^2 = 0.5$ for simplicity. Therefore, Equation (4) becomes:

$$\tilde{K}_{ij} = exp\left(-1 + \hat{K}_{ij}\right). \quad (5)$$

Each row in the RBF kernel matrix $\tilde{K}$ is now interpreted as a feature vector, going from step $S2$ to step $S3$ in Figure 2. In other words, each sample z_i is represented by a feature vector that contains the similarity between the respective sample z_i and all the samples in Z (step $S3$ in Figure 2). Since Z includes the test samples as well, the feature vector is inherently adapted to the test set. Indeed, it is easy to see that the features will be different if we choose to apply the string kernel approach on a set of test samples Y', such that $Y' \neq Y$. It is important to note that through the features, the subsequent classifier will have some information about the test samples at training time. More specifically, the feature vector conveys information about how similar is every test sample to every training sample. We next consider the linear kernel, which is given by the scalar product between the new feature vectors. To obtain the final linear kernel matrix, we simply need to compute the product between the RBF kernel matrix and its transpose (step $S4$ in Figure 2):

$$\ddot{K} = \tilde{K} \cdot \tilde{K}'. \quad (6)$$

In this way, the samples from the test set, which are included in Z, are used to obtain new (transductive) string kernels that are adapted to the test set at hand.

Transductive kernel classifier. After obtaining the transductive string kernels, we use a simple transductive learning approach that falls in the category of self-training methods (McClosky et al., 2006; Chen et al., 2011). The transductive approach is divided into two learning iterations. In the first iteration, a kernel classifier is trained on the training data and applied on the test data, just as usual. Next, the test samples are sorted by the classifier's confidence score to maximize the probability of correctly predicted labels in the top of the sorted list. In the second iteration, a fixed number of samples (1000 in the experiments) from the top of the list are added to the training set for another round of training. Even though a small percent (less than 8% in all experiments) of the predicted labels corresponding to the newly included samples are wrong, the classifier has the chance to learn some useful patterns (from the correctly predicted labels) only visible in the test data. The transductive kernel classifier (TKC) is based on the intuition that the added test samples bring more useful information than noise, since the majority of added test samples have correct labels. Finally, we would like to stress out that *the ground-truth test labels are never used in our transductive algorithm.*

The proposed transductive learning approaches are used together in a unified framework. As any other transductive learning method, the main disadvantage of the proposed framework is that the *unlabeled* test samples from the target domain need to be used in the training stage. Nevertheless, we present empirical results indicating that our approach can obtain *significantly better* accuracy rates in cross-domain polarity classification and Arabic dialect identification compared to state-of-the-art methods based on string kernels (Giménez-Pérez et al., 2017; Ionescu and Butnaru, 2017). We also report better results than other domain adaptation methods (Pan et al., 2010; Bollegala et al., 2013; Franco-Salvador et al., 2015; Sun et al., 2016; Huang et al., 2017).

3 Polarity Classification

Data set. For the cross-domain polarity classification experiments, we use the second version of Multi-Domain Sentiment Dataset (Blitzer et al., 2007). The data set contains Amazon product reviews of four different domains: Books (B), DVDs (D), Electronics (E) and Kitchen appliances (K). Reviews contain star ratings (from 1 to 5) which are converted into binary labels as follows: reviews rated with more than 3 stars are labeled as positive, and those with less than 3 stars as negative. In each domain, there are 1000 positive and 1000 negative reviews.

Baselines. We compare our approach with several methods (Pan et al., 2010; Bollegala et al., 2013; Franco-Salvador et al., 2015; Sun et al., 2016; Giménez-Pérez et al., 2017; Huang et al.,

Method	DEK→B	BEK→D	BDK→E	BDE→K
SST	76.3	78.3	83.9	85.2
KE-Meta	77.9	80.4	78.9	82.5
$K_{0/1}$ (sota)	82.0	81.9	83.6	85.1
$K_{\cap}$ (sota)	80.7	80.7	83.0	85.2
$\ddot{K}_{0/1}$	82.9	83.2*	84.8*	86.0*
$\ddot{K}_{\cap}$	82.5	82.9*	84.5*	86.1*
$\ddot{K}_{0/1}$ + TKC	**84.1***	**84.0***	**85.4***	86.9*
$\ddot{K}_{\cap}$ + TKC	83.8*	83.5*	85.0*	**87.1***

Table 1: Multi-source cross-domain polarity classification accuracy rates (in %) of our transductive approaches versus a state-of-the-art (sota) baseline based on string kernels (Giménez-Pérez et al., 2017), as well as SST (Bollegala et al., 2013) and KE-Meta (Franco-Salvador et al., 2015). The best accuracy rates are highlighted in bold. The marker * indicates that the performance is significantly better than the best baseline string kernel according to a paired McNemar's test performed at a significance level of 0.01.

2017) in two cross-domain settings. Using string kernels, Giménez-Pérez et al. (2017) reported better performance than SST (Bollegala et al., 2013) and KE-Meta (Franco-Salvador et al., 2015) in the multi-source domain setting. In addition, we compare our approach with SFA (Pan et al., 2010), KMM (Huang et al., 2007), CORAL (Sun et al., 2016) and TR-TrAdaBoost (Huang et al., 2017) in the single-source setting.

Evaluation procedure and parameters. We follow the same evaluation methodology of Giménez-Pérez et al. (2017), to ensure a fair comparison. Furthermore, we use the same kernels, namely the presence bits string kernel ($K_{0/1}$) and the intersection string kernel ($K_{\cap}$), and the same range of character n-grams (5-8). To compute the string kernels, we used the open-source code provided by Ionescu and Popescu (2016a). For the transductive kernel classifier, we select $r = 1000$ unlabeled test samples to be included in the training set for the second round of training. We choose Kernel Ridge Regression (Shawe-Taylor and Cristianini, 2004) as classifier and set its regularization parameter to 10^{-5} in all our experiments. Although Giménez-Pérez et al. (2017) used a different classifier, namely Kernel Discriminant Analysis, we observed that Kernel Ridge Regression produces similar results ($\pm 0.1\%$) when we employ the same string kernels. As Giménez-Pérez et al. (2017), we evaluate our approach in two cross-domain settings. In the multi-source setting, we train the models on all domains, except the one used for testing. In the single-source setting, we train the models on one of the four domains and we independently test the models on the remaining three domains.

Results in multi-source setting. The results for the multi-source cross-domain polarity classification setting are presented in Table 1. Both the transductive presence bits string kernel ($\ddot{K}_{0/1}$) and the transductive intersection kernel ($\ddot{K}_{\cap}$) obtain better results than their original counterparts. Moreover, according to the McNemar's test (Dietterich, 1998), the results on the DVDs, the Electronics and the Kitchen target domains are significantly better than the best baseline string kernel, with a confidence level of 0.01. When we employ the transductive kernel classifier (TKC), we obtain even better results. On all domains, the accuracy rates yielded by the transductive classifier are more than 1.5% better than the best baseline. For example, on the Books domain the accuracy of the transductive classifier based on the presence bits kernel (84.1%) is 2.1% above the best baseline (82.0%) represented by the intersection string kernel. Remarkably, the improvements brought by our transductive string kernel approach are statistically significant in all domains.

Results in single-source setting. The results for the single-source cross-domain polarity classification setting are presented in Table 2. We considered all possible combinations of source and target domains in this experiment, and we improve the results in each and every case. Without exception, the accuracy rates reached by the transductive string kernels are significantly better than the best baseline string kernel (Giménez-Pérez et al., 2017), according to the McNemar's test performed at a confidence level of 0.01. The highest improvements (above 2.7%) are obtained when the source domain contains Books reviews and the target domain contains Kitchen reviews. As in the multi-source setting, we obtain much better results when the transductive classifier is employed for the learning task. In all cases, the accuracy rates of the transductive classifier are more than 2% better than the best baseline string kernel. Remarkably, in four cases (E→B, E→D, B→K and D→K) our improvements are greater than 4%. The improvements brought by our transductive classifier based on string kernels are statistically significant in each and every case. In comparison with SFA (Pan et al., 2010), we obtain better results in all but one case (K→D). With respect to KMM (Huang et al., 2007), we also obtain better results in all but one case (B→E). Remarkably, we surpass the

Method	D→B	E→B	K→B	B→D	E→D	K→D	B→E	D→E	K→E	B→K	D→K	E→K
SFA	79.8	78.3	75.2	81.4	77.2	**78.5**	73.5	76.7	85.1	79.1	80.8	86.8
KMM	78.6	-	-	-	-	72.2	**76.9**	-	-	-	-	83.6
CORAL	78.3	-	-	-	-	73.9	76.3	-	-	-	-	83.6
TR-TrAdaBoost	74.7	69.1	70.6	79.6	71.8	74.4	74.9	75.9	83.1	77.8	75.7	83.7
$K_{0/1}$ (sota)	82.0	72.4	72.7	81.4	74.9	73.6	71.3	74.4	83.9	74.6	75.4	84.9
$K_{\cap}$ (sota)	82.1	72.4	72.8	81.3	75.1	72.9	71.8	74.5	84.4	74.9	75.1	84.9
$\ddot{K}_{0/1}$	83.3*	74.5*	74.3*	83.0*	76.9*	74.9*	74.0*	76.0*	85.4*	77.6*	77.3*	86.0*
$\ddot{K}_{\cap}$	83.2*	74.2*	74.0*	82.8*	76.4*	75.1*	74.2*	75.9*	85.2*	77.6*	77.3*	85.9*
$\ddot{K}_{0/1}$ + TKC	**84.9***	**78.5***	**76.6***	84.0*	**79.6***	76.4*	76.6*	**77.1***	**86.4***	**79.6***	**80.9***	**87.0***
$\ddot{K}_{\cap}$ + TKC	84.5*	**78.5***	75.8*	**84.2***	79.1*	76.5*	76.7*	76.8*	**86.4***	79.4*	80.5*	**87.0***

Table 2: Single-source cross-domain polarity classification accuracy rates (in %) of our transductive approaches versus a state-of-the-art (sota) baseline based on string kernels (Giménez-Pérez et al., 2017), as well as SFA (Pan et al., 2010), KMM (Huang et al., 2007), CORAL (Sun et al., 2016) and TR-TrAdaBoost (Huang et al., 2017). The best accuracy rates are highlighted in bold. The marker * indicates that the performance is significantly better than the best baseline string kernel according to a paired McNemar's test performed at a significance level of 0.01.

Method	Development	Test
Meta-classifier	-	71.65
$K_{0/1}$+$K_{\cap}$+K_{LRD}+K_{ivec} (sota)	64.17	76.27
$\ddot{K}_{0/1}$+$\ddot{K}_{\cap}$+$\ddot{K}_{LRD}$+$\ddot{K}_{ivec}$	65.42*	77.08
$\ddot{K}_{0/1}$+$\ddot{K}_{\cap}$+$\ddot{K}_{LRD}$+$\ddot{K}_{ivec}$ + TKC	**66.73***	**78.35***

Table 3: Arabic dialect identification accuracy rates (in %) of our adapted string kernels versus the 2017 ADI Shared Task winner (sota) (Ionescu and Butnaru, 2017) and the first runner up (Malmasi and Zampieri, 2017). The best accuracy rates are highlighted in bold. The marker * indicates that the performance is significantly better than (Ionescu and Butnaru, 2017) according to a paired McNemar's test performed at a significance level of 0.01.

other state-of-the-art approaches (Sun et al., 2016; Huang et al., 2017) in all cases.

4 Arabic Dialect Identification

Data set. The Arabic Dialect Identification (ADI) data set (Ali et al., 2016) contains audio recordings and Automatic Speech Recognition (ASR) transcripts of Arabic speech collected from the Broadcast News domain. The classification task is to discriminate between Modern Standard Arabic and four Arabic dialects, namely Egyptian, Gulf, Levantine, and Maghrebi. The training set contains 14000 samples, the development set contains 1524 samples, and the test contains another 1492 samples. The data set was used in the ADI Shared Task of the 2017 VarDial Evaluation Campaign (Zampieri et al., 2017).

Baseline. We choose as baseline the approach of Ionescu and Butnaru (2017), which is based on string kernels and multiple kernel learning. The approach that we consider as baseline is the winner of the 2017 ADI Shared Task (Zampieri et al., 2017). In addition, we also compare with the second-best approach (Meta-classifier) (Malmasi and Zampieri, 2017).

Evaluation procedure and parameters. Ionescu and Butnaru (2017) combined four kernels into a sum, and used Kernel Ridge Regression for training. Three of the kernels are based on character n-grams extracted from ASR transcripts. These are the presence bits string kernel ($K_{0/1}$), the intersection string kernel ($K_{\cap}$), and a kernel based on Local Rank Distance (K_{LRD}) (Ionescu, 2013). The fourth kernel is an RBF kernel (K_{ivec}) based on the i-vectors provided with the ADI data set (Ali et al., 2016). In our experiments, we employ the exact same kernels as Ionescu and Butnaru (2017) to ensure an unbiased comparison with their approach. As in the polarity classification experiments, we select $r = 1000$ unlabeled test samples to be included in the training set for the second round of training the transductive classifier, and we use Kernel Ridge Regression with a regularization of 10^{-5} in all our ADI experiments.

Results. The results for the cross-domain Arabic dialect identification experiments on both the development and the test sets are presented in Table 3. The domain-adapted sum of kernels obtains improvements above 0.8% over the state-of-the-art sum of kernels (Ionescu and Butnaru, 2017). The improvement on the development set (from 64.17% to 65.42%) is statistically significant. Nevertheless, we obtain higher and significant improvements when we employ the transductive classifier. Our best accuracy is 66.73% (2.56% above the baseline) on the development set and 78.35% (2.08% above the baseline) on the test set. The results show that our domain adaptation framework based on string kernels attains the best performance on the ADI Shared Task data set, and the improvements over the state-of-the-art are statistically significant, according to the McNemar's test.

References

Ahmed Ali, Najim Dehak, Patrick Cardinal, Sameer Khurana, Sree Harsha Yella, James Glass, Peter Bell, and Steve Renals. 2016. Automatic dialect detection in arabic broadcast speech. In *Proceedings of INTERSPEECH*, pages 2934–2938.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL*, pages 187–205.

D. Bollegala, D. Weir, and J. Carroll. 2013. Cross-Domain Sentiment Classification Using a Sentiment Sensitive Thesaurus. *IEEE Transactions on Knowledge and Data Engineering*, 25(8):1719–1731.

Andrei M. Butnaru and Radu Tudor Ionescu. 2018. UnibucKernel Reloaded: First Place in Arabic Dialect Identification for the Second Year in a Row. In *Proceedings of VarDial Workshop of COLING*, pages 77–87.

Minmin Chen, Kilian Weinberger, and John Blitzer. 2011. Co-Training for Domain Adaptation. In *Proceedings of NIPS*, pages 2456–2464.

Mădălina Cozma, Andrei Butnaru, and Radu Tudor Ionescu. 2018. Automated essay scoring with string kernels and word embeddings. In *Proceedings of ACL*, pages 503–509.

Thomas G. Dietterich. 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1923.

Marc Franco-Salvador, Fermin L. Cruz, Jose A. Troyano, and Paolo Rosso. 2015. Cross-domain polarity classification using a knowledge-enhanced meta-classifier. *Knowledge-Based Systems*, 86:46–56.

Rosa M. Giménez-Pérez, Marc Franco-Salvador, and Paolo Rosso. 2017. Single and Cross-domain Polarity Classification using String Kernels. In *Proceedings of EACL*, pages 558–563.

Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. 2007. Correcting sample selection bias by unlabeled data. In *Proceedings of NIPS*, pages 601–608.

Xingchang Huang, Yanghui Rao, Haoran Xie, Tak-Lam Wong, and Fu Lee Wang. 2017. Cross-Domain Sentiment Classification via Topic-Related TrAdaBoost. In *Proceedings of AAAI*, pages 4939–4940.

Radu Tudor Ionescu. 2013. Local Rank Distance. In *Proceedings of SYNASC*, pages 221–228.

Radu Tudor Ionescu. 2015. A Fast Algorithm for Local Rank Distance: Application to Arabic Native Language Identification. In *Proceedings of ICONIP*, volume 9490, pages 390–400.

Radu Tudor Ionescu and Andrei Butnaru. 2017. Learning to Identify Arabic and German Dialects using Multiple Kernels. In *Proceedings of VarDial Workshop of EACL*, pages 200–209.

Radu Tudor Ionescu and Marius Popescu. 2016a. Native Language Identification with String Kernels. In *Knowledge Transfer between Computer Vision and Text Mining*, Advances in Computer Vision and Pattern Recognition, chapter 8, pages 193–227. Springer International Publishing.

Radu Tudor Ionescu and Marius Popescu. 2016b. UnibucKernel: An Approach for Arabic Dialect Identification based on Multiple String Kernels. In *Proceedings of VarDial Workshop of COLING*, pages 135–144.

Radu Tudor Ionescu and Marius Popescu. 2017. Can string kernels pass the test of time in native language identification? In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 224–234.

Radu Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? a language-independent approach to native language identification. In *Proceedings of EMNLP*, pages 1363–1373.

Radu Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2016. String kernels for native language identification: Insights from behind the curtains. *Computational Linguistics*, 42(3):491–525.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Shervin Malmasi and Marcos Zampieri. 2017. Arabic Dialect Identification Using iVectors and ASR Transcripts. In *Proceedings of the VarDial Workshop of EACL*, pages 178–183.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective Self-training for Parsing. In *Proceedings of NAACL*, pages 152–159.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain Sentiment Classification via Spectral Feature Alignment. In *Proceedings of WWW*, pages 751–760.

Marius Popescu and Cristian Grozea. 2012. Kernel methods and string kernels for authorship analysis. In *Proceedings of CLEF (Online Working Notes/Labs/Workshop)*.

Marius Popescu, Cristian Grozea, and Radu Tudor Ionescu. 2017. HASKER: An efficient algorithm for string kernels. Application to polarity classification in various languages. In *Proceedings of KES*, pages 1755–1763.

Marius Popescu and Radu Tudor Ionescu. 2013. The Story of the Characters, the DNA and the Native Language. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 270–278.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of Frustratingly Easy Domain Adaptation. In *Proceedings of AAAI*, pages 2058–2065.

Marcos Zampieri, Shervin Malmasi, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, Jörg Tiedemann, Yves Scherrer, and Noëmi Aepli. 2017. Findings of the VarDial Evaluation Campaign 2017. In *Proceedings of VarDial Workshop of EACL*, pages 1–15.

Parameterized Convolutional Neural Networks for Aspect Level Sentiment Classification

Binxuan Huang
School of Computer Science
Carnegie Mellon University
`binxuanh@cs.cmu.edu`

Kathleen M. Carley
School of Computer Science
Carnegie Mellon University
`kathleen.carley@cs.cmu.edu`

Abstract

We introduce a novel parameterized convolutional neural network for aspect level sentiment classification. Using parameterized filters and parameterized gates, we incorporate aspect information into convolutional neural networks (CNN). Experiments demonstrate that our parameterized filters and parameterized gates effectively capture the aspect-specific features, and our CNN-based models achieve excellent results on SemEval 2014 datasets.

1 Introduction

Continuous growing of user generated text in social media platforms such as Twitter drives sentiment classification increasingly popular. The goal of sentiment classification is to detect whether a piece of text expresses a positive, a negative, or a neutral sentiment (Rosenthal et al., 2017). The majority of the literature focuses on general sentiment analysis (document level), not involving a specific topic or entity. When there are multiple aspects about an entity in a sentence, it is hard to determine the sentiment as a whole.

Differing from general sentiment classification, aspect level sentiment classification identifies opinions from text about specific entities and their aspects (Pontiki et al., 2015). For example, given a sentence "great food but the service was dreadful", the sentiment polarity about aspect "food" is positive while the sentiment polarity about "service" is negative. If we ignore the aspect information, it is hard to determine the sentiment for a target aspect, which accounts for a large portion of sentiment classification error (Jiang et al., 2011).

Recently, machine learning based approaches are becoming popular for this task. Representative approaches in literature include Support Vector Machine (SVM) with manually created features (Jiang et al., 2011; Wagner et al., 2014) and neural network based models (Tang et al., 2016; Wang et al., 2016; Huang et al., 2018). Because of neural networks' capacity of learning representations from data without feature engineering, they are of growing interest for this natural language processing task. The mainstream neural methods are either based on long short-term memory (Hochreiter and Schmidhuber, 1997) or memory networks (Sukhbaatar et al., 2015). None of them are using convolutional neural networks (CNN), which are good at capturing local patterns.

In the present work, we propose two simple yet effective convolutional neural networks with aspect information incorporated. The overall architecture differs significantly from previous work. Specifically, we design two novel neural units that take target aspects into account. One is parameterized filter, the other is parameterized gate. These units both are generated from aspect-specific features and are further applied on the sentence. Our experiments show that our two model variants work surprisingly well on this type of task.

2 Related Work

Aspect level sentiment classification is a branch of sentiment classification (Pang et al., 2002; Wang and Manning, 2012). It aims at identifying the sentiment polarity of one aspect target in a context sentence.

One typical early work tries to identify the aspect level sentiment polarity based on predefined language rules (Nasukawa and Yi, 2003). Nasukawa and Yi first perform dependency parsing on sentences. Then rules are applied to determine the sentiment about aspects. Standard machine learning algorithms like SVM are also widely used on this task. Jiang et al. create several target-dependent features, then they feed these target-dependent features with content features into an

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1091–1096
Brussels, Belgium, October 31 - November 4, 2018.

SVM classifier.

In recent years, aspect level sentiment classification is dominated by neural network based approaches. The majority of published works rely on the architecture of long short-term memory (LSTM) (Tang et al., 2016). Wang et al. (2016) use an attention vector generated from aspect embedding to better capture the important parts in sentences. Tay et al. (2018) introduce a word-aspect fusion operation to learn associative relationships between aspects and sentences. Huang et al. (2018) use an attention-over-attention layer to further improve the performance.

Another type of neural architectures known as memory network (Sukhbaatar et al., 2015) has also been used in this task. Tang et al. (2016) takes an aspect term as a query sent to external memory. Their model consists of multiple computational layers. Each layer is an attention model. One recent work Dyadic MemNN (Tay et al., 2017) places associative layers on top of memory networks to improve the performance.

The overall architecture in this paper differs significantly with all these previous works. To the best of our knowledge, this paper is the first attempt using convolutional neural networks (Kim, 2014; Huang and Carley, 2017) for aspect level sentiment classification.

3 Parameterized Convolutional Neural Networks

In this section, we introduce our method for aspect level sentiment classification, which is based on convolutional neural networks. We first describe CNN for general sentiment classification, then we introduce our two model variants Parameterized Filters for Convolutional Neural Networks (PF-CNN) and Parameterized Gated Convolutional Neural Networks (PG-CNN).

3.1 Problem Definition

In aspect level sentiment classification, we are given a sentence $s = [w_1, w_2, ..., w_i, ..., w_n]$ and an aspect target $t = [w_i, w_{i+1}, ..., w_{i+m-1}]$. The goal is to classify whether the sentiment towards the aspect in the sentence is positive, negative, or neutral.

3.2 Convolutional Neural Networks

We first briefly describe convolutional neural networks (CNN) for general sentiment classification (Kim, 2014). Given a sentence $s = [w_1, w_2, ..., w_i, ..., w_n]$, let $v_i \in R^k$ be the word vector for word w_i. A sentence of length n can be represented as a matrix $\mathbf{s} = [v_1, v_2, ..., v_n] \in R^{n \times k}$. A convolution filter $w \in R^{h \times k}$ with width h is applied to the word matrix to get high-level representative features. Specifically, for a word window $v_{i:i+h-1} \in R^{h \times k}$, a feature c_i is generated by

$$c_i = f(w \odot v_{i:i+h-1} + b) \tag{1}$$

where $\odot$ represents element-wise product, $b \in R$ is a bias term and f is a non-linear function. Sliding the filter window from the beginning of the word matrix till the end, we get a feature map $\mathbf{c} \in R^{n-h+1}$.

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}] \tag{2}$$

After that, a pooling operation is applied over the feature map to get one single general sentiment feature θ_g in each map. We use max pooling in the CNN for sentences.

$$\theta_g = pooling(\mathbf{c}) \tag{3}$$

We denote this process as $\theta_g = CNN_g(\mathbf{s}; w, b)$. Using d such convolutional filters, we can get a general sentiment feature vector $\Theta_g \in R^d$ without information from aspect terms.

3.3 Parameterized Filters

Standard convolutional neural networks do not consider information from aspect terms. Herein, our first model variant overcomes this issue by parameterizing filters using aspect terms. We call it Parameterized Filters for Convolutional Neural Networks (PF-CNN). The overall architecture is shown in the left of Figure 1.

Formally, given the aspect term with length m, $t = [w_i, w_{i+1}, ..., w_{i+m-1}]$ and the corresponding embedding matrix $\mathbf{t} = [v_i, v_{i+1}, ..., v_{i+m-1}] \in R^{m \times k}$, we first use another CNN_t to extract one single feature θ_t from $\mathbf{t}$.

$$\theta_t = CNN_t(\mathbf{t}; w_t, b_t) \tag{4}$$

where $w_t \in R^{h_t \times k}$, b_t are the convolutional filter, bias term for CNN_t. h_t is the width of filters applied on aspect targets. With $h_s \times k$ such filters and bias terms, we can get a feature matrix $\Theta_t \in R^{h_s \times k}$, where h_s is the filter width applied on sentences. We use average pooling in the CNN_t for aspects.

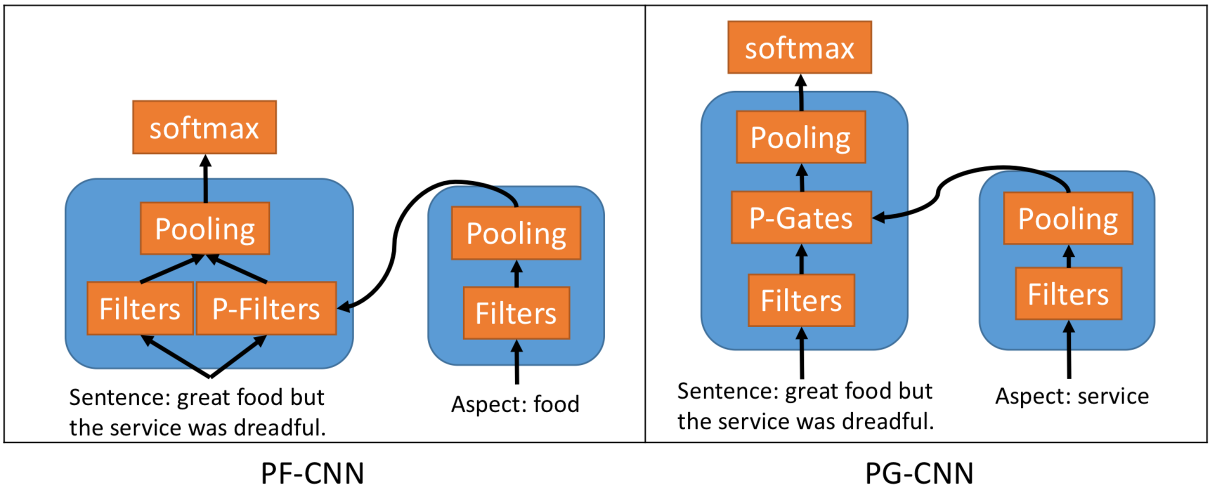

Figure 1: The overall architectures of PF-CNN and PG-CNN.

In the next step, Θ_t is further used as a convolutional filter applied on the sentence.

$$\theta_s = CNN_s(\mathbf{s}; \boldsymbol{\Theta}_t, b_s) \tag{5}$$

Using such d parameterized filters, we get the aspect-specific features $\Theta_s \in R^d$ with target term information. We further concatenate the targeted feature vector with general sentiment features as the final classification features $\Theta = [\Theta_g, \Theta_s]$.

3.4 Parameterized Gates

The second model variant we designed is called Parameterized Gated Convolutional Neural Networks (PG-CNN). The overall architecture is shown in the right of Figure 1.

Similar with PF-CNN, PG-CNN also utilizes a convolutional neural network to extract feature Θ_t from aspect terms, which instead is used as a gate (Dauphin et al., 2017) in the CNN applied on the sentence. The key step of PG-CNN is described in equation (6).

$$c_i = (w \odot v_{i:i+h-1} + b) \cdot \sigma(\Theta_t \odot v_{i:i+h-1} + b) \tag{6}$$

Instead of using a non-linear function f in equation (1), we use a gate $\sigma(\Theta_t \odot v_{i:i+h-1} + b)$ to control how much information passing to the next layer, where $\sigma(\cdot)$ is sigmoid function. For each general filter applied on the sentence, one parameterized gate is generated from the aspect.

After that, we generate the final classification feature Θ in the same way as standard CNN.

3.5 Final Classification

We feed the final classification feature into a linear layer to project Θ into the space of targeted classes:

$$x = W_l \cdot \Theta + b_l \tag{7}$$

where W_l and b_l are the weight matrix and bias. Following the linear layer, we use a softmax layer to compute the probability of class c.

$$P(y = c|x) = \frac{exp(x_c)}{\sum_{i \in C} exp(x_i)} \tag{8}$$

3.6 Model training

We train our model to minimize the cross-entropy loss function with L_2 regularization:

$$loss = -\sum_{(s,t)} \sum_{c \in C} I(y = c) log P(y = c|s, a) + \lambda ||p||^2$$

where $I(\cdot)$ is the indicator function and p is the set of all parameters in the convolutional layers and linear layer.

4 Experiments

4.1 Experiments Setting

Dataset

We adopt one widely used dataset from SemEval 2014 Task 4 (Pontiki et al., 2014). It contains two domain-specific datasets for laptops and restaurants. Each data point is a pair of a sentence and an aspect term. Experienced annotators tagged each pair with sentiment polarity. Following recent work (Tay et al., 2018), we take 500 training instances as development set[1]. Unfortunately, many works have not mentioned the usage of development set (Wang et al., 2016; Ma et al., 2017).

Hyperparameters and Training

We use rectifier as non-linear function f in the CNN_g, CNN_t and sigmoid in the CNN_s, filter window sizes of $1, 2, 3, 4$ with 100 feature maps

---

[1] The splits can be found at https://github.com/vanzytay/ABSA_DevSplits.

Dataset	Positive	Neutral	Negative
Laptop-Train	767	373	673
Laptop-Dev	220	87	193
Laptop-Test	341	169	128
Restaurant-Train	1886	531	685
Restaurant-Dev	278	102	120
Restaurant-Test	728	196	196

Table 1: Statistics of the datasets.

	Laptops		Restaurants	
Model	3-way	Binary	3-way	Binary
TD-LSTM	62.38	79.31	69.73	84.41
AT-LSTM	65.83	78.25	74.37	84.74
ATAE-LSTM	60.34	74.20	70.71	84.52
AF-LSTM	68.81	83.58	75.44	87.78
CNN	68.65	85.50	77.95	89.50
PF-CNN	**70.06**	**86.35**	**79.20**	90.15
PG-CNN	69.12	86.14	78.93	**90.58**

Table 2: Comparisons results with baselines. We use accuracy to measure the performance. Performances of baselines are cited from (Tay et al., 2018).

each, l_2 regularization term of 0.001 and mini-batch size of 25. Parameterized filters and gates have the same size and number as normal filters. They are generated uniformly by CNN with window sizes of $1, 2, 3, 4$, eg. among 100 parameterized filters with size 3, 25 of them are generated by aspect CNN with filter size 1, 2, 3, 4 respectively. The word embeddings are initialized with 300-dimensional Glove vectors (Pennington et al., 2014) and are fixed during training. For the out of vocabulary words we initialize them randomly from uniform distribution $U(-0.01, 0.01)$. We apply dropout on the final classification features of PG-CNN. The dropout rate is chosen as 0.3.

Training is done through mini-batch stochastic gradient descent with Adam update rule (Kingma and Ba, 2015). The initial learning rate is 0.001. If the training loss does not drop after every three epochs, we decrease the learning rate by half. We adopt early stopping based on the validation loss on development sets.

4.2 Results

We use accuracy metric to measure the performance. To show the effectiveness of our model, we compare it with several baseline methods. We list them as follows:

TD-LSTM uses two LSTM networks to model the preceding and following contexts surrounding the aspect term. The last hidden states of these two LSTM networks are concatenated for predicting the sentiment polarity (Tang et al., 2016).

AT-LSTM combines the sentence hidden states from a LSTM with the aspect term embedding to generate the attention vector. The final sentence representation is the weighted sum of the hidden states (Wang et al., 2016).

ATAE-LSTM further extends AT-LSTM by appending the aspect embedding into each word vector (Wang et al., 2016).

AF-LSTM introduces a word-aspect fusion attention to learn associative relationships between aspect and context words (Tay et al., 2018).

CNN uses the architecture proposed in (Kim, 2014) without explicitly considering aspect. We use filter window sizes of 1,2,3,4 with 100 maps each. Dropout rate is chosen as 0.5. Early stopping based on validation accuracy is applied.

Our two models achieve the best performance when compared to these baselines as shown in Table 2, which shows that our proposed neural units effectively captures the aspect-specific features. Compared to one recently proposed model AF-LSTM, our method achieve 2%-5% improvements. Surprisingly, a vanilla CNN works quite well on this problem. It even beats these well-designed LSTM models, which further proves that using CNN-based methods is a direction worth exploring.

4.3 Case Study & Discussion

Compared to a vanilla CNN, our two model variants could successfully distinguish the describing words for corresponding aspect targets. In the sentence “the appetizers are ok, but the service is slow”, a vanilla CNN outputs the same negative sentiment label for both aspect terms “appetizers” and “service”, while PF-CNN and PG-CNN successfully recognize that “slow” is only used for describing “service” and output neutral and negative labels for aspects “appetizers” and “service” respectively. In another example “the staff members are extremely friendly and even replaced my drink once when i dropped it outside”, our models also find out that positive and neutral sentiment for “staff” and “drink” respectively.

One thing we notice in our experiment is that a vanilla CNN ignoring aspects has comparable performance with some well-designed LSTM models in this aspect-level sentiment classification task.

For a sentence containing multiple aspects, we assume the majority of the aspect-level sentiment label is the sentence-level sentiment label. Using this labeling scheme, in the restaurant data, 1034 out of 1117 test points have the same sentence-level and aspect-level labels. Thus, a sentence-level classifier with accuracy 75% also classifies 70% aspect-labels correctly. A similar observation was made for the laptop dataset as well. Probably this is the reason why a vanilla CNN has comparable performance on these two datasets. For future research, a more balanced dataset would be helpful to overcome this issue.

5 Conclusion

We propose a novel method for aspect level sentiment classification. We introduce two novel neural units called parameterized filter and parameterized gate to incorporate aspect information into the convolutional neural network architecture. Comparisons with baseline methods show our model effectively learns the aspect-specific sentiment expressions. Our experiments demonstrate a significant improvement compared to multiple strong neural baselines.

To the best of our knowledge, our model is the first attempt using convolutional neural networks solving this problem. We hope this work could inspire future research exploring in this direction. It is also interesting to see whether such parameterized CNN architecture could benefit other natural language processing tasks involving text pairs like question answering task.

Acknowledgments

We would like to thank the reviewers for their helpful comments that greatly improved the article. We would also like to thank Sumeet Kumar for his valuable suggestions.

References

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th international conference on machine learning*.

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

Binxuan Huang and Kathleen M Carley. 2017. On predicting geolocation of tweets using convolutional neural networks. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 281–291. Springer.

Binxuan Huang, Yanglan Ou, and Kathleen M. Carley. 2018. Aspect level sentiment classification with attention-over-attention neural networks. In *Social, Cultural, and Behavioral Modeling*, pages 197–206, Cham. Springer International Publishing.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4068–4074.

Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77. ACM.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and

Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. In *AAAI*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. Dyadic memory networks for aspect-based sentiment analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 107–116. ACM.

Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. Dcu: Aspect-based polarity classification for semeval task 4.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*, pages 606–615.

Improving Multi-label Emotion Classification via Sentiment Classification with Dual Attention Transfer Network

Jianfei Yu♣, Luís Marujo♡, Jing Jiang♣, Pradeep Karuturi♡, William Brendel♡
♣ School of Information Systems, Singapore Management University, Singapore
♡ Snap Inc. Research, Venice, California, USA
♣ jfyu.2014@phdis.smu.edu.sg, jingjiang@smu.edu.sg
♡ {luis.marujo, pradeep.karuturi, william.brendel}@snap.com

Abstract

In this paper, we target at improving the performance of multi-label emotion classification with the help of sentiment classification. Specifically, we propose a new transfer learning architecture to divide the sentence representation into two different feature spaces, which are expected to respectively capture the general sentiment words and the other important emotion-specific words via a dual attention mechanism. Extensive experimental results demonstrate that our transfer learning approach can outperform several strong baselines and achieve the state-of-the-art performance on two benchmark datasets.

1 Introduction

In recent years, the number of user-generated comments on social media platforms has grown exponentially. In particular, social platforms such as Twitter allow users to easily share their personal opinions, attitudes and emotions about any topic through short posts. Understanding people's emotions expressed in these short posts can facilitate many important downstream applications such as emotional chatbots (Zhou et al., 2018b), personalized recommendations, stock market prediction, policy studies, etc. Therefore, it is crucial to develop effective emotion detection models to automatically identify emotions from these online posts.

ID	**Tweet**	**Emotion**
T1	AI revolution, soon is possible #fearless #good #goodness	joy, optimism
T2	Shitty is the worst feeling ever #depressed #anxiety	fear, sadness
T3	I am back lol. #revenge	joy, anger

Table 1: Example Tweets from SemEval-18 Task 1.

In the literature, emotion detection is typically modeled as a supervised multi-label classification problem, because each sentence may contain one or more emotions from a standard emotion set containing *anger*, *anticipation*, *disgust*, *fear*, *joy*, *love*, *optimism*, *pessimism*, *sadness*, *surprise* and *trust*. Table 1 shows three example sentences along with their emotion labels. Traditional approaches to emotion detection include lexicon-based methods (Wang and Pal, 2015), graphical model-based methods (Li et al., 2015b) and linear classifier-based methods (Quan et al., 2015; Li et al., 2015a). Given the recent success of deep learning models, various neural network models and advanced attention mechanisms have been proposed for this task and have achieved highly competitive results on several benchmark datasets (Wang et al., 2016; Abdul-Mageed and Ungar, 2017; Felbo et al., 2017; Baziotis et al., 2018; He and Xia, 2018; Kim et al., 2018).

However, these deep models must overcome a heavy reliance on large amounts of annotated data in order to learn a robust feature representation for multi-label emotion classification. In reality, large-scale datasets are usually not readily available and costly to obtain, partly due to the ambiguity of many informal expressions in user-generated comments. Conversely, it is easier to find datasets (especially in English) associated with another closely related task: sentiment classification, which aims to classify the sentiment polarity of a given piece of text (i.e., positive, negative and neutral). We expect that these resources may allow us to improve sentiment-sensitive representations and thus more accurately identify emotions in social media posts. To achieve these goals, we propose an effective transfer learning (TL) approach in this paper.

Most existing TL methods either 1) assume that both the source and the target tasks share the same sentence representation (Mou et al., 2016) or 2) divide the representation of each sentence into a

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1097–1102
Brussels, Belgium, October 31 - November 4, 2018.

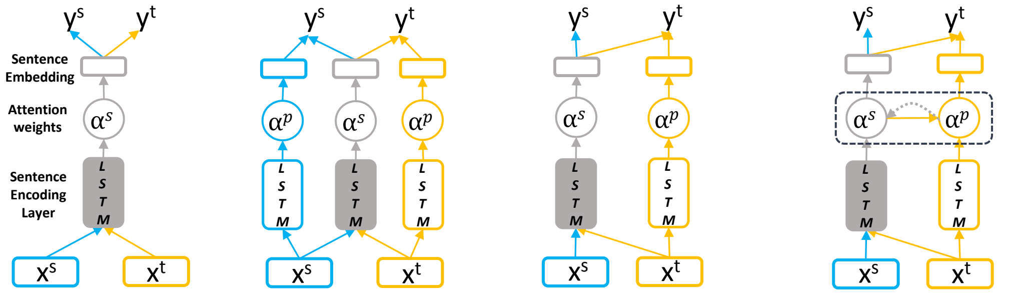

a. Fully-Shared (FS) b. Private-Shared-Private (PSP) c. Shared-Private (SP) d. Dual Attention Transfer Network

Figure 1: Overview of Different Transfer Learning Models.

shared feature space and two task-specific feature spaces (Liu et al., 2017; Yu et al., 2018), as demonstrated by Fig 1.a and Fig 1.b. However, when applying these TL approaches to our scenario, the former approach may lead the learnt sentence representation to pay more attention to general sentiment words such as *good* but less attention to the other sentiment-ambiguous words like *shock* that are also integral to emotion classification. The latter approach can capture both the sentiment and the emotion-specific words. However, some sentiment words only occur in the source sentiment classification task. These words tend to receive more attention in the source-specific feature space but less attention in the shared feature space, so they will be ignored in our emotion classification task. Intuitively, any sentiment word also indicates emotion and should not be ignored by our emotion classification task.

Therefore, we propose a shared-private (SP) model as shown in Fig 1.c, where we employ a shared LSTM layer to extract shared sentiment features for both sentiment and emotion classification tasks, and a target-specific LSTM layer to extract specific emotion features that are only sensitive to our emotion classification task. However, as pointed out by Liu et al. (2017) and Yu et al. (2018), it is not guaranteed that such a simple model can well differentiate the two feature spaces to extract shared and target-specific features as we expect. Take the sentence **T1** in Table 1 as an example. Both the shared and task-specific layers could assign higher attention weights to *good* and *goodness* due to their high frequencies in the training data but lower attention weights to *fearless* due to its rare occurrences. In this case, this SP model can only predict the *joy* emotion but ignores the *optimism* emotion. Hence, to enforce the orthogonality of the two feature spaces, we further introduce a dual attention mechanism, which feeds the attention weights in one feature space as extra inputs to compute those in the other feature space, and explicitly minimizes the similarity between the two sets of attention weights. Experimental results show that our dual attention transfer architecture can bring consistent performance gains in comparison with several existing transfer learning approaches, achieving the state-of-the-art performance on two benchmark datasets.

2 Methodology

2.1 Base Model for Emotion Classification

Given an input sentence, the goal of emotion analysis is to identify one or multiple emotions contained in it. Formally, let $\mathbf{x} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$ be the input sentence with n words, where $\mathbf{w}_j$ is a d-dimensional word vector for word w_j in the vocabulary $\mathcal{V}$, and is retrieved from a lookup table $\mathbf{E} \in \mathbb{R}^{d \times |\mathcal{V}|}$. Moreover, let $\mathcal{E}$ be a set of pre-defined emotion labels. Accordingly, for each $\mathbf{x}$, our task is to predict whether it contains one or more emotions in $\mathcal{E}$. We denote the output as $\mathbf{e} \in \{0,1\}^K$ where $\mathbf{e}_k \in \{0,1\}$ denotes whether or not $\mathbf{x}$ contains the k-th emotion. We further assume that we have a set of labeled sentences, denoted by $D^{\mathrm{e}} = \{\mathbf{x}^{(i)}, \mathbf{e}^{(i)}\}_{i=1}^{N}$.

Sentence Representation: We use the standard bi-directional Long Short Term Memory (Bi-LSTM) network to sequentially process each word in the input:

$$\overrightarrow{\mathbf{h}_j} = \mathrm{LSTM}(\overrightarrow{\mathbf{h}_{j-1}}, \mathbf{x}_j, \Theta_f),$$
$$\overleftarrow{\mathbf{h}_j} = \mathrm{LSTM}(\overleftarrow{\mathbf{h}_{j+1}}, \mathbf{x}_j, \Theta_b),$$

where Θ_f and Θ_b denotes all the parameters in the forward and backward LSTM. Then, for each word $\mathbf{x}_j$, its hidden state $\mathbf{h}_j \in \mathbb{R}^d$ is generated by concatenating $\overrightarrow{\mathbf{h}_j}$ and $\overleftarrow{\mathbf{h}_j}$ as $\mathbf{h}_j = [\overrightarrow{\mathbf{h}_j}; \overleftarrow{\mathbf{h}_j}]$.

For emotion classification, since emotion words are relatively more important for final predictions, we adopt the widely used attention mechanism (Bahdanau et al., 2014) to select the key words for sentence representation. Specifically, we first take the final hidden state $\mathbf{h}_n$ as a sentence summary vector $\mathbf{z}$, and then obtain the attention weight α_i for each hidden state $\mathbf{h}_j$ as follows:

$$u_j = \mathbf{v}^\top \tanh(\mathbf{W}_h \mathbf{h}_j + \mathbf{W}_z \mathbf{z}), \tag{1}$$

$$\alpha_j = \frac{\exp(u_j)}{\sum_{l=1}^{n} \exp(u_l)}, \tag{2}$$

where $\mathbf{W}_h, \mathbf{W}_z \in \mathbb{R}^{a \times d}$ and $\mathbf{v} \in \mathbb{R}^{a}$ are learnable parameters. The final sentence representation $\mathbf{H}$ is computed as:

$$\mathbf{H} = \sum_{j=1}^{n} \alpha_j \mathbf{h}_j.$$

Output Layer: We first apply a Multilayer Perceptron (MLP) with one hidden layer on top of $\mathbf{H}$, followed by normalizing it to obtain the probability distribution over all of the emotion labels:

$$p(\mathbf{e}^{(i)} \mid \mathbf{H}) = \mathbf{o}^{(i)} = \mathrm{softmax}(\mathrm{MLP}(\mathbf{H})).$$

Then, we propose to minimize the KL divergence between our predicted probability distribution and the normalized ground truth distribution as our objective function:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbf{e}_k^{(i)} \big(\log(\mathbf{e}_k^{(i)}) - \log(\mathbf{o}_k^{(i)}) \big).$$

During the test stage, we will select a threshold γ on the development set so that the emotion with scores higher than γ will be predicted as 1.

2.2 Transfer Learning Architecture

Due to the limited number of annotated data for multi-label emotion classification, here we resort to sentiment classification to consider a transfer learning scenario. Let $D^s = \{\mathbf{x}^{(m)}, y^{(m)}\}_{m=1}^{M}$ be another set of labeled sentences for sentiment classification, where $y^{(m)}$ is the ground-truth label indicating whether the m-th sentence is *positive*, *negative* or *neutral*.

2.2.1 Shared-Private (SP) Model

Intuitively, sentiment classification is a coarse-grained emotion analysis task, and can be fully leveraged to learn a more robust sentiment-sensitive representation. Therefore, we first use a shared attention-based Bi-LSTM layer to transform the input sentences in both tasks into a shared hidden representation $\mathbf{H_c}$, and also employ another task-specific Bi-LSTM layer to get the target-specific hidden representation $\mathbf{H_t}$. Next, we employ the following operations to map the hidden representations to the sentiment label y and the emotion label $\mathbf{e}$:

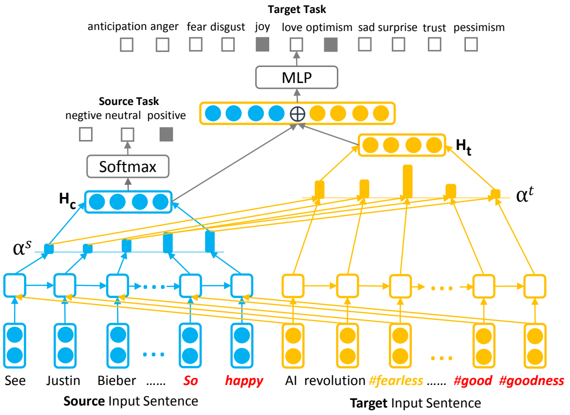

Figure 2: Dual Attention Transfer Network.

$$p(y^{(m)} | \mathbf{H_c}) = \mathrm{softmax}\big(\mathbf{W}^s \mathbf{H_c} + \mathbf{b}^s\big),$$
$$p(\mathbf{e}^{(i)} | \mathbf{H_c}, \mathbf{H_t}) = \mathrm{softmax}\big(\mathrm{MLP}([\mathbf{H_c}; \mathbf{H_t}])\big),$$

where $\mathbf{W}^s \in \mathbb{R}^{d \times 3}$ and $\mathbf{b}^s \in \mathbb{R}^3$ are the parameters for the source sentiment classification task.

2.2.2 Proposed Dual Attention Transfer Network (DATN)

As we introduced before, the shared and target-specific feature spaces in the above SP model are expected to respectively capture the general sentiment words and the task-specific emotion words. However, without any constraints, the two feature spaces may both tend to pay more attention to frequently occurring and important sentiment words like *great* and *happy*, but less to those rarely occurring but crucial emotion words like *anxiety* and *panic*. Therefore, to encourage the two feature spaces to focus on sentiment words and emotion-specific words respectively, we propose using the attention weights computed from the shared layer as extra inputs to compute the attention weights of the target-specific layer. Specifically, as shown in Fig. 2, we first use Eq.1 and Eq.2 to compute the attention weights α^s in the shared layer, and then use the following equation to obtain the attention weights α^t in the target specific layer:

$$u_j^t = \mathbf{v}^{t\top} \tanh(\mathbf{W}_h^t \mathbf{h}_j^t + w_\alpha \alpha_j^s + \mathbf{W}_z^t \mathbf{z}^t),$$
$$\alpha_j^t = \frac{\exp(u_j^t)}{\sum_{l=1}^n \exp(u_l^t)}.$$

In addition, we introduce another similarity loss to explicitly enforce the difference between the two attention weights and minimize the cosine similarity between α^s and α^t.

Finally, our combined objective function is defined as follows:

$$\mathcal{J} = -\frac{1}{M}\sum_{m=1}^{M} \log p(y_m|\mathbf{H_c}) + \mathcal{L}$$
$$+ \lambda \sum_{i=1}^{N} \text{cos\_sim}(\alpha_i^s, \alpha_i^t),$$

where λ is a hyperparameter used to control the effect of the similarity loss.

2.2.3 Model Details

During the training stage, we adopted the widely used alternating optimization strategy, which iteratively samples one mini-batch from D^{s} for only updating the parameters in the left part of our model, followed by sampling another mini-batch from D^{e} for updating all the parameters in our model. It is also worth noting that in Fig. 2, we first obtain the shared attention weights α^s and feed it as extra inputs to compute α^t. In fact, to differentiate the attention weights in the two feature spaces, we can also first compute α^t, followed by computing α^s based on α^t. We refer to these two variants of our model as DATN-1 and DATN-2 respectively.

3 Experiments

3.1 Experiment Settings

Datasets: We conduct experiments on both English and Chinese languages.

For **English**, we employ a widely used Twitter dataset from SemEval 2016 Task 4A (Nakov et al., 2016) as our source sentiment classification task. For our target emotion classification task, we use the Twitter dataset recently released by SemEval 2018 Task 1C (Mohammad et al., 2018), which contains 11 emotions as shown in the top of Fig. 2. To tokenize the tweets in our dataset, we follow (Owoputi et al., 2013) by adopting most of their preprocessing rules except that we split the hashtag into '#' and its subsequent word.

	Dataset	Train	Dev	Test	Words
E1	SemEval-18	6,838	886	3,259	32,557
S1	SemEval-16	28,631	-	-	40439
E2	Ren-CECps-1	13,841	1,972	3,602	40,099
S2	Ren-CECps-2	15,199	-	-	

Table 2: The number of sentences in each dataset.

For **Chinese**, we use a well known Chinese blog dataset Ren-CECps from (Quan and Ren, 2010), which contains 1487 documents with each sentence labeled by a sentiment label and 8 emotion labels: *anger*, *expectation*, *anxiety*, *joy*, *love*, *hate*, *sorrow* and *surprise*. Given the difficulty of finding a large-scale sentiment classification dataset specific to Chinese blogs, we simply divided the original dataset to form our source and target tasks[1]. The basic statistics of our two datasets are summarized in Table 2.

Parameter Settings: The word embedding size d is set to be 300 for E1 and 200 for E2, and the lookup table $\mathbf{E}$ is initialized by pre-trained word embeddings based on Glove[2]. The hidden dimension and the number of LSTM layers in both datasets are set to be 200 and 1. During training, Adam (Kingma and Ba, 2014) is used to schedule the learning rate, where the initial learning rate is set to be 0.001. Also, the dropout rate is set to 0.5. After tuning, λ is set as 0.05 for both datasets, and γ is set as 0.12 for E1 and 0.2 for E2. All the models are implemented with Tensorflow.

Evaluation Metrics: We take the official code from SemEval-18 Task 1C and use accuracy and Macro F1 score as main metrics. For E2, we follow (Zhou et al., 2018a) to use average precision (AP) and one error (OE) as secondary metrics.

3.2 Results

To better evaluate our proposed methods, we employed the following systems for comparison: 1) *Base*, training our base model in Section 2.1 only on D^{e}; 2) *FT* (Fine-Tuning), using D^{s} to pre-train the whole model, followed by using D^{e} to Fine Tune the model parameters; 3) *FS*, the Fully-Shared framework by (Mou et al., 2016) as shown in Fig 1.a; 4) *PSP* and *APSP*, the Private-Shared-Private framework and its extension with Adver-

---

[1] The first 560/80/160 documents are used as train/dev/test set for emotion classification, and the remaining 687 documents are used for sentiment classification.

[2] https://nlp.stanford.edu/projects/glove/.

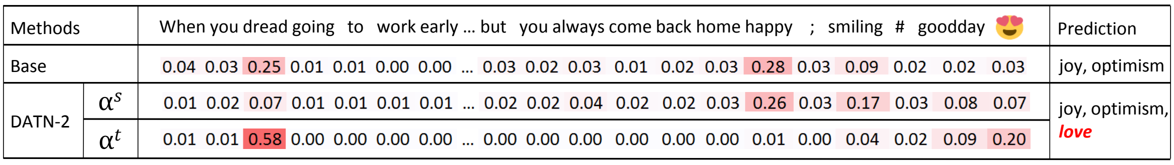

Methods		When you dread going to work early ... but you always come back home happy ; smiling # goodday 😍	Prediction
Base		0.04 0.03 0.25 0.01 0.01 0.00 0.00 ... 0.03 0.02 0.03 0.01 0.02 0.03 0.28 0.03 0.09 0.02 0.02 0.03	joy, optimism
DATN-2	α^s	0.01 0.02 0.07 0.01 0.01 0.01 0.01 ... 0.02 0.02 0.04 0.02 0.02 0.03 0.26 0.03 0.17 0.03 0.08 0.07	joy, optimism, ***love***
	α^t	0.01 0.01 0.58 0.00 0.00 0.00 0.00 ... 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.00 0.04 0.02 0.09 0.20	

Figure 3: Comparison of attention weights between *Base* and our *DATN-2* model on a test sentence from SemEval-18. Note that the ground truth emotion labels for this example are *joy*, *optimism* and *love*.

Methods	S1 → E1		S2 → E2			
	ACC↑	**F1↑**	**ACC↑**	**F1↑**	**AP↑**	**OE↓**
Base	0.569	0.521	0.368	0.399	0.648	0.531
FT	0.575	0.519	0.372	0.398	0.655	0.519
FS	0.577	0.526	0.386	0.403	0.662	0.507
PSP	0.579	0.531	0.384	0.405	0.658	0.517
APSP	0.580	0.540	0.389	0.399	0.670	0.499
SP	0.577	0.532	0.389	0.410	0.667	0.507
DATN-1	0.582	0.543	0.393	0.410	0.670	0.501
DATN-2	**0.583**	**0.544**	**0.400**	**0.420**	**0.674**	**0.498**
Rank 2	0.582	0.534	-	0.392	0.641	0.523
Rank 1	0.595	0.542	-	0.416	0.680	0.455
DATN-2*	***0.597***	***0.551***	-	-	-	-
Base†	-	-	0.445	0.426	0.725	0.425
DATN-2†	-	-	***0.457***	***0.444***	***0.732***	***0.415***

Table 3: The results of different transfer learning methods by averaging ten runs (top) and the comparison between our best model and the state-of-the-art systems (bottom). DATN-2* indicates the ensemble results of ten runs. Base† and DATN-2† denotes the average results of conducting ten-fold cross validation on the whole dataset for fair comparison, and here for the source and target tasks in DATN-2†, we use the same training data. For E1, Rank1 and Rank2 are the top two systems from the official leadboard; For E2, Rank1 and Rank2 are from (Zhou et al., 2016, 2018a).

sarial losses by (Liu et al., 2017) as shown in Fig 1.b; 5) *SP*, *DATN-1* and *DATN-2*, the Shared-Private model and two variants of our Dual Attention Transfer Network as shown in Fig 1.c and Fig 1.d.

In Table 3, we report the comparison results between our method and the baseline systems. It can be easily observed that 1) for transfer learning, although the performance of *SP* is similar to or even lower than some baseline systems, our proposed dual attention models, i.e., *DATN-1* and *DATN-2*, can generally boost *SP* to achieve the best results. To investigate the significance of the improvements, we combine each model's predictions of all emotion labels followed by treating them as a single label, and then perform McNemar's significance tests (Gillick and Cox, 1989). Finally, we verify that for English, *DATN-1* is significantly better than *Base*, *FT*, *FS* and *SP*, while *DATN-2* is significant better than all the methods except *APSP*; for Chinese, *DATN-1* and *DATN-2* are significantly better than all the compared methods. 2) Even compared with the state-of-the-art systems in E1 which also employ other external resources, including the affective embedding, emotion lexicon and sentiment classification datasets (Baziotis et al., 2018), the ensemble results of *DATN-2* can achieve slightly better performance; in addition, it is clear that our model can obtain the best performance in E2.

Furthermore, to obtain a better understanding of the advantage of our method, we choose one sentence from the test set of E1, and visualize the attention weights obtained by *Base* and *DATN-2* in Fig 3. We can see that *Base* pays more attention to those frequent emotion words while ignoring the less frequent but important emoji, and thus fails to predict the *love* emotion implied by the emoji. In contrast, with the proposed dual attention mechanism, *DATN-2* makes correct predictions since it can respectively capture the general sentiment words and the emotion-specific emojis.

4 Conclusion

In this paper, we proposed a dual attention-based transfer learning approach to leverage sentiment classification to improve the performance of multi-label emotion classification. Using two benchmark datasets, we show the effectiveness of the proposed transfer learning method.

Acknowledgments

We would like to thank Aletta Hiemstra, Andrés Monroy-Hernández and three anonymous reviewers for their valuable comments.

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Christos Baziotis, Nikos Athanasiou, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. 2018. NTUA-SLP at SemEval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning. *arXiv preprint arXiv:1804.06658*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Laurence Gillick and Stephen J Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *International Conference onAcoustics, Speech, and Signal Processing*.

Huihui He and Rui Xia. 2018. Joint binary neural network for multi-label learning with applications to emotion classification. *arXiv preprint arXiv:1802.00891*.

Yanghoon Kim, Hwanhee Lee, and Kyomin Jung. 2018. Attnconvnet at semeval-2018 task 1: Attention-based convolutional neural networks for multi-label emotion classification. *arXiv preprint arXiv:1804.00831*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li Li, Houfeng Wang, Xu Sun, Baobao Chang, Shi Zhao, and Lei Sha. 2015a. Multi-label text categorization with joint learning predictions-as-features method. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Shoushan Li, Lei Huang, Rong Wang, and Guodong Zhou. 2015b. Sentence-level emotion classification with label and context dependence. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Saif M Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *SemEval*.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *SemEval*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*.

Changqin Quan and Fuji Ren. 2010. Sentence emotion analysis and recognition based on emotion words using ren-cecps. *International Journal of Advanced Intelligence*.

Xiaojun Quan, Qifan Wang, Ying Zhang, Luo Si, and Liu Wenyin. 2015. Latent discriminative models for social emotion detection with emotional dependency. *ACM Transactions on Information Systems (TOIS)*, 34(1):2.

Yaqi Wang, Shi Feng, Daling Wang, Ge Yu, and Yifei Zhang. 2016. Multi-label chinese microblog emotion classification via convolutional neural network. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*.

Yichen Wang and Aditya Pal. 2015. Detecting emotions in social media: A constrained optimization approach. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Jianfei Yu, Minghui Qiu, Jing Jiang, Jun Huang, Shuangyong Song, Wei Chu, and Haiqing Chen. 2018. Modelling domain relationships for transfer learning on retrieval-based question answering systems in E-commerce. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*.

Deyu Zhou, Yang Yang, and He Yulan. 2018a. Relevant emotion ranking from text constrained with emotion relationships. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Deyu Zhou, Xuan Zhang, Yin Zhou, Quan Zhao, and Xin Geng. 2016. Emotion distribution learning from texts. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018b. Emotional chatting machine: emotional conversation generation with internal and external memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Learning Sentiment Memories for Sentiment Modification without Parallel Data

Yi Zhang, Jingjing Xu, Pengcheng Yang, Xu Sun
MOE Key Lab of Computational Linguistics, School of EECS, Peking University
`{zhangyi16,jingjingxu,yang_pc,xusun}@pku.edu.cn`

Abstract

The task of sentiment modification requires reversing the sentiment of the input and preserving the sentiment-independent content. However, aligned sentences with the same content but different sentiments are usually unavailable. Due to the lack of such parallel data, it is hard to extract sentiment independent content and reverse the sentiment in an unsupervised way. Previous work usually can not reconcile sentiment transformation and content preservation. In this paper, motivated by the fact the non-emotional context (e.g., "staff") provides strong cues for the occurrence of emotional words (e.g., "friendly"), we propose a novel method that automatically extracts appropriate sentiment information from the learned sentiment memories according to the specific context. Experiments show that our method substantially improves the content preservation degree and achieves the state-of-the-art performance. [1]

1 Introduction

Sentiment modification of natural language texts is a special task that connects sentiment analysis and natural language generation. It facilitates many NLP applications, such as news rewriting and automatic conversion of review attitude, which reduce the human effort. Sentiment modification presents two requirements: one is that the sentiment or the attitude of the text needs to be transformed to the opposite; the other is that the transformed text should maintain semantic relevance to the input text as much as possible.

Recently, there have been some researches which focus on the work of editing a sentence to alter specific attributes, like style and sentiment (Shen et al., 2017; Hu et al., 2017). Typically, the parallel data with the same content but different sentiment is not available. This line of work attempts to extract the attribute-independent content from a dense sentence representation by adversarial learning. However, it is hard to extract the attribute-independent content in such implicit ways, which makes these methods tend to generate input-irrelevant texts.

Most existing methods can not reconcile the performance of sentiment transformation and content preservation. Direct replacement of emotional words can keep the context but may lead to low-quality sentences. For example, given an input "*The food is cold like rock*", this method probably outputs "*The food is warm like rock*". State-of-the-art models using neural networks struggle to generate high-quality sentences. However, these models usually lead to poor content preservation. For instance, when the source text is "*This is a wonderful movie*", we expect an output like "*This movie is disappointing*". However, the generated sentence may be "*The waiters are very rude*", which has little relevance to the source text. In general, it is difficult to preserve semantic content and reverse the sentiment at the same time without parallel data.

To address this problem, we propose a novel model which performs well in both sentiment transformation and content preservation. Our model first learns two kinds of sentiment memories by explicitly separating emotional words. Then, according to the specific context, the model extracts appropriate sentiment information from the memory of target sentiment. The decoder takes the extracted memory and the context representation together to perform decoding. The overview of our model is shown in Figure 1. The main architecture of our model is a **S**entiment-**M**emory based **A**uto-**E**ncoder (**SMAE**). The proposed model achieves the state-of-the-art perfor-

---

[1] The code is available at `https://github.com/lancopku/SMAE`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1103–1108
Brussels, Belgium, October 31 - November 4, 2018.

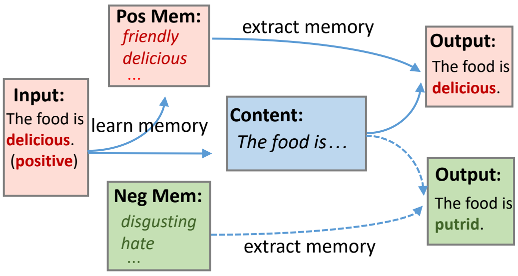

Figure 1: Illustration of the proposed model with a positive input. Solid and dashed lines indicate the training process and the testing process, respectively. The process with a negative input is in a similar way.

mance, especially improves content preservation degree by a large margin.

Our contributions are concluded as follows:

- We propose a method that uses sentiment memories to accomplish sentiment modification without any help of the parallel data.
- The proposed method improves the content preservation degree by a large margin when compared with current systems.

2 Related Work

Recently, there has been some studies for sentiment modification. Shen et al. (2017) learn an encoder that maps a sentence with its original style to a style-independent content representation. This is then passed to a style-dependent decoder for rendering. Fu et al. (2017) implement a multi-decoder auto-encoder (Bengio et al., 2009; Dai and Le, 2015) where the encoder is used to capture the content and the sentiment-specific decoders are used to generate target sentence. Hu et al. (2017) augment the unstructured variables z in vanilla VAE with a set of structured variables c each of which targets a salient and independent semantic feature of sentences, to control sentence sentiment. However, all of these work attempt to implicitly separate the non-emotional content from the emotional information in a dense sentence representation. Xu et al. (2018) explicitly filter out emotional words. They use two sentiment-specific decoders to attach sentiments to non-emotional context. The decoders bear all the burdens to generate sentiments. In our model, we use sentiment memories to assist generating sentiments with only one decoder, which results in fewer parameters.

The proposed sentiment-memory based auto-encoder (Bengio et al., 2009; Ma et al., 2018b) learns the idea of memory network (Weston et al., 2014; Sukhbaatar et al., 2015) but simplifies the process. Our work is also related to the generation tasks (Wang et al., 2017; Liu et al., 2018; Ma et al., 2018a; Lin et al., 2018). These tasks usually generate texts that preserve main information of input texts.

3 Proposed Model

We first use a variant of self-attention(Lin et al., 2017; Kim et al., 2017) mechanism to distinguish the emotional and non-emotional words. Then the positive words and negative words are used to update the corresponding memory modules. Finally, the decoder uses the target sentiment information extracted from the memory and the content representation to perform decoding.

3.1 Emotional Words Detection Model

We first find the emotional words that have the most discriminative power for sentiment polarity. This work is done by training a sentiment classifier with a simple self-attention mechanism. Here the sequence of inputs $\{h_1, ..., h_T\}$ are the hidden states of a LSTM, running over the words in the source sentence $\{x_1, ..., x_T\}$. The context vector can then be computed using a simple sum:

$$\boldsymbol{c} = \sum_{t=1}^{T} a_t \cdot \boldsymbol{h}_t \tag{1}$$

where a_t denotes the attention weight of the t-th word. The sentence vector $\boldsymbol{c}$ is then fed into a fully connected layer to predict the sentiment polarity of the source text. Since the words with obvious emotional tendencies will be given greater weights compared to those non-emotional words during training, a_t can be used to distinguish between emotional and non-emotional words.

The weights of standard attention mechanism sum to 1. When there are several emotional words, the sum 1 is distributed by these words. However, we expect that each emotional word has a weight close to 1 to identify its sentiment attribute. Hence, following (Kim et al., 2017), we modify the calculation of attention weights as follows to get more distinguishable weights:

$$a_t = \mathrm{sigmoid}(\boldsymbol{v}^T \boldsymbol{h}_t) \tag{2}$$

where $\boldsymbol{v}$ is the parameter vector. The sigmoid function follows our intention that giving each input word a distinguishable weight which is close to 1 or 0. However, these weights falls between 0 and 1. They still can not thoroughly distinguish the emotional words from non-emotional words without redundant information. Following Xu et al. (2018), we map attention weights to discrete values, 0 or 1, and we adopt their discrete method. The weights greater than the averaged attention value are assigned to 1 and the weights less than the averaged attention value are assigned to 0. The weight a_t after discretization is denoted as $\hat{a}_t$. Then, $\hat{a}_t$ can be regarded as the emotional word identifier. $1 - \hat{a}_t$ becomes non-emotional word identifier.

3.2 Sentiment-Memory Based Auto-Encoder

After the separation of emotional and non-emotional words, the proposed **SMAE** is used to process these two kinds of information. We employ the seq2seq based auto-encoder. Both the encoder and the decoder are LSTM networks (Hochreiter and Schmidhuber, 1997).

If $\boldsymbol{x}_i$ is a context word, then $\hat{a}_i$ is 0, causing $(1-\hat{a}_i)\boldsymbol{x}_i$ to be $\boldsymbol{x}_i$. Therefore, the sequence $\{(1-\hat{a}_1)\boldsymbol{x}_1, \cdots, (1-\hat{a_T})\boldsymbol{x}_T\}$ can be regarded as non-emotional word embedding sequence. It is fed into the LSTM encoder sequentially. we select $\boldsymbol{h}_T$ in the last state tuple $(\boldsymbol{h}_T, \boldsymbol{c}_T)$ of the encoder as the content representation of the input.

Meanwhile, the embeddings of the emotional words of the source text are used to update the sentiment-memory. Since we have two kinds of sentiments, positive and negative, we use $\boldsymbol{M}^{pos} \in \mathbb{R}^{e \times \gamma}$ and $\boldsymbol{M}^{neg} \in \mathbb{R}^{e \times \gamma}$ to denote the positive memory and the negative memory, respectively. e is the embedding size and γ is a hyper-parameter which controls the size of the memory.

We illustrate the following part by using positive input as an example. We first sum the embedding of the emotional words to get a vector representation of the emotional information, which is denoted as $\boldsymbol{s}^{pos} \in \mathbb{R}^e$. We then use a simple attention mechanism to find the columns in $\boldsymbol{M}^{pos}$ that are most closely related to the emotional information. The outer product of the transposition of emotional information $\boldsymbol{s}^{pos}$ and the attention weights $\boldsymbol{w}$ broadcasts the sentiment vector $\boldsymbol{s}^{pos}$ to a matrix. Then, the matrix is added to the existing memory $\boldsymbol{M}^{pos}$. Due to the attention weight $\boldsymbol{w}$, the columns that are most closely related to the emotional information are updated more with the sentiment information $\boldsymbol{s}^{pos}$. Formally, we have:

$$\boldsymbol{s}^{pos} = \sum_{i=1}^{T} \hat{a}_i \cdot \boldsymbol{x}_i \tag{3}$$

$$\boldsymbol{w} = softmax\left((\boldsymbol{s}^{pos})^T \boldsymbol{M}^{pos}\right) \tag{4}$$

$$\boldsymbol{M}^{pos} = \boldsymbol{M}^{pos} + \boldsymbol{s}^{pos} \otimes \boldsymbol{w} \tag{5}$$

where $\otimes$ denotes the outer product.

Previous work employ two sentiment-specific decoders to generate text based on the supposed non-emotional representation. The decoders bear all the burdens to generate sentiments. In our model, we extract some sentiment information from the sentiment-memories to assist decoding. Intuitively, the context word “staff” is more likely to be associated with the emotional word “friendly”, and “food” is more likely to be associated with “delicious”. So we use the context vector $\boldsymbol{s}^{con}$ to extract the corresponding sentiment memory that is more likely to be used in the future decoding. The context vector $\boldsymbol{s}^{con}$ is represented as the sum of the embedding of non-emotional words. Then $\boldsymbol{s}^{con}$ is used to compute the attention weights $\boldsymbol{u}$ over the columns of sentiment memory matrix. We sum these weighted columns as the extracted memory $\tilde{\boldsymbol{m}}$ and add $\tilde{\boldsymbol{m}}$ to the last cell state $\boldsymbol{c}_T$ of the encoder:

$$\boldsymbol{s}^{con} = \sum_{i=1}^{T} (1 - \hat{a}_i) \cdot \boldsymbol{x}_i \tag{6}$$

$$\boldsymbol{u} = softmax\left((\boldsymbol{s}^{con})^T \boldsymbol{M}^{pos}\right) \tag{7}$$

$$\tilde{\boldsymbol{m}} = \sum_{j=1}^{\gamma} u_j \cdot \boldsymbol{M}_j^{pos} \tag{8}$$

$$\tilde{\boldsymbol{c}}_T = \boldsymbol{c}_T + W \tilde{\boldsymbol{m}} \tag{9}$$

where u_j denotes the j-th value in vector $\boldsymbol{u}$, $\boldsymbol{M}_j^{pos}$ denotes the j-th column of $\boldsymbol{M}^{pos}$ and W is the parameter matrix. The new tuple $(\boldsymbol{h}_T, \tilde{\boldsymbol{c}}_T)$ then acts as the initial state of the decoder.

The negative input is processed in the same way. At the training stage, the decoder is encouraged to restore the source text. Therefore, the cross entropy loss function is optimized.

4 Experiments

4.1 Data Preprocessing

We use the Yelp Review Dataset (Yelp) provided by Yelp Dataset Challenge[2] to conduct experiments. Each item is a sentence from the review

---

[2] https://www.yelp.com/dataset/challenge

Model	ACC	BLEU
CEA	71.96	2.77
MAE	74.59	5.45
SMAE	**76.64** (+2.05)	**24.00** (+18.55)

Table 1: Performance of the proposed method and state-of-the-art systems.

on Yelp and is labeled as having either negative or positive sentiment. We train a CNN sentence classifier (Kim, 2014) to filter examples with ambiguous sentiment polarities (category probability $<$ 0.8). The processed dataset contains 510K, 20K, and 20K pairs for training, validation, and testing, respectively. The classifier achieves an accuracy of 94% on the processed dataset and is also used to test transformation accuracy.

4.2 Experiment Settings

We tune our hyper-parameters on the development set. The word embeddings are initialized randomly with a size of 128. The hidden size of the sentiment-memory based auto-encoder is 300. We use Adam optimizer (Kingma and Ba, 2014) with an initial learning rate set to 0.001 to train our model and the batch size is set to 64. The hyper-parameter γ which controls the size of memory matrix is 60.

4.3 Baselines

We compare our proposed method with two state-of-the-art systems that have been used for sentiment modification. We run the released code on our dataset.

Cross-aligned Auto-Encoder (CAE): This system, proposed by Shen et al. (2017), uses a shared latent content space across different sentiments and leverages refined alignment of latent representations to perform sentiment modification.

Multi-decoder Auto-Encoder (MAE): This system is proposed by Fu et al. (2017). They use a multi-decoder seq2seq model (Bengio et al., 2009; Dai and Le, 2015) where the encoder captures content information by adversarial learning (Goodfellow et al., 2014) and the sentiment-specific decoders are used to generate target sentences.

4.4 Results and Discussions

We use ACC to denote the transformation accuracy. Following Gan et al. (2017), we also compute BLEU (Papineni et al., 2002) between the output and the source text to evaluate the content preservation degree. A high BLEU score primarily indicates that the system can correctly preserve content by retaining the same words from the source sentence.

Model	Sentiment	Content	Fluency
CAE	6.55	4.46	5.98
MAE	6.64	4.43	5.36
SMAE	6.57	5.98	6.69

Table 2: Results of human evaluation.

Input: *Very helpful and informative staff!*
CAE: *Worst service ever.*
MAE: *Very nice here and poor!*
Proposed: *Very rude and careless staff !*

Input: *I will never go here again.*
CAE: *I love this place here!*
MAE: *I had say this place here.*
Proposed: *I will never go anywhere else.*

Input: *The worst and would never recommend anyone to use them.*
CAE: *The best place I 've been to go here!*
MAE: *The first experience is so happy and nice.*
Proposed: *The best and would definitely recommend anyone to use them.*

Table 3: Examples generated by the proposed method and baselines. In comparison, our model changes the sentiment of inputs with higher semantic relevance.

The experimental results of our proposed model and the baselines are shown in Table 1. Both baseline models have low BLEU score but high accuracy, which indicates that they may be trapped in a situation that they simply output a sentence with the target sentiment regardless of the content. The main reason is that these methods using adversarial learning attempt to implicitly separate the emotional information from the context information in a sentence vector. However, without parallel data, it is difficult to achieve such a goal. Our proposed SMAE model takes advantage of self-attention mechanism and explicitly removes the emotional words, leading to a significant improvement of content preservation and the state-of-the-art performance in terms of both metrics.

We also involve human evaluation to measure the quality of generated text. Each item contains an input and three outputs generated by different systems. Then 200 items are distributed to 2 annotators with linguistic background. The annota-

Models	ACC	BLEU
SMEA	76.64	24.00
SMEA (w/o memories)	14.08	26.09

Table 4: Ablation test of memory module.

The staff here is **very rude**.
It really is n't **worth coming** here .
Very pleased with this business.
Been here once and **loved** going **here**.

Table 5: The effectiveness of the memory module with examples. The red words are absent in the input but generated with the help of sentiment memories.

tors have no idea about which system the output is from. They are asked to score the output on three criteria on a scale from 1 to 10: the transformed sentiment degree, the content preservation degree, and the fluency. Table 2 shows the evaluation results. Our model has obvious advantage over the baseline systems in content preservation, and also performs well in other aspects.

Several randomly selected examples generated by different models are shown in Table 3. These examples clearly show our proposed model can generate sentences that are more semantically relevant to the input text compared to the baselines.

4.5 Effectiveness of Sentiment-Memories

To verify the effectiveness of the memory module of our model, we conduct ablation study by excluding the sentiment-memory module. The result is shown in Table 4. According to the result, the complete model achieves an improvement of 62.56% on transformation accuracy over the model that excludes the sentiment memories, which means the sentiment memories are key components to ensure successful sentiment modification. In addition, several examples are shown in Table 5 to visually demonstrate the effectiveness of the memory module. we can find that the proposed model is capable of generating appropriate emotional words (red words in Table 5) to adapt different contexts.

4.6 Error Analysis

To better interpret our model, we also analyze the failure examples whose sentiments are not transformed. We observe that in most cases, these inputs do not have emotional tendencies. Although we have filtered the sentiment-ambiguous examples in preprocessing, there are still a few ambiguous inputs such as “What can I say ?” and “Been here twice.”. Since our model tries to preserve non-emotional content. These words are easily kept and then the decoder barely depends on sentiment-memories. Thus, it is difficult to handle the sentiment transformation with these examples.

5 Conclusion

In this paper, we propose a model that first learns sentiment memories without parallel data and then automatically extracts sentiment information to adapt different contexts when decoding. Experimental results show that our method substantially improves the content preservation degree and achieves the state-of-the-art performance.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028). We thank all the reviewers for providing the constructive suggestions. Xu Sun is the corresponding author of this paper.

References

Yoshua Bengio et al. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3079–3087.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2017. Style transfer in text: Exploration and evaluation. *CoRR*, abs/1711.06861.

Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. 2017. Stylenet: Generating attractive visual captions with styles. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 955–964.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. *CoRR*, abs/1703.00955.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. *CoRR*, abs/1702.00887.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Junyang Lin, Xu SUN, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 163–169.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Shuming Ma, Xu Sun, Junyang Lin, and Xuancheng Ren. 2018a. A hierarchical end-to-end model for jointly improving text summarization and sentiment classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4251–4257.

Shuming Ma, Xu SUN, Junyang Lin, and Houfeng WANG. 2018b. Autoencoder as assistant supervisor: Improving text representation for chinese social media text summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 725–731.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. *CoRR*, abs/1705.09655.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *CoRR*, abs/1503.08895.

Kexiang Wang, Tianyu Liu, Zhifang Sui, and Baobao Chang. 2017. Affinity-preserving random walk for multi-document summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 210–220.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

Jingjing Xu, Xu SUN, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 979–988.

Joint Aspect and Polarity Classification for Aspect-based Sentiment Analysis with End-to-End Neural Networks

Martin Schmitt[1], **Simon Steinheber**[1,2], **Konrad Schreiber**[2] **and Benjamin Roth**[1]

[1]Center for Information and Language Processing, LMU Munich, Germany
[2]MaibornWolff GmbH, Munich, Germany
`martin@cis.lmu.de`

Abstract

In this work, we propose a new model for aspect-based sentiment analysis. In contrast to previous approaches, we jointly model the detection of aspects and the classification of their polarity in an end-to-end trainable neural network. We conduct experiments with different neural architectures and word representations on the recent GermEval 2017 dataset. We were able to show considerable performance gains by using the joint modeling approach in all settings compared to pipeline approaches. The combination of a convolutional neural network and fasttext embeddings outperformed the best submission of the shared task in 2017, establishing a new state of the art.

1 Introduction

Sentiment analysis (Pang and Lee, 2008) is the automatic detection of the sentiment expressed in a piece of text. Typically, this is modeled as a classification task with at least two classes (positive, negative), sometimes extended to three (neutral) or more fine-grained categories. Aspect-based sentiment analysis (ABSA) aims at a finer analysis, i.e. it requires that certain aspects of an entity in question be distinguished and the sentiment be classified with regard to each of them. An example can be seen in Figure 1.

German: Alle so “Yeah, Streik beendet” Bahn so “Okay, dafür werden dann natürlich die Tickets teurer” Alle so “Können wir wieder Streik haben?”

Translation: Everybody’s like “Yeah, strike’s over” Bahn goes “Okay, but therefore we’re going to raise the prices” Everybody’s like “Can we have the strike back?”

General sentiment:	neutral
Aspect sentiment:	Ticket purchase:negative General:positive

Figure 1: Example sentence with contained aspects and their polarity.

This introduces several new challenges. First, labeled data, which are needed to train statistical models, are more difficult to obtain. Therefore the amount of available training data is limited. Thus a good model for ABSA has to make the best possible use of the available data. Second, the detection of the subset of aspects that occur in a given piece of text is non-trivial. Errors introduced at this stage severely limit the performance on the overall ABSA task. Third, the general sentiment and the sentiment of each aspect can each be completely different from each other (cf. Figure 1). This means that a model has to be able to distinguish aspects in the text and make independent decisions for each of them.

We want to address each of these challenges by (1) leveraging unlabeled data by modeling word representations and (2) modeling aspect detection and classification of their polarity jointly in an end-to-end trainable system.

We evaluate our approach on the GermEval 2017 data, i.e. customer reviews about *Deutsche Bahn AG* on social media. We particularly address subtask C as the typical setting where two pieces of information have to be detected from raw text:

1. Which aspects are mentioned?
2. For each mentioned aspect, what is the polarity of its sentiment?

From the new state-of-the-art results we obtain, we conclude that modeling of word representations and joint modeling of aspects and polarity have not yet received the attention they deserve.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1109–1114
Brussels, Belgium, October 31 - November 4, 2018.

2 Related Work

Two recent shared tasks address ABSA: SemEval 2016 Task 5 (Pontiki et al., 2016) and GermEval 2017 (Wojatzki et al., 2017). The SemEval dataset is extremely small. The English laptop reviews, e.g., only contain 395 training instances for the prediction of 88 aspect categories and their polarities. Because of this sparsity, top-ranked systems rely on feature engineering and hand-crafted rules. GermEval is a larger dataset (~20K training instances, 20 aspect categories) and thus suited for our goal of evaluating the quality of fully automatic methods for learning aspect and polarity predictions. Furthermore the top systems at SemEval 2016, XRCE (Brun et al., 2016) and IIT-TUDA (Kumar et al., 2016), not only rely heavily on feature engineering but also separate the tasks of aspect detection and aspect polarity classification into two different parts of their pipeline.

The winners of GermEval 2017 rely on neural methods (Lee et al., 2017). They try to link all aspects to a sequence of tokens and model the task as a sequence labeling problem. This leads to problems because some aspects are not assigned to any token but still have to be detected and classified. Our approach always considers the complete document and produces the set of all detected aspects at once. Although Lee et al. (2017) incorporate some aspects of multi-task learning, the prediction of aspect category and polarity remains separated in each of their approaches. In our work, we show that a joint learning of these two tasks achieves better performance. The approach by Lee et al. (2017) also relies more heavily on external sources than ours. While we only collected a corpus of ~113K unlabeled German tweets, Lee et al. (2017) include annotated English data as well as a much larger unlabeled German corpus (Wikipedia, cf. Al-Rfou et al. (2013)) in their setting.

Ruder et al. (2016) propose another neural model for ABSA. Similarly to the approaches mentioned above, they assume that aspects have already been detected by some other system in a pipeline architecture, and they concentrate on polarity classification on the sentence level by unifying information from other sentences on the document level. We compare ourselves to this baseline and show improvements over pipeline approaches.

3 Proposed Model

3.1 Embedding Algorithm

Word2vec skip-gram (Mikolov et al., 2013) is a widely used algorithm to obtain pretrained vector representations for input words. Notably, Lee et al. (2017) use it for their experiments on the GermEval data. FastText (Grave et al., 2017) works in a similar fashion but has the advantage of incorporating subword information in the embedding learning process. So it can not only learn similar embeddings for word forms sharing a common stem but also generate embeddings for unseen words in the test set by combining the learned character ngram embeddings. This can be crucial when dealing with a morphologically rich language such as German. Glove (Pennington et al., 2014) — similar to word2vec — does not incorporate character-level information, but uses global rather than local information to learn its word embeddings.

We have trained each of these embedding learning algorithms on a corpus of ~113K tweets mentioning at least one of `@DB_info` and `@DB_Bahn`, two official accounts of *Deutsche Bahn AG* offering information and replying to questions. We collected these tweets specifically to build a document collection that is closely related to the domain of GermEval 2017. We also included the GermEval training set for the embedding training.

3.2 Pipeline LSTM (baseline)

We compare our proposed approach to the model described in (Ruder et al., 2016). They first encode each sentence with glove word embeddings and a bidirectional LSTM (Hochreiter and Schmidhuber, 1997). Then this output is concatenated with an embedding of the aspect addressed in the current sentence and finally fed in a document-level BiLSTM. As we are dealing with social media texts, our documents are already very short. So we do not split them into shorter units (sentences). Therefore the second hierarchy level of (Ruder et al., 2016), that combines the output of consecutive sentences in a document, is superfluous and omitted in our experiments. In all other aspects — including hyperparameters — we do as (Ruder et al., 2016), i.e. we duplicate a tweet for each aspect detected in it, concatenate an aspect embedding of size 15 to the output of the BiLSTM encoder, use dropout of 0.5 after the embedding layer and after LSTM cells, and apply a gradient clipping norm of 5. As Ruder et al. (2016) rely on the detected aspects to be given at test time, for a realistic comparison, we feed in the aspects as detected by the strong GermEval baseline system based on support vector machines. The so obtained system serves as our first baseline, representing a state-of-the-art pipeline system.

3.3 End-to-end LSTM

We modify the pipeline model described in the last section as follows: the aspect detection is integrated into the neural network architecture permitting an end-to-end optimization of the whole model during training. This is achieved by formatting the classifier output as a vector $\mathbf{z} \in \{0, 1, 2, 3\}^{|A|}$, where A is the set of all 20 aspects (e.g., *General, Ticket purchase, Design, Safety, . . .*). This corresponds to predicting one of the four classes *N/A*, *positive*, *negative* and *neutral* for each aspect. Specifically, we obtain a hidden representation of an input document X in the following manner:

$$\mathbf{v} = DO(BiLSTM(DO(embed(X)))) \tag{1}$$

where $embed \in \{word2vec, glove, fasttext\}$
and DO = dropout (Hinton et al., 2012).
The design choices for the BiLSTM in this step remain the same as in the baseline model.

Then, we transform the feature vector $\mathbf{v}$ extracted from the text X to a score vector $\hat{y}^{(a)}$ for each aspect $a \in A$ and apply softmax normalization:

$$\hat{y}^{(a)} = softmax(W^{(a)}\mathbf{v} + b^{(a)}) \tag{2}$$

where

$$softmax(\mathbf{x})_i = \frac{exp(x_i)}{\sum_{k=0}^{3} exp(x_k)} \quad \text{for } i = 0, \ldots, 3 \tag{3}$$

Thus for each aspect, we predict its presence or absence as well as its polarity in one step:

$$z^{(a)} = \arg\max_i \hat{y}_i^{(a)} \tag{4}$$

The loss is simply the cross entropy summed over all aspects:

$$L(\theta) = \sum_{a \in A} H(y^{(a)}, \hat{y}^{(a)}) \tag{5}$$

with

$$H(y, \hat{y}) = -\sum_i y_i \cdot log(\hat{y}_i) \tag{6}$$

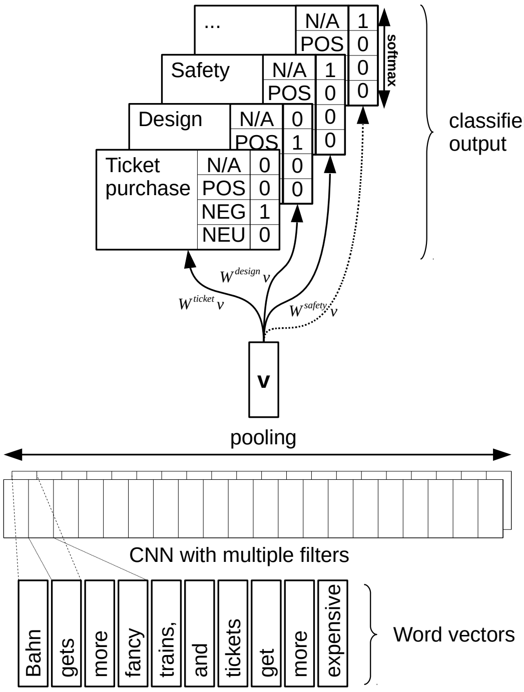

Figure 2: Schematic view of end-to-end CNN architecture.

3.4 End-to-End CNN

Keeping the formalization as an end-to-end task, we replace the BiLSTM by a convolutional neural network (CNN) as described in (Kim, 2014). As in their setting *CNN-non-static*, we use 300-dimensional word embeddings, a max-over-time pooling operation, filter sizes of $3, 4, 5$, and dropout with a rate of 0.5 (as before). We use ReLu ($f(x) = max(0, x)$) as our activation function, and 300 filters of each size, a number also found in related work on sentiment analysis (dos Santos and Gatti, 2014). Following (Kim, 2014), we do not apply dropout after the embedding layer:

$$\mathbf{v} = DO(CNN(embed(X))) \tag{7}$$

With Equation 7 replacing Equation 1, the aspect-wise classification for the end-to-end CNN then follows the same definitions as described in the previous section.

3.5 Pipeline CNN

In order to compare the effects of joint end-to-end and pipeline approaches across neural architectures, we also include an experiment where the CNN model from the previous section replaces the BiLSTM in the pipeline setting described in section 3.2.

4 Experiments

We conduct our experiments on the GermEval 2017 data (Wojatzki et al., 2017), i.e. customer feedback about *Deutsche Bahn AG* on social media, microblogs, news, and Q&A sites. The data were collected over the time of one year and manually annotated, resulting in a main dataset of about 26K documents, divided into a training, development, and test set using a random 80%/10%/10% split. About 1,800 documents from the last 3 months of the data collection period constitute a diachronic test set that can be used to test the robustness of a system over time. We keep the proposed data split and filter out training instances where the same aspect category was assigned two different polarity classes (which affects approximately 4% of the data). The development and test data remain the same.

We choose our hyperparameters based on the development data using the following procedure: we train initial models with a hyperparameter setting based on values we found in the literature, stochastic gradient descent with a learning rate of 0.01 (as in dos Santos and Gatti (2014)) and a mini-batch size of 10 (as in Ruder et al. (2016)). For the best-performing CNN and LSTM architectures (end-to-end + fasttext), we then refine the learning rate and batch size on the development data using random search in the range $\{0.001, 0.003, 0.01, 0.03, 0.1\}$ for learning rate and $\{5, 10, 20\}$ for batch size. For the CNN setting, this results in a learning rate of 0.03 and a batch size of 5 (which we then use for all CNN architectures in the final experiments). For the LSTM setting, this results in a learning rate of 0.01 and a batch size of 10 (which we then use for all LSTM architectures).

Training our models takes between 1-3 minutes per epoch on a GeForce GTX 1080 GPU, the end-to-end CNN being the fastest model to train.

5 Discussion

Aspect polarity Table 1 shows the results of our experiments, as well as the results of our strong baselines. Note that the majority class baseline already provides good results. This is due to highly unbalanced data; the aspect category "*Allgemein*" ("*general*"), e.g., constitutes 61.5% of the cases. This imbalance makes the task even more challenging.

Over all architectures, we observe a comparable or better performance when using fasttext embeddings instead of word2vec or glove. This backs our hypothesis that subword features are important for processing the morphologically rich German language. Leaving everything else unchanged, we can furthermore see an increase in performance for all settings, when switching from the pipeline to an end-to-end approach. The best performance (marked in bold) is achieved by a combination of CNN and FastText embeddings, which outperforms the highly adapted winning system of the shared task.

Aspect category only Even though our architectures are designed for the task of joint prediction of aspect category and polarity, we can also evaluate them on the detection of aspect categories only. Table 2 shows the results for this task. First of all, we can see that the SVM-based GermEval baseline model has very decent performance as it is practically on par with the best submission for the synchronic and even outperforms the best submission on the diachronic test set. It is therefore well-suited to serve as input to the pipeline LSTM model we compare with in our main task.

Comparing our architectures, we see again that fasttext embeddings always lead to equal or better performance. And even though we do not directly optimize our models for this task only, our best model (CNN+fasttext) outperforms all baselines, as well as the GermEval winning system.

Impact of domain-specific corpus We compare the domain-specific FastText embeddings to FastText embeddings trained on Wikipedia[1], which is approximately 100 times the size of our domain-specific corpus. We report the results in Table 3. The embeddings trained on Wikipedia show slightly lower performance on the dev set but slightly higher or equal performance on the test sets. We conclude that the main positive impact of FastText stems from its capability to model subword information and that a large domain-independent corpus or a small domain-specific corpus lead to similar performance gains.

6 Conclusion

We have presented a new approach to ABSA. By solving the two classification problems (aspect

---

[1] Downloaded from `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`.

		development set	synchronic test set	diachronic test set
Pipeline LSTM	+ word2vec	.350	.297	.342
End-to-end LSTM	+ word2vec	.378	.315	.383
Pipeline CNN	+ word2vec	.350	.298	.343
End-to-end CNN	+ word2vec	.400	.319	.388
Pipeline LSTM	+ glove	.350	.297	.342
End-to-end LSTM	+ glove	.378	.315	.384
Pipeline CNN	+ glove	.350	.298	.342
End-to-end CNN	+ glove	.415	.315	.390
Pipeline LSTM	+ fasttext	.350	.297	.342
End-to-end LSTM	+ fasttext	.378	.315	.384
Pipeline CNN	+ fasttext	.342	.295	.342
End-to-end CNN	+ fasttext	**.511**	**.423**	**.465**
majority class baseline		–	.315	.384
GermEval baseline		–	.322	.389
GermEval best submission		–	.354	.401

Table 1: Results on the GermEval data, **aspect + sentiment** task. Micro-averaged F1-scores for both aspect category and aspect polarity classification as computed by the GermEval evaluation script. In the bottom part of the table, we report results from (Wojatzki et al., 2017).

		development set	synchronic test set	diachronic test set
End-to-end LSTM	+ word2vec	.517	.442	.455
End-to-end CNN	+ word2vec	.521	.436	.470
End-to-end LSTM	+ glove	.517	.442	.456
End-to-end CNN	+ glove	.537	.457	.480
End-to-end LSTM	+ fasttext	.517	.442	.456
End-to-end CNN	+ fasttext	**.623**	**.523**	**.557**
majority class baseline		–	.442	.456
GermEval baseline		–	.481	.495
GermEval best submission		–	.482	.460

Table 2: Micro-averaged F1-scores for the prediction of **aspect categories only** (i.e. without taking polarity into account at all) as computed by the GermEval evaluation script. The results in the bottom part of the table are taken from (Wojatzki et al., 2017).

	dev	synchr. test	diachr. test
aspect + sent.	.502	.423	.465
aspect only	.610	.544	.571

Table 3: Results of the end-to-end CNN model with fasttext embeddings trained on the German Wikipedia.

categories + aspect polarity) inherent to ABSA in a joint manner, we observe significant performance gains for both of these tasks on the GermEval 2017 data. Our experiments also showed that word representations leveraging subword information are crucial for a challenging task like ABSA in a morphologically rich language, such as German. Furthermore we observed consistently better performance of CNN architectures in otherwise comparable scenarios, which suggests that CNNs cope better with the irregularities of user-written texts on social media, a research question we leave to future work. By establishing a new state of the art in aspect detection and polarity classification, we provide a new practical baseline for future research in this area.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable input. This work was partially supported by the European Research Council, Advanced Grant # 740516 NonSequeToR, and by a Ph.D. scholarship awarded to the first author by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

Caroline Brun, Julien Perez, and Claude Roux. 2016. XRCE at semeval-2016 task 5: Feedbacked ensemble modeling on syntactico-semantic knowledge for aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 277–281.

Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of tricks for efficient text classification. In *EACL (2)*, pages 427–431. Association for Computational Linguistics.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Ayush Kumar, Sarah Kohail, Amit Kumar, Asif Ekbal, and Chris Biemann. 2016. IIT-TUDA at semeval-2016 task 5: Beyond sentiment lexicon: Combining domain dependency and distributional semantics features for aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 1129–1135.

Ji-Ung Lee, Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Ukp tu-da at germeval 2017: Deep learning for aspect based sentiment detection. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 22–29. German Society for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30, San Diego, California. Association for Computational Linguistics.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016. A hierarchical model of reviews for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 999–1005. Association for Computational Linguistics.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78. ACL.

Michael Wojatzki, Eugen Ruppert, Sarah Holschneider, Torsten Zesch, and Chris Biemann. 2017. GermEval 2017: Shared Task on Aspect-based Sentiment in Social Media Customer Feedback. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 1–12, Berlin, Germany.

Representing Social Media Users for Sarcasm Detection

Y. Alex Kolchinski
Stanford University
kolchinski@stanford.edu

Christopher Potts
Stanford University
cgpotts@stanford.edu

Abstract

We explore two methods for representing authors in the context of textual sarcasm detection: a Bayesian approach that directly represents authors' propensities to be sarcastic, and a dense embedding approach that can learn interactions between the author and the text. Using the SARC dataset of Reddit comments, we show that augmenting a bidirectional RNN with these representations improves performance; the Bayesian approach suffices in homogeneous contexts, whereas the added power of the dense embeddings proves valuable in more diverse ones.

1 Introduction

Irony and sarcasm[1] are extreme examples of context-dependence in language. Given only the text *Great idea!* or *What a hardship!*, we cannot resolve the speaker's intentions unless we have insight into the circumstances of utterance – who is speaking, and to whom, and how the content relates to the preceding discourse (Clark, 1996). While certain texts are biased in favor of sarcastic uses (Kreuz and Caucci, 2007; Wallace et al., 2014), the non-literal nature of this phenomenon ensures that there is an important role for pragmatic inference (Clark and Gerrig, 1984).

The current paper is an in-depth study of one important aspect of the context dependence of sarcasm: the author. Our guiding hypotheses are that authors vary in their propensity for using sarcasm, that this propensity is influenced by more general facts about the context, and that authors have their own particular ways of indicating sarcasm. These hypotheses are well supported by psycholinguistic research (Colston and Lee, 2004; Gibbs, 2000; Dress et al., 2008), but our ability to test them at scale has until recently been limited by available annotated corpora. With the release of the Self-Annotated Reddit Corpus (SARC), Khodak et al. (2017) have helped to address this limitation. SARC is large and diverse, and its distribution of users across comments and forums makes it particularly well suited to modeling authors and their relationship to sarcasm.

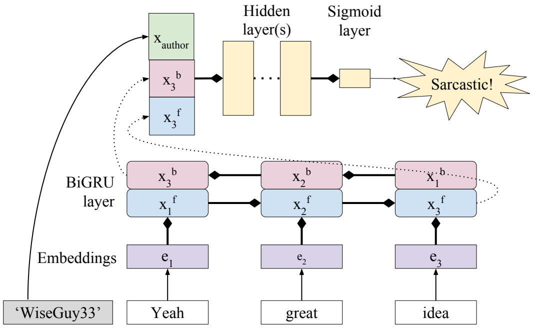

Figure 1: The model architecture. Look-ups are indicated by arrows, dense connections by diamonds. The author embedding can be null (a text-only baseline), a prior reflecting the author's propensity for sarcasm, or a learned embedding. There are potentially multiple layers between the initial example embedding and the output sigmoid layer.

Our core model of comment texts is a bidirectional RNN with GRU cells. To model authors, we propose two strategies for augmenting these RNN representations: a simple Bayesian method that captures only an author's raw propensity for sarcasm, and a dense embedding method that allows for complex interactions between author and text (Figure 1). We find that, on SARC, the simple Bayesian approach does remarkably well, especially in smaller, more focused forums. On the full SARC dataset, author embeddings are able to encode more kinds of variation and interaction with the text, and thus they achieve the highest predictive accuracy. These findings extend and reinforce

---

[1] We use "sarcasm" to include both sarcasm and irony, as the two are generally conflated in the literature we review.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1115–1121
Brussels, Belgium, October 31 - November 4, 2018.

the prior work on user-level modeling for sarcasm (Section 2), and they indicate that simple representation methods are effective here.

2 Previous Work

A substantial literature exists around sarcasm detection. Many of the prior studies focus on the analysis of Twitter posts, which lend themselves well to sarcasm detection with NLP methods because they are available in large quantities, they tend to correspond roughly to a single utterance, and users' hashtags in tweets (e.g., #sarcasm, #not) can provide imperfect but useful labels. A central theme of this literature is that bringing in contextual features helps performance.

González-Ibánez et al. (2011) trained classifiers using a combination of lexical and pragmatic features, including emoticons and whether the user was responding to another tweet (see also Felbo et al. 2017). Bamman and Smith (2015) extend this kind of analysis with additional information about the context. Of special interest here are their contextual features: the author's historical sentiment, topics, and terms; the addressee; and features drawn from historical interactions between the author and addressee. The study finds most features to be useful, but a model trained on the tweet and author features alone achieved essentially the same performance (84.9% accuracy) as a model trained on all features (85.1%).

In a similar vein, Rajadesingan et al. (2015) used a complex combination of features from users' Twitter histories, including sentiment, grammar, and word choice, as inputs into their model, and report a ≈7% gain in classification accuracy upon adding these features to a baseline n-gram classifier.

Recent papers have also applied deep learning methods to detecting sarcastic tweets. Poria et al. (2016) use a combination convolutional–SVM architecture with auxiliary sentiment input features. The architecture of Zhang et al. (2016) includes an RNN, and uses contextual features as well as tweet text for inputs.

Amir et al. (2016) extend the work of Bamman and Smith by generating author embeddings to reflect users' word-usage patterns (but not sarcasm history) in a manner similar to the paragraph vectors introduced by Le and Mikolov (2014). With the inclusion of these embeddings, their convolutional neural network (CNN) achieves a 2% gain in accuracy over that of Bamman and Smith.

Ghosh and Veale (2017) present a combination CNN/LSTM (long short-term memory RNN) architecture that takes as inputs user affect inferred from recent tweets as well as the text of the tweet and that of the parent tweet. When a tweet was addressed to someone by name, the name of the addressee was included in the text representation of the tweet, providing a loose link between interlocutors (West et al., 2014) and a ≈1% gain in performance for some data sets.

There has also been a small amount of previous work on Reddit data for sarcasm (Tay et al., 2018; Ghosh and Muresan, 2018). Wallace et al. (2014) explore a hand-labeled dataset of ≈3K Reddit comments from six subreddits. They report that, when human graders attempted to mark comments as sarcastic or not sarcastic, they needed additional context like subreddit norms and author history roughly 30% of the time, and that the comments which graders found ambiguous were largely the same as those on which a baseline bag-of-words classifier tended to make mistakes. In a follow-up study, Wallace et al. (2015) find that semantic cues for sarcasm differ by subreddit, and they show classifier accuracy gains when modeling subreddit-specific variation.

The work that is closest to our own is that of Hazarika et al. (2018), who also experiment on the SARC dataset. Their model learns author, forum, and text embeddings, and they show that all three kinds of representation contribute positively to the overall performance. We take a much simpler approach to author embeddings and do not include forum embeddings, and we report comparable performance (Section 6). We take this as further indication of the value of author features for modeling sarcasm.

3 The SARC Dataset

The Self-Annotated Reddit Corpus (SARC) was created by Khodak et al. (2017).[2] It includes an unprecedented 533M comments. The corpus is self-annotated in the sense that a comment is considered sarcastic if its author marked it with the "/s" tag. As a result, the positive examples are essentially those which the authors considered ambiguous enough to explicitly tag as sarcastic, meaning that the prediction problem is actually to identify which comments are not only sarcastic but

---

[2] `http://nlp.cs.princeton.edu/SARC/2.0/`

	Comments	% sarcastic
Entire corpus (bal.)	257,082	50.00
r/politics (bal.)	13,668	50.00
r/politics (unbal.)	309,925	3.06
r/AskReddit (bal.)	11,660	50.00
r/AskReddit (unbal)	1,548,803	0.53

Table 1: Basic statistics for SARC.

both sarcastic and not obviously so.

The dataset is filtered in numerous ways, and has good precision (only ≈1% false positive rate) but poor recall (2% false negatives relative to 0.25% true positives, or ≈11% recall). To alleviate the issues caused by low recall, the dataset also includes a balanced sample, where comments are supplied in pairs, both responding to the same parent comment and with exactly one of the two tagged as sarcastic. All comments are accompanied with ancestor comments from the original conversation, author information, and a score as voted on by Reddit users.

This dataset presents numerous advantages for sarcasm detection. For one, it is vastly larger than past sarcasm datasets, which enables the training of more sophisticated models. In addition, most work in sarcasm detection has focused on tweets, which are very short and tend to use abbreviated and atypical language. Reddit comments are not constrained by length and are therefore more representative of how people typically write. Finally, Reddit is organized into topically-defined communities known as subreddits, each of which has its own community norms and linguistic patterns. By making available large amounts of data from a number of subreddits, SARC facilitates the comparative analysis of subreddits, and more generally provides a view into the differences between communities.

Table 1 provides basic statistics on the entire corpus as well as the subreddits that we focus on in our experiments.

4 Models

Our baseline model is a bidirectional RNN with GRU cells (BiGRU; Cho et al. 2014). We tried variants with LSTM cells and did not observe a significant difference in performance. We therefore chose to use GRU cells as the model with fewer parameters.[3]

The inputs to the BiGRU model are users' comments, which are split into words (and in the case of conjunctions, subwords) and punctuation marks and are converted to word vectors. The final states of the two directions of the BiGRU are concatenated with each other and run through either a single fully-connected linear layer or two fully-connected linear layers with a rectified linear unit in between. The output of the final linear layer is fed through a sigmoid function which outputs the estimated probability of sarcasm. This baseline does not take author information into account: for each comment, only the words of the comment are considered as inputs.

The *Bayesian prior* model extends the BiGRU with the sarcastic and non-sarcastic comment counts for authors seen in the training data, which serves as a prior for sarcasm frequency. This version of the model takes as inputs both a representation of the comment and the author representation $x_{\text{author}} \in \mathbb{Z}^2_{\geq 0}$ to estimate the probability of sarcasm. The model can be interpreted as computing a posterior probability of sarcasm given both the comment and the prior of previous sarcastic and non-sarcastic comment counts – author modeling reduced to a Bernoulli prior. For previously unseen authors, x_{author} is set to $(0, 0)$.

The *author embedding* approach extends the baseline BiGRU in a more sophisticated way. Here, each author seen in the training data is associated with a randomly initialized embedding vector $x_{\text{author}} \in \mathbb{R}^{15}$, which is then provided as an input to the model along with a representation of the words of the comment. A special randomly initialized vector x_{UNK} is used for previously unseen authors. The author embeddings are updated during training, with the goal of learning more sophisticated individualized patterns of sarcasm than the Bayesian prior allows. We experimented with training the x_{UNK} vector on infrequently-seen authors (fewer than 5 comments in the training set) instead of using a random vector, and found some suggestions of improved performance. However, as the differences in performance were not substantial enough to change the relative performance of the different models, we report the results for the simpler random-x_{UNK} model.

---

[3] Our models and associated experiment code are available at `https://github.com/kolchinski/reddit-sarc`

5 Experiments

We conducted three sets of experiments, one for each model, to evaluate the effectiveness of the different approaches to author modeling. Each set of experiments was conducted on five datasets: the balanced version of the entire corpus as well as the balanced and unbalanced versions of the r/politics and r/AskReddit subcorpora (Table 1).

In all cases, the raw comment data was tokenized into words and punctuation marks, with components of contractions treated as individual words. We mapped tokens to FastText embedding vectors which had been trained, using subword infomation, on Wikipedia 2017, the UMBC webbase corpus, and the statmt.org news dataset (Mikolov et al., 2018). While vectors existed for nearly 100% of tokens generated, exceptions were mapped to a randomly initialized UNK vector.

All models were trained with early stopping on a randomly partitioned holdout set of either 5% of the data for balanced subreddit corpora or 1% for the others. The performance of the model, as used for hyperparameter tuning, was evaluated against a second holdout set, generated in the same manner as the first holdout set but disjoint from both it and the portion of the data used for training.

Hyperparameters were tuned to maximize model performance as evaluated in this manner, starting with a randomized search process and fine-tuned manually. The final evaluation was conducted against the test set, with a single randomly partitioned holdout set from the training data again used for early stopping. We applied dropout (Srivastava et al., 2014) during training before and between all linear layers. For additional regularization, we also applied an l2-norm penalty to the linear weights but not to the GRU weights.

We attempted other model variations, including multiple GRU layers and an attention mechanism for GRU outputs, but did not observe any gains in performance from the larger models.

6 Results and Discussion

6.1 Quantitive assessment

Table 2 reports the means of 10 runs to control for variation deriving from randomness in the optimization process (Reimers and Gurevych, 2017).

Where there is overlap between our experiments and those of Hazarika et al. (2018) (CASCADE), our model is highly competitive. We slightly under-perform on the full balanced dataset but come out ahead on r/politics. This is striking because our model makes use of much less information. First, unlike CASCADE, we do not have forum embeddings. Second, CASCADE author embeddings involve extensive feature engineering including “stylometric” and “personality” features. Our author embeddings, on the other hand, are either simple empirical estimates (Bayesian priors) or learned embeddings with random initializations, in both cases allowing simpler model specification and training, and more flexibility on the task for which they are used.

There is also evidence that the BiGRU yields better representations of texts than does Hazarika et al.’s CNN-based model. Our ‘No embed’ model is akin to their CASCADE with no contextual features, which achieves only 0.66 on the full balanced corpus and 0.70 on the r/politics balanced dataset. Both numbers are well behind our ‘No embed’. Unfortunately, we do not have space for a fuller study of the similarities and differences between our model and CASCADE.

Both of our methods for representing authors perform well. This is perhaps especially striking for the unbalanced experiments, where the percentage of sarcastic comments is tiny (Table 1). The two methods perform differently on individual forums than on the full dataset. For the r/politics and r/AskReddit communities, the Bayesian priors give the best results. The situation is reversed for the full dataset, where the high-dimensional embeddings outperform the Bayesian priors. This likely reflects two interacting factors. First, with smaller, more focused forums, it is harder to learn good author embeddings, so the simple prior is more reliable. Second, on the full dataset, there are more examples, and also more complex interactions between authors and their texts, so the added representational power of the embeddings proves justified.

6.2 Qualitative comparisons

Table 3 provides example predictions from the different models. Each example is taken from the holdout set of a run in which all three models were trained on the same training set and evaluation was conducted on the same holdout set.

For both sarcastic and non-sarcastic comments, author modeling can be helpful for disambiguation. For instance, in examples 1 and 2, omitting

		r/politics		r/AskReddit	
	Full balanced	balanced	unbalanced	balanced	unbalanced
No embed	74.8 [74.6, 74.9]	74.3 [74.1, 74.6]	58.7 [58.2, 59.1]	64.3 [63.4, 65.2]	56.9 [56.6, 57.2]
Bayesian prior	74.0 [73.7, 74.3]	77.6 [77.4, 77.9]	64.7 [64.6, 64.8]	69.1 [68.8, 69.4]	57.7 [57.6, 57.7]
15d embed	75.3 [74.8, 75.7]	75.1 [74.4, 75.8]	62.0 [59.9, 63.9]	66.0 [65.1, 66.8]	57.1 [56.6, 57.6]
Khodak et al. (2017)	75.8	76.5	27.0	–	–
CASCADE	77.0	75.0	–	–	–

Table 2: Mean macro-averaged F1 scores with bootstrapped 95% confidence intervals, based on 10 runs. CASCADE is the best system of Hazarika et al. (2018), and we report the strongest baseline numbers established by Khodak et al. (2017).

		Model Predictions of p(sarcastic)		
Reddit comment	Sarcastic?	No user rep.	Bayesian	Multidimensional
1. Good thing Trump is going to bring back all those low education high paying jobs.	Yes	.45	.68	.84
2. lol woops!	No	.78	.36	.25
3. The most ubiquitous undergarments I see these days.	Yes	.15	.17	.79
4. Such a deep confession, and it doesn't sound like the guy who wrote it is an asshole at all.	Yes	.33	.45	.86
5. It's not entirely impossible that there are recipe's that have yet to be discovered.	No	.23	.23	.81

Table 3: Examples selected to highlight differences between the models.

author modeling led to incorrect predictions, but including the frequency of the author's sarcasm use alone was enough to change the prediction from incorrect to correct.

In cases like examples 3 and 4, where the Bayesian prior was insufficient, including a model of the author's individualized patterns of sarcasm was much more powerful. That said, the more complex embedding model can misfire, as in example 5, where the simpler models make a correct prediction but it does not. This appeared to happen more for non-sarcastic examples, where the embedding model would occasionally strongly influence the predicted probability of sarcasm upward. Evidently, authors have more individualized patterns of sarcasm than of non-sarcasm.

Judging by the relative performance of the Bayesian and multidimensional-embedding models (Table 2), the multidimensional model wins more disagreements than it loses with the Bayesian model when there is more training data available. However, when there is not, it overfits to such a degree that its predictions of authors' sarcasm patterns are less useful than the Bayesian approach. This suggests a future direction of exploration: the most useful model of all may be one that expands in complexity for authors with more examples available, and shrinks for those who have fewer.

7 Conclusion

This paper evaluated two data-driven methods for modeling the role of the author in sarcasm detection. Both prove effective. As shown by Hazarika et al. (2018), similar techniques can be extended to other aspects of the context. While our experiments did not support adding these representations, we think listeners rely on them as well, so additional computational modeling work here is likely to prove fruitful.

References

Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mário J Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976*.

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *ICWSM*, pages 574–577.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Herbert H. Clark. 1996. *Using Language*. 'Using' Linguistic Books. Cambridge University Press.

Herbert H. Clark and Richard J. Gerrig. 1984. On the pretense theory of irony. *Journal of Experimental Psychology: General*, 113(1):121–126.

Herbert L. Colston and Sabrina Y. Lee. 2004. Gender differences in verbal irony use. *Metaphor and Symbol*, 19(4):289–306.

Megan L. Dress, Roger J. Kreuz, Kristen E. Link, and Gina M. Caucci. 2008. Regional variation in the use of sarcasm. *Journal of Language and Social Psychology*, 27(1):71–85.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

Aniruddha Ghosh and Tony Veale. 2017. Magnets for sarcasm: Making sarcasm detection timely, contextual and very personal. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 482–491.

Debanjan Ghosh and Smaranda Muresan. 2018. " with 1 follower i must be awesome: P". exploring the role of irony markers in irony recognition. *arXiv preprint arXiv:1804.05253*.

Raymond W. Gibbs. 2000. Irony in talk among friends. *Metaphor and Symbol*, 15(1-2):5–27.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. 2018. CASCADE: Contextual sarcasm detection in online discussion forums. ArXiv:1805.06413.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579*.

Roger J Kreuz and Gina M Caucci. 2007. Lexical influences on the perception of sarcasm. In *Proceedings of the Workshop on computational approaches to Figurative Language*, pages 1–4. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. 2016. A deeper look into sarcastic tweets using deep convolutional neural networks. *arXiv preprint arXiv:1610.08815*.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 97–106. ACM.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. ArXiv:1707.09861.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. 2018. Reasoning with sarcasm by reading inbetween. *arXiv preprint arXiv:1805.02856*.

Byron C Wallace, Eugene Charniak, et al. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1035–1044.

Byron C Wallace, Laura Kertz, Eugene Charniak, et al. 2014. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association* *for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 512–516.

Robert West, Hristo S. Paskov, Jure Leskovec, and Christopher Potts. 2014. Exploiting social network structure for person-to-person sentiment analysis. *Transactions of the Association for Computational Linguistics*, 2(2):297–310.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2449–2460.

Syntactical Analysis of the Weaknesses of Sentiment Analyzers

Rohil Verma
MIT
77 Massachusetts Avenue
rohil@mit.edu

Samuel Kim
MIT
77 Massachusetts Avenue
sungil@mit.edu

David Walter
MIT
77 Massachusetts Avenue
dwalter@mit.edu

Abstract

We carry out a syntactic analysis of two state-of-the-art sentiment analyzers, Google Cloud Natural Language and Stanford CoreNLP, to assess their classification accuracy on sentences with negative polarity items. We were motivated by the absence of studies investigating sentiment analyzer performance on sentences with polarity items, a common construct in human language. Our analysis focuses on two sentential structures: downward entailment and non-monotone quantifiers; and demonstrates weaknesses of Google Natural Language and CoreNLP in capturing polarity item information. We describe the particular syntactic phenomenon that these analyzers fail to understand that any ideal sentiment analyzer must. We also provide a set of 150 test sentences that any ideal sentiment analyzer must be able to understand.

1 Introduction

Sentiment analysis of texts, from relatively long product and movie reviews (Pang et al., 2002) to short tweets (Go et al., 2009), is a rich and evolving field. Probabilistic analyzers, such as Google's Natural Language client (Google, 2018) and Stanford's CoreNLP package (Manning et al., 2014), have improved in recent years, but major challenges remain in classifying practical sentences. In this paper, we focus on a particular grammatical phenomena analyzers often misclassify: the presence of specific polarity items.

As one of the fundamental components of natural language, polarity items (e.g. nothing, any, ever) are lexical items that can appear only in specific licensing contexts. Thus, we should be able to identify the grammatical polarity of a sentence by the presence of such polarity items, allowing us to determine its sentiment. However, these licensing contexts are challenging to identify and are generally different for each item (Baker, 1970). We aim to understand how negative polarity items are involved in misclassified sentences and use this knowledge to characterize the syntactic phenomenon an ideal sentiment analyzer must learn.

First, we present a brief background on sentiment analyzers and polarity items. In the next section, we describe our methodology in terms of what kinds of sentences we want to use and how we can best test the sentiment analyzers. Our methodology involves trying sentences with negative polarity items under two different licensing contexts, downward entailment and non-monotone quantifiers. We then evaluate variations of the sentences to show that the sentiment analyzers are not correctly using the polarity items. By exploring these misclassified sentences, we describe the particular syntactic configuration that leads to misclassification, presenting weaknesses in the state-of-the-art sentiment analyzers in understanding and handling polarity items.

2 Background and Previous Work

2.1 Sentiment Analyzers

Most sentiment analyzers (Wang et al., 2012; Pak and Paroubek, 2010; Cambria et al., 2013) are based on a statistical approach, relying on a conglomeration of sentiments of the individual words in a sample. The main assumption behind such statistical approaches is that keywords contain essential information to infer the sentiment of a whole sample. Therefore, this type of statistical approach does not readily consider complex syntactic interactions between individual words; instead, the main focus lies in the system's learning of the relevant knowledge through texts relevant to the sentiment analysis task.

Statistical methods often employ bag-of-words as input features and represent a document by the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1122–1127
Brussels, Belgium, October 31 - November 4, 2018.

summation of all bag-of-words features in that document. A model, such as a maximum entropy classifier or support vector machine (Mullen and Collier, 2004), can be trained to learn which words or combinations of words are relevant for sentiment analysis (Pang et al., 2002). With bag-of-words as the input of a model, we lose spatial structure for a document, so a classifier is incapable of differentiating "I knew the dog would never bite" from "The dog knew the man would never bite" or "Bite never I knew the dog would."

To overcome such challenges, deep learning models have gained popularity for this task. (Glorot et al., 2011) use domain adaptation to train an adversarial network, where two models are pitted against each other: one classifying sentiment and the other creating input documents. This approach allows the system to learn from data sets across multiple domains, increasing the flexibility of the sentiment classifier. However, the weakness of this approach is that the system uses a bigram bag-of-words as input, making it unable to learn long-distance syntactic phenomena.

Recent methods have proposed learning word embeddings (Tang et al., 2014) or applying deep neural architectures (Dos Santos and Gatti de Bayser, 2014) to extract context and sentiment from short texts, as these contain minimal information. Although these techniques have shown performance improvements, they have not been completely successful in capturing long-distance dependencies, leading to the proposal of memory networks (Weston et al., 2014) and attention-based mechanisms (Wang et al., 2016).

2.2 Polarity Items

In Emergence of Meaning (Crain, 2012), Crain puts forward polarity items, like "some" and "any" that are similar, but are sometimes interpreted differently. He presents the example of (1) "John didn't eat any of the kangaroo." and compares it to (2) "John didn't eat some of the kangaroo." The sentence with "some" implies that John did eat a part of the kangaroo, but there is a part of it that he did not eat. The sentence with "any" implies the stronger statement that John did not eat any of the kangaroos. These two interpretations differ due to the polarity of the two words. "Any" is only accepted in negative contexts, so it has negative polarity, whereas "some" can be accepted in both positive and negative contexts and possesses positive polarity. To observe that "any" only works in negative contexts, consider the sentence (3) "John ate any of the kangaroo." which has positive context, and is incorrect with the word "any".

Further, the words "any" and "some" can sometimes be used interchangeably and have the same interpretation. (4) "You'll never convince me that John didn't eat some/any of the kangaroo." contains (1) and (2). In (4) there is negation (never) in a higher clause; then the latter clause contains negation and "any"/"some". These practical examples demonstrate that word ordering in a sentence matters, and that polarity items can exist in complex statements requiring a fundamental understanding of human language for correct interpretation. Therefore, a model just working with bag-of-words or even n-gram features does not appear to be sufficient for practical sentences that require spatial or syntactic understanding.

Polarity items are permitted only within specific licensing contexts, which means they can only occur in specific sentential structures. Our paper explores two licensing contexts: downward entailment and non-monotone quantifiers. Under downward entailment, the sentence acts as a monotone decreasing function such that when parts of the sentence are removed monotonically, the relative strength of a statement monotonically decreases. For example, "nobody moved into the house" implies "nobody moved into the house quickly", so "nobody moved" is a monotone-decreasing phrase. On the other hand, non-monotone quantifiers lack clear downward or upward entailment (Giannakidou, 2002). For example, the phrase "exactly three men never moved" does not entail "exactly three men never moved quickly" and vice-versa, so it has non-monotone entailment. The linguistic phenomenon of licensing contexts and polarity items are a fundamental part of human language, and the metric to measure the performance of a sentiment analyzer should consider how it handles these polarity items.

3 Related Work

To the best of our knowledge, there are no previous studies investigating the weaknesses of Google Natural Language, CoreNLP, or other probabilistic sentiment analyzers in classifying sentences with polarity items.

	Classification accuracy									
	Downward entailment					Non-monotone quantifier				
	a	b	c	d	e	a’	b’	c’	d’	e’
Google NLP	0%	0%	0%	80%	80%	0%	0%	0%	93.3%	86.7%
Stanford CoreNLP	0%	0%	0%	33.3%	53.3%	0%	0%	0%	33.3%	66.7%

Table 1: Summary of classification accuracy on downward entailment and non-monotone quantifiers. Each category indicates 15 test sentences. a: Bill has never done anything [negative adjective], b: [subject phrase] has never done anything [negative adjective], c: Bill has not done anything [negative adjective], d: Bill has done something [negative adjective], e: [subject phrase] has done something [negative adjective]. a’-e’: instead of Bill, we use a non-monotone quantifier (e.g. exactly half).

4 Experiment

4.1 Methodology

We developed 150 test sentences derived from two base sentences, under two different licensing contexts: downward entailment and non-monotone quantifiers. We first tested on Google Cloud Natural Language (Google, 2018) and then repeated the experiment on the Stanford CoreNLP sentiment analyzer (Socher et al., 2013). Each base sentence consists of a subject phrase, a verb phrase, some polarity item(s), and some modifiers (e.g. painful). The rest of the test sentences consist of variations on the base sentences in the specific phrases, polarity items or modifiers used, allowing us to identify the sentence element responsible for misclassification. We chose 15 sentences with minimal sentiment ambiguity per category, allowing us to demonstrate with statistical confidence the inability of these sentiment analyzers to capture syntactic phenomena.

4.2 Results

For both downward entailment and non-monotone quantifiers, we considered two base sentences:

1. (A) has/have [never or not] done anything (B: negative adjectives). These include categories a, b, c, a’, b’ and c’.

2. (A) has/have done something (B: negative adjectives). These include categories d, e, d’, and e’.

For (A), we used subject phrases such as “I” and “Bill and his friends”, and in the case of non-monotone quantifiers, we used subjects such as “exactly half”, “99% of people”, and “exactly ten students”.

4.2.1 Downward entailment

Sentences in category a (eg: “Bill has never done anything terrible”) should be classified as positive. Both Google Natural Language and CoreNLP achieved 0% accuracy on sentences in this category as shown in Table 1. The sentiment analyzers classified these sentences as either negative or neutral. We consider neutral classification incorrect, as the overall sentence is expressing a positive sentiment towards the subject for possessing good moral character. Our varied adjectives reveals that the adjective used is not the reason for misclassfication.

To investigate misclassification further, we varied the subject phrase (A) with fifteen different subject phrases (category b). We observed that sentences such as “Bill has never done anything terrible” and “Bill in my English class has never done anything terrible.” are classified as negative by Google Natural Language and CoreNLP. The magnitude of the negative sentiment remains the same, indicating that minor variations in the subject phrases do not affect the sentiment of the sentence. However, while CoreNLP classifies “Bill has never done anything grumpy” as neutral, it classifies “Sally has never done anything grumpy” as negative. Google Natural Language classifies both as negative, although the score changes slightly. This example demonstrates another weakness of the sentiment analyzers: they treat subject phrases with the same context (“Bill” vs. “Sally”) differently.

Next, we show that the reason for misclassification is not the presence of “never” by varying the sentences to contain “not” (category c; e.g. “Bill has not done anything terrible”). Both Google Natural Language and CoreNLP classify all 15 sentences as negative and yield the same classification accuracy of 0%. We note that for some sentences there is a minor change in the score, but

```
(ROOT
(S
  (NP (NNP Bill))
  (VP (VBZ has)
    (ADVP (RB never))
    (VP (VBN done)
      (S (ADJP (NN anything)
        (JJ terrible)))))
  (.)))
```

Figure 1: Parse tree for “Bill has never done anything terrible.”

```
(ROOT
(S
(ADVP (RB Exactly))
  (NP (NN half))
  (VP (VBD had)
    (ADVP (RB never))
    (VP (VBN done)
      (S
        (ADJP (NN anything)
          (JJ terrible)))))
  (.)))
```

Figure 2: Parse tree for “Exactly half had never done anything terrible.”

the sentiment of sentences remains the same.

We now consider sentences in category (2) that are semantically opposite to sentences in category (1). For example, “Bill has done something terrible” or other variations (types d and e) should be unambiguously classified as negative. Here, Google Natural Language achieves a classification accuracy of 80% (12 out of 15 test sentences) compared to CoreNLP’s 33.3% (5 out of 15 test sentences). For Google Natural Language, it is evident that the element that causes misclassification is the negative polarity item that the analyzer fails to interpret. The analyzer fails to invert the negative sentiment of negative adjectives (B) and thus classifies the sentence as negative or neutral. CoreNLP, however, does poorly on all categories, although it does significantly better on sentences in category (2).

We now consider the parse tree of the original sentence “Bill has never done anything terrible” (Figure 1) to demonstrate the successful parse of “has never done anything terrible” as a single verb phrase. The failure of the analyzers to correctly determine sentiment implies that despite noting the sentence’s hierarchical structure, they are unable to understand long-term dependencies - they lack an understanding of the c-command relation (Radford, 2004).

4.2.2 Non-monotone quantifiers

We repeated the same analysis for non-monotone quantifiers by including non-monotone quantifiers within the subject phrase. For example, “Exactly half had never done anything terrible” and its sister sentences (e.g. “Ninety percent of parents had never done anything terrible”) should be classified as positive. Similar to our findings on downward entailment, both Google Natural Language and CoreNLP achieve a classification accuracy of 0% on the three classes of sentences within category (1): a total of 45 sentences from class (a’), (b’), and (c’) as shown in Table 1. Analyzers again determine “had never done anything terrible” as a single verb phrase, as shown in a parse tree in Figure 2.

For semantically opposite sentences (class (d’) and (e’)), we get a similar result as in downward entailing sentences, although the classification accuracy slightly increases for both analyzers. Again, we consider neutral classification as a misclassification, as the sentence clearly contains either a positive or negative sentiment towards the subject.

We interpret our experiment with non-monotone quantifiers as follows: (1) both CoreNLP and Google Natural Language lack an understanding of the negative polarity item “anything” and (2) the variation of subject phrases results in a minor change in the sentiment. We note that, as in the case of downward entailment, the failure of the analyzers implies an inability to understand the c-command relation (Radford, 2004). We note a limitation of our experiment: the selection of arbitrary adjectives. However, the 15 adjectives used are commonly used to describe humans, and the result was quite consistent. Further, we found coherent results for both downward entailment and non-monotone quantifiers and were able to highlight the lack of long-distance dependency understanding in state-of-the-art analyzers. Finally, we highlight the importance of assessing performance of sentiment analyzers using practical sentences that involve not only negation as in previous studies (Socher et al., 2013), but also polarity items such as the 150 sentences used in our experiment.

5 Discussion

Our experiment indicates that the Stanford CoreNLP sentiment analyzer (Socher et al., 2013) and Google Cloud Natural Language (Google, 2018) do not understand the c-command relation (Radford, 2004). We now argue that this issue is not a result of the training set used, but rather a fundamental inability of the model class. First, we note that long-term dependencies are a well-known weakness of probabilistic models, even those specifically designed to capture them, such as bidirectional RNNs or LSTMs (Zhang et al., 2018). Next, we note that the sentences that we tested are general sentences whose components could be found in any linguistic corpus; in fact, CoreNLP's treebank does contain annotated sentences containing the "... has never ..." or "... has no ..." constructions, and so the model should correctly analyze such sentences. As a result, we draw the conclusion that these models lack an understanding of long-term dependencies, and based on our experiments, they specifically fail to understand the c-command relation (Radford, 2004). Finally, we remind readers that the c-command relation is associated with Chomskyan grammars; and need not be necessary within other models of syntax. Consequently, learning the c-command relation may lead to better analyzers, but it need not be the only way to improve performance.

6 Conclusions

We evaluated two state-of-the-art sentiment analyzers, Stanford CoreNLP (Socher et al., 2013) and Google Cloud Natural Language (Google, 2018), using sentences with negative polarity items under two different licensing contexts: downward entailment and non-monotone quantifiers. Through such analysis, we noted that current analyzers lack a complete understanding of negative polarity items, and by extension, the c-command relation. We have also produced a set of sentences that can be used to test future analyzers. This work can be extended to validate other analyzers, test non-probabilistic sentiment analyzers or build, new improved sentiment analyzers. We have made the set of misclassified sentences available as supplementary material.

7 Acknowledgements

The authors would like to thank Professor Robert C. Berwick for his guidance in preparing this work.

References

C Lee Baker. 1970. Double negatives. *Linguistic inquiry*, 1(2):169–186.

Erik Cambria, Bjorn Schuller, Bing Liu, Haixun Wang, and Catherine Havasi. 2013. Statistical approaches to concept-level sentiment analysis. *IEEE Intelligent Systems*, 28(3):6–9.

Stephen Crain. 2012. *The emergence of meaning*, volume 135. Cambridge University Press.

Cicero Dos Santos and Maira Gatti de Bayser. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *International Conference on Computational Linguistics*.

Anastasia Giannakidou. 2002. Licensing and sensitivity in polarity items: from downward entailment to nonveridicality. *CLS*, 38:29–53.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Alec Go, Lei Huang, and Richa Bhayani. 2009. Twitter sentiment analysis. *Entropy*, 17:252.

Google. 2018. Google Cloud Natural Language. `https://cloud.google.com/natural-language/`. [Online; accessed 22-May-2018].

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

A Radford. 2004. English syntax: An introduction.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, volume 1, pages 1555–1565.

Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. 2012. A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120. Association for Computational Linguistics.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis : A survey.

Is *Nike* female? Exploring the role of sound symbolism in predicting brand name gender

Sridhar Moorthy
Rotman School of Management
University of Toronto
moorthy@rotman.utoronto.ca

Ruth Pogacar
Department of Marketing
Haskayne School of Business
University of Calgary
ruth.pogacar@gmail.com

Samin Khan
Computer Science Program
Cognitive Science Program
University of Toronto
samin.khan995@gmail.com

Yang Xu
Department of Computer Science
Cognitive Science Program
University of Toronto
yangxu@cs.toronto.edu

Abstract

Are brand names such as *Nike* female or male? Previous research suggests that the sound of a person's first name is associated with the person's gender, but no research has tried to use this knowledge to assess the gender of brand names. We present a simple computational approach that uses sound symbolism to address this open issue. Consistent with previous research, a model trained on various linguistic features of name endings predicts human gender with high accuracy. Applying this model to a data set of over a thousand commercially-traded brands in 17 product categories, our results reveal an overall bias toward male names, cutting across both male-oriented product categories as well as female-oriented categories. In addition, we find variation within categories, suggesting that firms might be seeking to imbue their brands with differentiating characteristics as part of their competitive strategy.

1 Introduction

When naming humans, clear gender conventions seem to exist in every society. For example, in the English-speaking world, *Jessica*, *Linda*, and *Nancy* are female names, while *John*, *Michael*, and *William* are male names. In turn, decades of gender-stereotyping research suggest that people associate particular genders with particular characteristics. For example, females are viewed as "warm," "expressive," and "emotional," while males are viewed as "assertive," "competent," and "rational" (Broverman et al., 1972; Spence and Helmreich, 1979). We ask whether any of this applies to commercially-traded brands, and is there a gender strategy underlying brand names? If so, what does this strategy say about firms' motivations in making these choices? Is it driven by product category characteristics or is it driven by competitive strategy considerations within each category?

In this paper, we take the first steps toward answering these questions. We start by developing a machine-learning method to predict the gender of human first names. Large labeled data sets of human first names are available from the U.S. and the U.K. to train such an algorithm. Using various linguistic features of such names—for example, "a" ending, sonorant ending (m, n, ng, l, r), etc.—we find that we can predict human gender quite accurately (approximately 80% success rates). We then use this algorithm to predict the masculinity-femininity of brand names (see Figure 1). For this purpose we have identified another data set where a large number of brand names are available, pre-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1128–1132
Brussels, Belgium, October 31 - November 4, 2018.

classified into many different product categories—detergents, analgesics, health and beauty aids, electronics, etc. Finally, having classified brand names by gender, we ask what the variation in brand name gender, within and across product categories, can tell us about the marketing strategy in brand name choices. By using NLP-methods to analyze brand names, we contribute not only a new application domain for such methods, but also, importantly, advance the branding literature by suggesting that brand name gender may be part of a brand's positioning strategy.

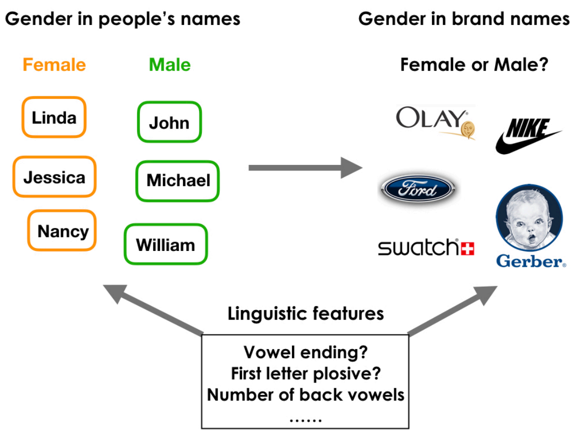

Figure 1: Illustration of our methodology. A gender classifier trained on people's names is applied to brand names.

2 Sound symbolism and gender of names

Existing work from linguistics has suggested that male and female names have different phonological properties. Here we summarize gender-related features of first names identified in the literature: 1) Names ending in the mid-central unstressed vowel sound 'uh' are almost always feminine (Lieberson and Bell, 1992); 2) Names ending in a vowel tend to be feminine (Slater and Feinman, 1985); 3) Names ending in a sonorant (m, n, ng, l, r) may be either masculine or feminine (Barry and Harper, 1995; Lieberson and Bell, 1992; Slater and Feinman, 1985); 4) Names ending in a 'fricative' or 'affricate' produced by restricting airflow through the mouth to create frication (s, z, f, v, h, sh, ch, dj) tend to be masculine (Barry and Harper, 1995); 5) Names ending in a 'plosive' produced by stopping airflow through the mouth (p, b, t, d, k, g) are almost always masculine (Barry and Harper, 1995); 6) Words and names featuring fricatives are more associated with femininity relative to those with plosives, which are associated with masculinity (Folkins and Lenrow, 1966; Guèvremont and Grohmann, 2014; Klink, 2000); 7) The front / back vowel distinction mirrors that of plosives and fricatives. Brand names with front vowels, such as "i," and "ee," seem more feminine than names with back vowels, such as "oh," "oo" (Klink, 2000).

Despite the rich literature on sound symbolism and gender, sparse attempts have been made to systematically examine the contribution of different linguistic features in gender prediction of names at scale (Bird et al., 2009). Although recent work has examined gender issues in web-crawled data (Zhao et al., 2017) and historical corpora (Garg et al., 2018), there has been no study on exploiting linguistic features to predict gender of brand names.

3 Computational methods

Table 1 summarizes the full set of features that we used for modeling. We took orthographic forms of names as a proxy for phonetic forms due to practical scalability. We started by considering names ending in the sound 'uh' as ending with the letter 'a', because 'a' accounts for the vast majority of instances of this sound. After this initial step, we dummy-coded names ending in a vowel with the letters 'a', 'e', 'i', 'o', 'u', and 'y' (i.e., any name ending in a vowel was coded 1 on this feature, whereas all other names were coded 0; see Table 1 for an illustration). We coded names ending in fricatives and plosives with the letters 'p', 'b', 't', 'd', 'k', 'g', 'f', 'v', 's', 'z', 'th', 'sh', 'ch', and 'dge'. We coded sonorant endings based on the letters 'm', 'n', 'ng', 'r', 'and 'l'.

We also considered word-initial occurrences of all the previously mentioned sounds, making symmetric hypotheses that word-initial vowels will predict femininity, word-initial plosives will predict masculinity, and that sonorants and fricatives may be predictive in some way. Coding was based on the same letters with some minor variations (the letters 'w' and 'y' were included with sonorants, and the letter 'j' was included with fricatives).

Finally, we considered the total number of occurrences of all the previously mentioned sounds, hypothesizing that the total number of vowels will predict femininity, total number of plosives will predict masculinity, and total number of sonorants and fricatives may be predictive in some way. Additionally, we consider two distinct categories of vowels: front vowels, represented by the letters 'i', and 'e', which we hypothesize will predict femininity, and back vowels, represented by the letters 'o' and 'u', which we hypothesize will be more predictive of masculinity. We excluded features such as stressed syllable and number of syllables that have been linked with name gender (Slater and Feinman, 1985) because they require manual coding and do not scale. However, we considered total number of letters as a proxy for name length.

Table 1: Features for gender prediction of people's and brand names, with example values for *Linda*.

Linguistic feature	**Value of *Linda***
A ending?	1
Vowel ending (a, e, i, o, u, y)?	1
Fricative ending (f,v,th,s,z,sh,ch,dge)?	0
Sonorant ending (m,n,ng,l,r)?	0
Plosive ending (p,b,t,d,k,g)?	0
1st letter vowel (a, e, i, o, u)?	0
1st letter fricative (f,v,th,s,z,sh,ch,j)?	0
1st letter sonorant (m,n,l,r,w,y)?	1
1st letter plosive (p,b,t,d,k,g)?	0
Number of letters	5
Number of front vowels	1
Number of back vowels	0
Number of vowels (a, e, i, o, u)	2
Number of fricatives (f,v,th,s,z,sh,ch,j)	0
Number of sonorants (m,n,l,r,w,y)	2
Number of plosives (p,b,t,d,k,g)	0

We used three simple models for name gender classification based on the features we have described. For all methods, we considered binary classification (female/male) and excluded "gender-neutral" names that are ambiguous. 1) To examine how each feature contributes to name gender prediction individually, we first considered a single-feature logistic model that determines gender of a name y from one of the features x: $\log \frac{p(y=female)}{p(y=male)} = \beta_0 + \beta_1 x$. 2) To examine how features combine to name gender prediction, we considered a multivariate version of the logistic model. We applied a sparsity constraint with automatic relevance determination (Yamashita et al., 2008) to explore the minimal set of features necessary for gender determination, taking into account the fact that the features are over-complete and not interdependent, e.g. "a ending" entails "vowel ending." We used the default settings on the hyper-parameters in the open-source Python package. [1] 3) We considered random forest as an alternative multivariate classifier, based on decision trees using the Gini impurity criterion and bootstrapped subsamples in ensemble averaging. We used the Python $sklearn$ package for this model.

4 Data

We draw data from three primary sources, two for people's names and one for brand names. For people's names, we relied on databases of U.S. and U.K. names available at `https://github.com/OpenGenderTracking/globalnamedata/tree/master/assets`. The U.S. data come from the yearly birth records maintained by the U.S Social Security Administration from 1880 to 2013; the U.K. data come from the UK Office of National Statistics, the Northern Ireland Statistics and Research Administration, and the Scotland General Register Office. After removing names that are labeled as both male and female, we ended up with 97102 unique English names (60984 female, 36118 male) to work with. For brand names, we relied on Kantar Media's Stradegy database. This database documents U.S. advertising spending by brands in virtually every product category. [2] In this case, after removing multi-word names that are derivative brands (e.g., *Ford Escort*) and a small number (66) of names that are common English words (e.g., *Coach*) based on the $\sim$5000 most frequent words in the British National Corpus, [3] we ended up with 1021 brand names in 17 product categories. We represented each name as a 16-dimensional vector based on the features described, and we made all data available in the supplementary materials.

5 Results

Evaluation. We tested all gender classifiers on the English dataset in a supervised setting. Table 2 summarizes the model performances in a five-fold cross validation with the data initially randomized.

Overall, all models predicted gender of people's names substantially above chance (50%). In particular, the multivariate models performed better than the single-feature model though the difference is small, suggesting that gender information is likely encoded in a restricted set of features. Figure 2 confirms this finding by showing the fitted weights on different features from the sparse logistic classifier. The top four features with the highest weights are "a ending," "plosive ending," "sonorant ending," and "fricative ending," suggesting the dominance of gender information in the ending of names. The same four features also yielded the highest predictive accuracies in the single-feature model except for the feature of "vowel ending" - 75.1%: a end-

---

[1] https://github.com/KamitaniLab/sml

[2] https://www.kantar.com/

[3] http://www.kilgarriff.co.uk/BNClists/lemma.al

ing 65%; plosive ending - 66%; sonorant ending - 72%; fricative ending - 66%.

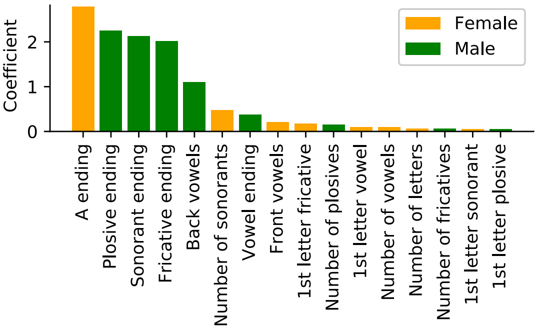

Figure 2: Feature coefficients in logistic regression.

We observed a small difference in predictive accuracy between the random forest model and the sparse logistic model, suggesting the features are relatively robust to classifier variation. We also observed that predictive accuracies on male names tend to be poorer than those on female names, possibly because some male names also end with vowels 27.1%, e.g., *Joshua*), but female names predominantly end with vowels 76.4%).

Table 2: Model accuracies on name gender prediction.

Model	Acc. %	Female %	Male %
Single-feature	75.1	76.4	72.9
Sparse logistic	80.5	83.9	74.9
Random forest	81.8	85.4	75.7

Brand name gender prediction. We applied the three classifiers to the brand name database, by using the weights estimated from the English name database. We used a conservative criterion in determining gender of brand names. In particular, we took majority vote from prediction of the three classifiers for any given brand name as opposed to relying on prediction from a single classifier. [4]

Although it is difficult to fully evaluate the accuracy of our classifiers on brand name gender, we identified "true" gender of a small set of brand names where etymology can be found from the Oxford English Dictionary, summarized in Table 3. We found that our procedure of gender classification yielded an accuracy of 83.3% on this small set, which is consistent with the accuracies we obtained with people's names. We refrained from evaluating our models against human judgments on brand name gender, because people's conceptions might be biased or primed given the products associated with brands (e.g., cosmetic brand names might be perceived as female). Figure 3 visualizes the brand names in the feature space we considered, with predicted gender and annotated example brands.

Table 3: Brand names with gender identified from the OED and model-predicted gender (Male vs. Female).

Brand	Etymology	True	Pred.
Amazon	Female warrior (1398)	F	M
Titan	Helio's father (1413)	M	M
Pandora	Woman (1581)	F	F
Hermes	Zeus' son (1605)	M	M
Nike	Goddess (1846)	F	F
Lincoln	US president (-)	M	M

Brand name gender distribution. We analyzed gender distributions across and within the 17 product categories. Across categories, we observed a strong asymmetry in frequency between male and female brand names. The scatter-plot in Figure 3 illustrates that male and female brand names separate in sound-attribute space. tSNE (Maaten and Hinton, 2008) with 17 feature attributes was used to generate this plot, which shows that male and female names separate along the first dimension, but not in the second dimension. This suggests that these names share commonalities, but they are also different.

We observed that the gross number of male brand names is significantly greater than the number of female brand names (binomial $p < 0.0001$). Several factors could contribute to this bias. For example, many present-day brands originated long before gender equality was valued, and the brand names that emerged from these male-dominated eras tend to skew masculine. It should also be noted that many brands are named after company founders (e.g., *Ford*), and surnames may tend to be more masculine than first names.

A more fine-grained analysis in the individual product categories revealed nuanced patterns in gender distribution (Figure 3). Although the male bias we observed at the broad level applies to 14 of 17 categories (binomial $p < 0.0001$) including some intuitive ones such as "power tools," "military," and "baseball equipment," this bias is surprisingly weak or non-existent in "car / trucks" and "men haircare" where we expect more male as opposed to female consumers. Second, brands where

---

[4] We also considered a more stringent criterion by analyzing only the subset of brand names where all three classifiers agreed on gender prediction ($N = 702$ out of 1021 names), and our results are similar based on that subset of brand names.

we expect more female consumers such as "jewelry," "cosmetics," and "womens underwear" have a relatively high proportion of male-oriented names. These patterns suggest that although male brand names might be an overall preferred convention, gender-polarized brands might have adopted a reversal strategy in naming with the opposite gender.

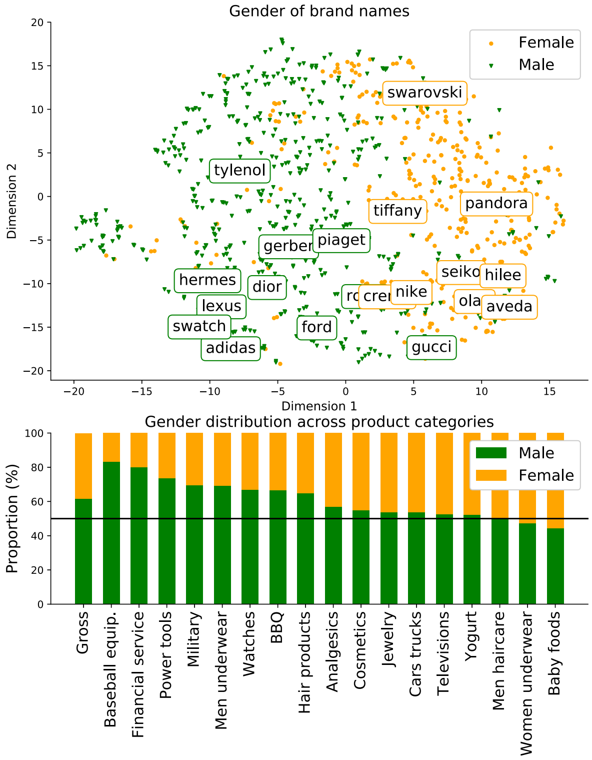

Figure 3: Predicted gender of 1021 brand names given 16 features and mapped a 2D space via tSNE. Bottom panel shows gender proportions in gross and 17 individual product categories.

6 Discussion

While the notion of brand name gender is an intriguing concept in theory, its practical measurement has proved elusive. This paper has shown that NLP-methods can be used fruitfully to get a handle on this problem. Our results suggest that brand names are more male-oriented than female-oriented overall. However, under this broad result, there are several interesting nuances. First, the overall preference for male names applies not only in categories that are primarily male-oriented, but also in some categories that are primarily female-oriented. Second, there is considerable within-category variation in brand name gender. Understanding these nuances is an important topic for future research.

References

Herbert Barry and Aylene S Harper. 1995. Increased choice of female phonetic attributes in first names. *Sex Roles*, 32(11-12):809–819.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: Analyzing text with the natural language toolkit*. O'Reilly Media.

Inge K Broverman, Susan Raymond Vogel, Donald M Broverman, Frank E Clarkson, and Paul S Rosenkrantz. 1972. Sex-role stereotypes: A current appraisal. *Journal of Social issues*, 28(2):59–78.

Carlyle Folkins and Peter B Lenrow. 1966. An investigation of the expressive values of graphemes. *The Psychological Record*, 16(2):193–200.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644.

Amélie Guèvremont and Bianca Grohmann. 2014. Can good news be bad? the role of brand communication strategy and brand commitment in the announcement of product improvements. *Journal of Marketing Communications*, 20(5):352–365.

Richard R Klink. 2000. Creating brand names with meaning: The use of sound symbolism. *Marketing Letters*, 11(1):5–20.

Stanley Lieberson and Eleanor O Bell. 1992. Children's first names: An empirical study of social taste. *American Journal of Sociology*, 98(3):511–554.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Anne Saxon Slater and Saul Feinman. 1985. Gender and the phonology of north american first names. *Sex Roles*, 13(7-8):429–440.

Janet T Spence and Robert L Helmreich. 1979. *Masculinity and femininity: Their psychological dimensions, correlates, and antecedents*. University of Texas Press.

Okito Yamashita, Masa-aki Sato, Taku Yoshioka, Frank Tong, and Yukiyasu Kamitani. 2008. Sparse estimation automatically selects voxels relevant for the decoding of fmri activity patterns. *NeuroImage*, 42(4):1414–1429.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Improving Large-Scale Fact-Checking using Decomposable Attention Models and Lexical Tagging

Nayeon Lee*, **Chien-Sheng Wu***, **Pascale Fung**
Human Language Technology Center (HLTC)
Center for Artificial Intelligence Research (CAiRE)
Hong Kong University of Science and Technology
`[nyleeaa,cwuak].connect.ust.hk, pascale@ece.ust.hk`

Abstract

Fact-checking of textual sources needs to effectively extract relevant information from large knowledge bases. In this paper, we extend an existing pipeline approach to better tackle this problem. We propose a neural ranker using a decomposable attention model that dynamically selects sentences to achieve promising improvement in evidence retrieval F1 by 38.80%, with (×65) speedup compared to a TF-IDF method. Moreover, we incorporate lexical tagging methods into our pipeline framework to simplify the tasks and render the model more generalizable. As a result, our framework achieves promising performance on a large-scale fact extraction and verification dataset with speedup.

1 Introduction

With the rapid growth of available textual information, automatic extraction and verification, also known as fact-checking, has become important in order to identify relevant and factual information from the ever-growing information pool. The FakeNews Challenge (Pomerleau and Rao) addresses fact-checking as a simple stance detection problem, where the article is verified by checking the stance agreement between an article's title and content. Similar to the FakeNews, (Rashkin et al., 2017; Vlachos and Riedel, 2014) focused on political statements from Politifact.com to verify the degree of truthfulness. However, they assume that the gold standard documents containing the evidence are already known, which overly simplifies the task.

Question Answering (QA) is similar to fact-checking in the sense that a question and its answers can be considered as a claim and evidence respectively, but the answers may come from a large-scale database. Several approaches (Chen et al., 2017a; Ryu et al., 2014; Ahn et al., 2004) proposed QA system utilizing resources such as Wikipedia, which is more comprehensive and incorporates wider world knowledge. However, the main focus is to identify only the "correct" answers that support a given question. Since the ability to refute is as important as to support, it does not fully address the verification problem of fact-checking.

Claim	Finding Dory was written by anyone but an American.
Evidence	**Finding_Dory**: Directed by Andrew Stanton with co-direction by Angus MacLane, the screenplay was *written by Stanton and Victoria Strouse* **Andrew_Stanton**: Andrew Stanton -LRB- born December 3, 1965 -RRB- is an *American film director* , screenwriter, producer and voice actor based at Pixar.
Label	REFUTE

Table 1: Example of verified claim with evidence from multiple Wikipedia pages

Recently, Thorne et al. (2018) proposed a public dataset to explore the complete process of the large-scale fact-checking. It is designed not only to verify claims but also to extract sets of related evidence. Nevertheless, the pipeline solution proposed in that paper suffers from following problems: 1) The overall performance (30.88% accuracy) still needs further improvement to be applicable to the evidence selection and classification, which also highlights the challenging nature of this task. 2) The evidence retrieval using Term Frequency-Inverse Document Frequency (TF-IDF) is time-consuming since the TF-IDF between a claim and set of candidate evidence cannot be computed in advance.

In this paper, we extend the original pipeline solution to achieve faster and better fact-checking results. Our main contributions are: 1) Propose a neural ranker using decomposable attention (DA) model for evidence selection to speed up (×65) and outperform related works. 2) Incorporate several lexical tag information to effectively simplify

---

* These two authors contributed equally.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1133–1138
Brussels, Belgium, October 31 - November 4, 2018.

k	DR_{tfidf}	DR_{rerank}
1	0.3145	0.6099
2	0.4321	0.7292
5	0.5895	0.8052
10	0.6916	0.8322
25	0.7882	0.8494
100	0.8886	0.8886

Table 2: Oracle document retrieval macro-recall in the test set (SUPPORT/REFUTE).

the problem and generalize the models. 3) Improve the overall fact extraction F1 by 38.80% and verification accuracy by 2.10% to achieve the state-of-the-art performance on the dataset.

2 Methodology

Our pipeline framework[1] has three main modules: document retrieval (DR), evidence selection (ES), and textual entailment recognition (TER). The goal is to verify a given claim with a set of evidence from Wikipedia (Table 1). The verification labels are support, refute and not enough information (NEI) to verify.

2.1 Lexical Tagging

In our framework, two lexical tags (i.e. part-of-speech (POS) and named entity recognition (NER)) are used to enhance the performance. We compute the tags for claims in advance using the Stanford CoreNLP (Manning et al., 2014) library. Using this information is helpful in the following ways: 1) it helps keyword extraction for each claim. 2) it reduces the out-of-vocabulary (OOV) problems related to name or organization entities, for better generalization. For example, a claim like "Michael Jackson and Justin Timberlake are friends," is replaced as "PERSON-1 and PERSON-2 are friends". In this way, we encourage our model to learn verification without dealing with the real entity values but the delexicalized indexed tokens.

2.2 Document Retrieval (DR)

For document retrieval, we extend the method of DrQA (Chen et al., 2017a), which calculates cosine similarity between query and document, using binned unigram and bigram TF-IDF features. We refer to this method as DR_{tfidf}.

Instead of directly selecting top k document using TF-IDF as in DR_{tfidf}, our document retriever DR_{rerank} use TD-IDF to reduce the search space from 5.4M to 100 documents. Re-ranking is then applied to select the top k documents. For re-rank, we defined a score function f_{rank} that ranks the relevance of the document by considering both the title and the content as follows:

$$r_{claim} = \frac{\text{POS}_{match}}{\text{POS}_{claim}}, r_{title} = \frac{\text{POS}_{match}}{\text{POS}_{title}},$$
$$f_{rank} = r_{claim} \times r_{title} \times tf\text{-}idf$$

To capture the relevance from the title, all the POS tags with high discriminating power (NN, NNS, NNP, NNPS, JJ, CD) of a claim are chosen as keywords. POS_{claim} and POS_{title} are the counts of such POS tags inside the claim and title respectively. POS_{match} is the count of common POS keywords in the claim and the title; r_{claim} is a ratio between POS_{match} and POS_{claim} to reward the documents with higher keyword hits; r_{title} is the ratio between POS_{match} and POS_{title} to penalize those documents with more candidate keywords as it is more likely to have keyword hits with more candidates. We incorporate the TF-IDF score ($tf\text{-}idf$) to ensure that the content information is not neglected. Our experiments show that our re-rank strategy increases the document recall compared to the single-step approach (Table 2). To decide on the optimal value for hyper-parameter k, full-pipeline performance was compared to evaluate the effect of k on final verification accuracy.

2.3 Evidence Selection (ES)

In this module, l sentences are extracted as possible evidence for the claim. Instead of selecting the sentences by recomputing sentence-level TF-IDF features between claim and document text as in Thorne et al. (2018), we propose a neural ranker using decomposable attention (DA) model (Parikh et al., 2016) to perform evidence selection. DA model does not require the input text to be parsed syntactically, nor is an ensemble, and it is faster without any recurrent structure. In general, using neural methods is better for the following reasons: 1) The TF-IDF may have limited ability to capture semantics compared to word representation learning 2) Faster inference time compared to TF-IDF methods that need real-time reconstruction.

The neural ranker DA_{rank} is trained using a fake task, which is to classify whether a given sentence is an evidence of a given claim or not. The

---

[1] https://github.com/HLTCHKUST/fact-checking

		TF-IDF	DA_{rank}			DA_{rank}+NER		
			1:1	1:4	1:9	1:1	1:4	1:9
l	2	0.847	0.170	0.889	0.889	0.109	0.889	**0.893**
	5	0.918	0.451	0.966	**0.968**	0.345	0.962	**0.968**
Time		3.57s	0.055s					

Table 3: Oracle evidence selection macro-recall in the test set using gold documents (SUPPORT/REFUTE).

output of DA_{rank} is considered as the evidence probability. The training samples are unbalanced since there are more unrelated sentences than evidence sentences. Note that the classifier's accuracy on the fake task is not crucial because the choice of evidence is based on the relative score of relevance compared to other candidates. Therefore, it is not necessary to balance positive and negative samples using up/down-sampling, to the contrary, making it unbalanced actually improved the performance (Table 3).

Unlike the k value which is fixed, the l value is selected dynamically based on the evidence score of DA_{rank}'. It is used as a confidence measure of the given sentence being an evidence. Evidence with the score below fixed threshold value th is eliminated. Hence, each claim will have a different number of l evidence. To decide on th, we also carry out the full-pipeline evaluation. We propose the dynamic selection of l because we hypothesize that any wrong evidence, or noise, from early module could harm the subsequent RTE module.

2.4 Recognizing Textual Entailment (RTE)

Given a claim and l possible evidence, a DA_{rte} classifier is trained to recognize the textual entailment to be support, refute or not enough information to verify (NEI). Same as Thorne et al. (2018), we use the decomposable attention (DA) between the claim and the evidence for RTE. DA model decomposes the RTE problem into subproblems, which can be considered as bi-direction word-level attention features. Note that the DA model is utilized over other models such as as Chen et al. (2017b); Glockner et al. (2018), because it is a simple but effective model.

Our DA_{rte} model must correctly decide whether a claim is NEI, when the evidence retrieved is irrelevant and insufficient. However, NEI claims have no annotated evidence, thus cannot be used to train RTE. To overcome this issue, same as (Thorne et al., 2018), the most probable NEI evidence are simulated by sampling sentences from the nearest page to the claim using the document retrieval module.

	MLP	DA_{rte}	DA_{rte}+NER
Accuracy (%)	63.2	78.4	**79.9**

Table 4: Oracle RTE classification accuracy in the test set using gold evidence.

3 Experimental setup

Dataset: FEVER dataset (Thorne et al., 2018) is a relatively large-scale dataset compared to other previous fact extraction and verification works, with around 5.4M Wikipedia documents and 185k samples. The claims are generated by altering sentences extracted from Wikipedia, with human-annotated evidence sentences and verification labels (e.g. Table 1). The training/validation/test sets of these three datasets are split in advance by the providers. Note that the test-set was equally split into 3 classes: Supported (3333), Refuted (3333), NEI (3333).

Training: We trained our models end-to-end using Adagrad optimizer (Duchi et al., 2011). The embedding size is set to 200 and initialized with GloVe (Pennington et al., 2014). The dropout rate is set to 0.2. In all the datasets, we tuned the hyper-parameters with grid-search over the validation set.

Evaluation: For each module, we independently measure oracle performance, where we assume gold standard documents and set of evidence are provided (oracle evaluation). For the final full-pipeline, we compare to and follow the metric defined in Thorne et al. (2018). NoScoreEv is a simple classification accuracy that only considers the correctness of the verification label. On the other hand, ScoreEv is a stricter measure that also considers the correctness of the retrieved evidence. Hence, it is a more meaningful measure because it considers the classification to be correct only if appropriate evidence is provided to justify the classification.

4 Experimental Results

4.1 Oracle Performance

Document Retrieval: As shown in Table 2, our count-based re-rank strategy outperforms the TF-IDF method.

Take $k = 1$ as an example, we achieve 60.99% recall using only one document, which is $\sim$30% higher than TF-IDF approach. Given that the recall upper-bound of re-rank is 0.8886 (k=100), our method manages to retrieve near the limit by just retrieving a few documents. Note that there is

k	DR results			RTE results				
	Macro Recall	Macro Precision	F1	Accuracy		Evidence		
				ScoreEv	NoScoreEv	Precision	Recall	F1
2	0.729	**0.383**	**0.498**	**0.412**	**0.541**	**0.169**	0.655	**0.269**
3	0.768	0.270	0.396	0.407	0.534	0.166	0.672	0.267
4	0.791	0.209	0.328	0.407	0.535	0.164	**0.676**	0.264
5	**0.805**	0.170	0.279	0.397	0.529	0.161	0.675	0.260

Table 5: Effect of de-noising DR modules on RTE score

th	ES results			RTE results				
	Macro Recall	Macro Precision	F1	Accuracy		Evidence		
				ScoreEv	NoScoreEv	Precision	Recalll	F1
0.2	**0.653**	0.275	0.353	0.405	0.540	0.337	**0.629**	0.439
0.4	0.607	0.349	**0.406**	0.418	**0.542**	0.481	0.586	0.528
0.6	0.535	**0.368**	**0.406**	**0.424**	0.525	0.618	0.517	**0.563**
0.8	0.413	0.330	0.348	0.416	0.484	**0.772**	0.400	0.527

Table 6: Effect of de-noising ES modules on RTE score

a trade-off between the document recall and the noise ratio (i.e. low precision). As shown in Table 5, $k = 2$ with a low recall but high precision and F1 has the highest accuracy. This means DR that can effectively leverage this trade-off (therefore, high F1) performs the best. Therefore, we select $k = 2$ for our full-pipeline.

Evidence Selection: In Table 3, the trained neural ranker achieves a promising recall of 96.8%. In the case of $l = 5$, our neural ranker can perform 5% better than the TF-IDF method. Here, we use fixed $l = 5$ results for fair comparison with TF-IDF method. The ratios in Table 3 are the ratio of negative samples we tried to train the fake task. For example, 1:4 means that four negative sentences are sampled for each true evidence sentence. The results further give supports to our assumption that using unbalanced up-sampling actually help train our neural ranker. Along with performance gain, we also achieve a drastic gain in inference time by around 65 times from 3.57 seconds to 0.055 seconds for each sample.

The full-pipeline results for different values of th is shown in Table 6. Similar to document retrieval, having a high F1 score is the most important factor in assuring high full-pipeline performance. This is because providing a succinct set of evidence makes the verification task easier for the RTE module. Therefore, we choose DA_{rank}+NER model with $th = 0.6$ for the full-pipeline.

Recognizing Textual Entailment: The oracle classification accuracy for RTE is shown in Table 4. The MLP is a simple multi-layer perceptron using TF and TF-IDF cosine similarity between the claim and evidence as features as shown in Thorne et al. (2018). The highest accuracy achieved is 79.9% using DA_{rte} with NER information, thus, we further evaluate the full-pipeline accuracy on this setting.

4.2 Full pipeline Performance

Combining each of our improved pipeline modules using $k = 2$, $th = 0.6$, the full pipeline results are shown in Table 7. Our proposed framework can achieve 42.43% in ScoreEv and 52.54% in NoScoreEv, which outperforms DR_{tfidf}+DA by 11.55% and 2.10%, respectively. The evidence retrieval F1 in our full framework is 56.3%, which is improved promisingly by 38.80%.

5 Related Work

Prior work (Vlachos and Riedel, 2014; Ciampaglia et al., 2015) have proposed fact-checking through entailment from knowledge bases. Some works have investigated fact verification using PolitiFact data (Wang, 2017; Rashkin et al., 2017) or FakeNews challenge (Pomerleau and Rao). Most closely related to our work, Thorne et al. (2018) addresses large-scale fact extraction and verification task using a pipeline approach. In addition, question answering (Dang et al., 2007; Chen et al., 2017a; Ryu et al., 2014; Ahn et al., 2004) and task-oriented dialog systems (Dhingra et al., 2017;

Model	Label Accuracy (%)		Label			Evidence F1
	ScoreEv	NoScoreEv	Precision	Recall	F1	
DR_{tfidf} + MLP *	21.80	38.75	0.500	0.387	0.310	0.175
DR_{tfidf} + DA *	30.88	50.44	0.530	0.520	0.517	
Proposed	**42.43**	**52.54**	**0.533**	**0.527**	**0.523**	**0.563**

Table 7: Full-pipeline evaluation on the test set using $k = 2$ and $th = 0.6$. The first and the second one (with *) are the baselines from Thorne et al. (2018).

Madotto et al., 2018) also have similar aspects to these works, although aiming at a different goal.

Other fields that are related to the particular individual modules of our system are the following: Document and evidence retrieval for identifying text segments and documents to support a given claim (Salton and Buckley, 1987; Le and Mikolov, 2014; Cartright et al., 2011; Bellot et al., 2013; Rinott et al., 2015). Recognizing textual entailment that aims to determine whether a hypothesis h can justifiably be inferred from a premise (Dang et al., 2007; Bowman et al., 2015; Parikh et al., 2016; Chen et al., 2017b; Glockner et al., 2018). In some of these work (Rinott et al., 2015; Rashkin et al., 2017), the lexical and linguistic features are leveraged to further improve the performance.

6 Conclusion

In this paper, we extend the pipeline framework for fact-checking and propose a neural ranker for evidence selection. Our experiments show that the usage of lexical tagging is helpful in simplifying the task and improving the generalization ability. Moreover, reducing noise in the input of RTE module, by de-noising the DR and SR modules, appears to be crucial for improving the overall performance. As a result, our ranker outperforms the TF-IDF method by 38.8% in evidence retrieval F1, with 65 times faster inference speed, achieving a promising performance in a large-scale fact extraction and verification dataset.

References

David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Müller, Maarten de Rijke, Stefan Schlobach, M Voorhees, and L Buckland. 2004. Using wikipedia at the trec qa track. In *TREC*. Citeseer.

Patrice Bellot, Antoine Doucet, Shlomo Geva, Sairam Gurajada, Jaap Kamps, Gabriella Kazai, Marijn Koolen, Arunav Mishra, Véronique Moriceau, Josiane Mothe, et al. 2013. Overview of inex 2013. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 269–281. Springer.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Marc-Allen Cartright, Henry A Feild, and James Allan. 2011. Evidence finding using a collection of books. In *Proceedings of the 4th ACM workshop on Online books, complementary social media and crowdsourcing*, pages 11–18. ACM.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. tx. *PloS one*, 10(6):e0128193.

Hoa Trang Dang, Diane Kelly, and Jimmy J Lin. 2007. Overview of the trec 2007 question answering track. In *Trec*, volume 7, page 63.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceddings of ACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. *arXiv preprint arXiv:1805.02266*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1468–1478. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Dean Pomerleau and Delip Rao. Fake news challenge.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence-an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450.

Pum-Mo Ryu, Myung-Gil Jang, and Hyun-Ki Kim. 2014. Open domain question answering using wikipedia-based knowledge model. *Information Processing & Management*, 50(5):683–692.

Gerard Salton and Chris Buckley. 1987. Term weighting approaches in automatic text retrieval. Technical report, Cornell University.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22.

William Yang Wang. 2017. ”liar, liar pants on fire”: A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426, Vancouver, Canada. Association for Computational Linguistics.

Harnessing Popularity in Social Media for Extractive Summarization of Online Conversations

Ryuji Kano[†] **Yasuhide Miura**[†] **Motoki Taniguchi**[†]
Yan-Ying Chen[‡] **Francine Chen**[‡] **Tomoko Ohkuma**[†]
[†]Fuji Xerox Co., Ltd.
{kano.ryuji, yasuhide.miura, motoki.taniguchi, ohkuma.tomoko}
@fujixerox.co.jp
[‡]FX Palo Alto Laboratory
{yanying, chen}@fxpal.com

Abstract

We leverage a popularity measure in social media as a distant label for extractive summarization of online conversations. In social media, users can vote, share, or bookmark a post they prefer. The number of these actions is regarded as a measure of popularity. However, popularity is not determined solely by content of a post, e.g., a text or an image it contains, but is highly based on its contexts, e.g., timing, and authority. We propose *Disjunctive model* that computes the contribution of content and context separately. For evaluation, we build a dataset where the informativeness of comments is annotated. We evaluate the results with ranking metrics, and show that our model outperforms the baseline models which directly use popularity as a measure of informativeness.

1 Introduction

Online conversations are increasingly significant for communication, e.g., Slack[1] for work communication and Reddit[2] for general discussion. To organize overwhelming information from these conversations, researchers have been working on summarizing online conversations (Bhatia et al., 2014; Carenini et al., 2007; Mehdad et al., 2013, 2014; Oya et al., 2014). State-of-the-art models in both abstractive (Rush et al., 2015) and extractive (Cheng and Lapata, 2016) summarization tasks are based on neural networks, but these models require large amounts of training data. In previous research, these data were created automatically by retrieving headlines and highlights of news articles edited by news editors. However, these methodologies cannot be applied to the summarization of online conversations because of a lack of summary annotations.

Distant labels have been used to train models, thereby reducing the need for manual labeling; some of these labels were also applied to the summarization task. Categories of news articles (Isonuma et al., 2017) and ratings of online reviews (Xiong and Litman, 2014) were used as distant labels in extractive summarization. However, these have been used as supplementary labels to enhance conventional summarization models, whereas we present labels which a model can solely be trained with.

We leverage a measure of popularity as a distant label. In social media, users can vote, share, or bookmark a post they prefer, and the number of these actions are regarded as indications of popularity. We assume that measures of popularity reflects the *informativeness*, the index required for a summary (Erkan and Radev, 2004), and validate whether popularity can be used as a distant label for extractive summarization. However, popularity is not solely determined by content, e.g., a text or an image, but is highly affected by contexts, e.g., timing, and authority (Cheng et al., 2017; Burghardt et al., 2017; Suh et al., 2010; Hessel et al., 2017; Jaech et al., 2015). Therefore, to utilize popularity as an indicator of informativeness, we need to exclude the effect of context.

To exclude the effect, we propose *Disjunctive model.* This model computes two scalar values; one from a content feature and the other from a context feature. These two values are then multiplied to predict the popularity. The scalar values can be interpreted as the contribution of content and context to the prediction. We assume that the contribution of content to indicate informativeness.

For evaluation, we build a test dataset where comments are annotated for informativeness. We measure informativeness as an index indicative of the best sentences to extract as a summary. We

---

[1] https://slack.com
[2] https://www.reddit.com

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1139–1145
Brussels, Belgium, October 31 - November 4, 2018.

select Reddit as a data source, where the *karma score*, a measure of popularity in Reddit, is known to be affected by contexts. Our test task is to extract informative posts. Because informativeness of each post is annotated via crowdsourcing, the extracts can be ranked, but the appropriate number is unknown. Therefore, we employ ranking metrics in the evaluation. Our experiment only use karma scores for training to verify that they reflect informativeness. The results show that our model outperforms baseline models that directly adopt karma scores as an indicator of informativeness. Furthermore, our model focus on a local feature of a single post, whereas conventional centrality-based models (Erkan and Radev, 2004; Mihalcea and Tarau, 2004) use a global context of posts, and the complementary hybrid of the both models outperform both the centrality-based models and our models.

The contributions of this paper are three fold. 1) Propose a model that harnesses a popularity measure as a distant label for extractive summarization. 2) Create a dataset of online conversations in which the informativeness of contents are annotated to verify that popularity does not correlate with informativeness because of the effect of context. 3) Demonstrate that our model, when combined with a centrality-based model, outperforms baseline models in predicting the informativeness of posts.

2 Related Work

Previous research of summarizing online conversations can be categorized into graph-based methods (Mehdad et al., 2013, 2014; Shang et al., 2018), template-based methods (Oya et al., 2014), and methods which use dialogue acts as a feature (Bhatia et al., 2014; Carenini et al., 2007). In previous research, few or no training data was adopted because of a lack of labeled data. Our model harnesses a vast amount of data from social media.

Many researchers used user-contributed labels from social media as distant labels. Xiong (2014) used review scores on a movie-rating site for a summarization task. For a sentiment analysis on Twitter[3], Davidov(2010) used hashtags, and Guibon (2017) used emoji. In our study, we leverage a popularity measure for a summarization task.

Factor analysis quantifies the contribution of each feature to the result, using a linear model. For example, Suh (2010) analyzed factors contributing to popularity in Twitter. Our model assumes a linear relationship between context and content, and thus enables to utilize the contribution of content as an indicator of informativeness.

3 Data

In this study, we work with Reddit threads. A thread is a set of comments, and the first posted comment is called a submission. Comments can be made in response to submissions as well as comments under the submissions, resulting in a thread being tree-structured. Submissions and comments can be upvoted or downvoted by readers, and karma scores are computed as upvotes minus downvotes. Karma scores follow Zipf's law (Cheng et al., 2017). Therefore, we smooth the karma scores as follows:

$$f(k) = \begin{cases} \log(k+1) & (k \geq 0) \\ 0 & (k < 0) \end{cases}$$

where k represents the karma score.

Reddit is organized into subreddits by topic. Posts from the subreddits AskMen, AskWomen, and AskReddit with 420,598, 247,012, and 644,034 comments, respectively, are collected and split into training and validation sets with a 4:1 ratio. The validation sets are used for early-stopping. All comments were posted from June, 1, 2016 through June, 1, 2017.

Manual Annotation We crowdsource the annotation of comments in terms of informativeness to utilize them as test data. Annotators are asked to choose 3 informative comments from 10. We define 10 comments as a *subset*; each comment is a reply to a submission. For submissions with more than 10 replies, posts with the top 10 karma scores are selected. For each subreddit, 130 subsets were annotated, for a total of 1,300 comments. Because 10 annotators vote for 3 different comments each, the number of votes for a comment ranged from 0 to 10. These numbers we refer to as the *annotated score*. The comments in each subset are shuffled to invalidate the effect of the order in which annotators read the comments.

Liu and Liu (2008) observed that the best summary differs among annotators, especially when summarizing conversations, consequently resulting in low Kappa scores. In their study, the Kappa statistics for six different annotators varied from

---

[3] https://twitter.com

Table 1: Examples of posts with low karma scores and high annotated scores, and vice versa.

karma score	annotated score	post
0	8	Martin Shkreli was streaming League of legends and my brother messaged him to see if he could get an invite to the group. They have played a few times since.
1	10	Same thing goes for being bitten by a dog, instead of instinctively pulling away...force your arm/hand down their throat. Super effective.
1	10	1 Make sure you have solid internet. 2 Find work from home, they actually exist book keeping, software testing, data entry, etc. 3 Work from home while earning a modest wage you wont get rich on those jobs, but it will certainly pay the bills.
360	0	Ill try this the next time my toddler bites me.
253	0	Im an English major who wants to go into marketing. Wat do
228	0	Sadly people buy the first thing that the see and this is what is in season.

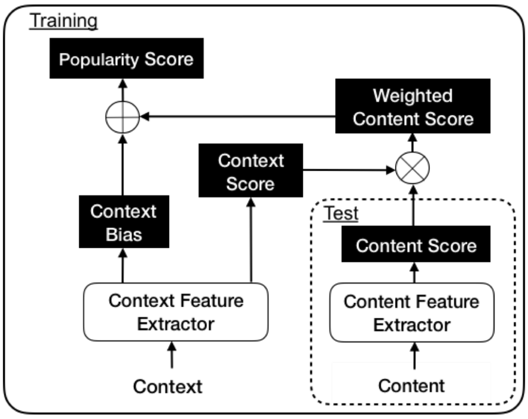

Figure 1: Description of the Disjunctive model. $\oplus$ and $\otimes$ represent addition and multiplication, respectively. The components of each model we use for training and testing are shown. Black and white squares represent scalars and functions, respectively.

0.11 to 0.35. In our study, the Fleiss Kappa coefficients (Fleiss and Cohen, 1973) of the annotated data are 0.252 for AskMen, 0.191 for AskWomen, and 0.213 for AskReddit.

The correlation coefficients of the karma scores versus the annotated scores are low: 0.063 for AskMen, 0.081 for AskReddit, and 0.107 for AskWomen. Table 1 shows some examples of posts with low karma scores and high annotated scores, and vice versa. It shows that there are informative posts with low karma scores, and non-informative posts with high karma scores. This implies that it is necessary to exclude the effect of context to leverage karma scores as distant labels for summarization.

4 Proposed Model

To exclude the effect of context from the popularity, we propose *Disjunctive model* (Figure 1). This model computes two scalar values, a *content score* and a *context score* from a content feature and a context feature, respectively by multiplying parameter vectors. The model is trained to predict popularity by multiplying the two scores and adding a *context bias*, which is also computed from a context feature. After training, the two scores represent the contribution of the content and context to popularity. We assume that the content score indicates informativeness. While training, the *popularity score* is used to predict the popularity, which is represented by the karma scores in our study. During evaluation, the content score is used for prediction of informativeness. The context score is constrained to be positive; otherwise, it can be either positive or negative, making it difficult to assume that a content score represents informativeness.

4.1 Context Feature Extractor

We use a multi-layer perceptron (MLP) to extract the features of the context of comments. Our study discusses six attributes of context: the karma score of a submission, the karma score of the previous comment, the depth in a thread, the relative time since the previous comment, the rank of the relative time among all replies to a previous comment, and the number of replies to a previous comment. The number of layers is set to 3, and the dimensions of each layer are 64.

4.2 Content Feature Extractor

We use two content extractors: long short-term memory (LSTM) as a basic language model, and a factored neural network (FNN) (Cheng et al., 2017) as a model that achieved state-of-the-art results in karma score prediction tasks. FNN, which is a language model, sequentially predicts the next words in a comment and its reply using an attention mechanism. As in the previous research, we pretrain this model using the same data used in the training, and fine-tune its parameters on the karma

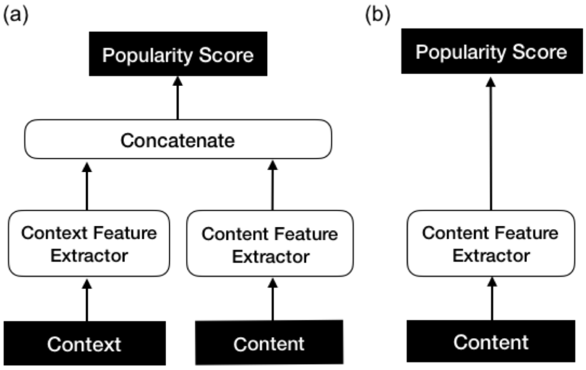

Figure 2: Description of baseline models. (a) Concat model. (b) Text model.

score prediction task. A single-layer LSTM and FNN are used and the last hidden layers are used as the content feature. The dimensions of the hidden layers are set to 64, and the dimensions of the word embedding are set to 256.

5 Experiments

We train the summarization model using karma scores as distant labels and evaluate the prediction of informativeness with the annotated dataset. As explained in Sec 3, the informativeness of each post is annotated via crowdsourcing and it is difficult to determine the appropriate number of posts needed to create a summary. Therefore, we employ ranking metrics for evaluation. In each subset, where subsets were defined in Sec 3, we rank each comment from 1 to 10 in terms of predicted scores and annotated scores. Ranks of tied scores are set randomly. To avoid randomness from affecting the result, we evaluate 100 times and compute an average as a result. We use three metrics: Spearman's Rho ($S\rho$), precision@3 (prec3), and Mean Reciprocal Rank (MRR) (Mcfee and Lanckriet, 2010).

5.1 Experimental Setting

Experiments are conducted by using mean-squared error as the loss function and Adam as the optimizer (Kingma and Ba, 2014). We replace words that appear fewer than five times with <unk>. There are 63,093 unique terms for AskMen, 53,589 for AskWomen, and 80,426 for AskReddit. The maximum length of each comment is clipped to 50. The mini-batch size is 64.

5.2 Baseline Model

We experiment with four baseline models. Two are supervised models as shown in Figure 2: the Concat model concatenates content and context features, and the Text model uses only content features. The other two are centrality-based unsupervised models: LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004). The unsupervised models only use content features. Disjunctive model use different scores for prediction in the training and in the test, as shown in Figure 1; however, Concat and Text models use a popularity score, the predicted value of a karma score, both in the training and in the test. This is because there is no substitute for the content score in these models.

5.3 Hybrid Model

The supervised models in our study, including the Disjunctive models, compute features from a single post only. To also harness the global information encoded in all posts in a subset, we build a Hybrid model which multiplies the scores from the Disjunctive model and the TextRank.

6 Results

The results of the experiments described in Sec 5 are shown in Table 2. The suffixes Disjunctive, Concat, and Text denote the supervised models described in Sec 4 and Sec 5. The prefixes LSTM and FNN indicate the models we use for content feature extractors. Among the supervised models, our Disjunctive models outperform both LSTM and FNN-based baseline models. In contrast, the results of the Concat models are poor. Unsupervised models which use the global feature of posts in a subset perform well. The FNNDisjunctive model combined with TextRank outperforms both the supervised models and the unsupervised models. To confirm that multiplication performs better in our task, we also experimented with Additive models, which simply add the context score and the content score instead of multiplying. Although better performing than the Text model, the performance was not as good as the Disjunctive model (Not shown in Table).

7 Discussion

Here we discuss the comparison of the results shown in Table 2, and how our Disjunctive model separate the effect of content and context.

Text Model vs Concat Model vs Disjunctive Model The Text model performs worse than our model. A possible reason is that karma scores can

Table 2: Result of ranking annotated scores. The best results among the supervised models are underlined, and the best results among all the models are bolded. The names of our models are also bolded.

Content Type	Model	AskMen			AskReddit			AskWomen		
		Sρ	MRR	prec3	Sρ	MRR	prec3	Sρ	MRR	prec3
Super. Local	LSTMConcat	0.021	0.279	0.297	0.028	0.294	0.323	0.146	0.315	0.367
	FNNConcat	0.032	0.276	0.298	0.068	0.325	0.323	0.129	0.309	0.346
	LSTMText	0.057	0.313	0.337	0.08	0.335	0.342	0.230	0.373	0.413
	FNNText	0.045	0.308	0.318	0.052	0.326	0.320	0.169	0.360	0.393
	LSTMDisjunctive	0.046	0.325	0.327	0.137	0.362	0.384	0.302	0.409	0.452
	FNNDisjunctive	0.164	0.375	0.378	0.196	0.411	0.414	0.259	0.402	0.435
Unsup. Global	TextRank	0.300	0.405	0.417	0.301	0.425	0.435	0.291	0.416	0.422
	LexRank	0.043	0.321	0.352	0.137	0.305	0.384	0.122	0.347	0.383
Hybrid	**TextRank+LSTMDisjuntive**	0.095	0.373	0.358	0.199	0.396	0.422	0.340	**0.438**	**0.468**
	TextRank+FNNDisjunctive	**0.319**	**0.436**	**0.452**	**0.336**	**0.452**	**0.448**	**0.345**	0.437	0.460

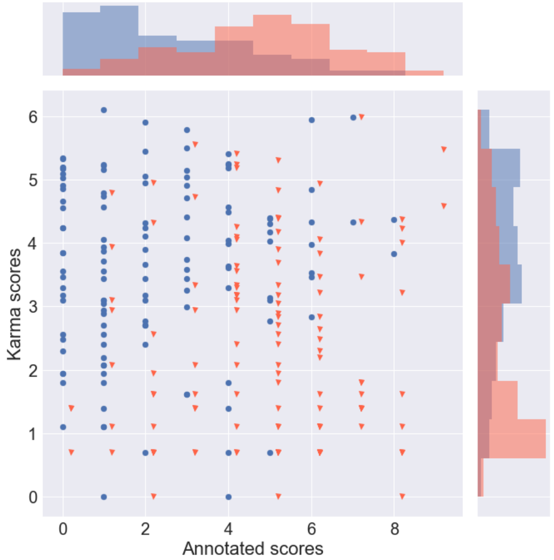

Figure 3: Karma scores and annotated scores of posts extracted by the LSTMConcat model (blue dots on the left) and the LSTMDisjunctive model (red dots shifted to the right for visibility) from the AskWomen dataset. Histograms of the karma scores and the annotated scores are also shown above and on the right.

be different even with similar text content because of different context, and this confuses the models that only use content features. Our model, by contrast, can avoid this problem because it considers the effect of context.

The Text model outperform the Concat model because context is a strong factor in predicting karma scores. If a model can use both content and context (as the Concat model does), it might overfit to context and ignore content. This does not happen in our model because it does not use context in test.

Hybrid Model vs Disjunctive Model The good performance results of the TextRank model indicate that global features of the posts are informative for ranking. While the supervised models just focus on one post at a time in each subset, the unsupervised models look at all the posts together in a subset. The hybrid model of the TextRank and the FNNDisjunctive models takes advantage of the complementary focus of the individual models, and outperforms both the supervised and the unsupervised models.

Separation of Content and Context The visualization in Figure 3 shows that our model can predict informativeness whereas the Concat model cannot. From each subset explained in Sec 3, we extract the post with the highest predicted score by the LSTMConcat and the LSTMDisjunctive. The karma scores and annotated scores of the extracted posts are plotted as blue and red dots, respectively. The karma scores are smoothed by the equation described in Sec 3. There are 130 subsets, for a total of 260 dots plotted. The Concat model extracts posts with low annotated scores but high karma scores, whereas the Disjunctive model extracts posts with high annotated scores regardless of the karma scores.

8 Conclusion

We proposed Disjunctive model that harnesses popularity as distant labels for use in extractive summarization. Our model was shown capable of separating the effects of content and context in a popularity measure and predicting the informativeness of content. To evaluate this, we built a Reddit dataset where informative comments were annotated. Our model, combined with a centrality-based model, outperformed the baseline models on the task of ranking posts to correspond to the rank of the annotated scores in three ranking metrics. Our models currently use only a single post as a feature. In the future, we will develop a model which uses a series of posts as a feature.

References

Sumit Bhatia, Prakhar Biyani, and Prasenjit Mitra. 2014. Summarizing online forum discussions – can dialog acts of individual messages help? In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2127–2131, Doha, Qatar. Association for Computational Linguistics.

Keith Burghardt, Emanuel F. Alsina, Michelle Girvan, William Rand, and Kristina Lerman. 2017. The myopia of crowds: Cognitive load and collective evaluation of answers on stack exchange. *PLOS ONE*, 12(3):e0173610+.

Giuseppe Carenini, Raymond T. Ng, and Xiaodong Zhou. 2007. Summarizing email conversations with clue words. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 91–100, New York, NY, USA. ACM.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2017. A factored neural network model for characterizing online discussions in vector space. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2296–2306, Copenhagen, Denmark. Association for Computational Linguistics.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494. Association for Computational Linguistics.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 241–249, Stroudsburg, PA, USA. Association for Computational Linguistics.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

Joseph L. Fleiss and Jacob Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and Psychological Measurement*, 33(3):613–619.

Gael Guibon, Magalie Ochs, and Parice Bellot. 2017. From emojis to sentiment analysis. WACAI 2016, hal-01529708.

Jack Hessel, Lillian Lee, and David Mimno. 2017. Cats and captions vs. creators and the clock: Comparing multimodal content to context in predicting relative popularity. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 927–936, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. 2017. Extractive summarization using multi-task learning with document classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2101–2110. Association for Computational Linguistics.

Aaron Jaech, Victoria Zayats, Hao Fang, Mari Ostendorf, and Hannaneh Hajishirzi. 2015. Talking to the crowd: What do people react to in online discussions? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Feifan Liu and Yang Liu. 2008. Correlation between rouge and human evaluation of extractive meeting summaries. In *Proceedings of ACL-08: HLT, Short Papers*, pages 201–204. Association for Computational Linguistics.

Brian Mcfee and Gert Lanckriet. 2010. Metric learning to rank. In *In Proceedings of the 27th annual International Conference on Machine Learning (ICML*.

Yashar Mehdad, Giuseppe Carenini, and Raymond T. Ng. 2014. Abstractive summarization of spoken and written conversations based on phrasal queries. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1220–1230, Baltimore, Maryland. Association for Computational Linguistics.

Yashar Mehdad, Giuseppe Carenini, Frank Tompa, and Raymond T. NG. 2013. Abstractive meeting summarization with entailment and fusion. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 136–146. Association for Computational Linguistics.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*.

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. Proceedings of the 8th International Natural Language Generation Conference, pages 45-53.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Guokan Shang, Wensi Ding, Zekun Zhang, Antoine Tixier, Polykarpos Meladianos, Michalis Vazirgiannis, and Jean-Pierre Lorré. 2018. Unsupervised abstractive meeting summarization with multi-sentence compression and budgeted submodular maximization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 664–674. Association for Computational Linguistics.

Bongwon Suh, Lichan Hong, Peter Pirolli, and Ed H. Chi. 2010. Want to be retweeted? large scale analytics on factors impacting retweet in twitter network. In *Proceedings of the 2010 IEEE Second International Conference on Social Computing*, SOCIALCOM '10, pages 177–184, Washington, DC, USA. IEEE Computer Society.

Wenting Xiong and Diane Litman. 2014. Empirical analysis of exploiting review helpfulness for extractive summarization of online reviews. Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics.

Identifying Locus of Control in Social Media Language

Masoud Rouhizadeh[*1] **Kokil Jaidka**[*2] **Laura Smith**[3]
H. Andrew Schwartz[4] **Anneke Buffone**[3] **Lyle H. Ungar**[2]
[1]Institute for Clinical & Translational Research, Johns Hopkins University
[2]Department of Computer & Information Science, University of Pennsylvania
[3]Department of Psychology, University of Pennsylvania
[4]Department of Computer Science, Stony Brook University
`mrou@jhu.edu, jaidka@sas.upenn.edu`

Abstract

Individuals express their locus of control, or "control", in their language when they identify whether or not they are in control of their circumstances. Although control is a core concept underlying rhetorical style, it is not clear whether control is expressed by *how* or by *what* authors write. We explore the roles of syntax and semantics in expressing users' sense of control –i.e. being "controlled by" or "in control of" their circumstances– in a corpus of annotated Facebook posts. We present rich insights into these linguistic aspects and find that while the language signaling control is easy to identify, it is more challenging to label it is internally or externally controlled, with lexical features outperforming syntactic features at the task. Our findings could have important implications for studying self-expression in social media.

1 Introduction

Language is a form of action, and written occurrences constitute a performance of power and control (Fairclough, 2001). Research in natural language processing has long focused on distilling the constituents of writing that conveys authority (Danescu-Niculescu-Mizil et al., 2012), dominance (Bradley and Lang, 1999), dogmatism (Fast and Horvitz, 2016), expertise (Levy and Potts, 2015) and politeness (Danescu-Niculescu-Mizil et al., 2012). These studies have shown how authors' use of certain lexical and syntactic patterns achieve specific rhetorical effects. In this study, we contribute to the growing literature with an analysis of how individuals express ***control***, and compare the insights and predictive power obtained from both, lexical and syntactic features.

We operationalize the key aspects of locus of control described by existing psychology theories (Rotter, 1966) to identify an external locus of control when authors express that they feel controlled by other people or the environment. On the other hand, authors communicate an internal locus of control when they ascribe the control of their decisions and circumstances to themselves. This study poses the question: *do writers rely more on content than syntax to convey their control?* Following its psychological underpinnings, we expect that overall lexical choice, self-reference, and certain verb categories would be indicative features for modeling author's locus of control. Our study attempts to validate these assumptions by answering two research questions:

- To what extent can *lexical* and *syntactic* features predict *control relevance* and *internal and external* locus of control?
- Which verb categories are more associated with control relevance and internal vs. external locus of control?

We train a multi-stage predictive model to classify posts on Facebook as (a) control-relevant or not, and (b) internal vs. external control. We find that syntactic and semantic features work well to identify control relevance, with verb categories and pronoun use being the most predictive features. However, determining internal vs. external control is a more challenging task, where lexical features vastly outperform syntax-based features. The best performance is seen by a model that combines lexical and syntactic features along with self-reference ratio. We will be releasing our tools and models along with annotated and anonymized datasets for the benefit of the research community.

2 Background

Locus of control, or "control", reflects the extent to which people ascribe the cause or control of events in their lives to themselves or the external

---

[*] Masoud Rouhizadeh & Kokil Jaidka co-lead this work.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1146–1152
Brussels, Belgium, October 31 - November 4, 2018.

factors (Rotter, 1966). The language indicating internal control in a given situation signals intentionality (the author is describing an action that he/she intended) and awareness (the author is aware of the effect of the action). Internal control is often associated with causing a given event or doing something that is clearly a choice. The external control language, on the other hand, is characterized by lack of intention or awareness, or by concrete mention of being controlled by others. It is usually lined to describing an out-of-control event, or something that is not a choice

Organizations and governments are attempting to better understand issues of locus of control and self-efficacy, which are closely related with physical health and job performance (Marmot et al., 1991; Harter et al., 2003). The study by (Jaidka et al., 2018b) offered to explore the relationship between locus of control (measured through surveys) and the Big 5 personality taxonomy (John and Srivastava, 1999), and touched briefly upon the content-related linguistic signals. However, the present study is more interested in comparing the lexical (aka open-vocabulary or bag-of-words) and syntactic signals of control that are embedded in language. If deployed on a large scale with the informed consent of the authors, it may allow unobtrusive, cost-effective estimates of well-being[1].

Signals within the language should provide insight into the cognitive sense of control experienced by an individual; however, to our knowledge, no study has examined whether these signals are stronger in the *way* people express themselves, or *what* they say. Although the discourse and lexical signals that convey status and dominance have been studied in previous work, they involve controlling *other* people (Danescu-Niculescu-Mizil et al., 2012; Fast and Horvitz, 2016), while locus of control is an assessment of the writer's own affairs – perhaps indicated through lexical features that focus on first person pronouns.

Dominance has also been qualitatively measured in terms of the affective context at the level of an utterance (Bradley and Lang, 1999); however, we found that this also did not correspond to our notion of locus of control. With reference to the syntactic features, we will test whether control may be understood through different verb forms, and active versus passive voice. Locus of control is different from semantic role labeling: for internal control, we are mainly interested in *whether* the author is the *agent* (rather than identifying who the agent is, in a sentence), and how much they control their life. External control includes a complex mixture of semantic roles such as *patient, experiencer, theme, recipient*, and *beneficiary*.

3 Locus of control data

Posts on Facebook capture self-expression by a diverse audience about their daily lives, which makes it a natural starting point for exploring the linguistic features of control. For data collection, we deployed a survey on Qualtrics comprising several demographic questions, the Sense of Control facet items from the MIDUS survey (Brim et al., 2004) and a 3-item health inventory measuring poor health, general health and weekly excercise, taken from the CDCs Behavioral Risk Factor Surveillance System (BRFSS) and the European Social Survey (ESS). We invited users to share access to their Facebook status updates and randomly sampled 2000 users who had posted at least 100 words (with 839 status updates on average). We then randomly sampled 2 status updates per user to obtain 4000 statuses. We cleaned up the text by removing URLs, hashtags, user handles, etc. and split each status into sentences using our in-house pattern-based sentence splitter inspired by CMU ARK Twitter Twokenize script (Owoputi et al., 2013). Finally, from each status update, we selected one sentence that has 2 or more words.

3.1 Locus of control annotation

Building a useful computational model requires labeled training data. We labeled the Facebook dataset using three trained annotators pursuing a Master's program in Psychology, to construct the first public corpus annotated with control. We asked the annotators to determine whether the author of the sentence is in control (internal control) or being controlled by others or circumstances (external control). To ensure quality work, we provided examples corresponding to each point on the scale. The examples provided to the annotators are also provided in the annotation scheme described in the Appendix.

In this paper, we focus on Stage 1 and Stage 2 of annotation, which distinguish control-relevant sentences and classify them as either internal or external control. We evaluate the reliability of

---

[1] http://www.pewinternet.org/2016/06/22/social-media-and-the-workplace/

the annotations by computing percent-agreement. The highest pairwise inter-annotator agreement between annotators is 90.2% for Stage 1 and 79% for Stage 2.

4 Methods

To train and test the classifiers for the Stage 1 and 2, we use the Differential Language Analysis ToolKit[2] (DLATK) (Schwartz et al., 2017), a Python package developed for social media text analysis. Lexical features are extracted in DLATK and syntax-based features are separately generated and imported into DLATK feature sets.

4.1 Lexical features

ngrams: Normalized frequency distribution over 1-, 2- and 3-grams

LIWC: Frequency distribution of categories from the Linguistic Inquiry and (LIWC) (Pennebaker et al., 2007).

ANEW: Weighted frequency distribution of the Dominance score in the Affective Norms in English Words (ANEW) (Bradley and Lang, 1999).

Word2Vec Topics: We use word clusters built on top of Word2Vec, by Preoţiuc-Pietro, Lampos and Aletras (2015). They built 200 open-ended word clusters by applying spectral clustering to the word-to-word similarity matrix from the neural embeddings Mikolov et. al (2013).

Pronouns: We clustered all pronouns (except for possessives) into 1st-, 2nd-, and 3rd-person regardless of their syntactic role.

4.2 Syntax-based features

We acquire dependency parses of our corpus by SyntaxNet (Andor et al., 2016) with Parsey McParseface model[3] that produces universal dependencies in *relation, head, dependent* triples in CONLL format. We obtain *subject-verb* tuples (SVs) and *subject-verb-object* triples (SVOs) from the dependency trees. In our in-house evaluation on a random set of 100 Tweets, SyntaxNet with the Parsey McParseface model outperforms the Stanford Parser (Socher et al., 2013) on extracting SVs and SVOs from social media (P=.75, R= 68, TN Rate =.90 for the former; P=.51, R= .55, TN Rate =.80 for the latter). SyntaxNet is also a better tool for our purpose compared to the Tweebo Parser (Kong et al., 2014), that only provides dependency graphs and not the relations.

Verb predicate features: We identify the verb classes using (a) linguistically-driven Levin's Classes (*Lev*) (Levin, 1993), and (b) an in-house manual verb clustering on the most frequent 130 verb into 40 semantic classes (*M*). Inspired by the previous research (Rouhizadeh et al., 2016), we extract five sets of dependency-driven verb predicate features. (1) **Pronouns-SVO**: occurrence of 1st, 2nd, and 3rd person pronouns in the subject/object positions, (2) **VerbCat-1-2-3PP** occurrence of 1st, 2nd, and 3rd person pronouns in the subject/object positions of each verb category (from the Levin's or our own verb classes), (3) **VerbCat-1PP**: occurrence of 1st or non-1st pronouns (i.e. self-reference ratio) in the subject/object positions of each verb category, (4) **VerbCat-SVO**: all words in the subject/object positions of each verb category, and (5) **VerbCat-all** categories of all the verbs in subject-verb and subject-verb-object contexts.

POS-ngrams: We capture shallow syntactic features by constructing Penn Part-of-Speech (POS) unigrams, bigrams and trigrams after tagging every word with their POS information by using the Python NLTK package (Bird et al., 2009).

5 Results

We train multiple classifiers on different sets of lexical and syntactic features after performing feature selection based on univariate regression between each feature and the outcome. We also perform dimensionality reduction by randomized principal component analysis (PCA). After evaluating a number of classification models available in Pythons sklearn package, we report the results from the logistic regression classifier, with L2 penalization, and inverse of regularization strength set to 0.1, which obtain the best results. Although we have experimented with a variety of feature combinations which are included in the Appendix, we discuss selected classifiers (for meaningful comparisons) as well as the best-performing ones.

5.1 Predicting control relevance

We report the results for prediction performance of the main feature in Table 1a. We see that lexical ngrams are moderately effective, whereas LIWC, ANEW, and w2v features are not helpful. On the other hand, pronouns appear to be very

---

[2] https://dlatk.wwbp.org

[3] `https://github.com/tensorflow/models/tree/master/research/syntaxnet`

(a) Control Relevance

	P	R	F1
Lexical features			
ngram	.854	.918	.885
Pronouns	.858	1.00	.923
Syntax-based features			
VerbCat-all (M)	.878	.997	.934
VerbCat-all (Lev)	.903	.997	.948
POS-ngram	.849	.913	.880
Best combination			
VerbCat-1PP (Lev)	.917	.990	**.952**
% Agreement for Human annotators: 90.2			

(b) Internal/External Control

	P	R	F1
Lexical features			
ngram	.726	.682	.702
Pronouns	.236	.204	.218
Syntax-based features			
VerbCat-all (M)	.532	.189	.280
VerbCat-all (Lev)	.597	.246	.347
POS-ngram	.635	.688	.659
Best combination			
ngram,LIWC,w2v,VerbCat-1PP(Lev)	.700	.737	**.718**
% Agreement for Human annotators: 79.0			

Table 1: Precision, recall and F1-measure for predicting control-relevance and internal/external control (10-fold CV) from lexical and syntactic features.

helpful, despite their linguistic and semantic simplicity. Among the syntax-based features, using the verb categories of SV and SVOs provide the second best F1 score (Levin's classes are better than our classes), and adding self-reference ratio to verb categories generates the best results. Interestingly, POS-ngrams perform on-par with the lexical ngrams, suggesting that the encoded syntactic information in pos-sequences is just as good as lexical information in ngrams with considerably less sparsity and more efficient computation.

5.2 Predicting Internal/External control

The best results for classifying internal vs. external control are in Table 1b. Unlike control relevance, ngrams are the most helpful features here and pronouns and verb category features result in a poor performance. LIWC, ANEW, and w2v do not noticeably improve the results when combined with ngrams, but adding verb categories and self-reference ratio to this combination creates the best result. This suggests that identifying internal/external control benefits from an ensemble of syntactic, semantic and lexical features and simple lexical features are more helpful than word embeddings. Similar to control relevance, POS-ngrams are helpful here although they are not as good as lexical ngrams.

5.3 Linguistic insights

Table 2 shows the most predictive verb categories (with subject or subject-object arguments) based on logistic regression coefficient. We see that not only does the magnitude of the relationship change, but it is possible that the direction can completely change (positive values are control relevance in Table 2a and external control in Table 2b. Interestingly, we see that verbs for audio/visual activities, demand, and start/end of processes, are more frequently used in control-related contexts, whereas using love and like verbs is an informative signal that a given sentence is not related to control. In addition, verbs of cognition, missing, feeling, and hope are positively associated with being controlled by others or the environment, whereas verbs indicating attempt are correlated with author's control of the situation. Although not complete, these semantic categories are intuitively well-associated with the psychological theory of the control concept. In the Appendix, we also identify the POS ngram patterns which are significantly correlated with both, control relevance and internal vs. external control.

6 Error Analysis

In observing the instances of misclassification, we observed the following general patterns:

- **False negatives:**
  – Control-relevant imperative sentences were frequently misclassified as control-irrelevant, e.g. "have a good day everyone"; "Find Jesus."; "Stay focused!"
  – Internal LoC sentences with modals and 'get' were misclassified as external LoC, e.g., "Finally got a best man!"; "Join me and you can save money on all you purchase here."
- **False positives:**
  – Control-irrelevant sentences with emotion verbs ("like", "hate") were often misclassified as control-relevant, e.g., "I Hate SNOW!" ; "I dont like cologne, but it smells nice."
  – External LoC sentences with possessive pronouns and the verb "make" were misclassified as internal LoC, e.g., "It made my day."; "Is your name math because i have a problem with you."

(a) Control Relevance

Verb category	r	Verbs
Audio-visual	.15**	see, say, look, talk, hear, speak
Demand	.10**	need, want, ask
Start/end	.08**	start, stop, follow, leave
Auxiliary	.07**	be, have, do
Buying	.06**	buy, spend, pay
Like	-.06**	like, love

(b) Internal/External Control

Verb category	r	Verbs
Cognition	.08**	know, think, mean, understand
Missing	.07**	miss
Feeling	.06**	feel
Auxiliary	.06**	be, have, do
Hope	.06**	hope, wish
Attempt	-.07**	try, check

Table 2: Logistic regression coefficient between verb categories and control (positive values are control relevance and external control). **: $p < 0.01$ after Benjamini-Hochberg correction. Features with positive coefficients are shaded blue and those with negative coefficients are shaded red.

7 Validation against psychological LoC

An ideal validation of language-based control against psychological LoC would find that users expressing an internal LoC in their social media posts would also be more likely to have an internal psychological LoC as measured by the Sense of Control scale (Brim et al., 2004), and poorer health as reported in previous work (Birnbaum et al., 2010). With only two observations per user, we speculate that our data does not afford us enough power to confirm this relationship. A Spearman correlation between the counts of internal and external-labeled sentences per user and their self-reported LoC and health items, revealed a weak association between authors of a higher number of sentences labeled as external LoC and their psychological LoC as per the survey questionnaire ($\rho = -.06, p < 0.05$, N = 2000), physical and general health (-0.08< ρ < -0.06, $p <$ 0.01, N = 2000). Authors with higher number of sentences labeled as internal LoC did not show a significant association with internal LoC, but were more likely to exercise regularly ($\rho = 0.04, p <$ 0.05, N = 2000). These findings, although weak, correspond in direction to the findings reported in the literature (Birnbaum et al., 2010), suggesting that future work should investigate to what extent the linguistic expressions of control can extrapolate to psychological LoC.

8 Conclusion

We describe a computational linguistic approach to identify the locus of the author's control in their social media writing. Utilizing its psychological underpinnings, we create an annotated dataset of 4000 sentences, labeled with control relevance and of internal and external control. We show that identifying control is largely dependent on syntactic information for control-relevance and lexical information for internal/external control. From the NLP standpoint, this suggests that solely using bag-of-word features may not be sufficient for predicting specific psychological outcomes. We have also distinguished our work against dominance and thematic roles, which may be the closest approximations to LoC; however these concepts do not completely translate into each other.

Differential language analyses identified interesting associations between verbs categories and control. We found that audio/visual verbs tend to occur significantly more in the control-related text, and some emotion verbs such as "miss" or "feel" are correlated with lack of control of the surroundings. A caveat of using language-based models is that their association with traits is correlational, not causal. Furthermore, the differences in platform affordances (Jaidka et al., 2018c) and diachronic drifts in language use over time (Jaidka et al., 2018a) imply that language models to identify control may need domain adaptation before they are applied on other corpora, language from other time periods and other social media platforms, as well as posthoc domain adaptation before they can scale to measure community traits (Rieman et al., 2017).

Existing psychological theories are mainly based on self-expressed and self-perceived locus of control in questionnaires, but they may be susceptible to self-report biases. Instead, we demonstrate that some of these constructs can reliably be extracted from language samples, such as social media posts, which are unsolicited self-expressions of control.

Internal locus of control has been argued to be important for mental and physical health, and to characterize well-run ("empowered") work teams; We hope that the model presented here can be used–with appropriate consent and privacy–for unobtrusive monitoring of LoC in many therapy and work settings.

Acknowledgement

This work was supported by a grant from the Templeton Religion Trust (ID #TRT0048). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Howard G Birnbaum, Ronald C Kessler, David Kelley, Rym Ben-Hamadi, Vijay N Joish, and Paul E Greenberg. 2010. Employer burden of mild, moderate, and severe major depressive disorder: mental health services utilization and costs, and work performance. *Depression and anxiety*, 27(1):78–89.

Margaret M Bradley and Peter J Lang. 1999. Affective norms for english words (ANEW): Instruction manual and affective ratings. Technical report, Citeseer.

Orville Gilbert Brim, Carol D Ryff, and Ronald C Kessler. 2004. *The MIDUS National Survey: An Overview.* University of Chicago Press.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of the 21st international conference on World Wide Web*, pages 699–708. ACM.

Norman Fairclough. 2001. *Language and power.* Pearson Education.

Ethan Fast and Eric Horvitz. 2016. Identifying dogmatism in social media: Signals and models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 690–699.

James K Harter, Frank L Schmidt, and Corey LM Keyes. 2003. Well-being in the workplace and its relationship to business outcomes: A review of the gallup studies. *Flourishing: Positive psychology and the life well-lived*, 2:205–224.

Kokil Jaidka, Niyati Chhaya, and Lyle Ungar. 2018a. Diachronic degradation of language models: Insights from social media. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 195–200.

Kokil Jaidka, Salvatore Giorgi, Anneke Buffone, Johannes Eichstaedt, Masoud Rouhizadeh, and Lyle Ungar. 2018b. Modeling and visualizing locus of control with facebook language. In *Proceedings of the International AAAI Conference on Web and Social Media.*

Kokil Jaidka, Sharath Chandra Guntuku, Anneke Buffone, H. Andrew Schwartz, and Lyle Ungar. 2018c. Facebook vs. twitter: Cross-platform differences in self-disclosure and trait prediction. In *Proceedings of the International AAAI Conference on Web and Social Media.*

Oliver P John and Sanjay Srivastava. 1999. The big five trait taxonomy: History, measurement, and theoretical perspectives. *Handbook of personality: Theory and research*, 2(1999):102–138.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation.* University of Chicago press.

Roger Levy and Christopher Potts. 2015. Negotiating lexical uncertainty and expertise with disjunction. In *Poster presented at the 89th Annual Meeting of the Linguistic Society of America, Portland.*

Michael G Marmot, Stephen Stansfeld, Chandra Patel, Fiona North, Jenny Head, Ian White, Eric Brunner, Amanda Feeney, and G Davey Smith. 1991. Health inequalities among british civil servants: the whitehall ii study. *The Lancet*, 337(8754):1387–1393.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

James W Pennebaker, Roger J Booth, and Martha E Francis. 2007. Linguistic inquiry and word count: LIWC [computer software]. *Austin, TX: liwc. net.*

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015. An analysis of the user occupational class through twitter content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1754–1764.

Daniel Rieman, Kokil Jaidka, H Andrew Schwartz, and Lyle Ungar. 2017. Domain adaptation from user-level facebook models to county-level twitter predictions. In *Proceedings of the Eighth International*

Joint Conference on Natural Language Processing (Volume 1: Long Papers), volume 1, pages 764–773.

Julian B Rotter. 1966. Generalized expectancies for internal versus external control of reinforcement. *Psychological monographs: General and applied*, 80(1):1.

Masoud Rouhizadeh, Lyle Ungar, Anneke Buffone, and H Andrew Schwartz. 2016. Using syntactic and semantic context to explore psychodemographic differences in self-reference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2059.

H Andrew Schwartz, Salvatore Giorgi, Maarten Sap, Patrick Crutchley, Lyle Ungar, and Johannes Eichstaedt. 2017. Dlatk: Differential language analysis toolkit. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 55–60.

Richard Socher, John Bauer, Christopher D Manning, et al. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 455–465.

Somm: Into the Model

Shengli Hu
Cornell University
sh2264@cornell.edu

Abstract

To what extent could the sommelier profession, or wine stewardship, be displaced by machine learning algorithms? There are at least three essential skills that make a qualified sommelier: wine theory, blind tasting, and beverage service, as exemplified in the rigorous certification processes of certified sommeliers and above (advanced and master) with the most authoritative body in the industry, the Court of Master Sommelier (hereafter CMS). We propose and train corresponding machine learning models that match these skills, and compare algorithmic results with real data collected from a large group of certified wine professionals. We find that our machine learning models outperform human sommeliers on most tasks — most notably in the section of blind tasting, where both hierarchically supervised Latent Dirichlet Allocation outperforms sommeliers' judgment calls by over 6% in terms of F1-score; in the section of beverage service — wine and food pairing, a modified Siamese neural networks based on BiLSTM achieves better results than sommeliers by 2%. This demonstrates, contrary to popular opinion in the industry, that the sommelier profession is at least to some extent automatable, barring economic (Kleinberg et al., 2017) and psychological (Dietvorst et al., 2015) complications.

1. Introduction and Related Work

Thanks to the Somm documentaries and a general increase in awareness about wine, sommeliers, and the Court of Master Sommeliers, there is now a certain celebrity status, a glamor associated with becoming a sommelier. When encountered with the question — "is the sommelier profession going to be negatively affected by recent advances in machine learning and artificial intelligence?" during informal interviews conducted by authors in the sommelier community, there appears to be a general consensus among professionals that the high standards of hospitality upheld by qualified sommeliers are well beyond the capabilities of machines.

The current study asks the question, to what extent would the sommelier profession be displaced by machine learning algorithms? What aspects of the sommelier profession could be outperformed, and therefore perhaps displaced by what kinds of applications of machine learning?

What makes a qualified sommelier or wine professional? According to the Court of Master Sommelier[1], one of the two organizations held in the highest esteem in the global industry, there are at least three indispensable components as exemplified in the certification exams leading up to the Master Sommelier diploma: theory, blind tasting, and service.

To satisfy the theoretical requirement, sommelier candidates are required to sit on a timed exam of various questions covering a wide range of wine topics including geography, soil, viticulture, laws, history, language, etc. without any officially structured study guides[2]. We argue that this particular task maps to the stream of research concerning *open-domain* Question Answering (hereafter, OQA), where the model is given a question and access to a large corpus (Chen et al., 2017), combining and therefore leveraging both the Information Retrieval (Weinberger et al., 2009) and Machine Comprehension literature (Hermann et al., 2015; Chen et al., 2016). In Section 3, we train an open-domain QA model building upon Chen et al.

---

[1] "The Court of Master Sommeliers sets the global standard of excellence for beverage service within the hospitality industry with integrity, exemplary knowledge, and humility." — https://www.mastersommeliers.org/

[2] There are indeed a list of recommended references and an unofficial source of study guides popular among candidates: https://www.guildsomm.com, which we use as our training data.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1153–1159
Brussels, Belgium, October 31 - November 4, 2018.

(2017), on a large corpus of wine topics drawn from recommended study resources by CMS. We contrast the machine performance with sommeliers' performance on equivalent test questions.

To satisfy the blind tasting requirement, candidates have to blind taste a flight of wines, precisely describe the wine, and accurately identify the grape varietal, the region (thus the country), the vintage, and the quality level of each. According to wine programs such as CMS or WSET[3], blind tasting consists of two steps — tasting and deduction. Tasting refers to the sensory experience associated with evaluating wines — color, aroma, favor, aftertaste, etc. Proficient candidates are expected be able to detect a wide range of characteristics of the focal wine, and precisely describe the wine with meaningful descriptors accordingly. Deduction is the logical process that leads the candidate to conclude on the identity of the wine given the characteristics he detects in the first step. According to wine educators and master sommeliers such as Geoff Kruth M.S., it is the deduction part of blind tasting that separates great blind tasters from mediocre ones, mostly due to the fact that it requires greater logical thinking and reasoning. We propose that the deduction step maps exactly to the machine learning task of *structured prediction* (Taskar et al., 2005; Belanger and McCallum, 2016; Barutcuoglu et al., 2006; Rousu et al., 2006). In Section 4, we demonstrate that a hierarchical supervised Latent Dirichlet Allocation model (Perotte et al., 2011; Nguyen et al., 2013) trained on a large corpus of textual descriptions of wines of different grape varietals, regions, vintages, and quality levels, outperforms sommeliers in deduction by a large margin.

To satisfy the service requirement, candidates are grilled on questions of wines and spirits, food and wine pairing, salesmanship, and service mechanics in a restaurant setting. In Section 5, we showcase a modified Siamese Neural Network (Yang et al., 2015; Mueller and Thyagarajan, 2016; Neculoiu et al., 2016; Pei et al., 2016; Bertinetto et al., 2016) coupled with Bidirectional Long Short-term Memory Networks (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997; Zhang et al., 2015) trained on corpora of wine reviews (Hendrickx et al., 2016) and cooking recipes (Tasse and Smith, 2008; Jermsurawong and Habash, 2015) outperforms sommeliers' performance.

2. Data Collection and Preprocessing

Our datasets consist of three parts: (1) *Study Resources*: a large corpus consisting of all the recommended resources for sommelier certification by the CMS, for the Question Answering — Theory Component detailed in Section 3; (2) *Wine Reviews*: a massive repository of expert wine reviews with rich meta-data, based on reviews from Decanter, Vinous, Wine Spectator, and Wine Enthusiast, the four widely recognized media outlets in the industry, for the Structured Prediction — Deduction in Blind Tasting Component detailed in Section 4; (3) *Survey Responses* from $1,305$ certified wine professionals, covering topics on theory, deductive tasting, and wine and food pairing, thus providing experts' performance data with which we compare results from our corresponding machine learning models in Section 3, Section 4, and Section 5.

2.1 Preprocessing

The study resource dataset consists of documents of various categories and topics from the GuildSomm. We treat texts under each sub-category as a document — there are 752 documents in our dataset and the average length of documents is $1,384$ words.

For the wine review dataset, we only consider wines for which we had at least 200 reviews in the training set, leading to $850,119$ reviews combined. When different names were used for the same grape, we normalize these to the same category. For instance, Pinot Bianco (Italy), Pinot Blanc (France), and Weissburgunder (Germany) are mapped together and renamed Pinot Blanc according to the wine grape encyclopedia (Robinson et al., 2013; Robinson and Harding, 2015). We preprocessed all the text data in standard procedures.

2.2 Summary of Datasets

We plot the country and point distributions of our review dataset in Figure 1 , grouped by media outlet. Interestingly, Vinous appears proportionally much more focused on Italian wines and its ratings are more skewed to the right compared to others (a.k.a. greater rating inflation), somehow contrary to the brand image; Wine Spectator is much more focused on Old World whereas Wine Enthusiast is

---

[3] Wine & Spirit Education Trust

more evenly distributed across countries.

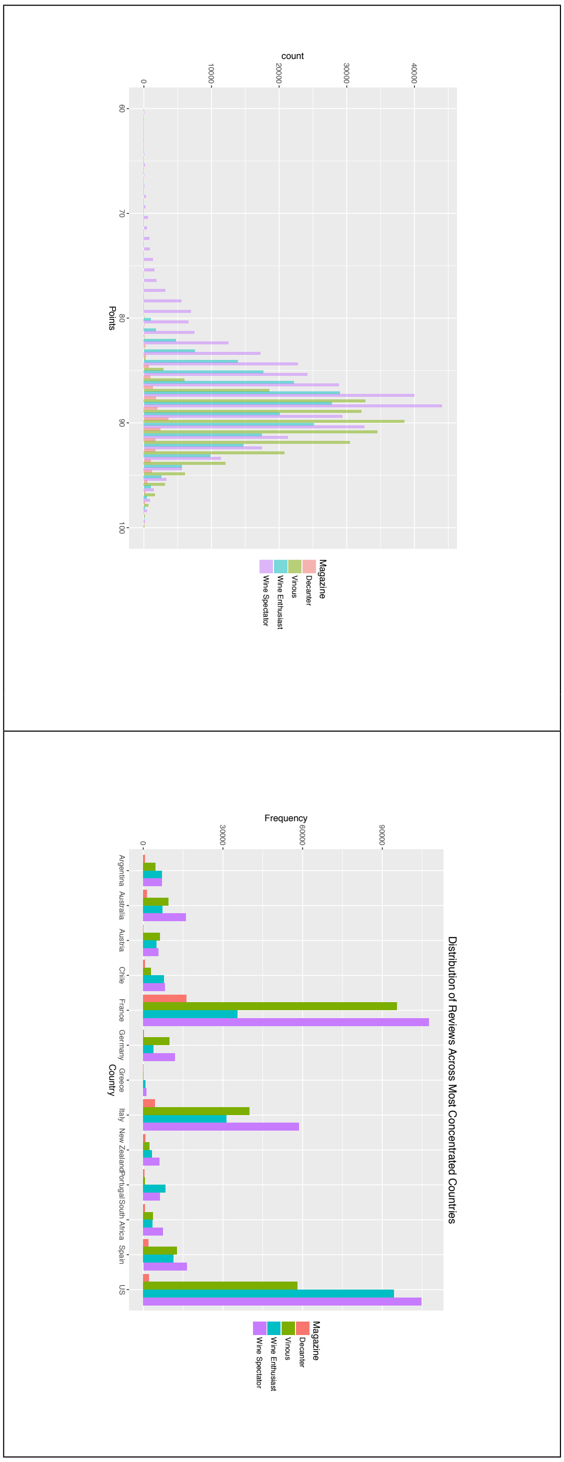

Figure 1: Country and Point Distribution of Wine Reviews

2.3 Survey Details

We administered timed online surveys to wine professionals in several active sommelier communities such as the Guild of Sommeliers, the Society of Wine Educators, etc. Each survey consists of three sections that correspond to the three components respectively — theory, deductive tasting, and pairing. Each section lasts no longer than 15 minutes and consists of 30 questions, randomly drawn from a large pool of practice questions from the Guild of Sommeliers and the Society of Wine Educators.

In the first section, we administered two sets of questions varying difficulty level — one on the level of certified sommelier (CMS level 2), the other on the level of advanced sommelier (CMS level 3). The Question Answering (wine theory) pool consists of $1,400$ questions (700 equivalent to level 2 difficulty, 700 equivalent to level 3 difficulty) from Society of Wine Educators' Certified Specialist of Wine and Certified Specialist of Spirits programs, and $1,480$ questions (740 at level 2 difficulty and 740 at level 3 difficulty) from Guild of Sommelier practice repository. $2,304$ questions were used for training, and the rest for testing. In the second section, we randomly drew 30 textual descriptions from our pool of $850,119$ reviews detailed in Section 2, and asked the subject to deduce the varietal, vintage range, region, and quality level of the wine being described. In the third section, we randomly drew 30 wines (Title and Tasting note) from the repository of wine reviews and 30 recipes from the CMU Recipe Database CURD (Tasse and Smith, 2008), and asked the subject to rate the pairings on a scale from 1 to 5. We circulated our survey to members of the Guild of Sommeliers and Society of Wine Educators communities and received $1,412$ responses. The first section of theory questions serves not only as a dataset compared against QA models, but also as a validation and screening procedure: we removed the responses with fewer than 18[4] correct answers to the 30 questions in section 1, reducing our sample size to $1,305$. Sommelier scores were calculated aggregating all the participants' answers.

3. Wine Theory: open-domain Question Answering

We implemented an open-domain Question Answering system modeled after Chen et al. (2017), consisting of a Document Retriever module and a Document Reader module.

The Document Retriever module finds the three

---

[4] We choose the cutoff rate of 60% because it is the the pass rate in real sommelier exams both for the certified and advanced.

most relevant documents by comparing documents and questions as TF-IDF weighted bag-of-word vectors that include bigrams. We also adopt the hashing of Weinberger et al. (2009) for mapping bigrams with an unsigned murmur3 hash.

The Document Reader module is essentially a bidirectional Long Short-term Memory Network (BiLSTM) (Hochreiter and Schmidhuber, 1997; Zhang et al., 2015) applied to each paragraph in relevant documents, the predicted answers of which are finally aggregated. For detailed procedures of implementation, we refer readers to Chen et al. (2017). The only differences are, our batch size is 25 and we adopted a dropout rate of 0.1. We document the results in Table 1. Surprisingly, our OQA results converge to the high levels of accuracies achievable by machine comprehension models. We argue that it is because our corpus is relatively small and concentrated on wine-related topics, which results in few complications arising from the integration of large-scale information retrieval and machine comprehension, and therefore more germane to single machine comprehension models. Note that we removed survey results below 60% accuracy, stacking the odds against us because now the sommeliers' performance results are inflated, which could provide partial explanations for OQA being behind. The comparison still looks promising, despite the 4.8% disparity in performance.

Data Generation	Training Set		Test Set	
	Exact Match	F1	Exact Match	F1
Sommeliers	NA	NA	67.1	71.7
OQA System	58.1	67.8	55.7	66.9

Table 1: Evaluation results of OQA in comparison with sommeliers' performance.

Comparing the accuracies across regions, we find sommeliers did much better than DrQA in old world regions while DrQA edged out on most new world regions. It might echo the greater emphasis of sommelier training in real life on the old world, and/or reflect the more complications introduced by French, Italian, German, and Spanish terminologies which we didn't correct for when dealing with the old world wine regions.

4. Blind Tasting: HSLDA

We implemented the Hierarchically Supervised Latent Dirichlet Allocation (HSLDA) model (Perotte et al., 2011) for deduction in blind tasting based on textual descriptions of wines, because of the natural fit in-between — the texts describing wines are hierarchically (from top to bottom: grape varietal, country, region, vintage, quality) and multiply (blends vs. monovarietals)) labeled bag-of-word (simple and performant for reviews) data. For model details, we refer readers to Perotte et al. (2011). We use the subset of wine reviews published in Wine Enthusiast and Decanter for this task. It contains $183,660$ wine descriptions for training and $39,150$ for testing. There are 41.1 terms on average in each document, with a 11.6 standard deviation. There are $12,132$ unique (sub-)categories in the form of "Sangiovese, Italy, Tuscany, 2015, Riserva". We use a Gaussian prior over the regression parameters where a range of values for μ, the mean prior parameter for regression coefficients are evaluated ($\mu \in \{-3, -2.5, -2, \ldots, 1\}$). We set the number of topics to 20 based on small sample testing and CMS tasting grid. Prior distributions of hyperparameters are gamma distributed with a shape parameter of 1 and a scale parameter of $1,000$.

Initial results were less satisfying and most errors occurred because mono-varietals were predicted to be blends and vice versa. Therefore we explored two solutions: (1) we separated our data into mono-varietals (Model 1), and blends (Model 2), and trained HSLDAd separately; (2) we created a smaller yet more balanced training set regarding mono-varietals and blends (Model 3).

In Model 1, we simplified "testable" (i.e., included in Court of Master Sommelier tasting exams) blends in our dataset such as "Southern Rhone red blend", "Marsanne Roussanne blend", "Sangiovese blend", and such were treated the same as the mono-varietals. Model 2 and 3 were trained using the exact blending grape varietals. We believe Model 1 is closest to the decision making processes encountered by sommeliers in CMS certification exams, whereas Model 3 is more likely to resemble sommelier challenges such as Top Somm.

We computed precision and recall of all the categories, yielding a $12,132 \times 12,132$ sparse confusion matrix for Model 3, a $11,672 \times 11,672$ sparse confusion matrix for Model 1, and a 386×386 confusion matrix for Model 2. We then averaged them to get a single real number measurement. Table 2 shows the average F1 scores of different models versus sommeliers' performance.

Likewise, the sommeliers' performance measures represent a conservative(ly higher) estimate since scores lower than 60% in section 1 were removed. We find the HSLDA model, especially of monovarietals, outperforms sommeliers by 6.3%, as measured by F1. In aggregate, sommeliers did significantly better in red and sparkling wines, and in French, German, and Californian wines, whereas Model 1 edged out in white wines, south America wines.

F1 Scores	Training Set	Testing Set
HSLDA1 Monovarietal	71.1	68.4
HSLDA2 Blend	62.5	59.1
HSLDA3 Balanced	59.8	56.4
Sommeliers	NA	62.1

Table 2: Evaluation results of HSLDAs in comparison with sommeliers' performance.

5. Food and Wine Pairing: Siamese Neural Networks with LSTM

For food and wine pairing, we trained an untied and modified version of Manhattan LSTM (Mueller and Thyagarajan, 2016), where we preprocessed the texts differently and applied LSTM-Based Importance Weighting in place of the original simple similarity function coupled with LSTM. We retained from the recipes only ingredients, serving ingredients and essential actions, which were passed to the BiLSTM (Zhang et al., 2015). For a given pair of wine and recipe descriptions, we applied a weight compatibility function $g(h_{T_a}^{(a)}, h_{T_b}^{(b)}) = exp(-||a^T h_{T_a}^{(a)} - b^T h_{T_b}^{(b)}||_1)$, where a^T and b^T are shared network weights applied to BiLSTM representations $h_{T_a}^{a}$ and $h_{T_b}^{b}$. For model architecture and other implementation details we refer readers to Mueller and Thyagarajan (2016) and Rücklé and Gurevych (2017).

We obtained our ground-truth labels for wine and recipe pairings on a scale from 1 to 5 using an automated weighting scheme based on wine and food pairing principles (Goldstein and Goldstein, 2006)[5] leveraging the GuildSomm tasting notes of grape varietals and recipe ingredients, under close guidance of a certified sommelier with the CMS. Details of the weighting scheme is included in our online supplementary documents. In the end we simplified our scale to binary — $\{1, 2\}$ converted to 0, $\{3, 4, 5\}$ converted to 1. We document our accuracies in Table 3. Surprisingly, the model edged out by 1.7%.

Accuracy	Training Set	Testing Set
Modified MaLSTM	82.3	79.8
Sommeliers	NA	78.1

Table 3: Evaluation results of Modified MaLSTM in comparison to sommeliers' performance.

6. Conclusion and Future Work

We examine how machine learning can be used to understand, assist, and improve human decision-making, echoing recent studies in computational social sciences (Dietvorst et al., 2015; Kleinberg et al., 2017). We dissect sommelier skills into three parts and train ML models for each. We show with our choices of suitable models, collection of valuable datasets and annotations, that ML algorithms outperform sommeliers in essential skills. Future work could improve on:

1. fine-tuning the Open-domain Question Answering for wine knowledge;
2. connecting our HSLDA or hierarchical multi-label classification to robotic sensors to fully mimic the blind tasting task;
3. exploring other simpler and more efficient ML models for pairing tasks;
4. training a joint multi-task model, since it is accepted in the industry that a solid knowledge of wine theory helps immensely in blind tasting and wine service. It would be interesting to quantify the synergy in the learning process;
5. exploring ML applications to other aspects of the service component.

References

Zafer Barutcuoglu, Robert E Schapire, and Olga G Troyanskaya. 2006. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992.

---

[5] One of the few recommended resources for certified sommelier candidates on wine and food pairing.

Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. 2016. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Berkeley J Dietvorst, Joseph P Simmons, and Cade Massey. 2015. Algorithm aversion: People erroneously avoid algorithms after seeing them err. *Journal of Experimental Psychology: General*, 144(1):114.

Evan Goldstein and Joyce Goldstein. 2006. *Perfect Pairings: A Master Sommeliers Practical Advice for Partnering Wine with Food*. Univ of California Press.

Iris Hendrickx, Els Lefever, Ilja Croijmans, Asifa Majid, and Antal van den Bosch. 2016. Very quaffable and great fun: Applying nlp to wine reviews. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 306–312.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

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

Jermsak Jermsurawong and Nizar Habash. 2015. Predicting the structure of cooking recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 781–786.

Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. 2017. Human decisions and machine predictions. *The Quarterly Journal of Economics*, 133(1):237–293.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157.

Viet-An Nguyen, Jordan L Boyd-Graber, and Philip Resnik. 2013. Lexical and hierarchical topic regression. In *Advances in neural information processing systems*, pages 1106–1114.

Wenjie Pei, David MJ Tax, and Laurens van der Maaten. 2016. Modeling time series similarity with siamese recurrent networks. *arXiv preprint arXiv:1603.04713*.

Adler J Perotte, Frank Wood, Noemie Elhadad, and Nicholas Bartlett. 2011. Hierarchically supervised latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 2609–2617.

Jancis Robinson and Julia Harding. 2015. *The Oxford companion to wine*. American Chemical Society.

Jancis Robinson, Julia Harding, and José Vouillamoz. 2013. *Wine Grapes: A complete guide to 1,368 vine varieties, including their origins and flavours*. Penguin UK.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. 2006. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7(Jul):1601–1626.

Andreas Rücklé and Iryna Gurevych. 2017. Representation learning for answer selection with lstm-based importance weighting. In *IWCS 201712th International Conference on Computational SemanticsShort papers*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM.

Dan Tasse and Noah A Smith. 2008. Sour cream: Toward semantic processing of recipes. *Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU-LTI-08-005*.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM.

Longqi Yang, Yin Cui, Fan Zhang, John P Pollak, Serge Belongie, and Deborah Estrin. 2015. Plateclick: Bootstrapping food preferences through an adaptive visual interface. In *Proceedings of the 24th acm international on conference on information and knowledge management*, pages 183–192. ACM.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 73–78.

Fine-Grained Emotion Detection in Health-Related Online Posts

Hamed Khanpour
Computer Science and Engineering
University of North Texas
Denton, TX 76207
`hamed.khanpour@microsoft.com`

Cornelia Caragea
Computer Science
Kansas State University
Manhattan, KS 66502
`ccaragea@ksu.edu`

Abstract

Detecting fine-grained emotions in online health communities provides insightful information about patients' emotional states. However, current computational approaches to emotion detection from health-related posts focus only on identifying messages that contain emotions, with no emphasis on the emotion type, using a set of handcrafted features. In this paper, we take a step further and propose to detect fine-grained emotion types from health-related posts and show how high-level and abstract features derived from deep neural networks combined with lexicon-based features can be employed to detect emotions.

1 Introduction

Emotions are an essential part of our lives and reflect feelings such as joy, sadness, and fear, which affect our overall wellbeing. Emotion detection from text using computational models has been extensively studied from data such as news headlines, social media, blog posts, and song lyrics (Katz et al., 2007; Abdul-Mageed and Ungar, 2017; Mohammad and Turney, 2013; Strapparava and Mihalcea, 2007; Liew and Turtle, 2016).

Recently, emotion detection started to emerge in online health communities (OHCs) (Khanpour et al., 2018; Biyani et al., 2014; Wang et al., 2012). OHCs provide a user-friendly environment for patients, their families and friends to share thoughts and socialize with each other on topics such as therapeutic processes, side effects, and mental and emotional health. Emotion detection in OHCs is substantially different from the general text due to a health-related vocabulary that people use in OHCs. For example, the phrase "hot flashes" may not have a specific meaning in the general domain, but it bears a very negative feeling in patients' posts. Similarly, the post: "Just received notice from my doctor, I have a positive Cologuard result" is associated with a very sad emotion in a health domain, i.e., when a test is positive, it means that the disease is present.

"My doctor's office is very clean, who cares when he has prescribed me a wrong medication for six months!"

Table 1: Example of an emotional message from an OHC that contains a sad emotion.

Despite of emergence of emotion detection in OHCs, most of the recent works have been devoted to high level emotion analysis, i.e., identifying messages that contain emotions, with no emphasis on the unique challenges associated with fine-grained emotion detection. In order to correctly detect the types of emotions present in messages posted in OHCs, a deep understanding of the text and the writer's intention are required. Table 1 shows an example of a message that contains a sad emotion, which is hidden in text. We ran several sentiment tools on this message, including Stanford CoreNLP (Socher et al., 2013), and interestingly, all showed a positive sentiment, while the emotion expressed is clearly one of sadness (mixed with sarcasm). Thus, even an approach to predict the negative sentiment of the message would not suffice.

In this paper, we analyze messages in OHCs to understand the most prominent emotions in health-related posts and propose a computational model that is able to exploit the semantic information from text and coherently combines high-level (abstract) features with surface and lexicon-based features to automatically detect fine-grained emotions. Our contributions are as follows:

1. We study fine-grained emotions and their distribution in messages posted in OHCs by constructing and analyzing two health-related datasets for fine-grained emotion detection.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1160–1166
Brussels, Belgium, October 31 - November 4, 2018.

Identifying emotions in patients' messages augments the capability of OHCs' moderators, caregivers, and doctors to provide high-quality services to OHCs' users and patients. To our knowledge, we are the first to address fine-grained emotion detection in OHCs.

2. We propose a computational model, called ConvLexLSTM, for emotion detection in OHCs. Our model combines the output of a Convolutional Neural Network (CNN) with lexicon-based features, which are all fed into a Long Short-Term Memory (LSTM) network that produces the final output via softmax. We show empirically that ConvLexLSTM significantly outperforms strong baselines and prior works. Moreover, we show that the proposed model continues to perform well even in the absence of lexicon features.
3. Finally, we apply ConvLexLSTM in a large scale experiment to study the correlation between US holidays and users' emotional states, which can help design smarter approaches to improve patients' moods.

2 Related Work

Emotion detection has been extensively studied in computational linguistics for a long time (Mohammad and Turney, 2013; Strapparava et al., 2004). The Ekman's basic emotion set, which includes six emotions: anger, disgust, fear, happiness, sadness, and surprise, is arguably the best well-known emotion categorization (Ekman, 1992). Strapparava and Mihalcea (2008) proposed knowledge-based and corpus-based methods for classifying emotions based on Ekman's basic emotions. Co-occurrence of general words with emotional words has beed used by Katz et al. (2007) for identifying emotion types latent in news headlines. Keyword-based approaches that are based on finding emotional words in text suffer from the inability to classify text that lacks specific keywords. Therefore, Bao et al. (2009) proposed to use topical relations between words and emotion types for emotion classification in online news. Strapparava et al. (2012) studied emotions from song lyrics.

Emotion detection has been studied in social media as well and brings additional challenges due to their informal context in which people do not follow grammatical rules and use many characters that do not occur in formal text (e.g., #, :)). Emotion lexicons derived from social media, e.g., based on *emotion word hashtags*, have been shown to improve models' performance for emotion detection (Mohammad, 2012; Sykora et al., 2013). Abdul-Mageed and Ungar (2017) used distant supervision to construct a large dataset from the general Twitter for fine-grained emotion detection and explored deep learning models to detect emotions. Liew and Turtle (2016) and Liew et al. (2016) also created a dataset of about $15,500$ tweets labeled with 28 emotion types, using the Amazon Mechanical Turk. Some studies combined knowledge-based and keyword-based approaches with linguistic features and used a machine learning algorithm to reasonably classify sentences with no emotional keywords (Yang et al., 2012; Neviarouskaya et al., 2010).

Emotional support is considered as one of the main advantages of using OHCs that brings better feelings (Khanpour et al., 2017; Zhang et al., 2017a,b; Qiu et al., 2011) and fewer mortality odds to patients (Kroenke et al., 2013). Interestingly, to this date, only very few studies have started to analyze emotions in OHCs using computational models (Wang et al., 2012; Biyani et al., 2014; Wang et al., 2014b,a). For example, Wang et al. (2012) used linear regression to identify emotional support from a cancer forum. Their model predicts to what extent each sentence contains either emotional or non-emotional support. Their features include: LIWC features, POS tags, message length, subjectivity intensity, and LDA topical features. Biyani et al. (2014) identified messages that contain emotions in a breast cancer forum using unigrams, POS tags, structural patterns, and five lexicons that contain strong and weak subjective words, cancer drugs, side-effects, and cancer procedures, and showed that lexicons features have a high impact on the results. Khanpour and Caragea (2018) used deep learning to extract therapeutic processes and side effects from patients' posts. Wang et al. (2014b) classified OHCs' messages based on the intention of the participant when writing messages (e.g., seeking or providing information) and used a combination of features from Wang et al. (2012) coupled with lexicon features used in Biyani et al. (2014).

3 Data Collection and Annotation

To study the most prominent emotions and their distribution in OHCs and to evaluate our model for fine-grained emotion detection in OHCs, we constructed two benchmark datasets, since, to our knowledge, no labeled dataset is available for this task. The first dataset is created by using data from Biyani et al. (2014), which contains 1066 sentences from the breast cancer discussion board in the Cancer Survivors' Network (CSN) of the American Cancer Society, denoted as B-DS. Note that Biyani et al. (2014) performed sentence level classification since longer messages often comprise different topics. A sentence level analysis can make a better estimation on the purpose of the commentator in writing that sentence, whether or not expressing his or her emotions. For the second dataset, we randomly selected 225 comments from 21 discussion threads in the lung cancer discussion board of CSN. We denote this second dataset as L-DS. Following Biyani et al. (2014), we extracted all sentences out of comments and chose sentences with a length greater than four words. We ended up with 1041 sentences in L-DS. In total, we have 2107 sentences labeled with emotion types.

For our annotation task, we followed the six emotions suggested by Ekman (1992). Our annotators were allowed to attribute one or more emotions to a single sentence, e.g., a sentence could be annotated as bearing *sadness* or a combination of *sadness* and *fear*. The annotation task was conducted iteratively following prior studies and guidelines, using three training rounds (DMello, 2016; Fort et al., 2016; Shanahan et al., 2006). In each round, 300 sentences drawn from both B-DS and L-DS were assigned to the annotators,[1] and we asked them to meet with the researchers in a group to discuss disagreements and document their discussion before the next 300 instances were assigned. Upon passing the training period, annotators were assigned to annotate the remaining sentences from B-DS and L-DS, and they ended up with 83% Kappa inter-annotator agreement. For the remaining 17%, the annotators expressed their views on each case in the presence of the researchers and finally 100% agreement was achieved during 20 days. All these sentences plus the 900 ones used during the three training rounds became part of the final dataset. Table 2 represents the distribution of emotions in 2107 sentences. Note that some sentences do not contain any emotion. As can be seen from the table, both datasets have a similar distribution of emotions, with joy and sadness being the most prominent.

Emotions	Lung	Breast	Percentage(%)
Anger	59	29	4.0
Disgust	4	1	0.2
Fear	33	39	3.4
Joy	368	470	39.7
Sadness	183	134	15.0
Surprise	8	3	0.5

Table 2: Emotion distributions in B-DS and L-DS with the percentage estimated from both L-DS and B-DS.

4 Model

Given a sentence of n words, we apply CNN to extract high-level (abstract) features that capture the semantic part of the text (Lai et al., 2015). We combine high-level features with surface-level and lexicon-based features. Our proposed model, ConvLexLSTM, is shown graphically in Figure 1. As can be seen from the figure, we use a combination of CNN and LSTM models, where the final feature vectors from CNN augmented with lexicon-based features are fed as input to the LSTM network. The architecture of our proposed classification model is close to the models described in Kim et al. (2016) and Xiao and Cho (2016), in which they applied a character-level CNN to create high-level features, whereas our model works at word-level and uses lexicon features. We use the word-level input to benefit from applying embedding vectors, trained on OHCs. We use the character-level model by Kim et al. (2016), denoted by C-ConvLSTM, as one of our baselines.

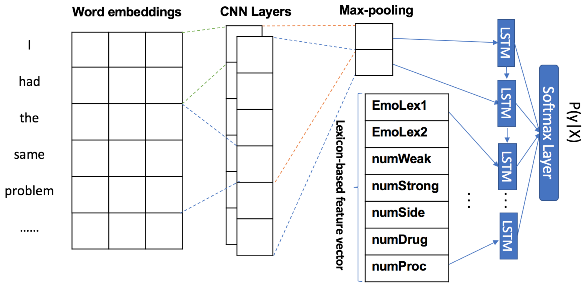

Figure 1: The structure of ConvLexLSTM.

Lexicon-based Features: Lexicon-based approaches for detecting emotions in the text have been the main stream of many models (Strapparava et al., 2004; Strapparava and Mihalcea, 2008; Mohammad, 2012; Liu, 2012).

In our work, we used the same lexicons that were provided by Biyani et al. (2014) such as weak subjective words (**numWeak**), strong subjective words (**numStrong**) (Stoyanov et al., 2005), cancer drugs (**numDrug**), side-effects (**numSide**), and therapeutic procedures (**numProc**). These

---

[1] Annotators were three graduate students.

lexicons address differentiation between emotional versus non-emotional messages. However, we need more granular information for differentiating between a variety of emotion types. Hence, we also used lexicons introduced by Strapparava and Mihalcea (2007), denoted as **EmoLex1**, and by Mohammad and Turney (2013), denoted as **EmoLex2**. We use frequencies of lexicon words to construct the lexicon-based feature vectors. Note that prior work (Biyani et al., 2014) showed that LIWC did not generate high quality features for classification in OHCs, and thus, we did not use it in our study.

5 Experiments and Results

Next, we describe the evaluation of ConvLexLSTM, using the *joy* and *sadness* emotions, which have at least 5% coverage in our labeled data (see Table 2). Also, since binary tasks are considered easier to learn than multi-class tasks (Bishop, 2006), we trained our models in the two-class setting: joy/non-joy (and sad/non-sad), by binarizing the datasets. In all experiments, we used word embeddings as input to the neural networks, which were generated with the W2vector module in Gensim (Řehůřek and Sojka, 2010) on the data (users' comments) from all discussion boards of the Cancer Survivors' Network (CSN) of the American Cancer Society, between June 2000 and June 2012. We also experimented with word embeddings generated using Wikipedia, but these embeddings resulted in lower performance as compared with those generated using CSN data. We estimated hyper-parameters for each deep neural network via a grid search over combinations of important hyper-parameters (e.g., learning rate, decay rate, dropout, number of layers, filter region size, and number of filters). The grid search was done on a development set that consists of removing 20% of instances from the training set in each iteration of 10-fold cross-validation. We report precision, recall and F1-score.

5.1 Performance of ConvLexLSTM

First, we evaluate ConvLexLSTM performance in an ablation experiment to determine the role played by each component for emotion detection. Specifically, we compare ConvLexLSTM with ConvLSTM (a model that has the same architecture as ConvLexLSTM, but does not use any external lexicon), CNN, LSTM, and support vector machine (SVM) with the (concatenated) features from the seven lexicons (described above).

	Method	B-DS			L-DS		
		Pr	**Re**	**F1**	**Pr**	**Re**	**F1**
Joy	ConvLexLSTM	**92.3**	**94.3**	**93.2**	**90.4**	**89.3**	**89.8**
	ConvLSTM	86.6	88.4	87.4	87.0	83.0	85.0
	CNN	85.0	84.0	84.5	82.2	82.8	82.5
	LSTM	86.0	86.6	86.3	85.0	83.0	84.0
	Seven-Lexicon	63.4	87.3	73.45	60.0	85.1	70.37
	C-ConvLSTM	86.2	87.0	86.6	85.0	82.0	83.47
	SWAT	66.0	68.0	67.0	65.5	66.7	66.0
	EmoSVM	81.0	82.0	81.5	82.0	80.0	81.0
Sad	ConvLexLSTM	**93.7**	**91.1**	**92.3**	**88.0**	**90.9**	**89.4**
	ConvLSTM	89.0	87.8	88.4	81.0	87.5	84.0
	CNN	83.2	83.6	83.4	81.7	80.5	81.0
	LSTM	87.4	85.8	86.6	83.2	83.6	83.4
	Seven-Lexicon	61.0	84.9	70.99	61.0	83.3	70.42
	C-ConvLSTM	85.0	83.6	84.3	83.7	82.1	82.9
	SWAT	65.0	66.0	65.5	64.0	65.0	64.5
	EmoSVM	80.5	81.7	81.0	79.0	78.0	78.5

Table 3: Emotion detection results using 10-fold cross validation. The numbers are percentages.

Table 3 shows the results of this comparison. As can be seen from the table, ConvLexLSTM achieves the best results consistently throughout all experiments in terms of all compared measures. This ablation experiment confirms our intuition that all components are contributing to the final emotion detection. For example, removing the seven lexicon features from ConvLexLSTM, which yields ConvLSTM, results in a drop in F1-score by 5.8% on *joy* in B-DS, and by 3.9% on *sadness* in B-DS. Still, ConvLSTM is the second performing model in terms of F1-score. These results show that our model can be successfully applied in a health domain even in the absence of health lexicons, which are often expensive to obtain. Not surprisingly, the SVM with the seven-lexicon based features (denoted as Seven-Lexicon) performs the worst among the compared models, suggesting that capturing the semantic information from text via deep neural networks improves emotion detection.

Second, we compare ConvLexLSTM with three baselines: C-ConvLSTM (i.e., a character-level CNN-LSTM) (Kim et al., 2016), SWAT (Katz et al., 2007) (i.e., an emotion detection model from SemEval-2007), and EmoSVM (i.e., an SVM with a set of handcrafted features: unigrams, bigrams, POS tags, the word-emotions association lexicon by Mohammad (2012), the WordNet-Affect lexicon by Strapparava et al. (2004), and the output of the Stanford sentiment tool by Socher et al. (2013). Table 3 shows the results of this comparison as well. As can be seen, ConvLexLSTM out-

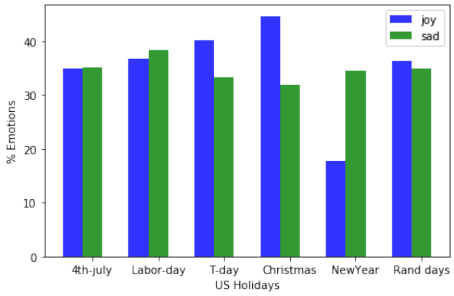

Figure 2: Joy and sadness throughout events.

performs all three baselines on both datasets, and more importantly, the character-level CNN-LSTM by Kim et al. (2016) (i.e., the C-ConvLSTM model). This result confirms our belief that applying word embedding vectors, which are trained directly on data from OHCs yields improvement in performance over character-level models.

It is worth mentioning that all deep neural networks, ConvLexLSTM, ConvLSTM, CNN, LSTM, and C-ConvLSTM, that capture high-level semantic features perform better than the traditional models on emotion detection. The lexicon-based features act as a complement (for the high-level semantic features) by looking into exact words in the text to generate appropriate features in ConvLexLSTM for emotion detection. With a paired T-test, the improvements of ConvLexLSTM over the compared models for F1-score are statistically significant for p-values < 0.05.

5.2 Impact of holidays on emotional states

We further analyzed the impact of several US holidays, i.e., 4th of July, Labor Day, Thanksgiving Day, Christmas Day, and New Year's Eve, on CSN users' emotional states, joy and sadness. For this experiment, we extracted messages from each event day itself and from two days before and two days after each event (holiday) from the entire CSN data. We also collected messages written on five random days (June 15, October 20, March 9, April 3, and January 29), which are not close to any event, to be used as a baseline for comparing the emotional states of participants in different holidays. Consistent with our labeled datasets, we removed messages with less than four tokens. We used our best performing model ConvLexLSTM to classify joy and sadness on these data.

Figure 2 shows the percentage of joy and sad predicted messages for each holiday and for the five random days. As can be seen from the figure, on the random days, the percentage of joy and sad emotions are similar, and so are they for 4th of July and Labor Day, whereas Christmas and Thanksgiving show more joyful spirits, possibly due to family gatherings and other social events around these holidays, in which people feel supported, and hence, feel better. Christmas shows an increase in joy and a slight decrease in sad emotions compared with Thanksgiving. Interestingly, around the New Year's Eve, the percentage of sad messages is almost twice higher compared with the percentage of joy messages, which can be attributed to the end of the holiday season and family gatherings and the beginning of a new challenging year.

6 Conclusion

In this paper, we addressed the problem of fine-grained emotion detection from OHCs messages. To this end, we first annotated a dataset from a cancer forum (i.e., the Cancer Survivors' Network of the American Cancer Society) with the six most common emotions suggested by Ekman (1992) and studied the most prominent emotions and their distribution in OHCs. We found that joy and sadness occur most frequently in the forum, followed by anger and fear. Not surprisingly, disgust and surprise appear the least number of times. We then proposed a computational model that combines the strengths of CNNs, LSTMs and lexicon-based approaches to capture the hidden semantics in OHCs messages and to provide a more insightful understanding of emotional messages by identifying their emotion types.

Our results are promising and show that our proposed model, with or without lexicon-based features, which are often expensive to obtain or maintain in a health domain, provides a better emotion type detection compared with strong baselines and prior works. Given our initial success, in the future, it would be valuable to construct a large health-related dataset to cover other types of emotions, e.g., anger or fear.

Acknowledgments

We are grateful to the American Cancer Society for making the Cancer Survivors' Network available to us. We thank Iulia Bivolaru, Krutarth Patel, and Manoj Panchagnula for their help with data preparation. We would also like to thank our reviewers for their insightful comments and feedback, which helped improve our paper.

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 718–728. Association for Computational Linguistics.

Shenghua Bao, Shengliang Xu, Li Zhang, Rong Yan, Zhong Su, Dingyi Han, and Yong Yu. 2009. Joint emotion-topic modeling for social affective text mining. In *Data Mining, 2009. ICDM'09.*, pages 699–704.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

Prakhar Biyani, Cornelia Caragea, Prasenjit Mitra, and John Yen. 2014. Identifying emotional and informational support in online health communities. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 827–836, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Sidney K DMello. 2016. On the influence of an iterative affect annotation approach on inter-observer and self-observer reliability. *IEEE Transactions on Affective Computing*, 7(2):136–149.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Karën Fort et al. 2016. *Collaborative Annotation for Reliable Natural Language Processing: Technical and Sociological Aspects*. Wiley Online Library.

Phil Katz, Matthew Singleton, and Richard Wicentowski. 2007. Swat-mp: the semeval-2007 systems for task 5 and task 14. In *4th international workshop on semantic evaluations*, pages 308–313.

Hamed Khanpour and Cornelia Caragea. 2018. Fine-grained information identification in health related posts. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 1001–1004.

Hamed Khanpour, Cornelia Caragea, and Prakhar Biyani. 2017. Identifying empathetic messages in online health communities. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017, Volume 2: Short Papers*, pages 246–251.

Hamed Khanpour, Cornelia Caragea, and Prakhar Biyani. 2018. Identifying emotional support in online health communities. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749. AAAI Press.

Candyce H Kroenke, Charles Quesenberry, Marilyn L Kwan, Carol Sweeney, Adrienne Castillo, and Bette J Caan. 2013. Social networks, social support, and burden in relationships, and mortality after breast cancer diagnosis in the life after breast cancer epidemiology (lace) study. *Breast cancer research and treatment*, 137(1):261–271.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2267–2273. AAAI Press.

Jasy Suet Yan Liew and Howard R. Turtle. 2016. Exploring fine-grained emotion detection in tweets. In *Proceedings of the Student Research Workshop, SRW@HLT-NAACL 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 73–80.

Jasy Suet Yan Liew, Howard R. Turtle, and Elizabeth D. Liddy. 2016. Emotweet-28: A fine-grained emotion corpus for sentiment analysis. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Saif M Mohammad. 2012. # emotional tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 246–255. Association for Computational Linguistics.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Alena Neviarouskaya, Helmut Prendinger, and Mitsuru Ishizuka. 2010. Emoheart: conveying emotions in second life based on affect sensing from text. *Advances in Human-Computer Interaction*, 2010:1.

Baojun Qiu, Kang Zhao, Prasenjit Mitra, Dinghao Wu, Cornelia Caragea, John Yen, Greta E Greer, and Kenneth Portier. 2011. Get online support, feel better–sentiment analysis and dynamics in an online cancer survivor community. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 274–281. IEEE.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *LREC*, pages 45–50, Valletta, Malta.

James G Shanahan, Yan Qu, and Janyce Wiebe. 2006. *Computing attitude and affect in text: Theory and applications*, volume 20. Springer.

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

Veselin Stoyanov, Claire Cardie, and Janyce Wiebe. 2005. Multi-perspective question answering using the opqa corpus. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 923–930. Association for Computational Linguistics.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *4th Semantic Evaluations*, pages 70–74.

Carlo Strapparava and Rada Mihalcea. 2008. Learning to identify emotions in text. In *ACM Applied computing*, pages 1556–1560.

Carlo Strapparava, Rada Mihalcea, and Alberto Battocchi. 2012. A parallel corpus of music and lyrics annotated with emotions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 2343–2346.

Carlo Strapparava, Alessandro Valitutti, et al. 2004. Wordnet affect: an affective extension of wordnet. In *LREC*, volume 4, pages 1083–1086.

Martin D Sykora, Thomas Jackson, Ann O'Brien, and Suzanne Elayan. 2013. Emotive ontology: Extracting fine-grained emotions from terse, informal messages.

Xi Wang, Kang Zhao, and Nick Street. 2014a. Predicting user engagement in online health communities based on social support activities. In *Ninth INFORMS Workshop on Data Mining and Analytics*.

Xi Wang, Kang Zhao, and Nick Street. 2014b. Social support and user engagement in online health communities. In *Smart Health*, pages 97–110. Springer.

Yi-Chia Wang, Robert Kraut, and John M Levine. 2012. To stay or leave?: the relationship of emotional and informational support to commitment in online health support groups. In *CSCW*, pages 833–842. ACM.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Hui Yang, Alistair Willis, Anne De Roeck, and Bashar Nuseibeh. 2012. A hybrid model for automatic emotion recognition in suicide notes. *Biomedical informatics insights*, 5(Suppl 1):17.

Shaodian Zhang, Tian Kang, Lin Qiu, Weinan Zhang, Yong Yu, and Noémie Elhadad. 2017a. Cataloguing treatments discussed and used in online autism communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 123–131. International World Wide Web Conferences Steering Committee.

Shaodian Zhang, Erin OCarroll Bantum, Jason Owen, Suzanne Bakken, and Noémie Elhadad. 2017b. Online cancer communities as informatics intervention for social support: conceptualization, characterization, and impact. *Journal of the American Medical Informatics Association*, 24(2):451–459.

The Remarkable Benefit of User-Level Aggregation for Lexical-based Population-Level Predictions

Salvatore Giorgi[1] **Daniel Preoţiuc-Pietro**[2] **Anneke Buffone**[1]
Daniel Rieman[1] **Lyle H. Ungar**[2] and **H. Andrew Schwartz**[3]
[1]Department of Psychology, University of Pennsylvania
[2]Computer and Information Science, University of Pennsylvania
[3]Computer Science, Stony Brook University
`sgiorgi@sas.upenn.edu`

Abstract

Nowcasting based on social media text promises to provide unobtrusive and near real-time predictions of community-level outcomes. These outcomes are typically regarding *people*, but the data is often aggregated without regard to users in the Twitter populations of each community. This paper describes a simple yet effective method for building community-level models using Twitter language aggregated by user. Results on four different U.S. county-level tasks, spanning demographic, health, and psychological outcomes show large and consistent improvements in prediction accuracies (e.g. from Pearson $r = .73$ to $.82$ for median income prediction or $r = .37$ to $.47$ for life satisfaction prediction) over the standard approach of aggregating all tweets. We make our aggregated and anonymized community-level data, derived from 37 billion tweets – over 1 billion of which were mapped to counties, available for research.

1 Introduction

Social media is an increasingly popular resource for large-scale population assessment which promises a cheap and non-intrusive complement to standard surveys with finer spatio-temporal scales (Coppersmith et al., 2015; Mowery et al., 2016; Wang et al., 2017). Twitter has been used — among other things — to measure community health (Paul and Dredze, 2011; Mowery et al., 2016; Eichstaedt et al., 2015), well-being (Schwartz et al., 2013), and public opinion on politics (O'Connor et al., 2010; Miranda Filho et al., 2015). By having access to measurements from multiple locations or communities, models trained on text data from social media can be used both to predict future measurements and to provide community estimates where these are lacking or are not robust. Such research is made possible by the massive amount of easily accessible user-generated data from public social media.

However, there has been little research on the way in which such data should be aggregated in order to compute community-level lexical feature estimates. One study explored the benefit of normalizing lexicon word count features as the proportion of users in each county which use at least one word from a given category (Culotta, 2014a). This method lead to a significant improvement but was limited to broad categories of a psychological based dictionary rather than normalizing frequencies of individual lexical features within users before aggregating to the county. Typically, data are aggregated in a "bag of words" style, disregarding tweets and authors (Schwartz et al., 2013; Eichstaedt et al., 2015; Curtis et al., 2018). We find, however, that giving equal weight to each user, rather than to each word or tweet, yields much more accurate community-level predictions.

In this paper, we conduct a series of experiments testing various simple yet intuitive aggregation methods. We show that choice of aggregation methods can result in substantial (one might even say "remarkable") boosts in accuracy when predicting U.S. county level outcomes (e.g. user-to-county aggregation yields a 7% to 27% increase in Pearson correlation). Contributions include (a) validation of aggregation approaches across four outcomes related to health, psychology, and demographics, (b) validation that aggregation has some effect on smaller sample of Twitter data, (c) show the effect of power tweeters (or "super users") and (d) release of resource-intensive community aggregated lexical data.

Related work. This is the first work we know of to explore simple aggregation techniques for population-level prediction tasks from language. Previous work has explored more sophisticated

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1167–1172
Brussels, Belgium, October 31 - November 4, 2018.

adjustments, such as addressing demographic-self selection bias in Twitter community predictions by re-weighting messages, finding small improvements (a 4.5% reduction in symmetric mean absolute percentage error) (Culotta, 2014b). In a political voting intention prediction application, (Lampos et al., 2013) modeled users and words jointly by learning separate regression weights for the two dimensions based on the intuition that each user contributes differently towards the outcome. However, their model was specifically adapted to problems that use time-series outcomes, rather than community-level aggregation. Distributions of lexical features are considered at multiple levels of analysis (message, user and community) in (Almodaresi et al., 2017) though each level considers one type of aggregation. Similar aggregation methods have been used in the context of topic modeling (Latent Dirichlet Allocation (Blei et al., 2003) and Author Topic Model (Rosen-Zvi et al., 2004)) by considering user, hashtag and conversation level aggregations (Alvarez-Melis and Saveski, 2016; Hong and Davison, 2010) but, again, community level aggregation and prediction tasks were not considered.

2 Data

Research was reviewed by an academic institutional review board and deemed exempt.

2.1 Twitter Data Collection

Twitter Sample A random 10% sample of the entire Twitter stream ('GardenHose') was collected between July 2009 and April 2014, which was then supplemented with a random 1% sample from May 2014 to February 2015. The total sample contains approximately 37.6 billion tweets (Preotiuc-Pietro et al., 2012).

County Mapping In order to map each tweet to a location within a county in the United States, we use both self-reported location information in user profiles and latitude/longitude coordinates associated with a tweet. If latitude/longitude coordinates are present then we trivially map the tweet to a county. The self-reported location information is a free text field and we use a cascading set of rules to map this field to a county. The rules are designed to avoid false positives (incorrect mappings) at the expense of fewer mappings. The full details of this process can be found in (Schwartz et al., 2013). Note that the latitude/longitude coordinates are a tweet attribute whereas the self-reported location is a user attribute yet both are used to map tweets to counties. Users are assigned a county by considering their earliest county mapped tweet.

In total, we are able to map 1.78 billion of the 37.6 billion tweets to a US county using the above-mentioned method. The county mapped data set was then filtered to contain only English tweets using the popular langid.py method (Lui and Baldwin, 2012), further reducing our tweet set to 1.64 billion tweets. For experiments with user-level data aggregation, we removed users who made relatively few (less than 30) posts in our data set.

Publicly Available Stream The standard publicly available Twitter stream outputs approximately 1% of the public Tweet volume. Since a 10% sample is not available to most researchers, we replicated a 1% sample by taking a random 10% of our county mapped, English filtered 10% sample. The same process of county mapping, language filtering and user selection was applied to this data resulting in 131 million county mapped English tweets from 1.57 millions users. Table 1 presents the data set statistics.

The County Tweet Lexical Bank The County Tweet Lexical Bank is a U.S. County level data set comprised of two feature sets[1] :

- an aggregated "bag-of-words" count vector across all the county's messages in order to preserve anonymity. The unigrams represent the most frequent words in the data set;[2] ;
- a "bag-of-topics" representation for each county, with 2000 social media-derived topics described in (Schwartz et al., 2013).

Both feature sets will be releases across the 2009-2015 time span as well as individual years. Yearly updates will be included as they become available. As we are only releasing aggregated word-level features, as opposed to raw Tweets, this data release is within Twitter's Terms of Service.

2.2 Outcomes

The following U.S. county demographic, psychological and health variables were used in our prediction tasks. Table 2 gives statistics for each county variable.

---

[1] Available at `https://github.com/wwbp`

[2] While 25,000 features were used in the predictive tasks we removed some features (@-mentions, URLs, etc.) from the data release to preserve anonymity.

	Number of Tweets				Number of Users
	Full Sample	County Mapped	English*	User Level**	
10%	37.6B	1.78B	1.64B	1.53B	5.25M
1%	-	-	199M	131M	1.57M

Table 1: Number of tweets in each section of the resource, including the total number of users. (*) The number of tweets used in the "all" experiments; (**) the number of tweets in the remaining experiments.

	N	Mean	Std Dev	Min	Max	Skew
Income	1750	4.66	0.11	4.33	5.07	0.47
Educat.	1750	21.57	9.46	5.70	70.30	1.20
Life Satis.	1952	3.39	0.03	3.26	3.51	0.02
Heart Dis.	2041	186.66	45.59	54.82	412.32	0.66

Table 2: Descriptives of U.S. County data used in the four prediction tasks.

Income and Education The census data for county median household income (log-transformed to reduce skewness; N=1,750) and percentage of people with a Bachelor's degree (N=1,750) were obtained from the 2010 U.S. Census Bureau's American Community Survey (ACS).

Life Satisfaction To assess subjective well-being we used the average response to the question "In general, how satisfied are you with in your life?" (1 = very dissatisfied and 5 = very satisfied) (Lawless and Lucas, 2011). Estimates are averaged across 2009 and 2010 (N=1,952).

Mortality Rates From the Centers of Disease Control and Prevention (CDC) we collected age-adjusted mortality rates for heart disease (N=2,041). Rates are averaged across 2010-2015.

3 Methods

3.1 Aggregation

Our aim is to use the user-level information based on the assumption that aggregating data first at the user-level would remove biases introduced by non-standard users of the platform. To this end, we explore three types of aggregation: (1) tweet to county, (2) county "bag of words" and (3) user to county.

Tweet to County Here we compute

$$\text{feat}_{i,j} = \frac{1}{N_j} \sum_{k} \mathbb{1}_i(\text{unigram}_k), \qquad (1)$$

where $\mathbb{1}_i$ denotes the indicator function for unigram$_i$. Here the ith feature for the jth unit of analysis (a U.S. county) is equal to the relative frequency of the unigram: the number of times each unigram was mentioned divided by N_j the total number of tweets from county j.

County Next, we use a method which was generally used in past research, which aggregates all messages to a community disregarding any meta-data, including tweet or user information. Previous state-of-the-art results using this method include life satisfaction r = .31 (Schwartz et al., 2013), atheroclerotic heart disease r = .42 (Eichstaedt et al., 2015) and education r = 0.15 (Culotta, 2014b). We therefore consider each county a "bag of words" using (1) with N_j equal to the number of unigrams from county j.

User to County The third method treats the unit of analysis (U.S. county) as a community of users. Therefore, feature weights are extracted at the user level, normalized and then averaged to communities:

$$\text{feat}_{i,j} = \frac{1}{N_j} \sum_{k \in U_j} r_k(\text{unigram}_i), \qquad (2)$$

where U_j is the set of users in county j, N_j is the total number of Twitter users in county j and $r_k(x)$ is the relative frequency of feature x for user k with $i \in \{\text{all unigrams}\}$ and $j \in \{\text{all counties}\}$.

Features

We use as features a list of 2,000 social media-derived topics generated from Latent Dirichlet Allocation (Blei et al., 2003) using the complete MyPersonality Facebook data set consisting of approximately 15 million posts (Schwartz et al., 2013). The topic loadings are computed from the most frequent 25,000 unigrams in our data set. We also use a subset of these unigrams as additional features in our models (25,000 reduced to 10,000).

Experimental setup

For each of the four county level Census and health variables we built three models using 10-fold cross validation with the following features: (1) unigrams, (2) topics and (3) unigrams + topics. For consistency across tasks we only considered counties with 100 or more 30+ tweet users (N=2,041).

	Income	Educat.	Life Satis.	Heart Disease
Tweet to County	.68	.80	.26	.70
County	.73	.80	.37	.70
User to County	**.82**	**.88**	**.47**	**.75**

(a) Unigrams + Topics, Pearson r

	Income	Educat.	Life Satis.	Heart Disease
Tweet to County	.67	.79	.22	.65
County	.72	.78	.37	.64
User to County	**.79**	**.87**	**.44**	**.73**

(b) Unigrams, Pearson r

	Income	Educat.	Life Satis.	Heart Disease
Tweet to County	.65	.77	.31	.71
County	.68	.80	.34	.71
User to County	**.81**	**.87**	**.47**	**.76**

(c) Topics, Pearson r

Table 3: Prediction results (Pearson r) for direct aggregation comparison on the 10% sample.

We used a feature selection pipeline which first removed all low variance features and then features that were not correlated with our census and health data. Principal component analysis was then applied to the reduced feature set for further dimensionality reduction. This preprocessing was used to avoid overfitting, since our model included more independent variables (2000 topic frequencies and/or 10k unigrams) than observations (at most 2,041 counties). For the prediction task we used linear regression with ℓ_2 regularization (Ridge regression) (Eichstaedt et al., 2015). The regression regularization parameter α was set to 1000 using grid search.

Because our initial dataset consisted of 37.6 billion tweets, using distributed IO was crucial for data aggregation and feature extraction. We used a Hadoop-style cluster consisting of 64 disks and 64 CPU cores across 8 physical machines. Over this cluster, we used Hadoop MapReduce for the county mapping step (taking approximately 1 week of wall clock runtime) and Spark for the feature aggregations (taking approximately 1 day of wall clock runtime). The entire pipeline of county mapping, English language filtering, feature extraction and prediction used the DLATK Python package (Schwartz et al., 2017)[3].

	Income	Educat.	Life Satis.	Heart Disease
User to County	**.82**	**.88**	**.47**	**.75**
$N_{user-tweets}$	1.350B	1.350B	1.356B	1.360B
Tweet to County (all)	.72	.81	.36	.71
County (all)	.73	.82	.31	.72
$N_{all-tweets}$	1.621B	1.621B	1.628B	1.634B

Table 4: Prediction results (Pearson r, using unigrams + topics) using full 10% data vs. users with 30+ tweets. The number of tweets used in each task is listed to highlight the fact that the "User to County" tasks use less tweets than the "all" tasks.

Experiments

Using the above setup we perform 3 experiments in order to explore the effects of data aggregation. We 1) directly compare aggregation methods using our 10% data; 2) compare aggregation methods using a 1% sample and, finally, 3) explore the effect of choosing an upper bound on the number of posts per Twitter users, looking at users with less than 50, 500, 1000 posts. This allows us to exclude frequent posters who are potentially organizations or bots.

4 Results and Discussion

Direct aggregation comparison. The results of our predictive experiments on the 10% data can be found in Table 3. Across all four tasks we see that the "User to County" approach outperforms the other aggregation methods, giving a higher Pearson r and obtaining state-of-the-art results for community-level predictions.

We see the largest gains for the "User to County" aggregation for the income outcome, with a 13 point increase in Pearson r for topics alone and a 9 point increase for unigrams + topics.

In Table 4 we remove the 30+ tweet requirement from the "Tweet to County" and "County" methods and compare against the "User to County" method (with the 30+ tweet requirement). Again we see the "User to County" method outperforms all others in spite of the fact that the "User to County" approach uses less data than both "all" approaches, which contains 108 million more tweets.

1% data. In Table 5 we repeat the above experiment on a 1% Twitter sample. Here we see that the "User to County" method outperforms both the "Tweet to County" and "County" methods (with all three tasks using the same number of

---

[3] Available at `https://github.com/dlatk`

	Income		Educat.		Life Satis.		Heart Disease	
Tweet to County	.71	.62	.77	.71	.35	.32	.64	.63
County	.70	.60	.76	.67	.32	.28	.62	.62
User to County	.76	.70	.79	.74	.39	.28	.66	.66
$N_{user-tweets}$	127M	130M	127M	130M	127M	130M	127M	131M
County (all)	.75	.67	.83	.77	.37	.34	.68	.66
$N_{all-tweets}$	191M	195M	191M	195M	191M	197M	191M	198M
$N_{counties}$	949	1750*	949	1750*	954	1952*	960	2041*

Table 5: 1% sample prediction results (Pearson r) using topics + unigrams. $*$ same counties as the 10% prediction task.

	Max Tweets	Income	Educat.	Life Satis.	Heart Disease	Num. Users Removed
County (all)	50	.73	.84	.34	.68	4,665,114
	500	.81	.87	.44	.75	611,661
	1000	.81	.87	.41	.75	217,517
	No Max	.73	.82	.31	.72	-
User to County	50	.68	.80	.34	.64	4,665,114
	500	.80	.87	.47	.76	611,661
	1000	.81	.87	.47	.76	217,517
	No Max	.81	.87	.48	.76	-

Table 6: Prediction results (Pearson r) using topics + unigrams. Users with more than "Max Tweets" number of tweets are removed from the sample.

tweets). When we compare the "User to County" and "County (all)" methods we see the "User to county" outperforming on two out of four tasks (Income and Life Satisfaction). Again, we note that the "User to County" is using less data than the "County (all)". While, across the board, the performance increase is not as substantial as in the 10% results, we see comparable performance between "User to County" and "County (all)" methods despite the difference in the number of tweets.

Super users. One theory why we see such large gains depending on aggregation technique is that aggregating through users negates the effects of super users – those who post an extraordinary amount (such as organizations or bots). We implemented a maximum tweet requirement in order to remove these users and see if that accounts for the difference. Here we use both the "User to County" and "County (all)" samples and report results in Table 6. These results demonstrate that by keeping only users with less than 500 tweets we get results close to our "User to County (No Max)" method using the user-naive "County (all)" scheme. This shows that relatively few users (in this case 611k) can significantly decrease performance, though still leaves a small gain from the user to county approach. As seen in the lower half of the table, this thresholding does not increase performance when using the "User to County" method, which suggests such users can still be beneficial if they are just treated such that they can't dominate a community. This highlights the benefit of our simple method: we do not need to consider optimizations which may not generalize across data, such as upper-bound thresholds on the number of tweets per user. Further, the user-to-county aggregation seems to provide at least a small benefit beyond removal of super users.

5 Conclusion

This study explored the benefit of aggregation techniques for streaming user-generated data from individual messages to community level data, the typical setting for nowcasting. We showed that by simply aggregating to users first and then taking the mean within a county, we can obtain large gains (remarkably, up to a 13 point increase in Pearson correlation) over typical aggregation methods common in past work. In order to foster nowcasting research utilizing this more ideal aggregation, we will release the County Tweet Lexical Bank – a large aggregated and anonymized county-level data set, and computed on more than 1.6 billion tweets posted over 5 years.

Future work in this area can look at adjusting models to account for other meta-data such as temporal variation and diversity and to adjust for selection biases present in social media, where the user base on social media is not representative of the population of the community (Greenwood et al., 2016).

Acknowledgments

This work was supported by a grant from the Templeton Religion Trust (ID #TRT0048). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

Fatemeh Almodaresi, Lyle H. Ungar, Vivek Kulkarni, Mohsen Zakeri, Salvatore Giorgi, and H. Andrew Schwartz. 2017. On the Distribution of Lexical Features in Social Media. In *Annual Meeting of the Association for Computational Linguistics*, ACL.

David Alvarez-Melis and Martin Saveski. 2016. Topic modeling in twitter: Aggregating tweets by conversations. *ICWSM*, 2016:519–522.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Glen Coppersmith, Mark Dredze, Craig Harman, and Kristy Hollingshead. 2015. From adhd to sad: Analyzing the language of mental health on twitter through self-reported diagnoses.

Aron Culotta. 2014a. Estimating county health statistics with twitter. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 1335–1344. ACM.

Aron Culotta. 2014b. Reducing Sampling Bias in Social Media Data for County Health Inference. In *Joint Statistical Meetings Proceedings*.

Brenda Curtis, Salvatore Giorgi, Anneke E. K. Buffone, Lyle H. Ungar, Robert D. Ashford, Jessie Hemmons, Dan Summers, Casey Hamilton, and H. Andrew Schwartz. 2018. Can twitter be used to predict county excessive alcohol consumption rates? *PLOS ONE*, 13(4):1–16.

Johannes C Eichstaedt, H Andrew Schwartz, Margaret L Kern, Gregory Park, Darwin R Labarthe, Raina M Merchant, Sneha Jha, Megha Agrawal, Lukasz A Dziurzynski, Maarten Sap, Christopher Weeg, Emily E Larson, Lyle H Ungar, and Martin EP Seligman. 2015. Psychological language on Twitter predicts county-level heart disease mortality. *Psychological Science*, 26:159–169.

Shannon Greenwood, Andrew Perrin, and Maeve Duggan. 2016. *Social Media Update*. Pew Research Center.

Liangjie Hong and Brian D Davison. 2010. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*, pages 80–88. ACM.

Vasileios Lampos, Danie Preoţiuc-Pietro, and Trevor Cohn. 2013. A User-Centric Model of Voting Intention from Social Media. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL.

Nicole M Lawless and Richard E Lucas. 2011. Predictors of regional well-being: A county level analysis. *Social Indicators Research*, 101(3):341–357.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics.

Renato Miranda Filho, Jussara M Almeida, and Gisele L Pappa. 2015. Twitter population sample bias and its impact on predictive outcomes: a case study on elections. In *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*, pages 1254–1261. IEEE.

Danielle Mowery, Albert Park, Mike Conway, and Craig Bryan. 2016. Towards automatically classifying depressive symptoms from twitter data for population health.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. *ICWSM*, 11(122-129):1–2.

Michael J. Paul and Mark Dredze. 2011. You are what you tweet: Analyzing twitter for public health. In *International Conference on Weblogs and Social Media (ICWSM)*, pages 265–272.

Daniel Preotiuc-Pietro, Sina Samangooei, Trevor Cohn, Nicholas Gibbins, and Mahesan Niranjan. 2012. Trendminer: An architecture for real time analysis of social media text. In *In Proceedings of the 6th International AAAI Conference on Weblogs and Social Media, Workshop on Real-Time Analysis and Mining of Social Streams, ICWSM*.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494. AUAI Press.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Richard E Lucas, Megha Agrawal, Gregory J Park, Shrinidhi K Lakshmikanth, Sneha Jha, Martin E P Seligman, and Lyle H Ungar. 2013. Characterizing geographic variation in well-being using tweets. In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*, ICWSM.

H Andrew Schwartz, Salvatore Giorgi, Maarten Sap, Patrick Crutchley, Lyle Ungar, and Johannes Eichstaedt. 2017. Dlatk: Differential language analysis toolkit. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 55–60.

Tao Wang, Markus Brede, Antonella Ianni, and Emmanouil Mentzakis. 2017. Detecting and characterizing eating-disorder communities on social media. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 91–100. ACM.

Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement

Jason Lee[*]
New York University
`jason@cs.nyu.edu`

Elman Mansimov[*]
New York University
`mansimov@cs.nyu.edu`

Kyunghyun Cho
New York University
CIFAR Azrieli Global Scholar
`kyunghyun.cho@nyu.edu`

Abstract

We propose a conditional non-autoregressive neural sequence model based on iterative refinement. The proposed model is designed based on the principles of latent variable models and denoising autoencoders, and is generally applicable to any sequence generation task. We extensively evaluate the proposed model on machine translation (En↔De and En↔Ro) and image caption generation, and observe that it significantly speeds up decoding while maintaining the generation quality comparable to the autoregressive counterpart.

1 Introduction

Conditional neural sequence modeling has become a *de facto* standard in a variety of tasks (see, e.g., Cho et al., 2015, and references therein). Much of this recent success is built on top of autoregressive sequence models in which the probability of a target sequence is factorized as a product of conditional probabilities of next symbols given all the preceding ones. Despite its success, neural autoregressive modeling has its weakness in decoding, i.e., finding the most likely sequence. Because of intractability, we must resort to suboptimal approximate decoding, and due to its sequential nature, decoding cannot be easily parallelized and results in a large latency (see, e.g., Cho, 2016). This has motivated the recent investigation into non-autoregressive neural sequence modeling by Gu et al. (2017) in the context of machine translation and Oord et al. (2017) in the context of speech synthesis.

In this paper, we propose a non-autoregressive neural sequence model based on iterative refinement, which is generally applicable to any sequence generation task beyond machine translation. The proposed model can be viewed as both a latent variable model and a conditional denoising autoencoder. We thus propose a learning algorithm that is hybrid of lowerbound maximization and reconstruction error minimization. We further design an iterative inference strategy with an adaptive number of steps to minimize the generation latency without sacrificing the generation quality.

We extensively evaluate the proposed conditional non-autoregressive sequence model and compare it against the autoregressive counterpart, using the state-of-the-art Transformer (Vaswani et al., 2017), on machine translation and image caption generation. In the case of machine translation, the proposed deterministic non-autoregressive models are able to decode approximately $2-3\times$ faster than beam search from the autoregressive counterparts on both GPU and CPU, while maintaining 90-95% of translation quality on IWSLT'16 En↔De, WMT'16 En↔Ro and WMT'14 En↔De. On image caption generation, we observe approximately $3\times$ and $5\times$ faster decoding on GPU and CPU, respectively, while maintaining 85% of caption quality.[1]

2 Non-Autoregressive Sequence Models

Sequence modeling in deep learning has largely focused on autoregressive modeling. That is, given a sequence $Y = (y_1, \ldots, y_T)$, we use some form of a neural network to parametrize the conditional distribution over each variable y_t given all the preceding variables, i.e.,

$$\log p(y_t | y_{<t}) = f_\theta(y_{<t}),$$

where f_θ is for instance a recurrent neural network. This approach has become a *de facto* standard in language modeling (Mikolov et al.,

---

[*] Equal Contribution

[1] We release the implementation, preprocessed datasets as well as trained models online at `https://github.com/nyu-dl/dl4mt-nonauto`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1173–1182
Brussels, Belgium, October 31 - November 4, 2018.

2010). When this is conditioned on an extra variable X, it becomes a conditional sequence model $\log p(Y|X)$ which serves as a basis on which many recent advances in, e.g., machine translation (Bahdanau et al., 2014; Sutskever et al., 2014; Kalchbrenner and Blunsom, 2013) and speech recognition (Chorowski et al., 2015; Chiu et al., 2017) have been made.

Despite the recent success, autoregressive sequence modeling has a weakness due to its nature of sequential processing. This weakness shows itself especially when we try to decode the most likely sequence from a trained model, i.e.,

$$\hat{Y} = \arg\max_{Y} \log p(Y|X).$$

There is no known polynomial algorithm for solving it exactly, and practitioners have relied on approximate decoding algorithms (see, e.g., Cho, 2016; Hoang et al., 2017). Among these, beam search has become the method of choice, due to its superior performance over greedy decoding, which however comes with a substantial computational overhead (Cho, 2016).

As a solution to this issue of slow decoding, two recent works have attempted non-autoregressive sequence modeling. Gu et al. (2017) have modified the Transformer (Vaswani et al., 2017) for non-autoregressive machine translation, and Oord et al. (2017) a convolutional network (Oord et al., 2016) for non-autoregressive modeling of waveform. Non-autoregressive modeling factorizes the distribution over a target sequence given a source into a product of conditionally independent per-step distributions:

$$p(Y|X) = \prod_{t=1}^{T} p(y_t|X),$$

breaking the dependency among the target variables across time. This allows us to trivially find the most likely target sequence by taking $\arg\max_{y_t} p(y_t|X)$ for each t, effectively bypassing the computational overhead and sub-optimality of decoding from an autoregressive sequence model.

This desirable property of exact and parallel decoding however comes at the expense of potential performance degradation (Kaiser and Bengio, 2016). The potential modeling gap, which is the gap between the underlying, true model and the neural sequence model, could be larger with the non-autogressive model compared to the autoregressive one due to challenge of modeling the factorized conditional distribution above.

3 Iterative Refinement for Deterministic Non-Autoregressive Sequence Models

3.1 Latent variable model

Similarly to two recent works (Oord et al., 2017; Gu et al., 2017), we introduce latent variables to implicitly capture the dependencies among target variables. We however remove any stochastic behavior by interpreting this latent variable model, introduced immediately below, as a process of iterative refinement.

Our goal is to capture the dependencies among target symbols given a source sentence without auto-regression by introducing L intermediate random variables and marginalizing them out:

$$p(Y|X) = \sum_{Y^0,\ldots,Y^L} \left(\prod_{t=1}^{T} p(y_t|Y^L, X) \right) \qquad (1)$$
$$\left(\prod_{t=1}^{T} p(y_t^L|Y^{L-1}, X) \right) \cdots \left(\prod_{t=1}^{T} p(y_t^0|X) \right).$$

Each product term inside the summation is modelled by a deep neural network that takes as input a source sentence and outputs the conditional distribution over the target vocabulary V for each t.

Deterministic Approximation The marginalization in Eq. (1) is intractable. In order to avoid this issue, we consider two approximation strategies; deterministic and stochastic approximation. Without loss of generality, let us consider the case of single intermediate latent variable, that is $L = 1$. In the *deterministic* case, we set $\hat{y}_t^0$ to the most likely value according to its distribution $p(y_t^0|X)$, that is $\hat{y}_t^0 = \arg\max_{y_t^0} p(y_t^0|X)$. The entire lower bound can then be written as:

$$\log p(Y|X) \geq \left(\sum_{t=1}^{T} \log p(y_t|\hat{Y}^L, X) \right) + \cdots$$
$$+ \left(\sum_{t=1}^{T} \log p(y_t^1|\hat{Y}^0, X) \right) + \left(\sum_{t=1}^{T} \log p(\hat{y}_t^0|X) \right).$$

Stochastic Approximation In the case of *stochastic* approximation, we instead sample $\hat{y}_t^0$ from the distribution $p(y_t^0|X)$. This results in the unbiased estimate of the marginal log-probability $\log p(Y|X)$. Other than the difference in whether most likely values or samples are used, the remaining steps are identical.

Latent Variables Although the intermediate random variables could be anonymous, we constrain them to be of the same type as the output Y is, in order to share an underlying neural network. This constraint allows us to view each conditional $p(Y^l|\hat{Y}^{l-1}, X)$ as a single-step of refinement of a rough target sequence $\hat{Y}^{l-1}$. The entire chain of L conditionals is then the L-step iterative refinement. Furthermore, sharing the parameters across these refinement steps enables us to dynamically adapt the number of iterations per input X. This is important as it substantially reduces the amount of time required for decoding, as we see later in the experiments.

Training For each training pair (X, Y^*), we first approximate the marginal log-probability. We then minimize

$$J_{\text{LVM}}(\theta) = -\sum_{l=0}^{L+1} \left(\sum_{t=1}^{T} \log p_\theta(y_t^*|\hat{Y}^{l-1}, X) \right), \tag{2}$$

where $\hat{Y}^{l-1} = (\hat{y}_1^{l-1}, \ldots, \hat{y}_T^{l-1})$, and θ is a set of parameters. We initialize $\hat{y}_t^0$ (t-th target word in the first iteration) as $x_{t'}$, where $t' = (T'/T) \cdot t$. T' and T are the lengths of the source X and target Y^*, respectively.

3.2 Denoising Autoencoder

The proposed approach could instead be viewed as learning a conditional denoising autoencoder which is known to capture the gradient of the log-density. That is, we implicitly learn to find a direction Δ_Y in the output space that maximizes the underlying true, data-generating distribution $\log P(Y|X)$. Because the output space is discrete, much of the theoretical analysis by Alain and Bengio (2014) are not strictly applicable. We however find this view attractive as it serves as an alternative foundation for designing a learning algorithm.

Training We start with a corruption process $C(Y|Y^*)$, which introduces noise to the correct output Y^*. Given the reference translation Y^*, we sample $\tilde{Y} \sim C(Y|Y^*)$ which becomes as an input to each conditional in Eq. (1). Then, the goal of learning is to maximize the log-probability of the original reference Y^* given the corrupted version. That is, to minimize

$$J_{\text{DAE}}(\theta) = -\sum_{t=1}^{T} \log p_\theta(y_t^*|\tilde{Y}, X). \tag{3}$$

Once this cost J_{DAE} is minimized, we can recursively perform the maximum-a-posterior inference, i.e., $\hat{Y} = \arg\max_Y \log p_\theta(Y|X)$, to find $\hat{Y}$ that (approximately) maximizes $\log p(Y|X)$.

Corruption Process C There is little consensus on the best corruption process for a sequence, especially of discrete tokens. In this work, we use a corruption process proposed by Hill et al. (2016), which has recently become more widely adopted (see, e.g., Artetxe et al., 2017; Lample et al., 2017). Each y_t^* in a reference target $Y^* = (y_1^*, \ldots, y_T^*)$ is corrupted with a probability $\beta \in [0, 1]$. If decided to corrupt, we either (1) replace y_{t+1}^* with this token y_t^*, (2) replace y_t^* with a token uniformly selected from a vocabulary of all unique tokens at random, or (3) swap y_t^* and y_{t+1}^*. This is done sequentially from y_1^* until y_T^*.

3.3 Learning

Cost function Although it is possible to train the proposed non-autoregressive sequence model using either of the cost functions above (J_{LVM} or J_{DAE},) we propose to stochastically mix these two cost functions. We do so by randomly replacing each term $\hat{Y}^{l-1}$ in Eq. (2) with $\tilde{Y}$ in Eq. (3):

$$J(\theta) = -\sum_{l=0}^{L+1} \left(\alpha_l \sum_{t=1}^{T} \log p_\theta(y_t^*|\hat{Y}^{l-1}, X) \right. \tag{4}$$
$$\left. +(1-\alpha_l) \sum_{t=1}^{T} \log p_\theta(y_t^*|\tilde{Y}, X) \right),$$

where $\tilde{Y} \sim C(Y|Y^*)$, and α_l is a sample from a Bernoulli distribution with the probability p_{DAE}. p_{DAE} is a hyperparameter. As the first conditional $p(Y^0|X)$ in Eq. (1) does not take as input any target Y, we set $\alpha_0 = 1$ always.

Distillation Gu et al. (2017), in the context of machine translation, and Oord et al. (2017), in the context of speech generation, have recently discovered that it is important to use knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016) to successfully train a non-autoregressive sequence model. Following Gu et al. (2017), we also use knowledge distillation by replacing the reference target Y^* of each training example

(X, Y^*) with a target Y^{AR} generated from a well-trained autoregressive counterpart. Other than this replacement, the cost function in Eq (4) and the model architecture remain unchanged.

Target Length Prediction One difference between the autoregressive and non-autoregressive models is that the former naturally models the length of a target sequence without any arbitrary upper-bound, while the latter does not. It is hence necessary to separately model $p(T|X)$, where T is the length of a target sequence, although during training, we simply use the length of each reference target sequence.

3.4 Inference: Decoding

Inference in the proposed approach is entirely deterministic. We start from the input X and first predict the length of the target sequence $\hat{T} = \arg\max_T \log p(T|X)$. Then, given X and $\hat{T}$ we generate the initial target sequence by $\hat{y}_t^0 = \arg\max_{y_t} \log p(y_t^0|X)$, for $t = 1, \ldots, T$ We continue refining the target sequence by $\hat{y}_t^l = \arg\max_{y_t} \log p(y_t^l|\hat{Y}^{l-1}, X)$, for $t = 1, \ldots, T$.

Because these conditionals, except for the initial one, are modeled by a single, shared neural network, this refinement can be performed as many iterations as necessary until a predefined stopping criterion is met. A criterion can be based either on the amount of change in a target sequence after each iteration (i.e., $D(\hat{Y}^{l-1}, \hat{Y}^l) \leq \epsilon$), or on the amount of change in the conditional log-probabilities (i.e., $|\log p(\hat{Y}^{l-1}|X) - \log p(\hat{Y}^{l-1}|X)| \leq \epsilon$) or on the computational budget. In our experiments, we use the first criterion and use Jaccard distance as our distance function D.

4 Related Work

Non-Autoregressive Neural Machine Translation Schwenk (2012) proposed a continuous-space translation model to estimate the conditional distribution over a target phrase given a source phrase, while dropping the conditional dependencies among target tokens. The evaluation was however limited to reranking and to short phrase pairs (up to 7 words on each side) only. Kaiser and Bengio (2016) investigated neural GPU (Kaiser and Sutskever, 2015), for machine translation. They evaluated both non-autoregressive and autoregressive approaches, and found that the non-autoregressive approach significantly lags behind the autoregressive variants. It however differs from our approach that each iteration does not output a refined version from the previous iteration. The recent paper by Gu et al. (2017) is most relevant to the proposed work. They similarly introduced a sequence of discrete latent variables. They however use supervised learning for inference, using the word alignment tool (Dyer et al., 2013). To achieve the best result, Gu et al. (2017) stochastically sample the latent variables and rerank the corresponding target sequences with an external, autoregressive model. This is in contrast to the proposed approach which is fully deterministic during decoding and does not rely on any extra reranking mechanism.

Parallel WaveNet Simultaneously with Gu et al. (2017), Oord et al. (2017) presented a non-autoregressive sequence model for speech generation. They use inverse autoregressive flow (IAF, Kingma et al., 2016) to map a sequence of independent random variables to a target sequence. They apply the IAF multiple times, similarly to our iterative refinement strategy. Their approach is however restricted to continuous target variables, while the proposed approach in principle could be applied to both discrete and continuous variables.

Post-Editing for Machine Translation Novak et al. (2016) proposed a convolutional neural network that iteratively predicts and applies token substitutions given a translation from a phase-based translation system. Unlike their system, our approach can edit an intermediate translation with a higher degree of freedom. QuickEdit (Grangier and Auli, 2017) and deliberation network (Xia et al., 2017) incorporate the idea of refinement into neural machine translation. Both systems consist of two autoregressive decoders. The second decoder takes into account the translation generated by the first decoder. We extend these earlier efforts by incorporating more than one refinement steps without necessitating extra annotations.

Infusion Training Bordes et al. (2017) proposed an unconditional generative model for images based on iterative refinement. At each step l of iterative refinement, the model is trained to maximize the log-likelihood of target Y given the weighted mixture of generated samples from the previous iteration $\hat{Y}^{l-1}$ and a corrupted target $\tilde{Y}$. That is, the corrupted version of target is "infused"

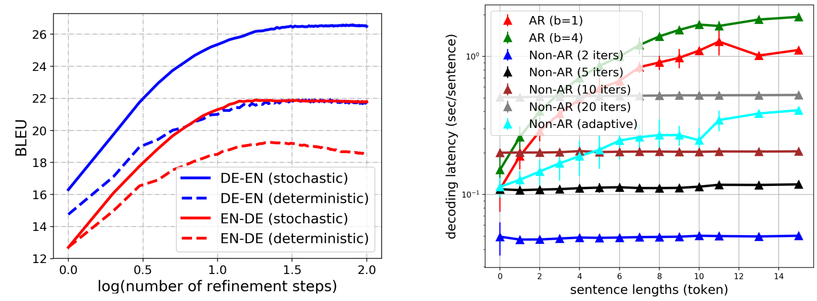

(a)

(b)

Figure 1: (a) BLEU scores on WMT'14 En-De w.r.t. the number of refinement steps (up to 10^2). The x-axis is in the logarithmic scale. (b) the decoding latencies (sec/sentence) of different approaches on IWSLT'16 En→De. The y-axis is in the logarithmic scale.

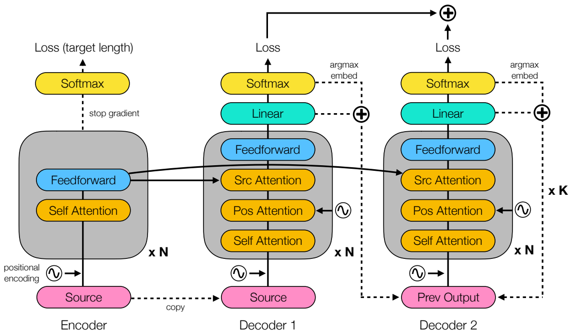

Figure 2: We compose three transformer blocks ("Encoder", "Decoder 1" and "Decoder 2") to implement the proposed non-autoregressive sequence model.

into generated samples during training. In the domain of text, however, computing a weighted mixture of two sequences of discrete tokens is not well defined, and we propose to stochastically mix denoising and lowerbound maximization objectives.

5 Network Architecture

We use three transformer-based network blocks to implement our model. The first block ("Encoder") encodes the input X, the second block ("Decoder 1") models the first conditional $\log p(Y^0|X)$, and the final block ("Decoder 2") is shared across iterative refinement steps, modeling $\log p(Y^l|\hat{Y}^{l-1}, X)$. These blocks are depicted side-by-side in Fig. 2. The encoder is identical to that from the original Transformer (Vaswani et al., 2017). We however use the decoders from Gu et al. (2017) with additional positional attention and use the highway layer (Srivastava et al., 2015) instead of the residual layer (He et al., 2016).

The original input X is padded or shortned to fit the length of the reference target sequence before being fed to Decoder 1. At each refinement step l, Decoder 2 takes as input the predicted target sequence $\hat{Y}^{l-1}$ and the sequence of final activation vectors from the previous step.

6 Experimental Setting

We evaluate the proposed approach on two sequence modeling tasks: machine translation and image caption generation. We compare the proposed non-autoregressive model against the autoregressive counterpart both in terms of generation quality, measured in terms of BLEU (Papineni et al., 2002), and generation efficiency, measured in terms of (source) tokens and images per second for translation and image captioning, respectively.

Machine Translation We choose three tasks of different sizes: IWSLT'16 En↔De (196k pairs), WMT'16 En↔Ro (610k pairs) and WMT'14 En↔De (4.5M pairs). We tokenize each sentence using a script from Moses (Koehn et al., 2007) and segment each word into subword units using BPE (Sennrich et al., 2016). We use 40k tokens from both source and target for all the tasks. For WMT'14 En-De, we use newstest-2013 and newstest-2014 as development and test sets. For WMT'16 En-Ro, we use newsdev-2016 and newstest-2016 as development and test sets. For IWSLT'16 En-De, we use test2013 for validation.

We closely follow the setting by Gu et al. (2017). In the case of IWSLT'16 En-De, we use the small model ($d_{\text{model}} = 278, d_{\text{hidden}} = 507, p_{\text{dropout}} = 0.1, n_{\text{layer}} = 5$ and $n_{\text{head}} = 2$).[2] For WMT'14 En-De and WMT'16 En-Ro, we use the base transformer by Vaswani et al. (2017) ($d_{\text{model}} = 512, d_{\text{hidden}} = 512, p_{\text{dropout}} = 0.1, n_{\text{layer}} = 6$ and $n_{\text{head}} = 8$). We use the warm-up learning rate scheduling (Vaswani et al., 2017) for the WMT tasks, while using linear annealing (from 3×10^{-4} to 10^{-5}) for the IWSLT task. We do not use label smoothing nor average multiple check-pointed models. These decisions were made based on the preliminary experiments. We train each model either on a single

---

[2] Due to the space constraint, we refer readers to (Vaswani et al., 2017; Gu et al., 2017) for more details.

		IWSLT'16 En-De				WMT'16 En-Ro				WMT'14 En-De				MS COCO		
		En→	De→	GPU	CPU	En→	Ro→	GPU	CPU	En→	De→	GPU	CPU	BLEU	GPU	CPU
AR	$b = 1$	28.64	34.11	70.3	32.2	31.93	31.55	55.6	15.7	23.77	28.15	54.0	15.8	23.47	4.3	2.1
	$b = 4$	28.98	34.81	63.8	14.6	32.40	32.06	43.3	7.3	24.57	28.47	44.9	7.0	24.78	3.6	1.0
NAT	FT	26.52	–	–	–	27.29	29.06	–	–	17.69	21.47	–	–	–	–	–
	FT+NPD	28.16	–	–	–	29.79	31.44	–	–	19.17	23.30	–	–	–	–	–
Our Model	$i_{\text{dec}} = 1$	22.20	27.68	573.0	213.2	24.45	25.73	694.2	98.6	13.91	16.77	511.4	83.3	20.12	17.1	8.9
	$i_{\text{dec}} = 2$	24.82	30.23	423.8	110.9	27.10	28.15	332.7	62.8	16.95	20.39	393.6	49.6	20.88	12.0	5.7
	$i_{\text{dec}} = 5$	26.58	31.85	189.7	52.8	28.86	29.72	194.4	29.0	20.26	23.86	139.7	23.1	21.12	6.2	2.8
	$i_{\text{dec}} = 10$	27.11	32.31	98.8	24.1	29.32	30.19	93.1	14.8	21.61	25.48	90.4	12.3	21.24	2.0	1.2
	Adaptive	27.01	32.43	125.9	29.3	29.66	30.30	118.3	16.5	21.54	25.43	107.2	20.3	21.12	10.8	4.8

Table 1: Generation quality (BLEU↑) and decoding efficiency (tokens/sec↑ for translation, images/sec↑ for image captioning). Decoding efficiency is measured sentence-by-sentence. AR: autoregressive models. b: beam width. i_{dec}: the number of refinement steps taken during decoding. Adaptive: the adaptive number of refinement steps. NAT: non-autoregressive transformer models (Gu et al., 2017). FT: fertility. NPD reranking using 100 samples.

P40 (WMT'14 En-De and WMT'16 En-Ro) or on a single P100 (IWSLT'16 En-De) with each minibatch consisting of approximately 2k tokens. We use four P100's to train non-autoregressive models on WMT'14 En-De.

Image Caption Generation: MS COCO We use MS COCO (Lin et al., 2014). We use the publicly available splits (Karpathy and Li, 2015), consisting of 113,287 training images, 5k validation images and 5k test images. We extract 49 512-dimensional feature vectors for each image, using a ResNet-18 (He et al., 2016) pretrained on ImageNet (Deng et al., 2009). The average of these vectors is copied as many times to match the length of the target sentence (reference during training and predicted during evaluation) to form the initial input to Decoder 1. We use the base transformer (Vaswani et al., 2017) except that n_{layer} is set to 4. We train each model on a single 1080ti with each minibatch consisting of approximately 1,024 tokens.

Target Length Prediction We formulate the target length prediction as classification, predicting the difference between the target and source lengths for translation and the target length for image captioning. All the hidden vectors from the n_{layer} layers of the encoder are summed and fed to a softmax classifier after affine transformation. We however do not tune the encoder's parameters for target length prediction. We use this length predictor only during test time. We find it important to accurately predict the target length for good overall performance. See Appendix A for an analysis on our length prediction model.

Training and Inference We use Adam (Kingma and Ba, 2014) and use $L = 3$ in Eq. (1) during training ($i_{\text{train}} = 4$ from hereon.) We use $p_{\text{DAE}} = 0.5$. We use the deterministic strategy for IWSLT'16 En-De, WMT'16 En-Ro and MS COCO, while the stochastic strategy is used for WMT'14 En-De. These decisions were made based on the validation set performance. After both the non-autogressive sequence model and target length predictor are trained, we decode by first predicting the target length and then running iterative refinement steps until the outputs of consecutive iterations are the same (or Jaccard distance between consecutive decoded sequences is 1). To assess the effectiveness of this adaptive scheme, we also test a fixed number of steps (i_{dec}). In machine translation, we remove any repetition by collapsing multiple consecutive occurrences of a token.

7 Results and Analysis

We make some important observations in Table 1. First, the generation quality improves across all the tasks as we run more refinement steps i_{dec} even beyond that used in training ($i_{\text{train}} = 4$), which supports our interpretation as a conditional denoising autoencoder in Sec. 3.2. To further verify this, we run decoding on WMT'14 (both directions) up to 100 iterations. As shown in Fig. 1 (a), the quality improves well beyond the number of refinement steps used during training.

Second, the generation efficiency decreases as more refinements are made. We plot the average seconds per sentence in Fig. 1 (b), measured on GPU while sequentially decoding one sentence at a time. As expected, decoding from the autoregressive model linearly slows down as the sen-

	i_{train}	p_{DAE}	distill	En→De rep	En→De no rep	De→En rep	De→En no rep
AR	$b=1$			28.64		34.11	
	$b=4$			28.98		34.81	
Our Models	1	0		14.62	18.03	16.70	21.18
	2	0		17.42	21.08	19.84	24.25
	4	0		19.22	22.65	22.15	25.24
	4	1		19.83	22.29	24.00	26.57
	4	0.5		20.91	23.65	24.05	28.18
	4	0.5	√	26.17	27.11	31.92	32.59

Table 2: Ablation study on the dev set of IWSLT'16.

tence length grows, while decoding from the non-autoregressive model with a fixed number of iterations has the constant complexity. However, the generation efficiency of non-autoregressive model decreases as more refinements are made. To make a smooth trade-off between the quality and speed, the adaptive decoding scheme allows us to achieve near-best generation quality with a significantly lower computational overhead. Moreover, the adaptive decoding scheme automatically increases the number of refinement steps as the sentence length increases, suggesting that this scheme captures the amount of information in the input well. The increase in latency is however less severe than that of the autoregressive model.

We also observe that the speedup in decoding is much clearer on GPU than on CPU. This is a consequence of highly parallel computation of the proposed non-autoregressive model, which is better suited to GPUs, showcasing the potential of using the non-autoregressive model with a specialized hardware for parallel computation, such as Google's TPUs (Jouppi et al., 2017). The results of our model decoded with adaptive decoding scheme are comparable to the results from (Gu et al., 2017), without relying on any external tool. On WMT'14 En-De, the proposed model outperforms the best model from (Gu et al., 2017) by two points.

Lastly, it is encouraging to observe that the proposed non-autoregressive model works well on image caption generation. This result confirms the generality of our approach beyond machine translation, unlike that by Gu et al. (2017) which was for machine translation or by Oord et al. (2017) which was for speech synthesis.

stochastic	distill	IWSLT'16 (En→)	WMT'14 (En→)
		23.65	7.56
√		22.80	16.56
	√	27.11	18.91
√	√	25.39	21.22

Table 3: Deterministic and stochastic approximation

Ablation Study We use IWSLT'16 En-De to investigate the impact of different number of refinement steps during training (denoted as i_{train}) as well as probability of using denoising autoencoder objective during training (denoted as p_{DAE}). The results are presented in Table 2.

First, we observe that it is beneficial to use multiple iterations of refinement during training. By using four iterations (one step of decoder 1, followed by three steps of decoder 2), the BLEU score improved by approximately 1.5 points in both directions. We also notice that it is necessary to use the proposed hybrid learning strategy to maximize the improvement from more iterations during training ($i_{\text{train}} = 4$ vs. $i_{\text{train}} = 4, p_{\text{DAE}} = 1.0$ vs. $i_{\text{train}} = 4, p_{\text{DAE}} = 0.5$.) Knowledge distillation was crucial to close the gap between the proposed deterministic non-autoregressive sequence model and its autoregressive counterpart, echoing the observations by Gu et al. (2017) and Oord et al. (2017). Finally, we see that removing repeating consecutive symbols improves the quality of the best trained models ($i_{\text{train}} = 4, p_{\text{DAE}} = 0.5$) by approximately +1 BLEU. This suggests that the proposed iterative refinement is not enough to remove repetitions on its own. Further investigation is necessary to properly tackle this issue, which we leave as a future work.

We then compare the deterministic and stochastic approximation strategies on IWSLT'16 En→De and WMT'14 En→De. According to the results in Table 3, the stochastic strategy is crucial with a large corpus (WMT'14), while the deterministic strategy works as well or better with a small corpus (IWSLT'16). Both of the strategies benefit from knowledge distillation, but the gap between the two strategies when the dataset is large is much more apparent without knowledge distillation.

7.1 Qualitative Analysis

Machine Translation In Table 4, we present three sample translations and their iterative refinement steps from the development set of IWSLT'16 (De→En). As expected, the sequence generated from the first iteration is a rough version of translation and is iteratively refined over multiple steps. By inspecting the underlined sub-sequences, we see that each iteration does not monotonically improve the translation, but overall modifies the

Src	seitdem habe ich sieben Häuser in der Nachbarschaft mit den Lichtern versorgt und sie funktionierenen wirklich gut .
Iter 1	and I 've been seven homes since in neighborhood with the lights and they 're really functional .
Iter 2	and I 've been seven homes in the neighborhood with the lights , and they 're a really functional .
Iter 4	and I 've been seven homes in neighborhood with the lights , and they 're a really functional .
Iter 8	and I 've been providing seven homes in the neighborhood with the lights and they 're a really functional .
Iter 20	and I 've been providing seven homes in the neighborhood with the lights , and they 're a very good functional .
Ref	since now , I 've set up seven homes around my community , and they 're really working .

Src	er sah sehr glücklich aus , was damals ziemlich ungewöhnlich war , da ihn die Nachrichten meistens deprimierten .
Iter 1	he looked very happy , which was pretty unusual the , because the news was were usually depressing .
Iter 2	he looked very happy , which was pretty unusual at the , because the news was s depressing .
Iter 4	he looked very happy , which was pretty unusual at the , because news was mostly depressing .
Iter 8	he looked very happy , which was pretty unusual at the time because the news was mostly depressing .
Iter 20	he looked very happy , which was pretty unusual at the time , because the news was mostly depressing .
Ref	there was a big smile on his face which was unusual then , because the news mostly depressed him .

Src	furchtlos zu sein heißt für mich , heute ehrlich zu sein .
Iter 1	to be , for me , to be honest today .
Iter 2	to be fearless , me , is to be honest today .
Iter 4	to be fearless for me , is to be honest today .
Iter 8	to be fearless for me , me to be honest today .
Iter 20	to be fearless for me , is to be honest today .
Ref	so today , for me , being fearless means being honest .

Table 4: Three sample De→En translations from the non-autoregressive sequence model. Source sentences are from the dev set of IWSLT'16. The first iteration corresponds to Decoder 1, and from thereon, Decoder 2 is repeatedly applied. Sub-sequences with changes across the refinement steps are underlined.

translation towards the reference sentence. Missing words are added, while unnecessary words are dropped. For instance, see the second example. The second iteration removes the unnecessary "were", and the fourth iteration inserts a new word "mostly". The phrase "at the time" is gradually added one word at a time.

Image Caption Generation Table 5 shows two examples of image caption generation. We observe that each iteration captures more and more details of the input image. In the first example (left), the bus was described only as a "yellow bus" in the first iteration, but the subsequent iterations refine it into "yellow and black bus". Similarly, "road" is refined into "lot". We notice this behavior in the second example (right) as well. The first iteration does not specify the place in which "a woman" is "standing on", which is fixed immediately in the second iteration: "standing on a tennis court". In the final and fourth iteration, the proposed model captures the fact that the "woman" is "holding" a racquet.

8 Conclusion

Following on the exciting, recent success of non-autoregressive neural sequence modeling by Gu et al. (2017) and Oord et al. (2017), we proposed a deterministic non-autoregressive neural sequence model based on the idea of iterative refinement. We designed a learning algorithm specialized to the proposed approach by interpreting the entire model as a latent variable model and each refinement step as denoising.

We implemented our approach using the Transformer and evaluated it on two tasks: machine translation and image caption generation. On both tasks, we were able to show that the proposed non-autoregressive model performs closely to the autoregressive counterpart with significant speedup in decoding. Qualitative analysis revealed that the iterative refinement indeed refines a target sequence gradually over multiple steps.

Despite these promising results, we observed that proposed non-autoregressive neural sequence model is outperformed by its autoregressive counterpart in terms of the generation quality. The following directions should be pursued in the future to narrow this gap. First, we should investigate better approximation to the marginal log-probability. Second, the impact of the corruption process on the generation quality must be studied. Lastly, further work on sequence-to-sequence model architectures could yield better results in non-autoregressive sequence modeling.

Acknowledgement

We thank support by AdeptMind, eBay, TenCent and NVIDIA. This work was partly supported by Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI) and Samsung Electronics (Improving Deep Learning using Latent Structure). We also thank Jiatao Gu for valuable feedback.

Generated Caption	
Iter 1	a yellow bus parked on parked in of parking road .
Iter 2	a yellow and black on parked in a parking lot .
Iter 3	a yellow and black bus parked in a parking lot .
Iter 4	a yellow and black bus parked in a parking lot .
Reference Captions	
a tour bus is parked on the curb waiting city bus parked on side of hotel in the rain . bus parked under an awning next to brick sidewalk a bus is parked on the curb in front of a building . a double decked bus sits parked under an awning	

Generated Caption	
Iter 1	a woman standing on playing tennis on a tennis racquet .
Iter 2	a woman standing on a tennis court a tennis racquet .
Iter 3	a woman standing on a tennis court a a racquet .
Iter 4	a woman standing on a tennis court holding a racquet .
Reference Captions	
a female tennis player in a black top playing tennis a woman standing on a tennis court holding a racquet . a female tennis player preparing to serve the ball . a woman is holding a tennis racket on a court a woman getting ready to reach for a tennis ball on the ground	

Table 5: Two sample image captions from the proposed non-autoregressive sequence model. The images are from the development set of MS COCO. The first iteration is from decoder 1, while the subsequent ones are from decoder 2. Subsequences with changes across the refinement steps are underlined.

References

Guillaume Alain and Yoshua Bengio. 2014. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1).

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Florian Bordes, Sina Honari, and Pascal Vincent. 2017. Learning to generate samples from noise through infusion training. In *ICLR*.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Katya Gonina, et al. 2017. State-of-the-art speech recognition with sequence-to-sequence models. *arXiv preprint arXiv:1712.01769*.

Kyunghyun Cho. 2016. Noisy parallel approximate decoding for conditional recurrent language model. *arXiv preprint arXiv:1605.03835*.

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11).

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *NIPS*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *ACL*.

David Grangier and Michael Auli. 2017. Quickedit: Editing text & translations via simple delete actions. *arXiv preprint arXiv:1711.04805*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *NAACL*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Cong Duy Vu Hoang, Gholamreza Haffari, and Trevor Cohn. 2017. Decoding as continuous optimization in neural machine translation. *arXiv preprint arXiv:1701.02854*.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*.

Łukasz Kaiser and Samy Bengio. 2016. Can active memory replace attention? In *NIPS*.

Łukasz Kaiser and Ilya Sutskever. 2015. Neural GPUs learn algorithms. *arXiv preprint arXiv:1511.08228*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *NIPS*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. 2014. Microsoft COCO: Common objects in context. In *ECCV*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Roman Novak, Michael Auli, and David Grangier. 2016. Iterative refinement for machine translation. *arXiv preprint arXiv:1610.06602*.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. 2017. Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. *Proceedings of COLING 2012: Posters*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *NIPS*.

Large Margin Neural Language Model

Jiaji Huang[1] **Yi Li**[1] **Wei Ping**[1] **Liang Huang**[1,2*]
[1] Baidu Research, Sunnyvale, CA, USA
[2] School of EECS, Oregon State University, Corvallis, OR, USA
{huangjiaji, liyi17, pingwei01, lianghuang}@baidu.com

Abstract

We propose a large margin criterion for training neural language models. Conventionally, neural language models are trained by minimizing perplexity (PPL) on grammatical sentences. However, we demonstrate that PPL may not be the best metric to optimize in some tasks, and further propose a large margin formulation. The proposed method aims to enlarge the *margin* between the "good" and "bad" sentences in a task-specific sense. It is trained end-to-end and can be widely applied to tasks that involve re-scoring of generated text. Compared with minimum-PPL training, our method gains up to 1.1 WER reduction for speech recognition and 1.0 BLEU increase for machine translation.

1 Introduction

Language models (LMs) estimate the likelihood of a symbol sequence $\{x^t\}_{t=0}^T$, based on the joint probability,

$$p(x^0, \ldots, x^T) = p(x^0) \prod_{t=1}^{T} p(x^t | x^0, \ldots, x^{t-1}). \tag{1}$$

To measure the quality of an LM, a commonly adopted metric is perplexity (PPL), defined as

$$\text{PPL} \triangleq \exp\left\{ -\frac{1}{T} \sum_{t=0}^{T} \log p(x^t | x^0, \ldots, x^{t-1}) \right\},$$

A good language model has a small PPL, being able to assign higher likelihoods to sentences that are more likely to appear.

LMs are widely applied in automatic speech recognition (ASR) (Yu and Deng, 2014) and machine translation (MT) (Koehn, 2009). Following Koehn (2009), one may interpret the language model as prior knowledge on the text to be inferred, which provides information complementary to the ASR or MT system itself. In practice, there are several ways to incorporate the language model. The simplest way may be re-scoring an n-best list returned by the ASR or MT system (Mikolov et al., 2010; Sundermeyer et al., 2012). A slightly more sophisticated way is to jointly consider the ASR/MT and language model in a beam search decoder (Amodei et al., 2016). Specifically, at each time step, the decoder appends every symbol in the vocabulary to each sequence in the current candidate set. For every hypothesis, a score is calculated as a linear combination of the log-likelihoods given by both the ASR/MT and language models. Then, only the top K hypotheses with the highest scores are retained, as an updated candidate set. More recently, Gulcehre et al. (2015) and Sriram et al. (2017) propose to predict the next symbol based on a fusion of the hidden states in the ASR/MT and language models. A gating mechanism is jointly trained to determine how much the language model should contribute.

The afore-discussed language models are generative in the sense that they merely model the joint distribution of a symbol sequence (Eq. (1)). While the research community is mostly focused on pushing the limit of PPL (e.g., Jozefowicz et al., 2016), very limited attention has been paid to the discrimination power of language models when they are applied to real tasks, such as ASR and MT (Li and Khudanpur, 2008). By contrast, discriminative language modeling aims at enhancing the performance in downstream applications. For example, existing works (Roark et al., 2004, 2007) often target at improving ASR accuracy. The key motivation underlying them is that the model should be able to discriminate between "good" and "bad" sentences in a task-specific sense, instead

---

[*] Contributions were made while at Baidu Research.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1183–1191
Brussels, Belgium, October 31 - November 4, 2018.

of just modeling grammatical ones. The common methodology (Dikic et al., 2013) is to build a binary classifier upon hand-crafted features extracted from the sentences. However, it is not obvious how these methods can utilize large unannotated corpus, which is often easily available, and the hand-crafted features are also ad hoc and may result in suboptimal performance.

In this work, we study how to improve the discrimination ability of a recurrent network-based neural language model (RNNLM). The goal is to enlarge the difference between the log-likelihoods of "good" and "bad" sentences. In an contrast to the existing works (Roark et al., 2004, 2007), our method does not rely on hand-crafted features, and is trained in end-to-end manner and able to take advantage of large external text corpus. In fact, it is a general training criterion that is transparent to the network architecture of the RNNLM, and can be applied to various text generation tasks, including ASR and MT. Experiments on state-of-art ASR and MT systems show its significant advantage over an LM trained by minimizing PPL.

2 Background on RNNLM

We first give some background knowledge on RNNLMs. The prototypical RNNLM (Mikolov et al., 2010) has one layer of recurrent cell and works as follows. Denote a sentence as $\mathbf{x} = [x^0, \ldots, x^t, \ldots]$, where the x^t's are words. Let $\vec{x}^t$ be the embedding vector for x^t. The recurrent cell takes in the embedding and produces a hidden state $\vec{h}^t$ by

$$\vec{h}^t = \sigma(U\vec{x}^t + V\vec{h}^{t-1}),$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is sigmoid activation function. $\vec{h}^{t-1}$ is the hidden state at the last timestep. U and V are learnable parameters. The $\vec{h}^t$ is then passed into a multi-way classifier to produce a probability distribution over the vocabulary (for the next word),

$$\vec{p} = \mathrm{softmax}(W\vec{h}^t + \vec{b}).$$

The W and $\vec{b}$ are also trainable parameters. The training objective is to maximize the log-likelihood of the next word, and the parameters are learned by back-propagation algorithm.

The vanilla recurrent cell can also be replaced by one or multiple layers of LSTM cells, which produces better results (Zaremba et al., 2014). In a more general form, the RNNLM can be represented as a conditional probability, $p_\theta(x^t|x^0, \ldots, x^{t-1})$, parameterized by θ. In the prototypical case, $\theta = [U, V, W, \vec{b}]$. We could define the *LM-score* of a sentence $\mathbf{x}$ as

$$\begin{aligned}\text{LM-score}(\mathbf{x}) &\triangleq \log p_\theta(\mathbf{x}) \\ &= \sum_t \log p_\theta(x^t|x^0, \ldots, x^{t-1}).\end{aligned}$$

The RNNLM is trained by maximizing the average LM-score over all the $\mathbf{x}$'s in a corpus, or equivalently, minimizing the PPL on the corpus.

3 Problem Formulation

We motivate and formulate a large margin training criterion in this section. Suppose for every reference sentence $\mathbf{x}_i$, we have a collection of hypotheses $\mathbf{x}_{i,j}, j = 1, \ldots, K$, usually obtained as the top-K candidates by a beam search decoder.

3.1 A Motivating Example

An RNNLM trained by minimizing PPL cannot guarantee a higher score on the "gold" reference than the inferior hypothesis, which is undesirable. One example is given in Tab. 1. The reference is taken from the text labels of dev93' set of Wall Street Journal (WSJ) dataset. The hypothesis is generated by a CTC-based (Graves et al., 2006) ASR system trained on WSJ training set. Words in red are mistakes made by the hypothesis. We then train an RNNLM on Common Crawl[1] copora by minimizing PPL. Training follows a typical setup (Jozefowicz et al., 2016) with a vocabulary of 400K the most frequent words. Any out-of-vocabulary word is replaced by an ⟨UNK⟩ token. The RNNLM is then employed to score the sentences. The LM-score of the erroneous hypothesis is higher than that of the reference. In fact, this is reasonable as "a decade as concerns" seems to be a more common phrase. In the training corpus, we find that "a decade as concerns" appears once, but "its defeat is confirmed" does not appear. Moreover, "a decade as" appears 2,280 times, but "its defeat is" appears only 24 times. However, this is undesirable because if there is another hypothesis that happens to be the same as reference, which will not be ranked as the best candidate.

It would be helpful if the LM can also learn from the imperfect hypotheses so that it can tell

---

[1] `http://web-language-models.s3-website-us-east-1.amazonaws.com/wmt16/deduped/en-new.xz`

	Sentence	LM-score
reference	coniston declined to discuss its plans for its defeat is confirmed but indicated that it doesn't plan to simply walk away	-116.52
hypothesis	coniston declined to discuss its plans for a decade as concerns but indicated that it doesn't plan to simply walk away	-112.65

Table 1: Reference and one hypothesis, scored by an RNNLM. Words in red are mistakes in the hypothesis. The RNNLM is trained on Common Crawl copora by minimizing PPL. We want the reference to be higher scored than the hypothesis, but it does not happen here.

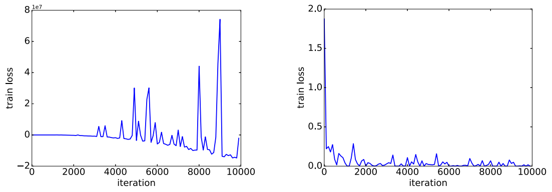

(a) (b)

Figure 1: Training losses of (a) straightforward formulation Eq. (2); and (b) large margin formulation Eq. (3)

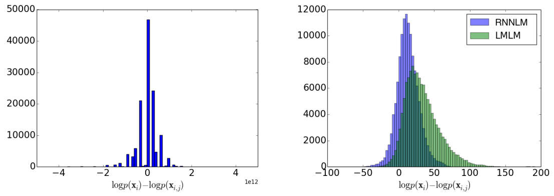

(a) (b)

Figure 2: Histogram of the margin $\log p(\mathbf{x}_i) - \log p(\mathbf{x}_{i,j})$. The more positive, the more the discrimination. (a) Straightforward formulation; (b) LMLM compared with RNNLM (a minimum-PPL LM trained on Common Crawl)

apart "good" and "bad" candidates. With this motivation, we train to assign larger LM-scores for the $\mathbf{x}_i$'s but smaller ones for the (imperfect) $\mathbf{x}_{i,j}$'s. A quantity of particular interest is $\log p(\mathbf{x}_i) - \log p(\mathbf{x}_{i,j})$, the *margin*/difference between the LM-scores of the references and the (imperfect) hypotheses. The intuition is that the more positive the margin, the better the LM is at discrimination.

3.2 Straightforward but Failed Formulation

Without loss of generality, we assume that all the $\mathbf{x}_{i,j}$'s are imperfect and different from $\mathbf{x}_i$. A straightforward way is to adopt the following objective:

$$\min_{\theta} \frac{1}{N}\sum_{i=1}^{N}\left(-\log p_\theta(\mathbf{x}_i) + \frac{1}{K}\sum_{j=1}^{K}\log p_\theta(\mathbf{x}_{i,j})\right). \tag{2}$$

Similar formulation is also seen in (Tachioka and Watanabe, 2015), where they only utilize one beam candidate, *i.e.*, $K = 1$. Optimization can be carried out by mini-batch stochastic gradient descent (SGD). Each iteration, SGD randomly samples a batch of i's and j's, computes stochastic gradient w.r.t. θ, and takes an update step. However, a potential problem with this formulation is that the second term (corresponding to the inferior hypotheses) may dominate the optimization. Specifically, the training is almost always driven by the $\mathbf{x}_{i,j}$'s, but does not effectively enhance the discrimination. We illustrate this fact in the following experiment.

Using the ASR system in section 3.1, we extract 256 beam candidates for every training example in Wall Street Journal (WSJ) dataset. Warm started from the pre-trained RNNLM in section 3.1, we apply SGD to minimize the loss in Eq. (2), with a mini-batch size of 128. The training loss is shown in Fig. 1a. We observe that the learning dynamic is very unstable, and deceases to be negative. The unbound decreasing is due to the second term in Eq. (2) being negative and dominating the training process. Next, we inspect $\log p_\theta(\mathbf{x}_i) - \log p_\theta(\mathbf{x}_{i,j})$, the *margin* between the scores of a ground-truth and a candidate. In Fig. 2a, we histogram the margins for all the i, j's in a dev set. The distribution appears to be symmetric around zero, which indicates poor discrimination ability. Given these facts, we conclude that the straightforward formulation in Eq. (2) is not effective.

3.3 Large Margin Formulation

To effectively utilize all the imperfect beam candidates, we propose the following objective,

$$\min_{\theta} \sum_{i=1}^{N}\sum_{j=1}^{B} \max\left\{0, \tau - (\log p_\theta(\mathbf{x}_i) - \log p_\theta(\mathbf{x}_{i,j}))\right\}, \tag{3}$$

where $\log p_\theta(\mathbf{x}_i) - \log_\theta(\mathbf{x}_{i,j})$ is the margin between the scores of a ground-truth $\mathbf{x}_i$ and a candidate $\mathbf{x}_{i,j}$. The hinge loss on the margin encourages the log-likelihood of the ground-truth to be at least τ larger than that of the imperfect hypothesis. We call an LM trained by the above formulation as Large Margin Language Model (LMLM).

We repeat the same experiment in section 3.2, but change the objective function to Eq. (3) and set $\tau = 1$. Fig. 1b shows the training loss, which steadily decreases and approaches zero rapidly. Compared with the learning curve of naive formulation (Fig. 1a), the large margin based training is much more stable. In Fig. 2b, we also examine the histogram of $\log p_\theta(\mathbf{x}_i) - \log p_\theta(\mathbf{x}_{i,j})$, where $p_\theta(\cdot)$ is now the LM learned by LMLM. Compared with the histogram by the conventional RNNLM, LMLM significantly moves the distribution to the positive side, indicating more discrimination.

3.4 Ranking Loss Type Formulation

In most cases, all beam candidates are imperfect. It may be beneficial to exploit the information that some candidates are relatively better than the others. We consider ranking them according to some metrics w.r.t. the ground-truth sentences. For ASR, the metric is WER, and for MT, the metric is BLEU score. We define $\mathbf{x}_{i,0} \triangleq \mathbf{x}_i$ and assume that the candidates $\{\mathbf{x}_{i,j}\}_{j=1}^{K}$ are sorted such that

$$\text{WER}(\mathbf{x}_i, \mathbf{x}_{i,j-1}) < \text{WER}(\mathbf{x}_i, \mathbf{x}_{i,j})$$

for ASR, and

$$\text{BLEU}(\mathbf{x}_i, \mathbf{x}_{i,j-1}) > \text{BLEU}(\mathbf{x}_i, \mathbf{x}_{i,j})$$

for MT. In other words, $\mathbf{x}_{i,j-1}$ has better quality than $\mathbf{x}_{i,j}$.

We then enforce the "better" sentences to have a score at least τ larger than those "worse" ones. This leads to the following formulation,

$$\min_\theta \sum_{i=1}^{N} \sum_{j=0}^{B-1} \sum_{k=j+1}^{B} \max \{0, \tau - (\log p_\theta(\mathbf{x}_{i,j}) - \log_\theta(\mathbf{x}_{i,k})), \}. \quad (4)$$

Compared with LMLM formulation Eq. (3), the above introduces more comparisons among the candidates, and hence more computational cost during training. We call this formulation ranking-loss-based LMLM (rLMLM).

To summarize this section, we have proposed LMLM and rLMLM that aim at discriminating between hypotheses in a task-specific (e.g., WER or BLEU) sense, instead of minimizing PPL.

4 Experiments on ASR

We apply the LMs trained under different criteria to rescore the beams in various ASR systems. In particular, we are interested in knowing which of the two training mechanisms is better: minimizing PPL (e.g., the RNNLM in Section 3.1), or fitting to the WER metric by the proposed methods.

Adapting an RNNLM to a specific domain has been of interest, especially to the speech community (Park et al., 2010; Chen et al., 2015; Ma et al., 2017). We adopt Ma et al. (2017) that fine-tune the softmax layer of RNNLM by minimizing the PPL on the text labels of training set. According to Ma et al. (2017), the reason not to fine-tune all the layers is due to the limited text labels in the target domain. Indeed, we also observe overfitting if adapting all layers, but adapting only the softmax layer effectively decreases the PPL on the text labels of dev sets. We refer to this fine-tuning as *RNNLM-adapted* in the following sections.

To make a fair comparison with the adapted model, we also use the RNNLM as an initialization for our LMLM and rLMLM. In total, there are four language models for rescoring the beams. RNNLM and its adapted version that aim at reducing PPL; and the two proposed methods, LMLM and rLMLM that try to fit to WER.

4.1 WSJ Dataset

The WSJ corpora consists of about 80 hours of read speech with texts drawn from a machine-readable corpus of Wall Street Journal news. We use the standard configuration of train si284 dataset for training, dev93 for development and eval92 for testing.

Our ASR model has one convolution layer, followed by 5 bidirectional RNNs and one fully connected layer, with a CTC loss on top. The text labels of the training set are used to train a 4-gram language model, which is employed in the ASR decoder. The beam search decoder has a beam width of 2000. Before beam rescoring, this ASR system achieves a WER of 12.16 on dev93 set and 7.69 on eval92 set. To put this into perspective, we list some previous state-of-the-art system in Tab. 2. Compared with them, our baseline is already very competitive.

4.1.1 WERs and PPLs

The out-of-vocabulary rate of WSJ text is only 0.28%, making the RNNLM reasonable to use.

We apply the RNNLM, RNNLM-adapted (Ma et al., 2017), LMLM and rLMLM to rescore the beams on dev and test set. The final score assigned to a beam is a weighted sum of the ASR and language model scores. The weight is found by minimizing the WER on the dev set.

Tab. 3 reports the WERs on dev93 and eval92 sets. All methods reduce the WER over the baseline without rescoring. However, LMLM and rLMLM are notably better than the other two methods. Moreover, although RNNLM and RNNLM-adapted achieve smaller PPLs on the text labels, the advantage does not transfer to WER.

ASR Models	WER	
	dev93	eval92
EESEN (Miao et al., 2015)	N/A	7.34
Attention (Bahdanau et al., 2016)	N/A	9.30
Gram-CTC (Liu et al., 2017)	N/A	6.75
5-layer Bidi-RNNs (baseline)	12.16	7.69

Table 2: Published WERs on WSJ dev93 and eval92 set

rescoring language model	WER		PPL	
	dev93	eval92	dev93	eval92
baseline (no rescore)	12.16	7.69	N/A	N/A
RNNLM	10.71	6.59	207.43	205.00
RNNLM-adapted	10.11	6.34	159.50	157.85
LMLM	**9.44**	*5.56*	575.83	563.69
rLMLM	*9.63*	**5.48**	345.60	348.32

Table 3: Rescore 2000-best list of WSJ dev93 and eval92 set. Digits in bold are the best and italics are the runner-ups. Lower PPL does not correspond to lower WER.

4.1.2 Correlation between scores and WERs

To better understand the proposed methods, we calculate the correlation coefficients between the hypotheses' WERs and their scores (by different language models). In specific, for every utterance in the test set, we have a set of beam candidates, their word level accuracies (100-WER) and scores given by an LM, from which a Pearson correlation coefficient can be calculated. We calculate the coefficients for all the utterances in the test set, and boxplot these coefficients in Fig. 3. The correlation coefficients by LMLM and rLMLM tend to be higher than RNNLM and RNNLM-adapted. This indicates that LMLM and rLMLM are more aligned with the goal of reducing WER.

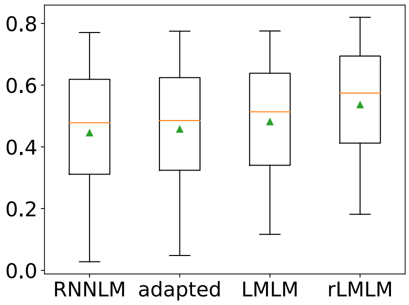

Figure 3: Correlation coefficients between word level accuracy ($1 - \text{WER}/100$) and LM-scores by the different LMs, higher is better. Red horizontal lines are medians. Green dots are means. Whiskers are 5% and 95% quantiles. Lower and upper box boundaries are 25% and 75% quantiles.

4.1.3 Case Study

Tab. 4 posts some examples from the test set. The first column lists the ground-truth labels, and their corresponding best candidates as re-ranked by the four LMs (see notes in the second column). Words in red are mistakes made by the candidate sentences. Scores of these sentences are listed in the last four columns. We have the following observations:

1. LMLM and rLMLM give worse scores on the ground-truth labels than RNNLM and RNNLM-adapted, which explains their higher PPL in Tab. 3.

2. In the first example, RNNLM and RNNLM-adapted assign higher scores to a shorter sentence. This is reasonable (though not necessarily desirable) as LM-score is a summation of log-probabilities, each of which is negative. In contrast, LMLM and rLMLM are able to assign higher scores to longer and better candidates.

3. In the other two examples, LMLM and rLMLM seem to favor more sensible sentences, though they are not more grammatical than those picked by RNNLM and RNNLM-adapted. We conjecture that since LMLM and rLMLM utilize beam candidates in their training, they capture and compensate for

references and hypotheses	reference or ranked 1st by	*LM-score*			
		RNNLM	RNNLM -adapted	LMLM	rLMLM
for such group rate coverage employers can charge the former workers and their families the average cost of providing the health benefits plus a two percent administrative fee	reference, LMLM	-144.74	-142.90	-162.61	-165.93
for such group rate coverage employers can charge the former workers and their families the average cost of providing the health benefits plus ␣ two percent administrative ␣	RNNLM, RNNLM-adapted	-144.53	-142.85	-172.66	-168.10
for such group rate coverage employers can charge the former workers and their families the average cost of providing their health benefits plus a two percent administrative fee	rLMLM	-146.72	-145.67	-163.73	-162.92
we'd like to see something that leads to real democracy says jaime bonilla vice secretary general	reference	-105.8	-106.28	-122.93	-108.42
we'd like to see something that leads the real democracy says jm bonier vice secretary general	RNNLM, RNNLM-adapted	-103.13	-102.84	-141.94	-118.38
we'd like to see something that leads to real democracy says jim bone vice secretary general	LMLM, rLMLM	-104.52	-105.37	-125.40	-104.73
the big shoe is going to drop when we see the trade number	reference	-64.28	-61.04	-80.63	-82.91
the big she was going to drop in to see the trade numbers	RNNLM, RNNLM-adapted	-64.32	-61.68	-94.26	-86.89
the big shoe is going to drop in ␣ see the trade number	LMLM, rLMLM	-67.53	-64.50	-84.09	-84.65

Table 4: Some "gold" references and best hypotheses (after rescoring by different language models) for eval92 set. In red are errors or missing word (denoted as '␣').

some weakness in the ASR, which is not achieved by RNNLM and RNNLM-adapted.

4.2 10K Speech Dataset

We further validate our methods on a larger noisy dataset collected by Liu et al. (2017). The dataset has about 10K hours of spontaneous speech. The utterances are corrupted by background noise, and a large portion of them are accented. Therefore it is much more challenging than WSJ. We adopt the same training-dev-test split as in Liu et al. (2017). In specific, there are 5.4M utterances for training, 2,066 for development and 2,054 for testing.

rescoring language model	WER		PPL	
	dev	test	dev	test
baseline (no rescore)	19.17	20.90	N/A	N/A
RNNLM	18.38	20.07	264.21	252.85
RNNLM-adapted	18.29	20.03	236.74	226.22
LMLM	*18.17*	*19.62*	2250.79	2095.39
rLMLM	**17.98**	**19.49**	1225.04	1152.63

Table 5: Rescore 2000-best list of our internal dev and test set. Digits in bold are the best and italics are the runner-ups.

The ASR we build has the same architecture as in Liu et al. (2017), except that its decoder integrates an in-domain 5-gram language model. This system achieves a WER of 19.17 on dev set, better than the reported 19.77 baseline in Liu et al. (2017). Based on the ASR, we repeat the same experiments in section 4.1. Tab. 5 reports WERs and PPLs on dev and test sets. Both LMLM and rLMLM outperform the other methods in WER, although their PPLs are higher. This trend is similar to that in Tab. 3.

5 Experiments on NMT

In this section, we experiment the large-margin criterion trained LM with a competitive Chinese-to-English NMT system. The NMT model is trained from 2M parallel sentence pairs. Following Shen et al. (2016), we use NIST 06 newswire portion (616 sentences) for development and NIST 08 newswire portion (691 sentences) for testing. We use OpenNMT-py[2] package with the default configuration to train the model: batch size is 64; word embedding size is 500; dropout rate is 0.3; target vocabulary size is 50K; number of epochs is 20, after which a minimum dev perplexity of 7.72

---

[2] `https://github.com/OpenNMT/OpenNMT-py`

is achieved.

5.1 BLEUs and PPLs

We use a beam size of 10 for decoding, and report case-insensitive 4-reference BLEU-4 scores (by calling "multi_bleu.perl"[3]). The NMT model achieves 35.18 BLEU score on dev set and 31.52 on test set (see table 6). To put this into perspective, Shen et al. (2016) trains their models on 2.56M pairs of sentences and reports a dev BLEU score of 32.7 (via MOSES) or 30.7 (via RNNsearch, beam size of 10). So our NMT model is already very competitive.

To construct the training data for LMLM and rLMLM, 10 beam candidates are extracted for every sentence in the training set. We then follow the same experimental steps outlined in section 4.1, except that the ASR score is now changed to NMT score. In addition, we also find that normalizing the LM score by sentence length can improve the re-scoring performance substantially. Tab. 6 compares the BLEU score after re-ranking by the different LMs. LMLM and rLMLM both improve upon the baseline significantly, and outperform RNNLM and RNNLM-adapted by a notable margin. We also observe that the PPLs of LMLM and rLMLM are much larger than those of RNNLM and RNNLM-adapted, suggesting that the PPL metric may be very poorly correlated with BLEU.

Interestingly, RNNLM-adapted does not show any gain in BLEU score over RNNLM. To understand this, we recall that NMT is trained by minimizing PPL on target text. Its decoder is implicitly an RNNLM on target language. We conjecture that adapting an LM to the target domain can only duplicate the functionality of the NMT decoder, which does not bring any additional benefit.

rescoring language model	BLEU		PPL	
	dev06	test08	dev06	test08
baseline (no rescore)	35.18	31.52	N/A	N/A
RNNLM	36.17	32.17	129.91	137.25
RNNLM-adapted	36.17	31.97	78.20	89.27
LMLM	*37.79*	*33.11*	7.75e5	3.73e6
rLMLM	**37.82**	**33.13**	2.68e5	1.12e6

Table 6: Rescore 10-best list for dev (nist 06) and test (nist 08) set. Digits in bold are the best and italics are the runner-ups.

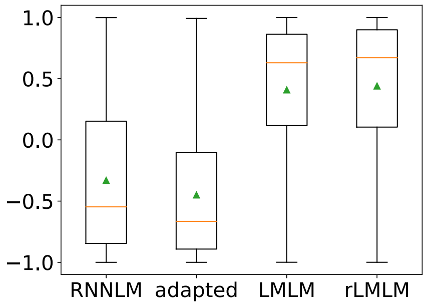

Figure 4: Correlation coefficients between BLEUs and LM-scores by the different LMs, higher is better. Red horizontal lines are medians. Green dots are means. Whiskers are 5% and 95% quantiles. Lower and upper box boundaries are 25% and 75% quantiles.

5.2 Correlation between scores and BLEUs

We measure the correlation between the LM scores and BLUEs. The calculation is done on dev06 set in the same way as Section 4.1.2, but now we change the WERs to BLEUs. The boxplot of the correlation coefficients are shown in Fig. 4. Compared with the boxplot in Fig. 3, now the correlation coefficients by all LMs are more dispersed. Sometimes, they even take negative values. The mean correlation by LMLM and rLMLM, however, is considerably higher than those by RNNLM and RNNLM-adapted.

6 Related Work

"*Language modeling is an art of determining the probability of a sequence of words*" (Goodman, 2001). In the past decades, there has been a trend of increasing the context that an LM can condition on. N-gram models (Chen and Goodman, 1996) assume that each symbol depends on the previous $N-1$ symbols. Feed forward neural network based LMs (Bengio et al., 2003) are not count based but they inherit the restrictive assumption. To model longer-term dependencies, RNNLMs (Mikolov et al., 2010) are proposed. RNNLMs often achieve smaller PPLs than the N-gram counterparts (Sundermeyer et al., 2012; Zaremba et al., 2014; Jozefowicz et al., 2016). This paper focuses on RNNLM-type architectures.

While these works all adopt PPL as the metric

---

[3] `https://github.com/OpenNMT/OpenNMT-py/blob/master/tools/multi-bleu.perl`

to optimize, sometimes one may optimize a task-specific objective. For example, Kuo et al. (2002); Roark et al. (2007) and Dikic et al. (2013) propose discriminative LMs to improve speech recognition. The common methodology therein is to fit a probabilistic model, e.g., conditional random field (Roark et al., 2004), to the space of text candidates, and maximize the probability at the desired candidate. The problem is often solved by perceptron algorithm. However, these methods all rely on ad-hoc choice of features, e.g., counts of n-grams where n varies in a small range (e.g.,1 to 3). Moreover, it is also not clear how these methods would take advantage of an existing language model (trained on large unsupervised corpus). Nevertheless, the same methodology can be extended to RNNLMs, thus avoiding the aforementioned limitations. For example, Auli and Gao (2014) train an RNNLM by favoring sentences with high BLEU scores and integrate it into a phrase-based MT decoder.

If we cast the problem of picking the best text sequence as a ranking problem, the aforementioned works can be considered as "pointwise" learning-to-rank approaches (Cossock and Zhang, 2008). In contrast, the proposed method is a "pairwise" approach (Liu, 2009), as it learns a neural language model by comparison between pairs of sentences. Earlier works in this fashion may date back to (Collins and Koo, 2005), which improves a semantic parser. Learning "by pairwise comparison" is also seen in several MT literatures. For example, Hopkins and May (2011) propose to train a phrase-based MT system by minimizing a pairwise ranking loss. Wiseman and Rush (2016) optimize the beam search process in a Neural Machine Translation (NMT) system. They enforce the score of a reference to be higher than that of its decoded k-th candidate by at least a unit margin.

Rather than optimizing the MT system itself, this work proposes a general method of training recurrent neural language models, which can benefit various text generation tasks, including speech recognition and machine translation.

7 Conclusions

We have proposed a large margin criterion for training recurrent neural language models. Rather than minimizing PPL, the proposed criterion is based on comparison between pairs of sentences. We have formulated two algorithms that implement the training criterion. One compares between references and imperfect hypotheses (LMLM), the other compares between all pairs of hypotheses (rLMLM). We applied the language models trained by these two algorithms to speech recognition and machine translation. Both of them demonstrate superior performance over their minimum-PPL counterparts. However, the performance gain from LMLM to rLMLM is small, although rLMLM is built on more pairwise comparisons and requires more training efforts. The efficiency with respect to the number of pairs is a future research topic.

References

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, and Jared Casper et al. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*.

Michael Auli and Jianfeng Gao. 2014. Decoder integration and expected bleu training for recurrent neural network language models. In *ACL*.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Xie Chen, Tian Tan, Xunying Liu, Pierre Lanchantin, Moquan Wan, Mark J.F. Gales, and Philip C. Woodland. 2015. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition. In *16th Annual Conference of the International Speech Communication Association*.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

David Cossock and Tong Zhang. 2008. Statistical analysis of bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154.

Erinç Dikic, Murat Semerci, Murat Saraçlar, and Ethem Alpaydin. 2013. Classification and ranking approaches to discriminative language modeling for asr. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(2):291–300.

Joshua T. Goodman. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *In Proceedings of the 23rd international conference on Machine learning*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *EMNLP*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Hong-Kwang Jeff Kuo, Eric Fosler-Lussier, Hui Jiang, and Chin-Hui Lee. 2002. Discriminative training of language models for speech recognition in acoustics. In *IEEE International Conference on Speech, and Signal Processing (ICASSP)*.

Zhifei Li and Sanjeev Khudanpur. 2008. Large-scale discriminative n-gram language models for statistical machine translation. In *AMTA*.

Hairong Liu, Zhenyao Zhu, Xiangang Li, and Sanjeev Satheesh. 2017. Gram-ctc: Automatic unit selection and target decomposition for sequence labelling. In *34th International Conference on Machine Learning*.

Tie-Yan Liu. 2009. *Learning to rank for information retrieval*, volume 3. Foundations and Trends® in Information Retrieval.

Min Ma, Michael Nirschl, Fadi Biadsy, and Shankar Kumar. 2017. Approaches for neural-network language model adaptation. In *Proceedings of Interspeech*.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. 2015. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding, pages 167–174. ieee, 2015. In *Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *11th Annual Conference of the International Speech Communication Association*.

Junho Park, Xunying Liu, Mark J. F. Gales, and Phil C. Woodland. 2010. Improved neural network based language modelling and adaptation. In *11th Annual Conference of the International Speech Communication Association*.

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech & Language*, 21(2):373–392.

Brian Roark, Murat Saraclar, and Michael Collins Mark Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *42nd Annual Meeting on Association for Computational Linguistics*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *ACL*.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2017. Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *13th Annual Conference of the International Speech Communication Association*.

Yuuki Tachioka and Shinji Watanabe. 2015. Discriminative method for recurrent neural network language models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Dong Yu and Li Deng. 2014. *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publish-ing Company.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Targeted Syntactic Evaluation of Language Models

Rebecca Marvin
Department of Computer Science
Johns Hopkins University
`becky@jhu.edu`

Tal Linzen
Department of Cognitive Science
Johns Hopkins University
`tal.linzen@jhu.edu`

Abstract

We present a dataset for evaluating the grammaticality of the predictions of a language model. We automatically construct a large number of minimally different pairs of English sentences, each consisting of a grammatical and an ungrammatical sentence. The sentence pairs represent different variations of structure-sensitive phenomena: subject-verb agreement, reflexive anaphora and negative polarity items. We expect a language model to assign a higher probability to the grammatical sentence than the ungrammatical one. In an experiment using this data set, an LSTM language model performed poorly on many of the constructions. Multi-task training with a syntactic objective (CCG supertagging) improved the LSTM's accuracy, but a large gap remained between its performance and the accuracy of human participants recruited online. This suggests that there is considerable room for improvement over LSTMs in capturing syntax in a language model.

1 Introduction

A language model (LM) defines a probability distribution over sequences of words. Recent technological advances have led to an explosion of neural network-based LM architectures. The most popular ones are based on recurrent neural networks (RNNs) (Elman, 1990; Mikolov et al., 2010), in particular Long Short-Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997). While a large number of alternative architectures have been proposed in the past few years, LSTMs are still highly competitive (Melis et al., 2018).

Language models are typically evaluated using perplexity: it is considered desirable for an LM to assign a high probability to held-out data from the same corpus as the training data. This measure conflates multiple sources of success (or failure) in predicting the next word: common collocations, semantics, pragmatics, syntax, and so on. The quality of the **syntactic** predictions made by the LM is arguably particularly difficult to measure using perplexity: since most sentences are grammatically simple and most words can be predicted from their local context, perplexity rewards LMs primarily for collocational and semantic predictions.

We propose to supplement perplexity with a metric that assesses whether the probability distribution defined by the model conforms to the grammar of the language. Following previous work (Lau et al., 2017; Linzen et al., 2016; Gulordava et al., 2018), we suggest that given two sentences that differ minimally from each other, one of which is grammatical and the other which is not, it is desirable for the model to assign a higher probability to the grammatical one.

The value of this approach can be illustrated with a recent study by Tran et al. (2018), where a standard LSTM language model was compared to an attention-only LM without recurrence (Vaswani et al., 2017). Although the attention-only model had somewhat better perplexity on the validation set, when the models were tested specifically on challenging subject-verb agreement dependencies, the attention-only model made three times as many errors as the LSTM. In other words, the LSTM learned more robust syntactic representations, but this advantage was not reflected in its average perplexity on the corpus, since syntactically challenging sentences are relatively infrequent.

Previous work on targeted syntactic evaluation of language models has identified syntactically challenging sentences in corpora (Linzen et al., 2016; Gulordava et al., 2018). While evaluation on naturally occurring examples is appealing, this approach has its limitations (see Section 2). In particular, syntactically challenging examples are sparsely represented in a corpus, their identifica-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1192–1202
Brussels, Belgium, October 31 - November 4, 2018.

tion requires a clean parsed corpus, and naturally occurring sentences are difficult to control for confounds. We contrast the naturalistic approach with a constructed dataset, which allows us to examine a much larger range of specific grammatical phenomena than has been possible before. We use templates to automatically create our test sentences, making it possible to generate a large test set while maintaining experimental control over our materials as well as a balanced number of examples of each phenomenon.

We test three LMs on the data set we develop: an n-gram baseline, an RNN LM trained on an unannotated corpus, and an RNN LM trained on a multitask objective: language modeling and Combinatory Categorial Grammar (CCG) supertagging (Bangalore and Joshi, 1999). We also conduct a human experiment using the same materials. The n-gram baseline largely performed at chance, suggesting that good performance on the task requires syntactic representations. The RNN LMs performed well on simple cases, but struggled on more complex ones. Multi-task training with a supervised syntactic objective improved the performance of the RNN, but it was still much weaker than humans. This suggests that our data set is challenging, especially when explicit syntactic supervision is not available, and can therefore motivate richer language modeling architectures.

2 Overview of the approach

2.1 Grammaticality and LM probability

How should grammaticality be captured in the probability distribution defined by an LM? The most extreme position would be that a language model should assign a probability of zero to ungrammatical sentences. For most applications, some degree of error tolerance is desirable, and it is not practical to assign a sentence a probability of exactly zero.[1] Following Linzen et al. (2016) and Gulordava et al. (2018), our desideratum for the language model is more modest: if two closely matched sentence differ only in their grammaticality, the probability of the grammatical sentence should be higher than the probability of the ungrammatical one. For example, the following minimal pair illustrates the fact that third-person present English verbs agree with the number of their subject:

(1) *Simple agreement*:
  a. The author laughs.
  b. *The author laugh.

We expect the probability of (1a) to be higher than the probability of (1b). Previous work has simplified this setting further by comparing the probability that the LM assigns to a single word that is the locus of ungrammaticality. In (1), for example, the LM would be fed the first two words of the sentence, and would be considered successful on the task if it predicts $P(laughs) > P(laugh)$.

The prediction setting is only applicable when the locus of ungrammaticality is a single word, rather than, say, the interaction between two words; moreover, the information needed to make the grammaticality decision needs to be available in the left context of the locus of grammaticality. These conditions do not always hold. Negative polarity items (NPIs), for example, are words like *any* and *ever* that can only be used in the scope of negation.[2] The grammaticality of placing a particular quantifier in the beginning of the sentences in (2) depends on whether the sentence contains an NPI later on:

(2) *Simple NPI:*
  a. No students have ever lived here.
  b. *Most students have ever lived here.

It would not be possible to compare these two sentences using the prediction task. In the current paper, we use the more general setting and compare the probability of the two complete sentences.

2.2 Data set construction

Previous work has used syntactically complex sentences identified from a parsed corpus. This approach has several limitations. If the corpus is automatically parsed, the risk of a parse error increases with the complexity of the construction (Bender et al., 2011). If the test set is restricted to sentences with gold parses, it can be difficult or impossible to find a sufficient number of examples of syntactically challenging cases. Moreover, using naturally occurring sentences can introduce

---

[1] Nor is it possible to have a threshold ϵ such that all grammatical sentences have probability higher than ϵ and all ungrammatical sentences have probability lower than ϵ, for the simple reason that there is an infinite number of grammatical sentences (Lau et al., 2017).

[2] In practice, the conditions that govern the distribution of NPIs are much more complicated, but this first approximation will suffice for the present purposes. For a review, see Giannakidou (2011).

confounds that may complicate the interpretation of the experiments (Ettinger et al., 2018).

To circumvent these issues, we use templates to automatically construct a large number of English sentence pairs (∼350,000). Our data set includes three phenomena that linguists consider to be sensitive to hierarchical syntactic structure (Everaert et al., 2015; Xiang et al., 2009): subject-verb agreement (described in detail in Sections 4.1 and 4.2), reflexive anaphora (Section 4.3) and negative polarity items (Section 4.4).

The templates can be described using non-recursive context-free grammars. We specify the preterminal symbols that make up a syntactic construction and have different terminal symbols that those preterminals could be mapped to. For example, the template for the simple agreement construction illustrated in (1) consists of the following rules:

(3) a. Simple agreement → D MS MV
 b. D → *the*
 c. MS → {*author*, *pilot*, . . .}
 d. MV → {*laughs*, *smiles*, . . .}

We generate all possible combinations of the terminals. The Supplementary Materials provide a full description of all our templates.[3]

While these examples are somewhat artificial, our goal is to isolate the syntactic capabilities of the model; it is in fact beneficial to minimize the semantic or collocational cues that can be used to identify the grammatical sentence. Gulordava et al. took this approach further and constructed "colorless green ideas" test cases by substituting random content words into sentences from a corpus. We take a more moderate position and avoid combinations that are very implausible or violate selectional restrictions (e.g., *the apple laughs*). We do this by having separate templates for animate and inanimate subjects and verbs so that the resulting sentences are always reasonably plausible.

3 Related work

Targeted evaluation: LM evaluation data sets using challenging prediction tasks have been proposed in the context of semantics and discourse comprehension (Zweig and Burges, 2011; Paperno et al., 2016). Evaluation sets consisting of challenging syntactic constructions have been constructed for parser evaluation (Rimell et al., 2009; Nivre et al., 2010; Bender et al., 2011), and minimal pair approaches have been proposed for evaluating image captioning (Shekhar et al., 2017) and machine translation systems (Sennrich, 2017), but no data sets exist that target a range of syntactic constructions for language model evaluation.

Acceptability judgments: Lau et al. (2017) compared the ability of different LMs to predict graded human acceptability judgments. The forced-choice approach used in the current paper has been shown to be effective in human acceptability judgment experiments (Sprouse and Almeida, 2017). In some early work, neural networks were trained explicitly to predict acceptability judgments (Lawrence et al., 1996; Allen and Seidenberg, 1999); Post (2011) likewise trained a classifier on top of a parser to predict grammaticality. Warstadt et al. (2018) use a transfer learning approach, where an unsupervised model is fine-tuned on acceptability prediction. Our work differs from those studies in that we do not advocate providing any explicit grammaticality signal to the LM at any point ("no negative evidence").

Syntax in LMs: There have been several proposals over the years to incorporate explicit syntax into LMs to overcome the inability of *n*-gram LMs to model long-distance dependencies (Jurafsky et al., 1995; Roark, 2001; Pauls and Klein, 2012). While RNN language models can in principle model longer dependencies (Mikolov et al., 2010; Linzen et al., 2016), in practice it can still be beneficial to inject syntax into the model. This can be done by combining it with a supervised parser (Dyer et al., 2016) or other multi-task learning objectives (Enguehard et al., 2017). Our work is orthogonal to this area of research, but can be seen as providing a potential opportunity to underscore the advantage of such syntax-infused models.

4 Data set composition

This section describes all of the types of sentence pairs included in our data set, which include examples of subject-verb agreement (Sections 4.1 and 4.2), reflexive anaphoras (Section 4.3) and negative polarity items (Section 4.4).

---

[3] The code, the data set and the Supplementary Materials can be found at `https://github.com/BeckyMarvin/LM_syneval`.

4.1 Subject-verb agreement

Determining the correct number of the verb is trivial in examples such as (1) above, in which the sentence only contains a single noun. By contrast, in cases where there are multiple nouns in the sentence, identifying which of them is the subject of a given verb requires understanding the structure of the sentence. In particular, the relevant subject is not necessarily the first noun of the sentence:

(4) *Agreement in a sentential complement:*
 a. The bankers knew the officer smiles.
 b. *The bankers knew the officer smile.

Here the verb *smiles* needs to agree with the embedded subject *officer* rather than the main clause subject *bankers*. The subject is also not necessarily the most recent noun before the verb: when the subject is modified by a phrase, a distracting noun ("attractor") often intervenes in the linear order of the sentence between the head of the subject and the verb. Two examples of such modifiers are prepositional phrases and relative clauses (RCs):

(5) *Agreement across a prepositional phrase:*
 a. The farmer near the parents smiles.
 b. *The farmer near the parents smile.

(6) *Agreement across a subject relative clause:*
 a. The officers that love the skater smile.
 b. *The officers that love the skater smiles.

We include all four possible configurations of noun number for each type of minimal pair; for (5), these would be:[4]

(7) a. The farmer near the parent smiles/*smile.
 b. The farmer near the parents smiles/*smile.
 c. The farmers near the parent smile/*smiles.
 d. The farmers near the parents smile/*smiles.

Sentences where the two nouns conflict in number are expected to be more challenging, but interpretable errors may certainly occur even when they do not. For example, the model may use the heuristic that sentences with multiple nouns are likely to have a plural verb (a heuristic that would be effective for coordination); alternatively, it might prefer singular verbs to plural ones regardless of whether the subject is singular or plural, simply because the singular form of the verb is more frequent.

Next, in verb phrase (VP) coordination, both of the verbs need to agree with the subject:

(8) *Short VP coordination:*
 a. The senator smiles and laughs.
 b. *The senator smiles and laugh.

We had both singular and plural subjects. The number of the verb immediately adjacent to the subject was always grammatical. This problem can in principle be solved with a trigram model (*smiles and laughs* is likely to be a more frequent trigram than *smiles and laugh*); to address this potential concern, we also included a coordination condition with a longer dependency:

(9) *Long VP coordination:*
 The manager writes in a journal every day and likes/*like to watch television shows.

4.2 Agreement and object relative clauses

We go into greater depth in object relative clauses, which most clearly require a hierarchical representation. In (10) and (11), the model needs to be able to distinguish the embedded subject (*parents*) from the main clause subject (*farmer*) when making its predictions:

(10) *Agreement across an object relative clause:*
 a. The farmer that the parents love swims.
 b. *The farmer that the parents love swim.

(11) *Agreement in an object relative clause:*
 a. The farmer that the parents love swims.
 b. *The farmer that the parents loves swims.

In keeping with the minimal pair approach, we never introduce two agreement errors at the same time: either the embedded verb or the main verb is incorrectly inflected, but not both.

We include a number of variations on the pattern in (11). First, we delete the relativizer *that*, with the hypothesis that the absence of an overt cue to structure will make the task more difficult:

(12) The farmer the parents love/*loves swims.

In another condition, we replace the main subject with an inanimate noun and keep the embed-

---

[4] The slash notation indicates the word that differs between the grammatical and ungrammatical sentence; for example, in (7a), the full sentence pair would be:

(i) a. The farmer near the parent smiles.
 b. *The farmer near the parent smile.

ded subject animate. We base this manipulation on human experimental work showing that similar nouns (for example, two animate nouns) are more likely to cause confusion during comprehension than dissimilar nouns, such as an animate and an inanimate noun (Van Dyke, 2007):

(13) The movies that the author likes are/*is good.

For a complete list of all the types of minimal pairs we include, see the Supplementary Materials.

4.3 Reflexive anaphora

A reflexive pronoun such as *himself* needs to have an antecedent from which it derives its interpretation. The pronoun needs to agree in number (and gender) with its antecedent:

(14) *Simple reflexive*:
  a. The senators embarrassed themselves.
  b. *The senators embarrassed herself.

There are structural conditions on the nouns to which a reflexive pronoun can be bound. One of these conditions requires the antecedent to be in the same clause as the reflexive pronoun. For example, (15b) cannot refer to a context in which *the pilot* embarrassed *the bankers*:

(15) *Reflexive in a sentential complement*:
  a. The bankers thought the pilot embarrassed himself.
  b. *The bankers thought the pilot embarrassed themselves.

Likewise, in the following minimal pair, sentence (16b) is ungrammatical, because the reflexive pronoun *themselves*, which is part of the main clause, cannot be bound to the noun phrase *the architects*, which is inside an embedded clause:

(16) *Reflexive across an object relative clause:*
  a. The manager that the architects like doubted himself.
  b. *The manager that the architects like doubted themselves.

4.4 Negative polarity items

Negative polarity items, introduced in example (2) above, are words that (to a first approximation) need to occur in the context of negation. Crucially for the purposes of the present work, the scope of negation is structurally defined. In particular the negative noun phrase needs to c-command the NPI: the syntactic non-terminal node that dominates the negative noun phrase must also dominate the NPI. This is the case in (17a), but not in (17b), where the negative noun phrase is too deep in the tree to c-command the NPI *ever* (Xiang et al., 2009; Everaert et al., 2015).

(17) *NPI across a relative clause:*
  a. No authors that the security guards like have ever been famous.
  b. *The authors that no security guards like have ever been famous.

All of the nouns and verbs in the NPI cases were plural. As in some of the agreement cases, we included a variant of (17) in which the subject was inanimate.

5 Experimental setup

To show how our challenge set can be used to evaluate the syntactic performance of LMs, we trained three LMs with increasing levels of syntactic sophistication. All of the LMs were trained on a 90 million word subset of Wikipedia (Gulordava et al., 2018). Our *n*-gram LM and LSTM LM do not require annotated data. The third model is also an LSTM LM, but it requires syntactically annotated data (CCG supertags).

***N*-gram model:** We trained a 5-gram model on the same 90M word corpus using the SRILM toolkit (Stolcke, 2002) which backs off to smaller *n*-grams using Kneser-Ney smoothing.

Single-task RNN: The RNN LM had two layers of 650 LSTM units, a batch size of 128, a dropout rate of 0.2, and a learning rate of 20.0, and was trained for 40 epochs (following the hyperparameters of Gulordava et al. 2018).

Multi-task RNN: In multi-task learning, the system is trained to optimize an objective function that combines the objective functions of several tasks. We combine language modeling with CCG supertagging, a task that predicts for each word in the sentence its CCG supertag (Bangalore and Joshi, 1999; Lewis et al., 2016). We simply sum the two objective functions with equal weights (Enguehard et al., 2017). Early stopping in this model is based on the combined loss on language modeling and supertagging. Supertags provide a large amount of syntactic information about the word; the sequence of supertags of a sentence strongly constrains the possible parses of the sentence. We use supertagging as a "scaffold" task (Swayamdipta et al., 2017): our goal is not to produce a competitive supertagger, but to induce better syntactic representations, which would then lead to improved language modeling. We used CCG-Bank (Hockenmaier and Steedman, 2007) as our CCG corpus.

Human evaluation: We designed a human experiment on Amazon Mechanical Turk that mirrored the task that was given to the LMs: both versions of a minimal pair were shown on the screen at the same time, and participants were asked to judge which one of them was more acceptable (for details, see the Supplementary Materials). We emphasize that we do not see human performance on complex syntactic dependencies as setting an upper bound on the performance that we should expect from an LM. There is a rich literature showing that humans make mistakes such as subject-verb agreement errors; in fact, most of the phenomena we test were inspired by work in psycholinguistics that studies these errors (Bock and Miller, 1991; Phillips et al., 2011). At the same time, while we do not see a reason not to aspire for 100% accuracy, we are interested in comparing LM and human errors: if the errors are similar, the two systems may be using similar representations.

6 Results

Local agreement: The overall accuracy per condition can be seen in Table 1. The *n*-gram LM's accuracy was only 79% for simple agreement and agreement in a sentential complement, both of which can be solved entirely using local context. This is because not all subject and verb combinations in our materials appeared verbatim in the 90M word training corpus; for those combinations, the model fell back on unigram probabilities, which in this context amounts to selecting the more frequent form of the verb.

Both RNNs performed much better than the *n*-gram model on the simple agreement case (single-task: 94%; multi-task: 100%), reflecting these models' ability to generalize beyond the specific bigrams that occurred in the corpus. Accuracy on agreement in a sentential complement was also very high (single-task: 99%; multi-task: 93%). This indicates that the RNNs do not rely on the heuristic whereby the first noun of the sentence is likely to be its subject. They did slightly worse but still very well on short VP coordination (both 90%); this dependency is also local, albeit across the word *and*.

Non-local agreement: The accuracy of the *n*-gram model on non-local dependencies (long VP coordination and agreement across a phrase or a clause) was very close to 50%. This suggests that local collocational information is not useful in these conditions. The single-task RNN also performed much more poorly on these conditions than on the local agreement conditions, though for the most part its accuracy was better than chance. Humans did worse on these dependencies as well, but their accuracy did not drop as sharply as the RNNs' (human accuracies ranged from 82% to 88%). In most of these cases, multi-task learning was very helpful; for example, accuracy in long VP coordination increased from 61% to 81%. Still, both RNNs performed poorly on agreement across an object RC, especially without *that*, whereas humans performed comparably on all non-local dependencies.

Agreement inside an object RC: This case is particularly interesting, because this dependency is purely local (see (11)), and the interference is from the distant sentence-initial noun. Although this configuration is similar to the sentential complement case, performance was worse both in RNNs and humans. However, RNNs performed *better* than humans, at least when the sentence included the overt relativizer *that*. This suggests that interference is sensitive to proximity in RNNs but to syntactic status in humans — humans appear to be confusing the main clause subject and the embedded subject (Wagers et al., 2009).

Reflexive anaphora: The RNNs' performance was significantly worse on simple reflexives (83%) than on simple agreement (94%), and did not differ between the single-task and multi-task models. By contrast, human performance did not differ between subject-verb agreement and reflexive anaphoras. The surprisingly poor performance for this adjacent dependency seems to be due to an asymmetry in accuracy between *himself* and *themselves* on the one hand (100% accuracy in the multi-task RNN) and *herself* on the other hand (49% accuracy).[5] Accuracy was very low for all

---

[5] This may be because *himself* and *themselves* are significantly more frequent than *herself*, and consequently the num-

	RNN	Multitask	*n*-gram	Humans	# sents
SUBJECT-VERB AGREEMENT:					
Simple	0.94	1.00	0.79	0.96	280
In a sentential complement	0.99	0.93	0.79	0.93	3360
Short VP coordination	0.90	0.90	0.51	0.94	1680
Long VP coordination	0.61	0.81	0.50	0.82	800
Across a prepositional phrase	0.57	0.69	0.50	0.85	44800
Across a subject relative clause	0.56	0.74	0.50	0.88	22400
Across an object relative clause	0.50	0.57	0.50	0.85	44800
Across an object relative (no *that*)	0.52	0.52	0.50	0.82	44800
In an object relative clause	0.84	0.89	0.50	0.78	44800
In an object relative (no *that*)	0.71	0.81	0.50	0.79	44800
REFLEXIVE ANAPHORA:					
Simple	0.83	0.86	0.50	0.96	560
In a sentential complement	0.86	0.83	0.50	0.91	6720
Across a relative clause	0.55	0.56	0.50	0.87	44800
NEGATIVE POLARITY ITEMS:					
Simple	0.40	0.48	0.06	0.98	792
Across a relative clause	0.41	0.73	0.60	0.81	31680

Table 1: Overall accuracies for the LSTMs, *n*-gram model and humans on each test case.

pronouns in the structurally complex case in which the dependency was across a relative clause (55% compared to 87% in humans).

NPIs: The dependency in simple NPIs spans only four words, so the *n*-gram model could in principle capture it. In practice, the *n*-gram model systematically selected the wrong answer, suggesting that it backed off to comparing the bigrams *no students* and *most students*, the first of which is presumably less frequent. Surprisingly, the *n*-gram model's accuracy was higher than 50% on NPIs across a relative clause, a dependency that spans more than five words. In this case, the bigrams *that the* and *the chef* (for example) happen to be more frequent than the *that no* and *no chef.* This difference was apparently strong enough to make up for the low-frequency bigram at the start of the sentence.

The RNNs did poorly on this task. The accuracy of the single-task model was around 40%. The multi-task did somewhat better on the simple NPIs (48%) and much better on the NPIs across a relative clause (73%). At the same time, an examination of the plot of log probability of each word in a sentence (Figure A.1 in the Supplementary Materials) suggests that the single-task RNN is in fact able to differentiate between the grammatical and ungrammatical sentences when it reaches the NPI, but this difference does not offset the overall probability advantage of the ungrammatical sentence (which is likely due to non-grammatical collocational factors). In any case, the fact that the *n*-gram baseline did not perform at chance suggests that there are non-syntactic cues to this task, complicating the interpretation of the performance of other LMs.

Perplexity: The perplexity of the *n*-gram model on the Wikipedia test data was 157.5, much higher than the perplexity of the single-task RNN (78.65) and the multi-task RNN (61.10). In other words, perplexity tracked accuracy on our syntactic data set – an unsatisfying outcome given our goal of dissociating perplexity and our syntactic evaluation method, but an expected one given that each model was conditioned on richer information than the previous one. In previous work, perplexity and syntactic judgment accuracy have been found to be partly dissociable (Kuncoro et al., 2018; Tran et al., 2018).

Lexical variation and frequency: There was considerable lexical variation in the results; we have mentioned the surprising asymmetry between *himself* and *herself* above. As another case study, we examine variation in the results of the simple agreement condition in the single-

---

ber representation learned for *herself* was not robust. Another possibility is that gender bias reduces the probability of an anaphoric relation between *herself* and words such as *surgeon* (Rudinger et al., 2018).

Main subject	Embedded subject	Single-task	Multi-task	Humans	Example sentence
Across an objective relative clause:					
Singular	Singular	0.83	0.77	0.96	The author that the minister likes laughs/*laugh.
Singular	Plural	0.51	0.30	0.90	The author that the ministers like laughs/*laugh.
Plural	Singular	0.18	0.53	0.77	The authors that the minister likes laugh/*laughs.
Plural	Plural	0.50	0.73	0.80	The authors that the ministers like laugh/*laughs.
Within an objective relative clause:					
Singular	Singular	0.73	0.92	0.94	The author that the minister likes/*like laughs.
Singular	Plural	0.91	0.81	0.72	The author that the ministers like/*likes laugh.
Plural	Singular	0.81	0.97	0.73	The authors that the minister likes/*like laugh.
Plural	Plural	0.87	0.84	0.76	The authors that the ministers like/*likes laugh.

Table 2: Accuracy within and across an object relative clause (only in the cases in which the main subject was animate and the relativizer *that* was present). The subject that the verb is expected to agree with is underlined.

task RNN. Accuracy varied by verb, ranging from *is* and *are*, which had 100% accuracy, to *swims*, where accuracy was only 60% (recall that average accuracy was 94%). This may be a frequency effect: either the LM is learning less robust number representations for infrequent verbs, or the tail of the distribution over the vocabulary is more fragile during word prediction. Pauls and Klein (2012) propose normalizing for unigram frequency when deriving acceptability judgments from an LM. Our preliminary experiments with this method did not significantly improve overall performance; regardless of the effectiveness of this method, such corrections should arguably not be necessary in an LM that adequately captures grammaticality.

7 Case study: agreement and object relative clauses

The overall results in Table 1 were averaged over all of the possible number configurations within each condition. In this section, we take a closer look at agreement in sentences with an object RC (see Table 2). This kind of finer-grained analysis helps explain the cases in which the LMs are failing, and might reveal some of the patterns or heuristics the LMs are using.

Performance in agreement across an object RC was poor. Both RNNs made attraction errors: they often preferred the verb that agreed in number with the irrelevant embedded subject to the verb that agreed with the correct main subject. The multi-task RNN showed greater symmetry between the simpler singular/singular and plural/plural cases, whereas the single-task RNN performed poorly even in these cases, often preferring a singular verb when both subjects were plural. This default preference for singular verbs matches the behavior of younger children (Franck et al., 2004).

Performance in agreement within an object RC was better; still, the single-task RNN made the most errors when both subjects were singular, perhaps due to a heuristic in which a sentence with multiple subjects is likely to have a plural verb (as in coordination sentences). By contrast, the multi-task model seemed to have a general bias towards singular subjects in this condition. Incidentally, the human results with object RCs were also unexpected: while attraction errors when the two subjects differ in number are to be expected (Wagers et al., 2009), our participants made a sizable number of errors even when both subjects were plural.

Despite the generally poor performance in object RCs, Figures A.2 and A.3 in the Supplementary Materials show that the single-task RNN is typically assigning a higher probability to the grammatical word of a minimal pair than to the ungrammatical word.

8 Discussion

We have described a template-based data set for targeted syntactic evaluation of language models. The data set consists of pairs of sentences that are matched except for their grammaticality; we consider a language model to capture the relevant aspects of the grammar of the language if it assigns a higher probability to the grammatical sentence than to the ungrammatical one.

An RNN language model performed very well on local subject-verb agreement dependencies, significantly outperforming an *n*-gram baseline.

This suggests that the task is a viable evaluation strategy. Even on simple cases, however, the RNN's accuracy was sensitive to the particular lexical items that occurred in the sentence; this would not be expected if its syntactic representations were fully abstract. The RNN's performance degraded markedly on non-local dependencies, approaching chance levels on agreement across an object relative clause. Multi-task training with a syntactic objective (CCG supertagging) mitigated this drop in performance for some but not all of the dependencies we tested. We conjecture that the benefits of the inductive bias conferred by multi-task learning will be amplified when the amount of training data is limited.

Our results contrast with the results of Gulordava et al. (2018), who obtained a prediction accuracy of 81% on English sentences from their test corpus and 74% on constructed sentences modeled after sentences from the corpus. It is likely that our sentences are more syntactically challenging than the ones they were able to find in the relatively small manually annotated treebank they used.

One limitation of our approach is that it is not always clear what constitutes a minimal grammaticality contrast. In the subject-verb agreement case, the contrast was clear: the two present-tense forms of the verb, e.g., *laugh* vs. *laughs*. Our NPI manipulations, on the other hand, were less successful: the members of the contrasts differed not only in their syntactic structure but also in low-level *n*-gram probabilities, making the performance on this particular contrast harder to interpret.

We emphasize that the goal of this article was not to advocate for LSTMs in particular as an effective architecture for modeling syntax; indeed, our results show that LSTM language models are far from matching naive annotators' performance on this task, let alone performing at 100% accuracy. We hope that our data set, and future extensions to other phenomena and languages, will make it possible to measure progress in syntactic language modeling and will lead to better understanding of the syntactic generalizations captured by language models.

9 Acknowledgments

We would like to thank Ming Xiang for sharing materials from human experiments that inspired many of our test cases. We also thank Brian Roark and the JHU Computational Psycholinguistics lab for discussion, and Brian Leonard for help conducting the human experiment.

References

Joseph Allen and Mark S. Seidenberg. 1999. The emergence of grammaticality in connectionist networks. In Brian MacWhinney, editor, *Emergentist approaches to language: Proceedings of the 28th Carnegie symposium on cognition*, pages 115–151. Mahwah, NJ: Lawrence Erlbaum Associates.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Emily M. Bender, Dan Flickinger, Stephan Oepen, and Yi Zhang. 2011. Parser evaluation over local and non-local deep dependencies in a large corpus. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 397–408. Association for Computational Linguistics.

Kathryn Bock and Carol A. Miller. 1991. Broken agreement. *Cognitive Psychology*, 23(1):45–93.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Émile Enguehard, Yoav Goldberg, and Tal Linzen. 2017. Exploring the syntactic abilities of RNNs with multi-task learning. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 3–14. Association for Computational Linguistics.

Allyson Ettinger, Ahmed Elgohary, Colin Phillips, and Philip Resnik. 2018. Assessing composition in sentence vector representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1790–1801. Association for Computational Linguistics.

Martin B. H. Everaert, Marinus A. C. Huybregts, Noam Chomsky, Robert C. Berwick, and Johan J. Bolhuis. 2015. Structures, not strings: Linguistics as part of the cognitive sciences. *Trends in Cognitive Sciences*, 19(12):729–743.

Julie Franck, Stephany Cronel-Ohayon, Laurence Chillier, Ulrich H. Frauenfelder, Cornelia Hamann, Luigi Rizzi, and Pascal Zesiger. 2004. Normal and pathological development of subject–verb agreement in speech production: A study on French children. *Journal of Neurolinguistics*, 17(2-3):147–180.

Anastasia Giannakidou. 2011. Negative and positive polarity items: Variation, licensing, and compositionality. In *Semantics: An international handbook of natural language meaning*, pages 1660–1712. Berlin: Mouton de Gruyter.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205. Association for Computational Linguistics.

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

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, G Tajchaman, and Nelson Morgan. 1995. Using a stochastic context-free grammar as a language model for speech recognition. In *Proceedings of ICASSP*, volume 1, pages 189–192. IEEE.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436. Association for Computational Linguistics.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2017. Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive Science*, (5):1202–1247.

Steve Lawrence, Lee C. Giles, and Santliway Fong. 1996. Can recurrent neural networks learn natural language grammars? In *IEEE International Conference on Neural Networks*, volume 4, pages 1853–1858.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *Proceedings of the International Conference on Learning Representations*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, pages 1045–1048, Makuhari, Chiba, Japan.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez Rodríguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 833–841. Coling 2010 Organizing Committee.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534. Association for Computational Linguistics.

Adam Pauls and Dan Klein. 2012. Large-scale syntactic language modeling with treelets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 959–968. Association for Computational Linguistics.

Colin Phillips, Matthew W. Wagers, and Ellen F. Lau. 2011. Grammatical illusions and selective fallibility in real-time language comprehension. In Jeffrey T. Runner, editor, *Experiments at the Interfaces, Syntax and Semantics 37*, pages 153–186. Bingley, U.K.: Emerald.

Matt Post. 2011. Judging grammaticality with tree substitution grammar derivations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 217–222. Association for Computational Linguistics.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821. Association for Computational Linguistics.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 8–14. Association for Computational Linguistics.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? Assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382. Association for Computational Linguistics.

Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich, Aurélie Herbelot, Moin Nabi, Enver Sangineto, and Raffaella Bernardi. 2017. FOIL it! Find one mismatch between image and language caption. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 255–265. Association for Computational Linguistics.

Jon Sprouse and Diogo Almeida. 2017. Design sensitivity and statistical power in acceptability judgment experiments. *Glossa*, 2(1):14.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Proccedings of the Seventh International Conference on Spoken Language Processing*, pages 901–904.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental RNNs and a syntactic scaffold. *arXiv preprint arXiv:1706.09528*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Julie A. Van Dyke. 2007. Interference effects from grammatically unavailable constituents during sentence processing. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 33(2):407–430.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Matthew W. Wagers, Ellen F. Lau, and Colin Phillips. 2009. Agreement attraction in comprehension: Representations and processes. *Journal of Memory and Language*, 61(2):206–237.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Ming Xiang, Brian Dillon, and Colin Phillips. 2009. Illusory licensing effects across dependency types: ERP evidence. *Brain and Language*, 108(1):40–55.

Geoffrey Zweig and Christopher J. C. Burges. 2011. The Microsoft Research sentence completion challenge. Technical report, Microsoft.

Rational Recurrences

Hao Peng♠ **Roy Schwartz**♠◇ **Sam Thomson**♣ **Noah A. Smith**♠◇
♠Paul G. Allen School of Computer Science & Engineering, University of Washington
♣School of Computer Science, Carnegie Mellon University
◇Allen Institute for Artificial Intelligence
{hapeng,roysch,nasmith}@cs.washington.edu, sthomson@cs.cmu.edu

Abstract

Despite the tremendous empirical success of neural models in natural language processing, many of them lack the strong intuitions that accompany classical machine learning approaches. Recently, connections have been shown between convolutional neural networks (CNNs) and weighted finite state automata (WFSAs), leading to new interpretations and insights. In this work, we show that some *recurrent* neural networks also share this connection to WFSAs. We characterize this connection formally, defining **rational recurrences** to be recurrent hidden state update functions that can be written as the Forward calculation of a finite set of WFSAs. We show that several recent neural models use rational recurrences. Our analysis provides a fresh view of these models and facilitates devising new neural architectures that draw inspiration from WFSAs. We present one such model, which performs better than two recent baselines on language modeling and text classification. Our results demonstrate that transferring intuitions from classical models like WFSAs can be an effective approach to designing and understanding neural models.

1 Introduction

Neural models, and in particular gated variants of recurrent neural networks (RNNs, e.g., Hochreiter and Schmidhuber, 1997; Cho et al., 2014), have become a core building block for state-of-the-art approaches in NLP (Goldberg, 2016). While these models empirically outperform classical NLP methods on many tasks (Zaremba et al., 2014; Bahdanau et al., 2015; Dyer et al., 2016; Peng et al., 2017, *inter alia*), they typically lack the intuition offered by classical models, making it hard to understand the roles played by each of their components. In this work we show that many neural models are more interpretable than previously thought, by drawing connections to weighted finite state automata (WFSAs). We study several recently proposed RNN architectures and show that one can use WFSAs to characterize their recurrent updates. We call such models **rational recurrences** (§3).[1] Analyzing recurrences in terms of WFSAs provides a new view of existing models and facilitates the development of new ones.

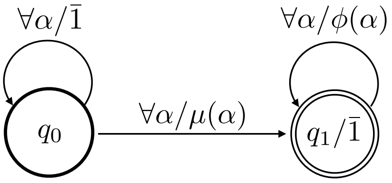

Figure 1: A two-state WFSA $\mathcal{B}$ described in §2. It is closely related to several models studied in this paper (§4.1). Bold circles indicate initial states, and double circles final states, which are associated with final weights. Arrows represent transitions, labeled by the symbols α they consume, and the weights as a function of α. Arcs not drawn are assumed to have weight $\bar{0}$. For brevity, $\forall\alpha$ means $\forall\alpha \in \Sigma$, with Σ being the alphabet.

In recent work, Schwartz et al. (2018) introduced SoPa, an RNN constructed from WFSAs, and thus rational by our definition. They also showed that a single-layer max-pooled CNN (LeCun, 1998) can be simulated by a set of simple WFSAs (one per output dimension), and accordingly are also rational. In this paper we broaden such efforts, and show that rational recurrences are in frequent use (Mikolov et al., 2014; Balduzzi and Ghifary, 2016; Lei et al., 2016, 2017a,b; Bradbury et al., 2017; Foerster et al., 2017). For instance, we will show in §4 that the WFSA diagrammed

---

[1] Where the term *regular* is used with unweighted FSAs (e.g., regular languages, regular expressions), *rational* is the weighted analog (e.g., rational series, Sakarovitch, 2009; rational kernels, Cortes et al., 2004).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1203–1214
Brussels, Belgium, October 31 - November 4, 2018.

in Figure 1 has strong connections to several of the models mentioned above.

Based on these observations, we then discuss potential approaches to deriving novel neural architectures from WFSAs (§5). As a case study, we present a new model motivated by the interpolation of a two-state WFSA and a three-state one, capturing (soft) unigram and bigram features, respectively. Our experiments show that in two tasks—language modeling and text classification—the proposed model outperforms recently proposed rational models (§6). Further extensions might lead to larger gains, and the rational recurrence view could facilitate easier exploration of such extensions. To promote such exploration, we publicly release our implementation at `https://github.com/Noahs-ARK/rational-recurrences`.

2 Background: Weighted Finite State Automata (WFSAs)

This section reviews weighted finite-state automata and semirings, which underly our analyses in §3. WFSAs extend nondeterministic *unweighted* finite-state automata by assigning weights to transitions, start states, and final states. Instead of simply accepting or rejecting a string, a WFSA returns a score for the string, and this score summarizes the weights along all paths through the WFSA that consume the string. In order for this summary score to be efficiently computable, weights are taken from a **semiring**.

Definition 1 (Kuich and Salomaa, 1986)**.** A **semiring** is a set $\mathbb{K}$ along with two associative binary operations on $\mathbb{K}$, $\oplus$ (addition) and $\otimes$ (multiplication), and two identity elements: $\bar{0}$ for addition, and $\bar{1}$ for multiplication. Semirings also require that addition is commutative, multiplication distributes over addition, and that multiplication by $\bar{0}$ annihilates (i.e., $\bar{0} \otimes a = a \otimes \bar{0} = \bar{0}$).

One common semiring is the **real** (or plus-times) semiring: $\langle \mathbb{R}, +, \cdot, 0, 1\rangle$. The other one used in this work is the **max-plus** semiring $\langle \mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0\rangle$. We refer the reader to Kuich and Salomaa (1986) for others.

Definition 2. A **weighted finite-state automaton (WFSA)** over a semiring $\mathbb{K}$ is a 5-tuple, $\mathcal{A} = \langle \Sigma, \mathcal{Q}, \tau, \lambda, \rho\rangle$,[2] with:

- a finite input alphabet Σ;
- a finite state set $\mathcal{Q}$;
- transition weights $\tau : \mathcal{Q} \times \mathcal{Q} \times (\Sigma \cup \{\varepsilon\}) \to \mathbb{K}$;
- initial weights $\lambda : \mathcal{Q} \to \mathbb{K}$;
- and final weights $\rho : \mathcal{Q} \to \mathbb{K}$.

$\varepsilon \notin \Sigma$ marks special **ε-transitions** that may be taken without consuming any input. $\mathcal{A}$ assigns a score $\mathcal{A}[\![\mathbf{x}]\!]$ to a string $\mathbf{x} = x_1 \dots x_n \in \Sigma^*$ by summing over the scores of all possible paths deriving $\mathbf{x}$. The score of each individual path is the product of the weights of the transitions it consists of. Formally:

Definition 3 (path score)**.** Let $\boldsymbol{\pi} = \pi_1 \dots \pi_n$ be a sequence of adjacent **transitions** in $\mathcal{A}$, with each transition $\pi_i = (q_i, q_{i+1}, z_i) \in \mathcal{Q} \times \mathcal{Q} \times (\Sigma \cup \{\varepsilon\})$. The path $\boldsymbol{\pi}$ **derives** string $\mathbf{x} \in \Sigma^*$, which is the substring of $\mathbf{z} = z_1 z_2 \dots z_n$ that excludes ε symbols (for example, if $\mathbf{z} = a\varepsilon bc\varepsilon\varepsilon\varepsilon d$, then $\mathbf{x} = abcd$). $\boldsymbol{\pi}$'s score in $\mathcal{A}$ is given by

$$\mathcal{A}[\boldsymbol{\pi}] = \lambda(q_1) \otimes \left(\bigotimes_{i=1}^{n} \tau(\pi_i) \right) \otimes \rho(q_{n+1}). \quad (1)$$

Definition 4 (string score)**.** Let $\Pi(\mathbf{x})$ denote the set of all paths in $\mathcal{A}$ that derive $\mathbf{x}$. Then the score assigned by $\mathcal{A}$ to $\mathbf{x}$ is defined to be

$$\mathcal{A}[\![\mathbf{x}]\!] = \bigoplus_{\boldsymbol{\pi} \in \Pi(\mathbf{x})} \mathcal{A}[\boldsymbol{\pi}]. \quad (2)$$

Because $\mathbb{K}$ is a semiring, $\mathcal{A}[\![\mathbf{x}]\!]$ can be computed in time linear in $|\mathbf{x}|$ by the Forward algorithm (Baum and Petrie, 1966). Here, for simplicity, we describe the Forward algorithm without ε-transitions.[3] Its dynamic program is given by:

$$\Omega_0(q) = \lambda(q), \quad (3a)$$

$$\Omega_{i+1}(q) = \bigoplus_{q' \in \mathcal{Q}} \Omega_i(q') \otimes \tau(q', q, x_i), \quad (3b)$$

$$\mathcal{A}[\![\mathbf{x}]\!] = \bigoplus_{q \in \mathcal{Q}} \Omega_n(q) \otimes \rho(q). \quad (3c)$$

$\Omega_i(q)$ gives the total score of all paths that derive $x_1 \dots x_i$ and end in state q.

Example 5. Figure 1 diagrams a WFSA $\mathcal{B}$, consisting of two states. A path starts from the initial state q_0 (with $\lambda(q_0) = \bar{1}$); it then takes any num-

---

[2] Some authors define τ, λ, and ρ to be partial functions—applying only a subset of transitions, initial states, and final states respectively. Our definition is equivalent, giving the weight functions value $\bar{0}$ wherever they were undefined.

[3] ε-transitions can be handled with a slight modification (Schwartz et al., 2018). Note though that if $\mathcal{A}$ contains a cycle of ε-transitions, then either $\mathbb{K}$ must follow the **star semiring** laws (Kuich and Salomaa, 1986), or the number of consecutive ε-transitions allowed must be capped.

ber of “self-loop” transitions, each consuming an input without changing the path score (since it's weighted by $\bar{1}$); it then consumes an input symbol α and takes a transition weighted by $\mu(\alpha)$, and reaches the final state q_1 (with $\rho(q_1) = \bar{1}$); it may further consume more input by taking self-loops at q_1, updating the path score by multiplying it by $\phi(\alpha)$ for each symbol α. Then from Definition 4, we can calculate that $\mathcal{B}$ gives the empty string score $\bar{0}$, and gives any nonempty string $\mathbf{x} = x_1 \ldots x_n \in \Sigma^+$ score $\mathcal{B}[\![\mathbf{x}]\!] =$

$$\bigoplus_{i=1}^{n-1} \left(\mu(x_i) \otimes \bigotimes_{j=i+1}^{n} \phi(x_j) \right) \oplus \mu(x_n). \tag{4}$$

$\mathcal{B}$ can be seen as capturing soft unigram patterns (Davidov et al., 2010), in the sense that it consumes one input symbol to reach the final state from the initial state. It is straightforward to design WFSAs capturing longer patterns by including more states (Schwartz et al., 2018), as we will discuss later in §4 and §5.

3 Rational Recurrences

Before formally defining rational recurrences in §3.2, we highlight the connection between WFSAs and RNNs using a motivating example (§3.1).

3.1 A Motivating Example

We describe a simplified RNN which strips away details of some recent RNNs, in order to highlight the behaviors of the forget gate and the input.

Example 6. For an input sequence $\mathbf{x} = x_1 \ldots x_n$, let the word embedding vector for x_t be $\mathbf{v}_t$. As in many gated RNN variants (Hochreiter and Schmidhuber, 1997; Cho et al., 2014), we use a forget gate $\mathbf{f}_t$, which is computed with an affine transformation followed by an elementwise sigmoid function $\boldsymbol{\sigma}$. The current input representation $\mathbf{u}_t$ is similarly computed, but with an optional nonlinearity (e.g., tanh) $\boldsymbol{g}$. The hidden state $\mathbf{c}_t$ can be seen as a weighted sum of the previous state and the new input, controlled by the forget gate.

$$\mathbf{f}_t = \boldsymbol{\sigma}\big(\mathbf{W}_f \mathbf{v}_t + \mathbf{b}_f\big), \tag{5a}$$
$$\mathbf{u}_t = (\mathbf{1} - \mathbf{f}_t) \odot \boldsymbol{g}\big(\mathbf{W}_u \mathbf{v}_t + \mathbf{b}_u\big), \tag{5b}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{u}_t. \tag{5c}$$

The hidden state $\mathbf{c}_t$ can then be used in downstream computation, e.g., to calculate output state $\mathbf{h}_t = \tanh(\mathbf{c}_t)$, which is then fed to an MLP classifier. We focus only on the recurrent computation.

In Example 6, both $\mathbf{f}_t$ and $\mathbf{u}_t$ depend only on the current input token x_t (through $\mathbf{v}_t$), and not the previous state. Importantly, the interaction with the previous state $\mathbf{c}_{t-1}$ is *not* via affine transformations followed by nonlinearities, as in, e.g., an Elman network (Elman, 1990), where $\mathbf{c}_t = \tanh(\mathbf{W}_c \mathbf{c}_{t-1} + \mathbf{W}_v \mathbf{v}_t + \mathbf{b}_c)$. As we will discuss later, this is important in relating this recurrent update function to WFSAs.

Since the recurrent update in Equation 5c is elementwise, for simplicity we focus on just the ith dimension. Unrolling it in time steps, we get

$$\begin{aligned} [\mathbf{c}_t]_i &= [\mathbf{f}_t]_i \, [\mathbf{c}_{t-1}]_i + [\mathbf{u}_t]_i \\ &= \sum_{j=1}^{t-1} \left([\mathbf{u}_j]_i \prod_{k=j+1}^{t} [\mathbf{f}_k]_i \right) + [\mathbf{u}_t]_i, \end{aligned} \tag{6}$$

where $[\cdot]_i$ denotes the ith dimension of a vector. As noted by Lee et al. (2017), the hidden state at time step t can be seen as a sum of previous input representations, weighted by the forget gate; longer histories typically get a smaller weight, since the forget gate values are between 0 and 1 due to the sigmoid function.

Let's recall the WFSA $\mathcal{B}$ (Figure 1 and Example 5) using the real semiring $\langle \mathbb{R}, +, \cdot, 0, 1 \rangle$. Equation 6 is recovered by parameterizing $\mathcal{B}$'s weight functions μ and ϕ with

$$\mu(x_t) = [\mathbf{u}_t]_i, \quad \phi(x_t) = [\mathbf{f}_t]_i. \tag{7}$$

Denote the resulting WFSA by $\mathcal{B}_i$, and we have:

Proposition 7. *Running a single layer RNN in Example 6 over any nonempty input string $\mathbf{x} \in \Sigma^+$, the ith dimension of its hidden state at time step t equals the score assigned by $\mathcal{B}_i$ to $\mathbf{x}_{:t}$:*

$$[\mathbf{c}_t]_i = \mathcal{B}_i[\![\mathbf{x}_{:t}]\!]. \tag{8}$$

In other words, the ith dimension of the RNN in Example 6 can be seen as a WFSA structurally equivalent to $\mathcal{B}$. Its weight functions are implemented as the ith dimension of Equations 5, and the learned parameters are the ith row of $\mathbf{W}$ and $\mathbf{b}$. Then it is straightforward to recover the full d-dimensional RNN, by collecting d such WFSAs, each of which is parametrized by a row in the $\mathbf{W}$s and $\mathbf{b}$s. Based on this observation, we are now ready to formally define rational recurrences.

3.2 Recurrences and Rationality

For a function $\mathbf{c}\colon \Sigma^* \to \mathbb{K}^d$, its *recurrence* is said to be the dependence of $\mathbf{c}(\mathbf{x}_{:t})$ on $\mathbf{c}(\mathbf{x}_{:t-1})$, for input sequence $\forall \mathbf{x} \in \Sigma^+$. We discuss a class of recurrences that can be characterized by WFSAs. The mathematical counterpart of WFSAs are *rational power series* (Berstel and Reutenauer, 1988), justifying naming such recurrences *rational*:

Definition 8 (rational recurrence)**.** The recurrence of $\mathbf{c}\colon \Sigma^* \to \mathbb{K}^d$ is said to be *rational*, if there exists a set of weighted finite state automata $\{\mathcal{A}_i\}_{i=1}^d$ over alphabet Σ and semiring $\langle \mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1}\rangle$ with both $\oplus$ and $\otimes$ taking constant time and space, such that $\forall \mathbf{x} \in \Sigma^*$,

$$\big[\mathbf{c}(\mathbf{x})\big]_i = \mathcal{A}_i[\![\mathbf{x}]\!],\ \forall i \in \{1, 2, \ldots, d\}.^4 \qquad (9)$$

It directly follows from Proposition 7 that

Corollary 9. *The recurrence in Example 6 is rational.*

4 Relationship to Existing Neural Models

This section studies several recently proposed neural architectures, and relates them to rational recurrences. §4.1 begins by relating some of them to the RNN defined in Example 6, and then to the WFSA $\mathcal{B}$ (Example 5). We then describe a WFSA similar to $\mathcal{B}$, but with one additional state, and discuss how it provides a new view of RNN models motivated by n-gram features (§4.2). In §4.3 we study rational recurrences that are not elementwise, using an existing model.

In the following discussion, we shall assume the real seimiring, unless otherwise noted.

4.1 Neural Architectures Related to $\mathcal{B}$

Despite its simplicity, Example 6 corresponds to several existing neural architectures. For instance, quasi-RNN (QRNN; Bradbury et al., 2017) and simple recurrent unit (SRU; Lei et al., 2017b) aim to speed up the recurrent computation. To do so, they drop the matrix multiplication dependence on the previous hidden state, resulting in similar recurrences to that in Example 6.[5] Other works start from different motivations, but land on similar recurrences, e.g., strongly-typed RNNs (T-RNN; Balduzzi and Ghifary, 2016) and its gated variants (T-LSTM and T-GRU), and structurally constrained RNNs (SCRN; Mikolov et al., 2014).

The analysis in §3.1 directly applies to SRU, T-RNN, and SCRN. In fact, Example 6 presents a slightly more complicated version of them. In these models, input representations are computed without the bias term or any nonlinearity: $\mathbf{u}_t = \mathbf{W}_u \mathbf{v}_t$. By Proposition 7 and Corollary 9:

Corollary 10. *The recurrences of single-layer SRU, T-RNN, and SCRN architectures are rational.*

It is slightly more complicated to analyze the recurrences of the QRNN, T-LSTM, and T-GRU. Although their hidden states $\mathbf{c}_t$ are updated in the same way as Equation 5c, the input representations and gates may depend on *previous inputs*. For example, in T-LSTM and T-GRU, the forget gate is a function of two consecutive inputs:

$$\mathbf{f}_t = \boldsymbol{\sigma}\left(\mathbf{V}_f \mathbf{v}_{t-1} + \mathbf{W}_f \mathbf{v}_t + \mathbf{b}_f\right). \qquad (10)$$

QRNNs are similar, but may depend on up to K tokens, due to the K-window convolutions. Eisner (2002) discuss finite state machines for second (or higher) order probabilistic sequence models. Following the same intuition, we sketch the construction of WFSAs corresponding to QRNNs with 2-window convolutions in Appendix A, and summarize the key results here:

Proposition 11. *The recurrences of single-layer T-GRU, T-LSTM, and QRNN are rational. In particular, a single-layer d-dimensional QRNN using K-window convolutions can be recovered by a set of d WFSAs, each with $O(2\,|\Sigma|^{K-1})$ states.*

The size of WFSAs needed to recover QRNN grows exponentially in the window size. Therefore, at least for QRNNs, Proposition 11 has more conceptual value than practical.

4.2 More than Two States

So far our discussion has centered on $\mathcal{B}$, a two-state WFSA capturing unigram patterns (Example 5). In the same spirit as going from unigram to n-gram features, one can use WFSAs with more states to capture longer patterns (Schwartz et al., 2018). In this section we augment $\mathcal{B}$ by introducing more states, and explore its relationship to some neural architectures motivated by n-gram features. We start with a three-state WFSA as an example, and then discuss more general cases.

Figure 2 diagrams a WFSA $\mathcal{C}$, augmenting $\mathcal{B}$ with another state. To reach the final state q_2, at

---

[4] We restrict that both operations take constant time and space, to exclude the use of arbitrarily complex semirings (§4.3).

[5] The SRU architecture discussed through this work is based on Lei et al. (2017b). In a later updated version, Lei et al. (2018) introduce diagonal matrix multiplication interaction in the hidden state updates, inspired by (Li et al., 2018), which yields a recurrence not obviously rational.

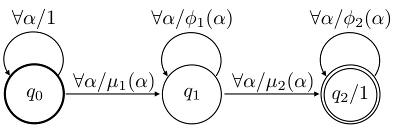

Figure 2: A three-state WFSA $\mathcal{C}$ discussed in §4.2.

least two transitions must be taken, in contrast to one in $\mathcal{B}$. History information is decayed by the self-loop at the final state q_2, assuming ϕ_2 is between 0 and 1. $\mathcal{C}$ has another self-loop over q_1, weighted by $\phi_1 \in (0,1)$. The motivation is to allow (but down-weight) nonconsecutive bigrams, as we will soon show.

The scores assigned by $\mathcal{C}$ can be inductively computed by applying the Forward algorithm (§2). Given input sequence $\mathbf{x}$ longer than one, let $\mathcal{C}[\![\mathbf{x}_{:0}]\!] = 0$, then $\mathcal{C}[\![\mathbf{x}_{:t+1}]\!] =$

$$\mathcal{C}[\![\mathbf{x}_{:t}]\!]\,\phi_2(x_{t+1}) + \beta_t\,\mu_2(x_{t+1}), \tag{11}$$

where

$$\beta_t = \beta_{t-1}\,\phi_1(x_t) + \mu_1(x_t), \tag{12}$$

and $\beta_0 = 0$. Unrolling β_t in time, we get $\beta_t =$

$$\sum_{j=1}^{t-1}\left(\mu_1(x_j)\prod_{k=j+1}^{t}\phi_1(x_k)\right) + \mu_1(x_t). \tag{13}$$

Due to the self-loop over state q_1, β_t can be seen as a weighted sum of the μ_1 terms up to x_t (Equaltion 13). The second product term in Equation 11 then provides multiplicative interactions between μ_2, and the weighted sum of μ_1s. In this sense, it captures nonconsecutive bigram features.

At a first glance, Equations 11 and 12 resemble recurrent convolutional neural networks (RCNN; Lei et al., 2016). RCNN is inspired by nonconsecutive n-gram features and low rank tensor factorization. It is later studied from a string kernel perspective (Lei et al., 2017a). Here we review its nonlinear bigram version:

$$\mathbf{c}_t^{(1)} = \mathbf{c}_{t-1}^{(1)} \odot \boldsymbol{\lambda}_t + \mathbf{u}_t^{(1)}, \tag{14a}$$

$$\mathbf{c}_t^{(2)} = \mathbf{c}_{t-1}^{(2)} \odot \boldsymbol{\lambda}_t + \mathbf{c}_{t-1}^{(1)} \odot \mathbf{u}_t^{(2)}, \tag{14b}$$

where the $\mathbf{u}_t^{(j)}$s are computed similarly to Equation 5b, and $\mathbf{c}_t^{(2)}$ is used as output for onward computation. Different strategies to computing $\boldsymbol{\lambda}_t$ were explored (Lei et al., 2015, 2016). When $\boldsymbol{\lambda}_t$ is a constant, or depends only on x_t, e.g., $\boldsymbol{\lambda}_t = \boldsymbol{\sigma}(\mathbf{W}_\lambda \mathbf{v}_t + \mathbf{b}_\lambda)$, the ith dimension of Equations 14 can be recovered from Equation 11, by letting

$$\mu_j(x_t) = [\mathbf{u}_t^{(j)}]_i,\ \phi_j(x_t) = [\boldsymbol{\lambda}_t]_i,\ j = 1, 2. \tag{15}$$

It is straightforward to generalize the above discussion to higher order cases: n-gram RCNN corresponds to WFSAs with $n+1$ states, constructed similarly to how we build $\mathcal{C}$ from $\mathcal{B}$ (Appendix B).

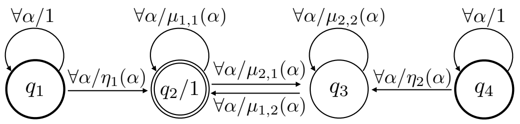

Figure 3: WFSA $\mathcal{D}_1$ discussed in §4.3. Two initial states q_1 and q_4 are used here.

Proposition 12. *For a single-layer RCNN with $\boldsymbol{\lambda}_t$ being a constant or depending only on x_t, the recurrence is rational.*

As noted later in §4.3, its recurrence may not be rational when $\boldsymbol{\lambda}_t = \boldsymbol{\sigma}(\mathbf{W}_c\mathbf{c}_{t-1} + \mathbf{W}_\lambda\mathbf{v}_t + \mathbf{b}_\lambda)$.

4.3 Beyond Elementwise Operations

So far we have discussed rational recurrences for models using elementwise recurrent updates (e.g., Equation 5c). This section uses an existing model as an example, to study a rational recurrence that is not elementwise. We focus on the input switched affine network (ISAN; Foerster et al., 2017). Aiming for efficiency and interpretability, it does not use any explicit nonlinearity; its affine transformation parameters depend only on the input:

$$\mathbf{c}_t = \mathbf{W}_{x_t}\mathbf{c}_{t-1} + \mathbf{b}_{x_t}. \tag{16}$$

Due to the matrix multiplication, the recurrence of a single-layer ISAN is not elementwise. Yet, we argue that it is rational. We will sketch the proof for a 2-dimensional case, and it is straightforward to generalize to higher dimensions (Appendix C).

We define two WFSAs, each recovering one dimension of ISAN's recurrent updates. Figure 3 diagrams one of them, $\mathcal{D}_1$. The other one, $\mathcal{D}_2$, is identical (including shared weights), except using q_3 instead of q_2 as the final state. For any nonempty input sequence $\mathbf{x} \in \Sigma^+$, the scores assigned by $\mathcal{D}_1$ and $\mathcal{D}_2$ can be inductively computed by applying the Forward algorithm. Letting $\mathcal{D}_1[\![\mathbf{x}_{:0}]\!] = \mathcal{D}_2[\![\mathbf{x}_{:0}]\!] = 0$, for $t \geq 1$

$$\begin{bmatrix}\mathcal{D}_1[\![\mathbf{x}_{:t}]\!]\\ \mathcal{D}_2[\![\mathbf{x}_{:t}]\!]\end{bmatrix} = \widetilde{\mathbf{W}}_{x_t}\begin{bmatrix}\mathcal{D}_1[\![\mathbf{x}_{:t-1}]\!]\\ \mathcal{D}_2[\![\mathbf{x}_{:t-1}]\!]\end{bmatrix} + \widetilde{\mathbf{b}}_{x_t}, \tag{17}$$

where

$$\widetilde{\mathbf{W}}_{x_t} = \begin{bmatrix} \mu_{1,1}(x_t) & \mu_{1,2}(x_t) \\ \mu_{2,1}(x_t) & \mu_{2,2}(x_t) \end{bmatrix}, \qquad (18)$$
$$\widetilde{\mathbf{b}}_{x_t} = \begin{bmatrix} \eta_1(x_t) \\ \eta_2(x_t) \end{bmatrix}.$$

Then Equation 16, in the case of hidden size 2, is recovered by letting $\mathbf{W}_{x_t} = \widetilde{\mathbf{W}}_{x_t}$ and $\mathbf{b}_{x_t} = \widetilde{\mathbf{b}}_{x_t}$.

Proposition 13. *The recurrence of a single-layer ISAN is rational.*

Corollary 14. *For a single-layer Elman network, in the absence of any nonlinearity, the recurrence is rational.*

Discussion. It is known that an Elman network can approximate any recursively computable partial function (Siegelmann and Sontag, 1995). On the other hand, in their single-layer cases, WFSAs (and thus models with rational recurrences) are restricted to rational series (Schützenberger, 1961). Therefore, we hypothesize that models like Elman networks, LSTMs, and GRUs, where the recurrences depend on previous states through affine transformations followed by nonlinearities, are not rational.

This work does not intend to propose rational recurrences as a concept general enough to include most existing RNNs. Rather, we wish to study a more constrained class of methods to better understand the connections between WFSAs and RNNs. Therefore in Definition 8, we restrict the semirings to be "simple," in the sense that both operations take constant time and space. Such a restriction aims to exclude the possibility of hiding arbitrarily complex computations inside the semiring, which might allow RNNs to satisfy the definition in a trivial and unilluminating way.

Such theoretical limitations might be less severe than they appear, since it is not yet entirely clear what they correspond to in practice, especially when multiple vertical layers of these models are used (Leshno and Schocken, 1993). We defer to future work the further study of the connections between WFSAs and Elman-style RNNs.

Closing this section, Table 1 summarizes the discussed recurrent neural architectures and their corresponding WFSAs.

5 Deriving Neural Models from WFSAs

Rational recurrences provide a new view of several recently proposed neural models. Based on such observations, this section aims to explore potential approaches to designing neural architectures in a more interpretable and intuitive way: by deriving them from WFSAs. §5.1 studies an interpolation of unigram and bigram features by combining 2-state and 3-state WFSAs (Figures 1 and 2). We then explore alternative semirings (§5.2), an approach orthogonal to what we've discussed so far.

	Models	Recurrence Function	WFSA
§4.1	SRU, SCRN T-RNN, QRNN	$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{u}_t$	$\mathcal{B}$
§4.2	RCNN	$\mathbf{c}_t^{(1)} = \mathbf{c}_{t-1}^{(1)} \odot \boldsymbol{\lambda}_t + \mathbf{u}_t^{(1)}$ $\mathbf{c}_t^{(2)} = \mathbf{c}_{t-1}^{(2)} \odot \boldsymbol{\lambda}_t + \mathbf{c}_{t-1}^{(1)} \odot \mathbf{u}_t^{(2)}$	$\mathcal{C}$
§4.3	ISAN	$\mathbf{c}_t = \mathbf{W}_{x_t}\mathbf{c}_{t-1} + \mathbf{b}_{x_t}$	$\mathcal{D}_1, \mathcal{D}_2$

Table 1: Recurrent neural network architectures discussed in §4 and their corresponding WFSAs. §4.1: SRU (Lei et al., 2017b), SCRN (Mikolov et al., 2014), T-RNN and its gated variants (Balduzzi and Ghifary, 2016), and QRNN (Bradbury et al., 2017); §4.2: RCNN (Lei et al., 2016); §4.3: ISAN (Foerster et al., 2017).

We note that our goal is not to devise new state-of-the-art architectures. Rather, we illustrate a new design process for neural architectures that draws inspiration from WFSAs. That said, in our experiments (§6), one of our new architectures performs as well as or better than strong baselines.

5.1 Aggregating Different Length Patterns

We start by presenting a straightforward extension to 2-state and 3-state rational models: one combining both. It is inspired by many classical NLP models, where unigram features and higher-order ones are interpolated.

Figure 4 diagrams a 4-state WFSA $\mathcal{F}$. Compared to $\mathcal{C}$ (Figure 2), $\mathcal{F}$ uses q_1 as a second final state, aiming to capture both unigram and bigram patterns, since a path is allowed to stop at q_1 after consuming one input. The final states are weighted by ρ_1 and ρ_2 respectively. Another notable modification is the additional state q_3, which is used to create a "shortcut" to reach q_2, together with an ε-transition. Specifically, starting from q_0, a path can now take the ε-transition and reach q_3, and then take a transition with weight μ_2 to reach q_2. Recall from §2, that ε-transitions do not consume any input, yet they can still be weighted by a (parameterized) function γ *not* depending on the inputs. The ε-transition allows for skipping the

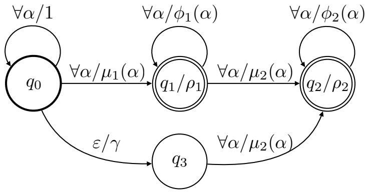

Figure 4: A WFSA $\mathcal{F}$ that combines both unigram and bigram features (§5.1). Two final states q_1 and q_2 are used, with weights ρ_1 and ρ_2, respectively.

first word in a bigram. It can be discouraged by using $\gamma \in (0, 1)$, just as we do in our experiments.

Deriving the neural architecture. As in §3, we relate hidden states of an RNN to the scores assigned by WFSAs to input strings. We then derive the neural architecture with a dynamic program. Here we keep the discussion self-contained by explicitly overviewing the procedure. It is a direct application of the Forward algorithm (§2), though now in a form that deals with the ε-transition. Such an approach applies, of course, to more general cases, as noted by Schwartz et al. (2018).

Given an input string $\mathbf{x} \in \Sigma^+$, let $z_t^{(j)}$ denote the total score of all paths landing in state q_j *just* after consuming x_t. Let $z_0^{(j)} = 0$, then for $t \geq 1$,

$$\begin{aligned}
z_t^{(0)} &= 1\\
z_t^{(1)} &= z_{t-1}^{(1)}\,\phi_1(x_t) + z_{t-1}^{(0)}\,\mu_1(x_t)\\
z_t^{(3)} &= z_t^{(0)}\,\gamma\\
z_t^{(2)} &= z_{t-1}^{(2)}\,\phi_2(x_t) + (z_{t-1}^{(1)} + z_{t-1}^{(3)})\,\mu_2(x_t)\\
\mathcal{F}[\![\mathbf{x}_{:t}]\!] &= \rho_1\, z_t^{(1)} + \rho_2\, z_t^{(2)}.
\end{aligned}$$

We now collect d of these WFSAs to construct an RNN, and we parameterize their weight functions with the technique we've been using:

$$\mathbf{c}_t^{(1)} = \mathbf{c}_{t-1}^{(1)} \odot \mathbf{f}_t^{(1)} + \mathbf{u}_t^{(1)}, \tag{19a}$$

$$\mathbf{c}_t^{(2)} = \mathbf{c}_{t-1}^{(2)} \odot \mathbf{f}_t^{(2)} + (\mathbf{c}_{t-1}^{(1)} + \mathbf{r}) \odot \mathbf{u}_t^{(2)}, \tag{19b}$$

$$\mathbf{c}_t = \mathbf{p}^{(1)} \odot \mathbf{c}_t^{(1)} + \mathbf{p}^{(2)} \odot \mathbf{c}_t^{(2)}, \tag{19c}$$

where

$$\mathbf{f}_t^{(j)} = \boldsymbol{\sigma}\big(\mathbf{W}_f^{(j)}\mathbf{v}_t + \mathbf{b}_f^{(j)}\big), \tag{20a}$$

$$\mathbf{u}_t^{(j)} = (\mathbf{1} - \mathbf{f}_t^{(j)}) \odot \boldsymbol{g}\big(\mathbf{W}_u^{(j)}\mathbf{v}_t + \mathbf{b}_u^{(j)}\big), \tag{20b}$$

$$\mathbf{p}^{(j)} = \boldsymbol{\sigma}(\mathbf{b}_p^{(j)}), \quad \mathbf{r} = \boldsymbol{\sigma}(\mathbf{b}_r). \tag{20c}$$

The $\mathbf{p}$ vectors correspond to the final state weights ρ_1 and ρ_2. Despite the similarities, $\mathbf{p}$ are different from output gates (Bradbury et al., 2017), since the former do not depend on the input, and are parameterized (through a sigmoid) by two leanred vectors $\mathbf{b}_p^{(j)}$. The same applies to $\mathbf{r}$ and $\mathbf{b}_r$, which correspond to the weights for ε-transitions γ.

Model	**Unigram**	**Bigram**	**Semiring**
RRNN($\mathcal{B}$)	✓		real
RRNN($\mathcal{B}$)$_{\text{m+}}$	✓		max-plus
RRNN($\mathcal{C}$)		✓	real
RRNN($\mathcal{F}$)	✓	✓	real

Table 2: Rational recurrent neural architectures compared in the experiments (§6.1).

5.2 Alternative Semirings

Our new understanding of rational recurrences allows us to consider a different kind of extension: replacing the semiring. We introduce an example, which modifies Example 6 by replacing its real (plus-times) semiring with the max-plus semiring $\langle \mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0\rangle$:

Example 15.

$$\mathbf{f}_t = \log \boldsymbol{\sigma}\big(\mathbf{W}_f\mathbf{v}_t + \mathbf{b}_f\big), \tag{21a}$$

$$\mathbf{u}_t = \boldsymbol{g}\big(\mathbf{W}_u\mathbf{v}_t + \mathbf{b}_u\big), \tag{21b}$$

$$\mathbf{c}_t = \max\{\mathbf{f}_t + \mathbf{c}_{t-1}, \mathbf{u}_t\}. \tag{21c}$$

Example 15 does not use the forget gate when computing $\mathbf{u}_t$ (Equation 21b), which is different from its plus-times counterpart, where $\mathbf{u}_t = (\mathbf{1} - \mathbf{f}_t) \odot \boldsymbol{g}\big(\mathbf{W}_u\mathbf{v}_t + \mathbf{b}_u\big)$. The reason is that, unlike the real semiring, the max-plus semiring lacks a well-defined negation. Possible alternatives include taking the log of a separate input gate, or using $\log(\mathbf{1} - \mathbf{f}_t)$, which we leave for future work.

Example 15 can be seen as replacing sum-pooling with max-pooling. Both max and sum-pooling have been used successfully in vision and NLP models. Intuitively, max-pooling "detects" the occurrence of a pattern while sum-pooling "counts" the occurrence of a pattern. One advantage of max operator is that the model's decisions can be back-traced and interpreted, as argued by Schwartz et al. (2018). Such a technique is applicable to all the models with rational recurrences.

6 Experiments

This section evaluates four rational RNNs on language modeling (§6.2) and text categorization (§6.3). Our goal is to compare the behaviors of models derived from different WFSAs, showing that our understanding of WFSAs allows us to improve existing rational models.

6.1 Compared Models

Our comparisons focus on the *recurrences* of the models, i.e., how the hidden states $\mathbf{c}_t$ are computed (e.g., Equations 5c and 19c). Therefore we follow Lei et al. (2017b) and use $\mathbf{u}_t^{(j)} = \mathbf{W}_u^{(j)} \mathbf{v}_t^{(j)}$ across all compared models, listed below and as well as in Table 2:

- RRNN($\mathcal{B}$), with real semiring (§4.1);
- RRNN($\mathcal{B}$)$_{\text{m+}}$, with max-plus semiring (§5.2);
- RRNN($\mathcal{C}$), with real semiring (§4.2);
- RRNN($\mathcal{F}$), with real semiring (§5.1).

We also compare to an LSTM baseline. Aiming to control for comfounding factors, we do not use highway connections in any of the models.[6] In the interest of space, the full architectures and hyperparameters are detailed in Appendices D and E.

6.2 Language Modeling

Dataset and implementation. We experiment with the Penn Treebank corpus (PTB; Marcus et al., 1993). We use the preprocessing and splits from Mikolov et al. (2010), resulting in a vocabulary size of 10K and 1M tokens.

Following standard practice, we treat the training data as one long sequence, split into mini batches, and train using BPTT truncated to 35 time steps (Williams and Peng, 1990). The input embeddings and output softmax weights are tied (Press and Wolf, 2017).

Results. Following Collins et al. (2017) and Melis et al. (2018), we compare models controlling for parameter budget. Table 3 summarizes language modeling perplexities on PTB test set. The middle block compares all models with two layers and 10M trainable parameters. RRNN($\mathcal{B}$) and RRNN($\mathcal{C}$) achieve roughly the same performance; interpolating both unigram and bigram features, RRNN($\mathcal{F}$) outperforms others by more than 2.9 test perplexity. For the three-layer and 24M setting (the bottom block), we observe similar trends, except that RRNN($\mathcal{C}$) slightly underperforms RRNN($\mathcal{B}$). Here RRNN($\mathcal{F}$) outperforms others by more than 2.1 perplexity.

Using a max-plus semiring, RRNN($\mathcal{B}$)$_{\text{m+}}$ *underperforms* RRNN($\mathcal{B}$) under both settings. Possible reasons could be the suboptimal design choice for computing input representations in the former (§5.2). Finally, most compared models outperform the LSTM baselines, whose numbers are taken from Lei et al. (2017b).[7]

Model	ℓ	# Params.	Dev.	Test
LSTM	2	24M	73.3	71.4
LSTM	3	24M	78.8	76.2
RRNN($\mathcal{B}$)	2	10M	73.1	69.2
RRNN($\mathcal{B}$)$_{\text{m+}}$	2	10M	75.1	71.7
RRNN($\mathcal{C}$)	2	10M	72.5	69.5
RRNN($\mathcal{F}$)	2	10M	**69.5**	**66.3**
RRNN($\mathcal{B}$)	3	24M	68.7	65.2
RRNN($\mathcal{B}$)$_{\text{m+}}$	3	24M	70.8	66.9
RRNN($\mathcal{C}$)	3	24M	70.0	67.0
RRNN($\mathcal{F}$)	3	24M	**66.0**	**63.1**

Table 3: Language modeling perplexity on PTB test set (lower is better). LSTM numbers are taken from Lei et al. (2017b). ℓ denotes the number of layers. Bold font indicates best performance.

Split	Amazon	SST	subj	CR
Train	20K	6.9K	8K	3.0K
Dev.	5K	0.9K	1K	0.4K
Test	25K	1.8K	1K	0.4K

Table 4: Number of instances in the text classification datasets (§6.3).

6.3 Text Classification

Implementation. We use unidirectional 2-layer architectures for all compared models. To build the classifiers, we feed the final RNN hidden states into a 2-layer tanh-MLP. Further implementation details are described in Appendix E.

Datasets. We experiment with four binary text classification datasets, described below.

- **Amazon** (electronic product review corpus; McAuley and Leskovec, 2013).[8] We focus on the positive and negative reviews.
- **SST** (Stanford sentiment treebank; Socher et al., 2013).[9] We focus on the binary classification task. SST provides labels for syntactic phrases; we experiment with a more realistic setup, and

---

[6] Thus RRNN($\mathcal{B}$) is essentially an SRU without highway connections. We denote it differently, to note its differences from the original implementation (Lei et al., 2017b). Similarly, we do not denote RRNN($\mathcal{C}$) as RCNN (Lei et al., 2016).

[7] Melis et al. (2018) point out that carefully tuning LSTMs can achieve much stronger performance, at the cost of exceptionally large amounts of computational resources for tuning.

[8] `http://riejohnson.com/cnn_data.html`

[9] `nlp.stanford.edu/sentiment/index.html`

Model	**Amazon**	**SST**	**subj**	**CR**
LSTM	$91.2_{\pm 0.3}$	$85.1_{\pm 0.6}$	$93.3_{\pm 0.6}$	$82.4_{\pm 1.5}$
RRNN($\mathcal{B}$)	$92.4_{\pm 0.1}$	$85.8_{\pm 0.3}$	$93.9_{\pm 0.4}$	$84.1_{\pm 1.0}$
RRNN($\mathcal{C}$)	$\mathbf{92.8}_{\pm 0.2}$	$84.8_{\pm 0.4}$	$93.8_{\pm 0.6}$	$84.5_{\pm 0.9}$
RRNN($\mathcal{B}$)$_{\text{m+}}$	$89.2_{\pm 3.1}$	$84.9_{\pm 0.4}$	$92.6_{\pm 0.5}$	$84.3_{\pm 0.5}$
RRNN($\mathcal{F}$)	$92.7_{\pm 0.2}$	$\mathbf{86.5}_{\pm 0.6}$	$\mathbf{94.8}_{\pm 0.5}$	$\mathbf{85.1}_{\pm 0.5}$

Table 5: Text classification test accuracy averaged over 5 runs. ± denotes standard deviation, and bold font indicates best averaged performance.

consider only complete sentences at either training or evaluating time.

- **subj** (Subjectivity dataset; Pang and Lee, 2004). As **subj** doesn't come with official splits, we randomly split it to train (80%), development (10%), and test (10%) sets.
- **CR** (customer reviews dataset; Hu and Liu, 2004).[10] As with **subj**, we randomly split this dataset using the same ratio.

Table 4 summarizes the sizes of the datasets.

Results. Table 5 summarizes text classification test accuracy. We report the average performance of 5 trials different only in random seeds. RRNN($\mathcal{F}$) outperforms all other models on 3 out of the 4 datasets. For Amazon, the largest one, we do not observe significant differences between RRNN($\mathcal{F}$) and RRNN($\mathcal{C}$), while both outperform others. This may suggest that the interpolation of unigram and bigram features by RRNN($\mathcal{F}$) is especially useful in small data setups. As in the language modeling experiments, RRNN($\mathcal{B}$)$_{\text{m+}}$ underperforms all other models in most cases, and in particular RRNN($\mathcal{B}$). These results provide evidence that replacing the real semiring in rational models might be challenging. We leave further exploration to future work.

7 Related Work

Weighted finite state automata. WFSAs were once popular among many sequential tasks (Mohri et al., 2002; Kumar and Byrne, 2003; Cortes et al., 2004; Pardo and Birmingham, 2005; Moore et al., 2006, *inter alia*), and are still successful in morphology (Dreyer, 2011; Cotterell et al., 2015; Rastogi et al., 2016, *inter alia*). Compared to neural networks, WFSAs are better understood theoretically and arguably more interpretable. They were recently revisited in combination with the former in, e.g., text generation (Ghazvininejad et al., 2016, 2017; Lin et al., 2017) and automatic music accompaniment (Forsyth, 2016).

Recurrent neural networks. RNNs (Elman, 1990; Jordan, 1989) prove to be strong models for sequential data (Siegelmann and Sontag, 1995). Besides the perhaps most notable gated variants (Hochreiter and Schmidhuber, 1997; Cho et al., 2014), extensive efforts have been devoted to developing alternatives (Balduzzi and Ghifary, 2016; Miao et al., 2016; Zoph and Le, 2017; Lee et al., 2017; Lei et al., 2017a; Vaswani et al., 2017; Gehring et al., 2017, *inter alia*). Departing from the above approaches, this work derives RNN architectures drawing inspiration from WFSAs.

Another line of work studied the connections between WFSAs and RNNs in terms of modeling capacity, both empirically (Kolen, 1993; Giles et al., 1992; Weiss et al., 2018, *inter alia*) and theoretically (Cleeremans et al., 1989; Visser et al., 2001; Chen et al., 2018, *inter alia*).

8 Conclusion

We presented *rational recurrences*, a new construction to study the recurrent updates in RNNs, drawing inspiration from WFSAs. We showed that rational recurrences are in frequent use by several recently proposed recurrent neural architectures, providing new understanding of them. Based on such connections, we discussed approaches to deriving novel neural architectures from WFSAs. Our empirical results demonstrate the potential of doing so. We publicly release our implementation at `https://github.com/Noahs-ARK/rational-recurrences`.

Acknowledgments

We thank Jason Eisner, Luheng He, Tao Lei, Omer Levy, members of the ARK lab at the University of Washington, and researchers at the Allen Institute for Artificial Intelligence for their helpful comments on an early version of this work, and the anonymous reviewers for their valuable feedback. We also thank members of the Aristo team at the Allen Institute for Artificial Intelligence for their support with the Beaker experimentation system. This work was supported in part by NSF grant IIS-1562364 and by the NVIDIA Corporation through the donation of a Tesla GPU.

---

[10] `http://www.cs.uic.edu/?liub/FBS/sentiment-analysis.html`

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

David Balduzzi and Muhammad Ghifary. 2016. Strongly-typed recurrent neural networks. In *Proc. of ICML*.

Leonard E. Baum and Ted Petrie. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563.

Jean Berstel, Jr. and Christophe Reutenauer. 1988. *Rational Series and Their Languages*. Springer-Verlag, Berlin, Heidelberg.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural network. In *Proc. of ICLR*.

Yining Chen, Sorcha Gilroy, Kevin Knight, and Jonathan May. 2018. Recurrent neural networks as weighted language recognizers. In *Proc. of NAACL*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of EMNLP*.

Axel Cleeremans, David Servan-Schreiber, and James L. McClelland. 1989. Finite state automata and simple recurrent networks. *Neural computation*, 1(3):372–381.

Jasmine Collins, Jascha Sohl-Dickstein, and David Sussillo. 2017. Capacity and trainability in recurrent neural networks. In *Proc. of ICLR*.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. 2004. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, 5:1035–1062.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015. Modeling word forms using latent underlying morphs and phonology. *TACL*, 3:433–447.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proc. of COLING*.

Markus Dreyer. 2011. *A Non-parametric Model for the Discovery of Inflectional Paradigms from Plain Text Using Graphical Models over Strings*. Ph.D. thesis, Johns Hopkins University.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Jakob N. Foerster, Justin Gilmer, Jan Chorowski, Jascha Sohl-Dickstein, and David Sussillo. 2017. Intelligible language modeling with input switched affine networks. In *Proc. of ICML*.

Jonathan P. Forsyth. 2016. *Automatic musical accompaniment using finite state machines*. Ph.D. thesis, New York University.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *Proc. of ICML*.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proc. of EMNLP*.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proc. of ACL, System Demonstrations*.

C. Lee Giles, Clifford B Miller, Dong Chen, Hsing-Hen Chen, Guo-Zheng Sun, and Yee-Chun Lee. 1992. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4(3):393–405.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *JAIR*, 57:345–420.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proc. of KDD*.

Michael I. Jordan. 1989. Serial order: A parallel, distributed processing approach. In *Advances in Connectionist Theory: Speech*. Erlbaum.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

John F. Kolen. 1993. Fool's gold: Extracting finite state machines from recurrent network dynamics. In *Proc. of NIPS*.

Werner Kuich and Arto Salomaa, editors. 1986. *Semirings, Automata, Languages*. Springer-Verlag.

Shankar Kumar and William Byrne. 2003. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proc. of NAACL*.

Yann LeCun. 1998. Gradient-based Learning Applied to Document Recognition. In *Proc. of the IEEE*.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. *arXiv:1705.07393*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding CNNs for text: non-linear, non-consecutive convolutions. In *Proc. of EMNLP*.

Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2017a. Deriving neural architectures from sequence and graph kernels. In *Proc. of ICML*.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez. 2016. Semi-supervised question retrieval with gated convolutions. In *Proc. of NAACL*.

Tao Lei, Yu Zhang, and Yoav Artzi. 2017b. Training RNNs as fast as CNNs. arXiv:1709.02755.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence. In *Proc. of EMNLP*.

Moshe Leshno and Shimon Schocken. 1993. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6:861–867.

Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. 2018. Independently recurrent neural network (IndRNN): Building A longer and deeper RNN. In *Proc. of CVPR*.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Proc. of NIPS*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proc. of RecSys*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *Proc. of ICLR*.

Yajie Miao, Jinyu Li, Yongqiang Wang, Shi-Xiong Zhang, and Yifan Gong. 2016. Simplifying long short-term memory acoustic models for fast training and decoding. In *Proc. of ICASSP*.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michaël Mathieu, and Marc'Aurelio Ranzato. 2014. Learning longer memory in recurrent neural networks. *arXiv:1412.7753*.

Tomas Mikolov, Martin Karafit, Luks Burget, Jan Cernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. of INTERSPEECH*.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Darren Moore, John Dines, Mathew Magimai-Doss, Jithendra Vepa, Octavian Cheng, and Thomas Hain. 2006. Juicer: A weighted finite-state transducer speech decoder. In *Proc. of MLMI*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. of ACL*.

Bryan Pardo and William Birmingham. 2005. Modeling form for on-line following of musical performances. In *Proc. of AAAI*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2018. Backpropagating through structured argmax using a spigot. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proc. of EACL*.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proc. of NAACL*.

Jacques Sakarovitch. 2009. Rational and recognisable power series. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, pages 105–174. Springer Berlin Heidelberg, Berlin, Heidelberg.

Marcel Paul Schützenberger. 1961. On the definition of a family of automata. *Information and Control*, 4(2-3):245–270.

Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. SoPa: Bridging CNNs, RNNs, and weighted finite-state machines. In *Proc. of ACL*.

Hava T. Siegelmann and Eduardo D. Sontag. 1995. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NIPS*.

Ingmar Visser, Maartje EJ Raijmakers, and Peter CM Molenaar. 2001. Hidden markov model interpretations of neural networks. In *Connectionist Models of Learning, Development and Evolution*, pages 197–206. Springer.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition. In *Proc. of ACL*.

Ronald J. Williams and Jing Peng. 1990. An efficient gradient-based algorithm for online training of recurrent network trajectories. *Neural computation*, 2(4):490–501.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv:1409.2329*.

Barret Zoph and Quoc V. Le. 2017. Neural architecture search with reinforcement learning. In *Proc. of ICLR*.

Efficient Contextualized Representation:
Language Model Pruning for Sequence Labeling

Liyuan Liu † **Xiang Ren** ♯ **Jingbo Shang** † **Xiaotao Gu** † **Jian Peng** † **Jiawei Han** †
† University of Illinois at Urbana-Champaign, Urbana, IL, USA
♯ University of Southern California, Los Angeles, CA, USA
†{ll2, shang7, xiaotao2, jianpeng, hanj}@illinois.edu ♯xiangren@usc.edu

Abstract

Many efforts have been made to facilitate natural language processing tasks with pre-trained language models (LMs), and brought significant improvements to various applications. To fully leverage the nearly unlimited corpora and capture linguistic information of multifarious levels, large-size LMs are required; but for a specific task, only parts of these information are useful. Such large-sized LMs, even in the inference stage, may cause heavy computation workloads, making them too time-consuming for large-scale applications. Here we propose to compress bulky LMs while preserving useful information with regard to a specific task. As different layers of the model keep different information, we develop a *layer selection* method for model pruning using sparsity-inducing regularization. By introducing the dense connectivity, we can detach any layer without affecting others, and stretch shallow and wide LMs to be *deep* and *narrow*. In model training, LMs are learned with *layer-wise dropouts* for better robustness. Experiments on two benchmark datasets demonstrate the effectiveness of our method.

1 Introduction

Benefited from the recent advances in neural networks (NNs) and the access to nearly unlimited corpora, neural language models are able to achieve a good perplexity score and generate high-quality sentences. These LMs automatically capture abundant linguistic information and patterns from large text corpora, and can be applied to facilitate a wide range of NLP applications (Rei, 2017; Liu et al., 2018; Peters et al., 2018).

Recently, efforts have been made on learning contextualized representations with pre-trained language models (LMs) (Peters et al., 2018). These pre-trained layers brought significant improvements to various NLP benchmarks, yielding up to 30% relative error reductions. However, due to high variability of language, gigantic NNs (e.g., LSTMs with 8,192 hidden states) are preferred to construct informative LMs and extract multifarious linguistic information (Peters et al., 2017). Even though these models can be integrated without retraining (using their forward pass only), they still result in heavy computation workloads during inference stage, making them prohibitive for real-world applications.

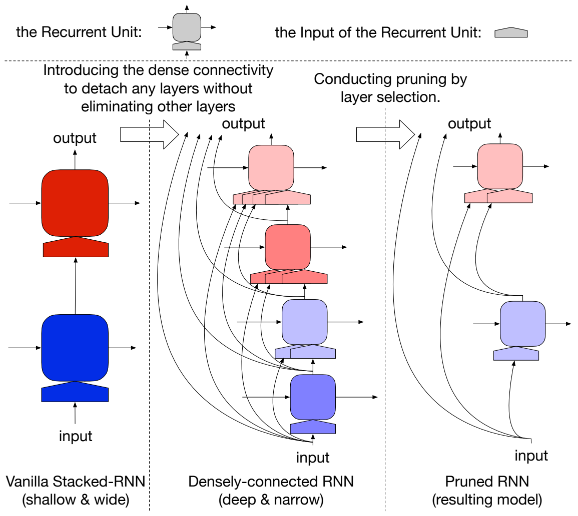

Figure 1: Leverage the dense connectivity to compress models via layer selection, and replace wide and shallow RNNs with deep and narrow ones.

In this paper, we aim to compress LMs for the end task in a plug-in-and-play manner. Typically, NN compression methods require the retraining of the whole model (Mellempudi et al., 2017). However, neural language models are usually composed of RNNs, and their backpropagations require significantly more RAM than their inference. It would become even more cumbersome when the target task equips the coupled LMs to capture information in both directions. Therefore, these methods do not fit our scenario very

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1215–1225
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

well. Accordingly, we try to compress LMs while *avoiding costly retraining*.

Intuitively, layers of different depths would capture linguistic information of different levels. Meanwhile, since LMs are trained in a task-agnostic manner, not all layers and their extracted information are relevant to the end task. Hence, we propose to compress the model by layer selection, which retains useful layers for the target task and prunes irrelevant ones. However, for the widely-used stacked-LSTM, directly pruning any layers will eliminate all subsequent ones. To overcome this challenge, we introduce the dense connectivity. As shown in Fig. 1, it allows us to detach any layers while keeping all remaining ones, thus creating the basis to avoid retraining. Moreover, such connectivity can stretch shallow and wide LMs to be deep and narrow (Huang et al., 2017), and enable a more fine-grained layer selection.

Furthermore, we try to retain the effectiveness of the pruned model. Specifically, we modify the L_1 regularization for encouraging the selection weights to be not only sparse but binary, which protects the retained layer connections from shrinkage. Besides, we design a layer-wise dropout to make LMs more robust and better prepared for the layer selection.

We refer to our model as LD-Net, since the *layer selection* and the *dense connectivity* form the basis of our pruning methods. For evaluation, we apply LD-Net on two sequence labeling benchmark datasets, and demonstrated the effectiveness of the proposed method. In the CoNLL03 Named Entity Recognition (NER) task, the F_1 score increases from 90.78±0.24% to 91.86±0.15% by integrating the unpruned LMs. Meanwhile, after pruning over 90% calculation workloads from the best performing model[1] (92.03%), the resulting model still yields 91.84±0.14%. Our implementations and pre-trained models would be released for futher study[2].

2 LD-Net

Given a input sequence of T word-level tokens, $\{x_1, x_2, \cdots, x_T\}$, we use $\mathbf{x}_t$ to denote the embedding of x_t. For a L-layers NN, we mark the input and output of the l^{th} layer at the t^{th} time stamp as $\mathbf{x}_{l,t}$ and $\mathbf{h}_{l,t}$.

2.1 RNN and Dense Connectivity

We represent one RNN layer as a function:

$$\mathbf{h}_{l,t} = F_l(\mathbf{x}_{l,t}, \mathbf{h}_{l,t-1}) \tag{1}$$

where F_l is the recurrent unit of l^{th} layer, it could be any RNNs variants, and the vanilla LSTMs is used in our experiments.

As deeper NNs usually have more representation power, RNN layers are often stacked together to form the final model by setting $\mathbf{x}_{l,t} = \mathbf{h}_{l-1,t}$. These vanilla stacked-RNN models, however, suffer from problems like the vanishing gradient, and it's hard to train very deep models.

Recently, the dense connectivity and residual connectivity have been proposed to handle these problems (He et al., 2016; Huang et al., 2017). Specifically, dense connectivity refers to adding direct connections from any layer to all its subsequent layers. As illustrated in Figure 1, the input of l^{th} layer is composed of the original input and the output of all preceding layers as follows.

$$\mathbf{x}_{l,t} = [\mathbf{x}_t, \mathbf{h}_{1,t}, \cdots, \mathbf{h}_{l-1,t}]$$

Similarly, the final output of the L-layer RNN is $\mathbf{h}_t = [\mathbf{x}_t, \mathbf{h}_{1,t}, \cdots, \mathbf{h}_{L,t}]$. With dense connectivity, we can detach any single layer without eliminating its subsequent layers (as in Fig. 1). Also, existing practices in computer vision demonstrate that such connectivities can lead to deep and narrow NNs and distribute parameters into different layers. Moreover, different layers in LMs usually capture linguistic information of different levels. Hence, we can compress LMs for a specific task by pruning unrelated or unimportant layers.

2.2 Language Modeling

Language modeling aims to describe the sequence generation. Normally, the generation probability of the sequence $\{x_1, \cdots, x_T\}$ is defined in a "forward" manner:

$$p(x_1, \cdots, x_T) = \prod_{t=1}^{T} p(x_t|x_1, \cdots, x_{t-1}) \tag{2}$$

Where $p(x_t|x_1, \cdots, x_{t-1})$ is computed based on the output of RNN, $\mathbf{h}_t$. Due to the dense connectivity, $\mathbf{h}_t$ is composed of outputs from different layers, which are designed to capture linguistic information of different levels. Similar to the bottleneck layers employed in the DenseNet (Huang et al., 2017), we add additional layers to unify such

---

[1] Based on their performance on the development sets

[2] `https://github.com/LiyuanLucasLiu/LD-Net`.

information. Accordingly, we add an projection layer with the ReLU activation function:

$$\mathbf{h}_t^* = \mathrm{ReLU}(W_{proj} \cdot \mathbf{h}_t + \mathbf{b}_{proj}) \tag{3}$$

Based on $\mathbf{h}_t^*$, it's intuitive to calculate $p(x_t|x_1, \cdots, x_{t-1})$ by the softmax function, i.e., $\mathrm{softmax}(W_{out} \cdot \mathbf{h}_t^* + \mathbf{b})$.

Since the training of language models needs nothing but the raw text, it has almost unlimited corpora. However, conducting training on extensive corpora results in a huge dictionary, and makes calculating the vanilla softmax intractable. Several techniques have been proposed to handle this problem, including adaptive softmax (Grave et al., 2017), slim word embedding (Li et al., 2018), the sampled softmax and the noise contrastive estimation (Józefowicz et al., 2016). Since the major focus of our paper does not lie in the language modeling task, we choose the adaptive softmax because of its practical efficiency when accelerated with GPUs.

2.3 Contextualized Representations

As pre-trained LMs can describe the text generation accurately, they can be utilized to extract information and construct features for other tasks. These features, referred as contextualized representations, have been demonstrated to be essentially useful (Peters et al., 2018). To capture information from both directions, we utilized not only forward LMs, but also backward LMs. Backward LMs are based on Eqn. 4 instead of Eqn. 2. Similar to forward LMs, backward LMs approach $p(x_t|x_{t+1}, \cdots, x_T)$ with NNs. For reference, the output of the RNN in backward LMs for x_t is recorded as $\mathbf{h}_t^r$.

$$p(x_1, \cdots, x_n) = \prod_{t=1}^{T} p(x_t|x_{t+1}, \cdots, x_T) \tag{4}$$

Ideally, the final output of LMs (e.g., $\mathbf{h}_t^*$) would be the same as the representation of the target word (e.g., x_{t+1}); therefore, it may not contain much context information. Meanwhile, the output of the densely connected RNN (e.g., $\mathbf{h}_t$) includes outputs from every layer, thus summarizing all extracted features. Since the dimensions of $\mathbf{h}_t$ could be too large for the end task, we add a non-linear transformation to calculate the contextualized representation ($\mathbf{r}_t$):

$$\mathbf{r}_t = \mathrm{ReLU}(W_{cr} \cdot [\mathbf{h}_t, \mathbf{h}_t^r] + \mathbf{b}_{cr}) \tag{5}$$

Our proposed method bears the same intuition as the ELMo (Peters et al., 2018). ELMo is designed for the vanilla stacked-RNN, and tries to calculate a weighted average of different layers' outputs as the contextualized representation. Our method, benefited from the dense connectivity and its narrow structure, can directly combine the outputs of different layers by concatenation. It does not assume the outputs of different layers to be in the same vector space, thus having more potential for transferring the constructed token representations. More discussions are available in Sec. 4.

2.4 Layer Selection

Typical model compression methods require retraining or gradient calculation. For the coupled LMs, these methods require even more computation resources compared to the training of LMs, thus not fitting our scenario very well.

Benefited from the dense connectivity, we are able to train deep and narrow networks. Moreover, we can detach one of its layer without eliminating all subsequent layers (as in Fig. 1). Since different layers in NNs could capture different linguistic information, only a few of them would be relevant or useful for a specific task. As a result, we try to compress these models by the task-guided layer selection. For i-th layer, we introduce a binary mask $z_i \in \{0, 1\}$ and calculate $\mathbf{h}_{l,t}$ with Eqn. 6 instead of Eqn. 1.

$$\mathbf{h}_{l,t} = z_i \cdot F_l(\mathbf{x}_{l,t}, \mathbf{h}_{l,t-1}) \tag{6}$$

With this setting, we can conduct a layer selection by optimizing the regularized empirical risk:

$$\min \mathcal{L} + \lambda_0 \cdot \mathcal{R} \tag{7}$$

where $\mathcal{L}$ is the empirical risk for the sequence labeling task and $\mathcal{R}$ is the sparse regularization.

The ideal choice for $\mathcal{R}$ would be the L_0 regularization of $\mathbf{z}$, i.e., $\mathcal{R}_0(\mathbf{z}) = |\mathbf{z}|_0$. However, it is not continuous and cannot be efficiently optimized. Hence, we relax z_i from binary to a real value (i.e., $0 \leq z_i \leq 1$) and replace $\mathcal{R}_0$ by:

$$\mathcal{R}_1 = |\mathbf{z}|_1$$

Despite the sparsity achieved by $\mathcal{R}_1$, it could hurt the performance by shifting all z_i far away from 1. Such shrinkage introduces additional noise in $\mathbf{h}_{l,t}$ and $\mathbf{x}_{l,t}$, which may result in ineffective pruned LMs. Since our goal is to conduct

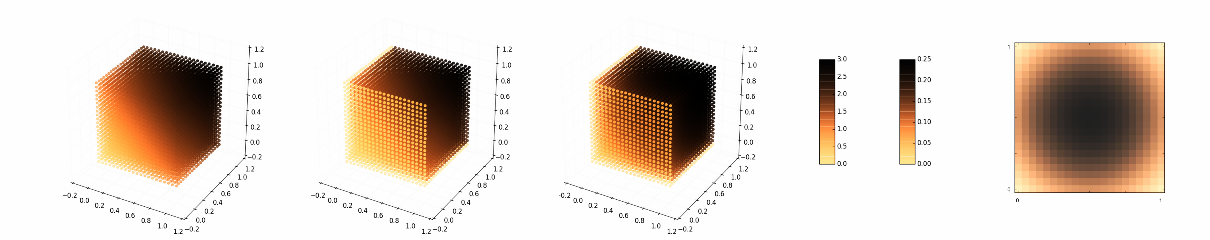

$\mathcal{R}_1(\mathbf{z})$ $\mathcal{R}_2(\mathbf{z})$ $\mathcal{R}_3(\mathbf{z})$ $\mathcal{R}_3(\mathbf{z})$ with $z_3 = 0$

Figure 2: Penalty values of various $\mathcal{R}$ for $\mathbf{z}$ with three dimensions. λ_1 has been set to 2 for $\mathcal{R}_2$ and $\mathcal{R}_3$.

pruning without retraining, we further modify the L_1 regularization to achieve sparsity while alleviating its shrinkage effect. As the target of $\mathcal{R}$ is to make $\mathbf{z}$ sparse, it can be "turned-off" after achieving a satisfying sparsity. Therefore, we extend $\mathcal{R}_1$ to a margin-based regularization:

$$\mathcal{R}_2 = \delta(|\mathbf{z}|_0 > \lambda_1)|\mathbf{z}|_1$$

In addition, we also want to make up the relaxation made on $\mathbf{z}$, i.e., relaxing its values from binary to $[0, 1]$. Accordingly, we add the penalty $|\mathbf{z}(1 - \mathbf{z})|_1$ to encourage $\mathbf{z}$ to be binary (Murray and Ng, 2010) and modify $\mathcal{R}_2$ into $\mathcal{R}_3$:

$$\mathcal{R}_3 = \delta(|\mathbf{z}|_0 > \lambda_1)|\mathbf{z}|_1 + |\mathbf{z}(1 - \mathbf{z})|_1$$

To compare $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$, we visualize their penalty values in Fig. 2. The visualization is generated for a 3-dimensional $\mathbf{z}$ while the targeted sparsity, λ_1, is set to 2. Comparing to $\mathcal{R}_1$, we can observe that $\mathcal{R}_2$ enlarges the optimal point set from $\mathbf{0}$ to all $\mathbf{z}$ with a satisfying sparsity, thus avoiding the over-shrinkage. To better demonstrate the effect of $\mathcal{R}_3$, we further visualize its penalties after achieving a satisfying sparsity (w.l.o.g., assuming $z_3 = 0$). One can observe that it penalizes non-binary $\mathbf{z}$ and favors binary values.

2.5 Layer-wise Dropout

So far, we've customized the regularization term for the layer-wise pruning, which protects the retained connections among layers from shrinking. After that, we try to further retain the effectiveness of the compressed model. Specifically, we choose to prepare the LMs for the pruned inputs, thus making them more robust to pruning.

Accordingly, we conduct the training of LMs with a layer-wise dropout. As in Figure 3, a random part of layers in the LMs are randomly dropped during each batch. The outputs of the dropped layers will not be passed to their subsequent recurrent layers, but will be sent to the projection layer (Eqn. 3) for predicting the next word. In other words, this dropout is only applied to the input of recurrent layers, which aims to imitate the pruned input without totally removing any layers.

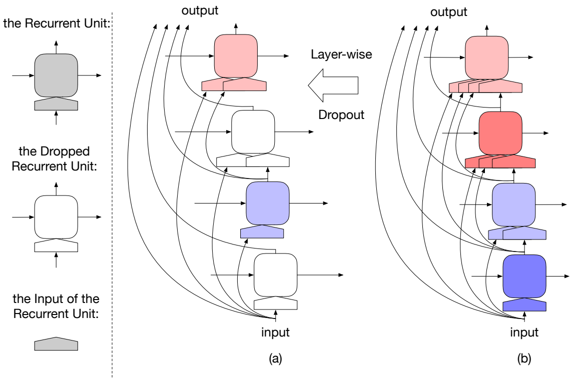

Figure 3: Layer-wise dropout conducted on a 4-layer densely connected RNN. (a) is the remained RNN. (b) is the original densely connected RNN.

3 Sequence Labeling

In this section, we will introduce our sequence labeling architecture, which is augmented with the contextualized representations.

3.1 Neural Architecture

Following the recent studies (Liu et al., 2018; Kuru et al., 2016), we construct the neural architecture as in Fig. 4. Given the input sequence $\{x_1, x_2, \cdots, x_T\}$, for t^{th} token (x_t), we assume its word embedding is $\mathbf{w}_t$, its label is y_t, and its character-level input is $\{c_{i,1}, c_{i,2}, \cdots, c_{i,\_}\}$, where $c_{i,\_}$ is the space character following x_t.

The character-level representations have become the required components for most of the state-of-the-art. Following the recent study (Liu et al., 2018), we employ LSTMs to take the character-level input in a context-aware manner, and mark its output for x_t as $\mathbf{c}_t$. Similar to the contextualized representation, $\mathbf{c}_t$ usually has more dimensions than $\mathbf{w}_t$. To integrate them together, we set the output dimension of Eqn. 5 as the dimension of $\mathbf{w}_t$, and project $\mathbf{c}_t$ to a new space with the same dimension number. We mark the projected character-level representation as $\mathbf{c}^*_t$.

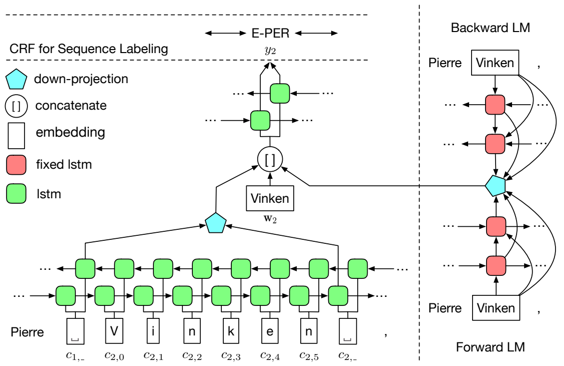

Figure 4: The proposed sequence labeling architecture with contextualized representations.

After projections, these vectors are concatenated as $\mathbf{v}_t = [\mathbf{c}_t^*; \mathbf{r}_t; \mathbf{w}_t], \forall i \in [1, T]$ and further fed into the word-level LSTMs. We refer to their output as $\mathbf{U} = \{\mathbf{u}_1, \cdots, \mathbf{u}_T\}$. To ensure the model to predict valid label sequences, we append a first-order conditional random field (CRF) layer to the model (Lample et al., 2016). Specifically, the model defines the generation probability of $\mathbf{y} = \{y_1, \cdots, y_T\}$ as

$$p(\mathbf{y}|\mathbf{U}) = \frac{\prod_{t=1}^{T} \phi(y_{t-1}, y_t, \mathbf{u}_t)}{\sum_{\hat{\mathbf{y}} \in \mathbf{Y}(\mathbf{U})} \prod_{t=1}^{T} \phi(\hat{y}_{t-1}, \hat{y}_t, \mathbf{u}_t)} \quad (8)$$

where $\hat{\mathbf{y}} = \{\hat{y}_1, \ldots, \hat{y}_T\}$ is a generic label sequence, $\mathbf{Y}(\mathbf{U})$ is the set of all generic label sequences for $\mathbf{U}$ and $\phi(y_{t-1}, y_t, \mathbf{u}_t)$ is the potential function. In our model, $\phi(y_{t-1}, y_t, \mathbf{u}_t)$ is defined as $\exp(W_{y_t}\mathbf{u}_t + b_{y_{t-1},y_t})$, where W_{y_t} and b_{y_{t-1},y_t} are the weight and bias parameters.

3.2 Model Training and Inference

We use the following negative log-likelihood as the empirical risk.

$$\mathcal{L} = -\sum_{\mathbf{U}} \log p(\mathbf{y}|\mathbf{U}) \quad (9)$$

For testing or decoding, we want to find the optimal sequence $\mathbf{y}^*$ that maximizes the likelihood.

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathbf{Y}(\mathbf{U})}{\text{argmax}}\ p(\mathbf{y}|\mathbf{U}) \quad (10)$$

Although the denominator of Eq. 8 is complicated, we can calculate Eqs. 9 and 10 efficiently by the Viterbi algorithm.

For optimization, we decompose it into two steps, i.e., model training and model pruning.

Model training. We set λ_0 to 0 and optimize the empirical risk without any regularization, i.e., $\min \mathcal{L}$. In this step, we conduct optimization with the stochastic gradient descent with momentum. Following (Peters et al., 2018), dropout would be added to both the coupled LMs and the sequence labeling model.

Model pruning. We conduct the pruning based on the checkpoint which has the best performance on the development set during the model training. We set λ_0 to non-zero values and optimize $\min \mathcal{L} + \lambda_0 \mathcal{R}_3$ by the projected gradient descent with momentum. Any layer i with $z_i = 0$ would be deleted in the final model to complete the pruning. To get a better stability, dropout is only added to the sequence labeling model.

4 Experiments

We will first discuss the capability of the LD-Net as language models, then explore the effectiveness of its contextualized representations.

4.1 Language Modeling

For comparison, we conducted experiments on the one billion word benchmark dataset (Chelba et al., 2013) with both LD-Net (with 1,600 dimensional projection) and the vanilla stacked-LSTM. Both kinds of models use word embedding (random initialized) of 300 dimension as input and use the adaptive softmax (with default setting) as an approximation of the full softmax. Additionally, as preprocessing, we replace all tokens occurring equal or less than 3 times with as UNK, which shrinks the dictionary from 0.79M to 0.64M.

The optimization is performed by the Adam algorithm (Kingma and Ba, 2014), the gradient is clipped at 5.0 and the learning rate is set to start from 0.001. The layer-wise dropout ratio is set to 0.5, the RNNs are unrolled for 20 steps without resetting the LSTM states, and the batch size is set to 128. Their performances are summarized in Table 1, together with several LMs used in our sequence labeling baselines. For models without official reported parameter numbers, we estimate their values (marked with†) by assuming they adopted the vanilla LSTM. Note that, for models 3, 5, 6, 7, 8, and 9, PPL refers to the averaged perplexity of the forward and the backward LMs.

We can observe that, for those models taking word embedding as the input, embedding composes the vast majority of model parameters. However, embedding can be embodied as a "sparse" layer which is computationally efficient. Instead, the intense calculations are conducted in

Network	Ind. #	Hid. #	Layer #	Param.# ($\cdot 10^7$)		PPL
				RNN	Others	
8192-1024 (Józefowicz et al., 2016)	1	8192	2	15.1$^\sharp$	163$^\sharp$	30.6
CNN-8192-1024 (Józefowicz et al., 2016)	2	8192	2	15.1$^\sharp$	89$^\sharp$	30.0
CNN-4096-512 (Peters et al., 2018)	3	4096	2	3.8$^\sharp$	40.6$^\sharp$	39.7
2048-512 (Peters et al., 2017)	4	2048	1	0.9$^\sharp$	40.6$^\sharp$	47.50
2048-Adaptive (Grave et al., 2017)	5	2048	2	5.2†	26.5†	39.8
vanilla LSTM	6	2048	2	5.3†	25.6†	40.27
	7	1600	2	3.2†	24.2†	48.85
LD-Net without Layer-wise Dropout	8	300	10	2.3†	24.2†	45.14
LD-Net with Layer-wise Dropout	9	300	10	2.3†	24.2†	50.06

Table 1: Performance comparison of language models. Models marked with† adopted adaptive softmax and the vanilla LSTMs, which has less softmax parameters. Models marked with$^\sharp$ employed sampled softmax LSTMs w. projection, which results in less RNN parameters w.r.t. the size of hidden states.

RNN layers and softmax layer for language modeling, or RNN layers for contextualized representations. At the same time, comparing the model 8192-1024 and CNN-8192-1024, their only difference is the input method. Instead of taking word embedding as the input, CNN-8192-1024 utilizes CNN to compose word representation from the character-level input. Despite the greatly reduced parameter number, the perplexity of the resulting models remains almost unchanged. Since replacing embedding layer with CNN would make the training slower, we only conduct experiments with models taking word embedding as the input.

Comparing LD-Net with other baselines, we think it achieves satisfactory performance with regard to the size of hidden states. It demonstrates the LD-Net's capability of capturing the underlying structure of natural language. Meanwhile, we find that the layer-wise dropout makes it harder to train LD-Net and its resulting model achieves less competitive results. However, as would be discussed in the next section, layer-wise dropout allows the resulting model to generate better contextualized representations and be more robust to pruning, even with a higher perplexity.

4.2 Sequence Labeling

Following TagLM (Peters et al., 2017), we evaluate our methods in two benchmark datasets, the CoNLL03 NER task (Tjong Kim Sang and De Meulder, 2003) and the CoNLL00 Chunking task (Tjong Kim Sang and Buchholz, 2000).

CoNLL03 NER has four entity types and includes the standard training, development and test sets.

CoNLL00 chunking defines eleven syntactic chunk types (e.g., `NP` and `VP`) in addition to `Other`. Since it only includes training and test sets, we sampled 1000 sentences from training set as a held-out development set (Peters et al., 2017).

In both cases, we use the BIOES labeling scheme (Ratinov and Roth, 2009) and use the micro-averaged F_1 as the evaluation metric. Based on the analysis conducted in the development set, we set $\lambda_0 = 0.05$ for the NER task, and $\lambda_0 = 0.5$ for the Chunking task. As discussed before, we conduct optimization with the stochastic gradient descent with momentum. We set the batch size, the momentum, and the learning rate to 10, 0.9, and $\eta_t = \frac{\eta_0}{1+\rho t}$ respectively. Here, $\eta_0 = 0.015$ is the initial learning rate and $\rho = 0.05$ is the decay ratio. Dropout is applied in our model, and its ratio is set to 0.5. For a better stability, we use gradient clipping of 5.0. Furthermore, we employ the early stopping in the development set and report averaged score across five different runs.

Regarding the network structure, we use the 30-dimension character-level embedding. Both character-level and word-level RNNs are set to one-layer LSTMs with 150-dimension hidden states in each direction. The GloVe 100-dimension pre-trained word embedding[3] is used as the initialization of word embedding $\mathbf{w}_t$, and will be fine-tuned during the training. The layer selection variables z_i are initialized as 1, remained unchanged

---

[3] `https://nlp.stanford.edu/projects/glove/`

Network (LMs Ind.#)	Avg. ppl	#FLOPs $(\cdot 10^6)$	F_1 score (avg±std)
NoLM (/)	/	3	94.42±0.08
R-ELMo (6)	40.27	108	96.19±0.07
R-ELMo (7)	48.85	68	95.86±0.04
LD-Net * (8)	45.14	51	96.01±0.07
LD-Net * (9)	50.06	51	96.05±0.08
LD-Net (8)	origin	51	96.13
	pruned	13	95.46±0.18
LD-Net (9)	origin	51	96.15
	pruned	10	95.66±0.04

Table 2: Performance comparisons in the CoNLL00 Chunking task. LD-Net maked with * are trained without pruning (layer selection).

during the model training and only be updated during the model pruning. All other variables are randomly initialized (Glorot and Bengio, 2010).

Compared methods. The first baseline, referred as NoLM, is our sequence labeling model without the contextualized representations, i.e., calculating $\mathbf{v}_t$ as $[\mathbf{c}_t^*; \mathbf{w}_t]$ instead of $[\mathbf{c}_t^*; \mathbf{r}_t; \mathbf{w}_t]$. Besides, ELMo (Peters et al., 2018) is the major baseline. To make comparison more fair, we implemented the ELMo model and use it to calculate the $\mathbf{r}_t$ in Eqn. 5 instead of $[\mathbf{h}_t, \mathbf{h}_t^r]$. Results of re-implemented models are referred with R-ELMo (λ is set to the recommended value, 0.1) and the results reported in its original paper are referred with O-ELMo. Additionally, since TagLM (Peters et al., 2017) with one-layer NNs can be viewed as a special case of ELMo, we also include its results.

Sequence labeling results. Table 2 and 3 summarizes the results of LD-Net and baselines. Besides the F_1 score and averaged perplexity, we also estimate FLOPs (i.e., the number of floating-point multiplication-adds) for the efficiency evaluation. Since our model takes both word-level and character-level inputs, we estimated the FLOPs value for a word-level input with 4.39 character-level inputs, while 4.39 is the averaged length of words in the CoNLL03 dataset.

Before the model pruning, LD-Net achieves a 96.05±0.08 F_1 score in the CoNLL00 Chunking task, yielding nearly 30% error reductions over the NoLM baseline. Also, it scores 91.86±0.15 F_1 in the CoNLL03 NER task with over 10% error reductions. Similar to the language modeling, we observe that the most complicated models achieve the best perplexity and provide the most improvements in the target task. Still, considering the number of model parameters and the resulting perplexity, our model demonstrates its effectiveness in generating contextualized representations. For example, comparing to our methods, R-ELMo (7) leverages LMs with the similar perplexity and parameter number, but cannot get the same improvements with our method on both datasets.

Network (LMs Ind.#)	Avg. ppl	#FLOPs $(\cdot 10^6)$	F_1 score (avg±std)
NoLM (/)	/	3	90.78±0.24
O-ELMo (3)	39.70	$79^\sharp$	92.22±0.10
TagLM (4)	47.50	$22^\sharp$	91.62±0.23
R-ELMo (6)	40.27	108	91.99±0.24
R-ELMo (7)	48.85	68	91.54±0.10
LD-Net * (8)	45.14	98	91.76±0.18
LD-Net * (9)	50.06	98	91.86±0.15
LD-Net (8)	origin	51	91.95
	pruned	5	91.55±0.06
LD-Net (9)	origin	51	92.03
	pruned	5	91.84±0.14

Table 3: Performance comparison in the CoNLL03 NER task. Models marked with† employed LSTMs with projection, which is more efficient than the vanilla LSTMs. LD-Net maked with * are trained without pruning (layer selection).

Actually, contextualized representations have strong connections with the skip-thought vectors (Kiros et al., 2015). Skip-thought models try to embed sentences and are trained by predicting the previous and afterwards sentences. Similarly, LMs encode a specific context as the hidden states of RNNs, and use them to predict future contexts. Specifically, we recognize the cell states of LSTMs are more like to be the sentence embedding (Radford et al., 2017), since they are only passed to the next time stamps. At the same time, because the hidden states would be passed to other layers, we think they are more like to be the token representations capturing necessary signals for predicting the next word or updating context representations[4]. Hence, LD-Net should be more

---

[4] We tried to combine the cell states with the hidden states to construct the contextualized representations by concatenation or weighted average, but failed to get better performance.

Network (LMs Ind.#)	FLOPs	Batch size	Peak RAM	Time (s)	Speed	
					10^3words/s	10^3sents/s
R-ELMo (6)	108	200	8Gb	32.88	22	0.4
LD-Net (9, origin)	51	80	8Gb	25.68	26	0.5
LD-Net (9, pruned)	5	80	4Gb	6.90	98	2.0
		500	8Gb	4.86 (**5X**)	166 (**6X**)	2.9 (**5X**)

Table 4: Speed comparison in the CoNLL03 NER task. We can observe that LD-Net (9, pruned) achieved about 5 times speed up on the wall-clock time over LD-Net (9, origin).

effective then ELMo, as concatenating could preserve all extracted signals while weighted average might cause information loss.

Although the layer-wise dropout makes the model harder to train, their resulting LMs generate better contextualized representations, even without the same perplexity. Also, as discussed in (Peters et al., 2018, 2017), the performance of the contextualized representation can be further improved by training larger models or using the CNN to represent words.

For the pruning, we started from the model with the best performance on the development set (referred with "origin"), and refer the performances of pruned models with "pruned" in Table 2 and 3. Essentially, we can observe the pruned models get rid of the vast majority of calculation while still retaining a significant improvement. We will discuss more on the pruned models in Sec. 4.4.

4.3 Speed Up Measurements

We use FLOPs for measuring the inference efficiency as it reflects the time complexity (Han et al., 2015), and thus is independent of specific implementations. For models with the same structure, improvements in FLOPs would result in monotonically decreasing inference time. However, it may not reflect the actual efficiency of models due to the model differences in parallelism. Accordingly, we also tested wall-clock speeds of our implementations.

Our implementations are based on the PyTorch 0.3.1[5], and all experiments are conducted on the CoNLL03 dataset with the Nvidia GTX 1080 GPU. Specifically, due to the limited size of CoNLL03 test set, we measure such speeds on the training set. As in Table 4, we can observe that, the pruned model achieved about 5 times speed up. Although there is still a large margin between the actual speed-up and the FLOPs speed-up, we think the resulting decode speed (166K words/s) is sufficient for most real-world applications.

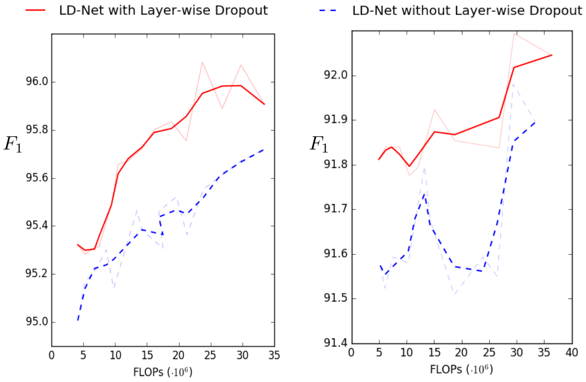

(a) ConNLL00 Chunking

(b) ConNLL03 NER

Figure 5: The performance of pruned models in two tasks w.r.t. their efficiency (FLOPs).

4.4 Case Studies

Effect of the pruning ratio. To explore the effect of the pruning ratio, we adjust λ_1 and visualize the performance of pruned models v.s. their FLOPs # in Fig 5. We can observe that LD-Net outperforms its variants and demonstrates its effectiveness.

As the pruning ratio becoming larger, we can observe the performance of LD-Net first increases a little, then starts dropping. Besides, in the CoNLL03 NER task, LMs can be pruned to a relatively small size without much loss of efficiency. As in Table 3, we can observe that, after pruning over 90% calculations, the error of the resulting model only increases about 2%, yielding a competitive performance. As for the CoNLL00 Chunking task, the performance of LD-Net decays in a faster rate than that in the NER task. For example, after pruning over 80% calculations, the error of the resulting model increases about 13%. Considering the fact that this corpus is only half the size of the CoNLL03 NER dataset, we can expect the resulting models have more dependencies on the LMs. Still, the pruned model achieves a 25% error reduction over the NoLM baseline.

We think it implies that ELMo works as token representations instead of sentence representations

[5] `http://pytorch.org/`

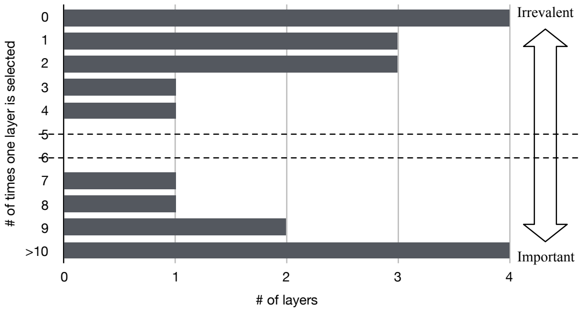

Figure 6: The performance of pruned models in two tasks w.r.t. their efficiency (FLOPs).

Layer selection pattern. We further studied the layer selection patterns. Specifically, we use the same setting of LD-Net (9) in Table 3, conduct model pruning using for 50 times, and summarize the statics in Figure 6. We can observe that network layers formulate two clear clusters, one is likely to be preserved during the selection, and the other is likely to be pruned. This is consistent with our intuition that some layers are more important than others and the layer selection algorithm would pick up layers meaningfully.

However, there is some randomness in the selection result. We conjugate that large networks trained with dropout can be viewed as a ensemble of small sub-networks (Hara et al., 2016), also there would be several sub-networks having the similar function. Accordingly, we think the randomness mainly comes from such redundancy.

Effectiveness of model pruning. Zhu and Gupta (2017) observed pruned large models consistently outperform small models on various tasks (including LM). These observations are consistent with our experiments. For example, LD-Net achieves 91.84 after pruning on the CoNLL03 dataset. It outperforms TagLM (4) and R-ELMo (7), whose performances are 91.62 and 91.54. Besides, we trained small LMs of the same size as the pruned LMs (1-layer densely connected LSTMs). Its perplexity is 69 and its performance on the CoNLL03 dataset is 91.55 ± 0.19.

5 Related Work

Sequence labeling. Linguistic sequence labeling is one of the fundamental tasks in NLP, encompassing various applications including POS tagging, chunking, and NER. Many attempts have been made to conduct end-to-end learning and build reliable models without handcrafted features (Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016).

Language modeling. Language modeling is a core task in NLP. Many attempts have been paid to develop better neural language models (Zilly et al., 2017; Inan et al., 2016; Godin et al., 2017; Melis et al., 2017). Specifically, with extensive corpora, language models can be well trained to generate high-quality sentences from scratch (Józefowicz et al., 2016; Grave et al., 2017; Li et al., 2018; Shazeer et al., 2017). Meanwhile, initial attempts have been made to improve the performance of other tasks with these methods. Some methods treat the language modeling as an additional supervision, and conduct co-training for knowledge transfer (Dai and Le, 2015; Liu et al., 2018; Rei, 2017). Others, including this paper, aim to construct additional features (referred as contextualized representations) with the pre-trained language models (Peters et al., 2017, 2018).

Neural Network Acceleration. There are mainly three kinds of NN acceleration methods, i.e., prune network into smaller sizes (Han et al., 2015; Wen et al., 2016), converting float operation into customized low precision arithmetic (Hubara et al., 2018; Courbariaux et al., 2016), and using shallower networks to mimic the output of deeper ones (Hinton et al., 2015; Romero et al., 2014). However, most of them require costly retraining.

6 Conclusion

Here, we proposed LD-Net, a novel framework for efficient contextualized representation. As demonstrated on two benchmarks, it can conduct the layer-wise pruning for a specific task. Moreover, it requires neither the gradient oracle of LMs nor the costly retraining. In the future, we plan to apply LD-Net to other applications.

Acknowledgments

We thank Junliang Guo and all reviewers for their constructive comments. Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA). The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*.

Fréderic Godin, Joni Dambre, and Wesley De Neve. 2017. Improving language modeling using densely connected recurrent neural networks. *arXiv preprint arXiv:1707.06130*.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for gpus. In *International Conference on Machine Learning*, pages 1302–1310.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143.

Kazuyuki Hara, Daisuke Saitoh, and Hayaru Shouno. 2016. Analysis of dropout learning regarded as ensemble learning. In *International Conference on Artificial Neural Networks*, pages 72–79. Springer.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *CVPR*.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2018. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR*, abs/1611.01462.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921.

Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*.

Zhongliang Li, Raymond Kulhanek, Shaojun Wang, Yunxin Zhao, and Shuang Wu. 2018. Slim embedding layers for recurrent neural language models. *CoRR*, abs/1711.09873.

Liyuan Liu, Jingbo Shang, Frank F. Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. *CoRR*, abs/1709.04109.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *CoRR*, abs/1707.05589.

Naveen Mellempudi, Abhisek Kundu, Dheevatsa Mudigere, Dipankar Das, Bharat Kaul, and Pradeep Dubey. 2017. Ternary neural networks with fine-grained quantization. *CoRR*, abs/1705.01462.

Walter Murray and Kien-Ming Ng. 2010. An algorithm for nonlinear optimization problems with binary variables. *Computational Optimization and Applications*, 47(2):257–288.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *ACL*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Learning language in logic and CoNLL*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Natural language learning at NAACL-HLT*.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *CoRR*, abs/1710.01878.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutnk, and Jrgen Schmidhuber. 2017. Recurrent Highway Networks. In *ICML*.

Automatic Event Salience Identification

Zhengzhong Liu **Chenyan Xiong** **Teruko Mitamura** **Eduard Hovy**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
`{liu, cx, teruko, hovy}@cs.cmu.edu`

Abstract

Identifying the salience (i.e. importance) of discourse units is an important task in language understanding. While events play important roles in text documents, little research exists on analyzing their saliency status. This paper empirically studies the *Event Salience* task and proposes two salience detection models based on content similarities and discourse relations. The first is a feature based salience model that incorporates similarities among discourse units. The second is a neural model that captures more complex relations between discourse units. Tested on our new large-scale event salience corpus, both methods significantly outperform the strong frequency baseline, while our neural model further improves the feature based one by a large margin. Our analyses demonstrate that our neural model captures interesting connections between salience and discourse unit relations (e.g., scripts and frame structures).

1 Introduction

Automatic extraction of prominent information from text has always been a core problem in language research. While traditional methods mostly concentrate on the word level, researchers start to analyze higher-level discourse units in text, such as entities (Dunietz and Gillick, 2014) and events (Choubey et al., 2018).

Events are important discourse units that form the backbone of our communication. They play various roles in documents. Some are more central in discourse: connecting other entities and events, or providing key information of a story. Others are less relevant, but not easily identifiable by NLP systems. Hence it is important to be able to quantify the "importance" of events. For example, Figure 1 is a news excerpt describing a debate around a jurisdiction process: "*trial*" is central as the main discussing topic, while "*war*" is not.

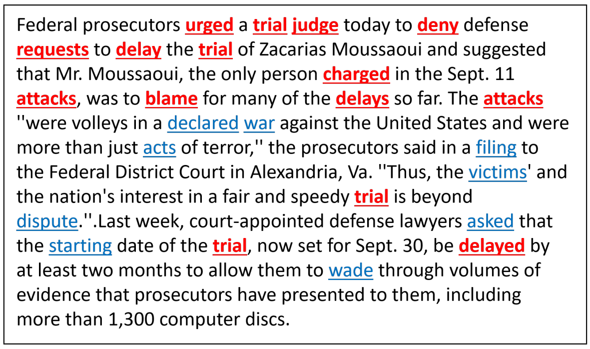

Federal prosecutors **urged** a **trial** **judge** today to **deny** defense **requests** to **delay** the **trial** of Zacarias Moussaoui and suggested that Mr. Moussaoui, the only person **charged** in the Sept. 11 **attacks**, was to **blame** for many of the **delays** so far. The **attacks** ''were volleys in a declared war against the United States and were more than just acts of terror,'' the prosecutors said in a filing to the Federal District Court in Alexandria, Va. ''Thus, the victims' and the nation's interest in a fair and speedy **trial** is beyond dispute.''.Last week, court-appointed defense lawyers asked that the starting date of the **trial**, now set for Sept. 30, be **delayed** by at least two months to allow them to wade through volumes of evidence that prosecutors have presented to them, including more than 1,300 computer discs.

Figure 1: Examples annotations. Underlying words are annotated event triggers; the red bold ones are annotated as salient.

Researchers are aware of the need to identify central events in applications like detecting salient relations (Zhang et al., 2015), and identifying climax in storyline (Vossen and Caselli, 2015). Generally, the salience of discourse units is important for language understanding tasks, such as document analysis (Barzilay and Lapata, 2008), information retrieval (Xiong et al., 2018), and semantic role labeling (Cheng and Erk, 2018). Thus, proper models for finding important events are desired.

In this work, we study the task of **event salience detection**, to find events that are most relevant to the main content of documents. To build a salience detection model, one core observation is that salient discourse units are forming discourse relations. In Figure 1, the "*trial*" event is connected to many other events: "*charge*" is pressed before "*trial*"; "*trial*" is being "*delayed*".

We present two salience detection systems based on the observations. First is a feature based learning to rank model. Beyond basic features like frequency and discourse location, we design features using cosine similarities among events and entities, to estimate the *content organization* (Grimes, 1975): how lexical meaning of elements relates to each other. Similarities from within-sentence or across the whole document are used to capture

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1226–1236
Brussels, Belgium, October 31 - November 4, 2018.

interactions on both local and global aspects (§4). The model significantly outperforms a strong "Frequency" baseline in our experiments.

However, there are other discourse relations beyond lexical similarity. Figure 1 showcases some: the script relation (Schank and Abelson, 1977)[1] between "*charge*" and "*trial*", and the frame relation (Baker et al., 1998) between "*attacks*" and "*trial*" ("*attacks*" fills the "charges" role of "*trial*"). Since it is unclear which ones contribute more to salience, we design a Kernel based Centrality Estimation (`KCE`) model (§5) to capture salient specific interactions between discourse units automatically.

In `KCE`, discourse units are projected to embeddings, which are trained end-to-end towards the salience task to capture rich semantic information. A set of soft-count kernels are trained to weigh salient specific latent relations between discourse units. With the capacity to model richer relations, `KCE` outperforms the feature-based model by a large margin (§7.1). Our analysis shows that `KCE` is exploiting several relations between discourse units: including script and frames (Table 5). To further understand the nature of `KCE`, we conduct an *intrusion test* (§6.2), which requires a model to identify events from another document. The test shows salient events form tightly related groups with relations captured by `KCE`.

The notion of salience is subjective and may vary from person to person. We follow the empirical approaches used in entity salience research (Dunietz and Gillick, 2014). We consider the *summarization test*: an event is considered salient if a summary written by a human is likely to include it, since events about the main content are more likely to appear in a summary. This approach allows us to create a large-scale corpus (§3).

In this paper, we make three main contributions. First, we present two event salience detection systems, which capture rich relations among discourse units. Second, we observe interesting connections between salience and various discourse relations (§7.1 and Table 5), implying potential research on these areas. Finally, we construct a large scale event salience corpus, providing a testbed for future research. Our code, dataset and models are publicly available[2].

2 Related Work

Events have been studied on many aspects due to their importance in language. To name a few: event detection (Li et al., 2013; Nguyen and Grishman, 2015; Peng et al., 2016), coreference (Liu et al., 2014; Lu and Ng, 2017), temporal analysis (Do et al., 2012; Chambers et al., 2014), sequencing (Araki et al., 2014), script induction (Chambers and Jurafsky, 2008; Balasubramanian et al., 2013; Rudinger et al., 2015; Pichotta and Mooney, 2016).

However, studies on event salience are premature. Some previous work attempts to approximate event salience with word frequency or discourse position (Vossen and Caselli, 2015; Zhang et al., 2015). Parallel to ours, Choubey et al. (2018) propose a task to find the most dominant event in news articles. They draw connections between event coreference and importance, on hundreds of closed-domain documents, using several oracle event attributes. In contrast, our proposed models are fully learned and applied on more general domains and at a larger scale. We also do not restrict to a single most important event per document.

There is a small but growing line of work on entity salience (Dunietz and Gillick, 2014; Dojchinovski et al., 2016; Xiong et al., 2018; Ponza et al., 2018). In this work, we study the case for events.

Text relations have been studied in tasks like text summarization, which mainly focused on cohesion (Halliday and Hasan, 1976). Grammatical cohesion methods make use of document level structures such as anaphora relations (Baldwin and Morton, 1998) and discourse parse trees (Marcu, 1999). Lexical cohesion based methods focus on repetitions and synonyms on the lexical level (Skorochod'ko, 1971; Morris and Hirst, 1991; Erkan and Radev, 2004). Though sharing similar intuitions, our proposed models are designed to learn richer semantic relations in the embedding space.

Comparing to the traditional summarization task, we focus on events, which are at a different granularity. Our experiments also unveil interesting phenomena among events and other discourse units.

3 The Event Salience Corpus

This section introduces our approach to construct a large-scale event salience corpus, including methods for finding event mentions and obtaining saliency labels. The studies are based on the Annotated New York Times corpus (Sandhaus, 2008), a newswire corpus with expert-written abstracts.

---

[1] Scripts are prototypical sequences of events: a *restaurant* script normally contains events like "order", "eat" and "pay".

[2] `https://github.com/hunterhector/EventSalience`

3.1 Automatic Corpus Creation

Event Mention Annotation: Despite many annotation attempts on events (Pustejovsky et al., 2002; Brown et al., 2017), automatic labeling of them in general domain remains an open problem. Most of the previous work follows empirical approaches. For example, Chambers and Jurafsky (2008) consider all verbs together with their subject and object as events. Do et al. (2011) additionally include nominal predicates, using the nominal form of verbs and lexical items under the *Event* frame in FrameNet (Baker et al., 1998).

There are two main challenges in labeling event mentions. First, we need to decide which lexical items are event triggers. Second, we have to disambiguate the word sense to correctly identify events. For example, the word "phone" can refer to an entity (a physical phone) or an event (a phone call event). We use FrameNet to solve these problems. We first use a FrameNet based parser: Semafor (Das and Smith, 2011), to find and disambiguate triggers into frame classes. We then use the FrameNet ontology to select event mentions.

Our frame based selection method follows the Vendler classes (Vendler, 1957), a four way classification of eventuality: *states*, *activities*, *accomplishments* and *achievements*. The last three classes involve state change, and are normally considered as events. Following this, we create an "event-evoking frame" list using the following procedure:

1. We keep frames that are subframes of *Event* and *Process* in the FrameNet ontology.
2. We discard frames that are subframes of state, entity and attribute frames, such as *Entity*, *Attributes*, *Locale*, etc.
3. We manually inspect frames that are not subframes of the above-mentioned ones (around 200) to keep event related ones (including subframes), such as *Arson*, *Delivery*, etc.

This gives us a total of 569 frames. We parse the documents with Semafor and consider predicates that trigger a frame in the list as candidates. We finish the process by removing the light verbs[3] and reporting events[4] from the candidates, similar to previous research (Recasens et al., 2013).

Salience Labeling: For all articles with a human written abstract (around 664,911) in the New York Times Annotated Corpus, we extract event mentions. We then label an event mention as salient if we can find its lemma in the corresponding abstract (Mitamura et al. (2015) showed that lemma matching is a strong baseline for event coreference.). For example, in Figure 1, event mentions in bold and red are found in the abstract, thus labeled as salient. Data split is detailed in Table 1 and §6.

	Train	Dev	Test
# Documents	526126	64000	63589
Avg. # Word	794.12	790.27	798.68
Avg. # Events	61.96	60.65	61.34
Avg. # Entities	197.63	196.95	198.40
Avg. # Salience	8.77	8.79	8.90

Table 1: Dataset Statistics.

3.2 Annotation Quality

While the automatic method enables us to create a dataset at scale, it is important to understand the quality of the dataset. For this purpose, we have conducted two small manual evaluation study.

Our lemma-based salience annotation method is based on the assumption that lemma matching being a strong detector for event coreference. In order to validate this assumption, one of the authors manually examined 10 documents and identified 82 coreferential event mentions pairs between the text body and the abstract. The automatic lemma rule identifies 72 such pairs: 64 of these matches human decision, producing a precision of 88.9% (64/72) and a recall of 78% (64/82). There are 18 coreferential pairs missed by the rule.

The next question is: *is an event really important if it is mentioned in the abstract?* Although prior work (Dunietz and Gillick, 2014) shows that the assumption to be valid for entities, we study the case for events. We asked two annotators to manually annotate 10 documents (around 300 events) using a 5-point Likert scale for salience. We compute the agreement score using Cohen's Kappa (Cohen, 1960). We find the task to be challenging for human: annotators don't agree well on the 5-point scale (Cohens Kappa = 0.29). However, if we collapse the scale to binary decisions, the Kappa between the annotators raises to 0.67. Further, the Kappa between each annotator and automatic labels are 0.49 and 0.42 respectively. These agreement scores are also close to those reported in the entity salience tasks (Dunietz and Gillick, 2014).

---

[3] Light verbs carry little semantic information: "appear", "be", "become", "do", "have", "seem", "do", "get", "give", "go", "have", "keep", "make", "put", "set", "take".

[4] Reporting verbs are normally associated with the narrator: "argue", "claim", "say", "suggest", "tell".

While errors exist in the automatic annotation process inevitably, we find the error rate to be reasonable for a large-scale dataset. Further, our study indicates the difficulties for human to rate on a finer scale of salience. We leave the investigation of continuous salience scores to future work.

4 Feature-Based Event Salience Model

This section presents the feature-based model, including the features and the learning process.

4.1 Features

Our features are summarized in Table 2.

Basic Discourse Features: We first use two basic features similar to Dunietz and Gillick (2014): *Frequency* and *Sentence Location*. *Frequency* is the lemma count of the mention's syntactic head word (Manning et al., 2014). *Sentence Location* is the sentence index of the mention, since the first few sentences are normally more important. These two features are often used to estimate salience (Barzilay and Lapata, 2008; Vossen and Caselli, 2015).

Content Features: We then design several lexical similarity features, to reflect Grimes' content relatedness (Grimes, 1975). In addition to events, the relations between events and entities are also important. For example, Figure 1 shows some related entities in the legal domain, such as "*prosecutors*" and "*court*". Ideally, they should help promote the salience status for event "*trial*".

Lexical relations can be found both within-sentence (local) or across sentence (global) (Halliday and Hasan, 1976). We compute the local part by averaging similarity scores from other units in the same sentence. The global part is computed by averaging similarity scores from other units in the document. All similarity scores are computed using cosine similarities on pre-trained embeddings (Mikolov et al., 2013).

These lead to 3 content features: *Event Voting*, the average similarity to other events in the document; *Entity Voting*, the average similarity to entities in the document; *Local Entity Voting*, the average similarity to entities in the same sentence. Local event voting is not used since a sentence often contains only 1 event.

4.2 Model

A Learning to Rank (LeToR) model (Liu, 2009) is used to combine the features. Let ev_i denote the ith event in a document d. Its salience score is computed as:

$$f(ev_i, d) = W_f \cdot F(ev_i, d) + b \qquad (1)$$

where $F(ev_i, d)$ is the features for ev_i in d (Table 2); W_f and b are the parameters to learn.

The model is trained with pairwise loss:

$$\sum_{ev^+, ev^- \in d} \max(0, 1 - f(ev^+, d) + f(ev^-, d)), \qquad (2)$$
$$\text{w.r.t. } y(ev^+, d) = +1 \ \& \ y(ev^-, d) = -1.$$

$$y(e_i, d) = \begin{cases} +1, & \text{if } e_i \text{ is a salient entity in } d, \\ -1, & \text{otherwise.} \end{cases}$$

where ev^+ and ev^- represent the salient and non-salient events; y is the gold standard function. Learning can be done by standard gradient methods.

5 Neural Event Salience Model

As discussed in §1, the salience of discourse units is reflected by rich relations beyond lexical similarities, for example, script ("*charge*" and "*trial*") and frame (a "*trial*" of "*attacks*"). The relations between these words are specific to the salience task, thus difficult to be captured by raw cosine scores that are optimized for word similarities. In this section, we present a neural model to exploit the embedding space more effectively, in order to capture relations for event salience estimation.

5.1 Kernel-based Centrality Estimation

Inspired by the kernel ranking model (Xiong et al., 2017), we propose Kernel-based Centrality Estimation (KCE), to find and weight semantic relations of interests, in order to better estimate salience.

Formally, given a document d, the set of annotated events $\mathbb{V} = \{ev_1, \ldots ev_i \ldots, ev_n\}$, KCE first embed an event into vector space: $ev_i \xrightarrow{Emb} \overrightarrow{ev_i}$. The embedding function is initialized with pre-trained embeddings. It then extract K features for each ev_i:

$$\Phi_K(ev_i, \mathbb{V}) = \{\phi_1(\overrightarrow{ev_i}, \mathbb{V}), \ldots, \phi_k(\overrightarrow{ev_i}, \mathbb{V}), \ldots, \phi_K(\overrightarrow{ev_i}, \mathbb{V})\}, \qquad (3)$$

$$\phi_k(\overrightarrow{ev_i}, \mathbb{V}) = \sum_{ev_j \in \mathbb{V}} \exp\left(-\frac{(\cos(\overrightarrow{ev_i}, \overrightarrow{ev_j}) - \mu_k)^2}{2\sigma_k^2}\right). \qquad (4)$$

Name	Description
`Frequency`	The frequency of the event lemma in document.
`Sentence Location`	The location of the first sentence that contains the event.
`Event Voting`	Average cosine similarity with other events in document.
`Entity Voting`	Average cosine similarity with other entities in document.
`Local Entity Voting`	Average cosine similarity with entities in the sentence.

Table 2: Event Salience Features.

$\phi_k(\overrightarrow{ev_i}, \mathbb{V})$ is the k-th Gaussian kernel with mean μ_k and variance σ_k^2. It models the interactions between events in its kernel range defined by μ_k and σ_k. $\Phi_K(ev_i, \mathbb{V})$ enforces multi-level interactions among events — relations that contribute similarly to salience are expected to be grouped into the same kernels. Such interactions greatly improve the capacity of the model with negligible increase in the number of parameters. Empirical evidences (Xiong et al., 2017) have shown that kernels in this form are effective to learn weights for task-specific term pairs.

The final salience score is computed as:

$$f(ev_i, d) = W_v \cdot \Phi_K(ev_i, \mathbb{V}) + b, \tag{5}$$

where W_v is learned to weight the contribution of the certain relations captured by each kernel.

We then use the exact same learning objective as in equation (2). The pairwise loss is first backpropagated through the network to update the kernel weights W_v, assigning higher weights to relevant regions. Then the kernels use the gradients to update the embeddings, in order to capture the meaningful discourse relations for salience.

Since the features and `KCE` capture different aspects, combining them may give superior performance. This can be done by combining the two vectors in the final linear layer:

$$f(ev_i, d) = W_v \cdot \Phi_K(ev_i, \mathbb{V}) + W_f \cdot F(ev_i, d) + b \tag{6}$$

5.2 Integrating Entities into KCE

`KCE` is also used to model the relations between events and entities. For example, in Figure 1, the entity "*court*" is a frame element of the event "*trial*"; "*United States*" is a frame element of the event "*war*". It is not clear which pair contributes more to salience. We again let `KCE` to learn it.

Formally, let $\mathbb{E}$ be the list of entities in the document, i.e. $\mathbb{E} = \{en_1, \dots, en_i, \dots, en_n\}$, where en_i is the ith entity in document d. `KCE` extracts the kernel features about entity-event relations as follows:

$$\begin{aligned}\Phi_K(ev_i, \mathbb{E}) = \{&\phi_1(\overrightarrow{ev_i}, \mathbb{E}), \dots, \\ &\phi_k(\overrightarrow{ev_i}, \mathbb{E}), \dots, \phi_K(\overrightarrow{ev_i}, \mathbb{E})\},\end{aligned} \tag{7}$$

$$\phi_k(\overrightarrow{ev_i}, \mathbb{E}) = \sum_{en_j \in \mathbb{E}} \exp\left(-\frac{\left(\cos(\overrightarrow{ev_i}, \overrightarrow{en_j}) - \mu_k\right)^2}{2\sigma_k^2}\right) \tag{8}$$

similarly, en_i is embedded by: $en_i \xrightarrow{Emb} \overrightarrow{en_i}$, which is initialized by pre-trained entity embeddings.

We reach the full `KCE` model by combining all the vectors using a linear layer:

$$\begin{aligned}f(ev_i, d) = &W_e \cdot \Phi_K(ev_i, \mathbb{E}) + W_v \cdot \Phi_K(ev_i, \mathbb{V}) \\ &+ W_f \cdot F(ev_i, d) + b\end{aligned} \tag{9}$$

The model is again trained by equation (2).

6 Experimental Methodology

This section describes our experiment settings.

6.1 Event Salience Detection

Dataset: We conduct our experiments on the salience corpus described in §3. Among the 664,911 articles with abstracts, we sample 10% of the data as the test set and then randomly leave out another 10% documents for development. Overall, there are 4359 distinct event lexical items, at a similar scale with previous work (Chambers and Jurafsky, 2008; Do et al., 2011). The corpus statistics are summarized in Table 1.

Input: The inputs to models are the documents and the extracted events. The models are required to rank the events from the most to least salience.

Baselines: Three methods from previous researches are used as baselines: *Frequency*, *Location* and *PageRank*. The first two are often used to simulate saliency (Barzilay and Lapata, 2008; Vossen and Caselli, 2015). The *Frequency* baseline ranks events based on the count of the headword lemma; the *Location* baseline ranks events using the order of their appearances in discourse. Ties are broken randomly.

Similar to entity salience ranking with PageRank scores (Xiong et al., 2018), our *PageRank* baseline runs PageRank on a fully connected graph whose nodes are the events in documents. The edges are weighted by the embedding similarities between event pairs. We conduct supervised PageRank on this graph, using the same pairwise loss setup as in `KCE`. We report the best performance obtained by linearly combining *Frequency* with the scores obtained after a one-step random walk.

Evaluation Metric: Since the importance of events is on a continuous scale, the boundary between "important" and "not important" is vague. Hence we evaluate it as a ranking problem. The metrics are the precision and recall value at 1, 5 and 10 respectively. It is adequate to stop at 10 since there are less than 9 salient events per document on average (Table 1). We also report Area Under Curve (AUC). Statistical significance values are tested by permutation (randomization) test with $p < 0.05$.

Implementation Details: We pre-trained word embeddings with 128 dimensions on the whole Annotated New York Times corpus using Word2Vec (Mikolov et al., 2013). Entities are extracted using the TagMe entity linking toolkit (Ferragina and Scaiella, 2010). Words or entities that appear only once in training are replaced with special "unknown" tokens.

The hyper-parameters of the `KCE` kernels follow previous literature (Xiong et al., 2017). There is one exact match kernel ($\mu = 1, \sigma = 1e^{-3}$) and ten soft-match kernels evenly distributed between $(-1, 1)$, i.e. $\mu \in \{-0.9, -0.7, \ldots, 0.9\}$, with the same $\sigma = 0.1$.

The parameters of the models are optimized by Adam (Kingma and Ba, 2015), with batch size 128. The vectors of entities are initialized by the pre-trained embeddings. Event embeddings are initialized by their headword embedding.

6.2 The Event Intrusion Test: A Study

`KCE` is designed to estimate salience by modeling relations between discourse units. To better understand its behavior, we design the following **event intrusion test**, following the word intrusion test used to assess topic model quality (Chang et al., 2009).

Event Intrusion Test: The test will present to a model a set of events, including: the **origins**, all events from one document; the **intruders**, some events from another document. Intuitively, if events inside a document are organized around the core content, a model capturing their relations well should easily identify the intruder(s).

Specifically, we take a bag of unordered events $\{O_1, O_2, \ldots, O_p\}$, from a document O, as the origins. We insert into it intruders, events drawn from another document, I: $\{I_1, I_2, \ldots, I_q\}$. We ask a model to rank the mixed event set $M = \{O_1, I_1, O_2, I_2, \ldots\}$. We expect a model to rank the intruders I_i below the origins O_i.

Intrusion Instances: From the development set, we randomly sample 15,000 origin and intruding document pairs. To simplify the analysis, we only take documents with at least 5 salient events. The intruder events, together with the entities in the same sentences, are added to the origin document.

Metrics: AUC is used to quantify ranking quality, where events in O are positive and events in I are negative. To observe the ranking among the salient origins, we compute a separate AUC score between the intruders and the salient origins, denoted as SA-AUC. In other words, SA-AUC is the AUC score on the list with non-salient origins removed.

Experiments Details: We take the full `KCE` model to compute salient scores for events in the mixed event set M, which are directly used for ranking. Frequency is recounted. All other features (Table 2) are set to 0 to emphasize the relational aspects,

We experiment with two settings: 1. adding only the salient intruders. 2. adding only the non-salient intruders. Under both settings, the intruders are added one by one, allowing us to observe the score change regarding the number of intruders added. For comparison, we add a *Frequency* baseline, that directly ranks events by the Frequency feature.

7 Evaluation Results

This section presents the evaluations and analyses.

7.1 Event Salience Performance

We summarize the main results in Table 3.

Baselines: *Frequency* is the best performing baseline. Its precision at 1 and 5 are higher than 40%. *PageRank* performs worse than *Frequency* on all

Method	P@01		P@05		P@10		AUC	
`Location`	0.3555	–	0.3077	–	0.2505	–	0.5226	–
`PageRank`	0.3628	–	0.3438	–	0.3007	–	0.5866	–
`Frequency`	0.4542	–	0.4024	–	0.3445	–	0.5732	–
`LeToR`	0.4753†	+4.64%	0.4099†	+1.87%	0.3517†	+2.10%	0.6373†	+11.19%
`KCE (-EF)`	0.4420	−2.69%	0.4038	+0.34%	0.3464†	+0.54%	0.6089†	+6.23%
`KCE (-E)`	0.4861†‡	+7.01%	0.4227†‡	+5.04%	0.3603†‡	+4.58%	0.6541†‡	+14.12%
`KCE`	0.5049†‡	+11.14%	0.4277†‡	+6.29%	0.3638†‡	+5.61%	0.6557†‡	+14.41%

Method	R@01		R@05		R@10		W/T/L
`Location`	0.0807	–	0.2671	–	0.3792	–	–/–/–
`PageRank`	0.0758	–	0.2760	–	0.4163	–	–/–/–
`Frequency`	0.0792	–	0.2846	–	0.4270	–	–/–/–
`LeToR`	0.0836†	+5.61%	0.2980†	+4.70%	0.4454†	+4.31%	8037 / 48493 / 6770
`KCE (-EF)`	0.0714	−9.77%	0.2812	−1.18%	0.4321†	+1.20%	6936 / 48811 / 7553
`KCE (-E)`	0.0925†‡	+16.78%	0.3172†‡	+11.46%	0.4672†‡	+9.41%	11676 / 43294 / 8330
`KCE`	0.0946†‡	+19.44%	0.3215†‡	+12.96%	0.4719†‡	+10.51%	12554 / 41461 / 9285

Table 3: Event salience performance. (-E) and (-F) marks removing Entity information and Features from the full KCM model. The relative performance differences are computed against `Frequency`. W/T/L are the number of documents a method wins, ties, and loses compared to `Frequency`. † and ‡ mark the statistically significant improvements over `Frequency`†, `LeToR`‡ respectively.

Feature Groups	P@1	P@5	P@10	R@1	R@5	R@10	AUC
`Loc`	0.3548	0.3069	0.2497	0.0807	0.2671	0.3792	0.5226
`Frequency`	0.4536	0.4018	0.3440	0.0792	0.2846	0.4270	0.5732
+ `Loc`	0.4734	0.4097	0.3513	0.0835	0.2976	0.4436	0.6354
+ `Loc` + `Event`	0.4726	0.4101†	0.3516	0.0831	0.2969	0.4431	0.6365†
+ `Loc` + `Entity`	0.4739	0.4100	0.3518	0.0812	0.2955	0.4418	0.6374
+ `Loc` + `Entity` + `Event`	0.4739	0.4100	0.3518†	0.0832	0.2974	0.4452†	0.6374†
+ `Loc` + `Entity` + `Event` + `Local`	0.4754†	0.4100	0.3517†	0.0837	0.2981	0.4454†	0.6373†

Table 4: Feature Ablation Results. + sign indicates the additional features to `Frequency`. `Loc` is the sentence location feature. `Event` is the event voting feature. `Entity` is the entity voting feature. `Local` is the local entity voting feature. † marks the statistically significant improvements over + `Loc`.

the precision and recall metrics. *Location* performs the worst.

Feature Based: `LeToR` outperforms the baselines significantly on all metrics. Particularly, its P@1 value outperforms the *Frequency* baseline the most (4.64%), indicating a much better estimation on the most salient event. In terms of AUC, `LeToR` outperforms *Frequency* by a large margin (11.19% relative gain).

Feature Ablation: To understand the contribution of individual features, we conduct an ablation study of various feature settings in Table 4. We gradually add feature groups to the *Frequency* baseline. The combination of *Location* (sentence location) and *Frequency* almost sets the performance for the whole model. Adding each voting feature individually produces mixed results. However, adding all voting features improves all metrics. Though the margin is small, 4 of them are statistically significant over *Frequency*+*Location*.

Kernel Centrality Estimation: The `KCE` model further beats `LeToR` significantly on all metrics, by around 5% on AUC and precision values, and by around 10% on the recall values. Notably, the P@1 score is much higher, reaching 50%. The large relative gain on all the recall metrics and the high performance on precision show that `KCE` works really well on the top of the rank list.

Kernel Ablation: To understand the source of performance gain of `KCE`, we conduct an ablation study by removing its components: `-E` removes of entity kernels; `-EF` removes the entity kernels and the features. We observe a performance drop in both cases. Without entities and features, the model only using event information still performs similarly to *Frequency*. The drops are also a reflection of the small number of events (≈ 60 per document) comparing to entities (≈ 200 per document).

		Word2Vec	**KCE**
attack	kill	0.69	0.3
arrest	charge	0.53	0.3
USA (E)	war	0.46	0.3
911 attack (E)	attack	0.72	0.3
attack	trade	0.42	0.9
hotel (E)	travel	0.49	0.9
charge	murder	0.49	0.7
business (E)	increase	0.43	0.7
attack	walk	0.44	-0.3
people (E)	work	0.40	-0.3

Table 5: Similarities between event entity pairs. **Word2vec** shows the cosine similarity in pre-trained embeddings. **KCE** lists their closest kernel mean after training. (E) marks entities.

The study indicates that the relational signals and features contain different but both important information.

Discussion: The superior results of `KCE` demonstrate its effectiveness in predicting salience. So what additional information does it capture? We revisit the changes made by `KCE`: 1. it adjusts the embeddings during training. 2. it introduces weighted soft count kernels. However, the *PageRank* baseline also does embedding tuning but produces poor results, thus the second change should be crucial. We plot the learned kernel weights of `KCE` in Figure 2. Surprisingly, the salient decisions are not linearly related, nor even positively correlated to the weights. In fact, besides the "Exact Match" bin, the highest absolute weights actually appear at 0.3 and -0.3. This implies that embedding similarities do not directly imply salience, breaking some assumptions of the feature based model and *PageRank*.

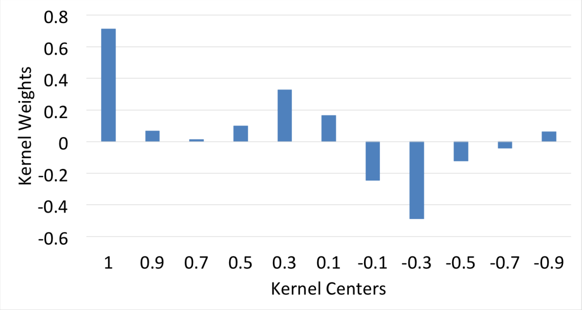

Figure 2: Learned Kernel Weights of `KCE`

Case Study: We inspect some pairs of events and entities in different kernels and list some examples in Table 5. The pre-trained embeddings are changed a lot. Pairs of units with different raw similarity values are now placed in the same bin. The pairs in Table 3 exhibit interesting types of relations: e.g.,"*arrest-charge*" and "*attack-kill*" form script-like chains; "911 attack" forms a quasi-identity relation (Recasens et al., 2010) with "attack"; "business" and "increase" are candidates as frame-argument structure. While these pairs have different raw cosine similarities, they are all useful in predicting salience. `KCE` learns to gather these relations into bins assigned with higher weights, which is not achieved by pure embedding based methods. The `KCE` has changed the embedding space and the scoring functions significantly from the original space after training. This partially explains why the raw voting features and PageRank are not as effective.

7.2 Intrusion Test Results

Figure 3 plots results of the intrusion test . The left figure shows the results of setting 1: adding non-salient intruders. The right one shows the results of setting 2: adding salient intruders. The AUC is 0.493 and the SA-AUC is 0.753 if all intruders are added.

The left figure shows that `KCE` successfully finds the non-salient intruders. The SA-AUC is higher than 0.8. Yet the AUC scores, which include the rankings of non-salience events, are rather close to random. This shows that the salient events in the origin documents form a more cohesive group, making them more robust against the intruders; the non-salient ones are not as cohesive.

In both settings, `KCE` produces higher SA-AUC than *Frequency* at the first 30%. However, in setting 2, `KCE` starts to produce lower SA-AUC than *Frequency* after 30%, then gradually drops to 0.5 (random). This phenomenon is expected since the asymmetry between origins and intruders allow `KCE` to distinguish them at the beginning. When all intruders are added, `KCE` performs worse because it relies heavily on the relations, which can be also formed by the salient intruders. This phenomenon is observed only on the salient intruders, which again confirms the cohesive relations are found among salient events.

In conclusion, we observe that the salient events form tight groups connected by discourse relations while the non-salient events are not as related. The observations imply that the main scripts in documents are mostly anchored by small groups of salient events (such as the "Trial" script in

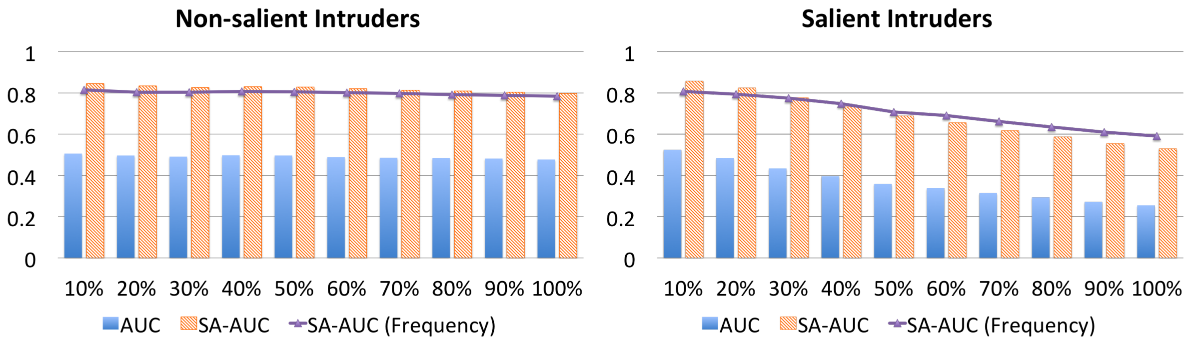

Figure 3: Intruder study results. X-axis shows the percentage of intruders inserted. Y-axis is the AUC score scale. The left and right figures are results from salient and non-salient intruders respectively. The blue bar is AUC. The orange shaded bar is SA-AUC. The line shows the SA-AUC of the frequency baseline.

Example 1). Other events may serve as "backgrounds" (Cheung et al., 2013). Similarly, Choubey et al. (2018) find that relations like event coreference and sequence are important for saliency.

8 Conclusion

We propose two salient detection models, based on lexical relatedness and semantic relations. The feature-based model with lexical similarities is effective, but cannot capture semantic relations like scripts and frames. The `KCE` model uses kernels and embeddings to capture these relations, thus outperforms the baselines and feature-based models significantly. All the results are tested on our newly created large-scale event salience dataset. While the automatic method inevitably introduces noises to the dataset, the scale enables us to study complex event interactions, which is infeasible via costly expert labeling.

Our case study shows that the salience model finds and utilize a variety of discourse relations: script chain (*attack* and *kill*), frame argument relation (*business* and *increase*), quasi-identity (*911 attack* and *attack*). Such complex relations are not as prominent in the raw word embedding space. The core message is that a salience detection module automatically discovers connections between salience and relations. This goes beyond prior centering analysis work that focuses on lexical and syntax and provide a new semantic view from the script and frame perspective.

In the intrusion test, we observe that the small number of salient events are forming tight connected groups. While `KCE` captures these relations quite effectively, it can be confused by salient intrusion events. The phenomenon indicates that the salient events are tightly connected, which form the main scripts of documents.

This paper empirically reveals many interesting connections between discourse phenomena and salience. The results also suggest that core script information may reside mostly in the salient events. Limited by the data acquisition method, this paper only models discourse salience as binary decisions. However, salience value may be continuous and may even have more than one aspects. In the future, we plan to investigate these complex settings. Another direction of study is large-scale semantic relation discovery, for example, frames and scripts, with a focus on salient discourse units.

Acknowledgement

This research was supported by DARPA grant FA8750-18-2-0018 funded under the AIDA program and National Science Foundation (NSF) grant IIS-1422676. Any opinions, findings, and conclusions in this paper are the authors and do not necessarily reflect the sponsors. We thank the anonymous reviewers whose suggestions helped clarify this paper.

References

Jun Araki, Zhengzhong Liu, Eduard Hovy, and Teruko Mitamura. 2014. Detecting Subevent Structure for Event Coreference Resolution. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4553–4558, Reykjavik, Iceland.

CF Baker, CJ Fillmore, and JB Lowe. 1998. The berkeley framenet project. *Proceeding ACL '98 Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 86–90.

Niranjan Balasubramanian, Stephen Soderland, and OE Mausam. 2013. Generating Coherent Event Schemas at Scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Breck Baldwin and Thomas S Morton. 1998. Dynamic Coreference-Based Summarization. *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing (EMNLP-3)*, pages 1–6.

Regina Barzilay and Mirella Lapata. 2008. Modeling Local Coherence: An Entity-Based Approach. In *Proceedings of the 43rd Annual Meeting of the ACL*, volume 34, pages 1–34.

Susan Windisch Brown, Claire Bonial, Leo Obrst, and Martha Palmer. 2017. The Rich Event Ontology. In *Proceedings of the Events and Stories in the News Workshop*, pages 87–97.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense Event Ordering with a Multi-Pass Architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *ACL '08 Meeting of the Association for Computational Linguistics*, pages 789–797.

Jonathan Chang, Sean Gerrish, Chong Wang, and David M Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. *Advances in Neural Information Processing Systems 22*, pages 288–296.

Pengxiang Cheng and Katrin Erk. 2018. Implicit Argument Prediction with Event Knowledge. In *NAACL 2018*, 2012.

JC Kit Cheung, H Poon, and Lucy Vanderwende. 2013. Probabilistic Frame Induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT 2013)*.

Prafulla Kumar Choubey, Kaushik Raju, and Ruihong Huang. 2018. Identifying the Most Dominant Event in a News Article by Mining Event Coreference Relations. In *NAACL 2018*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Dipanjan Das and Noah Smith. 2011. Semi-Supervised Frame-Semantic Parsing for Unknown Predicates. In *HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, volume 1, pages 1435–1444.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *EMNLP '11 Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint Inference for Event Timeline Construction. *EMNLP-CoNLL '12 Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, (July):677–687.

M. Dojchinovski, D. Reddy, T. Kliegr, T. Vitvar, and H. Sack. 2016. Crowdsourced corpus with entity salience annotations. In *Proceedings of the 10th International Conference on Language Resources and Evaluation, LREC*, pages 3307–3311.

Jesse Dunietz and Daniel Gillick. 2014. A New Entity Salience Task with Millions of Training Examples. In *Proceedings of the European Association for Computational Linguistics*, pages 205–209.

Günes Erkan and Dragomir R Radev. 2004. LexRank : Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM 2010*.

Joseph Evans Grimes. 1975. *The Thread of Discourse*. New York.

Michael Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR 2015*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL2013)*.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Zhengzhong Liu, Jun Araki, Eduard Hovy, and Teruko Mitamura. 2014. Supervised Within-Document Event Coreference using Information Propagation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4539–4544, Reykjavik, Iceland.

Jing Lu and Vincent Ng. 2017. Joint Learning for Event Coreference Resolution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 90–101.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the ACL: System Demonstrations*, pages 55–60.

Daniel Marcu. 1999. Discourse Trees are Good Indicators of Importance in Text. *Advances in Automatic Text Summarization*, pages 123–136.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 2th Advances in Neural Information Processing Systems 2013 (NIPS 2013)*, pages 3111–3119.

Teruko Mitamura, Zhengzhong Liu, and Eduard Hovy. 2015. Overview of TAC KBP 2015 Event Nugget Track. In *TAC KBP 2015*, pages 1–31.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17:21–48.

Thien Huu Nguyen and Ralph Grishman. 2015. Event Detection and Domain Adaptation with Convolutional Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371.

Haoruo Peng, Yangqi Song, and Dan Roth. 2016. Event Detection and Co-reference with Minimal Supervision. In *EMNLP 2016*.

Karl Pichotta and Raymond J. Mooney. 2016. Using Sentence-Level LSTM Language Models for Script Inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 279–289.

Marco Ponza, Paolo Ferragina, and Francesco Piccinno. 2018. SWAT: A System for Detecting Salient Wikipedia Entities in Texts.

James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Robert Gaizauskas, Andrea Setzer, Beth Sundheim, David Day, Lisa Ferro, and Dragomir. 2002. The TIMEBANK Corpus. *Natural Language Processing and Information Systems*, 4592:647–656.

Marta Recasens, Matthew Can, and Daniel Jurafsky. 2013. Same Referent, Different Words: Unsupervised Mining of Opaque Coreferent Mentions. *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (June):897–906.

Marta Recasens, Eduard Hovy, and M. Antònia Martí. 2010. A Typology of Near-Identity Relations for Coreference (NIDENT). *7th International Conference on Language Resources and Evaluation (LREC-2010)*, (i):149–156.

Rachel Rudinger, Vera Demberg, Ashutosh Modi, Benjamin Van Durme, and Manfred Pinkal. 2015. Learning to predict script events from domain-specific text. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 205–210.

Evan Sandhaus. 2008. The New York Times Annotated Corpus.

Roger C Schank and Robert P Abelson. 1977. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates.

E Skorochod'ko. 1971. Adaptive Method of Automatic Abstracting and Indexing. In *Proceedings of the IFIP Congress 71*.

Zeno Vendler. 1957. Verbs and times. *The Philosophical Review*, 66(2):143–160.

Piek Vossen and Tommaso Caselli. 2015. Storylines for structuring massive streams of news. In *Proceedings of the First Workshop on Computing News Story Lines*, pages 40–49.

Chenyan Xiong, Zhengzhong Liu, Jamie Callan, and Tie-Yan Liu. 2018. Towards Better Text Understanding and Retrieval through Kernel Entity Salience Modeling. In *SIGIR 2018*.

Chenyang Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–64.

Congle Zhang, Stephen Soderland, and Daniel S Weld. 2015. Exploiting Parallel News Streams for Unsupervised Event Extraction. volume 3, pages 117–129.

Temporal Information Extraction by Predicting Relative Time-lines

Artuur Leeuwenberg and **Marie-Francine Moens**
Department of Computer Science
KU Leuven, Belgium
`{tuur.leeuwenberg, sien.moens}@cs.kuleuven.be`

Abstract

The current leading paradigm for temporal information extraction from text consists of three phases: (1) recognition of events and temporal expressions, (2) recognition of temporal relations among them, and (3) time-line construction from the temporal relations. In contrast to the first two phases, the last phase, time-line construction, received little attention and is the focus of this work. In this paper, we propose a new method to construct a linear time-line from a set of (extracted) temporal relations. But more importantly, we propose a novel paradigm in which we directly predict start and end-points for events from the text, constituting a time-line without going through the intermediate step of prediction of temporal relations as in earlier work. Within this paradigm, we propose two models that predict in linear complexity, and a new training loss using TimeML-style annotations, yielding promising results.

1 Introduction

The current leading perspective on temporal information extraction regards three phases: (1) a *temporal entity recognition* phase, extracting events (blue boxes in Fig. 1) and their attributes, and extracting temporal expressions (green boxes), and normalizing their values to dates or durations, (2) a *relation extraction* phase, where temporal links (TLinks) among those entities, and between events and the document-creation time (DCT) are found (arrows in Fig. 1, left). And (3), construction of a time-line (Fig. 1, right) from the extracted temporal links, if they are temporally consistent. Much research concentrated on the first two steps, but very little research looks into step 3, time-line construction, which is the focus of this work.

In this paper, we propose a new time-line construction paradigm that evades phase 2, the relation extraction phase, because in the classical paradigm temporal relation extraction comes with many difficulties in training and prediction that arise from the fact that for a text with n temporal entities (events or temporal expressions) there are n^2 possible entity pairs, which makes it likely for annotators to miss relations, and makes inference slow as n^2 pairs need to be considered. Temporal relation extraction models consistently give lower performance than those in the entity recognition phase (UzZaman et al., 2013; Bethard et al., 2016, 2017), introducing errors in the time-line construction pipe-line.

The ultimate goal of our proposed paradigm is to predict from a text in which entities are already detected, for each entity: (1) a probability distribution on the entity's starting point, and (2) another distribution on the entity's duration. The probabilistic aspect is crucial for time-line based decision making. Constructed time-lines allow for further quantitative reasoning with the temporal information, if this would be needed for certain applications.

As a first approach towards this goal, in this paper, we propose several initial time-line models in this paradigm, that directly predict - in a linear fashion - start points and durations for each entity, using text with annotated temporal entities as input (shown in Fig. 1). The predicted start points and durations constitute a *relative time-line*, i.e. a total order on entity start and end points. The time-line is relative, as start and duration values cannot (yet) be mapped to absolute calender dates or durations expressed in seconds. It represents the relative temporal order and inclusions that temporal entities have with respect to each other by the quantitative start and end values of the entities. Relative time-lines are a first step toward our goal, building models that predict statistical absolute time-lines. To train our relative time-line models, we define novel loss functions that exploit TimeML-style an-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1237–1246
Brussels, Belgium, October 31 - November 4, 2018.

notations, used in most existing temporal corpora.

This work leads to the following contributions:

- A new method to construct a relative time-line from a set of temporal relations (TL2RTL).
- Two new models that, for the first time, directly predict (relative) time-lines - in linear complexity - from entity-annotated texts without doing a form of temporal relation extraction (S-TLM & C-TLM).
- Three new loss functions based on the mapping between Allen's interval algebra and the end-point algebra to train time-line models from TimeML-style annotations.

In the next sections we will further discuss the related work on temporal information extraction. We will describe the models and training losses in detail, and report on conducted experiments.

2 Related Work

2.1 Temporal Information Extraction

The way temporal information is conveyed in language has been studied for a long time. It can be conveyed directly through verb tense, explicit temporal discourse markers (e.g. *during* or *afterwards*) (Derczynski, 2017) or temporal expressions such as dates, times or duration expressions (e.g. *10-05-2010* or *yesterday*). Temporal information is also captured in text implicitly, through background knowledge about, for example, duration of events mentioned in the text (e.g. even without context, *walks* are usually shorter than *journeys*).

Most temporal corpora are annotated with TimeML-style annotations, of which an example is shown in Fig 1, indicating temporal entities, their attributes, and the TLinks among them.

The automatic extraction of TimeML-style temporal information from text using machine learning was first explored by Mani et al. (2006). They proposed a multinomial logistic regression classifier to predict the TLinks between entities. They also noted the problem of missed TLinks by annotators, and experimented with using temporal reasoning (temporal closure) to expand their training data.

Since then, much research focused on further improving the pairwise classification models, by exploring different types of classifiers and features, such as (among others) logistic regression and support vector machines (Bethard, 2013; Lin et al., 2015), and different types of neural network models, such as long short-term memory networks (LSTM) (Tourille et al., 2017; Cheng and Miyao, 2017), and convolutional neural networks (CNN) (Dligach et al., 2017). Moreover, different sieve-based approaches were proposed (Chambers et al., 2014; Mirza and Tonelli, 2016), facilitating mixing of rule-based and machine learning components.

Two major issues shared by these existing approaches are: (1) models classify TLinks in a pair-wise fashion, often resulting in an inference complexity of $O(n^2)$, and (2) the pair-wise predictions are made independently, possibly resulting in prediction of temporally inconsistent graphs. To address the second, additional temporal reasoning can be used at the cost of computation time, during inference (Chambers and Jurafsky, 2008; Denis and Muller, 2011; Do et al., 2012), or during both training and inference (Yoshikawa et al., 2009; Laokulrat et al., 2015; Ning et al., 2017; Leeuwenberg and Moens, 2017). In this work, we circumvent these issues, as we predict time-lines - in linear time complexity - that are temporally consistent by definition.

2.2 Temporal Reasoning

Temporal reasoning plays a central role in temporal information extraction, and there are roughly two approaches: (1) Reasoning directly with Allen's interval relations (shown in Table 1), by constructing rules like: If event X occurs before Y, and event Y before Z then X should happen before Z (Allen, 1990). Or (2), by first mapping the temporal interval expressions to expressions about interval end-points (start and endings of entities) (Vilain et al., 1990). An example of such mapping is that If event X occurs before Y then the end of X should be before the start of Y. Then reasoning can be done with end-points in a point algebra, which has only three point-wise relations $(=, <, >)$, making reasoning much more efficient compared to reasoning with Allen's thirteen interval relations.

Mapping interval relations to point-wise expressions has been exploited for model inference by Denis and Muller (2011), and for evaluation by UzZaman and Allen (2011). In this work, we ex-

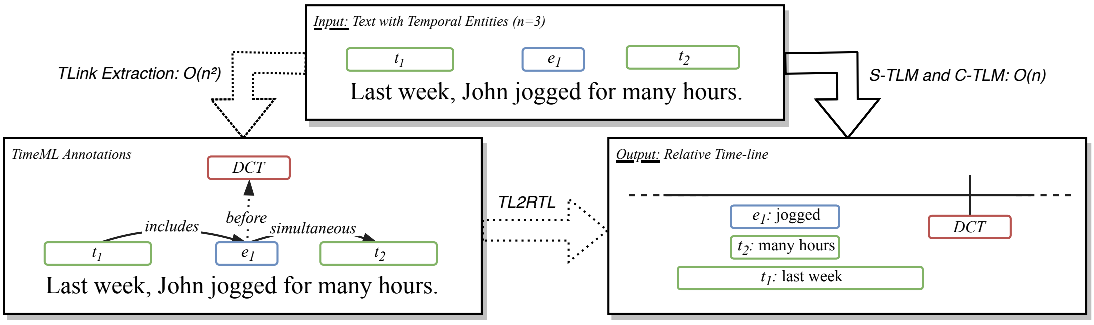

Figure 1: An overview of two paradigms: (1) The indirect approach (dashed arrows), where first TLinks are predicted from which we can build a relative time-line using TL2RTL. And (2), the direct approach (solid arrow), where a relative time-line is predicted directly from the input by S-TLM or C-TLM.

ploit it for the first time for model training, in our loss functions.

3 Models

We propose two model structures for direct time-line construction: (1) a simple context-independent model (S-TLM), and (2) a contextual model (C-TLM). Their structures are shown in Fig. 2. Additionally, we propose a method to construct relative time-lines from a set of (extracted) TLinks (TL2RTL). In this section we first explain the first two direct models S-TLM and C-TLM, and afterwards the indirect method TL2RTL.

3.1 Direct Time-line Models

Word representation

In both S-TLM and C-TLM, words are represented as a concatenation of a word embedding, a POS embedding, and a Boolean feature vector containing entity attributes such as the type, class, aspect, following (Do et al., 2012). Further details on these are given in the experiments section.

Simple Time-line Model (S-TLM)

For the simple context-independent time-line model, each entity is encoded by the word representation of the last word of the entity (generally the most important). From this representation we have a linear projection to the duration d, and the start s. S-TLM is shown by the dotted edges in Fig 2. An advantage of S-TLM is that it has very few parameters, and each entity can be placed on the time-line independently of the others, allowing parallelism during prediction. The downside is that S-TLM is limited in its use of contextual information.

Contextual Time-line Model (C-TLM)

To better exploit the entity context we also propose a contextual time-line model C-TLM (solid edges in Fig 2), that first encodes the full text using two bi-directional recurrent neural networks, one for entity starts (BiRNN$_s$), and one for entity durations (BiRNN$_d$).[1] On top of the encoded text we learn two linear mappings, one from the BiRNN$_d$ output of the last word of the entity mention to its duration d, and similarly for the start time, from the BiRNN$_s$ output to the entity's start s.

Predicting Start, Duration, and End

Both proposed models use linear mappings[2] to predict the start value s_i and duration d_i for the encoded entity i. By summing start s_i and duration d_i we can calculate the entity's end-point e_i.

$$e_i = s_i + \max(d_i, d_{min}) \tag{1}$$

Predicting durations rather than end-points makes it easy to control that the end-point lies after the start-point by constraining the duration d_i by a constant minimum duration value d_{min} above 0, as shown in Eq. 1.

Modeling Document-Creation Time

Although the DCT is often not found explicitly in the text, it is an entity in TimeML, and has TLinks to other entities. We model it by assigning it a text-independent start s_{DCT} and duration d_{DCT}.

Start s_{DCT} is set as a constant (with value 0). This way the model always has the same reference point, and can learn to position the entities w.r.t. the DCT on the time-line.

---

[1] We also experimented with sharing weights among BiRNN$_d$ and BiRNN$_s$. In our experiments, this gave worse performance, so we propose to keep them separate.

[2] Adding more layers did not improve results.

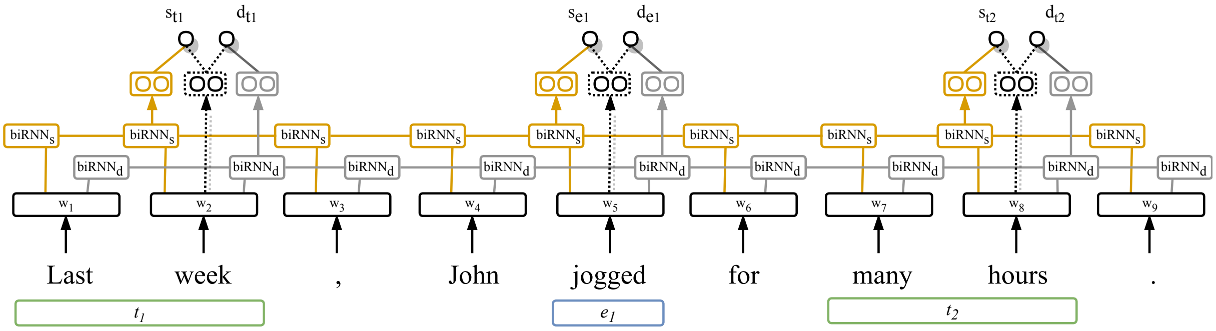

Figure 2: Schematic overview of our two time-line models: C-TLM (solid edges), exploiting entity context, and the simpler S-TLM (dotted edges), which is context independent. The models predict a starting point (s) and duration (d) for each given temporal entity (t_1, e_1, and t_2) in the input.

In contrast, DCT duration d_{DCT} is modeled as a single variable that is learned (initialized with 1). Since multiple entities may be included in the DCT, and entities have a minimum duration d_{min}, a constant d_{DCT} could possibly prevent the model from fitting all entities in the DCT. Modeling d_{DCT} as a variable allows growth of d_{DCT} and averts this issue.[3]

Training Losses

We propose three loss functions to train time-line models from TimeML-style annotations: a regular time-line loss L_τ, and two slightly expanded discriminative time-line losses, $L_{\tau ce}$ and $L_{\tau h}$.

Regular Time-line Loss (L_τ)

Ground-truth TLinks can be seen as constraints on correct positions of entities on a time-line. The regular time-line loss L_τ expresses the degree to which these constraints are met for a predicted time-line. If all TLinks are satisfied in the time-line for a certain text, L_τ will be 0 for that text.

As TLinks relate entities (intervals), we first convert the TLinks to expressions that relate the start and end points of entities. How each TLink is translated to its corresponding point-algebraic constraints is given in Table 1, following Allen (1990).

As can be seen in the last column there are only two point-wise operations in the point-algebraic constraints: an order operation ($<$), and an equality operation ($=$). To model to what degree each point-wise constraint is met, we employ hinge losses, with a margin m_τ, as shown in Eq. 2.

Table 1: Point algebraic interpretation (I_{PA}) of temporal links used to construct the loss function. The start and end points of event X are indicated by s_x and e_x respectively.

Allen Algebra	**Temporal Links**	**Point Algebra**
X precedes Y Y preceded by X	X before Y Y after X	$e_x < s_y$
X starts Y Y started by X	X begins Y Y begun by X	$s_x = s_y$ $e_x < e_y$
X finishes Y Y finished by X	X ends Y Y ended by X	$e_x = e_y$ $s_y < s_x$
X during Y Y includes X	X is included Y Y includes X	$s_y < s_x$ $e_x < e_y$
X meets Y Y met by X	X immediately before Y Y immediately after X	$e_x = s_y$
X overlaps Y Y overlapped by X	absent[4] absent[4]	$s_x < s_y$ $s_y < e_x$ $e_x < e_y$
X equals Y	X simultaneous Y X identity Y	$s_x = s_y$ $e_x = e_y$

To explain the intuition and notation: If we have a point-wise expression ξ of the form $x < y$ (first case of Eq. 2), then the predicted point $\hat{x}$ should be at least a distance m_τ smaller (or earlier on the time-line) than predicted point $\hat{y}$ in order for the loss to be 0. Otherwise, the loss represents the distance $\hat{x}$ or $\hat{y}$ still has to move to make $\hat{x}$ smaller than $\hat{y}$ (and satisfy the constraint). For the second case, if ξ is of the form $x = y$, then point $\hat{x}$ and $\hat{y}$ should lie very close to each other, i.e. at most a distance m_τ away from each other. Any distance further than the margin m_τ is counted as loss. Notice that if we set margin m_τ to 0, the second case becomes an L1 loss $|\hat{x} - \hat{y}|$. However, we use a small margin m_τ to promote some distance between ordered points and prevent con-

---

[3] Other combinations of modeling s_{DCT} and d_{DCT} as variable or constant decreased performance.

[4] No TLink for Allen's overlap relation is present in TimeML, also concluded by UzZaman and Allen (2011).

fusion with equality. Fig. 3 visualizes the loss for three TLinks.

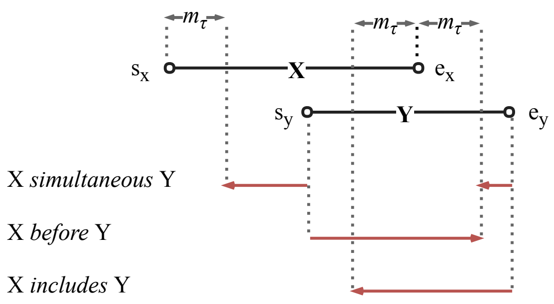

Figure 3: Visualization of the time-line loss L_τ with margin m_τ, for two events X and Y, and TLinks *simultaneous*, *before*, and *includes*. The red arrows' lengths indicate the loss per relation, i.e. how much the points should be shifted to satisfy each relation.

$$L_p(\xi|t,\theta) = \begin{cases} \max(\hat{x} + m_\tau - \hat{y}, 0) & \text{iff } x < y \\ \max(|\hat{x} - \hat{y}| - m_\tau, 0) & \text{iff } x = y \end{cases} \tag{2}$$

The total time-line loss $L_\tau(t|\theta)$ of a model with parameters θ on text t with ground-truth TLinks $R(t)$, is the sum of the TLink-level losses of all TLinks $r \in R(t)$. Each TLink-level loss $L_r(r|t,\theta)$ for TLink r is the sum of the point-wise losses $L_p(\xi|t,\theta)$ of the corresponding point-algebraic constraints $\xi \in I_{PA}(r)$ from Table 1.[5]

$$L_r(r|t,\theta) = \sum_{\xi \in I_{PA}(r)} L_p(\xi|t,\theta) \tag{3}$$

$$L_\tau(t,\theta) = \sum_{r \in R(t)} L_r(r|t,\theta) \tag{4}$$

Discriminative Time-line Losses

To promote a more explicit difference between the relations on the time-line we introduce two discriminative loss functions, $L_{\tau ce}$ and $L_{\tau h}$, which build on top of L_r. Both discriminative loss functions use an intermediate score $S(r|t,\theta)$ for each TLink r based on the predicted time-line. As scoring function, we use the negative L_r loss, as shown in Eq. 5.

$$S(r|t,\theta) = -L_r(r|t,\theta) \tag{5}$$

Then, a lower time-line loss $L_r(r|t,\theta)$ results in a higher score for relation type r. Notice that the maximum score is 0, as this is the minimum L_r.

Probabilistic Loss ($L_{\tau ce}$)

Our first discriminative loss is a cross-entropy based loss. For this the predicted scores are normalized using a soft-max over the possible relation types (TL). The resulting probabilities are used to calculate a cross-entropy loss, shown in Eq. 6. This way, the loss does not just promote the correct relation type but also distantiates from the other relation types.

$$L_{\tau ce}(t|\theta) = \sum_{r \in R(t)} r \cdot \log\left(\frac{e^{S(r|t,\theta)}}{\sum_{r' \in TL} e^{S(r'|t,\theta)}}\right) \tag{6}$$

Ranking Loss ($L_{\tau h}$)

When interested in discriminating relations on the time-line, we want the correct relation type to have the highest score from all possible relation types TL. To represent this perspective, we also define a ranking loss with a score margin m_h in Eq. 7.

$$L_{\tau h}(t|\theta) = \sum_{r \in R(t)} \sum_{r' \in TL \setminus \{r\}} \max(S(r'|t,\theta) - S(r|t,\theta) + m_h, 0) \tag{7}$$

Training Procedure

S-TLM and C-TLM are trained by by iterating through the training texts, sampling mini-batches of 32 annotated TLinks. For each batch we (1) perform a forward pass, (2) calculate the total loss (for one of the loss functions), (3) derive gradients using Adam[6] (Kingma and Ba, 2014), and (4) update the model parameters θ via back-propagation. After each epoch we shuffle the training texts. As stopping criteria we use early stopping (Morgan and Bourlard, 1990), with a patience of 100 epochs and a maximum number of 1000 epochs.

3.2 From TLinks to Time-lines (TL2RTL)

To model the indirect route, we construct a novel method, TL2RTL, that predicts relative time lines from a subset of TLinks, shown in Fig 1. One can choose any method to obtain a set of TLinks $R(t)$ from a text t, serving as input to TL2RTL.

---

[5] The TLink *during* and its inverse are mapped to *simultaneous*, following the evaluation of TempEval-3.

[6] Using the default parameters from the paper.

TL2RTL constructs a relative time-line, by assigning start and end values to each temporal entity, such that the resulting time-line satisfies the extracted TLinks $R(t)$ by minimizing a loss function that is 0 when the extracted TLinks are satisfied. TL2RTL on itself is a method and not a model. The only variables over which it optimizes the loss are the to be assigned starts and duration values.

In detail, for a text t, with annotated entities $E(t)$, we first extract a set of TLinks $R(t)$. In this work, to extract TLinks, we use the current state-of-the-art structured TLink extraction model by Ning et al. (2017). Secondly, we assign a start variable s_i, and duration variable d_i to each entity $i \in E(t)$. Similar to S-TLM and C-TLM, for each $i \in E(t)$, d_i is bounded by a minimum duration d_{min} to ensure start s_i always lies before end e_i. Also, we model the DCT start s_{DCT} as a constant, and its duration d_{DCT} as a variable. Then we minimize one of the loss functions L_τ, $L_{\tau ce}$, or $L_{\tau h}$ on the extracted TLinks $R(t)$, obtaining three TL2RTL variants, one for each loss. If the initially extracted set of TLinks $R(t)$ is consistent, and the loss is minimized sufficiently, all s_i and d_i form a relative time-line that satisfies the TLinks $R(t)$, but from which we can now also derive consistent TLinks for any entity pair, also the pairs that were not in $R(t)$. To minimize the loss we use Adam for 10k epochs until the loss is zero for each document.[7]

4 Experiments

4.1 Evaluation and Data

Because prediction of relative time-lines trained on TimeML-style annotations is new, we cannot compare our model directly to relation extraction or classification models, as the latter do not provide completely temporally consistent TLinks for all possible entity pairs, like the relative time-lines do. Neither can we compare directly to existing absolute time-line prediction models such as Reimers et al. (2018) because they are trained on different data with a very different annotation scheme.

To evaluate the quality of the relative time-line models in a fair way, we use TimeML-style test sets as follows: (1) We predict a time-line for each test-text, and (2) we check for all ground-truth annotated TLinks that are present in the data, what would be the derived relation type based on the predicted time-line, which is the relation type that gives the lowest time-line loss L_r. This results in a TLink assignment for each annotated pair in the TimeML-style reference data, and therefor we can use similar metrics. As evaluation metric we employ the temporal awareness metric, used in TempEval-3, which takes into account temporal closure (UzZaman et al., 2013). Notice that although we use the same metric, comparison against relation classification systems would be unfair, as our model assigns consistent labels to all pairs, whereas relation classification systems do not.

For training and evaluation we use two data splits, TE‡ and TD‡, exactly following Ning et al. (2017). Some statistics about the data are shown in Table 2.[8] The splits are constituted from various smaller datasets: the TimeBank (TB) (Pustejovsky et al., 2002), the AQUANT dataset (AQ), and the platinum dataset (PT) all from TempEval-3 (UzZaman et al., 2013). And, the TimeBank Dense (Cassidy et al., 2014) , and the Verb-Clause dataset (VC) (Bethard et al., 2007).

4.2 Hyper-parameters and Preprocessing

Hyper-parameters shared in all settings can be found in Table 3. The following hyper-parameters are tuned using grid search on a development set (union of TB and AQ): d_{min} is chosen from $\{1, 0.1, 0.01\}$, m_τ from $\{0, 0.025, 0.05, 0.1\}$, α_d from $\{0, 0.1, 0.2, 0.4, 0.8\}$, and α_{rnn} from $\{10, 25, 50\}$. We use LSTM (Hochreiter and Schmidhuber, 1997) as RNN units[9] and employ 50-dimensional GloVe word-embeddings pre-trained[10] on 6B words (Wikipedia and NewsCrawl) to initialize the models' word embeddings.

We use very simple tokenization and consider punctuation[11] or newline tokens as individual tokens, and split on spaces. Additionally, we lowercase the text and use the Stanford POS Tagger (Toutanova et al., 2003) to obtain POS.

---

[7] For some documents the extracted TLinks were temporally inconsistent, resulting in a non-zero loss. Nevertheless, $> 96\%$ of the extracted TLinks were satisfied.

[8] We explicitly excluded all test documents from training as some corpora annotated the same documents.

[9] We also experimented with GRU as RNN type, obtaining similar results.

[10] https://nlp.stanford.edu/projects/glove

[11] `,./\"'=+-;:()!?<>%&$*|[]`$\{\}$

Table 2: Dataset splits used for evaluation (indicated with ‡).

Split	**Training data**	**#TLinks**	**#Documents**	**Test data**	**#TLinks**	**#Documents**
TD‡	TD (train+dev)	4.4k	27	TD (test)	1.3k	9
TE3‡	TB, AQ, VC, TD (full)	17.5k	256	PT	0.9k	20

Table 3: Hyper-parameters from the experiments.

Hyper-parameter	**Value**
Document-creation starting time (s_{DCT})	0
Minimum event duration (d_{min})	0.1
Time-line margin (m_τ)	0.025
Hinge loss margin (m_h)	0.1
Dropout (α_d)	0.1
Word-level RNN units (α_{rnn})	25
Word-embedding size (α_{wemb})	50
POS-embedding size	10

Table 4: Evaluation of relative time-lines for each model and loss function, where L_* indicates the (unweighted) sum of L_τ, $L_{\tau ce}$, and $L_{\tau h}$.

	TE3‡			**TD‡**		
Model	**P**	**R**	**F**	**P**	**R**	**F**
Indirect: $O(n^2)$						
TL2RTL (L_τ)	53.5	51.1	52.3	59.1	61.2	60.1
TL2RTL ($L_{\tau ce}$)	53.9	51.7	**52.8**	61.2	60.7	60.9
TL2RTL ($L_{\tau h}$)	52.8	51.1	51.9	57.9	60.6	59.2
TL2RTL (L_*)	52.6	52.0	52.3	62.3	62.3	**62.3**
Direct: $O(n)$						
S-TLM (L_τ)	50.1	50.4	50.2	57.8	59.5	**58.6**
S-TLM ($L_{\tau ce}$)	50.1	50.0	50.1	53.4	53.5	53.5
S-TLM ($L_{\tau h}$)	51.5	51.7	**51.6**	55.1	56.4	55.7
S-TLM (L_*)	50.9	51.0	51.0	56.5	55.3	55.9
C-TLM (L_τ)	56.2	56.1	**56.1**	57.1	59.7	**58.4**
C-TLM ($L_{\tau ce}$)	54.4	55.4	54.9	52.4	57.3	54.7
C-TLM ($L_{\tau h}$)	55.7	55.5	55.6	55.3	54.9	55.1
C-TLM (L_*)	54.0	54.3	54.1	54.6	53.5	54.1

5 Results

We compared our three proposed models for the three loss functions L_τ, $L_{\tau ce}$, and $L_{\tau h}$, and their linear (unweighted) combination L_*, on TE3‡ and TD‡, for which the results are shown in Table 4.

A trend that can be observed is that overall performance on TD‡ is higher than that of TE3‡, even though less documents are used for training. We inspected why this is the case, and this is caused by a difference in class balance between both test sets. In TE3‡ there are many more TLinks of type *simultaneous* (12% versus 3%), which are very difficult to predict, resulting in lower scores for TE3‡ compared to TD‡. The difference in performance between the datasets is probably also be related to the dense annotation scheme of TD‡ compared to the sparser annotations of TE3‡, as dense annotations give a more complete temporal view of the training texts. For TL2RTL better TLink extraction[12] is also propagated into the final timeline quality.

If we compare loss functions L_τ, $L_{\tau ce}$, and $L_{\tau h}$, and combination L_*, it can be noticed that, although all loss functions seem to give fairly similar performance, L_τ gives the most robust results (never lowest), especially noticeable for the smaller dataset TD‡. This is convenient, because L_τ is fastest to compute during training, as it requires no score calculation for each TLink type. L_τ is also directly interpretable on the timeline. The combination of losses L_* shows mixed results, and has lower performance for S-TLM and C-TLM, but better performance for TL2RTL. However, it is slowest to compute, and less interpretable, as it is a combined loss.

Moreover, we can clearly see that on TE3‡, C-TLM performs better than the indirect models, across all loss functions. This is a very interesting result, as C-TLM is an order of complexity faster in prediction speed compared to the indirect models ($O(n)$ compared to $O(n^2)$ for a text with n entities).[13] We further explore why this is the case through our error analysis in the next section.

On TD‡, the indirect models seem to perform slightly better. We suspect that the reason for this is that C-TLM has more parameters (mostly the LSTM weights), and thus requires more data (TD‡ has much fewer documents than TE3‡) compared to the indirect methods. Another result supporting this hypothesis is the fact that the difference between C-TLM and S-TLM is small on the smaller

---

[12] F1 of 40.3 for TE3‡ and 48.5 for TD‡ (Ning et al., 2017)

[13] We do not directly compare prediction speed, as it would result in unfair evaluation because of implementation differences. However, currently, C-TLM predicts at ∼100 w/s incl. POS tagging, and ∼2000 w/s without. When not using POS, overall performance decreases consistently with 2-4 points.

	B	A	II	I	S
B	24.8%	4.7%	2.8%	1.6%	0.1%
A	5.0%	15.8%	3.2%	0.5%	0.0%
II	3.2%	3.2%	13.0%	0.6%	0.1%
I	4.0%	1.2%	1.0%	3.2%	0.0%
S	4.4%	3.0%	2.6%	1.3%	0.4%

	B	A	II	I	S
B	23.0%	8.2%	1.3%	0.9%	0.8%
A	4.7%	17.1%	1.8%	0.3%	0.5%
II	4.3%	4.4%	11.1%	0.4%	0.0%
I	1.6%	5.4%	0.5%	1.3%	0.5%
S	4.3%	4.1%	1.8%	0.6%	0.9%

Figure 4: On the **left**, the confusion matrix of C-TLM (L_τ), and on the **right** of TL2RTL ($L_{\tau ce}$), on TE3‡ for the top-5 most-frequent TLinks (together 95% of data): BEFORE (B), AFTER (A), IS INCLUDED (II), INCLUDES (I), and SIMULTANEOUS (S). Predictions are shown on the x-axis and ground-truth on the y-axis.

TD‡, indicating that C-TLM does not yet utilize contextual information from this dataset, whereas, in contrast, on TE3‡, the larger dataset, C-TLM clearly outperforms S-TLM across all loss functions, showing that when enough data is available C-TLM learns good LSTM weights that exploit context substantially.

6 Error Analysis

We compared predictions of TL2RTL(L_τ) with those of C-TLM (L_τ), the best models of each paradigm. In Table 4, we show the confusion matrices of both systems on TE3‡.

When looking at the overall pattern in errors, both models seem to make similar confusions on both datasets (TD‡ was excluded for space constraints). Overall, we find that *simultaneous* is the most violated TLink for both models. This can be explained by two reasons: (1) It is the least frequent TLink in both datasets. And (2), simultaneous entities are often co-referring events. Event co-reference resolution is a very difficult task on its own.

We also looked at the average token-distance between arguments of correctly satisfied TLinks by the time-lines of each model. For TL2RTL (L_τ) this is 13 tokens, and for C-TLM (L_τ) 15. When looking only at the TLinks that C-TLM (L_τ) satisfied and TL2RTL (L_τ) did not, the average distance is 21. These two observations suggest that the direct C-TLM (L_τ) model is better at positioning entities on the time-line that lie further away from each other in the text. An explanation for this can be error propagation of TLink extraction to the time-line construction, as the pairwise TLink extraction of the indirect paradigm extracts TLinks in a contextual window, to prune the $O(n^2)$ number of possible TLink candidates. This consequently prevents TL2RTL to properly position distant events with respect to each other.

Table 5: Example events from the top-shortest/longest durations and top-earliest/latest start values assigned by the model.

Short d	**Long** d	**Early** s	**Late** s
started	going	destroyed	realize
meet	expects	finished	bring
entered	recession	invaded	able
told	war	pronounced	got
arrived	support	created	work
allow	make	took	change
send	think	appeared	start
asked	created	leaving	reenergize

To get more insight in what the model learns we calculated mean durations and mean starts of C-TLM (L_τ) predictions. Table 5 contains examples from the top-shortest, and top-longest duration assignments and earliest and latest starting points. We observe that events that generally have more events included are assigned longer duration and vice versa. And, events with low start values are in the past tense and events with high start values are generally in the present (or future) tense.

7 Discussion

A characteristic of our model is that it assumes that all events can be placed on a single time-line, and that it does not assume that unlabeled pairs are temporally unrelated. This has big advantages: it results in fast prediction, and missed annotation do not act as noise to the training, as they do for pairwise models. Ning et al. (2018) argue that actual, negated, hypothesized, expected or opinionated events should possibly be annotated on separate time-axis. We believe such multi-axis representations can be inferred from the generated single time-lines if hedging information is recognized.

8 Conclusions

This work leads to the following three main contributions[14]: (1) Three new loss functions that connect the interval-based TimeML-annotations to points on a time-line, (2) A new method, TL2RTL, to predict relative time-lines from a set of predicted temporal relations. And (3), most importantly, two new models, S-TLM and C-TLM, that – to our knowledge for the first time – predict (relative) time-lines in linear complexity from text, by evading the computationally expensive (often $O(n^2)$) intermediate relation extraction phase in earlier work. From our experiments, we conclude that the proposed loss functions can be used effectively to train direct and indirect relative time-line models, and that, when provided enough data, the – much faster – direct model C-TLM outperforms the indirect method TL2RTL.

As a direction for future work, it would be very interesting to extend the current models, diving further into direct time-line models, and learn to predict absolute time-lines, i.e. making the time-lines directly mappable to calender dates and times, e.g. by exploiting complementary data sources such as the EventTimes Corpus (Reimers et al., 2016) and extending the current loss functions accordingly. The proposed models also provide a good starting point for research into probabilistic time-line models, that additionally model the (un)certainty of the predicted positions and durations of the entities.

Acknowledgments

The authors thank Geert Heyman and the reviewers for their constructive comments which helped us to improve the paper. This work was funded by the KU Leuven C22/15/16 project "MAchine Reading of patient recordS (MARS)", and by the IWT-SBO 150056 project "ACquiring CrUcial Medical information Using LAnguage TEchnology" (ACCUMULATE).

References

James F Allen. 1990. Maintaining knowledge about temporal intervals. *Readings in Qualitative Reasoning about Physical Systems*, pages 361–372.

Steven Bethard. 2013. ClearTK-TimeML: A minimalist approach to tempeval 2013. In *Proceedings of SemEval*, volume 2, pages 10–14. ACL.

Steven Bethard, James H. Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *Proceedings of ICSC*, pages 11–18.

Steven Bethard, Guergana Savova, Wei-Te Chen, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2016. Semeval-2016 task 12: Clinical tempeval. In *Proceedings of SemEval*, pages 1052–1062. ACL.

Steven Bethard, Guergana Savova, Martha Palmer, and James Pustejovsky. 2017. SemEval-2017 Task 12: Clinical TempEval. In *Proceedings of SemEval*, pages 565–572. ACL.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of ACL*, pages 501–506. ACL.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of EMNLP*, pages 698–706. ACL.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional LSTM over dependency paths. In *Proceedings of ACL*, volume 2, pages 1–6. ACL.

Pascal Denis and Philippe Muller. 2011. Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. In *Proceedings of IJCAI*, pages 1788–1793.

Leon RA Derczynski. 2017. *Automatically Ordering Events and Times in Text*, volume 677. Springer.

Dmitriy Dligach, Timothy Miller, Chen Lin, Steven Bethard, and Guergana Savova. 2017. Neural temporal relation extraction. In *Proceedings of EACL*, volume 2, pages 746–751.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of EMNLP-CoNLL*, pages 677–687. ACL.

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

---

[14] Code is available at: liir.cs.kuleuven.be/software.php

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Natsuda Laokulrat, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Stacking approach to temporal relation classification. *Journal of Natural Language Processing*, 22(3):171–196.

Artuur Leeuwenberg and Marie-Francine Moens. 2017. Structured learning for temporal relation extraction from clinical records. In *Proceedings of EACL*, volume 1, pages 1150–1158. ACL.

Chen Lin, Dmitriy Dligach, Timothy A Miller, Steven Bethard, and Guergana K Savova. 2015. Multilayered temporal modeling for the clinical domain. *Journal of the American Medical Informatics Association*, 23(2):387–395.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of COLING-ACL*, pages 753–760. ACL.

Paramita Mirza and Sara Tonelli. 2016. CATENA : Causal and temporal relation extraction from natural language texts. *Proceedings of COLING*, pages 64–75.

Nelson Morgan and Hervé Bourlard. 1990. Generalization and parameter estimation in feedforward nets: Some experiments. In *Advances in Neural Information Processing Systems*.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. *Proceedings of EMNLP*, pages 1038–1048.

Qiang Ning, Hao Wu, and Dan Roth. 2018. A multi-axis annotation scheme for event temporal relations. In *Proceedings of ACL*, pages 1318–1328. ACL.

James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Robert Gaizauskas, Andrea Setzer, Beth Sundheim, David Day, Lisa Ferro, and Dragomir. 2002. The TIMEBANK Corpus. *Natural Language Processing and Information Systems*, 4592:647–656.

Nils Reimers, Nazanin Dehghani, and Iryna Gurevych. 2016. Temporal anchoring of events for the timebank corpus. *Proceedings of ACL*, pages 2195–2204.

Nils Reimers, Nazanin Dehghani, and Iryna Gurevych. 2018. Event time extraction with a decision tree of neural classifiers. *Transactions of the Association for Computational Linguistics*, 6:77–89.

Julien Tourille, Olivier Ferret, Aurelie Neveol, and Xavier Tannier. 2017. Neural architecture for temporal relation extraction: A Bi-LSTM approach for detecting narrative containers. In *Proceedings of ACL*, pages 224–230.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL-HLT*, pages 173–180. ACL.

Naushad UzZaman and James F. Allen. 2011. Temporal evaluation. In *Proceedings of ACL*, HLT '11, pages 351–356, Stroudsburg, PA, USA. ACL.

Naushad UzZaman, Hector Llorens, Leon Derczynski, Marc Verhagen, James Allen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. *Second joint conference on lexical and computational semantics (* SEM)*, 2:1–9.

Marc Vilain, Henry Kautz, and Peter Van Beek. 1990. Constraint propagation algorithms for temporal reasoning: A revised report. In *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Elsevier.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of ACL-IJCNLP*, pages 405–413. ACL.

Jointly Multiple Events Extraction via Attention-based Graph Information Aggregation

Xiao Liu[†] and **Zhunchen Luo**[‡] and **Heyan Huang**[†*]
[†]School of Computer Science and Technology, Beijing Institute of Technology
100081 Beijing, China
`{xiaoliu,hhy63}@bit.edu.cn`
[‡]Information Research Center of Military Science, PLA Academy of Military Science
100142 Beijing, China
`zhunchenluo@gmail.com`

Abstract

Event extraction is of practical utility in natural language processing. In the real world, it is a common phenomenon that multiple events existing in the same sentence, where extracting them are more difficult than extracting a single event. Previous works on modeling the associations between events by sequential modeling methods suffer a lot from the low efficiency in capturing very long-range dependencies. In this paper, we propose a novel **J**ointly **M**ultiple **E**vents **E**xtraction (JMEE) framework to jointly extract multiple event triggers and arguments by introducing syntactic shortcut arcs to enhance information flow and attention-based graph convolution networks to model graph information. The experiment results demonstrate that our proposed framework achieves competitive results compared with state-of-the-art methods.

1 Introduction

Extracting events from natural language text is an essential yet challenging task for natural language understanding. When given a document, event extraction systems need to recognize event triggers with their specific types and their corresponding arguments with the roles. Technically speaking, as defined by the ACE 2005 dataset[1], a benchmark for event extraction (Grishman et al., 2005), the event extraction task can be divided into two subtasks, i.e., event detection (identifying and classifying event triggers) and argument extraction (identifying arguments of event triggers and labeling their roles).

In event extraction, it is a common phenomenon that multiple events exist in the same sentence. Extracting the correct multiple events from those sentences is much more difficult than in the one-event-one-sentence cases because those various types of events are often associated with each other. For example, in the sentence "He ***left*** the company, and planned to ***go*** home directly.", the trigger word ***left*** may trigger a *Transport* (a person left a place) event or an *End-Position* (a person retired from a company) event. However, if we take the following event triggered by ***go*** into consideration, we are more confident to judge it as a *Transport* event rather than an *End-Position* event. This phenomenon is quite common in our real world, as *Injure* and *Die* events are more likely to co-occur with *Attack* events than others, whereas *Marry* and *Born* events are less likely to co-occur with *Attack* events. As we investigated in ACE 2005 dataset, there are around 26.2% (1042/3978) sentences belong to this category.

Significant efforts have been dedicated to solving this problem. Most of them exploiting various features (Liu et al., 2016b; Yang and Mitchell, 2016; Li et al., 2013; Keith et al., 2017; Liu et al., 2016a; Li et al., 2015), introducing memory vectors and matrices (Nguyen et al., 2016), introducing more transition arcs (Sha et al., 2018), keeping more contextual information (Chen et al., 2015) into sentence-level sequential modeling methods like RNNs and CRFs. Some also seek features in document-level methods (Liao and Grishman, 2010; Ji and Grishman, 2008). However, sentence-level sequential modeling methods suffer a lot from the low efficiency in capturing very long-range dependencies while the feature-based methods require extensive human engineering, which also largely affects model performance. Besides, these methods do not adequately model the associations between events.

An intuitive way to alleviate this phenomenon is to introduce shortcut arcs represented by linguistic resources like dependency parsing trees to

---

[*] Corresponding author.

[1] `https://catalog.ldc.upenn.edu/ldc2006t06`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1247–1256
Brussels, Belgium, October 31 - November 4, 2018.

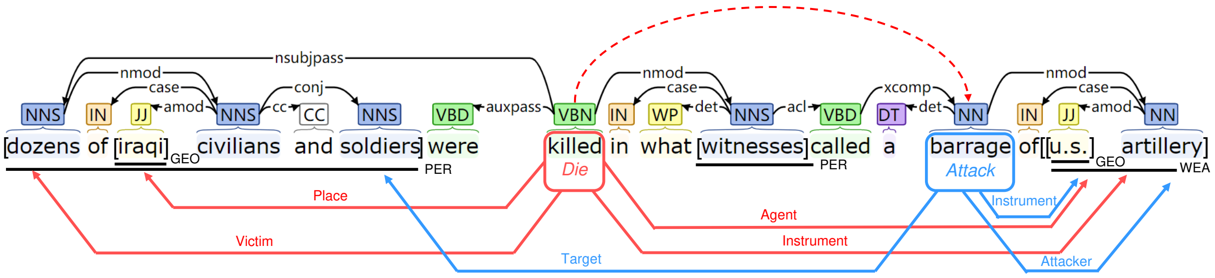

Figure 1: An example of dependency parsing result produced by Stanford CoreNLP. There are two events in the sentence: a *Die* event triggered by the word *killed* with four arguments in red and an *Attack* event triggered by the word *barrage* with three arguments in blue. The red dotted arc is the shortcut path consisting of three directed arcs from trigger *killed* to another trigger *barrage*.

drain the information flow from a point to its target through fewer transitions. Comparing to sequential order, modeling with these arcs often successfully reduce the needed hops from one event trigger to another in the same sentences. In Figure 1, for example, there are two events: a *Die* event triggered by the word *killed* with four arguments in red and an *Attack* event triggered by the word *barrage* with three arguments in blue. We need six hops from *killed* to *barrage* according to sequential order, but only three hops according to the arcs in dependency parsing tree (along the *nmod*-arc from *killed* to *witnesses*, along the *acl*-arc from *witnesses* to *called*, and along the *xcomp*-arc from *called* to *barrage*). These three arcs consist of a shortcut path[2], draining the dependency syntactic information flow from *killed* to *barrage* with fewer hops[3].

In this paper, we propose a novel **J**ointly **M**ultiple **E**vents **E**xtraction (JMEE) framework by introducing syntactic shortcut arcs to enhance information flow and attention-based graphic convolution networks to model the graph information. To implement modeling with the shortcut arcs, we adopt the graph convolutional networks (GCNs) (Kipf and Welling, 2016; Marcheggiani and Titov, 2017; Nguyen and Grishman, 2018) to learn syntactic contextual representations of each node by the representative vectors of its immediate neighbors in the graph. And then we utilize the syntactic contextual representations to extract triggers and arguments jointly by a self-attention mechanism to aggregate information especially keeping the associations between multiple events.

We extensively evaluate the proposed JMEE framework with the widely-used ACE 2005 dataset to demonstrate its benefits in the experiments especially in capturing the associations between events. To summary, our contribution in this work is as follows:

- We propose a novel joint event extraction framework JMEE based on syntactic structures which enhance information flow and alleviate the phenomenon where multiple events are in the same sentence.
- We propose a self-attention mechanism to aggregate information especially keeping the associations between multiple events and prove it is useful in event extraction.
- We achieve the state-of-the-art performance on the widely used datasets for event extraction using the proposed model with GCNs and self-attention mechanism.

2 Approach

Generally, event extraction can be cast as a multi-class classification problem deciding whether each word in the sentence forms a part of event trigger candidate and whether each entity in the sentence plays a particular role in the event triggered by the candidate triggers. There are two main approaches to event extraction: (i) the joint approach that extracts event triggers and arguments simultaneously as a structured prediction problem, and (ii) the pipelined approach that first performs trigger prediction and then identifies arguments in separate stages. We follow the joint approach that can effectively avoid the propagated errors in the pipeline.

Additionally, we extract events in sentence-level mainly for three reasons. Firstly, in our in-

---

[2] In a shortcut path which consists of existing arcs, some arcs may reverse their directions.

[3] The length of the longest path in a tree is always no more than the sequential length consisting of the same number of nodes, which means even in the worst cases, the shortcut path will not perform worse than sequential modeling.

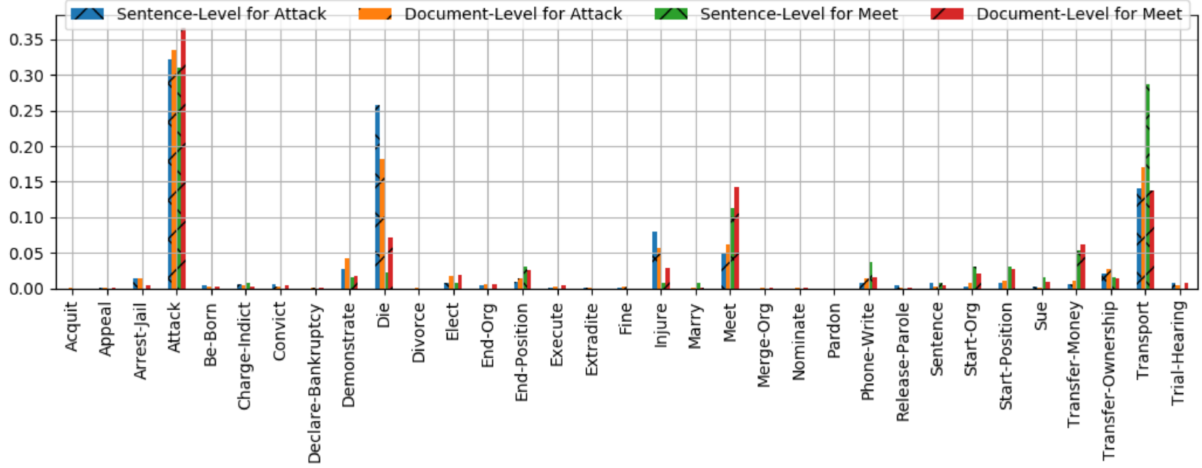

Figure 2: The conditional probability distributions of all the 33 event types in sentences and documents where an *Attack* event or a *Meet* event appears. The label on the top shows which histogram the color stands.

vestigation, we find that the document-level co-occurrence distributions of 33 types of events in the ACE 2005 dataset are relatively similar to the sentence-level co-occurrence distributions. In Figure 2, for example, the blue bars and the orange bars indicate the conditional probability distribution of all the 33 event types in sentences and documents, respectively, where an *Attack* event appears. While the green bars and the red bars indicate the sentence-level and document-level conditional probability distributions respectively. As we can see from this figure, the top three types of *Attack* event in co-occurrence relationships are *Die*, *Transport* and *Injure*, while those of *Meet* event are *Attack*, *Transport*, and *Transfer-Money*. Although different types of events have different co-occurrence relationships, the conditional probability distributions in two levels of the same event type are relatively similar[4]. Secondly, there are many off-the-shelf sentence-level linguistic resources in the NLP community which can offer analytical information about the shortcut paths of some structures, like dependency parsing trees, AMR parsing graphs, and semantic role labeling structures. Last but not least, we also find that events within the same sentences have more explicit relationships with each other than events in different sentences of a document, which means the associations between two events is more accessible to capture.

Let $W = w_1, w_2, ..., w_n$ be a sentence of length n where w_i is the i-th token. Similarly, let $E = e_1, e_2, ..., e_k$ be the entity mentions in the sentence where k is the number of the entity mentions. We apply the BIO annotation schema to assign trigger label t_i to each token w_i, as there are triggers that consist of multiple tokens. If we can get trigger candidates of certain types from trigger labels in BIO annotation schema, we then need to predict the roles (if any) that each entity mention e_j plays in such events.

Our JMEE framework consists of the following four modules: (i) word representation module that represents the sentence with vectors, (ii) syntactic graph convolution network module that performs convolution operations by introducing shortcut arcs from syntactic structures, (iii) self-attention trigger classification module that captures the associations between multiple events in a sentence, and (iv) argument classification that predicts the roles each entity mention e_j plays in the event candidates of specific types, as shown in Figure 3.

2.1 Word Representation

In the word representation module, each token w_i in the sentence is transformed to a real-valued vector x_i by looking up in embedding matrices and concatenating the following vectors:

- The word embedding vector of w_i: This is obtained by looking up a pre-trained word embedding matrix Glove (Pennington et al., 2014).
- The POS-tagging label embedding vector of w_i: This is generated by looking up the randomly initialized POS-tagging label embedding table.

---

[4] We only focus on the top-K co-occurrence relationships because the rest are too sparse for statistic analysis.

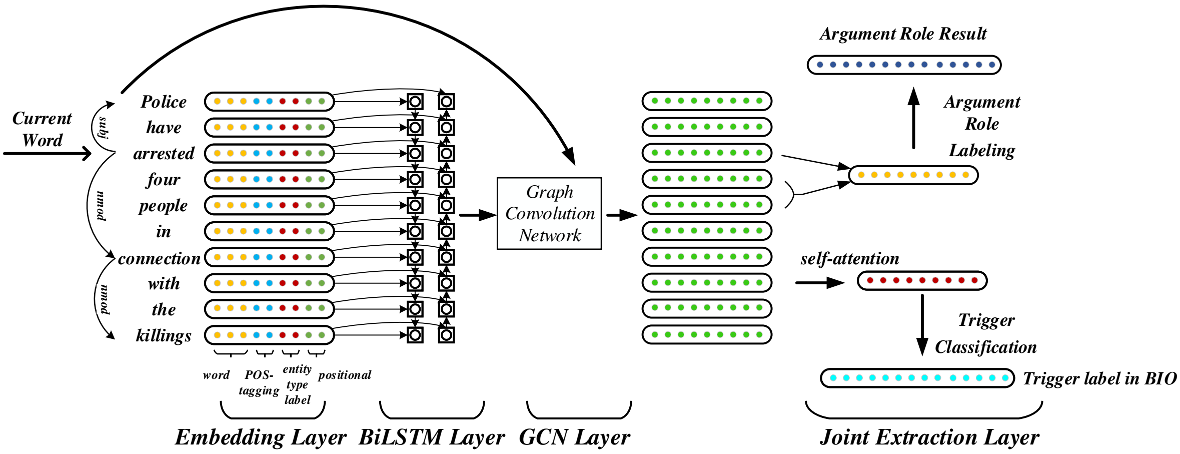

Figure 3: The architecture of our jointly multiple events extraction framework.

- The positional embedding vector of w_i: If w_c is the current word, we encode the relative distance $i - c$ from w_i to w_c as a real-valued vector by looking up the randomly initialized position embedding table (Nguyen et al., 2016; Liu et al., 2017; Nguyen and Grishman, 2018).

- The entity type label embedding vector of w_i: Similarly to the POS-tagging label embedding vector of w_i, we annotate the entity mentions in a sentence using BIO annotation schema and transform the entity type labels to real-valued vectors by looking up the embedding table. It should be noticed that we use the whole entity extent in ACE 2005 dataset which contains overlapping entity mentions and we sum all the possible entity type label embedding vectors for each token.

The transformation from the token w_i to the vector x_i essentially converts the input sentence W into a sequence of real-valued vectors $X = (x_1, x_2, ..., x_n)$, which will be feed into later modules to learn more effective representations for event extraction.

2.2 Syntactic Graph Convolution Network

Considering an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as the syntactic parsing tree for sentence W, where $\mathcal{V} = v_1, v_2, ..., v_n (|\mathcal{V}| = n)$ and $\mathcal{E}$ are sets of nodes and edges, respectively. In $\mathcal{V}$, each v_i is the node representing token w_i in W. Each edge $(v_i, v_j) \in \mathcal{E}$ is a directed syntactic arc from token w_i to token w_j, with the type label $K(w_i, w_j)$. Additionally, to allow information to flow against the direction, we also add reversed edge (v_j, v_i) with the type label $K'(w_i, w_j)$. Following Kipf and Welling (2016), we also add all the self-loops, i.e., (v_i, v_i) for any $v_i \in \mathcal{V}$. For example, in the dependency parsing tree shown in Figure 1, there are four arcs in the subgraph with only two nodes *"killed"* and *"witnesses"*: the dependency arc with the type label $K($*"killed"*, *"witnesses"*$) = nmod$, the revresed dependency arc with the additional type label $K($*"witnesses"*, *"killed"*$) = nmod'$, and the two self-loops of *"killed"* and *"witnesses"* with type label $K($*"killed"*, *"killed"*$) = K($*"witnesses"*, *"witnesses"*$) = loop$.

Therefore, in the k-th layer of syntactic graph convolution network module, we can calculate the graph convolution vector $h_v^{(k+1)}$ for node $v \in \mathcal{V}$ by:

$$h_v^{(k+1)} = f\Big(\sum_{u \in \mathcal{N}(v)} (W^{(k)}_{K(u,v)} h_u^{(k)} + b^{(k)}_{K(u,v)})\Big) \quad (1)$$

where $K(u, v)$ indicates the type label of the edge (u, v); $W^{(k)}_{K(u,v)}$ and $b^{(k)}_{K(u,v)}$ are the weight matrix and the bias for the certain type label $K(u, v)$, respectively; $\mathcal{N}(v)$ is the set of neighbors of v including v (because of the self-loop); f is the activation function. Moreover, we use the output of the word representation module x_i to initialize the node representation $h^0_{v_i}$ of the first layer of GCNs.

After applying the above two changes, the number of predefined directed arc type label (let us say, N) will be doubled (to $2N + 1$). It means we will have $2N + 1$ sets of parameter pairs $W_k^{(k)}$ and $b_k^{(k)}$ for a single layer of GCN. In this work, we use Stanford Parser (Klein and Manning, 2003) to generate the arcs in dependency parsing trees for sentences as the shortcut arcs. The current representation contains approximately 50 different grammatical relations, which is too high for the parameter number of a single layer of GCN and not compatible with the existing training data scale. To reduce the parameter numbers, following Marcheggiani and Titov (2017), we modify the definition of type label $K(w_i, w_j)$ to:

$$K(w_i, w_j) = \begin{cases} along, & (v_i, v_j) \in \mathcal{E} \\ rev, & i! = j \& (v_j, v_i) \in \mathcal{E} \\ loop, & i == j \end{cases} \tag{2}$$

where the new $K(w_i, w_j)$ only have three type labels.

As not all types of edges are equally informative for the downstream task, moreover, there are also noises in the generated syntactic parsing structures; we apply gates on the edges to weight their individual importances. Inspired by Dauphin et al. (2017); Marcheggiani and Titov (2017), we calculate a weight $g_{u,v}^{(k)}$ for each edge (u, v) indicating the importance for event extraction by:

$$g_{u,v}^{(k)} = \sigma(h_u^{(k)} V_{K(u,v)}^{(k)} + d_{K(u,v)}^{(k)}) \tag{3}$$

where σ is the logistic sigmoid function, $V_{K(u,v)}^{(k)}$ and $d_{K(u,v)}^{(k)}$ are the weight matrix and the bias of the gate. With this additional gating mechanism, the final syntactic GCN computation is formulated as

$$h_v^{(k+1)} = f\Big(\sum_{u \in \mathcal{N}(v)} g_{u,v}^{(k)} (W_{K(u,v)}^{(k)} h_u^{(k)} + b_{K(u,v)}^{(k)})\Big) \tag{4}$$

As stacking k layers of GCNs can model information in k hops, and sometimes the length of shortcut path between two triggers is less than k, to avoid information over-propagating, we adapt highway units (Srivastava et al., 2015), which allow unimpeded information flowing across stacking GCN layers. Typically, highway layers conduct nonlinear transformation as:

$$t = \sigma(W_T h_v^k + b_T) \tag{5}$$

$$\overline{h}_v^{(k+1)} = h_v^{(k+1)} + t \odot g(W_H h_v^k + b_H) + (1-t) \odot h_v^k \tag{6}$$

where σ is the sigmoid function; $\odot$ is the element-wise product operation; g is a nonlinear activation function; t is called transform gate and $(1-t)$ is called carry gate. Therefore, the input of the k-th GCN layers should be $\overline{h}^{(k)}$ instead of $h^{(k)}$.

The GCNs are designed to capture the dependencies between shortcut arcs, while the layer number of GCNs limits the ability to capture local graph information. However, in this cases, we find that leveraging local sequential context will help to expand the information flow without increasing the layer number of GCNs, which means LSTMs and GCNs maybe complementary. Therefore, instead of feeding the word representation $X = (x_1, x_2, ..., x_n)$ into the first GCN layer, we follow Marcheggiani and Titov (2017), apply Bidirectional LSTM (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) to encode the the word representation X as:

$$\overrightarrow{p}_t = \overrightarrow{LSTM}(\overrightarrow{p}_{t-1}, x_t) \tag{7}$$

$$\overleftarrow{p}_t = \overleftarrow{LSTM}(\overleftarrow{p}_{t-1}, x_t) \tag{8}$$

and the input of t-th token to GCNs is $\overline{x}_t = [\overrightarrow{p}_t, \overleftarrow{p}_t]$, where $[,]$ is the concatenation operation. The Bi-LSTM adaptively accumulates and abstracts the context for each token in the sentence.

2.3 Self-Attention Trigger Classification

When taking each token as the current word, we get the representation D from all tokens calculated by GCNs. Traditional event extraction systems often use max-pooling or its amelioration to aggregate information to each position. However, the max-pooling aggregation mechanisms tend to produce similar results after GCN modules in our framework. For example, if we get the aggregated vector Ag_i at each position i by this max-pooling mechanism $Ag_i = max\_pooling_{j=1}^{n}(H_j)$ with the GCNs output $\{H_j | j = 1, ..., n\}$ in which n is the sentence length, and the vector Ag_i is all the same at each position. Besides, predicting a trigger label for a token should take other possible trigger candidates into consideration. To capture the associations between triggers in a sentence, we design a self-attention mechanism to aggregate information especially keeping the associations between multiple events.

Given the current token w_i, the self-attention score vector and the context vector at position i are calculated as:

$$score = norm(exp(W_2 f(W_1 D + b_1) + b_2)) \tag{9}$$

$$C_i = [\sum_{j=1,j!=i}^{n} score_j * D_j, D_i] \tag{10}$$

where $norm$ means the normalization operation. Then we feed the context vector C_i into a fully-connected network to predict the trigger label in BIO annotation schema as:

$$\overline{C}_i = f(W_c C_i + b_c) \tag{11}$$

$$y_{t_i} = softmax(W_t \overline{C}_i + b_t) \tag{12}$$

where f is a non-linear activation and y_{t_i} is the final output of the i-th trigger label.

2.4 Argument Classification

When we have extracted an entire trigger candidate, which is meeting an *O* label after an *I-Type* label or a *B-Type* label, we use the aggregated context vector $\overline{C}$ to perform argument classification on the entity list in the sentence.

For each entity-trigger pair, as both the entity and the trigger candidate are likely to be a subsequence of tokens, we aggregate the context vectors of subsequences to trigger candidate vector T_i and entity vector E_j by average pooling along the sequence length dimension. Then we concatenate them together and feed into a fully-connected network to predict the argument role as:

$$y_{a_{ij}} = softmax(W_a[T_i, E_j] + b_a) \tag{13}$$

where $y_{a_{ij}}$ is the final output of which role the j-th entity plays in the event triggered by the i-th trigger candidate.

When training our framework, if the trigger candidate that we focus on is not a correct trigger, we set all the golden argument labels concerning the trigger candidate to *OTHER* (not any roles). With this setting, the labels of the trigger candidate will be further adjusted to reach a reasonable probability distribution.

2.5 Biased Loss Function

In order to train the networks, we minimize the joint negative log-likelihood loss function. Due to the data sparsity in the ACE 2005 dataset, we adapt our joint negative log-likelihood loss function by adding a bias item as:

$$\begin{aligned} J(\theta) = -\sum_{p=1}^{N}(\sum_{i=1}^{n_p} I(y_{t_i}) log(p(y_{t_i}|\theta)) \\ +\beta \sum_{i=1}^{t_p}\sum_{j=1}^{e_p} log(p(y_{a_{ij}}|\theta))) \end{aligned} \tag{14}$$

where N is the number of sentences in training corpus; n_p, t_p and e_p are the number of tokens, extracted trigger candidates and entities of the p-th sentence; $I(y_{t_i})$ is an indicating function, if y_{t_i} is not *O*, it outputs a fixed positive floating number α bigger than one, otherwise one; β is also a floating number as a hyper-parameter like α.

3 Experiments

3.1 Experiment Settings

Dataset, Resources and Evaluation Metric

We evaluate our JMEE framework on the ACE 2005 dataset. The ACE 2005 dataset annotate 33 event subtypes and 36 role classes, along with the NONE class and BIO annotation schema, we will classify each token into 67 categories in event detection and 37 categories in argument extraction. To comply with previous work, we use the same data split as the previous work (Ji and Grishman, 2008; Liao and Grishman, 2010; Li et al., 2013; Chen et al., 2015; Liu et al., 2016b; Yang and Mitchell, 2016; Nguyen et al., 2016; Sha et al., 2018). This data split includes 40 newswire articles (881 sentences) for the test set, 30 other documents (1087 sentences) for the development set and 529 remaining documents (21,090 sentences) for the training set.

We deploy the Stanford CoreNLP toolkit[5] to preprocess the data, including tokenizing, sentence splitting, pos-tagging and generating dependency parsing trees.

Also, we follow the criteria of the previous work (Ji and Grishman, 2008; Liao and Grishman, 2010; Li et al., 2013; Chen et al., 2015; Liu et al., 2016b; Yang and Mitchell, 2016; Nguyen et al., 2016; Sha et al., 2018) to judge the correctness of the predicted event mentions.

Hyperparameter Setting

For all the experiments below, in the word representation module, we use 300 dimensions for the embeddings and 50 dimensions for the rest

---

[5] `http://stanfordnlp.github.io/CoreNLP/`

Method	**Trigger Identification (%)**			**Trigger Classification (%)**			**Argument Identification (%)**			**Argument Role (%)**		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Cross-Event	N/A			68.7	68.9	68.8	50.9	49.7	50.3	45.1	44.1	44.6
JointBeam	76.9	65.0	70.4	73.7	62.3	67.5	69.8	47.9	56.8	64.7	44.4	52.7
DMCNN	**80.4**	67.7	73.5	75.6	63.6	69.1	68.8	51.9	59.1	62.2	46.9	53.5
PSL	N/A			75.3	64.4	69.4	N/A			N/A		
JRNN	68.5	**75.7**	71.9	66.0	**73.0**	69.3	61.4	64.2	62.8	54.2	56.7	55.4
dbRNN	N/A			74.1	69.8	71.9	71.3	64.5	67.7	66.2	52.8	58.7
JMEE	80.2	72.1	**75.9**	**76.3**	71.3	**73.7**	**71.4**	**65.6**	**68.4**	**66.8**	**54.9**	**60.3**

Table 1: Overall performance comparing to the state-of-the-art methods with golden-standard entities.

three embeddings including pos-tagging embedding, positional embedding and entity type embedding. In the syntactic GCN module, we use a three-layer GCN, a one-layer Bi-LSTM with 220 hidden units, self-attention with 300 hidden units and 200 hidden units for the rest transformation. We also set dropout rate to 0.5 and L2-norm to 1e-8. The batch size in our experiments is 32, and we utilize a maximum length $n = 50$ of sentences in the experiments by padding shorter sentences and cutting off longer ones. These hyperparameters are either randomly searched or chosen by experiences when tuning in the development set.

We use ReLU (Glorot et al., 2011) as our nonlinear activate function. We apply the stochastic gradient descent algorithm with mini-batches and the AdaDelta update rule (Zeiler, 2012). The gradients are computed using back-propagation. During training, besides the weight matrices, we also fine-tune all the embedding tables.

3.2 Overall Performance

We compare our performance with the following state-of-the-art methods:

1 **Cross-Event** is proposed by Liao and Grishman (2010), which uses document level information to improve the performance of event extraction;

2 **JointBeam** is the method proposed by Li et al. (2013), which extracts events based on structure prediction by manually designed features;

3 **DMCNN** is proposed by Chen et al. (2015), which uses dynamic multi-pooling to keep multiple events' information;

4 **PSL** is proposed by Liu et al. (2016b), which uses a probabilistic reasoning model to classify events by using latent and global information to encode the associations between events;

5 **JRNN** is proposed by Nguyen et al. (2016), which uses a bidirectional RNN and manually designed features to jointly extract event triggers and arguments.

6 **dbRNN** is proposed by Sha et al. (2018), which adds dependency bridges over Bi-LSTM for event extraction.

Table 1 shows the overall performance comparing to the above state-of-the-art methods with golden-standard entities. From the table, we can see that our JMEE framework achieves the best F_1 scores for both trigger classification and argument-related subtasks among all the compared methods. There is a significant gain with the trigger classification and argument role labeling performances, which is 2% higher over the best-reported models. These results demonstrate the effectivenesses of our method to incorporate with the graph convolution and syntactic shortcut arcs.

3.3 Effect on Extracting Multiple Events

To evaluate the effect of our framework for alleviating the multiple events phenomenon, we divide the test data into two parts (**1/1** and **1/N**) following Nguyen et al. (2016); Chen et al. (2015) and perform evaluations separately. **1/1** means that one sentence only has one trigger or one argument plays a role in one sentence; otherwise, **1/N** is used.

Table 2 illustrates the performance (F_1 scores) of **JRNN** (Nguyen et al., 2016), **DMCNN** (Chen et al., 2015), the two baseline model **Embedding+T** and **CNN** in Chen et al. (2015) and our framework in trigger classification subtask and argument role labeling subatsk. **Embedding+T** uses word embedding vectors and the traditional sentence-level features in Li et al. (2013), while

Stage	Model	1/1	1/N	all
Trigger	Embedding+T	68.1	25.5	59.8
	CNN	72.5	43.1	66.3
	DMCNN	74.3	50.9	69.1
	JRNN	**75.6**	64.8	69.3
	JMEE	75.2	**72.7**	**73.7**
Argument	Embedding+T	37.4	15.5	32.6
	CNN	51.6	36.6	48.9
	DMCNN	54.6	48.7	53.5
	JRNN	50.0	55.2	55.4
	JMEE	**59.3**	**57.6**	**60.3**

Table 2: System Performance on Single Event Sentences **(1/1)** and Multiple Event Sentences **(1/N)**

CNN is similar to **DMCNN**, except that it applies the standard max-pooling mechanism instead of the dynamic multi-pooling mechanism. We can see that our framework significantly outperforms all the other methods, especially in trigger classification subtask. In the **1/N** data split of triggers, our framework is 7.9% better than the **JRNN**, which demonstrates that our method of leveraging syntactic shortcut arcs and self-attention aggregation mechanism is helpful in alleviating the multiple events phenomenon.

3.4 Analysis of Self-Attention Mechanism

We use a sentence "police have arrested four people in connection with the killing" as an example to illustrate the captures features in our self-attention aggregation mechanism by transforming the attention scores to a row-wise heap map in Figure 4. There are two events in the sentence: an *Arrest-Jail* event triggered by *arrested* and a *Die* event triggered by *killings*. Additionally, the entity *police* plays an *Agent* role and the entity *four people* plays a *Person* role in the *Arrest-Jail* event.

As we can see from the Figure 4, in the row of *arrested*, there are relatively strong connections with *arrested* (self), *four people* (its argument) and *killings* (other event). And in the row of *killings*, there are also relatively strong connections with *killings* (self) and *arrested* (other event). Besides, the words *police*, *four* and *in* also have high scores with *killings*, which may mean be on account of the context information propagation though syntactic shortcut arcs.

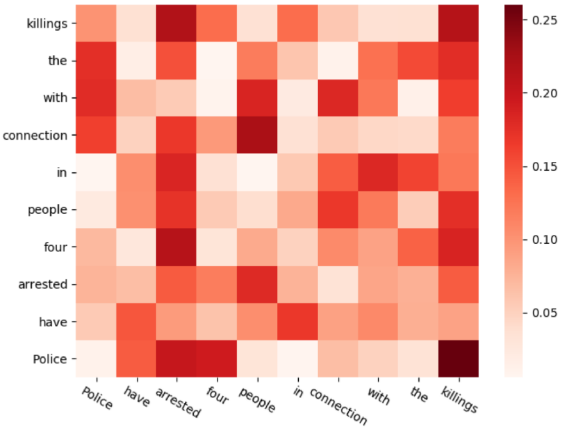

Figure 4: Visualization of the attention scores of a sentence containing two events: an *Arrest-Jail* event triggered by *arrested* and a *Die* event triggered by *killings*. Each row is the group of scores derived by the self-attention aggregation mechanism. Darker red means higher score and stronger interaction.

4 Related Work

There are several existing approaches exploiting the associations between events in the event extraction task. Some of them alleviate this phenomenon by exploiting various sentence-level features, such as ranking dependencies (McClosky et al., 2011), combinational features of triggers and arguments (Li et al., 2013), probabilistic soft logic information (Liu et al., 2016b,a), trigger-specific features and relational features (Yang and Mitchell, 2016; Keith et al., 2017). Others also seek features in document-level methods (Liao and Grishman, 2010; Ji and Grishman, 2008; Hong et al., 2011; Reichart and Barzilay, 2012; Lu and Roth, 2012). The feature-based methods require extensive human engineering, which also essentially affects model performances, and learn them from the unbalanced training data, however, it is difficult for sparse events.

There are also a group of deep learning methods using RNNs (Nguyen et al., 2016; Sha et al., 2018; Liu et al., 2018) and CNNs (Chen et al., 2015; Feng et al., 2016; Nguyen and Grishman, 2016) capturing the associations between events. However, sentence-level sequential modeling methods suffer a lot from the low efficiency in capturing very long-range dependencies. Besides, these methods do not fully model the associations between events.

5 Conclusion and Future Work

This paper presents a novel deep neural jointly multiple events extraction (JMEE) framework for the task of event extraction, especially for alleviating the multiple-event phenomenon. In our framework, we introduce syntactic shortcut arcs to enhance information flow and adapt the graph convolution network to capture the enhanced representation. Then a self-attention aggregation mechanism is applied to aggregate the associations between events. Besides, we jointly extract event triggers and arguments by optimizing a biased loss function due to the imbalances in the dataset. The experiment results demonstrate the effectiveness of our proposed framework. In the future, we plan to exploit the information of one argument which plays different roles in various events to do better in event extraction task.

Acknowledgments

We would like to thank Yansong Feng, Ying Zeng, Xiaochi Wei, Qian Liu and Changsen Yuan for their insightful comments and suggestions. We also very appreciate the comments from anonymous reviewers which will help further improve our work. This work is supported by National Natural Science Foundation of China (No. 61751201 and No. 61602490) and National Key R&D Plan (No. 2017YFB0803302).

References

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 167–176.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 933–941.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 66–71.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 315–323.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyu's english ace 2005 system description. *Journal on Satisfiability*, 51(11):1927–1938.

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

Yu Hong, Jianfeng Zhang, Bin Ma, Jian-Min Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *roceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1127–1136.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 254–262.

Katherine A. Keith, Abram Handler, Michael Pinkham, Cara Magliozzi, Joshua McDuffie, and Brendan O'Connor. 2017. Identifying civilians killed by police with distantly supervised entity-event extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1547–1557.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 73–82.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with abstract meaning representation. In *Proceedings of the 1st Workshop on Computing News Storylines*, pages 11–15.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018. Event detection via gated multilingual attention mechanism. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4865–4872.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016a. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2134–2143.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1789–1798.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016b. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2993–2999.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 835–844.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1626–1635.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5900–5907.

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

Roi Reichart and Regina Barzilay. 2012. Multi-event extraction guided by global constraints. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 70–79.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge RNN and tensor-based argument interaction. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5916–5923.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*, pages 2377–2385.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

RESIDE: Improving Distantly-Supervised Neural Relation Extraction using Side Information

Shikhar Vashishth[1] **Rishabh Joshi**[2] * **Sai Suman Prayaga**[1]
Chiranjib Bhattacharyya[1] **Partha Talukdar**[1]
[1] Indian Institute of Science
[2] Birla Institute of Technology and Science, Pilani
`{shikhar,chiru,ppt}@iisc.ac.in`
`f2014102@pilani.bits-pilani.ac.in, suman.sai14@gmail.com`

Abstract

Distantly-supervised Relation Extraction (RE) methods train an extractor by automatically aligning relation instances in a Knowledge Base (KB) with unstructured text. In addition to relation instances, KBs often contain other relevant side information, such as aliases of relations (e.g., *founded* and *co-founded* are aliases for the relation *founderOfCompany*). RE models usually ignore such readily available side information. In this paper, we propose RESIDE, a distantly-supervised neural relation extraction method which utilizes additional side information from KBs for improved relation extraction. It uses entity type and relation alias information for imposing soft constraints while predicting relations. RESIDE employs Graph Convolution Networks (GCN) to encode syntactic information from text and improves performance even when limited side information is available. Through extensive experiments on benchmark datasets, we demonstrate RESIDE's effectiveness. We have made RESIDE's source code available to encourage reproducible research.

1 Introduction

The construction of large-scale Knowledge Bases (KBs) like Freebase (Bollacker et al., 2008) and Wikidata (Vrandečić and Krötzsch, 2014) has proven to be useful in many natural language processing (NLP) tasks like question-answering, web search, etc. However, these KBs are not exhaustive. Relation Extraction (RE) attempts to fill this gap by extracting semantic relationships between entity pairs from plain text. This task can be modeled as a simple classification problem after the entity pairs are specified. Formally, given an entity pair (e_1,e_2) from the KB and an entity annotated sentence (or instance), we aim to predict the relation r, from a predefined relation set, that exists between e_1 and e_2. If no relation exists, we simply label it *NA*.

Most supervised relation extraction methods require large labeled training data which is expensive to construct. Distant Supervision (DS) (Mintz et al., 2009) helps with the construction of this dataset automatically, under the assumption that if two entities have a relationship in a KB, then all sentences mentioning those entities express the same relation. While this approach works well in generating large amounts of training instances, the DS assumption does not hold in all cases. Riedel et al. (2010); Hoffmann et al. (2011); Surdeanu et al. (2012) propose multi-instance based learning to relax this assumption. However, they use NLP tools to extract features, which can be noisy.

Recently, neural models have demonstrated promising performance on RE. Zeng et al. (2014, 2015) employ Convolutional Neural Networks (CNN) to learn representations of instances. For alleviating noise in distant supervised datasets, attention has been utilized by (Lin et al., 2016; Jat et al., 2018). Syntactic information from dependency parses has been used by (Mintz et al., 2009; He et al., 2018) for capturing long-range dependencies between tokens. Recently proposed Graph Convolution Networks (GCN) (Defferrard et al., 2016) have been effectively employed for encoding this information (Marcheggiani and Titov, 2017; Bastings et al., 2017). However, all the above models rely only on the noisy instances from distant supervision for RE.

Relevant side information can be effective for improving RE. For instance, in the sentence, *Microsoft was started by Bill Gates.*, the type information of *Bill Gates (person)* and *Microsoft (organization)* can be helpful in predicting the correct relation *founderOfCompany*. This is because every relation constrains the type of its target en-

---

[*] Research internship at Indian Institute of Science.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1257–1266
Brussels, Belgium, October 31 - November 4, 2018.

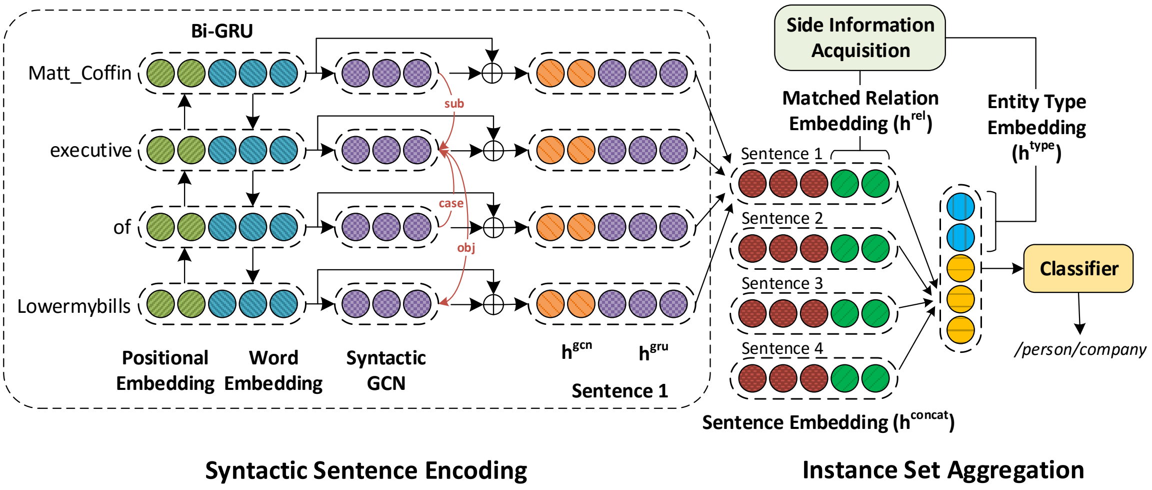

Figure 1: Overview of RESIDE. RESIDE first encodes each sentence in the bag by concatenating embeddings (denoted by ⊕) from Bi-GRU and Syntactic GCN for each token, followed by word attention. Then, sentence embedding is concatenated with relation alias information, which comes from the Side Information Acquisition Section (Figure 2), before computing attention over sentences. Finally, bag representation with entity type information is fed to a softmax classifier. Please see Section 5 for more details.

tities. Similarly, relation phrase *"was started by"* extracted using Open Information Extraction (Open IE) methods can be useful, given that the aliases of relation *founderOfCompany*, e.g., *founded, co-founded, etc.*, are available. KBs used for DS readily provide such information which has not been completely exploited by current models.

In this paper, we propose RESIDE, a novel distant supervised relation extraction method which utilizes additional supervision from KB through its neural network based architecture. RESIDE makes principled use of entity type and relation alias information from KBs, to impose soft constraints while predicting the relation. It uses encoded syntactic information obtained from Graph Convolution Networks (GCN), along with embedded side information, to improve neural relation extraction. Our contributions can be summarized as follows:

- We propose RESIDE, a novel neural method which utilizes additional supervision from KB in a principled manner for improving distant supervised RE.
- RESIDE uses Graph Convolution Networks (GCN) for modeling syntactic information and has been shown to perform competitively even with limited side information.
- Through extensive experiments on benchmark datasets, we demonstrate RESIDE's effectiveness over state-of-the-art baselines.

RESIDE's source code and datasets used in the paper are available at `http://github.com/malllabiisc/RESIDE`.

2 Related Work

Distant supervision: Relation extraction is the task of identifying the relationship between two entity mentions in a sentence. In supervised paradigm, the task is considered as a multi-class classification problem but suffers from lack of large labeled training data. To address this limitation, (Mintz et al., 2009) propose distant supervision (DS) assumption for creating large datasets, by heuristically aligning text to a given Knowledge Base (KB). As this assumption does not always hold true, some of the sentences might be wrongly labeled. To alleviate this shortcoming, Riedel et al. (2010) relax distant supervision for multi-instance single-label learning. Subsequently, for handling overlapping relations between entities (Hoffmann et al., 2011; Surdeanu et al., 2012) propose multi-instance multi-label learning paradigm.

Neural Relation Extraction: The performance of the above methods strongly rely on the quality of hand engineered features. Zeng et al. (2014)

propose an end-to-end CNN based method which could automatically capture relevant lexical and sentence level features. This method is further improved through piecewise max-pooling by (Zeng et al., 2015). Lin et al. (2016); Nagarajan et al. (2017) use attention (Bahdanau et al., 2014) for learning from multiple valid sentences. We also make use of attention for learning sentence and bag representations.

Dependency tree based features have been found to be relevant for relation extraction (Mintz et al., 2009). He et al. (2018) use them for getting promising results through a recursive tree-GRU based model. In RESIDE, we make use of recently proposed Graph Convolution Networks (Defferrard et al., 2016; Kipf and Welling, 2017), which have been found to be quite effective for modelling syntactic information (Marcheggiani and Titov, 2017; Nguyen and Grishman, 2018; Vashishth et al., 2018a).

Side Information in RE: Entity description from KB has been utilized for RE (Ji et al., 2017), but such information is not available for all entities. Type information of entities has been used by Ling and Weld (2012); Liu et al. (2014) as features in their model. Yaghoobzadeh et al. (2017) also attempt to mitigate noise in DS through their joint entity typing and relation extraction model. However, KBs like Freebase readily provide reliable type information which could be directly utilized. In our work, we make principled use of entity type and relation alias information obtained from KB. We also use unsupervised Open Information Extraction (Open IE) methods (Mausam et al., 2012; Angeli et al., 2015), which automatically discover possible relations without the need of any predefined ontology, which is used as a side information as defined in Section 5.2.

3 Background: Graph Convolution Networks (GCN)

In this section, we provide a brief overview of Graph Convolution Networks (GCN) for graphs with directed and labeled edges, as used in (Marcheggiani and Titov, 2017).

3.1 GCN on Labeled Directed Graph

For a directed graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the set of vertices and edges respectively, an edge from node u to node v with label l_{uv} is represented as (u, v, l_{uv}). Since, information in directed edge does not necessarily propagate along its direction, following (Marcheggiani and Titov, 2017) we define an updated edge set $\mathcal{E}'$ which includes inverse edges (v, u, l_{uv}^{-1}) and self-loops $(u, u, \top)$ along with the original edge set $\mathcal{E}$, where $\top$ is a special symbol to denote self-loops. For each node v in $\mathcal{G}$, we have an initial representation $x_v \in \mathbb{R}^d, \forall v \in \mathcal{V}$. On employing GCN, we get an updated d-dimensional hidden representation $h_v \in \mathbb{R}^d, \forall v \in \mathcal{V}$, by considering only its immediate neighbors (Kipf and Welling, 2017). This can be formulated as:

$$h_v = f\left(\sum_{u \in \mathcal{N}(v)} (W_{l_{uv}} x_u + b_{l_{uv}})\right).$$

Here, $W_{l_{uv}} \in \mathbb{R}^{d \times d}$ and $b_{l_{uv}} \in \mathbb{R}^d$ are label dependent model parameters which are trained based on the downstream task. $\mathcal{N}(v)$ refers to the set of neighbors of v based on $\mathcal{E}'$ and f is any non-linear activation function. In order to capture multi-hop neighborhood, multiple GCN layers can be stacked. Hidden representation of node v in this case after k^{th} GCN layer is given as:

$$h_v^{k+1} = f\left(\sum_{u \in \mathcal{N}(v)} \left(W_{l_{uv}}^k h_u^k + b_{l_{uv}}^k\right)\right).$$

3.2 Integrating Edge Importance

In automatically constructed graphs, some edges might be erroneous and hence need to be discarded. Edgewise gating in GCN by (Bastings et al., 2017; Marcheggiani and Titov, 2017) allows us to alleviate this problem by subduing the noisy edges. This is achieved by assigning a relevance score to each edge in the graph. At k^{th} layer, the importance of an edge (u, v, l_{uv}) is computed as:

$$g_{uv}^k = \sigma\left(h_u^k \cdot \hat{w}_{l_{uv}}^k + \hat{b}_{l_{uv}}^k\right), \qquad (1)$$

Here, $\hat{w}_{l_{uv}}^k \in \mathbb{R}^m$ and $\hat{b}_{l_{uv}}^k \in \mathbb{R}$ are parameters which are trained and $\sigma(\cdot)$ is the sigmoid function. With edgewise gating, the final GCN embedding for a node v after k^{th} layer is given as:

$$h_v^{k+1} = f\left(\sum_{u \in \mathcal{N}(v)} g_{uv}^k \times \left(W_{l_{uv}}^k h_u^k + b_{l_{uv}}^k\right)\right). \qquad (2)$$

4 RESIDE Overview

In multi-instance learning paradigm, we are given a bag of sentences (or instances) $\{s_1, s_2, ...s_n\}$ for a given entity pair, the task is to predict the relation between them. RESIDE consists of three components for learning a representation of a given bag, which is fed to a softmax classifier. We briefly present the components of RESIDE below. Each component will be described in detail in the subsequent sections. The overall architecture of RESIDE is shown in Figure 1.

1. **Syntactic Sentence Encoding:** RESIDE uses a Bi-GRU over the concatenated positional and word embedding for encoding the local context of each token. For capturing long-range dependencies, GCN over dependency tree is employed and its encoding is appended to the representation of each token. Finally, attention over tokens is used to subdue irrelevant tokens and get an embedding for the entire sentence. More details in Section 5.1.
2. **Side Information Acquisition:** In this module, we use additional supervision from KBs and utilize Open IE methods for getting relevant side information. This information is later utilized by the model as described in Section 5.2.
3. **Instance Set Aggregation:** In this part, sentence representation from syntactic sentence encoder is concatenated with the *matched relation embedding* obtained from the previous step. Then, using attention over sentences, a representation for the entire bag is learned. This is then concatenated with *entity type embedding* before feeding into the softmax classifier for relation prediction. Please refer to Section 5.3 for more details.

5 RESIDE Details

In this section, we provide the detailed description of the components of RESIDE.

5.1 Syntactic Sentence Encoding

For each sentence in the bag s_i with m tokens $\{w_1, w_2, ...w_m\}$, we first represent each token by k-dimensional GloVe embedding (Pennington et al., 2014). For incorporating relative position of tokens with respect to target entities, we use p-dimensional position embeddings, as used by (Zeng et al., 2014). The combined token embeddings are stacked together to get the sentence representation $\mathcal{H} \in \mathbb{R}^{m \times (k+2p)}$. Then, using Bi-GRU (Cho et al., 2014) over $\mathcal{H}$, we get the new sentence representation $\mathcal{H}^{gru} \in \mathbb{R}^{m \times d_{gru}}$, where d_{gru} is the hidden state dimension. Bi-GRUs have been found to be quite effective in encoding the context of tokens in several tasks (Sutskever et al., 2014; Graves et al., 2013).

Although Bi-GRU is capable of capturing local context, it fails to capture long-range dependencies which can be captured through dependency edges. Prior works (Mintz et al., 2009; He et al., 2018) have exploited features from syntactic dependency trees for improving relation extraction. Motivated by their work, we employ Syntactic Graph Convolution Networks for encoding this information. For a given sentence, we generate its dependency tree using Stanford CoreNLP (Manning et al., 2014). We then run GCN over the dependency graph and use Equation 2 for updating the embeddings, taking $\mathcal{H}^{gru}$ as the input. Since dependency graph has 55 different edge labels, incorporating all of them over-parameterizes the model significantly. Therefore, following (Marcheggiani and Titov, 2017; Nguyen and Grishman, 2018; Vashishth et al., 2018a) we use only three edge labels based on the direction of the edge $\{$*forward* $(\rightarrow)$, *backward* $(\leftarrow)$, *self-loop* $(\top)\}$. We define the new edge label L_{uv} for an edge (u, v, l_{uv}) as follows:

$$L_{uv} = \begin{cases} \rightarrow & \text{if edge exists in dependency parse} \\ \leftarrow & \text{if edge is an inverse edge} \\ \top & \text{if edge is a self-loop} \end{cases}$$

For each token w_i, GCN embedding $h^{gcn}_{i_{k+1}} \in \mathbb{R}^{d_{gcn}}$ after k^{th} layer is defined as:

$$h^{gcn}_{i_{k+1}} = f\left(\sum_{u \in \mathcal{N}(i)} g^k_{iu} \times \left(W^k_{L_{iu}} h^{gcn}_{u_k} + b^k_{L_{iu}} \right) \right).$$

Here, g^k_{iu} denotes edgewise gating as defined in Equation 1 and L_{iu} refers to the edge label defined above. We use ReLU as activation function f, throughout our experiments. The syntactic graph encoding from GCN is appended to Bi-GRU output to get the final token representation, h^{concat}_i as $[h^{gru}_i; h^{gcn}_{i_{k+1}}]$. Since, not all tokens are equally relevant for RE task, we calculate the degree of relevance of each token using attention as used in

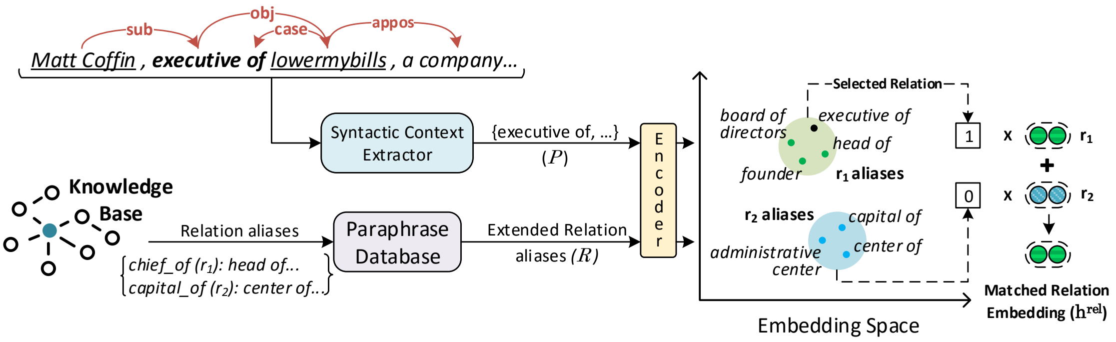

Figure 2: Relation alias side information extraction for a given sentence. First, Syntactic Context Extractor identifies relevant relation phrases $\mathcal{P}$ between target entities. They are then matched in the embedding space with the extended set of relation aliases $\mathcal{R}$ from KB. Finally, the relation embedding corresponding to the closest alias is taken as relation alias information. Please refer Section 5.2.

(Jat et al., 2018). For token w_i in the sentence, attention weight α_i is calculated as:

$$\alpha_i = \frac{\exp(u_i)}{\sum_{j=1}^{m}\exp(u_j)} \quad \text{where, } u_i = h_i^{concat} \cdot r.$$

where r is a random query vector and u_i is the relevance score assigned to each token. Attention values $\{\alpha_i\}$ are calculated by taking softmax over $\{u_i\}$. The representation of a sentence is given as a weighted sum of its tokens, $s = \sum_{j=1}^{m} \alpha_i h_i^{concat}$.

5.2 Side Information Acquisition

Relevant side information has been found to improve performance on several tasks (Ling and Weld, 2012; Vashishth et al., 2018b). In distant supervision based relation extraction, since the entities are from a KB, knowledge about them can be utilized to improve relation extraction. Moreover, several unsupervised relation extraction methods (Open IE) (Angeli et al., 2015; Mausam et al., 2012) allow extracting relation phrases between target entities without any predefined ontology and thus can be used to obtain relevant side information. In RESIDE, we employ Open IE methods and additional supervision from KB for improving neural relation extraction.

Relation Alias Side Information

RESIDE uses Stanford Open IE (Angeli et al., 2015) for extracting relation phrases between target entities, which we denote by $\mathcal{P}$. As shown in Figure 2, for the sentence *Matt Coffin, executive of lowermybills, a company..*, Open IE methods extract *"executive of"* between *Matt Coffin* and *lowermybills*. Further, we extend $\mathcal{P}$ by including tokens at one hop distance in dependency path from target entities. Such features from dependency parse have been exploited in the past by (Mintz et al., 2009; He et al., 2018). The degree of match between the extracted phrases in $\mathcal{P}$ and aliases of a relation can give important clues about the relevance of that relation for the sentence. Several KBs like Wikidata provide such relation aliases, which can be readily exploited. In RESIDE, we further expand the relation alias set using Paraphrase database (PPDB) (Pavlick et al., 2015). We note that even for cases when aliases for relations are not available, providing only the names of relations give competitive performance. We shall explore this point further in Section 7.3.

For matching $\mathcal{P}$ with the PPDB expanded relation alias set $\mathcal{R}$, we project both in a d-dimensional space using GloVe embeddings (Pennington et al., 2014). Projecting phrases using word embeddings helps to further expand these sets, as semantically similar words are closer in embedding space (Mikolov et al., 2013; Pennington et al., 2014). Then, for each phrase $p \in \mathcal{P}$, we calculate its cosine distance from all relation aliases in $\mathcal{R}$ and take the relation corresponding to the closest relation alias as a matched relation for the sentence. We use a threshold on cosine distance to remove noisy aliases. In RESIDE, we define a k_r-dimensional embedding for each relation which we call as *matched relation embedding* (h^{rel}). For a given sentence, h^{rel} is concatenated with its representation s, obtained from syntactic sentence encoder (Section 5.1) as shown in Figure 1. For sentences with $|\mathcal{P}| > 1$, we might get multiple matched relations. In such cases, we take the average of their embeddings. We hypothesize that this helps in improving the performance and find it to be true as shown in Section 7.

Entity Type Side Information

Type information of target entities has been shown to give promising results on relation extraction (Ling and Weld, 2012; Yaghoobzadeh et al., 2017). Every relation puts some constraint on the type of entities which can be its subject and object. For example, the relation *person/place_of_birth* can only occur between a *person* and a *location*. Sentences in distance supervision are based on entities in KBs, where the type information is readily available.

In RESIDE, we use types defined by FIGER (Ling and Weld, 2012) for entities in Freebase. For each type, we define a k_t-dimensional embedding which we call as *entity type embedding* (h^{type}). For cases when an entity has multiple types in different contexts, for instance, *Paris* may have types *government* and *location*, we take the average over the embeddings of each type. We concatenate the *entity type embedding* of target entities to the final bag representation before using it for relation classification. To avoid over-parameterization, instead of using all fine-grained 112 entity types, we use 38 coarse types which form the first hierarchy of FIGER types.

5.3 Instance Set Aggregation

For utilizing all valid sentences, following (Lin et al., 2016; Jat et al., 2018), we use attention over sentences to obtain a representation for the entire bag. Instead of directly using the sentence representation s_i from Section 5.1, we concatenate the embedding of each sentence with *matched relation embedding* h_i^{rel} as obtained from Section 5.2. The attention score α_i for i^{th} sentence is formulated as:

$$\alpha_i = \frac{\exp(\hat{s}_i \cdot q)}{\sum_{j=1}^{n} \exp(\hat{s}_j \cdot q)} \quad \text{where, } \hat{s}_i = [s_i; h_i^{rel}].$$

here q denotes a random query vector. The bag representation $\mathcal{B}$, which is the weighted sum of its sentences, is then concatenated with the *entity type embeddings* of the subject (h_{sub}^{type}) and object (h_{obj}^{type}) from Section 5.2 to obtain $\hat{\mathcal{B}}$.

$$\hat{\mathcal{B}} = [\mathcal{B}; h_{sub}^{type}; h_{obj}^{type}] \quad \text{where, } \mathcal{B} = \sum_{i=1}^{n} \alpha_i \hat{s}_i.$$

Finally, $\hat{\mathcal{B}}$ is fed to a softmax classifier to get the probability distribution over the relations.

$$p(y) = \text{Softmax}(W \cdot \hat{\mathcal{B}} + b).$$

Datasets	Split	# Sentences	# Entity-pairs
Riedel (# Relations: 53)	Train	455,771	233,064
	Valid	114,317	58,635
	Test	172,448	96,678
GIDS (# Relations: 5)	Train	11,297	6,498
	Valid	1,864	1,082
	Test	5,663	3,247

Table 1: Details of datasets used. Please see Section 6.1 for more details.

6 Experimental Setup

6.1 Datasets

In our experiments, we evaluate the models on Riedel and Google Distant Supervision (GIDS) dataset. Statistics of the datasets is summarized in Table 1. Below we described each in detail[1].

1. **Riedel:** The dataset is developed by (Riedel et al., 2010) by aligning Freebase relations with New York Times (NYT) corpus, where sentences from the year 2005-2006 are used for creating the training set and from the year 2007 for the test set. The entity mentions are annotated using Stanford NER (Finkel et al., 2005) and are linked to Freebase. The dataset has been widely used for RE by (Hoffmann et al., 2011; Surdeanu et al., 2012) and more recently by (Lin et al., 2016; Feng et al.; He et al., 2018).
2. **GIDS:** Jat et al. (2018) created Google Distant Supervision (GIDS) dataset by extending the Google relation extraction corpus[2] with additional instances for each entity pair. The dataset assures that the at-least-one assumption of multi-instance learning, holds. This makes automatic evaluation more reliable and thus removes the need for manual verification.

---

[1] Data splits and hyperparameters are in supplementary.

[2] https://research.googleblog.com/2013/04/50000-lessons-on-how-to-read-relation.html

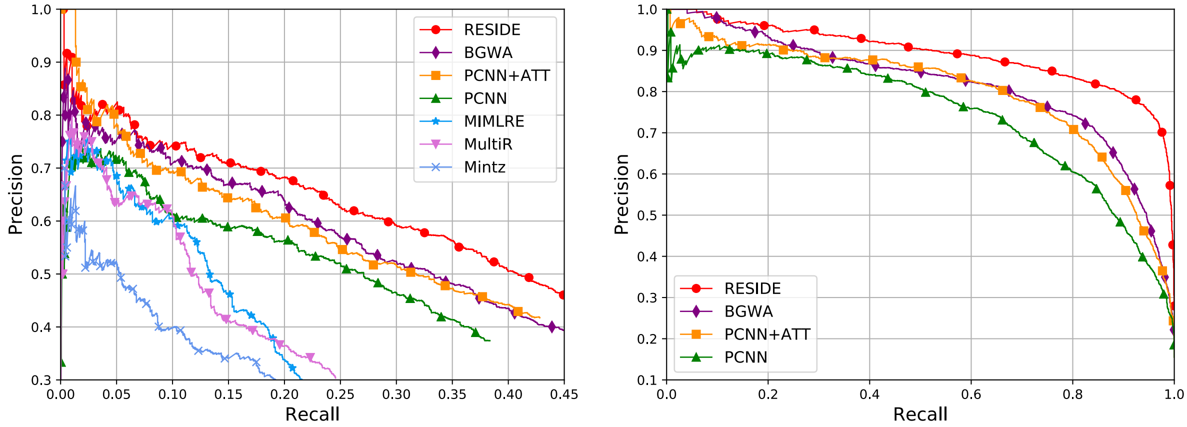

(a) Riedel dataset

(b) GIDS dataset

Figure 3: Comparison of Precision-recall curve. RESIDE achieves higher precision over the entire range of recall than all the baselines on both datasets. Please refer Section 7.1 for more details.

6.2 Baselines

For evaluating RESIDE, we compare against the following baselines:

- **Mintz:** Multi-class logistic regression model proposed by (Mintz et al., 2009) for distant supervision paradigm.
- **MultiR:** Probabilistic graphical model for multi instance learning by (Hoffmann et al., 2011)
- **MIMLRE:** A graphical model which jointly models multiple instances and multiple labels. More details in (Surdeanu et al., 2012).
- **PCNN:** A CNN based relation extraction model by (Zeng et al., 2015) which uses piecewise max-pooling for sentence representation.
- **PCNN+ATT:** A piecewise max-pooling over CNN based model which is used by (Lin et al., 2016) to get sentence representation followed by attention over sentences.
- **BGWA:** Bi-GRU based relation extraction model with word and sentence level attention (Jat et al., 2018).
- **RESIDE:** The method proposed in this paper, please refer Section 5 for more details.

6.3 Evaluation Criteria

Following the prior works (Lin et al., 2016; Feng et al.), we evaluate the models using held-out evaluation scheme. This is done by comparing the relations discovered from test articles with those in Freebase. We evaluate the performance of models with Precision-Recall curve and top-N precision (P@N) metric in our experiments.

7 Results

In this section we attempt to answer the following questions:

Q1. Is RESIDE more effective than existing approaches for distant supervised RE? (7.1)

Q2. What is the effect of ablating different components on RESIDE's performance? (7.2)

Q3. How is the performance affected in the absence of relation alias information? (7.3)

7.1 Performance Comparison

For evaluating the effectiveness of our proposed method, RESIDE, we compare it against the baselines stated in Section 6.2. We use only the neural baselines on GIDS dataset. The Precision-Recall curves on Riedel and GIDS are presented in Figure 3. Overall, we find that RESIDE achieves higher precision over the entire recall range on both the datasets. All the non-neural baselines could not perform well as the features used by them are mostly derived from NLP tools which can be erroneous. RESIDE outperforms PCNN+ATT and BGWA which indicates that incorporating side information helps in improving the performance of the model. The higher performance of BGWA and PCNN+ATT over PCNN shows that attention helps in distant supervised RE. Following (Lin et al., 2016; Liu et al., 2017), we also evaluate our method with different number of sentences. Results summarized in Table 2, show the improved precision of RESIDE in all test settings, as compared to the neural baselines, which demonstrates

	One			Two			All		
	P@100	P@200	P@300	P@100	P@200	P@300	P@100	P@200	P@300
PCNN	73.3	64.8	56.8	70.3	67.2	63.1	72.3	69.7	64.1
PCNN+ATT	73.3	69.2	60.8	77.2	71.6	66.1	76.2	73.1	67.4
BGWA	78.0	71.0	63.3	81.0	73.0	64.0	82.0	75.0	72.0
RESIDE	**80.0**	**75.5**	**69.3**	**83.0**	**73.5**	**70.6**	**84.0**	**78.5**	**75.6**

Table 2: P@N for relation extraction using variable number of sentences in bags (with more than one sentence) in Riedel dataset. Here, One, Two and All represents the number of sentences randomly selected from a bag. RESIDE attains improved precision in all settings. More details in Section 7.1

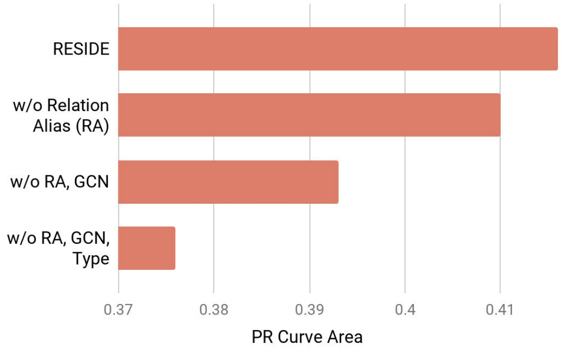

Figure 4: Performance comparison of different ablated version of RESIDE on Riedel dataset. Overall, GCN and side information helps RESIDE improve performance. Refer Section 7.2.

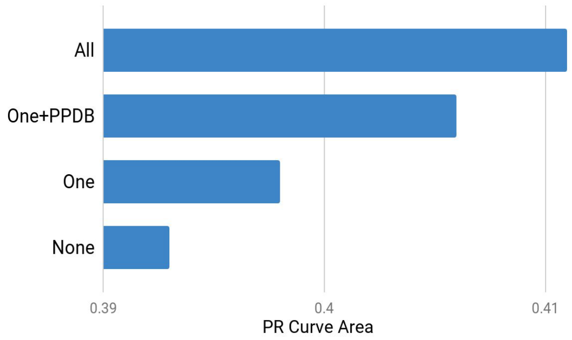

Figure 5: Performance on settings defined in Section 7.3 with respect to the presence of relation alias side information on Riedel dataset. RESIDE performs comparably in the absence of relations from KB.

the efficacy of our model.

7.2 Ablation Results

In this section, we analyze the effect of various components of RESIDE on its performance. For this, we evaluate various versions of our model with cumulatively removed components. The experimental results are presented in Figure 4. We observe that on removing different components from RESIDE, the performance of the model degrades drastically. The results validate that GCNs are effective at encoding syntactic information. Further, the improvement from side information shows that it is complementary to the features extracted from text, thus validating the central thesis of this paper, that inducing side information leads to improved relation extraction.

7.3 Effect of Relation Alias Side Information

In this section, we test the performance of the model in setting where relation alias information is not readily available. For this, we evaluate the performance of the model on four different settings:

- **None:** Relation aliases are not available.
- **One:** The name of relation is used as its alias.
- **One+PPDB:** Relation name extended using Paraphrase Database (PPDB).
- **All:** Relation aliases from Knowledge Base[3]

The overall results are summarized in Figure 5. We find that the model performs best when aliases are provided by the KB itself. Overall, we find that RESIDE gives competitive performance even when very limited amount of relation alias information is available. We observe that performance improves further with the availability of more alias information.

8 Conclusion

In this paper, we propose RESIDE, a novel neural network based model which makes principled use of relevant side information, such as entity type and relation alias, from Knowledge Base, for improving distant supervised relation extraction. RESIDE employs Graph Convolution Networks for

---

[3] Each relation in Riedel dataset is manually mapped to corresponding Wikidata property for getting relation aliases. Few examples are presented in supplementary material.

encoding syntactic information of sentences and is robust to limited side information. Through extensive experiments on benchmark datasets, we demonstrate RESIDE's effectiveness over state-of-the-art baselines. We have made RESIDE's source code publicly available to promote reproducible research.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. This work is supported in part by the Ministry of Human Resource Development (Government of India), CAIR (DRDO) and by a gift from Google.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL (1)*, pages 344–354. The Association for Computer Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv e-prints*, abs/1409.0473.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 3844–3852, USA. Curran Associates Inc.

Xiaocheng Feng, Jiang Guo, Bing Qin, Ting Liu, and Yongjie Liu. Effective deep memory networks for distant supervised relation extraction.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

A. Graves, A. r. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

Zhengqiu He, Wenliang Chen, Zhenghua Li, Meishan Zhang, Wei Zhang, and Min Zhang. 2018. See: Syntax-aware entity embedding for neural relation extraction.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

S. Jat, S. Khandelwal, and P. Talukdar. 2018. Improving Distantly Supervised Relation Extraction using Word and Entity Based Attention. *ArXiv e-prints*.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 94–100. AAAI Press.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1790–1795. Association for Computational Linguistics.

Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. 2014. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *COLING*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *CoRR*, abs/1703.04826.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 523–534, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Tushar Nagarajan, Sharmistha Jat, and Partha Talukdar. 2017. Candis: Coupled & attention-driven neural distant supervision. *CoRR*, abs/1710.09942.

Thien Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 425–430.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Shikhar Vashishth, Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018a. Dating documents using graph convolution networks. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018), July 15-20, 2018, Melbourne, Australia*.

Shikhar Vashishth, Prince Jain, and Partha Talukdar. 2018b. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1317–1327, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1183–1194. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Collective Event Detection via a Hierarchical and Bias Tagging Networks with Gated Multi-level Attention Mechanisms

Yubo Chen[1], **Hang Yang**[1], **Kang Liu**[1,2], **Jun Zhao**[1,2] and **Yantao Jia**[3]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Huawei Technologies Co., Ltd, Beijing, 100085, China
{yubo.chen, hang.yang, kliu, jzhao}@nlpr.ia.ac.cn, jamaths.h@163.com

Abstract

Traditional approaches to the task of ACE event detection primarily regard multiple events in one sentence as independent ones and recognize them separately by using sentence-level information. However, events in one sentence are usually interdependent and sentence-level information is often insufficient to resolve ambiguities for some types of events. This paper proposes a novel framework dubbed as **H**ierarchical and **B**ias **T**agging **N**etworks with **G**ated **M**ulti-level **A**ttention Mechanisms (HBTNGMA) to solve the two problems simultaneously. Firstly, we propose a hierarchical and bias tagging networks to detect multiple events in one sentence collectively. Then, we devise a gated multi-level attention to automatically extract and dynamically fuse the sentence-level and document-level information. The experimental results on the widely used ACE 2005 dataset show that our approach significantly outperforms other state-of-the-art methods.

1 Introduction

Event detection (ED) is a crucial subtask of event extraction, which aims to identify event triggers and classify them into specific types from texts. According to the task defined in Automatic Context Extraction[1] (ACE), given the following sentence S1, a robust ED system should be able to recognize two events: a *Die* event triggered by *died* and an *Attack* event triggered by *fired*.

S1: *In Baghdad, a cameraman* ***died*** *when an American tank* ***fired*** *on the Palestine Hotel.*

To this end, most methods (Ahn, 2006; Hong et al., 2011; Chen et al., 2015; Nguyen and Grishman, 2016; Liu et al., 2017) model ED as a multi-classification task and predict every word in the sentence separately to determine whether it triggers a specific type of event by using sentence-level information. However, they face two problems: (1) Neglecting event interdependency by separately predicting each event; (2) Sentence-level information is usually insufficient to resolve ambiguities for some types of events. In the following, we will use examples to illustrate these two problems specifically.

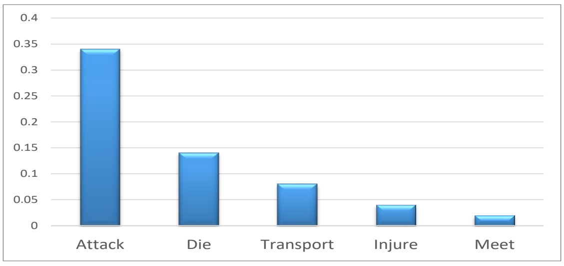

Figure 1: Top 5 event types that co-occur with *Attack* event in the same sentence in ACE 2005.

S2: *The project leader was* ***fired*** *for the* ***bankruptcy*** *of the subsidiary company.*

Event interdependency: In S1, *fired* triggers an *Attack* event, while it triggers an *End-Position* event in S2. Because of the ambiguity, a traditional approach may mislabel *fired* in S1 as a trigger of *End-Position* event. However, if we know *died* triggers a *Die* event in S1, which is easier to disambiguate, we tend to predict that *fired* triggers an *Attack* event. The reason is that the events mentioned in the same sentence tend to be semantically coherent and a *Die* event usually co-occurs with an *Attack* event. The similar phenomenon can be found in S2. We conduct a statistical analysis on ACE 2005 dataset, and find that nearly 30% sentences contain multiple events which is a proportion we can not ignore. To give an intuitive illustration, the top 5 event types that co-occur with *Attack* event in the same sentence are shown in Figure 1. We call such clues as event

---

[1] http://projects.ldc.upenn.edu/ace/

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1267–1276
Brussels, Belgium, October 31 - November 4, 2018.

interdependency. Some works (Li et al., 2013; Yang and Mitchell, 2016; Liu et al., 2016b) rely on a set of elaborately designed features and complicated natural language processing (NLP) tools to capture event interdependency. However, these methods lack generalization, take a large amount of human effort and are prone to error propagation problem. Though Nguyen et al. (2016) use a Recurrent Neural Networks (RNN) based classification model to capture the event interdependency between current event candidate and the former (left) predicted events, they miss the event interdependency between current event candidate and the later (right) predicted events, and the later events can not change the type of current event. The reason is that they classify the words of the sentence from left to right one by one and only use the former events to predict the later event types. We claim that both of the former and later predicted events are important to predict the event type of current trigger candidate. For example in S1, the former predicted *Die* event can help us to predict that *fired* triggers an *Attack* event, while in S2 the later predicted *bankruptcy* event can help us to predict that *fired* triggers an *End-Position* event. Thus, how to use a neural-based model to capture all event interdependencies (the interdependencies between the current event candidate and its former/later predicted events) in the whole sentence is a challenging problem.

Sentence-level and document-level information: Besides event interdependency, knowing that *American tank* is a weapon can also give us additional evidence to predict that *fired* triggers an *Attack* event in S1. Similarly in S2, knowing that *project leader* is a job title can also help us to predict that *fired* triggers an *End-Position* event. We call such clues as sentence-level information. However, sometimes it is difficult even for people to classify event types from an isolated sentence. We must resort to document-level information. For example, considering the following sentence with an ambiguous word *left*:

S3: *He **left** the company.*

It is hard to tell *left* triggers a *Transport* event which means that he left the place, or an *End-Position* event which means that he resigned from the company. However, if we read the whole document, a clue like "*He planned to go shopping before he went home, because he got off work early today.*" would give us more confidence to believe that *left* triggers a *Transport* event, while a clue like "*They held a party for his retirement.*" would indicate the aforementioned event is an *End-Position* event. We call such clues as document-level information. Moreover, the confidence of sentence-level and document-level information should be taken into consideration when using them together to construct a broader range of contextual information. For example in S3, document-level information will give us more evidence, while in S1 sentence-level information is enough to disambiguate the types of events. There have been some feature-based studies (Ji and Grishman, 2008; Liao and Grishman, 2010; Huang and Riloff, 2012) that construct rules to capture document-level information for improving sentence-level ED. However, they suffer from two problems: (1) The features they used often need to be manually designed and may involve error propagation from existing NLP tools; (2) Sentence-level and document-level information are integrated by a large number of fixed rules, which is complicated to construct and it will be far from complete. Thus, how to use a neural-based model to automatically extract sentence-level and document-level information and dynamically integrate them is another challenging problem.

In this paper, we propose a **H**ierarchical and **B**ias **T**agging **N**etworks with **G**ated **M**ulti-level **A**ttention Mechanisms (HBTNGMA) to address the two problems stated above simultaneously. To capture event interdependency and collectively detect multiple events in one sentence, we propose a hierarchical and bias tagging networks for event detection. In which, we exploit a hierarchical RNN-based tagging layer to capture all event interdependencies in the whole sentence and devise a bias objective function to reinforce the influence of trigger tags on the model[2]. To use a broader range of contextual information of the event candidate, we propose a gated multi-level attention, which can automatically extract sentence-level and document-level information and integrate them dynamically. In summary, the contributions of this paper are as follows:

- We propose a novel framework for event detection, which can automatically extract

---

[2] Compared with the task like Named Entities Recognition, the number of "O" tags is much more than the number of trigger tags in ED task, i.e. if we use the non-bias objective function and tag all words in one sentence as "O", we will gain a low loss. Thus we devise a bias objective function.

and dynamically integrate sentence-level and document-level information and collectively detect multiple events in one sentence.

- To capture event interdependency, we exploit a hierarchical and bias tagging networks to detect multiple events in one sentence collectively. To automatically extract and dynamically integrate contextual information, we devise a gated multi-level attention Mechanisms. To our knowledge, this is the first work to jointly use event interdependency, sentence-level information and document-level information via a neural tagging schema for event detection task.
- We conduct extensive experiments on a widely used ACE 2005 dataset, and the experimental results show that our approach significantly outperforms other state-of-the-art methods [3].

2 Task Description

Event detection (ED) is a crucial subtask of event extraction (EE). In this paper, we focus on ED task defined in ACE evaluation, where an event is defined as a specific occurrence involving one or more participants. Firstly, we introduce some ACE terminology to facilitate the understanding of this task: **Event trigger**: the main word or phrase that most clearly expresses the occurrence of an event. **Event arguments**: the mentions that are involved in an event (viz., participants). **Event mention**: a phrase or sentence within which an event is described, including a trigger and arguments.

Given an English text document, an ED system should identify event triggers and categorize their event types for each sentence. For instance, in the sentence "*He died in the hospital*", an ED system is expected to detect a *Die* event along with the trigger word "*died*". The ACE 2005 evaluation defines 8 event types and 33 subtypes, such as *Attack* or *Die*. Following previous works (Li et al., 2013; Chen et al., 2015; Liu et al., 2017; Nguyen and Grishman, 2016), we categorize triggers into these 33 subtypes.

3 Methodology

In this paper, we formulate event detection as a sequence labelling task. As shown in Figure 2, we label all words in one sentence collectively via a **H**ierarchical and **B**ias **T**agging **N**etworks with **G**ated **M**ulti-level **A**ttention Mechanisms (HBT-NGMA). We assign a tag for each word to indicate whether it triggers a specific type of event. We adopt the "BIO" tags schema, where tag "O" represent the "other" tag which means that the corresponding word does not trigger any event, tags "B-EventType" and "I-EventType" represent the "Begin-EventType" and "Inside-EventType" tag respectively. "EventType" means that the word triggers a specific type of event. "B" and "I" represent the position of the word in a trigger to solve the problem that a trigger word contains multiple words such as "take over", "go off" and so on. Thus, the total number of tags is $N_t = 2 * |N_{eventType}| + 1$, where $|N_{eventType}|$ is the size of the predefined event types and $|N_{eventType}| = 33$ in this paper as stated above.

Figure 2 describes the architecture of HBT-NGMA, which primarily involves the following four components: (i) embedding layer, which transforms each word into a continuous vector; (ii) BiLSTM layer, which uses a Bidirectional Long Short Term Memory (BiLSTM) to encode the semantics of each word considering the forward and backward information; (iii) gated multi-level attention, in which we propose a sentence-level and document-level attention to automatically extract sentence-level and document-level information respectively, and we devise a fusion gate to dynamically integrate them as context information; and (iv) Hierarchical tagging layer, in which we propose two Tagging LSTM (TLSTM1 and TLSTM2) and a tagging attention to automatically capture the event interdependency and tag all the words of the sequence collectively.

3.1 Embedding Layer

This paper uses the learned word embeddings as the source of basic features. Specifically, we use the Skip-gram model (Mikolov et al., 2013) to learn word embeddings on the NYT corpus.

Given a document $d = \{s_1, s_2, ..., s_i, ..., s_{N_s}\}$, where N_s is the number of sentences in the document. The i-th sentence s_i can be represented as token sequence $s_i = \{w_1, w_2, ..., w_t, ..., w_{N_w}\}$, where N_w is length of the sentence and w_t is the t-th token of the sentence. Assume that the word embedding for token w_t is e_t and we use it as the input of the following layer.

---

[3] Our source code, including all hyper-parameter settings and pre-trained word embeddings, is openly available at https://github.com/yubochen/NBTNGMA4ED

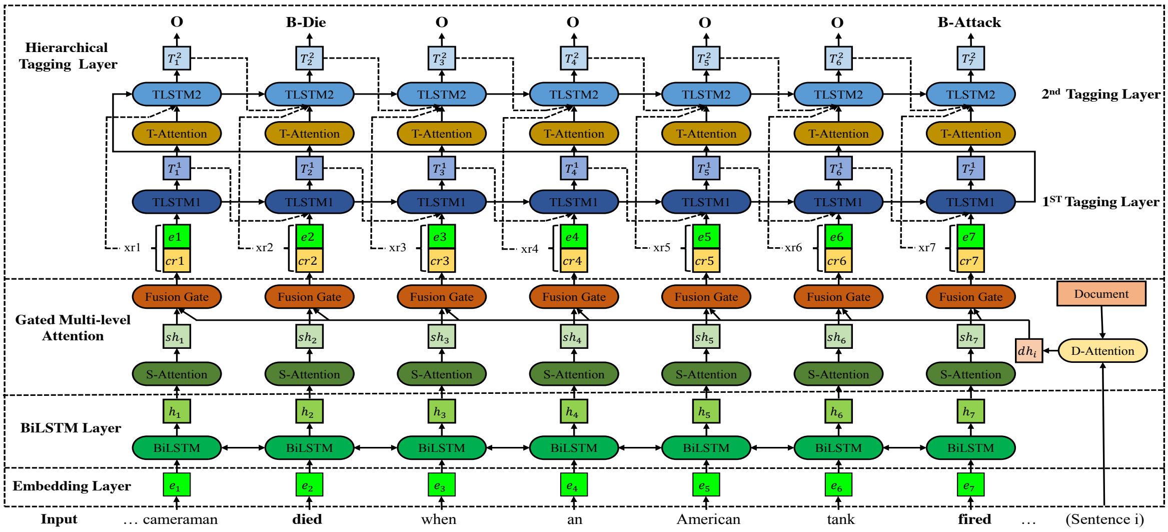

Figure 2: The architecture of our proposed hierarchical and bias tagging networks with gated multi-level attention.

3.2 BiLSTM Layer

In sequence labelling problems, the BiLSTM has been proven effective to capture the semantic information of each word (Lample et al., 2016). In this paper, we use the LSTM unit as described in (Zaremba and Sutskever, 2014). For each word w_t, the forward LSTM encodes w_t by considering the contextual information from word w_1 to w_t, which is marked as $\overrightarrow{h}_t$. Similarly, the backward LSTM encodes w_t based on the contextual information from w_{N_w} to w_t, which is marked as $\overleftarrow{h}_t$. Finally, we concatenate $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ to represent the information of the word w_t, denoted as $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$, and we concatenate $\overrightarrow{h}_{N_w}$ and $\overleftarrow{h}_1$ to represent the encoding information of the whole sentence s_i, denoted as $h_{s_i} = [\overrightarrow{h}_{N_w}, \overleftarrow{h}_1]$.

3.3 Gated Multi-level Attention

Gated multi-level attention primarily involves the following three components: (i) sentence-level attention layer, which automatically captures important sentence-level information by considering the current word; (ii) document-level attention layer, which automatically captures important document-level information by considering the current sentence; and (iii) fusion gate layer, which use a fusion gate to dynamically integrate sentence-level and document-level information.

Sentence-level Attention Layer

Sentence-level attention layer aims to capture the important clues in sentence level. For each candidate word w_t in the sentence, its sentence-level semantic information sh_t is calculated as follows:

$$sh_t = \sum_{k=1}^{N_w} \alpha_s^k h_k \tag{1}$$

where α_s^k is the weight of each word representation h_k. In this paper, we define α_s^k as following:

$$\alpha_s^k = \frac{exp(z_s^k)}{\sum_{j=1}^{N_w} exp(z_s^j)} \tag{2}$$

where z_s^k is the relatedness between the t-th word representation h_t and the k-th word representation h_k, modeled by bilinear attention as:

$$z_s^k = \tanh(h_t W_{sa} h_k^T + b_{sa}) \tag{3}$$

where W_{sa} is the weight matrix and b_{sa} is the bias term. Following above sentence-level attention mechanism, we can get the sentence-level information for each word w_t by considering the semantic information of the word w_t.

Document-level Attention Layer

Similar to sentence-level attention, document-level attention captures the vital clues in the document level. The document-level semantic information dh_i for i-th sentence calculated as follows:

$$\begin{aligned} dh_i &= \sum_{k=1}^{N_s} \alpha_d^k h_{s_k} \\ \alpha_d^k &= \frac{exp(z_d^k)}{\sum_{j=1}^{N_s} exp(z_d^j)} \\ z_d^k &= \tanh(h_{s_i} W_{da} h_{s_k}^T + b_{da}) \end{aligned} \tag{4}$$

where α_d^k is the weight of each sentence representation h_{s_k}, z_d^k is the relatedness between i-th sentence representation h_{s_i} and the k-th sentence representation h_{s_k}, W_{da} is the weight matrix and b_{da} is the bias term. Compared with sentence-level information, all words in the i-th sentence have the same document-level information dh_i.

Fusion Gate Layer

We devise a fusion gate to dynamically integrate sentence-level information sh_t and document-level information dh_i for the t-th word w_t in the i-th sentence s_i, and calculate the contextual information representation cr_t as follows:

$$cr_t = (G_t \odot sh_t) + ((1 - G_t) \odot dh_i) \tag{5}$$

where G_t is a fusion gate aims to model the confidence of clues provided by sentence-level information sh_t and document-level information dh_i, which is calculated as follows:

$$G = \sigma(W_g[sh_t, dh_i] + b_g) \tag{6}$$

where W_g is the weight matrix and b_g is the bias term, σ is a $sigmoid$ function and $\odot$ denotes element-wise multiplication. Finally, the contextual information cr_t of word w_t and its word embedding e_t are concatenated into a single vector $xr_t = [e_t, cr_t]$ as the feature representation of w_t.

3.4 Hierarchical Tagging Layer

In hierarchical tagging layer, we propose two Tagging LSTMs (TLSTM1 and TLSTM2) and a tagging attention to automatically capture the event interdependency and tag the sequence collectively.

The First Tagging Layer: TLSTM1

When detecting the tag of word w_t in TLSTM1, the inputs are: the feature representation xr_t obtained from embedding layer and gated multi-level attention layer, former predicted tag vector T^1_{t-1}, and former hidden vector h^1_{t-1} in TLSTM1. The detail operations are defined as follows:

$$\begin{aligned}
i^1_t &= \sigma(W^1_{i_x} xr_t + W^1_{i_h} h^1_{t-1} + W^1_{i_T} T^1_{t-1}) \\
f^1_t &= \sigma(W^1_{f_x} xr_t + W^1_{f_h} h^1_{t-1} + W^1_{f_T} T^1_{t-1}) \\
o^1_t &= \sigma(W^1_{o_x} xr_t + W^1_{o_h} h^1_{t-1} + W^1_{o_T} T^1_{t-1}) \\
u^1_t &= \varphi(W^1_{u_x} xr_t + W^1_{u_h} h^1_{t-1} + W^1_{u_T} T^1_{t-1}) \\
c^1_t &= i^1_t \odot u^1_t + f^1_t \odot c^1_{t-1} \\
h^1_t &= o^1_t \odot \varphi(c^1_t) \\
T^1_t &= W^1_T h^1_t + b^1_T
\end{aligned} \tag{7}$$

Where i_t is an input gate, u_t is an input modulation gate, f_t is a forget gate, o_t is an output gate, c_t is a memory cell and T^1_t is a predicted tagging vector.

The Second Tagging Layer: TLSTM2

Though the TLSTM1 can capture the interdependency between current event candidate and the former predicted event tags, it can not capture the interdependency between the current event candidate and the later predicted event tags. Thus, we devise a second tagging layer (TLSTM2) upon the LSTM1 to capture the interdependency between the current event candidate and both of former and later predicted event tags from TLSTM1. When detecting the tag of word w_t in TLSTM2, the inputs are: the feature representation xr_t, former predicted tag vector T^2_{t-1} in TLSTM2, the preliminary predicted information T^a_t calculated from TLSTM1, and former hidden vector h^2_{t-1} in TLSTM2. The detail operations are defined as follows:

$$\begin{aligned}
i^2_t &= \sigma(W^2_{i_x} xr_t + W^2_{i_h} h^2_{t-1} + W^2_{i_T} T^2_{t-1} + W^2_{i_a} T^a_t) \\
f^2_t &= \sigma(W^2_{f_x} xr_t + W^2_{f_h} h^2_{t-1} + W^2_{f_T} T^2_{t-1} + W^2_{f_a} T^a_t) \\
o^2_t &= \sigma(W^2_{o_x} xr_t + W^2_{o_h} h^2_{t-1} + W^2_{o_T} T^2_{t-1} + W^2_{o_a} T^a_t) \\
u^2_t &= \varphi(W^2_{u_x} xr_t + W^2_{u_h} h^2_{t-1} + W^2_{u_T} T^2_{t-1} + W^2_{u_a} T^a_t) \\
c^2_t &= i^2_t \odot u^2_t + f^2_t \odot c^2_{t-1} \\
h^2_t &= o^2_t \odot \varphi(c^2_t) \\
T^2_t &= W^2_T h^2_t + b^2_T
\end{aligned} \tag{8}$$

The unit structure of TLSTM2 is similar to the unit of TLSTM1. The parts need to pay attention are follows: (1) the initial hidden input h^2_0 of the TLSTM2 is the last hidden vector $h^1_{N_w}$ of the TLSTM1. (2) the preliminary predicted information T^a_t is calculated from TLSTM1 by using a tagging attention as follows.

Tagging Attention

Tagging attention aims to automatically encode the preliminary predicted information T^a_t for the word w_t and the details are as follows:

$$\begin{aligned}
T^a_t &= \sum\nolimits_{k=1}^{N_w} \alpha^k_T T^1_k \\
\alpha^k_T &= \frac{exp(z^k_T)}{\sum^{N_w}_{j=1} exp(z^j_T)} \\
z^k_T &= \tanh(T^1_t W_{ta} (T^1_k)^T + b_{ta})
\end{aligned} \tag{9}$$

where α^k_T is the weight of each preliminary predicted tag T^1_k, z^k_T is the relatedness between t-th preliminary predicted tag T^1_t and the k-th preliminary predicted tag T^1_k, W_{ta} is the weight matrix and b_{ta} is the bias term.

The final normalized tag probability for word w_t is based on the predicted tag vector T^2_t from

TLSTM2 and computed as follows:

$$\begin{aligned} O_t &= W_y T_t^2 + b_y \\ p(O_t^i|s_j,\theta) &= \frac{exp(O_t^i)}{\sum_{k=1}^{N_t} exp(O_t^k)} \end{aligned} \tag{10}$$

where $p(O_t^i|s_j,\theta)$ is the probability that assigning the i-th tag to word w_t in sentence s_j when parameters is θ, and N_t is the total number of tags.

3.5 Training with Bias Objective Function

In one sentence, the number of “O” tags is much more than the number of trigger tags. Thus, we devise a bias objective function $J(\theta)$ to reinforce the influence of trigger tags on the model, which is defined as follows:

$$\begin{aligned} J(\theta) = max \sum\nolimits_{j=1}^{N_{ts}} \sum\nolimits_{t=1}^{N_w} (\log p(O_t^{y_t}|s_j,\theta) \cdot I(O) \\ + \alpha \log p(O_t^{y_t}|s_j,\theta) \cdot (1 - I(O))) \end{aligned} \tag{11}$$

where N_{ts} is the number of training sentences, N_w is the length of sentence s_j, $p(O_t^{y_t}|s_j,\theta)$ is the normalized probabilities of tags defined in Formula 10 and y_t is the golden tag of word w_t in sentence s_j, α is the bias weight and the larger α will bring the greater influence of trigger tags on the model. Besides, $I(O)$ is a switching function to distinguish the loss of tag “O” and trigger tags, which is defined as follows:

$$I(O) = \begin{cases} 1, & if \quad tag = \text{“O”} \\ 0, & if \quad tag \neq \text{“O”} \end{cases} \tag{12}$$

To compute the network parameter θ, we maximize the log likelihood $J(\theta)$ through stochastic gradient descent over shuffled mini-batches with the Adadelta (Zeiler, 2012) update rule.

4 Experiments

4.1 Experimental Setting

Dataset and Evaluation Metrics

We conduct experiments on the widely used ACE 2005 dataset. For comparison, as the same as previous works (Liao and Grishman, 2010; Li et al., 2013; Chen et al., 2015; Nguyen et al., 2016; Liu et al., 2017), we used the same test set with 40 documents and the same development set with 30 documents and the rest 529 documents are used for training. Finally, we use *Precision* (P), *Recall* (R) and *F measure* (F_1) as the evaluation metrics as the same as previous work.

Hyper-parameter Setting

Hyper-parameters are tuned on the development dataset by grid search. We train the word embedding using Skip-gram algorithm [4] on the NYT corpus [5]. We set the dimension of word embeddings as 100, the dimension of tag vector as 20, all the size of LSTM in BiLSTM layer, TLSTM1 and TLSTM2 layer as 100, the bias parameter α in Formula 11 as 5, the batch size as 20, the learning rate as 0.001, the dropout rate as 0.5.

4.2 Our Method vs. State-of-the-art Methods

We select the following state-of-the-art methods for comparison, which can be classified as two types: separate and collective methods:

Separate methods: 1) **Li’s MaxEnt**: the method that detects events in one sentence separately by using human-designed features (Li et al., 2013). 2) **Liao’s CrossEvent** : the method that uses cross event information (Liao and Grishman, 2010). 3) **Hong’s CrossEntity**: the method that uses cross entity information (Hong et al., 2011). 4) **Chen’s DMCNN**: the dynamic multi-pooling convolutional neural networks method (Chen et al., 2015). 5) **Chen’s DMCNN+**: the DMCNN method argumented with automatically labeled data (Chen et al., 2017). 6) **Liu’s FrameNet** : the method that leverages FrameNet as extended training data to improve ED (Liu et al., 2016a). 7) **Liu’s ANN-Aug**: the method that use the annotated argument information via a supervised attention to improve ED (Liu et al., 2017).

Collective methods: 1) **Li’s Structure**: the method that collectively detects events by using human-designed features (Li et al., 2013). 2) **Yang’s JointEE**: the method that detects events and entities in one sentence jointly based on human-designed features (Yang and Mitchell, 2016). 3) **Nguyen’s JRNN**: the method that exploits a RNN model to collectively detects events by only using sentence-level information (Nguyen et al., 2016). 4) **Liu’s PSL** : the method that uses a probabilistic soft logic to detect events by using human-designed features (Liu et al., 2016b).

Experimental results are shown in Table 1. From the table, we have the following observations: (1) Among all the methods, our HBT-NGMA achieves the best performance. It can improve the best collective method’s F_1 by

---

[4] https://code.google.com/p/word2vec/

[5] https://catalog.ldc.upenn.edu/LDC2008T19

Methods	P	R	F_1
Li's MaxEnt (2013)	74.5	59.1	65.9
Liao's CrossEvent (2010)	68.7	68.9	68.8
Hong's CrossEntity (2011)	72.9	64.3	68.3
Chen's DMCNN (2015)	75.6	63.6	69.1
Chen's DMCNN+ † (2017)	75.7	66.0	70.5
Liu's FrameNet † (2016a)	77.6	65.2	70.7
Liu's ANN-Aug † (2017)	76.8	67.5	71.9
Li's Structure (2013)	73.7	62.3	67.5
Yang's JointEE (2016)	75.1	63.3	68.7
Nguyen's JRNN (2016)	66.0	73.0	69.3
Liu's PSL (2016b)	75.3	64.4	69.4
Ours HBTNGMA	**77.9**	**69.1**	**73.3**

Table 1: Overall performance on blind test data. The upper table illustrates the performance of separate ED systems and the lower illustrates collective ED systems. † means the method that uses external resources.

3.9% (*Liu's PSL*) and improve the best separate method's F_1 by 1.4% (*Liu's ANN-Aug*) although *Liu's ANN-Aug* uses FrameNet as external resources. We also perform a t-test ($p \leqslant 0.05$), which indicates that our method significantly outperforms all of the compared methods. (2) Comparing our HBTNGMA to separate methods, it achieves a better performance. It proves that collectively predicting multiple events in one sentence is effective. (3) Our HBTNGMA outperforms feature-based collective methods (*Li's Structure*, *Yang's JointEE* and *Liu's PSL*), it proves that our automatically learned features can efficiently capture semantic information from plain texts. (4) Compared with Nguyen's JRNN, HBTNGMA gains a 4.0% improvement on F_1 value. The reason is that Nguyen's JRNN only uses sentence-level information while our model exploits multi-level information, and our model can capture the interdependencies between the current event candidate and its former/later predicted events simultaneously.

4.3 Effect of The Hierarchical and Bias Tagging Networks

In this subsection, we prove the effectiveness of hierarchical and bias tagging networks for collective ED. We select following methods as baselines: 1) **LSTM+Softmax**: a simplified version of our HBTNGMA, which directly use a softmax layer to separately detect events after we get the feature representation xr_t of each word w_t. 2) **LSTM+CRF**: the method is similar to our HBTNGMA, which uses a CRF layer to tag words instead of our Hierarchical TLSTM (HTLSTM) tagging layer. 3) **LSTM+TLSTM**: the method is similar to our HBTNGMA, which only use a TLSTM1 and takes all tags have same influence in training loss (i.e. α in is set as 1) . 4) **LSTM+HTLSTM**: the method is similar to our HBTNGMA, which use a HTLSTM (TLSTM1+TLSTM2) and do not use bias objective function. And *LSTM+HTLSTM+Bias* is our proposed HBTNGMA. Moreover, we divide the testing data into two parts according the event number in a sentence (single event and multiple events) and perform evaluations separately.

Method	1/1	1/N	all
LSTM+Softmax	74.7	44.6	66.8
LSTM+CRF	75.1	49.5	68.5
LSTM+TLSTM	76.8	51.2	70.2
LSTM+HTLSTM	77.9	57.3	72.4
LSTM+HTLSTM+Bias	**78.4**	**59.5**	**73.3**

Table 2: Performance of different ED systems. 1/1 means one sentence that only has one event and 1/N means that one sentence has multiple events.

Table 2 shows the results. And we have the following observations: 1) Compared with *LSTM+Softmax*, LSTM-based collective ED methods (*LSTM+CRF*, *LSTM+TLSTM*, *LSTM+HTLSTM*, *LSTM+HTLSTM+Bias*) achieves a better performance. Surprisingly, the *LSTM+HTLSTM+Bias* yields a 14.9% improvement on the sentence contains multiple events over the *LSTM+Softmax*. It proves neural tagging schema is effective for ED task especially for the sentences contain multiple events. 2) The *LSTM+TLSTM* achieve better performances than *LSTM+CRF*. And the *LSTM+HTLSTM* achieve better performances than *LSTM+TLSTM*. The results prove the effectiveness of the TLSTM layer and HTLSTM layer. 3) Compared with *LSTM+HTLSTM*, the *LSTM+HTLSTM+Bias* gains a 0.9% improvement on all sentence. It demonstrates the effectiveness of our proposed bias objective function.

4.4 Effect of The Gated Multi-level Attention

This subsection studies the effectiveness of our gated multi-level attention. We adopt same architecture of our HBTNGMA as shown in Figure 2 with different level clues as baselines: 1) **Word Only** is the method only uses word embedding e_t to identify events. 2) **Word+SA** uses sentence-level attention to capture important sentence-level information as additional clues. 3) **Word+DA** uses document-level attention to capture important document-level information as additional clues. 4) **Word+Average MA** uses both of sentence-level and document-level attention to capture multi-level information and integrate them with a average gate (all the dimension of the fusion gate are set as 0.5), which is a special case of our proposed HBTNGMA. And *Word+Gated MA* is our proposed HBTNGMA model.

Method	P	R	F_1
Word Only	70.1	63.4	66.6
Word+SA	75.6	68.2	71.7
Word+DA	73.1	65.8	69.3
Word+Average MA	76.5	68.7	72.4
Word+Gated MA	**77.9**	69.1	**73.3**

Table 3: Performance of gated multi-level attention.

Results are shown in Table 3. From the results, we have the following observations: 1) Compared with *word only*, *Word+SA* achieves a better performance. We can make the same observation when comparing *Word+DA* with *word only*. It proves that both sentence-level and document-level information are helpful for ED task. 2) Compared with *Word+DA*, *Word+SA* achieves a better performance. It proves that in most of cases sentence-level information provides more clues than document-level information. 3) *Word+Gated MA* gains a 0.9% improvement than *Word+Average MA*. It demonstrates that the effectiveness of our fusion gate to dynamically integrate clues from multiple levels.

4.5 Case Study

Interesting Cases: Our neural tagging schema not only can model the interdependency between multiple events in one sentence as proved in Subsection 4.3, but also the "BIO" tagging schema can solve the multiple words trigger inherently. We conduct a statistical analysis on the experimental results, and find that nearly 50% cases with multiple word trigger was solved by our model. Example is shown in Figure 3.

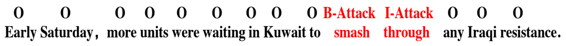

Figure 3: The example of case solved by our model.

Attention Visualization: As limited of space, we take one sentence with high sentence-level gated weight (example 1) and one sentence with high document-level gated weight (example 2) as examples for attention visualization. As shown in Figure 4, in example 1, sentence-level information plays more important role in disambiguating *fired*, and the words (*tank*, *died* and *Baghdad*) give us ample evidence to predict that *fired* triggers an *Attack* event. While document-level information plays a more important role in example 2. The surrounding sentence "*this is ... tiresome*." gives us more confidence to predict that *leave* triggers an *End-Position* event.

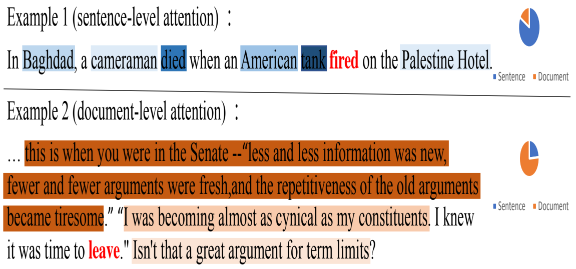
Example 1 (sentence-level attention)：

In Baghdad, a cameraman died when an American tank fired on the Palestine Hotel.

Example 2 (document-level attention)：

... this is when you were in the Senate --"less and less information was new, fewer and fewer arguments were fresh,and the repetitiveness of the old arguments became tiresome." "I was becoming almost as cynical as my constituents. I knew it was time to leave." Isn't that a great argument for term limits?

Figure 4: Attention visualization. The heat map indicate the contextual attention. Blue for sentence-level and orange for document-level. The pie chart indicate the fusion gate weight.

5 Related Works

Event detection is an increasingly hot and challenging research topic in NLP. Generally, existing approaches could roughly be divided into two groups: separate and collective methods.

Separate methods: These methods regard multiple events in one sentence as independent ones and recognize them separately. These methods include feature-based methods which exploit a diverse set of strategies to convert classification clues into feature vectors (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Huang and Riloff, 2012), and neural-based methods which use neural networks to automatically capture clues from plain texts (Chen et al., 2015; Nguyen and Grishman, 2015; Feng et al., 2016; Nguyen and Grishman, 2016; Chen et al., 2017; Duan et al., 2017; Liu et al., 2017). Though effective these methods, they neglect event interdependency by separately predicting each event.

Collective methods: These methods try to model the event interdependency and detect multiple events in one sentence collectively. However, nearly all of these methods are feature-based methods (McClosky et al., 2011; Li et al., 2013; Yang and Mitchell, 2016; Liu et al., 2016b), which rely on elaborately designed features and suffer error propagation from existing NLP tools. Nguyen et al. (2016) exploits a neural-based method to detect multiple events collectively. However, they only use the sentence-level information and neglect document-level clues, and can only capture the interdependencies between the current event candidate and its former predicted events. Moreover, there method can not handle the multiple words trigger problem.

6 Conclusion

This paper proposes a novel framework for event detection, which can automatically extract and dynamically integrate sentence-level and document-level information and collectively detect multiple events in one sentence. A hierarchical and bias tagging networks is proposed to detect multiple events in one sentence collectively. A gated multi-level attention is devised to automatically extract and dynamically integrate contextual information. The experimental results on the widely used dataset prove the effectiveness of the proposed method.

Acknowledgments

The research work is supported by the Natural Science Foundation of China (No.61533018 and No.61702512), and the independent research project of National Laboratory of Pattern Recognition. This work is also supported in part by Huawei Technologies Co., Ltd.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–8.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 409–419, Vancouver, Canada. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistic (ACL)*, pages 167–176.

Shaoyang Duan, Ruifang He, and Wenli Zhao. 2017. Exploiting document level information to improve event detection via recurrent neural networks. In *Proceedings of IJNLP*, volume 1, pages 352–361.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2, pages 66–71.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1127–1136.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Proceedings of AAAI*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 254–262.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*, pages 260–270.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 789–797.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016a. Leveraging framenet to improve automatic event detection. In *Proceedings of ACL*, pages 2134–2143. Association for Computational Linguistics.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1789–1798.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016b. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of AAAI*, pages 2993–2999.

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1626–1635.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Huu Thien Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistic (ACL)*, pages 365–371.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 886–891.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of NAACL*, pages 289–299, San Diego, California. Association for Computational Linguistics.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Valency-Augmented Dependency Parsing

Tianze Shi
Cornell University
tianze@cs.cornell.edu

Lillian Lee
Cornell University
llee@cs.cornell.edu

Abstract

We present a complete, automated, and efficient approach for utilizing valency analysis in making dependency parsing decisions. It includes extraction of valency patterns, a probabilistic model for tagging these patterns, and a joint decoding process that explicitly considers the number and types of each token's syntactic dependents. On 53 treebanks representing 41 languages in the Universal Dependencies data, we find that incorporating valency information yields higher precision and F1 scores on the core arguments (subjects and complements) and functional relations (e.g., auxiliaries) that we employ for valency analysis. Precision on core arguments improves from 80.87 to 85.43. We further show that our approach can be applied to an ostensibly different formalism and dataset, Tree Adjoining Grammar as extracted from the Penn Treebank; there, we outperform the previous state-of-the-art labeled attachment score by 0.7. Finally, we explore the potential of extending valency patterns beyond their traditional domain by confirming their helpfulness in improving PP attachment decisions.[1]

1 Introduction

Many dependency parsers treat attachment decisions and syntactic relation labeling as two independent tasks, despite the fact that relation labels carry important subcategorization information. For example, the number and types of the syntactic arguments that a predicate may take is rather restricted for natural languages — it is not common for an English verb to have more than one syntactic subject or more than two objects.

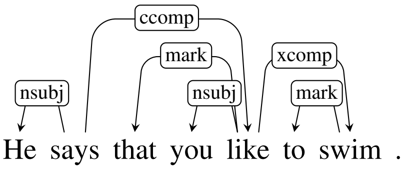

Figure 1: Sample annotation in UD, encoding the core valency pattern nsubj ⋄ ccomp for *says*, nsubj ⋄ xcomp for *like*, and so on (see §2-4)

In this work, we present a parsing approach that explicitly models subcategorization of (some) syntactic dependents as *valency patterns* (see Fig. 1 for examples), and operationalize this notion as extracted supertags. An important distinction from prior work is that our definition of valency-pattern supertags is relativized to a user-specified subset of all possible syntactic relations (see §3). We train supertaggers that assign probabilities of potential valency patterns to each token, and leverages these probabilities during decoding to guide our parsers so that they favor more linguistically plausible output structures.

We mainly focus on two subsets of relations in our analysis, those involving core arguments and those that represent functional relations, and perform experiments over a collection of 53 treebanks in 41 languages from the Universal Dependencies dataset (UD; Nivre et al., 2017). Our valency-aware parsers improve upon strong baseline systems in terms of output linguistic validity, measured as the accuracy of the assigned valency patterns. They also have higher precision and F1 scores on the subsets of relations under analysis, suggesting a potentially controlled way to balance precision-recall trade-offs.

We further show that our approach is not limited to a particular treebank annotation style. We apply our method to parsing another grammar formalism, Tree Adjoining Grammar, where dependency and valency also play an important role in both theory and parser evaluation. Our parser reaches a new state-of-the-art LAS score of 92.59, with more than 0.6 core-argument F1-score improve-

---

[1] Our implementation is available at `https://github.com/tzshi/valency-parser-emnlp18`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1277–1291
Brussels, Belgium, October 31 - November 4, 2018.

ment over our strong baseline parser.

Finally, we demonstrate the applicability of our valency analysis approach to other syntactic phenomena less associated with valency in its traditional linguistic sense. In a case study of PP attachment, we analyze the patterns of two syntactic relations commonly used in PP attachment, and include them in the joint decoding process. Precision of the parsers improves by an absolute 3.30% on these two relation types.

2 Syntactic Dependencies and Valencies

According to Nivre (2005), the modern dependency grammar can be traced back to Tesnière (1959), with its roots reaching back several centuries before the Common Era. The theory is centered on the notion of *dependency*, an asymmetrical relation between words of a sentence. Tesnière distinguishes three node types when analyzing simple predicates: verb equivalents that describe actions and events, noun equivalents as the arguments of the events, and adverb equivalents for detailing the (temporal, spatial, etc.) circumstances. There are two types of relations: (1) verbs dominate nouns and adverbs through a dependency relation; (2) verbs and nouns are linked through a *valency* relation. Tesnière compares a verb to an atom: a verb can attract a certain number of arguments, just as the valency of an atom determines the number of bonds it can engage in (Ágel and Fischer, 2015). In many descriptive lexicographic works (Helbig and Schenkel, 1959; Herbst et al., 2004), valency is not limited to verbs, but also includes nouns and adjectives. For more on the linguistic theory, see Ágel et al. (2003, 2006).

Strictly following the original notion of valency requires distinguishing between arguments and adjuncts, as well as obligatory and optional dependents. However, there is a lack of consensus as to how these categorizations may be distinguished (Tutunjian and Boland, 2008), and thus we adopt a more practical definition in this paper.

3 Computational Representation

Formally, we fix a set of syntactic relations $\mathcal{R}$, and define the *valency pattern* of a token w_i with respect to $\mathcal{R}$ as the linearly-ordered[2] sequence $a_{-j} \cdots a_{-1} \diamond a_1 \cdots a_k$: the $\diamond$ symbol denotes the center word w_i, and each a_l asserts the existence of a word w dominated by w_i via relation $a_l \in \mathcal{R}$, $w_i \xrightarrow{a_l} w$. For a_l and a_m, when $l < m$, the syntactic dependent for a_l linearly precedes the syntactic dependent for a_m. As an example, consider the UD-annotated sentence in Fig. 1. The token *says* has a core-relation[3] valency pattern nsubj $\diamond$ ccomp, and *like* has the pattern nsubj $\diamond$ xcomp. If we consider only functional relations, both *like* and *swim* have the pattern mark $\diamond$.[4] We sometimes employ the abbreviated notation $\alpha^L \diamond \alpha^R$, where α indicates a sequence and the letters L and R distinguish left dependencies from right dependencies.

Dataset	Subset	Syntactic Relations
UD	Core	nsubj, obj, iobj, csubj, ccomp, xcomp
	Func.	aux, cop, mark, det, clf, case
	PP (§8)	nmod, obl
TAG	Core	0 (subject), 1 (object), 2 (indirect object)
	Co-head	CO

Table 1: Sets of syntactic relations we used for valency analysis. UD subsets come from the official categorization in the annotation guidelines.

We make our definition of valency patterns dependent on choice of $\mathcal{R}$ not only because some dependency relations are more often obligatory and closer to the original theoretical definition of valency, but also because the utility of different types of syntactic relations can depend on the downstream task. For example, purely functional dependency labels are semantically vacuous, so they are often omitted in the semantic representations extracted from dependency trees for question answering (Reddy et al., 2016, 2017). There are also recent proposals for parser evaluation that downplay the importance of functional syntactic relations (Nivre and Fang, 2017).

---

[2] Our approach, whose full description is in §5, can be adapted to cases where linear ordering is de-emphasized. The algorithm merely requires a distinction between left and right dependents. We choose to encode linearity since it appears that most languages empirically exhibit word order *preferences* even if they allow for relatively free word order.

[3] UD core and functional relations are listed in Table 1.

[4] The (possibly counterintuitive) direction for *that* and *to* is a result of UD's choice of a content-word-oriented design.

4 Pilot Study: Sanity Checks

We consider two questions that need to be addressed at the outset:[5]

1. How well do the extracted patterns generalize to unseen data?
2. Do state-of-the-art parsers already capture the notion of valency implicitly, though they are not explicitly optimized for it?

The first question checks the feasibility of learning valency patterns from a limited amount of data; the second probes the potential for any valency-informed parsing approach to improve over current state-of-the-art systems.

To answer these questions, we use the UD 2.0 dataset for the CoNLL 2017 shared task (Zeman et al., 2017) and the system outputs[6] of the top five performing submissions (Dozat et al., 2017; Shi et al., 2017b; Björkelund et al., 2017; Che et al., 2017; Lim and Poibeau, 2017). Selection of treebanks is the same as in §6. We extract valency patterns relative to the set of 6 UD core arguments given in Table 1 because they are close to the original notion of valency and we hypothesize that these patterns should exhibit few variations. This is indeed the case: the average number of valency patterns we extract is 110.4 per training treebank, with Turkish (`tr`) having the fewest at 34, and Galician (`gl`) having the most at 298 patterns. We observe that in general, languages with higher degree of flexibility in word order tend to generate more patterns in the data, as our patterns encode linear word order information.

Next, we extract valency patterns from the test set and compare them against those from the training set. On average, out of the 55.4 patterns observed in the gold-standard test sets, only 5.5, or 9.98%, are new and unseen with respect to training. In comparison, 36.2% of the word types appearing in the test sets are not seen during training. This suggests that the valency pattern space is relatively restricted, and the patterns extracted from training sets do generalize well to test sets.

Finally, we consider the average number of valency patterns extracted from the top-performing system outputs and the number of those not observed in training.[7] All 5 systems are remarkably "hallucinatory" in inventing valency relations, introducing 16.8 to 35.5 new valency patterns, significantly larger than the actual number of unseen patterns. Below we show an error committed by the state-of-the-art Dozat et al. (2017) parser (upper half) as compared to the gold-standard annotation (lower half), and we highlight the core argument valency relations of the verb *bothers* in bold. The system incorrectly predicts *how come* to be a clausal subject.

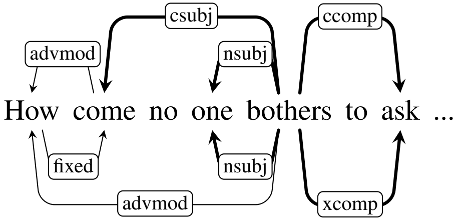

Each such non-existent new pattern implies at least some (potentially small) parsing error that can contribute to the degradation of downstream task performance.

5 Valency-Aware Dependency Parsing

5.1 Overview

Our model is based on the following probability factorization for a given sentence $x = w_1, \ldots, w_n$ and parse tree y for x:

$$P(y|x) = \frac{1}{Z_x} \prod_{i=1}^{n} P(v_i|w_i) P(h_i|w_i) P(r_i|w_i, h_i),$$

where Z_x is the normalization factor, v_i is the valency pattern extracted for w_i from y, h_i is the index of the syntactic governor of w_i, and r_i is the syntactic relation label of the dependency relation between w_{h_i} and w_i. We first assume that we have a feature extractor that associates each token in the sentence w_i with a contextualized feature vector $\mathbf{w}_i$, and explain how to calculate the factored probabilities (§5.2). Then we discuss decoding (§5.3) and training (§5.4). Our decoder can be viewed as a special-case implementation of head-automaton grammars (Alshawi, 1996; Eisner and Satta, 1999). Finally, we return to the issue of feature extraction (§5.5).

---

[5] We actually performed these sanity checks after implementation and experiments of our approach, because we missed this idea and because it requires access to test sets that we abstained from looking at during model development.

[6] Retrieved from `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2424`.

[7] The CoNLL 2017 shared task is an end-to-end parsing task, so the participating systems do not have access to gold-standard tokenization, which is a potential explanation for the presented analysis. On the other hand, the conclusion still holds even if we restrict to system outputs with perfect or nearly perfect segmentations.

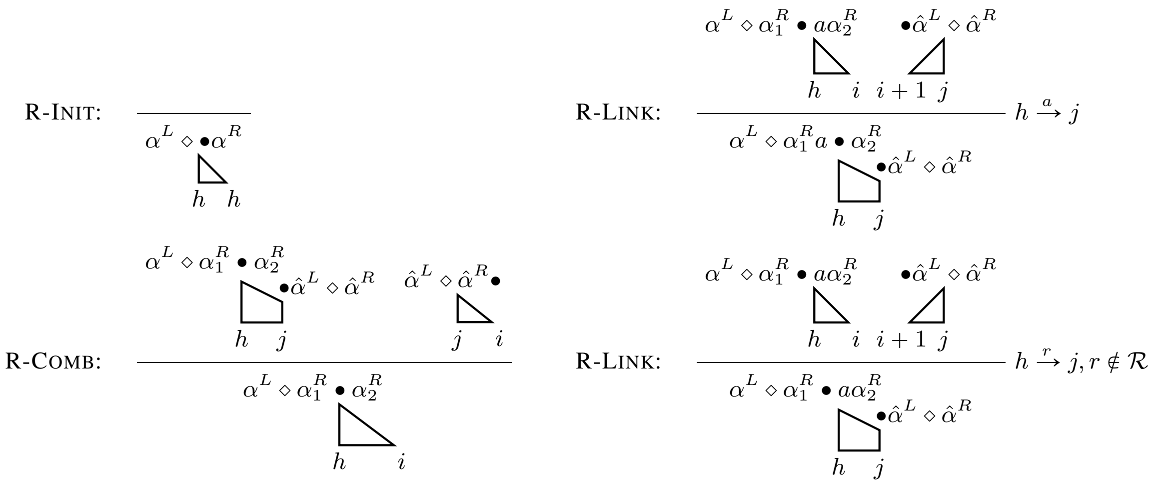

Figure 2: Eisner's (1996)/Eisner and Satta's (1999) algorithm, with valency-pattern annotations, incorporated as state information, shown explicitly. We show only the R-rules; the L-rules are symmetric.

5.2 Parameterization

We parameterize $P(v_i|w_i)$ as a softmax distribution over all candidate valency patterns:

$$P(v_i|w_i) \propto \exp(\text{score}_{v_i}^{\text{VAL}}(\mathbf{w}_i)),$$

where score$^{\text{VAL}}$ is a multi-layer perceptron (MLP).

For each word w_i, we generate a probability distribution over all potential syntactic heads in the sentence (Zhang et al., 2017). After we have selected the head of w_i to be w_{h_i}, we decide on the syntactic relation label based on another probability distribution. We use two softmax functions:

$$P(h_i|w_i) \propto \exp(\text{score}^{\text{HEAD}}(\mathbf{w}_{h_i}, \mathbf{w}_i)),$$
$$P(r_i|w_i, h_i) \propto \exp(\text{score}_{r_i}^{\text{LABEL}}(\mathbf{w}_{h_i}, \mathbf{w}_i)),$$

where both score$^{\text{HEAD}}$ and score$^{\text{LABEL}}$ are parameterized by deep biaffine scoring functions (Dozat and Manning, 2017).

5.3 Decoding

For joint decoding, we adopt the Eisner's (1996) algorithm annotated with valency patterns as the state information in Eisner and Satta (1999). The algorithm is depicted in Fig. 2. For each complete and incomplete span, visualized as triangles and trapezoids respectively, we annotate the head with its valency pattern. We adopt Earley's (1970) notation of $\bullet$ to outward-delimit the portion of a valency pattern, starting from the center word $\diamond$, that has already been collected within the span. INIT generates a minimal complete span with hypothesized valency pattern; the $\bullet$ is put adjacent to $\diamond$. COMB matches an incomplete span to a complete span with compatible valency pattern, yielding a complete analysis on the relevant side of $\diamond$. LINK either advances the $\bullet$ by attaching a syntactic dependent with the corresponding relation label, or attaches a dependent with a relation label irrelevant to the current valency analysis. This algorithm can be easily extended to cases where we analyze multiple subsets of valency relations simultaneously: we just need to annotate each head with multiple layers of valency patterns, one for each subset.[8]

The time complexity of a naïve dynamic programming implementation is $O(|V|^2|\alpha|n^3)$, where $|V|$ is the number of valency patterns and $|\alpha|$ is the maximum length of a valency pattern. In practice, $|V|$ is usually larger than n, making the algorithm prohibitively slow. We thus turn to A* parsing for a more practical solution.

A* parsing We take inspiration from A* CCG parsing (Lewis and Steedman, 2014; Lewis et al., 2016; Yoshikawa et al., 2017). The idea (see Alg. 1) is to estimate the best compatible full parse for every chart item (in our case, complete and incomplete spans), and expand the chart based on the estimated priority scores. Our factorization of probability scores allows the following admissible heuristic: for each span, we can optimistically estimate its best full parse score by assigning to

---

[8] To allow our model to account for unseen patterns in new data, we create a special wildcard valency pattern that allows dependents with arbitrary relations in the decoding process, and during training, treat valency patterns occurring fewer than 5 times as examples of the wildcard pattern.

Algorithm 1 Agenda-based best-first parsing algorithm, adapted from Lewis et al. (2016), Alg. 1.

Helper Functions: $\textsc{Init}(s)$ returns the set of spans generated by $\textsc{Init}$. $C.\textsc{Rules}(p)$ returns the set of spans that can be derived by combining p with existing entries in C through $\textsc{Comb}$ or $\textsc{Link}$.

```
procedure PARSE($s$)
    // Empty priority queue A
    $A \leftarrow \varnothing$
    // Initialize A with minimal complete spans
    for $p \in \textsc{Init}(s)$ do
        $A.\textsc{Insert}(p)$;
    // Empty chart C
    $C \leftarrow \varnothing$
    while $A \neq \varnothing$ do
        $p \leftarrow A.\textsc{PopMax}()$
        // Found the global optimal solution
        if $p$ is a full parse then return $p$
        else if $p \notin C$ then
            $C.\textsc{Add}(p)$
            // Extend the chart
            for $p' \in C.\textsc{Rules}(p)$ do
                $A.\textsc{Insert}(p')$
```

every token outside the span the best possible valency pattern, best possible attachment and best relation label.

5.4 Training

We train all components jointly and optimize for the cross entropy between our model prediction and the gold standard, or, equivalently, the sum of the log-probabilities for the three distributions comprising our factorization from §5.1. This can be thought of as an instance of multi-task learning (MTL; Caruana, 1997), which has been shown to be useful in parsing (Kasai et al., 2018). To further reduce error propagation, instead of using part-of-speech tags as features, we train a tagger jointly with our main parser components (Zhang and Weiss, 2016).

5.5 Feature Extraction

We adopt bi-directional long short-term memory networks (bi-LSTMs; Hochreiter and Schmidhuber, 1997) as our feature extractors, since they have proven successful in a variety of syntactic parsing tasks (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016; Stern et al., 2017; Shi et al., 2017a). As inputs to the bi-LSTMs, we concatenate one pre-trained word embedding, one randomly-initialized word embedding, and the output of character-level LSTMs for capturing sub-token level information (Ballesteros et al., 2015). The bi-LSTM output vectors at each timestep are then assigned to each token as its contextualized representation $\mathbf{w}_i$.

6 Experiments

Data and Evaluation Our main experiments are based on UD version 2.0, which was prepared for the CoNLL 2017 shared task (Zeman et al., 2017). We used 53 of the treebanks[9] across 41 languages that have train and development splits given for the shared task. In contrast to the shared-task setting, where word and sentence segmentation are to be performed by the system, we directly use the test-set gold segmentations in order to focus directly on parsing; but this does mean that the performance of our models cannot be directly compared to the officially-reported shared-task results. For evaluation, we report unlabeled and labeled attachment scores (UAS and LAS respectively). Further, we explicitly evaluate precision, recall and F1 scores (P/R/F) for the syntactic relations from Table 1, as well as valency pattern accuracies (VPA) involving those relations.

Implementation Details We use three-layer bi-LSTMs with 500 hidden units (250 in each direction) for feature extraction. The valency analyzer uses a one-hidden-layer MLP with ReLU activation function (Nair and Hinton, 2010), while the head selector and labeler use 512- and 128-dimensional biaffine scoring functions respectively. Our models are randomly initialized (Glorot and Bengio, 2010) and optimized with AMSgrad (Reddi et al., 2018) with initial learning rate 0.002. We apply dropout (Srivastava et al., 2014) to our MLPs and variational dropout (Gal and Ghahramani, 2016) to our LSTMs with a keep rate of 0.67 during training.

Efficiency Our A* parsers are generally reasonably efficient; for the rare ($< 1\%$) cases where the A* search does not finish within 500,000 chart expansion steps, we back off to a model without valency analysis. When analyzing three or more relation subsets, the initialization steps become pro-

---

[9] We exclude the two large treebanks `cs` and `ru_syntagrus` due to experiment resource constraints. There are other Czech and Russian treebanks in our selected collection.

			Core			Func.		
Subsets	UAS	LAS	#	VPA	P / R / F	#	VPA	P / R / F
Baseline	87.59	83.64	2.75	95.83	80.87 / 81.31 / 81.08	4.85	97.51	91.99 / 92.43 / 92.20
Core MTL	87.71	83.80	2.73	96.02	81.96 / **81.98** / 81.96	4.85	97.51	91.96 / 92.50 / 92.23
+ Joint Decoding	87.80	83.93	2.60	**96.68**	**85.43** / 81.75 / **83.53**	4.86	97.50	91.81 / 92.65 / 92.22
Func. MTL	87.67	83.71	2.75	95.80	80.69 / 81.21 / 80.94	4.84	97.58	92.30 / 92.57 / 92.44
+ Joint Decoding	87.72	83.75	2.75	95.80	80.64 / 81.32 / 80.96	4.80	**97.74**	**93.16** / 92.42 / 92.79
Core + Func. MTL	87.67	83.79	2.73	95.99	81.72 / 81.81 / 81.75	4.84	97.59	92.27 / 92.62 / 92.44
+ Joint Decoding	**87.81**	**83.99**	2.63	96.60	84.70 / 81.90 / 83.26	4.82	**97.74**	92.98 / **92.69** / **92.83**

Table 2: Macro-averaged results on UD 2.0 across 53 treebanks. Detailed results in the Suppl. Material. VPA=valency pattern accuracy; MTL=multi-task learning; #=average number of predicted attachments per sentence. Best results for each metrics are highlighted in bold.

Treebank	Baseline	Joint	ER	Treebank	Baseline	Joint	ER	Treebank	Baseline	Joint	ER
Dutch$_{\text{MAX}}$	84.91	89.83	32.63	Portuguese$_{\text{MAX}}$	92.24	93.46	15.66	Norwegian$_{\text{MIN}}$	90.38	91.41	10.76
Greek	85.82	89.50	25.95	Portuguese$_{\text{MIN}}$	86.90	88.88	15.10	Indonesian	81.53	83.51	10.75
Swedish$_{\text{MAX}}$	86.97	90.21	24.92	Spanish$_{\text{MAX}}$	85.04	87.29	15.06	Latvian	71.33	74.41	10.74
Finnish$_{\text{MAX}}$	88.14	90.87	22.96	Hungarian	79.11	82.13	14.45	French$_{\text{MIN}}$	90.56	91.51	10.01
Italian	87.04	90.00	22.85	Arabic	74.33	77.97	14.16	Basque	76.79	78.97	9.39
Latin$_{\text{MAX}}$	82.52	86.37	22.03	Urdu	70.47	74.63	14.07	Hindi	80.38	82.16	9.04
Danish	85.85	88.93	21.75	Dutch$_{\text{MIN}}$	76.03	79.18	13.13	German	81.05	82.73	8.85
Finnish$_{\text{MIN}}$	87.95	90.44	20.68	Swedish$_{\text{MIN}}$	85.51	87.36	12.76	Czech$_{\text{MIN}}$	76.68	78.64	8.42
Slovenian	86.03	88.59	18.31	Croatian	84.14	86.12	12.46	Polish	86.67	87.72	7.84
Old Slavonic	76.79	81.02	18.25	Gothic	72.15	75.60	12.38	A. Greek$_{\text{MIN}}$	59.00	62.17	7.74
French$_{\text{MAX}}$	89.86	91.71	18.21	A. Greek$_{\text{MAX}}$	74.31	77.48	12.34	Turkish	58.43	61.59	7.61
Estonian	72.02	77.09	18.09	Hebrew	80.27	82.60	11.80	Korean	83.33	84.55	7.31
Slovak	80.39	83.78	17.32	Persian	80.83	83.08	11.72	Chinese	73.81	75.66	7.07
Czech$_{\text{MAX}}$	85.58	88.02	16.90	Norwegian$_{\text{MAX}}$	91.20	92.21	11.50	English$_{\text{MIN}}$	84.69	85.60	5.96
Latin$_{\text{MIN}}$	76.60	80.55	16.89	Catalan	88.19	89.54	11.49	Vietnamese	48.45	51.49	5.91
Romanian	82.60	85.51	16.74	English$_{\text{MID}}$	84.26	86.05	11.37	Galician	72.17	73.70	5.49
English$_{\text{MAX}}$	90.96	92.43	16.20	Spanish$_{\text{MIN}}$	88.72	89.98	11.17	Japanese	91.87	91.88	0.14
Russian	82.17	85.05	16.13	Bulgarian	83.98	85.75	11.02	Average	81.08	83.53	13.80

Table 3: Treebank-specific F1 scores on core argument relations, comparing the baseline models to our Core MTL + joint decoding models, sorted by the error reduction (ER, %) rate. When comparing a model with performance s_2 against baseline score s_1, ER is defined as $(s_2 - s_1)/(1 - s_1)$. For languages with two or three treebanks, we include multiple entries differentiated by the subscripts MAX/MID/MIN, corresponding to the treebanks with the highest/median/lowest ER, respectively. A. Greek = Ancient Greek.

hibitively slow due to the large number of valency pattern combinations. Thus, we limit the number of combinations for each token to the highest-scoring 500.

Results on UD We present our main experimental results on UD in Table 2. The baseline system does not leverage any valency information (we only train the head selectors and labelers, and use the original Eisner decoder). We compare the baseline to settings where we train the parsers jointly with our proposed valency analyzers, distinguishing the effect of using this information only at training (multi-task learning; MTL) vs. both at training and decoding.

Including valency analysis into the training objective already provides a slight improvement in parsing performance, in line with the findings of Kasai et al. (2018). With our proposed joint decoding, there is a mild improvement to the overall UAS and LAS, and a higher boost to VPA. The output parse trees are now more precise in the analyzed valency relations: on core arguments, precision increases by as much as 4.56. As shown by Table 3, the performance gain of joint decoding varies across treebanks, ranging from an error reduction rate of over 30% (Dutch Lassy Small Treebank) on core argument relations to nearly 0% (Japanese). Overall, our approach exhibits a clearly positive impact on most of the treebanks in UD. We do not see performance correlating to language typology, although we do observe smaller error-reduction rates on treebanks with lower baseline performances, that is, on "harder" languages.

7 Parsing Tree Adjoining Grammar

Dependency and valency relations also play an important role in formalisms other than dependency grammar. In this section, we apply our proposed valency analysis to Tree Adjoining Grammar (TAG; Joshi and Schabes, 1997), because TAG derivation trees, representing the process of inserting obligatory arguments and adjoining modifiers, can be treated as a dependency representation (Rambow and Joshi, 1997). We follow prior art and use Chen's (2001) automatic conversion of the Penn Treebank (Marcus et al., 1993) into TAG derivation trees. The dataset annotation has labels 0, 1 and 2, corresponding to subject, direct object, and indirect object; we treat these as our core argument subset in valency analysis.[10] Additionally, we also analyze CO (co-head for phrasal verbs) as a separate singleton subset. We leave out adj (adjuncts) in defining our valency patterns. We strictly follow the experiment protocol of previous work (Bangalore et al., 2009; Chung et al., 2016; Friedman et al., 2017; Kasai et al., 2017, 2018), and report the results in Table 4. The findings are consistent with our main experiments: MTL helps parsing performance, and joint decoding further improves on core argument F1 scores, reaching a new state-of-the-art result of 92.59 LAS. The precision recall trade-off is pronounced for the CO relation subset.

8 Case Study on PP Attachment

Although valency information has traditionally been used to analyze complements or core arguments,[11] in this section, we show the utility of our approach in analyzing other types of syntactic relations. We choose the long-standing problem of prepositional phrase (PP) attachment (Hindle and Rooth, 1993; Brill and Resnik, 1994; Collins and Brooks, 1995; de Kok et al., 2017), which is known to be a major source of parsing mistakes (Kummerfeld et al., 2012; Ng and Curran, 2015). In UD analysis, PPs usually have the labels obl or nmod with respect to their syntactic parents, whereas adpositions are attached via a case relation, which is included in the functional relation subset. Thus, we add another relation subset, obl and nmod, to our valency analysis.

Table 5 presents the results for different combinations of valency relation subsets. We find that PP-attachment decisions are generally harder to make, compared with core and functional relations. Including them during training distracts other parsing objectives (compare Core + PP with only analyzing Core in §6). However, they do permit improvements on precision for PP attachment by 3.30, especially with our proposed joint decoding. This demonstrates the usage of our algorithm outside the traditional notions of valency — it can be a general method for training parsers to focus on specific subsets of syntactic relations.

---

[10] We choose not to use the sparse labels 3 and 4, which encode additional complements.

[11] There are also recent proposals to analyze valency without distinguishing complements and adjuncts (Čech et al., 2010).

			Core		CO	
	UAS	LAS	VPA	P / R / F	VPA	P / R / F
Friedman et al. (2017)	90.31	88.96	–	–	–	–
Kasai et al. (2017)	90.97	89.68	–	–	–	–
Kasai et al. (2018)	93.26	91.89	–	–	–	–
Baseline	93.66	92.44	97.06	92.45 / 92.76 / 92.60	99.22	73.11 / **87.20** / 79.54
Core + CO MTL	93.71	92.53	97.19	92.74 / 93.20 / 92.97	**99.24**	75.43 / 84.44 / 79.68
+ Joint Decoding	**93.75**	**92.59**	**97.47**	**93.27 / 93.22 / 93.24**	**99.24**	**76.06** / 83.70 / **79.70**

Table 4: Experimental results on parsing TAGs.

	UAS	LAS	Core P / R / F	Func. P / R / F	PP P / R / F
Baseline	87.59	83.64	80.87 / 81.31 / 81.08	91.99 / 92.43 / 92.20	77.29 / 77.99 / 77.62
PP MTL	87.67	83.70	80.61 / 81.23 / 80.91	92.03 / 92.50 / 92.26	78.30 / **78.38** / 78.32
+ Joint Decoding	87.68	83.69	79.93 / 81.50 / 80.69	91.92 / 92.51 / 92.21	**80.59** / 77.68 / **79.04**
Core + PP MTL	87.70	83.77	81.62 / 81.81 / 81.71	91.93 / 92.52 / 92.22	77.93 / 78.25 / 78.08
+ Joint Decoding	87.80	83.91	**84.18 / 81.97 / 83.05**	91.68 / **92.65** / 92.16	79.71 / 78.03 / 78.83
Core + Func. + PP MTL	87.67	83.75	81.35 / 81.68 / 81.50	92.18 / 92.61 / 92.39	77.99 / 78.22 / 78.08
+ Joint Decoding	**87.81**	**83.94**	83.88 / **81.97** / 82.90	**92.78** / 92.63 / **92.70**	79.54 / 78.11 / 78.78

Table 5: Experimental results involving analyzing PPs as valency patterns.

9 Further Related Work

Supertagging Supertagging (Bangalore and Joshi, 2010) has been proposed for and used in parsing TAG (Bangalore and Joshi, 1999; Nasr and Rambow, 2004), CCG (Curran and Clark, 2003; Curran et al., 2006), and HPSG (Ninomiya et al., 2006; Blunsom and Baldwin, 2006). Within dependency parsing, supertags have also been explored in the literature, but prior work mostly treats them as additional features. Ambati et al. (2013, 2014) use CCG supertags to improve dependency parsing results, while Ouchi et al. (2014, 2016) leverage dependency-based supertags as features. Faleńska et al. (2015) compare supertagging to parser stacking, where they extract supertags from base parsers to provide additional features for stacked parsers, instead of having a supertagger as a separate component.

Constrained Dependency Grammar Another line of research (Wang and Harper, 2004; Foth et al., 2006; Foth and Menzel, 2006; Bharati et al., 2002, 2009; Husain et al., 2011) utilizes supertags in dependency parsing within the framework of constraint dependency grammar (CDG; Maruyama, 1990; Heinecke et al., 1998). Constraints in CDG may be expressed in very general terms (and are usually hand-crafted for specific languages), so prior work in CDG involves a constraint solver that iteratively or greedily update hypotheses without optimality guarantees. In contrast, our work focuses on a special form of constraints — the valency patterns of syntactic dependents within a subset of relations — and we provide an efficient A*-based exact decoding algorithm.

Valency in Parsing To the best of our knowledge, there have been few attempts to utilize lexical valency information or to improve specifically on core arguments in syntactic parsing apart from CDG. Øvrelid and Nivre (2007) target parsing core relations in Swedish with specifically-designed features such as animacy and definiteness that are useful in argument realization. Jakubıček and Kovář (2013) leverage external lexicons of verb valency frames for reranking. Mirroshandel et al. (2012, 2013) and Mirroshandel and Nasr (2016) extract selectional constraints and subcategorization frames from large unannotated corpora, and enforce them through forest reranking. Our approach does not rely on external resources or lexicons, but directly extracts valency patterns from labeled dependency parse trees. Earlier works in this spirit include Collins (1997).

Semantic Dependency Parsing and Semantic Role Labeling The notion of valency is also used to describe predicate-argument structures that are adopted in semantic dependency parsing and semantic role labeling (Surdeanu et al., 2008; Hajič et al., 2009; Oepen et al., 2014, 2015). While semantic frames clearly have patterns, previous work (Punyakanok et al., 2008; Flanigan et al., 2014; Täckström et al., 2015; Peng et al., 2017; He et al., 2017) incorporates several types of constraints, including uniqueness and determinism constraints that require that certain labels appear as arguments for a particular predicate only once. They perform inference through integer linear programming, which is usually solved approximately, and cannot easily encode linear ordering constraints for the arguments.

A* parsing Best-first search uses a heuristic to expand the parsing chart instead of doing so exhaustively. It was first applied to PCFGs (Ratnaparkhi, 1997; Caraballo and Charniak, 1998; Sagae and Lavie, 2006), and then to dependency parsing (Sagae and Tsujii, 2007; Zhao et al., 2013; Vaswani and Sagae, 2016). Our probability factorization permits a simple yet effective A* heuristic. A* parsing was introduced for parsing PCFGs (Klein and Manning, 2003; Pauls and Klein, 2009), and has been widely used for grammar formalisms and parsers with large search spaces, for example CCG (Auli and Lopez, 2011) and TAG (Waszczuk et al., 2016, 2017). Our decoder is similar to the supertag and dependency factored A* CCG parser (Yoshikawa et al., 2017), which in turn builds upon the work of Lewis and Steedman (2014) and Lewis et al. (2016). Our model additionally adds syntactic relations into the probability factorizations.

10 Conclusions

We have presented a probability factorization and decoding process that integrates valency patterns into the parsing process. The joint decoder favors syntactic analyses with higher valency-pattern supertagging probabilities. Experiments on a large set of languages from UD show that our parsers are more precise in the subset of syntactic relations chosen for valency analysis, in addition to enjoying the benefits gained from jointly training the parsers and supertaggers in a multi-task learning setting.

Our method is not limited to a particular type of treebank annotation or a fixed subset of relations. We draw similar conclusions when we parse TAG derivation trees. Most interestingly, in a case study on PP attachment, we confirm the utility of our parsers in handling syntactic relations beyond the traditional domain of valency.

A key insight of this paper that departs from prior work on automatic extraction of supertags from dependency annotations is that our definition of valency patterns is relativized to a subset of syntactic relations. This definition is closer to the linguistic notion of valency and alleviates the data sparsity problems in that the number of extracted valency patterns is small. At the same time, the patterns generalize well, and empirically, they are effective in our proposed joint decoding process.

Our findings point to a number of directions for future work. First, the choice of subsets of syntactic relations for valency analysis impacts the parsing performance in those categories. This may suggest a controllable way to address precision-recall trade-offs targeting specific relation types. Second, we experimented with a few obvious subsets of relations; characterizing what subsets can be most improved with valency augmentation is an open question. Finally, our decoder builds upon projective dependency-tree decoding algorithms. In the future, we will explore the possibility of removing the projective constraint and the tree requirement, extending the applicability of valency patterns to other tasks such as semantic role labeling.

Acknowledgments

We thank the three anonymous reviewers for their insightful comments, Jungo Kasai for assistance in setting up the TAG parsing experiments, and Xilun Chen, Jason Eisner, Jungo Kasai and Ana Smith for discussion and comments. We also thank CoNLL'17 shared task organizers and participants for publicizing system outputs. TS and LL were supported in part by a Google Focused Research Grant to Cornell University. LL was also supported in part by NSF grant SES-1741441. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other sponsors.

References

Vilmos Ágel, Ludwig M. Eichinger, Hans Werner Eroms, Peter Hellwig, Hans Jürgen Heringer, and Henning Lobin, editors. 2003. *Dependency and valency: An international handbook of contemporary research*, volume 1. De Gruyter Mouton.

Vilmos Ágel, Ludwig M. Eichinger, Hans Werner Eroms, Peter Hellwig, Hans Jürgen Heringer, and Henning Lobin, editors. 2006. *Dependency and valency: An international handbook of contemporary research*, volume 2. De Gruyter Mouton.

Vilmos Ágel and Klaus Fischer. 2015. Dependency grammar and valency theory. In Bernd Heine and

Heiko Narrog, editors, *The Oxford Handbook of Linguistic Analysis*, 2nd edition. Oxford University Press, Oxford.

Hiyan Alshawi. 1996. Head automata and bilingual tiling: Translation with minimal representations (invited talk). In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 167–176, Santa Cruz, California, USA. Association for Computational Linguistics.

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2013. Using CCG categories to improve Hindi dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 604–609, Sofia, Bulgaria. Association for Computational Linguistics.

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2014. Improving dependency parsers using Combinatory Categorial Grammar. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 159–163, Gothenburg, Sweden. Association for Computational Linguistics.

Michael Auli and Adam Lopez. 2011. Efficient CCG parsing: A* versus adaptive supertagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1577–1585, Portland, Oregon, USA. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal. Association for Computational Linguistics.

Srinivas Bangalore, Pierre Boullier, Alexis Nasr, Owen Rambow, and Benoît Sagot. 2009. MICA: A probabilistic dependency parser based on tree insertion grammars (application note). In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 185–188, Boulder, Colorado. Association for Computational Linguistics.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics.*, 25(2):237–265.

Srinivas Bangalore and Aravind K. Joshi. 2010. *Supertagging: Using Complex Lexical Descriptions in Natural Language Processing*. The MIT Press.

Akshar Bharati, Samar Husain, Dipti Misra, and Rajeev Sangal. 2009. Two stage constraint based hybrid approach to free word order language dependency parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 77–80. Association for Computational Linguistics.

Akshar Bharati, Rajeev Sangal, and T. Papi Reddy. 2002. A constraint based parser using integer programming. *Proceedings of the International Conference on Natural Language Processing*.

Anders Björkelund, Agnieszka Falenska, Xiang Yu, and Jonas Kuhn. 2017. IMS at the CoNLL 2017 UD shared task: CRFs and perceptrons meet neural networks. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 40–51, Vancouver, Canada. Association for Computational Linguistics.

Phil Blunsom and Timothy Baldwin. 2006. Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 164–171, Sydney, Australia. Association for Computational Linguistics.

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th Conference on Computational Linguistics – Volume 2*, pages 1198–1204, Kyoto, Japan. Association for Computational Linguistics.

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Radek Čech, Petr Pajas, and Ján Mačutek. 2010. Full valency. Verb valency without distinguishing complements and adjuncts. *Journal of Quantitative Linguistics*, 17(4):291–302.

Wanxiang Che, Jiang Guo, Yuxuan Wang, Bo Zheng, Huaipeng Zhao, Yang Liu, Dechuan Teng, and Ting Liu. 2017. The HIT-SCIR system for end-to-end parsing of Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 52–62, Vancouver, Canada. Association for Computational Linguistics.

John Chen. 2001. *Towards efficient statistical parsing using lexicalized grammatical information*. Ph.D. thesis, University of Delaware.

Wonchang Chung, Suhas Siddhesh Mhatre, Alexis Nasr, Owen Rambow, and Srinivas Bangalore. 2016. Revisiting supertagging and parsing: How to use supertags in transition-based parsing. In *Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 85–92, Düsseldorf, Germany.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

Michael Collins and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proceedings of the 3rd Workshop on Very Large Corpora*, Cambridge, MA.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37.

James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics – Volume 1*, pages 91–98, Budapest, Hungary. Association for Computational Linguistics.

James R. Curran, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 697–704, Sydney, Australia. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communations of the ACM*, 13(2):94–102.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 457–464, College Park, Maryland, USA. Association for Computational Linguistics.

Agnieszka Faleńska, Anders Björkelund, Özlem Çetinoğlu, and Wolfgang Seeker. 2015. Stacking or supertagging for dependency parsing – what's the difference? In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 118–129, Bilbao, Spain. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.

Kilian A. Foth, Tomas By, and Wolfgang Menzel. 2006. Guiding a constraint dependency parser with supertags. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 289–296, Sydney, Australia. Association for Computational Linguistics.

Kilian A. Foth and Wolfgang Menzel. 2006. Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 321–328, Sydney, Australia. Association for Computational Linguistics.

Dan Friedman, Jungo Kasai, R. Thomas McCoy, Robert Frank, Forrest Davis, and Owen Rambow. 2017. Linguistically rich vector representations of supertags for TAG parsing. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 122–131, Umeå, Sweden. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado. Association for Computational Linguistics.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Johannes Heinecke, Jürgen Kunze, Wolfgang Menzel, and Ingo Schröder. 1998. Eliminative parsing with graded constraints. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics – Volume 1*, pages 526–530, Montreal, Quebec, Canada. Association for Computational Linguistics.

Gerhard Helbig and Wolfgang Schenkel. 1959. *Wörterbuch zur Valenz und Distribution deutscher Verben*. Bibliographisches Institut, Leipzig.

Thomas Herbst, David Heath, Ian F. Roe, and Dieter Götz. 2004. *A Valency Dictionary of English: A Corpus-Based Analysis of the Complementation Patterns of English Verbs, Nouns and Adjectives*, volume 40 of *Topics in English Linguistics*. De Gruyter Mouton, Berlin, Boston.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

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

Samar Husain, Raghu Pujitha Gade, and Rajeev Sangal. 2011. Linguistically rich graph based data driven parsing for Hindi. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 56–61, Dublin, Ireland. Association for Computational Linguistics.

Miloš Jakubiček and Vojtěch Kovář. 2013. Enhancing Czech parsing with verb valency frames. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 282–293, Samos, Greece. Springer.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages, Volume 3: Beyond Words*, pages 69–124. Springer, New York.

Jungo Kasai, Robert Frank, R. Thomas McCoy, Owen Rambow, and Alexis Nasr. 2017. TAG parsing with neural networks and vector representations of supertags. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722, Copenhagen, Denmark. Association for Computational Linguistics.

Jungo Kasai, Robert Frank, Pauli Xu, William Merrill, and Owen Rambow. 2018. End-to-end graph-based TAG parsing with neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1181–1194. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology – Volume 1*, pages 40–47, Edmonton, Canada. Association for Computational Linguistics.

Daniël de Kok, Jianqiang Ma, Corina Dima, and Erhard Hinrichs. 2017. PP attachment: Where do we stand? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 311–317, Valencia, Spain. Association for Computational Linguistics.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street Corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, San Diego, California. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 990–1000, Doha, Qatar. Association for Computational Linguistics.

KyungTae Lim and Thierry Poibeau. 2017. A system for multilingual dependency parsing based on bidirectional LSTM feature representations. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 63–70, Vancouver, Canada. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics*, pages 31–38, Pittsburgh, Pennsylvania. Association for Computational Linguistics.

Seyed Abolghasem Mirroshandel and Alexis Nasr. 2016. Integrating selectional constraints and subcategorization frames in a dependency parser. *Computational Linguistics*, 42(1):55–90.

Seyed Abolghasem Mirroshandel, Alexis Nasr, and Joseph Le Roux. 2012. Semi-supervised dependency parsing using lexical affinities. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 777–785. Association for Computational Linguistics.

Seyed Abolghasem Mirroshandel, Alexis Nasr, and Benoît Sagot. 2013. Enforcing subcategorization constraints in a parser using sub-parses recombining. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 239–247. Association for Computational Linguistics.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814, Haifa, Israel.

Alexis Nasr and Owen Rambow. 2004. Supertagging and full parsing. In *Proceedings of the 7th International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 56–63, Vancouver, Canada.

Dominick Ng and James R. Curran. 2015. Identifying cascading errors using constraints in dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1148–1158, Beijing, China. Association for Computational Linguistics.

Takashi Ninomiya, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Extremely lexicalized models for accurate and fast HPSG parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 155–163, Sydney, Australia. Association for Computational Linguistics.

Joakim Nivre. 2005. Dependency grammar and dependency parsing. Technical Report MSI 05133, Växjö University, School of Mathematics and Systems Engineering.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà Mỹ, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phương Lê Hồng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre and Chiao-Ting Fang. 2017. Universal dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies*, pages 86–95, Gothenburg, Sweden. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Work-*

shop on Semantic Evaluation (SemEval 2015), pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Hiroki Ouchi, Kevin Duh, and Yuji Matsumoto. 2014. Improving dependency parsers with supertags. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 154–158, Gothenburg, Sweden. Association for Computational Linguistics.

Hiroki Ouchi, Kevin Duh, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Transition-based dependency parsing exploiting supertags. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2059–2068.

Lilja Øvrelid and Joakim Nivre. 2007. When word order and part-of-speech tags are not enough – Swedish dependency parsing with rich linguistic features. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 447–451, Borovets, Bulgaria.

Adam Pauls and Dan Klein. 2009. K-best A* parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 958–966, Suntec, Singapore. Association for Computational Linguistics.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2048, Vancouver, Canada. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Owen Rambow and Aravind Joshi. 1997. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. volume 39, pages 167–190. John Benjamins, Amsterdam and Philadelphia.

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Providence, Rhode Island, USA.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of Adam and beyond. In *Proceedings of the 6th International Conference on Learning Representations*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101, Copenhagen, Denmark. Association for Computational Linguistics.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL Main Conference Poster Sessions*, pages 691–698, Sydney, Australia. Association for Computational Linguistics.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050, Prague, Czech Republic. Association for Computational Linguistics.

Tianze Shi, Liang Huang, and Lillian Lee. 2017a. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 12–23, Copenhagen, Denmark. Association for Computational Linguistics.

Tianze Shi, Felix G. Wu, Xilun Chen, and Yao Cheng. 2017b. Combining global models for parsing Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 31–39, Vancouver, Canada. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008:*

Proceedings of the Twelfth Conference on Computational Natural Language Learning, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Lucien Tesnière. 1959. *Éléments de Syntaxe Structurale*. Librairie C. Klincksieck, Paris.

Damon Tutunjian and Julie E. Boland. 2008. Do we need a distinction between arguments and adjuncts? Evidence from psycholinguistic studies of comprehension. *Language and Linguistics Compass*, 2(4):631–646.

Ashish Vaswani and Kenji Sagae. 2016. Efficient structured inference for transition-based parsing with neural networks and error states. *Transactions of the Association for Computational Linguistics*, 4:183–196.

Wen Wang and Mary P. Harper. 2004. A statistical constraint dependency grammar (CDG) parser. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 42–49, Barcelona, Spain. Association for Computational Linguistics.

Jakub Waszczuk, Agata Savary, and Yannick Parmentier. 2016. Promoting multiword expressions in A* TAG parsing. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 429–439, Osaka, Japan. The COLING 2016 Organizing Committee.

Jakub Waszczuk, Agata Savary, and Yannick Parmentier. 2017. Multiword expression-aware A* TAG parsing revisited. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 84–93, Umeå, Sweden. Association for Computational Linguistics.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 277–287, Vancouver, Canada. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566, Berlin, Germany. Association for Computational Linguistics.

Kai Zhao, James Cross, and Liang Huang. 2013. Optimal incremental parsing via best-first dynamic programming. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 758–768, Seattle, Washington, USA. Association for Computational Linguistics.

Unsupervised Learning of Syntactic Structure with Invertible Neural Projections

Junxian He **Graham Neubig** **Taylor Berg-Kirkpatrick**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{junxianh, gneubig, tberg}@cs.cmu.edu

Abstract

Unsupervised learning of syntactic structure is typically performed using generative models with discrete latent variables and multinomial parameters. In most cases, these models have not leveraged continuous word representations. In this work, we propose a novel generative model that jointly learns discrete syntactic structure and continuous word representations in an unsupervised fashion by cascading an *invertible* neural network with a structured generative prior. We show that the invertibility condition allows for efficient exact inference and marginal likelihood computation in our model so long as the prior is well-behaved. In experiments we instantiate our approach with both Markov and tree-structured priors, evaluating on two tasks: part-of-speech (POS) induction, and unsupervised dependency parsing without gold POS annotation. On the Penn Treebank, our Markov-structured model surpasses state-of-the-art results on POS induction. Similarly, we find that our tree-structured model achieves state-of-the-art performance on unsupervised dependency parsing for the difficult training condition where neither gold POS annotation nor punctuation-based constraints are available.[1]

1 Introduction

Data annotation is a major bottleneck for the application of supervised learning approaches to many problems. As a result, unsupervised methods that learn directly from unlabeled data are increasingly important. For tasks related to unsupervised syntactic analysis, discrete generative models have dominated in recent years – for example, for both part-of-speech (POS) induction (Blunsom and Cohn, 2011; Stratos et al., 2016) and unsupervised dependency parsing (Klein and Manning, 2004; Cohen and Smith, 2009; Pate and Johnson, 2016). While similar models have had success on a range of unsupervised tasks, they have mostly ignored the apparent utility of continuous word representations evident from supervised NLP applications (He et al., 2017; Peters et al., 2018). In this work, we focus on leveraging and explicitly representing continuous word embeddings within unsupervised models of syntactic structure.

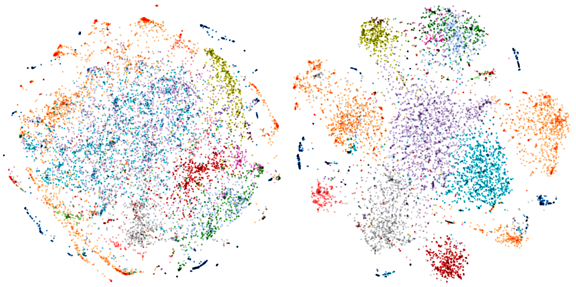

(a) Traditional pre-trained skip-gram embeddings (b) Learned latent embeddings from our approach

Figure 1: Visualization (t-SNE) of skip-gram embeddings (trained on one billion words with context window size equal to 1) and latent embeddings learned by our approach with a Markov-structured prior. Each node represents a word and is colored according to the most likely gold POS tag from the Penn Treebank (best seen in color).

Pre-trained word embeddings from massive unlabeled corpora offer a compact way of injecting a prior notion of word similarity into models that would otherwise treat words as discrete, isolated categories. However, the specific properties of language captured by any particular embedding scheme can be difficult to control, and, further, may not be ideally suited to the task at hand. For example, pre-trained skip-gram embeddings (Mikolov et al., 2013) with small context window size are found to capture the syntactic properties of language well (Bansal et al., 2014; Lin et al., 2015). However, if our goal is to separate syntactic categories, this embedding space is not ideal – POS categories correspond to overlap-

---

[1] Code is available at https://github.com/jxhe/struct-learning-with-flow.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1292–1302
Brussels, Belgium, October 31 - November 4, 2018.

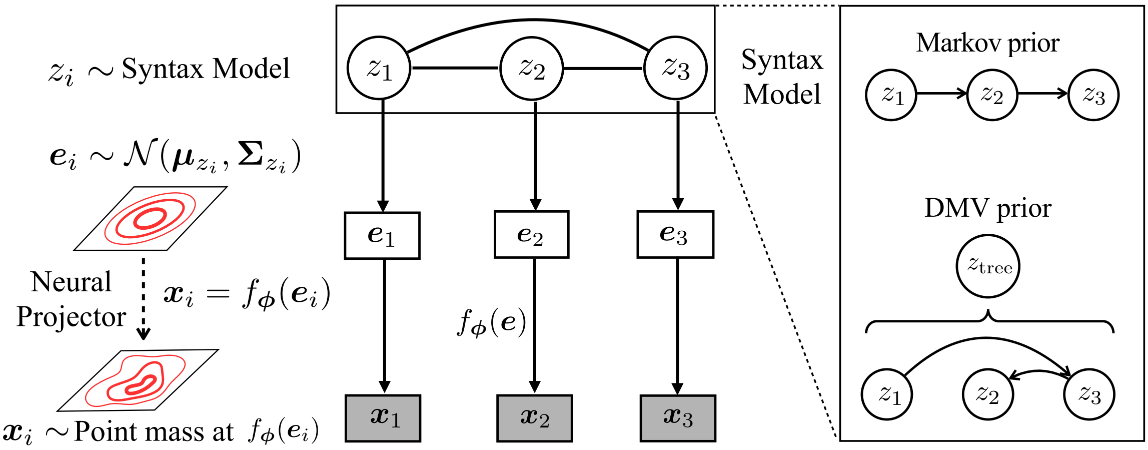

Figure 2: Depiction of proposed generative model. The syntax model is composed of discrete random variables, z_i. Each $\boldsymbol{e}_i$ is a latent continuous embeddings sampled from Gaussian distribution conditioned on z_i, while $\boldsymbol{x}_i$ is the observed embedding, deterministically derived from $\boldsymbol{e}_i$. The left portion depicts how the neural projector maps the simple Gaussian to a more complex distribution in the output space. The right portion depicts two instantiations of the syntax model in our approach: one is Markov-structured and the other is DMV-structured. For DMV, z_{tree} is the latent dependency tree structure.

ping interspersed regions in the embedding space, evident in Figure 1(a).

In our approach, we propose to learn a new latent embedding space as a projection of pre-trained embeddings (depicted in Figure 1(b)), while *jointly* learning latent syntactic structure – for example, POS categories or syntactic dependencies. To this end, we introduce a new generative model (shown in Figure 2) that first generates a latent syntactic representation (e.g. a dependency parse) from a discrete structured prior (which we also call the "syntax model"), then, conditioned on this representation, generates a sequence of latent embedding random variables corresponding to each word, and finally produces the observed (pre-trained) word embeddings by projecting these latent vectors through a parameterized non-linear function. The latent embeddings can be jointly learned with the structured syntax model in a completely unsupervised fashion.

By choosing an invertible neural network as our non-linear projector, and then parameterizing our model in terms of the projection's inverse, we are able to derive tractable exact inference and marginal likelihood computation procedures so long as inference is tractable in the underlying syntax model. In §3.1 we show that this derivation corresponds to an alternate view of our approach whereby we jointly learn a mapping of observed word embeddings to a new embedding space that is more suitable for the syntax model, but include an additional Jacobian regularization term to prevent information loss.

Recent work has sought to take advantage of word embeddings in unsupervised generative models with alternate approaches (Lin et al., 2015; Tran et al., 2016; Jiang et al., 2016; Han et al., 2017). Lin et al. (2015) build an HMM with Gaussian emissions on observed word embeddings, but they do not attempt to learn new embeddings. Tran et al. (2016), Jiang et al. (2016), and Han et al. (2017) extend HMM or dependency model with valence (DMV) (Klein and Manning, 2004) with multinomials that use word (or tag) embeddings in their parameterization. However, they do not represent the embeddings as latent variables.

In experiments, we instantiate our approach using both a Markov-structured syntax model and a tree-structured syntax model – specifically, the DMV. We evaluate on two tasks: part-of-speech (POS) induction and unsupervised dependency parsing without gold POS tags. Experimental results on the Penn Treebank (Marcus et al., 1993) demonstrate that our approach improves the basic HMM and DMV by a large margin, leading to the state-of-the-art results on POS induction, and state-of-the-art results on unsupervised dependency parsing in the difficult training scenario where neither gold POS annotation nor punctuation-based constraints are available.

2 Model

As an illustrative example, we first present a baseline model for Markov syntactic structure (POS induction) that treats a sequence of pre-trained word embeddings as observations. Then, we propose our novel approach, again using Markov structure, that introduces latent word embedding variables and a neural projector. Lastly, we extend our approach to more general syntactic structures.

2.1 Example: Gaussian HMM

We start by describing the Gaussian hidden Markov model introduced by Lin et al. (2015), which is a locally normalized model with multinomial transitions and Gaussian emissions. Given a sentence of length ℓ, we denote the latent POS tags as $\boldsymbol{z} = \{z_i\}_{i=1}^{\ell}$, observed (pre-trained) word embeddings as $\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^{\ell}$, transition parameters as $\boldsymbol{\theta}$, and Gaussian emission parameters as $\boldsymbol{\eta}$. The joint distribution of data and latent variables factors as:

$$p(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\eta}) = \prod\nolimits_{i=1}^{\ell} p_{\boldsymbol{\theta}}(z_i|z_{i-1}) p_{\boldsymbol{\eta}}(\boldsymbol{x}_i|z_i), \quad (1)$$

where $p_{\boldsymbol{\theta}}(z_i|z_{i-1})$ is the multinomial transition probability and $p_{\boldsymbol{\eta}}(\boldsymbol{x}_i|z_i)$ is the multivariate Gaussian emission probability.

While the observed word embeddings do inform this model with a notion of word similarity – lacking in the basic multinomial HMM – the Gaussian emissions may not be sufficiently flexible to separate some syntactic categories in the complex pre-trained embedding space – for example the skip-gram embedding space as visualized in Figure 1(a) where different POS categories overlap. Next we introduce a new approach that adds flexibility to the emission distribution by incorporating new latent embedding variables.

2.2 Markov Structure with Neural Projector

To flexibly model observed embeddings and yield a new representation space that is more suitable for the syntax model, we propose to cascade a neural network as a projection function, deterministically transforming the simple space defined by the Gaussian HMM to the observed embedding space. We denote the latent embedding of the i^{th} word in a sentence as $\boldsymbol{e}_i \in \mathbb{R}^{d_e}$, and the neural projection function as f, parameterized by $\boldsymbol{\phi}$. In the case of sequential Markov structure, our new model corresponds to the following generative process:

For each time step $i = 1, 2, \cdots, \ell$,

- Draw the latent state $z_i \sim p_{\boldsymbol{\theta}}(z_i|z_{i-1})$
- Draw the latent embedding $\boldsymbol{e}_i \sim \mathcal{N}(\boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i})$
- Deterministically produce embedding $\boldsymbol{x}_i = f_{\boldsymbol{\phi}}(\boldsymbol{e}_i)$

The graphical model is depicted in Figure 2. The deterministic projection can also be viewed as sampling each observation from a point mass at $f_{\boldsymbol{\phi}}(\boldsymbol{e}_i)$. The joint distribution of our model is:

$$\begin{aligned} &p(\boldsymbol{z}, \boldsymbol{e}, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\phi}) \\ &\quad = \prod\nolimits_{i=1}^{\ell} [p_{\boldsymbol{\theta}}(z_i|z_{i-1}) p_{\boldsymbol{\eta}}(\boldsymbol{e}_i|z_i) p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{e}_i)], \end{aligned} \quad (2)$$

where $p_{\boldsymbol{\eta}}(\cdot|z_i)$ is a conditional Gaussian distribution, and $p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{e}_i)$ is the Dirac delta function centered at $f_{\boldsymbol{\phi}}(\boldsymbol{e}_i)$:

$$p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{e}_i) = \delta(\boldsymbol{x}_i - f_{\boldsymbol{\phi}}(\boldsymbol{e}_i)) = \begin{cases} \infty & \boldsymbol{x}_i = f_{\boldsymbol{\phi}}(\boldsymbol{e}_i) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

2.3 General Structure with Neural Projector

Our approach can be applied to a broad family of structured syntax models. We denote latent embedding variables as $\boldsymbol{e} = \{\boldsymbol{e}_i\}_{i=1}^{\ell}$, discrete latent variables in the syntax model as $\boldsymbol{z} = \{z_k\}_{k=1}^{K}$ ($K \geqslant \ell$), where $z_1, z_2, \ldots, z_\ell$ are conditioned to generate $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_\ell$. The joint probability of our model factors as:

$$\begin{aligned} p(\boldsymbol{z}, \boldsymbol{e}, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\phi}) = &\prod\nolimits_{i=1}^{\ell} \left[p_{\boldsymbol{\eta}}(\boldsymbol{e}_i|z_i) p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{e}_i)\right] \\ &\cdot p_{\text{syntax}}(\boldsymbol{z}; \boldsymbol{\theta}), \end{aligned} \quad (4)$$

where $p_{\text{syntax}}(\boldsymbol{z}; \boldsymbol{\theta})$ represents the probability of the syntax model, and can encode any syntactic structure – though, its factorization structure will determine whether inference is tractable in our full model. As shown in Figure 2, we focus on two syntax models for syntactic analysis in this paper. The first is Markov-structured, which we use for POS induction, and the second is DMV-structured, which we use to learn dependency parses without supervision.

The marginal data likelihood of our model is:

$$\begin{aligned} p(\boldsymbol{x}) = &\sum\nolimits_{\boldsymbol{z}} \Big(p_{\text{syntax}}(\boldsymbol{z}; \boldsymbol{\theta}) \\ &\cdot \prod\nolimits_{i=1}^{\ell} \underbrace{\Big[\int_{\boldsymbol{e}_i} p_{\boldsymbol{\eta}}(\boldsymbol{e}_i|z_i) p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{e}_i) \mathrm{d}\boldsymbol{e}_i\Big]}_{p(\boldsymbol{x}_i|z_i)}\Big). \end{aligned} \quad (5)$$

While the discrete variables $\boldsymbol{z}$ can be marginalized out with dynamic program in many cases, it is generally intractable to marginalize out the latent continuous variables, $\boldsymbol{e}_i$, for an arbitrary projection f in Eq. (5), which means inference and learning may be difficult. In §3, we address this issue by constraining f to be invertible, and show that this constraint enables tractable exact inference and marginal likelihood computation.

3 Learning & Inference

In this section, we introduce an invertibility condition for our neural projector to tackle the optimization challenge. Specifically, we constrain our neural projector with two requirements: (1) $\dim(\boldsymbol{x}) = \dim(\boldsymbol{e})$ and (2) $f_{\boldsymbol{\phi}}^{-1}$ exists. Invertible transformations have been explored before in independent components analysis (Hyvärinen et al., 2004), gaussianization (Chen and Gopinath, 2001), and deep density models (Dinh et al., 2014, 2016; Kingma and Dhariwal, 2018), for unstructured data. Here, we generalize this style of approach to *structured* learning, and augment it with discrete latent variables (z_i). Under the invertibility condition, we derive a learning algorithm and give another view of our approach revealed by the objective function. Then, we present the architecture of a neural projector we use in experiments: a volume-preserving invertible neural network proposed by Dinh et al. (2014) for independent components estimation.

3.1 Learning with Invertibility

For ease of exposition, we explain the learning algorithm in terms of Markov structure without loss of generality. As shown in Eq. (5), the optimization challenge in our approach comes from the intractability of the marginalized emission factor $p(\boldsymbol{x}_i|z_i)$. If we can marginalize out $\boldsymbol{e}_i$ and compute $p(\boldsymbol{x}_i|z_i)$, then the posterior and marginal likelihood of our Markov-structured model can be computed with the forward-backward algorithm. We can apply Eq. (3) and obtain :

$$p(\boldsymbol{x}_i|z_i;\boldsymbol{\eta},\boldsymbol{\phi}) = \int_{\boldsymbol{e}_i} p_{\boldsymbol{\eta}}(\boldsymbol{e}_i|z_i)\delta(\boldsymbol{x}_i - f_{\boldsymbol{\phi}}(\boldsymbol{e}_i))\mathrm{d}\boldsymbol{e}_i.$$

By using the change of variable rule to the integration, which allows the integration variable $\boldsymbol{e}_i$ to be replaced by $\boldsymbol{x}'_i = f_{\boldsymbol{\phi}}(\boldsymbol{e}_i)$, the marginal emission factor can be computed in closed-form when the invertibility condition is satisfied:

$$\begin{aligned} & p(\boldsymbol{x}_i|\boldsymbol{z}_i;\boldsymbol{\eta},\boldsymbol{\phi}) \\ &= \int_{\boldsymbol{x}'_i} p_{\boldsymbol{\eta}}(f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{x}'_i)|z_i)\delta(\boldsymbol{x}_i - \boldsymbol{x}'_i)\left|\det\frac{\partial f_{\boldsymbol{\phi}}^{-1}}{\partial \boldsymbol{x}'_i}\right|\mathrm{d}\boldsymbol{x}'_i \\ &= p_{\boldsymbol{\eta}}(f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{x}_i)|z_i)\left|\det\frac{\partial f_{\boldsymbol{\phi}}^{-1}}{\partial \boldsymbol{x}_i}\right|, \end{aligned} \tag{6}$$

where $p_{\boldsymbol{\eta}}(\cdot|z)$ is a conditional Gaussian distribution, $\frac{\partial f_{\boldsymbol{\phi}}^{-1}}{\partial \boldsymbol{x}_i}$ is the Jacobian matrix of function $f_{\boldsymbol{\phi}}^{-1}$ at $\boldsymbol{x}_i$, and $\left|\det\frac{\partial f_{\boldsymbol{\phi}}^{-1}}{\partial \boldsymbol{x}_i}\right|$ represents the absolute value of its determinant. This Jacobian term is nonzero and differentiable if and only if $f_{\boldsymbol{\phi}}^{-1}$ exists.

Eq. (6) shows that we can directly calculate the marginal emission distribution $p(\boldsymbol{x}_i|z_i)$. Denote the marginal data likelihood of Gaussian HMM as $p_{\text{HMM}}(\boldsymbol{x})$, then the log marginal data likelihood of our model can be directly written as:

$$\begin{aligned} \log p(\boldsymbol{x}) &= \log p_{\text{HMM}}(f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{x})) \\ &\quad + \sum\nolimits_{i=1}^{\ell} \log\left|\det\frac{\partial f_{\boldsymbol{\phi}}^{-1}}{\partial \boldsymbol{x}_i}\right|, \end{aligned} \tag{7}$$

where $f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{x})$ represents the new sequence of embeddings after applying $f_{\boldsymbol{\phi}}^{-1}$ to each $\boldsymbol{x}_i$. Eq. (7) shows that the training objective of our model is simply the Gaussian HMM log likelihood with an additional Jacobian regularization term. From this view, our approach can be seen as equivalent to reversely projecting the data through $f_{\boldsymbol{\phi}}^{-1}$ to another manifold $\boldsymbol{e}$ that is directly modeled by the Gaussian HMM, with a regularization term. Intuitively, we optimize the reverse projection $f_{\boldsymbol{\phi}}^{-1}$ to modify the $\boldsymbol{e}$ space, making it more appropriate for the syntax model. The Jacobian regularization term accounts for the volume expansion or contraction behavior of the projection. Maximizing it can be thought of as preventing information loss. In the extreme case, the Jacobian determinant is equal to zero, which means the projection is non-invertible and thus information is being lost through the projection. Such "information preserving" regularization is crucial during optimization, otherwise the trivial solution of always projecting data to the same single point to maximize likelihood is viable.[2]

More generally, for an arbitrary syntax model the data likelihood of our approach is:

$$\begin{aligned} p(\boldsymbol{x}) &= \sum\nolimits_{\boldsymbol{z}} \Big(p_{\text{syntax}}(\boldsymbol{z}) \\ &\quad \cdot \prod\nolimits_{i=1}^{\ell} p_{\boldsymbol{\eta}}(f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{x}_i)|z_i)\left|\det\frac{\partial f_{\boldsymbol{\phi}}^{-1}}{\partial \boldsymbol{x}_i}\right|\Big). \end{aligned} \tag{8}$$

If the syntax model itself allows for tractable inference and marginal likelihood computation, the same dynamic program can be used to marginalize out $\boldsymbol{z}$. Therefore, our joint model inherits the tractability of the underlying syntax model.

---

[2] For example, all $\boldsymbol{e}_i$ could learn to be zero vectors, leading to the trivial solution of learning zero mean and zero variance Gaussian emissions achieving infinite data likelihood.

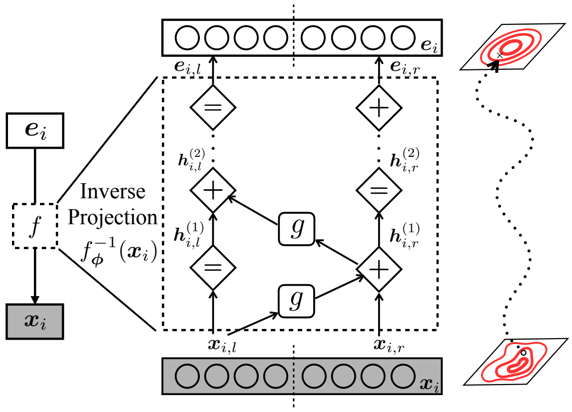

Figure 3: Depiction of the architecture of the inverse projection f_ϕ^{-1} that composes multiple volume-preserving coupling layers, with which we parameterize our model. On the right, we schematically depict how the inverse projection transforms the observed word embedding $\boldsymbol{x}_i$ to a point $\boldsymbol{e}_i$ in a new embedding space.

3.2 Invertible Volume-Preserving Neural Net

For the projection we can use an arbitrary invertible function, and given the representational power of neural networks they seem a natural choice. However, calculating the inverse and Jacobian of an arbitrary neural network can be difficult, as it requires that all component functions be invertible and also requires storage of large Jacobian matrices, which is memory intensive. To address this issue, several recent papers propose specially designed invertible networks that are easily trainable yet still powerful (Dinh et al., 2014, 2016; Jacobsen et al., 2018). Inspired by these works, we use the invertible transformation proposed by Dinh et al. (2014), which consists of a series of "coupling layers". This architecture is specially designed to guarantee a unit Jacobian determinant (and thus the invertibility property).

From Eq. (8) we know that only f_ϕ^{-1} is required for accomplishing learning and inference; we never need to explicitly construct f_ϕ. Thus, we directly define the architecture of f_ϕ^{-1}. As shown in Figure 3, the nonlinear transformation from the observed embedding $\boldsymbol{x}_i$ to $\boldsymbol{h}_i^{(1)}$ represents the first coupling layer. The input in this layer is partitioned into left and right halves of dimensions, $\boldsymbol{x}_{i,l}$ and $\boldsymbol{x}_{i,r}$, respectively. A single coupling layer is defined as:

$$\boldsymbol{h}_{i,l}^{(1)} = \boldsymbol{x}_{i,l}, \qquad \boldsymbol{h}_{i,r}^{(1)} = \boldsymbol{x}_{i,r} + g(\boldsymbol{x}_{i,l}), \qquad (9)$$

where $g : \mathbb{R}^{d_x/2} \rightarrow \mathbb{R}^{d_x/2}$ is the coupling function and can be any nonlinear form. This transformation satisfies $\dim(\boldsymbol{h}^{(1)}) = \dim(\boldsymbol{x})$, and Dinh et al. (2014) show that its Jacobian matrix is triangular with all ones on the main diagonal. Thus the Jacobian determinant is always equal to one (i.e. volume-preserving) and the invertibility condition is naturally satisfied.

To be sufficiently expressive, we compose multiple coupling layers as suggested in Dinh et al. (2014). Specifically, we exchange the role of left and right half vectors at each layer as shown in Figure 3. For instance, from $\boldsymbol{x}_i$ to $\boldsymbol{h}_i^{(1)}$ the left subset $\boldsymbol{x}_{i,l}$ is unchanged, while from $\boldsymbol{h}_i^{(1)}$ to $\boldsymbol{h}_i^{(2)}$ the right subset $\boldsymbol{h}_{i,r}^{(1)}$ remains the same. Also note that composing multiple coupling layers does not change the volume-preserving and invertibility properties. Such a sequence of invertible transformations from the data space $\boldsymbol{x}$ to $\boldsymbol{e}$ is also called normalizing flow (Rezende and Mohamed, 2015).

4 Experiments

In this section, we first describe our datasets and experimental setup. We then instantiate our approach with Markov and DMV-structured syntax models, and report results on POS tagging and dependency grammar induction respectively. Lastly, we analyze the learned latent embeddings.

4.1 Data

For both POS tagging and dependency parsing, we run experiments on the Wall Street Journal (WSJ) portion of the Penn Treebank.[3] To create the observed data embeddings, we train skip-gram word embeddings (Mikolov et al., 2013) that are found to capture syntactic properties well when trained with small context window (Bansal et al., 2014; Lin et al., 2015). Following Lin et al. (2015), the dimensionality d_x is set to 100, and the training context window size is set to 1 to encode more syntactic information. The skip-gram embeddings are trained on the one billion word language modeling benchmark dataset (Chelba et al., 2013) in addition to the WSJ corpus.

4.2 General Experimental Setup

For the neural projector, we employ rectified networks as coupling function g following Dinh et al. (2014). We use a rectified network with an input layer, one hidden layer, and linear output units, the number of hidden units is set to the same as the number of input units. The number of coupling layers are varied as 4, 8, 16 for both tasks.

---

[3] Preprocessing is different for the two tasks, we describe the details in the following subsections.

We optimize marginal data likelihood directly using Adam (Kingma and Ba, 2014). For both tasks in the fully unsupervised setting, we do not tune the hyper-parameters using supervised data.

4.3 Unsupervised POS tagging

For unsupervised POS tagging, we use a Markov-structured syntax model in our approach, which is a popular structure for unsupervised tagging tasks (Lin et al., 2015; Tran et al., 2016).

Setup. Following existing literature, we train and test on the entire WSJ corpus (49208 sentences, 1M tokens). We use 45 tag clusters, the number of POS tags that appear in WSJ corpus. We train the discrete HMM and the Gaussian HMM (Lin et al., 2015) as baselines. For the Gaussian HMM, mean vectors of Gaussian emissions are initialized with the empirical mean of all word vectors with an additive noise. We assume diagonal covariance matrix for $p(\boldsymbol{e}_i|z_i)$ and initialize it with the empirical variance of the word vectors. Following Lin et al. (2015), the covariance matrix is fixed during training. The multinomial probabilities are initialized as $\theta_{kv} \propto \exp(u_{kv})$, where $u_{kv} \sim U[0,1]$. For our approach, we initialize the syntax model and Gaussian parameters with the pre-trained Gaussian HMM. The weights of layers in the rectified network are initialized from a uniform distribution with mean zero and a standard deviation of $\sqrt{1/n_{\text{in}}}$, where n_{in} is the input dimension.[4] We evaluate the performance of POS tagging with both Many-to-One (M-1) accuracy (Johnson, 2007) and V-Measure (VM) (Rosenberg and Hirschberg, 2007). Given a model we found that the tagging performance is well-correlated with the training data likelihood, thus we use training data likelihood as a unsupervised criterion to select the trained model over 10 random restarts after training 50 epochs. We repeat this process 5 times and report the mean and standard deviation of performance.

System	**M-1**	**VM**
w/o hand-engineered features		
Discrete HMM	62.7	53.8
PYP-HMM (Blunsom and Cohn, 2011)	77.5	69.8
NHMM (basic) (Tran et al., 2016)	59.8	54.2
NHMM (+ Conv) (Tran et al., 2016)	74.1	66.1
NHMM (+ Conv & LSTM) (Tran et al., 2016)	79.1	71.7
Gaussian HMM (Lin et al., 2015)	75.4 (1.0)	68.5 (0.5)
Ours (4 layers)	79.5 (0.9)	73.0 (0.7)
Ours (8 layers)	**80.8** (1.3)	**74.1** (0.7)
Ours (16 layers)	73.2 (4.3)	70.5 (2.1)
w/ hand-engineered features		
Feature HMM (Berg-Kirkpatrick et al., 2010)	75.5	–
Brown (+ proto) (Christodoulopoulos et al., 2010)	76.1	68.8
Cluster (word-based) (Yatbaz et al., 2012)	80.2	72.1
Cluster (token-based) (Yatbaz et al., 2014)	79.5	69.1

Table 1: Unsupervised POS tagging results on entire WSJ, compared with other baselines and state-of-the-art systems. Standard deviation is given in parentheses when available.

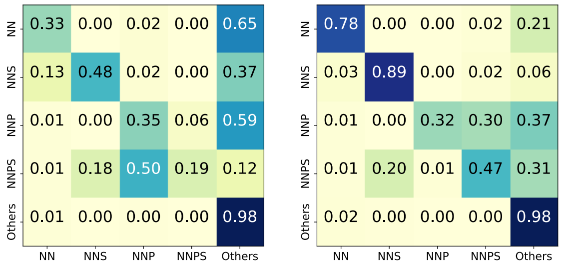

	NN	NNS	NNP	NNPS	Others
NN	0.33	0.00	0.02	0.00	0.65
NNS	0.13	0.48	0.02	0.00	0.37
NNP	0.01	0.00	0.35	0.06	0.59
NNPS	0.01	0.18	0.50	0.19	0.12
Others	0.01	0.00	0.00	0.00	0.98

(a) Gaussian HMM

	NN	NNS	NNP	NNPS	Others
NN	0.78	0.00	0.00	0.02	0.21
NNS	0.03	0.89	0.00	0.02	0.06
NNP	0.01	0.00	0.32	0.30	0.37
NNPS	0.01	0.20	0.01	0.47	0.31
Others	0.02	0.00	0.00	0.00	0.98

(b) Our approach

Figure 4: Normalized Confusion matrix for POS tagging experiments, row label represents the gold tag.

Results. We compare our approach with basic HMM, Gaussian HMM, and several state-of-the-art systems, including sophisticated HMM variants and clustering techniques with hand-engineered features. The results are presented in Table 1. Through the introduced latent embeddings and additional neural projection, our approach improves over the Gaussian HMM by 5.4 points in M-1 and 5.6 points in VM. Neural HMM (NHMM) (Tran et al., 2016) is a baseline that also learns word representation jointly. Both their basic model and extended Conv version does not outperform the Gaussian HMM. Their best model incorporates another LSTM to model long distance dependency and breaks the Markov assumption, yet our approach still achieves substantial improvement over it without considering more context information. Moreover, our method outperforms the best published result that benefits from hand-engineered features (Yatbaz et al., 2012) by 2.0 points on VM.

Confusion Matrix. We found that most tagging errors happen in noun subcategories. Therefore, we do the one-to-one mapping between gold POS tags and induced clusters and plot the normalized confusion matrix of noun subcategories in Figure 4. The Gaussian HMM fails to identify "NN" and "NNS" correctly for most cases, and it often recognizes "NNPS" as "NNP". In contrast, our approach corrects these errors well.

---

[4] This is the default parameter initialization in PyTorch.

4.4 Unsupervised Dependency Parsing without gold POS tags

For the task of unsupervised dependency parse induction, we employ the Dependency Model with Valence (DMV) (Klein and Manning, 2004) as the syntax model in our approach. DMV is a generative model that defines a probability distribution over dependency parse trees and syntactic categories, generating tokens and dependencies in a head-outward fashion. While, traditionally, DMV is trained using gold POS tags as observed syntactic categories, in our approach, we treat each tag as a latent variable, as described in §2.3.

Most existing approaches to this task are not fully unsupervised since they rely on gold POS tags following the original experimental setup for DMV. This is partially because automatically parsing from words is difficult even when using unsupervised syntactic categories (Spitkovsky et al., 2011a). However, inducing dependencies from words alone represents a more realistic experimental condition since gold POS tags are often unavailable in practice. Previous work that has trained from words alone often requires additional linguistic constraints (like sentence internal boundaries) (Spitkovsky et al., 2011a,b, 2012, 2013), acoustic cues (Pate and Goldwater, 2013), additional training data (Pate and Johnson, 2016), or annotated data from related languages (Cohen et al., 2011). Our approach is naturally designed to train on word embeddings directly, thus we attempt to induce dependencies without using gold POS tags or other extra linguistic information.

Setup. Like previous work we use sections 02-21 of WSJ corpus as training data and evaluate on section 23, we remove punctuations and train the models on sentences of length $\leqslant 10$, “head-percolation” rules (Collins, 1999) are applied to obtain gold dependencies for evaluation. We train basic DMV, extended DMV (E-DMV) (Headden III et al., 2009) and Gaussian DMV (which treats POS tag as unknown latent variables and generates observed word embeddings directly conditioned on them following Gaussian distribution) as baselines. Basic DMV and E-DMV are trained with Viterbi EM (Spitkovsky et al., 2010) on unsupervised POS tags induced from our Markov-structured model described in §4.3. Multinomial parameters of the syntax model in both Gaussian DMV and our model are initialized with the pre-trained DMV baseline. Other parameters are initialized in the same way as in the POS tagging experiment. The directed dependency accuracy (DDA) is used for evaluation and we report accuracy on sentences of length $\leqslant 10$ and all lengths. We train the parser until training data likelihood converges, and report the mean and standard deviation over 20 random restarts.

System	$\leqslant 10$	all
w/o gold POS tags		
DMV (Klein and Manning, 2004)	49.6	35.8
E-DMV (Headden III et al., 2009)	52.1	38.2
UR-A E-DMV (Tu and Honavar, 2012)	58.9	46.1
CS* (Spitkovsky et al., 2013)	72.0*	64.4*
Neural E-DMV (Jiang et al., 2016)	55.3	42.7
CRFAE (Cai et al., 2017)	37.2	29.5
Gaussian DMV	55.4 (1.3)	43.1 (1.2)
Ours (4 layers)	58.4 (1.9)	46.2 (2.3)
Ours (8 layers)	**60.2** (1.3)	**47.9** (1.2)
Ours (16 layers)	54.1 (8.5)	43.9 (5.7)
w/ gold POS tags (for reference only)		
DMV (Klein and Manning, 2004)	55.1	39.7
UR-A E-DMV (Tu and Honavar, 2012)	71.4	57.0
MaxEnc (Le and Zuidema, 2015)	73.2	65.8
Neural E-DMV (Jiang et al., 2016)	72.5	57.6
CRFAE (Cai et al., 2017)	71.7	55.7
L-NDMV (Big training data) (Han et al., 2017)	77.2	63.2

Table 2: Directed dependency accuracy on section 23 of WSJ, evaluating on sentences of length $\leqslant 10$ and all lengths. Starred entries ($*$) denote that the system benefits from additional punctuation-based constraints. Standard deviation is given in parentheses when available.

Comparison with other related work. Our model directly observes word embeddings and does not require gold POS tags during training. Thus, results from related work trained on gold tags are not directly comparable. However, to measure how these systems might perform without gold tags, we run three recent state-of-the-art systems in our experimental setting: UR-A E-DMV (Tu and Honavar, 2012), Neural E-DMV (Jiang et al., 2016), and CRF Autoencoder (CRFAE) (Cai et al., 2017).[5] We use unsupervised POS tags (induced from our Markov-structured model) in place of gold tags.[6] We also train basic DMV on gold tags and include several state-of-the-art results on gold tags as reference points.

Results. As shown in Table 2, our approach is able to improve over the Gaussian DMV by 4.8 points on length $\leqslant 10$ and 4.8 points on all

---

[5] For the three systems, we use implementations from the original papers (via personal correspondence with the authors), and tune their hyperparameters on section 22 of WSJ.

[6] Using words directly is not practical because these systems often require a transition probability matrix between input symbols, which requires too much memory.

System	M-1	VM
Ours (4 layers)	78.2	71.2
Ours (8 layers)	72.5	69.7
Ours (16 layers)	67.2	69.2

Table 3: Unsupervised POS tagging results of our approach on WSJ, with random initialization of syntax model.

lengths, which suggests the additional latent embedding layer and neural projector are helpful. The proposed approach yields, to the best of our knowledge,[7] state-of-the-art performance without gold POS annotation and without sentence-internal boundary information. DMV, UR-A E-DMV, Neural E-DMV, and CRFAE suffer a large decrease in performance when trained on unsupervised tags – an effect also seen in previous work (Spitkovsky et al., 2011a; Cohen et al., 2011). Since our approach induces latent POS tags jointly with dependency trees, it may be able to learn POS clusters that are more amenable to grammar induction than the unsupervised tags. We observe that CRFAE underperforms its gold-tag counterpart substantially. This may largely be a result of the model's reliance on prior linguistic rules that become unavailable when gold POS tag types are unknown. Many extensions to DMV can be considered orthogonal to our approach – they essentially focus on improving the syntax model. It is possible that incorporating these more sophisticated syntax models into our approach may lead to further improvements.

4.5 Sensitivity Analysis

Impact of Initialization. In the above experiments we initialize the structured syntax components with the pre-trained Gaussian or discrete baseline, which is shown as a useful technique to help train our deep models. We further study the results with fully random initialization. In the POS tagging experiment, we report the results in Table 3. While the performance with 4 layers is comparable to the pre-trained Gaussian initialization, deeper projections (8 or 16 layers) result in a dramatic drop in performance. This suggests that the structured syntax model with very deep projections is difficult to train from scratch, and a simpler projection might be a good compromise in the random initialization setting.

Different from the Markov prior in POS tagging experiments, our parsing model seems to be quite sensitive to the initialization. For example, directed accuracy of our approach on sentences of length $\leqslant 10$ is below 40.0 with random initialization. This is consistent with previous work that has noted the importance of careful initialization for DMV-based models such as the commonly used harmonic initializer (Klein and Manning, 2004). However, it is not straightforward to apply the harmonic initializer for DMV directly in our model without using some kind of pre-training since we do not observe gold POS.

System	M-1	VM
Gaussian HMM	72.0	65.0
Ours (4 layers)	76.4	69.3
Ours (8 layers)	76.8	69.4
Ours (16 layers)	67.3	62.0

Table 4: Unsupervised POS tagging results on WSJ, with fastText vectors as the observed embeddings.

System	$\leqslant 10$	all
Gaussian DMV	53.6	41.3
Ours (4 layers)	56.9	43.9
Ours (8 layers)	57.1	42.3
Ours (16 layers)	52.9	39.5

Table 5: Directed dependency accuracy on section 23 of WSJ, with fastText vectors as the observed embeddings.

Impact of Observed Embeddings. We investigate the effect of the choice of pre-trained embedding on performance while using our approach. To this end, we additionally include results using fastText embeddings (Bojanowski et al., 2017) – which, in contrast with skip-gram embeddings, include character-level information. We set the context windows size to 1 and the dimension size to 100 as in the skip-gram training, while keeping other parameters set to their defaults. These results are summarized in Table 4 and Table 5. While fastText embeddings lead to reduced performance with our model, our approach still yields an improvement over the Gaussian baseline with the new observed embeddings space.

4.6 Qualitative Analysis of Embeddings

We perform qualitative analysis to understand how the latent embeddings help induce syntactic structures. First we filter out low-frequency words and punctuations in WSJ, and visualize the rest words (10k) with t-SNE (Maaten and Hinton, 2008) under different embeddings. We assign each word with its most likely gold POS tags in WSJ and color them according to the gold POS tags.

---

[7] We tried to be as thorough as possible in evaluation by running top performing systems using our more difficult training setup when this was feasible – but it was not possible to evaluate them all.

Target	Skip-gram	Markov Structure
come	go came follow coming sit	be go do give follow
singing	dancing sing drumming dance dances	dancing drumming marching playing recording
cigars	cigarettes sodas champagne cigar rum	sodas bottles drinks pills cigarettes
newer	flashier fancier conventional low-end new-generation	softer lighter thinner darker smoother
fanciest	priciest up-scale loveliest fancier high-end	liveliest priciest smartest best-run fastest-growing

Table 6: Target words and their 5 nearest neighbors, based on skip-gram embeddings and our learned latent embeddings with Markov-structured syntax model.

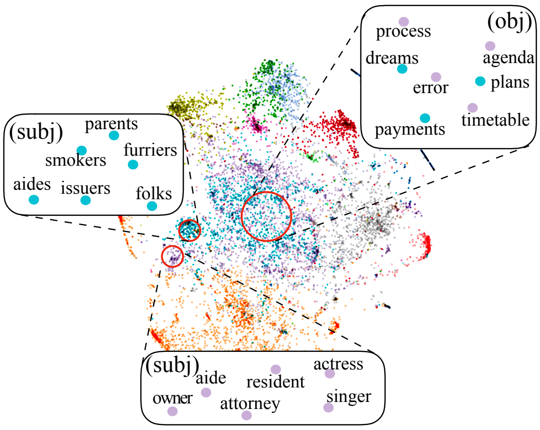

Figure 5: Visualization (t-SNE) of learned latent embeddings with DMV-structured syntax model. Each node represents a word and is colored according to the most likely gold POS tag in the Penn Treebank (best seen in color).

For our Markov-structured model, we have displayed the embedding space in Figure 1(b), where the gold POS clusters are well-formed. Further, we present five example target words and their five nearest neighbors in terms of cosine similarity. As shown in Table 6, the skip-gram embedding captures both semantic and syntactic aspects to some degree, yet our embeddings are able to focus especially on the syntactic aspects of words, in an unsupervised fashion without using any extra morphological information.

In Figure 5 we depict the learned latent embeddings with the DMV-structured syntax model. Unlike the Markov structure, the DMV structure maps a large subset of singular and plural nouns to the same overlapping region. However, two clusters of singular and plural nouns are actually separated. We inspect the two clusters and the overlapping region in Figure 5, it turns out that the nouns in the separated clusters are words that can appear as subjects and, therefore, for which verb agreement is important to model. In contrast, the nouns in the overlapping region are typically objects. This demonstrates that the latent embeddings are focusing on aspects of language that are specifically important for modeling dependency without ever having seen examples of dependency parses.

Some previous work has deliberately created embeddings to capture different notions of similarity (Levy and Goldberg, 2014; Cotterell and Schütze, 2015), while they use extra morphology or dependency annotations to guide the embedding learning, our approach provides a potential alternative to create new embeddings that are guided by structured syntax model, only using unlabeled text corpora.

5 Related Work

Our approach is related to flow-based generative models, which are first described in NICE (Dinh et al., 2014) and have recently received more attention (Dinh et al., 2016; Jacobsen et al., 2018; Kingma and Dhariwal, 2018). This relevant work mostly adopts simple (e.g. Gaussian) and fixed priors and does not attempt to learn interpretable latent structures. Another related generative model class is variational auto-encoders (VAEs) (Kingma and Welling, 2013) that optimize a lower bound on the marginal data likelihood, and can be extended to learn latent structures (Miao and Blunsom, 2016; Yin et al., 2018). Against the flow-based models, VAEs remove the invertibility constraint but sacrifice the merits of exact inference and exact log likelihood computation, which potentially results in optimization challenges (Kingma et al., 2016). Our approach can also be viewed in connection with generative adversarial networks (GANs) (Goodfellow et al., 2014) that is a likelihood-free framework to learn implicit generative models. However, it is non-trivial for a gradient-based method like GANs to propagate gradients through discrete structures.

6 Conclusion

In this work, we define a novel generative approach to leverage continuous word representations for unsupervised learning of syntactic structure. Experiments on both POS induction and unsupervised dependency parsing tasks demonstrate the effectiveness of our proposed approach. Future work might explore more sophisticated invertible projections, or recurrent projections that jointly transform the entire input sequence.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of ACL*.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of HLT-NAACL*, pages 582–590. Association for Computational Linguistics.

Phil Blunsom and Trevor Cohn. 2011. A hierarchical pitman-yor process hmm for unsupervised part of speech induction. In *Proceedings of ACL*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*.

Jiong Cai, Yong Jiang, and Kewei Tu. 2017. Crf autoencoder for unsupervised dependency parsing. In *Proceedings of EMNLP*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Scott Saobing Chen and Ramesh A Gopinath. 2001. Gaussianization. In *Advances in neural information processing systems*.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised pos induction: How far have we come? In *Proceedings of EMNLP*.

Shay B Cohen, Dipanjan Das, and Noah A Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of EMNLP*.

Shay B Cohen and Noah A Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of HLT-NAACL*.

Michael Collins. 1999. *HEAD-DRIVEN STATISTICAL MODELS FOR NATURAL LANGUAGE PARSING*. Ph.D. thesis, University of Pennsylvania.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of NAACL-HLT*.

Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*.

Wenjuan Han, Yong Jiang, and Kewei Tu. 2017. Dependency grammar induction with neural lexicalization and big training data. In *Proceedings of EMNLP*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of ACL*.

William P Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of HLT-NAACL*.

Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. 2004. *Independent component analysis*, volume 46. John Wiley & Sons.

Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. 2018. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of EMNLP*.

Mark Johnson. 2007. Why doesnt em find good hmm pos-taggers? In *Proceedings of the EMNLP-CoNLL*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of NAACL-HLT*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised pos induction with word embeddings. In *Proceedings of the NAACL-HLT*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of EMNLP*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

John K Pate and Sharon Goldwater. 2013. Unsupervised dependency parsing with acoustic cues. *Transactions of the Association for Computational Linguistics*.

John K Pate and Mark Johnson. 2016. Grammar induction from (lots of) words alone. In *Proceedings of COLING*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of HLT-NAACL*.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of ICML*.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of EMNLP-CoNLL*.

Valentin I Spitkovsky, Hiyan Alshawi, Angel X Chang, and Daniel Jurafsky. 2011a. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of EMNLP*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011b. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of CoNLL*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2012. Capitalization cues improve dependency grammar induction. In *Proceedings of NAACL-HLT Workshop on the Induction of Linguistic Structure*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of EMNLP*.

Valentin I Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *Proceedings of CoNLL*.

Karl Stratos, Michael Collins, and Daniel Hsu. 2016. Unsupervised part-of-speech tagging with anchor hidden markov models. *Transactions of the Association for Computational Linguistics*.

Ke M Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*.

Kewei Tu and Vasant Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of EMNLP-CoNLL*.

Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. 2012. Learning syntactic categories using paradigmatic representations of word context. In *Proceedings of EMNLP-CoNLL*.

Mehmet Ali Yatbaz, Enis Rıfat Sert, and Deniz Yuret. 2014. Unsupervised instance-based part of speech induction using probable substitutes. In *Proceedings of COLING*.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of ACL*.

Dynamic Oracles for Top-Down and In-Order Shift-Reduce Constituent Parsing

Daniel Fernández-González and **Carlos Gómez-Rodríguez**
Universidade da Coruña
FASTPARSE Lab, LyS Research Group, Departamento de Computación
Campus de Elviña, s/n, 15071 A Coruña, Spain
d.fgonzalez@udc.es, carlos.gomez@udc.es

Abstract

We introduce novel dynamic oracles for training two of the most accurate known shift-reduce algorithms for constituent parsing: the top-down and in-order transition-based parsers. In both cases, the dynamic oracles manage to notably increase their accuracy, in comparison to that obtained by performing classic static training. In addition, by improving the performance of the state-of-the-art in-order shift-reduce parser, we achieve the best accuracy to date (92.0 F1) obtained by a fully-supervised single-model greedy shift-reduce constituent parser on the WSJ benchmark.

1 Introduction

The shift-reduce transition-based framework was initially introduced, and successfully adapted from the dependency formalism, into constituent parsing by Sagae and Lavie (2005), significantly increasing phrase-structure parsing performance.

A shift-reduce algorithm uses a sequence of transitions to modify the content of two main data structures (a buffer and a stack) and create partial phrase-structure trees (or constituents) in the stack to finally produce a complete syntactic analysis for an input sentence, running in linear time. Initially, Sagae and Lavie (2005) suggested that those partial phrase-structure trees be built in a *bottom-up* manner: two adjacent nodes already in the stack are combined under a non-terminal to become a new constituent. This strategy was followed by many researchers (Zhang and Clark, 2009; Zhu et al., 2013; Watanabe and Sumita, 2015; Mi and Huang, 2015; Crabbé, 2015; Cross and Huang, 2016b; Coavoux and Crabbé, 2016; Fernández-González and Gómez-Rodríguez, 2018) who managed to improve the accuracy and speed of the original Sagae and Lavie's bottom-up parser. With this, shift-reduce algorithms have become competitive, and are the fastest alternative to perform phrase-structure parsing to date.

Some of these attempts (Cross and Huang, 2016b; Coavoux and Crabbé, 2016; Fernández-González and Gómez-Rodríguez, 2018) introduced *dynamic oracles* (Goldberg and Nivre, 2012), originally designed for transition-based dependency algorithms, to bottom-up constituent parsing. They propose to use these dynamic oracles to train shift-reduce parsers instead of a traditional *static oracle*. The latter follows the standard procedure that uses a gold sequence of transitions to train a model for parsing new sentences at test time. A shift-reduce parser trained with this approach tends to be prone to suffer from error propagation (i.e. errors made in previous states are propagated to subsequent states, causing further mistakes in the transition sequence). Dynamic oracles (Goldberg and Nivre, 2012) were developed to minimize the effect of error propagation by training parsers under closer conditions to those found at test time, where mistakes are inevitably made. They are designed to guide the parser through any state it might reach during learning time. This makes it possible to introduce error exploration to force the parser to go through non-optimal states, teaching it how to recover from mistakes and lose the minimum number of gold constituents.

Alternatively, some researchers decided to follow a different direction and explore non-bottom-up strategies for producing phrase-structure syntactic analysis.

On the one hand, (Dyer et al., 2016; Kuncoro et al., 2017) proposed a *top-down* transition-based algorithm, which creates a phrase structure tree in the stack by first choosing the non-terminal on the top of the tree, and then considering which should be its child nodes. In contrast to the bottom-up approach, this top-down strategy adds a lookahead

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1303–1313
Brussels, Belgium, October 31 - November 4, 2018.

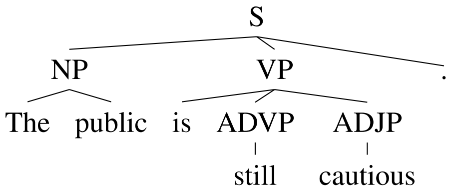

Figure 1: Simplified constituent tree, taken from English WSJ §22.

guidance to the parsing process, while it loses rich local features from partially-built trees.

On the other hand, Liu and Zhang (2017a) recently developed a novel strategy that finds a compromise between the strengths of top-down and bottom-up approaches, resulting in state-of-the-art accuracy. Concretely, this parser builds the tree following an *in-order* traversal: instead of starting the tree from the top, it chooses the non-terminal of the resulting subtree after having the first child node in the stack. In that way each partial constituent tree is created in a bottom-up manner, but the non-terminal node is not chosen when all child nodes are in the stack (as a purely bottom-up parser does), but after the first child is considered.

Liu and Zhang (2017a) report that the top-down approach is on par with the bottom-up strategy in terms of accuracy and the in-order parser yields the best accuracy to date on the WSJ. However, despite being two adequate alternatives to the traditional bottom-up strategy, no further work has been undertaken to improve their performance.[1]

We propose what, to our knowledge, are the first optimal dynamic oracles for both the top-down and in-order shift-reduce parsers, allowing us to train these algorithms with exploration. The resulting parsers outperform the existing versions trained with static oracles on the WSJ Penn Treebank (Marcus et al., 1993) and Chinese Treebank (CTB) benchmarks (Xue et al., 2005). The version of the in-order parser trained with our dynamic oracle achieves the highest accuracy obtained so far by a single fully-supervised greedy shift-reduce system on the WSJ.

2 Preliminaries

The original transition system of Sagae and Lavie (2005) parses a sentence from left to right by reading (moving) words from a *buffer* to a *stack*, where partial subtrees are built. This process is performed by a sequence of Shift (for reading) and Reduce (for building) transitions that will lead the parser through different states or parser configurations until a terminal one is reached. While in the bottom-up strategy the Reduce transition is in charge of labeling the partial subtree with a non-terminal at the same time the tree is built, Dyer et al. (2016) and Liu and Zhang (2017a) introduce a novel transition to choose the non-terminal on top, leaving the Reduce transition just to create the subtree under the previously decided non-terminal. We will now explain more in detail both the top-down and the in-order transition systems.

In both transition systems, parser configurations have the form $c = \langle \Sigma, i, f, \gamma, \alpha \rangle$, where Σ is a stack of constituents, i is the position of the leftmost unprocessed word in the buffer (which is the next to be pushed onto the stack), f is a boolean variable used by the in-order transition system to mark if a configuration is terminal or not and with no value in top-down parser configurations, γ is the set of constituents that have already been built, and α is the set of non-terminal nodes that are currently in the stack.

Each constituent is represented as a tuple (X, l, r), where X is a non-terminal and l and r are integers defining its span. Constituents are composed of one or several words or constituents, and just one non-terminal node on top. Each word w_i is represented as $(w, i, i+1)$. To define our oracles, we will need to represent each non-terminal node of the tree as (X, j), where j has the value of i when X is included in the stack and is used to keep them in order.[2]

For instance, the phrase-structure tree in Figure 1 can be decomposed as the following set of gold constituents: {(S, 0, 6), (NP, 0, 2), (VP, 2, 5), (ADVP, 3, 4), (ADJP, 4, 5)}. In addition, the ordered set of gold non-terminal nodes added to the stack while following a top-down strategy will be {(S, 0), (NP, 0), (VP, 2), (ADVP, 3), (ADJP, 4)} and, according to an in-order approach, {(NP, 1), (S, 2), (VP, 3), (ADVP, 4), (ADJP, 5)}. It is worth mentioning that the index of non-terminal nodes in the top-down method is the same as the leftmost span index of the constituent that it will produce. However, this does not hold in the in-order approach, as the leftmost child is fully processed before the node is added to the stack, so the index

---

[1] In parallel to this work, Fried and Klein (2018) present a non-optimal dynamic oracle for training the top-down parser.

[2] When two or more non-terminals share their labels within the tree, we use a secondary index to make them unique.

for the node will point to the leftmost span index of the second leftmost child.

Note that the information about the span of a constituent, the set of predicted constituents γ and the set α of predicted non-terminal nodes in the stack is not used by the original top-down and in-order parsers. However, we need to include it in parser configurations at learning time to allow an efficient implementation of the proposed dynamic oracles.

Given an input string $w_0 \cdots w_{n-1}$, the in-order parsing process starts at the initial configuration $c_s(w_0 \ldots w_{n-1}) = \langle [\,], 0, \mathit{false}, \{\}, \{\}\rangle$ and, after applying a sequence of transitions, it ends in a terminal configuration $\langle (S, 0, n), n, \mathit{true}, \gamma, \alpha\rangle$, where n is the number of words in the input sentence. The top-down transition system shares the same form for the initial and terminal configurations, except for the fact that variable f has no value in both cases.

Figure 2 shows the available transitions in the top-down algorithm. In particular, the Shift transition moves the first (leftmost) word in the buffer to the stack; the Non-Terminal-X transition pushes onto the stack the non-terminal node X that should be on top of a coming constituent, and the Reduce transition pops the topmost stack nodes until the first non-terminal node appears (which is also popped) and combines them into a constituent with this non-terminal node as their parent, pushing this new constituent into the stack. Note that every reduction action will add a new constituent to γ and remove a non-terminal node from α, and every Non-Terminal transition will include a new non-terminal node in α. Figure 3 shows the top-down transition sequence that produces the phrase-structure tree in Figure 1.

In Figure 4 we describe the available transitions in the in-order algorithm. The Shift, Non-Terminal-X and Reduce transitions have the same behavior as defined for the top-down transition system, except that the Reduce transition not only pops stack nodes until finding a non-terminal node (also removed from the stack), but also the node below this non-terminal node, and combines them into a constituent spanning all the popped nodes with the non-terminal node on top. And, finally, a Finish transition is also available to end the parsing process. Figure 5 shows the in-order transition sequence that outputs the constituent tree in Figure 1.

The standard procedure to train a greedy shift-reduce parser consists of training a classifier to approximate an *oracle*, which chooses optimal transitions with respect to gold parse trees. This classifier will greedily choose which transition sequence the parser should apply at test time.

Depending on the strategy used for training the parser, oracles can be static or dynamic. A static oracle trains the parser only on gold transition sequences, while a dynamic one can guide the parser through any possible transition path, allowing the exploration of non-optimal sequences.

3 Dynamic Oracles

Previous work such as (Cross and Huang, 2016b; Coavoux and Crabbé, 2016; Fernández-González and Gómez-Rodríguez, 2018) has introduced and successfully applied dynamic oracles for bottom-up phrase-structure parsing. We present dynamic oracles for training the top-down and in-order transition-based constituent parsers.

Goldberg and Nivre (2012) show that implementing a dynamic oracle reduces to defining a *loss function* on configurations to measure the distance from the best tree they can produce to the gold parse. This allows us to compute which transitions will lead the parser to configurations where the minimum number of mistakes are made.

3.1 Loss function

According to Fernández-González and Gómez-Rodríguez (2018), we can define a loss function in constituent parsing as follows: given a parser configuration c and a gold tree t_G, a loss function $\ell(c)$ is implemented as the minimum Hamming loss between t and t_G, ($\mathcal{L}(t, t_G)$), where t is the already-built tree of a configuration c' reachable from c (written as $c \rightsquigarrow t$). This Hamming loss is computed as the size of the symmetric difference between the set of constituents γ and γ_G in the trees t and t_G, respectively. Therefore, the loss function is defined as:

$$\ell(c) = \min_{\gamma | c \rightsquigarrow \gamma} \mathcal{L}(\gamma, \gamma_G) = |\gamma_G \setminus \gamma| + |\gamma \setminus \gamma_G|$$

and, according to the authors, it can be efficiently computed for a non-binary bottom-up transition system by counting the individually unreachable arcs from configuration c ($|\mathcal{U}(c, \gamma_G)|$) plus the erroneous constituents created so far ($|\gamma_c \setminus \gamma_G|$):

$$\ell(c) = \min_{\gamma | c \rightsquigarrow \gamma} \mathcal{L}(\gamma, \gamma_G) = |\mathcal{U}(c, \gamma_G)| + |\gamma_c \setminus \gamma_G|$$

$$
\begin{array}{ll}
\text{Shift:} & \langle \Sigma, i, /, \gamma, \alpha \rangle \Rightarrow \langle \Sigma | (w_i, i, i+1), i+1, /, \gamma \cup \{(w_i, i, i+1)\}, \alpha \rangle \\
\text{Non-Terminal-X:} & \langle \Sigma, i, /, \gamma, \alpha \rangle \Rightarrow \langle \Sigma | (X, i), i, /, \gamma, \alpha \cup \{(X, i)\} \rangle \\
\text{Reduce:} & \langle \Sigma | (X, j) | (Y_1, m_0, m_1) | ... | (Y_k, m_{k-1}, m_k), i, /, \gamma, \alpha \rangle \\
 & \qquad \Rightarrow \langle \Sigma | (X, j, m_k), i, /, \gamma \cup \{(X, j, m_k)\}, \alpha \setminus \{(X, j)\} \rangle
\end{array}
$$

Figure 2: Transitions of a top-down constituent parser.

Transition	Σ	Buffer
	[ ]	[The, ...]
NT_{S}	[S]	[The, ...]
NT_{NP}	[S, NP]	[The, ...]
SH	[S, NP, The]	[public, ...]
SH	[S, NP, The, public]	[is, ...]
RE	[S, **NP**]	[is, ...]
NT_{VP}	[S, **NP**, VP]	[is, ...]
SH	[S, **NP**, VP, is]	[still, ...]
NT_{ADVP}	[S, **NP**, VP, is, ADVP]	[still, ...]
SH	[S, **NP**, VP, is, ADVP, still]	[cautious, ...]
RE	[S, **NP**, VP, is, **ADVP**]	[cautious, ...]
NT_{ADJP}	[S, **NP**, VP, is, **ADVP**, ADJP]	[cautious, ...]
SH	[S, **NP**,VP, is, **ADVP**, ADJP, cautious]	[**.**]
RE	[S, **NP**, VP, is, **ADVP**, **ADJP**]	[**.**]
RE	[S, **NP**, **VP**]	[**.**]
SH	[S, **NP**, **VP**, **.**]	[ ]
RE	[**S**]	[ ]

Figure 3: Transition sequence for producing the constituent tree in Figure 1 using a top-down parser. SH = Shift, NT_{X} = Non-Terminal-X and RE = Reduce. Already-built constituents are marked in bold.

We adapt the latter to efficiently implement a loss function for the top-down and in-order strategies.

While in bottom-up parsing constituents are created at once by a Reduce transition, in the other two approaches a Non-Terminal transition begins the process by naming the future constituent and a Reduce transition builds it by setting its span and children. Therefore, a Non-Terminal transition that deviates from the non-terminals expected in the gold tree will eventually produce a wrong constituent in future configurations, so it should be penalized. Additionally, a sequence of gold Non-Terminal transitions may also lead to a wrong final parse if they are applied in an incorrect order. Then, the computation of the Hamming loss in top-down and in-order phrase-structure parsing adds two more terms to the bottom-up loss expression: (1) the number of predicted non-terminal nodes that are currently in the stack (α_c),[3] but not included in the set of gold non-terminal nodes (α_G), and (2) the number of gold non-terminal nodes in the stack that are out of order with respect to the order needed in the gold tree:

$$
\ell(c) = \min_{\gamma | c \rightsquigarrow \gamma} \mathcal{L}(\gamma, \gamma_G) = |\mathcal{U}(c, \gamma_G)| + |\gamma_c \setminus \gamma_G|
$$

$$
+ |\alpha_c \setminus \alpha_G| + out\_of\_order(\alpha_c, \alpha_G)
$$

This loss function is used to implement a dynamic oracle that, when given any parser configuration, will return the set of transitions τ that do not increase the overall loss (i.e., $\ell(\tau(c)) - \ell(c) = 0$), leading the system through optimal configurations that minimize Hamming loss with respect to t_G.

As suggested by (Coavoux and Crabbé, 2016; Fernández-González and Gómez-Rodríguez, 2018), *constituent reachability* can be used to efficiently compute the first term of the symmetric difference ($|\gamma_G \setminus \gamma|$), by simply counting the gold constituents that are individually unreachable from configuration c, as we describe in the next subsection.

The second and third terms of the loss ($|\gamma_c \setminus \gamma_G|$ and $|\alpha_c \setminus \alpha_G|$) can be trivially computed and are used to penalize false positives (extra erroneous constituents) so that final F-score is not harmed due to the decrease of precision, as pointed out by (Coavoux and Crabbé, 2016; Fernández-González and Gómez-Rodríguez, 2018). Note that it is crucial that the creation of non-gold Non-Terminal transitions is avoided, since these might not affect the creation of gold constituents, however, they will certainly lead the parser to the creation of extra erroneous constituents in future steps.

Finally, the function out_of_order of the last term can be implemented by computing the *longest increasing subsequence* of gold non-terminal nodes in the stack, where the order relation is given by the order of non-terminals (provided by their associated index) in the transition sequence that builds the gold tree (this order is unique, as none of our two parsers of interest have spurious ambiguity). Obtaining the longest increasing subsequence is a well-known problem solvable in time O(n log n) (Fredman, 1975), where n denotes the length of the input sequence. Once we have the largest possible sub-

---

[3] Note that we only consider predicted non-terminal nodes still in the stack, since wrong non-terminal nodes that have been already reduced are included in the loss as erroneous constituents.

Shift: $\langle \Sigma, i, \mathit{false}, \gamma, \alpha \rangle \Rightarrow \langle \Sigma | (w_i, i, i+1), i+1, \mathit{false}, \gamma \cup \{(w_i, i, i+1)\}, \alpha \rangle$

Non-Terminal-X: $\langle \Sigma, i, \mathit{false}, \gamma, \alpha \rangle \Rightarrow \langle \Sigma | (X, i), i, \mathit{false}, \gamma, \alpha \cup \{(X, i)\} \rangle$

Reduce: $\langle \Sigma | (Y_1, m_0, m_1) | (X, j) | ... | (Y_k, m_{k-1}, m_k), i, \mathit{false}, \gamma, \alpha \rangle$
$\Rightarrow \langle \Sigma | (X, m_0, m_k), i, \mathit{false}, \gamma \cup \{(X, m_0, m_k)\}, \alpha \setminus \{(X, j)\} \rangle$

Finish: $\langle (S, 0, n), n, \mathit{false}, \gamma, \alpha \rangle \Rightarrow \langle (S, 0, n), n, \mathit{true}, \gamma, \alpha \rangle$

Figure 4: Transitions of a in-order constituent parser.

Transition	Σ	Buffer
	[ ]	[The, ...]
SH	[The]	[public, ...]
NT_{NP}	[The, NP]	[public, ...]
SH	[The, NP, public]	[is, ...]
RE	[**NP**]	[is, ...]
NT_{S}	[**NP**, S]	[is, ...]
SH	[**NP**, S, is]	[still, ...]
NT_{VP}	[**NP**, S, is, VP]	[still, ...]
SH	[**NP**, S, is, VP, still]	[cautious, ...]
NT_{ADVP}	[**NP**, S, is, VP, still, ADVP]	[cautious, ...]
RE	[**NP**, S, is, VP, **ADVP**]	[cautious, ...]
SH	[**NP**, S, is, VP, **ADVP**, cautious]	[.]
NT_{ADJP}	[**NP**, S, is,VP, **ADVP**, cautious, ADJP]	[.]
RE	[**NP**, S, is, VP, **ADVP**, **ADJP**]	[.]
RE	[**NP**, S, **VP**]	[.]
SH	[**NP**, S, **VP**, **.**]	[ ]
RE	[**S**]	[ ]
FI	[**S**]	[ ]

Figure 5: Transition sequence for producing the constituent tree in Figure 1 using an in-order parser. SH = Shift, NT_X = Non-Terminal-X, RE = Reduce and FI = Finish. Already-built constituents are marked in bold.

sequence of gold non-terminal nodes in our configuration's stack that is compatible with the gold order, the remaining ones give us the number of erroneous constituents that we will unavoidably generate, even in the best case, due to building them in an incorrect order.

We will prove below that this loss formulation returns the exact loss and the resulting dynamic oracle is correct.

3.2 Constituent reachability

We now show how the computation of the set of reachable constituents developed for bottom-up parsing in (Coavoux and Crabbé, 2016; Fernández-González and Gómez-Rodríguez, 2018) can be extended to deal with the top-down and in-order strategies.

Top-down transition system Let γ_G and α_G be the set of gold constituents and the set of gold non-terminal nodes, respectively, for our current input. We say that a gold constituent $(X, l, r) \in \gamma_G$ is reachable from a configuration $c = \langle \Sigma, j, /, \gamma_c, \alpha_c \rangle$ with $\Sigma = [(Y_p, i_p, i_{p-1}) \cdots (Y_2, i_2, i_1) | (Y_1, i_1, j)]$, and it is included in the set of *individually reachable constituents* $\mathcal{R}(c, \gamma_G)$, iff it satisfies one of the following conditions:[4]

(i) $(X, l, r) \in \gamma_c$ (i.e. it has already been created and, therefore, it is reachable by definition).

(ii) $j \leq l < r \wedge (X, l) \notin \alpha_c$ (i.e. the words dominated by the gold constituent are still in the buffer and the non-terminal node that begins its creation has not been added to the stack yet; therefore, it can be still created after pushing the correct non-terminal node and shifting the necessary words).

(iii) $l \in \{i_k \mid 1 \leq k \leq p\} \wedge j \leq r \wedge (X, l) \in \alpha_c$ (i.e. its span is partially or completely in the stack and the corresponding non-terminal node was already added to the stack, then, by shifting more words or/and reducing, the constituent can still be created).

In-order transition system Let γ_G and α_G be the set of gold constituents and the set of gold non-terminal nodes, respectively, for our current input. We say that a gold constituent $(X, l, r) \in \gamma_G$ is reachable from a configuration $c = \langle \Sigma, j, \mathit{false}, \gamma_c, \alpha_c \rangle$ with $\Sigma = [(Y_p, i_p, i_{p-1}) \cdots (Y_2, i_2, i_1) | (Y_1, i_1, j)]$, and it is included in the set of *individually reachable constituents* $\mathcal{R}(c, \gamma_G)$, iff it satisfies one of the following conditions:

(i) $(X, l, r) \in \gamma_c$ (i.e. it has already been created).

(ii) $j \leq l < r$ (i.e. the constituent is entirely in the buffer, then it can be still built).

(iii) $l \in \{i_k \mid 1 \leq k \leq p\} \wedge j \leq r \wedge (X, m) \notin \alpha_c$ (i.e. its first child is still a totally- or partially-built constituent on top of the stack and the non-terminal node has not been created yet;

---

[4] Please note that elements from the stack can be an already-built constituent, a shifted word or a non-terminal node. Therefore, (Y_p, i_p, i_{p-1}), (Y_2, i_2, i_1) and (Y_1, i_1, j) should be represented as (Y_p, i_{p-1}), (Y_2, i_1) and (Y_1, j), respectively, when they are non-terminal nodes. We omit this for simplicity.

therefore, it has to wait till the first child is completed (if it is still pending) and, then, it can be still created by pushing onto the stack the correct non-terminal node and shifting more words if necessary).

(iv) $l \in \{i_k \mid 1 \leq k \leq p\} \wedge j \leq r \wedge (X, m) \in \alpha_c \wedge \exists(Y, l, m) \in \Sigma$ (i.e. its span is partially or completely in the stack, and its first child (which is an alredy-built constituent) and the non-terminal node assigned are adjacent, thus, by shifting more words or/and reducing, the constituent can still be built).

In both transition systems, the set of *individually unreachable constituents* $\mathcal{U}(c, \gamma_G)$ with respect to the set of gold constituents γ_G can be easily computed as $\gamma_G \setminus \mathcal{R}(c, \gamma_G)$ and will contain the gold constituents that can no longer be built.

3.3 Correctness

We will now prove that the above expression of $\ell(c)$ indeed provides the minimum possible Hamming loss to the gold tree among all the trees that are reachable from configuration c. This implies correctness (or optimality) of our oracle.

To do so, we first show that both algorithms are constituent-decomposable. This amounts to saying that if we take a set of m constituents that are tree-compatible (can appear together in a constituent tree, meaning that no pair of constituent spans overlap unless one is a subset of the other) and individually reachable from a configuration c, then the set is also reachable as a whole.

We prove this by induction on m. The base case ($m = 1$) is trivial. Let us suppose that constituent-decomposability holds for any set of m tree-compatible constituents. We will show that it also holds for any set T of $m+1$ tree-compatible constituents.

Let (X, l, r) be one of the constituents in T such that $r = \min\{r' \mid (X', l', r') \in T\}$ and $l = \max\{l' \mid (X', l', r) \in T\}$. Let $T' = T \setminus \{(X, l, r)\}$. Since T' has m constituents, by induction hypothesis, T' is a reachable set from configuration c.

Since (X, l, r) is individually reachable by hypothesis, it must satisfy at least one of the conditions for constituent reachability. As these conditions are different for each particular algorithm, we continue the proof separately for each:

Top-down constituent-decomposability In this case, we enumerated three constituent reachability conditions, so we divide the proof into three cases:

If the first condition holds, then the constituent (X, l, r) has already been created in c. Thus, it will still be present after applying any of the possible transition sequences that build T' starting from c. Hence, $T = T' \cup \{(X, l, r)\}$ is reachable from c.

If the second condition holds, then $j \leq l < r$ and the constituent (X, l, r) can be created by $l - j$ Shift transitions, followed by one Non-Terminal transition, $r - l$ Shift transitions and one Reduce transition. This will leave the parser in a configuration whose value of j is r, and where stack elements with left span index $\leq l$ (apart from those referencing the new non-terminal and its leftmost child) have not changed. Thus, constituents of T' are still individually reachable in this configuration, as their left span index is either $\geq r$ (and then they meet the second reachability condition) or $\leq l$ (and then they meet the third), so T is reachable from c.

Finally, if the third condition holds, then we can create (X, l, r) by applying $r - j$ Shift transitions followed by a sequence of Reduce transitions stopping when we obtain (X, l, r) on the stack (this will always happen after a finite number of such transitions, as the reachability condition guarantees that l is the left span index of some constituent already on the stack, and that (X, l) is on the stack). Following the same reasoning as in the previous case regarding the resulting parser configuration, we conclude that T is reachable from c.

With this we have shown the induction step, and thus constituent decomposability for the top-down parser.

In-order constituent decomposability The in-order parser has four constituent reachability conditions. Analogously to the previous case, we prove the reachability of T by case analysis.

If the first condition holds, then we have a situation where the constituent (X, l, r) has already been created in c, so reachability of T follows from the same reasoning as for the first condition in the top-down case.

If the second condition holds, we have $j \leq l < r$ and the constituent (X, l, r) can be created by $l - j + 1$ Shift transitions (where the last one shifts a word that will be assigned as left child of the new constituent), followed by the relevant Non-Terminal-X transition, $r - l - 1$ more Shift transitions and one Reduce transition. After this, the parser will be in a configuration where j takes the value r, where we can use the same reasoning as in the second condition of the top-down parser to show that all constituents of T' are still reachable, proving reachability of T.

For the third condition, the proof is analogous but the combination of transitions that creates the non-terminal starts with a sequence composed of Reduce transitions (when there is a non-terminal at the top of the stack) or Non-Terminal-Y transitions for arbitrary Y (when the top of the stack is a constituent) until the top node on the stack is a constituent with left span index l (this ensures that the constituent at the top of the stack can serve as leftmost child for our desired constituent), followed by a Non-Terminal-X, $r-j$ Shift transitions and one Reduce transition.

Finally, for the fourth condition, the reasoning is again analogous, but the computation leading to the non-terminal starts with as many Reduce transitions as non-terminal nodes located above (X, m) in the stack (if any). If we call j the index associated to the resulting transition, then it only remains to apply $r - j$ Shift transitions followed by a Reduce transition.

Optimality With this, we have shown constituent decomposability for both parsing algorithms. This means that, for a configuration c, and a set of constituents that are individually reachable from c, there is always some computation that can build them all. This facilitates the proof that the loss function is correct.

To finish the proof, we observe the following:

- Let c' be a final configuration reachable from c. The set $(\gamma_{c'} \setminus \gamma_G)$, representing erroneous constituents that have been built, will always contain at least $|\gamma_c \setminus \gamma_G|$, as the algorithm never deletes constituents.
- In addition, c' will contain one erroneous constituent for each element of $(\alpha_c \setminus \alpha_G)$, as once a non-terminal node is on the stack, there is no way to reach a final configuration without using it to create an erroneous constituent. Note that these erroneous constituents do not overlap those arising from the previous item, as γ_c stores already-built constituents and α_c non-terminals that have still not been used to build a constituent.
- Given a subset $\mathcal{S}$ of $\mathcal{R}(c, \gamma_G)$, the previously shown constituent decomposability property implies that there exists at least one transition sequence starting from c that generates the tree $\mathcal{S} \cup (\gamma_c \setminus \gamma_G) \cup E$, where E is a set of erroneous constituents containing one such constituent per element of $(\alpha_c \setminus \alpha_G)$. This tree has loss $|t_G| - (|\gamma_c \cup \mathcal{S}|) + |\gamma_c \setminus \gamma_G| + |\alpha_c \setminus \alpha_G|$. The term $|t_G| - (|\gamma_c \cup \mathcal{S}|)$ corresponds to missed constituents (gold constituents that have not been already created and are not created as part of $\mathcal{S}$), the other two to erroneous constituents.
- As we have shown that the erroneous constituents arising from $(\gamma_{c'} \setminus \gamma_G)$ and $(\alpha_c \setminus \alpha_G)$ are unavoidable, computations yielding a tree with minimum loss are those that maximize $|\gamma_c \cup \mathcal{S}|$ in the previous term. In general, the largest possible $|\mathcal{S}|$ is for $\mathcal{S} = \mathcal{R}(c, \gamma_G)$. In that case, we would correctly generate every reachable constituent and the loss would be

$$\ell(c) = |\mathcal{U}(c, \gamma_G)| + |\gamma_c \setminus \gamma_G| + |\alpha_c \setminus \alpha_G|$$

However, we additionally want to generate constituents in the correct order, and this may not be possible if we have already shifted some of them into the stack in a wrong order. The function out_of_order gives us the number of reachable constituents that are lost for this cause in the best case. Thus, indeed, the expression

$$\ell(c) = |\mathcal{U}(c, \gamma_G)| + |\gamma_c \setminus \gamma_G| + |\alpha_c \setminus \alpha_G| + out\_of\_order(\alpha_c, \alpha_G)$$

provides the minimum loss from configuration c.

4 Experiments

4.1 Data

We test the two proposed approaches on two widely-used benchmarks for constituent parsers: the Wall Street Journal (WSJ) sections of the English Penn Treebank[5] (Marcus et al., 1993) and version 5.1 of the Penn Chinese Treebank (CTB)[6] (Xue et al., 2005). We use the same predicted POS tags and pre-trained word embeddings as Dyer et al. (2016) and Liu and Zhang (2017a).

---

[5] Sections 2-21 are used as training data, Section 22 for development and Section 23 for testing

[6] Articles 001- 270 and 440-1151 are taken for training, articles 301-325 for system development, and articles 271-300 for final testing

4.2 Neural Model

To perform a fair comparison, we define the novel dynamic oracles on the original implementations of the top-down parser by Dyer et al. (2016) and in-order parser by Liu and Zhang (2017a), where parsers are trained with a traditional static oracle. Both implementations follow a stack-LSTM approach to represent the stack and the buffer, as well as a vanilla LSTM to represent the action history. In addition, they also use a bi-LSTM as a compositional function for representing constituents in the stack. Concretely, this consists in computing the composition representation s_{comp} as:

$$s_{comp} = (LSTM_{fwd}[e_{nt}, s_0, ..., s_m]; \\ LSTM_{bwd}[e_{nt}, s_m, ..., s_0])$$

where e_{nt} is the vector representation of a non-terminal, and s_i, $i \in [0, m]$ is the ith child node.

Finally, the exact same word representation strategy and hyper-parameter values as (Dyer et al., 2016) and (Liu and Zhang, 2017a) are used to conduct the experiments.

4.3 Error exploration

In order to benefit from training a parser by a dynamic oracle, errors should be made during the training process so that the parser can learn to avoid and recover from them. Unlike more complex error-exploration strategies as those studied in (Ballesteros et al., 2016; Cross and Huang, 2016b; Fried and Klein, 2018), we decided to consider a simple one that follows a non-optimal transition when it is the highest-scoring one, but with a certain probability. In that way, we easily simulate test time conditions, when the parser greedily chooses the highest-scoring transition, even when it is not an optimal one, placing the parser in an incorrect state.

In particular, we run experiments on development sets for each benchmark/algorithm with three different error exploration probabilities and choose the one that achieves the best F-score. Table 1 reports all results, including those obtained by the top-down and in-order parsers trained by a dynamic oracle without error exploration (equivalent to a traditional static oracle).

4.4 Results

Table 2 compares our system's accuracy to other state-of-the-art shift-reduce constituent parsers on the WSJ and CTB benchmarks. For comparison,

	Top-down		In-order	
Exp.	WSJ	CTB	WSJ	CTB
None	91.81	88.94	91.95	89.69
0.1	91.87	89.13	**92.05**	**89.91**
0.2	**91.99**	88.70	91.98	89.88
0.3	91.97	**89.20**	91.95	89.87

Table 1: F-score comparison of different error-exploration probabilities on WSJ §22 and CTB §301-325 for the top-down a in-order dynamic oracles.

Parser	Type	Strat	F1
(Cross and Huang, 2016a)	gs	bu	90.0
(Cross and Huang, 2016b)	gs	bu	91.0
(Cross and Huang, 2016b)	gd	bu	91.3
(Liu and Zhang, 2017a)	gs	bu	91.3
(Fernández-G and Gómez-R, 2018)	gs	bu	91.5
(Fernández-G and Gómez-R, 2018)	gd	bu	91.7
(Dyer et al., 2016)	gs	td	91.2
This work	gd	td	**91.7**
(Liu and Zhang, 2017a)	gs	in	91.8
This work	gd	in	**92.0**
(Zhu et al., 2013)	b	bu	90.4
(Watanabe and Sumita, 2015)	b	bu	90.7
(Liu and Zhang, 2017b)	b	bu	91.7
(Fried and Klein, 2018)	bp	td	91.6
(Fried and Klein, 2018)	bd	td	92.1
(Fried and Klein, 2018)	bp	in	92.2
(Stern et al., 2017b)	bg	td	92.6
(Stern et al., 2017a)	ch	bu	91.8
(Gaddy et al., 2018)	ch	bu	92.1
(Kitaev and Klein, 2018)	ch	bu	**93.6**

Parser	Type	Strat	F1
(Wang et al., 2015)	gs	bu	83.2
(Liu and Zhang, 2017a)	gs	bu	85.7
(Fernández-G and Gómez-R, 2018)	gs	bu	86.3
(Fernández-G and Gómez-R, 2018)	gd	bu	86.8
(Dyer et al., 2016)	gs	td	84.6
This work	gd	td	**85.3**
(Liu and Zhang, 2017a)	gs	in	86.1
This work	gd	in	**86.6**
(Zhu et al., 2013)	b	bu	83.2
(Watanabe and Sumita, 2015)	b	bu	84.3
(Liu and Zhang, 2017b)	b	bu	85.5
(Fried and Klein, 2018)	bd	td	85.5
(Fried and Klein, 2018)	bp	td	84.7
(Fried and Klein, 2018)	bp	in	**87.0**

Table 2: Accuracy comparison of state-of-the-art single-model fully-supervised constituent parsers on WSJ §23 (top) and CTB §271-300 (bottom). The "Type" column shows the type of parser: *gs* is a greedy parser trained with a static oracle, *gd* a greedy parser trained with a dynamic oracle, *b* a beam search parser, *bp* a beam search parser trained with a policy gradient method, *bd* a beam search parser trained with a non-optimal dynamic oracle, *bg* a generative beam search parser, and *ch* a chart-based parser. Finally, the "Strat" column describes the strategy followed (bu=bottom-up, td=top-down and in=in-order).

Parser	Oracle	#1	#2	#3	#4	#5
Top-down	static	90.98	88.76	85.01	76.63	77.35
	dynamic	**91.34** (+0.36)	**89.18** (+0.42)	**85.17** (+0.16)	**77.12** (+0.49)	**80.02** (+2.67)
In-order	static	91.36	89.21	85.15	77.08	79.02
	dynamic	**91.55** (+0.19)	**89.43** (+0.22)	**85.34** (+0.19)	**77.57** (+0.49)	**81.03** (+2.01)

Table 3: F-score on constituents with a number of children ranging from one to five on WSJ §23.

we also include some recent state-of-the-art parsers with global chart decoding that achieve the highest accuracies to date on WSJ, but are much slower than shift-reduce algorithms.

Top-down and in-order parsers benefit from being trained by these new dynamic oracles in both datasets. The top-down strategy achieves a gain of 0.5 and 0.7 points in F-score on WSJ and CTB benchmarks, respectively. The in-order parser obtains similar improvements on the CTB (0.5 points), but less notable accuracy gain on the WSJ (0.2 points). Although a case of diminishing returns might explain the latter, the in-order parser trained with the proposed dynamic oracle still achieves the highest accuracy to date in greedy transition-based constituent parsing on the WSJ.[7]

While this work was under review, Fried and Klein (2018) proposed to train the top-down and in-order parsers with a policy gradient method instead of custom designed dynamic oracles. They also present a non-optimal dynamic oracle for the top-down parser that, combined with more complex error-exploration strategies and size-10 beam search, significantly outperforms the policy gradient-trained version, confirming that even non-optimal dynamic oracles are a good option.[8]

4.5 Analysis

Dan Bikel's randomized parsing evaluation comparator (Bikel, 2004) was used to perform significance tests on precision and recall metrics on WSJ §23 and CTB §271-300. The top-down parser trained with dynamic oracles achieves statistically significant improvements ($p < 0.05$) in precision both on the WSJ and CTB benchmarks, and in recall on WSJ. The in-order parser trained with the proposed technique obtains significant improvements ($p < 0.05$) in recall in both benchmarks, although not in precision.

We also undertake an analysis to check if dynamic oracles are able to mitigate error propagation. We report in Table 3 the F-score obtained in constituents with different number of children on WSJ §23 by the top-down and in-order algorithms trained with both static and dynamic oracles. Please note that creating a constituent with a great number of children is more prone to suffer from error propagation, since a larger number of transitions is required to build it. The results seem to confirm that, indeed, dynamic oracles manage to alleviate error propagation, since improvements in F-score are more notable for larger constituents.

5 Conclusion

We develop the first optimal dynamic oracles for training the top-down and the state-of-the-art in-order parsers. Apart from improving the systems' accuracies in both cases, we achieve the best result to date in greedy shift-reduce parsing on the WSJ. In addition, these promising techniques could easily benefit from recent studies in error-exploration strategies and yield state-of-the-art accuracies in transition-based parsing in the near future. The parser's source code is freely available at `https://github.com/danifg/Dynamic-InOrderParser`.

Acknowledgments

This work has received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from MINECO (FFI2014-51978-C2-2-R, TIN2017-85160-C2-1-R) and from Xunta de Galicia (ED431B 2017/01).

---

[7] Note that the proposed dynamic oracles are orthogonal to approaches like beam search, re-ranking or semi-supervision, that can boost accuracy but at a large cost to parsing speed.

[8] Unfortunately, we cannot directly compare our approach to theirs, since they use beam-search decoding with size 10 in all experiments, gaining up to 0.3 points in F-score, while penalizing speed with respect to greedy decoding. However, by extrapolating the results above, we hypothesize that our optimal dynamic oracles (especially the one designed for the in-order algorithm) with their same training and beam-search decoding setup might achieve the best scores to date in shift-reduce parsing.

References

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack LSTM parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2005–2010.

Dan Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.

Maximin Coavoux and Benoit Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 172–182, Berlin, Germany. Association for Computational Linguistics.

Benoit Crabbé. 2015. Multilingual discriminative lexicalized phrase structure parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1847–1856, Lisbon, Portugal. Association for Computational Linguistics.

James Cross and Liang Huang. 2016a. Incremental parsing with minimal features using bi-directional LSTM. In *ACL (2)*. The Association for Computer Linguistics.

James Cross and Liang Huang. 2016b. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *EMNLP*, pages 1–11. The Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *HLT-NAACL*, pages 199–209. The Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018. Faster shift-reduce constituent parsing with a non-binary, bottom-up strategy. *arXiv*, 1804.07961 [cs.CL].

Michael L. Fredman. 1975. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29 – 35.

Daniel Fried and Dan Klein. 2018. Policy gradient as a proxy for dynamic oracles in constituency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 469–476.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 999–1010.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2675–2685.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *EACL (1)*, pages 1249–1258. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017a. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Jiangming Liu and Yue Zhang. 2017b. Shift-reduce constituent parsing with neural lookahead features. *TACL*, 5:45–58.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Haitao Mi and Liang Huang. 2015. Shift-reduce constituency parsing with dynamic programming and pos tag lattice. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1030–1035, Denver, Colorado. Association for Computational Linguistics.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pages 125–132.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017a. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 818–827.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017b. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1695–1700.

Zhiguo Wang, Haitao Mi, and Nianwen Xue. 2015. Feature optimization for constituent parsing via neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1138–1147.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, ACL 2015, 26-31 July 2015, Bejing, China, Volume 1: Long Papers*, pages 1169–1179.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, pages 162–171, Stroudsburg, PA, USA. Association for Computational Linguistics.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 434–443.

Constituent Parsing as Sequence Labeling

Carlos Gómez-Rodríguez
Universidade da Coruña
FASTPARSE Lab, LyS Group
Departamento de Computación
Campus de Elviña s/n, 15071
A Coruña, Spain
`carlos.gomez@udc.es`

David Vilares
Universidade da Coruña
FASTPARSE Lab, LyS Group
Departamento de Computación
Campus de Elviña s/n, 15071
A Coruña, Spain
`david.vilares@udc.es`

Abstract

We introduce a method to reduce constituent parsing to sequence labeling. For each word w_t, it generates a label that encodes: (1) the number of ancestors in the tree that the words w_t and w_{t+1} have in common, and (2) the nonterminal symbol at the lowest common ancestor. We first prove that the proposed encoding function is injective for any tree without unary branches. In practice, the approach is made extensible to all constituency trees by collapsing unary branches. We then use the PTB and CTB treebanks as testbeds and propose a set of fast baselines. We achieve 90% F-score on the PTB test set, outperforming the Vinyals et al. (2015) sequence-to-sequence parser. In addition, sacrificing some accuracy, our approach achieves the fastest constituent parsing speeds reported to date on PTB by a wide margin.

1 Introduction

Constituent parsing is a core problem in NLP where the goal is to obtain the syntactic structure of sentences expressed as a phrase structure tree.

Traditionally, constituent-based parsers have been built relying on chart-based, statistical models (Collins, 1997; Charniak, 2000; Petrov et al., 2006), which are accurate but slow, with typical speeds well below 10 sentences per second on modern CPUs (Kummerfeld et al., 2012).

Several authors have proposed more efficient approaches which are helpful to gain speed while preserving (or even improving) accuracy. Sagae and Lavie (2005) present a classifier for constituency parsing that runs in linear time by relying on a shift-reduce stack-based algorithm, instead of a grammar. It is essentially an extension of transition-based dependency parsing (Nivre, 2003). This line of research has been polished through the years (Wang et al., 2006; Zhu et al., 2013; Dyer et al., 2016; Liu and Zhang, 2017; Fernández-González and Gómez-Rodríguez, 2018).

With an aim more related to our work, other authors have reduced constituency parsing to tasks that can be solved faster or in a more generic way. Fernández-González and Martins (2015) reduce phrase structure parsing to dependency parsing. They propose an intermediate representation where dependency labels from a head to its dependents encode the nonterminal symbol and an attachment order that is used to arrange nodes into constituents. Their approach makes it possible to use off-the-shelf dependency parsers for constituency parsing. In a different line, Vinyals et al. (2015) address the problem by relying on a sequence-to-sequence model where trees are linearized in a depth-first traversal order. Their solution can be seen as a machine translation model that maps a sequence of words into a parenthesized version of the tree. Choe and Charniak (2016) recast parsing as language modeling. They train a generative parser that obtains the phrasal structure of sentences by relying on the Vinyals et al. (2015) intuition and on the Zaremba et al. (2014) model to build the basic language modeling architecture.

More recently, Shen et al. (2018) propose an architecture to speed up the current state-of-the-art chart parsers trained with deep neural networks (Stern et al., 2017; Kitaev and Klein, 2018). They introduce the concept of *syntactic distances*, which specify the order in which the splitting points of a sentence will be selected. The model learns to predict such distances, to then recursively partition the input in a top-down fashion.

Contribution We propose a method to transform constituent parsing into sequence labeling. This reduces it to the complexity of tasks such as part-of-speech (PoS) tagging, chunking or named-entity recognition. The contribution is two-fold.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1314–1324
Brussels, Belgium, October 31 - November 4, 2018.

First, we describe a method to linearize a tree into a sequence of labels (§2) of the same length of the sentence minus one.[1] The label generated for each word encodes the number of common ancestors in the constituent tree between that word and the next, and the nonterminal symbol associated with the lowest common ancestor. We prove that the encoding function is injective for any tree without unary branchings. After applying collapsing techniques, the method can parse unary chains.

Second, we use such encoding to present different baselines that can effectively predict the structure of sentences (§3). To do so, we rely on a recurrent sequence labeling model based on BILSTM's (Hochreiter and Schmidhuber, 1997; Yang and Zhang, 2018). We also test other models inspired in classic approaches for other tagging tasks (Schmid, 1994; Sha and Pereira, 2003). We use the Penn Treebank (PTB) and the Penn Chinese Treebank (CTB) as testbeds.

The comparison against Vinyals et al. (2015), the closest work to ours, shows that our method is able to train more accurate parsers. This is in spite of the fact that our approach addresses constituent parsing as a sequence labeling problem, which is simpler than a sequence-to-sequence problem, where the output sequence has variable/unknown length. Despite being the first sequence labeling method for constituent parsing, our baselines achieve decent accuracy results in comparison to models coming from mature lines of research, and their speeds are the fastest reported to our knowledge.

2 Linearization of *n*-ary trees

Notation and Preliminaries In what follows, we use bold style to refer to vectors and matrices (e.g $\mathbf{x}$ and $\mathbf{W}$). Let $\mathbf{w}$=$[w_1, w_2, ..., w_{|w|}]$ be an input sequence of words, where $w_i \in V$. Let $T_{|w|}$ be the set of constituent trees with $|w|$ leaf nodes that have no unary branches. For now, we will assume that the constituent parsing problem consists in mapping each sentence $\mathbf{w}$ to a tree in $T_{|w|}$, i.e., we assume that correct parses have no unary branches. We will deal with unary branches later.

To reduce the problem to a sequence labeling task, we define a set of labels L that allows us to encode each tree in $T_{|w|}$ as a unique sequence of labels in $L^{(|w|-1)}$, via an encoding function $\Phi_{|w|} : T_{|w|} \rightarrow L^{(|w|-1)}$. Then, we can reduce the constituent parsing problem to a sequence labeling task where the goal is to predict a function $F_{|w|,\theta} : V^{|w|} \rightarrow L^{|w|-1}$, where θ are the parameters to be learned. To parse a sentence, we label it and then decode the resulting label sequence into a constituent tree, i.e., we apply $F_{|w|,\theta} \circ \Phi^{-1}_{|w|}$.

For the method to be correct, we need the encoding of trees to be complete (every tree in $T_{|w|}$ must be expressible as a label sequence, i.e., $\Phi_{|w|}$ must be a function, so we have full coverage of constituent trees) and injective (so that the inverse function $\Phi^{-1}_{|w|}$ is well-defined). Surjectivity is also desirable, so that the inverse is a function on $L^{|w|-1}$, and the parser outputs a tree for any sequence of labels that the classifier can generate.

We now define our $\Phi_{|w|}$ and show that it is total and injective. Our encoding is not surjective per se. We handle ill-formed label sequences in §2.3.

2.1 The Encoding

Let w_i be a word located at position i in the sentence, for $1 \leq i \leq |w| - 1$. We will assign it a 2-tuple label $l_i = (n_i, c_i)$, where: n_i is an integer that encodes the number of common ancestors between w_i and w_{i+1}, and c_i is the nonterminal symbol at the lowest common ancestor.

Basic encodings The number of common ancestors may be encoded in several ways.

1. Absolute scale: The simplest encoding is to make n_i directly equal to the number of ancestors in common between w_i and w_{i+1}.
2. Relative scale: A second and better variant consists in making n_i represent the difference with respect to the number of ancestors encoded in n_{i-1}. Its main advantage is that the size of the label set is reduced considerably.

Figure 1 shows an example of a tree linearized according to both absolute and relative scales.

Encoding for trees with exactly k children For trees where all branchings have exactly k children, it is possible to obtain a even more efficient linearization in terms of number of labels. To do so, we take the relative scale encoding as our starting point. If we build the tree incrementally in a left-to-right manner from the labels, if we find a negative n_i, we will need to attach the word w_{i+1} (or a new subtree with that word as its leftmost leaf) to the $(-n_i + 2)$th node in the path going from

---

[1] A last dummy label is generated to fulfill the properties of sequence labeling tasks.

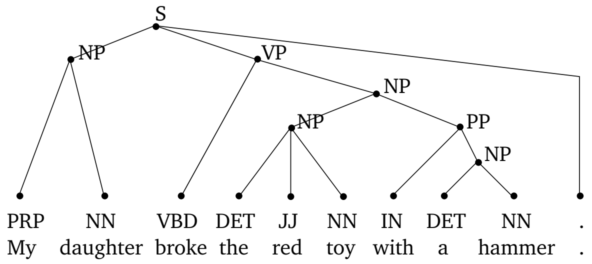

Linearized tree (absolute scale):

2NP 1S 2VP 4NP 4NP 3NP 4PP 5NP 1S

Linearized tree (relative scale):

2NP -1S 1VP 2NP 0NP -1NP 1PP 1NP -4S

Figure 1: An example of a constituency tree linearized applying both absolute and relative scales.

w_i to the root. If every node must have exactly k children, there is only one valid negative value of n_i: the one pointing to the first node in said path that has not received its kth child yet. Any smaller value would leave this node without enough children (which cannot be fixed later due to the left-to-right order in which we build the tree), and any larger value would create a node with too many children. Thus, we can map negative values to a single label. Figure 2 shows an example for the case of binarized trees ($k = 2$).

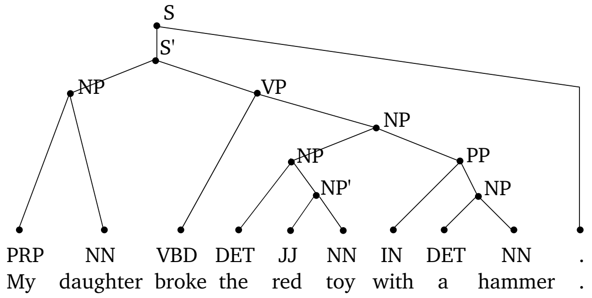

Linearized tree (absolute scale):

3NP 2S' 3VP 5NP 6NP' 4NP 5PP 6NP 1S

Linearized tree (relative scale):

3NP -1S' 1VP 2NP 1NP' -2NP 1PP 1NP -5S

Linearized tree (simplified relative scale):

3NP -S' 1VP 2NP 1NP' -NP 1PP 1NP -S

Figure 2: An example of a binarized constituency tree, linearized both applying absolute and relative scales.

Links to root Another variant emerged from the empirical observation that some tokens that are usually linked to the root node (such as the final punctuation in Figure 1) were particularly difficult to learn for the simpler baselines. To successfully deal with these cases in practice, it makes sense to consider a simplified annotation scheme where a node is assigned a special tag (ROOT, c_i) when it is directly linked to the root of the tree.

From now on, unless otherwise specified, we use the relative scale without the simplification for exactly k children. This will be the encoding used in the experiments (§4), because the size of the label set is significantly lower than the one obtained by relying on the absolute one. Also, it works directly with non-binarized trees, in contrast to the encoding that we introduce for trees with exactly k children, which is described only for completeness and possible interest for future work. For the experiments (§4), we also use the special tag (ROOT, c_i) to further reduce the size of the label set and to simplify the classification of tokens connected to the root, where $|n_i|$ is expected to be large.

2.2 Theoretical correctness

We now prove that $\Phi_{|w|}$ is a total function and injective for any tree in $T_{|w|}$. We remind that trees in this set have no unary branches. Later (in §2.3) we describe how we deal with unary branches. To prove correctness, we use the relative scale. Correctness for the other scales follows trivially.

Completeness Every pair of nodes in a rooted tree has at least one common ancestor, and a unique lowest common ancestor. Hence, for any tree in $T_{|w|}$, the label $l_i = (n_i, c_i)$ defined in Section 2.1 is well-defined and unique for each word w_i, $1 \leq i \leq |w| - 1$; and thus $\Phi_{|w|}$ is a total function from $T_{|w|}$ to $L^{(|w|-1)}$.

Injectivity The encoding method must ensure that any given sequence of labels corresponds to exactly one tree. Otherwise, we have to deal with ambiguity, which is not desirable.

For simplicity, we will prove injectivity in two steps. First, we will show that the encoding is injective if we ignore nonterminals (i.e., equivalently, that the encoding is injective for the set of trees resulting from replacing all the nonterminals in trees in $T_{|w|}$ with a generic nonterminal X). Then, we will show that it remains injective when we take nonterminals into account.

For the first part, let $\tau \in T_{|w|}$ be a tree where nonterminals take a generic value X. We represent the label of the ith leaf node as $\bullet_i$. Consider the representation of τ as a bracketed string, where a single-node tree with a node labeled A is represented by (A), and a tree rooted at R

with child subtrees $C_1 \dots C_n$ is represented as $(R(C_1 \dots C_n))$.

Each leaf node will appear in this string as a substring $(\bullet_i)$. Thus, the parenthesized string has the form $\alpha_0(\bullet_1)\alpha_1(\bullet_2)\dots\alpha_{|w|-1}(\bullet_{|w|})\alpha_w$, where the α_is are strings that can only contain brackets and nonterminals, as by construction there can be no leaf nodes between $(\bullet_i)$ and $(\bullet_{i+1})$.

We now observe some properties of this parenthesized string. First, note that each of the substrings α_i must necessarily be composed of zero or more closing parentheses followed by zero or more opening parentheses with their corresponding nonterminal, i.e., it must be of the form $[)]^*[(X]^*$. This is because an opening parenthesis followed by a closing parenthesis would represent a leaf node, and there are no leaf nodes between $(\bullet_i)$ and $(\bullet_{i+1})$ in the tree.

Thus, we can write α_i as $\alpha_{i)}\alpha_{i(}$, where $\alpha_{i)}$ is a string matching the expression $[)]^*$ and $\alpha_{i(}$ a string matching the expression $[(X]^*$. With this, we can write the parenthesized string for τ as

$$\alpha_{0)}\alpha_{0(}(\bullet_1)\alpha_{1)}\alpha_{1(}(\bullet_2)\alpha_{2)}\alpha_{2(}\dots(\bullet_{|w|})\alpha_{|w|)}\alpha_{|w|(}.$$

Let us now denote by β_i the string $\alpha_{i-1(}(\bullet_i)\alpha_{i)}$. Then, and taking into account that $\alpha_{0)}$ and $\alpha_{w(}$ are trivially empty in the previous expression due to bracket balancing, the expression for the tree becomes simply $\beta_1\beta_2\dots\beta_{|w|}$, where we know, by construction, that each β_i is of the form $[(X]^*(\bullet_i)[)]^*$.

Since we have shown that each tree in $T_{|w|}$ uniquely corresponds to a string $\beta_1\beta_2\dots\beta_{|w|}$, to show injectivity of the encoding, it suffices to show that different values for a β_i generate different label sequences.

To show this, we can say more about the form of β_i: it must be either of the form $[(X]^*(\bullet_i)$ or of the form $(\bullet_i)[)]^*$, i.e., it is not possible that β_i contains both opening parenthesis before the leaf node and closing parentheses after the leaf node. This could only happen if the tree had a subtree of the form $(X(\bullet_i))$, but this is not possible since we are forbidding unary branches.

Hence, we can identify each β_i with an integer number $\delta(\beta_i)$: 0 if β_i has neither opening nor closing parentheses outside the leaf node, $+k$ if it has k opening parentheses, and $-k$ if it has k closing parentheses. It is easy to see that $\delta(\beta_1)\delta(\beta_2)\dots\delta(\beta_{|w|-1})$ corresponds to the values n_i in the relative-scale label encoding of the tree τ. To see this, note that the number of unclosed parentheses at the point right after β_i in the string exactly corresponds to the number of common ancestors between the ith and $(i+1)$th leaf nodes. A positive $\delta(\beta_i) = k$ corresponds to opening k parentheses before β_i, so the number of common ancestors of w_i and w_{i+1} will be k more than that of w_{i-1} and w_i. A negative $\delta(\beta_i) = -k$ corresponds to closing k parentheses after β_i, so the number of common ancestors will conversely decrease by k. A value of zero means no opening or closing parentheses, and no change in the number of common ancestors.

Thus, different parenthesized strings $\beta_1\beta_2\dots\beta_{|w|}$ generate different label sequences, which proves injectivity ignoring nonterminals (note that $\delta(\beta_{|w|})$ does not affect injectivity as it is uniquely determined by the other values: it corresponds to closing all the parentheses that remain unclosed at that point).

It remains to show that injectivity still holds when nonterminals are taken into account. Since we have already proven that trees with different structure produce different values of n_i in the labels, it suffices to show that trees with the same structure, but different nonterminals, produce different values of c_i. Essentially, this reduces to showing that every nonterminal in the tree is mapped into a concrete c_i. That said, consider a tree $\tau \in T_{|w|}$, and some nonterminal X in τ. Since trees in T_w do not have unary branches, X has at least two children. Consider the rightmost word in the first child subtree, and call it w_i. Then, w_{i+1} is the leftmost word in the second child subtree, and X is the lowest common ancestor of w_i and w_{i+1}. Thus, $c_i = X$, and a tree with identical structure but a different nonterminal at that position will generate a label sequence with a different value of c_i. This concludes the proof of injectivity.

2.3 Limitations

We have shown that our proposed encoding is a total, injective function from trees without unary branches with yield of length $|w|$ to sequences of $|w| - 1$ labels. This will serve as the basis for our reduction of constituent parsing to sequence labeling. However, to go from theory to practice, we need to overcome two limitations of the theoretical encoding: non-surjectivity and the inability to encode unary branches. Fortunately, both can be overcome with simple techniques.

Handling of unary branches The encoding function $\Phi_{|w|}$ cannot directly assign the nonterminal symbols of unary branches, as there is not any pair of words (w_i, w_{i+1}) that have those in common. Figure 3 illustrates it with an example.

It is worth remarking that this is not a limitation of our encoding, but of any encoding that would facilitate constituent parsing as sequence labeling, as the number of nonterminal nodes in a tree with unary branches is not bounded by any function of $|w|$. The fact that our encoding works for trees without unary branches owes to the fact that such a tree cannot have more than $|w|-1$ non-leaf nodes, and therefore it is always possible to encode all of them in labels associated with $|w|-1$ leaf nodes.

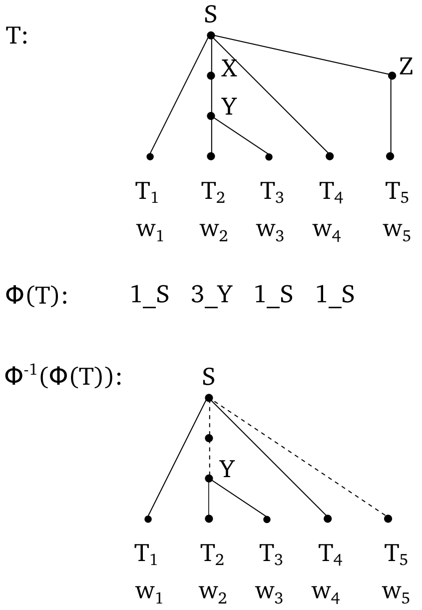

Figure 3: An example of a tree that cannot be directly linearized with our approach. w_i and T_i abstract over words and PoS tags. Dotted lines represent incorrect branches after applying and inverting our encoding naively without any adaptation for unaries. The nonterminal symbol of the second ancestor of w_2 (X) cannot be decoded, as no pair of words have X as their lowest common ancestor. A similar situation can be observed for the closest ancestor of w_5 (Z).

To overcome this issue, we follow a collapsing approach, as is common in parsers that need special treatment of unary chains (Finkel et al., 2008; Narayan and Cohen, 2016; Shen et al., 2018). For clarity, we use the name *intermediate unary chains* to refer to unary chains that end up into a nonterminal symbol (e.g. X → Y in Figure 3) and *leaf unary chains* to name those that yield a PoS tag (e.g. Z → T_5). Intermediate unary chains are collapsed into a chained single symbol, which can be encoded by $\Phi_{|w|}$ as any other nonterminal symbol. On the other hand, leaf unary chains are collapsed together with the PoS tag, but these cannot be encoded and decoded by relying on $\Phi_{|w|}$, as our encoding assumes a fixed sequence of leaf nodes and does not encode them explicitly. To overcome this, we propose two methods:

1. To use an extra function to enrich the PoS tags before applying our main sequence labeling function. This function is of the form $\Psi_{|w|} : V^{|w|} \rightarrow U^{|w|}$, where U is the set of labels of the leaf unary chains (without including the PoS tags) plus a dummy label $\varnothing$. $\Psi_{|w|}$ maps w_i to $\varnothing$ if there is no leaf unary chain at w_i, or to the collapsed label otherwise.

2. To extend our encoding function to predict them as a part of our labels l_i, by transforming them into 3-tuples (n_i, c_i, u_i) where u_i encodes the leaf unary chain collapsed label for w_i, if there is any, or none otherwise. We call this extended encoding function $\Phi'_{|w|}$.

The former requires to run two passes of sequence labeling to deal with leaf unary chains. The latter avoids this, but the number of labels is larger and sparser. In §4 we discuss how these two approaches behave in terms of accuracy and speed.

Non-surjectivity Our encoding, as defined formally in Section 2.1, is injective but not surjective, i.e., not every sequence of $|w|-1$ labels of the form (n_i, c_i) corresponds to a tree in $T_{|w|}$. In particular, there are two situations where a label sequence formally has no tree, and thus $\Phi^{-1}_{|w|}$ is not formally defined and we have to use extra heuristics or processing to define it:

- Sequences with conflicting nonterminals. A nonterminal can be the lowest common ancestor of more than two pairs of contiguous words when branches are non-binary. For example, in the tree in Figure 1, the lowest common ancestor of both “the” and “red” and of “red” and “toy” is the same NP node. This translates into $c_4 = NP$, $c_5 = NP$ in the label sequence. If we take that sequence and set $c_5 = VP$, we obtain a label sequence that does not strictly correspond to the encoding of any tree, as it contains a contradiction: two elements referencing the same node indicate different nonterminal labels. In practice, this problem is trivial to solve: when a label sequence encodes several conflicting nonterminals at a given position in the tree, we compute $\Phi_{|w|}^{-1}$ using the first such nonterminal and ignoring the rest.

- Sequences that produce unary structures. There are sequences of values n_i that do not correspond to a tree in $T_{|w|}$ because the only tree structure satisfying the common ancestor conditions of their values (the one built by generating the string of β_is in the injectivity proof) contains unary branchings, causing the problem described above where we do not have a specification for every nonterminal. An example of this is the sequence $(1, S), (3, Y), (1, S), (1, S)$ in absolute scaling, that was introduced in Figure 3. In practice, as unary chains have been previously collapsed, any generated unary node is considered as not valid and removed.

3 Sequence Labeling

Sequence labeling is an structured prediction task that generates an output label for every token in an input sequence (Rei and Søgaard, 2018). Examples of practical tasks that can be formulated under this framework in natural language processing are PoS tagging, chunking or named-entity recognition, which are in general fast. However, to our knowledge, there is no previous work on sequence labeling methods for constituent parsing, as an encoding allowing it was lacking so far.

In this work, we consider a range of methods ranging from traditional models to state-of-the-art neural models for sequence labeling, to test whether they are valid to train constituency-based parsers following our approach. We give the essential details needed to comprehend the core of each approach, but will mainly treat them as black boxes, referring the reader to the references for a careful and detailed mathematical analysis of each method. Appendix A specifies additional hyperparameters for the tested models.

Preprocessing We add to every sentence both beginning and end tokens.

3.1 Traditional Sequence Labeling Methods

We consider two baselines to train our prediction function $F_{|w|,\theta}$, based on popular sequence labeling methods used in NLP problems, such as PoS tagging or shallow parsing (Schmid, 1994; Sha and Pereira, 2003).

Conditional Random Fields (Lafferty et al., 2001) Let $\text{CRF}_{|w|,\theta}$ be its prediction function, a CRF model computes conditional probability distributions of the form $p(\mathbf{l}, \mathbf{w})$ such that $\text{CRF}_\theta(\mathbf{w}) = \mathbf{l} = \arg\max_{\mathbf{l}'} p(\mathbf{l}', \mathbf{w})$. In our work, the inputs to the CRF are words and PoS tags. To represent a word w_i, we are using information of the word itself and also contextual information from $\mathbf{w}_{[i-1:i+1]}$.[2] In particular:

- We extract the word form (lowercased), the PoS tag and its prefix of length 2, from $\mathbf{w}_{[i-1:i+1]}$. For these words we also include binary features: whether it is the first word, the last word, a number, whether the word is capitalized or uppercased.
- Additionally, for w_i we look at the suffixes of both length 3 and 2 (i.e. $w_{i[-3:]}$ and $w_{i[-2:]}$).

To build our CRF models, we relied on the `sklearn-crfsuite` library[3].

MultiLayer Perceptron (Rosenblatt, 1958) We use one hidden layer. Let $\text{MLP}_{|w|,\theta}$ be its prediction function, it treats sequence labeling as a set of independent predictions, one per word. The prediction for a word is computed as $softmax(\mathbf{W_2} \cdot relu(\mathbf{W_1} \cdot \mathbf{x} + \mathbf{b_1}) + \mathbf{b_2})$, where $\mathbf{x}$ is the input vector and W_i and b_i the weights and biases to be learned at layer i. We consider both a discrete (MLP_d) and an embedded (MLP_e) perceptron. For the former, we use as inputs the same set of features as for the CRF. For the latter, the vector $\mathbf{x}$ for w_i is defined as a concatenation of word and PoS tag embeddings from $\mathbf{w}_{[i-2:i+2]}$.[4]

To build our MLPs, we relied on `keras`.[5]

3.2 Sequence Labeling Neural Models

We are using NCRFPP++[6], a sequence labeling framework based on recurrent neural networks (RNN) (Yang and Zhang, 2018), and more specifically on bidirectional short-term memory networks (Hochreiter and Schmidhuber, 1997),

---

[2] We tried contextual information beyond the immediate previous and next word, but the performance was similar.

[3] https://sklearn-crfsuite.readthedocs.io/en/latest/

[4] In contrast to the discrete input, larger contextual information was useful.

[5] https://keras.io/

[6] https://github.com/jiesutd/NCRFpp, with PyTorch.

which have been successfully applied to problems such as PoS tagging or dependency parsing (Plank et al., 2016; Kiperwasser and Goldberg, 2016). Let $\text{LSTM}(\mathbf{x})$ be an abstraction of a standard long short-term memory network that processes the sequence $\mathbf{x} = [\mathbf{x}_1, ..., \mathbf{x}_{|\mathbf{x}|}]$, then a BILSTM encoding of its ith element, $\text{BILSTM}(\mathbf{x}, i)$ is defined as:

$$\text{BILSTM}(\mathbf{x}, i) = \mathbf{h}_i = \mathbf{h}_i^l \circ \mathbf{h}_i^r = \\ \text{LSTM}^l(\mathbf{x}_{[1:i]}) \circ \text{LSTM}^r(\mathbf{x}_{[|\mathbf{x}|:i]})$$

In the case of multilayer BILSTM'S, the timestep outputs of the BILSTM_m are fed as input to the BILSTM_{m+1}. The output label for each w_i is finally predicted as $softmax(W \cdot \mathbf{h}_i + b)$.

Given a sentence $[w_1, w_2, ..., w_{|w|}]$, the input to the sequence model is a sequence of embeddings $[\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w}_{|\mathbf{w}|}]$ where each $\mathbf{w_i} = \mathbf{w}_i \circ \mathbf{p}_i \circ \mathbf{ch}_i$, such that $\mathbf{w}_i$ and $\mathbf{p}_i$ are a word and a PoS tag embedding, and $\mathbf{ch}_i$ is a word embedding obtained from an initial character embedding layer, also based on a BILSTM. Figure 4 shows the architecture of the network.

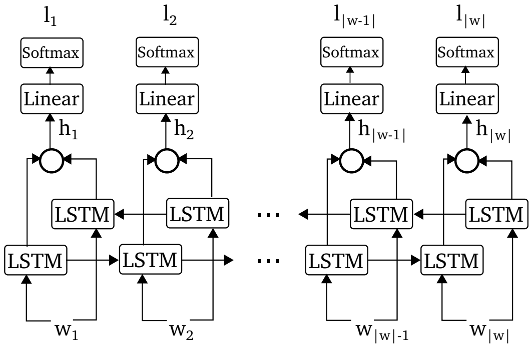

Figure 4: Architecture of the neural model

4 Experiments

We report results on models trained using the relative scale encoding and the special tag (ROOT,c_i). As a reminder, to deal also with leaf unary chains, we proposed two methods in §2.3: to predict them relying both on the encoding functions $\Phi_{|w|}$ and $\Psi_{|w|}$, or to predict them as a part of an enriched label predicted by the function $\Phi'_{|w|}$. For clarity, we are naming these models with the superscripts $^{\Psi,\Phi}$ and $^{\Phi'}$, respectively.

Datasets We use the Penn Treebank (Marcus et al., 1994) and its official splits: Sections 2 to 21 for training, 22 for development and 23 for testing. For the Chinese Penn Treebank (Xue et al., 2005): articles 001- 270 and 440-1151 are used for training, articles 301-325 for development, and articles 271-300 for testing. We use the version of the corpus with the predicted PoS tags of Dyer et al. (2016). We train the Φ models based on the predicted output by the corresponding Ψ model.

Metrics We use the F-score from the EVALB script. Speed is measured in sentences per second. As the problem is reduced to sequence labeling, we briefly comment on the accuracy (percentage of correctly predicted labels) of our baselines.

Source code It can be found at `https://github.com/aghie/tree2labels`

Hardware The models are run on a single thread of a CPU[7] and on a consumer-grade GPU[8]. In sequence-to-sequence work (Vinyals et al., 2015) the authors use a multi-core CPU (the number of threads was not specified), while we provide results on a single core for easier comparability. Parsing sentences on a CPU can be framed as an "embarrassingly parallel" problem (Hall et al., 2014), so speed can be made to scale linearly with the number of cores. We use the same batch size as Vinyals et al. (2015) for testing (128).[9]

4.1 Results

Table 1 shows the performance of our baselines on the PTB development set. It is worth noting that since we are using different libraries to train the models, these might show some differences in terms of performance/speed beyond those expected in theory. For the BILSTM model we test:

- $\text{BILSTM}_{m=1}$: It does not use pretrained word embeddings nor character embeddings. The number of layers m is set to 1.
- $\text{BILSTM}_{m=1,e}$: It adds pretrained word embeddings from GloVe (Pennington et al., 2014) for English and from the Gigaword corpus for Chinese (Liu and Zhang, 2017).
- $\text{BILSTM}_{m=1,e,ch}$: It includes character embeddings processed through a BILSTM.
- $\text{BILSTM}_{m=2,e}$: m is set to 2. No character embeddings.
- $\text{BILSTM}_{m=2,e,ch}$: m is set to 2.

---

[7] An Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz.

[8] A GeForce GTX 1080.

[9] A larger batch will likely result in faster parsing when executing the model on a GPU, but not necessarily on a CPU.

Model	**F-score**	**Acc.**	**Sent/s (CPU)**	**Sent/s (GPU)**
$\text{CRF}^{\Psi,\Phi}$	60.4	63.9	83	-
$\text{MLP}_d^{\Psi,\Phi}$	72.6	78.1	16	49
$\text{MLP}_e^{\Psi,\Phi}$	74.8	79.3	503	666
$\text{CRF}^{\Phi'}$	60.3	65.4	6	-
$\text{MLP}_d^{\Phi'}$	71.9	78.0	31	95
$\text{MLP}_e^{\Phi'}$	75.4	79.7	342	890
$\text{BILSTM}_{m=1}^{\Psi,\Phi}$	87.2	88.9	144	541
$\text{BILSTM}_{m=1,e}^{\Psi,\Phi}$	88.3	89.8	144	543
$\text{BILSTM}_{m=1,e,ch}^{\Psi,\Phi}$	88.5	90.0	120	456
$\text{BILSTM}_{m=2,e}^{\Psi,\Phi}$	89.7	90.7	72	476
$\text{BILSTM}_{m=2,e,ch}^{\Psi,\Phi}$	89.9	90.9	65	405
$\text{BILSTM}_{m=1}^{\Phi'}$	87.3	89.3	206	941
$\text{BILSTM}_{m=1,e}^{\Phi'}$	88.5	90.1	209	957
$\text{BILSTM}_{m=1,e,ch}^{\Phi'}$	88.0	90.0	180	808
$\text{BILSTM}_{m=2,e}^{\Phi'}$	89.8	90.9	119	842
$\text{BILSTM}_{m=2,e,ch}^{\Phi'}$	89.7	90.9	109	716

Table 1: Performance of the proposed sequence labeling methods on the development set of the PTB. For the CRF models the complexity is quadratic with respect to the number of labels, which causes $\text{CRF}^{\Phi'}$ to be particularly slow.

The Ψ, Φ and the Φ' models obtain similar F-scores. When it comes to speed, the $\text{BILSTMs}^{\Phi'}$ are notably faster than the $\text{BILSTMs}^{\Psi,\Phi}$. Φ' models are expected to be more efficient, as leaf unary chains are handled implicitly. In practice, Φ' is a more expensive function to compute than the original Φ, since the number of output labels is significantly larger, which reduces the expected gains with respect to the Ψ, Φ models. It is worth noting that our encoding is useful to train an MLP_e with a decent sense of phrase structure, while being very fast. Paying attention to the differences between F-score and Accuracy for each baseline, we notice the gap between them is larger for CRFs and MLPs. This shows the difficulties that these methods have, in comparison to the BILSTM approaches, to predict the correct label when a word w_{i+1} has few common ancestors with w_i. For example, let -10X be the right (relative scale) label between w_i and w_{i+1}, and let l_1=-1X and l_2=-9X be two possible wrong labels. In terms of accuracy it is the same that a model predicts $l1$ or $l2$, but in terms of constituent F-score, the first will be much worse, as many closed parentheses will remain unmatched.

Tables 2 and 3 compare our best models against the state of the art on the PTB and CTB test sets. The performance corresponds to models without reranking strategies, unless otherwise specified.

5 Discussion

We are not aware of work that reduces constituency parsing to sequence labeling. The work that can be considered as the closest to ours is that of Vinyals et al. (2015), who address it as a sequence-to-sequence problem, where the output sequence has variable/unknown length. In this context, even a one hidden layer perceptron outperforms their 3-layer LSTM model without attention, while parsing hundreds of sentences per second. Our best models also outperformed their 3-layer LSTM model with attention and even a simple BILSTM model with pretrained GloVe embeddings obtains a similar performance. In terms of F-score, the proposed sequence labeling baselines still lag behind mature shift-reduce and chart parsers. In terms of speed, they are clearly faster than both CPU and GPU chart parsers and are at least on par with the fastest shift-reduce ones. Although with significant loss of accuracy, if phrase-representation is needed in large-scale tasks where the speed of current systems makes parsing infeasible (Gómez-Rodríguez, 2017; Gómez-Rodríguez et al., 2017), we can use the simpler, less accurate models to get speeds well above any parser reported to date.

It is also worth noting that in their recent work, published while this manuscript was under review, Shen et al. (2018) developed a mapping of binary trees with n leaves to sequences of $n-1$ integers (Shen et al., 2018, Algorithm 1). This encoding is different from the ones presented here, as it is based on the height of lowest common ancestors in the tree, rather than their depth. While their purpose is also different from ours, as they use this mapping to generate training data for a parsing algorithm based on recursive partitioning using real-valued distances, their encoding could also be applied with our sequence labeling approach. However, it has the drawback that it only supports binarized trees, and some of its theoretical properties are worse for our goal, as the way to define the inverse of an arbitrary label sequence can be highly ambiguous: for example, a sequence of $n-1$ equal labels in this encoding can represent *any* binary tree with n leaves.

6 Conclusion

We presented a new parsing paradigm, based on a reduction of constituency parsing to sequence labeling. We first described a linearization function

Model	Testbed	CPU Run #Cores	CPU Run Sents/s	GPU Run #GPU	GPU Run Sents/s	F-score
Sequence labeling						
MLP$_e^{\Psi,\Phi}$	WSJ23	1	501	1	669	74.1
MLP$_e^{\Phi'}$	WSJ23	1	349	1	929	74.8
BILSTM$^{\Psi,\Phi}_{m=1,e}$	WSJ23	1	148	1	581	88.1
BILSTM$^{\Phi'}_{m=1,e}$	WSJ23	1	221	1	1016	88.3
BILSTM$^{\Psi,\Phi}_{m=2,e,ch}$	WSJ23	1	66	1	434	89.9
BILSTM$^{\Phi'}_{m=2,e,ch}$	WSJ23	1	115	1	780	90.0
BILSTM$^{\Psi,\Phi}_{m=2,e}$	WSJ23	1	74	1	506	90.0
BILSTM$^{\Phi'}_{m=2,e}$	WSJ23	1	126	1	898	90.0
Sequence-to-sequence						
3-layer LSTM	WSJ 23					<70
3-layer LSTM + Attention$^\diamond$ (Vinyals et al., 2015)	WSJ 23	Multi-core (number not specified)	120			88.3
Constituency parsing as dependency parsing						
Fernández-González and Martins (2015)$^\diamond$	WSJ23	1	41			90.2
Chart-based parsers						
Charniak (2000)*	WSJ23	1	6			89.5
Petrov and Klein (2007)*	WSJ23	1	6			90.1
Stern et al. (2017)$^\diamond$	WSJ23	16*	20			91.8
Kitaev and Klein (2018) +ELMo (Peters et al., 2018)$^\diamond$	WSJ23			2	70	95.1
Chart-based parsers with GPU-specific implementation						
Canny et al. (2013)$^\diamond$	WSJ(<30)			1	250	
Hall et al. (2014)$^\diamond$	WSJ(<40)			1	404	
Transition-based and other greedy constituent parsers						
Zhu et al. (2013)$^\diamond$	WSJ23	1	101			89.9
Zhu et al. (2013)+Padding$^\diamond$	WSJ23	1	90			90.4
Dyer et al. (2016)$^\triangleright$	WSJ23	1	17			91.2
Fernández and Gómez-Rodríguez (2018)$^\diamond$	WSJ23	1	18			91.7
Stern et al. (2017)$^\diamond$	WSJ23	16*	76			91.8
Liu and Zhang (2017)	WSJ23					91.8
Shen et al. (2018)	WSJ23			1	111	91.8

Table 2: Comparison against the state of the art.*Stern et al. (2017) report that they use a 16-core machine, but sentences are processed one-at-a-time. Hence, they do not exploit inter-sentence parallelism, but they may gain some speed from intra-sentence parallelism. $\diamond$ indicates the that the speed was reported in the paper itself. $*$ and $\triangleright$ indicate that the speeds were extracted from Zhu et al. (2013) and Fernández and Gómez-Rodríguez (2018).

Model	F-score
MLP$_e^{\Psi,\Phi}$	63.1
MLP$_e^{\Phi'}$	64.4
BILSTM$^{\Psi,\Phi}_{m=2,e,ch}$	84.4
BILSTM$^{\Phi'}_{m=2,e,ch}$	84.1
BILSTM$^{\Psi,\Phi}_{m=2,e}$	84.4
BILSTM$^{\Phi'}_{m=2,e}$	83.1
Zhu et al. (2013)	82.6
Zhu et al. (2013)+P	83.2
Dyer et al. (2016)	84.6
Liu and Zhang (2017)	86.1
Shen et al. (2018)	86.5
Fernández and Gómez-Rodríguez (2018)	86.8

Table 3: Performance on the CTB test set

to transform a constituent tree (with n leaves) into a sequence of $n-1$ labels that encodes it. We proved that this encoding function is total and injective for any tree without unary branches. We also discussed its limitations: how to deal with unary branches and non-surjectivity, and showed how these can be solved. We finally proposed a set of fast and strong baselines.

Acknowledgments

This work has received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from the TELEPARES-UDC project (FFI2014-51978-C2-2-R) and the ANSWER-ASAP project (TIN2017-85160-C2-1-R) from MINECO, and from Xunta de Galicia (ED431B 2017/01). We gratefully acknowledge NVIDIA Corporation for the donation of a GTX Titan X GPU.

References

John Canny, David Hall, and Dan Klein. 2013. A multi-teraflop constituency parser using GPUs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1898–1907.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018. Faster Shift-Reduce Constituent Parsing with a Non-Binary, Bottom-Up Strategy. *ArXiv e-prints*.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533. Association for Computational Linguistics.

Jenny Rose Finkel, Alex Kleeman, and Christopher D Manning. 2008. Efficient, feature-based, conditional random field parsing. *Proceedings of ACL-08: HLT*, pages 959–967.

Carlos Gómez-Rodríguez. 2017. Towards fast natural language parsing: FASTPARSE ERC Starting Grant. *Procesamiento del Lenguaje Natural*, 59.

Carlos Gómez-Rodríguez, Iago Alonso-Alonso, and David Vilares. 2017. How important is syntactic parsing accuracy? An empirical evaluation on rule-based sentiment analysis. *Artificial Intelligence Review*.

David Hall, Taylor Berg-Kirkpatrick, and Dan Klein. 2014. Sparser, better, faster GPU parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 208–217.

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

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shashi Narayan and Shay B. Cohen. 2016. Optimizing spectral learning for parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1546–1556, Berlin, Germany. Association for Computational Linguistics.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

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

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Marek Rei and Anders Søgaard. 2018. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 293–302. Association for Computational Linguistics.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132. Association for Computational Linguistics.

Helmut Schmid. 1994. Part-of-speech tagging with neural networks. In *Proceedings of the 15th Conference on Computational Linguistics - Volume 1*, COLING '94, pages 172–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Mengqiu Wang, Kenji Sagae, and Teruko Mitamura. 2006. A fast, accurate deterministic parser for chinese. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 425–432, Stroudsburg, PA, USA. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79, Melbourne, Australia. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 434–443.

Synthetic Data Made to Order: The Case of Parsing

Dingquan Wang and **Jason Eisner**
Department of Computer Science, Johns Hopkins University
{wdd,jason}@cs.jhu.edu

Abstract

To approximately parse an unfamiliar language, it helps to have a treebank of a similar language. But what if the closest available treebank still has the wrong word order? We show how to (stochastically) permute the constituents of an existing dependency treebank so that its surface part-of-speech statistics approximately match those of the target language. The parameters of the permutation model can be evaluated for quality by dynamic programming and tuned by gradient descent (up to a local optimum). This optimization procedure yields trees for a new artificial language that resembles the target language. We show that delexicalized parsers for the target language can be successfully trained using such "made to order" artificial languages.

1 Introduction

Dependency parsing is a core task in natural language processing (NLP). Given a sentence, a dependency parser produces a dependency tree, which specifies the typed head-modifier relations between pairs of words. While *supervised* dependency parsing has been successful (McDonald and Pereira, 2006; Nivre, 2008; Kiperwasser and Goldberg, 2016), unsupervised parsing can hardly produce useful parses (Mareček, 2016). So it is extremely helpful to have some treebank of supervised parses for training purposes.

1.1 Past work: Cross-lingual transfer

Unfortunately, manually constructing a treebank for a new target language is expensive (Böhmová et al., 2003). As an alternative, *cross-lingual transfer* parsing (McDonald et al., 2011) is sometimes possible, thanks to the recent development of multi-lingual treebanks (McDonald et al., 2013; Nivre et al., 2015; Nivre et al., 2017). The idea is to parse the sentences of the target language with a supervised parser trained on the treebanks of one or more source languages. Although the parser cannot be expected to know the words of the target language, it can make do with parts of speech (POS) (McDonald et al., 2011; Täckström et al., 2013; Zhang and Barzilay, 2015) or cross-lingual word embeddings (Duong et al., 2015; Guo et al., 2016; Ammar et al., 2016). A more serious challenge is that the parser may not know how to handle the word *order* of the target language, unless the source treebank comes from a closely related language (e.g., using German to parse Luxembourgish). Training the parser on trees from *multiple* source languages may mitigate this issue (McDonald et al., 2011) because the parser is more likely to have seen target part-of-speech sequences somewhere in the training data. Some authors (Rosa and Žabokrtský, 2015a,b; Wang and Eisner, 2016) have shown additional improvements by preferring source languages that are "close" to the target language, where the closeness is measured by distance between POS language models trained on the source and target corpora.

1.2 This paper: Tailored synthetic data

We will focus on delexicalized dependency parsing, which maps an input POS tag sequence to a dependency tree. We evaluate single-source transfer—train a parser on a single source language, and evaluate it on the target language. This is the setup of Zeman and Resnik (2008) and Søgaard (2011a).

Our novel ingredient is that *rather than seek a close source language that already exists, we create one*. How? Given a dependency treebank of a possibly distant source language, we stochastically permute the children of each node, according to some distribution that *makes the permuted language close to the target language*.

And how do we find this distribution? We adopt the tree-permutation model of Wang and Eisner (2016). We design a dynamic programming algorithm which, for any given distribution p in Wang and Eisner's family, can compute the *expected counts* of all POS bigrams in the permuted source treebank. This allows us to evaluate p by computing the divergence between the bigram POS language model formed by these expected counts,

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1325–1337
Brussels, Belgium, October 31 - November 4, 2018.

and the one formed by the *observed counts* of POS bigrams in the unparsed target language. In order to find a p that locally minimizes this divergence, we adjust the model parameters by stochastic gradient descent (SGD).

1.3 Key limitations in this paper

Better measures of surface closeness between two languages might be devised. However, even counting the expected POS N-grams is moderately expensive, taking time exponential in N if done exactly. So we compute only these local statistics, and only for $N = 2$. We certainly need $N > 1$ because the 1-gram distribution is not affected by permutation at all. $N = 2$ captures useful bigram statistics: for example, to mimic a verb-final language with prenominal modifiers, we would seek constituent permutations that result in matching its relatively high rate of `VERB`–`PUNCT` and `ADJ`–`NOUN` bigrams. While $N > 2$ might have improved the results, it was too slow for our large-scale experimental design. §7 discusses how richer measures could be used in the future.

We caution that throughout this paper, we assume that our corpora are annotated with gold POS tags, even in the target language (which lacks any gold training trees). This is an idealized setting that has often been adopted in work on unsupervised and cross-lingual transfer.§7 discusses a possible avenue for doing without gold tags.

2 Modeling Surface Realization

We begin by motivating the idea of tree permutation. Let us suppose that the dependency tree for a sentence starts as a labeled graph—a tree in which siblings are not yet ordered with respect to their parent or one another. Each language has some systematic way to *realize* its unordered trees as surface strings:[1] it imposes a particular order on the tree's word tokens. More precisely, a language specifies a *distribution* $p(\text{string} \mid \text{unordered tree})$ over a tree's possible realizations.

As an engineering matter, we now make the strong assumption that the unordered dependency trees are similar across languages. That is, we suppose that different languages use similar underlying syntactic/semantic graphs, but differ in how they realize this graph structure on the surface.

Thus, given a gold POS corpus $\mathbf{u}$ of the unknown target language, we may hope to explain its distribution of surface POS bigrams as the result of applying some target-language surface realization model to the distribution of cross-linguistically "typical" unordered trees. To obtain samples of the latter distribution, we use the treebanks of one or more *other* languages. The present paper evaluates our method when only a single source treebank is used. In the future, we could try tuning a mixture of all available source treebanks.

2.1 Realization is systematic

We presume that the target language applies the same stochastic realization model to all trees. All that we can optimize is the parameter vector of this model. Thus, we deny ourselves the freedom to realize each individual tree in an *ad hoc* way. To see why this is important, suppose the target language is French, whose corpus $\mathbf{u}$ contains many `NOUN`–`ADJ` bigrams. We could achieve such a bigram from the unordered source tree 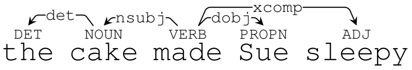

by ordering it to yield 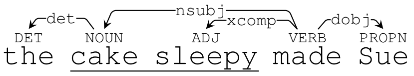

. However, that realization is not in fact appropriate for French, so that ordered tree would not be a useful training tree for French. Our approach should disprefer this tempting but incorrect realization, because any model with a high probability of this realization would, if applied *systematically* over the whole corpus, also yield sentences like `He sleepy made Sue`, with unwanted `PRON`–`ADJ` bigrams that would not match the surface statistics of French. We hope our approach will instead choose the realization model that is correct for French, in which the `NOUN`–`ADJ` bigrams arise instead from source trees where the `ADJ` is a dependent of the `NOUN`, yielding (e.g.) 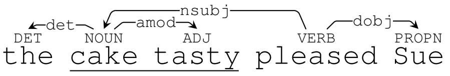

. This has the same POS sequence as the example above (as it happens), but now assigns the correct tree to it.

2.2 A parametric realization model

As our family of realization distributions, we adopt the log-linear model used for this purpose by Wang and Eisner (2016). The model assumes that the root node a of the unordered dependency tree selects an ordering $\pi(a)$ of the n_a nodes consisting

---

[1] Modeling this process was the topic of the recent Surface Realization Shared Task (Mille et al., 2018). Most relevant is work on *tree linearization* (Filippova and Strube, 2009; Futrell and Gibson, 2015; Puzikov and Gurevych, 2018).

of a and its $n_a - 1$ dependent children. The procedure is repeated recursively at the child nodes. This method can produce only projective trees.

Each node a draws its ordering $\pi(a)$ independently according to

$$p_{\boldsymbol{\theta}}(\pi \mid a) = \frac{1}{Z(a)} \exp \sum_{1 \le i < j \le n_a} \boldsymbol{\theta} \cdot \mathbf{f}(\pi, i, j) \quad (1)$$

which is a distribution over the $n_a!$ possible orderings. $Z(a)$ is a normalizing constant. $\mathbf{f}$ is a feature vector extracted from the ordered pair of nodes π_i, π_j, and $\boldsymbol{\theta}$ is the model's parameter vector of feature weights. See Appendix A for the feature templates, which are a subset of those used by Wang and Eisner (2016). These features are able to examine the tree's node labels (POS tags) and edge labels (dependency relations). Thus, when a is a verb, the model can assign a positive weight to "subject precedes verb" or "subject precedes object," thus preferring orderings with these features.

Following Wang and Eisner (2016, §3.1), we choose new orderings for the noun and verb nodes only,[2] preserving the source treebank's order at all other nodes a.

2.3 Generating training data

Given a source treebank B and some parameters $\boldsymbol{\theta}$, we can use equation (1) to randomly sample realizations of the trees in B. The effect is to reorder dependent phrases within those trees. The resulting permuted treebank B' can be used to train a parser for the target language.

2.4 Choosing parameters $\boldsymbol{\theta}$

So how do we choose $\boldsymbol{\theta}$ that works for the target language? Suppose $\mathbf{u}$ is a corpus of target-language POS sequences, using the same set of POS tags as B. We evaluate parameters $\boldsymbol{\theta}$ according to whether POS tag sequences in B' will be distributed like POS tag sequences in $\mathbf{u}$.

To do this, first we estimate a bigram language model $\hat{q}$ from the actual distribution q of POS sequences observed in $\mathbf{u}$. Second, let $p_{\boldsymbol{\theta}}$ denote the distribution of POS sequences that we expect to see in B', that is, POS sequences obtained by stochastically realizing observed trees in B according to $\boldsymbol{\theta}$. We estimate another bigram model $\hat{p}_{\boldsymbol{\theta}}$ from this distribution $p_{\boldsymbol{\theta}}$.

We then try to set $\boldsymbol{\theta}$, using SGD, to minimize a divergence $D(\hat{p}_{\boldsymbol{\theta}}, \hat{q})$ that we will define below.

2.4.1 Estimation of bigram models

Estimating $\hat{q}$ is straightforward: $\hat{q}(t \mid s) = c_q(st)/c_q(s)$, where $c_q(st)$ is the count of POS bigram st in the average[3] sentence of $\mathbf{u}$ and $c_q(s) = \sum_{t'} c_q(st')$. We estimate $\hat{p}_{\boldsymbol{\theta}}$ in the same way, where $c_p(st)$ denotes the *expected* count of st in a random POS sequence $\mathbf{y} \sim p_{\boldsymbol{\theta}}$. This is equivalent to choosing $\hat{q}, \hat{p}_{\boldsymbol{\theta}}$ to minimize the KL-divergences $\mathrm{KL}(q \mid\mid \hat{q}), \mathrm{KL}(p_{\boldsymbol{\theta}} \mid\mid \hat{p}_{\boldsymbol{\theta}})$. It ensures that each model's expected bigram counts match those in the POS sequences.

However, these maximum-likelihood estimates might overfit on our finite data, $\mathbf{u}$ and B. We therefore smooth both models by first adding $\lambda = 0.1$ to all bigram counts $c_q(st)$ and $c_p(st)$.[4]

2.4.2 Divergence of bigram models

We need a metric to evaluate $\boldsymbol{\theta}$. If p and q are bigram language models over POS sequences $\mathbf{y}$ (sentences), their Kullback-Leibler divergence is

$$\mathrm{KL}(p \mid\mid q) \overset{\text{def}}{=} \mathbb{E}_{\mathbf{y} \sim p}[\log p(\mathbf{y}) - \log q(\mathbf{y})] \quad (2)$$

$$= \sum_{s,t} c_p(st) \cdot (\log p(t \mid s) - \log q(t \mid s)) \quad (3)$$

where $\mathbf{y}$ ranges over POS sequences and st ranges over POS bigrams. These include bigrams where $s = \text{BOS}$ ("beginning of sequence") or $t = \text{EOS}$ ("end of sequence"), which are boundary tags that we take to surround $\mathbf{y}$.

All quantities in equation (3) can be determined directly from the (expected) bigram counts given by c_p and c_q. No other model estimation is needed.

A concern about equation (3) is that a single bigram st that is badly underrepresented in q may contribute an arbitrarily large term $\log \frac{p(t|s)}{q(t|s)}$. To limit this contribution to at most $\log \frac{1}{\alpha}$, for some small $\alpha \in (0, 1)$, we define $\mathrm{KL}_\alpha(p \mid\mid q)$ by a variant of equation (3) in which $q(t \mid s)$ has been replaced by $\tilde{q}(t \mid s) \overset{\text{def}}{=} \alpha p(t \mid s) + (1 - \alpha) q(t \mid s)$.[5]

---

[2] Specifically, the 93% of nodes tagged with `NOUN`, `PROPN`, `PRON` or `VERB` in Universal Dependencies format. In retrospect, this restriction was unnecessary in our setting, but it skipped only 4.4% of nodes on average (from 2% to 11% depending on language). The remaining nodes were nouns, verbs, or childless.

[3] A more familiar definition of c_q would use the *total* count in $\mathbf{u}$. Our definition, which yields the same bigram probabilities, is analogous to our definition of c_p. This c_p is needed for $\mathrm{KL}(p \mid\mid q)$ in (3), and c_q symmetrically for $\mathrm{KL}(q \mid\mid p)$.

[4] Ideally one should tune λ to minimize the language model perplexity on held-out data (e.g., by cross-validation).

[5] This is inspired by the α-skew divergence of Lee (1999,

Our final divergence metric $D(\hat{p}_{\boldsymbol{\theta}}, \hat{q})$ defines D as a linear combination of exclusive and inclusive KL_α divergences, which respectively emphasize $p_{\boldsymbol{\theta}}$'s precision and recall at matching q's bigrams:

$$D(p,q) = (1-\beta)\cdot\frac{\mathrm{KL}_{\alpha_1}(p \,||\, q)}{\mathbb{E}_{\mathbf{y}\sim p}[\,|\mathbf{y}|\,]} + \beta\cdot\frac{\mathrm{KL}_{\alpha_2}(q \,||\, p)}{\mathbb{E}_{\mathbf{y}\sim q}[\,|\mathbf{y}|\,]} \tag{4}$$

where $\beta, \alpha_1, \alpha_2$ are tuned by cross-validation to maximize the downstream parsing performance. The division by average sentence length converts KL from nats per sentence to nats per word,[6] so that the KL values have comparable scale even if B has much longer or shorter sentences than $\mathbf{u}$.

3 Algorithms

3.1 Efficiently computing expected counts

We now present a polynomial-time algorithm for computing the expected bigram counts c_p under $p_{\boldsymbol{\theta}}$ (or equivalently $\hat{p}_{\boldsymbol{\theta}}$), for use above. This averages expected counts from each unordered tree $\mathbf{x} \in B$. Algorithm 1 in the supplement gives pseudocode.

The insight is that rather than sampling a single realization of $\mathbf{x}$ (as B' does), we can use dynamic programming to sum efficiently over all of its exponentially many realizations. This gives an exact answer. It algorithmically resembles tree-to-string machine translation, which likewise considers the possible reorderings of a source tree and incorporates a language model by similarly tracking their surface N-grams (Chiang, 2007, §5.3.2).

For each node a of the tree $\mathbf{x}$, let the POS string $\mathbf{y}_a$ be the realization of the subtree rooted at a. Let $c_a(st)$ be the expected count of bigram st in $\mathbf{y}_a$, whose distribution is governed by equation (1). We allow $s = \textsc{bos}$ or $t = \textsc{eos}$ as defined in §2.4.2.

The c_a function can be represented as a sparse map from POS bigrams to reals. We compute c_a at each node a of $\mathbf{x}$ in a bottom-up order. The final step computes c_{root}, giving the expected bigram counts in $\mathbf{x}$'s realization $\mathbf{y}$ (that is, c_p in §2.4).

We find c_a as follows. Let $n = n_a$ and recall from §2.2 that $\pi(a)$ is an ordering of $a_1, \ldots, a_n$, where $a_1, \ldots, a_{n-1}$ are the child nodes of a, and a_n is a dummy node representing a's head token. Also, let a_0 and a_{n+1} be dummy nodes that always appear at the start and end of any ordering.

For all $0 \leq i \leq n$ and $1 \leq j \leq n+1$, let $p_a(i,j)$ denote the expected count of the $a_i a_j$ node bigram—the probability that $\pi(a)$ places node a_i immediately before node a_j. These *node bigram probabilities* can be obtained by enumerating all possible orderings π, a matter we return to below.

It is now easy to compute c_a:

$$c_a(st) = c_a^{\text{within}}(st) + c_a^{\text{between}}(st) \tag{5}$$

$$c_a^{\text{within}}(st) = \begin{cases} \sum_{i=1}^{n} c_{a_i}(st) & \text{if } s \neq \textsc{bos}, t \neq \textsc{eos} \\ 0 & \text{otherwise} \end{cases}$$

$$c_a^{\text{across}}(st) = \sum_{i=0}^{n}\sum_{j=1}^{n+1} p_a(i,j) c_{a_i}(s\,\textsc{eos}) c_{a_j}(\textsc{bos}\,t)$$

That is, c_a inherits all non-boundary bigrams st that fall within its child constituents (via c_a^{within}). It also counts bigrams st that cross the boundary between consecutive nodes (via c_a^{across}), where nodes a_i and a_j are consecutive with probability $p_a(i,j)$.

When computing c_a via (5), we will have already computed $c_{a_1}, \ldots, c_{a_{n-1}}$ bottom-up. As for the dummy nodes, a_n is realized by the length-1 string h where h is the head token of node a, while a_0 and a_{n+1} are each realized by the empty string. Thus, c_{a_n} simply assigns count 1 to the bigrams $\textsc{bos}\,h$ and $h\,\textsc{eos}$, and c_{a_0} and $c_{a_{n+1}}$ each assign expected count 1 to $\textsc{bos}\ \textsc{eos}$. (Notice that thus, $c_a^{\text{across}}(st)$ counts $\mathbf{y}_a$'s boundary bigrams—the bigrams st where $s = \textsc{bos}$ or $t = \textsc{eos}$—when $i = 0$ or $j = n+1$ respectively.)

3.2 Efficient enumeration over permutations

The main challenge above is computing the node bigram probabilities $p_a(i,j)$. These are marginals of $p(\pi \mid a)$ as defined by (1), which unfortunately is intractable to marginalize: there is no better way than enumerating all $n!$ permutations.

That said, there is a particularly efficient way to enumerate the permutations. The Steinhaus-Johnson-Trotter (SJT) algorithm (Sedgewick, 1977) does so in $O(1)$ time per permutation, obtaining each permutation by applying a single swap to the previous one. Only the features that are affected by this swap need to be recomputed. For our features (Appendix A), this cuts the runtime per permutation from $O(n^2)$ to $O(n)$.

Furthermore, the single swap of adjacent nodes only changes 3 bigrams (possibly including boundary bigrams). As a result, it is possible to

---

2001). Indeed, we may regard $\mathrm{KL}_\alpha(p \,||\, q)$ as the α-skew divergence between the unigram distributions $p(\cdot \mid s)$ and $q(\cdot \mid s)$, averaged over all s in proportion to $c_p(s)$. In principle, we could have used the α-skew divergence between the distributions $p(\cdot)$ and $q(\cdot)$ over POS sequences $\mathbf{y}$, but computing that would have required a sampling-based approximation (§7).

[6] Recall that the units of negated log-probability are called bits for log base 2, but nats for log base e.

obtain the marginal probabilities with $O(1)$ additional work per permutation. When a node bigram is destroyed, we increment its marginal probability by the total probability of permutations encountered since the node bigram was last created. This can be found as a difference of partial sums. The final partial sum is the normalizing constant $Z(a)$, which can be applied at the end. Pseudocode is given in supplementary material as Algorithm 2.

When we train the parameters $\boldsymbol{\theta}$ (§2.4), we must back-propagate through the whole computation of equation (4), which depends on tag bigram counts $c_a(st)$, which depend via (5) on expected node bigram counts $p_a(i, j)$, which depend via Algorithm 2 on the permutation probabilities $p(\pi \mid a)$, which depend via (1) on the feature weights $\boldsymbol{\theta}$.

4 Heuristics

4.1 Pruning high-degree trees

As a further speedup, we only train on trees with number of words < 40 and $\max_a n_a \leq 5$, so $n_a! \leq 120$.[7] We then produce the synthetic treebank B' (§2.3) by drawing a single realization of each tree in B for which $\max_a n_a \leq 7$. This requires sampling from up to $7! = 5040$ candidates per node, again using SJT.[8]

That is, in this paper we run exact algorithms (§3), but only on a subset of B. The subset is not necessarily representative. An improvement would use importance sampling, with a proposal distribution that samples the slower trees less often during SGD but upweights them to compensate.

§7 suggests a future strategy that would run on all trees in B via approximate, sampling-based algorithms. The exact methods would remain useful for calibrating the approximation quality.

4.2 Minibatch estimation of c_p

To minimize (4), we use the Adam variant of SGD (Kingma and Ba, 2014), with learning rate 0.01 chosen by cross-validation (§5.1).

SGD requires a stochastic estimate of the gradient of the training objective. Ordinarily this is done by replacing an expectation over the entire training set with an expectation over a minibatch.

Equation (2) with $p = \hat{p}_{\boldsymbol{\theta}}$ is indeed an expectation over sentences of B. It can be stochastically estimated as (3) where c_p gives the expected bigram counts averaged over only the sentences in a minibatch of B. These are found using §3's algorithms with the current $\boldsymbol{\theta}$. Unfortunately, the term $\log p(t \mid s)$ depends on bigram counts that should be derived from the *entire* corpus B in the same way. Our solution is to simply reuse the *minibatch* estimate of c_p for the latter counts. We use a large minibatch of 500 sentences from B so that this *drop-in estimate* does not introduce too much bias into the stochastic gradient: after all, we only need to estimate bigram statistics on 17 POS types.[9]

By contrast, the c_q values that are used for the expectation in the second term of (4) and in $\log q(t \mid s)$ do not change during optimization, so we simply compute them once from all of $\mathbf{u}$.

4.3 Informed initialization

Unfortunately the objective (4) is not convex, so the optimizer is sensitive to initialization (see §5.3 below for empirical discussion). Initializing $\boldsymbol{\theta} = \mathbf{0}$ (so that $p(\pi \mid a)$ is uniform) gave poor results in pilot experiments. Instead, we initially choose $\boldsymbol{\theta}$ to be the realization parameters of the source language, as estimated from the source treebank B. This is at least a linguistically realistic $\boldsymbol{\theta}$, although it may not be close to the target language.[10]

For this initial estimation, we follow Wang and Eisner (2016) and perform supervised training on B of the log-linear realization model (1), by maximizing the conditional log-likelihood of B, namely $\sum_{(\mathbf{x},\mathbf{t})\in B} \log p_{\boldsymbol{\theta}}(\mathbf{t} \mid \mathbf{x})$, where $(\mathbf{x}, \mathbf{t})$ are an unordered tree and its observed ordering in B. This initial objective is convex.[11]

5 Experiments

We performed a large-scale experiment requiring hundreds of thousands of CPU-hours. To our knowledge, this is the largest study of parsing transfer yet attempted.

---

[7] We found that this threshold worked much better than ≤ 4 and about as well as the much slower ≤ 6.

[8] This pruning heuristic retains 36.1% of the trees (averaging over the 20 development treebanks (§5.1)) for training, and 66.6% for actual realization. The latter restriction follows Wang and Eisner (2016, §4.2): they too discarded trees with nodes having $n_a \geq 8$.

[9] We also used the minibatch to estimate the average sentence length $\mathbb{E}_{\mathbf{y}\sim p}[\,|\mathbf{y}|\,]$ in (4), although here we could have simply used all of B since this value does not change.

[10] As an improvement, one could also try initial realization parameters for B that are estimated from treebanks of *other* languages. Concretely, the optimizer could start by selecting a "galactic" treebank from Wang and Eisner (2016) that is already close to the target language, according to (4), and try to make it even closer. We leave this to future work.

[11] Unfortunately, we did not regularize it, which probably resulted in initializing some parameters too close to $\pm\infty$ for the optimizer to change them meaningfully.

5.1 Data and setup

As our main dataset, we use Universal Dependencies version 1.2 (Nivre et al., 2015)—a set of 37 dependency treebanks for 33 languages, with a unified POS-tag set and relation label set.

Our evaluation metric was unnormalized attachment score (UAS) when parsing a target treebank with a parser trained on a (possibly permuted) source treebank. For both evaluation and training, we used only the training portion of each treebank.

Our parser was Yara (Rasooli and Tetreault, 2015), a fast and accurate transition-based dependency parser that can be rapidly retrained. We modified Yara to ignore the input words and use only the input *gold* POS tags (see §1.3). To train the Yara parser on a (possibly permuted) source treebank, we first train on 80% of the trees and use the remaining 20% to tune Yara's hyperparameters. We then retrain Yara on 100% of the source trees and evaluate it on the target treebank.

Similar to Wang and Eisner (2017), we use 20 treebanks (18 distinct languages) as *development data*, and hold out the remaining 17 treebanks for the final evaluation. We chose the hyperparameters $(\alpha_1, \alpha_2, \beta)$ of (4) to maximize the target-language UAS, averaged over all 376 transfer experiments where the source and target treebanks were development treebanks of different languages.[12] (See Appendix C for details.)

The next few sections perform some exploratory analysis on these 376 experiments. Then, for the final test in §5.4, we will evaluate UAS on all 337 transfer experiments where the source is a development treebank and the target is a test treebank of a different language.[13]

5.2 Exploratory analysis

We have assumed that a smaller divergence between source and target treebanks results in better transfer parsing accuracy. Figure 1 shows that these quantities are indeed correlated, both for the original source treebanks and for their "made to order" permuted versions.

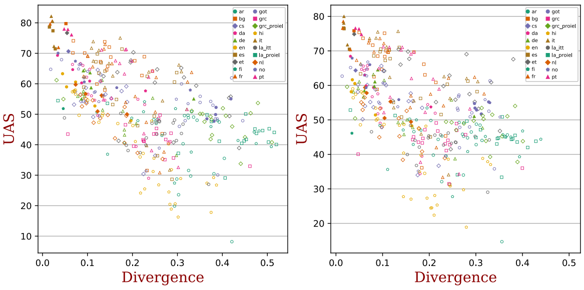

Figure 1: UAS is higher when divergence is lower. Each point represents a pair of source and target languages, whose shape and color identify the treebank of the target language (see legend). The marker is solid if the source and target languages belong to the same language family.[15] The left graph uses the original source treebank (Kendall's $\tau = -0.41$), while the right graph uses its permuted version ($\tau = -0.39$).

Thus, we hope that the optimizer will find a systematic permutation that reduces the divergence. Does it? Yes: Figures 5 and 6 in the supplementary material show that the optimizer almost always manages to reduce the objective on training data, as expected.

One concern is that our divergence metric might misguide us into producing dysfunctional languages whose trees cannot be easily recovered from their surface strings, i.e., they have no good parser. In such a language, the word order might be extremely free (e.g., $\boldsymbol{\theta} = \mathbf{0}$), or common constructions might be syntactically ambiguous. Fortunately, Appendix D shows that our synthetic languages appear natural with respect to their their parsability.

The above findings are promising. So does permuting the source language in fact result in better transfer parsing of the target language? We experiment on the 376 development pairs.

The solid lines in Figure 2 show our improvements on the dev data, with a simpler scatterplot given by in Figure 7 in the supplementary material. The upshot is that the synthetic source treebanks yield a transfer UAS of 52.92 on average. This is not yet a result on held-out test data: recall that 52.92 was the best transfer UAS achieved by any hyperparameter setting. That said, it is 1.00 points better than transferring from the original source treebanks, a significant difference (paired permutation test by language pair, $p < 0.01$).

Figure 2 shows that this average improvement is mainly due to the many cases where the source and target languages come from different families.

---

[12] We have 19*20=380 pairs in total, minus the four excluded pairs (grc, grc_proiel), (grc_proiel, grc), (la_proiel, la_itt) and (la_itt, la_proiel). Unlike Wang and Eisner (2017), we exclude duplicated languages in development and testing.

[13] Specifically, there are 3 duplicated sets: {grc, grc_proiel}, {la, la_proiel, la_itt}, and {fi, fi_ftb}. Whenever one treebank is used as the target language, we exclude the other treebanks in the same set.

[15] According to the family (and sub-family) information at `http://universaldependencies.org`.

Permutation tends to improve source languages that were doing badly to start with. However, it tends to hurt a source language that is already in the target language family.

A hypothetical experiment shows that permuting the source does have good *potential* to help (or at least not hurt) in both cases. The dashed lines in Figure 2—and the scatterplot in Figure 8—show the potential of the method, by showing the improvement we would get from permuting each source treebank using an "oracle" realization policy—the supervised realization parameters $\boldsymbol{\theta}$ that are estimated from the actual target treebank. The usefulness of this oracle-permuted source varies depending on the source language, but it is usually much better than the automatically-permuted version of the same source.

This shows that large improvements would be possible if we could only find the best permutation policy allowed by our model family. The question for future work is whether such gains can be achieved by a more sensitive permutation model than (1), a better divergence objective than (4), or a better search algorithm than §4.2. Identifying the best available source treebank, or the best mixture of all source treebanks, would also help greatly.

5.3 Sensitivity to initializer

Figure 2 makes clear that performance of the synthetic source treebanks is strongly correlated with that of their original versions. Most points in Figure 7 lie near the diagonal (Kendall's $\tau = 0.85$). Even with oracle permutation in Figure 8, the correlation remains strong ($\tau = 0.59$), suggesting that the choice of source treebank is important even beyond its effect on search initialization.

We suspected that when "made to order" source treebanks (more than the oracle versions) have performance close to their original versions, this is in part because the optimizer can get stuck near the initializer (§4.3). To examine this, we experimented with random restarts, as follows. In addition to informed initialization (§4.3), we optimized from 5 other starting points $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, I)$. From these 6 runs, we selected the final parameters that achieved the best divergence (4). As shown by Figure 9 in the supplement, greater gains appear to be possible with more aggressive search methods of this sort, which we leave to future work. We could also try non-random restarts based on the realization parameters of other languages, as suggested in footnote 10.

5.4 Final evaluation on the test languages

For our final evaluation (§5.1), we use the same hyperparameters (Appendix C) and report on single-source transfer to the 17 held-out treebanks.

The development results hold up in Figure 3. Using the synthetic languages yields 50.36 UAS on average—1.75 points over the baseline, which is significant (paired permutation test, $p < 0.01$).

In the supplementary material (Appendix E), we include some auxiliary experiments on multi-source transfer.

6 Related Work

6.1 Unsupervised parsing

Unsupervised parsing has remained challenging for decades (Mareček, 2016). Classical grammar induction approaches (Lari and Young, 1990; Carroll and Charniak, 1992; Klein and Manning, 2004; Headden III et al., 2009; Naseem et al., 2010) estimate a generative grammar to explain the sentences, for example by the Expectation-Maximization (EM) algorithm, and then use it to parse. Some such approaches try to improve the grammar model. For example, Klein and Manning (2004)'s dependency model with valence was the first to beat a trivial baseline; later improvements considered higher-order effects and punctuation (Headden III et al., 2009; Spitkovsky et al., 2012). Other approaches try to avoid search error, using strategies like convexified objectives (Wang et al., 2008; Gimpel and Smith, 2012), informed initialization (Klein and Manning, 2004; Mareček and Straka, 2013), search bias (Smith and Eisner, 2005, 2006; Naseem et al., 2010; Gillenwater et al., 2010), branch-and-bound search (Gormley and Eisner, 2013), and switching objectives (Spitkovsky et al., 2013).

The alternative of cross-lingual transfer has recently flourished thanks to the development of consistent cross-lingual datasets of POS-tagged (Petrov et al., 2012) and dependency-parsed (McDonald et al., 2013) sentences. McDonald et al. (2011) showed a significant improvement over grammar induction by simply using the delexicalized parser trained on other language(s). Subsequent improvements have come from re-weighting source languages (Søgaard, 2011b; Rosa and Žabokrtský, 2015a,b; Wang and Eisner, 2016), adapting the model to the target language using WALS (Dryer and Haspelmath, 2013) features (Naseem et al., 2012; Täckström et al., 2013;

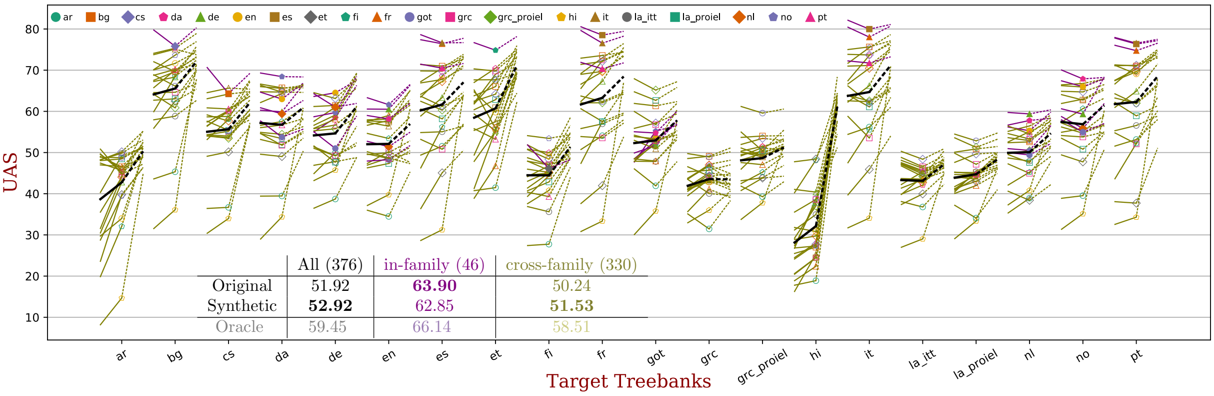

	All (376)	in-family (46)	cross-family (330)
Original	51.92	**63.90**	50.24
Synthetic	**52.92**	62.85	**51.53**
Oracle	59.45	66.14	58.51

Figure 2: Unlabeled attachment scores (UAS) from 376 pairs of development treebanks. Each column represents a target treebank, and each polyline within that column shows transfer from variants of a different source treebank. The three points on the polyline (from left to right) represent the target UAS for parsers trained on three sources: the original source treebank, the "made to order" permutation that attempts to match surface statistics of the target treebank, and an oracle permutation that uses a realization model trained on the target language. We use solid markers and purple lines if the transfer is *within-family* (source and target treebank from the same language family), and hollow and olive for cross-family transfer. The **black** polyline in each column is the mean of the others. The table in the lower left gives summary results; the number in each column header gives the number of points summarized. For each column, we boldface the better result between the "Synthetic" and "Original", or both if they are not significantly different (paired permutation test, $p < 0.01$). We also show the oracle permutation result in row "Oracle".

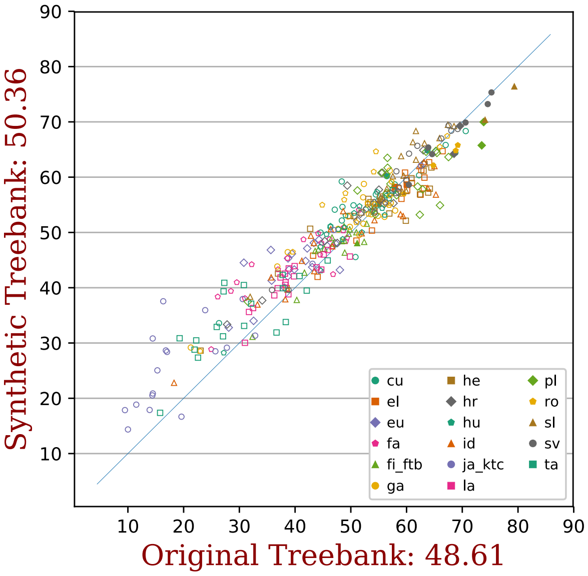

Figure 3: UAS on 337 language pairs from the training languages to the test languages.

Zhang and Barzilay, 2015; Ammar et al., 2016), and improving the lexical representations via multilingual word embeddings (Duong et al., 2015; Guo et al., 2016; Ammar et al., 2016) and synthetic data generation (§6.2).

6.2 Synthetic data generation

Our novel proposal ties into the recent interest in data augmentation in supervised machine learning. In unsupervised parsing, the most widely adopted synthetic data method has been *annotation projection*, which generates synthetic analyses of target-language sentences by "projecting" the analysis from a source-language translation. Of course, this requires bilingual corpora as an additional resource. Annotation projection was proposed by Yarowsky et al. (2001), gained promising results on sequence labelling tasks, and was later developed for unsupervised parsing (Hwa et al., 2005; Ganchev et al., 2009; Smith and Eisner, 2009; Tiedemann, 2014; Ma and Xia, 2014; Tiedemann et al., 2014). Recent work in this vein has mainly focused on improving the synthetic data, including reweighting the training trees (Agić et al., 2016) or pruning those that cannot be aligned well (Rasooli and Collins, 2015, 2017; Lacroix et al., 2016).

On the other hand, Wang and Eisner (2016) proposed to permute source language treebanks using word order realization models trained on other source languages. They generated on the order of 50,000 synthetic languages by "mixing and matching" a few dozen source languages. Their idea was that with a large set of synthetic languages, they could use them as supervised examples to train an unsupervised structure discovery system that could analyze any new language. Systems built with this dataset were competitive in single-source parser transfer (Wang and Eisner, 2016), typology prediction (Wang and Eisner, 2017), and parsing unknown languages (Wang and Eisner, 2018).

Our work in this paper differs in that our synthetic treebanks are "made to order." Rather than combine aspects of different treebanks and hope to get at least one combination that is close to the target language, we "combine" the source treebank with a POS corpus of the target language, which guides our customized permutation of the source.

Beyond unsupervised parsing, synthetic data has been used for several other tasks. In NLP, it has been used for complex tasks such as question-answering (QA) (Serban et al., 2016) and machine reading comprehension (Weston et al., 2016; Hermann et al., 2015; Rajpurkar et al., 2016), where highly expressive neural models are used and not enough real data is available to train them. In the playground of supervised parsing, Gulordava and Merlo (2016) conduct a controlled study on the parsibility of languages by generating treebanks with short dependency length and low variability of word order.

7 Conclusion & Future Work

We have shown how cross-lingual transfer parsing can be improved by permuting the source treebank to better resemble the target language on the surface (in its distribution of gold POS bigrams). The code is available at `https://github.com/wddabc/ordersynthetic`. Our work is grounded in the notion that by trying to explain the POS bigram counts in a target corpus, we can discover a stochastic realization policy for the target language, which correctly "translates" the source trees into appropriate target trees.

We formulated an objective for evaluating such a policy, based on KL-divergence between bigram models. We showed that the objective could be computed efficiently by dynamic programming, thanks to the limitation to bigram statistics.

Experimenting on the Universal Dependencies treebanks v1.2, we showed that the synthetic treebanks were—on average—modestly but significantly better than the corresponding real treebanks for single-source transfer (and in Appendix E, on multi-source transfer).

On the downside, Figure 7 shows that with our current method, permuting the source language to be more like the target language is helpful (on average) only when the source language is from a different language family. This contrast would be even more striking if we had a better optimizer: Figure 9 shows that SGD's initialization bias limits its permutation's benefit for cross-family training, as well as its harm for within-family training.

Several opportunities for future work have already been mentioned throughout the paper. We are also interested in experimenting with richer families of permutation distributions, as well as "conservative" distributions that tend to prefer the original source order. We could use entropy regularization (Grandvalet and Bengio, 2005) to encourage more "deterministic" patterns of realization in the synthetic languages.

We would also like to consider more sensitive divergence measures that go beyond bigrams, for example using recurrent neural network language models (RNNLMs) for $\hat{q}$ and $\hat{p}_{\boldsymbol{\theta}}$. This means abandoning our exact dynamic programming methods; we would also like to abandon exact exhaustive enumeration in order to drop §4.1's bounds on n. Fortunately, there exist powerful MCMC methods (Eisner and Tromble, 2006) that can sample from interesting distributions over the space of $n!$ permutations, even for large n. Thus, we could approximately sample from $p_{\boldsymbol{\theta}}$ by drawing permuted versions of each tree in B.

Given this change, a very interesting direction would be to graduate from POS language models to word language models, using cross-lingual unsupervised word embeddings (Ruder et al., 2017). This would eliminate the need for the gold POS tags that we unrealistically assumed in this paper (which are typically unavailable for a low-resource target language). Furthermore, it would enable us to harness richer lexical information beyond the 17 UD POS tags. After all, even a (gold) POS corpus might not be sufficient to determine the word order of the target language: "`NOUN VERB NOUN`" could be either subject-verb-object or object-verb-subject. However, "`water drink boy`" is presumably object-verb-subject. Thus, using cross-lingual embeddings, we would try to realize the unordered source trees so that their word strings, with few edits, can achieve high probability under a neural language model of the target.

Acknowledgements

This work was supported by National Science Foundation Grants 1423276 & 1718846. We are grateful to the state of Maryland for the Maryland Advanced Research Computing Center, a crucial resource. We thank Shijie Wu and Adithya Renduchintala for early discussion, Argo lab members for further discussion, and the 3 reviewers for quality comments.

References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association of Computational Linguistics*, 4:431–444.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague dependency treebank. In *Treebanks*, pages 103–127. Springer.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Statistically-Based Natural Language Processing Techniques: Papers from the Workshop*, pages 1–13, Menlo Park: AAAI Press. AAAI. Technical Report WS-92-01.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig. `http://wals.info/`.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 113–122.

Jason Eisner and Roy W. Tromble. 2006. Local search with very large-scale neighborhoods for optimal permutations in machine translation. In *Proceedings of the HLT-NAACL Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 57–75.

Katja Filippova and Michael Strube. 2009. Tree linearization in English: Improving language model based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 225–228.

Richard Futrell and Edward Gibson. 2015. Experiments with generative models for dependency tree linearization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1978–1983.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 369–377.

Jennifer Gillenwater, Kuzman Ganchev, Joo Graa, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 194–199.

Kevin Gimpel and Noah A. Smith. 2012. Concavity and initialization for unsupervised dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 577–581.

Matthew Gormley and Jason Eisner. 2013. Nonconvex global optimization for latent-variable models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 444–454.

Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 529–536. MIT Press.

Kristina Gulordava and Paola Merlo. 2016. Multilingual dependency parsing evaluation: A large-scale analysis of word order properties using artificial data. *Transactions of the Association for Computational Linguistics*, 4:343–356.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *AAAI*, pages 2734–2740.

William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association of Computational Linguistics*, 4:313–327.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 478–485.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063.

Karim Lari and Steve J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4(1):35–56.

Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 25–32.

Lillian Lee. 2001. On the effectiveness of the skew divergence for statistical language analysis. In *Proceedings of AISTATS*.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1337–1348.

David Mareček and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 281–290.

David Mareček. 2016. Twelve years of unsupervised dependency parsing. In *Proceedings of the 16th ITAT Conference on Information Technologies—Applications and Theory*, pages 56–62.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97.

Ryan McDonald and Fernando Pereira. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. 2018. The first multilingual surface realisation shared task (SR'18): Overview and evaluation results. In *Proceedings of the 1st Workshop on Multilingual Surface Realization (MSR), 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–12.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà Mỹ, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phng Lê H`ông, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre et al. 2015. Universal dependencies 1.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague. Data available at `http://universaldependencies.org`.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

Yevgeniy Puzikov and Iryna Gurevych. 2018. BinLin: A simple method of dependency tree linearization. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 13–28.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *Transactions of the Association for Computational Linguistics*, 5:279–293.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733 (version 2).

Rudolf Rosa and Zdeněk Žabokrtský. 2015a. KLcpos3 — a language similarity measure for delexicalized parser transfer. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 243–249.

Rudolf Rosa and Zdeněk Žabokrtský. 2015b. MSTParser model interpolation for multi-source delexicalized transfer. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 71–75.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. *Computing Research Repository*, arXiv:1706.04902.

Robert Sedgewick. 1977. Permutation generation methods. *ACM Computing Surveys*, 9(2):137–164.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598.

David A. Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 822–831.

Noah A. Smith and Jason Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Grammatical Inference Applications*, pages 73–82.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the International Conference on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL)*, pages 569–576.

Anders Søgaard. 2011a. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 682–686. Association for Computational Linguistics.

Anders Søgaard. 2011b. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 682–686.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2012. Three dependency-and-boundary models for grammar induction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 688–698. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071.

Jörg Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING): Technical Papers*, pages 1854–1864. Dublin City University and Association for Computational Linguistics.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 130–140.

Dingquan Wang and Jason Eisner. 2016. The Galactic Dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the Association of Computational Linguistics*, 4:491–505. Data available at `https://github.com/gdtreebank/gdtreebank`.

Dingquan Wang and Jason Eisner. 2017. Fine-grained prediction of syntactic typology: Discovering latent structure with *supervised* learning. *Transactions of the Association for Computational Linguistics (TACL)*, 5.

Dingquan Wang and Jason Eisner. 2018. Surface statistics of an unknown language indicate how to parse it. *Transactions of the Association for Computational Linguistics (TACL)*. To appear.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *Proceedings of ACL-HLT*, pages 532–540.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *Proceedings of the International Conference on Learning Representations*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1857–1867.

Tell-and-Answer: Towards Explainable Visual Question Answering using Attributes and Captions

Qing Li[1], **Jianlong Fu**[2], **Dongfei Yu**[1], **Tao Mei**[3], **Jiebo Luo**[4]
[1]University of Science and Technology of China
[2]Microsoft Research, Beijing, China
[3]JD AI Research, Beijing 100105, China
[4]University of Rochester, Rochester, NY
[1]`{sealq, ydf2010}@mail.ustc.edu.cn`, [2]`jianf@microsoft.com`,
[3]`tmei@jd.com`, [4]`jluo@cs.rochester.edu`

Abstract

In Visual Question Answering, most existing approaches adopt the pipeline of representing an image via pre-trained CNNs, and then using the uninterpretable CNN features in conjunction with the question to predict the answer. Although such end-to-end models might report promising performance, they rarely provide any insight, apart from the answer, into the VQA process. In this work, we propose to break up the end-to-end VQA into two steps: **explaining** and **reasoning**, in an attempt towards a more explainable VQA by shedding light on the intermediate results between these two steps. To that end, we first extract attributes and generate descriptions as explanations for an image. Next, a reasoning module utilizes these explanations in place of the image to infer an answer. The advantages of such a breakdown include: (1) the attributes and captions can reflect what the system extracts from the image, thus can provide some insights for the predicted answer; (2) these intermediate results can help identify the inabilities of the image understanding or the answer inference part when the predicted answer is wrong. We conduct extensive experiments on a popular VQA dataset and our system achieves comparable performance with the baselines, yet with added benefits of explanability and the inherent ability to further improve with higher quality explanations.

1 Introduction

Answering textual questions from images, which is referred to as visual question answering, presents fundamental challenges to both computer vision and natural language processing communities. Significant progress has been made on VQA in recent years (Antol et al., 2015; Zhu et al., 2016; Wu et al., 2016a; Yang et al., 2016; Goyal et al., 2017; Yu et al., 2017; Teney et al., 2017; Wang et al., 2017; Gurari et al., 2018; Anderson et al., 2018; Lu et al., 2018; Li et al., 2018). A widely used pipeline is to first encode an image with Convolutional Neural Networks (CNNs) and represent associated questions with Recurrent Neural Networks (RNNs), and then formulate the vision-to-language task as a classification problem on a list of answer candidates. Although promising performance has been reported, this end-to-end paradigm fails to provide any insight to illuminate the VQA process. In most cases, giving answers without any explanation cannot satisfy hu-

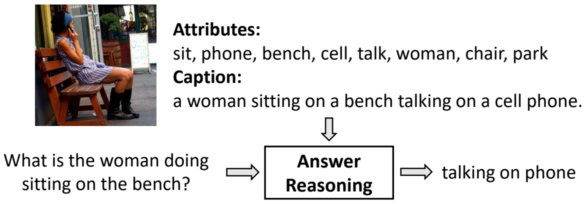

Figure 1: An example of explanation and reasoning in VQA. We first extract attributes in the image such as “sit”, “phone” and “woman.” A caption is also generated to encode the relationship between these attributes, e.g. “woman sitting on a bench.” Then a reasoning module uses these explanations to predict an answer “talking on phone.”

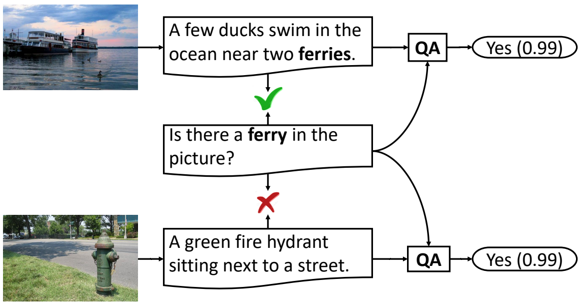

Figure 2: Two contrasting cases that show how the explanations can be used to determine if the system *guesses* the answer.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1338–1346
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

man users, especially when the predicted answer is not correct. More frustratingly, the system gives no hint about which part of such systems is the culprit for a wrong answer.

To address the above issues, we propose to break up the popular end-to-end pipeline into two steps: **explaining** and **reasoning**. The philosophy behind such a break-up is to mimic the image question answering process of human beings: first understanding the content of the image and then performing inference about the answer according to the understanding. As is shown in Fig.1, we first generate two-level explanations for an image via pre-trained attribute detectors and image captioning model: 1). word-level: attributes, indicating individual objects and attributes the system learns from the image. 2). sentence-level: captions, representing the relationship between the objects and attributes. Then the generated explanations and question are infused to a reasoning module to predict an answer. The reasoning module is mainly composed of LSTMs.

Our method has three benefits. First, these explanations are interpretable. According to the attributes and captions, we can tell what objects, attributes and their relationship the machine learns from the image as well as what information is lost during the image understanding step. In contrast, the fully-connected layer features of CNNs are usually uninterpretable to humans. When the predicted answer is correct, these attributes and captions can be provided for users as the supplementary explanations to the answer. Second, the separation of explaining and reasoning enables us to localize which step of the VQA process the error comes from when the predicted answer is wrong. If the explanations don't include key information to answer the question, the error is caused by missing information during the explaining step. Otherwise, the reasoning module should be responsible for the wrong answer. Third, the explanations can also indicate whether the system really finds key information from the image to answer the question or merely *guesses* an answer. Fig.2 presents two contrasting cases to illustrate this. In the first case, both the generated caption and the question include the key concept "ferry", so the answer "Yes" with a high probability is reliable. However, although the answer "Yes" has the same high probability in the second case, the caption is irrelevant to the question. The system sticks to a wrong answer even with the correct input from sentence generation. This is due to the training set bias that a large proportion of questions starting with "is there" in the training set have the answer "Yes".

To our knowledge, this is the first effort to break down the previous end-to-end pipeline to shed light on the VQA process. Our main contributions are summarized as follows:

- We propose to formulate VQA into two separate steps: **explaining** and **reasoning**. Our framework generate attributes and captions for images to shed light on why the system predicts any specific answer.
- We adopt several ways to measure the explanation quality and demonstrate strong correlation between explanation quality and VQA accuracy. The current system achieves comparable performance to the baselines and can naturally improve with explanation quality.
- Extensive experiments are conducted on the popular VQA dataset (Antol et al., 2015). We dissect all results according to the measurements of the quality of explanations to present a thorough analysis of the strength and weakness of our framework.

2 Related Work

There is a growing research interest in the task of visual question answering. In this section, we summarize recent advances from two directions.

Attention in VQA. The attention mechanism is firstly used in the machine translation task (Bahdanau et al., 2014) and then is brought into the vision-to-language tasks (Xu et al., 2015; You et al., 2016; Yang et al., 2016; Lu et al., 2016; Nam et al., 2017; Yu et al., 2017; Teney et al., 2017; Anderson et al., 2018; Teney et al., 2018; Liang et al., 2018). The visual attention in the vision-to-language tasks is used to address the problem of "where to look". In VQA, the question is used as a query to search for the relevant regions in the image. Yang *et al.* propose a stacked attention model which queries the image for multiple times to infer the answer progressively. Beyond the visual attention, Lu *et al.* exploit a hierarchical question-image co-attention strategy to attend to both related regions in the image and crucial words in the question. Attention mechanism can find the question-related regions in the image, which accounts for the answer to some extent. But the attended regions still don't explicitly exhibit what the system learns from the image and it is also not explained why these regions should be attended to.

High-level Concepts. In the scenario of vision-to-language, high-level concepts exhibit superior performance than the low-level or middle-level visual features of the image (Fang et al., 2015; Wu et al., 2016a,b). (Fang et al., 2015) first learn independent detectors for visual words based on a multi-instance learning framework and then generate descriptions for images based on the set of visually detected words via a maximum entropy language model. (Wu et al., 2016a,b) presents a thorough study on how much the high-level concepts can benefit the image captioning and visual question answering tasks. These work mainly uses high-level concepts to obtain a better performance. Different from these work, our paper is focused on fully exploiting the readability and understandability of attributes and captions to explain the process of visual question answering and use these explanations to analyze our system.

3 Methodology

In this section, we introduce the proposed framework for the breakdown of VQA. As illustrated in Figure 3, the framework consists of three modules: word prediction, sentence generation, and answer reasoning. Next, we describe the three modules in details.

3.1 Word Prediction

From the work (Wu et al., 2016a), we have learned that explicit high-level attributes can benefit vision-to-language tasks. In fact, besides performance gain, the readability and understandability of attributes also makes them an intuitive way to explain what the model learns from images.

We first build a word list based on MS COCO Captions (Chen et al., 2015). We extract the most N frequent words in all captions and filter them by lemmatization and removing stop words to determine a list of 256 words, which cover over 90% of the word occurrences in the dataset. Our words are not tense or plurality sensitive, for example, "horse" and "horses" are considered as the same word. This significantly decreases the size of our word list. Given the word list, every image is paired with multiple labels (words) according to its captions. Then we formulate word prediction as a multi-label classification task and fine-tune the ResNet-152 (He et al., 2016) on our image-words dataset by minimizing the element-wise sigmoid cross entropy loss:

$$J = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{V} -y_{ij}\log p_{ij} - (1-y_{ij})\log(1-p_{ij}) \tag{1}$$

where N is batch size, V is the size of word list, $\mathbf{y}_i = [y_{i1}, y_{i2}, ..., y_{iV}], y_{ij} \in \{0, 1\}$ is the label vector of the i^{th} image, $\mathbf{p}_i = [p_{i1}, p_{i2}, ..., p_{iV}]$ is the probability vector.

In the testing phase, instead of using region proposals like (Wu et al., 2016a), we directly feed the whole image into the word prediction CNN in order to keep simple and efficient. As a result, each image is encoded into a fixed-length vector, where each dimension represents the probability of the corresponding word occurring in the image.

Word Quality Evaluation. We adopt two metrics to evaluate the predicted words. The first measures the **accuracy** of the predicted words by computing cosine similarity between the label vector $\mathbf{y}$ and the probability vector $\mathbf{p}$:

$$a = \frac{\mathbf{y}^T\mathbf{p}}{||\mathbf{y}|| \cdot ||\mathbf{p}||} \tag{2}$$

However, this metric disregards the extent to which the predicted words are relevant to the question. Intuitively speaking, question-relevant explanations for images should be more likely to help predict right answers than irrelevant ones. Therefore, we propose another metric to measure the **relevance** between the words and the question. We first encode the question into a 0-1 vector $\mathbf{q}$ in terms of the word list. Then the relevance is computed as:

$$r = \frac{\mathbf{q}^T\mathbf{p}}{||\mathbf{q}|| \cdot ||\mathbf{p}||} \tag{3}$$

3.2 Sentence Generation

This section we talk about generating sentence-level explanations for images by using a pre-trained image captioning model. Similar to (Vinyals et al., 2015), we train an image captioning model by maximizing the probability of the correct caption given an image. Suppose we have an image $\mathcal{I}$ to be described by a caption $\mathcal{S} = \{s_1, s_2, ..., s_L\}, s_t \in \mathcal{V}$, where $\mathcal{V}$ is the vocabulary, L is the caption length. First the image $\mathcal{I}$ is represented by the activations of the first

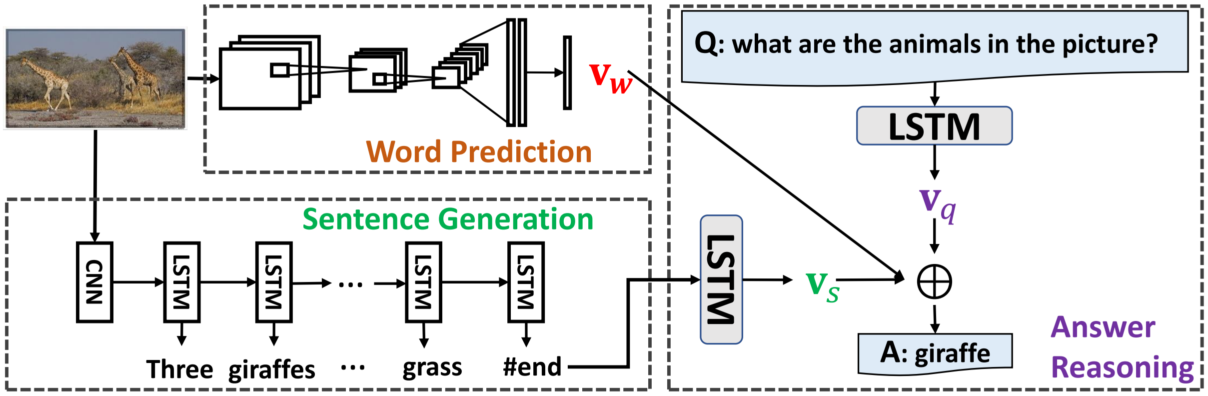

Figure 3: An overview of the proposed framework for VQA with three modules: word prediction (upper left), sentence generation (lower left), answer reasoning (right). **Explaining**: in word prediction, the image is fed into pre-trained visual detectors to extract word-level explanation, which is represented by probability vector $\mathbf{v}_w$; in sentence generation, we input the image to pre-trained captioning model to generate a sentence-level explanation. **Reasoning**: the caption and question are encoded by two different LSTMs into $\mathbf{v}_s$ and $\mathbf{v}_q$, respectively. Then $\mathbf{v}_q, \mathbf{v}_w$ and $\mathbf{v}_s$ are concatenated and fed to a fully connected layer with softmax to predict an answer.

fully connected layer of ResNet-152 pre-trained on ImageNet, denoted as $\mathbf{v}_i$. The caption $\mathcal{S}$ can be represented as a sequence of one-hot vector $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_L\}$. Then we formulate the caption generation problem as minimizing the cost function:

$$\begin{aligned} J(\mathbf{v}_i, \mathbf{S}) &= -\log P(\mathbf{S}|\mathbf{v}_i) \\ &= -\sum_{t=0}^{L} \log P(\mathbf{s}_t|\mathbf{v}_i, \mathbf{s}_1, ..., \mathbf{s}_{t-1}) \end{aligned} \tag{4}$$

where $P(\mathbf{s}_t|\mathbf{v}_i, \mathbf{s}_1, ..., \mathbf{s}_{t-1})$ is the probability of generating the word $\mathbf{s}_t$ given the image representation $\mathbf{v}_i$ and previous words $\{\mathbf{s}_1, ..., \mathbf{s}_{t-1}\}$. We employ a single-layer LSTM with 512-dimensional hidden states to model this probability. In the testing phase, the image is input to pre-trained image captioning model to generate sentence-level explanation.

Sentence Quality Evaluation. Similar to word quality evaluation, we evaluate the quality of the generated sentence from two perspectives: **accuracy** and **relevance**. The former one is an average fusion of four widely used metrics: BLEU@N, METEOR, ROUGE-L and CIDEr-D (Chen et al., 2015), which try to consider the accuracy of the generated sentence from different perspectives. Note that we normalize all the metrics into $[0, 1]$ before fusion. The latter metric is to measure the relevance between the generated sentence and the question. The binary TF weights are calculated over all words of the sentence to produce an integrated representation of the entire sentence, denoted by $\mathbf{s}$. Likewise, the question can be encoded to $\mathbf{q}$. The relevance is computed as:

$$r = \frac{\mathbf{q}^T \mathbf{s}}{||\mathbf{q}|| \cdot ||\mathbf{s}||} \tag{5}$$

3.3 Answer Reasoning

This section we discuss the reasoning module. Suppose we have an image $\mathcal{I}$ explained by the predicted words $\mathcal{W}$ and the generated sentence $\mathcal{S}$, the question $\mathcal{Q}$ and the answer $\mathcal{A}$. As shown in Fig.3, we denote the representations of the predicted words $\mathcal{W}$ as $\mathbf{v}_s$. The caption $\mathcal{S}$ and question $\mathcal{Q}$ are encoded by two different LSTMs into $\mathbf{v}_s$ and $\mathbf{v}_q$, respectively. What bears mentioning is that these two LSTMs share a common word-embedding matrix, but not other parameters, because the question and caption have different grammar structures and similar vocabularies. At last, the $\mathbf{v}_w$, $\mathbf{v}_s$, and $\mathbf{v}_q$ are concatenated and fed into a fully connected layer with softmax to predict the probability on a set of candidate answers:

$$\mathbf{v} = [\mathbf{v}_w^T \ \mathbf{v}_s^T \ \mathbf{v}_q^T]^T \tag{6}$$

$$\mathbf{p} = softmax(\mathbf{W}\mathbf{v} + \mathbf{b}) \tag{7}$$

where $\mathbf{W}, \mathbf{b}$ are the weight matrix and bias vector of the fully connected layer. The optimizing objective for the reasoning module is to minimize the cross entropy loss as:

$$J(\mathcal{I}, \mathcal{Q}, \mathcal{A}) = J(\mathcal{W}, \mathcal{S}, \mathcal{Q}, \mathcal{A}) = -\log \mathbf{p}(\mathcal{A}) \tag{8}$$

where $\mathbf{p}(\mathcal{A})$ denotes the probability of the ground truth $\mathcal{A}$.

4 Experiments and Analysis

4.1 Experiment Setting

Dataset. We evaluate our framework on VQA-real (Antol et al., 2015) dataset. For each image in VQA-real, 3 questions are annotated by different workers and each question has 10 answers from different annotators. We follow the official split and report our results on the open-ended task.

Metric. We use the accuracy: $\min(\frac{\#\text{humans giving that answer}}{3}, 1)$, i.e., an answer is deemed 100% accurate if at least three workers provided that exact answer.

Ablation Models. To analyze the contribution of word-level and sentence-level explanations, we ablate the full model and evaluate several variants as:

- **Word-based VQA**: use the feature concatenation of the predicted words and question in Eq.6.
- **Sentence-based VQA**: use the feature concatenation of the generated sentence and question in Eq.6.
- **Full VQA**: use the feature concatenation of words, sentence, and question in Eq.6.

4.2 Word-based VQA

Table 1: The relationship between word quality and accuracy (%).

(a) Word accuracy

Word accuracy	VQA accuracy
[0.0, 0.2)	46.30
[0.2, 0.8)	55.84
[0.8, 1.0)	58.52

(b) W-Q Relevance

W-Q Relevance	VQA accuracy
[0.0, 0.2)	54.69
[0.2, 0.8)	60.23
[0.8, 1.0)	76.15

An important characteristics of our framework is that the quality of explanations can influence the final VQA performance. In this section, we analyze the impact of the quality of predicted words on the VQA accuracy. We measure the quality from two sides: word accuracy and word-question relevance. Table 1a shows the relationship between word accuracy and VQA performance. We can learn that the more accurate the predicted words, the better the VQA performance. Similar to word accuracy, the more relevant to the question the predicted words, the better the VQA performance. Particularly, when the word-question relevance exceeds 0.8, the predicted words are highly pertinent to the question, boosting the VQA accuracy to 76.15%. This indicates high-quality word-level explanations can benefit the VQA performance a lot. As shown in Fig.4, word-question relevance has a bigger impact on the final VQA performance than word accuracy.

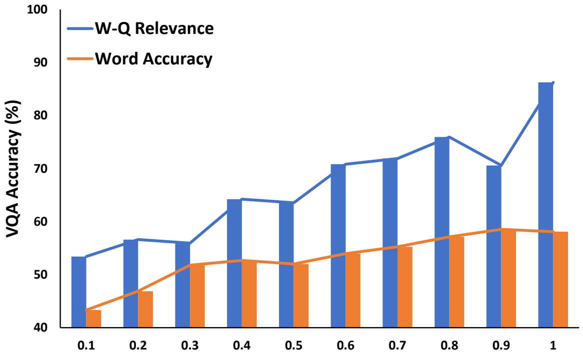

Figure 4: The comparison of the impact of word accuracy and word-question relevance on VQA performance.

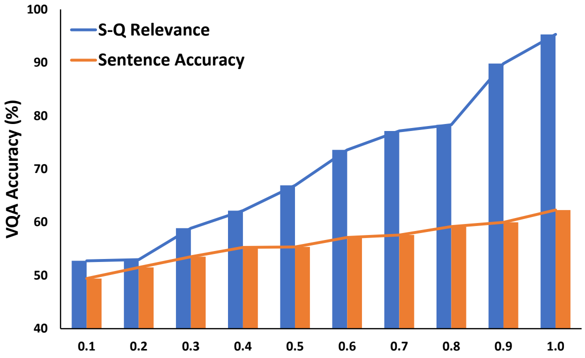

Figure 5: The comparison of the impact of sentence accuracy and sentence-question relevance on VQA performance.

4.3 Sentence-based VQA

Table 2: The relationship between sentence quality and accuracy.

(a) Sentence accuracy

Sentence accuracy	VQA accuracy
[0.0, 0.2)	51.29
[0.2, 0.8)	55.53
[0.8, 1.0)	61.33

(b) S-Q Relevance

S-Q Relevance	VQA accuracy
[0.0, 0.2)	52.79
[0.2, 0.8)	62.34
[0.8, 1.0)	89.81

In this section, we evaluate the sentence-based VQA model and analyze the relationship between the sentence quality and the VQA performance.

Table 3: Performance comparison on the validation split of VQA-real open-ended task when the sentence-based VQA model uses different sources of captions. (accuray in %)

Caption source	validation			
	All	Y/N	Num	Others
null	46.21	73.82	34.98	25.63
sentence generation	54.85	76.31	36.64	42.23
relevant groundtruth	56.05	77.42	41.04	44.34

Similar to the quality measurements of predicted words, we focus on the accuracy of the generated sentence itself and the relevance between sentence and question. As shown in Table 2a, the more accurate the generated sentence, the higher the VQA accuracy. The results suggest that the VQA performance can be further improved by a better image captioning model. From Table 2b, we can see that the more relevant to the question the generated sentence, the better the VQA performance. Once the relevance reaches 0.8, the accuracy can significantly increase to 89.81%. This proves that a question-related sentence is more likely to contain the key information for the VQA module to answer the question. As shown in Fig. 5, sentence-question relevance has greater influence on VQA performance than sentence accuracy does.

To further verify the causal relationship between sentence quality and VQA performance, we conduct the following control experiments. First, we evaluate sentence-based VQA model when feeding different sources of captions with ascending quality: *null* (only including an "#end" token), *sentence generation* and *relevant groundtruth* (selecting from the groundtruth captions the most relevant one to the question). As shown in Table 3, *sentence generation* performs much better than *null*. And using *relevant groundtruth* captions, the accuracy can improve by another 1.2 percent. Figure 6 presents an example to illustrate the effect of the sentence quality on the accuracy. From the above analysis, we can safely reach the conclusion that the VQA performance can be greatly improved by generating sentence-level explanations of high quality, especially of high relevance to the question.

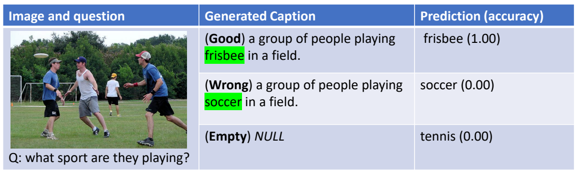

Image and question	Generated Caption	Prediction (accuracy)
Q: what sport are they playing?	(**Good**) a group of people playing frisbee in a field.	frisbee (1.00)
	(**Wrong**) a group of people playing soccer in a field.	soccer (0.00)
	(**Empty**) *NULL*	tennis (0.00)

Figure 6: A control case for comparing the accuracy when inputting captions of different quality. When getting a caption of high quality (the first one), the system can answer the question correctly. If we manually change the "frisbee" to "soccer", a wrong answer is predicted. When using an empty sentence, the system predicts the most popular answer "tennis" for this question.

4.4 Case Study

From the above evaluation of word-based and sentence-based VQA model, we conclude that the relevance between explanations (attributes/caption) and the question has a great impact on the final VQA performance. In this section we illustrate this conclusion by studying four possible types of cases: 1). *high relevance and correct answer*; 2). *low relevance and wrong answer*; 3). *high relevance but wrong answer*; 4). *low relevance but correct answer.*

High relevance and correct answer. From the first case in Fig. 7, we can see that the explanations for the image are highly relevant to the question: both the predicted attributes and the generated sentence contain the words "man" and "racket" occurring in the question. And the explanations also has key information that can predict the answer "tennis court." In this type of case, the system successfully extracts from the image the relevant information that covers the question, facilitating answer generation.

Low relevance and wrong answer. In the second case, although the attributes and caption can reflect part of the image content such as "man" and "food", they neglect the key information about the "glass" that is asked in the question. The absence of "glass" in the explanations produces a low explanation-question relevance score and leads the system to a wrong answer. In this type of case, two lessons can be derived from the low relevance: 1). as the explanations are irrelevant to the question, the system tends to predict the most frequent answer ("beer") for this question type ("what kind of drink ..."), which implies that the answer is actually *guessed* from the dataset bias; 2). the error comes from the image understanding part rather than the question answering module, because the system fails to extract from the image

Image	Explanations, Question and Answer
① **High relevance and correct answer**	
	**Attributes**: tennis, ball, man, racket, hit, court, play, player, swing, hold (0.87, 0.72) **Caption**: a man holding a tennis racket on a tennis court. (0.88, 0.68) **Question**: where is the man swinging the racket? **Answer**: tennis court (tennis court)
② **Low relevance and wrong answer**	
	**Attributes**: bicycle, man, sit, eat, bike, look, outside, food, person, table (0.06, 0.48) **Caption**: a man sitting at a table with a plate of food. (0.00, 0.34) **Question**: what kind of drink is in the glass? **Answer**: beer (water)
③ **High relevance but wrong answer**	
	**Attributes:** street, bus, cow, city, walk, car, drive, stand, road, white (0.51, 0.77) **Caption:** a cow that is walking in the street (0.51, 0.55) **Question:** what is walking next to the bus? **Answer:** car (cow)
④ **Low relevance but correct answer**	
	**Attributes:** woman, bear, teddy, hold, sit, glass, animal, large, lady (0.00, 0.19) **Caption:** a woman holding a sandwich in her hands (0.00, 0.30) **Question:** does the man need a haircut? **Answer:** yes (yes)

Figure 7: Four types of cases in our results: 1). high relevance and correct answer; 2). low relevance and wrong answer; 3). high relevance but wrong answer; 4). low relevance but correct answer. "(*,*)" behind the explanations (attributes/caption) denotes the explanation-question relevance score and explanation accuracy, respectively. Gray denotes groundtruth answers.

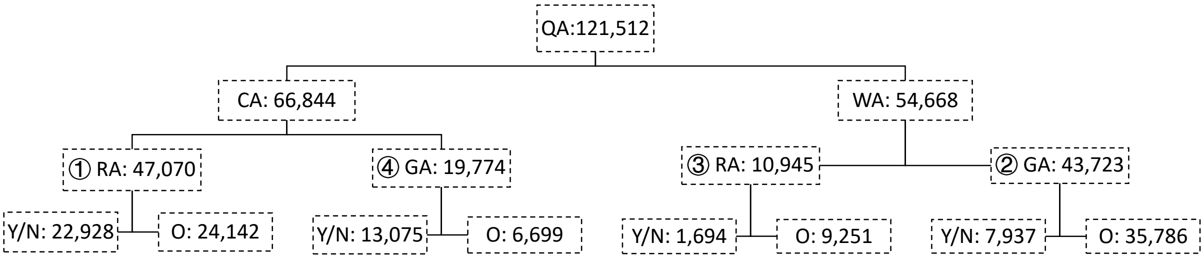

Figure 8: Dataset dissection according to the four types of cases. We define that the answer is guessed when the explanations are irrelevant to the question and otherwise reliable. The case numbers in the third row correspond to these in Fig.7. QA: all questions and answers. CA: questions with correct answers. WA: questions with wrong answers. GA: questions with guessed answers. RA: questions with reliable answers. Y/N: answer type "yes/no". O: answer types other than "yes/no".

enough information to answer the question in the first place. This error suggests that some improvements are needed in word prediction and sentence generation modules to generate more comprehensive explanations for the image.

High relevance but wrong answer. In the third case, we can see that although the system fails to predict the correct answer, the explanations for the image are indeed relevant to the question and the system also recognize the key information "cow." This indicates that the error is caused by the question answering module rather than the explanation generation part. The system can recognize that "a cow is walking in the street" and "a bus is in the street", but it fails to conclude that "the cow is next to the bus." This error may lie in the weakness of LSTM which struggles on such complex spatial relationship inference. In the following analysis, we would show that such cases only occupy a relatively small proportion of the whole dataset.

Low relevance but correct answer. In the last example of Fig. 7, we know from the explanations that the system mistakes the "man" in the image for "woman" and neglects the information about his "hair." The explanations, therefore, have a low relevance score, which indicates that the answer "yes" is guessed by the system. Although the guessed answer is correct, it cannot be credited to the correctness of the system. In fact, for this particular answer type "yes/no", the system has at least 50% chance to hit the right answer.

We dissect all the results in the dataset according to the above four types of cases, as shown in Fig. 8. Among the questions that the system answers correctly, nearly 30% are guessed. This discovery indicates that, buried in the seemingly promising performance, the system actually takes advantage of the dataset bias, rather than truly understands the image content. Over 65% of the answers that are correctly guessed belong to "yes/no", an answer type easier for the system to hit the right answer than other types. As for the questions to which the system predicts wrong answers, a large proportion (around 80%) has a low explanation-question relevance, which means that more efforts need to be put into improving the attributes detectors and image captioning model. Questions with other answer types account for more than 80% of the wrongly-guessed answers. This is not surprising because for these questions the system cannot rely on the dataset bias anymore, considering the great variety of the candidate answers.

4.5 Performance Comparison

Table 4: Performance comparison with the baselines. We show the performance on both test-dev and test-standard splits of VQA-real open-ended task. Human is the human performance for reference.

Method	test-dev				test-standard			
	All	Y/N	Num	Others	All	Y/N	Num	Others
LSTM Q+I (Antol et al., 2015)	53.74	78.94	35.24	36.42	54.06	79.01	35.55	36.80
Concepts (Wu et al., 2016a)	57.46	79.77	36.79	43.10	57.62	79.72	36.04	43.44
ACK (Wu et al., 2016b)	59.17	81.01	38.42	45.23	59.44	81.07	37.12	45.83
MCB w/n attention (Fukui et al., 2016)	60.80	81.20	35.10	49.30	-	-	-	-
Human (Antol et al., 2015)	-	-	-	-	83.30	95.77	83.39	72.67
Word-based VQA	56.76	77.57	35.21	43.85	-	-	-	-
Sentence-based VQA	57.91	78.03	36.73	45.52	-	-	-	-
Full VQA	59.93	79.32	38.41	48.25	60.07	79.09	38.25	48.57

In this section, we present the performance comparison between variants of our framework and the baselines. From Table 4, we can see that sentence-based VQA consistently outperforms word-based VQA, which indicates that sentence-level explanations are superior to word-level ones. The generated captions not only include the objects in the image, but also encode the relationship between these objects, which is important for predicting the correct answer. Furthermore, full VQA model obtains a better performance by combining attributes and captions.

Compared with the baselines, our framework achieves better performance than LSTM Q+I (Antol et al., 2015), Concepts (Wu et al., 2016a), and ACK (Wu et al., 2016b), which use CNN features, high-level concepts, and external knowledge, respectively. MCB without attention (Fukui et al., 2016) achieves better performance than ours and other methods, but it suffers from a high-dimensional feature (16,000 vs 1,280), which poses a limitation on the model's efficiency. The main advantage of our framework over other methods is that it not only predicts an answer to the question, but also generates human-readable attributes and captions to explain the answer. These explanations can help us understand what the system extracts from an image and their relevance to the question. As explanations improve, so would our system.

5 Discussions and Conclusions

In this work, we break up the end-to-end VQA pipeline into **explaining** and **reasoning**, and achieve comparable performance with the baselines. Different from previous work, our method first generates attributes and captions as explanations for an image and then feed these explanations to a question answering module to infer an answer. The merit of our method lies in that these attributes and captions allow a peek into the process of visual question answering. Furthermore, the relevance between these explanations and the question can act as indication whether the system really understands the image content.

It is worth noting although we also use the CNN-RNN combination, we generate words and captions as the explanations of images, thus allowing the VQA system to perform *reasoning on semantics* instead of unexplainable CNN features. Since the effectiveness of CNN for generating attributes and captions is well established, the use of CNN as a component does not contradict our high-level objective for explainable VQA. Our goal is not to immediately make a big gain in performance, but to propose a more powerful framework for VQA. Our current implementation already matches the baselines, but more importantly, provides the ability to explain and to improve.

Acknowledgements

Jiebo Luo would like to thank the support of Adobe and NSF Award #1704309.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *ICCV*, pages 2425–2433.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ICLR*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *CoRR*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *CVPR*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *EMNLP*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *CVPR*.

Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. 2018. Vizwiz grand challenge: Answering visual questions from blind people. In *CVPR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Qing Li, Qingyi Tao, Shafiq Joty, Jianfei Cai, and Jiebo Luo. 2018. Vqa-e: Explaining, elaborating, and enhancing your answers for visual questions. *ECCV*.

Junwei Liang, Lu Jiang, Liangliang Cao, Li-Jia Li, and Alexander G. Hauptmann. 2018. Focal visual-text attention for visual question answering. In *CVPR*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *NIPS*, pages 289–297.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2018. Neural baby talk. In *CVPR*, pages 7219–7228.

Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. 2017. Dual attention networks for multimodal reasoning and matching. *CVPR*.

Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. 2018. Tips and tricks for visual question answering: Learnings from the 2017 challenge. In *CVPR*.

Damien Teney, Lingqiao Liu, and Anton van den Hengel. 2017. Graph-structured representations for visual question answering. In *CVPR*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164.

Peng Wang, Qi Wu, Chunhua Shen, and Anton van den Hengel. 2017. The vqa-machine: Learning how to use existing vision algorithms to answer new questions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Qi Wu, Chunhua Shen, Lingqiao Liu, Anthony Dick, and Anton van den Hengel. 2016a. What value do explicit high level concepts have in vision to language problems? In *CVPR*.

Qi Wu, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2016b. Ask me anything: Free-form visual question answering based on knowledge from external sources. *CVPR*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *CVPR*, pages 21–29.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *CVPR*.

Dongfei Yu, Jianlong Fu, Tao Mei, and Yong Rui. 2017. Multi-level attention networks for visual question answering. In *CVPR*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *CVPR*, pages 4995–5004.

Learning a Policy for Opportunistic Active Learning

Aishwarya Padmakumar, Peter Stone and Raymond J. Mooney
Department of Computer Science
University of Texas at Austin
{aish,pstone,mooney}@cs.utexas.edu

Abstract

Active learning identifies data points to label that are expected to be the most useful in improving a supervised model. Opportunistic active learning incorporates active learning into interactive tasks that constrain possible queries during interactions. Prior work has shown that opportunistic active learning can be used to improve grounding of natural language descriptions in an interactive object retrieval task. In this work, we use reinforcement learning for such an object retrieval task, to learn a policy that effectively trades off task completion with model improvement that would benefit future tasks.

1 Introduction

In machine learning tasks where obtaining labeled examples is expensive, active learning is used to lower the cost of annotation without sacrificing model performance. Active learning allows a learner to iteratively query for labels of unlabeled data points that are expected to maximally improve the existing model. It has been used in a number of natural language processing tasks such as text categorization (Lewis and Gale, 1994), semantic parsing (Thompson et al., 1999) and information extraction (Settles and Craven, 2008).

The most commonly used framework for active learning is pool-based active learning, where the learner has access to the entire pool of unlabeled data at once, and can iteratively query for examples. In contrast, sequential active learning is a framework in which unlabeled examples are presented to the learner in a stream (Lewis and Gale, 1994). For every example, the learner can decide whether to query for its label or not. This results in an additional challenge – since the learner cannot compare all unlabeled data points before choosing queries, each query must be chosen based on local information only.

Multilabel active learning is the application of active learning in scenarios where multiple labels, that are not necessarily mutually exclusive, are associated with a data point (Brinker, 2006). These setups often suffer from sparsity, both in the number of labels that are positive for a data point, and in the number of positive data points per label.

Opportunistic active learning incorporates a form of multilabel sequential active learning into an interactive task. It was recently introduced for the task of interpreting natural-language object descriptions, motivated by the task of instructing a robot to retrieve a specific item (Thomason et al., 2017). In this task, a human describes one of a set of objects in unrestricted natural language and the agent must determine which object was described. The agent is allowed to ask questions about other objects in the current environment to obtain labels that allow it to learn classifiers for concepts used in such descriptions. As the questions are restricted to the objects available in the current interaction, the learning process across interactions can be seen as a form of multilabel sequential active learning. Further, the agent can either restrict itself to querying labels relevant to understanding the current description, or be opportunistic and query labels that can only aid future interactions – for example querying whether some object is “round” when trying to understand the description “a red box”.

More generally, in opportunistic active learning, an agent is engaged in a series of sequential decision-making tasks. The agent uses one or more supervised models to complete each task. Each task involves some sampled examples from a given feature space, and the agent is allowed to query for labels of these examples to improve its models for current and future tasks. Queries in this setting have a higher cost than in traditional active learning as the agent may choose to query for la-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1347–1357
Brussels, Belgium, October 31 - November 4, 2018.

bels that are not relevant for the current task, but expected to be of use for future tasks. Such opportunistic queries enable an agent to learn from a greater number of interactions, by allowing it to ask queries that would aid future tasks when it is sufficiently confident of completing the current task. They also allow an agent to focus on concepts that could have more impact than those relevant to the current task – for example by choosing a frequently used concept as opposed to a rare one. Further, identifying which queries are optimal for model improvement is more difficult as the agent does not have access to the entire pool of unlabeled examples at any given time, similar to sequential active learning settings.

Another sample application of opportunistic active learning could be in a task oriented dialog system providing restaurant recommendations to a user. In this case, a possible opportunistic query would be to ask the user for a Chinese restaurant they liked, when the user is searching for an Italian one. The query is not relevant to the immediate task of recommending an Italian restaurant but would improve the underlying recommendation system.

Prior work on using opportunistic active learning in understanding natural-language object descriptions has shown that an agent following an opportunistic policy, that queries for labels not necessarily relevant to the current interaction, learns to perform better at identifying objects correctly over time (Thomason et al., 2017). However, this work only compares static policies that select actions based on manually-engineered heuristics. In this work, we focus on *learning* an optimal policy for this task using reinforcement learning, in the spirit of other recent attempts to learn policies for different types of active learning (Fang et al., 2017; Woodward and Finn, 2017). This allows an agent to choose whether or not to be opportunistic based on the specific interaction as well as the overall statistics of the dataset.

Our learned policy outperforms a static baseline by improving its success rate on object retrieval while asking fewer questions on average. The learned policy also learns to distribute queries more uniformly across concepts than the baseline.

2 Related Work

Active learning methods aim to identify examples that are likely to be the most useful in improving a supervised model. A number of metrics have been proposed to evaluate examples, including uncertainty sampling (Lewis and Gale, 1994), density-weighted methods (Settles and Craven, 2008), expected error reduction (Roy and McCallum, 2001), query by committee (Seung et al., 1992), and the presence of conflicting evidence (Sharma and Bilgic, 2016); as surveyed by Settles (2010). Some of these metrics can be extended to the multilabel setting, by assuming that one-vs-all classifiers are learned for each label, and that all the learned classifiers are comparable (Brinker, 2006; Singh et al., 2009; Li et al.). Label statistics have also been incorporated into heuristics for selecting instances to be queried (Yang et al., 2009; Li and Guo, 2013). There have also been Bayesian approaches that select both an instance and label to be queried (Qi et al., 2009; Vasisht et al., 2014). Our work aims to learn a policy for choosing between queries that can use information from many such indicators, but learns to combine them appropriately for a given task.

Thomason et al. (2017) define the setting of opportunistic active learning, and apply it to an interactive task of grounding natural language descriptions of objects. They compare two static policies to demonstrate that using opportunistic queries improves task performance. We try to learn the optimal policy for this task using reinforcement learning, and compare to a policy similar to theirs.

Recently, there has been interest in using reinforcement learning to learn a policy for active learning. Fang et al. (2017) use deep Q-learning to acquire a policy that sequentially examines unlabeled examples and decides whether or not to query for their labels; using it to improve named entity recognition in low resource languages. Also, Bachman et al. (2017) use meta-learning to jointly learn a data selection heuristic, data representation and prediction function for a distribution of related tasks. They apply this to one shot recognition of characters from different languages, and in recommender systems. In contrast to these works, we learn a policy for a task that contains *both* possible actions that are active learning queries, *and* actions that complete the current task, thus resulting in a greater exploration-exploitation trade-off.

More similar to our setup is that of Woodward and Finn (2017) which uses reinforcement learning with a recurrent-neural-network-based Q-function in a sequential one-shot learning task to decide between predicting a label and acquiring the true label at a cost. This setup also has a higher cost than standard active learning where the test set is separated out. This is a continuous task without clearly separated interactions or episodes. In our setting, each episode or interaction allows for querying and requires completion of an interaction, which further increases the trade-off between model improvement and exploitation. Further, we consider a multilabel setting, which increases the number of actions at each decision step.

There are other works that employ various types of turn-taking interaction to learn models for language grounding. Some of these use a restricted vocabulary (Cakmak et al., 2010; Kulick et al., 2013), or additional knowledge of predicates (for example that "red" is a color) (Mohan et al., 2012). Others do not use active learning (Kollar et al., 2013; Parde et al., 2015; De Vries et al., 2017; Yu et al., 2017), or do not learn a policy that guides the interaction (Vogel et al., 2010; Thomason et al., 2016, 2017).

Also related to our work is the use of reinforcement learning in dialog tasks, such as slot-filling and recommendation (Wen et al., 2015; Pietquin et al., 2011), understanding natural language instructions or commands (Padmakumar et al., 2017; Misra et al., 2017), and open domain conversation (Serban et al., 2016; Das et al., 2017). These typically do not use active learning. In our task, the policy needs to trade-off model improvement against task completion.

3 Opportunistic Active Learning

Opportunistic Active Learning (OAL) is a setting that incorporates active learning queries into interactive tasks. Let $O = \{o_1, o_2, \dots o_n\}$ be a set of examples, and $M = \{m_1, m_2, \dots m_k\}$ be supervised models trained for different concepts, using these examples. For the problem of understanding natural-language object descriptions, O corresponds to the set of objects, M corresponds to the set of possible concepts that can be used to describe the objects, for example their categories (such as *ball* or *bottle*) or perceptual properties (such as *red* or *tall*).

In each interaction, an agent is presented with some subset $O_A \subseteq O$, and must make a decision based on some subset of the models $M_A \subseteq M$. Given a set of objects O_A and a natural language description l, M_A would be the set of classifiers corresponding to perceptual predicates present in l. The decision made by the agent is a guess about which object is being described by l. The agent receives a score or reward based on this decision, and needs to maximize expected reward across a series of such interactions. In the task of object retrieval, this is a 0/1 value indicating whether the guess was correct, and the agent needs to maximize the average guess success rate.

During the interaction, the agent may also query for the label of any of the examples present in the interaction $o \in O_A$, for any model $m \in M$. The agent is said to be opportunistic when it chooses to query for a label $m \notin M_A$, as this label will not affect the decision made in the current interaction, and can only help with future interactions. For example, given a description "*the red box*", asking whether an object is *red*, could help the agent make a better guess, but asking whether an object is *round*, would be an opportunistic query. Queries have a cost, and hence the agent needs to trade-off the number of queries with the success at guessing across interactions.

The agent participates in a sequence of such interactions, and the models improve from labels acquired over multiple interactions. Thus the agent's expected reward per interaction is expected to improve as more interactions are completed.

This setting differs from the traditional application of active learning in the following key ways:

- The agent cannot query for the label of any example from the unlabeled pool. It is restricted to the set of objects available in the current interaction, O_A.
- The agent is evaluated on the reward per interaction, rather than the final accuracy of the models in M.
- The agent may make opportunistic queries (for models $m \notin M_A$) that are not relevant to the current task.

Due to these differences, this setting provides challenges not seen in most active learning scenarios:

- Since the agent never sees the entire pool of unlabeled examples, it can neither choose queries that are globally optimal, nor use variance reduction strategies that still use near-optimal queries (such as sampling from a beam of near globally optimal queries).

- Since the agent is evaluated on task completion, it must learn to trade-off finishing the task with querying to improve the models.
- The agent needs to estimate the usefulness of a model across multiple interactions, to identify good opportunistic queries.

4 Task Setup

We consider an interactive task where an agent tries to learn to ground natural-language object descriptions. Grounded language understanding is the process of mapping natural-language referring expressions to object referents in the world (Thomason et al., 2016). We consider a grounded-language problem based on object retrieval – given a free form natural-language description of an object, the agent needs to identify which of a set of objects is best described by the phrase (Thomason et al., 2016; Guadarrama et al., 2014). In this work, objects are presented as images, but the methods are applicable to any feature representation of objects. We consider a task of interactive object retrieval where the agent is given a natural-language object description, and allowed to interact with the user before it attempts to guess the object being referred to.

In each interaction, the agent is presented with two sets of objects. The first set of objects is called the active training set, and is to be used by the agent to improve its model of object properties. The second set of objects is called the active test set, and the agent will have to retrieve an object from this set. The agent is provided with a natural language description of the object it is expected to retrieve.

Before guessing, the agent is allowed to ask queries of the following two types:

- Label queries - A yes/no question about whether a predicate can be used to describe one of the objects in the active training set, e.g. "Is this object yellow?".
- Example queries - Asking for an object, in the available training set, that can be described by a particular predicate, e.g. "Show me a white object in this set.". This is used for acquiring positive examples since most predicates tend to be sparse. [1]

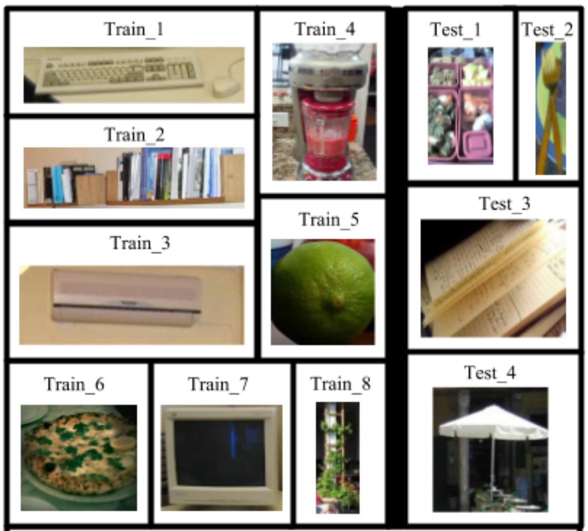

Oracle	*Target Description*	A **white umbrella**
Robot	*Label Query*	<Train_6, **yellow**>
Oracle	*Binary Label*	0
Robot	*Example Query*	<**white**>
Oracle	*Image*	Train_1
Robot	*Guess*	Test_4
Oracle	*Success (0/1)*	Correct

Figure 1: A sample OAL interaction. Perceptual predicates are marked in bold.

A sample interaction is shown in Figure 1. The agent goes through a series of such interactions, and needs to learn to maximize the number of correct guesses across interactions, without frustrating the user with too many queries. The separate active training set and active test set ensures that the agent needs to learn models for object descriptors. If queries and guessing were performed on the same set of objects, the agent could simply query whether each specific object satisfies each predicate in the description, and use this to guess.

In our experiments, we simulate such dialogs using the Visual Genome dataset (Krishna et al., 2017); which contains images with regions (crops) annotated with natural-language descriptions. Bounding boxes of objects present in the image are also annotated, along with attributes of objects. Region descriptions, objects and attributes are annotated using unrestricted natural language, which leads to a diverse set of predicates. Using the annotations, we can associate a

---

[1] Alternately, we could return all positive examples for the predicate in the active training set, but we chose to return a single example to allow the agent to minimize the amount of supervision obtained

list of objects and attributes relevant to each image region, and use these to answer queries from the agent.

For each interaction, we uniformly sample 4 regions to form the active test set, and 8 regions to form the active training set. [2] One region is then uniformly sampled from the active test set to be the target object. Its description, from annotations in the Visual Genome dataset, is provided to the agent to be grounded. The objects and attributes associated with active training regions are used to answer queries. A predicate is labeled as being applicable to a region if it is present in the list of objects and attributes associated with the region. In the rest of the paper, we use the terms object, image, and region interchangeably.

5 Methodology

5.1 Perceptual Predicates and Classifiers

We assume that the description provided is a conjunction of one-word predicates. Given a description, the agent tokenizes it and removes stopwords. Each remaining word is stemmed and treated as a perceptual predicate. This method allows the agent to learn an open vocabulary of predicates, but unable to handle multi-word predicates or non-compositional phrases.

The agent learns a separate binary classifier for each predicate, and we represent images with a "deep" feature representation obtained from the penultimate layer of the VGG network (Simonyan and Zisserman, 2014) pretrained on ImageNet (Russakovsky et al., 2015). The agent has no initial classifiers for any predicate, and learns these classifiers purely from labels acquired during interactions.

5.2 Grounding Descriptions

The learned perceptual classifiers are used to ground natural language descriptions as follows. Let $p_1, p_2, \ldots p_k$ be the predicates obtained from the natural language description. Let $d(p_i, o) \in \{-1, 1\}$ be the decision from the classifier for predicate p_i for object o, and $C(p_i)$ be the estimated F1 of the classifier for p_i. [3] Then the best guess, from the objects present, is chosen using the weighted sum of the decisions of the classifiers, using their estimated F1 as a weight:

$$o_{guess} = \text{argmax}_{o \in O_A} \sum_{i=1}^{k} d(p_i, o) * C(p_i)$$

5.3 MDP Formulation

We model the task as a Markov Decision Process (MDP). An MDP is a tuple $\langle S, A, T, R, \gamma \rangle$, where S is a set of states, A is a set of actions, T is a transition function, R is a reward function and γ is a discount factor. Each interaction is an episode in the MDP. At any point, the agent is in a state $s \in S$, in our case consisting of the VGG features of the images in the current interaction, the predicates in the current description, and the agent's classifiers. The agent can choose from among actions in A, which include an action for guessing, and an action for each possible query the agent can make, including both label and example queries. The guess action always terminates the episode, and query actions transition the agent to a state $s' \in S$ as one of the classifiers gets updated. The agent gets a reward for each action taken. Query actions have a small negative reward, and guessing results is a large positive reward when the guess is correct, and a large negative reward when the guess is incorrect. In our experiments, we treat the reward values as hyperparameters that can be tuned. The best results were obtained with a reward of 200 for a correct guess, -100 for an incorrect guess and -1 for each query.

5.4 Identifying Candidate Queries

In any interaction, the agent can make label or example queries. In a label query, the agent can ask for the label of any object for a specific predicate. If O_A is the set of objects present in the active training set of the current interaction, and P is the set of predicates that have been seen by the agent in all interactions so far, then the set of possible label queries is $P \times O_A$. Once the agent chooses a predicate p and object o to be queried, it obtains the corresponding label and can update its classifier for p. In an example query, the agent asks for a positive example for any predicate $p \in P$. The agent will either receive a positive label for p for some object $o \in O_A$ or learn that the label is negative $\forall\ o \in O_A$, and can appropriately update the classifier for p.

---

[2] The regions in the dataset are divided into separate pools from which the active training and active test sets are sampled (described as classifier-training and classifier-test sets in section 6.2), to ensure that the agent needs to learn classifiers that generalize across objects.

[3] F1 is estimated by cross-validation on the labels acquired for the predicate.

Since $|P|$ grows across interactions as the agent encounters more predicates in descriptions, the number of candidate actions in a state increases over time, so searching the entire space of possible queries can become intractable. Hence, we identify a beam of promising queries that are then provided as candidate actions for the policy to choose among. Uncertainty sampling is a common method in pool-based active learning to identify the best example to improve a classifier. For a given predicate p, we use this to choose the best label query involving that predicate, picking that object $o \in O_A$ which is closest to the hyperplane of the classifier for p.

However, it is more challenging to narrow down the number of predicates. Thomason et al. (2017) assume that an estimate of classifier accuracy is available, which is comparable across classifiers. They sample predicates with a probability inversely proportional to the estimated accuracy of the classifier. However, if the space of possible predicates is large, then this results in no classifier obtaining a reasonable number of training examples. In this scenario, it is desirable to focus on a small number of predicates, possibly stopping the improvement on a predicate once the classifier for it has been sufficiently improved. We sample queries from a distribution designed to capture this intuition. The probability assigned to a predicate by this distribution increases linearly, for estimated F1 below a threshold, and decreases linearly thereafter. [4] The number of queries sampled is a hyperparameter. We obtain the best results by sampling 3 queries of each type.

5.5 Baseline Static Policy

As a baseline, we use a static policy similar to that used by Thomason et al. (2017). At each state, a single label query and example query are sampled. The agent asks a fixed number of queries before guessing. Thomason et al. (2017) use thresholds that prevent queries from being asked when there are no predicates whose classifiers have sufficiently low estimated accuracy. Since we used a dataset with a much larger number of predicates, these thresholds were always crossed if the agent had even one candidate query.

5.6 Policy Learning

We use the REINFORCE algorithm (Williams, 1992) to learn a policy for the MDP. The agent learns a policy $\pi(a|s;\theta)$, parameterized with weights θ that computes the probability of taking action a in state s. Given a feature representation $f(s,a)$ for a state-action pair (s,a), the policy is of the form:

$$\pi(a|s;\theta) = \frac{e^{\theta^T f(s,a)}}{\sum_{a'} e^{\theta^T f(s,a')}}$$

where the denominator is a sum over all actions possible in state s. The weights are updated using a stochastic gradient ascent rule:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

where $J(\theta)$ is the expected return from the policy according to the distribution over trajectories induced by the policy.

The state consists of the predicates in the current description, the candidate objects, and the current classifiers. Since both the number of candidate objects and classifiers varies, and the latter is quite large, it is necessary to identify useful features for the task to obtain a vector representation needed by most learning algorithms. In our problem setting, the number of candidate actions available to the agent in a given state is variable. Hence we need to create features for state-action pairs, rather than just states.

5.7 Features for Policy Learning

The object retrieval task consists of two parts – identifying useful queries to improve classifiers, and correctly guessing the image being referred to by a given description. The current dialog length is also provided to influence the trade-off between guessing and querying.

5.7.1 Guess-success features

Let $P_A = \{p_1, p_2, \dots p_k\}$ be the predicates extracted from the current description. For each predicate $p \in P_A$, we have the estimated F1 of the classifier $C(p)$, and for each object o in the active test set, we have a decision $d(p,o) \in \{-1, 1\}$ from the classifier. We refer to $s(p,o) = d(p,o) * C(p)$ as the score of the classifier of p for object o. The following features are used to predict whether the current best guess is likely to be correct:

- Lowest, highest, second highest, and average estimated F1 among classifiers of predicates

---

[4] The equation for this distribution with some further discussion on its design is included in the supplementary material.

in P_A – learned thresholds on these values can be useful to decide whether to trust the guess.

- Highest score among regions in the active test set, and the differences between this and the second highest, and average scores respectively – a good guess is expected to have a high score to indicate relevance to the description, and substantial differences would indicate that the guess is discriminative. Similar features are also formed using the unweighted sum of decisions.
- An indicator of whether the two most confident classifiers agree on the decision of the top scoring region, which increases the likelihood of its being correct.

We compared directly using these features to training a regressor that uses them to predict the probability of a successful guess, and then using this as a higher-level policy feature. We found no difference between the two methods and the results reported directly use these features in the vector provided to the policy learner.

5.7.2 Query-evaluation features

The following features are expected to be useful in predicting whether it is useful to query for the label of a particular predicate:

- Indicator of whether the predicate is new or already has a classifier – this allows the policy to decide between strengthening existing classifiers or creating classifiers for novel predicates.
- Current estimated F1 of the classifier for the predicate – as there is more to be gained from improving a poor classifier.
- Fraction of previous dialogs in which the predicate has been used, and the agent's success rate in these – as there is more to be gained from improving a frequently used predicate but less if the agent already makes enough correct guesses for it.
- Is the query opportunistic – as these will not help the current guess.

Label queries also have an image region specified, and for these we have additional features that use the VGG feature space in which the region is represented for classification:

- Margin of the image region from the hyperplane of the classifier of the predicate – motivated by uncertainty sampling.
- Average cosine distance of the image region to others in the dataset – motivated by density weighting to avoid outliers.
- Fraction of the k-nearest neighbors of the region that are unlabeled for this predicate – motivated by density weighting to identify a data point that can influence many labels.

6 Experimental Methodology

6.1 Dataset

The Visual Genome dataset contains a total of 108,077 images with 540,6592 annotated regions. Since objects and attributes are annotated with free-form text rather than from a fixed, pre-defined vocabulary, there is considerable diversity in the language used for annotation. There are 80,908 unique objects annotated and 44,235 attributes. We assume that any objects that partially overlap with a region are present in it, as these are usually used in descriptions. Using the annotations, we can associate a list of objects and attributes relevant to each image region. We lower-case all annotations, remove special characters and perform stemming to help normalize terms.

6.2 Sampling dialogs

We want the agent to learn a policy that is independent of the actual predicates present at policy training and policy test time. In order to be able to evaluate this, we divide the set of possible regions into policy training and policy test regions as follows. We select all objects and attributes present in at least 1,000 regions. Half of these were randomly assigned to the policy test set. All regions that contain one of these objects or attributes are assigned to the policy test set, and the rest to the policy training set. Thus regions seen at test time may contain predicates seen during training, but will definitely contain at least one novel predicate. Further, the policy training and policy test sets are respectively partitioned into a classifier training and classifier test set using a uniform 60-40 split.

During policy training, the active training set of each dialog is sampled from the classifier-training subset of the policy-training regions, and the active test set of the dialog is sampled from the classifier-test subset of the policy-training set.

During policy testing, the active training set of each dialog is sampled from the classifier training subset of the policy test regions, and the active test set of the dialog is sampled from the classifier test subset of the policy test set.

6.3 Experiment phases

For efficiency, we run dialogs in batches, and perform classifier and policy updates at the end of each batch. We use batches of 100 dialogs each. Our experiment runs in 3 phases:

- Initialization – Since learning starting with a random policy can be difficult, we first run batches of dialogs on the policy training set using the static policy from section 5.5, and update the RL policy using states, actions and rewards seen in these dialogs. This "supervised" learning phase is used to initialize the RL policy.

- Training – We run batches of dialogs on the policy training set using the RL policy, starting it without any classifiers. In this phase, the policy is updated using its own experience.

- Testing – We fix the parameters of the RL policy, and run batches of dialogs on the policy test set. During this phase, the agent is again reset to start with no classifiers. We do this to ensure that performance improvements seen at test time are purely from learning a strategy for opportunistic active learning, not from acquiring useful classifiers in the process of learning the policy.

7 Experimental Results and Analysis

We initialize the policy with 10 batches of dialogs, and then train on another 10 batches of dialogs, both sampled from the policy training set. Following this, the policy weights are fixed, the agent is reset to start with no classifiers, and we test on 10 batches of dialogs from the policy test set. Table 1 compares the average success rate (fraction of successful dialogs in which the correct object is identified), and average dialog length (average number of system turns) of the best learned policy, and the baseline static policy on the final batch of testing. We also compare the effect of ablating the two main groups of features. The learned agent guesses correctly in a significantly higher fraction of dialogs compared to the static agent, using a significantly lower number of questions per dialog.

Policy	Success rate	Average Dialog Length
Learned	**0.44**	**12.95**
–Guess	0.37	**6.12**
–Query	0.35	**6.16**
Static	0.29	16

Table 1: Results on dialogs sampled from the policy test set after 10 batches of classifier training. *–Guess* and *–Query* are conditions with the guess and query features, respectively, ablated. Boldface indicates that the difference in that metric with respect to the *Static* policy is statistically significant according to an unpaired Welch t-test with $p < 0.05$.

When either the group of guess or query features is ablated, the success rate clearly decreases. While the mean success rate still remains above the baseline, the difference is no longer statistically significant. Further, at the end of the initialization phase, the average dialog length in all three conditions is about the same. In the two ablated conditions, the dialog length does not increase to become close to that of the static policy, which suggests that the agent does not learn that asking more queries improves dialog success. This is expected because the agent is either not able to evaluate the usefulness of queries, or the likelihood of success of a guess. However, in the learned policy with all features, the agent is able to identify a benefit in asking queries, and utilizes them to improve its success rate.

It is important to note that it is non-trivial to decide how to trade-off dialog success with dialog length. This should be decided for any given application by comparing the cost of an error with that of the user time involved in answering queries, and the reward function should be set appropriately based on this. Ideally, we would like to see an increase in dialog success rate *and* a decrease in dialog length, as is the case when comparing the learned and static policies. However, depending on the application, it may also be beneficial to see a smaller increase in success rate with a larger decrease in dialog length, as is the case in the ablated conditions.

We also explored ablating individual features. We found that the effect of ablating most single features is similar to that of ablating a group of features. The mean success rate decreases compared to the full policy with all features. It remains better than that of the static policy, but in most cases the difference stops being statistically significant. Among features for evaluating the guess, the removal of the difference between the two highest scores in the active test set has a fairly large effect, compared with the value of the highest score. This is expected because for retrieval it is sufficient if an object is simply scored higher than the other candidates. Further, since classifiers improve over time, the score threshold that indicates a good guess changes, and hence would be difficult to learn. An interesting result is that removal of features involving the predictions of the second best classifier has more effect than that of the best classifier. This is possibly because when noisy classifiers are in use, support of multiple classifiers is helpful. Among query evaluation features, we find, unsurprisingly, that removal of the feature providing the margin of the object in a label query affects performance much more than removal of features such as density and fraction of labeled neighbors, which merely indicate whether the object is an outlier. The full results of this experiment are included in the supplementary material.

Qualitatively, we found that the dialog success rate was higher for both short, and very long dialogs, with a decrease for dialogs of intermediate length. This suggests that longer dialogs are used to accumulate labels via opportunistic off-topic questions, as opposed to on-topic questions. The learned policy still suffers from high variance in dialog length suggesting that trading off task completion against model improvement is a difficult decision to learn. We find that the labels collected by the learned policy are more equitably distributed across predicates than the static policy, resulting in a tendency to have fewer classifiers of low estimated F1. There is relatively little difference in the number of predicates for which classifiers are learned. This suggests that the policy learns to focus on a few predicates, as the baseline does, but learn all of these equally well, in contrast to the baseline which has much higher variance in the number of labels collected per predicate.

8 Future Work

It would be interesting to examine how a policy learned using a dataset such as Visual Genome generalizes to a different domain such as images captured by a robot operating in an indoor environment, possibly with some fine-tuning using a smaller in-domain dataset. The simulation could also potentially be improved using positive-unlabeled learning methods (Liu et al., 2002; Li and Liu, 2003) instead of assuming that an object or attribute not labeled in an image region is not present in the image. It would also be interesting to compare the effectiveness of the opportunistic active learning framework, as well as the policy learning, across a variety of applications.

9 Conclusion

This paper has shown how to formulate an opportunistic active learning problem as a reinforcement learning problem, and learn a policy that can effectively trade-off opportunistic active learning queries against task completion. We evaluated this approach on the task of grounded object retrieval from natural language descriptions and learn a policy that retrieves the correct object in a larger fraction of dialogs than a previously proposed static baseline, while also lowering average dialog length.

Acknowledgements

This work is supported by an NSF NRI grant (IIS-1637736). A portion of this work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1305287, IIS-1637736, IIS-1651089, IIS-1724157), TxDOT, Intel, Raytheon, and Lockheed Martin. Peter Stone serves on the Board of Directors of Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

Philip Bachman, Alessandro Sordoni, and Adam Trischler. 2017. Learning algorithms for active learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 301–310, Sydney, Australia. PMLR.

Klaus Brinker. 2006. On active learning in multi-label classification. In *From Data and Information Analysis to Knowledge Engineering*, pages 206–213. Springer-Verlag.

Maya Cakmak, Crystal Chao, and Andrea L Thomaz. 2010. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development*, 2(2):108–118.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 326 – 335.

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. ACL.

Sergio Guadarrama, Erik Rodner, Kate Saenko, Ning Zhang, Ryan Farrell, Jeff Donahue, and Trevor Darrell. 2014. Open-vocabulary object retrieval. In *Robotics: Science and Systems*, volume 2, page 6.

Thomas Kollar, Jayant Krishnamurthy, and Grant Strimel. 2013. Toward interactive grounded language acqusition. In *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Johannes Kulick, Marc Toussaint, Tobias Lang, and Manuel Lopes. 2013. Active learning for teaching a robot grounded relational symbols. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 1451–1457. AAAI Press.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR '94*, pages 3–12. Springer London.

Xachan Li, Lei Wang, and Eric Sang. Multi-label SVM active learning for image classification. In *2004 International Conference on Image Processing, 2004. ICIP'04*, volume 4, pages 2207–2210. IEEE.

Xiaoli Li and Bing Liu. 2003. Learning to classify texts using positive and unlabeled data. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 587–592, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Xin Li and Yuhong Guo. 2013. Active learning with multi-label svm classification. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 1479–1485.

Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. 2002. Partially supervised classification of text documents. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 387–394, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. ACL.

Shiwali Mohan, Aaron H Mininger, James R Kirk, and John E Laird. 2012. Acquiring grounded representations of words with situated interactive instruction. In *Advances in Cognitive Systems*.

Aishwarya Padmakumar, Jesse Thomason, and Raymond J. Mooney. 2017. Integrated learning of dialog strategies and semantic parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 547–557, Valencia, Spain.

Natalie Parde, Adam Hair, Michalis Papakostas, Konstantinos Tsiakas, Maria Dagioglou, Vangelis Karkaletsis, and Rodney D. Nielsen. 2015. Grounding the meaning of words through vision and interactive gameplay. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1895–1901, Buenos Aires, Argentina.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. 2011. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing*, 7(3):1–21.

Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, and Hong-Jiang Zhang. 2009. Two-dimensional multilabel active learning with an efficient online adaptation model for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1880–1897.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 441–448, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, volume 16, pages 3776–3784.

Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP'08*. ACL.

H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 287–294, New York, NY, USA. ACM.

Manali Sharma and Mustafa Bilgic. 2016. Evidence-based uncertainty sampling for active learning. *Data Mining and Knowledge Discovery*, 31(1):164–202.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *Computing Research Repository*, arXiv:1409.1556.

Mohan Singh, Eoin Curran, and Pádraig Cunningham. 2009. Active learning for multi-label image annotation. In *Proceedings of the 19th Irish Conference on Artificial Intelligence and Cognitive Science*, pages 173–182.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Justin Hart, Peter Stone, and Raymond J. Mooney. 2017. Opportunistic active learning for grounding natural language descriptions. pages 67–76.

Jesse Thomason, Jivko Sinapov, Maxwell Svetlik, Peter Stone, and Raymond Mooney. 2016. Learning multi-modal grounded linguistic semantics by playing "I spy". In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3477–3483.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)*, pages 406–414, Bled, Slovenia.

Deepak Vasisht, Andreas Damianou, Manik Varma, and Ashish Kapoor. 2014. Active learning for sparse bayesian multilabel classification. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD'14*, pages 472–481. ACM Press.

Adam Vogel, Karthik Raghunathan, and Dan Jurafsky. 2010. Eye spy: Improving vision through dialog. In *Association for the Advancement of Artificial Intelligence*, pages 175–176.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. ACL.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer US.

Mark Woodward and Chelsea Finn. 2017. Active one-shot learning. *Computing Research Repository*, arXiv:1702.06559.

Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. 2009. Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD'09*. ACM Press.

Yanchao Yu, Arash Eshghi, and Oliver Lemon. 2017. Learning how to learn: An adaptive dialogue agent for incrementally learning visually grounded word meanings. In *Proceedings of the First Workshop on Language Grounding for Robotics*. ACL.

RecipeQA: A Challenge Dataset for Multimodal Comprehension of Cooking Recipes

Semih Yagcioglu, Aykut Erdem, Erkut Erdem and Nazli Ikizler-Cinbis
Hacettepe University Computer Vision Lab
Dept. of Computer Engineering, Hacettepe University, Ankara, TURKEY
semih.yagcioglu@hacettepe.edu.tr,{aykut,erkut,nazli}@cs.hacettepe.edu.tr

Abstract

Understanding and reasoning about cooking recipes is a fruitful research direction towards enabling machines to interpret procedural text. In this work, we introduce RecipeQA, a dataset for multimodal comprehension of cooking recipes. It comprises of approximately 20K instructional recipes with multiple modalities such as titles, descriptions and aligned set of images. With over 36K automatically generated question-answer pairs, we design a set of comprehension and reasoning tasks that require joint understanding of images and text, capturing the temporal flow of events and making sense of procedural knowledge. Our preliminary results indicate that RecipeQA will serve as a challenging test bed and an ideal benchmark for evaluating machine comprehension systems. The data and leaderboard are available at http://hucvl.github.io/recipeqa.

1 Introduction

There is a rich literature in natural language processing (NLP) and information retrieval on question answering (QA) (Hirschman and Gaizauskas, 2001), but recently deep learning has sparked interest in a special kind of QA, commonly referred to as reading comprehension (RC) (Vanderwende, 2007). The aim in RC research is to build intelligent systems with the abilities to read and understand natural language text and answer questions related to it (Burges, 2013). Such tests are appealing as they require joint understanding of the question and the related passage (*i.e.* context), and moreover, they can analyze many different types of skills in a rather objective way (Sugawara et al., 2017).

Despite the progress made in recent years, there is still a significant performance gap between humans and deep neural models in RC, and researchers are pushing forward our understanding of the limitations and capabilities of these approaches by introducing new datasets. Existing tasks for RC mainly differ in two major respects: the question-answer formats, *e.g.* cloze (fill-in-the-blank), span selection or multiple choice, and the text sources they use, such as news articles (Hermann et al., 2015; Trischler et al., 2017), fictional stories (Hill et al., 2016), Wikipedia articles (Kočiský et al., 2018; Hewlett et al., 2016; Rajpurkar et al., 2016) or other web sources (Joshi et al., 2017). A popular topic in computer vision closely related to RC is Visual Question Answering (VQA) in which context takes the form of an image in the comprehension task, where recent datasets have also been compiled, such as (Antol et al., 2015; Yu et al., 2015; Johnson et al., 2017; Goyal et al., 2017), to name a few.

More recently, research in QA has been extended to focus on the multimodal aspects of the problem where different modalities are being explored. Tapaswi et al. (2016) introduced MovieQA where they concentrate on evaluating automatic story comprehension from both video and text. In COMICS, Iyyer et al. (2017) turned to comic books to test understanding of closure, transitions in the narrative from one panel to the next. In AI2D (Kembhavi et al., 2016) and FigureQA (Kahou et al., 2018), the authors addressed comprehension of scientific diagrams and graphical plots. Last but not least, Kembhavi et al. (2017) has proposed another comprehensive and challenging dataset named TQA, which comprised of middle school science lessons of diagrams and texts.

In this study, we focus on *multimodal machine comprehension of cooking recipes* with images and text. To this end, we introduce a new QA dataset called *RecipeQA* that consists of recipe instructions and related questions (see Fig. 1 for an example text cloze style question). There are a handful of reasons why understanding and reasoning about

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1358–1368
Brussels, Belgium, October 31 - November 4, 2018.

Text Cloze Style Question **Context Modalities: Images and Descriptions of Steps**

Recipe: Last-Minute Lasagna

1. Heat oven to 375 degrees F. Spoon a thin layer of sauce over the bottom of a 9-by-13-inch baking dish.
2. Cover with a single layer of ravioli.
3. Top with half the spinach half the mozzarella and a third of the remaining sauce.
4. Repeat with another layer of ravioli and the remaining spinach mozzarella and half the remaining sauce.
5. Top with another layer of ravioli and the remaining sauce not all the ravioli may be needed. Sprinkle with the Parmesan.
6. Cover with foil and bake for 30 minutes. Uncover and bake until bubbly, 5 to 10 minutes.
7. Let cool 5 minutes before spooning onto individual plates.

Question	Choose the best text for the missing blank to correctly complete the recipe Cover. _____________. Bake. Cool, serve.
Answer	**A. Top, sprinkle** B. Finishing touches C. Layer it up D. Ravioli bonus round

Figure 1: An illustrative text cloze style question (context, question and answer triplet). The context is comprised of recipe description and images where the question is generated using the question titles. Each paragraph in the context is taken from another step, as also true for the images. Bold answer is the correct one.

recipes is interesting. Recipes are written with a specific goal in mind, that is to teach others how to prepare a particular food. Hence, they contain immensely rich information about the real world. Recipes consist of instructions, wherein one needs to follow each instruction to successfully complete the recipe. As a classical example in introductory programming classes, each recipe might be seen as a particular way of solving a task and in that regard can also be considered as an algorithm. We believe that recipe comprehension is an elusive challenge and might be seen as important milestone in the long-standing goal of artificial intelligence and machine reasoning (Norvig, 1987; Bottou, 2014).

Among previous efforts towards multimodal machine comprehension (Tapaswi et al., 2016; Kembhavi et al., 2016; Iyyer et al., 2017; Kembhavi et al., 2017; Kahou et al., 2018), our study is closer to what Kembhavi et al. (2017) envisioned in TQA. Our task primarily differs in utilizing substantially larger number of images – the average number of images per recipe in RecipeQA is 12 whereas TQA has only 3 images per question on average. Moreover, in our case, each image is aligned with the text of a particular step in the corresponding recipe. Another important difference is that TQA contains mostly diagrams or textbook images whereas RecipeQA consists of natural images taken by users in unconstrained environments.

Some of the important characteristics of RecipeQA are as follows:

- There are arbitrary numbers of steps in recipes and images in steps, respectively.
- There are different question styles, each requiring a specific comprehension skill.
- There exists high lexical and syntactic divergence between contexts, questions and answers.
- Answers require understanding procedural language, in particular keeping track of entities and/or actions and their state changes.
- Answers may need information coming from multiple steps (*i.e.* multiple images and multiple paragraphs).
- Answers inherently involve multimodal understanding of image(s) and text.

To sum up, we believe RecipeQA is a challenging benchmark dataset which will serve as a test bed for evaluating multimodal comprehension systems. In this paper, we present several statistical analyses on RecipeQA and also obtain baseline performances for a number of multimodal comprehension tasks that we introduce for cooking recipes.

2 RecipeQA Dataset

The Recipe Question Answering (RecipeQA) dataset is a challenging multimodal dataset that evaluates reasoning over real-life cooking recipes. It consists of approximately 20K recipes from 22 food categories, and over 36K questions. Fig. 2 shows an illustrative cooking recipe from our dataset. Each recipe includes an arbitrary number of steps containing both textual and visual elements. In particular, each step of a recipe is accompanied by a 'title', a 'description' and a set of illustrative 'images' that are aligned with the title and the description. Each of these elements can be considered as a different modality of the data. The questions in RecipeQA explore the multimodal aspects of the step-by-step instructions available in the recipes through a number of specific tasks that are described in Sec. 3, namely *textual cloze*, *visual cloze*, *visual coherency* and *visual ordering*.

2.1 Data Collection

We consider cooking recipes as the main data source for our dataset. These recipes were collected from Instructables[1], which is a how-to web site where users share all kinds of instructions including but not limited to recipes.

We employed a set of heuristics that helped us collect high quality data in an automatic manner. For instance, while collecting the recipes, we downloaded only the most popular recipes by considering the popularity as an objective measure for assessing the quality of a recipe. Our assumption is that the mostly viewed recipes contain less noise and include easy-to-understand instructions with high-quality illustrative images.

In total, we collected about 20K unique recipes from the food category of Instructables. We filtered out non-English recipes using a language identification (Lui and Baldwin, 2012), and automatically removed the ones with unreadable contents such as the ones that only contain recipe videos. Finally, as a post processing step, we normalized the description text by removing non-ASCII characters from the text.

2.2 Questions and Answers

For machine comprehension and reasoning, forming the questions and the answers is crucial for evaluating the ability of a model in understanding the content. Prior studies employed natural language questions either collected via crowdsourcing platforms such as SQuAD (Rajpurkar et al., 2016) or generated synthetically as in CNN/Daily Mail (Hermann et al., 2015). Using natural language questions is a good approach in terms of capturing human understanding, but crowdsourcing is often too costly and does not scale well as the size of the dataset grows. Synthetic question generation is a low-cost solution, but the quality of the generated questions is subject to question.

RecipeQA includes structured data about the cooking recipes that consists of step-by-step instructions, which helps us generate questions in a fully automatic manner without compromising the quality. Our questions test the semantics of the instructions of the recipes from different aspects through the tasks described in Sec. 3. In particular, we generate a set of multiple choice questions (the number of choices is fixed as four) by following a simple procedure which apply to all of our tasks with slight modifications.

In order to generate question-answer-context triplets, we first filtered out recipes that contain less than 3 steps or more than 25 steps. We also ignored the initial step of the recipes as our preliminary analysis showed that the first step of the recipes almost always is used by the authors to provide a narrative, *e.g.* why they love making that particular food, or how it makes sense to prepare a food for some occasion, and often is not relevant to the recipe instructions. In addition, we automatically removed some indicators such as step numbers that explicitly emphasize temporal order from the step titles while generating questions.

Given a task, we first randomly select a set of steps from each recipe and construct our questions and answers from these steps according to the task at hand. In particular, we employ the modality that the comprehension task is built upon to generate the candidate answers and use the remaining content as the necessary context for our questions. For instance, if the step titles are used within the candidate answers, the context becomes the descriptions and the images of the steps. As the average number of steps per recipe is larger than four, using this strategy, we can generate multiple context-question-answer triplets from a single recipe.

Candidate answers can be generated by selecting the distractors at random from the steps of other recipes. To make our dataset more challenging, we

---

[1] All materials from the *instructables.com* were downloaded in April 2018.

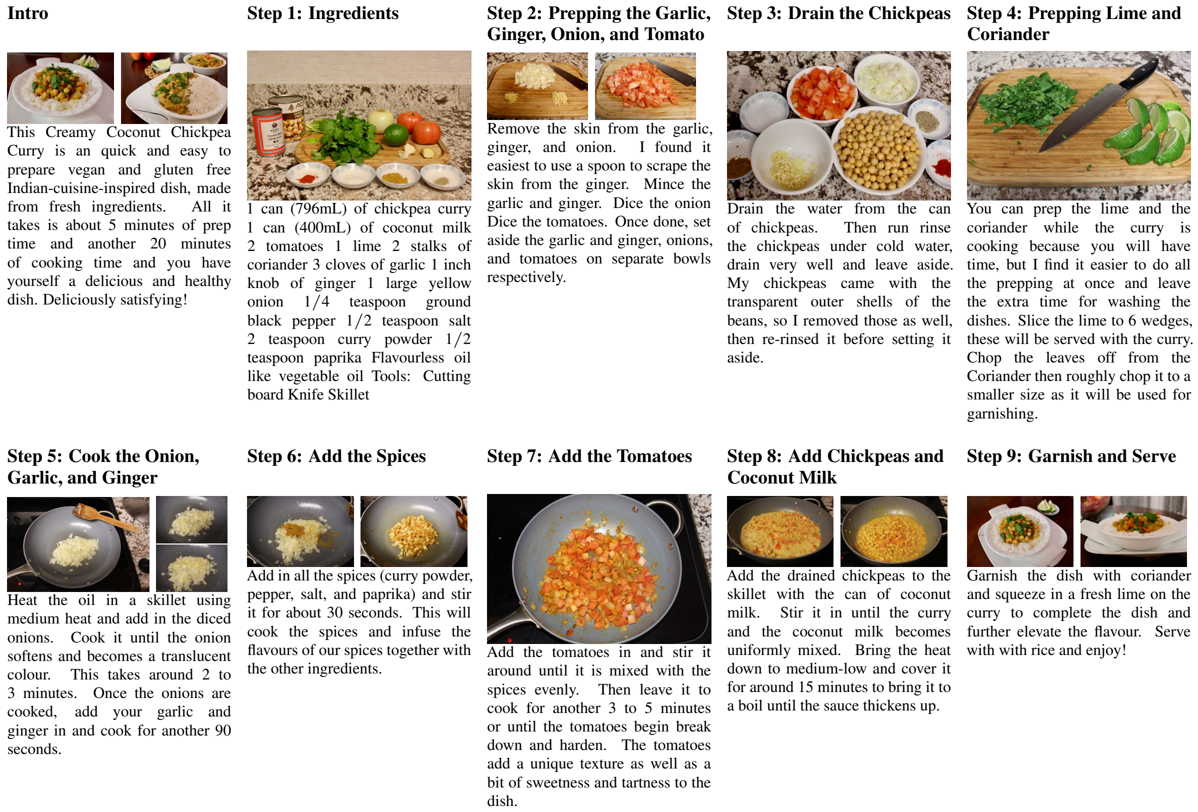

Intro

This Creamy Coconut Chickpea Curry is an quick and easy to prepare vegan and gluten free Indian-cuisine-inspired dish, made from fresh ingredients. All it takes is about 5 minutes of prep time and another 20 minutes of cooking time and you have yourself a delicious and healthy dish. Deliciously satisfying!

Step 1: Ingredients

1 can (796mL) of chickpea curry 1 can (400mL) of coconut milk 2 tomatoes 1 lime 2 stalks of coriander 3 cloves of garlic 1 inch knob of ginger 1 large yellow onion 1/4 teaspoon ground black pepper 1/2 teaspoon salt 2 teaspoon curry powder 1/2 teaspoon paprika Flavourless oil like vegetable oil Tools: Cutting board Knife Skillet

Step 2: Prepping the Garlic, Ginger, Onion, and Tomato

Remove the skin from the garlic, ginger, and onion. I found it easiest to use a spoon to scrape the skin from the ginger. Mince the garlic and ginger. Dice the onion Dice the tomatoes. Once done, set aside the garlic and ginger, onions, and tomatoes on separate bowls respectively.

Step 3: Drain the Chickpeas

Drain the water from the can of chickpeas. Then run rinse the chickpeas under cold water, drain very well and leave aside. My chickpeas came with the transparent outer shells of the beans, so I removed those as well, then re-rinsed it before setting it aside.

Step 4: Prepping Lime and Coriander

You can prep the lime and the coriander while the curry is cooking because you will have time, but I find it easier to do all the prepping at once and leave the extra time for washing the dishes. Slice the lime to 6 wedges, these will be served with the curry. Chop the leaves off from the Coriander then roughly chop it to a smaller size as it will be used for garnishing.

Step 5: Cook the Onion, Garlic, and Ginger

Heat the oil in a skillet using medium heat and add in the diced onions. Cook it until the onion softens and becomes a translucent colour. This takes around 2 to 3 minutes. Once the onions are cooked, add your garlic and ginger in and cook for another 90 seconds.

Step 6: Add the Spices

Add in all the spices (curry powder, pepper, salt, and paprika) and stir it for about 30 seconds. This will cook the spices and infuse the flavours of our spices together with the other ingredients.

Step 7: Add the Tomatoes

Add the tomatoes in and stir it around until it is mixed with the spices evenly. Then leave it to cook for another 3 to 5 minutes or until the tomatoes begin break down and harden. The tomatoes add a unique texture as well as a bit of sweetness and tartness to the dish.

Step 8: Add Chickpeas and Coconut Milk

Add the drained chickpeas to the skillet with the can of coconut milk. Stir it in until the curry and the coconut milk becomes uniformly mixed. Bring the heat down to medium-low and cover it for around 15 minutes to bring it to a boil until the sauce thickens up.

Step 9: Garnish and Serve

Garnish the dish with coriander and squeeze in a fresh lime on the curry to complete the dish and further elevate the flavour. Serve with with rice and enjoy!

Figure 2: A recipe of 'Creamy Coconut Chickpea Curry' with 9 steps, taken from Instructables.

employ a different strategy and select the distractors from the relevant modalities (titles, descriptions or images), which are not too far or too close from the correct answer. Specifically, we employ the following simple heuristic. We first find k nearest neighbors ($k = 100$) from other recipes. We then define an adaptive neighborhood by finding the closest distance to the query and remove the candidates that are too close. The remaining candidates are similar enough to be adversarial but not too similar to semantically substitute for the groundtruth. Finally, we randomly sample distractors from that pool. Details of the question generation procedure for each of the tasks are given in Sec. 3.

2.3 Dataset Statistics

RecipeQA dataset contains approximately 20K cooking recipes and over 36K question-answer pairs divided into four major question types reflecting each of the task at hand. The data is split into non-overlapping training, validation and test sets so that one set does not include a recipe and/or questions about that recipe which are available in other sets. There are 22 different food categories across our dataset whose distribution is shown in Fig. 3. While splitting the recipes into sets, we take into account these categories so that all the sets have a similar distribution of recipes across all the categories. In Table 1, we show the detailed statistics about our RecipeQA dataset. Moreover, to visualize the token frequencies, we also provide the word clouds of the titles and the descriptions from the recipes in Fig. 4.

	train	**valid**	**test**
# of recipes	15847	1963	1969
. . . avg. # of steps	5.99	6.01	6.00
. . . avg. # of tokens (titles)	17.79	17.40	17.67
. . . avg. # of tokens (descr.)	443.01	440.51	435.33
. . . avg. # of images	12.67	12.74	12.65
# of question-answers	29657	3562	3567
. . . textual cloze	7837	961	963
. . . visual cloze	7144	842	848
. . . visual coherence	7118	830	851
. . . visual ordering	7558	929	905

Table 1: RecipeQA dataset statistics.

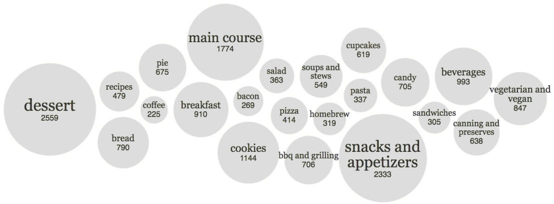

Figure 3: Distribution of the food categories across the RecipeQA.

3 Tasks

RecipeQA includes four different types of tasks: (1) Textual cloze, (2) Visual cloze, (3) Visual coherence, and (4) Visual ordering. Each of these tasks requires different reasoning skills as discussed in (Sugawara et al., 2017), and considers different modalities in their contexts and candidate answer sets. By modalities, we refer to the following pieces of information: (i) titles of steps, (ii) descriptions of steps and (iii) illustrative images of steps. While generating the questions for these tasks, we rather employ fixed templates as will be discussed below, which helps us to automatically construct question-answers pairs from the recipes with no human intervention. Using these tasks, we can easily evaluate complex relationships between different steps of a recipe via their titles, their descriptions and/or their illustrative images. Hence, our question-answers pairs are multimodal in nature. In the following, we provide a detailed description of each one of these tasks and discuss our strategies while selecting candidate answers.

3.1 Textual Cloze

Textual cloze style questions test the ability to infer missing text either in the title or in the step description by taking into account the question's context which includes a set of illustrative images besides text. While generating the question-answer pairs for this task, we randomly select a step from the candidate steps of a given recipe, hide its title and description, and ask for identifying this text amongst the multiple choices from the remaining modalities. To construct the distractor answers, we use the strategy in Sec. 2.2 that depends on the WMD (Kusner et al., 2015) distance measure. In Fig. 1, we provide a sample text cloze question from RecipeQA generated automatically in this way.

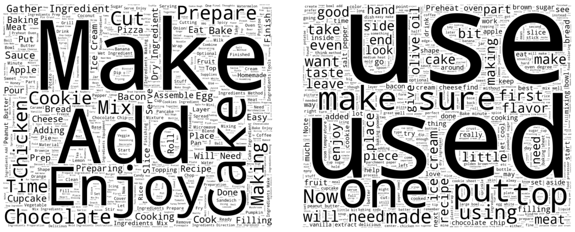

Step titles

Step descriptions

Figure 4: Word clouds of the tokens for the titles and the descriptions of the recipes from RecipeQA.

3.2 Visual Cloze

Visual cloze style questions test a skill similar to that of textual cloze task with the difference that the missing information in this task reside in the visual domain. Here, just like the textual cloze task, for a recipe we randomly select a step, hide its representative image, and ask to infer this image amongst the multiple choices. The context for this task is all textual and is in the form of a sequence of titles and descriptions. To construct the distractor images, we use Euclidean distances of 2048-d *pool5* features extracted from a ResNet-50 (He et al., 2016) pre-trained on ImageNet classification task. We show a sample visual cloze style question in Fig. 5 (second row).

3.3 Visual Coherence

Visual coherence style questions test the capability to identify an incoherent image in an ordered set of images given the titles and descriptions of the corresponding recipe as the context. Hence, to be successful at this task, a system needs to not only understand the relations between candidate steps, but also align and relate different modalities existing in the context and the answers. While generating the answer candidates for this task, we randomly select a single representative image from a single step and replace this image with a distractor image via employing the distractor selection strategy used for visual cloze task. In Fig. 5 (third row), we provide a sample visual coherence style question from RecipeQA.

3.4 Visual Ordering

Visual ordering questions test the ability of a system in finding a correctly ordered sequence given a jumbled set of representative images of a recipe. As in the previous visual tasks, the context of this task consists of the titles and descriptions of a recipe. To

Context Modalities: Titles and Descriptions of Steps

Recipe: Bacon Sushi

Step 1: **What You'll Need** This recipe makes enough bacon sushi to feed 2 - 4 people. 2 x 500g(1 lb.) packages of bacon (I chose an applewood smoked bacon, but any type would work). 3 tbsp. oil. 1 medium onion, finely diced. 1 1…

Step 2: **Cooking the Bacon** The bacon "nori" will have to be partially cooked before it can be rolled with the risotto filling. Preheat the oven to 350 degrees F. Lay half a package of bacon on the rack of the roasting pan, then bak…

Step 3: **Making the Risotto Filling** I once made risotto with sushi rice, since I had no Arborio rice on hand, and I decided that the starchiness was similar in the two. My experiment was a success, and the resulting dish was just as deli…

Step 4: **Jazzing Up the Risotto** Risotto is a wonderfully customizable dish, and a quick search on the internet will result in a multitude of variations. Here are two of my favorites: Asian mushroom risotto. 1 tbsp. oil. 1 package…

Step 5: **Rolling the Sushi** Cover the sushi rolling mat with a large piece of aluminum foil as protection from the risotto and bacon grease. (You don't want your next sushi dinner tasting like bacon. Or maybe you do...) Lay the stri…

Step 6: **Baking and Slicing** Preheat the oven to 350 degrees F. Place the aluminum foil-covered sushi rolls in the oven and bake for 20 minutes. This will warm all the ingredients and crisp the bacon a little more. It will also melt a…

Step 7: **And You're Done!** Serve the sushi with a light crispy vegetable side dish, such as refreshing cucumber sticks, or a green salad. White wine makes an excellent compliment to the meal, especially if it is the same wine used in …

Visual Cloze Style Question

Question Choose the best image for the missing blank to correctly complete the recipe

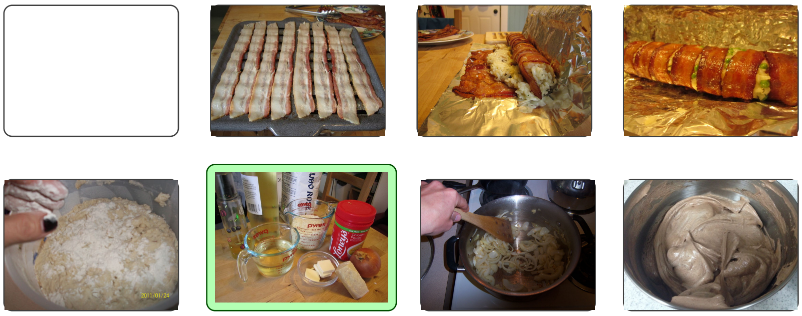

Answer A. **B.** C. D.

Visual Coherence Style Question

Question Select the incoherent image in the following sequence of images

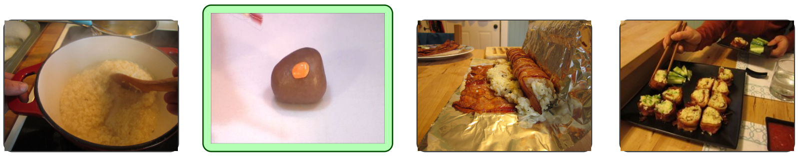

Answer A. **B.** C. D.

Visual Ordering Style Question

Question Choose the correct order of the images to make a complete recipe

(i) (ii) (iii) (iv)

Answer A. (iv)-(iii)-(ii)-(i) **B. (iv)-(iii)-(i)-(ii)** C. (i)-(ii)-(iii)-(iv) D. (ii)-(iv)-(i)-(iii)

Figure 5: Sample visual cloze, visual coherence and visual ordering style questions (context, question and answer triplet) taken from the RecipeQA training set (Question Ids: 2000-3708-0-1-4-5, 3000-3708-2-3-4-6, 4000-3708-1-2-3-6). Here, the context is comprised of step titles and descriptions where the questions are generated using the images in the recipe. The correct answers are shown with green frames or in bold.

successfully complete this task, the system needs to understand the temporal occurrence of a sequence of recipe steps and infer temporal relations between candidates, *i.e.* boiling the water first, putting the spaghetti next, so that the ordered sequence of images aligns with the given recipe. To generate answer choices, we simply use random permutations of the illustrative images in the recipe steps. In

Fig. 5 (last row), we illustrate this visual ordering task through an example question. Here, we should note that a similar task has been previously investigated by Agrawal et al. (2016) for visual stories where the task is to order a jumbled set of aligned image-description pairs.

4 Experiments

4.1 Data Preparation

Ingredient Detection. We employed the method proposed in (Salvador et al., 2017) to detect recipe ingredients. To learn more effective word embeddings, we transformed the ingredients with compound words such as *olive oil* into single word ingredients with a proper hyphenation as *olive_oil*.

Textual Embeddings. We trained a distributed memory model, namely Doc2Vec (Le and Mikolov, 2014) and used it to learn word level and document level embeddings while encoding the semantic similarity by taking into account the word order within the provided context. In this way, we can represent each word, sentence or paragraph by a fixed sized vector. In our experiments, we employed 100-d vectors to represent all of the textual modalities (titles and descriptions). We made sure that the embeddings encode semantically useful information by exploring nearest neighbors (see Fig. 6 for some examples.)

Query	Nearest Neighbor
Then add the green onion and garlic.	Then add the white onion, red pepper and garlic.
It will thicken some while it cools	Some cornflour to thicken.
Slowly whisk in the milk, scraping the bottom and sides with a heatproof spatula to make sure all the dry ingredients are mixed in.	Stir the dry ingredients in, incrementally, mixing on low speed and scraping with a spatula after each addition.

Figure 6: Sample nearest neighbors from the embeddings by the trained Doc2Vec model.

Visual Features. We used the final activation of the ResNet-50 (He et al., 2016) model trained on the ImageNet dataset (Russakovsky et al., 2015) to extract 2048-d dense visual representations. Then, we further utilized an autoencoder to decrease the dimension of the visual features to 100-d so that they become compatible in size with the text embeddings.

4.2 Baseline Models

Neural Baselines. For our neural baselines, we adapted the Impatient Reader model in (Hermann et al., 2015), which was originally developed only for the cloze style text comprehension questions in the CNN/Daily Mail dataset. In our implementation, we used a uni-directional stacked LSTM architecture with 3 layers, in which we feed the context of the question to the network in a sequential manner. Particularly, we preserve the temporal order of the steps of the recipe while feeding it to the neural model, by mimicking the most common reading strategy – reading from top to bottom. For the multimodal setting, since images are represented with vectors which are of the same size with the text embeddings, we also feed the images to the network in the same order they are presented in the recipe.

In order to account for different question types, we employ a modular architecture, which requires small adjustments to be made for each task. For instance, we place the candidate answers into query for the cloze style questions or remove the candidate answer from the query for the visual coherence type questions. In training our Impatient Reader baseline model, we use a cosine similarity function and employed the *hinge ranking loss* (Collobert et al., 2011) as follows:

$$L = \max\{0, M - cos(q, a_+) + cos(q, a_-)\} \quad (1)$$

where M is a scalar denoting the margin, a_+ represents the ground truth answer, and a_- corresponds to an incorrect answer which is sampled randomly from the whole answer space. For all of our experiments, we select M as 1.5 and employ a simple heuristic to prevent overfitting by following an early stopping scheme with patience set to 10 against the validation set accuracy after the initial epoch. For the optimizer, we use ADAM and set the learning rate to $1e-3$. The training took around 18 to 24 hours on GTX 1080Ti on a single GPU. We did not perform any hyperparameter tuning.

Simple Baselines. We adapt the Hasty Student model described in (Tapaswi et al., 2016), which does not consider the provided context and simply answers questions by only looking at the similarities or the dissimilarities between the elements in questions and the candidate answers.

For the textual close task, each candidate answer is compared against the titles or descriptions of

	Visual Cloze	Textual Cloze	Visual Coherence	Visual Ordering
Hasty Student	27.35	26.89	**65.80**	**40.88**
Impatient Reader (Text only)	–	28.03	–	–
Impatient Reader (Multimodal)	**27.36**	**29.07**	28.08	26.74

Table 2: Results for simple and neural models on the test set of RecipeQA dataset.

the steps by using WMD (Kusner et al., 2015) distance, where such distances are averaged. Then, the choice closest to all of the question steps is selected as the final answer. For the visual cloze task, a similar approach is carried out by considering images instead of text using deep visual features. For the visual coherence task, since the aim is to find the incoherent image among other images, the final answer is chosen as the most dissimilar one to the remaining images on average. Lastly, for the visual ordering task, first, the distances between each consecutive image pair in a candidate ordering of the jumbled image set is estimated. Then, each candidate ordering is scored based on the average of these pairwise distances and the choice with the minimum average distance is set as the final answer. In all these simple baseline models, we use the cosine distance to rank the candidates.

4.3 Baseline Results

We report the performance of the baseline models in Table 2 which indicates the ratio of correct answers against the total questions in the test. For the textual cloze, the comparison between text-only and multimodal Impatient Reader models shows that the additional visual modality helps the model to understand the question better and to provide more accurate answers. While for the cloze style questions, the Impatient Reader outperforms the Hasty student, for the visual coherence and visual ordering style questions Hasty student gives way better results. This demonstrates that better neural models are needed to be able to effectively deal with this kind of questions. Some qualitative examples are provided in the supplementary material.

5 Related Work

Question Answering has been studied extensively in the literature. With the success of deep learning approaches in question answering, comprehension and reasoning aspects of the task has attracted researchers to investigate QA as a medium to measure intelligence. Various datasets and methods have been proposed for measuring different aspects of the comprehension and reasoning problem. Each dataset has its own merits as well as weaknesses. Recently, a thorough analysis by (Chen et al., 2016) revealed that the required reasoning and inference level was quite simple for CNN/Daily Mail dataset (Hermann et al., 2015). To make reasoning task more realistic, new datasets such as SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2017), MSMARCO (Nguyen et al., 2016), CLEVR (Johnson et al., 2017), COMICS (Iyyer et al., 2017) and FigureQA (Kahou et al., 2018) have been proposed.

Dataset	#Images	#Questions	Modality
COMICS	1.2M	750K	Image/Text
MovieQA	408	14,944	Image/Video/Text
TQA	3,455	26,260	Image/Text
RecipeQA	250,730	36,786	Image/Text

Table 3: Comparison of the RecipeQA dataset to other multimodal machine comprehension datasets.

In the following, we briefly discuss the publicly available datasets that are closely related to our problem and provide an overview in Table 3.

The closest works to ours are (Iyyer et al., 2017), (Tapaswi et al., 2016) and (Kembhavi et al., 2017) where data multi-modality is the key aspect. COMICS dataset (Iyyer et al., 2017) focus on comic book narratives and explore visual cloze style questions, introducing a dataset consisting of drawings from comic books. The dataset is constructed from 4K Golden Age (1938-1954) comic books from the Digital Comics Museum and contains 1.2M panels with 2.5M textboxes. Three tasks are evaluated in this context, namely text cloze, visual cloze, character coherence. MovieQA dataset (Tapaswi et al., 2016), comprises of 15K crowdsourced questions about 408 movies. It consists of movie clips, subtitles, and snapshots, is about comprehending stories about movies. TQA dataset (Kembhavi et al., 2017), have 26K questions about 1K middle school science lessons with 3.5K images, mostly of diagrams and aims at addressing middle school knowledge acquisition using both images and text. Since the audience is middle school children, it requires limited reasoning.

RecipeQA substantially differentiates from the previous work in the following way. Our dataset consists of natural images that are taken by anonymous users in unconstrained environments, which is a major diversion from COMICS and TQA datasets.

It should also be noted that there has been a long history of research involving cooking recipes. Recent examples include parsing of recipes (Malmaud et al., 2014; Jermsurawong and Habash, 2015), aligning instructional text to videos (Malmaud et al., 2015; Sener et al., 2015), recipe text generation (Kiddon et al., 2016), learning cross-modal embeddings (Salvador et al., 2017), tracking entities and action transformations in recipes (Bosselut et al., 2018).

Finally, to our best knowledge, there is no dataset focusing on "how-to" instructions or recipes; hence, this work will be the first to serve multimodal comprehension of recipes having an arbitrary number of steps aligned with multiple images and multiple sentences.

6 Conclusion

We present RecipeQA, a dataset for multimodal comprehension of cooking recipes, which consists of roughly 20K cooking recipes with over 36K context-question-answer triplets. To our knowledge, RecipeQA is the first machine comprehension dataset that deals with understanding procedural knowledge in a multimodal setting. Each one of the four question styles in our dataset is specifically tailored to evaluate a particular skill and requires connecting the dots between different modalities. Results of our baseline models demonstrate that RecipeQA is a challenging dataset and we plan make it publicly available for other researchers to promote the development of new methods for multimodal machine comprehension. In the future, we also intend to extend the dataset by collecting natural language questions-answer pairs via crowdsourcing. We also hope that RecipeQA will serve other purposes for related research problems on cooking recipes as well.

Acknowledgments

We would like to thank our anonymous reviewers for their insightful comments and suggestions, which helped us improve the paper, Taha Sevim and Kenan Hagverdiyev for their help in building the RecipeQA challenge website, and NVIDIA Corporation for the donation of GPUs used in this research. This work was supported in part by a Hacettepe BAP fellowship (FBB-2016-11653) awarded to Erkut Erdem. Semih Yagcioglu was partly sponsored by STM A.Ş.

References

Harsh Agrawal, Arjun Chandrasekaran, Dhruv Batra, Devi Parikh, and Mohit Bansal. 2016. Sort story: Sorting jumbled images and captions into stories. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 925–931.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2425–2433.

Antoine Bosselut, Corin Ennis, Omer Levy, Ari Holtzman, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. In *International Conference on Learning Representations (ICLR)*.

Léon Bottou. 2014. From machine learning to machine reasoning - an essay. *Machine Learning*, 94(2):133–149.

Christopher JC Burges. 2013. Towards the machine comprehension of text: An essay. Technical report, Technical report, Microsoft Research Technical Report MSR-TR-2013-125.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the CNN/-Daily Mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

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

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1693–1701.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. WikiReading: A novel large-scale language understanding task over Wikipedia. In *Association for Computational Linguistics (ACL)*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations (ICLR)*.

Lynette Hirschman and Robert Gaizauskas. 2001. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300.

Mohit Iyyer, Varun Manjunatha, Anupam Guha, Yogarshi Vyas, Jordan Boyd-Graber, Hal Daumé III, and Larry Davis. 2017. The amazing mysteries of the gutter: Drawing inferences between panels in comic book narratives. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jermsak Jermsurawong and Nizar Habash. 2015. Predicting the structure of cooking recipes. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 781–786.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Association for Computational Linguistics (ACL)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Samira Ebrahimi Kahou, Adam Atkinson, Vincent Michalski, Akos Kadar, Adam Trischler, and Yoshua Bengio. 2018. FigureQA: An annotated figure dataset for visual reasoning. In *International Conference on Learning Representations (ICLR) Workshops*.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A diagram is worth a dozen images. In *European Conference on Computer Vision*, pages 235–251. Springer.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? Textbook question answering for multimodal machine comprehension. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 329—-339.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning (ICML)*, pages 957–966.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning (ICML)*, pages 1188–1196.

Marco Lui and Timothy Baldwin. 2012. `langid.py`: An off-the-shelf language identification tool. In *Association for Computational Linguistics (ACL) Demo Session*, pages 25–30.

Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. 2015. What's cookin'? Interpreting cooking videos using text, speech and vision. In *North American Association for Computational Linguistics (NAACL)*.

Jonathan Malmaud, Earl Wagner, Nancy Chang, and Kevin Murphy. 2014. Cooking with semantics. In *ACL 2014 Workshop on Semantic Parsing*, pages 33–38.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *NIPS 2016 Workshop on Cognitive Computation*.

Peter Norvig. 1987. A unified theory of inference for text understanding. Technical report, University of California at Berkeley, Berkeley, CA, USA.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet: Large scale visual recognition challenge. *International Journal of Computer Vision*, 113:211–252.

Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. 2017. Learning cross-modal embeddings for cooking recipes and food images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ozan Sener, Amir Zamir, Silvio Savarese, and Ashutosh Saxena. 2015. Unsupervised semantic parsing of video collections. In *IEEE International Conference on Computer Vision (ICCV)*.

Saku Sugawara, Hikaru Yokono, and Akiko Aizawa. 2017. Prerequisite skills for reading comprehension: Multi-perspective analysis of mctest datasets and systems. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 3089–3096.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. MovieQA: Understanding stories in movies through question-answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4631–4640.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *2nd Workshop on Representation Learning for NLP*.

Lucy Vanderwende. 2007. Answering and questioning for machine reading. In *AAAI Spring Symposium: Machine Reading*, page 91.

Licheng Yu, Eunbyung Park, Alexander C. Berg, and Tamara L Berg. 2015. Visual madlibs: Fill in the blank description generation and question answering. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2461–2469.

TVQA: Localized, Compositional Video Question Answering

Jie Lei **Licheng Yu** **Mohit Bansal** **Tamara L. Berg**
Department of Computer Science
University of North Carolina at Chapel Hill
`{jielei, licheng, mbansal, tlberg}@cs.unc.edu`

Abstract

Recent years have witnessed an increasing interest in image-based question-answering (QA) tasks. However, due to data limitations, there has been much less work on video-based QA. In this paper, we present TVQA, a large-scale video QA dataset based on 6 popular TV shows. TVQA consists of 152,545 QA pairs from 21,793 clips, spanning over 460 hours of video. Questions are designed to be compositional in nature, requiring systems to jointly localize relevant moments within a clip, comprehend subtitle-based dialogue, and recognize relevant visual concepts. We provide analyses of this new dataset as well as several baselines and a multi-stream end-to-end trainable neural network framework for the TVQA task. The dataset is publicly available at `http://tvqa.cs.unc.edu`.

1 Introduction

Now that algorithms have started to produce relevant and realistic natural language that can describe images and videos, we would like to understand what these models truly comprehend. The Visual Question Answering (VQA) task provides a nice tool for fine-grained evaluation of such multimodal algorithms. VQA systems take as input an image (or video) along with relevant natural language questions, and produce answers to those questions. By asking algorithms to answer different types of questions, ranging from object identification, counting, or appearance, to more complex questions about interactions, social relationships, or inferences about why or how something is occurring, we can evaluate different aspects of a model's multimodal semantic understanding.

As a result, several popular image-based VQA datasets have been introduced, including DAQUAR (Malinowski and Fritz, 2014), COCO-QA (Ren et al., 2015a), FM-IQA (Gao et al., 2015), Visual Madlibs (Yu et al., 2015), VQA (Antol et al., 2015), Visual7W (Zhu et al., 2016), etc. In addition, multiple video-based QA datasets have also been collected recently, e.g., MovieQA (Tapaswi et al., 2016), MovieFIB (Maharaj et al., 2017a), PororoQA (Kim et al., 2017), TGIF-QA (Jang et al., 2017), etc. However, there exist various shortcomings for each such video QA dataset. For example, MovieFIB's video clips are typically short (∼4 secs), and focused on purely visual concepts (since they were collected from audio descriptions for the visually impaired); MovieQA collected QAs based on text summaries only, making them very plot-focused and less relevant for visual information; PororoQA's video domain is cartoon-based; and TGIF-QA used predefined templates for generation on short GIFs.

With video-QA in particular, as opposed to image-QA, the video itself often comes with associated natural language in the form of (subtitle) dialogue. We argue that this is an important area to study because it reflects the real world, where people interact through language, and where many computational systems like robots or other intelligent agents will ultimately have to operate. As such, systems will need to combine information from what they see with what they hear, to pose and answer questions about what is happening.

We aim to provide a dataset that merges the best qualities from all of the previous datasets as well as focus on multimodal compositionality. In particular, we collect a new large-scale dataset that is built on natural video content with rich dynamics and realistic social interactions, where question-answer pairs are written by people observing both videos and their accompanying dialogues, encouraging the questions to require both vision and language understanding to answer. To further encourage this multimodal-QA quality, we ask people to write compositional questions consisting

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1369–1379
Brussels, Belgium, October 31 - November 4, 2018.

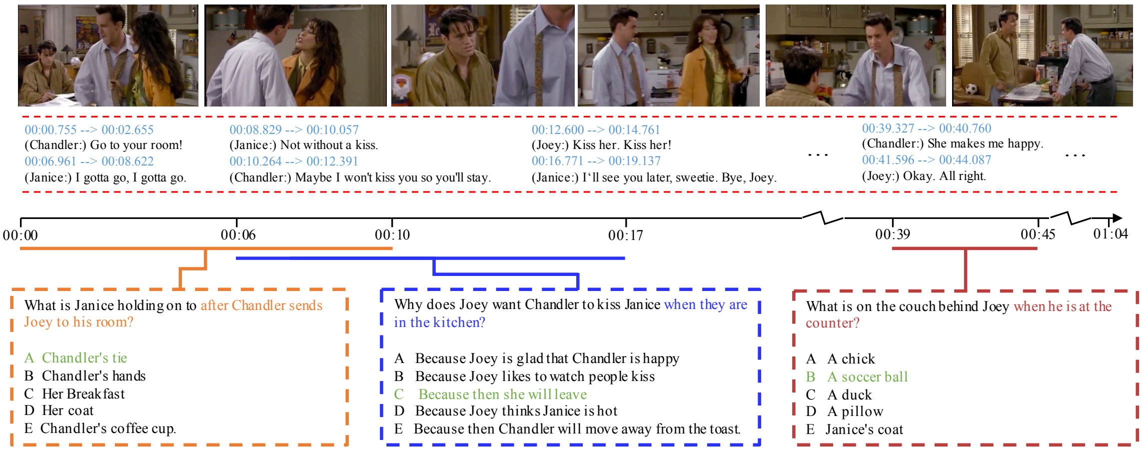

00:00.755 --> 00:02.655
(Chandler:) Go to your room!
00:06.961 --> 00:08.622
(Janice:) I gotta go, I gotta go.

00:08.829 --> 00:10.057
(Janice:) Not without a kiss.
00:10.264 --> 00:12.391
(Chandler:) Maybe I won't kiss you so you'll stay.

00:12.600 --> 00:14.761
(Joey:) Kiss her. Kiss her!
00:16.771 --> 00:19.137
(Janice:) I'll see you later, sweetie. Bye, Joey.

00:39.327 --> 00:40.760
(Chandler:) She makes me happy.
00:41.596 --> 00:44.087
(Joey:) Okay. All right.

What is Janice holding on to after Chandler sends Joey to his room?

A Chandler's tie
B Chandler's hands
C Her Breakfast
D Her coat
E Chandler's coffee cup.

Why does Joey want Chandler to kiss Janice when they are in the kitchen?

A Because Joey is glad that Chandler is happy
B Because Joey likes to watch people kiss
C Because then she will leave
D Because Joey thinks Janice is hot
E Because then Chandler will move away from the toast.

What is on the couch behind Joey when he is at the counter?

A A chick
B A soccer ball
C A duck
D A pillow
E Janice's coat

Figure 1: Examples from the TVQA dataset. All questions and answers are attached to 60-90 seconds long clips. For visualization purposes, we only show a few of the most relevant frames here. As illustrated above, some questions can be answered using subtitles or videos alone, while some require information from both modalities.

of two parts, a main question part, e.g. "What are Leonard and Sheldon arguing about" and a grounding part, e.g. "when they are sitting on the couch". This also leads to an interesting secondary task of QA temporal localization.

Our contribution is the TVQA dataset, built on 6 popular TV shows spanning 3 genres: medical dramas, sitcoms, and crime shows. On this data, we collected 152.5K human-written QA pairs (examples shown in Fig.1). There are 4 salient advantages of our dataset. First, it is large-scale and natural, containing 21,793 video clips from 925 episodes. On average, each show has 7.3 seasons, providing long range character interactions and evolving relationships. Each video clip is associated with 7 questions, with 5 answers (1 correct) for each question. Second, our video clips are relatively long (60-90 seconds), thereby containing more social interactions and activities, making video understanding more challenging. Third, we provide the dialogue (character name + subtitle) for each QA video clip. Understanding the relationship between the provided dialogue and the question-answer pairs is crucial for correctly answering many of the collected questions. Fourth, our questions are compositional, requiring algorithms to localize relevant moments (START and END points are provided for each question).

With the above rich annotation, our dataset supports three tasks: QA on the grounded clip, question-driven moment localization, and QA on the full video clip. We provide baseline experiments on both QA tasks and introduce a state-of-the-art language and vision-based model (leaving moment localization for future work).

2 Related Work

Visual Question Answering: Several image-based VQA datasets have recently been constructed, e.g., DAQUAR (Malinowski and Fritz, 2014), VQA (Antol et al., 2015), COCO-Q (Ren et al., 2015a), FM-IQA (Gao et al., 2015), Visual Madlibs (Yu et al., 2015), Visual7W (Zhu et al., 2016), CLEVR (Johnson et al., 2017), etc. Additionally, several video-based QA datasets have also been proposed, e.g. TGIF-QA (Jang et al., 2017), MovieFIB (Maharaj et al., 2017b), VideoQA (Zhu et al., 2017), LSMDC (Rohrbach et al., 2015), TRECVID (Over et al., 2014), MovieQA (Tapaswi et al., 2016), PororoQA (Kim et al., 2017) and MarioQA (Mun et al., 2017). However, none of these datasets provides a truly realistic, multimodal QA scenario where both visual and language understanding are required to answer a large portion of questions, either due to unrealistic video sources (PororoQA, MarioQA) or data collection strategy being more focused on either visual (MovieFIB, VideoQA, TGIF-QA) or language (MovieQA) sources. In comparison, our TVQA collection strategy takes a directly multimodal approach to construct a large-scale, real-video dataset by letting humans ask and answer questions while watching TV-show videos with associated dialogues.

Text Question Answering: The related task of text-based question answering has been extensively explored (Richardson et al., 2013; Weston et al., 2015; Rajpurkar et al., 2016; Hermann et al., 2015; Hill et al., 2015). Richardson et al. (2013) collected MCTest, a multiple choice QA dataset intended for open-domain reading comprehension.

With the same goal in mind, Rajpurkar et al. (2016) introduced the SQuAD dataset, but their answers are specific spans from long passages. Weston et al. (2015) designed a set of tasks with automatically generated QAs to evaluate the textual reasoning ability of artificial agents and Hermann et al. (2015); Hill et al. (2015) constructed the cloze dataset on top of an existing corpus. While questions in these text QA datasets are specifically designed for language understanding, TVQA questions require both vision understanding and language understanding. Although methods developed for text QA are not directly applicable to TVQA tasks, they can provide inspiration for designing suitable models.

Natural Language Object Retrieval: Language grounding addresses the task of object or moment localization in an image or video from a natural language description. For image-based object grounding, there has been much work on phrase grounding (Plummer et al., 2015; Wang et al., 2016b; Rohrbach et al., 2016) and referring expression comprehension (Hu et al., 2016; Yu et al., 2016; Nagaraja et al., 2016; Yu et al., 2017, 2018b). Recent work (Vasudevan et al., 2018) extends the grounding task to the video domain. Most recently, moment localization was proposed in (Hendricks et al., 2017; Gao et al., 2017), where the goal is to localize a short moment from a long video sequence given a query description. Accurate temporal grounding is a necessary step to answering our compositional questions.

3 TVQA Dataset

3.1 Dataset Collection

We collected our dataset on 6 long-running TV shows from 3 genres: 1) sitcoms: *The Big Bang Theory, How I Met Your Mother, Friends*, 2) medical dramas: *Grey's Anatomy, House*, 3) crime drama: *Castle*. There are in total 925 episodes spanning 461 hours. Each episode was then segmented into short clips. We first created clips every 60/90 seconds, then shifted temporal boudaries to avoid splitting subtitle sentences between clips. Shows that are mainly conversational based, e.g., *The Big Bang Theory*, were segmented into 60 seconds clips, while shows that are less cerebral, e.g. *Castle*, were segmented into 90 seconds clips. In the end, 21,793 clips were prepared for QA collection, accompanied with subtitles and aligned with transcripts to add character names. A sample clip is shown in Fig. 1.

Amazon Mechanical Turk was used for VQA collection on video clips, where workers were presented with both videos and aligned named subtitles, to encourage multimodal questions requiring both vision and language understanding to answer. Workers were asked to create questions using a compositional-question format: [What/How/Where/Why/...] ______ [when/before/after] ______. The second part of each question serves to localize the relevant video moment within a clip, while the first part poses a question about that moment. This compositional format also serves to encourage questions that require both visual and language understanding to answer, since people often naturally use visual signals to ground questions in time, e.g. *What was House saying before he leaned over the bed?* During data collection, we only used prompt words (when/before/after) to encourage workers to propose the desired, complex compositional questions. There were no additional template constraints. Therefore, most of the language in the questions is relatively free-form and complex.

Ultimately, workers pose 7 different questions for each video clip. For each question, we asked workers to annotate the exact video portion required to answer the question by marking the START and END timestamps as in Krishna et al. (2017). In addition, they provide 1 correct and 4 wrong answers for each question. Workers get paid $1.3 for a single video clip annotation. The whole collection process took around 3 months.

To ensure the quality of the questions and answers, we set up an online checker in our collection interface to verify the question format, allowing only questions that reflect our two-step format to be submitted. The collection was done in batches of 500 videos. For each harvested batch, we sampled 3 pairs of submitted QAs from each worker and checked the semantic correctness of the questions, answers, and timestamps.

3.2 Dataset Analysis

Multiple Choice QAs: Our QAs are multiple choice questions with 5 candidate answers for each question, for which only one is correct. Table 1 provides statistics of the QAs based on the first question word. On average, our questions contain 13.5 words, which is fairly long compared to other datasets. In general, correct answers tend

QType	#QA	Q. Len.	CA. Len.	WA. Len.
what	84768	13.3	4.9	4.3
who	17654	13.4	3.1	3.0
where	17777	12.5	5.2	4.8
why	15798	14.5	9.0	7.7
how	13644	14.4	5.7	5.1
others	2904	15.2	4.9	4.7
total	152545	13.5	5.2	4.6

Table 1: Statistics for different question types based on first question word. Q = question, CA = correct answer, WA = wrong answer. Length is defined as the number of words in the sentence.

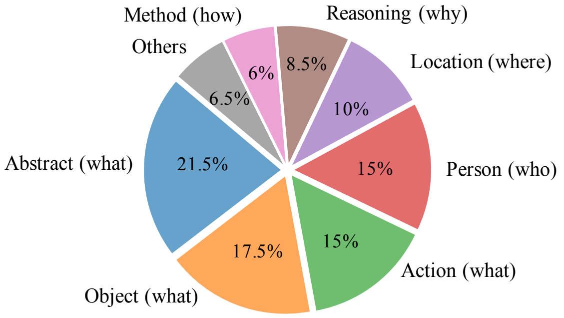

Figure 2: Distribution of question types based on answer types.

to be slightly longer than wrong answers. Fig. 2 shows the distribution of different questions types. Note "what" (Abstract, Object, Action), "who" (Person), "why" (Reasoning) and "where" (Location) questions form a large part of our data.

The negative answers in TVQA are written by human annotators. They are instructed to write false but relevant answers to make the negatives challenging. Alternative methods include sampling negative answers from other questions' correct answers, either based on semantic similarity (Das et al., 2017; Jang et al., 2017) or randomly (Antol et al., 2015; Das et al., 2017). The former is prone to introducing paraphrases of the ground-truth answer (Zhu et al., 2016). The latter avoids the problem of paraphrasing, but generally produces irrelevant negative choices. We show in Table 8 that our human written negatives are more challenging than randomly sampled negatives.

Moment Localization: The second part of our question is used to localize the most relevant video portion to answer the question. The prompt of "when", "after", "before" account for 60.03%, 30.19% and 9.78% respectively of our dataset. TVQA provides the annotated START and END timestamps for each QA. We show the annotated segment lengths in Fig. 3. We found most of the questions rely on relatively short moments (less than 15 secs) within a longer clip (60-90 secs).

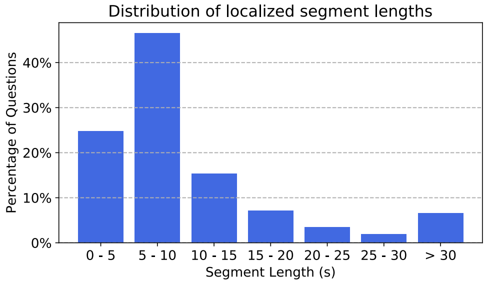

Figure 3: Distribution of localized segment lengths. The majority of our questions have timestamp localized segment with length less than 15 seconds.

Show	Genre	#Sea.	#Epi.	#Clip	#QA
BBT	sitcom	10	220	4,198	29,384
Friends	sitcom	10	226	5,337	37.357
HIMYM	sitcom	5	72	1,512	10,584
Grey	medical	3	58	1,427	9,989
House	medical	8	176	4,621	32,345
Castle	crime	8	173	4,698	32,886
Total	—	44	925	21,793	152,545

Table 2: Data Statistics for each TV show. BBT = *The Big Bang Theory*, HIMYM = *How I Met You Mother*, Grey = *Grey's Anatomy*, House = *House M.D.*, Epi = Episode, Sea. = Season

Show	Top unique nouns
BBT	game, mom, laptop, water, store, dinner, book, stair, computer, food, wine, glass, couch, date
Friends	shop, kiss, hair, sofa, jacket, counter, coffee, everyone, coat, chair, kitchen, baby, apartment
HIMYM	bar, beer, drink, job, dad, sex, restaurant, wedding, party, booth, dog, story, bottle, club, painting
Grey	nurse, side, father, hallway, scrub, chart, wife, window, life, family, chief, locker, head, surgery
House	cane, team, blood, test, brain, pill, office, pain, symptom, diagnosis, hospital, coffee, cancer, drug
Castle	gun, victim, picture, case, photo, body, murder, suspect, scene, crime, money, interrogation

Table 3: Top unique nouns in questions and correct answers.

Differences among our 6 TV Shows: The videos used in our dataset are from 6 different TV shows. Table 2 provides statistics for each show. A good way to demonstrate the difference among questions from TV shows is to show their top unique nouns. In Table 3, we present such an analysis. The top unique nouns in sitcoms (*BBT*, *Friends*, *HIMYM*) are mostly daily objects, scenes and actions, while medical dramas (*Grey, House*) questions contain more medical terms, and crime shows (*Castle*) feature detective terms. Although similar, there are also notable differences among shows in the same genre. For example, *BBT* con-

Dataset	V. Src.	QType	#Clips / #QAs	Avg. Len.(s)	Total Len.(h)	Q. Src.		Timestamp annotation
						text	video	
MovieFIB (Maharaj et al., 2017a)	Movie	OE	118.5k / 349k	4.1	135	✓	-	-
Movie-QA (Tapaswi et al., 2016)	Movie	MC	6.8k / 6.5k	202.7	381	✓	-	✓
TGIF-QA (Jang et al., 2017)	Tumblr	OE&MC	71.7k / 165.2k	3.1	61.8	✓	✓	-
Pororo-QA (Kim et al., 2017)	Cartoon	MC	16.1k / 8.9k	1.4	6.3	✓	✓	-
TVQA (our)	TV show	MC	21.8k / 152.5k	76.2	461.2	✓	✓	✓

Table 5: Comparison of TVQA to various existing video QA datasets. OE = open-ended, MC = multiple-choices. Q. Src. = Question Sources, it indicates where the questions are raised from. TVQA dataset is unique since its questions are based on both text and video, with additional timestamp annotation for each of them. It is also significantly larger than previous datasets in terms of total length of videos.

Character	Top unique nouns
Sheldon	Arthur, train, Kripke, flag, flash, Wil, logo, Barry, superhero, Spock, trek, sword
Leonard	Leslie, helium, robe, Dr, team, Kurt university, key, chess, Stephen
Howard	NASA, trick, van, language, summer, letter, Mike, station, peanut, Missy
Raj	Lucy, Claire, parent, music, nothing, Isabella, bowl, sign, back, India, number
Penny	basket, order, mail, mouth, cheesecake, factory shower, pizza, cream, Alicia, waitress, ice
Amy	Dave, meemaw, tablet, birthday, monkey, coat, brain, ticket, laboratory, theory, lip, candle
Bernadette	song, sweater, wedding, child, husband, everyone, necklace, stripper, weekend, airport

Table 4: Top unique nouns for characters in BBT.

VQA source	Human accuracy on test.
Question	31.84
Video and Question	61.73
Subtitle and Question	72.88
Video, Subtitle, and Question	89.41

Table 5: Human accuracy on test set based on different sources. As expected, humans get the best performance when given both videos and subtitles.

tains "game" and "laptop" while *HIMYM* contains "bar" and "beer", indicating the different major activities and topics in each show. Additionally, questions about different characters also mention different words, as shown in Table 4.

Comparison with Other Datasets: Table 5 presents a comparison of our dataset to some recently proposed video question answering datasets. In terms of total length of videos, TVQA is the largest, with a total of 461.2 hours of videos. MovieQA (Tapaswi et al., 2016) is most similar to our dataset, with both multiple choice questions and timestamp annotation. However, their questions and answers are constructed by people posing questions from a provided plot summary, then later aligned to the video clips, which makes most of their questions text oriented.

Human Evaluation on Usefulness of Video and Subtitle in Dataset: To gain a better understanding of the roles of videos and subtitles in the our dataset, we perform a human study, asking different groups of workers to complete the QA task in settings while observing different sources (subsets) of information:

- Question only.
- Video and Question.
- Subtitle and Question.
- Video, Subtitle, and Question.

We made sure the workers that have written the questions did not participate in this study and that workers see only one of the above settings for answering each question. Human accuracy on our test set under these 4 settings are reported in Table 5. As expected, compared to human accuracy based only on question-answer pairs (Q), adding videos (V+Q), or subtitles (S+Q) significantly improves human performance. Adding both videos and subtitles (V+S+Q) brings the accuracy to 89.41%. This indicates that in order to answer the questions correctly, both visual and textual understanding are essential. We also observe that workers obtain 31.84% accuracy given question-answer pairs only, which is higher than random guessing (20%). We ascribe this to people's prior knowledge about the shows. Note, timestamp annotations are not provided in these experiments.

4 Methods

We introduce a multi-stream end-to-end trainable neural network for Multi-Modal Video Question Answering. Fig. 4 gives an overview of our model. Formally, we define the inputs to the model as: a 60-90 second video clip V, a subtitle S, a question q, and five candidate answers $\{a_i\}_{i=0}^{4}$.

4.1 Video Features

Frames are extracted at 3 fps. We run Faster R-CNN (Ren et al., 2015b) trained on the Visual

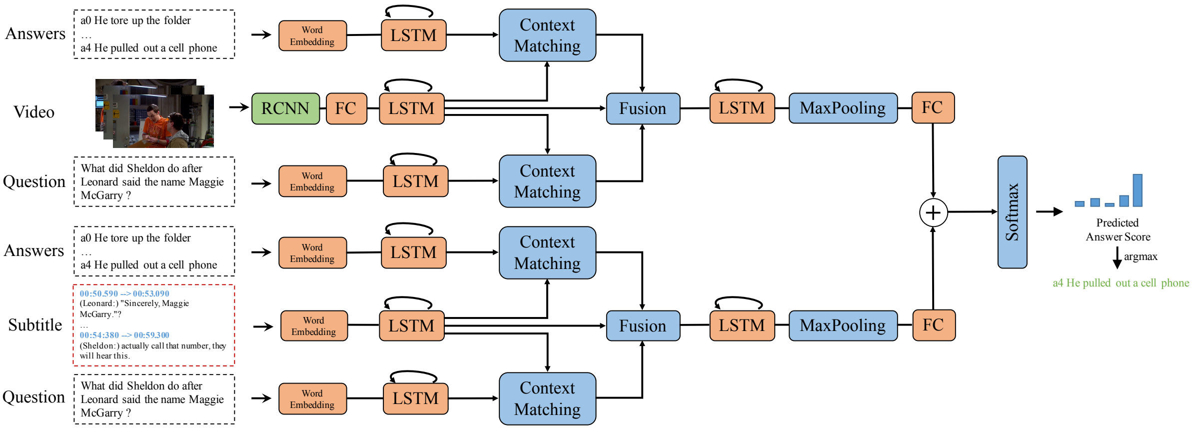

Figure 4: Illustration of our multi-stream model for Multi-Modal Video QA. Our full model takes different contextual sources (regional visual features, visual concept features, and subtitles) along with question-answer pair as inputs to each stream. For brevity, we only show regional visual features (upper) and subtitle (bottom) streams.

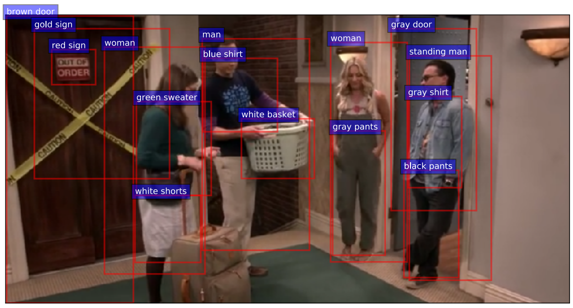

brown door, gold sign, red sign, woman, white shorts,
green sweater, man, blue shirt, white basket, woman,
gray pants, gray door, standing man, gray shirt, black pants

Figure 5: Faster R-CNN detection example. The detected object labels and attributes can be viewed as a description to the frame, which is potentially helpful to answer a visual question.

Genome (Krishna et al., 2017) to detect object and attribute regions in each frame. Both regional features and predicted detection labels can be used as model inputs. We also use ResNet101 (He et al., 2016) trained on ImageNet (Deng et al., 2009) to extract whole image features.

Regional Visual Features: On average, our videos contain 229 frames, with 16 detections per frame. It is not trivial to model such long sequences. For simplicity, we follow (Anderson et al., 2018; Karpathy and Fei-Fei, 2015) selecting the top-K regions[1] from each detected label across all frames. Their regional features are L2-normalized and stacked together to form our visual representation $V^{reg} \in \mathbb{R}^{n_{reg} \times 2048}$. Here n_{reg} is the number of selected regions.

Visual Concept Features: Recent work (Yin and Ordonez, 2017) found that using detected object labels as input to an image captioning system gave comparable performance to using CNN features directly. Inspired by this work, we also experiment with using detected labels as visual inputs. As shown in Fig. 5, we are able to detect rich visual concepts, including both objects and attributes, e.g. "white basket", which could be used to answer "What is Sheldon holding in his hand when everyone is at the door". We first gather detected concepts over all the frames to represent concept presence. After removing duplicate concepts, we use GloVe (Pennington et al., 2014) to embed the words. The resulting video representation is denoted as $V^{cpt} \in \mathbb{R}^{n_{cpt} \times 300}$, where n_{cpt} is the number of unique concepts.

ImageNet Features: We extract the pooled 2048D feature of the last block of ResNet101. Features from the same video clip are L2 normalized and stacked, denoted as $V^{img} \in \mathbb{R}^{n_{img} \times 2048}$, where n_{img} is the number of frames extracted from the video clip.

4.2 LSTM Encoders for Video and Text

We use a bi-directional LSTM (BiLSTM) to encode both textual and visual sequences. A subtitle S, which contains a set of sentences, is flattened into a long sequence of words and GloVe (Pennington et al., 2014) is used to embed the words. We stack the hidden states of the BiLSTM from both directions at each timestep to obtain the subtitle representation $H^S \in \mathbb{R}^{n_S \times 2d}$, where n_S is the number of subtitle words, d is the hidden size of the BiLSTM (set to 150 in our experiments). Similarly, we encode question $H^q \in \mathbb{R}^{n_q \times 2d}$, candidate answers $H^{a_i} \in \mathbb{R}^{n_{a_i} \times 2d}$, and visual con-

---

[1] Based on cross-validation, we find K=6 to perform best.

cepts $H^{cpt} \in \mathbb{R}^{n_{cpt} \times 2d}$. n_q and n_{a_i} are the number of words in question and answer a_i, respectively. Regional features V^{reg} and ImageNet features V^{img} are first projected into word vector space using a non-linear layer with tanh activation, then encoded using the same BiLSTM to obtain the regional representations $H^{reg} \in \mathbb{R}^{n_{reg} \times 2d}$ and $H^{img} \in \mathbb{R}^{n_{img} \times 2d}$, respectively.

4.3 Joint Modeling of Context and Query

We use a context matching module and BiLSTM to jointly model the contextual inputs (subtitle, video) and query (question-answer pair). The context matching module is adopted from the context-query attention layer from previous works (Seo et al., 2017; Yu et al., 2018a). It takes context vectors and query vectors as inputs and produces a set of context-aware query vectors based on the similarity between each context-query pair.

Taking the regional visual feature stream as an example (Fig. 4 upper stream), where H^{reg} is used as context input[2]. The question embedding, H^q, and answer embedding, H^{a_i}, are used as queries. After feeding context-query pairs into the context matching module, we obtain a video-aware-question representation, $G^{reg,q} \in \mathbb{R}^{n_{reg} \times 2d}$, and video-aware-answer representation, $G^{reg,a_i} \in \mathbb{R}^{n_{reg} \times 2d}$, which are then fused with video context:

$$\begin{aligned} M^{reg,a_i} = [&H^{reg}; G^{reg,q}; G^{reg,a_i}; \\ &H^{reg} \odot G^{reg,q}; H^{reg} \odot G^{reg,a_i}], \end{aligned}$$

where $\odot$ is element-wise product. The fused feature, $M^{reg,a_i} \in \mathbb{R}^{n_{reg} \times 10d}$, is fed into another BiLSTM. Its hidden states, $U^{reg,a_i} \in \mathbb{R}^{n_{reg} \times 10d}$, are max-pooled temporally to get the final vector, $u^{reg,a_i} \in \mathbb{R}^{10d}$, for answer a_i. We use a linear layer with softmax to convert $\{u^{reg,a_i}\}_{i=0}^{4}$ into answer probabilities. Similarly, we can compute the answer probabilities given subtitle as context (Fig. 4 bottom stream). When multiple streams are used, we simply sum up the scores from each stream as the final score (Wang et al., 2016a).

5 Experiments

For evaluation, we introduce several baselines and compare them to our proposed model.

In all experiments, setup is as follows. We split the TVQA dataset into 80% training, 10% validation, and 10% testing splits such that videos and their corresponding QA pairs appear in only one split. This results in 122,039 QA pairs for training, 15,253 QA pairs for validation, and 15,253 QA pairs for testing. We evaluate each model using multiple-choice question answering accuracy.

5.1 Baselines

Longest Answer: Table 1 indicates that the average length of the correct answers is longer than the wrong ones; thus, our first baseline simply selects the longest answer for each question.

Nearest Neighbor Search: In this baseline, we use Nearest Neighbor Search (NNS) to compute the closest answer to our question or subtitle. We embed sentences into vectors using TFIDF, SkipThought (Kiros et al., 2015), or averaged GloVe (Pennington et al., 2014) word vectors, then compute the cosine similarity for each question-answer pair or subtitle-answer pair. For TFIDF, we use bag-of-words to represent the sentences, assigning a TFIDF value for each word.

Retrieval: Due to the size of TVQA, there may exist similar questions and answers in the dataset. Thus, we also implement a baseline two-step retrieval approach: given a question and a set of candidate answers, we first retrieve the most relevant question in the training set, then pick the candidate answer that is closest to the retrieved question's correct answer. Similar approaches have also been used in dialogue systems (Jafarpour and Burges, 2010; Leuski and Traum, 2011), picking the appropriate responses to an utterance from a predefined human conversational corpus. Similar to NNS, we use TFIDF, SkipThought, and GloVe vectors with cosine similarity.

5.2 Results

Table 6 shows results from baseline methods and our proposed neural model. Our main results are obtained by using full-length video clips and subtitles, without using timestamps (*w/o ts*). We also run the same experiments using the localized video and subtitle segment specified by the ground truth timestamps (*w/ ts*). If not indicated explicitly, the numbers described below are from the experiments on full-length video clips and subtitles.

Baseline Comparison: Row 1 shows results of the longest answer baseline, achieving 30.41%

---

[2] For visual concept features and ImageNet features, we simply replace H^{reg} with H^{cpt} or H^{img} as the context.

	Method	Video Feature	Test Accuracy	
			w/o ts	w/ ts
0	Random	-	20.00	20.00
1	Longest Answer	-	30.41	30.41
2	Retrieval-Glove	-	22.48	22.48
3	Retrieval-SkipThought	-	24.24	24.24
4	Retrieval-TFIDF	-	20.88	20.88
5	NNS-Glove Q	-	22.40	22.40
6	NNS-SkipThought Q	-	23.79	23.79
7	NNS-TFIDF Q	-	20.33	20.33
8	NNS-Glove S	-	23.73	29.66
9	NNS-SkipThought S	-	26.81	37.87
10	NNS-TFIDF S	-	49.94	51.23
11	Our Q	-	43.34	43.34
12	Our V+Q	img	42.67	43.69
13	Our V+Q	reg	42.75	44.85
14	Our V+Q	cpt	43.38	45.41
15	Our S+Q	-	63.14	66.23
16	Our S+V+Q	img	63.57	66.97
17	Our S+V+Q	reg	63.19	67.82
18	Our S+V+Q	cpt	**65.46**	**68.60**

Table 6: Accuracy for different methods on TVQA test set. Q = Question, S = Subtitle, V = Video, img = ImageNet features, reg = regional visual features, cpt = visual concept features, ts = timestamp annotation. Human performance without timestamp annotation is reported in Table 5.

	Q	S+Q	V+Q			S+V+Q		
			img	reg	cpt	img	reg	cpt
what (55.62%)	44.11	62.29	44.96	45.93	47.44	63.88	65.28	66.05
who (11.55%)	36.55	68.33	35.75	34.85	34.68	67.76	67.20	67.99
where (11.67%)	42.58	56.97	47.13	48.43	48.20	61.97	63.71	61.46
how (8.98%)	41.17	71.97	41.17	42.41	40.95	71.17	70.80	71.53
why (10.38%)	45.23	78.65	46.05	45.36	45.48	78.33	77.13	78.77
other (1.80%)	36.50	74.45	37.23	36.50	33.58	73.72	72.63	74.09
all (100%)	42.77	65.15	43.78	44.40	45.03	66.44	67.17	67.70

Table 7: Accuracy of each question type using different models (w/ ts) on TVQA Validation set. Q = Question, S = Subtitle, V = Video, img = ImageNet features, reg = regional visual features, cpt = visual concept features. The percentage of each question type is shown in brackets.

(compared to random chance at 20%). As expected, the retrieval-based methods (row 2-4) and the answer-question similarity based methods (row 5-7) perform rather poorly, since no contexts (video or subtitle) are considered. When using subtitle-answer similarity to choose correct answers, Glove, SkipThought, and TFIDF based approaches (row 8-10) all achieve significant improvement over question-answer similarity. Notably, TFIDF (row 10) answers 49.94% of the questions correctly. Since our questions are raised by people watching the videos, it is natural for them to ask questions about specific and unique objects/locations/etc., mentioned in the subtitle. Thus, it is not surprising that TFIDF based similarity between answer and subtitle performs so well.

Variants of Our Model: Rows 11-18 show results of our model with different contextual inputs and features. The model that only uses question-answer pairs (row 11) achieves 43.34% accuracy. Compared to the subtitle model (row 15), adding video as additional sources (row 16-18) improves performance. Interestingly, adding video to the question only model (row 11) do not work as well (row 12-14). Our hypothesis is that the video feature streams may be struggling to learn models for answering textual questions, which degrades their ability to answer visual questions. Overall, the best performance is achieved by using all the contextual sources, including subtitles and videos (using concept features, row 18).

Comparison with Human Performance: Human performance without timestamp annotation is shown in Table 5. When using only questions (Table 6 row 11), our model outperforms humans (43.34% *vs* 31.84%) as it has access to all statistics of the questions and answers. When using videos or subtitles or both, humans perform significantly better than the models.

Models with Timestamp Annotation: Columns under *w/o ts* and *w/ ts* show a comparison between the same model using full-length videos/subtitles and using timestamp localized videos/subtitles. With timestamp annotation, the models perform consistently better than their counterpart without this information, indicating that localization is helpful for question answering.

Accuracy for Different Question Types: To gain further insight, we examined the accuracy of our models on different question types on the validation set (results in Table 7), all models using timestamp annotation. Compared to S+Q model, S+V+Q models get the most improvements on "what" and "where" questions, indicating these questions require additional visual information. On the other hand, adding video features did not improve S+Q performance on questions relying more on textual reasoning, e.g., "how" questions.

Human-Written Negatives vs. Randomly-Sampled Negatives For comparison, we create a new answer set by replacing the original human written negative answers with randomly sampled negative answers. To produce relevant negative answers, for each question, negatives are sampled (from the other QA pairs) within the same show.

00:00.688 --> 00:03.989
(Raj:)seemed like a chance to show off.
00:06.360 --> 00:08.243
There he is!
00:08.245 --> 00:10.412
(Raj:)There's my happy Hebraic homeboy.

What Raj and Howard are drinking when sat at the table ?

a0 Milk
a1 Beer √
a2 Juice
a3 Vodka
a4 Carrot juice .

(a)

01:04.288 --> 01:07.382
(Professor Jason Byford:)People often call with research questions, so I try to be helpful.
01:07.458 --> 01:09.178
(Professor Jason Byford:)She wanted to know what these symbols meant.

Why does the professor say he met with Susannah when Castle and Beckett are in his office ?
a0 She wanted to work on some research
a1 She wanted to take his class
a2 She discovered a new symbol
a3 She wanted to know the meaning of symbols √
a4 She needed him to translate some languages

(b)

00:36.402 --> 00:38.370
(Cuddy:)Problems in radiology.
00:39.372 --> 00:42.739
(Cuddy:)A radiation dosimeter badge turned positive.
00:42.842 --> 00:48.075
(Cuddy:)I could have a CT scanner …

Where is House when Cuddy comes to talk to him about the irradiated badge ?
a0 Wilson 's room
a1 The cafeteria
a2 His office
a3 The radiology department √
a4 Cuddy 's office

(c)

00:11.019 --> 00:13.510
(Rachel:)I'm not surprised. Have you seen them together?
00:13.722 --> 00:17.158
(Rachel:)- They're really cute.
00:18.293 --> 00:19.658
(Joey:) Cute ? This is Janice ! You remember Janice?

How was Phoebe 's hair done when Joey walked in ?

a0 Phoebe 's hair was in a bun
a1 Phoebe 's hair was down
a2 Phoebe ‘s hair was a half up do style √
a3 Phoebe 's hair was in a braid
a4 Phoebe 's hair was up in a hat

(d)

Figure 6: Example predictions from our best model. Top row shows correct predictions, bottom row shows failure cases. Ground truth answers are in green, and the model predictions are indicated by ✓. Best viewed in color.

Method	N.A. Src.	Video Feature	Val Accuracy	
			w/o ts	w/ ts
V+Q	Rand	cpt	84.64	85.01
S+Q	Rand	-	90.94	90.72
S+V+Q	Rand	cpt	91.55	92.00
V+Q	Human	cpt	43.03	45.03
S+Q	Human	-	62.99	65.15
S+V+Q	Human	cpt	64.70	67.70

Table 8: Accuracy on TVQA validation set with negative answers collected using different strategies. Negative Answer Source (N.A. Src.) indicates the collection method of the negative answers. Q = Question, S = Subtitle, V = Video, cpt = visual concept features, ts = timestamp annotation. All the experiments are conducted using the proposed multi-stream neural model.

Results are shown in Table 8. Performance on randomly sampled negatives is much higher than that of human written negatives, indicating that human written negatives are more challenging.

Qualitative Analysis: Fig. 6 shows example predictions from our S+V+Q model (row 18) using full-length video and subtitle. Fig. 6a and Fig. 6b demonstrate its ability to solve both grounded visual questions and textual reasoning question. Bottom row shows two incorrect predictions. We found that wrong inferences are mainly due to incorrect language inferences and the model's lack of common sense knowledge. For example, Fig. 6c, the characters are talking about radiology, the model is distracted to believe they are in the radiology department, while Fig. 6d shows a case of questions that need common sense to answer, rather than simply textual or visual cues.

6 Conclusion

We presented the TVQA dataset, a large-scale, localized, compositional video question answering dataset. We also proposed two QA tasks (with/without timestamps) and provided baseline experiments as a benchmark for future comparison. Our experiments show both visual and textual understanding are necessary for TVQA.

There is still a significant gap between the proposed baselines and human performance on the QA accuracy. We hope this novel multimodal dataset and the baselines will encourage the community to develop stronger models in future work. To narrow the gap, one possible direction is to enhance the interactions between videos and subtitles to improve multimodal reasoning ability. Another direction is to exploit human-object relations in the video and subtitle, as we observe that a large number of questions involve such relations. Additionally, temporal reasoning is crucial for answering the TVQA questions. Thus, future work also includes integrating better temporal cues.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and discussions. This research is supported by NSF Awards #1633295, 1562098, 1405822 and a Google Faculty Research Award, Bloomberg Data Science Research Grant, and ARO-YIP Award #W911NF-18-1-0336. The views contained in this article are those of the authors and not of the funding agency.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and vqa. *CVPR*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. *ICCV*, pages 2425–2433.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. *CVPR*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. *CVPR*, pages 248–255.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. 2015. Are you talking to a machine? dataset and methods for multilingual image question. In *NIPS*.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ramakant Nevatia. 2017. Tall: Temporal activity localization via language query. *ICCV*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. *ICCV*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *ICLR*.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016. Natural language object retrieval. *CVPR*.

Sina Jafarpour and Chris J.C. Burges. 2010. Filter, rank, and transfer the knowledge: Learning to chat. Technical report.

Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. 2017. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. *CVPR*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Kyung-Min Kim, Min-Oh Heo, Seong-Ho Choi, and Byoung-Tak Zhang. 2017. Deepstory: Video story qa by deep embedded memory networks. In *IJCAI*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Fei fei Li, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. *ICCV*, pages 706–715.

Anton Leuski and David R. Traum. 2011. Npceditor: Creating virtual human dialogue using information retrieval techniques. *AI Magazine*, 32:42–56.

Tegan Maharaj, Nicolas Ballas, Anna Rohrbach, Aaron Courville, and Christopher Pal. 2017a. A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering. *CVPR*.

Tegan Maharaj, Nicolas Ballas, Anna Rohrbach, Aaron Courville, and Christopher Pal. 2017b. A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering. *CVPR*.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *NIPS*.

Jonghwan Mun, Paul Hongsuck Seo, Ilchae Jung, and Bohyung Han. 2017. Marioqa: Answering questions by watching gameplay videos. In *ICCV*.

Varun K Nagaraja, Vlad I Morariu, and Larry S Davis. 2016. Modeling context between objects for referring expression understanding. In *ECCV*.

Paul Over, Jon Fiscus, Greg Sanders, David Joy, Martial Michel, George Awad, Alan Smeaton, Wessel Kraaij, and Georges Quénot. 2014. Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID*, page 52.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015a. Exploring models and data for image question answering. In *NIPS*.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015b. Faster r-cnn: Towards real-time object detection with region proposal networks. *TPAMI*, 39:1137–1149.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*.

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *ECCV*.

Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. 2015. A dataset for movie description. In *CVPR*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ICLR*.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. *CVPR*.

Arun Balajee Vasudevan, Dengxin Dai, and Luc Van Gool. 2018. Object referring in videos with language and human gaze. *CVPR*.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qi Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016a. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*.

Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016b. Learning deep structure-preserving image-text embeddings. In *CVPR*.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *ICLR*.

Xuwang Yin and Vicente Ordonez. 2017. Obj2text: Generating visually descriptive language from object layouts. *EMNLP*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018a. Qanet: Combining local convolution with global self-attention for reading comprehension. *ICLR*.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. 2018b. Mattnet: Modular attention network for referring expression comprehension. In *CVPR*.

Licheng Yu, Eunbyung Park, Alexander C. Berg, and Tamara L. Berg. 2015. Visual madlibs: Fill in the blank image generation and question answering. *ICCV*.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. 2016. Modeling context in referring expressions. In *ECCV*.

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. 2017. A joint speakerlistener-reinforcer model for referring expressions. In *CVPR*.

Linchao Zhu, Zhongwen Xu, Yi Yang, and Alexander G Hauptmann. 2017. Uncovering the temporal context for video question answering. *IJCV*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *CVPR*.

Localizing Moments in Video with Temporal Language

Lisa Anne Hendricks[1]*, **Oliver Wang**[2], **Eli Shechtman**[2],
Josef Sivic[2,3]*, **Trevor Darrell**[1], **Bryan Russell**[2]
[1] UC Berkeley, [2] Adobe Research, [3] INRIA

Abstract

Localizing moments in a longer video via natural language queries is a new, challenging task at the intersection of language and video understanding. Though moment localization with natural language is similar to other language and vision tasks like natural language object retrieval in images, moment localization offers an interesting opportunity to model temporal dependencies and reasoning in text. We propose a new model that explicitly reasons about different temporal segments in a video, and shows that temporal context is important for localizing phrases which include temporal language. To benchmark whether our model, and other recent video localization models, can effectively reason about temporal language, we collect the novel TEMPOral reasoning in video and language (TEMPO) dataset. Our dataset consists of two parts: a dataset with real videos and template sentences (TEMPO - Template Language) which allows for controlled studies on temporal language, and a human language dataset which consists of temporal sentences annotated by humans (TEMPO - Human Language).

1 Introduction

Consider the video and natural language query in Figure 1 where we seek to localize the desired moment in the video specified by the query. Queries like "the girl bends down" require understanding objects and actions, but do not require reasoning about different video moments. In contrast, queries like "the little girl talks after bending down" require reasoning about the *temporal relationship* between different actions ("talk" and "bend down"). Localizing natural language queries in video is an important challenge, recently studied in Hendricks et al. (2017) and Gao et al. (2017) with applications in areas such as video search and retrieval. We argue that to properly localize queries with temporal language, models must understand and reason about intra-video context.

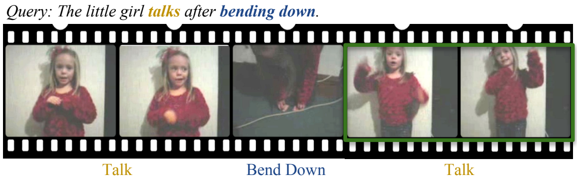

Figure 1: We consider localizing video moments which include *temporal language*. To properly localize "The little girl talks after bending down" localization models must understand how the action "talks" relates to the action "bend down."

Reasoning about intra-video context is difficult as we do not know a priori which moments should be involved in the contextual reasoning and different queries may require reasoning about different contextual moments. For example, in "the little girl talks after bending down", the relevant contextual moment "bending down" occurs just before the target moment "the little girl talks". This is in contrast to the query "the little girl talks *before* bending down" where the relevant contextual moment occurs just after. A limitation of current moment-localization models (Hendricks et al., 2017; Gao et al., 2017) is they consider query-independent video context when localizing moments. For example, when determining whether a proposed temporal region matches a natural language query, Gao et al. (2017) considers the proposed temporal region, as well as video regions just before and after the proposed region. Similarly, Hendricks et al. (2017) considers video context in the form of a global-context feature which represents the entire video. While both may implicitly include the appropriate contextual moment in their context feature, they do not explicitly determine the relevant context for the query.

To address this difficulty, we propose Moment

---

[*] Work done at Adobe during LAH's summer internship.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1380–1390
Brussels, Belgium, October 31 - November 4, 2018.

Localization with Latent Context (MLLC) which models video context as a *latent variable*. The latent variable enables the model to attend to different video contexts conditioned on the specific query/video pair, offering flexibility in the location and length of the contextual moment and overcoming the limitation of query-independent contextual reasoning. We validate the importance of latent context by showing that our model performs well both on simple queries without temporal words and more complex queries requiring temporal reasoning. Moreover, our formulation is generic and unifies approaches in Hendricks et al. (2017) and Gao et al. (2017), allowing us to ablate model component choices, as well as which kind of video context is best for localizing moments described with temporal language.

Though datasets used for moment localization in video (Hendricks et al., 2017; Regneri et al., 2013; Sigurdsson et al., 2016) include temporal language, as we will show, there is not enough temporal language to effectively train and evaluate models. We seek to extensively study this aspect, particularly with respect to temporal prepositions (Pratt-Hartmann, 2004). Thus, we collect the TEMPOral reasoning in video and language (TEMPO) dataset which builds off the recently collected DiDeMo dataset (Hendricks et al., 2017). The dataset consists of two parts: a dataset with real videos and sentences created with a template model (TEMPO - Template Language (TL)), and a dataset with real videos and newly collected user-provided temporal annotations (TEMPO - Human Language (HL)). Considering template sentences allows us to create a large dataset of sentences quickly for study of temporal language in a controlled setting. The human language data then allows us to see these trends transfer to more complex human-language queries. For data collection, we focus on the most common temporal referring words naturally occurring in language-and-video datasets.

Our contributions are twofold. (i) We are the first to study models for temporal language in video moment retrieval with natural language queries. To this end, we introduce TEMPO which includes examples of how humans use temporal language to refer to video moments. (ii) We propose MLLC for moment localization which treats video context as a latent variable and unifies prior approaches for moment localization. Our model outperforms prior work on TEMPO-TL and TEMPO-HL as well as the original DiDeMo dataset.

2 Related Work

Localizing Video Segments with Natural Language. Prior work has considered aligning natural language with video, e.g., instructional videos with transcribed text (Kiddon et al., 2015; Huang et al., 2017; Malmaud et al., 2014, 2015). Our work is most related to recent work in video moment retrieval with natural language (Gao et al., 2017; Hendricks et al., 2017). Both works take a natural language query and candidate video segment as input, and output a score for how well the natural language phrase aligns with the video segment. Gao et al. (2017) includes an additional loss to regress to start and end-points, whereas Hendricks et al. (2017) simplifies the problem by choosing from a discrete set of video segments. Importantly, to represent a proposed video segment, both models consider context features around a moment: Hendricks et al. (2017) uses global context by averaging features over an entire input video, and Gao et al. (2017) incorporates features adjacent to the proposed video segment. We argue that to do proper temporal reasoning, pre-determined, query independent context features may not cover all possible temporal relations. Thus, we propose to model the context as a *latent variable*, allowing our method to learn which context moments to consider as a function of the video and importantly, the query.

Both Gao et al. (2017) and Hendricks et al. (2017) collect data to test their models; Gao et al. (2017) considers the Charades (Sigurdsson et al., 2016) and TACoS (Regneri et al., 2013) datasets. While TACoS includes localized sentences, Charades only has sentences and activity detection localizations, so a semi-automatic method is used to align action detection annotations to visual descriptions in Charades. Hendricks et al. (2017) collected the Distinct Describable Moment (DiDeMo) dataset, which consists of Flickr (Thomee et al., 2016) videos with localized referring expressions. Both Charades and DiDeMo contain a large set of diverse videos (approximately 10,000 videos each). We chose to base TEMPO on DiDeMo because it contains more clip/sentence pairs (40,000 vs. 13,000), and is focused on general videos which we believe is an interesting and useful scenario, rather than being restricted to indoor activities.

Temporal Language. Prior work on temporal language processing has considered building explicit logical frameworks to process temporal prepositions like "during" or "until" (Pratt-Hartmann (2004), Konur (2008)). We do not derive a particular temporal logic, but rather learn to understand temporal language in a data driven fashion. Furthermore, we specifically consider how to understand temporal words commonly used when referring to video content. Other work has modeled dynamics for words which represent a change of state (e.g., "pick up") (Siskind (2001), Yu et al. (2015)) in limited environments. Though we limit the selection of temporal words in our study, the natural language in our data is open-world describing diverse events and how they relate to each other in video. Interpretation of temporal expressions in text ("The game happened on the 19^{th}") is a widely studied task (Angeli et al. (2012), Zhong et al. (2017)). Our work is distinctly different from this line of work as we specifically study temporal prepositions and how they refer to video.

Modeling Visual Relationships. A variety of papers have considered modeling spatial relationships in natural images (Dai et al., 2017; Hu et al., 2017; Peyre et al., 2017; Plummer et al., 2017). Our approach is analogous to this in the temporal domain; we hope to localize moments in videos. CLEVR, a synthetic visual question answering (VQA) dataset (Johnson et al., 2016), was created to allow researchers to systematically study the ability of models to perform complex reasoning. Our dataset is partially motivated by the success of CLEVR to enable researchers to study reasoning abilities of different models in a controlled setting. In contrast to CLEVR we consider a more diverse visual input in the form of real videos.

In the video domain, the TGIF-QA (Jang et al., 2017) and Mario-QA (Mun et al., 2016) datasets provide opportunities to study temporal reasoning for the task of VQA. The TGIF-QA dataset considers three types of temporal questions: before/after questions, repetition count, and determining a repeating action. Each question is accompanied by multiple choice answers. Videos we consider are much longer (25-30s as opposed to an average of 3.1s) which makes the use of temporal reasoning much more important. The MarioQA dataset is an additional VQA dataset designed to gauge temporal reasoning of VQA systems. Both TGIF-QA and MarioQA datasets include template-based natural language queries. In this paper, we consider synthetic queries similar to TGIF-QA and MarioQA, but also include human language queries. In addition, unlike the MarioQA dataset, that consists of synthetic data constructed from gameplay videos, our dataset consists of real visual inputs, and includes temporal grounding of natural language phrases. Finally, neither TGIF-QA nor MarioQA include temporal localization.

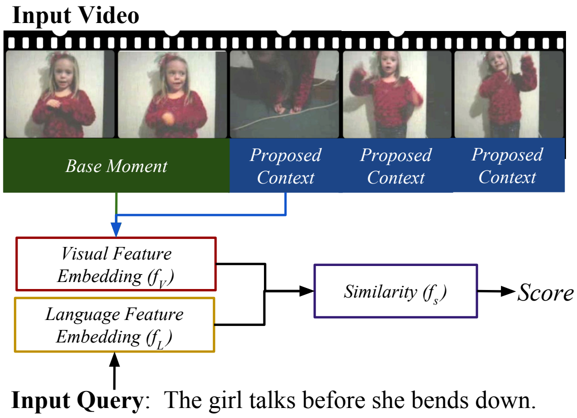

Figure 2: Our model, Moment Localization with Latent Context (MLLC), takes a video and a text query as input and outputs the moment in the video corresponding to the query. MLLC considers many different *context* moments (blue) for a specific *base* moment (green).

3 Moment Localization with Latent Context

Given a video v and natural-language query q describing a moment in the video, our goal is to output the moment $\tau = \left(\tau^{(s)}, \tau^{(e)}\right)$ where $\tau^{(s)}$ and $\tau^{(e)}$ are temporal start and end points in the video, respectively. In the following, we formulate a generic, unified model which encompasses prior approaches (Hendricks et al., 2017; Gao et al., 2017). This allows us to explore and evaluate trade offs for different model components and extensions which then leads to higher performance. Unlike prior work, we consider a latent context variable which enables our model to better reason about temporal language.

Let the moment τ corresponding to the text query be the *base* moment and the set of other video moments T_τ be possible *context* moments for τ. We define a scoring function between the video moment and natural-language query by maximizing over all possible context moments

$\tau' \in \mathrm{T}_\tau$,

$$s_\phi(v, q, \tau) = \max_{\tau' \in \mathrm{T}_\tau} f_\mathcal{S}\left(f_\mathcal{V}\left(v, \tau, \tau'\right), f_\mathcal{L}(q)\right), \tag{1}$$

where $f_\mathcal{V}$ and $f_\mathcal{L}$ are functions computing features over the video and language query, $f_\mathcal{S}$ is a similarity function, and ϕ are model parameters. This formulation is generic and trivially encompasses the MCN and TALL formulations by letting the set of possible context moments T_τ be their respective single-context moment. Figure 2 shows the generic structure of our model.

With this formulation, we seek to answer the following questions: (i) Which combination of model components performs best for the moment-retrieval task? Though our primary goal is localizing moments with temporal language, we believe a good base moment retrieval model is important for localizing moments with temporal language. (ii) How best to incorporate context for moment retrieval with temporal language? We first detail the different terms and outline different model design choices, where design choices marked with ***bold-italic font*** is ablated in Section 5. Components which are used in our final proposed Moment Localization with Latent Context (MLLC) model and prior models are summarized in Table 3.

Video feature $f_\mathcal{V}$. The video feature $f_\mathcal{V} = (g(v, \tau), g(v, \tau'), f_\mathcal{T}(\tau, \tau'))$ is a concatenation of visual features for the base $g(v, \tau)$ and context $g(v, \tau')$ moments and endpoint features $f_\mathcal{T}(\tau, \tau')$. To compute visual features g for a temporal region τ, per-frame features are averaged over the temporal region. Note that if the context moment consists of more than one contiguous temporal region, then the visual features are computed over each contiguous temporal region and then concatenated (c.f., before/after context in TALL, explained below). There are many choices for visual features. TALL (Gao et al., 2017) compares average fc_7 features (extracted from (Simonyan and Zisserman, 2014)) to features extracted with C3D (Tran et al., 2015) and LSTM features (Donahue et al., 2015). Surprisingly, C3D features only outperform average fc_7 features by a small margin. We use the visual features used in the MCN model (Hendricks et al., 2017), which are similar to the fc_7 features from (Gao et al., 2017), but included motion features as well, computed from optical flow (extracted with (Wang et al., 2016)). We then pass the extracted visual features through a MLP. Note that we learn separate embedding functions for RGB and optical flow inputs and combine scores from different input modalities using a late-fusion approach (Hendricks et al., 2017).

Endpoint feature $f_\mathcal{T}$. Modeling temporal context requires understanding how different temporal segments relate in time. Hendricks et al. (2017) suggest including temporal endpoint features (***TEF***) $f_\mathcal{T} = \left(\tau^{(s)}, \tau^{(e)}\right)$ for the base moment which encode when the moment starts and ends to better localize sentences which include words like “first” and “last”. Note that TALL (Gao et al., 2017) does not incorporate TEFs. In order to understand temporal relationships, it is important that models also include features which indicate when a context moment occurs. In addition to providing TEFs for base moments, we also experiment with concatenating TEFs for context moments (***conTEF***) $f_\mathcal{T} = \left(\tau^{(s)}, \tau^{(e)}, \tau'^{(s)}, \tau'^{(e)}\right)$.

Language feature $f_\mathcal{L}$. Text queries are transformed into a fixed-length vector with an LSTM (Hochreiter and Schmidhuber, 1997). Before inputting words into the LSTM, they are embedded in the Glove (Pennington et al., 2014) embedding space. The final layer of the LSTM is projected into the shared video-language embedding space with a fully connected layer. Gao et al. (2017) considers LSTM language features and Skip-thought encoders. Our main goal is to study how context impacts moment localization with temporal language, so we use the LSTM features used on the original DiDeMo dataset.

Similarity $f_\mathcal{S}$. Given video $f_\mathcal{V}$ and language $f_\mathcal{L}$ features, we consider three ways to encode similarity between the features. Like Hendricks et al. (2017), we consider a ***distance-based*** similarity $f_\mathcal{S} = \left(|f_\mathcal{V} - f_\mathcal{L}|^2\right)$. Second, we consider a fused-feature similarity (***mult***) where the Hadamard product $f_\mathcal{V} \odot f_\mathcal{L}$ between the two features are passed to a MLP. We also explore unit normalizing features before the Hadamard product (***normalized mult***). Finally, we consider the similarity (***TALL similarity***) which consists of the concatenation $(f_\mathcal{V}, f_\mathcal{L}, f_\mathcal{V} \odot f_\mathcal{L}, f_\mathcal{V} + f_\mathcal{L})$ and then passed to a MLP.

Context moments T_τ. We consider three sets of context moments. First, we consider the entire video as the context moment (***global***) following Hendricks et al. (2017). Second, we consider using the moments just before and after the base moment (***before/after***). Finally, we consider using the set of all possible moments (***latent*** context) which offers greatest flexibility in contextual reasoning.

Training loss. We consider two training losses. The first loss is the MCN ***ranking loss*** which encourages positive moment/query pairs to have a smaller distance in a shared embedding space than negative moment/query pairs. To sample negative moment/sentence pairs, they consider negative moments *within* a specific video (called intra-video negative moments) and negative moments in different videos (called inter-video negative moments). This sampling strategy leads to a small improvement in performance (approximately one point on all metrics) when compared to just using intra-video negative moments. We also consider the alignment loss used in TALL (***TALL loss***) which is the sum of two log-logistic functions over positive and negative training query/moment pairs (intra-video negatives are used).

Supervising context moments. For the temporal sentences in our newly collected dataset (Section 4), we have access to the ground-truth context moment during training. Thus, we can contrast a ***weakly supervised*** setting in which we optimize over the unknown latent context moments during learning and inference to a ***strongly supervised setting***.

Implementation details. Candidate base and context moments coincide to the pre-segmented five-second segments used when annotating DiDeMo. Moments may consist of any contiguous set of five-second segments. For a 30-second video partitioned into six five-second segments, there are 21 possible moments. All models were implemented in Caffe (Jia et al., 2014) and optimized with SGD. Models were trained for ~ 90 epochs with an initial learning rate of 0.05, which decreases every 30 epochs. Code is publicly released[*].

4 The TEMPO Dataset

We collect the TEMPOral reasoning in video and language (TEMPO) dataset based off the recently released DiDeMo dataset. Our dataset consists of two parts: TEMPO - Template Language (TL) and TEMPO - Human Language (HL). We create TEMPO - TL using language templates to augment the original sentences in DiDeMo with temporal words. The template allows us to generate a large number of sentences with known ground truth base and context moments. However, template language lacks the complexity of human language, so we then collect an additional fully user-constructed dataset, TEMPO - HL, consisting of sentences that contain specific temporal words.

Temporal Words in Current Datasets. We first analyze temporal words which occur in current natural language moment retrieval datasets. We consider temporal adjectives, adverbs, and prepositions found both by closely analyzing moment-localization datasets and consulting lists containing words which belong to different parts of speech. In particular, we rely on the preposition project (Litkowski and Hargraves, 2005)[†] to scrape relevant temporal words. Table 2 shows example temporal words and the number of times they occur in each dataset (TACoS (Regneri et al., 2013), Charades (Gao et al., 2017), DiDeMo (Hendricks et al., 2017)). Though all moment localization datasets use temporal words, they do not contain enough examples to reliably train and evaluate current models. Additionally, we observe that temporal words which are frequently used when describing video segments are different than those commonly used in text without video grounding. For example, in Pratt-Hartmann (2004), "during" is a common example, but we observe that "during" is infrequently used when describing video. Of temporal words, we focus on the four most common words, "before", "after", "then", and "while" when creating our dataset.

TEMPO - Template Language. To construct sentences in TEMPO-TL, we find adjacent moments in the DiDeMo dataset and fill in template sentences for "before", "after", and "then" temporal words. For "before", we use two templates: "X before Y" and "Before Y, X", where X and Y are sentences from the original DiDeMo dataset. Likewise for "after", we consider the templates "X after Y" and "After Y, X". For "then" we only consider one template, "X then Y."

TEMPO - Human Language. Though the template dataset is an interesting testbed for understanding temporal language, it is difficult to replicate the interesting complexities in human language. For example, when writing long sen-

---

[*] `https://people.eecs.berkeley.edu/~lisa_anne/tempo.html`

[†] `http://www.clres.com/prepositions.html`

	Endpoint Feature	Similarity	Context	Training Loss	Supervised Temp. Context
TALL (Gao et al., 2017)	None	TALL sim.	Before/After	TALL loss	None
MCN (Hendricks et al., 2017)	TEF	Distance-based	Global	Ranking	None
MLLC (ours)	**conTEF**	Normalized mult	**Latent**	Ranking	**Strongly sup.**

Table 1: Comparison of models. Bolded entries show our additions for localizing temporal language.

Dataset	Before	After	Then	While	Yet	During	Until
TACoS	50	62	731	82	23	0	4
Charades	281	27	1873	1165	0	3	1
DiDeMo	198	119	1021	266	16	21	22
TEMPO - TL	23,842	23842	11921	-	-	-	-
TEMPO - HL	6610	5495	5478	5425	-	-	-

Table 2: Word frequency of temporal words in natural language moment localization datasets.

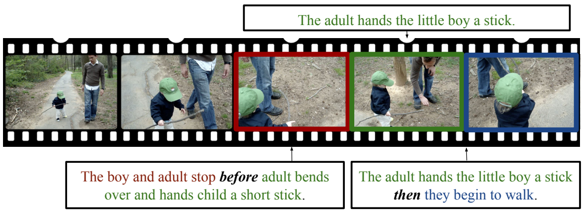

Figure 3: Example sentences in TEMPO - HL. The top sentence corresponds to the reference moment (shown in green). The bottom sentences are newly collected sentences which use temporal language.

tences with temporal prepositions, humans frequently make use of language structure such as coreference to form more cohesive statements.

To collect annotations, we follow the protocol in Hendricks et al. (2017) and segment videos into 5-second temporal segments. After collecting descriptions, we ensure descriptions are localizable by asking other workers to localize each moment. To collect data for "before", "after", and "then", we ask annotators to describe a segment in *relation* to a "reference" moment from the DiDeMo dataset. For example, if the DiDeMo dataset includes a localized phrase like "the cat jumps", annotators write a sentence which refers to the segment "the cat jumps" using a specific temporal word. We provide both the phrase ("the cat jumps") and the reference moment to annotators, and the annotators provide a sentence describing a new moment which references the reference moment.

TEMPO-HL includes unique properties which are hard to replicate with template data. Figure 3 depicts the base moment provided to workers, as well as descriptions from TEMPO-HL. In Figure 3, the description "The adult hands the little boy the stick then they walk away" includes an example of visual coreference ("they"). We note that use of pronouns is much more prevalent in TEMPO-HL, with 28.1% of sentences in TEMPO-HL including pronouns ("he", "she", "it") in contrast to 10.3% of sentences in the original DiDeMo dataset. Additionally, annotators will refer to the base moment with different language than originally used in the base moment (e.g., "the girl waves at the camera" versus the base moment "the girl looks at the camera and waves") in order to make their sentences more fluent.

5 Experiments

Evaluation Method. We follow the evaluation protocol defined for the DiDeMo dataset (Hendricks et al., 2017) over all possible combinations of the five-second video segments. We report rank at one (R@1), rank at five (R@5), and mean intersection over union (mIOU) using their aggregator over three out of the four human annotators. We compare our models on TEMPO-TL, TEMPO-HL, and the DiDeMo dataset. When training our models, we combine the DiDeMo dataset with TEMPO-TL or TEMPO-HL. This enables our model to concurrently learn to localize the simpler DiDeMo sentences with more complex TEMPO sentences.

Baselines. We compare to the two recently proposed approaches for video moment localization: MCN (Hendricks et al., 2017) and TALL (Gao et al., 2017). We adapt the implementation of TALL (Gao et al., 2017) to the DiDeMo dataset in three ways. First, we do not include the temporal localization loss required to regress to specific start and end points as DiDeMo, and thus also TEMPO, is pre-segmented, so the model does not need to compute exact start and end points. Second, the original TALL model uses C3D features.

For a fair comparison we train both models with the same RGB and flow features extracted as was done for the original MCN model. Finally, the MCN model proposes temporal endpoint features (TEF) to indicate when a proposed moment occurs within a video. We train TALL with and without the TEF and show that TEF improves performance on the original DiDeMo dataset.

Ablations. To ablate our proposed latent context, we compare to other models which share the same MLLC base network. We consider the MLLC model with global context and before/after context. We also train a model with weakly supervised (WS) latent context and strongly supervised (SS) latent context. We also train models both with and without context TEF (conTEF).

The MLLC Base Model. We first ablate our MLLC base model (Table 3). We train our models on TEMPO-TL and DiDeMo and evaluate on the original DiDeMo dataset. All models are trained with global context. We find that the ranking loss is preferable on the DiDeMo dataset (compare lines 1 and 2) and that TALL-similarity performs better than the distance based similarity of the MCN model (compare lines 1 and 5). A simpler version of the TALL-similarity, in which the concatenated element wise multiplication, element wise sum, and concatenation is replaced by a single normalized elementwise multiplication, increases R@1 by almost one point and increases mIoU by over two points (compare lines 5-7). We call our best model the MLLC-Base model (line 7). Our MLLC-Base model performs better than previous models (MCN line 1 and TALL line 3).

	Model	Similarity	Training Loss	R@1	R@5	mIoU
1	MCN	Dist.-based	Ranking	26.63	73.38	41.14
2	MCN	Dist.-based	TALL	23.89	76.54	35.69
3	TALL	TALL-sim.	TALL	8.04	36.32	22.68
4	TALL w/TEF	TALL-sim.	TALL	23.56	72.74	35.58
5	MCN	TALL-sim	Ranking	27.52	**79.07**	41.48
6	MCN	Mult	Ranking	28.19	78.97	43.21
7	MLLC-Base	Norm. Mult	Ranking	**28.37**	78.64	**43.65**

Table 3: To select our base network, we consider different variants on the two previously proposed moment retrieval methods, TALL (Gao et al., 2017) and MCN (Hendricks et al., 2017). Results reported on val.

Results: TEMPO - TL. We first compare different moment localization models on TEMPO - TL (Table 4). In particular, our model performs well on "before" and "after" words. Additionally, our MLLC model with global context outperforms both the MCN model (Hendricks et al., 2017) and the TALL (Gao et al., 2017) model when considering all sentence types, verifying the strength of our base MLLC model.

Comparing MLLC with global context and MLLC with before/after context (compare row 4 and 5), we note that before/after context is important for localizing "before" and "after" moments. However, our model with strong supervision (row 9) outperforms the model trained with before and after context, suggesting that learning to reason about which context moment is correct (as opposed to being explicitly provided with the context before and after the moment) is beneficial. We note that strong supervision (SS) outperforms weak supervision (WS) (compare rows 7 and 9) and that the context TEF is important for best performance (compare rows 8 and 9).

We note that though the MLLC-global model outperforms our full model for "then" on TEMPO-TL, our full model performs better on then for the TEMPO-HL (Table 6). One possibility is that the "then" moments in TEMPO-TL do not require context to properly localize the moment. Because TEMPO-TL is constructed from DiDeMo sentences, constituent sentence parts are *referring*. For example, given an example sentence from TEMPO-TL (e.g., "The cross is seen for the first time *then* window is first seen in room"), the model does not need to reason about the ordering of "cross seen for the first time" and "window is seen for the first time" because both moments only happen once in the video. In contrast, when considering the sentence "The adult hands the little boy a stick *then* they begin to walk" (from Figure 3), "begin to walk" could refer to multiple video moments. Consequently, our model must reason about the temporal ordering of reference moments to properly localize the video moment.

On TEMPO - TL, sentences differ from original DiDeMo sentences solely because of the use of temporal words. Thus, we can do a controlled study of how well models understand temporal words. If a model has good temporal reasoning, then if it can localize a reference moment "the dog jumps" it should be easier for the model to localize the moment "the dog sits after the dog jumps". To test whether models are capable of this, we look at only sentences in TEMPO - TL where the model has correctly localized the cor-

		TEMPO - Template Language (TL)										
		DiDeMo		Before		After		Then		**Average**		
		R@1	mIoU	R@1	mIoU	R@1	mIoU	R@1	mIoU	R@1	R@5	mIoU
1	Frequency Prior	10.71	20.67	17.85	24.22	22.42	25.76	0.00	24.73	12.74	52.58	23.84
2	MCN	24.85	37.92	32.28	38.67	26.08	35.44	25.07	53.94	27.07	73.36	41.49
3	TALL	20.95	32.09	27.13	32.41	26.30	34.27	4.84	36.75	19.80	64.66	33.88
4	MLLC- Global	26.32	40.37	31.92	38.26	25.37	35.59	**27.53**	**57.08**	27.78	74.14	42.82
5	MLLC B/A	26.04	39.60	34.04	40.46	28.50	38.18	25.60	54.37	28.54	74.92	43.15
6	MLLC (WS)	26.57	40.99	30.56	37.64	24.76	35.10	26.95	56.49	26.95	74.18	42.55
7	MLLC (WS + conTEF)	25.87	40.37	32.01	39.51	24.31	33.94	24.98	55.22	26.79	74.04	42.27
8	MLLC (SS)	26.09	40.12	28.45	34.38	23.79	33.92	24.27	55.00	25.65	73.60	40.86
9	MLLC (SS + conTEF)	**27.46**	**41.20**	**35.31**	**41.81**	**29.38**	**38.90**	26.83	54.97	**29.74**	**76.76**	**44.22**

Table 4: Comparison of different model performance for different temporal words on TEMPO - TL on our test set. We report scores for the three temporal words in TEMPO - TL as well as on the original DiDeMo dataset. We find that our model performs best when considering all sentence types. B/A indicated before/after context, WS indicates weak context supervision, and SS indicates strong context supervision.

	Before		After		Then	
Context	R@1	mIoU	R@1	mIoU	R@1	mIoU
Global	-1.07	-2.72	-7.59	-6.75	43.30	31.57
Before/After	2.77	2.03	11.47	12.08	42.92	29.09
Latent	7.78	37.55	8.58	10.39	50.09	33.64

Table 5: Difference between performance on full dataset and set on which reference moments are localized properly for different methods on TEMPO-TL.

responding context moment in DiDeMo (Table 5). We report the *difference* in performance when considering only sentences in which temporal context was properly localized and all sentences. On our model, performance on all three temporal word types increases when the context moment can be properly localized. When considering global context, performance on "before" and "after" actually decreases, suggesting global context does not understand temporal reasoning well. Finally, even when the context is correctly localized, there is still ample room for improvement on all three sentence types motivating future work on temporal reasoning for moment retrieval.

Results: TEMPO - HL. Table 6 compares performance on TEMPO - HL. We compare our best-performing model from training on the TEMPO-TL (strongly supervised MLLC and conTEF) to prior work (MCN and TALL) and to MLLC with global and before/after context. Performance on TEMPO-HL is considerably lower than TEMPO-TL suggesting that TEMPO-HL is harder than TEMPO-TL.

On TEMPO - HL, we observe similar trends as on TEMPO-TL. When considering all sentence types, MLLC has the best performance across all metrics. In particular, our model has the strongest performance for all sentence types considering the mIoU metric. In addition to performing better on temporal words, our model also performs better on the original DiDeMo dataset. As was seen in TEMPO-TL, including before/after context performs better than our model trained with global context for both "before" and "after" words.

The final row of Table 6 shows an upper bound in which the ground truth context is used at test time instead of the latent context. We note that results improve for "before", "after", and "then", suggesting that learning to better localize context will improve results for these sentence types.

Localizing Context Fragments. TEMPO-HL sentences can be broken into two parts: a base-sentence fragment (which refers to the base moment), and a context-sentence fragment (which refers to the context moment). For example, for the sentence "The girl holds the ball before throwing it,", "the girl holds the ball" is the base fragment and "throwing it" is the context fragment. A majority of the "before" and "after" sentences in TEMPO-HL are of the form "X before (or after) Y", so we can determine a list of sentence fragments by splitting sentences based on the temporal word. Given "before" and "after" sentences, we determine the ground truth context fragment by considering which reference moment was given to annotators. We can then measure how well models localize context fragments. Table 7 compares two approaches to localizing context fragments: inputting just the context fragment into MLLC

TEMPO - Human Language (HL)												
	DiDeMo		Before		After		Then		While		**Average**		
	R@1	mIoU	R@1	mIoU	R@1	mIoU	R@1	mIoU	R@1	mIoU	R@1	R@5	mIoU
Frequeny Prior	19.43	25.44	29.31	51.92	0.00	0.00	0.00	7.84	4.74	12.27	10.69	37.56	19.50
MCN	26.07	39.92	26.79	51.40	**14.93**	34.28	18.55	47.92	10.70	35.47	19.4	70.88	41.80
TALL + TEF	21.79	33.55	25.91	49.26	14.43	32.62	2.52	31.13	8.1	28.14	14.55	60.69	34.94
MLLC - Global	27.01	41.72	27.42	52.22	14.10	34.33	18.40	49.17	10.86	35.36	19.56	71.23	42.56
MLLC - B/A	26.47	40.39	31.95	55.89	**14.93**	34.78	17.36	47.52	**11.32**	35.52	20.40	70.97	42.82
MLLC (Ours)	**27.38**	**42.45**	**32.33**	**56.91**	14.43	**37.33**	**19.58**	**50.39**	10.39	**35.95**	**20.82**	**71.68**	**44.57**
MLLC (Ours) Context Sup. Test	27.39	42.25	52.58	80.37	36.48	75.79	36.05	70.51	10.39	35.87	32.58	79.86	60.96

Table 6: Comparison of different model performance on TEMPO - HL on the test set. “MLLC - Global” indicates our model with global context and “MLLC - B/A” indicated MLLC with before/after context.

	Before		After	
	R@1	mIoU	R@1	mIoU
Context Fragment	25.16	32.94	23.05	27.64
Full Sentence	**27.55**	**35.70**	**32.67**	**40.39**

Table 7: Comparison of different methods to localize context fragments (e.g., the text “she bends down” in the sentence “the girl talks after she bends down”). We compare localizing fragments with the MLLC model to localizing fragments with the latent context considered when localizing the whole query.

Figure 4: Moment localization predictions on TEMPO - HL using our model. In addition to the localized query, we show the selected context segment (blue line) that our model considers when localizing the query.

and reporting the context used by MLLC when inputting the entire query into our model. We find that our model reliably selects the correct context fragments, most likely because it can properly exploit temporal understanding of how the context fragment relates to the base fragment.

Visualizing Context. In addition to a localized query, we can also visualize which context moment the temporal query refers to. Figure 4 shows predicted moments and their corresponding context moments. For the query “The girl with a hat takes a drink before the girl without a hat waves”, the little girl in the hat drinks twice, but our model correctly localizes the time she drinks *before* the other girl waves. Likewise, for the moment “After zooming in to the dog, the dog darts across the grass and into the woods”, the dog darts towards the woods twice (at the beginning of the video and at the end). Our model properly localizes the moment when the dog runs towards the forest the second time as well as the context fragment “zooming in on dog” when localizing the moment.

Discussion. We show promising results on both TEMPO-TL and TEMPO-HL, but there is potential improvement for building better frameworks for understanding temporal language. In Table 6, strongly supervising context at test time improves overall results, suggesting that models which can better localize context text will outperform our current model. Though TEMPO and DiDeMo have over 60,000 sentences combined, visual content is quite diverse. Integrating outside data sources (e.g., image retrieval and captioning) could possibly improve results on moment localization, both with and without temporal language queries. Additionally, in Table 5, even when the MLLC model can properly localize context, it does not always properly localize temporal sentences indicating that improved temporal reasoning can also improve our results. We believe our dataset, analysis, and method are an important step towards better moment retrieval models that effectively reason about temporal language.

Acknowledgements

We thank Anna Rohrbach for helpful feedback.

References

Gabor Angeli, Christopher D Manning, and Daniel Jurafsky. 2012. Parsing time: Learning to interpret time expressions. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 446–455. Association for Computational Linguistics.

Bo Dai, Yuqi Zhang, and Dahua Lin. 2017. Detecting visual relationships with deep relational networks. *arXiv preprint arXiv:1704.03114*.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017. Tall: Temporal activity localization via language query. *ICCV 2017*.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. *ICCV 2017*.

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

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2017. Modeling relationships in referential expressions with compositional modular networks. In *Computer Vision and Pattern Recognition*.

De-An Huang, Joseph J. Lim, Li Fei-Fei, and Juan Carlos Niebles. 2017. Unsupervised visual-linguistic reference resolution in instructional videos. In *Computer Vision and Pattern Recognition*.

Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. 2017. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. *arXiv preprint arXiv:1704.04497*.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2016. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *arXiv preprint arXiv:1612.06890*.

C. Kiddon, G. T. Ponnuraj, L. Zettlemoyer, and Y. Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 conference on empirical methods in natural language processing (EMNLP)*.

Savas Konur. 2008. An interval logic for natural language semantics. *Advances in Modal Logic*, 7:177–191.

Kenneth C Litkowski and Orin Hargraves. 2005. The preposition project. In *Proceedings of the Second ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, pages 171–179.

Jon Malmaud, Earl J. Wagner, Nancy Chang, and Kevin Murphy. 2014. Cooking with semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 33–38.

Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. 2015. What's cookin'? Interpreting cooking videos using text, speech and vision. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 143–152.

Jonghwan Mun, Paul Hongsuck Seo, Ilchae Jung, and Bohyung Han. 2016. Marioqa: Answering questions by watching gameplay videos. *arXiv preprint arXiv:1612.01669*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Julia Peyre, Ivan Laptev, Cordelia Schmid, and Josef Sivic. 2017. Weakly-supervised learning of visual relations. *arXiv preprint arXiv:1707.09472*.

Bryan A Plummer, Arun Mallya, Christopher M Cervantes, Julia Hockenmaier, and Svetlana Lazebnik. 2017. Phrase localization and visual relationship detection with comprehensive image-language cues. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1928–1937.

Ian Pratt-Hartmann. 2004. Temporal prepositions and their logic. In *Temporal Representation and Reasoning, 2004. TIME 2004. Proceedings. 11th International Symposium on*, pages 7–8. IEEE.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics (TACL)*, 1:25–36.

Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. 2016. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Jeffrey Mark Siskind. 2001. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of artificial intelligence research*, 15:31–90.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2016. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer.

Haonan Yu, N Siddharth, Andrei Barbu, and Jeffrey Mark Siskind. 2015. A compositional framework for grounding language inference, generation, and acquisition in video. *Journal of Artificial Intelligence Research*, 52:601–713.

Xiaoshi Zhong, Aixin Sun, and Erik Cambria. 2017. Time expression analysis and recognition using syntactic token types and general heuristic rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 420–429.

CARD-660: Cambridge Rare Word Dataset – a Reliable Benchmark for Infrequent Word Representation Models

Mohammad Taher Pilehvar **Dimitri Kartsaklis** **Victor Prokhorov** **Nigel Collier**
Language Technology Lab, Department of Theoretical and Applied Linguistics
University of Cambridge, United Kingdom
`{mp792,dk426,vp361,nhc30}@cam.ac.uk`

Abstract

Rare word representation has recently enjoyed a surge of interest, owing to the crucial role that effective handling of infrequent words can play in accurate semantic understanding. However, there is a paucity of reliable benchmarks for evaluation and comparison of these techniques. We show in this paper that the only existing benchmark (the Stanford Rare Word dataset) suffers from low-confidence annotations and limited vocabulary; hence, it does not constitute a solid comparison framework. In order to fill this evaluation gap, we propose CAmbridge Rare word Dataset (CARD-660), an expert-annotated word similarity dataset which provides a highly reliable, yet challenging, benchmark for rare word representation techniques. Through a set of experiments we show that even the best mainstream word embeddings, with millions of words in their vocabularies, are unable to achieve performances higher than 0.43 (Pearson correlation) on the dataset, compared to a human-level upperbound of 0.90. We release the dataset and the annotation materials at `https://pilehvar.github.io/card-660/`.

1 Introduction

Words in a corpus of natural language utterances approximately follow a Zipfian distribution with their majority, in the "long tail" of frequency distribution, occurring rarely. The prominent distributional approach to semantic representation relies on enormous occurrences for each individual word; therefore, it falls short of learning accurate representations for rare words in the long tail. Moreover, it is unreasonable to expect that all words in the vocabulary of a language are observed in a text corpus, even if it is massive in size. Out-of-vocabulary (OOV) words pose one of the major ongoing challenges for word embedding techniques. Given that effective handling of rare and OOV words is crucial to accurate natural language understanding, several studies have focused on the topic during the past few years, resulting in a wide range of techniques.

However, despite the popularity of rare and subword semantic representation, the field of research has suffered from the lack of high quality generic evaluation benchmarks. A task-based evaluation, i.e., one which directly verifies the impact of representation models in a downstream NLP system, despite being very important, does not provide a solid base for comparing different models, given that small variations in the architecture, parameter setting, or initialisation can lead to performance differences. Moreover, such an evaluation would reflect the "suitability" of representations for that specific configuration and for that particular task, and might not be conclusive for other settings.

As far as generic evaluation is concerned, existing benchmarks generally target frequent words. An exception is the Stanford Rare Word (RW) Similarity dataset (Luong et al., 2013) which has been the standard evaluation benchmark for rare word representation techniques for the past few years. In Section 2.1, we will provide an in-depth analysis of RW and highlight that crowdsourcing the annotations, with no rigorous checkpoints, has compromised the reliability of the dataset. This is mainly reflected by the low inter-annotator agreement (IAA), a performance ceiling which is easily surpassed by many existing models.

To overcome this barrier and to fill the gap for a reliable benchmark for the evaluation of subword and rare word representation techniques, we introduce a new dataset, called CARD-660: Cambridge Rare Word Dataset. Compared to existing benchmarks, CARD-660 provides multiple advantages: (1) thanks to a manual curation by experts, we report IAA of around 0.90 (see Table 3) which is substantially higher than those for exist-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1391–1401
Brussels, Belgium, October 31 - November 4, 2018.

ing datasets; (2) word pairs are selected manually from a wide range of domains and, unlike existing datasets, are not bound to a specific resource; (3) word pairs in the dataset are balanced across the similarity scale; and (4) the huge gap between state of the art and IAA (more than 0.50 in terms of Spearman correlation) promises a challenging dataset with lots of potential for future research.

The paper is structured as follows. The following Section covers the related work, highlighting some of the issues with the RW dataset. Section 3 details the construction procedure for CARD-660. In Section 4, we analyse the dataset from different aspects, showing how it improves existing benchmarks. Section 5 reports our evaluation of mainstream word embeddings and recent word representation techniques on the dataset. Finally, concluding remarks are mentioned in Section 6.

2 Related Work

Word similarity datasets have been one of the oldest, still most prominent, benchmarks for the evaluation and comparison of semantic representation techniques. As a result, several word similarity datasets have been constructed during the past few decades; to name a few: RG-65 (Rubenstein and Goodenough, 1965), WordSim-353 (Finkelstein et al., 2002), YP-130 (Yang and Powers, 2005), MEN-3K (Bruni et al., 2014), SimLex-999 (Hill et al., 2015), and SimVerb-3500 (Gerz et al., 2016). Many of these English word similarity datasets have been translated to other languages to create frameworks for multilingual (Leviant and Reichart, 2015) or crosslingual (Camacho-Collados et al., 2017) semantic representation techniques. However, these datasets mostly target words that occur frequently in generic texts and, as a result, are not suitable for the evaluation of subword or rare word representation models.

One may opt for transforming a frequent-word benchmarks into an artificial rare word dataset by downsampling the dataset's words in the underlying training corpus (Sergienya and Schütze, 2015). However, this benchmark might not properly simulate a real-world rare word representation scenario (cf. Section 3.1).

2.1 Stanford RW Dataset

The Stanford Rare Word Similarity (RW) dataset is an exception as it is dedicated to evaluating infrequent word representations. The dataset has been regarded as the de facto standard evaluation benchmark for subword and rare word representation techniques. However, our analysis shows that RW suffers from multiple issues: (1) skewed distribution of the scores, (2) low-quality and inconsistent scores, and as a consequence, (3) low inter-annotator agreement.

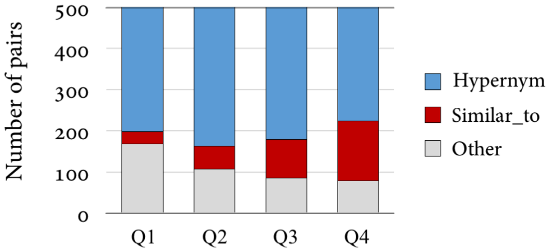

Figure 1: Distribution of relation types ("hypernymy", "similar_to", and others) across four quartiles (sorted by gold similarity scores) of the Stanford Rare Word Similarity dataset. The distribution of pairs with hypernymy relation is almost uniform across the quartiles, whereas one would expect many more pairs in the top quartiles (Q4 and Q3), given the high semantic similarity of hypernym-hyponyms.

The RW dataset comprises 2034 word pairs (i.e., $word_1$ – $word_2$). Candidates for $word_1$ were randomly sampled from Wikipedia documents, distributed across a wide range of frequencies (from 5 to 10,000) to ensure the inclusion of infrequent words. Given this automatic sampling, a measure was required to avoid noisy or junk words. To this end, a sampled word was checked in WordNet (Fellbaum, 1998) and was included only if it appeared in at least one synset. Hence, the vocabulary of the dataset is bound to that of WordNet. Words for $word_2$ were randomly picked from synsets that were directly connected to a synset of $word_1$, through various relations, such as hypernymy, holonymy, and attributes.

2.1.1 Distribution of scores

These word pairs were assigned similarity scores in $[0, 10]$. Given that all word pairs in the dataset are semantically-related according to WordNet, the scores form a skewed distribution biased towards the upper bound (see Figure 2 and Section 4.1 for more details).

2.1.2 Consistency of annotations

The scoring of the pairs has been carried out through crowdsourcing: (Amazon Mechanical) Turkers have provided ten scores for each word pair. The raters were restricted to only US-based workers and they were asked to self-certify themselves by indicating if they "knew" the word; this was used to "discard unreliable pairs." However, our analysis of the dataset clearly indicates that the above measures have not been adequate for guaranteeing quality annotations. For instance, the word *bluejacket* is paired with *submariner* in the dataset. According to WordNet (v3.0), a submariner ("a member of the crew of a submarine") is a bluejacket ("a serviceman in the navy"; a *navy_man*, *sailor*), hence a hypernymy relationship. One would expect a word to have high semantic similarity with its hypernym. However, the gold score for this pair is just 0.43 in the scale $[0, 10]$. Other examples include "untruth" (a false statement) vs. "statement" (again, with a hypernymy relationship) with a low similarity of 1.22. Apart from not being a rigorous evaluation, the self-certification does not verify if the annotator had knowledge of *various* possible meanings of a word. For instance, *decomposition* could refer to the analysis of vectors in algebra; but, when paired with *algebra*, the assigned score is only 0.75. Such examples clearly indicate that the annotators were not aware of specialised senses of some words (e.g., the algebraic meaning of decomposition), despite "knowing" the word.

In fact, there are numerous such pairs in the dataset. According to our estimate, 78% of the 2034 word pairs in the dataset are in a *hypernymy* or *similar_to* relationship. One would expect most of these (semantically similar) pairs to have been assigned high similarity scores which are closer to the upper bound of the similarity scale $[0, 10]$. However, as shown in Figure 1, these pairs are spread across the similarity scale, spanning from complete unrelatedness (lower bound) to identical semantics (synonymy). Having the words in the dataset sorted by their assigned gold scores, respectively, 66%, 79%, 83%, and 85% of the pairs in the first to fourth quartiles contain either "hypernymy" or "similar_to" relations (whereas one would expect most of these semantically-similar pairs to appear in the top quartiles).

Additionally, the dataset suffers from inconsistent annotations. For instance, the two almost identical pairs *tricolour-flag* and *tricolor-flag* were assigned substantially different scores, i.e., 5.80 and 0.71, respectively. This inconsistency is also reflected by high variances across annotators scores (cf. Section 4.3).

2.1.3 Inter-Annotator Agreement (IAA)

This validity metric reflects the homogeneity of the annotators' ratings and it is generally accepted as the upper bound for machine performance. IAA is widely used as a standard evaluation metric for the quality of word similarity datasets. A low IAA indicates a defective similarity scale or unreliable annotations.

In the RW dataset, "up to 10" annotations have been provided for the 2034 word pairs, each with a similarity score in $[0, 10]$ range. More precisely, 214 of the pairs are not provided with 10 scores, with the minimum number of scores for a pair being 7. The authors did not report IAA statistics for this dataset. Given that the annotators are not known for each pair in the released dataset, it is not straightforward to compute IAA.[1] According to a rough calculation, the average pairwise Spearman correlation between annotators' scores is 0.40, which is a significantly low figure compared to other existing word similarity datasets. We report an impressive IAA of 0.89 for our dataset (cf. Section 4.2).

3 The CARD-660 Dataset

3.1 Motivation

Due to a lack of reliable evaluation benchmarks, research in rare word representation has often resorted to artificial experimental setups such as corpus downsampling (Sergienya and Schütze, 2015; Herbelot and Baroni, 2017; Lazaridou et al., 2017). To this end, in order to simulate a rare word scenario, the rare word representation model is provided with only a limited number of occurrences for the target set of words, for instance by means of replacing the dataset's words with some other sequences of characters (e.g., by augmenting "UNK", such as "skyglowUNK" for "skyglow") in the training corpus. The computed representations on the "downsampled" training data are then either evaluated on a standard word similarity dataset (Sergienya and Schütze, 2015), such as RG-65, or compared against reference embeddings computed on a large training corpus (Herbelot and Baroni, 2017; Lazaridou et al., 2017).

However, due to the following three reasons, downsampling does not constitute a reliable

---

[1] The scores are further pruned down to only those that were within one standard deviation of the mean. This results in a further imbalanced set of scores, making the computation of IAA more challenging.

benchmark that can represent the challenging nature of the task: (1) it is unable to control the impact of second-order associations (words that frequently co-occur with the downsampled word) and cannot represent a real-world setting with novel rare usages; (2) given that morphological variations of a word (such as plural forms) are kept intact in this procedure, a subword technique can easily resort to these forms to compute the embedding for downsampled words; and (3) a constrained evaluation configuration in which the task is to estimate the embedding for a (rare) word using one or few occurrences (contexts) of it, limits the benchmark to a subset of corpus-based rare word representation techniques only. Moreover, the evaluation would require the comparison of the computed embeddings for rare words with a set of reference embeddings (computed on the full data). This dependency limits the ability of the benchmark in providing a direct evaluation of the rare word representation technique, independently from the impact of the model used to compute the reference embeddings.

The CARD-660 dataset aims at filling the gap for rigorous generic evaluation of rare word and subword representation models. In what follows in this section, we will detail the construction procedure of the dataset which was carefully planned to guarantee a challenging and reliable dataset.

3.2 Construction Procedure

The following four-phase procedure was used to construct the dataset:

(1) A set of 660 rare words were carefully selected from a wide range of domains;
(2) For each of these initial words, a pairing word was manually selected according to a randomly sampled score from the similarity scale (Section 3.2.2);
(3) All pairs were scored by 8 annotators;
(4) A final adjudication was performed to address disagreements (Section 3.2.3).

3.2.1 Similarity scale

We adopted the five-point Likert scale used for the annotation of the datasets in SemEval-2017 Task 2 (Camacho-Collados et al., 2017). The task reported high IAA scores which reflects the well-definedness and clarity of the scale. We provided annotators with the concise guideline shown in Table 1, along with several examples. Given the continuity of the scale, the annotators were given flexibility to select values in between the five points, whenever appropriate, with a step size of 0.5.

The annotators were asked in the guidelines to make sure they were familiar with *all* common meanings of the word (as defined by WordNet or other online dictionaries). To facilitate the annotation, the annotators were provided with the definitions of some of the words that were defined in WordNet or named entities that had Wikipedia pages. For others, we asked the annotators to check the word in online dictionaries, such as WordNet browser[3] and Wiktionary[4], or encyclopediae, such as Wikipedia.

3.2.2 Word pair selection

Unlike previous work (Luong et al., 2013), we did not rely on random sampling (pruned by frequency) of initial words from a specific dictionary, to prevent the dataset from being restricted to a specific resource or vocabulary. Instead, we carefully hand-picked word pairs from a wide range of domains. To construct the 660 pairs of the dataset (each pair is denoted as $w_1 - w_2$), we first picked 660 w_1 words. Our aim was to have a dataset that can ideally reflect the performance of rare word representation techniques in downstream NLP tasks. To this end, we picked initial words (w_1s) from different common NLP datasets and resources, listed in Table 2. For each text-based resource, a frequency list was obtained and rare words were carefully picked from the long tail of the list, cross-checking the frequency of words in the Google News dataset. For the other resources (such as Wiktionary), we checked a word against a large frequency list to ensure they are not frequent words. The list was computed on the 2.8B token ukWaC+WaCkypedia corpus (Baroni et al., 2009) and comprised 16.5M unique words.

In order to have a balanced distribution of scores in the dataset, we first assigned random integer scores in $[0-4]$ to the 660 initial w_1s. Then, with the corresponding score in mind, a pairing word (w_2) was selected for each w_1. We show in Section 4.1 that this strategy resulted in a uniformly distributed set of scores in the dataset.

The dataset comprises words from a wide range of genres and domains, including slang in so-

---

[2] http://www.fakenewschallenge.org

[3] http://wordnetweb.princeton.edu/perl/webwn

[4] www.wiktionary.org

Score	Interpretation	Example pair
4	**Synonyms.** The two words are different ways of referring to the same concept	*car automobile*
3	**Similar.** The two words are of the same nature, but slightly different in details	*car truck*
2	**Related.** The two words are closely related but they are not similar in their nature	*car driver*
1	**Same domain or slight relation.** The two words have distant relationship	*car tarmac*
0	**Completely unrelated.** The two words have nothing in common.	*car sky*

Table 1: The five-point Likert similarity scale used for the annotation of the dataset.

Task	Resource
Text classification.	BBC (Greene and Cunningham, 2006)
Sentiment analysis.	IMDB (Maas et al., 2011), Multi-Domain Sentiment Dataset (Blitzer et al., 2007)
Machine Translation.	Europarl (Koehn, 2005)
Question Answering.	AQUA-RAT (Ling et al., 2017), SQuAD (Rajpurkar et al., 2016)
BioMedical (entity recognition).	JNLPBA corpus (Kim et al., 2004)
Social media.	Twitter
Ontologies and online glossaries.	WordNet, Wiktionary
Named entities.	Freebase (Bollacker et al., 2008)
Veracity assessment.	FakeNews[2]

Table 2: Various datasets and resources used for rare word selection in CARD-660.

cial media (e.g., *2mrw* and *Mnhttn*), named entities (e.g., *Stephen_Hawking* and *Ursa_Major*), and domain specific terms (e.g., *erythroleukemia* and *NetMeeting*). Moreover, to have a rigorous testbed for subword representation techniques that emphasises the importance of semantic (rather than shallow) understanding of the words, the dataset contains several word pairs that have similar surface forms (hence, high string similarity) while being semantically distant, e.g., *infection-inflection* and *currency-concurrency*. There are also many compound words (e.g., *skyglow*, *musclebike*, and *logboat*) which makes the dataset particularly interesting for evaluating compositionality as well as for subword representation techniques.

3.2.3 Scoring and adjudication

Once the 660 word pairs were manually selected (by the first author), the initial scores were discarded and the words were shuffled (vertically and horizontally) to dispense any potential bias from the initial round of creation. Then, the pairs were assigned to 8 annotators (including all but first authors) who independently scored each and every pair according to the annotation guidelines (see Section 3.2.1). All annotators were PhD graduates or students in Computational Linguistics or related fields and were either native or fluent English speakers.

Once all pairs were scored by the annotators, we checked for disagreements. This check was intended to improve the dataset's quality through resolving simple annotation mistakes. For each annotator, we marked the i^{th} pair if for the assigned score s_i: $s_i \geq \mu_i + 1$ or $s_i \leq \mu_i - 1$, where μ_i is the average of the other seven annotators' scores for s_i. The annotator was then asked to (more carefully) re-score the marked pair by checking for its possible meanings. They were asked to keep their initial score if not convinced otherwise. The adjudication revealed that most disagreements were due to an annotator having misread a word or not been familiar with a specific meaning of it, or missing annotations. By average, 13.8% of the pairs were re-scored by each annotator.

4 Analysis

In this section we provide an analysis on the quality of CARD-660 from three different perspectives: distribution of scores, inter-annotator agreement, and consistency among annotators. We benchmark CARD-660 against the Stanford **RW** dataset and two standard word similarity datasets (cf. Section 2): SimVerb-3500 (**SV-3500**) and SimLex-999 (**SL-999**). The latter two datasets do not target rare words; however, given that their construction strategy is similar to that employed for creating RW (based on crowdsourcing), we included them in our analysis experiments to provide better insights. For the purpose of this evaluation, all the datasets were scaled to [0,10] to make them comparable.

4.1 Score Distribution

Figure 2 shows the distribution of pairs across the similarity scale, for CARD-660 and the three other datasets. As discussed in Section 2.1, RW is heavily biased towards the upper bound of the similar-

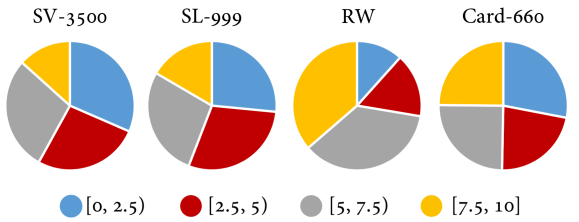

Figure 2: The distribution of word pairs across the four quartiles of the similarity scale for different datasets. A perfectly balanced dataset would have four equally sized slices.

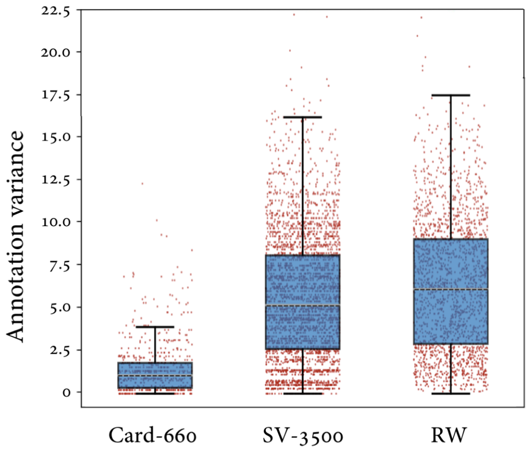

Figure 3: Annotation variance for word pairs across different datasets. Average variance for CARD-660 is 1.47, which is significantly lower than those for SV-3500 and RW: 5.64 and 6.34, respectively.

ity scale (with around 72% of the pairs in the upper half, i.e., $[5, 10]$). The skewed distribution in this dataset can be attributed to the automatic word pair selection from semantically-related words in WordNet (cf. Section 2.1). SV-3500 and SL-999 are skewed towards the lower bound, but to a smaller degree (around 59% of the pairs in $[0, 5)$). Thanks to the manual creation of CARD-660, we have a balanced set of pairs across the similarity scale (50-50% across the two halves).

4.2 Inter-Annotator Agreement

As mentioned in Section 2.1, IAA has been extensively used as a quality metric for word similarity datasets. Following standard practise, we measure two sets of IAA scores: (1) **Pairwise** is the averaged pairwise correlation between all possible rater pairings, and (2) **Mean** is the averaged correlation of each rater against the average of others.

Table 3 reports IAA statistics for CARD-660. Thanks to the manual scoring of the pairs by experts (as opposed to turkers), the IAA values for the dataset are very high, placing it among the best word similarity datasets in the literature. This is particularly interesting considering that, compared to standard word similarity datasets which contain mostly common words, our dataset comprises words that are semantically difficult to annotate due to their rare nature. The pairwise IAA score of 88.9 is significantly higher than the crowdsourced RW, with the estimated pairwise IAA score of around 40.0 (cf. Section 2.1). The same applies to other recent crowdsourced word similarity datasets for common words which usually report pairwise IAA scores below 70.0 (e.g., $\rho = 67.0$ for SL-999)[5].

	Mean		**Pairwise**	
	r	ρ	r	ρ
Initial	88.0±2.3	87.9±1.9	80.2±2.9	80.6±2.6
Final	93.5±1.4	93.1±1.2	88.9±1.7	88.9±1.7

Table 3: Inter-annotator agreement (IAA) scores before (*initial*) and after (*final*) adjudication (± standard deviation). IAA is shown in terms of Pearson r and Spearman ρ percentage correlations. The *final* scores are representative of the dataset's quality.

4.3 Consistency of Annotations

Despite being suitable for measuring linear relationships between scores, correlation cannot fully reflect the consistency between annotators. Two annotators can have perfect correlation, i.e., 1.0, even if they consistently provide different scores for the same pairs (therefore, having different average assigned scores). To check the consistency among annotators, i.e., if they had the same interpretation of the similarity scale, we compute variance across annotators for individual pairs.

The box and whisker (over scatter) plot in Figure 3 shows the distribution of annotator variances for the pairs in different datasets. Clearly, the score variances for CARD-660 are significantly lower than those for the two crowdsourced datasets, i.e., SimVerb-3500 and RW.[6] Specifically, for the majority of pairs in CARD-660 the annotation variance is lower than the other two datasets' first quartile (bottom of the blue square

---

[5] SimVerb-3500 reports a pairwise ρ of 84.0; however, our calculation did not agree with this figure. Personal communication with the authors revealed an issue in the computation of their IAA. The correct figure is instead 61.2.

[6] We are not able to report results for SimLex-999 since individual annotators' scores are not released for this dataset.

Embedding set	$\lvert V \rvert$	**Missed words**		**Missed pairs**		**Pearson r**		**Spearman ρ**	
		RW	CARD	RW	CARD	RW	CARD	RW	CARD
Glove Wikipedia-Gigaword (300d)	400K	7%	55%	12%	74%	34.9	15.1	34.4	15.7
Glove Common Crawl - uncased (300d)	1.9M	1%	36%	1%	50%	36.5	29.2	37.7	27.6
Glove Common Crawl - cased (300d)	2.2M	1%	29%	2%	44%	44.0	33.0	45.1	27.3
Glove Twitter (200d)	1.2M	29%	60%	48%	79%	17.7	13.7	15.3	11.7
Word2vec GoogleNews (300d)	3M	6%	48%	10%	75%	43.8	13.5	45.3	7.4
Word2vec Freebase (1000d)	1.4M	100%	85%	100%	92%	0.0	17.3	0.0	4.6
Dependency-based Wikipedia (300d)	174K	22%	60%	36%	80%	17.4	6.4	19.7	3.3
LexVec Common Crawl (300d)	2M	1%	41%	1%	55%	47.1	25.9	48.8	18.5
LexVec Wikipedia-NewsCrawl (300d)	370K	8%	58%	14%	78%	35.6	11.8	34.8	7.8
ConceptNet Numberbatch (300d)	417K	5%	37%	10%	53%	53.0	36.0	53.7	24.7
ConceptNet + Word2vec Freebase	1.6M	1%	22%	2%	45%	44.0	42.6	45.1	31.3
Glove cased CC + Word2vec Freebase	3.4M	11%	21%	10%	39%	53.0	38.8	53.7	32.7

Table 4: Pearson r and Spearman ρ correlation percentage performance of mainstream pre-trained word embeddings on the RW and CARD-660 datasets. Column $|V|$ shows the size of vocabulary for the corresponding embedding set.

which splits the lower 25% of the data from the top 75%). This indicates that our annotators had significantly higher degrees of agreement, reflecting the well-definedness of the similarity scale as well as the reliability of expert-based annotation (as opposed to crowdsourcing).

5 Evaluations

In the remainder of this paper, we provide two sets of experiments to showcase the challenging nature of our dataset. Specifically, in Section 5.1 we report the performance of common pretrained word embeddings on CARD-660, and in Section 5.2 we provide experimental results for state-of-the-art rare word representation techniques. In all experiments, we used the cosine similarity for comparing pairs of word embeddings.

5.1 Pre-trained Embeddings

As was mentioned earlier in the Introduction, it is not possible to enumerate the entire vocabulary of a natural language, even if massive corpora are used. A challenging rare word benchmark should ideally reflect this phenomenon. To verify this in our dataset, we experimented with a set of commonly used word embeddings trained on corpora with billions of tokens.

Table 4 provides correlation performance results for different embedding sets on the RW and CARD-660 datasets. Specifically, we considered different variants of Word2vec[7] (Mikolov et al., 2013) and Glove[8] (Pennington et al., 2014), two commonly-used word embeddings that are trained on massively large text corpora; Dependency-based embeddings[9] (Levy and Goldberg, 2014) which extends the Skip-gram model to handle dependency-based contexts; LexVec[10] (Salle et al., 2016) which improves the Skip-gram model to better handle frequent words; and ConceptNet Numberbatch[11] (Speer et al., 2017) which exploits lexical knowledge from multiple resources, such as Wiktionary and WordNet, and was the best performing system in SemEval 2017 Task 2. In the last two rows of the Table we also report results for two hybrid embeddings constructed by combining the pre-trained Freebase Word2vec, which mostly comprises named entities, with two of the best performing embeddings evaluated on the dataset. Given that the word embeddings are not comparable across two different spaces, we only compute the similarity between a pair only if both words are covered in the same space (with priority given to the non-Freebase embedding).

As can be seen in the Table, many of the embeddings yield high coverage for the RW dataset, with those trained on the Common Crawl (CC) corpus providing near full coverage. This highlights the limited vocabulary of the dataset (which is bound to that of WordNet). Also, many of

---

[7] https://code.google.com/archive/p/word2vec/

[8] https://nlp.stanford.edu/projects/glove/

[9] http://u.cs.biu.ac.il/~yogo/data/syntemb/deps.words.bz2

[10] https://github.com/alexandres/lexvec

[11] https://github.com/commonsense/conceptnet-numberbatch

the embeddings attain performance around 40.0 on RW, which is higher than the estimated IAA of the dataset. In contrast, CARD-660 proves to be significantly more challenging, with the highest coverage model (Glove CC and Word2vec Freebase hybrid model) missing around 40% of the pairs. Also, the best performance of 42.6 (r) and 32.7 (ρ) are substantially (around 50.0) lower than the IAA for the dataset (see Table 3).

5.2 Rare Word Representation Techniques

Rare and unseen word representation has been an active field of research during the past few years, with many different techniques proposed. In this experiment, we evaluate the performance of some of recent models on our dataset. These techniques can be broadly classified into two categories. The first group exploits the knowledge encoded for a rare word in external lexical resources (Section 5.2.1), whereas the second induces embeddings for rare words by extending the semantics of its subword units (Section 5.2.2).

5.2.1 Resource-based models

The basic assumption here is that a lexical resource, such as dictionary, provides high coverage for words in a language, even if they are rare. Resource-based models usually rely on WordNet as their external resource and estimate the embedding for a rare word by exploiting different types of lexical knowledge encoded for it in the resource. The **definition centroid** model of Lazaridou et al. (2017) takes WordNet word glosses (definitions) as semantic clue. An embedding is induced for an unseen word by averaging the content words' embeddings in its definition.[12] The **definition LSTM** strategy of Bahdanau et al. (2017) extends the centroid model by encoding the definition using an LSTM network (Hochreiter and Schmidhuber, 1997), in order to better capture the semantics and word order in the definition. **SemLand** (Pilehvar and Collier, 2017) also uses WordNet, but takes a different approach which benefits from the graph structure of WordNet. For an unseen word, SemLand extracts the set of its semantically related words from WordNet and induces an embedding for the unseen word by combining pre-trained embeddings for the related words.

5.2.2 Subword models

Resource-based models fall short of inducing embeddings for words that are not covered in the lexical resource. Subword models alleviate this limitation by breaking the word into its subword (Pinter et al., 2017; Bojanowski et al., 2017) or morphological units (Luong et al., 2013; Botha and Blunsom, 2014; Soricut and Och, 2015) and induce an embedding by composing the information available for these. **FastText** (Bojanowski et al., 2017) is one of the popular approaches of this type. The model first splits the unseen word into character ngrams (by default, 3- to 6-grams) and then computes the unseen word's embedding as the centroid of the embeddings of these character n-grams (which are available as a result of a specific training). We also report results for **Mimick** (Pinter et al., 2017), one of the most recent subword models. The technique learns a mapping function from strings to embeddings by training a Bi-LSTM network that encodes character sequences of a word to its pre-trained embedding.

5.3 Experimental Setup

We report results for the five techniques discussed in Sections 5.2.2 and 5.2.1. We used two of the best performing embedding sets, i.e., Glove cased CC and ConceptNet Numberbatch, to train the models (except FastText for which we use the pre-trained WikiNews subword embeddings[13]). In fact, the models were expected to provide improvements over these baseline embeddings by filling their gaps for unseen words.

Mimick was trained with the default parameters,[14] except for the hidden units which we set to 100, instead of the original 50, since the target embeddings in our experiments were larger (300d compared to 128d of the original model). For the Definition LSTM model, the input definitions were represented as sequences of 50d word embeddings, encoded using an LSTM layer of 100 units, and then passed to a dense layer with 300 neurons with linear activation function. The training was carried out with Mean Squared Error loss and the RMSprop optimizer, for 100 epochs with batch size 64.

---

[12] The original model is multimodal (text and images). Given that our focus is on texts, we follow Herbelot and Baroni (2017) and use the text modality only.

[13] https://fasttext.cc/docs/en/english-vectors.html

[14] https://github.com/yuvalpinter/Mimick/

Model	Missed words		Missed pairs		Pearson r		Spearman ρ	
	RW	CARD	RW	CARD	RW	CARD	RW	CARD
ConceptNet Numberbatch (300d)	5%	37%	10%	53%	53.0	36.0	53.7	24.7
+ Mimick (Pinter et al., 2017)	0%	0%	0%	0%	56.0	34.2	57.6	**35.6**
+ Definition centroid (Herbelot and Baroni, 2017)	0%	29%	0%	43%	59.1	42.9	60.3	33.8
+ Definition LSTM (Bahdanau et al., 2017)	0%	25%	0%	39%	58.6	41.8	59.4	31.7
+ SemLand (Pilehvar and Collier, 2017)	0%	29%	0%	43%	**60.5**	**43.4**	**61.7**	34.3
Glove Common Crawl - cased (300d)	1%	29%	2%	44%	44.0	33.0	45.1	27.3
+ Mimick (Pinter et al., 2017)	0%	0%	0%	0%	**44.7**	23.9	45.6	29.5
+ Definition centroid (Herbelot and Baroni, 2017)	0%	21%	0%	35%	43.5	35.2	45.1	31.7
+ Definition LSTM (Bahdanau et al., 2017)	0%	20%	0%	33%	24.0	23.0	22.9	19.6
+ SemLand (Pilehvar and Collier, 2017)	0%	21%	0%	35%	44.3	**39.5**	**45.8**	**33.8**
FastText (Bojanowski et al., 2017)	0%	3%	0%	5%	46.3	19.0	48.2	20.4

Table 5: Correlation performance of different rare and unseen word representation techniques on the Stanford RW and CARD-660 datasets (the best performance in each batch shown in bold; the overall best underlined).

5.4 Experimental Results

Table 5 reports the performance of different rare word representation techniques. Both pre-trained embeddings outperform the IAA of RW, with Glove covering 98% of the pairs. This severely limits the room for further meaningful experiments on the dataset. In contrast, on CARD-660 and similarly to the previous experiment, there are substantial gaps between IAA (cf. Table 3) and the best-performing models: SemLand and Mimick, with the respective figures of 45.5 (r) and 53.3 (ρ). These gaps suggest a difficult dataset which can serve future research in subword and rare word representation as a reliable benchmark.

The definition centroid model proves effective, despite its simplicity, whereas the WordNet-based SemLand provides the best results in most of the settings. Being constrained to the vocabulary of WordNet, the RW dataset does not constitute a challenging benchmark for WordNet-based models, with most of them providing near full coverage. However, these techniques are not as effective on our dataset, with the best WordNet-based model still missing around 33% of the pairs (with Glove pre-trained embeddings).

The CARD-660 dataset also proves a very difficult benchmark for subword models. Despite providing near full coverage, these models are unable to consistently improve the pre-trained word embedding baseline. This would suggest that the simple strategy of backing off to a word's characters might not always provide reliable means of estimating its semantics (e.g., the single-morpheme word *galaxy*, or the exocentric compound *honeymoon*). The results encourage further research on a more semantically-oriented handling of subwords, through learning more effective splitting and composition techniques.

6 Conclusions

Thanks to a carefully designed procedure and an expert-based curation, CARD-660 provides multiple advantages over existing benchmarks, including a very high IAA (average pairwise correlation of around 0.90). A series of experiments was carried out on the dataset, leading to two main conclusions: (1) the dataset proved a very challenging benchmark, with the best pre-trained embedding model still missing around 40% of the word pairs and the best rare word representation model hardly crossing into 40.0s (correlation performance); and (2) knowledge-based models are not enough to provide high coverage whereas subword models, which provide near-full coverage, are not semantically as effective. The significant gap between state of the art and IAA (around 50.0) encourages future research to take this dataset as a challenging, yet reliable, evaluation benchmark.

Acknowledgments

We gratefully acknowledge the funding support of EPSRC (N. Collier and D. Kartsaklis - Grant No. EP/M005089/1) and MRC (M. T. Pilehvar) Grant No. MR/M025160/1 for PheneBank. We would also like to thank Andreas Chatzistergiou, Costanza Conforti, Gamal Crichton, Milan Gritta, and Ehsan Shareghi for their contribution in creating the dataset.

References

Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2017. Learning to compute word embeddings on the fly. *CoRR*, abs/1706.00286.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, Vancouver, Canada.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of ICML*, pages 1899–1907, Beijing, China.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49(1):1–47.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26, Vancouver, Canada.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Lev Finkelstein, Gabrilovich Evgenly, Matias Yossi, Rivlin Ehud, Solan Zach, Wolfman Gadi, and Ruppin Eytan. 2002. Placing search in context: The concept revisited. *ACM Transactions of Information Systems*, 20(1):116–131.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of EMNLP*, pages 2173–2182.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 304–309, Copenhagen, Denmark.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

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

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, pages 70–75, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive Science*, 41(S4):677–705.

Ira Leviant and Roi Reichart. 2015. Judgment language matters: Multilingual vector space models for judgment language aware lexical semantics. *CoRR*, abs/1508.00106.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of CoNLL*, pages 104–113, Sofia, Bulgaria.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL-HLT*, pages 142–150, Portland, Oregon, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop at ICLR*, Scottsdale, Arizona.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*, pages 1532–1543, Doha, Qatar.

Mohammad Taher Pilehvar and Nigel Collier. 2017. Inducing embeddings for rare and unseen words by leveraging lexical resources. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 388–393, Valencia, Spain.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, Copenhagen, Denmark.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Alexandre Salle, Aline Villavicencio, and Marco Idiart. 2016. Matrix factorization using window sampling and negative sampling for improved word representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 419–424, Berlin, Germany. Association for Computational Linguistics.

Irina Sergienya and Hinrich Schütze. 2015. Learning better embeddings for rare words using distributional representations. In *Proceedings of EMNLP*, pages 280–285.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of NAACL-HLT*, pages 1627–1637, Denver, Colorado.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*, pages 4444–4451.

Dongqiang Yang and David M. W. Powers. 2005. Measuring semantic similarity in the taxonomy of WordNet. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science*, volume 38, pages 315–322, Newcastle, Australia.

Leveraging Gloss Knowledge in Neural Word Sense Disambiguation by Hierarchical Co-Attention

Fuli Luo[1], Tianyu Liu[1], Zexue He[2], Qiaolin Xia[1], Zhifang Sui[1] and Baobao Chang[1]
[1]Key Lab of Computational Linguistics, School of EECS, Peking University
[2]Computer Science and Technology, College of IST, Beijing Normal University
`{luofuli, tianyu0421, xql, chbb, szf}@pku.edu.cn`
`zexueh@mail.bnu.edu.cn`

Abstract

The goal of Word Sense Disambiguation (WSD) is to identify the correct meaning of a word in the particular context. Traditional supervised methods only use labeled data (context), while missing rich lexical knowledge such as the gloss which defines the meaning of a word sense. Recent studies have shown that incorporating glosses into neural networks for WSD has made significant improvement. However, the previous models usually build the context representation and gloss representation *separately*. In this paper, we find that the learning for the context and gloss representation can benefit from each other. Gloss can help to highlight the important words in the context, thus building a better context representation. Context can also help to locate the key words in the gloss of the correct word sense. Therefore, we introduce a co-attention mechanism to generate co-dependent representations for the context and gloss. Furthermore, in order to capture both word-level and sentence-level information, we extend the attention mechanism in a hierarchical fashion. Experimental results show that our model achieves the state-of-the-art results on several standard English all-words WSD test datasets.

1 Introduction

Word Sense Disambiguation (WSD) is a crucial task and long-standing problem in Natural Language Processing (NLP). Previous researches mainly exploit two kinds of resources. Knowledge-based methods (Lesk, 1986; Moro et al., 2014; Basile et al., 2014) exploit the lexical knowledge like gloss to infer the correct senses of ambiguous words in the context. However, supervised feature-based methods (Zhi and Ng, 2010; Iacobacci et al., 2016) and neural-based methods (Kågebäck and Salomonsson, 2016; Raganato et al., 2017a) usually use labeled data to train one or more classifiers.

Context	As they often **play** football together, they know each other quite well
Glosses	g_1: participate in games or sports
	g_2: perform music on an instrument
	g_3: behave in a certain way

Table 1: An example of the context and three glosses of different senses according to the target word ***"play"***. It shows that the words *"games/sports"* in the gloss g_1 can help to highlight the important words *"football"* in the context and ignore the words *"know each other"* which are useless for distinguishing the sense of word *"play"*. Meanwhile, the context can potentially help to stress on the words *"games/sports"* of the gloss g_1 which is actually the correct sense for the target word.

Although both lexical knowledge (especially gloss) and labeled data are of great help for WSD, previous supervised methods rarely take the integration of knowledge into consideration. To the best of our knowledge, Luo et al. (2018) are the first to directly incorporate the gloss knowledge from WordNet into a unified neural network for WSD. This model *separately* builds the context representation and the gloss representation as distributed vectors and later calculates their similarity in a memory network. However, we find that the learning of the representations of the context and gloss can contribute to each other. We use an example to illustrate our ideas. Table 1 shows that the red words are more important than the blue words when distinguishing the sense of the target word. In other words, we should pay more attention to the words which can "overlap" between the context and the gloss when generating the representations of context and gloss. Therefore, we introduce a co-attention mechanism to model the mutual influence between the representations of context and gloss.

Moreover, we find that both word-level and sentence-level information are crucial to WSD. As

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1402–1411
Brussels, Belgium, October 31 - November 4, 2018.

shown in Table 1, the local word *"football"* is crucial for distinguishing the sense of word *"play"*. However, in more complex sentences such as *"Investors played it carefully for maximum advantage"* [1], sentence-level information is necessary. Therefore, we extend the co-attention model in a hierarchical fashion to capture both the word-level and sentence-level semantic information.

The main contributions are listed as follows.

- We propose a novel way to integrate gloss knowledge into a neural network for WSD via a co-attention mechanism in order to build better representations of context and gloss. In this way, our model can benefit from both labeled data and lexical knowledge.
- We further extend the attention mechanism into a hierarchical architecture, since both word-level and sentence-level information are crucial to disambiguating the word sense.
- We conduct a series of experiments, which show that our models outperform the state-of-the-art systems on several standard English all-words WSD test datasets.

2 Related work

Lexical knowledge is a fundamental component of Word Sense Disambiguation and provides rich resources which are essential to associate senses with words (Navigli, 2009). Unsupervised knowledge-based methods have shown the effectiveness of textual knowledge such as gloss (Lesk, 1986; Basile et al., 2014) and the structural knowledge (Moro et al., 2014; Agirre et al., 2014) of the lexical databases. However, the prime shortcoming of knowledge-based methods is that they have worse performance than supervised methods, but they have wider coverage for the polysemous words, thanks to the use of large-scale knowledge resources (Navigli, 2009).

There are many other tasks such as Chinese Word Segmentation (Zhang et al., 2018), Language Modeling (Ahn et al., 2016), and LSTMs (Xu et al., 2016; Yang and Mitchell, 2017) show that integrating knowledge and labeled data into a unified system can achieve better performance than other methods which only learn from large scale labeled data. Therefore, it's a promising and challenging study to integrate labeled data and lexical knowledge into a unified system.

A few recent studies of WSD have exploited several ways to incorporate lexical resources into supervised systems. In the field of traditional feature-based methods (Chen et al., 2015; Rothe and Schütze, 2015), they usually utilize knowledge (to train word sense embeddings) as features of the classifier like the support vector machine (SVM). In the field of neural-based methods, Raganato et al. (2017a) regard lexical resource LEX which is extracted from the WordNet as an auxiliary classification task, and propose a multi-task learning framework for WSD and LEX. Luo et al. (2018) integrate the context and glosses of the target word into a unified framework via a memory network. It encodes the context and glosses of the target word *separately*, and then models the semantic relationship between the context vector and gloss vector in the memory module. What's more, Luo et al. (2018) utilize much more knowledge about gloss via its semantic relations such as hypernymy and hyponymy in WordNet. All studies listed above show that integrating lexical resources especially gloss into supervised systems of WSD can significantly improve the performance. Therefore, we follow this direction and seek a new way of better integrating gloss knowledge.

Instead of building representations for context and gloss separately, we use the inner connection between the gloss and the context to promote the representation of each other. The interaction process can be modeled by a co-attention mechanism which has made great progress in the question answering task (Xiong et al., 2016; Seo et al., 2016; Hao et al., 2017; Lu et al., 2016). We are enlightened by this iterative procedure and introduce it into WSD. We then make some adaptations to the output of the original co-attention model to get the score of each word sense.

3 The Co-Attention Model for WSD

In this section, we first give an overview of the **CAN**: co-attention neural network for WSD (Figure 1). And then, we extend it into a hierarchical architecture **HCAN** (Figure 2).

3.1 Overview

The overall architecture of the proposed non-hierarchical co-attention model is shown in Figure 1. It consists of three parts:

---

[1] *Play* in the sentence means *behave in a certain way*.

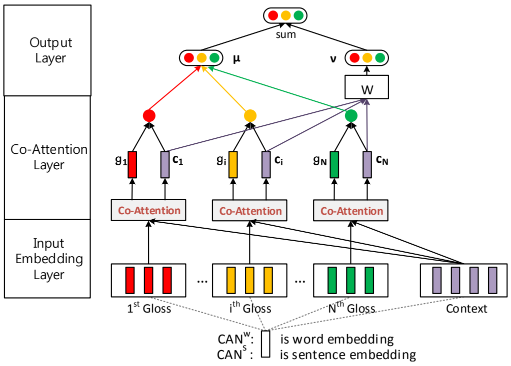

Figure 1: The proposed co-attention neural network.

- **Input Embedding Layer**: First of all, we encode the input context and each gloss [2] into distributed representations C and G, which are also called embeddings in the paper. In Figure 1, if C and G are word embeddings, we call the model CAN^w in the paper. If C and G are sentence embeddings, we call the model CAN^s.
- **Co-Attention Layer**: Then, each co-attention mechanism in this layer generates a context vector and a gloss vector according to the corresponding gloss and context representations. The outputs of the co-attention layer are N pairs of context vector and gloss vector.
- **Output Layer**: Finally, the output layer takes the N pairs of context vector and gloss vector as inputs and calculates the score of each word sense.

Figure 1 shows the non-hierarchical co-attention model which generates either word-level representations (CAN^w) or sentence-level representations (CAN^s). Since both the word-level and sentence-level representations can help to disambiguate the word sense, we extend CAN into a hierarchical model, named as HCAN (Figure 2). The extensions of each layer are listed as follows:

1. The input embedding layer is extended to two sub-layers in the hierarchical architecture which encodes both word-level and sentence-level representations.
2. The co-attention layer is also extended to two attention layers for capturing two different levels' attention.
3. The output layer merges the outputs of the two levels' co-attention layers and generates a sense probability over all word senses.

Since the non-hierarchical model CAN is a subset or a simplified version of the hierarchical model HCAN, the next sections are organized to illustrate the hierarchical co-attention model HCAN [3] shown in Figure 2.

3.2 Input Embedding Layer

We denote each input sentence (context or gloss) as a sequence of words $[x_1, x_2, \ldots, x_{T_x}]$, where T_x is the length of the input sentence.

3.2.1 Word Embedding

After looking up a pre-trained word embedding matrix $E_w \in \mathbb{R}^{d_w \times V}$, we transfer a one-hot vector x_i into a d_w-dimensional vector e_i. We treat $[e_1, e_2, \ldots, e_{T_x}]$ as the word-level representations of the sentence. Specifically, context's word-level representations are denoted as $[e^c_1, e^c_2, \ldots, e^c_n]$ and i-th gloss's word-level representations are denoted as $[e^{g_i}_1, e^{g_i}_2, \ldots, e^{g_i}_m]$, where n and m represent the max length of context and gloss.

3.2.2 Sentence Embedding

We utilize a bi-directional long short-term memory network (Bi-LSTM) to generate the hidden states of the input sentence. Each hidden state h_i is computed by the concatenation of the *forward* hidden state $\overrightarrow{h_i}$, and *backward* hidden state $\overleftarrow{h_i}$. So we treat $[h_1, h_2, \ldots, h_{T_x}]$ as the sentence-level representations. Specifically, context's sentence-level representations are denoted as $[h^c_1, h^c_2, \ldots, h^c_n]$ and i-th gloss's sentence-level representations are denoted as $[h^{g_i}_1, h^{g_i}_2, \ldots, h^{g_i}_m]$.

3.3 Co-Attention Layer

3.3.1 Co-Attention Mechanism

The right part of Figure 2 illustrates the co-attention mechanism which is the most crucial part of the model. The inputs are context representations $C \in \mathbb{R}^{d \times n}$ and gloss representations $G \in \mathbb{R}^{d \times m}$, where d is the dimension of the input representation vector. The outputs are the gloss-aware context vector $c \in \mathbb{R}^d$ and the context-aware gloss vector $g \in \mathbb{R}^d$. Therefore, we can define the co-attention mechanism as a function

$$(c, g) = CoAt(C, G) \tag{1}$$

---

[2] There are N glosses in total, where N is the sense number of the target word in the context.

[3] CAN^w and CAN^s will also be expressed in Section 3.4.

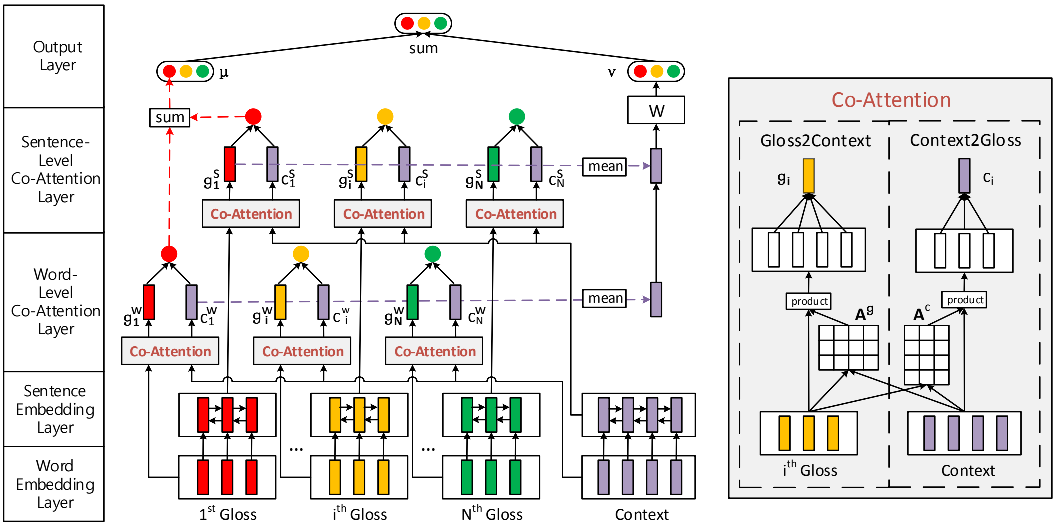

Figure 2: Hierarchical co-attention model for WSD.

Next, we give the detailed definition of the co-attention mechanism function $CoAt$. We begin to compute a similarity matrix A, in which each element A_{ij} indicates the similarity between i-th context word and j-th gloss word. The similarity matrix A is computed by

$$A = C^{\top}UG \in \mathbb{R}^{n\times m} \tag{2}$$

where $U \in \mathbb{R}^{d\times d}$ is a trainable parameter.

Based on the similarity matrix A, we can compute the gloss-to-context attention matrix A^c and context-to-gloss attention matrix A^g.

Gloss-to-Context Attention. Since each gloss word may focus on different context words, we can generate a context representation which is aware of a particular gloss word. Note that each element of A in the j-th column indicates the similarity between j-th gloss word and each context word. Thus, we can get the attention weight for each context word through a softmax function across the column of A:

$$A^c_{:j} = softmax(A_{:j}) \tag{3}$$

where $A_{:j}$ denotes j-th column of A and $A^c_{:j}$ denotes j-th column of $A^c \in \mathbb{R}^{n\times m}$.

Hence we can get the gloss-aware context representations $\hat{C}$ by a product of the initial context representations C and attention weight matrix A^c:

$$\hat{C} = CA^c \in \mathbb{R}^{d\times m} \tag{4}$$

Note that j-th column in $\hat{C}$ means the context representation according to the j-th gloss word. Therefore, we can get the final context vector c by summing across the column of $\hat{C}$:

$$c = \sum_j \hat{C}_{:j} \in \mathbb{R}^d \tag{5}$$

Context-to-Gloss Attention. Conversely, each context word may focus on different gloss words, we can generate a gloss representation which is aware of a particular context word. Since each element of A in the i-th row indicates the similarity between i-th context word and each gloss word, we can get the attention weight of each gloss word through a softmax function across the row of A (or across the column of $A^{\top}$)

$$A^g_{:j} = softmax(B_{:j}) \tag{6}$$

where $B = A^{\top}$, $B_{:j}$ denotes j-th column of B (also j-th row of A) and $A^g_{:j}$ denotes j-th column of $A^g \in \mathbb{R}^{m\times n}$.

Now we can get the context-aware gloss representations in the same way as Equation 4:

$$\hat{G} = GA^g \in \mathbb{R}^{d\times n} \tag{7}$$

Note that j-th column in $\hat{G}$ denotes the gloss representation according to j-th context word. Therefore, we can get the final gloss vector g by summing across the column of $\hat{G}$:

$$g = \sum_j \hat{G}_{:j} \in \mathbb{R}^d \tag{8}$$

3.3.2 Word-Level Co-Attention Layer

Since there are N glosses according to N different word senses, we use N independent co-attention mechanisms in both word-level and sentence-level co-attention layers. And each layer shares a same parameter U in Equation 2. For i-th word-level co-attention mechanism, the inputs are word embeddings of the context and i-th gloss (in Section 3.2.1). Define $C^w = [e_1^c, e_2^c, \dots, e_n^c]$ and $G_i^w = [e_1^{g_i}, e_2^{g_i}, \dots, e_m^{g_i}]$, thus the outputs of i-th word-level co-attention mechanism are computed as

$$(c_i^w, g_i^w) = CoAt(C^w, G_i^w) \tag{9}$$

Inspired by the well-known Lesk algorithm (Lesk, 1986) and its variants (Basile et al., 2014), the score of the i-th word sense can be computed as the similarity of the context vector c_i^w and the gloss vector g_i^w:

$$\beta_i^w = c_i^w \cdot g_i^w \tag{10}$$

The word-level context embedding vector $\hat{c}^w$ can be computed as the average of the N gloss-aware context vectors c_i^w:

$$\hat{c}^w = \frac{1}{N}\sum_{i=1}^{N} c_i^w \tag{11}$$

3.3.3 Sentence-Level Co-Attention Layer

Same to word-level co-attention layer, for the i-th co-attention mechanism, the inputs of sentence-level co-attention layer are Bi-LSTM hidden states of context and the i-th gloss (in Section 3.2.2). Define $C^s = [h_1^c, h_2^c, \dots, h_n^c]$ and $G_i^s = [h_1^{g_i}, h_2^{g_i}, \dots, h_m^{g_i}]$, thus the outputs of i-th sentence-level co-attention mechanism are computed as

$$(c_i^s, g_i^s) = CoAt(C^s, G_i^s) \tag{12}$$

Like Equation 10, we can also calculate a sentence-level score for the i-th word sense by a dot product of the context vector c_i^s and the gloss vector g_i^s:

$$\beta_i^s = c_i^s \cdot g_i^s \tag{13}$$

The sentence-level context embedding vector $\hat{c}^s$ is also computed as the average of N gloss-aware context vectors c_i^s:

$$\hat{c}^s = \frac{1}{N}\sum_{i=1}^{N} c_i^s \tag{14}$$

3.4 Output Layer

The output layer aims to calculate the scores of N senses of the target word x_t and finally outputs a sense probability distribution over the N senses. The final score of each sense is a weighted sum of two values: μ and ν. μ is the similarity score of gloss and context, which reveals the influence of knowledge. ν is generated by the context vector through a linear projection layer, which reveals the influence of labeled data. Finally, the probability distribution $\hat{y}$ over all the senses of the target word is computed as

$$\hat{y} = softmax(\lambda_{x_t}\mu + (1 - \lambda_{x_t})\nu))$$

where $\lambda_{x_t} \in [0, 1]$ is the parameter for word x_t.

For the non-hierarchical model CAN in Figure 1, the final score μ and ν are generated only by the outputs of the one level co-attention layer. Specifically, for the word-level co-attention model CAN^w:

$$\mu = [\beta_1^w, \beta_2^w, \dots, \beta_N^w] \tag{15}$$

$$\nu = W_{x_t}\hat{c}^w + b_{x_t} \tag{16}$$

For the sentence-level co-attention model CAN^s:

$$\mu = [\beta_1^s, \beta_2^s, \dots, \beta_N^s] \tag{17}$$

$$\nu = W_{x_t}\hat{c}^s + b_{x_t} \tag{18}$$

For the hierarchical co-attention model HCAN in Figure 2, the outputs of the word and sentence level layer are merged together to generate the final results. Therefore, the final similarity score between i-th gloss and context is computed as the weighted sum of word-level score β_i^w and sentence-level score β_i^s:

$$\beta_i = \alpha\beta_i^w + (1 - \alpha)\beta_i^s \tag{19}$$

Meanwhile, the final context embedding vector is also generated by a combination of two levels' context embedding vector: $\hat{c}^w$ and $\hat{c}^s$. In order to transfer from word-level encoding space to sentence-level encoding space, we introduce a non-linear projection layer on top of the word-level context vector $\hat{c}^w$. Therefore, the final context embedding vector $\hat{c}$ is generated by

$$\hat{c} = tanh(W\hat{c}^w + b) + \hat{c}^s \tag{20}$$

In total, for the hierarchical co-attention model HCAN:

$$\mu = [\beta_1, \beta_2, \dots, \beta_N] \tag{21}$$

$$\nu = W_{x_t}\hat{c} + b_{x_t} \tag{22}$$

It's noteworthy that in Equation 16, 18 and 22, each ambiguous word x_t has its corresponding weight matrix W_{x_t} and bias b_{x_t}.

During training, all model parameters θ are jointly learned by minimizing a cross-entropy loss between $\hat{y}$ and the true label y.

$$L(\theta) = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{N_i} y_{ij}\log\hat{y}_{ij} \tag{23}$$

where M is the number of examples in the dataset, N_i is the word sense number of i-th example, y_{ij} and $\hat{y}_{ij}$ are the true and predict probability of the i-th example belongs to j-th label.

4 Experiments and Evaluation

4.1 Datasets

Validation and Evaluation Datasets: We evaluate our model on several English all-words WSD datasets. For a fair comparison, we use the benchmark datasets proposed by Raganato et al. (2017b) which include five standard all-words fine-grained WSD datasets from the Senseval and SemEval competitions:

- Senseval-2 (Edmonds and Cotton, 2001, **SE2**): It consists of 2282 sense annotations, including nouns, verbs, adverbs and adjectives.

- Senseval-3 task 1 (Snyder and Palmer, 2004, **SE3**): It consists of 1850 sense annotations from three different domains (editorial, news story and fiction), including nouns, verbs, adverbs and adjectives.

- SemEval-07 task 17 (Pradhan et al., 2007, **SE7**): It consists of 455 sense annotations of nouns and verbs, which is the smallest among the five datasets. Like Luo et al. (2018) and Raganato et al. (2017a), we choose SE7 as the validation set.

- SemEval-13 task 12 (Navigli et al., 2013, **SE13**): It consists of 1644 sense annotations from thirteen documents of various domains. SE13 contains nouns only.

- SemEval-15 task 13 (Moro and Navigli, 2015, **SE15**): It's the latest WSD dataset, which consists of 1022 sense annotations from three heterogeneous domains.

Training Dataset: SemCor 3.0 is the largest manually annotated corpus for WSD, which was also used by Luo et al. (2018), Raganato et al. (2017a), Raganato et al. (2017b), Iacobacci et al. (2016), Zhi and Ng (2010), etc. It consists of 226,036 sense annotations from 352 documents, which includes nouns, verbs, adverbs and adjectives.

Knowledge Base: The original WordNet version of sense inventory for SemCor 3.0, SE2, SE3, SE7, SE13, SE15 are 1.4, 1.7, 1.7.1, 2.1, 3.0 and 3.0, respectively. Raganato et al. (2017b) map all the sense annotations in the training and test datasets to WordNet 3.0 via a semi-automatic method. Therefore, We choose WordNet 3.0 as the sense inventory for extracting the gloss.

Data	Noun	Verb	Adj	Adv
SE2	1066	517	445	254
SE3	900	588	350	12
SE7	159	296	0	0
SE13	1644	0	0	0
SE15	531	251	160	80
SemCor	87002	88334	31753	18947

Table 2: Statistics of the different parts of speech annotations in English all-words WSD train and test datasets.

.

4.2 Settings

We use the validation set (SE7) to find the optimal hyper parameters of our models: the word embedding size d_w, the hidden state size d_s of LSTM, the optimizer, etc. However, since there are no adverbs and adjectives in SE7, we randomly sample some adverbs and adjectives from training dataset into SE7 for validation. We use the pre-trained word embeddings [4]. The hidden state size d_s is 256. The mini-batch size is set to 32. The optimizer is Adam (Kingma and Ba, 2014) with 0.001 initial learning rate. In order to avoid over-fitting, we use dropout regularization on the outputs of LSTM and set drop rate to 0.5. Orthogonal initialization is used for initialing weights in LSTM and random uniform initialization with range [-0.1, 0.1] is used for others. Training runs for up to 50 epochs with early stopping if the validation loss doesn't improve within the last 5 epochs.

---

[4] We download the pre-trained word embeddings from `https://github.com/stanfordnlp/GloVe`

	Test Datasets				Concatenation of Test Datasets				
System	SE2	SE3	SE13	SE15	Noun	Verb	Adj	Adv	**All**
MFS baseline*	65.6	66.0	63.8	67.1	67.7	49.8	73.1	80.5	65.5
Lesk$_{ext+emb}$ (Basile et al., 2014)	63.0	63.7	66.2	64.6	70.0	51.1	51.7	80.6	64.2
Babelfy (Moro et al., 2014)	67.0	63.5	66.4	70.3	68.9	50.7	73.2	79.8	66.4
IMS (Zhi and Ng, 2010)*	70.9	69.3	65.3	69.5	70.5	55.8	75.6	82.9	68.9
IMS$_{+emb}$ (Iacobacci et al., 2016)*	72.2	70.4	65.9	71.5	71.9	56.6	75.9	84.7	70.1
Bi-LSTM (Kågebäck and Salomonsson, 2016)*	71.1	68.4	64.8	68.3	69.5	55.9	76.2	82.4	68.4
Bi-LSTM$_{+att.+LEX}$ (Raganato et al., 2017a)	72.0	69.4	66.4	72.4	71.6	57.1	75.6	83.2	69.9
Bi-LSTM$_{+att.+LEX+POS}$ (Raganato et al., 2017a)	72.0	69.1	66.9	71.5	71.5	57.5	75.0	83.8	69.9
GAS (Linear) (Luo et al., 2018)	72.0	70.0	66.7	71.6	71.7	57.4	76.5	83.5	70.1
GAS (Concatenation) (Luo et al., 2018)	72.1	70.2	67.0	71.8	72.1	57.2	76.0	84.4	70.3
GAS$_{ext}$ (Linear) (Luo et al., 2018)	72.4	70.1	67.1	72.1	71.9	58.1	76.4	84.7	70.4
GAS$_{ext}$ (Concatenation) (Luo et al., 2018)	72.2	**70.5**	67.2	72.6	72.2	57.7	76.6	**85.0**	70.6
CAN^{w}	72.3	69.8	65.5	71.1	71.1	57.3	76.5	84.7	69.8
CAN^{s}	72.2	70.2	**69.1**	72.2	**73.5**	56.5	76.6	80.3	70.9
HCAN	**72.8**	70.3	68.5	**72.8**	72.7	**58.2**	**77.4**	84.1	**71.1**

Table 3: F1-score (%) for fine-grained English all-words WSD on the test sets. **Bold** font indicates best systems. The * represents the systems which don't use any lexical knowledge. The five blocks list the baseline, 2 knowledge-based systems, 2 supervised feature-based systems, 7 neural-based systems and our models, respectively.

4.3 Results and Discussion

4.3.1 English all-words results

Table 3 shows the results on four test datasets and different parts of speech. Note that all the systems in Table 3 are trained on SemCor 3.0.

In the first block, we show the MFS baseline, which simply selects the most frequent sense in the training dataset.

In the second block, we show two latest knowledge-based (unsupervised) systems. Lesk$_{ext+emb}$ is a variant of the well-known Lesk algorithm (Lesk, 1986) which computes the overlap of gloss and context as the score of word sense. Babelfy (Moro et al., 2014) is a graph-based system performed on BabelNet (Navigli and Ponzetto, 2012). We can find that MFS is a strong baseline for knowledge-based systems.

In the third block, we show two traditional supervised systems which only learn from labeled data based on manual designed features. IMS (Zhi and Ng, 2010) is a flexible framework which trains K SVM classifiers for K polysemous words. Its variant IMS$_{+emb}$ (Iacobacci et al., 2016) adds word embedding features into IMS. Both of them train a dedicated classifier for each word individually. In other words, each target word has its own parameters. Therefore, IMS$_{+emb}$ is a hard to beat system for many neural networks which also only uses labeled data but builds a unified system for all the polysemous words.

In the fourth block, we show four latest neural networks. Except for Bi-LSTM (Kågebäck and Salomonsson, 2016), which is a baseline for neural models, the others all utilize not only labeled data but also lexical knowledge. Bi-LSTM$_{+att.+LEX}$ (Raganato et al., 2017a) and its variant Bi-LSTM$_{+att.+LEX+POS}$ are multi-task learning frameworks for WSD, POS tagging and LEX with context self-attention mechanism. GAS (Luo et al., 2018) is a gloss-augmented neural network in an improved memory network paradigm. The best neural network is GAS$_{ext}$ which extends from GAS and uses more gloss knowledge via the semantic relations in WordNet. [5]

In the last block, we give the performance of our proposed co-attention models for WSD. We can see that our best model HCAN improves state-of-the-art result by 0.5% on the concatenation of four datasets. Even though we use less gloss knowledge than the previous best system GAS$_{ext}$, our co-attention models can still get the best results on three test datasets. For non-hierarchical models, CAN^{s} performs much better than the CAN^{w}, which reveals that global sentence-level information is much more useful than local word-level information. Integration of these two levels' information (HCAN) can further boost the performance. What's more, we find that our best model HCAN performs best on all parts of speech, except for *adverbs*. However, there are only 346 examples about adverbs which account for 5% of the four test datasets, thus 1% drop on adverbs means only 4 examples are wrongly classified which will make little influence on the overall score.

---

[5] The released code can be found in `https://github.com/luofuli/word-sense-disambiguation`

	Test Datasets				Concatenation of Test Datasets				
System	SE2	SE3	SE13	SE15	Noun	Verb	Adj	Adv	**All**
Full Model	**72.8**	70.3	**68.5**	**72.8**	72.7	**58.2**	**77.4**	**84.1**	**71.1**
No Attention	70.2	68.1	67.6	68.9	71.2	54.3	74.3	82.1	68.8
W/O Word-level Attention	71.5	**70.4**	68.2	71.7	72.7	57.1	75.3	81.8	70.5
W/O Sentence-level Attention	70.0	69.5	66.8	70.3	71.3	55.2	74.4	83.0	69.1
W/O Context2Gloss Attention	70.7	69.7	68.2	71.0	72.2	55.2	75.5	82.7	69.9
W/O Gloss2Context Attention	72.3	70.3	68.2	71.9	**73.2**	56.4	75.4	83.8	70.7

Table 4: Ablation study of the proposed model HCAN.

.

4.4 Ablation Study

In this part, we further discuss the impacts of the components of our hierarchical model HCAN. In order to ablate the co-attention mechanism, we replace the co-attention function $CoAt$ in Equation 1 with a function Avg which simply calculates the average of input representation vectors. Specifically, in function Avg, the outputs $c = \sum_j C_{:j}$ and $g = \sum_j G_{:j}$.

We re-train HCAN by ablating certain components:

- *No Attention*: We totally replace the co-attention function $CoAt$ with Avg in both word-level and sentence-level co-attention layers. This is the baseline for comparison.
- *W/O Word-level Attention*: We replace the word-level co-attention function $CoAt$ with Avg. Note that this ablation model is different from CAN^s, for that the word-level representation vector $\hat{c}^w$ is used to calculate the final score in this ablation model.
- *W/O Sentence-level Attention*: We replace the sentence-level co-attention function $CoAt$ with Avg. Note that this ablation model is not same as CAN^w, for the sentence-level representation vector $\hat{c}^s$ is also used to calculate the final score in this ablation model.
- *W/O Context2Gloss Attention*: We remove the attention of generating the context vector, which means all elements in A^g are set to 1.
- *W/O Gloss2Context Attention*: We remove the attention of generating the gloss vector, which means all elements in A^c are set to 1.

Table 4 indicates the effectiveness of different components in the proposed model HCAN. It shows that without any attention mechanism, the overall score declines 2.3%.

Ablated versions without word, sentence level co-attention decline 0.6% and 2.0%, respectively. It reveals that sentence-level co-attention mechanism seems much more important to HCAN, which is consistent with the scores of CAN^s and CAN^w. However, we find that the results of ablated versions without word-level and sentence-level co-attention are worse than CAN^s and CAN^w. We hypothesize that it is because that the context and gloss vector generated from the layer (or level) which doesn't use attention mechanism may bring some noise to the final scores.

Without the context-to-gloss attention, the score declines 1.2% on concatenation of the four test datasets. Conversely, without the gloss-to-context attention, the score declines 0.4%. It is probably due to that the context-to-gloss attention which generates the context-aware gloss vector is more *direct* to find out the correct word sense.

In conclusion, the results in Table 4 show that all components in the proposed hierarchical co-attention model HCAN can contribute to boosting the performance of WSD.

5 Conclusions

In this paper, we investigate the problem of incorporating gloss knowledge into neural network for Word Sense Disambiguation. We find that the gloss can highlight the important words in the context, and later contribute to the representation of the context. Meanwhile, context can also help to focus on the words in gloss of the right word sense. Therefore, we propose a co-attention mechanism to model the gloss-to-context and context-to-gloss attention. Furthermore, in order to capture not only local word-level features but also global sentence-level features, we extend the co-attention model into a hierarchical architecture. The experimental results show that our proposed models achieve the state-of-the-art results on several standard English all-words WSD datasets.

Acknowledgments

We would like to thank Lei Sha, Jianmin Zhang and Junbing Liu for insightful comments and suggestions. This paper is supported by NSFC project 61772040 and 61751201. The contact authors are Baobao Chang and Zhifang Sui.

References

Eneko Agirre, Oier Lpez De Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *CoRR*, abs/1608.00318.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. An enhanced lesk word sense disambiguation algorithm through a distributional semantic model. In *Roceedings of COLING 2014, the International Conference on Computational Linguistics: Technical Papers*.

T. Chen, R. Xu, Y. He, and X. Wang. 2015. Improving distributed representation of word sense via wordnet gloss composition and context clustering. *Atmospheric Measurement Techniques*, 4(3):5211–5251.

Philip Edmonds and Scott Cotton. 2001. Senseval-2: overview. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5. Association for Computational Linguistics.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, Jun Zhao, Yanchao Hao, Yuanzhe Zhang, and Kang Liu. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Meeting of the Association for Computational Linguistics*, pages 221–231.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *The Meeting of the Association for Computational Linguistics*.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. *arXiv preprint arXiv:1606.03568*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries:how to tell a pine cone from an ice cream cone. In *Acm Special Interest Group for Design of Communication*, pages 24–26.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 289–297.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018. Incorporating glosses into neural word sense disambiguation. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Andrea Moro and Roberto Navigli. 2015. Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *International Workshop on Semantic Evaluation*, pages 288–297.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Roberto Navigli. 2009. Word sense disambiguation:a survey. *Acm Computing Surveys*, 41(2):1–69.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. Semeval-2013 task 12: Multilingual word sense disambiguation. In *The International Workshop on Semantic Evaluation*.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250.

Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. Semeval-2007 task 17: English lexical sample, srl and all words. In *International Workshop on Semantic Evaluations*, pages 87–92.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017a. Neural sequence learning models for word sense disambiguation. In *Conference on Empirical Methods in Natural Language Processing*.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017b. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proc. of EACL*, pages 99–110.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *arXiv preprint arXiv:1507.01127*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2016. Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling. *CoRR*, abs/1605.05110.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Meeting of the Association for Computational Linguistics*, pages 1436–1446.

Qi Zhang, Xiaoyu Liu, and Jinlan Fu. 2018. Neural networks incorporating dictionaries for chinese word segmentation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Zhong Zhi and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL 2010, Proceedings of the Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, System Demonstrations*, pages 78–83.

Weeding out Conventionalized Metaphors: A Corpus of Novel Metaphor Annotations

Erik-Lân Do Dinh, Hannah Wieland, and Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de

Abstract

We encounter metaphors every day, but only a few jump out on us and make us stumble. However, little effort has been devoted to investigating more novel metaphors in comparison to general metaphor detection efforts. We attribute this gap primarily to the lack of larger datasets that distinguish between conventionalized, i.e., very common, and novel metaphors. The goal of this paper is to alleviate this situation by introducing a crowdsourced novel metaphor annotation layer for an existing metaphor corpus. Further, we analyze our corpus and investigate correlations between novelty and features that are typically used in metaphor detection, such as concreteness ratings and more semantic features like the *Potential for Metaphoricity*. Finally, we present a baseline approach to assess novelty in metaphors based on our annotations.

1 Introduction

Metaphors have received considerable interest in NLP in recent years (see Shutova (2015)). Research questions range from direct detection of metaphors in text (*linguistic metaphors*) to finding mappings between conceptual source and target domains (*conceptual metaphors*).

However, an important aspect of metaphors—*novelty*—is often overlooked, or intentionally disregarded. Consider the metaphors (bold) in the following examples:

(1) We all live on **tight budgets**, but we still need to have some fun.

(2) They were beginning to attract a **penumbra** of gallery-goers, as though they were offering a guided tour.

The metaphor *tight budgets* in (1) is an often used collocation and therefore highly conventionalized. While the basic senses of *tight*—e.g., being physically close together or firmly attached—conflict with the more abstract *budget*, the metaphoric use as meaning *limited* can be readily understood. In contrast, the use of *penumbra* in (2) is more creative and novel. Its literal meaning is "an area covered by the outer part of a shadow."[1] Its metaphoric meaning is seldom encountered: Shadows follow objects that cast them, and especially penumbras can be perceived as having fuzzy outlines; attributes which are picked up by the metaphorical sense of a rather unspecified group of people following someone in differing vicinity.

Common linguistic metaphor definitions used in NLP (Steen et al., 2010; Tsvetkov et al., 2014) do not differentiate between *conventionalized* and *novel* metaphors. Some even allow for auxiliary verbs and prepositions to be annotated as metaphors when they are not used in their original sense (e.g., the non-spatially used *on* in "She wrote a study **on** metaphors"). While such cases can be filtered out rather easily from any given corpus—e.g., by using POS tag and lemma filters—many conventionalized metaphors persist. Existing work avoids this problem partially by only annotating certain grammatical constructions, such as adjective–noun or verb–object relations (Shutova et al., 2016; Rei et al., 2017). However, these too usually do not distinguish between conventionalized and novel metaphors.

Following Shutova (2015), we deem the distinction between conventionalized and novel metaphors important, because the meaning of conventionalized metaphors can usually be found in dictionaries or other resources like WordNet—novel metaphors on the other hand pose a more difficult challenge. But the lack of resources incorporating this distinction leads to few researchers

---

[1] https://www.macmillandictionary.com/dictionary/american/penumbra

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1412–1424
Brussels, Belgium, October 31 - November 4, 2018.

investigating novel metaphors, or the related measure of metaphoricity. They use small datasets that have been manually annotated by experts (Del Tredici and Bel, 2016), or focus crowdsourcing studies on a small number of instances (Dunn, 2014). It is only recently that any work has introduced larger-scale novel metaphor annotations (Parde and Nielsen, 2018). In contrast to our approach, they collect annotations on a relation level (see also Section 2).

Annotating metaphors is not an easy task, due to the inherent ambiguity and subjectivity. Therefore, we investigate approaches for annotating novelty in metaphors, before closing the resource gap for token-based annotations by creating a layer of novelty scores.

Our contributions are the following:

(1) We augment an existing metaphor corpus by assessing metaphor novelty on a token level using crowdsourcing, enabling larger research on novel metaphors,

(2) we analyze our corpus for correlation with features used for general metaphor detection or metaphoricity prediction, and

(3) we show that a baseline approach based on features usually used for (binary) metaphor detection can be useful for distinguishing novel and conventionalized metaphors.

In Section 2, we first discuss annotation guidelines that have been used for existing datasets, as well as prior work on novel metaphor detection and crowdsourcing in NLP. This discussion is followed by Section 3, where we detail the base corpus and the crowdsourced creation of our annotation layer. In Section 4, we describe our baseline for detecting novel metaphors and its results. We conclude in Section 5 with a summary of our contributions, and an outlook of what our newly introduced layer of annotations can enable.

2 Related Work

Annotating metaphors is a difficult task, because there is no single definition that can be adhered to. Instead, different researchers formulate their own versions, which can vary quite substantially.

The Pragglejaz Group (2007) created influential metaphor annotation guidelines, the *Metaphor Identification Procedure* (MIP). After reading a text, an annotator assesses each token as being used metaphorically or not; i.e., if the token has a more *basic* meaning that can be understood in comparison with the one it expresses in its current context. More basic is described as being:

- More concrete; what they evoke is easier to imagine, see, hear, feel, smell, and taste.
- Related to bodily action.
- More precise (as opposed to vague).
- Historically older.

They note that this basic meaning does not necessarily have to be the most frequent one. An extended version, MIPVU, was used to annotate parts of the British National Corpus (BNC Consortium, 2007, BNC), resulting in the VU Amsterdam Metaphor Corpus (Steen et al., 2010, VUAMC). Shutova and Teufel (2010) adapt the MIP as a prerequisite to annotating source and target domains of metaphor—the metaphoric mapping—in parts of the BNC.

Others use rather relaxed guidelines. Tsvetkov et al. (2014) rely on intuitive definitions by their annotators, not specifying metaphor more closely. They ask their annotators to mark words that “are used non-literally in the following sentences.” Jang et al. (2015) provide a Wikipedia definition of metaphor to users in a crowdsourcing study. Subsequently, the users are tasked with annotating forum posts by deciding “whether the highlighted word is used metaphorically or literally.”

A common trait of the listed works is that they do not distinguish between nuances of metaphoricity. Instead, they impose a binary scheme (metaphoric/literal) on the rather diffuse language phenomenon of metaphor. In contrast, Dunn (2014) introduces a scalar measurement of *metaphoricity* on a sentence level. He created a corpus of 60 genre-diverse sentences with varying levels of metaphoricity, which are rated in three crowdsourcing experiments: a binary selection, a ternary selection, and an ordering approach. The mean value for each sentence is then used as the metaphoricity value. Dunn (2014) uses this data to derive a computational measure of metaphoricity, which he then employs in an unsupervised system to label metaphoric sentences in the VUAMC. The evaluation, however, is only done on the binary labels provided therein.

In a similar direction, Del Tredici and Bel (2016) present their concept of *Potential of*

genre	tokens	metaphors	content tokens	content metaphors	novel metaphors
acprose	75,272	9,170	42,544	5,505	102
convrsn	57,249	3,841	22,019	1,774	25
fiction	54,115	5,349	26,935	3,174	94
news	51,672	7,580	29,346	4,727	132
total	238,308	25,940	120,844	15,180	353

Table 1: VUAMC corpus statistics. Content tokens include adjectives, adverbs, verbs (without *have*, *be*, *do*), and nouns. For the purpose of this table, metaphors with a novelty score higher than $T = 0.5$ are considered novel (the possible range is [–1,1]).

Metaphoricity (POM) of verbs. The POM describes the inherent potential of a verb to take on a metaphoric meaning, derived from its distributional behavior. They infer that low-POM verbs are only able to have low degrees of metaphoricity, thus can only evoke conventionalized metaphors. Therefore, they propose to exclude such low-POM verbs from novel metaphor detection systems.

Haagsma and Bjerva (2016) use violations of selectional preferences (Wilks, 1978) to find novel metaphors. Selectional preferences describe which semantic classes a verb prefers as its direct object. Since they are often mined from large corpora and based on frequency, they argue that this feature is more suited for novel metaphor detection than for general detection of (also conventionalized) metaphors. They evaluate their approach on the VUAMC; however, they acknowledge that their usage of this corpus is not optimal because it contains many conventionalized metaphors.

Parde and Nielsen (2018) create a corpus of novel metaphor annotations. For their crowdsourced annotation, they use a scale with four options. Unlike our approach, they annotate relations between words as novel or conventionalized. On the one hand, this is a sensible approach because generally the context of a word determines its metaphoricity (and indeed, its novelty in case of metaphoric use). On the other hand, such annotations lack the flexibility and ease of use of token-based annotations for which the context is not defined a priori.

We tackle the lack of data by annotating an existing corpus using crowdsourcing; i.e., by splitting up the task in many small chunks which different, non-expert annotators are instructed to complete. Crowdsourcing has been used for a variety of annotation tasks in NLP, often using different study designs. Snow et al. (2008) obtain good annotation results for five tasks with different setups: two tasks ask for numerical values (affect recognition, word similarity), two other tasks require a binary decision (textual entailment recognition, event ordering), and a final task provides three options for the annotators (word sense disambiguation). Sukhareva et al. (2016) utilize crowdsourcing to annotate semantic frames. They design their task as a decision tree, with annotators moving down the tree when annotating. Mohammad et al. (2016) use crowdsourcing for metaphor and emotion annotation in order to investigate their correlation. They employ an ordering approach to annotation, and only consider verbs that already contain a metaphoric sense in WordNet. Kiritchenko and Mohammad (2016) obtain annotations for sentiment associations via crowdsourcing. They use *Best–Worst Scaling* (Louviere and Woodworth, 1990), an annotation approach which creates scores from ranking annotations.

3 Corpus

To obtain a corpus of novel metaphor annotations, we employ an existing metaphor corpus. This can potentially reduce ambiguity for annotators and allows us to focus on the creation of novelty scores. We use the VU Amsterdam Metaphor Corpus (Steen et al., 2010) as the base corpus for our novelty annotations due to its comparably large size and genre diversity. It is comprised of over 200,000 tokens from four genres: *academic*, *fiction*, and *news* texts, and *conversation* transcripts (Table 1). Further, reusing existing annotations enables us to only query annotators for novelty of already annotated metaphors, instead of having them analyze every token.

Using crowdsourcing (Amazon Mechanical Turk), we first conduct a pilot study to choose among four different annotations methods. We then employ the best method to collect annotations and create an additional *novelty score* between 1 (novel) and −1 (conventionalized) for each token labeled as metaphor in the VUAMC. Note that non-content words like prepositions and auxiliary verbs (*have*, *be*, *do*) are filtered out beforehand. Our annotations/scores can be integrated into the original VUAMC resource. We make the annotations and scripts to embed them into the original corpus publicly available.[2]

3.1 Pilot Study

Similar to Dunn (2014), we consider multiple annotation approaches. We compare them in a pilot study using 210 metaphor tokens with their sentence context, randomly chosen from the *fiction* subcorpus. For the sake of a meaningful evaluation, we ensured that 25% of these metaphorically used tokens were novel metaphors. Before choosing one approach for the entire corpus, we compare the following four annotation approaches:

- **binary annotation**: crowd workers decide if a given metaphoric token is used in a novel or conventionalized way;
- **scale annotation**: crowd workers decide on the novelty of a given metaphoric token on a four-point scale, from *very novel* to *very conventionalized*;
- **scale annotation (no metaphor)**: crowd workers decide on the "unusualness" of a given token in its context on a four-point scale, from *I've never heard it before* to *I'm using it everyday*; without giving information that the tokens represent metaphors;
- **best–worst scaling**: crowd workers pick the most novel and the most conventionalized metaphor from four samples.

Conceptually, the *binary annotation* should put the least cognitive load on the annotator, resulting in fast annotation times and efficient completion of the task. However, this method does not allow for nuances in annotation. Dunn (2014) counters this problem by assigning as the score the percentage of "metaphoric" labels for an instance. But this solution is only feasible for smaller datasets, as it requires many annotations per instance to yield nuanced scores.

	IAA	F_1	avg assignment completion time
binary	0.38	0.75	1:39 min
scale	0.32	0.75	1:04 min
scale w/o met.	0.16	0.67	2:10 min
BWS	—	0.84	1:58 min

Table 2: Comparison of the approaches investigated in the pilot study. Shown are inter-annotator agreement (Krippendorff's α, after mapping the scale annotations to binary labels), evaluation against our silver standard (F_1), and average completion time for an assignment (in case of binary, scale, and scale without metaphor this amounts to one decision, for BWS to two). Note that there is no IAA for the BWS approach because no two tuples are the same.

The next two approaches, *scale annotation* and *scale annotation (no metaphor)*, try to mitigate this problem by introducing four options to choose from. We choose a scale of four instead of three options to force the annotators to indicate a preference, rather than allowing a "neutral" answer. The difference between both scale approaches is in the guideline descriptions; for the second scale approach (no metaphor), we remove any mention of metaphor, and instead use paraphrases (e.g., instead of "novel metaphor" we use "I have not seen it before"). Our motivation behind this rephrasing is that we want to avoid confusion especially with regards to very conventionalized metaphors. By strictly asking for novelty of the expression/usage, we potentially simplify the task for the annotator.

The last approach uses *best–worst scaling* (BWS, see also Section 2). It has the advantage of not explicitly asking the annotator for a decision on a singular token/metaphor. Instead, they are asked to compare (four) different metaphors and select the most novel and the most conventionalized. A disadvantage is the higher workload for the annotator, since they have to read four sentences for one assignment.

In Table 2, we give a short overview of inter-annotator agreement (Krippendorff's α), the average completion time of an assignment, and a comparison with our semi-automatically created silver standard (F_1). The latter is built by using the majority vote of all four methods; ties are resolved manually by looking into the individual an-

---

[2] https://github.com/UKPLab/emnlp2018-novel-metaphors

notations. To obtain binary labels from the two scale approaches for majority voting, we map the two "more novel" options to novel, and the other two options to conventionalized. For the BWS approach, we first average the number of novel metaphors from the other three methods. From a sorted, decreasing list of BWS scores we then mark this many metaphors as novel. We compare against our own annotations as a sanity check.

Regarding completion time, we observe that the scale method is the fastest by a wide margin. This discrepancy is somewhat surprising, as the scale method introduces two more options to consider compared to the binary method. Apparently, these additional, intermediate options make it easier for annotators to come to a decision, especially for edge cases. On the other hand, scale without metaphors takes twice as long as the scale method. The latter only differs from the former in its inclusion of *metaphor* in the task description and the labels, which we thus interpret as creating a setting for the annotators where they may expect to have some kind of intuition about the task, and thus are more confident (and faster) in completing. Given the long completion time in conjunction with the lower F_1 score and the very low IAA, it seems clear that omitting the metaphor information and using a more colloquial task description make the scale annotation (no metaphor) by far the most difficult for workers to complete, resulting in the least usable annotations. While best–worst scaling is time consuming as well, it yields the best results with regards to the silver standard ($F_1 = 0.84$) by a large margin. We thus choose BWS for annotating the full corpus. Further details on the pilot study can be found in Wieland (2018).

3.2 Corpus Creation

We design our guidelines (see Appendix A) to be simple, but include redundancy in the description to address frequent misunderstandings and ambiguities. We also explicitly mention idioms and unrecognizable metaphors as cases of conventionalized metaphors, because these were sources of confusion in our pilot study. Unrecognizable here means that a word is used in an established, commonly understood—but not the most concrete—sense; e.g., *hard* in "She fought **hard**." Additionally, the annotators are provided with example metaphors of differing novelty.

After filtering out prepositions and auxiliary verbs (*have*, *be*, *do*) using the POS tags supplied by the VUAMC, we collect annotations covering 15,180 metaphors in total (Table 1). We only include workers located in the US. For creation of the best–worst scaling tuples, and for aggregation of the annotations, we use the scripts provided by Kiritchenko and Mohammad (2016).[3] We use a best-worst scaling factor of 1.5 and four items per tuple. Thus, each metaphor appears in six different best-worst scaling comparisons. This results in 22,770 best–worst scaling items to be annotated.

3.3 Analysis

Overall statistics about our created annotations are shown in Table 1 (along with the already existing annotations). For the sake of this overview, we introduce a threshold $T = 0.5$, and treat metaphors with a BWS score equal to or above this threshold as novel, metaphors with a BWS score below the threshold as conventionalized. For example, in this way the metaphor "[...] the artistic temperament which kept her **tight-coiled** as a spring [...]" (0.514) is treated as novel, while "To **quench** [thirst] is more than to refresh [...]" (0.424) is treated as conventionalized. However, since we provide the scores, this threshold can be adjusted to suit a given application. Note that, while novel metaphors are arguably much more scarce than conventionalized ones, BWS creates scores which are approximately normally distributed, supporting our threshold choice.

Before we conduct a more in-depth analysis of the annotated metaphors, we show some examples. In Table 3, we list four novel and four conventionalized metaphors (as annotated). A good example for a novel metaphor is the description of "words [...] as a **coat-hanger**" in Table 3 (3). This usage cannot be found in dictionaries, and clearly constitutes creative language use. In contrast, the meaning *to experience something bad* of *to go through [a situation]* (ibid., (7)), or the sense *to do/conduct* of *to get [something] done* (ibid., (5)), are strongly conventionalized, as indicated by their inclusion in dictionaries.[4]

We also group the tokens by lemma to examine if certain words are more likely to be used in a novel metaphoric way. Inspecting the mean

---

[3] http://saifmohammad.com/WebPages/BestWorst.html

[4] e.g., https://www.macmillandictionary.com/dictionary/american/go-through#go-through__7, https://www.macmillandictionary.com/dictionary/american/get#get__60

no	score	metaphor in context
(1)	0.765	Ron Todd [...] warned that party leaders could not expect everybody to '**goose-step**' in the same direction once the policy had been carried.
(2)	0.750	Westerns have a **gladiatorial**, timeless quality.
(3)	0.735	Allan Ahlberg says: 'In the past, a lot of children's books seemed to be the work of talented illustrators whose pictures looked brilliant framed in a gallery, but when you tried to read the book, there was nothing there, because the words started as a **coat-hanger** to hang pictures on.'
(4)	0.727	thus one can and must say ... that each fight is the singularisation of all the circumstances of the social whole in movement and that by this singularisation, it **incarnates** the enveloping totalization which the historical process is.
(5)	−0.765	If the complaint is proved, a nuisance order is made requiring the defendant to **get** the necessary work done.
(6)	−0.765	Apart from some dark patches on the wall that he hadn't noticed **before**, there was nothing to see.
(7)	−0.774	In relation to the sentence stem 'A girl and her mother ...', girls often produce responses like 'often **go** through a bad patch for a year but once they learn to understand each other, become the best of friends' or 'can help each other with their problems'.
(8)	−0.871	The analyst is then forced on the defensive, explaining why **new** features can not be included because they are technically difficult or prohibitively expensive.

Table 3: Example uses of very novel and very conventionalized annotated metaphorical tokens (bold) in sentence context, according to our aggregated annotations. Scores near 1 denote strong novelty, scores near -1 indicate very conventionalized metaphors.

scores, we see the large conventionalization of words such as *get* (−0.37), *see* (−0.35), or *new* (−0.35). Examples for words used in a mostly novel way are *envelop* (0.53), *incarnate* (0.50), and *thrust* (0.46). In the following sections, we examine the correlations with frequency and POM, which give further weight to the examples.

Further, we investigate how novelty is distributed over the different subcorpora (also see Table 1). Somewhat expectedly, the metaphors used in the academic subcorpus are mostly conventionalized (1.9% novel metaphors). A larger percentage of novel metaphors can be found in the news (2.8%) and fiction (3.0%) subcorpora. The conversation subcorpus shows the least amount of novel metaphors (1.3%), which we interpret as natural—more novel metaphors require some amount of creativity to create, which is arguably easier in writing. We also show the distribution of novel metaphors across POS tags (Table 4). It is striking that the verbs are least represented among the novel metaphors, especially compared to their overall metaphoric occurrence. In contrast, adjectives/adverbs and nouns are more likely to be used in a novel way.

POS	tokens	metaphors	novel metaphors
nouns	47,171	5,513	145
verbs	27,831	6,513	88
adj/adv	45,842	3,154	120

Table 4: Metaphor annotations grouped by POS tags. For this overview, we treat metaphors with a score above $T = 0.5$ as novel.

Correlation with Token Frequency We compare how the token frequency is related to novelty. Intuitively, words that are seldom used should show a higher (average) novelty. In turn, we expect often used words to exhibit a wider range of (already conventionalized) senses, thus having lower (average) novelty. We obtain token frequencies from a Wikipedia dump, and correlate them with the mean novelty scores from our annotations (disregarding part-of-speech tags). The relation in terms of Spearman's rank correlation is $\rho = -0.60$, which indicates a moderate anti-correlation. This can also be observed in Figure 1: while high frequency seems to hint at conventionalized metaphors, low frequency is not necessarily

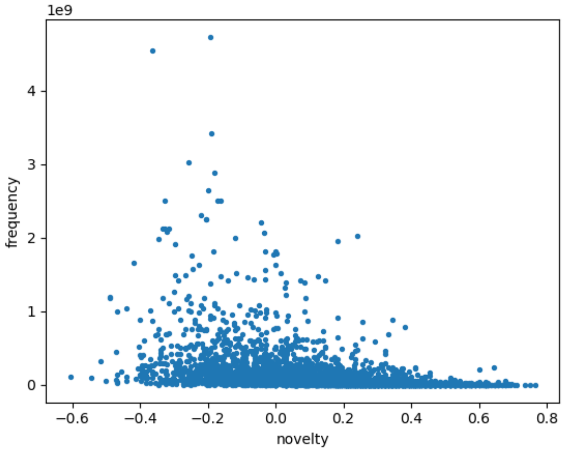

Figure 1: **Frequency.** Relation between average novelty score of metaphoric tokens and their frequency in Wikipedia (correlation of $\rho = -0.60$). Since we use the automatically created POS tags from the VUAMC to filter out non-content tokens, this can include erroneously tagged tokens. For an improved overview, we manually filtered out the five most frequent, incorrectly tagged tokens from this plot (*to*, *as*, *on*, *that*, and *this*).

an indication of novelty of metaphoric use.

Two prominent exceptions are the tokens *national* and *united* (Figure 1, upper right). While they show comparatively high novelty (both labeled as metaphorical only once in the VUAMC), they appear surprisingly often in Wikipedia. Another artifact of using Wikipedia as the background corpus can be seen on the left: *try* is only annotated as metaphoric in infinitive-compounds that are decidedly conventionalized (e.g., "trying to look"), yet it appears comparatively seldom in Wikipedia. However, we chose to use Wikipedia instead of the BNC in order to have an out-of-domain comparison with a more contemporary, larger background corpus.

Correlation with Concreteness The use of concreteness as a feature in automatic metaphor detection grounds in the Conceptual Metaphor Theory (Lakoff and Johnson, 1980). In short, metaphors are modeled as cognitive mappings between an often concrete source domain, and a usually more abstract target domain. For example, in "He **shot down** my arguments," the more concrete domain of ARMED CONFLICT (*shot*) is mapped to the rather abstract domain DISCUSSION (*argument*).

To analyze the relation between novelty and concreteness, we first extend the concreteness list by Brysbaert et al. (2014) using a technique similar to Mohler et al. (2014). For a given token *t*, we extract 20 approximate nearest neighbors *nn(t)* from Google News Embeddings (Mikolov et al., 2013) using Annoy.[5] The concreteness value for *t* is then computed by averaging its neighbors' concreteness values from the concreteness list.

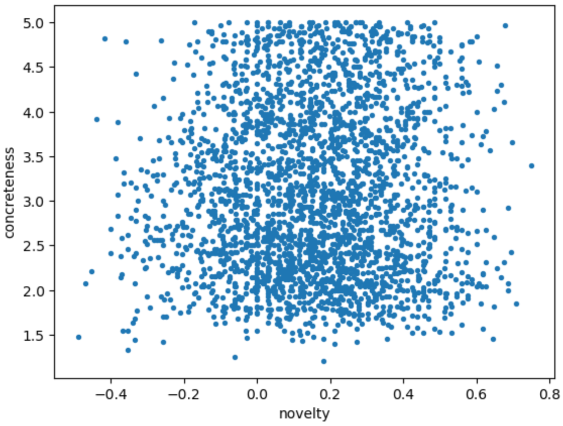

Figure 2: **Concreteness.** Relation between novelty score of metaphoric tokens and their concreteness, showing no discernible correlation ($\rho = 0.03$). A similar picture emerges if we only consider the manually annotated tokens included in the original concreteness list by Brysbaert et al. (2014).

Subsequently, we calculate the correlation between the average novelty and the concreteness of the lemmas. Both Pearson correlation ($r = 0.04$) and Spearman's rank correlation ($\rho = 0.03$) are close to zero and indicate no correlation (Figure 2). Thus, while concreteness has been shown to work well as a feature to distinguish between literal and non-literal language in general (Beigman Klebanov et al., 2014; Tsvetkov et al., 2014), it does not seem useful for discerning between novel and conventionalized metaphoric usage in particular. For example, the rather abstract *now* (1.48) and the very concrete *people* (4.82) are assigned similarly low novelty scores. On the other hand, *justice* has a quite high novelty score (0.65), while also being as abstract as *now*. One reason for this non-correlation between concreteness and novelty might be that the automatic induction of concreteness ratings introduces too much noise. However, an experiment where we only use tokens occurring in the manually composed list (Brysbaert et al., 2014) shows similarly low correlation. This could be influenced by artifacts in the concrete-

---

[5] https://github.com/spotify/annoy

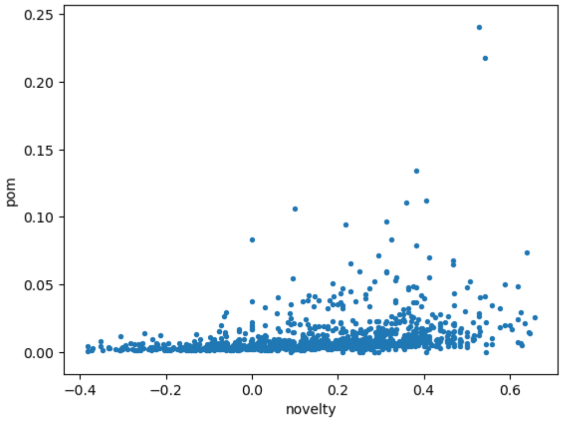

Figure 3: **POM.** Relation between average novelty score of verb lemmas and POM (correlation of $\rho = 0.52$).

ness list: as laid out by Beigman Klebanov et al. (2015), it exhibits some problems. For example, it shows high variance in the annotated concreteness scores for various non-concrete adjectives. But this does not explain the extent of the non-correlation. We thus believe that our results indeed indicate no relation between concreteness and metaphor novelty. And indeed, if a difference of concreteness between components of an expression hints at a metaphor, as is proposed by the conceptual metaphor theory, then it is plausible that it does not hint at novelty of the metaphoric expression at the same time. Consequently, we investigate a feature with more semantic capacity in the next subsection.

Correlation with POM The *Potential for Metaphoricity* (POM) of verbs was introduced by Del Tredici and Bel (2016) (Section 2). It denotes the a priori chance of a verb to occur in highly metaphoric contexts; in essence, it measures the contextual flexibility of a verb. Generally, very novel metaphors also display a high metaphoricity. Therefore, we expect that low-POM verbs (i.e., verbs that occur similarly often in many different contexts) exhibit a low novelty score on average and low variance, while high-POM verbs should show a higher average novelty score. The POM can be regarded as a variant of selectional preference strength, which measures how strongly a verb constrains its direct object in terms of semantic classes. As such, we forgo an analysis of selectional preference violations in favor of examining the POM. Hovy et al. (2013) generalize the notion of selectional preferences to other forms of grammatical relations. However, instead of generating scores, they use dependency trees in an SVM with tree kernels. The POM could be similarly generalized to all POS tags, e.g., by including head and dependent tokens as context.

We create the POM for all annotated verbs using the same procedure as Del Tredici and Bel (2016). First, we extract the context (i.e., subject and object) for each occurrence of a verb from a large, parsed corpus (Wikipedia). To compute context vectors, the word embeddings (Levy and Goldberg, 2014) of subject and object are averaged (if only one of the two is available, the embedding for this token serves as the context). For each verb, the context vectors are then clustered using Birch clustering (Zhang et al., 1996). Finally, the standard deviation between the sizes of the context clusters denotes the POM of the verb.

As with our previous experiments, we compute Spearman's rank correlation between the mean novelty scores of the verb lemmas and the corresponding POMs. We arrive at Spearman's $\rho = 0.52$, which indicates moderate correlation (Figure 3). Verbs like *pique* (POM: 0.218) and *slit* (0.134) are more often used in a novel metaphoric way, while low-POM verbs *show* and *see* (both: 0.01) are only used as conventional metaphors. Even though the correlation is only moderate, it supports the WordNet-based evaluation by Del Tredici and Bel (2016), who found that high-POM verbs generally induced novel metaphoric sentences. Note that their POM values are higher because they optimize the clustering parameters, which we leave at the default setting.

4 Baseline

While the main focus of this work is the analysis of our new annotation layer, we also create a simple baseline regression system for predicting the novelty scores. We run the system using two configurations: first with only word embeddings as input, then augmented with frequency and POM scores.

4.1 System

We implement a single-layer BiLSTM for predicting the novelty score. As input, we use a padded 11-token window (five before, five after the token) of dependency-based word embeddings by Levy and Goldberg (2014) that we also employed for the POM computation. The BiLSTM layer has 50 dimensions and ReLu activation. Training is done in maximally 20 epochs, but can halt earlier due to early stopping on the development set. The data is split into training (50%), development (25%), and test set (25%). We only conduct experiments on verbs, so that we can compare the performance when including additional features.

Following our analysis of typically used features for metaphor detection in relation to novelty, we incorporate the relative frequency of a token into the model, to investigate if the substantial correlation observed in Section 3.3 has an impact on our regression experiments. Further, we include the POM, as it also showed a moderate correlation with novelty. Both additional features are concatenated with the respective word embeddings and the resulting vectors are fed to the BiLSTM.

4.2 Results

We evaluate our results using mean absolute error (MAE), and average it over 10 runs with different random seeds. Recall that the possible values lie between −1 and 1, leading to a possible MAE between 0 (best) and 2 (worst). The baseline results over different runs are stable, we show the regression plot for one configuration in Figure 4.

The mean absolute error for the configuration using only the embeddings is MAE = 0.166. In contrast, we obtain MAE = 0.163 for the same configuration when we add the frequency and the POM features. Thus, while small, the latter model shows improvements over the word embedding baseline. As can be seen in the plot, there is still much room for improvement (e.g., we did not conduct extensive hyper-parameter optimization). The network makes errors both in underestimating ("they can be seen as **riddled** with holes") and overestimating novelty ("veins branching off it to **form** a network"). These errors are in many cases independent from POM and frequency features. Thus, while a better optimization (e.g., in the clustering step when creating the POM) might reduce estimation errors, we also need to consider further features for metaphor novelty estimation.

5 Conclusion

We presented a new layer of novelty scores for the VU Amsterdam Metaphor Corpus, created using crowdsourcing. To this end, we conducted a pilot study to choose an appropriate method for metaphor novelty annotation and found that best–worst scaling outperformed binary and scale methods. Our corpus analysis of typically used features for metaphor detection showed no correlation of novelty with concreteness. However, we found substantial correlation with frequency of tokens in a background corpus and with potential for metaphoricity, a context-based a priori metaphoricity measure. Further, we created a baseline to distinguish novel from conventionalized metaphors. For our approach, the latter two features could improve results only slightly, indicating a need for more sophisticated features.

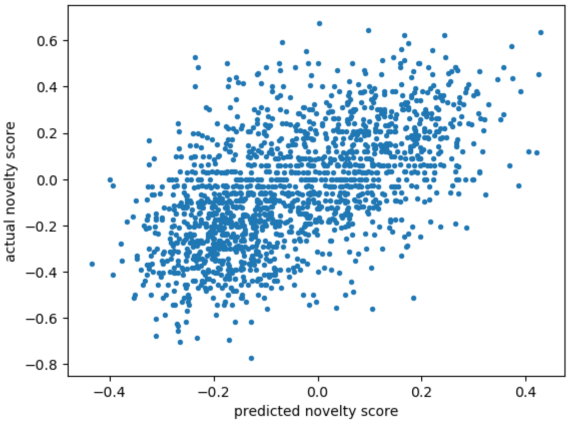

Figure 4: Predicted and actual novelty score of metaphoric tokens in the VUAMC for a baseline configuration (word embeddings only, without adding frequency and POM information).

Previous work in automatic metaphor processing has largely focused on general detection of linguistic and conceptual metaphors, mostly disregarding the subject of novelty. Our corpus enables new evaluation and training possibilities for detecting the latter. In future work, we want to develop more sophisticated methods to detect and distinguish novel metaphors. For example, we want to extend the notion of POM to nouns and adjectives, and investigate other a priori measures for metaphor novelty. Further, we want to jointly detect metaphors and score their novelty. Another interesting direction is to investigate the correlation between perceived novelty and the existence of dictionary definitions for metaphoric senses of a token or expression.

Acknowledgements

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 01UG1816B (CEDIFOR).

References

Beata Beigman Klebanov, Chee Wee Leong, and Michael Flor. 2015. Supervised Word-Level Metaphor Detection: Experiments with Concreteness and Reweighting of Examples. In *Proceedings of the Third Workshop on Metaphor in NLP*, pages 11–20, Denver, CO, USA. Association for Computational Linguistics.

Beata Beigman Klebanov, Chee Wee Leong, Michael Heilman, and Michael Flor. 2014. Different Texts, Same Metaphors: Unigrams and Beyond. In *Proceedings of the Second Workshop on Metaphor in NLP*, pages 11–17, Baltimore, MD, USA. Association for Computational Linguistics.

BNC Consortium. 2007. The British National Corpus, version 3 (BNC XML Edition).

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness Ratings for 40 Thousand Generally Known English Word Lemmas. *Behavior Research Methods*, 46(3):904–11.

Marco Del Tredici and Núria Bel. 2016. Assessing the Potential of Metaphoricity of Verbs Using Corpus Data. In *Proceedings of LREC 2016*, pages 4573–4577, Portorož, Slovenia. European Language Resources Association.

Jonathan Dunn. 2014. Measuring Metaphoricity. In *Proceedings of ACL 2014*, pages 745–751, Baltimore, MD, USA. Association for Computational Linguistics.

Hessel Haagsma and Johannes Bjerva. 2016. Detecting Novel Metaphor Using Selectional Preference Information. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 10–17, San Diego, CA, USA. Association for Computational Linguistics.

Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huiying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying Metaphorical Word Use with Tree Kernels. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 52–57, Atlanta, GA, USA. Association for Computational Linguistics.

Hyeju Jang, Seunghwan Moon, Yohan Jo, and Carolyn Penstein Rosé. 2015. Metaphor Detection in Discourse. In *Proceedings of SIGDIAL 2015*, pages 384–392, Prague, Czech Republic. Association for Computational Linguistics.

Svetlana Kiritchenko and Saif M. Mohammad. 2016. Capturing Reliable Fine-Grained Sentiment Associations by Crowdsourcing and Best–Worst Scaling. In *Proceeding of NAACL-HLT 2016*, pages 811–817, San Diego, CA, USA. Association for Computational Linguistics.

George Lakoff and Mark Johnson. 1980. *Metaphors We Live By*. Chicago University Press, Chicago, IL, US.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of ACL 2014*, pages 302–308, Baltimore, MD, USA.

Jordan J. Louviere and George G. Woodworth. 1990. Best–worst Analysis. Working paper. Department of Marketing and Economic Analysis, University of Alberta.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS 2013*, pages 3111–3119, Stateline, NV, USA. Curran Associates Inc.

Saif M. Mohammad, Ekaterina Shutova, and Peter Turney. 2016. Metaphor as a Medium for Emotion: An Empirical Study. In *Proceedings of *SEM 2016*, pages 23–33, Berlin, Germany. Association for Computational Linguistics.

Michael Mohler, Marc Tomlinson, David Bracewell, and Bryan Rink. 2014. Semi-Supervised Methods for Expanding Psycholinguistics Norms by Integrating Distributional Similarity with the Structure of WordNet. In *Proceedings of LREC 2014*, Reykjavik, Iceland. European Language Resources Association.

Nathalie Parde and Rodney D. Nielsen. 2018. A Corpus of Metaphor Novelty Scores for Syntactically-Related Word Pairs. In *Proceedings of LREC 2018*, pages 1535–1540, Miyazaki, Japan. European Language Resources Association.

Pragglejaz Group. 2007. MIP: A Method for Identifying Metaphorically Used Words in Discourse. *Metaphor and Symbol*, 22(1):1–39.

Marek Rei, Luana Bulat, Douwe Kiela, and Ekaterina Shutova. 2017. Grasping the Finer Point: A Supervised Similarity Network for Metaphor Detection. In *Proceedings of EMNLP 2017*, pages 1538–1547, Copenhagen, Denmark. Association for Computational Linguistics.

Ekaterina Shutova. 2015. Design and Evaluation of Metaphor Processing Systems. *Computational Linguistics*, 41(4):579–623.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black Holes and White Rabbits: Metaphor Identification with Visual Features. In *Proceedings of NAACL-HLT 2016*, pages 160–170, San Diego, CA, USA. Association for Computational Linguistics.

Ekaterina Shutova and Simone Teufel. 2010. Metaphor Corpus Annotated for Source–Target Domain Mappings. In *Proceedings of LREC 2010*, pages 3255–3261, Valetta, Malta. European Language Resources Association.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of EMNLP*

2008, pages 254–263, Honolulu, HI, USA. Association for Computational Linguistics.

Gerard J. Steen, Aletta G. Dorst, J. Berenike Herrmann, Anna Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A Method for Linguistic Metaphor Identification. From MIP to MIPVU.* John Benjamins Publishing Company, Amsterdam.

Maria Sukhareva, Judith Eckle-Kohler, Ivan Habernal, and Iryna Gurevych. 2016. Crowdsourcing a Large Dataset of Domain-Specific Context-Sensitive Semantic Verb Relations. In *Proceedings of LREC 2016*, pages 2131–2137, Portorož, Slovenia. European Language Resources Association.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor Detection with Cross-Lingual Model Transfer. In *Proceedings of ACL 2014*, pages 248–258, Baltimore, MD, USA. Association for Computational Linguistics.

Hannah Wieland. 2018. Crowd-Sourcing Novel Metaphor Annotations. Bachelor Thesis, TU Darmstadt.

Yorick Wilks. 1978. Making Preferences More Active. *Artificial Intelligence*, 11(3):197–223.

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of SIGMOD 1996*, pages 103–114, New York, NY, USA. Association for Computing Machinery.

A Guidelines

This task is about metaphors. A metaphor is a figure of speech that describes one thing by mentioning another thing. You can divide metaphors into two types: conventional and novel ones. In the following texts some words are marked as metaphorical.
You will be given four metaphors and your task is to decide which metaphor is the most conventional and which metaphor is the most novel.
Conventional metaphors are metaphors which are often used in everyday language.
In contrast novel metaphors which are usually not used in everyday language.
Please check the instructions and examples below before starting with the HIT!

Instructions and hints:

- Please read the whole context around the metaphors before deciding which metaphor is most novel or most conventional.
- You have to answer two questions per task:

  The first question is which metaphor is most *conventional*. That means you have to select the metaphor which is the most common in everyday language.

  The second question is which metaphor is the most *novel*. That means you have to select the metaphor that is the most uncommon in everyday language.
- Expressions that are so common that you cannot recognize them as a metaphor are candidates for the most conventional metaphor.
- Idioms are rather conventional metaphors.
- Each task is about four metaphors which are marked in red.
- The sentence which contain the metaphor are highlighted in bold letters.
- You have to fully complete all three tasks to get paid.

Examples:

1: “She **gave** him that idea.”
2: “I see the **point**.”
3: “Some books have to be **tasted**.”
4: “Time **flies.**”

- Answer: most conventional metaphor: **gave** (1); most novel metaphor: **tasted** (4).
- Explanation: “gave” is in the expression “to give someone an idea” so common that you can barely see the metaphor, that means it is extremely conventional. “to *taste* books” in contrast is a very uncommon use of the word “taste” and thus a novel metaphor. The other two metaphors are idioms which are conventional but not as conventional as “to give someone an idea.”.

Figure 5: Annotation guidelines for best-worst scaling HITs.

B Example of a Best-Worst Scaling HIT

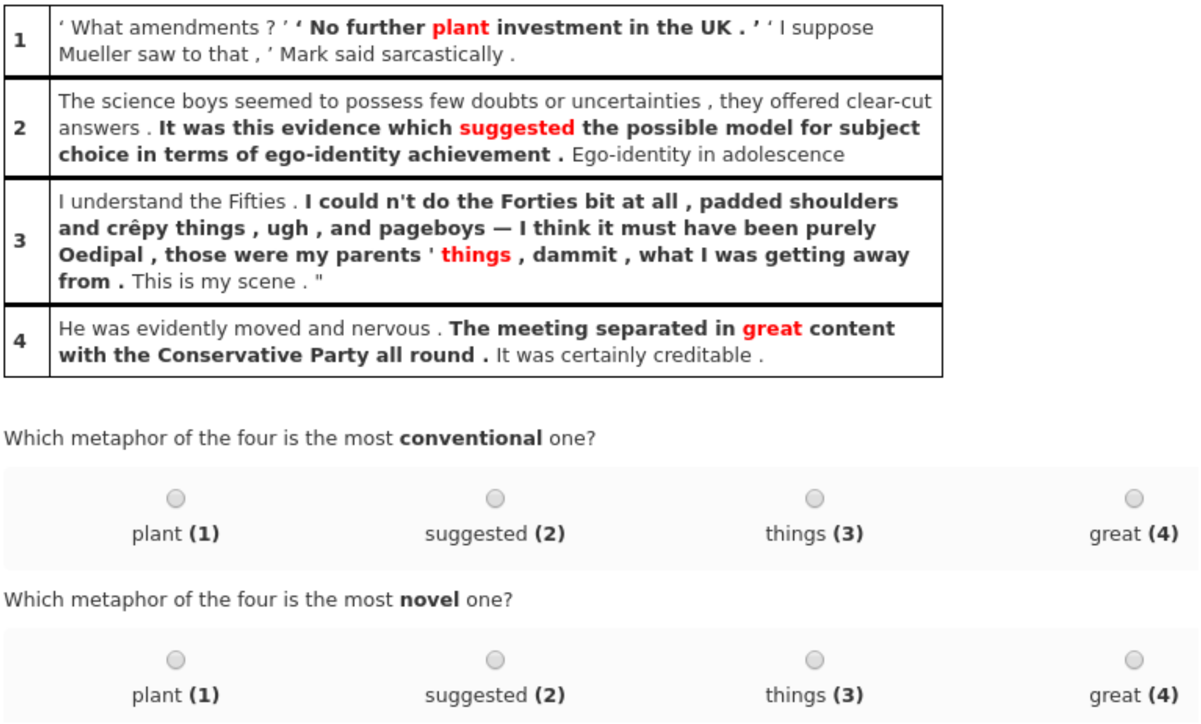

1	‘ What amendments ? ’ **‘ No further plant investment in the UK . ’** ‘ I suppose Mueller saw to that , ’ Mark said sarcastically .
2	The science boys seemed to possess few doubts or uncertainties , they offered clear-cut answers . **It was this evidence which suggested the possible model for subject choice in terms of ego-identity achievement .** Ego-identity in adolescence
3	I understand the Fifties . **I could n't do the Forties bit at all , padded shoulders and crêpy things , ugh , and pageboys — I think it must have been purely Oedipal , those were my parents ' things , dammit , what I was getting away from .** This is my scene . "
4	He was evidently moved and nervous . **The meeting separated in great content with the Conservative Party all round .** It was certainly creditable .

Which metaphor of the four is the most **conventional** one?

○ plant **(1)**
○ suggested **(2)**
○ things **(3)**
○ great **(4)**

Which metaphor of the four is the most **novel** one?

○ plant **(1)**
○ suggested **(2)**
○ things **(3)**
○ great **(4)**

Figure 6: Example of a best-worst scaling HIT, where annotators were asked to choose the most novel and the most conventionalized metaphor among 4 instances.

Streaming word similarity mining on the cheap

Olof Görnerup
RISE AI
SE-164 29 Kista, Sweden
olof.gornerup@ri.se

Daniel Gillblad
RISE AI
SE-164 29 Kista, Sweden
daniel.gillblad@ri.se

Abstract

Accurately and efficiently estimating word similarities from text is fundamental in natural language processing. In this paper, we propose a fast and lightweight method for estimating similarities from streams by explicitly counting second-order co-occurrences. The method rests on the observation that words that are highly correlated with respect to such counts are also highly similar with respect to first-order co-occurrences. Using buffers of co-occurred words per word to count second-order co-occurrences, we can then estimate similarities in a single pass over data without having to do prohibitively expensive similarity calculations. We demonstrate that this approach is scalable, converges rapidly, behaves robustly under parameter changes, and that it captures word similarities on par with those given by state-of-the-art word embeddings.

1 Introduction

Word similarities play an integral part in many natural language processing applications. Improving similarity estimates will therefore in turn potentially improve a broad range of areas, including word alignment (Songyot and Chiang, 2014), query expansion (Diaz et al., 2016), simplification (Biran et al., 2011), document classification (Arras et al., 2017), lexical substitution (McCarthy and Navigli, 2009) and many more.

The prevalent approach to estimate similarities is to first embed words in a vector space using techniques such as word2vec (Mikolov et al., 2013a,b) and GloVe (Pennington et al., 2014), and then calculate the similarities between words as the similarities between corresponding vectors. Finding all significant similarities among a set of words in this way, however, is computationally demanding due to the large number of pairwise similarity calculations involved (scaling as the square of the vocabulary size at worst). All-to-all similarity calculations are in particular strenuous, if at all feasible, in a streaming setting due to tight latency and memory constraints.

In this paper we address this by proposing a method that *finds significant similarities without calculating any similarities*. This seemingly contradictory feat is possible by explicitly counting *second-order* co-occurrences (SOCOs for short) and calculating correlations with respect to these. Two words w and v are said to have a SOCO if there is a third word u with whom both w and v co-occur (not necessarily together). For example, if the words *hot* and *coffee* co-occur at some point in a corpus or stream, and *hot* and *tea* co-occur at some other point in the same corpus or stream, then *coffee* and *tea* have a SOCO relation since they both co-occur with *hot*. The key observation then, as depicted in Fig. 1, is that *words that are highly correlated with respect to second-order co-occurrences are highly similar with respect to first-order co-occurrences*. This relation enables us to avoid pairwise similarity calculations altogether and instead acquire similarities directly from the SOCO counts.

The contribution of this paper is mainly twofold. Firstly, we introduce an operational definition of SOCO probabilities. To the best of our knowledge, this has not been done before in this explicit manner. Secondly, we apply this definition to efficiently estimate word similarities from streams. In practice we achieve this by keeping small buffers of co-occurred words per context word, and then incrementing SOCO counts of new co-occurred words and those in the buffer. Importantly, this enables us to pass over data only once. To ensure scalability in memory usage and runtime, approximate SOCO counts are then maintained in a count-min sketch table (Cormode and Muthukrishnan, 2005) that keeps the algorithm lightweight.

The benefits of our approach are not only com-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1425–1434
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

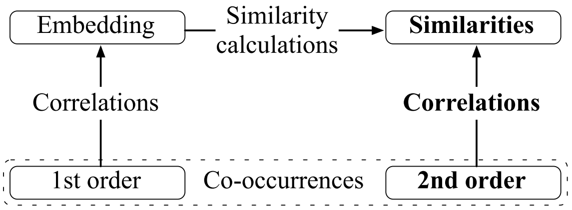

Figure 1: Diagram that describes the relationship between 1st and 2nd order co-occurrences. Instead of first embedding words with regard to 1st order co-occurrences and then calculating pairwise similarities between word vectors, we acquire similarities by directly correlating words with respect to their explicit 2nd order co-occurrences (encompassing labels in boldface).

putational in allowing us to avoid both multi-pass batch processing and the quadratic time complexity that comes with pairwise similarity calculations. Since the method is based on simple counting of SOCOs, it is also completely transparent and interpretable. By comparison, most first-order word embeddings – word2vec and GloVe are again good examples – are relatively opaque and difficult to interpret. The simplicity of our method also makes it straightforward to implement while having only a handful of parameters to tune compared to prevalent embedding methods that typically involve a large number hyperparameters.

The remainder of this paper is outlined as follows: Next we will relate our proposed method to current approaches for calculating word similarities. The method is described in Sec. 3, followed by a complexity analysis in Sec. 4 where we theoretically and experimentally show that our approach indeed is suitable for stream mining. In Sec. 5, we evaluate the method with respect to convergence, accuracy and parameter sensitivity. The paper is concluded in Sec. 6 with a summary and a discussion on possible future directions.

2 Related work

The explicit use of second-order co-occurrences has so far gained little attention in word similarity mining. The typical approach is rather to express word similarities as similarities between vector representations that in turn are based on first-order co-occurrences (this is also the case in (Islam and Inkpen, 2006), despite the name of their approach[1]).

There is a large body of work in this area, predominantly based on Harris' distributional hypothesis (Harris, 1954), from seminal approaches such as LSA/LSI (Deerwester et al., 1990), HAL (Lund and Burgess, 1996) and Random indexing (Kanerva et al., 2000) and onward. See (Levy et al., 2015) for an extensive review. A common approach then is to represent words in terms of co-occurrence correlations – e.g. using Pointwise mutual information (PMI) (Church and Hanks, 1990) and variants thereof (Levy et al., 2015) – either explicitly or through dimensionality reduction (Pennington et al., 2014). Another prevalent approach is to generate vector representations based on prediction tasks, where words are predicted from their contexts or vice versa; Continuous bag of words and Skip-gram (Mikolov et al., 2013a,b) are prominent examples. However, these methods are batch-based and typically require multiple passes over data (3 to 50 training epochs in the case of word2vec, for instance (Mikolov et al., 2013b)). Approaches that are more suitable for streaming data have also been developed, e.g. for calculating the most PMI-correlated words per word (Durme and Lall, 2009) and, more recently, neural network methods (primarily based on Skip-gram) adapted to enable incremental updating (Luo et al., 2015; Kaji and Kobayashi, 2017; Bamler and Mandt, 2017; Peng et al., 2017). Note though that in all of the above cases vector representations capture *first-order* co-occurrences. When using PMI for example, the correlation between two words is high if they co-occur more frequently than expected from chance, but high correlation does not equate high similarity (consider the words *red* and *wine* for example). Estimating word similarities would therefore require the extra step of calculating similarities between vectors, something that our approach bypasses by explicitly counting SOCOs.

3 Method

3.1 Second-order co-occurrences

We define the SOCO probability of words w and v as

$$P(w\!:\!v) := \sum_{u\in\mathcal{V}} P(u)P(w|u)P(v|u), \tag{1}$$

where $w\!:\!v$ denotes a SOCO and $\mathcal{V}$ is the vocabulary. That is, $P(w:v)$ is the probability that two

---

[1] In general, the term *second-order co-occurrence* is sometimes used in NLP to describe second-order representations based on first-order word vectors, such as in (Schütze, 1998). This does not involve explicit SOCO counts as in our case though.

randomly selected words co-occurring with a randomly selected word u are w and v. Since

$$\begin{aligned} P(w\!:\!v) &= \sum_{u\in\mathcal{V}} P(w,u)P(v|u) \\ &= \sum_{u\in\mathcal{V}} P(w)P(u|w)P(v|u), \end{aligned} \tag{2}$$

an alternative interpretation of SOCO is the chain of randomly selecting a word w, one of its co-occurring words u, and in turn one of *its* co-occurring words v.

3.2 Correlation measure

Our ansatz is that two words have a high degree of similarity if they are highly correlated with respect to SOCO, since this would imply that the words are relatively interchangeable in their respective contexts. We quantify a SOCO correlation using standard pointwise mutual information (PMI):

$$M(w\!:\!v) = \log_2 \frac{P(w\!:\!v)}{P(w)P(v)}, \tag{3}$$

where the denominator is the SOCO probability of w and v given that they are independent of u, since then

$$P(w\!:\!v) = \sum_{u\in\mathcal{V}} P(u)P(w)P(v). \tag{4}$$

3.3 Estimating correlations

Making the simplifying assumptions that a stream is stationary and that co-occurrence correlations decay rapidly with word-to-word distance in the stream (or corpus), we can estimate Eq. 3 in one pass using counters of word occurrences and SOCOs. See Fig. 2 for a schematic overview and Algorithm 1 for a detailed description of our approach.

3.3.1 Co-occurrence buffers

Approximate SOCO counts are maintained by keeping small buffers (on the order of 1 to 10 words) of previously observed words with a given context. The context of a word is here given by the position relative to the word (say, -1 for the preceding word) and the word occupying that position. For example, if we observe the word *fox* in the sequence

> *The quick brown fox jumps over the lazy dog.*

fox is added to the buffer of previously observed words with context (-1, *brown*) (*bear*, *bag* and *eyes* perhaps). Note that *fox* then has a SOCO relation with the words in the buffer. The SOCO counters of (*fox*, w) for all words w in the buffer are therefore incremented (for (*fox*, *bear*) for instance) before *fox* is added to the buffer. If the buffer is full – we cap the number of prior words stored – the oldest word is discarded prior to adding the new one.

(a) Second-order co-occurrences

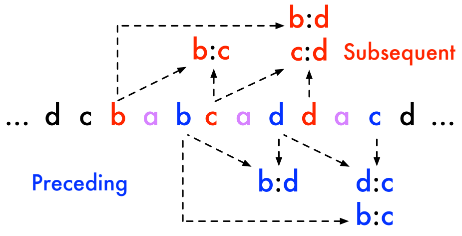

(b) Co-occurrence buffers

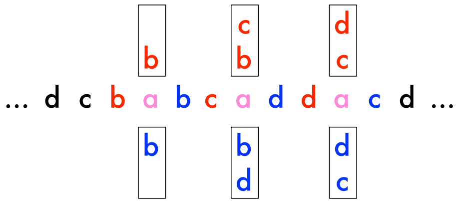

Figure 2: (a) Words that share a context word (the word a in this example) at a given position (as a preceding and subsequent word, in blue and red, respectively) are said to co-occur to the second order. For example, b and c have a SOCO relation $b:c$ since they both have a as a subsequent word. (b) We count second-order co-occurrences by keeping buffers of co-occurring words for a given context. The buffer for context (1, a) is shown above the sequence (i.e., when a occurs as a subsequent word) and (-1, a) below the sequence (a as a preceding word). Both buffers (implemented as queues, where the most recent observations are stored) are limited to two words in this example. Before a word is added to a buffer, the second-order co-occurrence counts of the word and the words in the buffer are incremented. For instance, the counts of $b\!:\!d$ and $c\!:\!d$ are incremented before the buffer $[c, b]$ is updated to $[d, c]$.

The same procedure is performed for all context positions in a sliding window of a given length (e.g., for positions $\{-2, -1, 1, 2\}$ if we consider symmetric contexts in a five-word window). By simultaneously estimating word frequencies by counting current words we can in this way incrementally maintain estimates of $M(w\!:\!v)$.

3.3.2 Probabilistic counting

Since the number of SOCOs may be very large, we keep approximate counts of these using a count-min sketch table (Cormode and Muthukrishnan, 2005). The table, denoted d, has h rows and g columns, where the rows are associated with h pairwise independent hash functions, f_i, and where entries are initialized to 0. When a SOCO $w : v$ for words w and v is observed, each hash function maps $w : v$ to an index $f_i(w : v)$ of row i. Entry $(i, f_i(w\!:\!v))$ is then incremented by 1.

After populating d the approximate count of $w\!:\!v$ is given by the minimum count of the entries $(i, f_i(w\!:\!v))$:

$$\tilde{c}(w\!:\!v) = \min_i d(i, f_i(w\!:\!v)). \tag{5}$$

Note that $\tilde{c}$ may overestimate the true count $c(w\!:\!v)$ due to hash collisions (this is indeed what keeps the data structure sublinear with respect to the number of SOCOs). To reduce overestimations we can employ conservative updates by only updating an entry $(i, f_i(w\!:\!v))$ if it is exceeded by the current incremented approximate SOCO count (Goyal et al., 2012):

$$d(i, f_i(w\!:\!v)) \leftarrow \max\{d(i, f_i(w\!:\!v)), \tilde{c}(w\!:\!v)\}. \tag{6}$$

This is the approach used in all experiments presented below.

3.3.3 Top-k correlations

Rather than storing all SOCO correlations, which is infeasible for large vocabularies and wide context ranges, we keep track of the k most correlated words for each word in the vocabulary. We achieve this scalably by adapting the approach proposed by Durme and Lall for calculating first-order co-occurrence correlations in streams (Durme and Lall, 2009). We keep track of occurred SOCOs in non-overlapping meta-windows (on the order of 10^5 to 10^7 tokens long). For each meta-window, we calculate PMIs for occurred SOCOs using exact word counts (having a comparably small memory footprint) and approximate SOCO counts, and then update priority queues with the most similar words per word accordingly. In this way the number of SOCOs stored is kept approximately constant as we consume the stream.

4 Complexity analysis

4.1 Time

For each token in the stream, updating word counts takes constant time using an associative array. The algorithm also goes through $2n$ context words, where n is the context range. For each context word, the count-min table that stores approximate SOCO counts is updated at most r times, where r is the maximum buffer size. Since each update takes $O(h)$ time for a count-min table with h rows (a hash function is called for each row), the time complexity for updating counters is $O(nrh)$. Updating a co-occurrence buffer, i.e. by inserting and (possibly) deleting a word index in a queue, takes $O(1)$ time, and so the total time complexity per token when consuming a meta-window is $O(nrh)$.

Algorithm 1 Estimate $M(w\!:\!v)$ for stream $[s_1, s_2, ..., s_m]$, vocabulary $\mathcal{V}$, context range n, buffer capacity r, meta-window size N and count-min (C-m) table with h rows and g columns. Note that m may approach infinity.

```
$l \leftarrow \{-n, ..., -1, 1, ..., n\}$ {context positions}
$c(w) \leftarrow 0; w \in \mathcal{V}$ {word counts}
$d_{i,j} \leftarrow 0; i = 1, ..., g, j = 1, ..., h$ {C-m table}
$t_w \leftarrow 0$ {total word count}
$t_s \leftarrow 0$ {total SOCO count}
$q(i, w) \leftarrow \emptyset; w \in \mathcal{V}, i \in l$ {context queues}
$\mathcal{S}(w) \leftarrow \emptyset; w \in \mathcal{V}$ {similar word queues}
$\mathcal{P} \leftarrow \emptyset$ {observed SOCOs}
for $i = n + 1$ to $m - n$ do {consume stream}
  $c(s_i) \leftarrow c(s_i) + 1$
  $t_w \leftarrow t_w + 1$
  for $j \in l$ do
    for $v \in q(j, s_{i+j}), v \neq s_i$ do
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{s_i\!:\!v\}$
      $d(s_i\!:\!v) \leftarrow d(s_i\!:\!v) + 1$
      $t_s \leftarrow t_s + 1$
    end for
    if $|q(j, s_{i+j})| = r$ then {buffer full}
      $q(j, s_{i+j})$.dequeue() {discard oldest}
    end if
    $q(j, s_{i+j})$.enqueue($s_i$) {add current}
  end for
  if $i$ mod $N = 0$ then {meta-window ends}
    for $w \in \mathcal{V}$ do
      $P(w) \leftarrow c(w)/t_w$
    end for
    for $w\!:\!v \in \mathcal{P}$ do
      $P(w\!:\!v) \leftarrow d(w\!:\!v)/t_s$
      $M = \log_2[P(w\!:\!v)/(P(w)P(v))]$
      Update $\mathcal{S}(w)$ with $v$ and priority $M$
    end for
    $\mathcal{P} \leftarrow \emptyset$ {clear observed SOCOs}
  end if
end for
```

Between meta-windows, we update the top cor-

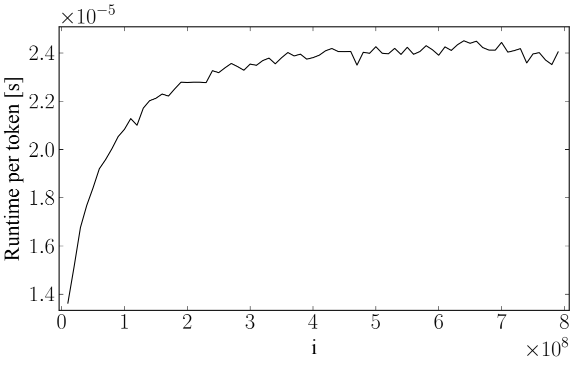

Figure 3: Runtime per token at stream position i (Python implementation run on a MacBook Pro with a 2.8 GHz Intel Core i7 processor and 16 GB of memory).

related words per word. We first update frequency estimates of words, which takes $O(|\mathcal{V}|)$ time. For each SOCO that has been observed in the meta-window, $|\mathcal{P}|$ of them, we update the priority queues with the most correlated words of the words involved in the SOCOs. Each such update takes $O(\log k)$, where the parameter k is the number of most correlated words of a word.

Altogether, the time complexity is hence $O(nrh + \frac{|\mathcal{V}|+|\mathcal{P}|\log k}{N})$ per token. Note that $n, r, h, k \ll |\mathcal{V}|, |\mathcal{P}|$, where n, r, h and k are all small fixed parameters ($n \sim 1$ to 10, $r \sim 10$, $k \sim 10$ to 100 and $h \sim 10$ typically). Given that the stream is roughly stationary (which has been the case in our experiments), the size of $\mathcal{P}$ is approximately constant. With a fixed vocabulary and N, the runtime per token is therefore also kept approximately constant. We confirm this experimentally, see Fig. 3, by running the algorithm on English Wikipedia (as of March 7, 2015) using a context range of one (window size three), a vocabulary constituted by the 10^4 most frequent words, meta-windows of length 10^7 and five-word co-occurrence buffers, a count-min table with 8 rows and $3.4 \cdot 10^7$ columns, and where the 10 most similar words per word are stored. As expected, the runtime per token increases initially as the co-occurrence buffers are filled up, and then converges towards a constant value.

4.2 Space

With regard to memory the algorithm requires that we keep a co-occurrence buffer for each word in the vocabulary and for each context position. The space complexity of the buffers is therefore $O(nr|\mathcal{V}|)$. In addition, there are $|\mathcal{V}|$ word counters and priority queues of size k for storing top-k similar words, a count-min sketch table with gh entries, where the fixed parameter g is the number of columns in the count-min table, and a index pair set of size $|\mathcal{P}|$ for storing occurred SOCOs that have occurred in a meta-window. The total space required is hence $O((nr + k)|\mathcal{V}| + gh + |\mathcal{P}|)$.

For example, assume we have a vocabulary with a million words, a context range of 5 (i.e., window size 5+1+5), a buffer size of 10, a count-min table with $3 \cdot 10^8$ columns and 8 rows, and that we keep the 10 most similar words per word. We then need to store $10^6 \cdot 2 \cdot 5 \cdot 10 = 10^8$ items in the co-occurrence buffers. If each item requires 4 bytes (a word index constituted by an unsigned integer), the buffers take up 400 MB of space. Add to that another 4 MB for the word counters (10^6 of them à 4 bytes), 40 MB ($10^6 \cdot 4 \cdot 10$ bytes) for the top 10 similar word indices per word and 9.6 GB ($8 \cdot 3 \cdot 10^8 \cdot 4$ bytes) for the count-min sketch table. Set the meta-window length so that $|\mathcal{P}| \leq 7.4 \cdot 10^8$ (taking up a bit less than 6 GB of memory at most) and we end up with a total footprint of approximately 16 GB – a memory requirement even met by many present-day laptops.

5 Evaluation

5.1 Examples

In Table 1 we show a set of examples of the most SOCO-correlated words per word as output by the algorithm (using Wikipedia, a context range of 2, buffer size of 10 and a count-min table with 8 rows and $2.7 \cdot 10^8$ columns). Although the examples are anecdotal, they illustrate that the method manages to mine word similarities that make intuitive sense, such that *Wednesday* is most similar to other days in the week (interestingly, it is most similar to adjacent days, *Tuesday* and *Thursday*) and *yellow* is most similar to other colors, etc.

5.2 Convergence

It is crucial that the method converges within a reasonable amount of time in order for it to be of practical use. To test this we run the algorithm and measure how the sets of top-k similar words change as the stream progresses. We quantify these changes using the Jaccard index between a word set of a word w at position i in the stream, $\mathcal{S}_i(w)$, and the corresponding word set at Δi tokens prior,

musician	increasing	croatian	wednesday	scholar	hermann	coventry	yellow
singer	reducing	yugoslav	thursday	scholars	heinrich	leicester	purple
pianist	growing	serbian	tuesday	translator	friedrich	norwich	pink
songwriter	increased	slovenian	monday	playwright	wilhelm	stoke	orange
guitarist	reduces	croatia	friday	philosopher	georg	swansea	blue
rapper	reduce	slovak	saturday	poet	wolfgang	cardiff	red

Table 1: Examples of the most correlated words per word with respect to second-order co-occurrence.

	SIMLEX	SIMVERB	MT-287	MT-771	WS-353
SOCO	0.41	0.25	0.59	0.56	0.71
FOCO	0.35	0.23	0.65	0.59	0.66
GLOVE	0.31	0.18	0.61	0.57	0.63
CBOW	0.34	0.22	0.66	0.57	0.69
SGM	0.41	0.32	0.67	0.60	0.72
Coverage	0.99	0.96	0.95	0.98	0.98

Table 2: Top rows: Spearman's rank correlation coefficients for different methods and benchmarks. Bottom row: fractions of benchmark word pairs covered by the methods and the corpus.

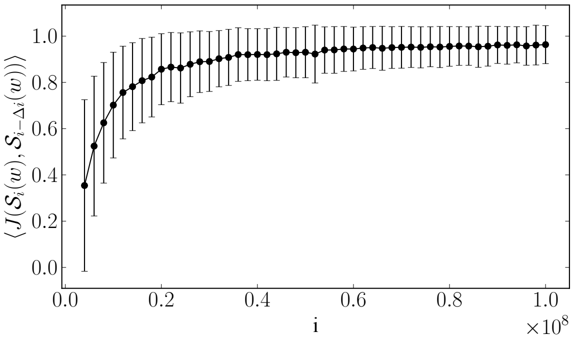

Figure 4: Average change of top-k word sets over words at stream position i. Standard deviation shown by error bars.

$\mathcal{S}_{i-\Delta i}(w)$. That is,

$$J(\mathcal{S}_i(w), \mathcal{S}_{i-\Delta i}(w)) = \frac{|\mathcal{S}_i(w) \cap \mathcal{S}_{i-\Delta i}(w)|}{|\mathcal{S}_i(w) \cup \mathcal{S}_{i-\Delta i}(w)|}, \tag{7}$$

where $J = 1$ if there is no change and the sets are identical. As seen in Fig. 4, the top-k sets converge as both the mean of $J(\mathcal{S}_i(w), \mathcal{S}_{i-\Delta i}(w))$ over words w tends towards one while the standard deviation decreases. In this experiment, applied on Wikipedia and a vocabulary of 10^4 words, the top ten words per word were stored. Further, $\Delta i = 10^6$ tokens, contexts of range one, a count-min table with 8 rows and $3.4 \cdot 10^7$ columns, and co-occurrence buffers of size five were used.

5.3 Accuracy

To quantitatively evaluate the accuracy of the method, we use a collection of established word similarity benchmarks: SimLex-999 (SIMLEX) (Hill et al., 2015), SimVerb (SIMVERB) (Gerz et al., 2016), MTurk-287 (MT-287) (Radinsky et al., 2011), MTurk-771 (MT-771) (Halawi et al., 2012) and WordSim (WS-353) (Finkelstein et al., 2001). Each of these benchmarks contains a set of word pairs and their similarities as judged by human annotators. Comparing these word rankings with rankings given by Eq. 3, we get an indication of how well our method captures human notions of similarity and relatedness. The agreement is quantified with the standard Spearman's rank correlation coefficient.

The results are also compared to the ranking agreements for popular word embeddings – GloVe (Pennington et al., 2014) (GLOVE), Continuous Bag of words (CBOW) and Skip-gram (SGM) (Mikolov et al., 2013a,b) – as well as for the pointwise mutual information between regular first-order co-occurrences (FOCO). In all these cases, word similarities are given by the cosine similarity,

$$\sigma(v_i, v_j) = \frac{v_i \cdot v_j}{|v_i|_2 |v_j|_2}, \tag{8}$$

where v_i and v_j are vectors associated with words i and j. All word embeddings are in 300 dimensions. Remaining parameters are set to default values, with the exception that no explicit word count thresholding is used in order to avoid loss of benchmark words. (Applying thresholds at 3-5 occurrences gives comparable results as those reported here, but for fewer word pairs.) For all benchmarks the One billion word corpus[2] (Chelba et al., 2013) is used with a vocabulary consisting of the $3 \cdot 10^4$ most frequent words, except the Stanford CoreNLP stopwords.[3] A context range of 2, count-min tables with 8 rows and $3.4 \cdot 10^7$ columns, and a buffer size of 10 are used throughout. Since many word pairs in the benchmarks are dissimilar we keep all SOCO correlations for the benchmark word pairs in this experiment. A small fraction of benchmark words are either not present in the corpus or captured by SOCO or the embedding methods due to sampling. To ensure a fair comparison between methods, only word pairs that are represented by all approaches are therefore considered.

The results are summarized in Table 2, where we note that the coverage (the fraction of benchmark pairs represented) is high, from 95% for MT-287 to 99% for SIMLEX. The relative performance of SOCO varies over benchmarks: in two out of five cases the performance is roughly on par with SGM, and compared to GLOVE, CBOW and FOCO, our method performs best in three out of five benchmarks. Thus the overall picture is that our approach indeed is able to capture meaningful similarity relations, and that it performs comparably to regular first-order word embedding methods.

5.4 Parameter sensitivity

The algorithm has two key parameters: the co-occurrence buffer capacity and the size of the count-min sketch table. Since the approximation errors induced by the latter is thoroughly analyzed in (Cormode and Muthukrishnan, 2005) we will here focus on how the buffer size influences similarity accuracy. Using the same corpus and benchmark suite as in Sec. 5.3, we evaluate how the accuracy varies with the buffer capacity. To again make a fair comparison, we then only include those benchmark word-pairs that are covered in all experiments.

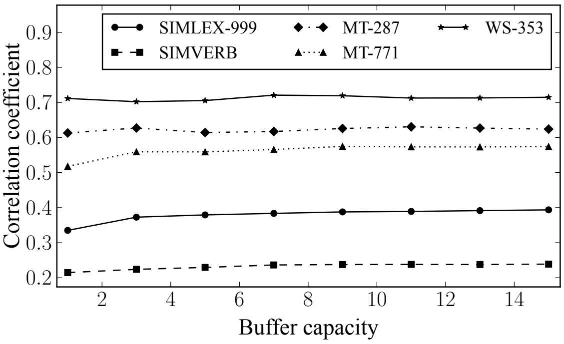

Figure 5: Spearman's rank correlation coefficients for different benchmarks and buffer capacities.

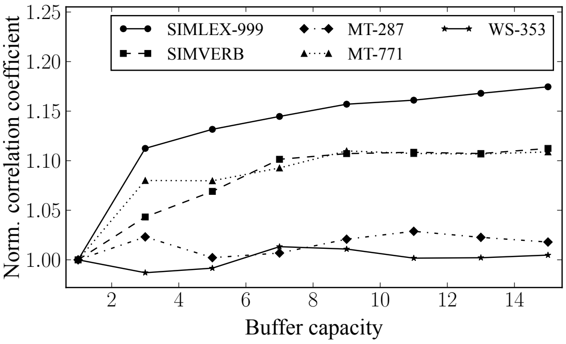

Figure 6: Relative Spearman's rank correlation coefficients with respect to correlations at buffer capacity 1 for different benchmarks and buffer capacities.

As seen in Fig. 5, the method is insensitive to the buffer capacity size as the accuracy stays approximately constant for buffers of sizes larger than 3. This is also the case in relative terms: see Fig. 6 where we plot the accuracy relative to the accuracy for buffers of size 1. The improvement is then largest for SIMLEX-999, where going from buffers of size 1 to 15 yields an approximate difference of 17% in accuracy. The relative accuracy, however, varies little from buffer capacity 3 and upward. With regard to coverage, see Fig. 7, the buffer capacity has a significant effect up until buffer size 9, after which the coverage settles at around 95-99%.

We can conclude that the method is robust with regard to buffer capacity, resulting in predictable and smooth changes in output, and that it suffices to keep small buffers to maintain both accuracy and coverage. Since the buffers in effect tend to store the most frequent co-occurrences per word, these results indicate that it is possible to accurately estimate similarities using only salient co-occurrences. This is also supported by Polajnar and Clark's finding (2014) that only a handful of the most frequent context words yields the best results when estimat-

---

[2] http://www.statmt.org/lm-benchmark/

[3] https://stanfordnlp.github.io/CoreNLP/

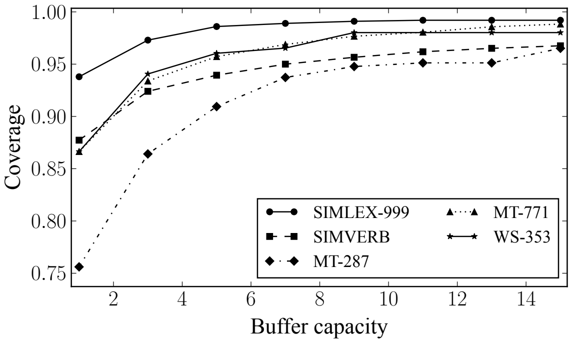

Figure 7: Coverage for different benchmarks and buffer capacities.

ing similarities from co-occurrence frequencies.

6 Conclusions

We have presented a method for estimating word similarities from corpora or streams using an explicit notion of SOCO. Our approach is simply to count such co-occurrences and calculate correlations between words with respect to these counts. Words that are highly correlated are then also highly similar with respect to first-order co-occurrences. By using co-occurrence buffers and count-min sketches for estimating SOCO counts, the method keeps both the runtime per token and memory usage constant while only needing one pass over data. These properties makes our approach ideal for low-cost stream mining.

Despite its simplicity and modest computational requirements, benchmark experiments show that the method performs comparably to calculating similarities between best-in-class word vectors. This not only makes our approach a feasible alternative for calculating word similarities on the cheap, but in some cases it may be the only viable option. Consider for instance an embedded system in a decentralized machine learning or edge computing scenario. Then real-time computing constraints and scarce memory would rule out both multi-pass word embeddings and pairwise similarity calculations in favor of similarities readily available from SOCO counts.

There are numerous possible future directions, exploring correlation measures other than PMI being one. Also, by grouping words in concurrence with finding top-k similar words per word, an extended method could be used for word clustering. Possible ways to find groups of inter-similar words – constituting abstract concepts (Görnerup et al., 2017) – is then to use label propagation on a graph (Raghavan et al., 2007) (with words constituting vertices and the top-k similar words directed edges), or agglomerative hierarchical clustering. How to do this efficiently and scalably is currently under study. Moreover, in this paper we have exclusively considered text data and word similarities. The method, however, is domain-agnostic and may be applied on other types of data, in and beyond the NLP domain. There is a wide range of potential application areas, presumably in everything from biology and physics to social science and economics – in essence in any domain where objects co-occur and where these co-occurrences carry some relevant information or meaning.

Acknowledgments

The authors thank the anonymous reviewer for bringing to their attention the link between most frequent co-occurrences and the sufficiency of small buffer sizes.

This work was funded by the Swedish Knowledge Foundation through the BIDAF project and by the Swedish Foundation for Strategic Research through the Continuous Deep Analytics project.

References

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. “What is Relevant in a Text Document?”: An Interpretable Machine Learning Approach. *PLOS ONE*.

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 380–389.

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting It Simply: A Context-aware Approach to Lexical Simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT ’11, pages 496–501. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Graham Cormode and S. Muthukrishnan. 2005. An Improved Data Stream Summary: The Count-min Sketch and Its Applications. *J. Algorithms*, 55(1):58–75.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 367–377. Association for Computational Linguistics.

Benjamin V. Durme and Ashwin Lall. 2009. Streaming Pointwise Mutual Information. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 1892–1900. Curran Associates, Inc.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 406–414. ACM.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In *EMNLP*.

Olof Görnerup, Daniel Gillblad, and Theodore Vasiloudis. 2017. Domain-agnostic discovery of similarities and concepts at scale. *Knowledge and Information Systems*, 51(2):531–560.

Amit Goyal, Hal Daumé, III, and Graham Cormode. 2012. Sketch Algorithms for Estimating Point Queries in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1093–1103. Association for Computational Linguistics.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1406–1414. ACM.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with genuine similarity estimation. *Comput. Linguist.*, 41(4):665–695.

A. Islam and D. Inkpen. 2006. Second order co-occurrence PMI for determining the semantic similarity of words. In *LREC 2006*, pages 1033–1038.

Nobuhiro Kaji and Hayato Kobayashi. 2017. Incremental skip-gram model with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 363–371. Association for Computational Linguistics.

P. Kanerva, J. Kristofersson, and A. Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, volume 1036.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Hongyin Luo, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2015. Online learning of interpretable word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1687–1692.

Diana McCarthy and Roberto Navigli. 2009. The english lexical substitution task. *Language Resources and Evaluation*, 43(2):139–159.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR).*

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Hao Peng, Jianxin Li, Yangqiu Song, and Yaopeng Liu. 2017. Incrementally learning the hierarchical softmax function for neural language models. In *AAAI*, pages 3267–3273. AAAI Press.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Tamara Polajnar and Stephen Clark. 2014. Improving distributional semantic vectors through context selection and normalisation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 230–238.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 337–346. ACM.

Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, 76:036106.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Journal of Computational Linguistics*, 24:97–123.

Theerawat Songyot and David Chiang. 2014. Improving word alignment using word similarity. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1840–1845, Doha, Qatar. Association for Computational Linguistics.

Memory, Show the Way:
Memory Based Few Shot Word Representation Learning

Jingyuan Sun[1,2], **Shaonan Wang**[1,2], **Chengqing Zong**[1,2,3]
[1] National Laboratory of Pattern Recognition, CASIA, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
`{jingyuan.sun, shaonan.wang,cqzong}@nlpr.ia.ac.cn`

Abstract

Distributional semantic models (DSMs) generally require sufficient examples for a word to learn a high quality representation. This is in stark contrast with human who can guess the meaning of a word from one or a few referents only. In this paper, we propose Mem2Vec, a memory based embedding learning method capable of acquiring high quality word representations from fairly limited context. Our method directly adapts the representations produced by a DSM with a longterm memory to guide its guess of a novel word. Based on a pre-trained embedding space, the proposed method delivers impressive performance on two challenging few-shot word similarity tasks. Embeddings learned with our method also lead to considerable improvements over strong baselines on NER and sentiment classification.

1 Introduction

Humans can learn a new word quickly from minimal exposure to its context, as in the following example:

> The *Labrador* runs happily towards me, barking and wagging its tail.

Even this is the first time one hears about *Labrador*, we can guess it should be an animal or even further a dog easily, since it runs, barks and has a tail. Such ability to efficiently acquire representation from small data, namely fast mapping, is thought to be the hallmark of human intelligence that a cognitive plausible agent should strive to reach (Xu and Tenenbaum, 2007; Lake et al., 2015).

However, as the mainstream of text representation learning in NLP, most distributed semantic models (DSMs) don't fare well in tiny data (Lazaridou et al., 2017; Herbelot and Baroni, 2017; Wang et al., 2016). Even if they have learned a lot of words, they still need sufficient examples to acquire a high-quality representation for a novel word. This not only constitutes a blow to DSM's cognitive plausibility but also limits its practical usage in NLP. Because plentiful enough data is not always available, especially in domain specific tasks. Even if a large corpora is at hand, low-frequency words in it are still more than highly frequent ones, according to the Zipfian distribution of natural language.

Given the above reasons, it's desirable to build a word embedding method capable of acquiring high quality representations with limited contexts, i.e., few shot word representation learning. We take lessons from hypothesis constraint (HC) theory to achieve this goal. HC is an influential proposal for human's fast mapping (Xu and Tenenbaum, 2007). It indicates that people learn a new word by eliminating incorrect hypotheses about the word meaning, based on usage of the target word and prior knowledge of context words. This is instructive to us since embedding a word in the high-dimensional vector space also faces nearly unlimited candidate hypotheses (Wang et al., 2018) . General DSMs can't efficiently handle these candidates, so they fall back on multiple context examples to find the path, while we propose to let a memory show the way. We augment DSM with an longterm memory to transfer knowledge from a large general domain corpora to adapt the representation learning on the small text. In context of the HC theory, we directly constrain the hypothesis a DSM makes about the target word by its current usage and prior knowledge acquired from a large corpora. Experiments show our method makes educated guess of a novel word efficiently with fairly limited examples, just as humans do in the fast mapping.

It's worth noting that us attaching importance

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1435–1444
Brussels, Belgium, October 31 - November 4, 2018.

to few-shot word representation learning doesn't mean we need to learn words, no matter frequent or rare, all in the few-shot way. DSMs have done pretty well in frequent words learning with large corpus. We augment a DSM with an external memory for few-shot representation learning, under the assumption that gradual acquisition of frequent words plus fast learning of rare words make an integrated word representation learning scheme. Our ultimate goal is certainly an all-round architecture that learns text representations from any amount of data. We believe Mem2Vec, which bridges the word representation learning from big data to small text, will be a building block of that ideal architecture.

The primary contribution of this work is a memory augmented word embedding model with a fast adaptation mechanism, capable of learning representations efficiently from tiny data. Experimental results show that the proposed Mem2Vec learns high quality target word representation with both single informative sentence and a few casual sentences as contexts. To show its performance in downstream applications, Mem2Vec is used to pre-train embeddings for three NER tasks and also surpasses strong baselines. Since our model transfers from general domain corpus to a target small text, it is highly possible to face the problem of domain shift. Mem2vec is impressively competent in tackling domain shift, as demonstrated in a series of cross-domain sentiment classification tasks.

2 Related Work

Rare Word Embedding Acquiring representations for rare words has long been a well-known challenge of natural language understanding (Herbelot and Baroni, 2017; Wang and Zong, 2017). Khodak et al. (2018) learn a linear transformation with pretrained word vectors and linear regression, which can be efficient adapted for novel words. Lazaridou et al. (2017) directly sum the context embedding of a novel word as its representation, based on a pre-trained embedding space. Though not explicitly stated, their idea actually matches the HC theory. They constrain the hypothesis solely within the current context of the target word which we think is not enough. We constrain the hypothesis with memory and the context. Another strand of solutions rely on auxiliary information, such as morphological structure (Luong et al., 2013; Kisselew et al., 2015) and external knowledge (Long et al., 2017). Lazaridou et al. (2013) derive morphologically complex words from sub-word parts with phrase composition methods. Ling et al. (2015) read characters of the rare word with a bidirectional LSTM to deal with open vocabulary problem in language modeling and NER. Hill et al. (2016) learn an embedding of a dictionary definition to match the pre-trained headword vector, while Weissenborn (2017) refines the word embeddings with explicit background knowledge from a commonsense knowledge base. Different from this strand of work, our method doesn't fall back on auxiliary information. We acquire knowledge from a large unlabeled general domain corpora which is widely available.

Cross Domain Word Embedding The knowledge accumulation phase of our model aims to learn an embedding space from a large general domain corpora. This is partially in line with cross domain word embedding work. Among these work, a strand of approach hypothesizes that a word frequent in multiple domains should mean nearly across these domains. Bollegala et al. (2015) call such word pivot, share its embeddings across domains and use them to predict the surrounding non-pivots. Yang et al. (2017) selectively incorporate source domain information to target domain word embeddings with a word-frequency-based regularization. These pivot-based methods have delivered improvements on sentiment analysis and NER. However, they have a defect that only limited target domain words benefit from the knowledge transfer.

Memory based Meta Learning Memory augmented neural networks (MANN) are widely used in different tasks for efficient recall of experience and fast adaptation to new knowledge (Bahdanau et al., 2014; Merity et al., 2017; Miller et al., 2016; Grave et al., 2017; Sprechmann et al., 2018; Wang et al., 2017). Intuitively, Meta-learning, which aims to train a model that quickly adapts to a new task, should benefit from memory architecture, and empirically it does do (Santoro et al., 2016; Duan et al., 2016; Wang et al., 2016; Munkhdalai and Yu, 2017; Kaiser et al., 2017). The memory we use is closely related to (Kaiser et al., 2017), but still get three major differences. First,they only retrieve the single nearest neighbor from the memory while we retrieve an average of the K nearest neighbors weighted by how they match the current context. Second, they focus on supervised learn-

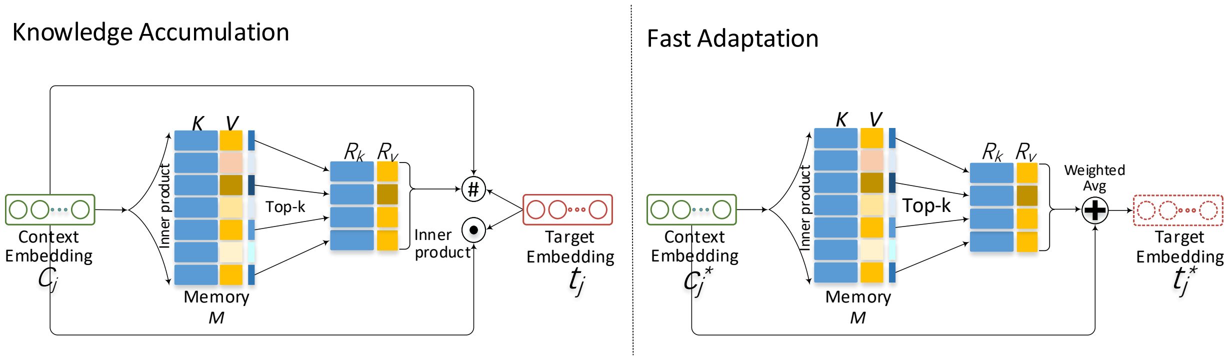

Figure 1: The proposed model architecture. *K* and *V* respectively denotes the key and value vector of the memory. # refers to equation(3). Left: Knowledge accumulation phase. The model learns word embeddings and store prototypes in memory. Right: Fast Adaptation phase. Prototypes retrieved from memory with the given context are combined with the context embedding to form the target word representation.

ing and don't have a fast adaptation mechanism for acquiring representation. Third, they update the memory according to whether the returned value is strictly the same as the target. However, synonyms are common in natural language text. We thus take a softer criterion and update the memory according to vector similarity between the addressed value and the target word embedding.

3 Methods

Our model in brief is a neural network based DSM augmented by a longterm memory. As showed in Fig.1, it operates in two consecutive phases, first accumulating knowledge and then doing fast adaptation on new words, just as the human learning process goes. In the knowledge acquisition phase, we train the memory augmented DSM to learn a semantic space. We also accumulate similar contexts of target words in the memory and gradually form "prototype" representations. The pre-trained embedding space and the saved prototypes are just the knowledge acquired. The fast adaption phase occurs when we need to learn a new word from minimal context. In this phase, we directly combine the context embedding and retrieve content from the memory to form the target word representation.

In the following sections, we will first introduce the memory architecture and the content based addressing. We then detail how exactly our model runs respectively in the knowledge accumulation and fast adaption phase.

3.1 Memory Addressing

M is a non-parametric key-value memory which stores a key-value pair (k_i, v_i) in each memory slot i. Inspired by (Kaiser et al., 2017), we keep an additional vector A tracking the age of slots. The initial age of all is zero. So the whole memory M looks like $(K_{m\times ks}, V_m, A_m)$ where m denotes memory size and ks denotes key vector size. Given a normalized query q, its nearest neighbor in M is defined as any of the keys that maximize the cosine similarity with q:

$$\mathrm{NN}(q, M) = \arg\max_i \; q \cdot K_i. \tag{1}$$

During training, a query to the memory M searches k nearest neighbors which is a natural extension to (1):

$$(n_1, ..., n_k) = \mathrm{NN_k}(q, M). \tag{2}$$

We take an average weighted by how the addressed memory slots match the query:

$$R_K, R_V = \sum_{k=i}^{K} \mathrm{softmax}\left(\frac{q \cdot M[n_k]}{\sqrt{d_k}}\right) \cdot M[n_k]. \tag{3}$$

This is actually a dot-product attention on the k nearest neighbors. R_K, R_V are the final output of the memory. Note that here we use softmax with temperature T:

$$softmax(a) = \frac{\frac{e^a}{T}}{\Sigma_{i=1}^{n}\frac{e^i}{T}}. \tag{4}$$

T is normally set to 1. Using a higher value for T produces a softer probability distribution. We set different temperatures in the knowledge accumulation and fast adaptation phase, which will be detailed in the following subsections.

3.2 Knowledge Accumulation

Given a target word embedding t_j with its context embedding c_j as input, we query the memory with c_j as (1)-(3) and retrieve R_k, R_V. The semantic relation between the current example and the retrieved content from memory is hoped to be consistent from context to target words, so we derive the following loss:

$$\mathcal{L}_m = \sum_{t_j, c_j \in D} log\sigma\left((R_K - R_V) \cdot (c_j - t_j)\right) \quad (5)$$

We also hope the target word representation fully incorporates context information and stay far from negative examples, so we also inherit the loss from (Mikolov et al., 2013):

$$\begin{aligned} \mathcal{L}_s = \sum_{t_j, c_j \in D} \#(t_j, c_j)\Big(log\sigma(t_j \cdot c_j) \\ + \sum_{i=1}^{k} \mathbb{E}_{p_i \sim P(t_j)}[log\sigma\ (t_j \cdot -p_i)]\Big), \end{aligned} \quad (6)$$

where D denotes the whole corpora, and $\#(t_j, c_j)$ means times the target and the context word cooccur. The word p_i is a negative sample sampled from the distribution $P(t_j)$, as Mikolov et al. (2013) do. We minimize the sum of $\mathcal{L}_m$ and $\mathcal{L}_s$:

$$\mathcal{L} = \mathcal{L}_m + \mathcal{L}_s \quad (7)$$

Memory Update. After each query, we update a memory slot according to how frequently the key is addressed and how useful the addressed value is. The update is done piecewise according to similarity between the addressed values $(V_{n_1}, ..., V_{n_k})$ and the target word. For all the addressed values V_{n_i} whose similarity to the target word is higher than the threshold β, we only update its corresponding key by taking a weighted average of the current key and the query:

$$K[n_i] \leftarrow \frac{q + K[n_i]}{||q + K[n_i]||}. \quad (8)$$

Otherwise, it means no addressed value correlates enough with the target, we then choose memory slots n' with maximum age and rewrite the stored items in it:

$$K[n'] \leftarrow q, V[n'] \leftarrow t_j. \quad (9)$$

The age of each updated slot will be reset to zero while all other non-updated slots get incremented by 1 in age. Memory updated in this way gradually accumulates similar contexts of a word into the same slot, which in another word, forms the prototype representation of a word.

3.3 Fast Adaptation

Now we show how to poll the memory to efficiently learn a new word representation from limited context. This is where the hypothesis constraint takes place. To be specific, given a new word embedding t_j^* to be learned and its context embedding c_j^*, we retrieve memory relevant to c_j^* as (2)-(3) and get R_K^*. Then we adapt context embedding with the retrieved memory to form the target word representation:

$$t_j^* = \alpha R_K^* + (1 - \alpha) c_j^*, \quad (10)$$

where α can be tuned a hyper-parameter or learned with regression models. Actually we also try to incorporate R_V^*, but the aggregated prototype R_K^* seems to continuously perform better.

We here pay additional attention to the softmax temperature T. T is emphasized since it conditions how the model "treats" the retrieved memory. Contexts are fairly limited in the few-shot case, so how the retrieved memory is treated crucially affects the quality of the learned representation. A higher temperature leads to a softer attention distribution , which means the model will be more likely to sample from all retrieved contents. A lower temperature makes the model focus more on the memory with highest similarity to the query. We predict a slightly higher temperature will generally be better in the fast adaptation phase. Since the HC theory points out that hypotheses are not in strict mutual-exclusions, they overlap with each other which corresponds to the higher-temperature condition. We will test this in the experiments.

4 Few-shot Word Similarity Tasks

We test the proposed method on two few-shot word similarity tasks. Fig.2 gives examples of the two tasks. In the following subsections we will introduce these datasets in detail and show the performance of tested methods on these tasks.

4.1 Tasks and Datasets

Definitional Nonce Task We evaluate on the Definitional Nonce dataset (Herbelot and Baroni,

Nonce Definition
Provided Context :
___ international inc is an american multinational conglomerate company that produces a variety of commercial and consumer products engineering services and aerospace systems for a wide variety of customers from private consumers to major corporations and governments
Ground Truth Word: Honeywell

Chimera-l2
Provided Context :
1. Canned sardines and ____ between two slices of whole meal bread and thinly spread Flora Original.
2. Erm, ____, low fat dairy products, incidents of heart disease for those who have an olive oil rich diet.

Probe Words: rhubarb, onion, pear, strawberry, limousine, cushion
Human Response: 2.57, 4.43 , 3.86, 3.71, 1.43, 2.14

Figure 2: Examples of the Nonce Definition and Chimera Task

2017) to simulate the process where a competent speaker learns a novel word from one informative sentence. 1000 words are included in the dataset as targets, with 700 for training and 300 for testing. Each target word corresponds to only one sentence extracted from its Wikipedia definition as context. All context sentences have been manually checked to be definitional enough to describe the corresponding target words. After tuning parameters on training data, the model is required to learn the target word representation with the provided context in test set. Learned representations are assessed by similarity to ground truth vectors produced in exposure to the whole corpora. We use the Reciprocal Rank (RR) of the ground vector in all nearest neighbors to the learnt representation for fair comparison of different methods, following Herbelot and Baroni (2017)'s settings. The mean value of RR over all test instances in the dataset is calculated as the final score.

Chimera Task Our second evaluation on the Chimera dataset (Lazaridou et al., 2017) means to simulate the case where a speaker learns the new word in a more casual multi-sentence context, not as highly informative as definitions in the Nonce dataset. There are 3 sub-tasks in Chimera:L2, L4 and L6, respectively providing 2, 4, 6 sentences as context to for each of the 330 instances in the dataset. The tested model needs to learn target word representation from the provided contexts. The similarity between learned embeddings and each of the probe words is measured and compared to human judgments by Spearman correlation. The final score is the average Spearman across all test pairs.

4.2 Baselines

Our model is compared to several baselines, including Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), SUM (Lazaridou et al., 2017) and N2V (Herbelot and Baroni, 2017). Glove and Word2Vec are representatives of traditional DSMs. With them we want to test how exactly traditional DSMs perform in the few shot representation learning without any additional mechanism for small data. SUM and N2V are proposed especially for learning on small corpus. They adapt Word2Vec's skip-gram structure for incremental learning and show improvements on the Chimera dataset. They partially match the HC theory which Mem2Vec is based on. Note that several rare word learning methods (Long et al., 2017; Xu et al., 2014; Lazaridou et al., 2013) that rely on auxiliary information don't apply with most of our task settings. In the Nonce and Chimera task, context for target word learning is strictly limited for fair comparison, so external knowledge is banned. And the target word, as showed in Fig.2, is just a slot which doesn't provide any morphological hints, so sub-word methods are also excluded.

Both the above baselines and the proposed Mem2Vec use a dump of Wikipedia to learn a fundamental semantic space. To be specific, N2V and SUM use embeddings pre-trained by Word2Vec, while Mem2Vec acquires prior-knowledge, all from that Wiki corpora. We calculate correlation with the similarity ratings in the MEN and SIMLEX dataset to verify if the pre-trained semantic space is ready for use.

Task / Model	Nonce		Chimera		
	MRR	Med. Rank	L2 ρ	L4 ρ	L6 ρ
Word2Vec	0.00007	111012	0.1459	0.2457	0.2498
GloVe	0.00008	108002	0.1402	0.2397	0.2533
SUM	0.03686	861	0.3176	0.3534	0.3880
N2V	0.04907	623	0.3120	0.3628	0.3790
Mem2Vec	**0.05416**	**518**	**0.3301**	**0.3717**	**0.3897**

Table 1: Results of Nonce Definition and Chimera task. MRR and Med Rank respectively denotes mean reciprocal rank and median rank of the ground truth word. ρ denotes precision.

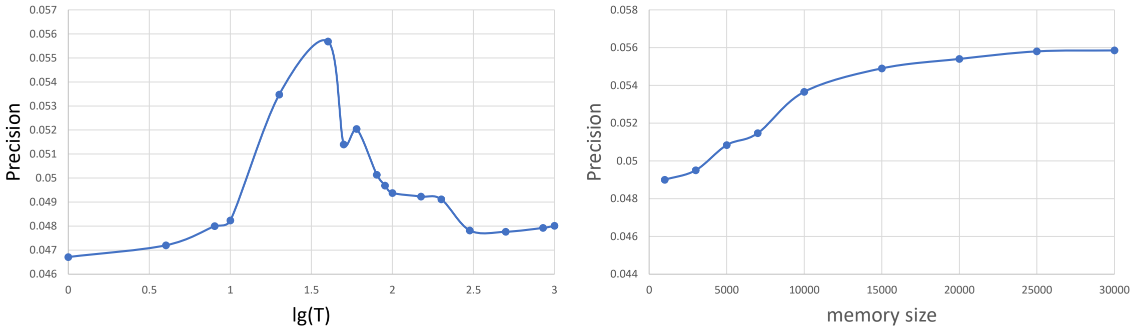

Figure 3: Performance of Nonce Definitional Task under different softmax temperature (left) and memory size (right). The left figure is in semilog coordinate.

4.3 Results

Nonce Definition Task We show the results of Nonce Definition Task in Table 1. Before analyzing the results we need to explain that MRR achieved by the tested models seems pretty low. This is no odd since matching the ground truth word in a vocabulary of 210,512 sets a potentially very large denominator in the reciprocal rank calculation. Our model achieves an MRR of 0.05416, which means the median rank of the true vector is 518 in the challenging two hundred thousand neighbors, surpassing all the baselines. N2V and SUM also deliver satisfactory performance with N2V working better. We are sorry to find that the naive Word2Vec and GloVe totally fail in the Nonce task, supporting the importance of adapting traditional DSMs for few-shot word representation learning.

Chimera Task The results on 3 chimera tasks are shown in Table 1, too. Mem2vec out-performs baselines in all the 3 context length settings. SUM performs steadily well from Nonce to the Chimera task, suggesting the effectiveness of constraining hypothesis space with contexts. But the continuous improvement of Mem2Vec over SUM confirms the advantage of our model, which incorporates “global” semantic information from the memory with the local contexts. N2V also works here but not as well as in the Nonce task, probably because the contexts in chimera are not as informative. Such performance drop may indicate N2V’s limited scalability to downstream NLP tasks since not all real world texts are as informative as in Nonce Definition Task. We will test this speculation with NER tasks in section 5.

4.4 Memory Parameter Analysis

We are interested to know how the two key parameters of memory, the softmax temperature T, and the memory size influence the quality of learned representations. We use the Nonce Definition task as the testbed. While studying the influence of one parameter, the other parameters are fixed. We run the model for 3 times with each candidate parameter and calculate the average precision as the final score.

Softmax Temperature Fig.3 (left) shows task performance under different softmax temperatures in semilog coordinate. We are a little bit surprised to find that it roughly fits a normal distribution and a mid-high temperature leads to best performance. A mid-high T means the model doesn’t give too large or too small weights to certain retrieved memory. This meets the HC theory about how humans weight the constrained hypothesis. We don't just trust a single hypothesis, nor do we treat all the hypotheses equally. The experiments shows similar principle also applies to our model.

Memory Size Fig.3 (right) shows task performance under different memory size. We find that increasing the memory size does lead to improved performance . But the improvements tend to be minor after the memory size is larger than 20,000. We owe it to the fact that we does not save specific examples, we accumulate similar contexts together in the memory to form prototypes. While we retrieve the memory, prototypes can be combined in different ways to represent multiple examples, thus a smaller memory can also work as well as the bigger one.

5 Extrinsic Tasks

We hope that the learned representations not only perform well on word similarity tasks but also apply to downstream NLP tasks. NER on domain specific datasets is an ideal benchmark. Named entities in these datasets are relatively low in frequency and not well covered by general domain corpus, tough for a traditional DSM to learn. Besides, while transferring from general domain corpus to a target small text, domain shift is a highly possible issue. We test if Mem2Vec could tackle the domain shift with a series of cross-domain sentiment analysis tasks.

5.1 Tasks and datasets

Domain Specific NER We use BioNLP11-species (Kim et al., 2011), AnatEMs (Pyysalo and Ananiadou, 2013) and NCBI-disease (Doğan et al., 2014) dataset, respectively from taxonomy, anatomy and pathology literatures. We train embeddings with tested methods to initialize the recognizer, whose performance then demonstrates whether the tested models learn representations well for rare words.

Cross Domain Sentiment Classification cross domain sentiment classification on Amazon Review dataset (Blitzer et al., 2007) is chosen as a benchmark. This dataset includes reviews from 4 product categories: books, DVDs, kitchens and electronics, suitable for the cross-domain setting. Using one as source domain and one as the target, we get 16 pairs for experiments. We train the classifier with source domain data and directly test it on the target domain, using the pre-trained embeddings as input feature. Note that through this task we also test how Mem2Vec performs when transferring from a small text, since in all the above experiments we learn prior knowledge from a large corpora.

5.2 Baselines

Except for the four baselines considered in word similarity tasks, we also compare with DAREP (Bollegala et al., 2015) and CRE (Yang et al., 2017) in the NER and sentiment classification tasks. They are both pivot-based methods for cross domain embedding learning which fare well in some downstream tasks. Besides we introduce SCL(Blitzer et al., 2006), a well-cited cross-domain sentiment analyser, as a baseline only for the sentiment classification task.

For NER, we use pre-trained embeddings by the tested methods as only input features for a LSTM-CRF recognition model (Lample et al., 2016). We simply mix the Wikipedia corpora with a dump of PubMed as our source corpora. Note that N2V and SUM can't be directly used to pre-train embeddings for downstream tasks since they focus on novel word learning. We thus explicitly divide words which occur less than 5 times as rare words while others as frequent words. N2V and SUM learn the frequent words with Word2Vec and learn the rare words in their own way. This setting also applies to the sentiment classification task.

For sentiment classification, we use a multi-layer perceptron (MLP) as the classifier, with one hidden layer of 400 nodes, ReLu activation and softmax output function.

5.3 Results

Named Entity Recognition Table 2 shows the results of domain specific named entity recognition. Used for pre-training embeddings, Mem2Vec achieves higher F1-score than all the baselines. It first surpasses CRE and DAREP that only bring slight improvements over Word2Vec. CRE and DAREP are both methods which relies on words with cooccurance patterns in source and target domain as the pivots for cross-domain transfer. This indicates the advantage of Mem2Vec over the traditional word frequency based methods in fast mapping cases where word cooccurrence pattern is not clear.

Our improvements over the N2V and SUM are more obvious than in the two word similarity

Task Model	AnatEM			BioNLP			NCBI		
	P	R	F1	P	R	F1	P	R	F1
Word2Vec	76.12	69.80	72.82	73.13	54.79	62.64	75.22	75.37	74.39
GloVe	75.83	67.04	71.14	72.58	53.35	61.50	75.76	72.33	74.01
N2V	76.81	66.8	71.46	73.91	54.21	62.54	72.45	74.37	73.30
SUM	77.06	69.01	72.81	74.36	58.58	62.25	74.89	74.02	74.45
DAREP	79.03	**67.95**	73.07	**77.18**	54.19	63.67	78.76	75.60	77.15
CRE	80.04	67.90	73.47	76.74	56.98	65.40	78.98	**76.63**	77.79
Mem2Vec	**81.23**	67.90	**73.96**	76.70	**57.81**	**65.92**	**79.56**	**76.63**	**78.06**

Table 2: Results of domain specific Named Entity Recognition. P, R, F1 respectively denotes precision, recall and F1 score

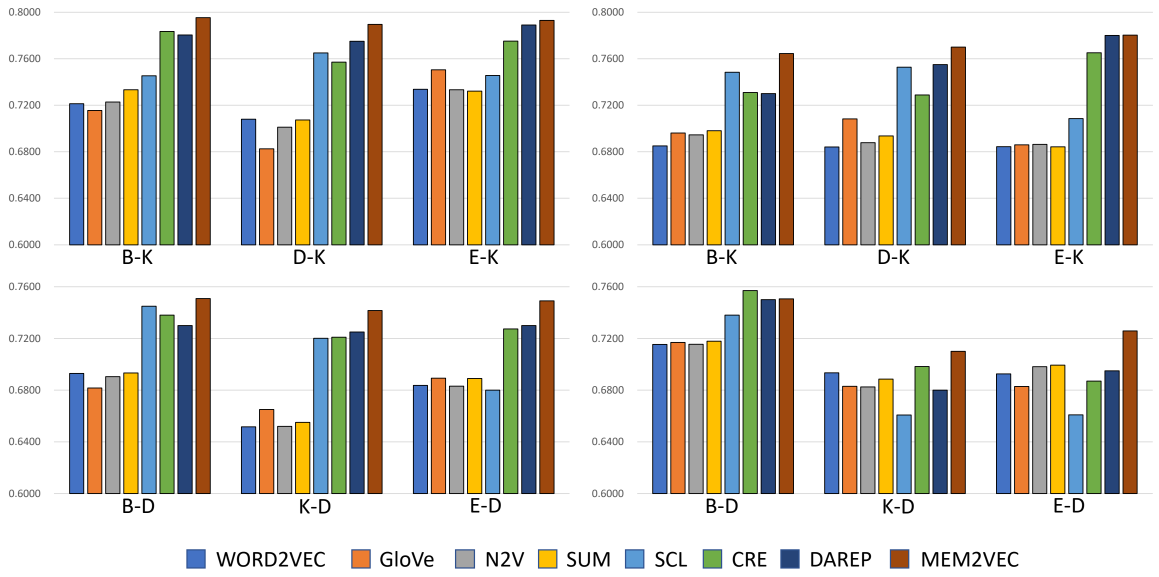

Figure 4: Results of cross domain sentiment classification on Amazon Review dataset. B denotes books, D for DVD, E for electronics, K for kitchen. B-K means B is the source domain and K is the target domain.

tasks. This again affirms our speculation that constraining hypothesis solely with the context is not enough. In the setting of NER, the context of one named entity is likely to be filled with other named entities which are also low in frequency. Directly summing the context as SUM does or taking risk to enlarge the window size as N2V may lead to over-fitting. While every training step of our method incorporates relative information from all the experienced examples stored in the memory, alleviating the danger of learn representations that over fits the local contexts.

In addition, it's worth noting that parameter tuning for N2V is no picnic. In our experiments, the original settings: high learning rate, large window size and short iteration span don't lead to satisfactory performance. More conservative parameter selection gets N2V back in track but departs from its fast mapping intention.

Sentiment Classification Fig.4 shows the results of Amazon Review sentiment classification. Mem2Vec delivers impressive performance, beating all the baselines in 10 of the total 12 pairs, including CRE and DAREP. This demonstrates the advantage of the memory as a transfer medium over the word- frequency based transfer of CRE and DAREP. But CRE and DAREP are still strong baselines in the cross domain task, surpassing SCL by a large margin. N2V and SUM are built for learning representation from small data, but they don't consider the possible domain discrepancy when using pre-trained embeddings on the target small text. So they don't bring much improvements over Word2Vec and GloVe. This also reminds us that to get the few-shot word representation learning methods in practical use, domain shift should be properly addressed.

6 Conclusion

We presented an integrated representation learning scheme which gradually learns from a big corpora and quickly adapt on tiny data. It accumulates knowledge with a long-term memory to adapt the representation learning of a novel word, in the few-shot learning case. Such adaptation means to constrain the "guess" of a DSM for the novel word according to the most relevant representation learning experience, inspired by hypothesis constraint theory for fast mapping. Experiments show the proposed method learns high quality representation from both highly informative and less definitional contexts in limited size. Pre-trained embeddings with our model also lead to improvements in Named Entity Recognition and sentiment analysis.

This work is our effort towards an ideal word representation learning scheme which learns from any amount of data. In the future work, we will explore more effective memory addressing and updating approaches to boost the few-shot representation learning. We believe not all examples are equally important and worth memorizing. Learning to memorize core examples should alleviate the data-hungry of representation learning methods.

Acknowledgments

The research work descried in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002103 and also supported by the Natural Science Foundation of China under Grant No. 61333018.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of international conference on learning representations (ICLR)*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. 2016. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the 2017 conference on empirical methods in natural language processing (EMNLP)*.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*.

Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events. *ICLR*.

Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart, and Sanjeev Arora. A la carte embedding: Cheap but effective induction of semantic feature vectors. In *Proceedings of ACL, year=2018, organization=Association for Computational Linguistics*.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. 2011. Overview of bionlp shared task 2011. In *Proceedings of the BioNLP shared task 2011 workshop*, pages 1–6. Association for Computational Linguistics.

Max Kisselew, Sebastian Padó, Alexis Palmer, and Jan Šnajder. 2015. Obtaining a better understanding of distributional models of german derivational morphology. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 58–63.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive science*, 41(S4):677–705.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1517–1526.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Teng Long, Emmanuel Bengio, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2017. World knowledge for reading comprehension: Rare entity prediction with hierarchical lstms using external descriptions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 825–834.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of international conference on learning representations (ICLR)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sampo Pyysalo and Sophia Ananiadou. 2013. Anatomical entity mention recognition at literature scale. *Bioinformatics*, 30(6):868–875.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pages 1842–1850.

Pablo Sprechmann, Siddhant M Jayakumar, Jack W Rae, Alexander Pritzel, Adrià Puigdomènech Badia, Benigno Uria, Oriol Vinyals, Demis Hassabis, Razvan Pascanu, and Charles Blundell. 2018. Memory-based parameter adaptation.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*.

Shaonan Wang, Jiajun Zhang, Nan Lin, and Chengqing Zong. 2017. Investigating inner properties of multimodal representation and semantic compositionality with brain-based componential semantics. In *AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 5964–5972.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2018. Learning multimodal word representation via dynamic fusion methods. pages 5973–5980.

Shaonan Wang and Chengqing Zong. 2017. Comparison study on critical components in composition model for phrase representation. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(3):16.

Dirk Weissenborn. 2017. Reading twice for natural language understanding. *CoRR*, abs/1706.02596.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1219–1228. ACM.

Fei Xu and Joshua B Tenenbaum. 2007. Word learning as bayesian inference. *Psychological review*, 114(2):245.

Wei Yang, Wei Lu, and Vincent Zheng. 2017. A simple regularization-based algorithm for learning cross-domain word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2898–2904.

Disambiguated skip-gram model

Karol Grzegorczyk[1,2] and **Marcin Kurdziel**[1]
[1]Department of Computer Science,
AGH University of Science and Technology, Kraków, Poland
[2]Allegro, Poznań, Poland
{kgr,kurdziel}@agh.edu.pl

Abstract

We present *disambiguated skip-gram*: a neural-probabilistic model for learning multi-sense distributed representations of words. Disambiguated skip-gram jointly estimates a skip-gram-like context word prediction model and a word sense disambiguation model. Unlike previous probabilistic models for learning multi-sense word embeddings, disambiguated skip-gram is end-to-end differentiable and can be interpreted as a simple feed-forward neural network. We also introduce an effective pruning strategy for the embeddings learned by disambiguated skip-gram. This allows us to control the granularity of representations learned by our model. In experimental evaluation disambiguated skip-gram improves state-of-the are results in several word sense induction benchmarks.

1 Introduction

Distributed representations of words find applications in a broad range of tasks, from natural language parsing (Socher et al., 2013) to image captioning (Karpathy and Fei-Fei, 2015). Their usefulness led to a renewed interest in word embedding algorithms. The most popular algorithms of this kind learn word vectors in an unsupervised manner, e.g., from word contexts (Mikolov et al., 2013a) or from statistics of word co-occurrence (Pennington et al., 2014). Unsupervised learning of word embeddings has a clear advantage: both general and domain-specific text corpora are available for a number of languages, which greatly reduces the cost of training. That said, unsupervised learning of word embeddings comes with its own challenges. One of the most important is word ambiguity: words in a natural language often have more than one meaning. The word *mouse*, for example, may mean a pointing device or an animal. Word embedding algorithms often do not recognize this language feature and estimate only one vector representation per word. This may lead to suboptimal word representations.

The main contribution of this work is *disambiguated skip-gram*: a neural-probabilistic model for learning distributed representations of words that capture word ambiguity. Disambiguated skip-gram builds upon the *skip-gram* model introduced by Mikolov et al. (2013a,b). Skip-gram constructs word embeddings via an auxiliary prediction task: given a word in a sentence, skip-gram attempts to predict the surrounding words. To this end, skip-gram defines a simple softmax model for the conditional probability of observing a context word c given the center word w:

$$p\left(c \mid w\right) = \frac{\mathrm{e}^{\mathbf{v}_w^{\mathrm{T}}\mathbf{u}_c}}{\sum_{c' \in D} \mathrm{e}^{\mathbf{v}_w^{\mathrm{T}}\mathbf{u}_{c'}}}, \qquad (1)$$

where D is the vocabulary. This log-bilinear model assigns two embedding vectors to every word $w \in D$: an *input embedding vector* $\mathbf{v}_w$ and an *output embedding vector* $\mathbf{u}_w$. Skip-gram defines the training objective for a single example as the log-probability: $\log p\left(c \mid w\right)$. By maximizing this objective, skip-gram estimates input and output vectors that reflect semantic relations between words that occur in similar contexts. The input vectors are then used as word embeddings.

The main idea behind disambiguated skip-gram is to jointly learn to disambiguate words and predict their contexts. We therefore extend skip-gram with a parametric word sense disambiguation model. This allows us to discover word senses in an unsupervised manner, while preserving the simplicity of the skip-gram approach. In particular, unlike previous probabilistic models for multi-sense word embeddings, disambiguated skip-gram can be seen as a simple feed-forward neural network amendable to end-to-end training with back-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1445–1454
Brussels, Belgium, October 31 - November 4, 2018.

propagation. Furthermore, disambiguated skip-gram admits an effective pruning strategy for the learned word sense embeddings. In particular, we control the granularity of the learned representations by penalizing the entropy of the probability distributions learned by the disambiguation model. We then marginalize word sense probabilities over the training examples and prune embeddings with low marginal probability.

We have carried out an extensive experimental evaluation of disambiguated skip-gram. Our results demonstrate that the multi-sense word embeddings learned by disambiguated skip-gram improve state-of-the-art results in the word sense induction task.

2 Disambiguated skip-gram model

Let $X = [w_1, \ldots, w_n]$, where each $w_i \in D$, be a sequence of words from a vocabulary D. By $\mathcal{C}_{w_i}$ we denote the context of the word w_i in X. For example, $\mathcal{C}_{w_i}$ can be a set of words that are no further than l positions from w_i and are in the same sentence as w_i. To simplify notation we will usually omit the sequence index i and write $w \in X$ for an element of the input sequence X and $\mathcal{C}_w$ for its context. We will also use notation $\mathbf{y}_w$ for a vector $\mathbf{y}$ (from some set of vectors indexed by the vocabulary words) corresponding to the word w. Note that in this case we disregard the position of w in X and use just the word as the index. In particular, if some word w occurs multiple times in X, all occurrences share a single vector $\mathbf{y}_w$.

Similarly to skip-gram, the disambiguated skip-gram model constructs word embeddings by learning to predict context words $c \in \mathcal{C}_w$ given the center word $w \in X$. However, disambiguated skip-gram explicitly accounts for word ambiguity. To this end, we represent each word $d \in D$ by a set of k *sense embedding vectors* $\mathbf{v}_{dz}$, indexed by $z \in \{1, \ldots k\}$, and an *output embedding vector* $\mathbf{u}_d$. We then parametrize the conditional probability $p\,(c \mid w, z = j)$ of observing a word c in the context of the word w in its j-th sense with a softmax model similar to the original skip-gram parametrization:

$$p\,(c \mid w, z = j) = \frac{\mathrm{e}^{\mathbf{v}_{wj}^{\mathrm{T}} \mathbf{u}_c}}{\sum_{c' \in D} \mathrm{e}^{\mathbf{v}_{wj}^{\mathrm{T}} \mathbf{u}_{c'}}}. \qquad (2)$$

Furthermore, in this work we assume that a sense of the word $w \in X$ can be guessed from its context $\mathcal{C}_w$[1], i.e. that:

$$z_w \sim p\,(z = j \mid w, \mathcal{C}_w)\,, \quad j = 1, \ldots, k, \qquad (3)$$

where z_w is an index of the sense of the word $w \in X$. Given this assumption, we parametrize the probability distribution for z_w using a softmax model similar to Eq. 2. That is, for each word $d \in D$ we introduce k *sense disambiguation vectors* $\mathbf{q}_{ds}$, $s \in \{1, \ldots, k\}$, and a *context embedding vector* $\mathbf{r}_d$. The conditional probability that the word $w \in X$ occurs in its j-th sense is then modelled as:

$$p\,(z = j \mid w, \mathcal{C}_w) = \frac{\mathrm{e}^{\mathbf{q}_{wj}^{\mathrm{T}} \bar{\mathbf{r}}_w}}{\sum_{s=1,\ldots,k} \mathrm{e}^{\mathbf{q}_{ws}^{\mathrm{T}} \bar{\mathbf{r}}_w}}, \qquad (4)$$

where $\bar{\mathbf{r}}_w$ is a vector representation of $\mathcal{C}_w$. We represent $\mathcal{C}_w$ by an average of context embedding vectors:

$$\bar{\mathbf{r}}_w = \frac{1}{\#\mathcal{C}_w} \sum_{c \in \mathcal{C}_w} \mathbf{r}_c. \qquad (5)$$

We can now define the training objective for a single word $w \in X$ as the expected negative log-likelihood of observing the context $\mathcal{C}_w$ under the distribution of senses of the center word w:

$$\mathcal{L}\,(\boldsymbol{\phi}, w) = \mathbb{E}_{z_w \sim p(z=j \mid w, \mathcal{C}_w)} \left[-\log \prod_{c \in \mathcal{C}_w} p\,(c \mid w, z = z_w) \right], \qquad (6)$$

with the parameters: $\boldsymbol{\phi} = \{\mathbf{v}_{dz}, \mathbf{u}_d, \mathbf{q}_{dz}, \mathbf{r}_d \mid d \in D, z = 1, \ldots, k\}$.

The objective in Eq. 6 is inconvenient for gradient-based optimization, because the expectation is taken with respect to a probability distribution that is a function of model parameters. In principle, we can estimate the gradient of this objective with the score function estimator (Glynn, 1990; Williams, 1992). Unfortunately, the score function estimator suffers from a high variance, even when used with a control variate. One can also derive a low-variance unbiased gradient estimator for certain probability distributions, by expressing the samples as a differentiable function of model parameters and a random variable from some independent fixed distribution (Kingma and Welling, 2013). This approach is not directly applicable to our case, because categorical distribution do not admit reparametrization with a differentiable function. That said, a simple and effective

---

[1] This assumption follows from an observation that two different meanings of a given word will often have vastly different contexts.

biased gradient estimator for an expectation with respect to a categorical distribution was recently proposed by Jang et al. (2016) and Maddison et al. (2016). The basic idea is to reparametrize the samples from the categorical distribution with the *Gumbel-Max trick* (Gumbel, 1954) and then approximate the non-differentiable max operator with a softmax function with temperature hyper-parameter. This can be seen as a reparametrization trick for a continuous relaxation to discrete samples from the categorical distribution.

In our case, the samples from $p(z = j \mid w, \mathcal{C}_w)$ take the form:

$$z_{wj} = \begin{cases} 1 & \text{if } j = \arg\max_{s}(\xi_s + \log p(z = s \mid w, \mathcal{C}_w)), \\ 0 & \text{otherwise,} \end{cases}$$

where ξ_s are i.i.d. samples from the standard Gumbel distribution $f(0,1)$. Note that the samples $\mathbf{z}_w = [z_{w1}, \ldots, z_{wk}]$ are now one-hot encoded. The continuous relaxation to $\mathbf{z}_w$ is:

$$\begin{aligned} \tilde{\mathbf{z}}_w &= [\tilde{z}_{wj}(\xi_1, \ldots, \xi_k, \mathcal{C}_w) \mid j = 1 \ldots k], \\ \tilde{z}_{wj}(\xi_1, \ldots, \xi_k, \mathcal{C}_w) &= \\ &\frac{\mathrm{e}^{[\xi_j + \log p(z=j \mid w, \mathcal{C}_w)]/\tau}}{\sum_{s=1,\ldots,k} \mathrm{e}^{[\xi_s + \log p(z=s \mid w, \mathcal{C}_w)]/\tau}}, \end{aligned} \quad (7)$$

where τ is the temperature hyper-parameter. When $\tau \to 0$, we recover the samples from $p(z = j \mid w, \mathcal{C}_w)$. However, for $\tau > 0$ the samples $\tilde{\mathbf{z}}_w$ are no longer discrete. In this case we consider a relaxed sense embedding vector:

$$\tilde{\mathbf{v}}_w = \sum_{j=1,\ldots,k} \tilde{z}_{wj}(\xi_1, \ldots, \xi_k, \mathcal{C}_w)\, \mathbf{v}_{wj} \quad (8)$$

and model the conditional probability of observing a word c in the context of the word w as:

$$p(c \mid w, \tilde{\mathbf{v}}_w) = \frac{\mathrm{e}^{\tilde{\mathbf{v}}_w^{\mathrm{T}} \mathbf{u}_c}}{\sum_{c' \in D} \mathrm{e}^{\tilde{\mathbf{v}}_w^{\mathrm{T}} \mathbf{u}_{c'}}}. \quad (9)$$

Our training objective for a single word $w \in X$ then takes the form:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\phi}, w) = &\mathbb{E}_{\xi_1, \ldots, \xi_k \sim f(0,1)} \\ &\left[-\log \prod_{c \in \mathcal{C}_w} p(c \mid w, \tilde{\mathbf{v}}_w) \right]. \end{aligned} \quad (10)$$

The relaxed objective in Eq. 10 is tractable and differentiable with respect to the parameters $\boldsymbol{\phi}$.

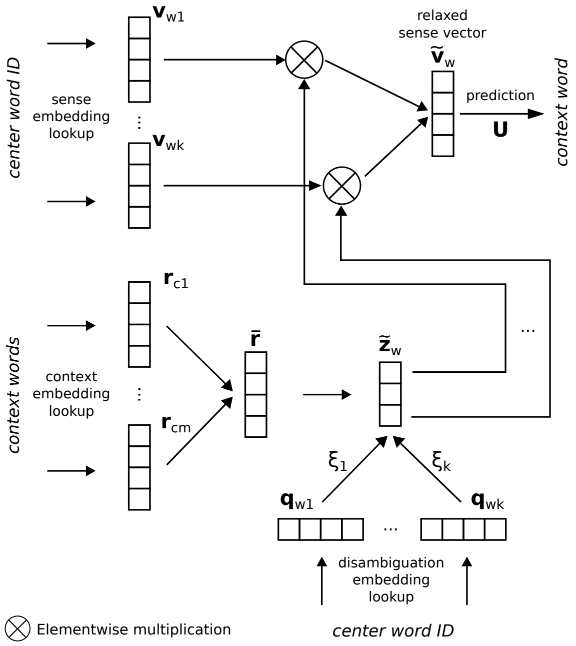

Figure 1: Disambiguated skip-gram model.

When $\tau \to 0$, it becomes equivalent to the objective in Eq. 6. In practice, we approximate the expectation in Eq. 10 with a one-sample Monte Carlo estimator. In these settings disambiguated skip-gram can be seen as a simple feed-forward neural network pictured in Fig. 1. During training, the network jointly estimates a sense disambiguation model (Eq. 4) and a context word prediction model (Eq. 2), which we use to construct multi-sense word embeddings.

2.1 Pruning word senses

The disambiguated skip-gram model is parametric, i.e. it allocates a fixed number of sense embedding vectors to each word, even though different words have different number of discernible senses. That said, we can prune the sense embedding vectors by considering their probabilities according to the learned disambiguation model. In particular, after training we estimate the marginal probability:

$$p(d, j) = \frac{1}{m_d} \sum_{\substack{w_i \in X, \\ w_i = d}} p(z = j \mid w_i, \mathcal{C}_{w_i}) \quad (11)$$

for each word $d \in D$ and each sense index $j \in \{1, \ldots, k\}$. We then prune sense embedding vectors with low marginal probability, e.g., $p(d, j) < 0.05$. The normalizing factor m_d in Eq. 11 is the number of occurrences of the word d in X.

The above pruning technique can be extended to allow for an explicit control over the granularity of the learned sense representations. To this end, we use an entropy regularization term similar to the one studied by Pereyra et al. (2017) in classification networks. In disambiguated skip-gram the granularity of the learned representations is controlled by the disambiguation model (Eq. 4). Therefore, we extend the objective of our model (Eq. 10) by adding to it an entropy S of the probability distribution $p\left(z=j \mid w, \mathcal{C}_w\right)$:

$$\mathcal{L}_r\left(\boldsymbol{\phi}, w\right)=\mathcal{L}\left(\boldsymbol{\phi}, w\right)+\gamma S\left(\boldsymbol{\phi}, w\right), \qquad (12)$$

where:

$$S\left(\boldsymbol{\phi}, w\right)= -\sum_{j=1}^{k} p\left(z=j \mid w, \mathcal{C}_w\right) \log p\left(z=j \mid w, \mathcal{C}_w\right).$$

The hyper-parameter γ, which we further call *entropy cost*, controls the strength of the regularization and, in turn, the granularity of the learned sense representations. In particular, $\gamma > 0$ encourages the model to learn more coarse-grained sense representations, whereas $\gamma < 0$ increases the granularity of the learned senses.

3 Related work

Algorithms for learning distributed multi-sense representations of words have been a focus of several recent works. Initial approaches to this task relied on clustering word contexts. One of the first algorithms of this kind was proposed by Huang et al. (2012). They learn multi-sense word representations in three steps. First, they estimate vector representations of words using a feed-forward neural language model. Next, they calculate average word vector for each context in the training corpus, cluster these context representations and relabel each word in the corpus to a word sense represented by the nearest cluster. Finally, they train the language model on the relabelled corpus and obtain vector representations for word senses.

Neelakantan et al. (2014) proposed the Multi-Sense Skip-gram (MSSG) model, that jointly learns context cluster prototypes and word sense embeddings. Their model extends skip-gram by maintaining context clusters for every word in the vocabulary. Given a training example with a center word w and its context representation c, they infer the word sense for w by a hard assignment of the context representation c to the cluster with the nearest centroid. Afterwards, they perform a skip-gram-like update on the vector representation of the selected word sense and the output vectors of the context words. Neelakantan et al. also proposed a non-parametric version of MSSG (NP-MSSG), in which the number of clusters, and in turn the number of word senses, increases during training. They use a simple heuristic to determine the number of word senses: NP-MSSG allocates a new sense for the center word w when the similarity between the context representation c and the nearest cluster centroid falls below some predefined threshold. Neelakantan et al. demonstrated that MSSG and NP-MSSG outperform the Huang et al. algorithm on a contextual word similarity task.

A disadvantage of the Huang et al. and Neelakantan et al. algorithms is that they do not follow a principled statistical approach, but instead rely on hard clustering of context vectors. This has been addressed in more recent algorithms, which learn multi-sense word representations in a probabilistic framework. Concurrent to Neelakantan et al. work, Tian et al. (2014) proposed a probabilistic Multi-Prototype Skip-Gram (MPSG) model. MPSG extends the skip-gram model by adding to each position in the input text a latent variable that encodes the index of the word sense at that position. Furthermore, for each word in the vocabulary MPSG maintains a fixed number of sense embedding vectors and a single output vector. These parameters define a softmax model for the conditional probability of observing a context word given the center word and the latent sense index. Finally, MPSG models the conditional probability of observing a context word given the center word with a mixture model whose components correspond to the senses of the center word. Tian et al. derived an expectation maximization algorithm for estimation of softmax parameters and prior sense probabilities in their model. MPSG was evaluated in a contextual word similarity task, where its performance was similar to that of the Huang et al. algorithm.

Bartunov et al. (2016) proposed the AdaGram model, which can be seen as a non-parametric version of MPSG. Similarly to MPSG, AdaGram introduces latent variables for word sense indexes in the input text. However, unlike MPSG, AdaGram does not assume a fixed number of word senses.

Instead, it defines the prior over word senses via a Dirichlet process. As a result, AdaGram automatically learns the number of senses for all words in the vocabulary. Unfortunately, defining the prior over word senses via a Dirichlet process gives an intractable model likelihood. Bartunov et al. therefore optimize variational lower bound of the AdaGram model likelihood using a stochastic variational inference algorithm. Bartunov et al. evaluated AdaGram performance on several word sense induction benchmarks. They demonstrated that AdaGram consistently outperforms MSSG, NP-MSSG and MPSG models in these benchmarks. AdaGram has been recently extended to handle parallel multilingual text corpora (Upadhyay et al., 2017).

For the prediction of context words (Eq. 2) disambiguated skip-gram adopts the softmax model used in MPSG. However, in contrast to the previous works, disambiguated skip-gram learns a parametric model for the conditional probability distribution over senses of the center word given the context words (Eq. 4). This allows us to define the training objective for disambiguated skip-gram as the expected negative log-likelihood of observing the context words under the distribution of senses of the center word. We use a biased low-variance gradient estimator for this objective, which enables stable end-to-end training with backpropagation.

The main goal of the AdaGram model is to automatically discover the number of word senses for the vocabulary words. This does not mean, however, that the number of senses learned by AdaGram is independent of model hyper-parameters. On the contrary, the number of senses learned by AdaGram is directly controlled by the hyper parameter α in the Dirichlet process used to define the prior over word meanings (Bartunov et al., 2016). Disambiguated skip-gram controls the number of learned senses by penalizing the entropy of the conditional probability distribution in the sense disambiguation model (Eq. 12). The entropy cost γ in this approach performs a function similar to the hyper-parameter α in AdaGram.

In addition to the works discussed above, word ambiguity was also modelled using topic models (Liu et al., 2015), large bi-directional language models (Peters et al., 2018) or subword information (Athiwaratkun et al., 2018). Also, Li and Jurafsky (2015) evaluated multi-sense embeddings in several downstream tasks. They found that multi-sense embeddings improve performance in tasks such as POS tagging or identification of semantic relations. They also identify downstream tasks which do not benefit from sense disambiguation. In sentiment analysis, for example, word sentiment usually does not depend on the inferred sense.

4 Experiments

We conducted a number of experiments to evaluate the quality of multi-sense word embeddings learned by disambiguated skip-gram. This section reports results from our evaluation. First, we report qualitative results from our model for several polysemous words. We then compare the performance of disambiguated skip-gram with several competing algorithms on a set of word sense induction tasks. Finally, we evaluate the effect of the entropy cost on the learned representations.

It is worth noting that the quality of multi-sense word embeddings was formerly assessed in contextual word similarity experiments. However, Bartunov et al. (2016) demonstrated that contextual word similarity experiments do not reflect the quality of multi-sense representations. In particular, the best performance in contextual word similarity task is often achieved by the baseline skip-gram model, which does not recognize word senses. This can be attributed to the fact that skip-gram objective directly optimizes similarity of vector representations of words that appear in similar contexts. Multi-sense models, on the other hand, solve a harder task: they disambiguate words in contexts and then model the similarities between the discovered senses. Bartunov et al. focus, therefore, on the performance of multi-sense embeddings in the word sense induction task. We adopt their evaluation methodology in this work.

We trained our disambiguated skip-gram models on the Westbury Lab Wikipedia corpus (Shaoul and Westbury, 2010). We optimized the models using mini-batch stochastic gradient descend with momentum.

4.1 Qualitative results

We begin our evaluation by presenting senses discovered by disambiguated skip-gram for several ambiguous words. For the demonstration we trained four 300-dimensional models with three sense embedding vectors allocated to each word

Word	$\gamma = 0.0$		$\gamma = 0.25$		$\gamma = 0.5$	
	p	Nearest neighbors	p	Nearest neighbors	p	Nearest neighbors
fox	0.52	nbc cbs network syndication espn	0.60	nbc cbs abc syndication network	0.68	cbs nbc abc cable colmes
	0.25	miller allen plummer crowe buck	0.22	miller allen terry russell soper	0.18	allen russell miller turner berry
	0.24	badger wolf coyote weasel marten	0.18	badger squirrel weasel raccoon marten	0.14	badger marten raccoon beaver mink
net	0.41	ebitda earnings annualized taxable depreciation	0.43	ebitda annualized jpy deadweight gni	0.77	ebitda deadweight isk annualized deducting
	0.32	trawl streamline maximising minimises counteracts	0.33	trawl minimises maximising streamlines stickiness	0.23	crossbar puck lob header dribbled
	0.26	crossbar puck lob sliothar offside	0.24	crossbar puck lob offside dribbled	0	
rock	0.41	band indie punk alternative supergroup	0.7	alternative glam progressive indie psychedelic	0.72	alternative punk indie glam progressive
	0.34	punk rockabilly pop psychedelia funk	0.17	boulder basalt outcrop quartzite cliffs	0.17	basalt boulders quartzite cliff outcropping
	0.26	boulder quartzite granite sandstone basalt	0.13	granite bluff pine pigeon ledge	0.11	pine bluff eagle pigeon turtle
plant	0.45	flowering perennial shrub grass fungus	0.46	flowering grasses shrub fungus herbaceous	0.48	flowering shrub grasses herbaceous fungus
	0.38	refinery smelter petrochemical processing factory	0.45	refinery factory megawatt smelter cogeneration	0.48	refinery factory smelter megawatt sellafield
	0.18	factory botanical laboratory farm nurseryman	0.08	weed planted shed grinder laboratory	0.04	
mouse	0.47	mickey rabbit goofy cat porky	0.5	rat mice rodent mus elegans	0.51	mice rodent rat mus elegans
	0.35	cursor joystick trackball touchpad touchscreen	0.49	rabbit goofy cat porky tigger	0.49	rabbit goofy porky tigger tweety
	0.19	rodent vole shrew pygmy rat	0.01		0	
apple	0.46	macintosh imac iigs iie iic	0.64	macintosh iigs imac iie iic	0.95	macintosh blackberry iigs imac apricot
	0.28	wozniak macworld macintosh ipod sculley	0.25	wozniak blackberry tomato potato popcorn	0.04	
	0.26	strawberry peach raspberry blueberry plum	0.11	peach pecan persimmon prune blueberry	0.01	
table	0.40	sortable column lookup hashed tray	0.48	sortable column lookup tray hashed	0.66	sortable tray column chairs buckets
	0.38	foosball carom languishing pool slipping	0.36	foosball ept languishing pool leaderboard	0.31	standings ept foosball leaderboard ittf
	0.22	sortable list alphabetical descending brackets	0.16	sortable descending list alphabetically please	0.03	

Table 1: Nearest neighbors and marginal probabilities p of word sense embedding vectors discovered by the disambiguated skip-gram model for several ambiguous words. Sense embedding vectors with a marginal probability $p < 0.05$ are pruned from the learned model.

and the entropy cost γ ranging from 0.0 to 0.5. For each of the evaluated words sense embeddings we calculated the cosine similarity to the remaining words and selected 5 nearest neighbours. The results are reported in Tab. 1.

Disambiguated skip-gram discovered main meanings of our test words. For example, the meanings discovered for the word *fox* correspond to a broadcasting company, an animal and a family name. The meanings discovered for the word *mouse* correspond to a cartoon character, a computer mouse and an animal.

Results in Tab. 1 also demonstrate that disambiguated skip-gram will often expose an internal structure in a word meaning, if that meaning appears in different contexts. For example, disambiguated skip-gram learned two embeddings for the word *plant* corresponding to its *factory* meaning: one related to heavy industry and one related to a farm or a plant nursery. This is a consequence of the fact that disambiguated skip-gram discovers word senses using only the information about the contexts in which these words occur. In particular, it does not employ any supervision from an external knowledge base. Bartunov et al. (2016) refer to a related phenomenon in the AdaGram embeddings as the semantic resolution of the model.

4.2 Word-sense induction experiments

To compare disambiguated skip-gram with state-of-the-art competing algorithms we assessed its performance in a set of word sense induction tasks. In this evaluation we follow the experimental setup from (Bartunov et al., 2016), allowing for direct comparison with the results reported therein. In particular, we evaluated disambiguated skip-gram on the datasets from the SemEval-2007 Task 2 competition (SE-2007), SemEval-2010 Task 14 competition (SE-2010), SemEval-2013 Task 13 competition (SE-2013) and the Wikipedia Word-sense Induction (WWSI) dataset introduced by Bartunov et al. The test datasets consist of between 4664 and 36354 examples. Each example provides a ground truth sense of a center word and the context in which this sense appeared. The goal is to recognize the sense of the center word given the context. We use the preprocessed versions of SE-2007, SE-2010 and WWSI datasets made available by Bartunov et al. For the SemEval-2013 Task 13 we use the original competition dataset (Jurgens and Klapaftis, 2013) and follow the preprocessing steps reported in (Bartunov et al., 2016).

We use a simple procedure for resolving word senses. That is, we average all sense embedding vectors of all context words and select the sense z_w of the center word w whose embedding vector is most similar to the average vector:

$$z_w = \arg\max_j \cos\left(\mathbf{v}_{wj}, \bar{\mathbf{c}}_w\right), \tag{13}$$

where:

$$\bar{\mathbf{c}}_w = \left(k \cdot \#\mathcal{C}_w\right)^{-1} \sum_{c \in \mathcal{C}_w} \sum_{s=1}^{k} \mathbf{v}_{cs}. \tag{14}$$

The intuition behind this procedure is that we expect the average to preserve a shared component in the embedding vectors, namely embeddings for senses related to the sense of the center word, and cancel out embeddings of unrelated senses. In addition to averaging sense embedding vectors we also experimented with averaging output vectors of context words. However, this approach usually gave slightly worse results.

Following Bartunov et al. (2016), we use adjusted rand index (Hubert and Arabie, 1985) to compare ground truth senses for a given word with the senses inferred from disambiguated skip-gram embeddings. The final performance on a benchmark task is the average of adjusted rand index values over all test words in the task.

For this evaluation we trained a 300-dimensional disambiguated skip-gram model with 5 sense embedding vectors allocated to each word and no entropy cost ($\gamma = 0.0$). The comparison between our model and MSSG, NP-MSSG, MPSG and AdaGram is reported in Tab. 2. The results for MSSG, NP-MSSG, MPSG and AdaGram are taken from Bartunov et al. (2016). Multi-sense embedding vectors learned by disambiguated skip-gram outperform baseline methods on the SE-2007, SE-2010 and WWSI benchmarks, and achieve the second best result on the SE-2013 benchmark.

	SE-2007	SE-2010	SE-2013	WWSI
MSSG	0.048	0.085	0.033	0.194
NP-MSSG	0.033	0.044	0.033	0.110
MPSG	0.044	0.077	0.014	0.160
AdaGram	0.069	0.097	**0.061**	0.286
Disamb. skip-gram	**0.077**	**0.117**	0.045	**0.292**

Table 2: Performance of multi-sense word embedding methods in word sense induction tasks. The reported performance metric is the adjusted rand index averaged over all test words in the benchmark task. Results for all models except the disambiguated skip-gram (Disamb. skip-gram) are taken from (Bartunov et al., 2016).

4.3 Effect of the entropy cost

Results reported in Tab. 1 demonstrate that the entropy cost γ (Eq. 12) indeed allows for pruning senses learned by disambiguated skip-gram. In particular, when the entropy cost increases, disambiguated skip-gram allocates more of the marginal probability mass (Eq. 11) to the frequent meanings of the modelled words and, in effect, learns coarser representations.

For a quantitative evaluation of the effect of entropy cost on the learned representations we trained 50-dimensional disambiguated skip-gram models with γ ranging from 0.0 to 1.0. All models allocate 5 sense embedding vectors to every word in the vocabulary. In Tab. 3 we report an average number of senses per word with marginal probability $p \geq 0.05$, depending on the value of the entropy cost. In Fig. 2 we also report histograms of marginal probabilities for selected entropy cost values.

Entropy cost	0.0	0.1	0.25	0.5	0.75	1.0
Avg. sense num.	4.7	4.3	3.7	3.2	2.8	2.5

Table 3: Average number of senses per word with marginal probability $p \geq 0.05$, learned by disambiguated skip-gram models with different values of the entropy cost.

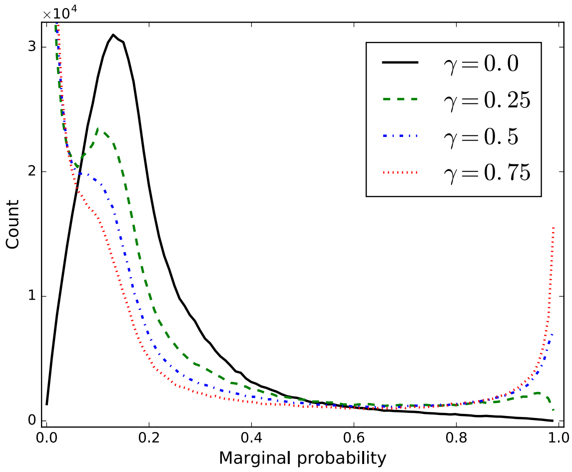

Figure 2: Histograms of marginal probabilities of word senses learned by disambiguated skip-gram models with different values of the entropy cost.

Histograms in Fig. 2 confirm our observation from the qualitative evaluation: when the entropy cost increases, disambiguated skip-gram learns more peaked distributions for the conditional sense probability $p(z = j \mid w, \mathcal{C}_w)$. This translates to coarser sense representations. In particular, the model with no entropy cost learned an average of 4.7 senses per word with marginal probability $p \geq 0.05$ (Tab. 3). This number decreases with an increasing entropy cost, reaching an average of 2.5 senses per word for $\gamma = 1.0$.

We also evaluated 50- and 300-dimensional models with different entropy costs in the word sense induction tasks. In each case we pruned senses with marginal probability $p < 0.05$. Results from this evaluation (Tab. 4) indicate that the desired granularity of the learned sense representations depends on the underlying task. In the WWSI benchmark the best performing models had no entropy cost, while in the SemEval tasks small entropy cost usually improved results. The results agree for both model dimensionalities.

Dim.	Entropy Cost	SE 2007	SE 2010	SE 2013	WWSI
50	No	0.064	0.107	0.040	**0.304**
	0.25	**0.083**	**0.116**	0.043	0.244
	0.5	0.064	0.091	**0.045**	0.182
300	No	0.077	**0.117**	0.045	**0.292**
	0.25	**0.079**	0.113	0.045	0.259
	0.5	0.065	0.091	**0.049**	0.183

Table 4: Performance of disambiguated skip-gram models with different entropy costs in the word sense induction tasks. The reported performance metric is the adjusted rand index averaged over all test words in the benchmark task.

5 Conclusions

In this work we developed disambiguated skip-gram: a novel neural-probabilistic model for learning multi-sense distributed representations of words. Unlike previous probabilistic models for multi-sense word embeddings, disambiguated skip-gram is a simple feed-forward neural network and can be trained end-to-end with backpropagation. In experimental evaluation disambiguated skip-gram improved over the state-of-the-art results in three out of four benchmark datasets and ranked second the fourth.

Disambiguated skip-gram optimizes expected log-likelihood of the context prediction model under the distribution of word senses parametrized by the disambiguation model. We choose to optimize this objective with a biased but low-variance gradient estimator. However, parallel to this work there has been a significant progress in gradient-based training of models with discrete latent variables. Specifically, Tucker et al. (2017) proposed an unbiased low-variance gradient estimator, called REBAR, that is applicable to models with categorical latent variables. REBAR may allow to efficiently optimize the original disambiguated skip-gram objective (Eq. 6), instead of the relaxed objective (Eq. 10). This may further improve the quality of embeddings learned with our approach.

Acknowledgments

This research was supported by National Science Centre, Poland grant no. 2013/09/B/ST6/01549 “Interactive Visual Text Analytics (IVTA): Development of novel, user-driven text mining and visualization methods for large text corpora exploration”. The paper was partially financed by AGH University of Science and Technology Statutory Fund. The paper was also partially financed by Allegro. Experiments for this work were supported by the PL-Grid infrastructure and by the “HPC Infrastructure for Grand Challenges of Science and Engineering” project co-financed by the European Regional Development Fund under the Innovative Economy Operational Programme. Last but not least, we would like to thank Professor Witold Dzwinel for overall guidance and support.

References

Ben Athiwaratkun, Andrew Wilson, and Anima Anandkumar. 2018. Probabilistic FastText for multi-sense word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11. Association for Computational Linguistics.

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 130–138.

Peter W Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84.

Emil Julius Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*. 33. US Govt. Print. Office.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2(1):193–218.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*.

David Jurgens and Ioannis P Klapaftis. 2013. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Proceedings of the 7th international workshop on semantic evaluation*, pages 290–299. Association for Computational Linguistics.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3128–3137.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1722–1732. Association for Computational Linguistics.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2418–2424.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 14, pages 1532–1543. Association for Computational Linguistics.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke

Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Cyrus Shaoul and Chris Westbury. 2010. The Westbury lab Wikipedia corpus.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 455–465.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 151–160. Dublin City University and Association for Computational Linguistics.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. 2017. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems 30*, pages 2624–2633.

Shyam Upadhyay, Kai-Wei Chang, Matt Taddy, Adam Kalai, and James Zou. 2017. Beyond bilingual: Multi-sense word embeddings using multilingual context. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 101–110. Association for Computational Linguistics.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Picking Apart Story Salads

Su Wang[1,2] **Eric Holgate**[1] **Greg Durrett**[3] **Katrin Erk**[1]
[1]Department of Linguistics
[2]Department of Statistics and Data Science
[3]Department of Computer Science
The University of Texas at Austin
shrekwang@utexas.edu holgate@utexas.edu
gdurrett@cs.utexas.edu katrin.erk@mail.utexas.edu

Abstract

During natural disasters and conflicts, information about what happened is often confusing, messy, and distributed across many sources. We would like to be able to automatically identify relevant information and assemble it into coherent narratives of what happened. To make this task accessible to neural models, we introduce *Story Salads*, mixtures of multiple documents that can be generated at scale. By exploiting the Wikipedia hierarchy, we can generate salads that exhibit challenging inference problems. Story salads give rise to a novel, challenging clustering task, where the objective is to group sentences from the same narratives. We demonstrate that simple bag-of-words similarity clustering falls short on this task and that it is necessary to take into account global context and coherence.

1 Introduction

When a natural disaster strikes or a conflict arises, it is often hard to determine what happened. Information is messy and confusing, spread out over many messages, buried in irrelevant text, and even conflicting. For example, when flight MH-17 crashed in Ukraine in 2014, there were initially many theories of what happened, including a missile strike initiated by Russia-affiliated militants, a missile strike by the Ukrainian military, and a terrorist attack. There was no single coherent interpretation of what happened, but multiple, messy narratives, a *story salad*. We would like to be able to automatically identify relevant information and assemble it into coherent narratives of what happened. This task is also the subject of an upcoming task at the Text Analysis Conference.[1]

Picking apart a story salad is a hard task that could in principle make use of arbitrary amounts of inference. But it is also a task in which coherence judgments could play a large role, the simplest being topical coherence, but also narrative coherence (Chambers and Jurafsky, 2008, 2009; Pichotta and Mooney, 2016; Mostafazadeh et al., 2017), overall textual coherence (Barzilay and Lapata, 2008; Logeswaran et al., 2018), and coherence in the description of entities. This makes it an attractive task for neural models.

(A) Some of the prisoners were survivors of the Battle of Qala-i-Jangi in Mazar-i-Sharif. (A) Chechnya came under the influence of warlords. (B) The U.S. invaded Afghanistan the same year when several Taliban prisoners were shot. (A) Russian federal troops entered Chechnya and ended its independence. (A) The Russian casualties included at least two commandos killed and 11 wounded. (B) The dead were buried in the same grave under the authority of Commander Kamal.

Figure 1: A story salad involving two articles, about a Russian military operation in Chechnya (A) and about a U.S. operation in Afghanistan (B). These two articles are topically similar but their mixture can still be disentangled based on narrative coherence.

To make the task accessible to neural models, we propose a simple method for creating simulated story salad data at scale: we mix together sentences from different documents. Figure 1 shows an example mixture of two articles from Wikipedia, one on the Russia-Chechnya conflict and one on a conflict between the U.S. and Afghanistan. By controlling how similar the source documents are, we can flexibly adjust the difficulty of the task. In particular, as we show below, we can generate data that exhibits challenging inference problems by exploiting the Wikipedia category structure.[2] While this data is still simpler than story salads arising naturally, it approximates the task, is sufficiently challenging for modeling, and can be generated in large amounts.[3]

---

[1] https://tac.nist.gov/2018/SM-KBP/index.html

[2] In particular, while we do not focus on creating mixtures with conflicting information, it can often be found in mixtures created based on Wikipedia categories, as shown in Figure 2.

[3] The story salad data is available at http://www.katrinerk.com/home/software-and-data/picking-apart-story-salads-1. The Wikipedia

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1455–1465
Brussels, Belgium, October 31 - November 4, 2018.

We explore some initial models for our Story Salad task. As the aim of the task is to group story pieces into stories, we start with straightforward clustering based on topic similarity. But topic similarity is clearly not enough to group the right pieces together. For example, the two articles in Figure 1 are both about armed conflicts, but the Russia-Chechnya sentences in the example form a group *in contrast to* the U.S.-Afghanistan sentences. To model this, we learn sentence embeddings adapted to the clustering task and with access to global information about the salad at hand. We also test an extension where to decide whether to group two sentences together, the model mutually attends to the sentences during encoding in order to better focus on the commonalities and differences of these two sentences. Both extensions lead to better models (6-13% improvement in accuracy with a model incorporating both), confirming that the task requires more than just general topical similarity. But there is much room for improvement, in particular on salads generated to be more difficult, where performance is around 15 points lower than on arbitrary mixtures.

2 Related Work

Building on early work in *script learning* (Schank and Abelson, 1977), Chambers and Jurafsky (2008) introduce *narrative schema* and propose the "narrative cloze" task where the modeling objective is to predict the event happening next. The topic has since seen many extensions and variants coupled with increasingly sophisticated models (Chambers and Jurafsky, 2009) including neural networks (Granroth-Wilding and Clark, 2016; Pichotta and Mooney, 2016; Mostafazadeh et al., 2017). This line of work is related to story salads in that our aim of separating entangled narratives in a document mixture also leverages within-narrative coherence. Our work, however, is very different from narrative cloze: (i) we group sentences/events rather than predicting what happens next; (ii) crucially, the narrative coherence in story salads is *in context*, in that a narrative clustering is only meaningful with respect to a particular document mixture (see Section 5, 6), while in narrative cloze the next event is predicted on a "global" level.[4]

Working with labeled story salad examples, we draw inspiration from previous work on supervised clustering (Bilenko et al., 2004; Finley and Joachim, 2005). We also take advantage of the recent success of deep learning in leveraging a continuous semantic space (Pennington et al., 2014; Kiros et al., 2015; Mekala et al., 2017; Wieting and Gimpel, 2017; Wieting et al., 2017) for word/sentence/event encoding; neural components for enhanced supervised clustering (Bilenko et al., 2004), in particular LSTMs (Hochreiter and Schmidhuber, 1997; Dai and Le, 2015), CNNs (Kim, 2014; Conneau et al., 2017), and attention mechanisms (Bahdanau et al., 2015; Hermann et al., 2015; Lin et al., 2017). By exploring our ability to pick apart story salads with these state-of-the-art NLP modeling tools, we attempt to (i) show the value of the story salad task as a new NLP task that warrants extensive research; (ii) understand the nature of the task and the challenges it sets forth for NLP research in general.

The task of picking apart story salads is related to the task of conversation disentanglement (Elsner and Charniak, 2008; Wang and Oard, 2009; Jiang et al., 2018), which is a clustering task of dividing a transcript into a set of distinct conversations. While superficially similar to our Story Salad task, conversation disentanglement focuses on dialogues and has many types of metadata available, such as time stamps, discourse information, and chat handles. Existing systems draw heavily on this metadata. Another related task is the distinction of on-topic and off-topic documents (Bekkerman and Crammer, 2008), which is defined in terms of topical relatedness. In comparison, the story salad task offers opportunities for more in-depth reasoning, as we show below.

3 Data

Natural story salads arise when multiple messy narratives exist to describe the same event or outcome. Often this is because each contribution to the explanation only addresses a small aspect of the larger picture. We can directly simulate the confusion this kind of discourse creates by taking multiple narratives, cutting them into small pieces, and mixing them together.

---

salads are available for download directly and we have provided code to reconstruct the NYT salads from English Gigaword 5 (available as LDC2003T05).

[4] The story salad task is more similar to multichoice narrative cloze (Granroth-Wilding and Clark, 2016) in this regard, but formulated categorically differently.

Dataset	Salads	Total Words	μ Words/Salad	cos (test)
NYT	573,681	217,841,716	379.726	0.33
NYT-HARD	1,000	20,149	438.220	0.56
WIKI	500,000	197,175,135	394.350	0.35
WIKI-HARD	50,374	21,266,243	422.167	0.64

Table 1: Statistics of the datasets we present. The average topic cosine similarity scores (cos) between the two narratives in document mixtures are computed from the test sets. The NYT, WIKI and WIKI-HARD salads are divided into 80%/20% train/test splits, while the smaller NYT-HARD is treated entirely as test.

Data generation. Story salads are generated by combining content from source documents and randomizing the sentence order of the resulting mixture. In order to ensure appropriately sized salads, we require that each source document contain at least eight sentences. Furthermore, to avoid problematically large salads, we pull paragraphs from source documents one at a time until the eight sentence minimum is met. While this procedure can be used to mix any number of documents, we currently present mixtures of two documents.

We utilize two different corpora as sources for story salad generation: (i) the subset of New York Times articles presented within English Gigaword (Graff and Cieri, 2003) and (ii) English Wikipedia[5] (Wikipedia contributors, 2004). An overview of the datasets is available in Table 1.

Gigaword. From the New York Times subset of Gigaword, we compiled a set of 573,681 mixtures we call NYT. Each mixture in this set is constructed from source articles pulled from the same month and year. Because this temporal constraint is the only restriction put on what articles can be mixed, it is possible for a salad to be constructed from topically disparate source documents (e.g., a restaurant review and a political story). We intend NYT to be relatively easy on the whole as a result of this design choice.

However, it is also possible for articles about dominant news stories and trends (e.g., the OJ Simpson trial in the summer of 1994) to be mixed as a result of the same temporal constraint. We therefore pulled out a curated subset of NYT consisting only of salads generated from highly topically similar source documents which we call NYT-HARD. This subset consists of the 1,000 salads where the source documents are most topically similar. We calculate topic similarity scores by computing the cosine similarity between the average word embeddings for each source document (denoted cos hereafter)

$$\cos(d) = \frac{g(\omega_1) \cdot g(\omega_2)}{\| g(\omega_1) \| \| g(\omega_2) \|} \qquad (1)$$

where ω_1 and ω_2 are the source documents, g is a function that computes the average word embedding of a document, and d is the salad under evaluation. The cos scores on the test portion of the datasets are presented in Table 1.

Wikipedia. From Wikipedia, we present an additional set of 500k salads constructed by combining random articles which we call WIKI.

We also leverage Wikipedia category membership as a form of human-annotated topic information. We use this to create a set of 50,374 salads, henceforth called WIKI-HARD, by restricting the domain of articles to only those appearing in categories containing the words *conflict* and *war*. Each mixture in this set is generated from source articles from the same category in order to produce highly difficult mixtures. We intend this to be a challenge set in this domain as the constituent articles for a given mixture are intentionally selected to be closely related. While we have used the category information to construct an intentionally very difficult set for this paper, we note that this procedure can be used to create sets of varying difficulty.

The fact that WIKI-HARD is generated from human-annotated category labels differentiates it from NYT-HARD in the source of its difficulty. After manually reviewing 20 samples from each *-HARD dataset, we found that NYT-HARD more frequently contains salads that are impossible for humans to pick apart while WIKI-HARD more frequently contains salads that are possible, though challenging. In particular, in 9 out of 20 WIKI-HARD salads we found that access to world knowledge and inference would be beneficial. Nevertheless, the two *-HARD datasets are both high in topic similarity (Table 1).

In Figure 2 we present sentences from a sample WIKI-HARD salad that can be solved with world knowledge. In this salad, we learn about two individuals. We can tell that Randle, born in 1855, is unlikely to also have been enrolled in high school in 1913 at the age of 58. We also learn that Randle was a doctor, while Martins, the other individual, was involved in theater. From this, we can deduce that the individual who "also worked as a wrestler" is more likely to be Martins than Randle.

---

[5] Wikipedia dump pulled on January 20, 2018.

(A) John K. Randle was born on 1 February 1855, son of Thomas Randle, a liberated slave from an Oyo village in the west of what is now Nigeria. (B) In 1913 he was enrolled in Eko Boys High School but dropped out. (B) Martins joined the theatre and from there took on various theatre jobs to survive. (A) Born in Sierra Leone , he was one of the first West Africans to qualify as a doctor in the United Kingdom. (B) He also worked as a wrestler (known as "Black Butcher Johnson").

Figure 2: A story salad from WIKI-HARD, sourced from articles belonging to the *Nigerian people of World War I* category. The sentences from this salad have been rearranged for clearer presentation.

Event Representation. Finally, we explore a form of document representation that has been shown to be useful in narrative schema learning, a related task. We include variants of NYT and NYT-HARD with story salads consisting of event tuple representations instead of natural language sentence representations, as in Pichotta and Mooney (2016). We label these variants as NYT-EVENT and NYT-EVENT-HARD. Event tuples are in the form <VERB, SUBJ, DOBJ, PREP, POBJ>, where as many preposition and prepositional object pairs as necessary are allowed.[6]

Summary. The story salads we present here are, in the end, simpler than those that occur naturally in the news or on social media: for one thing, sentences drawn from a document written by a single author should exhibit a high degree of coherence. We have also shown that we can use Wikipedia category annotations to produce large-scale story salad datasets with customizable levels of difficulty, enabling us to increase the difficulty of the task as performance increases. In the following section, we see that both our standard and *-HARD mixtures are challenging for current models. Furthermore, our WIKI-HARD dataset contains salads featuring conflicting information and is an attractive setting for building models with deeper reasoning capabilities.

4 Models

We treat the story salad task as a narrative clustering task where, in our dataset, each salad is comprised of two clusters. Accordingly, the first baselines we consider are standard clustering approaches.

Baselines. Our first baseline is a simple *uniform baseline* (hereafter **UNIF**), where we assign all sentences in a document mixture to a single cluster. Under UNIF the clustering accuracy is the percentage of the majority-cluster sentences, e.g. if a mixture has 7 sentences from one narrative and 3 from the other, then the accuracy is 0.7.

Additionally, we explore a family of baselines that consist of clustering off-the-shelf sentence embeddings. We choose k-medoids[7] (hereafter **KM**) as our clustering algorithm. For sentence embeddings, we experimented with (i) averaged 300D GloVe embeddings (Pennington et al., 2014), which have been shown to produce surprisingly strong performance in a variety of text classification tasks (Iyyer et al., 2015; Coates and Bollegala, 2018); (ii) skip-thought embeddings (Kiros et al., 2015); and (iii) SCDV (Mekala et al., 2017), a multisense-aware sentence embedding algorithm which builds upon pretrained GloVe embeddings using a Gaussian mixture model. Averaged GloVe embeddings gave the best performance in our experiments; to avoid clutter, we only report those results henceforth.

Neural supervised clustering. Our baselines work directly on sentence embeddings and therefore ignore the task-specific supervision available in our labeled training data. Inspired by the work in Bilenko et al. (2004) and Finley and Joachim (2005) on supervised clustering, we aim to exploit this supervision using a learned distance metric in our clustering.[8]

Figure 3 shows our model, which produces a distribution $P(\texttt{same} \mid s_1, s_2, d)$: the probability that two sentences s_1 and s_2 taken from document mixture d are in the same cluster. We train this model as a binary classifier on sampled pairs of sentences to distinguish same-narrative sentence pairs (positive examples) from different-narrative pairs (negative examples). $1-P(\texttt{same} \mid s_1, s_2, d)$ is then used by the clusterer as the pairwise distance metric. Given the pairwise distance between all sentence pairs in a mixture, the KM algorithm

---

[6] Event tuples are extracted via the extractor presented in Cheng and Erk (2018), and copular verbs are not treated as events, meaning that some sentences translate to null events.

[7] K-medoids is chosen as a substitute for k-means because the latter does not extend easily to our classifier-aided neural models: it does not work when only pairwise distances are available. In empirical evaluation we found k-means and k-medoids to produce very similar accuracy scores when using off-the-shelf embeddings. Experiments with hierarchical agglomerative clustering (not reported here) showed it to perform worse than either method.

[8] In early experiments, another strong candidate we tried is a joint model of a sentence autoencoder and a clustering algorithm (Yang et al., 2017). However, this produces subpar performance (weaker than the strongest baseline), due partially to scalability issues in learning these jointly.

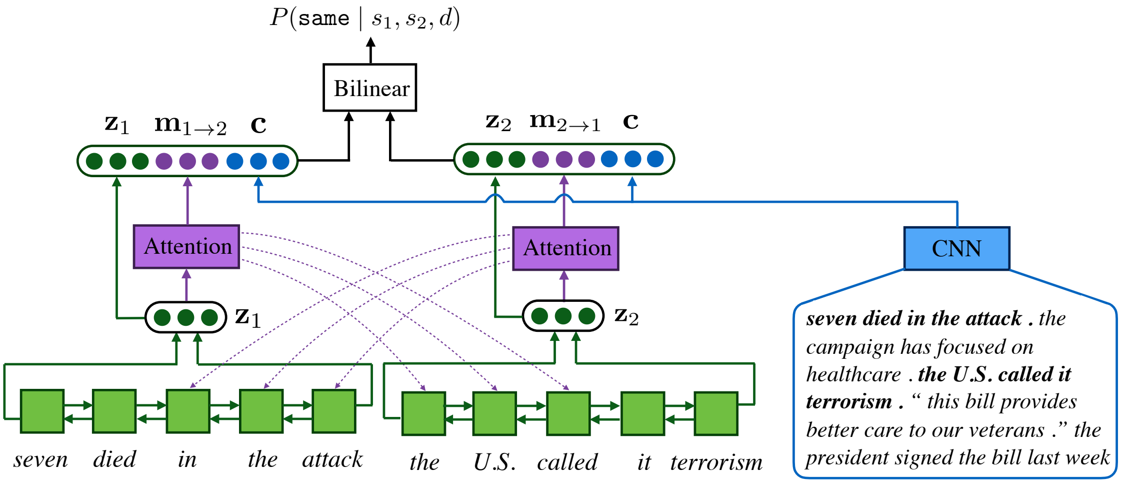

Figure 3: BiLSTM sentence pair classifier to determine whether s_1 and s_2 are from the same narrative, augmented with a mutual attention and a context reader. The three subcomponents — the BiLSTM, the mutual attention mechanism, and the context reader — each produce vectors, denoted as $\boldsymbol{z}, \boldsymbol{m}, \boldsymbol{c}$ respectively. In the basic BILSTM model, only $\boldsymbol{z}$ is fed to the bilinear layer (Eq. 2), while more sophisticated models incorporate the additional mutual attention and context vectors.

can then be applied to cluster sentences into two narratives.

Our classifier is a neural network model built on top of LSTM sentence encoders, which perform well at similar text classification tasks (Dai and Le, 2015; Liu et al., 2016).[9] Denoting a sentence as the list of embeddings of its constituent words: $s = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M\}$, we first encode it as a sentence embedding $\boldsymbol{z}$ with a bidirectional LSTM $\boldsymbol{z} = \texttt{BiLSTM}(s)$ and then compute the probability score with a bilinear layer:

$$P(\texttt{same} \mid s_1, s_2) = \sigma(\boldsymbol{z}_1^T W \boldsymbol{z}_2) \tag{2}$$

This model corresponds to the green subset of Figure 3.

Stronger models. There are two additional effects we might want our model to capture. First, whether two sentences are from the same narrative cannot be determined globally: there aren't two "globally-contrasted"[10] narratives (or bag-of-words based topics) from which sentences are sampled. In other words, sentences are always (pairwise) compared *in the context* of the document mixture from which they are drawn. Second, we want to capture more in-depth interactions between sentences: our sentence embedding scheme for a sentence s_1 should exploit its point of comparison s_2 and encode s_1 with a view of similarities to and differences with s_2. This type of technique has been useful in tasks like natural language inference (NLI) (Bowman et al., 2015; Peters et al., 2018).

To improve contextualization, we add a CNN-based context encoder to the BiLSTM classifier: the reader embeds the whole document salad at hand into a vector. Formally, we compute $\boldsymbol{c} = \text{CNN}(d)$, where in this case CNN denotes a single convolution layer with max pooling in the style of Kim (2014) and d is the concatenation of all sentences in the mixture. This component is shown in blue in Figure 3. The context vector $\boldsymbol{c}$ is then appended to $\boldsymbol{z}$ and fed into the bilinear layer.

To capture the interaction between two sentences in a pair, we employ a *mutual attention* mechanism, which is similar to the attentive reader (Hermann et al., 2015). Let $\boldsymbol{e}_{i,1\ldots n}$ denote the BiLSTM outputs for the tokens of sentence i. Given the encoding $\boldsymbol{z}_1$ of sentence s_1, we compute attention weights and a representation of s_2 as follows:

$$\boldsymbol{\alpha}_{1\to2} = \text{softmax}_j(\boldsymbol{z}_1^\top \boldsymbol{e}_{2,j})$$
$$\boldsymbol{m}_{1\to2} = \sum_j \alpha_{1\to2,j}\, \boldsymbol{e}_{2,j}$$

We compute $\boldsymbol{m}_{2\to1}$ analogously. This process is shown in purple in Figure 3. The $\boldsymbol{m}$ vectors are used as additional inputs to the bilinear layer.

For comprehensive ablation, we experiment with four variants of neural classifiers: (i) BiL-

---

[9] Experiments with convolutional encoders here yielded somewhat worse results.

[10] Two stories may be on the same topic and still form clearly different narratives. For example, both narratives in Figure 1 are regarding military conflict.

STM alone (**BILSTM**); (ii) BiLSTM + mutual attention (**BILSTM-MT**); (iii) BiLSTM + context (**BILSTM-CTX**); and (iv) BiLSTM + mutual attention and context (**BILSTM-MT-CTX**).

Event-based models. For the event-based variants of the datasets, NYT-EVENT and NYT-EVENT-HARD, we build three models: (i) FFNN-BILSTM: we input a sentence as a sequence of event embeddings rather than word embeddings as in BILSTM, where a feedforward layer maps the words in an event tuple to an event embedding; (ii) FFNN-BILSTM-MT-CTX: replacing the base BILSTM in (i) with our best model which is enhanced with mutual attention and contextualization; (iii) FFNN-BILSTM-MT-CTX-PRETRAIN: a variant of (ii) that is based on the event embedding pretraining method[11] described in Weber et al. (2018), where events are encoded with a feedforward net (same as (i)) and trained with a word2Vec-like objective, encouraging events that co-occur in the same narrative to have more similar embeddings.

5 Experiments and Analysis

Experimental setup. To stave off sparsity, we impose a vocabulary cut by using only the 100k most frequent lemmas. To evaluate on NYT, NYT-EVENT, WIKI and WIKI-HARD, we sample 20k unique salads (from their respective test portions[12]) to use for both the sentence and event versions of the experiments. For WIKI-HARD, the training combines the training portions of both WIKI and WIKI-HARD. For NYT-HARD, we train on the training portion of NYT and evaluate on NYT-HARD in full as a test set.

All the neural components are constructed with TensorFlow and use the same hyperparameters across variants: a 2-layer BiLSTM, learning rate 1e-5 with Adam (Kingma and Ba, 2014), dropout (Srivastava et al., 2014) rate 0.3 on all layers, and Xavier initialization (Glorot and Bengio, 2010). To create training pairs for the neural classifiers, we randomly sample sentence pairs balanced between same-narrative and different-narrative pairs. We train with a batch size of 32 and stop when an epoch yields less than 0.001% accuracy improvement on the validation set, which is 5% of mixtures sampled from the training data beforehand (the models are not trained on the validation sample). For KM we use the default configurations of off-the-shelf software.[13]

Model	NYT	WIKI	NYT-HARD	WIKI-HARD
UNIF	52.7	50.9	52.5	51.2
KM	76.4	74.9	59.8	60.4
BILSTM	78.5	76.2	55.3	59.8
BILSTM-MT	80.8	78.8	56.7	61.3
BILSTM-CTX	82.6	78.9	63.8	63.7
BILSTM-MT-CTX	**84.9**	**81.8**	**68.0**	**66.6**

Table 2: Clustering accuracy (CA) results from the sentence based experiments. More sophisticated models do better across all datasets, particularly on *-HARD tasks, which are substantially more challenging.

Evaluation. We evaluate all models in terms of a *clustering accuracy* metric (hereafter CA), which is a simple extension from the conventional accuracy metric: we calculate the ratio of correctly clustered sentences in a document mixture, averaged over test mixtures. Given a document mixture d_i, we call its component documents A and B. Let `pred` be a function that does the clustering by mapping each sentence $s_{i,n}$ of mixture d_i to either A or B, and $\texttt{true}_{AB}$ a function that returns the original pseudo-labels (i.e. $\{A, B\}$) as they are, and $\texttt{true}_{BA}$ flips the pseudo-labels, i.e. $A \rightarrow B$ and $B \rightarrow A$. Then the clustering accuracy for document d_i by `pred` is

$$\begin{aligned} \text{CA}(d_i, \texttt{pred}, \texttt{true}) = \max\{&\text{C}(d_i, \texttt{pred}, \texttt{true}_{AB}), \\ &\text{C}(d_i, \texttt{pred}, \texttt{true}_{BA})\} \end{aligned}$$

$$\text{C}(d_i, \texttt{pred}, \texttt{true}) = \frac{1}{N_i} \sum_n \mathbb{1}[\texttt{pred}(s_{i,n}) = \texttt{true}(s_{i,n})]$$

where $s_{i,n}$ is the n-th sentence of mixture d_i.

Sentence based models. First we evaluate the sentence based models. We first run the UNIF baseline on all our datasets, where we obtain near-50% clustering accuracy. This indicates that the data are all balanced in the number of sentences in the two narratives of the mixtures. We then run k-medoids (KM) on sentence embeddings as a baseline to compare to the classifier-aided models. The results are summarized in Table 2.

We first observe that the KM is a strong baseline and outperforms the supervised BILSTM system in the harder settings. Adding the mutual attention mechanism and contextualization, however, im-

---

[11] In Weber et al. (2018), a more complex tensor-based model is applied. Using exactly same method in our experiments we obtain weaker results.

[12] Test sets available with data release.

[13] `github.com/letiantian/kmedoids`

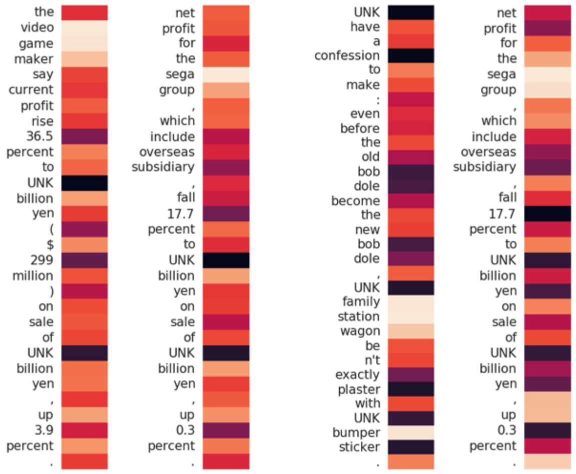

(a) from same narrative

(b) from different narratives

Figure 4: Attention weight heatmaps for a random sample with BILSTM-MT-CTX. Lighter color indicates higher attention weights. The two heatmaps in the same block are for the attention weights of one sentence attending to the other. In (a), we see related concepts being identified (*video game* and *sega*), while in (b), we see a contrast (*family station wagon* and *sega group*).

prove BILSTM substantially. In addition, the performance boost from the two components seems more or less orthogonal, as shown by the much stronger accuracy of the combined model (i.e. BILSTM-MT-CTX) than the models with a single component (i.e. BILSTM-MT and BILSTM-CTX). Overall, the large margin of KM and classifier-aided models above the UNIF baseline indicates that separating story salads is a valid task where generalizable patterns can be exploited by machine learning techniques.

Why would the mutual attention mechanism help? Plotting the attention weights of randomly selected samples, we see distributionally similar words being attended to in Figure 4a. Intuitively, a BiLSTM compresses a sentence into a single vector, leading to information loss (Conneau et al., 2018). Mutual attention enriches this representation by allowing us access to detailed information in sentences at word-level resolution by capturing lexical similarity. Even more interestingly, we observe a synergistic effect between mutual attention and contextualization: with the context reader added, we see high attention weights on words/phrases which bear little distributional similarity but are important for connecting/contrasting different narratives. For example, in Figure 4b, *sega group* and *family station wagon* are selected by the attention, despite not having similar words in the other sentences. These words are crucial in identifying the two narratives in this mixture: one is about a Japanese video game company, the other is on vehicle manufacturing in the U.S.

Type	Model	NYT	WIKI
-CONTEXT	BILSTM	-0.40^*	-0.43^*
	BILSTM-MT	-0.38^*	-0.40^*
+CONTEXT	BILSTM-CTX	-0.31^*	-0.30^*
	BILSTM-MT-CTX	-0.27^*	-0.25^*

Table 3: Spearman's ρ correlation between clustering accuracy (CA) and topic similarity (COS) in the evaluation with NYT and WIKI. The p-values are all below 0.01 (indicated by *). Contextualized models (+CONTEXT) are more robust to high topic similarity than their uncontextualized counterparts (-CONTEXT), indicated by the lower negative correlation between their accuracy and topic similarity.

Another observation is that all models see drastic reduction in accuracy in the *-HARD version of the data. In fact, the clustering accuracy corresponds well with our topic similarity metric (COS, Eq. 1; Table 1) across models. In addition, COS is negatively correlated with clustering accuracy for all mixtures (Table 3).

From the results we also see that contextualization brings clear performance improvement. This supports our hypothesis that the Story Salad task is a nonstandard clustering task where the contrast of two narratives is only meaningful *in the context* of the particular mixture where they reside, rather than on a corpus-general level. Taking the example in Figure 1, the Russian-Chechnya and the U.S.-Afghanistan narratives are contrasted in that mixture, but one can easily imagine a mixture where they are in the same narrative and are contrasted to another narrative on business affairs. Further, contextualized models are less vulnerable to the performance reduction on mixtures with high topic similarity: for one thing, contextualization improves performance over the base BILSTM on both regular and *-HARD datasets. Secondly, computing the correlation between clustering accuracy and topic similarity, we see a lower negative correlation for contextualized models, true for both NYT and WIKI datasets (Table 3).

Event based models. While the accuracy scores in the event based experiments are in general lower than those in the sentence based (Table 4), overall we observe the same pattern that

Model	NYT-EVENT	NYT-EVENT-HARD
KM	64.7	55.3
FFNN-BILSTM	64.9	54.8
*-MT-CTX	66.8	59.1
*-MT-CTX-PRETRAIN	**70.2**	**61.4**

Table 4: Clustering accuracy (CA) results from the event based experiments. *-MT-CTX is a short hand for FFNN-BILSTM-MT-CTX. The same notation applies for the following models.

mutual attention and contextualization contribute substantially to the performance. More interestingly, the performance reduction on the topically highly similar *-HARD is more mild compared to the sentence based experiments, which provides initial evidence that event-based narrative encoding allows the models to be more robust to distraction by lexical overlap in topically similar narratives. Finally we see that the event pretraining with Weber et al. (2018)'s technique brings additional improvement over a contextualized system.

The results open up a door for future work: (i) our simple models do not make use of coreference, narrative schema or world knowledge, which are intuitively promising components to introduce (see, e.g., the salad in Figure 2); (ii) more sophisticated model architectures may help capture the information missed by our models: moving from the sentence version to the event version, we lose many words which may have provided crucial cues in the sentence-based experiments.

Error analysis. In order to understand the errors made by each model, we performed a manual analysis of a small sample of bad clusterings. In a sample of 60 mixtures from NYT (test set), we considered all clusterings for which accuracy was less than 0.65. Among the 60 mixtures, the base model had an accuracy this low for 27 mixtures, the BILSTM-MT and BILSTM-CTX model had 13 low-accuracy mixtures each, and BILSTM-MT-CTX had 3. Each mixture was manually annotated by 2 annotators as being sourced from (i) thematically closely related documents (e.g., two stories on the same political event), (ii) thematically distinct documents (e.g., a political story and a sports story), or (iii) cannot tell.

(A) lehman brothers be one of several investment bank eager to get UNK hand on state asset, across the nation and in massachusetts (A) former massachusetts governor william f. weld, a staunch supporter of privatization during UNK administration, have UNK in the hall of the state house, now as a corporate lawyer try to drum up support for the sale of lucrative state asset. (B) officially, the rays option dukes, 22, to class a vero beach and place UNK on the temporary inactive list, where UNK will remain for an undetermined amount of time as UNK undergo counseling. (B) UNK apparently will receive UNK $ 380,000 major-league salary .

Figure 5: An example of (preprocessed) sentences from two unrelated documents being that have been clustered into a single cluster by the base model. Document (A) is an article about proposed privatization of public assets, while Document (B) is an article about happenings in Major League Baseball.

Our analysis showed that the base BILSTM model has difficulty even in cases where the source documents for the salad are thematically distinct. This was the case in 9 of 27 bad clusterings. The BILSTM-MT, BILSTM-CTX and BILSTM-MT-CTX models not only have many fewer bad clusterings, they also show low accuracy almost exclusively in mixtures of related documents (2 cases of distinct documents for BILSTM-CTX, none for BILSTM-MT or BILSTM-MT-CTX). Figure 5 shows an example of a bad clustering of two unrelated documents, produced by the base BILSTM model.

In a second study, we rated the same 60 samples by their difficulty for a human, focusing in particular on mixtures that went from low performance (0.5-0.65) in the BILSTM model to high performance (0.8-1.0) in another model. For BILSTM-CTX we find that only 2 out of 11 mixtures with such marked improvement over BILSTM were hard for humans; for BILSTM-MT only 1 out of 9 markedly improved mixtures was hard for humans. But for BILSTM-MT-CTX, 8 out of 17 markedly improved mixtures were hard for humans, indicating that more sophisticated models do better not only on easy but also on hard cases.

In a third small study, we compare NYT-HARD and WIKI-HARD for their difficulty for humans, looking at 20 mixtures each. Here, very interestingly, we find more mixtures that are impossible for humans in NYT-HARD (10 cases, example in Figure 6) than WIKI-HARD (3 cases). This presents a clear discrepancy between difficulty for humans and difficulty for models: the models do better on NYT-HARD which is harder for us. While we would not want to draw strong conclusions from a small sample, this hints at possibilities of future work where world knowledge, which is likely to be orthogonal to the information picked up by the models, can be introduced to improve performance (e.g. Wang et al. (2018)).

Note that unlike many other NLP tasks where

(A) The most basic question face the country on energy be how to keep supply and demand in line. The Democrats would say : “what can UNK do to make good use of what UNK have get?” (B) Oil price dominate the 31-minute news conference, hold here near pittsburgh. (B) Vice President Al Gore hold UNK first news conference in 67 day on Friday, defend UNK call for the release of oil from the government’s stockpile and and vow that UNK would “confront friend and foe alike” over the marketing of violent entertainment to child, despite the million in donation UNK receive from Hollywood. (A) With oil price up, consumer agitate and the winter heating season loom, Vice President Al Gore and Gov. George W. Bush be go at UNK on energy policy, seek to draw sharp distinction over an issue on which both candidate have political vulnerability. (B) On other topic, Gore say UNK be not nervous about the upcoming debate, but be incredulous when a reporter ask whether UNK be confident UNK have the election lock up. (A) Bush, who criticize the decision as a political ploy to drive down price just ahead of election day, be schedule to discuss energy policy in a speech on Friday. (B) On Friday, Bush call Gore a “flip-flopper”, say UNK proposal to tap into the reserve be a political ploy.

Figure 6: Part of a story salad that is impossible for a human to pick apart (source: NYT-HARD). “UNK” represents out-of-vocabulary tokens, and all the words are lemmatized. Both narratives, i.e. (A) and (B) involve the characters Al Gore and George Bush, and both are on the topic of energy, with strongly overlapping vocabulary.

human performance sets the ceiling for the best achievable results (e.g. span-prediction based question answering (Rajpurkar et al., 2016), where all the information needed for the correct answer is available in the input), successfully picking apart narratives in a story salad may require consulting an external knowledge base, which affords machine learning models a clear advantage over humans. For example, recognizing that *Commander Kamal* is likely to be Afghani based on his name, which is not knowledge every reader possesses, would allow us to successfully cluster the sentence with the U.S.-Afghanistan narrative rather than the Russian-Chechnya narrative.

6 Conclusion

We have presented a technique to generate *Story Salads*, mixtures of multiple narratives, at scale. We have demonstrated that the difficulty of these mixtures can be manipulated either based on document similarity or based on human-created document categories. This data gives rise to a challenging binary clustering task (but easily extended to n-ary), where the aim is to group sentences that come from the same original narrative. As coherence plays an important role in this task, the task is related to work on narrative schemas (Chambers and Jurafsky, 2008; Pichotta and Mooney, 2016) and textual coherence (Barzilay and Lapata, 2008; Logeswaran et al., 2018). The automated and scalable data generation technique allows for the use of neural models, which need large amounts of training data.

Conducting a series of preliminary experiments on the data with common unsupervised clustering algorithms (Cao and Yang, 2010) and variants of neural network-based (Kim, 2014; Dai and Le, 2015; Liu et al., 2016) supervised clustering (Bilenko et al., 2004; Finley and Joachim, 2005) models, we have (i) verified the validity of the task where generalizable patterns can be learned through machine learning techniques; (ii) shown that this is a nonstandard clustering task in which the contrast between narratives is *in context* as opposed to global; (iii) found that there is a class of mixtures that are doable for humans but very difficult for our current models, and that in particular the category-based method creates a high proportion of such mixtures.

Our work opens up a large number of directions for future research. First, while our models obtain strong results on simpler story salads, they have low performance on more difficult mixtures with high topical similarity. Second, there are many intuitively promising sources of information that we have not explored, such as coreference. And third, our models rely on pairwise similarity-based coherence learning, which leads to the natural question of whether structured prediction would improve performance.

Acknowledgments

This research was supported by the DARPA AIDA program under AFRL grant FA8750-18-2-0017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, DoD or the US government. We acknowledge the Texas Advanced Computing Center for providing grid resources that contributed to these results. We are grateful to the anonymous reviewers for helpful discussions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of NIPS*.

Regina Barzilay and Mirella Lapata. 2008. Modeling Local Coherence: An Entity-Based Approach. *Computational Linguistics*, 34(1):1–34.

Ron Bekkerman and Koby Crammer. 2008. One-class Clustering in the Text Domain. In *Proceedings of EMNLP*, pages 41–50.

Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. 2004. Integrating Constraints and Metric Learning in Semi-Supervised Clustering. In *Proceedings of ICML.*

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A Large Annotated Corpus for Learning Natural Language Inference. In *Proceedings of EMNLP*.

Danyang Cao and Bingru Yang. 2010. An Improved K-Medoids Clustering Algorithm. In *Proceedings of ICCAE, IEEE.*

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *Proceedings of ACL.*

Nathanael Chambers and Daniel Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and Their Participants. In *Proceedings of ACL.*

Pengxiang Cheng and Katrin Erk. 2018. Implicit Argument Prediction with Event Knowledge. In *Proceedings of NAACL.*

Joshua Coates and Danushka Bollegala. 2018. Frustratingly Easy Meta-Embedding Computing Meta-Embeddings by Averaging Source Word Embeddings. In *Proceedings of NAACL.*

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What You Can Cram into a Single Vector: Probing Sentence Embeddings for Linguistic Properties. In *Proceedings of ACL.*

Alexis Conneau, Holger Schwenk, Yann LeCun, and Loïc Barrault. 2017. Very Deep Convolutional Networks for Text Classification. In *Proceedings of EACL.*

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Proceedings of NIPS.*

Micha Elsner and Eugene Charniak. 2008. You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement. In *Proceedings of ACL.*

Thomas Finley and Thorsten Joachim. 2005. Supervised Clustering with Support Vector Machines. In *Proceedings of ICML.*

Xavier Glorot and Yoshua Bengio. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of ICML.*

David Graff and Christopher Cieri. 2003. English Gigaword LDC2003T05. *Linguistic Data Consortium.*

Mark Granroth-Wilding and Stephen Clark. 2016. What Happens Next? Event Prediction Using a Compositional Neural Network model. In *Proceedings of AAAI.*

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Proceedings of NIPS.*

Sepp Hochreiter and Juergen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Mohit Iyyer, Vrun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of ACL.*

Jyun-Yu Jiang, Francine Chen, Yang-Ying Chen, and Wei Wang. 2018. Learning to Disentangle Interleaved Conversational Threads with a Siamese Hierarchical Network and Similarity Ranking. In *Proceedings of NAACL.*

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of EMNLP.*

Diederik P. Kingma and Jimmy Ba. 2014. Adam: a Method for Stochastic Optimization. In *Proceedings of ICLR.*

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Proceedings of NIPS.*

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. In *Proceedings of ICLR.*

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent Neural Network for Text Classification with Multi-task Learning. In *Proceedings of IJCAI.*

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2018. Sentence Ordering and Coherence Modeling Using Recurrent Neural Networks. In *Proceedings of AAAI.*

Dheeraj Mekala, Vivek Gupta, Bhargavi Paranjape, and Harish Karnick. 2017. SCDV: Sparse Composite Document Vectors using soft clustering over distributional representations. In *Proceedings of EMNLP.*

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James F. Allen. 2017. LSDSem 2017 Shared Task: The Story Cloze Test. In *LSDSem 2017 Shared Task.*

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Embeddings. In *Proceedings of EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of NAACL*.

Karl Pichotta and Raymond Mooney. 2016. Using Sentence-Level LSTM Language Models for Script Inference. In *Proeedings of ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of EMNLP*.

Roger C. Schank and Robert P. Abelson. 1977. Scripts, Plans, Goals and Understanding. *Lawrence Erlbaum*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a Simple Way to Prevent Neural Network from Overfitting. *JMLR*, 15:1929–1958.

Lidan Wang and Douglas W. Oard. 2009. Context-based Message Expansion for Disentanglement of Interleaved Text Conversation. In *Proceedings of NAACL*.

Su Wang, Greg Durrett, and Katrin Erk. 2018. Modeling Semantic Plausibility by Injecting World Knowledge. In *Proceedings of NAACL*.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Event Representations with Tensor-based Compositions. In *Proceedings of AAAI*.

John Wieting and Kevin Gimpel. 2017. Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings. In *Proceedings of ACL*.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning Paraphrastic Sentence Embeddings from Back-Translated Bitext. In *Proceedings of EMNLP*.

Wikipedia contributors. 2004. Plagiarism — Wikipedia, the free encyclopedia. [Online; accessed 20-January-2018].

Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. 2017. Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering. In *Proceedings of ICML*.

Dynamic Meta-Embeddings for Improved Sentence Representations

Douwe Kiela[†], **Changhan Wang**[†] **and Kyunghyun Cho**[†‡⋆]
[†] Facebook AI Research; [‡] New York University; [⋆] CIFAR Global Scholar
`{dkiela,changhan,kyunghyuncho}@fb.com`

Abstract

While one of the first steps in many NLP systems is selecting what pre-trained word embeddings to use, we argue that such a step is better left for neural networks to figure out by themselves. To that end, we introduce dynamic meta-embeddings, a simple yet effective method for the supervised learning of embedding ensembles, which leads to state-of-the-art performance within the same model class on a variety of tasks. We subsequently show how the technique can be used to shed new light on the usage of word embeddings in NLP systems.

1 Introduction

It is no exaggeration to say that word embeddings have revolutionized NLP. From early distributional semantic models (Turney and Pantel, 2010; Erk, 2012; Clark, 2015) to deep learning-based word embeddings (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2016), word-level meaning representations have found applications in a wide variety of core NLP tasks, to the extent that they are now ubiquitous in the field (Goldberg, 2016).

A sprawling literature has emerged about what types of embeddings are most useful for which tasks. For instance, there has been extensive work on understanding what word embeddings learn (Levy and Goldberg, 2014b), evaluating their performance (Milajevs et al., 2014; Schnabel et al., 2015; Bakarov, 2017), specializing them for certain tasks (Maas et al., 2011; Faruqui et al., 2014; Kiela et al., 2015; Mrkšić et al., 2016; Vulić and Mrkšić, 2017), learning sub-word level representations (Wieting et al., 2016; Bojanowski et al., 2016; Lee et al., 2016), et cetera.

One of the first steps in designing many NLP systems is selecting what kinds of word embeddings to use, with people often resorting to freely available pre-trained embeddings. While this is often a sensible thing to do, the usefulness of word embeddings for downstream tasks tends to be hard to predict, as downstream tasks can be poorly correlated with word-level benchmarks. An alternative is to try to combine the strengths of different word embeddings. Recent work in so-called "meta-embeddings", which ensembles embedding sets, has been gaining traction (Yin and Schütze, 2015; Bollegala et al., 2017; Muromägi et al., 2017; Coates and Bollegala, 2018). Meta-embeddings are usually created in a separate preprocessing step, rather than in a process that is dynamically adapted to the task. In this work, we explore the supervised learning of task-specific, dynamic meta-embeddings, and apply the technique to sentence representations.

The proposed approach turns out to be highly effective, leading to state-of-the-art performance within the same model class on a variety of tasks, opening up new areas for exploration and yielding insights into the usage of word embeddings.

Why Is This a Good Idea? Our technique brings several important benefits to NLP applications. First, it is embedding-agnostic, meaning that one of the main (and perhaps most important) hyperparameters in NLP pipelines is made obsolete. Second, as we will show, it leads to improved performance on a variety of tasks. Third, and perhaps most importantly, it allows us to overcome common pitfalls with current systems:

- **Coverage** One of the main problems with NLP systems is dealing with out-of-vocabulary words: our method increases lexical coverage by allowing systems to take the union over different embeddings.
- **Multi-domain** Standard word embeddings

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1466–1477
Brussels, Belgium, October 31 - November 4, 2018.

are often trained on a single domain, such as Wikipedia or newswire. With our method, embeddings from different domains can be combined, optionally while taking into account contextual information.

- **Multi-modality** Multi-modal information has proven useful in many tasks (Baroni, 2016; Baltrušaitis et al., 2018), yet the question of multi-modal fusion remains an open problem. Our method offers a straightforward solution for combining information from different modalities.
- **Evaluation** While it is often unclear how to evaluate word embedding performance, our method allows for inspecting the weights that networks assign to different embeddings, providing a direct, task-specific, evaluation method for word embeddings.
- **Interpretability and Linguistic Analysis** Different word embeddings work well on different tasks. This is well-known in the field, but knowing *why* this happens is less well-understood. Our method sheds light on which embeddings are preferred in which linguistic contexts, for different tasks, and allows us to speculate as to why that is the case.

Outline In what follows, we explore dynamic meta-embeddings and show that this method outperforms the naive concatenation of various word embeddings, while being more efficient. We apply the technique in a BiLSTM-max sentence encoder (Conneau et al., 2017) and evaluate it on well-known tasks in the field: natural language inference (SNLI and MultiNLI; §4), sentiment analysis (SST; §5), and image-caption retrieval (Flickr30k; §6). In each case we show state-of-the-art performance within the class of single sentence encoder models. Furthermore, we include an extensive analysis (§7) to highlight the general usefulness of our technique and to illustrate how it can lead to new insights.

2 Related Work

Thanks to their widespread popularity in NLP, a sprawling literature has emerged about learning and applying word embeddings—much too large to fully cover here, so we focus on previous work that combines multiple embeddings for downstream tasks.

Maas et al. (2011) combine unsupervised embeddings with supervised ones for sentiment classification. Yang et al. (2016) and Miyamoto and Cho (2016) learn to combine word-level and character-level embeddings. Contextual representations have been used in neural machine translation as well, e.g. for learning contextual word vectors and applying them in other tasks (McCann et al., 2017) or for learning context-dependent representations to solve disambiguation problems in machine translation Choi et al. (2016).

Neural tensor skip-gram models learn to combine word, topic and context embeddings (Liu et al., 2015); context2vec (Melamud et al., 2016) learns a more sophisticated context representation separately from target embeddings; and Li et al. (2016) learn word representations with distributed word representation with multi-contextual mixed embedding. Recent work in "meta-embeddings", which ensembles embedding sets, has been gaining traction (Yin and Schütze, 2015; Bollegala et al., 2017; Muromägi et al., 2017; Coates and Bollegala, 2018)—here, we show that the idea can be applied in context, and to sentence representations. Furthermore, these works obtain meta-embeddings as a preprocessing step, rather than learning them dynamically in a supervised setting, as we do here. Similarly to Peters et al. (2018), who proposed deep contextualized word representations derived from language models and which led to impressive performance on a variety of tasks, our method allows for contextualization, in this case of embedding set weights.

There has also been work on learning multiple embeddings per word (Chen et al., 2014; Neelakantan et al., 2015; Vu and Parker, 2016), including a lot of work in sense embeddings where the senses of a word have their own individual embeddings (Iacobacci et al., 2015; Qiu et al., 2016), as well as on how to apply such sense embeddings in downstream NLP tasks (Pilehvar et al., 2017).

The question of combining multiple word embeddings is related to multi-modal and multi-view learning. For instance, combining visual features from convolutional neural networks with word embeddings has been examined (Kiela and Bottou, 2014; Lazaridou et al., 2015), see Baltrušaitis et al. (2018) for an overview. In multi-modal semantics, for instance, word-level embeddings from different modalities are often mixed via concatenation $\mathbf{r} = [\alpha\mathbf{u}, (1-\alpha)\mathbf{v}]$ (Bruni et al., 2014). Here, we dynamically learn the weights to combine representations. Recently, related dynamic multi-modal fusion methods have also been explored (Wang et al., 2018; Kiros et al., 2018). There has also been work on unifying multi-view embeddings from different data sources (Luo et al., 2014).

The usefulness of different embeddings as initialization has been explored (Kocmi and Bojar, 2017), and different architectures and hyperparameters have been extensively examined (Levy et al., 2015). Problems with evaluating word embeddings intrinsically are well known (Faruqui et al., 2016), and various alternatives for evaluating word embeddings in downstream tasks have been proposed (e.g., Tsvetkov et al., 2015; Schnabel et al., 2015; Ettinger et al., 2016). For more related work with regard to word embeddings and their evaluation, see Bakarov (2017).

Our work can be seen as an instance of the well-known attention mechanism (Bahdanau et al., 2014), and its recent sentence-level incarnations of self-attention (Lin et al., 2017) and inner-attention (Cheng et al., 2016; Liu et al., 2016), where the attention mechanism is applied within the same sentence instead of for aligning multiple sentences. Here, we learn (optionally contextualized) attention weights for different embedding sets and apply the technique in sentence representations (Kiros et al., 2015; Wieting et al., 2015; Hill et al., 2016; Conneau et al., 2017).

3 Dynamic Meta-Embeddings

Commonly, NLP systems use a single type of word embedding, e.g., word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) or FastText (Bojanowski et al., 2016). We propose giving networks access to multiple types of embeddings, allowing a network to learn which embeddings it prefers by predicting a weight for each embedding type, optionally depending on the context.

For a sentence of s tokens $\{\mathbf{t}_j\}_{j=1}^{s}$, we have n word embedding types, leading to sequences $\{\mathbf{w}_{i,j}\}_{j=1}^{s} \in \mathbb{R}^{d_i}$ $(i = 1, 2, \ldots, n)$. We center each type of word embedding to zero mean.

Naive baseline We compare to naive concatenation as a baseline. Concatenation is a sensible strategy for combining different embedding sets, because it provides the sentence encoder with all of the information in the individual embeddings:

$$\mathbf{w}_j^{CAT} = [\mathbf{w}_{1,j}, \mathbf{w}_{2,j}, \ldots, \mathbf{w}_{n,j}].$$

The downside of concatenating embeddings and giving that as input to an RNN encoder, however, is that the network then quickly becomes inefficient as we combine more and more embeddings.

DME For dynamic meta-embeddings, we project the embeddings into a common d'-dimensional space by learned linear functions $\mathbf{w}'_{i,j} = \mathbf{P}_i \mathbf{w}_{i,j} + \mathbf{b}_i$ $(i = 1, 2, \ldots, n)$ where $\mathbf{P}_i \in \mathbb{R}^{d' \times d_i}$ and $\mathbf{b}_i \in \mathbb{R}^{d'}$. We then combine the projected embeddings by taking the weighted sum

$$\mathbf{w}_j^{DME} = \sum_{i=1}^{n} \alpha_{i,j} \mathbf{w}'_{i,j}$$

where $\alpha_{i,j} = g(\{\mathbf{w}'_{i,j}\}_{j=1}^{s})$ are scalar weights from a self-attention mechanism:

$$\alpha_{i,j} = g(\mathbf{w}'_{i,j}) = \phi(\mathbf{a} \cdot \mathbf{w}'_{i,j} + b) \qquad (1)$$

where $\mathbf{a} \in \mathbb{R}^{d'}$ and $b \in \mathbb{R}$ are learned parameters and ϕ is a softmax (or could be a sigmoid or tanh, for gating). We also experiment with an **Unweighted** variant of this approach, that just sums up the projections.

CDME Alternatively, we can make the self-attention mechanism context-dependent, leading to contextualized DME (CDME):

$$\alpha_{i,j} = g(\{\mathbf{w}'_{i,j}\}_{j=1}^{s}) = \phi(\mathbf{a} \cdot \mathbf{h}_j + b) \qquad (2)$$

where $\mathbf{h}_j \in \mathbb{R}^{2m}$ is the j^{th} hidden state of a BiLSTM taking $\{\mathbf{w}'_{i,j}\}_{j=1}^{s}$ as input, $\mathbf{a} \in \mathbb{R}^{2m}$ and $b \in \mathbb{R}$. We set $m = 2$, which makes the contextualization very efficient.

Sentence encoder We use a standard bidirectional LSTM encoder with max-pooling (BiLSTM-Max), which computes two sets of s hidden states, one for each direction:

$$\overrightarrow{\mathbf{h}_j} = \overrightarrow{\mathrm{LSTM}_j}(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_j)$$
$$\overleftarrow{\mathbf{h}_j} = \overleftarrow{\mathrm{LSTM}_j}(\mathbf{w}_j, \mathbf{w}_{j+1}, \ldots, \mathbf{w}_s)$$

The hidden states are subsequently concatenated for each timestep to obtain the final hidden states, after which a max-pooling operation is applied over their components to get the final sentence representation:

$$\mathbf{h} = \max(\{[\overrightarrow{\mathbf{h}_j}, \overleftarrow{\mathbf{h}_j}]\}_{j=1,2,\ldots,s})$$

Model	SNLI	MNLI
InferSent (Conneau et al., 2017)	84.5	-
NSE (Munkhdalai and Yu, 2017)	84.6	-
G-TreeLSTM (Choi et al., 2017)	86.0	-
SSE (Nie and Bansal, 2017)	86.1	73.6
ReSan (Shen et al., 2018)	86.3	-
GloVe BiLSTM-Max (8.6M)	85.2±.3	70.0±.5
FastText BiLSTM-Max (8.6M)	85.2±.2	70.3±.3
Naive baseline (9.8M)	85.6±.3	71.1±.2
Naive baseline (61.3M)	86.0±.5	73.0±.2
Unweighted DME (8.6M)	86.3±.4	74.4±.2
DME (8.6M)	86.2±.2	74.4±.2
CDME (8.6M)	86.4±.3	74.1±.2
DME* (9.0M)	86.7±.2	74.3±.4
CDME* (9.0M)	86.5±.2	74.9±.5

Table 1: Accuracy scores on the Stanford Natural Language Inference (SNLI) and MultiNLI Mismatched (MNLI) tasks. DME=Dynamic Meta-Embeddings; CDME=Contextualized Dynamic Meta-Embeddings; *=multiple different embedding sets (see Section 4). Number of parameters included in parenthesis. Results averaged over five runs with different random seeds, using a BiLSTM-Max sentence encoder.

4 Natural Language Inference

Natural language inference, also known as recognizing textual entailment (RTE), is the task of classifying pairs of sentences according to whether they are neutral, entailing or contradictive. Inference about entailment and contradiction is fundamental to understanding natural language, and there are two established datasets to evaluate semantic representations in that setting: SNLI (Bowman et al., 2015) and the more recent MultiNLI (Williams et al., 2017).

The SNLI dataset consists of 570k human-generated English sentence pairs, manually labeled for entailment, contradiction and neutral. The MultiNLI dataset can be seen as an extension of SNLI: it contains 433k sentence pairs, taken from ten different genres (e.g. fiction, government text or spoken telephone conversations), with the same entailment labeling scheme.

We train sentence encoders with dynamic meta-embeddings using two well-known and often-used embedding types: FastText (Mikolov et al., 2018; Bojanowski et al., 2016) and GloVe (Pennington et al., 2014). Specifically, we make use of the 300-dimensional embeddings trained on a similar WebCrawl corpus, and compare three scenarios: when used individually, when naively concatenated or in the dynamic meta-embedding setting (unweighted, context-independent DME and contextualized CDME). We also compare our approach against other models in the same class—in this case, models that encode sentences individually and do not allow attention *across* the two sentences.[1] We include InferSent (Conneau et al., 2017), which also makes use of a BiLSTM-Max sentence encoder.

In addition, we include a setting where we combine not two, but six different embedding types, adding FastText wiki-news embeddings[2], English-German and English-French embeddings from Hill et al. (2014), as well as the BOW2 embeddings from Levy and Goldberg (2014a) trained on Wikipedia.

4.1 Implementation Details

The two sentences are represented individually using the sentence encoder. As is standard in the literature, the sentence representations are subsequently combined using $\mathbf{m} = [\mathbf{u}, \mathbf{v}, \mathbf{u} * \mathbf{v}, |\mathbf{u} - \mathbf{v}|]$. We train a two-layer classifier with rectifiers on top of the combined representation. Notice that there is no interaction (e.g., attention) between the representations of $\mathbf{u}$ and $\mathbf{v}$ for this class of model.

We use 256-dimensional embedding projections, 512-dimensional BiLSTM encoders and an MLP with 1024-dimensional hidden layer in the classifier. The initial learning rate is set to 0.0004 and dropped by a factor of 0.2 when dev accuracy stops improving, dropout to 0.2, and we use Adam for optimization (Kingma and Ba, 2014). The loss is standard cross-entropy.

For MultiNLI, which has no designated validation set, we use the in-domain *matched* set for validation and report results on the out-of-domain *mismatched* set.

4.2 Results

Table 1 shows the results. We report accuracy scores averaged over five runs with different random seeds, together with their standard deviation, for the SNLI and MultiNLI datasets. We include two versions of the naive baseline: one with a 512-dimensional BiLSTM encoder; and a bigger one with 2048 dimensions. Both naive baseline models outperform the single encoders that have only GloVe or FastText embeddings. This shows how

---

[1] This is a common distinction, see e.g. the SNLI leaderboard at `https://nlp.stanford.edu/projects/snli/`.

[2] See `https://fasttext.cc/`

Model	SST
Const. Tree LSTM (Tai et al., 2015)	88.0
DMN (Kumar et al., 2016)	88.6
DCG (Looks et al., 2017)	89.4
NSE (Munkhdalai and Yu, 2017)	89.7
GloVe BiLSTM-Max (4.1M)	88.0±.1
FastText BiLSTM-Max (4.1M)	86.7±.3
Naive baseline (5.4M)	88.5±.4
Unweighted DME (4.1M)	89.0±.2
DME (4.1M)	88.7±.6
CDME (4.1M)	89.2±.4
CDME*-Softmax (4.6M)	89.3±.5
CDME*-Sigmoid (4.6M)	89.8±.4

Table 2: Sentiment classification accuracy results on the binary SST task. For DCG we compare against their best single sentence model (Looks et al., 2017). *=multiple different embedding sets (see Section 4). Number of parameters included in parenthesis. Results averaged over ten runs with different random seeds.

including more than one embeddings can help performance. Next, we observe that the DME embeddings outperform the naive concatenation baselines, while having fewer parameters. Differences between the three DME variants are small and not significant, although we do note that we found the highest maximum performance for the contextualized version, which adds very few additional parameters. It is important to note that the imposition of weighting thus is not detrimental to performance, which means that DME and CDME provide additional interpretability without sacrificing performance.

Finally, we obtain results for using the six different embedding types (marked *), and show that adding in more embeddings increases performance further. To our knowledge, these numbers constitute the state of the art within the model class of single sentence encoders on these tasks.

5 Sentiment

To showcase the general applicability of the proposed approach, we also apply it to a case where we have to classify a single sentence, namely, sentiment classification. Sentiment analysis and opinion mining have become important applications for NLP research. We evaluate on the binary SST task (Socher et al., 2013), consisting of 70k sentences with a corresponding binary (positive or negative) sentiment label.

	Image		Caption	
Model | R@:	1	10	1	10
VSE++	32.3	72.1	43.7	82.1
FastText (15M)	35.6	74.7	47.1	82.7
ImageNet (29M)	25.6	63.1	36.6	72.2
Naive (32M)	34.4	73.9	46.4	82.2
Unweighted DME (15M)	35.9	75.0	48.9	83.7
DME (15M)	36.5	75.5	49.7	83.6
CDME (15M)	36.5	75.6	49.0	83.8

Table 3: Image and caption retrieval results (R@1 and R@10) on Flickr30k dataset, compared to VSE++ baseline (Faghri et al., 2017). VSE++ numbers in the table are with ResNet features and random cropping, but no fine-tuning. Number of parameters included in parenthesis; averaged over five runs with std omitted for brevity.

5.1 Implementation Details

We use 256-dimensional embedding projections, 512-dimensional BiLSTM encoders and an MLP with 512-dimensional hidden layer in the classifier. The initial learning rate is set to 0.0004 and dropped by a factor of 0.2 when dev accuracy stops improving, dropout to 0.5, and we use Adam for optimization. The loss is standard cross-entropy. We calculate the mean accuracy and standard deviation based on ten random seeds.

5.2 Results

Table 2 shows a similar pattern as we observed with NLI: the naive baseline outperforms the single-embedding encoders; the DME methods outperform the naive baseline, with the contextualized version appearing to work best. Finally, we experiment with replacing ϕ in Eq. 1 and 2 with a sigmoid gate instead of a softmax, and observe improved performance on this task, outperforming the comparable models listed in the table. These results further strengthen the point that having multiple different embeddings helps, and that we can learn to combine those different embeddings efficiently, in interpretable ways.

6 Image-Caption Retrieval

An advantage of the proposed approach is that it is inherently capable of dealing with multi-modal information. Multi-modal semantics (Bruni et al., 2014) often combines linguistic and visual representations via concatenation with a global weight

α, i.e., $\mathbf{v} = [\alpha\mathbf{v}_{ling}, (1-\alpha)\mathbf{v}_{vis}]$. In DME we instead learn to combine embeddings dynamically, optionally based on context. The representation for a word then becomes grounded in the visual modality, and we encode on the word-level *what things look like*.

We evaluate this idea on the Flickr30k image-caption retrieval task: given an image, retrieve the correct caption; and vice versa. The intuition is that knowing what something looks like makes it easier to retrieve the correct image/caption. While this work was under review, a related method was published by Kiros et al. (2018), which takes a similar approach but evaluates its effectiveness on COCO and uses Google images. We obtain word-level *visual embeddings* by retrieving relevant images for a given label from ImageNet in the same manner as Kiela and Bottou (2014), taking the images' ResNet-152 features (He et al., 2016) and subsequently averaging those. We then learn to combine textual (FastText) and visual (ImageNet) word representations in the caption encoder used for retrieving relevant images.

6.1 Implementation Details

Our loss is a max-margin rank loss as in VSE++ (Faghri et al., 2017), a state-of-the-art method on this task. The network architecture is almost identical to that system, except that we use DME (with 256-dimensional embedding projection) and a 1024-dimensional caption encoder. For the Flickr30k images that we do retrieval over, we use random cropping during training for data augmentation and use a ResNet-152 for feature extraction. We tune the sizes of the encoders and use a learning rate of 0.0003 and a dropout rate of 0.1.

6.2 Results

Table 3 shows the results, comparing against VSE++. First, note that the ImageNet-only embeddings don't work as well as the FastText ones, which is most likely due to poorer coverage. We observe that DME outperforms naive and FastText-only, and outperforms VSE++ by a large margin. These findings confirm the intuition that knowing what things look like (i.e., having a word-level visual representation) improves performance in visual retrieval tasks (i.e., where we need to find relevant images for phrases or sentences)—something that sounds obvious but has not really been explored before, to our knowledge. This showcases DME's usefulness for fusing embeddings in multi-modal tasks.

7 Discussion & Analysis

Aside from improved performance, an additional benefit of learning dynamic meta-embeddings is that they enable inspection of the weights that the network has learned to assign to the respective embeddings. In this section, we perform a variety of smaller experiments in order to highlight the usefulness of the technique for studying linguistic phenomena, determining appropriate training domains and evaluating word embeddings. We compute the contribution of each word embedding type as follows:

$$\beta_{i,j} = \frac{\|\alpha_{i,j}\mathbf{w'}_{i,j}\|_2}{\sum_{k=1}^{n}\|\alpha_{k,j}\mathbf{w'}_{k,j}\|_2}$$

7.1 Visualizing Attention

Figure 1 shows the attention weights for a CDME model trained on SNLI, using the aforementioned six embedding sets. The sentence is from the SNLI validation set. We observe that different embeddings are preferred for different words. The figure is meant to illustrate possibilities for analysis, which we turn to in the next section.

7.2 Linguistic Analysis

We perform a fine-grained analysis of the behavior of DME on the validation set of SNLI. Figure 3 shows a breakdown of the average attention weights per part of speech. Figure 4 shows a similar breakdown for open versus closed class. The analysis allows us to make several interesting observations: it appears that this model prefers GloVe embeddings, followed by the two FastText embeddings (trained on Wikipedia and Common Crawl). For open class words (e.g., nouns, verbs, adjectives and adverbs), those three embedding types are strongly preferred, while closed class words get more evenly divided attention. The embeddings from Levy and Goldberg (2014a) get low weights, possibly because the method is complementary with FastText-wiki, which was trained on a more recent version of Wikipedia.

We can further examine the attention weights by analyzing them in terms of frequency and concreteness. We use Norvig's Google N-grams corpus frequency counts[3] to divide the words into fre-

---

[3] `http://norvig.com/mayzner.html`

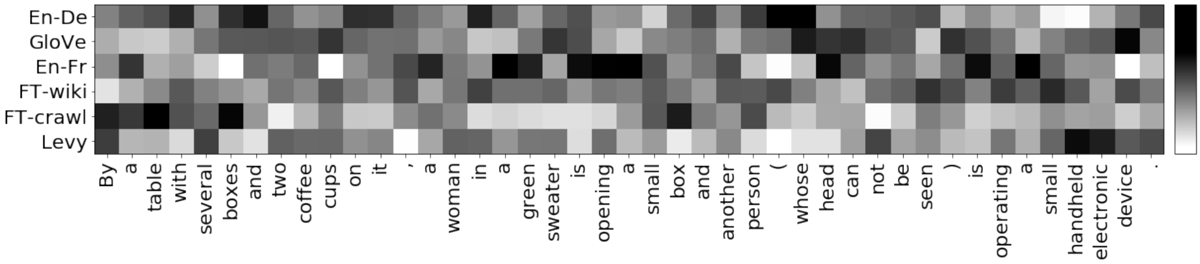

Figure 1: Example visualization of a sentence from the SNLI dev set.

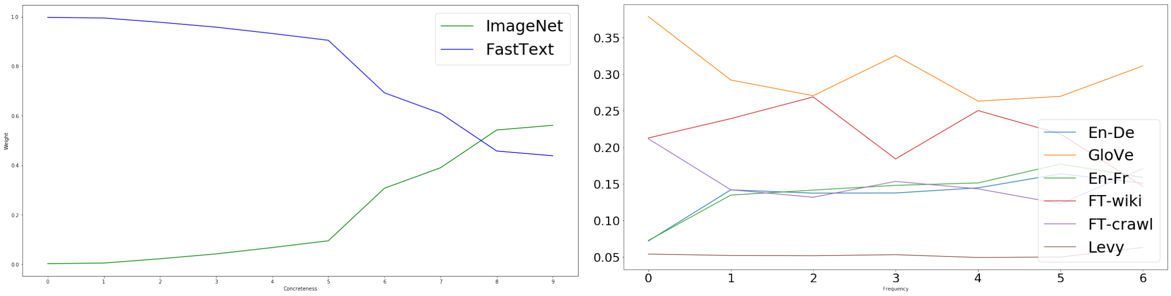

Figure 2: Concreteness weights (left) for Flickr30k model and Frequency weights (right) for SNLI model with multiple embeddings. Visual ImageNet embeddings are preferred for more concrete words. GloVe is strongly preferred for low-frequency words.

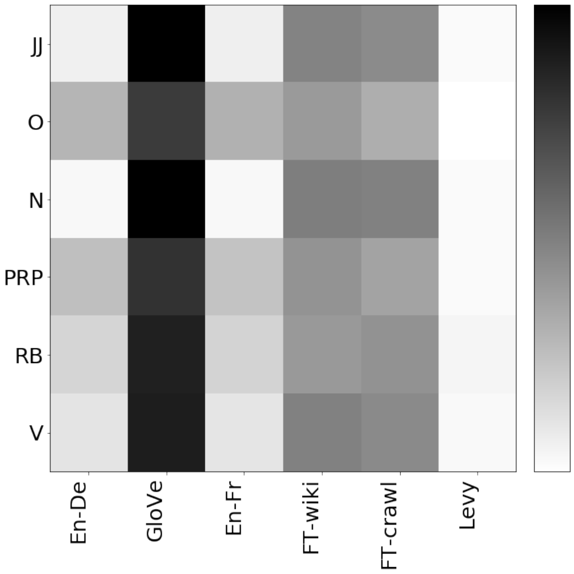

Figure 3: Pos tags and weights by embedding type.

quency bins. Figure 2 (right) shows the average attention weights per frequency bin, ranging from low to high. We observe a clear preference for GloVe, in particular for low-frequency words. For concreteness, we use the concreteness ratings from Brysbaert et al. (2014). Figure 2 (left) shows the average weights per concreteness bin for a model trained on Flickr30k. We can clearly see that visual embeddings get higher weights as the words become more and more concrete.

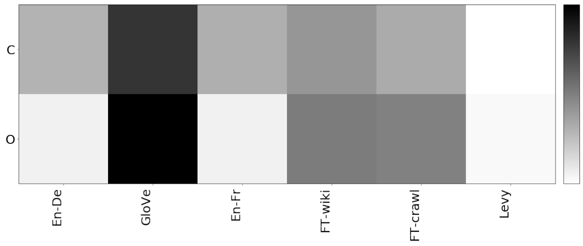

Figure 4: Weights for open/closed class words.

There are of course intricate relationships between concreteness, frequency, POS tags and open/closed class words: closed class words are often frequent and abstract, while open class words could be more concrete, etc. It is beyond the scope of the current work to explore these further, but we hope that others will pursue this direction in future work.

7.3 Multi-domain Embeddings

The MultiNLI dataset consists of various genres. This allows us to inspect the applicability of source domain data for a specific genre. We train embeddings on three kinds of data: Wikipedia, the Toronto Books Corpus (Zhu et al., 2015) and the English OpenSubtitles[4]. We examine the atten-

---

[4] `http://opus.nlpl.eu/OpenSubtitles.php`

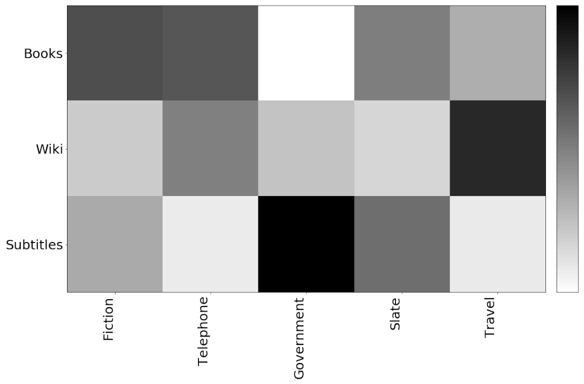

Figure 5: Multi-domain weights on MultiNLI.

Model	Levy	LEAR	SNLI
CDME	0.33	0.67	85.3±.9
Model	GloVe	Refined	SST
CDME	0.59	0.41	89.0±.4

Table 4: Accuracy and learned weights on SNLI using LEAR (Vulić and Mrkšić, 2017) or SST using sentiment-refined embeddings using the specialization method from Yu et al. (2017).

tion weights on the five genres in the in-domain (matched) set, consisting of fiction; transcriptions of spoken telephone conversations; government reports, speeches, letters and press releases; popular culture articles from the Slate Magazine archive; and travel guides.

Figure 5 shows the average attention weights for the three embedding types over the five genres. We observe that Toronto Books, which consists of fiction, is very appropriate for the fiction genre, while Wikipedia is highly preferred for the travel genre, perhaps because it contains a lot of factual information about geographical locations. The government genre makes more use of OpenSubtitles. The spoken telephone genre does not appear to prefer OpenSubtitles, which we might have expected given that that corpus would contain spoken dialogue, but Toronto books, which does include written dialogue.

7.4 Specialization

The above shows that we can use DME to analyze different embeddings on a task. Given the recent interest in the community in specializing, retro-fitting and counter-fitting word embeddings for given tasks, we examine whether the lexical-level benefits of specialization extend to sentence-level downstream tasks. After all, one of the main motivations behind work on lexical entailment is that it allows for better downstream textual entailment. Hence, we take the LEAR embeddings by Vulić and Mrkšić (2017), which do very well on the HyperLex lexical entailment evaluation dataset (Vulić et al., 2017). We compare their best-performing embeddings against the original embeddings that were used for specialization, derived from the BOW2 embeddings of Levy and Goldberg (2014a). Similarly, we use the technique of Yu et al. (2017) for refining GloVe embeddings for sentiment, and evaluate model performance on the SST task.

Table 4 shows that LEAR embeddings get high weights compared to the original source embeddings ("Levy" in the table). Our analysis showed that LEAR was particularly favored for verbs (with average weights of 0.75). The sentiment-refined embeddings were less useful, with the original GloVe embeddings receiving higher weights. These preliminary experiments show how DME models can be used for analyzing the performance of specialized embeddings in downstream tasks.

Note that different weighting mechanisms might give different results—we found that the normalization strategy and the depth of the network significantly influenced weight assignments in our experiments with specialized embeddings.

7.5 Examining Contextualization

We examined models trained on SNLI and looked at the variance of the attention weights per word in the dev set. If contextualization is important for getting the classification decision correct, then we would expect big differences in the attention weights per word depending on the context. Upon examination, we only found relatively few differences. In part, this may be explained by the small size of the dev set, but for the Glove+FastText model we inspected there were only around twenty words with any variance at all, which suggests that the field needs to work on more difficult semantic benchmark tasks. The words, however, where characterized by their polysemy, in particular by having both noun and verb senses. The following words were all in the top 20 most context-dependent words: *mob*, *boards*, *winds*, *trains*, *pitches*, *camp*.

8 Conclusion

We argue that the decision of which word embeddings to use in what setting should be left to the neural network. While people usually pick one type of word embeddings for their NLP systems and then stick with it, we find that dynamically learned meta-embeddings lead to improved results. In addition, we showed that the proposed mechanism leads to better interpretability and insightful linguistic analysis. We showed that the network learns to select different embeddings for different data, different domains and different tasks. We also investigated embedding specialization and examined more closely whether contextualization helps. To our knowledge, this work constitutes the first effort to incorporate multi-modal information on the language side of image-caption retrieval models; and the first attempt at incorporating meta-embeddings into large-scale sentence-level NLP tasks.

In future work, it would be interesting to apply this idea to different tasks, in order to explore what kinds of embeddings are most useful for core NLP tasks, such as tagging, chunking, named entity recognition, parsing and generation. It would also be interesting to further examine specialization and how it transfers to downstream tasks. Using this method for evaluating word embeddings in general, and how they relate to sentence representations in particular, seems a fruitful direction for further exploration. In addition, it would be interesting to explore how the attention weights change during training, and if, e.g., introducing entropy regularization (or even negative entropy) might improve results or interpretability further.

Acknowledgments

We thank the anonymous reviewers for their comments. We also thank Marcus Rohrbach, Laurens van der Maaten, Ivan Vulić, Edouard Grave, Tomas Mikolov and Maximilian Nickel for helpful suggestions and discussions with regard to this work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Amir Bakarov. 2017. A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536*.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2018. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Marco Baroni. 2016. Grounding distributional semantics in the visual world. *Language and Linguistics Compass*, 10(1):3–13.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Danushka Bollegala, Kohei Hayashi, and Ken-ichi Kawarabayashi. 2017. Think globally, embed locally - locally linear meta-embedding of words. *arxiv preprint arXiv:1709.06671*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artifical Intelligence Research*, 49:1–47.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46(3):904–911.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. 2016. Context-dependent word representation for neural machine translation. *arXiv preprint arXiv:1607.00578*.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-lstms. *arXiv preprint arXiv:1707.02786*.

Stephen Clark. 2015. Vector space models of lexical meaning. *Handbook of Contemporary Semantic Theory, The*, pages 493–522.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding—computing meta-embeddings by averaging source word embeddings. In *Proceedings of NAACL-HLT*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of EMNLP*.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 134–139.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. Vse++: Improved visual-semantic embeddings. *arXiv preprint arXiv:1707.05612*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.(JAIR)*, 57:345–420.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014. Embedding word similarity with neural machine translation. *arXiv preprint arXiv:1412.6448*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of ACL*, volume 1, pages 95–105.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 36–45.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of EMNLP*, pages 2044–2048.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jamie Kiros, William Chan, and Geoffrey Hinton. 2018. Illustrative language understanding: Large-scale visual grounding with image search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 922–933.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Tom Kocmi and Ondřej Bojar. 2017. An exploration of word embedding initialization in deep-learning tasks. *arXiv preprint arXiv:1711.09160*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. *arXiv preprint arXiv:1501.02598*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Jianqiang Li, Jing Li, Xianghua Fu, Md Abdul Masud, and Joshua Zhexue Huang. 2016. Learning distributed word representation with multi-contextual mixed embedding. *Knowledge-Based Systems*, 106:220–230.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proceedings of IJCAI*, pages 1284–1290.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. 2017. Deep learning with dynamic computation graphs. *arXiv preprint arXiv:1702.02181*.

Yong Luo, Jian Tang, Jun Yan, Chao Xu, and Zheng Chen. 2014. Pre-trained multi-view word embedding using two-side neural network. In *Proceedings of AAAI*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL*, pages 142–150.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Learning word vectors for 157 languages. In *Proceedings of LREC*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. *arXiv preprint arXiv:1408.6179*.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. *arXiv preprint arXiv:1606.01700*.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural semantic encoders. In *Proceedings of ACL*, volume 1, page 397.

Avo Muromägi, Kairit Sirts, and Sven Laur. 2017. Linear ensembles of word embedding models. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 96–104.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. *arXiv preprint arXiv:1708.02312*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Mohammad Taher Pilehvar, José Camacho-Collados, Roberto Navigli, and Nigel Collier. 2017. Towards a seamless integration of word senses into downstream NLP applications. *arXiv preprint arXiv:1710.06632*.

Lin Qiu, Kewei Tu, and Yong Yu. 2016. Context-dependent sense embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 183–191.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of EMNLP*, pages 298–307.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. *arXiv preprint arXiv:1801.10296*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of EMNLP*, pages 2049–2054.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Thuy Vu and D Stott Parker. 2016. k-embeddings: Learning conceptual embeddings for words using context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1262–1267.

Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. 2017. Hyperlex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 43(4):781–835.

Ivan Vulić and Nikola Mrkšić. 2017. Specialising word vectors for lexical entailment. *arXiv preprint arXiv:1710.06371*.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2018. Learning multimodal word representation via dynamic fusion methods. *arXiv preprint arXiv:1801.00532*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. 2016. Words or characters? fine-grained gating for reading comprehension. *arXiv preprint arXiv:1611.01724*.

Wenpeng Yin and Hinrich Schütze. 2015. Learning word meta-embeddings by using ensembles of embedding sets. *arXiv preprint arXiv:1508.04257*.

Liang-Chih Yu, Jin Wang, K Robert Lai, and Xuejie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 534–539.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of ICCV*, pages 19–27.

A Probabilistic Model for Joint Learning of Word Embeddings from Texts and Images

Melissa Ailem[1,2], **Bowen Zhang**[2], **Aurelien Bellet**[1], **Pascal Denis**[1] **and Fei Sha**[2,3]
[1] INRIA, France
[2] University of Southern California, Los Angeles, CA
[1] `{melissa.ailem,aurelien.bellet,pascal.denis}@inria.fr`
[2] `{zhan734, ailem, feisha}@usc.edu` [3] `fsha@netflix.com`*

Abstract

Several recent studies have shown the benefits of combining language and perception to infer word embeddings. These multimodal approaches either simply combine pre-trained textual and visual representations (e.g. features extracted from convolutional neural networks), or use the latter to bias the learning of textual word embeddings. In this work, we propose a novel probabilistic model to formalize how linguistic and perceptual inputs can work in concert to explain the observed word-context pairs in a text corpus. Our approach learns textual and visual representations jointly: latent visual factors couple together a skip-gram model for co-occurrence in linguistic data and a generative latent variable model for visual data. Extensive experimental studies validate the proposed model. Concretely, on the tasks of assessing pairwise word similarity and image/caption retrieval, our approach attains equally competitive or stronger results when compared to other state-of-the-art multimodal models.

1 Introduction

Continuous-valued vector representation of words has been one of the key components in neural architectures for natural language processing (Mikolov et al., 2013; Pennington et al., 2014; Levy and Goldberg, 2014). The main idea is based on the distributional hypothesis (Harris, 1954), which states that words used in similar contexts have similar semantic meanings. To this end, words are mapped to points in an Euclidean space such that the displacement between their coordinates (i.e., embeddings) reflects similarity and difference in semantics (Pennington et al., 2014). As such, word embeddings have been shown to be useful in determining semantic and syntactic similarity between individual words (Mikolov et al., 2013; Baroni et al., 2014; Levy et al., 2015), as well as in downstream NLP tasks, e.g., sentiment analysis, question answering, and coreference resolution, just to name a few.

Most existing approaches rely solely on text corpora to infer word representations. While successful, the embeddings produced by such models do not necessarily reflect all inherent aspects of human semantic knowledge, such as the perceptual aspect (Feng and Lapata, 2010). This has motivated many researchers to explore different ways to infuse visual information, often represented in the form of pre-computed visual features, into word embeddings (Kiela and Bottou, 2014; Silberer et al., 2017; Collell et al., 2017; Lazaridou et al., 2015). The main theme is to take either the text embeddings, or the visual features or both *as such* to derive multimodal embeddings: through concatenation (Kiela and Bottou, 2014), or by treating visual features as regression targets (Lazaridou et al., 2015; Collell et al., 2017).

Despite the success of these prior efforts in yielding multimodal embeddings and applying them to downstream NLP tasks, there are still several deficiencies. In particular, the visual features (as such) are not guaranteed to be suitable for the word embedding task since they are typically optimized independently for another objective (e.g., image classification). Hence, fusing *pre-computed* word representations and visual features may not be a good strategy.

To address the above issues, we explore a new way to integrate linguistic and perceptual information. We develop a new model which jointly learns word embeddings from text and extracts latent visual information, from pre-computed visual features, that could supplement the linguistic embeddings in modeling the co-occurrence of words

---

[*] On leave from U of Southern California

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1478–1487
Brussels, Belgium, October 31 - November 4, 2018.

and their contexts in a corpus. Instead of using pre-trained visual features as it is or as regression targets, we posit that they contain latent perceptual information that could complement text in representing words.

More specifically, the proposed model consists of two components. The visual component is an unsupervised probabilistic model for learning latent factors that generates the visual data. The linguistic component is a revised SKIP-GRAM model in which the text embeddings work *in concert* with the latent visual factors to explain the occurrence of word-context pairs in a corpus. One advantage of our joint modeling is that it allows two-way interaction. On one hand, the linguistic information can guide the extraction of latent visual factors. On the other hand, the extracted visual factors can improve the modeling of word-context co-occurrences in text data. Another appealing property of our model is its natural ability to propagate perceptual information to the embeddings of words lacking visual features (e.g., abstract words) during learning.

We conduct extensive quantitative and qualitative experiments to examine and understand the effectiveness of our approach, on the tasks of word similarity and image/caption retrieval. We show its matching or stronger performance when compared to other state-of-the-art approaches for learning multimodal embeddings.

2 Our Approach

We start by introducing the problem setup and notations. We then describe our model, namely PIXIE (ProbabIlistic teXtual Image Embeddings), for joint learning of word representations from text and perceptual information.

2.1 Setup and Background

We are given a corpus of H tokens (words): $w_1, \ldots, w_i, \ldots, w_H$. From the corpus, we form a collection of word-context pairs $\delta_{w,c} = (w, c)$, such that $w \in V_w$, $c \in V_c$, with V_w and V_c denoting respectively the word and context vocabularies. As in most previous work, the contexts for word w_i are the words that surround it in a L-sized window. We introduce the binary indicator variables y_{wc}, such that $y_{wc} = 1$ if $\delta_{w,c}$ appears in our collection, and $y_{wc} = 0$ otherwise.

For some words with visual grounding (we will refer to them as *visual words*), we have access to a visual representation $\boldsymbol{x}_w$. In practice, we use convolutional net features (see Section 4 for details).

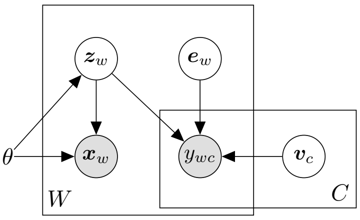

Figure 1: Plate representation of our model PIXIE. The model consists of a generative model for visual data and a conditional model for text data. The latent visual factors $\boldsymbol{z}$ and text embedding $\boldsymbol{e}$ jointly predict the word-context pair's label y.

SKIP-GRAM WITH NEGATIVE SAMPLING (SGNS) The SGNS's objective is to learn word representations that are good at distinguishing the observed pairs ($y_{wc} = 1$) from non-observed or "negative" pairs ($y_{wc} = 0$), using logistic regression. Formally, SGNS maximizes the following log-likelihood:

$$\sum_{w,c} [y_{wc} \log \sigma(\boldsymbol{v}_c^\top \boldsymbol{e}_w) + (1 - y_{wc}) \log \sigma(-\boldsymbol{v}_c^\top \boldsymbol{e}_w)], \quad (1)$$

where $\sigma(\cdot)$ is the sigmoid function, $\boldsymbol{v}_c$, $\boldsymbol{e}_w$ denote respectively the vectors for the context c and target word w. The second term in (1) is intractable due to the large number of possible negative pairs, and is approximated by sampling N negative examples $\{c'_i\}_{i=1}^N$ for every observed pair of words and their contexts. This gives rise to the following objective function for each observed pair:

$$\log \sigma(\boldsymbol{v}_c^\top \boldsymbol{e}_w) + \sum_{i=1}^{N} \log \sigma(-\boldsymbol{v}_{c'_i}^\top \boldsymbol{e}_w), \qquad (2)$$

where c'_i is a (negative) context that does not appear in the context of w (Mikolov et al., 2013). In practice, criterion (2) is optimized in an online fashion, by using Stochastic Gradient Descent (SGD) over the observed pairs δ_{wc} in the corpus. Each observed pair δ_{wc} typically occurs several times in the corpus, therefore performing SGD over the corpus amounts to weighting equation (2) by the number of occurrence of each pair.

2.2 Joint Visual and Text Modeling

We now describe our model, namely PIXIE (ProbabIlistic teXtual Image Embeddings) illustrated in Fig. 1, for joint learning of word representations from textual and perceptual information.

Formally, PIXIE is a probabilistic model of image features $\boldsymbol{x}$ and word-context pairs' labels $\boldsymbol{y}$. Similar to SKIP-GRAM, PIXIE represents each word w and context c with low dimensional embeddings noted respectively $\boldsymbol{e}_w \in \mathbb{R}^K$ and $\boldsymbol{v}_c \in \mathbb{R}^K$. PIXIE further assumes latent visual factors, $\boldsymbol{z}_w \in \mathbb{R}^K$, for each word's visual representation $\boldsymbol{x}_w$. Next we describe the two main components of PIXIE, namely *textual* and *perceptual*, in more details.

Perceptual Component Each visual vector $\boldsymbol{x}_w$ is drawn conditional on its latent representation $\boldsymbol{z}_w$, i.e., $\boldsymbol{x}_w \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_w)$, with $p_{\boldsymbol{\theta}}(\boldsymbol{z}) = \mathcal{N}(0, \mathbf{I})$. Since $\boldsymbol{x}_w$ is real valued, we let $p_{\boldsymbol{\theta}}(\boldsymbol{x}_w|\boldsymbol{z}_w)$ be a Gaussian parameterized by a *generative neural network* (or decoder). That is,

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_w|\boldsymbol{z}_w) = \mathcal{N}(\boldsymbol{x}_w|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}_w), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z}_w)). \quad (3)$$

For tractability purposes, $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ is restricted to be diagonal. Moreover, both the co-variance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z}_w)$ (its diagonal) and the mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ are the outputs of a decoder network with parameters $\boldsymbol{\theta}$ and input $\boldsymbol{z}_w$.

Textual Component To model the occurrence/absence of word-context pair δ_{wc} in the linguistic corpus, we adopt a Bernoulli (Ber) model:

$$p(y_{wc}|\,\boldsymbol{e}_w, \boldsymbol{v}_c, \boldsymbol{z}_w) = \text{Ber}(\sigma[f(\boldsymbol{e}_w, \boldsymbol{v}_c, \boldsymbol{z}_w)]) \quad (4)$$

The function $f(\cdot)$ defines how multimodal embeddings are fused. While many choices can be experimented, we use the simple additive model:

$$f(\boldsymbol{e}_w, \boldsymbol{v}_c, \boldsymbol{z}_w) = \boldsymbol{v}_c^{\mathsf{T}}(\boldsymbol{e}_w + \boldsymbol{z}_w). \quad (5)$$

For words without visual representation, we simply set the corresponding latent factors $\boldsymbol{z}_w$ to the zero vector. Note that, without the visual factors $\boldsymbol{z}_w$, equation (4) reduces to the Skip-Gram with negative sampling objective (1).

Joint Model The perceptual and the textual information interact through the shared latent $\boldsymbol{z}_w$. The joint model of the above two sources of information takes the following form:

$$p(\boldsymbol{x}_w, y_{wc}|\,\boldsymbol{e}_w, \boldsymbol{v}_c) = \int p_{\boldsymbol{\theta}}(\boldsymbol{z}_w) p_{\boldsymbol{\theta}}(\boldsymbol{x}_w|\boldsymbol{z}_w) p(y_{wc}|\,\boldsymbol{e}_w, \boldsymbol{v}_c, \boldsymbol{z}_w) d\boldsymbol{z}_w \quad (6)$$

The intuition behind our joint formulation is to let the textual information guide the extraction of latent visual factors $\boldsymbol{z}_w$. Through equations (4) and (5), the model will put high probability on factors $\boldsymbol{z}_w$ reflecting patterns that can supplement the linguistic embeddings $\boldsymbol{e}_w$ in explaining the word-context co-occurrences. Thus, the extracted latent visual factors can contribute to improve the performance on predicting the occurrence of a word and its contexts in the linguistic corpus, which would encourage the model to leverage the perceptual information. The underlying assumption here is to infer visual and textual embeddings that can work in concert to represent words.

Visual Information Propagation Equation (5) implies that the embeddings $\boldsymbol{z}$, $\boldsymbol{e}$ and $\boldsymbol{v}$ will affect each other during the learning process. Interestingly, if a non-visual word w_1 shares a similar context c with a visual word w_2, then the factor $\boldsymbol{z}_{w_2}$ will affect $\boldsymbol{e}_{w_1}$ via $\boldsymbol{v}_c$. In other words, our formulation makes it possible to implicitly propagate perceptual information from one word to another through shared contexts. We illustrate this aspect in our experiments.

2.3 Approximate Inference and Learning

Training PIXIE amounts to inferring the posterior over the visual latent factors, $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}, y)$, as well as finding the decoder's parameters $\boldsymbol{\theta}$, the word and context embeddings, $\boldsymbol{e}$ and $\boldsymbol{v}$, that maximize the likelihood (6). However, as in many complex probabilistic models, the likelihood (due to the integral over $\boldsymbol{z}$) and the posterior are intractable. We therefore resort to approximation techniques. More precisely, we rely on Variational Inference (VI) (Blei et al., 2017). The idea of VI is to introduce a tractable approximate posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ (the variational distribution) and optimize a lower bound on the likelihood, known as Evidence Lower BOund (ELBO). The latter can be written for each word w as follows:

$$\mathcal{L}_w = \mathbb{E}_q[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_w|\boldsymbol{z}_w) + \sum_c \log p(y_{wc}|\boldsymbol{e}_w, \boldsymbol{v}_c, \boldsymbol{z}_w)] - \text{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}_w|\boldsymbol{x}_w)\|p_{\boldsymbol{\theta}}(\boldsymbol{z}_w)) \quad (7)$$

where $\text{KL}(\cdot\|\cdot)$ is the Kullback-Leibler divergence. The variational distribution is chosen to be a multivariate Gaussian parameterized by an *inference network* (or encoder) which takes $\boldsymbol{x}$ as input, namely

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}_w|\boldsymbol{x}_w) = \mathcal{N}(\boldsymbol{z}_w|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}_w), \boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}_w)), \quad (8)$$

where we drop the dependency on all y_{wc} variables to be computationally tractable. The pair of encoder and decoder neural networks gives rise to the interpretation of PIXIE's visual component (formed by $\boldsymbol{z}$ and $\boldsymbol{x}$) as a probabilistic autoencoder. In fact, if we drop the textual part in PIXIE, namely y, $\boldsymbol{e}$ and $\boldsymbol{v}$, then we recover the Variational Auto-Encoder (VAE) (Kingma and Welling, 2013).

Lastly, we approximate the intractable (i) expectation with respect to $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and the (ii) sum over the negative pairs in (7), by relying on a Monte Carlo estimator of $\mathcal{L}$. Concretely, for (ii) we use negative sampling as in (2). Concerning (i), for every observed $\boldsymbol{x}_w$, we sample $\{\boldsymbol{z}_w^{(j)}\}_{j=1}^J$ from $q_{\boldsymbol{\phi}}(\boldsymbol{z}_w|\boldsymbol{x}_w)$ using the *reparameterization trick* (Kingma and Welling, 2013), i.e., $\boldsymbol{z}_w^{(j)} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}_w) + \boldsymbol{\Sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}_w)\boldsymbol{\epsilon}$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. Then we approximate $\mathcal{L}$ with:

$$\begin{aligned}
&\widetilde{\mathcal{L}}_w = \\
&\tfrac{1}{J}\sum_j \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_w|\boldsymbol{z}_w^{(j)}) - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}_w|\boldsymbol{x}_w)\|p_{\boldsymbol{\theta}}(\boldsymbol{z}_w)) \\
&+ \tfrac{1}{J}\sum_{c,j} y_{wc} \log \sigma[\boldsymbol{v}_c^{\mathsf{T}}(\boldsymbol{e}_w + \boldsymbol{z}_w^{(j)})] \\
&+ \tfrac{1}{J}\sum_{c,j}\sum_{i=1}^{N} y_{wc} \log(\sigma[-\boldsymbol{v}_{c'_i}^{\mathsf{T}}(\boldsymbol{e}_w + \boldsymbol{z}_w^{(j)})]). \qquad (9)
\end{aligned}$$

The last two summands correspond to the familiar conditional likelihood term in the SGNS model, augmented with latent visual factors.

We optimize the objective (9) via SGD with respect to both the encoder/decoder networks parameters ($\boldsymbol{\theta}$ and $\boldsymbol{\phi}$) and the embeddings ($\boldsymbol{e}$ and $\boldsymbol{v}$). We evaluate the gradients of $\widetilde{\mathcal{L}}$, with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ using backpropagation. Similarly, the gradient with respect to $\boldsymbol{e}$ and $\boldsymbol{v}$ can be easily carried out using automatic differentiation tools. Our learning procedure is summarized in Algorithm 1.

Algorithm 1 Variational PIXIE

```
Input: $\boldsymbol{x}$, $y$, sample sizes B and J
Steps:
Randomly initialize $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{e}$ and $\boldsymbol{v}$
repeat
    • Draw a minibatch $W^B$ of words: $w^{(1)}, \ldots, w^{(B)}$
    • For each $\boldsymbol{x}_w$ with $w \in W^B$, sample $\{\boldsymbol{z}_w^{(j)}\}_{j=1}^J$ from
      $q_{\boldsymbol{\phi}}(\boldsymbol{z}_w|\boldsymbol{x}_w)$ using the reparameterization trick. For each
      observed pair $\delta_{wc}$ draw $N$ negative examples $\{c'_i\}_{i=1}^N$.
    • Compute the estimator $\widetilde{\mathcal{L}}_{W^B} \leftarrow \sum_{w \in W^B} \widetilde{\mathcal{L}}_w$
    • Compute the gradient: $\mathbf{G} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{e},\boldsymbol{v}} \widetilde{\mathcal{L}}_{W^B}$
    • Use $\mathbf{G}$ to update $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{e}$ and $\boldsymbol{v}$ (e.g., with ADAM)
until convergence
return $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{e}$ and $\boldsymbol{v}$
```

Inference Once the parameters of the model are learned, for any given word with or without visual representation, we can compute its multimodal embedding. As a short hand, let the binary variable m_w denote whether or not the word w has a visual representation. The multimodal embedding for w can be written as

$$\boldsymbol{s}_w = \boldsymbol{e}_w + m_w \boldsymbol{\mu}(\boldsymbol{x}_w), \qquad (10)$$

where $\boldsymbol{\mu}(\boldsymbol{x}_w) = \mathbb{E}_q(\boldsymbol{z}_w)$ is the output of the encoding neural network, cf. Eq. (8).

In our experiments, we have also studied an alternative way to compose multimodal embeddings by concatenating the two vectors $\boldsymbol{e}_w$ and $\boldsymbol{\mu}(\boldsymbol{x}_w)$

$$\boldsymbol{t}_w = [\boldsymbol{e}_w \;\; m_w \boldsymbol{\mu}(\boldsymbol{x}_w)]. \qquad (11)$$

Note that for non-visual words, only zeros are appended to $\boldsymbol{e}_w$. One advantage of $\boldsymbol{t}$ over $\boldsymbol{s}$ is that if one uses distances to measure similarity, $\boldsymbol{t}$ can be seen as a simple summation of distances in two different spaces (in terms of $\boldsymbol{e}$ and $\boldsymbol{\mu}$ respectively).

3 Related Work

Combining language and perception has been recently considered in various NLP tasks such as machine translation (Calixto and Liu, 2017), visual question generation (Mostafazadeh et al., 2016), image captioning (Klein et al., 2015), etc. In this work, we focus on learning word embeddings from images and texts.

Multimodal embeddings have been studied in several recent research work. One strategy is to obtain word embeddings from linguistic data and visual data *independently* and then proceed with some kind of fusion steps. Kiela and Bottou (2014) simply concatenates pre-trained linguistic word embeddings and visual features computed by convolutional nets. Bruni et al. (2014) performs an additional step of dimensionality reduction via singular value decomposition. Silberer et al. (2017) extend on this work by feeding the linguistic embedding and visual features into a stacked auto-encoder for nonlinear dimensionality reduction. The above-mentioned approaches perform a two-stage process to derive multimodal representations (unimodal inference followed by fusion) and have been evaluated only on words for which both perceptual and textual representations are available.

A standing question is how to propagate visual information from words with visual features to words lacking them (for instance, abstract words). While the previous methods fall short on that, the recent work by Collell et al. (2017) addresses this challenge by learning a mapping from language to vision, using a set of words with known linguistic embeddings and visual features. This mapping can then be used to infer visual representations for new words from their textual embeddings.

All the aforementioned methods rely on independently pre-trained linguistic embeddings and visual features. In this work, we propose a different strategy, which consists in adapting those representations so that the information can be fused in earlier stages. In this respect, the closest work to ours is (Lazaridou et al., 2015), which proposes to augment the SKIP-GRAM objective function with a term mapping the textual embeddings to the visual features. Crudely, the linguistic embeddings must therefore predict both the text co-occurrences and (pre-trained) visual features. We emphasize two key differences with our approach. First, instead of performing a regression or mapping from the textual embeddings to the visual features, our model learns to infer perceptual latent factors to retain only the portion of visual information that can supplement the linguistic embeddings in representing words. Second, while Lazaridou et al. (2015) combines two objectives, we use a joint probabilistic model integrating both visual and text information in a principled way. Specifically, our model seeks latent factors that are good at explaining the word-context co-occurrences. For instance, a visual feature of (an image of) OCEAN often contains information about SKY and BLUE — such visual information could be beneficial to predict co-occurrence of tokens in the context of OCEAN. This desiderata further strengthens the learned embeddings to be visually grounded. In our experiments, we show that our approach tends to group concrete visually similar concepts together.

4 Experiments

In this section, we evaluate our model and contrast it to other competing approaches on two different tasks: word similarity and image/caption retrieval.

4.1 Setup

Text corpus We use the Text8 WIKIPEDIA corpus[1] containing over 17 million tokens. Text8 was pre-processed to contain only letters and nonconsecutive spaces. After removing infrequent words, we obtain a vocabulary of $50,000$ unique words.

Image features We use the ImageNet dataset (Russakovsky et al., 2015), including the fall ImageNet 2011 release (Deng et al., 2009). It contains $14,188,125$ images organized according to $21,842$ synsets of WordNet (Fellbaum, 1998). Each synset contains 600 images on average. To extract image features, we rely on the Caffe toolkit (Jia et al., 2014) and use the GoogLeNet convolutional neural nets (Szegedy et al., 2015) pre-trained on the 1000 synsets of ILSVRC 2012. The 1024-dimensional activation of the pooling units (before the softmax layer) are then taken as our image features.

Visual representation of words For each word in the vocabulary, we recover all the synsets that it belongs to using the WordNet interface of the NLTK module (Python) (Bird et al., 2009). We then remove the synsets not covered by our ImageNet dataset. This results in 9,713 words, out of the 50,000 words in the vocabulary. For each visual word, we randomly draw 1,000 distinct images in ImageNet. If the number of images for a word is less than 1,000, we increase the coverage using images belonging to the hypernyms of the considered word's synsets, as in (Kiela and Bottou, 2014). We then take the average of these features as the word's visual representation $\boldsymbol{x}$.

Hyper-parameter setting For all models, we set the dimension of linguistic and visual embeddings, $\boldsymbol{e}$ and $\boldsymbol{z}$, to 100, following many previous works. In our model, the encoder/decoder neural networks are implemented as one-hidden-layer neural nets with 500 hidden units each. The dimensions of the inputs and the outputs of the decoder neural networks are 100 and $1,024$ respectively ($1,024$ and 100 for the encoder). For the encoder, the hidden units are hyperbolic tangent, and the output units are linear. For the decoder, the hidden units are hyperbolic while the outputs are sigmoid. For SGNS, we set the window size L to 10 and the number of negative samples to 64. Our model is learned by Stochastic Gradient Descent using the ADAM optimizer (Kingma and Ba, 2014) with a learning rate set to 0.001.

Table 1: Datasets for the task of word similarity.

Datasets	#word pairs
MEN (Bruni et al., 2014)	3000
EN-MC (Miller and Charles, 1991)	31
EN-RG (Rubenstein and Goodenough, 1965)	65
SimLex (Hill et al., 2015)	999
MTurk (Radinsky et al., 2011; Halawi et al., 2012)	287
WORDSIM (Finkelstein et al., 2001)	350
REL (Agirre et al., 2009)	150
SIM (Agirre et al., 2009)	200
SEMSIM (Silberer and Lapata, 2014)	5494
VISSIM (Silberer and Lapata, 2014)	5494

---

[1] http://mattmahoney.net/dc/textdata

Table 2: Results on word similarity task. Reported are the Spearman's rank order correlation between model prediction and human judgment (higher is better and bolds highlight the best methods). See text for details.

	Semantic/taxonomic similarity										General relatedness						Visual similarity		REL+SIM	
Models	SEMSIM		SimLex		SIM		EN-RG		EN-MC		MEN		REL		MTurk		VISSIM		WORDSIM	
	100%	98%	100%	39%	100%	44%	100%	72%	100%	73%	100%	54%	100%	53%	100%	26%	100%	98%	100%	39%
CNN	-	0.49	-	0.41	-	0.49	-	0.54	-	0.46	-	0.54	-	0.20	-	0.18	-	0.53	-	0.28
VAE	-	0.65	-	0.43	-	0.51	-	0.56	-	0.55	-	0.62	-	0.22	-	0.40	-	0.62	-	0.37
SGNS	0.50	0.50	0.33	0.35	0.66	0.66	0.60	0.55	0.60	0.52	0.65	0.67	**0.56**	0.51	**0.65**	0.63	0.38	0.38	**0.61**	0.60
CNN⊕SGNS†	-	0.67	-	0.48	-	0.65	-	0.60	-	0.55	-	0.74	-	0.44	-	0.51	-	**0.63**	-	0.56
VAE⊕SGNS	-	0.70	-	0.51	-	0.67	-	0.61	-	0.60	-	**0.76**	-	0.45	-	0.55	-	**0.63**	-	0.56
V-SGNS‡	0.58	0.58	0.29	0.30	0.66	0.71	**0.73**	**0.73**	**0.69**	0.69	0.64	0.65	0.51	0.52	0.60	0.65	0.42	0.42	0.59	0.64
IV-SGNS§ (LINEAR)	0.49	0.50	0.31	0.33	0.55	0.61	0.58	0.56	0.59	0.65	0.60	0.62	0.41	0.38	0.57	**0.71**	0.36	0.37	0.46	0.51
IV-SGNS§ (NONLINEAR)	0.44	0.44	0.30	0.32	0.53	0.59	0.54	0.53	0.59	0.63	0.57	0.59	0.40	0.37	0.56	**0.71**	0.32	0.33	0.44	0.48
PIXIE$_{+}$	0.63	0.63	0.35	0.48	0.63	**0.72**	0.65	0.60	0.62	0.62	0.64	0.73	0.46	0.56	0.55	0.55	0.54	0.54	0.50	0.59
PIXIE$_{\oplus}$	**0.71**	**0.71**	**0.39**	**0.53**	**0.68**	0.71	**0.73**	**0.73**	**0.69**	**0.71**	**0.68**	**0.76**	0.52	**0.59**	0.60	0.59	0.60	0.61	0.58	**0.65**

†:(Kiela and Bottou, 2014), ‡: (Lazaridou et al., 2015), §: (Collell et al., 2017)

4.2 Task 1: Word Similarity

Datasets Word similarity is a common type of evaluation task for measuring the effectiveness of word embeddings. To this end, we retain 10 benchmark datasets consisting of pairs of words associated with similarity scores given by human judges. Table 1 summarizes their basic properties. There are different types of similarities being assessed. SEMSIM, SimLex, SIM, EN-RG and EN-MC focus on *semantic or taxonomic similarity* — e.g. CAR is similar to AUTOMOBILE. MEN, REL and MTurk consider *general relatedness* — e.g. CAR is related to GARAGE. VISSIM is about *visual similarity* — e.g. GOOSE looks like SWAN. Note that SIM and REL are the similarity and relatedness subsets of the full WORDSIM dataset (Finkelstein et al., 2001) respectively. VISSIM contains the same word pairs as SEMSIM.

Competing models We benchmark our model PIXIE against several strong uni- and multi-modal models listed below:

- SGNS: Skip-Gram with Negative Sampling (Mikolov et al., 2013). Without the visual component, PIXIE reduces to SGNS. We can thus assess the impact of the perceptual information by comparing PIXIE to SGNS.
- VAE: Vatiational Auto-Encoder (Kingma and Welling, 2013), which corresponds to the visual-specific component of PIXIE.
- CNN: Visual features extracted from a convolutional neural net as described in Section 4.1.
- CNN⊕SGNS (Kiela and Bottou, 2014): Concatenation of CNN and SKIP-GRAM embeddings.
- VAE⊕SGNS: Concatenation of VAE and SKIP-GRAM embeddings.
- V-SGNS (Lazaridou et al., 2015): A multimodal approach which augments SGNS with a term that treats CNN visual features as regression targets. Comparisons with V-SGNS will allow us to evaluate the impact of our modeling assumptions.
- IV-SGNS (Collell et al., 2017): Learns a mapping from SGNS embeddings to CNN visual features.

Due to a large degree of discrepancies in experimental setups across previously published methods and results,[2] we re-implemented all the baselines and evaluate them under the same conditions.[3] For (Lazaridou et al., 2015), we implemented its model "A" as model "B" is comparable according to the original authors. For (Collell et al., 2017), we implemented both linear and nonlinear variants.

Evaluation metrics We use the cosine to measure the similarity between word representations. To assess the coherence between human ratings and models' predictions, we use the Spearman correlation coefficient.

4.2.1 Main results

The results across different datasets are shown in Table 2. We perform evaluations under two settings: by considering (i) word similarity between visual words only and (ii) between all words (column 100% in Table 2). For the models CNN, VAE and their concatenation with SGNS embeddings, the latter setting is not applicable. The two last rows correspond to the multimodal embeddings inferred from our model. In particular, PIXIE$_{+}$

---

[2] For instance, Lazaridou et al. (2015) reports only 5,100 visual words, nearly half of what we have defined. Collell et al. (2017) used pre-trained GloVe word vectors obtained from a different corpus.

[3] For each method we use the hyper-parameters recommended by the authors.

(resp. PIXIE$_{\oplus}$) represents the multimodal embeddings built using Eq. (10) (resp. Eq. (11)).

Overall, we note that PIXIE$_{\oplus}$ offers the best performance in almost all situations. This provides strong empirical support for the proposed model. Below, we discuss the above results in more depth to better understand them and characterize the circumstances in which our model performs better.

How relevant is our formulation? Except PIXIE and V-SGNS, most of the multimodal competing methods rely on independently pre-computed linguistic embeddings. As Table 2 shows, PIXIE and V-SGNS are often the best performing multimodal models, which provides empirical evidence that accounting for perceptual information while learning word embeddings from text is beneficial. Moreover, the superior performance of PIXIE$_{\oplus}$ over V-SGNS suggests that our model does a better job at combining perception and language to learn word representations.

Joint learning is beneficial PIXIE$_{\oplus}$ outperforms VAE⊕SGNS in almost all cases, which demonstrates the importance of joint learning.

Where does our approach perform better? On datasets that focus on *semantic/taxonomic similarity*, our approach dominates all other methods.

On datasets focusing on *general relatedness*, our approach obtains mixed results. While dominating other approaches on MEN, it tends to perform worst than SGNS on MTurk and REL (under the 100% setting). One possible explanation is that general relatedness tends to focus more on "extrapolating" from one word to another word (such as SWAN is related to LAKE), while our approach better models more concrete relationships (such as SWAN is related to GOOSE). The low performance of CNN and VAE confirms this hypothesis.

On the VISSIM dataset focusing on *visual similarity*, both CNN⊕SGNS and VAE⊕SGNS perform the best, strongly suggesting that visual and linguistic data are complementary. Our approach comes very close to these two methods. Note that our learning objective is to jointly explain visual features and word-context co-occurrences. Thus, two visually similar words, which never occur within the same context, could be mapped into slightly different directions in the latent space.

Visual Propagation Here we wish to evaluate the ability of our model to propagate perceptual information to words lacking visual features. To this end, we randomly select a subset of $2,000$ words for which we have visual features, and we train our model under two different settings: the visual features of the selected 2K words (i) are taken into account (PIXIE$_{\oplus}$), (ii) are ignored, i.e. set to zero (PIXIE$_{\oplus}^{(*)}$). We then perform evaluations, under the two settings, on the datasets of Table 1 considering only pairs composed of words in the above subset of 2K words. As baselines for this experiment, we consider SGNS and the multimodal approaches which can propagate perceptual information, namely V-SGNS and IV-SGNS, as well as their outputs when the 2K visual features are ignored (denoted by V-SGNS$^{(*)}$, and IV-SGNS$^{(*)}$).

Table 3: Spearman's score on subsets of visual words. The symbol $(*)$ indicates the visual features for the subset of 2K words have been ignored when training. Bold highlights the best performing method. Blue color highlights the best performing model under the $(*)$ setting.

	Semantic/taxonomic					Relatedness			Visual	REL +SIM
models	SEMSIM	SimLex	SIM	EN-RG	EN-MC	MEN	REL	MTurk	VISSIM	WORDSIM
SGNS	0.55	0.40	0.67	0.57	0.52	0.68	0.53	0.75	0.41	0.63
V-SGNS	0.61	0.32	**0.72**	0.71	0.69	0.69	0.52	0.75	0.44	**0.66**
IV-SGNS (LINEAR)	0.50	0.40	0.60	0.59	0.60	0.66	0.41	0.72	0.37	0.54
PIXIE$_{\oplus}$	**0.71**	**0.59**	**0.72**	**0.73**	**0.71**	**0.77**	**0.57**	0.74	**0.60**	0.62
V-SGNS$^{(*)}$	0.58	0.29	**0.72**	0.68	0.61	0.67	0.44	**0.76**	0.43	0.60
IV-SGNS$^{(*)}$	0.50	0.40	0.61	0.58	0.58	0.65	0.40	0.72	0.37	0.54
PIXIE$_{\oplus}^{(*)}$	0.60	0.43	**0.72**	0.65	0.60	0.70	0.56	**0.76**	0.45	0.63

The results are given in Table 3. We observe that PIXIE$_{\oplus}^{(*)}$ outperforms SGNS in almost all cases. Recall that, if we ignore the visual features for all words, PIXIE reduces to SGNS. We can therefore attribute the performance improvement of PIXIE$_{\oplus}^{(*)}$ over SGNS to the propagation of visual information to the subset of 2K words. Compared to multimodal methods, PIXIE$_{\oplus}^{(*)}$ (resp. PIXIE$_{\oplus}$) performs better than V-SGNS$^{(*)}$ (resp. V-SGNS) and IV-SGNS$^{(*)}$ (resp. IV-SGNS) in almost situations. This suggests that our formulation allows perceptual information to propagate better.

Table 4: Word pair cosine similarity computed based on SGNS and PIXIE$_{\oplus}^{(*)}$ embeddings.

Word pairs	SGNS	PIXIE$_{\oplus}^{(*)}$
(chicken, turkey)	0.35	0.55
(helicopter, jet)	0.63	0.76
(falcon, hawk)	0.49	0.70
(cathedral, chapel)	0.69	0.80
(cup, mug)	0.39	0.46

Table 5: 10 nearest neighboring words to the query words in different embedding spaces generated by different methods. Only "visual" words contain direct visual representations in our dataset. The concreteness score of each query word is reported between parenthesis (see text for details).

	Query word	SGNS	V-SGNS	PIXIE$_{\oplus}$
Visual	goose (4.81)	quail, pig, shark, gull, smoky, sooty, bald, owl, guppy, bird	fowl, quail, duck, bat, bulldog, puppy, warbler, blossom, wolfhound, crows	geese, duck, swans, swan, teal, loon, albatross, ostrich, gull, eider
	brave (1.26)	courageous, young, man, fearless, heroic, thief, horrible, adventures, carefree, cowardly	heroes, valiant, ominous, wondrous, fearless, wanton, sabers, beast, excalibur, carefree	bodyguard, wives, benefactors, womanizer, heroes, valiant, immortal, housewives, fearless, warrior
Non Visual	birthstone (4.25)	heaviest, yeti, koala, snowfall, intrusions, mourning, amalthea, gleaming, incidentally, dolly	emerald, lily, lavender, earthy, olive, delicacy, acacia, belladonna, flower, poppy	beryl, emerald, lily, sandalwood, hula, guinevere, pearls, holly, jasmine, jewels
	savagery (1.73)	swamp, man, crazy, madman, mysterious, thief, jeffrey, rage, mad, hardy	mad, zombies, beast, fabulous, mysterious, nightmare, ghosts, alien, mayhem, bandits	cannibals, evil, legends, werewolves, beast, haunting, ghosts, zombies, mayhem, thrillers

In Table 4, we report the cosine similarity between 5 semantically/visually coherent word pairs (from our subset of 2K words). Although the visual vectors of these words were removed during training, the PIXIE's word embeddings of each pair correlate better as compared to their SGNS counterparts, which provides further support to the propagation of visual information under PIXIE.

4.2.2 Qualitative analysis

Table 5 displays several qualitative examples of word similarity. We have selected 4 words: *goose*, *brave*, *birthstone* and *savagery*. The first two have visual feature representations in our training dataset and the last two do not. Furthermore, for each case, we chose one concrete and one abstract word.[4] For each word, we identify their nearest neighbors in the embedding space.

For the visual words, there is a noticeable difference between our method and others. For instance, for word *goose*, SGNS expresses more "general" relatedness and returns other animals like *pig* or *shark*, while our approach is more specific and tends to give visually similar neighbors by focusing on *goose* looks-like birds. V-SGNS's result is somewhat in between. On the abstract word *brave*, we observe that PIXIE$_{\oplus}$ tends to select more explicit embodiments of the adjective *brave* than SGNS and V-SGNS.

Moving towards the non-visual words, we do not seem to find a consistent discrepancy pattern between V-SGNS and PIXIE$_{\oplus}$, though, as for visual words, both methods seem to select more explicit exemplars compared to SGNS. For instance, for the abstract word *savagery*, both multimodal approaches suggest *cannibals* and *zombies*.

Table 6: Results for image (I) ↔ sentence (S) retrieval.

Models	I → S			S → I		
	K=1	K=5	K=10	K=1	K=5	K=10
SGNS	23.1	49.0	61.6	16.6	41.0	53.8
V-SGNS	21.9	51.7	64.2	16.2	42.0	54.8
IV-SGNS (LINEAR)	22.7	50.5	61.7	17.1	42.6	55.4
PIXIE$_{+}$	24.2	52.5	65.4	17.5	43.8	56.2
PIXIE$_{\oplus}$	**25.7**	**55.7**	**67.7**	**18.4**	**44.9**	**56.9**

4.3 Task 2: Image and Caption Retrieval

We now study the usefulness of the learned word embeddings for the tasks of image and caption retrieval. Our hypothesis is that multimodal word embeddings will perform better for downstream tasks involving multimodal information.

Experimental setup We use the Flickr30K dataset (Young et al., 2014) containing 31,000 images and 155,000 sentences (5 captions per image). The sentences describe the images. The task is to identify the best sentence describing an image or to identify the best image depicting a sentence. We follow the data split setting provided by Karpathy and Fei-Fei (2015), in which 1,000 images are used for validation and 1,000 for testing. The rest is used for training.

The retrieval models compute the proximity between the image features and the sentence embeddings. For image features, we use the precomputed features provided by Faghri et al. (2017), which are extracted from the FC7 layer of VGG-19 (Simonyan and Zisserman, 2014). These 4,096-dimensional features are then linearly mapped to 1,024-dimensional features. For sentences, we use an one GRU-layer over the sequences of the word embeddings, resulting in 1,024-dimensional sentence embeddings.

We use a triplet loss to train the retrieval model such that the inner product between the corresponding image feature and the sentence is greater than the inner products with incorrect sentences (or im-

---

[4] We rely on the concreteness ratings made available by Brysbaert et al. (2014), ranging from 1 to 5.

ages) (Kiros et al., 2014). The linear mapping and the GRU layer are then optimized to minimize the loss. We use the ADAM optimizer with the learning rate of 0.0002 and divide it by 10 every 15 epochs and we train the model for 30 epochs. Note that we do not *fine-tune* either the original visual feature or the word embeddings.

Results Table 6 summarizes the results. The evaluation metrics are accuracies at top-K (K=1, 5, or 10) retrieved sentences or images. Our model consistently outperforms SGNS and other competing multimodal methods, which provides additional support for the benefits of our approach.

5 Conclusion

We propose PIXIE, a novel probabilistic model joining textual and perceptual information to infer multimodal word embeddings. In our model, both linguistic and visual latent factors work in concert to explain the co-occurrences of words and their contexts in a corpus. Empirical results show that our model achieves equally competitive or stronger results when compared to state-of-the-art methods for multimodal embeddings.

Currently our model relies on unsupervised learning to infer visual factors. Explicit knowledge of similar and dissimilar visual categories could potentially disentangle latent factors better for alignment with linguistic data. How to incorporate visual domain knowledge more explicitly into the model would be an interesting direction for future research. While we build on skip-gram, the idea of PIXIE could be extended to other word embedding models, e.g., Glove (Pennington et al., 2014), ELMO (Peters et al., 2018), etc.

Acknowledgements

This work was partially supported by the virtual lab Inria@SiliconValley, ANR Grant GRASP No. ANR-16-CE33-0011-01, and a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020, NSF IIS-1065243, 1451412, 1513966/ 1632803/1833137, 1208500, CCF-1139148, a Google Research Award, an Alfred P. Sloan Research Fellowship, gifts from Facebook and Netflix, and ARO# W911NF-12-1-0241 and W911NF-15-1-0484. We thank anonymous reviewers for their suggestions and comments.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL HLT*, pages 19–27.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Ressearch*, 49(2014):1–47.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46(3):904–911.

Iacer Calixto and Qun Liu. 2017. Incorporating global visual features into attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, pages 992–1003.

Guillem Collell, Ted Zhang, and Marie-Francine Moens. 2017. Imagined Visual Representations as Multimodal Embeddings. In *AAAI*, pages 4378–4384.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. VSE++: Improving Visual-Semantic Embeddings with Hard Negatives. *arXiv preprint arXiv:1707.05612*.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 91–99. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*, pages 406–414.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *SIGKDD*, pages 1406–1414.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, pages 675–678.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137.

Douwe Kiela and Léon Bottou. 2014. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics. In *EMNLP*, pages 36–45.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. 2015. Associating neural word embeddings with deep image representations using fisher vectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4437–4446.

Angeliki Lazaridou, Marco Baroni, et al. 2015. Combining Language and Vision with a Multimodal Skip-gram Model. In *NAACL HLT*, pages 153–163.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *WWW*, pages 337–346.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2017. Visually grounded meaning representations. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2284–2297.

Carina Silberer and Mirella Lapata. 2014. Learning Grounded Meaning Representations with Autoencoders. In *ACL*, pages 721–732.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*, pages 1–9.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Transfer and Multi-Task Learning for Noun–Noun Compound Interpretation

Murhaf Fares **Stephan Oepen** **Erik Velldal**

Department of Informatics
University of Oslo
murhaff|oe|erikve@ifi.uio.no

Abstract

In this paper, we empirically evaluate the utility of transfer and multi-task learning on a challenging semantic classification task: semantic interpretation of noun–noun compounds. Through a comprehensive series of experiments and in-depth error analysis, we show that transfer learning via parameter initialization and multi-task learning via parameter sharing can help a neural classification model generalize over a highly skewed distribution of relations. Further, we demonstrate how dual annotation with two distinct sets of relations over the same set of compounds can be exploited to improve the overall accuracy of a neural classifier and its F_1 scores on the less frequent, but more difficult relations.

1 Introduction

Noun–noun compound interpretation is the task of assigning semantic relations to pairs of nouns (or more generally, pairs of noun phrases in the case of multi-word compounds). For example, given the nominal compound *street protest*, the task of compound interpretation is to predict the semantic relation holding between *street* and *protest* (a locative relation in this example). Given the frequency of noun–noun compounds in natural language – e.g. 3% of the tokens in the British National Corpus (Burnard, 2000) are part of noun–noun compounds (Ó Séaghdha, 2008) – and its relevance to other natural language processing (NLP) tasks such as question answering and information retrieval (Nakov, 2008), noun–noun compound interpretation has been the focus of much work, in theoretical linguistics (Li, 1972; Downing, 1977; Levi, 1978; Finin, 1980; Ryder, 1994), psycholinguistics (Gagné and Shoben, 1997; Marelli et al., 2017), and computational linguistics (Lauer, 1995; Nakov, 2007; Ó Séaghdha and Copestake, 2009; Girju et al., 2009; Kim and Baldwin, 2013; Dima and Hinrichs, 2015). In computational linguistics, noun–noun compound interpretation is, by and large, approached as an automatic classification problem. Hence several machine learning (ML) algorithms and models have been used to learn the semantics of nominal compounds, including Maximum Entropy (Tratz and Hovy, 2010), Support Vector Machines (Ó Séaghdha and Copestake, 2013) and Neural Networks (Dima and Hinrichs, 2015; Vered and Waterson, 2018). These models use information from lexical semantics such as WordNet-based features and distributional semantics such as word embeddings. Nonetheless, noun–noun compound interpretation remains one of the more difficult NLP problems because: 1) noun–noun compounding, as a linguistic construction, is very productive and 2) the semantics of noun–noun compounds is not easily derivable from the compounds' constituents (Vered and Waterson, 2018). Our work, in part, contributes to advancing NLP research on noun–noun compound interpretation through the use of transfer and multi-task learning.

The interest in using transfer learning (TL) and multi-task learning (MTL) in NLP has surged over the past few years, showing 'mixed' results depending on the so-called main and auxiliary tasks involved, model architectures and datasets, among other things (Collobert and Weston, 2008; Mou et al., 2016; Søgaard and Goldberg, 2016; Martínez Alonso and Plank, 2017; Bingel and Søgaard, 2017). These 'mixed' results, coupled with the fact that neither TL nor MTL has been applied to noun–noun compounds interpretation before, motivate our extensive empirical study on the use of TL and MTL for compound interpretation, not only to supplement existing research on the utility of TL and MTL for semantic NLP tasks in general, but also to determine their benefits for compound interpretation in particular.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1488–1498
Brussels, Belgium, October 31 - November 4, 2018.

One of the primary motivations for using multi-task learning is to improve generalization by "leveraging the domain-specific information contained in the training signals of *related* tasks" Caruana (1997). In this work, we show that TL and MTL can indeed be used as a kind of *regularizer* to learn to predict infrequent relations given a highly skewed distribution of relations from the noun–noun compound dataset of Fares (2016) which is especially well suited for TL and MTL experimentation as detailed in Section 3.

Our contributions can be summarized as:

1. Through careful result analysis, we find that TL and MTL (mainly on the embedding layer) do improve the overall accuracy and the F_1 scores of the less frequent relations in a highly skewed dataset, in comparison to a strong single-task learning baseline.

2. Even though our work focuses on TL and MTL, to the best of our knowledge, we are the first to report experimental results on the comparatively recent dataset of Fares (2016).

2 Related Work

Noun–Noun Compound Interpretation Existing approaches to noun–noun compound interpretation vary depending on the taxonomy of compound relations as well as the machine learning models and features used to learn those relations. For example, Ó Séaghdha (2007) defines a coarse-grained set of relations (viz. six relations based on theoretical work by Levi (1978)), whereas Tratz and Hovy (2010) assume a considerably more fine-grained taxonomy of 43 relations. Others question the very assumption that noun–noun compounds are interpretable using a finite, predefined set of relations (Downing, 1977; Finin, 1980) and propose alternative paraphrasing-based approaches (Nakov, 2007; Shwartz and Dagan, 2018). We here focus on the approaches that cast the interpretation problem as a classification task over a finite predefined set of relations. A wide variety of machine learning models have been already applied to learn this task, including nearest neighbor classifiers using semantic similarity based on lexical resources (Kim and Baldwin, 2005), kernel-based methods like SVMs using lexical and relational features (Ó Séaghdha and Copestake, 2009), Maximum Entropy models with a relatively large selection of lexical and surface form features such as synonyms and affixes (Tratz and Hovy, 2010) and, most recently, neural networks either solely relying on word embeddings to represent noun–noun compounds (Dima and Hinrichs, 2015) or word embeddings and so-called path embeddings (which encode information about lemmas and part-of-speech tags, inter alia) in a combined paraphrasing and classification approach (Vered and Waterson, 2018). Of the aforementioned studies, Tratz and Hovy (2010); Dima and Hinrichs (2015); Vered and Waterson (2018) have all used the same dataset by Tratz and Hovy (2010). To the best of our knowledge, TL and MTL have never been applied to compound interpretation before, and in the following we therefore review some of the previous work on TL and MTL on other NLP tasks.

Transfer and Multi-Task Learning A number of recent studies have presented comprehensive experiments on the use of TL and MTL for a variety of NLP tasks including named entity recognition and semantic labeling (Martínez Alonso and Plank, 2017), sentence-level sentiment classification (Mou et al., 2016), super-tagging and chunking (Bingel and Søgaard, 2017) and semantic dependency parsing (Peng et al., 2017). The common thread among the findings of these studies is that the benefits of TL and MTL largely depend on the properties of the tasks at hand, such as the skewedness of the data distribution (Martínez Alonso and Plank, 2017), the semantic similarity between the source and target tasks (Mou et al., 2016), the learning pattern of the auxiliary and main tasks where "target tasks that quickly plateau" benefit most from "non-plateauing auxiliary tasks" (Bingel and Søgaard, 2017) and the "structural similarity" between the tasks (Peng et al., 2017). In addition to the difference in the NLP tasks they experiment with, the aforementioned studies assume slightly different definitions of TL and MTL (cf. Section 4). Our work is similar in spirit to that of Peng et al. (2017) in the sense that we use TL and MTL to learn different 'formalisms' (semantic annotations of noun–noun compounds in our case) on the *same* dataset. However, our experimental setup is more similar to the work by Mou et al. (2016) in that we experiment with parameter initialization on all the layers of the neural model and simultaneously train one MTL model on two sets of relations (cf. Section 5).

3 Task Definition and Dataset

Given a set of labeled pairs of nouns, each a noun–noun compound, the task is simply to learn to classify the semantic relations holding between each pair of compound constituents. The difficulty of this task, obviously, depends on the label set used and its distribution, among other things. For all the experiments presented in this paper, we adapt the noun–noun compounds dataset created by Fares (2016) which consists of compounds annotated with two different taxonomies of relations; in other words, for each noun–noun compound there are two distinct relations, drawing on different linguistic schools. The dataset was derived from existing linguistic resources, such as `NomBank` (Meyers et al., 2004) and the Prague Czech-English Dependency Treebank 2.0 (Hajič et al., 2012, `PCEDT`). Our motivation for using this dataset is twofold: first, dual annotation with relations over the same underlying set of compounds maximally enables TL and MTL perspectives; second, alignment of two distinct annotation frameworks over the same data facilitates contrastive analysis and comparison across frameworks.

More specifically, we use a subset of the dataset created by Fares (2016), by focusing on type-based instances of so-called two-word compounds.[1] The original dataset by Fares (2016) also includes multi-word compounds (i.e. compounds consisting of more than two nouns) and more than just one instance per compound type. Furthermore, we define a three-way split of the dataset; Table 1 presents the number of compound types per split and the vocabulary size of each split (i.e. the number of unique words in each split); the latter is also broken down in terms of words occurring in the right-most position (right constituents) and the left-most position (left constituents).[2] Overall, the two label sets consists of 35 so-called `PCEDT` functors and 18 `NomBank` argument and adjunct relations. As detailed in Section 7.1, these label sets are far from being uniformly distributed.

	Train	Dev	Test
Compounds	6932	920	1759
Vocab size	4102	1163	1772
Right constituents	2304	624	969
Left constituents	2405	618	985

Table 1: Characteristics of the noun–noun compound dataset used in our experiments. The numbers in this table correspond to a (sub)set of the dataset by Fares (2016), see Section 3.

Abstractly, many relations in `PCEDT` and `NomBank` describe similar semantic concepts, since they annotate the semantics of the same text. For example, Fares (2016) reports that the temporal and locative relations in `NomBank` (`ARGM-TMP` and `ARGM-LOC`, respectively) and their counterparts in `PCEDT` (`TWHEN` and `LOC`) exhibit a relatively consistent behavior across frameworks as they annotate many of the same compounds. However, Fares (2016) also points out that some abstractly similar relations do not align well in practice; for example, the functor `AIM` in `PCEDT` and the modifier argument `ARGM-PNC` in `NomBank` express a somewhat similar semantic concept (purpose) but the overlap between the sets of compounds they annotate in practice is rather small. Nonetheless, it is plausible to assume that the semantic similarity in the label sets—whenever it exists—can be exploited in the form of transfer and multi-task learning, not least because the overall distribution of the relations in the two frameworks is different.

4 Transfer vs. Multi-Task Learning

In this section, we use the notations and definitions by Pan and Yang (2010) to define our setup for transfer and multi-task learning.

Our classification task $\mathcal{T}$ can be defined in terms of all training pairs (X, Y) and a probability distribution $P(X)$, where $X = x_i, \ldots, x_N \in \mathcal{X}$ and $Y = y_i, \ldots, y_N \in \mathcal{Y}$; $\mathcal{X}$ is the input feature space, $\mathcal{Y}$ is the set of all labels and N is the size of the training data. A task's domain $\mathcal{D}$ is defined by $\{\mathcal{X}, P(X)\}$. Our goal is to learn a function $f(X)$ that predicts Y based on the input features X. Assuming two ML tasks, $\mathcal{T}_a$ and $\mathcal{T}_b$, we would train two models (i.e. learn two separate functions f_a and f_b) to predict Y_a and Y_b in a single-task learning setup. However, if $\mathcal{T}_a$ and $\mathcal{T}_b$ are related somehow, either explicitly or implicitly, TL and MTL can improve the generalization of either task or both (Caruana, 1997; Pan and Yang, 2010; Mou et al., 2016). Two tasks are considered related

---

[1] Two-word compounds consist of two whitespace-separated constituents. A single constituent, however, can be a 'simple' noun (e.g. *system*) or a hyphenated noun (e.g. *land-ownership*) leading to compounds like *land-ownership system*. The representation of compounds with hyphenated constituents is explained in Section 5.1

[2] We use the terms left and right constituents, instead of modifier and head nouns, because the dataset does not make explicit the notion of 'headedness'.

when their domains, $\mathcal{D}_a$ and $\mathcal{D}_b$, are similar but their label sets are different $\mathcal{Y}_a \neq \mathcal{Y}_b$ or when their domains are different but their label sets are identical, i.e. $\mathcal{Y}_a = \mathcal{Y}_b$ (Pan and Yang, 2010).[3] As such, noun–noun compound interpretation over the dataset of Fares (2016) is a well suited candidate for TL and MTL, because the training examples are identical, i.e. $X_{PCEDT} = X_{NomBank}$, but the label sets are different $\mathcal{Y}_{PCEDT} \neq \mathcal{Y}_{NomBank}$.

For the sake of clarity, we distinguish between transfer learning and multi-task learning in this paper, even though these two terms are at times used somewhat interchangeably in the literature. For example, what we call TL and MTL in this paper are both referred to as transfer learning by Mou et al. (2016). We define TL as using the parameters (i.e. weights in neural networks) of one model trained on $\mathcal{T}_a$ to initialize another model for $\mathcal{T}_b$. Mou et al. (2016) refer to this method as "parameter initialization".[4] MTL, on the other hand, here refers to training (parts of) the same model to learn $\mathcal{T}_a$ and $\mathcal{T}_b$, i.e. learning one set of parameters for both tasks. The idea is to train a single model simultaneously on the two tasks where one task is considered to introduce inductive bias which would help the model generalize over the main task. Note, however, that this does not necessarily mean that we eventually want to use a single model to predict both label sets in practice (cf. Section 5.3).

5 Neural Classification Models

Here we present the neural classification models used in our experiments. To isolate the effect of TL and MTL, we first present a single-task learning model, which serves as our baseline model, and then we use the same model to apply TL and MTL.

5.1 Single-Task Learning Model

In our single-task learning (STL) setup, we train and fine-tune a feed-forward neural network based on the neural classifier proposed by Dima and Hinrichs (2015), which consists of: 1) input layer, 2) embedding layer, 3) hidden layer, and 4) output layer. The input layer is simply two integers specifying the indices of a compound's constituents in the embedding layer where the word embedding vectors are stored; the selected word embedding vectors are then fed to a fully connected hidden layer whose size is the same as the number of dimensions of the word embedding vectors. Finally, a *softmax* function is applied on the output layer and the most likely relation is selected.

The compound's constituents are represented using a 300-dimensional word embedding model trained on an English Wikipedia dump (dated February 2017) and English Gigaword Fifth Edition (Parker et al., 2011) using GloVe (Pennington et al., 2014). The embedding model was trained by Fares et al. (2017) who provide more details on the hyperparameters used to train the embedding model.[5] When looking up a word in the embedding model, if the word is not found we check if the word is uppercased and look up the same word in lowercase. If a word is hyphenated and is not found in the embedding vocabulary, we split it on the hyphen and average the vectors of its parts (if they exist in the vocabulary). If after these steps the word is still not found, we use a designated vector for unknown words.

Architecture and Hyperparameters Our choice of hyperparameters is motivated by several rounds of experimentation on the single-task learning model as well as the choices made by Dima and Hinrichs (2015).

The weights of the embedding layer (i.e. the word embeddings) are updated during training in all the models. The optimization function we use in all the models is Adaptive Moment Estimation, known as *Adam* (Kingma and Ba, 2015) with $\eta = 0.001$ (the default learning rate). The loss function is negative-log likelihood (aka categorical cross-entropy). We use a *Sigmoid* activation function on the hidden layer units. All the models are trained using mini-batches of size five. The maximum number of epochs is set to 50, but we also use an early stopping criterion on the model's accuracy on the validation split (i.e. training is interrupted if the validation accuracy doesn't improve over five consecutive epochs). We implement all the models in *Keras* with *TensorFlow* as backend. All the TL and MTL models are trained with the same hyperparameters of the STL model.[6]

---

[3] When the label sets are identical, TL practically becomes a technique for *domain adaptation*. Though these two terms have also been used interchangeably (Chung et al., 2018).

[4] Using *pretrained* word embeddings as input representation is in a sense a form of unsupervised transfer learning, but in this work we focus on transfer learning based on supervised learning.

[5] vectors.nlpl.eu/repository

[6] github.com/ltgoslo/fun-nom

5.2 Transfer Learning Models

Transfer learning in our experiments amounts to training an STL model on `PCEDT` relations, for example, and then using (some of) its weights to initialize another model for `NomBank` relations. Given the architecture of the neural classifier described in Section 5.1, we identify three ways to implement TL: 1) **TL$_{\mathbf{E}}$**: Transfer of the embedding layer weights, 2) **TL$_{\mathbf{H}}$**: Transfer of the hidden layer weights, and 3) **TL$_{\mathbf{EH}}$**: Transfer of both the embedding and hidden layer weights. Furthermore, we distinguish between transfer learning from `PCEDT` to `NomBank` and vice versa; that is, either task can be used as main task or auxiliary task. Hence, we either start by training on `NomBank` and use the weights of the corresponding transfer layer to initialize the `PCEDT` model or the other way around. In total, this leads to six setups, as shown in Table 2. Note that we do not apply TL (or MTL) on the output layer because it is task- or dataset-specific (Mou et al., 2016).

5.3 Multi-Task Learning Models

In MTL, we train one model simultaneously to learn both `PCEDT` and `NomBank` relations, and therefore all the MTL models have two objective functions and two output layers. We implement two MTL setups: **MTL$_{\mathbf{E}}$**, which consists of a shared embedding layer but two task-specific hidden layers, and **MTL$_{\mathbf{F}}$**, which, apart from the output layer, does not have task-specific layers, i.e. both the embedding and hidden layers are shared. We distinguish between the auxiliary and main tasks based on which validation accuracy (`NomBank`'s or `PCEDT`'s) is monitored by the early stopping criterion. Hence we end up with a total of four MTL models as shown in Table 3.

Model	`NomBank`		`PCEDT`	
	Dev	Test	Dev	Test
STL	78.15	76.75	58.80	56.05
TL$_{\text{E}}$	78.37	**78.05**	59.57	**57.42**
TL$_{\text{H}}$	78.15	78.00	59.24	56.51
TL$_{\text{EH}}$	**78.48**	78.00	**59.89**	56.68

Table 2: Accuracy (%) of the transfer learning models.

Model	`NomBank`		`PCEDT`	
	Dev	Test	Dev	Test
STL	**78.15**	76.75	58.80	56.05
MTL$_{\text{E}}$	77.93	78.45	**59.89**	**56.96**
MTL$_{\text{F}}$	76.74	**78.51**	58.91	56.00

Table 3: Accuracy (%) of the MTL models.

6 Experimental Results

Tables 2 and 3 present the accuracies of the different TL and MTL models on the development and test splits in `NomBank` and `PCEDT`. The top row in both tables shows the accuracy of the STL model. All the models were trained on the training split only. There are several observations one can draw from these tables. First, the accuracy of the STL models drops when the models are evaluated on the test split, whether on `NomBank` or `PCEDT`. Second, all the TL models achieve better accuracy on the test split of `NomBank` even though transfer learning does not remarkably improve accuracy on the development split of the same dataset. The MTL models, especially MTL$_{\text{F}}$, have a negative effect on the development accuracy of `NomBank`, but we still see the same improvement, as in TL, on the test split. Third, both the TL and MTL models exhibit less consistent effects on `PCEDT` (on both the development and test splits) compared to `NomBank`; for example, all the TL models lead to about 1.25 points absolute improvement in accuracy on `NomBank`, whereas in `PCEDT` TL$_{\text{E}}$ is clearly better than the other two TL models (TL$_{\text{E}}$ improves over the STL accuracy by 1.37 points).

Overall, the accuracy of the STL models drops when evaluated on the test split of `NomBank` and `PCEDT` (in comparison to their accuracy on the development split); this might be an indicator of overfitting, especially because we select the model that performs best on the development split in our stopping criterion. Both TL and MTL, on the other hand, improve accuracy on the test splits, even though the same stopping criterion was used for STL, TL and MLT. We interpret this result as improvement in the models' generalization ability. However, given that these improvements are relatively small, we next take a closer look at the results to understand if and how TL and MTL help.

7 Results Analysis

This section presents a systematic analysis of the performance of the models based on insights from the dataset used in the experiments as well as the classification errors of the models. The discussion in the following sections is based on the results on the test split rather than the development split, primarily because the former is larger in size.[7]

---

[7] One can also argue that result analysis on the test split is stricter than on the validation split. While using an early stop-

	A0	A1	A2	A3	LOC	MNR	TMP
Count	132	1282	153	75	25	25	27
STL	49.82	87.54	**45.78**	60.81	28.57	29.41	**66.67**
TL_E	**55.02**	87.98	41.61	60.14	27.91	33.33	63.83
TL_H	54.81	87.93	42.51	60.00	25.00	**35.29**	65.31
TL_EH	53.62	87.95	42.70	61.11	29.27	33.33	65.22
MTL_E	54.07	88.34	42.86	61.97	**30.00**	28.57	**66.67**
MTL_F	53.09	**88.41**	38.14	**62.69**	00.00	00.00	52.17

Table 4: Per-label F_1 score on the `NomBank` test split.

	ACT	TWHEN	APP	PAT	REG	RSTR
Count	89	14	118	326	216	900
STL	43.90	42.11	22.78	42.83	20.51	68.81
TL_E	49.37	**70.97**	27.67	41.60	**30.77**	**69.67**
TL_H	53.99	62.07	25.00	**43.01**	26.09	68.99
TL_EH	49.08	64.52	**28.57**	42.91	28.57	69.08
MTL_E	**54.09**	66.67	24.05	42.03	27.21	69.31
MTL_F	47.80	42.11	25.64	40.73	19.22	68.89

Table 5: Per-label F_1 score on the `PCEDT` test split.

7.1 Relation Distribution

To demonstrate the difficulty of the problem at hand, we plot the distribution of the most frequent relations in `NomBank` and `PCEDT` across the three data splits in Figure 1. We find that almost 71.18% of the relations in the `NomBank` training split are of type `ARG1` (proto-typical patient), and 52.20% of the `PCEDT` relations are of type `RSTR` (an underspecified adnominal modifier). Such highly skewed distributions of the relations makes learning some of the other relations more difficult, if not impossible in some cases. In fact, of the 15 `NomBank` relations observed in the test split, five relations are never predicted by any of the STL, TL and MTL models, and of the 26 `PCEDT` relations observed in the test split only six are predicted. That said, the non-predicted relations are extremely infrequent in the training split (e.g. 23 `PCEDT` functors occur less than 20 times in the training split), and it is therefore questionable if an ML model will be able to learn them under any circumstances.

From this imbalanced distribution of relations, it immediately follows that accuracy alone, as evaluation measure, is not sufficient to identify the best performing model. Therefore, in the following section we report, and analyze, the F_1 scores of the predicted `NomBank` and `PCEDT` relations across all the STL, TL and MTL models.

7.2 Per-Relation F_1 Scores

Tables 4 and 5 show the per-relation F_1 scores for `NomBank` and `PCEDT`, respectively. Note that we only include the results for the relations that are actually predicted by at least one of the models.

We observe several interesting patterns in Tables 4 and 5. First, the MTL_F model seems to be confusing for both datasets: it leads to substantially degraded F_1 scores on four `NomBank` relations, including the locative modifier `ARGM-LOC` and manner modifier `ARGM-MNR` (shortened to `LOC` and `MNR` in Table 4) which the model is no longer able to predict. The same model has the worst F_1 score, compared to all other models, for two `PCEDT` relations, `REG` (which expresses a circumstance) and `PAT` (patient). Given that the MTL_F model achieves the highest accuracy on the `NomBank` test split (cf. Table 3), it becomes all the more evident that mere accuracy scores are not enough to judge the utility of TL and MTL for this task (and dataset).

Second, with the exception of the MTL_F model, all the TL and MTL models consistently improve the F_1 score of all the `PCEDT` relations except `PAT`. Most notably, the F_1 scores of the relations `TWHEN` and `ACT` see a remarkable boost, compared to other `PCEDT` relations, when only the embedding layer's weights are shared (MTL_E) or transfered (TL_E). This result can be partly explained by looking at the correspondence matrices between `NomBank` arguments and `PCEDT` functors shown in Tables 7 and 6, which show how the `PCEDT` functors map to `NomBank` arguments in the training split (Table 6) and the other way around (Table 7). From Table 6, we see that 80% of the compounds annotated as `TWHEN` in `PCEDT` were annotated as `ARGM-TMP` in `NomBank`. In addition, 47% of `ACT` (Actor) relations map to `ARG0` (Proto-Agent) in `NomBank`, even though this mapping is not as clear as one would have hoped, it is still relatively high if we consider how other `PCEDT` relations map to `ARG0`. The correspondence matrices also show that the assumed theoretical similarities between the `NomBank` and `PCEDT` relations do not always hold. Nonetheless, even such 'imperfect' correspondence can provide a 'training signal' that help the TL and MTL models learn relations such as `TWHEN` and `ACT`.

Since the TL_E model outperforms STL on predicting `REG` by ten absolute points, we inspected

---

ping criterion based on the validation data can help prevent overfitting on the training data, we still choose a model that achieves the best accuracy on the validation split. In addition, it's unclear if early stopping helps when the validation split is not fully representative of the problem (Prechelt, 2012).

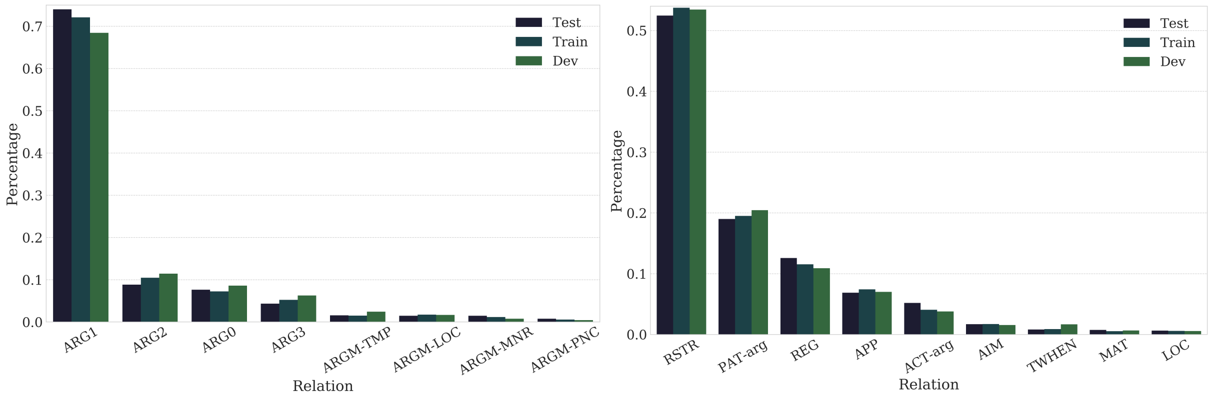

Figure 1: Distribution of NomBank relations (left) and PCEDT relations (right)

all the REG compounds that were correctly classified by the TL_E model but were misclassified by the STL model and found that the latter misclassified them as RSTR which indicates that TL from NomBank helps the TL_E model recover from the STL's over-generalization in RSTR prediction.

The two NomBank relations that receive the highest boost in F_1 score (about five absolute points) are ARG0 and ARGM-MNR, but the improvement in the latter relation corresponds to only one more compound which might well be predicted correctly by chance. Overall, TL and MTL from NomBank to PCEDT is more helpful than the other way around. One way to explain this is considering the first rows in Tables 6 and 7, where we see that five PCEDT relations (including the four most frequent ones) map to ARG1 in NomBank in more than 60% of the cases for each relation. This means that the weights learned to predict PCEDT relations offer little or no inductive bias for NomBank relations. Whereas if we consider the mapping from NomBank to PCEDT, we see that even though many NomBank arguments map to RSTR in PCEDT the percentages are lower, and hence the mapping is more 'diverse' (i.e. discriminative) which seems to help the TL and MTL models learn the less frequent PCEDT relations.

For completeness, we investigate why the PCEDT functor AIM is never predicted even though it is more frequent than TWHEN (cf. Figure 1). We find that AIM is almost always misclassifed as RSTR by all the models. Furthermore, we discover that AIM and RSTR have the highest lexical overlap in the training set among all other pairs of relations in PCEDT: 78.35% of the left constituents and 73.26% of the right constituents of the compounds annotated as AIM occur in other compounds annotated as RSTR. This explains why none of the models manage to learn the relation AIM but raises a question about the models' ability to learn relational representations; we pursue this question further in Section 7.3.

	RSTR	PAT	REG	APP	ACT	AIM	TWHEN
A1	0.70	0.90	0.78	0.62	0.47	0.65	0.10
A2	0.11	0.05	0.10	0.21	0.03	0.12	0.03
A0	0.06	0.01	0.04	0.13	0.47	0.07	-
A3	0.06	0.02	0.06	0.02	0.01	0.06	-
LOC	0.02	0.01	0.00	0.01	0.01	0.01	0.02
TMP	0.01	-	0.00	0.00	-	-	0.80
MNR	0.02	0.00	0.00	-	0.01	-	-
Count	3617	1312	777	499	273	116	59

Table 6: Correspondence matrix between PCEDT functors and NomBank arguments. Slots with '-' mean zero, 0.00 is a very small number but not zero.

	A1	A2	A0	A3	LOC	TMP	MNR
RSTR	0.51	0.54	0.47	0.63	0.66	0.36	0.78
PAT	0.24	0.09	0.03	0.08	0.07	-	0.05
REG	0.12	0.11	0.07	0.13	0.02	0.01	0.01
APP	0.06	0.14	0.13	0.03	0.05	0.01	-
ACT	0.03	0.01	0.26	0.01	0.03	-	0.03
AIM	0.02	0.02	0.02	0.02	0.01	-	-
TWHEN	0.00	0.00	-	-	0.01	0.46	-
Count	4932	715	495	358	119	103	79

Table 7: Correspondence matrix between NomBank arguments and PCEDT functors.

Finally, to clearly demonstrate the benefits of TL and MTL for NomBank and PCEDT, we report the F_1 macro-average scores in Table 8 (which is arguably the appropriate evaluation measure for imbalanced classification problems). Note that the relations that are not predicted by any of the models are not included in computing the macro-average. From Table 8 it becomes crystal clear that TL and MTL on the embedding layer yield remarkable improvements for PCEDT with about 7–

Model	`NomBank`	`PCEDT`
STL	52.66	40.15
TL$_{\text{E}}$	52.83	**48.34**
TL$_{\text{H}}$	52.98	46.52
TL$_{\text{EH}}$	**53.31**	47.12
MTL$_{\text{E}}$	53.21	47.23
MTL$_{\text{F}}$	42.07	40.73

Table 8: Macro-average F_1 score on the test split.

Model	`NomBank`			`PCEDT`		
	L	R	L&R	L	R	L&R
Count	351	286	72	351	286	72
STL	27.92	39.51	50.00	45.01	47.55	41.67
TL$_{\text{E}}$	25.93	36.71	48.61	**43.87**	47.55	41.67
TL$_{\text{H}}$	26.21	38.11	50.00	46.15	49.30	47.22
TL$_{\text{EH}}$	26.50	38.81	52.78	45.87	47.55	43.06
MTL$_{\text{E}}$	24.50	**33.22**	**38.89**	44.44	**47.20**	43.06
MTL$_{\text{F}}$	**22.79**	34.27	40.28	44.16	47.90	**38.89**

Table 9: Generalization error on the subset of unseen compounds in the test split. L: Left constituent. R: Right constituent. L&R: Completely unseen.

8 absolute points increase in macro-average F_1, in contrast to just 0.65 in the best case on `NomBank`.

7.3 Generalization on Unseen Compounds

Now we turn to analyze the models' ability to generalize over compounds unseen in the training split. Recent work by Dima (2016) and Vered and Waterson (2018) suggest that the gains achieved in noun–noun compound interpretation using word embeddings and somewhat similar neural classification models are in fact a by-product of *lexical memorization* (Levy et al., 2015); in other words, the classification models learn that a specific set of nouns is a strong indicator of a specific relation. Therefore, in order to gauge the role of lexical memorization in our models also, we quantify the number of unseen compounds that the STL, TL and MTL models predict correctly.

We distinguish between 'partly' and 'completely' unseen compounds. A compound is considered 'partly' unseen if one of its constituents (right or left) is not seen in the training data at all. A completely unseen compound is one whose left *and* right constituent are not seen in the training data (i.e. completely unseen compounds are the subset of compounds in the test split that have zero lexical overlap with the training split). Overall, almost 20% of the compounds in the test split have an unseen left constituent, about 16% of the compounds have unseen right constituent and 4% are completely unseen. In Table 9, we compare the performance of the different models on these three groups in terms of the proportion of compounds a model *misclassifies* in each group.

From Table 9 we see that TL and MTL reduce the `NomBank` generalization error in all cases, except TL$_{\text{H}}$ and TL$_{\text{EH}}$ on completely unseen compounds; the latter leads to higher generalization error. The MTL models lead to the biggest error reduction across the three types of unseen compounds; MTL$_{\text{E}}$ leads to about six points error reduction on compounds with unseen right constituent and eleven points on completely unseen ones, and MTL$_{\text{F}}$ reduces the error on unseen left constituent by five points. Note, however, that these results have to be read together with the *Count* row in Table 9 to get a complete picture. For instance, an eleven-point decrease in error on completely unseen compounds amounts to eight compounds. In `PCEDT`, the largest error reduction on unseen left constituents is 1.14 points which amounts to four compounds, 0.35 (just one compound) on unseen right constituents and 2.7 (or two compounds) on completely unseen compounds.

Since we see larger reductions in the generalization error in `NomBank`, we manually inspect the compounds that led to these reductions; i.e. we inspect the distribution of relations in the set of the correctly predicted unseen compounds. The MTL$_{\text{E}}$ model reduces the generalization error on completely unseen compounds by a total of eight compounds compared to the STL model, but seven of these compounds are annotated with `ARG1` which is the most frequent relation in `NomBank`. When it comes to the unseen right constituents, the 24 compounds MTL$_{\text{E}}$ improves on consist of 18 `ARG1` compounds, 5 `ARG0` compounds and one `ARG2` compound. We see a similar pattern upon inspecting the gains of the TL$_{\text{E}}$ model; where most of the improvement arises from predicting more `ARG1` and `ARG0` correctly.

The majority of the partly or completely unseen compounds that were misclassified by all models are *not* of type `ARG1` in `NomBank` or `RSTR` in `PCEDT`. This, together with the fact that the correctly predicted unseen compounds are annotated with the most frequent relations, indicate that the classification models rely on lexical memorization to learn the interpretation of compound relations.

Finally, to complement our understanding of the effect of lexical memorization, we plot the ratio of relation-specific constituents in `NomBank` and

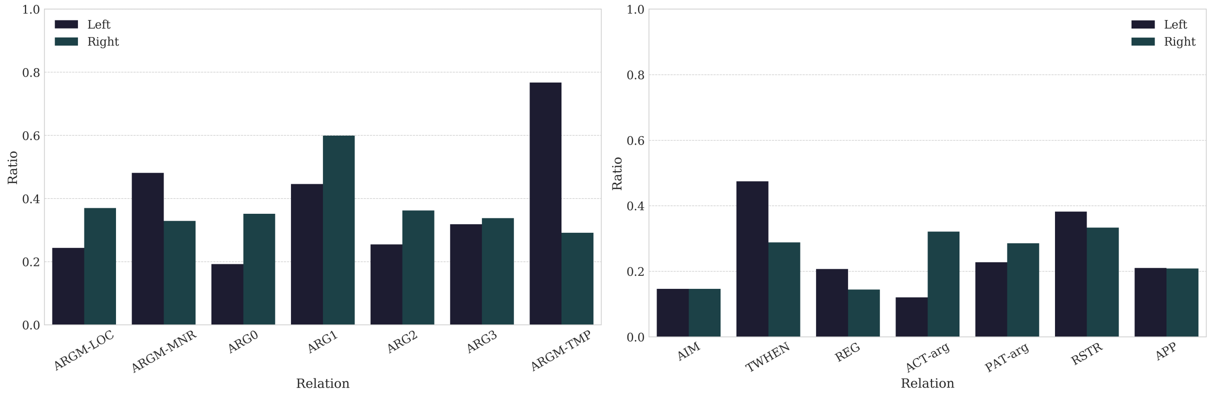

Figure 2: Ratio of relation-specific constituents in `NomBank` (left) and `PCEDT` (right).

`PCEDT` in Figure 2. We define relation-specific constituents as left or right constituents that only occur with one specific relation in the training split, and their ratio is simply their proportion in the overall set of left or right constituents per relation. Looking at Figure 2, we see that `NomBank` relations have higher ratios of relation-specific constituents in comparison to `PCEDT`, which arguably makes learning the former comparatively easier if the model is only to rely on lexical memorization. Furthermore, `ARGM-TMP` in `NomBank` and `TWHEN` in `PCEDT` stand out from other relations in Figure 2, which are also the two relations with the second highest F_1 score in their respective dataset—except in STL on `PCEDT` (cf. Tables 4 and 5). Lexical memorization is, therefore, the most likely explanation of such relatively high F_1 scores. We also observe some correlation between lower ratios of relation-specific constituents and relatively low F_1 scores, e.g. `APP` and `REG` in `PCEDT`. Based on these observations, we cannot rule out that our models exhibit some degree of lexical memorization effects, even though manual result analysis also reveals ‘counter-examples’ where the models generalize and make correct predictions where lexical memorization is impossible.

8 Conclusion

Transfer and multi-task learning for NLP currently receive a lot of attention, but for the time being there remains considerable uncertainty about which task properties and experimental settings actually are effective. In this work, we seek to shed light on the utility of TL and MTL perspectives on the semantic interpretation of noun–noun compounds. Through a comprehensive series of minimally contrasting experiments and in-depth analysis of results and prediction errors, we demonstrate the ability of both TL and MTL to mitigate the challenges of class imbalance and substantially improve prediction of low-frequency relations. In a nutshell, our TL and in particular MTL models make quantitatively and qualitatively better predictions, especially so on the ‘hardest’ inputs involving at least one constituent not seen in the training data—but clear indicators of remaining ‘lexical memorization’ effects arise from our error analysis of unseen compounds.

In general, transfer of representations or sharing across tasks is most effective at the embedding layers, i.e. the model-internal representation of the two compound constituents involved. In multi-task learning, full sharing of the model architecture across tasks worsens the model’s ability to generalize on the less frequent relations.

We experience the dataset by Fares (2016) as an interesting opportunity for innovative neural approaches to compound interpretation, as it relates this sub-problem to broad-coverage semantic role labeling or semantic dependency parsing in `PCEDT` and `NomBank`. In future work, we plan to incorporate other NLP tasks defined over these frameworks to learn noun–noun compound interpretation using TL and MTL. Such tasks include semantic role labeling of nominal predicates in `NomBank` annotations as well as verbal predicates in `PropBank` (Kingsbury and Palmer, 2002).

References

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Associa-*

tion for Computational Linguistics: Volume 2, Short Papers, page 164 – 169, Valencia, Spain. Association for Computational Linguistics.

Lou Burnard. 2000. Reference guide for the British National Corpus version 1.0.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28(1):41 – 75.

Yu-An Chung, Hung-yi Lee, and James Glass. 2018. Supervised and Unsupervised Transfer Learning for Question Answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, page 1585 – 1594. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning, ICML*, page 160 – 167.

Corina Dima. 2016. On the Compositionality and Semantic Interpretation of English Noun Compounds. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, page 27 – 39. Association for Computational Linguistics.

Corina Dima and Erhard Hinrichs. 2015. Automatic Noun Compound Interpretation using Deep Neural Networks and Word Embeddings. In *Proceedings of the 11th International Conference on Computational Semantics*, page 173 – 183, London, UK. Association for Computational Linguistics.

Pamela Downing. 1977. On the Creation and Use of English Compound Nouns. *Language*, 53(4):810 – 842.

Murhaf Fares. 2016. A Dataset for Joint Noun-Noun Compound Bracketing and Interpretation. In *Proceedings of the ACL 2016 Student Research Workshop*, page 72 – 79. Association for Computational Linguistics.

Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. 2017. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, page 271 – 276, Gothenburg, Sweden. Association for Computational Linguistics.

Timothy Wilking Finin. 1980. The Semantic Interpretation of Nominal Compounds. In *Proceedings of the First Annual National Conf. on Artificial Intelligence*. AAAI Press.

Christina L. Gagné and Edward J. Shoben. 1997. Influence of Thematic Relations on the Comprehension of Modifier–Noun Combinations. *Journal of Experimental Psychology: Learning Memory and Cognition*, 23(1):71 – 87.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2009. Classification of semantic relations between nominals. *Language Resources and Evaluation*, 43(2):105 – 121.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, page 3153 – 3160, Istanbul, Turkey.

Su Nam Kim and Timothy Baldwin. 2005. Automatic Interpretation of Noun Compounds Using WordNet Similarity. In *Second International Joint Conference on Natural Language Processing*, page 945 – 956, Berlin, Heidelberg. Springer Berlin Heidelberg.

Su Nam Kim and Timothy Baldwin. 2013. A lexical semantic approach to interpreting and bracketing English noun compounds. *Natural Language Engineering*, 19(03):385 – 407.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*.

Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, page 1989 – 1993, Las Palmas, Spain.

Mark Lauer. 1995. *Designing Statistical Language Learners. Experiments on Noun Compounds*. Doctoral dissertation, Macquarie University, Sydney, Australia.

Judith N Levi. 1978. *The syntax and semantics of complex nominals*. Academic Press.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do Supervised Distributional Methods Really Learn Lexical Inference Relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 970 – 976, Denver, Colorado. Association for Computational Linguistics.

Charles Na Li. 1972. *Semantics and the Structure of Compounds in Chinese*. Ph.D. thesis, University of California, Berkeley.

Marco Marelli, Christina L. Gagné, and Thomas L. Spalding. 2017. Compounding as Abstract Operation in Semantic Space: Investigating relational effects through a large-scale, data-driven computational model. *Cognition*, 166:207 – 224.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? Semantic sequence prediction under varying data conditions. In

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, page 44 – 53, Valencia, Spain. Association for Computational Linguistics.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. Annotating noun argument structure for NomBank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, page 803 – 806, Lisbon, Portugal.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How Transferable are Neural Networks in NLP Applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, page 479 – 489, Austin, Texas. Association for Computational Linguistics.

Preslav Nakov. 2008. Noun Compound Interpretation Using Paraphrasing Verbs: Feasibility Study. In *Artificial Intelligence: Methodology, Systems, and Applications*, page 103 – 117, Berlin, Heidelberg. Springer Berlin Heidelberg.

Preslav Ivanov Nakov. 2007. *Using the Web as an Implicit Training Set: Application to Noun Compound Syntax and Semantics*. Doctoral dissertation, EECS Department, University of California, Berkeley.

Diarmuid Ó Séaghdha. 2007. Annotating and Learning Compound Noun Semantics. In *Proceedings of the ACL 2007 Student Research Workshop*, page 73 – 78, Prague, Czech Republic.

Diarmuid Ó Séaghdha. 2008. Learning Compound Noun Semantics. Technical Report UCAM-CL-TR-735, University of Cambridge, Computer Laboratory, Cambridge, UK.

Diarmuid Ó Séaghdha and Ann Copestake. 2009. Using lexical and relational similarity to classify semantic relations. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, page 621 – 629, Athens, Greece. Association for Computational Linguistics.

Diarmuid Ó Séaghdha and Ann Copestake. 2013. Interpreting compound nouns with kernel methods. *Journal of Natural Language Engineering*, 19(3):331 – 356.

Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345 – 1359.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition LDC2011T07. Technical report, Technical Report. Linguistic Data Consortium, Philadelphia.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Meeting of the Association for Computational Linguistics*, page 2037 – 2048, Vancouver, Canada.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1532 – 1543, Doha, Qatar. Association for Computational Linguistics.

Lutz Prechelt. 2012. Early stopping — but when? In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, pages 53–67. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mary Ellen Ryder. 1994. *Ordered chaos: The interpretation of English noun-noun compounds*, volume 123. University of California Press.

Vered Shwartz and Ido Dagan. 2018. Paraphrase to Explicate: Revealing Implicit Noun-Compound Relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 1200 – 1211. Association for Computational Linguistics.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, page 231 – 235. Association for Computational Linguistics.

Stephen Tratz and Eduard Hovy. 2010. A Taxonomy, Dataset, and Classifier for Automatic Noun Compound Interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, page 678 – 687, Uppsala, Sweden. Association for Computational Linguistics.

Shwartz Vered and Chris Waterson. 2018. Olive Oil is Made *of* Olives, Baby Oil is Made *for* Babies: Interpreting Noun Compounds using Paraphrases in a Neural Model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, page 218 – 224, New Orleans, Louisiana. Association for Computational Linguistics.

Dissecting Contextual Word Embeddings: Architecture and Representation

Matthew E. Peters[*1], **Mark Neumann**[*1], **Luke Zettlemoyer**[2], **Wen-tau Yih**[1]
[1]Allen Institute for Artificial Intelligence, Seattle, WA, USA
[2]Paul G. Allen Computer Science & Engineering, University of Washington
{matthewp,markn,scottyih}@allenai.org lsz@cs.washington.edu

Abstract

Contextual word representations derived from pre-trained bidirectional language models (biLMs) have recently been shown to provide significant improvements to the state of the art for a wide range of NLP tasks. However, many questions remain as to how and why these models are so effective. In this paper, we present a detailed empirical study of how the choice of neural architecture (e.g. LSTM, CNN, or self attention) influences both end task accuracy and qualitative properties of the representations that are learned. We show there is a tradeoff between speed and accuracy, but all architectures learn high quality contextual representations that outperform word embeddings for four challenging NLP tasks. Additionally, all architectures learn representations that vary with network depth, from exclusively morphological based at the word embedding layer through local syntax based in the lower contextual layers to longer range semantics such coreference at the upper layers. Together, these results suggest that unsupervised biLMs, independent of architecture, are learning much more about the structure of language than previously appreciated.

1 Introduction

Contextualized word embeddings (Peters et al., 2018) derived from pre-trained bidirectional language models (biLMs) have been shown to substantially improve performance for many NLP tasks including question answering, entailment and sentiment classification (Peters et al., 2018), constituency parsing (Kitaev and Klein, 2018; Joshi et al., 2018), named entity recognition (Peters et al., 2017), and text classification (Howard and Ruder, 2018). Despite large gains (typical relative error reductions range from 10–25%), we do not yet fully understand why or how these models work in practice. In this paper, we take a step towards such understanding by empirically studying how the choice of neural architecture (e.g. LSTM, CNN, or self attention) influences both direct end-task accuracies and the types of neural representations that are induced (e.g. how do they encode notions of syntax and semantics).

Previous work on learning contextual representations has used LSTM-based biLMs, but there is no prior reason to believe this is the best possible architecture. More computationally efficient networks have been introduced for sequence modeling including including gated CNNs for language modeling (Dauphin et al., 2017) and feed forward self-attention based approaches for machine translation (Transformer; Vaswani et al., 2017). As RNNs are forced to compress the entire history into a hidden state vector before making predictions while CNNs with a large receptive field and the Transformer may directly reference previous tokens, each architecture will represent information in a different manner.

Given such differences, we study whether more efficient architectures can also be used to learn high quality contextual vectors. We show empirically that all three approaches provide large improvements over traditional word vectors when used in state-of-the-art models across four benchmark NLP tasks. We do see the expected tradeoff between speed and accuracy between LSTMs and the other alternatives, but the effect is relatively modest and all three networks work well in practice.

Given this result, it is important to better understand what the different networks learn. In a detailed quantitative evaluation, we probe the learned representations and show that, in every case, they represent a rich hierarchy of contextual information throughout the layers of the network in an analogous manner to how deep CNNs

---

[*] These authors contributed equally to this work.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1499–1509
Brussels, Belgium, October 31 - November 4, 2018.

trained for image classification learn a hierarchy of image features (Zeiler and Fergus, 2014). For example, we show that in contrast to traditional word vectors which encode some semantic information, the word embedding layer of deep biLMs focuses exclusively on word morphology. Moving upward in the network, the lowest contextual layers of biLMs focus on local syntax, while the upper layers can be used to induce more semantic content such as within-sentence pronominal coreferent clusters. We also show that the biLM activations can be used to form phrase representations useful for syntactic tasks. Together, these results suggest that large scale biLMs, independent of architecture, are learning much more about the structure of language than previous appreciated.

2 Contextual word representations from biLMs

To learn contextual word representations, we follow previous work by first training a biLM on a large text corpus (Sec. 2.1). Then, the internal layer activations from the biLM are transferred to downstream tasks (Sec. 2.3).

2.1 Bidirectional language models

Given a sequence of N tokens, $(t_1, t_2, ..., t_N)$, a biLM combines a forward and backward language model to jointly maximize the log likelihood of both directions:

$$\sum_{k=1}^{N} \big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \overrightarrow{\Theta}) \\ + \log p(t_k \mid t_{k+1}, \ldots, t_N; \overleftarrow{\Theta}) \big) ,$$

where $\overrightarrow{\Theta}$ and $\overleftarrow{\Theta}$ are the parameters of the forward and backward LMs respectively.

To compute the probability of the next token, state-of-the-art neural LMs first produce a context-insensitive token representation or word embedding, $\mathbf{x}_k$, (with either an embedding lookup or in our case a character aware encoder, see below). Then, they compute L layers of context-dependent representations $\overrightarrow{\mathbf{h}}_{k,i}$ where $i \in [1, L]$ using a RNN, CNN or feed forward network (see Sec. 3). The top layer output $\overrightarrow{\mathbf{h}}_{k,L}$ is used to predict the next token using a Softmax layer. The backward LM operates in an analogous manner to the forward LM. Finally, we can concatenate the forward and backward states to form L layers of contextual representations, or context vectors, at each token position: $\mathbf{h}_{k,i} = [\overrightarrow{\mathbf{h}}_{k,i}; \overleftarrow{\mathbf{h}}_{k,i}]$. When training, we tie the weights of the word embedding layers and Softmax in each direction but maintain separate weights for the contextual layers.

2.2 Character based language models

Fully character aware models (Kim et al., 2015) are considerably more parameter efficient then word based models but more computationally expensive then word embedding based methods when training. During inference, these differences can be largely eliminated by pre-computing embeddings for a large vocabulary and only falling back to the full character based method for rare words. Overall, for a large English language news benchmark, character aware models have slightly better perplexities then word based ones, although the differences tend to be small (Józefowicz et al., 2016).

Similar to Kim et al. (2015), our character-to-word encoder is a five-layer sub-module that first embeds single characters with an embedding layer then passes them through 2048 character n-gram CNN filters with max pooling, two highway layers (Srivastava et al., 2015), and a linear projection down to the model dimension.

2.3 Deep contextual word representations

After pre-training on a large data set, the internal representations from the biLM can be transferred to a downstream model of interest as contextual word representations. To effectively use all of the biLM layers, Peters et al. (2018) introduced ELMo word representations, whereby all of the layers are combined with a weighted average pooling operation, $\mathbf{ELMo}_k = \gamma \sum_{j=0}^{L} s_j \mathbf{h}_{k,j}$. The parameters s are optimized as part of the task model so that it may preferentially mix different types of contextual information represented in different layers of the biLM. In Sec. 4 we evaluate the relative effectiveness of ELMo representations from three different biLM architectures vs. pre-trained word vectors in four different state-of-the-art models.

3 Architectures for deep biLMs

The primary design choice when training deep biLMs for learning context vectors is the choice of the architecture for the contextual layers. However, it is unknown if the architecture choice is important for the quality of learned representations. To study this question, we consider two alterna-

Architecture	# layers	Perplexity	# params. (M)	Inference (ms) 1 sentence	Inference (ms) 64 sentences
LSTM	2	39.7	76 / 94	44 / 46	66 / 85
LSTM	4	37.5	151 / 153	85 / 86	102 / 118
Transformer	6	40.7	38 / 56	12 / 13	22 / 44
Gated CNN	16	44.5	67 / 85	9 / 11	29 / 55

Table 1: Characteristics of the different biLMs in this study. For each model, the table shows the number of layers used for the contextual representations, the averaged forward and backward perplexities on the 1 Billion Word Benchmark, the number of parameters (in millions, excluding softmax) and the inference speed (in milliseconds with a Titan X GPU, for sentences with 20 tokens, excluding softmax). For the number of parameters and inference speeds we list both the values for just the contextual layers and all layers needed to compute context vectors.

tives to LSTMs as described below. See the appendix for the hyperparameter details.

3.1 LSTM

Among the RNN variants, LSTMs have been shown to provide state-of-the-art performance for several benchmark language modeling tasks (Józefowicz et al., 2016; Merity et al., 2018; Melis et al., 2018). In particular, the LSTM with projection introduced by Sak et al. (2014) allows the model to use a large hidden state while reducing the total number of parameters.This is the architecture adopted by Peters et al. (2018) for computing ELMo representations. In addition to the pre-trained 2-layer biLM from that work,[1] we also trained a deeper 4-layer model to examine the impact of depth using the publicly available training code.[2] To reduce the training time for this large 4-layer model, we reduced the number of parameters in the character encoder by first projecting the character CNN filters down to the model dimension before the two highway layers.

3.2 Transformer

The Transformer, introduced by Vaswani et al. (2017), is a feed forward self-attention based architecture. In addition to machine translation, it has also provided strong results for Penn Treebank constituency parsing (Kitaev and Klein, 2018) and semantic role labeling (Tan et al., 2018). Each identical layer in the encoder first computes a multi-headed attention between a given token and all other tokens in the history, then runs a position wise feed forward network.

To adapt the Transformer for bidirectional language modeling, we modified a PyTorch based re-implementation (Klein et al., 2017)[3] to mask out future tokens for the forward language model and previous tokens for the backward language model, in a similar manner to the decoder masking in the original implementation. We adopted hyper-parameters from the “base” configuration in Vaswani et al. (2017), providing six layers of 512 dimensional representations for each direction.

Concurrent with our work, Radford et al. (2018) trained a large forward Transformer LM and fine tuned it for a variety of NLP tasks.

3.3 Gated CNN

Convolutional architectures have also been shown to provide competitive results for sequence modeling including sequence-to-sequence machine translation (Gehring et al., 2017). Dauphin et al. (2017) showed that architectures using Gated Linear Units (GLU) that compute hidden representations as the element wise product of a convolution and sigmoid gate provide perplexities comparable to large LSTMs on large scale language modeling tasks.

To adapt the Gated CNN for bidirectional language modeling, we closely followed the publicly available ConvSeq2Seq implementation,[4] modified to support causal convolutions (van den Oord et al., 2016) for both the forward and backward directions. In order to model a wide receptive field at the top layer, we used a 16-layer deep model, where each layer is a $[4, 512]$ residual block.

3.4 Pre-trained biLMs

Table 1 compares the biLMs used in the remainder of this study. All models were trained on the 1

---

[1] `http://allennlp.org/elmo`

[2] `https://github.com/allenai/bilm-tf`

[3] `http://nlp.seas.harvard.edu/2018/04/03/attention.html`

[4] `https://github.com/pytorch/fairseq`

Architecture	**MultiNLI**	**SRL**	**Constituency Parsing**	**NER**
GloVe	77.0 / 76.0	81.4	91.8	89.9 ± 0.35
LSTM 2-layer	79.6 / 79.3	84.6	**93.9**	**91.7 ± 0.26**
LSTM 4-layer	**80.1 / 79.7**	**84.7**	**93.9**	91.5 ± 0.12
Transformer	79.4 / 78.7	84.1	93.7	91.1 ± 0.26
Gated CNN	78.3 / 77.9	84.1	92.9	91.2 ± 0.14

Table 2: Test set performance comparison using different pre-trained biLM architectures. The performance metric is accuracy for MultiNLI and F_1 score for the other tasks. For MultiNLI, the table shows accuracy on both the matched and mismatched portions of the test set.

Billion Word Benchmark (Chelba et al., 2014) using a sampled softmax with 8192 negative samples per batch. Overall, the averaged forward and backward perplexities are comparable across the models with values ranging from 37.5 for the 4-layer LSTM to 44.5 for the Gated CNN. To our knowledge, this is the first time that the Transformer has been shown to provide competitive results for language modeling. While it is possible to reduce perplexities for all models by scaling up, our goal is to compare representations across architectures for biLMs of approximately equal skill, as measured by perplexity.

The Transformer and CNN based models are faster than the LSTM based ones for our hyperparameter choices, with speed ups of 3-5X for the contextual layers over the 2-layer LSTM model.[5] Speed ups are relatively faster in the single element batch scenario where the sequential LSTM is most disadvantaged, but are still 2.3-3X for a 64 sentence batch. As the inference speed for the character based word embeddings could be mostly eliminated in a production setting, the table lists timings for both the contextual layers and all layers of the biLM necessary to compute context vectors. We also note that the faster architectures will allow training to scale to large unlabeled corpora, which has been shown to improve the quality of biLM representations for syntactic tasks (Zhang and Bowman, 2018).

4 Evaluation as word representations

In this section, we evaluate the quality of the pre-trained biLM representations as ELMo-like contextual word vectors in state-of-the-art models across a suite of four benchmark NLP tasks. To do so, we ran a series of controlled trials by swapping out pre-trained GloVe vectors (Pennington et al., 2014) for contextualized word vectors from each biLM computed by applying the learned weighted average ELMo pooling from Peters et al. (2018).[6] Each task model only includes one type of pre-trained word representation, either GloVe or ELMo-like, this is a direct test of the transferability of the word representations. In addition, to isolate the general purpose LM representations from any task specific supervision, we did not fine tune the LM weights.

Table 2 shows the results. Across all tasks, the LSTM architectures perform the best. All architectures improve significantly over the GloVe only baseline, with relative improvements of 13% – 25% for most tasks and architectures. The gains for MultiNLI are more modest, with relative improvements over GloVe ranging from 6% for the Gated CNN to 13% for the 4-layer LSTM. The remainder of this section provides a description of the individual tasks and models with details in the Appendix.

4.1 MultiNLI

The MultiNLI dataset (Williams et al., 2018) contains crowd sourced textual entailment annotations across five diverse domains for training and an additional five domains for testing. Our model is a re-implementation of the ESIM sequence model (Chen et al., 2017). It first uses a biLSTM to encode the premise and hypothesis, then computes an attention matrix followed by a local inference layer, another biLSTM inference composition layer, and finally a pooling operation before

---

[5] While the CNN and Transformer implementations are reasonably well optimized, the LSTM biLM is not as it does not use an optimized CUDA kernel due to the use of the projection cell.

[6] Generally speaking, we found adding pre-trained GloVe vectors in addition to the biLM representations provided a small improvement across the tasks.

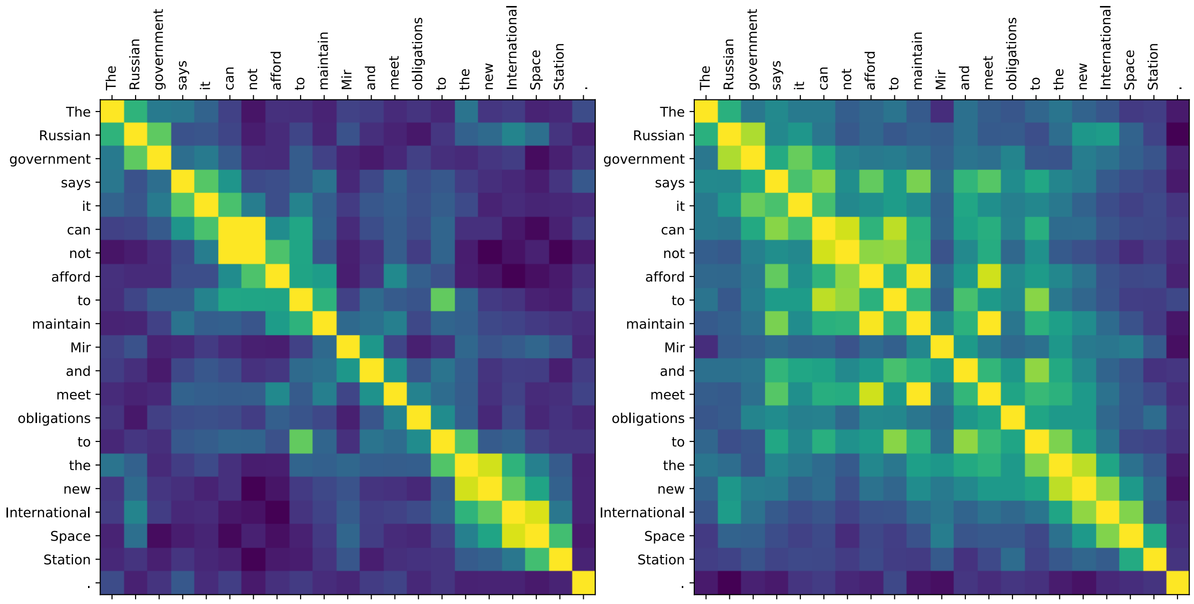

Figure 1: Visualization of contextual similarity between all word pairs in a single sentence using the 4-layer LSTM. The left panel uses context vectors from the bottom LSTM layer while the right panel uses the top LSTM layer. Lighter yellow-colored areas have higher contextual similarity.

the output layer. With the 2-layer LSTM ELMo representations, it is state-of-the-art for SNLI (Peters et al., 2018). As shown in Table 2, the LSTMs perform the best, with the Transformer accuracies 0.2% / 0.6% (matched/mismatched) less then the 2-layer LSTM. In addition, the contextual representations reduce the matched/mismatched performance differences showing that the biLMs can help mitigate domain effects. The ESIM model with the 4-layer LSTM ELMo-like embeddings sets a new state-of-the-art for this task, exceeding the highest previously published result by 1.3% matched and 1.9% mismatched from Gong et al. (2018).

4.2 Semantic Role Labeling

The Ontonotes 5.0 Dataset (Pradhan et al., 2013) contains predicate argument annotations for a variety of types of text, including conversation logs, web data, and biblical extracts. For our model, we use the deep biLSTM from He et al. (2017) who modeled SRL as a BIO tagging task. With ELMo representations, it is state-of-the-art for this task (Peters et al., 2018). For this task, the LSTM based word representations perform the best, with absolute improvements of 0.6% of the 4-layer LSTM over the Transformer and CNN.

4.3 Constituency parsing

The Penn Treebank (Marcus et al., 1993) contains phrase structure annotation for approximately 40k sentences sourced from the Wall Street Journal. Our model is the Reconciled Span Parser (RSP; Joshi et al., 2018), which, using ELMo representations, achieved state of the art performance for this task. As shown in Table 2, the LSTM based models demonstrate the best performance with a 0.2% and 1.0% improvement over the Transformer and CNN models, respectively. Whether the explicit recurrence structure modeled with the biLSTM in the RSP is important for parsing is explored in Sec. 5.3.

4.4 Named entity recognition

The CoNLL 2003 NER task (Sang and Meulder, 2003) provides entity annotations for approximately 20K sentences from the Reuters RCV1 news corpus. Our model is a re-implementation of the state-of-the-art system in Peters et al. (2018) with a character based CNN word representation, two biLSTM layers and a conditional random field (CRF) loss (Lafferty et al., 2001). For this task, the 2-layer LSTM performs the best, with averaged F_1 0.4% - 0.8% higher then the other biLMs averaged across five random seeds.

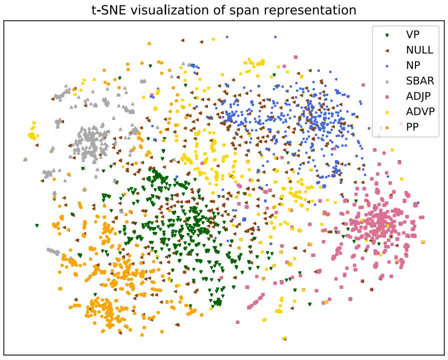

Figure 2: t-SNE visualization of 3K random chunks and 500 unlabeled spans ("NULL") from the CoNLL 2000 chunking dataset.

5 Properties of contextual vectors

In this section, we examine the intrinsic properties of contextual vectors learned with biLMs, focusing on those that are independent of the architecture details. In particular, we seek to understand how familiar types of linguistic information such as syntactic or coreferent relationships are represented throughout the depth of the network. Our experiments show that deep biLMs learn representations that vary with network depth, from morphology in the word embedding layer, to local syntax in the lowest contextual layers, to semantic relationships such as coreference in the upper layers.

We gain intuition and motivate our analysis by first considering the inter-sentence contextual similarity of words and phrases (Sec. 5.1). Then, we show that, in contrast to traditional word vectors, the biLM word embeddings capture little semantic information (Sec. 5.2) that is instead represented in the contextual layers (Sec. 5.3). Our analysis moves beyond single tokens by showing that a simple span representation based on the context vectors captures elements of phrasal syntax.

5.1 Contextual similarity

Nearest neighbors using cosine similarity are a popular way to visualize the relationships encoded in word vectors and we can apply a similar method to context vectors. As the biLMs use context vectors to pass information between layers in the network, this allows us to visualize how information is represented throughout the network.

Intra-sentence similarity Fig. 1 shows the intra-sentence contextual similarity between all pairs of words in single sentence using the 4-layer LSTM.[7] From the figure, we make several observations. First, the lower layer (left) captures mostly local information, while the top layer (right) represents longer range relationships. Second, at the lowest layer the biLM tends to place words from the same syntactic constituents in similar parts of the vector space. For example, the words in the noun phrase "the new international space station" are clustered together, similar to "can not" and "The Russian government".

In addition, we can see how the biLM is implicitly learning other linguistic information in the upper layer. For example, all of the verbs ("says", "can", "afford", "maintain", "meet") have high similarity suggesting the biLM is capturing part-of-speech information. We can also see some hints that the model is implicitly learning to perform coreference resolution by considering the high contextual similarity of "it" to "government", the head of "it"s antecedent span. Section 5.3 provides empirical support for these observations.

Span representations The observation that the biLM's context vectors abruptly change at syntactic boundaries suggests we can also use them to form representations of spans, or consecutive token sequences. To do so, given a span of S tokens from indices s_0 to s_1, we compute a span representation $\mathbf{s}_{(s_0,s_1),i}$ at layer i by concatenating the first and last context vectors with the element wise product and difference of the first and last vectors:

$$\mathbf{s}_{(s_0,s_1),i} = [\mathbf{h}_{s_0,i}; \mathbf{h}_{s_1,i}; \mathbf{h}_{s_0,i} \odot \mathbf{h}_{s_1,i}; \mathbf{h}_{s_0,i} - \mathbf{h}_{s_1,i}].$$

Figure 2 shows a t-SNE (Maaten and Hinton, 2008) visualization of span representations of 3,000 labeled chunks and 500 spans not labeled as chunks from the CoNLL 2000 chunking dataset (Sang and Buchholz, 2000), from the first layer of the 4-layer LSTM. As we can see, the spans are clustered by chunk type confirming our intuition that the span representations capture elements of syntax. Sec. 5.3 evaluates whether we can use these span representations for constituency parsing.

Unsupervised pronominal coref We hypothesize that the contextual similarity of coreferential mentions should be similar, as in many cases it is possible to replace them with their referent. If true, we should be able to use contextual similarity to perform unsupervised coreference reso-

---

[7] See appendix for visualizations of the other models.

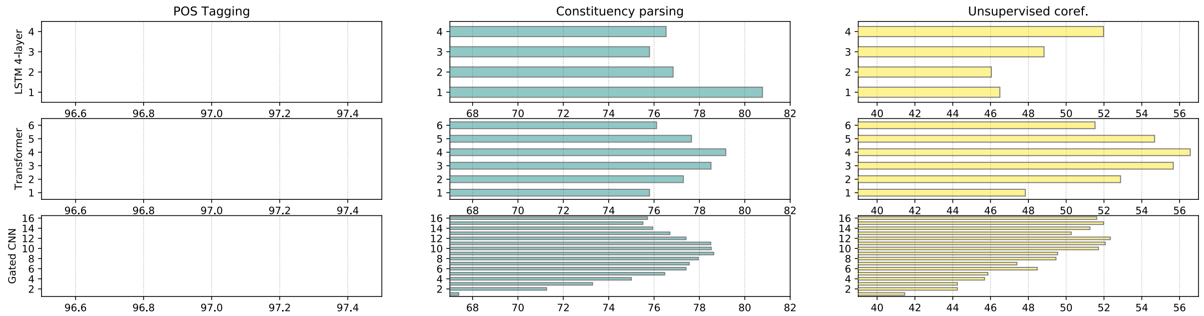

Figure 3: Various methods of probing the information stored in context vectors of deep biLMs. Each panel shows the results for all layers from a single biLM, with the first layer of contextual representations at the bottom and last layer at the top. From top to bottom, the figure shows results from the 4-layer LSTM, the Transformer and Gated CNN models. From left to right, the figure shows linear POS tagging accuracy (%; Sec. 5.3), linear constituency parsing (F_1; Sec. 5.3), and unsupervised pronominal coreference accuracy (%; Sec. 5.1).

Representation	**Syntactic**	**Semantic**
GloVe	77.9	**79.2**
n-gram hash	72.3	0.5
LSTM 4-layer	74.2	11.5
Transformer	**87.1**	48.8
Gated CNN	83.6	26.3

Table 3: Accuracy (%) for word vector analogies. In addition to the 300 dimension 840B GloVe vectors, the table contains results from a character n-gram hash and the context insensitive word embedding layer ($\mathbf{x}_k$) from the biLMs.

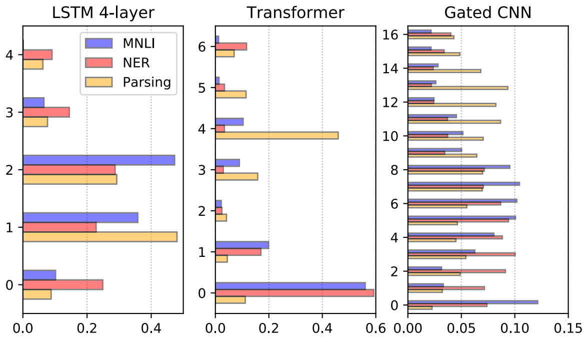

Figure 4: Normalized layer weights $\mathbf{s}$ for the tasks in Sec. 4. The vertical axis indexes the layer in the biLM, with layer 0 the word embedding $\mathbf{x}_k$.

lution. To test this, we designed an experiment as follows. To rule out trivially high mention-mention similarities due to lexical overlap, we restricted to pronominal coreference resolution. We took all sentences from the development set of the OntoNotes annotations in the CoNLL 2012 shared task (Pradhan et al., 2012) that had a third-person personal pronoun[8] and antecedent in the same sentence (904 sentences), and tested whether a system could identify the head word of the antecedent span given the pronoun location. In addition, by restricting to pronouns, systems are forced to rely on context to form their representation of the pronoun, as the surface form of the pronoun is uninformative. As an upper bound on performance, the state-of-the-art coreference model from Lee et al. (2017)[9] finds an antecedent span with the head word 64% of the time. As a lower bound on performance, a simple baseline that chooses the closest noun occurring before the pronoun has an accuracy of 27%, and one that chooses the first noun in the sentence has an accuracy of 35%. If we add an additional rule and further restrict to antecedent nouns matching the pronoun in number, the accuracies increase to 41% and 47% respectively.

To use contextual representations to solve this task, we first compute the mean context vector of the smallest constituent with more then one word containing the pronoun and subtract it from the pronoun's context vector. This step is motivated by the above observation that local syntax is the dominant signal in the contextualized word vectors, and removing it improves the accuracies of our method. Then, we choose the noun with the highest contextual similarity to the adjusted context vector that occurs before the pronoun and matches it in number.

The right hand column of Fig. 3 shows the results for all layers of the biLMs. Accuracies for the

---

[8] he, him, she, her, it, them, they

[9] `http://allennlp.org/models`

models peak between 52% and 57%, well above the baseline, with the Transformer overall having the highest accuracy. Interestingly, accuracies only drop 2-3% compared to 12-14% in the baseline if we remove the assumption of number agreement and simply consider all nouns, highlighting that the biLMs are to a large extent capturing number agreement across coreferent clusters. Finally, accuracies are highest at layers near the top of each model, showing that the upper layer representations are better at capturing longer range coreferent relationships then lower layers.

5.2 Context independent word representation

The word analogy task introduced in Mikolov et al. (2013) are commonly used as intrinsic evaluations of word vectors. Here, we use them to compare the word embedding layer from the biLMs to word vectors. The task has two types of analogies: syntactic with examples such as "bird:birds :: goat:goats", and semantic with examples such as "Athens:Greece :: Oslo:Norway". Traditional word vectors score highly on both sections. However, as shown in Table 3, the word embedding layer $\mathbf{x}_k$ from the biLMs is markedly different with syntactic accuracies on par or better then GloVe, but with very low semantic accuracies. To further highlight this distinction, we also computed a purely orthographically based word vector by hashing all character 1, 2, and 3-grams in a word into a sparse 300 dimensional vector. As expected, vectors from this method had near zero accuracy in the semantic portion, but scored well on the syntactic portion, showing that most of these analogies can be answered with morphology alone. As a result, we conclude that the word representation layer in deep biLMs is only faithfully encoding morphology with little semantics.

5.3 Probing contextual information

In this section, we quantify some of the anecdotal observations made in Sec. 5.1. To do so, we adopt a series of linear probes (Belinkov et al., 2017) with two NLP tasks to test the contextual representations in each model layer for each biLM architecture. In addition to examining single tokens, we also depart from previous work by examining to what extent the span representations capture phrasal syntax.

Our results show that all biLM architectures learn syntax, including span-based syntax; and part-of-speech information is captured at lower layers then constituent structure. When combined with the coreference accuracies in Sec. 5.1 that peak at even higher layers, this supports our claim that deep biLMs learn a hierarchy of contextual information.

POS tagging Peters et al. (2018) showed that the contextual vectors from the first layer of the 2-layer LSTM plus a linear classifier was near state-of-the-art for part-of-speech tagging. Here, we test whether this result holds for the other architectures. The second row of Fig. 3 shows tagging accuracies for all layers of the biLMs evaluated with the Wall Street Journal portion of Penn Treebank (Marcus et al., 1993). Accuracies for all of the models are high, ranging from 97.2 to 97.4, and follow a similar trend with maximum values at lower layers (bottom layer for LSTM, second layer for Transformer, and third layer for CNN).

Constituency parsing Here, we test whether the span representations introduced in Sec. 5.1 capture enough information to model constituent structure. Our linear model is a very simple and independently predicts the constituent type for all possible spans in a sentence using a linear classifier and the span representation. Then, a valid tree is built with a greedy decoding step that reconciles overlapping spans with an ILP, similar to Joshi et al. (2018).

The third row in Fig. 3 shows the results. Remarkably, predicting spans independently using the biLM representations alone has F_1 of near 80% for the best layers from each model. For comparison, a linear model using GloVe vectors performs very poorly, with F_1 of 18.1%. Across all architectures, the layers best suited for constituency parsing are at or above the layers with maximum POS accuracies as modeling phrasal syntactic structure requires a wider context then token level syntax. Similarity, the layers most transferable to parsing are at or below the layers with maximum pronominal coreference accuracy in all models, as constituent structure tends to be more local than coreference (Kuncoro et al., 2017).

5.4 Learned layer weights

Fig. 4 plots the softmax-normalized layer weights $\mathbf{s}$ from each biLM, learned as part of the tasks in Sec. 4. The SRL model weights are omitted as they close to constant since we had to regularize them to stabilize training. For constituency parsing, $\mathbf{s}$ mirrors the layer wise linear parsing results, with the largest weights near or at the same layers as maximum linear parsing. For both NER and MultiNLI, the Transformer focuses heavily on the word embedding layer, $\mathbf{x}_k$, and the first contextual layer. In all cases, the maximum layer weights occur below the top layers as the most transferable contextual representations tend to occur in the middle layers, while the top layers specialize for language modeling.

6 Related work

In addition to biLM-based representations, McCann et al. (2017) learned contextualized vectors with a neural machine translation system (CoVe). However, as Peters et al. (2018) showed the biLM based representations outperformed CoVe in all considered tasks, we focus exclusively on biLMs.

Liu et al. (2018) proposed using densely connected RNNs and layer pruning to speed up the use of context vectors for prediction. As their method is applicable to other architectures, it could also be combined with our approach.

Several prior studies have examined the learned representations in RNNs. Karpathy et al. (2015) trained a character LSTM language model on source code and showed that individual neurons in the hidden state track the beginning and end of code blocks. Linzen et al. (2016) assessed whether RNNs can learn number agreement in subject-verb dependencies. Our analysis in Sec. 5.1 showed that biLMs also learn number agreement for coreference. Kádár et al. (2017) attributed the activation patters of RNNs to input tokens and showed that a RNN language model is strongly sensitive to tokens with syntactic functions. Belinkov et al. (2017) used linear classifiers to determine whether neural machine translation systems learned morphology and POS tags. Concurrent with our work, Khandelwal et al. (2018) studied the role of context in influencing language model predictions, Gaddy et al. (2018) analyzed neural constituency parsers, Blevins et al. (2018) explored whether RNNs trained with several different objectives can learn hierarchical syntax, and Conneau et al. (2018) examined to what extent sentence representations capture linguistic features. Our intrinsic analysis is most similar to Belinkov et al. (2017); however, we probe span representations in addition to word representations, evaluate the transferability of the biLM representations to semantic tasks in addition to syntax tasks, and consider a wider variety of neural architectures in addition to RNNs.

Other work has focused on attributing network predictions. Li et al. (2016) examined the impact of erasing portions of a network's representations on the output, Sundararajan et al. (2017) used a gradient based method to attribute predictions to inputs, and Murdoch et al. (2018) decomposed LSTMs to interpret classification predictions. In contrast to these approaches, we explore the types of contextual information encoded in the biLM internal states instead of focusing on attributing this information to words in the input sentence.

7 Conclusions and future work

We have shown that deep biLMs learn a rich hierarchy of contextual information, both at the word and span level, and that this is captured in three disparate types of network architectures. Across all architecture types, the lower biLM layers specialize in local syntactic relationships, allowing the higher layers to model longer range relationships such as coreference, and to specialize for the language modeling task at the top most layers. These results highlight the rich nature of the linguistic information captured in the biLM's representations and show that biLMs act as a general purpose feature extractor for natural language, opening the way for computer vision style feature re-use and transfer methods.

Our results also suggest avenues for future work. One open question is to what extent can the quality of biLM representations be improved by simply scaling up model size or data size? As our results have show that computationally efficient architectures also learn high quality representations, one natural direction would be exploring the very large model and data regime.

Despite their successes biLM representations are far from perfect; during training, they have access to only surface forms of words and their order, meaning deeper linguistic phenomena must be learned "tabula rasa". Infusing models with explicit syntactic structure or other linguistically motivated inductive biases may overcome some of the limitations of sequential biLMs. An alternate direction for future work combines the purely unsupervised biLM training objective with existing annotated resources in a multitask or semi-supervised manner.

References

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James R. Glass. 2017. What do neural machine translation models learn about morphology? In *ACL*.

T. Blevins, O. Levy, and L. Zettlemoyer. 2018. Deep RNNs Encode Soft Hierarchical Syntax. In *ACL*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

Qian Chen, Xiao-Dan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *ACL*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Yann Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *ICML*.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? An analysis. In *NAACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *ACL workshop for NLP Open Source Software*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Luheng He, Kenton Lee, Mike Lewis, and Luke S. Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *ACL*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *ACL*.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv*, abs/1602.02410.

Ákos Kádár, Grzegorz Chrupala, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43:761–780.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. In *ICLR workshop*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *ACL*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. In *AAAI 2016*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *ACL*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *ACL*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *EACL*.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv*, abs/1612.08220.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. In *TACL*.

Liyuan Liu, Xiang Ren, Jingbo Shang, Jian Peng, and Jiawei Han. 2018. Efficient contextualized representation: Language model pruning for sequence labeling. In *EMNLP*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS 2017*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *ICLR*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. In *ICLR*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from lstms. In *ICLR*.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. In *arXiv*, volume abs/1609.03499.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *CoNLL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *EMNLP-CoNLL Shared Task*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Hasim Sak, Andrew W. Senior, and Franoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *CoNLL/LLL*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *NIPS*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *ICML*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv*, abs/1212.5701.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833.

Kelly Zhang and Samuel R. Bowman. 2018. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *arXiv preprint*.

Preposition Sense Disambiguation and Representation

Hongyu Gong, Jiaqi Mu, Suma Bhat, Pramod Viswanath
University of Illinois at Urbana-Champaign
`{hgong6, jiaqimu2, spbhat2, pramodv}@illinois.edu`

Abstract

Prepositions are highly polysemous, and their variegated senses encode significant semantic information. In this paper we match each preposition's left- and right context, and their interplay to the geometry of the word vectors to the left and right of the preposition. Extracting these features from a large corpus and using them with machine learning models makes for an efficient preposition sense disambiguation (PSD) algorithm, which is comparable to and better than state-of-the-art on two benchmark datasets. Our reliance on no linguistic tool allows us to scale the PSD algorithm to a large corpus and learn sense-specific preposition representations. The crucial abstraction of preposition senses as word representations permits their use in downstream applications–phrasal verb paraphrasing and preposition selection–with new state-of-the-art results.

1 Introduction

English prepositions form a closed class showing no inflectional variation and are some of the most frequent words. A computational-linguistic understanding of prepositions remains challenging owing to their highly polysemous nature and frequent participation in idiomatic expressions (Saint-Dizier, 2006). In this paper, we study the problem of sense disambiguation for prepositions.

She blinked **with** *confusion*. (Manner & Mood)
He *combines* professionalism **with** humor. (Accompanier)
He *washed* a small red teacup **with** *water*. (Means)

Table 1: Examples showing polysemous behavior of *with* and the TPP senses.

The highly polysemous nature of prepositions drives several syntactic and semantic processes. For instance, the preposition *with* has 18 senses listed in The Preposition Project (TPP) (Litkowski and Hargraves, 2005), examples of which, are shown in Table 1. We notice that *with* indicates an emotional state in *with confusion* and refers to an accompanier in *combine with*, while it suggests the idea of a tool or means in *wash with water*. Thus, preposition sense disambiguation (PSD) is vital for natural language understanding and a closer look at the function of prepositions in specific contexts is an important computational step.

Previous approaches to PSD (for instance, (Ye and Baldwin, 2007; Hovy et al., 2011)) have relied on linguistic tools and resources (the minimum of which involves dependency parsers and POS taggers) to capture the crucial contextual information of prepositions. We depart from prior art by using *no linguistic resources or tools* other than a set of word representations (trained on a large corpus). We interpret preposition senses as groups of similar contexts, where each instance of the preposition 'sense' is represented as a vector of context-dependent features. We find a simple feature extraction process that creatively harnesses the *geometry* of word representations and contributes to a scalable PSD algorithm. Our algorithm can reach near and even beat state-of-the-art performance on two benchmark datasets (SemEval 2007 and OEC); this is true in both unsupervised and supervised PSD settings.

A PSD algorithm that *efficiently scales to a large corpus* naturally paves the way for distributed representations of the preposition senses: we enrich the corpus with sense-specific information of prepositions using our PSD algorithm. Next, we repurpose an off-the-shelf word representation algorithm (Word2vec (Mikolov et al., 2013)) to relearn word representations with the key aspect that the length of the context surrounding the prepositions is crucially reduced. Sense-specific preposition representations thus learnt are strongly validated by using them in two

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1510–1521
Brussels, Belgium, October 31 - November 4, 2018.

applications–phrasal verb paraphrasing and preposition selection–using available datasets. We released our PSD system and paraphrasing dataset [1] available.

We summarize our contributions below:

- **Novel Perspective of Preposition Behavior**: We provide a novel selectional aspect of the context that best represents the sense of a preposition, where we match classical ideas from linguistics with the appropriate geometry of word embeddings. The standard view focuses on the left context (attachment) and the right context (complement) of the preposition; in this paper, we include the *interplay* between these two elements via an appropriate geometric representation.

- **Resource-independent Disambiguation**: We rely only on a set of trained word representations and no other language processing tool, where almost all prior approaches have included at least POS tagging and dependency parsing. Our results are comparable to, or better than, state-of-the-art on standard benchmarks.

- **Preposition Sense Representation Learning**: To the best of our knowledge, this is the first work on preposition sense representation. The power of our sense representation is reflected in the experimental comparisons with strong baseline approaches to phrasal verb paraphrasing and preposition selection, where we demonstrate the superiority of our approach that uses sense representation of prepositions.

2 Related Works

We place our work in the context of related studies in preposition representation and **Preposition Sense Disambiguation**: Preposition disambiguation has been explored on the SemEval dataset via various methods and external resources (part of speech taggers, chunkers, dependency parsers, named entity extractors, WordNet based supersense taggers and semantic role labelers) since 2007 (Yuret, 2007; Ye and Baldwin, 2007; Tratz and Hovy, 2009; Hovy et al., 2011; Popescu et al., 2007; Tratz and Hovy, 2011; Srikumar and Roth, 2013). Recently, (Gonen and Goldberg, 2016) use a multilingual parallel corpus processed using sequence to sequence neural networks for preposition disambiguation and achieve an accuracy within 5% of the state-of-the-art, which includes (Litkowski, 2013; Hovy et al., 2010; Srikumar and Roth, 2013). We note that we achieve the comparable performance as (Gonen and Goldberg, 2016) using *only* word embeddings.

Preposition Representation: Representation learning is fundamental to machine learning models (Wu et al., 2018a,b; Xu et al., 2018). Word embeddings such as Word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) have been widely recognized for their ability to capture linguistic regularities (including syntactic and semantic relations). On the other hand, no linguistic property of their prepositional embeddings is known; to the best of our knowledge, we propose the *first* sense-specific prepositional embeddings and demonstrate their linguistic regularities. A recent unsupervised approach by Gong et al. learns preposition representations to encode the syntactics and semantics by capturing their attachment and complement properties. Distantly related is (Hashimoto and Tsuruoka, 2015), which learns embeddings of prepositions acting as verb adjuncts by the factorization of a predicate tensor. Similarly, (Belinkov et al., 2014) explores the use of preposition representations optimized for the task of prepositional phrase attachment, but do not analyze their sense-specificity.

Sense-specific Representations: Several prior studies have sought polysemy-aware alternatives to word representations that take into account the context of the target word, including (Erk and Padó, 2008; Mitchell and Lapata, 2008; Reisinger and Mooney, 2010a; Thater et al., 2011; Dinu et al., 2012). More recently polysemy disambiguation for word embeddings have been proposed using external resources such as WordNet (Rothe and Schütze, 2015) or in an unsupervised way (Mu et al., 2017; Song et al., 2016; Arora et al., 2016; Neelakantan et al., 2014) with the latter two limiting the number of senses and validated for only nouns and verbs. The approach of (Neelakantan et al., 2014) is roughly similar to our baseline method using the average context vector. Our unsupervised approach is similar to that in (Reisinger and Mooney, 2010a), but limited to prepositions and uses novel features described next.

---

[1] `https://github.com/HongyuGong/PrepositionSenseDisambiguation.git`

3 Preposition Sense Disambiguation

The key intuition behind our sense disambiguation approach is the modern descriptive linguistic view (Huddleston, 1984; DeCarrico, 2000): the sense of a preposition in any sentence is driven by both its *attachment* and its *complement*; classical prescriptive linguistics had focused only on the latter (Beal, 2004), pp. 110, (Cobbett, 1823), pp. 16, (Lowth, 1762), pp. 8, 91.

Referring again to the examples in Table 1 we point out that italicized words determine the sense of "with." In the first sentence, the word 'confusion,' appearing as the right context of the preposition, is the complement of 'with', from which we infer that 'with' encodes the sense of 'manner'. In the second sentence, the accompanier sense of 'with' is because of its governor (attachment), the verb 'combine' appearing in the left context. In the last sentence, the sense of 'with' is 'by means of' and is determined by *both* the verb in its left context and the argument in its right context. Consider a new sentence with changed right context: 'He washed a small cup with a handle.' Here 'with' functions as an attribute. Again, changing its left context we get the sentence 'He asked for a small cup with water', where 'with' serves as an attribute instead of encoding the sense of means.

That the *left* and *right* context and their *interplay* are critical to prepositional sense disambiguation is also well established in the literature (Hovy et al., 2011; Litkowski and Hargraves, 2007). We match these linguistic properties to appropriate *geometric* objects within the space of word embeddings; the word embeddings are borrowed off-the-shelf – this work uses word2vec. We describe this next, focusing first on the left context, next on the right context and then on their interplay.

Left context feature v_ℓ is the average of the vectors of the left k_ℓ words (here k_ℓ is a parameter roughly taking values 1 through 4). This simple geometric operation is motivated by recent works (Faruqui et al., 2015; Kenter et al., 2016; Yu et al., 2014) representing a sentence by the average of its constituent words robustly and successfully in a variety of downstream settings. Although prior work (Hovy et al., 2010) points out that fixed window sizes are insufficient, when compared to using specific syntactic features (e.g., POS tags and dependency as done in prior works), we will see that the semantic information embedded in word vectors largely compensates for this limitation.

Right context feature v_r is the average of the vectors of the right k_r words (here k_r is a parameter roughly taking values 1 through 4). This is identical to the method adopted for the left context.

We model the **Context-interplay feature** v_{inter} to geometrically relate to both the left and the right contexts as follows. We choose it to be the vector *closest* to both the subspace spanned by the left context word vectors and that spanned by the right context word vectors. This geometric representation appears crucial to capture the prepositional-sense when the interplay between the contexts matters decisively, as seen empirically in our extensive experiments.

Let v_i^ℓ and v_j^r be the left- and the right context word vectors respectively. A precise mathematical definition of v_{inter} is below:

$$v_{\text{inter}} = \operatorname*{argmin}_{v:\|v\|_2=1} \Big(\min_{a_1,\ldots,a_{k_\ell}} \|v - \sum_{i=1}^{k_\ell} a_i v_i^\ell\|_2^2 + \min_{b_1,\ldots,b_{k_r}} \|v - \sum_{j=1}^{k_r} b_j v_j^r\|_2^2 \Big) , \tag{1}$$

where $\{a_i\}_{i=1}^{k_\ell}$ and $\{b_j\}_{j=1}^{k_r}$ are scalars. It is easy to find optimal $\{a_i^*\}$ and $\{b_j^*\}$ to solve the inner minimization problem. We have $a_i^* = \frac{v^T v_i^\ell}{\|v_i^\ell\|^2}$, and $b_j^* = \frac{v^T v_j^r}{\|v_j^r\|^2}$.

The minimization problem (1) is a quadratic optimization problem, so we can find a closed form solution to the unit vector v_{inter}. Suppose that we stack context word vectors $\{v_i^\ell\}$ and $\{v_j^r\}$ as a matrix $V_{d\times(k_\ell+k_r)}$, where d is the dimension of word vectors. The optimal $d-$dimension vector v_{inter} is the first principal component of matrix V.

Unsupervised learning of the senses of a given preposition is conducted by *clustering* its instances represented as a concatenation of the three feature vectors, while harnessing the large number of instances of each preposition in the large WikiCorpus (here we fix $k_\ell = k_r = 2$ and use k-means clustering). If the features do capture the prepositional sense efficiently, then the same-sense instances belong to the same cluster. Based on this intuition, we label each cluster with the dominant label of the training instances within this cluster. Given a test instance, we assign it to the nearest cluster (based on its Euclidean distance), and tag it with the cluster label. We note that the preposition senses are not balanced in the training dataset leading to a situation where frequent senses dominate more than one cluster. We address this by setting the number of clusters k equal to twice the number of senses and find that it separates the infrequent senses from frequent ones, about as well as traditional approaches such as those based on information criteria (Sugar and James, 2003) and the elbow method (Ketchen Jr and Shook, 1996).

Supervised learning of the senses using the three feature vectors was conducted based on the training examples provided in the benchmark PSD datasets. We did this using the standard support vector machines (SVM) (Cortes and Vapnik, 1995), multilayer perceptron (MLP) (Glorot and Bengio, 2010) and weighted k-nearest neighbor (k-NN) (Andoni and Indyk, 2006) classifiers. Each of these allows potentially different weighting of the three features in a context dependent way. The parameters were tuned to maximize the disambiguation accuracy on the development set provided in the benchmark PSD datasets. These experiments are discussed in detail next.

4 Experiments on Sense Disambiguation

The PSD algorithms were validated on the general sense disambiguation task using two datasets provided by TPP. We begin by introducing two benchmarks: SemEval and OEC datasets.

The **SemEval Dataset** consists of 34 prepositions instantiated by $24,663$ sentences covering 332 senses. Among them, $16,557$ sentences are used as training instances (**semtrain**) and 8096 sentences are test instances (**semtest**) for the preposition disambiguation task.

The **OEC dataset** consists of $7,650$ sentences collected from the Oxford English Corpus. Since these sentences included more prepositions than those in the SemEval dataset, we chose $3,587$ sentences that included the same 34 prepositions as used in the SemEval task.

Word embeddings. The word embeddings we used in our experiments were trained on the most recent scrape of the English Wikipedia with the Word2Vec CBOW model (Mikolov et al., 2013), with dimension 300. The linear combination of three vectors v_ℓ, v_r and v_{inter} is the feature to k-NN classifier.

Unsupervised PSD was performed by clustering the training instances fom the SemEval dataset using k-means. In the evaluation phase, each test instance was assigned to the closest cluster, and its sense was the dominant training sense within this cluster. In Table 2 we report the disambiguation accuracy on **semtest**, a new state-of-the-art result.

Supervised PSD was conducted by first conducting a 80/20 split of **semtrain** into training and development sets. The disambiguation accuracy calculated on both **semtest** and OEC datasets is reported in Table 3, using standard off-the-shelf classifiers. The sense disambiguation can be regarded as a multi-class classification problem. We used the SVM classifier with a linear kernel and its penalty parameter C as a tunable parameter, the MLP classifier with one hidden layer, and the number of neurons as a tunable parameter, and the k-NN classifier (weighted k-NN), with the number of nearest neighbors and the feature weights as tunable parameters (a linear combination of the three vectors v_ℓ, v_r and v_{inter} was the feature input to the k-NN classifier); all tunable parameters were tuned using the development set. The output dimension of these classifiers was the number of senses of prepositions. Additionally, the context window sizes k_ℓ and k_r were parameters for all the three classifiers, each tuned on the development set.

Baseline. Recent works have shown that the average word embedding serves as a good representation of the compositional sentential semantics (Faruqui et al., 2015; Kenter et al., 2016; Yu et al., 2014), and this single feature – the average of all context word vectors (both to the left and the right) – serves as a natural baseline.

Results. In both the unsupervised and supervised disambiguation settings, the best performance is achieved by using *all three* features, v_ℓ, v_r and v_i.

As summarized in Table 2, our unsupervised method achieves a 2.4% improvement over state-of-art (Hovy et al., 2011). The results in the supervised setting, tabulated in Table 3, reveal that the weighted k-NN classifier performs best. Denoting left, right and interplay features by ℓ, r, i respectively, Table 2 and 3 report our experimental results using only subset combinations of these features on the two disambiguation tasks.

An ablation analysis of the features reveals that the context-interplay feature is most beneficial when testing on the OEC dataset, but on the SemEval dataset, the left context feature appears to be the most beneficial. A likely explanation to this behavior is that several instances in **semtrain** and **semtest** share the governors the prepo-

System	State-of-art (Hovy et al., 2011)	k-means clustering				
		average	(ℓ, r)	(ℓ, i)	(r, i)	(ℓ, r, i)
Accuracy	0.56	0.555	0.561	0.565	0.534	**0.584**

Table 2: Performance of the unsupervised PSD compared with the state-of-the-art. (ℓ, i), (ℓ, r) and (r, i) correspond to feature ablation results.

	SemEval Dataset					OEC Dataset				
Feature Type	average	(ℓ, r)	(ℓ, i)	(r, i)	(ℓ, r, i)	average	(ℓ, r)	(ℓ, i)	(r, i)	(ℓ, r, i)
SVM	0.712	0.765	0.775	0.700	0.782	0.305	0.330	0.333	0.325	0.351
MLP	0.712	0.758	0.780	0.704	0.777	0.322	0.353	0.353	0.347	0.375
Weighted k-NN	0.731	0.781	0.792	0.733	**0.804**	0.329	0.341	0.380	0.367	**0.400**

Table 3: Supervised disambiguation on SemEval and OEC datasets.

sitions attach to. Hence the left feature (encoding the governor information) helps disambiguation on **semtest**. The governors and complements in OEC instances differ from those in **semtrain**. Therefore, the context-interplay feature provides more general context information than provided by the left and right context features by themselves for sense disambiguation on the OEC dataset.

A side-by-side comparison of the performance of our supervised approach with prior approaches is shown in Table 4. We note that the accuracy of our system is significantly better than that of the best PSD system in SemEval 2007 (11% higher accuracy), and 8% higher on the OEC dataset. It is noteworthy that while (Litkowski, 2013) fared better than our system with the SemEval data, our system outperformed (Litkowski, 2013) on the OEC dataset. Also we achieve performance comparable to the recent work (Gonen and Goldberg, 2016) which had access to a multilingual translation corpus (and other linguistic tools).

4.1 Spatial Expression Disambiguation

Prepositions such as ‘in’ and ‘on’ are used to encode spatial relations between the point of attachment and the complement of the preposition, but their senses show diversity depending on the context. For example, ‘on’ refers to the support from above in the sentence *clothes on the rack*, while it refers to support from below in *clothes on the desk*. These are instances of a phenomenon in natural language in which the mere concatenation of lexical information is not sufficient to derive the meaning of the phrase but the interactions among the meanings of the words is to be considered (termed compositional distributional semantic models (Marelli et al., 2014; Ritter et al., 2015)). We hypothesize that the relative placement of the objects involved in these phrases is achieved by considering the spatial senses of the prepositions involved, which in turn, is done by considering the interaction of its contexts as done by our approach. For this study, we focus on the disambiguation of the spatial senses encoded using the prepositions ‘in’ and ‘on’.

Dataset. (Ritter et al., 2015) studied ways of combining the meanings of the words in context to arrive at the meaning of the phrase which included spatial expressions using the prepositions ‘in’ and ‘on’. Their dataset consists of 420 training examples and 80 test examples, covering 5 types of locative expressions and given a sentence, the task is to arrive at the kind of locative expression encoded by the preposition. As examples, the preposition *in* refers to full containment in the sentence “an apple in the refrigerator”, whereas it refers to partial containment in “finger in the ring”. Similarly, the spatial relations represented by the preposition *on* is classified into three categories: adhesion to vertical surface (e.g., “sign on the building”), support by horizontal surface (e.g., “leaf on the ground”) and support from above (e.g., “bat on the branch”). A key observation here is that the spatial sense of the preposition (equivalently, spatial category) is a function of the two objects connected by the preposition.

Task. Given a sentence, we need to classify each occurrence of the preposition to its spatial category; for our study this is a fine-grained intra-preposition sense disambiguation problem (intra-sense because the sense is one of the spatial senses, albeit different from the TPP senses).

Method. The very small size of the training set in this experiment calls for a reduction in the feature dimension. Accordingly, we use a weighted linear combination of the three features used in our PSD algorithm and the resulting feature is $v = v_\ell + \beta v_r + \gamma v_{\text{inter}}$.

Dataset	System	Resources	Accuracy
SemEval	Our system	English corpus	0.804
	(Litkowski, 2013)	lemmatizer, dependency parser, WordNet	**0.86**
	(Srikumar and Roth, 2013)	dependency parser, WordNet	0.85
	(Gonen and Goldberg, 2016)	multilingual corpus, aligner, dependency parser	0.81
	(Ye and Baldwin, 2007)	chunker, dependency parser, named entity extractor, WordNet	0.69
OEC	Our system	English corpus	**0.40**
	(Litkowski, 2013)	lemmatizer, dependency parser, WordNet	0.32

Table 4: Preposition disambiguation performance comparison on SemEval and OEC datasets.

sense	1	2	3	4
closest words	backwards, reverse, angles, diagonal, between, forward	wearing, dress, hats, dresses, trousers, sleeves, pants, jacket	back, inside, underneath, from, into, where, onto	where, near, from, at, southern, northern, during
example	in all directions, move in, differ in	dress in black, in leather, in size	in the mail	in the UK, in Argentina
TPP sense	Manner_or_Degree	VariableQuality	ThingEntered	ThingEnclosed

Table 5: Example senses of the preposition “in”.

Recall that the hyperparameters β and γ of the k-NN classifier on the PSD task were tuned on the (mismatched) SemEval development set. We then generated the weighted features $v = v_\ell + \beta v_r + \gamma v_{\text{inter}}$ using the tuned values of β, γ from the PSD task. Classification within the spatial sense disambiguation is now conducted using a Multi-Layer Perceptron (MLP) with the feature vector v.

Baseline. The state-of-the-art method (Ritter et al., 2015) used the inclusion of the left and the right noun vectors as features, which is equivalent to adding the left and the right context features with a context window size of 1 in our set-up; this serves as our baseline.

Our method achieves an accuracy of **77**%, a significant improvement over the baseline accuracy of 71% in (Ritter et al., 2015). We note that this improvement is achieved even though we tuned the hyperparameters β and γ on the mismatched (but relatively bigger) SemEval dataset. To complete the comparison, we found (via grid search) that the best β, γ values result in only a slightly higher accuracy of 79%. This performance adds credence to the conclusion that the geometric features ($v_\ell, v_r, v_{\text{inter}}$) do indeed represent the *preposition in its context* efficiently and accurately.

5 Preposition Sense Representation

Thus far, we have empirically validated our disambiguation algorithm on standardized, but still stylized, datasets. A more thorough analysis is enabled by conducting preposition sense disambiguation on a very large unlabeled corpus. Such is the goal of this section, where we scale our lightweight PSD algorithm on a large corpus and learn sense-specific prepositional representations. The quality of the representations serves as an “extrinsic” evaluation of our PSD algorithm; this validation is done by repurposing datasets meant for other tasks.

Standard embedding methods do not account for the inherent polysemy in words. This is exacerbated in the case of prepositions. Indeed, to the best of our knowledge, no linguistic properties of the standard embeddings (say, word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014)) are known for preposition vectors. Recent works that learn sense-specific embeddings use the distinct “topics” assumed by the senses of a given word (as in (Rothe and Schütze, 2015) that explicitly uses WordNet senses) and have only been validated with respect to nouns and verbs.

Below, we validate the quality of the resulting sense representations in two tasks, where prepositional senses play an important role: (a) phrasal verb paraphrasing, and (b) preposition selection.

5.1 Phrasal Verb Paraphrasing

Prepositions often act as a connection between verbs and complements, carrying nontrivial semantic information. We used the trained k-NN classifier on TPP senses to label the prepositions in the English Wikipedia corpus, and learned sense-specific embeddings. These sense-specific representations are readily interpretable in terms of the extensive-resources of TPP. Table 5 shows several senses of "in" together with their nearest neighbors in the vector space.

Embedding	Global	Simplex	Sense
Accuracy	0.44	0.44	**0.73**

Table 6: Accuracy on phrasal verbs paraphrasing.

To validate the sense-specific preposition representation, we infer the meaning of verb-particle construction (VPC), such as *climb down* with sense embeddings. This is a lexical paraphrasing task of finding one word that captures the meaning of VPC (e.g., *climb down* = *descend*).

Dataset. Because a dataset for paraphrasing of VPCs was not available, we created one (which is made available in the supplementary material). It consists of 91 phrasal verbs, extracted from the VPC datasets in (Baldwin, 2005), (McCarthy et al., 2003) and the online Oxford dictionary[2].

For each VPC instance, we first disambiguated the preposition sense in the given context using the supervised method described in Section 3. We consider a linear approximation of phrasal embeddings under three settings:

(1) *Sense-specific embedding*, approximating the representation of a VPC as the sum of the vectors of its verb and its preposition with a specific sense. Thus we have $v_{\text{vp}}^{\text{sense}} = v_{\text{verb}} + v_{\text{prep}}^{\text{sense}}$.

(2) *Global embedding baseline*: $v_{\text{vp}}^{\text{global}} = v_{\text{verb}} + v_{\text{prep}}^{\text{global}}$, where $v_{\text{prep}}^{\text{global}}$ is the global preposition embedding disregarding its sense.

(3) *Simplex embedding baseline* approximates the phrasal embedding to be just the verb embedding, i.e., $v_{\text{vp}}^{\text{simplex}} = v_{\text{verb}}$.

For each approximate phrasal embedding ($v_{\text{vp}}^{\text{sense}}$, $v_{\text{vp}}^{\text{global}}$, $v_{\text{vp}}^{\text{simplex}}$), we list the nearest three verbs (excluding the verb in the phrase) as its paraphrase, with the distance measured by the cosine similarity between the word vectors.

Two proficient English speakers set the gold standard for whether the paraphrase was valid or not (for polysemous verbs, we consider the verb to be a valid paraphrase if it conveys the meaning in any of its senses) and reconciled disagreements. We used accuracy as the evaluation metric, which is the percent of phrasal verbs with a valid paraphrase among candidates.

Results. We note that sense representations are able to capture the nuance of polysemous verb phrases. As is shown in Fig. 1, the phrase "fight for" has more than one meaning depending on the sense of its preposition. In the expression "fight for human rights", *for* carries the sense of purpose. Since the expression is semantically equivalent to "defend human rights", "defend" can paraphrase the phrase "fight for". In another context "fight for the prize" where *for* is related with benefits, "fight for" should be paraphrased as "win" and the expression "win the prize" is similar to "fight for the prize".

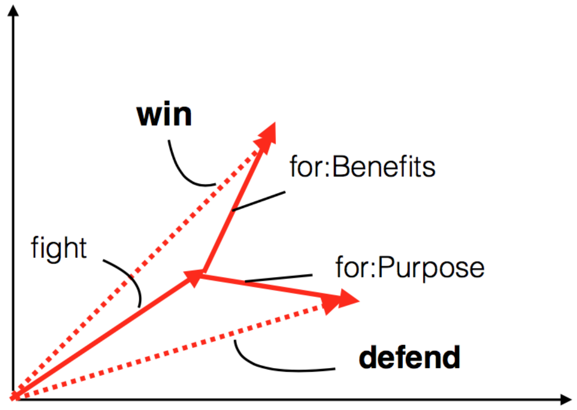

Figure 1: Paraphrasing polysemous verb phrases.

Some examples of phrasal verbs and paraphrases are shown in Table 7, with valid paraphrases highlighted. We report the performance of different embeddings in Table 6, where we notice that paraphrasing with the preposition sense embedding has a much higher accuracy than the two baselines. This validates the sense-specific preposition embedding suggesting its use in automatic paraphrasing of VPCs. Examples of paraphrases are shown in Table 7. A more detailed analysis of the results are in the appendix.

5.2 Preposition Selection

Given the polysemous and idiosyncratic nature of prepositions, choosing a preposition to fit a context can be a particularly challenging task for non-native English learners. Not surprisingly, preposition errors constitute the largest category of grammatical errors made by English learners (Chodorow et al., 2007). In this work, we show how the sense representations adequately capture the prepositional semantics, thus aiding preposition selection. Since TPP senses are fine-grained, we limit the senses available to be concrete or abstract by conflating the TPP senses to one of these

---

[2] `https://en.oxforddictionaries.com`

sentence	phrasal verb	paraphrasing		
		sense	global	simplex
The teaching is **carried on** in the form of folklore.	carried on	**conducted**	laid	placed
he **brought in** new ideas in the discussion.	brought in	**introduced**	came	came
She could not **keep from** crying.	keep from	**avoid**	get	maintain
Without a word he leaned forward and switched on the engine.	switched on	**starting**	shifted	reverted
I have certainly been **kicked in** the teeth by those bastards.	kicked in	**knocked**	throw	**knocked**
I have chosen to **block off** the easy track and so turn it into a dead end.	block off	**stopped**	cleared	cleared
The Rishon Le Zion killings **sparked off** a wave of sympathy protests.	sparked off	**ensued**	**spurred**	**ignited**
Stanley **put down** his paper and glared at her.	put down	**laid**	slammed	brought

Table 7: Paraphrasing of phrasal verbs.

two types (Reisinger and Mooney, 2010b). For example, "in a room" stands for the concrete sense of the preposition 'in', while "in her heart" corresponds to the abstract sense. In this part, we consider two senses for each preposition.

Dataset. We used three datasets, which consist of sentences marked with grammatical corrections out of which we only chose those with preposition errors. The Cambridge First Certificate in English (FCE) dataset contains $60,279$ prepositions with 4.8% error, the CoNLL dataset has $3,241$ prepositions with 4.7% errors and the Stack Exchange (SE) dataset has $15,814$ prepositions with 38.2% error (Prokofyev et al., 2014). Owing to its size, the FCE dataset was used for training, and the other two were used for testing. For each sentence with a preposition the task is to replace it with the correct one if it is used incorrectly.

Method. We classify all occurrences of each preposition sense into the two senses (abstract and concrete) by using the unsupervised PSD approach described in Section 4 to cluster the available senses. Then the prepositions in Wikipedia are labeled with one of the two senses, again using the unsupervised PSD approach. We then train sense embeddings on the newly labeled corpus with word2vec.

For a given sentence in the preposition selection task, we first disambiguate the sense of the preposition by checking which cluster it is closest to. The selection task is divided into preposition error detection and error correction. At the detection stage, we decide whether a preposition is used appropriately in the sentence. For this, we use as features the cosine similarity between the preposition sense embedding and the average word embedding in the context (the context size is 3), the rank of the preposition among all preposition choices with respect to the cosine similarity just mentioned, and the probability that the current preposition is replaced estimated from the training corpus. A decision tree classifier is used with these features to identify preposition errors.

At the second stage, we replace the current preposition p with another one if an error was detected at the first stage. Suppose that we consider replacing preposition p with q. We first disambiguate preposition q's sense given the context. We then use preposition q's sense embedding, the left context vector v_ℓ, the right context vector v_r, the interplay vector v_{inter} and the probability that q takes the place of p in the training corpus as input features to a two-layer MLP with 500 and 10 units in each layer. The MLP outputs a scalar to estimate how well the preposition q fits in p's context. The preposition with highest score is selected as the replacement.

Baseline. The state-of-the-art on preposition selection is one of the baselines, which makes use of lexical statistics from a large corpus as well as part-of-speech tags (Prokofyev et al., 2014). Also to evaluate the advantage of preposition sense representation over word representation, we have another baseline which uses the same classifier but the input features are the word embeddings instead of sense embeddings. The word embeddings were trained on Wikipedia English corpus with word2vec CBOW model.

Result. We compare the sense embedding-based approach against baselines in Table 8. As we can see, the use of sense representation achieves comparable performance to the state-of-the-art without using external linguistic tools. It also outperforms the baseline with word representation by a large margin.

6 Discussion

Resource-independence: Previous approaches to PSD relied on a part-of-speech tagger or depen-

Dataset	Method	Precision	Recall	F1 score
CoNLL	State-of-the-art	0.259	0.361	**0.302**
	Word representation	0.156	0.158	0.157
	Sense representation	0.279	0.283	0.281
SE	State-of-the-art	0.270	0.296	0.282
	Word representation	0.245	0.259	0.252
	Sense representation	0.281	0.297	**0.289**

Table 8: Performance in preposition selection.

dency parser to extract words modified by and modifying a preposition. In general, these words occur in the preposition's local context. We have allowed the context window to be a tunable parameter so that the classifier can learn to cover informative words in the context, and thus effectively captures the dependency information in a resource-independent fashion.

Context feature: The context averaging approach, which disregards context word order, suffers in accuracy compared to models that use left and right context words. This indicates that information about the word order relative to the preposition is useful in preposition disambiguation. Additionally, our use of the context-interplay feature combines the information on *both* sides of the preposition to infer its underlying sense.

Suppose that three expressions *a cup of medicine*, *professor of humanity* and *professor of mathematics* are in the training corpus, and the senses of the preposition *of* are 'contents', 'possessor' and 'field'. Given a test instance *professor of medicine*, it would be hard for the method with only the left or the right feature to decide the preposition sense since the test instance has the same word as each of the training instance, and their features in these two baselines are similar. However, the interplay vector in *professor of medicine* is closer to that in "professor of mathematics" than to other two training instances. The interplay feature prompts that *of* refers to a field (or species) instead of contents or possessor.

Data-driven insights into context dependence: Knowing the weights on the context features in the k-NN supervised PSD classifier, we can infer the extent to which prepositions rely on the complement and the attachment. For example, we found that in the case of the prepositions *behind* (occurring in, "shut behind her", "dip behind clouds"), *to* (e.g., "testify to the depth", "mumbling to himself"), and *with* (e.g., "amalgamated with her old school", and "rub with bare hands"), the verbs they attach to strongly influence their sense. For other prepositions such as *during* (e.g., "during the incident") and *on* (e.g., "on his hands"), the complement noun has more influence on the senses than governors.

Sense helps paraphrasing: We observe that sense-specific preposition representation helps improve phrasal verb paraphrasing greatly. Working with the VPC dataset and the simplistic model of compositionality, we interpret the results as positive indicators of the viability of using sense-specific prepositional embeddings to paraphrase verb-particle constructions. In the case of *light verbs*, whose meaning is determined largely by the particles they combine with, (e.g., *come down* $\sim$ *fall*), a valid paraphrase is found in the top 3 candidates when the sense-specific representation is used, and not when the simplex or the global representation is used.

7 Conclusion

This paper studies the preposition sense disambiguation by encoding the attachment and complement properties into context features. The disambiguation method performs well on three standard PSD tasks and readily scales to a large corpus. The resulting sense-specific representations are shown to capture semantics of preposition senses in our quantitative analysis. They are also shown to aid two downstream tasks: phrasal verb paraphrasing and preposition selection.

Acknowledgments

This work is supported by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network. We thank EMNLP anonymous reviewers for their constructive suggestions.

References

Alexandr Andoni and Piotr Indyk. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 459–468. IEEE.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. Linear algebraic structure of word senses, with applications to polysemy. *arXiv preprint arXiv:1601.03764*.

Timothy Baldwin. 2005. Deep lexical acquisition of verb–particle constructions. *Computer Speech & Language*, 19(4):398–414.

Joan C Beal. 2004. *Grammars and grammarians*, volume English in modern times 1700-1945, 89-123. London, Great Britain: Arnold.

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. 2014. Exploring compositional architectures and word vector representations for prepositional phrase attachment. *Transactions of the Association for Computational Linguistics*, 2:561–572.

Martin Chodorow, Joel R Tetreault, and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the fourth ACL-SIGSEM workshop on prepositions*, pages 25–30. Association for Computational Linguistics.

William Cobbett. 1823. *A grammar of the English language*. London: B. Bensley.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Jeanette S DeCarrico. 2000. *The structure of English: Studies in form and function for language teaching*, volume 1. University of Michigan Press/ESL.

Nicole Dehé. 2002. *Particle verbs in English: Syntax, information structure and intonation*, volume 59. John Benjamins Publishing.

Georgiana Dinu, Stefan Thater, and Sören Laue. 2012. A comparison of models of word meaning in context. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 611–615, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joseph E Emonds. 1985. A unified theory of syntactic categories. *Studies in generative grammar*, (19):1–356.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906. Association for Computational Linguistics.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

Hila Gonen and Yoav Goldberg. 2016. Semi supervised preposition-sense disambiguation using multilingual data. *arXiv preprint arXiv:1611.08813*.

Hongyu Gong, Suma Bhat, and Pramod Viswanath. 2018. Embedding syntax and semantics of prepositions via tensor decomposition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 896–906.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2015. Learning embeddings for transitive verb disambiguation by implicit tensor factorization. *ACL-IJCNLP 2015*, page 1.

Dirk Hovy, Stephen Tratz, and Eduard Hovy. 2010. What's in a preposition?: dimensions of sense disambiguation for an interesting word class. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 454–462. Association for Computational Linguistics.

Dirk Hovy, Ashish Vaswani, Stephen Tratz, David Chiang, and Eduard Hovy. 2011. Models and training for unsupervised preposition sense disambiguation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 323–328. Association for Computational Linguistics.

Rodney Huddleston. 1984. *Introduction to the Grammar of English*. Cambridge University Press.

Ray Jackendoff. 2002. English particle constructions, the lexicon, and the autonomy of syntax. *Verb-particle explorations*, pages 67–94.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, page 941951.

David J Ketchen Jr and Christopher L Shook. 1996. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, pages 441–458.

Ken Litkowski. 2013. Preposition disambiguation: Still a problem. *CL Research, Damascus, MD*, pages 1–8.

Ken Litkowski and Orin Hargraves. 2007. Semeval-2007 task 06: Word-sense disambiguation of prepositions. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 24–29. Association for Computational Linguistics.

Kenneth C Litkowski and Orin Hargraves. 2005. The preposition project. In *Proceedings of the Second ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, pages 171–179.

Robert Lowth. 1762. *A short introduction to English grammar*. London: A. Miller & R. & J. Dodsby.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 1–8.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pages 73–80. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. *Proceedings of ACL-08: HLT*, pages 236–244.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. Geometry of polysemy. *Proceedings of ICLR*.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. *Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Octavian Popescu, Sara Tonelli, and Emanuele Pianta. 2007. Irst-bp: Preposition disambiguation based on chain clarifying relationships contexts. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 191–194. Association for Computational Linguistics.

Roman Prokofyev, Ruslan Mavlyutov, Martin Grund, Gianluca Demartini, and Philippe Cudré-Mauroux. 2014. Correct me if i'm wrong: Fixing grammatical errors by preposition ranking. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 331–340. ACM.

Joseph Reisinger and Raymond J. Mooney. 2010a. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 109–117, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joseph Reisinger and Raymond J Mooney. 2010b. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

Samuel Ritter, Cotie Long, Denis Paperno, Marco Baroni, Matthew Botvinick, and Adele Goldberg. 2015. Leveraging preposition ambiguity to assess compositional distributional models of semantics. *Lexical and Computational Semantics (* SEM 2015)*, page 199.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *arXiv preprint arXiv:1507.01127*.

Patrick Saint-Dizier. 2006. *Syntax and semantics of prepositions*, volume 29. Springer Science & Business Media.

Linfeng Song, Zhiguo Wang, Haitao Mi, and Daniel Gildea. 2016. Sense embedding learning for word sense induction. *arXiv preprint arXiv:1606.05409*.

Vivek Srikumar and Dan Roth. 2013. Modeling semantic relations expressed by prepositions. *Transactions of the Association for Computational Linguistics*, 1:231–242.

Catherine A Sugar and Gareth M James. 2003. Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association*, 98(463):750–763.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *IJCNLP*.

Stephen Tratz and Dirk Hovy. 2009. Disambiguation of preposition sense using linguistically motivated features. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, pages 96–100. Association for Computational Linguistics.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268. Association for Computational Linguistics.

Aline Villavicencio. 2006. Verb-particle constructions in the world wide web. In *Syntax and Semantics of Prepositions*, pages 115–130. Springer.

Lingfei Wu, Ian En-Hsu Yen, Fangli Xu, Pradeep Ravikuma, and Michael Witbrock. 2018a. D2ke: From distance to kernel and embedding. *arXiv preprint arXiv:1802.04956*.

Lingfei Wu, Ian En-Hsu Yen, Jinfeng Yi, Fangli Xu, Qi Lei, and Michael Witbrock. 2018b. Random warping series: A random features method for time-series embedding. In *International Conference on Artificial Intelligence and Statistics*, pages 793–802.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Patrick Ye and Timothy Baldwin. 2007. Melb-yb: Preposition sense disambiguation using rich semantic features. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 241–244. Association for Computational Linguistics.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Deniz Yuret. 2007. Ku: Word sense disambiguation by substitution. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 207–213. Association for Computational Linguistics.

Auto-Encoding Dictionary Definitions into Consistent Word Embeddings

Tom Bosc
Mila, Université de Montréal
tom.bosc@umontreal.ca

Pascal Vincent
Mila, Université de Montréal, CIFAR,
Facebook AI Research
pascal.vincent@umontreal.ca

Abstract

Monolingual dictionaries are widespread and semantically rich resources. This paper presents a simple model that learns to compute word embeddings by processing dictionary definitions and trying to reconstruct them. It exploits the inherent recursivity of dictionaries by encouraging consistency between the representations it uses as inputs and the representations it produces as outputs. The resulting embeddings are shown to capture semantic similarity better than regular distributional methods and other dictionary-based methods. In addition, the method shows strong performance when trained exclusively on dictionary data and generalizes in one shot.

1 Introduction

Dense, low-dimensional, real-valued vector representations of words known as *word embeddings* have proven very useful for NLP tasks (Turian et al., 2010). They can be learned as a by-product of solving a particular task (Collobert et al., 2011). Alternatively, one can pretrain generic embeddings based on co-occurrence counts or using an unsupervised criterion such as predicting nearby words (Bengio et al., 2003; Mikolov et al., 2013). These methods implicitly rely on the distributional hypothesis (Harris, 1954; Sahlgren, 2008), which states that words that occur in similar contexts tend to have similar meanings.

It is common to study the relationships captured by word representations in terms of either *similarity* or *relatedness* (Hill et al., 2016). "Coffee" is related to "cup" as coffee is a beverage often drunk in a cup, but "coffee" is not similar to "cup" in that coffee is a beverage and cup is a container. Methods relying on the distributional hypothesis often capture relatedness very well, reaching human performance, but fare worse in capturing similarity and especially in distinguishing it from relatedness (Hill et al., 2016).

It is useful to specialize word embeddings to focus on either relation in order to improve performance on specific downstream tasks. For instance, Kiela et al. (2015) report that improvements on relatedness benchmarks also yield improvements on document classification. In the other direction, embeddings learned by neural machine translation models capture similarity better than distributional unsupervised objectives (Hill et al., 2014).

There is a wealth of methods that postprocess embeddings to improve or specialize them, such as retrofitting (Faruqui et al., 2014). On similarity benchmarks, they are able to reach correlation coefficients close to inter-annotator agreements. But these methods rely on additional resources such as paraphrase databases (Wieting et al., 2016) or graphs of lexical relations such as synonymy, hypernymy, and their converse (Mrkšić et al., 2017).

Rather than relying on such curated lexical resources that are not readily available for the majority of languages, we propose a method capable of improving embeddings by leveraging the more common resource of monolingual dictionaries.[1] Lexical databases such as WordNet (Fellbaum, 1998) are often built from dictionary definitions, as was proposed earlier by Amsler (1980). We propose to bypass the process of explicitly building a lexical database – during which information is structured but information is also lost – and instead directly use its detailed source: dictionary definitions. The goal is to obtain better representations for more languages with less effort.

The ability to process new definitions is also desirable for future natural language understanding systems. In a dialogue, a human might want to explain a new term by explaining it in his own words,

---

[1] See Appendix A for a list of online monolingual dictionaries.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1522–1532
Brussels, Belgium, October 31 - November 4, 2018.

and the chatbot should understand it. Similarly, question-answering systems should also be able to grasp definitions of technical terms that often occur in scientific writing.

We expect the embedding of a word to represent its meaning compactly. For interpretability purposes, it would be desirable to be able to generate a definition from that embedding, as a way to verify what information it captured. Case in point: to analyse word embeddings, Noraset et al. (2017) used RNNs to produce definitions from pretrained embeddings, manually annotated the errors in the generated definitions, and found out that more than half of the wrong definitions fit either the antonyms of the defined words, their hypernyms, or related but different words. This points in the same direction as the results of intrinsic evaluations of word embeddings: lexical relationships such as lexical entailment, similarity and relatedness are conflated in these embeddings. It also suggests a new criterion for evaluating word representations, or even learning them: they should contain the necessary information to reproduce their definition (to some degree). In this work, we propose a simple model that exploits this criterion. The model consists of a definition autoencoder: an LSTM processes the definition of a word to yield its corresponding word embedding. Given this embedding, the decoder attempts to reconstruct the bag-of-words representation of the definition. Importantly, to address and leverage the recursive nature of dictionaries – the fact that words that are used inside a definition have their own associated definition – we train this model with a consistency penalty that ensures proximity of the embeddings produced by the LSTM and those that are used by the LSTM.

Our approach is self-contained: it yields good representations when trained on nothing but dictionary data. Alternatively, it can also leverage existing word embeddings and is then especially apt at specializing them for the similarity relation. Finally, it is also extremely data-efficient, as it permits to create representations of new words in one shot from a short definition.

2 Model

2.1 Setting and motivation

We suppose that we have access to a dictionary that maps words to one or several definitions. Definitions themselves are sequences of words. Our training criterion is built on the following principle: we want the model to be able to recover the definition from which the representation was built. This objective should produce similar embeddings for words which have similar definitions. Our hypothesis is that this will help capture semantic similarity, as opposed to relatedness. Reusing the previous example, “coffee” and “cup” should get different representations in virtue of having very different definitions, while “coffee” and “tea” should get similar representations as they are both defined as beverages and plants.

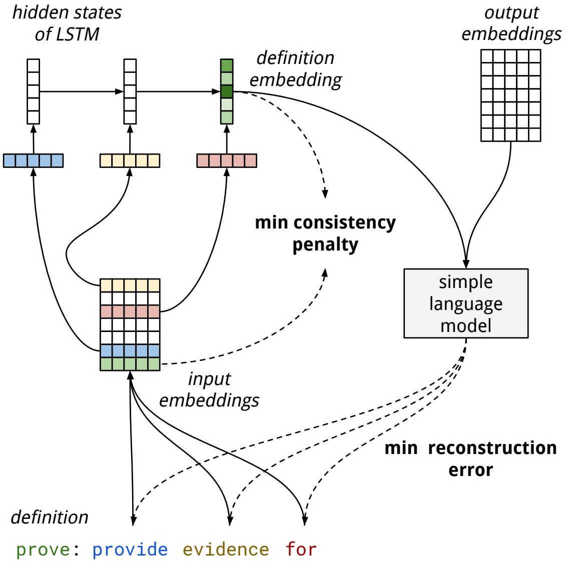

Figure 1: **Overview of the CPAE model.**

We chose to compute a single embedding per word in order to avoid having to disambiguate word senses. Indeed, word sense disambiguation remains a challenging open problem with mixed success on downstream task applications (Navigli, 2009). Also, recent papers have shown that a single word vector can capture polysemy and that having several vectors per word is not strictly necessary (Li and Jurafsky, 2015) (Yaghoobzadeh and Schütze, 2016). Thus, when a word has several definitions, we concatenate them to produce a single embedding.

2.2 Autoencoder model

Let $\mathcal{V}^{\mathcal{D}}$ be the set of all words that are *used* in definitions and $\mathcal{V}^{\mathcal{K}}$ the set of all words that are *defined*. We let $w \in \mathcal{V}^{\mathcal{K}}$ be a word and $D_w = (D_{w,1}, \dots, D_{w,T})$ be its definition, where $D_{w,t}$ is the index of a word in vocabulary $\mathcal{V}^{\mathcal{D}}$. We encode such a definition D_w by processing it with an

LSTM (Hochreiter and Schmidhuber, 1997).

The LSTM is parameterized by Ω and a matrix E of size $|\mathcal{V}^{\mathcal{D}}| \times m$, whose i^{th} row E_i contains an m-dimensional *input embedding* for the i^{th} word of $\mathcal{V}^{\mathcal{D}}$. These input embeddings can either be learned by the model or be fixed to a pretrained embedding. The last hidden state computed by this LSTM is then transformed linearly to yield an m-dimensional *definition embedding* h. Thus the encoder whose parameters are $\theta = \{E, \Omega, W, b\}$ computes this embedding h as

$$h = f_\theta(D_w) = W \, \mathrm{LSTM}_{E,\Omega}(D_w) + b.$$

The subsequent decoder can be seen as a conditional language model trained by maximum likelihood to regenerate definition D_w given definition embedding $h = f_\theta(D_w)$. We use a simple conditional unigram model with a linear parametrization $\theta' = \{E', b'\}$ where E' is a $|\mathcal{V}^{\mathcal{D}}| \times m$ matrix and b' is a bias vector. [2]

We maximize the log-probability of definition D_w under that model:

$$\begin{aligned} \log p_{\theta'}(D_w|h) &= \sum_t \log p_{\theta'}(D_{w,t}|h) \\ &= \sum_t \log \frac{e^{\left\langle E'_{D_{w,t}}, h\right\rangle + b'_{D_{w,t}}}}{\sum_k e^{\left\langle E'_k, h\right\rangle + b'_k}} \end{aligned} \quad (1)$$

where $\langle , \rangle$ denotes an ordinary dot product. We call E' the *output embedding* matrix. The basic autoencoder training objective to minimize over the dictionary can then be formulated as

$$J_r(\theta', \theta) = - \sum_{w \in \mathcal{V}^{\mathcal{K}}} \log p_{\theta'}(D_w | f_\theta(D_w)).$$

2.3 Consistency penalty

We introduced 3 different embeddings: a) *definition embeddings* h, produced by the definition encoder, are the embeddings we are ultimately interested in computing; b) *input embeddings* E are used by the encoder as inputs; c) *output embeddings* E' are compared to definition embeddings to yield a probability distribution over the words in the definition. We propose a soft weight-tying scheme that brings the input embeddings closer to the definition embeddings. We call this term a *consistency* penalty because its goal is to to ensure that the embeddings used by the encoder (input embeddings) and the embeddings produced by the encoder (definition embeddings) are consistent with each other. It is implemented as

$$J_p(\theta) = \sum_{w \in \mathcal{V}^{\mathcal{D}} \cap \mathcal{V}^{\mathcal{K}}} d(E_w, f_\theta(D_w))$$

where d is a distance. In our experiments, we choose d to be the Euclidian distance. The penalty is only applied to some words because $\mathcal{V}^{\mathcal{D}} \neq \mathcal{V}^{\mathcal{K}}$. Indeed, some words are defined but are not used in definitions and some words are used in definitions but not defined. In particular, inflected words are not defined. To balance the two terms, we introduce two hyperparameters $\lambda, \alpha \geq 0$ and the complete objective is

$$J(\theta', \theta) = \alpha J_r(\theta', \theta) + \lambda J_p(\theta).$$

We call the model *CPAE*, for *Consistency Penalized AutoEncoder* when $\alpha > 0$ and $\lambda > 0$ (see Figure 1).[3]

The consistency penalty is a cheap proxy for dealing with the circularity found in dictionary definitions. We want the embeddings of the words in definitions to be compositionally built from their definition as well. The recursive process of fetching definitions of words in definitions does not terminate, because all words are defined using other words. To counter that, our model uses input embeddings that are brought closer to definition embeddings and vice versa in an asynchronous manner.

Moreover, if λ is chosen large enough, then $E_w \approx f_\theta(D_w)$ after optimisation. This means that the definition embedding for w is close enough to the corresponding input embedding to be used by the encoder for producing other definition embeddings for other words. In that case, the model could enrich its vocabulary by computing embeddings for new words and consistently reusing them as inputs for defining other words.

Finally, the consistency penalty can be used to leverage pretrained embeddings and bootstrap the learning process. For that purpose, the encoder's input embeddings E can be fixed to pretrained embeddings. These provide targets to the encoder but

---

[2] We have tried using a LSTM decoder but it didn't yield good representations. It might overfit because the set of dictionary definitions is small. Also, using teacher forcing, we condition on ground-truth words, making it easier to predict the next words. More work is needed to address these issues.

[3] Our implementation is available at `https://github.com/tombosc/cpae`

also helps the encoder to produce better definition embeddings in virtue of using input embeddings that already contain meaningful information.

To summarize, the consistency penalty has several motivations. Firstly, it deals with the fact that the recursive process of building representation of words out of definitions does not terminate. Secondly, it is a way to enrich the vocabulary with new words dynamically. Finally, it is a way to integrate prior knowledge in the form of pretrained embeddings.

In order to study the two terms of the objective in isolation, we introduce two special cases. When $\lambda = 0$ and $\alpha > 0$, the model reduces to *AE* for *Autoencoder*. When $\alpha = 0$ and $\lambda > 0$, we retrieve *Hill's model*, as presented by Hill et al. (2015).[4] Hill's model is simply a recurrent encoder that uses pretrained embeddings as targets so it only makes sense in the case we use fixed pretrained embeddings.

3 Related work

3.1 Extracting lexical knowledge from dictionaries

There is a long history of attempts to extract and structure the knowledge contained in dictionaries. Amsler (1980) studies the possibility of automatically building taxonomies out of dictionaries, relying on the syntactic and lexical regularities that definitions display. One relation is particularly straightforward to identify: it is the *is a* relation that translates to hypernymy. Dictionary definitions often contain a *genus* which is the hypernym of the defined word, as well as a *differentia* which differentiates the hypernym from the defined word. For example, the word "hostage" is defined as "a prisoner who is held by one party to insure that another party will meet specified terms", where "prisoner" is the genus and the rest is the differentia.

To extract such relations, early works by Chodorow et al. (1985) and Calzolari (1984) use string matching heuristics. Binot and Jensen (1987) operate at the syntactic parse level to detect these relations. Whether based on the string representation or the parse tree of a definition, these rule-based systems have helped to create large lexical databases. We aim to reduce the manual labor involved in designing the rules and directly obtaining representations from raw definitions.

3.2 Improving word embeddings using lexical resources

Postprocessing methods for word embeddings use lexical resources to improve already trained word embeddings irrespective of how they were obtained. When it is used with fixed pretrained embeddings, our method can be seen as a postprocessing method.

Postprocessing methods typically have two terms for trading off conservation of distributional information that is brought by the original vectors with the new information from lexical resources. There are two main ways to preserve distributional information: Attract-Repel (Vulić and Mrkšić, 2017), retrofitting (Mrkšić et al., 2017) and our method control the distance between the original vector and the postprocessed vector so that the new vector does not drift too far away from the original vector. Counter-Fitting (Mrkšić et al., 2016) and dict2vec (Tissier et al., 2017) ensure that the neighbourhood of a vector in the original space is roughly the same as the neighbourhood in the new space.

Finally, methods differ by the nature of the lexical resources they use. To our knowledge, dict2vec is the only technique that uses dictionaries. Other postprocessing methods use various data from WordNet: sets of synonyms and sometimes antonyms, hypernyms, and hyponyms. For instance, Lexical Entailment Attract-Repel (LEAR) uses all of these (Vulić and Mrkšić, 2017). Other methods rely on paraphrase databases (Wieting et al., 2016).

3.3 Dictionaries and word embeddings

We now turn to the most relevant works that involve dictionaries and word embeddings.

Dict2vec (Tissier et al., 2017) combines the word2vec skip-gram objective (predicting all the words that appear in the context of a target word) with a cost for predicting related words. These related words either form *strong pairs* or *weak pairs* with the target word. Strong pairs have a greater influence in the cost. They are pairs of words that are in the neighbourhood of the target word in the

---

[4] It is not exactly their model as we use Euclidian distance instead of the cosine distance or the ranking loss. They also explore several variants where the input embeddings are learned, which we didn't find to produce any improvement. We haven't experimented with the ranking loss, but the cosine distance does not seem to improve over Euclidian. Finally, they also use a simple encoder that averages word vectors, which we found to be inferior.

original embedding, as well as pairs of words for which the definitions make reference to each other. Weak pairs are pairs of words where only one word appears in the definition of the other. Unlike dict2vec, our method can be used as either a standalone or a postprocessing method (when used with pretrained embeddings). It also focuses on handling and leveraging the recursivity found in dictionary definitions with the consistency penalty whereas dict2vec ignores this aspect of the structure of dictionaries.

Besides dict2vec, Hill et al. (2015) train neural language models to predict a pretrained word embedding given a definition. Their goal was to learn a general-purpose sentence encoder useful for downstream tasks. Noraset et al. (2017) propose the task of generating definitions based on word embeddings for interpretability purposes. Our model unifies these two approaches into an autoencoder. However, we have a different goal: that of creating or improving word representations. Their methods assume that pretrained embeddings are available to provide either targets or inputs, whereas our model is unsupervised, and the use of pretrained embeddings is optional.

Bahdanau et al. (2017) present a related model that produces embeddings from definitions such that it improves performance on a downstream task. By contrast our approach is used either stand-alone or as as a postprocessing step, to produce general-purpose embeddings at a lesser computational cost. The core novelty is the way we leverage the recursive structure of dictionaries.

Finally, Herbelot and Baroni (2017) also aim at learning representations for word embeddings in a few shots. The method consists of fine-tuning word2vec hyperparameters and can learn in one or several passes, but it is not specifically designed to handle dictionary definitions.

4 Experiments

4.1 Setup

We experiment on English to benefit from the many evaluation benchmarks available. The dictionary we use is that of WordNet (Fellbaum, 1998). WordNet contains graphs of linguistic relations such as synonymy, antonymy, hyponymy, etc. but also definitions. We emphasize that our method trains exclusively on the definitions and is thus applicable to any electronic dictionary.

However, in order to *evaluate* the quality of embeddings on unseen definitions, WordNet relations comes in handy: we use the sets of synonyms to split the dictionary into a train set and a test set, as explained in Section 7. Moreover, WordNet has a wide coverage and high quality, so we do not need to aggregate several dictionaries as done by Tissier et al. (2017). Finally, WordNet is explicitly made available for research purposes, therefore we avoid technical and legal difficulties associated with crawling proprietary online dictionaries.

We do not include part of speech tags that go with definitions. WordNet does not contain function words but contains homonyms of function words. We filter these out.

4.2 Similarity and relatedness benchmarks

Evaluating the learned representations is a complex issue (Faruqui et al., 2016). Indeed, different evaluation methods yield different rankings of embeddings: there is no single embedding that outperforms others on all tasks (Schnabel et al., 2015) and thus no single best evaluation method.

We focus on intrinsic evaluation methods. In particular, we study how different models trade off similarity and relatedness. We use benchmarks which consist of pairs of words scored according to some criteria. They vary in terms of annotation guidelines, number of annotators, selection of the words, etc. To evaluate our embeddings, we score each pair by computing the cosine similarity between the corresponding word vectors. Then the predicted scores and the ground truth are ranked and the correlation between the ranks is measured by Spearman's ρ. We leave aside analogy prediction benchmarks as they suffer from many problems (Linzen, 2016; Rogers et al., 2017).

We adopt one of the methods proposed by Faruqui et al. (2016) and use separate datasets for model selection. We choose the development set to be the development set of SimVerb3500 (Gerz et al., 2016) and MEN (Bruni et al., 2014), the only benchmarks with a standard train/test split.

We justified our emphasis on the similarity relation in Section 1: capturing this relation remains a challenge, and we hypothesize that dictionary data should improve representations in that respect. The model selection procedure reflects that we want embeddings specialized in similarity. To do that, we set the validation loss as a weighted mean which weights SimVerb twice as MEN.

4.3 Baselines

The objective function presented in section 2 gives us 3 different models: CPAE, AE, and Hill's model. The objective of CPAE comprises the sum of the objective of Hill's model and of AE. We compare the CPAE model to both of these to evaluate the individual contribution of the two terms to the performance. In addition, when we use external corpora to pretrain embeddings, we compare these models to dict2vec and retrofitting. The hyperparameter search is described in Appendix C.

The test benchmarks for the similarity relation includes SimLex999 (Hill et al., 2016) and more particularly SimLex333, a challenging subset of SimLex999 which contains only highly related pairs but in which similarity scores vary a lot. For relatedness, we use MEN (Bruni et al., 2014), RG (Rubenstein and Goodenough, 1965), WS353 (Finkelstein et al., 2001), SCWS (Huang et al., 2012), and MTurk (Radinsky et al., 2011; Halawi et al., 2012). The evaluation is carried out by a modified version of the Word Embeddings Benchmarks project.[5] Conveniently, all these benchmarks contain mostly lemmas, so we do not suffer too much from the problem of missing words.[6]

5 Results in the dictionary-only setting

In the first evaluation round, we train models only using a single monolingual dictionary. This allows us to check our hypothesis that dictionaries contain information for capturing the similarity relation between words.

Our baselines are regular distributional models: GloVe (Pennington et al., 2014) and word2vec (Mikolov et al., 2013). They are trained on the concatenation of defined words with their definitions.

Such a formatting introduces spurious co-occurrences that do not otherwise appear in free text. But these baselines are not designed for dictionaries and cannot deal with their particular structure.

We compare these models to the autoencoder model without (AE) and with (CPAE) the consistency penalty. In this setting, we cannot use Hill's model as it requires pretrained embeddings as targets. We also trained an additional CPAE model with pretrained word2vec embeddings trained on the concatenated definitions. The results are presented in Table 1.

GloVe is outperformed by word2vec by a large margin so we ignore this model in later experiments. Word2vec captures more relatedness than CPAE (+10.7 on MEN-t, +13.5 on MT, +13.2 on WS353) but less similarity than CPAE. The difference in the nature of the relations captured is exemplified by the scores on SimLex333. This subset of SimLex999 focuses on pairs of words that are very related but that can be either similar or dissimilar. On this subset, CPAE fares better than word2vec (+13.1).

The consistency penalty improves performance on every dataset. This penalty provides targets to the encoder, but these targets are themselves learned and change during the learning process. The exact dynamics of the system are unknown. It can be seen as a regularizer because it puts strong weight-sharing constraints on both types of embeddings. It also resembles *bootstrapping* in reinforcement learning, which consists of building estimates of values functions on top of over estimates (Sutton and Barto, 1998).

The last model is the CPAE model that uses the word2vec embeddings pretrained on the dictionary data. This combination not only equals other models on some benchmarks but outperforms them, sometimes by a large margin (+6.3 on SimLex999 and +7.5 on SimVerb3500 compared to CPAE, +6.1 on SCWS, +5.4 on MT compared to word2vec). Thus, the two kinds of algorithms are complementary through the different relationships that they capture best. The pretraining helps in two different ways, by providing quality input embeddings and targets to the encoder. The pretrained word2vec targets are already remarkably good. That is why the chosen consistency penalty coefficient selected is very high ($\lambda = 64$). The model can pay a small cost and deviate from the targets in order to encode information about the definitions.

To sum up, dictionary data contains a lot of data relevant to modeling the similarity relationship. Autoencoder based models learn different relationships than regular distributional methods. The consistency penalty is a very helpful prior and regularizer for dictionary data, as it always helps,

---

[5] Original project available at `https://github.com/kudkudak/word-embeddings-benchmarks`, modified version distributed with our code.

[6] Missing words are not removed from the dataset, but they are assigned a null vector.

	Development		Similarity			Relatedness				
	SV-d	MENd	SL999	SL333	SV-t	RG	SCWS	MENt	MT	353
GloVe	12.0	54.8	19.8	-9.1	7.8	57.5	46.8	57.0	49.4	44.4
word2vec	35.2	62.3	34.5	16.0	36.4	65.7	54.5	59.9	56.1	**61.9**
AE	34.9	42.7	35.6	26.8	32.5	64.8	50.2	42.2	38.6	41.4
CPAE ($\lambda = 8$)	42.8	48.5	39.5	29.1	34.8	67.1	54.3	49.2	42.6	48.7
CPAE-P ($\lambda = 64$)	**44.1**	**65.1**	**45.8**	**30.9**	**42.3**	**72.0**	**60.4**	**63.8**	**61.5**	61.3

Table 1: **Positive effect of the consistency penalty and word2vec pretraining.** Spearman's correlation coefficient $\rho \times 100$ on benchmarks. Without pretraining, autoencoders (AE and CPAE) improve on similarity benchmarks while capturing less relatedness than distributional methods. The consistency penalty (CPAE) helps even without pretrained targets. Our method, combined with pretrained embeddings on the same dictionary data (CPAE-P), significantly improves on every benchmark.

regardless of what relationship we focus on. Finally, our model can drastically improve embeddings that were trained on the same data but with a different algorithm.

6 Improving pretrained embeddings

We have seen that CPAE with pretraining is very efficient. But does this result generalizes to other kind of pretraining data? To answer this question, we experiment using embeddings pretrained on the first 50 million tokens of a Wikipedia dump, as well as the entire Wikipedia dump. We compare our method to existing postprocessing methods such as dict2vec and retrofitting, which also aims at improving embeddings with external lexical resources.

Retrofitting, which operates on graphs, is not tailored for dictionary data, which consists in pairs of words along with their definitions. We build a graph where nodes are words and edges between nodes correspond to the presence of one of the words into the definition of another. Obviously, we lose word order in the process.

The results for the small corpus are presented in Table 2. By comparing Table 2 with Table 1, we see that word2vec does worse on similarity than when trained on dictionary data, but better on relatedness. Both dict2vec and retrofitting improve with regards to word2vec on similarity benchmarks and seem roughly on par. However, dict2vec fails to improve on relatedness benchmarks, whereas retrofitting sometimes improves (as in RG, MEN, and MT), sometimes equals (SCWS) and does worse (353).

We do an ablation study by comparing Hill's model and AE with CPAE. Recall that Hill's model lacks the reconstruction cost while AE lacks the consistency penalty. Firstly, CPAE always improves over AE. Thus, we confirm the results of the previous section on the importance of the consistency penalty. In that setting, it is more obvious why this penalty helps, as it now provides pretrained targets to the encoder. Secondly, CPAE improves over Hill on all similarity benchmarks by a large margin (+12.2 on SL999, +13.7 on SL333, +16.1 on SV3500). It is sometimes slightly worse on relatedness benchmarks (-3.3 on MEN-t, -5.6 on MT), other times better or equal. We conclude that both terms of the CPAE objective matter.

We see identical trends when using the full Wikipedia dump. As expected, CPAE can still improve over even higher quality embeddings by roughly the same margins. The results are presented in Appendix D.

Remarkably, the best model among all our experiments is CPAE in Table 1 and uses only the dictionary data. This supports our hypothesis that dictionaries contain similarity-specific information.

7 Generalisation on unseen definitions

A model that uses definitions to produce word representations is appealing because it could be extremely data-efficient. Unlike regular distributional methods which iteratively refine their representation as occurrences accumulate, such a model could output a representation in one shot. We now evaluate CPAE in a setting where some definitions are not seen during training.

The dictionary is split into train, validation (for early stopping) and test splits. The algorithm for splitting the dictionary puts words in batches. It ensures two things: firstly, that words which share at least one definition are in the same batch, and secondly, that each word in a batch is associated

	Development		Similarity			Relatedness				
	SV-d	MEN-d	SL999	SL333	SV-t	RG	SCWS	MEN-t	MT	353
word2vec	21.7	71.1	33.2	6.9	21.2	68.5	**65.8**	71.5	61.1	**65.3**
retrofitting	28.5	**71.6**	36.9	14.1	26.1	**78.9**	65.7	**74.4**	**62.8**	60.7
dict2vec	26.3	63.5	36.2	15.5	22.0	69.2	63.8	63.9	54.9	60.8
Hill	26.9	63.3	27.7	12.9	21.7	72.9	58.4	64.1	54.0	52.3
AE	33.5	47.0	33.1	20.4	32.5	66.0	52.0	46.4	40.2	43.7
CPAE ($\lambda = 4$)	**39.5**	60.8	**39.9**	**26.6**	**37.8**	69.7	59.2	60.8	48.4	55.6

Table 2: **Improving pretrained embeddings computed on a small corpus**. Spearman's correlation coefficient $\rho \times 100$ on benchmarks. All methods use pretrained embeddings. All methods (except maybe Hill) manage to improve the embeddings. Retrofitting outperforms dict2vec and efficiently specializes for relatedness. CPAE outperforms AE and allows to trade off relatedness for similarity.

	Similarity			Relatedness			
	999	333	SV-t	353	MEN	SCWS	MT
All	36.5	27.7	36.9	43.7	48.8	53.2	41.5
Train	40.0	28.5	36.7	46.9	53.4	57.1	42.9
Test	27.5	25.4	38.5	42.5	44.1	40.3	39.1

Table 3: **One pass generalisation.** Spearman's correlation coefficient $\rho \times 100$ on benchmarks. The model is CPAE (without pretrained embeddings). All: all pairs in the benchmarks. Train: pairs for which both words are in the training or validation set. Test: pairs which contain at least one word in the test set. Correlation is lower for test pairs but remains strong ($\rho > 0.3$): the model has good generalisation abilities.

with all its definitions. We can then group batches to build the training and the test sets such that the test set does not contain synonyms of words from the other sets. We sort the batches by the number of distinct definitions they contain. We use the largest batch returned by the algorithm as the training set: it contains mostly frequent and polysemous words. The validation and the test sets, on the contrary, contain many multiword expressions, proper nouns, and rarer words. More details are given in Appendix B.1.

We train CPAE only on the train split of the dictionary, with randomly initialized input embeddings. Table 3 presents the same correlation coefficients as in the previous tables but also distinguishes between two subsets of the pairs: the pairs for which all the definitions were seen during training (*train*) and the pairs for which at least one word was defined in the test set (*test*). Unfortunately, there are not enough pairs of words which both appear in the test set to be able to compute significant correlations. On small-sized benchmarks, correlation coefficients are sometimes not significant so we do not report them (when p-value > 0.01).

The scores of CPAE on the test pairs are quite correlated with the ground truth: except on SimLex999 and SCWS, there is no drop in correlation coefficients between the two sets. The scores of Hill's model follow similar trends, but are lower on every benchmark so we do not report them. This shows that recurrent encoders are able to generalize and produce coherent embeddings as a function of other embeddings in one pass.

8 Conclusion and future work

We have focused on capturing the similarity relation. It is a challenging task which we have proposed to solve using dictionaries, as definitions seem to encode the relevant kind of information.

We have presented an alternative for learning word embeddings that uses dictionary definitions. As a definition autoencoder, our approach is self-contained, but it can alternatively be used to improve pretrained embeddings, and includes Hill's model (Hill et al., 2015) as a special case.

In addition, our model leverages the inherent recursivity of dictionaries via a consistency penalty, which yields significant improvements over the vanilla autoencoder.

Our method outperforms dict2vec and retrofitting on similarity benchmarks by a quite large margin. Unlike dict2vec, our method can be used as a postprocessing method which does not require going through the original pretraining corpus, it has fewer hyperparameters, and it generalises to new words.

We see several directions for future work.

Firstly, more work is needed to evaluate the representations on other languages and tasks.

Secondly, solving downstream tasks requires representations for the inflected words as well. We have set aside this issue by focusing on benchmarks involving lemmas. To address it in future work, we might want to split word representations into a lexical and a morphological part. With such a split representation, we could postprocess only the lexical component, and all the words, whether inflected or not, would benefit from this. This seems desirable for postprocessing methods in general and would make them more suitable for synthetic languages.

Thirdly, dictionary defines every sense of words, so we could produce one embedding per sense (Chen et al., 2014) (Iacobacci et al., 2015). This requires potentially complicated modifications to our model as we would need to disambiguate senses inside each definition. However, some class of words might benefit a lot from such representations, for example words that can be used as different parts of speech.

Lastly, a more speculative direction could be to study iterative constructions of the set of embeddings. As our algorithm can generalize in one shot, we could start the training with a small set of words and their definitions and iteratively broaden the vocabulary and refine the representations without retraining the model. This could be useful in discovering a set of semantic primes from which one can define all the other words (Wierzbicka, 1996).

Acknowledgements

We thank NSERC and Facebook for financial support, Calcul Canada for computational resources, Dzmitry Bahdanau and Stanisław Jastrzębski for contributing to the code on which the implementation is based, the developers of Theano (Theano Development Team, 2016), Blocks and Fuel (van Merriënboer et al., 2015) as well as Laura Ball for proofreading.

References

Robert Alfred Amsler. 1980. The structure of the merriam-webster pocket dictionary.

Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2017. Learning to compute word embeddings on the fly. *CoRR*, abs/1706.00286.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Jean-Louis Binot and Karen Jensen. 1987. A semantic expert using an online standard dictionary. In *Proceedings of the 10th international joint conference on Artificial intelligence-Volume 2*, pages 709–714. Morgan Kaufmann Publishers Inc.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, 49(2014):1–47.

Nicoletta Calzolari. 1984. Detecting patterns in a lexical data base. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics*, pages 170–173. Association for Computational Linguistics.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035.

Martin S Chodorow, Roy J Byrd, and George E Heidorn. 1985. Extracting semantic hierarchies from a large on-line dictionary. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pages 299–304. Association for Computational Linguistics.

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

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2014. Retrofitting Word Vectors to Semantic Lexicons. *arXiv:1411.4166 [cs]*. ArXiv: 1411.4166.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. *arXiv preprint arXiv:1707.06556*.

Felix Hill, Kyunghyun Cho, Sébastien Jean, Coline Devin, and Yoshua Bengio. 2014. Embedding word similarity with neural machine translation. *CoRR*, abs/1412.6448.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2015. Learning to understand phrases by embedding the dictionary. *arXiv preprint arXiv:1504.00548*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

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

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 95–105.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? *EMNLP 2015*.

Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies. *arXiv preprint arXiv:1606.07736*.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *arXiv preprint arXiv:1706.00374*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California. Association for Computational Linguistics.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition Modeling: Learning to Define Word Embeddings in Natural Language. In *AAAI*, pages 3259–3266.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Olivier Picard, Alexandre Blondin-Massé, Stevan Harnad, Odile Marcotte, Guillaume Chicoisne, and Yassine Gargouri. 2009. Hierarchies in dictionary definition space. *arXiv preprint arXiv:0911.5703*.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Anna Rogers, Aleksandr Drozd, and Bofang Li. 2017. The (Too Many) Problems of Analogical Reasoning with Word Vectors. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 135–148.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Magnus Sahlgren. 2008. The distributional hypothesis. *Italian journal of linguistics*, 20(1):33–54.

Tobias Schnabel, Igor Labutov, David M Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings.

Richard S Sutton and Andrew G Barto. 1998. Introduction to reinforcement learning.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Julien Tissier, Christophe Gravier, and Amaury Habrard. 2017. Dict2vec: Learning Word Embeddings using Lexical Dictionaries. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 254–263.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Ivan Vulić and Nikola Mrkšić. 2017. Specialising word vectors for lexical entailment. *arXiv preprint arXiv:1710.06371*.

Anna Wierzbicka. 1996. *Semantics: Primes and universals: Primes and universals*. Oxford University Press, UK.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2016. Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 236–246, Berlin, Germany. Association for Computational Linguistics.

Spot the Odd Man Out: Exploring the Associative Power of Lexical Resources

Gabriel Stanovsky[1,2] and Mark Hopkins[3]

[1]Allen Institute for Artificial Intelligence, Seattle, WA
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA
[3]Department of Mathematics, Reed College, Portland, OR
gabis@allenai.org, hopkinsm@reed.edu

Abstract

We propose Odd-Man-Out, a novel task which aims to test different properties of word representations. An Odd-Man-Out puzzle is composed of 5 (or more) words, and requires the system to choose the one which does not belong with the others. We show that this simple setup is capable of teasing out various properties of different popular lexical resources (like WordNet and pre-trained word embeddings), while being intuitive enough to annotate on a large scale. In addition, we propose a novel technique for training multi-prototype word representations, based on unsupervised clustering of ELMo embeddings, and show that it surpasses all other representations on all Odd-Man-Out collections.

1 Introduction

Correctly disambiguating the sense of a polysemous word (e.g., "***spring** is a beautiful season*" versus "*John was ready to **spring** into action*") is a crucial part of various NLP tasks, such as translation, question answering, or textual entailment. The state-of-the-art, and the de-facto common practice for essentially all of these tasks, involves neural networks (see (Goldberg, 2015) for a recent survey), which are commonly initialized with pre-trained word vectors, such as Word2Vec (Mikolov et al., 2013a) or GloVe (Pennington et al., 2014).

These widely-used representations often significantly improve performance in downstream tasks, as they are able to leverage large amounts of unstructured data. However, most of the popular collections of word embeddings assign only one vector to each word, thus shifting the burden of word disambiguation to deeper, task-specific layers, which commonly rely on data of much smaller scales.

While there has been a significant body of work around *sense embeddings* (i.e., embedding senses, instead of lexical units), evaluating such representations remains the subject of debate (Faruqui et al., 2016; Gladkova and Drozd, 2016).

In this work, we propose a new evaluation task called Odd-Man-Out. The goal of an Odd-Man-Out puzzle is simple. Given a set of words[1] like **cherry, orange, apple, grass, grape**, the objective is to identify the word that does not belong (here, the answer is **grass**, because it is not a fruit). While there are often multiple relationships among the words, we will show that non-experts typically agree on the odd-man-out, and are able to generate hard puzzles on a large scale, using a novel crowdsourcing protocol.

Following the creation of this large test set, we conduct a thorough analysis of the ability of various lexical resources to correctly solve the task. In doing so, we embrace the suggestion of Gladkova and Drozd (2016), which argue for "a shift from absolute ratings of word embeddings towards more exploratory evaluations that would aim not for generic scores, but for identification of strengths and weaknesses of embeddings", and conduct rigorous analysis of each representation. Overall, we find that all lexical resources are prone to miss associations which are intuitive for humans, leaving ample room for future improvement. Moreover, we show empirical evidence that lexical resources that do not account for polysemy are handicapped by this weakness.

Finally, we propose a new sense embedding technique, which leverages the recent introduction of ELMo embeddings (Peters et al., 2018) by performing unsupervised clustering over a large unstructured corpus. We show that this new resource surpasses all other baselines on various Odd-Man-Out datasets.

---

[1] In this paper, we use the term "word" loosely to also include the multi-word expressions like "fire engine" and "magnifying glass."

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1533–1542
Brussels, Belgium, October 31 - November 4, 2018.

We make all our code and models publicly available.[2]

2 Existing Evaluation Methods

In this section, we briefly survey some existing evaluation methods for lexical resources, and discuss their pros and cons.

2.1 Word Similarity

The *word similarity* task (Rubenstein and Goodenough, 1965) has been a dominant approach to assess word vector quality. In this task, systems are required to score the similarity of two words on a numeric scale, sometimes without context and sometimes in a sentential context (Huang et al., 2012). For instance, the pair (tiger, mammal) has a similarity score of 6.85 (out of 10) on the WordSim dataset (Finkelstein et al., 2001).

A common criticism (Faruqui et al., 2016; Gladkova and Drozd, 2016) of the word similarity task is that "similarity" is subjective, and conflates several different potential relationships between words. For instance, (Faruqui et al., 2016) questions why the pair (cup, coffee) should be considered more similar than (car, train), as it is according to the WordSim dataset.

Odd-man-out puzzles naturally disambiguate the nebulous concept of "similarity", because each puzzle implicitly defines a specific relationship. The words "car" and "train" may be similar, but this similarity is irrelevant is the context of a puzzle like **car, train, checkered flag, racetrack, pit stop**.

2.2 Analogies

Analogies like "king is to queen as man is to..." (Mikolov et al., 2013b; Jurgens et al., 2012) are related to the odd-man-out task. Analogies, however, are best suited to particular relationships, such as hypernym → hyponym (which are the subject of extensive research, e.g., (Shwartz et al., 2016)), while odd-man-out puzzles can capture a broader range of associations (see for instance, the auto-racing puzzle from the previous section). Analogies are also more subject to ambiguity, since the premise can involve only two words. For instance, the puzzle "cherry is to strawberry as grass is to..." could refer to the fact that cherry and strawberry are both fruits, both red, or both red fruits. Odd-man-out puzzles provide a simple way of reducing ambiguity: adding more choices to the puzzle.

2.3 Word Sense Disambiguation and Induction

Word sense disambiguation (Navigli, 2009) is a popular way to evaluate polysemous word representations. The common criticism is that systems are rewarded based on their ability to classify words according to a fixed inventory of senses, whose granularity is regarded by some as too coarse and others as too fine. An alternative is word sense induction (Manandhar et al., 2010), which allows systems to cluster word senses without an agreed-upon sense inventory. However, there is not an obvious evaluation metric. The two metrics used in SemEval 2010 Task 14 (Manandhar et al., 2010) yielded highly divergent system rankings.

2.4 Word Context Relevance

A recent evaluation method is Word Context Relevance (Arora et al., 2016; Sun et al., 2017a). The task is to identify whether a particular bag of words is "relevant" to a target word. For instance, "tie" is considered relevant to the bag "winner, score, tied, completion, identical, results, sports," but irrelevant to the bag "domestic, hog, pig, culinary, eaten, cooked, fat". This task has the attractiveness of being a simple binary evaluation, but demands only that a model can identify a broad sense of relatedness, not the ability to pinpoint specific relationships.

2.5 Lexical Substitution

Lexical substitution tasks (McCarthy and Navigli, 2007; Biemann, 2013; Kremer et al., 2014), in which the task is to determine whether one word can replace another word in a particular context, are effective, but are restricted in the kind of relationships they can test (mainly synonymy).

3 Odd Man Out Datasets

In this section, we describe the creation of several Odd-Man-Out datasets. We begin by describing a small-scale, curated annotation, then show how to scale the annotation using crowdsourcing techniques. In the subsequent sections, we use these datasets to explore the properties of a wide array of lexical resources.

---

[2] `https://github.com/gabrielStanovsky/odd-man-out`

3.1 Expert annotation

To create our first odd-man-out datasets, we used categories from the card game Anomia.[3] Anomia is a slapjack-style game in which players attempt to name instances of a particular category as quickly as possible. Categories include: "percussion instrument", "Mexican food", and "Michael Jackson song". A team of 4 people, trained in-house, created one odd-man-out puzzle for each category, yielding a total of 404 puzzles. For instance, the puzzle corresponding to "percussion instrument" is **clarinet, drum, xylophone, tambourine, cymbals**, where clarinet is the odd-man-out. The puzzle-writing team attempted to make the odd-man-out similar to the category, e.g. clarinet is a musical instrument but not a percussion instrument.

We divided the puzzles into 2 sets, one for puzzles comprised of common words (like the "percussion instruments" example above), and one for puzzles comprised of proper nouns (like the puzzle derived from the category "Michael Jackson songs"). Each set contained exactly 202 puzzles. We further divided each of these sets into a development set (used for error analysis) of size 100 and a test set of size 102. Henceforth, we will refer to these four datasets as ANOMIACOMMONDEV, ANOMIACOMMONTEST,ANOMIAPROPERDEV, andANOMIAPROPERTEST.

There are two potential pitfalls in creating these puzzles. First, the underlying category may not be detectable by humans. Second, we may inadvertently create an ambiguous puzzle with multiple odd-men-out. For instance, given the category "vegetables", we might create the puzzle **grass, celery, cucumber, carrot, lettuce**, for which the intended odd-man-out is **grass**, but the answer **carrot** is also a reasonable odd-man-out for the category "things that are green."

To ensure the answerability of our puzzles, we administered the ANOMIACOMMONDEV to five college-educated individuals, three of whom were native English speakers. Both groups did well. The native speakers averaged 95.3% accuracy, while the non-native speakers averaged 91.5%.

3.2 Crowdsourcing

Following the success of the curated annotation, as described above, we devised a crowdsourcing protocol to achieve annotation at a much larger scale. In this section, we describe this protocol and show that the Odd-Man-Out task is intuitive enough to be collected and annotated with high agreement by non-trained annotators, using a small expert seed annotation. This enables us to efficiently obtain a large set of 500K *hard* "training" samples on a small budget of $400. Finally, we also validate a set of 1000 puzzles, which we use in following sections for testing purposes.

Crowdsourcing protocol Our semi-automatic crowdsourcing protocol starts from a seed set of about 250 polysemous words,[4] and is composed of the following three consecutive stages:

1. **Seed categorization (expert)**: given a polysemous word w, we ask expert annotators to come up with at least two categories c_1, c_2, which describe w. For example, given w = "*bat*", the corresponding categories can be c_1 = "*nocturnal mammal*" and c_2 = "*baseball instrument*". This categorization was performed by the authors of this paper, and was done in about 3 person hours.

2. **Category expansion (crowdsourced)**: Given a seed word and a corresponding category (w, c) turkers are asked to provide five more examples of the category c , which are similar to, but different than, w. For example, given ("*bat*", "*nocturnal mammal*") turkers are expected to provide examples such as "*beaver*", "*badger*", or "*hedgehog*", while for ("*bat*", "*baseball instrument*"), proper answers include "*ball*" or "*cap*". We used the Amazon Mechanical Turk (AMT)[5] platform, paying 15¢ per elicitation.

3. **Puzzle creation (automatic)**: For each polysemous seed word w, belonging to categories c_1, c_2, we automatically create Odd-Man-Out puzzles by concatenating to w each possible combination of three words from c_1 (c_2), while the intended odd-man-out is any word from c_2 (c_1). For example, using the previous

---

[3] https://boardgamegeek.com/boardgame/142271/anomia-party-edition

[4] `https://en.wikipedia.org/wiki/List_of_true_homonyms`, following (Sun et al., 2017b).

[5] https://www.mturk.com

seed word and categories, we get the following puzzle: ("*cap*", "*bat*", "*beaver*", "*badger*", "*hedgehog*"). Creating puzzles by combining words from c_1 and c_2 in this manner ensures that task is challenging, as it requires to disambiguate the polysemous word (e.g., "*bat*").

Performing the process described above with the 249 polysemous seed words yielded a large corpus of $497,365$ puzzles, thanks to the combination process, with a vocabulary of $1,312$ different words. To create a gold high quality test corpus, and to assess the validity of this semi-automatic process, we sampled 1000 puzzles and administered each puzzle to three annotators on AMT, paying 6¢ per response. We found that for 84.3% of the instances there was a majority vote agreement on the intended odd-man-out word. We refer to this set of 843 puzzles as CROWDSOURCED843. Table 1 shows examples of the crowdsourced puzzles.

From an examination of turkers disagreement, we can attribute the vast majority to noise in one of the annotation stages, e.g., turkers which provide examples of the wrong sense of the seed word, resulting in invalid puzzles (where all of the words belong to the same category), or turkers marking the wrong odd-man-out, thus invalidating an otherwise correct puzzle.

Overall, the cost of the annotation and validation was below $500, for a large high-quality annotated resource and a smaller gold standard test corpus. Both corpora are made available.[6]

4 Taxonomy-Based Solvers

In this section, we show how to create odd-man-out solvers for taxonomies like WordNet (Miller, 1992).

Define a *taxonomy* as a triple (V, E, L), where (V, E) is a directed acyclic graph, and L maps each vertex V to a string. A simple example is shown in Figure 1, where the each vertex v is labeled with $L(v)$.

We create an odd-man-out solver from a taxonomy as follows:

- The *specificity* of a vertex v is defined as the reciprocal of the number of its descendants. For instance, the specificity of any leaf in Figure 1 is 1, while the specificity of the vertex labeled "element" is $\frac{1}{6}$.

Category	Puzzle
construction	**crane**, *pelican*, excavator, hoist, upraise
guitar part	**fret**, *crying*, inlays, truss rod, neck
alcoholic drinks	**gin**, *poker*, vodka, tequila, wine
card games	**gin**, *vodka*, bridge canasta, uno

Table 1: Selection of examples from the crowdsourced Odd-Man-Out dataset. The polysemous seed word appears in bold, while the correct answer is in italics. The last two examples demonstrate how a polysemous word (*gin*) can participate in two different puzzles.

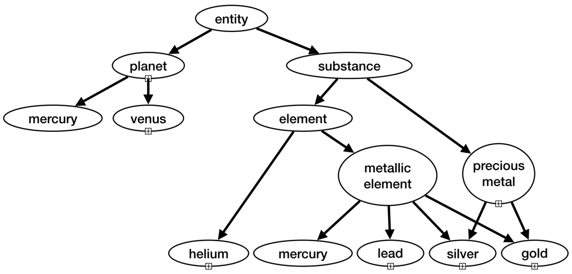

Figure 1: Example taxonomy.

- Given an odd-man-out puzzle $w_1, ..., w_n$, the *explanation* of word w_k is the vertex v of highest specificity such that: (i) for each word w_j such that $j \neq k$, there exists some descendent v' of v where $L(v') = w_j$, (ii) there does not exist a descendent v' of v such that $L(v') = w_k$. For instance, the explanation of **helium** with respect to the puzzle **helium, mercury, lead, silver, gold** is the node labeled "metallic element."
- If some word does not have an explanation, or if there is no word whose explanation is uniquely most specific, then the solver abstains from answering. Otherwise, the solver returns the word with the most specific explanation.

Using WordNet 3.0 (Miller, 1992; Fellbaum, 1998) as the taxonomy, the solver correctly solves

---

[6] `https://github.com/gabrielStanovsky/odd-man-out`

Puzzle	Explanation
chicken, screwdriver, margarita, mimosa, daiquiri	mixed drink
silver, steel, brass, bronze, pewter	alloy
canoe, school, flock, herd, pack	animal group
nightgown, afternoon, morning, evening, midnight	abstraction
king, *president*, queen, prince, princess	leader
dinghy, *crab*, boat, canoe, raft	travel (verb)

Table 2: Some of the WordNet solver's answers and explanations on AnomiaCommonDev. The solver's answer is in bold, while the correct answer is in italics.

40.6% of the AnomiaCommon puzzles, answering incorrectly for 13.4% and abstaining for 46.0% of the puzzles. Unsurprisingly (since WordNet focuses on common words), the WordNet solver gets only 1 out of 202 AnomiaProper puzzles, abstaining from all the rest. On Crowdsourced843, the solver correctly solves 22.0% of the puzzles, answering 15.1% incorrectly and abstaining from the rest.

4.1 Error Analysis

A nice property of the taxonomy-based solver is that it provides an explanation for its answer (i.e. the "explanation" vertex defined above). In Table 2, we show some of the WordNet solver's answers and explanations (specifically we show the name of the WordNet synset corresponding to the explanation vertex). For 85.4% of its correct answers, the solver also returns the correct explanation (like the "mixed drink," "alloy," and "animal group" explanations). In the handful of cases when it does not, it finds either an incorrect or an overly vague explanation (like the "abstraction" explanation) that still results in the correct answer.

Typically, the incorrect answers result from incompleteness in the WordNet taxonomy. For instance, the word "king" is not a hyponym of "leader," even though "president," "queen," "prince," and "princess" all are. In the bottommost puzzle of Table 2, the error results from "raft" not being considered a watercraft (or a conveyance of any kind) by WordNet. Because of this, the solver finds a more tenuous connection between "crab" and three of the other words, leveraging a rare sense of "crab" as a verb meaning "to move like a crab."

5 Embedding-Based Solvers

In this section, we create odd-man-out solvers from collections of word embeddings. Specifically, we experiment with two types of embeddings: (1) traditional word embeddings, which map words to a single vector representation (Subsection 5.2), and (2) sense embeddings, which map words to a *set* of vectors, each pertaining to a different sense of the word (Subsection 5.3).

5.1 Embedding Evaluation Framework

For the sake of comparison, we use a common framework to create odd-man-out solvers based on traditional word embeddings and sense embeddings. In this framework, given n puzzle options, we find the subset of $n-1$ words of maximal similarity (given some similarity score), and return the excluded word as the odd-man-out.

Specifically, define an *embedding* as a function that maps every word to a *set* (possibly empty) of real vectors. Note that this definition is general enough to allow for sense embeddings as well as traditional word embeddings, for which the returned sets are either singletons or the empty set (in case the word is not in the embedding's vocabulary).

Next, given a similarity score σ that maps any vector pair to a real number and an embedding E, define the *cohesion* $\kappa_{\sigma,E}$ of a set of words W as:

$$\kappa_{\sigma,E}(W) = \max_{v_1 \in E(w_1),\dots,v_n \in E(w_n)} \sum_{1 \le i < j \le n} \sigma(v_i, v_j) \tag{1}$$

Cohesion is undefined if any word $w \in W$ maps to the empty set. Throughout this paper we employ the widely used *cosine similarity* as our similiarity score.

Finally, given an embedding E and puzzle $W = \langle w_1, \dots, w_n \rangle$, we create an odd-man-out solver as follows:

1. If $E(w_i)$ is the empty set for any puzzle choice $w_i \in W$, then the solver abstains from answering.

2. Otherwise, the solver returns the word $\hat{w} \in W$ whose omission from the puzzle word set

Embedding Map	Training	AnomiaCommon			AnomiaProper			Crowdsourced		
	Tokens	C	W	A	C	W	A	C	W	A
ELMo clusters ($K = 5$)	1B + 2B*	**76.7**	**13.9**	**9.4**	**42.6**	**17.8**	**39.6**	**55.5**	**18.8**	**25.6**
w2v.googlenews	100B	61.9	25.2	12.9	40.1	14.9	45.0	46.3	28.8	24.9
glove.commoncrawl2	840B	60.9	23.8	15.4	32.2	14.4	53.5	47.1	28.4	24.6
glove.commoncrawl1	42B	57.4	29.2	13.4	30.7	17.8	51.5	40.1	36.3	23.7
glove.wikipedia	6B	54.5	24.3	21.3	29.2	10.9	59.9	42.7	29.0	28.4
Neelakantan	1B	35.2	25.7	39.1	18.3	14.4	67.3	32.6	27.3	40.2
w2v.freebase	100B	22.3	28.7	49.0	34.2	14.9	51.0	9.9	11.3	78.3
WordNet	-	40.6	13.4	46.0	0.5	0.0	99.5	22.0	15.1	63.0

Table 3: Performance (%Correct, %Wrong, %Abstained) of the different odd-man-out solvers on the AnomiaTest and crowdsourcing datasets, using different vector directories. * We used ELMo clusters pretrained on 1B tokens and clustered on a 2B token Wikipedia dump.

yields maximal cohesion:

$$\hat{w} = \operatorname*{argmax}_{w_i \in W} \kappa_\sigma(W \setminus \{w_i\}) \qquad (2)$$

5.2 Word Embeddings Solvers

Table 3 shows the results of embedding-based solvers on the Anomia and crowdsourced datasets, using several different pre-trained embedding maps. We find the best performance on AnomiaCommon and AnomiaProper using the word2vec vectors trained on 100 billion tokens from the Google News corpus.[7]

An analysis of the best embedding-based solver (w2v.googlenews) on AnomiaCommon-Dev shows that a high proportion of incorrect answers were polysemous (20 out of the 27 incorrect answers). A selection of these are shown in Table 4. We observe that for the "cocktails" puzzle, the word2vec solver zeroes in on "screwdriver" (whose dominant sense is the tool, not the cocktail), and for the "types of lettuce" puzzle, the solver selects "iceberg" (whose dominant sense is "large floating block of ice," not the lettuce).

To provide additional evidence of this bias, we came up with 5 cocktails whose dominant sense is the cocktail itself (mint julep, mai tai, mojito, martini, and bloody mary) and 5 polysemous cocktails (old fashioned, hurricane, cosmopolitan, zombie, and Manhattan). We then replaced "screwdriver" in the "cocktails" puzzle with each of these options and solved the resulting puzzle with the w2v.googlenews solver. In all 5 monosemous instances, the correct answer of "chicken" was selected. In 4 of the 5 polysemous instances (the exception being "Manhattan"), the polysemous replacement was selected. We repeated this experiment with the "groups of animals" puzzle, and again all 5 monosemous replacements yielded the correct answer, while 3 of 5 polysemous replacements were incorrectly chosen by the solver.

In a more rigorous experiment, we randomly generated 500 puzzles in the following way. Given a category (hyponym), we came up with 10 instances (hypernyms) of that category. For example, the category "type of transport" yielded: train, car, bus, airplane, helicopter, boat, ferry, taxi, tram, and monorail. We did this for 10 categories, yielding a total of 100 words. From these 10 lists, we randomly generated 500 puzzles by sampling 4 words from one list and 1 word from another. We call this puzzle set Hypernyms500.

The w2v.googlenews solver performs impressively on Hypernyms500, getting 90.8% correct, with only 46 incorrect answers (and no abstentions). Tellingly however, only 3 words are responsible for 67% of the incorrect answers: saw (as a kind of tool), lead (as a kind of metal), and rose (as a kind of flower). All three of these words have at least one dominant alternate sense. Given that there are 100 possible incorrect answers (which all appear with roughly equal frequency in the puzzles), the fact that only 3 of them comprise 67% of the incorrect answers suggests that the inability of vector directories to model multiple senses is an Achilles heel.

5.3 ELMo Sense Vectors

The previous analysis suggests that embeddings that explicitly model multiple senses may be important for the odd-man-out task. In this section,

---

[7] https://code.google.com/archive/p/word2vec

Category	Puzzle
cocktails	**screwdriver**, *chicken*, margarita mimosa, daiquiri
mammals	**bear**, *cobra*, cat dog, leopard
nocturnal animals	**bat**, *robin*, owl raccoon, coyote
military ranks	**private**, *judo*, general corporal, colonel
whales	**blue**, *grizzly*, humpback orca, beluga
groups of animals	**school**, *canoe*, flock herd, pack
shades of blue	**navy**, *amber*, cobalt azure, sky
aquatic animals	**seal**, *horse*, whale manatee, dolphin
bird sounds	**tweet**, *snore*, chirp cluck, quack
types of lettuce	**iceberg**, *serrano*, romaine butterhead, bibb
political parties	**green**, *garden*, democratic republican, libertarian

Table 4: Selection of errors made by the w2v.googlenews solver on ANOMIACOMMON-DEV, indicating a tendency to choose polysemous words as the odd-man-out. The incorrect selection is in bold, while the correct answer is in italics.

we investigate this further.

Background: sense vectors There is a significant literature on how to learn a one-to-many mapping from words to vector representations. An early paradigm (Reisinger and Mooney, 2010; Huang et al., 2012; Liu et al., 2015; Wu and Giles, 2015) took a 2-pass approach. First, they clustered contexts of a target word like "bank." For instance, "the bank had an ATM" and "I got money from the bank" might fall into one cluster, while "I fished at the river bank" might fall into a second cluster. Second, they annotated each instance of the target word with its sense cluster and then learned a standard one-to-one vector mapping from the annotated corpus. For instance, they would learn vector representations using the sentences "the bank-1 had an ATM," "I got money from the bank-1," and "I fished at the river bank-2." Other researchers (Neelakantan et al., 2014; Tian et al., 2014; Chen et al., 2014; Li and Jurafsky, 2015; Bartunov et al., 2016) focused on modifying the vector learning model itself, typically the skip-gram model (Mikolov et al., 2013b), to directly learn multiple embeddings for each word. Additional work focused on using the technique of retrofitting (Faruqui et al., 2014) to adapt pretrained word vectors into sense vectors using auxiliary resources like WordNet (Jauhar et al., 2015) or parallel corpora (Ettinger et al., 2016). Other work (Guo et al., 2014; Suster et al., 2016; Upadhyay et al., 2017) used parallel corpora as the main signal for learning sense vectors.

Background: ELMo Recently, Peters et al. (2018) introduced the concept of Embeddings from Language Models (ELMo). ELMo dynamically represents each word based on the context with which it appears, achieved by representing a word in a sentence using its representation from a pretrained bidirectional Language Model (biLM), encoded using a bi-directional RNN (Schuster and Paliwal, 1997). Subsequently, the same word may get different representations in different contexts. For example, the representation of "*bank*" may differ between "*She fished by the river **bank***" and "*She deposited her check at the nearest **bank***". While ELMo embeddings were recently proven extremely beneficial in various sentence-level tasks (e.g., semantic role labeling, question answering, and textual entailment), many lexical tasks require the interpretation of words out of context, to which ELMo cannot be readily applied. We suggest to port ELMo's effectiveness back to the context-free setting, and propose a first approach for doing so.

Unsupervised ELMo clustering We compute pre-trained ELMo embedding (using the AllenNLP framework (Gardner et al., 2018)) *in context* of sentences in a large corpus C, while recording the observed representation $r_{w,s} \in \mathbb{R}^d$ of each word w, along with the corresponding context in which it appeared, i.e., the sentence $s \in C$. The result of this process is an embedding space *per word*, $r_w = \{r_{w,s} \mid s \in C, w \in s\}$. Ideally, similar senses of w would appear in similar contexts, and would therefore be closer in r_w, while different senses would be further apart. Following this intuition, we cluster each word to K clusters, using the k-means algorithm (Lloyd, 1982), whereby every vector in a cluster is interpreted as pertaining to the same sense of w. We collect the centroid

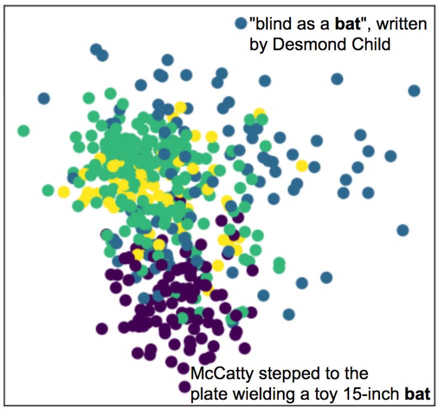

Figure 2: An example of a 2d projection of an ELMo embedding space for the word "bat" with 4 senses (denoted by different colors). Two example sentence excerpts appear next to their respective cluster.

vector w_i (where $1 \le i \le K$) of each cluster as a "sense vector" of w. Using $K = 1$, we get a single representation per word, averaging the different senses of w, comparable to "traditional" word embeddings. Conversely, we can get senses of arbitrarily fine granularity by increasing the number of clusters. These sense representations can then be readily plugged in to the embedding solver, as described in Subsection 5.1. We computed the sense vectors using a 2 billion token (97 Million sentences) Wikipedia dump from January 2018, which was extracted using WikiExtractor,[8] and tokenized with the SpaCy 2.0 toolkit (Honnibal and Montani, 2017).[9] These pre-computed vectors, which are readily applicable for other tasks, are made publicly available. Figure 2 depicts the resulting embedding space and clustering for the word "bat".

Evaluation We start by estimating an ideal value for K (the number of clusters). In Figure 3 we evaluate the performance of the ELMo clustering method on the expert development set, using different values for K, compared to the second best performing vectors on that dataset (Word2Vec). Several observations can be made based on this analysis. First, using $K = 1$, Word2Vec outperforms ELMo. This may be explained due to Word2Vec's larger training set (100B tokens versus only 1B). However, as K increases, ELMo clusters outperform the baseline,

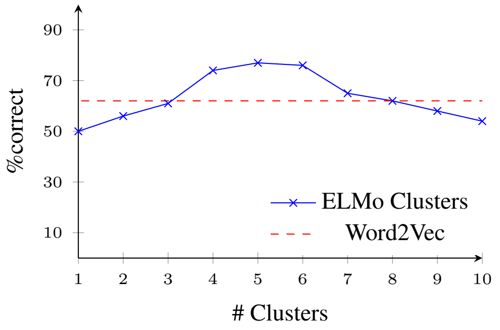

Figure 3: Performance of the ELMo cluster model (blue line) by the varying number of cluster (K), compared to the best performing baseline (word2vec), in the red horizontal line.

reaching its maximum performance at $K = 5$, indicating that a finer level of sense granularity is beneficial for focusing on the intended word sense. Inversely, having too many clusters hurts performance. This may be due to over-specification, harming the sense generalization obtained by a smaller number of clusters. Based on this tuning, we fixed the value of K to 5, and repeated the experiments from Section 5 (see the first row in Table 3). ELMo sense vectors clearly outperform all previous baselines on all Odd-Man-Out datasets. This improved performance can be attributed both to ELMo's better ability to capture context, as well as to the finer sense representation, as opposed to the single representation per word in most of the other baselines. We also evaluated another publicly available collection of sense embeddings (Neelakantan et al., 2014), but it did not perform on par with solvers based on conventional single-sense embeddings.

6 Discussion: Hypernyms vs. Other Associations

The attentive reader may have wondered why the w2v.googlenews solver performed so much better on the HYPERNYMS dataset (over 90% correct) than on the ANOMIACOMMON dataset (approximately 62% correct). Hypothesizing that embedding-based solvers can identify hypernym-hyponym relationships more easily than other associations, we created another odd-man-out dataset using the same methodology as HYPERNYMS.

This time, given a *color*, we came up with 8 to 13 objects that are typically associated with that color. For example, the category "yellow" yielded

---

[8] `https://github.com/attardi/wikiextractor`

[9] `https://spacy.io`

(among others): taxi, canary, corn kernel, daffodil, lemon, sun, and school bus. We did this for 10 colors. From these 10 lists, we again randomly generated 500 puzzles by sampling 4 words from one list and 1 word from another. We call this puzzle set COLORS500.

The w2v.googlenews solver performed considerably worse on COLORS500 than on HYPERNYMS500. Out of the puzzles it attempted, it guessed only 30% correctly (compared to the random chance baseline of 20%). There are several possible reasons why this dataset may be more difficult for an embedding-based solver. Possibly colors are not easily identifiable using embeddings built from language cues (maybe people do not often explicitly talk about how lemons are yellows). But perhaps it is also true that using the cosine similarity of the embeddings is not a good way to spot non-fundamental properties (like color) that link a group of words. We leave open how best to identify more oblique relationships.

7 Conclusion

We presented a new task for the evaluation of lexical resources, the Odd-Man-Out task. We showed that the task can be annotated reliably on a large scale. Following the creation of several Odd-Man-Out datasets, we conducted analyses showing that current word representations are suboptimal, especially in the presence of polysemous words. We concluded with a novel ELMo clustering sense-embedding technique which surpasses all baselines on the Odd-Man-Out task.

Acknowledgements

We would like to thank Santosh Divvala, Fereshteh Sadeghi, Marco Valenzuela, and Isaac Cowhey, who initiated this project during an AI2 hackathon.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. Linear algebraic structure of word senses, with applications to polysemy. *CoRR*, abs/1601.03764.

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry P. Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *AISTATS*.

Christian Biemann. 2013. Creating a system for lexical substitutions from scratch using crowdsourcing. *Language Resources and Evaluation*, 47:97–122.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*.

Allyson Ettinger, Philip Resnik, and Marine Carpuat. 2016. Retrofitting sense-specific word vectors using parallel text. In *HLT-NAACL*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. In *RepEval@ACL*.

Christiane D. Fellbaum. 1998. Book reviews: Wordnet: An electronic lexical database.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20:116–131.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Anna Gladkova and Aleksandr Drozd. 2016. Intrinsic evaluations of word embeddings: What can we do better? In *RepEval@ACL*.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *COLING*.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*.

Sujay Kumar Jauhar, Chris Dyer, and Eduard H. Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *HLT-NAACL*.

David Jurgens, Saif Mohammad, Peter D. Turney, and Keith J. Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *SemEval@NAACL-HLT*.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us - analysis of an "all-words" lexical substitution corpus. In *EACL*.

Jiwei Li and Daniel Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *EMNLP*.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI*.

Stuart P. Lloyd. 1982. Least squares quantization in pcm. *IEEE Trans. Information Theory*, 28:129–136.

Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer Pradhan. 2010. Semeval-2010 task 14: Word sense induction and disambiguation. In *SemEval@ACL*.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *SemEval@ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A. Miller. 1992. Wordnet: A lexical database for english. *Commun. ACM*, 38:39–41.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41:10:1–10:69.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew D McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *HLT-NAACL*.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8:627–633.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2389–2398, Berlin, Germany. Association for Computational Linguistics.

Yifan Sun, Nikhil Rao, and Weicong Ding. 2017a. A simple approach to learn polysemous word embeddings. *CoRR*, abs/1707.01793.

Yifan Sun, Nikhil Rao, and Weicong Ding. 2017b. A simple approach to learn polysemous word embeddings. *CoRR*, abs/1707.01793.

Simon Suster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. In *HLT-NAACL*.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*.

Shyam Upadhyay, Kai-Wei Chang, Matt Taddy, Adam Tauman Kalai, and James Y. Zou. 2017. Beyond bilingual: Multi-sense word embeddings using multilingual context. In *Rep4NLP@ACL*.

Zhaohui Wu and C. Lee Giles. 2015. Sense-aware semantic analysis: A multi-prototype word representation model using wikipedia.

Neural Multitask Learning for Simile Recognition

Lizhen Liu[†], **Xiao Hu**[†], **Wei Song**[†*], **Ruiji Fu**[‡], **Ting Liu**[§], **Guoping Hu**[‡]
[†]Information Engineering, Capital Normal University, Beijing, China
[‡]iFLYTEK Research, Beijing, China
[§]Harbin Institute of Technology, Harbin, China
{liz_liu7480,xiaohu,wsong}@cnu.edu.cn, {rjfu, gphu}@iflytek.com, tliu@ir.hit.edu.cn

Abstract

Simile is a special type of metaphor, where comparators such as *like* and *as* are used to compare two objects. Simile recognition is to recognize simile sentences and extract simile components, i.e., the *tenor* and the *vehicle*. This paper presents a study of simile recognition in Chinese. We construct an annotated corpus for this research, which consists of 11.3k sentences that contain a comparator. We propose a neural network framework for jointly optimizing three tasks: simile sentence classification, simile component extraction and language modeling. The experimental results show that the neural network based approaches can outperform all rule-based and feature-based baselines. Both simile sentence classification and simile component extraction can benefit from multitask learning. The former can be solved very well, while the latter is more difficult.

1 Introduction

A metaphor is a figure of speech that describes an object or action in a way that isn't literally true. Metaphors are common in human language. Shutova and Teufel (2010) reported that 241 among 760 sentences in an annotated corpus contain a metaphor. The use of metaphors helps to explain an idea or realize rhetorical effects through an analogical procedure. Metaphor analysis has been drawn more attention for expanding current natural language processing (NLP) to high-level semantic tasks (Carbonell, 1980).

Metaphors reflect creative thought of humans. On the other hand, inferring the meaning of a metaphor has to integrate background knowledge, which makes it difficult to automatically recognize metaphors in language. Previous work on metaphor recognition mainly depends on linguistic cues (Goatly, 2011) and selectional preference violation on a pair of concepts (Fass, 1991) or their domains (Mason, 2004). The domains can be created by knowledge bases such as WordNet (Mason, 2004) or based on automatic clustering (Shutova et al., 2010).

In this paper, we focus on a special type of metaphor—simile. A simile is a figure of speech that directly compares two things using connecting words such as *like*, *as*, *than* in English and "像" or "犹如" in Chinese. Due to the use of such comparators, it is much easier to locate similes compared with locating other types of metaphors. As a result, it is possible to collect and annotate large scale of simile sentences and investigate data driven simile recognition. This task is to find simile sentences and extract simile components, i.e., the *tenor* and the *vehicle*. The mined simile structures can potentially be used to support general metaphor analysis, where large scale training data is lacking.

However, simile recognition is still challenging due to the diversity of syntactic roles of a word and the distinction between metaphorical and literal comparisons. As shown in Table 1, a sentence containing a comparator may not trigger a simile. It is necessary to analyze the relationship between meanings of concepts. And It is also difficult to define a complete set of rules to extract the objects to be compared with high accuracy and coverage.

This paper presents an end-to-end neural network framework for simile sentence recognition. Specifically, we make following contributions:

- We build a dataset consisting of 11.3k sentences containing a frequently used comparator "像" for simile recognition in Chinese, which can support data-driven approaches. In contrast to English, datasets on simile or

---

[*] corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1543–1553
Brussels, Belgium, October 31 - November 4, 2018.

1. 这个[孩子]$_{tenor}$ 壮得**像**一头[牛]$_{vehicle}$ This [boy]$_{tenor}$ is as strong *as* a [bull]$_{vehicle}$	Simile
2. 这个孩子长得**像**爸爸 The boy looks *like* his father	Literal
3. 他拍拍叔叔的肩膀,**像**是告诉他不要难过 He patted his uncle *as if* telling him not to be sad	Literal
4. **像**他这样的学生，应该更加努力 The students *like* him should work even harder	Literal

Table 1: Sentences that contain the comparator “像”.

metaphor analysis are relatively less in Chinese. This dataset provides a new resource for related research. [1]

- We propose a neural multitask learning framework jointly optimizing three tasks: simile sentence classification, simile component extraction and language modeling. Simile classification is to determine whether a sentence with a comparator contains a simile, without knowing exactly what the tenor and the vehicle are. Simile component extraction aims to locate the tenor and the vehicle in a simile sentence. Intuitively, the two tasks should benefit each other. We design our model to enhance interactions between the two tasks. We also borrow the idea of Rei (2017) by incorporating a language modeling task, which attempts to predict neighbor words. All three tasks consider the whole sentence so that rich context information is involved.

- We conduct comprehensive experiments. The results demonstrate that the neural end-to-end framework is superior to feature-based and rule-based baselines and every single model can benefit from multitask learning. Simile sentence classification can be solved very well, while simile component extraction is more chellenging. With multitask learning enhanced classifier and extractor, a classification-then-extraction method achieves the best performance for simile component extraction.

2 Related Work

2.1 Metaphor/Simile Analysis

Metaphor analysis becomes active in recent years. The tasks include metaphor recognition, metaphor explanation and metaphor generation (Shutova et al., 2013; Veale, 1995; Jang et al., 2016).

Simile is a special type of metaphor with the comparator and it is relatively easier to locate metaphorical parts. Niculae and Danescu-Niculescu-Mizil (2014) aimed to distinguish a comparison from figurative or literal in product reviews using a series of linguistic cues as features. It is similar to simile sentence classification. So we take it as a baseline. In their work, they assumed that the components can be correctly recognized. In our work, we use an automated component extractor instead.

Syntactic patterns are often used for extracting potential simile components and semantic analysis is then used to distinguish similes from literal comparisons (Niculae and Yaneva, 2013; Niculae, 2013). The main limitation is that such pattern based method is difficult to deal with sentences with complex structures. As a result, the coverage is relatively small.

Qadir et al. (2016) used syntactic structures, dictionary definitions, statistical cooccurrence, and word embedding vectors to infer implicit properties in similes. Qadir et al. (2015) also built a classifier with lexical features, semantic features, and sentiment features to infer the affective polarity of simile in twitters. Veale and Hao (2007) and Veale (2012a) utilized knowledge generated by similes to deal with metaphor and irony, and Veale (2012b) built a lexical stereotype model from similes. These work demonstrates the wide applications of simile recognition.

In Chinese, Li et al. (2008) proposed a feature-based method for simile recognition. Their evaluation was done on a small dataset. The annotated data in this work is much larger.

2.2 Multitask Learning for NLP

Many researchers have proposed to jointly learn multiple tasks with shared representations (Collobert and Weston, 2008). Improvements are reported on joint models between closely related tasks, such as text classification (Liu et al., 2016), POS tagging and parsing (Zhang and Weiss, 2016), parsing and named entity recognition (NER) (Finkel and Manning, 2010), NER and linking (Luo et al., 2015), extraction of entities and relations (Miwa and Bansal, 2016). Bingel and Søgaard (2017) offered a systematic view of relations between different tasks.

---

[1] The dataset is at `https://github.com/cnunlp/Chinese-Simile-Recognition-Dataset`

3 Task and Data

3.1 Task Description

A metaphor is as a matter of cross-domain mappings in conceptual structure which are expressed in language. Lakoff and Johnson (2008) explains it as a mapping between the *target* and the *source*, corresponding to the terms *tenor* and *vehicle*. The tenor is the subject to which attributes are ascribed, while the vehicle is the object whose attributes are borrowed.

Simile can be seen as a special type of metaphor, which is signaled by explicit markers such as *like* or *as* in English and “像” or “犹如” in Chinese. We call such words **comparators** (Hanks, 2012). Notice that a sentence containing a comparator doesn’t guarantee that it is a simile sentence. Consider the examples in Table 1. All sentences contain the comparator “像”. The first two sentences form comparison structures, but the first one triggers a cross-domain concept mapping, while the second one is a literal comparison. The word “像” in the third sentence means *as if*, rather than forming a comparison, and the comparator in the fourth sentence is to give examples.

Simile Recognition task can be defined as: Given a sentence containing a comparator, determine whether it is a simile sentence, if so, extract the *tenor* and the *vehicle* from it.

Simile recognition involves two subtasks.

Simile Sentence Classification (SC). For a sentence containing a comparator, determine whether the comparator triggers a metaphorical comparison, in another word, whether the sentence is a simile sentence.

Simile Component Extraction (CE). For a simile sentence, extract text spans that are corresponding to the *tenor* and the *vehicle* objects respectively.

Both tasks have a realistic significance. Simile can be seen as a rhetorical device for making thoughts or expressions more vivid. Simile sentence classification could be used to provide a signal to evaluate rhetorical effects of writings. Simile component extraction is potentially useful for building cognitive knowledge base.

3.2 Data

We construct a dataset from Chinese student essays written in Mandarin Chinese. We focus on the comparator “像”, which is the most often used simile comparator in Chinese. The data annotation involves the following steps: (1) we sampled sentences that contain the word “像” from a sentence index built on more than 20,000 student essays; (2) we asked two annotators to label every sentence as a simile sentence or not; (3) the annotators further annotated boundaries of simile components, the tenor and the vehicle, in simile sentences.

Two points are important for annotation: (1) the definition of simile and (2) the boundary of simile components.

We desire that a simile sentence should satisfy at least two standards. First, there exists explicit tenor and vehicle, which are from different semantic domains. Second, the tenor and vehicle should have similar properties. We provided a manual and positive/negative examples to annotators. Even so, there are still fuzzy cases. The two annotators first labeled a sample of 200 sentences independently and then we measured their agreement and asked them to discuss the disagreements. After discussion, they labeled another set of sampled sentences to check whether they really reached a consensus. This process iterated several times. Their final inner-annotator agreement on simile sentence classification can reach to 91%. Finally, they labeled all sentences in the whole dataset.

Next, the annotators should further label simile components in simile sentences, which are usually noun phrases. Boundaries of simile components are required to be annotated as compact as possible until they can’t be simplified any more. In most cases, we asked the annotators to label the head noun phrase without a modifier. A modifier often plays a role as a *shared property*. For example, in the sentence 男孩的脸像一个红苹果(*The boy’s face is like a red apple*), 脸(*face*) and 苹果(*apple*) would be annotated as the tenor and vehicle respectively, while 男孩的(*boy’s*) and 红(*red*) are not included. In contrast, in the sentence 天气像孩子的脸说变就变(*The weather is like a child’s face, which changes unpredictably*), 孩子的脸*(child’s face)* is preferred to 脸(*face*) as a component, because 脸(*face*) alone can’t capture the property well. We used one annotator’s annotation of 200 sentences as the gold answer and the other’s annotation as the prediction to computer the F_1 score, which is 93.57% on all components and 90.7% on tenors and 96.47% on vehicles.

Table 2 shows the basic statistics of our dataset.

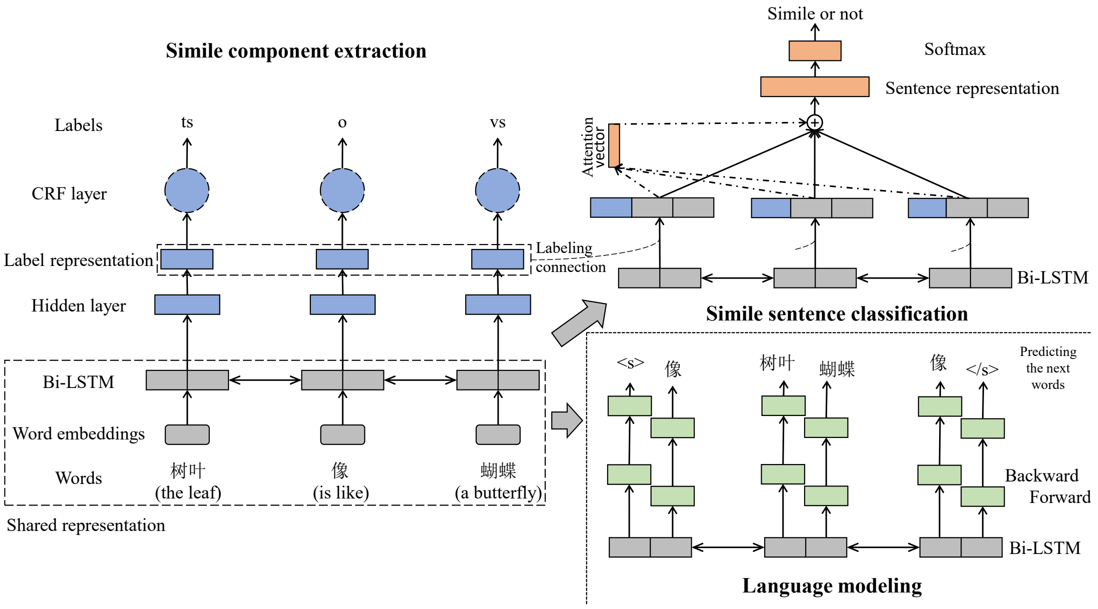

Figure 1: The proposed multitask learning framework, which jointly optimizes three tasks.

#Sentence	11337
#Simile sentence	5088
#Literal sentence	6249
#Token	334k
#Tenor	5183
#Vehicle	5119
#Unique tenor concept	1680
#Unique vehicle concept	1972
#Tenor-vehicle pair	5214
#Unique tenor-vehicle pair	4521
Avg. #token per tenor	1.033
Avg. #token per vehicle	1.056
Avg. #token per sentence	29.47
Avg. #pair per simile sentence	1.024

Table 2: Statistics of the annotated simile dataset

4 Multitask Learning Approach

4.1 Motivation

Intuitively, the two subtasks in simile recognition can benefit each other and the interactions between them should not be ignored. If the component extractor knows that a sentence contains a simile, it would be more confident to extract the tenor and the vehicle. On the other hand, if the component extractor tells the classifier that the tenor and the vehicle likely exist, the classifier gets additional information for decision.

Therefore, we propose a multitask learning approach to combine them. Our approach jointly optimizes three tasks: simile sentence classification, simile component extraction and language modeling. Language modeling is used as auxiliary task, which can help capture local information. Figure 1 illustrates the main framework, which is based on neural networks. We will first explain the representation layers that are shared by multiple tasks, and then introduce separate prediction layers for individual tasks.

4.2 Shared Representation

Word embedding layer. We first map words to dense distributed word embeddings. Since our dataset is not so large, we make use of pre-trained word embeddings, which are trained on a much larger corpus with Word2Vec toolkit (Mikolov et al., 2013).

Sentence representation layer. Recurrent neural networks (RNNs) have become the natural choice for handling sequential data to capture long-range dependencies. Given a sentence $X = (x_1, x_2, ..., x_n)$ containing n words as an input, the RNNs produce $H = (h_1, h_2, ..., h_n)$ as the hidden states to represent the semantic of partial sequence so far. Recently, Long Short Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997) has been proved more effective in various NLP tasks. Therefore, we use LSTM as the basic memory cell. At time step t, LSTM takes the hidden state from the previous time step and the word embedding from the current step as input, and produces a new hidden state, as shown in Formula 1.

$$\overrightarrow{h_t} = LSTM(x_t, LSTM(\overrightarrow{h_{t-1}})) \quad (1)$$

The LSTM architecture is sensitive to word order, and the bidirectional LSTM (Schuster and Paliwal, 2002) allows model to look arbitrarily far at both the past and the future for the sake of grasping the whole sentence. Noted the forward LSTM as $\overrightarrow{h}$, and the backward as $\overleftarrow{h}$. Bidirectional LSTM concatenates the forward and backward states as the representation at the tth time step, i.e., $h_t = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right]$.

4.3 Task 1: Simile Component Extraction

We view simile component extraction as a sequence labeling problem. We convert the annotated dataset to IOBES scheme (indicating *Inside, Outside, Beginning, Ending, Single*) (Ratinov and Roth, 2009). We use different prefixes to distinguish the tenor and the vehicle components. For example, *tb* and *vb* indicate the beginning of a *tenor* and a *vehicle* respectively.

4.3.1 Neural Sequence Labeling Model

Conditional Random Field (CRF) (Lafferty et al., 2001) is a standard solution in such scenario to exploit the dependency among labels.To further make use of the dense representation of words, we build a CRF layer on the shared representation layers following (Lample et al., 2016).

Formally, $H = (h_1, h_2, ..., h_n)$ is a sequence of hidden states produced by the bidirectional LSTM for a sentence X and $y = (y_1, y_2, ..., y_n)$ is the tag sequence, $y_i \in L$ and $|L| = k$. Define $\psi(H, y)$ as the score of the sequence.

$$\psi(H, y) = \sum_{t=0}^{n} A_{y_t, y_{t+1}} + \sum_{t=1}^{n} P_{t, y_t} \tag{2}$$

where $A \in \mathbb{R}^{k \times k}$ is a transition matrix and $A_{y_t, y_{t+1}}$ records the score of a transition from current label y_t to next label y_{t+1}; $P = (p_1, ..., p_n) \in \mathbb{R}^{n \times k}$ is the emission matrix and P_{t, y_t} represents the score of assigning tag y_t to x_t.

Here, h_t is the tth hidden state that is as assigned by the bidirectional LSTM. It is first mapped to a hidden layer through a feedforward layer. After a non-linear activation transition $\tanh$, the output of the hidden layer is mapped to a k-dimension vector p_t, through another feedforward layer.

$$p_t = W_p \cdot \tanh(W_t h_t) \tag{3}$$

where W_t and W_p are parameter matrixes. p_t can be seen as a tag score vector given the current word without considering context words.

Taking the whole state sequence into account, the probability of tag sequence y given sentence X is:

$$p(y|H) = \frac{\psi(H, y)}{\sum_{\widetilde{y} \in Y} e^{\psi(H, \widetilde{y})}} \tag{4}$$

where Y indicates all possible sequences. Learning algorithm attempts to optimize the model by maximizing the log-likelihood of correct tag sequence. Thus, the loss function is

$$\begin{aligned} E_{ce} &= -log(p(y|H)) \\ &= -\psi(H, y) + \log \sum_{\widetilde{y} \in Y} e^{\psi(H, \widetilde{y})} \end{aligned} \tag{5}$$

4.4 Task 2: Simile Sentence Classification

The second task is simile sentence classification. To fully exploit contextual information, we consider all words in a sentence. For each word, instead of using hidden state h_t only, we combine h_t and its score vector p_t as a representation s_t:

$$s_t = [h_t; p_t]$$

Since p_t is directly related to the component extraction task, this labeling connection operation increases the interaction between the two tasks.

However, words in a sentence should not contribute the same for classification. Intuitively, the words corresponding to the tenor or the vehicle or near comparators should be more important. Therefore, we introduce the attention mechanism, which was firstly proposed for neural machine translation (Bahdanau et al., 2014).

Given the sequence of expanded word representations $S = (s_1, s_2, ..., s_n)$, the attention vector is computed via:

$$\alpha = softmax(\tanh(W_\alpha S)) \tag{6}$$

where W_α is a parameter matrix. The semantic representation of the sentence is:

$$r = \alpha^T \cdot S \tag{7}$$

This representation is fed into an activation and a softmax layer to generate the probability distribution. The loss function is the negative log-likelihood of the correct classification tag:

$$E_{sc} = -\log(p(y|s)). \tag{8}$$

4.5 Task 3: Language Modeling

Although LSTM can capture long dependencies, simile structure may be more related to local contexts. In many cases, the comparator and the vehicle are near, and similes often have some collocations involving the comparator such as "像...一样(the same as)" or "就像(just like)". As a result, we attempt to emphasize such local information.

Inspired by (Rei, 2017), we also incorporate language modeling as an auxiliary task. For each word, we let the model predict the next word. In our case, the representation of each word h_t is firstly mapped into a low dimension vector space through a nonlinear transform.

$$\overrightarrow{m_t} = \tanh\left(\overrightarrow{W_m} \cdot \overrightarrow{h_t}\right) \tag{9}$$

And the vehicle word is predicted by maximizing the probability of the specific next word, which is generated by a softmax layer.

$$P\left(w_{t+1}|\overrightarrow{m_t}\right) = softmax\left(\overrightarrow{W_q} \cdot \overrightarrow{m_t}\right) \tag{10}$$

where, $\overrightarrow{m_t}$ indicates the forward language modeling specific features. W_m, W_q are trainable parameters.

The loss function for a sequence is defined as the sum of the negative log-likelihood of the predicted words.

$$\overrightarrow{E_{lm}} = -\sum_{t=1}^{n-1} \log\left(P\left(w_{t+1}|\overrightarrow{m_t}\right)\right) \tag{11}$$

We can also predict the previous words in the same way and get another loss function noted as $\overleftarrow{E_{lm}}$. The losses in double direction are summed to be the loss function for language modeling task.

$$E_{lm} = \overrightarrow{E_{lm}} + \overleftarrow{E_{lm}} \tag{12}$$

4.6 The Final Loss Function

The final loss function for each sentence is a weighted sum of task-specific loss functions.

$$E = \gamma \cdot E_{lm} + \delta \cdot E_{ce} + \epsilon \cdot E_{sc} \tag{13}$$

where γ, δ, ϵ are non-negative weights, which are used to control the importance of three tasks. In experiments, they are hyper-parameters assigned beforehand, and we constrain the sum of γ, δ and ϵ to one.

5 Evaluation

5.1 Settings

The dataset is randomly divided into 5 folds, 4 of which are used as training set and validation set (80% for training, 20% for validation), and the rest one fold is used as test set. All models were trained on the training set. The best hyperparameters were gained based on the results on the validation set. The results reported were all evaluated on the test set.

We conduct word segmentation, part-of-speech (POS) tagging and dependency parsing with HIT-LTP[2].The word embeddings were pre-trained using Word2Vec (Mikolov et al., 2013), on a large essay corpus crawled from the web. We adopt the Theano framework (Theano Development Team, 2016) to implement neural network models.

The dimension of word embeddings is 50. The hidden size of LSTM is 128 for each direction. The dimension of activation layers for component extraction, simile sentence classification and language modeling are set to 64, 32 and 64 respectively. A dropout layer (Srivastava et al., 2014) is used between the word embedding layer and the bidirectional LSTM layer with the probability of 0.5. Moreover, early stopping (Prechelt, 1998) is adopted to finish the learning process. The AdaDelta (Zeiler, 2012) strategy is used for parameter optimization with a learning rate of 1.0.

5.2 Evaluating Simile Sentence Classification

5.2.1 Comparisons

We compare the following systems.

Feature based approaches. With manually designed features, we build two Random Forest classifiers to determine whether a sentence contains a simile. **Baseline1** follows (Niculae and Danescu-Niculescu-Mizil, 2014). It first extracts candidate simile components and then uses a classifier to determine whether they form a simile. We adopt our best neural component extractor (will be introduced in Section 5.3.1) to extract components. The classifier uses features including: (1) bag-of-words; (2) corresponding occurrence within constituents; (3) word embeddings of extracted components. Niculae and Danescu-Niculescu-Mizil (2014) also used domain specific information and lexicon knowledge, which our data lacks. **Baseline2** is based on (Li et al., 2008), which doesn't

---

[2] `https://github.com/HIT-SCIR/pyltp`

Model	Simile Classification		
	P	R	F_1
Baseline1	0.6523	0.4752	0.5498
Baseline2	0.7661	0.7832	0.7745
Singletask (SC)	0.7751	0.8895	0.8284
Multitask (SC+CE)	0.8056	0.8886	0.8450
Multitask (SC+LM)	0.8021	0.9105	0.8525
Multitask (SC+CE+LM)	**0.8084**	**0.9220**	**0.8615**

Table 3: Experimental results on simile sentence classification. SC: simile sentence classification; CE: component extraction; LM: language modeling.

need to identify components beforehand. The features include: (1) the tokens and POS tags of the words around the comparator within a fixed window (set to 5 in experiments); (2) the tokens, POS tags and dependency relation tags of the words that have dependency relations with the comparator.

Singletask(SC). This system is a simplified version of our proposed model in Section 4 by considering the simile sentence classification task only.

Multitask learning approaches. The full architecture is described in Section 4. To see their contributions, we add simile component extraction, language modeling and their combination incrementally.

5.2.2 Results

Table 3 shows the performance of the systems. The results are reported with the precision (P), recall (R), and their harmonic mean F_1 score (F_1).

The two feature based methods perform differently. Baseline1 performs poorly. The reason may be that the classification depends on the performance of component extraction, while even our best component extractor performs far from perfect, which brings error propagation. In addition, classifying with component related features only ignores much context, which further decreases the performance. Baseline2 considers context windows and outperforms baseline1 largely. This confirms our intuition that context information implies the semantic of simile expression.

Furthermore, we have other observations: (1) neural network based approaches largely outperform feature-based classifiers;(2) multitask learning approaches outperform every single task approach and other baselines. Both the component extraction and the language modeling task contribute for simile sentence classification. Component extraction improves the precision and language modeling improves both the precision and the recall. Combining them together can achieve the best performance. The improvement of F_1 score can reach to 3.3% compared with the best single task model.

5.3 Evaluating Simile Component Extraction

5.3.1 Comparisons

We compare the following systems for simile component extraction.

Rule based approach. We follow (Niculae and Yaneva, 2013) to design syntactic patterns for extraction. We convert the original patterns to fit the outputs of the parser we used.

CRF model. Since we view component extraction as a sequence labeling problem, a CRF model with manually designed feature templates is used as a baseline. Feature template is designed for every word. The features include the tokens and their POS tags within a fixed context window (set to 5 in experiments). We also use dependency parsing based features to capture dependencies between words.

Singletask(CE). We remove the simile classification and language modeling modules from the multitask learning framework introduced in Section 4 to build an end-to-end single task component extractor.

Pipeline approaches. Pipeline approaches first classify a sentence as simile or not, and then extract components from simile sentences. We investigate two combinations: RandomForest→CRF, we use baseline2 for sentence classification and CRF for component extraction; SingleSC→SingleCE, we use neural network based single task sentence classifier and component extractor.

Multitask learning approaches. We exploit simile sentence classification and language modeling modules to enhance component extraction in our multitask learning framework.

5.3.2 Results and Discussion

A component (i.e., the tenor or the vehicle) is judged to be correct only if both the boundary and the tag exactly match the gold answer. For a simile sentence, we should extract both the tenor and the vehicle rather than only partial components. Therefore, we use *pair-wise* level precision (P), recall (R) and F_1 score (F_1) for evaluation. A tenor-vehicle pair is viewed as correct only if both components are correct.

Table 4 shows the results of various systems and settings on two test sets. The first dataset consists

Model	Gold simile sentences			Whole test set		
	P	R	F_1	P	R	F_1
Rule based	0.4094	0.1805	0.2505		—	
CRF	0.5619	0.5907	0.5760	0.3157	0.3698	0.3406
Singletask (CE)	0.7297	0.7854	0.7564	0.5580	0.6489	0.5998
RandomForest → CRF		—		0.4591	0.4980	0.4778
SingleSC → SingleCE		—		0.5720	0.7074	0.6325
Multitask (CE+SC)		—		0.5409	0.6400	0.5861
Multitask (CE+LM)	0.7530	0.7876	0.7699	0.5741	0.7015	0.6306
Multitask (CE+SC+LM)		—		0.5599	0.6989	0.6211
Optimized pipeline		—		0.6160	0.7361	0.6707

Table 4: Experimental results on component extraction. Experiments on dataset of simile sentences assume that the sentence classifier is perfect. CE: component extraction; SC: simile sentence classification; LM: language modeling.

of all manually labeled simile sentences in the test set and the second dataset is the whole test set. We want to compare how component extraction systems work when they know whether a sentence contains a simile or not. We report and discuss the results from the following aspects.

The effect of simile sentence classification. First, we can compare the results in the middle column and the rightmost column in Table 4. It is clear that the component extraction systems work much better when they know whether a sentence contains a simile or not. Second, we can see that both pipelines (the feature-based and the neural network based) achieve a better performance compared with extracting components directly using either the CRF model or the neural single task model. Third, Multitask(CE+SC) doesn't bring significant improvements compared with the single task neural model. These observations indicate that simile sentence classification is suitable to be a pre-processing for simile component classification. It is necessary to further study how to use high level predictions (sentence classification) to learn better representations for consistently improving local predictions (simile component extraction).

Rule based, feature-based and neural models. We can see that even on gold simile sentences, the rule based method doesn't work well. The poor performance of the rule based approach is due to the following reasons. First, the rule-based method is difficult to deal with complex sentence structures. It often fails when there are multiple subordinate clauses. Second, the comparator "像" in Chinese has multiple syntactic roles, sometimes is used as a verb, sometimes is used as a preposition. Third, the accuracy of Chinese dependency parser still has room to be improved.

The CRF method performs significantly better, because it considers more contextual signals. Our neural single task model achieves large improvements on both datasets. This verifies the effectiveness of the end-to-end approach. Neural models can see a long range of context and learn features automatically. The word embeddings learned on external resources implicitly have semantic domain information, which is not only useful for generalization but also important for figurative language processing.

The effect of language modeling. Surprisingly, using language modeling as an auxiliary task is very useful, especially when dealing with noisy sentences. It gains a 1.3% F_1 improvement on the gold simile sentences due to the improvement on the precision and a 3% F_1 improvement on the whole test set due to a large improvement on the recall. Generally, language modeling may help learn better task specific representations, especially when data size is limited (Rei, 2017). Another reason may be that language modeling aims to make local predictions, the same as simile component extraction. Additional information from the same level may be more useful.

Some observations help understand the effect of language modeling. Figure 2 illustrates the relative distance of tenors and vehicles to the comparator. Both tenors and vehicles tend to occur near the comparator and have a clear preference on which side of the comparator. Tenors are more dispersed compared with vehicles, which may increase the difficulty. As shown in Table 5, the performance on identifying tenors is obviously worse than identifying vehicles in both settings.

Figure 3 shows the distribution of extracted components by two settings on the whole test set. We can see that with language modeling, Mul-

Model	Tenor			Vehicle		
	P	R	F_1	P	R	F_1
Singletask(CE)	0.7156	0.6935	0.7044	0.7789	0.8313	0.8043
Multitask (CE+LM)	0.6792	0.7881	0.7296	0.7393	0.9026	0.8128

Table 5: Experimental results on identifying individual types of components.

titask(CE+LM) makes predictions more aggressively compared with Singletask(CE) and tends to recognize more nearby components. We also observe that Multitask(CE+LM) identifies more correct UNK components, which are out of vocabulary or low frequency concepts. This means that language modeling leads the model to consider more local contextual patterns.

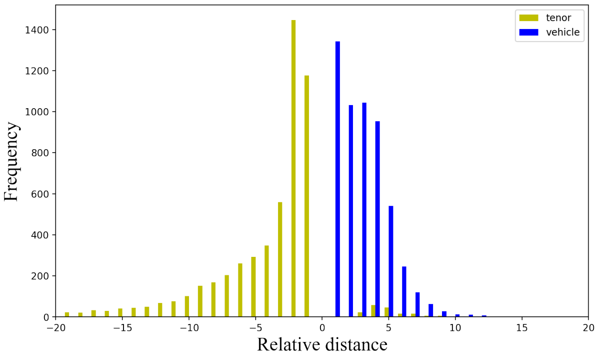

Figure 2: Relative distance of tenors and vehicles to the comparator in the dataset.

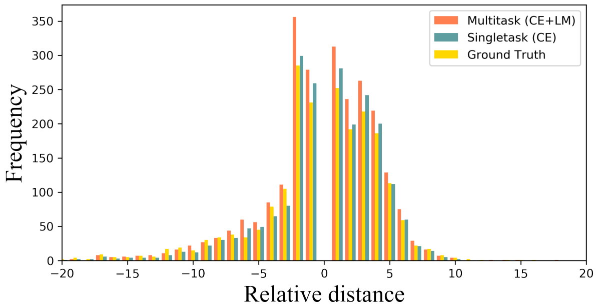

Figure 3: The distribution of extracted simile components on the whole test set.

Optimized Pipeline. According to the results and analysis, we can summarize that (1) simile sentence classification can achieve good performance by jointly optimizing three tasks; (2) simile components can be improved with language modeling as an auxiliary task; (3) simile sentence classification is suitable to be used as a pre-precessing for simile component classification. Therefore, we could build an optimized pipeline. We first use the enhanced simile sentence classifier to filter simile sentences and then use the enhanced component extractor to extract tenors and vehicles. As shown in Table 4, the optimized pipeline performs better than the strongest multitask learning setting.

However, in all settings, the precision scores are lower compared with the recall scores. This indicates that compared with identifying surface patterns, distinguishing metaphorical from literal meanings is much harder and more external knowledge should be incorporated.

6 Conclusion

This paper presented a study on simile recognition by exploiting neural networks. We construct a manually annotated dataset for advancing the research on simile analysis in Chinese. We propose a multitask learning framework, which jointly optimizes three tasks: simile sentence classification, simile component extraction and language modeling. The experimental results demonstrate the effectiveness of proposed approaches. It shows that simile sentence classification and simile component extraction both benefit from multitask learning. Simile sentence classification can achieve a high performance and simile component extraction still has a lot of room to improve.

In future, we plan to extend this work in several aspects: (1) enrich the simile component structure by adding shared properties or events so that the extracted structures would be more useful for metaphor processing; (2) improve representation learning for recognition by incorporating external knowledge; (3) apply simile recognition to study the use of figurative language in writings.

Acknowledgements

The research work is funded by the National Natural Science Foundation of China (No.61876113), High-level Teachers in Beijing Municipal Universities in the Period of 13th Five-year Plan (CIT&TCD20170322),Beijing Educational Committee Science and Technology Development Planned(No. KM201610028015),Humanity & Social Science general project of Ministry of Education(No.14YJC740087).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169. Association for Computational Linguistics.

Jaime G Carbonell. 1980. Metaphor: A key to extensible semantic analysis. In *Proceedings of the 18th annual meeting on Association for Computational Linguistics*, pages 17–21. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Dan Fass. 1991. met*: A method for discriminating metonymy and metaphor by computer. *Computational Linguistics*, 17(1):49–90.

Jenny Rose Finkel and Christopher D. Manning. 2010. Hierarchical joint learning: Improving joint parsing and named entity recognition with non-jointly labeled data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 720–728. Association for Computational Linguistics.

Andrew Goatly. 2011. *The language of metaphors*. Routledge.

Patrick Hanks. 2012. The roles and structure of comparisons, similes, and metaphors in natural language (an analogical system). *Prose (in honor of the Dickens Bicentennial)*, page 5.

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

Hyeju Jang, Yohan Jo, Qinlan Shen, Michael Miller, Seungwhan Moon, and Carolyn Rose. 2016. Metaphor detection with topic transition, emotion and cognition in context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 216–225. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, volume 951, pages 282–289.

George Lakoff and Mark Johnson. 2008. *Metaphors we live by*. University of Chicago press.

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

Bin Li, Li-li Yu, Min Shi, and Wei-guang Qu. 2008. Computation of chinese simile with ”xiang”. *Journal of Chinese Information Processing*, 22(6):27–32.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *IJCAI*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888. Association for Computational Linguistics.

Zachary J Mason. 2004. Cormet: a computational, corpus-based conventional metaphor extraction system. *Computational linguistics*, 30(1):23–44.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116. Association for Computational Linguistics.

Vlad Niculae. 2013. Comparison pattern matching and creative simile recognition. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, pages 110–114.

Vlad Niculae and Cristian Danescu-Niculescu-Mizil. 2014. Brighter than gold: Figurative language in user generated comparisons. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2008–2018. Association for Computational Linguistics.

Vlad Niculae and Victoria Yaneva. 2013. Computational considerations of comparisons and similes. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 89–95. Association for Computational Linguistics.

Lutz Prechelt. 1998. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767.

Ashequl Qadir, Ellen Riloff, and Marilyn Walker. 2015. Learning to recognize affective polarity in similes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 190–200. Association for Computational Linguistics.

Ashequl Qadir, Ellen Riloff, and Marilyn A. Walker. 2016. Automatically inferring implicit properties in similes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1223–1232. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Conll '09 design challenges and misconceptions in named entity recognition. In *CoNLL '09: Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130, Vancouver, Canada. Association for Computational Linguistics.

M. Schuster and K.K. Paliwal. 2002. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1002–1010, Beijing, China. Coling 2010 Organizing Committee.

Ekaterina Shutova and Simone Teufel. 2010. Metaphor corpus annotated for source - target domain mappings. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, pages 3255–3261. European Languages Resources Association (ELRA).

Ekaterina Shutova, Simone Teufel, and Anna Korhonen. 2013. Statistical metaphor processing. *Computational Linguistics*, 39(2):301–353.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Tony Veale. 1995. Metaphor, memory and meaning: Symbolic and connectionist issues in metaphor interpretation.

Tony Veale. 2012a. A computational exploration of creative similes. *Metaphor in Use: Context, culture, and communication*, 38:329–343.

Tony Veale. 2012b. A context-sensitive, multi-faceted model of lexico-conceptual affect. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 75–79. Association for Computational Linguistics.

Tony Veale and Yanfen Hao. 2007. Learning to understand figurative language: from similes to metaphors to irony. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 29, pages 683–688.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566. Association for Computational Linguistics.

Structured Alignment Networks for Matching Sentences

Yang Liu*♠, Matt Gardner◇, and Mirella Lapata♠

♠University of Edinburgh
◇Allen Institute for Artificial Intelligence
yang.liu2@ed.ac.uk mattg@allenai.org mlap@inf.ed.ac.uk

Abstract

Many tasks in natural language processing involve comparing two sentences to compute some notion of relevance, entailment, or similarity. Typically, this comparison is done either at the word level or at the sentence level, with no attempt to leverage the inherent structure of the sentence. When sentence structure is used for comparison, it is obtained during a non-differentiable pre-processing step, leading to propagation of errors. We introduce a model of *structured alignments* between sentences, showing how to compare two sentences by matching their latent structures. Using a structured attention mechanism, our model matches candidate spans in the first sentence to candidate spans in the second sentence, simultaneously discovering the tree structure of each sentence. Our model is fully differentiable and trained only on the matching objective. We evaluate this model on two tasks, entailment detection and answer sentence selection, and find that modeling latent tree structures results in superior performance. Analysis of the learned sentence structures shows they can reflect some syntactic phenomena.

1 Introduction

There are many tasks in natural language processing that require matching two sentences: natural language inference (Bowman et al., 2015; Nangia et al., 2017) and paraphrase detection (Wang et al., 2017b) are classification tasks over sentence pairs, and question answering often requires an alignment between a question and a passage of text that may contain the answer (Tan et al., 2016a; Rajpurkar et al., 2016; Joshi et al., 2017).

Most neural models for these tasks perform comparisons between the two sentences either at the word level (Parikh et al., 2016), or at the sentence level (Bowman et al., 2015). Word-level comparisons ignore the inherent structure of the sentences being compared, at best relying on a recurrent neural network such as an LSTM (Hochreiter and Schmidhuber, 1997) to incorporate some amount of context from neighboring words into each word's representation. Sentence-level comparisons can incorporate the structure of each sentence individually (Bowman et al., 2016; Tai et al., 2015), but cannot easily compare substructures between the sentences, as these are all squashed into a single vector. Some models do incorporate sentence structure by comparing subtrees between the two sentences (Zhao et al., 2016; Chen et al., 2017), but require pipelined approaches where a parser is run in a non-differentiable preprocessing step, losing the benefits of end-to-end training.

In this paper we propose a method, which we call *structured alignment networks*, to perform comparisons between substructures in two sentences, in a more interpretable way, and without relying on an external, non-differentiable parser. We use a structured attention mechanism (Kim et al., 2017; Liu and Lapata, 2018) to compute a *structured alignment* between the two sentences, jointly learning a latent tree structure for each sentence and aligning spans between the two sentences.

Our method constructs a CKY chart for each sentence using the inside-outside algorithm (Manning et al., 1999), which is fully differentiable (Li and Eisner, 2009; Gormley et al., 2015). This chart has a node for each possible span in the sentence, and a score for the likelihood of that span being a constituent in a parse of the sentence, marginalized over all possible parses. We take these two charts and find alignments between them, representing each span in each sentence with structured attention over spans in the other sentence. These

---

[*] Work done during an internship at Allen Institute for Artificial Intelligence.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1554–1564
Brussels, Belgium, October 31 - November 4, 2018.

span representations, weighted by the span's likelihood, are then used to compare the two sentences. In this way, we can perform comparisons between sentences by leveraging their internal structure in an end-to-end, fully differentiable model, trained only on one final objective. Our model helps obtain more precise representations of the sentence pair, with the learned tree structures and the alignment between them, and provides better interpretability, which most neural models lack in sentence matching tasks.

We evaluate this model on two sentence comparison datasets: SNLI (Bowman et al., 2015) and TREC-QA (Voorhees and Tice, 2000). We find that comparing sentences at the span level consistently outperforms comparing at the word level. Additionally, the learned sentence structures represent well-formed trees that reflect some syntactic phenomena.

2 Word-level Comparison Baseline

We first describe a common word-level comparison model, called *decomposable attention* (Parikh et al., 2016). This model was first proposed for the natural language inference task, but similar mechanisms have been used in many other tasks, such as for aligning question and passage words in the bi-directional attention model for question answering (Seo et al., 2017). This model serves as our main point of comparison, as our latent tree matching model simply replaces the word-level comparisons in decomposable attention model with span comparisons.

The decomposable attention model consists of three steps: *attend*, *compare*, and *aggregate*. As input, the model takes two sentences a and b represented by sequences of word embeddings $[\boldsymbol{a}_1, \cdots, \boldsymbol{a}_m]$ and $[\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n]$. In the *attend* step, the model computes attention scores for each pair of words across the two input sentences and normalizes them as a soft alignment from a to b (and vice versa):

$$\boldsymbol{e}_{ij} = F_1(\boldsymbol{a}_i)^T F_1(\boldsymbol{b}_j) \tag{1}$$

$$\boldsymbol{B}_i = \sum_{j=1}^{n} \frac{exp(\boldsymbol{e}_{ij})}{\sum_{k=1}^{n} exp(\boldsymbol{e}_{ik})} \boldsymbol{b}_j \tag{2}$$

$$\boldsymbol{A}_j = \sum_{i=1}^{m} \frac{exp(\boldsymbol{e}_{ij})}{\sum_{k=1}^{m} exp(\boldsymbol{e}_{kj})} \boldsymbol{a}_i \tag{3}$$

where F_1 is a feed-forward neural network, $\boldsymbol{B}_i$ is the weighted summation of the words in b that are softly aligned to word $\boldsymbol{a}_i$ and vice versa for $\boldsymbol{A}_j$.

In the *compare* step, the input vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$ are concatenated with their corresponding attended vector $\boldsymbol{B}_i$ and $\boldsymbol{A}_j$, and fed into a feed-forward neural network, giving a comparison between each word and the words it aligns to in the other sentence:

$$\boldsymbol{v}_{ai} = F_2([\boldsymbol{a}_i, \boldsymbol{B}_i]) \quad \forall i \in [1, \cdots, m] \tag{4}$$

$$\boldsymbol{v}_{bj} = F_2([\boldsymbol{b}_j, \boldsymbol{A}_j]) \quad \forall j \in [1, \cdots, n] \tag{5}$$

The *aggregate* step is a simple summation of $\boldsymbol{v}_{ai}$ and $\boldsymbol{v}_{bj}$ for each word in sentence a and b, and the two resulting fixed-length vectors are concatenated and fed into a linear layer with $\boldsymbol{W}_y$ as the weight matrix, followed by a softmax layer for predicting the distribution $\boldsymbol{y}$:

$$\boldsymbol{v}_a = \sum_{i=1}^{m} \boldsymbol{v}_{ai} \quad \boldsymbol{v}_b = \sum_{j=1}^{n} \boldsymbol{v}_{bj} \tag{6}$$

$$\boldsymbol{y} = softmax(\boldsymbol{W}_y[\boldsymbol{v}_a, \boldsymbol{v}_b])) \tag{7}$$

The decomposable attention model completely ignores the order and context of words in the sequence. There are some efforts to strengthen the decomposable attention model with a recurrent neural network (Liu and Lapata, 2018) or intra-sentence attention (Parikh et al., 2016). However, these models amount to simply changing the input vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, and still only perform a word-level alignment between the two sentences.

3 Structured Alignment Networks

Language is inherently tree structured, and the meaning of sentences comes largely from composing the meanings of subtrees (Chomsky, 2002). It is natural, then, to compare the meaning of two sentences by comparing their substructures (MacCartney and Manning, 2009). For example, when determining the relationship between two sentences in Figure 1, the ideal units of comparison are spans determined by subtrees: "*is in Seattle*" compared to "*based in Washington state*".

The challenge with comparing spans drawn from subtrees is that the tree structure of the sentence is latent and must be inferred, either during pre-processing or in the model itself. In this section we present a model that operates on the *latent* tree structure of each sentence, comparing *all possible* spans in one sentence with *all possible* spans in the second sentence, weighted by *how*

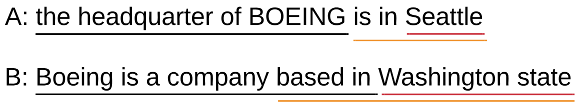

A: the headquarter of BOEING is in Seattle

B: Boeing is a company based in Washington state

Figure 1: Example span alignments of a sentence pair, where different colors indicate matching spans. Note that some spans overlap, which cannot happen in a single tree; our model considers *all possible* span comparisons, weighted by the spans' marginal likelihood.

likely each span is to appear as a constituent in a parse of the sentence. We use the non-terminal nodes of a binary constituency parse to represent spans. Because of this choice of representation, we can use the nodes in a CKY parsing chart to efficiently marginalize span likelihood over all possible parses for each sentence, and compare nodes in each sentence's chart.

3.1 Learning Latent Constituency Trees

A constituency parser can be partially formalized as a graphical model with the following cliques (Klein and Manning, 2004): the latent variables $c_{ikj} \in 0, 1$ for all $i < j$, indicating whether the span from the i-th token to the j-th token ($span_{ij}$) is a constituency node built from the merging of sub-node $span_{ik}$ and $span_{(k+1)j}$. Given a sentence $x = [x_i, \cdots, x_n]$, the probability of a tree z is,

$$p(z|x) = \frac{\prod_{c_{ikj} \in z} p(c_{ikj} = 1)}{\sum_{z' \in Z} \prod_{c_{ikj} \in z'} p(c_{ikj} = 1)} \quad (8)$$

where Z represents all possible constituency trees for x.

The parameters for the graph-based CRF constituency parser are $\boldsymbol{\delta}_{ikj}$ reflecting the scores of $span_{ij}$ forming a binary constituency node with k as the splitting point. It is possible to calculate the marginal probability of each constituency node $p(c_{ijk} = 1|x)$ using the inside-outside algorithm (Klein and Manning, 2003). Although the inside-outside algorithm is constrained to generate a binary tree, this is not a severe limitation, as most structures can be easily binarized (Finkel et al., 2008).

In a typical constituency parser, the score $\boldsymbol{\delta}_{ikj}$ is parameterized according to the production rules of a grammar, e.g., with normalized categorical distributions for each non-terminal. Our unlabeled grammar effectively has only a single production rule, however, we parameterize these scores as bilinear functions operating on the representations of the two subtrees being merged. For the inside pass, as illustrated in Figure 2a, the inside score $\boldsymbol{\alpha}_{ij}$ for span from position i to j is marginalized over the splitting points k:

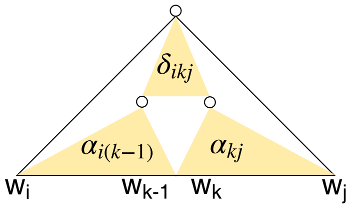

(a) Inside pass

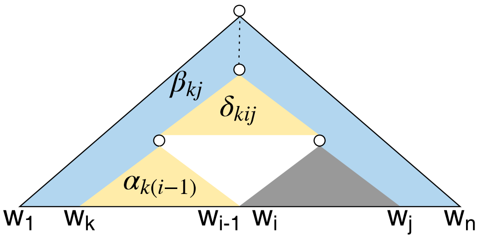

(b) Outside pass

Figure 2: The inside-outside algorithm. (a) is the process for calculating the inside score $\boldsymbol{\alpha}_{ij}$. Three yellow spaces indicate $\boldsymbol{\alpha}_{i(k-1)}$, $\boldsymbol{\alpha}_{kj}$ and $\boldsymbol{\delta}_{ikj}$. (b) is a part of the process for calculating the outside score $\boldsymbol{\beta}_{ij}$, with target span $span_{ij}$ as the right child of a non-terminal. The blue space indicates $\boldsymbol{\beta}_{kj}$ and two yellow spaces indicate $\boldsymbol{\alpha}_{k(i-1)}$ and $\boldsymbol{\delta}_{kij}$.

$$\boldsymbol{\delta}_{ikj} = \boldsymbol{sp}_{ik}^T \boldsymbol{W} \boldsymbol{sp}_{(k+1)j} \quad (9)$$

$$\boldsymbol{\alpha}_{ij} = \sum_{i \,<\, k \,\le\, j} \boldsymbol{\delta}_{ikj} \boldsymbol{\alpha}_{i(k-1)} \boldsymbol{\alpha}_{kj} \quad (10)$$

where $\boldsymbol{sp}_{ij} \in \mathbb{R}^d$ is the representation for the span, and $\boldsymbol{W} \in \mathbb{R}^{d*d}$ is the weight matrix. This process is calculated recursively from bottom to root, generating the score for each possible constituent.

For the outside pass, the outside score $\boldsymbol{\beta}_{ij}$ is:

$$\begin{aligned} \boldsymbol{\beta}_{ij} = & \sum_{1 \le k < i} \boldsymbol{\delta}_{kij} \boldsymbol{\alpha}_{k(i-1)} \boldsymbol{\beta}_{kj} \\ & + \sum_{j < k \le n} \boldsymbol{\delta}_{ijk} \boldsymbol{\alpha}_{(j+1)k} \boldsymbol{\beta}_{ik} \end{aligned} \quad (11)$$

where the first term is the score for $span_{ij}$ being the right child on a non-terminal node and the second term is the score for $span_{ij}$ being the left child. In Figure 2b, we illustrate the outside process with the target span $span_{ij}$ being the right child of a non-terminal node. This process is calculated recursively from root to bottom.

The normalized marginal probability $\boldsymbol{\rho}_{ij}$ for each span $span_{ij}$, where $1 \leq i < n, i < j \leq n$ can be calculated by:

$$\boldsymbol{\rho}_{ij} = \boldsymbol{\alpha}_{ij}\boldsymbol{\beta}_{ij}/\boldsymbol{\alpha}_{0n} \tag{12}$$

To compute the representations of all possible spans, we use Long Short-Term Memory Neural Networks (LSTMs; Hochreiter and Schmidhuber 1997) with max-pooling and minus features (Cross and Huang, 2016; Liu and Lapata, 2017). We represent each sentence as a sequence of word embeddings $[\boldsymbol{w}_{sos}, \boldsymbol{w}_1, \cdots, \boldsymbol{w}_t, \cdots, \boldsymbol{w}_n, \boldsymbol{w}_{eos}]$ and run a bidirectional LSTM to obtain the output vectors. $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t]$ is the output vector for the t^{th} word, and $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$ are the output vectors from the forward and backward directions, respectively. We represent a constituent from position i to j with a span vector $\boldsymbol{sp}_{ij}$:

$$\boldsymbol{sp}_{ij}^{maxpool} = \max(\boldsymbol{h}_i, \cdots, \boldsymbol{h}_j) \tag{13}$$

$$\boldsymbol{sp}_{ij}^{minus} = [\overrightarrow{\boldsymbol{h}}_j - \overrightarrow{\boldsymbol{h}}_{i-1}, \overleftarrow{\boldsymbol{h}}_i - \overleftarrow{\boldsymbol{h}}_{j+1}] \tag{14}$$

$$\boldsymbol{sp}_{ij} = [\boldsymbol{sp}_{ij}^{maxpool}, \boldsymbol{sp}_{ij}^{minus}] \tag{15}$$

where $\max(\boldsymbol{x}_i, \cdots, \boldsymbol{x}_j)$ is the max-pooling operation over the sequence of output vectors within this constituent.

After applying the parsing process on two sentences, we obtain the marginal probabilities for all potential spans of the two constituency trees, which can then be used for aligning.

3.2 Learning Structured Alignments

After learning latent constituency trees for each sentence, we are able to perform span-level comparisons between the two sentences, instead of the word-level comparisons done by the decomposable attention model. The structure of these two comparison models is the same, but the basic elements of our structured alignment model are spans instead of words, and the marginal probabilities obtained from the inside-outside algorithm are used as a *re-normalization* value for incorporating structural information into the alignments.

For sentence a, we have the representation $\boldsymbol{sp}_{ij}^a$ for each $span_{ij}$ and its marginal probability $\boldsymbol{\rho}_{ij}^a$. And for sentence b, we also get $\boldsymbol{sp}_{ij}^b$ and $\boldsymbol{\rho}_{ij}^b$. The attention scores are computed between all pairs of spans across the two sentences, and the attended vectors can be calculated as:

$$\boldsymbol{e}_{ij,kl} = F_1(\boldsymbol{sp}_{ij}^a)^T F_1(\boldsymbol{sp}_{kl}^b) \tag{16}$$

$$\boldsymbol{B}_{ij} = \sum_{k=1}^{n}\sum_{l=k}^{n} \frac{exp(\boldsymbol{e}_{ij,kl} + ln(\boldsymbol{\rho}_{kl}^b))}{\sum_{s=1}^{n}\sum_{t=s}^{n} exp(\boldsymbol{e}_{ij,st} + ln(\boldsymbol{\rho}_{st}^b))} \boldsymbol{sp}_{kl}^b \tag{17}$$

$$\boldsymbol{A}_{kl} = \sum_{i=1}^{m}\sum_{j=i}^{m} \frac{exp(\boldsymbol{e}_{ij,kl} + ln(\boldsymbol{\rho}_{ij}^a))}{\sum_{s=1}^{m}\sum_{t=s}^{m} exp(\boldsymbol{e}_{st,kl} + ln(\boldsymbol{\rho}_{st}^a))} \boldsymbol{sp}_{ij}^a \tag{18}$$

Then, the span vectors are concatenated with the attended vectors and fed into a feed-forward neural network:

$$\boldsymbol{v}_{ij}^a = F_2([\boldsymbol{sp}_{ij}^a, \boldsymbol{B}_{ij}]) \tag{19}$$

$$\boldsymbol{v}_{kl}^b = F_2([\boldsymbol{sp}_{kl}^b, \boldsymbol{A}_{kl}]) \tag{20}$$

To aggregate these vectors, instead of using direct summation, we apply weighted summation with the marginal probabilities as weights:

$$\boldsymbol{v}_a = \sum_{i=1}^{m}\sum_{j=i}^{m} \boldsymbol{\rho}_{ij}^a \boldsymbol{v}_{ij}^a; \boldsymbol{v}_b = \sum_{k=1}^{n}\sum_{l=1}^{n} \boldsymbol{\rho}_{kl}^b \boldsymbol{v}_{kl}^b \tag{21}$$

where $\boldsymbol{\rho}^a$ and $\boldsymbol{\rho}^b$ work like the self-attention mechanism of (Lin et al., 2017) to replace the summation pooling step. We use a softmax function to compute the predicted distribution $\boldsymbol{y}$ of the input sentence pair:

$$\boldsymbol{y} = softmax(\boldsymbol{W}_y[\boldsymbol{v}_a, \boldsymbol{v}_b]) \tag{22}$$

4 Experiments

We evaluate our structured alignment model on two natural language matching tasks: question answering as sentence selection and natural language inference. We view our approach as a module for replacing the widely-used word-level alignment which can be plugged into other neural models. For that reason, our experiments are not intended to show performance improvements over state-of-the-art neural network architectures. Rather our evaluation studies aim to address three questions: (a) whether our methods can be trained effectively in an end-to-end fashion; (b) whether they yield improvements over standard word-level alignment models; and (c) whether they can learn plausible latent constituency tree structures.

Models	MAP	MRR
Word-level Attention	0.764	0.842
Simple Span Alignment	0.772	0.851
Simple Span Alignment + External Parser	0.780	0.846
Structured Alignment (Shared Parameters)	0.780	0.860
Structured Alignment (Separated Parameters)	**0.786**	**0.860**
QA-LSTM (Tan et al., 2016b)	0.730	0.824
Attentive Pooling Network (Santos et al., 2016)	0.753	0.851
Pairwise Word Interaction (He and Lin, 2016)	0.777	0.836
Lexical Decomposition and Composition (Wang et al., 2016)	0.771	0.845
Noise-Contrastive Estimation (Rao et al., 2016)	0.801	**0.877**
BiMPM (Wang et al., 2017b)	**0.802**	0.875

Table 1: Results of our models (top) and previously proposed systems (bottom) on the TREC-QA test set.

For both tasks, we initialize our model with 300D 840B GloVe word embeddings (Pennington et al., 2014). The hidden size for the BiLSTM is 150. The feed-forward networks F_1 and F_2 are two-layer perceptrons with ReLU as the hidden activation function and the size of the hidden and output layers is set to 300. All hyperparameters are selected based on the model's performance on the development set.

4.1 Answer Sentence Selection

We first study the effectiveness of our model for answer sentence selection tasks. Given a question, answer sentence selection aims to rank a list of candidate answer sentences based on their relatedness to the question. We experiment on the TREC-QA dataset (Wang et al., 2007), in which all questions with only positive or negative answers are removed. This leaves us with 1,162 training questions, 65 development questions and 68 test questions. Experimental results are listed in Table 1. We measure performance by the mean average precision (MAP) and mean reciprocal rank (MRR) using the standard TREC evaluation script.

In the first block of Table 1, we compare our model and variants thereof against several baselines. The first baseline is the *Word-level Decomposable Attention* model strengthened with a bidirectional LSTM for obtaining a contextualized representation for each word. The second baseline is a *Simple Span Alignment* model; we use an MLP layer over the LSTM outputs to calculate the unnormalized scores and replace the inside-outside algorithm with a simple softmax function to obtain the probability distribution over all candidate spans. We also introduce a pipelined baseline where we extract constituents from trees parsed by the CoreNLP (Manning et al., 2014) constituency parser, and use the *Simple Span Alignment* model to only align these constituents.

As shown in Table 1, we use two variants of the *Structured Alignment* model, since the structure of the question and the answer sentence may be different; the first model shares parameters across the question and the answer for computing the structures, while the second one uses separate parameters. We view the sentence selection task as a binary classification problem and the final ranking is based on the predicted probability of the sentence containing the correct answer (positive label). We apply dropout to the output of the BiLSTM with dropout ratio set to 0.2. All parameters (including word embeddings) are updated with AdaGrad (Duchi et al., 2011), and the learning rate is set to 0.05.

Table 1 (second block) also reports the performance of various comparison systems and state-of-the-art models. As can be seen, on both MAP and MRR metrics, structured alignment models perform better than the decomposable attention model, showing that structural bias is helpful for matching a question to the correct answer sentence. We also observe that using separate parameters achieves higher scores on both metrics. The simple span alignment model obtains results similar to the decomposable attention model, suggesting that the shallow softmax distribution is ineffective for capturing structural information and may even introduce redundant noise. The pipelined model with an external parser also slightly im-

Models	Accuracy	# Parameters
Word-level Attention	85.8	1.1M
Simple Span Alignment	85.4	1.26M
Simple Span Alignment + External Parser	85.6	1.17M
Structured Alignment	**86.3**	1.44M
Classifier with handcrafted features (Bowman et al., 2015)	78.2	-
LSTM encoders (Bowman et al., 2015)	80.6	3.0M
LSTM with inter-attention (Rocktäschel et al., 2016)	83.5	252K
Matching LSTMs (Wang and Jiang, 2015)	86.1	1.9M
LSTMN with deep attention fusion (Cheng et al., 2016)	86.3	3.4M
Enhanced BiLSTM Inference Model (Chen et al., 2016)	**88.0**	4.3M
Densely Interactive Inference Network (Gong et al., 2017)	**88.0**	-

Table 2: Test accuracy (%) on the SNLI dataset. Wherever available we also provide the number of parameters (excluding embeddings).

proves upon the baseline, but still cannot outperform the end-to-end trained structured alignment model which achieves results comparable with several strong baselines with fewer parameters. As mentioned earlier, our model could be used as a plug-in component for other more complex models, and may boost their performance by modeling the latent structures. At the same time, the structured alignment can provide better interpretability for sentence matching tasks, which is a defect of most neural models.

4.2 Natural Language Inference

The second task we consider is natural language inference, where the input is a pair of premise and hypothesis sentences, and the goal is to predict whether the premise entails the hypothesis, contradicts the hypothesis, or neither. For this task, we use the Stanford NLI dataset (Bowman et al., 2015). After removing sentences with unknown labels, we obtain 549,367 pairs for training, 9,842 for development and 9,824 for testing.

We compare our model against the same baselines used in the question answering task. All parameters (including word embeddings) are updated with AdaGrad (Duchi et al., 2011), and the learning rate is set to 0.05. Dropout is used with ratio 0.2. The structured alignment model in this experiment uses shared parameters for computing latent tree structures, since both the premise and hypothesis are declarative sentences.

The results of our experiments are shown in Table 2. Similar to the answer selection task, the tree matching model outperforms the decomposable model. Our structured alignment model gains 0.5% in accuracy over the baseline word-level comparison model without any additional annotation, simply from introducing a structural bias in the alignment between the sentences. Simple span alignment, however, is not helpfult and even slightly degrades the performance over the word-level model.

4.3 Analysis of Learned Tree Structures

In this section, we give a brief qualitative analysis of the learned tree structures. We present the CKY charts for two randomly-selected sentence pairs in the SNLI test set in Figure 3. Recall that the CKY chart shows the likelihood of each span appearing as a constituent in the parse of the sentence, marginalized over all possible parses. By visualizing these span probabilities, we can see that the model learns structures which correspond to known syntactic structures.

In subfigure (a), we can see that *band is playing* is a very-likely span, as is *at a large venue*. In subfigure (b), the phrases *performing at a local bar* and *at a local bar or club* also receive high probabilities. For the second sentence pair, we see that the model can even resolve some attachment ambiguities correctly. The prepositional phrase *with green feathers*, has a very low score for being attached to *women*. Instead, the model prefers to attach it to *lingerie*, forming the span *lingerie with green feathers*. We also present the top-5 spans and their alignments in subfigures (c) and (d), which can be used to interpret model decisions for sentence matching tasks.

The analysis above and our experimental results in the previous section suggest that our

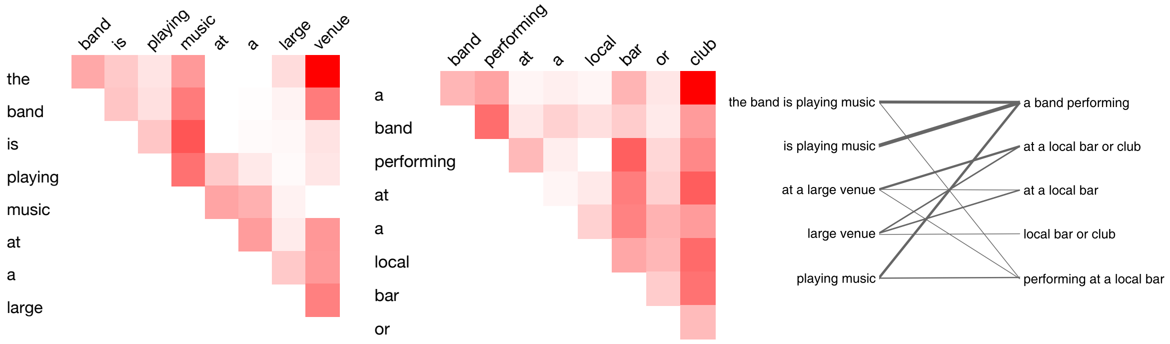

(a) Premise Sentence

(b) Hypothesis sentence

(c) Alignment of top-5 spans

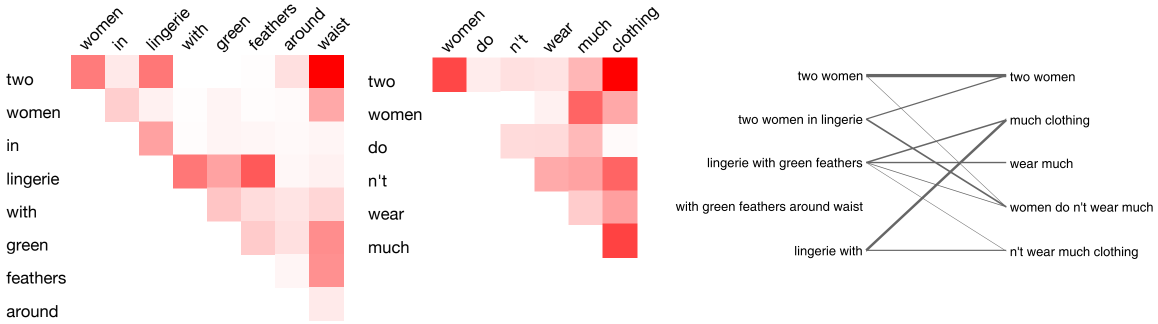

(d) Premise Sentence

(e) Hypothesis sentence

(f) Alignment of top-5 spans

Figure 3: (a), (b), (d), and (e) are CKY charts showing marginalized span probabilities for sentence pairs in the SNLI test set. Each cell uses depth of the color to represent the probability of the span (from the i-th word to the j-th word) forming a proper constituent. (c) and (f) are the alignments of the top-5 spans from hypothesis sentence to the premise sentence, where the boldness of the lines indicates the probability of spans being aligned.

model is able to learn tree structures which are closely related to syntax, and in addition reflect the semantic-level characteristics of the task at hand. In both question answering and natural language inference tasks, we observe that structured alignment leads to performance improvements over word-level models. This is in contrast to prior work (Williams et al., 2017), where the discovery of tree structures based on a semantic objective is not helpful. Although we use the same supervision signal in our model, a difference between the two approaches is that they are trying to learn tree structures for each sentence *independently*, performing comparisons at the sentence level only. Comparing spans directly forces the model to induce trees with comparable constituents, giving the model a stronger inductive bias.

Although our main goal is not to induce a grammar, we perform some simple experiments to compare the learned latent trees with parser-generated ones. We parse the sentences in both test-sets with the CoreNLP (Manning et al., 2014) constituency parser to obtain silver trees. Based on the parsing part of the trained structured alignment model, we compute the marginal probabilities of test sentences and feed them into CKY algorithm (Younger, 1967) to find the most likely constituency trees. We then convert both silver and latent trees to sets of constituent brackets, and calculate the accuracy of the learned brackets against the silver parses. We use different combinations of training- and test-sets to examine the transferability of the learned tree structures. The results are shown in Table 3. We can see that although our model does not have any tree-structured input during training, it can still outperform the left-branching (LB) and right-branching baselines (RB) and achieve some consistency with the parser generated trees.

5 Related Work

Sentence comparison models The Stanford natural language inference dataset (Bowman et al., 2015), and the expanded multi-genre natural language inference dataset (Nangia et al., 2017), are

trained on tested on	SNLI	TREC	LB	RB
SNLI	**15.1**	10.7	12.8	6.0
TREC	12.3	**11.4**	10.5	3.2

Table 3: Brackets accuracy of latent learned trees against silver trees from CoreNLP parser. We show the transferability of the learned parser by applying it on a test-set different from the training-set. For example, we train the structured alignment model on the TREC-QA data, and apply it on SNLI to obtain the tree distributions. LB and RB are left- and right-branching baselines.

the most well-known recent sentence comparison tasks. The literature on this comparison task is far too extensive to include here, although the recent shared task on Multi-NLI gives a good survey of sentence-level comparison models (Nangia et al., 2017). Some of these models use sentence structures, which are obtained either in a latent fashion (Bowman et al., 2016) or during preprocessing (Zhao et al., 2016), but they squash all of the structure into a single vector, losing the ability to easily compare substructures between the two sentences.

For models doing a word-level comparison, the decomposable attention model (Parikh et al., 2016), which we have discussed already in this paper, is the most salient example, although many similar models exist in the literature (Chen et al., 2017; Wang et al., 2017b). The idea of word-level alignments between a question and a passage is also pervasive in the recent question answering literature (Seo et al., 2017; Wang et al., 2017a).

Finally, and most similar to our approach, several models have been proposed that directly compare subtrees between two sentences (Chen et al., 2017; Zhao et al., 2016). However, all of these models are pipelined; they obtain the sentence structure in a non-differentiable preprocessing step, losing the benefits of end-to-end training. Ours is the first model to allow comparison between *latent* tree structures, trained end-to-end on the comparison objective.

Structured attention While it has long been known that inference in graphical models is differentiable (Li and Eisner, 2009; Domke, 2011), and using inference in, e.g., a CRF (Lafferty et al., 2001) as the last layer in a neural network is common practice (Liu and Lapata, 2017; Lample et al., 2016), the use of inference algorithms as intermediate layers in end-to-end neural networks is a recent development. Kim et al. (2017) were the first to use inference to compute structured attentions over latent sentence variables, inducing tree structures trained on the end-to-end objective. Liu and Lapata (2018) showed how to do this more efficiently, although their work is still limited to structured attention over a single sentence. Our model is the first to include latent structured alignments between two sentences.

Grammar Induction Unsupervised grammar induction is a well-studied problem (Cohen and Smith, 2009). The most recent work in this direction was the Neural E-DMV model of Jiang et al. (2016). While our goal is not to induce a grammar, we do produce a probabilistic grammar as a byproduct of our model. Our results suggest that training on more complex objectives may be a good way to pursue grammar induction in the future; forcing the model to construct consistent, comparable subtrees between the two sentences is a strong signal for grammar induction. Very recently, a few models attempt to infer latent dependency tree structures with neural models in sentence modeling tasks (Yogatama et al., 2017; Choi et al., 2018).

6 Conclusions

In this paper we have considered the problem of comparing two sentences in natural language processing models. We have shown how to move beyond word- and sentence-level comparison to comparing spans between two sentences, without the need for an external parser. Through experiments on sentence comparison datasets, we have seen that span comparisons consistently outperform word-level comparisons, with no additional supervision. The proposed model can be trained effectively, in an end-to-end fashion and is able to induce plausible tree structures.

Our results have several implications for future work. First, the success of span comparisons over word-level comparisons suggests that it may be advantageous to include such comparisons in more complex models, either for comparing two sentences directly, or as intermediate parts of models for more complex tasks, such as reading comprehension. Second, our model's ability to infer trees from a semantic objective is intriguing, and suggestive of future opportunities in grammar induction research. The use of the inside-outside algorithm unavoidably renders the full model er (by 5–8 times) compared to the decomposable attention model. We hope to find a more efficient way to accelerate this dynamic programming method on a GPU.

Acknowledgments This research is supported by a Google PhD Fellowship to the first author. We acknowledge the financial support of the European Research Council (Lapata; award number 681760).

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477, Berlin, Germany.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Enhancing and combining sequential and tree LSTM for natural language inference. *arXiv preprint arXiv:1609.06038*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas.

Jihun Choi, Kang Min Yoo, and Sang goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5094–5101, New Orleans, Louisiana.

Noam Chomsky. 2002. *Syntactic structures*. Walter de Gruyter.

Shay Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82, Boulder, Colorado.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas.

Justin Domke. 2011. Parameter learning with truncated message-passing. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2937–2943, Colorado Springs, Colorado.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL-08: HLT*, pages 959–967, Columbus, Ohio.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

Matthew Gormley, Mark Dredze, and Jason Eisner. 2015. Approximation-aware dependency parsing by belief propagation. *Transactions of the Association for Computational Linguistics*, 3:489–501.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, San Diego, California.

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

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. In *Proceedings of the 5th International Cofnerence on Learning Representations*, Toulon, France.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics, Main Volume*, pages 478–485, Barcelona, Spain.

Dan Klein and Christopher D Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15*, pages 3–10. MIT Press.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, Williamstown, Massachusetts.

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

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 40–51, Singapore.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Yang Liu and Mirella Lapata. 2017. Learning contextually informed representations for linear-time discourse parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1289–1298, Copenhagen, Denmark.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the international conference on computational semantics*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland.

Christopher D Manning, Hinrich Schütze, et al. 1999. *Foundations of statistical natural language processing*, volume 999. MIT Press.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R Bowman. 2017. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. *arXiv preprint arXiv:1707.08172*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management Conference*, pages 1913–1916, Indianapolis, Indiana.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the 4th International Cofnerence on Learning Representations*, San Juan, Puerto Rico.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceeding of the 5th International Cofnerence on Learning Representations*, Toulon, France.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016a. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473, Berlin, Germany.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016b. LSTM-based deep learning models for non-factoid answer selection. In *Proceedings of the ICLR 2016 Workshop Track*, San Juan, Puerto Rico.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207, Athens, Greece.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with LSTM. *arXiv preprint arXiv:1512.08849*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017a. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017b. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144–4150, Melbourne, Australia.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1340–1349, Osaka, Japan.

Adina Williams, Andrew Drozdov, and Samuel R Bowman. 2017. Learning to parse from a semantic objective: It works. is it syntax? *arXiv preprint arXiv:1709.01121*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *Proceedings of the 5th International Cofnerence on Learning Representations*, Toulon, France.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208.

Kai Zhao, Liang Huang, and Mingbo Ma. 2016. Textual entailment with structured attentions and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2248–2258, Osaka, Japan.

Compare, Compress and Propagate: Enhancing Neural Architectures with Alignment Factorization for Natural Language Inference

Yi Tay[†], **Luu Anh Tuan**[*], **Siu Cheung Hui**[ϕ]
[†ϕ]Nanyang Technological University, Singapore
[*]Institute for Infocomm Research, A*Star Singapore
ytay017@e.ntu.edu.sg[†], at.luu@i2r.a-star.edu.sg[*]
asschui@ntu.edu.sg[ϕ]

Abstract

This paper presents a new deep learning architecture for Natural Language Inference (NLI). Firstly, we introduce a new architecture where alignment pairs are compared, compressed and then propagated to upper layers for enhanced representation learning. Secondly, we adopt factorization layers for efficient and expressive compression of alignment vectors into scalar features, which are then used to augment the base word representations. The design of our approach is aimed to be conceptually simple, compact and yet powerful. We conduct experiments on three popular benchmarks, SNLI, MultiNLI and SciTail, achieving competitive performance on all. A lightweight parameterization of our model also enjoys a ≈ 3 times reduction in parameter size compared to the existing state-of-the-art models, e.g., ESIM and DIIN, while maintaining competitive performance. Additionally, visual analysis shows that our propagated features are highly interpretable.

1 Introduction

Natural Language Inference (NLI) is a pivotal and fundamental task in language understanding and artificial intelligence. More concretely, given a premise and hypothesis, NLI aims to detect whether the latter *entails* or *contradicts* the former. As such, NLI is also commonly known as Recognizing Textual Entailment (RTE). NLI is known to be a significantly challenging task for machines whose success often depends on a wide repertoire of reasoning techniques.

In recent years, we observe a steep improvement in NLI systems, largely contributed by the release of the largest publicly available corpus for NLI - the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) which comprises $570K$ hand labeled sentence pairs. This has improved the feasibility of training complex neural models, given the fact that neural models often require a relatively large amount of training data.

Highly competitive neural models for NLI are mostly based on soft-attention alignments, popularized by (Parikh et al., 2016; Rocktäschel et al., 2015). The key idea is to learn an alignment of sub-phrases in both sentences and learn to compare the relationship between them. Standard feed-forward neural networks are commonly used to model similarity between aligned (decomposed) sub-phrases and then aggregated into the final prediction layers.

Alignment between sentences has become a staple technique in NLI research and many recent state-of-the-art models such as the Enhanced Sequential Inference Model (ESIM) (Chen et al., 2017b) also incorporate the alignment strategy. The difference here is that ESIM considers a non-parameterized comparison scheme, i.e., *concatenating* the subtraction and element-wise product of aligned sub-phrases, along with two original sub-phrases, into the final comparison vector. A bidirectional LSTM is then used to aggregate the compared alignment vectors.

This paper presents a new neural model for NLI. There are several new novel components in our work. Firstly, we propose a *compare, compress and propagate* (ComProp) architecture where compressed alignment features are propagated to upper layers (such as a RNN-based encoder) for enhancing representation learning. Secondly, in order to achieve an efficient propagation of alignment features, we propose alignment factorization layers to reduce each alignment vector to a single scalar valued feature. Each scalar valued feature is used to augment the base word representation, allowing the subsequent RNN encoder layers to benefit from not only global but also cross sentence information.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1565–1575
Brussels, Belgium, October 31 - November 4, 2018.

There are several major advantages to our proposed architecture. Firstly, our model is relatively compact, i.e., we compress alignment feature vectors and augment them to word representations instead. This is to avoid large alignment (or match) vectors being propagated across the network. As a result, our model is more parameter efficient compared to ESIM since the width of the middle layers of the network is now much smaller. To the best of our knowledge, this is the first work that explicitly employs such a paradigm.

Secondly, the explicit usage of compression enables improved interpretabilty since each alignment pair is compressed to a scalar and hence, can be easily visualised. Previous models such as ESIM use subtractive operations on alignment vectors, edging on the intuition that these vectors represent contradiction. Our model is capable of visually demonstrating this phenomena. As such, our design choice enables a new way of deriving insight from neural NLI models.

Thirdly, the alignment factorization layer is expressive and powerful, combining ideas from standard machine learning literature (Rendle, 2010) with modern neural NLI models. The factorization layer tries to decompose the alignment vector (constructed from the variations of $a - b$, $a \odot b$ and $[a; b]$), learning higher-order feature interactions between each compared alignment. In other words, it models the second-order (pairwise) interactions between *each* feature in *every* alignment vector using factorized parameters, allowing more expressive comparison to be made over traditional fully-connected layers (FC). Moreover, factorization-based models are also known to be able to model low-rank structure and reduce risks of overfitting. The effectiveness of the factorization alignment over alternative baselines such as feed-forward neural networks is confirmed by early experiments.

The major contributions of this work are summarized as follows:

- We introduce a *Compare, Compress and Propagate* (ComProp) architecture for NLI. The key idea is to use the myriad of generated comparison vectors for augmentation of the base word representation instead of simply aggregating them for prediction. Subsequently, a standard compositional encoder can then be used to learn representations from the augmented word representations. We show that we are able to derive meaningful insight from visualizing these augmented features.
- For the first time, we adopt expressive factorization layers to model the relationships between soft-aligned sub-phrases of sentence pairs. Empirical experiments confirm the effectiveness of this new layer over standard fully connected layers.
- Overall, we propose a new neural model - CAFE (**C**omProp **A**lignment-**F**actorized **E**ncoders) for NLI. Our model achieves state-of-the-art performance on SNLI, MultiNLI and the new SciTail dataset, outperforming existing state-of-the-art models such as ESIM. Ablation studies confirm the effectiveness of each proposed component in our model.

2 Related Work

Natural language inference (or textual entailment recognition) is a long standing problem in NLP research, typically carried out on smaller datasets using traditional methods (Maccartney, 2009; Dagan et al., 2006; MacCartney and Manning, 2008; Iftene and Balahur-Dobrescu, 2007).

The relatively recent creation of $570K$ human annotated sentence pairs (Bowman et al., 2015) have spurred on many recent works that use neural networks for NLI. Many advanced neural architectures have been proposed for the NLI task, with most exploiting some variants of neural attention which learn to pay attention to important segments in a sentence (Parikh et al., 2016; Chen et al., 2017b; Wang and Jiang, 2016b; Rocktäschel et al., 2015; Yu and Munkhdalai, 2017a).

Amongst the myriad of neural architectures proposed for NLI, the ESIM (Chen et al., 2017b) model is one of the best performing models. The ESIM, primarily motivated by soft subphrase alignment in (Parikh et al., 2016), learns alignments between BiLSTM encoded representations and aggregates them with another BiLSTM layer. The authors also propose the usage of subtractive composition, claiming that this helps model contradictions amongst alignments.

Compare-Aggregate models are also highly popular in NLI tasks. While this term was coined by (Wang and Jiang, 2016a), many prior NLI models follow this design (Wang et al., 2017; Parikh et al., 2016; Gong et al., 2017; Chen et al., 2017b). The key idea is to aggregate matching features and pass them through a dense layer for prediction. (Wang et al., 2017) proposed BiMPM, which adopts multi-perspective cosine matching across sequence pairs. (Wang and Jiang, 2016a) proposed a one-way attention and convolutional aggregation layer. (Gong et al., 2017) learns representations with highway layers and adopts ResNet for learning features over an interaction matrix.

There are several other notable models for NLI. For instance, models that leverage directional self-attention (Shen et al., 2017) or Gumbel-Softmax (Choi et al., 2017). DGEM is a graph based attention model which was proposed together with a new entailment challenge dataset, SciTail (Khot et al., 2018). Pretraining have been known to also be highly useful in the NLI task. For instance, contextualized vectors learned from machine translation (McCann et al., 2017) (CoVe) or language modeling (Peters et al., 2018) (ELMo) have showned to be able to improve performance when integrated with existing NLI models.

Our work compares and compresses alignment pairs using factorization layers which leverages the rich history of standard machine learning literature. Our factorization layers incorporate highly expressive factorization machines (FMs) (Rendle, 2010) into neural NLI models. In standard machine learning tasks, FMs remain a very competitive choice for learning feature interactions (Xiao et al., 2017) for both standard classification and regression problems. Intuitively, FMs are adept at handling data sparsity (typically interactions) by using factorized parameters to approximate a feature matching matrix. This makes it suitable in our model architecture since feature interaction between subphrase alignment pairs is typically very sparse as well.

A recent work (Beutel et al., 2018) reports an interesting empirical study pertaining to the ability of standard FC layers and their ability to model 'cross features' (or multiplicative features). Their overall finding suggests that while standard ReLU FC layers are able to approximate 2-way or 3-way features, they are extremely inefficient in doing so (requiring either very wide or deep layers). This further motivates the usage of FMs in this work and is well aligned with our empirical results, i.e., strong competitive performance with reasonably small parameterization.

3 Our Proposed Model

In this section, we provide a layer-by-layer description of our model architecture. Our model accepts two sentences as an input, i.e., P (premise) and H (hypothesis). Figure 1 illustrates a high-level overview of our proposed model architecture.

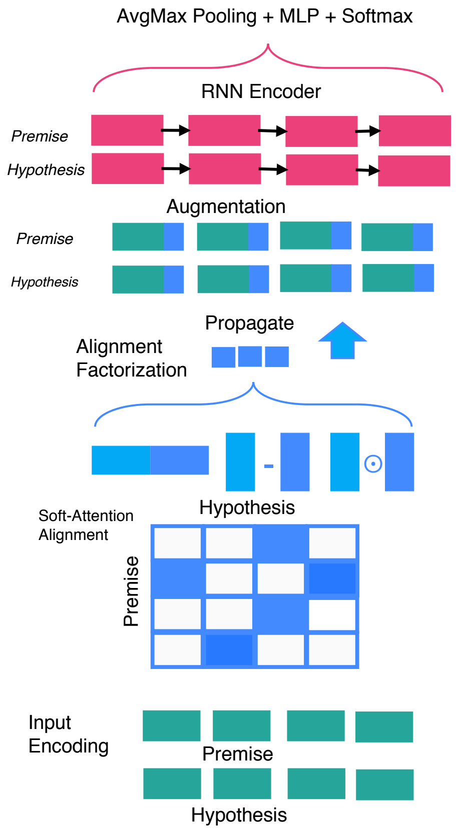

Figure 1: High level overview of our proposed architecture (*best viewed in color*). Alignment vectors are compressed and then propagated to upper representation learning layers (RNN encoders). Intra-attention is omitted in this diagram due to the lack of space.

3.1 Input Encoding Layer

This layer aims to learn a k-dimensional representation for each word. Following (Gong et al., 2017), we learn feature-rich word representations by concatenating word embeddings, character embeddings and syntactic (part-of-speech tag) embeddings (provided in the datasets). Character representations are learned using a convolutional encoder with max pooling function and is commonly used in many relevant literature (Wang et al., 2017; Chen et al., 2017c).

Highway Encoder Subsequently, we pass each concatenated word vector into a two layer highway network (Srivastava et al., 2015) in order to learn a k-dimensional representation. Highway networks are gated projection layers which learn adaptively control how much information is being carried to the next layer. Our strategy is similar to (Parikh et al., 2016) which trains the projection layer in place of tuning the embedding matrix. The usage of highway layers over standard projection layers is empirically motivated. However, an intuition would be that the gates in this layer adapt to learn the relative importance of each word to the NLI task. Let $H(.)$ and $T(.)$ be single layered affine transforms with ReLU and sigmoid activation functions respectively. A single highway network layer is defined as:

$$y = H(x, W_H) \cdot T(x, W_T) + C \cdot x \quad (1)$$

where $C = (1 - T(x, W_T))$ and $W_H, W_T \in \mathbb{R}^{r \times d}$ Notably, the dimensions of the affine transform might be different from the size of the input vector. In this case, an additional nonlinear transform is used to project x to the same dimensionality. The output of this layer is $\bar{P} \in \mathbb{R}^{k \times \ell_P}$ (premise) and $\bar{H} \in \mathbb{R}^{k \times \ell_H}$ (hypothesis), with each word converted to a r-dimensional vector.

3.2 Soft-Attention Alignment Layer

This layer describes two soft-attention alignment techniques that are used in our model.

Inter-Attention Alignment Layer This layer learns an alignment of sub-phrases between $\bar{P}$ and $\bar{H}$. Let $F(.)$ be a standard projection layer with ReLU activation function. The alignment matrix of two sequences is defined as follows:

$$e_{ij} = F(\bar{p}_i)^\top \cdot F(\bar{h}_j) \quad (2)$$

where $E \in \mathbb{R}^{\ell_p \times \ell_h}$ and $\bar{p}_i, \bar{h}_j$ are the i-th and j-th word in the premise and hypothesis respectively.

$$\beta_i = \sum_{j=1}^{\ell_p} \frac{exp(e_{ij})}{\sum_{k=1}^{\ell_p} exp(e_{ik})} \bar{p}_j \quad (3)$$

$$\alpha_j = \sum_{i=1}^{\ell_h} \frac{exp(e_{ij})}{\sum_{k=1}^{\ell_h} exp(e_{kj})} \bar{h}_i \quad (4)$$

where β_i is the sub-phrase in $\bar{P}$ that is softly aligned to h_i. Intuitively, β_i is a weighted sum across $\{p_j\}_{j=1}^{\ell_p}$, selecting the most relevant parts of $\bar{P}$ to represent h_i.

Intra-Attention Alignment Layer This layer learns a *self-alignment* of sentences and is applied to both $\bar{P}$ and $\bar{H}$ independently. For the sake of brevity, let $\bar{S}$ represent either $\bar{P}$ or $\bar{H}$, the intra-attention alignment is computed as:

$$s'_i = \sum_{j=1}^{\ell_p} \frac{exp(f_{ij})}{\sum_{k=1}^{\ell_p} exp(f_{ik})} \bar{s}_j \quad (5)$$

where $f_{ij} = G(\bar{s}_i)^\top \cdot G(\bar{s}_j)$ and $G(.)$ is a nonlinear projection layer with ReLU activation function. The intra-attention layer models similarity of each word with respect to the entire sentence, capturing long distance dependencies and 'global' context of the entire sentence.

3.3 Alignment Factorization Layer

This layer aims to learn a scalar valued feature for each comparison between aligned sub-phrases. Firstly, we introduce our factorization operation, which lives at the core of our neural model.

Factorization Operation Given an input vector x, the factorization operation (Rendle, 2010) is defined as:

$$Z(x) = w_0 + \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle\, x_i\, x_j \quad (6)$$

where $Z(x)$ is a scalar valued output, $\langle .; . \rangle$ is the dot product between two vectors and w_0 is the global bias. Factorization machines model low-rank structure within the matching vector producing a scalar feature. The parameters of this layer are $w_0 \in \mathbb{R}, w \in \mathbb{R}^r$ and $v \in \mathbb{R}^{r \times k}$. The first term $\sum_{i=1}^{n} w_i\, x_i$ is simply a linear term. The second term $\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle\, x_i\, x_j$ captures all pairwise interactions in x (the input vector) using the factorization of matrix v.

Inter-Alignment Factorization This operation compares the alignment between inter-attention aligned representations, i.e., (β_i, h_i) and (α_j, p_j). Let (a, b) represent an alignment pair, we apply the following operations:

$$y_c = Z([a; b]) \; ; \; y_s = Z(a - b) \; ; \; y_m = Z(a \odot b) \quad (7)$$

where $y_c, y_s, y_m \in \mathbb{R}$, $Z(.)$ is the factorization operation, $[.; .]$ is the concatenation operator and $\odot$ is the element-wise multiplication. The intuition of modeling subtraction is targeted at capturing contradiction. However, instead of simply concatenating the extra comparison vectors, we compress them using the factorization operation. Finally, for each alignment pair, we obtain three scalar-valued features which map precisely to a word in the sequence.

Intra-Alignment Factorization Next, for each sequence, we also apply alignment factorization on the intra-aligned sentences. Let (s, s') represent an *intra-aligned* pair from either the premise or hypothesis, we compute the following operations:

$$v_c = Z([s; s'])\ ;\ v_s = Z(s - s')\ ;\ v_m = Z(s \odot s') \tag{8}$$

where $v_c, v_s, v_m \in \mathbb{R}$ and $Z(.)$ is the factorization operation. Applying alignment factorization to intra-aligned representations produces another three scalar-valued features which are mapped to each word in the sequence. Note that each of the *six* factorization operations has its own parameters but shares them amongst all words in the sentences.

3.4 Propagation and Augmentation

Finally, the *six* factorized features are then aggregated[1] via concatenation to form a final feature vector that is propagated to upper representation learning layers via augmentation of the word representation $\bar{P}$ or $\bar{H}$.

$$u_i = [s_i; f^i_{intra}; f^i_{inter}] \tag{9}$$

where s_i is i-th word in $\bar{P}$ or $\bar{H}$, and f^i_{intra} and f^i_{inter} are the intra-aligned $[v_c; v_s; v_m]$ and inter-aligned $[y_c; y_s; y_m]$ features for the i-th word in the sequence respectively. Intuitively, f^i_{intra} augments each word with global knowledge of the sentence and f^i_{inter} augments each word with cross-sentence knowledge via inter-attention.

3.5 Sequential Encoder Layer

For each sentence, the augmented word representations $u_1, u_2, \ldots u_\ell$ are then passed into a sequential encoder layer. We adopt a standard vanilla LSTM encoder.

$$h_i = LSTM(u, i), \forall i \in [1, \ldots \ell] \tag{10}$$

where ℓ represents the maximum length of the sequence. Notably, the parameters of the LSTM are *siamese* in nature, sharing weights between both premise and hypothesis. We do not use a bidirectional LSTM encoder, as we found that it did not lead to any improvements on the held-out set. A logical explanation would be because our word representations are already augmented with global (intra-attention) information. As such, modeling in the reverse direction is unnecessary, resulting in some computational savings.

Pooling Layer Next, to learn an overall representation of each sentence, we apply a pooling function across all hidden outputs of the sequential encoder. The pooling function is a concatenation of temporal max and average (avg) pooling.

$$x = [\max([h_1, \cdots h_\ell]); avg([h_1, \cdots h_\ell])] \tag{11}$$

where x is a final $2k$-dimensional representation of the sentence (premise or hypothesis). We also experimented with *sum* and *avg* standalone poolings and found *sum* pooling to be relatively competitive.

3.6 Prediction Layer

Finally, given a fixed dimensional representation of the premise x_p and hypothesis x_h, we pass their concatenation into a two-layer h-dimensional highway network. Since the highway network has been already defined earlier, we omit the technical details here. The final prediction layer of our model is computed as follows:

$$y_{out} = H_2(H_1([x_p; x_h; x_p \odot x_h; x_p - x_h])) \tag{12}$$

where $H_1(.), H_2(.)$ are highway network layers with ReLU activation. The output is then passed into a final linear softmax layer.

$$y_{pred} = softmax(W_F \cdot y_{out} + b_F) \tag{13}$$

where $W_F \in \mathbb{R}^{h \times 3}$ and $b_F \in \mathbb{R}^3$. The network is then trained using standard multi-class cross entropy loss with L2 regularization.

4 Experiments

In this section, we describe our experimental setup and report our experimental results.

---

[1] Following (Parikh et al., 2016), we may also concatenate the intra-aligned vector to u_i which we found to have speed up convergence.

Model	Params	Train	Test
Single Model (w/o Cross Sentence Attention)			
300D Gumbel TreeLSTM (Choi et al., 2017)	2.9M	91.2	85.6
300D DISAN (Shen et al., 2017)	2.4M	91.1	85.6
300D Residual Stacked Encoders (Nie and Bansal, 2017)	9.7M	89.8	85.7
600D Gumbel TreeLSTM (Choi et al., 2017)	10M	93.1	**86.0**
300D CAFE (w/o CA)	3.7M	87.3	85.9
Single Models			
100D LSTM with attention (Rocktäschel et al., 2015)	250K	85.3	83.5
300D mLSTM (Wang and Jiang, 2016b)	1.9M	92.0	86.1
450D LSTMN + deep att. fusion (Cheng et al., 2016)	3.4M	88.5	86.3
200D DecompAtt + Intra-Att (Parikh et al., 2016)	580K	90.5	86.8
300D NTI-SLSTM-LSTM (Yu and Munkhdalai, 2017b)	3.2M	88.5	87.3
300D re-read LSTM (Sha et al., 2016)	2.0M	90.7	87.5
BiMPM (Wang et al., 2017)	1.6M	90.9	87.5
448D DIIN (Gong et al., 2017)	4.4M	91.2	88.0
600D ESIM (Chen et al., 2017b)	4.3M	92.6	88.0
150D CAFE (SUM+2x200D MLP)	750K	88.2	87.7
200D CAFE (SUM+2x400D MLP)	1.4M	89.4	88.1
300D CAFE (SUM+2x600D MLP)	3.5M	89.2	88.3
300D CAFE (AVGMAX+300D HN)	4.7M	89.8	**88.5**
Ensemble Models			
600D ESIM + 300D Tree-LSTM (Chen et al., 2017b)	7.7M	93.5	88.6
BiMPM (Wang et al., 2017)	6.4M	93.2	88.8
448D DIIN (Gong et al., 2017)	17.0M	92.3	88.9
300D CAFE (Ensemble)	17.5M	92.5	**89.3**
External Resource Models			
BiAttentive Classification + CoVe + Char (McCann et al., 2017)	22M	88.5	88.1
KIM (Chen et al., 2017a)	4.3M	94.1	88.6
ESIM + ELMo (Peters et al., 2018)	8.0M	91.6	88.7
200D CAFE (AVGMAX + 200D MLP) + ELMo	1.4M	89.5	**89.0**

Table 1: Performance comparison of all published models on the SNLI benchmark.

4.1 Experimental Setup

To ascertain the effectiveness of our models, we use the SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2017) benchmarks which are standard and highly competitive benchmarks for the NLI task. We also include the newly released SciTail dataset (Khot et al., 2018) which is a binary entailment classification task constructed from science questions. Notably, SciTail is known to be a difficult dataset for NLI, made evident by the low accuracy scores even though it is binary in nature.

SNLI The state-of-the-art competitors on this dataset are the BiMPM (Wang et al., 2017), ESIM (Chen et al., 2017b) and DIIN (Gong et al., 2017). We compare against competitors across three settings. The first setting disallows cross sentence attention. In the second setting, cross sentence is allowed. The last (third) setting is a comparison between model ensembles while the first two settings only comprise single models. Note that we consider the 1st setting to be relatively less important (since our focus is not on the encoder itself) but still report the results for completeness.

MultiNLI We compare on two test sets (*matched* and *mismatched*) which represent in-domain and out-domain performance. The main competitor on this dataset is the ESIM model, a powerful state-of-the-art SNLI baseline. We also compare with ESIM + Read (Weissenborn, 2017).

SciTail This dataset only has one official setting. We compare against the reported results of ESIM (Chen et al., 2017b) and DecompAtt (Parikh et al.,

2016) in the original paper. We also compare with DGEM, the new model proposed in (Khot et al., 2018).

Across all experiments and in the spirit of fair comparison, we only compare with works that (1) do not use extra training data and (2) do not use external resources (such as external knowledge bases, etc.). However, for the sake of completeness, we still report their scores[2] (McCann et al., 2017; Chen et al., 2017a; Peters et al., 2018).

4.2 Implementation Details

We implement our model in TensorFlow (Abadi et al., 2015) and train them on Nvidia P100 GPUs. We use the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.0003. L2 regularization is set to 10^{-6}. Dropout with a keep probability of 0.8 is applied after each fully-connected, recurrent or highway layer. The batch size is tuned amongst $\{128, 256, 512\}$. The number of latent factors k for the factorization layer is tuned amongst $\{5, 10, 50, 100, 150\}$. The size of the hidden layers of the highway network layers are set to 300. All parameters are initialized with xavier initialization. Word embeddings are preloaded with $300d$ GloVe embeddings (Pennington et al., 2014) and fixed during training. Sequence lengths are padded to batch-wise maximum. The batch order is (randomly) sorted within buckets following (Parikh et al., 2016).

4.3 Experimental Results

Table 1 reports our results on the SNLI benchmark. On the cross sentence (single model setting), the performance of our proposed CAFE model is extremely competitive. We report the test accuracy of CAFE at different extents of parameterization, i.e., varying the size of the LSTM encoder, width of the pre-softmax hidden layers and final pooling layer. CAFE obtains 88.5% accuracy on the SNLI test set, an extremely competitive score on the extremely popular benchmark. Notably, competitive results can be also achieved with a much smaller parameterization. For example, CAFE also achieves 88.3% and 88.1% test accuracy with only $3.5M$ and $1.5M$ parameters respectively. This outperforms the state-of-the-art ESIM and DIIN models with only a fraction of the parameter cost. At 88.1%, our model has about three times less parameters than ESIM/DIIN (i.e., 1.4M versus 4.3M/4.4M). Moreover, our lightweight adaptation achieves 87.7% with only $750K$ parameters, which makes it extremely performant amongst models having the same amount of parameters such as the decomposable attention model (86.8%).

Finally, an ensemble of 5 CAFE models achieves 89.3% test accuracy, the best test scores on the SNLI benchmark to date[3]. Overall, we believe that the good performance of our CAFE can be attributed to (1) the effectiveness of the ComProp architecture (i.e., providing word representations with global and local knowledge for better representation learning) and (2) the expressiveness of alignment factorization layers that are used to decompose and compare word alignments. More details are given at the ablation study. Finally, we emphasize that CAFE is also relatively lightweight, efficient and fast to train given its performance. A single run on SNLI takes approximately 5 minutes per epoch with a batch size of 256. Overall, a single run takes $\approx$ 3 hours to get to convergence.

	MultiNLI		SciTail
Model	Match	Mismatch	-
Majority	36.5	35.6	60.3
NGRAM$^{\#}$	-	-	70.6
CBOW$^{\flat}$	65.2	64.8	-
BiLSTM$^{\flat}$	69.8	69.4	-
ESIM$^{\#,\flat}$	72.4	72.1	70.6
DecompAtt$^{\#}$ -	-	-	72.3
DGEM$^{\#}$	-	-	70.8
DGEM + Edge$^{\#}$	-	-	77.3
ESIM†	76.3	75.8	-
ESIM + Read†	77.8	77.0	-
CAFE	78.7	77.9	**83.3**
CAFE Ensemble	**80.2**	**79.0**	-

Table 2: Performance comparison (accuracy) on MultiNLI and SciTail. Models with $\dagger$, $\#$ and $\flat$ are reported from (Weissenborn, 2017), (Khot et al., 2018) and (Williams et al., 2017) respectively.

Table 2 reports our results on the MultiNLI and SciTail datasets. On MultiNLI, CAFE significantly outperforms ESIM, a strong state-of-the-art model on both settings. We also outperform the ESIM + Read model (Weissenborn, 2017). An ensemble of CAFE models achieve competitive re-

---

[2] Additionally, we added ELMo (Peters et al., 2018) to our CAFE model at the embedding layer. We report CAFE + ELMo under external resource models. This was done post review after EMNLP. Due to resource constraints, we did not train CAFE + ELMo ensembles but a single run (and single model) of CAFE + ELMo already achieves 89.0 score on SNLI.

[3] As of 22nd May 2018, the deadline of the EMNLP submisssion.

sult on the MultiNLI dataset. On SciTail, our proposed CAFE model achieves state-of-the-art performance. The performance gain over strong baselines such as DecompAtt and ESIM are $\approx 10\% - 13\%$ in terms of accuracy. CAFE also outperforms DGEM, which uses a graph-based attention for improved performance, by a significant margin of 5%. As such, empirical results demonstrate the effectiveness of our proposed CAFE model on the challenging SciTail dataset.

4.4 Ablation Study

	Match	Mismatch
Original Model	79.0	78.9
(1a) Rm FM for 1L-FC	77.7	77.9
(1b) Rm FM for 1L-FC (ReLU)	77.3	77.5
(1c) Rm FM for 2L-FC (ReLU)	76.6	76.4
(2) Remove Char Embed	78.1	78.3
(3) Remove Syn Embed	78.3	78.4
(4) Remove Inter Att	75.2	75.6
(5) Replace HW Pred. with FC	77.7	77.9
(6) Replace HW Enc. with FC	78.7	78.7
(7) Remove Sub Feat	77.9	78.3
(8) Remove Mul Feat	78.7	78.6
(9) Remove Concat Feat	77.9	77.6
(10) Add Bi-directional	78.3	78.4

Table 3: Ablation study on MultiNLI development sets. HW stands for Highway.

Table 3 reports ablation studies on the MultiNLI development sets. In (1), we replaced all FM functions with regular full-connected (FC) layers in order to observe the effect of **FM versus FC**. More specifically, we experimented with several FC configurations as follows: (a) 1-layer linear, (b) 1-layer ReLU (c) 2-layer ReLU. The 1-layer linear setting performs the best and is therefore reported in Table 3. Using ReLU seems to be worse than nonlinear FC layers. Overall, the best combination (option a) still experienced a decline in performance in both development sets.

In (2-3), we explore the utility of using character and syntactic embeddings, which we found to have helped CAFE marginally. In (4), we remove the inter-attention alignment features, which naturally impact the model performance significantly. In (5-6), we explore the effectiveness of the highway layers (in prediction layers and encoding layers) by replacing them to FC layers. We observe that both highway layers have marginally helped the overall performance. Finally, in (7-9), we remove the alignment features based on their composition type. We observe that the *Sub* and *Concat* compositions were more important than the *Mul* composition. However, removing any of the three will result in some performance degradation. Finally, in (10), we replace the LSTM encoder with a BiLSTM, observing that adding bi-directionality did not improve performance for our model.

4.5 Linguistic Error Analysis

We perform a linguistic error analysis using the supplementary annotations provided by the MultiNLI dataset. We compare against the model outputs of the ESIM model across 13 categories of linguistic phenenoma (Williams et al., 2017). Table 4 reports the result of our error analysis. We observe that our CAFE model generally outperforms ESIM on *most categories*.

	Matched		Mismatched	
	ESIM	CAFE	ESIM	CAFE
Conditional	100	70	60	**85**
Word overlap	50	**82**	62	**87**
Negation	**76**	**76**	71	**80**
Antonym	67	**82**	58	**80**
Long Sentence	75	**79**	69	**77**
Tense Difference	73	**82**	79	**89**
Active/Passive	88	**100**	**91**	90
Paraphrase	**89**	88	84	**95**
Quantity/Time	33	**53**	54	**62**
Coreference	**83**	80	75	**83**
Quantifier	69	**75**	72	**80**
Modal	78	**81**	76	**81**
Belief	65	**77**	67	**83**

Table 4: Linguistic Error Analysis on MultiNLI dataset.

On the mismatched setting, CAFE outperforms ESIM in 12 out of 13 categories, losing only in one percentage point in *Active/Passive* category. On the matched setting, CAFE is outperformed by ESIM very marginally on coreference and paraphrase categories. Despite generally achieving much superior results, we noticed that CAFE performs poorly on *conditionals*[4] on the matched setting. Measuring the absolute ability of CAFE, we find that CAFE performs extremely well in handling linguistic patterns of *paraphrase detection* and *active/passive*. This is likely to be attributed by the alignment strategy that CAFE and ESIM both exploits.

4.6 Interpreting and Visualizing with CAFE

Finally, we also observed that the propagated features are highly interpretable, giving insights to the inner workings of the CAFE model. Figure 2 shows a visualization of the feature values from an example in the SNLI test set. The ground

---

[4] This only accounts for 5% of samples.

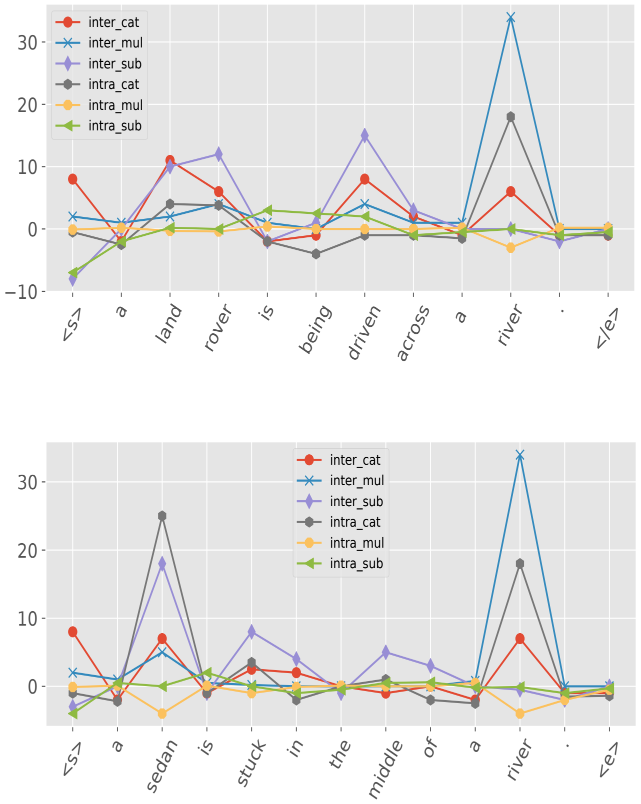

Figure 2: Visualization of *six* Propagated Features (*Best viewed in color*). Legend is denoted by {inter,intra} followed by the operations *mul*, *sub* or *cat* (concat).

truth is *contradiction*. Based on the above example we make several observations. Firstly, *inter_mul* features mostly capture identical words (or semantically similar words), i.e., *inter_mul* features for *'river'* spikes in both sentences. Secondly, *inter_sub* spikes on conflicting words that might cause contradiction, e.g., *'sedan'* and *'land rover'* are not the same vehicle. Another interesting observation is that we notice the *inter_sub* features for *driven* and *stuck* spiking. This also validates the observation of (Chen et al., 2017b), which shows what the *sub* vector in the ESIM model is looking out for contradictory information. However, our architecture allows the inspection of these vectors since they are compressed via factorization, leading to larger extents of explainability - a quality that neural models inherently lack. We also observed that intra-attention (e.g., intra_cat) features seem to capture the more important words in the sentence (*'river'*, *'sedan'*, *'land rover'*).

5 Conclusion

We proposed a new neural architecture, CAFE for NLI. CAFE achieves very competitive performance on three benchmark datasets. Extensive ablation studies confirm the effectiveness of FM layers over FC layers. Qualitatively, we show how different compositional operators (e.g., sub and mul) behave in NLI task and shed light on why subtractive composition helps in other models such as ESIM.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems .

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 632–642.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. 2017a. Natural language inference with external knowledge. *arXiv preprint arXiv:1711.04289* .

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 1657–1668. https://doi.org/10.18653/v1/P17-1152.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017c. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*. pages 36–40.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 551–561.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-lstms. *arXiv preprint arXiv:1707.02786* .

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*. Springer-Verlag, Berlin, Heidelberg, MLCW'05, pages 177–190.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *CoRR* abs/1709.04348.

Adrian Iftene and Alexandra Balahur-Dobrescu. 2007. Hypothesis transformation and semantic variability rules used in recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics, Stroudsburg, PA, USA, RTE '07, pages 125–130.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Bill Maccartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford, CA, USA. AAI3364139.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '08, pages 521–528.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*. pages 6297–6308.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*. pages 41–45.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, pages 995–1000.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664* .

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read LSTM unit for textual entailment recognition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 2870–2879.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *CoRR* abs/1709.04696.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR* abs/1505.00387.

Shuohang Wang and Jing Jiang. 2016a. A compare-aggregate model for matching text sequences. *CoRR* abs/1611.01747.

Shuohang Wang and Jing Jiang. 2016b. Learning natural language inference with LSTM. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 1442–1451.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. pages 4144–4150. https://doi.org/10.24963/ijcai.2017/579.

Dirk Weissenborn. 2017. Reading twice for natural language understanding. *CoRR* abs/1706.02596.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR* abs/1704.05426.

Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* .

Hong Yu and Tsendsuren Munkhdalai. 2017a. Neural semantic encoders. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. pages 397–407.

Hong Yu and Tsendsuren Munkhdalai. 2017b. Neural tree indexers for text understanding. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. pages 11–21.

Convolutional Interaction Network for Natural Language Inference

Jingjing Gong[‡], **Xipeng Qiu**[*‡], **Xinchi Chen**[‡], **Dong Liang**[§ †], **Xuanjing Huang**[‡]
[‡] Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
[‡] School of Computer Science, Fudan University
[§] State Key Laboratory of Information Security, Institute of Information Engineering CAS
{jjgong15, xpqiu, xinchichen13, xjhuang}@fudan.edu.cn,liangdong@iie.ac.cn

Abstract

Attention-based neural models have achieved great success in natural language inference (NLI). In this paper, we propose the Convolutional Interaction Network (CIN), a general model to capture the interaction between two sentences, which can be an alternative to the attention mechanism for NLI. Specifically, CIN encodes one sentence with the filters dynamically generated based on another sentence. Since the filters may be designed to have various numbers and sizes, CIN can capture more complicated interaction patterns. Experiments on three very large datasets demonstrate CIN's efficacy.

1 Introduction

Natural language inference (NLI) is a pivotal and challenging natural language processing (NLP) task. The goal of NLI is to identify the logical relationship (entailment, neutral, or contradiction) between a premise and a corresponding hypothesis. Generally, NLI is also related to many other NLP tasks under the paradigm of semantic matching of two texts, such as question answering Hu et al. (2014); Wan et al. (2016) and information retrieval Liu et al. (2015), and so on. An essential challenge is to capture the semantic relevance of two sentences. Due to the semantic gap (or lexical chasm) problem, natural language inference is still a challenging problem.

Recently, deep learning is raising a substantial interest in natural language inference and has achieved some great progresses Hu et al. (2014); Parikh et al. (2016); Chen et al. (2017a). To model the complicated semantic relationship between two sentences, previous models heavily utilize various attention mechanism Bahdanau et al. (2014); Vaswani et al. (2017) to build the interaction at different granularity (word, phrase and sentence level), such as ABCNN Yin et al. (2016), Attention LSTM Rocktäschel et al. (2015), bidirectional attention LSTM Chen et al. (2017a), and so on. While attention is very successful in natural language inference, its mechanism is quite simple and can be regarded as a weighted sum of the target vectors. This paradigm results in a lack of flexibility in more complicated interaction model.

In this paper, we propose a new interaction module, called Convolutional Interaction Network (CIN), which can serve as an alternative module of attention mechanism. Specifically, CIN utilizes convolutional neural network to extract the valued features (or representations) from sentences. In the case of NLI, whether a feature of one sentence being important depends on another sentence. Inspired by the idea of using one network to generate the parameters of another network Ha et al. (2016a); De Brabandere et al. (2016), we introduce a filter generation network to dynamically generate convolutional filters. Each sentence is convolved by a dynamically generated filter by another sentence. Thus, the convolved features of one sentence can be regarded as context-aware representations under the influence of another sentence.

The contributions of this paper can be summarized as follows.

1. CIN is a new interaction model, invented as an alternative module to the attention model. CIN can also capture both the intra- or inter-interactions of two sentences.

2. Compared to attention model, CIN is more general and flexible to capture the complicated interaction. As discussed in Section

---

[*] Corresponding Author.

[†] Contribution during internship at Fudan University.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1576–1585
Brussels, Belgium, October 31 - November 4, 2018.

3.3, the attention model is approximately equivalent to a special case of CIN.

3. We perform extensive empirical studies on three very large datasets. Experiment results demonstrate that our proposed architecture is effective for natural language inference.

2 Attentive Interaction for Natural Language Inference

Currently, the dominative method for natural language inference is to use attention mechanism to model the interaction between two sentence.

Given two input sentences $x = [x_1, x_2, \cdots, x_m]$ and $y = [y_1, y_2, \cdots, y_n]$ with length m and n respectively, we first encode them into two vectorial sequences

$$X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m] \in \mathbb{R}^{d\times m}, \quad (1)$$

$$Y = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n] \in \mathbb{R}^{d\times n}. \quad (2)$$

The encoder usually consists of one or several CNN/RNN layers to get the context-aware token representations.

To capture the interaction between two sentence, various neural attentions can be used, such as sentence2word attention Rocktäschel et al. (2015), word2word attention Parikh et al. (2016); Chen et al. (2017a).

Word2word Attentive Interaction The word2word attention captures the dependency between two words x_i and y_j from the concerned two sentences respectively. The word2word attention computes a similarity matrix M, in which each element $m_{i,j}$ is the alignment score between $\mathbf{x}_i$ and $\mathbf{y}_j$.

$$m_{i,j} = f(\mathbf{x}_i, \mathbf{y}_j),\ 1 \le i \le m, 1 \le j \le n, \quad (3)$$

where f is a score function.

There are two most prevalent attention functions: multiplicative attention and additive attention. Multiplicative attention is:

$$f(\mathbf{x}_i, \mathbf{y}_j) = \mathbf{x}_i^{\mathrm{T}} \mathbf{y}_j. \quad (4)$$

Additive attention computes a compatibility function by a feed-forward network with a single hidden layer.

$$f(\mathbf{x}_i, \mathbf{y}_j) = \mathbf{w}^{\top} \sigma(\mathbf{W}_1 \mathbf{x}_i + \mathbf{W}_2 \mathbf{y}_j + \mathbf{b}), \quad (5)$$

where $\mathbf{w}$, $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{b}$ are learnable parameters.

While these two kinds of attentions have similar performance, the multiplicative attention is more popular in practice since it requires less computation power and have less memory demand with optimized matrix multiplication. With multiplicative attention, we can compute the *mimic* representations for both X and Y.

$$\bar{X} = Y \mathrm{softmax}(Y^{\mathrm{T}} X) \in \mathbb{R}^{d\times m}, \quad (6)$$

$$\bar{Y} = X \mathrm{softmax}(X^{\mathrm{T}} Y) \in \mathbb{R}^{d\times n}, \quad (7)$$

where $\mathrm{softmax}(\cdot)$ is column-wise normalization function. Each vector $\bar{\mathbf{x}}_i \in \bar{X}$ is called as *mimic vector*, which is a weighted summation of $\{\mathbf{y}_j\}_{j=1}^{n}$. Intuitively, the mimic vector $\bar{\mathbf{x}}_i$ provides the related information of token x_i extracted from sentence Y.

Prediction After interaction, a prediction module is used to aggregate the interaction information and extract the fix-length representation of two sentences. Finally, the final sentence representations are fed into a feed-forward network to predict the relationship between two sentences.

3 Convolutional Interaction Network

In this section, we propose a new interaction method by utilizing dynamic convolutional filters, called Convolutional Interaction Network (CIN). CIN can serve as an alternative module of attention mechanism.

We first briefly introduce how the convolution works over text sequence, then describe the proposed model and its connection to attention model.

3.1 Convolution over Sequence

Convolution is an effective operation in deep neural networks, which convolves the input with a set of filters to extract non-linear compositional features. Although originally designed for computer vision, convolutional models have subsequently shown to be effective for NLP and have achieved excellent performance in sentence modeling Kim (2014); Kalchbrenner et al. (2014), and other traditional NLP tasks Hu et al. (2014); Zeng et al. (2014); Gehring et al. (2017).

Given a sentence representation $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m] \in \mathbb{R}^{d\times m}$, a convolutional filter $W^{(f)} \in \mathbb{R}^{d\times kd}$, the convolution process is defined as

$$\mathbf{x}'_i = f\left(W^{(f)}[\mathbf{x}_{i-[k/2]}, \cdots, \mathbf{x}_{i+[k/2]}] + \mathbf{b}^{(f)}\right), \quad (8)$$

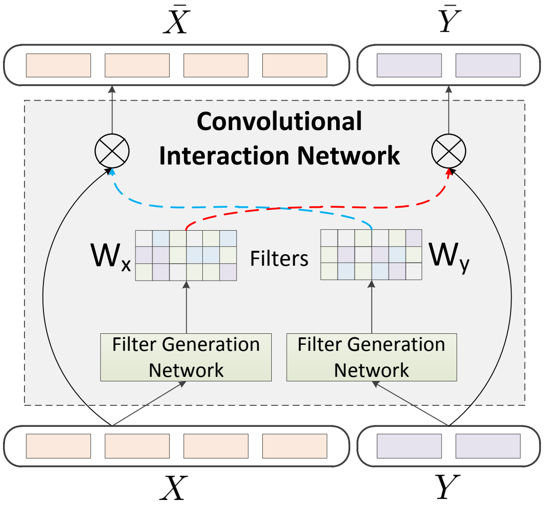

Figure 1: Convolutional Interaction Network. $\otimes$ denotes the convolution operation.

where $f(\cdot)$ is a non-linear activation function, such as ReLU, k indicates the size of convolution window, and $\mathbf{b}^{(f)} \in \mathbb{R}^d$ is a bias vector.

The convolution can be abbreviated as

$$X' = f(W^{(f)} \otimes X) \in \mathbb{R}^{d \times m}, \tag{9}$$

where $\otimes$ denotes the convolutional operation. To ensure the output of convolution has equal length as to the input, we pad $\lfloor \frac{k}{2} \rfloor$ zero vectors on both sides of the input.

3.2 Convolutional Interaction Network

Convolution is very effective when it comes to extracting useful features from a sentence. But for NLI, whether a word (or feature) being important in one sentence depends on another sentence. Therefore, a better convolution operation should have the ability to extract substantial features from one sentence according to another sentence. Thus, the convolutional filter should be dynamically changeable. Inspired by Jia et al. (2016); Ha et al. (2016b), we propose a *filter generation network* (FGN) to generate a dynamical filter, which is used to extract the context-aware information.

Given two sentences x, y, and their representations $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m] \in \mathbb{R}^{d \times m}$ and $Y = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n] \in \mathbb{R}^{d \times n}$, the filter for each sentence is generated according to the other sentence by

$$W_x^{(f)} = \text{FGN}(X) \in \mathbb{R}^{d \times \tau d}, \tag{10}$$
$$W_y^{(f)} = \text{FGN}(Y) \in \mathbb{R}^{d \times \tau d}, \tag{11}$$

where τ is the width of filter, $\text{FGN}(\cdot)$ is the filter generation network. A detailed implementation of FGN is presented in Section 3.4.

Now we can convolve the two sentences with the generated filters.

$$\bar{X} = f(W_y^{(f)} \otimes X) \in \mathbb{R}^{d \times m}, \tag{12}$$
$$\bar{Y} = f(W_x^{(f)} \otimes Y) \in \mathbb{R}^{d \times n}, \tag{13}$$

where the attained matrix $\bar{X}$ and $\bar{Y}$ can be regarded as the context-aware representation of sentences x and y, depending on each other.

Figure 1 gives an illustration of CIN.

3.3 Connection to Attentive Interaction

CIN is more general than attention model. Assuming that we set $k = 1$ and FGN to be a function of $\text{FGN}(X) = XX^{\mathrm{T}}$, Eq. (12) and (13) of CIN can be written as

$$\bar{X} = (YY^{\mathrm{T}})X = Y(Y^{\mathrm{T}}X), \tag{14}$$
$$\bar{Y} = (XX^{\mathrm{T}})Y = X(X^{\mathrm{T}}Y). \tag{15}$$

Compared to Eq. (6) and (7), under the above assumption, CIN is equivalent to the word2word multiplicative attention model without softmax normalization.

3.4 An Implementation of Filter Generation Network (FGN)

To generate the dynamic filters, the key factor is how to choose the filter generation network $\text{FGN}(\cdot)$ in Eq. (10) and (11). Although many sophisticated networks can be employed, we give an simple implementation in this paper.

For ease of presentation, we only describe how we generate dynamical filter according to sentence x. The same procedure is utilized for sentence y.

Firstly, we summarize the information of sentence x with an over-time k-max pooling on X,

$$U_x = \text{ReLU}(W_u \otimes X) \in \mathbb{R}^{d \times m}, \tag{16}$$
$$\mathbf{z}_x^{1:k} = \text{k-max}(U_x) \in \mathbb{R}^{d \times k}, \tag{17}$$

where U_x is a non-linear transformation of X by convolution filter $W_u \in \mathbb{R}^{d \times d}$. The idea of k-max pooling is to capture the most important features (the k highest values) from sentence X.

Then we generate k filters W_x^j for $j = 1, \cdots, k$ by

$$W_x^j = \text{ReLU}\left(P\text{diag}(\mathbf{z}_x^j)Q + B\right), \tag{18}$$

where $P \in \mathbb{R}^{\frac{d}{k} \times d}$, $Q \in \mathbb{R}^{d \times \tau d}$ and $B \in \mathbb{R}^{\frac{d}{k} \times \tau d}$ are learnable parameters.

The final filter is obtained by concatenating the k generated filters,

$$W_x^{(f)} = [W_x^1; W_x^2; \cdots; W_x^k] \in \mathbb{R}^{d \times \tau d}. \tag{19}$$

Similar to x, we can also obtain the dynamic filters $W_y^{(f)}$ according to the sentence y.

4 Incorporating CIN into a Deep Network Architecture for NLI

Our overall network architecture for NLI is based on a successful model proposed by Chen et al. (2017a). The major difference is that we use CIN to capture the interaction, instead of bi-directional attention.

The network architecture consists of three components: (1) an encoding layer; (2) convolutional interaction layers; (3) a prediction layer. Figure 2 gives an illustration.

4.1 Encoding Layer

The input of natural language inference task is a pair of sentences x and y. Since each word in a sentence is a symbol that can not be directly processed by neural networks, we need first map each word to a d dimensional embedding vector.

Thus, the two sentences are mapped to two matrix $E_x \in \mathbb{R}^{d_e \times m}$ and $E_y \in \mathbb{R}^{d_e \times n}$ respectively. We also use syntactical and lexical information such as part of speech tagging information, exact match feature and character representation. In this paper, exact match value of each word is set to 1(default to be 0) if the word concerned share the same stem or lemma with any word in counterpart sentence. Character representation of the word is obtained using a convolution neural network followed by a max pooling along sequence length dimension as same as Kim (2014). The final representation of word is a concatenation of word embedding, character encoded vector, POS tagging embedding and exact match feature. Both character embedding and POS tagging embedding are randomly initialized. All embeddings are updated during training.

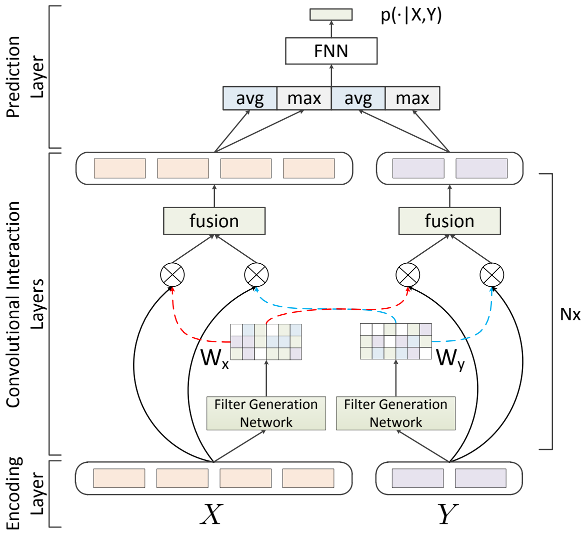

Figure 2: The overall network architecture for natural language inference. The N_x means the number of the stacking interaction layers.

We use bi-directional LSTM (BiLSTM) Hochreiter and Schmidhuber (1997) to incorporate the forward and backward context information of sequence. Thus, we can get the phrase-level encoding of two input sentences,

$$X = [\text{Bi-LSTM}(E_x); E_x], \tag{20}$$
$$Y = [\text{Bi-LSTM}(E_y); E_y], \tag{21}$$

where $X \in \mathbb{R}^{d \times m}$ and $Y \in \mathbb{R}^{d \times n}$ are the phrase-level encoding representation of sentence x and y respectively.

4.2 Convolutional Interaction Layers

In the interaction layers, we use our proposed CIN to model the interaction between two sentences.

We first dynamically generate context-aware filters $W_x^{(f)}$ and $W_y^{(f)}$ based on the sentence encodings X and Y respectively, which are further used for both intra-sentence and inter-sentence interaction.

Intra-Sentence Interaction The intra-sentence convolutional interaction is to convolve one sentence by the filter generated by itself.

$$X_{intra} = f(W_x^{(f)} \otimes X), \tag{22}$$
$$Y_{intra} = f(W_y^{(f)} \otimes Y), \tag{23}$$

The role of the intra-sentence convolutional interaction is the same as self-attention Shen et al. (2017), which is also very useful in NLI.

Inter-Sentence Interaction The inter-sentence interaction takes filters generated from the counterpart sentence to convolve the inputs.

$$X_{inter} = f(W_x^{(f)} \otimes Y), \tag{24}$$

$$Y_{inter} = f(W_y^{(f)} \otimes X), \tag{25}$$

The inter-sentence convolutional interaction plays a role similar to the cross-attention between two sentences.

Fusion Layer After CIN, we can fuse two kinds of context-aware representations of each sentence. For sentence x, the X_{intra} and X_{inter} represent the extracted features under consideration of information of itself and sentence y respectively.

To efficiently utilize X_{intra} and X_{inter}, a fusion layer is used. We use the comparing operation proposed in Chen et al. (2017a) to fuse the two kinds of representation. Let $\mathbf{u}_i$ and $\mathbf{v}_i$ be intra and inter attentive vector of the i-th word in sentence x, a heuristic and effective composition operator is used to combine two vectors.

$$\bar{\mathbf{x}}_i^{(c)} = [\mathbf{u}_i; \mathbf{v}_i; \mathbf{u}_i - \mathbf{v}_i; \mathbf{u}_i \odot \mathbf{v}_i; |\mathbf{u}_i - \mathbf{v}_i|], \tag{26}$$

$$\mathbf{x}_i^{(c)} = \mathrm{ReLU}(\mathbf{W}_c \bar{\mathbf{x}}_i^c + \mathbf{W}_x \mathbf{x}_i + \mathbf{b}_c), \tag{27}$$

Thus, we can obtain two fused representations $X^{(c)}$ and $Y^{(c)}$ for two sentences, which are further fed into the prediction layer or another stacked interaction layer. The interaction layers can be stacked for N_x times to capture the complicated matching information.

4.3 Prediction Layer

After interaction layers, an aggregation layer is employed to aggregate the two sequences of fustion vectors $X^{(c)}$ and $Y^{(c)}$ into a fixed-length matching vectors. The aggregation component usually consists of another BiLSTM layer and a following pooling layer. We then perform max pooling over time for both $X^{(c)}$ and $Y^{(c)}$ to get two fix representation vector for two sentences, $\mathbf{p}$ and $\mathbf{h}$:

$$\mathbf{p} = \max(X^{(c)}), \tag{28}$$

$$\mathbf{h} = \max(Y^{(c)}), \tag{29}$$

where the functions max is the max pooling operations over time steps.

Finally, the pooled vectors are composed as one relation vector and fed into a feed-forward network to predict the relationship between two sentences. Specially, the two-layer feed-forward network has one hidden layers with *tanh* activation and *softmax* output layer in our experiments.

	Train	Dev	Test	Len(P)	Len(H)	Vocab
SNLI	549K	9.8K	9.8K	14	8	36K
MultiNLI[1]	392K	9.8K	9.8K	22	11	85K
MultiNLI[2]	392K	9.8K	9.8K	22	11	85K
Quora	384K	10K	10K	12	12	107K

Table 1: Statistics of three datasets: SNLI, MultiNLI, Quora. Len(P) and Len(H) refer to the average length of two sentences. MultiNLI[1] and MultiNLI[2] indicate the in-domain and cross-domain datasets.

$$\mathbf{m} = [\mathbf{p}; \mathbf{h}; \mathbf{p} - \mathbf{h}; \mathbf{p} * \mathbf{h}; |\mathbf{p} - \mathbf{h}|], \tag{30}$$

$$p(\cdot|x, y) = \mathrm{FNN}(\mathbf{m}). \tag{31}$$

5 Training

Given a trainset $\{x^{(i)}, y^{(i)}, t^{(i)}\}_{i=1}^N$, the objective is to minimize a cross entropy loss $\mathcal{J}(\theta)$:

$$\mathcal{J}(\theta) = -\frac{1}{N}\sum_i \log p(t^{(i)}|x^{(i)}, y^{(i)}) + \lambda||\theta||_2^2, \tag{32}$$

where θ represents all the connection weights.

We use the Adam optimizer Kingma and Ba (2014) with an initial learning rate of 0.0004. Default L2 regularization λ is set to 10^{-6}. To avoid overfitting, dropout is applied after each fully connected, recurrent or convolutional layer.

Initialization We take advantage of pre-trained word embeddings such as Glove Pennington et al. (2014) to transfer more knowledge from vast unlabeled data. For the words that don't appear in Glove, we randomly initialize their embeddings from a normal distribution with mean 0.0 and standard deviation 0.1.

The network weights are initialized with Xavier normalization Glorot and Bengio (2010) to maintain the variance of activations throughout the forward and backward passes. Biases are uniformly set to zero when the network is constructed.

5.1 Datasets

To make quantitative evaluation, our model was evaluated on three well known datasets: Stanford Natural Language Inference dataset (SNLI), MultiNLI dataset and Quora Question pair dataset (Quora). Detailed statistical information of these datasets is shown in Table 1.

Models	**Train**	**Test**
Handcrafted features (Bowman et al., 2015)	99.7	78.2
LSTM with attention (Rocktäschel et al., 2015)	85.3	83.5
Match-LSTM (Wang and Jiang, 2016)	92.0	86.1
Decomposable attention model (Parikh et al., 2016)	90.5	86.8
BiMPM (Zhiguo Wang, 2017)	90.9	87.5
NTI-SLSTM-LSTM (Munkhdalai and Yu, 2017)	88.5	87.3
Re-read LSTM (Sha et al., 2016)	90.7	87.5
DIIN (Gong et al., 2017)	91.2	88.0
ESIM (Chen et al., 2017a)	92.6	88.0
CIN	**93.2**	**88.0**
ESIM (Chen et al., 2017a) (Ensemble)	93.5	88.6
BiMPM (Zhiguo Wang, 2017) (Ensemble)	93.2	88.8
DIIN (Gong et al., 2017) (Ensemble)	92.3	88.9
CIN (Ensemble)	**94.3**	**89.1**

Table 2: Performance on SNLI dataset.

SNLI The SNLI corpus Bowman et al. (2015) consists of 570,152 sentence pairs. Each sentence pair is labeled as one of entailment, contradiction and neutral relation.

MultiNLI Orgnized the same as SNLI, MultiNLI corpus Williams et al. (2017) is another dataset for NLI, it contains 433,000 sentence pairs. Like SNLI, each pair is labeled with one of entailment, contradiction and neutral label. Difference between MultiNLI and SNLI is that, MultiNLI have in-domain test set and development set as well as an out-of-domain test and development set.

Quora The Quora Question pair dataset have over 400k question pairs, each question pair is assigned with a binary label to indicate if the pair are paraphrase to each other. We evaluate our model on the data which was previously partitioned by Zhiguo Wang (2017)

5.2 Overall Results

We use the accuracy to evaluate the performance of our convolutional interaction network (CIN) and other models on SNLI, MultiNLI and Quora.

SNLI Table 2 shows the results of different models on the train set and test set of SNLI. The first row gives a baseline model with handcrafted features presented by Bowman et al. (2015). All the other models are attention-based neural networks. Wang and Jiang (2016) exploits the long short-term memory (LSTM) for NLI. Parikh et al. (2016) uses attention to decompose the problem into subproblems that can be solved separately. Chen et al. (2017a) incorporates the chain LSTM and tree LSTM jointly. Zhiguo Wang (2017) proposes a bilateral multi-perspective matching for NLI.

In Table 2, the second block gives the single models. As we can see, our proposed model CIN achieves 88.0% in accuracy on SNLI test set. Compared to the previous work, CIN obtains competitive performance.

To further improve the performance of NLI systems, researchers have built ensemble models. Ensemble systems obtained the best performance on SNLI. Our ensemble model obtains 89.1% in accuracy and outperforms the current state-of-the-art model.

Overall, single model of CIN performs competitively well and outperforms the previous models on ensemble scenarios for the natural language inference task.

MultiNLI Table 3 shows the performance of different models on MultiNLI. The original aim of this dataset is to evaluate the quality of sentence representations. Recently this dataset is also used to evaluate the interaction model involving attention mechanism.

The first line of Table 3 gives a baseline model without interaction. The second block of Table 3 gives the attention-based models. The proposed

Models	Match	Mismatch
BiLSTM (Williams et al., 2017)	67.0	67.6
InnerAtt (Balazs et al., 2017)	72.1	72.1
ESIM (Chen et al., 2017a)	72.3	72.1
Gated-Att BiLSTM (Chen et al., 2017b)	73.2	73.6
ESIM (Chen et al., 2017a)	76.3	75.8
CIN	**77.0**	**77.6**

Table 3: Performance on MultiNLI test set.

Models	Test
Siamese-CNN	79.60
Multi-Perspective CNN	81.38
Siamese-LSTM	82.58
Multi-Perspective-LSTM	83.21
L.D.C	85.55
BiMPM (Zhiguo Wang, 2017)	88.17
CIN	**88.62**

Table 4: Performance on Quora question pair dataset.

model, CIN, achieves the accuracies of 77.0% and 77.6% on the *match* and *mismatch* test sets respectively. The results show that our model outperforms the other models.

Quora Table 4 shows the performance of different models on the Quora test set. The baselines on Table 4 are all implemented in Zhiguo Wang (2017). The Siamese-CNN model and Siamese-LSTM model encode sentences with CNN and LSTM respectively, and then predict the relationship between them based on the cosine similarity. Multi-Perspective-CNN and Multi-Perspective-LSTM are transformed from Siamese-CNN and Siamese-LSTM respectively by replacing the cosine similarity calculation layer with their multi-perspective cosine matching function. The L.D.C is a general compare-aggregate framework that performs word-level matching followed by a aggregation of convolution neural networks. As we can see, our model outperforms the baselines and achieve 88.62% in the test sets of Quora corpus.

Models	Dev	Test
CIN	**88.6**	**88.0**
Remove whole interaction	85.6	85.1
Remove intra-attention	88.1	87.7

Table 5: Ablation experiment on SNLI dataset.

Premise
(1) A girl playing a violin along with a group of people
(2) A girl playing a violin along with a group of people

Hypothesis
(1) A girl is playing an instrument .
(2) A girl is playing an instrument .

Table 6: Gradient visualization of premise and hypothesis. (1) Gradient scale of X, Y on encoding layer. (2) Gradient scale of $X^{(c)}$, $Y^{(c)}$ on first CIN layer. Darker color corresponds to a higher scale of gradient, and implies a higher contribution to the final prediction.

5.3 Model Ablation

To better understand the performance of our model, we analyze the effect of each key component of the proposed model. As illustrated in Table 5, the first row is the full CIN model. By dropping convolutional interaction layers, the performance decreases to 85.1% on the test set, which indicate the interaction information is crucial for NLI. By just dropping intra-attention layer, the performance decreases to 87.7% on the test set. According to the results, all of the components positively contribute to the final performance.

5.4 Case Study

To give an intuitive understanding of how our model works, we give an analysis on the following case from the test set.

Premise: A girl playing a violin along with a group of people.
Hypothesis: A girl is playing an instrument.
Label: Entailment.

The visualization results are produced from model with two stacked CINs. X, Y is the hidden states at encoding layer, and $X^{(c)}$, $Y^{(c)}$ is the hidden states at first CIN layer. For a hidden state

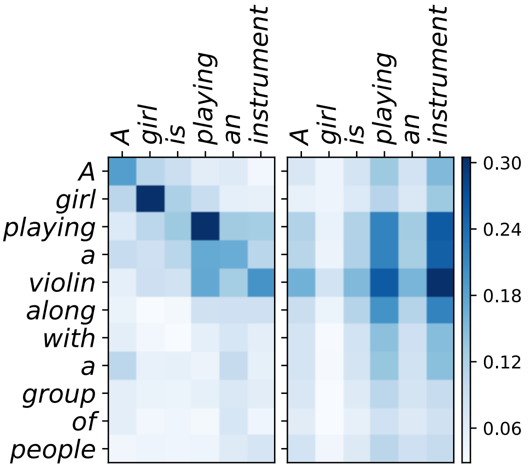

(a) (b)

Figure 3: A visualization of word to word correlation. Darker color correspond to a higher correlation. (a) Correlation of $X^{\mathrm{T}}Y$ at encoder layer. (b) Correlation of ${X^{(c)}}^{\mathrm{T}}Y^{(c)}$ at first CIN layer.

$\mathbf{x}_i$ of word x_i, we can calculate its gradient scale $||\frac{\partial \mathcal{J}}{\partial \mathbf{x}_i}||^2$ to show its contribution to final prediction.

Table 6 shows the gradient scales of hidden states of each word in the encoding layer and the first CIN layer. As we can see, some phrases (like *playing a violin* and *playing an instrument*) instead of isolated words (like *violin* and *instrument*) become more focused after a CIN layer. It implies CIN could capture some higher level patterns.

Figure 3 gives a visualization of correlations of hidden states of two sentences. (a) shows the correlations after the encoding layer, the same words are most correlated. This is because embedding layer and encoding layer are shared between premise and hypothesis. (b) shows the correlations after the first CIN layer, the correlation exists between phrases $\{$*playing a violin* vs. *playing an instrument*$\}$, instead of the same words. The interaction layer connects *playing* in Premise to Hypothesis *instrument*, and connects *playing* in Hypothesis to Premise *violin*. Thus, the correlation between *instrument* in Hypothesis and *violin* in Premise are boosted, as we know these are important to reasoning.

6 Related Work

There are mainly two threads of work related to ours.

One thread of work is using attention-based model for natural language inference (NLI). NLI has been widely investigated for many years. Benefiting from the development of deep learning and the availability of large-scale annotated datasets, deep neural models have achieved great success. Rocktäschel et al. (2015) firstly use LSTM with attention for text matching task. Wang and Jiang (2016) use word-by-word attention to exploit the word-level match. Parikh et al. (2016) propose a new framework to model the relationship between two sentences using interact-compare-aggregate architecture. Chen et al. (2017a) incorporates the chain LSTM and tree LSTM jointly. Zhiguo Wang (2017) use self-attention mechanism to capture contextual information from the whole sentence.

Unlike the above models, we use an alternative model to capture the complicate interaction information of two sentences.

Another thread of work is the idea of using one network to generate the parameters of another network. De Brabandere et al. (2016) proposed the dynamic filter network to implicitly learn a variety of filtering operations. Ha et al. (2016a) proposed the model hypernetwork, which uses a small network to generate the weights for a larger network.

Unlike these models, our dynamical filter is employed for interaction. Therefore, a filter generation function is proposed to capture the related intra and inter dependent information of a sentence pair.

7 Conclusion

In this paper, we propose an alternative interaction model, Convolutional Interaction Network (CIN), for natural language inference. CIN utilizes the dynamic convolutional filters to model the interaction between two sentences. Specifically, each sentence is convolved by dynamical filters generated based on another sentence. CIN is more general and flexible since the filters may have various numbers and sizes, thereby capturing more complicated interaction patterns. Experiments on three very large datasets demonstrate the efficacy of our proposed model.

In future work, we hope to improve the extensibility of CIN and apply it to other NLP tasks, such as machine comprehension.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. We would like to thank the anonymous reviewers for their valuable comments. The research work is supported by Shanghai Municipal Science and Technology Commission (No. 17JC1404100 and 16JC1420401), and National Natural Science Foundation of China (No. 61672162 and 61751201), Fundamental Theory and Cutting-Edge Technology Research Program of Institute of Information Engineering, CAS (Grant No. Y7Z0281102).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jorge A Balazs, Edison Marrese-Taylor, Pablo Loyola, and Yutaka Matsuo. 2017. Refining raw sentence representations for textual entailment recognition via attention. *arXiv preprint arXiv:1707.03103*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017a. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Recurrent neural network-based sentence encoder with gated attention for natural language inference. *arXiv preprint arXiv:1708.01353*.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. 2016. Dynamic filter networks. In *Neural Information Processing Systems (NIPS)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1243–1252.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

David Ha, Andrew Dai, and Quoc V Le. 2016a. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

David Ha, Andrew Dai, and Quoc V Le. 2016b. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. 2016. Dynamic filter networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 667–675.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural tree indexers for text understanding. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 1, page 11. NIH Public Access.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read lstm unit for textual entailment recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2870–2879.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI*.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *TACL*, 4:259–272.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.

Radu Florian Zhiguo Wang, Wael Hamza. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4144–4150.

Lessons from Natural Language Inference in the Clinical Domain

Alexey Romanov
Department of Computer Science
University of Massachusetts Lowell*
Lowell, MA 01854
aromanov@cs.uml.edu

Chaitanya Shivade
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
cshivade@us.ibm.com

Abstract

State of the art models using deep neural networks have become very good in learning an accurate mapping from inputs to outputs. However, they still lack generalization capabilities in conditions that differ from the ones encountered during training. This is even more challenging in specialized, and knowledge intensive domains, where training data is limited. To address this gap, we introduce MedNLI[1] – a dataset annotated by doctors, performing a natural language inference task (NLI), grounded in the medical history of patients. We present strategies to: 1) leverage transfer learning using datasets from the open domain, (e.g. SNLI) and 2) incorporate domain knowledge from external data and lexical sources (e.g. medical terminologies). Our results demonstrate performance gains using both strategies.

1 Introduction

Natural language inference (NLI) is the task of determining whether a given *hypothesis* can be inferred from a given *premise*. This task, formerly known as recognizing textual entailment (RTE) (Dagan et al., 2006) has long been a popular task among researchers. Moreover, contribution of datasets from past shared tasks (Dagan et al., 2009), and recent research (Bowman et al., 2015; Williams et al., 2018) have pushed the boundaries for this seemingly simple, but challenging problem.

The Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) is a large, high quality dataset and serves as a benchmark to evaluate NLI systems. However, it is restricted to a single text genre (Flickr image captions) and mostly consists of short and simple sentences. The MultiNLI corpus (Williams et al., 2018) which introduced NLI corpora from multiple genres (e.g. fiction, travel) was a welcome step towards addressing these limitations. MultiNLI offers diversity in linguistic phenomena, which makes it more challenging.

Following these efforts, we explore the problem of NLI in the clinical domain. Language inference in specialized domains such as medicine is extremely complex and remains unexplored by the machine learning community. Moreover, since this domain has a distinct sublanguage (Friedman et al., 2002), clinical text also presents unique challenges (abbreviations, inconsistent punctuation, misspellings, etc.) that differentiate it from open-domain data (Meystre et al., 2008).

In this paper, we address these gaps and make the following contributions:

- Introduce MedNLI - a new, publicly available, expert annotated dataset for NLI in the clinical domain.
- A systematic comparison of several state-of-the-art open domain models on MedNLI.
- A study of transfer learning techniques from the open domain to the clinical domain.
- Techniques for incorporating domain-specific knowledge from knowledge bases (KB) and domain specific data into neural networks.

2 The MedNLI dataset

Let us recall the procedure followed for creating the SNLI dataset: annotators were presented with captions for a Flickr photo (the *premise*) without the photos themselves. They were asked to write three sentences (*hypotheses*): 1) A clearly true description of the photo, 2) A clearly false description, and 3) A description that might be true or false. This procedure produces three training pairs

---

[*] Work done during an internship at IBM Research

[1] https://jgc128.github.io/mednli/

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1586–1596
Brussels, Belgium, October 31 - November 4, 2018.

#	Premise	Hypothesis	Label
1	ALT , AST , and lactate were elevated as noted above	patient has abnormal lfts	entailment
2	Chest x-ray showed mild congestive heart failure	The patient complains of cough	neutral
3	During hospitalization , patient became progressively more dyspnic requiring BiPAP and then a NRB	The patient is on room air	contradiction
4	She was not able to speak , but appeared to comprehend well	Patient had aphasia	entailment
5	T1DM : x 7yrs , h/o DKA x 6 attributed to poor medication compliance , last A1c [** 3-23 **] : 13.3 % 2	The patient maintains strict glucose control	contradiction
6	Had an ultimately negative esophagogastroduodenoscopy and colonoscopy	Patient has no pain	neutral
7	Aorta is mildly tortuous and calcified .	the aorta is normal	contradiction

Table 1: Examples from the development set of MedNLI

of sentences for each premise with three different labels: entailment, contradiction, and neutral, respectively. In order to produce a comparable dataset, we used the same approach adjusted for the clinical domain.

2.1 Premise sampling and hypothesis generation

As the source of *premise* sentences, we used the MIMIC-III v1.3 (Johnson et al., 2016) database. With de-identified records of 38,597 patients, it is the largest repository of publicly available clinical data. Along with medications, lab values, vital signs, etc. MIMIC-III contains 2,078,705 clinical notes written by healthcare professionals in English. The *hypothesis* sentences were generated by clinicians.

Clinical notes are typically organized into sections such as `Chief Complaint`, `Past Medical History`, `Physical Exam`, `Impression`, etc. These sections can be easily identified since the formatting for associated section headers often resembles capital letters, followed by a colon. The clinicians in our team suggested `Past Medical History` to be the most informative section of a clinical note, from which critical inferences can be drawn about the patient.

Therefore, we segmented these notes into sections using a simple rule based program capturing the formatting of these section headers. We extracted the `Past Medical History` section and used a sentence splitter trained on biomedical articles (Lingpipe, 2008) to get a pool of candidate *premises*. We then randomly sampled a subset from these candidates and presented them to the clinicians for annotation. Figure 1 shows the exact prompt shown to the clinicians for the annotation task. SNLI annotations are grounded since they are associated with captions of the same image. We seek to achieve the same goal by grounding the annotations against the medical history of the same patient.

> You will be shown a sentence from the `Past Medical History` section of a de-identified clinical note. Using only this sentence, your knowledge about the field of medicine, and common sense:
>
> - Write one alternate sentence that is **definitely** a **true** description of the patient. Example, for the sentence “Patient has type II diabetes” you could write “Patient suffers from a chronic condition“
> - Write one alternate sentence that **might be** a **true** description of the patient. Example, for the sentence “Patient has type II diabetes” you could write “Patient has hypertension”
> - Write one sentence that is **definitely** a **false** description of the patient. Example, for the sentence “Patient has type II diabetes” you could write “The patient’s insulin levels are normal without any medications.”

Figure 1: Annotation prompt shown to clinicians

As discussed earlier, examples shown in Table 1 depict unique challenges that involve reasoning over domain-specific knowledge. For instance, the first three examples require the knowledge about clinical terminology. The fourth example requires awareness of medications and the last example elicits knowledge about radiology images. We make the MedNLI dataset available[2] through the

---

[2] https://jgc128.github.io/mednli/

MIMIC-III derived data repository. Thus, any individual certified to access MIMIC-III can also access MedNLI.

2.2 Annotation collection

Conclusions in the clinical domain are known to be context dependent and a source of multiple uncertainties (Han et al., 2011). We had to ensure that such subjective interpretations do not result in annotation conflicts affecting the quality of the dataset. To ensure agreement, we worked with clinicians and generated annotation guidelines for a pilot study. Two board certified radiologists worked on the annotation task, and were presented with the 100 unique premises each.

Some premises, often marred by de-identification artifacts, did not contain any information from which useful inferences could be drawn, e.g. `This was at the end of [**Month (only) 1702**] of this year.` Such sentences were deemed as invalid for the task and discarded based on clinician judgment. The MIMIC-III dataset contains many de-identification artifacts associated with dates and names (persons and places) which also makes MedNLI more challenging.

After discarding 16 premises, the result of hypothesis generation was a set of 552 pairs. To calculate agreement, we presented pairs generated by one clinician, and sought annotations from the other clinician, determining if the inference was "Definitely true", "Maybe true", or "Definitely false" (Bowman et al., 2015). Comparison of these annotations resulted in a Cohen's kappa of $\kappa = 0.78$. While this is substantial if not perfect agreement by itself (McHugh, 2012), it is particularly good given the challenging nature of NLI and the complexity of the domain.[3]

On reviewing the annotations, we found that labeling differences between "Definitely true" and "Maybe true" were the major source of disagreement. This was primarily because one clinician would think of a scenario that is generally true, while the other would think of assumptions (e.g. patient might be lying, or patient might be pregnant) when it would not.

A discussion with clinicians concluded that the annotation guideline was clear and any person with a formal background of medicine should be able to complete the task successfully. To generate the final dataset, we recruited two additional clinicians, both board certified medical students pursuing their residency programs. Unlike SNLI, we did not collect multiple annotations per sentence pair because of the time and funding constraints.

2.3 Dataset statistics

Together, the four clinicians worked on a total of 4,683 premises over a period of six weeks. The resulting dataset consists of 14,049 unique sentence pairs. Following Bowman et al. (2015), we split the dataset into training, development, and testing subsets and ensured that no premise was overlapping between the three subsets. Table 2 presents key statistics of MedNLI.

Dataset size	
Training pairs	11232
Development pairs	1395
Test pairs	1422
Average sentence length in tokens	
Premise	20.0
Hypothesis	5.8
Maximum sentence length in tokens	
Premise	202
Hypothesis	20

Table 2: Key statistics of the dataset

3 Models

To establish a baseline performance on MedNLI, we experimented with a feature-based system. To further explore the performance of modern neural networks-based systems, we experimented several models of various degrees of complexity: Bag of Words (BOW), InferSent (Conneau et al., 2017) and ESIM (Chen et al., 2017). Note that our goal here is not to outperform existing models, but to explore the relative gain of the proposed methods, and compare them to a baseline. We used the same set of hyperparameters in all models to ensure that any difference in performance is exclusively due to the algorithms.

Feature-based system We used a gradient boosting classifier incorporating a variety of hand crafted features. Apart from standard NLP features, we also infused clinical knowledge from the Unified Medical Language System (UMLS) (Bodenreider, 2004). Each terminology in the UMLS can be viewed as a graph where nodes represent medical concepts, and edges represent relations

---

[3] Rajpurkar et al. (2017) report F1 < 0.45 for four radiologists when compared among themselves

between them. These are canonical relationships found in ontologies such as *IS A* and *SYNONYMY*. For instance, *diabetes IS A disorder of the endocrine system*. The domain specific features we added to the model represent similarity between UMLS concepts from the premise and the hypothesis, based how close they appear in the UMLS graph (Pedersen et al., 2007). Following (Shivade et al., 2015; Pedersen et al., 2007) we used the SNOMED-CT terminology in our experiments.

The groups below summarize the feature sets used in our model (35 features in total):

1. BLEU score
2. Number of tokens (e.g. min, max, difference)
3. Negations (e.g. keywords such as *no, do not*)
4. TF-IDF similarity (e.g. cosine, euclidean)
5. Edit distances (e.g. Levenshtein)
6. Embedding similarity (e.g. cosine, euclidean)
7. UMLS similarity features (e.g. shortest path distance between UMLS concepts)

Bag of words We use a bag-of-words (BOW) model as a simple baseline for the NLI task: the *Sum of words* model by Bowman et al. (2015) with a small modification. While Bowman et al. (2015) use $tanh$ as the activation function in the model, we use ReLU, since it trained faster and achieved better results (Glorot et al., 2011). In order to represent an input sentence as a single vector, this architecture simply sums up the vectors of individual tokens. The premise and hypothesis vectors are then concatenated and passed through a multi-layer neural network. Recent work shows that even this straightforward approach encodes a non-trivial amount of information about the sentence (Adi et al., 2017).

InferSent InferSent (Conneau et al., 2017) is a model for sentence representation that demonstrated close to state-of-the-art performance across a number of tasks in NLP (including NLI) and computer vision. The main differences from the BOW model are as follows:

- A bidirectional LSTM encoder of input sentences and a max-pooling operation over timesteps are used to get a vector for the premise (p) and for the hypothesis (h);
- A more complex scheme of interaction between the vectors p and h to get a single vector z that contains all the information needed to produce a decision about the relationship between the input sentences: $z = [p, h, |p - h|, p * h]$.

ESIM The ESIM model, developed by Chen et al. (2017), is shown in Figure 2. It is a fairly complex model that makes use of two bidirectional LSTM networks. The basic idea of ESIM is as follows:

- The first LSTM produces a sequence of hidden states.
- Pairwise attention matrix e is computed between all tokens in the premise and the hypothesis to produce new sequences of "attended" hidden states, which are then fed into the second LSTM.
- Max and average pooling are performed over the output of the LSTMs.
- The output of the pooling operations is combined in a way similar to the InferSent model.

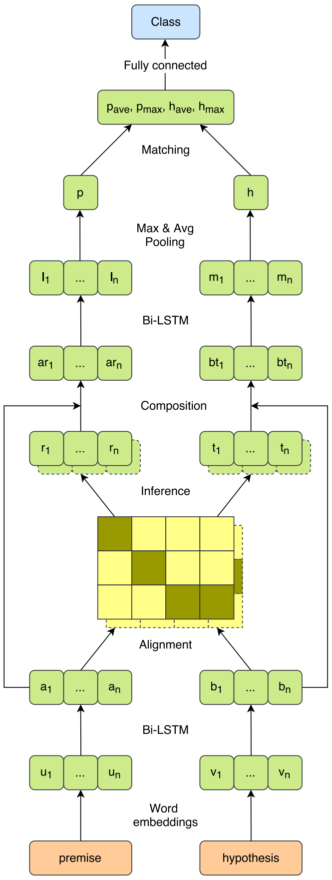

Figure 2: ESIM model. Dashed blocks illustrate the knowledge-directed attention matrix and the corresponding vectors (see Section 4.2.2 for details).

The three aforementioned models exemplify the architectures that are, perhaps, the most widely used for NLI task, spanning from simple bag-of-words approaches to complicated models with Bi-LSTM and inter-sentence attention. We additionally experimented with a plain Bi-LSTM model as well as GRU (Cho et al., 2014), but since their performance was not remarkable (in the same range as BOW) we do not report it here.

4 Transfer learning

Given the existence of larger general-domain NLI datasets such as SNLI and MultiNLI, it stands to reason to try to leverage them to improve the performance in the clinical domain. Transfer learning has been shown to improve performance on variety of tasks such as: machine translation on low-resource languages (Zoph et al., 2016) and also some tasks from the bio-medical domain in particular (Sahu and Anand, 2017; Lee et al., 2018). To see if a corresponding boost would be possible for the NLI task, we investigated three common transfer learning techniques on the MedNLI dataset using SNLI and five different genres from MultiNLI.

Direct transfer is the simplest method of transfer learning. After training a model on a large *source domain* dataset, the model is directly tested on the *target domain* dataset. If the source and the target domains are similar to some extent, one can achieve a reasonable accuracy by simply applying a model pre-trained on the source domain to the target domain. In our case the source domain is general domain in SNLI and the various genres in MultiNLI, and the target domain is clinical.

Sequential transfer is the most widely used technique. After pre-training the model on a large source domain, the model is further fine-tuned using the smaller training data of the target domain. The assumption is that while the model would learn domain-specific features, it would also learn some domain-independent features that will be useful for the target domain. Furthermore, the fine-tuning process would affect the learned features from the source domain and make them more suitable for the target domain.

Multi-target transfer is a more complex method involving separation of the model into three components (or layers):

- *The shared component* is trained on both the source and target domains;
- *The source domain component* is trained only during the pre-training phase and does not participate in the prediction of the target domain;
- *The target domain component* is trained during the fine-tuning stage and it produces the predictions together with the shared component.

The motivation for multi-target transfer is that performance should be improved by splitting deeper layers of the model into domain-specific parts and having a shared block early in the network, where it presumably learns domain-independent features.

4.1 Word embeddings

Another way to improve the accuracy on the target domain is to use domain-specific word embeddings instead of, or, in addition to, open-domain ones. For example, Stanovsky et al. (2017) achieved state of the art results on the task of recognizing Adverse Drug Reaction using graph-based embeddings trained on the “Drugs” and “Diseases” categories from DBpedia (Lehmann et al., 2015), as well as embeddings trained on web-pages categorized as “medical domain”.

We experimented with the following publicly available general-domain word embeddings:

- **GloVe$_{\text{[CC]}}$**: GloVe embeddings (Pennington et al., 2014), trained on Common Crawl[4].
- **fastText$_{\text{[Wiki]}}$**: fastText embeddings (Bojanowski et al., 2017), trained on Wikipedia.
- **fastText$_{\text{[CC]}}$**: fastText embeddings, trained on Common Crawl.

Furthermore, we trained fastText embeddings on the following domain-specific corpora:

- **fastText$_{\text{[BioASQ]}}$**:A collection of PubMed abstracts from the BioASQ challenge data (Tsatsaronis et al., 2015). This data includes abstracts from 12,834,585 scientific articles from the biomedical domain.
- **fastText$_{\text{[MIMIC-III]}}$**: Clinical notes for patients from the MIMIC-III database (Johnson et al., 2016): 2,078,705 notes with 320 tokens in each on average.

Finally, we experimented with initializing word embeddings with pre-trained vectors from general

---

[4] `http://commoncrawl.org/`

domain and further training on a domain-specific corpus:

- $\text{GloVe}_{[\text{CC}]} \rightarrow \text{fastText}_{[\text{BioASQ}]}$: GloVe embeddings for initialization, and the BioASQ data for fine-tuning.
- $\text{GloVe}_{[\text{CC}]} \rightarrow \text{fastText}_{[\text{BioASQ}]} \rightarrow \text{fastText}_{[\text{MIMIC-III}]}$: GloVe embeddings for initialization, and two consequent fine-tuning using the BioASQ and MIMIC-III data.
- $\text{fastText}_{[\text{Wiki}]} \rightarrow \text{fastText}_{[\text{MIMIC-III}]}$: fastText Wikipedia embeddings for initialization, and the MIMIC-III data for fine-tuning.

Experiments using other approaches to word embeddings, such as `word2vec` (Mikolov et al., 2013) and CoVe (McCann et al., 2017) did not show any gains. All the above trained embeddings are available for download.

4.2 Knowledge integration

Since understanding medical texts requires domain-specific knowledge, we experimented with different ways of incorporating such knowledge into the systems. First, we can modify the input to the system so it carries a portion of clinical information. Second, we can modify the model itself, integrating domain knowledge directly into it.

The UMLS is the largest, publicly available, and regularly updated database of medical terminologies, concepts, and relationships between them. It can be viewed as a graph where clinical concepts are nodes, connected by edges representing relations, such as synonymy, parent-child, etc. Following past work, we restricted to the SNOMED-CT terminology in UMLS and experimented with two techniques for incorporating knowledge: retrofitting and attention.

4.2.1 Retrofitting

Retrofitting (Faruqui et al., 2015) modifies pre-trained word embeddings based on an ontology. The basic idea is to try to bring the representations of the concepts that are connected in the ontology closer to one another in vector space. The authors showed that retrofitting using WordNet (Fellbaum, 1998) synsets improves accuracy on several word-level tasks, as well as sentiment analysis.

4.2.2 Knowledge-directed attention

Attention proved to be a useful technique for many NLP tasks, starting from machine translation (Bahdanau et al., 2015) to parsing (Vinyals et al., 2015) and NLI itself (Parikh et al., 2016; Rocktäschel et al., 2016). In most models (including the ESIM model that we use in our experiments) attention is learned in an end-to-end fashion. However, if we have knowledge about relationships between concepts, we could leverage it to explicitly tell the model to attend to specific concepts during the processing of the input sentence.

For example, there is an edge in SNOMED-CT from the concept *Lung consolidation* to *Pneumonia*. Using this information, during the processing of a sentence pair

- **Premise** The patient has *pneumonia*.
- **Hypothesis** The patient has a *lung* disease.

the model could attend to the token *lung* while processing *pneumonia*.

We propose to integrate this knowledge in a way similar to how attention is used in the ESIM model. Specifically, we calculate the attention matrix $e \in \mathbb{R}^{n \times m}$ between all pairs of tokens a_i and b_j in the inputs sentences, where n is the length of the hypothesis and m is the length of the premise. The value in each cell reflects the length of the shortest path l_{ij} between the corresponding concepts of the premise and the hypothesis in SNOMED-CT (the shorter is the path, the higher is the value).

This process could be informally described as follows: each token $\tilde{a_i}$ of the premise is a weighted sum of relevant tokens b_j of the hypothesis, according to the medical ontology, and vice versa. This enables the medical domain knowledge to be integrated directly into the system.

We used the original tokens a_i as well as the attended $\tilde{a_i}$ inside the model for both InferSent and ESIM. For InferSent, we simply concatenate them across the time dimension:

$$\hat{a} = [a_1, a_2, \ldots, a_n, \tilde{a_1}, \tilde{a_2}, \ldots, \tilde{a_n}]$$

where n is the length of the inputs sequence. For the ESIM model, we concatenate a_i and $\tilde{a_i}$ before passing them to the composition layer (see Figure 2 and Section 3.3 in the original paper (Chen et al., 2017)). This enables the model to learn the relative importance of both the token and the knowledge directed attention.

Set	Features	BOW	InferSent	ESIM
Dev	51.9	71.9	**76.0**	74.4
Test	51.9	70.2	**73.5**	73.1

Table 3: Baseline accuracy on the development and the test set of MedNLI for different models.

5 Results and discussion

We implemented all models using PyTorch[5] and trained them with the Adam optimizer (Kingma and Ba, 2015) until the validation loss showed no improvement for 5 epochs. The epoch with the lowest loss on the validation set was selected for testing. We used the GloVe word embeddings (Pennington et al., 2014) in all experiments, except for subsection 5.3. In all experiments we report the average result of 6 different runs, with the same hyperparameters and different random seeds. Medical concepts in SNOMED-CT were identified in the premise and hypothesis sentences using Metamap (Aronson and Lang, 2010). The code for all experiments is publicly available.[6]

5.1 Baselines

Table 3 shows the baseline results: the performance of a model when trained and tested on the MedNLI dataset. The feature-based system performed the worst. As for neural networks-based systems, the BOW model showed the lowest performance on the both development and test sets. The InferSent model, in contrast, achieved the highest accuracy, despite ESIM outperforming it on SNLI. This could be attributed to the fact that ESIM has twice as many parameters as InferSent, and so InferSent overfits less to the smaller MedNLI dataset.

5.2 Transfer learning

As expected, Table 4 shows that direct transfer is worse than the baseline but is still better than a random baseline of 33.3%. Sequential and multi-target transfer learning, in contrast, yields a considerable gain for all the models. The maximum gain is 2.4%, 0.9%, and 0.3% for the BOW, InferSent, and ESIM models correspondingly.

Second, note that the biggest SNLI domain gave the most boost in only two out of six cases, implying that the size of the domain should not be the most important factor in choosing the source domain for transfer learning. The best accuracy for all the models was obtained with the "slate" domain from MultiNLI corpus with sequential transfer (note, however, that the accuracy of ESIM is actually lower than the baseline accuracy). This is consistent with observations of Williams et al. (2018). Finally, although some domains are better for particular transfer learning methods with particular models, there is no single combination that works for all cases.

5.3 Word embeddings

Table 5 shows that simply using of the embeddings trained on the MIMIC-III notes significantly increases the accuracy for all the models. Furthermore, the InferSent models achieves a 3.1% boost with the fastText Wikipedia embeddings, fine-tuned on the MIMIC-III data. Note that the results $\text{fastText}_{[\text{Wiki}]}$ are worse than the baseline $\text{GloVe}_{[\text{CC}]}$ for all models, which could be due to the source corpus size. However, the results on BioASQ are worse than on MIMIC-III, despite the significantly larger corpus of the BioASQ embeddings. Overall, our experiments show the benefit of domain-specific rather than general-domain word embeddings.

5.4 Knowledge integration

5.4.1 Retrofitting

Table 6 shows that retrofitting only hurts the performance. This is in contrast with the results of the original study, where retrofitting was beneficial not only for word-level tasks but also for tasks such as sentiment analysis (Faruqui et al., 2015). We hypothesize that although WordNet and UMLS are structurally similar, significant differences in the content (Burgun and Bodenreider, 2001) might be the reason for these results. Retrofitting should be more useful when it is used on a WordNet-like database where the main relation is synonymy, and tested on tasks such as word similarity tests or sentiment analysis. The UMLS semantic network is more complex and contains relations that may not be suitable for retrofitting.

Moreover, retrofitting works only on directly related concepts in a knowledge graph (although it might affect, to some extent, indirectly related concepts by transitivity). However, as Figure 3 shows, UMLS contains few training pairs that have such concepts (namely, pairs with a path of

---

[5] `https://pytorch.org/`

[6] `https://jgc128.github.io/mednli/`

Source domain	Direct transfer			Sequential transfer			Multi-target transfer		
	BOW	InferSent	ESIM	BOW	InferSent	ESIM	BOW	InferSent	ESIM
snli	-21.8	**-24.2**	-22.8	1.8	-1.8	-2.5	**2.4**	-2.5	-0.7
fiction	**-21.6**	-25.6	**-21.4**	1.3	0.4	-0.5	1.4	0.1	**0.3**
government	-23.8	-27.2	-26.2	1.0	0.8	-0.7	1.3	0.2	0.2
slate	-23.2	-25.7	-21.6	**1.9**	**0.9**	**-0.2**	1.1	**0.6**	-0.1
telephone	-25.7	-27.3	-25.6	1.7	-0.2	-1.1	1.2	0.4	-0.1
travel	-25.4	-29.1	-23.5	1.6	0.0	-0.7	0.2	-0.3	0.1

Table 4: Absolute gain in accuracy with respect to the baseline (see Table 3) on the MedNLI test set for different transfer learning modes. Bold indicates the best source domain for each model and transfer.

Embeddings	BOW	InferSent	ESIM
fastText$_{[\text{Wiki}]}$	-3.5	-3.5	-4.4
fastText$_{[\text{CC}]}$ (600B)	-0.6	1.3	-0.3
fastText$_{[\text{BioASQ}]}$ (2.3B)	0.5	0.6	0.2
fastText$_{[\text{MIMIC-III}]}$ (0.8B)	**1.1**	2.3	1.2
GloVe$_{[\text{CC}]}$ → fastText$_{[\text{BioASQ}]}$	0.2	0.7	1.4
GloVe$_{[\text{CC}]}$ → fastText$_{[\text{BioASQ}]}$ → fastText$_{[\text{MIMIC-III}]}$	0.9	2.7	**1.8**
fastText$_{[\text{Wiki}]}$ → fastText$_{[\text{MIMIC-III}]}$	0.1	**3.1**	1.7

Table 5: Absolute gain in accuracy with respect to the baseline (GloVe$_{[\text{CC}]}$) for different word embeddings (the number in parentheses reflects the number of tokens in the corresponding training corpora).

length 1). In contrast, the lengths of the shortest path in SNLI using WordNet fall close to 1. This suggests that the medical inferences represented in MedNLI requires more complex reasoning, typically involving multiple steps.

As a sanity check, we applied retrofitting to the GloVe embeddings and tested the InferSent model on the "fiction" domain from the MultiNLI corpus. We used the code and lexicons provided by Faruqui et al. (2015) and confirmed that retrofitting hurts the performance in that case as well.

BOW	InferSent	ESIM
-1.7	-2.0	-2.7

Table 6: Absolute gain in accuracy using retrofitting for MedNLI.

5.4.2 Knowledge-directed attention

To evaluate the potential of knowledge-directed attention, let us consider its effect on a baseline embedding (GloVe$_{[\text{CC}]}$) and a fastText embedding trained on MIMIC-III (fastText$_{[\text{MIMIC-III}]}$) that showed good performance in section 5.3.

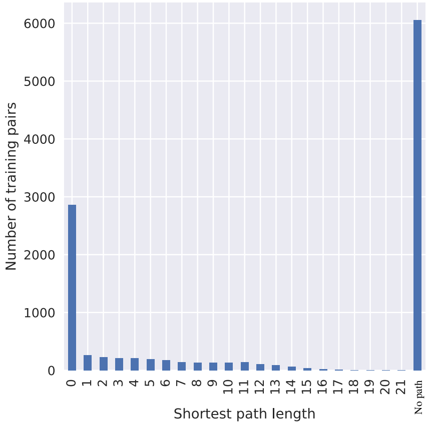

Figure 3: Lengths of the shortest paths between concepts in the premise and the hypothesis. 0 indicates that they contain the same concept.

Knowledge-directed attention showed positive effect with the InferSent model on GloVe$_{[\text{CC}]}$ (0.3 gain), and was not detrimental to ESIM. However, in case of the fastText$_{[\text{MIMIC-III}]}$ embeddings knowledge-directed attention was beneficial to both models, as shown in Table 7. Note that while retrofitting can use only direct relations during the training process, our method incorporates information about relationships of any length, which is a necessity (as evident from Figure 3).

Embedding	InferSent	ESIM
GloVe$_{[\text{CC}]}$	0.3	0.0
fastText$_{[\text{MIMIC-III}]}$	0.2	0.3

Table 7: Absolute gain in accuracy using knowledge-directed attention.

6 Error analysis

The neutral class is the hardest to recognize for all models. Majority errors stem from confusion between entailment and the neutral class. Use of domain-specific embeddings trained on MIMIC-

Category	**Premise**	**Hypothesis**	**Predicted**	**Expected**
Numerical Reasoning	WBC 12 , Hct 41 .	WBC slightly elevated	contradiction	entailment
World Knowlegde	No known sick contacts	No recent travel	entailment	neutral
Abbreviations	No CP or fevers.	Patient has no angina	neutral	entailment
Medical Knowledge	EKG showed T-wave depression in V3-5, with no prior EKG for comparison.	Patient has a normal EKG	neutral	contradiction
Negations	Head CT was negative for bleed.	The patient has intracranial hemorrhage	neutral	contradiction

Table 8: Representative errors made by different models

III result in gains which are equally distributed across all three classes. Interestingly, gains from knowledge-directed attention stem mostly (60%) from the neutral class. Moreover, 87% of these neutral predictions were predicted as entailment before adding the knowledge directed attention.

We categorized the errors made by all the models in four broad categories. Table 8 outlines representative errors made by most models in these categories. Numerical reasoning such as *abnormal lab value* $\rightarrow$ *disease* or *abnormal vital sign* $\rightarrow$ *finding* are very hard for a model to learn unless it has seen multiple instances of the same numerical value.[7] The first step is to learn what values are *abnormal* and the next is to actually perform the inference. Many inferences require world knowledge that could be deemed close to open domain NLI. While these are very subtle, some are quite domain specific (e.g. *emergency admission* $\nrightarrow$ *planned visit*). Abbreviations are ubiquitously found in clinical text. While some are standard and therefore frequent, clinicians tend to use non standard abbreviations making inference harder. Finally, many inferences are at the core of reasoning with clinical knowledge. While training on large datasets maybe a natural but impractical solution, this is an open research problem for researchers in the community.

7 Limitations

Unlike SNLI and MultiNLI, each example in the MedNLI dataset was single annotated. However, this was the best we could do in the limited time and resources available. Very recently Gururangan et al. (2018) discovered annotation artifacts in NLI datasets. Since we followed the exact same process, we found them to be present in MedNLI as well. The premise-oblivious text-classifier that achieves 67.0 F1 on SNLI, and 53.9 on Multi-NLI achieves 61.9 on MedNLI.

8 Conclusion

We have presented MedNLI, an expert annotated, public dataset for natural language inference in the clinical domain. To the best of our knowledge, MedNLI is the first dataset of its kind. Our experiments with several state-of-the-art models provide a strong baseline for this dataset. Our work compliments the current efforts in NLI by presenting thorough experiments for the specialized and knowledge intensive field of medicine. We also demonstrated that a simple use of domain-specific word embeddings provides a performance boost. Finally, we also presented a method for integrating domain ontologies into the training regime of models. We hope the released code and dataset with clear benchmarks help advance research in clinical NLP and the NLI task.

Acknowledgments

This work would not have been possible without Adam Coy, Andrew Colucci, Chanida Thammachart, and Hassan Ahmad – the clinicians in our team who helped us in creating the dataset. We are grateful to Vandana Mukherjee and Tanveer Syeda-Mahmood for supporting the project. We would also like to thank Anna Rumshisky and Anna Rogers for their help in this work. Most importantly, we would like to thank Leo Anthony Celi and Alistair Johnson from the MIMIC team for helping us in making MedNLI publicly available.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained anal-

---

[7] The symbol $\rightarrow$ represents entailment relationship

ysis of sentence embeddings using auxiliary prediction tasks. In *Proceedings of ICLR*.

Alan R Aronson and François-Michel Lang. 2010. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): Integrating biomedical terminology. *Nucleic Acids Research*, 32(suppl_1):D267–D270.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

Anita Burgun and Olivier Bodenreider. 2001. Comparing terms, concepts and semantic classes in WordNet and the Unified Medical Language System. In *Proceedings of the NAACL Workshop: WordNet and other lexical resources: Applications, extensions and customizations.*, pages 77–82.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Procedings of ACL*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of EMNLP*.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):i–xvii.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Carol Friedman, Pauline Kra, and Andrey Rzhetsky. 2002. Two biomedical sublanguages: a description based on the theories of Zellig Harris. *Journal of Biomedical Informatics*, 35(4):222–235.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 315–323.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. *Proceedings of NAACL*.

Paul KJ Han, William MP Klein, and Neeraj K Arora. 2011. Varieties of uncertainty in health care: a conceptual taxonomy. *Medical Decision Making*, 31(6):828–838.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Liwei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. 2018. Transfer learning for named-entity recognition with neural networks. *Proceedings of LREC*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Lingpipe. 2008. LingPipe 4.1.0.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of NIPS*.

Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282.

Stephane M. Meystre, Guergana K. Savova, Karin C. Kipper-Schuler, and John F. Hurdle. 2008. Extracting information from textual documents in the electronic health record: a review of recent research. *Yearbook of Medical Informatics*, 35:128–44.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of EMNLP*.

Ted Pedersen, Serguei VS Pakhomov, Siddharth Patwardhan, and Christopher G Chute. 2007. Measures of semantic similarity and relatedness in the biomedical domain. *Journal of Biomedical Informatics*, 40(3):288–299.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. 2017. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of ICLR*.

Sunil Kumar Sahu and Ashish Anand. 2017. What matters in a transferable neural network model for relation classification in the biomedical domain? *arXiv preprint arXiv:1708.03446*.

Chaitanya Shivade, Courtney Hebert, Marcelo Lopetegui, Marie-Catherine De Marneffe, Eric Fosler-Lussier, and Albert M Lai. 2015. Textual inference for eligibility criteria resolution in clinical trials. *Journal of Biomedical Informatics*, 58:S211–S218.

Gabriel Stanovsky, Daniel Gruhl, and Pablo Mendes. 2017. Recognizing mentions of adverse drug reaction in social media using knowledge-infused recurrent models. In *Proceedings of EACL*.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):138.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of EMNLP*.

Question Generation from SQL Queries Improves Neural Semantic Parsing

Daya Guo[1*], **Yibo Sun**[3*], **Duyu Tang**[2], **Nan Duan**[2], **Jian Yin**[1],
Hong Chi[2], **James Cao**[2], **Peng Chen**[2], **and Ming Zhou**[2]
[1] The School of Data and Computer Science, Sun Yat-sen University.
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, P.R.China
[2] Microsoft Research [3] Harbin Institute of Technology
`{guody5@mail2,issjyin@mail}.sysu.edu.cn`
`{dutang,nanduan,hongchi,jcao,peche,mingzhou}@microsoft.com`
`ybsun@ir.hit.edu.cn`

Abstract

We study how to learn a semantic parser of state-of-the-art accuracy with less supervised training data. We conduct our study on WikiSQL, the largest hand-annotated semantic parsing dataset to date. First, we demonstrate that question generation is an effective method that empowers us to learn a state-of-the-art neural network based semantic parser with thirty percent of the supervised training data. Second, we show that applying question generation to the full supervised training data further improves the state-of-the-art model. In addition, we observe that there is a logarithmic relationship between the accuracy of a semantic parser and the amount of training data.

1 Introduction

Semantic parsing aims to map a natural language utterance to an executable program (logical form) (Zelle and Mooney, 1996; Wong and Mooney, 2007; Zettlemoyer and Collins, 2007). Recently, neural network based approaches (Dong and Lapata, 2016; Jia and Liang, 2016; Xiao et al., 2016; Guu et al., 2017; Dong et al., 2018) have achieved promising performance in semantic parsing. However, neural network approaches are data hungry, which performances closely correlate with the volume of training data. In this work, we study the influence of training data on the accuracy of neural semantic parsing, and how to train a state-of-the-art model with less training data.

We conduct the study on WikiSQL (Zhong et al., 2017), the largest hand-annotated semantic parsing dataset which is larger than other datasets in terms of both the number of logical forms and the number of schemata. The task is to map a natural language question to a SQL query. We use a state-of-the-art end-to-end semantic parser based on neural networks (detailed in Section 3), and vary the number of supervised training instances. Results show that there is a logarithmic relationship between accuracy and the amount of training data, which is consistent with the observations in computer vision tasks (Sun et al., 2017).

We further study how to achieve state-of-the-art parsing accuracy with less supervised data, since annotating a large scale semantic parsing dataset requires funds and domain expertise. We achieve this through question generation, which generates natural language questions from SQL queries. Our question generation model is based on sequence-to-sequence learning. Latent variables (Cao and Clark, 2017) are introduced to increase the diversity of generated questions. The artificially generated question-SQL pairs can be viewed as pseudo-labeled data, which can be combined with a small amount of human-labeled data to train the semantic parser.

Results on WikiSQL show that the state-of-the-art logical form accuracy drops from 60.7% to 53.7% with only thirty percent of training data, while increasing to 61.0% when we combine the pseudo-labeled data generated from the question generation model. Applying the question generation model to full training data brings further improvements with 3.0% absolute gain. We further conduct a transfer learning experiment that applies our approach trained on WikiSQL to WikiTableQuestions (Pasupat and Liang, 2015). Results show that incorporating generated instances improves the state-of-the-art neural semantic parser (Krishnamurthy et al., 2017).

2 Overview of the Approach

Our task aims to map a question to a SQL query, which is executable over a table to yield the an-

---

[*] Work done while this author was an intern at Microsoft Research.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1597–1607
Brussels, Belgium, October 31 - November 4, 2018.

swer. Formally, the task takes a question q and a table t consisting of n column names and $n \times m$ cells as the input, and outputs a SQL query y. In this section, we describe an overview of our approach, which is composed of several components.

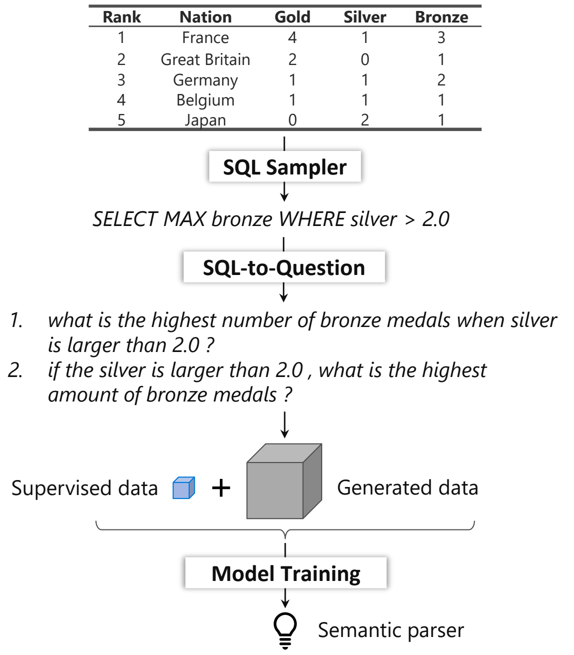

Figure 1: An overview of our approach that improves semantic parsing with question generation.

Figure 1 gives an overview of our approach. First, given a table, a SQL query sampler is used to sample valid, realistic, and representative SQL queries. Second, a question generation component takes SQL queries as inputs to obtain natural language questions. Here, the question generation model is learnt from a small-scale supervised training data that consists of SQL-question pairs. Lastly, the generated question-SQL pairs are viewed as the pseudo-labeled data, which are combined with the supervised training data to train the semantic parser.

Since we conduct the experiment on WikiSQL dataset, we follow Zhong et al. (2017) and use the same template-based SQL sampler, as summarized in Table 1. The details about the semantic parser and the question generation model will be introduced in Sections 3 and Section 4, respectively.

3 Semantic Parsing Model

We use a state-of-the-art end-to-end semantic parser (Sun et al., 2018) that takes a natural language question as the input and outputs a SQL query, which is executed on a table to obtain the answer. To make the paper self-contained, we briefly describe the approach in this section.

Format of a Sampled SQL Query	
SELECT *agg_op* *agg_col* From *table* WHERE *cond1_col* *cond1_op* *cond1* AND *cond2_col* ...	
Sampling Rules	
Variable	Sampling range
agg_col or *cond_col*	The aggregation column *agg_col* and the condition column *cond_col* can be one of columns in the table.
agg_op	The aggregation operator *agg_op* can be empty or *COUNT*. If the type of *agg_col* is numeric, *agg_op* can additionally be one of *MAX* and *MIN*.
cond_op	The condition operator *cond_op* is =. If the type of *cond_col* is numeric, *cond_op* can additionally be one of > and <.
cond	The condition value *cond* can be any cell value under the *cond_col*. If the type of *cond_col* is numeric, *cond* can be numerical value sampled from minimum value to maximum value in the *cond_col*.
Filter Rules	
1.The condition will be removed if doing the action does not change the execution result. 2.We only save the sampled SQL queries that produce non-empty result set.	

Table 1: The SQL sampler of (Zhong et al., 2017).

The semantic parser is abbreviated as **STAMP**, which is short for Syntax- and Table- Aware seMantic Parser. Based on the encoder-decoder framework, STAMP takes a question as the input and generates a SQL query. It extends pointer networks (Zhong et al., 2017; Vinyals et al., 2015) by incorporating three "channels" in the decoder, in which the column channel predicts column names, the value channel predicts table cells and the SQL channel predicts SQL keywords. An additional switching gate selects which channel to be used for generation. In STAMP, the probability of a token to be generated is calculated as Equation 1, where $p_z(\cdot)$ is the probability of the channel z_t to be chosen, and $p_w(\cdot)$ is the probability distribution of generating a word y_t from the selected channel.

$$p(y_t|y_{<t},x) = \sum_{z_t} p_w(y_t|z_t,y_{<t},x)p_z(z_t|y_{<t},x) \tag{1}$$

Specifically, the encoder takes a question as the input, uses bidirectional RNN with GRU cells to compute the hidden states, and feeds the concatenation of both ends as the initial state of the decoder. The decoder has another GRU to calculate the hidden states.

Each channel is implemented with an attentional neural network. In the SQL channel, the input of the attention module includes the decoder hidden state and the embedding of the SQL keyword to be calculated (i.e. e_i^{sql}).

$$p_w^{sql}(i) \propto exp(W_{sql}[h_t^{dec}; e_i^{sql}]) \tag{2}$$

In the column channel, the vector of a column name includes two parts, as given in Equation 3. The first vector (h_i^{col}) is calculated with a bidirectional GRU because a column name might contain multiple words. The second vector is a question-aware cell vector, which is weighted averaged over the cell vectors belonging to the column. Cell vectors (h_i^{cell}) are also obtained by a bidirectional GRU. The importance of a cell is measured by the number of co-occurred question words, which is further normalized through a $softmax$ function to yield the final weight $\alpha_j^{cell} \in [0, 1]$.

$$p_w^{col}(i) \propto exp(W_{col}[h_t^{dec}; h_i^{col}; \sum_{j \in col_i} \alpha_j^{cell} h_j^{cell}]) \tag{3}$$

In the value channel, the model has two distributions and weighted average them as Equation 4. Similar to $p^{sql}(\cdot)$, a standard cell distribution $\hat{p}_w^{cell}(\cdot)$ is calculated over the cells belonging to the last predicted column name. They incorporate an additional probability distribution $\alpha^{cell}(\cdot)$ based on the aforementioned word co-occurrence. The hyper parameter λ is tuned on the dev set.

$$p_w^{cell}(j) = \lambda \hat{p}_w^{cell}(j) + (1 - \lambda)\alpha_j^{cell} \tag{4}$$

Please see more details on model training and inference in Sun et al. (2018).

4 Question Generation Model

In this section, we present our SQL-to-question generation approach, which takes a SQL query as the input and outputs a natural language question. Our approach is based on sequence-to-sequence learning (Sutskever et al., 2014; Bahdanau et al., 2015). In order to replicate rare words from SQL queries, we adopt the copying mechanism. In addition, we incorporate latent variables to increase the diversity of generated questions.

4.1 Encoder-Decoder

Encoder: A bidirectional RNN with gated recurrent unit (GRU) (Cho et al., 2014) is used as the encoder to read a SQL query $x = (x_1, ..., x_T)$. The forward RNN reads a SQL query in a left-to-right direction, obtaining hidden states $(\overrightarrow{h_1}, ..., \overrightarrow{h_T})$. The backward RNN reads reversely and outputs $(\overleftarrow{h_1}, ..., \overleftarrow{h_T})$. We then get the final representation $(h_1, ..., h_T)$ for each word in the query, where $h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$. The representation of the source sentence $h_x = ([\overrightarrow{h_T}; \overleftarrow{h_1}])$ is used as initial hidden state of the decoder.

Decoder: We use a GRU with an attention mechanism as the decoder. At each time-step t, the attention mechanism obtains the context vector c_t that is computed same as the multiplicative attention (Luong et al., 2015). Afterwards, the concatenation of the context vector, the embedding of the previously predicted word y_{t-1}, and the last hidden state s_{t-1} is fed to the next step.

$$s_t = GRU(s_{t-1}, y_{t-1}, c_t) \tag{5}$$

After obtaining hidden states s_t, we adopt the copying mechanism that predicts a word from the target vocabulary or from the source sentence (detailed in Subsection 4.2).

4.2 Incorporating Copying Mechanism

In our task, the generated question utterances typically include informative yet low-frequency words such as named entities or numbers. Usually, these words are not included in the target vocabulary but come from SQL queries. To address this, we follow CopyNet (Gu et al., 2016) and incorporate a copying mechanism to select whether to generate from the vocabulary or copy from SQL queries.

The probability distribution of generating the t-th word is calculated as Equation 6, where $\psi_g(\cdot)$ and $\psi_c(\cdot)$ are scoring functions for generating from the vocabulary $\boldsymbol{\nu}$ and copying from the source sentence x, respectively.

$$p(y_t|y_{<t}, x) = \frac{e^{\psi_g(y_t)} + e^{\psi_c(y_t)}}{\sum_{v \in \boldsymbol{\nu}} e^{\psi_g(v)} + \sum_{w \in \mathbf{x}} e^{\psi_c(w)}} \tag{6}$$

The two scoring functions are calculated as follows, where W_g and W_c are model parameters, v_i is the one-hot indicator vector for y_i and h_i is the hidden state of word y_i in the source sentence.

$$\begin{aligned} \psi_g(y_i) &= v_i^T W_g s_t \\ \psi_c(y_i) &= tanh({h_i}^T W_c) s_t \end{aligned} \tag{7}$$

4.3 Incorporating Latent Variable

Increasing the diversity of generated questions is very important to improve accuracy, generalization, and stability of the semantic parser, since this increases the mount of training data and produces more diverse questions for the same intent. In this work, we incorporate stochastic latent variables (Cao and Clark, 2017; Serban et al., 2017) to the sequence-to-sequence model in order to increase question diversity.

Specifically, we introduce a latent variable $z \sim p(z)$, which is a standard Gaussian distribution $\mathcal{N}(0, I_n)$ in our case, and calculate the likelihood of a target sentence y as follows:

$$p(y|x) = \int_z p(y|z,x)p(z)\,dz \tag{8}$$

We maximize the evidence lower bound (ELBO), which decomposes the loss into two parts, including (1) the KL divergence between the posterior distribution and the prior distribution, and (2) a cross-entropy loss between the generated question and the ground truth.

$$\begin{aligned} logp(y|x) \geq -D_{KL}(Q(z|x,y)||p(z)) \\ +E_{z \sim Q} logp(y|z,x) \end{aligned} \tag{9}$$

The KL divergence in Equation 9 is calculated as follow, where n is the dimensionality of z.

$$\begin{aligned} &D_{KL}(Q(z|x,y)||p(z)) = \\ &-\frac{1}{2}\sum_{j=1}^{n}(1 + log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) \end{aligned} \tag{10}$$

$Q(z|x,y)$ is a posterior distribution with Gaussian distribution. The mean μ and standard deviation σ are calculated as follows, where h_x and h_y are representations of source and target sentences in the encoder, respectively. Similar to h_x, h_y is obtained by encoding the target sentence.

$$\begin{aligned} \mu &= W_\mu[h_x; h_y] + b_\mu \\ log(\sigma^2) &= W_\sigma[h_x; h_y] + b_\sigma \end{aligned} \tag{11}$$

4.4 Training and Inference

At the training phase, we sample z from $Q(z|x,y)$ using the re-parametrization trick (Kingma and Welling, 2014), and concatenate the source last hidden state h_x and z as the initial state of the decoder. Since the model tends to ignore the latent variables by forcing the KL divergence to 0 (Bowman et al., 2016), we add a variable weight to the KL term during training. At the inference phase, the model will generate different questions by first sampling z from $p(z)$, concatenating h_x and z as the initial state of the decoder, and then decoding deterministically for each sample.

Here, we list our training details. We set the dimension of the encoder hidden state as 300, and the dimension of the latent variable z as 64. We use dropout with a rate of 0.5, which is applied to the inputs of RNN. Model parameters are initialized with uniform distribution, and updated with stochastic gradient decent. Word embedding values are initialized with Glove vectors (Pennington et al., 2014). We set the learning rate as 0.1 and the batch size as 32. We tune hyper parameters on the development, and use beam search in the inference process.

5 Experiment

We conduct experiments on the WikiSQL dataset[1] (Zhong et al., 2017). WikiSQL is the largest hand-annotated semantic parsing dataset which is an order of magnitude larger than other datasets in terms of both the number of logical forms and the number of schemata (tables). WikiSQL is built by crowd-sourcing on Amazon Mechanical Turk, including 61,297 examples for training, and 9,145/17,284 examples for development/testing. Each instance consists of a natural language question, a SQL query, a table and a result. Here, we follow Zhong et al. (2017) to use two evaluation metrics. One is logical form accuracy (Acc$_{lf}$), which measures the percentage of exact string match between the generated SQL queries and the ground truth SQL queries. Since different logical forms might obtain the same result, another metric is execution accuracy (Acc$_{ex}$), which is the percentage of the generated SQL queries that result in the correct answer.

5.1 Impact of Data Size

We study how the number of training instances affects the accuracy of semantic parsing.

In this experiment, we randomly sample 20 subsets of examples from the WikiSQL training data, incrementally increased by 3K examples (about 1/20 of the full WikiSQL training data). We use the same training protocol and report the accuracy of the STAMP model on the dev set. Results are given in Figure 2. It is not surprising that more training examples bring higher accuracy. Interestingly, we observe that both accuracies of the neural network based semantic parser grow logarith-

---

[1] `https://github.com/salesforce/WikiSQL`

Methods	Training Data	Dev		Test	
		Acc_{lf}	Acc_{ex}	Acc_{lf}	Acc_{ex}
Attentional Seq2Seq	100%	23.3%	37.0%	23.4%	35.9%
Aug.PntNet (Zhong et al., 2017)	100%	44.1%	53.8%	43.3%	53.3%
Aug.PntNet (re-implemented by us)	100%	51.5%	58.9%	52.1%	59.2%
Seq2SQL (Zhong et al., 2017)	100%	49.5%	60.8%	48.3%	59.4%
SQLNet (Xu et al., 2017)	100%	–	69.8%	–	68.0%
STAMP	30%	54.6%	69.7%	53.7%	68.9%
STAMP + QG	30%	61.6%	74.4%	61.2%	73.9%
STAMP	100%	61.5%	74.8%	60.7%	74.4%
STAMP + QG	100%	64.3%	76.5%	63.7%	75.5%

Table 2: Performance of different approaches on the WikiSQL dataset. The two evaluation metrics are logical form accuracy (Acc_{lf}) and execution accuracy (Acc_{ex}). The settings of the training data represent the proportion of supervised data we use.

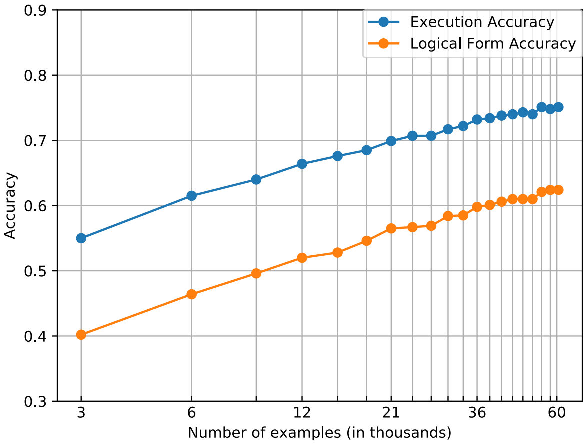

Figure 2: Semantic parsing accuracies of the STAMP model on WikiSQL. The x-axis is the training data size in log-scale, and the y-axis includes two evaluation metrics Acc_{lf} and Acc_{ex}.

mically as training data expands, which is consistent with the observations in computer vision tasks (Sun et al., 2017).

5.2 Model Comparisons

We report the results of existing methods on WikiSQL, and demonstrate that question generation is an effective way to improve the accuracy of semantic parsing. Zhong et al. (2017) implement several methods, including **Attentional Seq2Seq**, which is a basic attentional sequence-to-sequence learning baseline; **Aug.PntNet**, which is an augmented pointer network in which words of the target sequence come from the source sequence; and **Seq2SQL** which extends Aug.PntNet by further learning two separate classifiers for SELECT aggregator and SELECT column. Xu et al. (2017) develop **SQLNet**, which uses two separate models to predict SELECT and WHERE clauses, respectively, and introduce a sequence-to-set neural network to predict the WHERE clause. **STAMP** stands for the semantic parser which has been described in Section 3.

From Table 2, we can see that STAMP performs better than existing systems when trained on the full WikiSQL training dataset, achieving state-of-the-art execution accuracy and logical form accuracy on WikiSQL. We further conduct experiments to demonstrate the effectiveness of our question generation driven approach. We run the entire pipeline (STAMP+QG) with different percentages of training data. The second column "Training Data" in Table 2 and the x-axis in Figure 3 represent the proportion of WikiSQL training data we use for training the QG model and semantic parser. That is to say, STAMP +QG with 30% means that we sample 30% WikiSQL training data to train the QG model, and then combine QG generated data and exactly the same 30% WikiSQL training data we sampled before to train the semantic parser. In this experiment, we sample five SQL queries for each table in the training data, resulting in 43.5K SQL queries. Applying the QG model on these SQL queries, we get 92.8K SQL-question pairs. From Figure 3, we see that accuracy increases as the amount of supervised training data expands. Results show that QG empowers the STAMP model to achieve the same accuracy on WikiSQL dataset with 30% of the training data. Applying QG to the STAMP model under the full setting brings further improvements, resulting in new state-of-the-art accuracies.

Methods	Dev			Test		
	Acc_{sel}	Acc_{agg}	Acc_{where}	Acc_{sel}	Acc_{agg}	Acc_{where}
Aug.PntNet (re-implemented by us)	80.9%	89.3%	62.1%	81.3%	89.7%	62.1%
Seq2SQL (Zhong et al., 2017)	89.6%	90.0%	62.1%	88.9%	90.1%	60.2%
SQLNet (Xu et al., 2017)	91.5%	90.1%	74.1%	90.9%	90.3%	71.9%
STAMP	89.4%	89.5%	77.1%	88.9%	89.7%	76.0%
STAMP+QG	89.7%	90.1%	79.8%	89.1%	90.2%	79.0%

Table 3: Fine-grained accuracies on the WikiSQL dev and test sets. Logical form accuracy (Acc_{lf}) is evaluated on SELECT column (Acc_{sel}) , SELECT aggregator (Acc_{agg}), and WHERE clause (Acc_{where}), respectively. All these models are trained on the full WikiSQL training data.

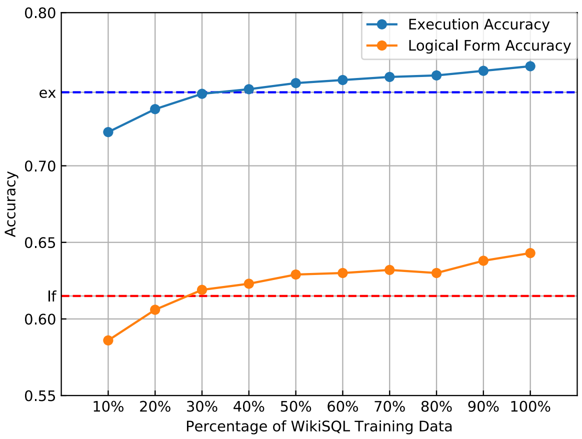

Figure 3: Accuracies of STAMP+QG with different portions of supervised data. Dashed lines are Acc_{lf} and Acc_{ex} of STAMP on the full training data.

5.3 Fine-grained Accuracies

Since SQL queries in WikiSQL consist of SELECT column, SELECT aggregator, and WHERE clause, we report fine-grained accuracies with regard to these aspects, respectively.

From Table 3, we observe that the main advantage of STAMP+QG over STAMP comes from the prediction of the WHERE clause, which is also the main challenge of the WikiSQL dataset. We further analyze STAMP and STAMP+QG on the WHERE clause by splitting the dev and test sets into three groups according to the number of conditions in the WHERE clause. From Table 4, we see that combining QG is helpful when the number of WHERE conditions is more than one. The main reason is that dominant instances in the WikiSQL training set have only one WHERE condition, as shown in Table 5, thus the model might not have memorized enough patterns for the other two limited-data groups. Therefore, the pseudo-labeled instances generated by our SQL sampler and QG approach are more precious to the limited-data groups (i.e #where =2 and #where≥3).

#where	STAMP		STAMP+QG	
	dev	test	dev	test
$= 1$	80.9%	80.2%	81.5%	80.9%
$= 2$	65.1%	65.4%	68.3%	66.9%
≥ 3	44.1%	48.2%	53.4%	51.9%

Table 4: Execution accuracy (Acc_{ex}) on different groups of WikiSQL dev and test sets.

#where	supervised data	generated data
$= 1$	69.1%	55.4%
$= 2$	24.1%	33.0%
≥ 3	6.1%	11.4%

Table 5: Distribution of the number of WHERE conditions in supervised and generated data.

5.4 Influences of Different QG Variations

To better understand how various components in our QG model impact the overall performance, we study different QG model variations. We use three evaluation metrics, including two accuracies and BLEU score (Papineni et al., 2002). The BLEU score evaluates the question generation.

Methods	Scale	BLEU	Acc_{lf}	Acc_{ex}
s2s	30%	20.6	59.0%	72.1%
s2s+lv	30%	22.1	60.0%	72.3%
s2s+cp	30%	29.6	60.8%	73.5%
s2s+cp+lv	30%	29.5	61.2%	73.9%
s2s	100%	26.0	62.6%	74.9%
s2s+lv	100%	26.3	63.0%	75.3%
s2s+cp	100%	31.5	63.2%	75.6%
s2s+cp+lv	100%	31.6	63.7%	75.5%

Table 6: Performances of different question generation variations.

SQL	SELECT COUNT 2nd leg WHERE aggregate = 7-2
Question (ground truth)	what is the total number of 2nd leg where aggregate is 7-2
Question (s2s + cp)	how many 2nd leg with aggregate being 7-2
Question (s2s + cp + lv)	(1) what is the total number of 2nd leg when the aggregate is 7-2 ? (2) how many 2nd leg with aggregate being 7-2 (3) name the number of 2nd leg for 7-2

Table 7: Generated examples from different question generation model variations.

Results are shown in Table 6, in which **s2s** represents the basic attentional sequence-to-sequence learning model (Luong et al., 2015), **cp** means the copying mechanism, and **lv** stands for the latent variable. We can see that incorporating a latent variable improves QG model performance, especially in limit-supervision scenarios. This is consistent with our intuition that the performance of the QG model is improved by incorporating the copying mechanism, since rare words of great importance mainly come from the input sequence.

To better understand the impact of incorporating a latent variable, we show examples generated by different QG variations in Table 7. We can see that incorporating a latent variable empowers the model to generate diverse questions for the same intent.

5.5 Transfer Learning on WikiTableQuestions

In this part, we conduct an extensional experiment on WikiTableQuestions[2] (Pasupat and Liang, 2015) in a transfer learning scenario to verify the effectiveness of our approach. WikiTableQuestions contains 22,033 complex questions on 2,108 Wikipedia tables. Each instance consists of a natural language question, a table and an answer. Following Pasupat and Liang (2015), we report development accuracy which is averaged over the first three 80-20 training data splits. Test accuracy is reported on the train-test data.

In this experiment, we apply the QG model learnt from WikiSQL to improve the state-of-the-art semantic parser (Krishnamurthy et al., 2017) on this dataset. Different from WikiSQL, this dataset requires question-answer pairs for training. Thus, we generate question-answer pairs by follow steps. We first sample SQL queries on the tables from WikiTableQuestions, and then use our QG model to generate question-SQL pairs. Afterwards, we obtain question-answer pairs by executing SQL queries. The generated question-answer pairs will be combined with the original WikiTableQuestions training data to train the model.

	Dev	Test
Pasupat and Liang (2015)	37.0%	37.1%
Neelakantan et al. (2016)	37.5%	37.7%
Haug et al. (2017)	-	38.7%
Zhang et al. (2017)	40.4%	43.7%
STAMP (WikiSQL)	-	14.5%
STAMP (WikiSQL) + QG	-	15.2%
NSP	41.9%	43.8%
NSP + QG	42.2%	44.2%

Table 8: Accuracy (Acc_{ex}) of different approaches on WikiTableQuestion dev and test sets.

Results are shown in Table 8, in which **NSP** is short for the state-of-the-art neural semantic parser (Krishnamurthy et al., 2017). Since the train-test data used in NSP is different from others, we retrain the NSP under the same protocol. **STAMP (WikiSQL)** means that the STAMP model trained on WikiSQL is directly tested on WikiTableQuestions. Despite applying QG slightly improves STAMP in this setting, the low accuracy reflects the different question distribution between these two datasets. In the supervised learning setting, we can see that incorporating QG further improves the accuracy of NSP from 43.8% to 44.2%.

5.6 Discussion

To better understand the limitations of our QG model, we analyze a randomly selected set of 100 questions. We observe that 27% examples do not correctly express the meanings of SQL queries, among which the majority of them miss information from the WHERE clause. This problem might be mitigated by incorporating a dedicated encoder/decoder that takes into account the SQL structure. Among the other 73% of examples that correctly express SQL queries, there are two po-

---

[2] `https://nlp.stanford.edu/software/sempre/wikitable/`

tential directions to make further improvements. The first direction is to leverage table information such as the type of a column name or column-cell correlations. For instance, without knowing that cells under the column name "*built*" are all building years, the model hardly predicts a question "*what is the average building year for superb?*" for "*SELECT AVG built WHERE name = superb*". The second direction is to incorporate common knowledge, which would help the model to predict *the earliest week* rather than *the lowest week*.

6 Related Work

Semantic Parsing. Semantic parsing is a fundamental problem in NLP that maps natural language utterances to logical forms, which could be executed to obtain the answer (denotation) (Zettlemoyer and Collins, 2005; Liang et al., 2011; Berant et al., 2013; Krishnamurthy and Kollar, 2013; Pasupat and Liang, 2016; Iyer et al., 2017). Existing works can be classified into three areas, including (1) the language of the logical form, e.g. first-order logic, lambda calculus, lambda dependency-based compositional semantics (lambda DCS) and structured query language (SQL); (2) the form of the knowledge base, e.g. facts from large collaborative knowledge bases, semi-structured tables and images; and (3) the supervision used for learning the semantic parser, e.g. question-denotation pairs and question-logical form pairs. In this work, we regard the table as the knowledge base, which is critical for accessing relational databases with natural language, and also for serving information retrieval for structured data. We use SQL as the logical form, which has a broad acceptance to the public. In terms of supervision, this work uses a small portion of question-logical form pairs to initialize the QA model and train the QG model, and incorporate more generated question-logical form pairs to further improve the QA model.

Question Generation Our work also relates to the area of question generation, which has drawn plenty of attention recently partly influenced by the remarkable success of neural networks in text generation. Studies in this area are classified based on the definition of the answer, including a sentence (Heilman, 2011), a topic word (Chali and Hasan, 2015), a fact (including a subject, a relation phrase and an object) from knowledge bases (Serban et al., 2016), an image (Mostafazadeh et al., 2016), etc. Recent studies in machine reading comprehension generate questions from an answer span and its context from the document (Du et al., 2017; Golub et al., 2017). Wang et al. (2015) first generate logical forms, and then use AMTurkers to paraphrase them to get natural language questions. Iyer et al. (2017) use a template-based approach based on the Paraphrase Database (Ganitkevitch et al., 2013) to generate questions from SQL. In this work, we generate questions from logical forms, in which the amount of information from two directions are almost identical. This differs from the majority of existing studies because a question typically conveys less semantic information than the answer.

Improving QA with QG This work also relates to recent studies that uses a QG model to improve the performance of a discriminative QA model (Wang et al., 2017; Yang et al., 2017; Duan et al., 2017; Konstas et al., 2017). The majority of these works generate a question from an answer, while there also exists a recent work (Dong et al., 2017) that generates a question from a question through paraphrasing. In addition, Tang et al. (2017) consider QA and QG as dual tasks, and further improve the QG model in a dual learning framework. These works fall into three categories: (1) regarding the artificially generated results as additional training instances (Yang et al., 2017; Golub et al., 2017); (2) using generated questions to calculate additional features (Duan et al., 2017; Dong et al., 2017); and (3) using the QG results as additional constraints in the training objectives (Tang et al., 2017). This work belongs to the first direction. Our QG approach takes a logical form as the input, and considers the diversity of generated questions by incorporating latent variables.

7 Conclusion

In this paper, we observe the logarithmic relationship between the accuracy of a semantic parser and the amount of training data, and present an approach that improves neural semantic parsing with question generation. We show that question generation helps us obtain a state-of-the-art neural semantic parser with less supervised data, and further improves the state-of-the-art model with full annotated data on WikiSQL and WikiTableQuesions datasets. In future work, we would like to make use of table information and external knowledge to improve our QG model. We also plan to apply the approach to other tasks.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61472453, U1401256, U1501252, U1611264,U1711261,U1711262). Thanks to the anonymous reviewers for their helpful comments and suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceeding of ICLR*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, 5, page 6.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *conference on computational natural language learning*, pages 10–21.

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. *arXiv preprint arXiv:1702.05962*.

Yllias Chali and Sadid A Hasan. 2015. Towards topic-to-question generation. *Computational Linguistics*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. *arXiv preprint arXiv:1708.06022*.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 743–753. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 835–844.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1051–1062.

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. 2017. Neural multi-step reasoning for question answering on semi-structured tables. *arXiv preprint arXiv:1702.06589*.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *international conference on learning representations*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 590–599.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *empirical methods in natural language processing*, pages 1412–1421.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. *arXiv preprint arXiv:1603.06059*.

Arvind Neelakantan, Quoc V Le, Martin Abadi, Andrew McCallum, and Dario Amodei. 2016. Learning a natural language interface with neural programmer. *arXiv preprint arXiv:1611.08945*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 23–32.

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

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *ACL*, pages 588–598.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. *arXiv preprint arXiv:1707.02968*.

Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic parsing with syntax- and table-aware sql generation. *arXiv preprint arXiv:1804.08338*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. *arXiv preprint arXiv:1705.10513*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1332–1342.

Yuk Wah Wong and Raymond J Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Annual Meeting-Association for computational Linguistics*, 1, page 960.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350, Berlin, Germany. Association for Computational Linguistics.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 658–666.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*, pages 678–687.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1225–1234, Copenhagen, Denmark. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

SemRegex: A Semantics-Based Approach for Generating Regular Expressions from Natural Language Specifications

Zexuan Zhong[1], **Jiaqi Guo**[2], **Wei Yang**[3], **Jian Peng**[1], **Tao Xie**[1],
Jian-Guang Lou[4], **Ting Liu**[2], **Dongmei Zhang**[4]
[1]University of Illinois at Urbana-Champaign
[2]Xi'an Jiaotong University, [3]University of Texas at Dallas
[4]Microsoft Research Asia
`{zexuan2,jianpeng,taoxie}@illinois.edu`
`jasperguo2013@stu.xjtu.edu.cn, tingliu@mail.xjtu.edu.cn`
`wei.yang@utdallas.edu, {jlou,dongmeiz}@microsoft.com`

Abstract

Recent research proposes syntax-based approaches to address the problem of generating programs from natural language specifications. These approaches typically train a sequence-to-sequence learning model using a syntax-based objective: maximum likelihood estimation (MLE). Such syntax-based approaches do not effectively address the goal of generating semantically correct programs, because these approaches fail to handle *Program Aliasing*, i.e., semantically equivalent programs may have many syntactically different forms. To address this issue, in this paper, we propose a semantics-based approach named SemRegex. SemRegex provides solutions for a subtask of the program-synthesis problem: generating regular expressions from natural language. Different from the existing syntax-based approaches, SemRegex trains the model by maximizing the expected semantic correctness of the generated regular expressions. The semantic correctness is measured using the DFA-equivalence oracle, random test cases, and distinguishing test cases. The experiments on three public datasets demonstrate the superiority of SemRegex over the existing state-of-the-art approaches.

1 Introduction

Translating natural language (NL) descriptions into executable programs is a fundamental problem for computational linguistics. An end user may have difficulty to write programs for a certain task, even when the task is already specified in NL. For some tasks, even for developers, who have experience in writing programs, it can be time consuming and error prone to write programs based on the NL description of the task. Naturally, automatically synthesizing programs from NL can help alleviate the preceding issues for both end users and developers.

Recent research proposes syntax-based approaches to address some tasks of this problem in different domains, such as regular expressions (regex) (Locascio et al., 2016), Bash scripts (Lin et al., 2017), and Python programs (Yin and Neubig, 2017). These approaches typically train a sequence-to-sequence learning model using maximum likelihood estimation (MLE). Using MLE encourages the model to output programs that are syntactically similar with the ground-truth programs in the training set. However, such syntax-based training objective deviates from the goal of synthesizing semantically equivalent programs. Specifically, these syntax-based approaches fail to handle the problem of *Program Aliasing* (Bunel et al., 2018), i.e., a semantically equivalent program may have many syntactically different forms. Table 1 shows some examples of the *Program Aliasing* problem. Both Program 1 and Program 2 are desirable outputs for the given NL specification but one of them is penalized by syntax-based approaches if the other one is used as the ground truth, compromising the overall effectiveness of these approaches.

In this paper, we focus on generating regular expressions from NL, an important task of the program-synthesis problem, and propose SemRegex, a semantics-based approach to generate regular expressions from NL specifications. Regular expressions are widely used in various applications, and "regex" is one of the most common tags in Stack Overflow[1] with more than $190,000$ related questions. The huge number of regex-related questions indicates the importance of this task.

Different from the existing syntax-based approaches, SemRegex alters the syntax-based training objective of the model to a semantics-based objective. To encourage the translation model to generate semantically correct regular expressions, instead of MLE, SemRegex trains the model by maximizing the expected semantic correctness of

---

[1] `https://stackoverflow.com/questions/tagged/regex`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1608–1618
Brussels, Belgium, October 31 - November 4, 2018.

Table 1: Examples of *Program Aliasing*: for each NL specification, Program 2 is semantically equivalent to Program 1; however, if Program 1 is the ground truth in the training set, Program 2 is penalized by syntax-based approaches although it is a desirable program.

Domain	**NL Specification**	**Program 1**	**Program 2**
Regex	Match lines that start with an uppercase vowel and end with ‘X’	`([AEIOUaeiou]&[A-Z]).*X`	`([AEIOU].*)&(.*X)`
Bash	Rename file ‘f1’ to ‘f1.txt’	`mv ’f1’ ’f1.txt’`	`cp ’f1’ ’f1.txt’; rm ’f1’`
Python	Assign the greater value of ‘a’ and ‘b’ to variable ‘c’	`c = a if a > b else b`	`c = [b, a][a > b]`

generated regular expressions. We follow the technique of policy gradient (Williams, 1992) to estimate the gradients of the semantics-based objective and perform optimization.

The measurement of semantic correctness serves as a key part in the semantics-based objective, which should represent the semantics of programs. In this paper, we convert a regular expression to a minimal Deterministic Finite Automaton (DFA). Such conversion is based on the insight that semantically equivalent regular expressions have the same minimal DFAs. We define the semantic correctness of a generated regular expression as whether its corresponding minimal DFA is the same as the ground truth's minimal DFA.

When our approach is applied on domains other than regular expressions such as Python programs and Bash scripts, a perfect equivalence oracle such as minimal DFAs may not be feasibly available. To handle a more general case, we propose correctness assessment based on test cases for regular expression; such correctness assessment can be easily generalized for other tasks of program synthesis. Concretely, we generate test cases to represent semantics of the ground truth. For a generated regular expression, we assess its semantic correctness by checking whether it can pass all the test cases. However, a regular expression may have infinite positive (i.e., matched) or negative (i.e., unmatched) string examples; thus, we cannot perfectly represent the semantics. To use limited string examples to differentiate whether a generated regular expression is semantically correct or not, we propose an intelligent strategy for test generation to generate distinguishing test cases instead of just using random test cases.

We evaluate SemRegex on three public datasets: NL-RX-Synth, NL-RX-Turk (Locascio et al., 2016), and KB13 (Kushman and Barzilay, 2013). We compare SemRegex with the existing state-of-the-art approaches on the task of generating regular expressions from NL specifications. Our evaluation results show that SemRegex outperforms the start-of-the-art approaches on all of three datasets. The evaluation results confirm that by maximizing semantic correctness, the model can output more correct regular expressions even when the regular expressions are syntactically different from the ground truth.

In summary, this paper makes the following three main contributions. (1) We propose a semantics-based approach to optimize the semantics-based objective for the task of generating regular expressions from NL specifications. (2) We introduce the measurement of semantic correctness based on test cases, and propose a strategy to generate distinguishing test cases, in order to better measure the semantic correctness than using random test cases. (3) We evaluate our approach on three public datasets. The evaluation results show that our approach outperforms the existing state-of-the-art on all of the three datasets.

2 Problem Formulation

Consider the problem of automatically generating a regular expression R given an NL specification S as an input. Let $S = s_1, s_2, \ldots, s_m$ denote the NL specification, where s_i represents a word in the vocabulary; let $R = r_1, r_2, \ldots, r_n$ denote the regular expression, where r_i is a valid character in the regular expression.

We assume that we have a training set consisting of K NL and regular expression pairs:

$$\mathcal{D} = \left\{ (S^{(i)}, R^{(i)}) \right\}_{i=1..K}$$

Given an NL specification, it is possible to have multiple regular expressions fitting the specification. In the training set, only one regular expression is provided for each NL specification.

3 SemRegex Approach

In this section, we illustrate our SemRegex approach in detail. First, we introduce our model, which is a sequence-to-sequence learning model.

Next, we alter the standard Maximum Likelihood Estimation (MLE) objective to maximize semantic correctness. We leverage policy gradient to train the model with the semantics-based objective. Finally, we discuss how to measure semantic correctness.

3.1 Model

It is natural to apply a machine-translation model on the program-synthesis problem. We follow a previous attempt (Locascio et al., 2016) to use a sequence-to-sequence learning model (Sutskever et al., 2014) augmented with the attention mechanism (Bahdanau et al., 2014). The model consists of an encoder network and a decoder network. In both the encoder network and decoder network, we use LSTM (Hochreiter and Schmidhuber, 1997) units that can be summarized as follow:

$$\begin{aligned} i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ \tilde{c}_t &= \phi(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\ h_t &= o_t \circ \phi(c_t) \end{aligned}$$

where σ is the sigmoid function, ϕ is the hyperbolic tangent function, and $\circ$ is the element-wise multiplication; weight matrices W and U along with biases b are learnable parameters of the model. In the encoder network, the input x_t is an embedding vector of the word s_t in the NL input sequence. In the decoder network, the input x_t is an embedding vector of the previous character r_{t-1} in the output regular expression. The hidden vectors h_t of the encoder network are fed into an attention layer (Bahdanau et al., 2014) to output an overall representation of the input sentence considering the output position. The hidden vectors h_t of the decoder network are fed into a dense layer $z_t = W_z h_t$, where z_t holds the dimension of the vocabulary size of the regular expression. z_t is the output of the decoder network to predict the output character $r_t = \arg\max_j z_{t,j}$.

A softmax function is applied on z_t to obtain a probability distribution on output character candidates. The probability of character j at output position t is as follow:

$$p(r_t = j | r_{<t}, S) = \frac{e^{z_{t,j}}}{\sum_{j'} e^{z_{t,j'}}}$$

3.2 Training

Let θ represent all learnable parameters in the model. We discuss two objective functions of θ to train the model.

Maximum Likelihood Estimation (MLE). A sequence-to-sequence learning model learns the distribution of regular expressions R given an input NL sentence S:

$$p_\theta(R|S) = \prod_{t=1}^{T} p_\theta(r_t | r_{<t}, S)$$

By default, the sequence-to-sequence learning model uses maximum likelihood estimation (MLE) for training, i.e., maximizing the likelihood of mapping the input sequence to the output sequence for each pair in the training set. Specifically, the optimal parameters θ^* are obtained as follow:

$$\begin{aligned} \theta^* &= \arg\max_\theta \prod_{(S^{(i)},R^{(i)})\in\mathcal{D}} p_\theta(R^{(i)}|S^{(i)}) \\ &= \arg\max_\theta \sum_{(S^{(i)},R^{(i)})\in\mathcal{D}} \log p_\theta(R^{(i)}|S^{(i)}) \end{aligned}$$

Gradient descent is used to search out optimal parameters θ^*.

However, MLE fails to consider the fact that semantically equivalent regular expressions might be syntactically different. The MLE objective function forces the model to generate syntactically similar regular expressions, but penalizes semantically equivalent and syntactically different regular expressions. Such a syntax-based training objective does not fit our task's objective (i.e., generating any semantically correct regular expression).

Maximizing Semantic Correctness. To encourage the model to generate any semantically correct regular expression, we alter the MLE training objective function to maximize semantic correctness.

For an NL specification, we define a reward of a predicted regular expression $\boldsymbol{r}(R)$ as its semantic correctness (we discuss how to measure the correctness later in this section). We encourage the model to generate regular expressions to maximize expected rewards instead of MLE. Concretely, we train the model parameters θ to maximize the following objective function:

$$\begin{aligned} J(\theta) &= \sum_{(S^{(i)},R^{(i)})\in\mathcal{D}} \mathbb{E}_{R\sim p_\theta(\cdot|S^{(i)})} \boldsymbol{r}(R) \\ &= \sum_{(S^{(i)},R^{(i)})\in\mathcal{D}} \sum_R p_\theta(R|S^{(i)}) \boldsymbol{r}(R) \end{aligned}$$

However, to compute the expected reward, we need to go over all possible regular expressions, and the number of all possible regular expressions is infinite. To address this problem, we use the Monte Carlo estimate as the approximation of the expected value. Specifically, M regular expressions $R_1, \dots, R_M$ are sampled following the output probability of the model. We average the reward of each sample to estimate the expected reward:

$$J(\theta) \approx \sum_{(S^{(i)},R^{(i)})\in\mathcal{D}} \sum_{j=1}^{M} \frac{1}{M}\boldsymbol{r}(R_j), \quad \text{where } R_j \sim p_\theta(\cdot|S^{(i)})$$

In order to compute the gradient of the expected reward and to maximize the objective using gradient descent, we employ the REINFORCE technique of policy gradient (Williams, 1992), which is based on the following estimation:

$$\nabla_\theta J(\theta) \approx \sum_{(S^{(i)},R^{(i)})\in\mathcal{D}} \sum_{j=1}^{M} \frac{1}{M}\boldsymbol{r}(R_j)\nabla_\theta \log p_\theta(R_j|S^{(i)}), \quad \text{where } R_j \sim p_\theta(\cdot|S^{(i)})$$

In practice, we subtract the mean reward of all samples to reduce the variance of estimated gradient (Williams, 1992). The final gradient estimate is as follow:

$$\nabla_\theta J(\theta) \approx \sum_{(S^{(i)},R^{(i)})\in\mathcal{D}} \sum_{j=1}^{M} \frac{1}{M}\tilde{\boldsymbol{r}}(R_j)\nabla_\theta \log p_\theta(R_j|S^{(i)}),$$

$$\text{where } \tilde{\boldsymbol{r}}(R_j) = \boldsymbol{r}(R_j) - \sum_{j'=1}^{M} \frac{1}{M}\boldsymbol{r}(R_{j'})$$

The overall training algorithm is summarized in Algorithm 1. We initialize θ by pre-training the model using MLE on the training set. For each pair in training set, we sample M regular expressions to estimate the gradient.

3.3 Measurement of Semantic Correctness

In this paper, we propose two types of measuring semantic correctness based on minimal DFAs and test cases, respectively.

Minimal DFAs. We convert a regular expression to a minimal DFA and utilize the fact that equivalent regular expressions have the same minimal DFAs even when they are syntactically different (Hopcroft et al., 1979). For example, the minimal DFA of regular expression “`([ABab]&[A-Z]).*X`” is shown in Figure 1(a). A syntactically different regular expression “`((A|B).*)&(.*X)`” can be converted to the same minimal DFA as shown in Figure 1(a), indicating that these two regular expressions are semantically equivalent.

Algorithm 1: Policy-gradient method to maximize semantic correctness

```
Input: Training set: $\mathcal{D} = \{(S^{(i)}, R^{(i)})\}$
Initilize $\theta$ from pretrained model using MLE on $\mathcal{D}$ ;
for each epoch do
    for $(S^{(i)}, R^{(i)}) \in \mathcal{D}$ do
        Sample $R_1, \dots, R_M$ using current model ;
        Get rewards $\boldsymbol{r}(R_1), \dots, \boldsymbol{r}(R_M)$ ;
        Estimate $\nabla_\theta J(\theta)$ using (3.2) ;
        Update $\theta$ using $\nabla_\theta J(\theta)$ by gradient descent ;
    end
end
```

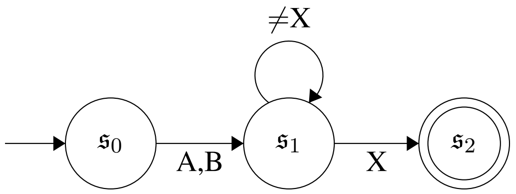

(a)

Path	String example
$\mathfrak{s}_0 \xrightarrow{A} \mathfrak{s}_1 \xrightarrow{X} \mathfrak{s}_2$	`AX`
$\mathfrak{s}_0 \xrightarrow{B} \mathfrak{s}_1 \xrightarrow{K} \mathfrak{s}_1 \xrightarrow{X} \mathfrak{s}_2$	`BKX`
$\mathfrak{s}_0 \xrightarrow{B} \mathfrak{s}_1 \xrightarrow{X} \mathfrak{s}_2$	`BX`

(b)

Figure 1: Minimal DFA converted from “`([ABab]&[A-Z]).*X`” and generated string examples, where $\mathfrak{s}_0$ represents the start state and $\mathfrak{s}_2$ is the only accept state.

We check whether two regular expressions are equivalent by checking whether their corresponding minimal DFAs are the same. When the policy-gradient method is performed, if a sampled regular expression R is equivalent to the ground truth, then $\boldsymbol{r}(R) = 1$; otherwise $\boldsymbol{r}(R) = 0$.

Test Cases. A perfect equivalence oracle such as using the minimal DFA may not be feasibly available for some tasks, e.g., when our approach is applied on other domains such as generating Bash scripts and Python programs. To handle a more general case, we propose correctness measurement based on test cases. We generate test cases (i.e., inputs and expected outputs) and check whether a program can pass the test cases that are generated from the ground truth to approximately check whether the program and the ground truth are equivalent.

Given a regular expression R, we generate test cases that contain positive (acceptable/matched) and negative (unacceptable/unmatched) string examples. Here we consider only positive examples because negative examples can be obtained by generating positive examples of its complement regular expression $\sim R$. To generate positive string examples from regular expression R, we convert R to its corresponding minimal DFA. Each positive string example corresponds to a path from the start state of the minimal DFA to any accept state[2], and vice versa. Thus, we generate paths randomly from the start state to any accept state, and convert the paths to their corresponding strings as shown in Figure 1(b). To generate distinct string examples, we aim to generate paths to cover as many transitions as possible. In particular, we mask all transitions that have been covered by previously generated paths. When we generate a new path, the not-covered transitions have higher priority to be explored than covered ones.

Because complex regular expressions may accept/match or reject/unmatch infinite string examples, we augment random generation with a new strategy to generate distinguishing test cases to better represent the semantics. Considering that the generated test cases are used to check whether a Monte-Carlo sampled regular expression is equivalent to the ground truth in the policy-gradient method, only test cases that can differentiate an incorrect sample and the ground truth are useful. Based on such insight, we give preference to test cases that differentiate Monte-Carlo samples and the ground truth. A challenge here is that we do not know the samples before performing the policy-gradient method. However, we find that there is a high chance to get the same samples repeatedly when the model is pre-trained using MLE on the training set, because sampling is following the distribution learned by the pre-trained model. Based on the observation, we use the Beam Search algorithm on the pre-trained model to obtain B most likely samples $\hat{R}_1, \ldots, \hat{R}_B$. We generate string examples that can differentiate these samples and the ground truth, named as distinguishing string examples. For each $\hat{R}$ and ground truth R, we construct a new regular expression $R\&(\sim\hat{R})$, and generate its string examples that can differentiate R and $\hat{R}$.

The overall idea of our strategy for generating string examples is shown in Algorithm 2. Once we have a set of positive and negative string examples, we define the reward of a regular expression as $\boldsymbol{r}(R) = 1$ if it can pass all the test cases, and $\boldsymbol{r}(R) = 0$ otherwise.

When extending SemRegex on other languages where a perfect equivalence oracle is not available, it is desirable to use a technique to generate test cases for a program. There exist techniques (discussed in Section 5) to generate test cases for a general executable program.

Algorithm 2: Generating distinguishing test cases for regular expressions

```
Input: Training set: $\mathcal{D} = \{(S^{(i)}, R^{(i)})\}$,
       the number of examples to generate:
       $T$, and a pre-trained model
Output: Positive and negative example
        sets $\mathcal{P}^{(i)}$ and $\mathcal{N}^{(i)}$
for $(S^{(i)}, R^{(i)}) \in \mathcal{D}$ do
    $\mathcal{P}^{(i)} \leftarrow \emptyset$ ;
    $\mathcal{N}^{(i)} \leftarrow \emptyset$ ;
    Beam search on pre-trained model to
      obtain $\hat{R}_1, \ldots, \hat{R}_B$ ;
    repeat
        Randomly pick a $j$ in $[1, B]$ ;
        $R_p \leftarrow R^{(i)}\&(\sim\hat{R}_j)$ ;
        $R_n \leftarrow (\sim R^{(i)})\&\hat{R}_j$ ;
        Generate an example $p$ from $R_p$ ;
        Generate an example $n$ from $R_n$ ;
        $\mathcal{P}^{(i)} \leftarrow \mathcal{P}^{(i)} \cup \{p\}$ ;
        $\mathcal{N}^{(i)} \leftarrow \mathcal{N}^{(i)} \cup \{n\}$ ;
    until $|\mathcal{P}^{(i)}| \geq T$ && $|\mathcal{N}^{(i)}| \geq T$;
end
```

4 Experiments

We evaluate the effectiveness of SemRegex by comparing it to the state-of-the-art approaches. We also study how using different measurements of correctness impacts the effectiveness of SemRegex.

---

[2] A DFA has one start state and a set of accept states.

4.1 Experiment Setup

Datasets. We conduct our experiments on three public datasets for the task of generating regular expressions from NL specifications.

- **KB13.** KB13 (Kushman and Barzilay, 2013) includes 824 pairs of NL and regular expression. When conducting data labeling, labeling workers are asked to generate the NL specifications to capture a subset of the lines in a file. Then programmers are asked to generate regular expressions for these NL specifications written by the labeling workers. We split the data into 75% training and 25% testing sets, following what the authors of KB13 do.

- **NL-RX-Synth.** NL-RX-Synth (Locascio et al., 2016) is a synthetic dataset much larger than KB13. Its authors define a small grammar for parsing regular expressions to NL. The grammar is used to stochastically generate 10,000 regular expressions and their corresponding synthetic NL specifications. We split the pairs into 65% training, 10% development, and 25% testing sets, following what the authors of NL-RX-Synth do.

- **NL-RX-Turk.** NL-RX-Turk (Locascio et al., 2016) comes from the NL-RX-Synth dataset. Instead of directly using synthetic NL descriptions in the dataset, the authors of NL-RX-Turk ask labeling workers to paraphrase the synthetic specifications. The dataset also consists of 10,000 pairs of NL and regular expression. We split the pairs into 65% training, 10% development, and 25% testing sets, following what the authors of NL-RX-Turk do.

Training Setting. We use a two-layer stacked LSTM architecture in both the encoder and decoder networks. The dimensions of encoder and decoder hidden states are set to 256. We use random embedding layers with the dimension of 128 for both input and output words. We also tune our hyper-parameters on the development set. The best results are obtained when the learning rate $= 0.001$ and the batch size = 25. We use the Monte-Carlo method to sample $M = 10$ regular expressions to estimate the gradient. To generate distinguishing string examples, we perform Beam Search to obtain $B = 10$ most likely samples. Before performing the policy-gradient method, we pre-train the model using MLE for 100 epochs. Then we train the model for 40 epochs using the policy-gradient method, and choose the model with the best effectiveness on the development set. Our model is implemented in TensorFlow (Abadi et al., 2016).

4.2 Results and Analysis

Comparison Results. We demonstrate the effectiveness of our approach by comparing it to the existing approaches including Semantic-Unify (Kushman and Barzilay, 2013) and Deep-Regex(MLE) (Locascio et al., 2016). We also compare the results of our approach with different measurements of semantic correctness. Table 2 shows the comparison results of different approaches, with detailed discussion as follows.

- **Semantic-Unify.** Semantic-Unify (Kushman and Barzilay, 2013) learns to parse NL to regular expressions. Similarly, DFA equivalence is applied as a semantic unification when training the parser.

- **Deep-Regex(MLE).** Deep-Regex(MLE) (Locascio et al., 2016) regards the problem as a black-box task of machine translation without utilizing any domain knowledge of regular expressions. A syntax-based objective (MLE) is used to train the model. To the best of our knowledge, Deep-Regex(MLE) is the state-of-the-art approach on these three datasets.

- **SemRegex(DFA Oracle).** In SemRegex (DFA Oracle), we use the oracle of DFA equivalence to measure semantic correctness. SemRegex(DFA Oracle) outperforms Deep-Regex(MLE), the existing state-of-the-art approach, by an accuracy increase of 12.6% on KB13, 2.9% on NL-RX-Synth, and 4.1% on NL-RX-Turk, respectively. Compared to Deep-Regex(MLE), the results demonstrate the effectiveness of maximizing semantic correctness during the training phase. SemRegex(DFA Oracle) shows more improvement on the KB13 dataset over Deep-Regex(MLE) than on other datasets. Such result indicates that supervised learning based on MLE is less effective to learn from a small training set. When the policy-gradient method is used, Monte-Carlo samples can provide more information beyond only training samples especially on a small training set;

Table 2: Effectiveness comparison of different approaches (using DFA-equivalence accuracy as metrics)

Approach	**KB13**	**NL-RX-Synth**	**NL-RX-Turk**
Semantic-Unify	65.5%	46.3%	38.6%
Deep-Regex(MLE)	65.6%	88.7%	58.2%
SemRegex(DFA Oracle)	**78.2%**	**91.6%**	**62.3%**
SemRegex(Distinguishing Test Cases)	77.5%	90.2%	61.3%
SemRegex(Random Test Cases)	66.5%	90.2%	59.5%

such more information significantly improves the effectiveness.

- **SemRegex(Distinguishing Test Cases).** When we do not have access to an oracle such as DFA equivalence, we can generate test cases to define semantic correctness. SemRegex(Distinguishing Test Cases) uses Algorithm 2 to generate distinguishing test cases (10 positive examples and 10 negative examples) that differentiate the results returned by Beam Search and the ground truth. The results show that by using distinguishing test cases, SemRegex(Distinguishing Test Cases) outperforms Deep-Regex(MLE), an existing syntax-based approach, on all of three datasets. Meanwhile, the effectiveness of SemRegex(Distinguishing Test Cases) drops no more than 1.4% on accuracy compared to SemRegex(DFA Oracle). Such result indicates that limited distinguishing test cases generated by the proposed strategy can well represent the semantics.

- **SemRegex(Random Test Cases).** SemRegex(Random Test Cases) generates random test cases instead of distinguishing test cases. It outperforms the existing approaches (Semantic-Unify and Deep-Regex(MLE)) because random test cases can still represent the semantics and differentiate some inequivalent regular expressions. Compared to SemRegex(Distinguishing Test Cases), its effectiveness shows a big drop on KB13 and a slight drop on NL-RX-Turk. Such results indicate the benefit of distinguishing test cases over random test cases.

Effectiveness of Semantics-Based Objective. To understand the effect of using a semantics-based learning objective, we record the semantic accuracy (DFA equivalence) and syntactic accuracy (exact-match) on the NL-RL-Turk testing set after each epoch as shown in Figure 2. During pre-training (epochs 1 to 100), we use MLE to train the model to increase both semantic accuracy and syntactic accuracy iteratively. Then, we alter the training objective to maximize the expected semantic correctness. We notice that while semantic accuracy continues increasing for about 10%, the syntactic accuracy does not show a significant growth after pre-training. Such result indicates that the model is no longer encouraged to generate regular expressions that are syntactically equivalent to the ground truths. Instead, the model learns to generate semantically correct regular expressions.

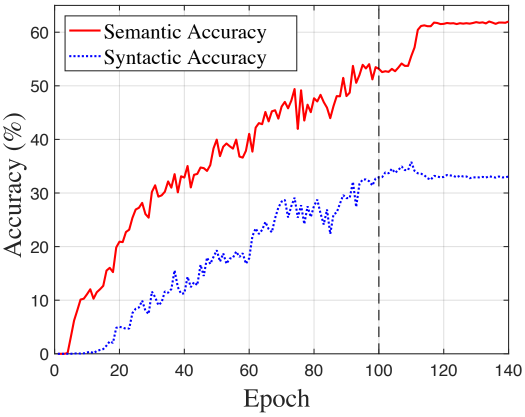

Figure 2: Semantic accuracy (DFA equivalence) and syntactic accuracy (exact-match) on the NL-RL-Turk testing set after each epoch. The training objective is replaced to maximize expected correctness after 100 epochs. The correctness is measured by the DFA-equivalence oracle in this figure.

Analysis of Semantic Correctness Based on Test Cases. The correctness measurements based on test cases serve as an approximate oracle. Figure 3 shows an example of how the approximate oracle helps make improvement. Furthermore, we evaluate how the correctness based on test cases is close to the DFA-equivalence oracle. In Monte-Carlo estimate, we count the samples with the approximate oracle that equals to the minimal DFA oracle. When using random test cases, there are 89.8% samples with the approximate oracle that equals to the minimal DFA oracle. When using distinguish-

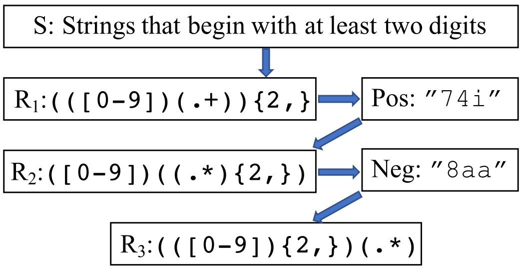

Figure 3: An example of how test cases help with training. At the beginning of the policy-gradient method, the model outputs an incorrect answer R_1, which cannot pass a positive test case. R_1 gets penalized because it receives a reward 0. Then the model changes to output an incorrect answer R_2, which cannot pass a negative test case. Similarly, R_2 gets penalized as training continues. R_3 receives a reward 1 because it passes all test cases, resulting in an increase of its likelihood from the model in iterations. Finally, the model outputs the correct answer R_3.

ing test cases, such percentage increases to 96.3%. Such result illustrates that test cases are able to approximately check the semantic equivalence even when the test cases are generated randomly. The result also suggests that distinguishing test cases represent the semantics more effectively than random test cases.

Impact of the Number of Test Cases. We study how the number of test cases impacts the effectiveness. We enumerate the number of distinguishing or random positive/negative string examples from $T = 1$ to $T = 10$ to show the impact on the effectiveness ($T = 0$ refers to using MLE to train the model). As shown in Figure 4, when more distinguishing test cases are used, higher accuracy is reached. However, more random test cases make limited improvement.

5 Related Work

Program Synthesis. Our work falls into the general topic of program synthesis. Program synthesis is the problem of automatically generating programs from high-level specifications (Gulwani et al., 2017). There has been a lot of progress made in this area, classified based on (1) the form of specifications, e.g., NL descriptions (Yin and Neubig, 2017; Guu et al., 2017; Lin et al., 2017; Krishnamurthy and Mitchell, 2012; Liu et al., 2018), input-output examples (Balog et al., 2017; Chen et al., 2018; Kalyan et al., 2018), and hybrid of the two preceding types of specifications (Manshadi et al., 2013; Raza et al., 2015); (2) the programming languages, e.g., LISP (Biermann, 1978), Python (Yin and Neubig, 2017; Rabinovich et al., 2017), SQL (Zhong et al., 2017; Sun et al., 2018), and Domain-Specific Languages (DSL) such as FlashFill (Gulwani, 2011). In this paper, we focus on an important subtask of the program-synthesis problem: generating regular expressions from NL.

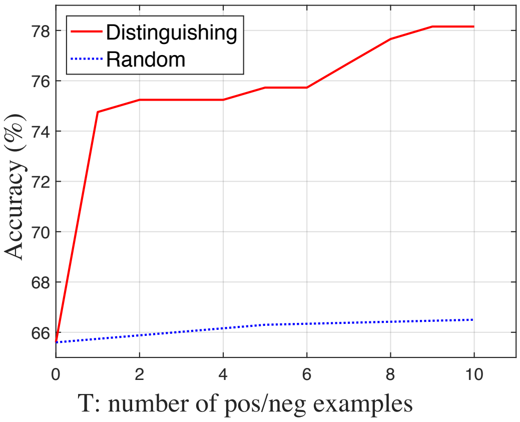

Figure 4: Impact of the number of distinguishing or random test cases on accuracy on the KB13 dataset.

Generating Regular Expressions. Recent research has attempted to automatically generate regular expressions from NL specifications. Ranta (1998) propose a rule-based approach to build an NL interface for regular expressions. Kushman and Barzilay (2013) develop an approach for learning a probabilistic grammar model to parse an NL description into a regular expression. Locascio et al. (2016) regard the problem as a black-box task of machine translation, and train a sequence-to-sequence learning model to address the problem. There exists also a lot of work focusing on generating regular expressions from string examples. Recent work typically uses an evolutionary algorithm to address the problem (Svingen, 1998; Cetinkaya, 2007; Bartoli et al., 2012, 2016).

Inspired by our previous study (Zhong et al., 2018), in this paper, we leverage the help of string examples generated from ground truths to improve the state of the art for the problem of generating regular expressions from NL. Compared with previous state-of-the-art approaches (Locascio et al., 2016) that maximize the likelihood of ground truths in the training set, SemRegex leverages the policy-gradient method to encourage the model to generate semantically correct regular expressions.

Generating Test Cases. When SemRegex is applied on domains other than synthesizing regular expressions, a perfect equivalence oracle such as using the minimal DFA may not be feasibly available. In order to handle a more general case, we propose to generate test cases from the ground truths to measure the semantic correctness of a program candidate. State-of-the-art test-generation techniques are typically based on Dynamic Symbolic Execution (DSE) (Godefroid et al., 2005). Given a program that we want to generate test cases for, DSE executes the program for some seed test cases, and at the same time collects symbolic constraints from branch statements along the execution path. Then DSE generates new test cases to cover different branches in iterations by flipping a branching node in previous execution path. In this way, DSE is able to generate test cases that can be used to approximately check the semantic equivalence. Furthermore, DSE can effectively generate distinguishing test cases for two executable programs by relating these two programs in a single execution (Taneja and Xie, 2008). Various DSE tools have been implemented for different programming languages, such as PyExZ3 (Python) (Ball and Daniel, 2015), JPF-SE (Java) (Anand et al., 2007), Pex (C#) (Tillmann and De Halleux, 2008; Tillmann et al., 2014; Li et al., 2009), and CUTE (C) (Sen et al., 2005).

6 Conclusion

We have proposed SemRegex, a semantics-based approach to generate regular expressions from NL specifications. SemRegex trains a sequence-to-sequence model by maximizing the expected semantic correctness. We measure the semantic correctness using the DFA-equivalence oracle, random test cases, and distinguishing test cases. Our evaluation results show that SemRegex outperforms the existing start-of-the-art approaches on three public datasets.

Acknowledgments

The work from the authors at the University of Illinois at Urbana-Champaign was supported in part by National Science Foundation under grants no. CNS-1513939, CNS-1564274, and CCF-1816615. The work from the authors at Xi'an Jiaotong University was supported by National Natural Science Foundation of China (61632015, 61772408, 61721002).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467.

Saswat Anand, Corina S Păsăreanu, and Willem Visser. 2007. JPF–SE: A symbolic execution extension to Java PathFinder. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 134–138.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Thomas Ball and Jakub Daniel. 2015. Deconstructing dynamic symbolic execution. Technical Report MSR-TR-2015-95, Microsoft.

Matej Balog, Alexander L Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2017. DeepCoder: Learning to write programs. In *International Conference on Learning Representations*.

Alberto Bartoli, Giorgio Davanzo, Andrea De Lorenzo, Marco Mauri, Eric Medvet, and Enrico Sorio. 2012. Automatic generation of regular expressions from examples with genetic programming. In *Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1477–1478.

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. 2016. Inference of regular expressions for text extraction from examples. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1217–1230.

Alan W Biermann. 1978. The inference of regular LISP programs from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(8):585–600.

Rudy Bunel, Matthew Hausknecht, Jacob Devlin, Rishabh Singh, and Pushmeet Kohli. 2018. Leveraging grammar and reinforcement learning for neural program synthesis. In *International Conference on Learning Representations*.

Ahmet Cetinkaya. 2007. Regular expression generation through grammatical evolution. In *Annual Conference Companion on Genetic and Evolutionary Computation*, pages 2643–2646.

Xinyun Chen, Chang Liu, and Dawn Song. 2018. Towards synthesizing complex programs from input-output examples. In *International Conference on Learning Representations*.

Patrice Godefroid, Nils Klarlund, and Koushik Sen. 2005. DART: directed automated random testing. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 213–223.

Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. In *ACM SIGPLAN Symposium on Principles of Programming Languages*, pages 317–330.

Sumit Gulwani, Alex Polozov, and Rishabh Singh. 2017. Program synthesis. *Foundations and Trends in Programming Languages*, 4(1–2):1–119.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Annual Meeting of the Association for Computational Linguistics*, pages 1051–1062.

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

John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. 1979. *Introduction to automata theory, languages, and computation*. Addison-wesley Reading.

Ashwin Kalyan, Abhishek Mohta, Alex Polozov, Dhruv Batra, Prateek Jain, and Sumit Gulwani. 2018. Neural-guided deductive search for real-time program synthesis from examples. In *International Conference on Learning Representations*.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 826–836.

Nuo Li, Tao Xie, Nikolai Tillmann, Jonathan de Halleux, and Wolfram Schulte. 2009. Reggae: Automated test generation for programs using complex regular expressions. In *IEEE/ACM International Conference on Automated Software Engineering*.

Xi Victoria Lin, Chenglong Wang, Deric Pang, Kevin Vu, Luke Zettlemoyer, and Michael D. Ernst. 2017. Program synthesis from natural language using recurrent neural networks. Technical Report UW-CSE-17-03-01, University of Washington Department of Computer Science and Engineering.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*.

Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Conference on Empirical Methods in Natural Language Processing*, pages 1918–1923.

Mehdi Hafezi Manshadi, Daniel Gildea, and James F Allen. 2013. Integrating programming by example and natural language programming. In *AAAI Conference on Artificial Intelligence*, pages 661–667.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Annual Meeting of the Association for Computational Linguistics*, pages 1139–1149.

Aarne Ranta. 1998. A multilingual natural-language interface to regular expressions. In *International Workshop on Finite State Methods in Natural Language Processing*, pages 79–90.

Mohammad Raza, Sumit Gulwani, and Natasa Milic-Frayling. 2015. Compositional program synthesis from natural language and examples. In *International Joint Conferences on Artificial Intelligence*, pages 792–800.

Koushik Sen, Darko Marinov, and Gul Agha. 2005. CUTE: a concolic unit testing engine for C. In *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 263–272.

Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic parsing with syntax- and table-aware SQL generation. In *Annual Meeting of the Association for Computational Linguistics*, pages 361–372.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Annual Conference on Neural Information Processing Systems*, pages 3104–3112.

Borge Svingen. 1998. Learning regular languages using genetic programming. *Genetic Programming*, pages 374–376.

Kunal Taneja and Tao Xie. 2008. DiffGen: Automated regression unit-test generation. In *IEEE/ACM International Conference on Automated Software Engineering*, pages 407–410.

Nikolai Tillmann and Jonathan De Halleux. 2008. Pex–White box test generation for .NET. In *International Conference on Tests and Proofs*, pages 134–153.

Nikolai Tillmann, Jonathan de Halleux, and Tao Xie. 2014. Transferring an automated test generation tool to practice: From Pex to Fakes and Code Digger. In *IEEE/ACM International Conference on Automated Software Engineering*, pages 385–396.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Annual Meeting of the Association for Computational Linguistics*, pages 440–450.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Zexuan Zhong, Jiaqi Guo, Wei Yang, Tao Xie, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2018. Generating regular expressions from natural language specifications: Are we there yet? In *Workshop on NLP for Software Engineering*.

Decoupling Structure and Lexicon for Zero-Shot Semantic Parsing

Jonathan Herzig[1] and **Jonathan Berant**[1,2]

[1]Tel-Aviv University
[2]Allen Institute for Artificial Intelligence
`jonathan.herzig@cs.tau.ac.il, joberant@cs.tau.ac.il`

Abstract

Building a semantic parser quickly in a new domain is a fundamental challenge for conversational interfaces, as current semantic parsers require expensive supervision and lack the ability to generalize to new domains. In this paper, we introduce a zero-shot approach to semantic parsing that can parse utterances in unseen domains while only being trained on examples in other source domains. First, we map an utterance to an abstract, domain-independent, logical form that represents the structure of the logical form, but contains slots instead of KB constants. Then, we replace slots with KB constants via lexical alignment scores and global inference. Our model reaches an average accuracy of 53.4% on 7 domains in the OVERNIGHT dataset, substantially better than other zero-shot baselines, and performs as good as a parser trained on over 30% of the target domain examples.

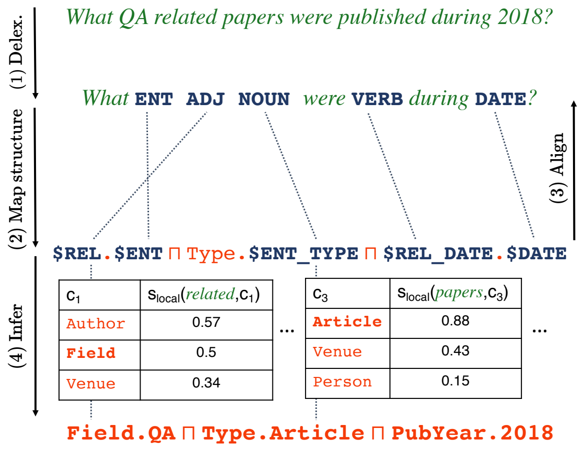

Figure 1: A test utterance is delexicalized (1) and mapped to its abstract logical form (2). Slots ("\$" variables) are then aligned to the abstract utterance (3), and are filled with the top assignment in terms of local and global scores (4). Logical forms throughout this paper are in λ-DCS (Liang, 2013).

1 Introduction

Semantic parsing, the task of mapping natural language utterances into executable logical forms, is a key paradigm in developing conversational interfaces (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2011; Berant and Liang, 2015). The recent success of conversational interfaces such as Amazon Alexa, Google Assistant, Apple Siri, and Microsoft Cortana has led to soaring interest in developing methodologies for training semantic parsers quickly in any new domain and from little data.

Prior work focused on alleviating data collection by training from weak supervision (Clarke et al., 2010; Liang et al., 2011; Kwiatkowski et al., 2013; Artzi and Zettlemoyer, 2013), or developing protocols for fast data collection through paraphrasing (Berant and Liang, 2014; Wang et al., 2015) or a human-in-the-loop (Iyer et al., 2017). However, all these approaches rely on supervised training data in the target domain and ignore data collected previously for other domains.

In this paper, we propose an alternative, zero-shot approach to semantic parsing, where no labeled or unlabeled examples are provided in the target domain, but annotated examples from other domains are available. This is a challenging setup as in semantic parsing each dataset is associated with its own knowledge-base (KB) and thus all target domain KB constants (relations and entities) are unobserved at training time. Moreover, this is a natural use-case as more and more conversational interfaces are developed in multiple domains.

Our approach is motivated by recent work (Herzig and Berant, 2017; Su and Yan, 2017; Fan et al., 2017; Richardson et al., 2018) that showed that while the lexicon and KB constants in dif-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1619–1629
Brussels, Belgium, October 31 - November 4, 2018.

ferent domains vary, the structure of language composition repeats across domains. Therefore, we propose that by abstracting away the domain-specific lexical items of an utterance, we can learn to map the structure of an abstract utterance to an abstract logical form that does not include any domain-specific KB constants, using data from other domains only.

Figure 1 illustrates this approach. A test utterance in the target domain is delexicalized and mapped to an abstract, domain-independent representation, where some content words are replaced by abstract tokens (step 1). Then, a structure-mapping model maps this representation into an abstract logical form that contains slots instead of KB constants (step 2). A major technical challenge at this point is to replace slots in the abstract logical form with KB constants from the target domain. We show that it is possible to learn a domain-independent lexical alignment model that aligns each slot to a word in the original utterance (step 3). This alignment, combined with a global inference procedure (step 4) allows one to find the best assignment of KB constants and produce a final logical form. Importantly, both of our models are trained from data in *other* domains only.

We show that our zero-shot framework parses 7 different unseen domains from the OVERNIGHT dataset with an average denotation accuracy of 53.4%. This result dramatically outperforms several natural baselines, and achieves the same result as training a parser on over 30% of the fully supervised target domain examples. To our knowledge, this work is the first to train a zero-shot semantic parser that can handle unseen domains. All our code is available at `https://github.com/jonathanherzig/zero-shot-semantic-parsing`.

2 Background

Neural Semantic Parsing Sequence-to-sequence models (Sutskever et al., 2014) were recently proposed for semantic parsing (Jia and Liang, 2016; Dong and Lapata, 2016). In this setting, a sequence of input language tokens $x_1, \ldots, x_m$ is mapped to a sequence of output logical tokens $z_1, \ldots, z_n$. We briefly review the model by Jia and Liang (2016), which we use as part of our framework, and also as a baseline.

The **encoder** is a BiLSTM (Hochreiter and Schmidhuber, 1997) that converts $x_1, \ldots, x_m$ into a sequence of context sensitive states. The attention-based **decoder** (Bahdanau et al., 2015; Luong et al., 2015) is an LSTM language model additionally conditioned on the encoder states. Formally, the decoder is defined by:

$$p(z_j = w \mid x, z_{1:j-1}) \propto \exp(U[s_j, c_j]),$$
$$s_{j+1} = LSTM([\phi^{(out)}(z_j), c_j], s_j),$$

where s_j are decoder states, U and the embedding function $\phi^{(out)}$ are the decoder parameters, and the context vector, c_j, is the result of global attention (Luong et al., 2015). We also employ attention-based copying (Jia and Liang, 2016), but omit details for brevity.

Semantic Parsing over Multiple KBs Recently, Herzig and Berant (2017), Su and Yan (2017) and Fan et al. (2017) proposed to exploit structural regularities in language across different domains. These works pooled together examples from multiple datasets in different domains, each corresponding to a separate KB, and trained a single sequence-to-sequence model over all examples, sharing parameters across domains. They showed that this substantially improves parsing accuracy. While these works implicitly capture linguistic regularities across domains, they rely on annotated data in the target domain. We, conversely, explicitly decouple structure mapping from the assignment of KB constants, and thus can tackle the zero-shot setting where no target domain examples are available. This is the focus of the next section.

3 Zero-Shot Semantic Parsing

3.1 Overview

Following the empirical success of sharing structural information between different semantic parsing domains, we propose in this paper to take a more radical approach and to explicitly decouple semantic parsing into a structure mapping model and a lexicon mapping model. We now provide an overview of our approach and explain how this decoupling facilitates zero-shot semantic parsing.

We assume access to D different source domains, where for every domain d we receive a KB $\mathcal{K}_d$, and a training set of pairs of utterances and logical forms $\{(x_i, z_i)\}_{i=1}^{N_d}$. We further assume a lexicon L that maps each KB constant in $\mathcal{K}_d$ to a short phrase that describes it (e.g., $L($`PubYear`$)\rightarrow$*"publication year"*), as in Wang

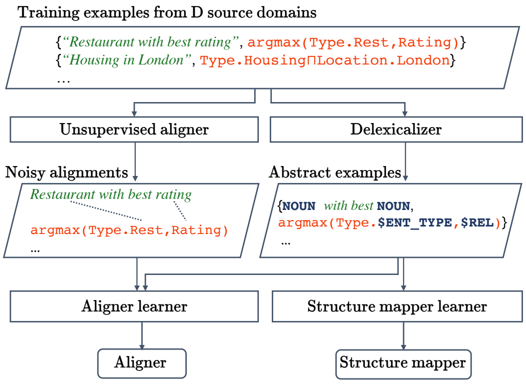

Figure 2: Training flow. Examples in source domains are delexicalized. Abstract examples are used to train both the structure mapper learner and aligner learner, where the aligner learner uses noisy alignments as labels.

Lexical representation
"What meetings have no more than 3 attendees?"
$\texttt{Type.Meeting} \sqcap \mathbf{R}[\lambda x.\text{count}(\texttt{Attendee}.x)].\leq.\texttt{3}$
"Which recipe needs no more than two ingredients?"
$\texttt{Type.Recipe} \sqcap \mathbf{R}[\lambda x.\text{count}(\texttt{IngredientOf}.x)].\leq.\texttt{2}$

Abstract representation
"What NOUN have no more than NUM NOUN?"
$\texttt{Type.\$ENT\_TYPE} \sqcap \mathbf{R}[\lambda x.\text{count}(\texttt{\$REL}.x)].\leq.\texttt{\$NUM}$
"Which NOUN VERB no more than NUM NOUN."
$\texttt{Type.\$ENT\_TYPE} \sqcap \mathbf{R}[\lambda x.\text{count}(\texttt{\$REL}.x)].\leq.\texttt{\$NUM}$

Figure 3: Examples in different domains (CALENDAR and RECIPES) in their original and abstract representations. A similar structural regularity (a comparative structure) maps to an identical abstract logical form.

et al. (2015). Finally, we assume a pre-trained, static, embedding function $\phi(w) \in \mathbb{R}^f$ for every word w, used to measure cross-domain lexical semantic similarity. Our goal is to train a semantic parser that maps a new utterance x to the correct logical form z from a new domain d_{new} given $\mathcal{K}_{d_{\text{new}}}$.

Figure 2 describes the flow of our training procedure: we first employ a simple rule-based method to transform training examples to an abstract representation, where content words (in utterances) and KB constants (in logical forms) are delexicalized. We then train the following two models that decouple structure from lexicon (a) The structure mapper that maps abstract utterances to abstract logical forms. (b) The aligner that provides an alignment from abstract logical form tokens to abstract utterance tokens. Training the aligner is challenging because no gold alignments between the abstract utterance and abstract logical form are available. To overcome this challenge we propose a distillation strategy: we obtain noisy supervision by training a state-of-the-art unsupervised alignment model on the D source domains. Then, we train a second supervised alignment model that receives abstract utterances, abstract logical forms, and target noisy alignments as input and learns to predict the noisy alignments.

Once the two models are trained, we can tackle a new domain without training examples (Figure 1). Given an utterance from the target domain, we first abstract it using the delexicalizer, and then predict its abstract structure using the structure mapper. We treat delexicalized logical form tokens as slots to be filled with KB constants. Candidate assignments are then scored locally according to the semantic similarity of a KB constant (represented by its entry value in the lexicon L) to words the slot aligns to according to the aligner. For this we use the pre-trained embedding function $\phi(\cdot)$ as the only cross-domain information. Finally, we choose a final assignment of KB constants by exactly maximizing a global scoring function, which takes into account both local alignment scores as well as global constraints.

We next describe in detail the four components of our framework: the delexicalizer, structure mapper, aligner, and inference procedure.

3.2 Delexicalizer

The goal of the delexicalizer is to strip utterances and logical forms from their domain-specific components and preserve domain-independent parts. We note that it is possible that some words contain both domain-specific and domain-general aspects (*"cheapest"*). However, we conjecture that it is possible to decompose examples in a manner that enables zero-shot semantic parsing.

The output of the delexicalizer is an abstract representation that should manifest structural linguistic regularities across domains (Figure 3). For example, a comparative structure will correspond to the same abstract logical form in different domains. In this representation, used as input to our models, content words and KB constants are transformed to an abstract type. This rule-based preprocessing step is applied to all D source domain training examples (utterances and logical forms), and to target domain utterances at test time. We now describe the process of delexicalization.

Source	Category	Abstract Type	Examples
Utterance	Noun	`NOUN`	*“cuisines”*, *“housing”*, *“time”*
	Verb	`VERB`	*“published”*, *“born”*, *“posted”*
	Adjective	`ADJ`	*“high”*, *“cooking”*, *“monthly”*
	Number	`NUM`	*“4”*, *“three”*
	Date	`DATE`	*“2018”*, *“january 2nd”*
	Entity	`ENT`	*“midtown”*, *“alice”*, *“dinner”*
Logical Form	Number	`$NUM`	`1,2,3`
	Date	`$DATE`	`1_6_2018`
	Entity	`$ENT`	`MidtownWest, CentralOffice`
	Entity type	`$ENT_TYPE`	`Person, Location, Recipe`
	Numerical entity	`$ENT_NUM`	`Rent, Size, CookingTime`
	Binary relation	`$REL`	`Author, Attendee`
	Unary relation	`$REL_UNARY`	`AllowsCats, WonAward`
	Numerical relation	`$REL_NUM`	`Height, Length, StarRating`
	Date relation	`$REL_DATE`	`PostingDate, PubYear`

Table 1: Categories of content words and KB constants, and their corresponding abstract type notation.

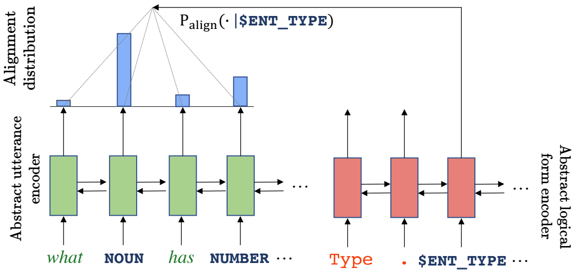

Figure 4: The aligner model. Alignments are derived by comparing the slot hidden state against all utterance hidden states.

Utterances Table 1 describes the full list of abstraction rules. We delexicalize several categories of content words and keep function words, which describe the utterance structure, in their lexicalized form. Specifically, any verb[1] whose lemma is not *“be”* or *“do”* is delexicalized. All nouns are delexicalized, except for a small vocabulary of three words (*“average”*, *“total”*, and *“number”*), which denote a domain-general operation. Adjectives tend to distribute more evenly between domain-specific words and domain-general words, thus discriminating them is harder (e.g., *“outdoor”*, *“wide”* and *“cooking”* are domain-specific words while *“minimum”*, *“same”* and *“many”* are domain-general words). Thus, we take a statistical approach and only delexicalize adjectives that are unique to the domain (i.e., did not appear in the training set of any other source domain). We also delexicalize dates and numbers, and identify entities in the utterance by string matching against the entities in the KB. These are then delexicalized to their corresponding abstract type (Table 1).

Logical Forms We delexicalize all KB constants to their abstract type, which is given as part of the KB schema (Table 1).

3.3 Structure Mapper

As a first step towards predicting the lexical logical form, we map an abstract utterance, to an abstract logical form. The model is the neural semantic parser described in Section 2, only here the input and output are the abstract examples in all D domains, which the delexicalizer outputs. The model utilizes a single encoder-decoder pair shared across all domains. As Figure 3 suggests, the model should learn, e.g., that a noun modified by a wh-question often maps to `$ENT_TYPE`, and that *“no more than”* maps to the $\leq$ operator.

3.4 Aligner

The output of the structure mapper is an abstract logical form that contains slots instead of KB constants. To predict a complete logical form, we must assign a KB constant to each slot.

We observe that the description of a KB constant that appears in the logical form (`Article`) is often semantically similar to some word in the utterance (*“paper”*). Thus, we can obtain signal for the identity of a KB constant by solving an alignment problem: each slot can be aligned to words in the utterance that have similar meaning to that of the gold KB constant. Naturally, in some cases a KB constant is not semantically similar to any utterance word (e.g., the relation `Field` in Figure 1), which we will mediate by using a global inference procedure (Section 3.5).

Thus, our goal is to learn a model that given an abstract utterance-logical form pair $(x^{\text{abs}}, z^{\text{abs}})$ produces an alignment matrix A, where A_{ij} corresponds to the alignment probability $p(x_j^{\text{abs}} \mid z_i^{\text{abs}})$. A central challenge is that no gold alignments are provided in any domain. Therefore, we adopt a “distillation approach”, where we train a supervised model over abstract examples to mimic the predictions of an unsupervised model that has access to the full lexicalized examples.

Specifically, we use a standard unsupervised word aligner (Dyer et al., 2013), which takes all lexicalized examples $\{(x_i, z_i)\}_{i=1}^{N_d}$ in all D domains and produces an Alignment matrix A^* for every example, where $A^*_{ij} = 1$ iff token i in the logical form is aligned to token j in the utterance. Then, we treat A^* as gold alignments and gener-

---

[1] Numbers, dates and part-of-speech tags are extracted using Stanford CoreNLP (Manning et al., 2014).

ate examples $(x^{\text{abs}}, z^{\text{abs}}, A^*)$ to train the aligner. Learning alignments over abstract representations is possible, as a slot in a specific context tends to align to specific types of abstract words (e.g., Figure 3 suggests that a relation that is aggregated, often aligns to the `NOUN` that appears after the `NUM` in the abstract utterance).

We now present our alignment model, depicted in Figure 4. The model uses two different BiLSTMs to encode x^{abs} and z^{abs} to their context sensitive states $b_1, \ldots, b_m$ and $s_1, \ldots, s_n$ respectively. We model the alignment probability $p_{\text{align}}(x_j^{\text{abs}} \mid z_i^{\text{abs}})$ with a bi-linear form similar to attention (Luong et al., 2015):

$$e_{ij} = s_i^T W b_j,$$

$$p_{\text{align}}(x_j^{\text{abs}} \mid z_i^{\text{abs}}) = \frac{\exp(e_{ij})}{\sum_{j'=1}^{m} \exp(e_{ij'})},$$

where the parameters W are learned during training. We train the model to minimize the negative log-likelihood of gold alignments while considering only alignments of slots (since we only align slots at test time). The cross-entropy loss for a training example $(x^{\text{abs}}, z^{\text{abs}}, A^*)$ is then given by:

$$-\sum_{i:i \in S_z}^{n} \sum_{j=1}^{m} A_{ij}^* \log p_{\text{align}}(x_j^{\text{abs}} \mid z_i^{\text{abs}}),$$

where S_z are the slot indices in z^{abs}.

Our model can be viewed as an attention model, dedicated to aligning logical form tokens to utterance tokens. Using a separate alignment model rather than the attention weights of the structure mapper has two advantages: First, alignments are generated given the entire generated sequence z^{abs} rather than just a prefix. Second, our model focuses its capacity on the alignment task without worrying about generation of z^{abs}. In Section 4, we will demonstrate that training a dedicated aligner substantially improves performance.

3.5 Inference

The aligner provides a distribution over utterance tokens for every slot in the abstract logical form. To compute the final logical form, we must replace each slot with a KB constant. Formally, let $(z_{j_1}^{\text{abs}}, \ldots, z_{j_l}^{\text{abs}})$ be the sequence of slots in z^{abs} and denote them for simplicity as $y = (y_1, \ldots, y_l)$. Our goal is to predict a sequence of KB constants $c = (c_1, \ldots, c_l)$, where each c_i is chosen from a candidate set $\mathcal{C}(y_i)$ that is determined by the abstract token y_i according to Table 1 (e.g., if y_i is `$REL`, then $\mathcal{C}(y_i)$ is the set of binary relations).

Our scoring function depends on alignments computed by the aligner. However, because slots are independent in the aligner, we introduce a few global constraints that capture the dependence between different slots. Formally, we wish to find c^* that maximizes the following scoring function, which depends on the utterance x, the slot sequence y, the abstract logical form z^{abs}, the alignment matrix A and the embedding function ϕ:

$$\arg\max_{c} \sum_{i=1}^{l} \left(s_{\text{local}}(c_i, y_i, x, A, \phi)\right) + s_{\text{global}}(c, z^{\text{abs}}).$$

We now describe our scoring functions in detail.

Local Score Because inference is applied only at test time, we have access to the lexicalized utterance and not only the abstract one. Thus, the aligner outputs a distribution over words for a slot y (e.g., in Figure 1, `$REL_DATE` aligns with high probability to `VERB`, which corresponds to the word *"published"*). Each word, in turn, has different semantic similarity to each KB constant in $\mathcal{C}(y)$. Intuitively, we would like to assign a KB constant that has high similarity with words the slot is aligned to. Thus, we define s_{local} of a KB constant c_i for every slot y_i to be its expected semantic similarity under the alignment distribution:

$$s_{\text{local}}(c_i, y_i, x, A, \phi) = E_{x \sim p_{\text{align}}(x^{\text{abs}} \mid y_i)}[\mathit{sim}_\phi(x, c_i)]$$

$$= \sum_{j=1}^{m} p_{\text{align}}(x_j^{\text{abs}} \mid y_i) \cdot \mathit{sim}_\phi(x_j, c_i).$$

We define $\mathit{sim}_\phi(x_j, c_i)$ to be the cosine similarity between the embedding $\phi(x_j)$ and the embedding $\phi(c_i)$ (scaled to the range [0,1]), where $\phi(c_i)$ is defined to be the average embedding of all words in $L(c_i)$, that is, $\mathit{sim}(x_j, c_i) = \frac{1+\cos(\phi(x_j), \phi(c_i))}{2}$.

Global Score Utilizing only a local scoring function raises several concerns. First, slots are treated independently and dependencies between slots are ignored, which might result in a final logical form that is globally inconsistent. For example, we could generate the logical form `Birthplace.ComputerScience`, which is semantically dubious. Second, some KB constants do not align to any word in the utterance and appear in the logical form only implicitly. For example, the logical form in Figure 1 contains the `Field` relation, however *"field"* is implicit in the utterance. Therefore, we define $exe_{\mathcal{K}}(z)$ to be true iff z executes against $\mathcal{K}$ without errors, and define a global score that prevents assignments c that result in a logical form z such that $exe_{\mathcal{K}}(z)$ is false.

Moreover, we can use similar constraints to prevent logical forms that are highly unlikely according to our prior knowledge. Specifically, we define $once(z)$ to be true iff each date, named entity, and number in the logical form z appear exactly once. We then define a global score that prevents logical forms in which $once(z)$ is false. Empirically, we find such assignments to be mostly wrong (e.g., `Type.Article ⊓ (Field.QA ⊔ Field.QA)`).

Formally, our scoring function is defined as:

$$s_{\text{global}}(c, z^{abs}) = \begin{cases} 0 & exe_{\mathcal{K}}(z^{abs}|_c), once(z^{\text{abs}}|_c) \\ -\infty & \text{otherwise,} \end{cases}$$

where $z^{abs}|_c$ is the result of assigning the KB constants c to the slots in z^{abs}.

Inference Algorithm. While each local scoring function can be efficiently maximized independently, the global constraints that depend on the entire assignment c make inference more complicated. However, because the global scoring function introduces hard constraints, an exact and efficient inference algorithm is still possible. Our inference algorithm generates solutions one-by-one sorted by the local scoring function only. Then, it checks for each one whether it satisfies the global constraints defined by s_{global}, and stops once a satisfying solution is found, which is guaranteed to maximize our scoring function. While in the worst case, this procedure is exponential in the size of c, in practice solutions are found after only a few steps. We also always halt after T steps if a solution has not been found.

Algorithm 1 describes the details of our inference procedure. We define *cands* to be a data structure that contains l lists of candidate KB constants (a list for each slot), sorted according to the local scoring function s_{local} in descending order. Additionally, *getAssign*$(cands, a)$ is a function that accesses *cands*, and retrieves the assignment with indices a. For example, *getAssign*$(cands, \{0\}^l)$ retrieves the top scoring local assignment. Last, we define $a_{\text{inc}(i)}$ to be the indices a, where a_i is incremented by 1.

Algorithm 1 Exact inference algorithm

```
Input: $cands, T$
Output: $c^*$ - the top scoring assignment
$horizon \leftarrow \emptyset$                    ▷ Max heap
$a_{\text{init}} \leftarrow \{0\}^l$
$push(horizon, a_{\text{init}})$
for $t \leftarrow 1$ to $T$ do
    $a \leftarrow pop(horizon)$
    $c \leftarrow getAssign(cands, a)$
    if $s_{\text{global}}(c, z^{\text{abs}}) = 0$ then
        return $c$
    for $i \leftarrow 1$ to $l$ do
        $push(horizon, a_{\text{inc}(i)})$
return $NULL$
```

The algorithm proceeds as follows. First we initialize a maximum heap *horizon* into which we will dynamically push candidate assignments. Then, we iteratively pop the best current assignment from the heap, and check if it satisfies the global constraints. If it does, we return this assignment and stop. Otherwise, we generate the next possible candidates, one from each list (there is no need to add more than one because candidates are sorted). If no satisfying assignment is found after T steps, we return $NULL$. It is easy to show that when the algorithm returns an assignment it is guaranteed to be the one that maximizes our global scoring function.

4 Experiments

4.1 Experimental Setup

Data We evaluated our method on the OVERNIGHT semantic parsing dataset, which contains $13,682$ examples of language utterances paired with logical forms across eight domains, which were chosen to explore diverse types of language phenomena. As described, our approach depends on having linguistic regularities repeat across domains. However two domains contain logical forms that are based on neo-davidsonian semantics for treating events with multiple arguments. Since such logical forms are completely absent in six domains, it is not possible for our method to generalize to those in our zero-shot approach. Therefore, we do not evaluate on the BASKETBALL domain, in which 98% of the examples contain such logical forms, and omit all examples (68%) that contain such logical forms in the SOCIAL domain. We evaluated on the same train/test split as Wang et al. (2015), using the same accuracy metric, i.e., the proportion of questions for which the denotations of the

Model	Blocks	Calendar	Housing	Publications	Recipes	Restaurants	Social	Avg.
InLex	59.9	73.8	72.0	79.5	79.2	76.2	83.4	74.8
InAbstract	39.6	57.7	51.9	59.6	66.2	68.4	66.1	58.5
CrossLex	0.0	0.0	0.5	0.0	1.4	1.2	1.0	0.6
CrossLexRep	6.5	1.8	2.6	1.9	16.7	6.6	1.7	5.4
ZeroShot	28.3	53.6	52.4	55.3	60.2	61.7	62.4	53.4

Table 2: Test accuracy for all models on all domains.

	In-domain	**Cross-domain**
Lexical	InLex	CrossLex
Abstract	InAbstract	ZeroShot

Table 3: Evaluated models.

predicted and gold logical forms are equal. We additionally used the lexicon L they provided with descriptions for KB constants.

Evaluated Models We evaluated different models (Table 3) according to the following two attributes. Firstly, whether the model is trained on target domain data (*in-domain*) or on source domains data only (*cross-domain*). Secondly, we trained the neural semantic parser described in Section 2 over the lexical data representation (*lexical*), or in comparison trained our model over the abstract representation (*abstract*).

As CrossLex can not generate KB constants unseen during training, we additionally implemented CrossLexRep. In this model, we added an additional step that modifies the output of CrossLex: we replaced a generated KB constant with its most similar KB constant from the target KB that also shares its abstract type.

Implementation Details In all experiments, for our embedding function $\phi(\cdot)$, we used pre-trained GloVe (Pennington et al., 2014) vectors with dimension 300. In a single experiment we considered one domain as the target domain, while other domains were the source domains (and repeated for all domains). For InLex, CrossLex and CrossLexRep we used exactly the same experimental setup as Jia and Liang (2016). For our zero-shot model, we used 20% of the training data as a development set for tuning hyper-parameters. We first tuned parameters for the structure mapper, and used the best setting for tuning the aligner.

We provide the list of hyper-parameters and their values for our zero-shot framework. **Structure mapper:** number of epochs (22, using early stopping), hidden unit dimension (300), word vector dimension (100), learning rate (0.1 with SGD optimizer), $L2$ regularization (0.001). At test time, we used beam search with beam size 5, and then picked the highest-scoring logical form that we could infer an assignment for. **Aligner:** number of epochs (30, using early stopping), hidden unit dimension (250), word vector dimension (100), learning rate (0.0002 with Adam optimizer), dropout rate over hidden states (0.4). For both models, word vectors are updated during training. **Inference:** we used $T = 500$ steps, after which we halted.

4.2 Results

We trained all models above and evaluated on the test set for all seven domains. Results show (Table 2) that ZeroShot substantially outperforms other zero-shot baselines. CrossLex performs poorly, as it can only generate KB constants seen during training. CrossLexRep performs better, as it can generate KB constants from the target domain, however, generating the correct constant usually fails. This highlights the challenge in the zero-shot semantic parsing setting.

For baselines trained on target domain data, InLex (re-implementation of (Jia and Liang, 2016)) achieved average accuracy of 74.8, which is comparable to the 74.4 average accuracy they report on our seven domains. Training on the target domain with our method InAbstract achieved 58.5% average accuracy, which shows that while the abstract representation in our framework loses some valuable information, it is still successful. Importantly, the performance of ZeroShot (53.4%) is only slightly lower than InAbstract, showing that our model degrades gracefully and generalizes well across domains compared to CrossLex.

Model Ablations We now measure the effect of different components of our framework on denotation accuracy. We examined the effect of removing components completely, or replacing them with simpler ones. Thus, the following ablated models can be viewed as additional baselines.

Model	Blocks	Calendar	Housing	Publications	Recipes	Restaurants	Social	Avg.
ZEROSHOT	29.2	60.4	57.3	47.7	59.5	63.0	64.0	54.5
-GLOBALHEUR	29.2	60.4	56.7	46.9	60.1	46.0	62.2	51.6
-ALIGNER	22.6	57.5	52.7	46.9	58.4	43.0	60.4	48.8
-INFERENCE	28.2	51.5	44.7	43.0	42.2	44.5	45.8	42.8
-ALIGNER,INFERENCE	20.4	38.8	32.7	37.5	37.0	15.5	36.9	31.2

Table 4: Development accuracy for all ablations.

1. -ALIGNER: Replacing the alignment distribution from the aligner with alignment distribution from the decoder of the structure mapper.
2. -INFERENCE: Discarding the global scoring function and maximizing each slot independently.
3. -ALIGNER,INFERENCE: Discarding both of our main technical contributions.
4. -GLOBALHEUR: Discarding the global inference heuristics (denoted as $once(z)$).

Table 4 shows that ablating each of the components hurts performance. Discarding our two main technical contributions results in 31.2% accuracy compared to 54.5% in the full model. Performing inference with global constraints dramatically improve performance, showing that using the alignment model alone results often in incoherent logical forms. Our dedicated aligner also improves performance compared to alignments learned by the decoder of the structure mapper. This is pronounced without global constraints (a drop from 42.8% to 31.2%), but is less severe when global inference is used (a drop from 54.5% to 48.8%).

Intrinsic Analysis While we evaluated performance above via denotation accuracy, we now evaluate our framework's modules with different metrics (on the development set). We evaluated the structure mapper by measuring the exact match of the top candidate in the beam to the gold abstract logical form (49.1%). We further evaluated the aligner by measuring alignment accuracy for top candidate alignments, in comparison to the unsupervised aligner output (72.9%).

Finally, we measured inference performance in the following ways. The fraction of cases where inference succeeded within T steps is 70% (as some predicted abstract logical forms are not valid in terms of their syntax), and the average number of steps in case of success (3.67 steps). In addition, the fraction of correct global assignments given an abstract logical form that exactly matches the gold one is 77.0%. To conclude, results show that the structure mapping problem is harder than slot filling, for which we learned good alignments and performed fast and mostly accurate inference.

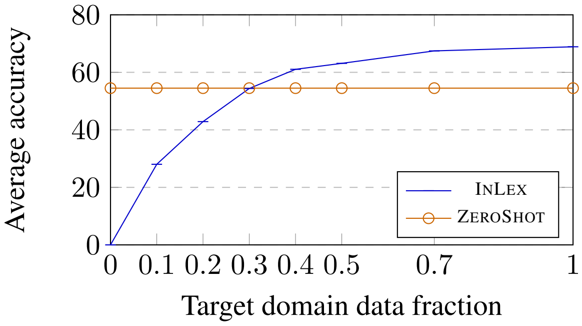

Figure 5: Learning curve for INLEX, compared to ZEROSHOT average performance.

Valuation To estimate the value of our zero-shot framework in terms of target domain examples, we plot a learning curve (Figure 5) that shows development set average accuracy for INLEX (trained on target domain data). In comparison, ZEROSHOT utilizes no target domain data, thus it is fixed. As Figure 5 shows, our framework's value is equal to 30% of the target domain training data. In our setting this equals to 400 examples manually-annotated with full logical forms. Note that this value is gained every time a semantic parser for a new domain is needed. Moreover, our parser can be used as an initial system, deployed to begin training from user interaction directly.

Limitations We now outline some of the limitations of our approach for zero-shot semantic parsing. We hypothesized that language regularities repeat across domains, however as mentioned above, neo-davidsonian semantics occurs mostly in one domain in the OVERNIGHT dataset and thus we were not able to generalize to it. Our parser also obtained low accuracy in BLOCKS. This domain contains mostly spatial language, different from other domains in OVERNIGHT. Specifically, prepositions, which we did not lexicalize map to relations in the KB (e.g., *"below"* and *"above"* map to the relations `Below` and `Above`). This shows the challenge involved in decomposing the structure from the lexicon with rules. In addition, since some spatial relations in this domain are semantically similar (`Length`, `Width` and `Height`), we found it hard to rank them correctly during inference. This stresses that in our framework, we assume KB constants to be sufficiently distinguishable in the pre-trained embedding space, which is not always the case.

5 Related Work

While zero-shot executable semantic parsing is still under-explored, some works focused on the open-vocabulary setting which handles unseen relations by replacing a formal KB with a probabilistic database learned from a text corpus (Choi et al., 2015; Gardner and Krishnamurthy, 2017).

Our abstract utterance representation is related to other attempts to generate intermediate representations that improve generalization such as dependency trees (Reddy et al., 2016), syntactic CCG parses (Krishnamurthy and Mitchell, 2015), abstract templates (Abujabal et al., 2017; Goldman et al., 2018) or masked enitites (Dong and Lapata, 2016). Our abstract logical form representation is similar to that Dong and Lapata (2018) used in to guide the decoding of the full logical form. The main difference with our work is that we focus on a comprehensive abstract representation tailored for zero-shot semantic parsing.

It is worth mentioning other work that inspected various aspects of zero-shot parsing. Bapna et al. (2017) focused on frame semantic parsing, and assumed that relations appear across different domains to learn a better mapping in the target domain. Also in frame semantic parsing, Ferreira et al. (2015) utilized word embeddings to map words to unseen KB relations. Finally, Lake and Baroni (2017) inspected whether neural semantic parsers can handle types of compositionality that were unseen during training. The main difference between their work and ours is that we focus on a scenario where a compositional logical form is generated, but the target KB constants do not appear in any of the source domains.

6 Conclusion

In this paper we address the challenge of zero-shot semantic parsing. We introduce a model that can parse utterances in unseen domains by decoupling structure mapping from lexicon mapping, and demonstrate its success on 7 domains from the OVERNIGHT dataset.

In future work, we would like to automatically learn a delexicalizer from data, tackle zero-shot parsing when the structure distribution in the target domain is very different from the source domains, and apply our framework to datasets where only denotations are provided.

Acknowledgments

We thank Kyle Richardson, Vivek Srikumar and the anonymous reviewers for their constructive feedback. This work was completed in partial fulfillment for the PhD degree of the first author. Herzig was supported by a Google PhD fellowship. This research was partially supported by The Israel Science Foundation grant 942/16 and The Blavatnik Computer Science Research Fund.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th international conference on world wide web*, pages 1191–1200. International World Wide Web Conferences Steering Committee.

Y. Artzi and L. Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)*, 1:49–62.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero shot frame semantic parsing for domain scaling. In *Interspeech 2017*.

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

J. Berant and P. Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics (TACL)*, 3:545–558.

Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. 2015. Scalable semantic parsing with partial ontologies. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference* *on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1311–1320.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*, pages 18–27.

L. Dong and M. Lapata. 2016. Language to logical form with neural attention. In *Association for Computational Linguistics (ACL)*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 48–56, Vancouver, Canada. Association for Computational Linguistics.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015. Zero-shot semantic parser for spoken language understanding. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Matt Gardner and Jayant Krishnamurthy. 2017. Open-vocabulary semantic parsing with both distributional statistics and formal knowledge. In *AAAI*, pages 3195–3201.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. Weakly supervised semantic parsing with abstract examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1809–1819. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 623–628. Association for Computational Linguistics.

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

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

Jayant Krishnamurthy and Tom M Mitchell. 2015. Learning a compositional semantics for freebase with an open predicate vocabulary. *Transactions of the Association for Computational Linguistics*, 3:257–270.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1512–1523.

Brenden M Lake and Marco Baroni. 2017. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv preprint arXiv:1711.00350*.

P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv*.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

M. Luong, H. Pham, and C. D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421.

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The stanford coreNLP natural language processing toolkit. In *ACL system demonstrations*.

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Kyle Richardson, Jonathan Berant, and Jonas Kuhn. 2018. Polyglot semantic parsing in apis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 720–730. Association for Computational Linguistics.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1235–1246, Copenhagen, Denmark. Association for Computational Linguistics.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Y. Wang, J. Berant, and P. Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

A Span Selection Model for Semantic Role Labeling

Hiroki Ouchi[2,3] **Hiroyuki Shindo**[1,2] **Yuji Matsumoto**[1,2]
[1] Nara Institute of Science and Technology
[2] RIKEN Center for Advanced Intelligence Project (AIP)
[3] Tohoku University
hiroki.ouchi@riken.jp, { shindo, matsu }@is.naist.jp

Abstract

We present a simple and accurate span-based model for semantic role labeling (SRL). Our model directly takes into account all possible argument spans and scores them for each label. At decoding time, we greedily select higher scoring labeled spans. One advantage of our model is to allow us to design and use span-level features, that are difficult to use in token-based BIO tagging approaches. Experimental results demonstrate that our ensemble model achieves the state-of-the-art results, 87.4 F1 and 87.0 F1 on the CoNLL-2005 and 2012 datasets, respectively.

1 Introduction

Semantic Role Labeling (SRL) is a shallow semantic parsing task whose goal is to recognize the predicate-argument structure of each predicate. Given a sentence and a target predicate, SRL systems have to predict semantic arguments of the predicate. Each argument is a *span*, a unit that consists of one or more words. A key to the argument span prediction is how to represent and model spans.

One popular approach to it is based on BIO tagging schemes. State-of-the-art neural SRL models adopt this approach (Zhou and Xu, 2015; He et al., 2017; Tan et al., 2018). Using features induced by neural networks, they predict a BIO tag for each word. Words at the beginning and inside of argument spans have the “B” and “I” tags, and words outside argument spans have the tag “O.” While yielding high accuracies, this approach reconstructs argument spans from the predicted BIO tags instead of directly predicting the spans.

Another approach is based on labeled span prediction (Täckström et al., 2015; FitzGerald et al., 2015). This approach scores each span with its label. One advantage of this approach is to allow us to design and use span-level features, that are difficult to use in BIO tagging approaches. However, the performance has lagged behind that of the state-of-the-art BIO-based neural models.

To fill this gap, this paper presents a simple and accurate span-based model. Inspired by recent span-based models in syntactic parsing and coreference resolution (Stern et al., 2017; Lee et al., 2017), our model directly scores all possible labeled spans based on span representations induced from neural networks. At decoding time, we greedily select higher scoring labeled spans. The model parameters are learned by optimizing log-likelihood of correct labeled spans.

We evaluate the performance of our span-based model on the CoNLL-2005 and 2012 datasets (Carreras and Màrquez, 2005; Pradhan et al., 2012). Experimental results show that the span-based model outperforms the BiLSTM-CRF model. In addition, by using contextualized word representations, ELMo (Peters et al., 2018), our ensemble model achieves the state-of-the-art results, 87.4 F1 and 87.0 F1 on the CoNLL-2005 and 2012 datasets, respectively. Empirical analysis on these results shows that the label prediction ability of our span-based model is better than that of the CRF-based model. Another finding is that ELMo improves the model performance for span boundary identification.

In summary, our main contributions include:

- A simple span-based model that achieves the state-of-the-art results.
- Quantitative and qualitative analysis on strengths and weaknesses of the span-based model.
- Empirical analysis on the performance gains by ELMo.

Our code and scripts are publicly available.[1]

---

[1] https://github.com/hiroki13/span-based-srl.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1630–1642
Brussels, Belgium, October 31 - November 4, 2018.

2 Model

We treat SRL as *span selection*, in which we select appropriate spans from a set of possible spans for each label. This section formalizes the problem and provides our span selection model.

2.1 Span Selection Problem

Problem Setting

Given a sentence that consists of T words $w_{1:T} = w_1, \cdots, w_T$ and the target predicate position index p, the goal is to predict a set of labeled spans $Y = \{\langle i, j, r\rangle_k\}_{k=1}^{|Y|}$.

$$\textbf{Input}: X = \{w_{1:T}, p\},$$
$$\textbf{Output}: Y = \{\langle i, j, r\rangle_k\}_{k=1}^{|Y|} \ .$$

Each labeled span $\langle i, j, r\rangle$ consists of word indices i and j in the sentence ($1 \leq i \leq j \leq T$) and a semantic role label $r \in \mathcal{R}$.

One simple method to predict Y is to select the highest scoring span (i, j) from all possible spans $\mathcal{S}$ for each label r,

$$\underset{(i,j)\in\mathcal{S}}{\text{argmax}} \ \textsc{Score}_r(i, j), \ r \in \mathcal{R} \ . \tag{1}$$

Function $\textsc{Score}_r(i, j)$ returns a real value for each span $(i, j) \in \mathcal{S}$ (described in Section 2.2 in more detail). The number of possible spans $\mathcal{S}$ in the input sentence $w_{1:T}$ is $\frac{T(T+1)}{2}$, and $\mathcal{S}$ is defined as follows,

$$\mathcal{S} = \{(i, j) \mid i, j \in \{1, \cdots, T\}, i \leq j\} \ .$$

Note that some semantic roles may not appear in the sentence. To deal with the absence of some labels, we define the predicate position span (p, p) as a Null span and train a model to select the Null span when there is no span for the label.[2]

Example

Consider the following sentence with the set of correct labeled spans Y.

$$\begin{array}{cccc} \text{She}_1 & \underline{\text{kept}_2} & \text{a}_3 & \text{cat}_4 \\ [\ \texttt{A0}\ ] & & [& \texttt{A1} \quad] \end{array}$$

$$Y = \{\ \langle 1, 1, \texttt{A0}\rangle, \ \langle 3, 4, \texttt{A1}\rangle,$$
$$\langle 2, 2, \texttt{A2}\rangle, \cdots, \langle 2, 2, \texttt{TMP}\rangle\ \}$$

The input sentence is $w_{1:4}$ = "She kept a cat", and the target predicate position is $p = 2$. The correct labeled span $\langle 1, 1, \texttt{A0}\rangle$ indicates that the A0 argument is "She", and $\langle 3, 4, \texttt{A1}\rangle$ indicates that the A1 argument is "a cat". The other labeled spans $\langle 2, 2, *\rangle$ indicate there are no arguments.

All the possible spans in this sentence are as follows,

$$\mathcal{S}_{w_{1:4}} = \{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2),$$
$$(2, 3), (2, 4), (3, 3), (3, 4), (4, 4)\} \ ,$$

where the predicate span $(2, 2)$ is treated as the Null span. Among these candidates, we select the highest scoring span for each label. As a result, we can obtain correct labeled spans Y.

2.2 Scoring Function

As the scoring function for each span in Eq. 1, we model normalized distribution over all possible spans $\mathcal{S}$ for each label r,

$$\begin{aligned} \textsc{Score}_r(i, j) &= \mathrm{P}_\theta(i, j \mid r) \\ &= \frac{\exp(\mathrm{F}_\theta(i, j, r))}{\sum\limits_{(i', j')\in\mathcal{S}} \exp(\mathrm{F}_\theta(i', j', r))} \ , \end{aligned} \tag{2}$$

where function F_θ returns a real value.

We train the parameters θ of F_θ on a training set,

$$\begin{aligned} \mathcal{D} &= \{(X^{(n)}, Y^{(n)})\}_{n=1}^{|\mathcal{D}|} \ , \\ X &= \{w_{1:T}, p\} \ , \\ Y &= \{\langle i, j, r\rangle_k\}_{k=1}^{|Y|} \ . \end{aligned}$$

To train the parameters θ of F_θ, we minimize the cross-entropy loss function,

$$\mathcal{L}(\theta) = \sum_{(X,Y)\in\mathcal{D}} \ell_\theta(X, Y) \ , \tag{3}$$
$$\ell_\theta(X, Y) = \sum_{\langle i,j,r\rangle\in Y} \log \mathrm{P}_\theta(i, j|r) \ ,$$

where function $\ell_\theta(X, Y)$ is a loss for each sample.

2.3 Function F_θ

Function F_θ in Eq. 2 consists of three types of functions; the base feature function f_{base}, the span feature function f_{span} and the labeling function f_{label} as follows,

$$\mathbf{h}_{1:T} = f_{base}(w_{1:T}, p) \ , \tag{4}$$
$$\mathbf{h}_s = f_{span}(\mathbf{h}_{1:T}, s) \ , \tag{5}$$
$$\mathrm{F}_\theta(i, j, r) = f_{label}(\mathbf{h}_s, r) \ . \tag{6}$$

---

[2] Since the predicate itself can never be an argument of its own, we define the position as the Null span.

Firstly, f_{base} calculates a base feature vector $\mathbf{h}_t$ for each word $w_t \in w_{1:T}$. Then, from a sequence of the base feature vectors $\mathbf{h}_{1:T}$, f_{span} calculates a span feature vector $\mathbf{h}_s$ for a span $s = (i, j)$. Finally, using $\mathbf{h}_s$, f_{label} calculates the score for the span $s = (i, j)$ with a label r.

Each function in Eqs. 4, 5 and 6 can arbitrarily be defined. In Section 3, we describe our functions used in this paper.

2.4 Inference

The simple argmax inference (Eq. 1) selects one span for each label. While this argmax inference is computationally efficient, it faces the following two problematic issues.

(a) The argmax inference sometimes selects spans that overlap with each other.

(b) The argmax inference cannot select multiple spans for one label.

In terms of (a), for example, when $\langle 1, 3, \texttt{A0} \rangle$ and $\langle 2, 4, \texttt{A1} \rangle$ are selected, a part of these two spans overlaps. In terms of (b), consider the following sentence.

He <u>came</u> to the U.S. yesterday at 5 p.m.
`[A0]      [  A4  ][ TMP  ][ TMP ]`

In this example, the label `TMP` is assigned to the two spans ("yesterday" and "at 5 p.m."). Semantic role labels are mainly categorized into (i) *core labels* or (ii) *adjunct labels*. In the above example, the labels `A0` and `A4` are regarded as core labels, which indicate obligatory arguments for the predicate. In contrast, the labels like `TMP` are regarded as adjunct labels, which indicate optional arguments for the predicate. As the example shows, adjunct labels can be assigned to multiple spans.

To deal with these issues, we use a greedy search that keeps the consistency among spans and can return multiple spans for adjunct labels. Specifically, we greedily select higher scoring labeled spans subject to two constraints.

Overlap Constraint: Any spans that overlap with the selected spans cannot be selected.

Number Constraint: While multiple spans can be selected for each adjunct label, at most one span can be selected for each core label.

As a precise description of this algorithm, we describe the pseudo code and its explanation in Appendix A.

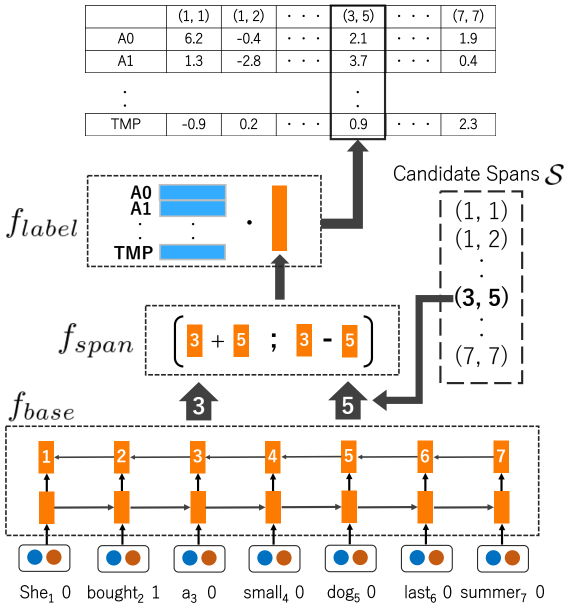

Figure 1: Overall architecture of our BiLSTM-span model.

3 Network Architecture

To compute the score for each span, we have introduced three functions ($f_{base}, f_{span}, f_{label}$) in Section 2.3. As an instantiation of each function, we use neural networks. This section describes our neural networks for each function and the overall network architecture.

3.1 BiLSTM-Span Model

Figure 1 illustrates the overall architecture of our model. The first component f_{base} uses bidirectional LSTMs (BiLSTMs) (Schuster and Paliwal, 1997; Graves et al., 2005, 2013) to calculate the base features. From the base features, the second component f_{span} extracts span features. Based on them, the final component f_{label} calculates the score for each labeled span. In the following, we describe these three components in detail.

Base Feature Function

As the base feature function f_{base}, we use BiLSTMs,

$$f_{base}(w_{1:T}, p) = \textsc{BiLSTM}(w_{1:T}, p) \ .$$

There are some variants of BiLSTMs. Following the deep SRL models proposed by Zhou and Xu (2015) and He et al. (2017), we stack BiLSTMs in an interleaving fashion. The stacked BiLSTMs process an input sequence in a left-to-right manner at odd-numbered layers and in a right-to-left manner at even-numbered layers.

The first layer of the stacked BiLSTMs receives word embeddings $\mathbf{x}^{word} \in \mathbb{R}^{d^{word}}$ and predicate mark embeddings $\mathbf{x}^{mark} \in \mathbb{R}^{d^{mark}}$. As the word embeddings, we can use existing word embeddings. The mark embeddings are created from the mark feature which has a binary value. The value is 1 if the word is the target predicate and 0 otherwise. For example, at the bottom part of Figure 1, the word "bought" is the target predicate and assigned 1 as its mark feature.

Receiving these inputs, the stacked BiLSTMs calculates the hidden states until the top layer. We use these hidden states as the input feature vectors $\mathbf{h}_{1:T}$ for the span feature function f_{span} (Eq. 5). Each vector $\mathbf{h}_t \in \mathbf{h}_{1:T}$ has d^{hidden} dimensions. We provide a detailed description of the stacked BiLSTMs in Appendix B.

Span Feature Function

From the base features induced by the BiLSTMs, we create the span feature representations,

$$f_{span}(\mathbf{h}_{1:T}, s) = [\mathbf{h}_i + \mathbf{h}_j; \mathbf{h}_i - \mathbf{h}_j] \ , \quad (7)$$

where the addition and subtraction features of the i-th and j-th hidden states are concatenated and used as the feature for a span $s = (i, j)$. The resulting vector $\mathbf{h}_s$ is a $2d^{hidden}$ dimensional vector.

The middle part of Figure 1 shows an example of this process. For the span $(3, 5)$, the span feature function f_{span} receives the 3rd and 5th features ($\mathbf{h}_3$ and $\mathbf{h}_5$). Then, these two vectors are added, and the 5th vector is subtracted from the 3rd vector. The resulting vectors are concatenated and given to the labeling function f_{label}.

Our design of the span features is inspired by the span (or segment) features used in syntactic parsing (Wang and Chang, 2016; Stern et al., 2017; Teranishi et al., 2017). While these neural span features cannot be used in BIO-based SRL models, they can easily be incorporated into span-based models.

Labeling Function

Taking a span representation $\mathbf{h}_s$ as input, the labeling function f_{label} returns the score for the span $s = (i, j)$ with a label r. Specifically, we use the following labeling function,

$$f_{label}(\mathbf{h}_s, r) = \mathbf{W}[r] \cdot \mathbf{h}_s \ , \quad (8)$$

where $\mathbf{W} \in \mathbb{R}^{|\mathcal{R}| \times 2d^{hidden}}$ has a row vector associated with each label r, and $\mathbf{W}[r]$ denotes the r-th row vector. As the result of the inner product of $\mathbf{W}[r]$ and $\mathbf{h}_s$, we obtain the score for a span (i, j) with a label r.

The upper part of Figure 1 shows an example of this process. The span representation $\mathbf{h}_s$ for the span $s = (3, 5)$ is created from addition and subtraction of $\mathbf{h}_3$ and $\mathbf{h}_5$. Then, we calculate the inner product of $\mathbf{h}_s$ and $\mathbf{W}[r]$. The score for the label `A0` is 2.1, and the score for the label `A1` is 3.7. In the same manner, by calculating the scores for all the spans $\mathcal{S}$ and labels $\mathcal{R}$, we can obtain the score matrix (at the top part of Figure 1).

3.2 Ensembling

We propose an ensemble model that uses span representations from multiple models. Each base model trained with different random initializations has variance in span representations. To take advantage of it, we introduce a variant of a mixture of experts (MoE) (Shazeer et al., 2017), [3]

$$\mathbf{h}_s^{\text{moe}} = \mathbf{W}_s^{\text{moe}} \cdot \sum_{m=1}^{M} \alpha_m \, \mathbf{h}_s^{(m)} \ , \quad (9)$$

$$f_{label}^{\text{moe}}(\mathbf{h}_s^{\text{moe}}, r) = \mathbf{W}^{\text{moe}}[r] \cdot \mathbf{h}_s^{\text{moe}} \ . \quad (10)$$

Firstly, we combine span representations $\mathbf{h}_s^{(m)}$ from each model $m \in \{1, \cdots, M\}$. $\mathbf{W}_s^{\text{moe}}$ is a parameter matrix and $\{\alpha_m\}_{m=1}^{M}$ are trainable, softmax-normalized parameters. Then, using the combined span representation $\mathbf{h}_s^{\text{moe}}$, we calculate the score in the same way as Eq. 8. We use the same greedy search algorithm used for our base model (Section 2.4).

During training, we update only the parameters of the ensemble model, i.e., $\{\mathbf{W}_s^{\text{moe}}, \mathbf{W}^{\text{moe}}, \{\alpha_m\}_{m=1}^{M}\}$. That is, we fix the parameters of each trained model m. As the loss function, we use the cross-entropy (Eq. 3).

4 Experiments

4.1 Datasets

We use the CoNLL-2005 and 2012 datasets[4]. We follow the standard train-development-test split and use the official evaluation script[5] from the CoNLL-2005 shared task on both datasets.

---

[3] One popular ensemble model for SRL is the product of experts (PoE) model (FitzGerald et al., 2015; He et al., 2017; Tan et al., 2018). In our preliminary experiments, we tried the PoE model but it did not improve the performance.

[4] We use the version of OntoNotes downloaded at: http://cemantix.org/data/ontonotes.html.

[5] The script can be downloaded at: http://www.lsi.upc.edu/ srlconll/soft.html

		Development			Test WSJ			Test Brown			Test ALL		
Emb	Model	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SENNA	CRF	81.7	81.3	81.5	83.3	82.5	82.9	72.6	70.0	71.3	81.9	80.8	81.4
	SPAN	83.6	81.4	82.5	84.7	82.3	83.5	76.0	70.4	73.1	83.6	80.7	82.1
	SPAN (Ensemble)	85.6	82.6	84.1	86.6	83.6	85.1	78.2	71.8	74.8	85.5	82.0	83.7
ELMo	CRF	86.6	86.8	86.7	87.4	87.3	87.3	78.5	78.3	78.4	86.2	86.1	86.1
	SPAN	87.4	86.3	86.9	88.2	87.0	87.6	79.9	77.5	78.7	87.1	85.7	86.4
	SPAN (Ensemble)	**88.0**	**86.9**	**87.4**	**89.2**	**87.9**	**88.5**	**81.0**	**78.4**	**79.6**	**88.1**	**86.6**	**87.4**

Table 1: Experimental results on the CoNLL-2005 dataset, in terms of precision (P), recall (R) and F1. The bold numbers denote the highest precision, recall and F1 scores among all the models.

		Development			Test		
Emb	Model	P	R	F1	P	R	F1
SENNA	CRF	82.8	81.9	82.4	82.9	81.9	82.4
	SPAN	84.3	81.5	82.9	84.4	81.7	83.0
	SPAN (Ensemble)	86.0	83.0	84.5	86.1	83.3	84.7
ELMo	CRF	86.1	**85.8**	85.9	86.0	**85.7**	85.9
	SPAN	87.2	85.5	86.3	87.1	85.3	86.2
	SPAN (Ensemble)	**88.6**	85.7	**87.1**	**88.5**	85.5	**87.0**

Table 2: Experimental results on the CoNLL-2012 dataset.

	CoNLL-05			CoNLL12
	WSJ	Brown	ALL	
Single Model				
ELMo-Span	**87.6**	78.7	**86.4**	**86.2**
He+ 18	87.4	**80.4**	-	85.5
Peters+ 18	-	-	-	84.6
Strubell+ 18	83.9	72.6	-	-
Tan+ 18	84.8	74.1	83.4	82.7
He+ 17	83.1	72.1	81.6	81.7
Zhou+ 15	82.8	69.4	81.1	81.3
FitzGerald+ 15	79.4	71.2	-	79.6
Täckström+ 15	79.9	71.3	-	79.4
Toutanova+ 08	79.7	67.8	-	-
Punyakanok+ 08	79.4	67.8	77.9	-
Ensemble Model				
ELMo-Span	**88.5**	**79.6**	**87.4**	**87.0**
Tan+ 18	86.1	74.8	84.6	83.9
He+ 17	84.6	73.6	83.2	83.4
FitzGerald+ 15	80.3	72.2	-	80.1
Toutanova+ 08	80.3	68.8	-	-
Punyakanok+ 08	79.4	67.8	77.9	-

Table 3: Comparison with existing models. The numbers denote F1 scores on each test set.

4.2 Baseline Model

For comparison, as a model based on BIO tagging approaches, we use the BiLSTM-CRF model proposed by Zhou and Xu (2015). The BiLSTMs for the base feature function f_{base} are the same as those used in our BiLSTM-span model.

4.3 Model Setup

As the base function f_{base}, we use 4 BiLSTM layers with 300 dimensional hidden units. To optimize the model parameters, we use Adam (Kingma and Ba, 2014). Other hyperparameters are described in Appendix C in detail.

Word Embeddings

Word embeddings have a great influence on SRL models. To validate the model performance, we use two types of word embeddings.

- Typical word embeddings, SENNA[6] (Collobert et al., 2011)
- Contextualized word embeddings, ELMo[7] (Peters et al., 2018)

SENNA and ELMo can be regarded as different types of embeddings in terms of the context sensitivity. SENNA and other typical word embeddings always assign an identical vector to each word regardless of the input context. In contrast, ELMo assigns different vectors to each word depending on the input context. In this work, we use these word embeddings that have different properties.[8] These embeddings are fixed during training.

Training

As the objective function, we use the cross-entropy $\mathcal{L}_\theta$ in Eq. 3 with L2 weight decay,

$$\mathcal{L}_\theta = \sum_{(X,Y)\in\mathcal{D}} \ell_\theta(X,Y) + \frac{\lambda}{2}||\theta||^2 \ , \qquad (11)$$

where the hyperparameter λ is the coefficient governing the L2 weight decay.

---

[6] http://ronan.collobert.com/senna/

[7] http://allennlp.org/elmo

[8] In our preliminary experiments, we also used the GloVe embeddings (Pennington et al., 2014), but the performance was worse than SENNA.

4.4 Results

We report averaged scores across five different runs of the model training.

Tables 1 and 2 show the experimental results on the CoNLL-2005 and 2012 datasets. Overall, our span-based ensemble model using ELMo achieved the best F1 scores, 87.4 F1 and 87.0 F1 on the CoNLL-2005 and CoNLL-2012 datasets, respectively. In comparison with the CRF-based single model, our span-based single model consistently yielded better F1 scores regardless of the word embeddings, SENNA and ELMO. Although the performance difference was small between these models using ELMO, it seems natural because both models got much better results and approached to the performance upper bound.

Table 3 shows the comparison with existing models in F1 scores. Our single and ensemble models using ELMO achieved the best F1 scores on all the test sets except the Brown test set.

5 Analysis

To better understand our span-based model, we addressed the following questions and obtained the following findings.

Questions

- **(a)** What are strengths and weaknesses of our span-based model compared with the CRF-based model?
- **(b)** What aspect of SRL does ELMo improve?

Findings

- **(a)** While the CRF-based model is better at span boundary identification (Section 5.1), the span-based model is better at label prediction, especially for A2 (Section 5.2).
- **(b)** ELMo improves the model performance for span boundary identification (Section 5.1).

In addition, we have conducted qualitative analysis on span and label representations learned in the span-based model (Section 5.3).

5.1 Performance for Span Boundary Identification

We analyze the results predicted by the single models. We evaluate F1 scores only for the span boundary match, shown by Table 4. We regard a predicted boundary $\langle i, j, * \rangle$ as correct if it matches the gold annotation regardless of its label.

		CoNLL-05		CoNLL-12	
EMB	MODEL	F1	diff	F1	diff
SENNA	SPAN	86.6	-0.4	87.3	-0.6
	CRF	87.0		87.9	
ELMO	SPAN	90.5	-0.7	90.3	-0.6
	CRF	91.2		90.9	

Table 4: F1 scores only for span boundary match.

		CoNLL-05		CoNLL-12	
EMB	MODEL	Acc.	diff	Acc.	diff
SENNA	SPAN	95.3	+1.5	95.1	+1.5
	CRF	93.8		93.6	
ELMO	SPAN	96.1	+0.9	95.7	+1.3
	CRF	95.2		94.4	

Table 5: Accuracies only for semantic role labels.

On both datasets, the CRF-based models achieved better F1 than that of the span-based models. Also, compared with SENNA, ELMO yielded much better F1 by over 3.0. This suggests that a factor of the overall SRL performance gain by ELMO is the improvement of the model ability to identify span boundaries.

5.2 Performance for Label Prediction

We analyze labels of the predicted results. For labeled spans whose boundaries match the gold annotation, we evaluate the label accuracies. As Table 5 shows, the span-based models outperformed the CRF-based models. Also, interestingly, the performance gap between SENNA and ELMO was not so big as that for span boundary identification.

Label-wise Performance

Table 6 shows F1 scores for frequent labels on the CoNLL-2005 and 2012 datasets. For A0 and A1, the performances of the CRF-based and span-based models were almost the same. For A2, the span-based models outperformed the CRF-based model by about 1.0 F1 on the both datasets. [9]

Label Confusion Matrix

Figure 2 shows a confusion matrix for labeling errors of the span-based model using ELMo.[10] Following He et al. (2017), we only count predicted arguments that match the gold span boundaries.

---

[9] The PNC label got low scores on the CoNLL-2012 dataset in Table 6. Almost all the gold PNC (purpose) labels are assigned to only the news article domain texts of the CoNLL-2012 dataset. The other 6 domain texts have no or very few PNC labels. This can lead to the low performance.

[10] We have observed the same tendency of labeling confusions between the models using ELMo and SENNA.

	CoNLL-2005				CoNLL-2012			
	Senna		ELMo		Senna		ELMo	
Label	Crf	Span	Crf	Span	Crf	Span	Crf	Span
A0	89.9	90.2	93.0	93.2	89.9	90.0	92.5	92.5
A1	83.2	83.8	89.1	89.2	84.7	85.1	88.7	89.0
A2	70.9	73.1	80.0	81.2	78.6	79.4	83.2	84.2
A3	64.4	71.2	78.8	78.5	61.9	62.9	69.0	70.7
ADV	59.3	61.9	68.1	67.0	63.2	63.7	67.5	67.0
DIR	43.2	47.3	56.6	54.5	54.1	52.0	61.1	59.7
LOC	58.2	60.5	68.1	68.3	65.8	65.0	72.0	72.0
MNR	61.4	61.3	66.5	67.7	64.4	65.7	70.5	71.1
PNC	57.3	60.2	68.8	67.7	18.5	13.7	20.2	16.1
TMP	81.8	82.7	86.1	86.0	82.2	82.3	86.1	86.2
Overall	81.5	82.5	86.7	86.9	82.4	82.9	85.9	86.3

Table 6: F1 Scores for frequent labels on the development set of the CoNLL-2005 and 2012 datasets.

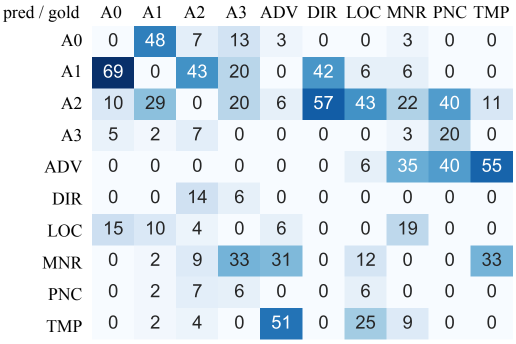

pred / gold	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	0	48	7	13	3	0	0	3	0	0
A1	69	0	43	20	0	42	6	6	0	0
A2	10	29	0	20	6	57	43	22	40	11
A3	5	2	7	0	0	0	0	3	20	0
ADV	0	0	0	0	0	0	6	35	40	55
DIR	0	0	14	6	0	0	0	0	0	0
LOC	15	10	4	0	6	0	0	19	0	0
MNR	0	2	9	33	31	0	12	0	0	33
PNC	0	2	7	6	0	0	6	0	0	0
TMP	0	2	4	0	51	0	25	9	0	0

Figure 2: Confusion matrix for labeling errors of our span-based model using ELMo. Each cell shows the percentage of predicted labels for each gold label.

The span-based model confused A0 and A1 arguments the most. In particular, the model confused them for ergative verbs. Consider the following two sentences:

People start their own business ...
`[ A0 ]`

.. Congress has started to jump on ...
`[  A1  ]`

where the constituents located at the syntactic subjective position fulfill a different role A0 or A1 according to their semantic properties, such as animacy. Such arguments are difficult for SRL models to correctly identify.

Another point is the confusions of A2 with DIR and LOC. As He et al. (2017) pointed out, A2 in a lot of verb frames represents semantic relations such as direction or location, which can cause the confusions of A2 with such location-related adjuncts. To remedy these two problematic issues, it can be a promising approach to incorporate frame knowledge into SRL models by using verb frame dictionaries.

“· · · toy makers to move [across the border] .” GOLD:A2 PRED:DIR		
Nearest neighbors of “across the border”		
1	DIR	across the Hudson
2	DIR	outside their traditional tony circle
3	DIR	across the floor
4	DIR	through this congress
5	A2	off their foundations
6	DIR	off its foundation
7	DIR	off the center field wall
8	A3	out of bed
9	A2	through cottage rooftops
10	DIR	through San Francisco

Table 7: Example of the CoNLL-2005 development set, in which our model misclassified the label for the span “across the border”. We collect 10 nearest neighbors of this span from the training set.

5.3 Qualitative Analysis on Our Model

On Span Representations

Our span-based model computes and uses span representations (Eq. 7) for label prediction. To investigate a relation between the span representations and predicted labels, we qualitatively analyze nearest neighbors of each span representation with its predicted label. Specifically, for each predicted span in the development set, we collect 10 nearest neighbor spans with their gold labels from the training set.

Table 7 shows 10 nearest neighbors of a span “across the border” for the predicate “move”. The label of this span was misclassified, i.e., the predicted label is DIR but the gold is A2. Looking at its nearest neighbor spans, they have different gold labels, such as DIR, A2 and A3. Like this case, we have observed that spans with a misclassified label often have their nearest neighbors with inconsistent labels.

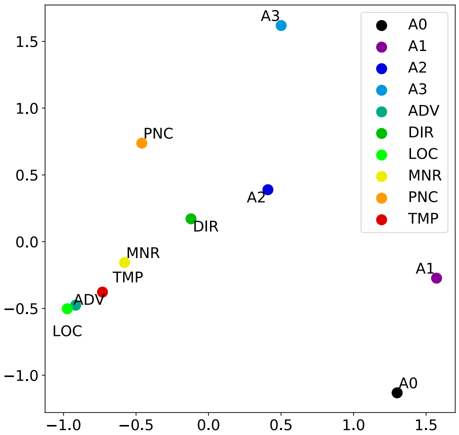

Figure 3: Label embedding distribution of our span-based model.

On Label Embeddings

We analyze the label embeddings in the labeling function (Eq. 8). Figure 3 shows the distribution of the learned label embeddings. The adjunct labels are close to each other, which are likely to be less discriminative. Also, the core label A2 is close to the adjunct label DIR, which are often confused by the model. To enhance the discriminative power, it is promising to apply techniques that keep label representations far away from each other (Wen et al., 2016; Luo et al., 2017).

6 Related Work

6.1 Semantic Role Labeling Tasks

Automatic SRL has been widely studied (Gildea and Jurafsky, 2002). There have been two main styles of SRL.

- FrameNet-style SRL (Baker et al., 1998)
- PropBank-style SRL (Palmer et al., 2005)

In this paper, we have tackled PropBank-style SRL.[11]

In PropBank-style SRL, there have been two main task settings.

- Span-based SRL: CoNLL-2004 and 2005 shared tasks (Carreras and Marquez, 2004; Carreras and Màrquez, 2005)
- Dependency-based SRL: CoNLL-2008 and 2009 shared tasks (Surdeanu et al., 2008; Hajič et al., 2009)

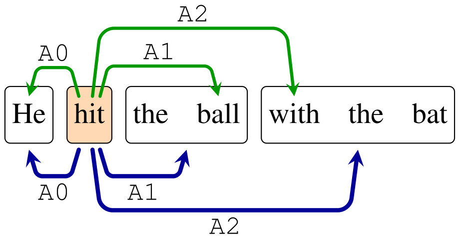

Figure 4: Example of dependency-based SRL (the upper part) and span-based SRL (the lower part).

Figure 4 illustrates an example of span-based and dependency-based SRL. In dependency-based SRL (at the upper part of Figure 4), the correct A2 argument for the predicate “hit” is the word “with”. On one hand, in span-based SRL (at the lower part of Figure 4), the correct A2 argument is the span “with the bat”.

For span-based SRL, the CoNLL-2004 and 2005 shared tasks (Carreras and Marquez, 2004; Carreras and Màrquez, 2005) provided the task settings and datasets. In the task settings, various SRL models, from traditional pipeline models to recent neural ones, have been proposed and competed with each other (Pradhan et al., 2005; He et al., 2017; Tan et al., 2018). For dependency-based SRL, the CoNLL-2008 and 2009 shared tasks (Surdeanu et al., 2008; Hajič et al., 2009) provided the task settings and datasets. As in span-based SRL, recent neural models achieved high-performance in dependency-based SRL (Marcheggiani et al., 2017; Marcheggiani and Titov, 2017; He et al., 2018b; Cai et al., 2018). This paper focuses on span-based SRL.

6.2 BIO-based SRL Models

Span-based SRL can be solved as BIO sequential tagging (Hacioglu et al., 2004; Pradhan et al., 2005; Màrquez et al., 2005).

Neural models State-of-the-art SRL models use neural networks based on the BIO tagging approach. The pioneering neural SRL model was proposed by Collobert et al. (2011). They use convolutional neural networks (CNNs) and CRFs. Instead of CNNs, Zhou and Xu (2015) and He et al. (2017) used stacked BiLSTMs and achieved strong performance without syntactic inputs. Tan et al. (2018) replaced stacked BiLSTMs with self-attention architectures. Strubell et al. (2018) improved the self-attention SRL model by incorporating syntactic information.

---

[11] Detailed descriptions on FrameNet-style and PropBank-style SRL can be found in Baker et al. (1998); Das et al. (2014); Kingsbury and Palmer (2002); Palmer et al. (2005).

Word representations Typical word representations, such as SENNA (Collobert et al., 2011) and GloVe (Pennington et al., 2014), have been used and contributed to the performance improvement (Collobert et al., 2011; Zhou and Xu, 2015; He et al., 2017). Recently, Peters et al. (2018) integrated contextualized word representation, ELMo, into the model of He et al. (2017) and improved the performance by 3.2 F1 score. Strubell and McCallum (2018) also integrated ELMo into the model of Strubell et al. (2018) and reported the performance improvement.

6.3 Span-based SRL Models

Another line of approaches to SRL is labeled span modeling (Xue and Palmer, 2004; Koomen et al., 2005; Toutanova et al., 2005).

Typical models Typically, in this approach, models firstly identify candidate argument spans (argument identification) and then classify each span into one of the semantic role labels (argument classification). For inference, several effective methods have been proposed, such as structural constraint inference by using integer linear programming (Punyakanok et al., 2008) or dynamic programming (Täckström et al., 2015; FitzGerald et al., 2015).

Recent span-based model A very recent work, He et al. (2018a), proposed a span-based SRL model similar to our model. They also used BiLSTMs to induce span representations in an end-to-end fashion. A main difference is that while they model $\mathrm{P}(r|i,j)$, we model $\mathrm{P}(i,j|r)$. In other words, while their model seeks to select an appropriate label for each span (*label selection*), our model seeks to select appropriate spans for each label (*span selection*). This point distinguishes between their model and ours.

FrameNet span-based model For FrameNet-style SRL, Swayamdipta et al. (2017) used a segmental RNN (Kong et al., 2016), combining bidirectional RNNs with semi-Markov CRFs (Sarawagi and Cohen, 2004). Their model computes span representations using BiLSTMs and learns a conditional distribution over all possible labeled spans of an input sequence. Although we cannot compare our results with theirs, we can regard that our model is simpler and effective for PropBank-style SRL.

6.4 Span-based Models in Other NLP Tasks

In syntactic parsing, Wang and Chang (2016) proposed an LSTM-based sentence segment embedding method named LSTM-Minus. Stern et al. (2017); Kitaev and Klein (2018) incorporated the LSTM Minus into their parsing model and achieved the best results in constituency parsing. In coreference resolution, Lee et al. (2017, 2018) presented an end-to-end coreference resolution model, which considers all spans in a document as potential mentions and learn distributions over possible antecedents for each. Our model can be regarded as an extension of their model.

7 Conclusion and Future Work

We have presented a simple and accurate span-based model. We treat SRL as *span selection* and our model seeks to select appropriate spans for each label. Experimental results have demonstrated that despite the simplicity, the model outperforms a strong BiLSTM-CRF model. Also, our span-based ensemble model using ELMo achieves the state-of-the-art results on the CoNLL-2005 and 2012 datasets. Through empirical analysis, we have obtained some interesting findings. One of them is that the span-based model is better at label prediction compared with the CRF-based model. Another one is that ELMo improves the model performance for span boundary identification.

An interesting direction for future work concerns evaluating span representations from our span-based model. Since the investigation on the characteristics of the representations can lead to interesting findings, it is worthwhile evaluating them intrinsically and extrinsically. Another promising direction is to explore methods of incorporating frame knowledge into SRL models. We have observed that a lot of label confusions arise due to the lack of such knowledge. The use of frame knowledge to reduce these confusions is a straightforward approach.

Acknowledgments

This work was partially supported by JST CREST Grant Number JPMJCR1513 and JSPS KAKENHI Grant Number 18K18109. We are grateful to the members of the NAIST Computational Linguistics Laboratory, the members of Tohoku University Inui-Suzuki Laboratory, Kentaro Inui, Jun Suzuki, Yuichiro Matsubayashi, and the anonymous reviewers for their insightful comments.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of ACL-COLING*, pages 86–90.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of COLING*, pages 2753–2765.

Xavier Carreras and Lluís Marquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL*, pages 89–97.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*, pages 152–164.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. In *Journal of Machine Learning Research*.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of EMNLP*, pages 960–970.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pages 249–256.

Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Proceedings of Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional LSTM networks for improved phoneme classification and recognition. In *Proceedings of International Conference on Artificial Neural Networks*, pages 799–804.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proceedings of CoNLL*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*, pages 1–18.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of ACL*, pages 364–369.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what ' s next. In *Proceedings of ACL*, pages 473–483.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of ACL*, pages 2061–2071.

D.P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *Proceedings of LREC*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of ACL*, pages 2676–2686.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *Proceedings of ICLR*.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of CoNLL*, pages 181–184.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of EMNLP*, pages 188–197.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of NAACL*, pages 687–692.

Lingkun Luo, Xiaofang Wang, Shiqiang Hu, Chao Wang, Yuxing Tang, and Liming Chen. 2017. Close yet distinctive domain adaptation. *arXiv preprint arXiv:1704.04235*.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of CoNLL*, pages 411–420.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*, pages 1506–1515.

Lluís Màrquez, Pere Comas, Jesús Giménez, and Neus Catala. 2005. Semantic role labeling as sequential tagging. In *Proceedings of CoNLL*, pages 193–196.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of CoNLL*, pages 217–220.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of EMNLP-CoNLL*, pages 1–40.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Proceedings of NIPS*, pages 1185–1192.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, pages 2673–2681.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of ACL*, pages 818–827.

Emma Strubell and Andrew McCallum. 2018. Syntax helps elmo understand semantics: Is syntax still relevant in a deep neural architecture for srl? In *Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*, pages 19–27.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. *arXiv preprint arXiv:1804.08199*.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*, pages 159–177.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. *arXiv preprint arXiv:1706.09528*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of ACL*, 3:29–41.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of AAAI*.

Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Coordination boundary identification with similarity and replaceability. In *Proceedings of IJCNLP*, pages 264–272.

Kristina Toutanova, Aria Haghighi, and Christopher D Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL*, pages 589–596.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of ACL*, pages 2306–2315.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. 2016. A discriminative feature learning approach for deep face recognition. In *Proceedings of ECCV*, pages 499–515.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL-IJCNLP*, pages 1127–1137.

A Span-Consistent Greedy Search

Algorithm 1 Span-Consistent Greedy Search

```
 1: Input: Score Matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{R}|\times|\mathcal{S}|}$,
 2:        Predicate Position Index $p$
 3:        Core Label Set $\mathcal{R}^{(\text{core})}$
 4: spans $\leftarrow \phi$
 5: used_cores $\leftarrow \phi$
 6: $\mathcal{U} \leftarrow \{(i, j, r, \text{score}) \in flatten(\mathbf{M})\}$
 7: $\mathcal{U} \leftarrow filter(\mathcal{U}, p)$
 8: for $(i, j, r, \text{score}) \in sort(\mathcal{U})$ do
 9:   if $r \notin$ used_cores and
10:     is_overlap($(i, j)$, spans) is False then
11:       spans $\leftarrow$ spans $\cup \{\langle i, j, r\rangle\}$
12:       if $r \in \mathcal{R}^{(\text{core})}$ then
13:         used_cores $\leftarrow$ used_cores $\cup \{r\}$
14: return spans
```

Algorithm 1 describes the pseudo code of the greedy search algorithm introduced in Section 2.4. This algorithm receives the three inputs (line 1-3). $\mathbf{M}$ is the score matrix illustrated at the top part of Figure 1 in Section 3. Each cell of the matrix represents the score of each span. p is a target predicate position index. $\mathcal{R}^{(\text{core})}$ is the set of core labels. At line 4, the variable "spans" is initialized. This variable stores the selected spans to be returned as the output. At line 5, the variable "used_cores" is initialized. This variable keeps track of the already selected core labels.

At line 6, the score matrix $\mathbf{M}$ is converted to tuples, (i, j, r, score), by the function $flatten(\cdot)$. These tuples are stored in the variable $\mathcal{U}$. At line 7, from $\mathcal{U}$, we remove the tuples that fall into any one of the followings, (i) the tuples whose boundary (i, j) overlaps with the predicate position p or (ii) the tuples whose score is lower than that of the predicate span tuples. In terms of (i), since spans whose boundary (i, j) overlaps with the predicate position, $i \leq p \leq j$, can never be a correct argument, we remove such tuples. In terms of (ii), we remove the tuples $(*, *, r, \text{score})$ whose score is lower than that of the predicate span tuple (p, p, r, score). In Section 2, we define the predicate span (p, p) as the NULL span, implying that we can regard the spans whose score is lower than that of the NULL span as an inappropriate argument. Thus, we remove such tuples from the set of the candidates $\mathcal{U}$.

The main processing starts from line 8. Based on the scores, the function $sort(\cdot)$ sorts the tuples (i, j, r, score) in a descending order. At line 9-10, there are constraints for output spans. At line 9, "$r \notin$ used_cores" represents the constraint that at most one span can be selected for each core label. At line 10, the function is_overlap$(\cdot)$ takes as input a span (i, j) and the set of the selected spans, and returns the boolean value ("True" or "False") that represents whether the span overlaps with any one of the selected spans or not.

At line 11, the span is added to the set of the selected spans. At line 12-13, if the label r is included in the core labels $\mathcal{R}^{(\text{core})}$, the label is added to "used_cores". At line 14, as the final output, the set of the selected spans "spans" is returned.

B BiLSTMs

As the base feature function f_{base} (Eq. 4 in Section 2.3), we use BiLSTMs,

$$f_{base}(w_{1:T}, p) = \text{BiLSTM}(w_{1:T}, p) \ .$$

In particular, we use the stacked BiLSTMs in an interleaving fashion (Zhou and Xu, 2015; He et al., 2017). The stacked BiLSTMs process an input sequence in a left-to-right manner for odd-numbered layers and in a right-to-left manner for even-numbered layers.

The stacked BiLSTMs consist of L layers. The hidden state in each layer $\ell \in \{1, \cdots, L\}$ is calculated as follows,

$$\mathbf{h}_t^{(\ell)} = \begin{cases} \text{LSTM}^{(\ell)}(\mathbf{x}_t^{(\ell)},\ \mathbf{h}_{t-1}^{(\ell)}) & (\ell = \text{odd}) \\ \text{LSTM}^{(\ell)}(\mathbf{x}_t^{(\ell)},\ \mathbf{h}_{t+1}^{(\ell)}) & (\ell = \text{even}) \ . \end{cases}$$

Both of the odd- and even-numbered layers receive $\mathbf{x}_t^{(\ell)}$ as the first input of the LSTM. For the second input, odd-numbered layers receive $\mathbf{h}_{t-1}^{(\ell)}$, whereas even-numbered layers receive $\mathbf{h}_{t+1}^{(\ell)}$.

Between the LSTM layers, we use the following connection (Zhou and Xu, 2015),

$$\mathbf{x}_t^{(\ell+1)} = \text{ReLU}(\mathbf{W}^{(\ell)} \cdot [\mathbf{x}_t^{(\ell)}; \mathbf{h}_t^{(\ell)}]) \ .$$

Here, we firstly concatenate $\mathbf{x}_t^{(\ell)}$ and $\mathbf{h}_t^{(\ell)}$, and then calculate the inner product of the concatenated vector and the parameter matrix $\mathbf{W}^{(\ell)}$ with the rectified linear units (ReLU). As a result, we obtain the input representation $\mathbf{x}_t^{(\ell+1)}$ for the next ($\ell + 1$-th) LSTM layer.

In the first layer, LSTM$^{(1)}$ receives an input feature vector $\mathbf{x}_t^{(1)}$. Following He et al. (2017), we create this vector by concatenating a word embedding and predicate mark embedding,

$$\mathbf{x}_t^{(1)} = [\mathbf{x}_t^{word}; \mathbf{x}_t^{mark}] \ ,$$

where $\mathbf{x}^{word} \in \mathbb{R}^{d^{word}}$ and $\mathbf{x}^{mark} \in \mathbb{R}^{d^{mark}}$. The mark embedding is created from the binary mark feature. The value is 1 if the word is the target predicate and 0 otherwise.

After the L-th LSTM layer runs, we obtain $\mathbf{x}_{1:T}^{(L+1)} = \mathbf{x}_1^{(L+1)}, \cdots, \mathbf{x}_T^{(L+1)}$. We use them as the input of the span feature function f_{span} (Eq. 5 in Section 2.3), i.e., $\mathbf{h}_{1:T} = \mathbf{x}_{1:T}^{(L+1)}$. Each vector $\mathbf{h}_t \in \mathbf{h}_{1:T}$ has d^{hidden} dimensions.

C Hyperparameters

Name	Value
Word Embedding d^{word}	50-dimensional SENNA 1024-dimensional ELMo
Mark Embedding d^{mark}	50-dimensional vector
LSTM Layers L	4
LSTM Hidden Units d^{hidden}	300 dimensions
Mini-batch Size	32
Optimization	Adam
Learning Rate	0.001
L2 Regularization λ	0.0001
Dropout Ratio for BiLSTMs	0.1
Dropout Ratio for ELMo	0.5

Table 8: Hyperparameters for our span-based model.

C.1 Span-based Model

Table 8 lists the hyperparameters used for our span-based model.

Word representation setup As word embeddings $\mathbf{x}^{word}$, we use two types of embeddings, (i) SENNA (Collobert et al., 2011), 50-dimensional word vectors ($d^{word} = 50$), and (ii) ELMo (Peters et al., 2018), 1024-dimensional vectors ($d^{word} = 1024$). During training, we fix these word embeddings (not update them). As predicate mark embeddings $\mathbf{x}^{mark}$, we use randomly initialized 50-dimensional vectors ($d^{mark} = 50$). During training, we update them.

Network setup As the base feature function f_{base}, we use 4 stacked BiLSTMs (2 forward and 2 backward LSTMs) with 300-dimensional hidden units ($d^{hidden} = 300$). Following He et al. (2017), we initialize all the parameter matrices in BiLSTMs with random orthonormal matrices (Saxe et al., 2013). Other parameters are initialized following Glorot and Bengio (2010), and bias parameters are initialized with zero vectors.

Regularization We set the coefficient λ for the L2 weight decay (Eq. 11 in Section 4.3) to 0.0001. We apply dropout (Srivastava et al., 2014) to the input vectors of each LSTM with dropout ratio of 0.1 and the ELMo embeddings with dropout ratio of 0.5.

Training To optimize the parameters, we use Adam (Kingma and Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is initialized to 0.001. After training 50 epochs, we halve the learning rate every 25 epochs. Parameter updates are performed in mini-batches of 32. The number of training epochs is set to 100. We save the parameters that achieve the best F1 score on the development set and evaluate them on the test set. Training our model on the CoNLL-2005 training set takes about one day and on the CoNLL-2012 training set takes about two days on a single GPU, respectively.

C.2 Ensemble Model

Our ensemble model uses span representations $\mathbf{h}_s^{(m)}$ from base models $m \in \{1, \cdots, M\}$ (Section 3.2). We use 5 base models ($M = 5$) learned over different runs. Note that, during training, we fix the parameters of the five base models and update only the parameters of the ensemble model.

Network setup The parameter matrix $\mathbf{W}_s^{\text{moe}}$ (Eq. 9 in Section 3.2) is initialized with the identity matrix. The scalar parameters $\{\alpha_m\}_{m=1}^{M}$ (Eq. 9) are initialized with 0. Each row vector $\mathbf{W}^{\text{moe}}[r]$ of the parameter matrix $\mathbf{W}^{\text{moe}}$ (Eq. 10) is initialized with the averaged vector over the row vectors $\mathbf{W}^{(m)}[r]$ of each model m, i.e., $\frac{1}{M}\sum_{m=1}^{M} \mathbf{W}^{(m)}[r]$.

Training To optimize the parameters, we use Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to 0.0001. Parameter updates are performed in mini-batches of 8. The number of training epochs is set to 20. We save the parameters that achieve the best F1 score on the development set and evaluate them on the test set. Training one ensemble model on the CoNLL-2005 and 2012 training sets takes about one day on a single GPU.

Mapping Language to Code in Programmatic Context

Srinivasan Iyer*, **Ioannis Konstas†**, **Alvin Cheung*** **and Luke Zettlemoyer***
*Paul G. Allen School of Computer Science and Engineering, Univ. of Washington, Seattle, WA
{sviyer, akcheung, lsz}@cs.washington.edu
†Heriot-Watt University, Edinburgh, UK
i.konstas@hw.ac.uk

Abstract

Source code is rarely written in isolation. It depends significantly on the programmatic context, such as the class that the code would reside in. To study this phenomenon, we introduce the task of generating class member functions given English documentation and the programmatic context provided by the rest of the class. This task is challenging because the desired code can vary greatly depending on the functionality the class provides (e.g., a sort function may or may not be available when we are asked to "return the smallest element" in a particular member variable list). We introduce CONCODE, a new large dataset with over 100,000 examples consisting of Java classes from online code repositories, and develop a new encoder-decoder architecture that models the interaction between the method documentation and the class environment. We also present a detailed error analysis suggesting that there is significant room for future work on this task.

1 Introduction

Natural language can be used to define complex computations that reuse the functionality of rich, existing code bases. However, existing approaches for automatically mapping natural language (NL) to executable code have considered limited language or code environments. They either assume fixed code templates (i.e., generate only parts of a method with a predefined structure; Quirk et al., 2015), a fixed context (i.e., generate the body of the same method within a single fixed class; Ling et al., 2016), or no context at all (i.e., generate code tokens from the text alone; Oda et al., 2015). In this paper, we introduce new data and methods for learning to map language to source code within the context of a real-world programming environment, with application to generating member functions from documentation for automatically collected Java class environments.

```
public class SimpleVector implements Serializable {
  double[] vecElements;
  double[] weights;

  NL Query: Adds a scalar to this vector in place.
  Code to be generated automatically:
  public void add(final double arg0) {
    for (int i = 0; i < vecElements.length; i++){
      vecElements[i] += arg0;
    }
  }

  NL Query: Increment this vector
  Code to be generated automatically:
  public void inc() {
    this.add(1);
  }
}
```

Figure 1: Code generation based on the class environment and method documentation. The figure shows a class where the programmer wants to automatically generate the `add` method from documentation, assuming the rest of the class is already written. The system needs to understand that `vecElements` is the vector to be augmented, and that the method must take in a scalar parameter as the element to be added. The model also needs to disambiguate between the member variables `vecElements` and `weights`.

The presence of rich context provided by an existing code environment better approximates the way programmers capitalize on code re-use, and also introduces new language understanding challenges. Models must (a) map the NL to environment variables, library API calls and user-defined methods found elsewhere in the class based on their names, types and signatures, and (b) decide on the structure of the resulting code. For example, in Figure 1, to generate the method `inc()` from the corresponding NL, *Increment this vector*, it is crucial to know of the existence of class method `add()`. This helps us decide if it should directly call `add()` or generate the method from scratch by iterating through the `vecElements` array and incrementing each element. Similarly, for generating the `add()` method, the code needs to use the class variable

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1643–1652
Brussels, Belgium, October 31 - November 4, 2018.

`vecElements` correctly. Overall, the code environment provides rich information relating to the intent of the developer, and can be used to considerably reduce ambiguity in the NL documentation.

To learn such a code generator, we use a specialized neural encoder-decoder model that (a) encodes the NL together with representations based on subword units for environment identifiers (member variables, methods) and data types, and (b) decodes the resulting code using an attention mechanism with multiple steps, by first attending to the NL, and then to the variables and methods, thus also learning to copy variables and methods. This two-step attention helps the model to match words in the NL with representations of the identifiers in the environment. Rather than directly generating the output source code tokens (Dong and Lapata, 2016; Iyer et al., 2017), the decoder generates production rules from the grammar of the target programming language similar to Rabinovich et al. (2017), Yin and Neubig (2017), and Krishnamurthy et al. (2017) and therefore, guarantees the syntactic well-formedness of the output.

To train our model, we collect and release CONCODE, a new dataset comprising over 100,000 (class environment, NL, code) tuples by gathering Java files containing method documentation from public Github repositories. This is an order of magnitude larger than existing datasets that map NL to source code for a general purpose language (MTG from Ling et al. (2016) has 13k examples), contains a larger variety of output code templates than existing datasets built for a specific domain, and is the first to condition on the environment of the output code. Also, by design, it contains examples from several domains, thus introducing open-domain challenges of new identifiers during test time (some e.g. class environments are LookupCommand, ColumnFileReader and ImageSequenceWriter). Our model achieves an exact match accuracy of 8.6% and a BLEU score (a metric for partial credit; Papineni et al., 2002) of 22.11, outperforming retrieval and recent neural methods. We also provide an extensive ablative analysis, quantifying the contributions that come from the context and the model, and suggesting that our work opens up various areas for future investigation.

NL query:
Adds a scalar to this vector in place

Variables: [Type, Name]
double[] vecElements
double[] weights

Environment

Methods: [Return Type, Name, ParameterList]
void inc ()
float dotProduct (SimpleVector other)
float multiply(float scalar)

Source code:

```
public void add(final double arg0) {
   for (int i = 0; i < vecElements.length(); i++){
      vecElements[i] += arg0;
   }
}
```

AST Derivation:

```
MemberDeclaration-->MethodDeclaration
MethodDeclaration-->
  TypeTypeOrVoid IdentifierNT FormalParameters
   MethodBody
TypeTypeOrVoid-->void
IdentifierNT-->add
FormalParameters-->( FormalParameterList )
FormalParameterList-->FormalParameter
 ...
Primary-->IdentifierNT
IdentifierNT-->i
Nt_68-->+=
Expression-->Primary
Primary-->IdentifierNT
IdentifierNT-->arg0
```

Figure 2: Our task involves generating the derivation of the source code of a method based on the NL documentation, class member variables (names and data types), and other class member methods (method names and return types), which form the code environment.

2 Task Definition

We introduce the task of generating source code from NL documentation, conditioned on the class environment the code resides in. The environment comprises two lists of entities: (1) class member variable names with their data types (for example, `double[] vecElements` as seen in Figure 2), and (2) member function names together with their return types (for example, `void inc()`).[1] Formally, let $q^{(i)}$, $a^{(i)}$ denote the NL and source code respectively for the i^{th} training example, where $a^{(i)}$ is a sequence of production rules that forms the derivation of its underlying source code. The environment comprises a list of variables names ${v^{(i)}}_{1..|v^{(i)}|}$ and their corresponding types ${t^{(i)}}_{1..|t^{(i)}|}$, as well as method names ${m^{(i)}}_{1..|m^{(i)}|}$ and their return types

---

[1] The method parameters and body can be used as well but we leave this to future work.

$r^{(i)}{}_{1..|r^{(i)}|}$. Our goal is to generate the derivation of $a^{(i)}$ given $q^{(i)}$ and the environment (see Figure 2).

3 Models

We evaluate a number of encoder-decoder models that generate source code derivations from NL and the class environment. Our best model encodes all environment components broken down into subword units (Sennrich et al., 2016) separately, using Bi-LSTMs and decodes these contextual representations to produce a sequence of valid production rules that derive syntactically valid source code. The decoder also uses a two-step attention mechanism to match words in the NL with environment components, and then uses a supervised copy mechanism (Gu et al., 2016a) to incorporate environment elements in the resulting code. We describe this architecture below.

3.1 Encoder

The encoder computes contextual representations of the NL and each component in the environment. Each word of the NL, q_i, is embedded into a high dimensional space using Identifier matrix I (denoted as $\mathbf{q_i}$) followed by the application of a n-layer bidirectional LSTM (Hochreiter and Schmidhuber, 1997). The hidden states of the last layer $(h_1, \cdots, h_z)$ are passed on to the attention layer, while the hidden states at the last token are used to initialize the decoder.

$$h_1, \cdots, h_z = \text{BiLSTM}(\mathbf{q_1}, \ldots, \mathbf{q_z})$$

To encode variables and methods, the variable types (t_i) and method return types (r_i) are embedded using a type matrix T (denoted as $\mathbf{t_i}$ and $\mathbf{r_i}$). To encode the variable and method names (v_i, m_i), they are first split based on camel-casing, and each component is embedded using I, represented as $\mathbf{v_{i1}}, \ldots, \mathbf{v_{ij}}$ and $\mathbf{m_{i1}}, \ldots, \mathbf{m_{ik}}$. The encoded representation of the variable and method names is the final hidden state of the last layer of a Bi-LSTM over these embeddings ($\mathbf{v_i}$ and $\mathbf{m_i}$). Finally, a 2-step Bi-LSTM is executed on the concatenation of the variable type embedding and the variable name encoding. The corresponding hidden states form the final representations of the variable type and the variable name $(\hat{t_i}, \hat{v_i})$ and are passed on to the attention mechanism. Method return types and names are processed identically using the same Bi-LSTMs and embedding matrices $(\hat{r_i}, \hat{m_i})$.

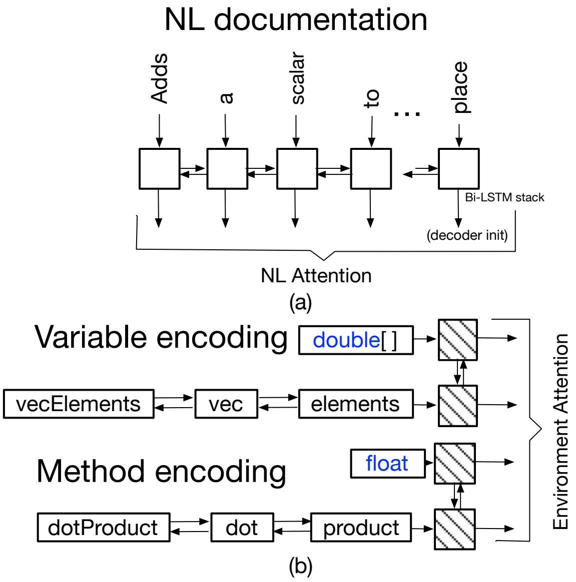

Figure 3: The encoder creates contextual representations of the NL (a), the variables and the methods (b). Variable (method) names are split based on camel-casing and encoded using a BiLSTM. The variable (method) type and name are further contextualized using another BiLSTM.

$$\begin{aligned}
\mathbf{t_i} &= t_i T;\ \mathbf{v_{ij}} = v_{ij} I \\
\mathbf{r_i} &= r_i T;\ \mathbf{m_{ik}} = m_{ik} I \\
\mathbf{v_i} &= \text{BiLSTM}(\mathbf{v_{i1}}, \ldots, \mathbf{v_{ij}}) \\
\mathbf{m_i} &= \text{BiLSTM}(\mathbf{m_{i1}}, \ldots, \mathbf{m_{ik}}) \\
\hat{t_i}, \hat{v_i} &= \text{BiLSTM}(\mathbf{t_i}, \mathbf{v_i}) \\
\hat{r_i}, \hat{m_i} &= \text{BiLSTM}(\mathbf{r_i}, \mathbf{m_i})
\end{aligned}$$

Figure 3 shows an example of the encoder.

3.2 Decoder

We represent the source code to be produced as a sequence of production rules (a_t at step t), with a non-terminal n_t on the left hand side and a combination of terminal and non-terminal symbols on the right hand side (see Figure 2). The first non-terminal is *MemberDeclaration*. Subsequently, every non-terminal is expanded in a depth first left to right fashion, similar to Yin and Neubig (2017). The probability of a source code snippet is decomposed as a product of the conditional probability of generating each step in the sequence of rules conditioned on the previously generated rules. Our decoder is an LSTM-based RNN that produces a context vector c_t at each time step, which is used to compute a distribution over next actions.

$$p(a_t|a_{<t}) \propto exp(W^{n_t} c_t) \qquad (1)$$

Here, W^{n_t} is a $|n_t| \times H$ matrix, where $|n_t|$ is the total number of unique production rules that n_t can be expanded to. The context vector c_t is computed using the hidden state s_t of an n-layer decoder LSTM cell and attention vectors over the NL and the context (z_t and e_t), as described below.

Decoder LSTM The decoder uses an n-layer LSTM whose hidden state s_t is computed based on the current non-terminal n_t to be expanded, the previous production rule a_{t-1}, the parent production rule $\text{par}(n_t)$ that produced n_t, the previous decoder LSTM state s_{t-1}, and the decoder state of the LSTM cell that produced n_t, denoted as s_{n_t}.

$$s_t = \text{LSTM}(n_t, a_{t-1}, \text{par}(n_t), s_{t-1}, s_{n_t}) \qquad (2)$$

We use an embedding matrix N to embed n_t and matrix A to embed a_{t-1} and $\text{par}(n_t)$. If a_{t-1} is a rule that generates a terminal node that represents an identifier or literal, it is represented using a special rule *IdentifierOrLiteral* to collapse all these rules into a single previous rule.

Two-step Attention At time step t, the decoder first attends to every token in the NL representation, h_i, using the current decoder state, s_t, to compute a set of attention weights α_t, which are used to combine h_i into an NL context vector z_t. We use a general attention mechanism (Luong et al., 2015), extended to perform multiple steps.

$$\alpha_{t,i} = \frac{\exp(s_t^{\mathrm{T}} \mathbf{F} h_i)}{\sum_i \exp(s_t^{\mathrm{T}} \mathbf{F} h_i)}$$
$$z_t = \sum_i \alpha_{t,i} h_i$$

The context vector z_t is used to attend over every type (return type) and variable (method) name in the environment, to produce attention weights β_t that are used to combine the entire context $x = [t : v : r : m]$ into an environment context vector e_t.[2]

$$\beta_{t,j} = \frac{\exp(z_t^{\mathrm{T}} \mathbf{G} x_j)}{\sum_j \exp(z_t^{\mathrm{T}} \mathbf{G} x_j)}$$
$$e_t = \sum_j \beta_{t,j} x_j$$

Finally, c_t is computed using the decoder state and both context vectors z_t and e_t:

$$c_t = tanh(\hat{W}[s_t : z_t : e_t])$$

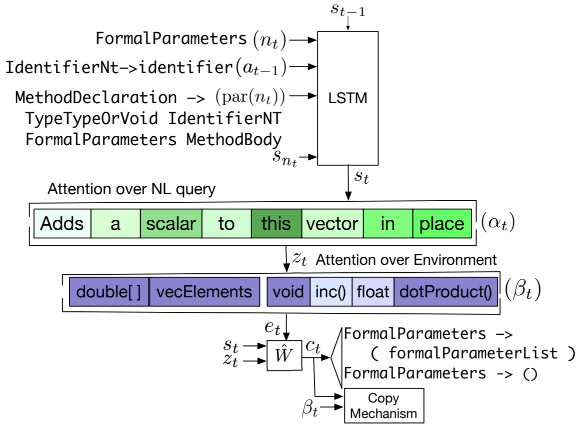

Figure 4: The hidden state s_t of our decoder is a function of the previous hidden state, current non-terminal, previous production rule, parent rule, and the parent hidden state. s_t is used to attend on the NL and compress it into z_t, which is then used to attend over the environment variables and methods to generate e_t. The decoder uses all these context vectors to produce a distribution over valid right hand side values of the current non-terminal, and also learns to copy from the environment.

Supervised Copy Mechanism Since the class environment at test time can belong to previously unseen new domains, our model needs to learn to copy variables and methods into the output. We use the copying technique of Gu et al. (2016a) to compute a copy probability at every time step t using vector b of dimensionality H.

$$\text{copy}(t) = \sigma(b^T c_t)$$

Since we only require named identifiers or user defined types to be copied, both of which are produced by production rules with n_t as *IdentifierNT*, we make use of this copy mechanism only in this case. Identifiers can be generated by directly generating derivation rules (see equation 1), or by copying from the environment. The probability of copying an environment token x_j, is set to be the attention weights $\beta_{t,j}$ computed earlier, which attend exactly on the environment types and names which we wish to be able to copy. The copying process is supervised by preprocessing the production rules to recognize identifiers that can be copied from the environment, and both the generation and the copy probabilities are weighted by $1 - \text{copy}(t)$ and $\text{copy}(t)$ respectively. The LSTM decoder with attention mechanism is illustrated in Figure 4.

3.3 Baseline Models

Retrieval We evaluate a retrieval baseline, where the output source code for a test example is the training example source code whose NL is closest

---

[2] ":" denotes concatenation.

in terms of cosine similarity to the test NL using a tf-idf representation. We then replace all occurrences of environment variables and methods in the chosen training source code with similarly typed variables and methods from the environment of the test example, and we break ties randomly.

Seq2seq We evaluate a Seq2Seq baseline by representing the NL and context as a sequence formed by the concatenation of the NL, the variables and the methods with separators between them. The variables (methods) are represented with the type (return type) followed by the name, with a different separator between them. The encoder is an n-layer LSTM which initializes an LSTM-based decoder using its final hidden states. The decoder uses an attention mechanism (Luong et al., 2015) over the encoder states to produce a conditional distribution over the next source code token (not production rule) given all the previous tokens. We replace UNK tokens in the output with source tokens having the most attention weight. We also attempted to evaluate the Seq2Tree model of Dong and Lapata (2016) but the redundancy in the model resulted in extremely long output sequences which did not scale. Experiments on a smaller dataset gave comparable results to Seq2seq.

Seq2prod This baseline corresponds to the action sequence model by Yin and Neubig (2017), with a BiLSTM over a sequence representation of the NL and context (same as Seq2seq), and a decoder that learns to generate a derivation of the AST of the underlying source code, similar to our model. The decoder uses the same attention mechanism as the Seq2seq, however, it uses supervised copying from the entire input sequence to handle unknown words encountered during testing.

4 CONCODE

We built a new dataset (CONCODE) from public Java projects on Github that contains environment information together with NL (Javadoc-style method comments) and code tuples. We gather Java files from $\sim$33,000 repositories, which are then split into train, development, and test sets based on repository, rather than purely randomly. Dividing based on the repository keeps the domains in the test set separate from the training set, therefore providing near zero-shot conditions that should truly test the ability of models to generalize to associate unseen NL tokens with previously unseen environment variables and methods. We also remove all examples from the development and test sets where the NL is exactly present in the training set. We further eliminate all Java classes that inherit from parent classes, since the resulting code may use variables and methods inherited from parent classes that reside in separate source files. While this is an important and interesting feature, we leave it for future work.

	Count
Train	100,000
Valid/Test	2,000/2,000
Average NL Length	13.73
Average Code Characters	119
Average Code Tokens	26.3
Average # Environment Variables	4.89
Average # Environment Methods	10.95
Average AST Nodes	79.6
# Node Types	153
# Production Rules	342*
% Getters	16.74%
% Setters	3.39%
% using Class Variables	68%
% using Class Methods	16.2%
% Unknown Types	7.65%

Table 1: Statistics of our dataset of (NL, context and code) tuples collected from Github. *Does not include rules that generate identifiers.

Every method that contains a Javadoc comment is treated as a training example. The Javadoc comment forms the NL, the method body is the target code to be generated, and all member variables, as well as other member method signatures are extracted and stored as part of the context. The Javadoc is preprocessed by stripping special symbols such as `@link`, `@code`, `@inheritDoc` and special fields such as `@params`. Methods that do not parse are eliminated and the rest are pre-processed by renaming locally defined variables canonically, beginning at *loc0* and similarly for arguments, starting with *arg0*. We also replace all method names with the word *function* since it doesn't affect the semantics of the resulting program. Generating informative method names has been studied by Allamanis et al. (2015a). We replace all string literals in the code with constants as they are often debug messages. Finally, we use an ANTLR java grammar[3] that is post-processed by adding additional non-terminals and rules to eliminate wildcard symbols in the grammar, in order to convert the source code into a sequence of production rules. The resulting dataset contains 100,000 examples for training, and 2000 examples each for development and testing, respectively. Table 1 summarizes the various statistics. We observe that on average, an environment contains $\sim$5 variables and $\sim$11 methods. Around 68% of the target code uses class member variables, and 16% of them use member methods, from the environment. Based on a frequency cutoff of 2 on the training set, we find that 7% of the types in the development set code are unknown, hence they need to be copied from the environment. Since CONCODE is extracted from a diverse set of online repositories, there is a high variety of code templates in the dataset compared to existing datasets. For example, a random baseline on the Hearthstone card game dataset (Ling et al., 2016) gives a BLEU score of 40.3, but only a score of 10.2 on CONCODE. We plan to release all code and data from this work.[4]

5 Experimental Setup

We restrict all models to examples where the length of the combination of the NL and the context is at most 200 tokens and the length of the output source code is at most 150 tokens. Source NL tokens are lower-cased, camel-cased identifiers are split and lower-cased, and are used together with the original version. The vocabulary for identifiers uses a frequency threshold of 7, resulting in a vocabulary of $32,600$ tokens. The types vocabulary uses a threshold of 2 resulting in $22,324$ types. We include all 153 non-terminals and 342 previous rules. We use a threshold of 2 for output production rules to filter out the long tail of rules creating identifiers and literals, resulting in $18,135$ output rules. Remaining values are replaced with the UNK symbol.

Hyperparameters We use an embedding size H of 1024 for identifiers and types. All LSTM cells use 2-layers and a hidden dimensionality of 1024 (512 on each direction for BiLSTMs). We use an embedding size of 512 for encoding non-terminals and rules in the decoder. We use dropout with $p = 0.5$ in between LSTM layers and at the output of the decoder over c_t. We train our model for 30 epochs using mini-batch gradient descent with a batch size of 20, and we use Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001 for optimization. We decay our learning rate by 80% based on performance on the development set after every epoch.

Model	Exact	BLEU
Retrieval	2.25 (1.65)	20.27 (18.15)
Seq2Seq	3.2 (2.9)	23.51 (21.0)
Seq2Prod	6.65 (5.55)	21.29 (20.55)
Ours	8.6 (7.05)	22.11 (21.28)

Table 2: Exact match accuracy and BLEU score (for partial credit) on the test (development) set, comprising 2000 examples from previously unseen repositories.

Model	Exact	BLEU
Ours	7.05	21.28
-Variables	1.6	20.78
-Methods	6.2	21.74
-Two step attention	5.75	17.2
-Camel-case encoding	5.7	21.83

Table 3: Ablation of model features on the development set.

Inference and Metrics Inference is done by first encoding the NL and context of the test example. We maintain a stack of symbols starting with the non-terminal, *MemberDeclaration*, and at each step, a non-terminal (terminals are added to the output) is popped off the stack to run a decoding step to generate the next set of symbols to push onto the stack. The set of terminals generated along the way forms the output source code. We use beam search and maintain a ranked list of partial derivations (or code tokens for Seq2seq) at every step to explore alternate high-probability derivations. We use a beam size of 3 for all neural models. We copy over source tokens whenever preferred by the model output. We restrict the output to 150 tokens or 500 production rules.

To evaluate the quality of the output, we use Exact match accuracy between the reference and generated code. As a measure of partial credit, we also compute the BLEU score (Papineni et al., 2002), following recent work on code generation (Ling et al., 2016; Yin and Neubig, 2017). BLEU is an n-gram precision-based metric that will be higher when more subparts of the predicted code match the provided reference.

---

[3] `https://github.com/antlr/grammars-v4`

[4] `https://github.com/sriniiyer/concode`

NL: Gets the value of the tags property. This accessor method returns a reference to the live list, not a snapshot.

Variables:
```
String validationPattern;
List<String> tags;
```
Methods:
```
String getValidationPattern
void setValidationPattern
```

Code:
```
List <String> function() {
  if ( tags == null ) {
    tags = new ArrayList <String>();}
  return this.tags; }
```

(a)

NL: Convert mixed case to underscores.

Variables:
```
NamingStrategy INSTANCE;
```
Methods:
```
String classToTableName
String collectionTableName
String tableName
String columnName
String addUnderscores
```

Code:
```
String function (String arg0) {
   return addUnderscores (arg0);
}
```

(b)

NL: Skips the next char

Variables:
```
String str;
int pos
```

Methods:
```
char ch()
int pos()
int length()
int gatherInt()
boolean hasNext()
```

Code:
```
void function () {
  pos ++ ;
}
```

(c)

NL: Returns the execution data store with data for all loaded classes.

Variables:
```
SessionInfoStore sessionInfos
ExecutionDataStore executionData
```

Methods:
```
void load
SessionInfoStore getSessionInfoStore
void save
```

Code:
```
ExecutionDataStore function () {
  return executionData ;
}
```

(d)

NL: Return a library that loads the core from code.jquery.com

Prediction:
```
JQueryLibrary function
   (String arg0) {
  return new JQueryLibrary ( arg0 ) ;
}
```
Reference:
```
JQueryLibrary function(String arg0) {
  return new JQueryLibrary (
      BASE_RESOURCE_URL + “string" +
        arg0 + “string" ) ;
}
```

(e)

NL: Empty the violations list

Variables:
```
long numPptEntries
List<Violation> violations
```

Prediction:
```
void function() {
  violations.clear() ;
}
```
Reference:
```
void function () {
 violations =
  new ArrayList<Violation>();
}
```

(f)

NL: Clear the update -timestamps data.

Variables:
```
boolean DEBUG_ENABLED
String REGION_NAME
TimestampsRegion region
```

Prediction:
```
void function() {
  region.clear() ;
}
```
Reference:
```
void function() {
  region.evictAll();
}
```

(h)

NL: Returns true if this registry maps one or more keys to the specified value.

Variables:
```
Map _values
Map _register
```
Methods:
```
IWidgetIdentifier get
WidgetLocator add
```

Prediction:
```
boolean function
  (IWidgetIdentifier arg0) {
  return _values.contains(arg0);
}
```
Reference:
```
boolean function
  (IWidgetIdentifier arg0) {
  return _register.containsValue(arg0);
}
```

(i)

Prediction:
```
List < ? > function ( ) {
  return iTransformers ; }
```

Reference:
```
Transformer [ ] function ( ) {
  return iTransformers ; }
```

(g)

Figure 5: Analysis of our model output on development set examples. Some environment variables and methods are omitted for space. (a)-(d) represent cases where the model exactly produced the reference output. (e)-(g) are cases where the model output is very reasonable for a practical setting. In (f), the model produces a better solution than the reference. In (h), the context lacks information to produce the reference code. The model chooses the wrong element in (i) and could be improved by better encoder representations.

6 Results

We present results for the context based code generation task on the test and dev sets in Table 2. Our model outperforms all baselines and achieves a gain of 1.95% exact match accuracy and 0.82 BLEU points, with respect to the next best models. The combination of independently encoding subword units and applying a two-step attention mechanism helps the model learn to better associate the correct variables/methods from the context and the language in the NL. Figure 5 (a) shows an example output of our model, which produces code structure intermixed with member variables (tags). In (b) our model learns to call method `addUnderscores` (an UNK in the vocabulary) with its correct return type (`String`). Similarly, in (d) our model also successfully learns to use a previously unseen type (`ExecutionDataStore`) when making use of the corresponding variable. (c) is an example of where the NL does not directly refer to the variable to be used. The mismatch between dev and test results is because we ensure that the dev and test examples come from non-overlapping Github repositories, resulting in different distributions.

Using a constrained decoder that generates syntax tree rules instead of tokens leads to significant gains in terms of exact match score (6.65 for Seq2prod vs 3.2 for Seq2seq), and shows that this is an important component of code generation systems. Seq2prod, however, fails on examples (b)-(d), since it is harder to learn to match the NL tokens with environment elements. Finally, all neural models outperform the retrieval baseline with member substitution.

To understand which components of the data and the model contribute most to the output, we perform an ablation study on the development set (Table 3). Removing the variables leads to a significant hit in exact match accuracy since 68% of examples (Table 1) refers to class variables; a similar but smaller reduction is incurred by removing methods. The presence of these components makes this task more challenging and also more aligned with programming scenarios in practice. Removing the two-step attention mechanism leads to a 1.3% drop in accuracy since the attention on the NL is unable to interact with the attention on the environment variables/methods. Removing camel-case encoding leads to a small drop mainly because many variable (method) names are single words.

Category	Fraction
Totally Wrong	62%
Marginally Correct	9%
Mostly Correct	16%
Exact Match	11%
Semantically Equivalent	2%

Table 4: Qualitative distribution of errors on the development set.

7 Error Analysis

Subfigures 5(e)-(i) show cases where our model output did not exactly match the reference. In (e)-(g), the model output is semantically equivalent to the reference and is a very reasonable prediction in a practical setting. For example, in (e) the only difference between the prediction and the reference is the string concatenations to the url. Interestingly, in example (f) the prediction is a cleaner way to achieve the same effect as the reference, and this is a great example of the application of these models for suggesting standard and efficient code. Unfortunately, our model is penalized by the exact match metric here. Similarly, in (g), the model uses a generic list (`List<?>`) in place of the specific type (`Transformer[]`). Example (h) demonstrates a case where the model is unaware of methods that can be called on class members (specifically that `evictAll` is a member of the `TimestampsRegion` class), and requires augmenting the environment with additional member type documentation, which we believe will be an important area for future work. Example (i) calls for richer encoder representations, since our model incorrectly uses the `_values` variable instead of `_register`, as it is unable to associate the word "registry" with the right elements.

We further perform a qualitative analysis of 100 predictions on our development set (Table 4) and find that there is significant room for improvement with 71% of the predictions differing significantly from their references. 16% of the predictions are very close to their references with the difference being 1-2 tokens, while 11% are exactly correct. 2% of the predictions were semantically equivalent but not exactly equal to their references.

8 Related Work

There is significant existing research on mapping NL directly to executable programs in the form of logical forms (Zettlemoyer and Collins, 2005), λ-DCS (Liang et al., 2013), regular expressions (Kushman and Barzilay, 2013; Locascio et al., 2016), database queries (Iyer et al., 2017; Zhong et al., 2017) and general purpose programs (Balog et al., 2016; Allamanis et al., 2015b). Ling et al. (2016) generate Java and Python source code from NL for card games, conditioned on categorical card attributes. Manshadi et al. (2013) generate code based on input/output examples for applications such as database querying. Gu et al. (2016b) use neural models to map NL queries to a sequence of API calls, and Neelakantan et al. (2015) augment neural models with a small set of basic arithmetic and logic operations to generate more meaningful programs. In this work, we introduce a new task of generating programs from NL based on the environment in which the generated code resides, following the frequently occurring pattern in large repositories where the code depends on the types and availability of variables and methods in the environment.

Neural encoder-decoder models have proved effective in mapping NL to logical forms and also for directly producing general purpose programs. Ling et al. (2016) use a sequence-to-sequence model with attention and a copy mechanism to generate source code. Instead of directly generating a sequence of code tokens, recent methods focus on constrained decoding mechanisms to generate syntactically correct output using a decoder that is either grammar-aware or has a dynamically-determined modular structure paralleling the structure of the abstract syntax tree (AST) of the code (Dong and Lapata, 2016; Rabinovich et al., 2017; Krishnamurthy et al., 2017; Yin and Neubig, 2017). Our model also uses a grammar-aware decoder similar to Yin and Neubig (2017) to generate syntactically valid parse trees, augmented with a two-step attention mechanism (Chen et al., 2016), followed by a supervised copying mechanism (Gu et al., 2016a) over the class environment.

Recent models for mapping NL to code have been evaluated on datasets containing highly templated code for card games (Hearthstone & MTG; Ling et al., 2016), or manually labeled per-line comments (DJANGO; Oda et al., 2015). These datasets contain $\sim$20,000 programs with short textual descriptions possibly paired with categorical data, whose values need to be copied onto the resulting code from a single domain. In this work, we collect a new dataset of over 100,000 NL and code pairs, together with the corresponding class environment. Each environment and NL describe a specific domain and the dataset comprises thousands of different domains, that poses additional challenges. Having an order of magnitude more data than existing datasets makes training deep neural models very effective, as we saw in the experimental evaluation. While massive amounts of Github code have been used before for creating datasets on source code only (Allamanis and Sutton, 2013, 2014; Allamanis et al., 2016), we instead extract from Github a dataset of NL and code with an emphasis on context, in order to learn to map NL to code within a class.

9 Conclusion

In this paper, we introduce new data and methods for learning to generate source code from language within the context of a real-world code base. To train models for this task, we collect and release CONCODE, a large new dataset of NL, code, and context tuples from online repositories, featuring code from a variety of domains. We also introduced a new encoder decoder model with a specialized context encoder which outperforms strong neural baselines by 1.95% exact match accuracy. Finally, analysis suggests that even richer models of programmatic context could further improve these results.

Acknowledgements

The research was supported in part by DARPA (FA8750-16-2-0032), the ARO (ARO-W911NF-16-1-0121), the NSF (IIS-1252835, IIS-1562364, IIS1546083, IIS-1651489, OAC-1739419), the DOE (DE-SC0016260), the Intel-NSF CAPA center, and gifts from Adobe, Amazon, Google, and Huawei. The authors thank the anonymous reviewers for their helpful comments.

References

Miltiadis Allamanis, Earl T Barr, Christian Bird, and Charles Sutton. 2015a. Suggesting accurate method and class names. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 38–49.

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *International Conference on Machine Learning*, pages 2091–2100.

Miltiadis Allamanis and Charles Sutton. 2013. Mining source code repositories at massive scale using language modeling. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 207–216. IEEE Press.

Miltiadis Allamanis and Charles Sutton. 2014. Mining idioms from source code. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 472–483. ACM.

Miltiadis Allamanis, Daniel Tarlow, Andrew Gordon, and Yi Wei. 2015b. Bimodal modelling of source code and natural language. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 2123–2132.

Matej Balog, Alexander L Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2016. Deepcoder: Learning to write programs. *arXiv preprint arXiv:1611.01989*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/-Daily Mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016a. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. 2016b. Deep api learning. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 631–642, New York, NY, USA. ACM.

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

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 826–836. Association for Computational Linguistics.

Percy Liang, I. Michael Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2).

Wang Ling, Phil Blunsom, Edward Grefenstette, Moritz Karl Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609. Association for Computational Linguistics.

Nicholas Locascio, Karthik Narasimhan, Eduardo De Leon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1918–1923, Austin, Texas. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and D. Christopher Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Mehdi Hafezi Manshadi, Daniel Gildea, and James F Allen. 2013. Integrating programming by example and natural language programming. In *AAAI*.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2015. Neural programmer: Inducing latent programs with gradient descent. *arXiv preprint arXiv:1511.04834*.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 574–584. IEEE.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 878–888, Beijing, China. Association for Computational Linguistics.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task

Tao Yu Michihiro Yasunaga Kai Yang Rui Zhang
Dongxu Wang Zifan Li Dragomir R. Radev
Department of Computer Science, Yale University
`{tao.yu, michihiro.yasunaga, k.yang, r.zhang, dragomir.radev}@yale.edu`

Abstract

Most existing studies in text-to-SQL tasks do not require generating complex SQL queries with multiple clauses or sub-queries, and generalizing to new, unseen databases. In this paper we propose SyntaxSQLNet, a syntax tree network to address the *complex* and *cross-domain* text-to-SQL generation task. SyntaxSQLNet employs a SQL specific syntax tree-based decoder with SQL generation path history and table-aware column attention encoders. We evaluate SyntaxSQLNet on a new large-scale text-to-SQL corpus containing databases with multiple tables and complex SQL queries containing multiple SQL clauses and nested queries. We use a database split setting where databases in the test set are unseen during training. Experimental results show that SyntaxSQLNet can handle a significantly greater number of complex SQL examples than prior work, outperforming the previous state-of-the-art model by 9.5% in exact matching accuracy. To our knowledge, we are the first to study this complex text-to-SQL task. Our task and models with the latest updates are available at https://yale-lily.github.io/seq2sql/spider.

1 Introduction

Text-to-SQL task is one of the most important subtask of semantic parsing in natural language processing (NLP). It maps natural language sentences to corresponding SQL queries.

In recent years, some state-of-the-art methods with Seq2Seq encoder-decoder architectures are able to obtain more than 80% exact matching accuracy on some complex text-to-SQL benchmarks such as ATIS and GeoQuery. These models seem to have already solved most problems in this area.

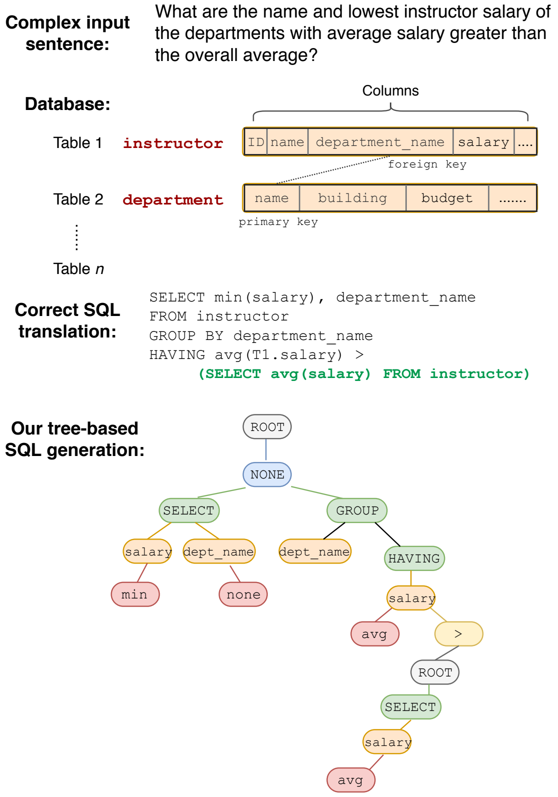

What are the name and lowest instructor salary of the departments with average salary greater than the overall average?

```
SELECT min(salary), department_name
FROM instructor
GROUP BY department_name
HAVING avg(T1.salary) >
     (SELECT avg(salary) FROM instructor)
```

Figure 1: To address the complex text-to-SQL generation task, SyntaxSQLNet employs a tree-based SQL generator. For example, our model can systematically generate a nested query as illustrated above.

However, as (Finegan-Dollak et al., 2018) show, because of the problematic task definition in the traditional datasets, most of these models just learn to match semantic parsing results, rather than truly learn to understand the meanings of inputs and generalize to new programs and databases. More specifically, most existing complex text-to-SQL datasets have less than 500 SQL labels. They are expanded by paraphrasing 4-10 questions for each SQL query. Under the standard train and test split (Zettlemoyer and Collins, 2005), most queries in the test set also appear in the train set. The WikiSQL dataset recently developed by (Zhong et al., 2017) is much larger and does use different databases for training and testing, but it only contains very simple SQL queries and

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1653–1663
Brussels, Belgium, October 31 - November 4, 2018.

database schemas.

To address those issues in the current semantic parsing datasets, Yu et al. (2018b) have developed a large-scale human labeled text-to-SQL dataset consisting of about 6,000 complex SQL queries and 200 databases with multiple tables. This dataset defines a new *complex* and *cross-domain* text-to-SQL task that requires models to generalize well to both new SQL queries and databases. The task cannot be solved easily without truly understanding the semantic meanings of the input questions.

In this paper, we propose SyntaxSQLNet, a SQL specific syntax tree network to address the aforementioned task. Specifically, to generate complex SQL queries with multiple clauses, selections and sub-queries, we develop a syntax tree-based decoder with SQL generation path history. To make our model learn to generalize to new databases with new tables and columns, we also develop a table-aware column encoder. Our contributions are as follows:

- We propose SQL specific syntax tree networks for the complex and cross-domain text-to-SQL task, which is even able to solve nested queries on new, unseen databases. We are the first to develop a methodology for this challenging semantic parsing task.
- We introduce a SQL specific syntax tree-based decoder with SQL path history and table-aware column attention encoders. Even with no hyper-parameter tuning, our model can significantly outperform the previous best models, with 4.8% boost in exact matching accuracy. Error analysis shows that our model is able to generalize, and solve much more complex (e.g., nested) queries in unseen databases than prior work.
- We also develop a cross-domain data augmentation method to generate more diverse training examples across databases, which further improves the exact matching accuracy by 7.5%. As a result, our model achieves 22.0% accuracy, a 12.3% total improvement compared with the previous best models.

2 Related Work

Semantic parsing maps natural language to formal meaning representations. There are a range of representations, such as logic forms and executable programs (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Das et al., 2010; Liang et al., 2011; Banarescu et al., 2013; Artzi and Zettlemoyer, 2013; Reddy et al., 2014; Berant and Liang, 2014; Pasupat and Liang, 2015).

As a sub-task of semantic parsing, the text-to-SQL problem has been studied for decades (Warren and Pereira, 1982; Popescu et al., 2003a, 2004; Li et al., 2006; Giordani and Moschitti, 2012; Wang et al., 2017b). The methods proposed in the database community (Li and Jagadish, 2014; Yaghmazadeh et al., 2017) tend to involve hand feature engineering and user interactions with the systems. In this work, we focus on recent neural network-based approaches (Yin et al., 2016; Zhong et al., 2017; Xu et al., 2017; Wang et al., 2017a; Iyer et al., 2017). Dong and Lapata (2016) introduce a sequence-to-sequence (seq2seq) approach to converting texts to logical forms. Most previous work focuses on a specific table schema. Zhong et al. (2017) publish the WikiSQL dataset and propose a seq2seq model with reinforcement learning to generate SQL queries. Xu et al. (2017) further improve the results on the WikiSQL task by using a SQL-sketch based approach employing a sequence-to-set model. Dong and Lapata (2018) propose a coarse-to-fine model which achieves the new state-of-the-art performances on several datasets including WikiSQL. Their model first generate a sketch of the target program. Then the model fills in missing details in the sketch.

Our syntax tree-based decoder is related to recent work that exploits syntax information for code generation tasks (Yin and Neubig, 2017; Rabinovich et al., 2017). Yin and Neubig (2017) introduce a neural model that transduces a natural language statement into an abstract syntax tree (AST). While they format the generation process as a seq2seq decoding of rules and tokens, our model uses a module for each grammar component, and calls them recursively to generate a SQL syntax tree. Similarly, Rabinovich et al. (2017) propose abstract syntax networks that use a collection of recursive modules for decoding. Our model differs from theirs in the following points. First, we exploit a SQL specific grammar instead of AST. AST-based models have to predict many non-terminal rules before predicting the terminal tokens, involving more steps. Whereas, our SQL-specific grammar enables direct prediction of SQL tokens. Second, our model uses different sequence-to-set modules to avoid the "ordering issue" (Xu et al., 2017) in many code generation tasks. Third, different from (Rabinovich et al., 2017), we pass a pre-order traverse of SQL decoding history to each module. This provides each module with important dependence information: e.g., if a SQL query has `GROUP BY`, it is very likely that the grouped column have appeared in `SELECT` too.

In addition to the distinction in model design, our work differs from theirs in the data and task definition. They aim to develop general syntax model for code generation via abstract syntax trees. Instead, we are interested in solving the complex and cross-domain SQL query generation problem; this motivates us to take advantage of SQL specific syntax for decoding, which guides systematic generation of complex SQL queries.

3 Problem Formulation

This work aims to tackle the complex text-to-SQL task that involves multiple tables, SQL clauses and nested queries. Further, we use separate databases for training and testing, aiming to develop models that generalize to new databases.

Dataset. We use *Spider* (Yu et al., 2018b) as the main dataset, which contains 10,181 questions, 5,693 unique complex SQL queries, and 200 databases with multiple tables.

Task and Challenges.

- The dataset contains a large number of complex SQL labels, which involve more tables, SQL clauses, and nested queries than prior datasets such as WikiSQL. Existing models developed for the WikiSQL task cannot handle those complex SQL queries in the Spider dataset.
- The dataset contains 200 databases (∼138 domains), and different databases are used for training and testing. Unlike most previous semantic parsing tasks (e.g., ATIS), this task requires models to generalize to new, unseen databases.

In sum, we train and test models on different complex SQL queries from different databases in this task. This aims to ensure that models can make the correct prediction only when they truly understand the meaning of the questions under the given database, rather than by mere memorization.

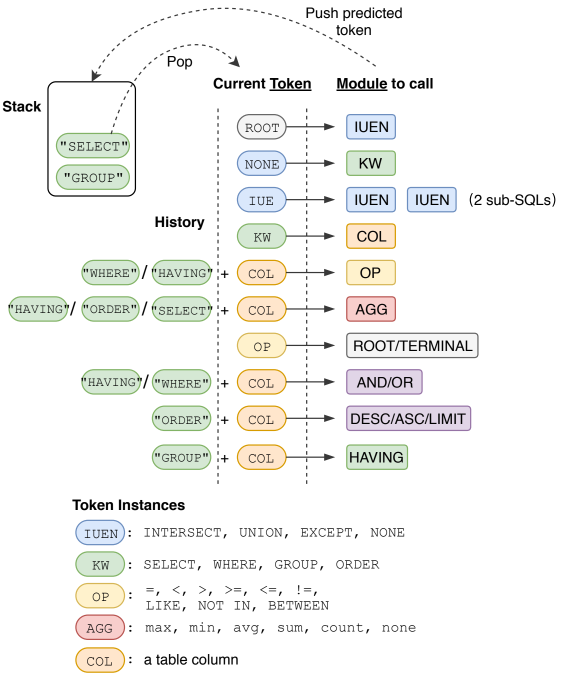

Figure 2: Our modules and SQL grammar used in decoding process. A round symbol represents a SQL tokens, a table column, etc. A square symbol indicates a module that predicts the next SQL token from its corresponding token instances with the same color.

4 Methodology

Similar to (Rabinovich et al., 2017), our model structures the decoder as a collection of recursive modules. However, as we discussed in the related work section, we make use of a SQL specific grammar to guide the decoding process, which allows us to take advantage of SQL queries' well-defined structure. [1]

4.1 Module Overview

Our model decomposes the SQL decoding process into 9 modules to handle the prediction of different SQL components such as keywords, operators, and columns. We provide the overview in this section and more details in later sections.

Figure 2 illustrates our modules and SQL grammar used in decoding process. A round symbol represents a SQL token, such as `SELECT`, `WHERE`, a table column, etc. A square symbol indicates a module that predicts the next SQL token from its corresponding token instances with the same color. Specifically, we have the following modules.

- **IUEN Module**, predicting `INTERCEPT`,

---

[1] Please check out our website for the latest updates on the model at `https://yale-lily.github.io/seq2sql/spider`

`UNION`, `EXCEPT`, and `NONE`, which determines if we need to call itself again to generate nested queries.

- **KW Module**, predicting keywords from `WHERE`, `GROUP BY`, and `ORDER BY`. All queries in our dataset have `SELECT`.
- **COL Module**, predicting table columns.
- **OP Module**, for $=$, $>$, $<$, $>=$, $<=$, $!=$, `LIKE`, `NOT IN`, `IN`, `BETWEEN`.
- **AGG Module**, predicting aggregators from `MAX`, `MIN`, `SUM`, `COUNT`, `AVG`, and `NONE`.
- **Root/Terminal Module**, predicting the `ROOT` of a new subquery or terminal value. It also enables our model to generate nested queries.
- **AND/OR Module**, predicting the presence of AND or OR operator between two conditions.
- **DESC/ASC/LIMIT Module**, predicting the keywords associated with `ORDER BY`. It is invoked only when `ORDER BY` is predicted before.
- **HAVING Module**, predicting the presence of `HAVING` for `GROUP BY` clause. It is invoked only when `GROUP BY` is predicted earlier.

4.2 SQL Grammar

In order to structure our decoder to generate complex queries, we consider a SQL grammar. It determines which module to be invoked at each recursive decoding step. Figure 2 illustrates our SQL grammar. During decoding process, given the current SQL token and the SQL history (the tokens we have gone over to reach the current token), we determine which module to invoke, and predict the next SQL token to generate.

To invoke some modules such as HAVING and OP during decoding, we not only check the type of current token instance but also see whether the type of the previously decoded SQL token is `GROUP` for HAVING module, and `WHERE` or `HAVING` for OP module.

In the grammar, IUEN and Root/Terminal modules are able to generate `ROOT`, which can activate IUEN module again. In this way, our model can recursively generate nested subqueries, and can also predict two or more subqueries in queries that have `EXCEPT`, `INTERSECT`, and `UNION`.

4.3 Input Encoder

Our inputs of each module consist of three types of information: question, table schema, and current SQL decoding history path. We encode a question sentence by a bi-directional LSTM, BiLSTM^{Q}. We encode table schema and history path in the manners described below.

4.3.1 Table-Aware Column Representation

In order to generalize to new databases in testing, it is important to make our model learn to obtain necessary information from a database schema. While SQLNet (Xu et al., 2017) only needed the column names as WikiSQL dataset only contained one table per question-SQL pair, *Spider*'s databases contain multiple tables. To address this setting, we propose to use both table and column names to construct column embeddings.

Specifically, given a database, we first obtain embedding for each table by taking the average embedding of the words constituting the table name (e.g., for table name `student_id`, we average the word embeddings for `student` and `id`). Next, for each column, we obtain the initial column name embedding in the same manner, and then concatenate the corresponding table embedding, and the type information of the column (string, or number, primary/foreign key) in a way similar to (Yu et al., 2018a) to produce a column embedding. On top, we apply another level of BiLSTM ($\text{BiLSTM}^{\text{COL}}$) that connects all columns in the database, to obtain high-level column embeddings. This way, our encoding scheme can effectively capture both the global (table names) and local (column names and types) information in the database schema to understand the question in the context of the given database.

Similarly to (Yu et al., 2018a), while the the order of column names or table names does not matter in practice, the use of BiLSTM performed better than the direct use of input column embeddings.

4.3.2 SQL Decoding History

In addition to question and column information, we also pass the SQL query's current decoding history as an input to each module. This enables us to use the information of previous decoding states to predict the next SQL token. For example, in Figure 1, the COL module would be more likely to predict `salary` in the subquery by considering the path history which contains `salary` for `HAVING`, and `SELECT` in the main query.

In contract, each module in SQLNet does not consider the previous decoded SQL history.

Hence, if directly applied to our recursive SQL decoding steps, each module would just predict the same output every time it is invoked. By passing the SQL history, each module is able to predict a different output according to the history every time it is called during the recursive SQL generation process. Also, the SQL history can improve the performance of each module on long and complex queries because the history helps the model capture the relations between clauses.

Predicted SQL history is used during test decoding. For training, we first traverse each node in the the gold query tree in pre-order to generate gold SQL path history for each training example used in different modules.

4.3.3 Attention for Input Encoding

For each module, like SQLNet, we apply the attention mechanism to encode question representation. We also employs this technique on SQL path history encoding. The specific formulas used are described in the next section.

4.4 Module Details

Similarly to SQLNet (Xu et al., 2017), we employ a sketch-based approach for each module. We apply a seq2set prediction framework introduced by (Xu et al., 2017), to avoid the order issue that happens in seq2seq based models for SQL generation. For example, in Figure 1, `SELECT salary, dept_name` is the same as `SELECT dept_name, salary`. The traditional seq2seq decoder generates each of them one by one in order; hence the model could get penalized even if the prediction and gold label are the same as sets. To avoid this problem, SQLNet predicts them together in one step so that their order does not affect the model's training process. For instance, in Figure 1, our model invokes the COL module to predict `salary` and `dept_name` and push to stack at the same time. However, SQLNet trains each modules independently, so no information passes through each component.

We first describe how to compute the conditional embedding $\mathbf{H}_{1/2}$ of an embedding $\mathbf{H}_1$ given another embedding $\mathbf{H}_2$:

$$\mathbf{H}_{1/2} = \mathbf{softmax}(\mathbf{H}_1 \mathbf{W} \mathbf{H}_2^\top)\mathbf{H}_1.$$

Here $\mathbf{W}$ is a trainable parameter. Moreover, we get a probability distribution from a given score matrix $\mathbf{U}$ by

$$\mathcal{P}(\mathbf{U}) = \mathbf{softmax}\left(\mathbf{V}\mathbf{tanh}(\mathbf{U})\right),$$

where $\mathbf{V}$ is a trainable parameter.

We denote the hidden states of LSTM on question embeddings, path history, and columns embeddings as $\mathbf{H}_{\text{Q}}$, $\mathbf{H}_{\text{HS}}$, and $\mathbf{H}_{\text{COL}}$ respectively. In addition, we denote the hidden states of LSTM on multiple keywords embeddings and keywords embeddings as $\mathbf{H}_{\text{MKW}}$ and $\mathbf{H}_{\text{KW}}$ respectively. Finally, we use $\mathbf{W}$ to denote trainable parameters that are not shared between modules. The output of each module is computed as follows:

IUEN Module In the IUEN module, since only one of the multiple keywords from {`INTERCEPT`, `UNION`, `EXCEPT`, `NONE`} will be used, we compute the probabilities by

$$P_{\text{IUEN}} = \mathcal{P}\left(\mathbf{W}_1\mathbf{H}_{\text{Q/MKW}}^\top + \mathbf{W}_2\mathbf{H}_{\text{HS/MKW}}^\top + \mathbf{W}_3\mathbf{H}_{\text{MKW}}^\top\right)$$

KW Module In the KW module, we first predict the number of keywords in the SQL query and then predict the keywords from {`SELECT`, `WHERE`, `GROUP BY`, `ORDER BY`}.

$$P_{\text{KW}}^{\text{num}} = \mathcal{P}\left(\mathbf{W}_1^{\text{num}}\mathbf{H}_{\text{Q/KW}}^{\text{num}}{}^\top + \mathbf{W}_2^{\text{num}}\mathbf{H}_{\text{HS/KW}}^{\text{num}}{}^\top\right)$$

$$P_{\text{KW}}^{\text{val}} = \mathcal{P}\left(\mathbf{W}_1^{\text{val}}\mathbf{H}_{\text{Q/KW}}^{\text{val}}{}^\top + \mathbf{W}_2^{\text{val}}\mathbf{H}_{\text{HS/KW}}^{\text{val}}{}^\top + \mathbf{W}_3^{\text{val}}\mathbf{H}_{\text{KW}}{}^\top\right)$$

COL Module Similarly, in the COL module, we first predict the number of columns in the SQL query and then predict which ones to use.

$$P_{\text{COL}}^{\text{num}} = \mathcal{P}\left(\mathbf{W}_1^{\text{num}}\mathbf{H}_{\text{Q/COL}}^{\text{num}}{}^\top + \mathbf{W}_2^{\text{num}}\mathbf{H}_{\text{HS/COL}}^{\text{num}}{}^\top\right)$$

$$P_{\text{COL}}^{\text{val}} = \mathcal{P}\left(\mathbf{W}_1^{\text{val}}\mathbf{H}_{\text{Q/COL}}^{\text{val}}{}^\top + \mathbf{W}_2^{\text{val}}\mathbf{H}_{\text{HS/COL}}^{\text{val}}{}^\top + \mathbf{W}_3^{\text{val}}\mathbf{H}_{\text{COL}}{}^\top\right)$$

OP Module In the OP module, for each predicted column from the COL module that is in the `WHERE` clause, we first predict the number of operators on it then predict which operators to use from $\{=, >, <, >=, <=, !=$, `LIKE`, `NOT IN`, `IN`, `BETWEEN`$\}$. We use $\mathbf{H}_{\text{CS}}$ to denote the embedding of one of the predicted columns from the COL module.

$$P_{\text{OP}}^{\text{num}} = \mathcal{P}\left(\mathbf{W}_1^{\text{num}}\mathbf{H}_{\text{Q/CS}}^{\text{num}}{}^\top + \mathbf{W}_2^{\text{num}}\mathbf{H}_{\text{HS/CS}}^{\text{num}}{}^\top + \mathbf{W}_3^{\text{num}}\mathbf{H}_{\text{CS}}{}^\top\right)$$

$$P_{\text{OP}}^{\text{val}} = \mathcal{P}\left(\mathbf{W}_1^{\text{val}}\mathbf{H}_{\text{Q/CS}}^{\text{val}}{}^\top + \mathbf{W}_2^{\text{val}}\mathbf{H}_{\text{HS/CS}}^{\text{val}}{}^\top + \mathbf{W}_3^{\text{val}}\mathbf{H}_{\text{CS}}{}^\top\right)$$

AGG Module In the AGG module, for each predicted column from the COL module that is in the `SELECT` clause, we first predict the number of aggregators on it then predict which aggregators to use from {`MAX`, `MIN`, `SUM`, `COUNT`, `AVG`, `NONE`}

$$P_{\text{AGG}}^{\text{num}} = \mathcal{P}\left(\mathbf{W}_1^{\text{num}}\mathbf{H}_{\text{Q/CS}}^{\text{num}}{}^\top + \mathbf{W}_2^{\text{num}}\mathbf{H}_{\text{HS/CS}}^{\text{num}}{}^\top + \mathbf{W}_3^{\text{num}}\mathbf{H}_{\text{CS}}{}^\top\right)$$

$$P_{\text{AGG}}^{\text{val}} = \mathcal{P}\left(\mathbf{W}_1^{\text{val}}\mathbf{H}_{\text{Q/CS}}^{\text{val}}{}^\top + \mathbf{W}_2^{\text{val}}\mathbf{H}_{\text{HS/CS}}^{\text{val}}{}^\top + \mathbf{W}_3^{\text{val}}\mathbf{H}_{\text{CS}}{}^\top\right)$$

Root/Terminal Module To predict nested subqueries, we add a module to predict if there is a new "ROOT" after an operator, which allows the model to decode queries recursively. For each predicted column from the COL module that is in the `WHERE` clause, we first call OP module, and then predict whether the next decoding step is a "ROOT" node or a value terminal node by

$$P_{\text{RT}} = \mathcal{P}\left(\mathbf{W}_1\mathbf{H}^{\top}_{\text{Q/CS}} + \mathbf{W}_2\mathbf{H}^{\top}_{\text{HS/CS}} + \mathbf{W}_3\mathbf{H}^{\top}_{\text{CS}}\right)$$

AND/OR Module For each condition column predicted from the COL module with number bigger than 1, we predict from {`AND`, `OR`} by

$$P_{\text{AO}} = \mathcal{P}\left(\mathbf{W}_1\mathbf{H}^{\top}_{\text{Q}} + \mathbf{W}_2\mathbf{H}^{\top}_{\text{HS}}\right)$$

DESC/ASC/LIMIT Module In the DAL module, for each predicted column from the COL module that is in the `ORDER BY` clause, we predict from {`DESC`, `ASC`, `DESC LIMIT`, `ASC LIMIT`} by

$$P_{\text{DAL}} = \mathcal{P}\left(\mathbf{W}_1\mathbf{H}^{\top}_{\text{Q/CS}} + \mathbf{W}_2\mathbf{H}^{\top}_{\text{HS/CS}} + \mathbf{W}_3\mathbf{H}^{\top}_{\text{CS}}\right)$$

HAVING Module In the HAVING module, for each predicted column from the COL module that is in the `GROUP BY` clause, we predict whether it is in the `HAVING` clause by

$$P_{\text{HAVING}} = \mathcal{P}\left(\mathbf{W}_1\mathbf{H}^{\top}_{\text{Q/CS}} + \mathbf{W}_2\mathbf{H}^{\top}_{\text{HS/CS}} + \mathbf{W}_3\mathbf{H}^{\top}_{\text{CS}}\right)$$

4.5 Recursive SQL Generation

The SQL generation process is a process of activating different modules recursively. As illustrated in Figure 2, we employ a stack to organize our decoding process. At each decoding step, we pop one SQL token instance from the stack, and invoke a module based on the grammar to predict the next token instance, and then push the predicted instance into the stack. The decoding process continues until the stack is empty.

More specifically, we initialize a stack with only `ROOT` at the first decoding step. At the next step, the stack pops `ROOT`. As illustrated in Figure 2, `ROOT` actives the `IUEN` module to predict if there is `EXCEPT`, `INTERSECT` or `UNION`. If so, there are two subqueries to be generated in the next step. If the model predicts `NONE` instead, it will be pushed into the stack. The stack pops `NONE` at next step. For example, in Figure 2, the current popped token is `SELECT`, which is a instance of keyword (KW) type. It calls the COL module to predict a column name, which will be pushed to the stack.

4.6 Data Augmentation

Even though *Spider* already has a significantly larger number of complex queries than existing datasets, the number of training examples for some complex SQL components is still limited. A widely used way is to conduct data augmentation to generate more training examples automatically. Many studies (Berant and Liang, 2014; Iyer et al., 2017; Su and Yan, 2017) have shown that data augmentation can bring significant improvement in performance. In prior work, data augmentation was typically performed within a single domain dataset. We propose a cross-domain data augmentation method to expand our training data for complex queries. Cross-domain data augmentation is more difficult than the in-domain setting because question-program pairs tend to have domain specific words and phrases. To tackle this issue, we first create a list of universal patterns for question-SQL pairs, based on the human labeled pairs from all the different training databases in *Spider*. To do so, we use a script to remove (and later fill in) all the table/column names and value tokens in the labeled question-SQL pairs, and then group together the same SQL query patterns. Consequently, each SQL query pattern has a list of about 5-20 corresponding questions. In our task, we want to generate more complex training examples. Thus, we filter out simple SQL query patterns by measuring the length and the number of SQL keywords used. We obtain about 280 different complex SQL query patterns from the 4,000+ SQL labels in the train set of our corpus. We then select the 50 most frequent complex SQL patterns that contain multiple SQL components and nested subqueries. After this, we manually edit the selected SQL patterns and their corresponding list of questions to make sure that the table/column/value slots in the questions have one-to-one correspondence to the slots in the corresponding SQL query. For each slot, we also add column type or table information. Thus, for example, columns with string type do not appear in the column slot with integer type during data augmentation (i.e., slot refilling) process. In this way, our question-SQL patterns are generated based on existing human labeled examples, which ensures that the generated training examples are natural.

Once we have the one-to-one slot mapping between questions and SQL queries, we apply a script that takes a new database schema with type information and generates new question-SQL examples by filling empty slots. Specifically, for each database in WikiSQL, we first randomly sample 10 question-SQL patterns. We randomly sample columns from the database schema based on its type: for example, if the slot type in the pattern is "number", and then we only sample from columns with "real" type in the current database. We then refill the slots in both the question and SQL query with the selected column names. Similarly, we also refill table/value slots.

By this data augmentation method, we finally obtain about 98,000 question and SQL pairs using about 18,000 WikiSQL databases with one single table. Additionally, we also use the original WikiSQL training dataset in experiments.

5 Experiments

Dataset In our experiments, we use Spider (Yu et al., 2018b), the new large-scale human annotated text-to-SQL dataset with complex SQL queries and cross-domain databases. In addition to their originally annotated data, their training split includes 752 queries and 1659 questions from six existing datasets: Restaurants (Tang and Mooney, 2001; Popescu et al., 2003b), GeoQuery (Zelle and Mooney, 1996), Scholar (Iyer et al., 2017), Academic (Li and Jagadish, 2014), Yelp and IMDB (Yaghmazadeh et al., 2017). In total, this dataset consists of 11,840 questions, 6,445 unique complex SQL queries, and 206 databases with multiple tables. We follow (Yu et al., 2018b), and use 130, 36, 40 databases for train, development, test, respectively (randomly split).

5.1 Evaluation

We use evaluation metrics including SQL Component matching and Exact matching. To compute the component matching scores, we first decompose predicted queries on SQL clauses including `SELECT`, `WHERE`, `GROUP BY`, `ORDER BY`, and `KEYWORDS` separately. After that, we evaluate each predicted clause and the ground truth as bags of several sub-components, and check whether or not these two sets of components match exactly. F1 scores are reported for each of the above 5 clauses. Exact matching score is 1 if the model predicts all clauses correctly for a given example.

To better understand model performance on different queries, (Yu et al., 2018b) divide SQL queries into 4 levels: easy, medium, hard, extra hard. The definition of difficulty is based on the number of SQL components, selections, and conditions. Queries that contain more SQL keywords are considered harder.

5.2 Experimental Settings

Our model is implemented in PyTorch (Paszke et al., 2017). We build each module based on the TypeSQL (Yu et al., 2018a) implementation. We use pre-trained GloVe (Pennington et al., 2014) embeddings for all question, SQL history, and schema tokens. All word embeddings are fixed. For each experiment, the dimension and dropout rate of all hidden layers is set to 120 and 0.3 respectively. We use Adam (Kingma and Ba, 2015) with the default hyperparameters for optimization, with a batch size of 64. The same loss functions in (Xu et al., 2017) are used.

- **Training data:** *Spider* (plus examples from 6 existing datasets) + WikiSQL + data augmentation
- **Model architecture:** history path + table-aware column encoding

We will conduct ablation studies to analyze the effect of each of the proposed techniques.

5.3 Baseline Models

To demonstrate the efficacy of our model in addressing the complex, cross-domain text-to-SQL task, we compare the performance of our model with several previous state-of-the-art models in the text-to-SQL task. As the dataset and task definition used in this work are fundamentally different from prior work using datasets such as GeoQuery, WikiSQL, we adapted these models to our task in the same way as (Yu et al., 2018b). Specifically:

Seq2Seq with Attention or Copying In order to make the models aware of the table schema information, Yu et al. (2018b) pass the models with a vocabulary that contains SQL keywords and column names of the given database.

(Iyer et al., 2017) Iyer et al. (2017) apply an attention based seq2seq model similar to (Luong et al., 2015) to text-to-SQL tasks. Yu et al. (2018b) adapt their model without user interaction to the task.

SQLNet & TypeSQL Xu et al. (2017) introduce SQLNet, which employs a column attention mechanism and a sketch-based method to generates SQL queries as a slot-filling task. Yu et al. (2018a) improves SQLNet by utilizing word types extracted from a knowledge graph or table content

Method	Easy	Medium	Hard	Extra Hard	All
Seq2Seq	17.9%	2.7%	1.3%	0.6%	5.4%
Seq2Seq+Attention (Dong and Lapata, 2016)	17.9%	2.9%	1.8%	1.3%	5.7%
Seq2Seq+Copying	15.1%	3.4%	1.0%	1.3%	5.2%
Iyer et al. (2017)	7.9%	2.1%	1.3%	1.1%	3.1%
SQLNet (Xu et al., 2017)	23.7%	5.9%	2.3%	0.3%	8.3%
TypeSQL (Yu et al., 2018a)	29.6%	6.1%	2.3%	0.3%	9.7%
SyntaxSQLNet	**43.1%**	**19.2%**	**17.8%**	**4.8%**	**22.0%**
-augment	36.9%	16.7%	9.5%	1.6%	17.6%
-wikiSQL -augment	34.1%	11.2%	9.0%	2.2%	14.5%
-table -wikiSQL -augment	30.1%	7.3%	4.5%	1.9%	10.9%
-history -table -wikiSQL -augment	18.8%	4.5%	0.0%	0.0%	6.1%

Table 1: Exact match accuracy on SQL queries with different hardness levels.

Method	`SELECT`	`WHERE`	`GROUP BY`	`ORDER BY`	`KEYWORDS`
Seq2Seq	13.7%	3.7%	3.2%	4.9%	8.9%
Seq2Seq+Attention	14.0%	5.0%	3.2%	6.1%	9.4%
Seq2Seq+Copying	12.0%	2.7%	5.2%	6.9%	6.7%
Iyer et al. (2017)	6.3%	1.9%	3.0%	3.6%	3.5%
SQLNet	24.0%	18.0%	11.8%	47.1%	61.9%
TypeSQL	36.2%	14.7%	6.4%	49.5%	59.4%
SyntaxSQLNet	**48.2%**	**31.6%**	**28.9%**	**58.4%**	**68.9%**
-augment	45.3%	29.2%	22.0%	47.3%	67.5%
-wikiSQL -augment	39.0%	20.6%	17.1%	50.2%	65.8%
-table -wikiSQL -augment	29.0%	15.0%	13.1%	45.5%	67.0%
-history -table -wikiSQL -augment	18.6%	10.5%	4.1%	30.0%	49.1%

Table 2: F1 scores of Component Matching on all SQL queries.

to help the model better understand entities and numbers in natural language inputs. As they are originally designed for WikiSQL, to conduct experiments on Spider, Yu et al. (2018b) extend their `SELECT` and `WHERE` modules to other SQL components.

6 Results and Discussion

Table 1 presents our test results on the *Spider* dataset with database splitting. Our model with full history and data augmentation achieves 22.0% exact matching on all SQL queries, which is a 12.3% absolute increase compared to the previous best model, TypeSQL.

6.1 Comparison with Baselines

Even though our individual modules are similar to SQLNet and TypeSQL, our syntax-aware decoder allows the modules to incorporate the SQL decoding history, which helps to achieve a significant gain in exact matching for queries of all hardness levels. Specifically, even without our data augmentation technique, SyntaxSQLNet outperforms the previous best, TypeSQL, by 5%. This result suggests that the syntax and history information is beneficial for this complex text-to-SQL task.

Moreover, the tree-based decoder enables SyntaxSQLNet to systematically generate nested queries, boosting the performance for Hard/Extra Hard. As Table 1 shows, SyntaxSQLNet achieves particularly high scores 17.8% and 4.8% for Hard and Extra Hard, which contain nested queries. The Seq2Seq models suffer from generating ungrammatical queries, yielding very low exact matching accuracy on Hard and Extra Hard SQL queries. In contrast, our model generates valid SQL queries by enforcing the syntax.

For the detailed component matching results in Table 2, our model consistently outperforms other previous work by significant margins. Specifically, our model improve F1 score for most of the SQL components by more than 10%.

6.2 Ablation Study

In order to understand the techniques that are responsible for the performance of our model, we perform an ablation study where we remove one of the proposed techniques from our model at a time. The exact match scores are shown in the same tables as other baselines.

6.2.1 Data Augmentation

Our model's exact matching performance on all queries drops 4.4% by excluding data augmentation technique. This drop is particularly large for `GROUP BY` and `ORDER BY` components (Table 2), for which the original Spider dataset has a relatively small number of training examples. Thus, our cross-domain data augmentation technique significantly benefits the model performance by extending to a much larger training dataset.

6.2.2 SQL Decoding History

In order to gain more insight into how our SQL decoding history addresses complex SQL, we report our model's performance without SQL path history. As shown in the Table 1, the model's performance drops about 4.8% on exacting matching metric without the SQL history input. More importantly, its performance on hard and extra hard SQL queries decreases to 0%. This indicates that our model is able to predict nested queries thanks to the SQL decoding history.

6.2.3 Column Encoding

To see how our table-aware column encoding affects performance of our model, we also report the model's result without using table information for our column encoding. After excluding the table embedding from column embeddings, the test performance goes down by about 4%. This drop is especially large for Medium/Hard SQL queries and `SELECT` component, where the correct column prediction is a key. This result shows that the table-aware encoding is very important to predict the correct columns in unseen, complex dasebases (with many foreign keys).

7 Conclusion

In this paper, we presented a syntax tree-based model to address complex and cross-domain text-to-SQL task. Utilizing a SQL specific syntax decoder, as well as SQL path history and table-aware column attention encoders, our model outperforms previous work by a significant margin. The ablation study demonstrates that our proposed techniques are able to predict nested, complex SQL queries correctly even for unseen databases.

Acknowledgement

We thank Graham Neubig, Tianze Shi, and three anonymous reviewers for their helpful feedback and discussion on this work.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association forComputational Linguistics*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic frame-semantic parsing. In *NAACL*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan Dhanalakshmi Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *ACL 2018*. Association for Computational Linguistics.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to sql queries with generative parsers discriminatively reranked. In *COLING (Posters)*, pages 401–410.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *CoRR*, abs/1704.08760.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *The 3rd International Conference for Learning Representations, San Diego*.

Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *VLDB*.

Yunyao Li, Huahai Yang, and HV Jagadish. 2006. Constructing a generic natural language interface for an xml database. In *EDBT*, volume 3896, pages 737–754. Springer.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NIPS 2017 Workshop*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543. ACL.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *Proceedings of the 20th international conference on Computational Linguistics*, page 141. Association for Computational Linguistics.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003a. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003b. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *ACL (1)*, pages 1139–1149. Association for Computational Linguistics.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. *CoRR*, abs/1704.05974.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *ECML*, volume 1, pages 466–477. Springer.

Chenglong Wang, Marc Brockschmidt, and Rishabh Singh. 2017a. Pointing out sql queries from text. *Technical Report*.

Chenglong Wang, Alvin Cheung, and Rastislav Bodik. 2017b. Synthesizing highly expressive sql queries from input-output examples. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 452–466. ACM.

David HD Warren and Fernando CN Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics*, 8(3-4):110–122.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-2007)*, Prague, Czech Republic.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: Query synthesis from natural language. *Proc. ACM Program. Lang.*, 1(OOPSLA):63:1–63:26.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural enquirer: Learning to query tables in natural language. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2308–2314.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *ACL (1)*, pages 440–450. Association for Computational Linguistics.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In

Proceedings of NAACL. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR. AAAI Press/MIT Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Cross-lingual Decompositional Semantic Parsing

Sheng Zhang
Johns Hopkins University

Xutai Ma
Johns Hopkins Univeristy

Rachel Rudinger
Johns Hopkins University

Kevin Duh
Johns Hopkins University

Benjamin Van Durme
Johns Hopkins University

Abstract

We introduce the task of cross-lingual decompositional semantic parsing: mapping content provided in a source language into a decompositional semantic analysis based on a target language. We present: (1) a form of decompositional semantic analysis designed to allow systems to target varying levels of structural complexity (shallow to deep analysis), (2) an evaluation metric to measure the similarity between system output and reference semantic analysis, (3) an end-to-end model with a novel annotating mechanism that supports intra-sentential coreference, and (4) an evaluation dataset on which our model outperforms strong baselines by at least 1.75 F_1 score.

1 Introduction

We are concerned here with representing the semantics of multiple natural languages in a single semantic analysis. Renewed interest in semantic analysis has led to a surge of proposed new frameworks, e.g., GMB (Basile et al., 2012), AMR (Banarescu et al., 2013), UCCA (Abend and Rappoport, 2013), and UDS (White et al., 2016), as well as further calls to attend to existing efforts, e.g., Episodic Logic (EL) (Schubert and Hwang, 2000; Schubert, 2000; Hwang and Schubert, 1994; Schubert, 2014), or Discourse Representation Theory (Kamp, 1981; Heim, 1988).

Many of these efforts are limited to the analysis of English, but with a number of exceptions, e.g., recent efforts by Bos et al. (2017), ongoing efforts in Minimal Recursion Semantics (MRS) (Copestake et al., 1995), multilingual FrameNet annotation and parsing (Fung and Chen, 2004; Padó and Lapata, 2005), among others. For many languages, semantic analysis cannot be performed directly, owing to a lack of training data. While there is active work in the community focused on rapid construction of resources for low resource languages (Strassel and Tracey, 2016), it remains an expensive and perhaps infeasible solution to assume in-language annotated resources for developing semantic parsing technologies. In contrast, bitext is easier to get: it occurs often without researcher involvement,[1] and even when not available, it may be easier to find bilingual speakers that can translate a text, than it is to find experts that will create in-language semantic annotations. In addition, we are simply further along in being able to automatically understand English than we are other languages, resulting from the bias in investment in English-rooted resources.

Therefore, we propose the task of cross-lingual decompositional semantic parsing, which aims at transducing a sentence in the source language (e.g., Chinese sentence in Fig. 1b) into a decompositional semantic analysis derived based on English, via bitext. The efforts of decompositional semantics (White et al., 2016) focus on approaches to annotating meaning based on fine-grained scalar judgments which reflect the ambiguity of language, and the underspecification of meaning in context. Our contributions include:

(1) A form of decompositional semantic analysis allowing systems to target varying levels of structural complexity.

(2) An evaluation metric to measure the similarity between system and reference semantic analysis.

(3) An encoder-decoder model for cross-lingual decompositional semantic parsing. With a coreference annotating mechanism, the model solves intra-sentential coreference explicitly.

(4) The first evaluation dataset for cross-lingual decompositional semantic parsing.[2]

Experiments demonstrate our model achieves 38.78% F_1, outperforming strong baselines.

---

[1] For example, owing to a government decree.

[2] `http://decomp.io`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1664–1675
Brussels, Belgium, October 31 - November 4, 2018.

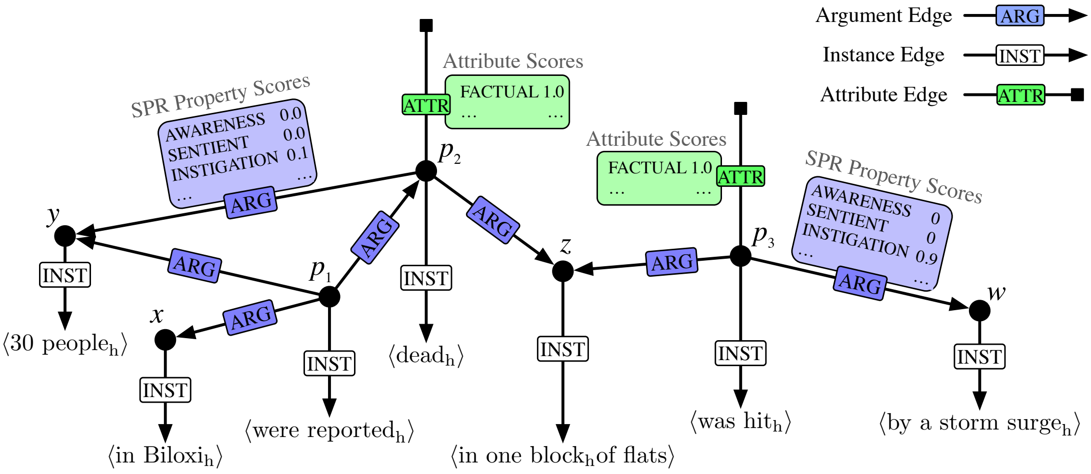

(a) UDS graph representation.

据	报道	，	在	比洛克西	，	三十	人	死	于	一	栋	被	风暴	潮	袭击	的	公寓楼
were	reported	,	in	Biloxi	,	30	people	dead	in	one	block	PTCP	storm	surge	hit	which	flats

"*30 people were reported dead in one block of flats which was hit by a storm surge*."

(b) Chinese sentence with Leipzig gloss.

Figure 1: Input and output of cross-lingual decompositional semantic parsing.

2 Semantic Analysis

The goal of cross-lingual decompositional semantic parsing is to provide a semantic analysis which can be used for various types of deep and shallow processing on the target language side. Many forms of semantic analysis are potentially suitable for this goal, e.g., AMR (Banarescu et al., 2013), UCCA (Abend and Rappoport, 2013), and Universal Decompositional Semantics (White et al., 2016). Here we choose Universal Decompositional Semantics (UDS), but note that our approach is applicable to other potential graph semantic formalisms.

The reasons for choosing UDS are three-fold: (1) **Compatibility**: UDS relates to Robust Minimal Recursion Semantics (RMRS) (Copestake, 2007), aiming for a maximal degree of semantic compatibility. With UDS, shallow analysis, such as predicate-argument extraction (Zhang et al., 2017a), can be regarded as producing a semantics which is underspecified and reusable with respect to deeper analysis, such as lexical semantics and inference (White et al., 2016). (2) **Robustness and Speed**: There exists a robust framework, PredPatt (White et al., 2016), for automatically creating UDS from raw sentences and their Universal Dependencies. PredPatt has been shown to be fast and accurate enough to process large volumes of text (Zhang et al., 2017c). (3) **Cross-lingual validity**: PredPatt is based purely on non-lexical and linguistically well-founded patterns from Universal Dependencies, which is designed to be cross-linguistically consistent.

There are three forms to represent UDS: flat, graph, or linearized representations. They are created for different purposes, and are inter-convertible. Flat representation relates to RMRS (Copestake, 2007), and we defer its description to Appendix A.

2.1 Graph Representation

The graph representation as shown in Fig. 1a is developed to improve ease of readability, parser evaluation, and integration with lexical semantics. The structure of the graph representation is a tuple $\mathcal{G} = \langle V, E \rangle$: a set of variables V (e.g., p_1 and x), and a set of edges E. There are 3 types of edges: (1) Argument edges describe argument relations between variable pairs. Deeper analysis such as Semantic Proto-Role (SPR) properties (Reisinger et al., 2015) can be attached to argument edges. SPR analysis can be considered as a scalar regression problem (White et al., 2016), where each predicate-argument pair is annotated with scalar values for different SPR properties. (2) Instance

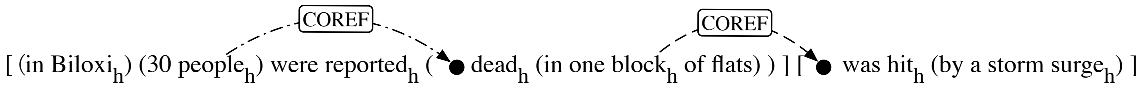

Figure 2: UDS linearized representation. Deeper analysis such as SPR and factuality is not shown.

edges describe instances of variables in the target language. The subscript "h" indicates the syntactic head of an instance. (3) Attribute edges are unary, which describe various attributes of variables, such as event factuality (Saurí and Pustejovsky, 2009) and word senses (Miller, 1995). The graph representation can be viewed as an underspecified version of Dependency Minimal Recursion Semantics (DMRS) (Copestake, 2009) due to the underspecification of scope. Different from DMRS, the graph representation is linked cleanly to Universal Dependency syntax via PredPatt.

2.2 Linearized Representation

The linearized representation aims to facilitate learning of semantic parsers. Recently parsers based on RNN that make use of linearized representation have achieved state-of-the-art performance in constituency parsing (Choe and Charniak, 2016), logical form prediction (Dong and Lapata, 2016; Jia and Liang, 2016), and AMR parsing (Barzdins and Gosko, 2016; Peng et al., 2017). There was also work on predicting linearized semantic representations before RNN based approaches (Wong and Mooney, 2006).

Fig. 2 shows an example of UDS linearized representation. Intra-sentential coreference occurs when an instance refers to an antecedent, where we replace the instance with a special symbol "•" and add a COREF link between "•" and its antecedent. The linearized representation can be viewed as a sequence of tokens with a list of COREF links. Brackets, parentheses, and the special symbol "•" are all considered as tokens in this representation. The COREF links are drawn as a visual convenience, and the actual linearized representation achieves this via co-indexing, and is thus fully linear. We describe the procedure of converting graph representation to linearized representation in Appendix B.

3 Related Work

Our work synthesizes two strands of research, semantic analysis and cross-lingual learning.

The semantic analysis targeted in this work is akin to that of Hobbs (2003), but our eventual goal is to transduce texts from arbitrary human languages into a *"...broad, language-like, inference-enabling [semantic representation] in the spirit of Montague..."* (Schubert, 2015). Unlike efforts such as by Schubert and colleagues that directly target such an analysis, we are pursuing a strategy that incrementally increases the complexity of the target analysis in accordance with our ability to fashion models capable of producing it.[3] Embracing underspecification in the name of tractability is exemplified by MRS (Copestake et al., 2005; Copestake, 2009), the so-called *slacker semantics*, and we draw inspiration from that work. Analyses such as AMR (Banarescu et al., 2013) also make use of underspecification, but usually this is only implicit: certain aspects of meaning are simply not annotated. Unlike AMR, but akin to decisions made in PropBank (Palmer et al., 2005) (which forms the majority of the AMR ontological backbone), we target an analysis with a close correspondence to natural language syntax. Unlike interlingua (Mitamura et al., 1991; Dorr and Habash, 2002) that maps the source language into an intermediate analysis, and then maps it into the target language, we are not concerned with generating text from the semantic analysis. Substantial prior work on semantic analyses exists, including HPSG-based analyses (Copestake et al., 2005), CCG-based analyses (Steedman, 2000; Baldridge and Kruijff, 2002; Bos et al., 2004), and Universal Dependencies based analyses (White et al., 2016; Reddy et al., 2017). See (Schubert, 2015; Abend and Rappoport, 2017) for further discussion.

Cross-lingual learning has previously been applied to various NLP tasks. Yarowsky et al. (2001); Padó and Lapata (2009); Evang and Bos (2016); Faruqui and Kumar (2015) focused on projecting existing annotations on source-language text to the target language. Zeman and Resnik (2008); Ganchev et al. (2009); McDonald et al. (2011); Naseem et al. (2012); Wang and Manning (2014) enabled model transfer by shar-

---

[3] E.g., in Fig. 1a we recognize "*by a storm surge*" as an initial structural unit, with multiple potential analysis, which may be further refined based on the capabilities of a given cross-lingual semantic parser.

ing features or model parameters for different languages. Sudo et al. (2004); Zhang et al. (2017a,b); Mei et al. (2018) worked on cross-lingual information extraction and demonstrated the advantages of end-to-end learning. In this work, we explore end-to-end cross-lingual learning.

4 Evaluation Metric $\boldsymbol{S}$

UDS can be represented in three forms. Evaluating such forms is crucial to the development of parsing algorithms. However, there is no method directly available for evaluation. Related methods come from semantic parsing, whose results are mainly evaluated in three ways: (1) task correctness (Tang and Mooney, 2001), which evaluates on a specific NLP task that uses the parsing results; (2) whole-parse correctness (Zettlemoyer and Collins, 2005), which counts the number of parsing results that are completely correct; and (3) Smatch (Cai and Knight, 2013), which computes the number of exactly matched edges between two semantic structures.

Nevertheless, our task needs an evaluation metric that can be used regardless of specific tasks or domains, and is able to differentiate two UDS graph representations with similar instances, SPR analysis, or attributes. We design an evaluation metric $\boldsymbol{S}$ that computes the similarity between two graph representations.

As described in Section 2.1, the graph representation is a tuple $\mathcal{G} = (V, E)$. For two graphs $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$, we define the score $\boldsymbol{S}$ as the maximum soft edge matching score between $\mathcal{G}_1$ and $\mathcal{G}_2$:

$$\boldsymbol{S}(\mathcal{G}_1, \mathcal{G}_2) = \max_{m \in \mathcal{M}} \Big[\sum_{(e_1^{(i)}, e_2^{(j)}) \in \mathcal{P}} f_T(e_1^{(i)}, e_2^{(j)}) \Big]$$

where m is a mapping from variables in V_1 to variables in V_2. Given a mapping m, $\mathcal{P}$ is a set of edge pairs: for each pair $(e_1^{(i)}, e_2^{(j)})$, variables(s) in $e_1^{(i)}$ are mapped to variables(s) in $e_2^{(j)}$ via m. f_T computes the matching score for a pair of edges belonging to type $T \in \{\text{ARG, INST, ATTR}\}$. The matching score is normalized to $[0, 1]$.

The precision and recall are computed by $\boldsymbol{S}(\mathcal{G}_1, \mathcal{G}_2)/|E_1|$, and $\boldsymbol{S}(\mathcal{G}_1, \mathcal{G}_2)/|E_2|$ respectively.

In this work, $f_{\text{ARG}} = f_{\text{ATTR}} = e^{-\text{MAE}}$, where MAE computes the mean absolute error between two set of scores s_1 and s_2: $\sum_i^n |s_1^{(i)} - s_2^{(i)}|/n$. $f_{\text{INST}} = \text{BLEU}$ (Papineni et al., 2002) which compute the BLEU score of an instance pair.[4]

Finding an optimal variable mapping m that yields the highest $\boldsymbol{S}$ is NP-complete. We instead adopt a strategy used in Smatch (Cai and Knight, 2013) that does a hill-climbing search with smart initialization plus 4 random restarts, and has been shown to give the best trade-off between accuracy and speed. Smatch for evaluating semantic structures can be considered as a special case of $\boldsymbol{S}$, where $f_T = \delta$, the Kronecker delta.

5 Model

We formulate the task of cross-lingual decompositional semantic parsing as a joint problem of sequence-to-sequence learning, coreference resolution and decompositional semantic analysis. The input is a sentence X in the source language, e.g., the Chinese sentence in Fig. 1b. The output is a UDS linearized representation (Y, C, D) based on the target language: Y is a sequence of tokens; C is a set of COREF links; and D is a set of scores for decompositional analysis, such as SPR and factuality.

The goal is to learn a conditional probability distribution $P(Y, C, D|X)$ whose most likely configuration, given the input sentence, outputs the true UDS linearized representation with decompositional analysis. While the standard encoder-decoder framework shows the state-of-the-art performance in sequence-to-sequence learning (Choe and Charniak, 2016; Jia and Liang, 2016; Barzdins and Gosko, 2016), it cannot directly solve intra-sentential conference and decompositional semantic analyses in our task. To achieve this goal, we propose an encoder-decoder architecture incorporated with a *coreference annotating* mechanism[5] and *decompositional analysis*. As illustrated in Fig. 3, **Encoder** transforms the input sequence into hidden states; **Decoder** reads the hidden states, and then at each time step generates a token and creates its COREF link; **Decompositional Analysis**, based on the decoder output, performs SPR analysis for predicate-argument pairs, and factuality analysis for predicates.

---

[4] Future work could consider, e.g., a modified BLEU that considers Levenshtein distance between tokens for a more robust partial-scoring in the face of transliteration errors.

[5] Similar coreference mechanism has been proposed by Ji et al. (2017).

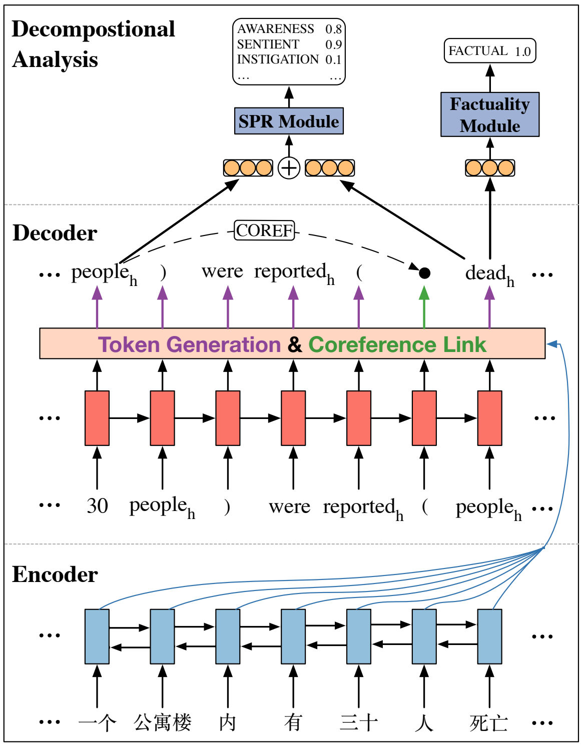

Figure 3: Illustration of the model architecture.

5.1 Encoder

The encoder employs a bidirectional recurrent neural network (Schuster and Paliwal, 1997) with LSTM units (Hochreiter and Schmidhuber, 1997). It encodes the input $X = x_1, \ldots, x_N$[6] into a sequence of hidden states $\boldsymbol{h} = h_1, \ldots, h_N$. Each hidden state h_i is a concatenation of a left-to-right hidden state $\overrightarrow{h_i}$ and a right-to-left hidden state $\overleftarrow{h_i}$,

5.2 Decoder

Given the encoder hidden states, the decoder predicts the linearized representation (as shown in Fig. 2) according to the conditional probability $P(Y, C \mid X)$ which is decomposed as a product of the decoding probabilities at each time step t:

$$P(Y, C \mid X) = \prod_{t=1}^{M} P(y_t, c_t \mid y_{<t}, c_{<t}, X) \quad (1)$$

where y_t is the decoded token at time step t, and c_t is the source of the COREF link for y_t, i.e., the antecedent of y_t. The set of possible antecedents of y_t is $\mathcal{A}(t) = \{\epsilon, y_1, \ldots, y_{t-1}\}$: a dummy antecedent ϵ and all preceding tokens. ϵ represents a scenario, where the token is not a special symbol "•", and it refers to none of the preceding tokens. $y_{<t}$ and $c_{<t}$ are the preceding tokens and their antecedents. We omit $y_{<t}$ and $c_{<t}$ from the notation when the context is unambiguous.

The decoding probability at each time step t is decomposed as

$$P(y_t, c_t) = P(y_t)P(c_t|y_t) \quad (2)$$

where $P(y_t)$ is the token generation probability, and $P(c_t|y_t)$ is the antecedent probability.

Token Generation: The probability distribution of the generated token y_t is defined as

$$P(y_t) = \text{softmax}(\text{FFNN}_\text{g}(s_t, a_t)) \quad (3)$$

where FFNN_g is a two-layer feed-forward neural network over the decoder hidden state s_t and the attention-weighted vector a_t. s_t is computed by

$$s_t = \text{RNN}(y_{t-1}, s_{t-1}), \quad (4)$$

where RNN is a recurrent neural network using LSTM. a_t is computed by the attention mechanism (Bahdanau et al., 2014; Luong et al., 2015),

$$a_t = \sum_{i}^{N} \alpha_{t,i} h_i, \quad (5)$$

$$\alpha_{t,i} = \frac{\exp\left(s_t^\top (W_a h_i + b_a)\right))}{\sum_{j=1}^{N} \exp\left(s_t^\top (W_a h_j + b_a)\right)}, \quad (6)$$

where W_a is a transform matrix and b_a is a bias.

Coref Link: The probability of y_t referring to the preceding token y_k, i.e., $c_t = y_k$, is defined as

$$P(c_t = y_k|y_t) = \frac{\exp\left(\text{SCORE}(y_t, y_k)\right)}{\sum_{y'_k \in \mathcal{A}(t)} \exp\left(\text{SCORE}(y_t, y_{k'})\right)}, \quad (7)$$

$\text{SCORE}(y_t, y_k)$ is a pairwise score for a COREF link from y_k to y_t, defined as:

$$\text{SCORE}(y_t, y_k) = s_\text{c}(y_t) + s_\text{p}(y_k) + s_\text{a}(y_t, y_k) \quad (8)$$

There are three factors in this pairwise score, which is akin to Lee et al. (2017): (1) $s_\text{c}(y_t)$, whether y_t should refer to a preceding instance; (2) $s_\text{p}(y_k)$, whether y_k shoud be a candidate source of such a coreference; and (3) $s_\text{a}(y_t, y_k)$, whether y_k is an antecedent of y_t.

Fig. 4 shows the details of the scoring architecture. At the core of the three factors are vector representations $\gamma(y_t)$ for each token y_t, which is described in detail in the following section. Given

---

[6] For simplicity, we use X (and Y) to represent both tokens as well as their word embeddings.

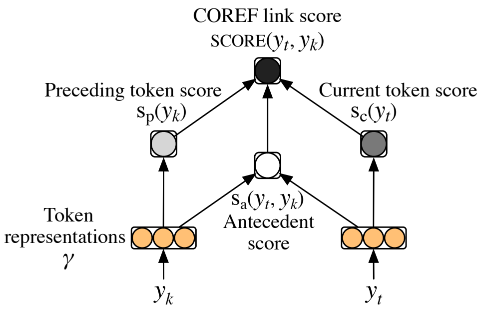

Figure 4: Scoring architecture in the copy mechanism between a preceding token y_k and the currently considered token y_t.

the currently considered token y_t and a preceding token y_k, the scoring functions above are computed via standard feed-foward neural networks:

$$s_{\text{c}}(y_t) = w_{\text{c}} \cdot \text{FFNN}_{\text{c}}(\gamma(y_t)) \quad (9)$$

$$s_{\text{p}}(y_k) = w_{\text{p}} \cdot \text{FFNN}_{\text{p}}(\gamma(y_k)) \quad (10)$$

$$s_{\text{a}}(y_t, y_k) = w_{\text{a}} \cdot \text{FFNN}_{\text{a}}([\gamma(y_t), \gamma(y_k), \gamma(y_t) \circ \gamma(y_k)]) \quad (11)$$

where $\cdot$ denotes dot product, $\circ$ denotes element-wise multiplication, and FFNN denotes a two-layer feed-foward neural network over the input. The input of FFNN_{a} is a concatenation of vector representations $\gamma(y_t)$ and $\gamma(y_k)$, and their explicit element-wise similarity $\gamma(y_t) \circ \gamma(y_k)$.

Token representations: To accurately predict COREF link scores as well as decompositional analysis (which is described in the following section), we consider three types of information in each token representation $\gamma(y_t)$: (1) the token itself y_t, (2) on the decoder side, the preceding context $y_{<t}$, and (3) on the encoder side, the input sequence $X = x_1, \ldots, x_N$.

The lexical information of the token itself y_t is represented by its word embedding e_t. The preceding context $y_{<t}$ is encoded by the decoder RNN in Equation (4). We use the decoder hidden state s_t to represent the preceding context information. The encoder-side context is represented by an attention-weighted weight a_t defined in Equation (6). All the above information is concatenated to produce the final token representation $\gamma(y_t)$:

$$\gamma(y_t) = [e_t, s_t, a_t] \quad (12)$$

5.3 Decompositional Analyses

The decompositional analyses D contains scores for Semantic Proto-Role (SPR) properties D_{SPR}, and scores for event factuality D_{FACT}.

SPR: Given a predicate-argument pair (y_i, y_j), we denote the score for SPR property p as $D^{(y_i,y_j)}_{\text{SPR}_p}$. As shown in Fig. 3, we concatenate the token representations of predicate and argument head tokens $\gamma(y_i)$ and $\gamma(y_j)$ as the input to a SPR module. We employ the state-of-the-art SPR module in Rudinger et al. (2018a), defined as:

$$\hat{D}^{(y_i,y_j)}_{\text{SPR}_p} = W_{\text{SPR}_p}\text{ReLU}(W_{\text{shared}}[\gamma(y_i), \gamma(y_j)]) \quad (13)$$

where $\mathbf{W}_{\text{shared}}$ is the weight matrix shared across all properties. $\mathbf{W}_{\text{SPR}_p}$ is the weight matrix for SPR property p. Then, the log-likelihood of the score of SPR property p is defined as the negative L_2 loss, i.e., $-|\hat{D}^{(y_i,y_j)}_{\text{SPR}_p} - D^{(y_i,y_j)}_{\text{SPR}_p}|^2$.

Factuality: We consider predicting event factuality as a scalar regression problem (White et al., 2016), and denote the factuality score of predicate y_k as $D^{(y_k)}_{\text{FACT}}$. As shown in Fig. 3, we take the token representation of predicate head token $\gamma(y_k)$ as the input to the state-of-the-art factuality module (Rudinger et al., 2018c):

$$\hat{D}^{(y_k)}_{\text{FACT}} = V_2\text{ReLU}\left(V_1\gamma(y_k) + b_1\right) + b_2, \quad (14)$$

where V_1 and V_2 are weight matrices, and b_1 and b_2 are biases. The log-likelihood of factuality score is defined as negative of the Huber loss (Huber, 1964) with $\delta = 1$.

We assume conditional independence among decompositional analysis:

$$P(D|X,Y,C) = \prod_{(y_i,y_j)} \prod_{p} (D^{(y_i,y_j)}_{\text{SPR}_p}|X,Y,C) \prod_{y_k} P(D^{(y_k)}_{\text{FACT}}|X,Y,C) \quad (15)$$

5.4 Learning

Given the input sentence X, the output sequence of tokens Y, and the COREF links C, and the decompositional analysis D, the objective is to minimize the below negative log-likelihood:

$$\begin{aligned}\mathcal{L} &= -\log P(Y, C, D|X) \\ &= -\sum_{t=1}^{M}[\mu_1 \log P(y_t) + \mu_2 \log P(c_t|y_t)] - \\ &\quad \mu_3 \log P(D|X,Y,C)\end{aligned}$$

To increase the convergence rate, we pretrain the model by setting the weights $\mu_1 = 1$ and $\mu_2 = \mu_3 = 0$ to only optimize the token generation accuracy. After the model converges, we set $\mu_2 = \mu_3 = 1$ and lower $\mu_1 = 0.1$.

	S metric			BLEU$_{\text{INST}}$	MAE$_{\text{SPR}}$	MAE$_{\text{FACT}}$
	Precision	Recall	F1			
Pipeline	35.08	30.10	32.39	15.03	N/A	N/A
Variant (a)	39.31	32.93	35.84	16.74	0.75	1.11
Variant (b)	42.76	33.20	37.38	17.71	0.74	1.14
Variant (c)	41.74	33.28	37.03	18.01	0.80	1.14
Our model	**45.33**	**33.88**	**38.78**	**19.61**	**0.71**	**1.06**

Table 1: Evaluation of results on the **test** set.
(**In-domain test** results are shown in Appendix D.)

6 Experiments

We now describe the evaluation data, baselines, and experimental results. Hyperparameter settings are reported in Appendix C.

6.1 Data

We choose Chinese as the source language and English as the target language. For **test**, we selected 270 sentences from the Universal Dependencies (UD) English Treebank (Silveira et al., 2014) test set, which have human-annotated SPR (White et al., 2016) and factuality (Rudinger et al., 2018c) analyses. We then created linearized representations for these sentences using PredPatt based on their gold UD syntax. Meanwhile, the Chinese translations of these sentences were created by crowdworkers on Amazon Mechanical Turk. The test dataset will be released upon publication. For **training**, we first collected about 1.8M Chinese-English sentence bitexts from the GALE project (Cohen, 2007), then tokenized Chinese sentences with Stanford Word Segmenter (Chang et al., 2008). We created linearized representations for English sentences using PredPatt based on automatic UD syntax generated by SyntaxNet Parser (Andor et al., 2016), and added SPR and factuality annotations using the state-of-the-art models (Rudinger et al., 2018b,c) trained on SPR v2.x and It-happened v2.0 respectively.[7] We hold out 20K training sentences for **validation** and **in-domain test**. Table 2 reports the dataset statistics.

	No. sents	Source
Train	1,879,172	GALE
Validation	10,000	GALE
In-domain Test	10,000	GALE
Test	270	UD Treebank

Table 2: Statistics of the evaluation data.

6.2 Variants

We evaluate our model described in Section 5 and three variants: **(a)** We replace the coreference annotating mechanism by randomly choosing an antecedent from all preceding instances. **(b)** We preprocess the data by replacing the special symbol "●" with the syntactic head of its antecedent. During training and testing, we replace the coreference annotating mechanism with a heuristic that solves coreference by randomly choosing an antecedent among preceding instances which have the same syntactic head. **(c)** We remove the decoder-side information in the token representation $\gamma(y_t)$ defined in Equation (12) and only keep the encoder-side information a_t. We also include a **Pipeline** approach where Chinese sentences are first translated into English by a neural machine translation system (Klein et al., 2017) and are then annotated by a UD parser (Andor et al., 2016). The UDS linearized representation of **Pipeline** are created by PredPatt based the automatic UD parses.

6.3 Results

Table 1 reports the experimental results on the **test** set. Results on the **in-domain test** set are similar and shown in Appendix D. In Table 1, **S metric** (defined in Section 4) measures the similarity between predicted and reference graph representations. Based on the optimal variable mapping provided by the **S metric**, we are able to evaluate our model and the variants in different aspects: **BLEU$_{\text{INST}}$** measures the BLEU score of all matched instance edges; **MAE$_{\text{SPR}}$** measures the

---

[7] Both datasets are available at `http://decomp.net`

mean absolute error of SPR property scores of all matched argument edges; and $\mathbf{MAE}_{\text{FACT}}$ measures the mean absolute error of factuality scores of all matched attribute edges.

Overall, our proposed model outperforms the variants in every aspect. Variants (a) and (b) use simple heuristics to solve coreference, and achieve reasonable results: they both employ sequence-to-sequence models to predict graph representations, which can be considered a replica of state-of-the-art approaches for structured prediction (Choe and Charniak, 2016; Barzdins and Gosko, 2016; Peng et al., 2017). Compared to our model which employs the coreference annotating mechanism, these two variants suffer notable loss in the precision of S metric. As a result, their performance drops on the other metrics. Variant (c) only uses the encoder-side information for token representation, resulting in significant loss in MAE_{SPR} and MAE_{FACT}. In the pipeline approach, each component is trained independently. During test, residual errors from each component are propagated through the pipeline. As expected, it shows a significant performance drop.

	Precision	Recall	F1
Variant (a)	10.38	31.23	15.58
Variant (b)	88.42	50.59	64.36
Variant (c)	84.12	35.99	50.41
Our model	**96.63**	**97.62**	**97.12**

Table 3: Coreference evaluation (MUC) based on forced decoding.

Coreference occurs 589 times in the test set. To evaluate the coreference accuracy of our model, we force the decoder to generate the reference target sequence, and only predict coreference via the copy mechanism, or its variants. In Table 3, we report the precision, recall, and F_1 for the standard MUC using the official coreference scorer of the CoNLL-2011/2012 shared tasks (Pradhan et al., 2014). Since coreference in our setup occurs at the sentence level, our model achieves high performance. Variant (a) randomly choosing antecedents performs poorly, whereas variant (b), which solves coreference only based on syntactic heads, achieves a relatively high score. Variant (c) demonstrates that only using encoder-side information in the coreference annotating mechanism leads a significant performance drop.

	Our Model	Monolingual SOTA
awareness	0.852	0.879
change location	0.491	0.492
change possession	0.448	0.488
changed	0.307	0.352
change state	0.362	0.373
existed after	0.426	0.478
existed before	0.602	0.618
existed during	0.336	0.358
instigation	0.597	0.59
partitive	0.317	0.359
sentient	0.849	0.88
volition	0.818	0.837
was for benefit	0.566	0.578
was used	0.268	0.203

Table 4: Pearson coefficient of each SPR property.

Since our model and the state-of-the-art monolingual SPR model (Rudinger et al., 2018c) use the same test set, we are able to compare the performance of our model against the monolingual model by forcing the decoder and the coreference mechanism to create the reference graph representation and only predicting the SPR property scores. Table 4 shows the Pearson coefficient of each SPR property. While our model only has the access to the sentence in the source language during the encoding stage,[8] the performance is comparable to the state-of-the-art monolingual model.

7 Conclusions

We introduce the task of cross-lingual decompositional semantic parsing, which maps content provided in a source language into decompositional analysis based on a target language. We present: UDS graph/linearized representations as the target semantic interface, the $\boldsymbol{S}$ metric for evaluation, and the Chinese-English decompositional semantic parsing dataset. We propose an end-to-end learning approach with a coreference annotating mechanism which outperforms three strong baselines. We separately evaluate the coreference mechanism and SPR prediction, showing promising results. The representations for cross-lingual decompositional semantics, the evaluation metric, and the evaluation dataset provided in this work

---

[8] The state-of-the-art monolingual SPR model directly encodes the sentence in the target language.

will be beneficial to the increasing interests in semantic analysis and cross-lingual applications.

Acknowledgments

Thank you to the anonymous reviewers for their feedback. This work was supported in part by the JHU Human Language Technology Center of Excellence (HLTCOE), and DARPA LORELEI and AIDA. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria. Association for Computational Linguistics.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 77–89, Vancouver, Canada. Association for Computational Linguistics.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jason Baldridge and Geert-Jan Kruijff. 2002. Coupling ccg and hybrid logic dependency semantics. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 319–326, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Guntis Barzdins and Didzis Gosko. 2016. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on amr parsing accuracy. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1143–1147, San Diego, California. Association for Computational Linguistics.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *LREC 2012, Eighth International Conference on Language Resources and Evaluation*.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a ccg parser. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Johan Bos, Kilian Evang, Johannes Bjerva, Lasha Abzianidze, Hessel Haagsma, Rik van Noord, Pierre Ludmann, and Duc-Duy Nguyen. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 242–247.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 224–232, Columbus, Ohio. Association for Computational Linguistics.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

Jordan Cohen. 2007. The gale project: A description and an update. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 237–237. IEEE.

Ann Copestake. 2007. Semantic composition with (robust) minimal recursion semantics. In *Proceedings of the Workshop on Deep Linguistic Processing*, pages 73–80. Association for Computational Linguistics.

Ann Copestake. 2009. ***Invited Talk:*** slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 1–9, Athens, Greece. Association for Computational Linguistics.

Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. 1995. Translation using minimal recursion semantics. In *In Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2-3):281–332.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Bonnie Dorr and Nizar Habash. 2002. Interlingua approximation: A generation-heavy approach. In *In Proceedings of AMTA-2002*. University of Chicago Press.

Kilian Evang and Johan Bos. 2016. Cross-lingual learning of an open-domain semantic parser. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 579–588. The COLING 2016 Organizing Committee.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1351–1356, Denver, Colorado. Association for Computational Linguistics.

Pascale Fung and Benfeng Chen. 2004. Biframenet: Bilingual frame semantics resources construction by cross-lingual induction. In *In Proceedings of the 20th International Conference on Computational Linguistics*, pages 931–935.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 369–377, Suntec, Singapore. Association for Computational Linguistics.

Irene Heim. 1988. *The Semantics of Definite and Indefinite Noun Phrases*. Ph. d. dissertation, New York and London.

Jerry R. Hobbs. 2003. Discourse and Inference.

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

Peter J Huber. 1964. Robust estimation of a location parameter. *The annals of mathematical statistics*, pages 73–101.

Chung Hee Hwang and Lenhart K. Schubert. 1994. Interpreting tense, aspect and time adverbials: A compositional, unified approach. In *Temporal Logic*, pages 238–264, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839, Copenhagen, Denmark. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

H Kamp. 1981. A theory of truth and semantic representation, 277-322, jag groenendijk, tmv janssen and mbj stokhof, eds.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Hongyuan Mei, Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2018. Halo: Learning Semantics-Aware Representations for Cross-Lingual Information Extraction. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 142–147. Association for Computational Linguistics.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Teruko Mitamura, Eric H. Nyberg, and Jaime G. Carbonell. 1991. An efficient interlingua translation system for multi-lingual document production. In *Proceedings of Machine Translation Summit III*, pages 2–4.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637, Jeju Island, Korea. Association for Computational Linguistics.

Sebastian Padó and Mirella Lapata. 2005. Cross-linguistic projection of role-semantic information. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 859–866, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection of semantic roles. *J. Artif. Int. Res.*, 36(1):307–340.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural amr parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375, Valencia, Spain. Association for Computational Linguistics.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–35, Baltimore, Maryland. Association for Computational Linguistics.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101, Copenhagen, Denmark. Association for Computational Linguistics.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488.

Rachel Rudinger, Adam Teichert, Ryan Culkin, Sheng Zhang, and Benjamin Van Durme. 2018a. Neural davidsonian semantic proto-role labeling. *arXiv preprint arXiv:1804.07976*.

Rachel Rudinger, Adam Teichert, Ryan Culkin, Sheng Zhang, and Benjamin Van Durme. 2018b. Neural davidsonian semantic proto-role labeling. *arXiv preprint arXiv:1804.07976*.

Rachel Rudinger, Aaron Steven White, and Benjamin Van Durme. 2018c. Neural models of factuality. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Roser Saurí and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. *Language Resources and Evaluation*, 43(3):227.

Lenhart Schubert. 2014. From treebank parses to episodic logic and commonsense inference. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 55–60, Baltimore, MD. Association for Computational Linguistics.

Lenhart Schubert. 2015. Semantic representation. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Lenhart K Schubert. 2000. The situations we talk about. In *Logic-based artificial intelligence*, pages 407–439. Springer.

Lenhart K. Schubert and Chung Hee Hwang. 2000. Natural language processing and knowledge representation. chapter Episodic Logic Meets Little Red Riding Hood: A Comprehensive Natural Representation for Language Understanding, pages 111–174. MIT Press, Cambridge, MA, USA.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Stephanie Strassel and Jennifer Tracey. 2016. Lorelei language packs: Data, tools, and resources for technology development in low resource languages.

In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2004. Cross-lingual information extraction system evaluation. In *Proceedings of the 20th international Conference on Computational Linguistics*, page 882. Association for Computational Linguistics.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*, pages 466–477. Springer.

Mengqiu Wang and Christopher D. Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association of Computational Linguistics*, 2:55–66.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 658–666, Arlington, Virginia, United States. AUAI Press.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2017a. MT/IE: Cross-lingual open information extraction with neural sequence-to-sequence models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 64–70, Valencia, Spain. Association for Computational Linguistics.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2017b. Selective Decoding for Cross-lingual Open Information Extraction. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 832–842, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. 2017c. An Evaluation of PredPatt and Open IE via Stage 1 Semantic Role Labeling. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*.

Learning to Learn Semantic Parsers from Natural Language Supervision

Igor Labutov[*]
LAER AI, Inc.
New York, NY
igor.labutov@laer.ai

Bishan Yang[*]
LAER AI, Inc.
New York, NY
bishan.yang@laer.ai

Tom Mitchell
Carnegie Mellon University
Pittsburgh, PA
tom.mitchell@cmu.edu

Abstract

As humans, we often rely on language to learn language. For example, when corrected in a conversation, we may learn from that correction, over time improving our language fluency. Inspired by this observation, we propose a learning algorithm for training semantic parsers from supervision (feedback) expressed in natural language. Our algorithm learns a semantic parser from users' corrections such as *"no, what I really meant was before his job, not after"*, by also simultaneously learning to parse this natural language feedback in order to leverage it as a form of supervision. Unlike supervision with gold-standard logical forms, our method does not require the user to be familiar with the underlying logical formalism, and unlike supervision from denotation, it does not require the user to know the correct answer to their query. This makes our learning algorithm naturally scalable in settings where existing conversational logs are available and can be leveraged as training data. We construct a novel dataset of natural language feedback in a conversational setting, and show that our method is effective at learning a semantic parser from such natural language supervision.

1 Introduction

Semantic parsing is a problem of mapping a natural language utterance into a formal meaning representation, e.g., an executable logical form (Zelle and Mooney, 1996). Because the space of all logical forms is large but constrained by an underlying structure (i.e., all trees), the problem of learning a semantic parser is commonly formulated as an instance of *structured prediction*.

Historically, approaches based on *supervised learning* of structured prediction models have emerged as some of the first and still remain common in the semantic parsing community (Zettlemoyer and Collins, 2005, 2009; Kwiatkowski et al., 2010). A well recognized practical challenge in supervised learning of structured models is that fully annotated structures (e.g., logical forms) that are needed for training are often highly labor-intensive to collect. This problem is further exacerbated in semantic parsing by the fact that these annotations can only be done by people familiar with the underlying logical language, making it challenging to construct large scale datasets by non-experts.

Over the years, this practical observation has spurred many creative solutions to training semantic parsers that are capable of leveraging weaker forms of supervision, amenable to non-experts. One such weaker form of supervision relies on logical form denotations (i.e, the results of a logical form's execution) – rather than the logical form itself, as "supervisory" signal (Clarke et al., 2010; Liang et al., 2013; Berant et al., 2013; Pasupat and Liang, 2015; Liang et al., 2016; Krishnamurthy et al., 2017). In Question Answeing (QA), for example, this means the annotator needs only to know the answer to a question, rather than the full SQL query needed to obtain that answer. Paraphrasing of utterances already annotated with logical forms is another practical approach to scale up annotation without requiring experts with a knowledge of the underlying logical formalism (Berant and Liang, 2014; Wang et al., 2015).

Although these and similar methods do reduce the difficulty of the annotation task, collecting even these weaker forms of supervision (e.g., denotations and paraphrases) still requires a dedicated annotation event, that occurs outside of the normal interactions between the end-user and the semantic parser[1]. Furthermore, expert knowledge may still be required, even if the underlying logi-

---

[*] Work done while at Carnegie Mellon University.

[1] although in some cases existing datasets can be leveraged to extract this information

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1676–1690
Brussels, Belgium, October 31 - November 4, 2018.

User Utterance	Before [September 10th], how many places was [Brad] employed?
System NLG	find employment [Brad] had, prior to [September 10th]
User Feedback	*Yes, but I also want you to count how many placed Brad was employed at.*

Figure 1: Example (i) user's original utterance, (ii) confirmation query generated by inverting the original parse, (iii) user's generated feedback towards the confirmation query (i.e., original parse) .

cal form does not need to be given by the annotator (QA denotations, for example, require the annotator to know the correct answer to the question – an assumption which doesn't hold for end-users who asked the question with the goal of obtaining the answer). In contrast, our goal is to leverage natural language feedback and corrections that may occur naturally as part of the continuous interaction with the non-expert end-user, as training signal to learn a semantic parser.

The core challenge in leveraging natural language feedback as a form of supervision in training semantic parsers, however, is the challenge of correctly parsing that feedback to extract the supervisory signal embedded in it. Parsing the feedback, just like parsing the original utterance, requires its own semantic parser trained to interpret that feedback. Motivated by this observation, our main contribution in this work is a semi-supervised learning algorithm that learns a task parser (e.g., a question parser) from feedback utterances while simultaneously learning a parser to interpret the feedback utterances. Our algorithm relies only on a small number of annotated logical forms, and can continue learning as more feedback is collected from interactions with the user.

Because our model learns from supervision that it simultaneously learns to interpret, we call our approach *learning to learn semantic parsers from natural language supervision*.

2 Problem Formulation

Formally, the setting proposed in this work can be modelled as follows: (i) the user poses a natural language input u_i (e.g., a question) to the system, (ii) the system parses the user's utterance u_i, producing a logical form $\hat{y}_i$, (iii) the system communicates $\hat{y}_i$ to the user in natural language in the form of a confirmation (i.e., *"did you mean . . . "*), (iv) in response to the system's confirmation, the user generates a feedback utterance f_i, which may be a correction of $\hat{y}_i$ expressed in natural language. The observed variables in a single interaction are the task utterance u_i, the predicted task logical form $\hat{y}_i$ and the user's feedback f_i; the true logical form y_i is hidden. See Figure 1 for an illustration.

A key observation that we make from the above formulation is that learning from such interactions can be effectively done in an *offline* (i.e., non-interactive) setting, using only the logs of past interactions with the user. Our aim is to formulate a model that can learn a task semantic parser (i.e., one parsing the original utterance u) from such interaction logs, without access to the true logical forms (or denotations) of the users' requests.

2.1 Modelling conversational logs

Formally, we propose to learn a semantic parser from conversational logs represented as follows:

$$D = \{(u_i, \hat{y}_i, f_i)\}_{i,\dots,N}$$

where u_i is the user's task utterance (e.g., a question), $\hat{y}_i$ is the system's original parse (logical form) of that utterance, f_i is the user's natural language feedback towards that original parse and N is the number of dialog turns in the log. Note that the original parse $\hat{y}_i$ of utterance u_i could come from any semantic parser that was deployed at the time the data was logged – there are no assumptions made on the source of how $\hat{y}_i$ was produced.

Contrast this with the traditional learning settings for semantic parsers, where the user (or annotator) provides the correct logical form y_i or the execution of the correct logical form $[\![y_i]\!]$ (denotation) (Table 1). In our setting, instead of the correct logical form y_i, we only have access to the logical form produced by whatever semantic parser was interacting with the user at the time the data was collected, i.e., $\hat{y}_i$ is *not* necessarily the correct logical form (though it could be). In this work, however, we focus only on corrective feedback (i.e., cases where $\hat{y} \neq y$) as our main objective is to evaluate the ability to interpret rich natural language feedback (which is only given when the original parse is incorrect). Our key hypothesis in this work is that although we do not observe y_i directly, combining $\hat{y}_i$ with the user's feedback f_i should be sufficient to get a better estimate of the correct logical form (closer to y_i), which we can then leverage as a "training example" to improve the task parser (and the feedback parser).

Supervision	**Dataset**
Full logical forms	$\{(u_i, y_i)\}_{i,\ldots,N}$
Denotations	$\{(u_i, [\![y_i]\!])\}_{i,\ldots,N}$
Binary feedback	$\{(u_i, [\![\hat{y}_i = y_i]\!])\}_{i,\ldots,N}$
NL feedback (this work)	$\{(u_i, \hat{y}_i, f_i)\}_{i,\ldots,N}$

Table 1: Different types of supervision used in literature for training semantic parsers, and the corresponding data needed for each type of supervision. Notation used in the table: u corresponds to user utterance (language), y corresponds to a gold-standard logical form parse of u, $\hat{y}$ corresponds to a predicted logical form, $[\![\cdot]\!]$ is the result of executing an expression inside the brackets and f is the user's feedback expressed in natural language in the context of an utterance and a predicted logical form.

3 Natural Language Supervision

3.1 Learning problem formulation

In this section, we propose a learning algorithm for training a semantic parser from natural language feedback. We will use the terms *task parser* and *feedback parser* to refer to the two distinct parsers used for parsing the user's original task utterance (e.g., question) and parsing the user's follow-up feedback utterance respectively. Our learning algorithm does not assume any specific underlying model for the two parsers aside from the requirement that each parser specifies a probability distribution over logical forms given the utterance (for the task parser) and the utterance plus feedback (for the feedback parser):

Task Parser: $P(y \mid u; \theta_t)$
Feedback Parser: $P(y \mid u, f, \hat{y}; \theta_f)$

where θ_t and θ_f parametrize the task and feedback parsers respectively. Note that the key distinction between the task and feedback parser models is that in addition to the user's original task utterance u, the feedback parser also has access to the user's feedback utterance f, and the original parser's prediction $\hat{y}$. We now introduce a joint model that combines task and feedback parsers in a way that encourages the two models to agree:

$$P(y \mid u, f, \hat{y}; \theta_t, \theta_f) = \frac{1}{Z} \underbrace{P(y \mid u; \theta_t)}_{\text{task parser}} \underbrace{P(y \mid u, f, \hat{y}; \theta_f)}_{\text{feedback parser}}$$

At training time, our objective is to maximize the above joint likelihood by optimizing the parser parameters θ_t and θ_f of the task and feedback parsers respectively. The intuition behind optimizing the joint objective is that it encourages the "weaker" model that does not have access to the feedback utterance to agree with the "stronger" model that does (i.e., using the feedback parser's prediction as a noisy "label" to bootstrap the task parser), while conversely encouraging a more "complex" model to agree with a "simpler" model (i.e., using the simpler task parser model as a "regularizer" for the more complex feedback parser; note that the feedback parser generally has higher model complexity compared to the task parser because it incorporates additional parameters to account for processing the feedback input).[2] Note the feedback parser output is not simply the meaning of the feedback utterance in isolation, but instead the revised semantic interpretation of the original utterance, guided by the feedback utterance. In this sense, the task faced by the feedback parser is to both determine the meaning of the feedback, and to apply that feedback to repair the original interpretation of u_i. Note that this model is closely related to co-training (Blum and Mitchell, 1998) (and more generally multi-view learning (Xu et al., 2013)) and is also a special case of a product-of-experts (PoE) model (Hinton, 1999).

3.2 Learning

The problem of maximizing the joint-likelihood $P(y \mid u, f, \hat{y}; \theta_t, \theta_f)$ can be approached as a standard problem of maximum likelihood estimation in the presence of latent variables (i.e., unobserved logical form y), suggesting the application of the Expectation Maximization (EM) algorithm for learning parameters θ_t and θ_f. The direct application of EM in our setting, however, is faced with a complication in the E-step. Because the hidden variables (logical forms y) are structured, computing the posterior over the space of all logical forms and taking the expectation of the log-likelihood with respect to that posterior is generally intractable (unless we assume certain factorized forms for the logical form likelihood).

Instead, we propose to approximate the E-step by replacing the expectation of the log joint-likelihood with its point estimate, using the maximum a posteriori (MAP) estimate of the posterior over logical forms y to obtain that estimate (this is sometimes referred to as "hard-EM"). Obtaining the MAP estimate of the posterior over log-

---

[2] Note that in practice, to avoid locally optimal degenerate solutions (e.g., where the feedback parser learns to trivially agree with the task parser by learning to ignore the feedback), some amount of labeled logical forms would be required to pre-train both models.

ical forms may itself be a difficult optimization problem, and the specific algorithm for obtaining it would depend on the internal models of the task and the feedback parser.

As an alternative, we propose a simple, MCMC based algorithm for approximating the MAP estimate of the posterior over logical forms, that does not require access to the internals of the parser models, i.e., allowing us to conveniently treat both parsers as "black boxes". The only assumption that we make is that it's easy to sample logical forms from the individual parsers (though not necessarily from the joint model), as well as to compute the likelihoods of those sampled logical forms under at least one of the models.

We use Metropolis Hastings to sample from the posterior over logical forms, using one of the parser models as the proposal distribution. Specifically, if we choose the feedback parser as the proposal distribution, and initialize the Markov Chain with a logical form sampled from the task parser, it can be shown that the acceptance probability r for the first sample conveniently simplifies to the following expression:

$$r = \min\left(1, \frac{P(\hat{y}_f \mid u, \theta_t)}{P(\hat{y}_t \mid u, \theta_t)}\right) \tag{1}$$

where $\hat{y}_t$ and $\hat{y}_f$ are logical forms sampled from the task and feedback parsers respectively. Intuitively the above expression compares the likelihood of the logical form proposed by the task parser to the likelihood of the logical form proposed by the feedback parser, but with *both* likelihoods computed under the same *task* parser model (making the comparison fair). The proposed parse is then accepted with a probability proportional to that ratio. See Algorithm 2 for details.

Finally, given this MAP estimate of y, we can perform optimization over the parser parameters θ_t and θ_f using a single step with stochastic gradient descent before re-estimating the latent logical form y. We also approximate the gradients of each parser model by ignoring the gradient terms associated with the log of the normalizer Z. [3]

3.3 Task Parser Model

Our task parser model is implemented based on existing attention-based encoder-decoder models (Bahdanau et al., 2014; Luong et al., 2015). The encoder takes in an utterance (tokenized) u and computes a context-sensitive embedding h_i for each token u_i using a bidirectional Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997), where h_i is computed as the concatenation of the hidden states at position i output by the forward LSTM and the backward LSTM. The decoder generates the output logical form y one token at a time using another LSTM. At each time step j, it generates y_j based on the current LSTM hidden state s_j, a summary of the input context c_j, and attention scores a_{ji}, which are used for attention-based copying as in (Jia and Liang, 2016). Specifically,

$$p(y_j = w, w \in V_{out} | u, y_{1:j-1}) \propto \exp\left(W_o[s_j; c_j]\right)$$

Algorithm 1: Semantic Parser Training from Natural Language Supervision

```
Input     : $D = \{(u_i, \hat{y}_i, f_i)\}_{1,\ldots,N}$
Output    : Task parser parameters $\theta_t$; Feedback parser
            parameters $\theta_f$
Parameter: Number of training epochs $T$
for $t = 1$ to $T$ do
    for $i = 1$ to $N$ do
        $\hat{y}_i^f \leftarrow$ MH-MAP $(u_i, \hat{y}_i, f_i, \theta_t, \theta_f)$ ;
        $\nabla\theta_t \leftarrow \nabla \log P(\hat{y}_i^f \mid u_i; \theta_t)$ ;
        $\nabla\theta_f \leftarrow \nabla \log P(\hat{y}_i^f \mid u_i, f_i, \hat{y}_i; \theta_f)$ ;
        $\theta_t \leftarrow$ SGD $(\theta_t, \nabla\theta_t)$ ;
        $\theta_f \leftarrow$ SGD $(\theta_f, \nabla\theta_f)$ ;
    end
end
```

Algorithm 2: Metropolis Hastings-based MAP estimation of latent semantic parse

```
Input     : $u, \hat{y}, f, \theta_t, \theta_f$
Output    : latent parse $\hat{y}^f$
Parameter: Number of sampling iterations $N$
Function MH-MAP $(u, \hat{y}, f, \theta_t, \theta_f)$ :
    $samples \leftarrow [\,]$
    // sample parse from task parser
    Sample $\hat{y}_t \sim P(y \mid u, \theta_t)$
    $\hat{y}_{curr} \leftarrow \hat{y}_t$
    for $i = 1$ to $N$ do
        // sample parse from feedback parser
        Sample $\hat{y}_f \sim P(y \mid u, f, \hat{y}, \theta_f)$
        $r \leftarrow \min\left(1, \frac{P(\hat{y}_f | u, \theta_t)}{P(\hat{y}_{curr} | u, \theta_t)}\right)$
        Sample $accept \sim$ Bernoulli$(r)$
        if $accept$ then
            $p_t \leftarrow P(\hat{y}_f \mid u, \theta_t)$
            $p_f \leftarrow P(\hat{y}_f \mid u, f, \hat{y}, \theta_f)$
            $samples[\hat{y}_f] \leftarrow p_t \cdot p_f$
            $\hat{y}_{curr} \leftarrow \hat{y}_f$
        end
    end
    $\hat{y}^f \leftarrow$ argmax $samples$
    return $\hat{y}^f$
```

---

[3] We have experimented with a sampling based approximation of the true gradients with contrastive divergence (Hinton, 2002), however, found that our approximation works sufficiently well empirically. See Algorithm 1 for more details of the complete algorithm.

$$p(y_j = u_i | u, y_{1:j-1}) \propto \exp(a_{ji})$$

where $y_j = w$ denotes that y_j is chosen from the output vocabulary V_{out}; $y_j = u_i$ denotes that y_j is a copy of u_i; $a_{ji} = s_j^T W_a h_i$ is an attention score on the input word u_i; $c_j = \sum_i \alpha_i h_i, \alpha_i \propto \exp(a_{ji})$ is a context vector that summarizes the encoder states; and W_o and W_a are matrix parameters to be learned. After generating y_j, the decoder LSTM updates its hidden state s_{j+1} by taking as input the concatenation of the embedding vector for y_j and the context vector c_j.

An important problem in semantic parsing for conversation is resolving references of people and things mentioned in the dialog context. Instead of treating coreference resolution as a separate problem, we propose a simple way to resolve it as a part of semantic parsing. For each input utterance, we record a list of previously-occurring entity mentions $m = \{m_1, ..., m_L\}$. We consider entity mentions of four types: persons, organizations, times, and topics. The encoder now takes in an utterance u concatenated with m[4], and the decoder can generate a referenced mention through copying.

The top parts of Figure 7 and Figure 8 visualize the attention mechanism of the task parser, where the decoder attends to both the utterance and the conversational context during decoding. Note that the conversational context is only useful when the utterance contains reference mentions.

3.4 Feedback Parser Model

Our feedback parser model is an extension of the encoder-decoder model in the previous section. The encoder consists of two bidirectional LSTMs: one encodes the input utterance u along with the history of entity mentions m and the other encodes the user feedback f. At each time step j during decoding, the decoder computes attention scores over each word u_i in utterance u as well as each feedback word f_k based on the decoder hidden state s_j, the context embedding b_k output by the feedback encoder, and a learnable weight matrix W_e: $e_{jk} = s_j^T W_e b_k$. The input context vector c_j in the previous section is updated by adding a context vector that attends to both the utterance and the feedback:

$$c'_j = \sum_i \alpha_i h_i + \sum_k \beta_k b_k$$

where $\alpha_i \propto \exp(a_{ji})$ and $\beta_k \propto \exp(e_{jk})$. Accordingly, the decoder is allowed to copy words from the feedback:

$$p(y_j = f_k | u, f, y_{1:j-1}) \propto \exp(e_{jk})$$

The bottom parts of Figure 7 and Figure 8 visualize the attention mechanism of the feedback parser, where the decoder attends to the utterance, the conversational context, and the feedback during decoding.

Note that our feedback model does not explicitly incorporate the original logical form $\hat{y}$ that the user was generating their feedback towards. We experimented with a number of ways to incorporate $\hat{y}$ in the feedback parser model, but found it most effective to instead use it during MAP inference for the latent parse y. In sampling a logical form in Algorithm 2, we simply reject the sample if the sampled logical form matches $\hat{y}$.

4 Dataset

In this work we focus on the problem of semantic parsing in a conversational setting and construct a new dataset for this task. Note that a parser is required to resolve references to the conversational context during parsing, which in turn may further amplify ambiguity and propagate errors to the final logical form. Conveniently, the same conversational setting also offers a natural channel for correcting such errors via natural language feedback users can express as part of the conversation.

4.1 Dataset construction

We choose *conversational search* in the domain of email and biographical research as a setting for our dataset. Large enterprises produce large amounts of text during daily operations, e.g., emails, reports, and meeting memos. There is an increasing need for systems that allow users to quickly find information over text through search. Unlike regular tasks like booking airline tickets, search tasks often involve many facets, some of which may or may not be known to the users when the search begins. For example, knowing where a person works may triggers a followup question about whether the person communicates with someone internally about certain topic. Such search tasks will often be handled most naturally through dialog.

Figure 1 shows example dialog turns containing the user's utterance (question) u, system's confirmation of the original parse $\hat{y}$, and the user's feedback f. To simplify and scale data collection, we

---

[4] The mentions are ordered based on their types and each mention is wrapped by special boundary symbols [and].

decouple the dialog turns and collect feedback for each dialog turn in isolation. We do that by showing a worker on Mechanial Turk the original utterance u, the system's natural language confirmation generated by inverting the original logical form $\hat{y}$, and the dialog context summarized in a table. See Figure 2 for a screenshot of the Mechanical Turk interface. The dialog context is summarized by the entities (of types person, organization, time and topic) that were mentioned earlier in the conversation and could be referenced in the original question u shown on the screen[5]. The worker is then asked to type their feedback given $(u, \hat{y})$ and the dialog context displayed on the screen. Turkers are instructed to write a natural language feedback utterance that they might otherwise say in a real conversation when attempting to correct another person's incorrect understanding of their question.

We recognize that our data collection process results in only an approximation of a true multi-turn conversation, however we find that this approach to data collection offers a convenient trade-off for collecting a large number of controlled and diverse context-grounded interactions. Qualitatively we find that turkers are generally able to imagine themselves in the hypothetical ongoing dialog, and are able to generate realistic contextual feedback utterances using only the context summary table provided to them.

The initial dataset of questions paired with original logical form parses $\{u_i, \hat{y}_i\}_{1,\ldots,N}$ that we use to solicit feedback from turkers, is prepared offline. In this separate offline task we collect a dataset of 3556 natural language questions, annotated with gold standard logical forms, in the same domain of email and biographical research. We parse each utterance in this dataset with a floating grammar-based semantic parser trained using a structured perceptron algorithm (implemented in SEMPRE (Berant et al., 2013)) on a subset of the questions. We then construct the dataset for feedback collection by sampling the logical form $\hat{y}$ from the first three candidate parses in the beam produced by the grammar-based parser. We use this grammar-based parser intentionally as a very different model from the one that we would ultimately train (LSTM-based parser) on the conversational logs produced by the original parser.

We retain 1285 out of the 3556 annotated questions to form a test set. The rest 2271 questions we pair with between one and three predicted parses $\hat{y}$ sampled from the beam produced by the grammar-based parser, and present each pair of original utterance and predicted logical form $(u_i, \hat{y}_i)$ to a turker who then generates a feedback utterance f_i. In total, we collect 4321 question/original parse/feedback triples $(u_i, \hat{y}_i, f_i)$ (averaging approximately 1.9 feedback utterances per question).

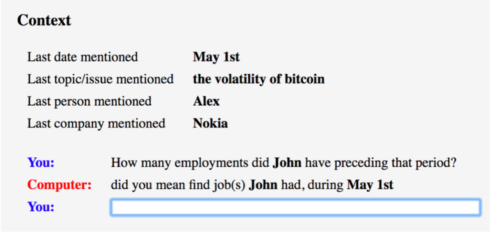

Context

Last date mentioned	**May 1st**
Last topic/issue mentioned	**the volatility of bitcoin**
Last person mentioned	**Alex**
Last company mentioned	**Nokia**

You: How many employments did **John** have preceding that period?
Computer: did you mean find job(s) **John** had, during **May 1st**
You:

Figure 2: Screenshot of the Mechanical Turk web interface used to collect natural language feedback.

5 Experiments

The key hypothesis that we aim to evaluate in our experiments is whether natural language feedback is an effective form of supervision for training a semantic parser. To achieve this goal, we control and measure the effect that the number of feedback utterances used during training has on the resulting performance of the task semantic parser on a held-out test set. Across all our experiments we also use a small seed training set (300 questions) that contains gold-standard logical forms to pre-train the task and feedback parsers. The number of "unlabeled" questions[6] (i.e., questions not labeled with gold standard logical forms but that have natural language feedback) ranges from 300, 500, 1000 to 1700 representing different experimental settings. For each experimental setting, we rerun the experiment 10 times, re-sampling the questions in both the training and unlabeled sets, and report the averaged results. The test-set remains fixed across all experiments and contains 1285 questions labeled with gold-standard logical forms. The implementation details for the task parser and the feedback parser are included in the appendix.

---

[5] zero or more of the context entities may actually be referenced in the original utterance

[6] Note that from hereon we will refer to the portion of the data that contains natural language feedback as the only form of supervision as "unlabeled data", to emphasize that it is not labeled with gold standard logical forms (in contrast to the seed training set).

5.1 Models and evaluation metrics

In our evaluations, we compare the following four models:

- **MH (full model)** Joint model described in Section 3 and in Algorithm 1 (using Metropolis Hastings-based MAP inference described in Algorithm 2).

- **MH (no feedback)** Same as the full model, except we ignore the feedback f and the original logical form $\hat{y}$ information in the feedback model. Effectively, this reduces the model of the feedback parser to that of the task parser. Because during training, both models would be initialized differently, we may still expect the resulting model averaging effects to aid learning.

- **MH (no feedback + reject $\hat{y}$)** Same as the above baseline without feedback, but we incorporate the knowledge of the original logical form $\hat{y}$ during training. We incorporate $\hat{y}$ using the same method as described in Section **??**.

- **Self-training** Latent logical form inference is performed using only the task parser (using beam search). Feedback utterance f and original logical form $\hat{y}$ are ignored. Task parser parameters θ_t are updated in the same way as in Algorithm 1.

Note that all models are exposed to the same training seed set, and differ only in the way they take advantage of the unlabeled data. We perform two types of evaluations of each model:

- **Generalization performance** we use the learned task parser to make predictions on held-out data. This type of evaluation tests the ability of the parser trained with natural language feedback to generalize to unseen utterances.

- **Unlabeled data performance** we use the learned task parser to make predictions on the unlabeled data that was used in training it. Note that for each experimental condition, we perform this evaluation only on the portion of the unlabeled data that was used during training. This ensures that this evaluation tests the model's ability to "recover" correct logical forms from the questions that have natural language feedback associated with them.

6 Results

6.1 Generalization performance

Figure 3a shows test accuracy as a function of the number of unlabeled questions (i.e., questions containing only feedback supervision without gold standard logical forms) used during training, across all four models. As expected, using more unlabeled data generally improves generalization performance. The **self-training** baseline is the only exception, where performance starts to deteriorate as the ratio of unlabeled to labeled questions increases beyond a certain point. This behavior is not necessarily surprising – when the unlabeled examples significantly outnumber the labeled examples, the model may more easily veer away to local optima without being strongly regularized by the loss on the small number of labeled examples.

Interestingly, the **MH (no feedback)** baseline is very similar to **self-training**, but has a significant performance advantage that does not deteriorate with more unlabeled examples. Recall that the **MH (no feedback)** model modifies the full model described in Algorithm 1 by ignoring the feedback f and the original logical form $\hat{y}$ in the feedback parser model $P(y \mid u, \hat{y}, f; \theta_f)$. This has the effect of reducing the model of the feedback parser into the model of the task parser $P(y \mid u; \theta_t)$. The training on unlabeled data proceeds otherwise in the same way as described in Algorithm 1. As a result, the principle behind the **MH (no feedback)** model is the same as that behind **self-training**, i.e., a single model learns from its own predictions. However, different initializations of the two copies of the task parser, and the combined model averaging appears to improve the robustness of the model sufficiently to keep it from diverging as the amount of unlabeled data is increased.

The **MH (no feedback + reject $\hat{y}$)** baseline also does not observe the feedback utterance f, but incorporates the knowledge of the original parse $\hat{y}$. As described in Section **??**, this knowledge is incorporated during MAP inference of the latent parse, by rejecting any logical form samples that match $\hat{y}$. As expected, incorporating the knowledge of $\hat{y}$ improves the performance of this baseline over the one that does not.

Finally, the full model that incorporates both, the feedback utterance f and the original logical form $\hat{y}$ outperforms the baselines that incorporate only some of that information. The performance gain over these baselines grows as more questions with natural language feedback supervision are made available during training. Note that both this and the **MH (no feedback + reject $\hat{y}$)** model incorporate the knowledge of the original logical form $\hat{y}$, however, the performance gain from incorporating the knowledge of $\hat{y}$ without the feedback is relatively small, indicating that the gains from the model that observes feedback is primarily from its ability to interpret it.

6.2 Performance on unlabeled data

Figure 3*b* shows accuracy on the unlabeled data, as a function of the number of unlabeled questions used during training, across all four models. The questions used in evaluating the model's accuracy on unlabeled data are the same unlabeled questions used during training in each experimental condition. The general trend and the relationship between baselines is consistent with the generalization performance on held-out data in Figure 3*a*. One of the main observations is that accuracy on unlabeled training examples remains relatively flat, but consistently high ($> 80\%$), across all models regardless of the amount of unlabeled questions used in training (within the range that we experimented with). This suggests that while the models are able to accurately recover the underlying logical forms of the unlabeled questions regardless of the amount of unlabeled data (within our experimental range), the resulting generalization performance of the learned models is significantly affected by the amount of unlabeled data (more is better).

6.3 Effect of feedback complexity on performance

Figure 4 reports parsing accuracy on unlabeled questions as a function of the number of corrections expressed in the feedback utterance paired with that question. Our main observation is that the performance of the full model (i.e., the joint model that uses natural language feedback) deteriorates for questions that are paired with more complex feedback (i.e., feedback containing more corrections). Perhaps surprisingly, however, is that all models (including those that do not incorporate feedback) deteriorate in performance for questions paired with more complex feedback. This is explained by the fact that more complex feedback is generally provided for more difficult-to-parse questions. In Figure 4, we overlay the parsing performance with the statistics on the average length of the target logical form (number of predicates).

A more important take-away from the results in Figure 4, is that the model that takes advantage of natural language feedback gains an even greater advantage over models that do not use feedback when parsing more difficult questions. This means that the model is able to take advantage of more complex feedback (i.e., with more corrections) even for more difficult to parse questions.

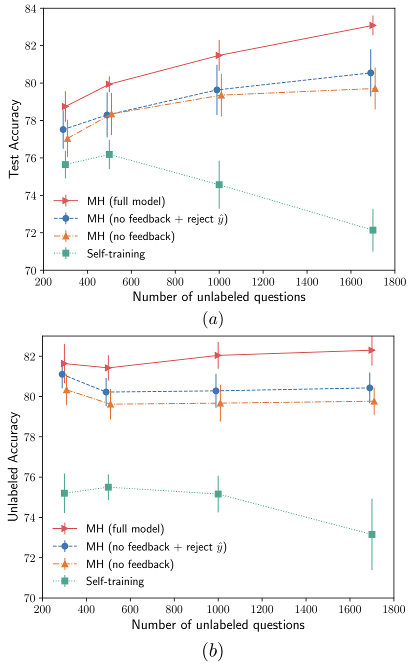

(a)

(b)

Figure 3: *(a)* Parsing accuracy on held-out questions as a function of the number of unlabeled questions used during training. *(b)* Parsing accuracy on unlabeled questions as a function of the number of unlabeled questions used during training. In both panels, all parsers were initialized using 300 labeled examples consisting of questions and their corresponding logical form.

7 Related Work

Early semantic parsing systems map natural language to logical forms using inductive logical programming (Zelle and Mooney, 1996). Modern systems apply statistical models to learn from pairs of sentences and logical forms (Zettlemoyer and Collins, 2005, 2009; Kwiatkowski et al., 2010). As hand-labeled logical forms are very costly to obtain, different forms of weak supervision have been explored. Example works include learning from pairs of sentences and answers by querying a database (Clarke et al., 2010;

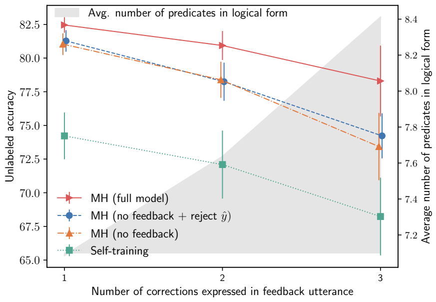

Figure 4: Parsing accuracy on unlabeled questions, partitioned by feedback complexity (i.e., number of corrections expressed in a single feedback utterance).

Liang et al., 2013; Berant et al., 2013; Pasupat and Liang, 2015; Liang et al., 2016; Krishnamurthy et al., 2017); learning from indirect supervision from a large-scale knowledge base (Reddy et al., 2014; Krishnamurthy and Mitchell, 2012); learning from conversations of systems asking for and confirming information (Artzi and Zettlemoyer, 2011; Thomason et al., 2015; Padmakumar et al., 2017); and learning from interactions with a simulated world environment (Branavan et al., 2009; Artzi and Zettlemoyer, 2013; Goldwasser and Roth, 2014; Misra et al., 2015). The supervision used in these methods is mostly in the form of binary feedback, partial logical forms (e.g., slots) or execution results. In this paper, we explore a new form of supervision – natural language feedback. We demonstrate that such feedback not only provides rich and expressive supervisory signals for learning but also can be easily collected via crowd-sourcing. Recent work (Iyer et al., 2017) trains an online language-to-SQL parser from user feedback. Unlike our work, their collected feedback is structured and is used for acquiring more labeled data during training. Our model jointly learns from questions and feedback and can be trained with limited labeled data.

There has been a growing interest on machine learning from natural language instructions. Much work has been done in the setting where an autonomous agent learns to complete a task in an environment, for example, learning to play games by utilizing text manuals (Branavan et al., 2012; Eisenstein et al., 2009; Narasimhan et al., 2015) and guiding policy learning using high-level human advice (Kuhlmann et al., 2004; Squire et al., 2015; Harrison et al., 2017). Recently, natural language explanations have been used to augment labeled examples for concept learning (Srivastava et al., 2017) and to help induce programs that solve algebraic word problems (Ling et al., 2017). Our work is similar in that natural language is used as additional supervision during learning, however, our natural language annotations consist of user feedback on system predictions instead of explanations of the training data.

8 Conclusion and Future Work

In this work, we proposed a novel task of learning a semantic parser directly from end-users' open-ended natural language feedback during a conversation. The key advantage of being able to learn from natural language feedback is that it opens the door to learning continuously through natural interactions with the user, but it also presents a challenge of how to interpret such feedback. In this work we introduced an effective approach that simultaneously learns two parsers: one parser that interprets natural language questions and a second parser that also interprets natural language feedback regarding errors made by the first parser.

Our work is, however, limited to interpreting feedback contained in a single utterance. A natural generalization of learning from natural language feedback is to view it as part of an integrated dialog system capable of both interpreting feedback and asking appropriate questions to solicit this feedback (e.g., connecting to the work by (Padmakumar et al., 2017)). We hope that the problem we introduce in this work, together with the dataset that we release, inspires the community to develop models that can learn language (e.g., semantic parsers) through flexible natural language conversation with end users.

Acknowledgements

This research was supported in part by AFOSR under grant FA95501710218, and in part by the CMU InMind project funded by Oath.

References

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the conference on empirical methods in natural language processing*, pages 421–432. Association for Computational Linguistics.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1:49–62.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1415–1425.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Satchuthananthavale RK Branavan, Harr Chen, Luke S Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 82–90. Association for Computational Linguistics.

SRK Branavan, David Silver, and Regina Barzilay. 2012. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 18–27. Association for Computational Linguistics.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 958–967. Association for Computational Linguistics.

Dan Goldwasser and Dan Roth. 2014. Learning from natural instructions. *Machine learning*, 94(2):205–232.

Brent Harrison, Upol Ehsan, and Mark O Riedl. 2017. Guiding reinforcement learning exploration using natural language. *arXiv preprint arXiv:1707.08616*.

Geoffrey E Hinton. 1999. Products of experts.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

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

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765. Association for Computational Linguistics.

Gregory Kuhlmann, Peter Stone, Raymond Mooney, and Jude Shavlik. 2004. Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer. In *The AAAI-2004 workshop on supervisory control of learning and adaptive systems*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233. Association for Computational Linguistics.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Dipendra Kumar Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Associ-*

ation for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), volume 1, pages 992–1002.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*.

Aishwarya Padmakumar, Jesse Thomason, and Raymond J Mooney. 2017. Integrated learning of dialog strategies and semantic parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 547–557.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association of Computational Linguistics*, 2(1):377–392.

Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. 2015. Grounding english commands to reward functions. *Robotics: Scienceand Systems*.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2017. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1527–1536.

Jesse Thomason, Shiqi Zhang, Raymond J Mooney, and Peter Stone. 2015. Learning to interpret natural language commands through human-robot dialog. In *IJCAI*, pages 1923–1929.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1332–1342.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

Luke S Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 976–984. Association for Computational Linguistics.

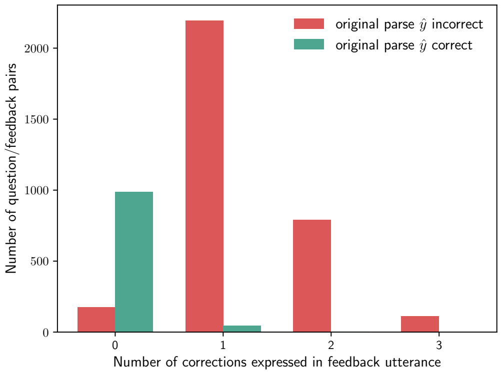

Figure 5: A histogram of the number of corrections expressed in a natural language feedback utterance in our data (0 corrections means that the user affirmed the original parse as correct). We partition the feedback utterances according to whether the original parse $\hat{y}$ that the feedback was provided towards is known to be correct (i.e., matches the gold standard logical form parse).

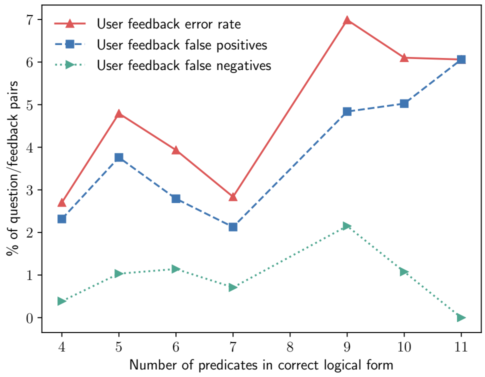

Figure 6: Analysis of noise in feedback utterances contained in our dataset. *Feedback false positives* refers to feedback that incorrectly identifies the wrong original parse $\hat{y}$ as correct. *Feedback false negatives* refers to feedback that incorrectly identifies a correct original parse $\hat{y}$ as wrong. Users are more likely to generate false positives (i.e., miss the error) in their feedback for parses of more complex utterances (as measured by the number of predicates in the gold-standard logical form).

A Analysis of natural language feedback

The dataset we collect creates a unique opportunity to study the nature and the limitations of the corrective feedback that users generate in response to an incorrect parse. One of our hypotheses stated in Section 1 is that natural language affords users to express richer feedback than for example possible with binary (correct/incorrect) mechanism, by allowing users to explicitly refer to and fix what they see as incorrect with the original prediction. In this section we analyze the feedback utterances in our dataset to gain deeper insight into the types of feedback users generate, and the possible limitations of natural language as a source of supervision for semantic parsers.

Figure 5 breaks down the collected feedback by the number of corrections expressed in the feedback utterance. The number of corrections ranges from 0 (no corrections, i.e., worker considers the original parse $\hat{y}$ to be the correct parse of u) to more than 3 corrections.[7] The number of corrections is self-annotated by the workers who write the feedback – we instruct workers to count a correction constrained to a single predicate or a single entity as a single correction, and tally all such corrections in their feedback utterance after they have written it[8]. Because we also know the ground truth of whether the original parse $\hat{y}$ (i.e., parse that the user provided feedback towards) was correct (i.e., $\hat{y}$ matches gold standard parse y), we can partition the number of corrections by whether the original parse $\hat{y}$ was correct (green bars in Figure 5) or incorrect (red bars), allowing us to evaluate the accuracy of some of the feedback.

From Figure 5, we observe that users provide feedback that ranges in the number of corrections, with the majority of feedback utterances making one correction to the original logical form $\hat{y}$ (only 4 feedback utterances expressed more than 3 corrections). Although we do not have gold-standard annotation for the true number of errors in the original logical form $\hat{y}$, we can nevertheless obtain some estimate of the noise in natural language feedback by analyzing cases where we know that the original logical form $\hat{y}$ was correct, yet the user generated feedback with at least one correction. We will refer to such feedback instances as *feedback false negatives*. Similarly, cases where the original logical form $\hat{y}$ is incorrect, yet the user provided no corrections in their feedback, we refer to as *feedback false positives*. The number of feedback false negatives and false positives can be

---

[7] Note that in cases where the user indicates that they made 0 corrections, the feedback utterance that they write is often a variation of a confirmation such as *"that's right"* or *"that's correct"*

[8] we give these instructions to workers in an easy to understand explanation without invoking technical jargon

obtained directly from Figure 5. Generally, we observe that users are more likely to provide more false negatives ($\approx 4\%$ of all feedback) than false positives ($\approx 1\%$) in the feedback they generate.

It is also instructive to consider what factors may contribute to the observed noise in user generated natural language feedback. Our hypothesis is that more complex queries (i.e., longer utterances and longer logical forms) may result in a greater cognitive load to identify and correct the error(s) in the original parse $\hat{y}$. In Figure 6 we investigate this hypothesis by decomposing the percentage of feedback false positives and false negatives as a function of the number of predicates in the gold standard logical form (i.e., one that the user is trying to recover by making corrections in the original logical form $\hat{y}$). Our main observation is that users tend to generate more false positives (i.e., incorrectly identify an originally incorrect logical form $\hat{y}$ as correct) when the target logical form is longer (i.e., the query utterance u is more complex). The number of false negatives (i.e., incorrectly identifying a correct logical form as incorrect) is relatively unaffected by the complexity of the query (i.e., number of predicates in the target logical form). One conclusion that we can draw from this analysis is that we can expect user-generated feedback to miss errors in more complex queries, and models that learn from users' natural language feedback need to have a degree of robustness to such noise.

B Implementation Details

We tokenize user utterances (questions) and feedback using the Stanford CoreNLP package. In all experiments, we use 300-dimensional word embeddings, initialized with word vectors trained using the Paraphrase Database PPDB (Wieting et al., 2015), and we use 128 hidden units for LSTMs. All parameters are initialized uniformly at random. We train all the models using Adam with initial learning rate 10^{-4} and apply L2 gradient norm clipping with a threshold of 10. In all the experiments, we pre-train the task parser and the feedback parser for 20 epochs, and then switch to semi-supervised training for 10 epochs. Pre-training takes roughly 30 minutes and the semi-supervised training process takes up to 7 hours on a Titan X GPU.

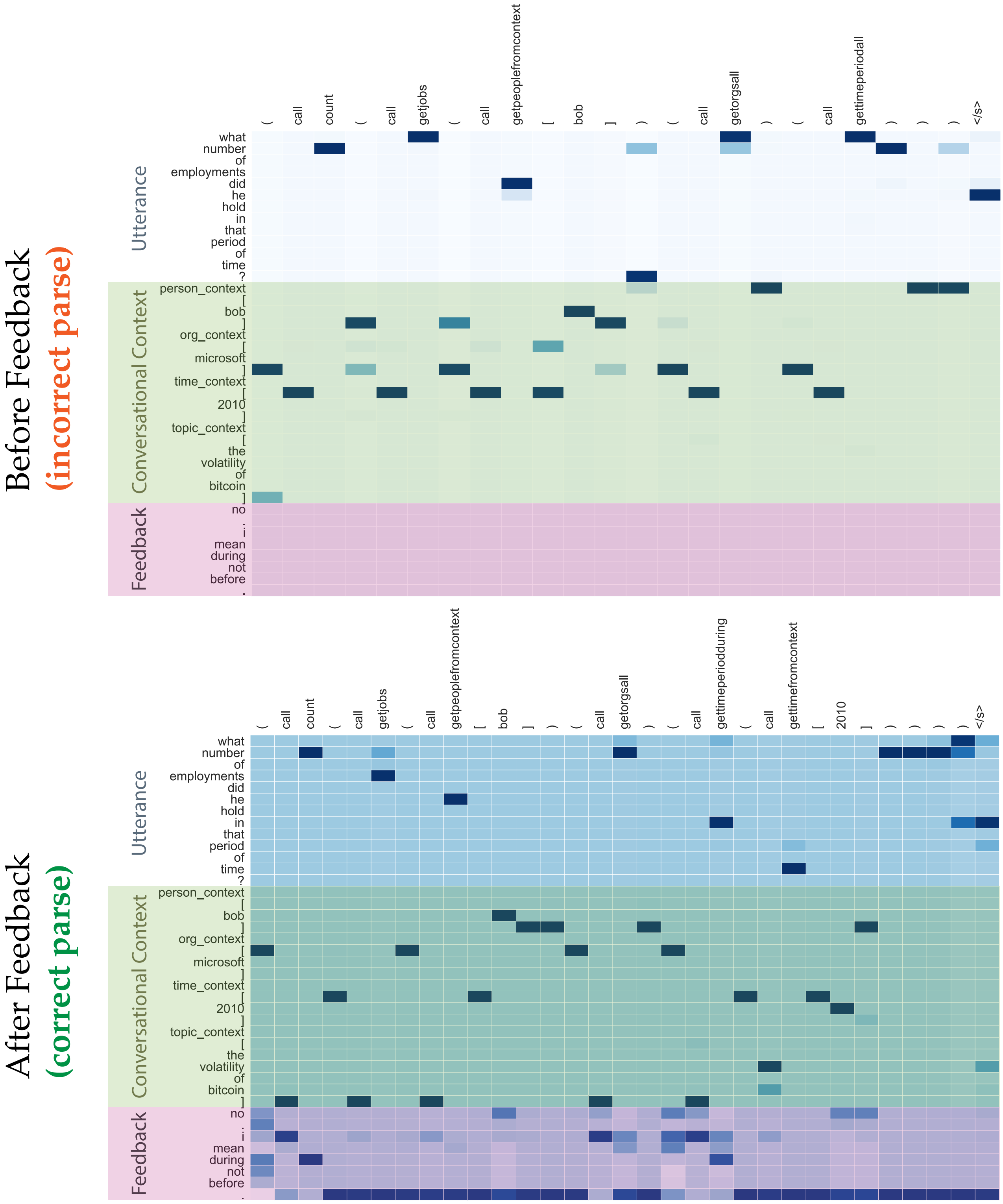

Figure 7: Visualization of the attention mechanism in the task parser (top) and the feedback parser (bottom) for parsing the same utterance. We partition the input to the parser into three groups: the original utterance u being parsed (blue), the conversational context (green) and the feedback utterance f (red). This example was parsed incorrectly before incorporating feedback, but parsed correctly after its incorporation.

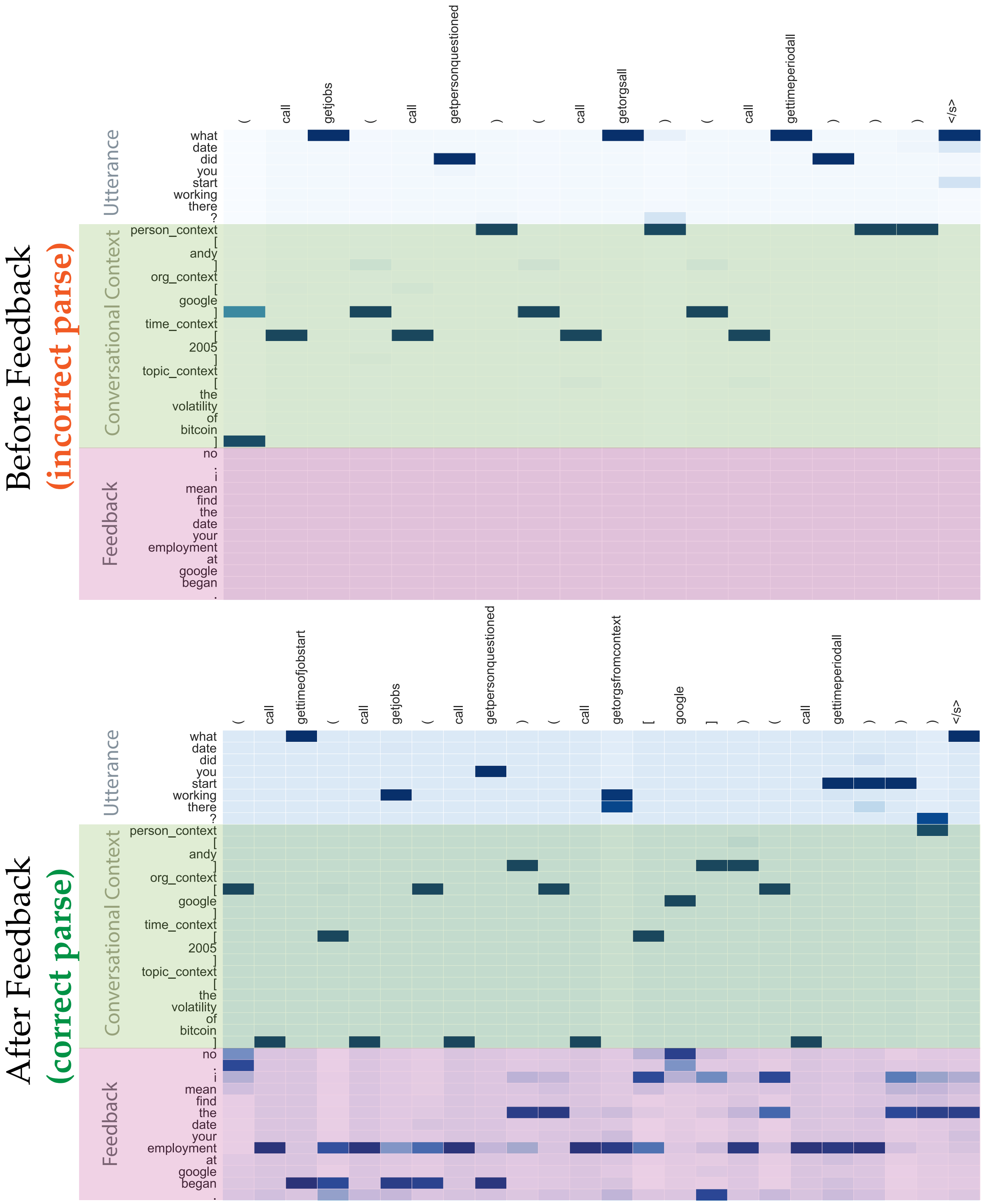

Figure 8: Visualization of the attention mechanism in the task parser (top) and the feedback parser (bottom) for parsing the same utterance. We partition the input to the parser into three groups: the original utterance u being parsed (blue), the conversational context (green) and the feedback utterance f (red). This example was parsed incorrectly before incorporating feedback, but parsed correctly after its incorporation.

DeepCx: A transition-based approach for shallow semantic parsing with complex constructional triggers

Jesse Dunietz
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
jdunietz@cs.cmu.edu

Jaime Carbonell and **Lori Levin**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{jgc,lsl}@cs.cmu.edu

Abstract

This paper introduces the SURFACE CONSTRUCTION LABELING (SCL) task, which expands the coverage of Shallow Semantic Parsing (SSP) to include frames triggered by complex constructions. We present DeepCx, a neural, transition-based system for SCL. As a test case for the approach, we apply DeepCx to the task of tagging causal language in English, which relies on a wider variety of constructions than are typically addressed in SSP. We report substantial improvements over previous tagging efforts on a causal language dataset. We also propose ways DeepCx could be extended to still more difficult constructions and to other semantic domains once appropriate datasets become available.

(1) WE must `regulate` to **inhibit** *unsound practices*.
(2) THIS **opens the way to** *applying the law more widely*.
(3) Judy's comments were **SO** OFFENSIVE that *I left*.

(4) We headed out **in spite of** the bad weather.
(5) We value any contribution, **no matter** its size.
(6) Strange **as** it seems, there's been a run of crazy dreams!

(7) **More** boys wanted to play **than** girls.
(8) Andrew is **as** annoying **as** he **is** useless.
(9) I'm poor**er than** I'd like.

Table 1: Examples of causal (1–3), concessive (4–6), and comparative (7–9) constructions, with triggers bolded. Arguments of causal examples are annotated as in the BECAUSE annotation scheme, with CAUSES in blue small caps, *effects* in red italics, and `means` in purple typewriter text.

1 Introduction

Shallow semantic parsing (SSP) aims to tag the triggers of semantic relations and the phrases between which those relations hold. However, words are not the only bearers of relational meaning: multi-word expressions (MWEs) and even arbitrarily complex constructions can express relations and evoke semantic frames (see, e.g., Fillmore et al., 2012). For example, causation, concession, and comparison are frequently expressed using complex constructions (see Table 1). MWE research has made strides in identifying MWE strings (see, e.g., Baldwin and Kim, 2010), but little work has addressed tagging arguments of such constructional triggers; many of the examples in Table 1 remain a challenge for conventional SSP.

This paper introduces the broader task of SURFACE CONSTRUCTION LABELING (SCL; §3). Like SSP, SCL aims to tag the surface elements of a sentence that express and participate in relational meanings. But in SCL, triggers are not just words or lexical units, but instances of constructions of the sort described by CONSTRUCTION GRAMMAR (CxG; Fillmore et al., 1988; Goldberg, 1995): arbitrarily complex sets of tokens that carry meaning. These constructions can consist of single words, fixed MWEs, gappy MWEs, or even grammatical patterns.

We propose a transition system for SCL that can tag multi-word, possibly gappy sequences of tokens as triggers and arguments (§4). As a test case for the approach, we address English causal language, a valuable target for semantic analysis in its own right. (Extensions to arbitrary frame and role types would be straightforward with appropriate data; see §8.) The transitions can handle most types of constructions, including those with multiple arguments, missing arguments, and even triggers that overlap or are interleaved with arguments. We present DeepCx, a neural network that tags causal language using this transition system (§5). We also describe experiments applying DeepCx to the BECAUSE corpus (§6), showing that DeepCx significantly outperforms prior construction-based work on predicting causal frames (§7). Finally, we discuss how the transition system and tagger model could be adapted to more difficult SCL tasks (§8).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1691–1701
Brussels, Belgium, October 31 - November 4, 2018.

2 Background and related work

2.1 Shallow semantic parsing

SCL of course inherits from SSP, which has a venerable tagging tradition. For PropBank data, dozens of taggers have been developed (see Carreras and Màrquez, 2004, 2005; Surdeanu et al., 2008; Hajič et al., 2009). These typically focus on argument tagging, since PropBank triggers are readily identified by their POS tags. One popular design is a multi-stage pipeline that identifies argument spans and then labels them. Another alternative is BIO-style classification of argument words, either with conventional classifiers or with neural networks (e.g., Collobert et al., 2011; Foland and Martin, 2015). More recent systems (e.g., Täckström et al., 2015; Roth and Lapata, 2016) use neural networks to score and label possible argument spans or heads.

FrameNet-based tagging is more difficult, as triggers must be identified and disambiguated. Many FrameNet taggers have taken a pipeline approach (see, e.g., Baker et al., 2007; Das et al., 2014) in which targets are first identified with a whitelist or simple rules. They are then assigned frames, which determine the available frame elements, and finally the frame elements are identified and labeled. Again, neural networks have also been used to score argument spans and heads (Täckström et al., 2015; FitzGerald et al., 2015; Roth, 2016).

Systems in both paradigms are constrained by their underlying representations. PropBank covers only verbs and certain nominal and adjectival predicates. FrameNet's frame-evoking elements are broader, including verbs, prepositions, adverbs, conjunctions, and even some MWEs, but must still be single words or MWEs that act like words. As Table 1 demonstrates, some semantic domains, such as causality, demand a more flexible approach.

2.2 Construction grammar

CxG, which posits that the fundamental units of language are CONSTRUCTIONS—pairings of meanings with arbitrary linguistic forms. For instance, *so X that Y* (example 3) is characterized by a single construction, where the form is *so* $\langle$*adjective* $X\rangle$ $\langle$*finite clausal complement* $Y\rangle$ and the meaning is X *to an extreme that causes* Y. Following Dunietz et al. (2017a,b), we borrow two core insights of CxG: first, that morphemes, words, MWEs, and grammar are **all on equal footing** as "learned pairings of form and function" (Goldberg, 2013); and second, that constructions pair patterns of surface forms **directly with meanings**. Thus, we can tag any surface realizations of constructions as meaning-bearing triggers (hence "surface construction labeling").

2.3 Causal language

To test the SCL approach, we examine causal language, which conveys essential semantic information and is especially rich in constructional triggers.

Our data representation for causal language comes from the BECAUSE 2.1 corpus (Dunietz et al., 2017b), which focuses on what causal meanings are explicitly stated in the text. It defines causal language as any construction which presents one event, state, action, or entity as promoting or hindering another, and which includes at least one lexical trigger to anchor the annotation. For each instance of causal language, up to three spans are annotated: the **causal connective** (the trigger of the causal relation), the **cause** span, the **effect** span, and occasionally a **means** span (if a means by which the cause produces the effect is specified). See Table 1 for examples of analyses of causal language under the BECAUSE scheme. The corpus includes 4,867 sentences (123,674 tokens) of news articles and Congressional hearing transcripts fully annotated for causal language.

The only prior work on construction-based semantic parsing that we know of is Causeway (Dunietz et al., 2017a), also based on the BECAUSE corpus. Causeway detects causal connectives using lexico-syntactic patterns, then applies heuristics and classifiers to tag arguments and remove false positives. It achieves moderate performance, but requires extensive tuning and feature engineering.

2.4 Transition-based systems

Transition-based systems have primarily been used for dependency parsing (e.g., Nivre et al., 2007; Nivre, 2008; Chen et al., 2014; Choi and Palmer, 2011a). Indeed, our system borrows many implementation elements from Dyer et al. (2015), who describe a shift-reduce parser that embeds the stack and buffer as LSTMs. This parser employs the novel STACK LSTM data structure—an LSTM augmented with a stack pointer, enabling it to be rewound to a previous state.

Transition systems have been developed for semantic tasks, as well. Titov et al. (2009), Henderson et al. (2008), and Swayamdipta et al. (2016) explore extensions of dependency parsing that interleave semantic parsing actions with syntactic parsing actions. Google's SLING (Ringgaard et al., 2017) applies a custom-designed transition scheme for frame-based parsing and coreference resolution. Vilares and Gómez-Rodríguez (2018) develop a transition system for Abstract Meaning Representation parsing, and TUPA (Hershcovich et al., 2017) does the same for Universal Conceptual Cognitive Annotation. Both can handle discontinuous or reentrant graph structures. Most directly relevant to DeepCx is Choi and Palmer's (2011b) work, which defines a novel transition system for PropBank parsing. Our similar scheme for parsing causal constructions builds on this one, extending it for cases where the spans are not contiguous.

3 The SCL task for causal language

An SCL task closely resembles an SSP task, except that the triggers can be complex constructions. As a corollary, the arguments can also be discontinuous and/or overlap with each other or the trigger.

In the case of causal language, we define the task as reproducing the core elements of the BECAUSE scheme: connective, cause, effect, and means spans. Following Dunietz et al. (2017a), we split the task into two parts: discovering causal connectives (**connective discovery**) and delimiting and labeling the arguments (**argument ID**). Producing the additional metadata that BECAUSE records for each instance is left to future work.

Each span is defined as a set of tokens. This excludes sublexical constructions; we return to this limitation in §8.

4 Transition system

Like Choi and Palmer, DeepCx's transition system first searches for a connective word, and once it has found one, compares it with each word to the right and to the left. In each comparison, it selects a transition that labels the word as unrelated to the current connective word, as another connective word (or FRAGMENT), or as a member of some argument span(s). Once all words have been compared to the current connective, the system advances to the next possible initial connective word. In the worst case, then, each sentence takes $O(n^2)$ transitions.

Table 2 gives the full set of transitions. The transitions act on a state tuple $(\lambda_1, \lambda_2, a, \lambda_3, \lambda_4, s, A)$. a is the index of the current possible "connective anchor"—the word being tentatively treated as the initial (i.e., leftmost) word of a connective. λ_1 is the list of word indices to a's left that have not yet been compared with it, and λ_2 represents the words to the left of a that have already been compared. Likewise, λ_3 and λ_4 contain the indices of compared and uncompared words, respectively, to the right. Thus, words move from λ_1 to λ_2 and from λ_4 to λ_3 as they are compared with a. s is a boolean indicating whether we are currently comparing words in the sentence to a, i.e., whether a has been confirmed as a connective anchor. A is a set of partially-constructed causal language instances. Each instance consists of a set of connective word indices plus one set of argument word indices for each argument type. For formal description, we represent A as a set of labeled arcs. The head a of each arc is the connective anchor of a causal language instance i (an arbitrary identifier). The label of the arc indicates what role the tail t plays with respect to i: Cause, Effect, or Means if t is a member of the corresponding argument span, and Frag if t is a connective fragment other than a.

As the algorithm scans from left to right, it assigns a to each word index in turn. If it decides a is not a connective anchor, it issues a NO-CONN and moves on. If a is deemed to start a connective, a new instance is initialized with a NEW-CONN. DeepCx proceeds to compare a with each word to its left, in right-to-left order (i.e., starting from the closest word), then each word to the right (in left-to-right order). For each comparison, it issues a LEFT/RIGHT-ARC, CONN-FRAG, or NO-ARC, depending on whether the comparison word is deemed part of an argument, part of the connective, or neither. For simplicity, we always consider the leftmost connective word to anchor the connective, so all CONN-FRAG transitions occur between a connective word and a word to its right. After all words have been compared with a (i.e., once λ_1 and λ_4 are empty), an automatic SHIFT transition advances a to the next connective anchor candidate.

The initial state is: $\lambda_1 = \lambda_2 = \lambda_3 = []$, $a = 1$, $\lambda_4 = [w_1 \mathinner{..} w_n]$, $s = \mathsf{f}$, and $A = \varnothing$, where w_i is the ith word in the sentence. The algorithm terminates when $a = n$ and either $\lambda_3 = \lambda_4 = []$ or s is false—i.e., when no words remain to a's right, and either a is not a connective anchor or all words in the sentence have been compared with it. An example transition sequence is shown in Table 3.

Some transitions have preconditions, shown inline Table 2 in a smaller font. In addition, several transitions have constraints on their ordering to ensure semantic well-formedness. These constraints

Transition schema	**Effect and preconditions**
NO-CONN	$(\boldsymbol{\lambda_1}, \lambda_2{=}[\,], \boldsymbol{a}, \lambda_3{=}[\,], [\boldsymbol{w} \mid \boldsymbol{\lambda_4}], s{=}\mathsf{f}, A) \Rightarrow ([\boldsymbol{\lambda_1} \mid \boldsymbol{a}], \lambda_2, \boldsymbol{w}, \lambda_3, \boldsymbol{\lambda_4}, s, A)$
NEW-CONN	$(\lambda_1, \lambda_2, a, \lambda_3, \lambda_4, \boldsymbol{s}{=}\mathsf{f}, A) \Rightarrow (\lambda_1, \lambda_2, a, \lambda_3, \lambda_4, \mathbf{t}, A)$
NO-ARC-LEFT	$([\boldsymbol{\lambda_1} \mid \boldsymbol{w}], \boldsymbol{\lambda_2}, a, \lambda_3{=}[\,], \lambda_4, s{=}\mathsf{t}, A) \Rightarrow (\boldsymbol{\lambda_1}, [\boldsymbol{w} \mid \boldsymbol{\lambda_2}], a, \lambda_3, \lambda_4, s, A)$
NO-ARC-RIGHT	$(\lambda_1{=}[\,], \lambda_2, a, \boldsymbol{\lambda_3}, [\boldsymbol{w} \mid \boldsymbol{\lambda_4}], s{=}\mathsf{t}, A) \Rightarrow (\lambda_1, \lambda_2, a, [\boldsymbol{\lambda_3} \mid \boldsymbol{w}], \boldsymbol{\lambda_4}, s, A)$
LEFT-ARC$_x$	$([\boldsymbol{\lambda_1} \mid \boldsymbol{w}], \boldsymbol{\lambda_2}, a, \lambda_3{=}[\,], \lambda_4, s{=}\mathsf{t}, \boldsymbol{A}) \Rightarrow (\boldsymbol{\lambda_1}, [\boldsymbol{w} \mid \boldsymbol{\lambda_2}], a, \lambda_3, \lambda_4, s, \boldsymbol{A} \cup \{\boldsymbol{a} \xrightarrow{\boldsymbol{x_i}} \boldsymbol{w}\})$
RIGHT-ARC$_x$	$(\lambda_1{=}[\,], \lambda_2, a, \boldsymbol{\lambda_3}, [\boldsymbol{w} \mid \boldsymbol{\lambda_4}], s{=}\mathsf{t}, \boldsymbol{A}) \Rightarrow (\lambda_1, \lambda_2, a, [\boldsymbol{\lambda_3} \mid \boldsymbol{w}], \boldsymbol{\lambda_4}, s, \boldsymbol{A} \cup \{\boldsymbol{a} \xrightarrow{\boldsymbol{x_i}} \boldsymbol{w}\})$
CONN-FRAG	$(\lambda_1{=}[\,], \lambda_2, a{\neq}\,w, \lambda_3, [w \mid \lambda_4], s{=}\mathsf{t}, \boldsymbol{A}) \Rightarrow (\lambda_1, \lambda_2, a, \lambda_3, [w \mid \lambda_4], s{=}\mathsf{t}, \boldsymbol{A} \cup \{\boldsymbol{a} \xrightarrow{\mathsf{FRAG}_i} \boldsymbol{w}\})$
SHIFT	$(\boldsymbol{\lambda_1}{=}[\,], \boldsymbol{\lambda_2}, \boldsymbol{a}, [\boldsymbol{w} \mid \boldsymbol{\lambda_3}], \boldsymbol{\lambda_4}{=}[\,], \boldsymbol{s}{=}\mathsf{t}, A) \Rightarrow (\boldsymbol{\lambda_2}, [\,], \boldsymbol{w}, [\,], \boldsymbol{\lambda_3}, \mathbf{f}, A)$
SPLIT	See text (§4.1)

Table 2: The DeepCx transitions. Pre- and post-transition states are expressed as tuples $(\lambda_1, \lambda_2, a, \lambda_3, \lambda_4, s, A)$. x stands for Cause, Effect, Means, or any combination thereof. i indicates the instance under construction; thus, x_i denotes an argument or fragment arc of instance i. Elements changed by the transition are bolded. Preconditions (small font in the starting states) enforce a consistent transition order by delaying rightward actions until all leftward actions are completed.

Transition	$\boldsymbol{\lambda_1}$	$\boldsymbol{\lambda_2}$	$\boldsymbol{a}$	$\boldsymbol{\lambda_3}$	$\boldsymbol{\lambda_4}$	$\boldsymbol{s}$	$\boldsymbol{A}$
–	$[\,]$	$[\,]$	1	$[\,]$	$[1..7]$	f	$\varnothing$
NO-CONN	$\mathbf{[1]}$	$[\,]$	$\mathbf{2}$	$[\,]$	$\mathbf{[2..7]}$	f	$\varnothing$
NO-CONN	$\mathbf{[1,2]}$	$[\,]$	$\mathbf{3}$	$[\,]$	$\mathbf{[3..7]}$	f	$\varnothing$
NO-CONN	$\mathbf{[1..3]}$	$[\,]$	$\mathbf{4}$	$[\,]$	$\mathbf{[4..7]}$	f	$\varnothing$
NEW-CONN	$[1..3]$	$[\,]$	4	$[\,]$	$[4..7]$	**t**	$\{\textbf{because}_4\ (\bullet,\ \bullet)\}$
LEFT-ARC$_{\text{Effect}}$	$\mathbf{[1,2]}$	$\mathbf{[3]}$	4	$[\,]$	$[4..7]$	t	$\{\text{because}_4\ (\textbf{moved}_{\mathbf{3}},\ \bullet)\}$
LEFT-ARC$_{\text{Effect}}$	$[1]$	$\mathbf{[2,3]}$	4	$[\,]$	$[4..7]$	t	$\{\text{because}_4\ (\textbf{they}_{\mathbf{2}}\ \text{moved}_3,\ \bullet)\}$
NO-ARC-LEFT	$\mathbf{[\,]}$	$\mathbf{[1..3]}$	4	$[\,]$	$[4..7]$	t	$\{\text{because}_4\ (\text{they}_2\ \text{moved}_3,\ \bullet)\}$
NO-ARC-RIGHT	$[\,]$	$[1..3]$	4	$\mathbf{[4]}$	$\mathbf{[5..7]}$	t	$\{\text{because}_4\ (\text{they}_2\ \text{moved}_3,\ \bullet)\}$
CONN-FRAG	$[\,]$	$[1..3]$	4	$\mathbf{[4,5]}$	$\mathbf{[6,7]}$	t	$\{\text{because}_4\textbf{/of}_{\mathbf{5}}\ (\text{they}_2\ \text{moved}_3,\ \bullet)\}$
RIGHT-ARC$_{\text{Cause}}$	$[\,]$	$[1..3]$	4	$\mathbf{[4..6]}$	$\mathbf{[7]}$	t	$\{\text{because}_4/\text{of}_5\ (\text{they}_2\ \text{moved}_3,\ \textbf{the}_{\mathbf{6}})\}$
RIGHT-ARC$_{\text{Cause}}$	$[\,]$	$[1..3]$	4	$\mathbf{[4..7]}$	$[\,]$	t	$\{\text{because}_4/\text{of}_5\ (\text{they}_2\ \text{moved}_3,\ \text{the}_6\ \textbf{schools}_{\mathbf{7}})\}$
SHIFT	$\mathbf{[1..4]}$	$[\,]$	$\mathbf{5}$	$[\,]$	$\mathbf{[5..7]}$	**f**	$\{\text{because}_4/\text{of}_5\ (\text{they}_2\ \text{moved}_3,\ \text{the}_6\ \text{schools}_7)\}$
NO-CONN	$\mathbf{[1..5]}$	$[\,]$	$\mathbf{6}$	$[\,]$	$\mathbf{[6,7]}$	f	$\{\text{because}_4/\text{of}_5\ (\text{they}_2\ \text{moved}_3,\ \text{the}_6\ \text{schools}_7)\}$
NO-CONN	$\mathbf{[1..6]}$	$[\,]$	$\mathbf{7}$	$[\,]$	$\mathbf{[7]}$	f	$\{\text{because}_4/\text{of}_5\ (\text{they}_2\ \text{moved}_3,\ \text{the}_6\ \text{schools}_7)\}$
NO-CONN	–	–	–	–	–	–	–

Table 3: The sequence of oracle transitions and states for *Well*$_1$*, they*$_2$ *moved*$_3$ *because*$_4$ *of*$_5$ *the*$_6$ *schools*$_7$. Elements altered by the transition are bolded. Causal language instances are notated as connective(Cause, Effect).

are listed in the supplementary material (§A.3).

4.1 SPLIT transitions

In BECAUSE, a word from one connective can also be part of another connective. This most often occurs with conjoined arguments where portions of the connective are repeated. For example, in *it'll take luck **for** us **to** succeed or even **to** survive, succeed* and *survive* are considered Effects of two different causal instances whose connectives share the *for*. The SPLIT action handles such cases by completing the current causal language instance and starting a new one, copying all connective and argument words up to the repeated connective word.

4.2 Differences from Choi and Palmer

Our scheme differs fourfold from Choi and Palmer:

1. They assume oracle PropBank predicates. DeepCx, lacking oracle connectives, starts new causal language instances with NEW-CONNs, and adds s to the state to track whether such a transition has occurred.
2. Unlike PropBank, BECAUSE allows a connective to include multiple content words. Our system therefore adds a CONN-FRAG transition.
3. A connective word can be part of an argument—e.g., in *enough food to live,* the connective and Cause both include *enough.* DeepCx therefore

compares each connective anchor with itself. (This is why for each new connective anchor a, λ_4 starts out with a as its first element. It is also why the CONN-FRAG action does not advance to the next potential argument word: a connective fragment can be part of an argument.)

4. PropBank never posits two predicates for a single verb, but in BECAUSE, multiple connectives can share a connective word. This case is handled by the new SPLIT transition (see §4.1).

5 DeepCx neural network architecture

Given the experience of previous shallow semantic parsers (e.g., Roth, 2016), we expected performance to depend heavily on syntactic information. We therefore built our system on top of Dyer et al.'s LSTM parser, allowing us to directly incorporate the parser's embeddings. For example, a token's embedding can incorporate the parser's internal embedding of the subtree rooted at that token.

At each step, the network computes a high-dimensional state vector summarizing the internal data structures. That state feeds into a k-dimensional output layer, where k is the number of transition types seen in training. Each vector component is the predicted log probability that the corresponding transition should come next. At test time, the highest-scoring predicted action is taken; in training, gold-standard actions are executed instead.

Figure 1 shows a schematic of the neural network structure. We elaborate on its components below.

5.1 Final state and prediction layers

Beyond λ_{1-4}, the inputs to the state vector are:

- h, the history of actions so far for the sentence.
- d, the path in the dependency parse tree between anchor a and the token being compared with it.
- The lists of tokens making up the connective (o), Cause (c), Effect (e), and Means (m) spans for the causal instance currently under construction.

The parser state $\mathbf{s}$ at each timestep is defined as:

$$\mathbf{s} = \max\left\{\mathbf{0}, \mathbf{W}_s\left[\boldsymbol{\lambda}_1; \boldsymbol{\lambda}_2; \mathbf{a}; \boldsymbol{\lambda}_3; \boldsymbol{\lambda}_4; \mathbf{o}; \mathbf{c}; \mathbf{e}; \mathbf{m}; \mathbf{d}; \mathbf{h}\right] + \mathbf{b}_s\right\},$$

where $\mathbf{b}_s$ is a bias term, $\mathbf{W}$ is a learned parameter matrix, and any other bold variable $\mathbf{x}$ indicates an embedding of a variable x (described in §5.2). max indicates a component-wise ReLU.

The predicted probability of each transition T is computed from $\mathbf{s}$ using a softmax unit:

$$p\left(T \mid \mathbf{s}\right) = \exp\left(\mathbf{g}_T^\top \mathbf{s} + q_T\right)/z,$$

where $\mathbf{g}_T$ is a learned embedding of T, q_T is a bias for T, and z is a normalizing constant.

5.2 Embedding the inputs to the state

5.2.1 Embedding a token

Following Dyer et al. (2015), each token t is represented as a concatenation of three vector inputs:

- $\tilde{\mathbf{w}}_t$, a fixed word embedding for t's surface form.
- $\mathbf{w}_t$, a small additional word embedding of t, which allows the network to learn task-specific representations of words related to causality. This is the only component of a token's representation that is trained specifically for this task (i.e., that does not use an embedding from a pretrained language model or a syntactic parsing model).
- $\mathbf{p}_t$, the LSTM parser's internal embedding of the POS tag it assigned to t in preprocessing.

The concatenation is passed through a linear transformation $\mathbf{V}$ (with a bias $\mathbf{b}_t$) and a ReLU:

$$\mathbf{t} = \max\left\{\mathbf{0}, \mathbf{V}\left[\tilde{\mathbf{w}}_t; \mathbf{w}_t; \mathbf{p}_t\right] + \mathbf{b}_t\right\}$$

5.2.2 Embedding a list of tokens

For each input to the final state vector that is a list of tokens, we add an LSTM cell to the network.

For the spans of the instance under construction—i.e., the connective, Cause, Effect, and Means spans—embedding token lists is straightforward: whenever a transition adds a token to one of these lists, that token's embedding is added to the corresponding LSTM's input sequence. The LSTM's updated output is then used for all subsequent actions until another transition modifies the span.

The procedure for embedding the λ's is more involved. As transitions are taken, tokens may need to be moved between lists—e.g., the argument token is moved from λ_1 to λ_2 after a LEFT-ARC transition, and the connective anchor token is moved from λ_4 to λ_1 on a NO-CONN.

We implement these transfers using stack LSTMs. Initially, all tokens' embeddings are input to λ_4, but in **reverse order**, so that the leftmost token is added last. Then, whenever λ_4's leftmost token t is to be moved—i.e., on a SHIFT, NO-CONN,

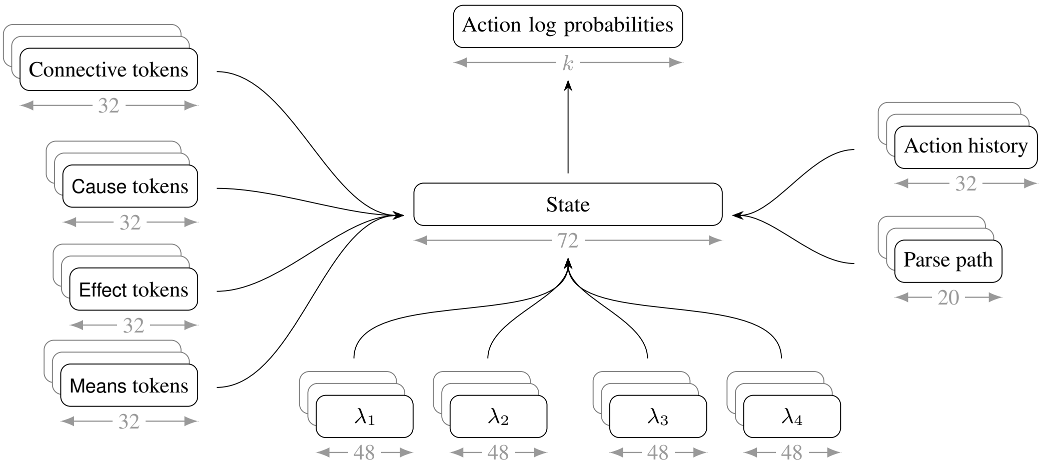

Figure 1: Schematic of the overall neural network architecture. Each lone box represents a vector. Stacked boxes represent LSTMs: at any given time, the state is a single vector, but that state encodes a series of inputs.

RIGHT-ARC, or NO-ARC-RIGHT—the λ_4 LSTM is rewound one step to its state before t was added. Storing λ_4 in reverse order offers the added benefit of tokens closer to the anchor holding greater sway, since LSTMs favor recently added inputs.

λ_1 and λ_2 are a mirror image of λ_4 and λ_3, respectively. Tokens are added to λ_1 on either a SHIFT or a NO-CONN. Thus, the λ_1 LSTM ends up representing an in-order list of tokens up to the current a. If a is then flagged as a connective anchor, tokens to its left are moved from λ_1 to λ_2 as they are compared. The rightmost token t in λ_1 is the first to be compared, so the λ_1 LSTM is rewound to remove t. t's embedding is then added to λ_2, leaving λ_2 with a reversed list of compared tokens.

5.2.3 Embedding a dependency path

The syntactic relationship between the connective anchor a and a candidate argument word t is given to the network as a DEPENDENCY PATH—the series of labels on the dependency arcs between a and t. For instance, consider the dependency parse below:

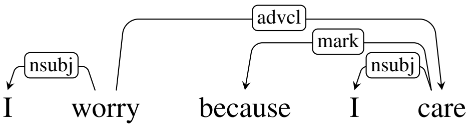

Here, the path from the first *I* to *because* would be $\bigcirc \xleftarrow{\text{nsubj}} \bigcirc \xrightarrow{\text{advcl}} \bigcirc \xrightarrow{\text{mark}} \bigcirc$, where the blank nodes take the place of the words *I, worry, care,* and *because* in the dependency graph.

To embed a dependency path, we again use the output of an LSTM cell, where each input is an embedding of a dependency label: for a label x, we directly use the LSTM parser's embedding for the **syntactic** parse action LEFT-ARC(x), if available, or RIGHT-ARC(x) otherwise. We add one extra bit to each arc's embedding to indicate whether it was traversed forward or backward in this path.[1]

5.2.4 Embedding the action history

During training, DeepCx learns vector representations of each action. To embed the action history, these action embeddings are fed as inputs into yet another LSTM cell. This LSTM's output is the embedding of the history thus far.

5.3 Implementation details

DeepCx is implemented using a refactored version of the LSTM parser codebase that performs identically to the original.[2] The neural network framework, which also underlies the LSTM parser, is an early version of DyNet (Neubig et al., 2017). The LSTM parser model is pretrained on the usual Penn Treebank (Marcus et al., 1994) sections (training: 02–21; development: 22).

For $\tilde{\mathbf{w}}$, we use the same "structured skip n-gram" word embeddings as the LSTM parser. See Dyer et al. (2015) for details about the embedding approach, hyperparameters, and training corpora. DeepCx gives no special treatment to out-of-vocabulary items, other than using the $\mathbf{0}$ vector for words not included in the pretrained embeddings.

---

[1] This embedding is similar to that proposed by Roth and Lapata (2016). However, their dependency paths include the words encountered along the way and their POS tags. We experimented with adding these elements to our dependency paths, but found that they consistently reduced performance.

[2] `https://github.com/clab/lstm-parser/tree/easy-to-use`.

The code for DeepCx is available on GitHub.[3]

5.3.1 Dimensionalities

The pretrained LSTM parser model uses the same dimensionalities as the original LSTM parser.

Token embeddings are 48-dimensional; $\mathbf{w}$ is 10-dimensional. The remaining DeepCx neural network dimensionalities used in the experiments reported below are shown in Figure 1. All LSTM cells use two layers of LSTMs before the final output. These values were chosen as an intuitive balance between values that worked well for other projects and what we could reasonably expect to train with the amount of data we have. Early experiments showed little sensitivity to dimensionality.

6 Experiments

6.1 Experimental setup and training setup

Due to the small corpus size, all experiments use 20-fold cross-validation, split by sentence. Within each fold, the available data—i.e., everything but the fold's held-out test set—is randomized, then split into 80% training and 20% development. After each sentence has been fed through the network, taking gold-standard transitions (see §5), backpropagation is run on all predictions for the sentence. Development set performance is evaluated every 2500 sentences. After each epoch, the training and development sets are re-randomized and re-split.[4] Training ends when either the connective-level F_1 score[5] on the development data hits 0.999 or 85% of the past five epochs' evaluations have yielded lower scores than their immediate predecessors. All systems used the same folds. See the supplementary materials (§A.4) for training parameters.

6.2 Network variants tested in experiments

Ablation studies In addition to the vanilla configuration described above, we examined which non-essential model components contribute to performance. We were particularly interested in the effects of parse information. We tested eliminating the following components of the DeepCx model: (1) $\mathbf{w}$, the task-specific word embeddings, which could contribute to overfitting; (2) $\mathbf{a}$, the action history; and (3) $\mathbf{d}$, the parse path between the connective anchor and the current comparison token.

Argument identification alone DeepCx has no separate argument tagging phase, so we tested performance on the subtask of argument identification by providing DeepCx with oracle transitions only for actions that act on the connective—i.e., NO-CONN, NEW-CONN, CONN-FRAG, and SPLIT. The system was then responsible for deciding between NO-ARC, LEFT-ARC, and RIGHT-ARC transitions.

Restricting generalization One of the strengths of the transition-based approach is its ability to recognize previously unseen forms of causal language that resemble known connectives semantically and/or linguistically. Given our relatively small dataset, however, it seemed possible that the system would not have enough data to make meaningful generalizations. We therefore tested a variant where DeepCx would refuse to allow a test-time NEW-CONN or CONN-FRAG transition unless adding the putative connective word would match the initial word sequence of some connective seen in training.

6.3 Evaluation metrics

For connective discovery we measure precision, recall, and F_1, requiring connectives to match exactly. For argument ID, we split metrics for Causes and Effects (we omit Means, as there are too few in the corpus to evaluate reliably). For each argument type, we report F_1 of connective/argument pairs, where matches must match exactly; F_1 of connective/argument pairs, where half of the larger span's tokens must match; and the average Jaccard index for gold vs. predicted spans, **given a correct connective**. Punctuation is excluded from evaluation.

Jaccard indices convey how close argument tagging is when it does not match exactly. This metric is computed only over true positive connectives, as argument overlap cannot be evaluated automatically for false positives. Thus, Jaccard indices are **not directly comparable** between systems—they represent how well argument ID works given the previous stage, rather than in an absolute sense.

7 Results and analysis

Results are shown in Table 4. For comparison, we also report on the best Causeway configurations.

All significance tests below are paired, two-tailed t-tests on the results from all 20 folds.

---

[3] `https://github.com/duncanka/lstm-causality-tagger`.

[4] Reusing the development data means the network can end up memorizing. However, early experiments with dedicated development data showed lower scores, presumably because too much training data was lost from each fold. Of course, our final evaluation is still performed on the fold's held-out data.

[5] For the experiment with oracle connectives, action-level prediction accuracy is used instead of F_1 score.

	Connectives			Causes			Effects		
System variant	**P**	**R**	$\mathbf{F_1}$	$\mathbf{F_1}$	$\mathbf{F_1@.5}$	**J**	$\mathbf{F_1}$	$\mathbf{F_1@.5}$	**J**
Best Causeway-S	62.8	46.2	*53.1*	37.9	42.5	81.0	24.8	38.7	73.3
Best Causeway-L	63.4	45.1	52.5	*38.8*	*43.5*	83.7	*30.4*	*40.7*	78.4
Vanilla DeepCx	*63.4*	55.8	*59.2*	*43.9*	*50.6*	83.0	41.0	*51.7*	82.2
No $\mathbf{w}$	62.7	56.2	59.1	42.8	49.0	81.9	*41.8*	*51.7*	82.5
No $\mathbf{d}$	62.7	***56.5***	*59.2*	42.7	49.1	80.9	39.6	51.1	80.5
No $\mathbf{a}$	61.3	54.1	57.3	40.3	48.3	81.6	36.9	50.4	81.1
No novel connectives	65.4	**56.5**	**60.5**	**44.7**	***51.0***	82.6	**42.8**	***52.6***	82.1
Oracle connectives	–	–	–	73.5	80.9	79.7	67.8	82.8	80.7

Table 4: Results for all variants of DeepCx tested. As before, J indicates Jaccard index. For $P/R/F_1$ scores, the best non-oracle results are bolded, and the best results within each of the top two sections are italicized.

7.1 Overall performance

The results show the DeepCx transition system to be a promising approach for SCL.

The vanilla configuration unmistakably eclipses Causeway at connective discovery with a margin of 6.1 F_1 points, driven primarily by recall. Both F_1 scores have high standard deviations across folds (3.6–4.7 points), but the scores covary; some folds are simply harder. DeepCx usually leads Causeway by at least 5 points, making the difference highly statistically significant ($p \ll 0.001$). The gap comes primarily from recall, where DeepCx averages 9.6–10.7 points higher than Causeway. [6]

On end-to-end argument identification, DeepCx again outperforms Causeway, particularly on recall, with a 5–6-point gap in F_1. The Jaccard indices for Causes and Effects are in the low 80's, indicating extensive overlap with gold-standard spans. They are on par with Causeway for Causes and higher for Effects, despite the fact that DeepCx's higher recall gives it more chances to be docked for mismatches.

7.2 Argument identification alone

Argument ID scores remain high when oracle connectives are provided. Naturally, the end-to-end argument scores improve dramatically compared to non-oracle connectives, but the more important question is what fraction of the previous errors remain when connective discovery is no longer a source of error. With oracle connectives, DeepCx achieves 73.5% F_1 on Causes and 67.8% on Effects, implying that the vanilla configuration's argument error was split roughly half and half between connective discovery failures and argument ID failures.

However, the F_1 metrics reflect exact span matches; it is counted as a mismatch if even a single word is off. Because in this experiment the system's entire task is to tag arguments, the Jaccard indices give an absolute measure of overlap between predicted and gold argument spans. By that measure, the neural network's treatment of argument identification transitions looks quite robust. Jaccard indices do drop by a few points compared to non-oracle connectives, as expected: with the oracle, arguments are evaluated for every gold-standard instance, including more difficult ones that the vanilla configuration misses. But despite the more exhaustive assessment, DeepCx maintains Jaccard indices of $\sim$80% for Causes and Effects.

7.3 Model ablation studies

No pieces of the model beyond the bare essentials improved connective scores. Removing these components did marginally lower argument ID scores, but few differences were statistically significant.

The meager effects of parse paths came as a surprise; indeed, our reason for building on the LSTM parser was to lean on its parse embeddings. That these paths made little difference suggests that the bulk of the information they provide is available in some isomorphic form from simpler inputs.

7.4 Constraining to known connectives

Constraining DeepCx to known connectives yields an interesting tradeoff. On the one hand, it boosts precision ($p < 0.036$) and raises F_1 slightly ($p < 0.09$). Inspecting the vanilla system's outputs accentuates the risks of letting it run wild inventing connectives: its odder proposals included *an unfair effort to, is insanity, eight,* and the dollar sign.

On the other hand, some generalizations were surprisingly perceptive. For instance, the phrase *allowing states greater opportunity to regulate* was not marked by annotators because *allowing* here seems to mean "providing." But DeepCx proposed

---

[6] Recall should perhaps have even been higher: in at least three cases, DeepCx was penalized for correctly flagging connectives that had been missed by annotators.

allowing opportunity to as a connective—a plausible candidate for annotation. Elsewhere DeepCx tagged *catalyst for* and *fuel* (as in *fueled skepticism*), both arguably annotator omissions.

Ultimately, then, whether to permit novel connectives depends on the user's prioritization of precision, recall, and discovery.

8 What's needed for other constructions and domains?

Although the DeepCx transitions were designed for BECAUSE, it would be straightforward, given appropriate corpora, to extend the transition scheme and model structure to arbitrary frames and role labels as in PropBank and FrameNet. The scheme's arc transitions would need variants for each possible role type, as is standard in existing transition-based SSP (e.g., SLING, Choi and Palmer). Likewise, NEW-CONN could be changed to NEW-CONN(frame); the space of arc transitions for constructing the rest of that instance could then be pruned to those relevant to the frame. As for the tagger state, there are several straightforward ways to modify it for open-ended role and frame labels. One option is to represent each instance's arguments as a list of ⟨*role label*, *list of tokens*⟩ tuples, and to add a frame label variable that is embedded as part of the state. Alternatively, we could follow SLING in providing the tagger a list of ⟨*frame label*, *role label*, *token*⟩ tuples.

Applying SCL to domains beyond causality would be particularly useful for relations like comparison and concession (see Table 1), where complex constructions abound. But as Fillmore et al. (2012) observe, many frames possess the odd non-lexical-unit trigger. For example, the Motion frame can be evoked by the "verb-way" construction (*sang our way across Europe*), and Measurement by the abstract pattern ⟨*number*⟩ ⟨*unit*⟩ ⟨*noun*⟩ (as in *twelve-inch-thick*). Expanding SSP to cover constructions would allow parsing these cases, which are individually rare but collectively form a fat tail of frame instances.

DeepCx already covers most constructional quirks that interfere with SSP, including discontinuous trigger and argument spans, overlaps between arguments, overlaps between trigger words and arguments, and overlaps between triggers. Still, several extensions might be needed for the full gamut of arbitrary constructions. Most notably, our scheme operates on words, but plenty of constructions are sub-lexical (e.g., the comparative *-er*). One solution would be to operate on morphemes instead. Unfortunately, tagging would then be subject to errors in morphological analysis, and morpheme- or character-based embeddings would be needed. A simpler but less elegant solution would be to tag the entire word containing the morpheme (e.g., *bigger*) as part of the construction.

A second challenge is constructions with no lexical trigger, as in *I can't come; I have rehearsal*. The simplest fix would be to add a JUXT transition as a sibling of NEW-CONN. This transition would anchor a new relation instance at the boundary between the words currently being compared, indicating that the mere juxtaposition of two argument spans conveys a relation between them.

Cross-sentential constructions—e.g., discourse connectives whose arguments can be in another sentence—pose a third challenge: our sentence-oriented scheme ignores sentential juxtaposition and cross-sentential grammatical relations as construction possibilities. While it would not be too difficult to alter the scheme to allow, say, arguments in the previous k sentences, it might make randomized training more difficult.

Finally, SPLITs make strong assumptions about how two connectives sharing words will interact. Constructions violating these assumptions may require more drastic surgery on the scheme.

9 Contributions and takeaways

This paper has introduced surface construction labeling as an expansion of shallow semantic parsing. It has also presented DeepCx, a neural transition framework unifying connective discovery and argument ID for causal constructions. DeepCx achieves strong performance on parsing such constructions. Although the transition system targets causal language, its flexibility makes it promising for other domains, as well. We hope DeepCx will inspire further work on SCL. This includes applying more sophisticated tagging techniques such as bidirectional LSTMs, attention, and dynamic oracles, but most importantly developing new data and tasks to which the approach can be applied.

Acknowledgments

The authors thank Miguel Ballesteros, Chris Dyer, Eduard Hovy, Nathan Schneider, and Todd Snider for their invaluable help in developing these ideas, offering feedback, and critiquing draft writeups.

References

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval'07 task 19: frame semantic structure extraction. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007)*, pages 99–104. Association for Computational Linguistics.

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*, volume 2, pages 267–292. CRC Press, Boca Raton, USA.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL 2004)*, pages 89–97. Association for Computational Linguistics.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL 2005)*, pages 152–164. Association for Computational Linguistics.

Wenliang Chen, Yue Zhang, and Min Zhang. 2014. Feature embedding for dependency parsing. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers (COLING 2014)*, pages 816–826. Association for Computational Linguistics.

Jinho D. Choi and Martha Palmer. 2011a. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT '11)*, volume 2, pages 687–692. Association for Computational Linguistics.

Jinho D Choi and Martha Palmer. 2011b. Transition-based semantic role labeling using predicate argument clustering. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 37–45. Association for Computational Linguistics.

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

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2017a. Automatically tagging constructions of causation and their slot-fillers. In *Transactions of the Association for Computational Linguistics*, volume 5, pages 117–133. Association for Computational Linguistics.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2017b. The BECauSE corpus 2.0: Annotating causality and overlapping relations. In *Proceedings of the 11th Linguistic Annotation Workshop (LAW XI)*, pages 95–104. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, pages 334–343. Association for Computational Linguistics.

Charles J. Fillmore, Paul Kay, and Mary Catherine O'Connor. 1988. Regularity and idiomaticity in grammatical constructions: The case of *let alone*. *Language*, 64(3):501–538.

Charles J. Fillmore, Russell Lee-Goldman, and Russell Rhodes. 2012. The FrameNet constructicon. *Sign-Based Construction Grammar*, pages 309–372.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 17–21. Association for Computational Linguistics.

William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 279–288. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. Proceedings of Machine Learning Research.

Adele Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. Chicago University Press.

Adele E. Goldberg. 2013. Constructionist approaches. In *The Oxford Handbook of Construction Grammar*, Oxford Handbooks in Linguistics, pages 15–31. Oxford University Press USA.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth SIGNLL Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18. Association for Computational Linguistics.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 178–182. Association for Computational Linguistics.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for ucca. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1127–1138. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology (HLT '94)*, pages 114–119. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. ArXiv:1701.03980.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Michael Ringgaard, Rahul Gupta, and Fernando C. N. Pereira. 2017. SLING: a framework for frame semantic parsing. ArXiv:1710.07032.

Michael Roth. 2016. Improving frame semantic parsing via dependency path embeddings. In *Book of Abstracts of the 9th International Conference on Construction Grammar*, pages 165–167.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1192–1202. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 159–177. Association for Computational Linguistics.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016)*, pages 187–197. Association for Computational Linguistics.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence (IJCAI '09)*, pages 1562–1567, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

David Vilares and Carlos Gómez-Rodríguez. 2018. A transition-based algorithm for unrestricted amr parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 142–149. Association for Computational Linguistics.

What It Takes to Achieve 100% Condition Accuracy on WikiSQL

Semih Yavuz and **Izzeddin Gur** and **Yu Su** and **Xifeng Yan**
Department of Computer Science, University of California, Santa Barbara
`{syavuz,izzeddingur,ysu,xyan}@cs.ucsb.com`

Abstract

WikiSQL is a newly released dataset for studying the natural language sequence to SQL translation problem. The SQL queries in WikiSQL are simple: Each involves one relation and does not have any join operation. Despite of its simplicity, none of the publicly reported structured query generation models can achieve an accuracy beyond 62%, which is still far from enough for practical use. In this paper, we ask two questions, "Why is the accuracy still low for such simple queries?" and "What does it take to achieve 100% accuracy on WikiSQL?" To limit the scope of our study, we focus on the `WHERE` clause in SQL. The answers will help us gain insights about the directions we should explore in order to further improve the translation accuracy. We will then investigate alternative solutions to realize the potential ceiling performance on WikiSQL. Our proposed solution can reach up to 88.6% condition accuracy on the WikiSQL dataset.

1 Introduction

A large amount of world's data is stored in relational databases, which require users to master structured query languages such as SQL to query data. It might not be convenient for many users who do not have programming background. Towards removing this huge barrier between non-expert users and data, building reliable natural language interfaces to databases has been a long-standing open problem (Woods, 1973; Androutsopoulos et al., 1995; Popescu et al., 2003).

Recently, there is a renewed interest in natural language interfaces to databases due to the advance in deep learning and the new release of large-scale annotated data such as WikiSQL (Zhong et al., 2017). WikiSQL includes a large collection of questions and their corresponding SQL queries. While the queries in WikiSQL are quite simple: Each involves one relation and does not have join operations, none of the publicly reported SQL generation models (Zhong et al., 2017; Xu et al.) can achieve an accuracy beyond 62%, which is far from enough for practical use. It is not clear yet what level of parsing capabilities are needed to achieve high performance, ideally close to 100% accuracy, on this task.

In this paper, we aim to figure out the level of language understanding required to perform well on the WikiSQL task. Specifically, we focus on the `WHERE` clause generation, which is the most challenging part of this task as reported in (Xu et al.): The accuracies for other clauses like `SELECT` and `AGGREGATE` are over 90% whereas the accuracy for `WHERE` is only around 70%. We aim to conduct the following two studies: (i) Understanding the difficulties of the task and (ii) Investigating alternative solutions to realize the potential ceiling performance.

To this end, we first conduct a careful analysis on a subset of the WikiSQL data to identify the main challenges. This analysis leads to two important observations: (i) ∼17% of the questions are either too ambiguous (hard) or require external knowledge to answer, (ii) ∼68% of the questions can be answered by exact match or simple paraphrasing, however we surprisingly find that the current best system (Xu et al.) can only get less than 80% accuracy on such simple questions. Deeper analysis on the second observation leads to the conclusion that most of the errors in this category are due to wrongly generated condition values as shown by the example in Figure 1. One can resort to soft/hard look-up based approaches over table content as in previous work (Mou et al., 2017; Iyer et al., 2017) or user interaction (which is also applicable to the first observation) to more accurately recognize the entities in questions for better condition value generation.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1702–1711
Brussels, Belgium, October 31 - November 4, 2018.

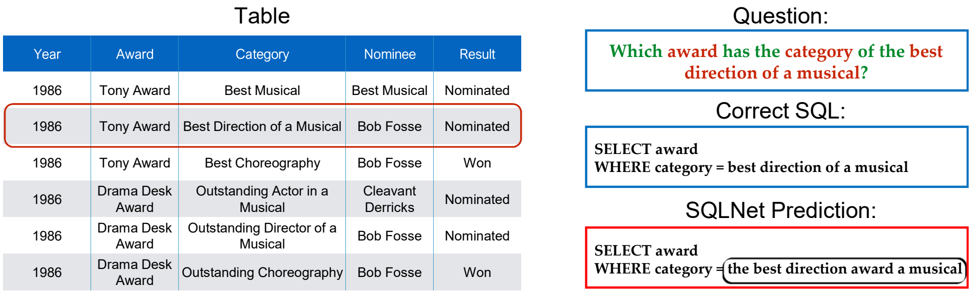

Table

Year	Award	Category	Nominee	Result
1986	Tony Award	Best Musical	Best Musical	Nominated
1986	Tony Award	Best Direction of a Musical	Bob Fosse	Nominated
1986	Tony Award	Best Choreography	Bob Fosse	Won
1986	Drama Desk Award	Outstanding Actor in a Musical	Cleavant Derricks	Nominated
1986	Drama Desk Award	Outstanding Director of a Musical	Bob Fosse	Nominated
1986	Drama Desk Award	Outstanding Choreography	Bob Fosse	Won

Question:

Which award has the category of the best direction of a musical?

Correct SQL:

SELECT award
WHERE category = best direction of a musical

SQLNet Prediction:

SELECT award
WHERE category = the best direction award a musical

Figure 1: An example from WikiSQL. SQLNet makes a wrong prediction on the condition value of the `WHERE` clause.

As a second contribution, we propose to use table content as additional data source to address the aforementioned wrongly generated condition value problem, and investigate solutions to realize the potential performance limit (upper bound) on WikiSQL. Note that neither Seq2SQL (Zhong et al., 2017) nor SQLNet (Xu et al.) utilize table content. However, we show that it is not straightforward to achieve a high accuracy even in the scenario where table content is available as an optimal external knowledge through our model ablation results and error analysis. We demonstrate that our proposed solutions can reach up to 88.6% `WHERE` condition accuracy, almost matching the performance on `SELECT` and `AGGREGATE`.

2 Background

The WikiSQL dataset introduced in (Zhong et al., 2017) is created from a large number of tables extracted from Wikipedia, employing Amazon Mechanical Turk for annotation. An example from the dataset is provided in Figure 1: It consists of a table t, a SQL query s, and its corresponding natural language question q. There can be multiple conditions in the `WHERE` clause of a SQL query, each of which consists of a table column, an operator ($=, <, >$, etc.), and a condition value.

Instead of asking annotators to write SQL queries for given questions and tables, the authors (Zhong et al., 2017) facilitate the annotation process by paraphrasing generated questions. This raises concerns that the resulting dataset is limited to only simple queries. We acknowledge this concern. However, it is still a large, valuable dataset towards the goal of building ultimate natural language interfaces to databases. If the existing or newly proposed solutions can not solve this task with high accuracy, how can we advance to more complicated ones? Any insight and solution to this task can help us build more advanced SQL synthesis algorithms in future.

3 WikiSQL Data Analysis

In this section, we aim to deliver a thorough analysis of the WikiSQL dataset. (Zhong et al., 2017; Xu et al.) made an assumption that only table schema is available to the model, and table content (i.e., table cell values) is not available. We want to answer the following question: As the dataset creation process involves several heuristics and predefined templates to simplify the annotation task, what kind of capabilities does it still require to answer the resulting questions successfully? To this end, we randomly sample 100 examples from the development set of WikiSQL for analysis.

3.1 Categorization

We manually categorize these 100 examples in terms of the capability needed to predict the correct `WHERE` clause as described below. We also provide illustrative examples for each category in Table 1. If an example belongs to multiple categories, we include it in all the categories that apply. In Table 2, we present the break down of the examples over these categories.

Exact match. For each condition in the SQL query, its column name appears in the question around the neighborhood of its condition value with exactly the same surface form.

Paraphrase. For at least one of the conditions in the SQL query, its column name is paraphrased in the question, hence the inference requires certain paraphrasing capabilities.

Partial clue. For at least one of the conditions in the SQL query, its column name is not explicitly mentioned in the question, not even in paraphrased form. However, there are still partial semantic clues for inference.

External knowledge. For at least one of the conditions in the SQL query, there is no clue in the question to infer its column name. Inferring this column name from the question requires external knowledge regarding the type of its condition

Category	Question	Pseudo SQL Query	Table Columns
Exact match	In what state was the **electorate** fowler?	SELECT state WHERE **electorate** EQL fowler	member, party, **electorate**, state, term in office
Paraphrase	What was the date of the game after week 5 **against the Houston Oilers?**	SELECT date WHERE week GT 5 AND **opponent EQL houston oilers**	week, date, **opponent**, result, attendance
Partial clue	Who had the most points in the game **on March 7**?	SELECT high points WHERE **date EQL march 7**	game, **date**, team, score, high points, ...
External-knowledge	Name the callback date **for amway arena**	SELECT callback date WHERE **audition venue EQL amway arena**	episode air data, audition city, **audition venue**, callback date, ...
Ambiguous	List the branding **for krca-tv**	SELECT branding WHERE **callsign EQL krca-tv**	branding, **callsign**, channel, power (kw), ...

Table 1: Representative examples for each category. **EQL** and **GT** correspond to the = and > operators. Highlighted parts of the question and the pseudo SQL query are provided to indicate clues for the category.

Category	# Questions	Percentage
Exact match	59	54.1%
Paraphrase	16	14.7%
Partial clue	15	13.8%
External knowledge	11	10.1%
Ambiguous	8	7.3%

Table 2: Number of examples in different categories.

Category	Seq2set+CA	Seq2set+CA+WE
Exact match	67.8%	72.9%
Paraphrase	75.0%	68.8%
Partial clue	80.0%	80.0%
External knowledge	54.6%	45.5%
Ambiguous	37.5%	37.5%
Total	67.0%	67.9%

Table 3: Accuracy breakdown of SQLNet.

value that can be detected from the question.

Ambiguous. For this category, even with the external knowledge it is almost impossible (even for humans) to confidently infer the correct condition column from the question.

3.2 Performance Breakdown of SQLNet

In Table 3, we show the performance breakdown of SQLNet (Xu et al.), the state-of-the-art model for WikiSQL, on the selected 100 examples. It has two variants with comparable performance, so we show both of them. As expected, both variants of SQLNet perform poorly on examples that are either too ambiguous or require external knowledge. However, it is surprising that the accuracy on examples that need only exact matching or simple paraphrasing is also not very high, especially considering the paraphrasing capabilities of deep learning models gained from distributed representations. We find that most of these errors are due to wrongly generated condition values as illustrated in Figure 1, where SQLNet fails to even produce a valid phrase. This is indeed important prior knowledge that can be effectively outsourced by resorting to soft/hard look-up based approaches (Mou et al., 2017; Iyer et al., 2017) instead of fully relying on models to precisely generate it.

The above observations exhibit an opportunity to incorporate external knowledge in the condition generation process. We opt to use the table content as the knowledge source to address the wrong condition value problem, which is not used in the existing models (Zhong et al., 2017; Xu et al.). Next we will show that it is not trivial to leverage table content.

4 Our Solutions

As motivated in the previous sections, our main objective here is to investigate solutions to realize the potential ceiling performance on WikiSQL when using the table content as an additional knowledge source. We first describe an attentional RNN-based model that serves as our baseline, and propose several variants for `WHERE` clause generation, each addressing a specific weakness.

4.1 Task Formulation

Given a question q and a table t, our objective is to generate the `WHERE` clause of the SQL query corresponding to the question. In addition, we assume that the table t can be queried while generating the `WHERE` clause. Each condition in the `WHERE` clause is represented as a triple of (`COLUMN`, `OP`, `VALUE`). SQLNet (Xu et al.) generates each of these individual components by first predicting `COLUMN` and `OP`, and then generating `VALUE` using pointer networks (Vinyals et al.). In

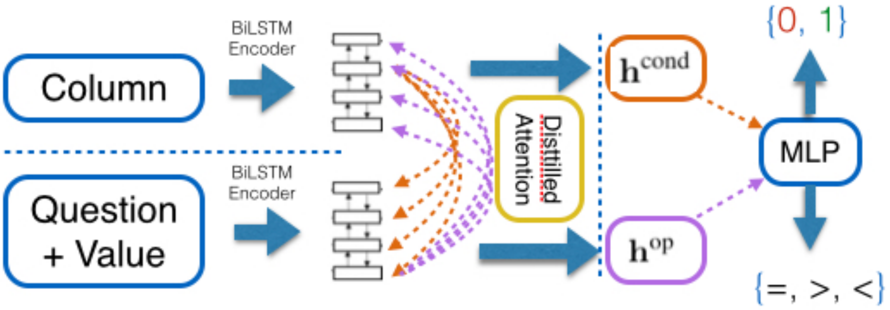

Figure 2: Model overview.

contrast, we propose a two step approach to tackle this problem in the reverse way, taking advantage of the content of table t as additional knowledge: (i) Generate the condition `VALUE` from the question, and (ii) predict which `COLUMN` and `OP` apply to this `VALUE`.

4.1.1 Candidate Generation

Our objective in candidate generation process is to produce a set of $(\texttt{COLUMN}, \texttt{VALUE})$ pairs from question q and table t, where `VALUE` is an *n-gram* in q and `COLUMN` is a column in t. Similar to prior work (Zhong et al., 2017; Xu et al.), we assume that `VALUE` occurs in the question.

We first generate the set $\mathcal{N}$ of all *n-grams* from the question. Then, for each candidate $v \in \mathcal{N}$ and each column $c \in t$, we query table t to check whether value v is contained in any row of column c. Then, we create a set $\mathcal{C} = \{(c, v) \colon v \in c\}$ of $(\texttt{COLUMN}, \texttt{VALUE})$ pairs as our final candidates for the `WHERE` clause. We note here that `VALUE` may not necessarily be an *n-gram* in question for other potential external knowledge or NLIDB tasks. In such scenarios, we can alternatively resort to soft look-up based approaches like the ones proposed in (Mou et al., 2017; Bordes et al., 2015).

4.1.2 Condition Prediction

Given a question q, SQL table t, and a set $\mathcal{C}$ of $(\texttt{COLUMN}, \texttt{VALUE})$ pair as candidates for `WHERE` clause, our objective is to learn two mappings:

$$f_{\text{cond}} : (q, c, v) \mapsto \{0, 1\} \qquad (1)$$

$$f_{\text{op}} : (q, c, v) \mapsto \{=, >, <\} \qquad (2)$$

where f_{cond} determines whether to select $(c, v) \in \mathcal{C}$ as a condition in `WHERE` clause and f_{op} predicts the operator `OP` $\in \{=, >, <\}$. These two mappings together can fully determine the final `WHERE` clause. Note that there can be multiple conditions in the `WHERE` clause.

4.2 Model Overview

We design a neural network model (Figure 2) to instantiate the mappings f_{cond} and f_{op}. We first describe the general structure of the model, consisting of the following steps:

Value Context. We define *value context* as the context of the value in the question for a given pair of question q and value v. Later the question encoder will condition on this information to make predictions. We will investigate different options for value context in Section 4.3. For now, we define it as a transformation g_{context} which maps each (q, v) to its value context.

Value Abstraction. Inspired by (Yavuz et al., 2016), we define *value abstraction* as a transformation g_{abstract} that replaces the surface form of `VALUE` in the question with a single token `ENTITY`. For the running example in Figure 1, applying value abstraction maps the question to "*Which award has the category of the ENTITY?*" It further informs the question encoder regarding the location of the `VALUE` in the question.

Encoding. Given question $q = (q_1, q_2, \ldots, q_m)$, column $c = (c_1, c_2, \ldots, c_n)$, and value $v = (v_1, v_2, \ldots, v_k)$. Before encoding, we first apply the aforementioned textual transformations and obtain value context

$$q' = (q'_1, q'_2, \ldots, q'_l) = g_{\text{context}} \circ g_{\text{abstract}}(q, v).$$

We first encode all the words into a vector space using an embedding matrix $E \in \mathbb{R}^{d \times |\mathcal{V}|}$, where $\mathcal{V}$ denotes the vocabulary and d is the embedding dimension. Let $\mathbf{q'}_i$ denote the embedding of word q'_i. To obtain the contextual embeddings, we use a bi-directional LSTM with hidden unit size h:

$$\overrightarrow{\mathbf{h}^{\text{q}}}_i = LSTM^{\text{q}}_{\text{fwd}}(\overrightarrow{\mathbf{h}^{\text{q}}}_{i-1}, \mathbf{q'}_i) \qquad (3)$$

$$\overleftarrow{\mathbf{h}^{\text{q}}}_i = LSTM^{\text{q}}_{\text{bwd}}(\overleftarrow{\mathbf{h}^{\text{q}}}_{i+1}, \mathbf{q'}_i) \qquad (4)$$

with $\overrightarrow{\mathbf{h}}_0 = \mathbf{0}$ and $\overleftarrow{\mathbf{h}}_{l+1} = \mathbf{0}$. We combine forward and backward contextual embeddings for each word in the question to obtain $\mathbf{h}^{\mathrm{q}}_i = \mathbf{concat}(\overrightarrow{\mathbf{h}^{\mathrm{q}}}_i, \overleftarrow{\mathbf{h}^{\mathrm{q}}}_i) \in \mathbb{R}^{2h}$.

Similarly, we obtain the contextual embedding $\mathbf{h}^{\mathrm{col}}_j \in \mathbb{R}^{2h}$ for each word c_j in column c with another bi-directional LSTM but shared word embedding matrix.

Distilled Attention. The objective of this step is to distill the most relevant information from both the value context and the column for the final prediction. We first compute the attention score of each word $c_j \in c$ on the words $q'_i \in q'$:

$$S^{(\mathrm{col}\rightarrow\mathrm{q})}_{j,i} = \mathbf{h}^{\mathrm{col}}_j \mathbf{W}^{(\mathrm{col}\rightarrow\mathrm{q})} \mathbf{h}^{\mathrm{q}}_i \tag{5}$$

$$P^{(\mathrm{col}\rightarrow\mathrm{q})}_j = \mathrm{softmax}_i \; S^{(\mathrm{col}\rightarrow\mathrm{q})}_{j,i} \tag{6}$$

where $\mathbf{W}^{(\mathrm{col}\rightarrow\mathrm{q})} \in \mathbb{R}^{2h\times 2h}$ is a model parameter to be learned and $P^{(\mathrm{col}\rightarrow\mathrm{q})}_j \in \mathbb{R}^{l}$ represents an attention probability distribution of word c_j over the words of value context q'. Let $P^{(\mathrm{col}\rightarrow\mathrm{q})} \in \mathbb{R}^{n\times l}$ denote the column-wise concatenation of $P^{(\mathrm{col}\rightarrow\mathrm{q})}_j$ indicating the unified representation of attention matrix from column words onto value context. Similarly, we compute the attention from value context (question) to column and get $P^{(\mathrm{q}\rightarrow\mathrm{col})} \in \mathbb{R}^{l\times n}$.

The intuition behind *distilled attention* is to allow two-way comparison to clean up the attention weights. To exemplify this point better, consider a scenario where a word $c_j \in c$ attends on a word $q'_i \in q'$ with high probability, but q'_i is much more relevant to another word $c_{j'} \in c$. We leverage reverse attention $P^{(\mathrm{q}\rightarrow\mathrm{col})}$ to distill the attention weights of column words on the value context. To this end, we define distilled attention weights as

$$P = (P^{(\mathrm{q}\rightarrow\mathrm{col})})^{\top} \odot P^{(\mathrm{col}\rightarrow\mathrm{q})} \tag{7}$$

where $P \in \mathbb{R}^{n\times l}$ becomes our final attention weights and $\odot$ indicates the Hadamard product.

Value Context and Column Representations. Having defined how the encodings and attention weights are computed, we now describe the final value context and column representations. First, we compute a value context vector for each word $c_j \in c$ using the distilled attention weights by

$$\mathbf{h}^{(\mathrm{q}\rightarrow\mathrm{col})}_j = \sum_{i=1}^{l} P_{j,i}\, \mathbf{h}^{\mathrm{q}}_i \tag{8}$$

and fuse it with the corresponding column context vector by

$$\mathbf{h}^{\mathrm{cond}}_j = \tanh(\mathbf{W}^{\mathrm{cond}}_0 \mathbf{h}^{\mathrm{col}}_j + \mathbf{W}^{\mathrm{cond}}_1 \mathbf{h}^{(\mathrm{q}\rightarrow\mathrm{col})}_j) \tag{9}$$

$$\mathbf{h}^{\mathrm{op}}_j = \tanh(\mathbf{W}^{\mathrm{op}}_0 \mathbf{h}^{\mathrm{col}}_j + \mathbf{W}^{\mathrm{op}}_1 \mathbf{h}^{(\mathrm{q}\rightarrow\mathrm{col})}_j) \tag{10}$$

where $\mathbf{W}^{\mathrm{cond}}_0, \mathbf{W}^{\mathrm{cond}}_1, \mathbf{W}^{\mathrm{op}}_0, \mathbf{W}^{\mathrm{op}}_1 \in \mathbb{R}^{2h\times 2h}$ are trainable model parameters. Finally, we apply a pooling layer to obtain fixed-size representations

$$\mathbf{h}^{\mathrm{cond}} = \frac{1}{n}\sum_j \mathbf{h}^{\mathrm{cond}}_j \text{ and } \mathbf{h}^{\mathrm{op}} = \frac{1}{n}\sum_j \mathbf{h}^{\mathrm{op}}_j \tag{11}$$

for *condition* and *operator* predictions.

Prediction. So far we have obtained two different representations $\mathbf{h}^{\mathrm{cond}} \in \mathbb{R}^{2h}$ and $\mathbf{h}^{\mathrm{op}} \in \mathbb{R}^{2h}$ as the latent unified representations of question, column, and value triple (q, c, v). We than use these representations to make the final predictions:

$$\mathbf{p}^{\mathrm{cond}} = \mathrm{softmax}(\mathbf{U}^{\mathrm{cond}}\, \mathbf{h}^{\mathrm{cond}}) \in \mathbb{R}^2 \tag{12}$$

$$\mathbf{p}^{\mathrm{op}} = \mathrm{softmax}(\mathbf{U}^{\mathrm{op}}\, \mathbf{h}^{\mathrm{op}}) \in \mathbb{R}^3 \tag{13}$$

where $\mathbf{U}^{\mathrm{cond}} \in \mathbb{R}^{2\times 2h}$ and $\mathbf{U}^{\mathrm{op}} \in \mathbb{R}^{3\times 2h}$ are model parameters. The final prediction mappings are then defined as:

$$f_{\mathrm{cond}}(q, c, v) = \arg\max_i \mathbf{p}^{\mathrm{cond}}_i \tag{14}$$

$$f_{\mathrm{op}}(q, c, v) = \arg\max_i \mathbf{p}^{\mathrm{op}}_i \tag{15}$$

Training Objective. Let $\mathcal{T} = \{(q, c, v, l, o)\}$ denote a set of training tuples where $l \in \{0, 1\}$ denotes whether to include a condition with (c, v) as $(\texttt{COLUMN}, \texttt{VALUE})$ pair and if so, $o \in \{=, >, <\}$ indicates which `OP` to apply. Our loss function consists of two components:

$$J(\Theta) = J_{\mathrm{cond}}(\Theta) + l\; J_{\mathrm{op}}(\Theta) \tag{16}$$

where $J_{\mathrm{cond}}(\Theta) = -\log(\mathbf{p}^{\mathrm{cond}}_l)$ and $J_{\mathrm{op}}(\Theta) = -\log(\mathbf{p}^{\mathrm{op}}_o)$ are simply negative log-likelihood losses for *condition* and *operator* predictions.

Inference. We also employ an inference schema where we make a simple assumption that a condition value v may be a part of *only one* condition. Hence, we group candidate $\{(c, v)\}$ pairs by value v, and create a candidate column set $C_v\colon \{c\colon (c, v)\}$ for each unique candidate value v. Based on the probabilities $p^{\mathrm{cond}}(c, v)$ computed by the trained model, we select the column $c \in C_v$ with the maximum probability for each candidate value v, hence form the set of $(\texttt{COLUMN}, \texttt{VALUE})$ pairs to be included in condition this way.

4.3 Model Variants

In this section, we discuss variants of our model with different choices for the value context. Consider the running example in Figure 1 and assume

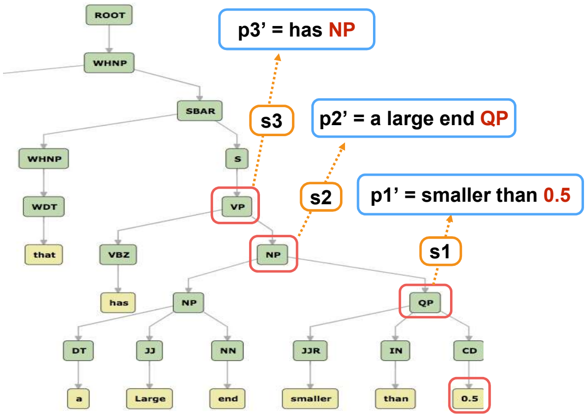

Figure 3: Partial view of parse tree for question "*Which taper/ft that has a Large end smaller than 0.5*" illustrating parse tree based value contexts.

hypothetically that *award* is a candidate condition column for value *best direction of a musical*. When we use the whole question as the value context, eliminating *award* from being a condition `COLUMN` for this `VALUE` becomes very challenging for the model as it is not informed enough regarding the finer context of the value.

4.3.1 Base Model

The base model simply uses identity mapping for value context. More precisely, we use the whole question as the context for the candidate condition `VALUE`. So, $g_{\text{context}}(q, v) = q$.

4.3.2 Window-based model

The objective of *window-based* model is to get value contexts that can provide more clean context information by leveraging the context window around the candidate value. In this case, we first identify the span $[start, end]$ for value v in the question q. Based on its span and a predetermined context window size w, we define $g_{\text{context}}(q, v) = (q_{start-w}, \ldots, q_{start}, \ldots, q_{end}, \ldots, q_{end+w})$.

4.3.3 Parse tree-based model

We also analyze a simple parse tree based model that hierarchically split up the value context into multiple contexts based on the constituency parse tree of the question. To motivate this, consider the question in Figure 3. Window-based value context for 0.5 is *"a large end smaller than ENTITY"* and candidate table columns to apply this value are *"large end"* and *"small end"*. Based on this context, a model will likely assign a higher probability to *"large end"* than *"small end"*. The syntactic structure of the question can potentially help reduce such ambiguities.

Parse tree-based value contexts. As highlighted in Figure 3, we use nested phrase-level subtrees that contain the candidate `VALUE`. Moreover, to better inform the model regarding the type of phrases, we use phrase level constituent tags of subtrees in two ways: (i) to inform the parent tree with the type of its subtree containing candidate `VALUE`, (ii) to apply a tag-specific affine transformation on phrase representations. To this end, we first select the r-lowest subtrees $s_1 \subset s_2 \subset \ldots \subset s_r$ of the question's parse tree containing `VALUE` along with their corresponding phrase-level constituency tags[1] $t_1, t_2, \ldots, t_r$, respectively. We then iteratively form multiple values contexts as nested phrases of $p'_1, p'_2, \ldots, p'_r$ shown in Figure 3 as follows: (i) p'_1 is equal to the phrase formed by the words at the leaf nodes of subtree s_1, and then (ii) iteratively form p'_i as the phrase formed by the words under subtree s_i by replacing its subphrase corresponding to subtree s_{i-1} with its constituency tag t_{i-1} for $i > 1$.

Modifications to General Model. We now describe how the general model defined in Section 4.2 is adapted to accommodate the multiple nested value contexts of different types. In this process, we aim to capture two types of important information: (i) syntactic phrase-level types of value contexts, and (ii) their distance to `VALUE`.

We first compute a value context vector $\mathbf{h}_{j,k}^{(\text{q}\rightarrow\text{col})} \in \mathbb{R}^{2h}$ for each word $c_j \in c$ and each value context p'_k as in Eq. 8 using the same encoding and attention mechanisms. However, we replace the affine transformation layers defined in Eq. 9 and 10 with tag and distance specific ones as follows:

$$\mathbf{h}_j^{\text{cond}} = \tanh(\mathbf{W}_0^{\text{cond}}\mathbf{h}_j^{\text{col}} + \frac{1}{r}\sum_{k=1}^{r}\mathbf{W}_{k,t_k}^{\text{cond}}\mathbf{h}_{j,k}^{(\text{q}\rightarrow\text{col})}) \tag{17}$$

$$\mathbf{h}_j^{\text{op}} = \tanh(\mathbf{W}_0^{\text{op}}\mathbf{h}_j^{\text{col}} + \frac{1}{r}\sum_{k=1}^{r}\mathbf{W}_{k,t_k}^{\text{op}}\mathbf{h}_{j,k}^{(\text{q}\rightarrow\text{col})}) \tag{18}$$

where $\mathbf{W}_0^{\text{cond}}, \mathbf{W}_{k,t_k}^{\text{cond}}, \mathbf{W}_0^{\text{op}}, \mathbf{W}_{k,t_k}^{\text{op}} \in \mathbb{R}^{d\times 2h}$ for $k = 1, 2, \ldots, r$ are model parameters. It is important to note here that k determines the distance of value context to `VALUE` and t_k indicates its tag, effectively making the fusion layers above *tag* and

---

[1] We use the Penn treebank annotation conventions that are described in Bies et al. (1995) at http://languagelog.ldc.upenn.edu/myl/PennTreebank1995.pdf

Model	Dev	Test
No External Knowledge		
Seq2Sql (Zhong et al., 2017)	62.1%	60.2%
SqlNet-(Seq2Set + CA)	72.1%	70.0%
SqlNet-(Seq2Set + CA + WE)	74.1%	71.9%
SqlNet*-(Seq2Set + CA)	72.3%	70.9%
SqlNet*-(Seq2Set + CA + WE)	73.8%	71.7%
In Our Scenario		
SqlNet*-(Seq2Set + CA)	82.2%	80.9%
SqlNet*-(Seq2Set + CA + WE)	**83.5%**	**81.8%**
Stamp + RL (Sun et al., 2018)	77.3%	76.3%
TypeSql + TC (w/ Freebase) (Yu et al., 2018)	**92.8%**	**87.9%**
Our Baseline	**77.2%**	**77.1%**
SqlMaster	84.8%	83.9%
SqlMaster + VA	86.2%	86.1%
SqlMaster (Window-Based)	87.4%	87.1%
SqlMaster (Window-Based) + VA	87.9%	87.6%
SqlMaster (Tree-Based)	88.7%	88.3%
SqlMaster (Tree-Based) + VA	**88.9%**	**88.6%**
Our UpperBound	**92.1%**	**91.4%**

Table 4: `WHERE` clause accuracy. +VA denotes that *Value Abstraction* is applied. SqlNet* refers to our reimplementation of SQLNet. SqlMaster refers to our proposed models. Only TypeSql + TC leverages Freebase on top of table content among the models reported in our scenario. WHERE clause accuracy of TypeSql (w/o Freebase) is not reported.

distance specific. The rest of the model exactly follows the Eq. from 11 through 15 with the same training objective and inference schemes.

5 Experiments

In this section, we discuss the details of the experiments and present our main results.

5.1 Training Details

For training our neural networks, we only keep the words appearing at least 3 times in the whole training data and the rest of the words are replaced with `UNK` token. Word embeddings are initialized with pretrained GloVe (Pennington et al., 2014) vectors[2], and updated during the training. We take the dimension of word embeddings and the size of LSTM hidden layer both equal to 100. The model parameters are optimized using Adam (Kingma and Ba, 2015) with batch size of 32 and a decaying learning rate starting with 0.001. We apply gradient clipping to 5 when its norm exceeds this value. We use early stopping based on the model accuracy on the development set. We report our results with a model snapshot achieving the best accuracy on the development set. Our models are implemented in *tensorflow* (Abadi et al., 2016). The code is available at `https://github.com/semihyavuzz/sql_master`.

Model	Dev	Test
SqlNet (Xu et al.)	63.2%	61.3%
DialSql (Gur et al., 2018)	70.9%	69.0%
TypeSql (w/o Freebase) (Yu et al., 2018)	66.5%	64.9%
TypeSql + TC (w/ Freebase) (Yu et al., 2018)	**79.2%**	**75.4%**
SqlMaster (Ours, w/o Freebase)	**73.1%**	**72.4%**

Table 5: Full query-match (QM) accuracy. For SqlMaster, we combine our WHERE clause predictions with the SELECT and AGGREGATE clause predictions of SQLNet. QM result for TypeSql + TC (w/o Freebase) is not reported.

5.2 Main Results

In Table 4, we present our main results in comparison with the related works. Baseline refers to a baseline for our models where the `WHERE` clause accuracy is computed by assuming that each candidate $(\texttt{COLUMN}, \texttt{VALUE})$ pair is included with corresponding `OP` being equality. On the other hand, UpperBound accuracy is computed by assuming f_{cond} and f_{op} makes 100% correct mapping of whether to include a candidate $(\texttt{COLUMN}, \texttt{VALUE})$ pair and which `OP` to apply on this condition. In other words, errors in UpperBound exist due to wrong candidate generation.

As shown in Table 4, our models surpass the previous results by a large margin as well as its variants improving upon each other. A portion of these improvements definitely come from assuming and using the table itself as the optimal external knowledge. Acknowledging this fact, we make the following more important conclusions from Table 4, 5, and 6: (i) Comparison of the performance results across scenarios (SqlNet vs. SqlMaster or TypeSql) reveals that there is a large room for improvement when an external knowledge base is used, (ii) Comparison of our own models with its variants demonstrates that each component/extension incorporated brings a considerable performance improvement, justifying its potential power to be used in other related NLP tasks, (iii) Comparing our models with SqlNet and TypeSql+TC within our scenario provides further clues/justifications towards effectiveness of our proposed Tree-Based model, (iv) SqlMaster performs comparably to TypeSql + TC (Yu et al., 2018) on WHERE condition predictions despite the fact that we do not use Freebase, which is exploited in (Yu et al., 2018) to identify named entities of certain Freebase types (e.g., *person*, *place*, *country*, *organization*, *sport*), (v) Our SqlMaster (Tree-Based) + VA model achieves 88.6% on test portion of WikiSQL, which almost reaches the upper bound of 91.4%, demonstrating a great promise for future work in this do-

---

[2] More specifically, we use 100D vectors from http://nlp.stanford.edu/data/glove.6B.zip

Category	SQLNET	SQLMASTER
EXACT MATCH	72.9%	86.4%
PARAPHRASE	68.8%	93.8%
PARTIAL CLUE	80.0%	93.3%
EXTERNAL KNOWLEDGE	45.5%	90.9%
AMBIGUOUS	37.5%	75.0%
TOTAL	67.9%	88.1%

Table 6: Accuracy breakdown of SQLNET compared to SQLMASTER over the categories obtained by hand-analysis on the randomly selected examples as explained in Section 3.

main, and finally (vi) When the performance of our model is compared to SQLNET over the categories as shown in Table 6, we observe that it consistently improves the performance of over all the categories, but most noticeably on EXTERNAL KNOWLEDGE and AMBIGUOUS ones which were the main categories inspiring this work and proposed approaches motivated by the analysis provided in Section 3.

Evaluation in Our Scenario. We adapt SQLNet (Xu et al.) results to our scenario via a post-processing step as follows: For each of their predicted condition in `WHERE` clause, if column c and operator o are both correct, but value v is wrong, then we replace this value with the one in our generated candidates that maps to the column c when the mapping is unique (only one value maps to c).

5.3 Error Analysis

In this section, we provide an error analysis of our models to better understand what are the remaining challenges to achieve UPPERBOUND performance. To this end, we analyzed 100 randomly sampled examples from development set on which our best model fails. 41% of these errors are due to our models not being able to perform a good semantic understanding of the value context. 36% of the errors correspond to ambiguous questions that lack sufficient information to disambiguate between correct and wrong column. A good representative example for this category is "*Who was the director of king's speech?*", where our model predicts "*winner and nominees*" column for the condition value "*king's speech*" while the correct column is "*original title*". The remaining 18% and 5% of the errors are caused by sparsity of column names and wrong labelling problems, respectively.

6 Related Work

Research on natural language interfaces to databases (NLIDBs) and semantic parsing has spanned several decades. Early rule-based NLIDBs (Woods, 1973; Androutsopoulos et al., 1995; Popescu et al., 2003) employ carefully designed rules to map natural language questions to formal representations like SQL queries. While having a high precision, rule-based systems are brittle when facing with language variations and usually only admit inputs from a restricted vocabulary. The rise of statistical models (Zettlemoyer and Collins, 2005; Kate et al., 2005; Berant et al., 2013) and neural network models (Yih et al., 2015; Dong and Lapata, 2016; Sun et al., 2016; Zhong et al., 2017; Xu et al.) has enabled NLIDBs that are more robust to language variations. Such systems allow users to formulate questions with greater flexibility instead of having to probe and adapt to the boundary of rule-based systems.

Along with the advance in modeling is the development of benchmarks for training and testing NLIDB models. Early benchmarks are mostly curated by experts (Zelle and Ray, 1996; Zettlemoyer and Collins, 2007). State-of-the-art models (Dong and Lapata, 2016) have achieved a high accuracy of 80% to 90% on these benchmarks. In the recent years, a number of large-scale, crowd-sourced benchmarks have been constructed with the goal to train and test NLIDBs in a more real-world setting, notably WebQuestions (Berant et al., 2013) and GraphQuestions (Su et al., 2016) for knowledge bases, and WikiSQL (Zhong et al., 2017) for SQL queries to relational databases. The best accuracies on these benchmarks are still far from enough for real use, typically in the range of 20% to 60%.

Besides releasing WikiSQL, Zhong et al. (2017) propose an approach (Seq2SQL) to solve this task. Seq2SQL leverages the pointer-networks (Vinyals et al.) to generate linearized SQL queries token-by-token using the input question and table schema. They report significant performance improvement over (Dong and Lapata, 2016), a generic sequence-to-tree approach proposed for semantic parsing. More recently, Xu et al. propose a sketch-based sequence-to-set approach (SQLNet) eliminating sequence-to-sequence structure employed in (Zhong et al., 2017), when the order does not matter. In our work, we provide a careful analysis of SQLNet results to better understand the limitations of this model on the WikiSQL task. Inspired by this analysis, we propose novel solutions to realize close to upper-bound condition accuracy in the scenario where SQL table is available as an optimal external knowledge. Another recent work (Yu et al., 2018) also focuses on using external knowledge (Freebase) along with the table content to generate SQL queries in a type aware fashion. A concurrent line of related work exploits graph-to-sequence neural models with the aim to better exploit syntactic information in the input question (Xu et al., 2018a,b). On the other hand, Gur et al. (2018) takes an orthogonal approach and introduces a dialogue-based query refinement mechanism where a candidate SQL query (generated by any black-box model) is refined by interactively validating and updating modular segments of the query with users. The authors show that by having successful interactions with users, not only the accuracy of the candidate queries can be improved but also new insights into limitations of current query generation systems can be gained.

There are also a number of recent studies on semantic parsing for semi-structured tables. For example, Pasupat and Liang (2015) develop the WikiTableQuestions benchmark, where the task is to find table cells in HTML table to answer questions, while Jauhar et al. (2016) focus on multi-choice questions. On the other hand, Sun et al. (2016) study how to answer user questions with table cells from millions of HTML tables. These studies directly find cells of semi-structured tables as answers, instead of generating SQL queries for relational databases.

7 Conclusion

In this paper, we thoroughly analyzed the recently released WikiSQL dataset and the performance breakdown of SQLNet. Through the analysis, we identified an opportunity/need to further explore the potentials of incorporating external knowledge in the structured query generation process. In this direction, we developed alternative solutions to explore the potential performance limits for this task in the scenario where table content can be used. We showed that our proposed systems can reach up to 88.6% accuracy in condition generation and provided a discussion regarding what the remaining challenges were through an error analysis. We consider solving the WikiSQL task as a necessary preliminary step towards realizing natural language interfaces to databases in full fledge.

Acknowledgements

This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053 and NSF 1528175. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*.

Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(1):29–81.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for treebank ii style penn treebank project. In *Linguistic Data Consortium*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. 2018. Dialsql: Dialogue based structured query generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sujay Kumar Jauhar, Peter Turney, and Eduard Hovy. 2016. Tables as semi-structured knowledge for question answering. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. 2005. Learning to transform natural to formal languages. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Lili Mou, Zhengdong Lu, Hang Li, and Zhi Jin. 2017. Coupling distributed and symbolic execution for natural language queries. In *International Conference on Machine Learning (ICML)*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *World Wide Web (WWW)*.

Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic parsing with syntax- and table-aware sql generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems (NIPS)*.

William A Woods. 1973. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the American Federation of Information Processing Societies Conference*.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018a. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018b. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Xiaojun Xu, Chang Liu, and Dawn Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

John M Zelle and Mooney Ray. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Better Transition-Based AMR Parsing with a Refined Search Space

Zhijiang Guo and **Wei Lu**
Singapore University of Technology and Design
8 Somapah Road, Singapore, 487372
`zhijiang_guo@mymail.sutd.edu.sg, luwei@sutd.edu.sg`

Abstract

This paper introduces a simple yet effective transition-based system for Abstract Meaning Representation (AMR) parsing. We argue that a well-defined search space for a transition system is crucial for building an effective parser. We propose to conduct the search in a refined search space based on a new compact AMR graph and an improved oracle. Our end-to-end parser achieves the state-of-the-art performance on various datasets with minimal additional information.[1]

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a formalism that captures the semantics of a sentence with a rooted directed graph, in which nodes represent the *concepts* and edges represent the *relations* between concepts. An example AMR graph together with its corresponding sentence are illustrated in Figure 1.

AMR parsing, the task of transforming a sentence into its AMR graph, is a challenging task as it requires the parser to learn to predict not only concepts, which consist of predicates, lemmas, named entities, wiki-links and co-references, but also a large number of relation types based on relatively sparse training data (Peng et al., 2017). Given the challenges, many state-of-the-art AMR parsers employ various external resources and adopt a pipeline approach (Flanigan et al., 2016; Wang and Xue, 2017; Foland and Martin, 2017; van Noord and Bos, 2017). Recently, Damonte et al. (2016); Ballesteros and Al-Onaizan (2017); Peng et al. (2018) have successfully developed AMR parsers in an end-to-end fashion using a transition-based approach.

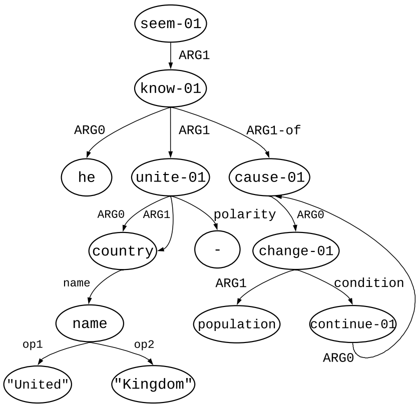

It seems that he knows under the population changes (if it continues) that United Kingdom will not unite.

Figure 1: An example AMR graph for a sentence.

The transition-based approach (Yamada and Matsumoto, 2003; Nivre, 2003, 2004) has been popular among many NLP tasks, including syntactic parsing (Zhang and Clark, 2011; Chen and Manning, 2014), named entity recognition (Lample et al., 2016), and semantic parsing (Cheng et al., 2017). Different from the graph-based approach for structured prediction (e.g., conditional random fields (Lafferty et al., 2001)), such an approach is able to maintain a good balance between efficiency and accuracy (Nivre and McDonald, 2008), and has achieved state-of-the-art results on a number of tasks (Swayamdipta et al., 2016; Shi et al., 2017; Cheng et al., 2017).

While the transition-based approach is promising for AMR parsing, existing transition-based AMR parsers still cannot attain the state-of-the-art results on such a task. We observe that the key to the development of an effective transition-based system is a properly defined search space, and argue that the search space used in existing transition systems needs to be refined. Inspired by (Wang et al., 2015), we design a new compact AMR graph representation. Transition actions designed based on such a compact graph enable our parser to generate the target structure with fewer actions and to better capture the correspondence

---

[1] Dynet (Neubig et al., 2017) is used to implement our parser. We make the supplementary material and code available at `http://statnlp.org/research/sp`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1712–1722
Brussels, Belgium, October 31 - November 4, 2018.

between concepts and tokens in the sentence.

Oracle, the algorithm used at the training time for specifying the action sequence that can recover the gold AMR graph, is also crucial for the transition-based system. A good oracle will be able to teach the parser how to find a proper path in the search space. The oracle requires alignment information between words and concepts. We identify limitations associated with current practice for finding such alignment information, and propose a new approach that integrates both rules and unsupervised learning.

Experiments show that our system that makes use of POS tags as the only external resource performs competitively on benchmark datasets. To the best of our knowledge, the parser achieves the highest score on the standard LDC2014T12 dataset. Our parser also yields competitive scores on the LDC2015E86 dataset and the more recent LDC2017T10 dataset. On the popular newswire section of LDC2014T12 dataset, our parser outperforms the previous best system by around 3 points in terms of F_1 measure.

2 Related Work

Since the AMR graph encodes rich information, it has been explored for many downstream applications such as language generation (Song et al., 2016), information extraction (Huang et al., 2016) and machine comprehension (Sachan and Xing, 2016). Currently, most parsers can be categorized into 4 classes: 1) *Tree*: such approaches incrementally convert a dependency tree into its corresponding AMR graph (Wang et al., 2015; Goodman et al., 2016; Barzdins and Gosko, 2016); 2) *Graph*: the graph-based models calculate scores of edges and then use a maximum spanning connected subgraph algorithm to select edges that will constitute the graph (Werling et al., 2015; Flanigan et al., 2016); 3) *Seq2seq*: the models adapted from sequence-to-sequence (Sutskever et al., 2014) methods (Peng et al., 2017; Konstas et al., 2017); 4) *Transition*: the transition-based methods, whose input is the plain text sentence and the output is the corresponding graph (Zhou et al., 2016; Damonte et al., 2016; Ballesteros and Al-Onaizan, 2017; Peng et al., 2018).

Apart from these models, Peng et al. (2015) introduce a synchronous hyperedge replacement grammar solution. Pust et al. (2015) regard the task as a machine translation problem, while Artzi et al. (2015) adapt combinatory categorical grammar (Steedman and Baldridge, 2011) for it. Foland and Martin (2017) decompose the parsing task into many subtasks. Then they use multiple bidirectional LSTMs for identifying different types of concepts and relations and iteratively combine these components to form the AMR graph.

Because the mapping between AMR concepts and tokens in the input sentence is latent, external aligners have been developed for training purpose. The most popular aligner is JAMR (Flanigan et al., 2016), which greedily aligns input tokens to graph fragments by using a static template. Another aligner is ISI (Pourdamghani et al., 2014), which is a statistical approach that borrows techniques from statistical machine translation.

3 Approach

We adopt a transition-based approach for AMR parsing. We first propose a new compact representation for AMR graph. Based on our new representation, we further present a novel technique for constructing the action sequence used for training our model. As we will see later, both newly introduced techniques are crucial for building an improved transition-based system within our refined search space.

3.1 Compact AMR Graph

Inspired by (Wang et al., 2015), we design a representation called *compact AMR graph* to simplify concepts and relations of an AMR graph, which makes the learning of our transition system easier. The construction of our compact AMR graph involves removing concepts and relations from an original AMR graph.

Remove Concepts: First we categorize AMR concepts into 2 types:

- **Lexical**: concepts which are converted directly from tokens in the sentence into certain expressions ranging from predicates with sense tags, lemmas to tokens with quotation marks. One example is the concept seem-01 shown in Figure 1, which is converted from the token *seems* with 01 as the sense tag.
- **Non-Lexical**: concepts which are invoked by their child concepts rather than from tokens in the sentence directly. Examples include country and name in Figure 1.[2]

A non-lexical concept is invoked by its child concepts, while a lexical concept corresponds to

---

[2] A list of non-lexical concepts are provided in the supplementary material.

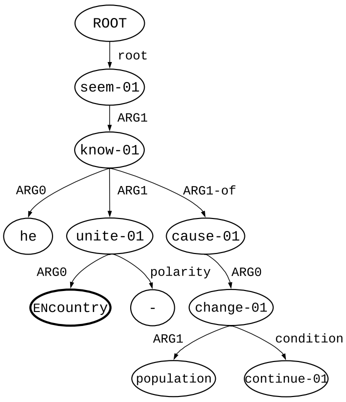

Figure 2: The compact AMR graph for the example in Figure 1.

a certain token in the input sentence. Inspired by (Lu et al., 2008) which learns a semantic parser by aligning words and semantic units, we compress a subgraph rooted by a non-lexical concept into a new concept, which directly corresponds to a contiguous span in the sentence.

For example, a subgraph that consists of non-lexical concepts: country, name and lexical concepts "United", "Kingdom" in Figure 1 can be compressed into one non-lexical concept ENcountry, which can be aligned to the span *"United Kingdom"* in the input sentence as shown in Figure 2.

Removing concepts in the graph enables the parser to build it with fewer actions. Empirically we find that 9% fewer actions are required in LDC2015E86 on average. Also, all concepts can be invoked by a contiguous span in the sentence, which helps the parser to capture the correspondence between concepts and tokens better.

Remove Relations: We also refine the search space by eliminating certain relations in the original AMR graph. The number of relation types is relatively large in the AMR corpus, which leads to a large search space. In order to introduce *reentrancy*[3], attached concepts have to stay in the stack rather than being removed as in most of transition-based AMR parsers.

Take cause-01 shown in the Figure 1 as an example, which is a reentrancy node headed by know-01 and continue-01. After it has been attached to its parent know-01 and child change-01, it still needs to stay in the stack and wait for another relation headed by continue-01. These two concepts are potentially far away from each other in the input sentence, which means cause-01 has to stay in the stack for a long time. In general, the longer a word has to wait to get assigned the more opportunities there are for something to go awry (Manning and Schütze, 1999).

During the testing phase, the parser does not know whether cause-01 is a reentrancy node. Therefore, the parser needs to add all possible relation attachment actions (nearly half of the action space) into the valid action set as long as cause-01 exists in the stack, which makes the parser harder to train.

Thus, we define several properties that the compact graph should respect to further refine the search space[4]:

- **Acyclicity**: cycles are forbidden. The relation ARG0 between cause-01 and continue-01 shown in Figure 1 is removed in the compact AMR graph Figure 2.
- **Simple**: for each parent-child concept pair, only one relation is attached. There exist two relations ARG0 and ARG1 between concepts unite and country shown in Figure 1. One of them is removed in Figure 2.
- **Non-terminal restricted**: only a subset of concepts are allowed to have children. For example, lexical concept – in Figure 1 can only be a terminal node in the compact AMR graph, which means that it can be removed from the stack once it has been attached.
- **Reentrancy restricted**: reentrancy is forbidden for certain concepts, which means these concepts only have one parent concept. An example is that lexical concepts – cannot be reentrancy in the compact AMR graph.

After incorporating these constraints, relation attachment actions are forbidden at many states, which refines the search space. Even though such constraints might prevent us from generating some valid AMR graphs, in practice we can convert the compact AMR graph back to the AMR graph without much loss[5].

3.2 Oracle

Another key to successful search is the oracle, an algorithm that produces a sequence of actions that lead to the gold AMR graph. The action sequence generated by the oracle is significant as it tells

---

[3] One concept can participate in multiple relations as concept country in Figure 1.

[4] In practice, we impose constraints on the valid action set of the parser for each state to ensure the growing structure always respect these properties. Details are provided in supplementary material.

[5] 95% of graphs in the training set of LDC2015E86 dataset satisfy these constraints, which means no relation needs to be removed for most of the graphs.

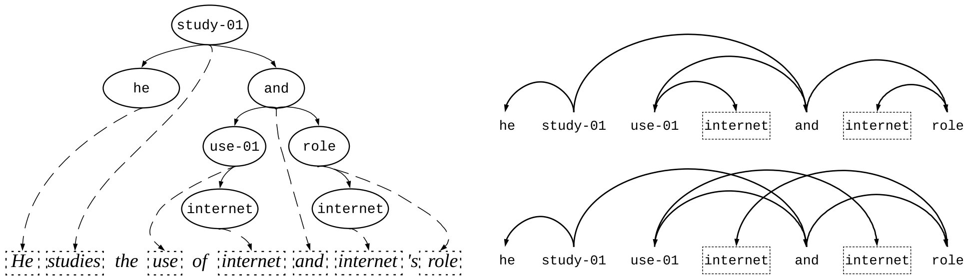

Figure 3: An alignment example for a sentence that has two identical tokens *internet* and the relation arcs between concepts if we put them on a semi-plane.

the model which actions should be taken during training. The oracle requires, as input, the alignment between tokens in the sentences and concepts in the graph. Almost all prior transition-based parsers have used the JAMR aligner for identifying such alignment information. However, we find that the JAMR aligner suffers from two issues, which may lead to errors that may be propagated to the parser through the oracle.

Non-Projectivity Caused by Alignment Error: Consider a sentence that has two identical tokens–*internet*. Figure 3 illustrates the alignments between concepts and tokens. Since our parser predicts the concepts from left to right, we can draw edges between aligned concepts on a semi-plane to verify whether it has crossing arcs.

If the alignment is correct, the AMR graph is projective as shown at the top right of Figure 3. However, the JAMR is a rule-based aligner, which uses a set of heuristics to do fuzzy matching. Its matching mechanism tends to make mistakes when there exist multiple identical words. Such alignment errors may lead to crossing arcs that our parser may not be able to handle.

Assume an alignment error occurs as illustrated in the bottom right of Figure 3. Such an error will force the parser to attach relations between wrong concepts. Sometimes these concepts are far from each other, which potentially leads to more crossing arcs. For example, ARG1 is attached to the wrong internet, causing the undesired crossing arc.

Empty Alignments:Another issue is that as the corpus gets larger, the JAMR aligner yields more empty alignments as shown in Table 3. Since the set of matching rules is static, unseen patterns in a larger corpus may result in sub-optimal alignment information.

Hybrid Aligner: In order to address these issues, we first try the ISI aligner (Pourdamghani et al., 2014). According to preliminary results (Table 3), we find that the performance of such an unsupervised model is not as good as the JAMR aligner when aligning relation and non-lexical concepts. Therefore, we propose a hybrid aligner, which combines unsupervised learning and rule-based strategy[6].

JAMR does not consider information about the structure whereas the unsupervised models can capture *locality* (Wang and Xue, 2017) – the assumption that words that are adjacent in the source sentence tend to align to words that are closer in the target sentence (Vogel et al., 1996). Structural information can also be incorporated into the model to allow it to capture locality beyond linearity (Wang and Xue, 2017). This property of the unsupervised aligner can alleviate the problem of non-projectivity caused by alignment error. Similar to what is done in the JAMR aligner, we also design rules based on properties of AMR graphs to improve alignments of non-lexical concepts.

In the preprocessing stage of the hybrid aligner, we remove all relations. As non-lexical concepts can be aligned to their child concepts in the compact graph, we then remove all non-lexical concepts. Our unsupervised method is based on IBM word alignment models (Brown et al., 1993).

In the postprocessing stage, we align non-lexical concepts iteratively to the same span that its child concepts are aligned to. For example, non-lexical concepts country and name shown in Figure 1 are removed in preprocessing. During postprocessing, their alignments come from child concepts “United” and “Kingdom”, they are aligned to the 16th token *“United”* and the 17th token *“Kingdom”* respectively. Therefore, non-lexical concepts country and name can be aligned to the span 16-17, which is *“United Kingdom”*.

3.3 Transition System

The transition system consists of a stack S containing words that have been processed, a buffer

---

[6] Details of the rules and how to apply them in the hybrid aligner are provided in the supplementary material.

Action	$State_t \rightarrow State_{t+1}$
SHIFT	$(S, u\|B) \rightarrow (u\|S, B)$
REDUCE	$(u\|S, B) \rightarrow (S, B)$
RIGHTLABEL(r)	$(u\|S, v\|B) \rightarrow (g_r(u, v, r)\|S, v\|B)$
LEFTLABEL(r)	$(u\|S, v\|B) \rightarrow (u\|S, g_r(v, u, r)\|B)$
SWAP	$(u, v\|S, B) \rightarrow (v, u\|S, B)$
MERGE	$(u\|S, v\|B) \rightarrow (S, g_m(u, v)\|B)$
PRED(n)	$(S, u\|B) \rightarrow (S, n\|B)$
ENTITY(ℓ)	$(S, u\|B) \rightarrow (S, g_\ell(u, \ell)\|B)$
GEN(n)	$(S, u\|B) \rightarrow (S, u, n\|B)$

Table 1: Definition of actions. $(u|S)/(u, v|S)$: item u (or u, v) is at the top of the stack. g_r, g_m and g_ℓ represent composition functions described in Section 4.3.

B containing words to be processed. Initially, S_1 is empty and B_1 contains the whole input sentence and a end-of-sentence symbol at the end. Execution ends on time step t such that B_t is empty and S_t contains a single structure.

Motivated by (Henderson et al., 2013; Ballesteros and Al-Onaizan, 2017), we design 9 types of actions summarized in Table 1. An example for parsing a sentence into its compact AMR graph shown in Figure 4 is provided in Table 2.

- **SHIFT**: removes an item from the front of the buffer and pushes it to the stack.
- **REDUCE**: pops the item on top of the stack.
- **RIGHTLABEL(r)**: creates a relation arc from the item on top of the stack to the item at the front of the buffer. Since these two items are not removed, they can be attached by another relation arc in the future. Therefore, reentrancy is allowed.
- **LEFTLABEL(r)**: creates a relation in the reverse direction as RIGHTLABEL.
- **SWAP**: swaps the top two items on the stack. This action allows non-projective relations (Nivre, 2009) and helps to introduce reentrancy. Repetitive SWAP actions are disallowed to avoid infinite swapping.
- **PRED(n)**: predicts the predicate and lemma concepts corresponding to the item at the front of the buffer. This action requires a look-up table M generated during the training phase. For example, *meet* is mapped to concepts such as meet, meet-01 and meet-02 in M. If the token *meet* is at the front of the buffer, then we add all its corresponding concepts based on M to the valid action sets.
- **MERGE**: removes the item at the front of the buffer and the item on top of the stack, then a composition function is applied to merge them into a new item. The item will be pushed back to the front of the buffer. This action serves to generate non-lexical concepts, whose corresponding span is larger than two. For example, if *"Iftik"* is on top of the stack and *"Ahmed"* is at the front of the buffer, a MERGE action will be applied. This action can be applied recursively if the span is larger than two.
- **ENTITY(ℓ)**: pops the item at the front of the buffer, and labels it with an entity label. This action is designed for named entities in non-lexical concepts. For example, after merging *"Iftik"* and *"Ahmed"*, ENTITY(ENperson) will be used to label the item.
- **GEN(n)**: creates a non-lexical concept invoked by the item at the front of the buffer, which can be any type of concept or composed representation after MERGE transition. Then the non-lexical concept is pushed back to the buffer right after the item. This action can be applied recursively. An example is that concept have-org-role-91 is invoked by *official*. If token *official* is at the front of the buffer, GEN(have-org-role-91) will be applied.

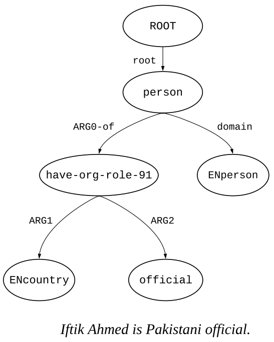

Figure 4: An example sentence with its compact AMR graph.

4 Stack LSTMs

A classifier is required to decide which action to take at each time step, given the current state.

4.1 Stack LSTMs for AMR parsing

Stack LSTMs (Dyer et al., 2015) are LSTMs (Hochreiter and Schmidhuber, 1997) that allow stack operations. Using stack LSTMs, each state can be represented using the contents of the stack, buffer and a list with the history of actions.

Let $\mathbf{s}_t$, $\mathbf{b}_t$ and $\mathbf{a}_t$ denote the summaries of stack, buffer and the history of actions at time step t respectively. The parser state $\mathbf{y}_t$ is given by:

$$\mathbf{y}_t = \max\{\mathbf{0}, \mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}\} \quad (1)$$

Action	Stack	Buffer	Concept/Relation
		Iftik, Ahmed, is, Pakistani, official, $	-
SHIFT	*Iftik*	*Ahmed, is, Pakistani, official, $*	-
MERGE		*(Iftik, Ahmed), is, Pakistani, official, $*	-
ENTITY (ENperson)		ENperson, *is, Pakistani, official, $*	ENperson
SHIFT	ENperson	*is, Pakistani, official, $*	-
SHIFT	ENperson, *is*	*Pakistani, official, $*	-
ENTITY (ENcountry)	ENperson, *is*	ENcountry, *official, $*	ENcountry
REDUCE	ENperson	ENcountry, *official, $*	-
SHIFT	ENperson, ENcountry	*official, $*	-
GEN (person)	ENperson, ENcountry	*official*, person, *$*	person
GEN (have-org-role-91)	ENperson, ENcountry	*official*, have-org-role-91, person, *$*	have-org-role-91
PRED (official)	ENperson, ENcountry	official, have-org-role-91, person, *$*	official
SHIFT	ENperson, ENcountry, *official*	have-org-role-91, person, *$*	-
LEFTLABEL (ARG2)	ENperson, ENcountry, *official*	have-org-role-91, person, *$*	have-org-role-91$\xrightarrow{\text{ARG2}}$official
REDUCE	ENperson, ENcountry	have-org-role-91, person, *$*	-
LEFTLABEL (ARG1)	ENperson, ENcountry	have-org-role-91, person, *$*	have-org-role-91$\xrightarrow{\text{ARG1}}$ENcountry
REDUCE	ENperson	have-org-role-91, person, *$*	-
SHIFT	ENperson, have-org-role-91	person, *$*	-
LEFTLABEL (ARG0-of)	ENperson, have-org-role-91	person, *$*	person$\xrightarrow{\text{ARG0−of}}$have-org-role-91
SWAP	have-org-role-91, ENperson	person, *$*	-
LEFTLABEL (domain)	have-org-role-91, ENperson	person, *$*	person$\xrightarrow{\text{domain}}$ENperson
REDUCE	have-org-role-91	person, *$*	-
SHIFT	have-org-role-91, person	*$*	-
PRED (ROOT)	have-org-role-91, person	ROOT	ROOT
LEFTLABEL (root)	have-org-role-91, person	ROOT	ROOT$\xrightarrow{\text{root}}$person
REDUCE	have-org-role-91	ROOT	-
REDUCE		ROOT	-
SHIFT	ROOT		-

Table 2: Example action sequence for parsing the sentence: *Iftik Ahmed is Pakistani official.* Here *$* is the end-of-sentence symbol.

where $\mathbf{W}$ is a learned parameter matrix, and $\mathbf{d}$ is a bias term. To learn a better representation, we apply the attention mechanism (Bahdanau et al., 2014) to the words the buffer contains. After getting the representation, we can use it to compute the probability of the next action:

$$p(z|\mathbf{y}_t) = \frac{\exp(g_{z_t}^{\top}\mathbf{y}_t + q_{z_t})}{\sum_{z' \in \mathcal{A}} \exp(g_{z'}^{\top}\mathbf{y}_t + q_{z'})} \quad (2)$$

where g_z is a vector representing the embedding of the action z and q_z is a bias term. The set $\mathcal{A}$ represents the valid action set in each time step. This set varies for different parsing states due to the constraints we made for the compact AMR graph.

4.2 Representations and UNK strategy

Following (Dyer et al., 2015), we concatenate three vector representations: pretrained word vector (Ling et al., 2015), learned word vector and learned vector for the POS tag, followed by a linear map to get the representation of each input token. For relation label representations and generated concepts, we simply use the learned embedding of the parser action that was applied to construct the relation.

Dyer et al. (2015) show that these representations can deal flexibly with out-of-vocabulary (OOV) words[7]. We extend this UNK strategy to AMR parsing. Apart from stochastically replacing (with $p = 0.2$) each singleton in the training data, we also replace the original PRED(n) as PRED(*UNK*) if the token should be selected as a concept. At the postprocessing stage, if an AMR concept is generated by PRED(*UNK*) we will use its lemma form for nouns or the most frequent sense for verbs to replace the "UNK" placeholder. This strategy makes the classifier trained not only to predict whether an OOV word should be selected as an AMR concept but also to predict the correct concept for the in-vocabulary word.

4.3 Composition Functions

We use recursive neural networks (Socher et al., 2013) to compute the representations of partially constructed structures.

For relation attachments, a composition function g_r is used to combine representations of AMR parent concept ($\mathbf{h}$), child concept ($\mathbf{d}$) and the corresponding relation label ($\mathbf{r}$) as:

$$g_r(\mathbf{h}, \mathbf{d}, \mathbf{r}) = \tanh(\mathbf{U}_r[\mathbf{h}; \mathbf{d}; \mathbf{r}] + \mathbf{b}_r) \quad (3)$$

where $\mathbf{U}_r$ and $\mathbf{b}_r$ are parameters in the model, as $\mathbf{U}_m$, $\mathbf{b}_m$, $\mathbf{U}_\ell$ and $\mathbf{b}_\ell$ in equation (4) and (5).

For generated non-lexical concepts, if terminal concepts are lexical concepts as shown in Figure 4 we will use a composition function g_m to get their ($\mathbf{c}_1$ and $\mathbf{c}_2$) merged representation:

$$g_m(\mathbf{c}_1, \mathbf{c}_2) = \tanh(\mathbf{U}_m[\mathbf{c}_1; \mathbf{c}_2] + \mathbf{b}_m) \quad (4)$$

---

[7] Both OOV words in the parsing data and OOV words in the pretraining language model can be represented.

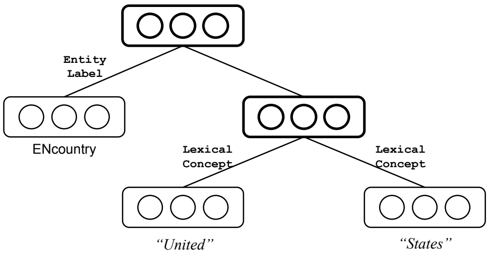

Figure 5: Example of a partial structure rooted by a non-lexical concept. *"United"* and *"States"* are merged by g_m. Then g_ℓ is applied to combine the merged representation and the label representation of ENcompany to arrive at the final representation.

Next, another composition function g_ℓ is applied to get the representation of the labeled generated concept ($\mathbf{e}$ and its entity label $\mathbf{l}$):

$$g_\ell(\mathbf{e}, \mathbf{l}) = \tanh(\mathbf{U}_\ell[\mathbf{e}; \mathbf{l}] + \mathbf{b}_\ell) \qquad (5)$$

5 Experiments and Results

We first evaluate the effectiveness of our hybrid aligner. Then we report the final results by incorporating it into our parser. We conduct experiments on the datasets LDC2014T12, LDC2015E86 and LDC2017T10. The newswire section of LDC2014T12 is popular so we also report the performance on it. All AMR parsing results are evaluated by using Smatch (Cai and Knight, 2013).

5.1 Aligner Evaluation

As the hybrid aligner is designed to tackle non-projectivity caused by alignment error and null alignment, our experiments focus on these aspects.

Non-Projectivity Issue: From Table 3, we find that the percentage of graphs that our hybrid aligner cannot recover is significantly less on these datasets when compared to the use of the JAMR and ISI aligners. We find that the percentage drops less on LDC2017T10. The reason behind this is that many AMR concepts do not have alignments (11.1%) if we use JAMR aligner. The probability of crossing arcs drops owing to the decrease of identified concepts. The ISI aligner shares the same issue. It yields the lowest crossing arcs rate since lots of concepts are not aligned.

Empty Alignment Issue: On the other hand, our hybrid aligner yields less empty alignments. This potentially helps our parser to achieve better performance, since it predicts lexical concepts solely relying on the look-up table generated by the aligner. If we do not have alignment between a certain token and its concept on the training set, the parser is less likely to predict that token as a concept during the testing phase.

Dataset	Aligner	F_1(%)	CR(%)	Null(%)
2014N	JAMR	92.70	14.4	5.1
	Hybrid	**96.12**	**8.8**	**2.0**
2014	JAMR	89.99	16.9	7.0
	Hybrid	**95.27**	**10.5**	**2.6**
2015	JAMR	86.23	16.5	8.9
	Hybrid	**93.05**	11.9	**2.9**
	ISI	71.92	**3.4**	20.1
2017	JAMR	84.41	13.6	11.1
	Hybrid	**92.69**	10.2	**3.4**
	ISI	75.31	**3.2**	18.2

Table 3: Alignment results evaluation. F_1 indicates the final score obtained by the action sequence generated by the aligner during training process. CR(%) indicates the portion that the parser cannot handle because of non-projectivity. Null(%) indicates the percentage of AMR concepts that are not aligned. As the ISI alignments did not include in the LDC2014T12, we can not compare on this dataset.

Improvements on these two aspects allow the aligner to yield better action sequences for the purposes of training our parser. From Table 3, we can see that if we follow the action sequence generated by the hybrid aligner, our parser will achieve a higher Smatch score consistently. Improvement gain gets larger as the corpus gets larger.

5.2 Parser Evaluation

Then we evaluate our parser on the same datasets. As illustrated in Table 4, most models incorporate additional features such as dependency trees, named entities, non-lexical role labels and external corpora. As stack LSTMs use POS tags to get the representation of the input, our parser does not use external resources other than POS tags obtained by using NLTK (Loper and Bird, 2002). We also try to evaluate our parser by removing the POS tags. The performance drops around 1.2 F_1 points on average. POS tags help the parser to select the correct sense when PRED action is applied.

Comparison with other parsers: Currently, all state-of-the-art models either have a relatively high complexity or adopt a pipeline approach. Wang and Xue (2017) incorporate a module called Factor Concept Labels consisting of Bi-LSTMs and CNN based on CAMR (Wang et al., 2016). Another popular parser is JAMR (Flanigan et al., 2016). The relation identification stage of JAMR has the complexity of $O(|V|^2 \log |V|)$, where $|V|$ is the number of concepts. RIGA (Barzdins and Gosko, 2016) is an ensemble system that combines CAMR and seq2seq model. Johnson et al. (2018) view AMR graph as the structure *AM alge-*

Parser	Type	Features					Pipeline	$F_1(\%)$			
		POS	**DEP**	**NER**	**SRL**	**Other**		**2014N**	**2014**	**2015**	**2017**
Flanigan et al. (2014)	Graph	√	√	×	×	No	Yes	59	58	-	-
Flanigan et al. (2016)	Graph	√	√	√	×	No	Yes	-	66	67	-
Werling et al. (2015)	Graph	√	√	√	×	No	Yes	62	-	-	-
Artzi et al. (2015)	Others	√	×	×	×	No	Yes	67	-	-	-
Pust et al. (2015)	Others	×	×	√	×	Word	Yes	-	67.1	-	-
Zhou et al. (2016)	Transition	√	√	√	√	No	No	71	66	-	-
Damonte et al. (2016)	Transition	√	√	√	√	No	No	-	64	64	-
Goodman et al. (2016)	Tree	√	×	√	×	No	Yes	70	-	64	-
Barzdins and Gosko (2016)	Tree	√	√	√	×	No	Yes	-	-	67.2	-
Wang et al. (2015)	Tree	√	√	√	×	No	Yes	70	66.5	-	-
Wang et al. (2016)	Tree	√	√	√	√	No	Yes	-	66.5	67.3	-
Wang and Xue (2017)	Tree	√	√	√	√	No	Yes	-	68.1	68.1	-
Peng et al. (2017)	Seq2seq	×	×	×	×	No	No	-	-	52	-
Konstas et al. (2017)	Seq2seq	×	×	√	×	Giga	No	-	-	62.1	-
Ballesteros and Al-Onaizan (2017)	Transition	√	√	×	×	No	No	69	64	-	-
Foland and Martin (2017)	Others	×	×	√	×	No	Yes	-	-	70.7±0.2	-
Buys and Blunsom (2017)	Seq2seq	√	×	√	×	No	No	-	-	-	61.9
van Noord and Bos (2017)	Seq2seq	√	×	×	×	Silver	No	-	-	68.5	71.0
Peng et al. (2018)	Transition	√	√	×	×	No	No	-	-	64	-
Vilares and Gómez-Rodríguez (2018)	Transition	√	√	√	×	No	No	-	-	64	-
Lyu and Titov (2018)	Others	√	×	√	×	No	Yes	-	-	**73.7±0.2**	**74.4±0.2**
Johnson et al. (2018)	Tree	√	×	√	×	No	No	-	-	70.2±0.3	71.0±0.5
This work (full model)	Transition	√	×	×	×	No	No	**74.0±0.5**	**68.3±0.4**	68.7±0.3	69.8±0.3
no compact AMR graph	Transition	√	×	×	×	No	No	72.1	66.7	67.2	68.9
JAMR aligner	Transition	√	×	×	×	No	No	72.6	65.4	65.8	66.3
no compact AMR graph, *JAMR* aligner	Transition	√	×	×	×	No	No	70.3	63.9	64.6	65.3
no UNK strategy	Transition	√	×	×	×	No	No	72.2	66.4	67.2	68.8
no POS	Transition	×	×	×	×	No	No	72.4	66.9	67.3	68.8

Table 4: Comparison with other parsers on LDC2014T12, LDC2015E86 and LDC2017T10 datasets. 2014N refers to the newswire section of LDC2014T12. We categorize parsers based on their types as discussed in Section 2. We also list down the features used by each system. POS: POS tags; DEP: dependency trees; NER: named entities; SRL: semantic role labels. Other features include: Word (WordNet for concept identification), GIGA (20M unlabeled Gigaword), and Silver (100k additional training pairs created by using CAMR and JAMR). "JAMR aligner" indicates that the parser is trained by action sequence generated by the JAMR aligner. "no UNK strategy" shows results without our UNK strategy. "no compact AMR graph" shows results without constraints on the compact AMR graph. Following several previous work, we also report standard deviation for the full model.

bra defined in (Groschwitz et al., 2017). Since *AM algebra* can be viewed as dependency trees over the sentence, they can train a dependency parser to map the sentence into this structure. Different from the structure used in CAMR (Wang et al., 2015), this structure can be directly transformed to AMR graph by using postprocessing rather than relying on another transition-based system. The complexity of their projective decoder is $O(n^5)$, where n is the length of the sentence. Zhou et al. (2016) is also a transition-based parser, which adopts and improves beam search strategy. In contrast, our system does not use a beam, but we anticipate improved results when a beam is used.

For seq2seq models, they generally require less features and build the AMR graph in an end-to-end way. However, these models usually suffer from data sparsity issue (Peng et al., 2017). In order to address this issue, these models utilize external corpora. Konstas et al. (2017) achieves 62.1 score by using 20M unlabeled Gigaword sentences for paired training. van Noord and Bos (2017) use an additional training corpus of 100k sentences called Silver generated by an ensemble system consisting of JAMR and CAMR parsers. Without training on this additional dataset, the performance of their model is 64.0 on LDC2015E86.

Our end-to-end parser has *linear* time complexity and it exhibits competitive results on the datasets with only POS tags as additional features. Foland and Martin (2017) only use named entities as an external resource and they report the second highest F_1 on LDC2015E86. Their system also adopts a pipeline approach. The concept identification phase requires 5 different LSTMs to discover different kinds of concepts based on carefully designed features. Then they connect these components into a single graph. Unlike previous work, the very recent work by Lyu and Titov (2018) treat the alignments as latent variables in a joint probabilistic model, which improves the parsing performance substantially. They report the highest scores on LDC2015E86 and LDC2017T10 datasets. Their parser requires 5 different BiLSTMs for concept identification, alignment prediction, relation identification and root identification. Though our parser constructs the AMR graph in an end-to-end fashion, it can achieve 68.7 and 69.8 Smatch score respectively on the same test set with a simple architecture.

Metric	**LDC2015E86**						**LDC2017T10**			
	W'15	**F'16**	**D'16**	**V'18**	**J'18**	**Ours**	**vN'17**	**L'18**	**J'18**	**Ours**
Smatch	67	67	64	64	**70**	69	71	**74**	71	70
Unlabeled	69	69	69	68	**73**	72	74	**77**	74	73
No WSD	64	68	65	65	**71**	69	72	**76**	72	71
Reentrancies	41	42	41	44	**46**	**46**	**52**	**52**	49	49
Concepts	80	**83**	**83**	**83**	**83**	**83**	82	**86**	**86**	84
Named Ent.	75	79	**83**	**83**	79	80	79	**86**	78	80
Wikification	0	**75**	64	70	71	68	65	**76**	71	70
Negations	18	45	48	47	**52**	49	**62**	58	57	48
SRL	60	60	56	57	**63**	61	66	**70**	64	63

Table 5: Detailed results for the LDC2015E86 and LDC2017T10 test set. W'15 is Wang et al. (2015)'s parser. F'16 is Flanigan et al. (2016)'s parser, D'16 is Damonte et al. (2016)'s parser. V'18 is Vilares and Gómez-Rodríguez (2018)'s parser. J'18 is Johnson et al. (2018)'s parser. vN'17 is van Noord and Bos (2017)'s parser with 100K additional training pairs. L'18 is Lyu and Titov (2018)'s parser.

In order to investigate how our parser performs on predicting *Named Entities*, *Reentrancies*, *Concepts*, *Negations*, etc, we also use the fine-grained evaluation tool (Damonte et al., 2016) and compare to systems which reported these scores. The results are shown in Table 5. We obtain relatively high results for *Concepts*, *Named Entities* and *Wikification*. Also, we achieve good performance on reentrancy identification though we remove many reentrant edges during training. This indicates that the compact AMR graph encodes necessary information. On the other hand, our model does not perform so well on predicating *Negations*. We believe one reason is that our parser is a word-level model. It is hard for it to capture morphological features such as prefixes "un", "in", "il", etc. We anticipate better performance when the character-based representations are used.

Does the compact AMR graph help: In order to better investigate the effect of different modules used in the parser, we also evaluate our parser by removing certain modules. Because our transition system is built based on the compact AMR graph, we could not evaluate the parser by isolating this representation completely. Therefore, we choose to remove some constraints defined on the compact AMR graph to investigate its effect. We can see that our representation improves the performance of our parser on three datasets, which indicates that a refined search space is beneficial to our system. When our parser is trained on the action sequence generated by the JAMR aligner as previous models, it still achieves a competitive score especially on the newswire section.

Impact of the hybrid aligner: Experiments also illustrate that the hybrid aligner consistently helps our parser. When we train our parser by using the action sequence generated by the JAMR aligner, we still achieve competitive results on the newswire section. However, the performance drops as the corpus gets larger. The largest drop occurs when we apply the parser on LDC2017T10, on which correct senses of predicates cannot be selected for many unseen words. We hypothesize that it is because the quality of alignment degrades as the corpus gets larger, which prevents the parser from learning how to find a good path in the search space during training. The hybrid aligner can alleviate this issue by generating a look-up table that has broader coverage. After incorporating the hybrid aligner, our parser achieves the best results of 68.3 on the full test set of LDC2014T12.

6 Conclusion and Future Work

We present a novel transition-based system which refines the search space. Experiments show that our parser is able to achieve state-of-the-art performance with a simple architecture and minimal additional resources. We believe our end-to-end system is helpful in practical settings. In the future, we would also like to investigate if it is possible for alignments to be treated as latent variables, which can be learned in a joint manner within the current framework.

Acknowledgments

We would like to thank the anonymous reviewers for their thoughtful comments. We would also like to thank Zhiyang Teng, Zhanming Jie, Bailin Wang, Aldrian Obaja Muis, Miguel Ballesteros, Swabha Swayamdipta, Bill Yuchen Lin and Yujian Wu for their helpful suggestions. This work is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proc. of EMNLP*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using Stack-LSTMs. In *Proc. of EMNLP*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for SemBanking. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Guntis Barzdins and Didzis Gosko. 2016. Riga at SemEval-2016 task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proc. of NAACL-HLT*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proc. of ACL*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proc. of ACL*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proc. of ACL*.

Marco Damonte, Shay B Cohen, and Giorgio Satta. 2016. An incremental parser for abstract meaning representation. In *Proc. of EACL*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proc. of SemEval*.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of ACL*.

William Foland and James H Martin. 2017. Abstract meaning representation parsing using LSTM recurrent neural networks. In *Proc. of ACL*.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. UCL+Sheffield at SemEval-2016 task 8: Imitation learning for AMR parsing with an alpha-bound. In *Proc. of SemEval*.

Jonas Groschwitz, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2017. A constrained graph algebra for semantic parsing with AMRs. In *Proc. of IWCS*.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational linguistics*, 39(4):949–998.

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

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proc. of ACL*.

Mark Johnson, Alexander Koller, Jonas Groschwitz, Meaghan Fowlie, and Matthias Lindemann. 2018. AMR dependency parsing with a typed semantic algebra. In *Proc. of ACL*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence models for parsing and generation. In *Proc. of ACL*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL-HLT*.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of Word2Vec for syntax problems. In *Proc. of NAACL-HLT*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *CoRR*, cs.CL/0205028.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proc. of EMNLP*.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proc. of ACL*.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT*.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proc. of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proc. of AFNLP*.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. *Proc. of ACL*.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *arXiv preprint arXiv:1705.09980*.

Xiaochang Peng, Daniel Gildea, and Giorgio Satta. 2018. AMR parsing with cache transition systems. In *Proc. of AAAI*.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proc. of CoNLL*.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proc. of EACL*.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *Proc. of EMNLP*.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proc. of EMNLP*.

Mrinmaya Sachan and Eric Xing. 2016. Machine comprehension using rich semantic representations. In *Proc. of ACL*.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proc. of EMNLP*.

Richard Socher, John Bauer, Christopher D Manning, et al. 2013. Parsing with compositional vector grammars. In *Proc. of ACL*.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. AMR-to-text generation as a traveling salesman problem. In *Proc. of EMNLP*.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and explicit models of grammar*, pages 181–224.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Greedy, joint syntactic-semantic parsing with Stack LSTMs. In *Proc. of SemEval*.

David Vilares and Carlos Gómez-Rodríguez. 2018. A transition-based algorithm for unrestricted AMR parsing. In *Proc. of NAACL-HLT*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. of the 16th conference on Computational linguistics-Volume 2*.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at SemEval-2016 task 8: An extended transition-based AMR parser. In *Proc. of SemEval*.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of AMR parsing. In *Proc. of EMNLP*.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *Proc. of NAACL-HLT*.

Keenon Werling, Gabor Angeli, and Christopher Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proc. of ACL*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proc. of EMNLP*.

Heuristically Informed Unsupervised Idiom Usage Recognition

Changsheng Liu and **Rebecca Hwa**
Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15260, USA
{changsheng,hwa}@cs.pitt.edu

Abstract

Depending on the surrounding context, an idiomatic expression may be interpreted figuratively or literally. This paper proposes an unsupervised learning method for recognizing the intended usages of idioms. We treat the possible usages as a latent variable in probabilistic models and train them in a linguistically motivated feature space. Crucially, we show that distributional semantics serves as a helpful heuristic for formulating a *literal usage metric* to estimate the likelihood that the idiom is intended literally. This information can then guide the unsupervised training process for the probabilistic models. Experiments show that our overall model performs competitively against supervised methods.

1 Introduction

Many idiomatic expressions may be interpreted both figuratively or literally. Their intended usages depend on how they fit with their contexts. For example, the idiom ”spill the beans” is used figuratively in the first instance below, and literally in the second:

> *(1)* [fig.] *The* ***beans have been spilled****. From what I've read on Twitter I could probably fill out the forms and submit it to the FISA court. I don't know what the big secret is.*[1]
>
> *(2)* [lit.] ***Spill the beans****, flip the fruit, bust open a box of hot pockets. Make a general mess of the kitchen.*[2]

This type of ambiguity is commonplace – prior work suggests that about half out of a sample of 60 idioms have a clear literal meaning as well as a figurative one (Fazly et al., 2009). Being able to distinguish the intended usage of an idiom in context has been shown to benefit many natural language processing (NLP) applications, e.g., machine translation and sentiment analysis (Salton et al., 2014; Williams et al., 2015).

While supervised models for idiom usage recognition have had some successes, they require appropriately annotated training examples (Peng et al., 2014; Byrne et al., 2013; Liu and Hwa, 2017). A more challenging problem is to recognize idiom usages without a dictionary or some annotated examples (Korkontzelos et al., 2013). Some previous unsupervised models tried to exploit linguistic differences in usages. For example, Fazly et al.(2009) observed that an idiom appearing in its canonical form is usually used figuratively; Sporleder and Li(2009) relied on the break in lexical coherence between the idioms and the context to signal a figurative usage. These heuristics, however, are not always applicable because the distinctions they depend upon may not be present or obvious. To improve generalization across different idioms and usage contexts, we need a more reliable heuristic, and appropriately incorporate it into an unsupervised learning framework.

We propose a heuristic that differentiates usages based on distributional semantics (Harris, 1954; Turney and Pantel, 2010). Our key insight is that when an idiom is used literally, its relationship with its context is more predictable than when it is used figuratively. This is because the literal meaning of an idiom is compositional (Katz and Giesbrecht, 2006), and the constituent words that make up the idiom are also meant literally. For example, in instance (2), *spill* is meant literally and can take on objects other than *beans*; moreover, one of the context words, *mess*, can often be seen to co-

---

[1] `https://twitter.com/BTeboe/status/958792419302100993`

[2] `https://twitter.com/DukeRaccoon/status/477530732173471744`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1723–1731
Brussels, Belgium, October 31 - November 4, 2018.

occur with *spill* in other text, even without *beans*. Our strategy is to represent an idiom's literal usage in terms of the word embeddings of the idiom's constituent words and other words they frequently co-occur with. Then, for any instance in which the idiom's usage is not known, we only need to determine the semantic similarity between that instance and the idiom's literal representation. We define a *literal usage metric* that estimates the likelihood that an instance would be labeled ”literal”.

While the literal usage metric captures the distributional semantic information of the context, we find that some other linguistic cues are also significant for usage detection (such as whether the subject of the sentence is a person); therefore, we allow our model to further refine through unsupervised methods. Specifically, we treat the usage (*figurative* or *literal*) as a hidden variable in probabilistic latent variable models, and we define a set of features that are linguistically relevant for idiom usage detection as observables. We integrate our literal usage metric with the latent variable models by treating the metric outputs as *soft labels* to guide the latent variable models toward grouping by usages.

We hypothesize that unsupervised learning in a more linguistically motivated feature space, informed by soft labels from a semantically driven metric, will produce more robust classifiers. We conduct experiments comparing our approach against other supervised and unsupervised baselines. Results suggest that our approach achieves performances that are competitive to supervised models.

2 Related Work

Despite the common perception that idioms are mainly used figuratively, many can also be meant literally. A number of models have been proposed in the literature to recognize an idiom's usages under different context. Many rely on specific linguistic property to draw a clear-cut decision boundary between literal and figurative usages. For example, Fazly et al. (2009) proposed a method that relies on the concept of *canonical form*. Based on the observation that while literal usages are less syntactically restricted, figurative usages tend to occur in a small number of canonical form(s). As shown in the examples above, however, this rule of thumb does not always hold. Sporleder and Li (2009) proposed a method by building a cohesion graph to include all content words in the context; if removing the idiom improves cohesion, they assume the instance is figurative. Later, Li and Sporleder (2009) used their cohesion graph method to label a subset of the test data with high confidence. This subset is then passed on as training data to the supervised classifier, which then labels the remainder of the dataset.

When manually annotated examples are available, supervised classifiers are effective. Rajani et al. (2014) extracted all non-stop-words in the context and used them as ”bag of words” features to train a L2 regularized Logistic Regression (L2LR) classifier (Fan et al., 2008). As local context of an idiom holds clues for discriminating between its literal and figurative usages, Liu and Hwa (2017) find that context representation also plays a significant role in idiom usage recognition. They took an adaptive approach, applying supervised ensemble learning over three classifiers based on different context representations (Peng et al., 2014; Birke and Sarkar, 2006; Rajani et al., 2014).

3 Our Approach

Given a target idiomatic expression and a collection of instances in which the idiom occurs, our proposed system (Figure 1) determines whether the idiom in each instance is meant figuratively or literally. We first build a **Literal Usage Representation** for each idiom by leveraging the distributional semantics of its constituents (Sec 3.1). Given an instance of idiom, we can determine its usage by the semantic similarity between the context of the instance and the **Literal Usage Representation**. We define a **Literal Usage Metric** to transform the semantic similarity score into soft label, i.e., an initial rough estimation of the instance's usage (Sec 3.2). Finally, we treat the soft labels as distant supervision for downstream probabilistic latent variable models, in which the usages are considered as the hidden variables and are represented over a set of features.

3.1 Literal Usage Representation

An idiom co-occurs with different sets of words depending on whether it is meant literally or figuratively. For example, when used literally, *get wind* is more likely to co-occur with words such as *rain*, *storm* or *weather*; in contrast, when used figuratively, it frequently co-occurs with *rumor* or

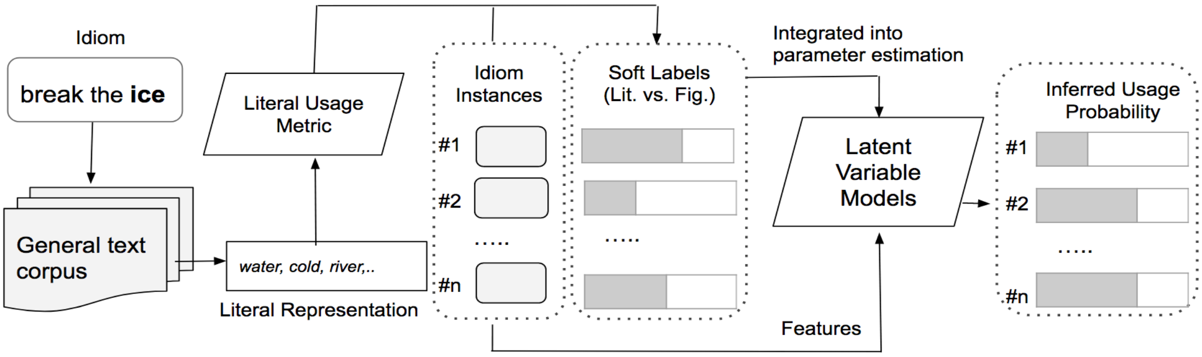

Figure 1: An overview of our unsupervised idiom usage recognition model

story, etc. Comparing the two sets of words associated with the idiom, we see that the literal set of words also tend to co-occur with just *wind*, a constituent word within the idiom. Therefore, even without annotated data or dictionary, we may still approximate a representation for the literal meaning of an idiom by the idiom's constituent words and their semantic relationship to other words. To do so, we begin by initializing a *literal meaning set* to just the idiom's main constituent words[3]; we then grow the set by adding two types of semantically related words. First, we look for co-occuring words in a large textual corpus (e.g., (David et al., 2005)): for each constituent word w, we randomly sample s sentences that contain w from the corpus; we extract the top n most frequent words (excluding stop words) and add them to the literal meaning set. Second, we look for words that are semantically close in a word embedding space: we train a continuous bag-of-words (CBOW) embedding model (Mikolov et al., 2013) and add additional t words that are the most related to w using cosine similarity.

All together, the literal usage representation is a collection of vectors, i.e., the embeddings of the words in the final extended literal meaning set. The size of the set depends on parameters s, n, and t; if the chosen values are too small, we do not end up with a word collection that is representative enough; if the numbers are too large, we would only be wasting computing resources chasing Zipfian tails. Parameter setting choices are discussed further in the experiment section.

3.2 Literal Usage Metrics

Among all the instances to be classified, we expect the context words of the literal cases to be more semantically close to the literal usage representation we just formed. Let L denote the set of words in the literal usage representation for the target idiom. For each instance, let C be the set of non-stop context words in the instance. We calculate s, the semantic similarity score between the context of the instance and the literal usage representation as follows:

$$s = \frac{1}{|C|} \sum_{c \in C} \frac{1}{|L|} \sum_{l \in L} sim(c, l) \qquad (1)$$

where c denotes a word in C, l denotes a word in L and $sim(c, l)$ refers to the cosine similarity between the word embeddings of c and l.

Let $S = \{s_1, s_2, ...s_n\}$ be the set of semantic similarity scores for all the instances we wish to classify. Instances with higher scores are more likely to use the idiom literally. A naive literal usage metrics is to choose a predefined threshold for all idioms and label all the instances with score above the threshold as literal usages. This approach is unlikely to work well in practice. As noted by previous work, idioms have different levels of **semantic analyzability** (Gibbs et al., 1989; Cacciari and Levorato, 1998). When an idiom has a high degree of semantic analyzability, its contextual words will be more semantically close to the literal usage representation, thus a higher threshold is needed.

In this work, we select a different decision threshold for each idiom adaptively based on the similarity scores distribution. And most importantly, rather than generate a hard label, we transform these scores into a probabilistic metric,

---

[3] We observe that the nouns tend to be the most indicative of the idiom's literal meaning, but if the idiom does not contain any noun, we back off to any constituent word that is not a stop word.

where 0 means the usage in the instance is almost certainly figurative while 1.0 means it is literal.

We propose a metric based on the principle of **Minimum Variance (MinV)**. That is, we first sort the scores in S and choose the threshold (from these scores) that minimizes the sum of variances of the two resulting clusters. For each instance i, we then apply the following metric to estimate the probability that the idiom in instance i is meant **literally** based on its semantic similarity score s_i :

$$Pr_i = \frac{1}{1 + e^{-k*(s_i - t)}} \tag{2}$$

where k is a constant weighting factor and t indicates the learned threshold. The intuition is that the larger the difference between s_i and the threshold is, the more likely the instance i is literal; the probability of literal usage is not linearly correlated to the difference, we use the sigmoid function to account for this non-linearity. We incorporate *k* to scale the value of the difference since it is generally very small (close to 0). Without *k*, all the Pr values gravitate toward 0.5, rendering the soft label being equivalent to random guess. We set k to 5 for all the idioms based on a development set.

3.3 Heuristically Informed Usage Recognition

The soft label, generated by MinV (the literal usage metric), captures the distributional semantic information of the context. In practice, there are a variety of other linguistic features which are also informative of the intended usage of idiom. We explore probabilistic latent variable models over a collection of features that are linguistically relevant for idiom usage detection. The soft label is integrated into the unsupervised learning of hidden usages as a distant supervision. In this section, we will describe the proposed features in the latent variable models and how we integrate the soft label into the learning process.

3.3.1 Latent Variable Models

To predict an idiom's usage in instances, we consider two representative probabilistic latent variable models: Latent Dirichlet Allocation (LDA) (Blei et al., 2003)[4] and unsupervised Naive Bayes (NB). For both models, the latent variable is the idiom usage (figurative vs. literal); the observables are linguistic features that can be extracted from the instances, described below:

Subordinate Clause We encode a binary feature indicating whether the target expression is followed by a subordinate clause (the Stanford Parser (Chen and Manning, 2014) is used). This feature is useful for some idioms such as *in the dark*. It usually suggests a figurative usage as in *You've kept us totally in the dark about what happened that night*.

Selectional Preference Violation of selectional preference is normally a signal of figurative usage (e.g., having an abstract entity as the subject of *play with fire*). We encode this feature if the head word of the idiom is a verb and focus on the subject of the verb. We apply Stanford Name Entity tagger (Finkel et al., 2005) with 3 classes ("Location", "Person", "Organization") on the sentence containing the idiom. If the subject is labeled as an Entity, its class will be encoded in the feature vector. Pronouns such as "I" and "he" also indicate the subject is a "Person". However, they are normally not tagged by Stanford Name Entity tagger. To overcome this issue, we add Part-of-Speech of the subject into the feature vector.

Abstractness Abstract words refer to things which are hard to perceive directly with our senses. Abstractness has been shown to be useful in the detection of metaphor, another type of figurative language (Turney et al., 2011). A figurative usage of an idiomatic phrase may have relatively more abstract contextual words. For example, in the sentence *She has lived life in the fast lane*, the word *life* is considered as an abstract word. This is a useful indicator that *in the fast lane* is used figuratively. We use the MRC Psycholinguistic Database Machine Usable Dictionary (Coltheart, 1981) which contains a list of 4295 words with their abstractness measure between 100 and 700. We calculate the average abstractness score for all the contextual words (with stop words being removed) in the sentence containing the idiom. The score is then transformed into categorical feature to overcome sparsity problem based on the following criteria: concrete (450 - 700), medium (350 - 450), abstract (100 - 350).

Neighboring Words Words preceding and following the idiomatic expression can be very informative in terms of usage recognition. For example, words such as *relax* or *shower* before the idiom *in hot water* often signal a literal usage.

---

[4] Although originally conceived for modeling document content, LDA can be applied to any kind of discrete input

Part-of-Speech of the Neighboring Words Class of neighboring words might be useful as well. For example, a pronoun preceding *dog's age* generally indicates a literal usage, as in *I think my dog's age is starting to catch up. She sometimes needs help to jump on to my bed*, while a determiner usually marks a figurative usage, as in *It's been a dog's age since I've used Twitter.*

3.3.2 Incorporating Soft Label into Usage Recognition

Given a collection of instances and their features, either LDA or NB can separate the instances into two groups (hopefully, by usages), but it does not associate the right label (i.e., "figurative" or "literal") to the groups. We do not want to rely on any manual annotations for this step. Therefore, we integrate the automatically generated soft labels (based on MinV, our literal usage metric) into the unsupervised learning procedure as a weak form of supervision. Formally, we want to estimate each instance's posterior distribution over (literal/figurative) usages θ_{du} and usage-feature distribution ϕ_{uf}. For LDA, we derive a Gibbs sampling algorithm which incorporates the soft label into the learning procedure. We refer it as informed Gibbs sampling (infGibbs). For unsupervised naive Bayes model, we adapt the classical Expectation-Maximization algorithm to integrate the soft label. We refer it as informed Expectation-Maximization (infEM).

Informed Gibbs Sampling The Gibbs sampling algorithm (Griffiths and Steyvers, 2004) used in traditional LDA initializes each word token a random hidden topic. The system needs to interpret the learned topics post-hoc, e.g., by human annotation. In our case, for each feature f in each instance, an initial random usage **biased** by the instance's soft label is assigned to f (i.e., a Bernoulli trial). Since the soft label explicitly encodes an instance's literal and figurative usage distribution, we do not need to interpret the learned usages at the end of the algorithm. Based on these assignments, we build a feature-usage counting matrix C^{FU} and instance-usage counting matrix C^{DU} with dimensions $|F| \times 2$ and $|D| \times 2$ respectively ($|F|$ is the feature size and $|D|$ is the number of instances): $C^{FU}_{i,j}$ is the count of feature i assigned to usage j; $C^{DU}_{d,j}$ is the count of features assigned to usage j in instance d. Then for each feature f in each instance, we resample a new usage for f and matrices C^{FU} and C^{DU} will be updated accordingly. This step will be repeated for T times. The resampling equation is:

$$p(u_i = j | u_{-i}, f) \propto p_j \cdot \frac{C^{f_i}_{-i,j} + \beta}{C^{(*)}_{-i,j} + |F|\beta} \cdot \frac{C^{d_i}_{-i,j} + \alpha}{C^{d_i}_{-i,*} + |U|\alpha} \tag{3}$$

where i indexes features in the instance d, j is an index into literal and figurative usages, $*$ indicates a summation over that dimension and $-$ means excluding the corresponding instance. The first factor p_j is the soft label encoding prior usage distribution. The second factor represents the probability of feature f under usage j ($C^{f_i}_{-i,j}$ is the count of the feature f assigned to usage j, excluding the current usage assignment u_i). The third factor represents the probability of usage j in the current instance ($C^{d_i}_{-i,j}$ is the count of linguistic features which are assigned to usage j in the current instance, excluding the current feature f). The value of $|U|$ is 2, representing the number of usages (i.e., figurative and literal). α and β are the hyper-parameters from the Dirichlet priors (we set both of them to 1). The core idea of Equation 3 is to integrate both distribution semantic information (soft label, the first factor) and linguistically motivated features (the second and third factors) into the inference procedure.

The matrices of C^{FU} and C^{DU} from the last $10\% * T$ iterations are averaged and then normalized to approximate the true usage-feature distribution ϕ_{uf} and instance-usage distribution θ_{du} respectively. The final result is determined by θ_{du}, i.e., assigning each instance with the usage of probability higher than 0.5. We do average to have a more stable result because an accidental bad sampling would affect our model negatively if we only use the C^{FU} and C^{DU} from the last iteration. This procedure is important for some idioms if their feature space is sparse. The iteration number T is set to 500 based on a development set.

Informed Expectation Maximization Combining a Naive Bayes classifier with the EM algorithm has been widely used in text classification and word sense disambiguation (Hristea, 2013; Nigam et al., 2000). In our case, we want to construct a model to recover the missing literal and figurative labels of the instances of the target idiom. This section describes two extensions to the basic EM algorithm for idiom usage recognition. The extensions help improve parameter estimation by taking the automatically learned soft labels into consideration.

Our informed EM method extends a basic version for NB (Hristea, 2013), where the initial parameter values θ_{du} and ϕ_{uf} are chosen randomly. At each iteration, the E-step of the algorithm estimates the expectations of the missing values (i.e. the literal and figurative usage) given the latest iteration of the model parameters; the M-step maximizes the likelihood of the model parameters using the previously-computed expectations of the missing values. As we've done with extending Gibbs sampling for LDA, we also perform two similar adaptations on conventional EM for NB to incorporate soft labels. First, we assign each instance an initial usage distribution θ_{du} directly using the soft label, and then initialize the usage-feature distribution ϕ_{uf} using these assignments. We refer it as informed initialization. Second, in the E-step, we multiply the expectation result of the basic EM with the soft label as the new expected usage for each instance (i.e., updating θ_{du}). The M-step is the same as basic EM to update the usage-feature distribution ϕ_{uf}.

4 Evaluation

We conduct experiments to address three questions:

1. How effective is our overall approach? How does it compare against previous work?

2. How effective is our literal usage metric (i.e., MinV) compared to other heuristics?

3. How effective is our literal usage metric at informing downstream learning processes?

4.1 Experimental Setup

Models Our main experiments will evaluate the two variants of the proposed fully unsupervised model as described in section 3: MinV+infGibbs and MinV+infEM. We report the average performance of our models over 5 runs. Performing multiple runs is necessary because we have a sampling process. They are compared with three baseline unsupervised models: Sporleder and Li (2009), Li and Sporleder (2009)[5] and Fazly et al. (2009); and two baseline supervised models: Rajani et al. (2014) and Liu and Hwa (2017) (using 5-fold cross validation).

Parameter setting Recall that in order to build the literal usage representation of an idiom, we need to sample s sentences that contain each constituent word w from an external corpus; extract from them the top n most frequently co-occurring words with w; then separately find t words that are semantically similar to w using word embeddings. To set parameters with values in reasonable ranges, we evaluated MinV on a small development set. We picked 10 idioms that are different from the evaluation set, scraped 50 instances from the web for each idiom, and labeled them ourselves. We find that $s >= 100$, n=10, and t=5 yield good results.

We use the gensim toolkit (Řehůřek and Sojka, 2010) and train our word embedding model using the continuous bag of word model on Text8 Corpus[6]. Negative sampling is applied as the training method; the min_count is set to 2. For the other parameters, we use the default settings in gensim.

Evaluative Data Our goal is to compare all the methods under two public available corpora: SemEval 2013 Task 5B corpus (Korkontzelos et al., 2013), which is used by prior supervised methods (Liu and Hwa, 2017; Rajani et al., 2014) and verb–noun combination (VNC) dataset (Cook et al., 2008), which is used by a prior unsupervised method (Fazly et al., 2009). However, there are some methods-datasets conflicts that have to be resolved. Because the idioms in the SemEval dataset are all in their canonical forms, and because the idioms are not restricted to the verb-noun combination, we cannot evaluate the method by Fazly et al. on this dataset (as their method is tailored to verb-noun combination). Some idioms from the VNC dataset are almost always used figuratively (or literally), which presents a problem for supervised methods. To facilitate full comparisons, we select the subset of idioms from the VNC corpus whose number of literal and figurative instances are both higher than 10. A summary of the two corpora is shown in Table 1. Note that each instance in SemEval corpus has about 3∼5 sentences; for consistency, we use 3 sentences as the context: the sentence with the target idiom and two neighboring sentences.

Evaluation metric Following the convention in prior works, we report the F-score for the recognition of figurative usages and the overall accuracy.

---

[5] We replace Normalized Google Distance (NGD) with word embeddings to measure the semantic relatedness between words due to the query frequency restriction on the API of NGD.

[6] From `http://mattmahoney.net/dc/text8.zip`

	SemEval	VNC
# of Idiom	10	11
# of Literal	1185	239
# of Figurative	1186	470
Idiom Type	Mixture	Verb-Noun
Syntactic Form	Canonical	Mixture
Context Size	3 ∼ 5 Sentences	1 Sentence

Table 1: Statistics of the two corpora

4.2 The Performance of Our Full Models

Table 2 shows the result of our models and the other comparative methods. Our proposed models show consistent performance across the two corpora, outperforming the unsupervised baselines from Sporleder and Li (2009), Li and Sporleder (2009) and the supervised model from Rajani et al. (2014). Moreover, there is no statistical significance in the F-score difference between the supervised ensemble model from Liu and Hwa (2017) and our models.

On the VNC corpus, our models have comparable average scores as that of Fazly et al. (2009); our scores are more stable across different idioms. While the method of Fazly et al. is nearly perfect for some idioms (0.98 on ”take heart”), it performs poorly for others (e.g., 0.33 on ”pull leg”). Their algorithm has trouble with idioms whose canonical and non-canonical forms can appear frequently both in literal and figurative usages.

4.3 Effectiveness of MinV

The core of our approach is MinV, the literal usage metric we developed to generate soft labels to guide the unsupervised learning. This experiment examines its effectiveness by creating usage classifications directly from it (i.e., if MinV predicts a probability of >0.5, predict ”literal”). We compare MinV against two alternative heuristics.

MinV is based on two core ideas. First, if an idiom is used figuratively, we expect to see a big difference (low similarity scores) between its context and the semantic representation of idiom’s literal usage. The idea is similar to that of Sporleder and Li (2009), but they relied on lexical chain instead of distributional semantics. Second, instead of choosing a predefined threshold to separate the raw semantic similarity scores, we select a different decision threshold for each idiom adaptively based on the distribution of the scores. So as an alternative, we compare MinV against a Fixed-Threshold heuristic that labels an instance as ”literal” if its raw score is higher than some global threshold (set to 0.346 based on development data).

In Table 3, we observe that Minv outperforms both Sporleder and Li’s model as well as Fixed-Threshold, but using MinV by itself is not sufficient. It has great fluctuations, e.g., the F-Score for individual idioms varies from 0.43 to 0.88. Recall that MinV +infGibbs has a smaller fluctuation across different idioms in Table 2. These results suggest that the subsequent learning process is effective.

Through error analysis, we find two major factors contributing to the performance fluctuation. First, the context itself could be misleading. An error case of **play ball** by MinV is:

All 10-year-old Minnie Cruttwell wants to do is play with the boys , but the ***Football Association*** *are not playing ball. She is a* ***member*** *of a mixed* ***team*** *called Balham Blazers , but the FA say she must* ***join*** *a girls’* ***team*** *when she is 12.*

The context words in bold (which are related to the word ”ball”) mislead MinV to predict a ”literal” usage when it is actually a ”figurative” usage (since an organization such as the Football Association cannot literally *play ball*). Second, not all content words in the context are relevant for distinguishing the idiom’s usage. A future direction is to prune contextual words more intelligently.

4.4 Integration of MinV into Learning

We have argued that an advantage of using a metric with a probabilistic interpretation instead of a binary class heuristic is that its scores can be incorporated into subsequent learning models as soft labels. In this set of experiments, we evaluate the impact of the metric on the learning methods. First, we consider unsupervised learning without input from the literal usage metric. We cluster the instances with the original Gibbs sampling and EM algorithms and then label the two clusters with the majority usage within the clusters. Second, we explore using the information from the literal usage metric as ”noisy gold standard” to perform supervised training on a nearest neighbors (NN) classifier. Specifically, the literal and figurative instances labeled by MinV with high confidence (top 30%) are used as example set. Then for each test instance, we calculate its cosine similarity (in feature space) to the literal and figurative example sets and assign the label of the closest set. We refer this model as MinV +NN.

		SemEval		VNC	
Type	Model	Avg. F_{fig}	Avg.Acc	Avg. F_{fig}	Avg.Acc
Unsupervised	Sporleder & Li	0.58* (0.42 $\sim$ 0.72)	0.52*(0.32 $\sim$ 0.7)	0.61* (0.46 $\sim$ 0.73)	0.57*(0.41 $\sim$ 0.75)
	Li & Sporleder	0.64* (0.41 $\sim$ 0.76)	0.62*(0.43 $\sim$ 0.71)	0.67* (0.48 $\sim$ 0.77)	0.66*(0.52 $\sim$ 0.77)
	Fazly et al.	-	-	0.73 (0.33 $\sim$ 0.98)	0.74 (0.35 $\sim$ 0.98)
Supervised	Rajani et al.	0.71* (0.54 $\sim$ 0.83)	0.75(0.67 $\sim$ 0.81)	0.69* (0.49 $\sim$ 0.8)	0.7*(0.6 $\sim$ 0.79)
	Liu & Hwa	0.77 (0.68 $\sim$ 0.85)	0.77(0.71 $\sim$ 0.85)	0.75 (0.65 $\sim$ 0.88)	0.75(0.67 $\sim$ 0.89)
Our Model	**MinV + infGibbs**	0.75 (0.64 $\sim$ 0.91)	0.74(0.63 $\sim$ 0.87)	0.73 (0.64 $\sim$ 0.86)	0.75(0.66 $\sim$ 0.83)
	MinV + infEM	0.73 (0.58 $\sim$ 0.88)	0.73(0.61 $\sim$ 0.85)	0.72 (0.62 $\sim$ 0.87)	0.72(0.6 $\sim$ 0.84)

Table 2: The performances of different models. Avg. F_{fig} denotes average figurative F-score, Avg.Acc denotes average accuracy. We report the range in the parenthesis. * indicates the difference is significant with our MinV+ infGibbs model at the 95% confidence level. Since the method from Fazly et al. (2009) restricted their experiment to VNC type, we only report their performance on the VNC corpus.

Model	Avg. F_{fig}	Avg.Acc
Fixed-Threshold	0.6 (0.23 $\sim$ 0.82)	0.62 (0.47 $\sim$ 0.83)
MinV	0.66 (0.43 $\sim$ 0.88)	0.65 (0.51 $\sim$ 0.89)
Sporleder & Li	0.59 (0.42 $\sim$ 0.73)	0.54(0.32 $\sim$ 0.75)

Table 3: A comparison of classifying by different heuristics. Results are averaged across all the idioms in the two corpora.

Model	Avg. F_{fig}	Avg.Acc
Gibbs	0.58 (0.31 $\sim$ 0.78)	0.57 (0.4 $\sim$ 0.78)
EM	0.56 (0.31 $\sim$ 0.71)	0.6 (0.42 $\sim$ 0.77)
MinV+NN	0.68 (0.41 $\sim$ 0.83)	0.67 (0.55 $\sim$ 0.86)

Table 4: The performance of MinV+NN and models without soft label on all the idioms in the two corpora

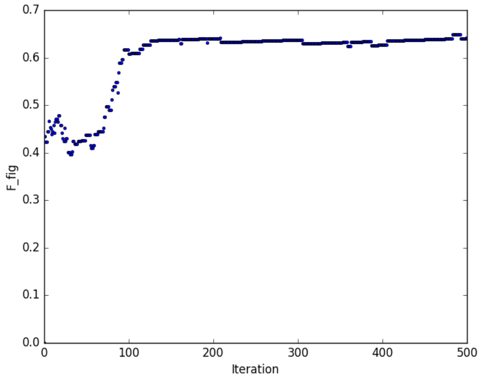

Figure 2: The performance of MinV+infGibbs on the idiom ”break a leg”

Table 4 shows the performances of the new models, which are all worse than our full models MinV +infGibbs and MinV +infEM. This highlights the advantage of integrating distributional semantic information and local features into one single learning procedure. Without the informed prior (encoded by the soft labels), the Gibbs sampling and EM algorithms only seek to maximize the probability of the observed data, and may fail to learn the underlying usage structure.

The model MinV +NN is not as competitive as our full models. It is too sensitive to the selected instances. Even though the training examples are instances that MinV is the most confident about, there are still mislabelled instances. These ”noisy training examples” would lead the NN classifier to make unreliable predictions. In contrast, our unsupervised learning is less sensitive to the performance of MinV; it can achieve a decent performance for an idiom even when the quality of the soft labels is poor. For example, when using MinV as a stand-alone model for *break a leg*, its figurative F-score is only 0.43, but through further training, the full model MinV+infGibbs achieves 0.64. Fig. 2 shows the training curve. A possible reason for this phenomenon is that the soft label is integrated into the learning process by biasing the sampling procedure (see Equation 3). We only encourage our model to follow the distributional semantic evidence captured by soft label and do not force it. So if there are strong evidences encoded by the linguistically motivated features in the instances to overcome the soft label it still has the freedom to do so.

5 Conclusion

We have presented an unsupervised method for idiom usage recognition built upon the heuristic that instances that use the idiom literally are semantically closer to constituent words of the idiom. Experimental results on two different corpora suggest that our models are competitive against supervised methods and prior unsupervised methods.

References

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *EACL*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Lorna Byrne, Caroline Fenlon, and John Dunnion. 2013. IIRG: A naive approach to evaluating phrasal semantics. *In Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation*, 45(4).

Cristina Cacciari and Maria Chiari Levorato. 1998. The effect of semantic analyzability of idioms in metalinguistic tasks. *Metaphor and Symbol*, 13(3):159–177.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

Max Coltheart. 1981. The mrc psycholinguistic database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2008. The vnc-tokens dataset.

Graff David, Kong Junbo, Chen Ke, and Maeda Kazuaki. 2005. English gigaword second edition ldc2005t12. *Linguistic Data Consortium*.

Rong En Fan, Kai Wei Chang, Cho Jui Hsieh, Xiang Rui Wang, and Chih Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Afsaneh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised type and token identification of idiomatic expressions. *Computational Linguistics*, 35(1):61–103.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, pages 363–370.

Raymond W Gibbs, Nandini P Nayak, and Cooper Cutting. 1989. How to kick the bucket and not decompose: Analyzability and idiom processing. *Journal of memory and language*, 28(5):576–593.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Florentina T. Hristea. 2013. *The Naïve Bayes Model in the Context of Word Sense Disambiguation*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multiword expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 12–19.

Ioannis Korkontzelos, Torsten Zesch, Fabio Massimo Zanzotto, and Chris Biemann. 2013. Semeval-2013 task 5: Evaluating phrasal semantics.

Linlin Li and Caroline Sporleder. 2009. Classifier combination for contextual idiom detection without labelled data. In *EMNLP*, pages 315–323.

Changsheng Liu and Rebecca Hwa. 2017. Representations of context in recognizing the figurative and literal usages of idioms. In *AAAI*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2):103–134.

Jing Peng, Anna Feldman, and Ekaterina Vylomova. 2014. Classifying idiomatic and literal expressions using topic models and intensity of emotions. *EMNLP*, pages 2019–2027.

Nazneen Fatema Rajani, Edaena Salinas, and Raymond Mooney. 2014. Using abstract context to detect figurative language.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Giancarlo Salton, Robert Ross, and John Kelleher. 2014. An empirical study of the impact of idioms on phrase based statistical machine translation of english to brazilian-portuguese.

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *EACL*, pages 754–762.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *EMNLP*, pages 680–690.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Lowri Williams, Christian Bannister, Michael Arribas-Ayllon, Alun Preece, and Irena Spasić. 2015. The role of idioms in sentiment analysis. *Expert Systems with Applications*, 42(21):7375–7385.

Coming to Your Senses: on Controls and Evaluation Sets in Polysemy Research

Haim Dubossarsky[1], **Eitan Grossman**[2] and **Daphna Weinshall**[3]
[1] Edmond and Lily Safra Center for Brain Sciences
[2] Department of Linguistics
[3] School of Computer Science and Engineering
The Hebrew University of Jerusalem, 91904 Jerusalem, Israel
`haim.dub@gmail.com, {eitan.grossman,daphna}@mail.huji.ac.il`

Abstract

The point of departure of this article is the claim that sense-specific vectors provide an advantage over normal vectors due to the polysemy that they presumably represent. This claim is based on performance gains observed in gold standard evaluation tests such as word similarity tasks. We demonstrate that this claim, at least as it is instantiated in prior art, is unfounded in two ways. Furthermore, we provide empirical data and an analytic discussion that may account for the previously reported improved performance. First, we show that ground-truth polysemy degrades performance in word similarity tasks. Therefore word similarity tasks are not suitable as an evaluation test for polysemy representation. Second, random assignment of words to senses is shown to improve performance in the same task. This and additional results point to the conclusion that performance gains as reported in previous work may be an artifact of random sense assignment, which is equivalent to sub-sampling and multiple estimation of word vector representations. Theoretical analysis shows that this may on its own be beneficial for the estimation of word similarity, by reducing the bias in the estimation of the cosine distance.

1 Introduction

Polysemy is a fundamental feature of natural languages, which typically have many polysemic words. *Chair*, for example, can refer to either a piece of furniture or to a person in charge of a meeting. Therefore both theoretical linguistics and computational linguistics seek to establish principled methods of identifying the senses that together constitute the meaning of words.

It is commonly assumed or claimed that standard word embeddings are unable to capture polysemy (Iacobacci et al., 2015), which results in sub-optimal performance in gold standard evaluation tests such as word similarity tasks, and potentially hamper performance in downstream tasks. The corollary assumption is that sense-specific representations will lead to improved performance on these evaluation tests. This assumption is conceptually attractive, since it makes sense that sense-specific representations are more accurate than global representations which conflate the different senses of a word into a single representation. For example, in translating 'chair,' it is reasonable that performance should improve if the two senses are represented separately. This view is supported by several studies (Huang et al., 2012; Neelakantan et al., 2014; Chen et al., 2014; Li and Jurafsky, 2015; Iacobacci et al., 2015; Mancini et al., 2017), which argue that sense-specific representations lead to improved performance in word similarity tasks.

Ideally, such claims about polysemy should be evaluated using a gold standard evaluation set that is tailored specifically for polysemous words. As this set does not exist, tasks involving word similarity tests have been used as a proxy (see Section 2). The underlying hypothesis is that enriching word vector representations with polysemic information should express itself in performance gains in these tasks. However, this hypothesis has never been tested directly, and the ability of word similarity tasks to directly benefit from polysemic information must first be validated if they are to serve as genuine evaluation sets in polysemy research. Until then, the validity of any reported positive effects of sense-specific representations on evaluation tests is to be treated with caution.

In this paper our first aim is to assess the validity of *word similarity tasks* as proper evaluation tests for polysemic word representations. We use two independent corpora in order to obtain polysemic vectors: (i) a sense-annotated corpus, and (ii) an artificially-induced annotated corpus, constructed

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1732–1740
Brussels, Belgium, October 31 - November 4, 2018.

by using an established method that we modify for our purposes. Surprisingly, our analyses show that even the most accurate sense-specific word vectors do not improve performance. In fact, peak performance is achieved when polysemic information is *ignored*, i.e., when all different annotated senses are collapsed to a single word, as they naturally appear in a text. These counter-intuitive results indicate that the word similarity tasks are not a suitable test to evaluate polysemic representations.

Although these negative results point to the inadequacy of the evaluation test, they also stress a necessary critical analysis of the sense-specific vectors. Specifically, they give rise to the following question: why might previously reported polysemic representations show superior performance in these inadequate evaluation tests?

One explanation for the reported effects may lie in an inherent property of sense-specific representations. The procedure of assigning a word occurrence to a particular sense amounts to a sampling procedure. This sampling procedure itself, regardless of its validity (whether its sense assignments are correct or not), may be the true source of the reported performance gains. To test this hypothesis, we created a control condition in which word occurrences are randomly assigned to different senses. Determining that an effect is attributable to *genuine polysemy* can only be established if a similar effect is lacking or significantly reduced in this control condition.

We demonstrate that performance gains are indeed obtained for a corpus with randomly assigned senses. In addition, we modify two models for sense-specific polysemy representation (Li and Jurafsky, 2015; Mancini et al., 2017) to randomly assign words to senses, and observe that the effect size remains unchanged between the original and random conditions.

In support of our empirical findings, we discuss the difficulty of obtaining an unbiased estimator for the cosine distance between two normalized random variable vectors. This distance is the basis of all word similarity tasks that serve to evaluate performance, and thus it may provide a partial explanation for the empirical findings, under the assumption that words are better represented as a population of vectors. Specifically, the true source of the reported performance gains may be an artifact of a purely statistical benefit that derives from the assignment of words to particular senses, or separate sub-samples, which subsequently reduces the bias of the similarity estimator.

We thus identify two independent pitfalls in NLP research on polysemy representation. First, the inadequacy of currently-used evaluation tests to properly assess polysemic representations. And second, the essence of polysemic representations, whose reported benefits might not come from polysemic information per se, but rather from other unrelated sources.

2 Background

Previous attempts to use polysemic information for enriching word representation used normal unannotated corpora, and therefore disambiguated the different senses of words before exploiting any sense-specific information. Prior art has either taken an automatic approach to detect word senses based on the statistics extracted from texts (Huang et al., 2012; Neelakantan et al., 2014; Li and Jurafsky, 2015), or relied on external lexical resources (e.g., WordNet or BabelNet) which guarantee that the detected senses are mapped to true sense distinctions in natural language (Chen et al., 2014; Iacobacci et al., 2015; Mancini et al., 2017). Importantly, both types of models report marked performance gains in evaluation tests.

This kind of approach produces (i) global vectors that represent a word's meaning as a single vector (with no subdivision into distinct senses), as well as (ii) sense-specific vectors representing individual senses of words, determined in the disambiguation step, as separate vectors. For example, such approaches would represent the meaning of *chair* as a single vector, as well as distinct vectors for each of its multiple senses, e.g., "chair (person)" and "chair (furniture)".

In order to evaluate performance, the vectors created by the models are standardly evaluated using word similarity tasks (the most common are WordSim-353 (Finkelstein et al., 2001) and Stanford's Contextual Word Similarities (SCWS) (Huang et al., 2012)). These tasks comprise pairs of words and the similarity scores assigned to them by human annotators. For example, the similarity between *table* and *chair* might be rated as 0.8 (i.e., human annotators found these words to be very similar, but not perfectly so), while the similarity between *table* and *tree* might be rated as 0.3 (not very similar). The embedding models produce similarity scores for each word pair by computing the cosine-distance between the word vectors for each pair. The model's performance is then evaluated as the rank-order similarity in the order of pairs (Spearman correlation) between the human annotators' scores and the scores produced by the model. In line with the assumption discussed above, one would predict that the rank-order similarity produced by the sense-specific vectors should outperform the one produced by the global vectors. In particular, more accurate sense-specific vectors should produce better results in these tasks; conversely, better performance on these tasks is interpreted as indicating that polysemy has been captured more accurately. We directly test these two predictions in this paper.

Computing word similarity is straightforward when each word is represented as a single vector, but it is less so when the meaning of a polysemic word is represented by multiple sense-specific vectors. This problem of *matching* the senses relevant for a specific word pair, i.e., matching the "person" sense of *chair* with the correct sense of the word *meeting*, poses a major hurdle for meaningful comparison, and has been tackled in three different ways: (i) *average* over all similarity scores between all the different possible pairs; (ii) *weighted average* over these scores according to the probability of senses assigned by the disambiguation model; or (iii) *selection* of the most suitable sense according to the disambiguation model, and using only the corresponding similarity score.

Intuitively, the third approach should outperform the others, as it is based on the clearest distinction between the relevant and non-relevant senses. However, this naïve prediction is not supported by previous studies. Rather, the best results are usually obtained for *average* and *weighted average*, followed by *global* (ignoring polysemy), while *selection* falls far behind the others. This counter-intuitive finding suggests that the observed benefit may be less related to sense disambiguation than previously supposed[1].

3 Task validation

Generally, before any task can be used as an evaluation testbed for polysemy discovery algorithms or polysemous representations, we argue that the *task* itself should be validated as suitable (or not) for the intended purpose. We propose the following **task validation** methodology: (i) Start by identifying a corpus where polysemic information is known for a significant number of words. (ii) Compute two sets of word representations: $\mathcal{A}_1$ - which computes a single representation for all words in the corpus, and $\mathcal{A}_2$ - which computes multiple representations for each polysemic word in the corpus based on the different *known* senses of the word. (iii) Evaluate the task using the two representation sets $\mathcal{A}_1$ and $\mathcal{A}_2$. Only if significant performance gains can be shown when using $\mathcal{A}_2$ as compared to $\mathcal{A}_1$, the task can be used to evaluate polysemy representation.

3.1 Polysemy induction

A major drawback of the proposed methodology is that such annotated corpora are scarce, and the largest among them are still small (OntoNotes (Weischedel et al., 2013) comprises 1.5 million words, cf. unannotated corpora (e.g. Wikipedia) which are about 1000 times larger). We therefore articulate a methodology to generate a *task validation test* from **any corpus** and **evaluation task**, even without prior annotation of polysemy, that is based on the *pseudo-words* approach (Gale et al., 1992; Pilehvar and Navigli, 2014).

More specifically, we induce polysemy in a natural corpus by pairing words, and collapsing every pair of words into a single word-form while keeping their "original identity" as *polysemy annotation*. For example *ring* and *table* may be collapsed to a single word with two senses, *table*^{1} and *table*^{2} respectively. The new corpus is polysemic with respect to the collapsed words, while all other words keep a single sense. This corpus has most of the features of a natural corpus, but unlike most natural corpora (and all large corpora), it contains polysemy annotation. Subsequently, the relevant items in the *word similarity tasks* are collapsed in the same way, making the items polysemous, and thus the tasks suitable as *validation tests*.

With these polysemy-induced corpus and word similarity tasks, we follow the methodology for *task validation* described above. Only if a model that is based on multiple representations per polysemous word leads to a significant performance gain in the task as compared to a model with single representation for each word then the task un-

---

[1] Iacobacci et al. (2015) and Mancini et al. (2017) only reported results on *selection*, for which they found performance gains. For consistent comparison with other models we report *average* (but using their code), noting that it also provides performance gains over *global* vectors.

der examination should be considered adequate to evaluate the utility of polysemy representation of word senses.

3.2 Methods

Word embedding model *word2vec-SkipGram* model (Mikolov et al., 2013) is used to obtain vector representations for words and senses. The model is separately trained over the corpus: first producing sense-specific vectors according to the annotated senses, and next producing global vectors by ignoring the annotated senses and collapsing all their occurrences to a single word. Throughout the analyses we use an embedding size of 300d, a window size of 5 words from each side of the target word, negative-sampling of 5 words, and an initial learning rate of 0.025.

Sense-annotated corpus We use OntoNotes (Weischedel et al., 2013), the largest available corpus annotated for word senses. This allows us to circumvent the problem of first disambiguating the words' senses, and thus to directly test the utility of using polysemic information in word vector representations. The corpus contains 1.5 million English tokens, comprising about 50k English word types, of which 8675 word types are sense-annotated. Because the annotation is not uniform throughout the corpus (words are not annotated every time they appear), which can bias the analysis described below, we extract a subset of the corpus by removing sentences where polysemous words are not annotated, thus removing 40% of the corpus. Stopwords as well as words occurring less than 10 times are ignored by the word embedding models. All words are lowercased.

Sense-induced corpus Wikipedia dump (04/2017) is the original corpus from which a polysemic version is induced. We separately used a list of semantically-aware pseudowords (Pilehvar and Navigli, 2013), in addition to random pairs from the 6000 most frequent words to collapse them into a single word-form (see Section 3.1). Stopwords and infrequent words (<300 tokens) are ignored by the word embedding models.

Evaluation Polysemic word representations are evaluated on the two word similarity tasks described. Crucially, the problem of *matching* the relevant sense in these tests (described in Section 2) is tackled by taking the *average* of the sense-specific representations and comparing it to the *global* word representations[2].

3.3 Results

Results clearly demonstrate that global representations are significantly superior to sense-specific representations in both evaluation tests and across corpora, as shown in Table 1 and Figure 1.

	OPTIMAL	GLOBAL	AVERAGE
WS-353			
OntoNotes		44.7	41.3
Induced Rand.	70	68.7	66.4
Induced S.A.	70	70.2	67.1
SCWS			
OntoNotes		64.0	62.6
Induced Rand.	66	64.3	55.2
Induced S.A.	66	65.9	65.7

Table 1: Word similarity scores for OntoNotes and induced-polysemy Wikipedia. OPTIMAL: performance obtained for the original Wikipedia (apply only for the induced method), GLOBAL: sense information ignored, AVERAGE: sense-specific vectors averaged. S.A. & Rand.: pairing methods for polysemy induction, semantically-aware and random, respectively.

The critical comparison between the *global* and *average* conditions is expected to show better performance for the latter under the assumption that polysemic information improves performance. This difference (marked as *actual gain* in Figure 1), surprisingly shows an opposite outcome, which means that word similarity tasks fail to demonstrate the value of polysemy representation in improving performance.

Complementarily, we report performance for the *optimal* condition. In this condition, word similarity scores are obtained for a model that was trained on the the original Wikipedia corpus (before polysemy induction), and thus represents an upper bound for the performance of any polysemy model that would be trained on an induced polysemy version of the same corpus (i.e., the maximum *ideal gain* a model can achieve over its *global* vectors). The fact that performance is highest for that condition is thus expected, and reassures us that the induced polysemy procedure has worked as planned for both kinds of induction.

Overall, our results points to a failure of polysemy models to improve performance over *global* vectors by *averaging* sense-specific vectors. This worsening in performance of the sense-specific

---

[2] Recall that *average* was reported to be one of the best performing matching methods is previous work.

vectors stands in marked contrast to previous studies which reported performance gains (see Table 2 and *Reported gains* in Figure 1).

	GLOBAL	AVERAGE
WS-353		
Huang (2012)	22.8	71.3
Neelakantan (2014)	69.2	70.9
Li (2015)*	61.0	67.8
Mancini (2017)*	49.1	55.6
SCWS		
Huang (2012)	58.6	62.8
Neelakantan (2014)	65.5	67.2
Chen (2014)	64.2	66.2
Li (2015)	64.6	66.4
Mancini (2017) *	57.2	62.1

Table 2: Previously reported results in word similarity tasks. *Reproduced (see section 4.1) where consistent comparison on the same tasks is not originally reported.

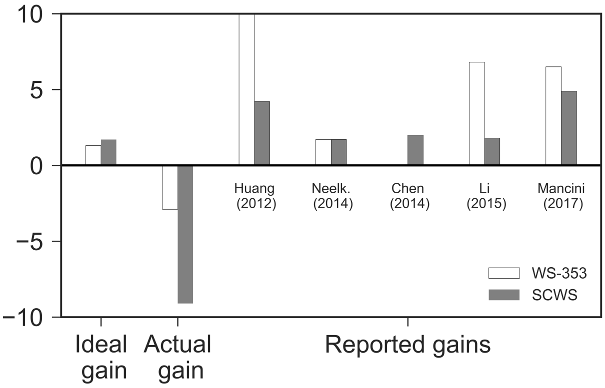

Figure 1: Summary of results in Tables 1, 2, showing *performance gains* across conditions. *Ideal gain*: max. performance gains using fully annotated corpus with perfect sense matching, *Actual gain*: de-facto gain (global - average), and *Reported gains*: previous reported results. Color (dark and white) marks the tasks.

3.4 Discussion

The main result described above is negative, demonstrating that *word similarity tasks* are not suitable to evaluate polysemy representation. However, the methodology we proposed for *polysemy induction* constitutes a positive contribution, as it can be used to effectively test any task for its utility in the evaluation of polysemy representation while using any corpora. This may aid in finding tasks which are more suitable to serve as gold standard evaluation tests for polysemy. Moreover, the use of polysemy induction for these purposes adds yet another type of control to the NLP toolbox; such controls are scarcely implemented in NLP studies (but see Dubossarsky et al. (2017)).

Importantly, the inability of the induced polysemy models to produce positive performance gains as reported in prior art may indicate that these reported gains do not reflect benefits from true polysemy, but rather from an unknown factor that boosts performance.

4 The statistical signature of polysemy

In order to further investigate the reason for the lack of performance gains, we analyzed the properties of the sense-specific vectors in the induced polysemy conditions, and compared them to those obtained by previous studies.

We looked at the pairwise similarity between the different sense-specific representations of the same word. We start from the observation that polysemy is inherently defined by word senses that are distinguishable from each other. Importantly, we compared the models that did not produce performance gains in word similarity tasks to those which did report such gains.

4.1 Analysis and results

We tested four sets of sense-specific representations: two from our polysemy induction models (see Section 3.2) and two from Li and Jurafsky (2015)[3] and Mancini et al. (2017)[4] which reported performance gains. For each word in each set, the average cosine-distance between its different sense-specific vectors is computed. The distribution of these average distances within a specific set is defined as its ”polysemic signature”, which is then compared across sets.

The results shown in Figure 2 reveal marked differences in the polysemic signature between the four sets. A high polysemic signature is seen for the two induced polysemy sets, which are the only sets with guaranteed semantically different senses. In these sets, the polysemy model that is based on the random pairing of words has a higher polysemic signature than the one based on semantically-aware pairing. This is well expected, as semantically-aware pseudowords are designed to simulate ”true polysemy” in which the different senses of a word are still semantically related, unlike random pairing which produces a ”coarse” distinction more similar to homonymy (Pilehvar and Navigli, 2014).

The critical comparison to the two sets of polysemy models that did find performance gains

---

[3] https://github.com/jiweil/mutli-sense-embedding

[4] http://lcl.uniroma1.it/sw2v/

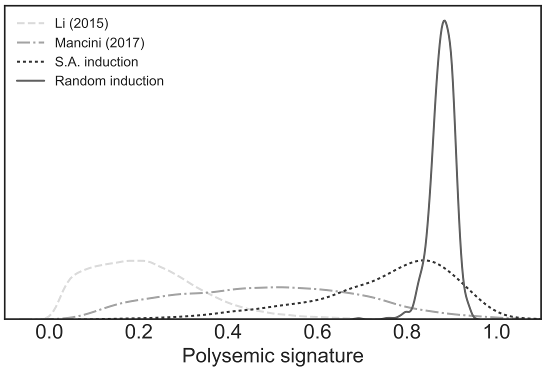

Figure 2: Density distribution of polysemic signatures for the four sets, see text for details.

shows that these models exhibit a smaller polysemic signature, as the different senses seem to be more similar to one another. Crucially, even the polysemic signature of Mancini et al. (2017), which exhibit an intermediate polysemic signature, shows greater similarity in its senses as compared to the sense-aware polysemy induction model, which presumably represents a more subtle (and ecological) model for polysemy.

4.2 Discussion

The broader implications of these results on our research hypothesis can be understood in the context of the findings reported in Section 3.3. The results described in Figure 1 demonstrate a marked difference between previously *reported gains* and the *actual gain* condition which shows a **worsening** of performance in the same task when polysemic information is included. The results demonstrated in Figure 2 can be described as a negative image of those presented in Figure 1. Specifically, the *actual gain* condition of the induced-polysemy has the largest polysemic signature as compared to the other conditions.

Together, these results indicate that the condition that demonstrates polysemy most clearly shows the poorest performance in the evaluation tests. The converse is also true: the conditions that diverge from the clearest polysemic representation show heightened performance in the evaluation tests. A gold standard for polysemy representation should entail that given optimal vector representations, performance on the evaluation tests would be optimal, and vice versa. Since our results demonstrate that the directions of optimal vector representation and optimal test performance are opposite, we are led to the following conclusions: First, polysemy models which provide improvement in word similarity tasks may not necessarily be suitable for polysemous vector representations. Second, word similarity tasks are not suitable evaluation tests for polysemy representation.

5 Theoretical discussion

In this section we recall and analyze some properties of the cosine distance, and describe how they may partially account for the empirical observations discussed in this article. The crucial point is to model the contextual representation of words as a distribution over some vector space.

Let X_i denote the random variable which captures the contextual representation of word i. Let $\{X_i^l, X_j^l \in \mathbb{R}^d\}_{l=1}^L$ denote a sample of such representations for words i, j respectively, where L denotes the sample size. d corresponds to the dimension of the vector space when using word2vec representations, or the number of words in the dictionary when using explicit representations (e.g., PPMI) (see Section 3.2). To simplify the analysis, we further assume that $\|X^l\| = 1 \; \forall l$.

The similarity between two words i, j can be plausibly measured (as customarily done) by the cosine distance between their contextual representations, namely, $\mathbb{E}\left[X_i X_j\right]$:

$$\mathbb{E}\left[X_i X_j\right] = \mathbb{E}\left[X_i\right] \mathbb{E}\left[X_j\right] + \operatorname{cov}(X_i, X_j) \quad (1)$$

Thus the average distance is not equivalent to the distance between the average representations alone, but has to include an additional bias term - $\operatorname{cov}(X_i, X_j)$ - which reflects the statistical dependence between the two vector representations X_i, X_j. This term is significant, because the contextual representation of two words is likely to exhibit strong dependence, especially when the words are more similar.

This is where the problem lies. In the process of generating words' representations, we start from a sample of sentences and generate a single representation. This representation is essentially our estimate of $\mathbb{E}\left[X_i\right]$ for word i. When multiplying two such representations in order to compute the cosine distance between them, we obtain an estimate for $\mathbb{E}\left[X_i\right] \mathbb{E}\left[X_j\right]$, which is *not* a good estimate for $\mathbb{E}\left[X_i X_j\right]$ because of the bias term in (1).

Ideally, in order to provide an unbiased estimate of $\mathbb{E}\left[X_i X_j\right]$, we should divide the sample of sentences into mini-batches, compute the appropriate contextual representation for both words

i, j from each mini-batch, and then directly estimate $\mathbb{E}[X_i X_j]$ by taking the average multiplication of the corresponding representations in each mini-batch. Interestingly, in the process of generating polysemous representations, whether relying on true polysemy or arbitrary polysemy, we essentially accomplish the same goal: for each word, a mini-batch is replaced by the subset of sentences in which only one of the word's meanings is present[5].

If sense matching (see Section 2) is achieved by way of *average* or *weighted average*, it implies that our estimate of word similarity should improve with the number of senses used in the analysis, especially when the assignment is arbitrary. Of course, any improvement is hampered by the deterioration in the quality of the contextual representation computed from the smaller mini-batch sample, and therefore improvement is only expected for a small number of real or artificial ”senses”.

6 Performance gains revisited

The empirical findings presented so far converge on the conclusion that the performance gains reported in prior art may not stem from the utility of polysemic information, as previously claimed, but are the result of an alternative source. In the theoretical discussion we argue that random sense annotation is equivalent to sub-sampling and multiple estimation of contextual vector representations, and that this alone may be beneficial for the task performance of word similarity. It is reasonable to conclude that these repeated sampling procedures may have produced the reported performance gains. Here we test this hypothesis directly.

We propose a simple control condition, in which senses are randomly assigned to words in a corpus, and sense-specific vectors are produced in the same way as before. Determining that an effect is reliably attributed to *genuine polysemy* can only be established if a similar effect is lacking or significantly reduced in this control condition.

6.1 Random sense assignment

We achieve random sense assignments in 2 ways:

Sampling from a known distribution. For the entire corpus and vocabulary (100k words, and note that Neelakantan et al. (2014) and Li and Jurafsky (2015) also took this entire vocabulary approach), we assigned senses at random from a categorical distribution under two conditions. In the first we used a uniform prior which produced equal sense probabilities for each word, and in the second condition we used a biased prior in which one sense predominates. We also experimented with different number of senses per word. We found that the results differed only slightly between the conditions and across the different number of senses. The same regular embedding model was trained as before.

Sampling from an unknown distribution. To test the hypothesis more directly against the sense distributions used in prior work, we reproduced sense-specific vectors of two models using their code: (1) for Li and Jurafsky (2015) we kept their Chinese-Restaurant-Process probabilistic mechanism, where senses are assigned to words based on the similarity of their contexts. We only shuffled the elements of the final vector of sense assignments produced by the model, and (2) for Mancini et al. (2017) we shuffled between the sense tags of each word in their original sense-annotated Wikipedia. For further comparisons we used the original code unchanged to reproduce another set of global and sense-specific vectors for each model separately to serve as a baseline.

6.2 Performance boost due to word sampling

Table 3 shows the results of these simulations. Regular embedding with random sense assignments shows marked performance gains of the sense-specific vectors over the global vectors. In fact, the effect reported in Neelakantan et al. (2014) (which report the highest score in the SCWS task, see Table 2) is replicated almost exactly, perhaps due to the fact that they also used a fixed number of senses for each word as we did in this simulation. Furthermore, the reanalysis of previous models lead to almost identical results in the original and random conditions, which means that randomly assigning words to senses does not weaken the effect.

Together, these three control simulations clearly show that an effect of the same magnitude as previously reported in several studies emerges under random sense assignment. Therefore, our findings strongly undermine the assumption that the reported effects are in any way related to actual polysemy, and strongly suggests that it is repeated sub-sampling that boosts performance.

---

[5] For the purpose of this discussion we ignore sentences in which a word appears more than once.

	ORIGINAL		RANDOM	
	GLOB.	AVE.	GLOB.	AVE.
WS-353				
Regular embd			70	69.8
Li (2015)	61.0	67.8	60.5	68.2
Mancini (2017)	49.1	55.6	58.3	64.5
SCWS				
Regular embd			65.7	67.3
Li (2015)	58.9	66.2	57.4	66.2
Mancini (2017)	57.2	62.1	62.4	65.2

Table 3: Word similarity scores for random sense assignments, compared to the original (where available).

7 Summary and Discussion

Here we investigate the common-held view that polysemy representation improves performance. Specifically, we question the validity of word similarity tasks as a suitable evaluation test that would allow drawing such conclusions. To test the claim that resolving the polysemy of words improves performance in these tasks, as was repeatedly reported in prior art, we used real-world polysemy in two independent conditions: (i) a human sense-annotated corpus and (ii) a corpus in which polysemy was induced in a controlled artificial fashion. In both conditions, the performance in word similarity tasks *deteriorated*. We claim that this negative finding alone may suffice to determine that word similarity tasks are not suitable tests for evaluating polysemy representation.

However, if the word similarity task is inadequate to evaluate polysemy, why would it show high gains for polysemic representations as reported in many previous studies? To investigate this question we first ask whether polysemic information per se is required to drive such effects, or whether these effects are artificially caused by the procedures by which polysemous representations are created. To test this, we set out to demonstrate that even representations that bear no genuine polysemic information could nonetheless yield improved performance due to a methodological artifact. We thus created control conditions, in which we randomly assign word occurrences to senses, and found that randomly-produced sense-specific vectors indeed show marked improvements in performance. Since this effect cannot stem from polysemy (which is lacking in these conditions), it may only be the result of a methodological artifact - the sampling procedure entailed by the assignment of words to senses.

The existence of a sampling artifact is supported by our theoretical discussion, showing that multiple vector sampling can lower the bias of the estimator of the cosine distance between two vectorial random variables. The underlying assumption is that a better model for contextual word representation should employ a population of vectors. Interestingly, the conclusion that word vectors are better if constructed from multiple representations might apply to word vectors in general, and not just to sense-specific vectors. However, such a claim is outside the scope of this study and remains for future research.

This account is further supported by our *polysemic signature* analysis, which shows that the similarity between the senses in the polysemic models that produced performance gains is greater than in the models that did not produce this effect (the polysemy-induced models). This finding is in line with the sampling artifact account, as sense-specific vectors which are based on random sense assignments are expected to be more similar compared to sense-specific vectors which are based on true polysemic distinctions.

We stress that our analysis does not argue for or against the accuracy of sense-specific vectors in capturing true polysemy (other tests exist for that purpose). Instead, it focuses only on the true source of previously reported performance gains of this kind of representation, and on the validity of word similarity tasks for their evaluation.

All in all, we provide converging evidence, both experimental and theoretical, that word similarity tasks do not provide a marker for the utility of polysemic information. Rather, performance gains in word similarity tasks are an artifact of the procedure by which polysemic representations are created. These conclusions join a general skepticism expressed in the literature about the use of word similarity tasks for the evaluation of word vectors (Hill et al., 2015; Avraham and Goldberg, 2016; Batchkarov et al., 2016; Faruqui et al., 2016).

Essentially, our findings mean that there is no solid empirical foundation to the claim that polysemic information improves performance in evaluation tests. In fact, they corroborate the general impression that polysemic representation does not improve performance on most downstream tasks (Li and Jurafsky, 2015). It may be the case that sense-specific vectors can or will show heightened performance on evaluation tests, or improve downstream tasks. This, however, will have to be demonstrated on the basis of tasks whose suitability has been properly validated. Moreover, any effects reported will have to be supported by demonstrating that these effects are absent or strongly reduced in a properly articulated condition that should control for the sampling artifact.

Critically, until a reliable evaluation test exists, research on polysemic word representation is seriously hampered. In fact, a re-evaluation of past models would be in place, as both ”positive” and ”negative” results that were previously reported are in fact invalid.

Acknowledgments

We thank Matan Ben-Yosef for his invaluable contribution in executing and manipulating the code of Li and Jurafsky (2015).

References

Oded Avraham and Yoav Goldberg. 2016. Improving reliability of word similarity evaluation by redesigning annotation task and performance measure. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 106–110.

Miroslav Batchkarov, Thomas Kober, Jeremy Reffin, Julie Weeds, and David Weir. 2016. A critique of word similarity as a method for evaluating distributional semantic models. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 7–12.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*, pages 1025–1035.

Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. 2017. Outta control: Laws of semantic change and inherent biases in word representation models. In *Proceedings of EMNLP*, pages 1136–1145.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 30–35.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

William A Gale, Kenneth W Church, and David Yarowsky. 1992. Work on statistical methods for word sense disambiguation. In *AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, volume 54, page 60.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 95–105.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings EMNLP*, pages 1722–1732.

Massimiliano Mancini, Jose Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2017. Embedding words and senses together via joint knowledge-enhanced training. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 100–111.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Arvind Neelakantan, Jeevan Shankar, Re Passos, and Andrew Mccallum. 2014. Efficient nonparametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*.

Mohammad Taher Pilehvar and Roberto Navigli. 2013. Paving the way to a large-scale pseudosense-annotated dataset. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1100–1109.

Mohammad Taher Pilehvar and Roberto Navigli. 2014. A large-scale pseudoword-based evaluation framework for state-of-the-art word sense disambiguation. *Computational Linguistics*, 40(4):837–881.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Predicting Semantic Relations using Global Graph Properties

Yuval Pinter
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA
`uvp@gatech.edu`

Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA
`jacobe@gatech.edu`

Abstract

Semantic graphs, such as WordNet, are resources which curate natural language on two distinguishable layers. On the local level, individual relations between synsets (semantic building blocks) such as hypernymy and meronymy enhance our understanding of the words used to express their meanings. Globally, analysis of graph-theoretic properties of the entire net sheds light on the structure of human language as a whole. In this paper, we combine global and local properties of semantic graphs through the framework of Max-Margin Markov Graph Models (M3GM), a novel extension of Exponential Random Graph Model (ERGM) that scales to large multi-relational graphs. We demonstrate how such global modeling improves performance on the local task of predicting semantic relations between synsets, yielding new state-of-the-art results on the WN18RR dataset, a challenging version of WordNet link prediction in which "easy" reciprocal cases are removed. In addition, the M3GM model identifies multirelational motifs that are characteristic of well-formed lexical semantic ontologies.

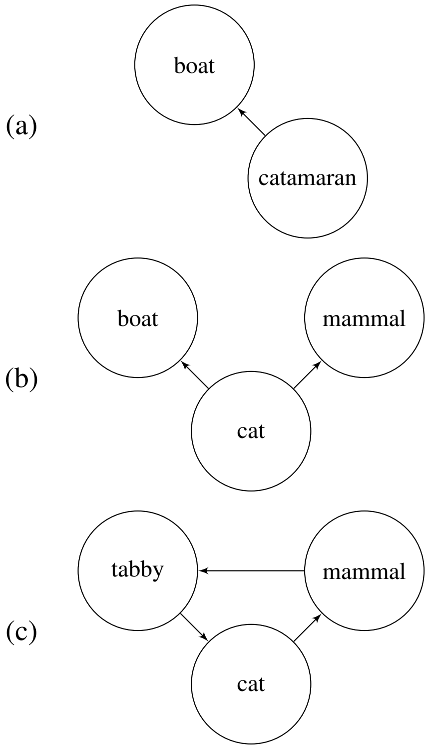

Figure 1: Probable (a) and improbable (b-c) structures in a hypothetical hypernym graph.

1 Introduction

Semantic graphs, such as WordNet (Fellbaum, 1998), encode the structural qualities of language as a representation of human knowledge. On the local level, they describe connections between specific semantic concepts, or **synsets**, through individual edges representing relations such as hypernymy ('is-a') or meronymy ('is-part-of'); on the global level, they encode emergent regular properties in the induced relation graphs. Local properties have been subject to extensive study in recent years via the task of **relation prediction**, where individual edges are found based mostly on distributional methods that embed synsets and relations into a vector space (e.g. Socher et al., 2013; Bordes et al., 2013; Toutanova and Chen, 2015; Neelakantan et al., 2015). In contrast, while the structural regularity and significance of global aspects of semantic graphs is well-attested (Sigman and Cecchi, 2002), global properties have rarely been used in prediction settings. In this paper, we show how global semantic graph features can facilitate in local tasks such as relation prediction.

To motivate this approach, consider the hypothetical hypernym graph fragments in Figure 1: in (a), the semantic concept (synset) 'catamaran' has

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1741–1751
Brussels, Belgium, October 31 - November 4, 2018.

a single hypernym, 'boat'. This is a typical property across a standard hypernym graph. In (b), the synset 'cat' has two hypernyms, an unlikely event. While a local relation prediction model might mistake the relation between 'cat' and 'boat' to be plausible, for whatever reason, a high-order graph-structure-aware model should be able to discard it based on the knowledge that a synset should not have more than one hypernym. In (c), an impossible situation arises: a cycle in the hypernym graph leads each of the participating synsets to be predicted by transitivity as its own hypernym, contrary to the relation's definition. However, a purely local model has no explicit mechanism for rejecting such an outcome.

In this paper, we examine the effect of global graph properties on the link structure via the WordNet relation prediction task. Our hypothesis is that features extracted from the entire graph can help constrain local predictions to structurally sound ones (Guo et al., 2007). Such features are often manifested as aggregate counts of small subgraph structures, known as **motifs**, such as the number of nodes with two or more outgoing edges, or the number of cycles of length 3. Returning to the example in Figure 1, each of these features will be affected when graphs (b) and (c) are evaluated, respectively.

To estimate weights on local and global graph features, we build on the **Exponential Random Graph Model** (ERGM), a log-linear model over networks utilizing global graph features (Holland and Leinhardt, 1981). In ERGMs, the likelihood of a graph is computed by exponentiating a weighted sum of the features, and then normalizing over all possible graphs. This normalization term grows exponentially in the number of nodes, and in general cannot be decomposed into smaller parts. Approximations are therefore necessary to fit ERGMs on graphs with even a few dozen nodes, and the largest known ERGMs scale only to thousands of nodes (Schmid and Desmarais, 2017). This is insufficient for WordNet, which has an order of 10^5 nodes.

We extend the ERGM framework in several ways. First, we replace the maximum likelihood objective with a margin-based objective, which compares the observed network against alternative networks; we call the resulting model the Max-Margin Markov Graph Model (M3GM), drawing on ideas from structured prediction (Taskar et al., 2004). The gradient of this loss is approximated by importance sampling over candidate negative edges, using a local relational model as a proposal distribution. The complexity of each epoch of estimation is thus linear in the number of edges, making it possible to scale up to the 10^5 nodes in WordNet.[1] Second, we address the multi-relational nature of semantic graphs, by incorporating a combinatorial set of labeled motifs. Finally, we link graph-level relational features with distributional information, by combining the M3GM with a dyad-level model over word sense embeddings.

We train M3GM as a re-ranker, which we apply to a a strong local-feature baseline on the WN18RR dataset (Dettmers et al., 2018). This yields absolute improvements of 3-4 points on all commonly-used metrics. Model inspection reveals that M3GM assigns importance to features from all relations, and captures some interesting inter-relational properties that lend insight into the overall structure of WordNet.[2]

2 Related Work

Relational prediction in semantic graphs. Recent approaches to relation prediction in semantic graphs generally start by embedding the semantic concepts into a shared space and modeling relations by some operator that induces a score for an embedding pair input. We use several of these techniques as base models (Nickel et al., 2011; Bordes et al., 2013; Yang et al., 2014); detailed description of these methods is postponed to Section 3.2. Socher et al. (2013) generalize over the approach of Nickel et al. (2011) by using a bilinear tensor which assigns multiple parameters for each relation; Shi and Weninger (2017) project the node embeddings in a translational model similar to that of Bordes et al. (2013); Dettmers et al. (2018) apply a convolutional neural network by reshaping synset embeddings to 2-dimensional matrices. None of these embedding-based approaches incorporate structural information; in general, improvements in embedding-based methods are expected to be complementary to our approach.

---

[1] Although in principle the number of edges could grow quadratically with the number of nodes, Steyvers and Tenenbaum (2005) show that semantic graphs like WordNet tend to be very sparse, so that the number of observed edges grows roughly linearly with the number of nodes.

[2] Our code is available at `http://www.github.com/yuvalpinter/m3gm`.

Some recent works compose single edges into more intricate motifs, such as Guu et al. (2015), who define a task of **path prediction** and compose various functions to solve it. They find that compositionalized bilinear models perform best on WordNet. Minervini et al. (2017) train link-prediction models against an adversary that produces examples which violate structural constraints such as symmetry and transitivity. Another line of work builds on local neighborhoods of relation interactions and automatic detection of relations from syntactically parsed text (Riedel et al., 2013; Toutanova et al., 2015). Schlichtkrull et al. (2017) use Graph Convolutional Networks to predict relations while considering high-order neighborhood properties of the nodes in question. In general, these methods aggregate information over local neighborhoods, but do not explicitly model structural motifs.

Our model introduces interaction features between relations (e.g., hypernyms and meronyms) for the goal of relation prediction. To our knowledge, this is the first time that relation interaction is explicitly modeled into a relation prediction task. Within the ERGM framework, Lu et al. (2010) train a limited set of combinatory path features for social network link prediction.

Scaling exponential random graph models. The problem of approximating the denominator of the ERGM probability has been an active research topic for several decades. Two common approximation methods exist in the literature. In **Maximum Pseudolikelihood Estimation** (**MPLE**; Strauss and Ikeda, 1990), a graph's probability is decomposed into a product of the probability for each edge, which in turn is computed based on the ERGM feature difference between the graph excluding the edge and the full graph. **Monte Carlo Maximum Likelihood Estimation** (**MCMLE**; Snijders, 2002) follows a sampling logic, where a large number of graphs is randomly generated from the overall space under the intuition that the sum of their scores would give a good approximation for the total score mass. The probability for the observed graph is then estimated following normalization conditioned on the sampling distribution, and its precision increases as more samples are gathered. Recent work found that applying a parametric bootstrap can increase the reliability of MPLE, while retaining its superiority in training speed (Schmid and Desmarais, 2017). Despite this result, we opted for an MCMLE-based approach for M3GM, mainly due to the ability to keep the number of edges constant in each sampled graph. This property is important in our setup, since local edge scores added or removed to the overall graph score can occasionally dominate the objective function, giving unintended importance to the overall edge count.

3 Max-Margin Markov Graph Models

Consider a graph $G = (V, E)$, where V is a set of vertices and $E = \{(s_i, t_i)\}_{i=1}^{|E|}$ is a set of directed edges. The ERGM scoring function defines a probability over $\mathcal{G}_{|V|}$, the set of all graphs with $|V|$ nodes. This probability is defined as a log-linear function,

$$P_{\text{ERGM}}(G) \propto \psi_{\text{ERGM}}(G) = \exp\left(\boldsymbol{\theta}^T \mathbf{f}(G)\right), \quad (1)$$

where $\mathbf{f}$ is a feature function, from graphs to a vector of feature counts. Features are typically counts of motifs — small subgraph structures — as described in the introduction. The vector $\boldsymbol{\theta}$ is the parameter to estimate.

In this section we discuss our adaptation of this model to the domain of semantic graphs, leveraging their idiosyncratic properties. Semantic graphs are composed of multiple relation types, which the feature space needs to accommodate; their nodes are linguistic constructs (semantic concepts) associated with complex interpretations, which can benefit the graph representation through incorporating their embeddings in $\mathbb{R}^d$ into a new scoring model. We then present our M3GM framework to perform reliable and efficient parameter estimation on the new model.

3.1 Graph Motifs as Features

Based on common practice in ERGM feature extraction (e.g., Morris et al., 2008), we select the following graph features as a basis:

- Total edge count;
- Number of cycles of length k, for $k \in \{2, 3\}$;
- Number of nodes with exactly k outgoing (incoming) edges, for $k \in \{1, 2, 3\}$;
- Number of nodes with **at least** k outgoing (incoming) edges, for $k \in \{1, 2, 3\}$;
- Number of paths of length 2;

- Transitivity: the proportion of length-2 paths $u \rightarrow v \rightarrow w$ where an edge $u \rightarrow w$ also exists.

Semantic graphs are multigraphs, where multiple relationships (hypernymy, meronymy, derivation, etc.) are overlaid atop a common set of nodes. For each relation r in the relation inventory $\mathcal{R}$, we denote its edge set as E_r, and redefine $E = \bigcup_{r \in \mathcal{R}} E_r$, the union of all labeled edges. Some relations do not produce a connected graph, while others may coincide with each other frequently, possibly in regular but intricate patterns: for example, derivation relations tend to occur between synsets in the higher, more abstract levels of the hypernym graph. We represent this complexity by expanding the feature space to include relation-sensitive **combinatory motifs**. For each feature template from the basis list above, we extract features for all possible combinations of relation types existing in the graph. Depending on the feature type, these could be relation singletons, pairs, or triples; they may be order-sensitive or order-insensitive. For example:

- A combinatory 'transitivity' feature will be extracted for the proportion of paths $u \xrightarrow{hypernym} v \xrightarrow{meronym} w$ where an edge $u \xrightarrow{has\_part} w$ also exists.
- A combinatory '2-outgoing' feature will be extracted for the number of nodes with exactly one *derivation* and one *has_part*.

The number of features thus scales in $O(|\mathcal{R}|^K)$ for a feature basis which involves up to K edges in any feature, and so our 17 basis features (with $K = 3$) generate a combinatory feature set with roughly 3,000 features for the 11-relation version of WordNet used in our experiments (see Section 4.1).

3.2 Local Score Component

In classical ERGM application domains such as social media or biological networks, nodes tend to have little intrinsic distinction, or at least little meaningful intrinsic information that may be extracted prior to applying the model. In semantic graphs, however, the nodes represent synsets, which are associated with information that is both valuable to predicting the graph structure and approximable using unsupervised techniques such as embedding into a common d-dimensional vector space based on copious amounts of available data. We thus modify the traditional scoring function from eq. (1) to include node-specific information, by introducing a relation-specific **association operator** $\mathcal{A}^{(r)} : V \times V \rightarrow \mathbb{R}$:

$$\psi_{\text{ERGM+}}(G) = \\ = \exp\left(\boldsymbol{\theta}^T \mathbf{f}(G) + \sum_{r \in \mathcal{R}} \sum_{(s,t) \in E_r} \mathcal{A}^{(r)}(s,t)\right). \quad (2)$$

The association operator generalizes various models from the relation prediction literature:

TransE (Bordes et al., 2013) embeds each relation r into a vector in the shared space, representing a 'difference' between sources and targets, to compute the association score under a translational objective,

$$\mathcal{A}^{(r)}_{\text{TRANSE}}(s,t) = -\|\boldsymbol{e}_s + \boldsymbol{e}_r - \boldsymbol{e}_t\|.$$

BiLin (Nickel et al., 2011) embeds relations into full-rank matrices, computing the score by a bilinear multiplication,

$$\mathcal{A}^{(r)}_{\text{BILIN}}(s,t) = \boldsymbol{e}_s^T \mathbf{W}_r \boldsymbol{e}_t.$$

DistMult (Yang et al., 2014) is a special case of **BiLin** where the relation matrices are diagonal, reducing the computation to a ternary dot product,

$$\mathcal{A}^{(r)}_{\text{DISTMULT}}(s,t) = \langle \boldsymbol{e}_s, \boldsymbol{e}_r, \boldsymbol{e}_t \rangle = \sum_{i=1}^{d} e_{s_i}\, e_{r_i}\, e_{t_i}.$$

3.3 Parameter Estimation

The probabilistic formulation of ERGM requires the computation of a normalization term that sums over all possible graphs with a given number of nodes, $\mathcal{G}_N$. The set of such graphs grows at a rate that is super-exponential in the number of nodes, making exact computation intractable even for networks that are orders of magnitude smaller than semantic graphs like WordNet. One solution is to approximate probability using a variant of the Monte Carlo Maximum Likelihood Estimation (MCMLE) produce,

$$\log P(G) \approx \log \psi(G) - \log \frac{|\mathcal{G}_{|V|}|}{M} \sum_{\tilde{G} \sim \mathcal{G}_{|V|}}^{M} \psi(\tilde{G}), \quad (3)$$

where M is the number of networks $\tilde{G}$ sampled from $\mathcal{G}_{|V|}$, the space of all (multirelational) edge sets on nodes V. Each $\tilde{G}$ is referred to as a *negative sample*, and the goal of estimation is to assign low scores to these samples, in comparison with the score assigned to the observed network G.

Network samples can be obtained using edgewise negative sampling. For each edge $s \xrightarrow{r} t$ in the training network G, we remove it temporarily and consider T alternative edges, keeping the source s and relation r constant, and sampling a target $\tilde{t}$ from a **proposal distribution** Q. Every such substitution produces a new graph $\tilde{G}$,

$$\tilde{G} = G \cup \{s \xrightarrow{r} \tilde{t}\} \setminus \{s \xrightarrow{r} t\}. \quad (4)$$

Large-margin objective. Rather than approximating the log probability, as in MCMLE estimation, we propose a margin loss objective: the log score for each negative sample $\tilde{G}$ should be below the log score for G by a margin of at least 1. This motivates the hinge loss,

$$\mathcal{L}(\mathbf{\Theta}, \tilde{G}; G) = \Big(1 - \log \psi_{\text{ERGM+}}(G) + \log \psi_{\text{ERGM+}}(\tilde{G})\Big)_+, \quad (5)$$

where $(x)_+ = \max(0, x)$. Recall that the scoring function $\psi_{\text{ERGM+}}$ includes both the local association score for the alternative edge and the global graph features for the resulting graph. However, it is not necessary to recompute all association scores; we need only subtract the association score for the deleted edge $s \xrightarrow{r} t$, and add the association score for the sampled edge $s \xrightarrow{r} \tilde{t}$.

The overall loss function is the sum over $N = |E| \times T$ negative samples, $\{\tilde{G}^{(i)}\}_{i=1}^{N}$, plus an L_2 regularizer on the model parameters,

$$\mathcal{L}(\mathbf{\Theta}; G) = \lambda ||\mathbf{\Theta}||_2^2 + \sum_{i=1}^{N} \mathcal{L}(\mathbf{\Theta}, \tilde{G}^{(i)}). \quad (6)$$

Proposal distribution. The proposal distribution Q used to sample negative edges is defined to be proportional to the local association scores of edges not present in the training graph:

$$Q(\tilde{t} \mid s, r, G) \propto \begin{cases} 0 & s \xrightarrow{r} \tilde{t} \in G \\ \mathcal{A}^{(r)}(s, \tilde{t}) & s \xrightarrow{r} \tilde{t} \notin G \, . \end{cases} \quad (7)$$

By preferring edges that have high association scores, the negative sampler helps push the M3GM parameters away from likely false positives.

4 Relation Prediction

We evaluate M3GM on the relation graph edge prediction task.[3] Data for this task consists of a set of labeled edges, i.e. tuples of the form (s, r, t), where s and t denote source and target entities, respectively. Given an edge from an evaluation set, two prediction instances are created by hiding the source and target side, in turn. The predictor is then evaluated on its ability to predict the hidden entity, given the other entity and the relation type.[4]

4.1 WN18RR Dataset

A popular relation prediction dataset for WordNet is the subset curated as WN18 (Bordes et al., 2013, 2014), containing 18 relations for about 41,000 synsets extracted from WordNet 3.0. It has been noted that this dataset suffers from considerable leakage: edges from reciprocal relations such as hypernym / hyponym appear in one direction in the training set and in the opposite direction in dev / test (Socher et al., 2013; Dettmers et al., 2018). This allows trivial rule-based baselines to achieve high performance. To alleviate this concern, Dettmers et al. (2018) released the **WN18RR** set, removing seven relations altogether. However, even this dataset retains four symmetric relation types: *also see*, *derivationally related form*, *similar to*, and *verb group*. These symmetric relations can be exploited by defaulting to a simple rule-based predictor.

4.2 Metrics

We report the following metrics, common in ranking tasks and in relation prediction in particular: **MR**, the Mean Rank of the desired entity; **MRR**, Mean Reciprocal Rank, the main evaluation metric; and **H@**k, the proportion of Hits (true entities) found in the top k of the lists, for $k \in \{1, 10\}$. Unlike some prior work, we do not type-restrict the possible relation predictions (so, e.g., a *verb group* link may select a noun, and that would count against the model).

4.3 Systems

We evaluate a single-rule baseline, three association models, and two variants of the M3GM re-

---

[3] Sometimes referred to as Knowledge Base Completion, e.g. in Socher et al. (2013).

[4] We follow prior work in excluding the following from the ranked lists: the known entity (no self loops); entities from the training set which fit the instance; other entities in the evaluation set.

ranker trained on top of the best-performing association baseline.

4.3.1 RULE

We include a single-rule baseline that predicts a relation between s and t in the evaluation set if the same relation was encountered between t and s in the training set. All other models revert to this baseline for the four symmetric relations.

4.3.2 Association Models

The next group of systems compute local scores for entity-relation triplets. They all encode entities into embeddings $\boldsymbol{e}$. Each of these systems, in addition to being evaluated as a baseline, is also used for computing association scores in M3GM, both in the proposal distribution (see Section 3.3) and for creating lists to be re-ranked (see below): **TRANSE**, **BILIN**, **DISTMULT**. For detailed descriptions, see Section 3.2.

4.3.3 Max-Margin Markov Graph Model

The M3GM is applied as a re-ranker. For each relation and source (target), the top K candidate targets (sources) are retrieved based on the local association scores. Each candidate edge is introduced into the graph, and the score $\psi_{\text{ERGM+}}(G)$ is used to re-rank the top-K list.

We add a variant to this protocol where the graph score and association score are weighted by α and $1-\alpha$, repsectively, before being summed. We tune a separate α_r for each relation type, using the development set's mean reciprocal rank (MRR). These hyperparameter values offer further insight into where the M3GM signal benefits relation prediction most (see Section 6).

Since we do not apply the model to the symmetric relations (scored by the RULE baseline), they are excluded from the sampling protocol described in eq. (5), although their edges do contribute to the combinatory graph feature vector $\mathbf{f}$.

Our default setting backpropagates loss into only the graph weight vector $\boldsymbol{\theta}$. We experiment with a model variant which backpropagates into the association model and synset embeddings as well.

4.4 Synset Embeddings

For the association component of our model, we require embedding representations for WordNet synsets. While unsupervised word embedding techniques go a long way in representing wordforms (Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014), they are not immediately applicable to the semantically-precise domain of synsets. We explore two methods of transforming pre-trained word embeddings into synset embeddings.

Averaging. A straightforward way of using word embeddings to create synset embeddings is to collect the words representing the synset as surface form within the WordNet dataset and average their embeddings (Socher et al., 2013). We apply this method to pre-trained GloVe embeddings (Pennington et al., 2014) and pre-trained FastText embeddings (Bojanowski et al., 2017), averaging over the set of all wordforms in all lemmas for each synset, and performing a case-insensitive query on the embedding dictionary. For example, the synset '*determine.v.01*' lists the following lemmas: 'determine', 'find', 'find_out', 'ascertain'. Its vector is initialized as

$$\frac{1}{5}(\boldsymbol{e}_{determine} + 2 \cdot \boldsymbol{e}_{find} + \boldsymbol{e}_{out} + \boldsymbol{e}_{ascertain}).$$

AutoExtend retrofitting + Mimick. AutoExtend is a method developed specifically for embedding WordNet synsets (Rothe and Schütze, 2015), in which pre-trained word embeddings are retrofitted to the tripartite relation graph connecting wordforms, lemmas, and synsets. The resulting synset embeddings occupy the same space as the word embeddings. However, some WordNet senses are not represented in the underlying set of pre-trained word embeddings.[5] To handle these cases, we trained a character-based model called MIMICK, which learns to predict embeddings for out-of-vocabulary items based on their spellings (Pinter et al., 2017). We do not modify the spelling conventions of WordNet synsets before passing them to Mimick, so e.g. '*mask.n.02*' (the second synset corresponding to 'mask' as a noun) acts as the input character sequence as is.

Random initialization. In preliminary experiments, we attempted training the association models using randomly-initialized embeddings. These proved to be substantially weaker than distributionally-informed embeddings and we do not report their performance in the results section. We view this finding as strong evidence to support the necessity of a distributional signal in a type-level semantic setup.

---

[5] We use the out-of-the-box vectors supplied in `http://www.cis.lmu.de/~sascha/AutoExtend`.

	System	MR	MRR	H@10	H@1
	Rule	13396	35.26	35.27	35.23
1	DistMult	1111	43.29	50.73	39.67
2	BiLin	**738**	45.36	52.93	41.37
3	TransE	2231	**46.07**	**55.65**	**41.41**
4	M3GM	2231	47.94	**57.72**	43.26
5	M3GM$_{\alpha_r}$	2231	**48.30**	57.59	**43.78**

Table 1: Results on development set (all metrics except MR are x100). M3GM lines use TransE as their association model. In M3GM$_{\alpha_r}$, the graph component is tuned post-hoc against the local component per relation.

System	MR	MRR	H@10	H@1
Rule	13396	35.26	35.26	35.26
ComplEx[†]	5261	44	51	41
ConvE[†]	5277	46	48	39
ConvKB[†]	2554	24.8	52.5	
TransE	2195	46.59	55.55	42.26
M3GM$_{\alpha_r}$	**2193**	**49.83**	**59.02**	**45.37**

Table 2: Main results on test set. [†] These models were not re-implemented, and are reported as in Nguyen et al. (2018) and in Dettmers et al. (2018).

4.5 Setup

Following tuning experiments, we train the association models on synset embeddings with $d = 300$, using a negative log-likelihood loss function over 10 negative samples and iterating over symmetric relations once every five epochs. We optimize the loss using AdaGrad with $\eta = 0.01$, and perform early stopping based on the development set mean reciprocal rank. M3GM is trained in four epochs using AdaGrad with $\eta = 0.1$. We set M3GM's re-rank list size $K = 100$ and, following tuning, the regularization parameter $\lambda = 0.01$ and negative sample count per edge $T = 10$. Our models are all implemented in DyNet (Neubig et al., 2017).

5 Results

Table 1 presents the results on the development set. Lines 1-3 depict the results for local models using averaged FastText embedding initialization, showing that the best performance in terms of MRR and top-rank hits is achieved by TransE. Mean Rank does not align with the other metrics; this is an interpretable tradeoff, as both BiLin and DistMult have an inherent preference for correlated synset embeddings, giving a stronger fallback for cases where the relation embedding is completely off, but allowing less freedom for separating strong cases from correlated false positives, compared to a translational objective.

Effect of global score. There is a clear advantage to re-ranking the top local candidates using the score signal from the M3GM model (line 4). These results are further improved when the graph score is weighted against the association component per relation (line 5). We obtain similar improvements when re-ranking the predictions from DistMult and BiLin.

The M3GM training procedure is not useful in fine-tuning the association model via backpropagation: this degrades the association scores for true edges in the evaluation set, dragging the re-ranked results along with them to about a 2-point drop relative to the untuned variant.

Table 2 shows that our main results transfer onto the test set, with even a slightly larger margin. This could be the result of the greater edge density of the combined training and dev graphs, which enhance the global coherence of the graph structure captured by M3GM features. To support this theory, we tested the M3GM model trained on only the training set, and its test set performance was roughly one point worse on all metrics, as compared with the model trained on the training+dev data.

Synset embedding initialization. We trained association models initialized on AutoExtend+Mimick vectors (see Section 4.4). Their performance, inferior to averaged FastText vectors by about 1-2 MRR points on the dev set, is somewhat at odds with findings from previous experiments on WordNet (Guu et al., 2015). We believe the decisive factor in our result is the size of the training corpus used to create FastText embeddings, along with the increase in resulting vocabulary coverage. Out of 124,819 lemma tokens participating in 41,105 synsets, 118,051 had embeddings available (94.6%; type-level coverage 88.1%). Only 530 synsets (1.3%) finished this initialization process with no embedding and were assigned random vectors. AutoExtend, fit for embeddings from Mikolov et al. (2013) which were trained on a smaller corpus, offers a weaker signal: 13,377 synsets (32%) had no vector and needed Mimick initialization.

Positive	
1	$s \xrightarrow{member\_meronym} \textcircled{t}$
2	$s \xrightarrow{has\_part} \textcircled{t}$
3	$s \xrightarrow{hypernym} \textcircled{t} \xrightarrow{derivationally\_related\_form} u$
Negative	
4	$s \xrightarrow{hypernym} \textcircled{t}$
5	$\textcircled{s} \overset{hypernym}{\longleftrightarrow} \textcircled{t}$
6	$s \xrightarrow{member\_meronym} \textcircled{t} \xrightarrow{instance\_hypernym} u$
7	$s_1 \xrightarrow{has\_part} \textcircled{t} \xleftarrow{verb\_group} s_2$

Table 3: Select heavyweight features (motifs) following best dev set training using M3GM. Circled nodes count towards the motif.

Source	Relation	Correct target	Outranking local target(s)
indian lettuce	*hypernym*	herb	garden lettuce
austria	*has_part*	vienna	germany, hungary, france, european union

Table 4: Successful M3GM re-ranking examples.

Relation r	α_r	Relation r	α_r
mem. of domain usage	0.78	hypernym	0.64
mem. of domain region	0.77	domain topic of	0.38
member meronym	0.67	has part	0.33
instance hypernym	0.65		

Table 5: Graph score weights found for relations on the dev set. Zero means graph score is not considered at all for this relation, one means only it is considered.

6 Graph Analysis

As a consequence of the empirical experiment, we aim to find out what M3GM has learned about WordNet. Table 3 presents a sample of top-weighted motifs. Lines 1 and 2 demonstrate that the model prefers a broad scattering of targets for the *member_meronym* and *has_part* relations[6], which are flat and top-downwards hierarchical, respectively, while line 4 shows that a multitude of unique *hypernym*s is undesired, as expected from a bottom-upwards hierarchical relation. Line 5 enforces the asymmetry of the hypernym relation.

Lines 3, 6, and 7 hint at deeper interactions between the different relation types. Line 3 shows that the model assigns positive weights to hypernyms which have derivationally-related forms, suggesting that the derivational equivalence classes in the graph tend to exist in the higher, more abstract levels of the hypernym hierarchy, as noted in Section 3.1. Line 6 captures a semantic conflict: synsets located in the lower, specific levels of the graph can be specified either as *instances* of abstract concepts[7], or as *members* of less specific concrete classes, but not as both. Line 7 may have captured a nodal property – since *part_of* is a relation which holds between nouns, and *verb_group* holds between verbs, this negative weight assignment may be the manifestation of a part-of-speech uniqueness constraint. In addition, in features 3 and 7 we see the importance of symmetric relations (here *derivationally_related_form* and *verb_group*, respectively), which manage to be represented in the graph model despite not being directly trained on.

Table 4 presents examples of relation targets successfully re-ranked thanks to these features. The first false connection created a new unique hypernym, 'garden lettuce', downgraded by the graph score through incrementing the count of negatively-weighted feature 4. In the second case, 'vienna' was brought from rank 10 to rank 1 since it incremented the count for the positively-weighted feature 2, whereas all targets ranked above it by the local model were already *has_part*-s, mostly of 'europe'.

The α_r values weighing the importance of M3GM scores in the overall function, found per relation through grid search over the development set, are presented in Table 5. It appears that for all but two relations, the best-performing model preferred the signal from the graph features to that from the association model ($\alpha_r > 0.5$). Based on the surface properties of the different relation graphs, the decisive factor seems to be that *synset_domain_topic_of* and *has_part* pertain mostly to very common concepts, offering good local signal from the synset embeddings, whereas the rest include many long-tail, low-frequency synsets that require help from global features to detect regularity.

---

[6] Example edges: 'America' → 'American', 'face' → 'mouth', respectively.

[7] Example *instance_hypernym* edge: 'Rome' → 'national capital'.

7 Conclusion

This paper presents a novel method for reasoning about semantic graphs like WordNet, combining the distributional coherence between individual entity pairs with the structural coherence of network motifs. Applied as a re-ranker, this method substantially improves performance on link prediction. Our analysis of results from Table 3, lines 6 and 7, suggests that adding graph motifs which qualify their adjacent nodes in terms of syntactic function or semantic category may prove useful.

From a broader perspective, M3GM can do more as a probabilistic model than predict individual edges. For example, consider the problem of linking a new entity into a semantic graph, given only the vector embedding. This task involves adding multiple edges simultaneously, while maintaining structural coherence. Our model is capable of scoring bundles of new edges, and in future work, we plan to explore the possibility of combining M3GM with a search algorithm, to automatically extend existing knowledge graphs by linking in one or more new entities.

We also plan to explore multilingual applications. To some extent, the structural parameters estimated by M3GM are not specific to English: for example, hypernymy cannot be symmetric in any language. If the structural parameters estimated from English WordNet are transferable to other languages, then the combination of M3GM and multilingual word embeddings could facilitate the creation and extension of large-scale semantic resources across many languages (Fellbaum and Vossen, 2012; Bond and Foster, 2013; Lafourcade, 2007).

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. We discussed fast motif-counting algorithms with Polo Chau and Oded Green, and received early feedback from Jordan Boyd-Graber, Erica Briscoe, Martin Hyatt, Bryan Leslie Lee, Martha Palmer, and Oren Tsur. This research was funded by the Defense Threat Research Agency under award HDTRA1-15-1-0019.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Francis Bond and Ryan Foster. 2013. Linking and extending an open multilingual wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1352–1362.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

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

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Christiane Fellbaum and Piek Vossen. 2012. Challenges for a multilingual wordnet. *Language Resources and Evaluation*, 46(2):313–326.

Fan Guo, Steve Hanneke, Wenjie Fu, and Eric P Xing. 2007. Recovering temporally rewiring networks: A model-based approach. In *Proceedings of the 24th international conference on Machine learning*, pages 321–328. ACM.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327, Lisbon, Portugal. Association for Computational Linguistics.

Paul W. Holland and Samuel Leinhardt. 1981. An exponential family of probability distributions for directed graphs. *Journal of the american Statistical association*, 76(373):33–50.

Mathieu Lafourcade. 2007. Making people play for lexical acquisition with the jeuxdemots prototype. In *SNLP'07: 7th international symposium on natural language processing*, page 7.

Zhengdong Lu, Berkant Savas, Wei Tang, and Inderjit S. Dhillon. 2010. Supervised link prediction using multiple sources. In *2010 IEEE International Conference on Data Mining*, pages 923–928.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of International Conference on Learning Representations*.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2017. Adversarial sets for regularising neural link predictors. *arXiv preprint arXiv:1707.07596*.

Martina Morris, Mark S. Handcock, and David R. Hunter. 2008. Specification of exponential-family random graph models: terms and computational aspects. *Journal of statistical software*, 24(4):1548.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base inference. In *2015 aaai spring symposium series*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 327–333, New Orleans, Louisiana. Association for Computational Linguistics.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816.

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

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending Word Embeddings to Embeddings for Synsets and Lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China. Association for Computational Linguistics.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *stat*, 1050:17.

Christian S. Schmid and Bruce A. Desmarais. 2017. Exponential random graph models with big networks: Maximum pseudolikelihood estimation and the parametric bootstrap. *arXiv preprint arXiv:1708.02598*.

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *AAAI*, volume 17, pages 1236–1242.

Mariano Sigman and Guillermo A. Cecchi. 2002. Global organization of the wordnet lexicon. *Proceedings of the National Academy of Sciences*, 99(3):1742–1747.

Tom AB Snijders. 2002. Markov chain monte carlo estimation of exponential random graph models. *Journal of Social Structure*, 3(2):1–40.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26*.

Mark Steyvers and Joshua B. Tenenbaum. 2005. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive science*, 29(1):41–78.

David Strauss and Michael Ikeda. 1990. Pseudolikelihood estimation for social networks. *Journal of the American Statistical Association*, 85(409):204–212.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin markov networks. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on*

Continuous Vector Space Models and their Compositionality, pages 57–66, Beijing, China. Association for Computational Linguistics.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, volume 15, pages 1499–1509.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Learning Scalar Adjective Intensity from Paraphrases

Anne Cocos[⋆]
acocos@seas.upenn.edu

Veronica Wharton[⋆]
whartonv@seas.upenn.edu

Ellie Pavlick[†]
epavlick@brown.edu

Marianna Apidianaki[⋆◇]
marapi@seas.upenn.edu

Chris Callison-Burch[⋆]
ccb@upenn.edu

[⋆] Department of Computer and Information Science, University of Pennsylvania
[†] Department of Computer Science, Brown University
[◇] LIMSI, CNRS, Université Paris-Saclay, 91403 Orsay

Abstract

Adjectives like *warm*, *hot*, and *scalding* all describe temperature but differ in intensity. Understanding these differences between adjectives is a necessary part of reasoning about natural language. We propose a new paraphrase-based method to automatically learn the relative intensity relation that holds between a pair of scalar adjectives. Our approach analyzes over 36k adjectival pairs from the Paraphrase Database under the assumption that, for example, paraphrase pair *really hot* ↔ *scalding* suggests that *hot* < *scalding*. We show that combining this paraphrase evidence with existing, complementary pattern- and lexicon-based approaches improves the quality of systems for automatically ordering sets of scalar adjectives and inferring the polarity of indirect answers to *yes/no* questions.

1 Introduction

Semantically similar adjectives are not fully interchangeable in context. Although *hot* and *scalding* are related, the statement *"the coffee was hot"* does not imply the coffee was *scalding*. *Hot* and *scalding* are scalar adjectives that describe *temperature*, but they are not interchangeable because they vary in intensity. A native English speaker knows that their relative intensities are given by the ranking *hot* < *scalding*. Understanding this distinction is important for language understanding tasks such as sentiment analysis (Pang et al., 2008), question answering (de Marneffe et al., 2010), and textual inference (Dagan et al., 2006).

Existing lexical resources such as WordNet (Miller, 1995; Fellbaum, 1998) do not include the relative intensities of adjectives. As a result, there have been efforts to automate the process of learning intensity relations (e.g. Sheinman and Tokunaga (2009), de Melo and Bansal (2013), Wilkinson (2017), etc.). Many existing approaches rely on *pattern-based* or *lexicon-based* methods to predict the intensity ranking of adjectives. Pattern-based approaches search large corpora for lexical patterns that indicate an intensity relationship – for example, *"not just X, but Y"* implies $X < Y$. As with pattern-based approaches for other tasks (such as hypernym discovery (Hearst, 1992)), they are precise but have relatively sparse coverage of comparable adjectives, even when using web-scale corpora (de Melo and Bansal, 2013; Ruppenhofer et al., 2014). Lexicon-based approaches employ resources that map an adjective to a real-valued number that encodes both intensity and polarity (e.g. *good* might map to 1 and *phenomenal* to 5, while *bad* maps to -1 and *awful* to -3). They can also be precise, but may not cover all adjectives of interest.

particularly pleased	↔	*ecstatic*
quite limited	↔	*restricted*
rather odd	↔	*crazy*
so silly	↔	*dumb*
completely mad	↔	*crazy*

Figure 1: Examples of paraphrases from PPDB of the form $RB\ JJ_u \leftrightarrow JJ_v$ which can be used to infer pairwise intensity relationships ($JJ_u < JJ_v$).

We propose paraphrases as a new source of evidence for the relative intensity of scalar adjectives. A paraphrase is a pair of words or phrases with approximately similar meaning, such as *really great* ↔ *phenomenal*. Adjectival paraphrases can be exploited to uncover intensity relationships. A paraphrase pair of the above form, where one phrase is composed of an intensifying adverb and an adjective (*really great*) and the other is a single-word adjective (*phenomenal*), provides evidence that *great* < *phenomenal*. By drawing this evidence from large, automatically-generated

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1752–1762
Brussels, Belgium, October 31 - November 4, 2018.

paraphrase resources like the Paraphrase Database (PPDB) [1] (Ganitkevitch et al., 2013; Pavlick et al., 2015), it is possible to obtain high-coverage pairwise adjective intensity predictions at reasonably high accuracy.

We demonstrate the usefulness of paraphrase evidence for inferring relative adjective intensity in two tasks: ordering sets of adjectives along an intensity scale, and inferring the polarity of indirect answers to *yes/no* questions. In both cases, we find that combining the relatively noisy, but high-coverage, paraphrase evidence with more precise but low-coverage pattern- or lexicon-based evidence improves overall quality.

2 Related Work

Noting that adding adjective intensity relations to WordNet (Miller, 1995; Fellbaum, 1998) would be useful, Sheinman et al. (2013) propose a system for automatically extracting sets of same-attribute adjectives from WordNet 'dumbbells' – consisting of two direct antonyms at the poles and satellites of synonymous/related adjectives incident to each antonym (Gross and Miller, 1990) – and ordering them by intensity. The annotations, however, are not in WordNet as of its latest version (3.1).

Work on adjective intensity generally focuses on two related tasks: clustering adjectives based on the attributes they modify, and ranking same-attribute adjectives by intensity. With respect to the former, common approaches involve clustering adjectives by their contexts (Hatzivassiloglou and McKeown, 1993; Shivade et al., 2015). We do not focus on the clustering task in this paper, but concentrate on the ranking task.

Approaches to the task of ranking scalar adjectives by their intensity mostly fall under the paradigms of *pattern-based* or *lexicon-based* approaches. *Pattern-based* approaches work by extracting lexical (Sheinman and Tokunaga, 2009; de Melo and Bansal, 2013; Sheinman et al., 2013) or syntactic (Shivade et al., 2015) patterns indicative of an intensity relationship from large corpora. For example, the patterns "X, but not Y" and "not just X but Y" provide evidence that X is an adjective less intense than Y.

Lexicon-based approaches are derived from the premise that adjectives can provide information about the sentiment of a text (Hatzivassiloglou and McKeown, 1993). These methods draw upon a lexicon that maps adjectives to real-valued scores encoding both sentiment polarity and intensity. The lexicon might be compiled automatically – for example, from analyzing adjectives' appearance in star-valued product or movie reviews (de Marneffe et al., 2010; Rill et al., 2012; Sharma et al., 2015; Ruppenhofer et al., 2014) – or manually. In our experiments we utilize the manually-compiled SO-CAL lexicon (Taboada et al., 2011).

Our paraphrase-based approach to inferring relative adjective intensity is based on paraphrases that combine adjectives with adverbial modifiers. A tangentially related approach is Collex (Ruppenhofer et al., 2014), which is motivated by the intuition that adjectives with extreme intensities are modified by different adverbs from adjectives with more moderate intensities: extreme adverbs like *absolutely* are more likely to modify extreme adjectives like *brilliant* than are moderate adverbs like *very*. Unlike Collex, which requires predetermined sets of 'end-of-scale' and 'normal' adverbial modifiers, our approach learns the identity and relative importance of intensifying adverbs.

Relative intensity is just one of several dimensions of gradable adjective semantics. In addition to intensity scales, a comprehensive model of scalar adjective semantics might also incorporate notions of intensity range (Morzycki, 2015), adjective class (Kamp and Partee, 1995), and scale membership according to meaning (Hatzivassiloglou and McKeown, 1993). In this paper we take the position that relative intensity is worth studying on its own because it is an important component of adjective semantics, usable directly for some NLP tasks such as sentiment analysis (Pang et al., 2008), and as part of a more comprehensive model for other tasks like question answering (de Marneffe et al., 2010).

3 Paraphrase-based Intensity Evidence

Adjectival paraphrases provide evidence about the relative intensity of adjectives. A paraphrase of the form $RB\ JJ_u \leftrightarrow JJ_v$ – where one phrase is comprised of an adjective modified by an intensifying adverb ($RB\ JJ_u$), and the other is a single-word adjective (JJ_v) – is evidence that the first adjective is less intense than the second ($JJ_u < JJ_v$). We propose a new method for encoding this evidence and using it to make pairwise adjective intensity predictions. First, a graph (JJGRAPH) is formed to represent over 36k adjectival paraphrases hav-

---

[1] `www.paraphrase.org`

Round 1

very	hard	↔	harder
kinda	hard	↔	harder
so	hard	↔	harder
pretty	hard	↔	harder

⇓

Round 2

very	*pleasant*	↔	*delightful*
kinda	*hard*	↔	*tricky*
so	*wonderful*	↔	*brilliant*
pretty	*simple*	↔	*plain*

⇓

Round 3

more	pleasant	↔	delightful
really	hard	↔	tricky
truly	wonderful	↔	brilliant
quite	simple	↔	plain

Figure 2: Bootstrapping process for identifying intensifying adverbs. The adverbs found in Rounds 1 and 3 are used to build intensifying edges in JJGRAPH.

ing the specified form. Next, data in the graph are used to make pairwise adjective intensity predictions.

3.1 Identifying Intensifying Adverbs

In JJGRAPH, nodes are adjectives, and each directed edge ($\text{JJ}_u \underset{\text{RB}}{\longrightarrow} \text{JJ}_v$) corresponds to an adjectival paraphrase of the form (*RB* $JJ_u \leftrightarrow JJ_v$) – for example, (*very tall* ↔ *large*) – where one 'phrase' (JJ_v) is an adjective and the other (RB JJ_u) is an adjectival phrase containing an adverb and adjective (see Figure 1 for examples).

Adverbs in PPDB can be intensifying or de-intensifying. An *intensifying* adverb (e.g. *very*, *totally*) strengthens the adjectives it modifies. In contrast, a *de-intensifying* adverb (e.g. *slightly*, *somewhat*) weakens the adjectives it modifies. Since edges in JJGRAPH ideally point in the direction of increasing intensity, the first step in the process of creating JJGRAPH is to identify a set of adverbs that are likely intensifiers to be included as edges.

For this purpose, we generate a set R of likely intensifying adverbs within PPDB using a bootstrapping approach (Figure 2). The process starts with a small seed set of adjective pairs having a known intensity relationship. The seeds are pairs (j_u, j_v) from PPDB-XXL[2] such that j_u is a base-form adjective (e.g. *hard*), and j_v is its **comparative** or **superlative** form (e.g. *harder* or *hardest*). Using the seeds, we identify intensifying adverbs by finding adjectival paraphrases in PPDB of the form $(r_i j_u \leftrightarrow j_v)$; because $j_u < j_v$, adverb r_i is inferred to be intensifying (Round 1). All such r_i are added to initial adverb set R_1. The process continues by extracting paraphrases $(r_i j_{u'} \leftrightarrow j_{v'})$ with $r_i \in R_1$, indicating additional adjective pairs $(j_{u'}, j_{v'})$ with intensity direction inferred by r_i (Round 2). Finally, the adjective pairs extracted in this second iteration are used to identify additional intensifying adverbs R_3, which are added to the final set $R = R_1 \cup R_3$ (Round 3).

In all, this process generates a set of 610 adverbs. Examination of the set shows that the process does capture many intensifying adverbs like *very* and *abundantly*, and excludes many de-intensifying adverbs appearing in PPDB like *far less* and *not as*. However, due to the noise inherent in PPDB itself and in the bootstrapping process, there are also a few de-intensifying adverbs included in R (e.g. *hardly*, *kind of*) as well as adverbs that are neither intensifying nor de-intensifying (e.g. *ecologically*). It will be important to take this noise into consideration when using JJGRAPH to make pairwise intensity predictions.

3.2 Building JJGRAPH

JJGRAPH is built by extracting all 36,756 adjectival paraphrases in PPDB of the specified form, where the adverb belongs to R. The resulting graph has 3,704 unique adjective nodes. JJGRAPH is a multigraph, as there are frequently multiple intensifying relationships between pairs of adjectives. For example, the paraphrases (*pretty hard* ↔ *tricky*) and (*really hard* ↔ *tricky*) are both present in PPDB. There can also be contradictory or cyclic edges in JJGRAPH, as in the example depicted in the JJGRAPH subgraph in Figure 3, where the adverb *really* connects *tasty* to *lovely* and vice versa. Self-edges are allowed (e.g. (*really hard* ↔ *hard*)).

3.3 Pairwise Intensity Prediction

Examining the directed adverb edges between two adjectives j_u and j_v in JJGRAPH provides evidence about the relative intensity relationship between them. However, it has just been noted that JJGRAPH is noisy, containing both contradictory/cyclic edges and adverbs that are not uniformly intensifying. Rather than try to eliminate cycles, or manually annotate each adverb with a weight corresponding to its intensity and polarity

---

[2] PPDB comes in six increasingly large sizes from S to XXXL; larger collections have wider coverage but lower precision. Our work uses XXL.

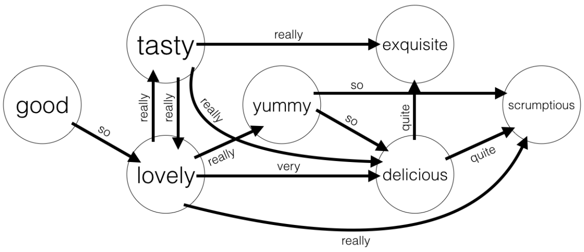

Figure 3: A subgraph of JJGRAPH, depicting its directed graph structure.

(Ruppenhofer et al., 2015; Taboada et al., 2011), we aim to learn these weights automatically in the process of predicting pairwise intensity.

Given adjectives (j_u, j_v), we build a classifier that outputs a score from 0 to 1 indicating the predicted likelihood that $j_u < j_v$. Its binary features correspond to adverb edges from j_u to j_v and from j_v to j_u in JJGRAPH. The feature space includes only adverbs from R that appear at least 10 times in JJGRAPH, resulting in features for $m = 259$ unique adverbs in each direction (i.e. from j_u to j_v and vice versa) for $2m = 518$ binary features total. Note that while all adverb features correspond to predicted intensifiers from R, there are some features that are actually de-intensifying due to the noise inherent in the bootstrapping process (Section 3.1).

We train the classifier on all 36.7k edges in JJGRAPH, based on a simplifying assumption that all adverbs in R are indeed intensifiers. For each adjective pair (j_u, j_v) with one or more direct edges from j_u to j_v, a positive training instance for pair (j_u, j_v) and a negative training instance for pair (j_v, j_u) are added to the training set. A logistic regression classifier is trained on the data, using elastic net regularization and 10-fold cross validation to tune parameters.

The model parameters output by the training process are in a feature weights vector $w \in \mathbb{R}^{2m}$ (with no bias term) which can be used to generate a paraphrase-based score for each adjective pair:

$$score_{\text{pp}}(j_u, j_v) = \frac{1}{1 + \exp^{-wx_{uv}}} - 0.5 \qquad (1)$$

where x_{uv} is the binary feature vector for adjective pair (j_u, j_v). The decision boundary 0.5 is subtracted from the sigmoid activation function so that pairs predicted to have the directed relation $j_u < j_v$ will have a positive score, and those predicted to have the opposite directional relation will have a negative score. Pairs with no connecting edges in JJGRAPH score 0.

4 Other Intensity Evidence

Our experiments compare the proposed paraphrase approach with existing pattern- and lexicon-based approaches.

4.1 Pattern-based Evidence

We experiment with the pattern-based approach of de Melo and Bansal (2013). Given a pair of adjectives to be ranked by their intensity, de Melo and Bansal (2013) cull intensity patterns from Google *n*-Grams (Brants and Franz, 2009) as evidence of their intensity order. Specifically, they identify 8 types of *weak-strong* patterns (e.g. *“X, but not Y”*) and 7 types of *strong-weak* patterns (e.g. *“not X, but still Y”*) that are used as evidence about the directionality of the intensity relationship between adjectives. Given an adjective pair (j_u, j_v), an overall pattern-based *weak-strong* score is calculated:

$$score_{\text{pat}}(j_u, j_v) = \frac{(W_u - S_u) - (W_v - S_v)}{\text{count}(j_u) \cdot \text{count}(j_v)} \qquad (2)$$

where W_u and S_u quantify the pattern evidence for the *weak-strong* and *strong-weak* intensity relations respectively for the pair (j_u, j_v), and W_v and S_v quantify the pattern evidence for the pair (j_v, j_u). W_u and S_u are calculated as:

$$\begin{aligned} W_u &= \frac{1}{P_1} \sum_{p_1 \in P_{ws}} \text{count}(p_1(j_u, j_v)) \\ S_u &= \frac{1}{P_2} \sum_{p_2 \in P_{sw}} \text{count}(p_2(j_u, j_v)) \end{aligned} \qquad (3)$$

W_v and S_v are calculated similarly by swapping the places of j_u and j_v. For example, given pair (*good*, *great*), W_u might incorporate evidence from patterns *“good, but not great”* and *“not only good but great”*, while S_v might incorporate evidence from the pattern *“not great, just good”*. P_{ws} denotes the set of *weak-strong* patterns, P_{sw} denotes the set of *strong-weak* patterns, and P_1 and P_2 give the total counts of all occurrences of any pattern in P_{ws} and P_{sw} respectively. The score is normalized by the frequencies of j_u and j_v in order to avoid bias due to high-frequency adjectives. As with the paraphrase-based scoring mechanism (Equation 1), scores output by this method can be positive or negative, with positive scores being indicative of a *weak-strong* relationship from j_u to j_v. Note that $score(j_u, j_v) = -score(j_v, j_u)$.

4.2 Lexicon-based Evidence

We use the manually-compiled SO-CAL[3] lexicon as our third, lexicon-based method for inferring intensity. The SO-CAL lexicon assigns an integer weight in the range $[-5, 5]$ to 2,826 adjectives. The sign of the weight encodes sentiment polarity (positive or negative), and the value encodes intensity (e.g. *atrocious*, with a weight of -5, is more intense than *unlikable*, with a weight of -3). SO-CAL is used to derive a pairwise intensity prediction for adjectives (j_u,j_v) as follows:

$$score_{\text{socal}}(j_u, j_v) = |L(j_v)| - |L(j_u)|, \quad \text{iff sign}(j_u) = \text{sign}(j_v) \tag{4}$$

where $L(j_v)$ gives the lexicon weight for j_v. Note that $score_{socal}$ is computed only for adjectives having the same polarity direction in the lexicon; otherwise the score is undefined. This is because adjectives belonging to different half scales, such as *freezing* and *steaming*, are frequently incomparable in terms of intensity (de Marneffe et al., 2010).

4.3 Combining Evidence

While the pattern-based and lexicon-based pairwise intensity scores are known to be precise but low-coverage (de Melo and Bansal, 2013; Ruppenhofer et al., 2015), we expect that the paraphrase-based score will produce higher coverage at lower accuracy. Thus we also experiment with scoring methods that combine two or three score types. When combining two metrics **x** and **y** to generate a score for a pair (j_u, j_v), we simply use the first metric **x** if it can be reliably calculated for the pair, and back off to metric **y** otherwise. More formally, the combined score for metrics **x** and **y** is given by:

$$\begin{aligned} score_{\text{x+y}}(j_u, j_v) &= \alpha_{\text{x}} \cdot g_{\text{x}}(score_{\text{x}}(j_u, j_v)) \\ &+ (1 - \alpha_{\text{x}}) \cdot g_{\text{y}}(score_{\text{y}}(j_u, j_v)) \end{aligned} \tag{5}$$

where $\alpha_{\text{x}} \in \{0, 1\}$ is a binary indicator corresponding to the condition that $score_{\text{x}}$ can be reliably calculated for the adjective pair, and $g_{\text{x}}(\cdot)$ is a scaling function (see below). If $\alpha_{\text{x}} = 1$, then $score_{\text{x}}$ is used. Otherwise, if $\alpha_{\text{x}} = 0$, then we default to $score_{\text{y}}$. When combining three metrics x, y, and z, the combined score is given by:

$$\begin{aligned} score_{\text{x+y+z}}(j_u, j_v) &= \alpha_{\text{x}} \cdot g_{\text{x}}(score_{\text{x}}(j_u, j_v)) \\ &+ (1 - \alpha_{\text{x}}) \cdot score_{\text{y+z}}(j_u, j_v) \end{aligned} \tag{6}$$

The criteria for having $\alpha_{\text{x}} = 1$ varies depending on the metric type. For pattern-based evidence (**x**='pat'), $\alpha_{\text{x}} = 1$ when adjectives j_u and j_v appear together in any of the intensity patterns culled from Google *n*-grams (e.g. "j_u, but not j_v" exists). For lexicon-based evidence (**x**='socal'), $\alpha_{\text{x}} = 1$ when both j_u and j_v are in the SO-CAL vocabulary, and have the same polarity (i.e. are both positive or both negative). For paraphrase-based evidence (**x**='pp'), $\alpha_{\text{x}} = 1$ when j_u and j_v have one or more edges directly connecting them in JJGRAPH.

Since the metrics to be combined may have different ranges, we use a scaling function $g_{\text{x}}(\cdot)$ to make the scores output by each metric directly comparable:

$$g_{\text{x}}(w) = \text{sign}(w) \cdot \left(\frac{\log(|w|) - \mu_{\text{x}}}{\sigma_{\text{x}}} + \gamma_{\text{x}} \right) \tag{7}$$

where μ_{x} and σ_{x} are the estimated population mean and standard deviation of $\log(score_{\text{x}})$ (estimated over all adjective pairs in the dataset), and γ_{x} is an offset equal to at least $5\sigma_{\text{x}}$.

5 Ranking Adjective Sets by Intensity

The first experimental application for the different paraphrase evidence is an existing model for predicting a global intensity ordering within a set of adjectives. Global ranking models are useful for inferring intensity comparisons between adjectives for which there is no explicit evidence. For example, in ranking three adjectives like *warm*, *hot*, and *scalding*, there may be direct evidence indicating *warm* < *hot* and *hot* < *scalding*, but no way of directly comparing *warm* to *scalding*. Global ranking models infer that *warm* < *scalding* based on evidence from the other adjective pairs in the scale.

5.1 Global Ranking Model

We adopt the mixed-integer linear programming (MILP) approach of de Melo and Bansal (2013) for generating a global intensity ranking. This model takes a set of adjectives $A = \{a_1, \ldots, a_n\}$

---

[3] `https://github.com/sfu-discourse-lab/SO-CAL`

Dataset	# of Scales	Min/Max/Mean Scale Size	# of Unordered (unequal) Pairs	Example Scale
deMelo	87	3 / 8 / 3.90	524 (466)	$\{\text{clean}\} < \{\text{spotless, immaculate}\}$
Crowd	79	2 / 8 / 3.18	293 (250)	$\{\text{low}\} < \{\text{limited}\} < \{\text{scarce}\}$
Wilkinson	21	2 / 5 / 2.81	61 (61)	$\{\text{dry}\} < \{\text{arid}\} < \{\text{parched}\}$

Table 1: Characteristics of the scalar adjective datasets used for evaluation. The deMelo scale example shows an instance of an equally-intense pair (*spotless*, *immaculate*).

and directed, pairwise adjective intensity scores $score(a_i, a_j)$ as input, and assigns each adjective a_i a place along a linear scale $x_i \in [0, 1]$. The adjectives' assigned values define the global ordering. If the predicted weights used as input are inconsistent, containing cycles, the model resolves these by choosing the globally optimal solution.

Recall that all pairwise scoring metrics produce a positive score for adjective pair (j_u, j_v) when it is likely that $j_u < j_v$, and a negative score otherwise. Consequently, the MILP approach should result in $x_u < x_v$ when $score(j_u, j_v)$ is positive, and $x_u > x_v$ otherwise. This goal is achieved by maximizing the objective function:

$$\sum_{u,v} \text{sign}(x_v - x_u) \cdot score(j_u, j_v) \tag{8}$$

de Melo and Bansal (2013) propose a MILP formulation for maximizing this objective, which we utilize in our experiments. Note that while de Melo and Bansal (2013) incorporate synonymy evidence from WordNet in their ranking method, we do not implement this part of the model.

5.2 Experiments

We experiment with using each of the paraphrase-, pattern-, and lexicon-based pairwise scores as input to the global ranking model in isolation. To examine how the scoring methods perform when used in combination, we also test all possible combinations of 2 and 3 scores.

Experiments are run over three distinct test sets (Table 1). Each dataset contains ordered sets of scalar adjectives belonging to the same *scale*. In general, scalar adjectives describing the same attribute can be ordered along a full scale (e.g. *freezing* to *sweltering*), or a half scale (*warm* to *sweltering*); all three test sets group adjectives into half scales. The three datasets are described here, and their characteristics are given in Table 1.

deMelo (de Melo and Bansal, 2013)[4]. 87 adjective sets are extracted from WordNet 'dumbbell' structures (Gross and Miller, 1990), and partitioned into half-scale sets based on their pattern-based evidence in the Google N-Grams corpus (Brants and Franz, 2009). Sets are manually annotated for intensity relations (<, >, and =).

Wilkinson (Wilkinson and Oates, 2016). Twelve adjective sets are generated by presenting crowd workers with small seed sets (e.g. *huge, small, microscopic*), and eliciting similar adjectives. Sets are automatically cleaned for consistency, and then annotated for intensity by crowd workers. While the original dataset contains full scales, we manually sub-divide these into 21 half-scales for use in this study. Details on the modification from full- to half-scales are in the Supplemental Material.

Crowd. We also crowdsourced a new set of adjective scales with high coverage of the PPDB vocabulary. In a three-step process, we first asked crowd workers whether pairs of adjectives describe the same attribute (e.g. temperature) and therefore should belong along the same scale. Second, sets of same-scale adjectives were refined over multiple rounds. Finally, workers ranked the adjectives in each set by intensity. The final dataset includes 293 adjective pairs along 79 scales.

We measure the agreement between the gold standard ranking of adjectives along each scale and the predicted ranking using three commonly-used metrics:

Pairwise accuracy. For each pair of adjectives along the same scale, we compare the predicted ordering of the pair after global ranking (<, >, or =) to the gold-standard ordering of the pair, and report overall accuracy of the pairwise predictions.

Kendall's tau (τ_b). This metric computes the rank correlation between the predicted ($r_P(J)$) and gold-standard ($r_G(J)$) ranking permutations of each adjective scale J, incorporating a correction for ties. Values for τ_b range from -1 to 1, with extreme values indicating a perfect negative

---

[4] `http://demelo.org/gdm/intensity/`

		Score Accuracy (before ranking)		Global Ranking Results			
Test Set	Score Type	Coverage	Pairwise Acc.	Pairwise Acc.	Avg. τ_b	ρ	Example Predicted Scale
deMelo	$score_{\text{pat}}$	0.48	**0.844**	0.650	**0.633**	**0.583**	$\{$clean$\}$ < $\{$spotless, immaculate$\}$*
	$score_{\text{pp}}$	0.33	0.458	0.307	0.071	0.090	$\{$immaculate, clean$\}$ < $\{$spotless$\}$
	$score_{\text{socal}}$	0.28	0.546	0.246	0.110	0.019	$\{$clean$\}$ < $\{$spotless$\}$ < $\{$immaculate$\}$
	$score_{\text{pat+socal}}$	0.61	0.757	**0.653**	0.609	0.533	$\{$clean$\}$ < $\{$spotless$\}$ < $\{$immaculate$\}$
	$score_{\text{pat+socal+pp}}$	**0.70**	0.722	0.644	0.564	0.482	$\{$clean$\}$ < $\{$spotless$\}$ < $\{$immaculate$\}$
Crowd	$score_{\text{pat}}$	0.11	**0.784**	0.321	0.203	0.221	$\{$limited, low, scarce$\}$
	$score_{\text{pp}}$	0.74	0.676	$0.597^{\dagger\dagger}$	$0.437^{\dagger}$	0.405	$\{$low$\}$ < $\{$limited$\}$ < $\{$scarce$\}$*
	$score_{\text{socal}}$	0.35	0.757	0.421	0.342	0.293	$\{$limited, low, scarce$\}$
	$score_{\text{socal+pp}}$	0.81	0.687	$0.621^{\dagger\dagger}$	$0.470^{\dagger\dagger}$	0.465	$\{$low$\}$ < $\{$limited$\}$ < $\{$scarce$\}$*
	$score_{\text{socal+pat+pp}}$	**0.82**	0.694	**0.639**††	**0.495**††	**0.480**	$\{$low$\}$ < $\{$limited$\}$ < $\{$scarce$\}$*
Wilkinson	$score_{\text{pat}}$	0.44	0.852	0.475	0.441	0.435	$\{$quick$\}$ < $\{$speedy, fast$\}$
	$score_{\text{pp}}$	0.80	0.753	0.639	0.419	0.450	$\{$quick$\}$ < $\{$fast$\}$ < $\{$speedy$\}$*
	$score_{\text{socal}}$	0.31	**0.895**	0.312	0.317	0.422	$\{$fast$\}$ < $\{$speedy$\}$ < $\{$quick$\}$
	$score_{\text{pat+pp}}$	**0.89**	0.833	$0.738^{\dagger\dagger}$	0.605	0.564	$\{$quick$\}$ < $\{$fast$\}$ < $\{$speedy$\}$*
	$score_{\text{pat+socal+pp}}$	**0.89**	0.833	**0.754**††	**0.638**	**0.611**	$\{$quick$\}$ < $\{$fast$\}$ < $\{$speedy$\}$*

††: $p \leq .01$ †: $p \leq .05$

Table 2: Pairwise relation prediction and global ranking results for each score type in isolation, and for the best-scoring combinations of 2 and 3 score types on each dataset. For the global ranking accuracy and average τ_b results, we denote with the † symbol scores for metrics incorporating paraphrase-based evidence that significantly out-perform both $score_{\text{pat}}$ and $score_{\text{socal}}$ under the paired Student's t-test, using the Anderson-Darling test to confirm that scores conform to a normal distribution (Fisher, 1935; Anderson and Darling, 1954; Dror et al., 2018). Example output is also given, with correct rankings starred.

or positive correlation, and a value of 0 indicating no correlation between predicted and gold rankings. We report τ_b as a weighted average over scales in each dataset, where weights correspond to the number of adjective pairs in each scale.

Spearman's rho (ρ). We report the Spearman's ρ rank correlation coefficient between predicted ($r_P(J)$) and gold-standard ($r_G(J)$) ranking permutations. For each dataset, we calculate this metric just once by treating each adjective in a particular scale as a single data point, and calculating an overall ρ for all adjectives from all scales.

5.3 Experimental Results

The results of the global ordering experiment, reported in Table 2, are organized as follows: *Score Accuracy* pertains to performance of the scoring methods alone – prior to global ranking – while *Global Ranking Results* pertains to performance of each scoring method as used in the global ranking algorithm. Within *Score Accuracy* there are two metrics. *Coverage* gives the percent of unique same-scale adjective pairs from the test set that can be directly scored using the given method. For $score_{\text{pat}}$, covered pairs are all those that appear together in any recognized pattern; for $score_{\text{pp}}$, covered pairs are those directly connected in JJGRAPH by one or more direct edges; for $score_{\text{socal}}$, covered pairs are all those for which both adjectives are in the SO-CAL lexicon and the metric is defined. *Pairwise Accuracy* gives the accuracy of the scoring method (before global ranking) on *just the covered pairs*, meaning that the subset of pairs scored by each method varies. Within *Global Ranking Results*, we report pairwise accuracy, weighted average τ_b, and ρ calculated over *all pairs* after ranking – including both pairs that are covered by the scoring method, and those whose pairwise intensity relationship has been inferred by the ranking algorithm.

The results indicate that the pairwise score accuracies (before ranking) for $score_{\text{pat}}$ and $score_{\text{socal}}$ are higher than those of $score_{\text{pp}}$ for all datasets, but that their coverage is relatively limited. The one exception is the deMelo dataset, where $score_{\text{pat}}$ has high coverage because the dataset was compiled specifically by finding adjective pairs that matched lexical patterns in the corpus. For all datasets, highest coverage is achieved using one of the combined metrics that incorporates paraphrase-based evidence.

The impact of these trends is visible on the

Global Ranking Results. When using pairwise intensity scores to compute the global ranking, higher coverage by a metric drives better results, as long as the metric's accuracy is reasonably high. Thus the paraphrase-based $score_{\text{pp}}$, with its high coverage, gets better global ranking results than the other single-method scores for two of the three datasets. Further, we find that boosting coverage with a combined metric that incorporates paraphrase evidence produces the highest post-ranking pairwise accuracy scores overall for all three datasets, and the highest average τ_b and ρ on the Crowd and Wilkinson datasets. We conclude that incorporating paraphrase evidence can improve the quality of this model for ordering adjectives along a scale because it gives high coverage with reasonably high quality.

The performance trends on the deMelo dataset differ from those on the Crowd and Wilkinson datasets. In particular, $score_{\text{pp}}$ and $score_{\text{socal}}$ have substantially lower pre-ranking pairwise accuracy on the pairs they cover in the deMelo dataset than they do for Crowd and Wilkinson: $score_{\text{pp}}$ has an accuracy of just 0.458 on covered pairs in the deMelo dataset, compared with 0.676 and 0.753 on the Crowd and Wilkinson datasets, and score differences for $score_{\text{socal}}$ are similar. The near-random prediction accuracies of $score_{\text{pp}}$ and $score_{\text{socal}}$ on deMelo before ranking lead to near-zero correlation values on this dataset after global ranking. To explore possible reasons for these results, we assessed the level of human agreement with each dataset in terms of pairwise accuracy. For each test set, we asked five crowd workers to classify the intensity direction for each adjective pair (j_u, j_v) in all scales as less than (<), greater than (>), or equal (=). We found that humans agreed with the 'gold standard' direction 65% of the time on the Bansal dataset, versus 70% of the time on the Crowd and Wilkinson datasets. It is possible that the more difficult nature of the Bansal dataset, coupled with its method of compilation (i.e. favoring adjective pairs that co-occur with pre-defined intensity patterns), lead to the lower coverage and lower accuracy of $score_{\text{pp}}$ and $score_{\text{socal}}$ on this dataset.

6 Indirect Question Answering

The second task that we address is answering indirect *yes* or *no* questions. de Marneffe et al. (2010) observed that answers to such polar questions frequently omit an explicit *yes* or *no* response. In some cases the implied answer depends on the relative intensity of adjective modifiers in the question and answer. For example, in the exchange:

Q: *Was he a successful ruler?*
A: *Oh, a tremendous ruler.*

the implied answer is *yes*, which is inferred because *successful* $\leq$ *tremendous* in terms of relative intensity. Conversely, in the exchange:

Q: *Does it have a large impact?*
A: *It has a medium-sized impact.*

the implied answer is *no* because *large* > *medium-sized.*

de Marneffe et al. (2010) compiled an evaluation set for this task by extracting 123 examples of such indirect question-answer pairs (IQAP) from dialogue corpora. In each exchange, the implied answer (annotated by crowd workers to be *yes* or *no*[5]) depends on the relative intensity relationship between modifiers in the question and answer texts. In their original paper, the authors utilize an automatically-compiled lexicon to make a polarity prediction for each IQAP.

6.1 Predicting Answer Polarity

Our goal is to see whether paraphrase-based scores are useful for predicting the polarity of answers in the IQAP dataset. As before, we compare the quality of predictions made using the paraphrase-based evidence with predictions made using pattern-based, lexicon-based, and combined scoring metrics.

To use the pairwise scores for inference, we employ a decision procedure nearly identical to that of de Marneffe et al. (2010). If j_q and j_a are scorable (i.e. have a scorable intensity relationship along the same half-scale), then $j_q \leq j_a$ implies the answer is *yes* (first example above), and $j_q > j_a$ implies the answer is *no* (second example). If the pair of adjectives is not scorable, then the predicted answer is *no*, as the pair could be antonyms or completely unrelated. If either j_q or j_a is missing from the scoring vocabulary, the adjectives are impossible to compare and therefore the prediction is *uncertain*. The full decision procedure is given in Figure 4.

---

[5] The original dataset contains two additional examples where the answer is annotated as *uncertain*, but de Marneffe et al. (2010) exclude them from the results and so do we.

Given: A dialogue exchange consisting of a polar question and answer, where the answer depends on the relative intensities of distinct modifiers j_q and j_a in the question and answer respectively:

1. if j_q or j_a are missing from the score vocabulary, predict "UNCERTAIN"
2. else, if $score(\mathrm{JJ}_q, \mathrm{JJ}_a)$ is undefined, predict "NO"
3. else, if $score(\mathrm{JJ}_q, \mathrm{JJ}_a) \geq 0$, predict "YES"
4. else, predict "NO"
5. If the question or answer contains negation, map a "YES" answer to "NO and a "NO" answer to "YES"

Figure 4: Decision procedure for using pairwise intensity scores for predicting polarity of an IQAP instance, based on de Marneffe et al. (2010).

6.2 Experiments

The decision procedure in Figure 4 is carried out for the 123 IQAP instances in the dataset, varying the score type. We report the accuracy, and macro-averaged precision, recall, and F1-score of the 85 *yes* and 38 *no* instances, in Table 3 alongside the percent of instances with adjectives out of vocabulary. Only the combined scores for the two best-scoring combinations, $score_{\text{socal+pp}}$ and $score_{\text{socal+pat+pp}}$, are reported.

Method	%OOV	Acc.	P	R	F
all-"YES"	.00	**.691**	.346	.500	.409
deMarneffe (2010)	.02	.610	.597	.594	.596
$score_{\text{socal}}$	.33	.504	**.710**	.481	.574
$score_{\text{pp}}$	.09	.496	.568	.533	.550
$score_{\text{pat}}$	.07	.407	.524	.491	.507
$score_{\text{socal+pp}}$	.09	.634	.690	.663	.676
$score_{\text{socal+pat+pp}}$	.06	.642	.684	**.683**	**.684**

Table 3: Accuracy and macro-averaged precision (P), recall (R), and F1-score (F) over *yes* and *no* responses on 123 question-answer pairs. The percent of pairs having one or both adjectives out of the score vocabulary is listed as %OOV.

The simplest baseline of predicting all answers to be "YES" gets highest accuracy in this imbalanced test set, but all score types perform better than the all-"YES" baseline in terms of precision and F1-score. Bouyed by its high precision, the $score_{\text{socal}}$ – which is derived from a manually-compiled lexicon – scored higher than $score_{\text{pp}}$ and $score_{\text{pat}}$. But it mis-predicted 33% of pairs as *uncertain* because of its limited overlap with the IQAP vocabulary. Meanwhile, $score_{pp}$ had relatively high coverage and a mid-level F-score, while $score_{pat}$ scored poorly on this dataset due to its sparsity; while all modifiers in the IQAP dataset are in the Google N-grams vocabulary, most do not have observed patterns and therefore return predictions of "NO" (item 2 in Figure 4). As in the global ranking experiments, the paraphrase-based evidence is complementary to the lexicon-based evidence, and thus the combined $score_{\text{socal+pp}}$ and $score_{\text{socal+pat+pp}}$ produce significantly better accuracy than any score in isolation (McNemar's test, $p < .01$), and also out-perform the original expected ranking method of de Marneffe et al. (2010) (although they do not beat the best-reported score on this dataset, F-score=0.706 (Kim and de Marneffe, 2013)).

7 Conclusion

We have proposed adjectival paraphrases as a new source of evidence for predicting intensity relationships between scalar adjectives. While paraphrase-based intensity evidence produces pairwise predictions that are less precise than those produced by pattern- or lexicon-based evidence, the coverage is substantially higher. Thus paraphrases can be successfully used as a complementary source of information for reasoning about adjective intensity. We will publish the JJGRAPH resource, datasets, and code upon publication.

Acknowledgments

This material is based in part on research sponsored by the following organizations: the Allen Institute for Artificial Intelligence (AI2) Key Scientific Challenges program, the Google Ph.D. Fellowship program, the French National Research Agency under project ANR-16-CE33-0013, and DARPA under grant numbers FA8750-13-2-0017 (the DEFT program) and HR0011-15-C-0115 (the LORELEI program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government.

We are grateful to our anonymous reviewers for their thoughtful and constructive comments.

References

Theodore W Anderson and Donald A Darling. 1954. A test of goodness of fit. *Journal of the American statistical association*, 49(268):765–769.

Thorsten Brants and Alex Franz. 2009. Web 1T 5-gram, 10 European languages version 1. *Linguistic Data Consortium, Philadelphia*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment: First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, chapter The PASCAL Recognising Textual Entailment Challenge. Springer Berlin Heidelberg, Berlin, Heidelberg.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1383–1392.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Ronald Aylmer Fisher. 1935. The design of experiments.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, pages 758–764, Atlanta, Georgia.

Derek Gross and Katherine J Miller. 1990. Adjectives in wordnet. *International Journal of lexicography*, 3(4):265–277.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1993. Towards the automatic identification of adjectival scales: Clustering adjectives according to meaning. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics (ACL)*, pages 172–182, Columbus, Ohio.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2 (COLING)*, pages 539–545, Nantes, France.

Hans Kamp and Barbara Partee. 1995. Prototype theory and compositionality. *Cognition*, 57(2):129–191.

Joo-Kyung Kim and Marie-Catherine de Marneffe. 2013. Deriving adjectival scales from continuous space word representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1625–1630, Seattle, Washington.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2010. Was it good? It was provocative. Learning the meaning of scalar adjectives. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 167–176, Uppsala, Sweden.

Gerard de Melo and Mohit Bansal. 2013. Good, great, excellent: Global inference of semantic intensities. *Transactions of the Association for Computational Linguistics*, 1:279–290.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Marcin Morzycki. 2015. *Modification*. Cambridge University Press.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL) (Volume 2: Short Papers)*, pages 425–430, Beijing, China.

Sven Rill, J. vom Scheidt, Johannes Drescher, Oliver Schütz, Dirk Reinel, and Florian Wogenstein. 2012. A generic approach to generate opinion lists of phrases for opinion mining applications. In *Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining (WISDOM)*, Beijing, China.

Josef Ruppenhofer, Jasper Brandes, Petra Steiner, and Michael Wiegand. 2015. Ordering adverbs by their scaling effect on adjective intensity. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 545–554, Hissar, Bulgaria.

Josef Ruppenhofer, Michael Wiegand, and Jasper Brandes. 2014. Comparing methods for deriving intensity scores for adjectives. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden.

Raksha Sharma, Mohit Gupta, Astha Agarwal, and Pushpak Bhattacharyya. 2015. Adjective intensity and sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods for Natural Language Processing (EMNLP)*, Lisbon, Portugal.

Vera Sheinman, Christiane Fellbaum, Isaac Julien, Peter Schulam, and Takenobu Tokunaga. 2013. Large, huge or gigantic? Identifying and encoding intensity relations among adjectives in WordNet. *Language resources and evaluation*, 47(3):797–816.

Vera Sheinman and Takenobu Tokunaga. 2009. Adjscales: Visualizing differences between adjectives for language learners. *IEICE TRANSACTIONS on Information and Systems*, 92(8):1542–1550.

Chaitanya P. Shivade, Marie-Catherine de Marneffe, Eric Fosler-Lussier, and Albert M. Lai. 2015. Corpus-based discovery of semantic intensity scales. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL HLT)*, pages 483–493, Denver, Colorado.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Bryan Wilkinson. 2017. *Identifying and Ordering Scalar Adjectives Using Lexical Substitution*. Ph.D. thesis, University of Maryland, Baltimore County.

Bryan Wilkinson and Tim Oates. 2016. A gold standard for scalar adjectives. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, Portoro, Slovenia.

Pointwise HSIC: A Linear-Time Kernelized Co-occurrence Norm for Sparse Linguistic Expressions

Sho Yokoi [1,2] **Sosuke Kobayashi** [3] **Kenji Fukumizu** [4] **Jun Suzuki** [1,2] **Kentaro Inui** [1,2]
[1] Tohoku University [2] RIKEN Center for Advanced Intelligence Project
{yokoi,jun.suzuki,inui}@ecei.tohoku.ac.jp
[3] Preferred Networks, Inc. [4] The Institute of Statistical Mathematics
sosk@preferred.jp fukumizu@ism.ac.jp

Abstract

In this paper, we propose a new kernel-based co-occurrence measure that can be applied to sparse linguistic expressions (e.g., sentences) with a very short learning time, as an alternative to pointwise mutual information (PMI). As well as deriving PMI from mutual information, we derive this new measure from the Hilbert–Schmidt independence criterion (HSIC); thus, we call the new measure the pointwise HSIC (PHSIC). PHSIC can be interpreted as a smoothed variant of PMI that allows various similarity metrics (e.g., sentence embeddings) to be plugged in as kernels. Moreover, PHSIC can be estimated by simple and fast (linear in the size of the data) matrix calculations regardless of whether we use linear or nonlinear kernels. Empirically, in a dialogue response selection task, PHSIC is learned thousands of times faster than an RNN-based PMI while outperforming PMI in accuracy. In addition, we also demonstrate that PHSIC is beneficial as a criterion of a data selection task for machine translation owing to its ability to give high (low) scores to a consistent (inconsistent) pair with other pairs.

1 Introduction

Computing the co-occurrence strength between two linguistic expressions is a fundamental task in natural language processing (NLP). For example, in collocation extraction (Manning and Schütze, 1999), word bigrams are collected from corpora and then strongly co-occurring bigrams (e.g., "New York") are found. In dialogue response selection (Lowe et al., 2015), pairs comprising a context and its response sentence are collected from dialogue corpora and the goal is to rank the candidate responses for each given context sentence. In either case, a set of linguistic expression pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ is first collected and then the co-occurrence strength of a (new) pair (x, y) is computed.

		Robustness to Sparsity	Learning Time
PMI			
$\log \frac{n \cdot c(x,y)}{\sum_{y'} c(x,y') \sum_{x'} c(x',y)}$	Eq. 1		✓
$\log \frac{\widehat{\mathbf{P}}_{\mathrm{RNN}}(y \mid x)}{\widehat{\mathbf{P}}_{\mathrm{RNN}}(y)}$	Eq. 2	✓	
PHSIC			
$(\phi(x)-\overline{\phi(x)})^\top \widehat{C}_{XY} (\psi(y)-\overline{\psi(y)})$	Sec. 5.1	✓	✓
$(\boldsymbol{a}-\overline{\boldsymbol{a}})^\top \widehat{C}_{\mathrm{ICD}} (\boldsymbol{b}-\overline{\boldsymbol{b}})$	Sec. 5.2	✓	✓

Table 1: The proposed co-occurrence norm, PHSIC, eliminates the trade-off between robustness to data sparsity and learning time, which PMI has (Section 1).

Pointwise mutual information (PMI) (Church and Hanks, 1989) is frequently used to model the co-occurrence strength of linguistic expression pairs. There are two typical types of PMI estimation (computation) method. One is a counting-based estimator using maximum likelihood estimation, sometimes with smoothing techniques, for example,

$$\widehat{\mathrm{PMI}}_{\mathrm{MLE}}(x, y; \mathcal{D}) = \log \frac{n \cdot c(x,y)}{\sum_{y'} c(x,y') \sum_{x'} c(x',y)}, \tag{1}$$

where $c(x, y)$ denotes the frequency of the pair (x, y) in given data $\mathcal{D}$. This is easy to compute and is commonly used to measure co-occurrence between words, such as in collocation extraction[1]; however, when data $\mathcal{D}$ is sparse, i.e., when x or y is a phrase or sentence, this approach is unrealistic. The second method uses recurrent neural networks (RNNs). Li et al. (2016) proposed to em-

---

[1] In collocation extraction, simple counting $c(x, y) \propto \widehat{\mathbf{P}}(x, y)$, rather than PMI, ranks undesirable function-word pairs (e.g., "of the") higher (Manning and Schütze, 1999).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1763–1775
Brussels, Belgium, October 31 - November 4, 2018.

ploy PMI to suppress *dull* responses for utterance generation in dialogue systems[2]. They estimated $\mathbf{P}(y)$ and $\mathbf{P}(y|x)$ using RNN language models and estimated PMI as follows:

$$\widehat{\mathrm{PMI}}_{\mathrm{RNN}}(x, y; \mathcal{D}) = \log \frac{\widehat{\mathbf{P}}_{\mathrm{RNN}}(y|x)}{\widehat{\mathbf{P}}_{\mathrm{RNN}}(y)}. \quad (2)$$

This way of estimating PMI is applicable to sparse language expressions; however, learning RNN language models is computationally costly.

To eliminate this trade-off between robustness to data sparsity and learning time, in this study we propose a new kernel-based co-occurrence measure, which we call the *pointwise Hilbert–Schmidt independence criterion (PHSIC)* (see Table 1). Our contributions are as follows:

- We formalize PHSIC, which is derived from HSIC (Gretton et al., 2005), a kernel-based dependence measure, in the same way that PMI is derived from mutual information (Section 3).
- We give an intuitive explanation why PHSIC is robust to data sparsity. PHSIC is a "smoothed variant of PMI", which allows various similarity metrics to be plugged in as kernels (Section 4).
- We propose fast estimators of PHSIC, which are reduced to a simple and fast matrix calculation regardless of whether we use linear or nonlinear kernels (Section 5).
- We empirically confirmed the effectiveness of PHSIC, i.e., its robustness to data sparsity and learning time, in two different types of experiment, a dialogue response selection task and a data selection task for machine translation (Section 6).

2 Problem Setting

Let X and Y denote random variables on $\mathcal{X}$ and $\mathcal{Y}$, respectively. In this paper, we deal with the tasks of taking a set of linguistic expression pairs

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n} \underset{\text{i.i.d.}}{\sim} \mathbf{P}_{XY}, \quad (3)$$

which is regarded as a set of i.i.d. samples drawn from a joint distribution $\mathbf{P}_{XY}$, and then measuring the "co-occurrence strength" for each given pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Such tasks include collocation extraction and dialogue response selection (Section 1).

3 Pointwise HSIC

In this section, we give the formal definition of PHSIC, a new kernel-based co-occurrence measure. We show a summary of this section in Table 2. Intuitively, PHSIC is a "kernelized variant of PMI."

3.1 Dependence Measure

As a preliminary step, we introduce the simple concept of dependence (see **Dependence Measure** in Table 2). Recall that random variables X and Y are *independent* if and only if the joint probability density $\mathbf{P}_{XY}$ and the product of the marginals $\mathbf{P}_X\mathbf{P}_Y$ are equivalent. Therefore, we can measure the *dependence* between random variables X and Y via the difference between $\mathbf{P}_{XY}$ and $\mathbf{P}_X\mathbf{P}_Y$.

Both the mutual information and the Hilbert–Schmidt independence criterion, to be described below, are such dependence measures.

3.2 MI and PMI

We briefly review the well-known mutual information and PMI (see **MI & PMI** in Table 2).

The *mutual information (MI)*[3] between two random variables X and Y is defined by

$$\mathrm{MI}(X, Y) := \mathrm{KL}[\mathbf{P}_{XY} \| \mathbf{P}_X\mathbf{P}_Y] \quad (4)$$

(Cover and Thomas, 2006), where $\mathrm{KL}[\cdot\|\cdot]$ denotes the Kullback–Leibler (KL) divergence. Thus, $\mathrm{MI}(X, Y)$ is the degree of dependence between X and Y measured by the KL divergence between $\mathbf{P}_{XY}$ and $\mathbf{P}_X\mathbf{P}_Y$.

Here, by definition of the KL divergence, MI can be represented in the form of the expectation over $\mathbf{P}_{XY}$, i.e., the summation over all possible pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$:

$$\mathrm{MI}(X, Y) = \underset{(x,y)}{\mathbf{E}}\left[\log \frac{\mathbf{P}_{XY}(x, y)}{\mathbf{P}_X(x)\mathbf{P}_Y(y)}\right]. \quad (5)$$

The shaded part in Equation (5) is actually the *pointwise mutual information (PMI)* (Church and Hanks, 1989):

$$\mathrm{PMI}(x, y; X, Y) := \log \frac{\mathbf{P}_{XY}(x, y)}{\mathbf{P}_X(x)\mathbf{P}_Y(y)}. \quad (6)$$

Therefore, $\mathrm{PMI}(x, y)$ can be thought of as the contribution of (x, y) to $\mathrm{MI}(X, Y)$.

---

[2] In dialogue response selection or generation, a simple conditional probability $\widehat{\mathbf{P}}(y|x)$, rather than PMI, ranks *dull* responses (e.g., "I don't know.") higher (Li et al., 2016).

[3] Conventionally, mutual information is denoted by $I(X; Y)$; in this paper, however, for notational consistency, mutual information is denoted by $\mathrm{MI}(X, Y)$.

	Dependence Measure the dependence between X and Y (the difference between $\mathbf{P}_{XY}$ and $\mathbf{P}_X\mathbf{P}_Y$)	**Co-occurrence Measure** the contribution of (x, y) to the dependence between X and Y
MI & PMI	$\mathrm{MI}(X, Y) = \mathrm{KL}[\mathbf{P}_{XY} \| \mathbf{P}_X\mathbf{P}_Y]$ $= \underset{(x,y)}{\mathbf{E}}\left[\log \frac{\mathbf{P}_{XY}(x,y)}{\mathbf{P}_X(x)\mathbf{P}_Y(y)}\right]$	$\mathrm{PMI}(x, y; X, Y)$ $= \log \frac{\mathbf{P}_{XY}(x,y)}{\mathbf{P}_X(x)\mathbf{P}_Y(y)}$
HSIC & PHSIC	$\mathrm{HSIC}(X, Y; k, \ell) = \mathrm{MMD}^2_{k,\ell}[\mathbf{P}_{XY}, \mathbf{P}_X\mathbf{P}_Y]$ $= \underset{(x,y)}{\mathbf{E}}\left[(\phi(x) - m_X)^\top C_{XY}(\psi(y) - m_Y)\right]$ $= \underset{(x,y)}{\mathbf{E}}\left[\underset{(x',y')}{\mathbf{E}}[\widetilde{k}(x,x')\widetilde{\ell}(y,y')]\right]$	$\mathrm{PHSIC}(x, y; X, Y, k, \ell)$ $= (\phi(x) - m_X)^\top C_{XY}(\psi(y) - m_Y)$ $= \underset{(x',y')}{\mathbf{E}}[\widetilde{k}(x,x')\widetilde{\ell}(y,y')]$

Table 2: Relationship between the mutual information (MI), the pointwise mutual information (PMI), the Hilbert–Schmidt independence criterion (HSIC), and the pointwise HSIC (PHSIC). As well as defining PMI as the contribution to MI, we define PHSIC as the contribution to HSIC. In short, PHSIC is a "kernelized PMI" (Section 3).

3.3 HSIC and PHSIC

As seen in the previous section, PMI can be derived from MI. Here, we consider replacing MI with the Hilbert–Schmidt independence criterion (HSIC). Then, in analogy with the relationship between PMI and MI, we derive PHSIC from HSIC (see **HSIC & PHSIC** in Table 2).

Let $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $\ell\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ denote positive definite kernels on $\mathcal{X}$ and $\mathcal{Y}$, respectively (intuitively, they are similarity functions between linguistic expressions). The *Hilbert–Schmidt independence criterion (HSIC)* (Gretton et al., 2005), a kernel-based dependence measure, is defined by

$$\mathrm{HSIC}(X, Y; k, \ell) := \mathrm{MMD}^2_{k,\ell}[\mathbf{P}_{XY}, \mathbf{P}_X\mathbf{P}_Y], \quad (7)$$

where $\mathrm{MMD}[\cdot, \cdot]$ denotes the maximum mean discrepancy (MMD) (Gretton et al., 2012), which measures the difference between random variables on a kernel-induced feature space. Thus, $\mathrm{HSIC}(X, Y; k, \ell)$ is the degree of dependence between X and Y measured by the MMD between $\mathbf{P}_{XY}$ and $\mathbf{P}_X\mathbf{P}_Y$, while MI is measured by the KL divergence (Equation (4)).

Analogous to MI in Equation (5), HSIC can be represented in the form of the expectation on $\mathbf{P}_{XY}$ by a simple deformation:

$$\begin{aligned}&\mathrm{HSIC}(X, Y; k, \ell)\\ &= \underset{(x,y)}{\mathbf{E}}\left[(\phi(x) - m_X)^\top C_{XY}(\psi(y) - m_Y)\right] && (8)\\ &= \underset{(x,y)}{\mathbf{E}}\left[\underset{(x',y')}{\mathbf{E}}[\widetilde{k}(x,x')\widetilde{\ell}(y,y')]\right], && (9)\end{aligned}$$

where

$$\phi(x) := k(x, \cdot), \quad \psi(y) := \ell(y, \cdot), \quad (10)$$

$$m_X := \mathbf{E}_x[\phi(x)], \quad m_Y := \mathbf{E}_y[\psi(y)], \quad (11)$$

$$C_{XY} := \underset{(x,y)}{\mathbf{E}}\left[(\phi(x) - m_X)(\psi(y) - m_Y)^\top\right], \quad (12)$$

$$\begin{aligned}\widetilde{k}(x, x') &:= k(x, x') - \mathbf{E}_{x'}[k(x, x')]\\ &\quad - \mathbf{E}_x[k(x, x')] + \mathbf{E}_{x,x'}[k(x, x')]. \quad (13)\end{aligned}$$

At first glance, these equations are somewhat complicated; however, the estimators of PHSIC we actually use are reduced to a simple matrix calculation in Section 5. Unlike MI in Equation (5), HSIC has two representations: Equation (8) is the representation in feature space and Equation (9) is the representation in data space.

Similar to the relationship between MI and PMI (Section 3.2), we define the *pointwise Hilbert–Schmidt independence criterion (PHSIC)* by the shaded parts in Equations (8) and (9):

$$\begin{aligned}&\mathrm{PHSIC}(x, y; X, Y, k, \ell)\\ &:= (\phi(x) - m_X)^\top C_{XY}(\psi(y) - m_Y) && (14)\\ &= \underset{(x',y')}{\mathbf{E}}[\widetilde{k}(x,x')\widetilde{\ell}(y,y')]\,. && (15)\end{aligned}$$

Namely, $\mathrm{PHSIC}(x, y)$ is defined as the contribution of (x, y) to $\mathrm{HSIC}(X, Y)$.

In summary, we define PHSIC such that "MI:PMI = HSIC:PHSIC" holds (see Table 2).

4 PHSIC as Smoothed PMI

This section gives an intuitive explanation for the first feature of PHSIC, i.e., the robustness to data sparsity, using Table 3. In short, we show that PHSIC is a "smoothed variant of PMI."

First, the maximum likelihood estimator of PMI

	add scores	deduct scores
$\widehat{\mathrm{PMI}}(x, y; \mathcal{D}) = \boxed{\log \frac{n \cdot \sum_i \mathbb{I}[x = x_i \wedge y = y_i]}{\sum_i \mathbb{I}[x = x_i] \sum_i \mathbb{I}[y = y_i]}}$	$\begin{array}{c} (x, y) \\ \Vert \quad \Vert \\ \mathcal{D} = \{\dots, (x_i, y_i), \dots\} \end{array}$	$\begin{array}{ccc} (x, y) & & (x, y) \\ \Vert \quad \nparallel & & \nparallel \quad \Vert \\ \{\dots, (x_i, y_i), & \dots, & (x_i, y_i), \dots\} \end{array}$
$\widehat{\mathrm{PHSIC}}(x, y; \mathcal{D}, k, \ell) = \boxed{\frac{1}{n} \sum_i \widehat{\widetilde{k}}(x, x_i) \widehat{\widetilde{\ell}}(y, y_i)}$	$\begin{array}{ccc} (x, y) & & (x, y) \\ \wr \quad \wr & & \not\wr \quad \not\wr \\ \{\dots, (x_i, y_i), & \dots, & (x_i, y_i), \dots\} \end{array}$	$\begin{array}{ccc} (x, y) & & (x, y) \\ \wr \quad \not\wr & & \not\wr \quad \wr \\ \{\dots, (x_i, y_i), & \dots, & (x_i, y_i), \dots\} \end{array}$

Table 3: Comparison of estimators of PMI and PHSIC in terms of methods of matching the given (x, y) and the observed (x_i, y_i) in $\mathcal{D}$. PMI matches them in an exact manner, while PHSIC smooths the matching using kernels. Therefore, PHSIC is expected to be robust to data sparsity (Section 4).

in Equation (1) can be rewritten as

$$\widehat{\mathrm{PMI}}(x, y; \mathcal{D}) = \boxed{\log \frac{n \cdot \sum_i \mathbb{I}[x = x_i \wedge y = y_i]}{\sum_i \mathbb{I}[x = x_i] \sum_i \mathbb{I}[y = y_i]}}, \quad (16)$$

where $\mathbb{I}[\text{condition}] = 1$ if the condition is true and $\mathbb{I}[\text{condition}] = 0$ otherwise. According to Equation (16), $\widehat{\mathrm{PMI}}(x, y)$ is the amount computed by repeating the following operation (see the first row in Table 3):

> *collate the given (x, y) and the observed (x_i, y_i) in $\mathcal{D}$ in order, and add the scores if (x, y) and (x_i, y_i) match exactly or deduct the scores if either the x side or the y side (but nor both) matches.*

Moreover, an estimator of PHSIC in data space (Equation (15)) is

$$\widehat{\mathrm{PHSIC}}(x, y; \mathcal{D}, k, \ell) = \boxed{\frac{1}{n} \sum_i \widehat{\widetilde{k}}(x, x_i) \widehat{\widetilde{\ell}}(y, y_i)}, \quad (17)$$

where $\widehat{\widetilde{k}}(\cdot, \cdot)$ and $\widehat{\widetilde{\ell}}(\cdot, \cdot)$ are similarity functions centered on the data[4]. According to Equation (17), $\widehat{\mathrm{PHSIC}}(x, y)$ is the amount computed by repeating the following operation (see the second row in Table 3):

> *collate the given (x, y) and the observed (x_i, y_i) in $\mathcal{D}$ in order, and add the scores if the similarities on the x and y sides are both higher (both $\widehat{\widetilde{k}}(x, x_i) > 0$ and $\widehat{\widetilde{\ell}}(y, y_i) > 0$ hold)[5] or deduct the scores if the similarities on either the x or y sides are similar but those on the other side are not similar.*

As described above, when comparing the estimators of PMI and PHSIC from the viewpoint of "methods of matching the given (x, y) and the observed (x_i, y_i)," it is understood that PMI matches them in an exact manner, while PHSIC smooths the matching using kernels (similarity functions).

With this mechanism, even for completely unknown pairs, it is possible to estimate the co-occurrence strength by referring to observed pairs through the kernels. Therefore, PHSIC is expected to be robust to data sparsity and can be applied to phrases and sentences.

Available Kernels for PHSIC In NLP, a variety of similarity functions (i.e., positive definite kernels) are available. We can freely utilize such resources, such as cosine similarity between sentence embeddings. For a more detailed discussion, see Appendix A.

5 Empirical Estimators of PHSIC

Recall that we have two types of empirical estimator of PMI, the maximum likelihood estimator (Equation (1)) and the RNN-based estimator (Equation (2)). In this section, we describe how to rapidly estimate PHSIC from data. When using the linear kernel or cosine similarity (e.g., cosine similarity between sentence embeddings), PHSIC can be efficiently estimated in feature space (Section 5.1). When using a nonlinear kernel such as the Gaussian kernel, PHSIC can also be estimated efficiently in data space via a simple matrix decomposition (Section 5.2).

5.1 Estimation Using Linear Kernel or Cosine

When using the linear kernel or cosine similarity, the estimator of PHSIC in feature space (14) is as follows:

---

[4] To be exact, $\widehat{\widetilde{k}}(x, x') := k(x, x') - \frac{1}{n}\sum_{j=1}^{n} k(x, x_j) - \frac{1}{n}\sum_{i=1}^{n} k(x_i, x') + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i, x_j)$, which is an estimator of the centered kernel $\widetilde{k}(x, x')$ in Equation (13).

[5] In addition, the scores are added if the similarity on the x side and that on the y side are both lower, that is, if $\widehat{\widetilde{k}}(x, x_i) < 0$ and $\widehat{\widetilde{\ell}}(y, y_i) < 0$ hold.

$$\widehat{\mathrm{PHSIC}}_{\mathrm{feature}}(x, y; \mathcal{D}, k, \ell) = (\phi(x) - \overline{\phi(x)})^\top \widehat{C}_{XY}(\psi(y) - \overline{\psi(y)}), \tag{18}$$

where

$$\phi(x) = \begin{cases} x & (k(x, x') = x^\top x') \\ x/\|x\| & (k(x, x') = \cos(x, x')) \end{cases}, \tag{19}$$

$$\overline{\phi(x)} := \frac{1}{n}\sum_{i=1}^{n} \phi(x_i), \quad \overline{\psi(y)} := \frac{1}{n}\sum_{i=1}^{n} \psi(y_i), \tag{20}$$

$$\widehat{C}_{XY} := \frac{1}{n}\sum_{i=1}^{n} \phi(x_i)\psi(y_i)^\top - \overline{\phi(x)}\,\overline{\psi(y)}^\top. \tag{21}$$

Generally in kernel methods, a feature map $\phi(\cdot)$ induced by a kernel $k(\cdot, \cdot)$ is unknown or high-dimensional and it is difficult to compute estimated values in feature space[6]. However, when we use the linear kernel or cosine similarity, feature maps can be explicitly determined (Equation (19)).

Computational Cost When learning Equation (18) with feature maps $\phi\colon \mathcal{X} \to \mathbb{R}^d$ and $\psi\colon \mathcal{Y} \to \mathbb{R}^d$, computing the vectors $\overline{\phi(x)}, \overline{\psi(y)} \in \mathbb{R}^d$ and the matrix $\widehat{C}_{XY} \in \mathbb{R}^{d \times d}$ takes $\mathcal{O}(nd^2)$ time and $\mathcal{O}(nd)$ space (linear in the size of the input, n). When estimating PHSIC(x, y), computing $\phi(x), \psi(y) \in \mathbb{R}^d$ and Equation (18) takes $\mathcal{O}(d^2)$ time (constant; does not depend on the size of the input, n).

5.2 Estimation Using Nonlinear Kernels

When using a nonlinear kernel such as the Gaussian kernel, it is necessary to estimate PHSIC in data space. Using a simple matrix decomposition, this can be achieved with the same computational cost as the estimation in feature space. See Appendix B for a detailed derivation.

6 Experiments

In this section, we provide empirical evidence for the greater effectiveness of PHSIC than PMI, i.e., a very short learning time and robustness to data sparsity. Among the many potential applications of PHSIC, we choose two fundamental scenarios, (re-)ranking/classification and data selection.

- In the ranking/classification scenario (measuring the co-occurrence strength of *new data pairs* with reference to observed pairs), PHSIC is applied as a criterion for the dialogue response selection task (Section 6.2).
- In the data selection/filtering scenario (ordering the entire set of *observed data pairs* according to the co-occurrence strength), PHSIC is also applied as a criterion for data selection in the context of machine translation (Section 6.3).

6.1 PHSIC Settings

To take advantage of recent developments in representation learning, we used several pre-trained models for encoding sentences into vectors and several kernels between these vectors for PHSIC.

Encoders As sentence encorders, we used two pre-trained models without fine-tuning. First, the sum of the word vectors effectively represents a sentence (Mikolov et al., 2013a):

$$\boldsymbol{x} = \textstyle\sum_{w \in x} \mathrm{vec}(w), \quad \boldsymbol{y} = \textstyle\sum_{w \in y} \mathrm{vec}(w). \tag{22}$$

For $\mathrm{vec}(\cdot)$, we used the pre-trained **`fastText`** model[7], which is a high-accuracy and popular word embedding model (Bojanowski et al., 2017); models in 157 languages are publicly distributed (Grave et al., 2018). Second, we also used a DNN-based sentence encoder, called the universal sentence encoder (Cer et al., 2018), which utilizes the deep averaging network (**`DAN`**) (Iyyer et al., 2015). The pre-trained model for English sentences we used is publicly available[8].

Kernels As kernels between these vectors, we used cosine similarity (**`cos`**)

$$k(\boldsymbol{x}, \boldsymbol{x}') = \cos(\boldsymbol{x}, \boldsymbol{x}') \tag{23}$$

and the Gaussian kernel (also known as the radial basis function kernel; **`RBF`** kernel)

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\sigma^2}\right), \tag{24}$$

and similarly for $\ell(\boldsymbol{y}, \boldsymbol{y}')$. The experiments are ran with hyperparameter $\sigma = 1.0$ for the RBF kernel, and $d = 100$ for incomplete Cholesky decomposition (for more detail, see Section B).

6.2 Ranking: Dialogue Response Selection

In the first experiment, we applied PHSIC as a ranking criterion of the task of dialogue response

---

[6] One of the characteristics of kernel methods is that an intractable estimation in feature space is replaced with an efficient estimation in data space.

[7] `https://fasttext.cc/docs/en/english-vectors.html`, `https://fasttext.cc/docs/en/crawl-vectors.html`

[8] `https://www.tensorflow.org/hub/modules/google/universal-sentence-encoder/1`

selection (Lowe et al., 2015); in the task, pairs comprising a context (previous utterance sequence) and its response are collected from dialogue corpora and the goal is to rank the candidate responses for each given context sentence.

The task entails sentence sequences (very sparse linguistic expressions); moreover, Li et al. (2016) pointed out that (RNN-based) PMI has a positive impact on suppressing *dull* responses (e.g., "I don't know.") in dialogue systems. Therefore, PHSIC, another co-occurrence measure, is also expected to be effective for this. With this setting, where the validity of PMI is confirmed, we investigate whether PHSIC can replace RNN-based PMI in terms of both learning time and robustness to data sparsity.

Experimental Settings

Dataset For the training data, we gathered approximately 5×10^5 reply chains from Twitter, following Sordoni et al. (2015)[9]. In addition, we randomly selected $\{10^3, 10^4, 10^5\}$ reply chains from that dataset. Using these small subsets, we confirmed the effect of the difference in the size of the training set (data sparseness) on the learning time and predictive performance.

For validation and test data, we used a small (approximately 2000 pairs each) but highly reliable dataset created by Sordoni et al. (2015)[10], which consists only of conversations given high scores by human annotators. Therefore, this set was not expected to include *dull* responses.

For each dataset, we converted each context-message-response triple into a context-response pair by concatenating the context and message following Li et al. (2016). In addition, to convert the test set (positive examples) to ten-choice multiple-choice questions, we shuffled the combinations of context and response to generate pseudo-negative examples.

Evaluation Metrics We adopted the following evaluation metrics for the task: (i) `ROC-AUC` (the area under the receiver operating characteristic curve), (ii) `MRR` (the mean reciprocal rank), and (iii) `Recall@{1,2}`.

	Config			Size of Training Set n			
				10^3	10^4	10^5	5×10^5
RNN-PMI	*Dim.*	*Init.*					
	300	`fastText`	Total	20.6	99.2	634.3	4042.5
			$\widehat{\mathbf{P}}(y)$	8.0	23.6	294.6	1710.1
			$\widehat{\mathbf{P}}(y\|x)$	12.6	75.6	339.7	2332.4
	1200	`fastText`	Total	49.0	162.0	1751.3	13054.9
			$\widehat{\mathbf{P}}(y)$	16.3	57.2	671.0	5512.1
			$\widehat{\mathbf{P}}(y\|x)$	32.7	104.8	1080.3	7542.8
PHSIC	*Encoder*	*Kernel*					
	`fastText`	`cos`		**0.0**	**0.1**	**0.5**	**2.8**
	`DAN`	`cos`		**0.0**	**0.1**	**0.4**	**1.8**

Table 4: **Learning time** [s] for each model and each size of training set for the dialogue response task. Each row denotes a model; each column denotes the number of training data n. The text appended to each baseline model denotes the number of dimension of hidden layers (*Dim.*) and the method of initialization the embedding layer (*Init.*). The text appended to each proposed model denotes the pre-trained models used to encode sentences into vectors (*Encoder*) and the kernel between these vectors (*Kernel*). The best result (the shortest learning time) in each column is in bold.

Experimental Procedure We used the following procedure: (i) train the model with a set of context-response pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$; (ii) for each context sentence x in the test data, rank the candidate responses $\{y_j\}_{j=1}^{10}$ by the model; and (iii) report three evaluation metrics.

Baseline Measures As baseline measures, both (1) an RNN language model $\widehat{\mathbf{P}}_{\mathrm{RNN}}(y)$ (Mikolov et al., 2010) and (2) a conditional RNN language model $\widehat{\mathbf{P}}_{\mathrm{RNN}}(y|x)$ (Sutskever et al., 2014) were trained, and (3) PMI based on these language models, **RNN-PMI**, was also used for experiments (see Equation (2)). We trained these models with all combinations of the following settings: (a) the number of dimensions of the hidden layers being 300 or 1200 and (b) the initialization of the embedding layer being **`random`** (uniform on $[-0.1, 0.1]$) or **`fastText`**. For more detailed settings, see Appendix C.

Experimental Results

Learning Time Table 4 shows the experimental results of the learning time[11]. Regardless of the size of the training set n, the learning time for

---

[9] We collected tweets after 2017 for our training set to avoid duplication with the test set, which contains tweets from the year 2012.

[10] `https://www.microsoft.com/en-us/download/details.aspx?id=52375`

[11] The computing environment was as follows:
(i) CPU: Xeon E5-1650-v3 (3.5 GHz, 6 Cores);
(ii) GPU: GTX 1080 (8 GB).

PHSIC is much shorter than that of the RNN-based method. For example, even when the size of the training set n is 5×10^5, PHSIC is approximately 1400–4000 times faster than RNN-based PMI. This is because the estimators of PHSIC are reduced to a deterministic and efficient matrix calculation (Section 5), whereas neural network-based models involve the sequential optimization of parameters via gradient descent methods.

Robustness to Data Sparsity Table 5 shows the experimental results of the predictive performance. When the size of the training data is small ($n = 10^3, 10^4$), that is, when the data is extremely sparse, the predictive performance of PHSIC hardly deteriorates while that of PMI rapidly decays as the number of data decreases. This indicates that PHSIC is more robust to data sparsity than RNN-based PMI owing to the effect of kernels. Moreover, PHSIC with the simple cosine kernel outperforms the RNN-based model regardless of the number of data, while the learning time of PHSIC is thousands of times shorter than those of the baseline methods (Section 6.2).

Additionally we report Spearman's rank correlation coefficient between models to verify whether PHSIC shows similar behavior to PMI. See Appendix D for more detail.

6.3 Data Selection for Machine Translation

The aim of our second experiment was to demonstrate that PHSIC is also beneficial as a criterion of data selection. To achieve this, we attempted to apply PHSIC to a parallel corpus filtering task that has been intensively discussed in recent (neural) machine translation (MT, NMT) studies. This task was first adopted as a shared task in the third conference on machine translation (WMT 2018)[12].

Several existing parallel corpora, especially those automatically gathered from large-scale text data, such as the Web, contain unacceptable amounts of noisy (low-quality) sentence pairs that greatly affect the translation quality. Therefore, the development of an effective method for parallel corpus filtering would potentially have a large influence on the MT community; discarding such noisy pairs may improve the translation quality and shorten the training time.

We expect PHSIC to give low scores to *exceptional* sentence pairs (misalignments or missing translations) during the selection process because PHSIC assigns low scores to pairs that are highly inconsistent with other pairs (see Section 4). Note that applying RNN-based PMI to a parallel corpus selection task is unprofitable since obtaining RNN-based PMI also has an identical computational cost for training a sequence-to-sequence model for MT, and thus, we cannot expect a reduction of the total training time.

Experimental Settings

Dataset We used the ASPEC-JE corpus[13], which is an official dataset used for the MT-evaluation shared task held in the fourth workshop on Asian translation (WAT 2017)[14] (Nakazawa et al., 2017). ASPEC-JE consists of approximately three million (3M) Japanese–English parallel sentences from scientific paper abstracts. As discussed by Kocmi et al. (2017), ASPEC-JE contains many low-quality parallel sentences that have the potential to significantly degrade the MT quality. In fact, they empirically revealed that using only the reliable part of the training parallel corpus significantly improved the translation quality. Therefore, ASPEC-JE is a suitable dataset for evaluating the data selection ability.

Model For our data selection evaluation, we selected the Transformer architecture (Vaswani et al., 2017) as our baseline NMT model, which is widely-used in the NMT community and known as one of the current state-of-the-art architectures. We utilized `fairseq`[15], a publicly available tool for neural sequence-to-sequence models, for building our models.

Experimental Procedure We used the following procedure for this evaluation: (1) rank all parallel sentences in a given parallel corpus according to each criterion, (2) extract the top K ranked parallel sentences, (3) train the NMT model using the extracted parallel sentences, and (4) evaluate the translation quality of the test data using a typical MT automatic evaluation measure, i.e., BLEU (Papineni et al., 2002)[16]. In our experiments we evaluated PHSIC with K = 0.5M and 1M.

---

[12] `http://www.statmt.org/wmt18/parallel-corpus-filtering.html`

[13] `http://lotus.kuee.kyoto-u.ac.jp/ASPEC/`

[14] `http://lotus.kuee.kyoto-u.ac.jp/WAT/WAT2017/`

[15] `https://github.com/pytorch/fairseq`

[16] We used `multi-bleu.perl` in the Moses tool (`https://github.com/moses-smt/mosesdecoder`).

Models	*Config*		**Size of Training Set** n			
			10^3	10^4	10^5	5×10^5
Chance Level			.50; .29; .10, .20	.50; .29; .10, .20	.50; .29; .10, .20	.50; .29; .10, .20
	Dim.	*Init.*				
$\widehat{\mathbf{P}}_{\mathrm{RNN}}(y)$	1200	`fastText`	.50; .29; .10, .21	.50; .30; .11, .21	.50; .30; .10, .21	.50; .30; .13, .25
$\widehat{\mathbf{P}}_{\mathrm{RNN}}(y\|x)$	1200	`fastText`	.50; .29; .10, .21	.50; .30; .10, .21	.52; .31; .11, .23	.54; .32; .13, .25
RNN-PMI	300	`random`	.51; .30; .10, .21	.51; .30; .11, .22	.58; .35; .14, .29	.69; .46; .25, .42
		`fastText`	.51; .29; .09, .20	.56; .34; .15, .25	.66; .41; .20, .36	.76; .56; .36, .54
	1200	`random`	.50; .29; .11, .20	.51; .30; .10, .19	.57; .35; .14, .29	.70; .47; .26, .44
		`fastText`	.51; .30; .11, .20	.52; .32; .13, .23	.65; .42; .21, .36	.75; .54; .34, .52
	Encoder	*Kernel*				
PHSIC	`fastText`	`cos`	.61; .38; .17, .33	.62; .40; .19, .34	.62; .40; .19, .34	.62; .40; .19, .34
	`DAN`	`cos`	**.77; .58; .40, .56**	**.78; .57; .39, .56**	**.78; .58; .41, .57**	**.78; .58; .40, .57**

Table 5: **Predictive performance** for each model and each training set size for the dialogue response selection task: `ROC-AUC`; `MRR`; `Recall@1,2`. The best result in each column is in bold. The other notation is the same as in Table 4.

Selection Criteria			**# of Selected Data** K		
			0.5M	1M	3M
(all the training set)			-	-	41.02
Random			34.26	39.82	-
fast_align			38.63	40.56	-
	Encoder	*Kernel*			
PHSIC	`fastText`	`RBF`	**38.95**	**40.95**	-

Table 6: **BLEU scores** with the Transformer for each data selection criterion and each size of selected data K for the parallel corpus filtering task.“Random” represents the baseline method of selecting sentences at random.

Baseline Measure As a baseline measure, we utilize a publicly available script[17] of **fast_align** (Dyer et al., 2013), which is one of the state-of-the-art word aligner. We firstly used the fast_align for the training set $\mathcal{D} = \{(x_i, y_i)\}_i$ to obtain the word alignment between each sentence pair (x_i, y_i), i.e., a set of aligned word pairs with its probabilities. We then computed the co-occurrence score of (x_i, y_i) with sentence-length normalization, i.e., the average log probability of aligned word pairs.

Experimental Results

Table 6 shows the results of our data selection evaluation. It is common knowledge in NMT that more data gives better performance in general. However, we observed that PHSIC successfully extracted beneficial parallel sentences from the noisy parallel corpus; the result using 1M data extracted from the 3M corpus by PHSIC was almost the same as that using 3M data (the decrease in the BLEU score was only 0.07), whereas that by random extraction reduced the BLEU score by 1.20.

This was actually a surprising result because PHSIC utilizes only monolingual similarity measures (kernels) without any other language resources. This indicates that PHSIC can be applied to a language pair poor in parallel resources. In addition, the surface form and grammatical characteristics between English and Japanese are extremely different[18]; therefore, we expect that PHSIC will work well regardless of the similarity of the language pair.

7 Related Work

Dependence Measures Measuring independence or dependence (correlation) between two random variables, i.e., estimating dependence from a set of paired data, is a fundamental task in statistics and a very wide area of data science. To measure the complex nonlinear dependence that real data has, we have several choices.

First, information-theoretic MI (Cover and Thomas, 2006) and its variants (Suzuki et al., 2009; Reshef et al., 2011) are the most commonly used dependence measures. However, to the best of our knowledge, there is no practical method of computing MIs for large-multi class high-dimensional

---

[17] `https://github.com/clab/fast_align`

[18] For example, word order; English is an SVO (subject-verb-object) language and Japanese is an SOV (subject-object-verb) language.

(having a complex generative model) discrete data, such as sparse linguistic data.

Second, several kernel-based dependence measures have been proposed for measuring nonlinear dependence (Akaho, 2001; Bach and Jordan, 2002; Gretton et al., 2005). The reason why kernel-based dependence measures work well for real data is that they do not explicitly estimate densities, which is difficult for high-dimensional data. Among them, HSIC (Gretton et al., 2005) is popular because it has a simple estimation method, which is used for various tasks such as feature selection (Song et al., 2012), dimensionality reduction (Fukumizu et al., 2009), and unsupervised object matching (Quadrianto et al., 2009; Jagarlamudi et al., 2010). We follow this line.

Co-occurrence Measures First, In NLP, PMI (Church and Hanks, 1989) and its variants (Bouma, 2009) are the de facto co-occurrence measures between *dense* linguistic expressions, such as words (Bouma, 2009) and simple narrative-event expressions (Chambers and Jurafsky, 2008). In recent years, positive PMI (PPMI) has played an important role as a component of word vectors (Levy and Goldberg, 2014).

Second, there are several studies in which the pairwise ranking problem has been solved by using deep neural networks (DNNs) in NLP. Li et al. (2016) proposed a PMI estimation using RNN language models; this was used as a baseline model in our experiments (see Section 6.2). Several studies have used DNN-based binary classifiers modeling $\mathbf{P}(C = \text{positive} \mid (x, y))$ to solve the given ranking problem directly (Hu et al., 2014; Yin et al., 2016; Mueller and Thyagarajan, 2016) (these networks are sometimes called Siamese neural networks). Our study focuses on comparing co-occurrence measures. It is unknown whether Siamese NNs capture the co-occurrence strength; therefore we did not deal with Siamese NNs in this paper.

Finally, to the best of our knowledge, Yokoi et al. (2017)'s paper is the first study that suggested converting HSIC to a pointwise measure. The present study was inspired by their suggestion; here, we have (i) provided a formal definition (population) of PHSIC; (ii) analyzed the relationship between PHSIC and PMI; (iii) proposed linear-time estimation methods; and (iv) experimentally verified the computation speed and robustness to data sparsity of PHSIC for practical applications.

8 Conclusion

The NLP community has commonly employed PMI to estimate the co-occurrence strength between linguistic expressions; however, existing PMI estimators have a high computational cost when applied to sparse linguistic expressions (Section 1). We proposed a new kernel-based co-occurrence measure, the pointwise Hilbert–Schmidt independent criterion (PHSIC). As well as defining PMI as the contribution to mutual information, PHSIC is defined as the contribution to HSIC; PHSIC is intuitively a "kernelized variant of PMI" (Section 3). PHSIC can be applied to sparse linguistic expressions owing to the mechanism of smoothing by kernels. Comparing the estimators of PMI and PHSIC, PHSIC can be interpreted as a smoothed variant of PMI, which allows various similarity metrics to be plugged in as kernels (Section 4). In addition, PHSIC can be estimated in linear time owing to the efficient matrix calculation, regardless of whether we use linear or nonlinear kernels (Section 5). We conducted a ranking task for dialogue systems and a data selection task for machine translation (Section 6). The experimental results show that (i) the learning of PHSIC was completed thousands of times faster than that of the RNN-based PMI while outperforming it in ranking accuracy (Section 6.2); and (ii) even when using a nonlinear kernel, PHSIC can be applied to a large dataset. Moreover, PHSIC reduces the amount of training data to one third without sacrificing the output translation quality (Section 6.3).

Future Work Using the PHSIC estimator in feature space (Equation (18)), we can generate the most appropriate $\psi(y)$ for a given $\phi(x)$ (uniquely, up to scale). That is, if a DNN-based sentence decoder is used, y (a sentence) can be restored from $\psi(y)$ (a feature vector) so that generative models of strong co-occurring sentences can be realized.

Acknowledgments

We are grateful to anonymous reviewers for their helpful comments. We also thank Weihua Hu for useful discussions, Kenshi Yamaguchi for collecting data, and Paul Reisert for proofreading. This work was supported in part by JSPS KAKENHI Grant Number JP15H01702 and JST CREST Grant Number JPMJCR1513, Japan.

References

Shotaro Akaho. 2001. A kernel method for canonical correlation analysis. In *IMPS*, pages 1–7.

Francis R. Bach and Michael I. Jordan. 2002. Kernel Independent Component Analysis. *JMLR*, 3(Jul):1–48.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *TACL*, 5:135–146.

Gerlof Bouma. 2009. Normalized (Pointwise) Mutual Information in Collocation Extraction. In *GSCL*, pages 31–40.

Razvan C Bunescu and Raymond J Mooney. 2006. Subsequence Kernels for Relation Extraction. In *NIPS*, pages 171–178.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *CoRR*, abs/1803.1.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *ACL*, pages 789–797.

Kenneth Ward Church and Patrick Hanks. 1989. Word Association Norms, Mutual Information, and Lexicography. In *ACL*, pages 76–83.

Michael Collins and Nigel Duffy. 2002. Convolution Kernels for Natural Language. In *NIPS*, pages 625–632.

Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory*. John Wiley & Sons.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised Sequence Learning. In *NIPS*, pages 3079–3087.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *NAACL-HLT*, pages 644–648.

Shai Fine and Katya Scheinberg. 2001. Efficient SVM Training Using Low-Rank Kernel Representations. *JMLR*, 2(Dec):243–264.

Kenji Fukumizu, Francis R. Bach, and Michael I. Jordan. 2009. Kernel dimension reduction in regression. *Annals of Statistics*, 37(4):1871–1905.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning Word Vectors for 157 Languages. In *LREC*, pages 3483–3487.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A Kernel Two-Sample Test. *JMLR*, 13(Mar):723–773.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. 2005. Measuring Statistical Dependence with Hilbert-Schmidt Norms. In *ALT*, pages 63–77.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning Distributed Representations of Sentences from Unlabelled Data. In *NAACL-HLT*, pages 1367–1377.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS*, pages 2042–2050.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *ACL/IJCNLP*, pages 1681–1691.

Jagadeesh Jagarlamudi, Seth Juarez, and Hal Daumé III. 2010. Kernelized Sorting for Natural Language Processing. In *AAAI*, pages 1020–1025.

Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *NIPS*, pages 3294–3302.

Tom Kocmi, Dušan Variš, and Ondřej Bojar. 2017. Cuni nmt system for wat 2017 translation tasks. In *WAT*, pages 154–159.

Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In *NIPS*, pages 2177–2185.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL-HLT*, pages 110–119.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *SIGDIAL*, pages 285–294.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, pages 3111–3119.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013b. Efficient Estimation of Word Representations in Vector Space. In *Proc. of the Workshop on ICLR*, pages 1–12.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, pages 1045–1048.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1429.

Alessandro Moschitti. 2006. Making Tree Kernels practical for Natural Language Learning. In *EACL*, volume 6, pages 113–120.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *AAAI*, 2012, pages 2786–2792.

Toshiaki Nakazawa, Shohei Higashiyama, Chenchen Ding, Hideya Mino, Isao Goto, Hideto Kazawa, Yusuke Oda, Graham Neubig, and Sadao Kurohashi. 2017. Overview of the 4th workshop on asian translation. In *WAT*, pages 1–54.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*, pages 1532–1543.

Novi Quadrianto, Le Song, and Alex J. Smola. 2009. Kernelized sorting. In *NIPS*, pages 1289–1296.

David N. Reshef, Yakir A. Reshef, Hilary K. Finucane, Sharon R. Grossman, Gilean McVean, Peter J. Turnbaugh, Eric S. Lander, Michael Mitzenmacher, and Pardis C. Sabeti. 2011. Detecting Novel Associations in Large Data Sets. *Science*, 334(6062):1518–1524.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Le Song, Alex Smola, Arthur Gretton, Justin Bedo, and Karsten Borgwardt. 2012. Feature Selection via Dependence Maximization. *JMLR*, 13:1393–1434.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. In *NAACL-HLT*, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*, pages 3104–3112.

Taiji Suzuki, Masashi Sugiyama, Takafumi Kanamori, and Jun Sese. 2009. Mutual information estimation reveals global associations between stimuli and biological processes. *BMC Bioinformatics*, 10(Suppl 1):S52.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a Next-Generation Open Source Framework for Deep Learning. In *LearningSys*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*, pages 5998–6008.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards Universal Paraphrastic Sentence Embeddings. In *ICLR*, pages 1–19.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *TACL*, 4(1):259–272.

Sho Yokoi, Daichi Mochihashi, Ryo Takahashi, Naoaki Okazaki, and Kentaro Inui. 2017. Learning Co-Substructures by Kernel Dependence Maximization. In *IJCAI*, pages 3329–3335.

A Available Kernels for PHSIC

Similarity between Sentence Vectors A variety of vector representations of phrases and sentences based on the distributional hypothesis have recently been proposed, including sentence encoders (Kiros et al., 2015; Dai and Le, 2015; Iyyer et al., 2015; Hill et al., 2016; Cer et al., 2018) and the sum of word embeddings; it is known as *additive compositionality* (Mitchell and Lapata, 2010; Mikolov et al., 2013a; Wieting et al., 2015) that we can express the meaning of phrases and sentences well with the sum of word vectors (e.g., `word2vec` (Mikolov et al., 2013b), `GloVe` (Pennington et al., 2014), and `fastText` (Bojanowski et al., 2017)). Note that various pre-trained models of sentence encoders and word embeddings have also been made available.

The cosine of these vectors, which is a positive definite kernel, can be used as a convenient and highly accurate similarity function between phrases or sentences. Other major kernels can also be used, such as the RBF kernel, the Laplacian kernel, and polynomial kernels.

Structured Kernels Various structured kernels for NLP, such as tree kernels, which capture fine structure of sentences such as syntax, were devised in the support vector machine era (Collins and Duffy, 2002; Bunescu and Mooney, 2006; Moschitti, 2006).

Combinations We can freely combine the previously mentioned kernels because the sum and the product of positive definite kernels are also positive definite kernels (Shawe-Taylor and Cristianini, 2004, Proposition 3.22).

B Derivation of Fast PHSIC Estimation in Data Space

Although estimators of HSIC and PHSIC depend on kernels k, ℓ and data $\mathcal{D}$, hereinafter, we use the following notation for the sake of simplicity:

$$\widehat{\mathrm{HSIC}}(X,Y) := \widehat{\mathrm{HSIC}}(X,Y;\mathcal{D},k,\ell), \tag{25}$$

$$\widehat{\mathrm{PHSIC}}(x,y) := \widehat{\mathrm{PHSIC}}(x,y;\mathcal{D},k,\ell). \tag{26}$$

Naïve Estimation Fist, an estimator of PHSIC in the data space (15) is

$$\widehat{\mathrm{PHSIC}}_{\mathrm{kernel}}(x,y) = (\boldsymbol{k}-\overline{\boldsymbol{k}})^\top(\tfrac{1}{n}H)(\boldsymbol{\ell}-\overline{\boldsymbol{\ell}}), \tag{27}$$

where $\boldsymbol{k} := (k(x,x_1),\dots,k(x,x_n))^\top \in \mathbb{R}^n$, so as $\boldsymbol{\ell}$; and vector $\overline{\boldsymbol{k}} := \frac{1}{n}K\mathbf{1}$ denotes empirical mean of $\{\boldsymbol{k}_i\}_{i=1}^n$, so as $\overline{\boldsymbol{\ell}}$. This estimation has a large computational cost. When learning, computing the vectors $\overline{\boldsymbol{k}}, \overline{\boldsymbol{\ell}}$ takes $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space. When estimating PHSIC, computing $\boldsymbol{k}, \boldsymbol{\ell}$ and multiplying the matrix $\frac{1}{n}H$ takes $\mathcal{O}(n)$ time.

Fast Estimation via Incomplete Cholesky Decomposition Equation (27) has a large computational cost because it is necessary to construct the Gram matrices K and $L \in \mathbb{R}^{n\times n}$. In kernel methods, several methods have been proposed for approximating Gram matrices at low cost without constructing them explicitly, such as *incomplete Cholesky decomposition* (Fine and Scheinberg, 2001).

By incomplete Cholesky decomposition, from data points $\{x_1,\dots,x_n\} \subseteq \mathcal{X}$ and a positive definite kernel $k\colon \mathcal{X}\times\mathcal{X} \to \mathbb{R}$, a matrix $A = (\boldsymbol{a}_1,\dots,\boldsymbol{a}_n)^\top \in \mathbb{R}^{n\times d}$ ($d \ll n$) can be obtained with $\mathcal{O}(nd^2)$ time complexity. This makes it possible to approximate the Gram matrix K by vectors $\boldsymbol{a}_i \in \mathbb{R}^d$ without configuring the entire of K:

$$\boldsymbol{a}_i^\top \boldsymbol{a}_j \approx k(x_i,x_j) \tag{28}$$

$$AA^\top \approx K. \tag{29}$$

Also, for HSIC, an efficient approximation method utilizing incomplete Cholesky decomposition has been proposed (Gretton et al., 2005, Lemma 2):

$$\widehat{\mathrm{HSIC}}_{\mathrm{ICD}}(X,Y) = \frac{1}{n^2}\|(HA)^\top B\|_{\mathrm{F}}^2, \tag{30}$$

where $A = (\boldsymbol{a}_1,\dots,\boldsymbol{a}_n)^\top \in \mathbb{R}^{n\times d}$ is a matrix satisfying $AA^\top \approx K$ computed via incomplete Cholesky decomposition, so as B ($BB^\top \approx L$). Equation (30) can be represented in the form of the expectation on data points:

$$\widehat{\mathrm{HSIC}}_{\mathrm{ICD}}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}\Big[(\boldsymbol{a}_i-\overline{\boldsymbol{a}})^\top \widehat{C}_{\mathrm{ICD}}(\boldsymbol{b}_i-\overline{\boldsymbol{b}})\Big] \tag{31}$$

$$\widehat{C}_{\mathrm{ICD}} := \frac{1}{n}(HA)^\top B \in \mathbb{R}^{d\times d}, \tag{32}$$

where vector $\overline{\boldsymbol{a}} := \frac{1}{n}A^\top\mathbf{1} \in \mathbb{R}^d$ denotes empirical mean of $\{\boldsymbol{a}_i\}_{i=1}^n$, so as $\overline{\boldsymbol{b}} := \frac{1}{n}B^\top\mathbf{1}$.

Recall that $\mathrm{PHSIC}(x,y)$ is the contribution of (x,y) to $\mathrm{HSIC}(X,Y)$ (see Section 3.3); PHSIC then can be efficiently estimated by the shaded part of Equation (31):

$$\widehat{\mathrm{PHSIC}}_{\mathrm{ICD}}(x,y) = (\boldsymbol{a}-\overline{\boldsymbol{a}})^\top \widehat{C}_{\mathrm{ICD}}(\boldsymbol{b}-\overline{\boldsymbol{b}}). \tag{33}$$

Here, the vector $\boldsymbol{a} \in \mathbb{R}^d$ corresponding to the new x can be calculated by “performing from halfway”

Models	*Config*			`(A)`	`(B)`	`(C)`	`(D)`
	Dim.	*Init.*					
RNN-PMI	300	`fastText`	`(A)`	–	.42	.12	.27
	1200	`fastText`	`(B)`	.42	–	.12	.26
	Encoder	*Kernel*					
PHSIC	`fastText`	`cos`	`(C)`	.12	.12	–	.16
	`DAN`	`cos`	`(D)`	.27	.26	.16	–

Table 7: **Spearman's** ρ between the co-occurrence scores computed by the models in the dialogue response selection task (Section 6.2). The size of training set n is 5×10^5. The other notation is the same as in Table 4.

on the incomplete Cholesky decomposition algorithm. Let $x^{(1)}, \ldots, x^{(d)}$ denote the dominant x_is adopted during decomposition algorithm. The jth element of $\boldsymbol{a}$ can be computed as follows:

$$\boldsymbol{a}[j] = \Big[k(x, x^{(j)}) - \sum_{m=1}^{j-1} \boldsymbol{a}[m] A_{jm}\Big] \,/\, A_{jj}, \quad (34)$$

so as $\boldsymbol{b} \in \mathbb{R}^d$ corresponding to the new y. The estimation via incomplete Cholesky decomposition (33) is extremely efficient compared to the naive estimation (27); Equation (33)'s computational complexity is equivalent to the estimation in the feature space (18).

C Detailed Settings for Learning RNNs

Detailed settings for learning RNNs used in this research are as follows:

- Hidden layers: single layer LSTMs (Hochreiter and Schmidhuber, 1997)
- Vocabulary: words with a frequency: 10 or more ($n = 5 \times 10^5$), 2 or more (otherwise)
- Dropout rate: 0.1 (300-dim), 0.3 (1200-dim)
- Batch size: 64
- Max epoch number: 5 ($n = 5 \times 10^5$), 30 (otherwise)
- Deep learning framework: `Chainer` (Tokui et al., 2015)

D Correlation Between Models in Dialogue Response Selection Task

Table D shows Spearman's rank correlation coefficient (Spearman's ρ) between the co-occurrence scores on the test set computed by the models in the dialogue response selection task (Section 6.2). This shows that the behavior of RNN-based PMI and PHSIC are considerably different. Furthermore, interestingly, the behavior of PHSICs using different kernels is also different. Possible reasons for these observations are as follows: (1) the difference in the dependence measures (MI or HSIC) on which each model is based; (2) the validity or numerical stability of estimating PMI with RNN language models; and (3) differences in the behavior of PHSIC originating from differences in the plugged in kernels. A more detailed analysis of the compatibility between tasks and measures (or kernels) is attractive future work.

Neural Related Work Summarization with a Joint Context-driven Attention Mechanism

Yongzhen Wang[1]*, **Xiaozhong Liu**[2,3], **Zheng Gao**[2]
[1]School of Maritime Economics and Management, Dalian Maritime University, Dalian, China
[2]School of Informatics, Computing and Engineering, Indiana University Bloomington,
Bloomington, IN, USA
[3]Alibaba Group, Hangzhou, China
`*kuadmu@163.com` `liu237@indiana.edu` `gao27@umail.iu.edu`

Abstract

Conventional solutions to automatic related work summarization rely heavily on human-engineered features. In this paper, we develop a neural data-driven summarizer by leveraging the seq2seq paradigm, in which a joint context-driven attention mechanism is proposed to measure the contextual relevance within full texts and a heterogeneous bibliography graph simultaneously. Our motivation is to maintain the topic coherency between a related work section and its target document, where both the textual and graphic contexts play a big role in characterizing the relationship among scientific publications accurately. Experimental results on a large dataset show that our approach achieves a considerable improvement over a typical seq2seq summarizer and five classical summarization baselines.

1 Introduction

In scientific fields, scholars need to contextualize their contribution to help readers acquire an understanding of their research papers. For this purpose, the related work section of an article serves as a pivot to connect prior domain knowledge, in which the innovation and superiority of current work are displayed by a comparison with previous studies. While citation prediction can assist in drafting a reference collection (Nallapati et al., 2008), consuming all these papers is still a laborious job, where authors must read every source document carefully and locate the most relevant content cautiously.

As a solution in saving authors' efforts, automatic related work summarization is essentially a topic-biased multi-document problem (Cong and Kan, 2010), which relies heavily on human-engineered features to retrieve snippets from the references. Most recently, neural networks enable a data-driven architecture *sequence-to-sequence* (seq2seq) for natural language generation (Bahdanau et al., 2014, 2016), where an encoder reads a sequence of words/sentences into a context vector, from which a decoder yields a sequence of specific outputs. Nonetheless, compared to scenarios like machine translation with an end-to-end nature, aligning a related work section to its source documents is far more challenging.

To address the summarization alignment, former studies try to apply an attention mechanism to measure the saliency/novelty of each candidate word/sentence (Tan et al., 2017), with the aim of locating the most representative content to retain primary coverage. However, toward summarizing a related work section, authors should be more creative when organizing text streams from the reference collection, where the selected content ought to highlight the topic bias of current work, rather than retell each reference in a compressed but balanced fashion. This motivates us to introduce the contextual relevance and characterize the relationship among scientific publications accurately.

Generally speaking, for a pair of documents, a larger lexical overlap often implies a higher similarity in their research backgrounds. Yet such a hypothesis is not always true when sampling content from multiple relevant topics. Take "DSSM"[1] as an example, from viewpoint of the abstract similarity, those references investigating "Information Retrieval", "Latent Semantic Model" or "Clickthrough Data Mining" could be of more importance in correlation and should be greatly sampled for the related work section. But in reality, this article spends a bit larger chunk of texts (about 58%) to elaborate "Deep Learning" during the literature review, which is quite difficult for machines to grasp the contextual relevance therein. In addi-

---

* Corresponding author

[1] Learning deep structured semantic models for web search using clickthrough data (Huang et al., 2013)

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1776–1786
Brussels, Belgium, October 31 - November 4, 2018.

tion, other situations like emerging new concepts also suffer from the terminology variation or paraphrasing in varying degrees.

In this study, we utilize a heterogeneous bibliography graph to embody the relationship within a scalable scholarly database. Over the recent past, there is a surge of interest in exploiting diverse relations to analyze bibliometrics, ranging from literature recommendation (Yu et al., 2015) to topic evolvement (Jensen et al., 2016). In a graphical sense, interconnected papers transfer the credit among each other directly/indirectly through various patterns, such as paper citation, author collaboration, keyword association and releasing on series of venues, which constitutes the graphic context for outlining concerned topics. Unfortunately, a variety of edge types may pollute the information inquiry, where a slice of edges are not so important as the others on sampling content. Meanwhile, most existing solutions in mining heterogeneous graphs depend on the human supervision, e.g., *hyperedge* (Bu et al., 2010) and *metapath* (Swami et al., 2017). This is usually not easy to access due to the complexity of graph schemas.

Our contribution is threefold: First, we explore the *edge-type usefulness distribution* (EUD) on a heterogeneous bibliography graph, which enables the relationship discovery (between any pair of papers) for sampling the interested information. Second, we develop a novel seq2seq summarizer for the automatic related work summarization, where a joint context-driven attention mechanism is proposed to measure the contextual relevance within both textual and graphic contexts. Third, we conduct experiments on 8,080 papers with native related work sections, and experimental results show that our approach outperforms a typical seq2seq summarizer and five classical summarization baselines significantly.

2 Related Work

This study touches on several strands of research within automatic related work summarization and seq2seq summarizer as follows.

The idea of creating a related work section automatically is pioneered by Cong and Kan (2010) who design two rule-based strategies to extract sentences for general and detailed topics respectively. Subsequently, Hu and Wan (2014) exploit *probabilistic latent semantic indexing* to split candidate texts into different topic-biased parts, then apply several regression models to learn the importance of each sentence. Similarly, Widyantoro and Amin (2014) transform the summarization problem into classifying rhetorical categories of sentences, where each sentence is represented as a feature vector containing word frequency, sentence length and etc. Most recently, Chen and Hai (2016) construct a graph of representative keywords, in which a *minimum steiner tree* is figured out to guide the summarization as finding the least number of sentences to cover the discriminated nodes. In general, compared to traditional summaries, the automatic related work summarization receives less concerns over the past. However, these existing solutions cannot work without manual intervention, which limits the application scale to an extremely small size (see Table 1).

Authors	Number of papers
Cong and Kan (2010)	20
Hu and Wan (2014)	1,050
Widyantoro and Amin (2014)	50
Chen and Hai (2016)	3

Table 1: Data scales of previous studies on automatic related work summarization.

The earliest seq2seq summarizer stems from Rush et al. (2015) which utilizes a feed-forward network for compressing sentences, and later is expanded by Chopra et al. (2016) with a *recurrent neural network* (RNN). On this basis, Nallapati et al. (2016a,c) and Chen et al. (2016) both present a set of RNN-based models to address various aspects of abstractive summarization. Typically, Cheng and Lapata (2016) propose a general seq2seq summarizer, where an encoder learns the representation of documents while a decoder generates each word/sentence using an attention mechanism. With further research, Nallapati et al. (2016b) extend the sentence compression by trying a hierarchical attention architecture and a limited vocabulary during the decoding phase. Next, Narayan et al. (2017) leverage the side information as an attention cue to locate focus regions for summaries. Recently, inspired by *PageRank*, Tan et al. (2017) introduce a graph-based attention mechanism to tackle the saliency problem. Nonetheless, these methods all discuss the single-document scenario, which is far from the nature of automatic related work summarization.

In this study, derived from the general seq2seq summarizer of Cheng and Lapata (2016), we propose a joint context-driven attention mechanism to measure the contextual relevance within full texts and a heterogeneous bibliography graph simultaneously. To our best knowledge, we make the first attempt to develop a neural data-driven solution for the automatic related work summarization, and the practice of using the joint context as an attention cue is also less explored to date. Besides, this study is launched on a dataset with up to 8,080 papers, which is much greater than previous studies and makes our results more convincing.

Since text summarization via word-by-word generation is not mature at present (Cheng and Lapata, 2016; Nallapati et al., 2016b; Tan et al., 2017), we adopt the extractive sentential fashion for our summarizer, where a related work section is created by extracting and linking sentences from a reference collection. Meanwhile, this study follows the mode of Cong and Kan (2010) who assume that the collection is given as part of the input, and do not consider the citation sentences of each reference.

3 Methodology

3.1 Problem Formulation

To adapt the seq2seq paradigm, we formulate the automatic related work summarization into a sequential text generation problem as follows.

Given an unedited paper $\mathtt{t}$ (target document) and its n-size reference collection $\mathtt{R_t} = \{\mathtt{r}^{\mathtt{t}}_{1:n}\}$, we draw up a related work section for $\mathtt{t}$ by selecting sentences from $\mathtt{R_t}$. To be specific, each reference (source document) will be traversed one time sequentially, and without loss of generality, in the descending order of their significance to $\mathtt{t}$. Consequently, all sentences to be selected are concatenated into an m-length sequence $\mathtt{S_t} = \{\mathtt{s}^{\mathtt{t}}_{1:m}\}$ to feed the summarizer. For each candidate sentence $\mathtt{s}^{\mathtt{t}}_j$, once being visited, a label $\mathtt{y}^{\mathtt{t}}_j \in \{0, 1\}$ will be determined synchronously based on whether or not this sentence should be covered into the output. Our objective is to maximize the log-likelihood probability of observed labels $\mathtt{Y_t} = \{\mathtt{y}^{\mathtt{t}}_{1:m}\}$ under $\mathtt{R_t}$, $\mathtt{S_t}$ and summarizer parameters θ, as shown below.

$$\max \sum_{j=1}^{m} \log \Pr(\mathtt{y}^{\mathtt{t}}_j \mid \mathtt{R_t}; \mathtt{S_t}; \theta) \tag{1}$$

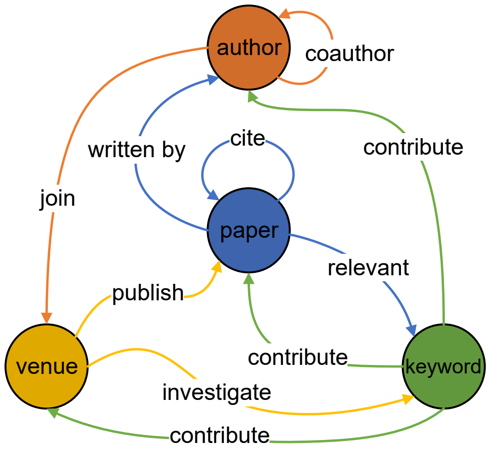

Figure 1: Heterogeneous bibliography graph.

3.2 Random Walk on Heterogeneous Bibliography Graph

Prior works have illustrated that one of the most promising channels for information recommendation is the community network (Guo and Liu, 2015). In this study, we verify this hypothesis toward the content sampling of scientific summarization, by investigating heterogeneous relations among different kinds of objects such as papers, authors, keywords and venues.

For measuring the relationship among scientific publications, we introduce a directed graph $\mathtt{G} = (\mathtt{V}, \mathtt{E})$ to contain various bibliographical connections, as shown in Figure 1, which involves four objects and ten edge types in total. Each edge $\mathtt{e}_{j,i} \in \mathtt{E}$ is assigned a value $\frac{\pi(\mathtt{e}_{j,i})}{z} \in [0, 1]$ to indicate the transition probability between two nodes $\mathtt{v}_j, \mathtt{v}_i \in \mathtt{V}$, where $\pi(\mathtt{e}_{j,i}) \in \mathbb{R}$ returns the unknown edge-type usefulness of $\mathtt{e}_{j,i}$, and $z \in \mathbb{R}$ is a normalizing weight. For most of edge types, we model the weight as one divided by the number of outgoing links of the same kind. But regarding the "contribution" category, the weight modeling is accomplished by *PageRank with Priors* (White and Smyth, 2003). Note that different edge types usually take very uneven importance in one particular task (Yu et al., 2015), and it is quite difficult to enable the classical heterogeneous graph mining without expert defined paths for *random walk* (Bu et al., 2010; Swami et al., 2017).

In this study, we propose an unsupervised approach to capture the connectivity diversity, by introducing an optimal EUD for navigating *random walkers* on the heterogeneous bibliography graph. Given a target document $\mathtt{t}$, the optimized usefulness assignment can help those walkers lock a top-n recommendation $\bar{\mathtt{R}}_\mathtt{t}$ to best match the reference collection $\mathtt{R}_\mathtt{t}$, as shown in Eq. 2. On this basis, a well-performing algorithm *node2vec* (Grover and Leskovec, 2016) is adopted to conduct an unsupervised *random walk* to vectorize every node $\forall \mathtt{v}_* \in \mathtt{V}$ into a d-dimensional embedding $\varphi(\mathtt{v}_*) \in \mathbb{R}^d$ so that any edge $\forall \mathtt{e}_* \in \mathtt{E}$ can be calculated therefrom. Specifically, we employ *evolutionary algorithm* (EA) to tune the EUD, which enjoys advantages over conventional gradient methods in both convergence speed and accuracy.

$$\arg\max \sum_{\mathtt{t}} \sum_{j=1}^{n} \log \Pr(\mathtt{r}_j^\mathtt{t} \in \bar{\mathtt{R}}_\mathtt{t} \mid \text{EUD}) \tag{2}$$

EA Setup We use an array of real numbers $\mathtt{x}_{1:10}$ to code an individual in the population, where $\mathtt{x}_j \in [0,1]$ denotes the usefulness of j-th edge type. Given an EUD, *PageRank* (Page, 1998) runs on graph to infer the relative importance of each node for each target document, and a fitness function is applied to judge how well this EUD satisfies locating the ground truth references as Eq. 3, in which if $\mathtt{r}_j^\mathtt{t}$ belongs to $\bar{\mathtt{R}}_\mathtt{t}$, then $\alpha(\mathtt{r}_j^\mathtt{t}, \bar{\mathtt{R}}_\mathtt{t}) \in \mathbb{N}$ returns the ranking of $\mathtt{r}_j^\mathtt{t}$ within $\bar{\mathtt{R}}_\mathtt{t}$, and otherwise a big penalty coefficient to prevent irrelevant references to be recommended. Like most other optimizations, this procedure starts with a randomly generated population.

$$\max \frac{1}{\sum_{\mathtt{t}} \sum_{j=1}^{n} \left| j - \alpha(\mathtt{r}_j^\mathtt{t}, \bar{\mathtt{R}}_\mathtt{t}) \right|} \tag{3}$$

EA Operator We choose the operator from *differential evolution* (Das and Suganthan, 2011) to generate offsprings for each individual. The basic idea is to utilize the difference between different individuals to disturb each trial object. First, three distinct individuals $\mathtt{x}_{1:10}^{r1}, \mathtt{x}_{1:10}^{r2}, \mathtt{x}_{1:10}^{r3}$ are sampled randomly from current population to create a variant $\mathtt{x}_{1:10}^{\text{var}}$, as shown in Eq. 4, where $f \in \mathbb{R}$ indicates the scaling factor. Next, $\mathtt{x}_{1:10}^{\text{var}}$ is crossed with a trial object $\mathtt{x}_{1:10}^{\text{tri}}$ to build a hybrid one $\mathtt{x}_{1:10}^{\text{hyb}}$ as Eq. 5, in which $c \in [0,1]$ denotes the crossover factor and $u \in [0,1]$ represents an uniform random number. At last, the fitnesses of $\mathtt{x}_{1:10}^{\text{tri}}$ and $\mathtt{x}_{1:10}^{\text{hyb}}$ are compared, and the better one will be saved as the offspring into a new round of evolution.

$$\mathtt{x}_j^{\text{var}} = \mathtt{x}_j^{r1} + f \times (\mathtt{x}_j^{r2} - \mathtt{x}_j^{r3}) \tag{4}$$

$$\mathtt{x}_j^{\text{hyb}} = \begin{cases} \mathtt{x}_j^{\text{var}}, & \text{if} \quad u \le c \\ \mathtt{x}_j^{\text{tri}}, & \text{otherwise} \end{cases} \tag{5}$$

3.3 Neural Extractive Summarization

As Figure 2 shows, we model our seq2seq summarizer with a hierarchical encoder and an attention-based decoder, as described below.

Hierarchical Encoder Our encoder consists of two major layers, namely a *convolutional neural network* (CNN) and a *long-short-term memory* (LSTM)-based RNN. Specifically, the CNN deals with word-level texts to derive sentence-level meanings, which are then taken as inputs to the RNN for handling longer-range dependency within lager units like a paragraph and even a whole paper. This conforms to the nature of document that is composed from words, sentences and higher levels of abstraction (Narayan et al., 2017).

Consider a sentence of p words $\mathtt{s}_j^\mathtt{t} = \{\mathtt{w}_{j,1:p}^\mathtt{t}\}$, where each word $\mathtt{w}_{j,i}^\mathtt{t}$ can be represented by a d-dimensional embedding $\phi(\mathtt{w}_{j,i}^\mathtt{t}) \in \mathbb{R}^d$. Previous studies have illustrated the strength of CNN in presenting sentences, because of its capability to learn compressed expressions and address sentences with variable lengths (Kim, 2014). First, a convolution kernel $k \in \mathbb{R}^{d \times q \times d}$ is applied to each possible window of q words to construct a list of feature maps as:

$$\mathtt{g}_{j,i}^\mathtt{t} = \tanh\left(k \times \phi(\mathtt{w}_{j,i:i+q-1}^\mathtt{t}) + b\right) \tag{6}$$

where $b \in \mathbb{R}^d$ denotes the bias term. Next, *max-over-time pooling* (Collobert et al., 2011) is performed on all generated features to obtain the sentence embedding as:

$$\phi(\mathtt{s}_j^\mathtt{t}) = \max_{1 \le i \le d} \left(\mathtt{g}_{j,1:p-q+1}^\mathtt{t}[i,:]\right) \tag{7}$$

where $[i,:]$ denotes the i-th row of matrix. Given a sequence of sentences $\mathtt{S}_\mathtt{t} = \{\mathtt{s}_{1:m}^\mathtt{t}\}$, we then take the RNN to yield an equal-length array of hidden states, in which LSTM has proved to alleviate the vanishing gradient problem when training long sequences (Hochreiter and Schmidhuber, 1997). Each hidden state can be viewed as a local representation with focusing on current and former sentences together, which is updated as: $\mathtt{h}_j^\mathtt{t} = \text{LSTM}\big(\phi(\mathtt{s}_j^\mathtt{t}), \mathtt{h}_{j-1}^\mathtt{t}\big) \in \mathbb{R}^d$.

In practice, we use multiple kernels with various widths to produce a group of embeddings for each

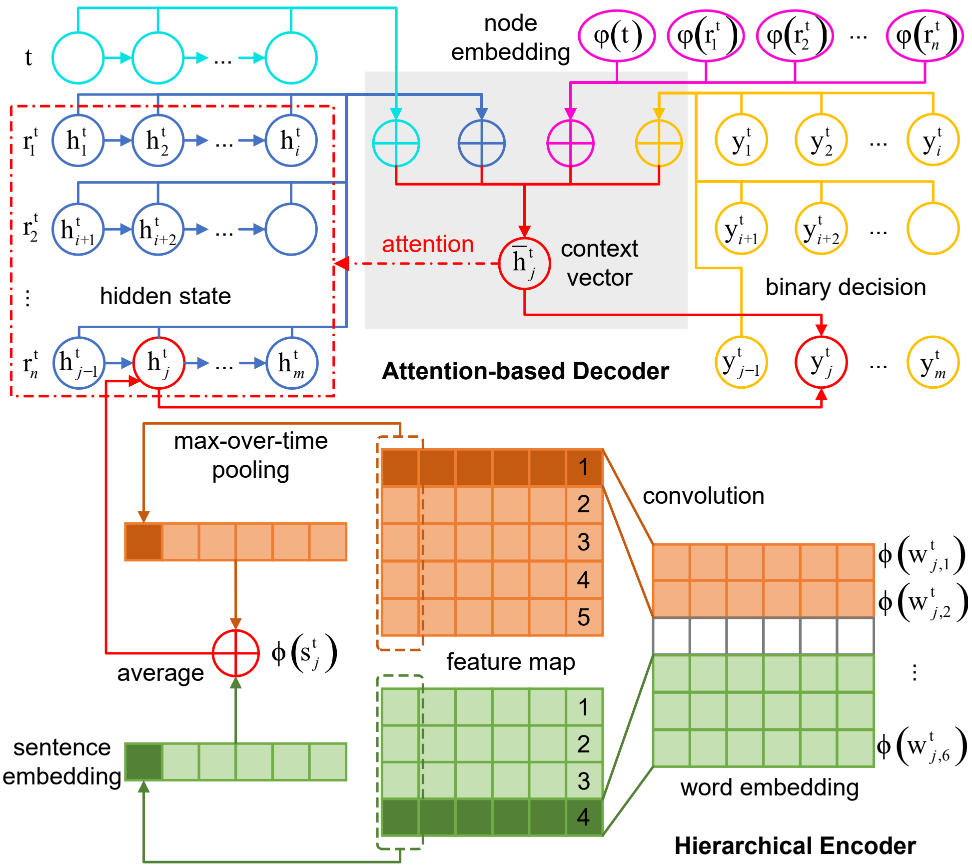

Figure 2: Framework of our seq2seq summarizer.

sentence, and average them to capture the information inside different n-grams. As Figure 2 (bottom) shows, the sentence $\mathtt{s}_j^{\mathtt{t}}$ involves six words, and two kernels of widths two (orange) and three (green) abstract a set of five and four feature maps respectively. Meanwhile, since *rhetorical structure theory* (Mann and Thompson, 2009) points out that association must exist in any two parts of coherent texts, RNN is only applicable to manage the sentence relation within a single document, because we cannot expect the dependency between two sections from different references.

Attention-based Decoder Our decoder labels each sentence $\mathtt{s}_j^{\mathtt{t}}$ as 0/1 sequentially, according to whether it is salient or novel enough, plus if relevant to the target document `t` or not. As shown in Figure 2 (top), the binary decision $\mathtt{y}_j^{\mathtt{t}}$ is made by both the hidden state $\mathtt{h}_j^{\mathtt{t}}$ and the context vector $\bar{\mathtt{h}}_j^{\mathtt{t}}$ from an attention mechanism (grey background). In particular, this attention (red dash line) is acted as an intermediate stage to determine which sentences to highlight so as to provide the contextual information for current decision (Bahdanau et al., 2014). Given $\mathtt{H}_{\mathtt{t}} = \{\mathtt{h}_{1:m}^{\mathtt{t}}\}$, this decoder returns the probability of $\mathtt{y}_j^{\mathtt{t}} = 1$ as below:

$$\Pr(\mathtt{y}_j^{\mathtt{t}} = 1 \mid \mathtt{R}_{\mathtt{t}}; \mathtt{S}_{\mathtt{t}}; \theta) = \text{sigmoid}\big(\delta(\mathtt{h}_j^{\mathtt{t}}, \bar{\mathtt{h}}_j^{\mathtt{t}})\big) \quad (8)$$

$$\bar{\mathtt{h}}_j^{\mathtt{t}} = \sum_{i=1}^{m} \mathtt{a}_{j,i} \mathtt{h}_i^{\mathtt{t}} \quad (9)$$

where $\delta(\mathtt{h}_j^{\mathtt{t}}, \bar{\mathtt{h}}_j^{\mathtt{t}}) \in \mathbb{R}$ denotes a fully connected layer with as input the concatenation of $\mathtt{h}_j^{\mathtt{t}}$ and $\bar{\mathtt{h}}_j^{\mathtt{t}}$, and $\mathtt{a}_{j,i} \in [0, 1]$ is the attention weight indicating how much the supporting sentence $\mathtt{s}_i^{\mathtt{t}}$ contributes to extracting the candidate one $\mathtt{s}_j^{\mathtt{t}}$.

Apart from saliency and novelty two traditional attention factors (Chen et al., 2016; Tan et al., 2017), we focus on the contextual relevance within both textual and graphic contexts to distinguish the relationship from near to far, as shown in Eq. 10 and Eq. 11. To be specific: 1) $\mathtt{h}_j^{\mathtt{t}\,\mathrm{T}} W_{\mathtt{s}} \mathtt{h}_i^{\mathtt{t}}$ represents the saliency of $\mathtt{s}_i^{\mathtt{t}}$ to $\mathtt{s}_j^{\mathtt{t}}$; 2) $-\mathtt{d}_j^{\mathtt{t}\,\mathrm{T}} W_{\mathtt{n}} \mathtt{h}_i^{\mathtt{t}}$ indicates the novelty of $\mathtt{s}_i^{\mathtt{t}}$ to the dynamic output $\mathtt{d}_j^{\mathtt{t}}$; 3) $\phi(\mathtt{t})^{\mathrm{T}} W_{\mathtt{t}} \mathtt{h}_i^{\mathtt{t}}$ denotes the relevance of $\mathtt{s}_i^{\mathtt{t}}$ to `t` from the textual context; 4) $\varphi(\mathtt{t})^{\mathrm{T}} W_{\mathtt{g}} \varphi(\mathtt{h}_i^{\mathtt{t}})$ refers to the relevance from the graphic context. More concretely, $W_* \in \mathbb{R}^d$ characterizes the learnable matrix, $\phi(\mathtt{t})$ returns the average of hidden states from $\mathtt{t}$, $\varphi(\mathtt{t})$ and $\varphi(\mathtt{h}_i^{\mathtt{t}})$ return the node embeddings of both $\mathtt{t}$ and the source document that $\mathtt{h}_i^{\mathtt{t}}$ belongs to respectively. Note that $\phi(\cdot)$ and $\varphi(\cdot)$ represent two distinct embedding spaces, where the former reflects the lexical collocations of corpus, and the latter embodies the connectivity patterns of associated graph.

$$\begin{aligned} \mathtt{a}_{j,i} = \quad & \mathtt{h}_j^{\mathtt{t}\,\mathrm{T}} W_{\mathtt{s}} \mathtt{h}_i^{\mathtt{t}} && \text{\# saliency} \\ & -\mathtt{d}_j^{\mathtt{t}\,\mathrm{T}} W_{\mathtt{n}} \mathtt{h}_i^{\mathtt{t}} && \text{\# novelty} \\ & +\phi(\mathtt{t})^{\mathrm{T}} W_{\mathtt{t}} \mathtt{h}_i^{\mathtt{t}} && \text{\# relevance}_1 \\ & +\varphi(\mathtt{t})^{\mathrm{T}} W_{\mathtt{g}} \varphi(\mathtt{h}_i^{\mathtt{t}}) && \text{\# relevance}_2 \end{aligned} \tag{10}$$

$$\mathtt{d}_j^{\mathtt{t}} = \sum_{i=1}^{j-1} \Pr(\mathtt{y}_j^{\mathtt{t}} = 1 \mid \mathtt{R}_{\mathtt{t}}; \mathtt{S}_{\mathtt{t}}; \theta) \times \mathtt{h}_i^{\mathtt{t}} \tag{11}$$

The basic idea behind our attention mechanism is as follows: if a supporting sentence more resembles a candidate one, or overlaps less with the dynamic output, or is more relevant to the target document, then it can provide more contextual information to facilitate current decision on being extracted or not, thereby taking a higher weight in the generated context vector. This innovative attention will guide our goal related work section to maximize the representativeness of selected sentences (saliency & novelty), while minimizing the semantic distance to the target document (relevance). This is consistent with the way that scholars consume a reference collection, with the min-max objective in their minds.

4 Experiment

4.1 Experimental Setup

This section presents the experimental setup for assessing our approach, including 1) dataset used for training and testing, 2) implementation details, 3) contrast methods and evaluation metrics.

Dataset We conduct experiments on a dataset[2] created from the ACM digital library, where metadata and full texts are derived from PDF files. To be detailed, this dataset includes 371,891 papers, 779,810 authors, 9,204 keywords and 807 venues in total. Note that we ignore the keyword with frequency below a certain threshold, and adopt *greedy matching* of Guo et al. (2013) to generate pseudo keywords for papers lacking topic descriptions. For each target document, the references are traversed by the descending order of the cited number in related work section (primary) and in full paper (secondary) successively. We first apply a series of pre-processings such as lowercasing and stemming to standardize candidate sentences, then remove those which are too short/long ($<$ 7 or $>$ 80 words). On this basis, a total of 8,080 papers are selected to evaluate our approach, each containing more than 15 references found in the dataset and a related work section of at least 500 words. But as for the heterogeneous bibliography graph, all source data have to be imported to ensure the structural integrity of communities. Besides, this graph should be constructed year-by-year to preclude the effect of later publications on earlier ones.

Implementation We use *Tensorflow* for implementation, where both the dimensions of embedding and hidden state are equally 128. For the CNN, *word2vec* (Mikolov et al., 2013) is utilized to initialize the word embeddings, which can be further tuned during the training phase. Meanwhile, we follow the work of Kim (2014) to apply a list of kernels with widths $\{3, 4, 5\}$. As for the RNN, each LSTM module is set to one single layer, and all input documents are padded to the same length, along with a mark to indicate the real number of sentences. Based on these settings, we train our summarizer using *Adam* with the default in Kingma and Ba (2014), and perform *mini-batch cross-entropy* training with a batch of one target document for 20 epochs.

To create training data for our summarizer, each reference needs to be annotated with the ground truth in advance, i.e., candidate sentences are tagged with 0/1 for indicating summary-worthy or not. Specifically, we follow a heuristic practice of Cao et al. (2016) and Nallapati et al. (2016b) to compute ROUGE-2 score (Lin and Hovy, 2003) for each sentence, in terms of the native related work sections (gold standards). Next, those sentences with high scores are chosen as the positive samples, and the rest as the negative ones, such that the total score of selected sentences is maximized with respect to the gold standard. As for

---

[2] To help readers reproduce the experiment outcome, we share part of the experiment data while the copyrighted information is removed. `https://github.com/kuadmu/2018EMNLP`

testing, we relax the number of sentences to be selected, and focus on the classification probability from Eq. 8. In this study, *cross validation* is applied to split the dataset into ten parts equally at random, in which nine are used for training and the other one for testing.

Evaluation We adopt the widely used toolkit ROUGE (Lin and Hovy, 2003) to evaluate the summarization performance automatically. In particular, we report ROUGE-1 and ROUGE-2 (unigram and bigram overlapping) as a way to assess the informativeness, and ROUGE-L (the longest common subsequence) as a means to assess the fluency, in terms of fixed bytes of gold standards.

To validate the proposed attention mechanism, we compare our approach (denoted as $\text{P.}_{\text{S+N+Rteg+EUD}}$) against six variants, including: 1) P._{void}: a plain seq2seq summarizer without attentions; 2) P._{S}: use the saliency as an only attention factor; 3) $\text{P.}_{\text{S+N}}$: leverage both the saliency and novelty; 4) $\text{P.}_{\text{S+N+Rt}}$: incorporate the relevance from the textual context; 5) $\text{P.}_{\text{S+N+Rtog}}$: gain the relevance from the graphic context of a homogeneous citation graph; 6) $\text{P.}_{\text{S+N+Rteg}}$: utilize the heterogeneous bibliography graph, but with each edge type the same usefulness.

In addition, we also select six representative summarization methods as a benchmark group. The first one is the general seq2seq summarizer by Cheng and Lapata (2016), denoted as PointerNet, which employs an attention mechanism to extract sentences directly after reading them. Following are five classical generic solutions, including: 1) Luhn (Luhn, 1958): a heuristic summarization based on word frequency and distribution; 2) MMR (Carbonell and Goldstein, 1998): a diversity-based re-ranking to produce summaries; 3) LexRank (Erkan et al., 2004): a graph-based summary technique inspired by *PageRank* and *HITS*; 4) SumBasic (Nenkova and Vanderwende, 2005): a frequency-based summarizer with duplication removal; 5) NltkSum (Acanfora et al., 2014): a *natural language tookit* (NLTK)-based implementation for summarization.

For clarity, Luhn, LexRank and SumBasic are analogous to the work of Hu and Wan (2014) which extracts sentences scoring the highest in significance, and they are also contrasted in the latest studies on neural summarizers (Chen et al., 2016; Tan et al., 2017). Meanwhile, MMR often serves as a part/post-processing of existing techniques to avoid the redundancy (Cohan and Goharian, 2017), and we introduce NltkSum to investigate the impact of grammatical/semantic analysis to the automatic related work summarization. Note that former studies specially for this task require extensive human involvements (see Table 1), thus we cannot apply them to such a large dataset of this study.

4.2 Results and Discussion

Table 2 reports the evaluation comparison over ROUGE metrics. From the top half, all scores appear a gradual upward trend with incorporation of saliency, novelty, relevance (from both textual and graphic contexts) and EUD into consideration one after another, which demonstrates the validity of our attention mechanism for summarizing related work sections. To be specific, we further reach the following conclusions:

1) P._{void} vs. P._{S} vs. $\text{P.}_{\text{S+N}}$: Both saliency and novelty are two effective factors to locate the required content for summaries, which is consistent with prior studies.

2) $\text{P.}_{\text{S+N}}$ vs. $\text{P.}_{\text{S+N+Rt}}$: Contextual relevance does contribute to address the alignment between a related work section and its source documents.

3) $\text{P.}_{\text{S+N+Rt}}$ vs. $\text{P.}_{\text{S+N+Rtog}}$: Textual context alone cannot provide entire evidence to characterize the relationship among scientific publications exactly.

4) $\text{P.}_{\text{S+N+Rtog}}$ vs. $\text{P.}_{\text{S+N+Rteg}}$: Heterogeneous bibliography graph involves richer contextual information than a homogeneous citation graph.

5) $\text{P.}_{\text{S+N+Rteg}}$ vs. $\text{P.}_{\text{S+N+Rteg+EUD}}$: EUD plays an indispensable role in organizing accurate contextual relevance on a heterogeneous graph.

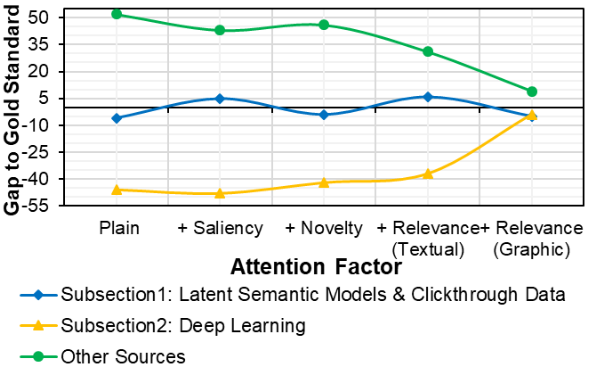

Figure 3: Number of extracted words on each reference cluster under different attention factors.

Continuing the "DSSM", Figure 3 visualizes the number of extracted words on each reference

Methods	ROUGE-1	ROUGE-2	ROUGE-L
$P._{\text{void}}$	26.85*	6.38*	14.22*
$P._{\text{S}}$	26.98*	6.48*	14.36*
$P._{\text{S+N}}$	27.29*	6.65*	14.43*
$P._{\text{S+N+Rt}}$	27.63*	6.72*	14.46*
$P._{\text{S+N+Rtog}}$	27.82*	7.00*	14.55*
$P._{\text{S+N+Rteg}}$	28.56*	7.40	14.70*
$P._{\text{S+N+Rteg+EUD}}$	**29.18**	**7.63**	**14.89**
Luhn	25.76*	5.08*	13.50*
MMR	25.55*	5.14*	13.99*
LexRank	25.07*	5.12*	13.95*
SumBasic	28.01*	5.44*	13.93*
NltkSum	28.07*	6.36*	14.87
PointerNet	27.06*	6.53*	14.41*

* indicates *Wilcoxon signed-rank test* $p < 0.01$, compared with $P._{\text{S+N+Rteg+EUD}}$

Table 2: Rouge evaluation (%) on 8,080 papers from ACM digital library.

cluster[3] under different attention factors. It can be seen that only after adding the relevance especially that from the graphic context into attentions, our summarizer can correctly sample the content from "Deep Learning" (yellow line), and eliminate that originated from "Other Sources" by a big margin (green line). As this example falls into the methodology transferring, a host of its involved word collocations are not idiomatic combinations yet, such as "Deep Neural Network" co-occurs with "Clickthrough Data" that is more frequently related to "Latent Semantic Analysis" at that time, which results in a somewhat biased textual context. By contrast, the graphic context will suffer less from this bias because it characterizes the connectivity patterns (real-time setup) instead of n-gram statistics, thus offering a more robust measure for the contextual relevance.

The bottom half of Table 2 illustrates the superiority of our approach over six representative summarization methods. Above all, Luhn, LexRank and MMR three summarizers that simply exploit shallow text features (word frequency and associated sentence similarity) to measure either significance or redundancy fall far behind the plain variant $P._{\text{void}}$, which partly reflects the strength of seq2seq paradigm in summarizing a related work section. Second, with combination of significance and redundancy, SumBasic achieves a drastic increase on ROUGE-1 and a mild raise on ROUGE-2 respectively, but it still cannot improve ROUGE-L marginally. This is because simple text statistics cannot present deeper levels of natural language understanding to catch larger-grained units of co-occurrence. Third, NltkSum benefits from a NLTK library so as to access grammatical/semantic supports, thereby having the best informativeness (ROUGE-1 and ROUGE-2) among the five generic baselines, and meanwhile a comparable fluency (ROUGE-L) with our approach. Finally, as a deep learning solution, although PointerNet takes both hidden states and previously labeled sentences into account, at each decoding step it focuses on only current and just one previous sentences, lacking a comprehensive consideration on saliency, novelty and more importantly the contextual relevance ($< P._{\text{S+N}}$).

To better verify the summarization performance, we also conduct a human evaluation on 35 papers containing more than 30 references in the dataset. We assign a number of raters to compare each generated related work section against the gold standard, and judge by three independent aspects as: 1) How compliant is the related work section to the target document? 2) How intuitive is the related work section for readers to grasp the key content? 3) How useful is the related work section for researchers to prepare their final literature reviews? Note that we do not allow any ties during the comparison, and each property is assessed with a 5-point scale of 1 (worst) to 5 (best).

Table 3 displays how often raters rank each summarizer as the 1st, 2nd and so on, in terms of

---

[3] We pack the references cited in the same subsection of the related work section as one reference cluster.

Methods	1st	2nd	3rd	4th	5th	6th	7th	Mean Ranking
Luhn	0.04	0.07	0.09	0.13	0.17	0.23	0.29	5.26
MMR	0.05	0.07	0.11	0.16	0.19	0.22	0.20	4.82
LexRank	0.06	0.09	0.11	0.14	0.17	0.19	0.27	4.93
SumBasic	0.09	0.13	0.18	0.18	0.18	0.15	0.10	4.10
NltkSum	0.21	0.21	0.20	0.15	0.10	0.07	0.04	3.00
PointerNet	0.14	0.20	0.18	0.15	0.13	0.11	0.08	3.54
$\text{P}_{\text{S+N+Rteg+EUD}}$	0.40	0.22	0.14	0.09	0.06	0.04	0.02	2.34

Table 3: Human evaluation (proportion) on 35 papers with more than 30 references in the dataset.

best-to-worst. Specifically, our approach comes the 1st on 40% of the time, which is followed by NltkSum that is considered the best on 21% of the time (almost half of ours), and PointerNet with quite equal proportions on each ranking. Furthermore, the other four summarizers account for obviously lower ratings in general. To attain the statistical significance, *one-way analysis of variance* (ANOVA) is performed on the obtained ratings, and the results show that our approach is better than all six contrast methods significantly ($p < 0.01$), which means that the conclusion drawn by Table 2 is sustained.

5 Conclusion

In this paper, we highlight the contextual relevance for the automatic related work summarization, and analyze the graphic context to characterize the relationship among scientific publications accurately. We develop a neural data-driven summarizer by leveraging the seq2seq paradigm, where a joint context-driven attention mechanism is proposed to measure the contextual relevance within full texts and a heterogeneous bibliography graph simultaneously. Extensive experiments demonstrate the validity of the proposed attention mechanism, and the superiority of our approach over six representative summarization baselines.

In future work, an appealing direction is to organize the selected sentences in a logical fashion, e.g., by leveraging a topic hierarchy tree to determine the arrangement of the related work section (Cong and Kan, 2010). We also would like to take the citation sentences of each reference into consideration, which is another concise and universal data source for scientific summarization (Chen and Hai, 2016; Cohan and Goharian, 2017). At the end of this paper, we believe that extractive methods are by no means the final solutions for literature review generation due to plagiarism concerns, and we are going to put forward a fully abstractive version in further studies.

Acknowledgement

We would like to thank the anonymous reviewers for their valuable comments. This work is partially supported by the National Science Foundation of China under grant No. 71271034.

References

Joseph Acanfora, Marc Evangelista, David Keimig, and Myron Su. 2014. Natural language processing: generating a summary of flood disasters. *Cell*, 41(2):383–94.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *Proceedings of the 41st IEEE ICASSP International Conference on Acoustics, Speech and Signal Processing, Shanghai, China*, pages 4945–4949.

Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. 2010. Music recommendation by unified hypergraph:combining social media information and music content. In *Proceedings of the ACM SIGMM International Conference on Multimedia, Amsterdam, Netherlands*, pages 391–400.

Ziqiang Cao, Wenjie Li, Sujian Li, Furu Wei, and Yanran Li. 2016. Attsum: Joint learning of focusing and summarization with neural attention. *arXiv preprint arXiv:1604.00125*.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, USA*, pages 335–336.

Jingqiang Chen and Zhuge Hai. 2016. Summarization of related work through citations. In *Proceedings of the 12th IEEE SKG International Conference on Semantics, Knowledge and Grids, Beijing, China*, pages 54–61.

Qian Chen, Xiaodan Zhu, Si Wei, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the ACM IJCAI International Joint Conference on Artificial Intelligence, New York, USA*, pages 2754–2760.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th ACL Annual Meeting of the Association for Computational Linguistics, Berlin, Germany*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the NAACL Conference of the North American Chapter of the Association for Computational Linguistics, San Diego, USA*, pages 93–98.

Arman Cohan and Nazli Goharian. 2017. Scientific article summarization using citation-context and article's discourse structure. *arXiv preprint arXiv:1704.06619*, pages 390–400.

Ronan Collobert, Jason Weston, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(1):2493–2537.

Duy Vu Hoang Cong and Min Yen Kan. 2010. Towards automated related work summarization. In *Proceedings of the 23rd ACM COLING International Conference on Computational Linguistics, Beijing, China*, pages 427–435.

Swagatam Das and Ponnuthurai Nagaratnam Suganthan. 2011. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31.

Erkan, Radev, and R Dragomir. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Qiqihar Junior Teachers College*, 22:2004.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, Usa*, pages 855–864.

Chun Guo and Xiaozhong Liu. 2015. Automatic feature generation on heterogeneous graph for music recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile*, pages 807–810.

Chun Guo, Jinsong Zhang, and Xiaozhong Liu. 2013. Scientific metadata quality enhancement for scholarly publications. *Ischools*.

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

Yue Hu and Xiaojun Wan. 2014. Automatic generation of related work sections in scientific papers: an optimization approach. In *Proceedings of the ACL EMNLP Conference on Empirical Methods in Natural Language Processing, Doha, Qatar*, pages 1624–1633.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM CIKM international Conference on Information & Knowledge Management, San Francisco, USA*, pages 2333–2338.

Scott Jensen, Xiaozhong Liu, Yingying Yu, and Stasa Milojevic. 2016. Generation of topic evolution trees from heterogeneous bibliographic networks. *Journal of Informetrics*, 10(2):606–621.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Eprint Arxiv*.

Diederik Kingma and Jimmy Ba. 2014. Adam: a method for stochastic optimization. *Computer Science*.

Chin Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the NAACL The Annual Conference of the North American Chapter of the Association for Computational Linguistics, Stroudsburg, USA*, pages 71–78.

H. P. Luhn. 1958. *The automatic creation of literature abstracts*. IBM Corp.

William C. Mann and Sandra A. Thompson. 2009. Rhetorical structure theory: Toward a functional theory of text organization. *Text & Talk*, 8(3):243–281.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computer Science*.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016a. Sequence-to-sequence rnns for text summarization. In *Proceedings of the International Conference on Learning Representations, Workshop track, San Juan, Puerto Rico*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016b. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *arXiv preprint arXiv:1611.04230v1*.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira Dos Santos, Caglar Gulcehre, and Bing Xiang. 2016c. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023v5*.

Ramesh M. Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. 2008. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Usa*, pages 542–550.

Shashi Narayan, Nikos Papasarantopoulos, Shay B. Cohen, and Mirella Lapata. 2017. Neural extractive summarization with side information. *arXiv preprint arXiv:1704.04530*.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research*.

L Page. 1998. The pagerank citation ranking : Bringing order to the web, online manuscript. *Stanford Digital Libraries Working Paper*, 9(1):1–14.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the ACL EMNLP Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal*, pages 379–389.

Ananthram Swami, Ananthram Swami, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, Canada*, pages 135–144.

Jiwei Tan, Xiaojun Wan, Jianguo Xiao, Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th ACL Annual Meeting of the Association for Computational Linguistics, Vancouver, Canada*, pages 1171–1181.

Scott White and Padhraic Smyth. 2003. Algorithms for estimating relative importance in networks. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, USA*, pages 266–275.

Dwi H Widyantoro and Imaduddin Amin. 2014. Citation sentence identification and classification for related work summarization. In *Proceedings of the ICACSIS International Conference on Advanced Computer Science and Information Systems*, pages 291–296.

Yingying Yu, Xiaozhong Liu, and Zhuoren Jiang. 2015. Random walk and feedback on scholarly network. In *Proceedings of the 1st ACM GSB@SIGIR International Workshop on Graph Search and Beyond, Santiago, Chile*, pages 33–37.

Improving Neural Abstractive Document Summarization with Explicit Information Selection Modeling*

Wei Li[1,2,3] **Xinyan Xiao**[2] **Yajuan Lyu**[2] **Yuanzhuo Wang**[1]
[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]Baidu Inc., Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
weili.ucas.ict@gmail.com, {xiaoxinyan,lvyajan}@baidu.com,
wangyuanzhuo@ict.ac.cn

Abstract

Information selection is the most important component in document summarization task. In this paper, we propose to extend the basic neural encoding-decoding framework with an information selection layer to explicitly model and optimize the information selection process in abstractive document summarization. Specifically, our information selection layer consists of two parts: gated global information filtering and local sentence selection. Unnecessary information in the original document is first globally filtered, then salient sentences are selected locally while generating each summary sentence sequentially. To optimize the information selection process directly, distantly-supervised training guided by the golden summary is also imported. Experimental results demonstrate that the explicit modeling and optimizing of the information selection process improves document summarization performance significantly, which enables our model to generate more informative and concise summaries, and thus significantly outperform state-of-the-art neural abstractive methods.

1 Introduction

Document summarization is the task of generating a fluent and condensed summary for a document while retaining the gist information. There are two prominent approaches: extractive methods and abstractive methods. Extractive methods generate summary for a document by directly selecting salient sentences from the original document. On the contrary, abstractive methods synthesize information from the input document to generate summary using arbitrary words and expressions - as human usually do. Recent neural models enable an end-to-end framework for natural language generation, which inspires the research on abstractive document summarization.

Most existing work directly apply the neural encoding-decoding framework, which first encodes the input into an abstract representation and then decodes the output based on the encoded information. Although the encoding-decoding framework has achieved huge success on some text generation tasks like machine translation (Bahdanau et al., 2014) and image caption (Vinyals et al., 2015), the performance on abstractive document summarization is much less convincing. Since document summarization is a special natural language generation task that requires information selection, the performance of current neural abstractive methods even has a considerable gap from extractive methods.

The most essential prerequisite for a practical document summarization system is that the generated summary must contain the salient information of the original document. Since a document is a long sequence of multiple sentences, both global document information and local inter-sentence relations need to be properly modeled in the information selection process. Although the encoding-decoding framework has implicitly modeled the information selection process via end-to-end training, we argue that abstractive document summarization shall benefit from explicitly modeling and optimizing it by capturing both the global document information and local inter-sentence relations.

In this paper, we propose to extend the encoding-decoding framework to model the information selection process explicitly. We treat the document summarization as a three-phase task: document encoding, information selection and summary decoding. Correspondingly, our model consists of three layers: a document encoder layer, an information selection layer and a

---

*This work was done while the first author was doing internship at Baidu Inc.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1787–1796
Brussels, Belgium, October 31 - November 4, 2018.

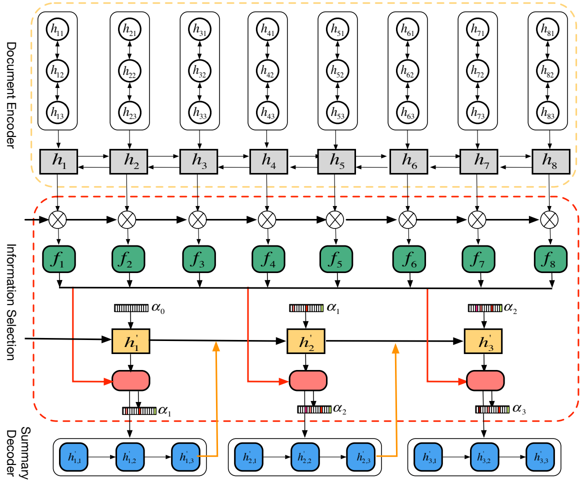

Figure 1: Our abstractive document summarization model, which mainly consists of three layers: document encoder layer (the top part), information selection layer (the middle part) and summary decoder layer (the bottom part).

summary decoder layer, as shown in Figure 1. In our model, both the document and summary are processed sentence by sentence, to better capture the inter-sentence relations. The information selection layer consists of two parts: gated global information filtering and local sentence selection. Unnecessary information in the original document are first globally filtered by a gated network, then important sentences are selected locally while generating each summary sentence sequentially. Moreover, we propose to optimize the information selection process with distantly-supervised training. Our proposed method combines the strengths of extractive methods and abstractive methods, which is able to tackle the factors of saliency, non-redundancy, coherence and fluency under a unified framework. We conduct extensive experiments on benchmark datasets and the results demonstrate that the explicit modeling and distantly-supervised optimizing of the information selection process improves document summarization performance significantly, which enables our model to significantly outperforms previous state-of-the-art neural abstractive methods.

2 Our Model

As shown in Figure 1, our model consists of a hierarchical document encoder, an information selection layer and an attention-equipped decoder. Firstly, the hierarchical encoder encodes the document sentence by sentence, and word by word in each sentence. Then the information selection layer selects and filters the sentence representations based on the global document representation. A sentence selection RNN is used to select salient and relevant sentences while generating each summary sentence sequentially based on the tailored sentence representations. At last, the summary decoder produces the output summary to paraphrase and generalize the selected sentences.

In the following, we denote h_i, $h_{i,j}$ as the hidden state of the i-th sentence and the j-th word of the i-th sentence in the document encoder part, respectively. In the information selection and summary decoder part, we denote h'_t, $h'_{t,k}$ as the hidden state of the t-th summary sentence and the k-th word in the t-th summary sentence, respectively.

2.1 Document Encoder

A document d is a sequence of sentences $d = \{s_i\}$, and each sentence is a sequence of words $s_i = \{w_{i,j}\}$. A hierarchical encoder, which consists of two levels: word level and sentence level similar to (Nallapati et al., 2016), is used to encode the document from both word and sentence level.

The word-level encoder is a bidirectional Gated Recurrent Unit (GRU) (Chung et al., 2014), which encodes the words of a sentence into sentence representation. The word encoder sequentially updates its hidden state after receiving a word, which is formulated as:

$$h_{i,j} = BiGRU(h_{i,j-1}, e_{i,j}) \qquad (1)$$

where $h_{i,j}$ and $e_{i,j}$ denotes the hidden state and embedding of word $w_{i,j}$, respectively.

The concatenation of the forward and backward final hidden states in the word-level encoder is indicated as the vector representation x_i of the sentence s_i, which is used as input to the sentence-level encoder. The sentence encoder is also a bidirectional GRU, which updates its hidden state after receiving each sentence representation by:

$$h_i = BiGRU(h_{i-1}, x_i) \qquad (2)$$

where h_i denotes the hidden state of sentence s_i.

The concatenation of the forward and backward final states in the sentence-level encoder is used as the vector representation of document $\hat{\mathbf{d}}$.

2.2 Information Selection

Document summarization is a special natural language generation task which requires information compression. It needs to remove the unnecessary information and select salient information from the input document to produce a condensed summary. However, it is difficult for the basic encoder-decoder framework to learn the process of salient information selection, which has also been noticed by several previous work (Tan et al., 2017a,b). To tackle the challenge, we extend the basic encoder-decoder framework by adding an information selection layer to model the information selection process explicitly. Our information selection layer consists of two parts: gated global information filtering that used to remove the unnecessary information of a document, and local sentence selection that used to select salient sentences from a document sequentially to produce summary sentences.

Gated Global Information Filtering

Inspired by studies on how human write text summaries by first skimming the document and deleting unnecessary material (Brown and Day, 1983), we design a gated global information filtering network to filter unnecessary information of a document based on the global document representation before the summary decoder generates summary. Concretely, the gated information filtering network makes use of the document representation $\hat{\mathbf{d}}$, which represents the global information of a document, to filter sentences based on the sentence representation h_i.

For each source sentence s_i, the gate network takes the document representation $\hat{\mathbf{d}}$ and sentence representation h_i as inputs to compute the gate vector g_i:

$$g_i = \sigma(W_g h_i + U_g \hat{\mathbf{d}} + b_g) \tag{3}$$

where W_g and U_g denote weight matrices, b_g the bias vector, and σ the sigmoid activation function.

Then each sentence s_i can be filtered by the gate vector g_i as follows:

$$f_i = h_i \odot g_i \tag{4}$$

where f_i indicates the representation of sentence s_i after information filtering, and $\odot$ denotes element-wise multiplication.

Note that, we filter sentences in micro semantic dimensions rather than filtering whole sentences. The tailored sentence representations are used as input to the sentence selection network and summary decoder, which can help to detect salient sentences and improve informativeness of the generated summary.

Local Sentence Selection

We explicitly model the local sentence selection process which selects several target sentences to generate a summary sentence. Concretely, we apply a RNN layer to sequentially select target sentences for each summary sentence, shown as in Figure 1. The sentence-selection RNN uses the document representation $\hat{\mathbf{d}}$ as initial state $h_0^{'}$, and sequentially predicts the sentence selection vector α_t as follows:

$$\alpha_t^i = \frac{e^{\phi(f_i, h_t')}}{\sum_l e^{\phi(f_l, h_t')}} \tag{5}$$

$$\phi(f_i, h_t') = v^T tanh(W_f f_i + W_h h_t' + b). \tag{6}$$

where α_t^i indicates the weight of source sentence s_i when generating the t-th summary sentence, and $h_t^{'}$ denotes the hidden state of sentence selection layer when generating the t-th summary sentence. v, W_f and W_h are weight matrices, and b is the bias vector. Note that, the sentence selection vector α_t is computed based on the tailored sentence representation f_i.

The sentence-selection RNN uses a single unidirectional GRU, which updates its state by:

$$h_t^{'} = GRU(h_{t-1}^{'}, x_t^{'}) \tag{7}$$

where $x_t^{'}$ denotes the input of current sentence-selection step. $x_t^{'}$ combines both the previous sentence selection vector α_{t-1} and the encoded representation of previous generated sentence $r_{t-1}^{'}$ by $x_t^{'} = tanh(W_r r_{t-1}^{'} + W_\alpha \alpha_{t-1} + b_x)$, where W_r, W_α, and b_x denote learnable parameters.

The representation of the selected source sentences is computed by:

$$q_t = \sum_j \alpha_t^j f_j \tag{8}$$

which is used as initial state of the summary decoder to generate a summary sentence to paraphrase and generalize the selected sentences.

2.3 Summary Decoder

On top of the document encoder and the information selection layer, we use GRU with attention as the summary decoder to realize each summary sentence word by word.

At each word decoding step k in the t-th summary sentence, the GRU reads the previous word embedding $e_{t,k-1}$ and context vector $c_{t,k-1}$ as inputs to compute the new hidden state $h'_{t,k}$ by:

$$h'_{t,k} = GRU(h'_{t,k-1}, c_{t,k-1}, e_{t,k-1}) \tag{9}$$

We import attention mechanism to help locate relevant words to be copied or paraphrased within the selected source sentences in each word generation step. The attention distribution $\beta^i_{t,k}$ of the kth word of the tth summary sentence over the sentences in the ith document can be computed as:

$$\beta^{i,j}_{t,k} = \alpha^i_t \frac{e^{\phi(h_{i,j}, h'_{t,k})}}{\sum_l e^{\phi(h_{i,l}, h'_{t,k})}} \tag{10}$$

where α^i_t denotes the weight of the ith source sentence, used to normalize the word attention distributions. Then the word-level context vector when generating the kth word at the tth sentence generation step can be computed as: $\mathbf{c}_{t,k} = \sum_i \sum_j \beta^{i,j}_{t,k} h_{i,j}$, which is also incorporated into the word decoder.

At each word generation step, the vocabulary distribution is calculated from the context vector $\mathbf{c}_{t,k}$ and the decoder state $h'_{t,k}$ by:

$$P_{vocab}(w'_{t,k}) = softmax(W_v(W_c[h'_{t,k}, c_{t,k}] + b_c) + b_v) \tag{11}$$

where W_v and W_c are learned parameters. The copy mechanism based on the word attention is also imported into the decoder to alleviate the OOV problems as in (See et al., 2017).

2.4 Model Learning with Distant Supervision

Despite the end-to-end training for the performance of generated summary, we also directly optimize the sentence selection decisions by importing supervision for the sentence selection vector α_t in Equation 5. While there is no explicit supervision for sentence selection, we define a simple approach for labeling sentences based on the reference summaries. To simulate the sentence selection process on human-written abstracts, we compute the words-matching similarities (based on TF-IDF cosine similarity) between a reference-summary sentence and corresponding source document sentences and normalize them into distantly-labelled sentence selection vector p_t. Then the sentence selection loss is defined as:

$$loss_{sel} = \sum_t D_{KL}(\alpha_t, p_t) \tag{12}$$

where $D_{KL}(\alpha_t, p_t)$ indicates the KL-divergence between distribution α_t and p_t. The sentence selection loss is imported into the final loss function to be optimized with the summary generation component together.

The loss function $\mathcal{L}$ of the model is the mix of the negative log-likelihood of generating summaries over training set $\mathcal{T}$, and the sentence selection loss of distantly-supervised training:

$$\mathcal{L} = \sum_{(X,Y)\in\mathcal{T}} -logP(Y|X;\theta) + \lambda loss_{sel} \tag{13}$$

where λ is a hyper-parameter tuned on the validation set. (X, Y) denotes a document-summary pair in the training set.

3 Experiments

3.1 Dataset

We conduct our experiments on a large-scale corpus of *CNN/DailyMail*, which has been widely used for exploration on summarizing documents with multi-sentence summaries. The corpus are originally constructed in (Hermann et al., 2015) by collecting human generated highlights from news stories in the CNN and DailyMail Website, which contains input document of about 800 tokens on average and multi-sentence summaries of up to 200 tokens. We use the same version of data with (See et al., 2017), which totally has 280,125 training pairs, 13,367 validation pairs and 11,489 test pairs after discarding the examples with empty article text. Some of previous work (Nallapati et al., 2016, 2017; Paulus et al., 2017; Tan et al., 2017a) use the anonymized version of data, which has been pre-processed to replace each named entity with an unique identifier. By contrast, we use the non-anonymized data similar to (See et al., 2017), which is a more favorable and challenging problem because it requires no pre-processing.

3.2 Implementation Details

Model Parameters For all experiments, the word-level encoder and summary decoder both use 256-dimensional hidden states, and the sentence-level encoder and sentence selection network both use 512-dimensional hidden states. We use pre-trained Glove (Pennington et al., 2014) vector for initialization of word embeddings. The dimension of word embeddings is 100, which will be further trained in the model. We use a vocabulary of 50k words for both encoder and decoder.

Method	Rouge-1	Rouge-2	Rouge-L
Lead-3	40.34	17.70	36.57
SummaRuNNer-abs	37.5	14.5	33.4
SummaRuNNer	39.6	16.2	35.3
Seq2seq-baseline	36.64	15.66	33.42
ABS-temp-attn	35.46	13.30	32.65
Graph-attention	38.1	13.9	34.0
Deep-reinforced	39.87	15.82	**36.90**
Coverage	39.53	17.28	36.38
Our Model	**41.54**	**18.18**	36.47

Table 1: Rouge F_1 scores on the test set. All our ROUGE scores have a **95% confidence interval of at most ± 0.25** as reported by the official ROUGE script.

Method	Informat.	Concise	Coherent	Fluent
Lead-3	3.49^*	3.19^*	3.86	4.07^*
Seq2seq-b.	3.11^*	2.95^*	3.08^*	3.51^*
Coverage	3.41^*	3.25^*	3.37	3.72
Our Model	3.76	3.49	3.65	3.80

Table 2: Human evaluation results. * indicates the difference between **Our Model** and other models are statistic significant ($p < 0.1$) by two-tailed t-test.

We use dropout (Srivastava et al., 2014) with probability $p = 0.5$. After tuning on the validation set, parameter λ is set as 0.2.

Model Training We use Adagrad (Duchi et al., 2011) algorithm with learning rate 0.1 and an initial accumulator value of 0.1 to optimize the model parameters θ. During training, we use gradient clipping with a maximum gradient norm of 2. Our model is trained on a single Tesla K40m GPU with a batch size of 16 and an epoch is set containing 10,000 randomly sampled documents. Convergence is reached within 300 epochs.

Hierarchical Beam Search To improve information correctness and avoid redundancy during the summary decoding process, we use the hierarchical beam search algorithm with reference mechanism (Tan et al., 2017a) to generate multi-sentence summaries. Similar to (Tan et al., 2017a), the beam sizes for word decoder and sentence decoder are 15 and 2, respectively.

3.3 Baselines

We compare our system with the results of state-of-the-art neural summarization approaches reported in recent papers, which contain both abstractive models and extractive models. The extractive models include **SummaRuNNer** (Nallapati et al., 2017), while **SummaRuNNer-abs** is similar to SummaRuNNer but is trained directly on the abstractive summaries. **Lead-3** is a strong extractive baseline which uses the first 3 sentences of the document as summary. The abstractive models include:

1) **Seq2seq-baseline**, which uses the basic seq2seq encoder-decoder structure with attention mechanism and incorporates with the copy mechanism as in (See et al., 2017).
2) **ABS-temp-attn** (Nallapati et al., 2016), which uses *Temporal Attention* on the seq2seq architecture to overcome the repetition problem.
3) **Graph-attention** (Tan et al., 2017a), which uses a graph-ranking based attention mechanism based on a hierarchical architecture to identify important source sentences.
4) **Deep-reinforced** (Paulus et al., 2017), which trains the seq2seq encoder-decoder model with reinforcement learning techniques.
5) **Coverage** (See et al., 2017), which is an extension of the Seq2seq-baseline model by importing coverage mechanism to control repetitions in summary.

3.4 Evaluation

ROUGE Evaluation

We evaluate our models with the standard ROUGE metric (Lin, 2004) and obtain ROUGE scores using the `pyrouge` package. Results in Table 1 show that our method has significant improvement over state-of-the-art neural abstractive baselines as well as extractive baselines. Note that, the **Deep-reinforced** model achieves the best ROUGE-L performance because it directly optimizes the ROUGE-L metric. Comparing with the current state-of-the-art model **Coverage**, our model achieves significant better performance on ROUGE-1 and ROUGE-2 metrics, and comparable performance on ROUGE-L metric, which demonstrates that our model is more effective in selecting salient information from a document to produce an informative summary while keeping the ability to generate fluent and correct sentences.

Human Evaluation with Case Analysis

In addition to the ROUGE evaluation, we also conducted human evaluation on 50 random samples from CNN/DailyMail test set and compared the summaries generated by our method with the outputs of **Lead-3**, **Seq2seq-baseline** and **Coverage**. Three data annotators were asked to compare the generated summaries with the human summaries, and assess each summary from four independent

Gold Reference: faith and hope howie were born with one body and two faces on may 8 . they tragically died in hospital just 19 days after they were born . parents simon howie and renee young visit their grave at pinegrove in western sydney fortnightly . they arrived on thursday to find the grave bare of all the girls ' mementos . staff had cleared entire baby section and thrown belongings in rubbish .
Seq2Seq-baseline: faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition known as disrosopus . faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition known as disrosopus . faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition known as disrosopus .
Coverage: faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition known as disrosopus . they died in hospital less than a month after they were born and their parents , simon howie and renee young , laid them to rest at pinegrove memorial park in sydney 's west.
Our Model: faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition known as disrosopus. they died in hospital less than a month after they were born and their parents , simon howie and renee young , laid them to rest at pinegrove memorial park in sydney 's west. family members have visited the grave every week to leave mementos and flowers for faith and hope , but when mr howie and ms young arrived on thursday they found the site completely bare .

Table 3: Examples of generated summaries. The Seq2Seq-baseline model generates repeated sentences and loses salient information. The Coverage model reduces repetitions, but also loses salient information. Our model can select more salient information from the original document and generate more informative summary.

perspectives: (1) **Informative**: How informative the summary is? (2) **Concise**: How concise the summary is? (3) **Coherent**: How coherent (between sentences) the summary is? (4) **Fluent**: How fluent, grammatical the sentences of a summary are? Each property is assessed with a score from 1(worst) to 5(best) by three annotators. The average results are presented in Table 2.

The results show that our model consistently outperforms the **Seq2seq-baseline** model and the previous state-of-the-art method **Coverage**. An example of comparison of the generated summaries by our model with the two abstractive models (w.r.t the reference summary) is shown in Table 3[1]. The summary generated by **Seq2Seq-Baseline** usually contains repetition of sentences, which seriously affects its informativeness, conciseness as well as coherence. For example, the sentence "*faith and hope howie were dubbed the miracle twins when they were born ...*" is repeated three times in Table 3. The **Coverage** model effectively alleviates the information repetition problem, however, it loses some salient information that should be included in the summary. For example, the information about "**mementos**" and "**family members visit the grave**" is lost in the example shown in Table 3. The summary generated by our method obviously contains more salient information, which shows the effectiveness of the information selection component in our model. According to the results in Table 2, the sentence-level modeling of document and summary in our model also makes the generated summaries achieve better inter-sentence coherence. Compared with the strong extractive baseline **Lead-3**, our model is able to generate more informative and concise summaries, which shows the advantage of abstractive methods. The fluency scores also show the good ability of our model to generate fluent and grammatical sentences. The human evaluation results demonstrate that our model is able to generate more informative, concise and coherent summaries than the baselines.

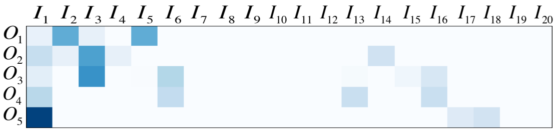

(a) Gold Reference

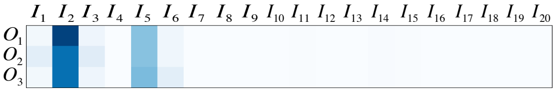

(b) Seq2Seq-baseline

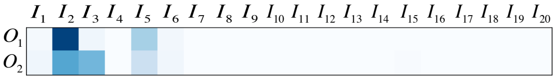

(c) Coverage

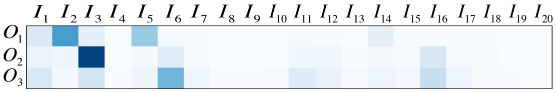

(d) Our Model

Figure 2: Visualization of sentence selection vectors. I_i and O_i indicate the i-th sentence of the input and output, respectively. Obviously, our model can detect more salient sentences that are included in the reference summary.

The visualization of the sentence selection vectors of the gold reference summary and the three abstractive models when generating the presented examples in Table 3 are shown in Figure 2[2]. The figure shows that **Seq2Seq-baseline** fails to detect all important source sentences and attend to the same sentences repeatedly, which result in generating repeated summary sentences. **Coverage** learns to reduce repetitions, but fails to detect all the salient information. Obviously, our method is more effective in selecting salient and relevant source sentences from the document to generate more informative summary. Furthermore, our

---

[1] More examples are shown in the supplementary material

[2] The sentence selection vectors of the Seq2seq-baseline mode and the Coverage model are computed by summing the attention weights of all words in each sentence and then normalized across sentences.

Method	Rouge-1	Rouge-2	Rouge-L
Our Model	**41.54**	**18.18**	**36.47**
– *distS*	40.02	17.54	34.87
– *distS&gateF*	39.26	16.96	33.92
– *infoSelection*	36.64	15.66	33.42

Table 4: Comparison results of removing different components of our method.

method tends to focus on different sets of source sentences when generating different summary sentences. The results verify that the information selection component in our model significantly improves the information selection process in document summarization.

4 Discussion

In this section, we first validate the effectiveness of each component of our model, then compare the performance of information selection of our method with several extractive methods, and finally analyze the effects of golden summary length on the performance of our model.

4.1 Model Validation

To further verify the effectiveness of each component in our model, we conduct several ablation experiments. "– *distS*" denotes removing the distant supervision for sentence selection (set λ as 0). "– *distS&gateF*" denotes removing both the distant supervision for sentence selection training and the global gated information filtering component. "– *infoSelection*" denotes removing the whole information selection layer and do not explicitly modeling the information selection process, which is actually the Seq2seq-baseline model.

Results on the test set are shown in Table 4. Our method much outperforms all the comparison systems and removing each component of our model one by one will leads to sustained significant performance declining, which verifies the effectiveness of each component in our model. The global gated information filtering network removes unnecessary information from the original document and helps generate more informative summary. The distantly-supervised training for sentence selection decisions helps the model learn to detect important and relevant source sentences for each summary sentence. The results verify that explicitly modeling the information selection process significantly improves the document summarization performance.

Method	Rouge-1	Rouge-2	Rouge-L
SummaRuNNer-abs	37.5	14.5	33.4
SummaRuNNer	39.6	16.2	35.3
OurExtractive	**40.41**	**18.30**	36.30
– *distS*	37.06	16.55	33.23
– *distS&gateF*	36.25	16.22	32.59

Table 5: Comparsion results of sentence selection.

length	Method	Rouge-1	Rouge-2	Rouge-L
< 75 (81.82%)	Our Mod.	**39.90**	**16.91**	35.19
	Coverage	38.90	16.81	**35.82**
$[75, 100)$ (12.64%)	Our Mod.	**47.13**	**22.44**	**40.81**
	Coverage	42.89	19.72	39.41
$[100, 125)$ (4.00%)	Our Mod.	**50.49**	**24.23**	**43.68**
	Coverage	41.78	19.00	38.41
> 125 (1.54%)	Our Mod.	**50.25**	**23.98**	**41.19**
	Coverage	39.57	17.93	36.33

Table 6: Comparison results w.r.t different length of reference summary. < 75 indicates the reference summary has less than 75 words (occupy 81.82% of test set), $[75, 100)$ denotes the number of words in reference summary is between 75 and 100 (occupy 12.64% of test set).

4.2 Effectiveness of Information Selection

To verify the performance of sentence selection in our model, we add a comparison system **OurExtractive** which is almost the same as our model, but replaces the summary decoder by a sentence extractor. The sentence extractor extracts the source sentence with the largest weight in each sentence generation step. "– *distS*" denotes removing the distant supervision for sentence selection training in our model. "– *distS&gateF*" denotes removing both the distant supervision for sentence selection training and the gated global information filtering component.

Results in Table 5 show that our simple extractive method **OurExtractive** significantly outperforms state-of-the-art neural extractive baselines, which demonstrates the effectiveness of the information selection component in our model. Moreover, **OurExtractive** significantly outperforms the two comparison systems which remove different components of our model one by one. The results show that both the gated global information filtering and distant supervision training are effective for improving information selection in document summarization. Our proposed method effectively combines the strengths of extractive methods and abstractive methods into a unified framework.

4.3 Effects of Summary Length

We further compare our method with the **Coverage** model by evaluating them on the test set with different length of golden reference summaries. The results are shown in Table 6, which demonstrate that our method is better at generating long summary for long document. As the golden summary becoming longer, our system will obtain larger advantages over the baseline (from +1.0 Rouge-1, +0.1 Rouge-2 and -0.63 Rouge-L for summary less than 75 words, rising to +10.68 Rouge-1, +6.05 Rouge-2 and +4.86 Rouge-L for summaries more than 125 words). The results also verify that our method is more effective in selecting salient information from documents, especially for long documents.

5 Related Work

Existing exploration on document summarization mainly can be categorized to **extractive methods** and **abstractive methods**.

5.1 Extractive Summarization Methods

Neural networks have been widely investigated on extractive document summarization task. Earlier work attempts to use deep learning techniques to improve sentence ranking or scoring (Cao et al., 2015a,b; Yin and Pei, 2015). Some recent work solves the sentence extraction and document modeling in an end-to-end framework. Cheng and Lapata (2016) propose an encoder-decoder approach where the encoder hierarchically learns the representation of sentences and documents while an attention-based sentence extractor extracts salient sentences sequentially from the original document. Nallapati et al. (2017) propose a recurrent neural network-based sequence-to-sequence model for sequential labelling of each sentence in the document. Neural models are able to leverage large-scale corpora and achieve better performance than traditional methods.

5.2 Abstractive Summarization Methods

As the seq2seq learning with neural networks achieve huge success in sequence generation tasks like machine translation, it also shows great potential in text summarization area, especially for abstractive methods. Some earlier researches studied the use of seq2seq learning for abstractive sentence summarization (Takase et al., 2016; Rush et al., 2015; Chopra et al., 2016). These models are trained on a large corpus of news documents which are usually shortened to be the first one or two sentences, and their headlines.

Later, some work explored the seq2seq models on document summarization, which produce a multi-sentence summary for a document. The seq2seq models usually exhibit some undesirable behaviors, such as inaccurately reproducing factual details, unable to deal with out-of-vocabulary (OOV) words and repetitions. To alleviate these issues, copying mechanism (Gu et al., 2016; Gulcehre et al., 2016; Nallapati et al., 2016) has been incorporated into the encoder-decoder architecture. Distraction-based attention model (Chen et al., 2016) and coverage mechanism (See et al., 2017) have also been investigated to alleviate the repetition problem. To better train the seq2seq model on tasks with long documents and multi-sentence summaries, a deep reinforced model was proposed to combine the standard words predication with teacher forcing learning and the global sequence prediction training with reinforcement learning (Paulus et al., 2017). Recently, Tan et al. (2017a) propose to leverage the hierarchical encoder-decoder architecture on generating multi-sentence summaries, and incorporate sentence-ranking into the summary generation process based on the graph-based attention mechanism. Different from these neural-based work, our model explicitly models the information selection process in document summarization by extending the encoder-decoder framework with an information selection layer. Our model captures both the global document information and local inter-sentence relations, and optimize the information selection process directly via distantly-supervised training, which effectively combines the strengths of extractive methods and abstractive methods.

6 Conclusion

In this paper, we have analyzed the necessity of explicitly modeling and optimizing of the information selection process in document summarization, and verified its effectiveness by extending the basic neural encoding-decoding framework with an information selection layer and optimizing it with distantly-supervised training. Our information selection layer consists of a gated global information filtering network and a local RNN sentence selection network. Experimental results demonstrate that both of them are effective for helping select salient information during the summary generation process, which significantly improves the document summarization performance. Our model combines the strengths of extractive methods and abstractive methods, which can generate more informative and concise summaries, and thus achieves state-of-the-art abstractive document summarization performance and is also competitive with state-of-the-art extractive models.

Acknowledgments

This work was supported by National Key Research and Development Program of China under grants 2016YFB1000902 and 2017YFC0820404, and National Natural Science Foundation of China under grants 61572469, 91646120, 61772501 and 61572473. We thank the anonymous reviewers for their helpful comments about this work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ann L Brown and Jeanne D Day. 1983. Macrorules for summarizing texts: The development of expertise. *Journal of verbal learning and verbal behavior*, 22(1):1–14.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015a. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI*, pages 2153–2159.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and WANG Houfeng. 2015b. Learning summary prior representation for extractive summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 829–833.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. *arXiv preprint arXiv:1610.08462*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *AAAI*, 1:1.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *EMNLP*, pages 1054–1059.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017a. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1171–1181.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017b. From neural sentence summarization to headline generation: A coarse-to-fine approach. *IJCAI*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Wenpeng Yin and Yulong Pei. 2015. Optimizing sentence modeling and selection for document summarization. In *IJCAI*.

Don't Give Me the Details, Just the Summary!
Topic-Aware Convolutional Neural Networks for Extreme Summarization

Shashi Narayan Shay B. Cohen Mirella Lapata
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB
`shashi.narayan@ed.ac.uk`, `{scohen,mlap}@inf.ed.ac.uk`

Abstract

We introduce *extreme summarization*, a new single-document summarization task which does not favor extractive strategies and calls for an abstractive modeling approach. The idea is to create a short, one-sentence news summary answering the question "What is the article about?". We collect a real-world, large scale dataset for this task by harvesting online articles from the British Broadcasting Corporation (BBC). We propose a novel abstractive model which is conditioned on the article's topics and based entirely on convolutional neural networks. We demonstrate experimentally that this architecture captures long-range dependencies in a document and recognizes pertinent content, outperforming an oracle extractive system and state-of-the-art abstractive approaches when evaluated automatically and by humans.[1]

SUMMARY: *A man and a child have been killed after a light aircraft made an emergency landing on a beach in Portugal.*

DOCUMENT: Authorities said the incident took place on Sao Joao beach in Caparica, south-west of Lisbon.
The National Maritime Authority said a middle-aged man and a young girl died after they were unable to avoid the plane.
[*6 sentences with 139 words are abbreviated from here.*]
Other reports said the victims had been sunbathing when the plane made its emergency landing.
[*Another 4 sentences with 67 words are abbreviated from here.*]
Video footage from the scene carried by local broadcasters showed a small recreational plane parked on the sand, apparently intact and surrounded by beachgoers and emergency workers.
[*Last 2 sentences with 19 words are abbreviated.*]

Figure 1: An abridged example from our extreme summarization dataset showing the document and its one-line summary. Document content present in the summary is color-coded.

1 Introduction

Automatic summarization is one of the central problems in Natural Language Processing (NLP) posing several challenges relating to *understanding* (i.e., identifying important content) and *generation* (i.e., aggregating and rewording the identified content into a summary). Of the many summarization paradigms that have been identified over the years (see Mani, 2001 and Nenkova and McKeown, 2011 for a comprehensive overview), single-document summarization has consistently attracted attention (Cheng and Lapata, 2016; Durrett et al., 2016; Nallapati et al., 2016, 2017; See et al., 2017; Tan and Wan, 2017; Narayan et al., 2017; Fan et al., 2017; Paulus et al., 2018; Pasunuru and Bansal, 2018; Celikyilmaz et al., 2018; Narayan et al., 2018a,b).

Neural approaches to NLP and their ability to learn continuous features without recourse to pre-processing tools or linguistic annotations have driven the development of large-scale document summarization datasets (Sandhaus, 2008; Hermann et al., 2015; Grusky et al., 2018). However, these datasets often favor extractive models which create a summary by identifying (and subsequently concatenating) the most important sentences in a document (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018b). Abstractive approaches, despite being more faithful to the actual summarization task, either lag behind extractive ones or are mostly extractive, exhibiting a small degree of abstraction (See et al., 2017; Tan and Wan, 2017; Paulus et al., 2018; Pasunuru and Bansal, 2018; Celikyilmaz et al., 2018).

In this paper we introduce *extreme summariza-*

---

[1] Our dataset, code, and demo are available at: `https://github.com/shashiongithub/XSum`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1797–1807
Brussels, Belgium, October 31 - November 4, 2018.

tion, a new single-document summarization task which is not amenable to extractive strategies and requires an abstractive modeling approach. The idea is to create a short, one-sentence news summary answering the question "What is the article about?". An example of a document and its extreme summary are shown in Figure 1. As can be seen, the summary is very different from a headline whose aim is to encourage readers to read the story; it draws on information interspersed in various parts of the document (not only the beginning) and displays multiple levels of abstraction including paraphrasing, fusion, synthesis, and inference. We build a dataset for the proposed task by harvesting online articles from the British Broadcasting Corporation (BBC) that often include a first-sentence summary.

We further propose a novel deep learning model which we argue is well-suited to the extreme summarization task. Unlike most existing abstractive approaches (Rush et al., 2015; Chen et al., 2016; Nallapati et al., 2016; See et al., 2017; Tan and Wan, 2017; Paulus et al., 2018; Pasunuru and Bansal, 2018; Celikyilmaz et al., 2018) which rely on an encoder-decoder architecture modeled by recurrent neural networks (RNNs), we present a *topic-conditioned* neural model which is based entirely on convolutional neural networks (Gehring et al., 2017b). Convolution layers capture long-range dependencies between words in the document more effectively compared to RNNs, allowing to perform document-level inference, abstraction, and paraphrasing. Our convolutional encoder associates each word with a topic vector capturing whether it is representative of the document's content, while our convolutional decoder conditions each word prediction on a document topic vector.

Experimental results show that when evaluated automatically (in terms of ROUGE) our topic-aware convolutional model outperforms an oracle extractive system and state-of-the-art RNN-based abstractive systems. We also conduct two human evaluations in order to assess (a) which type of summary participants prefer and (b) how much key information from the document is preserved in the summary. Both evaluations overwhelmingly show that human subjects find our summaries more informative and complete. Our contributions in this work are three-fold: a new single document summarization dataset that encourages the development of abstractive systems; corroborated by analysis and empirical results showing that extractive approaches are not well-suited to the extreme summarization task; and a novel topic-aware convolutional sequence-to-sequence model for abstractive summarization.

2 The XSum Dataset

Our extreme summarization dataset (which we call XSum) consists of BBC articles and accompanying single sentence summaries. Specifically, each article is prefaced with an introductory sentence (aka summary) which is professionally written, typically by the author of the article. The summary bears the HTML class "story-body__introduction," and can be easily identified and extracted from the main text body (see Figure 1 for an example summary-article pair).

We followed the methodology proposed in Hermann et al. (2015) to create a large-scale dataset for extreme summarization. Specifically, we collected 226,711 Wayback archived BBC articles ranging over almost a decade (2010 to 2017) and covering a wide variety of domains (e.g., News, Politics, Sports, Weather, Business, Technology, Science, Health, Family, Education, Entertainment and Arts). Each article comes with a unique identifier in its URL, which we used to randomly split the dataset into training (90%, 204,045), validation (5%, 11,332), and test (5%, 11,334) set. Table 1 compares XSum with the CNN, DailyMail, and NY Times benchmarks. As can be seen, XSum contains a substantial number of training instances, similar to DailyMail; documents and summaries in XSum are shorter in relation to other datasets but the vocabulary size is sufficiently large, comparable to CNN.

Table 2 provides empirical analysis supporting our claim that XSum is less biased toward extractive methods compared to other summarization datasets. We report the percentage of novel n-grams in the target gold summaries that do not appear in their source documents. There are 36% novel unigrams in the XSum reference summaries compared to 17% in CNN, 17% in DailyMail, and 23% in NY Times. This indicates that XSum summaries are more abstractive. The proportion of novel constructions grows for larger n-grams across datasets, however, it is much steeper in XSum whose summaries exhibit approximately 83% novel bigrams, 96% novel trigrams, and 98% novel 4-grams (comparison datasets display

Datasets	# docs (train/val/test)	avg. document length		avg. summary length		vocabulary size	
		words	sentences	words	sentences	document	summary
CNN	90,266/1,220/1,093	760.50	33.98	45.70	3.59	343,516	89,051
DailyMail	196,961/12,148/10,397	653.33	29.33	54.65	3.86	563,663	179,966
NY Times	589,284/32,736/32,739	800.04	35.55	45.54	2.44	1,399,358	294,011
XSum	204,045/11,332/11,334	431.07	19.77	23.26	1.00	399,147	81,092

Table 1: Comparison of summarization datasets with respect to overall corpus size, size of training, validation, and test set, average document (source) and summary (target) length (in terms of words and sentences), and vocabulary size on both on source and target. For CNN and DailyMail, we used the original splits of Hermann et al. (2015) and followed Narayan et al. (2018b) to preprocess them. For NY Times (Sandhaus, 2008), we used the splits and pre-processing steps of Paulus et al. (2018). For the vocabulary, we lowercase tokens.

Datasets	% of novel n-grams in gold summary				LEAD			EXT-ORACLE		
	unigrams	bigrams	trigrams	4-grams	R1	R2	RL	R1	R2	RL
CNN	16.75	54.33	72.42	80.37	29.15	11.13	25.95	50.38	28.55	46.58
DailyMail	17.03	53.78	72.14	80.28	40.68	18.36	37.25	55.12	30.55	51.24
NY Times	22.64	55.59	71.93	80.16	31.85	15.86	23.75	52.08	31.59	46.72
XSum	**35.76**	**83.45**	**95.50**	**98.49**	**16.30**	**1.61**	**11.95**	**29.79**	**8.81**	**22.65**

Table 2: Corpus bias towards extractive methods in the CNN, DailyMail, NY Times, and XSum datasets. We show the proportion of novel n-grams in gold summaries. We also report ROUGE scores for the LEAD baseline and the extractive oracle system EXT-ORACLE. Results are computed on the test set.

around 47–55% new bigrams, 58–72% new trigrams, and 63–80% novel 4-grams).

We further evaluated two extractive methods on these datasets. LEAD is often used as a strong lower bound for news summarization (Nenkova, 2005) and creates a summary by selecting the first few sentences or words in the document. We extracted the first 3 sentences for CNN documents and the first 4 sentences for DailyMail (Narayan et al., 2018b). Following previous work (Durrett et al., 2016; Paulus et al., 2018), we obtained LEAD summaries based on the first 100 words for NY Times documents. For XSum, we selected the first sentence in the document (excluding the one-line summary) to generate the LEAD. Our second method, EXT-ORACLE, can be viewed as an upper bound for extractive models (Nallapati et al., 2017; Narayan et al., 2018b). It creates an oracle summary by selecting the best possible set of sentences in the document that gives the highest ROUGE (Lin and Hovy, 2003) with respect to the gold summary. For XSum, we simply selected the single-best sentence in the document as summary.

Table 2 reports the performance of the two extractive methods using ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-L (RL) with the gold summaries as reference. The LEAD baseline performs extremely well on CNN, DailyMail and NY Times confirming that they are biased towards extractive methods. EXT-ORACLE further shows that improved sentence selection would bring further performance gains to extractive approaches. Abstractive systems trained on these datasets often have a hard time beating the LEAD, let alone EXT-ORACLE, or display a low degree of novelty in their summaries (See et al., 2017; Tan and Wan, 2017; Paulus et al., 2018; Pasunuru and Bansal, 2018; Celikyilmaz et al., 2018). Interestingly, LEAD and EXT-ORACLE perform poorly on XSum underlying the fact that it is less biased towards extractive methods.

In line with our findings, Grusky et al. (2018) have recently reported similar extractive biases in existing datasets. They constructed a new dataset called “Newsroom” which demonstrates a high diversity of summarization styles. XSum is not diverse, it focuses on a single news outlet (i.e., BBC) and a unifrom summarization style (i.e., a single sentence). However, it is sufficiently large for neural network training and we hope it will spur further research towards the development of abstractive summarization models.

3 Convolutional Sequence-to-Sequence Learning for Summarization

Unlike tasks like machine translation and paraphrase generation where there is often a one-to-one semantic correspondence between source and target words, document summarization must distill the content of the document into a few important facts. This is even more challenging for our task, where the compression ratio is extremely

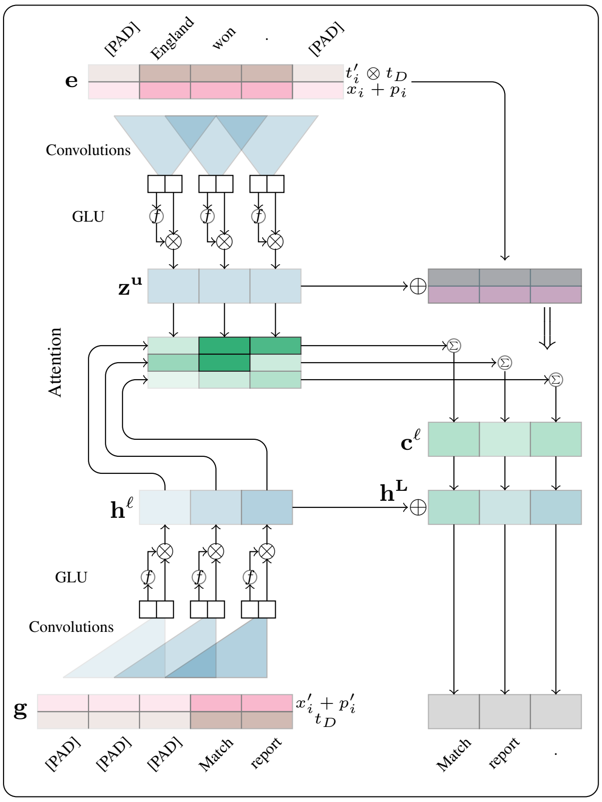

Figure 2: Topic-conditioned convolutional model for extreme summarization.

high, and pertinent content can be easily missed.

Recently, a convolutional alternative to sequence modeling has been proposed showing promise for machine translation (Gehring et al., 2017a,b) and story generation (Fan et al., 2018). We believe that convolutional architectures are attractive for our summarization task for at least two reasons. Firstly, contrary to recurrent networks which view the input as a chain structure, convolutional networks can be stacked to represent large context sizes. Secondly, hierarchical features can be extracted over larger and larger contents, allowing to represent long-range dependencies efficiently through shorter paths.

Our model builds on the work of Gehring et al. (2017b) who develop an encoder-decoder architecture for machine translation with an attention mechanism (Sukhbaatar et al., 2015) based exclusively on deep convolutional networks. We adapt this model to our summarization task by allowing it to recognize pertinent content (i.e., by foregrounding salient words in the document). In particular, we improve the convolutional encoder by associating each word with a vector representing topic salience, and the convolutional decoder by conditioning each word prediction on the document topic vector.

Model Overview At the core of our model is a simple convolutional block structure that computes intermediate states based on a fixed number of input elements. Our convolutional encoder (shown at the top of Figure 2) applies this unit across the document. We repeat these operations in a stacked fashion to get a multi-layer hierarchical representation over the input document where words at closer distances interact at lower layers while distant words interact at higher layers. The interaction between words through hierarchical layers effectively captures long-range dependencies.

Analogously, our convolutional decoder (shown at the bottom of Figure 2) uses the multi-layer convolutional structure to build a hierarchical representation over what has been predicted so far. Each layer on the decoder side determines useful source context by attending to the encoder representation before it passes its output to the next layer. This way the model remembers which words it previously attended to and applies multi-hop attention (shown at the middle of Figure 2) per time step. The output of the top layer is passed to a softmax classifier to predict a distribution over the target vocabulary.

Our model assumes access to word and document topic distributions. These can be obtained by any topic model, however we use Latent Dirichlet Allocation (LDA; Blei et al. 2003) in our experiments; we pass the distributions obtained from LDA directly to the network as additional input. This allows us to take advantage of topic modeling without interfering with the computational advantages of the convolutional architecture. The idea of capturing document-level semantic information has been previously explored for recurrent neural networks (Mikolov and Zweig, 2012; Ghosh et al., 2016; Dieng et al., 2017), however, we are not aware of any existing convolutional models.

Topic Sensitive Embeddings Let D denote a document consisting of a sequence of words $(w_1, \ldots, w_m)$; we embed D into a distributional space $\mathbf{x} = (x_1, \ldots, x_m)$ where $x_i \in \mathbb{R}^f$ is a column in embedding matrix $M \in \mathbb{R}^{V \times f}$ (where V is the vocabulary size). We also embed the absolute word positions in the document $\mathbf{p} = (p_1, \ldots, p_m)$ where $p_i \in \mathbb{R}^f$ is a column in position matrix

$P \in \mathbb{R}^{N \times f}$, and N is the maximum number of positions. Position embeddings have proved useful for convolutional sequence modeling (Gehring et al., 2017b), because, in contrast to RNNs, they do not observe the temporal positions of words (Shi et al., 2016). Let $t_D \in \mathbb{R}^{f'}$ be the topic distribution of document D and $\mathbf{t}' = (t'_1, \ldots, t'_m)$ the topic distributions of words in the document (where $t'_i \in \mathbb{R}^{f'}$). During encoding, we represent document D via $\mathbf{e} = (e_1, \ldots, e_m)$, where e_i is:

$$e_i = [(x_i + p_i); (t'_i \otimes t_D)] \in \mathbb{R}^{f+f'},$$

and $\otimes$ denotes point-wise multiplication. The topic distribution t'_i of word w_i essentially captures how topical the word is in itself (local context), whereas the topic distribution t_D represents the overall theme of the document (global context). The encoder essentially enriches the context of the word with its topical relevance to the document.

For every output prediction, the decoder estimates representation $\mathbf{g} = (g_1, \ldots, g_n)$ for previously predicted words $(w'_1, \ldots, w'_n)$ where g_i is:

$$g_i = [(x'_i + p'_i); t_D] \in \mathbb{R}^{f+f'},$$

x'_i and p'_i are word and position embeddings of previously predicted word w'_i, and t_D is the topic distribution of the input document. Note that the decoder does not use the topic distribution of w'_i as computing it on the fly would be expensive. However, every word prediction is conditioned on the topic of the document, enforcing the summary to have the same theme as the document.

Multi-layer Convolutional Structure Each convolution block, parametrized by $W \in \mathbb{R}^{2d \times kd}$ and $b_w \in \mathbb{R}^{2d}$, takes as input $X \in \mathbb{R}^{k \times d}$ which is the concatenation of k adjacent elements embedded in a d dimensional space, applies one dimensional convolution and returns an output element $Y \in \mathbb{R}^{2d}$. We apply Gated Linear Units (GLU, $v : \mathbb{R}^{2d} \rightarrow \mathbb{R}^{d}$, Dauphin et al. 2017) on the output of the convolution Y. Subsequent layers operate over the k output elements of the previous layer and are connected through residual connections (He et al., 2016) to allow for deeper hierarchical representation. We denote the output of the ℓth layer as $\mathbf{h}^\ell = (h^\ell_1, \ldots, h^\ell_n)$ for the decoder network, and $\mathbf{z}^\ell = (z^\ell_1, \ldots, z^\ell_m)$ for the encoder network.

Multi-hop Attention Our encoder and decoder are tied to each other through a multi-hop attention mechanism. For each decoder layer ℓ, we compute the attention a^ℓ_{ij} of state i and source element j as:

$$a^\ell_{ij} = \frac{\exp(d^\ell_i \cdot z^u_j)}{\sum_{t=1}^{m} \exp(d^\ell_i \cdot z^u_t)},$$

where $d^\ell_i = W^\ell_d h^\ell_i + b^\ell_i + g_i$ is the decoder state summary combining the current decoder state h^ℓ_i and the previous output element embedding g_i. The vector $\mathbf{z^u}$ is the output from the last encoder layer u. The conditional input c^ℓ_i to the current decoder layer is a weighted sum of the encoder outputs as well as the input element embeddings e_j:

$$c^\ell_i = \sum_{j=1}^{m} a^\ell_{ij}(z^u_j + e_j).$$

The attention mechanism described here performs multiple attention “hops” per time step and considers which words have been previously attended to. It is therefore different from single-step attention in recurrent neural networks (Bahdanau et al., 2015), where the attention and weighted sum are computed over $\mathbf{z^u}$ only.

Our network uses multiple linear layers to project between the embedding size $(f + f')$ and the convolution output size $2d$. They are applied to $\mathbf{e}$ before feeding it to the encoder, to the final encoder output $\mathbf{z^u}$, to all decoder layers $\mathbf{h}^\ell$ for the attention score computation, and to the final decoder output $\mathbf{h^L}$ before the softmax. We pad the input with $k-1$ zero vectors on both left and right side to ensure that the output of the convolutional layers matches the input length. During decoding, we ensure that the decoder does not have access to future information; we start with k zero vectors and shift the covolutional block to the right after every prediction. The final decoder output $\mathbf{h^L}$ is used to compute the distribution over the target vocabulary T as:

$$p(y_{i+1}|y_1, \ldots, y_i, D, t_D, \mathbf{t}') = \\ \text{softmax}(W_o h^L_i + b_o) \in \mathbb{R}^T.$$

We use layer normalization and weight initialization to stabilize learning.

Our topic-enhanced model calibrates long-range dependencies with globally salient content. As a result, it provides a better alternative to vanilla convolutional sequence models

(Gehring et al., 2017b) and RNN-based summarization models (See et al., 2017) for capturing cross-document inferences and paraphrasing. At the same time it retains the computational advantages of convolutional models. Each convolution block operates over a fixed-size window of the input sequence, allowing for simultaneous encoding of the input, ease in learning due to the fixed number of non-linearities and transformations for words in the input sequence.

4 Experimental Setup

In this section we present our experimental setup for assessing the performance of our **T**opic-aware **Conv**olutional **S**equence **to** **S**equence model which we call T-CONVS2S for short. We discuss implementation details and present the systems used for comparison with our approach.

Comparison Systems We report results with various systems which were all trained on the XSum dataset to generate a one-line summary given an input news article. We compared T-CONVS2S against three extractive systems: a baseline which randomly selects a sentence from the input document (RANDOM), a baseline which simply selects the leading sentence from the document (LEAD), and an oracle which selects a single-best sentence in each document (EXT-ORACLE). The latter is often used as an upper bound for extractive methods. We also compared our model against the RNN-based abstractive systems introduced by See et al. (2017). In particular, we experimented with an attention-based sequence to sequence model (SEQ2SEQ), a pointer-generator model which allows to copy words from the source text (PTGEN), and a pointer-generator model with a coverage mechanism to keep track of words that have been summarized (PTGEN-COVG). Finally, we compared our model against the vanilla convolution sequence to sequence model (CONVS2S) of Gehring et al. (2017b).

Model Parameters and Optimization We did not anonymize entities but worked on a lower-cased version of the XSum dataset. During training and at test time the input document was truncated to 400 tokens and the length of the summary limited to 90 tokens.

The LDA model (Blei et al., 2003) was trained on XSum documents (training portion). We therefore obtained for each word a probability distribution over topics which we used to estimate $\mathbf{t}'$; the topic distribution t_D can be inferred for any new document, at training and test time. We explored several LDA configurations on held-out data, and obtained best results with 512 topics. Table 3 shows some of the topics learned by the LDA model.

T1:	charge, court, murder, police, arrest, guilty, sentence, boy, bail, space, crown, trial
T2:	church, abuse, bishop, child, catholic, gay, pope, school, christian, priest, cardinal
T3:	council, people, government, local, housing, home, house, property, city, plan, authority
T4:	clinton, party, trump, climate, poll, vote, plaid, election, debate, change, candidate, campaign
T5:	country, growth, report, business, export, fall, bank, security, economy, rise, global, inflation
T6:	hospital, patient, trust, nhs, people, care, health, service, staff, report, review, system, child

Table 3: Example topics learned by the LDA model on XSum documents (training portion).

For SEQ2SEQ, PTGEN and PTGEN-COVG, we used the best settings reported on the CNN and DailyMail data (See et al., 2017).[2] All three models had 256 dimensional hidden states and 128 dimensional word embeddings. They were trained using Adagrad (Duchi et al., 2011) with learning rate 0.15 and an initial accumulator value of 0.1. We used gradient clipping with a maximum gradient norm of 2, but did not use any form of regularization. We used the loss on the validation set to implement early stopping.

For CONVS2S[3] and T-CONVS2S, we used 512 dimensional hidden states and 512 dimensional word and position embeddings. We trained our convolutional models with Nesterov's accelerated gradient method (Sutskever et al., 2013) using a momentum value of 0.99 and renormalized gradients if their norm exceeded 0.1 (Pascanu et al., 2013). We used a learning rate of 0.10 and once the validation perplexity stopped improving, we reduced the learning rate by an order of magnitude after each epoch until it fell below 10^{-4}. We also applied a dropout of 0.2 to the embeddings, the decoder outputs and the input of the convolutional blocks. Gradients were normalized by the number of non-padding tokens per mini-batch. We also

---

[2] We used the code available at `https://github.com/abisee/pointer-generator`.

[3] We used the code available at `https://github.com/facebookresearch/fairseq-py`.

used weight normalization for all layers except for lookup tables.

All neural models, including ours and those based on RNNs (See et al., 2017) had a vocabulary of 50,000 words and were trained on a single Nvidia M40 GPU with a batch size of 32 sentences. Summaries at test time were obtained using beam search (with beam size 10).

5 Results

Automatic Evaluation We report results using automatic metrics in Table 4. We evaluated summarization quality using F$_1$ ROUGE (Lin and Hovy, 2003). Unigram and bigram overlap (ROUGE-1 and ROUGE-2) are a proxy for assessing informativeness and the longest common subsequence (ROUGE-L) represents fluency.[4]

On the XSum dataset, Seq2Seq outperforms the LEAD and RANDOM baselines by a large margin. PtGen, a Seq2Seq model with a "copying" mechanism outperforms EXT-ORACLE, a "perfect" extractive system on ROUGE-2 and ROUGE-L. This is in sharp contrast to the performance of these models on CNN/DailyMail (See et al., 2017) and Newsroom datasets (Grusky et al., 2018), where they fail to outperform the LEAD. The result provides further evidence that XSum is a good testbed for abstractive summarization. PtGen-Covg, the best performing abstractive system on the CNN/DailyMail datasets, does not do well. We believe that the coverage mechanism is more useful when generating multi-line summaries and is basically redundant for extreme summarization.

ConvS2S, the convolutional variant of Seq2Seq, significantly outperforms all RNN-based abstractive systems. We hypothesize that its superior performance stems from the ability to better represent document content (i.e., by capturing long-range dependencies). Table 4 shows several variants of T-ConvS2S including an encoder network enriched with information about how topical a word is on its own (enc$_{t'}$) or in the document (enc$_{(t',t_D)}$). We also experimented with various decoders by conditioning every prediction on the topic of the document, basically encouraging the summary to be in the same theme as the document (dec$_{t_D}$) or letting the decoder decide the theme of the summary.

Models	R1	R2	RL
Random	15.16	1.78	11.27
LEAD	16.30	1.60	11.95
EXT-ORACLE	29.79	8.81	22.66
Seq2Seq	28.42	8.77	22.48
PtGen	29.70	9.21	23.24
PtGen-Covg	28.10	8.02	21.72
ConvS2S	31.27	11.07	25.23
T-ConvS2S (enc$_{t'}$)	31.71	11.38	25.56
T-ConvS2S (enc$_{t'}$, dec$_{t_D}$)	31.71	11.34	25.61
T-ConvS2S (enc$_{(t',t_D)}$)	31.61	11.30	25.51
T-ConvS2S (enc$_{(t',t_D)}$, dec$_{t_D}$)	**31.89**	**11.54**	**25.75**

Table 4: ROUGE results on XSum test set. We report ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-L (RL) F$_1$ scores. Extractive systems are in the upper block, RNN-based abstractive systems are in the middle block, and convolutional abstractive systems are in the bottom block.

Models	% of novel n-grams in generated summaries			
	unigrams	bigrams	trigrams	4-grams
LEAD	0.00	0.00	0.00	0.00
EXT-ORACLE	0.00	0.00	0.00	0.00
PtGen	27.40	73.33	90.43	96.04
ConvS2S	**31.26**	**79.50**	**94.28**	**98.10**
T-ConvS2S	30.73	79.18	94.10	98.03
GOLD	35.76	83.45	95.50	98.49

Table 5: Proportion of novel n-grams in summaries generated by various models on the XSum test set.

Interestingly, all four T-ConvS2S variants outperform ConvS2S. T-ConvS2S performs best when both encoder and decoder are constrained by the document topic (enc$_{(t',t_D)}$,dec$_{t_D}$). In the remainder of the paper, we refer to this variant as T-ConvS2S.

We further assessed the extent to which various models are able to perform rewriting by generating genuinely abstractive summaries. Table 5 shows the proportion of novel n-grams for LEAD, EXT-ORACLE, PtGen, ConvS2S, and T-ConvS2S. As can be seen, the convolutional models exhibit the highest proportion of novel n-grams. We should also point out that the summaries being evaluated have on average comparable lengths; the summaries generated by PtGen contain 22.57 words, those generated by ConvS2S and T-ConvS2S have 20.07 and 20.22 words, respectively, while GOLD summaries are the longest with 23.26 words. Interestingly, PtGen trained on XSum only copies 4% of 4-grams in the source document, 10% of trigrams, 27% of bigrams, and 73% of unigrams. This is in sharp contrast to Pt-

---

[4] We used `pyrouge` to compute all ROUGE scores, with parameters "-a -c 95 -m -n 4 -w 1.2."

EXT-ORACLE	Caroline Pidgeon is the Lib Dem candidate, Sian Berry will contest the election for the Greens and UKIP has chosen its culture spokesman Peter Whittle.	[34.1, 20.5, 34.1]
PTGEN	UKIP leader Nigel Goldsmith has been elected as the new mayor of London to elect a new conservative MP.	[45.7, 6.1, 28.6]
CONVS2S	London mayoral candidate Zac Goldsmith has been elected as the new mayor of London.	[53.3, 21.4, 26.7]
T-CONVS2S	Former London mayoral candidate Zac Goldsmith has been chosen to stand in the London mayoral election.	[50.0, 26.7, 37.5]
GOLD	Zac Goldsmith will contest the 2016 London mayoral election for the conservatives, it has been announced.	
Questions	(1) Who will contest for the conservatives? (Zac Goldsmith) (2) For what election will he/she contest? (The London mayoral election)	
EXT-ORACLE	North-east rivals Newcastle are the only team below them in the Premier League table.	[35.3, 18.8, 35.3]
PTGEN	Sunderland have appointed former Sunderland boss Dick Advocaat as manager at the end of the season to sign a new deal.	[45.0, 10.5, 30.0]
CONVS2S	Sunderland have sacked manager Dick Advocaat after less than three months in charge.	[25.0, 6.7, 18.8]
T-CONVS2S	Dick Advocaat has resigned as Sunderland manager until the end of the season.	[56.3, 33.3, 56.3]
GOLD	Dick Advocaat has resigned as Sunderland boss, with the team yet to win in the Premier League this season.	
Questions	(1) Who has resigned? (Dick Advocaat) (2) From what post has he/she resigned? (Sunderland boss)	
EXT-ORACLE	The Greater Ardoyne residents collective (GARC) is protesting against an agreement aimed at resolving a long-running dispute in the area.	[26.7, 9.3, 22.2]
PTGEN	A residents' group has been granted permission for GARC to hold a parade on the outskirts of Crumlin, County Antrim.	[28.6, 5.0, 28.6]
CONVS2S	A protest has been held in the Republic of Ireland calling for an end to parading parading in North Belfast.	[42.9, 20.0, 33.3]
T-CONVS2S	A protest has been held in North Belfast over a protest against the Orange Order in North Belfast.	[45.0, 26.3, 45.0]
GOLD	Church leaders have appealed to a nationalist residents' group to call off a protest against an Orange Order parade in North Belfast.	
Questions	(1) Where is the protest supposed to happen? (North Belfast) (2) What are they protesting against? (An Orange Order parade)	

Table 6: Example output summaries on the XSum test set with [ROUGE-1, ROUGE-2 and ROUGE-L] scores, goldstandard reference, and corresponding questions. Words highlighted in blue are either the right answer or constitute appropriate context for inferring it; words in red lead to the wrong answer.

GEN trained on CNN/DailyMail exhibiting mostly extractive patterns; it copies more than 85% of 4-grams in the source document, 90% of trigrams, 95% of bigrams, and 99% of unigrams (See et al., 2017). This result further strengthens our hypothesis that XSum is a good testbed for abstractive methods.

Human Evaluation In addition to automatic evaluation using ROUGE which can be misleading when used as the only means to assess the informativeness of summaries (Schluter, 2017), we also evaluated system output by eliciting human judgments in two ways.

In our first experiment, participants were asked to compare summaries produced from the EXT-ORACLE baseline, PTGEN, the best performing system of See et al. (2017), CONVS2S, our topic-aware model T-CONVS2S, and the human-authored gold summary (GOLD). We did not include extracts from the LEAD as they were significantly inferior to other models.

The study was conducted on the Amazon Mechanical Turk platform using *Best-Worst Scaling* (BWS; Louviere and Woodworth 1991; Louviere et al. 2015), a less labor-intensive alternative to paired comparisons that has been shown to produce more reliable results than rating scales (Kiritchenko and Mohammad, 2017). Participants were presented with a document and summaries generated from two out of five systems and were asked to decide which summary was *better* and which one was *worse* in order of informativeness (does the summary capture important information in the document?) and fluency (is the summary written in well-formed English?). Examples of system summaries are shown in Table 6. We randomly selected 50 documents from the XSum test set and compared all possible combinations of two out of five systems for each document. We collected judgments from three different participants for each comparison. The order of summaries was

Models	Score	QA
EXT-ORACLE	-0.121	15.70
PTGEN	-0.218	21.40
CONVS2S	-0.130	30.90
T-CONVS2S	**0.037**	**46.05**
GOLD	0.431	97.23

Table 7: System ranking according to human judgments and QA-based evaluation.

randomized per document and the order of documents per participant.

The score of a system was computed as the percentage of times it was chosen as best minus the percentage of times it was selected as worst. The scores range from -1 (worst) to 1 (best) and are shown in Table 7. Perhaps unsurprisingly human-authored summaries were considered best, whereas, T-CONVS2S was ranked 2nd followed by EXT-ORACLE and CONVS2S. PTGEN was ranked worst with the lowest score of -0.218. We carried out pairwise comparisons between all models to assess whether system differences are statistically significant. GOLD is significantly different from all other systems and T-CONVS2S is significantly different from CONVS2S and PTGEN (using a one-way ANOVA with posthoc Tukey HSD tests; $p < 0.01$). All other differences are not statistically significant.

For our second experiment we used a question-answering (QA) paradigm (Clarke and Lapata, 2010; Narayan et al., 2018b) to assess the degree to which the models retain key information from the document. We used the same 50 documents as in our first elicitation study. We wrote two fact-based questions per document, just by reading the summary, under the assumption that it highlights the most important content of the news article. Questions were formulated so as not to reveal answers to subsequent questions. We created 100 questions in total (see Table 6 for examples). Participants read the output summaries and answered the questions as best they could without access to the document or the gold summary. The more questions can be answered, the better the corresponding system is at summarizing the document as a whole. Five participants answered questions for each summary.

We followed the scoring mechanism introduced in Clarke and Lapata (2010). A correct answer was marked with a score of one, partially correct answers with a score of 0.5, and zero otherwise. The final score for a system is the average of all its question scores. Answers again were elicited using Amazon's Mechanical Turk crowdsourcing platform. We uploaded the data in batches (one system at a time) to ensure that the same participant does not evaluate summaries from different systems on the same set of questions.

Table 7 shows the results of the QA evaluation. Based on summaries generated by T-CONVS2S, participants can answer 46.05% of the questions correctly. Summaries generated by CONVS2S, PTGEN and EXT-ORACLE provide answers to 30.90%, 21.40%, and 15.70% of the questions, respectively. Pairwise differences between systems are all statistically significant ($p < 0.01$) with the exception of PTGEN and EXT-ORACLE. EXT-ORACLE performs poorly on both QA and rating evaluations. The examples in Table 6 indicate that EXT-ORACLE is often misled by selecting a sentence with the highest ROUGE (against the gold summary), but ROUGE itself does not ensure that the summary retains the most important information from the document. The QA evaluation further emphasizes that in order for the summary to be felicitous, information needs to be embedded in the appropriate context. For example, CONVS2S and PTGEN will fail to answer the question "Who has resigned?" (see Table 6 second block) despite containing the correct answer "Dick Advocaat" due to the wrong context. T-CONVS2S is able to extract important entities from the document with the right theme.

6 Conclusions

In this paper we introduced the task of "extreme summarization" together with a large-scale dataset which pushes the boundaries of abstractive methods. Experimental evaluation revealed that models which have abstractive capabilities do better on this task and that high-level document knowledge in terms of topics and long-range dependencies is critical for recognizing pertinent content and generating informative summaries. In the future, we would like to create more linguistically-aware encoders and decoders incorporating co-reference and entity linking.

Acknowledgments We gratefully acknowledge the support of the European Research Council (Lapata; award number 681760), the European Union under the Horizon 2020 SUMMA project (Narayan, Cohen; grant agreement 688139), and Huawei Technologies (Cohen).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, USA.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2754–2760, New York, USA.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 484–494, Berlin, Germany.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 933–941, Sydney, Australia.

Adji B. Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2017. Topicrnn: A recurrent neural network with long-range semantic dependency. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1998–2008, Berlin, Germany.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *CoRR*, abs/1711.05217.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017a. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 123–135, Vancouver, Canada.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017b. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1243–1252, Sydney, Australia.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual LSTM (CLSTM) models for large scale NLP tasks. *CoRR*, abs/1602.06291.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. NEWSROOM: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, USA.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, Las Vegas, USA.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Morgan, Kaufmann.

Svetlana Kiritchenko and Saif Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 465–470, Vancouver, Canada.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram cooccurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 71–78, Edmonton, Canada.

Jordan J Louviere, Terry N Flynn, and Anthony Alfred John Marley. 2015. *Best-worst scaling: Theory, methods and applications*. Cambridge University Press.

Jordan J Louviere and George G Woodworth. 1991. Best-worst scaling: A model for the largest difference judgments. *University of Alberta: Working Paper*.

Inderjeet Mani. 2001. *Automatic Summarization*. Natural language processing. John Benjamins Publishing Company.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Proceedings of the Spoken Language Technology Workshop*, pages 234–239. IEEE.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3075–3081, San Francisco, California USA.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany.

Shashi Narayan, Ronald Cardenas, Nikos Papasarantopoulos, Shay B. Cohen, Mirella Lapata, Jiangsheng Yu, and Yi Chang. 2018a. Document modeling with external attention for sentence extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, USA.

Shashi Narayan, Nikos Papasarantopoulos, Shay B. Cohen, and Mirella Lapata. 2017. Neural extractive summarization with side information. *CoRR*, abs/1704.04530.

Ani Nenkova. 2005. Automatic text summarization of newswire: Lessons learned from the Document Understanding Conference. In *Proceedings of the 29th National Conference on Artificial Intelligence*, pages 1436–1441, Pittsburgh, Pennsylvania, USA.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2–3):103–233.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, pages 1310–1318, Atlanta, GA, USA.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, USA.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, BC, Canada.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal.

Evan Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12).

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Short Papers*, pages 41–45, Valencia, Spain.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083, Vancouver, Canada.

Xing Shi, Kevin Knight, and Deniz Yuret. 2016. Why neural translations are the right length. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2278–2282, Austin, Texas.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Morgan, Kaufmann.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, pages 1139–1147, Atlanta, GA, USA.

Jiwei Tan and Xiaojun Wan. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1171–1181, Vancouver, Canada.

Improving Abstraction in Text Summarization

Wojciech Kryściński *
KTH Royal Institute of Technology
wkr@kth.se

Romain Paulus
Salesforce Research
rpaulus@salesforce.com

Caiming Xiong
Salesforce Research
cxiong@salesforce.com

Richard Socher
Salesforce Research
rsocher@salesforce.com

Abstract

Abstractive text summarization aims to shorten long text documents into a human readable form that contains the most important facts from the original document. However, the level of actual abstraction as measured by novel phrases that do not appear in the source document remains low in existing approaches. We propose two techniques to improve the level of abstraction of generated summaries. First, we decompose the decoder into a contextual network that retrieves relevant parts of the source document, and a pretrained language model that incorporates prior knowledge about language generation. Second, we propose a novelty metric that is optimized directly through policy learning to encourage the generation of novel phrases. Our model achieves results comparable to state-of-the-art models, as determined by ROUGE scores and human evaluations, while achieving a significantly higher level of abstraction as measured by n-gram overlap with the source document.

1 Introduction

Text summarization concerns the task of compressing a long sequence of text into a more concise form. The two most common approaches to summarization are *extractive* (Dorr et al., 2003; Nallapati et al., 2017), where the model extracts salient parts of the source document, and *abstractive* (Paulus et al., 2017; See et al., 2017), where the model not only extracts but also concisely paraphrases the important parts of the document via generation. We focus on developing a summarization model that produces an increased level of abstraction. That is, the model produces concise summaries without only copying long passages from the source document.

A high quality summary is shorter than the original document, conveys only the most important and no extraneous information, and is semantically and syntactically correct. Because it is difficult to gauge the correctness of the summary, evaluation metrics for summarization models use word overlap with the ground-truth summary in the form of ROUGE (Lin, 2004) scores. However, word overlap metrics do not capture the *abstractive* nature of high quality human-written summaries: the use of paraphrases with words that do not necessarily appear in the source document.

The state-of-the-art abstractive text summarization models have high word overlap performance, however they tend to copy long passages of the source document directly into the summary, thereby producing summaries that are not abstractive (See et al., 2017).

We propose two general extensions to summarization models that improve the level of abstraction of the summary while preserving word overlap with the ground-truth summary. Our first contribution decouples the extraction and generation responsibilities of the decoder by factoring it into a contextual network and a language model. The contextual network has the sole responsibility of extracting and compacting the source document whereas the language model is responsible for the generation of concise paraphrases. Our second contribution is a mixed objective that jointly optimizes the n-gram overlap with the ground-truth summary while encouraging abstraction. This is done by combining maximum likelihood estimation with policy gradient. We reward the policy with the ROUGE metric, which measures word overlap with the ground-truth summary, as well as a novel abstraction reward that encourages the generation of words not in the source document.

We demonstrate the effectiveness of our contributions on a encoder-decoder summarization

---

[*] Work performed while at Salesforce Research.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1808–1817
Brussels, Belgium, October 31 - November 4, 2018.

Article
(cnn) to allay possible concerns, boston prosecutors released video friday of the shooting of a police officer last month that resulted in the killing of the gunman. the officer wounded, john moynihan, is white. angelo west, the gunman shot to death by officers, was black. after the shooting, community leaders in the predominantly african-american neighborhood of (...)
Human-written summary
boston police officer john moynihan is released from the hospital. video shows that the man later shot dead by police in boston opened fire first. moynihan was shot in the face during a traffic stop.
Generated summary (See et al., 2017)
boston prosecutors released video friday of the shooting of a police officer last month. the gunman shot to death by officers , was black . one said the officers were forced to return fire. he was placed in a medically induced coma at a boston hospital.
Generated summary (Liu et al., 2018)
boston prosecutors released video of the shooting of a police officer last month . the shooting occurred in the wake of the boston marathon bombing. the video shows west sprang out and fired a shot with a pistol at officer's face.
Our summary (ML+RL ROUGE+Novel, with LM)
new: boston police release video of shooting of officer , john moynihan. new: angelo west had several prior gun convictions , police say. boston police officer john moynihan, 34, survived with a bullet wound . he was in a medically induced coma at a boston hospital , a police officer says.

Table 1: Summaries generated by different models for the same CNN/Daily Mail article. The highlighted spans indicate phrases of 3 tokens or more that are copied word-by-word from the original article.

model. Our model obtains state-of-the-art ROUGE-L scores, and ROUGE-1 and ROUGE-2 performance comparable to state-of-the-art methods on the CNN/DailyMail dataset. Moreover, we significantly outperform all previous abstractive approaches in our abstraction metrics. Table 1 shows a comparison of summaries generated by our model and previous abstractive models, showing less copying and more abstraction in our model.

2 Model

2.1 Base Model and Training Objective

The base model follows the encoder-decoder architecture with temporal attention and intra-attention proposed by Paulus et al. (2017). Let $E \in \mathbb{R}^{n \times d^{\text{emb}}}$ denote the matrix of d^{emb} dimensional word embeddings of the n words in the source document. The encoding of the source document h^{enc} is computed via a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) whose output has dimension d^{hid}.

$$h^{\text{enc}} = \text{BiLSTM}\left(E\right) \in \mathbb{R}^{n \times d^{\text{hid}}} \tag{1}$$

The decoder uses temporal attention over the encoded sequence that penalizes input tokens that previously had high attention scores. Let h_t^{dec} denote the decoder state at time t. The temporal attention context at time t, c_t^{tmp}, is computed as

$$s_{ti}^{\text{tmp}} = \left(h_t^{\text{dec}}\right)^{\intercal} W^{\text{tmp}} h_i^{\text{enc}} \in \mathbb{R} \tag{2}$$

$$q_{ti}^{\text{tmp}} = \frac{\exp(s_{ti}^{\text{tmp}})}{\sum_{j=1}^{t-1} \exp(s_{ji}^{\text{tmp}})} \in \mathbb{R} \tag{3}$$

$$\alpha_{ti}^{\text{tmp}} = \frac{q_{ti}^{\text{tmp}}}{\sum_{j=1}^{n} q_{tj}^{\text{tmp}}} \in \mathbb{R} \tag{4}$$

$$c_t^{\text{tmp}} = \sum_{i=1}^{n} \alpha_{ti}^{\text{tmp}} h_i^{\text{enc}} \in \mathbb{R}^{d^{\text{hid}}} \tag{5}$$

where we set q_{ti}^{tmp} to $\exp(s_{ti}^{\text{tmp}})$ for $t = 1$.

The decoder also attends to its previous states via intra-attention over the decoded sequence. The intra-attention context at time t, c_t^{int}, is computed as

$$s_{ti}^{\text{int}} = \left(h_t^{dec}\right)^{\intercal} W^{\text{int}} h_i^{\text{dec}} \in \mathbb{R} \tag{6}$$

$$c_t^{\text{int}} = \sum_{i=1}^{t-1} \left(\frac{s_{ti}^{\text{int}}}{\sum_{j=1}^{n} s_{tj}^{\text{int}}} \right) h_i^{\text{dec}} \in \mathbb{R}^{d^{\text{hid}}} \tag{7}$$

The decoder generates tokens by interpolating between selecting words from the source document via a pointer network as well as selecting words from a fixed output vocabulary. Let z_t denote the ground-truth label as to whether the tth

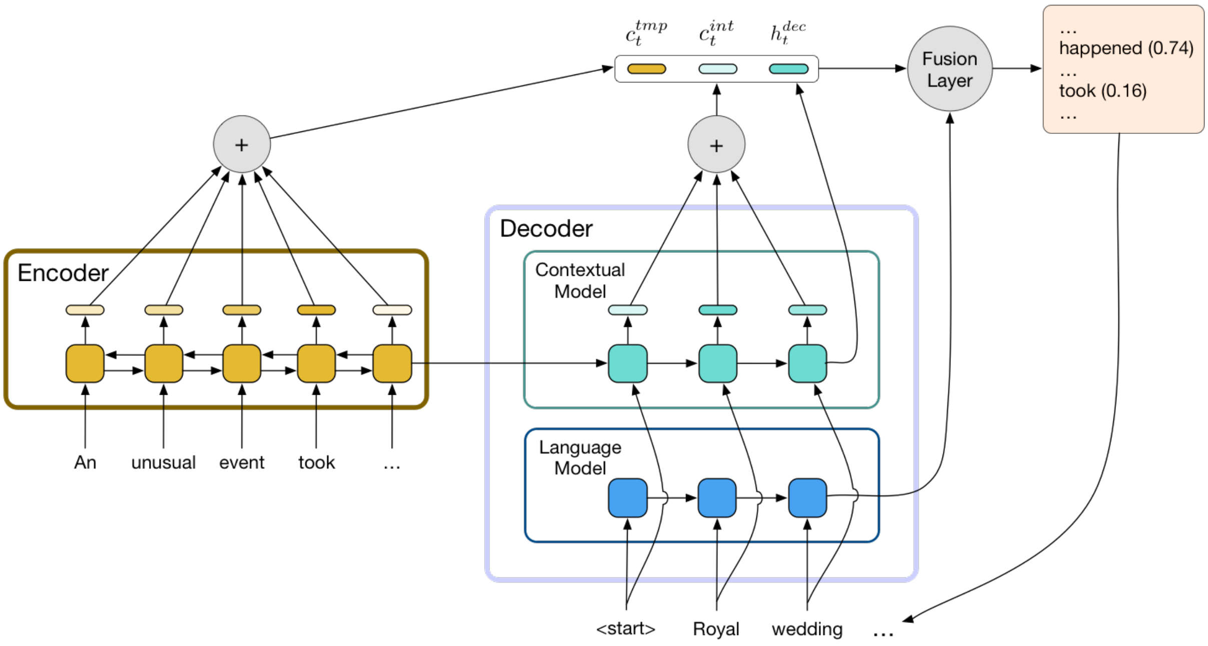

Figure 1: The network architecture with the decoder factorized into separate contextual and language models. The reference vector, composed of context vectors c_t^{tmp}, c_t^{int}, and the hidden state of the contextual model h_t^{dec}, is fused with the hidden state of the language model and then used to compute the distribution over the output vocabulary.

output word should be generated by the selecting from the output vocabulary as opposed to from the source document. We compute $p(z_t)$, the probability that the decoder generates from the output vocabulary, as

$$\begin{aligned} r_t &= [h_t^{\text{dec}}; c_t^{\text{tmp}}; c_t^{\text{int}}] \in \mathbb{R}^{3d^{\text{hid}}} && (8) \\ p(z_t) &= \text{sigmoid}(W^z r_t + b^z) \in \mathbb{R} && (9) \end{aligned}$$

The probability of selecting the word y_t from a fixed vocabulary at time step t is defined as

$$p^{\text{gen}}(y_t) = \text{softmax}\left(W^{\text{gen}} r_t + b^{\text{gen}}\right) \tag{10}$$

We set $p^{\text{cp}}(y_t)$, the probability of copying the word y_t from the source document, to the temporal attention distribution α_t^{tmp}. The joint probability of using the generator and generating the word y_t at time step t, $p(z_t, y_t)$, is then

$$p(z_t, y_t) = p(y_t \mid z_t) p(z_t) \tag{11}$$

the likelihood of which is

$$\begin{aligned} &\log p(z_t, y_t) = \log p(y_t \mid z_t) + \log p(z_t) \\ &= z_t \log p^{\text{gen}}(y_t) + (1 - z_t) \log p^{\text{cp}}(y_t) \\ &\quad + z_t \log p(z_t) + (1 - z_t) \log\left(1 - p\left(z_t\right)\right) \\ &= z_t \left(\log p^{\text{gen}}(y_t) + \log p(z_t)\right) \\ &\quad + (1 - z_t)\left(\log p^{\text{cp}}(y_t) + \log\left(1 - p\left(z_t\right)\right)\right) \end{aligned} \tag{12}$$

The objective function combines maximum likelihood estimation with policy learning. Let m denote the length of the ground-truth summary, The maximum likelihood loss L^{ml} is computed as

$$L^{\text{ml}} = -\sum_{t=1}^{m} \log p(z_t, y_t) \tag{13}$$

Policy learning uses ROUGE-L as its reward function and a self-critical baseline using the greedy decoding policy (Rennie et al., 2016). Let y^{sam} denote the summary obtained by sampling from the current policy p, y^{gre} and z^{gre} the summary and generator choice obtained by greedily choosing from $p(z_t, y_t)$, $R(y)$ the ROUGE-L score of the summary y, and Θ the model parameters. The policy learning loss is

$$\begin{aligned} \hat{R} &= R\left(y^{\mathrm{sam}}\right) - R\left(y^{\mathrm{gre}}\right) & (14) \\ L^{\mathrm{pg}} &= -\mathbb{E}_{\substack{z^{\mathrm{sam}} \sim p(z), \\ y^{\mathrm{sam}} \sim p(y|z)}} [\hat{R}] & (15) \end{aligned}$$

where we use greedy predictions by the model according to eq. (13) as a baseline for variance reduction. The policy gradient, as per Schulman et al. (2015), is

$$\nabla_{\Theta} L^{\mathrm{pg}} \approx -\hat{R} \sum_{t=1}^{m} \nabla_{\Theta} \log p\left(z_t^{\mathrm{sam}}, y_t^{\mathrm{sam}}\right) \quad (16)$$

The final loss is a mixture between the maximum likelihood loss and the policy learning loss, weighted by a hyperparameter γ.

$$L = (1-\gamma) L^{\mathrm{ml}} + \gamma L^{\mathrm{pg}} \quad (17)$$

2.2 Language Model Fusion

The decoder is an essential component of the base model. Given the source document and the previously generated summary tokens, the decoder both extracts relevant parts of the source document through the pointer network as well as composes paraphrases from the fixed vocabulary. We decouple these two responsibilities by augmenting the decoder with an external language model. The language model assumes responsibility of generating from the fixed vocabulary, and allows the decoder to focus on attention and extraction. This decomposition has the added benefit of easily incorporating external knowledge about fluency or domain specific styles via pre-training the language model on a large scale text corpora.

The architecture of our language model is based on Merity et al. (2018). We use a 3-layer unidirectional LSTM with weight-dropped LSTM units.

Let e_t denote the embedding of the word generated during time step t. The hidden state of the language model at the l-th layer is

$$h_{l,t}^{\mathrm{lm}} = \mathrm{LSTM}_3^{\mathrm{lm}}\left(e_{t-1}, h_{l,t-1}^{\mathrm{lm}}\right) \quad (18)$$

At each time step t, we combine the hidden state of the last language model LSTM layer, $h_{3,t}^{\mathrm{lm}}$, with r_t defined in eq. (8) in a fashion similar to Sriram et al. (2017). Let $\odot$ denote element-wise multiplication. We use a gating function whose output g_t filters the content of the language model hidden state.

$$\begin{aligned} f_t &= \mathrm{sigmoid}\left(W^{\mathrm{lm}}[r_t; h_{3,t}^{\mathrm{lm}}] + b^{\mathrm{lm}}\right) & (19) \\ g_t &= W^{\mathrm{fuse}}([r_t; g_t \odot h_{3,t}^{\mathrm{lm}}]) + b^{\mathrm{fuse}} & (20) \\ h_t^{\mathrm{fuse}} &= \mathrm{ReLU}\left(g_t\right) & (21) \end{aligned}$$

We then replace the output distribution of the language model $p^{\mathrm{gen}}\left(y_t\right)$ in eq. 10 with

$$p^{\mathrm{gen}}\left(y_t\right) = \mathrm{softmax}\left(W^{\mathrm{gen}} h_t^{\mathrm{fuse}} + b^{\mathrm{gen}}\right) \quad (22)$$

2.3 Abstractive Reward

In order to produce an abstractive summary, the model cannot exclusively copy from the source document. In particular, the model needs to parse large chunks of the source document and create concise summaries using phrases not in the source document. To encourage this behavior, we propose a novelty metric that promotes the generation of novel words.

We define a *novel* phrase in the summary as one that is not in the source document. Let $\mathrm{ng}\left(x, n\right)$ denote the function that computes the set of unique n-grams in a document x, x^{gen} the generated summary, x^{src} the source document, and $\|s\|$ the number of words in s. The unnormalized novelty metric N is defined as the fraction of unique n-grams in the summary that are novel.

$$N\left(x^{\mathrm{gen}}, n\right) = \frac{\left\|\mathrm{ng}\left(x^{\mathrm{gen}}, n\right) - \mathrm{ng}\left(x^{\mathrm{src}}, n\right)\right\|}{\left\|\mathrm{ng}\left(x^{\mathrm{gen}}, n\right)\right\|} \quad (23)$$

To prevent the model for receiving high novelty rewards by outputting very short summaries, we normalize the metric by the length ratio of the generated and ground-truth summaries. Let x^{gt} denote the ground-truth summary. We define the novelty metric as

$$R^{\mathrm{nov}}\left(x^{\mathrm{gen}}, n\right) = N\left(x^{\mathrm{gen}}, n\right) \frac{\|x^{\mathrm{gen}}\|}{\|x^{\mathrm{gt}}\|} \quad (24)$$

We incorporate the novelty metric as a reward into the policy gradient objective in eq. (15), alongside the original ROUGE-L metric. In doing so, we encourage the model to generate summaries that both overlap with human written ground-truth summaries as well as incorporate novel words not in the source document:

$$R(y) = \lambda^{\text{rou}} R^{\text{rou}}(y^{\text{sam}}) + \lambda^{\text{nov}} R^{\text{nov}}(y^{\text{sam}}) \quad (25)$$

where λ^{rou} and λ^{nov} are hyperparameters that control the weighting of each reward.

3 Experiments

3.1 Datasets

We train our model on the CNN/Daily Mail dataset (Hermann et al., 2015; Nallapati et al., 2016). Previous works on abstractive summarization either use an anonymized version of this dataset or the original article and summary texts. Due to these different formats, it is difficult to compare the overall ROUGE scores and performance between each version. In order to compare against previous results, we train and evaluate on both versions of this dataset. For the anonymized version, we follow the pre-processing steps described in Nallapati et al. (2016), and the pre-processing steps of See et al. (2017) for the the full-text version.

We use named entities and the source document to supervise the model regarding when to use the pointer and when to use the generator (e.g. z_t in eq. (13). Namely, during training, we teach the model to point from the source document if the word in the ground-truth summary is a named entity, an out-of-vocabulary word, or a numerical value that is in the source document. We obtain the list of named entities from Hermann et al. (2015).

3.2 Language Models

For each dataset version, we train a language model consisting of a 400-dimensional word embedding layer and a 3-layer LSTM with each layer having a hidden size of 800 dimensions, except the last input layer which has an output size of 400. The final decoding layer shares weights with the embedding layer (Inan et al., 2017; Press and Wolf, 2016). We also use DropConnect (Wan et al., 2013) in the hidden-to-hidden connections, as well as the non-monotonically triggered asynchronous gradient descent optimizer from Merity et al. (2018).

We train this language model on the CNN/Daily Mail ground-truth summaries only, following the same training, validation, and test splits as our main experiments.

3.3 Training details

The two LSTMs of our bidirectional encoder are 200-dimensional, and out decoder LSTM is 400-dimensional. We restrict the input vocabulary for the embedding matrix to 150,000 tokens, and the output decoding layer to 50,000 tokens. We limit the size of input articles to the first 400 tokens, and the summaries to 100 tokens. We use scheduled sampling (Bengio et al., 2015) with a probability of 0.25 when calculating the maximum-likelihood training loss. We also set $n = 3$ when computing our novelty reward $R^{nov}(x_{gen}, n)$. For our final training loss using reinforcement learning, we set $\gamma = 0.9984$, $\lambda^{rou} = 0.9$, and $\lambda^{nov} = 0.1$. Finally, we use the trigram repetition avoidance heuristic defined by Paulus et al. (2017) during beam search decoding to ensure that the model does not output twice the same trigram in a given summary, reducing the amount of repetitions.

3.4 Novelty baseline

We also create a novelty baseline by taking the outputs of our base model, without RL training and without the language model, and inserting random words not present in the article after each summary token with a probability $r = 0.0005$. This baseline will intuitively have a higher percentage of novel n-grams than our base model outputs while being very similar to these original outputs, hence keeping the ROUGE score difference relatively small.

4 Results

4.1 Quantitative analysis

We obtain a validation and test perplexity of 65.80 and 66.61 respectively on the anonymized dataset, and 81.13 and 82.98 on the full-text dataset with the language models described in Section 3.2.

The ROUGE scores and novelty scores of our final summarization model on both versions of the CNN/Daily Mail dataset are shown in Table 2. We report the ROUGE-1, ROUGE-2, and ROUGE-L F-scores as well as the percentage of novel n-grams, marked NN-n, in the generated summaries, with n from 1 to 4. Results are omitted in cases where they have not been made available by previous authors. We also include the novel n-gram scores for the ground-truth summaries as a comparison to indicate the level of abstraction of human written summaries.

Model	**R-1**	**R-2**	**R-L**	**NN-1**	**NN-2**	**NN-3**	**NN-4**
			anonymized				
Ground-truth summaries	-	-	-	*14.40*	*52.07*	*71.63*	*80.84*
ML+RL, intra-attn (Paulus et al., 2017)	39.87	**15.82**	36.9	1.04	10.86	21.53	29.27
ML+RL ROUGE+Novel, with LM (ours)	**40.02**	15.53	**37.44**	**3.54**	**21.91**	**37.48**	**47.13**
			full-text				
Ground-truth summaries	-	-	-	*13.55*	*49.97*	*70.32*	*80.02*
Pointer-gen + coverage (See et al., 2017)	39.53	17.28	36.38	0.07	2.24	6.03	9.72
SumGAN (Liu et al., 2018)	39.92	17.65	36.71	0.22	3.15	7.68	11.84
RSal (Pasunuru and Bansal, 2018)	40.36	17.97	37.00	-	2.37	6.00	9.50
RSal+Ent RL (Pasunuru and Bansal, 2018)	**40.43**	**18.00**	37.10	-	-	-	-
ML+RL ROUGE+Novel, with LM (ours)	40.19	17.38	**37.52**	**3.25**	**17.21**	**30.46**	**39.47**

Table 2: Comparison of ROUGE (R-) and novel n-gram (NN-) test results for our model and other abstractive summarization models on the CNN/Daily Mail dataset.

Even though our model outputs significantly fewer novel n-grams than human written summaries, it has a much higher percentage of novel n-grams than all the previous abstractive approaches. It also achieves state-of-the-art ROUGE-L performance on both dataset versions, and obtains ROUGE-1 and ROUGE-2 scores close to state-of-the-art results.

4.2 Ablation study

In order to evaluate the relative impact of each of our individual contributions, we run ablation studies comparing our model ablations against each other and against the novelty baseline. The results of these different models on the validation set of the anonymized CNN/Daily Mail dataset are shown in Table 3. Results show that our base model trained with the maximum-likelihood loss only and using the language model in the decoder (ML, with LM) has higher ROUGE scores, novel unigrams, and novel bigrams scores than our base model without the language model (ML). ML with LM also beats the novelty baseline for these metrics. When training these models with reinforcement learning using the ROUGE reward (ML+RL ROUGE and ML+RL ROUGE with LM), the model with language model obtains higher ROUGE-1 and ROUGE-2 scores. However, it also loses its novel unigrams and bigrams advantage. Finally, using the mixed ROUGE and novelty rewards (ML+RL ROUGE+Novel) produces both higher ROUGE scores and more novel unigrams with the language model than without it. This indicates that the combination of the language model in the decoder and the novelty reward during training makes our model produce more novel unigrams while maintaining high ROUGE scores.

4.3 ROUGE vs novelty trade-off

In order to understand the correlation between ROUGE and novel n-gram scores across different architectures, and to find the model type that gives the best trade-off between each of these metrics, we plot the ROUGE-1 and novel unigram scores for the five best iterations of each model type on the anonymized dataset, as well as the ROUGE-2 and novel bigram scores on a separate plot. We also include the novelty baseline described in Section 4.2 for values of r between 0.005 and 0.035. For each model type, we indicate the Pareto frontier by a line plot (Ben-Tal, 1980), illustrating which models of a given type give the best combination of ROUGE and novelty scores. These plots are shown in Figure 2.

These plots show that there exist an inverse correlation between ROUGE and novelty scores in all model types, illustrating the challenge of choosing a model that performs well in both. Given that, our final model (ML+RL ROUGE+Novel, with LM) provides the best trade-off of ROUGE-1 scores compared to novel unigrams, indicated by the higher Pareto frontier in the first plot. Similarly, our final model gives one of the best trade-offs of ROUGE-2 scores to novel bigrams, even though the same model without LM produces more novel

Model	R-1	R-2	R-L	NN-1	NN-2	NN-3	NN-4
ML	39.21	15.47	36.27	2.47	14.1	25.35	33.46
ML with nov. baseline, $r = 0.0005$	38.62	15.06	35.75	3.12	14.96	26.45	34.76
ML with LM	39.43	15.68	36.45	3.36	15.25	26.06	33.57
ML+RL ROUGE	41.02	16.62	38.13	2.2	12.88	24.16	32.5
ML+RL ROUGE, with LM	**41.06**	**16.84**	38.01	2.06	10.9	19.78	26.33
ML+RL ROUGE+Novel	40.61	15.84	38.06	3.19	**22.79**	**39.9**	**50.61**
ML+RL ROUGE+Novel, with LM	40.72	15.95	**38.14**	**3.49**	21.89	37.31	46.85

Table 3: Ablation study on the validation set of the anonymized CNN/Daily Mail dataset.

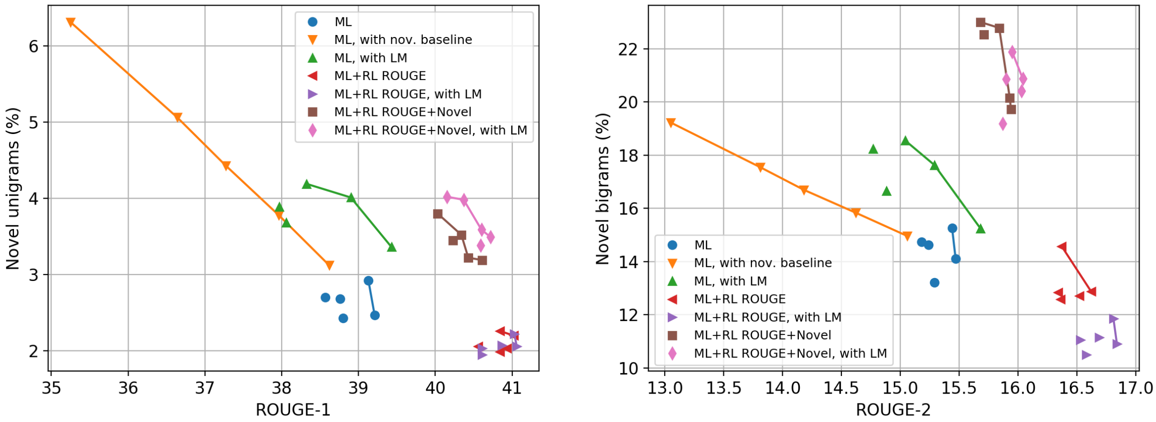

Figure 2: ROUGE and novel n-grams results on the anonymized validation set for different runs of each model type. Lines indicates the Pareto frontier for each model type.

bigrams with a lower ROUGE-2 score.

4.4 Qualitative evaluation

In order to ensure the quality of our model outputs, we ask 5 human evaluators to rate 100 randomly selected full-text test summaries, giving them two scores from 1 to 10 respectively for readability and relevance given the original article. We also include the full-text test outputs from See et al. (2017) and Liu et al. (2018) for comparison. Evaluators are shown different summaries corresponding to the same article side by side without being told which models have generated them. The mean score and confidence interval at 95% for each model and each evaluation criterion are reported in Table 4. These results show that our model matches the relevance score of See et al. (2017) and Liu et al. (2018), but is slightly inferior to them in terms of readability.

5 Related work

Text summarization. Existing summarization approaches are usually either extractive or abstractive. In extractive summarization, the model selects passages from the input document and combines them to form a shorter summary, sometimes with a post-processing step to ensure final coherence of the output (Neto et al., 2002; Dorr et al., 2003; Filippova and Altun, 2013; Colmenares et al., 2015; Nallapati et al., 2017). While extractive models are usually robust and produce coherent summaries, they cannot create concise summaries that paraphrase the source document using new phrases.

Abstractive summarization allows the model to paraphrase the source document and create concise summaries with phrases not in the source document. The state-of-the-art abstractive summarization models are based on sequence-to-sequence models with attention (Bahdanau et al., 2015). Extensions to this model include a self-attention mechanism (Paulus et al., 2017) and an article coverage vector (See et al., 2017) to prevent repeated phrases in the output summary. Different training procedures have also been used improve the ROUGE score (Paulus et al., 2017) or textual

Model	Readability	Relevance
Pointer-gen + coverage (See et al., 2017)	**6.76** ± 0.17	**6.73** ± 0.17
SumGAN (Liu et al., 2018)	6.79 ± 0.16	6.74 ± 0.17
ML+RL ROUGE+Novel, with LM	6.35 ± 0.19	6.63 ± 0.18

Table 4: Mean and confidence interval at 95% of human evaluation scores on the full-text test outputs. Individual summaries are rated from 1 to 10, a higher score indicating higher quality, for readability and relevance separately.

entailment (Pasunuru and Bansal, 2018) with reinforcement learning; as well as generative adversarial networks to generate more natural summaries (Liu et al., 2018).

Several datasets have been used to train and evaluate summarization models. The Gigaword (Graff and Cieri, 2003) and some DUC datasets (Over et al., 2007) have been used for headline generation models (Rush et al., 2015; Nallapati et al., 2016), where the generated summary is shorter than 75 characters. However, generating longer summaries is a more challenging task, especially for abstractive models. Nallapati et al. (2016) have proposed using the CNN/Daily Mail dataset (Hermann et al., 2015) to train models for generating longer, multi-sentence summaries up to 100 words. The New York Times dataset (Sandhaus, 2008) has also been used as a benchmark for the generation of long summaries (Durrett et al., 2016; Paulus et al., 2017).

Training strategies for sequential models. The common approach to training models for sequence generation is maximum likelihood estimation with teacher forcing. At each time step, the model is given the previous ground-truth output and predicts the current output. The sequence objective is the accumulation of cross entropy losses from each time step.

Despite its popularity, this approach for sequence generation is suboptimal due to exposure bias (Huszar, 2015) and loss-evaluation mismatch (Wiseman and Rush, 2016). Goyal et al. (2016) propose one way to reduce exposure bias by explicitly forcing the hidden representations of the model to be similar during training and inference. Bengio et al. (2015) and Wiseman and Rush (2016) propose an alternate method that exposes the network to the test dynamics during training. Reinforcement learning methods (Sutton and Barto, 1998), such as policy learning (Sutton et al., 1999), mitigate the mismatch between the optimization objective and the evaluation metrics by directly optimizing evaluation metrics. This approach has led to consistent improvements in domains such as image captioning (Zhang et al., 2017) and abstractive text summarization (Paulus et al., 2017).

A recent approach to training sequential models utilizes generative adversarial networks to improving the human perceived quality of generated outputs (Fedus et al., 2018; Guimaraes et al., 2017; Liu et al., 2018). Such models use an additional discriminator network that distinguishes between natural and generated output to guide the generative model towards outputs akin to human-written text.

6 Conclusions

We introduced a new abstractive summarization model which uses an external language model in the decoder, as well as a new reinforcement learning reward to encourage summary abstraction. Experiments on the CNN/Daily Mail dataset show that our model generates summaries that are much more abstractive that previous approaches, while maintaining high ROUGE scores close to or above the state of the art. Future work could be done on closing the gap to match human levels of abstraction, which is still very far ahead from our model in terms of novel n-grams. Including mechanisms to promote paraphrase generation in the summary generator could be an interesting direction.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Aharon Ben-Tal. 1980. Characterization of pareto and lexicographic optimal solutions. In *Multiple Crite-*

ria Decision Making Theory and Application, pages 1–11. Springer.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Carlos A Colmenares, Marina Litvak, Amin Mantrach, and Fabrizio Silvestri. 2015. Heads: Headline generation as sequence prediction using an abstract feature-rich space. In *HLT-NAACL*, pages 133–142.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *HLT-NAACL*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *ACL*.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the ______. In *ICLR*.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of EMNLP*, pages 1481–1491. Citeseer.

Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *NIPS*.

David Graff and C Cieri. 2003. English gigaword, linguistic data consortium.

Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. 2017. Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models. *CoRR*, abs/1705.10843.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

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

Ferenc Huszar. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *CoRR*, abs/1511.05101.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *ICLR*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2018. Generative adversarial network for abstractive text summarization. In *AAAI*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. In *ICLR*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Ramesh Nallapati, Bowen Zhou, Çağlar Gülçehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *Proceedings of SIGNLL Conference on Computational Natural Language Learning*.

Joel Larocca Neto, Alex A Freitas, and Celso AA Kaestner. 2002. Automatic text summarization using a machine learning approach. In *Brazilian Symposium on Artificial Intelligence*, pages 205–215. Springer.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Inf. Process. Manage.*, 43(6):1506–1520.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. *CoRR*, abs/1804.06451.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *ICLR*.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *Proceedings of EMNLP*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. 2015. Gradient estimation using stochastic computation graphs. In *NIPS*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2017. Cold fusion: Training seq2seq models together with language models. *CoRR*, abs/1708.06426.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *ICML*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Li Zhang, Flood Sung, Feng Liu, Tao Xiang, Shaogang Gong, Yongxin Yang, and Timothy M. Hospedales. 2017. Actor-critic sequence training for image captioning. *CoRR*, abs/1706.09601.

Content Selection in Deep Learning Models of Summarization

Chris Kedzie and **Kathleen McKeown**
Department of Computer Science
Columbia University
{kedzie,kathy}@cs.columbia.edu

Hal Daumé III
University of Maryland, College Park
Microsoft Research, New York City
hal@cs.umd.edu

Abstract

We carry out experiments with deep learning models of summarization across the domains of news, personal stories, meetings, and medical articles in order to understand how content selection is performed. We find that many sophisticated features of state of the art extractive summarizers do not improve performance over simpler models. These results suggest that it is easier to create a summarizer for a new domain than previous work suggests and bring into question the benefit of deep learning models for summarization for those domains that do have massive datasets (i.e., news). At the same time, they suggest important questions for new research in summarization; namely, new forms of sentence representations or external knowledge sources are needed that are better suited to the sumarization task.

1 Introduction

Content selection is a central component in many natural language generation tasks, where, given a generation goal, the system must determine which information should be expressed in the output text (Gatt and Krahmer, 2018). In summarization, content selection is usually accomplished through sentence (and, occasionally, phrase) extraction. Despite being a key component of both extractive and abstractive summarization systems, it is is not well understood how deep learning models perform content selection with only word and sentence embedding based features as input. Non-neural network approaches often use frequency and information theoretic measures as proxies for content salience (Hong and Nenkova, 2014), but these are not explicitly used in most neural network summarization systems.

In this paper, we seek to better understand how deep learning models of summarization perform content selection across multiple domains (§ 4): news, personal stories, meetings, and medical articles (for which we collect a new corpus).[1] We analyze several recent sentence extractive neural network architectures, specifically considering the design choices for sentence encoders (§ 3.1) and sentence extractors (§ 3.2). We compare Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) based sentence representations to the simpler approach of word embedding averaging to understand the gains derived from more sophisticated architectures. We also question the necessity of auto-regressive sentence extraction (i.e. using previous predictions to inform future predictions), which previous approaches have used (§ 2), and propose two alternative models that extract sentences independently.

Our main results (§ 5) reveal:

1. Sentence position bias dominates the learning signal for news summarization, though not for other domains.[2] Summary quality for news is only slightly degraded when content words are omitted from sentence embeddings.
2. Word embedding averaging is as good or better than either RNNs or CNNs for sentence embedding across all domains.
3. Pre-trained word embeddings are as good, or better than, learned embeddings in five of six datasets.
4. Non auto-regressive sentence extraction performs as good or better than auto-regressive extraction in all domains.

Taken together, these and other results in the paper suggest that we are over-estimating the abil-

---

[1] Data preprocessing and implementation code can be found here: https://github.com/kedz/nnsum/tree/emnlp18-release

[2] This is a known bias in news summarization (Nenkova, 2005).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1818–1828
Brussels, Belgium, October 31 - November 4, 2018.

ity of deep learning models to learn robust and meaningful content features for summarization. In one sense, this might lessen the burden of applying neural network models of content to other domains; one really just needs in-domain word embeddings. However, if we want to learn something other than where the start of the article is, we will need to design other means of sentence representation, and possibly external knowledge representations, better suited to the summarization task.

2 Related Work

The introduction of the CNN-DailyMail corpus by Hermann et al. (2015) allowed for the application of large-scale training of deep learning models for summarization. Cheng and Lapata (2016) developed a sentence extractive model that uses a word level CNN to encode sentences and a sentence level sequence-to-sequence model to predict which sentences to include in the summary. Subsequently, Nallapati et al. (2017) proposed a different model using word-level bidirectional RNNs along with a sentence level bidirectional RNN for predicting which sentences should be extracted. Their sentence extractor creates representations of the whole document and computes separate scores for salience, novelty, and location. These works represent the state-of-the-art for deep learning-based extractive summarization and we analyze them further in this paper.

Other recent neural network approaches include, Yasunaga et al. (2017), who learn a graph-convolutional network (GCN) for multi-document summarization. They do not closely examine the choice of sentence encoder, which is one of the focuses of the present paper; rather, they study the best choice of graph structure for the GCN, which is orthogonal to this work.

Non-neural network learning-based approaches have also been applied to summarization. Typically they involve learning n-gram feature weights in linear models along with other non-lexical word or structural features (Berg-Kirkpatrick et al., 2011; Sipos et al., 2012; Durrett et al., 2016). In this paper, we study representation learning in neural networks that can capture more complex word level feature interactions and whose dense representations are more compatible with current practices in NLP.

The previously mentioned works have focused on news summarization. To further understand the content selection process, we also explore other domains of summarization. In particular, we explore personal narrative summarization based on stories shared on Reddit (Ouyang et al., 2017), workplace meeting summarization (Carletta et al., 2005), and medical journal article summarization (Mishra et al., 2014).

While most work on these summarization tasks often exploit domain-specific features (e.g. speaker identification in meeting summarization (Galley, 2006; Gillick et al., 2009)), we purposefully avoid such features in this work in order to understand the extent to which deep learning models can perform content selection using only surface lexical features. Summarization of academic literature (including medical journals), has long been a research topic in NLP (Kupiec et al., 1995; Elhadad et al., 2005), but most approaches have explored facet-based summarization (Jaidka et al., 2017), which is not the focus of our work.

3 Methods

The goal of extractive text summarization is to select a subset of a document's text to use as a summary, i.e. a short gist or excerpt of the central content. Typically, we impose a budget on the length of the summary in either words or bytes. In this work, we focus on *sentence* extractive summarization, where the basic unit of extraction is a sentence and impose a word limit as the budget.

We model the sentence extraction task as a sequence tagging problem, following (Conroy and O'Leary, 2001). Specifically, given a document containing n sentences $s_1, \ldots, s_n$ we generate a summary by predicting a corresponding label sequence $y_1, \ldots, y_n \in \{0, 1\}^n$, where $y_i = 1$ indicates the i-th sentence is to be included in the summary. Each sentence is itself a sequence of word embeddings $s_i = w_1^{(i)}, \ldots, w_{|s_i|}^{(i)}$ where $|s_i|$ is the length of the sentence in words. The word budget $c \in \mathbb{N}$ enforces a constraint that the total summary word length $\sum_{i=1}^{n} y_i \cdot |s_i| \leq c$.

For a typical deep learning model of extractive summarization there are two main design decisions: *a)* the choice of *sentence encoder* which maps each sentence s_i to an embedding h_i, and *b)* the choice of *sentence extractor* which maps a sequence of sentence embeddings $h = h_1, \ldots, h_n$ to a sequence of extraction decisions $y = y_1, \ldots, y_n$.

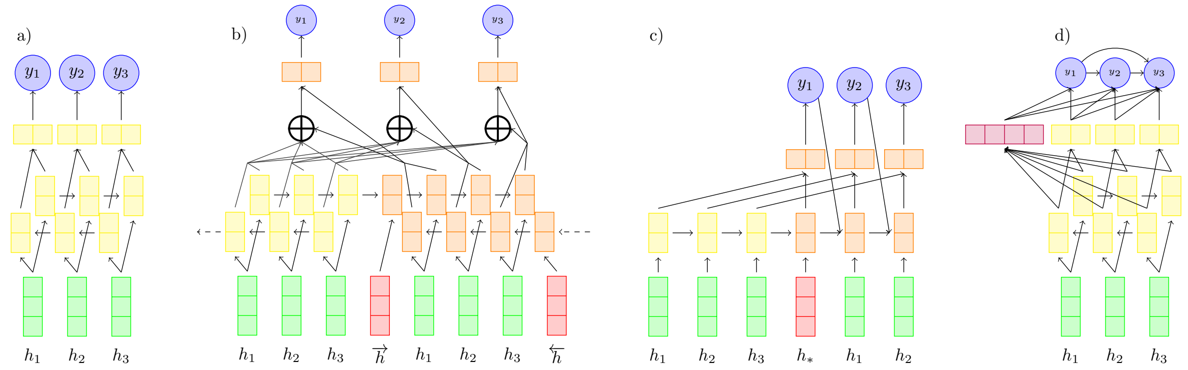

Figure 1: Sentence extractor architectures: a) RNN, b) Seq2Seq, c) Cheng & Lapata, and d) SummaRunner. The $\bigoplus$ indicates attention. Green blocks repesent sentence encoder output and red blocks indicates learned "begin decoding" embeddings. Vertically stacked yellow and orange boxes indicate extractor encoder and decoder hidden states respectively. Horizontal orange and yellow blocks indicate multi-layer perceptrons. The purple blocks represent the document and summary state in the SummaRunner extractor.

3.1 Sentence Encoders

We experiment with three architectures for mapping sequences of word embeddings to a fixed length vector: averaging, RNNs, and CNNs. Hyperparameter settings and implementation details can be found in Appendix A.

Averaging Encoder Under the averaging encoder, a sentence embedding h is simply the average of its word embeddings, i.e. $h = \frac{1}{|s|}\sum_{i=1}^{|s|} w_i$.

RNN Encoder When using the *RNN* sentence encoder, a sentence embedding is the concatenation of the final output states of a forward and backward RNN over the sentence's word embeddings. We use a Gated Recurrent Unit (GRU) for the RNN cell (Chung et al., 2014).

CNN Encoder The *CNN* sentence encoder uses a series of convolutional feature maps to encode each sentence. This encoder is similar to the convolutional architecture of Kim (2014) used for text classification tasks and performs a series of "one-dimensional" convolutions over word embeddings. The final sentence embedding h is a concatenation of all the convolutional filter outputs after max pooling over time.

3.2 Sentence Extractors

Sentence extractors take sentence embeddings $h_{1:n}$ and produce an extract $y_{1:n}$. The sentence extractor is essentially a discriminative classifier $p(y_{1:n}|h_{1:n})$. Previous neural network approaches to sentence extraction have assumed an auto-regressive model, leading to a semi-Markovian factorization of the extractor probabilities $p(y_{1:n}|h) = \prod_{i=1}^{n} p(y_i|y_{<i}, h)$, where each prediction y_i is dependent on *all* previous y_j for all $j < i$. We compare two such models proposed by Cheng and Lapata (2016) and Nallapati et al. (2017). A simpler approach that does not allow interaction among the $y_{1:n}$ is to model $p(y_{1:n}|h) = \prod_{i=1}^{n} p(y_i|h)$, which we explore in two proposed extractor models that we refer to as the RNN and Seq2Seq extractors. Implementation details for all extractors are in Appendix B.

Previously Proposed Sentence Extractors We consider two recent state-of-the-art extractors.

The first, proposed by Cheng and Lapata (2016), is built around a sequence-to-sequence model. First, each sentence embedding[3] is fed into an encoder side RNN, with the final encoder state passed to the first step of the decoder RNN. On the decoder side, the same sentence embeddings are fed as input to the decoder and decoder outputs are used to predict each y_i. The decoder input is weighted by the previous extraction probability, inducing the dependence of y_i on $y_{<i}$. See Figure 1.c for a graphical layout of the extractor.

Nallapati et al. (2017) proposed a sentence extractor, which we refer to as the SummaRunner Extractor, that factorizes the extraction probability into contributions from different sources. First, a bidirectional RNN is run over the sentence em-

---

[3] Cheng and Lapata (2016) used an CNN sentence encoder with this extractor architecture; in this work we pair the Cheng & Lapata extractor with several different encoders.

beddings[4] and the output is concatenated. A representation of the whole document is made by averaging the RNN output. A summary representation is also constructed by taking the sum of the previous RNN outputs weighted by their extraction probabilities. Extraction predictions are made using the RNN output at the i-th step, the document representation, and i-th version of the summary representation, along with factors for sentence location in the document. The use of the iteratively constructed summary representation creates a dependence of y_i on all $y_{<i}$. See Figure 1.d for a graphical layout.

Proposed Sentence Extractors We propose two sentence extractor models that make a stronger conditional independence assumption $p(y|h) = \prod_{i=1}^{n} p(y_i|h)$, essentially making independent predictions conditioned on h.

RNN Extractor Our first proposed model is a very simple bidirectional RNN based tagging model. As in the RNN sentence encoder we use a GRU cell. The forward and backward outputs of each sentence are passed through a multi-layer perceptron with a logsitic sigmoid output to predict the probability of extracting each sentence. See Figure 1.a for a graphical layout.

Seq2Seq Extractor One shortcoming of the RNN extractor is that long range information from one end of the document may not easily be able to affect extraction probabilities of sentences at the other end. Our second proposed model, the Seq2Seq extractor mitigates this problem with an attention mechanism commonly used for neural machine translation (Bahdanau et al., 2014) and abstractive summarization (See et al., 2017). The sentence embeddings are first encoded by a bidirectional GRU. A separate decoder GRU transforms each sentence into a query vector which attends to the encoder output. The attention weighted encoder output and the decoder GRU output are concatenated and fed into a multi-layer perceptron to compute the extraction probability. See Figure 1.b for a graphical layout.

4 Datasets

We perform our experiments across six corpora from varying domains to understand how different biases within each domain can affect content selection. The corpora come from the news domain (CNN-DailyMail, New York Times, DUC), personal narratives domain (Reddit), workplace meetings (AMI), and medical journal articles (PubMed). See Table 1 for dataset statistics.

Dataset	Train	Valid	Test	Refs
CNN/DM	287,113	13,368	11,490	1
NYT	44,382	5,523	6,495	1.93
DUC	516	91	657	2
Reddit	404	24	48	2
AMI	98	19	20	1
PubMed	21,250	1,250	2,500	1

Table 1: Sizes of the training, validation, test splits for each dataset and the average number of test set human reference summaries per document.

CNN-DailyMail We use the preprocessing and training, validation, and test splits of See et al. (2017). This corpus is a mix of news on different topics including politics, sports, and entertainment.

New York Times The New York Times (NYT) corpus (Sandhaus, 2008) contains two types of abstracts for a subset of its articles. The first summary is an archival abstract and the second is a shorter online teaser meant to entice a viewer of the webpage to click to read more. From this collection, we take all articles that have a concatenated summary length of at least 100 words. We create training, validation, and test splits by partitioning on dates; we use the year 2005 as the validation data, with training and test partitions including documents before and after 2005 respectively.

DUC We use the single document summarization data from the 2001 and 2002 Document Understanding Conferences (DUC) (Over and Liggett, 2002). We split the 2001 data into training and validation splits and reserve the 2002 data for testing.

AMI The AMI corpus (Carletta et al., 2005) is a collection of real and staged office meetings annotated with text transcriptions, along with abstractive summaries. We use the prescribed splits.

---

[4] Nallapati et al. (2017) use an RNN sentence encoder with this extractor architecture; in this work we pair the SummaRunner extractor with different encoders.

Extractor	Enc.	CNN/DM		NYT		DUC 2002		Reddit		AMI		PubMed	
		M	R-2	M	R-2	M	R-2	M	R-2	M	R-2	M	R-2
Lead	–	24.1	24.4	30.0	32.3	25.1	21.5	**20.1**	**10.9**	12.3	2.0	15.9	9.3
RNN	Avg.	**25.2**	25.4	29.8	34.7	**26.8**	22.7	**20.4**	**11.4**	**17.0**	**5.5**	19.8	17.0
	RNN	25.1	25.4	29.6	34.9	**26.8**	22.6	**20.2**	**11.4**	16.2	**5.2**	19.7	16.6
	CNN	25.0	25.1	29.0	33.7	**26.7**	**22.7**	**20.9**	**12.8**	14.4	3.2	19.9	16.8
Seq2Seq	Avg.	**25.2**	**25.6**	**30.5**	**35.7**	**27.0**	**22.8**	**20.9**	**13.6**	**17.0**	**5.5**	**20.1**	**17.7**
	RNN	**25.1**	25.3	30.2	**35.9**	**26.7**	22.5	**20.5**	**12.0**	16.1	**5.3**	19.7	16.7
	CNN	25.0	25.1	29.9	35.1	**26.7**	**22.7**	**20.7**	**13.2**	14.2	2.9	19.8	16.9
Cheng & Lapata	Avg.	25.0	25.3	30.4	**35.6**	**27.1**	**23.1**	**20.9**	**13.6**	**16.7**	**6.1**	**20.1**	**17.7**
	RNN	25.0	25.0	**30.3**	**35.8**	**27.0**	**23.0**	**20.3**	**12.6**	**16.3**	**5.0**	19.7	16.7
	CNN	**25.2**	25.1	29.9	35.0	**26.9**	**23.0**	**20.5**	**13.4**	14.3	2.8	19.9	16.9
Summa Runner	Avg.	25.1	25.4	30.2	35.4	26.7	22.3	**21.0**	**13.4**	**17.0**	**5.6**	19.9	17.2
	RNN	25.1	25.2	30.0	35.5	26.5	22.1	**20.9**	**12.5**	**16.5**	**5.4**	19.7	16.5
	CNN	24.9	25.0	29.3	34.4	26.4	22.2	**20.4**	**12.3**	14.5	3.2	19.8	16.8
Oracle	–	31.1	36.2	35.3	48.9	31.3	31.8	24.3	16.2	8.1	3.9	24.1	25.0

Table 2: METEOR (M) and ROUGE-2 recall (R-2) results across all extractor/encoder pairs. Results that are statistically indistinguishable from the best system are shown in bold face.

Reddit Ouyang et al. (2017) collected a corpus of personal stories shared on Reddit[5] along with multiple extractive and abstractive summaries. We randomly split this data using roughly three and five percent of the data validation and test respectively.

PubMed We created a corpus of 25,000 randomly sampled medical journal articles from the PubMed Open Access Subset[6]. We only included articles if they were at least 1000 words long and had an abstract of at least 50 words in length. We used the article abstracts as the ground truth human summaries.

4.1 Ground Truth Extract Summaries

Since we do not typically have ground truth extract summaries from which to create the labels y_i, we construct gold label sequences by greedily optimizing ROUGE-1, using the algorithm in Appendix C. We choose to optimize for ROUGE-1 rather than ROUGE-2 similarly to other optimization based approaches to summarization (Sipos et al., 2012; Durrett et al., 2016) which found this to be the easier target to learn.

5 Experiments

We evaluate summary quality using ROUGE-2 recall (Lin, 2004); ROUGE-1 and ROUGE-LCS trend similarity in our experiments. We use target word lengths of 100 words for news, and 75, 290, and 200 for Reddit, AMI, and PubMed respectively. We also evaluate using METEOR (Denkowski and Lavie, 2014).[7] Summaries are generated by extracting the top ranked sentences by model probability $p(y_i = 1|y_{<i}, h)$, stopping when the word budget is met or exceeded. We estimate statistical significance by averaging each document level score over the five random initializations. We then test the difference between the best system on each dataset and all other systems using the approximate randomization test (Riezler and Maxwell, 2005) with the Bonferroni correction for multiple comparisons, testing for significance at the 0.05 level.

5.1 Training

We train all models to minimize the weighted negative log-likelihood

$$\mathcal{L} = -\sum_{\substack{s,y \in \mathcal{D} \\ h=\text{enc}(s)}} \sum_{i=1}^{n} \omega(y_i) \log p\left(y_i | y_{<i}, h\right)$$

---

[5] `www.reddit.com`

[6] `https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/`

[7] We use the default settings for METEOR and use remove stopwords and no stemming options for ROUGE, keeping defaults for all other parameters.

Ext.	Emb.	CNN/DM	NYT	DUC	Reddit	AMI	PubMed
Seq2Seq	Fixed	**25.6**	**35.7**	**22.8**	**13.6**	5.5	**17.7**
	Learn	25.3 (0.3)	**35.7** (0.0)	**22.9** (-0.1)	**13.8** (-0.2)	**5.8** (-0.3)	16.9 (0.8)
C&L	Fixed	**25.3**	**35.6**	**23.1**	**13.6**	**6.1**	**17.7**
	Learn	24.9 (0.4)	35.4 (0.2)	**23.0** (0.1)	**13.4** (0.2)	**6.2** (-0.1)	16.4 (1.3)
Summa Runner	Fixed	**25.4**	**35.4**	**22.3**	**13.4**	**5.6**	**17.2**
	Learn	25.1 (0.3)	35.2 (0.2)	**22.2** (0.1)	12.6 (0.8)	**5.8** (-0.2)	16.8 (0.4)

Table 3: ROUGE-2 recall across sentence extractors when using fixed pretrained embeddings or when embeddings are updated during training. In both cases embeddings are initialized with pretrained GloVe embeddings. All extractors use the averaging sentence encoder. When both learned and fixed settings are bolded, there is no signifcant performance difference. RNN extractor is omitted for space but is similar to Seq2Seq. Difference in scores shown in parenthesis.

Ablation	CNN/DM	NYT	DUC	Reddit	AMI	PubMed
all words	**25.4**	**34.7**	22.7	**11.4**	5.5	**17.0**
-nouns	25.3† (0.1)	34.3† (0.4)	22.3† (0.4)	10.3† (1.1)	3.8† (1.7)	15.7† (1.3)
-verbs	25.3† (0.1)	34.4† (0.3)	22.4† (0.3)	10.8 (0.6)	5.8 (-0.3)	16.6† (0.4)
-adj/adv	25.3† (0.1)	34.4† (0.3)	22.5 (0.2)	9.5† (1.9)	5.4 (0.1)	16.8† (0.2)
-function	25.2† (0.2)	34.5† (0.2)	**22.9**† (-0.2)	10.3† (1.1)	**6.3**† (-0.8)	16.6† (0.4)

Table 4: ROUGE-2 recall after removing nouns, verbs, adjectives/adverbs, and function words. Ablations are performed using the averaging sentence encoder and the RNN extractor. Bold indicates best performing system. † indicates significant difference with the non-ablated system. Difference in score from *all words* shown in parenthesis.

over the training data $\mathcal{D}$ using stochastic gradient descent with the ADAM optimizer (Kingma and Ba, 2014). $\omega(0) = 1$ and $\omega(1) = N_0/N_1$ where N_y is the number of training examples with label y. We trained for a maximum of 50 epochs and the best model was selected with early stopping on the validation set according to ROUGE-2. Each epoch constitutes a full pass through the dataset. The average stopping epoch was: CNN-DailyMail, 16.2; NYT, 21.36; DUC, 37.11; Reddit, 36.59; AMI, 19.58; PubMed, 19.84. All experiments were repeated with five random initializations. Unless specified, word embeddings were initialized using pretrained GloVe embeddings (Pennington et al., 2014) and we did not update them during training. Unknown words were mapped to a zero embedding. See Appendix D for more optimization and training details.

5.2 Baselines

Lead As a baseline we include the lead summary, i.e. taking the first x words of the document as summary, where x is the target summary length for each dataset (see the first paragraph of § 5). While incredibly simple, this method is still a competitive baseline for single document summarization, especially on newswire.

Oracle To measure the performance ceiling, we show the ROUGE/METEOR scores using the extractive summary which results from greedily optimizing ROUGE-1. I.e., if we had clairvoyant knowledge of the human reference summary, the oracle system achieves the (approximate) maximum possible ROUGE scores. See Appendix C for a detailed description of the oracle algorithm.

5.3 Results

The results of our main experiment comparing the different extractors/encoders are shown in Table 2. Overall, we find no major advantage when using the CNN and RNN sentence encoders over the averaging encoder. The best performing encoder/extractor pair either uses the averaging encoder (five out of six datasets) or the differences are not statistically significant.

When looking at extractors, the Seq2Seq extractor is either part of the best performing system (three out of six datasets) or is not statistically distinguishable from the best extractor.

Overall, on the news and medical journal domains, the differences are quite small with the dif-

Ext.	Order	CNN/DM		NYT		DUC		Reddit		AMI		PubMed	
Seq2Seq	In-Order	**25.6**		**35.7**		**22.8**		**13.6**		5.5		**17.7**	
	Shuffled	21.7	(3.9)	25.6	(10.1	21.2	(1.6)	**13.5**	(0.1)	**6.0**	(-0.5)	14.9	(2.8)

Table 5: ROUGE-2 recall using models trained on in-order and shuffled documents. Extractor uses the averaging sentence encoder. When both in-order and shuffled settings are bolded, there is no signifcant performance difference. Difference in scores shown in parenthesis.

Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas. The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph. An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo. **On Saturday, Hurricane Florence was downgraded to a tropical storm and its remnants pushed inland from the U.S. Gulf Coast.** Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night.	Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas. The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph. An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo. Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night. **Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds and up to 12 feet feet to Puerto Rico's south coast.**

Table 6: Example output of Seq2Seq extractor (left) and Cheng & Lapata Extractor (right). This is a typical example, where only one sentence is different between the two (shown in bold).

ferences between worst and best systems on the CNN/DM dataset spanning only .56 of a ROUGE point. While there is more performance variability in the Reddit and AMI data, there is less distinction among systems: no differences are significant on Reddit and every extractor has at least one configuration that is indistinguishable from the best system on the AMI corpus. This is probably due to the small test size of these datasets.

Word Embedding Learning Given that learning a sentence encoder (averaging has no learned parameters) does not yield significant improvement, it is natural to consider whether learning word embeddings is also necessary. In Table 3 we compare the performance of different extractors using the averaging encoder, when the word embeddings are held fixed or learned during training. In both cases, word embeddings are initialized with GloVe embeddings trained on a combination of Gigaword and Wikipedia. When learning embeddings, words occurring fewer than three times in the training data are mapped to an unknown token (with learned embedding).

In all but one case, fixed embeddings are as good or better than the learned embeddings. This is a somewhat surprising finding on the CNN/DM data since it is reasonably large, and learning embeddings should give the models more flexibility to identify important word features.[8] This suggests that we cannot extract much generalizable learning signal from the content other than what is already present from initialization. Even on PubMed, where the language is quite different from the news/Wikipedia articles the GloVe embeddings were trained on, learning leads to significantly worse results.

POS Tag Ablation It is also not well explored what word features are being used by the encoders. To understand which classes of words were most important we ran an ablation study, selectively removing nouns, verbs (including participles and auxiliaries), adjectives & adverbs, and function words (adpositions, determiners, conjunctions). All datasets were automatically tagged using the spaCy part-of-speech (POS) tagger[9]. The embeddings of removed words were replaced with a zero vector, preserving the order and position of the non-ablated words in the sentence. Ablations were performed on training, validation, and test partitions, using the RNN extractor with averaging encoder. Table 4 shows the results of the POS tag ablation experiments. While removing any word class from the representation generally hurts performance (with statistical significance), on the news domains, the absolute values of the

---

[8] The AMI corpus is an exception here where learning *does* lead to small performance boosts, however, only in the Seq2Seq extractor is this diference significant; it is quite possible that this is an artifact of the very small test set size.

[9] https://github.com/explosion/spaCy

differences are quite small (.18 on CNN/DM, .41 on NYT, .3 on DUC) suggesting that the model's predictions are not overly dependent on any particular word types. On the non-news datasets, the ablations have a larger effect (max differences are 1.89 on Reddit, 2.56 on AMI, and 1.3 on PubMed). Removing nouns leads to the largest drop on AMI and PubMed. Removing adjectives and adverbs leads to the largest drop on Reddit, suggesting the intensifiers and descriptive words are useful for identifying important content in personal narratives. Curiously, removing the function word POS class yields a significant improvement on DUC 2002 and AMI.

Document Shuffling Sentence position is a well known and powerful feature for news summarization (Hong and Nenkova, 2014), owing to the intentional lead bias in the news article writing[10]; it also explains the difficulty in beating the lead baseline for single-document summarization (Nenkova, 2005; Brandow et al., 1999). In examining the generated summaries, we found most of the selected sentences in the news domain came from the lead paragraph of the document. This is despite the fact that there is a long tail of sentence extractions from later in the document in the ground truth extract summaries (31%, 28.3%, and 11.4% of DUC, CNN/DM, and NYT training extract labels come from the second half of the document). Because this lead bias is so strong, it is questionable whether the models are learning to identify important content or just find the start of the document. We conduct a sentence order experiment where each document's sentences are randomly shuffled during training. We then evaluate each model performance on the unshuffled test data, comparing to the model trained on unshuffled data; if the models trained on shuffled data drop in performance, then this indicates the lead bias is the relevant factor.

Table 5 shows the results of the shuffling experiments. The news domains and PubMed suffer a significant drop in performance when the document order is shuffled. By comparison, there is no significant difference between the shuffled and in-order models on the Reddit domain, and shuffling actually improves performance on AMI. This suggest that position is being learned by the models in the news/journal article domain even when the model has no explicit position features, and that this feature is more important than either content or function words.

6 Discussion

Learning content selection for summarization in the news domain is severely inhibited by the lead bias. The summaries generated by all systems described here–the prior work and our proposed simplified models–are highly similar to each other and to the lead baseline. The Cheng & Lapata and Seq2Seq extractors (using the averaging encoder) share 87.8% of output sentences on average on the CNN/DM data, with similar numbers for the other news domains (see Table 6 for a typical example). Also on CNN/DM, 58% of the Seq2Seq selected sentences also occur in the lead summary, with similar numbers for DUC, NYT, and Reddit. Shuffling reduces lead overlap to 35.2% but the overall system performance drops significantly; the models are not able to identify important information without position.

The relative robustness of the news domain to part of speech ablation also suggests that models are mostly learning to recognize the stylistic features unique to the beginning of the article, and not the content. Additionally, the drop in performance when learning word embeddings on the news domain suggests that word embeddings alone do not provide very generalizable content features compared to recognizing the lead.

The picture is rosier for non-news summarization where part of speech ablation leads to larger performance differences and shuffling either does not inhibit content selection significantly or leads to modest gains. Learning better word-level representations on these domains will likely require much larger corpora, something which might remain unlikely for personal stories and meetings.

The lack of distinction among sentence encoders is interesting because it echoes findings in the generic sentence embedding literature where word embedding averaging is frustratingly difficult to outperform (Iyyer et al., 2015; Wieting et al., 2015; Arora et al., 2016; Wieting and Gimpel, 2017). The inability to learn useful sentence representations is also borne out in the SummaRunner model, where there are explicit similarity computations between document or summary representations and sentence embeddings; these computations do not seem to add much to the per-

---

[10] `https://en.wikipedia.org/wiki/Inverted_pyramid_(journalism)`

formance as the Cheng & Lapata and Seq2Seq models which lack these features generally perform as well or better. Furthermore, the Cheng & Lapata and SummaRunner extractors both construct a history of previous selection decisions to inform future choices but this does not seem to significantly improve performance over the Seq2Seq extractor (which does not). This suggests that we need to rethink or find novel forms of sentence representation for the summarization task.

A manual examination of the outputs revealed some interesting failure modes, although in general it was hard to discern clear patterns of behaviour other than lead bias. On the news domain, the models consistently learned to ignore quoted material in the lead, as often the quotes provide color to the story but are unlikely to be included in the summary (e.g. *"It was like somebody slugging a punching bag."*). This behavior was most likely triggered by the presence of quotes, as the quote attributions, which were often tokenized as separate sentences, would subsequently be included in the summary despite also not containing much information (e.g. *Gil Clark of the National Hurricane Center said Thursday*).

7 Conclusion

We have presented an empirical study of deep learning based content selection algorithms for summarization. Our findings suggest such models face stark limitations on their ability to learn robust features for this task and that more work is needed on sentence representation for summarization.

8 Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable feedback. Thanks goes out as well to Chris Hidey for his helpful comments.

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract # FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics.

Ronald Brandow, Karl Mitze, and Lisa Rau. 1999. Automatic condensation of electronic publications by sentence selection. In Jan Fagerberg, David C. Mowery, and Richard R. Nelson, editors, *Advances in Automatic Text Summarization*, chapter 19, pages 293–303. MIT Press, Oxford.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. 2005. The ami meeting corpus: A pre-announcement. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 28–39. Springer.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

John M Conroy and Dianne P O'Leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. *arXiv preprint arXiv:1603.08887*.

Noemie Elhadad, M-Y Kan, Judith L Klavans, and KR McKeown. 2005. Customization in a unified framework for summarizing medical literature. *Artificial intelligence in medicine*, 33(2):179–198.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 364–372. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Dan Gillick, Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tur. 2009. A global optimization framework for meeting summarization. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4769–4772. IEEE.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 712–721.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

Kokil Jaidka, Muthu Kumar Chandrasekaran, Sajal Rustagi, and Min-Yen Kan. 2017. Insights from cl-scisumm 2016: the faceted scientific document summarization shared task. *International Journal on Digital Libraries*, pages 1–9.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73. ACM.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Rashmi Mishra, Jiantao Bian, Marcelo Fiszman, Charlene R Weir, Siddhartha Jonnalagadda, Javed Mostafa, and Guilherme Del Fiol. 2014. Text summarization in the biomedical domain: a systematic review of recent research. *Journal of biomedical informatics*, 52:457–467.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Ani Nenkova. 2005. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *AAAI*, volume 5, pages 1436–1441.

Jessica Ouyang, Serina Chang, and Kathy McKeown. 2017. Crowd-sourced iterative annotation for narrative summarization corpora. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 46–51.

Paul Over and Walter Liggett. 2002. Introduction to duc: An intrinsic evaluation of generic news text summarization systems. *Proc. DUC. http://wwwnlpir. nist. gov/projects/duc/guidelines/2002. html*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Stefan Riezler and John T Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for mt. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 57–64.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Large-margin learning of submodular summarization models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 224–233. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. *arXiv preprint arXiv:1705.00364*.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. *arXiv preprint arXiv:1706.06681*.

Improved Semantic-Aware Network Embedding with Fine-Grained Word Alignment

Dinghan Shen, Xinyuan Zhang, Ricardo Henao, Lawrence Carin
Department of Electrical and Computer Engineering
Duke University, Durham, NC, USA
{dinghan.shen, xy.zhang, r.henao, lcarin}@duke.edu

Abstract

Network embeddings, which learn low-dimensional representations for each vertex in a large-scale network, have received considerable attention in recent years. For a wide range of applications, vertices in a network are typically accompanied by rich textual information such as user profiles, paper abstracts, *etc*. We propose to incorporate semantic features into network embeddings by matching important words between text sequences for all pairs of vertices. We introduce a *word-by-word* alignment framework that measures the compatibility of embeddings between word pairs, and then adaptively accumulates these alignment features with a simple yet effective aggregation function. In experiments, we evaluate the proposed framework on three real-world benchmarks for downstream tasks, including link prediction and multi-label vertex classification. Results demonstrate that our model outperforms state-of-the-art network embedding methods by a large margin.

1 Introduction

Networks are ubiquitous, with prominent examples including social networks (*e.g.*, Facebook, Twitter) or citation networks of research papers (*e.g.*, arXiv). When analyzing data from these real-world networks, traditional methods often represent vertices (nodes) as one-hot representations (containing the connectivity information of each vertex with respect to all other vertices), usually suffering from issues related to the inherent sparsity of large-scale networks. This results in models that are not able to fully capture the relationships between vertices of the network (Perozzi et al., 2014; Tu et al., 2016). Alternatively, network embedding (*i.e.*, network representation learning) has been considered, representing each vertex of a network with a *low-dimensional vector* that preserves information on its similarity relative to other vertices. This approach has attracted considerable attention in recent years (Tang and Liu, 2009; Perozzi et al., 2014; Tang et al., 2015; Grover and Leskovec, 2016; Wang et al., 2016; Chen et al., 2016; Wang et al., 2017a; Zhang et al., 2018).

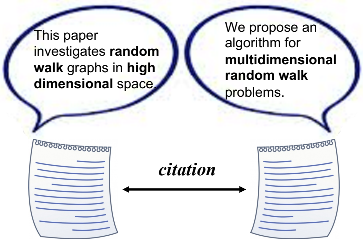

Figure 1: Example of the text information (abstracts) associated to two papers in a citation network. Key words for matching are highlighted.

Traditional network embedding approaches focus primarily on learning representations of vertices that preserve local structure, as well as internal structural properties of the network. For instance, Isomap (Tenenbaum et al., 2000), LINE (Tang et al., 2015), and Grarep (Cao et al., 2015) were proposed to preserve first-, second-, and higher-order proximity between nodes, respectively. DeepWalk (Perozzi et al., 2014), which learns vertex representations from random-walk sequences, similarly, only takes into account structural information of the network. However, in real-world networks, vertices usually contain rich textual information (*e.g.*, user profiles in Facebook, paper abstracts in arXiv, user-generated content on Twitter, *etc*.), which may be leveraged effectively for learning more informative embeddings.

To address this opportunity, Yang et al. (2015) proposed text-associated DeepWalk, to incorporate textual information into the vectorial representations of vertices (embeddings). Sun et al. (2016) employed deep recurrent neural networks to integrate the information from vertex-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1829–1838
Brussels, Belgium, October 31 - November 4, 2018.

associated text into network representations. Further, Tu et al. (2017) proposed to more effectively model the semantic relationships between vertices using a mutual attention mechanism.

Although these methods have demonstrated performance gains over structure-only network embeddings, the relationship between text sequences for a pair of vertices is accounted for solely by comparing their sentence embeddings. However, as shown in Figure 1, to assess the similarity between two research papers, a more effective strategy would compare and align (via local-weighting) individual important words (keywords) within a pair of abstracts, while information from other words (*e.g.*, stop words) that tend to be less relevant can be effectively ignored (down-weighted). This alignment mechanism is difficult to accomplish in models where text sequences are first embedded into a common space and then compared in pairs (He and Lin, 2016; Parikh et al., 2016; Wang and Jiang, 2017; Wang et al., 2017b; Shen et al., 2018a).

We propose to learn a semantic-aware Network Embedding (NE) that incorporates word-level alignment features abstracted from text sequences associated with vertex pairs. Given a pair of sentences, our model first aligns each word within one sentence with keywords from the other sentence (adaptively up-weighted via an attention mechanism), producing a set of fine-grained matching vectors. These features are then accumulated via a simple but efficient aggregation function, obtaining the final representation for the sentence. As a result, the word-by-word alignment features (as illustrated in Figure 1) are explicitly and effectively captured by our model. Further, the learned network embeddings under our framework are adaptive to the specific (local) vertices that are considered, and thus are context-aware and especially suitable for downstream tasks, such as link prediction. Moreover, since the word-by-word matching procedure introduced here is highly parallelizable and does not require any complex encoding networks, such as Long Short-Term Memory (LSTM) or Convolutional Neural Networks (CNNs), our framework requires significantly less time for training, which is attractive for large-scale network applications.

We evaluate our approach on three real-world datasets spanning distinct network-embedding-based applications: link prediction, vertex classification and visualization. We show that the proposed word-by-word alignment mechanism efficiently incorporates textual information into the network embedding, and consistently exhibits superior performance relative to several competitive baselines. Analyses considering the extracted word-by-word pairs further validate the effectiveness of the proposed framework.

2 Proposed Methods

2.1 Problem Definition

A network (graph) is defined as $\boldsymbol{G} = \{\boldsymbol{V}, \boldsymbol{E}\}$, where $\boldsymbol{V}$ and $\boldsymbol{E}$ denote the set of N vertices (nodes) and edges, respectively, where elements of $\boldsymbol{E}$ are two-element subsets of $\boldsymbol{V}$. Here we only consider undirected networks, however, our approach (introduced below) can be readily extended to the directed case. We also define $\boldsymbol{W}$, the symmetric $\mathbb{R}^{N\times N}$ matrix whose elements, w_{ij}, denote the weights associated with edges in $\boldsymbol{V}$, and $\boldsymbol{T}$, the set of text sequences assigned to each vertex. Edges and weights contain the structural information of the network, while the text can be used to characterize the semantic properties of each vertex. Given network $\boldsymbol{G}$, with the *network embedding* we seek to encode each vertex into a low-dimensional vector $\boldsymbol{h}$ (with dimension much smaller than N), while preserving structural and semantic features of $\boldsymbol{G}$.

2.2 Framework Overview

To incorporate both structural and semantic information into the network embeddings, we specify two types of (latent) embeddings: (i) $\boldsymbol{h}_s$, the *structural embedding*; and (ii) $\boldsymbol{h}_t$, the *textual embedding*. Specifically, each vertex in $\boldsymbol{G}$ is encoded into a low-dimensional embedding $\boldsymbol{h} = [\boldsymbol{h}_s; \boldsymbol{h}_t]$. To learn these embeddings, we specify an objective that leverages the information from both $\boldsymbol{W}$ and $\boldsymbol{T}$, denoted as

$$\mathcal{L} = \sum_{\boldsymbol{e}\in\boldsymbol{E}} \mathcal{L}_{\text{struct}}(\boldsymbol{e}) + \mathcal{L}_{\text{text}}(\boldsymbol{e}) + \mathcal{L}_{\text{joint}}(\boldsymbol{e})\,, \quad (1)$$

where $\mathcal{L}_{\text{struct}}$, $\mathcal{L}_{\text{text}}$ and $\mathcal{L}_{\text{joint}}$ denote structure, text, and joint structure-text training losses, respectively. For a vertex pair $\{v_i, v_j\}$ weighted by w_{ij}, $\mathcal{L}_{\text{struct}}(v_i, v_j)$ in (1) is defined as (Tang et al., 2015)

$$\mathcal{L}_{\text{struct}}(v_i, v_j) = w_{ij} \log p(\boldsymbol{h}_s^i | \boldsymbol{h}_s^j)\,, \quad (2)$$

where $p(\boldsymbol{h}_s^i|\boldsymbol{h}_s^j)$ denotes the conditional probability between structural embeddings for vertices $\{v_i, v_j\}$. To leverage the textual information in $\boldsymbol{T}$, similar text-specific and joint structure-text training objectives are also defined

$$\mathcal{L}_{\text{text}}(v_i, v_j) = w_{ij}\alpha_1 \log p(\boldsymbol{h}_t^i|\boldsymbol{h}_t^j)\,, \tag{3}$$

$$\mathcal{L}_{\text{joint}}(v_i, v_j) = w_{ij}\alpha_2 \log p(\boldsymbol{h}_t^i|\boldsymbol{h}_s^j) \tag{4}$$

$$+\, w_{ij}\alpha_3 \log p(\boldsymbol{h}_s^i|\boldsymbol{h}_t^j)\,, \tag{5}$$

where $p(\boldsymbol{h}_t^i|\boldsymbol{h}_t^j)$ and $p(\boldsymbol{h}_t^i|\boldsymbol{h}_s^j)$ (or $p(\boldsymbol{h}_s^i|\boldsymbol{h}_t^j)$) denote the conditional probability for a pair of text embeddings and text embedding given structure embedding (or *vice versa*), respectively, for vertices $\{v_i, v_j\}$. Further, α_1, α_2 and α_3 are hyper-parameters that balance the impact of the different training-loss components. Note that structural embeddings, $\boldsymbol{h}_s$, are treated directly as parameters, while the text embeddings $\boldsymbol{h}_t$ are learned based on the text sequences associated with vertices.

For all conditional probability terms, we follow Tang et al. (2015) and consider the second-order proximity between vertex pairs. Thus, for vertices $\{v_i, v_j\}$, the probability of generating $\boldsymbol{h}_i$ conditioned on $\boldsymbol{h}_j$ may be written as

$$p(\boldsymbol{h}^i|\boldsymbol{h}^j) = \frac{\exp\left({\boldsymbol{h}^j}^T \boldsymbol{h}^i\right)}{\sum_{k=1}^{N} \exp\left({\boldsymbol{h}^j}^T \boldsymbol{h}^k\right)}\,. \tag{6}$$

Note that (6) can be applied to both structural and text embeddings in (2) and (3).

Inspired by Tu et al. (2017), we further assume that vertices in the network play different roles depending on the vertex with which they interact. Thus, for a given vertex, the text embedding, $\boldsymbol{h}_t$, is adaptive (specific) to the vertex it is being conditioned on. This type of context-aware textual embedding has demonstrated superior performance relative to context-free embeddings (Tu et al., 2017). In the following two sections, we describe our strategy for encoding the text sequence associated with an edge into its adaptive textual embedding, via word-by-context and word-by-word alignments.

2.3 Word-by-Context Alignment

We first introduce our base model, which re-weights the importance of individual words within a text sequence in the context of the edge being considered. Consider text sequences associated with two vertices connected by an edge, denoted $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$ and contained in $\boldsymbol{T}$. Text sequences $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$ are of lengths M_a and M_b, respectively, and are represented by $\mathbf{X}_a \in \mathbb{R}^{d\times M_a}$ and $\mathbf{X}_b \in \mathbb{R}^{d\times M_b}$, respectively, where d is the dimension of the word embedding. Further, $\boldsymbol{x}_a^{(i)}$ denotes the embedding of the i-th word in sequence $\boldsymbol{t}_a$.

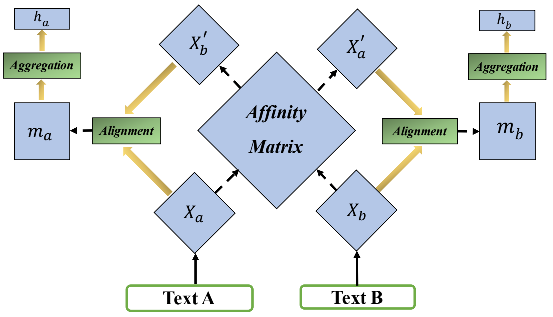

Figure 2: Schematic of the proposed *fine-grained word alignment module* for incorporating textual information into a network embedding. In this setup, word-by-word matching features are explicitly abstracted to infer the relationship between vertices.

Our goal is to encode text sequences $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$ into *counterpart-aware* vectorial representations $\boldsymbol{h}_a$ and $\boldsymbol{h}_b$. Thus, while inferring the adaptive textual embedding for sentence $\boldsymbol{t}_a$, we propose re-weighting the importance of each word in $\boldsymbol{t}_a$ to explicitly account for its alignment with sentence $\boldsymbol{t}_b$. The weight α_i, corresponding to the i-th word in $\boldsymbol{t}_a$, is generated as:

$$\alpha_i = \frac{\exp(\tanh(\mathbf{W}_1 \boldsymbol{c}_b + \mathbf{W}_2 \boldsymbol{x}_a^{(i)}))}{\sum_{j=1}^{M_a} \exp(\tanh(\mathbf{W}_1 \boldsymbol{c}_b + \mathbf{W}_2 \boldsymbol{x}_a^{(j)}))}\,, \tag{7}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are model parameters and $\boldsymbol{c}_b = \sum_{i=1}^{M_b} \boldsymbol{x}_i^b$ is the context vector of sequence $\boldsymbol{t}_b$, obtained by simply averaging over all the word embeddings in the sequence, similar to fastText (Joulin et al., 2016). Further, the *word-by-context* embedding for sequence $\boldsymbol{t}_a$ is obtained by taking the weighted average over all word embeddings

$$\boldsymbol{h}_a = \textstyle\sum_{i=1}^{M_a} \alpha_i \boldsymbol{x}_a^{(i)}\,. \tag{8}$$

Intuitively, α_i may be understood as the relevance score between the ith word in $\boldsymbol{t}_a$ and sequence $\boldsymbol{t}_b$. Specifically, keywords within $\boldsymbol{t}_a$, in the context of $\boldsymbol{t}_b$, should be assigned larger weights, while less important words will be correspondingly down-weighted. Similarly, $\boldsymbol{h}_b$ is encoded as a weighted embedding using (7) and (8).

2.4 Fine-Grained Word-by-Word Alignment

With the alignment in the previous section, word-by-context matching features α_i are modeled; however, the *word-by-word* alignment information (*fine-grained*), which is key to characterize the relationship between two vertices (as discussed in the above), is not explicitly captured. So motivated, we further propose an architecture to explicitly abstract word-by-word alignment information from $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$, to learn the relationship between the two vertices. This is inspired by the recent success of Relation Networks (RNs) for relational reasoning (Santoro et al., 2017).

As illustrated in Figure 2, given two input embedding matrices $\mathbf{X}_a$ and $\mathbf{X}_b$, we first compute the affinity matrix $\mathbf{A} \in \mathbb{R}^{M_b \times M_a}$, whose elements represent the affinity scores corresponding to all word pairs between sequences $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$

$$\mathbf{A} = \mathbf{X}_b^T \mathbf{X}_a \,. \qquad (9)$$

Subsequently, we compute the context-aware matrix for sequence $\boldsymbol{t}_b$ as

$$\mathbf{A}_b = \text{softmax}(\mathbf{A}) \,, \qquad \widetilde{\mathbf{X}}_b = \mathbf{X}_b \mathbf{A}_b \,, \qquad (10)$$

where the softmax$(\cdot)$ function is applied column-wise to $\mathbf{A}$, and thus $\mathbf{A}_b$ contains the attention weights (importance scores) across sequence $\boldsymbol{t}_b$ (columns), which account for each word in sequence $\boldsymbol{t}_a$ (rows). Thus, matrix $\widetilde{\mathbf{X}}_b \in \mathbb{R}^{d \times M_a}$ in (10) constitutes an attention-weighted embedding for $\mathbf{X}_b$. Specifically, the i-th column of $\widetilde{\mathbf{X}}_b$, denoted as $\widetilde{\boldsymbol{x}}_b^{(i)}$, can be understood as a weighted average over all the words in $\boldsymbol{t}_b$, where higher attention weights indicate better alignment (match) with the i-th word in $\boldsymbol{t}_a$.

To abstract the word-by-word alignments, we compare $\boldsymbol{x}_a^{(i)}$ with $\widetilde{\boldsymbol{x}}_b^{(i)}$, for $i = 1, 2, ..., M_a$, to obtain the corresponding matching vector

$$\boldsymbol{m}_a^{(i)} = f_{\text{align}}\left(\boldsymbol{x}_a^{(i)}, \widetilde{\boldsymbol{x}}_b^{(i)}\right) \,, \qquad (11)$$

where $f_{\text{align}}(\cdot)$ represents the alignment function. Inspired by the observation in Wang and Jiang (2017) that simple comparison/alignment functions based on element-wise operations exhibit excellent performance in matching text sequences, here we use a combination of element-wise subtraction and multiplication as

$$f_{\text{align}}(\boldsymbol{x}_a^{(i)}, \widetilde{\boldsymbol{x}}_a^{(i)}) = [\boldsymbol{x}_a^{(i)} - \widetilde{\boldsymbol{x}}_a^{(i)}; \boldsymbol{x}_a^{(i)} \odot \widetilde{\boldsymbol{x}}_a^{(i)}] \,,$$

where $\odot$ denotes the element-wise Hadamard product, then these two operations are concatenated to produce the matching vector $\boldsymbol{m}_a^{(i)}$. Note these operators may be used individually or combined as we will investigate in our experiments.

Subsequently, matching vectors from (11) are aggregated to produce the final textual embedding $\boldsymbol{h}_t^a$ for sequence $\boldsymbol{t}_a$ as

$$\boldsymbol{h}_t^a = f_{\text{aggregate}}\left(\boldsymbol{m}_a^{(1)}, \boldsymbol{m}_a^{(2)}, ..., \boldsymbol{m}_a^{(M_a)}\right) \,, \qquad (12)$$

where $f_{\text{aggregate}}$ denotes the aggregation function, which we specify as the max-pooling pooling operation. Notably, other commutative operators, such as summation or average pooling, can be otherwise employed. Although these aggregation functions are simple and invariant to the order of words in input sentences, they have been demonstrated to be highly effective in relational reasoning (Parikh et al., 2016; Santoro et al., 2017). To further explore this, in Section 5.3, we conduct an ablation study comparing different choices of alignment and aggregation functions.

The representation $\boldsymbol{h}_b$ can be obtained in a similar manner through (9), (10), (11) and (12), but replacing (9) with $\mathbf{A} = \mathbf{X}_a^T \mathbf{X}_b$ (its transpose). Note that this word-by-word alignment is more computationally involved than word-by-context; however, the former has substantially fewer parameters to learn, provided we no longer have to estimate the parameters in (7).

2.5 Training and Inference

For large-scale networks, computing and optimizing the conditional probabilities in (1) using (6) is computationally prohibitive, since it requires the summation over all vertices $\boldsymbol{V}$ in $\boldsymbol{G}$. To address this limitation, we leverage the negative sampling strategy introduced by Mikolov et al. (2013), *i.e.*, we perform computations by sampling a subset of negative edges. As a result, the conditional in (6) can be rewritten as:

$$p(\boldsymbol{h}^i | \boldsymbol{h}^j) = \log \sigma\left({\boldsymbol{h}^j}^T \boldsymbol{h}^i\right) + \sum_{i=1}^{K} \mathbb{E}_{\boldsymbol{h}^i \sim P(v)} \left[\log \sigma\left(-{\boldsymbol{h}^j}^T \boldsymbol{h}^i\right)\right] \,,$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function. Following Mikolov et al. (2013), we set the noise distribution $P(v) \propto d_v^{3/4}$, where d_v is the out-degree of vertex $v \in \boldsymbol{V}$. The number of negative samples K is treated as a hyperparameter. We use Adam (Kingma and Ba, 2014) to update the model parameters while minimizing the objective in (1).

3 Related Work

Network embedding methods can be divided into two categories: (*i*) methods that solely rely on the structure, *e.g.*, vertex information; and (*ii*) methods that leverage both the structure the network and the information associated with its vertices.

For the first type of models, DeepWalk (Perozzi et al., 2014) has been proposed to learn node representations by generating node contexts via truncated random walks; it is similar to the concept of Skip-Gram (Mikolov et al., 2013), originally introduced for learning word embeddings. LINE (Tang et al., 2015) proposed a principled objective to explicitly capture first-order and second-order proximity information from the vertices of a network. Further, Grover and Leskovec (2016) introduced a biased random walk procedure to generate the neighborhood for a vertex, which infers the node representations by maximizing the likelihood of preserving the local context information of vertices. However, these algorithms generally ignore rich heterogeneous information associated with vertices. Here, we focus on incorporating *textual information* into network embeddings.

To learn semantic-aware network embeddings, Text-Associated DeepWalk (TADW) (Yang et al., 2015) proposed to integrate textual features into network representations with matrix factorization, by leveraging the equivalence between DeepWalk and matrix factorization. CENE (Content-Enhanced Network Embedding) (Sun et al., 2016) used bidirectional recurrent neural networks to abstract the semantic information associated with vertices, which further demonstrated the advantages of employing textual information. To capture the interaction between sentences of vertex pairs, Tu et al. (2017) further proposed Context-Aware Network Embedding (CANE), that employs a mutual attention mechanism to adaptively account for the textual information from neighboring vertices. Despite showing improvement over structure-only models, these semantic-aware methods cannot capture word-level alignment information, which is important for inferring the relationship between node pairs, as previously discussed. In this work, we introduce a Word-Alignment-based Network Embedding (WANE) framework, which aligns and aggregates word-by-word matching features in an explicit manner, to obtain more informative network representations.

4 Experimental Setup

Datasets We investigate the effectiveness of the proposed WANE model on two standard network-embedding-based tasks, *i.e.*, link prediction and multi-label vertex classification. The following three real-world datasets are employed for quantitative evaluation: (*i*) *Cora*, a standard paper citation network that contains 2,277 machine learning papers (vertices) grouped into 7 categories and connected by 5,214 citations (edges) (*ii*) *HepTh*, another citation network of 1,038 papers with abstract information and 1,990 citations; (*iii*) *Zhihu*, a network of 10,000 active users from Zhihu, the largest Q&A website in China, where 43,894 vertices and descriptions of the Q&A topics are available. The average lengths of the text in the three datasets are 90, 54, and 190, respectively. To make direct comparison with existing work, we employed the same preprocessing procedure[1] of Tu et al. (2017).

Training Details For fair comparison with CANE (Tu et al., 2017), we set the dimension of network embedding for our model to 200. The number of negative samples K is selected from $\{1, 3, 5\}$ according to performance on the validation set. We set the batch size as 128, and the model is trained using Adam (Kingma and Ba, 2014), with a learning rate of 1×10^{-3} for all parameters. Dropout regularization is employed on the word embedding layer, with rate selected from $\{0.5, 0.7, 0.9\}$, also on the validation set. Our code will be released to encourage future research.

Baselines To evaluate the effectiveness of our framework, we consider several strong baseline methods for comparisons, which can be categorized into two types: (*i*) models that only exploit *structural* information: MMB (Airoldi et al., 2008), DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), and node2vec (Grover and Leskovec, 2016). (*ii*) Models that take both *structural* and *textual* information into account: Naive combination (which simply concatenates the structure-based embedding with CNN-based text embeddings, as explored in (Tu et al., 2017), TADW (Yang et al., 2015), CENE (Sun et al.,

---

[1] https://github.com/thunlp/CANE

%Training Edges	15%	25%	35%	45%	55%	65%	75%	85%	95%
MMB	54.7	57.1	59.5	61.9	64.9	67.8	71.1	72.6	75.9
DeepWalk	56.0	63.0	70.2	75.5	80.1	85.2	85.3	87.8	90.3
LINE	55.0	58.6	66.4	73.0	77.6	82.8	85.6	88.4	89.3
node2vec	55.9	62.4	66.1	75.0	78.7	81.6	85.9	87.3	88.2
Naive combination	72.7	82.0	84.9	87.0	88.7	91.9	92.4	93.9	94.0
TADW	86.6	88.2	90.2	90.8	90.0	93.0	91.0	93.4	92.7
CENE	72.1	86.5	84.6	88.1	89.4	89.2	93.9	95.0	95.9
CANE	86.8	91.5	92.2	93.9	94.6	94.9	95.6	96.6	97.7
WANE	86.1	90.9	92.3	93.1	93.4	94.5	95.1	95.4	95.9
WANE-*wc*	88.7	92.1	92.9	94.4	94.8	95.1	95.7	96.5	97.4
WANE-*ww*	**91.7**	**93.3**	**94.1**	**95.7**	**96.2**	**96.9**	**97.5**	**98.2**	**99.1**

Table 1: AUC scores for link prediction on the *Cora* dataset.

%Training Edges	15%	25%	35%	45%	55%	65%	75%	85%	95%
MMB	54.6	57.9	57.3	61.6	66.2	68.4	73.6	76.0	80.3
DeepWalk	55.2	66.0	70.0	75.7	81.3	83.3	87.6	88.9	88.0
LINE	53.7	60.4	66.5	73.9	78.5	83.8	87.5	87.7	87.6
node2vec	57.1	63.6	69.9	76.2	84.3	87.3	88.4	89.2	89.2
Naive combination	78.7	82.1	84.7	88.7	88.7	91.8	92.1	92.0	92.7
TADW	87.0	89.5	91.8	90.8	91.1	92.6	93.5	91.9	91.7
CENE	86.2	84.6	89.8	91.2	92.3	91.8	93.2	92.9	93.2
CANE	90.0	91.2	92.0	93.0	94.2	94.6	95.4	95.7	96.3
WANE	88.5	90.7	91.1	92.6	93.5	94.2	94.9	95.3	95.8
WANE-*wc*	90.1	91.4	91.9	94.1	95.3	95.9	96.5	96.9	97.2
WANE-*ww*	**92.3**	**94.1**	**95.7**	**96.7**	**97.5**	**97.5**	**97.7**	**98.2**	**98.7**

Table 2: AUC scores for link prediction on the *HepTh* dataset.

2016), and CANE (Tu et al., 2017). It is worth noting that unlike all these baselines, WANE explicitly captures word-by-word interactions between text sequence pairs.

Evaluation Metrics We employ AUC (Hanley and McNeil, 1982) as the evaluation metric for link prediction, which measures the probability that vertices within an existing edge, randomly sampled from the test set, are more similar than those from a random pair of non-existing vertices, in terms of the inner product between their corresponding embeddings.

For multi-label vertex classification and to ensure fair comparison, we follow Yang et al. (2015) and employ a linear SVM on top of the learned network representations, and evaluate classification accuracy with different training ratios (varying from 10% to 50%). The experiments for each setting are repeated 10 times and the average test accuracy is reported.

5 Experimental Results

We experiment with three variants for our WANE model: (*i*) WANE: where the word embeddings of each text sequence are simply average to obtain the sentence representations, similar to (Joulin et al., 2016; Shen et al., 2018c). (*ii*) WANE-*wc*: where the textual embeddings are inferred with word-by-context alignment. (*iii*) WANE-*ww*: where the word-by-word alignment mechanism is leveraged to capture word-by-word matching features between available sequence pairs.

5.1 Link Prediction

Table 1 presents link prediction results for all models on Cora dataset, where different ratios of edges are used for training. It can be observed that when only a small number of edges are available, *e.g.*, 15%, the performances of structure-only methods is much worse than semantic-aware models that have taken textual information into consideration The perfromance gap tends to be smaller when a larger proportion of edges are employed for training. This highlights the importance of incorporating associated text sequences into network embeddings, especially in the case of representing a relatively sparse network. More importantly, the proposed WANE-*ww* model consistently outperforms other semantic-aware NE models by a substantial margin, indicating that our model better abstracts word-by-word alignment features from the text sequences available, thus yields more informative network representations.

Further, WANE-*ww* also outperforms WANE or WANE-*wc* on a wide range of edge training pro-

%Training Edges	15%	25%	35%	45%	55%	65%	75%	85%	95%
MMB	51.0	51.5	53.7	58.6	61.6	66.1	68.8	68.9	72.4
DeepWalk	56.6	58.1	60.1	60.0	61.8	61.9	63.3	63.7	67.8
LINE	52.3	55.9	59.9	60.9	64.3	66.0	67.7	69.3	71.1
node2vec	54.2	57.1	57.3	58.3	58.7	62.5	66.2	67.6	68.5
Naive combination	55.1	56.7	58.9	62.6	64.4	68.7	68.9	69.0	71.5
TADW	52.3	54.2	55.6	57.3	60.8	62.4	65.2	63.8	69.0
CENE	56.2	57.4	60.3	63.0	66.3	66.0	70.2	69.8	73.8
CANE	56.8	59.3	62.9	64.5	68.9	70.4	71.4	73.6	75.4
WANE	52.1	56.6	60.7	64.2	67.5	69.1	71.3	72.8	73.9
WANE-*wc*	55.2	59.9	64.2	68.1	71.3	73.4	75.6	76.3	78.8
WANE-*ww*	**58.7**	**63.5**	**68.3**	**71.9**	**74.9**	**77.0**	**79.7**	**80.0**	**82.6**

Table 3: AUC scores for link prediction on the *Zhihu* dataset.

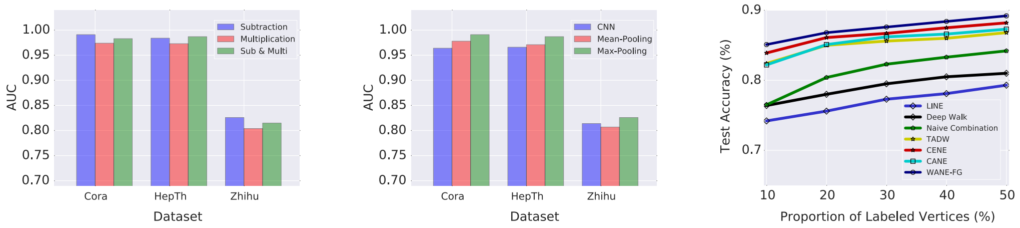

(a) f_{align} (b) $f_{\text{aggregate}}$ (c) vertex classification

Figure 3: (a, b) Ablation study on the choice of different alignment and aggregation functions. (c) Test accuracy of *supervised* vertex classification on the *Cora* dataset.

portions. This suggests that: (*i*) adaptively assigning different weights to each word within a text sequence (according to its paired sequence) tends to be a better strategy than treating each word equally (as in WANE). (*ii*) Solely considering the context-by-word alignment features (as in WANE-*wc*) is not as efficient as abstracting word-by-word matching information from text sequences. We observe the same trend and the superiority of our WANE-*ww* models on the other two datasets, HepTh and Zhihu datasets, as shown in Table 2 and 3, respectively.

5.2 Multi-label Vertex Classification

We further evaluate the effectiveness of proposed framework on vertex classification tasks with the Cora dataset. Similar to Tu et al. (2017), we generate the global embedding for each vertex by taking the average over its *context-aware* embeddings with all other connected vertices. As shown in Figure 3(c), semantic-aware NE methods (including naive combination, TADW, CENE, CANE) exhibit higher test accuracies than semantic-agnostic models, demonstrating the advantages of incorporating textual information. Moreover, WANE-*ww* consistently outperforms other competitive semantic-aware models on a wide range of labeled proportions, suggesting that explicitly capturing word-by-word alignment features is not only useful for vertex-pair-based tasks, such as link prediction, but also results in better global embeddings which are required for vertex classification tasks. These observations further demonstrate that WANE-*ww* is an effective and robust framework to extract informative network representations.

Semi-supervised classification We further consider the case where the training ratio is less than 10%, and evaluate the learned network embedding with a semi-supervised classifier. Following Yang et al. (2015), we employ a Transductive SVM (TSVM) classifier with a linear kernel (Joachims, 1998) for fairness. As illustrated in Table 4, the proposed WANE-*ww* model exhibits superior performances in most cases. This may be due to the fact that WANE-*ww* extracts information from the vertices and text sequences jointly, thus the obtained vertex embeddings are less noisy and perform more consistently with relatively small training ratios (Yang et al., 2015).

5.3 Ablation Study

Motivated by the observation in Wang and Jiang (2017) that the advantages of different functions to match two vectors vary from task to task, we further explore the choice of alignment and aggregation functions in our WANE-*ww* model. To match the word pairs between two sequences, we experimented with three types of operations: *sub-*

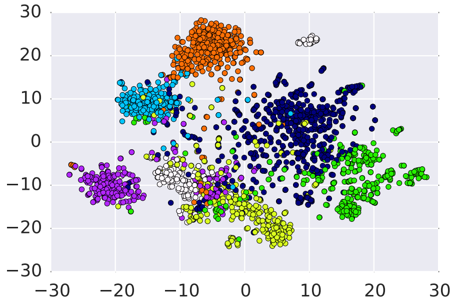

Figure 4: *t*-SNE visualization of the learned network embeddings on the Cora dataset.

traction, *multiplication*, and *Sub & Multi* (the concatenation of both approaches). As shown in Figure 3(a) and 3(b), element-wise *subtraction* tends to be the most effective operation performance-wise on both Cora and Zhihu datasets, and performs comparably to *Sub & Multi* on the HepTh dataset. This finding is consistent with the results in Wang and Jiang (2017), where they found that simple comparison functions based on *element-wise* operations work very well on matching text sequences.

In terms of the aggregation functions, we compare (one-layer) *CNN*, *mean-pooling*, and *max-pooling* operations to accumulate the matching vectors. As shown in Figure 3(b), *max-pooling* has the best empirical results on all three datasets. This may be attributed to the fact that the *max-pooling* operation is better at selecting important word-by-word alignment features, among all matching vectors available, to infer the relationship between vertices.

5.4 Qualitative Analysis

Embedding visualization To visualize the learned network representations, we further employ *t*-SNE to map the low-dimensional vectors of the vertices to a 2-D embedding space. We use the *Cora* dataset because there are labels associated with each vertex and WANE-*ww* to obtain the network embeddings.

As shown in Figure 4 where each point indicates one paper (vertex), and the color of each point indicates the *category* it belongs to, the embeddings of the same label are indeed very close in the 2-D plot, while those with different labels are relatively farther from each other. Note that the model is not trained with any label information, indicating that WANE-*ww* has extracted meaningful patterns from the text and vertex information available.

Baseline Models	1%	3%	7%	10%
Text Only	33.0	43.0	57.1	62.8
Naive Combination	67.4	70.6	75.1	77.4
TADW	72.1	77.0	79.1	81.3
CENE	**73.8**	79.1	81.5	84.5
CANE	72.6	78.2	80.4	83.4
WANE-*ww* (ours)	73.4	**79.6**	**82.7**	**85.1**

Table 4: *Semi-supervised* vertex classification results on the *Cora* dataset.

Case study The proposed word-by-word alignment mechanism can be used to highlight the most informative words (and the corresponding matching features) wrt the relationship between vertices. We visualize the norm of matching vector obtained in (11) in Figure 5 for the Cora dataset. It can be observed that matched key words, *e.g.*, 'MCMC', 'convergence', between the text sequences are indeed assigned higher values in the matching vectors. These words would be selected preferentially by the final *max-pooling* aggregation operation. This indicates that WANE-*ww* is able to abstract important word-by-word alignment features from paired text sequences.

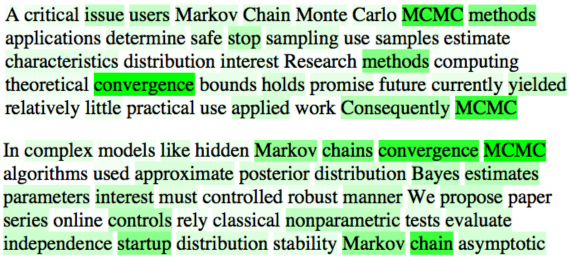
A critical issue users Markov Chain Monte Carlo MCMC methods applications determine safe stop sampling use samples estimate characteristics distribution interest Research methods computing theoretical convergence bounds holds promise future currently yielded relatively little practical use applied work Consequently MCMC

In complex models like hidden Markov chains convergence MCMC algorithms used approximate posterior distribution Bayes estimates parameters interest must controlled robust manner We propose paper series online controls rely classical nonparametric tests evaluate independence startup distribution stability Markov chain asymptotic

(a) Sentence pairs associated with Edge #1

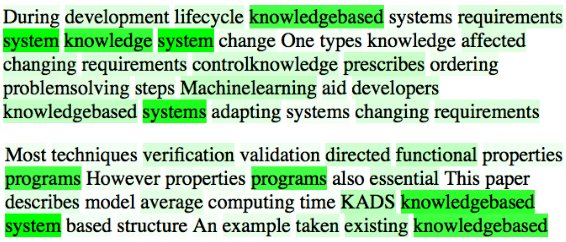
During development lifecycle knowledgebased systems requirements system knowledge system change One types knowledge affected changing requirements controlknowledge prescribes ordering problemsolving steps Machinelearning aid developers knowledgebased systems adapting systems changing requirements

Most techniques verification validation directed functional properties programs However properties programs also essential This paper describes model average computing time KADS knowledgebased system based structure An example taken existing knowledgebased

(b) Sentence pairs associated with Edge #2

Figure 5: Visualization of the word-level *matching vectors*. Darker shades represent larger values of the norm of $\boldsymbol{m}^{(i)}$ at each word position.

6 Conclusions

We have presented a novel framework to incorporate the semantic information from vertex-associated text sequences into network embeddings. An *align-aggregate* framework is introduced, which first aligns a sentence pair by capturing the word-by-word matching features, and then adaptively aggregating these word-level alignment information with an efficient *max-pooling* function. The semantic features abstracted are further encoded, along with the structural information, into a shared space to obtain the final network embedding. Compelling experimental results on several tasks demonstrated the advantages of our approach. In future work, we aim to leverage abundant unlabeled text data to abstract more informative sentence representations (Dai and Le, 2015; Zhang et al., 2017; Shen et al., 2017; Tang and de Sa, 2018) . Another interesting direction is to learn *binary* and compact network embedding, which could be more efficient in terms of both computation and memory, relative to its continuous counterpart (Shen et al., 2018b).

Acknowledgments This research was supported in part by DARPA, DOE, NIH, ONR and NSF.

References

Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. 2008. Mixed membership stochastic blockmodels. *JMLR* 9:1981–2014.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*. ACM, pages 891–900.

Jifan Chen, Qi Zhang, and Xuanjing Huang. 2016. Incorporate group information to enhance network embedding. In *CIKM*. ACM, pages 1901–1904.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*. pages 3079–3087.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*. ACM, pages 855–864.

James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology* 143(1):29–36.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL*. pages 937–948.

Thorsten Joachims. 1998. Making large-scale svm learning practical. Technical report, Technical report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. pages 3111–3119.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *EMNLP* .

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. ACM, pages 701–710.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *NIPS*. pages 4974–4983.

Dinghan Shen, Renqiang Min Martin, Yitong Li, and Lawrence Carin. 2018a. Learning context-sensitive convolutional filters for text processing. In *EMNLP*.

Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Lawrence Carin, and Ricardo Henao. 2018b. Nash: Toward end-to-end neural architecture for generative semantic hashing. *arXiv preprint arXiv:1805.05361* .

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018c. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *ACL*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2017. Deconvolutional latent-variable model for text sequence matching. *arXiv preprint arXiv:1709.07109* .

Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. 2016. A general framework for content-enhanced network representation learning. *arXiv preprint arXiv:1610.02906* .

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. International World Wide Web Conferences Steering Committee, pages 1067–1077.

Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *SIGKDD*. ACM, pages 817–826.

Shuai Tang and Virginia R de Sa. 2018. Multi-view sentence representation learning. *arXiv preprint arXiv:1805.07443* .

Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290(5500):2319–2323.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *ACL*. volume 1, pages 1722–1731.

Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. 2016. Max-margin deepwalk: Discriminative learning of network representation. In *IJCAI*. pages 3889–3895.

Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *SIGKDD*. ACM, pages 1225–1234.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *ICLR*.

Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017a. Community preserving network embedding. In *AAAI*. pages 203–209.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017b. Bilateral multi-perspective matching for natural language sentences. In *IJCAI*.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *IJCAI*. pages 2111–2117.

Xinyuan Zhang, Yitong Li, Dinghan Shen, and Lawrence Carin. 2018. Diffusion maps for textual network embedding. *arXiv preprint arXiv:1805.09906* .

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*. pages 4169–4179.

Learning Context-Aware Convolutional Filters for Text Processing

Dinghan Shen[1], **Martin Renqiang Min**[2], **Yitong Li**[1], **Lawrence Carin**[1]
[1] Duke University [2] NEC Laboratories America
dinghan.shen@duke.edu, renqiang@nec-labs.com, yitong.li@duke.edu, lcarin@duke.edu

Abstract

Convolutional neural networks (CNNs) have recently emerged as a popular building block for natural language processing (NLP). Despite their success, most existing CNN models employed in NLP share the same learned (and static) set of filters for all input sentences. In this paper, we consider an approach of using a small *meta network* to learn context-aware convolutional filters for text processing. The role of meta network is to abstract the contextual information of a sentence or document into a set of *input-aware* filters. We further generalize this framework to model sentence pairs, where a *bidirectional* filter generation mechanism is introduced to encapsulate co-dependent sentence representations. In our benchmarks on four different tasks, including ontology classification, sentiment analysis, answer sentence selection, and paraphrase identification, our proposed model, a modified CNN with context-aware filters, consistently outperforms the standard CNN and attention-based CNN baselines. By visualizing the learned *context-aware* filters, we further validate and rationalize the effectiveness of proposed framework.

1 Introduction

In the last few years, convolutional neural networks (CNNs) have demonstrated remarkable progress in various natural language processing applications (Collobert et al., 2011), including sentence/document classification (Kim, 2014; Zhang et al., 2015; Wang et al., 2018), text sequence matching (Hu et al., 2014; Yin et al., 2016; Shen et al., 2017), generic text representations (Gan et al., 2016; Tang et al., 2018), language modeling (Dauphin et al., 2017), machine translation (Gehring et al., 2017) and abstractive sentence summarization (Gehring et al., 2017). CNNs are typically applied to tasks where feature extraction and a corresponding supervised task are approached jointly (LeCun et al., 1998). As an encoder network for text, CNNs typically convolve a set of filters, of window size n, with an input-sentence embedding matrix obtained via *word2vec* (Mikolov et al., 2013) or Glove (Pennington et al., 2014). Different filter sizes n may be used within the same model, exploiting meaningful semantic features from different n-gram fragments.

The learned weights of CNN filters, in most cases, are assumed to be fixed regardless of the input text. As a result, the rich contextual information inherent in natural language sequences may not be fully captured. As demonstrated in Cohen and Singer (1999), the context of a word tends to greatly influence its contribution to the final supervised tasks. This observation is consistent with the following intuition: when reading different *types* of documents, *e.g.*, academic papers or newspaper articles, people tend to adopt distinct strategies for better and more effective understanding, leveraging the fact that the same words or phrases may have different meaning or imply different things, depending on context.

Several research efforts have sought to incorporate *contextual information* into CNNs to adaptively extract text representations. One common strategy is the *attention mechanism*, which is typically employed on top of a CNN (or Long Short-Term Memory (LSTM)) layer to guide the extraction of semantic features. For the embedding of a single sentence, Lin et al. (2017) proposed a self-attentive model that attends to different parts of a sentence and combines them into multiple vector representations. However, their model needs considerably more parameters to achieve performance gains over traditional CNNs. To match sentence pairs, Yin et al. (2016) introduced an attention-based CNN model, which re-weights the convolution inputs or outputs, to extract interdepen-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1839–1848
Brussels, Belgium, October 31 - November 4, 2018.

dent sentence representations. Wang et al. (2016); Wang and Jiang (2017) explore a *compare and aggregate* framework to directly capture the word-by-word matching between two paired sentences. However, these approaches suffer from the problem of high matching complexity, since a similarity matrix between pairwise words needs to be computed, and thus it is computationally inefficient or even prohibitive when applied to long sentences (Mou et al., 2016).

In this paper, we propose a generic approach to learn *context-aware* convolutional filters for natural language understanding. In contrast to traditional CNNs, the convolution operation in our framework does not have a fixed set of filters, and thus provides the network with stronger modeling flexibility and capacity. Specifically, we introduce a *meta network* to generate a set of context-aware filters, conditioned on specific input sentences; these filters are adaptively applied to either the same (Section 3.2) or different (Section 3.3) text sequences. In this manner, the learned filters vary from sentence to sentence and allow for more fine-grained feature abstraction.

Moreover, since the generated filters in our framework can adapt to different conditional information available (labels or paired sentences), they can be naturally generalized to model sentence pairs. In this regard, we propose a novel *bidirectional* filter generation mechanism to allow interactions between sentence pairs while constructing *context-aware* representations.

We investigate the effectiveness of our Adaptive Context-sensitive CNN (ACNN) framework on several text processing tasks: ontology classification, sentiment analysis, answer sentence selection and paraphrase identification. We show that the proposed methods consistently outperforms the standard CNN and attention-based CNN baselines. Our work provides a new perspective on how to incorporate contextual information into text representations, which can be combined with more sophisticated structures to achieve even better performance in the future.

2 Related Work

Learning deep text representations has attracted much attention recently, since they can potentially benefit a wide range of NLP applications (Collobert et al., 2011; Kim, 2014; Wang et al., 2017a; Shen et al., 2018a; Tang and de Sa, 2018; Zhang et al., 2018). CNNs have been extensively investigated as the encoder networks of natural language. Our work is in line with previous efforts on improving the adaptivity and flexibility of convolutional neural networks (Jeon and Kim, 2017; De Brabandere et al., 2016). Jeon and Kim (2017) proposed to enhance the transformation modeling capacity of CNNs by adaptively learning the filter shapes through backpropagation. De Brabandere et al. (2016) introduced an architecture to generate the future frames conditioned on given image(s), by adapting the CNN filter weights to the motion within previous video frames. Although CNNs have been widely adopted in a large number of NLP applications, improving the adaptivity of vanilla CNN modules has been considerably less studied. To the best of our knowledge, the work reported in this paper is the first attempt to develop more flexible and adjustable CNN architecture for modeling sentences.

Our use of a meta network to generate parameters for another network is directly inspired by the recent success of hypernetworks for text-generation tasks (Ha et al., 2017), and dynamic parameter-prediction for video-frame generation (De Brabandere et al., 2016). In contrast to these works that focus on generation problems, our model is based on context-aware CNN filters and is aimed at abstracting more informative and predictive sentence features. Most similar to our work, Liu et al. (2017) designed a meta network to generate compositional functions over tree-structured neural networks for encapsulating sentence features. However, their model is only suitable for encoding individual sentences, while our framework can be readily generalized to capture the interactions between sentence pairs. Moreover, our framework is based on CNN models, which is advantageous due to fewer parameters and highly parallelizable computations relative to sequential-based models.

3 Model

3.1 Basic CNN for text representations

The CNN architectures in (Kim, 2014; Collobert et al., 2011) are typically utilized for extracting sentence representations, by a composition of a convolutional layer and a *max-pooling* operation over all resulting feature maps. Let the words of a sentence of length T (padded where necessary) be x_1, x_2, ... , x_T. The sentence can be represented as a matrix $\boldsymbol{X} \in \mathbb{R}^{d \times T}$, where each column represents a d-dimensional embedding of the corresponding word.

In the convolutional layer, a set of filters with weights $\boldsymbol{W} \in \mathbb{R}^{K \times h \times d}$ is convolved with every window of h words within the sentence, *i.e.*, $\{x_{1:h}, x_{2:h+1}, \dots, x_{T-h+1:T}\}$, where K is the number of output feature maps (and filters). In this manner, feature maps $\boldsymbol{p}$ for these h-gram text fragments are generated as:

$$\boldsymbol{p}_i = f(\boldsymbol{W} \times x_{i:i+h-1} + b) \qquad (1)$$

where $i = 1, 2, ..., T - h + 1$ and $\times$ denotes the convolution operator at the ith shift location. Parameter $b \in \mathbb{R}^K$ is the bias term and $f(\cdot)$ is a non-linear function, implemented as a rectified linear unit (ReLU) in our experiments. The output feature maps of the convolutional layer, *i.e.*, $\boldsymbol{p} \in \mathbb{R}^{K \times (T-h+1)}$ are then passed to the pooling layer, which takes the maximum value in every row of $\boldsymbol{p}$, forming a K-dimensional vector, $\boldsymbol{z}$. This operation attempts to keep the most salient feature detected by every filter and discard the information from less fundamental text fragments. Moreover, the *max-over-time* nature of the pooling operation (Collobert et al., 2011) guarantees that the size of the obtained representation is independent of the sentence length.

Note that in basic CNN sentence encoders, filter weights are the same for different inputs, which may be suboptimal for feature extraction (De Brabandere et al., 2016), especially in the case where conditional information is available.

3.2 Learning context-sensitive filters

The proposed architecture to learn context-sensitive filters is composed of two principal modules: (*i*) a filter generation module, which produces a set of filters conditioned on the input sentence; and (*ii*) an adaptive convolution module, which applies the generated filters to an input sentence (this sentence may be either the same as or different from the first input, as discussed further in Section 3.3). The two modules are jointly differentiable, and the overall architecture can be trained in an end-to-end manner. Since the generated filters are sample-specific, our ACNN feature extractor for text tends to have stronger predictive power than a basic CNN encoder. The general ACNN framework is shown schematically in Figure 1.

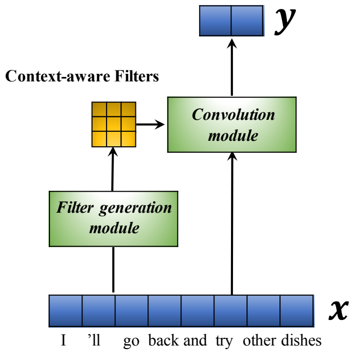

Figure 1: The general ACNN framework. Notably, the input sentences to filter generating module and convolution module could be different (see Section 3.3).

Filter generation module Instead of utilizing fixed filter weights $\boldsymbol{W}$ for different inputs (as (1)), our model generates a set of filters conditioned on the input sentence $\boldsymbol{X}$. Given an input $\boldsymbol{X}$, the filter-generation module can be implemented, in principle, as any deep (differentiable) architecture. However, in order to handle input sentences of variable length common in natural language, we design a generic filter generation module to produce filters with a predefined size.

First, the input $\boldsymbol{X}$ is encapsulated into a fixed-length vector (code) $\boldsymbol{z}$ with the dimension of l, via a basic CNN model, where one convolutional layer is employed along with the pooling operation (as described in Section 3.1). On top of this hidden representation $\boldsymbol{z}$, a deconvolutional layer, which performs transposed operations of convolutions (Radford et al., 2016), is further applied to produce a unique set of filters for $\boldsymbol{X}$ (as illustrated in Figure 1):

$$\boldsymbol{z} = \mathrm{CNN}(\boldsymbol{X}; \boldsymbol{\theta}_e), \qquad (2)$$

$$\boldsymbol{f} = \mathrm{DCNN}(\boldsymbol{z}; \boldsymbol{\theta}_d)\,, \qquad (3)$$

where $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$ are the learned parameters in each layer of the filter-generating module, respectively. Specifically, we convolve $\boldsymbol{z}$ with a filter of size (f_s, l, k_x, k_y), where f_s is the number of generated filters and the kernel size is (k_x, k_y). The output will be a tensor of shape (f_s, k_x, k_y). Since the dimension of hidden representation $\boldsymbol{z}$ is independent of input-sentence length, this framework guarantees that the generated filters are of the same shape and size for every sentence. Intuitively, the encoding part of filter generation module abstracts the information from sentence $\boldsymbol{X}$ into $\boldsymbol{z}$. Based on this representation, the deconvolutional up-sampling layer determines a set of fixed-size, fine-grained filters $\boldsymbol{f}$ for the specific input.

Adaptive convolution module The adaptive convolution module takes as inputs the generated filters $\boldsymbol{f}$ and an input sentence. This sentence and the input to the filter-generation module may be identical (as in Figure 1) or different (as in Figure 2). With the sample-specific filters, the input sentence is adaptively encoded, again, via a basic CNN architecture as in Section 3.1, *i.e.*, one convolutional and one pooling layer. Notably, there are no additional parameters in the adaptive convolution module (no bias term is employed).

Our ACNN framework can be seen as a generalization of the basic CNN, which can be represented as an ACNN by setting the outputs of the filter-generation module to a constant, regardless of the contextual information from input sentence(s). Because of the learning-to-learn (Thrun and Pratt, 2012) nature of the proposed ACNN framework, it tends to have greater representational power than the basic CNN.

3.3 Extension to text sequence matching

Considering the ability of our ACNN framework to generate context-aware filters, it can be naturally generalized to the task of text sequence matching. In this section, we will describe the proposed Adaptive Question Answering (AdaQA) model in the context of answer sentence selection task. Note that the corresponding model can be readily adapted to other sentence matching problems as well (see Section 5.2).

Given a factual question q (associated with a list of candidate answers $\{a_1, a_2, \dots, a_m\}$ and their corresponding labels $\boldsymbol{y} = \{y_1, y_2, \dots, y_m\}$), the goal of the model is to identify the correct answers from the set of candidates. For $i = 1, 2, \dots, m$, if a_i correctly answers q, then $y_i = 1$, and otherwise $y_i = 0$. Therefore, the task can be cast as a classification problem where, given an unlabeled question-answer pair (q_i, a_i), we seek to predict the judgement y_i.

Conventionally, a question q and an answer a are independently encoded by two basic CNNs to fixed-length vector representations, denoted $\boldsymbol{h}_q$ and $\boldsymbol{h}_a$, respectively. They are then directly employed to predict the judgement $\boldsymbol{y}$. This strategy could be suboptimal, since no communication (information sharing) occurs between the question-answer pair until the top prediction layer. Intuitively, while the model is inferring the representation for a question, if the meaning of the answer is taken into account, those features that are relevant for final prediction are more likely to be extracted. So motivated, we propose an adaptive CNN-based question-answer (AdaQA) model for this problem. The AdaQA model can be divided into three modules: filter generation, adaptive convolution, and matching modules, as depicted schematically in Figure 2. Assume there is a question-answer pair to be matched, represented by word-embedding matrices, *i.e.* $\boldsymbol{Q} \in \mathbb{R}^{T_q \times d}$ and $\boldsymbol{A} \in \mathbb{R}^{T_a \times d}$, where d is the embedding dimension and T_q and T_a are respective sentence lengths. First, they are passed to two filter-generation modules, to produce two sets of filters that encapsulate features of the corresponding input sentences. Similar to the setup in Section 3.2, we also employ a two-step process to produce the filters. For a question $\boldsymbol{Q}$, the generating process is:

$$\boldsymbol{z}_q = \text{CNN}(\boldsymbol{Q}; \boldsymbol{\theta}_e^q), \tag{4}$$

$$\boldsymbol{f}_q = \text{DCNN}(\boldsymbol{z}_q; \boldsymbol{\theta}_d^q) \tag{5}$$

where CNN and DCNN denote the basic CNN unit and deconvolution layer, respectively, as discussed in Section 2.1. Parameters $\boldsymbol{\theta}_e^q$ and $\boldsymbol{\theta}_d^q$ are to be learned. The same process can be utilized to produce encodings $\boldsymbol{z}_a$ and filters $\boldsymbol{f}_a$ for the answer input, $\boldsymbol{A}$, with parameters $\boldsymbol{\theta}_e^a$ and $\boldsymbol{\theta}_d^a$, respectively.

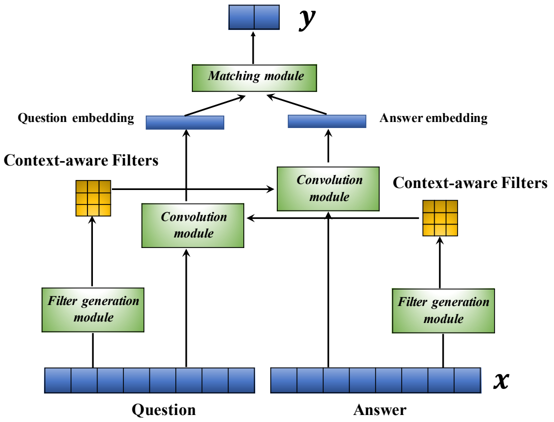

Figure 2: Schematic description of Adaptive Question Answering (AdaQA) model.

The two sets of filter weights are then passed to adaptive convolution modules, along with $\boldsymbol{Q}$ and $\boldsymbol{A}$, to obtain the extracted question and answer embeddings. That is, the question embedding is convolved with the filters produced by the answer and *vise versa* (ψ_q and ψ_a are the bias terms to be learned). The key idea is to abstract information from the answer (or question) that is pertinent to the corresponding question (or answer).

Compared to a Siamese CNN architecture (Bromley et al., 1994), our model selectively encapsulates the most important features for judgement prediction, removing less vital information. We then employ the question and answer representations $\boldsymbol{h}_q \in \mathbb{R}^{n_h}$, $\boldsymbol{h}_a \in \mathbb{R}^{n_h}$ as inputs to the matching module (where n_h is the dimension of question/answer embeddings). Following Mou et al. (2016), the matching function is defined as:

$$\boldsymbol{t} = [\boldsymbol{h}_q; \boldsymbol{h}_a; \boldsymbol{h}_q - \boldsymbol{h}_a; \boldsymbol{h}_q \odot \boldsymbol{h}_a] \tag{6}$$

$$p(\boldsymbol{y} = 1|\boldsymbol{h}_q, \boldsymbol{h}_a) = \text{MLP}(\boldsymbol{t}; \boldsymbol{\eta}') \tag{7}$$

where $-$ and $\odot$ denote an element-wise subtraction and element-wise product, respectively. $[\boldsymbol{h}_a; \boldsymbol{h}_b]$ indicates that $\boldsymbol{h}_a$ and $\boldsymbol{h}_b$ are stacked as column vectors. The resulting matching vector $\boldsymbol{t} \in \mathbb{R}^{4n_h}$ is then sent through an MLP layer (with sigmoid activation function and parameters $\boldsymbol{\eta}'$ to be learned) to model the desired conditional distribution $p(y_i = 1|\boldsymbol{h}_q, \boldsymbol{h}_a)$.

Notably, we share the weights of filter generating networks for both the question and answer, so that the model adaptivity for answer selection can be improved without an excessive increase in the number of parameters. All three modules in AdaQA model are jointly trained end-to-end. Note that the AdaQA model proposed can be readily adapted to other sentence matching tasks, such as paraphrase identification (see Section 5.2).

3.4 Connections to attention mechanism

The adaptive *context-aware* filter generation mechanism proposed here bears close resemblance to attention mechanism (Yin et al., 2016; Bahdanau et al., 2015; Xiong et al., 2017) widely adopted in the NLP community, in the sense that both methods intend to incorporate rich *contextual information* into text representations. However, attention is typically operated on top of the hidden units preprocessed by CNN or LSTM layers, and assigns different weights to each unit according to a *context vector*. By contrast, in our *context-aware* filter generation mechanism, the contextual information is inherently encoded into the convolutional filters, which directly interact with the input sentence during the convolution encoding operation. Notably, according to our experiments, the proposed filter generation module can be readily combined with (standard) attention mechanisms to further enhance the modeling expressiveness of CNN encoder.

Dataset	**# train/ test**	**average #w**	**vocabulary**
Yelp P.	560k/ 38k	138	25,709
DBpedia	560k/ 70k	56	21,666
WikiQA	20,360/ 2,352	7/ 26	10,000
SelQA	66,438/ 19,435	8/ 24	20,000
Quora	390k/ 5,000	13/ 13	20,000

Table 1: Dataset statistics.

4 Experimental Setup

Datasets We investigate the effectiveness of the proposed ACNN framework on both document classification and text sequence matching tasks. Specifically, we consider two large-scale document classification datasets: Yelp Reviews Polarity, and DBPedia ontology datasets (Zhang et al., 2015). For Yelp reviews, we seek to predict a binary label (positive or negative) regarding one review about a restaurant. DBpedia is extracted from Wikipedia by crowd-sourcing and is categorized into 14 non-overlapping ontology classes, including *Company*, *Athlete*, *Natural Place*, *etc*. We sample 15% of the training data as the validation set, to select hyperparameters for our models and perform early stopping. For sentence matching, we evaluate the AdaQA model on two datasets for open-domain question answering: WikiQA (Yang et al., 2015) and SelQA (Jurczyk et al., 2016). Given a question, the task is to rank the corresponding candidate answers, which, in the case of WikiQA, are sentences extracted from the summary section of a related Wikipedia article. To facilitate comparison with existing results (Yin et al., 2016; Yang et al., 2015; Shen et al., 2018b), we truncate the candidate answers to a maximum length of 40 tokens for all experiments on the WikiQA dataset. We also consider the task of paraphrase identification with the Quora Question Pairs dataset, with the same data splits as in (Wang et al., 2017b). A summary of all datasets is presented in Table 1.

Training Details For the document classification experiments, we randomly initialize the word embeddings uniformly within $[-0.001, 0.001]$ and update them during training. For the generated filters, we set the window size as $h = 5$, with $K = 100$ feature maps (the dimension of z is set as 100). For direct comparison, we employ the same filter shape/size settings as in our basic CNN implementation. A one-layer architecture is utilized for both the CNN baseline and the ACNN model, since we did not observe significant performance gains with a multilayer architecture. The minibatch size is set as 128, and a dropout rate of 0.2 is utilized on the embedding layer. We observed that a larger dropout rate (*e.g.*, 0.5) will hurt performance on document classifications and make training significantly slower.

For the sentence matching tasks, we initialized the word embeddings with 50-dimensional Glove (Pennington et al., 2014) word vectors pretrained from Wikipedia 2014 and Gigaword 5 (Pennington et al., 2014) for all model variants. As for the filters, we set the window size as $h = 5$, with $K = 300$ feature maps. As described in Section 3.3, the vector $\boldsymbol{t}$, output from the matching module, is fed to the prediction layer, implemented as a one-layer MLP followed by the sigmoid function. We use Adam (Kingma and Ba, 2014) to train the models, with a learning rate of 3×10^{-4}. Dropout (Srivastava et al., 2014), with a rate of 0.5, is employed on the word embedding layer. The hyperparameters are selected by choosing the best model on the validation set. All models are implemented with TensorFlow (Abadi et al., 2016) and are trained using one NVIDIA GeForce GTX TITAN X GPU with 12GB memory.

Baselines For document classification, we consider several baseline models: (i) ngrams (Zhang et al., 2015), a bag-of-means method based on TFIDF representations built by choosing the 500,000 most frequent n-grams (up to 5-grams) from the training set and use their corresponding counts as features; (ii) small/large word CNN (Zhang et al., 2015): 6 layer word-based convolutional networks, with 256/1024 features at each layer, denoted as small/large, respectively; (iii) deep CNN (Conneau et al., 2016): deep convolutional neural networks with 9/17/29 layers. To evaluate the effectiveness of proposed AdaQA model, we compare it with several CNN-based sequence matching baselines, including Vanilla CNN (Jurczyk et al., 2016; Santos et al., 2017), attentive pooling networks (dos Santos et al., 2016), and ABCNN (Yin et al., 2016) (where an attention mechanism is employed over the two sentence representations).

Evaluation Metrics For document categorization and paraphrase identification tasks, we employ the percentage of correct predictions on the test set to evaluate and compare different models.

Model	Yelp P.	DBpedia
CNN-based Baseline Models		
ngrams*	4.36	1.37
ngrams TFIDF*	4.56	1.31
Small word CNN*	5.54	1.85
Large word CNN*	4.89	1.72
Self-attentive Embedding ‡	3.92	1.14
Deep CNN (9 layer)†	4.88	1.35
Deep CNN (17 layer)†	4.50	1.40
Deep CNN (29 layer)†	4.28	1.29
Our Implementations		
S-CNN	14.48	22.35
S-ACNN	**6.41**	**5.16**
M-CNN	4.58	1.66
M-ACNN	**3.89**	**1.07**

Table 2: Test error rates on document classification tasks (in percentages). S-*model* indicates that the *model* has one single convolutional filter, while M-*model* indicates that the *model* has multiple convolutional filters. Results marked with $*$ are reported by (Zhang et al., 2015), † are reported by (Conneau et al., 2016), and ‡ are reported by (Lin et al., 2017).

For the answer sentence selection task, mean average precision (MAP) and mean reciprocal rank (MRR) are utilized as the corresponding evaluation metrics.

5 Experimental Results

5.1 Document Classification

To explicitly explore whether our ACNN model can leverage the input-aware filter weights for better sentence representation, we perform a comparison between the basic CNN and ACNN models with only a *single* filter, which are denoted as S-CNN, S-ACNN, respectively (this setting may not yield best overall performance, since only a single filter is used, but it allows us to isolate the impact of adaptivity). As illustrated in Table 2, S-ACNN significantly outperforms S-CNN on both datasets, demonstrating the advantage of the filter-generation module in our ACNN framework. As a result, with only one convolutional filter and thus very limited modeling capacity, our S-ACNN model tends to be much more expressive than the basic CNN model, due to the flexibility of applying different filters to different sentences.

We further experiment on both ACNN and CNN models with *multiple* filters. The corresponding document categorization accuracies are presented in Table 2. Although we only use one convolution layer for our ACNN model, it already outperforms other CNN baseline methods with much deeper architectures. Moreover, our method ex-

Model	MAP	MRR
CNN-based Baseline Models		
bigram CNN + *Cnt**	0.6520	0.6652
Attentive Pooling Network	0.6886	0.6957
ABCNN	0.6921	0.7127
Our Implementations		
CNN	0.6752	0.6890
ACNN (self-adaptive)	0.6897	0.7032
AdaQA (one-way)	0.7005	0.7161
AdaQA (two-way)	0.7107	0.7304
AdaQA (two-way) + att.	**0.7325**	**0.7428**

Table 3: Results of our models on WikiQA dataset, compared with previous CNN-based methods.

Model	MAP	MRR
CNN-based Baseline Models		
CNN: *baseline**	0.8320	0.8420
CNN: *average + word**	0.8400	0.8494
CNN: *aver + emb**	0.8466	0.8568
CNN: *hinge_loss*‡	0.8758	0.8812
CNN-DAN‡	0.8655	0.8730
Our Implementations		
CNN	0.8644	0.8720
ACNN (self-adaptive)	0.8739	0.8801
AdaQA (one-way)	0.8823	0.8889
AdaQA (two-way)	0.8914	0.8983
AdaQA (two-way) + att.	**0.9021**	**0.9103**

Table 4: Results of our models on SelQA dataset, compared with previous CNN-based methods. Results marked with $*$ are from (Jurczyk et al., 2016), and marked with ‡ are from (Santos et al., 2017).

hibits higher accuracy than n-grams, which is a very strong baseline as shown in (Zhang et al., 2015). We attribute the superior performance of the ACNN framework to its stronger (adaptive) feature-extraction ability. Moreover, our M-ACNN also achieves slightly better performance than self-attentive sentence embeddings proposed in Lin et al. (2017), which requires significant more parameters than our method.

Effect of number of filters To further demonstrate that the performance gains in document categorization experiments originates from the improved adaptivity of our ACNN framework, we implement the basic CNN model with different numbers of filter sizes, ranging from 1 to 1000. As illustrated in Figure 3(a), when the filter size is larger than 100, the test accuracy of the standard CNN model does not show any noticeable improvement with more filters. More importantly, even with a filter size of 1000, the classification accuracy of the CNN is worse than that of the ACNN model with the filter number restricted to 100. Given these observations, we believe that the boosted categorization accuracy does come from the improved flexibility and thus better feature extraction of our ACNN framework.

5.2 Answer Sentence Selection

To elucidate the role of different parts (modules) in our AdaQA model, we implement several model variants for comparison: (*i*) a "vanilla" CNN model that independently encodes two sentence representations for matching; (*ii*) a self-adaptive ACNN-based model where the question/answer sentence generates adaptive filters only to convolve with the input itself; (*iii*) a one-way ACNN model where only the answer sentence representation is extracted with adaptive filters, which are generated conditioned on the question; (*iv*) a two-way AdaQA model as described in Section 2.4, where both sentences are adaptively encoded, with filters generated conditioned on the other sequence; (*v*) considering that the proposed filter generation mechanism is complementary to the attention layer typically employed in sequence matching tasks (see Section 3.4), we experiment with another model variant that combines the proposed *context-aware* filter generation mechanism with the multi-perspective attention layer introduced in (Wang et al., 2017b).

Tables 3 and 4 show experimental results of our models on WikiQA and SelQA datasets, along with other state-of-the-art methods. Note that the self-adaptive ACNN model variant, which generates filters only for the input itself (without any interactions before the top matching module), slightly outperforms the vanilla CNN Siamese model. Combined with the results in document categorization experiments, we believe that our ACNN framework, in its simplest form, can be utilized as a powerful feature extractor for transforming natural language sentences into fixed-length vectors. More importantly, our two-way AdaQA model exhibits superior results compared with the one-way variant as well as other CNN-based baseline models on the WikiQA dataset. This observation indicates that the *bidirectional* filter generation mechanism is strongly associated with the performance gains. While combined with the multi-perspective attention layers, adopted after the ACNN encoding layer, our two-way AdaQA model achieves even better performance. This suggests that the proposed strategy is complemen-

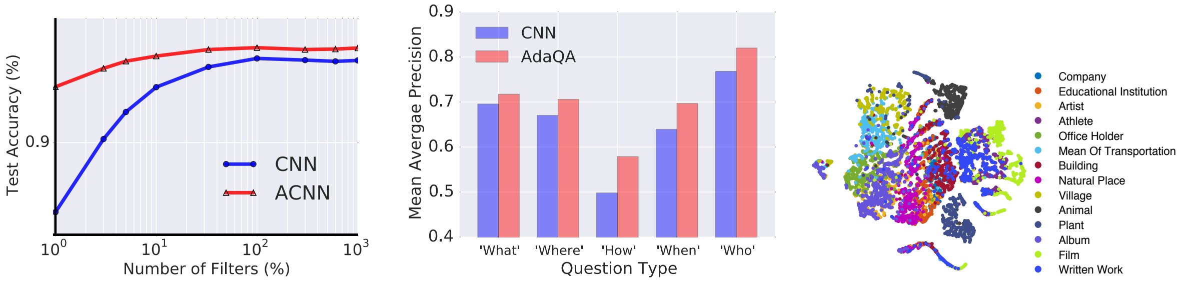

(a) Effect of filter number (b) Different question types (c) t-SNE visualization

Figure 3: Comprehensive study of the proposed ACNN framework, including (a) the number of filters (Yelp dataset), and (b) performance *vs* question types (WikiQA dataset), and (c) t-SNE visualization of learned filter weights (DBpedia dataset).

Model	Accuracy
Siamese-CNN	0.7960
Multi-Perspective-CNN	0. 8138
AdaQA (two-way)	0.8516
AdaQA (two-way) + att.	**0.8794**

Table 5: Results on the Quora Question Pairs dataset.

tary, in terms of the incorporation of rich contextual information, to the standard attention mechanism. The same trend is also observed on the SelQA dataset (as shown in Table 4), which is a much larger dataset than WikiQA.

Notably, our model yields significantly better results than an attentive pooling network and ABCNN (attention-based CNN) baselines. We attribute the improvement to two potential advantages of our AdaQA model: (*i*) for the two previous baseline methods, the interaction between question and answer takes place either before or after convolution. However, in our AdaQA model, the communication between two sentences is inherent in the convolution operation, and thus can provide the abstracted features with more flexibility; (*ii*) the *bidirectional* filter generation mechanism in our AdaQA model generates co-dependent representations for the question and candidate answer, which could enable the model to recover from initial local maxima corresponding to incorrect predictions (Xiong et al., 2017).

Paragraph Identification Considering that the proposed AdaQA model can be readily generalized to other text sequence matching problems, we further evaluate the proposed framework on the paraphrase identification task with the Quora question pairs dataset. To ensure a fair comparison, we employ the same data splits as in (Wang et al., 2017b). As illustrated in Table 5, our two-way AdaQA model again exhibits superior performances compared with basic CNN models (as reported in (Wang et al., 2017b)).

5.3 Discussion

Reasoning ability To associate the improved answer sentence selection results with the reasoning capabilities of our AdaQA model, we further categorize the questions in the WikiQA test set into 5 types containing: 'What', 'Where', 'How', 'When' or 'Who'. We then calculate the MAP scores of the basic CNN and our AdaQA model on different question types. Similar to the findings in (Miao et al., 2016), we observe that the 'How' question is the hardest to answer, with the lowest MAP scores. However, our AdaQA model improves most over the basic CNN on the 'How' type question, see Figure 3(b). Further comparing our results with NASM in (Miao et al., 2016), our AdaQA model (with a MAP score of 0.579) outperforms their reported 'How' question MAP scores (0.524) by a large margin, indicating that the adaptive convolutional filter-generation mechanism improves the model's ability to read and reason over natural language sentences.

Filter visualization To better understand what information has been encoded into our *context-aware* filters, we visualize one of the filters for sentences within the test set (on the DBpedia dataset) with t-SNE. The corresponding results are shown in Figure 3(c). It can be observed that the filters for documents with the same label (ontology) are grouped into clusters, indicating that for different types of document, ACNN has leveraged distinct convolutional filters for better feature extraction.

6 Conclusions

We presented a *context-aware* convolutional filter-generation mechanism, introducing a meta network to adaptively produce a set of input-aware filters. In this manner, the filter weights vary from sample to sample, providing the CNN encoder network with more modeling flexibility and capacity.

This framework is further generalized to model question-answer sentence pairs, leveraging a two-way feature abstraction process. We evaluate our models on several document-categorization and sentence matching benchmarks, and they consistently outperform the standard CNN and attention-based CNN baselines, demonstrating the effectiveness of our framework.

Acknowledgments This research was supported in part by DARPA, DOE, NIH, ONR and NSF.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *NIPS*, pages 737–744.

William W Cohen and Yoram Singer. 1999. Context-sensitive learning methods for text categorization. *TOIS*, 17(2):141–173.

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

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *EACL*.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. *ICML*.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. 2016. Dynamic filter networks. In *NIPS*.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2016. Learning generic sentence representations using convolutional neural networks. *arXiv preprint arXiv:1611.07897*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *ICML*.

David Ha, Andrew Dai, and Quoc V Le. 2017. Hypernetworks. *ICLR*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, pages 2042–2050.

Yunho Jeon and Junmo Kim. 2017. Active convolution: Learning the shape of convolution for image classification. *CVPR*.

Tomasz Jurczyk, Michael Zhai, and Jinho D Choi. 2016. Selqa: A new benchmark for selection-based question answering. In *ICTAI*, pages 820–827. IEEE.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *EMNLP*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *ICLR*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Dynamic compositional neural networks over tree structure. *IJCAI*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICML*, pages 1727–1736.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*.

Cıcero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR, abs/1602.03609*.

Cicero Nogueira dos Santos, Kahini Wadhawan, and Bowen Zhou. 2017. Learning loss functions for semi-supervised learning via discriminative adversarial networks. *arXiv preprint arXiv:1707.02198*.

Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Lawrence Carin, and Ricardo Henao. 2018a. Nash: Toward end-to-end neural architecture for generative semantic hashing. In *ACL*.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018b. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *ACL*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2017. Deconvolutional latent-variable model for text sequence matching. *arXiv preprint arXiv:1709.07109*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia Sa. 2018. Speeding up context-based sentence representation learning with non-autoregressive convolutional decoding. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 69–78.

Shuai Tang and Virginia R de Sa. 2018. Multi-view sentence representation learning. *arXiv preprint arXiv:1805.07443*.

Sebastian Thrun and Lorien Pratt. 2012. *Learning to learn*. Springer Science & Business Media.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. *ICLR*.

Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. 2017a. Topic compositional neural language model. *arXiv preprint arXiv:1712.09783*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017b. Bilateral multi-perspective matching for natural language sentences. *IJCAI*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. *COLING*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. *ICLR*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*, pages 2013–2018.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *TACL*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.

Xinyuan Zhang, Yitong Li, Dinghan Shen, and Lawrence Carin. 2018. Diffusion maps for textual network embedding. *arXiv preprint arXiv:1805.09906*.

Deep Relevance Ranking Using Enhanced Document-Query Interactions

Ryan McDonald[1,2], **Georgios-Ioannis Brokos**[1] and **Ion Androutsopoulos**[1]

[1]Dept. of Informatics, Athens University of Economics and Business, Greece
[2]Google AI

Abstract

We explore several new models for document relevance ranking, building upon the Deep Relevance Matching Model (DRMM) of Guo et al. (2016). Unlike DRMM, which uses context-insensitive encodings of terms and query-document term interactions, we inject rich context-sensitive encodings throughout our models, inspired by PACRR's (Hui et al., 2017) convolutional n-gram matching features, but extended in several ways including multiple views of query and document inputs. We test our models on datasets from the BIOASQ question answering challenge (Tsatsaronis et al., 2015) and TREC ROBUST 2004 (Voorhees, 2005), showing they outperform BM25-based baselines, DRMM, and PACRR.

1 Introduction

Document relevance ranking, also known as *ad-hoc retrieval* (Harman, 2005), is the task of ranking documents from a large collection using the query and the text of each document only. This contrasts with standard information retrieval (IR) systems that rely on text-based signals in conjunction with network structure (Page et al., 1999; Kleinberg, 1999) and/or user feedback (Joachims, 2002). Text-based ranking is particularly important when (i) click-logs do not exist or are small, and (ii) the network structure of the collection is non-existent or not informative for query-focused relevance. Examples include various domains in digital libraries, e.g., patents (Azzopardi et al., 2010) or scientific literature (Wu et al., 2015; Tsatsaronis et al., 2015); enterprise search (Hawking, 2004); and personal search (Chirita et al., 2005).

We investigate new deep learning architectures for document relevance ranking, focusing on *term-based interaction models*, where query terms (*q-terms* for brevity) are scored relative to a document's terms (*d-terms*) and their scores are aggregated to produce a relevance score for the document. Specifically, we use the Deep Relevance Matching Model (DRMM) of Guo et al. (2016) (Fig. 1), which was shown to outperform strong IR baselines and other recent deep learning methods. DRMM uses pre-trained word embeddings for q-terms and d-terms, and cosine similarity histograms (outputs of ⊗ in Fig. 1), each capturing the similarity of a q-term to all the d-terms of a particular document. The histograms are fed to an MLP (dense layers of Fig. 1) that produces the (document-aware) score of each q-term. Each q-term score is then weighted using a gating mechanism (topmost box nodes in Fig. 1) that examines properties of the q-term to assess its importance for ranking (e.g., common words are less important). The sum of the weighted q-term scores is the relevance score of the document. This ignores entirely the contexts where the terms occur, in contrast to recent *position-aware* models such as PACRR (Hui et al., 2017) or those based on recurrent representations (Palangi et al., 2016).

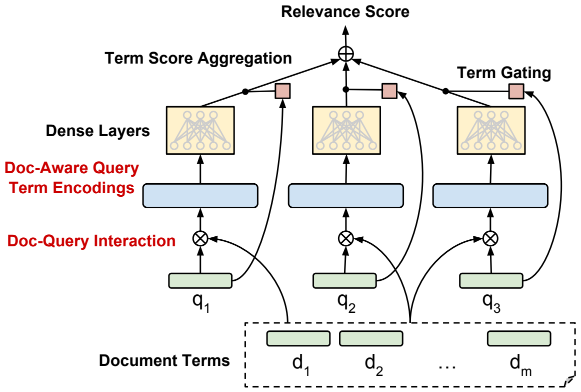

Figure 1: Illustration of DRMM (Guo et al., 2016) for a query of three terms and a document of m terms.

In order to enrich DRMM with context-sensitive representations, we need to change fundamentally how q-terms are scored. This is because rich

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1849–1860
Brussels, Belgium, October 31 - November 4, 2018.

context-sensitive representations – such as input term encodings based on RNNs or CNNs – require end-to-end training and histogram construction is not differentiable. To account for this we investigate novel query-document interaction mechanisms that are differentiable and show empirically that they are effective ways to enable end-to-end training of context-sensitive DRMM models. This is the primary contribution of this paper.

Overall, we explore several extensions to DRMM, including: PACRR-like convolutional n-gram matching features (§3.1); context-sensitive term encodings (§3.2); query-focused attention-based document representations (§3.3); pooling to reward denser term matches and turn rich term representations into fixed-width vectors (§3.4); multiple views of terms, e.g., context sensitive, insensitive, exact matches (§3.5).

We test our models on data from the BIOASQ biomedical question answering challenge (Tsatsaronis et al., 2015) and TREC ROBUST 2004 (Voorhees, 2005), showing that they outperform strong BM25-based baselines (Robertson and Zaragoza, 2009), DRMM, and PACRR.[1]

2 Related Work

Document ranking has been studied since the dawn of IR; classic term-weighting schemes were designed for this problem (Sparck Jones, 1988; Robertson and Sparck Jones, 1988). With the advent of statistical NLP and statistical IR, probabilistic language and topic modeling were explored (Zhai and Lafferty, 2001; Wei and Croft, 2006), followed recently by deep learning IR methods (Lu and Li, 2013; Hu et al., 2014; Palangi et al., 2016; Guo et al., 2016; Hui et al., 2017).

Most document relevance ranking methods fall within two categories: representation-based, e.g., Palangi et al. (2016), or interaction-based, e.g., Lu and Li (2013). In the former, representations of the query and document are generated independently. Interaction between the two only happens at the final stage, where a score is generated indicating relevance. End-to-end learning and backpropagation through the network tie the two representations together. In the interaction-based paradigm, explicit encodings between pairs of queries and documents are induced. This allows direct modeling of exact- or near-matching terms (e.g., synonyms), which is crucial for relevance ranking. Indeed, Guo et al. (2016) showed that the interaction-based DRMM outperforms previous representation-based methods. On the other hand, interaction-based models are less efficient, since one cannot index a document representation independently of the query. This is less important, though, when relevance ranking methods rerank the top documents returned by a conventional IR engine, which is the scenario we consider here.

One set of our experiments ranks biomedical texts. Several methods have been proposed for the BIOASQ challenge (Tsatsaronis et al., 2015), mostly based on traditional IR techniques. The most related work is of Mohan et al. (2017), who use a deep learning architecture. Unlike our work, they focus on user click data as a supervised signal, and they use context-insensitive representations of document-query interactions. The other dataset we experiment with, TREC ROBUST 2004 (Voorhees, 2005), has been used extensively to evaluate traditional and deep learning IR methods.

Document relevance ranking is also related to other NLP tasks. Passage scoring for question answering (Surdeanu et al., 2008) ranks passages by their relevance to the question; several deep networks have been proposed, e.g., Tan et al. (2015). Short-text matching/ranking is also related and has seen recent deep learning solutions (Lu and Li, 2013; Hu et al., 2014; Severyn and Moschitti, 2015). In document relevance ranking, though, documents are typically much longer than queries, which makes methods from other tasks that consider pairs of short texts not directly applicable.

Our starting point is DRMM, to which we add richer representations inspired by PACRR. Hence, we first discuss DRMM and PACRR further.

2.1 DRMM

We have already presented an overview of DRMM. For gating (topmost box nodes of Fig. 1), Guo et al. (2016) use a linear self-attention:

$$g_i = \text{softmax}\Big(w_g^T\, \phi_g(q_i); q_1, \dots, q_n\Big)$$

where $\phi_g(q_i)$ is the embedding $e(q_i)$ of the i-th q-term, or its IDF, $\text{idf}(q_i)$, and w_g is a weights vector. Gating aims to weight the (document-aware) score of each q-term (outputs of dense layers in Fig. 1) based on the importance of the term. We found that $\phi_g(q_i) = [e(q_i); \text{idf}(q_i)]$, where ';' is concatenation, was optimal for all DRMM-based models.

---

[1] The code and data of our experiments, including word embeddings, are available at `https://github.com/nlpaueb/deep-relevance-ranking`.

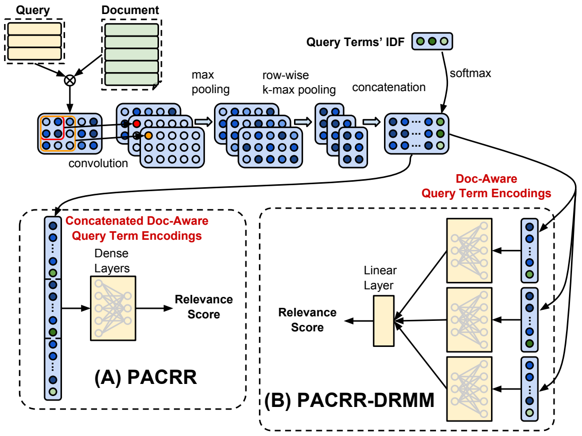

Figure 2: PACRR (Hui et al., 2017) and PACRR-DRMM.

The crux of the original DRMM are the bucketed cosine similarity histograms (outputs of $\otimes$ nodes in Fig. 1), each capturing the similarity of a q-term to all the d-terms. In each histogram, each bucket counts the number of d-terms whose cosine similarity to the q-term is within a particular range. Consider a document with three terms, with cosine similarities, s, to a particular q-term q_i 0.5, 0.1, -0.3, respectively. If we used two buckets $-1 \leq s < 0$ and $0 \leq s \leq 1$, then the input to the dense layers for q_i would be $\langle 1, 2 \rangle$. The fixed number of buckets leads to a fixed-dimension input for the dense layers and makes the model agnostic to different document and query lengths – one of DRMM's main strengths. The main disadvantage is that bucketed similarities are independent of the contexts where terms occur. A q-term 'regulated' will have a perfect match with a d-term 'regulated', even if the former is 'up regulated' and the latter is 'down regulated' in context. Also, there is no reward for term matches that preserve word order, or multiple matches within a small window.

2.2 PACRR

In PACRR (Hui et al., 2017), a query-document term similarity matrix *sim* is computed (Fig. 2A). Each cell (i, j) of *sim* contains the cosine similarity between the embeddings of a q-term q_i and a d-term d_j. To keep the dimensions $l_q \times l_d$ of *sim* fixed across queries and documents of varying lengths, queries are padded to the maximum number of q-terms l_q, and only the first l_d terms per document are retained.[2] Then, convolutions (Fig. 2A) of different kernel sizes $n \times n$ ($n = 2, \ldots, l_g$) are applied to *sim* to capture n-gram query-document similarities. For each size $n \times n$, multiple kernels (filters) are used. Max pooling is then applied along the dimension of the filters (max value of all filters), followed by row-wise k-max pooling to capture the strongest k signals between each q-term and all the d-terms. The resulting matrices are concatenated into a single matrix where each row is a document-aware q-term encoding; the IDF of the q-term is also appended, normalized by applying a softmax across the IDFs of all the q-terms. Following Hui et al. (2018), we concatenate the rows of the resulting matrix into a single vector, which is passed to an MLP (Fig. 2A, dense layers) that produces a query-document relevance score.[3]

The primary advantage of PACRR over DRMM is that it models context via the n-gram convolutions, i.e., denser n-gram matches and matches preserving word order are encoded. However, this context-sensitivity is weak, as the convolutions operate over the similarity matrix, not directly on terms or even term embeddings. Also, unlike DRMM, PACRR requires padding and hyperparameters for maximum number of q-terms (l_q) and d-terms (l_d), since the convolutional and dense layers operate over fixed-size matrices and vectors. On the other hand, PACRR is end-to-end trainable – though Hui et al. (2017) use fixed pre-trained embeddings – unlike DRMM where the bucketed histograms are not differentiable.

3 New Relevance Ranking Models

3.1 PACRR-DRMM

In a DRMM-like version of PACRR, instead of using an MLP (dense layers, Fig. 2A) to score the *concatenation* of all the (document-aware) q-term encodings, the MLP *independently* scores each q-term encoding (the same MLP for all q-terms, Fig. 2B); the resulting scores are aggregated via a linear layer. This version, PACRR-DRMM, performs better than PACRR, using the same number of hidden layers in the MLPs. Likely this is due to the fewer parameters of its MLP, which is shared across the q-term representations and operates on shorter input vectors. Indeed, in early experiments PACRR-DRMM was less prone to over-fitting.

In PACRR-DRMM, the scores of the q-terms (outputs of dense layers, Fig. 2B) are not weighted by a gating mechanism, unlike DRMM (Fig. 1). Nevertheless, the IDFs of the q-terms, which are appended to the q-term encodings (Fig. 2B), are a

---

[2] We use PACRR-*firstk*, which Hui et al. (2017) recommend when documents fit in memory, as in our experiments.

[3] Hui et al. (2017) used an additional LSTM, which was later replaced by the final concatanation (Hui et al., 2018).

form of term-gating (shortcut passing on information about the terms, here their IDFs, to upper layers) applied before scoring the q-terms. By contrast, in DRMM (Fig. 1) term-gating is applied after q-term scoring, and operates on $[e(q_i); \text{idf}(q_i)]$.

3.2 Context-sensitive Term Encodings

In their original incarnations, DRMM and PACRR use pre-trained word embeddings that are insensitive to the context of a particular query or document where a term occurs. This contrasts with the plethora of systems that use context-sensitive word encodings (for each particular occurrence of a word) in virtually all NLP tasks (Bahdanau et al., 2014; Plank et al., 2016; Lample et al., 2016). In general, this is achieved via RNNs, e.g., LSTMs (Gers et al., 2000), or CNNs (Bai et al., 2018).

In the IR literature, context-sensitivity is typically viewed through two lenses: term proximity (Büttcher et al., 2006) and term dependency (Metzler and Croft, 2005). The former assumes that the context around a term match is also relevant, whereas the latter aims to capture when multiple terms (e.g., an n-gram) must be matched together. An advantage of neural network architectures like RNNs and CNNs is that they can capture both.

In the models below (§§3.3–3.4), an encoder produces the *context-sensitive encoding* of each q-term or d-term from the pre-trained embeddings. To compute this we use a standard BILSTM encoding scheme and set the context-sentence encoding as the concatenation of the last layer's hidden states of the forward and backward LSTMs at each position. As is common for CNNs and even recent RNN term encodings (Peters et al., 2018), we use the original term embedding $e(t_i)$ as a residual and combine it with the BILSTM encodings. Specifically, if $\overrightarrow{h}(t_i)$ and $\overleftarrow{h}(t_i)$ are the last layer's hidden states of the left-to-right and right-to-left LSTMs for term t_i, respectively, then we set the context-sensitive term encoding as:

$$c(t_i) = [\overrightarrow{h}(t_i) + e(t_i); \overleftarrow{h}(t_i) + e(t_i)] \quad (1)$$

Since we are adding the original term embedding to each LSTM hidden state, we require the dimensionality of the hidden layers to be equal to that of the original embedding. Other methods were tried, including passing all representations through an MLP, but these had no effect on performance.

This is an orthogonal way to incorporate context into the model relative to PACRR. PACRR creates a query-document similarity matrix and computes n-gram convolutions over the matrix. Here we incorporate context directly into the term encodings; hence similarities in this space are already context-sensitive. One way to view this difference is the point at which context enters the model – directly during term encoding (Eq. 1) or after term similarity scores have been computed (PACRR, Fig. 2).

3.3 ABEL-DRMM

Using the context-sensitive q-term and d-term encodings of §3.2 (Eq. 1), our next extension to DRMM is to create *document-aware* q-term encodings that go beyond bucketed histograms of cosine similarities, the stage in Fig. 1 indicated by $\otimes$. We focus on differentiable encodings to facilitate end-to-end training from inputs to relevance scores.

Figure 3 shows the sub-network that computes the document-aware encoding of a q-term q_i in the new model, given a document $d = \langle d_1, \ldots, d_m \rangle$ of m d-terms. We first compute a dot-product[4] attention score $a_{i,j}$ for each d_j relative to q_i:

$$a_{i,j} = \text{softmax}\Big(c(q_i)^T\, c(d_j); d_1, \ldots, d_m\Big) \quad (2)$$

where $c(t)$ is the context-sensitive encoding of t (Eq. 1). We then sum the context-sensitive encodings of the d-terms, weighted by their attention scores, to produce an attention-based representation d_{q_i} of document d from the viewpoint of q_i:

$$d_{q_i} = \sum_j a_{i,j}\; c(d_j) \quad (3)$$

The Hadamard product (element-wise multiplication, $\odot$) between the (L2-normalized) document representation d_{q_i} and the q-term encoding $c(q_i)$ is then computed and used as the fixed-dimension document-aware encoding $\phi_H(q_i)$ of q_i (Fig. 3):

$$\phi_H(q_i) = \frac{d_{q_i}}{||d_{q_i}||} \odot \frac{c(q_i)}{||c(q_i)||} \quad (4)$$

The $\otimes$ nodes and lower parts of the DRMM network of Fig. 1 are now replaced by (multiple copies of) the sub-network of Fig. 3 (one copy per q-term), with the $\odot$ nodes replacing the $\otimes$ nodes. We call the resulting model Attention-Based ELement-wise DRMM (ABEL-DRMM).

Intuitively, if the document contains one or more terms d_j that are similar to q_i, the attention

---

[4] Dot-products have a larger range than other similarity functions, encouraging low entropy attention distributions.

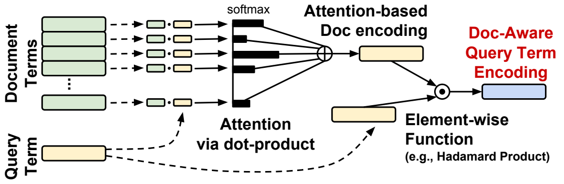

Figure 3: ABEL-DRMM sub-net. From context-aware q-term and d-term encodings (Eq. 1), it generates fixed-dimension *document-aware q-term encodings* to be used in DRMM (Fig. 1, replacing ⊗ nodes).

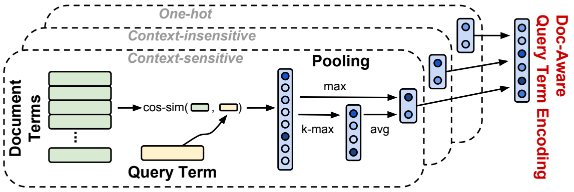

Figure 4: POSIT-DRMM with multiple views (+MV). Three two-dimensional document-aware q-term encodings, one from each view, are produced, concatenated, and used in DRMM (Fig. 1, replacing ⊗ nodes).

mechanism will have emphasized mostly those terms and, hence, d_{q_i} will be similar to $c(q_i)$, otherwise not. This similarity could have been measured by the cosine similarity between d_{q_i} and $c(q_i)$, but the cosine similarity assigns the same weight to all the dimensions, i.e., to all the (L2 normalized) element-wise products in $\phi_H(q_i)$, which cosine similarity just sums. By using the Hadamard product, we pass on to the upper layers of DRMM (the dense layers of Fig. 1), which score each q-term with respect to the document, all the (normalized) element-wise products of $\phi_H(q_i)$, allowing the upper layers to learn which element-wise products (or combinations of them) are important when matching a q-term to the document.

Other element-wise functions can also be used to compare d_{q_i} to $c(q_i)$, instead of the Hadamard product (Eq. 4). For example, a vector containing the squared terms of the Euclidean distance between d_{q_i} and $c(q_i)$ could be used instead of $\phi_H(q_i)$. This change had no effect on ABEL-DRMM's performance on development data. We also tried using $[d_{q_i}; c(q_i)]$ instead of $\phi_H(q_i)$, but performance on development data deteriorated.

ABEL-DRMM is agnostic to document length, like DRMM. ABEL-DRMM, however, is trainable end-to-end, unlike the original DRMM. Still, both models do not reward higher density matches.

3.4 POSIT-DRMM

Ideally, we want models to reward both the maximum match between a q-term and a document, but also the average match (between several q-terms and the document) to reward documents that have a higher density of matches. The document-aware q-term scoring of ABEL-DRMM does not account for this, as the attention summation hides whether a single or multiple terms were matched with high similarity. We also want models to be end-to-end trainable, like ABEL-DRMM.

Figure 4 (context-sensitive box) outlines a simple network that produces document-aware q-term encodings, replacing the ABEL-DRMM sub-network of Fig. 3 in the DRMM framework. We call the resulting model POoled SImilariTy DRMM (POSIT-DRMM). As in ABEL-DRMM, we compute an attention score $a_{i,j}$ for each d_j relative to q_i, now using cosine similarity (cf. Eq. 2):

$$a_{i,j} = \frac{c(q_i)^T c(d_j)}{||c(q_i)||\ ||c(d_j)||} \tag{5}$$

However, we do not use the $a_{i,j}$ scores to compute a weighted average of the encodings of the d-terms (cf. Eq. 3), which is also why there is no softmax in $a_{i,j}$ above (cf. Eq. 2).[5] Instead, we concatenate the attention scores of the m d-terms:

$$a_i = \langle a_{i,1}, \dots, a_{i,j}, \dots, a_{i,m} \rangle^T$$

and we apply two pooling steps on a_i to create a 2-dimensional document-aware encoding $\phi_P(q_i)$ of the q-term q_i (Fig. 4). First max-pooling, which returns the single best match of q_i in the document. Then average pooling over a k-max-pooled version of a_i, which represents the average similarity for the top k matching terms:

$$\phi_P(q_i) = \Big\langle \max(a_i), \text{avg}\Big(\text{k-max}(a_i)\Big) \Big\rangle^T$$

POSIT-DRMM has many fewer parameters than the other models. The input to the upper q-term scoring dense layers of the DRMM framwork (Fig. 1) for ABEL-DRMM has the same dimensionality as pre-trained term embeddings, on the order of hundreds. By contrast, the input dimensionality here is 2. Hence, POSIT-DRMM does not require deep dense layers, but uses a single layer (depth 1). More information on hyperparameters is provided in Appendix A (supplementary material).

POSIT-DRMM is closely related to PACRR (and PACRR-DRMM). Like POSIT-DRMM, PACRR first

---

[5] The $a_{i,j}$s still need to be normalized for input to the upper layers, but they do not need to be positive summing to 1. This is why we use cosine similarity in Eq. 5 instead of dot-products combined with softmax of Eq. 2.

computes cosine similarities between all q-terms and d-terms (Fig. 2). It then applies n-gram convolutions to the similarity matrix to inject context-awareness, and then pooling to create document-aware q-term representations. Instead, POSIT-DRMM relies on the fact that the term encodings are now already context sensitive (Eq. 1) and thus skips the n-gram convolutions. Again, this is a choice of when context is injected – during term encoding or after computing similarity scores.

Mohan et al.'s work (2017) is related in the sense that for each q-term, document-aware encodings are built over the best matching (Euclidean distance) d-term. But again, term encodings are context-insensitive pre-trained word embeddings and the model is not trained end-to-end.

3.5 Multiple Views of Terms (+MV)

An extension to ABEL-DRMM and POSIT-DRMM (or any deep model) is to use multiple views of terms. The basic POSIT-DRMM produces a two-dimensional document-aware encoding of each q-term (Fig. 4, context-sensitive box) viewing the terms as their context-sensitive encodings (Eq. 1). Another two-dimensional document-aware q-term encoding can be produced by viewing the terms directly as their pre-trained embeddings without converting them to context-sensitive encodings (Fig. 4, context-insensitive box). A third view uses one-hot vector representations of terms, which allows exact term matches to be modeled, as opposed to near matches in embedding space. Concatenating the outputs of the 3 views, we obtain 6-dimensional document-aware q-term encodings, leading to a model dubbed POSIT-DRMM+MV. An example of this multi-view document-aware query term representation is given in Fig. 5 for a query-document pair from BIOASQ's development data.

The multi-view extension of ABEL-DRMM (ABEL-DRMM+MV) is very similar, i.e., it uses context-sensitive term encodings, pre-trained term embeddings, and one-hot term encodings in its three views. The resulting three document-aware q-term embeddings can be summed or concatenated, though we found the former more effective.

3.6 Alternative Network Structures

The new models (§§3.1–3.5) were selected by experimenting on development data. Many other extensions were considered, but not ultimately used as they were not beneficial empirically, including deeper and wider RNNs or CNN encoders (Bai et al., 2018); combining document-aware encodings from all models; different attention mechanisms, e.g., multi-head (Vaswani et al., 2017).

Pointer Networks (Vinyals et al., 2015) use the attention scores directly to select an input component. POSIT-DRMM does this via max and average pooling, not argmax. We implemented Pointer Networks – argmax over ABEL-DRMM attention to select the best d-term encoding – but empirically this was similar to ABEL-DRMM. Other architectures considered in the literature include the K-NRM model of Xiong et al. (2017). This is similar to both ABEL-DRMM and POSIT-DRMM in that it can be viewed as an end-to-end version of DRMM. However, it uses kernels over the query-document interaction matrix to produce features per q-term.

The work of Pang et al. (2017) is highly related and investigates many different structures, specifically aimed at incorporating context-sensitivity. However, unlike our work, Pang et al. first extract contexts (n-grams) of documents that match q-terms. Multiple interaction matrices are then constructed for the entire query relative to each of these contexts. These document contexts may match one or more q-terms allowing the model to incorporate term proximity. These interaction matrices can also be constructed using exact string match similar to POSIT-DRMM+MV.

4 Experiments

We experiment with ad-hoc retrieval datasets with hundreds of thousands or millions of documents. As deep learning models are computationally expensive, we first run a traditional IR system[6] using the BM25 score (Robertson and Zaragoza, 2009) and then re-rank the top N returned documents.

4.1 Methods Compared

All systems use an extension proposed by Severyn and Moschitti (2015), where the relevance score is combined via a linear model with a set of *extra features*. We use four extra features: z-score normalized BM25 score; percentage of q-terms with exact match in the document (regular and IDF weighted); and percentage of q-term bigrams matched in the document. The latter three features were taken from Mohan et al. (2017).

In addition to the models of §§2.1, 2.2, 3.1–3.5, we used the following baselines: Standard Okapi

---

[6] We used Galago (`http://www.lemurproject.org/galago.php`, v.3.10). We removed stop words and applied Krovetz's stemmer (Krovetz, 1993).

BM25 (**BM25**); and BM25 re-ranked with a linear model over the four extra features (**BM25+extra**). These IR baselines are very strong and most recently proposed deep learning models do *not* beat them.[7] DRMM and PACRR are also strong baselines and have shown superior performance over other deep learning models on a variety of data (Guo et al., 2016; Hui et al., 2017).[8]

All hyperparameters were tuned on development data and are available in Appendix A. All models were trained using Adam (Kingma and Ba, 2014) with batches containing a randomly sampled negative example per positive example[9] and a pair-wise loss. As the datasets contain only documents marked as relevant, negative examples were sampled from the top N documents (returned by BM25) that had not been marked as relevant.

We evaluated the models using the TREC ad-hoc retrieval evaluation script[10] focusing on MAP, Precision@20 and nDCG@20 (Manning et al., 2008). We trained each model five times with different random seeds and report the mean and standard deviation for each metric on test data; in each run, the model selected had the highest MAP on the development data. We also report results for an **oracle**, which re-ranks the N documents returned by BM25 placing all human-annotated relevant documents at the top. To test for statistical significance between two systems, we employed two-tailed stratified shuffling (Smucker et al., 2007; Dror et al., 2018) using the model with the highest development MAP over the five runs per method.

4.2 BioASQ Experiments

Our first experiment used the dataset of the document ranking task of BIOASQ (Tsatsaronis et al., 2015), years 1–5.[11] It contains 2,251 English biomedical questions, each formulated by a biomedical expert, who searched (via PubMed[12]) for, and annotated relevant documents. Not all relevant documents were necessarily annotated, but the data includes additional expert relevance judgments made during the official evaluation.[13]

The document collection consists of approx. 28M 'articles' (titles and abstracts only) from the 'MEDLINE/PubMed Baseline 2018' collection.[14] We discarded the approx. 10M articles that contained only titles, since very few of these were annotated as relevant. For the remaining 18M articles, a document was the concatenation of each title and abstract. Consult Appendix B for further statistics of the dataset. Word embeddings were pre-trained by applying word2vec (Mikolov et al., 2013) (see Appendix A for hyper-parameters) to the 28M 'articles' of the MEDLINE/PubMed collection. IDF values were computed over the 18M articles that contained both titles and abstracts.

The 1,751 queries of years 1–4 were used for training, the first 100 queries of year 5 (batch 1) for development, and the remaining 400 queries of year 5 (batches 2–5) as test set. We set $N = 100$, since even using only the top 100 documents of BM25, the oracle scores are high. PubMed articles published after 2015 for the training set, and after 2016 for the development and test sets, were removed from the top N (and replaced by lower ranked documents up to N), as these were not available at the time of the human annotation.

Table 1 reports results on the BIOASQ test set, averaged over five runs as well as the single best run (by development MAP) with statistical significance. The enhanced models of this paper perform better than BM25 (even with extra features), PACRR, and DRMM. There is hardly any difference between PACRR and DRMM, but our combination of the two (PACRR-DRMM) surpasses them both on average, though the difference is statistically significant ($p < 0.05$) only when comparing to PACRR. Models that use context-sensitive term encodings (ABEL-DRMM, POSIT-DRMM) outperform other models, even PACRR-style models that incorporate context at later stages in the network. This is true both on average and by statistical significance over the best run. The best model on average is POSIT-DRMM+MV, though it is not significantly different than POSIT-DRMM.

4.3 TREC Robust 2004 Experiments

Our primary experiments were on the BIOASQ dataset as it has one of the largest sets of queries

---

[7] See, for example, Table 2 of Guo et al. (2016).

[8] For PACRR/PACRR-DRMM, we used/modified the code released by Hui et al. (2017, 2018). We use our own implementation of DRMM, which performs roughly the same as Guo et al. (2016), though the results are not directly comparable due to different random partitions of the data.

[9] We limit positive examples to be in the top N documents.

[10] `https://trec.nist.gov/trec_eval/` (v9.0)

[11] `http://bioasq.org/`.

[12] `https://www.ncbi.nlm.nih.gov/pubmed/`

[13] Our results are, thus, not comparable to those of participating systems, since experts did not consider our outputs.

[14] See `https://www.nlm.nih.gov/databases/download/pubmed_medline.html`.

AVERAGE OVER FIVE RUNS WITH STD. DEV.

System	MAP	P@20	nDCG@20
Traditional IR Baselines			
BM25	46.1 ± .0	25.5 ± .0	55.4 ± .0
BM25+extra	48.7 ± .0	26.6 ± .0	58.1 ± .0
Deep Learning Baselines			
PACRR	49.1 ± .2	27.1 ± .1	58.5 ± .2
DRMM	49.3 ± .2	27.2 ± .2	58.5 ± .3
Deep Learning with Enhanced Interactions			
PACRR-DRMM	49.9 ± .1	27.4 ± .1	59.3 ± .1
ABEL-DRMM	50.3 ± .2	27.5 ± .1	59.6 ± .2
+MV	50.4 ± .2	27.4 ± .2	59.7 ± .3
POSIT-DRMM	50.7 ± .2	27.8 ± .1	60.1 ± .2
+MV	51.0 ± .1	27.9 ± .1	60.3 ± .2
Oracle	72.8 ± .0	37.5 ± .0	80.7 ± .0

BEST RUN WITH STAT. SIG.

System	MAP	P@20	nDCG@20
Traditional IR Baselines			
BM25	46.1	25.5	55.4
BM25+extra	48.7 (1)	26.6 (1)	58.1 (1)
Deep Learning Baselines			
PACRR	49.1 (1)	27.0 (1-2)	58.6 (1)
DRMM	49.3 (1-2)	27.1 (1-2)	58.8 (1-2)
Deep Learning with Enhanced Interactions			
PACRR-DRMM	50.0 (1-3)	27.3 (1-2)	59.4 (1-3)
ABEL-DRMM	50.2 (1-4)	27.5 (1-4)	59.4 (1-3)
+MV	50.5 (1-4)	27.6 (1-4)	59.8 (1-4)
POSIT-DRMM	50.7 (1-4,6)	27.9 (1-7)	60.1 (1-4,6)
+MV	51.0 (1-7)	27.7 (1-4)	60.3 (1-7)
Oracle	72.8	37.5	80.7

Table 1: Performance on BIOASQ test data. Statistically significant ($p < 0.05$) difference from BM25^1; BM25+extra^2; PACRR^3; DRMM^4; PACRR-DRMM^5; ABEL-DRMM^6; ABEL-DRMM+MV^7.

AVERAGE OVER FIVE RUNS WITH STD. DEV.

System	MAP	P@20	nDCG@20
Traditional IR Baselines			
BM25	23.8 ± .0	35.4 ± .0	42.5 ± .0
BM25+extra	25.0 ± .0	36.7 ± .0	43.2 ± .0
Deep Learning Baselines			
PACRR	25.8 ± .2	37.2 ± .4	44.3 ± .4
DRMM	25.6 ± .6	37.0 ± .8	44.4 ± .6
Deep Learning with Enhanced Interactions			
PACRR-DRMM	25.9 ± .4	37.3 ± .7	44.4 ± .7
ABEL-DRMM	26.3 ± .4	38.0 ± .6	45.6 ± .4
+MV	26.5 ± .4	38.0 ± .5	45.5 ± .4
POSIT-DRMM	27.0 ± .4	38.3 ± .6	45.7 ± .5
+MV	27.2 ± .3	38.6 ± .6	46.1 ± .4
Oracle	68.0 ± .0	82.1 ± .0	93.1 ± .0

BEST RUN WITH STAT. SIG.

System	MAP	P@20	nDCG@20
Traditional IR Baselines			
BM25	23.8	35.4	42.5
BM25+extra	25.0 (1)	36.7 (1)	43.2 (1)
Deep Learning Baselines			
PACRR	25.8 (1-2)	37.4 (1-2)	44.5 (1-2)
DRMM	25.9 (1-2)	37.2 (1-2)	44.4 (1-2)
Deep Learning with Enhanced Interactions			
PACRR-DRMM	25.9 (1-2)	37.6 (1-2,4)	44.5 (1-2)
ABEL-DRMM	26.1 (1-5)	38.0 (1-5)	45.4 (1-5)
+MV	26.4 (1-6)	38.2 (1-5)	45.8 (1-6)
POSIT-DRMM	27.1 (1-7)	38.8 (1-7)	46.2 (1-7)
+MV	27.1 (1-7)	38.9 (1-7)	46.4 (1-7)
Oracle	68.0	82.1	93.1

Table 2: Performance on TREC ROBUST test data. Statistically significant ($p < 0.05$) difference from BM25^1; BM25+extra^2; PACRR^3; DRMM^4; PACRR-DRMM^5; ABEL-DRMM^6; ABEL-DRMM+MV^7.

(with manually constructed relevance judgments) and document collections, making it a particularly realistic dataset. However, in order to ground our models in past work we also ran experiments on TREC ROBUST 2004 (Voorhees, 2005), which is a common benchmark. It contains 250 queries[15] and 528K documents. As this dataset is quite small, we used a 5-fold cross-validation. In each fold, approx. $\frac{3}{5}$ of the queries were used for training, $\frac{1}{5}$ for development, and $\frac{1}{5}$ for testing. We applied word2vec to the 528K documents to obtain pre-trained embeddings. IDF values were computed over the same corpus. Here we used $N = 1000$, as the oracle scores for $N = 100$ were low.

Table 2 shows the TREC ROBUST results, which largely mirror those of BIOASQ. POSIT-DRMM+MV is still the best model, though again not significantly different than POSIT-DRMM. Furthermore, ABEL-DRMM and POSIT-DRMM are clearly better than the deep learning baselines,[16] but unlike BIOASQ, there is no statistically significant difference between PACRR-DRMM and the two deep learning baselines. Even though the scores are quite close (particularly MAP) both ABEL-DRMM and POSIT-DRMM are statistically different from PACRR-DRMM, which was not the case for BIOASQ. ABEL-DRMM+MV is significantly different than ABEL-DRMM on the best run for MAP and nDCG@20, unlike BIOASQ where there was no statistically significant difference between the two methods. However, on average over 5 runs, the systems show little difference.

5 Discussion

An interesting question is how well the deep models do without the extra features. For BIOASQ, the best model's (POSIT-DRMM+MV) MAP score drops from 48.1 to 46.2 on the development set, which is higher than the BM25 baseline (43.7), but on-par with BM25+EXTRA (46.0). We should

---

[15] We used the 'title' fields of the queries.

[16] The results we report for our implementation of DRMM are slightly different than those of Guo et al. (2016). There are a number of reasons for why this might be the case: there is no standard split of the data; non-standard preprocessing of the documents; the original DRMM paper reranks the top documents returned by Query Likelihood and not BM25.

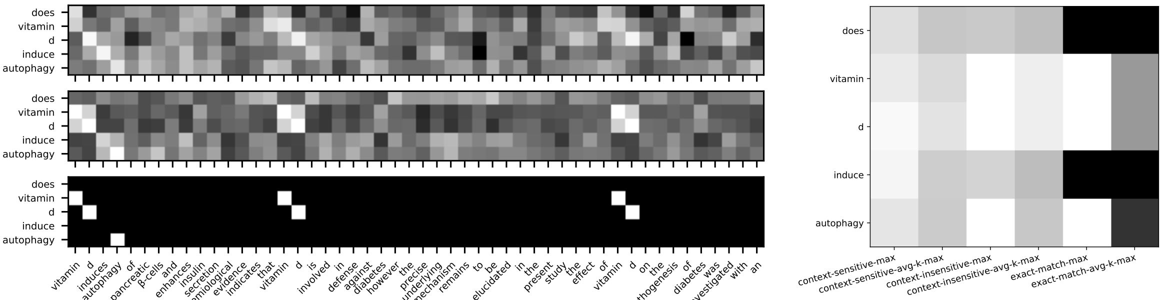

Figure 5: Left: Cosine similarities (POSIT-DRMM attention) of query and document terms, with context-sensitive, context-insensitive, and exact match views of the terms (top to bottom). Document truncated to 50 words. White is stronger. Right: Corresponding POSIT-DRMM+MV 6-dimensional document-aware query term encodings.

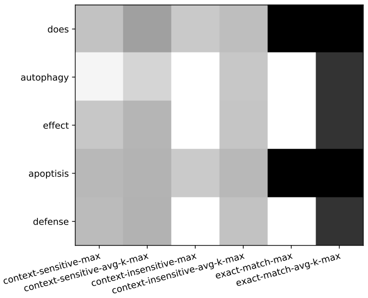

Figure 6: POSIT-DRMM+MV 6-dimensional document-aware q-term encodings for 'Does autophagy induce apoptosis defense?' and the same document as Fig. 5.

note, however, that on this set, the DRMM baseline without the extra features (which include BM25) is actually lower than BM25 (MAP 42.5), though it is obviously adding a useful signal, since DRMM with the extra features performs better (46.5).

We also tested the contribution of context-sensitive term encodings (Eq. 1). Without them, i.e., using directly the pre-trained embeddings, MAP on BIOASQ development data dropped from 47.6 to 46.3, and from 48.1 to 47.0 for ABEL-DRMM and POSIT-DRMM, respectively.

Fig. 5 shows the cosine similarities (attention scores, Eq. 5) between q-terms and d-terms, using term encodings of the three views (Fig. 4), for a query "Does Vitamin D induce autophagy?" and a relevant document from the BIOASQ development data. POSIT-DRMM indeed marks this as relevant. In the similarities of the context-insensitive view (middle left box) we see multiple matches around 'vitamin d' and 'induce autophagy'. The former is an exact match (white squares in lower left box) and the latter a soft match. The context-sensitive view (upper left box) smooths things out and one can see a straight diagonal white line matching 'vitamin d induce autophagy'. The right box of Fig. 5 shows the 6 components (Fig. 4) of the document-aware q-term encodings. Although some terms are not matched exactly, the context sensitive max and average pooled components (two left-most columns) are high for all q-terms. Interestingly, 'induce' and 'induces' are not an exact match (leading to black cells for 'induce' in the two right-most columns) and the corresponding context-insensitive component of (third cell from left) is low. However, the two components of the context-sensitive view (two left-most cells of 'induce') are high, esp. the max-pooling component (left-most).Finally, 'vitamin d' has multiple matches leading to a high average k-max pooled value, which indicates that the importance of that phrase in the document.

Fig. 6 shows the 6 components of the document-aware q-term encodings for another query and the same document, which is now irrelevant. In the max pooling columns of the exact match and context-insensitive view (columns 3, 5), the values look quite similar to those of Fig. 5. However, POSIT-DRMM scores this query-document pair low for two reasons. First, in the average-k-max pooling columns (columns 2, 4, 6) we get lower values than Fig. 5, indicating that there is less support for this pair in terms of density. Second, the context sensitive values (columns 1, 2) are much worse, indicating that even though many exact matches exist, in context, the meaning is not the same.

We conclude by noting there is still quite a large gap between the current best models and the oracle re-ranking scores. Thus, there is head room for improvements through more data or better models.

Acknowledgements

We thank the reviewers for their constructive feedback that greatly improved this work. Oscar Täckström gave thorough input on an early draft of this work. Finally, AUEB's NLP group provided many suggestions over the course of the work.

References

Leif Azzopardi, Wim Vanderbauwhede, and Hideo Joho. 2010. Search system requirements of patent analysts. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 775–776, Geneva, Switzerland.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Stefan Büttcher, Charles LA Clarke, and Brad Lushman. 2006. Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 621–622. ACM.

Paul Alexandru Chirita, Rita Gavriloaie, Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu. 2005. Activity based metadata for semantic desktop search. In *European Semantic Web Conference*, pages 439–454, Heraklion, Greece.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia.

Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64, Indianapolis, IN.

Donna K Harman. 2005. The TREC ad hoc experiments. Technical report, National Institute of Standards and Technology (NIST).

David Hawking. 2004. Challenges in enterprise search. In *Proceedings of the 15th Australasian Database Conference-Volume 27*, pages 15–24, Dunedin, New Zealand.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27*, pages 2042–2050.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, Copenhagen, Denmark.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 279–287, Marina Del Rey, CA.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142, Edmonton, Canada.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632.

Robert Krovetz. 1993. Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 191–202, Pittsburgh, PA.

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

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, pages 1367–1375, Lake Tahoe, NV.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Donald Metzler and W Bruce Croft. 2005. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 3111–3119, Lake Tahoe, Nevada.

Sunil Mohan, Nicolas Fiorini, Sun Kim, and Zhiyong Lu. 2017. Deep learning for biomedical information retrieval: Learning textual relevance from click logs. In *BioNLP 2017*, pages 222–231.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 257–266. ACM.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Stephen E. Robertson and Karen Sparck Jones. 1988. Document retrieval systems. chapter Relevance Weighting of Search Terms, pages 143–160. Taylor Graham Publishing, London, UK.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382, Santiago, Chile.

Mark D Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 623–632. ACM.

Karen Sparck Jones. 1988. Document retrieval systems. chapter A Statistical Interpretation of Term Specificity and Its Application in Retrieval, pages 132–142. Taylor Graham Publishing, London, UK.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to rank answers on large online QA collections. In *Proceedings of the 46th Annual Meeting for the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, pages 719–727, Columbus, OH.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. LSTM-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the BioASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):138.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008, Long Beach, CA.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28*, pages 2692–2700, Montreal, Canada.

Ellen M Voorhees. 2005. The TREC robust retrieval track. *ACM SIGIR Forum*, 39(1):11–20.

Xing Wei and W Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185, Seattle, WA.

Jian Wu, Kyle Mark Williams, Hung-Hsuan Chen, Madian Khabsa, Cornelia Caragea, Suppawong Tuarob, Alexander G Ororbia, Douglas Jordan, Prasenjit Mitra, and C Lee Giles. 2015. Citeseerx: AI in a digital library search engine. *AI Magazine*, 36(3):35–48.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–64. ACM.

Chengxiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342, New Orleans, LA.

Learning Neural Representation for CLIR with Adversarial Framework

Bo Li, Ping Cheng
School of Computer Science
Central China Normal University
Wuhan, China
`libo@mail.ccnu.edu.cn`

Abstract

The existing studies in cross-language information retrieval (CLIR) mostly rely on general text representation models (e.g., vector space model or latent semantic analysis). These models are not optimized for the target retrieval task. In this paper, we follow the success of neural representation in natural language processing (NLP) and develop a novel text representation model based on adversarial learning, which seeks a task-specific embedding space for CLIR. Adversarial learning is implemented as an interplay between the generator process and the discriminator process. In order to adapt adversarial learning to CLIR, we design three constraints to direct representation learning, which are (1) a matching constraint capturing essential characteristics of cross-language ranking, (2) a translation constraint bridging language gaps, and (3) an adversarial constraint forcing both language and source invariant to be reached more efficiently and effectively. Through the joint exploitation of these constraints in an adversarial manner, the underlying cross-language semantics relevant to retrieval tasks are better preserved in the embedding space. Standard CLIR experiments show that our model significantly outperforms state-of-the-art continuous space models and approaches the strong machine translation and monolingual baselines.

1 Introduction

Text representation is a crucial problem in most natural language processing (NLP) and information retrieval (IR) tasks. In monolingual IR, early research works mostly use vector space models for query-document semantic matching (Salton et al., 1975), which suffer from the problem of *synonymy* and *polysemy*. In order to bridge the lexical gaps, latent semantic models such as latent semantic analysis (LSA) (Deerwester et al., 1990) have been proposed to abstract away from surface text forms to approximate semantics. More recently, text representation learned with neural networks is attracting increasing attention of the IR community (Mitra and Craswell, 2017) and positive results have been reported on various evaluation data sets (Fan et al., 2018).

Compared to the prosperity in monolingual IR, there have been less advancements in CLIR where documents are written in a language different from that of queries. In addition to document ranking, CLIR models need to cross the language barriers, which makes the task intuitively more difficult than monolingual IR. Traditional approaches reduce CLIR to its monolingual counterpart via performing some way of translation on queries or/and documents. The typical translation process is performed with either off-the-shelf machine translation (MT) systems or multilingual dictionaries (Nie, 2010). However, MT based approaches are far from being a suitable solution for solving CLIR problems (refer to detailed analysis in (Zhou et al., 2012)). Dictionary-based approaches suffer from the same problem of lexical gaps as in the monolingual case (Gupta et al., 2017). An efficient cross-language representation is in need for CLIR, which is expected to be able to cross both the language and lexical gaps.

The most intuitive idea one can have so as to represent text in cross-language settings is to extend those models in monolingual environment. For instance, we note studies such as the extension of LSA in (Littman et al., 1998), the extension of principle component analysis (PCA) in (Platt et al., 2010), the extension of autoencoder model in (Chandar et al., 2014), and the extension of word2vec (Mikolov et al., 2013) in (Vulić and Moens, 2015). These approaches construct cross-language and semantic-rich representation of text, which can be applied to CLIR directly. However, all the models listed here aim to learn general text

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1861–1870
Brussels, Belgium, October 31 - November 4, 2018.

representation where the objective is to capture term proximity rather than relevance that is essential for retrieval task (Zamani and Croft, 2017). A recent work (Gupta et al., 2017) tries to learn task-specific representation for CLIR. However, their model only captures ranking signals in monolingual settings, which does not necessarily generalize well in CLIR.

In this paper, we propose to learn task-specific text representation for CLIR via a novel adversarial learning framework. Following the convention in generative adversarial networks (GAN) (Goodfellow et al., 2014), our representation learning model is realized as an interplay between two processes, an embedding generator (G) and an adversarial discriminator (D), conducted as a min-max game. With the GAN framework, we design three constraints to direct the representation learning process. CLIR is essentially a ranking problem and we develop a matching constraint to make sure that documents can be ranked in the right order given a query in another language. The matching constraint considers both cross-language and monolingual pairwise ranking signals, which is superior to previous studies (e.g., (Gupta et al., 2017)) only considering monolingual matching signals. Meanwhile, a translation constraint is imposed on the latent representation to bridge the language gaps. These two constraints direct the encoding networks to generate a language-invariant and task-specific representation in the embedding space. Lastly, an adversarial constraint is proposed to force both language and source invariant to be reached more efficiently and effectively. Through the joint exploitation of these constraints in an adversarial manner, the embedding space being optimal for CLIR will then result through the convergence of this process. Comprehensive CLIR experiments reveal that our model is superior to state-of-the-art continuous space models and approaches the machine translation and monolingual baselines.

2 Related work

Text representation has been a long-standing research question in IR. Classic methods such as vector space model are not able to deal with lexical gaps between queries and documents, resulting in inferior retrieval performance. Latent semantic approaches such as LSA (Deerwester et al., 1990) and latent dirichlet allocation (LDA) (Blei et al., 2003) abstract away from surface text forms to alleviate sparsity and approximate semantics. More recently, learning based approaches with neural networks have gained great success in NLP (Baroni et al., 2014) and started to attract increasing interests of the IR community. In terms of word level embedding, word2vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014) are two models that have been cited frequently in recent literature. These two models provide semantic-rich representations to bridge lexical gaps between queries and documents, which have been used broadly in neural IR studies (Ganguly et al., 2015; Zheng and Callan, 2015; Zamani and Croft, 2016).

The above studies deal with monolingual text representation, which are related to the cross-language models presented below. As for CLIR, typical approaches reduce CLIR to its monolingual counterparts via performing some way of translation. Machine translation systems such as Google translator[1] have been widely used to translate queries or documents, which serve as a default and convenient translation option in CLIR. It is however far from being a suitable solution for solving CLIR problems (a detailed analysis can be found in (Zhou et al., 2012)). An alternative solution is to rely on multilingual dictionaries to perform lexicon-level translation, which is mostly in combination with either language modeling strategy (Kraaij et al., 2003) or query structuring framework (Pirkola, 1998). However, dictionary-based methods still suffer from the lexical gap problem which reduces their performance in CLIR.

In fact, researchers have extended the models in monolingual settings and developed continuous space models for cross-language tasks to capture rich semantics. These cross-language extensions can be applied to CLIR directly. For instance, Littman et al. (1998) extend LSA to its cross-language version CL-LSA by concatenating document-term matrix of parallel data which acts as dual-language documents to be learned by LSA. Such a methodology leads to a dual-language semantic space in which terms from both languages are represented. Vinokourov et al. (2002) use parallel data to find most likely correlations between projected vectors based on canonical component analysis technique. The OPCA model (Platt et al., 2010) starts with the basic model

---

[1] https://translate.google.com

PCA that is then made discriminative by encouraging comparable document pairs to have similar vector representation. Compared to CL-LSA, OPCA avoids the use of artificial concatenated documents. More recently, neural models have been employed to learn cross-language representations. For instance, autoencoder is extended to a bilingual version BAE in (Chandar et al., 2014) which learns vectorial word representations from aligned sentences. Yih et al. (2011) develop S2Net to learn a projection matrix to map the corresponding term vectors into a latent space where similar documents are close. S2Net is implemented with Siamese neural network framework. Vulić and Moens (2015) first merge two documents from the aligned document pair in a comparable corpus and then train word2vec on the pseudo-bilingual document to obtain cross-language embeddings. The above approaches learn general text representation that captures term proximity rather than relevance which is important for retrieval task (Zamani and Croft, 2017). A recent work (Gupta et al., 2017) tries to learn task-specific embeddings for CLIR. However, it learns ranking signals by preserving pairwise ranking in monolingual settings prior to a transfer learning process to another language, which does not necessarily generalize well in CLIR.

One can find from above analysis that, most existing approaches, either based on neural networks or not, learn general embeddings irrelevant to CLIR. We argue that task-specific embeddings are superior, a fact that is inspired by monolingual IR studies and that will actually be validated by CLIR experiments in this paper. To this end, we will learn cross-language and task-specific embeddings for CLIR via a novel text representation model based on adversarial learning (Goodfellow et al., 2014).

3 Representation learning framework

We will present in this section a neural representation learning framework for CLIR. As discussed before, the framework is realized based on adversarial learning as an interplay between the generator process and the discriminator process. We will develop three constraints, namely a matching constraint, a translation constraint and an adversarial constraint, to direct the learning of cross-language and target-specific text embeddings. For ease of presentation, let us assume in CLIR we have a source language query q_s and a target language document d_t. The translation of q_s in the target language is q_t. The learning framework is illustrated in figure 1, which consists of an adversarial network NN_{adv}, three dimension adaptation networks NN_{dim} and three encoding networks respectively for q_t, d_t and q_s.

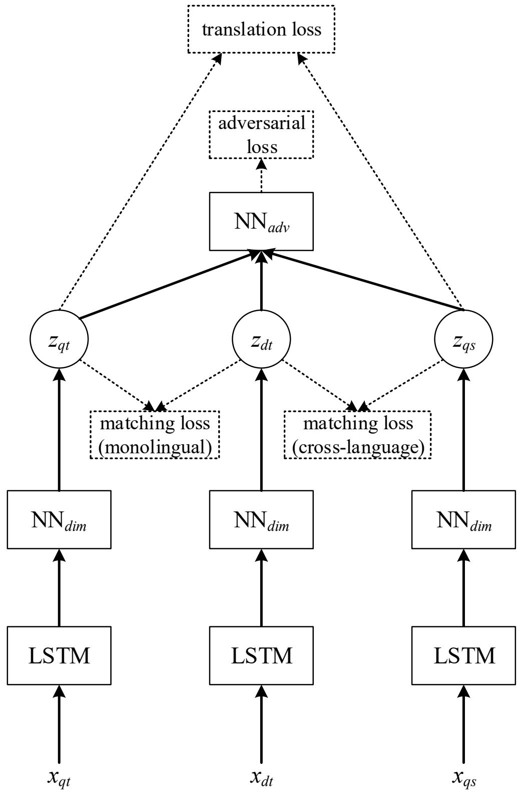

Figure 1: Text representation learning model for CLIR with adversarial framework.

3.1 Text representation networks

There have been various approaches one can use to encode sentences/documents into dense vectors. For instance, models based on convolutional neural networks (Kalchbrenner et al., 2014) and models based on recurrent neural networks (Liu et al., 2016) have been popular choices.

In order to map queries and documents into the embedding space, we make use of recurrent neural network with the long short-term memory (LSTM) architecture that can deal with vanishing and exploring gradient problems (Hochreiter and Schmidhuber, 1997). We present here derivation details of LSTM for clarification sake. The LSTM framework consists of several gates to control the cell state in the network. Firstly, a forget gate f (a sigmoid layer) functions according to:

$$f^\tau = \sigma(W_f \cdot [h^{\tau-1}, x^\tau] + b_f)$$

Then, an input gate i (a sigmoid layer) and a tanh layer work together as follows:

$$i^\tau = \sigma(W_i \cdot [h^{\tau-1}, x^\tau] + b_i)$$
$$\widetilde{C}^\tau = tanh(W_c \cdot [h^{\tau-1}, x^\tau] + b_c)$$

With the forget gate f, the input gate i and the new value $\widetilde{C}$, one can update the cell state C as:

$$C^\tau = f^\tau * C^{\tau-1} + i^\tau * \widetilde{C}^\tau$$

Lastly, an output gate o (a sigmoid layer) outputs:

$$o^\tau = \sigma(W_o \cdot [h^{\tau-1}, x^\tau] + b_o)$$
$$h^\tau = o^\tau * tanh(C^\tau)$$

In above equations, x^τ is the input at time step τ. h^τ and $h^{\tau-1}$ denote the hidden states at time steps τ and $\tau - 1$. All W and b are parameters. For brevity, we can write the update process as:

$$h^\tau = LSTM(h^{\tau-1}, x^\tau)$$

Given a text sequence $x = (x^1, x^2, \ldots, x^l)$, typical methods take the output h^l of LSTM at the last time step l as the concentrated representation of the whole sequence x (Sutskever et al., 2014). Since queries in IR tasks tend to be short and noisy, we make use of Bidirectional LSTM with pooling (Tan et al., 2015) to obtain a more effective text representation from all the hidden states $h^{1:l}$. The sequence x is fed from left to right into $LSTM_a$ and from right to left into $LSTM_b$. The new hidden state h^τ_{ab} at time step τ is obtained by concatenating the hidden states of $LSTM_a$ and $LSTM_b$ at their respective time step τ. Since max-pooling has been proven to be efficient in similar tasks (Tan et al., 2015), the latent representation z_x of x can be formulated as:

$$z_x = NN_{dim}(MaxPooling(h^{1:l}_{ab}))$$

where x can be q_s, q_t or d_t. NN_{dim} is designed to adapt the output dimension and to allow further flexibility for representation learning.

3.2 Matching constraint and Translation constraint

Document ranking is the central problem in both monolingual IR and CLIR tasks. CLIR differs itself from its monolingual counterpart in that the language gap needs to be crossed prior to the retrieval process. Since the choice of translation strategies (query, document or both) affects the design of other components in our model, we will discuss the translation constraint in section 3.2.1 prior to matching constraints in sections 3.2.2 and 3.2.3.

3.2.1 Translation constraint

The translation constraint is developed to minimize the differences between a pair of parallel texts, which serves as a basic requirement in the translation scenario. Such a constraint directs the learning of language-invariant text representation for CLIR. We follow the arguments in previous studies (Vilares et al., 2016) and choose to translate queries in our model, since it is computationally expensive to translate large-scale document collections in practice. In this paper, we directly employ Google translator to translate queries, which is a popular choice for machine translation that leads to state-of-the-art translation performance. The translation constraint is then imposed on the embedding vectors z_{q_s} and z_{q_t} of the queries q_s and q_t. The translation loss L_{tra} on a set QP of query pairs can be defined with the squared L2 norm, which is:

$$L_{tra} = \sum_{(q_s, q_t) \in QP} \|z_{q_s} - z_{q_t}\|_2^2$$

3.2.2 Cross-language matching constraint

The matching constraint captures essential characteristics of cross-language ranking. Following the practice in learning to rank (Liu, 2009), we model document ranking in the *pairwise* style where the relevance information is in the form of preferences between pairs of documents with respect to individual queries. In the model for CLIR, since we have matching signals from both monolingual text pairs and cross-language text pairs, the model can benefit from complementary knowledge from two resources. The monolingual pairwise matching constraint will be introduced in section 3.2.3.

Similar to neural models in monolingual settings (Huang et al., 2013), the cross-language pairwise matching constraint is placed on top of the embedding vectors of source language query and target language documents. In figure 1, let us assume x_{q_s} has a relevant document $x_{d_{t+}}$ and an irrelevant document $x_{d_{t-}}$ according to annotated text pairs. In training, the positive sample $x_{d_{t+}}$

for x_{q_s} can be chosen as the most relevant texts according to annotation, and the negative sample $x_{d_{t-}}$ is picked randomly from the data collection. The cross-language matching constraint encourages the hidden representation of $x_{d_{t+}}$ to be near to the hidden representation of x_{q_s} in the semantic-rich embedding space. Meanwhile, it asks the hidden representation of $x_{d_{t-}}$ to be far from that of x_{q_s}. We follow typical neural IR models and make use of cosine as the distance measure of hidden vectors. The probability that d_{t+} is ranked higher than d_{t-} given q_s can be derived as:

$$\hat{P}(q_s) = \sigma[\beta_c \cdot (\cos(z_{q_s}, z_{d_{t+}}) - \cos(z_{q_s}, z_{d_{t-}}))]$$

where σ is the sigmoid function with a hyper-parameter β_c controlling its shape. The cross-language matching loss L_{matc} on cross-language triplet set QD_c can be defined with cross-entropy loss as:

$$L_{matc} = \sum_{(q_s, d_{t+}, d_{t-}) \in QD_c} CE[P(q_s), \hat{P}(q_s)]$$

where CE denotes the cross-entropy operator between two distributions and $P(q_s)$ is the actual counterpart of $\hat{P}(q_s)$ estimated from annotation with a strategy similar to that in (Dehghani et al., 2017).

3.2.3 Monolingual matching constraint

The monolingual matching constraint L_{matm} can be built in a way similar to that of L_{matc}. L_{matm} is imposed on a set QD_m of monolingual triplet (q_t, d_{t+}, d_{t-}) as:

$$L_{matm} = \sum_{(q_t, d_{t+}, d_{t-}) \in QD_m} CE[P(q_t), \tilde{P}(q_t)]$$

where $P(q_t)$ is the actual counterpart of $\tilde{P}(q_t)$ estimated from annotation. $\tilde{P}(q_t)$ denotes the probability that d_{t+} is ranked higher than d_{t-} given q_t. It can be computed with the sigmoid function as:

$$\tilde{P}(q_t) = \sigma[\beta_m \cdot (\cos(z_{q_t}, z_{d_{t+}}) - \cos(z_{q_t}, z_{d_{t-}}))]$$

where β_m is a hyper-parameter.

3.2.4 Embedding generator constraint

Since our model is implemented with adversarial framework, we propose to model the representation generator G, which embodies the process of language-invariant and task-specific embedding of queries and documents into a latent subspace, under a combination of three constraints introduced above. The translation constraint aims to guarantee language invariant when translating queries. The cross-language matching constraint explicitly captures cross-language ranking signals from cross-language text pairs. The monolingual matching constraint takes monolingual ranking into account so as to complement the cross-language ranking signals.

Combing the three constraints above, we obtain a comprehensive constraint that should be obeyed by the embedding generator process. With the regularization term L_{reg} equaling to the sum of Frobenius norms of all weight matrices in the text embedding phase, we can write the embedding generator constraint L_G as:

$$L_G(\theta_G) = \gamma_1 \cdot L_{tra} + \gamma_2 \cdot L_{matc} + \gamma_3 \cdot L_{matm} + L_{reg}$$

where θ_G denotes the set of parameters in the generator networks, and γ_1, γ_2, γ_3 are hyper-parameters.

3.3 Adversarial constraint

We will introduce the adversarial constraint in this part. GAN (Goodfellow et al., 2014) simultaneously trains a generative model G and a disriminative model D in a competing way. G generates samples from a source of noise w that satisfies $w \sim P_n(w)$ and tries to capture the real data distribution P_r. D learns to distinguish between the generated samples from G and the true data sampled from P_r (in practice, from training data). The training procedure for G is to try its best to fool D. Let us assume that G generates samples satisfying the distribution P_g that is implicitly decided by $G(w)$. The GAN value function $V(G, D)$ on which D and G play the minmax game can be written as:

$$\begin{aligned} \min_G \max_D V(D, G) = & E_{x \sim P_r}[\log D(x)] \\ & + E_{x \sim P_g}[\log(1 - D(x))] \end{aligned} \quad (1)$$

Theoretical analysis has indicated that playing the minmax game as above amounts to minimizing the Jensen-Shannon divergence between P_g and P_r.

We follow the general idea of GAN and develop an adversarial component on top of the embedding space in figure 1. We note that GAN has been used in representation learning in a similar way as in (Bousmalis et al., 2016; Liu et al., 2017). In our model in figure 1, the adversarial component NN_{adv} acts as the discriminator D which tries its best to detect whether the embedding vector z is encoded from x_{q_t}, x_{d_t} or x_{q_s}. In this paper, NN_{adv} is implemented as a neural network with a softmax output layer. The output of NN_{adv} then corresponds to a probability distribution vector over the input sources. Let us denote the ground truth label of the current input z to NN_{adv} as l_z which indicates the source that z is encoded from. We can adjust equation 1 to our settings and obtain the adversarial loss L_{adv} on a query set Q_t and a document set D_t in the target language, as well as a query set Q_s in the source language, which can be written as:

$$L_{adv} = \min_{G} \max_{D} \sum_{x \in Q_t, D_t, Q_s} \log NN_{adv}(z_x) \circ l_{z_x}$$

where $\circ$ is the inner product operator.

3.4 Training procedure

Following the training convention of GAN (Goodfellow et al., 2014), the process of learning the language-invariant and task-specific text representation for CLIR should be conducted by jointly minimizing the generator constraint L_G and the adversarial loss L_{adv}, which leads us to the combined objective function L as:

$$L = L_G + L_{adv}$$

According to the rule of playing the minmax game in GAN, G tries its best to maximize the probability that D makes a mistake and D tries its best to distinguish between real data and generated data (in our case, various input sources). The theoretical requirement behind GAN that D is maintained near its optimal solution as long as G changes slowly enough motivates us to update the discriminator part k steps per update of the generator part in the iterative optimization process. Based on these discussions, the minmax optimization process can be derived as:

1. Optimize D when fixing G through: $\hat{\theta}_D = \arg\max_{\theta_D} L(\hat{\theta}_G, \theta_D)$
2. Optimize G when fixing D through: $\hat{\theta}_G = \arg\min_{\theta_G} L(\theta_G, \hat{\theta}_D)$

The optimization can be implemented with mini-batch gradient ascent (for θ_D) and descent (for θ_G).

4 Experiments and results

In this section, we conduct CLIR experiments so as to compare our text representation model with several other models.

4.1 Data sets

4.1.1 CLIR evaluation sets

To perform CLIR experiments, we rely on broadly used data sets released in the bilingual tasks of the cross-language evaluation forum (CLEF) [2]. We choose to use the data from the year 2000 to 2004. Table 1 lists the characteristics of the data set, which include number of documents (N_d), number of distinct words (N_w), the average document length (DL_{avg}) and the number of queries (N_q) in each task. We use source language queries in French (Fr), German (De) and Italian (It) to retrieve target language documents in English (En). Queries from year 2000 to 2002 are combined to a single task in table 1 since they have the same target set.

Table 1: CLIR dataset statistics (k = thousand).

Dataset	N_d	N_w	DL_{avg}	N_q
CLEF00-02	113k	173k	311	140
CLEF03	169k	233k	284	60
CLEF04	56k	120k	231	50

4.1.2 Training set

In order to train the representation learning model, we need to construct a data set consisting of annotated text pairs. We combine AOL queries (Pass et al., 2006) and a set of news titles downloaded from the news sites[3] to constitute training query set of diversity. Following the previous work (Gupta et al., 2017), we sample a balanced subset (1M) from such query set and use these queries to retrieve the data collection with BM25. For each training query, we take the top retrieved texts as positive samples, and the negative samples are selected randomly from the data collection. In addition to the pseudo-labeled text pairs of low quality, we combine the LETOR4.0 dataset (Qin and Liu, 2013) that is developed for evaluating learning to rank models. The LETOR4.0 dataset consists of relevance judgments of higher quality compared to

---

[2] http://www.clef-initiative.eu

[3] We fetch 2.8M web pages from several news websites such as ChinaDaily (www.chinadaily.com.cn) and XinhuaNews (www.xinhuanet.com).

pseudo-labeled data. The two data resources can complement each other in the training process.

In our experiments, the pseudo-labeled data is used to train the whole model and the LETOR dataset is employed to fine tune the parameters relevant to the source queries and target documents which are more important for the cross-language retrieval task.

4.2 Experimental settings

4.2.1 Experimental setup

The terms are initialized as the 512d word2vec vectors trained on Wikipedia dump corpus[4]. The term embeddings are fed into the LSTM model of which the hidden unit number is chosen from $\{64, 128, 256, 512\}$. The adversarial network NN_{adv} is as a three-layer feed-forward network with softmax on top of the last layer. NN_{dim} is implemented as a feed-forward network with layer dimension chosen from $\{32, 64, 128, 256\}$ and hidden layer number chosen from $\{1, 2\}$. The values of hyper-parameters γ_1, γ_2 and γ_3 are chosen from $\{0.01, 0.1, 1, 10, 100\}$. The learning rate is selected from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. Those hyper-parameters are tuned on the validation set which is 20% of the training queries randomly selected.

For evaluation, we present results in terms of mean average precision (MAP). Statistically significant differences between various models are determined using the paired t-test with $p < 0.05$.

4.2.2 Baseline approaches

We make use of three categories of baselines for CLIR experiments.

1. Monolingual run (MON): a baseline with target language queries that are strictly parallel to source language queries.

2. Machine translation (MT): a baseline with target-language queries translated by machine translation system from source-language queries.

3. Cross-language text representation models: baselines that rely on continuous space models for cross-language text representation. We make use here of S2Net (Yih et al., 2011), BAE (Chandar et al., 2014), and XCNN (Gupta et al., 2017) for the CLIR task.

4.3 Results and analysis

4.3.1 Comparisons to state-of-the-art

Table 2 lists the experimental results on CLEF dataset for our model (the column OURS) and all baseline models. There are three data collections and three language pairs, amounting to nine cross-language retrieval tasks. Except the strong baselines MON and MT, our model shows the best overall performance among all CLIR strategies. Indeed, our model outperforms all continuous space baselines (i.e., S2Net, BAE and XCNN) with statistical significance in almost all cases. Our model decreases slightly from the strong MT baseline in most retrieval tasks with only one degradation being significant on 03(De-En). Furthermore, one can find that our model approaches the monolingual baseline very much in all retrieval tasks with all MAP ratios around or over 90%. In our experiments, we have not performed comparisons to CL-LSA (Littman et al., 1998) and its variant OPCA (Platt et al., 2010), because they have been consistently outperformed by other CLIR strategies with a large margin (Schauble and Sheridan, 1997; Nie, 2010; Vulić et al., 2011).

Among all continuous space baselines, the most recent model XCNN shows the best performance. XCNN always outperforms linear projection methods S2Net with significance. It also significantly outperforms the non-linear model BAE in all cases. This is coincident with previous conclusions in (Gupta et al., 2017) due to the fact that XCNN learns target-specific representation for CLIR but the other models do not. Our model also tries to learn task-specific representation for CLIR, which significantly outperforms XCNN in most cases according to the results in table 2. The reasons might be that (1) our method is modeled in a more effective adversarial learning framework. (2) we explicitly capture cross-language ranking signals in embedding generator in addition to monolingual ranking signals used in XCNN. (3) our model can jointly capture the translation knowledge and document ranking knowledge in a unified framework.

4.3.2 Variant of our model

Our model can be customized easily by altering the constraints to direct the representation learning process. Since the specificity of our model comes from the adversarial learning framework that has never been investigated in CLIR, we re-

---

[4] https://dumps.wikimedia.org

Table 2: Retrieval performance (MAP scores) of all models on CLEF collections. $+(m/x)$ or $-(m/x)$ indicates that the improvements or degradations with respect to MT/XCNN are statistically significant. The highest value in each row (except the MON and MT baselines) is marked in bold. The percentages in the last column denote the MAP ratio of our model with respect to the MON baseline.

Data	Lang	MON	MT	S2Net	BAE	XCNN	OURS	PROP
00-02	Fr-En	0.469	0.431	0.330^{-m}_{-x}	0.369^{-m}_{-x}	0.401^{-m}	**0.424**$_{+x}$	90.4%
	De-En	0.469	0.447	0.341^{-m}_{-x}	0.381^{-m}_{-x}	0.420^{-m}	**0.435**$_{+x}$	92.8%
	It-En	0.469	0.439	0.339^{-m}_{-x}	0.374^{-m}_{-x}	0.409^{-m}	**0.426**$_{+x}$	90.8%
03	Fr-En	0.498	0.471	0.352^{-m}_{-x}	0.383^{-m}_{-x}	0.431^{-m}	**0.456**$_{+x}$	91.6%
	De-En	0.498	0.462	0.358^{-m}_{-x}	0.390^{-m}_{-x}	0.430^{-m}	**0.439**$^{-m}$	88.2%
	It-En	0.498	0.468	0.367^{-m}_{-x}	0.395^{-m}_{-x}	0.439^{-m}	**0.467**$_{+x}$	93.8%
04	Fr-En	0.517	0.483	0.378^{-m}_{-x}	0.402^{-m}_{-x}	0.442^{-m}	**0.470**$_{+x}$	90.9%
	De-En	0.517	0.482	0.382^{-m}_{-x}	0.419^{-m}_{-x}	0.447^{-m}	**0.473**$_{+x}$	91.5%
	It-En	0.517	0.477	0.385^{-m}_{-x}	0.411^{-m}_{-x}	0.458	**0.481**$_{+x}$	93.0%

move the constraint L_{adv} from the original model M and obtain the variant M_{adv}. In this case, M_{adv} can be optimized with standard mini-batch gradient descent approach, without playing the minmax game. We redo above CLIR experiments with the same settings as above and obtain the retrieval results of M_{adv} in table 3.

Table 3: Retrieval performance (MAP scores) of the variant M_{adv} on CLEF collections. $+$ or $-$ indicates that the improvements or degradations with respect to our original model M are statistically significant. The higher value in each row is marked in bold.

Data	Lang	M	M_{adv}
00-02	Fr-En	**0.424**	0.412^{-}
	De-En	**0.435**	0.418^{-}
	It-En	**0.426**	0.424
03	Fr-En	**0.456**	0.440^{-}
	De-En	**0.439**	0.435
	It-En	**0.467**	0.448^{-}
04	Fr-En	**0.470**	0.453^{-}
	De-En	**0.473**	0.465
	It-En	**0.481**	0.469

From the results one can find that when removing the adversarial component from the original model, M_{adv} decreases from the original model M in all retrieval tasks. The differences that are significant appear in 5 out of 9 retrieval tasks. The results demonstrate that learning generator and discriminator in a competing style within the adversarial learning framework leads to representation of higher quality, which eventually supports efficient CLIR. If we compare the variant M_{adv} with the XCNN model in table 2, we find that M_{adv} still performs better than XCNN in most cases. Such a comparison implicitly indicates that the joint exploitation of monolingual matching constraint, cross-language matching constraint and translation constraint in a single model is more efficient than using them separately as in the XCNN model.

5 Conclusions

In this paper, we propose a novel text representation approach for CLIR based on the adversarial learning framework. The learning framework is implemented as an interplay between an embedding generator process and an adversarial discriminator process, which leads to an optimal representation that is both language invariant and domain specific. The embedding generator is learned such that it explicitly considers both cross-language and monolingual pairwise ranking signals. In this way, it can ensure that the learned embeddings benefit from both sources and are directly optimized for CLIR. To the best of our knowledge, it is the first time adversarial learning has been applied to CLIR. Experiments on various language pairs in CLEF data collection show that our model is significantly better than other latent semantic models for CLIR. Indeed, our model approaches the performance of machine translation and monolingual baselines.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work was supported by the Fundamental Research Funds for Central Universities of CCNU (No. CCNU15A05062).

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL, pages 238–247.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS, pages 343–351.

A P Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS, pages 1853–1861.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 65–74.

Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 375–384.

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. 2015. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 795–798.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS, pages 2672–2680.

Parth Gupta, Rafael E. Banchs, and Paolo Rosso. 2017. Continuous space models for clir. *Information Processing & Management*, 53(2):359 – 370.

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

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM, pages 2333–2338.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL, pages 655–665.

Wessel Kraaij, Jian-Yun Nie, and Michel Simard. 2003. Embedding web-based statistical translation models in cross-language information retrieval. *Computational Linguistics*, 29(3):381–419.

Michael L. Littman, Susan T. Dumais, and Thomas K. Landauer. 1998. Automatic cross-language information retrieval using latent semantic indexing. In *Cross-Language Information Retrieval*, pages 51–62, Boston, MA. Springer.

Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016. Deep fusion lstms for text semantic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 1034–1043.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 1–10.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS, pages 3111–3119.

Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. *CoRR*, abs/1705.01509.

Jian-Yun Nie. 2010. Cross-language information retrieval. *Synthesis Lectures on Human Language Technologies*, 3(1):1–125.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems*, InfoScale.

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

Ari Pirkola. 1998. The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 55–63.

John C. Platt, Kristina Toutanova, and Wen-tau Yih. 2010. Translingual document representations from discriminative projections. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 251–261.

Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *CoRR*, abs/1306.2597.

G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Peter Schauble and Paraic Sheridan. 1997. Cross-language information retrieval (clir) track overview. In *Proceedings of the sixth Text REtrieval Conference*, TREC.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS, pages 3104–3112.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108.

Jesus Vilares, Manuel Vilares, Miguel A. Alonso, and Michael P. Oakes. 2016. On the feasibility of character n-grams pseudo-translation for cross-language information retrieval tasks. *Computer Speech and Language*, 36(C):136–164.

Alexei Vinokourov, John Shawe-Taylor, and Nello Cristianini. 2002. Inferring a semantic representation of text via cross-language correlation analysis. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS, pages 1497–1504.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2011. Cross-language information retrieval with latent topic models trained on a comparable corpus. In *Proceedings of the 7th Asia Conference on Information Retrieval Technology*, AIRS, pages 37–48.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 363–372.

Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL, pages 247–256.

Hamed Zamani and W. Bruce Croft. 2016. Embedding-based query language models. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR, pages 147–156.

Hamed Zamani and W. Bruce Croft. 2017. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 505–514.

Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 575–584.

Dong Zhou, Mark Truran, Tim Brailsford, Vincent Wade, and Helen Ashman. 2012. Translation techniques in cross-language information retrieval. *ACM Computing Surveys*, 45(1):1:1–1:44.

AD3: Attentive Deep Document Dater

Swayambhu Nath Ray
Indian Institute of Science
Bangalore, India
`swayambhunath93@gmail.com`

Shib Sankar Dasgupta
Indian Institute of Science
Bangalore, India
`s.s.dasgupta.iisc@gmail.com`

Partha Talukdar
Indian Institute of Science
Bangalore, India
`ppt@iisc.ac.in`

Abstract

Knowledge of the creation date of documents facilitates several tasks such as summarization, event extraction, temporally focused information extraction etc. Unfortunately, for most of the documents on the Web, the time-stamp metadata is either missing or can't be trusted. Thus, predicting creation time from document content itself is an important task. In this paper, we propose Attentive Deep Document Dater (AD3), an attention-based neural document dating system which utilizes both context and temporal information in documents in a flexible and principled manner. We perform extensive experimentation on multiple real-world datasets to demonstrate the effectiveness of AD3 over neural and non-neural baselines.

1 Introduction

Many natural language processing tasks require document creation time (DCT) information as a useful additional metadata. Tasks such as information retrieval (Li and Croft, 2003; Dakka et al., 2008), temporal scoping of events and facts (Allan et al., 1998; Talukdar et al., 2012b), document summarization (Wan, 2007) and analysis (de Jong et al., 2005a) require precise and validated creation time of the documents. Most of the documents obtained from the Web either contain DCT that cannot be trusted or contain no DCT information at all (Kanhabua and Nørvåg, 2008). Thus, predicting the time of these documents based on their content is an important task, often referred to as *Document Dating*.

A few generative approaches (de Jong et al., 2005b; Kanhabua and Nørvåg, 2008) as well as a discriminative model (Chambers, 2012) have been previously proposed for this task. Kotsakos et al. (2014) employs term-burstiness resulting in improved precision on this task.

Recently proposed NeuralDater (Vashishth et al., 2018) uses a graph convolution network (GCN) based approach for document dating, outperforming all previous models by a significant margin. NeuralDater extensively uses the syntactic and temporal graph structure present within the document itself. Motivated by NeuralDater, we explicitly develop two different methods: a) *Attentive Context Model*, and b) *Ordered Event Model*. The first component tries to accumulate knowledge across documents, whereas the latter uses the temporal structure of the document for predicting its DCT.

Motivated by the effectiveness of attention based models in different NLP tasks (Yang et al., 2016a; Bahdanau et al., 2014), we incorporate attention in our method in a principled fashion. We use attention not only to capture context but also for feature aggregation in the graph convolution network (Hamilton et al., 2017). Our contributions are as follows.

- We propose Attentive Deep Document Dater (AD3), the first attention-based neural model for time-stamping documents.
- We devise a novel method for label based attentive graph convolution over directed graphs and use it for the document dating task.
- Through extensive experiments on multiple real-world datasets, we demonstrate AD3's effectiveness over previously proposed methods.

AD3 source code and datasets used in the paper are available at https://github.com/malllabiisc/AD3

2 Related Work

Document Time-Stamping: Initial attempts

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1871–1880
Brussels, Belgium, October 31 - November 4, 2018.

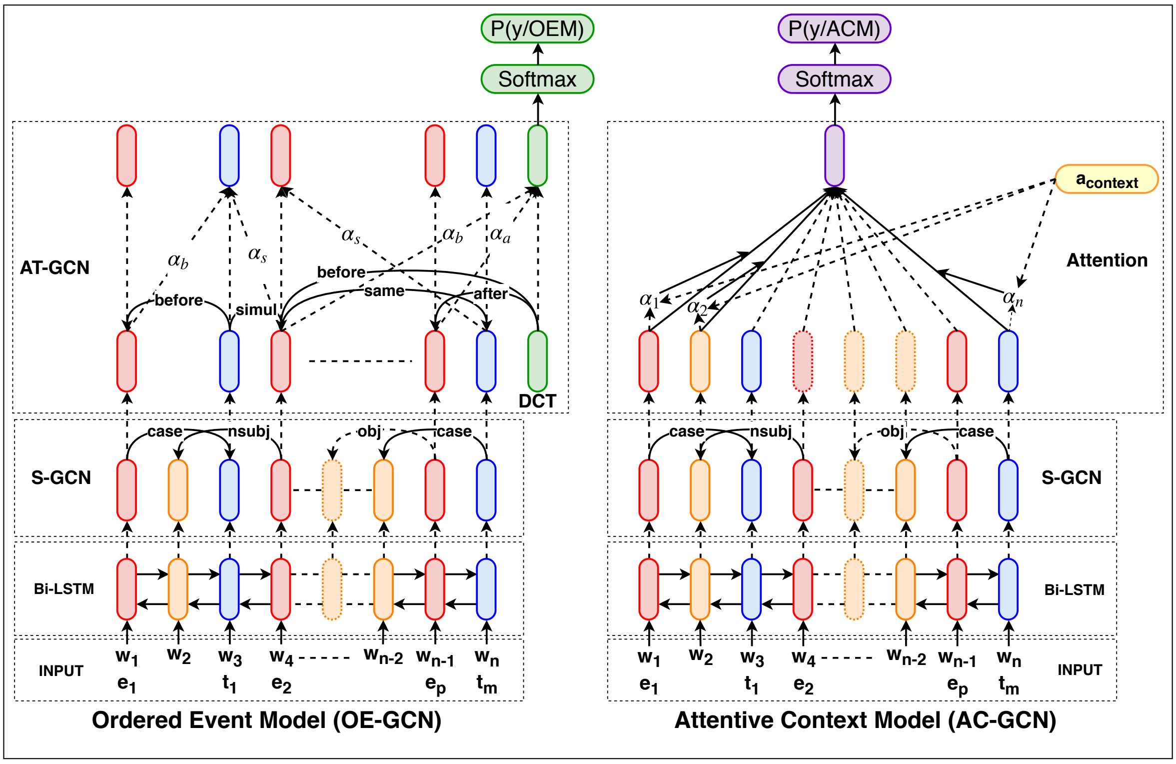

Figure 1: Two proposed models a) Ordered Event Model (left) and b) Attentive Context Model (right), where w_i are the words of a document (D), e_i are the words signifying events and t_i are the temporal tokens as detected in the document. Both models use Bi-LSTM and S-GCN (Syntactic-GCN, see Section 3.2.2) in the initial part of their pipeline. Ordered Event Model (OE-GCN) uses a label based attentive graph convolutional network for encoding the DCT, whereas Attentive Context Model (AC-GCN) uses a word attention based model to encode the document. $\alpha_i(\forall\, i \in [1, n])$ denotes attention over the words of document and α_a, α_b and α_s denote attention over nodes connected with edge labels *AFTER*, *BEFORE* and *SIMULTANEOUS*, respectively. OE-GCN provides the probability scores over the years given the encoded DCT, while AC-GCN provides the probability scores given the context of the document. Both the models are trained separately.

made for document time-stamping task include statistical language models proposed by de Jong et al. (2005b) and Kanhabua and Nørvåg (2008). (Chambers, 2012) use temporal and hand-crafted features extracted from documents to predict DCT. They propose two models, one of which learns the probabilistic constraints between year mentions and the actual creation time, whereas the other one is a discriminative model trained on hand-crafted features. Kotsakos et al. (2014) propose a term-burstiness (Lappas et al., 2009) based statistical method for the task. Vashishth et al. (2018) propose a deep learning based model which exploits the temporal and syntactic structure in documents using graph convolutional networks (GCN).

Event Ordering System: The task of extracting temporally rich events and time expressions and ordering between them is introduced in the TempEval challenge (UzZaman et al., 2013; Verhagen et al., 2010). Various approaches (McDowell et al., 2017; Mirza and Tonelli, 2016) made for solving the task use sieve-based architectures, where multiple classifiers are ranked according to their precision and their predictions are weighted accordingly resulting in a temporal graph structure. A method to extract temporal ordering among relational facts was proposed in (Talukdar et al., 2012a).

Graph Convolutional Network (GCN): GCN (Kipf and Welling, 2016) is the extension of convolutional networks over graphs. In different NLP tasks such as semantic-role labeling (Marcheggiani and Titov, 2017), neural machine translation (Bastings et al., 2017), and event detection (Nguyen and Grishman, 2018), GCNs have proved to be effective. We extensively use GCN for capturing both syntactic and temporal aspect of the document.

Attention Network: Attention networks have been well exploited for various tasks such as document classification (Yang et al., 2016b), question answering (Yang et al., 2016a), machine translation (Bahdanau et al., 2014; Vaswani et al., 2017). Recently, attention over graph structure has been shown to work well by Veličković et al. (2018). Taking motivation from them, we deploy an attentive convolutional network on temporal graph for the document dating problem.

3 Background: GCN & NeuralDater

The task of document dating can be modeled as a multi-class classification problem. Following prior work, we shall focus on DCT prediction at the year-granularity in this paper. In this section, we summarize the previous state-of-the-art model NeuralDater (Vashishth et al., 2018), before moving onto our method. An overview of graph convolutional network (GCN) (Kipf and Welling, 2016) is also necessary as it is used in NeuralDater as well as in our model.

3.1 Graph Convolutional Network

GCN for Undirected Graph: Consider an undirected graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the set of n vertices and set of edges respectively. Matrix $\mathcal{X} \in \mathbb{R}^{n \times m}$, whose rows are input representation of node u, where $x_u \in \mathbb{R}^m$, $\forall\ u \in \mathcal{V}$, is the input feature matrix. The output hidden representation $h_v \in \mathbb{R}^d$ of a node v after a single layer of graph convolution operation can be obtained by considering only the immediate neighbours of v, as formulated in (Kipf and Welling, 2016). In order to capture information at multi-hop distance, one can stack layers of GCN, one over another.

GCN for Directed Graph: Consider a labelled edge from node u to v with label $l(u, v)$, denoted collectively as $(u, v, l(u, v))$. Based on the assumption that information in a directed edge need not only propagate along its direction, Marcheggiani and Titov (2017) added opposite edges viz., for each $(u, v, l(u, v))$, $(v, u, l(u, v)^{-1})$ is added to the edge list. Self loops are also added for passing the current embedding information. When GCN is applied over this modified directed graph, the embedding of the node v after k^{th} layer will be,

$$h_v^{k+1} = f\left(\sum_{u \in \mathcal{N}(v)} \left(W^k_{l(u,v)} h^k_u + b^k_{l(u,v)}\right)\right).$$

We note that the parameters $W^k_{l(u,v)}$ and $b^k_{l(u,v)}$ in this case are edge label specific. h^k_u is the input to the k^{th} layer. Here, $\mathcal{N}(v)$ refers to the set of neighbours of v, according to the updated edge list and f is any non-linear activation function (e.g., ReLU: $f(x) = \max(0, x)$).

3.2 NeuralDater

In this sub-section, we provide a brief overview of the components of the NeuralDater (Vashishth et al., 2018). Given a document D with n tokens $w_1, w_2, \cdots w_n$, NeuralDater extracts a temporally rich embedding of the document in a principled way as explained below:

3.2.1 Context Embedding

Bi-directional LSTM is employed for embedding each word with its context. The GloVe representation of the words $X \in \mathbb{R}^{n \times k}$ is transformed to a context aware representation $H^{cntx} \in \mathbb{R}^{n \times k}$ to get the context embedding. This is essentially shown as the Bi-LSTM in Figure 1.

3.2.2 Syntactic Embedding

In this step, the context embeddings are further processed using GCN over the dependency parse tree of the sentences in the document, in order to capture long range connection among words. The syntactic dependency structure is extracted by Stanford CoreNLP's dependency parser (Manning et al., 2014). NeuralDater follows the same formulation of GCN for directed graph as described in Section 3.1, where additional edges are added to the graph to model the information flow. Again following (Marcheggiani and Titov, 2017), NeuralDater does not allocate separate weight matrices for different types of dependency edge labels, rather it considers only three type of edges: a) edges that exist originally, b) the reverse edges that are added explicitly, and c) self loops. The S-GCN portion of Figure 1 represents this component.

More formally, $H^{cntx} \in \mathbb{R}^{n \times k}$ is transformed to $H^{syn} \in \mathbb{R}^{n \times k_{syn}}$ by applying S-GCN.

3.2.3 Temporal Embedding

In this layer, NeuralDater exploits the Event-Time graph structure present in the document. CATENA (Mirza and Tonelli, 2016), current state-of-the-art temporal and causal relation extraction algorithm, produces the temporal graph from the event time annotation of the document. GCN applied over this Event-Time graph, namely T-GCN, chooses n_T number of tokens out of total n tokens from the document for further revision in their embeddings. Note that T is the total number of events and time mentions present in the document. A special node DCT is added to the graph and its embedding is jointly learned. Note that this layer learns both label and direction specific parameters.

3.2.4 Classifier

Finally, the DCT embedding concatenated with the average pooled syntactic embedding is fed to a softmax layer for classification. This whole procedure is trained jointly.

4 Attentive Deep Document Dater (AD3): Proposed Method

In this section, we describe Attentive Deep Document Dater (AD3), our proposed method. AD3 is inspired by NeuralDater, and shares many of its components. Just like in NeuralDater, AD3 also leverages two main types of signals from the document – syntactic and event-time – to predict the document's timestamp. However, there are crucial differences between the two systems. Firstly, instead of concatenating embeddings learned from these two sources as in NeuralDater, AD3 treats these two models completely separate and combines them at a later stage. Secondly, unlike NeuralDater, AD3 employs attention mechanisms in each of these two models. We call the resulting models Attentive Context Model (AC-GCN) and Ordered Event Model (OE-GCN). These two models are described in Section 4.1 and Section 4.2, respectively.

4.1 Attentive Context Model (AC-GCN)

Recent success of attention-based deep learning models for classification (Yang et al., 2016b), question answering (Yang et al., 2016a), and machine translation (Bahdanau et al., 2014) have motivated us to use attention during document dating. We extend the syntactic embedding model of NeuralDater (Section 3.2.2) by incorporating an attentive pooling layer. We call the resulting model AC-GCN. This model (right side in Figure 1) has two major components.

- **Context Embedding and Syntactic Embedding**: Following NeuralDater, we used Bi-LSTM and S-GCN to capture context and long-range syntactic dependencies in the document (Please refer to Section 3.2.1, Section 3.2.2 for brief description). The syntactic embedding, $H^{syn} \in \mathbb{R}^{n \times k_{syn}}$ is then fed to an Attention Network for further processing. Note that, k_{syn} is the dimension of the output of Syntactic-GCN and n is the number of tokens in the document.

- **Attentive Embedding**: In this layer, we learn the representation for the whole document through word level attention network. We learn a context vector, $u_s \in \mathbb{R}^s$ with respect to which we calculate attention for each token. Finally, we aggregate the token features with respect to their attention weights in order to represent the document. More formally, let $h_t^{syn} \in \mathbb{R}^{k_{syn}}$ be the syntactic representation of the t^{th} token in the document. We take non-linear projection of it in $\mathbb{R}^s$ with $W_s \in \mathbb{R}^{s \times k_{syn}}$. Attention weight α_t for t^{th} token is calculated with respect to the context vector u_t^T as follows.

$$u_t = \tanh(W_s h_t^{syn}),$$

$$\alpha_t = \frac{\exp(u_t^T u_s)}{\sum_t \exp(u_t^T u_s)}.$$

  Finally, the document representation for the AC-GCN is computed as shown below.

$$d_{\text{AC-GCN}} = \sum_t \alpha_t h_t^{syn}$$

  This representation is fed to a softmax layer for the final classification.

The final probability distribution over years predicted by the AC-GCN is given below.

$$\text{P}_{\text{AC-GCN}}(y|D) = \text{Softmax}(W \cdot d_{\text{AC-GCN}} + b).$$

4.2 Ordered Event Model (OE-GCN)

The OE-GCN model is shown on the left side of Figure 1. Just like in AC-GCN, context and syntactic embedding is also part of OE-GCN. The syntactic embedding is fed to the Attentive Graph Convolution Network (AT-GCN) where the graph is obtained from the time-event ordering algorithm CATENA (Mirza and Tonelli, 2016). We describe these components in detail below.

4.2.1 Temporal Graph

We use the same process used in NeuralDater (Vashishth et al., 2018) for procuring the Temporal Graph from the document. CATENA (Mirza and Tonelli, 2016) generates 9 different temporal links between events and time expressions present in the document. Following Vashishth et al. (2018), we choose 5 most frequent ones - AFTER, BEFORE, SIMULTANEOUS, INCLUDES, and IS INCLUDED – as labels. The temporal graph is constructed from the partial ordering between event verbs and time expressions.

Let $\mathcal{E}_T$ be the edge list of the Temporal Graph. Similar to (Marcheggiani and Titov, 2017; Vashishth et al., 2018), we also add reverse edges for each of the existing edge and self loops for passing current node information as explained in Section 3.1. The new edge list $\mathcal{E}_T^{'}$ is shown below.

$$\mathcal{E}_{\mathrm{T}}' = \mathcal{E}_{\mathrm{T}} \cup \{(j, i, l(i,j)^{-1}) \mid (i, j, l(i,j)) \in \mathcal{E}_{\mathrm{T}}\} \\ \cup \{(i, i, \text{self}) \mid i \in \mathcal{V})\}.$$

The reverse edges are added with reverse labels like AFTER^{-1}, BEFORE^{-1} etc . Finally, we get 10 labels for our temporal graph and we denote the set of edge labels by $\mathcal{L}$.

4.2.2 Attentive Graph Convolution (AT-GCN)

Since the temporal graph is automatically generated, it is likely to have incorrect edges. Ideally, we would like to minimize the influence of such noisy edges while computing temporal embedding. In order to suppress the noisy edges in the Temporal Graph and detect important edges for reasoning, we use attentive graph convolution (Hamilton et al., 2017) over the Event-Time graph. The attention mechanism learns the aggregation function jointly during training. Here, the main objective is to calculate the attention over the neighbouring nodes with respect to the current node for a given label. Then the embedding of the current node is updated by mixing neighbouring node embedding according to their attention scores. In this respect, we propose a label-specific attentive graph convolution over directed graphs.

Let us consider an edge in the temporal graph from node i to node j with type l, where $l \in \mathcal{L}$ and $\mathcal{L}$ is the label set. The label set $\mathcal{L}$ can be divided broadly into two coarse labels as done in Section 3.2.2. The attention weights are specific to only these two type of edges to reduce parameter and prevent overfitting. For illustration, if there exists an edge from node i to j then the edge types will be,

- $L(i,j) = \rightarrow$ if $(i,j,l(i,j)) \in \mathcal{E}_T^{'}$,
  i.e., if the edge is an original event-time edge.
- $L(i,j) = \leftarrow$ if $(i,j,l(i,j)^{-1}) \in \mathcal{E}_T^{'}$,
  i.e., if the edge is added later.

First, we take a linear projection ($W^{atten}_{L(i,j)} \in \mathbb{R}^{F \times k_{syn}}$) of both the nodes in $\mathbb{R}^F$ in order to map both of them in the same direction-specific space. The concatenated vector $[W^{atten}_{L(i,j)} \times h_i; W^{atten}_{L(i,j)} \times h_j]$, signifies the importance of the node j w.r.t. node i. A non linear transformation of this concatenation can be treated as the importance feature vector between i and j.

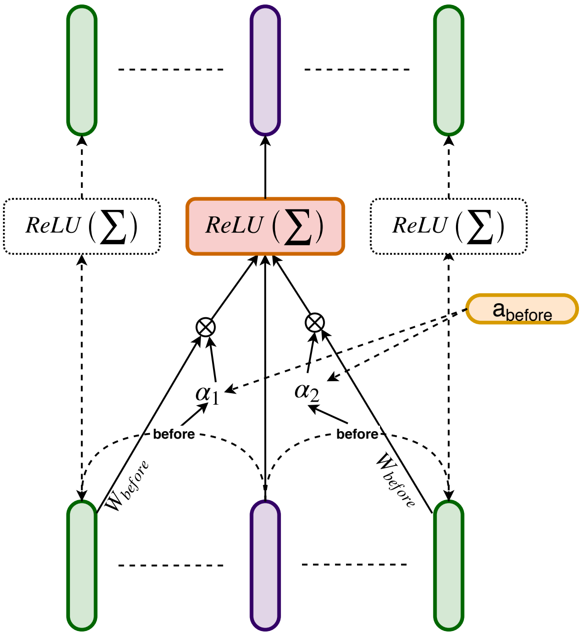

Figure 2: Attentive Graph Convolution (AT-GCN). In this layer, we learn attention weights for every edge based on label and direction. The attention weights are learnt using a context vector. The final representation of every node is a summation of weighted convolution over neighboring nodes based on labels.

$$e_{ij} = \tanh[W^{atten}_{L(i,j)} \times h_i; W^{atten}_{L(i,j)} \times h_j].$$

Now, we compute the attention weight of node j for node i with respect to a direction-specific context vector $a_{L(i,j)} \in \mathbb{R}^{2F}$, as follows.

$$\alpha_{ij}^{l(i,j)} = \frac{\exp\left(a^T_{L(i,j)} e_{ij}\right)}{\sum\limits_{k \in \mathcal{N}_i^{l(i,\cdot)}} \exp\left(a^T_{L(i,j)} e_{ik}\right)},$$

where, $\alpha_{ij}^{l(i,j)} = 0$ if node i and j is not connected through label l. $\mathcal{N}^{l(i,\cdot)}$ denotes the subset of the neighbourhood of node i with label l only. Please note that, although the linear transform weight ($W^{atten}_{L(i,j)} \in \mathbb{R}^{F \times k_{syn}}$) is specific to the coarse labels L, but for each finer label $l \in \mathcal{L}$ we get these convex weights of attentions. Figure 2 illustrates the above description w.r.t. edge type BEFORE.

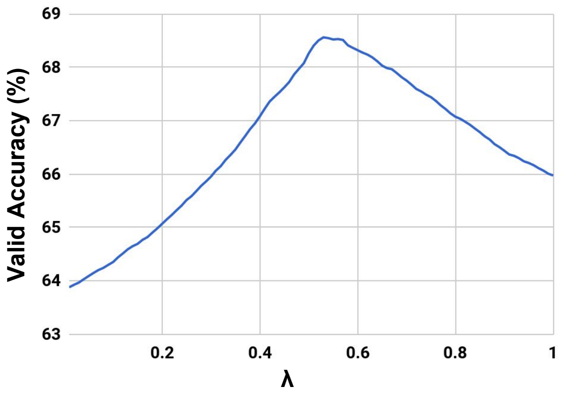

Figure 3: Variation of validation accuracy with λ (for APW dataset). We observe that AC-GCN and OE-GCN are both important for the task as we get optimal λ = 0.52.

Finally, the feature aggregation is done according to the attention weights. Prior to that, another label specific linear transformation is taken to perform the convolution operation. Then, the updated feature for node i is calculated as follows.

$$h_i^{k+1} = f\left(\sum_{l\in\mathcal{L}} \sum_{j\in\mathcal{N}_i^{l(i,\cdot)}} \alpha_{ij}^{l(i,j)} \left(W_{l(i,j)} h_j + b_{l(i,j)}\right)\right).$$

where, $\alpha_{ii} = 1$, $\mathcal{N}^{l(i,\cdot)}$ denotes the subset of the neighbourhood of node i with label l only. Note that, $\alpha_{ij}^{l(i,j)} = 0$ when $j \notin \mathcal{N}^{l(i,\cdot)}$. To illustrate formally, from Figure 2, we see that weight α_1 and α_2 is calculated specific to label type BEFORE and the neighbours which are connected through BEFORE is being multiplied with W_{before} prior to aggregation in the $ReLU$ block.

Now, after applying attentive graph convolution network, we only consider the representation of Document Creation Time (DCT), h_{DCT}, as the document representation itself. h_{DCT} is now passed through a fully connected layer prior to softmax. Prediction of the OE-GCN for the document D will be given as

$$\mathrm{P}_{\text{OE}-\text{GCN}}(y|D) = \text{Softmax}(W \cdot d_{\text{DCT}} + b).$$

4.3 AD3: Attentive Deep Document Dater

In this section, we propose an unified model by mixing both AC-GCN and OE-GCN. Even on validation data, we see that performance of both the models differ to a large extent. This significant difference (McNemar test p $<$ 0.000001) motivated the unification. We take convex combination of the output probabilities of the two models as shown below.

$$\mathrm{P}_{joint}(y|D) = \lambda \mathrm{P}_{\text{AC}-\text{GCN}}(y|D) + (1-\lambda)\mathrm{P}_{\text{OE}-\text{GCN}}(y|D).$$

Datasets	# Docs	Start Year	End Year
APW	675k	1995	2010
NYT	647k	1987	1996

Table 1: Details of datasets used. Please refer Section 5 for details.

The combination hyper-parameter λ is tuned on the validation data. We obtain the value of λ to be 0.52 (Figure 3) and 0.54 for APW and NYT datasets, respectively. This depicts that the two models are capturing significantly different aspects of documents, resulting in a substantial improvement in performance when combined.

5 Experimental Setup

Dataset: Experiments are carried out on the Associated Press Worldstream (APW) and New York Times (NYT) sections of the Gigaword corpus (Parker et al., 2011). We have used the same 8:1:1 split as Vashishth et al. (2018) for all the models. For quantitative details please refer to Table 1.

Evaluation Criteria: In accordance with prior work (Chambers, 2012; Kotsakos et al., 2014; Vashishth et al., 2018) the final task is to predict the publication year of the document. We give a brief description of the baselines below.

Baseline Methods:

- **MaxEnt-Joint** (Chambers, 2012): This method engineers several hand-crafted temporally influenced features to classify the document using MaxEnt Classifier.

- **BurstySimDater** (Kotsakos et al., 2014): This is a purely statistical method which uses lexical similarity and term burstiness (Lappas et al., 2009) for dating documents in arbitrary length time frame. For our experiments, we used a time frame length of 1 year.

- **NeuralDater** (Vashishth et al., 2018): This is the first deep neural network based approach for the document dating task. Details are provided in Section 3.2.

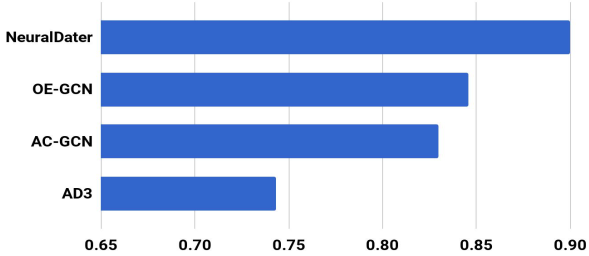

Figure 4: Mean absolute deviation (in years; lower is better) between a model's top prediction and the true year in the APW dataset. We find that all of our proposed methods outperform the previous state-of-the-art NeuralDater. Please refer to Section 6.1 for details.

Method	APW	NYT
BurstySimDater	45.9	38.5
MaxEnt-Joint	52.5	42.5
NeuralDater	64.1	58.9
Attentive NeuralDater [6.2]	66.2	60.1
OE-GCN [4.2]	63.9	58.3
AC-GCN [4.1]	65.6	60.3
AD3 [4.3]	**68.2**	**62.2**

Table 2: Accuracy (%) of different methods on the APW and NYT datasets for the document dating problem (higher is better). The unified model significantly outperforms all previous models.

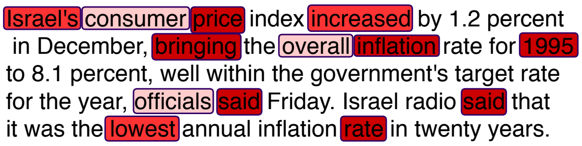
Israel's consumer price index increased by 1.2 percent
in December, bringing the overall inflation rate for 1995
to 8.1 percent, well within the government's target rate
for the year, officials said Friday. Israel radio said that
it was the lowest annual inflation rate in twenty years.

Figure 5: Visualization of the attention of AC-GCN. AC-GCN captures the intuitive tokens as seen in the figure. Darker shade implies higher attention. The correct DCT is 1996.

Method	Accuracy
T-GCN of NeuralDater	61.8
OE-GCN	**63.9**
S-GCN of NeuralDater	63.2
AC-GCN	**65.6**

Table 3: Accuracy (%) comparisons of component models with and without Attention. This results show the effectiveness of both word attention and Graph Attention for this task. Please see Section 6.2 for more details.

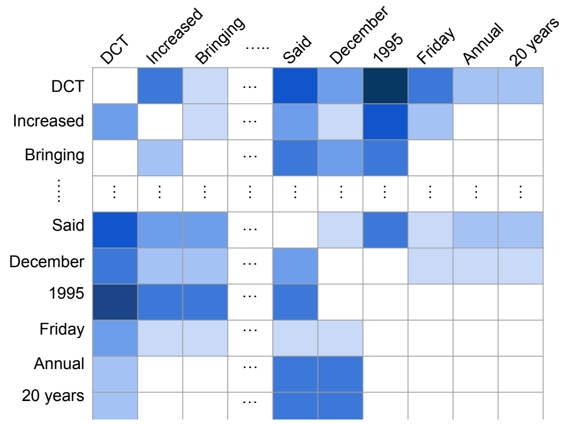

Figure 6: Visualization of the average edge attention of the temporal graph as learnt by OE-GCN for the document shown in Figure 5. Darker color implies higher attention. The correct DCT is 1996.

Hyperparameters: We use 300-dimensional GloVe embeddings and 128-dimensional hidden state for both GCNs and BiLSTM with 0.8 dropout. We use Adam (Kingma and Ba, 2014) with 0.001 learning rate for training. For OE-GCN we use 2-layers of AT-GCN. 1-layer of S-GCN is used for both the models.

6 Results

6.1 Performance Analysis

In this section, we compare the effectiveness of our method with that of prior work. The deep network based NeuralDater model in (Vashishth et al., 2018) outperforms previous feature engineered (Chambers, 2012) and statistical methods (Kotsakos et al., 2014) by a large margin. We observe a similar trend in our case. Compared to the state-of-the-art model NeuralDater, we gain, on an average, a 3.7% boost in accuracy on both the datasets (Table 2).

Among individual models, OE-GCN performs at par with NeuralDater, while AC-GCN outperforms it. The empirical results imply that AC-GCN by itself is effective for this task. The relatively worse performance of OE-GCN can be attributed to the fact that it only focuses on the Event-Time information and leaves out most of the contextual information. However, it captures various different ($p < 0.000001$, McNemar's test, 2-tailed) aspects of the document for classification, which motivated us to propose an ensemble of the two models. This explains the significant boost in performance of AD3 over NeuralDater as well as the individual models. It is worth mentioning that although AC-GCN and OE-GCN do not provide significant boosts in accuracy, their predictions have considerably lower mean-absolute-deviation as shown in Figure 4.

We concatenated the DCT embedding provided by OE-GCN with the document embedding provided by AC-GCN and trained in an end to end joint fashion like NeuralDater. We see that even with a similar training method, the Attentive NeuralDater model on an average, performs 1.6% better in terms of accuracy, once again proving the efficacy of attention based models over normal models.

6.2 Effectiveness of Attention

Attentive Graph Convolution (Section 4.2.2) proves to be effective for OE-GCN, giving a 2% accuracy improvement over non-attentive T-GCN of NeuralDater (Table 3). Similarly the efficacy of word level attention is also prominent from Table 3.

We have also analyzed our models by visualizing attentions over words and attention over graph nodes. Figure 5 shows that AC-GCN focuses on temporally informative words such as ”said” (for tense) or time mentions like “1995”, alongside important contextual words like “inflation”, “Israel” etc. For OE-GCN, from Figure 6 we observe that “DCT” and time-mention ‘1995’ grabs the highest attention. Attention between “DCT” and other event verbs indicating past tense are quite prominent, which helps the model to infer 1996 (which is correct) as the most likely time-stamp of the document. These analyses provide us with a good justification for the performance of our attentive models.

7 Discussion

Apart from empirical improvements over previous models, we also perform a qualitative analysis of the individual models. Figure 7 shows that the performance of AC-GCN improves with the length of documents, thus indicating that richer context leads to better model prediction. Figure 8 shows how the performance of OE-GCN improves with the number of event-time mentions in the document, thus further reinforcing our claim that more temporal information improves model performance.

Vashishth et al. (2018) reported that their model got confused by the presence of multiple misleading time mentions. AD3 overcomes this limitation using attentive graph convolution, which successfully filters out noisy time mentions as is evident from Figure 8.

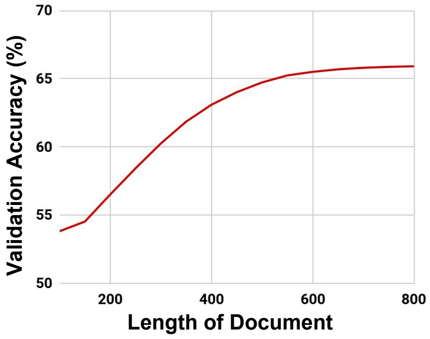

Figure 7: Variation of validation accuracy (%) with respect to length of documents (for APW dataset) for AC-GCN. Documents having more than 100 tokens are selected for this analysis. Please see Section 7.

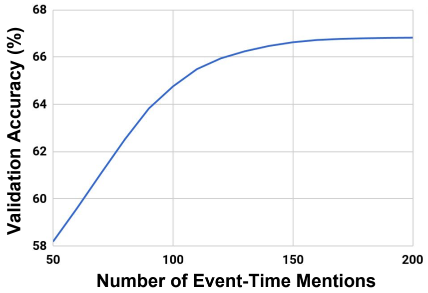

Figure 8: Variation of validation accuracy (%) with number of event-time mentions in documents (for APW dataset) for OE-GCN. Documents with more than 100 tokens are selected for this analysis. Please see Section 7.

8 Conclusion

We propose AD3, an ensemble model which exploits both syntactic and temporal information in a document explicitly to predict its creation time (DCT). To the best of our knowledge, this is the first application of attention based deep models for dating documents. Our experimental results demonstrate the effectiveness of our model over all previous models. We also visualize the attention weights to show that the model is able to choose what is important for the task and filter out noise inherent in language. As part of future work, we would like to incorporate external knowledge as a side information for improved time-stamping of documents.

Acknowledgments

This work is supported by the Ministry of Human Resource Development (MHRD), Government of India.

References

James Allan, Ron Papka, and Victor Lavrenko. 1998. On-line new event detection and tracking. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 37–45, New York, NY, USA. ACM.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. *CoRR*, abs/1704.04675.

Nathanael Chambers. 2012. Labeling documents with timestamps: Learning from their time expressions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 98–106, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wisam Dakka, Luis Gravano, and Panagiotis G. Ipeirotis. 2008. Answering general time sensitive queries. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 1437–1438, New York, NY, USA. ACM.

Franciska M.G. de Jong, H. Rode, and Djoerd Hiemstra. 2005a. *Temporal Language Models for the Disclosure of Historical Text*. KNAW. Imported from EWI/DB PMS [db-utwente:inpr:0000003683].

Franciska M.G. de Jong, H. Rode, and Djoerd Hiemstra. 2005b. *Temporal Language Models for the Disclosure of Historical Text*. KNAW. Imported from EWI/DB PMS [db-utwente:inpr:0000003683].

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216.

Nattiya Kanhabua and Kjetil Nørvåg. 2008. Improving temporal language models for determining time of non-timestamped documents. In *Proceedings of the 12th European Conference on Research and Advanced Technology for Digital Libraries*, ECDL '08, pages 358–370, Berlin, Heidelberg. Springer-Verlag.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Dimitrios Kotsakos, Theodoros Lappas, Dimitrios Kotzias, Dimitrios Gunopulos, Nattiya Kanhabua, and Kjetil Nørvåg. 2014. A burstiness-aware approach for document dating. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 1003–1006, New York, NY, USA. ACM.

Theodoros Lappas, Benjamin Arai, Manolis Platakis, Dimitrios Kotsakos, and Dimitrios Gunopulos. 2009. On burstiness-aware search for document sequences. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 477–486, New York, NY, USA. ACM.

Xiaoyan Li and W. Bruce Croft. 2003. Time-based language models. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 469–475, New York, NY, USA. ACM.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *CoRR*, abs/1703.04826.

Bill McDowell, Nathanael Chambers, Alexander Ororbia II, and David Reitter. 2017. Event ordering with a generalized model for sieve prediction ranking. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 843–853, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 64–75. The COLING 2016 Organizing Committee.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition ldc2011t07. dvd. *Philadelphia: Linguistic Data Consortium*.

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012a. Acquiring temporal constraints between relations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 992–1001. ACM.

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012b. Coupled temporal scoping of relational facts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 73–82. ACM.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 1–9.

Shikhar Vashishth, Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. Dating documents using graph convolution networks. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018), July 15-20, 2018, Melbourne, Australia*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 57–62. Association for Computational Linguistics.

Xiaojun Wan. 2007. Timedtextrank: Adding the temporal dimension to multi-document summarization. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 867–868, New York, NY, USA. ACM.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. 2016a. Stacked attention networks for image question answering. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 21–29.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016b. Hierarchical attention networks for document classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1480–1489.

Gromov-Wasserstein Alignment of Word Embedding Spaces

David Alvarez-Melis
CSAIL, MIT
`dalvmel@mit.edu`

Tommi S. Jaakkola
CSAIL, MIT
`tommi@mit.edu`

Abstract

Cross-lingual or cross-domain correspondences play key roles in tasks ranging from machine translation to transfer learning. Recently, purely unsupervised methods operating on monolingual embeddings have become effective alignment tools. Current state-of-the-art methods, however, involve multiple steps, including heuristic post-hoc refinement strategies. In this paper, we cast the correspondence problem directly as an optimal transport (OT) problem, building on the idea that word embeddings arise from metric recovery algorithms. Indeed, we exploit the *Gromov-Wasserstein* distance that measures how similarities between pairs of words relate across languages. We show that our OT objective can be estimated efficiently, requires little or no tuning, and results in performance comparable with the state-of-the-art in various unsupervised word translation tasks.

1 Introduction

Many key linguistic tasks, within and across languages or domains, including machine translation, rely on learning cross-lingual correspondences between words or other semantic units. While the associated alignment problem could be solved with access to large amounts of parallel data, broader applicability relies on the ability to do so with largely mono-lingual data, from Part-of-Speech (POS) tagging (Zhang et al., 2016), dependency parsing (Guo et al., 2015), to machine translation (Lample et al., 2018). The key subtask of bilingual lexical induction, for example, while long standing as a problem (Fung, 1995; Rapp, 1995, 1999), has been actively pursued recently (Artetxe et al., 2016; Zhang et al., 2017a; Conneau et al., 2018).

Current methods for learning cross-domain correspondences at the word level rely on distributed representations of words, building on the observation that mono-lingual word embeddings exhibit similar geometric properties across languages Mikolov et al. (2013). While most early work assumed some, albeit minimal, amount of parallel data (Mikolov et al., 2013; Dinu et al., 2014; Zhang et al., 2016), recently fully-unsupervised methods have been shown to perform on par with their supervised counterparts (Conneau et al., 2018; Artetxe et al., 2018). While successful, the mappings arise from multiple steps of processing, requiring either careful initial guesses or post-mapping refinements, including mitigating the effect of frequent words on neighborhoods. The associated adversarial training schemes can also be challenging to tune properly (Artetxe et al., 2018).

In this paper, we propose a direct optimization approach to solving correspondences based on recent generalizations of optimal transport (OT). OT is a general mathematical toolbox used to evaluate correspondence-based distances and establish mappings between probability distributions, including discrete distributions such as point-sets. However, the nature of mono-lingual word embeddings renders the classic formulation of OT inapplicable to our setting. Indeed, word embeddings are estimated primarily in a relational manner to the extent that the algorithms are naturally interpreted as metric recovery methods (Hashimoto et al., 2016). In such settings, previous work has sought to bypass this lack of *registration* by jointly optimizing over a matching and an orthogonal mapping (Rangarajan et al., 1997; Zhang et al., 2017b). Due to the focus on distances rather than points, we instead adopt a relational OT formulation based on the Gromov-Wasserstein distance that measures how distances between pairs of words are mapped across languages. We show that the resulting mapping admits an efficient solution and requires little or no tuning.

In summary, we make the following contributions:

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1881–1890
Brussels, Belgium, October 31 - November 4, 2018.

- We propose the use of the Gromov-Wasserstein distance to learn correspondences between word embedding spaces in a fully-unsupervised manner, leading to a theoretically-motivated optimization problem that can be solved efficiently, robustly, in a single step, and requires no post-processing or heuristic adjustments.

- To scale up to large vocabularies we realize an extended mapping to words not part of the original optimization problem.

- We show that the proposed approach performs on par with state-of-the-art neural network based methods on benchmark word translation tasks, while requiring a fraction of the computational cost and/or hyper-parameter tuning.

2 Problem Formulation

In the unsupervised bilingual lexical induction problem we consider two languages with vocabularies V_x and V_y, represented by word embeddings $X = \{\mathbf{x}^{(i)}\}_{i=1}^{n}$ and $Y = \{\mathbf{y}^{(j)}\}_{j=1}^{m}$, respectively, where $\mathbf{x}^{(i)} \in \mathcal{X} \subset \mathbb{R}^{d_x}$ corresponds to $w_i^x \in V_x$ and $\mathbf{y}^{(j)} \in \mathcal{Y} \subset \mathbb{R}^{d_y}$ to $w_j^y \in V_y$. For simplicity, we let $m = n$ and $d_x = d_y$, although our methods carry over to the general case with little or no modifications. Our goal is to learn an alignment between these two sets of words without any parallel data, i.e., we learn to relate $\mathbf{x}^{(i)} \leftrightarrow \mathbf{y}^{(j)}$ with the implication that w_i^x translates to w_j^y.

As background, we begin by discussing the problem of learning an explicit map between embeddings in the supervised scenario. The associated training procedure will later be used for extending unsupervised alignments (Section 3.2).

2.1 Supervised Maps: Procrustes

In the supervised setting, we learn a map $T : \mathcal{X} \rightarrow \mathcal{Y}$ such that $T(\mathbf{x}^{(i)}) \approx \mathbf{y}^{(j)}$ whenever w_j^y is a translation of w_i^x. Let $\mathbf{X}$ and $\mathbf{Y}$ be the matrices whose columns are vectors $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(j)}$, respectively. Then we can find T by solving

$$\min_{T \in \mathcal{F}} \|\mathbf{X} - T(\mathbf{Y})\|_F^2 \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm $\|A\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2}$. Naturally, both the difficulty of finding T and the quality of the resulting alignment depend on the choice of space $\mathcal{F}$. A classic approach constrains T to be orthonormal matrices, i.e., rotations and reflections, resulting in the orthogonal Procrustes problem

$$\min_{\mathbf{P} \in O(n)} \|\mathbf{X} - \mathbf{P}\mathbf{Y}\|_F^2 \tag{2}$$

where $O(n) = \{\mathbf{P} \in \mathbb{R}^{n \times n} \mid \mathbf{P}^\top\mathbf{P} = \mathbf{I}\}$. One key advantage of this formulation is that it has a closed-form solution in terms of a singular value decomposition (SVD), whereas for most other choices of constraint set $\mathcal{F}$ it does not. Given an SVD decomposition $\mathbf{U}\Sigma\mathbf{V}^\top$ of $\mathbf{X}\mathbf{Y}^\top$, the solution to problem (2) is $\mathbf{P}^* = \mathbf{U}\mathbf{V}^\top$ (Schönemann, 1966). Besides obvious computational advantage, constraining the mapping between spaces to be orthonormal is justified in the context of word embedding alignment because orthogonal maps preserve angles (and thus distances), which is often the only information used by downstream tasks (e.g., for nearest neighbor search) that rely on word embeddings. (Smith et al., 2017) further show that orthogonality is required for self-consistency of linear transformations between vector spaces.

Clearly, the Procrustes approach only solves the supervised version of the problem as it requires a known correspondence between the columns of $\mathbf{X}$ and $\mathbf{Y}$. Steps beyond this constraint include using small amounts of parallel data (Zhang et al., 2016) or an unsupervised technique as the initial step to generate pseudo-parallel data (Conneau et al., 2018) before solving for $\mathbf{P}$.

2.2 Unsupervised Maps: Optimal Transport

Optimal transport formalizes the problem of finding a minimum cost mapping between two point sets, viewed as discrete distributions. Specifically, we assume two empirical distributions over embeddings, e.g.,

$$\mu = \sum_{i=1}^{n} \mathbf{p}_i \delta_{\mathbf{x}^{(i)}}, \quad \nu = \sum_{j=1}^{m} \mathbf{q}_j \delta_{\mathbf{y}^{(i)}} \tag{3}$$

where $\mathbf{p}$ and $\mathbf{q}$ are vectors of probability weights associated with each point set. In our case, we usually consider uniform weights, e.g., $\mathbf{p}_i = 1/n$ and $\mathbf{q}_j = 1/m$, although if additional information were provided (such as in the form of word frequencies), those could be naturally incorporated via $\mathbf{p}$ and $\mathbf{q}$ (see discussion at the end of Section 3). We find a *transportation map* T realizing

$$\inf_T \left\{ \int_{\mathcal{X}} c(\mathbf{x}, T(\mathbf{x})) d\mu(\mathbf{x}) \mid T_{\#}\mu = \nu \right\}, \tag{4}$$

where the cost $c(\mathbf{x}, T(\mathbf{x}))$ is typically just $\|\mathbf{x} - T(\mathbf{x})\|$ and $T_{\#}\mu = \nu$ implies that the source points must exactly map to the targets. However, such a map need not exist in general and we instead follow a relaxed Kantorovich's formulation. In this case, the set of transportation plans is a polytope:

$$\Pi(\mathbf{p}, \mathbf{q}) = \{\Gamma \in \mathbb{R}_+^{n \times m} \mid \Gamma \mathbb{1}_n = \mathbf{p},\ \Gamma^\top \mathbb{1}_n = \mathbf{q}\}.$$

The cost function is given as a matrix $\mathbf{C} \in \mathbb{R}^{n \times m}$, e.g., $C_{ij} = \|\mathbf{x}^{(i)} - \mathbf{y}^{(j)}\|$. The total cost incurred by Γ is $\langle \Gamma, C \rangle := \sum_{ij} \Gamma_{ij} C_{ij}$. Thus, the discrete optimal transport (DOT) problem consists of finding a plan Γ that solves

$$\min_{\Gamma \in \Pi(\mathbf{p}, \mathbf{q})} \langle \Gamma, \mathbf{C} \rangle. \tag{5}$$

Problem (5) is a linear program, and thus can be solved exactly in $O(n^3 \log n)$ with interior point methods. However, regularizing the objective leads to more efficient optimization and often better empirical results. The most common such regularization, popularized by Cuturi (2013), involves adding an entropy penalization:

$$\min_{\Gamma \in \Pi(\mathbf{p}, \mathbf{q})} \langle \Gamma, \mathbf{C} \rangle - \lambda H(\Gamma). \tag{6}$$

The solution of this strictly convex optimization problem has the form $\Gamma^* = \operatorname{diag}(\mathbf{a}) \, \mathbf{K} \operatorname{diag}(\mathbf{b})$, with $\mathbf{K} = e^{-\frac{\mathbf{C}}{\lambda}}$ (element-wise), and can be obtained efficiently via the Sinkhorn-Knopp algorithm, a matrix-scaling procedure which iteratively computes:

$$\mathbf{a} \leftarrow \mathbf{p} \oslash \mathbf{K}\mathbf{b} \quad \text{and} \quad \mathbf{b} \leftarrow \mathbf{q} \oslash \mathbf{K}^\top \mathbf{a}, \tag{7}$$

where $\oslash$ denotes entry-wise division. The derivation of these updates is immediate from the form of Γ^* above, combined with the marginal constraints $\Gamma \mathbb{1}_n = \mathbf{p}$, $\Gamma^\top \mathbb{1}_n = \mathbf{q}$ (Peyré and Cuturi, 2018).

Although simple, efficient and theoretically-motivated, a direct application of discrete OT for unsupervised word translation is not appropriate. One reason is that the mono-lingual embeddings are estimated in a relative manner, leaving, e.g., an overall rotation unspecified. Such degrees of freedom can dramatically change the entries of the cost matrix $C_{ij} = \|\mathbf{x}^{(i)} - \mathbf{y}^{(j)}\|$ and the resulting transport map. One possible solution is to simultaneously learn an optimal coupling and an orthogonal transformation (Zhang et al., 2017b). The transport problem is then solved iteratively, using $C_{ij} = \|\mathbf{x}^{(i)} - \mathbf{P}\mathbf{y}^{(j)}\|$, where $\mathbf{P}$ is in turn chosen to minimize the transport cost (via Procrustes). While promising, the resulting iterative approach is sensitive to initialization, perhaps explaining why Zhang et al. (2017b) used an adversarially learned mapping as the initial step. The computational cost can also be prohibitive (Artetxe et al., 2018) though could be remedied with additional development.

We adopt a theoretically well-founded generalization of optimal transport for pairs of points (their distances), thus in line with how the embeddings are estimated in the first place. We explain the approach in detail in the next Section.

3 Transporting across unaligned spaces

In this section we introduce the Gromov-Wasserstein distance, describe an optimization algorithm for it, and discuss how to extend the approach to out-of-sample vectors.

3.1 The Gromov Wasserstein Distance

The classic optimal transport requires a distance between vectors *across* the two domains. Such a metric may not be available, for example, when the sample sets to be matched do not belong to the same metric space (e.g., different dimension). The Gromov-Wasserstein distance (Mémoli, 2011) generalizes optimal transport by comparing the metric spaces directly instead of samples across the spaces. In other words, this framework operates on distances between pairs of points calculated within each domain and measures how these distances compare to those in the other domain. Thus, it requires a weaker but easy to define notion of *distance between distances*, and operates on pairs of points, turning the problem from a linear to a quadratic one.

Formally, in its discrete version, this framework considers two measure spaces expressed in terms of within-domain similarity matrices $(\mathbf{C}, \mathbf{p})$ and $(\mathbf{C}', \mathbf{q})$ and a loss function defined between *similarity pairs*: $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, where $L(C_{ik}, C'_{jl})$ measures the discrepancy between the distances $d(\mathbf{x}^{(i)}, \mathbf{x}^{(k)})$ and $d'(\mathbf{y}^{(j)}, \mathbf{y}^{(l)})$. Typical choices for L are $L(a, b) = \frac{1}{2}(a - b)^2$ or $L(a, b) = \mathrm{KL}(a|b)$. In this framework, $L(C_{ik}, C'_{jl})$ can also be understood as the cost of "matching" i to j *and* k to l.

All the relevant values of $L(\cdot, \cdot)$ can be put in a 4-th order tensor $\mathbf{L} \in \mathbb{R}^{N_1 \times N_1 \times N_2 \times N_2}$, where $\mathbf{L}_{ijkl} = L(C_{ik}, C'_{jl})$. As before, we seek a cou-

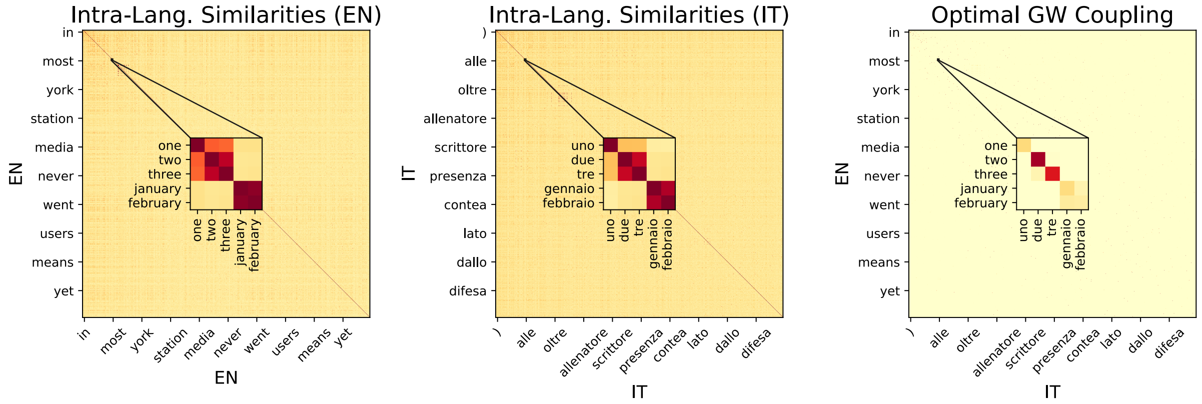

Figure 1: The Gromov-Wasserstein distance is well suited for the task of cross-lingual alignment because it relies on *relational* rather than *positional* similarities to infer correspondences across domains. Computing it requires two intra-domain similarity (or equivalently cost) matrices (**left** & **center**), and it produces an optimal coupling of source and target points with minimal discrepancy cost (**right**).

pling Γ specifying how much mass to transfer between each pair of points from the two spaces. The Gromov-Wasserstein problem is then defined as solving

$$\mathrm{GW}(\mathbf{C}, \mathbf{C}', \mathbf{p}, \mathbf{q}) = \min_{\Gamma \in \Pi(\mathbf{p},\mathbf{q})} \sum_{i,j,k,l} \mathbf{L}_{ijkl} \Gamma_{ij} \Gamma_{kl} \quad (8)$$

Compared to problem (5), this version is substantially harder since the objective is now not only non-linear, but non-convex too.[1] In addition, it requires operating on a fourth-order tensor, which would be prohibitive in most settings. Surprisingly, this problem can be optimized efficiently with first-order methods, whereby each iteration involves solving a traditional optimal transport problem (Peyré et al., 2016). Furthermore, for suitable choices of loss function L, Peyré et al. (2016) show that instead of the $O(N_1^2 N_2^2)$ complexity implied by naive fourth-order tensor product, this computation reduces to $O(N_1^2 N_2 + N1 N_2^2)$ cost. Their approach consists of solving (5) by projected gradient descent, which yields iterations that involve projecting onto $\Pi(\mathbf{p}, \mathbf{q})$ a pseudo-cost matrix of the form

$$\hat{\mathbf{C}}_\Gamma(\mathbf{C}, \mathbf{C}', \Gamma) = \mathbf{C}_{xy} - h_1(\mathbf{C}) \Gamma h_2(\mathbf{C}')^\top \quad (9)$$

where

$$\mathbf{C}_{xy} = f_1(\mathbf{C}) \mathbf{p} \mathbb{1}_m^\top + \mathbb{1}_n \mathbf{q}^\top f_2(\mathbf{C}')^\top$$

and f_1, f_2, h_2, h_2 are functions that depend on the loss L. We provide an explicit algorithm for the case $L = L_2$ at the end of this section.

Once we have solved (8), the optimal transport coupling Γ^* provides an explicit (soft) matching between source and target samples, which for the problem of interest can be interpreted as a probabilistic translation: for every pair of words $(w_{src}^{(i)}, w_{trg}^{(j)})$, Γ^*_{ij} provides a likelihood that these two words are translations of each other. This itself is enough to translate, and we show in the experiments section that Γ^* by itself, without any further post-processing, provides high-quality translations. This stands in sharp contrast to mapping-based methods, which rely on nearest-neighbor computation to infer translations, and thus become prone to hub-word effects which have to be mitigated with heuristic post-processing techniques such as Inverted Softmax (Smith et al., 2017) and Cross-Domain Similarity Scaling (CSLS) (Conneau et al., 2018). The transportation coupling Γ, being normalized *by construction*, requires no such artifacts.

The Gromov-Wasserstein problem (8) possesses various desirable theoretical properties, including the fact that for a suitable choice of the loss function it is indeed a distance:

Theorem 3.1 (Mémoli 2011)**.** *With the choice* $L = L_2$, $GW^{\frac{1}{2}}$ *is a distance on the space of metric measure spaces.*

Solving problem (8) therefore yields a fascinating accompanying notion: the *Gromov-Wasserstein distance between languages*, a measure of semantic discrepancy purely based on the relational characterization of their word embeddings. Owing to Theorem 3.1, such values can be

---

[1] In fact, the discrete (Monge-type) formulation of the problem is essentially an instance of the well-known (and NP-hard) quadratic assignment problem (QAP).

interpreted as distances, so that, e.g., the triangle inequality holds among them. In Section 4.4 we compare various languages in terms of their GW-distance.

Finally, we note that whenever word frequency counts are available, those would be used for $\mathbf{p}$ and $\mathbf{q}$. If they are not, but words are sorted according to occurrence (as they often are in popular off-the-shelf embedding formats), one can estimate rank-probabilities such as Zipf power laws, which are known to accurately model multiple languages (Piantadosi, 2014). In order to provide a fair comparison to previous work, throughout our experiments **we use uniform distributions** so as to avoid providing our method with additional information not available to others.

3.2 Scaling Up

While the pure Gromov-Wasserstein approach leads to high quality solutions, it is best suited to small-to-moderate vocabulary sizes,[2] since its optimization becomes prohibitive for very large problems. For such settings, we propose a two-step approach in which we first match a subset of the vocabulary via the optimal coupling, after which we learn an orthogonal mapping through a modified Procrustes problem. Formally, suppose we solve problem (8) for a reduced matrices $\mathbf{X}_{1:k}$ and $\mathbf{Y}_{i:k}$ consisting of the first columns k of $\mathbf{X}$ and $\mathbf{Y}$, respectively, and let Γ^* be the optimal coupling. We seek an orthogonal matrix that best recovers the barycentric mapping implied by Γ^*. Namely, we seek to find $\mathbf{P}$ which solves:

$$\min_{\mathbf{P}\in O(n)} \|\mathbf{X}\Gamma^* - \mathbf{P}\mathbf{Y}\|_2^2 \tag{10}$$

Just as problem (2), it is easy to show that this Procrustes-type problem has a closed form solution in terms of a singular value decomposition. Namely, the solution to (10) is $\mathbf{P}^* = \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U}\Sigma\mathbf{V}^* = \mathbf{X}_{1:m}\Gamma^*\mathbf{Y}_{1:m}^\top$. After obtaining this projection, we can immediately map the rest of the embeddings via $\hat{\mathbf{y}}^{(j)} = \mathbf{P}^*\mathbf{y}^{(j)}$.

We point out that this two-step procedure resembles that of Conneau et al. (2018). Both ultimately produce an orthogonal mapping obtained by solving a Procrustes problems, but they differ in the way they produce pseudo-matches to allow for such second-step: while their approach relies on an adversarially-learned transformation, we use an explicit optimization problem.

Algorithm 1 Gromov-Wasserstein Computation for Word Embedding Alignment

```
Input: Source and target embeddings $\mathbf{X}, \mathbf{Y}$. Regularization $\lambda$. Probability vectors $\mathbf{p}, \mathbf{q}$.
// Compute intra-language similarities
$\mathbf{C}_s \leftarrow \cos(\mathbf{X}, \mathbf{X}), \quad \mathbf{C}_t \leftarrow \cos(\mathbf{Y}, \mathbf{Y})$
$\mathbf{C}_{st} \leftarrow \mathbf{C}_s^2 \mathbf{p} \mathbb{1}_m^\top + \mathbb{1}_n \mathbf{q} (\mathbf{C}_t^2)^\top$
while not converged do
  // Compute pseudo-cost matrix (Eq. (9))
  $\hat{\mathbf{C}}_\Gamma \leftarrow \mathbf{C}_{st} - 2\mathbf{C}_s \Gamma \mathbf{C}_t^\top$
  // Sinkhorn iterations (Eq. (7))
  $\mathbf{a} \leftarrow \mathbb{1}, \quad \mathbf{K} \leftarrow \exp\{-\hat{\mathbf{C}}_\Gamma / \lambda\}$
  while not converged do
    $\mathbf{a} \leftarrow \mathbf{p} \oslash \mathbf{K}\mathbf{b}, \ \mathbf{b} \leftarrow \mathbf{q} \oslash \mathbf{K}^\top \mathbf{a}$
  end while
  $\Gamma \leftarrow \mathrm{diag}(\mathbf{a})\, \mathbf{K}\, \mathrm{diag}(\mathbf{b})$
end while
// Optional step: Learn explicit projection
$\mathbf{U}, \Sigma, \mathbf{V}^\top \leftarrow \mathrm{SVD}(\mathbf{X}\Gamma\mathbf{Y}^\top)$
$\mathbf{P} = \mathbf{U}\mathbf{V}^\top$
return $\Gamma, \mathbf{P}$
```

We end this section by discussing parameter and configuration choices. To leverage the fast algorithm of Peyré et al. (2016), we always use the L_2 distance as the loss function L between cost matrices. On the other hand, we observed throughout our experiments that the choice of cosine distance as the metric in both spaces consistently leads to better results, which agrees with common wisdom on computing distances between word embeddings. This leaves us with a single hyper-parameter to control: the entropy regularization term λ. By applying any sensible normalization on the cost matrices (e.g., dividing by the mean or median value), we are able to almost entirely eliminate sensitivity to that parameter. In practice, we use a simple scheme in all experiments: we first try the same fixed value ($\lambda = 5 \times 10^{-5}$), and if the regularization proves too small (by leading to floating point errors), we instead use $\lambda = 1 \times 10^{-4}$. We never had to go beyond these two values in all our experiments. We emphasize that at no point we use train (let alone test) supervision available with many datasets—model selection is done solely in terms of the unsupervised objective. Pseudocode for the full method (with $L = L_2$ and cosine similarity) is shown here as Algorithm 1.

---

[2] As shown in the experimental section, we are able to run problems of size in the order of $|V_s| \approx 10^5 \approx |V_t|$ on a single machine **without** relying on GPU computation.

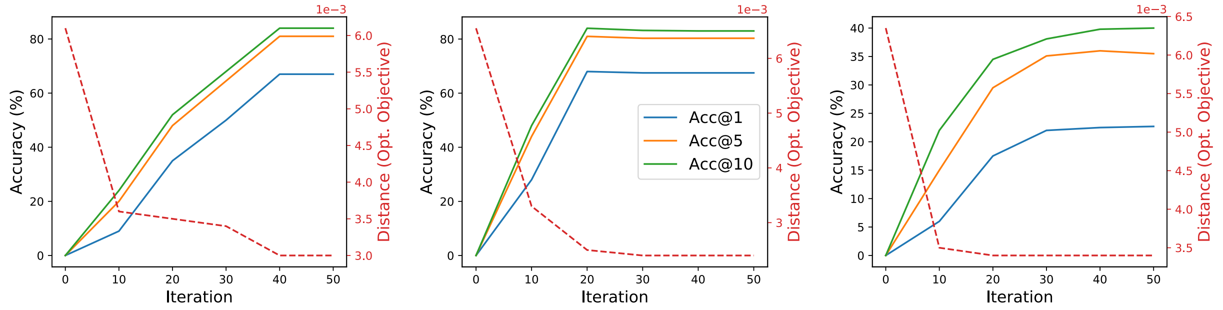

(a) $\text{EN} \rightarrow \text{FR}$, 15K words, $\lambda = 5 \cdot 10^{-4}$ (b) $\text{EN} \rightarrow \text{FR}$, 15K words, $\lambda = 10^{-4}$ (c) $\text{EN} \rightarrow \text{RU}$, 15K words, $\lambda = 10^{-4}$

Figure 2: **Training dynamics for the Gromov-Wasserstein alignment problem**. The algorithm provably makes progress in each iteration, and the objective (red dashed line) closely follows the metric of interest (translation accuracy, not available during training). More related languages (e.g., EN →FR in 2b,2a) lead to faster optimization, while more distant pairs yield slower learning curves (EN →RU, 2c).

4 Experiments

Through this experimental evaluation we seek to: (i) understand the optimization dynamics of the proposed approach (§4.2), evaluate its performance on benchmark cross-lingual word embedding tasks (§4.3), and (iii) qualitatively investigate the notion of distance-between-languages it computes (§4.4). Rather than focusing solely on prediction accuracy, we seek to demonstrate that the proposed approach offers a fast, principled, and robust alternative to state-of-the-art multi-step methods, delivering comparable performance.

4.1 Evaluation Tasks and Methods

Datasets We evaluate our method on two standard benchmark tasks for cross-lingual embeddings. First, we consider the dataset of Conneau et al. (2018), which consists of word embeddings trained with FASTTEXT (Bojanowski et al., 2017) on Wikipedia and parallel dictionaries for 110 language pairs. Here, we focus on the language pairs for which they report results: English (EN) from/to Spanish (ES), French (FR), German (DE), Russian (RU) and simplified Chinese (ZH). We do not report results on Esperanto (EO) as dictionaries for that language were not provided with the original dataset release.

For our second set of experiments, we consider the—substantially harder[3]—dataset of (Dinu et al., 2014), which has been extensively compared against in previous work. It consists of embeddings and dictionaries in four pairs of languages; EN from/to ES, IT, DE, and FI (Finnish).

Methods To see how our fully-unsupervised method compares with methods that require (some) cross-lingual supervision, we follow (Conneau et al., 2018) and consider a simple but strong baseline consisting of solving a procrustes problem directly using the available cross-lingual embedding pairs. We refer to this method simply as PROCRUSTES. In addition, we compare against the fully-unsupervised methods of Zhang et al. (2017a), Artetxe et al. (2018) and Conneau et al. (2018).[4] As proposed by the latter, we use CSLS whenever nearest neighbor search is required, which has been shown to improve upon naive nearest-neighbor retrieval in multiple work.

4.2 Training Dynamics of G-W

As previously mentioned, our approach involves only two optimization choices, one of which is required only for very large settings. When running Algorithm 1 for the full set of embeddings is infeasible (due to memory limitations), one must decide what fraction of the embeddings to use during optimization. In our experiments, we use the largest possible size allowed by memory constraints, which was found to be $K = 20,000$ for the personal computer we used.

The other—more interesting—optimization choice involves the entropy regularization parameter λ used within the Sinkhorn iterations. Large regularization values lead to denser optimal coupling Γ^*, while less regularization leads to sparser solutions,[5] at the cost of a harder (more

---

[3] We discuss the difference in hardness of these two benchmark datasets in Section 4.3.

[4] Despite its relevance, we do not include the OT-based method of Zhang et al. (2017b) in the comparison because their implementation required use of proprietary software.

[5] In the limit $\lambda \rightarrow 0$, when $n = m$, the solution converges

			En-Es		En-Fr		En-De		En-It		En-Ru	
	Supervision	Time	→	←	→	←	→	←	→	←	→	←
Procrustes	5K words	3	77.6	77.2	74.9	75.9	68.4	67.7	73.9	73.8	47.2	58.2
Procrustes + CSLS	5K words	3	81.2	82.3	81.2	82.2	73.6	71.9	76.3	**75.5**	51.7	63.7
(Conneau et al., 2018)	None	957	**81.7**	**83.3**	**82.3**	82.1	74.0	72.2	77.4	76.1	**52.4**	**61.4**
G-W ($\lambda = 10^{-4}$)	None	70	78.3	79.5	79.3	78.3	69.6	66.9	75.3	74.1	26.1	35.4
G-W ($\lambda = 10^{-5}$)	None	37	**81.7**	80.4	81.3	78.9	71.9	**72.8**	**78.9**	75.2	45.1	43.7

Table 1: Performance (P@1) of unsupervised and minimally-supervised methods on the dataset of Conneau et al. (2018). The time columns shows the average runtime in minutes of an instance (i.e., one language pair) of the method in this task on the same quad-core **CPU** machine.

non-convex) optimization problem.

In Figure 2 we show the training dynamics of our method when learning correspondences between word embeddings from the dataset of Conneau et al. (2018). As expected, larger values of λ lead to smoother improvements with faster runtime-per-iteration, at a price of some drop in performance. In addition, we found that computing GW distances between closer languages (such as En and Fr) leads to faster convergence than for more distant ones (such as En and Ru, in Fig. 2c).

Worth emphasizing are three desirable optimization properties that set apart the Gromov-Wasserstein distance from other unsupervised alignment approaches, particularly adversarial-training ones: (i) the objective decreases monotonically (ii) its value closely follows the true metric of interest (translation, which naturally is not available during training) and (iii) there is no risk of degradation due to *overtraining*, as is the case for adversarial-based methods trained with stochastic gradient descent (Conneau et al., 2018).

4.3 Benchmark Results

We report the results on the dataset of Conneau et al. (2018) in Table 1. The strikingly high performance of all methods on this task belies the hardness of the general problem of unsupervised cross-lingual alignment. Indeed, as pointed out by Artetxe et al. (2018), the fastText embeddings provided in this task are trained on very large and highly comparable—across languages—corpora (Wikipedia), and focuses on closely related pairs of languages. Nevertheless, we carry out experiments here to have a broad evaluation of our approach in both *easier* and *harder* settings.

Next, we present results on the more challenging dataset of (Dinu et al., 2014) in Table 2. Here, we rely on the results reported by (Artetxe et al., 2018) since by the time of writing the present work their implementation was not available yet.

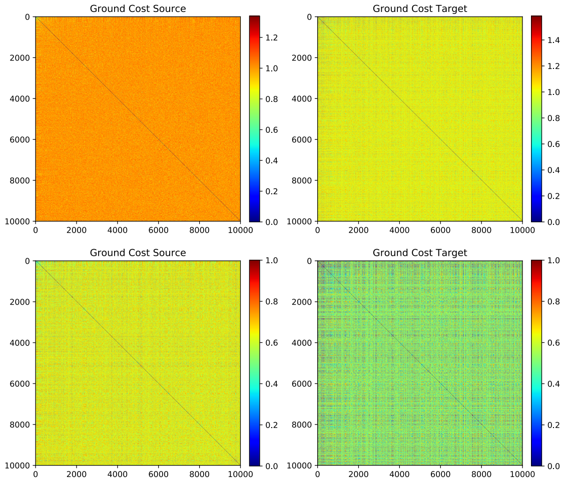

Figure 3: **Top**: Word embeddings trained on non-comparable corpora can lead to uneven distributions of pairwise distances as shown here for the En-Fi pair of (Dinu et al., 2014). **Bottom:** Normalizing the cost matrices leads to better optimization and improved performance.

Part of what makes this dataset hard is the wide discrepancy between word distance across languages, which translates into uneven distance matrices (Figure 3), and in turn leads to poor results for G-W. To account for this, previous work has relied on an initial whitening step on the embeddings. In our case, it suffices to normalize the pairwise similarity matrices to the same range to obtain substantially better results. While we have observed that careful choice of the regularization parameter λ can obviate the need for this step, we opt for the normalization approach since it allows us to optimize without having to tune λ.

We compare our method (with and without nor-

---

to a permutation matrix, which gives a hard-matching solution to the transportation problem (Peyré and Cuturi, 2018).

	En-It		En-De		En-Fi		En-Es	
	P@1	Time	P@1	Time	P@1	Time	P@1	Time
(Zhang et al., 2017a)†	0	46.6	0	46.0	0.07	44.9	0.07	43.0
(Conneau et al., 2018)†	45.40	46.1	47.27	45.4	1.62	44.4	36.20	45.3
(Artetxe et al., 2018)†	48.53	8.9	**48.47**	7.3	**33.50**	12.9	**37.60**	9.1
G-W	44.4	35.2	37.83	36.7	6.8	15.6	12.5	18.4
G-W + Normalize	**49.21**	36	46.5	33.2	18.3	42.1	**37.60**	38.2

Table 2: Results of unsupervised methods on the dataset of Dinu et al. (2014) with runtimes in minutes. Those marked with † are from (Artetxe et al., 2018). Note that their runtimes correspond to GPU computation, while ours are CPU-minutes, so the numbers are not directly comparable.

malization) against alternative approaches in Table 2. Note that we report the runtimes of Artetxe et al. (2018) as-is, which are obtained by running on a Titan XP GPU, while our runtimes are, as before, obtained purely by CPU computation.

4.4 Qualitative Results

As mentioned earlier, Theorem 3.1 implies that the optimal value of the Gromov-Wasserstein problem can be legitimately interpreted as a distance between languages, or more explicitly, between their word embedding spaces. This distributional notion of distance is completely determined by pair-wise geometric relations between these vectors. In Figure 4 we show the values $\mathrm{GW}(\mathbf{C}_s, \mathbf{C}_t, \mathbf{p}, \mathbf{q})$ computed on the fastText word embeddings of Conneau et al. (2018) corresponding to the most frequent 2000 words in each language.

Overall, these distances conform to our intuitions: the cluster of romance languages exhibits some of the shortest distances, while classical Chinese (Zh) has the overall largest discrepancy with all other languages. But somewhat surprisingly, Russian is relatively close to the romance languages in this metric. We conjecture that this could be due to Russian's rich morphology (a trait shared by romance languages but not English). Furthermore, both Russian and Spanish are pro-drop languages (Haspelmath, 2001) and share syntactic phenomena, such as dative subjects (Moore and Perlmutter, 2000; Melis et al., 2013) and differential object marking (Bossong, 1991), which might explain why Es is closest to Ru overall.

On the other hand, English appears remarkably isolated from all languages, equally distant from its germanic (De) and romance (Fr) cousins. Indeed, other aspects of the data (such as corpus size) might be underlying these observations.

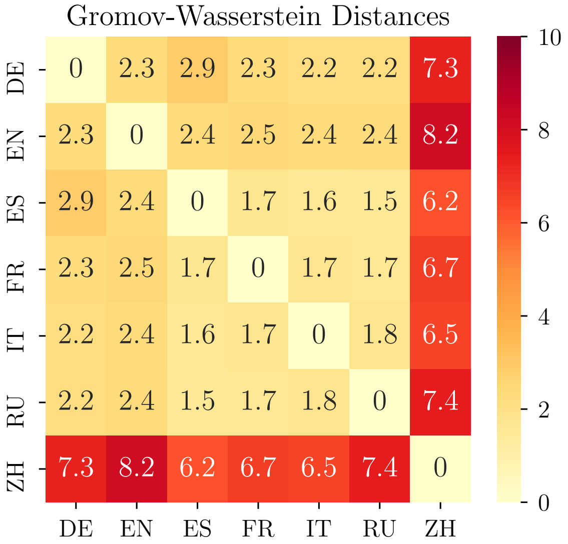

	DE	EN	ES	FR	IT	RU	ZH
DE	0	2.3	2.9	2.3	2.2	2.2	7.3
EN	2.3	0	2.4	2.5	2.4	2.4	8.2
ES	2.9	2.4	0	1.7	1.6	1.5	6.2
FR	2.3	2.5	1.7	0	1.7	1.7	6.7
IT	2.2	2.4	1.6	1.7	0	1.8	6.5
RU	2.2	2.4	1.5	1.7	1.8	0	7.4
ZH	7.3	8.2	6.2	6.7	6.5	7.4	0

Figure 4: Pairwise language Gromov-Wasserstein distances obtained as the minimal transportation cost (8) between word embedding similarity matrices. Values scaled by 10^2 for easy visualization.

5 Related Work

Study of the problem of bilingual lexical induction goes back to Rapp (1995) and Fung (1995). While the literature on this topic is extensive, we focus here on recent fully-unsupervised and minimally-supervised approaches, and refer the reader to one of various existing surveys for a broader panorama (Upadhyay et al., 2016; Ruder et al., 2017).

Methods with coarse or limited parallel data. Most of these fall in one of two categories: methods that learn a mapping from one space to the other, e.g., as a least-squares objective (e.g., (Mikolov et al., 2013)) or via orthogonal transformations Zhang et al. (2016); Smith et al. (2017); Artetxe et al. (2016), and methods that find a common space on which to project both sets of embeddings (Faruqui and Dyer, 2014; Lu et al., 2015).

Fully Unsupervised methods. Conneau et al. (2018) and Zhang et al. (2017a) rely on adversarial training to produce an initial alignment between the spaces. The former use pseudo-matches derived from this initial alignment to solve a Procrustes (2) alignment problem. Our Gromov-Wasserstein framework can be thought of as providing an alternative to these adversarial training steps, albeit with a concise optimization formulation and producing explicit matches (via the optimal coupling) instead of depending on nearest neighbor search, as the adversarially-learnt mappings do.

Zhang et al. (2017b) also leverage optimal transport distances for the cross-lingual embedding task. However, to address the issue of non-alignment of embedding spaces, their approach follows the joint optimization of the transportation and procrustes problem as outlined in Section 2.2. This formulation makes an explicit modeling assumption (invariance to unitary transformations), and requires repeated solution of Procrustes problems during alternating minimization. Gromov-Wasserstein, on the other hand, is more flexible and makes no such assumption, since it directly deals with similarities rather than vectors. In the case where it is required, such an orthogonal mapping can be obtained by solving a single procrustes problem, as discussed in Section 3.2.

6 Discussion and future work

In this work we provided a direct optimization approach to cross-lingual word alignment. The Gromov-Wasserstein distance is well-suited for this task as it performs a relational comparison of word-vectors across languages rather than word-vectors directly. The resulting objective is concise, and can be optimized efficiently. The experimental results show that the resulting alignment framework is fast, stable and robust, yielding near state-of-the-art performance at a computational cost orders of magnitude lower than that of alternative fully unsupervised methods.

While directly solving Gromov-Wasserstein problems of reasonable size is feasible, scaling up to large vocabularies made it necessary to learn an explicit mapping via Procrustes. GPU computations or stochastic optimization could help avoid this secondary step.

Acknowledgments

The authors would like to thank the anonymous reviewers for helpful feedback. The work was partially supported by MIT-IBM grant "Adversarial learning of multimodal and structured data", and Graduate Fellowships from Hewlett Packard and CONACYT.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789—-798. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Georg Bossong. 1991. Differential object marking in Romance and beyond. *New analyses in Romance linguistics*, pages 143–170.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word Translation Without Parallel Data. In *International Conference on Learning Representations*.

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292—-2300.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568.*

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

Pascale Fung. 1995. Compiling bilingual lexicon entries from a non-parallel English-Chinese corpus. In *Third Workshop on Very Large Corpora.*

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations.

In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1234–1244.

Tatsunori B Hashimoto, David Alvarez-Melis, and Tommi S Jaakkola. 2016. Word Embeddings as Metric Recovery in Semantic Spaces. *Transactions of the Association for Computational Linguistics*, 4:273–286.

Martin Haspelmath. 2001. The European linguistic area: Standard Average European. In *Language typology and language universals: An international handbook*, volume 2, pages 1492–1510. de Gruyter.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised Machine Translation Using Monolingual Corpora Only. *International Conference on Learning Representations*.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256.

Chantal Melis, Marcela Flores, and A Holvoet. 2013. On the historical expansion of non-canonically marked 'subjects' in Spanish. *The diachronic Typology of Non-Canonical Subjects, Amsterdam/Philadelphia, Benjamins*, pages 163–184.

Facundo Mémoli. 2011. Gromov–Wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4):417–487.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting Similarities among Languages for Machine Translation. *arXiv preprint arXiv:1309.4168v1*, pages 1–10.

John Moore and David M. Perlmutter. 2000. What does it take to be a dative subject? *Natural Language and Linguistic Theory*, 18(2):373–416.

Gabriel Peyré and Marco Cuturi. 2018. Computational Optimal Transport. Technical report.

Gabriel Peyré, Marco Cuturi, and Justin Solomon. 2016. Gromov-Wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pages 2664–2672.

Steven T Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review*, 21(5):1112–1130.

Anand Rangarajan, Haili Chui, and Fred L Bookstein. 1997. The Softassign Procrustes Matching Algorithm. *Lecture Notes in Computer Science*, 1230:29–42.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 320–322. Association for Computational Linguistics.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 519–526. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual embedding models. *arXiv preprint arXiv:1706.04902*.

Peter H. Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Samuel L Smith, David H P Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *International Conference on Learning Representations*.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. *arXiv preprint arXiv:1604.00425*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017a. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1959–1970.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Earth Mover's Distance Minimization for Unsupervised Bilingual Lexicon Induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten Pairs to Tag – Multilingual POS Tagging via Coarse Mapping between Embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1307–1317, San Diego, California. Association for Computational Linguistics.

Deep Probabilistic Logic: A Unifying Framework for Indirect Supervision

Hai Wang[1]* **Hoifung Poon**[2]
[1] Toyota Technological Institute at Chicago, Chicago, Illinois, USA
[2] Microsoft Research, Redmond, WA, USA
`haiwang@ttic.edu` `hoifung@microsoft.com`

Abstract

Deep learning has emerged as a versatile tool for a wide range of NLP tasks, due to its superior capacity in representation learning. But its applicability is limited by the reliance on annotated examples, which are difficult to produce at scale. Indirect supervision has emerged as a promising direction to address this bottleneck, either by introducing labeling functions to automatically generate noisy examples from unlabeled text, or by imposing constraints over interdependent label decisions. A plethora of methods have been proposed, each with respective strengths and limitations. Probabilistic logic offers a unifying language to represent indirect supervision, but end-to-end modeling with probabilistic logic is often infeasible due to intractable inference and learning. In this paper, we propose deep probabilistic logic (DPL) as a general framework for indirect supervision, by composing probabilistic logic with deep learning. DPL models label decisions as latent variables, represents prior knowledge on their relations using weighted first-order logical formulas, and alternates between learning a deep neural network for the end task and refining uncertain formula weights for indirect supervision, using variational EM. This framework subsumes prior indirect supervision methods as special cases, and enables novel combination via infusion of rich domain and linguistic knowledge. Experiments on biomedical machine reading demonstrate the promise of this approach.

1 Introduction

Deep learning has proven successful in a wide range of NLP tasks (Bahdanau et al., 2014; Bengio et al., 2003; Clark and Manning, 2016; Hermann et al., 2015; Sutskever et al., 2014). The versatility stems from its capacity of learning a compact representation of complex input patterns (Goodfellow et al., 2016). However, success of deep learning is bounded by its reliance on labeled examples, which are expensive and time-consuming to produce. Indirect supervision has emerged as a promising direction for breaching the annotation bottleneck. A powerful paradigm is *joint inference* (Chang et al., 2007; Poon and Domingos, 2008; Druck et al., 2008; Ganchev et al., 2010), which leverages linguistic and domain knowledge to impose constraints over interdependent label decisions. More recently, another powerful paradigm, often loosely called *weak supervision*, has gained in popularity. The key idea is to introduce labeling functions to automatically generate (noisy) training examples from unlabeled text. *Distant supervision* is a prominent example that used existing knowledge bases for this purpose (Craven and Kumlien, 1999; Mintz et al., 2009). *Data programming* went further by soliciting labeling functions from domain experts (Ratner et al., 2016; Bach et al., 2017).

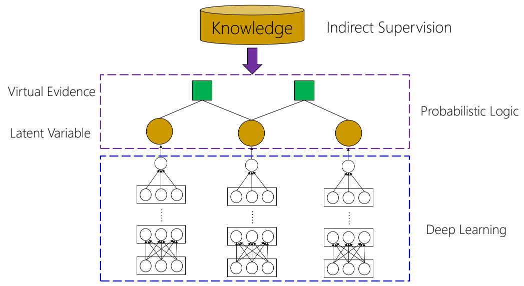

Figure 1: Deep Probabilistic Logic: A general framework for combining indirect supervision strategies by composing probabilistic logic with deep learning. Learning amounts to maximizing conditional likelihood of virtual evidence given input by summing up latent label decisions.

Indirect-supervision methods have achieved remarkable successes in a number of NLP tasks, but they also exhibit serious limitations. Distant supervision often produces incorrect labels, whereas labeling functions from data programming vary in

---

[*] This work was conducted at Microsoft Research.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1891–1902
Brussels, Belgium, October 31 - November 4, 2018.

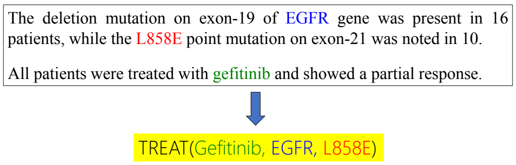

The deletion mutation on exon-19 of EGFR gene was present in 16 patients, while the L858E point mutation on exon-21 was noted in 10.

All patients were treated with gefitinib and showed a partial response.

Figure 2: Example of cross-sentence relation extraction for precision cancer treatment.

quality and coverage, and may contradict with each other on individual instances. Joint inference incurs greater modeling complexity and often requires specialized learning and inference procedures.

Since these methods draw on diverse and often orthogonal sources of indirect supervision, combining them may help address their limitations and amplify their strengths. Probabilistic logic offers an expressive language for such an integration, and is well suited for resolving noisy and contradictory information (Richardson and Domingos, 2006). Unfortunately, probabilistic logic generally incurs intractable learning and inference, often rendering end-to-end modeling infeasible.

In this paper, we propose **deep probabilistic logic (DPL)** as a unifying framework for indirect supervision (Figure 1). Specifically, we made four contributions. First, we introduce a modular design to compose probabilistic logic with deep learning, with a supervision module that represents indirect supervision using probabilistic logic, and a prediction module that performs the end task using a deep neural network. Label decisions are modeled as latent variables and serve as the interface between the two modules.

Second, we show that all popular forms of indirect supervision can be represented in DPL by generalizing virtual evidence (Subramanya and Bilmes, 2007; Pearl, 2014). Consequently, these diverse methods can be easily combined within a single framework for mutual amplification.

Third, we show that our problem formulation yields a well-defined learning objective (maximizing conditional likelihood of virtual evidence). We proposed a modular learning approach by decomposing the optimization over the supervision and prediction modules, using variational EM, which enables us to apply state-of-the-art methods for probabilistic logic and deep learning.

Finally, we applied DPL to biomedical machine reading (Quirk and Poon, 2017; Peng et al., 2017). Biomedicine offers a particularly attractive application domain for exploring indirect supervision. Biomedical literature grows by over one million each year[1], making it imperative to develop machine reading methods for automating knowledge curation (Figure 2). While crowd sourcing is hardly applicable, there are rich domain knowledge and structured resources to exploit for indirect supervision. Using cross-sentence relation extraction and entity linking as case studies, we show that distant supervision, data programming, and joint inference can be seamlessly combined in DPL to substantially improve machine reading accuracy, without requiring any manually labeled examples.[2]

2 Related Work

Distant supervision This paradigm was first introduced for binary relation extraction (Craven and Kumlien, 1999; Mintz et al., 2009). In its simplest form, distant supervision generates a positive example if an entity pair with a known relation co-occurs in a sentence, and samples negative examples from co-occurring entity pairs not known to have the given relation. It has recently been extended to cross-sentence relation extraction (Quirk and Poon, 2017; Peng et al., 2017). In principle, one simply looks beyond single sentences for co-occurring entity pairs. However, this can introduce many false positives and prior work used a small sliding window and filtering (minimal-span) to mitigate training noise. Even so, accuracy is relatively low. Both Quirk and Poon (2017) and Peng et al. (2017) used ontology-based string matching for entity linking, which also incurs many false positives, as biomedical entities are highly ambiguous (e.g., PDF and AAAS are gene names). Distant supervision for entity linking is relatively underexplored, and prior work generally focuses on Freebase entities, where links to the corresponding Wikipedia articles are available for learning (Huang et al., 2015).

Data Programming Instead of annotated examples, domain experts are asked to produce labeling functions, each of which assigns a label to an instance if the input satisfies certain conditions, often specified by simple rules (Ratner et al., 2016). This paradigm is useful for semantic tasks, as high-precision text-based rules are often easy to come by. However, there is no guarantee on broad coverage, and labeling functions are still noisy and may contradict with each other. The common denoising strategy assumes that labeling functions make random mistakes, and focuses on estimating their

---

[1] `http://ncbi.nlm.nih.gov/pubmed`

[2] The DPL code and datasets will be made available at `http://hanover.azurewebsites.net`.

accuracy and correlation (Ratner et al., 2016; Bach et al., 2017). A more sophisticated strategy also models instance-level labels and uses instance embedding to estimate instance-level weight for each labeling function (Liu et al., 2017).

Joint Inference Distant supervision and data programming focus on infusing weak supervision on individual labels. Additionally, there is rich linguistic and domain knowledge that does not specify values for individual labels, but imposes hard or soft constraints on their joint distribution. For example, if two mentions are coreferent, they should agree on entity properties (Poon and Domingos, 2008). There is a rich literature on joint inference for NLP applications. Notable methodologies include constraint-driven learning (Chang et al., 2007), general expectation (Druck et al., 2008), posterior regularization (Ganchev et al., 2010), and probabilistic logic (Poon and Domingos, 2008). Constraints can be imposed on relational instances or on model expectations. Learning and inference are often tailor-made for each approach, including beam search, primal-dual optimization, weighted satisfiability solvers, etc. Recently, joint inference has also been used in denoising distant supervision. Instead of labeling all co-occurrences of an entity pair with a known relation as positive examples, one only assumes that at least one instance is positive (Hoffmann et al., 2011; Lin et al., 2016).

Probabilistic Logic Probabilistic logic combines logic's expressive power with graphical model's capability in handling uncertainty. A representative example is Markov logic (Richardson and Domingos, 2006), which define a probability distribution using weighted first-order logical formulas as templates for a Markov model. Probabilistic logic has been applied to incorporating indirect supervision for various NLP tasks (Poon and Domingos, 2007, 2008; Poon and Vanderwende, 2010), but its expressive power comes at a price: learning and inference are generally intractable, and end-to-end modeling often requires heavy approximation (Kimmig et al., 2012). In DPL, we limit the use of probabilistic logic to modeling indirect supervision in the supervision module, leaving end-to-end modeling to deep neural network in the prediction module. This alleviates the computational challenges in probabilistic logic, while leveraging the strength of deep learning in distilling complex patterns from high-dimension data.

Knowledge-Rich Deep Learning Infusing knowledge in neural network training is a long-standing challenge in deep learning (Towell and Shavlik, 1994). Hu et al. (2016a,b) first used logical rules to help train a convolutional neural network for sentiment analysis. DPL draws inspiration from their approach, but is more general and theoretically well-founded. Hu et al. (2016a,b) focused on supervised learning and the logical rules were introduced to augment labeled examples via posterior regularization (Ganchev et al., 2010). DPL can incorporate both direct and indirect supervision, including posterior regularization and other forms of indirect supervision. Like DPL, Hu et al. (2016b) also refined uncertain weights of logical rules, but they did it in a heuristic way by appealing to symmetry with standard posterior regularization. We provide a novel problem formulation using generalized virtual evidence, which shows that their heuristics is a special case of variational EM and opens up opportunities for other optimization strategies.

Deep generative models also combine deep learning with probabilistic models, but focus on uncovering latent factors to support generative modeling and semi-supervised learning (Kingma and Welling, 2013; Kingma et al., 2014). Knowledge infusion is limited to introducing structures among the latent variables (e.g., Markov chain) (Johnson et al., 2016). In DPL, we focus on learning a discriminative model for predicting the latent labels, using a probabilistic model defined by probabilistic logic to inject indirect supervision.

3 Deep Probabilistic Logic

In this section, we introduce deep probabilistic logic (DPL) as a unifying framework for indirect supervision. Label decisions are modeled as latent variables. Indirect supervision is represented as generalized virtual evidence, and learning maximizes the conditional likelihood of virtual evidence given input. We first review the idea of virtual evidence and show how it can be generalized to represent any form of indirect supervision. We then formulate the learning objective and show how it can be optimized using variational EM.

Given a prediction task, let $\mathcal{X}$ denote the set of possible inputs and $\mathcal{Y}$ the set of possible outputs. The goal is to train a prediction module $\Psi(x, y)$ that scores output y given input x. Without loss of generality, we assume that $\Psi(x, y)$ defines the conditional probability $P(y|x)$ using a deep neural network with a softmax layer at the top. Let $X = (X_1, \cdots, X_N)$ denote a sequence of inputs and $Y = (Y_1, \cdots, Y_N)$ the corresponding outputs. We consider the setting where Y are unobserved, and $\Psi(x, y)$ is learned using indirect supervision.

Virtual evidence Pearl (Pearl, 2014) first introduced the notion of virtual evidence, which has been used to incorporate label preference in semi-supervised learning (Reynolds and Bilmes, 2005; Subramanya and Bilmes, 2007; Li, 2009) and grounded learning (Parikh et al., 2015). Suppose we have a prior belief on the value of y, it can be represented by introducing a binary variable v as a dependent of y such that $P(v = 1|y = l)$ is proportional to the prior belief of $y = l$. $v = 1$ is thus an observed evidence that imposes soft constraints over y. Direct supervision (i.e., observed label) for y is a special case when the belief is concentrated on a specific value $y = l^*$ (i.e., $P(v = 1|y = l) = 0$ for any $l \neq l^*$). The virtual evidence v can be viewed as a reified variable for a potential function $\Phi(y) \propto P(v = 1|y)$. This enables us to generalize virtual evidence to arbitrary potential functions $\Phi(X, Y)$ over the inputs and outputs. In the rest of the paper, we will simply refer to the potential functions as virtual evidences, without introducing the reified variables explicitly.

DPL Let $K = (\Phi_1, \cdots, \Phi_V)$ be a set of virtual evidence derived from prior knowledge. DPL comprises of a supervision module over K and a prediction module over all input-output pairs (Figure 1), and defines a probability distribution:

$$P(K, Y|X) \propto \prod_v \Phi_v(X, Y) \cdot \prod_i \Psi(X_i, Y_i)$$

Without loss of generality, we assume that virtual evidences are log-linear factors, which can be compactly represented by weighted first-order logical formulas (Richardson and Domingos, 2006). Namely, $\Phi_v(X, Y) = \exp(w_v \cdot f_v(X, Y))$, where $f_v(X, Y)$ is a binary feature represented by a first-order logical formula. A hard constraint is the special case when $w_v = \infty$ (in practice, it suffices to set it to a large number, e.g., 10). In prior use of virtual evidence, w_v's are generally pre-determined from prior knowledge. However, this may be suboptimal. Therefore, we consider a general Bayesian learning setting where each w_v is drawn from a pre-specified prior distribution $w_v \sim P(w_v|\alpha_v)$. Fixed w_v amounts to the special case when the prior is concentrated on the preset value. For uncertain w_v's, we can compute their maximum a posteriori (MAP) estimates and/or quantify the uncertainty.

Distant supervision Virtual evidence for distant supervision is similar to that for direct supervision. For example, for relation extraction, distant supervision from a knowledge base of known relations will set $f_{KB}(X_i, Y_i) = \mathbb{I}[\texttt{In-KB}(X_i, r) \wedge Y_i = r]$, where $\texttt{In-KB}(X_i, r)$ is true iff the entity tuple in X_i is known to have relation r in the KB.

Data programming Virtual evidence for data programming is similar to that for distant supervision: $f_L(X_i, Y_i) = \mathbb{I}[L(X_i) = Y_i]$, where $L(X_i)$ is a labeling function provided by domain experts. Labeling functions are usually high-precision rules, but errors are still common, and different functions may assign conflicting labels to an instance. Existing denoising strategy assumes that each function makes random errors independently, and resolves the conflicts by weighted votes (Ratner et al., 2016). In DPL, this can be done by simply treating error probabilities as uncertain parameters and inferring them during learning.

Joint inference Constraints on instances or model expectations can be imposed by introducing the corresponding virtual evidence (Ganchev et al., 2010) (Proposition 2.1). The weights can be set heuristically (Chang et al., 2007; Mann and McCallum, 2008; Poon and Domingos, 2008) or iteratively via primal-dual methods (Ganchev et al., 2010). In addition to instance-level constraints, DPL can incorporate arbitrary high-order soft and hard constraints that capture the interdependencies among multiple instances. For example, identical mentions in proximity probably refer to the same entity, which is useful for resolving ambiguous mentions by leveraging their unambiguous coreferences (e.g., an acronym in apposition of the full name). This can be represented by the virtual evidence $f_{\texttt{Joint}}(X_i, Y_i, X_j, Y_j) = \mathbb{I}[\texttt{Coref}(X_i, X_j) \wedge Y_i = Y_j]$, where $\texttt{Coref}(X_i, X_j)$ is true iff X_i and X_j are coreferences. Similarly, the common denoising strategy for distant supervision replaces the mention-level constraints with type-level constraints (Hoffmann et al., 2011). Suppose that $X_E \subset X$ contains all X_i's with co-occurring entity tuple E. The new constraints simply impose that, for each E with known relation $r \in KB$, $Y_i = r$ for at least one $X_i \in X_E$. This can be represented by a high-order factor on $(X_i, Y_i : X_i \in X_E)$.

Algorithm 1 DPL Learning

```
**Input:** Virtual evidences $K = \Phi_{1:V}$, deep neural network $\Psi$, inputs $X = (X_1, \cdots, X_N)$, unobserved outputs $Y = (Y_1, \cdots, Y_N)$.
**Output:** Learned prediction module $\Psi^*$
**Initialize:** $\Phi^0 \sim$ priors, $\Psi^0 \sim$ uniform.
for $t = 1 : T$ do
    $q^t(Y) \leftarrow \arg\min_q D_{KL}(\prod_i q_i(Y_i) \,||\, \prod_v \Phi_v^{t-1}(X, Y) \cdot \prod_i \Psi^{t-1}(X_i, Y_i))$
    $\Phi^t \leftarrow \arg\min_\Phi D_{KL}(q^t(Y) \,||\, \prod_v \Phi_v(X, Y))$
    $\Psi^t \leftarrow \arg\min_\Psi D_{KL}(q^t(Y) \,||\, \prod_i \Psi(X_i, Y_i))$
end for
return $\Psi^* = \Psi^T$.
```

Parameter learning Learning in DPL maximizes the conditional likelihood of virtual evidences $P(K|X)$. We can directly optimize this objective by summing out latent Y to compute the gradient and run backpropagation. In this paper, however, we opted for a modular approach using variational EM. See Algorithm 1.

In the E-step, we compute a variational approximation $q(Y) = \prod_i q_i(Y_i)$ by minimizing its KL divergence with $P(Y|K, X)$, which amounts to computing marginal probabilities $q_i(Y_i) = P(Y_i|K, X) = \sum_{Y_{-i}} P(Y_i, Y_{-i}|K, X)$, with current parameters Φ, Ψ. This is a standard probabilistic inference problem. Exact inference is generally intractable, but there are a plethora of approximate inference methods that can efficiently produce an estimate. We use loopy belief propagation (Murphy et al., 1999) in this paper, by conducting message passing in $P(K, Y|X)$ iteratively. Note that this inference problem is considerably simpler than end-to-end inference with probabilistic logic, since the bulk of the computation is encapsulated by Ψ.

Inference with high-order factors of large size can be challenging, but there is a rich body of literature for handling such structured factors in a principled way. In particular, in distant supervision denoising, we alter the message passing schedule so that each at-least-one factor will compute messages to its variables jointly by renormalizing their current marginal probabilities with noisy-or (Keith et al., 2017), which is essentially a soft version of dual decomposition (CarøE and Schultz, 1999).

In the M-step, we treat the variational approximation $q_i(Y_i)$ as probabilistic labels, and use them to optimize Φ and Ψ via standard supervised learning, which is equivalent to minimizing the KL divergence between the probabilistic labels and the conditional likelihood of Y given X under the supervision module (Φ) and prediction module (Ψ), respectively. For the prediction module, this optimization reduces to standard deep learning. Likewise, for the supervision module, this optimization reduces to standard parameter learning for log-linear models (i.e., learning all w_v's that are not fixed). Given the probabilistic labels, it is a convex optimization problem with a unique global optimum. Here, we simply use gradient descent, with the partial derivative for w_v being $\mathbb{E}_{\Phi(Y,X)}\,[f_v(X, Y)] - \mathbb{E}_{q(Y)}\,[f_v(X, Y)]$. For a tied weight, the partial derivative will sum over all features that originate from the same template. The second expectation can be done by simple counting. The first expectation, on the other hand, requires probabilistic inference in the graphical model. But it can be computed using belief propagation, similar to the E-step, except that the messages are limited to factors within the supervision module (i.e., messages from Ψ are not longer included). Convergence is usually fast, upon which the marginal for each Y_i is available, and $\mathbb{E}_{\Phi(Y,X)}\,[f_v(X, Y)]$ is simply the fraction of Y that renders $f_v(X, Y)$ to be true. Again, this parameter learning problem is much simpler than end-to-end learning with probabilistic logic, as it focuses on refining uncertain weights for indirect supervision, rather than learning complex input patterns for label prediction (handled in deep learning).

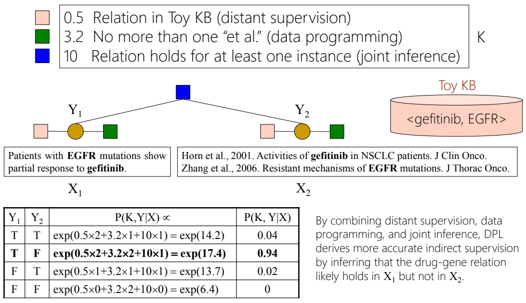

Y_1	Y_2	$P(K,Y\|X) \propto$	$P(K, Y\|X)$
T	T	$\exp(0.5\times2+3.2\times1+10\times1) = \exp(14.2)$	0.04
T	**F**	$\mathbf{\exp(0.5\times2+3.2\times2+10\times1) = \exp(17.4)}$	**0.94**
F	T	$\exp(0.5\times1+3.2\times1+10\times1) = \exp(13.7)$	0.02
F	F	$\exp(0.5\times0+3.2\times2+10\times0) = \exp(6.4)$	0

By combining distant supervision, data programming, and joint inference, DPL derives more accurate indirect supervision by inferring that the drug-gene relation likely holds in X_1 but not in X_2.

Figure 3: Example of DPL combining various indirect supervision using probabilistic logic. The prediction module is omitted to avoid clutter.

Example Figure 3 shows a toy example on how DPL combines various indirect supervision for predicting drug-gene interaction (e.g., gefitinib can be used to treat tumors with EGFR mutations). Indirect supervision is modeled by probabilistic logic, which defines a joint probability distribution over latent labeling decisions for drug-gene mention pairs in unlabeled text. Here, distant supervision prefers classifying mention pairs of known relations, whereas the data programming formula opposes classifying instances resembling citations, and the joint inference formula ensures that at least one mention pair of a known relation is classified as positive. Formula weight signifies the confidence in the indirect supervision, and can be refined iteratively along with the prediction module.

Handling label imbalance One challenge for distant supervision is that negative examples are often much more numerous. A common strategy is to subsample negative examples to attain a balanced dataset. In preliminary experiments, we found that this was often suboptimal, as many informative negative examples were excluded from training. Instead, we restored the balance by up-weighting positive examples. In DPL, an additional challenge is that the labels are probabilistic and change over iterations. In this paper, we simply used hard EM, with binary labels set using 0.5 as the probability threshold, and the up-weighting coefficient recalculated after each E-step.

4 Biomedical Machine Reading

There is a long-standing interest in biomedical machine reading (e.g., Morgan et al. (2008); Kim et al. (2009)), but prior studies focused on supervised approaches. The advent of big biomedical data creates additional urgency for developing scalable approaches that can generalize to new reading tasks. For example, genome sequencing cost has been dropping faster than Moore's Law, yet oncologists can only evaluate tumor sequences for a tiny fraction of patients, due to the bottleneck in assimilating relevant knowledge from publications. Recently, Peng et al. (2017) formulated precision oncology machine reading as cross-sentence relation extraction (Figure 2) and developed the state-of-the-art system using distant supervision. While promising, their results still leave much room to improve. Moreover, they used heuristics to heavily filter entity candidates, with significant recall loss.

In this section, we use cross-sentence relation extraction as a case study for combining indirect supervision using deep probabilistic logic (DPL). First, we show that DPL can substantially improve machine reading accuracy in a head-to-head comparison with Peng et al. (2017), using the same entity linking method. Next, we apply DPL to entity linking itself and attain similar improvement. Finally, we consider further improving the recall by removing the entity filter. By applying DPL to joint entity linking and relation extraction, we more than doubled the recall in relation extraction while attaining comparable precision as Peng et al. (2017) with heavy entity filtering.

Evaluation Comparing indirect supervision methods is challenging as there is often no annotated test set for evaluating precision and recall. In such cases, we resort to the standard strategy used in prior work by reporting *sample precision* (estimated proportion of correct system extractions) and *absolute recall* (estimated number of correct system extractions). Absolute recall is proportional to recall and can be used to compare different systems (modulo estimation errors).

Datasets We used the same unlabeled text as Peng et al. (2017), which consists of about one million full text articles in PubMed Central (PMC)[3]. Tokenization, part-of-speech tagging, and syntactic parsing were conducted using SPLAT (Quirk et al., 2012), and Stanford dependencies (de Marneffe et al., 2006) were obtained using Stanford CoreNLP (Manning et al., 2014). For entity ontologies, we used DrugBank[4] and Human Gene Ontology (HUGO)[5]. DrugBank contains 8257 drugs; we used the subset of 599 cancer drugs. HUGO contains 37661 genes. For knowledge bases, we used the Gene Drug Knowledge Database (GDKD) (Dienstmann et al., 2015) and the Clinical Interpretations of Variants In Cancer (CIVIC)[6]. Together, they contain 231 drug-gene-mutation triples, with 76 drugs, 35 genes and 123 mutations.

4.1 Cross-sentence relation extraction

Let $e_1, \cdots, e_m$ be entity mentions in text T. Relation extraction can be formulated as classifying whether a relation R holds for $e_1, \cdots, e_m$ in T. To enable a head-to-head comparison, we used the same cross-sentence setting as Peng et al. (2017), where T spans up to three consecutive sentences and R represents the ternary interaction over drugs, genes, and mutations (whether the drug is relevant for treating tumors with the given gene mutation).

Entity linking In this subsection, we used the entity linker from Literome (Poon et al., 2014) to identify drug, gene, and mutation mentions, as in Peng et al. (2017). This entity linker first identifies candidate mentions by matching entity names

---

[3] `www.ncbi.nlm.nih.gov/pmc`

[4] `www.drugbank.ca`

[5] `www.genenames.org`

[6] `civic.genome.wustl.edu`

Distant Supervision: GDKD, CIVIC
Data Programming (Entity) Mention matches entity name exactly. Mention not a stop word. Mention not following figure designation. Mention's POS tags indicate it is a noun.
Data Programming (Relation) Less than 30% of words are numbers in each sentence. No more than three consecutive numbers. No more than two "et al". No more than three tokens start with uppercase. No more than three special characters. No more than three keywords indicative of table or figure. Entity mentions do not overlap.
Joint Inference: Relation holds in at least one instance.

Table 1: DPL combines three indirect supervision strategies for cross-sentence relation extraction

or synonyms in domain ontologies, then applies heuristics to filter candidates. The heuristics are designed to enhance precision, at the expense of recall. For example, one heuristics would filter candidates of length less than four, which eliminates key cancer genes such as ER or AKT.

Prediction module We used the same graph LSTM as in Peng et al. (2017) to enable head-to-head comparison on indirect supervision strategies. Briefly, a graph LSTM generalizes a linear-chain LSTM by incorporating arbitrary long-ranged dependencies, such as syntactic dependencies, discourse relations, coreference, and connections between roots of adjacent sentences. A word might have precedents other than the prior word, and its LSTM unit is expanded to include a forget gate for each precedent. See Peng et al. (2017) for details.

Supervision module We used DPL to combine three indirect supervision strategies for cross-sentence relation extraction (Table 1). For distant supervision, we used GDKD and CIVIC as in Peng et al. (2017). For data programming, we introduced labeling functions that aim to correct entity and relation errors. Finally, we incorporated joint inference among all co-occurring instances of an entity tuple with the known relation by imposing the at-least-one constraint (i.e., the relation holds for at least one of the instances). For development, we sampled 250 positive extractions from DPL using only distant supervision (Peng et al., 2017) and excluded them from future training and evaluation.

Experiment results We compared DPL with the state-of-the-art system of Peng et al. (2017). We also conducted ablation study to evaluate the impact of indirect-supervision strategies. For a fair comparison, we used the same probability threshold in all cases (an instance is classified as positive if the normalized probability score is at least 0.5). For each system, sample precision was estimated by sampling 100 positive extractions and manually determining the proportion of correct extractions by an author knowledgeable about this domain. Absolute recall is estimated by multiplying sample precision with the number of positive extractions.

System	Prec.	Abs. Rec.	Unique
Peng 2017	0.64	6768	2738
DPL + `EMB`	**0.74**	**8478**	**4821**
DPL	0.73	7666	4144
− `DS`	0.29	7555	4912
− `DP`	0.67	4826	2629
− `DP (ENTITY)`	0.70	7638	4074
− `JI`	0.72	7418	4011

Table 2: Comparison of sample precision and absolute recall (all instances and unique entity tuples) in test extraction on PMC. DPL + `EMB` is our full system using PubMed-trained word embedding, whereas DPL uses the original Wikipedia-trained word embedding in Peng et al. (2017). Ablation: DS (distant supervision), DP (data programming), JI (joint inference).

Pred. Mod.	Prec.	Abs. Rec.	Unique
BiLSTM	0.60	6243	3427
Graph LSTM	0.73	7666	4144

Table 3: Comparison of sample precision and absolute recall (all instances and unique entity tuples) in test extraction on PMC. Both use same indirect supervision and Wikipedia-trained word embedding.

Table 2 shows the results. DPL substantially outperformed Peng et al. (2017), improving sample precision by ten absolute points and raising absolute recall by 25%. Combining disparate indirect supervision strategies is key to this performance gain, as evident from the ablation results. While distant supervision remained the most potent source of indirect supervision, data programming and joint inference each contributed significantly. Replacing out-of-domain (Wikipedia) word embedding with in-domain (PubMed) word embedding (Pyysalo et al., 2013) also led to a small gain.

Peng et al. (2017) only compared graph LSTM and linear-chain LSTM in automatic evaluation, where distant-supervision labels were treated as ground truth. They found significant but relatively small gains by graph LSTM. We conducted additional manual evaluation comparing the two in

Distant Supervision: HGNC
Data Programming No verbs in POS tags. Mention not a common word. Mention contains more than two characters or one word. More than 30% of characters are upper case. Mention contains both upper and lower case characters. Mention contains both character and digit. Mention contains more than six characters. Dependency label from mention to parent indicative of direct object.
Joint Inference Identical mentions nearby probably refer to the same entity. Appositive mentions probably refer to the same entity. Nearby mentions that match synonyms of same entity probably refer to the given entity.

Table 4: DPL combines three indirect supervision strategies for entity linking.

DPL. Surprisingly, we found rather large performance difference, with graph LSTM outperforming linear-chain LSTM by 13 absolute points in precision and raising absolute recall by over 20% (Table 3). This suggests that Peng et al. (2017) might have underestimated the performance gain by graph LSTM using automatic evaluation.

4.2 Entity linking

Let m be a mention in text and e be an entity in an ontology. The goal of entity linking is to predict $\texttt{Link}(\texttt{m}, \texttt{e})$, which is true iff m refers to e, for every candidate mention-entity pair m, e. We focus on genes in this paper, as they are particularly noisy.

Prediction module We used BiLSTM with attention over the ten-word windows before and after a mention. The embedding layer is initialized by word2vec embedding trained on PubMed abstracts and full text (Pyysalo et al., 2013). The word embedding dimension was 200. We used 5 epochs for training, with Adam as the optimizer. We set learning rate to 0.001, and batch size to 64.

Supervision module As in relation extraction, we combined three indirect supervision strategies using DPL (Table 4). For distant supervision, we obtained all mention-gene candidates by matching PMC text against the HUGO lexicon. We then sampled a subset of 200,000 candidate instances as positive examples. We sampled a similar number of noun phrases as negative examples. For data programming, we introduced labeling functions that used mention characteristics (longer names are less ambiguous) or syntactic context (genes are more likely to be direct objects and nouns). For joint inference, we leverage linguistic phenomena related to coreference (identical, appositive, or synonymous mentions nearby are likely coreferent).

System	Acc.	F1	Prec.	Rec.
String Match	0.18	0.31	0.18	1.00
DS	0.64	0.71	0.62	0.83
DS + DP	0.66	0.71	0.62	0.83
DS + DP + JI	**0.70**	**0.76**	0.68	0.86

Table 5: Comparison of gene entity linking results on a balanced test set. The string-matching baseline has low precision. By combining indirect supervision strategies, DPL substantially improved precision while retaining reasonably high recall.

	F1	Precision	Recall
GNormPlus	0.78	0.74	0.81
DPL	0.74	0.68	0.80

Table 6: Comparison of gene entity linking results on BioCreative II test set. GNormPlus is the state-of-the-art system trained on thousands of labeled examples. DPL used only indirect supervision.

Experiment results For evaluation, we annotated a larger set of sample gene-mention candidates and then subsampled a balanced test set of 550 instances (half are true gene mentions, half not). These instances were excluded from training and development. Table 5 compares system performance on this test set. The string-matching baseline has a very low precision, as gene mentions are highly ambiguous, which explains why Peng et al. (2017) resorted to heavy filtering. By combining indirect supervision strategies, DPL improved precision by over 50 absolute points, while retaining a reasonably high recall (86%). All indirect supervision strategies contributed significantly, as the ablation tests show. We also evaluated DPL on BioCreative II, a shared task on gene entity linking (Morgan et al., 2008). We compared DPL with GNormPlus (Wei et al., 2015), the state-of-the-art supervised system trained on thousands of labeled examples in BioCreative II training set. Despite using zero manually labeled examples, DPL attained comparable F1 and recall (Table 6). The difference is mainly in precision, which indicates opportunities for more indirect supervision.

4.3 Joint entity and relation extraction

An important use case for machine reading is to improve knowledge curation efficiency by offering extraction results as candidates for curators to vet. The key to practical adoption is attaining high recall with reasonable precision (Peng et al., 2017). The entity filter used in Peng et al. (2017) is not ideal in this aspect, as it substantially reduced recall. In this

System	Prec	Abs. Rec.	Unique
Peng 2017	0.31	11481	5447
DPL (RE)	0.52	17891	8534
+ EL (TRN)	0.55	**21881**	**11047**
+ EL (TRN/TST)	**0.61**	20378	10291

Table 7: Comparison of sample precision and absolute recall (all instances and unique entity tuples) when all gene mention candidates are considered. Peng et al. (2017) used distant supervision only. RE: DPL relation extraction. EL: using DPL entity linking in RE training (TRN) and/or test (TST).

Gene	Drug	Mut.	Gene-Mut.	Relation
27%	4%	20%	45%	24%

Table 8: Error analysis for DPL relation extraction.

subsection, we consider replacing the entity filter by the DPL entity linker Table 7. Specifically, we added one labeling function to check if the entity linker returns a normalized probability score above $p_{\texttt{TRN}}$ for gene mentions, and filtered test instances if the gene mention score is lower than $p_{\texttt{TST}}$. We set $p_{\texttt{TRN}} = 0.6$ and $p_{\texttt{TST}} = 0.3$ from preliminary experiments. The labeling function discouraged learning from noisy mentions, and the test-time filter skips an instance if the gene is likely wrong. Not surprisingly, without entity filtering, Peng et al. (2017) suffered large precision loss. All DPL versions substantially improved accuracy, with significantly more gains using the DPL entity linker.

4.4 Discussion

Scalability DPL is efficient to train, taking around 3.5 hours for relation extraction and 2.5 hours for entity linking in our PubMed-scale experiments, with 25 CPU cores (for probabilistic logic) and one GPU (for LSTM). For relation extraction, the graphical model of probabilistic logic contains around 7,000 variables and 70,000 factors. At test time, it is just an LSTM, which predicted each instance in less than a second. In general, DPL learning scales linearly in the number of training instances. For distant supervision and data programming, DPL scales linearly in the number of known facts and labeling functions. As discussed in Section 3, joint inference with high-order factors is more challenging, but can be efficiently approximated. For inference in probabilistic logic, we found that loopy belief propagation worked reasonably well, converging after 2-4 iterations. Overall, we ran variational EM for three iterations, using ten epochs of deep learning in each M-step. We found these worked well in preliminary experiments and used the same setting in all final experiments.

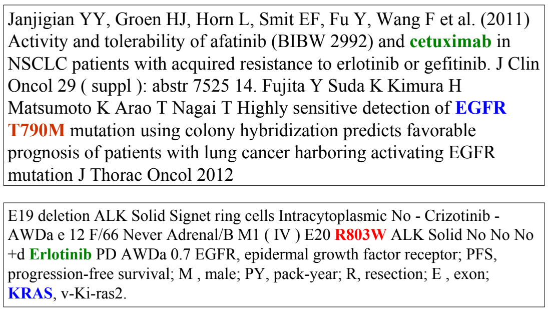
Janjigian YY, Groen HJ, Horn L, Smit EF, Fu Y, Wang F et al. (2011) Activity and tolerability of afatinib (BIBW 2992) and **cetuximab** in NSCLC patients with acquired resistance to erlotinib or gefitinib. J Clin Oncol 29 (suppl): abstr 7525 14. Fujita Y Suda K Kimura H Matsumoto K Arao T Nagai T Highly sensitive detection of **EGFR** **T790M** mutation using colony hybridization predicts favorable prognosis of patients with lung cancer harboring activating EGFR mutation J Thorac Oncol 2012

E19 deletion ALK Solid Signet ring cells Intracytoplasmic No - Crizotinib - AWDa e 12 F/66 Never Adrenal/B M1 (IV) E20 **R803W** ALK Solid No No No +d **Erlotinib** PD AWDa 0.7 EGFR, epidermal growth factor receptor; PFS, progression-free survival; M , male; PY, pack-year; R, resection; E , exon; **KRAS**, v-Ki-ras2.

Figure 4: Example of relation-extraction errors corrected by DPL with additional indirect supervision.

Accuracy To understand more about DPL's performance gain over distant supervision, we manually inspected some relation-extraction errors fixed by DPL after training with additional indirect supervision. Figure 4 shows two such examples. While some data programming functions were introduced to prevent errors stemming from citations or flattened tables, none were directly applicable to these examples. This shows that DPL can generalize beyond the original indirect supervision.

While the results are promising, there is still much to improve. Table 8 shows estimated precision errors for relation extraction by DPL. (Some instances have multiple errors.) Entity linking can incorporate more indirect supervision. Joint entity linking and relation extraction can be improved by feeding back extraction results to linking. Improvement is also sorely needed in classifying mutations and gene-mutation associations. The prediction module can also be improved, e.g., by adding attention to graph LSTM. DPL offers a flexible framework for exploring all these directions.

5 Conclusion

We introduce DPL as a unifying framework for indirect supervision, by composing probabilistic logic with deep learning. Experiments on biomedical machine reading show that this enables novel combination of disparate indirect supervision methodologies, resulting in substantial gain in accuracy. Future directions include: combining DPL with deep generative models; exploring alternative optimization strategies; applications to other domains.

6 Acknowledgements

We thank David McAllester, Chris Quirk, and Scott Yih for useful discussions, and the three anonymous reviewers for helpful comments.

References

Stephen H Bach, Bryan He, Alexander Ratner, and Christopher Ré. 2017. Learning the structure of generative models without labeled data. In *International Conference on Machine Learning*, pages 273–282.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Claus C CarøE and Rüdiger Schultz. 1999. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 280–287.

Kevin Clark and Christopher D Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 643–653.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*.

Rodrigo Dienstmann, In Sock Jang, Brian Bot, Stephen Friend, and Justin Guinney. 2015. Database of genomic biomarkers for cancer drugs and clinical targetability in solid tumors. *Cancer Discovery*, 5.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602. ACM.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the Forty-Ninth Annual Meeting of the Association for Computational Linguistics*.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016a. Harnessing deep neural networks with logic rules. In *Proceedings of the 2016 Conference on Association for Computational Linguistics*.

Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric Xing. 2016b. Deep neural networks with massive learned knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1670–1679.

Hongzhao Huang, Larry Heck, and Heng Ji. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *arXiv preprint arXiv:1504.07678*.

Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. 2016. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pages 2946–2954.

Katherine Keith, Abram Handler, Michael Pinkham, Cara Magliozzi, Joshua McDuffie, and Brendan O'Connor. 2017. Identifying civilians killed by police with distantly supervised entity-event extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1547–1557.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9. Association for Computational Linguistics.

Angelika Kimmig, Stephen Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2012. A short introduction to probabilistic soft logic. In *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*, pages 1–4.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Xiao Li. 2009. On the use of virtual evidence in conditional random fields. In *Proceedings of the 2009*

Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3, pages 1289–1297. Association for Computational Linguistics.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 2124–2133.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 46–56.

Gideon S Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. *Proceedings of ACL-08: HLT*, pages 870–878.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the Fifty-Second Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the Forty-Seventh Annual Meeting of the Association for Computational Linguistics and the Fourth International Joint Conference on Natural Language Processing*.

Alexander A Morgan, Zhiyong Lu, Xinglong Wang, Aaron M Cohen, Juliane Fluck, Patrick Ruch, Anna Divoli, Katrin Fundel, Robert Leaman, Jörg Hakenberg, et al. 2008. Overview of biocreative ii gene normalization. *Genome biology*, 9(2):S3.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc.

Ankur P Parikh, Hoifung Poon, and Kristina Toutanova. 2015. Grounded semantic parsing for complex knowledge extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 756–766.

Judea Pearl. 2014. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

Nanyun Peng, Hoifung Poon, Chris Quirk, and Kristina Toutanova Wen tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Hoifung Poon and Pedro Domingos. 2007. Joint inference in information extraction. In *AAAI*, volume 7, pages 913–918.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the conference on empirical methods in natural language processing*, pages 650–659. Association for Computational Linguistics.

Hoifung Poon, Chris Quirk, Charlie DeZiel, and David Heckerman. 2014. Literome: PubMed-scale genomic knowledge base in the cloud. *Bioinformatics*, 30(19).

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 813–821. Association for Computational Linguistics.

S. Pyysalo, F. Ginter, H. Moen, T. Salakoski, and S. Ananiadou. 2013. Distributional semantics resources for biomedical text processing. In *Proceedings of LBM 2013*, pages 39–44.

Chris Quirk, Pallavi Choudhury, Jianfeng Gao, Hisami Suzuki, Kristina Toutanova, Michael Gamon, Wentau Yih, and Lucy Vanderwende. 2012. MSR SPLAT, a language analysis toolkit. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics:Demonstration Session*.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the Fifteenth Conference on European chapter of the Association for Computational Linguistics*.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575.

Sheila M Reynolds and Jeff A Bilmes. 2005. Part-of-speech tagging using virtual evidence and negative training. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 459–466. Association for Computational Linguistics.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Amarnag Subramanya and Jeff Bilmes. 2007. Virtual evidence for training speech recognizers using partially labeled data. In *The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 165–168. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Geoffrey G Towell and Jude W Shavlik. 1994. Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165.

Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. 2015. Gnormplus: an integrative approach for tagging genes, gene families, and protein domains. *BioMed research international*, 2015.

Deriving Machine Attention from Human Rationales

Yujia Bao[1], **Shiyu Chang**[2], **Mo Yu**[2], **Regina Barzilay**[1]
[1]Computer Science and Artificial Intelligence Lab, MIT
[2]MIT-IBM Watson AI Lab, IBM Research
{yujia, regina}@csail.mit.edu, shiyu.chang@ibm.com, yum@us.ibm.com

Abstract

Attention-based models are successful when trained on large amounts of data. In this paper, we demonstrate that even in the low-resource scenario, attention can be learned effectively. To this end, we start with discrete human-annotated rationales and map them into continuous attention. Our central hypothesis is that this mapping is general across domains, and thus can be transferred from resource-rich domains to low-resource ones. Our model jointly learns a domain-invariant representation and induces the desired mapping between rationales and attention. Our empirical results validate this hypothesis and show that our approach delivers significant gains over state-of-the-art baselines, yielding over 15% average error reduction on benchmark datasets.[1]

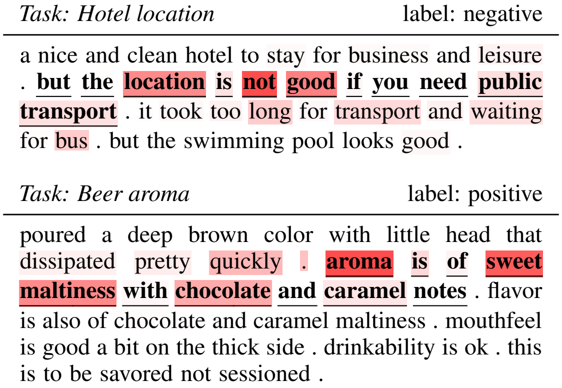

Task: Hotel location label: negative

a nice and clean hotel to stay for business and leisure . **but the location is not good if you need public transport** . it took too long for transport and waiting for bus . but the swimming pool looks good .

Task: Beer aroma label: positive

poured a deep brown color with little head that dissipated pretty quickly . **aroma is of sweet maltiness with chocolate and caramel notes** . flavor is also of chocolate and caramel maltiness . mouthfeel is good a bit on the thick side . drinkability is ok . this is to be savored not sessioned .

Figure 1: Examples of rationales versus oracle attention. Words are highlighted according to their relative attention scores. Human rationales are shown in bold with underlines.

1 Introduction

Attention-based models have become architectures of choice for many NLP tasks. In addition to significant performance gains, these models are attractive, as attention is often used as a proxy for human interpretable rationales. Their success, however, is conditioned on access to large amounts of training data. To make these models applicable in low-resource scenarios, we utilize this connection in the opposite direction. Specifically, we propose an approach to map human rationales to high-performing attention, and use this attention to guide models trained in low-resource scenarios.

The notions of rationale and attention are closely related. Both of them highlight word importance for the final prediction. In the case of rationale, the importance is expressed as a hard selection, while attention provides a soft distribution over the words. Figure 1 illustrates this relatedness. One obvious approach to improve low-resource performance is to directly use human rationales as a supervision for attention generation. The implicit assumption behind this method is that machine-generated attention should mimic human rationales. However, rationales on their own are not adequate substitutes for machine attention. Instead of providing a soft distribution, human rationales only provide the binary indication about relevance. Furthermore, rationales are subjectively defined and often vary across annotators. Finally, human rationales are not customized for a given model architecture. In contrast, machine attention is always derived as a part of a specific model architecture.

To further understand this connection, we empirically compare models informed by human rationales and those by high-quality attention. To obtain the latter, we derive an "oracle" attention using a large amount of annotations. This "oracle" attention is then used to guide a model that only has access to a small subset of this training data. Not only does this model outperform the oracle-free variant, but it also yields substantial gains over its counterpart trained with human ra-

---

[1] Our code and data are available at `https://github.com/YujiaBao/R2A`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1903–1913
Brussels, Belgium, October 31 - November 4, 2018.

tionales — 89.98 % vs 85.22 % average accuracy on three aspects of hotel review (see Section 4 for details). In practice, however, this "oracle" attention is not available. To employ this method, we need to find a way to obtain a substitute for the "oracle" attention.

In this paper, we show how to achieve this goal using rationales. Specifically, we learn a mapping from human rationales to high-quality attention (R2A). We hypothesize that this mapping is generalizable across tasks and thus can be transferred from resource-rich tasks.[2] Figure 1 illustrates that in both tasks, attention weighs rationale words in a similar fashion: highlighting task-specific nouns and adjectives, while downplaying functional words. To learn and apply this mapping we need access to rationales in both source and target tasks. In the target task, we assume rationales are provided by humans. In the source task(s), collecting rationales at scale is infeasible. Therefore, we use machine-generated rationales (Lei et al., 2016) as a proxy.

Our R2A model consists of three components. The first one is an attention-based model for the source task(s) that provides supervision for attention generation. The second component focuses on learning a domain-invariant representation to support transfer. The third component combines this invariant representation and rationales together to generate the attention. These three components are trained jointly to optimize the overall objective. Once the model is trained, we apply it to the target task to generate attention from human rationales. This attention is consequently used to supervise the training of the target classifier.

We evaluate our approach on two transfer settings: aspect transfer within single domain and domain transfer across multiple domains. Our experiments demonstrate that our approach delivers significant performance improvements over the baselines. For instance, the average error reduction over the best baseline in domain transfer is over 15%. In addition, both qualitative and quantitative analyses confirm that our R2A model is capable of generating high-quality attention for target tasks.

2 Related Work

Attention-based models Attention has been shown to be effective when the model is trained on large amounts of training data (Bahdanau et al., 2014; Luong et al., 2015; Rush et al., 2015; Yang et al., 2016; Lin et al., 2017; Chen et al., 2017; Vaswani et al., 2017). In this setting, typically no additional supervision is required for learning the attention. Nevertheless, further refining attention by extra supervision has been shown to be beneficial. Examples include using word alignments to learn attention in neural machine translation (Liu et al., 2016), employing argument words to supervise attention in event detection (Liu et al., 2017), utilizing linguistically-motivated annotations to guide attention in constituency parsing (Kamigaito et al., 2017). These supervision mechanisms are tailored to specific applications. In contrast, our approach is based on the connection between rationales and attention, and can be used for multiple applications.

Rationale-based models Zaidan et al. (2007) was the first to explore the value of rationales in low-resource scenarios. They hypothesize that the model confidence should decrease when the rationale words are removed from the inputs, and validate this idea for linear models. Recent work (Zhang et al., 2016) explores the potential of integrating rationales with more complex neural classifiers. In their model, human rationales are directly used to guide the sentence-level attention for a CNN-based classifier. To reach good performance, their model still requires a sufficient amount of training data. Our work differs from theirs as we discern the intrinsic difference between human rationales and machine attention. Moreover, we learn a model to map human rationales into high-quality attention so as to provide a richer supervision for low-resource models.

Transfer learning When labeled data on the target task is available, existing approaches typically transfer the knowledge by either fine-tuning an encoder trained on the source tasks(s) (Conneau et al., 2017; Peters et al., 2018) or multi-task learning on all tasks with a shared encoder (Collobert et al., 2011). In this paper, we explore the transferability of the task-specific attention through human rationales. We believe this will further assist learning in low-resource scenarios.

Our work is also related to unsupervised domain

---

[2] In this paper, we consider a more general setting where one domain contains multiple tasks. Also, we assume having one source domain. However, our proposed method is a general framework and can be easily adapted to problems with multiple source domains.

adaptation, as the R2A model has never seen any target annotations during training. Existing methods commonly adapt the classifier by aligning the representations between the source and target domains (Glorot et al., 2011; Chen et al., 2012; Zhou et al., 2016; Ganin et al., 2016; Zhang et al., 2017). In contrast, our model adapts the mapping from rationales to attention; thus after training, it can be applied to different target tasks.

3 Method

Problem formulation We assume that we have N source tasks $\{\mathcal{S}_i\}_{i=1}^N$, where each of them has sufficient amounts of labeled examples. Using existing methods (Lei et al., 2016), we can generate rationales for each source example automatically (see Appendix 1 for details). In the target task $\mathcal{T}$, we only have a limited amount of labeled examples with large amounts of unlabeled data. For those labeled examples, we assume access to human-annotated rationales.

Overview Our goal is to improve classification performance on the target task by learning a mapping from human rationales to high-quality machine attention (R2A). Given the scarcity of our target data, we learn this mapping on resource rich tasks where high-quality attention can be readily obtained during training. Next, the mapping between rationales and attention derived from the source tasks is exported into the target task. To enable this transfer, models have to operate over an invariant representation which we construct via an adversarial objective. Once the mapping is derived, we can translate human rationales in the target task into high-quality attention. This generated attention is then used to provide additional training signal for an attention-based classifier for the target task. The overall pipeline is shown in Figure 2.

Alternatively, we can view the R2A mapping as a meta model that produces a prior over the attention distribution across different tasks.

Model architecture Figure 3 illustrates the architecture of our R2A model, which consists of three components.

- **Multi-task learning** In order to learn the R2A mapping, we need annotation for the attention. This module generates high-quality attention as an intermediate result by minimizing the prediction error on the source tasks (Section 3.1).

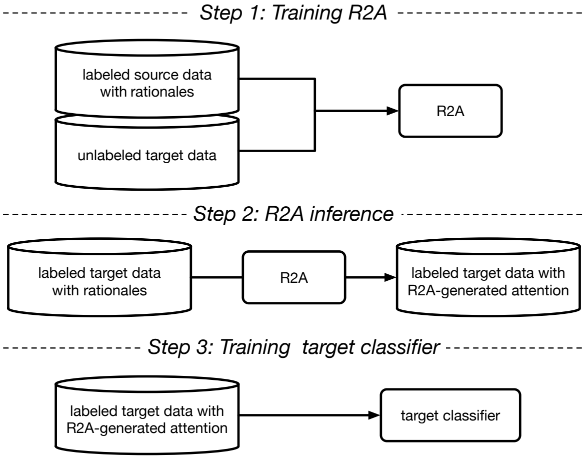

Figure 2: Overall pipeline of our approach (Section 3.4). The R2A mapping is learned from labeled source data and unlabeled target data. Then we applied it to the target task to derive attention based on human rationales. Finally, a target classifier is trained under the supervision of both the annotated labels and the R2A-generated attention.

- **Domain-invariant encoder** This module aims to transform the contextualized representation obtained from the first module into a domain-invariant version. We achieve this goal through domain adversarial training over the source data and the unlabeled target data (Section 3.2).
- **Attention generation** This module learns to predict the intermediate attention obtained from the first module based on the domain-invariant representation and the rationales (Section 3.3).

3.1 Multi-task learning

The goal of the multi-task learning module is to learn good attention for each source task. This learned attention will be used later to supervise the attention generation module. This module takes the input text from the source tasks and predicts the labels. To accomplish the previously stated goal, we minimize the prediction error over all labeled source data.

Let (x^t, y^t) be a training instance from any source task $t \in \{\mathcal{S}_1, \ldots \mathcal{S}_N\}$. We first encode the input sequence x^t into hidden states: $h^t = \mathbf{enc}(x^t)$, where $\mathbf{enc}$ is a bi-directional LSTM (Hochreiter and Schmidhuber, 1997) that is shared across all source tasks. For each position i, the dense vector h_i^t encodes the content and context information of the word x_i^t. We then pass the

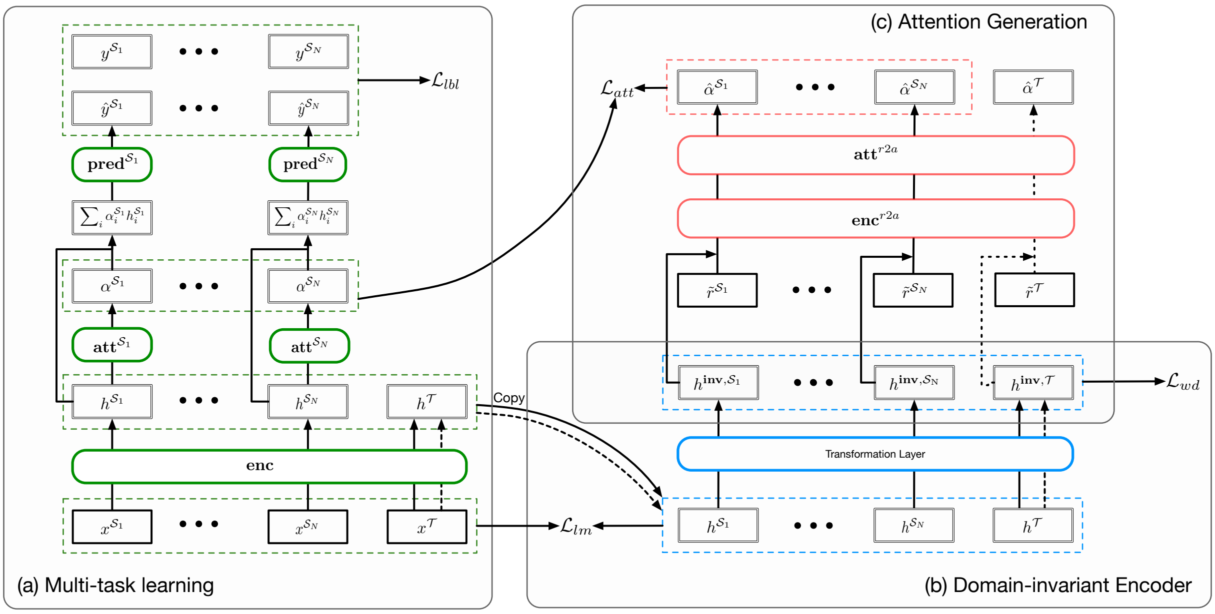

Figure 3: Architecture of the R2A model. The model is comprised of (a) a multi-task learning component, (b) a domain-invariant encoder, and (c) an attention generation component. Solid arrows denote computations for training, while dotted arrows denote computations for inference.

sequence h^t on to a task-specific attention module to produce attention $\alpha^t = \mathbf{att}^t(h^t)$ as follows:

$$\tilde{h}_i^t = \tanh(W_{\mathbf{att}}^t h_i^t + b_{\mathbf{att}}^t),$$
$$\alpha_i^t = \frac{\exp(\langle \tilde{h}_i^t, q_{\mathbf{att}}^t \rangle)}{\sum_j \exp(\langle \tilde{h}_j^t, q_{\mathbf{att}}^t \rangle)},$$

where $\langle \cdot, \cdot \rangle$ denotes inner product and $W_{\mathbf{att}}^t$, $b_{\mathbf{att}}^t$, $q_{\mathbf{att}}^t$ are learnable parameters. We predict the label of x^t using the weighted sum of its contextualized representation: $\hat{y}^t = \mathbf{pred}^t(\sum_i \alpha_i^t h_i^t)$, where $\mathbf{pred}^t$ is a task-specific multi-layer perceptron. We train this module to minimize the loss, $\mathcal{L}_{lbl}$, between the prediction and the annotated label for all source tasks. We use cross entropy loss for classification tasks and mean square loss for regression tasks.

3.2 Domain-invariant encoder

Supplied with large amounts of source data and unlabeled target data, this module has two goals: 1) learning a general encoder for both source and target corpora, and 2) learning domain-invariant representation. This module enables effective transfer—especially in the presence of significant variance between the source and target domains. We achieve the first goal by optimizing a language modeling objective and the second goal by minimizing the Wasserstein distance between the source and target distribution.

Let x be an input sequence, and $h \triangleq [\overrightarrow{h}; \overleftarrow{h}]$ be its corresponding contextualized representation obtained from $\mathbf{enc}$. Here, $\overrightarrow{h}$ and $\overleftarrow{h}$ denote the output sequence of the forward and backward LSTM, respectively. In order to support transfer, $\mathbf{enc}$ should be general enough to effectively represent both source and target corpora. For this reason, we ground the encoder by a language modeling component (Bengio et al., 2003; Mikolov et al., 2011). Specifically, we employ two Softmax classifiers to predict the word x_i based on $\overrightarrow{h}_{i-1}$ and $\overleftarrow{h}_{i+1}$ respectively. We minimize the cross-entropy loss $\mathcal{L}_{lm}$ over all source data and unlabeled target data.

The representation h is domain-specific as it is trained to encode useful features for language modeling and the source tasks. To obtain an invariant representation, we employ a transformation layer and propose to align the transformed representation so that it is not distinguishable whether it comes from the source or the target. Specifically, we transform the representation h_i at each position i linearly and obtain

$$h_i^{\mathbf{inv}} = W_{\mathbf{inv}} h_i + b_{\mathbf{inv}},$$

where $W_{\mathbf{inv}}$ and $b_{\mathbf{inv}}$ are learnable parameters. We minimize the Wasserstein distance (Arjovsky et al., 2017) between the distribution of $h^{\mathbf{inv}}$ from the source and the one from the target, denoted as $\mathbb{P}_{\mathcal{S}}$ and $\mathbb{P}_{\mathcal{T}}$, respectively. Since $h^{\mathbf{inv}}$ is a sequence of variable length, L, we summarize it by its first and last element via concatenation, $[h_1^{\mathbf{inv}}; h_L^{\mathbf{inv}}]$. The training objective is defined as:

$$\mathcal{L}_{wd} = \sup_{\|f\|_L \leq K} \mathbb{E}_{h^{\mathbf{inv}} \sim \mathbb{P}_{\mathcal{S}}} \left[f([h_1^{\mathbf{inv}}; h_L^{\mathbf{inv}}]) \right] - \mathbb{E}_{h^{\mathbf{inv}} \sim \mathbb{P}_{\mathcal{T}}} \left[f([h_1^{\mathbf{inv}}; h_L^{\mathbf{inv}}]) \right],$$

where the supremum is over all K-Lipschitz scalar functions f. Following Gulrajani et al. (2017), we approximate f by a multi-layer perceptron, and use gradient penalty to fulfill the Lipschitz constraint.

3.3 Attention generation

The goal of this module is to generate high-quality attention for each task. This module combines the domain-invariant representation together with task-specific rationales as its input and predicts task-specific attention scores. We minimize the distance between the predicted attention and the intermediate attention obtained from the multi-task learning module.

For any source task $t \in \{\mathcal{S}_i\}_{i=1}^N$, we denote r^t as the task-specific rationales corresponding to the input text x^t, and denote $h^{\mathbf{inv},t}$ as the domain-invariant representation of x^t. For each position i, we first concatenate r_i^t with the frequency of x_i^t occurring as a rationale from all training examples of this task. We denote this augmented sequence as $\tilde{r}^t$. This frequency term provides the unigram likelihood of each word being a rationale for the task. Then we employ a sequence encoder $\mathbf{enc}^{r2a}$ and an attention module $\mathbf{att}^{r2a}$ to predict the attention scores:

$$\begin{aligned} u^t &= \mathbf{enc}^{r2a}([h^{\mathbf{inv},t}; \tilde{r}^t]), \\ \tilde{u}_i^t &= \tanh(W_{\mathbf{att}}^{r2a} u_i^t + b_{\mathbf{att}}^{r2a}), \\ \hat{\alpha}_i^t &= \frac{\exp(\langle \tilde{u}_i^t, q_{\mathbf{att}}^{r2a} \rangle)}{\sum_j \exp(\langle \tilde{u}_j^t, q_{\mathbf{att}}^{r2a} \rangle)}, \end{aligned}$$

where $W_{\mathbf{att}}^{r2a}$, $b_{\mathbf{att}}^{r2a}$ and $q_{\mathbf{att}}^{r2a}$ are learnable parameters, and both $\mathbf{enc}^{r2a}$ and $\mathbf{att}^{r2a}$ are shareable across all tasks. We minimize the distance between $\hat{\alpha}^t$ and the α^t obtained from the first multi-task learning module over all source data:

$$\mathcal{L}_{att} = \sum_{(\alpha^t, \hat{\alpha}^t), t \in \{\mathcal{S}_i\}_{i=1}^N} \mathrm{d}(\alpha^t, \hat{\alpha}^t),$$

where $\mathrm{d}(\cdot,\cdot)$ can be any distance metric. In this paper, we consider a soft-margin cosine distance:

$$\mathrm{d}(a, b) \triangleq \max(0, 1 - \cos(a, b) - 0.1),$$

where $\cos(\cdot,\cdot)$ denotes the cosine similarity.

3.4 Pipeline

Training R2A We train the three aforementioned modules jointly using both the source data and the unlabeled target data. The overall objective is given by:

$$\mathcal{L} = \mathcal{L}_{lbl} + \lambda_{att}\mathcal{L}_{att} + \lambda_{lm}\mathcal{L}_{lm} + \lambda_{wd}\mathcal{L}_{wd}. \quad (1)$$

The λs are hyper-parameters that control the importance of each training objective and $\mathcal{L}$s represent the corresponding loss functions.

R2A inference Once the R2A model is trained, we can generate attention for each labeled target example based on its human-annotated rationales.

Training target classifier When testing the performance on the target task, of course, we are neither provided with labels nor rationales. In order to make predictions for the target task, we train a new classifier under the supervision of both the labels and the R2A-generated attention. Specifically, this target classifier shares the same architecture as the source one in the multi-task learning module. We minimize the prediction loss, $\mathcal{L}_{lbl}^{\mathcal{T}}$, on the labeled target data together with the cosine distance, $\mathcal{L}_{att}^{\mathcal{T}}$, between the R2A-generated attention and the attention generated by this target classifier. The joint objective for this target classifier is defined as

$$\mathcal{L} = \mathcal{L}_{lbl}^{\mathcal{T}} + \lambda_{att}^{\mathcal{T}}\mathcal{L}_{att}^{\mathcal{T}}, \quad (2)$$

where $\lambda_{att}^{\mathcal{T}}$ controls the importance of $\mathcal{L}_{att}^{\mathcal{T}}$. For better transfer, we initialize the encoder in the target classifier as $\mathbf{enc}$ from the trained R2A model.

4 Experimental Setup

4.1 Datasets

We evaluate our approach on two transfer settings: transfer among aspects within the same domain and transfer among different domains.

Aspect transfer We first consider the transfer problem between multiple aspects of one domain: beer review. We use a subset of the BeerAdvocate[3] review dataset (McAuley et al., 2012) introduced by Lei et al. (2016). This dataset contains reviews with ratings (in the scale of $[0, 1]$) from three aspects of the beer: look, aroma and palate. We treat

---

[3] `https://www.beeradvocate.com`

Beer Aspects	Source Train	Source Dev	Target Train‡	Target Dev	Target Test
Look	43,351	10,170	200	200	4,014
Aroma	39,825	8,772	200	200	4,212
Palate	30,041	7,152	200	200	3,804

Table 1: Statistics of the beer review dataset. ‡ denotes data with human-annotated rationales.

Hotel Aspects	Target Train‡	Target Dev	Target Test	Target Unlabeled
Location	200	200	1,808	14,472
Cleanliness	200	200	12,684	14,472
Service	200	200	18,762	14,472

Table 2: Statistics of the hotel review dataset. ‡ denotes data with human-annotated rationales.

any two aspects as the source and the other one as the target. We consider a classification setting for each target task. Specifically, reviews with ratings ≤ 0.4 are labeled as negative and those with ≥ 0.6 are labeled as positive. We form our dataset by randomly selecting an equal number of positive and negative examples. Then we randomly select 200 examples and ask human annotators to provide rationales (see Appendix 2 for details). These 200 examples are treated as our labeled training data for the target aspect. Unlabeled target data is not required since both source and target tasks are from the same domain. Table 1 summarizes the statistics of the beer review dataset.

Domain transfer Our second experiment focuses on domain transfer from beer reviews to different aspects of hotel reviews. We use the TripAdvisor[4] hotel review dataset (Wang et al., 2010), with the following three aspects as our transfer target: location, cleanliness, and service. For each aspect, reviews with ratings > 3 are labeled as positive and those with < 3 are labeled as negative. Similarly, we collect human rationales for 200 examples and treat them as our training data (see Appendix 2 for details). Table 2 summarizes the statistics of the hotel review dataset. In this experiment, data from all three aspects of the beer reviews are treated as the source tasks.

4.2 Baselines

We compare our approach (OURS) with four types of baselines:

Basic classifier We train a linear SVM using bag-of-ngrams representation on the labeled target data. We combine uni-gram, bi-grams, and tri-grams as features and use tf-idf weighting.

Rationale augmented classifiers We evaluate two previous methods that incorporate human rationales during training: 1) rationale augmented SVM (RA-SVM) (Zaidan et al., 2007), an SVM-based model that utilizes human rationales to regularize the decision boundary of the classifier; 2) rationale augmented CNN (RA-CNN) (Zhang et al., 2016). RA-CNN first trains a CNN-based sentence classifier to estimate the probability that a given sentence contains rationale words. Then RA-CNN scales the contribution of each sentence to the overall representation in proportion to these estimates. The final prediction is made from this overall representation.

Transfer methods We compare against two variants of our method: 1) TRANS, an attention-based classifier that does not use human rationales from the target task; 2) RA-TRANS, an attention-based classifier that directly uses human rationales to supervise the attention. Specifically, TRANS only optimizes the cross-entropy loss $\mathcal{L}_{lbl}^{\mathcal{T}}$ in the objective (Eq. (2)). For RA-TRANS, the term $\mathcal{L}_{att}^{\mathcal{T}}$ in the objective Eq. (2) is replaced by the cosine distance between human rationales and the attention generated by itself. Note that both models still have the ability to transfer, as their encoders are both initialized from $\mathbf{enc}$, which has been trained on source data and unlabeled target data.

Oracle We also report the performance of an ORACLE which shares the same architecture as ours but is supervised by the oracle attention. The oracle attention is derived from a held-out dataset with large-scale annotations for the target task (see Appendix 3 for details). This helps us analyze the contribution of our R2A approach in isolation of the inherent limitations of the target tasks.

4.3 Implementation details

We use pre-trained fastText embeddings (Bojanowski et al., 2017), a 200-dimension bi-directional LSTM (Hochreiter and Schmidhuber, 1997) for the language encoder, and a 50-dimension bi-directional LSTM for the R2A encoder. Dropout (Srivastava et al., 2014) is applied with drop rate 0.1 on the word embeddings and the last hidden layers of the classifiers. All

---

[4] `https://www.tripadvisor.com`

Source	Target	SVM	RA-SVM‡	RA-CNN‡	TRANS†	RA-TRANS‡†	OURS‡†	ORACLE†
Beer aroma+palate	Beer look	74.41	74.83	74.94	72.75	76.41	**79.53**	80.29
Beer look+palate	Beer aroma	68.57	69.23	67.55	69.92	76.45	**77.94**	78.11
Beer look+aroma	Beer palate	63.88	67.82	65.72	74.66	73.40	**75.24**	75.50

Table 3: Accuracy of transferring between aspects. Models with † use labeled data from source aspects. Models with ‡ use human rationales on the target aspect.

Source	Target	SVM	RA-SVM‡	RA-CNN‡	TRANS†	RA-TRANS‡†	OURS‡†	ORACLE†
Beer look + Beer aroma + Beer palate	Hotel location	78.65	79.09	79.28	80.42	82.10	**84.52**	85.43
	Hotel cleanliness	86.44	86.68	89.01	86.95	87.15	**90.66**	92.09
	Hotel service	85.34	86.61	87.91	87.37	86.40	**89.93**	92.42

Table 4: Accuracy of transferring between domains. Models with † use labeled data from source domains and unlabeled data from the target domain. Models with ‡ use human rationales on the target task.

parameters are optimized using Adam (Kingma and Ba, 2014). We set the initial learning rate to 0.001 and divide it by 10 once the performance on the development set plateaus. For RA-TRANS, OURS and ORACLE, we tuned $\lambda^{\mathcal{T}}_{att}$ from $\{10^2, 10^1, 10^0, 10^{-1}, 10^{-2}\}$. For domain transfer, we set $\lambda_{lm} = 0.1$, $\lambda_{wd} = 0.01$ and $\lambda_{att} = 0.01$. For aspect transfer, we adapt the same hyperparameters, but set $\lambda_{wd} = 0$ as both source tasks and the target task come from the same domain. To encourage the R2A-generated attention to be consistent with the provided rationales in aspect transfer, we augment the overall training objective of R2A in Eq. (3.3) by a consistency regularization, which is computed from the cosine distance between the R2A-generated attention and the provided rationales.

In addition, computing $\mathcal{L}_{lm}$ is both time and memory inefficient because the complexity is linear to the size of the vocabulary, which can be very large. To expedite the training, we adopt a technique proposed by Mikolov et al. (2011), which randomly splits the entire vocabulary into a predefined number of bins and minimizes the loss of the bin prediction instead of the exact token prediction. We set the bin size as 100 for our experiment.

5 Results

Aspect transfer Table 3 summarizes the results of aspect transfer on the beer review dataset. Our model (OURS) obtains substantial gains in accuracy over the baselines across all three target aspects. It closely matches the performance of ORACLE with only 0.40% absolute difference.

Model	Hotel location	Hotel cleanliness	Hotel service
OURS	**84.52**	**90.66**	**89.93**
w/o $\mathcal{L}_{wd}$	82.36	89.79	89.61
w/o $\mathcal{L}_{lm}$	82.47	90.05	89.75

Table 5: Ablation study on domain transfer from beer to hotel.

Specifically, all rationale-augmented methods (RA-SVM, RA-TRANS and OURS) outperform their rationale-free counterparts on average. This confirms the value of human rationales in the low-resource settings. We observe that the transfer baseline that directly uses rationale as augmented supervision (RA-TRANS) underperforms ORACLE by a large margin. This validates our hypothesis that human rationales and attention are different.

Domain transfer Table 4 presents the results of domain transfer using 200 training examples. We use the three aspects of the beer review data together as our source tasks while use the three aspects of hotel review data as the target. Our model (OURS) shows marked performance improvement. The error reduction over the best baseline is 15.08% on average.

We compare the learning curve in Figure 4. We observe that the performance of our model steadily improves as more annotations are provided, and the improvement over other baselines is significant and consistent.

Ablation study Table 5 presents the results of an ablation study of our model in the setting of domain transfer. As this table indicates, both the language modeling objective and the Wasserstein

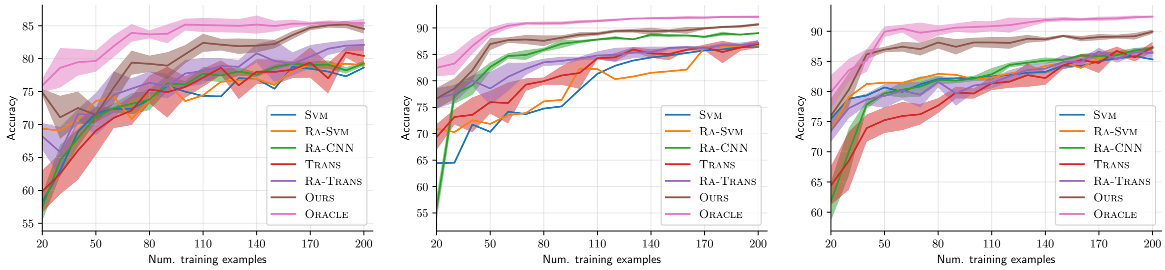

Figure 4: Learning curve of transferring from beer review to three aspects of hotel review: location (left), cleanliness (center) and service (right). For neural methods, we ran five different random seeds and plot their mean with their standard deviation.

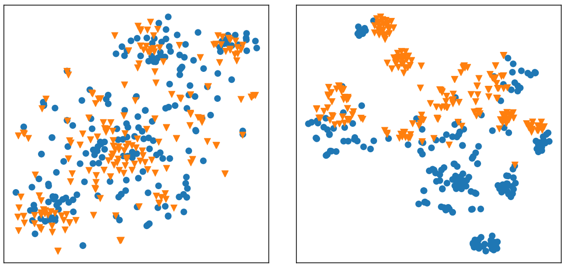

(a) OURS

(b) OURS w/o $\mathcal{L}_{wd}$

Figure 5: t-SNE visualization of the learned hidden representation[5] for beer review (blue circle) and hotel review (orange triangle).

Target	Human rationales	R2A-generated attention
Location	0.5185	**0.2371**
Cleanliness	0.5948	**0.3141**
Service	0.5833	**0.2871**

Table 6: Avg. cosine distance to the oracle attention over the target training set. The R2A is trained on beer reviews with unlabeled hotel reviews.

distance contribute similarly to the task, with the Wasserstein distance having a bigger impact.

Visualization of representation Figure 5 visualizes the hidden representation of 200 beer reviews and 200 hotel reviews using t-SNE (Maaten and Hinton, 2008). We observe that our model successfully aligns the source and the target feature distribution. This indicates the effectiveness of optimizing the Wasserstein distance objective.

Analysis of R2A-generated attention In order to validate that the trained R2A model is able to generate task-specific attention from human rationales, we perform both qualitative and quantitative analysis on the R2A-generated attention in the setting of domain transfer. It is worth pointing out that our R2A model has never seen any labeled hotel reviews during training.

Table 6 presents the average cosine distance between the R2A-generated attention and the oracle attention over the target training set. Compared with human rationales, the R2A-generated attention is much closer to the oracle attention. This explains the large performance boost of our method.

Figure 6 visualize the R2A-generated attention on the same hotel review with human rationales corresponding to three different aspects. We observe that the trained R2A model is able to produce task-specific attention scores corresponding to the provided human rationale. For example, given the rationale sentence "not the cleanest rooms but bed was clean and so was bathroom", R2A recognizes that not every token is equally important, and the attention should focus more on "clean", "cleanest", "rooms" and "bathroom".

Annotating rationales versus annotating more labeled data Providing rationales for the training data roughly doubles the annotation cost (Zaidan et al., 2007). Given the same annotation budget, a natural question is: shall we collect a few labeled examples with rationales or annotate more labeled examples? To answer this question, we vary the number of training examples in the target task. Figure 7 shows the corresponding learning curve of a classifier that is trained without rationales. The reference line represents the accuracy of our approach trained on 200 examples with rationales. We notice that in order to reach the same level of performance, the rationale-free classifier requires 800, 3100, and 1900 labeled examples on the three target tasks respectively.

---

[5] Since the hidden representation is a sequence of variable length, we applied t-SNE on the concatenation of the first and last element: $[h_1^{\mathbf{inv}}; h_L^{\mathbf{inv}}]$.

Task: Hotel location Oracle attention

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! **for location and price , this ca n't be beaten** , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel location R2A-generated attention

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! **for location and price , this ca n't be beaten** , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel cleanliness Oracle attention

you get what you pay for . **not the cleanest rooms but bed was clean and so was bathroom** . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel cleanliness R2A-generated attention

you get what you pay for . **not the cleanest rooms but bed was clean and so was bathroom** . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel service Oracle Attention

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . **service was excellent** , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel service R2A-generated attention

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . **service was excellent** , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Figure 6: Oracle attention versus R2A-generated attention on an example of hotel review for three different tasks. Words are highlighted according to the attention scores. Human rationales are shown in bold with underlines. The oracle attention is derived from large amounts of labeled hotel reviews. The R2A is trained on labeled beer reviews with unlabeled hotel reviews.

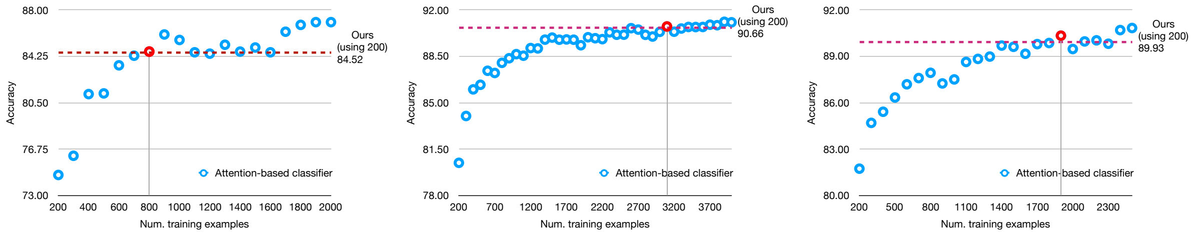

Figure 7: Learning curve of an attention-based classifier on three tasks: hotel location (left), hotel cleanliness (center), hotel service (right). The performance of our approach trained on 200 examples with human rationales is shown as a reference.

6 Conclusion

In this paper, we propose a novel approach that utilizes the connection between human rationales and machine attention to improve the performance of low-resource tasks. Specifically, we learn a transferrable mapping from rationales to high-quality attention on resource-rich tasks. The learned mapping is then used to provide an additional supervision for the target task. Experimental results on both aspect and domain transfer validate that the R2A-generated attention serves as a better form of supervision. Our model produces high-quality attention for low-resource tasks.

Acknowledgments

We thank the MIT NLP group and the reviewers for their helpful discussion and comments. This work is supported by MIT-IBM Watson AI Lab. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1870–1879.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*.

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

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779.

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

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2017. Supervised attention for sequence-to-sequence constituency parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 7–12.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. *arXiv preprint arXiv:1609.04186*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1789–1798.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1020–1025. IEEE.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 783–792. ACm.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using annotator rationales to improve machine learning for text categorization. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 260–267.

Ye Zhang, Iain Marshall, and Byron C Wallace. 2016. Rationale-augmented convolutional neural networks for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 795. NIH Public Access.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association of Computational Linguistics*, 5(1):515–528.

Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. 2016. Bi-transferring deep neural networks for domain adaptation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 322–332.

Semi-Supervised Sequence Modeling with Cross-View Training

Kevin Clark[1] **Minh-Thang Luong**[2] **Christopher D. Manning**[1] **Quoc V. Le**[2]
[1]Computer Science Department, Stanford University [2]Google Brain
`kevclark@cs.stanford.edu, thangluong@google.com, manning@cs.stanford.edu, qvl@google.com`

Abstract

Unsupervised representation learning algorithms such as word2vec and ELMo improve the accuracy of many supervised NLP models, mainly because they can take advantage of large amounts of unlabeled text. However, the supervised models only learn from task-specific labeled data during the main training phase. We therefore propose Cross-View Training (CVT), a semi-supervised learning algorithm that improves the representations of a Bi-LSTM sentence encoder using a mix of labeled and unlabeled data. On labeled examples, standard supervised learning is used. On unlabeled examples, CVT teaches auxiliary prediction modules that see restricted views of the input (e.g., only part of a sentence) to match the predictions of the full model seeing the whole input. Since the auxiliary modules and the full model share intermediate representations, this in turn improves the full model. Moreover, we show that CVT is particularly effective when combined with multi-task learning. We evaluate CVT on five sequence tagging tasks, machine translation, and dependency parsing, achieving state-of-the-art results.[1]

1 Introduction

Deep learning models work best when trained on large amounts of labeled data. However, acquiring labels is costly, motivating the need for effective semi-supervised learning techniques that leverage unlabeled examples. A widely successful semi-supervised learning strategy for neural NLP is pre-training word vectors (Mikolov et al., 2013). More recent work trains a Bi-LSTM sentence encoder to do language modeling and then incorporates its context-sensitive representations into supervised models (Dai and Le, 2015; Peters et al., 2018). Such pre-training methods perform unsupervised representation learning on a large corpus of unlabeled data followed by supervised training.

A key disadvantage of pre-training is that the first representation learning phase does not take advantage of labeled data – the model attempts to learn generally effective representations rather than ones that are targeted towards a particular task. Older semi-supervised learning algorithms like self-training do not suffer from this problem because they continually learn about a task on a mix of labeled and unlabeled data. Self-training has historically been effective for NLP (Yarowsky, 1995; McClosky et al., 2006), but is less commonly used with neural models. This paper presents Cross-View Training (CVT), a new self-training algorithm that works well for neural sequence models.

In self-training, the model learns as normal on labeled examples. On unlabeled examples, the model acts as both a teacher that makes predictions about the examples and a student that is trained on those predictions. Although this process has shown value for some tasks, it is somewhat tautological: the model already produces the predictions it is being trained on. Recent research on computer vision addresses this by adding noise to the student's input, training the model so it is robust to input perturbations (Sajjadi et al., 2016; Wei et al., 2018). However, applying noise is difficult for discrete inputs like text.

As a solution, we take inspiration from multi-view learning (Blum and Mitchell, 1998; Xu et al., 2013) and train the model to produce consistent predictions across different *views* of the input. Instead of only training the full model as a student, CVT adds auxiliary prediction modules – neural networks that transform vector representations into predictions – to the model and also trains them as students. The input to each student prediction module is a subset of the model's intermediate rep-

---

[1] Code will be made available at `https://github.com/tensorflow/models/tree/master/research/cvt_text`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1914–1925
Brussels, Belgium, October 31 - November 4, 2018.

resentations corresponding to a restricted view of the input example. For example, one auxiliary prediction module for sequence tagging is attached to only the "forward" LSTM in the model's first Bi-LSTM layer, so it makes predictions without seeing any tokens to the right of the current one.

CVT works by improving the model's representation learning. The auxiliary prediction modules can learn from the full model's predictions because the full model has a better, unrestricted view of the input. As the auxiliary modules learn to make accurate predictions despite their restricted views of the input, they improve the quality of the representations they are built on top of. This in turn improves the full model, which uses the same shared representations. In short, our method combines the idea of representation learning on unlabeled data with classic self-training.

CVT can be applied to a variety of tasks and neural architectures, but we focus on sequence modeling tasks where the prediction modules are attached to a shared Bi-LSTM encoder. We propose auxiliary prediction modules that work well for sequence taggers, graph-based dependency parsers, and sequence-to-sequence models. We evaluate our approach on English dependency parsing, combinatory categorial grammar supertagging, named entity recognition, part-of-speech tagging, and text chunking, as well as English to Vietnamese machine translation. CVT improves over previously published results on all these tasks. Furthermore, CVT can easily and effectively be combined with multi-task learning: we just add additional prediction modules for the different tasks on top of the shared Bi-LSTM encoder. Training a unified model to jointly perform all of the tasks except machine translation improves results (outperforming a multi-task ELMo model) while decreasing the total training time.

2 Cross-View Training

We first present Cross-View Training and describe how it can be combined effectively with multi-task learning. See Figure 1 for an overview of the training method.

2.1 Method

Let $\mathcal{D}_l = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ represent a labeled dataset and $\mathcal{D}_{ul} = \{x_1, x_2, ..., x_M\}$ represent an unlabeled dataset We use $p_\theta(y|x_i)$ to denote the output distribution over classes produced by the model with parameters θ on input x_i. During CVT, the model alternates learning on a minibatch of labeled examples and learning on a minibatch of unlabeled examples. For labeled examples, CVT uses standard cross-entropy loss:

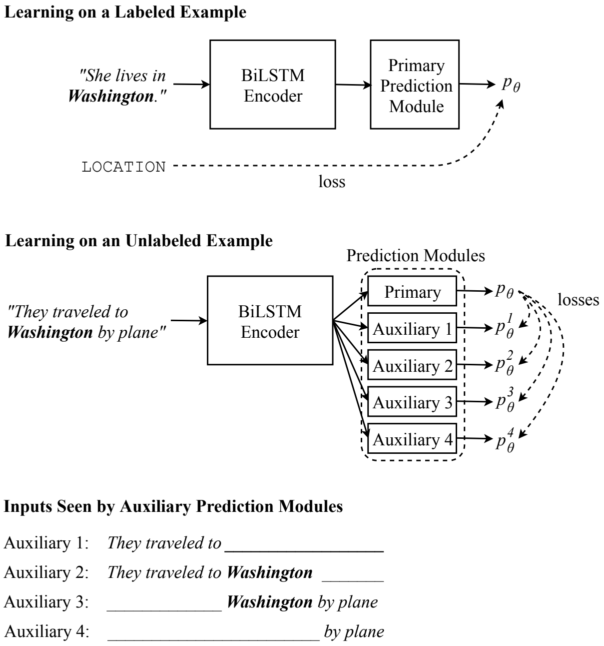

Figure 1: An overview of Cross-View Training. The model is trained with standard supervised learning on labeled examples. On unlabeled examples, auxiliary prediction modules with different views of the input are trained to agree with the primary prediction module. This particular example shows CVT applied to named entity recognition. From the labeled example, the model can learn that "Washington" usually refers to a location. Then, on unlabeled data, auxiliary prediction modules are trained to reach the same prediction without seeing some of the input. In doing so, they improve the contextual representations produced by the model, for example, learning that "traveled to" is usually followed by a location.

$$\mathcal{L}_{\text{sup}}(\theta) = \frac{1}{|\mathcal{D}_l|} \sum_{x_i, y_i \in \mathcal{D}_l} CE(y_i, p_\theta(y|x_i))$$

CVT adds k auxiliary prediction modules to the model, which are used when learning on unlabeled examples. A prediction module is usually a small neural network (e.g., a hidden layer followed by a softmax layer). Each one takes as input an intermediate representation $h^j(x_i)$ produced by the model (e.g., the outputs of one of the LSTMs in a Bi-LSTM model). It outputs a distribution over labels $p_\theta^j(y|x_i)$. Each h^j is chosen such that it only uses a part of the input x_i; the particular choice can depend on the task and model architecture. We propose variants for several tasks in Section 3. The auxiliary prediction modules are only used during training; the test-time prediction come from the primary prediction module that produces p_θ.

On an unlabeled example, the model first produces soft targets $p_\theta(y|x_i)$ by performing inference. CVT trains the auxiliary prediction modules to match the primary prediction module on the unlabeled data by minimizing

$$\mathcal{L}_{\text{CVT}}(\theta) = \frac{1}{|\mathcal{D}_{ul}|} \sum_{x_i \in \mathcal{D}_{ul}} \sum_{j=1}^{k} D(p_\theta(y|x_i), p_\theta^j(y|x_i))$$

where D is a distance function between probability distributions (we use KL divergence). We hold the primary module's prediction $p_\theta(y|x_i)$ fixed during training (i.e., we do not back-propagate through it) so the auxiliary modules learn to imitate the primary one, but not vice versa. CVT works by enhancing the model's representation learning. As the auxiliary modules train, the representations they take as input improve so they are useful for making predictions even when some of the model's inputs are not available. This in turn improves the primary prediction module, which is built on top of the same shared representations.

We combine the supervised and CVT losses into the total loss, $\mathcal{L} = \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{CVT}}$, and minimize it with stochastic gradient descent. In particular, we alternate minimizing $\mathcal{L}_{\text{sup}}$ over a minibatch of labeled examples and minimizing $\mathcal{L}_{\text{CVT}}$ over a minibatch of unlabeled examples.

For most neural networks, adding a few additional prediction modules is computationally cheap compared to the portion of the model building up representations (such as an RNN or CNN). Therefore our method contributes little overhead to training time over other self-training approaches for most tasks. CVT does not change inference time or the number of parameters in the fully-trained model because the auxiliary prediction modules are only used during training.

2.2 Combining CVT with Multi-Task Learning

CVT can easily be combined with multi-task learning by adding additional prediction modules for the other tasks on top of the shared Bi-LSTM encoder. During supervised learning, we randomly select a task and then update $\mathcal{L}_{\text{sup}}$ using a minibatch of labeled data for that task. When learning on the unlabeled data, we optimize $\mathcal{L}_{\text{CVT}}$ jointly across all tasks at once, first running inference with all the primary prediction modules and then learning from the predictions with all the auxiliary prediction modules. As before, the model alternates training on minibatches of labeled and unlabeled examples.

Examples labeled across many tasks are useful for multi-task systems to learn from, but most datasets are only labeled with one task. A benefit of multi-task CVT is that the model creates (artificial) all-tasks-labeled examples from unlabeled data. This significantly improves the model's data efficiency and training time. Since running prediction modules is computationally cheap, computing $\mathcal{L}_{\text{CVT}}$ is not much slower for many tasks than it is for a single one. However, we find the all-tasks-labeled examples substantially speed up model convergence. For example, our model trained on six tasks takes about three times as long to converge as the average model trained on one task, a 2x decrease in total training time.

3 Cross-View Training Models

CVT relies on auxiliary prediction modules that have restricted views of the input. In this section, we describe specific constructions of the auxiliary prediction modules that are effective for sequence tagging, dependency parsing, and sequence-to-sequence learning.

3.1 Bi-LSTM Sentence Encoder

All of our models use a two-layer CNN-BiLSTM (Chiu and Nichols, 2016; Ma and Hovy, 2016) sentence encoder. It takes as input a sequence of words $x_i = [x_i^1, x_i^2, ..., x_i^T]$. First, each word is represented as the sum of an embedding vector and the output of a character-level Convolutional Neural Network, resulting in a sequence of vectors $v = [v^1, v^2, ..., v^T]$. The encoder applies a two-layer bidirectional LSTM (Graves and Schmidhuber, 2005) to these representations. The first layer runs a Long Short-Term Memory unit (Hochreiter and Schmidhuber, 1997) in the forward direction (taking v^t as input at each step t) and the backward direction (taking v^{T-t+1} at each step) to produce vector sequences $[\overrightarrow{h}_1^1, \overrightarrow{h}_1^2, ... \overrightarrow{h}_1^T]$ and $[\overleftarrow{h}_1^1, \overleftarrow{h}_1^2, ... \overleftarrow{h}_1^T]$. The output of the Bi-LSTM is the concatenation of these vectors: $h_1 = [\overrightarrow{h}_1^1 \oplus \overleftarrow{h}_1^1, ..., \overrightarrow{h}_1^T \oplus \overleftarrow{h}_1^T]$. The second Bi-LSTM layer works the same, producing outputs h_2, except it takes h_1 as input instead of v.

3.2 CVT for Sequence Tagging

In sequence tagging, each token x_i^t has a corresponding label y_i^t. The primary prediction module for sequence tagging produces a probability distribution over classes for the t^{th} label using a one-hidden-layer neural network applied to the corresponding encoder outputs:

$$\begin{aligned} p(y^t|x_i) &= \text{NN}(h_1^t \oplus h_2^t) \\ &= \text{softmax}(U \cdot \text{ReLU}(W(h_1^t \oplus h_2^t)) + b) \end{aligned}$$

The auxiliary prediction modules take $\overrightarrow{h}_1(x_i)$ and $\overleftarrow{h}_1(x_i)$, the outputs of the forward and backward LSTMs in the first[2] Bi-LSTM layer, as inputs. We add the following four auxiliary prediction modules to the model (see Figure 2):

$$\begin{aligned} p_\theta^{\text{fwd}}(y^t|x_i) &= \text{NN}^{\text{fwd}}(\overrightarrow{h}_1^t(x_i)) \\ p_\theta^{\text{bwd}}(y^t|x_i) &= \text{NN}^{\text{bwd}}(\overleftarrow{h}_1^t(x_i)) \\ p_\theta^{\text{future}}(y^t|x_i) &= \text{NN}^{\text{future}}(\overrightarrow{h}_1^{t-1}(x_i)) \\ p_\theta^{\text{past}}(y^t|x_i) &= \text{NN}^{\text{past}}(\overleftarrow{h}_1^{t+1}(x_i)) \end{aligned}$$

The "forward" module makes each prediction without seeing the right context of the current token. The "future" module makes each prediction without the right context or the current token itself. Therefore it works like a neural language model that, instead of predicting which token comes next in the sequence, predicts which class of token comes next. The "backward" and "past" modules are analogous.

3.3 CVT for Dependency Parsing

In a dependency parse, words in a sentence are treated as nodes in a graph. Typed directed edges connect the words, forming a tree structure describing the syntactic structure of the sentence. In particular, each word x_i^t in a sentence $x_i = x_i^1, ..., x_i^T$ receives exactly one in-going edge (u, t, r) going from word x_i^u (called the "head") to it (the "dependent") of type r (the "relation"). We use a graph-based dependency parser similar to the one from Dozat and Manning (2017). This treats dependency parsing as a classification task where the goal is to predict which in-going edge $y_i^t = (u, t, r)$ connects to each word x_i^t.

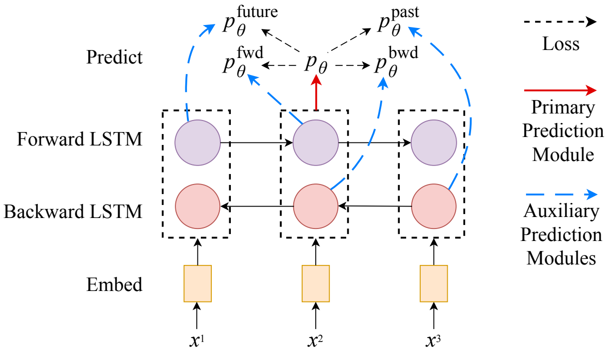

Figure 2: Auxiliary prediction modules for sequence tagging models. Each one sees a restricted view of the input. For example, the "forward" prediction module does not see any context to the right of the current token when predicting that token's label. For simplicity, we only show a one layer Bi-LSTM encoder and only show the model's predictions for a single time step.

First, the representations produced by the encoder for the candidate head and dependent are passed through separate hidden layers. A bilinear classifier applied to these representations produces a score for each candidate edge. Lastly, these scores are passed through a softmax layer to produce probabilities. Mathematically, the probability of an edge is given as:

$$p_\theta((u, t, r)|x_i) \propto e^{s(h_1^u(x_i) \oplus h_2^u(x_i), h_1^t(x_i) \oplus h_2^t(x_i), r)}$$

where s is the scoring function:

$$\begin{aligned} s(z_1, z_2, r) = \; & \text{ReLU}(W_{\text{head}} z_1 + b_{\text{head}})(W_r + W) \\ & \text{ReLU}(W_{\text{dep}} z_2 + b_{\text{dep}}) \end{aligned}$$

The bilinear classifier uses a weight matrix W_r specific to the candidate relation as well as a weight matrix W shared across all relations. Note that unlike in most prior work, our dependency parser only takes words as inputs, not words and part-of-speech tags.

We add four auxiliary prediction modules to our model for cross-view training:

$$\begin{aligned} p_\theta^{\text{fwd-fwd}}((u, t, r)|x_i) &\propto e^{s^{\text{fwd-fwd}}(\overrightarrow{h}_1^u(x_i), \overrightarrow{h}_1^t(x_i), r)} \\ p_\theta^{\text{fwd-bwd}}((u, t, r)|x_i) &\propto e^{s^{\text{fwd-bwd}}(\overrightarrow{h}_1^u(x_i), \overleftarrow{h}_1^t(x_i), r)} \\ p_\theta^{\text{bwd-fwd}}((u, t, r)|x_i) &\propto e^{s^{\text{bwd-fwd}}(\overleftarrow{h}_1^u(x_i), \overrightarrow{h}_1^t(x_i), r)} \\ p_\theta^{\text{bwd-bwd}}((u, t, r)|x_i) &\propto e^{s^{\text{bwd-bwd}}(\overleftarrow{h}_1^u(x_i), \overleftarrow{h}_1^t(x_i), r)} \end{aligned}$$

Each one has some missing context (not seeing either the preceding or following words) for the candidate head and candidate dependent.

---

[2] Modules taking inputs from the second Bi-LSTM layer would not have restricted views because information about the whole sentence gets propagated through the first layer.

3.4 CVT for Sequence-to-Sequence Learning

We use an encoder-decoder sequence-to-sequence model with attention (Sutskever et al., 2014; Bahdanau et al., 2015). Each example consists of an input (source) sequence $x_i = x_i^1, ..., x_i^T$ and output (target) sequence $y_i = y_i^1, ..., y_i^K$. The encoder's representations are passed into an LSTM decoder using a bilinear attention mechanism (Luong et al., 2015). In particular, at each time step t the decoder computes an attention distribution over source sequence hidden states as $\alpha_j \propto e^{h^j W_\alpha \bar{h}^t}$ where $\bar{h}^t$ is the decoder's current hidden state. The source hidden states weighted by the attention distribution form a context vector: $c_t = \sum_j \alpha_j h^j$. Next, the context vector and current hidden state are combined into an attention vector $a_t = \tanh(W_a[c_t, h_t])$. Lastly, a softmax layer predicts the next token in the output sequence: $p(y_i^t | y_i^{<t}, x_i) = \text{softmax}(W_s a_t)$.

We add two auxiliary decoders when applying CVT. The auxiliary decoders share embedding and LSTM parameters with the primary decoder, but have different parameters for the attention mechanisms and softmax layers. For the first one, we restrict its view of the input by applying attention dropout, randomly zeroing out a fraction of its attention weights. The second one is trained to predict the next word in the target sequence rather than the current one: $p_\theta^{\text{future}}(y_i^t | y_i^{<t}, x_i) = \text{softmax}(W_s^{\text{future}} a_{t-1}^{\text{future}})$. Since there is no target sequence for unlabeled examples, we cannot apply teacher forcing to get an output distribution over the vocabulary from the primary decoder at each time step. Instead, we produce hard targets for the auxiliary modules by running the primary decoder with beam search on the input sequence. This idea has previously been applied to sequence-level knowledge distillation by Kim and Rush (2016).

4 Experiments

We compare Cross-View Training against several strong baselines on seven tasks:

Combinatory Categorial Grammar (CCG) Supertagging: We use data from CCGBank (Hockenmaier and Steedman, 2007).

Text Chunking: We use the CoNLL-2000 data (Tjong Kim Sang and Buchholz, 2000).

Named Entity Recognition (NER): We use the CoNLL-2003 data (Tjong Kim Sang and De Meulder, 2003).

Fine-Grained NER (FGN): We use the OntoNotes (Hovy et al., 2006) dataset.

Part-of-Speech (POS) Tagging: We use the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993).

Dependency Parsing: We use the Penn Treebank converted to Stanford Dependencies version 3.3.0.

Machine Translation: We use the English-Vietnamese translation dataset from IWSLT 2015 (Cettolo et al., 2015). We report (tokenized) BLEU scores on the tst2013 test set.

We use the 1 Billion Word Language Model Benchmark (Chelba et al., 2014) as a pool of unlabeled sentences for semi-supervised learning.

4.1 Model Details and Baselines

We apply dropout during training, but not when running the primary prediction module to produce soft targets on unlabeled examples. In addition to the auxiliary prediction modules listed in Section 3, we find it slightly improves results to add another one that sees the whole input rather than a subset (but unlike the primary prediction module, does have dropout applied to its representations). Unless indicated otherwise, our models have LSTMs with 1024-sized hidden states and 512-sized projection layers. See the supplementary material for full training details and hyperparameters. We compare CVT with the following other semi-supervised learning algorithms:

Word Dropout. In this method, we only train the primary prediction module. When acting as a teacher it is run as normal, but when acting as a student, we randomly replace some of the input words with a `REMOVED` token. This is similar to CVT in that it exposes the model to a restricted view of the input. However, it is less data efficient. By carefully designing the auxiliary prediction modules, it is possible to train the auxiliary prediction modules to match the primary one across many different views of the input a once, rather than just one view at a time.

Virtual Adversarial Training (VAT). VAT (Miyato et al., 2016) works like word dropout, but adds noise to the word embeddings of the student instead of dropping out words. Notably, the noise is chosen adversarially so it most changes the model's prediction. This method was applied successfully to semi-supervised text classification

Method	CCG	Chunk	NER	FGN	POS	Dep. Parse		Translate
	Acc.	F1	F1	F1	Acc.	UAS	LAS	BLEU
Shortcut LSTM (Wu et al., 2017)	95.1				97.53			
ID-CNN-CRF (Strubell et al., 2017)			90.7	86.8				
JMT† (Hashimoto et al., 2017)		95.8			97.55	94.7	92.9	
TagLM* (Peters et al., 2017)		96.4	91.9					
ELMo* (Peters et al., 2018)			92.2					
Biaffine (Dozat and Manning, 2017)						95.7	94.1	
Stack Pointer (Ma et al., 2018)						95.9	94.2	
Stanford (Luong and Manning, 2015)								23.3
Google (Luong et al., 2017)								26.1
Supervised	94.9	95.1	91.2	87.5	97.60	95.1	93.3	28.9
Virtual Adversarial Training*	95.1	95.1	91.8	87.9	97.64	95.4	93.7	–
Word Dropout*	95.2	95.8	92.1	88.1	97.66	95.6	93.8	29.3
ELMo (our implementation)*	95.8	96.5	92.2	88.5	97.72	96.2	94.4	29.3
ELMo + Multi-task*†	95.9	96.8	92.3	88.4	**97.79**	96.4	94.8	–
CVT*	95.7	96.6	92.3	88.7	97.70	95.9	94.1	**29.6**
CVT + Multi-task*†	96.0	96.9	92.4	88.4	97.76	96.4	94.8	–
CVT + Multi-task + Large*†	**96.1**	**97.0**	**92.6**	**88.8**	97.74	**96.6**	**95.0**	–

Table 1: Results on the test sets. We report the mean score over 5 runs. Standard deviations in score are around 0.1 for NER, FGN, and translation, 0.02 for POS, and 0.05 for the other tasks. See the supplementary materials for results with them included. The +Large model has four times as many hidden units as the others, making it similar in size to the models when ELMo is included. * denotes semi-supervised and † denotes multi-task.

by Miyato et al. (2017).

ELMo. ELMo incorporates the representations from a large separately-trained language model into a task-specific model. Our implementaiton follows Peters et al. (2018). When combining ELMo with multi-task learning, we allow each task to learn its own weights for the ELMo embeddings going into each prediction module. We found applying dropout to the ELMo embeddings was crucial for achieving good performance.

4.2 Results

Results are shown in Table 1. CVT on its own outperforms or is comparable to the best previously published results on all tasks. Figure 3 shows an example win for CVT over supervised learning.

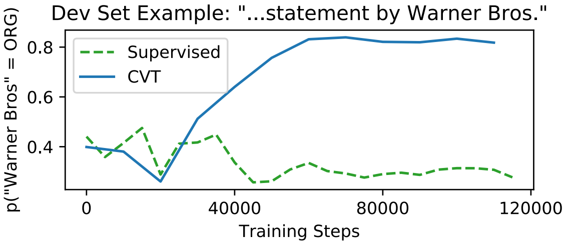

Figure 3: An NER example that CVT classifies correctly but supervised learning does not. "Warner" only occurs as a last name in the train set, so the supervised model classifies "Warner Bros" as a person. The CVT model also mistakenly classifies "Warner Bros" as a person to start with, but as it sees more of the unlabeled data (in which "Warner" occurs thousands of times) it learns that "Warner Bros" is an organization.

Of the prior results listed in Table 1, only TagLM and ELMo are semi-supervised. These methods first train an enormous language model on unlabeled data and incorporate the representations produced by the language model into a supervised classifier. Our base models use 1024 hidden units in their LSTMs (compared to 4096 in ELMo), require fewer training steps (around one pass over the billion-word benchmark rather than many passes), and do not require a pipelined training procedure. Therefore, although they perform on par with ELMo, they are faster and simpler to train. Increasing the size of our CVT+Multi-task model so it has 4096 units in its LSTMs like ELMo improves results further so they are significantly better than the ELMo+Multi-task ones. We suspect there could be further gains from combining our method with language model pre-training, which we leave for future work.

CVT + Multi-Task. We train a single shared-encoder CVT model to perform all of the tasks except machine translation (as it is quite different and requires more training time than the other ones). Multi-task learning improves results on all of the tasks except fine-grained NER, sometimes by large margins. Prior work on many-task NLP such as Hashimoto et al. (2017) uses complicated architectures and training algorithms. Our result shows that simple parameter sharing can be enough for effective many-task learning when the model is big and trained on a large amount of data.

Interestingly, multi-task learning works better in conjunction with CVT than with ELMo. We hypothesize that the ELMo models quickly fit to the data primarily using the ELMo vectors, which perhaps hinders the model from learning effective representations that transfer across tasks. We also believe CVT alleviates the danger of the model "forgetting" one task while training on the other ones, a well-known problem in many-task learning (Kirkpatrick et al., 2017). During multi-task CVT, the model makes predictions about unlabeled examples across all tasks, creating (artificial) all-tasks-labeled examples, so the model does not only see one task at a time. In fact, multi-task learning plus self training is similar to the Learning without Forgetting algorithm (Li and Hoiem, 2016), which trains the model to keep its predictions on an old task unchanged when learning a new task. To test the value of all-tasks-labeled examples, we trained a multi-task CVT model that only computes $\mathcal{L}_{\text{CVT}}$ on one task at a time (chosen randomly for each unlabeled minibatch) instead of for all tasks in parallel. The one-at-a-time model performs substantially worse (see Table 2).

Model	CCG	Chnk	NER	FGN	POS	Dep.
CVT-MT	95.7	97.4	96.0	86.7	97.74	94.4
w/out parallel	95.4	97.1	95.6	86.3	97.71	94.1

Table 2: Dev set performance of multi-task CVT with and without producing all-tasks-labeled examples.

Model Generalization. In order to evaluate how our models generalize to the dev set from the train set, we plot the dev vs. train accuracy for our different methods as they learn (see Figure 4). Both CVT and multi-task learning improve model generalization: for the same train accuracy, the models get better dev accuracy than purely supervised learning. Interestingly, CVT continues to improve in dev set accuracy while close to 100% train accuracy for CCG, Chunking, and NER, perhaps because the model is still learning from unlabeled data even when it has completely fit to the train set. We also show results for a smaller multi-task + CVT model. Although it generalizes at least as well as the larger one, it halts making progress on the train set earlier. This suggests it is important to use sufficiently large neural networks for multi-task learning: otherwise the model does not have the capacity to fit to all the training data.

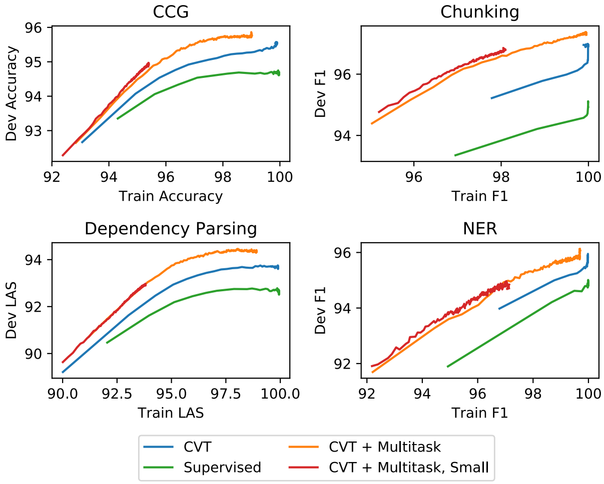

Figure 4: Dev set vs. Train set accuracy for various methods. The "small" model has 1/4 the LSTM hidden state size of the other ones (256 instead of 1024).

Auxiliary Prediction Module Ablation. We briefly explore which auxiliary prediction modules are more important for the sequence tagging tasks in Table 3. We find that both kinds of auxiliary prediction modules improve performance, but that the future and past modules improve results more than the forward and backward ones, perhaps because they see a more restricted and challenging view of the input.

Model	CCG	Chnk	NER	FGN	POS
Supervised	94.8	95.5	95.0	86.0	97.59
CVT	95.6	97.0	95.9	87.3	97.66
no fwd/bwd	–0.1	–0.2	–0.2	–0.1	–0.01
no future/past	–0.3	–0.4	–0.4	–0.3	–0.04

Table 3: Ablation study on auxiliary prediction modules for sequence tagging.

Training Models on Small Datasets. We explore how CVT scales with dataset size by varying the amount of training data the model has ac-

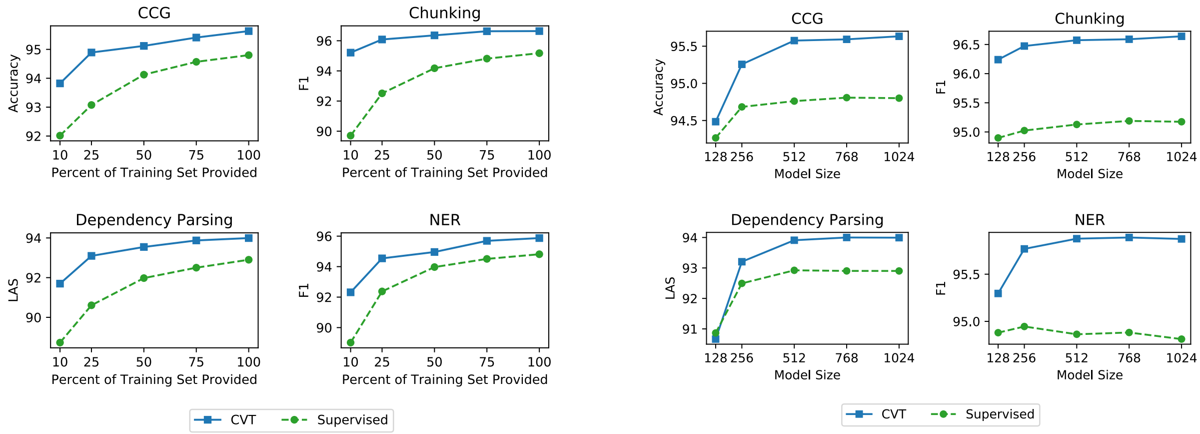

Figure 5: Left: Dev set performance vs. percent of the training set provided to the model. Right: Dev set performance vs. model size. The x axis shows the number of hidden units in the LSTM layers; the projection layers and other hidden layers in the network are half that size. Points correspond to the mean of three runs.

cess to. Unsurprisingly, the improvement of CVT over purely supervised learning grows larger as the amount of labeled data decreases (see Figure 5, left). Using only 25% of the labeled data, our approach already performs as well or better than a fully supervised model using 100% of the training data, demonstrating that CVT is particularly useful on small datasets.

Training Larger Models. Most sequence taggers and dependency parsers in prior work use small LSTMs (hidden state sizes of around 300) because larger models yield little to no gains in performance (Reimers and Gurevych, 2017). We found our own supervised approaches also do not benefit greatly from increasing the model size. In contrast, when using CVT accuracy scales better with model size (see Figure 5, right). This finding suggests the appropriate semi-supervised learning methods may enable the development of larger, more sophisticated models for NLP tasks with limited amounts of labeled data.

Generalizable Representations. Lastly, we explore training the CVT+multi-task model on five tasks, freezing the encoder, and then only training a prediction module on the sixth task. This tests whether the encoder's representations generalize to a new task not seen during its training. Only training the prediction module is very fast because (1) the encoder (which is by far the slowest part of the model) has to be run over each example only once and (2) we do not back-propagate into the encoder. Results are shown in Table 4.

Model	CCG	Chnk	NER	FGN	POS	Dep.
Supervised	94.8	95.6	95.0	86.0	97.59	92.9
CVT-MT frozen	95.1	96.6	94.6	83.2	97.66	92.5
ELMo frozen	94.3	92.2	91.3	80.6	97.50	89.4

Table 4: Comparison of single-task models on the dev sets. "CVT-MT frozen" means we pretrain a CVT + multi-task model on five tasks, and then train only the prediction module for the sixth. "ELMo frozen" means we train prediction modules (but no LSTMs) on top of ELMo embeddings.

Training only a prediction module on top of multi-task representations works remarkably well, outperforming ELMo embeddings and sometimes even a vanilla supervised model, showing the multi-task model is building up effective representations for language. In particular, the representations could be used like skip-thought vectors (Kiros et al., 2015) to quickly train models on new tasks without slow representation learning.

5 Related Work

Unsupervised Representation Learning. Early approaches to deep semi-supervised learning pretrain neural models on unlabeled data, which has been successful for applications in computer vision (Jarrett et al., 2009; LeCun et al., 2010) and NLP. Particularly noteworthy for NLP are algorithms for learning effective word embeddings (Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014) and language model pretraining (Dai and Le, 2015; Ramachandran et al., 2017; Peters et al., 2018; Howard and Ruder, 2018; Radford et al., 2018). Pre-training on other tasks such as machine translation has also been studied (McCann et al., 2017). Other approaches train "thought vectors" representing sentences through unsupervised (Kiros et al., 2015; Hill et al., 2016) or supervised (Conneau et al., 2017) learning.

Self-Training. One of the earliest approaches to semi-supervised learning is self-training (Scudder, 1965), which has been successfully applied to NLP tasks such as word-sense disambiguation (Yarowsky, 1995) and parsing (McClosky et al., 2006). In each round of training, the classifier, acting as a "teacher," labels some of the unlabeled data and adds it to the training set. Then, acting as a "student," it is retrained on the new training set. Many recent approaches (including the consistentency regularization methods discussed below and our own method) train the student with soft targets from the teacher's output distribution rather than a hard label, making the procedure more akin to knowledge distillation (Hinton et al., 2015). It is also possible to use multiple models or prediction modules for the teacher, such as in tri-training (Zhou and Li, 2005; Ruder and Plank, 2018).

Consistency Regularization. Recent works add noise (e.g., drawn from a Gaussian distribution) or apply stochastic transformations (e.g., horizontally flipping an image) to the student's inputs. This trains the model to give consistent predictions to nearby data points, encouraging distributional smoothness in the model. Consistency regularization has been very successful for computer vision applications (Bachman et al., 2014; Laine and Aila, 2017; Tarvainen and Valpola, 2017). However, stochastic input alterations are more difficult to apply to discrete data like text, making consistency regularization less used for natural language processing. One solution is to add noise to the model's word embeddings (Miyato et al., 2017); we compare against this approach in our experiments. CVT is easily applicable to text because it does not require changing the student's inputs.

Multi-View Learning. Multi-view learning on data where features can be separated into distinct subsets has been well studied (Xu et al., 2013). Particularly relevant are co-training (Blum and Mitchell, 1998) and co-regularization (Sindhwani and Belkin, 2005), which trains two models with disjoint views of the input. On unlabeled data, each one acts as a "teacher" for the other model. In contrast to these methods, our approach trains a single unified model where auxiliary prediction modules see different, but not necessarily independent views of the input.

Self Supervision. Self-supervised learning methods train auxiliary prediction modules on tasks where performance can be measured without human-provided labels. Recent work has jointly trained image classifiers with tasks like relative position and colorization (Doersch and Zisserman, 2017), sequence taggers with language modeling (Rei, 2017), and reinforcement learning agents with predicting changes in the environment (Jaderberg et al., 2017). Unlike these approaches, our auxiliary losses are based on self-labeling, not labels deterministically constructed from the input.

Multi-Task Learning. There has been extensive prior work on multi-task learning (Caruana, 1997; Ruder, 2017). For NLP, most work has focused on a small number of closely related tasks (Luong et al., 2016; Zhang and Weiss, 2016; Søgaard and Goldberg, 2016; Peng et al., 2017). Many-task systems are less commonly developed. Collobert and Weston (2008) propose a many-task system sharing word embeddings between the tasks, Hashimoto et al. (2017) train a many-task model where the tasks are arranged hierarchically according to their linguistic level, and Subramanian et al. (2018) train a shared-encoder many-task model for the purpose of learning better sentence representations for use in downstream tasks, not for improving results on the original tasks.

6 Conclusion

We propose Cross-View Training, a new method for semi-supervised learning. Our approach allows models to effectively leverage their own predictions on unlabeled data, training them to produce effective representations that yield accurate predictions even when some of the input is not available. We achieve excellent results across seven NLP tasks, especially when CVT is combined with multi-task learning.

Acknowledgements

We thank Abi See, Christopher Clark, He He, Peng Qi, Reid Pryzant, Yuaho Zhang, and the anonymous reviewers for their thoughtful comments and suggestions. We thank Takeru Miyato for help with his virtual adversarial training code and Emma Strubell for answering our questions about OntoNotes NER. Kevin is supported by a Google PhD Fellowship.

References

Philip Bachman, Ouais Alsharif, and Doina Precup. 2014. Learning with pseudo-ensembles. In *NIPS*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*. ACM.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The IWSLT 2015 evaluation campaign. In *International Workshop on Spoken Language Translation*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

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

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*.

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

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Carl Doersch and Andrew Zisserman. 2017. Multi-task self-supervised visual learning. *arXiv preprint arXiv:1708.07860*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *HLT-NAACL*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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

Julia Hockenmaier and Mark Steedman. 2007. CCG-bank: a corpus of CCG derivations and dependency structures extracted from the Penn treebank. *Computational Linguistics*, 33(3):355–396.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *HLT-NAACL*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. 2017. Reinforcement learning with unsupervised auxiliary tasks. In *ICLR*.

Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. 2009. What is the best multi-stage architecture for object recognition? In *IEEE Conference on Computer Vision*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114 13:3521–3526.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *ICLR*.

Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. 2010. Convolutional networks and applications in vision. In *ISCAS*. IEEE.

Zhizhong Li and Derek Hoiem. 2016. Learning without forgetting. In *ECCV*.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *IWSLT*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNN-CRF. In *ACL*.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *ACL*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The Penn treebank. *Computational linguistics*, 19(2):313–330.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *ACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *ICLR*.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing with virtual adversarial training. In *ICLR*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *https://blog.openai.com/language-unsupervised*.

Prajit Ramachandran, Peter J Liu, and Quoc V Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *EMNLP*.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *ACL*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *EMNLP*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *ACL*.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *NIPS*.

H Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.

Vikas Sindhwani and Mikhail Belkin. 2005. A co-regularization approach to semi-supervised learning with multiple views. In *ICML Workshop on Learning with Multiple Views*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate sequence labeling with iterated dilated convolutions. In *EMNLP*.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *ICLR*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Antti Tarvainen and Harri Valpola. 2017. Weight-averaged consistency targets improve semi-supervised deep learning results. In *Workshop on Learning with Limited Labeled Data, NIPS*.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *CoNLL*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *HLT-NAACL*.

Xiang Wei, Zixia Liu, Liqiang Wang, and Boqing Gong. 2018. Improving the improved training of Wasserstein GANs. In *ICLR*.

Huijia Wu, Jiajun Zhang, and Chengqing Zong. 2017. Shortcut sequence tagging. *arXiv preprint arXiv:1701.00576*.

Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *ACL*.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*.

A Probabilistic Annotation Model for Crowdsourcing Coreference

Silviu Paun[1], Jon Chamberlain[2], Udo Kruschwitz[2], Juntao Yu[1] and Massimo Poesio[1]
[1]School of Electronic Engineering and Computer Science, Queen Mary University of London
[2]School of Computer Science and Electronic Engineering, University of Essex

Abstract

The availability of large scale annotated corpora for coreference is essential to the development of the field. However, creating resources at the required scale via expert annotation would be too expensive. Crowdsourcing has been proposed as an alternative; but this approach has not been widely used for coreference. This paper addresses one crucial hurdle on the way to make this possible, by introducing a new model of annotation for aggregating crowdsourced anaphoric annotations. The model is evaluated along three dimensions: the accuracy of the inferred mention pairs, the quality of the post-hoc constructed silver chains, and the viability of using the silver chains as an alternative to the expert-annotated chains in training a state of the art coreference system. The results suggest that our model can extract from crowdsourced annotations coreference chains of comparable quality to those obtained with expert annotation.

1 Introduction

The task of identifying and resolving anaphoric reference to discourse entities, known in NLP as coreference resolution, has long been considered a core aspect of language interpretation (Poesio et al., 2016b), also because of its role in applications such as summarization (Baldwin and Morton, 1998; Steinberger et al., 2007), information extraction (Humphreys et al.) or question answering (Morton, 1999; Zheng, 2002).

In the 1990s the field made a paradigmatic turn towards corpus based approaches initiated by campaigns such as MUC (Grishman and Sundheim, 1995; Chinchor, 1998) and since then we have seen the development of a range of data-driven approaches, spurred by the development of ever larger and richer datasets. Nowadays, a variety of datasets exist for several languages (Poesio et al., 2016a). These include medium-scale multilingual datasets such as ONTONOTES (Pradhan et al., 2007; Weischedel et al., 2011), which led to the most recent evaluation campaigns, in particular CONLL 2012 (Pradhan et al., 2012), and are used in most current research (Björkelund and Kuhn, 2014; Martschat and Strube, 2015; Clark and Manning, 2016; Lee et al., 2017). However, there are still many languages and domains for which no such resources are available, and even for English much larger corpora than ONTONOTES will eventually be required.

However, annotating data on the scale required to train state of the art systems using traditional expert annotation would be unaffordable. One alternative is to employ **crowdsourcing**, either via platforms like Amazon Mechanical Turk and Crowdflower, or using **Games-With-A-Purpose** (Poesio et al., 2017). Studies such as (Snow et al., 2008; Raykar et al., 2010) have shown that when a sufficiently large number of workers is employed, expert-level quality can be achieved, at a fraction of the cost required to create such resources using traditional methods. The one effort to create a large-scale coreference corpus entirely through crowdsourcing, the *Phrase Detectives* project (Poesio et al., 2013; Chamberlain et al., 2016; Chamberlain, 2016), employs the *Phrase Detectives* game with a purpose. The *Phrase Detectives* corpus consists of 843 documents for a total of 1.2 million tokens and 392,741 markables; at present, 563 documents for a total of 360,000 tokens have been annotated.[1] A second coreference corpus created using crowdsourcing (in the context of a trivia game) also exists, the

---

[1] Note that although the *Phrase Detectives* corpus is slightly smaller in terms of tokens than the currently largest coreference corpus for English, the CONLL 2012 dataset (Pradhan et al., 2012), it has about twice the number of markables, 390,000 vs. 190,000.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1926–1937
Brussels, Belgium, October 31 - November 4, 2018.

Quiz Bowl dataset (Guha et al., 2015).[2]

However, such existing corpora are not widely used yet. One of the reasons for this is the lack of suitable **aggregation methods** for anaphora. Crowdsourced annotations require aggregation methods to select among the different interpretations produced by the crowd. Standard practice for crowdsourced data analysis has seen a shift in recent years from simple majority vote to much more effective aggregation methods (Smyth et al., 1994; Quoc Viet Hung et al., 2013; Sheshadri and Lease, 2013; Carpenter, 2008; Hovy et al., 2013; Passonneau and Carpenter, 2014). Probabilistic models of annotation, in particular, make it possible to characterize the accuracy of the annotators and correct for their bias (Dawid and Skene, 1979; Passonneau and Carpenter, 2014), to account for item-level effects (e.g.: difficulty) (Whitehill et al., 2009), and to employ different pooling strategies (Carpenter, 2008). However, existing models of annotation cannot be used for anaphora. Such methods assume that coders choose between a fixed set of general labels, the same labels across all annotated items. In anaphoric annotation, by contrast, coders relate markables to coreference chains which depend on the markables that are annotated in that given document (Passonneau, 2004; Artstein and Poesio, 2008)

Contributions In this paper we propose a mention pair-based approach to aggregating crowdsourced anaphoric annotations. Concretely, we introduce a new model of annotation capable of inferring the most likely mention pairs from crowd-annotated anaphoric relations. We then use these pairs to build the most likely coreference chains. This approach to building chains is evaluated on both crowdsourced and synthetic (via simulation) coreference datasets. The evaluations include assessing the accuracy of the inferred mention pairs; the quality of the chains; and the viability of using these chains derived from mention pairs as an alternative to gold chains when training a state of the art coreference system. We conclude by also demonstrating the quality of the proposed model in a standard annotation task. The implementation is available as supplementary material.

2 A Mention-Pair Model of Annotation

Traditional models of annotation (Dawid and Skene, 1979; Smyth et al., 1994; Raykar et al., 2010; Hovy et al., 2013) are specified assuming the annotations are chosen among a general set of classes that is consistent across the annotated items. This is the case in a type of annotation closely related to anaphoric annotation, **information status** annotation (Nissim et al., 2004; Riester et al., 2010). In this type of annotation, an annotator marks a mention as either **discourse old** (DO) – referring to an existing entity (coreference chain) – or as **discourse-new** (DN) – introducing a new coreference chain, but without specifying *which* coreference chain the mention belongs to, if any. We will refer below to categories such as DN and DO as **(general) classes**.

Traditional models of annotation can model this type of annotation, but not the task of anaphoric annotation proper. In standard annotation schemes for anaphora/coreference (Poesio et al., 2016a) the annotator may mark a mention as referring to a discourse new entity as above; but in case the mention is identified as discourse-old, this entity, or **coreference chain**–the set of coreferring mentions–is also specified. The available coreference chains differ from document to document.

Our proposal for a probabilistic model of this type of annotation is based on one of the most widely used models of coreference *resolution*: the **mention pair** model. In the mention pair model, the task of linking the mention to a coreference chain/entity is split in two parts: classifying mention pairs as coreferring or not, and subsequent clustering (Soon et al., 2001; Hoste, 2016). The model we propose addresses the first part.

More formally, the crowdsourced data to be modeled consists of I mentions (indexed by i) annotated by a total of J coders (indexed by j). Each mention i has N_i annotations (indexed by n), for a total of M_i distinct **labels** (indexed by m). Each label m of mention i belongs to a **class** $z_{i,m}$. The label of a mention could be the ID of the antecedent, in case that mention is annotated as belonging to the discourse old (general) class; or could be discourse new or another general class (e.g.: property, non referring). In these latter cases, the labels coincide with the classes they

---

[2] Another corpus creation project using crowdsourcing (and also games) for anaphoric annotation is the *Groningen Meaning Bank* (Bos et al., 2017). However, in the GMB crowdsourcing is not used to generate interpretations: players correct automatically annotated interpretations rather than providing the annotations themselves. Another crucial difference is that interpretations are not aggregated in the sense discussed below; rather, an expert adjudicates between the interpretations produced by players.

belong to.

An important difficulty we had to address is label sparsity. The solution we propose is to transform the mention-level annotations into a series of binary decisions with respect to each candidate label. In the extended literature this is often referred to as the binary relevance method (Tsoumakas and Katakis, 2007; Madjarov et al., 2012). We then model these (label-level) decisions as the result of the sensitivity (the true positive rate) and specificity (the true negative rate) of the annotators which we assume are class dependent. This latter assumption allows inferring different levels of annotator ability for each class (e.g.: capturing that DO labels are generally harder compared to DN).

The graphical model of our **Mention Pair Annotations** model (MPA) is presented in Figure 1, while the generative process is given below:

- For every class $h \in \{1, 2, ..., K\}$:
  - Draw class specific true label likelihood $\pi_h \sim \mathsf{Beta}(a, b)$
- For every annotator $j \in \{1, 2, ..., J\}$:
  - For every class $h \in \{1, 2, ..., K\}$:
    - Draw sensitivity $\alpha_{j,h} \sim \mathsf{Beta}(d, e)$
    - Draw specificity $\beta_{j,h} \sim \mathsf{Beta}(t, u)$
- For every mention $i \in \{1, 2, ..., I\}$:
  - For every candidate label $m \in \{1, 2, ..., M_i\}$:
    - Draw true label indicator $c_{i,m} \sim \mathsf{Bern}(\pi_{z_{i,m}})$
    - For every position $n \in \{1, 2, ..., N_i\}$:
      - If $c_{i,m} = 1$ then draw decision $y_{i,m,n} \sim \mathsf{Bern}(\alpha_{jj[i,m,n],z_{i,m}})$[3]
      - Otherwise, draw decision $y_{i,m,n} \sim \mathsf{Bern}(1 - \beta_{jj[i,m,n],z_{i,m}})$

The model addresses the first part of the mention pair framework: the posterior of the true label indicators is used to link each mention with the most likely label, obtaining the mention pairs. The coreference chains are then built by following the link structure from the inferred pairs.

Note that for a traditional annotation task with no distinction between generic classes and specific labels the MPA model is equivalent to training K binary Bayesian versions of the Dawid and Skene (1979) model (one for each general class) on data processed using the binary relevance method. Note also that whereas traditional models of annotation assume one true class per annotated item, an implicit benefit of our approach is allowing for potentially multiple true classes, which can be useful to detect ambiguity (Poesio and Artstein, 2005), but we don't exploit that in this work.

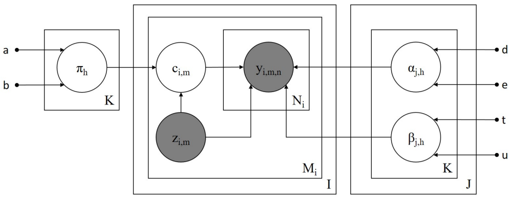

Figure 1: Plate diagram for MPA

2.1 Parameter Estimation

We infer the parameters of the proposed model using Variational Inference (VI). Unlike Markov Chain Monte Carlo (MCMC) approaches (e.g.: Gibbs Sampling, Hamiltonian Monte Carlo), VI is deterministic, fast, and benefits from a clear convergence criterion (Blei et al., 2017).

Specifically we approximate the intractable posterior $p(\theta|D)$ with a variational distribution $q(\theta)$ such that the Kullback-Leibler (KL) divergence between the two distributions is minimized. It can be shown this minimization is equivalent to maximizing the evidence lower bound (ELBO) below:

$$\begin{aligned}\mathcal{L} &= E_q[\log p(\pi, \alpha, \beta, c, y|a, b, d, e, t, u, z)] \\ &- E_q[\log q(\pi, \alpha, \beta, c|\lambda, \eta, \gamma, \mu, \theta, \epsilon, \phi, \zeta)]\end{aligned} \quad (1)$$

We need a variational distribution q that is tractable under expectations. Following common practice (Blei et al., 2003; Hoffman et al., 2013; Blei et al., 2017), we choose q to be in the mean field variational family where each hidden variable is independent and governed by its own parameter. Elegant solutions have been derived for models whose complete conditionals are in the exponential family (Blei and Jordan, 2006; Hoffman et al., 2013). Concretely, we used the fact that the natural parameters of the variational distributions are equal to the expected value of the natural parameters of the corresponding complete conditionals.

The derivations are standard in the VI literature (see, for example, Hoffman et al., 2013). (To save space, we only provide here the update formulas of

---

[3] Notation: jj[i,m,n] returns the index of the annotator who made the n-th decision on the m-th label of mention i.

the variational parameters; supplementary details are in the Appendix.)

Equations (2) and (3) give the variational update formulas for the class-level true label likelihood. We have $q(\pi_h|\lambda_h, \eta_h) = \mathsf{Beta}(\lambda_h, \eta_h)$, where:

$$\lambda_h = a + \sum_{i,m}^{I,M_i} I(z_{i,m} = h) E_q[I(c_{i,m} = 1)] \quad (2)$$

$$\eta_h = b + \sum_{i,m}^{I,M_i} I(z_{i,m} = h) E_q[I(c_{i,m} = 0)] \quad (3)$$

In Equation (4) and (5) we list the variational update formulas for the class-level annotator sensitivity. We have $q(\alpha_{j,h}|\gamma_{j,h}, \mu_{j,h}) = \mathsf{Beta}(\gamma_{j,h}, \mu_{j,h})$, where:

$$\begin{aligned} \gamma_{j,h} = d + \sum_{i,m,n}^{I,M_i,N_i} & I(jj[i,m,n] = j) \\ & I(z_{i,m} = h) I(y_{i,m,n} = 1) E_q[I(c_{i,m} = 1)] \end{aligned} \quad (4)$$

$$\begin{aligned} \mu_{j,h} = e + \sum_{i,m,n}^{I,M_i,N_i} & I(jj[i,m,n] = j) \\ & I(z_{i,m} = h) I(y_{i,m,n} = 0) E_q[I(c_{i,m} = 1)] \end{aligned} \quad (5)$$

In Equations (6) and (7) we list the variational update formulas for the class-level annotator specificity. We have $q(\beta_{j,h}|\theta_{j,h}, \epsilon_{j,h}) = \mathsf{Beta}(\theta_{j,h}, \epsilon_{j,h})$, where:

$$\begin{aligned} \theta_{j,h} = t + \sum_{i,m,n}^{I,M_i,N_i} & I(jj[i,m,n] = j) \\ & I(z_{i,m} = h) I(y_{i,m,n} = 0) E_q[I(c_{i,m} = 0)] \end{aligned} \quad (6)$$

$$\begin{aligned} \epsilon_{j,h} = u + \sum_{i,m,n}^{I,M_i,N_i} & I(jj[i,m,n] = j) \\ & I(z_{i,m} = h) I(y_{i,m,n} = 1) E_q[I(c_{i,m} = 0)] \end{aligned} \quad (7)$$

In Equations (8) and (9) we list the variational update formulas for the true label indicator. We have $q(c_{i,m}|\phi_{i,m}) = \mathsf{Bern}(\phi_{i,m})$, where $\zeta_{i,m} = 1 - \phi_{i,m}$ and:

$$\begin{aligned} \log \phi_{i,m} \propto\ & E_q[\log \pi_{z_{i,m}}] + \\ & + \sum_{n=1}^{N_i} I(y_{i,m,n} = 1) E_q[\log \alpha_{jj[i,m,n], z_{i,m}}] + \\ & + I(y_{i,m,n} = 0) E_q[\log(1 - \alpha_{jj[i,m,n], z_{i,m}})] \end{aligned} \quad (8)$$

$$\begin{aligned} \log \zeta_{i,m} \propto\ & E_q[\log(1 - \pi_{z_{i,m}})] + \\ & + \sum_{n=1}^{N_i} I(y_{i,m,n} = 0) E_q[\log \beta_{jj[i,m,n], z_{i,m}}] + \\ & + I(y_{i,m,n} = 1) E_q[\log(1 - \beta_{jj[i,m,n], z_{i,m}})] \end{aligned} \quad (9)$$

Finally, for the above formulas, we used the fact that $E_q[I(c_{i,m} = 1)] = \phi_{i,m}$. The other expectations can be easily calculated noting that for a distribution part of the exponential family, the first derivative of the log normalizer is equal to the expected value of the sufficient statistics (Blei et al., 2003). For example, $E_q[\log \pi_{z_{i,m}}] = \Psi(\lambda_{z_{i,m}}) - \Psi(\lambda_{z_{i,m}} + \eta_{z_{i,m}})$, where $\Psi(.)$ is the digamma function. Similar observations apply to the α and β related expectations.

The algorithm, known as Coordinate Ascent Variational Inference (CAVI) (Blei et al., 2017), involves iterating between Equations (2), (3), (4), (5), (6), (7), (8) and (9) until convergence. The ELBO expressed in Equation (1) is guaranteed to increase at every step. Convergence is achieved when the ELBO plateaus. Throughout the experiments we used non-informative, uniform priors.

3 Evaluation

We carried out a series of evaluations of increasing complexity of our MPA model. We first assess the accuracy of the inferred mention pairs. Second, we cluster the pairs into appropriate coreference chains and evaluate the quality of these chains. Third, we assess the viability of using silver chains as an alternative to the gold chains when training a state of the art coreference system. Finally, we conclude the evaluation with a performance check in a standard annotation task.

3.1 Datasets

The largest coreference dataset with crowdsourced annotations is the *Phrase Detectives* corpus. A subset of this corpus is the *Phrase Detectives 1.0* dataset (Chamberlain et al., 2016), which also includes gold annotations and can therefore be used to evaluate the accuracy of MPA at mention-pair and coreference chain inference, but is too small to train a state-of-the-art coreference system. To carry out this second type of evaluation we used the approach, common in the crowdsourcing literature (Carpenter, 2008; Raykar et al., 2010; Hovy et al., 2013; Felt et al., 2014), of generating simulated datasets by corrupting the gold standard

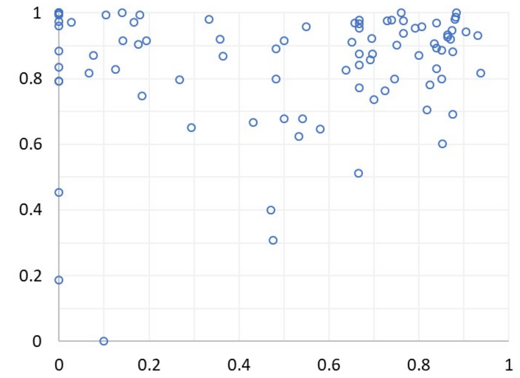

Figure 2: Sensitivity profiles extracted from the PD corpus: DO (x-axis) vs. DN (y-axis)

of an existing corpus. For this purpose, we use the CONLL-2012 dataset (Pradhan et al., 2012), at present the standard dataset for coreference resolution.

3.1.1 Crowdsourced Data

The *Phrase Detectives (PD) 1.0* dataset has been annotated using the *Phrase Detectives* game with a purpose.[4] The annotation scheme for PD is based on that for the ARRAU corpus (Poesio et al., 2018). Players have to label predefined[5] markables with one of the following categories: **non-referring** (e.g., for expletives), **discourse-new**, **discourse-old** (in which case an antecedent is also marked, the most recent mention belonging to the antecedent's coreference chain), or **property** (for appositions and copular structures). The PD 1.0 dataset is the portion of the corpus that contains, in addition to the annotations by the players, a gold label for each markable. The coreference chains are obtained using a simple clustering of the mention pairs. An important limitation of this corpus is its small size (around 6000 markables from 45 documents), making it unfit for the training and evaluation of state of the art supervised systems.

3.1.2 Synthetic Data

The CONLL-2012 dataset specifies gold chains, not mention pairs. So we need first to extract appropriate mention pairs from these chains. To do this, for each mention we select as gold label the closest mention from its gold chain (or discourse new if the mention is the first in its chain).

Simulation	Profile Type	Error Distribution
1	Synthetic	Uniform
2	Synthetic	Sparse
3	PD-inspired	Uniform
4	PD-inspired	Sparse

Table 1: Simulation summary

Data	Method	Accuracy avg.	Accuracy s.d.
PD 1.0	MV	84.32	-
	MPA	91.43	-
Synthetic Uniform	MV	85.09	0.52
	MPA	90.12	0.52
Synthetic Sparse	MV	76.55	0.46
	MPA	85.92	0.60
PD-inspired Uniform	MV	89.26	0.47
	MPA	97.38	0.28
PD-inspired Sparse	MV	82.72	0.56
	MPA	94.36	0.33

Table 2: Mention pair accuracy results. Each simulated scenario is randomly generated 10 times (summary is in terms of average result and standard deviation)

The simulations are then generated by extracting from each gold label a number of 'crowdsourced labels' produced by (simulated) annotators with varying degrees of ability. We considered a range of simulated scenarios, all sharing the following settings:

- 10 distinct annotators per mention and 20 distinct mentions per annotator. The annotators receive random mentions to annotate.[6]
- Each annotator is assigned randomly a **profile**. The profiles indicate the sensitivity of the annotators with respect to discourse old and new. For example, the (DO 0.8, DN 0.9) profile indicates that, given a mention whose true class is DO, the annotator has 0.8 probability of getting it right; and of 0.9 for DN. We considered both profiles reflecting the actual profiles of players in *Phrase Detectives* (Chamberlain, 2016) and synthetic profiles.
- 5 choices for the annotators to choose from for each mention: the correct label, the DN

---

[4] `http://www.phrasedetectives.org`

[5] In standard annotation projects markables are predefined for better agreement. The markables used in PD are automatically identified, but players can highlight errors in markable identification that can then be corrected.

[6] This equal load reflects work distribution as found in microtask crowdsourcing rather than in games such as *Phrase Detectives*, where a few players do most of the work (Chamberlain, 2016).

PD 1.0	Method	MUC			BCUB			CEAFE			Avg. F1
		P	R	F1	P	R	F1	P	R	F1	
Singletons included	MV	95.18	69.44	80.30	95.53	78.79	86.36	79.04	95.12	86.34	84.33
	MPA	92.87	86.07	89.34	94.79	88.56	91.57	90.53	94.27	92.36	91.09
	Stanford	65.55	59.70	62.49	79.83	74.54	77.09	77.74	85.41	81.40	73.66
Singletons excluded	MV	95.18	69.44	80.30	93.36	46.05	61.68	64.23	55.17	59.35	67.11
	MPA	92.87	86.07	89.34	88.46	72.83	79.89	79.65	76.32	77.95	82.39
	Stanford	65.55	59.70	62.49	51.09	39.16	44.33	41.44	49.02	44.91	50.58

Table 3: The quality of the coreference chains on the PD 1.0 dataset

PD 1.0	Method	P	R	F1
Non Referring scores	MV	82.98	20.00	32.23
	MPA	75.14	66.67	70.65

Table 4: Non-referring scores for the PD 1.0 dataset

label (without including it twice if this is the correct label), and the 3 (or 4 if the correct label is DN) incorrect DO antecedents situated closest to the mention.

The range of options considered in the simulation is specified by two aspects: the sensitivity from the annotator profiles and the distribution of the errors they make. We use the following two profile types:

- Synthetic profiles: 5 profiles covering a wide range of abilities (DO 0.8, DN 0.9), (DO 0.7, DN 0.8), (DO 0.4, DN 0.5), (DO 0.3, DN 0.4), (DO 0.2, DN 0.3). The profiles roughly correspond to two experts and three novices whose class sensitivities are relatively close – with extra mass associated with DN because this class is generally easier compared to DO.

- *Phrase Detectives* inspired profiles: from the PD annotators who annotated more than 10 DO and 10 DN mentions (thresholds set to have a minimum confidence) we extracted a total of 89 profiles. This gave us much more interesting sensitivity pairs compared to the ones from the synthetic profiles, i.e., contrasting class abilities – see Figure 2.

We also considered a range of ways in which annotators may make mistakes:

- Distribute the errors uniformly random given the remaining mass (1 - sensitivity)

- Distribute the errors in a sparse manner, i.e., assume that some errors will be more likely than others. This can be achieved by drawing randomly from a 4-dimensional (4 = number of errors) uniform Dirichlet for each mention. The annotator probabilities over the 5 choices will then consist of their sensitivity, and the error distribution normalized with respect to the remaining mass.

The settings just discussed lead to 4 simulations summarized in Table 1.

3.2 Evaluation 1: Mention Pair Accuracy

We use MPA to link each mention with the most likely label based on the posterior of the true label indicators. We then assess the accuracy of the inferred mention pairs against the gold standard, i.e., the agreement with the gold mention pairs. In this task the proposed model is compared against a majority vote baseline where each mention is paired with the most voted label.[7]

The evaluation is conducted on the crowd-sourced annotated PD 1.0 dataset and on simulated data generated from the CONLL-2012 test set. The results, summarized in Table 2, indicate the mention pairs inferred by our model (MPA) obtain a much better level of agreement with the gold mention pairs, compared with the output of the majority vote (MV) baseline. MV implicitly assumes equal expertise among the annotators, which has repeatedly been shown to be false in annotation practice (Poesio and Artstein, 2005; Passonneau and Carpenter, 2014; Plank et al., 2014).

3.3 Evaluation 2: Silver Chain Quality

After the mention pairs have been inferred using MPA, producing the coreference chains – we will henceforth refer to the coreference chains thus ob-

---

[7] Throughout the paper we report the best majority vote result after 10 random rounds of splitting ties.

CoNLL 2012 Test Dataset			MUC			BCUB			CEAFE			Avg. F1
Simulation	Method		P	R	F1	P	R	F1	P	R	F1	
None	Stanford		89.78	73.88	81.06	83.93	59.22	69.44	73.87	60.57	66.56	72.35
Synthetic Uniform	MV	avg.	88.27	86.00	87.12	73.92	70.81	72.33	70.62	76.73	73.55	77.67
		s.d.	0.38	0.35	0.36	0.83	0.52	0.62	0.49	0.60	0.50	0.47
	MPA	avg.	90.92	91.97	91.44	75.51	80.14	77.75	81.98	78.81	80.37	83.19
		s.d.	0.48	0.36	0.41	1.20	0.67	0.93	0.74	1.16	0.93	0.75
Synthetic Sparse	MV	avg.	81.99	79.01	80.47	65.91	62.64	64.23	60.61	68.00	64.09	69.59
		s.d.	0.32	0.43	0.38	0.45	0.51	0.40	0.39	0.24	0.28	0.32
	MPA	avg.	87.90	88.24	88.07	70.67	73.91	72.25	74.62	73.63	74.12	78.15
		s.d.	0.47	0.44	0.45	0.96	0.65	0.77	0.75	0.93	0.82	0.66
PD-inspired Uniform	MV	avg.	91.84	88.28	90.02	80.94	74.19	77.41	75.13	84.93	79.73	82.39
		s.d.	0.36	0.49	0.42	0.61	0.84	0.66	0.88	0.55	0.72	0.58
	MPA	avg.	97.42	97.20	97.31	91.61	91.53	91.57	93.87	94.58	94.23	94.37
		s.d.	0.27	0.28	0.27	1.05	1.28	1.15	0.67	0.61	0.63	0.68
PD-inspired Sparse	MV	avg.	86.86	81.70	84.20	74.26	65.42	69.56	65.45	78.51	71.39	75.05
		s.d.	0.49	0.54	0.51	0.63	0.48	0.52	0.67	0.55	0.60	0.53
	MPA	avg.	94.86	94.09	94.47	85.24	84.42	84.83	87.52	89.91	88.70	89.33
		s.d.	0.32	0.36	0.34	0.70	0.75	0.71	0.60	0.49	0.54	0.52

Table 5: The quality of the coreference chains on the CoNLL-2012 test set. Each simulated scenario is randomly generated 10 times (summary reported in terms of average result and standard deviation)

tained as **silver** coreference chains[8] – is a straightforward clustering task: we simply follow the link structure from the pairs. In this Section we assess the quality of the silver chains using standard coreference metrics – in particular, the Extended Scorer introduced in (Poesio et al., 2018) which extends the official CONLL scorer to include in the evaluation system-predicted singletons and non referring expressions, both of which are annotated in *Phrase Detectives*; when singletons and non-referring expressions are not considered, the Extended Scorer is identical to the official scorer.

As in the previous experiment, the evaluation is conducted on the crowdsourced annotated PD 1.0 dataset and on simulated data generated from the CONLL-2012 test set. We compare silver chains produced using our MPA model, using MV, and using the Stanford deterministic coreference system (Stanford) (Lee et al., 2011). To run the latter on PD 1.0, we used the default annotators of the CoreNLP toolkit (Manning et al., 2014) to supply the information required by the coreference system and switched off the post-processing to output singleton clusters; for the CONLL-2012 data we set the $dcoref.replicate.conll = true$ to run exactly the same method as Lee et al. (2011). On both datasets we evaluated on gold mentions.

Table 3 summarizes the results on the crowdsourced annotated PD 1.0 dataset. The silver chains obtained using our MPA model are of a far better quality than those of baseline alternatives such as MV and Stanford. Note also that even the simple MV baseline built from crowdsourced annotations yields much better chains compared to a standard coreference system such as the Stanford system. This underlines the advantage of crowdsourced annotations for coreference over automatically produced annotations. In Table 4 we present the scores of MPA and MV on cases of non referring. In this case, as well, the probabilistic model substantially outperforms the MV baseline.

In Table 5 we present the results obtained on simulated data from the CONLL-2012 test set. The results follow a similar trend to those observed using actual annotations: a much better quality of the chains produced using the mention pairs inferred by our MPA model, across all the simulated

---

[8] Our use of the term 'silver standard' should not be confused with the other common use of standard generated out of automatic annotations.

			MUC			BCUB			CEAFE			Avg. F1
Simulation	Method		P	R	F1	P	R	F1	P	R	F1	
None	Gold		78.40	73.40	75.80	68.60	61.80	65.00	62.70	59.00	60.80	67.20
	Stanford		79.87	63.67	70.86	71.63	47.85	57.37	58.55	48.08	52.80	60.34
Synthetic Uniform	MV	avg.	78.67	67.51	72.65	67.68	51.41	58.41	59.59	52.63	55.89	62.32
		s.d.	0.87	0.73	0.13	1.48	0.99	0.27	0.62	0.64	0.34	0.22
	MPA	avg.	78.27	70.21	74.02	66.67	56.06	60.90	61.44	54.92	57.99	64.30
		s.d.	0.64	0.57	0.23	1.03	0.81	0.31	0.50	0.51	0.27	0.24
Synthetic Sparse	MV	avg.	77.95	64.45	70.55	66.80	47.21	55.29	57.64	49.64	53.34	59.73
		s.d.	0.75	1.18	0.52	1.17	1.75	0.89	0.64	1.10	0.76	0.72
	MPA	avg.	77.99	68.82	73.11	66.01	53.76	59.25	60.25	53.46	56.65	63.01
		s.d.	0.55	0.51	0.25	0.93	0.84	0.42	0.43	0.38	0.27	0.28
PD-inspired Uniform	MV	avg.	78.99	68.44	73.33	68.42	52.86	59.63	59.99	54.04	56.85	63.27
		s.d.	0.58	0.60	0.14	0.92	0.75	0.38	0.79	0.27	0.44	0.27
	MPA	avg.	78.35	72.39	75.25	67.65	59.89	63.53	62.32	57.70	59.92	66.23
		s.d.	0.24	0.34	0.15	0.58	0.42	0.20	0.21	0.38	0.25	0.14
PD-inspired Sparse	MV	avg.	78.72	65.46	71.47	67.99	48.43	56.55	58.29	51.33	54.59	60.87
		s.d.	0.70	0.68	0.33	1.43	0.96	0.55	0.57	0.62	0.53	0.45
	MPA	avg.	78.34	71.47	74.75	67.43	58.11	62.42	61.90	56.77	59.22	65.46
		s.d.	0.44	0.62	0.20	0.83	0.94	0.26	0.27	0.49	0.28	0.23

Table 6: Results of a state of the art coreference system trained on silver chains obtained in different ways. Each simulated scenario is randomly generated 10 times (summary is in terms of average result and standard deviation)

scenarios. Furthermore, the MV baseline achieves better chains compared to the Stanford system in 3 out of 4 simulation settings, again showcasing the potential of crowdsourced annotations.

3.4 Training on Silver Chains

In this Section we assessed the viability of using the (silver) chains extracted from crowdsourcing as an alternative to gold chains when training a state of the art coreference system. Concretely, we train the best-performing current system Lee et al. (2017) on chains produced using our MPA model, the MV baseline and the Stanford deterministic system (Lee et al., 2011) (used mainly for calibration, i.e., an alternative baseline that's not based on crowdsourced annotations). We also include the results obtained using actual gold chains.

The results are in Table 6. Across all simulated scenarios, the silver chains produced by our MPA model obtain the closest performance to training on gold chains, and the best result is only 1 percentage point less than the result with gold chains. Again, the MV chains lead to better performance than those obtained using a system (Stanford). These results, once again, indicate the utility of crowdsourced annotations for coreference tasks.

3.5 Traditional Crowdsourcing Tasks

In this Section we show that MPA is state of the art also on traditional crowdsourcing datasets, where annotations fall into general classes that are consistent across the annotated items. This evaluation was done on the datasets (*WSD*, *RTE* and *TEMP*) introduced by Snow et al. (2008) and widely used as benchmarks in the literature on annotation models (Hovy et al., 2013; Carpenter, 2008).

We compare the results against a majority vote baseline and two well-known state of the art models: a Bayesian version of the Dawid and Skene (1979) (DS) model and MACE (Hovy et al., 2013). We implement DS ourselves using variational inference, while for MACE, we simply report the published results. As in Hovy et al. (2013) the assessment is done in terms of accuracy against the gold standard. The results, presented in Table 7, indicate the proposed model achieves performance on par with the state of the art.

4 Related Work

To our knowledge, this is the first paper proposing a model of crowdsourced annotations for coreference. We did draw inspiration however from existing mention pair models of coreference and traditional models of annotation.

The so-called mention pair model is one of the early machine learning approaches to coreference resolution, made popular by Soon et al. (2001). The model is based on a two step procedure: a classification step which identifies the coreferent mention pairs, followed by a clustering step which builds the coreference chains from the aforementioned pairs. The diversity of mention pair models comes from the distinct approaches taken for each of the two steps (Hoste, 2016). Although we follow a similar two step procedure, there are also important differences. Our way of identifying the mention pairs is completely unsupervised, and relies entirely on the crowdsourced annotations. Furthermore, we pair every mention with only one label, reducing the second step of clustering mention pairs into appropriate coreference chains to a simple grouping task guided by a unique path which arises from the pairs.

All existing probabilistic models of annotation (Dawid and Skene, 1979; Smyth et al., 1994; Raykar et al., 2010; Hovy et al., 2013; Passonneau and Carpenter, 2014) assume the annotations fall into a general set of classes that is consistent across the annotated items. This is clearly not the case in a coreference resolution task, a limitation we had to address. We first transformed the annotations into a series of (per label) binary decisions, approach often referred to, in the multi-class classification literature, as the binary relevance method (Tsoumakas and Katakis, 2007; Madjarov et al., 2012). The transformation avoids modeling the sparse labels directly. We further exploited the fact that the annotations fall into a general set of classes and assumed the inter-label decisions are the result of the class-dependent ability of the annotators.

5 Conclusions

Crowdsourced annotations are an increasingly popular alternative to expert annotation. Even so, their viability for coreference annotation had not been explored so far. This paper is a first step to filling this gap. We introduced a mention pair-based approach to aggregating crowdsourced anaphoric annotations and assessed the quality of the inferred pairs, of the post-hoc constructed coreference chains, and the viability of using the inferred chains as an alternative to gold chains when training a state of the art coreference system. Throughout the experiments, the model introduced was superior to baseline alternatives such as majority vote and chains obtained automatically using a coreference system, across both genuinely crowdsourced and simulated coreference datasets. Furthermore, even the annotation-based baseline achieved results consistently better than those obtained by automatic coreference resolvers, strengthening the case for using crowdsourced annotations to create coreference datasets.

	RTE	TEMP	WSD
MV	90.00	93.00	99.00
MACE	93.00	94.00	99.00
DS	93.00	94.00	99.00
MPA	93.00	94.00	99.00

Table 7: Accuracy on standard crowdsourced data

Acknowledgments

Paun, Chamberlain, Juntao and Poesio are supported by the DALI project, funded by ERC.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596. An early version of this paper has been circulating since 2005 as “Kappa^3 = Alpha (or Beta)”. This version is still available from the ARRAU website.

Breck Baldwin and Thomas S. Morton. 1998. Dynamic coreference-based summarization. In *Proceedings of the Third Conference on Empirical Methods for Natural Language Processing*, pages 1–6.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 47–57.

David M. Blei and Michael I. Jordan. 2006. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Johan Bos, Valerio Basile, Kilian Evang, Noortje J. Venhuizen, and Johannes Bjerva. 2017. The Groningen Meaning Bank. In *The Handbook of Linguistic Annotation*, chapter 18, pages 463–496. Springer.

Bob Carpenter. 2008. Multilevel Bayesian models of categorical data annotation. Available at http://lingpipe-blog.com/lingpipe-white-papers.

Jon Chamberlain. 2016. *Harnessing Collective Intelligence on Social Networks*. Ph.D. thesis, University of Essex, School of Computer Science and Electronic Engineering.

Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. 2016. Phrase detectives corpus 1.0: Crowdsourced anaphoric coreference. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2016)*, Portoroz, Slovenia.

Nancy A. Chinchor. 1998. Overview of MUC-7/MET-2. In *Proceedings of the* 7^{th} *Message Understanding Conference (MUC7)*.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Alexander Philip Dawid and Allan M. Skene. 1979. Maximum likelihood estimation of observer errorrates using the EM algorithm. *Applied Statistics*, 28(1):20–28.

Paul Felt, Robbie Haertel, Eric K. Ringger, and Kevin D. Seppi. 2014. MOMRESP: A Bayesian model for multi-annotator document labeling. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2014)*, Reykjavik.

Ralph Grishman and Beth Sundheim. 1995. Design of the MUC-6 evaluation. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pages 1–11, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anupam Guha, Mohit Iyyer, Danny Bouman, and Jordan Boyd-Graber. 2015. Removing the training wheels: A coreference dataset that entertains humans and challenges computers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Veronique Hoste. 2016. The mention-pair model. In *Anaphora Resolution: Algorithms, Resources and Applications*. Springer.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130.

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. Event coreference for information extraction. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, CONLL Shared Task '11, pages 28–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197.

Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and SašO Deroski. 2012. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084–3104.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association of Computational Linguistics*, 3(1):405–418.

Thomas S. Morton. 1999. Using coreference for question answering. In *Proceedings of the Workshop on Coreference and Its Applications*, CorefApp '99, pages 85–89, Stroudsburg, PA, USA. Association for Computational Linguistics.

Malvina Nissim, Shipra Dingare, Jean Carletta, and Mark Steedman. 2004. An annotation scheme for information status in dialogue. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2004)*.

Rebecca J. Passonneau. 2004. Computing reliability for coreference annotation. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

Barbara Plank, Dirk Hovy, and Anders Sogaard. 2014. Linguistically debatable or just plain wrong? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Massimo Poesio and Ron Artstein. 2005. The reliability of anaphoric annotation, reconsidered: Taking ambiguity into account. In *Proceedings of ACL Workshop on Frontiers in Corpus Annotation*, pages 76–83.

Massimo Poesio, Jon Chamberlain, and Udo Kruschwitz. 2017. Crowdsourcing. In *Handbook of Linguistic Annotation*. Springer.

Massimo Poesio, Jon Chamberlain, Udo Kruschwitz, Livio Robaldo, and Luca Ducceschi. 2013. Phrase Detectives: Utilizing collective intelligence for internet-scale language resource creation. *ACM Transactions on Intelligent Interactive Systems*, 3(1).

Massimo Poesio, Yulia Grishina, Varada Kolhatkar, Nafise Moosavi, Ina Roesiger, Adam Roussel, Fabian Simonjetz, Alexandra Uma, Olga Uryupina, Juntao Yu, and Heike Zinsmeister. 2018. Anaphora resolution with the ARRAU corpus. In *Proceedings of the NAACL Workshop on Computational Models of Reference, Anaphora and Coreference (CRAC)*, pages 11–22, New Orleans.

Massimo Poesio, Sameer Pradhan, Marta Recasens, Kepa Rodriguez, and Yannick Versley. 2016a. Annotated corpora and annotation tools. In *Anaphora Resolution: Algorithms, Resources and Applications*. Springer.

Massimo Poesio, Roland Stuckardt, and Yannick Versley. 2016b. *Anaphora Resolution: Algorithms, Resources and Applications*. Springer, Berlin.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning (CoNLL 2012)*, Jeju, Korea.

Sameer S. Pradhan, Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2007. Ontonotes: A unified relational semantic representation. In *International Conference on Semantic Computing (ICSC 2007)*, pages 517–526.

Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Lam Ngoc Tran, and Karl Aberer. 2013. An evaluation of aggregation techniques in crowdsourcing. In *Web Information Systems Engineering – WISE 2013*, pages 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322.

Arndt Riester, David Lorenz, and Nina Seeman. 2010. A recursive annotation scheme for referential information status. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2010)*, pages 717–722.

Aashish Sheshadri and Matthew Lease. 2013. SQUARE: A Benchmark for Research on Computing Crowd Consensus. In *Proceedings of the 1st AAAI Conference on Human Computation (HCOMP)*, pages 156–164.

Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. 1994. Inferring ground truth from subjective labelling of Venus images. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, NIPS'94, pages 1085–1092, Cambridge, MA, USA. MIT Press.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Josef Steinberger, Massimo Poesio, Mijail Kabadjov, and Karel Jezek. 2007. Two uses of anaphora resolution in summarization. *Information Processing and Management*, 43(6):1663–1680. Special issue on Summarization.

Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 2007:1–13.

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. Ontonotes: A large training corpus for enhanced processing. *Handbook of Natural Language Processing and Machine Translation. Springer.*

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. 2009.

Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems 22*, pages 2035–2043. Curran Associates, Inc.

Zhiping Zheng. 2002. Answerbus question answering system. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 399–404, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

A Supplementary Parameter Estimation Details

This Section gives supplementary details for the derivations involved in the parameter estimation process.

In Equation (10) we derive the complete conditional of the class-specific true label likelihood:

$$\begin{aligned}
p(\pi_h|...) &\propto p(\pi_h|a,b) \prod_{i,m}^{I,M_i} p(c_{i,m}|\pi_h)^{I(z_{i,m}=h)} \\
&\propto \pi_h^{a-1}(1-\pi_h)^{b-1} \times \\
&\times \prod_{i,m}^{I,M_i} \pi_h^{I(c_{i,m}=1)I(z_{i,m}=h)} \\
&\times (1-\pi_h)^{I(c_{i,m}=0)I(z_{i,m}=h)} \\
&\propto \pi_h^{a-1+\sum_{i,m}^{I,M_i} I(z_{i,m}=h)I(c_{i,m}=1)} \\
&(1-\pi_h)^{b-1+\sum_{i,m}^{I,M_i} I(z_{i,m}=h)I(c_{i,m}=0)} \\
&\propto \mathsf{Beta}(a + \sum_{i,m}^{I,M_i} I(z_{i,m}=h)I(c_{i,m}=1), \\
b &+ \sum_{i,m}^{I,M_i} I(z_{i,m}=h)I(c_{i,m}=0))
\end{aligned} \tag{10}$$

The corresponding variational distribution has the same form, i,e, $q(\pi_h|\lambda_h, \eta_h) = \mathsf{Beta}(\lambda_h, \eta_h)$. Taking the expectation of the natural parameters of the above distribution gives the variational update formulas for the class-level true label likelihood expressed in the paper. For example, for the λ_h variational parameter we have:

$$\begin{aligned}
\lambda_h &= a + \sum_{i,m}^{I,M_i} I(z_{i,m}=h) E_q[I(c_{i,m}=1)] \\
&= a + \sum_{i,m}^{I,M_i} I(z_{i,m}=h) \phi_{i,m}
\end{aligned} \tag{11}$$

Similar steps were taken to derive the variational parameters associated with the sensitivity α and specificity β.

In Equation (12) we derive the complete conditional associated with the positive outcome of the true label indicator:

$$\begin{aligned}
p(c_{i,m}=1|...) &\propto p(c_{i,m}=1|\pi_{z_{i,m}}) \times \\
&\times \prod_{n}^{N_i} p(y_{i,m,n}|\alpha_{jj[i,m,n],z_{i,m}}) \\
&\propto \pi_{z_{i,m}} \prod_{n}^{N_i} \alpha_{jj[i,m,n],z_{i,m}}^{I(y_{i,m,n}=1)} \\
&(1-\alpha_{jj[i,m,n],z_{i,m}})^{I(y_{i,m,n}=0)} \\
&\propto \exp\{\log \pi_{z_{i,m}} + \\
&\sum_{n=1}^{N_i} I(y_{i,m,n}=1) \log \alpha_{jj[i,m,n],z_{i,m}} + \\
&I(y_{i,m,n}=0) \log(1-\alpha_{jj[i,m,n],z_{i,m}})\}
\end{aligned} \tag{12}$$

The corresponding variational distribution has the same form, i,e, $q(c_{i,m}|\phi_{i,m}) = \mathsf{Bern}(\phi_{i,m})$. Taking the necessary expectations leads to the update formula expressed in the paper. Concretely, we have:

$$\begin{aligned}
\log \phi_{i,m} &\propto E_q[\log \pi_{z_{i,m}}] + \\
&+ \sum_{n=1}^{N_i} I(y_{i,m,n}=1) E_q[\log \alpha_{jj[i,m,n],z_{i,m}}] + \\
&+ I(y_{i,m,n}=0) E_q[\log(1-\alpha_{jj[i,m,n],z_{i,m}})] \\
&\propto \Psi(\lambda_{z_{i,m}}) - \Psi(\lambda_{z_{i,m}} + \eta_{z_{i,m}}) + \\
&+ \sum_{n=1}^{N_i} I(y_{i,m,n}=1)[\Psi(\gamma_{jj[i,m,n],z_{i,m}}) - \\
&\Psi(\gamma_{jj[i,m,n],z_{i,m}} + \mu_{jj[i,m,n],z_{i,m}})] + \\
&+ I(y_{i,m,n}=0)[\Psi(\mu_{jj[i,m,n],z_{i,m}}) - \\
&\Psi(\gamma_{jj[i,m,n],z_{i,m}} + \mu_{jj[i,m,n],z_{i,m}})]
\end{aligned} \tag{13}$$

The update formula for the negative outcome of the true label indicator $\zeta_{i,m}$ is derived in a similar manner. Following the above derivations should also make it straightforward to expand the ELBO.

For completeness, we make a note of the digamma function $\Psi()$ – this is the first derivative of the $\log \Gamma$ function and can be computed using a Taylor approximation.

A Deterministic Algorithm for Bridging Anaphora Resolution

Yufang Hou
IBM Research Ireland
`yhou@ie.ibm.com`

Abstract

Previous work on bridging anaphora resolution (Poesio et al., 2004; Hou et al., 2013b) use syntactic preposition patterns to calculate word relatedness. However, such patterns only consider NPs' head nouns and hence do not fully capture the semantics of NPs. Recently, Hou (2018) created word embeddings (*embeddings_PP*) to capture associative similarity (i.e., relatedness) between nouns by exploring the syntactic structure of noun phrases. But *embeddings_PP* only contains word representations for nouns. In this paper, we create new word vectors by combining *embeddings_PP* with *GloVe*. This new word embeddings (*embeddings_bridging*) are a more general lexical knowledge resource for bridging and allow us to represent the meaning of an NP beyond its head easily. We therefore develop a deterministic approach for bridging anaphora resolution, which represents the semantics of an NP based on its head noun and modifications. We show that this simple approach achieves the competitive results compared to the best system in Hou et al. (2013b) which explores Markov Logic Networks to model the problem. Additionally, we further improve the results for bridging anaphora resolution reported in Hou (2018) by combining our simple deterministic approach with Hou et al. (2013b)'s best system *MLN II*.

1 Introduction

Anaphora plays a major role in discourse comprehension and accounts for the coherence of a text. In contrast to *identity anaphora* which indicates that a noun phrase refers back to the same entity introduced by previous descriptions in the discourse, *bridging anaphora* or *associative anaphora* links anaphors and antecedents via lexico-semantic, frame or encyclopedic relations. *Bridging resolution* is the task to recognize bridging anaphors (e.g., **distribution arrangements** in Example 1[1]) and find links to their antecedents (e.g., *dialysis products* in Example 1).

(1) While the discussions between Delmed and National Medical Care have been discontinued, Delmed will continue to supply *dialysis products* through National Medical after their exclusive agreement ends in March 1990, Delmed said. In addition, Delmed is exploring **distribution arrangements** with Fresenius USA, Delmed said.

Most previous empirical research on bridging (Poesio and Vieira, 1998; Poesio et al., 2004; Markert et al., 2003; Lassalle and Denis, 2011; Hou et al., 2013b) focus on *bridging anaphora resolution*, a subtask of bridging resolution that aims to choose the antecedents for bridging anaphors. For this substask, most previous work (Poesio et al., 2004; Lassalle and Denis, 2011; Hou et al., 2013b) calculate semantic relatedness between an anaphor and its antecedent based on word co-occurrence counts using certain syntactic patterns. However, such patterns only consider head noun knowledge and hence are not sufficient for bridging relations which require the semantics of modification. In Example 1, in order to find the antecedent (*dialysis products*) for the bridging anaphor "**distribution arrangements**", we have to understand the semantics of the modification "**distribution**".

Over the past few years, word embeddings gained a lot popularity in the NLP community. State-of-the-art word vectors such as word2vec skip-gram (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) have been shown to perform well across a variety of NLP tasks, including textual entailment (Rocktäschel et al., 2016), reading

---

[1] All examples, if not specified otherwise, are from ISNotes (Markert et al., 2012). Bridging anaphors are typed in boldface, antecedents in italics throughout this paper.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1938–1948
Brussels, Belgium, October 31 - November 4, 2018.

comprehension (Chen et al., 2016) and coreference resolution (Lee et al., 2017).

Recently, Hou (2018) found that these vanilla word embeddings capture both "genuine" similarity and relatedness, and hence they are not suitable for bridging anaphora resolution which requires lexical association knowledge instead of semantic similarity information between synonyms or hypernyms. Hou (2018) created word embeddings for bridging (*embeddings_PP*) by exploring the syntactic structure of noun phrases (NPs) to derive contexts for nouns in the GloVe model.

However, *embeddings_PP* only contains the word representations for nouns. In this paper, we improve *embeddings_PP* by combining it with *GloVe*. The resulting word embeddings (*embeddings_bridging*) are a more general lexical knowledge resource for bridging anaphora resolution. Compared to *embeddings_PP*, the coverage of lexicon in *embeddings_bridging* is much larger. Also the word representations for nouns without the suffix "_PP" are more accurate because they are trained on many more instances in the vanilla *GloVe*. Based on this general vector space, we develop a deterministic algorithm to select antecedents for bridging anaphors. Our approach combines the semantics of an NP's head with the semantics of its modifications by vector average using *embeddings_bridging*. We show that this simple, efficient method achieves the competitive results on ISNotes for the task of bridging anaphora resolution compared to the best system in Hou et al. (2013b) which explores Markov Logic Networks to model the problem.

The main contributions of our work are: (1) a general word representation resource[2] for bridging; and (2) a simple yet competitive deterministic approach for bridging anaphora resolution which models the meaning of an NP based on its head noun and modifications.

2 Related Work

Lexical/world knowledge for bridging: Hou et al. (2013a) explored various lexico-semantic features for bridging anaphora recognition. Hou (2016) proposed an attention-based LSTM model with pre-trained word embeddings for information status classification and reported moderate results for bridging recognition. Previous work on bridging anaphora resolution (Poesio et al., 2004; Lassalle and Denis, 2011; Hou et al., 2013b) explored word co-occurrence counts in certain syntactic preposition patterns to calculate word relatedness. For instance, the big hit counts of the query "*the door of the house*" in large corpora could indicate that *door* and *house* stand in a part-of relation. These patterns encode associative relations between nouns which cover a variety of bridging relations. Unlike previous work which only consider a small number of prepositions per anaphor, the PP context model (Hou, 2018) uses all prepositions for all nouns in big corpora. It also includes the possessive structure of NPs. In this paper, we further improve Hou (2018)'s *embeddings_PP* by combining it with the vanilla *GloVe*. The resulting word embeddings (*embeddings_bridging*) are a more general lexical knowledge resource for bridging resolution. In addition, it enables efficient computation of word association strength through low-dimensional matrix operations.

Bridging anaphora resolution: regarding the methods to select antecedents for bridging anaphors, Poesio et al. (2004) applied a pairwise model combining lexical semantic features as well as salience features to perform mereological bridging resolution in the GNOME corpus[3]. To address the data sparseness problem (e.g., some part-of relations are not covered by WordNet), they used the Web to estimate the part-of relations expressed by certain syntactic constructions. Based on the method proposed by Poesio et al. (2004), Lassalle and Denis (2011) developed a system that resolves mereological bridging anaphors in French. The system was enriched with meronymic information extracted from raw texts. Such information was extracted in a bootstrapping fashion by iteratively collecting meronymic pairs and the corresponding syntactic patterns. Lassalle and Denis (2011) evaluated their system on mereological bridging anaphors annotated in the DEDE corpus and reported an accuracy of 23%.

Markert et al. (2012) released a corpus called ISNotes which contains unrestricted bridging annotations. Based on this corpus, Hou et al. (2013b) proposed a joint inference framework for bridging anaphora resolution using Markov logic networks (Domingos and Lowd, 2009). The framework resolves all bridging anaphors in one document together by modeling that semantically re-

---

[2] *embeddings_bridging* can be downloaded from `https://doi.org/10.5281/zenodo.1403164`

[3] The GNOME corpus is not publicly available.

Noun Phrases	Extracted Noun Pairs
travelers *in the train station*	*travelers_PP – station*
travelers *from the airport*	*travelers_PP – airport*
hotels *for travelers*	*hotels_PP – travelers*
the ***destination*** *for travelers*	*destination_PP – travelers*
the company's new appointed ***chairman***	*chairman_PP – company*

Table 1: Examples of noun phrases as well as the extracted noun pairs in *embeddings_PP*. Bold indicates the head noun of an NP.

lated anaphors are likely to share the same antecedent.

ISNotes is a challenging corpus for bridging. First, bridging anaphors are not limited to definite NPs as in previous work (Poesio et al., 1997, 2004; Lassalle and Denis, 2011). Also in ISNotes, the semantic relations between anaphor and antecedent are not restricted to meronymic relations. We therefore choose ISNotes to evaluate our algorithm for bridging anaphora resolution. Our approach is deterministic and simple, but achieves the competitive results compared to the advanced machine learning-based approach (Hou et al., 2013b). We also improve the result reported in Hou (2018) on the same corpus by combining our deterministic approach with the best system from Hou et al. (2013b).

Just recently, two new corpora (Rösiger, 2018a; Poesio et al., 2018) with bridging annotations have become available and we notice that the definitions of bridging in these corpora are different from the bridging definition in ISNotes. We apply our algorithm with small adaptations to select antecedents for bridging anaphors on these corpora. The moderate results demonstrate that *embeddings_bridging* is a general word representation resource for bridging.

3 Word Representations for Bridging

3.1 Word Embeddings Based on PP Contexts (*embeddings_PP*)

We briefly describe Hou (2018)'s *embeddings_PP* in this section. *embeddings_PP* released by Hou (2018) contains 100-dimensional vectors for 276k nouns. It is trained over 197 million noun pairs extracted from the automatically parsed Gigaword corpus (Parker et al., 2011; Napoles et al., 2012). The author generates these noun pairs by exploring the syntactic prepositional and possessive structures of noun phrases. These two structures encode a variety of bridging relations between anaphors and their antecedents. For instance, the prepositional structure in "*the door of the house*" indicates the part-of relation between "*door*" and "*house*". More specifically, for NPs containing the prepositional structure (e.g., **X** *preposition* **Y**) or the possessive structure (e.g., **Y** *'s* **X**), the author extracts the noun pair "X_PP–Y". Note that the head of the NP is always on the left and the noun modifier is always on the right. In addition, the suffix "_PP" is added for the nouns on the left. Table 1 shows a few examples of noun phrases together with the extracted noun pairs.

Hou (2018) showed that the suffix "_PP" plays an important role for the model to learn the asymmetric relations between the head nouns and their noun modifiers from the extracted noun pairs. For instance, among the top five nearest neighbors in *embeddings_PP*, "president_PP" is mostly related to countries or organizations (e.g., "federation", "republic", or "USA"), while "president" is mostly related to words which have the same semantic type as "president" (e.g., "minister", "mayor", or "governor").

3.2 Word Representations for Bridging (*embeddings_bridging*)

embeddings_PP described in the previous section only contains word representations for nouns. To improve the coverage of lexical information, we create a general word representation resource *embeddings_bridging* by merging *embeddings_PP* with the original GloVe vectors trained on Gigaword and Wikipedia datasets. Specifically, given the 100 dimension word embeddings *embeddings_PP* and *GloVe*, we first create a 100 dimension vector v_{filler} with the value of each dimension as 0.1[4]. Let $v1_w$ represent the vector for the word w in *GloVe*, $v2_w$ represent the vector for the word w in *embeddings_PP*, if a word w appears

---

[4] Theoretically, any 100 dimension random vector with uniform distribution could be used as v_{filler}.

	embeddings_bridging	*embeddings_PP*
dimension size	200	100
vocabulary size	532,768	276,326
word type	all	nouns

Table 2: Comparison between *embeddings_bridging* and *embeddings_PP*

Category	**Relation**	**prototypical word pair example**	***embeddings _bridging***	***embeddings _PP***	**GloVe**
PART–WHOLE	*Object: Component*	{face: nose}	**0.43**	0.27	0.40
PART–WHOLE	*Event: Feature*	{wedding: bride}	**0.46**	0.29	0.15
PART–WHOLE	*Creature: Possession*	{author: copyright}	**0.23**	0.14	0.21
PART–WHOLE	*Activity: Stage*	{buying: shopping}	**0.32**	0.30	0.25

Table 3: Spearman's rank correlation coefficient ρ for typical *PART–WHOLE* bridging relations using *embeddings_bridging*, *embeddings_PP* and *GloVe*.

both in *GloVe* and in *embeddings_PP*, its vector in *embeddings_bridging* is the concatenation of $v1_w$ and $v2_w$. For the word w_1 which only appears in *GloVe*, its vector in *embeddings_bridging* is the concatenation of $v1_{w_1}$ and v_{filler}. Finally, for the word w_2 which only appears in *embeddings_PP* (all the words with the suffix "*_PP*"), we construct its vector by concatenating v_{filler} and $v2_{w_2}$. The resulting 200 dimension word embeddings (*embeddings_bridging*) is a general lexical resource for bridging. Table 2 compares the main features between *embeddings_bridging* and *embeddings_PP*. In the next section, we will compare *embeddings_bridging* with *embeddings_PP* and the original *GloVe* on a few typical bridging relations in the task of measuring relational similarity (Jurgens et al., 2012). Moreover, in Section 5.3 and Section 5.4, we show that using *embeddings_bridging* yields better results than using *embeddings_PP* for bridging anaphora resolution.

3.3 Measuring Relational Similarity on Typical Bridging Relations

We evaluate our *embeddings_bridging* quantitatively using a few typical bridging relations from SemEval-2012 Task 2 (Jurgens et al., 2012). The shared task aims to rank word pairs by the degree to which they are prototypical members of a given relation class. For instance, given the prototypical word pairs {*wedding–bride, rodeo–cowboy, banquet–food*} for the relation *Event:Feature*, we would like to know among the input word pairs {*school–students, circus–clown, meal–food, lion–zoo*}, which one represents the relation best.

SemEval-2012 Task 2 contains 79 relation classes chosen from Bejar et al. (1991). These relations fall into ten main categories, including *SIMILAR*, *PART–WHOLE*, *CONTRAST* and more. Each relation class is paired with a few prototypical word pairs and a list of around 40 word pairs which are ranked by humans according to their degree of similarity to the corresponding relation. We choose all typical bridging relations under the *PART–WHOLE* category and evaluate our *embeddings_bridging* in terms of ranking the list of word pairs for each relation. Spearman's rank correlation coefficient ρ is used to evaluate a system by comparing the system's ranking of the word pairs against the gold standard ranking.

Following Zhila et al. (2013), we calculate the relational similarity between word pairs using cosine similarity. Let (w_1, w_2) and (w_3, w_4) be the two word pairs, v_1, v_2, v_3, v_4 be the corresponding vectors for these words. We first normalize all word vectors to unit vectors, then the relational similarity between (w_1, w_2) and (w_3, w_4) is calculated as:

$$\frac{(v1 - v2) \cdot (v3 - v4)}{\parallel v1 - v2 \parallel \parallel v3 - v4 \parallel} \tag{1}$$

For each chosen relation class, we rank the list of word pairs according to their mean relational similarity to the given prototypical word pairs. Table 3 shows the results of Spearman's rank correlation coefficient ρ for each typical bridging relation using *embeddings_bridging*, *embeddings_PP*, and *GloVe*, respectively. Note that when using *embeddings_bridging* and *embeddings_PP*, we add the suffix "_PP" to the potential bridging anaphor for

bridging anaphor: **distribution arrangements**				
Ante. Candidates	**Head**	**Head + Modifiers**	$dist_h$	$dist_{hm}$
the discussions between Delmed and National Medical Care	{*discussions*}	{*discussions*}	0.05	-0.10
Delmed	{*delmed*}	{*delmed*}	—	—
National Medical Care	{*care*}	{*care*}	**0.08**	0.10
dialysis products	{*products*}	{*dialysis, products*}	0.06	**0.17**
National Medical	{*medical*}	{*medical*}	0.02	-0.01
their	{*their*}	{*their*}	-0.05	-0.01
their exclusive agreement	{*agreement*}	{*exclusive, agreement*}	0.07	0.03

Table 4: The cosine similarities between the bridging anaphor **distribution arrangements** and its antecedent candidates for Example 1. $dist_h$ indicates the cosine similarity between {**arrangements_PP**} and the candidate head, $dist_{hm}$ the cosine similarity between {**distribution_PP**, **arrangements_PP**} and Head+Modifiers. "–" means *Delmed* is not present in *embeddings_bridging* and therefore we neglect this candidate.

each word pair (e.g., {wedding: bride_PP}). As shown in Table 3, using *embeddings_bridging* performs better than both using *embeddings_PP* and using the vanilla GloVe vectors on these four part-of relation classes. This partially indicates that *embeddings_bridging* could capture lexical knowledge for bridging relations.

4 A Deterministic Algorithm for Bridging Anaphora Resolution

In this section, we describe our deterministic algorithm based on *embeddings_bridging* for bridging anaphora resolution. For each anaphor a, we construct the list of antecedent candidates E_a using NPs preceding a from the same sentence as well as from the previous two sentences. Hou et al. (2013b) found that globally salient entities are likely to be the antecedents of all anaphors in a text. We approximate this by adding NPs from the first sentence of the text to E_a. This is motivated by the fact that ISNotes is a newswire corpus and globally salient entities are often introduced in the beginning of an article. We exclude an NP from E_a if it is a bridging anaphor because a bridging anaphor is rarely to be an antecedent for another bridging anaphor. We also exclude NPs whose semantic types are "time" from E_a if a is not a time expression. This is because time expressions are related to a lot of words in the corpus in which we learned *embeddings_bridging* from. Therefore we only keep them as the antecedent candidates for bridging anaphors whose semantic types are "time" (see Example 2).

(2) As a presidential candidate in *1980*, George Bush forthrightly expressed his position on abortion in an interview with Rolling Stone magazine published **that March**.

Given an anaphor a and its antecedent candidate list E_a, we predict the most semantically related NP among all NPs in E_a as the antecedent for a. In case of a tie, the closest one is chosen to be the predicted antecedent.

The relatedness is measured via cosine similarity between the vector representation of the anaphor and the vector representation of the candidate. More specifically, given a noun phrase np_1, we first construct a list N which consists of the head and all common nouns (e.g., *earthquake victims*), adjectives (e.g., *economical sanctions*), and ed/ing participles (e.g., *the collapsed roadway* and *the landing site*) appearing before the head. If np_1 contains a post-modifier NP np_2 via the preposition "of", we also add the above premodifiers and the head of np_2 to the list N (e.g., *the policies of racial segregation*). Finally, the noun phrase np_1 is represented as a vector v using the following formula, where the suffix "_PP" is added to each n if np_1 is a bridging anaphor and its semantic type is not time:

$$v = \frac{\sum_{n \in N} embeddings\_bridging_n}{|N|} \qquad (2)$$

The underlying intuition of adding NP modifications to the list N is that the above mentioned modifiers also represent core semantics of an NP, therefore we should consider them when selecting antecedents for bridging anaphors. For instance, as shown in Table 4, for Example 1, the cosine similarity between {**arrangements_PP**} and

$\{$*products*$\}$ is 0.06, while the cosine similarity between $\{$**distribution_PP**, **arrangements_PP**$\}$ and $\{$*dialysis*, *products*$\}$ is 0.17.

If none of the words in N is present in *embeddings_bridging*, we simply neglect the noun phrase $np1$. Note that we do not add the suffix "_PP" to a bridging anaphor representing time information, because such an anaphor is likely to have the same semantic type antecedent (see Example 2). Therefore we use semantic similarity instead of relatedness to find its antecedent.

5 Experiments

5.1 Dataset

For the task of bridging anaphora resolution, we use the dataset ISNotes[5] released by Markert et al. (2012). This dataset contains around 11,000 NPs annotated for information status including 663 bridging NPs and their antecedents in 50 texts taken from the WSJ portion of the OntoNotes corpus (Weischedel et al., 2011). As stated in Section 2, bridging anaphors in ISNotes are not limited to definite NPs as in previous work (Poesio et al., 1997, 2004; Lassalle and Denis, 2011). The semantic relations between anaphor and antecedent in the corpus are quite diverse: only 14% of anaphors have a part-of/attribute-of relation with the antecedent and only 7% of anaphors stand in a set relationship to the antecedent. 79% of anaphors have "other" relation with their antecedents. This includes encyclopedic or frame relations such as *restaurant* – **the waiter** as well as context-specific relations such as *palms* – **the thieves**. In Example 1, "*dialysis products*" is the "theme" of the **distribution arrangements**. More specifically, "*dialysis products*" belongs to the frame element "Individuals" in the "Dispersal" frame that is triggered by "**distribution arrangements**".

5.2 Experimental Setup

Following Hou et al. (2013b)'s experimental setup, we resolve bridging anaphors to entity antecedents. Entity information is based on the OntoNotes coreference annotation. We also use the OntoNotes named entity annotation to assign NPs the semantic type "*time*" if their entity types are "date" or "time".

In Hou et al. (2013b), features are extracted by using entity information. For instance, the raw hit counts of the preposition pattern query (e.g., **arrangements** of *products*) for a bridging anaphor a and its antecedent candidate e is the maximum count among all instantiations of e. In our experiments, we simply extend the list of antecedent candidates E_a (described in Section 4) to include all instantiations of the original entities in E_a. Note that our simple antecedent candidate selection strategy (described in Section 4) allows us to include 76% of NP antecedents compared to 77% in *pairwise model III* from Hou et al. (2013b) where they add top 10% salient entities as additional antecedent candidates. In Hou et al. (2013b), salient entities on each text are measured through the lengths of the coreference chains based on the gold coreference annotation.

Following Hou et al. (2013b), we measure *accuracy* on the number of bridging anaphors, instead of on all links between bridging anaphors and their antecedent instantiations. We calculate how many bridging anaphors are correctly resolved among all bridging anaphors.

5.3 Using NP Head Alone

Given an anaphor a and its antecedent candidate list E_a, we predict the most related NP among all NPs in E_a as the antecedent for a[6]. The relatedness is measured via cosine similarity between the head of the anaphor (plus the postfix "_PP" if the anaphor is not a time expression) and the head of the candidate. We run experiments on the following four word embeddings: the original GloVe vectors trained on Gigaword and Wikipedia 2014 dump (*GloVe_GigaWiki14*), GloVe vectors that we trained on Gigaword only (*GloVe_Giga*), word vectors from Hou (2018) (*embeddings_PP*), and our word representation resource described in Section 3.2 (*embeddings_bridging*). Note that for the first two word vectors, we do not add the suffix "_PP" to the anaphor's head since such words do not exist in *GloVe_GigaWiki14* and *GloVe_Giga*.

Table 5 lists the results for bridging anaphora resolution based on different word representation resources[7]. We notice that there is not

---

[5] `http://www.h-its.org/en/research/nlp/isnotes-corpus`

[6] In case of a tie, the closest one is chosen to be the predicted antecedent.

[7] Note that the results for the first three word embeddings are slight better than the ones reported in Hou (2018). This is due to the improved antecedent candidate selection strategy described in Section 4.

	acc
GloVe_GigaWiki14	21.42
GloVe_Giga	21.87
embeddings_PP	**33.03**
embeddings_bridging	**34.84**

Table 5: Results of using NP head alone for bridging anaphora resolution based on different word representation resources. Bold indicates statistically significant differences over the baselines (two-sided paired approximate randomization test, $p < 0.01$).

much difference between *GloVe_GigaWiki14* and *GloVe_Giga*. We find that using *embeddings_PP* achieves an accuracy of 33.03% on the ISNotes corpus, which outperforms the results based on *GloVe_GigaWiki14* and *GloVe_Giga* by a large margin. Using *embeddings_bridging* further improves the result by 1.8%. Although the improvement is not significant, we suspect that the representations for words without the suffix "_PP" in *embeddings_bridging* are more accurate because they are trained on many more instances in the vanilla GloVe vectors (*GloVe_GigaWiki14*).

5.4 Using NP Head + Modifiers

We carried out experiments using the deterministic algorithm described in Section 4 together with different word embeddings. Again we do not add the suffix "_PP" to the bridging anaphors for *GloVe_GigaWiki14* and *GloVe_Giga*.

Table 6 lists the best results of the two models for bridging anaphora resolution from Hou et al. (2013b). *pairwise model III* is a pairwise mention-entity model based on various semantic, syntactic and lexical features. *MLN model II* is a joint inference framework based on Markov logic networks (Domingos and Lowd, 2009). It models that semantically or syntactically related anaphors are likely to share the same antecedent and achieves an accuracy of 41.32% on the ISNotes corpus.

The results for *GloVe_GigaWiki14* and *GloVe_Giga* are similar on two settings (*using NP head* vs. *using NP head + modifiers*). For *embeddings_PP*, the result on *using NP head + modifiers* (31.67%) is worse than the result on *using NP head* (33.03%). However, if we apply *embeddings_PP* to a bridging anaphor's head and modifiers, and only apply *embeddings_PP* to the head noun of an antecedent candidate, we get an accuracy of 34.53%. Although the differences are not significant, it confirms that the information from the modifiers of the antecedent candidates in *embeddings_PP* hurts the performance. This corresponds to our observations in the previous section that the representations for words without the suffix "_PP" in *embeddings_PP* are not as good as in *embeddings_bridging* due to less training instances.

	acc
models from Hou et al. (2013b)	
pairwise model III	36.35
MLN model II	**41.32**
NP head + modifiers	
GloVe_GigaWiki14	20.52
GloVe_Giga	20.81
embeddings_PP	31.67
embeddings_bridging	**39.52**

Table 6: Results of using NP head plus modifications in different word representations for bridging anaphora resolution compared to the best results of two models from Hou et al. (2013b). Bold indicates statistically significant differences over the other models (two-sided paired approximate randomization test, $p < 0.01$).

Finally, our method based on *embeddings_bridging* achieves an accuracy of 39.52%, which is competitive to the best result (41.32%) reported in Hou et al. (2013b). There is no significant difference between *NP head + modifiers* based on *embeddings_bridging* and *MLN model II* (randomization test with $p < 0.01$).

To gain an insight into the contribution of *embeddings_bridging* on different relation types, we analyze the results of our method using *embeddings_bridging* on three relation types: *set-of*, *part-of*, and *other*. The accuracies on these three relation types are 17.78%, 50.0%, and 39.16%, respectively. This suggests that in the future we should include more context for bridging anaphors that hold the *set-of* relation to their antecedents, because the head nouns of such anaphors often do not bear any specific meanings (e.g., **Another**).

5.5 Analysis of Modifiers

To better understand the role of NP modifiers in our method, we carried out experiments on *embeddings_bridging* using different set of modifiers (see Table 7). It seems that among all three types of modifiers, compared to using NP head alone, adding noun modifiers has the positive impact (36.65% on *NP head + noun modifiers* vs. 34.84% on *NP head*). Although adding only ad-

embeddings_bridging	**acc**
NP head	34.84
+ *all modifiers*	39.52
+ *noun modifiers*	**36.65**
+ *adjective modifiers*	34.84
+ *ed/ing participle modifiers*	34.84
+ *noun&adjective modifiers*	**38.31**
+ *noun&ed/ing participle modifiers*	**36.80**
+ *adjective&ed/ing participle modifiers*	34.84

Table 7: Results of using NP head plus different modifications in *embeddings_bridging*.

jective modifiers does not have influence on results, combining them with noun modifiers yields some improvement over adding only noun modifiers (38.31% on *NP head* + *noun&adjective modifiers* vs. 36.65% on *NP head* + *noun modifiers*). On the other hand, ed/ing participle modifiers only have a small positive impact over *NP head* + *noun modifiers* when combining with noun modifiers.

5.6 Combining *NP Head* + *Modifiers* with *MLN II*

For bridging anaphora resolution, Hou (2018) integrates a much simpler deterministic approach by combining an NP head with its noun modifiers (appearing before the head) based on *embeddings_PP* into the *MLN II* system (Hou et al., 2013b). Similarly, we add a constraint on top of *MLN II* using our deterministic approach (*NP head* + *modifiers*) based on *embeddings_bridging*. Table 8 lists the results of different systems[8] for bridging anaphora resolution in ISNotes. It shows that combining our deterministic approach (*NP Head* + *modifiers*) with *MLN II* slightly improves the result compared to Hou (2018).

Although combining *NP Head* + *modifiers* with *MLN II* achieves significant improvement over *NP Head* + *modifiers*, we think the latter has its own value. Our deterministic algorithm is simpler and more efficient compared to *MLN model II* + *embeddings_bridging*, which contains many complicated features and might be hard to migrate to other bridging corpora. Moreover, our algorithm is "unsupervised" and requires no training when applied to other English bridging corpora.

5.7 Resolving Bridging Anaphors in Other Corpora

Recently, two new corpora containing bridging annotation have become available. The BASHI corpus (Rösiger, 2018a) contains 459 bridging NPs and their antecedents in 50 World Street Journal articles. Similar to ISNotes, BASHI includes both definite and indefinite referential bridging anaphors. In addition, comparative anaphora is also considered as bridging anaphora in BASHI.

Another new corpus for bridging is the second release of the ARRAU corpus, which contains 5,512 bridging pairs in three different domains (Poesio et al., 2018). However, most bridging links in ARRAU are purely lexical bridging pairs, and only a small subset of the annotated pairs contains truly anaphoric bridging anaphors (Rösiger et al., 2018). Following Rösiger et al. (2018), we focus on resolving bridging anaphors in the news text domain (RST).

Based on *embeddings_bridging*, we apply our deterministic algorithm with small adaptations to resolve bridging anaphors to entity antecedents on the BASHI and ARRAU (RST) corpora. Specifically, for the BASHI corpus, we do not add NPs from the first sentence to the list of antecedent candidates E_a. This is because the phenomenon of globally salient antecedents being linked to all anaphors in a text is less obvious in BASHI. In addition, comparative anaphors often have the same semantic class as their antecedents, therefore we do not add the suffix "_PP" to a bridging anaphor if it is a comparative anaphor.

For the ARRAU corpus, we construct the list of antecedent candidates E_a using NPs preceding a from the same sentence as well as from the previous ten sentences. Since most bridging pairs in ARRAU are lexical bridging (e.g., *Tokyo* – **Japan**, *other nations* – **Britain**) and anaphors often have the same semantic type as their antecedents, we do not add the suffix "_PP" to bridging anaphors.

Table 9 lists the results of bridging anaphora resolution in the BASHI and ARRAU corpora, respectively. On the test set of the ARRAU (RST) corpus, Rösiger (2018b) proposed a modi-

---

[8] We also reimplement the algorithms from Schulte im Walde (1998) and Poesio et al. (2004) as baselines (Table 8). Schulte im Walde (1998) resolved bridging anaphors to the closest antecedent candidate in a high-dimensional space. We use the 2,000 most frequent words (adjectives, common nouns, proper nouns, and lexical verbs) from Gigaword as the context words. Poesio et al. (2004) applied a pairwise model combining lexical semantic features and salience features to perform mereological bridging resolution in the GNOME corpus. We use a Naive Bayes classifier with standard settings in WEKA (Witten and Frank, 2005) and apply the best first strategy to select the antecedent for each anaphor.

	System	acc
Baselines	*Schulte im Walde (1998)*	13.68
	Poesio et al. (2004)	18.85
Models from Hou et al. (2013b)	*pairwise model III*	36.35
	MLN model II	41.32
Hou (2018)	*MLN model II + embeddings_PP (NP head + noun pre-modifiers)*	**45.85**
This work	*embeddings_bridging (NP head + modifiers)*	39.52
	MLN model II + embeddings_bridging (NP head + modifiers)	**46.46**

Table 8: Results of different systems for bridging anaphora resolution in ISNotes. Bold indicates statistically significant differences over the other models (two-sided paired approximate randomization test, $p < 0.01$).

Corpus	Bridging Type	# of Anaphors	acc
BASHI	referential, including comparative anaphora	452	27.43
BASHI	referential, excluding comparative anaphora	344	29.94
ARRAU (RST Train)	mostly lexical, some referential	2,325	31.44
ARRAU (RST Test)	mostly lexical, some referential	639	32.39

Table 9: Results of resolving bridging anaphors in other corpora. Number of bridging anaphors is reported after filtering out a few problematic cases on each corpus.

fied rule-based system based on Hou et al. (2014)'s work and reported an accuracy of 39.8% for bridging anaphora resolution. And our algorithm achieves an accuracy of 32.39% using only *embeddings_bridging*. Overall, the reasonable performance on these two corpora demonstrates that *embeddings_bridging* is a general word representation resource for bridging.

6 Conclusions

We improve the word representation resource *embeddings_PP* (Hou, 2018) by combining it with *GloVe*. The resulting word embeddings (*embeddings_bridging*) are a more general word representation resource for bridging. Based on *embeddings_bridging*, we propose a deterministic approach for choosing antecedents for bridging anaphors. We show that this simple and efficient method achieves the competitive result on bridging anaphora resolution compared to the advanced machine learning-based approach in Hou et al. (2013b) which is heavily dependent on a lot of carefully designed complex features. We also demonstrate that using *embeddings_bridging* yields better results than using *embeddings_PP* for bridging anaphora resolution.

For the task of bridging anaphora resolution, Hou et al. (2013b) pointed out that considering only head noun knowledge is not enough and future work needs to explore wider context to resolve context-specific bridging relations. In this work we explore the context within NPs—that is, we combine the semantics of certain modifications and the head by vector average using *embeddings_bridging*. But in some cases, knowledge about NPs themselves is not enough for resolving bridging. For instance, in Example 3, knowing that **any loosening** has the ability to "rekindle inflation" from the context of the second sentence can help us to find its antecedent "*the high rates*" (which is used to against inflation).

(3) Chancellor of the Exchequer Nigel Lawson views *the high rates* as his chief weapon against inflation, which was ignited by tax cuts and loose credit policies in 1986 and 1987. Officials fear that **any loosening** this year could rekindle inflation or further weaken the pound against other major currencies.

In the future, we will study how to integrate context outside of NPs for the task of choosing antecedents for bridging anaphors. Also we hope that our word representation resource will facilitate other related research problems such as semantic role labeling.

Acknowledgments

The author appreciates the valuable feedback from the anonymous reviewers and would like to thank Massimo Poesio for sharing the ARRAU corpus.

References

I.I. Bejar, R. Chaffin, and S.E. Embretson. 1991. *Cognitive and psychometric analysis of analogical problem solving*. Springer-Verlag.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/Daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,* Berlin, Germany, 7–12 August 2016, pages 2358–2367.

Pedro Domingos and Daniel Lowd. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan Claypool Publishers.

Yufang Hou. 2016. Incremental fine-grained information status classification using attention-based LSTMs. In *Proceedings of the 26th International Conference on Computational Linguistics,* Osaka, Japan, 11–16 December 2016, pages 1880–1890.

Yufang Hou. 2018. Enhanced word representations for bridging anaphora resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* New Orleans, Louisiana, 1–6 June 2018, pages 1–7.

Yufang Hou, Katja Markert, and Michael Strube. 2013a. Cascading collective classification for bridging anaphora recognition using a rich linguistic feature set. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 814–820.

Yufang Hou, Katja Markert, and Michael Strube. 2013b. Global inference for bridging anaphora resolution. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Atlanta, Georgia, 9–14 June 2013, pages 907–917.

Yufang Hou, Katja Markert, and Michael Strube. 2014. A rule-based system for unrestricted bridging resolution: Recognizing bridging anaphora and finding links to antecedents. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 2082–2093.

David A. Jurgens, Peter D. Turney, Saif M. Mohammad, and Keith J. Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics,* Montréal, Canada, 7–8 June 1999, pages 356–364.

Emmanuel Lassalle and Pascal Denis. 2011. Leveraging different meronym discovery methods for bridging resolution in French. In *Proceedings of the 8th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2011),* Faro, Algarve, Portugal, 6–7 October 2011, pages 35–46.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* Copenhagen, Denmark, 7–11 November 2017, pages 188–197.

Katja Markert, Yufang Hou, and Michael Strube. 2012. Collective classification for fine-grained information status. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics,* Jeju Island, Korea, 8–14 July 2012, pages 795–804.

Katja Markert, Malvina Nissim, and Natalia N. Modjeska. 2003. Using the web for nominal anaphora resolution. In *Proceedings of the EACL Workshop on the Computational Treatment of Anaphora.* Budapest, Hungary, 14 April 2003, pages 39–46.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction & Web-scale Knowledge Extraction (AKBC-WEKEX)* Montréal, Québec, Canada, 7-8 June 2012, pages 95–100.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition. LDC2011T07.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 1532–1543.

Massimo Poesio, Yulia Grishina, Varada Kolhatkar, Nafise Sadat Moosavi, Ina Rösiger, Adam Roussel, Fabian Simonjetz, Alexandra Uma, Olga Uryupina, Juntao Yu, and Heike Zinsmeister. 2018. Anaphora resolution with the ARRAU corpus. In *Proceedings of the Workshop on Computational Models of Reference, Anaphora and Coreference.* New Orleans, Louisiana, June 6, 2018, pages 11–22.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, 21–26 July 2004, pages 143–150.

Massimo Poesio and Renata Vieira. 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216.

Massimo Poesio, Renata Vieira, and Simone Teufel. 1997. Resolving bridging references in unrestricted text. In *Proceedings of the ACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Text, Madrid, Spain, July 1997*, pages 1–6.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the 4th International Conference on Learning Representations,* San Juan, Puerto Rico, 2-4 May 2016.

Ina Rösiger. 2018a. BASHI: A corpus of wall street journal articles annotated with bridging links. In *Proceedings of the 11th International Conference on Language Resources and Evaluation,* Miyazaki, Japan, 7–12 May 2018, pages 382–388.

Ina Rösiger. 2018b. Rule- and learning-based methods for bridging resolution in the ARRAU corpus. In *Proceedings of the Workshop on Computational Models of Reference, Anaphora and Coreference.* New Orleans, Louisiana, June 6, 2018, pages 23–33.

Ina Rösiger, Arndt Riester, and Jonas Kuhn. 2018. Bridging resolution: Task definition, corpus resources and rule-based experiments. In *Proceedings of the 27th International Conference on Computational Linguistics,* Santa Fe, New-Mexico, USA, 20–26 August 2018, pages 3516–3528.

Sabine Schulte im Walde. 1998. Resolving bridging descriptions in high-dimensional space. Master's thesis, University of Edinburgh, Centre for Cognitive Science.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2011. OntoNotes release 4.0. LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition. Morgan Kaufmann, San Francisco, Cal.

Alisa Zhila, Scott Wen-tau Yih, Geoffrey Zweig, Chris Meek, and Tomas Mikolov. 2013. Combining heterogeneous models for measuring relational similarity. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Atlanta, Georgia, 9–14 June 2013, pages 1000–1009.

A Knowledge Hunting Framework for Common Sense Reasoning

Ali Emami[1], Noelia De La Cruz[1], Adam Trischler[2], Kaheer Suleman[2], and Jackie Chi Kit Cheung[1]

[1]School of Computer Science, Mila/McGill University
[2]Microsoft Research Montreal
{*ali.emami, noelia.delacruz*}*@mail.mcgill.ca*
{*adam.trischler, kasulema*}*@microsoft.com*
jcheung@cs.mcgill.ca

Abstract

We introduce an automatic system that achieves state-of-the-art results on the Winograd Schema Challenge (WSC), a common sense reasoning task that requires diverse, complex forms of inference and knowledge. Our method uses a knowledge hunting module to gather text from the web, which serves as evidence for candidate problem resolutions. Given an input problem, our system generates relevant queries to send to a search engine, then extracts and classifies knowledge from the returned results and weighs them to make a resolution. Our approach improves F1 performance on the full WSC by 0.21 over the previous best and represents the first system to exceed 0.5 F1. We further demonstrate that the approach is competitive on the Choice of Plausible Alternatives (COPA) task, which suggests that it is generally applicable.

1 Introduction

The importance of common-sense reasoning in natural language processing, particularly for syntactic and semantic disambiguation, has long been recognized. Almost 30 years ago, Dahlgren et al. (1989) proposed systems that use common sense to disambiguate parse trees, word senses, and quantifier scope. Although the resolution of certain ambiguities depends chiefly on linguistic patterns (e.g., the number and gender of an antecedent for pronoun disambiguation), many cases depend on world knowledge, shared points of reference, and an understanding of what is plausible—concepts often grouped under the term "common sense."

Various tasks have been devised to test common-sense reasoning in automatic systems. Two of the most popular are the *Winograd Schema Challenge* (WSC) (Levesque et al., 2011) and the *Choice of Plausible Alternatives* (COPA) (Roemmele et al., 2011). Both require a system to assess the relative plausibility of two scenarios.

WSC problems are short passages containing a target pronoun that must be correctly resolved to one of two possible antecedents. They come in pairs which differ slightly and result in adverse correct resolutions. As an example:

(1) a. Jim yelled at Kevin because *he* was so upset. (Answer: Jim)
b. Jim comforted Kevin because *he* was so upset. (Answer: Kevin)

WSC problem pairs ("twins," using the terminology of Hirst (1988)) are carefully controlled such that heuristics involving syntactic salience, the number and gender of the antecedent, or other simple syntactic and semantic cues are ineffective. This distinguishes the task from the standard coreference resolution problem. Performant systems must make common sense inferences; i.e., that someone who yells is likely to be upset, and that someone who is upset tends to be comforted. Additional examples are shown in Table 1.

WSC problems are simple for people to solve (human participants in one study performed at 92% accuracy (Bender, 2015)) but difficult for automatic systems. This is because common sense reasoning encompasses many types of reasoning (causal, spatio-temporal, etc.) and requires a wide breadth of knowledge.

COPA is a related task that tests a system's ability to recognize causality (Roemmele et al., 2011). Each instance comprises a premise and two candidate causes or effects, where the correct choice is the candidate that is more *plausible*.

Previous approaches to common sense reasoning, for instance based on logical formalisms (Bailey et al., 2015) or deep neural models (Liu et al., 2016), have solved only restricted subsets of the WSC with high precision. They have been tailored

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1949–1958
Brussels, Belgium, October 31 - November 4, 2018.

1 a)	The man couldn't lift his son because he was so weak. (Answer: the man)
1 b)	The man couldn't lift his son because he was so heavy. (Answer: son)
2 a)	The older students were bullying the younger ones, so we punished them. (Answer: the older students)
2 b)	The older students were bullying the younger ones, so we rescued them. (Answer: the younger ones)
3 a)	Sam tried to paint a picture of shepherds with sheep, but they ended up looking more like golfers. (Answer: shepherds)
3 b)	Sam tried to paint a picture of shepherds with sheep, but they ended up looking more like dogs. (Answer: sheep)

Table 1: Examples of Winograd instances.

for manually selected subsets that demand a specific type of reasoning (Sharma et al., 2015; Liu et al., 2016). Others have developed systems for relaxed common sense datasets with looser constraints (Rahman and Ng, 2012; Peng et al., 2015; Kruengkrai et al., 2014). In parallel, more general work on common sense reasoning aims to develop a repository of common knowledge using semi-automatic methods (e.g., Cyc (Lenat, 1995) and ConceptNet (Liu and Singh, 2004)). However, such knowledge bases are necessarily incomplete.

In this work, we propose a general method to resolve common sense problems like WSC and COPA. Contrary to previous work, we aim to solve *all* problem instances rather than a restricted subset. Our method is based on on-the-fly *knowledge hunting* and operates in four stages. First, it parses an input problem into a representation schema. Next it generates search queries from the populated schema. It sends these to a search engine, and the next stage parses and filters the results. Finally, it classifies and weighs the results as evidence for respective candidate resolutions.

Our approach arises from the hypothesis that there is too much common sense to encode it all statically; e.g., within a knowledge base or a neural model (using existing techniques). Even modern NLP corpora composed of billions of words are unlikely to offer good coverage of common sense, or if they do, instances of specific knowledge are likely to be "long-tailed" and difficult for statistical systems to model effectively. Information retrieval (IR) techniques can sidestep these issues by returning targeted results and by using the *entire* indexed internet as a knowledge source. Scenarios that appear in natural text can offer implicit or explicit evidence for the plausibility of related scenarios in common sense problems. To solve (1a), the following search result contains the relevant knowledge without the original ambiguity:

(2) I got really upset with her and I started to yell at her because...

Here, the same entity, *I*, is the subject of both *upset* and *yell at*, which is strong evidence for resolving the original statement. This information can be extracted from a syntactic parse of the retrieved passage with standard NLP tools.

As we will demonstrate experimentally, our knowledge hunting approach achieves an F1 score of 0.51 on the WSC, improving significantly over the previous state-of-the-art (0.3 F1). When tested on the similar COPA task, a simplified knowledge hunting system performs competitively with the previous best. To our knowledge, this is the first method that tackles multiple common sense tasks with strong performance on each. Thus, knowledge-hunting embodies some of the general capabilities that we desire of automatic systems for common sense reasoning.[1]

2 Related Work

There is increasing interest in using IR approaches to address difficult coreference problems. For example, a recent system (Rahman and Ng, 2012) uses web query information to retrieve evidence for the coreference decision in a Winograd-like corpus. Other systems (Kobdani et al., 2011; Ratinov and Roth, 2012; Bansal and Klein, 2012; Zheng et al., 2013; Peng et al., 2015; Sharma et al., 2015) rely on similar techniques, i.e., using search-query counts or co-occurrence statistics and word alignment methods to relate antecedents with pronouns.

Most recent approaches have tackled the Winograd problem by simplifying it in one of two ways. First, systems have been developed exclusively for Rahman and Ng's expanded Winograd-like corpus. These include Rahman and Ng (2012)'s system itself, achieving 73% accuracy, and Peng et al. (2015)'s system (76%). Kruengkrai et al.

---

[1] Code to reproduce these results are available at https://github.com/aemami1/Wino-Knowledge-Hunter

(2014) use sentence alignment of web query snippets to achieve 70% accuracy on a subset of the expanded corpus. Many instances in this corpus can be resolved using associations between candidate antecedents and the query predicate. For example, "Lions eat zebras because they are predators." Many of the above systems simply query "Lions are predators" versus "zebras are predators" to make a resolution decision. This exploitation is often the top contributor to such systems' overall accuracy (Rahman and Ng, 2012), but fails to apply in the majority (if not all) of the original Winograd instances.[2] Our work alleviates this issue by generating search queries that are based exclusively on the predicates of the Winograd instance, not the antecedents, and by considering the strength of the evidence.

Other systems do tackle the original, more difficult Winograd instances, but only a small, author-selected subset. The selection is based often on knowledge-type constraints. Sharma et al. (2015)'s knowledge-hunting module focused on a subset of 71 instances that exhibit *causal* relationships; Liu et al. (2016)'s neural association model focused on a similar causal subset of 70 instances, for which events were extracted manually; and finally, a recent system by Huang and Luo (2017) focused on 49 instances. While these approaches demonstrate that difficult coreference problems can be resolved when they adhere to certain knowledge or structural constraints, they may fail to generalize to other settings. This factor often goes unnoticed when systems are compared only in terms of precision; accordingly, we use an F1-driven comparison that does not enable precision boosting at the cost of recall.

Concurrently with our work, Trinh and Le (2018) introduced a system composed of 14 ensembled language models, pre-trained in an unsupervised manner, that achieves up to 63.7% accuracy on the Winograd Schema Challenge. Compared to our approach, their method requires training multiple language models with vast amounts of data, which is much more expensive.

Other Common-sense Tasks: There are various other Turing-test alternatives that directly or indirectly assess common-sense reasoning. These include Pronoun Disambiguation Problems (more generalized, Winograd-like passages without the twist of a special word or twin) (Morgenstern et al., 2016), the Narrative cloze task (Taylor, 1953), or its more difficult counterpart, the NarrativeQA Reading Comprehension Challenge (Kočiskỳ et al., 2017).

The COPA task was proposed by Roemmele et al. (2011), who also measured the performance of several systems. The most successful used Pointwise Mutual Information (PMI) statistics (Church and Hanks, 1990) between words in the premise and each alternative obtained from a large text corpus (as an implicit way to estimate causal association). More recent work showed that applying the same PMI-based technique on a corpus of stories yields better results (Gordon et al., 2011). The current state-of-the-art approaches leverage co-occurrence statistics extracted using causal cues (Luo et al., 2016; Sasaki et al., 2017).

Extended Work: Previously, Emami et al. (2018) proposed a similar knowledge hunting framework to tackle the Winograd Schema Challenge. This work modifies and extends their approach. Our modifications include a query-filtering step and various other tweaks that improve results by 0.05 F1 for our best model. In addition, we added further experiments and an ablation study that explores the performance of different model components. Finally, we adapted our method to a new dataset, COPA, on which we achieve respectable results. Accordingly, we change the general takeaway of the previous work from a method with strong performance on a single dataset to one that generalizes and performs well on various tasks.

3 Knowledge Hunting Framework

Our framework takes as input a problem instance and processes it through four stages to make a final coreference decision. First, it fits the instance to a semantic representation schema. Second, it generates a set of queries that capture the predicates in the instance's clauses and sends these to a search engine, which retrieves text snippets that closely match the schema. The returned snippets are then parsed and filtered. Finally, the snippets are resolved to their respective antecedents and the results are mapped to a best guess for the original instance's resolution. We detail these stages below, grounding our description in Winograd instances.

---

[2] This is why we do not evaluate our method directly on the expanded corpus.

3.1 Semantic Representation Schema

The first step is to perform a partial parse of each instance into a shallow semantic representation; that is, a general skeleton of each of the important semantic components in the order that they appear. This is performed using rules related to the syntactic parse of the sentence determined by Stanford CoreNLP (Manning et al., 2014).

In general, Winograd instances can be separated into a *context* clause, which introduces the two competing antecedents, and a *query* clause, which contains the target pronoun to be resolved. We use the following notation to define the components in our representation schema:

E_1, E_2	the candidate antecedents
$Pred_C$	the context predicate
$+$	discourse connective
P	the target pronoun
$Pred_Q$	the query predicate

E_1 and E_2 are noun phrases in the sentence. In the WSC, these two are specified without ambiguity. $Pred_C$ is the context predicate composed of the verb phrase that relates both antecedents to some event. The context contains E_1, E_2, and the context predicate $Pred_C$. The context and the query clauses are often connected by a discourse connective $+$. The query contains the target pronoun, P, which is also specified unambiguously. In addition, preceding or succeeding P is the query predicate, $Pred_Q$, a verb phrase involving the target pronoun. Table 2 shows sentence pairs in terms of each of these components.

3.2 Query Generation

Based on the parse, the system generates queries to send to a search engine. The goal is to retrieve text snippets that resemble the original instance. Queries are of the form:

$+Term_C$ $+Term_Q$ $-$“Winograd”$-E_1$

We assume that the search queries are composed of two components, $Term_C$ and $Term_Q$, which are strings that represent the events occurring in the first (context) and second (query) clause of the sentence, respectively. By excluding search results that may contain *Winograd* or E_1, we ensure that we do not cheat by retrieving some rewording of the original Winograd instance.

The next task is to construct two query sets, C and Q, whose elements are possible entries for $Term_C$ and $Term_Q$, respectively. We identify the root verbs in the context and query clauses, along with any modifying adjective, using the dependency parse of the sentence determined by Stanford CoreNLP (Manning et al., 2014). We add the root verbs and adjectives into the sets C and Q along with their broader verb phrases (again identified directly using the dependency tree).

Augmenting the query set with WordNet We use WordNet (Kilgarriff, 2000) to construct an augmented query set that contains synonyms for the verbs and adjectives involved in a representation. In particular, we include the synonyms listed for the top synset of the same part of speech as the extracted verb or adjective.

Query filtering Automated query generation sometimes yields terms that are irrelevant to the disambiguation task. This can add noise to the results. To address this, we implement a semantic similarity algorithm that filters root verbs and modifying adjectives from the query sets according to their relevance to other terms. We estimate relative relevance using Wu-Palmer (Wu and Palmer, 1994) similarity scores from WordNet and filter as follows. For each passage, the semantic filter (i) computes similarity scores for every possible combination of $\{Term_C, Term_Q\}$ (if both $Term_C$ and $Term_Q$ are single words); (ii) determines the maximum similarity score s; and (iii) discards any term whose highest similarity score from step (i) is less than αs, where $0 < \alpha < 1$. We tune α, a hyperparameter, on Rahman and Ng’s expanded corpus (Rahman and Ng, 2012).

We hypothesize that terms in the query and context clauses more pertinent to the task have higher mutual similarity scores than irrelevant terms. To illustrate this, consider the query sets generated for Example 2a, Table 1: {“bullying”, “younger”, “older”} and {“punished”}. Applying the semantic filter yields the new sets {“bullying”} and {“punished”}, where the irrelevant terms *younger* and *older* have been removed.

Manual query construction To understand the impact of the query generation step, we also manually produced representations for all Winograd instances. We limited the size of these sets to five to prevent a blowing-up of search space during knowledge extraction. In Table 3, we show ex-

Pair	$Pred_C$	E_1	E_2	$Pred_Q$	P	Alternating Word (POS)
1	couldn't lift	the man	his son	was so heavy	he	weak/heavy (adjective)
2	were bullying	the older students	the younger ones	punished	them	punished/rescued (verb)
3	tried to paint	shepherds	sheep	ended up looking more like	they	golfers/dogs (noun)

Table 2: Winograd sentence pairs from Table 1, parsed into the representation schema that we define.

Sentence: The trophy doesn't fit into the brown suitcase because it is too large.

Query Generation Method	C	Q
Automatic	{"doesn't fit into", "brown", "fit" }	{"large", "is too large"}
Automatic, with synonyms	{"doesn't fit into", "brown", "accommodate", "fit", "suit" }	{"large", "big", "is too large" }
Manual	{"doesn't fit into", "fit into","doesn't fit" }	{"is too large", "too large", "large" }

Table 3: Query generation techniques on an example Winograd sentences

amples of generated queries for C and Q using the various techniques.

3.3 Parsing the Search Results

From the search results, we obtain a set of text snippets that we filter for similarity to the original problem instance. First, $Term_C$ and $Term_Q$ are restricted to occur in the same snippet, but are allowed to occur in any order. We filter the passed sentences further to ensure that they contain at least two entities that corefer. These may be structured as follows:

$$\begin{array}{lcl} E_1'\ Pred_C'\ E_2' & + & E_3'\ Pred_Q' \\ E_1'\ Pred_C'\ E_2' & + & Pred_Q'\ E_3' \\ E_1'\ Pred_C' & + & E_3'\ Pred_Q' \\ E_1'\ Pred_C' & + & Pred_Q'\ E_3' \end{array}$$

We call these *evidence sentences*. They exhibit a structure similar to the corresponding Winograd instance, but with different entities and event order. $Pred_C'$ and $Pred_Q'$ (resulting from the queries $Term_C$ and $Term_Q$, resp.) should be similar if not identical to $Pred_C$ and $Pred_Q$ from the Winograd sentence. However, E_1', E_2', and E_3' may not have the same semantic type, potentially simplifying their coreference resolution. A sentence for which E_3' refers to E_1' is subsequently labelled *evidence-agent*, and one for which E_3' refers to E_2', *evidence-patient*. The exception to this rule is when an event occurs in the passive voice (e.g., *was called*), which reverses the conventional order of the agent and patient. Another exception is in the case of *causative alternation*, where a verb can be used both transitively and intransitively. The latter case can also reverse the conventional order of the agent and patient (e.g., *he opened the door* versus *the door opened*).

As an example of coreference simplification, a valid evidence sentence is: *He tried to call* ***her*** *but* ***she*** *wasn't available.* Here, the sentence can be resolved on the basis of the gender of the antecedents; E_3' (the pronoun *she*) refers to the patient, E_2'. Accordingly, the sentence is considered an evidence-patient.

3.4 Antecedent Selection

We collect and reason about the set of retrieved sentences using a selection process that (i) resolves E_3' to either E_1' or E_2' using CoreNLP's coreference resolution module (rendering them evidence-agent or evidence-patient); and (ii) uses both the count and individual features of the evidence sentences to resolve the original Winograd instance. For example, the more similar evidence-**agents** there are for the sentence *Paul tried to call George on the phone, but he wasn't successful*, the more likely it is that the process would guess *Paul*, the **agent**, to be the correct referent of the target pronoun.

To map each sentence to either an evidence-agent or evidence-patient, we developed a rule-based algorithm that uses the syntactic parse of an input sentence. This algorithm outputs an evidence label along with a list of features. The features indicate: which two entities co-refer according to Stanford CoreNLP's resolver, and to which category of E_1', E_2', or E_3' each belong; the token length of the sentence's search terms, $Term_C$ and $Term_Q$; the order of the sentence's search terms; whether the sentence is in active or passive voice; and whether or not the verb is causative alternating. Some of these features are straightforward to extract (like token length and order, and coreferring entities given by CoreNLP), while others require various heuristics. To map each coreferring entity in the snippet to E_1', E_2', or E_3' (corresponding loosely to context subject, context object, and query entity, respectively), we consider their position relative to the predicates in the original Winograd instance. That is, E_1' precedes $Term_C$, E_2' succeeds $Term_C$, and E_3' may precede or succeed $Term_Q$ depending on the Winograd instance. To determine the voice, we use a list of auxiliary verbs and verb phrases (e.g., *was*, *had been*, *is*, *are being*) that switch the voice from active to passive (e.g., "they are being bullied" vs "they bullied") whenever one of these precedes $Term_C$ or $Term_Q$ (if they are verbs). Similarly, to identify causative alternation, we use a list of causative alternating verbs (e.g., *break*, *open*, *shut*) to identify the phenomenon whenever $Term_C$ or $Term_Q$ is used intransitively.

These features determine the evidence label, evidence-agent (EA) or evidence-patient (EP), according to the following rules:

$$L(e) = \begin{cases} \text{EA}, & \text{if } E_3' \text{ refers to } E_1', \text{ active (1)} \\ \text{EA}, & \text{if } E_3' \text{ refers to } E_2', \text{ passive (2)} \\ \text{EP}, & \text{if } E_3' \text{ refers to } E_2', \text{ active (3)} \\ \text{EP}, & \text{if } E_3' \text{ refers to } E_1', \text{ passive (4)} \\ \text{EP}, & \text{if } E_3' \text{ refers to } E_1', \text{ causative (5)} \end{cases}$$

Cases (2), (4), and (5) account for the passive and causative constructions, which alter the mapping from syntactic role to semantic role.

In addition to determining the evidence label, the features are used in a heuristic that generates scores (called *strengths*) for each evidence sentence:

$$\mathit{Str}(e) = \mathit{LenScore}(e) + \mathit{OrderScore}(e)$$

$$\mathit{LenScore}(e) = \begin{cases} 2, & \text{if } len(Term_Q) > 1 \\ 2, & \text{if } len(Term_C) > 1 \\ 1, & \text{otherwise} \end{cases}$$

$$\mathit{OrderScore}(e) = \begin{cases} 2, & \text{if } Term_C \prec Term_Q \\ 1, & \text{if } Term_Q \prec Term_C \end{cases}$$

As an example of scoring for an actual snippet, let us consider "She **tried to call** for him and then search for him herself, but **wasn't successful**," returned for $Term_C$=*tried to call*, and $Term_Q$=*wasn't successful*.

Here, both $Term_Q$ and $Term_C$ are multi-word search terms, and $Term_C$ precedes $Term_Q$ as in the original Winograd sentence. Its overall evidence strength is 4, the highest possible score. On the other hand, for the retrieved snippet "Has your husband **tried** Sudafed and was it **successful**?" for $Term_Q$=*tried*, and $Term_C$=*successful*, the evidence strength would be 3. We designed the scoring system to capture the structural similarity of a snippet to its corresponding Winograd instance. We observed that a greater quantity of snippets can be retrieved for less specific search terms, but with increasing noise; we sought to account for this with the features described above. Note also that our use of the word *features* is intentional. While the weights assigned for the length and order scores could be optimized, as parameters, we consider it inappropriate to do so on the WSC since it is widely used as a *test set*. We set these weights according to our best guess and validated our choices through experiments on the set of Winograd-like sentences provided in Rahman and Ng (2012).

We run the above four processes on all snippets retrieved for the input Winograd instance. The sum of strengths for the evidence-agents is finally compared to that of the evidence-patients to make the resolution decision.

4 Experiments and Results

We tested several versions of our framework on the original 273 Winograd sentences (135 pairs and one triple). These vary in the method of query generation: automatic *vs.* automatic with synonyms *vs.* manual. We compared these systems with previous work on the basis of Precision (P), Recall (R), and F1.

We used Stanford CoreNLP's coreference resolver (Raghunathan et al., 2010) during query generation to identify the predicates from the syntactic parse, as well as during antecedent selection to retrieve the coreference chain of a candidate evidence sentence. Python's Selenium package was used for web-scraping and Bing-USA and Google (top two pages per result) were the search engines. The search results comprise a list of document snippets that contain the queries (for example, "yelled at" and "upset"). We extract the sentence/s within each snippet that contain the query terms, with the added restriction that the terms should be within 70 characters of each other to encourage relevance.

	# Correct	P	R	F1
AGQ	77	0.56	0.28	0.38
AGQ+F	80	0.63	0.29	0.40
AGQS	114	0.57	0.42	0.48
AGQS+F	119	0.60	**0.44**	**0.51**
S2015	49	**0.92**	0.18	0.30
Systems with manual information:				
L2017	43	0.61	0.15	0.25
MGQ	118	0.60	0.43	0.50

Table 4: Coverage and performance on the original Winograd Schema Challenge (273 sentences).

Table 4 shows the precision, recall, and F1 of our framework's variants: automatically generated queries (AGQ), automatically generated queries with synonyms (AGQS), and manually generated queries (MGQ). We test the automatic systems with (+F) and without the semantic similarity filter. We compare these to the systems of Sharma et al. (2015) (S2015) and Liu et al. (2017) (L2017). The system developed by Liu et al. (2017) uses elements extracted manually from the problem instances, so is most closely comparable to our MGQ method. Our best automated framework, AGQS+F, outperforms S2015 by 0.21 F1, achieving much higher recall (0.44 vs 0.18). Our results show that the framework with manually generated queries (MGQ) performs better than its automatic counterpart, AGQ, with an F1 of 0.50. AGQS+F slightly outperforms MGQ despite being fully automatic.

The power of our approach lies in its generality, i.e., its improved coverage of the problem set. It produces an answer for over 70% of instances. This surpasses previous methods, which only admit specific instance types, by nearly 50%.

The random baseline on this binary task achieves a P/R/F1 of 0.5. We can artificially raise the F1 performance of all systems above 0.5 by randomly guessing an answer in cases where the system makes no decision. For AGQS+F, for example, if we take a random decision on the cases (74) with no retrieved evidence, we get an accuracy of 57.1%. However, we think it is important that systems are compared transparently based on which instances they admit and when they are capable of making a prediction.

5 Error Analysis

To get a sense of the performance of our heuristics in classifying evidence sentences in the antecedent selection step, we manually labelled sentences retrieved by the AGQS system for 40 Winograd instances. The categories are evidence-agent, evidence-patient, or neither (insufficient evidence). This amounts to a total of 876 evidence sentences. We compared these labels to those assigned by our system. In total, 703 of the 876 evidence sentences were labelled correctly (81%). Of the 173 incorrect cases, 110 were marked as insufficient evidence. Our system is forced to label these as agent or patient.

Evidence sentences were insufficient for a variety of reasons. Most frequently, they were structurally incomplete or grammatically incorrect, despite passing as valid through CoreNLP and our initial coreference heuristics. In general, our coreference heuristics filter strongly: over all Winograd instances, they filter a total of 50,110 retrieved sentences down to only 3,097 (0.0617 acceptance rate). As for the 63 cases of sufficient evidence sentences that were labelled incorrectly, the issue was either errors in the coreference information from the CoreNLP pipeline or errors in our heuristics for reasoning about the coreference information. We show examples of these various sources of error in supplementary Table S1. At any rate, the corrected labels (with the 110 insufficient evidence removed and the 63 cases corrected) did not result in a shift in any of the 40 coreference decisions.

In Table 5, we show a sample resolution that our system makes on a problem instance,[3] including some evidence that was retrieved and labelled automatically and the evidence strengths that led to the resolution. These examples reveal that, indeed, general knowledge of what is plausible appears in natural text. Our system successfully leverages this knowledge for common sense reasoning.

---

[3] We provide more examples in a supplementary file.

WSC Instance:	The man couldn't lift his son because he was so weak. Answer: the man (**Agent**)
Evidence and labels: (query terms in bold)	"However I **was so weak** that I **couldn't lift**" → **EA** "She **was so weak** she couldn't **lift**" → **EA** "I could not stand without falling immediately and I **was so weak** that I **couldn't lift**" → **EA** "It hurts to **lift** my leg and its kind of **weak**" → **EP**
Stats and resolution:	Agent evidence strength: 97 Patient evidence strength: 72 Number of scraped sentences: 109 Resolution: **Agent**

Table 5: Example Resolution for a WSC problem.

We also include an example evidence snippet that yields a "misleading" label. Generally, sources of misleading snippets include incomplete or imprecise query generation (e.g. in Table 5, querying only "lift" instead of "couldn't lift"), errors in the automatic parsing of sentences (e.g., in supplementary Table S1.1.b, "lift" is incorrectly labelled as a verb via the parse tree, despite being a noun), and insufficient filtering of noisy sentences that are not relevant to the problem instance or are incomplete (e.g. in supplementary Table S1.2.b, the sentence is incomplete and indicates a misleading resolution).

6 Generalization to COPA

To investigate the generality of our knowledge-hunting approach, we adapted it to the Choice of Plausible Alternatives (COPA). We evaluated our basic automatic models that did not use the semantic similarity filter for this check.

COPA has a slightly different form that necessitates some modifications. As an example,

(3) The climbers reached the peak of the mountain. What happened as a result?
   a. They encountered an avalanche.
   b. **They congratulated each other.**

During query generation, as before, the set C contains terms extracted from the context sentence. Instead of a single set Q as in the WSC, we generate two query sets Q_1 and Q_2, that contain terms extracted for the first and second candidate sentences. Because entities in the candidate sentences can contribute to the answer (unlike in the WSC), we modified the query generation rules to extract more than just predicates. Specifically, the extraction procedure uses the syntactic parse tree of the phrase to back-off from extracting the clause containing the subject and verb phrase, to only the verb phrase, to only the verbs or adjectives that are rooted in the verb phrase. For the running example, our system generates these three sets: C={"The climbers reached the peak", "reached the peak", "reached"}, Q_1={"They encountered an avalanche", "encountered an avalanche", "encountered"}, and Q_2={"They congratulated each other", "congratulated each other", "congratulated"}.

We query the web for sentences that contain terms in (C, Q_1) and (C, Q_2), with one added restriction: for problem instances in which the relation is *cause*, the system only extracts sentences in which $Term_C$ precedes $Term_{Q_1}$ or $Term_{Q_2}$; when the relation is *result* (as in our running example), $Term_C$ succeeds $Term_{Q_1}$ or $Term_{Q_2}$. As for the WSC, the final decision is determined from the evidence snippets according to their strengths.

	Dev	Test
Goodwin et al. (2012)	–	63.4
AGQS	64.0	65.1
Gordon et al. (2011)	62.8	65.4
AGQ	65.8	66.2[4]
Luo et al. (2016)	–	70.2
Sasaki et al. (2017)	–	**71.2**

Table 6: Model accuracy (%) on COPA.

We tuned the system's evidence-scoring heuristics on COPA's 500 validation instances. In Table 6, we compare our system's performance on the 500 test instances to previous work on the basis of precision (which in the full-coverage case equates to accuracy). Our simpler AGQ method achieves 66.2% accuracy, which is respectable, although not state-of-the-art. As indicated by the lower performance of AGQS, synonyms from WordNet did not improve performance on COPA. Without the semantic-similarity filtering, synonyms may add noise to the retrieved results. It has also been shown that multi-word expressions are prevalent and important for COPA (Sasaki et al., 2017), which we have not

---

[4] This precision can be inflated to 67.2 by randomly guessing on the 10 examples for which there were no search results.

specifically attempted to handle with our method. We believe that this is a promising direction of improvement for our approach in future work.

7 Conclusion

We developed a knowledge-hunting framework to tackle the Winograd Schema Challenge, a task that requires common-sense knowledge and reasoning. Our system involves a semantic representation schema and an antecedent selection process that acts on web-search results. We evaluated the performance of our framework on the original set of WSC instances, achieving F1-performance that significantly exceeded the previous state-of-the-art. A simple port of our approach to COPA suggests that it has the potential to generalize. In the future we will study how this commonsense reasoning technique can contribute to solving "edge cases" and difficult examples in more general coreference tasks.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada.

References

Dan Bailey, Amelia Harrison, Yuliya Lierler, Vladimir Lifschitz, and Julian Michael. 2015. The winograd schema challenge and reasoning about correlation. In *In Working Notes of the Symposium on Logical Formalizations of Commonsense Reasoning*.

Mohit Bansal and Dan Klein. 2012. Coreference semantics from web features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 389–398. Association for Computational Linguistics.

David Bender. 2015. Establishing a human baseline for the winograd schema challenge. In *MAICS*, pages 39–45.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Kathleen Dahlgren, Joyce McDowell, and Edward P Stabler. 1989. Knowledge representation for commonsense reasoning with text. *Computational linguistics*, 15(3):149–170.

Ali Emami, Adam Trischler, Kaheer Suleman, and Jackie Chi Kit Cheung. 2018. A generalized knowledge hunting framework for the winograd schema challenge. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 25–31.

Travis Goodwin, Bryan Rink, Kirk Roberts, and Sanda M Harabagiu. 2012. Utdhlt: Copacetic system for choosing plausible alternatives. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 461–466. Association for Computational Linguistics.

Andrew S Gordon, Cosmin Adrian Bejan, and Kenji Sagae. 2011. Commonsense causal reasoning using millions of personal stories. In *AAAI*.

Graeme Hirst. 1988. Semantic interpretation and ambiguity. *Artificial intelligence*, 34(2):131–177.

Wenguan Huang and Xudong Luo. 2017. Commonsense reasoning in a deeper way: By discovering relations between predicates. In *ICAART (2)*, pages 407–414.

Adam Kilgarriff. 2000. Wordnet: An electronic lexical database.

Hamidreza Kobdani, Hinrich Schütze, Michael Schiehlen, and Hans Kamp. 2011. Bootstrapping coreference resolution using word associations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 783–792. Association for Computational Linguistics.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *arXiv preprint arXiv:1712.07040*.

Canasai Kruengkrai, Naoya Inoue, Jun Sugiura, and Kentaro Inui. 2014. An example-based approach to difficult pronoun resolution. In *PACLIC*, pages 358–367.

Douglas B Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. 2016. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*.

Quan Liu, Hui Jiang, Zhen-Hua Ling, Xiaodan Zhul, Si Wei, and Yu Hu. 2017. Combing context and commonsense knowledge through neural networks for solving winograd schema problems.

Zhiyi Luo, Yuchen Sha, Kenny Q Zhu, Seung-won Hwang, and Zhongyuan Wang. 2016. Commonsense causal reasoning between short texts. In *KR*, pages 421–431.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Leora Morgenstern, Ernest Davis, and Charles L Ortiz Jr. 2016. Planning, executing, and evaluating the winograd schema challenge. *AI Magazine*, 37(1):50–54.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving hard coreference problems. *Urbana*, 51:61801.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501. Association for Computational Linguistics.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: the winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2012. Learning-based multi-sieve co-reference resolution with knowledge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1234–1244. Association for Computational Linguistics.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, pages 90–95.

Shota Sasaki, Sho Takase, Naoya Inoue, Naoaki Okazaki, and Kentaro Inui. 2017. Handling multiword expressions in causality estimation. In *IWCS 2017—12th International Conference on Computational Semantics—Short papers*.

Arpit Sharma, Nguyen Ha Vo, Somak Aditya, and Chitta Baral. 2015. Towards addressing the winograd schema challenge-building and using a semantic parser and a knowledge hunting module. In *IJCAI*, pages 1319–1325.

Wilson L Taylor. 1953. “cloze procedure”: a new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Z Wu and M Palmer. 1994. verb semantics and lexical selection in proceedings of the 32nd annual meeting of the association for computational linguistics. *New Mexico*.

Jiaping Zheng, Luke Vilnis, Sameer Singh, Jinho D Choi, and Andrew McCallum. 2013. Dynamic knowledge-base alignment for coreference resolution. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 153–162.

Mapping Text to Knowledge Graph Entities using Multi-Sense LSTMs

Dimitri Kartsaklis[*] **Mohammad Taher Pilehvar** **Nigel Collier**
Department of Theoretical and Applied Linguistics
University of Cambridge
Cambridge, UK
`{dk426;mp792;nhc30}@cam.ac.uk`

Abstract

This paper addresses the problem of mapping natural language text to knowledge base entities. The mapping process is approached as a composition of a phrase or a sentence into a point in a multi-dimensional entity space obtained from a knowledge graph. The compositional model is an LSTM equipped with a dynamic disambiguation mechanism on the input word embeddings (a *Multi-Sense LSTM*), addressing polysemy issues. Further, the knowledge base space is prepared by collecting random walks from a graph enhanced with textual features, which act as a set of semantic bridges between text and knowledge base entities. The ideas of this work are demonstrated on large-scale text-to-entity mapping and entity classification tasks, with state of the art results.

1 Introduction

The task of associating a well-defined action, concept or piece of knowledge to a natural language utterance or text is a common problem in natural language processing and generic artificial intelligence (Tellex et al., 2011), and can emerge in many different forms. In NLP, the ability to code text into an entity of a knowledge graph finds applications in tasks such as question answering and information retrieval, or any task that involves some form of mapping a definition to a term (Hill et al., 2016; Rimell et al., 2016). Further, it can be invaluable in providing solutions to domain-specific challenges, for example medical concept normalisation (Limsopatham and Collier, 2016) and identification of adverse drug reactions (O'Connor et al., 2014).

This paper details a model for efficiently mapping unrestricted text at the level of phrases and sentences to the entities of a knowledge base (KB)—a task also referred to as text *grounding* or *normalisation*. The model aims at characterising short focused texts, such as definitions or tweets. Given a medical KB, for example, a tweet of the form "Can't sleep, too tired to think straight" would be mapped to the entity *Insomnia*, while in the context of a lexical ontology the definition "Device that detects planets" would be associated to the entity *Telescope*.

Note that such a task cannot be approached as standard classification, since the "classes" (entities) are usually in one-to-one correspondence with the available inputs. To address this we propose the use of a continuous vector space for embedding the entities of the KB graph, where text is projected by a neural network. We rely on the notion of distributional semantics, where a word is represented as a multi-dimensional vector obtained either by collecting co-occurrence statistics with a selected set of contexts or by directly optimising an objective function in a neural network-based architecture (Collobert and Weston, 2008). Interestingly, similar techniques can be used for the multi-dimensional representation of nodes in a KB graph; for example, by collecting random walks following the edges of a graph it is possible for one to construct an artificial "corpus", to which a distributional model applies in the usual way (Perozzi et al., 2014).

By exploiting this representational compatibility, we treat the process of text-to-entity mapping as a transformation from a textual vector space where words live, to a KB vector space created from a graph and populated by vectors representing entities. A sentence is coded as a sequence of word vectors, composed by a modified Long Short-Term Memory network (LSTM—Hochreiter and Schmidhuber, 1997) into a multi-dimensional point in the entity space. One of our aims is to specifically deal with lexical ambiguity and polysemy which can be an important factor

---

[*] This paper is dedicated to the memory of Euripides Kartsaklis, a man who loved technology.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1959–1970
Brussels, Belgium, October 31 - November 4, 2018.

for the task at hand. To this end, each word is associated with a number of sense embeddings, and the LSTM is extended with an attentional disambiguation mechanism that dynamically selects and updates the right sense vector for each word given its context during training. We dub this formulation *Multi-Sense LSTM* (MS-LSTM).

An important issue is the provision of a set of reliable anchors; that is, points in one-to-one correspondence between the two representations that would enforce some degree of structural similarity between pieces of text and KB entities and thus make the mapping more efficient. We deal with this problem by extending the original KB graph with nodes corresponding to *textual features*, i.e. to words strongly associated to a specific entity and collected from various resources. A novel sampling strategy is detailed for incorporating these nodes to random walks, which are then fed to the skipgram model for producing an entity space. The results indicate that the textual nodes, being words *and* KB entities at the same time, do an extremely effective job in transforming the geometry of the entity space to the benefit of mapping the textual modality.

The proposed model is evaluated in three tasks: text-to-entity mapping on a dataset extracted from SNOMED CT[1], a medical knowledge base of 327,000 concepts; a reverse dictionary task based on WordNet (Miller, 1998), where the goal is to associate a multi-word definition to the correct lemma (Hill et al., 2016); and document classification on the Cora dataset (McCallum et al., 2000). The results demonstrate the effectiveness of our methods by improving the current state of the art.

2 Background

Aligning meaning between text and entities in a knowledge graph is a task traditionally based on heuristic methods exploiting text features such as string matching, word weighting, syntactic relations, or dictionary lookups (McCallum et al., 2005; Lu et al., 2011; O'Connor et al., 2014). Machine learning techniques have been also exploited in various forms, for example Leaman et al. (2013) use a pairwise learning-to-rank technique to learn the similarity between different terms, while Limsopatham and Collier (2015) apply statistical machine translation to "translate" social media text to domain-specific terminology. There is little work based on neural networks; the most relevant to us is a study by Hill et al. (2016), who tested a number of compositional neural architectures trained to approximate word embeddings on a reverse dictionary task. Compared to their work, this paper proposes the use of a distinct target space for representing ontological knowledge, where every entity in the graph lives.

The goal of a graph embedding method is to embed components of a knowledge graph into a low-dimensional space. One research direction focuses on the relations, i.e. the edges of the graph (Bordes et al., 2013; Socher et al., 2013; Xiao et al., 2016) and aims at tasks such as link prediction and KB completion. Such work is outside the scope of the current paper, the subject of which is the efficient low-dimensional representation of *entities* (nodes). In this line of research, a prevalent method involves the collection of a set of random walks, starting from each node in the graph (Perozzi et al., 2014; Grover and Leskovec, 2016). There is a direct analogy between such a set of random walks and a text corpus: each node corresponds to a word and the sequence of nodes visited during a random walk is analogous to a sentence. Thus, any distributional model that takes as input this artificial "corpus" can generate multi-dimensional representations of the nodes in the graph. Random walks have also been used for KB inference (Lao et al., 2011) with success.

While random walk-based methods are not the only way to construct graph spaces—alternatives include factorisation (Ahmed et al., 2013) and deep autoencoders (Wang et al., 2016)—they have been found very effective in capturing multiple aspects of the graph structure (Wang et al., 2017; Goyal and Ferrara, 2017). The current paper proposes a random walk generation strategy that improves and complements existing approaches.

The idea of using textual features to improve the entity vectors is not well explored, and most of the existing work focuses again on the representation of relations (Xie et al., 2016; Wang et al., 2014; Wang and Li, 2016) as opposed to entities. Closer to us is the work of Yamada et al. (2017) and Yang et al. (2015), with the latter to incorporate text features in the concept embeddings by exploiting matrix factorisation properties.

Representing the meaning of words using a number of sense vectors is an old and well-established idea in NLP—see for example (Schütze, 1998; Reisinger and Mooney, 2010; Neelakantan et al., 2014). However, most of the relevant research is evaluated on intrinsic tasks

---

[1] https://www.snomed.org/snomed-ct

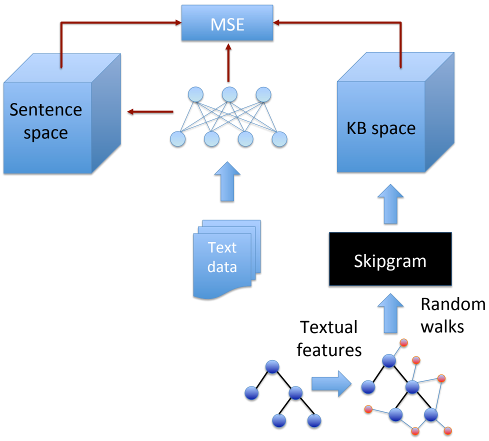

Figure 1: The text-to-entity mapping system in a nutshell. The red nodes indicate textual features, while “MSE” stands for mean squared error.

such as word similarity, while in the few works based in real end tasks, disambiguation is usually treated as a prior stand-alone step (Kartsaklis and Sadrzadeh, 2013; Li and Jurafsky, 2015; Pilehvar et al., 2017). The crucial difference of this work is that the ambiguity resolution mechanism is part of the compositional model itself, and the sense embeddings are trained simultaneously with the rest of the parameters. A close work is by Cheng and Kartsaklis (2015), who used a siamese network with an integrated disambiguation mechanism for paraphrase detection. For more information on multi-sense embeddings see (Camacho-Collados and Pilehvar, 2018).

3 Methodology

Fig. 1 provides a high-level illustration of our methodology, consisting of two stages: (1) the KB graph is extended with weighted textual features, and an artificial “corpus” of random walks is created and used as input to the skipgram model (Mikolov et al., 2013) for generating an enhanced KB space—this part is covered in §3.1; (2) the transformation from text to entities is performed by a supervised multi-sense compositional model, which generates a point in the KB space for every input text. This is achieved with an LSTM recurrent network, equipped with an attentional mechanism that provides a finer level of granularity to the different ways a word is used in the data—we detail this part in §3.2.

3.1 Textual features for entity vectors

For our KB space, we follow the generic recipe proposed by Perozzi et al. (2014) and we assemble an artificial corpus of random walks from the KB graph, which is then used as input to the skipgram model (Mikolov et al., 2013). For a random walk of nodes $n_1 n_2 \ldots n_T$ and a context window size c, skipgram maximises the following quantity:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(n_{t+j}|n_t) \qquad (1)$$

i.e. for a target node n_t, the objective is to predict all other nodes in the same context. As a consequence, two vectors of the resulting space will be close if their corresponding nodes occur in topological proximity within the graph. However, while such a topology allows perhaps for meaningful comparisons between points in this space, it is not directly compatible with the task of mapping text to entities. The reason is that the communities formed in a KB graph (and thus the topology of the resulting vector space) mostly reflect domain-specific hierarchies and ontological relationships that are not necessarily evident by the textual representations referring to the entities. An important question therefore with regard to the proposed methodology is how to provide meaningful links between the two representations that would allow for the efficient translation of one form (text) to another (entities).

In this work, we address this problem by associating every node in the graph with a set of textual features, each one of which is weighted according to their importance with respect to the node. Our methodology is as follows: For each entity, we collect all available textual descriptions found in the knowledge base itself and the English portion of BabelNet (Navigli and Ponzetto, 2012), which is a very large dictionary integrating numerous resources, such as WordNet, Wikipedia, FrameNet and many others. The textual descriptions are treated as short documents, and each word in them is assigned a specific TF-IDF value, forming the set of textual features for the specific entity.

The KB graph is extended in the following way: Let T_c be the set of textual features for an entity c; then, for each t in T_c, we add an edge (c, t) with weight tf-idf$_c(t)$, where tf-idf$_c(t)$ is the TF-IDF value of t with respect to c. In contrast to Perozzi et al. (2014) who utilise a uniform node sampling strategy, we define the random walk generation process as follows: Given a randomly selected node n, let $C_n = \{c_1, c_2, \cdots c_N\}$ be the set of all entity nodes in its immediate vicinity, and $T_n = \{t_1, t_2, \cdots t_M\}$ the set of all textual features

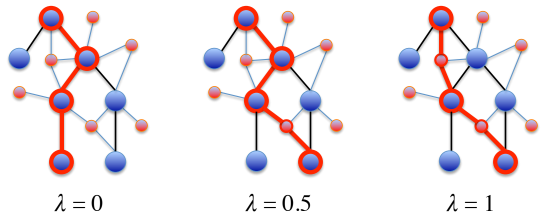

$\lambda = 0$ $\lambda = 0.5$ $\lambda = 1$

Figure 2: Effect of λ parameter. Blue nodes indicate entities, red nodes textual features, and red paths refer to random walks. As λ increases, the probability of "hops" between originally unlinked nodes increases accordingly.

of n; the next node x in the path is drawn from a categorical distribution defined as below:

$$P_X(x) = \begin{cases} (1-\lambda)\dfrac{1}{N} & \text{if } x \in C_n \\ \lambda\dfrac{\text{tf-idf}_n(t_i)}{\sum_{j=1}^{M}\text{tf-idf}_n(t_j)} & \text{if } x \in T_n \end{cases} \tag{2}$$

for X a discrete random variable with range $C_n \cup T_n$. In the above, λ defines the proportion of the probability mass allocated to textual features, when both C_n and T_n are non-empty; if one of the sets is empty, all of the probability mass is allocated to the other set, and λ becomes irrelevant. Further, in contrast to what is the case for the textual nodes, the probabilities of the entity nodes in Equation 2 are defined uniformly, since we lack any mechanism for fine-tuning them in a way that objectively reflects the importance of the nodes.

It is instructive to examine how the above sampling strategy works. As expected, setting $\lambda = 0$ will result in a sampling process that ignores the textual features and produces a path comprised solely of entity nodes; this is equivalent to the original model by Perozzi et al. (2014), known as DeepWalk. On the other hand, the effect of setting $\lambda = 1$ is less intuitive: Recall that, by construction, each textual node is connected only to entity nodes; that is, when the current node is textual, the next node will be always sampled from C_n. Therefore, setting $\lambda = 1$ creates paths following an alternating pattern, where each entity node is followed by a textual node, which in turn is followed by an entity node. Values of λ between 0 and 1 scale this behaviour accordingly (Fig. 2).

Advantages. The introduction of textual features in the graph achieves two goals. Firstly, the textual nodes serve as links between entities which, although perhaps related to each other in some way, lie in different parts of the KB graph (e.g. being parts of different hierarchies). As a result, points that would normally be unjustifiably apart of each other in the vector space are now brought closer, providing additional coherence. This behaviour is controlled by the λ parameter, as Fig. 2 shows. Fig. 3 presents an illustrative example, taken from a real random walk on SNOMED CT.

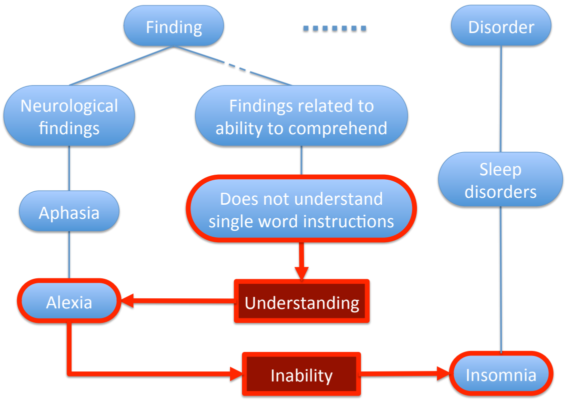

Figure 3: Linking of distant concepts with textual features (red boxes) for $\lambda = 1$. The textual feature *understanding* correctly links a related medical finding (lying on a different branch) to the condition known as *alexia*. Further, due to the presence of *inability* in their contexts, the vectors of *alexia* and *insomnia* (concepts originally quite far apart in the graph) will now have a common part reflecting that they are both conditions related to forms of incompetence.

The second advantage of introducing textual features in the graph is a consequence of the dual nature of these features in the context of learning: they essentially represent words, but since they are also nodes of the graph, they get vector representations exactly as every other normal entity in the knowledge base. The textual features, therefore, paired with their assigned vectors, form a set of *anchors* that links pieces of text with the KB space, and can be used to support the training process of the mapping system. In §4.1 we will see that this approach leads to substantial improvements in the accuracy of the model.

3.2 A multi-sense LSTM

We now proceed to present our neural architecture for text-to-entity mapping. The goal of the model is, given a certain piece of text, to produce a point in the KB space corresponding to an appropriate entity or concept. The model is trained on pairs of texts and entity vectors created from a graph extended with textual features, as discussed in §3.1.

Our architecture needs to explicitly take into account the fact that the task at hand is very sen-

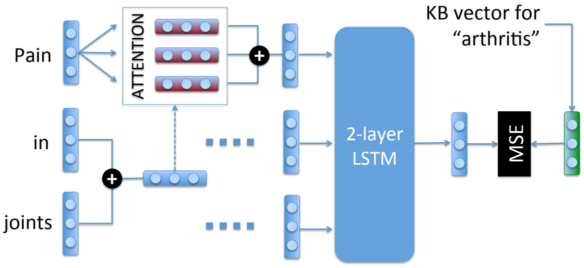

Figure 4: Detailed architecture of the Multi-Sense LSTM (MS-LSTM), shown here for the first word of a phrase. Red vectors refer to senses, while the green vector is a target vector.

sitive to lexical ambiguity. Specifically, while it is true that the level of homonymy (words having more than one disjoint meanings) is substantially decreased when we move from the generic domain to more specialised domains, on the other hand the increase in polysemy (words with many slightly different meanings) is exponential. As an example, while the lemma for the word "fever" in a dictionary usually contains two or three definitions, the term occurs in many dozens of different forms and contexts in SNOMED. Note that most of the different uses of the term correspond to distinct KB nodes, a fact that makes the job of a text-to-entity mapping system especially hard.[2] This motivates the employment of a dedicated mechanism that would handle the extra complexity imposed by the polysemous words.

The compositional setting of this paper, equipped with such a mechanism, is shown in Fig. 4. It consists of a generic word embedding layer, a word sense disambiguation layer, and two consecutive LSTM networks responsible for encoding the embeddings into a vector in the KB space. The objective is to minimise the mean squared error between the predicted vectors and the target vectors (prepared as in §3.1):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} \|y_i - f(x_i)\|^2 \qquad (3)$$

where N is the number of training examples, x the input text, y the target entity vector, and f the neural network.

To address the polysemy issues discussed above, every word is associated with a single generic embedding and k sense embeddings, where k is a fixed number. These sense embeddings can be seen as centroids of clusters denoting different uses of the word in the training set, and are dynamically updated during training. Specifically, for each word w_i in a training example, a context vector c_i is computed as the average of the *generic* vectors of all other words in the sentence. The probability of each sense vector s_{ij} given this context is then calculated via an attentional mechanism equipped with a softmax layer, as follows:

$$p(s_{ij}|c_i) = \frac{\exp\,(w_j'^{\intercal} s_{ij}')}{\sum_{l=1}^{k} \exp\,(w_l'^{\intercal} s_{il}')} \qquad (4)$$

where $s_{ij}' = \tanh(W s_{ij} + U c_i)$, and W, U and W' the parameters of the attentional network. Each sense vector is subsequently updated by addition of the context vector weighted by its similarity with the specific sense:

$$s_{ij}^{t+1} = s_{ij}^{t} + (s_{ij}^{t}{}^{\intercal} c_i^t) c_i^t \qquad (5)$$

The output of the attention is a weighted sum of the sense vectors given their probabilities (i.e. we apply *soft attention*), which is used as input to the compositional network—a 2-layer LSTM. The overall model is optimised on the MSE of the LSTM's output vector and the target entity vector. At inference time, a predicted vector $\hat{y}$ can be classified to the entity with the closest vectorial representation according to some metric.

4 Experiments

The ideas presented in the previous sections are evaluated on three tasks, two of which are related to text-to-entity mapping, and one to classification of KB entities. The purpose of the classification task (§4.3) is to provide a direct comparison of the textually enhanced vectors against vectors produced by the original graph, but *independently* of the compositional part. On the other hand, the text mapping experiments (§§4.1, 4.2) evaluate the overall architecture of Fig. 1 (including the compositional model and the dynamic disambiguation mechanism) on appropriate end tasks. Comparisons are provided with the most relevant previous work. Specifically, in all tasks, no inclusion of textual features corresponds to the standard DeepWalk model of Perozzi et al. (2014); in §4.2 our compositional architecture is compared to the work of Hill et al. (2016) in their reverse dictionary task; and §4.3 compares our method for textually enhancing the entity space with that of Yang et al. (2015), and other state-of-the-art deep models. The last subsection, §4.4, examines a few selected cases from a qualitative perspective.

---

[2] See also §4.4 for some concrete examples.

4.1 Text-to-entity mapping

We begin with a large scale text-to-entity mapping experiment. We construct a dataset of 21,000 medical concepts extracted from SNOMED CT, each of which is associated with a multi-word textual description, taken from the knowledge base or BabelNet. The criterion for including a concept in the dataset was the availability of at least one textual description with 4 or more words. The objective of the task is to associate each one of these descriptions to the correct concept. Given a predicted vector $\hat{v}$, we assemble a list of all candidate concept vectors ranked by their cosine similarity with $\hat{v}$. We compute strict accuracy (based on how many times the vector of the correct concept is at the top of the list) and accuracy on the first 20 elements of the list. Further, we also present results based on the mean reciprocal rank (MRR).

In all experiments, we create KB vectors of 150 dimensions by applying the skipgram objective on a set of random walks of length 20, and with window size of 5. The graph is extended with 102,500 textual nodes weighted by their TF-IDF values with regard to the corresponding entities and selected as described in §3.1 (textual features that occur in the testing set are not taken into account). Each node in the graph serves as the starting point of 10 random walks. For the compositional model, we use embeddings of 150 dimensions, and 200-dimensional hidden states. The attentional mechanism is implemented as a 2-layer MLP, with 50 units allocated to the hidden layer for each sense. The overall model contains two dropout layers for regularisation purposes, and is optimised with Adam (Kingma and Ba, 2015) ($\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$).[3]

Following usual practice, we split our dataset in three parts: a training set (14,754 instances), a testing set (4,187 instances), and a development set (2,000 instances). We use the dev set to optimise the two main hyper-parameters of our model, namely the probability mass given to textual features (λ) and the number of senses for each word (k). The experiments on the dev set showed that increasing the probability mass for the inclusion of textual features in the random walks leads to consistently better performance for all tested models, so for the main experiment we set λ to its highest possible value, 1.00.[4] Further, a number of senses equal to 3 achieved the highest performance.

Model	Target space	MRR	Acc	Acc-20
Baseline 1	W2V-GoogleNews	0.25	0.19	0.41
Baseline 2	W2V-PubMed	0.17	0.12	0.31
Least squares	DeepWalk	0.19	0.10	0.49
	TF vectors	**0.49**	**0.37**	**0.79**
CCA	DeepWalk	0.36	0.24	0.70
	TF vectors	**0.71**	**0.60**	**0.94**
Standard LSTM (150 dim.)	DeepWalk	0.30	0.20	0.58
	TF vectors	**0.82**	**0.73**	**0.97**
Standard LSTM ($k \times 150$ dim.)	DeepWalk	0.33	0.23	0.59
	TF vectors	**0.86**	**0.80**	**0.97**
MS-LSTM	DeepWalk	0.36	0.26	0.60
	TF vectors	**0.89**	**0.84**	**0.98**
MS-LSTM + anchors	TF vectors	**0.94**	**0.90**	**1.00**

Table 1: Results for the SNOMED dataset. For the MS-LSTM we set $k = 3$, while *TF vectors* refers to our textually enhanced vectors ($\lambda = 1$). The difference between MS-LSTM and LSTM is s.s. with $p < 0.01$ according to a two-tailed z-test.

We compare our MS-LSTM with a number of baselines: In *Baselines 1* and *2* a vector for each textual description is computed as the average of pre-computed word vectors, and compared to concept vectors prepared in a similar way, i.e. by averaging pre-computed vectors for all words in the qualified name of the entities. We used two different word spaces, a standard Word2Vec space created from Google News[5] and a custom Word2Vec model trained on a corpus of 4B tokens from medical articles indexed in PubMed[6]. In *Least squares* and *CCA*, an averaged vector for each textual description is again computed as before, and a linear mapping is learned between the textual space and the KB space, using least squares and canonical correlation analysis (Hardoon et al., 2004).

In *Standard LSTM*, we use a configuration similar to that of Fig. 4, but *without* the multi-sense aspect; here, the word embeddings are just parameters of the model randomly initialised before training. Further, we also test a standard LSTM where the length of the single embeddings is k times bigger (k is the number of senses in the MS-LSTM), so that the overall dimensionality of embeddings in LSTM and MS-LSTM is the same.

The results are presented in Table 1. Each model is tested against two target KB spaces, one consisting of simple DeepWalk vectors[7] and one of textually enhanced vectors (TF vectors, $\lambda = 1$) according to the procedure of §3.1. There are three observations: (1) Using the enhanced vectors as a

---

[3] Python code will be released at https://github.com/cambridgeltl/SIPHS.

[4] Recall that this means half of the nodes in a random walk will be textual (see Fig. 2).

[5] https://code.google.com/archive/p/word2vec

[6] http://www.ncbi.nlm.nih.gov/pubmed/

[7] In our setting, this is equivalent to having $\lambda = 0$.

target space improves the performance of *all* tested models by a large margin; (2) the MS-LSTM configuration of Fig. 4 achieves the highest overall performance, showing that explicitly handling polysemy during the composition is beneficial for the task at hand; and (3) despite the equal dimensionality between the two models, the standard LSTM with the long embeddings presents performance inferior to that of the MS-LSTM.

The last row of the table presents results after extending the training dataset with the textual anchors, that is, all the textual features paired with their learned KB vectors, as described in the *Advantages* section in §3.1. Specifically, recall that each textual feature (a word or a two-word compound), being also a node in the graph, is associated with a vector according to the process of §3.1. It is possible for one then to use these (*textual feature*, *vector*) pairs as additional examples during the training of the MS-LSTM. The last row of Table 1 shows the results after extending the training set with the 102,500 textual features. This setting achieves the highest performance, increasing further the strict accuracy by 6%, to 0.90.

4.2 Reverse dictionary

We proceed to the reverse dictionary task of Hill et al. (2016), the goal of which is to return a candidate term given a definition. Many forms of this task have been proposed in the past, see for example (Kartsaklis et al., 2012; Turney, 2014; Rimell et al., 2016). In (Hill et al., 2016), the authors test a number of supervised models under two evaluation modes: (1) "seen", in which the testing instances are also included in the training set; and (2) "unseen", where the evaluation is done on a held-out set. In both cases the datasets consisted of 500 term-definition pairs from WordNet.

We treat WordNet as a graph, the edges of which are defined by the various relationships between the synsets. This graph is further extended with 96,734 textual nodes extracted from the synset descriptions. We compute synset vectors of 150 dimensions, on random walks of length 20 and with window size of 5. For the seen evaluation, we train the compositional model on the totality of WordNet 3.0 synsets (117,659) and their descriptions. For the unseen evaluation, we remove from the graph any textual features occurring in the testing part, and create a new set of synset vectors; further, any testing instance is removed from the training set of the compositional model. The evaluation is done by comparing the predicted vector with the vectors of all WordNet synsets (a search space of 117,659 points) and creating a ranked list as before, by cosine similarity. Following (Hill et al., 2016), we compute accuracy on top-10 and top-100. λ and k are tuned on a dev set of 2,000 synsets, showing a behaviour very similar to that of the SNOMED task.

Model	Acc-10	Acc-100
Seen (500 WordNet definitions)		
OneLook (Hill et al., 2016)	0.89	0.91
RNN cosine (Hill et al., 2016)	0.48	0.73
Std LSTM (150 dim.) + TF vec.	0.86	0.96
Std LSTM ($k \times 150$ dim.) + TF vec.	0.93	0.98
MS-LSTM +TF vectors	**0.95**	**0.99**
MS-LSTM +TF vectors + anchors	**0.96**	**0.99**
Unseen (500 WordNet definitions)		
RNN w2v cosine (Hill et al., 2016)	0.44	0.69
BOW w2v cosine (Hill et al., 2016)	0.46	0.71
Std LSTM (150 dim.) + TF vec.	0.72	0.88
Std LSTM ($k \times 150$ dim.) + TF vec.	0.77	0.90
MS-LSTM + TF vectors	**0.79**	**0.90**
MS-LSTM + TF vectors + anchors	**0.80**	**0.91**

Table 2: Results for the reverse dictionary task, compared with the highest numbers reported by Hill et al. (2016). *TF vectors* refers to textually enhanced vectors with $\lambda = 1$. For the MS-LSTM, k is set to 3.

Table 2 shows the results, based on a MS-LSTM setup similar to that of §4.1. Note that the MS-LSTM achieves 0.95-0.96 top-10 accuracy for the seen evaluation, significantly higher not only than the best model of Hill et al. (2016), but also higher than OneLook, a commercial system with access to more than 1000 dictionaries. It also presents considerably higher performance in the unseen evaluation. We are not aware of any other models with higher performance on the specific task.

4.3 Document classification

Our last experiment is a document classification task, performed on Cora (McCallum et al., 2000), a dataset containing 2708 machine learning papers linked by citation relationships into a graph. Each document is a short text extracted from the title or the abstract of the paper. The task is to predict the category of a document (a total of 7 classes), given its vector—so here we only evaluate the textually enhanced vectors as inputs to a classifier, independently of the compositional part.

In Table 3 we report results for two evaluation settings. In Evaluation 1, we provide a comparison with the method of Yang et al. (2015) who include textual features in graph embeddings based on matrix factorisation, and two topic models used as baselines in their paper. Using the same clas-

Model	Accuracy
Evaluation 1 (training ratio=0.50)	
PLSA (Hofmann, 1999)	0.68
NetPLSA (Mei et al., 2008)	0.85
TADW (Yang et al., 2015)	0.87
Linear SVM + DeepWalk vectors	0.85
Linear SVM + TF vectors	**0.88**
Evaluation 2 (training ratio=0.05)	
Planetoid (Yang et al., 2016)	0.76
GCN (Kipf and Welling, 2017)	0.81
GAT (Veličković et al., 2018)	**0.83**
Linear SVM + DeepWalk vectors	0.72
Linear SVM + TF vectors	**0.82**

Table 3: Results for the Cora dataset. *TF vectors* refers to textually enhanced KB vectors ($\lambda = 0.5$). Difference between our best models and GAT/GCN/TADW are *not* s.s.

sification algorithm (a linear SVM) and training ratio (0.50) with them, we present state-of-the-art results for vectors of 150 dimensions, prepared by a graph extended with 1422 textual features. We set $\lambda = 0.5$ by tuning on a dev set of 677 randomly selected entries from the training data.[8]

In Evaluation 2, using the same linear SVM classifier and λ as before, we reduce the training ratio to 0.05 in order to make our task comparable to the experiments reported by Veličković et al. (2018) for a number of deep learning models: specifically, the graph attention network (GAT) of Veličković et al. (2018), the graph convolutional network (GCN) of Kipf and Welling (2017), and the Planetoid model of Yang et al. (2016). Again, our simple setting presents results within the state of the art range, comparable to (or better than) those of much more sophisticated models that have been specifically designed for the task of node classification. We consider this as a strong indication for the effectiveness of the textually enhanced vectors as representations of KB entities.

Fig. 5 provides a visualisation of the Cora classes based on node vectors created with $\lambda = 0$ and $\lambda = 0.5$, correspondingly, demonstrating the impact of textual features in terms of cluster coherence and separation.

4.4 Qualitative evaluation

Table 4 compares the performance of the multi-sense approach with that of the single-sense model for a number of selected cases of text mapping. The predictions in the top part (for definitions taken from the unseen evaluation of the reverse dictionary task) show that, in contrast to the single-sense model, the multi-sense approach was able to capture subtle variations of meaning between different synsets due to polysemy, as motivated in §3.2. The lower part of the table contains short phrases with ambiguous words, specifically selected to demonstrate the effect of the multi-sense approach. In all these cases, the multi-sense model was able to effectively disambiguate the ambiguous parts of the phrase by using the available context, and predict a very relevant synset; in contrast, the predictions of the single-sense model were based on choosing a wrong sense.

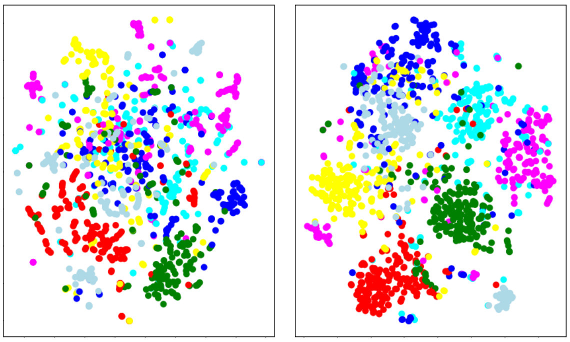

Figure 5: Visualisation of the Cora classes based on a 2D t-SNE projection of the node vectors before the inclusion of textual features (left) and after (right).

Finally, Table 5 presents the derived senses for word *table*, expressed as lists of nearest neighbouring words in the space. The model was able to effectively distinguish between a table as a kitchen furniture (sense 2), and a table as a structured way of presenting data (senses 1 and 3).

5 Discussion

The experimental work shows that using a graph embedding space as a target for mapping text to entities is an effective approach. This was mostly evident in the reverse dictionary task of §4.2, where the model was found to perform substantially better than previous approaches by Hill et al. (2016), who used a compositional architecture similar to ours but optimised on the *word* embeddings of the target terms. Note that this is suboptimal in the sense that, unless specific measures are taken, a word embedding reflects ambiguous meaning; therefore, trying to associate a definition like "keyboard musical instrument with pipes" to the vector for word "organ" introduces a certain amount of noise in the model, since the definition will be partly associated with features related to

---

[8] We also attempted a second classification experiment on a dataset of 200k concepts extracted from SNOMED, observing a similar behaviour of λ (details are not reported due to space). This difference in the behaviour of λ between text-to-entity mapping and classification tasks is discussed in §5.

Definition from the unseen dataset of the reverse dictionary task	$k = 3$ (correct pred.)	$k = 1$ (wrong pred.)
the branch of engineering that deals with things smaller than 100 nm	nanotechnology	microelectronics
floor consisting of open space at the top of a house just below roof	loft	balcony
a board game for two players; pieces move according to dice throws	backgammon	checkers
an address of a religious nature	sermon	rogation

Example short phrase with ambiguous words	$k = 3$ prediction	$k = 1$ prediction
a rechargeable cell	nickel-cadmium battery	karyolysis (*biological process*)
a state capital	Curitiba (*Brazilian state capital*)	assert (*verb*)
the lap of a person	upper side of thighs	lapper (*garment*)
a band named Queen	band leader	neckband (*garment*)

Table 4: Qualitative comparison of a few selected cases for multi- and single-sense LSTMs.

Sense 1.	formulation, uncommonly, rauwolfia, cardiology, hypodermic, malleability, points, optic, dendrite, rubiaceae, nonparametric, meninges, deviation, anesthetics
Sense 2.	tableware, meal, expectation, heartily, kitchen, hum, eating, forestay, suitors, croupier, companionship, restaurant, dishes, candles, cup, tea
Sense 3.	reassigned, projective, ultracentrifuge, polemoniaceous, thyronine, assumptions, lymphocyte, atomic, difficulties, intracellular, virgil, elementary, cartesian

Table 5: Derived senses for word *table*, visualised as lists of nearest neighbouring words in the vector space.

the "body part" sense of the word. In our model, homonymy issues are resolved by design: each point in the target space corresponds to a well-defined unambiguous concept or synset. Further, the attentional mechanism of Fig. 4 handles subtle variations of each distinct sense due to polysemy.

The effectiveness of the textual feature mechanism was demonstrated in every task we attempted, but to different extents. As our tuning on the dev sets showed, for tasks closer to text-to-entity mapping (§§4.1-4.2) the more the textual features in the random walks, the better the results were. However, the best performance on the classification task came by λ values between 0.50 and 0.75, i.e. by walks visiting more entity nodes than textual nodes. The reason is that entity classification is a task very sensitive to the topology of the KB graph, since entities belonging to a specific class are very likely to be located at the same sub-hierarchy, hence in topological proximity. On the other hand, one of the motivations for introducing textual features was exactly to broaden the context of a node by connecting distant parts of the graph (see Figures 2-3). So, while small amounts of textual features can be still useful for classification purposes, excessive use introduces unwanted noise in the model.

The dynamic disambiguation mechanism integrated in the compositional architecture improved further the performance of the model. This finding is consistent with previous work on simpler tensor-based models, which showed that applying some form of word sense disambiguation when composing word vectors can provide consistent improvements on end tasks such as sentence similarity and paraphrase detection (Kartsaklis and Sadrzadeh, 2013; Kartsaklis et al., 2013).

6 Conclusion and future work

We presented and evaluated a text-to-entity mapping system based on a continuous KB space enhanced with textual features and capable of handling polysemy. The reasonable next step will be to extend our methods for modelling the *relations* (edges) of a KB graph, which will allow applications in tasks such as link prediction and KB completion. Furthermore, having a mechanism that translates arbitrary text to points in a continuous space creates many opportunities for interesting research. For example, while the size of a knowledge base is finite, the space itself consists of infinite number of points, each one of which corresponds to a valid—yet not explicitly stated in the KB—entity of the same domain. The exciting question of how can we exploit this extra information—for instance in order to *enrich* the knowledge base with new data—constitutes one of our future directions.

Acknowledgments

The authors would like to thank Victor Prokhorov and Ehsan Shareghi for useful discussions on the paper, as well as the anonymous reviewers for their suggestions. The idea of a dynamic disambiguation mechanism embedded in a neural architecture originally appeared in (Cheng and Kartsaklis, 2015) as part of a generic multi-prototype embedding model. Kartsaklis and Collier acknowledge support by EPSRC grant EP/M005089/1; Pilehvar and Collier are supported by MRC grant No. MR/M025160/1. We are also grateful to NVIDIA Corporation for the donation of a Titan XP GPU.

References

Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 37–48, New York, NY, USA. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Jose Camacho-Collados and Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. To appear in *Journal of Artificial Intelligence Research* (JAIR).

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1531–1542, Lisbon, Portugal. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Palash Goyal and Emilio Ferrara. 2017. Graph embedding techniques, applications, and performance: A survey. *arXiv preprint arXiv:1705.02801*.

Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA. ACM.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation*, 9(8):1735–1780.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1590–1601, Seattle, Washington, USA. Association for Computational Linguistics.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *Proceedings of 24th International Conference on Computational Linguistics (COLING 2012): Posters*, pages 549–558, Mumbai, India. The COLING 2012 Organizing Committee.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating disambiguation from composition in distributional semantics. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 114–123, Sofia, Bulgaria. Association for Computational Linguistics.

Diederik P Kingma and Lei Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference of Learning Representations (ICLR)*.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, Edinburgh, Scotland, UK. Association for Computational Linguistics.

R. Leaman, Rezarta Islamaj Dogan, and Z. Lu. 2013. Dnorm: Disease name normalization with pairwise learning to rank. *Bioinformatics*, 29:2909–2917.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732. Association for Computational Linguistics.

Nut Limsopatham and Nigel Collier. 2015. Adapting phrase-based machine translation to normalise medical terms in social media messages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1675–1680, Lisbon, Portugal. Association for Computational Linguistics.

Nut Limsopatham and Nigel Collier. 2016. Normalising medical concepts in social media texts by learning semantic representation. In *Proceedings of the Association of Computational Linguistics Annual Meeting*, pages 1014–1023, Berlin, Germany.

Zhiyong Lu, Hung-Yu Kao, Chih-Hsuan Wei, Minlie Huang, Jingchen Liu, Cheng-Ju Kuo, Chun-Nan Hsu, Richard Tzong-Han Tsai, Hong-Jie Dai, Naoaki Okazaki, Han-Cheol Cho, Martin Gerner, Illes Solt, Shashank Agarwal, Feifan Liu, Dina Vishnyakova, Patrick Ruch, Martin Romacker, Fabio Rinaldi, Sanmitra Bhattacharya, Padmini Srinivasan, Hongfang Liu, Manabu Torii, Sergio Matos, David Campos, Karin Verspoor, Kevin M. Livingston, and W. John Wilbur. 2011. The gene normalization task in BioCreative III. *BMC Bioinformatics*, 12(8):S2.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 388–395, Arlington, Virginia, United States. AUAI Press.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. 2008. Topic modeling with network regularization. In *Proceedings of the 17th international conference on World Wide Web*, pages 101–110. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the Workshop at the International Conference on Learning Representations (ICLR)*.

George Miller. 1998. WordNet: An Electronic Lexical Database. *Communications of the ACM*, 38(11):39–41.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient nonparametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha,Quatar. Association for Computational Linguistics.

Karen O'Connor, Pranoti Pimpalkhute, Azadeh Nikfarjam, Rachel Ginn, Karen L Smith, and Graciela Gonzalez. 2014. Pharmacovigilance on Twitter? mining tweets for adverse drug reactions. In *AMIA annual symposium proceedings*, volume 2014, page 924. American Medical Informatics Association.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Mohammad Taher Pilehvar, Jose Camacho-Collados, Roberto Navigli, and Nigel Collier. 2017. Towards a seamless integration of word senses into downstream nlp applications. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1857–1869, Vancouver, Canada. Association for Computational Linguistics.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

L. Rimell, J. Maillard, T. Polajnar, and S. Clark. 2016. RELPRON: a relative clause evaluation dataset for compositional distributional semantics. *Computational Linguistics*.

H. Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24:97–123.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*.

Peter D Turney. 2014. Semantic composition and decomposition: From recognition to generation. *arXiv preprint arXiv:1405.7908*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference of Learning Representations (ICLR)*.

Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1225–1234, New York, NY, USA. ACM.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1591–1601.

Zhigang Wang and Juanzi Li. 2016. Text-enhanced representation learning for knowledge graph. In *Proceedings of the Twenty-Fifth International Joint*

Conference on Artificial Intelligence, IJCAI'16, pages 1293–1299. AAAI Press.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. Transg : A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2316–2325, Berlin, Germany. Association for Computational Linguistics.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2659–2665. AAAI Press.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning distributed representations of texts and entities from knowledge base. *Transactions of the Association for Computational Linguistics*, 5:397–411.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *IJCAI*, pages 2111–2117.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International Conference of Machine Learning (ICLR)*, pages 40–48.

Differentiating Concepts and Instances for Knowledge Graph Embedding

Xin Lv, Lei Hou[*], Juanzi Li, Zhiyuan Liu
Department of Computer Science and Technology,
Tsinghua University, China 100084
`{lv-xl8@mails.,houlei@,lijuanzi@,liuzy@}tsinghua.edu.cn`

Abstract

Concepts, which represent a group of different instances sharing common properties, are essential information in knowledge representation. Most conventional knowledge embedding methods encode both entities (concepts and instances) and relations as vectors in a low dimensional semantic space equally, ignoring the difference between concepts and instances. In this paper, we propose a novel knowledge graph embedding model named TransC by differentiating concepts and instances. Specifically, TransC encodes each concept in knowledge graph as a sphere and each instance as a vector in the same semantic space. We use the relative positions to model the relations between concepts and instances (i.e., `instanceOf`), and the relations between concepts and sub-concepts (i.e., `subClassOf`). We evaluate our model on both link prediction and triple classification tasks on the dataset based on YAGO. Experimental results show that TransC outperforms state-of-the-art methods, and captures the semantic transitivity for `instanceOf` and `subClassOf` relation. Our codes and datasets can be obtained from `https://github.com/davidlvxin/TransC`.

1 Introduction

Knowledge graphs (KGs) aim at semantically representing the world's truth in the form of machine-readable graphs composed of triple facts. Knowledge graph embedding encodes each element (entities and relations) in knowledge graph into a continuous low-dimensional vector space. The learned representations make the knowledge graph essentially computable and have been proved to be helpful for knowledge graph completion and information extraction (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015b; Ji et al., 2015, 2016).

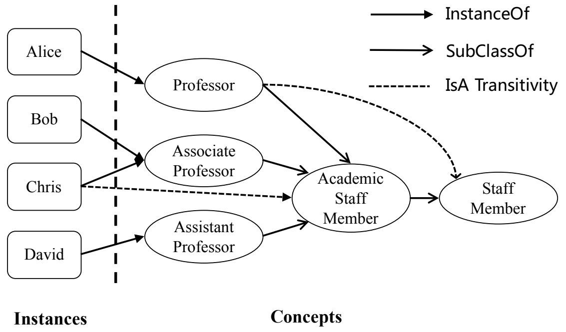

Figure 1: An example of concepts, instances, and isA transitivity.

In recent years, various knowledge graph embedding methods have been proposed, among which the translation-based models are simple and effective with good performances. Inspired by word2vec (Mikolov et al., 2013), given a triple (h, r, t), TransE learns vector embeddings $\mathbf{h}$, $\mathbf{r}$ and $\mathbf{t}$ which satisfy $\mathbf{r} \approx \mathbf{t} - \mathbf{h}$. Afterwards, TransH (Wang et al., 2014), TransR/CTransR (Lin et al., 2015b) and TransD (Ji et al., 2015), etc are proposed to address the problem of TransE when modeling 1-to-N, N-to-1, and N-to-N relations. As extensions of RESCAL(Nickel et al., 2011), which is a bilinear model, HolE(Nickel et al., 2016), DistMult(Yang et al., 2014) and ComplEx(Trouillon et al., 2016) achieve the state-of-the-art performances. Meanwhile, there are also some different methods using a variety of external information such as entity types (Xie et al., 2016), textual descriptions (Wang and Li, 2016), as well as logical rules to strengthen representations of knowledge graphs (Wang et al., 2015; Guo et al., 2016; Rocktäschel et al., 2015).

However, all these methods ignore to distinguish between concepts and instances, and regard both as entities to make a simplification. Actually, concepts and instances are organized differently in many real world datasets like YAGO (Suchanek

---

[*] corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1971–1979
Brussels, Belgium, October 31 - November 4, 2018.

et al., 2007), Freebase (Bollacker et al., 2008), and WordNet (Miller, 1995). Hierarchical concepts in these knowledge bases provide a natural way to categorize and locate instances. Therefore, the common simplification in previous work will lead to the following two drawbacks:

Insufficient concept representation: Concepts are essential information in knowledge graph. A concept is a fundamental category of existence (Rosch, 1973) and can be reified by all of its actual or potential instances. Figure 1 presents an example of concepts and instances about university staffs. Most knowledge embedding methods encode both concepts and instances as vectors, cannot explicitly represent the difference between concepts and instances.

Lack transitivity of both isA relations: `instanceOf` and `subClassOf` (generally known as isA) are two special relations in knowledge graph. Different from most other relations, isA relations exhibit transitivity, e.g., the dotted lines in Figure 1 represent the facts inferred by isA transitivity. The indiscriminate vector representation for all relations in previous work cannot reserve this property well (see Section 5.3 for details).

To address these issues, we propose a novel translation embedding model named TransC in this paper. Inspired by (Tenenbaum et al., 2011), concepts in people's mind are organized hierarchically and instances should be close to concepts that they belong to. Hence in TransC, each concept is encoded as a sphere and each instance as a vector in the same semantic space, and relative positions are employed to model the relations between concepts and instances. More specifically, `instanceOf` relation is naturally represented by checking whether an instance vector is inside a concept sphere. For the `subClassOf` relation, we enumerate and quantify four possible relative positions between two concept spheres. We also define loss functions to measure the relative positions and optimize knowledge graph embeddings. Finally, we incorporate them into translation-based models to jointly learn the knowledge representations of concepts, instances and relations.

Experiments on real world datasets extracted from YAGO show that TransC outperforms previous work like TransE, TransD, HolE, DistMult and ComplEx in most cases. The contributions of this paper can be summarized as follows:

1. To the best of our knowledge, we are the first to propose and formalize the problem of knowledge graph embedding which differentiates between concepts and instances.

2. We propose a novel knowledge embedding method named TransC, which distinguishes between concepts and instances and deals with the transitivity of isA relations.

3. We construct a new dataset based on YAGO for evaluation. Experiments on link prediction and triple classification demonstrate that TransC successfully addresses the above problems and outperforms state-of-the-art methods.

2 Related Work

There are a variety of models for knowledge graph embedding. We divide them into three kinds and introduce them respectively.

2.1 Translation-based Models

TransE (Bordes et al., 2013) regards a relation $\mathbf{r}$ as a translation from $\mathbf{h}$ to $\mathbf{t}$ for a triple (h, r, t) in training set. The vector embeddings of this triple should satisfy $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. Hence, $\mathbf{t}$ should be the nearest neighbor of $\mathbf{h} + \mathbf{r}$, and the loss function is

$$f_r(h, t) = ||\mathbf{h} + \mathbf{r} - \mathbf{t}||_2^2. \qquad (1)$$

TransE is suitable for 1-to-1 relations, but it has problems when handling 1-to-N, N-to-1, and N-to-N relations.

TransH (Wang et al., 2014) attempts to alleviate the problems of TransE above. It regards a relation vector $\mathbf{r}$ as a translation on a hyperplane with $\mathbf{w}_r$ as the normal vector. The vector embeddings will be first projected to the hyperplane of relation r and get $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r$ and $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r$. The loss function of TransH is

$$f_r(h, t) = ||\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp||_2^2. \qquad (2)$$

TransR/CTransR (Lin et al., 2015b) addresses the issue in TransE and TransH that some entities are similar in the entity space but comparably different in other specific aspects. It sets a transfer matrix $\mathbf{M}_r$ for each relation r to map entity embedding to relation vector space. Its loss function is

$$f_r(h, t) = ||\mathbf{M}_r \mathbf{h} + \mathbf{r} - \mathbf{M}_r \mathbf{t}||_2^2. \qquad (3)$$

TransD (Ji et al., 2015) considers the different types of entities and relations at the same time. Each relation-entity pair (r, e) will have a mapping matrix $\mathbf{M}_{re}$ to map entity embedding into relation vector space. And the projected vectors could be defined as $\mathbf{h}_\perp = \mathbf{M}_{rh}\mathbf{h}$ and $\mathbf{t}_\perp = \mathbf{M}_{rt}\mathbf{t}$. The loss function of TransD is

$$f_r(h, t) = ||\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp||_2^2. \tag{4}$$

There are many other translation-based models in recent years. For example, TranSparse (Ji et al., 2016) simplifies TransR by enforcing the sparseness on the projection matrix, PTransE (Lin et al., 2015a) considers relation paths as translations between entities for representation learning, (Xiao et al., 2016a) proposes a manifold-based embedding principle (ManifoldE) for precise link prediction, TransF (Feng et al., 2016) regards relation as translation between head entity vector and tail entity vector with flexible magnitude, (Xiao et al., 2016b) proposes a new generative model TransG, and KG2E (He et al., 2015) uses Gaussian embedding to model the data uncertainty. All these models can be seen in (Wang et al.).

2.2 Bilinear Models

RESCAL(Nickel et al., 2011) is the first bilinear model. It associates each entity with a vector to capture its latent semantics. Each relation is represented as a matrix which models pairwise interactions between latent factors.

Many extensions of RESCAL have been proposed by restricting bilinear functions in recent years. For example, DistMult (Yang et al., 2014) simplifies RESCAL by restricting the matrices representing relations to diagonal matrices. HolE(Nickel et al., 2016) combines the expressive power of RESCAL with the efficiency and simplicity of DistMult. It represents both entities and relations as vectors in $\mathbb{R}^d$. ComplEx(Trouillon et al., 2016) extends DistMult by introducing complex-valued embeddings so as to better model asymmetric relations.

2.3 External Information Learning Models

External information like textual information is significant for knowledge representation. TEKE (Wang and Li, 2016) uses external context information in a text corpus to represent both entities and words into a joint vector space with alignment models. DKRL (Xie et al., 2016) directly learns entity representations from entity descriptions. (Wang et al., 2015; Guo et al., 2016; Rocktäschel et al., 2015) use logical rules to strengthen representations of knowledge graphs.

All models above do not differentiate between concepts and instances. To the best of our knowledge, our proposed TransC is the first attempt which represents concepts, instances, and relations differently in the same space.

3 Problem Formulation

In this section, we formulate the problem of knowledge graph embedding with concepts and instances. Before that, we first introduce the input knowledge graph.

Knowledge Graph $\mathcal{KG}$ describes concepts, instances, and the relations between them. It can be formalized as $\mathcal{KG} = \{\mathcal{C}, \mathcal{I}, \mathcal{R}, \mathcal{S}\}$. $\mathcal{C}$ and $\mathcal{I}$ denote the sets of concepts and instances respectively. Relation set $\mathcal{R}$ can be formalized as $\mathcal{R} = \{r_e, r_c\} \cup \mathcal{R}_l$, where r_e is an `instanceOf` relation, r_c is a `subClassOf` relation, and $\mathcal{R}_l$ is the instance relation set. Therefore, the triple set $\mathcal{S}$ can be divided into three disjoint subsets:

1. `InstanceOf` triple set $\mathcal{S}_e = \{(i, r_e, c)_k\}_{k=1}^{n_e}$, where $i \in \mathcal{I}$ is an instance, $c \in \mathcal{C}$ is a concept, and n_e is the size of $\mathcal{S}_e$.

2. `SubClassOf` triple set $\mathcal{S}_c = \{(c_i, r_c, c_j)_k\}_{k=1}^{n_c}$, where $c_i, c_j \in \mathcal{C}$ are concepts, c_i is a sub-concept of c_j, and n_c is the size of $\mathcal{S}_c$.

3. Relational triple set $\mathcal{S}_l = \{(h, r, t)_k\}_{k=1}^{n_l}$, where $h, r \in \mathcal{I}$ are head instance and tail instance, $r \in \mathcal{R}_l$ is an instance relation, and n_l is the size of $\mathcal{S}_l$.

Given knowledge graph $\mathcal{KG}$, **knowledge graph embedding with concepts and instances** aims at learning embeddings for instances, concepts, and relations in the same space $\mathbb{R}^k$. For each concept $c \in \mathcal{C}$, we learn a sphere $s(\mathbf{p}, m)$ with $\mathbf{p} \in \mathbb{R}^k$ and m denoting the sphere center and radius. For each instance $i \in \mathcal{I}$ and instance relation $r \in \mathcal{R}_l$, we learn a low-dimensional vector $\mathbf{i} \in \mathbb{R}^k$ and $\mathbf{r} \in \mathbb{R}^k$ respectively. Specifically, the `instanceOf` and `subClassOf` representations are well-designed so that the transitivity of isA relations can be reserved, namely, `instanceOf-subClassOf`

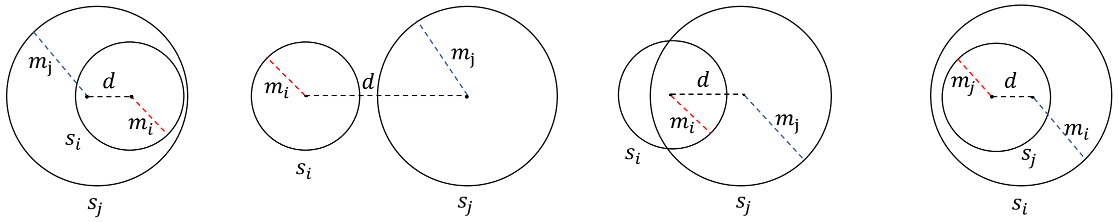

(a) $d < |m_i - m_j| \wedge m_i < m_j$ (GOAL) (b) $d \geq |m_i + m_j|$ (c) $|m_i - m_j| \leq d < |m_i + m_j|$ (d) $d < |m_i - m_j| \wedge m_i \geq m_j$

Figure 2: Four relative positions between sphere s_i and s_j.

transitivity shown in Equation 5:

$$(i, r_e, c_1) \in S_e \wedge (c_1, r_c, c_2) \in S_c \rightarrow (i, r_e, c_2) \in S_e, \quad (5)$$

and `subClassOf-subClassOf` transitivity shown in Equation 6:

$$(c_1, r_c, c_2) \in S_c \wedge (c_2, r_c, c_3) \in S_c \rightarrow (c_1, r_c, c_3) \in S_c. \quad (6)$$

Based on the definition, how to model concepts and isA relations is critical to solve this problem.

4 Our Approach

To differentiate between concepts and instances for knowledge graph embedding, we propose a novel method named TransC. We define different loss functions to measure the relative positions in embedding space, and then jointly learn the representations of concepts, instances, and relations based on the translation-based models.

4.1 TransC

We have three kinds of triples in our triple set $\mathcal{S}$ and define different loss function for them respectively.

InstanceOf Triple Representation. For a given `instanceOf` triple (i, r_e, c), if it is a true triple, $\mathbf{i}$ should be inside the sphere s to represent the `instanceOf` relation between them. Actually, there is another relative position that $\mathbf{i}$ is outside the sphere s. In this condition, the embeddings still need to be optimized. The loss function is defined as

$$f_e(i, c) = ||\mathbf{i} - \mathbf{p}||_2 - m. \quad (7)$$

SubClassOf Triple Representation. For a `subClassOf` triple (c_i, r_c, c_j), just like before, concepts c_i, c_j are encoded as spheres $s_i(\mathbf{p}_i, m_i)$ and $s_j(\mathbf{p}_j, m_j)$. We first denote the distance between the centers of the two spheres as

$$d = ||\mathbf{p}_i - \mathbf{p}_j||_2. \quad (8)$$

If (c_i, r_c, c_j) is a true triple, sphere s_i should be inside sphere s_j (Figure 2a) to represent the `subClassOf` relation between them. Actually, there are three other relative positions between sphere s_i and s_j (as shown in Figure 2). We also have three loss functions under these three conditions:

1. s_i is separate from s_j (Figure 2b). The embeddings still need to be optimized. In this condition, the two spheres need to get closer in optimalization. Therefore, the loss function is defined as

$$f_c(c_i, c_j) = ||\mathbf{p}_i - \mathbf{p}_j||_2 + m_i - m_j. \quad (9)$$

2. s_i intersects with s_j (Figure 2c). This condition is similar to condition 1. The loss function is defined as

$$f_c(c_i, c_j) = ||\mathbf{p}_i - \mathbf{p}_j||_2 + m_i - m_j. \quad (10)$$

3. s_j is inside s_i (Figure 2d). It is different from our target and we should reduce m_j and increase m_i. Hence, the loss function is

$$f_c(c_i, c_j) = m_i - m_j. \quad (11)$$

Relational Triple Representation. For a relational triple (h, r, t), TransC will learn low-dimensional vectors $\mathbf{h}, \mathbf{t}, \mathbf{r} \in \mathbb{R}^k$ for instances and relations. Just like TransE (Bordes et al., 2013), the loss function of this kind of triples is defined as

$$f_r(h, t) = ||\mathbf{h} + \mathbf{r} - \mathbf{t}||_2^2. \quad (12)$$

After having embeddings above, TransC can easily deal with the transitivity of isA relations. If we have true triples (i, r_e, c_i) and (c_i, r_c, c_j), which means $\mathbf{i}$ is inside the sphere s_i and s_i is inside s_j, we can get a result that $\mathbf{i}$ is also inside the sphere s_j. It can be concluded that (i, r_e, c_j) is a true triple and TransC can handle `instanceOf`-`subClassOf` transitivity. Similarly, if we have true triples (c_i, r_c, c_j) and (c_j, r_c, c_k), we can get a result that sphere s_i is inside sphere s_k. It means (c_i, r_e, c_k) is a true triple and TransC can deal with `subClassOf`-`subClassOf` transitivity.

In experiments, we enforce constrains as $||\mathbf{h}||_2 \leq 1$, $||\mathbf{r}||_2 \leq 1$, $||\mathbf{t}||_2 \leq 1$ and $||\mathbf{p}||_2 \leq 1$.

4.2 Training Method

For `instanceOf` triples, we use ξ and ξ' to denote a positive triple and a negative triple. $\mathcal{S}_e$ and $\mathcal{S}'_e$ are used to describe the positive triple set and negative triple set. Then we can define a margin-based ranking loss for `instanceOf` triples:

$$\mathcal{L}_e = \sum_{\xi \in \mathcal{S}_e} \sum_{\xi' \in \mathcal{S}'_e} [\gamma_e + f_e(\xi) - f_e(\xi')]_+, \tag{13}$$

where $[x]_+ \triangleq \max(0, x)$ and γ_e is the margin separating positive triplets and negative triplets. Similarly, for `subClassOf` triples, we will have a ranking loss:

$$\mathcal{L}_c = \sum_{\xi \in \mathcal{S}_c} \sum_{\xi' \in \mathcal{S}'_c} [\gamma_c + f_c(\xi) - f_c(\xi')]_+, \tag{14}$$

and for relational triples, we will have a ranking loss:

$$\mathcal{L}_l = \sum_{\xi \in \mathcal{S}_l} \sum_{\xi' \in \mathcal{S}'_l} [\gamma_l + f_r(\xi) - f_r(\xi')]_+. \tag{15}$$

Finally, we define the overall loss function as linear combinations of these three functions:

$$\mathcal{L} = \mathcal{L}_e + \mathcal{L}_c + \mathcal{L}_l. \tag{16}$$

The goal of training TransC is to minimize the above function, and iteratively update embeddings of concepts, instances, and concepts.

Every triple in our training set has a label to indicate whether the triple is positive or negative. But existing knowledge graph only contains positive triples. We need to generate negative triples by corrupting positive triples. For a relational triple (h, r, t), we replace h or t to generate a negative triple (h', r, t) or (h, r, t'). For example, we get h' by randomly picking from a set $\mathcal{M}_t = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \cdots \cup \mathcal{M}_n$, where n is the number of concepts that t belongs to and $\mathcal{M}_i = \{a | a \in \mathcal{I} \wedge (a, r_e, c_i) \in \mathcal{S}_e \wedge (t, r_e, c_i) \in \mathcal{S}_e \wedge t \neq a\}$. For the other two kinds of triples, we follow the same policy to generate negative triples. We also use two strategies "unif" and "bern" described in (Wang et al., 2014) to replace instances or concepts.

DataSets	YAGO39K	M-YAGO39K
#Instance	39,374	39,374
#Concept	46,110	46,110
#Relation	39	39
#Relational Triple	354,997	354,997
#`InstanceOf` Triple	442,836	442,836
#`SubClassOf` Triple	30,181	30,181
#Valid (Relational Triple)	9,341	9,341
#Test (Relational Triple)	9,364	9,364
#Valid (`InstanceOf` Triple)	5,000	8,650
#Test (`InstanceOf` Triple)	5,000	8,650
#Valid (`SubClassOf` Triple)	1,000	1,187
#Test (`SubClassOf` Triple)	1,000	1,187

Table 1: Statistics of YAGO39K and M-YAGO39K.

5 Experiments and Analysis

We evaluate our method on two typical tasks commonly used in knowledge graph embedding: link prediction (Bordes et al., 2013) and triple classification (Socher et al., 2013).

5.1 Datasets

Most previous work used FB15K and WN18 (Bordes et al., 2013) for evaluation. But these two datasets are not suitable for our model because FB15K mainly consists of instances and WN18 mainly contains concepts. Therefore, we use another popular knowledge graph YAGO (Suchanek et al., 2007) for evaluation, which contains a lot of concepts from WordNet and instances from Wikipedia. We construct a subset of YAGO named YAGO39K for evaluation through the following steps:

(1) We randomly select some relational triples like (h, r, t) from the whole YAGO dataset as our relational triple set $\mathcal{S}_l$.

(2) For every instance and instance relation existed in our relational triples, we save it to construct instance set $\mathcal{I}$ and instance relation set $\mathcal{R}_l$ respectively.

(3) For every `instanceOf` triple (i, r_e, c) in YAGO, if $i \in \mathcal{I}$, we save this triple to construct `instanceOf` triple set $\mathcal{S}_e$.

(4) For every concept existed in `instanceOf` triple set $\mathcal{S}_e$, we save it to construct concept set $\mathcal{C}$.

(5) For every `subClassOf` triple (c_i, r_c, c_j) in YAGO, if $c_i \in \mathcal{C} \wedge c_j \in \mathcal{C}$, we save this triple to construct `subClassOf` triple set $\mathcal{S}_c$.

(6) Finally, we achieve our triple set $\mathcal{S} = \mathcal{S}_e \cup \mathcal{S}_c \cup \mathcal{S}_l$ and our relation set $\mathcal{R} = \{r_e, r_c\} \cup \mathcal{R}_l$.

To evaluate every model's performance in handling the transitivity of isA relations, we generate some new triples based on YAGO39K using the transitivity of isA relations. These new triples will

Experiments	Link Prediction					Triple Classification(%)			
Metric	MRR		Hits@N(%)			Accuracy	Precision	Recall	F1-Score
	`Raw`	`Filter`	`1`	`3`	`10`				
TransE	0.114	0.248	12.3	28.7	51.1	92.1	92.8	91.2	92.0
TransH	0.102	0.215	10.4	24.0	45.1	90.8	91.2	90.3	90.8
TransR	0.112	0.289	15.8	33.8	56.7	91.7	91.6	91.9	91.7
TransD	0.113	0.176	8.9	19.0	35.4	89.3	88.1	91.0	89.5
HolE	0.063	0.198	11.0	23.0	38.4	92.3	92.6	91.9	92.3
DistMult	**0.156**	0.362	22.1	43.6	66.0	93.5	93.9	93.0	93.5
ComplEx	0.058	0.362	29.2	40.7	48.1	92.8	92.6	**93.1**	92.9
TransC (unif)	0.087	**0.421**	28.3	50.0	69.2	93.5	94.3	92.6	93.4
TransC (bern)	0.112	0.420	**29.8**	**50.2**	**69.8**	**93.8**	**94.8**	92.7	**93.7**

Table 2: Experimental results on link prediction and triple classification for relational triples. Hits@N uses results of "Filter" evaluation setting.

be added to valid and test datasets of YAGO39K to create a new dataset named M-YAGO39K. Specific steps are described as follows:

(1) For every `instanceOf` triple (i, r_e, c) in valid and test dataset, if (c, r_c, c_j) exists in training dataset, we save a new `instanceOf` triple (i, r_e, c_j).

(2) For every `subClassOf` triple (c_i, r_c, c_j) in valid and test dataset, if (c_j, r_c, c_k) exists in training dataset, we save a new `subClassOf` triple (c_i, r_c, c_k).

(3) We add these new triples to valid and test dataset of YAGO39K to get M-YAGO39K.

The statistics of YAGO39K and M-YAGO39K are shown in Table 1.

5.2 Link Prediction

Link Prediction aims to predict the missing h or t for a relational triple (h, r, t). In this task, we need to give a ranking list of candidate instances from the knowledge graph, instead of only giving one best result.

For every test relational triple (h, r, t), we remove the head or tail instance and replace it with all instances existed in knowledge graph, and rank these instances in ascending order of distances calculated by loss function f_r. Just like (Bordes et al., 2013), we use two evaluation metrics in this task: (1) the mean reciprocal rank of all correct instances (MRR) and (2) the proportion of correct instances that rank no larger than N (Hits@N). A good embedding model should achieve a high MRR and a high Hits@N. We note that a corrupted triple may also exist in knowledge graph, which should also be regarded as a correct prediction. However, the above evaluations do not handle this issue and may underestimate the results. Hence, we filter out every triple appeared in our knowledge graph before getting the ranking list. The first evaluation setting is called "Raw" and the second one is called "Filter." We report the experiment results on both settings.

In this task, we use dataset YAGO39K for evaluation. We select learning rate λ for SGD among {0.1, 0.01, 0.001}, the three margins γ_l, γ_e and γ_c among {0.1, 0.3, 0.5, 1, 2}, the dimension of instance vectors and relation vectors n among {20, 50, 100}. The best configurations are determined according to the Hits@10 in valid set. The optimal configurations are: $\gamma_l = 1$, $\gamma_e = 0.1$, $\gamma_c = 1$, $\lambda = 0.001$, $n = 100$ and taking L_2 as dissimilarity. We train every model for 1000 rounds in this task.

Evaluation results on YAGO39K are shown in Table 2. From the table, we can conclude that: (1) TransC significantly outperforms other models in terms of Hits@N. This indicates that TransC can use isA triples' information better than other models, which is helpful for instance representation learning. (2) TransC performs a little bit worse than DistMult in some settings. The reason may be that we determine the best configurations only according to the Hits@10, which may lead to a low MRR. (3) The "bern" sampling trick works well for TransC.

5.3 Triple Classification

Triple Classification aims to judge whether a given triple is correct or not, which is a binary classification task. This triple can be a relational triple, an `instanceOf` triple or a `subClassOf` triple.

Negative triples are needed for evaluation of binary classification. Hence, we construct some negative triples following the same setting in (Socher et al., 2013). There are as many true triples as negative triples in both valid and test set.

For triple classification, we set a threshold δ_r for every relation r. For a given test triple, if its

Datasets	YAGO39K				M-YAGO39K			
Metric	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
TransE	82.6	83.6	81.0	82.3	71.0↓	81.4↓	54.4↓	65.2↓
TransH	82.9	83.7	81.7	82.7	70.1↓	80.4↓	53.2↓	64.0↓
TransR	80.6	79.4	**82.5**	80.9	70.9↓	73.0↓	66.3↓	69.5↓
TransD	83.2	84.4	81.5	82.9	72.5↓	73.1↓	71.4↓	72.2↓
HolE	82.3	86.3	76.7	81.2	74.2↓	81.4↓	62.7↓	70.9↓
DistMult	**83.9**	**86.8**	80.1	**83.3**	70.5↓	86.1↓	49.0↓	62.4↓
ComplEx	83.3	84.8	81.1	82.9	70.2↓	84.4↓	49.5↓	62.4↓
TransC (unif)	80.2	81.6	80.0	79.7	**85.5**↑	**88.3**↑	81.8↑	85.0↑
TransC (bern)	79.7	83.2	74.4	78.6	85.3↑	86.1↑	**84.2**↑	**85.2**↑

Table 3: Experimental results on `instanceOf` triple classification(%).

Datasets	YAGO39K				M-YAGO39K			
Metric	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
TransE	77.6	72.2	89.8	80.0	76.9↓	72.3↑	87.2↓	79.0↓
TransH	80.2	76.4	87.5	81.5	79.1↓	72.8↓	92.9↑	81.6↑
TransR	80.4	74.7	91.9	82.4	80.0↓	73.9↓	92.9↑	82.3↓
TransD	75.9	70.6	88.8	78.7	76.1↑	70.7↑	89.0↑	78.8↑
HolE	70.5	73.9	63.3	68.2	66.6↓	72.3↓	53.7↓	61.7↓
DistMult	61.9	68.7	43.7	53.4	60.7↓	71.7↑	35.5↓	47.7↓
ComplEx	61.6	71.5	38.6	50.1	59.8↓	65.6↓	41.4↑	50.7↑
TransC (unif)	82.9	77.1	93.7	84.6	83.0↑	77.5↑	**93.1**↓	84.7↑
TransC (bern)	**83.7**	**78.1**	**93.9**	**85.2**	**84.4**↑	**80.7**↑	90.4↓	**85.3**↑

Table 4: Experimental results on `subClassOf` triple classification(%).

loss function is smaller than δ_r, it will be classified as positive, otherwise negative. δ_r is obtained by maximizing the classification accuracy on valid set.

In this task, we use dataset YAGO39K and M-YAGO39K for evaluation. Parameters are selected in the same way as in link prediction. The best configurations are determined by accuracy in valid set. The optimal configurations for YAGO39K are: $\gamma_l = 1$, $\gamma_e = 0.1$, $\gamma_c = 0.1$, $\lambda = 0.001$, $n = 100$ and taking L_2 as dissimilarity. The optimal configurations for M-YAGO39K are: $\gamma_l = 1$, $\gamma_e = 0.1$, $\gamma_c = 0.3$, $\lambda = 0.001$, $n = 100$ and taking L_2 as dissimilarity. For both datasets, we traverse all the training triples for 1000 rounds.

Our datasets have three kinds of triples. Hence, we do experiments on them respectively. Experimental results for relational triples, `instanceOf` triples, and `subClassOf` triples are shown in Table 2, Table 3, and Table 4 respectively. In Table 3 and Table 4, a rising arrow means performance of this model have a promotion from YAGO39K to M-YAGO39K and a down arrow means a drop.

From Table 2, we can learn that: (1) TransC outperforms all previous work in relational triple classification. (2) The "bern" sampling trick works better than "unif" in TransC.

From Table 3 and Table 4, we can conclude that: (1) On YAGO39K, some compared models perform better than TransC in `instanceOf` triple classification. This is because that `instanceOf` has most triples (53.5%) among all relations in YAGO39K. This relation is trained superabundant times and nearly achieves the best performance, which has an adverse effect on other triples. TransC can find a balance between them and all triples achieve a good performance. (2) On YAGO39K, TransC outperforms other models in `subClassOf` triple classification. As shown in Table 1, `subClassOf` triples are much less than `instanceOf` triples. Hence, other models can not achieve the best performance under the bad influence of `instanceOf` triples. (3) On M-YAGO39K, TransC outperforms previous work in both `instanceOf` triple classification and `subClassOf` triple classification, which indicates that TransC can handle the transitivity of isA relations much better than other models. (4) After comparing experimental results in YAGO39K and M-YAGO39K, we can find that most previous work's performance suffers a big drop in `instanceOf` triple classification and a small drop in `subClassOf` triple classification. This shows that previous work can not deal with `instanceOf-subClassOf` transitivity well. (5) In TransC, nearly all performances have a significant promotion from YAGO39K to M-YAGO39K. Both `instanceOf-subClassOf` transitivity and `subClassOf-subClassOf` transitivity are solved well in TransC.

5.4 Case Study

We have shown that TransC have a good performance for knowledge graph embedding and dealing with transitivity of isA relations. In this section, we present an example of finding new `instanceOf` triples and `subClassOf` triples using results of TransC.

As shown in Figure 3, *New York City* is an instance and others are concepts. The solid lines represent the triples from our datasets and the dotted lines represent the facts inferred by our model. TransC can find two new `instanceOf` triples (*New York City*, `instanceOf`, *City*) and (*New York City*, `instanceOf`, *Municipality*). It can also find a new `subClassOf` triple (*Port Cities*, `subClassOf`, *City*). Following the transitivity of isA relations, we can know all these three new triples are right. Unfortunately, most previous work regards these three triples as wrong, which means they can not handle transitivity of isA relations well.

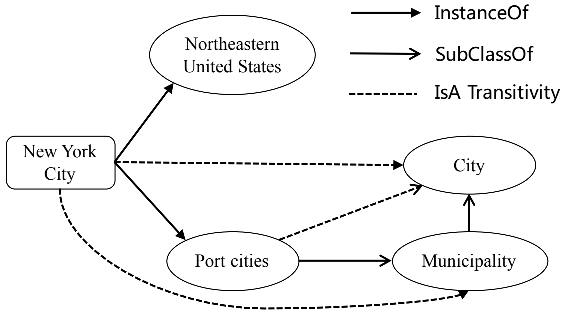

Figure 3: An inference example of TransC.

6 Conclusion and Future Work

In this paper, we propose a new knowledge embedding model named TransC. TransC embeds instances, concepts, and relations in the same space to deal with the transitivity of isA relations. We create a new dataset YAGO39K for evaluation. Experiment results show that TransC outperforms previous translation-based models in most cases. Besides, It can also handle the transitivity of isA relations much better than other models. In our future work, we will explore the following research directions: (1) Sphere is a simple model to represent a concept in semantic space, but it still have some limits since it is too naive. we will try to find a more expressive model instead of spheres to represent concepts. (2) A concept may have different meanings in different triples. We will try to use several typical vectors of instances as a concept's centers to represent different meanings of a concept. Then a concept can have different embeddings in different triples.

Acknowledgments

The work is supported by NSFC key project (No. 61533018, U1736204, 61661146007), Ministry of Education and China Mobile Research Fund (No. 20181770250), and THUNUS NExT Co-Lab.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Jun Feng, Minlie Huang, Mingdong Wang, Mantong Zhou, Yu Hao, and Xiaoyan Zhu. 2016. Knowledge graph embedding by flexible translation. In *KR*.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly embedding knowledge graphs and logical rules. In *EMNLP*.

Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with gaussian embedding. In *CIKM*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL*.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *AAAI*.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*.

Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016. Holographic embeddings of knowledge graphs. In *AAAI*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *NAACL*.

Eleanor H. Rosch. 1973. Natural categories. *Cognitive Psychology*, 4(3):328–350.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. 2011. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *IJCAI*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

Zhigang Wang and Juan-Zi Li. 2016. Text-enhanced representation learning for knowledge graph. In *IJCAI*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016a. From one point to a manifold: knowledge graph embedding for precise link prediction. In *AAAI*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016b. Transg: A generative model for knowledge graph embedding. In *ACL*.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

One-Shot Relational Learning for Knowledge Graphs

Wenhan Xiong[†], **Mo Yu**[*], **Shiyu Chang**[*], **Xiaoxiao Guo**[*], **William Yang Wang**[†]
[†] University of California, Santa Barbara
[*] IBM Research
{xwhan, william}@cs.ucsb.edu, yum@us.ibm.com, {shiyu.chang, xiaoxiao.guo}@ibm.com

Abstract

Knowledge graphs (KGs) are the key components of various natural language processing applications. To further expand KGs' coverage, previous studies on knowledge graph completion usually require a large number of training instances for each relation. However, we observe that long-tail relations are actually more common in KGs and those newly added relations often do not have many known triples for training. In this work, we aim at predicting new facts under a challenging setting where only one training instance is available. We propose a one-shot relational learning framework, which utilizes the knowledge extracted by embedding models and learns a matching metric by considering both the learned embeddings and one-hop graph structures. Empirically, our model yields considerable performance improvements over existing embedding models, and also eliminates the need of re-training the embedding models when dealing with newly added relations.[1]

1 Introduction

Large-scale knowledge graphs (Suchanek et al., 2007; Vrandečić and Krötzsch, 2014; Bollacker et al., 2008; Auer et al., 2007; Carlson et al., 2010) represent every piece of information as binary relationships between entities, usually in the form of triples *i.e.* *(subject, predicate, object)*. This kind of structured knowledge is essential for many downstream applications such as Question Answering and Semantic Web.

Despite KGs' large scale, they are known to be highly incomplete (Min et al., 2013). To automatically complete KGs, extensive research efforts (Nickel et al., 2011; Bordes et al., 2013; Yang et al., 2014; Trouillon et al., 2016; Lao and Cohen, 2010; Neelakantan et al., 2015; Xiong et al., 2017; Das et al., 2017; Chen et al., 2018) have been made to build relational learning models that could infer missing triples by learning from existing ones. These methods explore the statistical information of triples or path patterns to infer new facts of existing relations; and have achieved considerable performance on various public datasets.

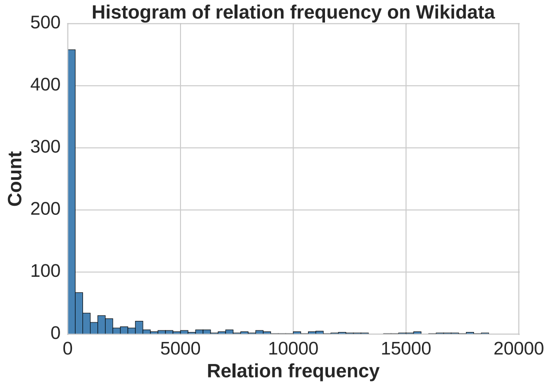

Figure 1: The histogram of relation frequencies in Wikidata. There are a large portion of relations that only have a few triples.

However, those datasets (*e.g.* FB15k, WN18) used by previous models mostly only cover common relations in KGs. For more practical scenarios, we believe the desired KG completion models should handle two key properties of KGs. First, as shown in Figure 1, a large portion of KG relations are actually long-tail. In other words, they have very few instances. But intuitively, the fewer training triples that one relation has, the more KG completion techniques could be of use. Therefore, it is crucial for models to be able to complete relations with limited numbers of triples. However, existing research usually assumes the availability of sufficient training triples for all relations, which

---

[1] Code and datasets could be found at `https://github.com/xwhan/One-shot-Relational-Learning`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1980–1990
Brussels, Belgium, October 31 - November 4, 2018.

limits their usefulness on sparse long-tail relations.

Second, to capture up-to-date knowledge, real-world KGs are often dynamic and evolving at any given moment. New relations will be added whenever new knowledge is acquired. If a model can predict new triples given only a small number of examples, a large amount of human effort could be spared. However, to predict target relations, previous methods usually rely on well-learned representations of these relations. In the dynamic scenario, the representations of new relations cannot be sufficiently trained given limited training instances, thus the ability to adapt to new relations is also limited for current models.

In contrast to previous methods, we propose a model that depends only on the entity embeddings and local graph structures. Our model aims at learning a matching metric that can be used to discover more similar triples given one reference triple. The learnable metric model is based on a permutation-invariant network that effectively encodes the one-hop neighbors of entities, and also a recurrent neural network that allows multi-step matching. Once trained, the model will be able to make predictions about any relation while existing methods usually require fine-tuning to adapt to new relations. With two newly constructed datasets, we show that our model can achieve consistent improvement over various embedding models on the one-shot link prediction task.

In summary, our contributions are three-fold:

- We are the first to consider the long-tail relations in the link prediction task and formulate the problem as few-shot relational learning;
- We propose an effective one-shot learning framework for relational data, which achieves better performance than various embedding-based methods;
- We also present two newly constructed datasets for the task of one-shot knowledge graph completion.

2 Related Work

Embedding Models for Relational Learning Various models have been developed to model relational KGs in continous vector space and to automatically infer missing links. RESCAL (Nickel et al., 2011) is one of the earlier work that models the relationship using tensor operations. Bordes et al. (2013) proposed to model relationships in the 1-D vector space. Following this line of research, more advanced models such as DistMult (Yang et al., 2014), ComplEx (Trouillon et al., 2016) and ConvE (Dettmers et al., 2017) have been proposed. These embedding-based models usually assume enough training instances for all relations and entities and do not pay attention to those sparse symbols. More recently, several models (Shi and Weninger, 2017; Xie et al., 2016) have been proposed to handle unseen entities by leveraging text descriptions. In contrast to these approaches, our model deals with long-tail or newly added relations and focuses on one-shot relational learning without any external information, such as text descriptions of entities or relations.

Few-Shot Learning Recent deep learning based few-shot learning approaches fall into two main categories: (1) *metric based approaches* (Koch, 2015; Vinyals et al., 2016; Snell et al., 2017; Yu et al., 2018), which try to learn generalizable metrics and the corresponding matching functions from a set of training tasks. Most methods in this class adopt the general matching framework proposed in deep siamese network (Koch, 2015). One example is the Matching Networks (Vinyals et al., 2016), which make predictions by comparing the input example with a small labeled support set; (2) *meta-learner based approaches* (Ravi and Larochelle, 2017; Munkhdalai and Yu, 2017; Finn et al., 2017; Li et al., 2017), which aim to learn the optimization of model parameters (by either outputting the parameter updates or directly predicting the model parameters) given the gradients on few-shot examples. One example is the LSTM-based meta-learner (Ravi and Larochelle, 2017), which learns the step size for each dimension of the stochastic gradients. Besides the above categories, there are also some other styles of few-shot learning algorithms, *e.g.* Bayesian Program Induction (Lake et al., 2015), which represents concepts as simple programs that best explain observed examples under a Bayesian criterion.

Previous few-shot learning research mainly focuses on vision and imitation learning (Duan et al., 2017) domains. In the language domain, Yu et al. (2018) proposed a multi-metric based approach for text classification. To the best of our knowledge, this work is the first research on few-shot learning for knowledge graphs.

3 Background

3.1 Problem Formulation

Knowledge graphs $\mathcal{G}$ are represented as a collection of triples $\{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ are the entity set and relation set. The task of knowledge graph completion is to either predict unseen relations r between two existing entities: $(h, ?, t)$ or predict the tail entity t given the head entity and the query relation: $(h, r, ?)$. As our purpose is to infer unseen facts for newly added or existing long-tail relations, we focus on the latter case. In contrast to previous work that usually assumes enough triples for the query relation are available for training, this work studies the case where only one training triple is available. To be more specific, the goal is to rank the true tail entity t_{true} higher than other candidate entities $t \in \mathcal{C}_{h,r}$, given only an example triple (h_0, r, t_0). The candidates set is constructed using the entity type constraint (Toutanova et al., 2015). It is also worth noting that when we predict new facts of the relation r, we only consider a closed set of entities, *i.e.* no unseen entities during testing. For open-world settings where new entities might appear during testing, external information such as text descriptions about these entities are usually required and we leave this to future work.

3.2 One-Shot Learning Settings

This section describes the settings for the training and evaluation of our one-shot learning model.

The goal of our work is to learn a metric that could be used to predict new facts with one-shot examples. Following the standard one-shot learning settings (Vinyals et al., 2016; Ravi and Larochelle, 2017), we assume access to a set of training tasks. In our problem, each training task corresponds to a KG relations $r \in \mathcal{R}$, and has its own training/testing triples: $T_r = \{D_r^{train}, D_r^{test}\}$. This task set is often denoted as the *meta-training* set, $\mathbb{T}_{meta-train}$.

To imitate the one-shot prediction at evaluation time, there is only one triple (h_0, r, t_0) in each D_r^{train}. The $D_r^{test} = \{(h_i, r, t_i, \mathcal{C}_{h_i,r})\}$ consists of the testing triples of r with ground-truth tail entities t_i for each query (h_i, r), and the corresponding tail entity candidates $\mathcal{C}_{h_i,r} = \{t_{ij}\}$ where each t_{ij} is an entity in $\mathcal{G}$. The metric model can thus be tested on this set by ranking the candidate set $\mathcal{C}_{h_i,r}$ given the test query (h_i, r) and the labeled triple in D_r^{train}. We denote an arbitrary ranking-loss function as $\ell_\theta(h_i, r, t_i | \mathcal{C}_{h_i,r}, D_r^{train})$, where θ represents the parameters of our metric model. This loss function indicates how well the metric model works on tuple $(h_i, r, t_i, \mathcal{C}_{h_i,r})$ while observing only one-shot data from D_r^{train}. The objective of training the metric model, *i.e.* the meta-training objective, thus becomes:

$$\min_\theta \mathbb{E}_{T_r} \left[\sum_{(h_i,r,t_i,\mathcal{C}_{h_i,r}) \in D_r^{test}} \frac{\ell_\theta(h_i, r, t_i | \mathcal{C}_{h_i,r}, D_r^{train})}{|D_r^{test}|} \right], \tag{1}$$

where T_r is sampled from the meta-training set $\mathbb{T}_{meta-train}$, and $|D_r^{test}|$ denotes the number of tuples in D_r^{test}.

Once trained, we can use the model to make predictions on new relations $r' \in \mathcal{R}'$, which is called the *meta-testing* step in literature. These meta-testing relations are unseen from meta-training, *i.e.* $\mathcal{R}' \cap \mathcal{R} = \phi$. Each meta-testing relation r' also has its own one-shot training data $D_{r'}^{train}$ and testing data $D_{r'}^{test}$, defined in the same way as in meta-training. These meta-testing relations form a meta-test set $\mathbb{T}_{meta-test}$.

Moreover, we leave out a subset of relations in $\mathbb{T}_{meta-train}$ as the meta-validation set $\mathbb{T}_{meta-validation}$. Because of the assumption of one-shot learning, the meta-testing relations do not have validation sets like in the traditional machine learning setting. Otherwise, the metric model will actually see more than one-shot labeled data during meta-testing, thus the one-shot assumption is violated.

Finally, we assume that the method has access to a *background knowledge graph* $\mathcal{G}'$, which is a subset of $\mathcal{G}$ with all the relations from $\mathbb{T}_{meta-train}$, $\mathbb{T}_{meta-validation}$ and $\mathbb{T}_{meta-test}$ removed.

4 Model

In this section, we describe the proposed model for similarity metric learning and also the corresponding loss function ℓ we use to train our model.

The core of our proposed model is a similarity function $\mathcal{M}((h, t), (h', t') | \mathcal{G}')$. Thus for any query relation r, as long as there is one known fact (h_0, r, t_0), the model could predict the likelihood of testing triples $\{(h_i, r, t_{ij}) | t_{ij} \in \mathcal{C}_{h_i,r}\}$, based on the matching score between each (h_i, t_{ij}) and (h_0, t_0). The implementation of the above matching function involves two sub-problems: (1) the representations of entity pairs; and (2) the comparison function between two entity-pair representations. Our overall model, as shown in Figure 2,

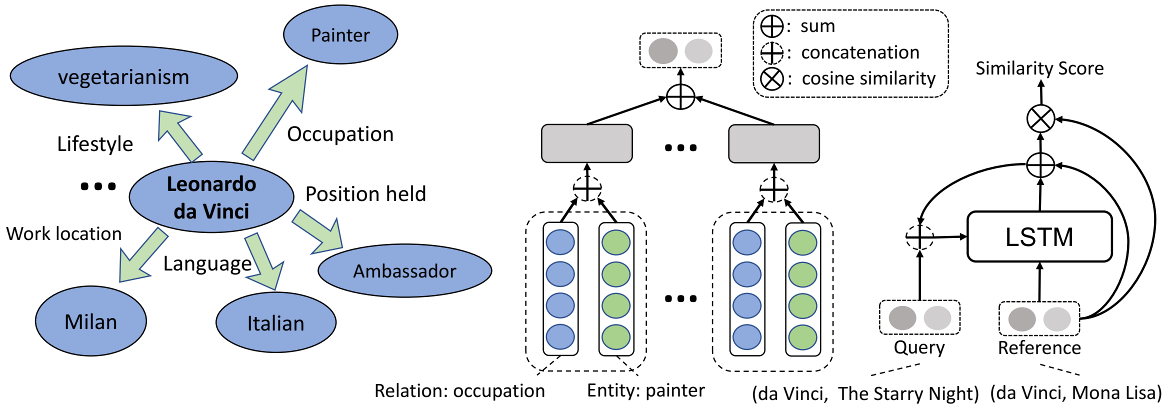

a) Local graph of entity *Leonardo da Vinci* **b) Neighbor Encoder** **c) Matching Processor**

Figure 2: a) and b): Our **neighbor encoder** operating on entity *Leonardo da Vinci*; c): The **matching processor**.

deals with the above two problems with two major components respectively:

- **Neighbor encoder** (Figure 2b), aims at utilizing the local graph structure to better represent entities. In this way, the model can leverage more information that KG provides for every entity within an entity pair.
- **Matching processor** (Figure 2c), takes the vector representations of any two entity pairs from the neighbor encoder; then performs multi-step matching between two entity-pairs and outputs a scalar as the similarity score.

4.1 Neighbor Encoder

This module is designed to enhance the representation of each entity with its local connections in knowledge graph.

Although the entity embeddings from KG embedding models (Bordes et al., 2013; Yang et al., 2014) already have relational information encoded, previous work (Neelakantan et al., 2015; Lin et al., 2015a; Xiong et al., 2017) showed that explicitly modeling the structural patterns, such as paths, is usually beneficial for relationship prediction. In view of this, we propose to use a neighbor encoder to incorporate graph structures into our metric-learning model. In order to benefit from the structural information while maintaining the efficiency to easily scale to real-world large-scale KGs, our neighbor encoder only considers entities' local connections, *i.e.* the one-hop neighbors.

For any given entity e, its local connections form a set of (*relation, entity*) tuples. As shown in Figure 2a, for the entity *Leonardo da Vinci*, one of such tuples is (*occupation, painter*). We refer this neighbor set as as $\mathcal{N}_e = \{(r_k, e_k) | (e, r_k, e_k) \in \mathcal{G}'\}$. The purpose of our neighbor encoder is to encode $\mathcal{N}_e$ and output a vector as the latent representation of e. Because this is a problem of encoding sets with varying sizes, we hope the encoding function can be (1) invariant to permutations and also (2) insensitive to the size of the neighbor set. Inspired by the results from (Zaheer et al., 2017), we use the following function f that satisfies the above properties:

$$f(\mathcal{N}_e) = \sigma(\frac{1}{|\mathcal{N}_e|} \sum_{(r_k,e_k)\in\mathcal{N}_e} C_{r_k,e_k}). \tag{2}$$

where C_{r_k,e_k} is the feature representation of a relation-entity pair (r_k, e_k) and σ is the activation function. In this paper we set $\sigma = \tanh$ which achieves the best performance on $\mathbb{T}_{meta-validation}$.

To encode every tuple $(r_k, e_k) \in \mathcal{N}_e$ into C_{r_k,e_k}, we first use an embedding layer **emb** with dimension d (which can be pre-trained using existing embedding-based models) to get the vector representations of r_k and e_k:

$$v_{r_k} = \mathbf{emb}(r_k), v_{e_k} = \mathbf{emb}(e_k)$$

Dropout (Srivastava et al., 2014) is applied here to the vectors v_{r_k}, v_{e_k} to achieve better generalization. We then apply a feed-forward layer to encode the interaction within this tuple:

$$C_{r_k,e_k} = W_c(v_{r_k} \oplus v_{e_k}) + b_c, \tag{3}$$

where $W_c \in R^{d\times 2d}, b_c \in R^d$ are parameters to be learned and $\oplus$ denotes concatenation.

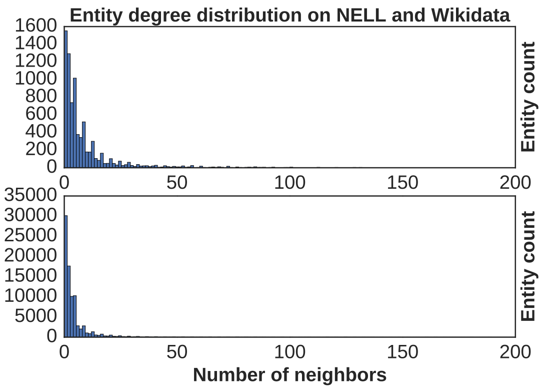

Figure 3: The distribution of entities' degrees (numbers of neighbors) on our two datasets. Since we work on closed-set of entities, we draw the figure by considering the intersection between entities in our background knowledge $\mathcal{G}'$ and the entities appearing in $\mathbb{T}_{meta-train}$, $\mathbb{T}_{meta-validation}$ or $\mathbb{T}_{meta-test}$. Note that all triples in $\mathbb{T}_{meta-train}$, $\mathbb{T}_{meta-validation}$ or $\mathbb{T}_{meta-test}$ are removed from $\mathcal{G}'$. Upper: NELL; Lower: Wikidata.

To enable batching during training, we manually specify the maximum number of neighbors and use all-zero vectors as "dummy" neighbors. Although different entities have different degrees (number of neighbors), the degree distribution is usually very concentrated, as shown in Figure 3. We can easily find a proper bound as the maximum number of neighbors to batch groups of entities.

The neighbor encoder module we propose here is similar to the Relational Graph Convolutional Networks (Schlichtkrull et al., 2017) in the sense that we also use the shared kernel $\{W_c, b_c\}$ to encode the neighbors of different entities. But unlike their model that operates on the whole graph and performs multiple steps of information propagation, we only encode the local graphs of the entities and perform one-step propagation. This enables us to easily apply our model to large-scale KGs such as Wikidata. Besides, their model also does not operate on pre-trained graph embeddings. We leave the investigation of other graph encoding strategies, e.g. (Xu et al., 2018; Song et al., 2018), to future work.

4.2 Matching Processor

Given the neighbor encoder module, now we discuss how we can do effective similarity matching based on our recurrent matching processor. By applying $f(\mathcal{N}_e)$ to the reference entity pair (h_0, t_0) and any query entity pair (h_i, t_{ij}), we get two neighbor vectors for each: $[f(\mathcal{N}_{h_0}); f(\mathcal{N}_{t_0})]$ and $[f(\mathcal{N}_{h_i}); f(\mathcal{N}_{t_{ij}})]$. To get a similarity score that can be used to rank (h_i, t_{ij}) among other candidates, we can simply concatenate the $f(\mathcal{N}_h)$ and $f(\mathcal{N}_t)$ in each pair to form a single pair representation vector, and calculate the cosine similarity between pairs. However, this simple metric model turns out to be too shallow and does not give good performance. To enlarge our model's capacity, we leverage a LSTM-based (Hochreiter and Schmidhuber, 1997) recurrent "processing" block (Vinyals et al., 2015, 2016) to perform multi-step matching. Every process step is defined as follows:

Algorithm 1 One-shot Training

```
Input:
a) Meta-training task set $\mathbb{T}_{meta-training}$;
b) Pre-trained KG embeddings (excluding relation in $\mathbb{T}_{meta-training}$);
c) Initial parameters $\theta$ of the metric model;
for epoch = 0:M-1 do
    Shuffle the tasks in $\mathcal{T}_{meta-learning}$
    for $T_r$ in $\mathcal{T}_{meta-learning}$ do
        Sample one triple as the reference
        Sample a batch $B^+$ of query triples
        Pollute the tail entity of query triples to get $B^-$
        Calculate the matching scores for triple in $B^+$ and $B^-$
        Calculate the batch loss $\mathcal{L} = \sum_B \ell$
        Update $\theta$ using gradient $g \propto \nabla \mathcal{L}$
    end for
end for
```

$$
\begin{aligned}
h'_{k+1}, c_{k+1} &= LSTM(q, [h_k \oplus s, c_k]) \\
h_{k+1} &= h'_{k+1} + q \\
score_{k+1} &= \frac{h_{k+1} \odot s}{\|h_{k+1}\| \, \|s\|},
\end{aligned}
\tag{4}
$$

where $LSTM(x, [h, c])$ is a standard LSTM cell with input x, hidden state h and cell state c, and $s = f(\mathcal{N}_{h_0}) \oplus f(\mathcal{N}_{t_0}), q = f(\mathcal{N}_{h_i}) \oplus f(\mathcal{N}_{t_{ij}})$ are the concatenated neighbor vectors of the reference pair and query pair. After K processing steps[2], we use $score_K$ as the final similarity score between the query and support entity pair. For every query $(h_i, r, ?)$, by comparing (h_i, t_{ij}) with (h_0, t_0), we can get the ranking scores for every $t_{ij} \in \mathcal{C}_{h_i,r}$.

4.3 Loss Function and Training

For a query relation r and its reference/training triple (h_0, r, t_0), we collect a group of positive (true) query triples $\{(h_i, r, t_i^+) | (h_i, r, t_i^+) \in \mathcal{G}\}$ and construct another group negative (false) query

---

[2] K is a hyperparameter to be tuned.

triples $\{(h_i, r, t_i^-)|(h_i, r, t_i^-) \notin \mathcal{G}\}$ by polluting the tail entities. Following previous embedding-based models, we use a hinge loss function to optimize our model:

$$\ell_\theta = max(0, \gamma + score_\theta^- - score_\theta^+), \quad (5)$$

where $score_\theta^+$ and $score_\theta^-$ are scalars calculated by comparing the query triple $(h_i, r, t_i^+/t_i^-)$ with the reference triple (h_0, r, t_0) using our metric model, and the margin γ is a hyperparameter to be tuned. For every training episode, we first sample one task/relation T_r from the meta-training set $\mathbb{T}_{meta-training}$. Then from all the known triples in T_r, we sample one triple as the reference/training triple D_r^{train} and a batch of other triples as the positive query/test triples D_r^{test}. The detail of the training process is shown in Algorithm 1. Our experiments are discussed in the next section.

5 Experiments

5.1 Datasets

Dataset	# Ent.	# R.	# Triples	# Tasks
NELL-One	68,545	358	181,109	67
Wiki-One	4,838,244	822	5,859,240	183

Table 1: Statistics of the Datasets. # Ent. denotes the number of unique entities and # R. denotes the number of all relations. # Tasks denotes the number of relations we use as one-shot tasks.

Existing benchmarks for knowledge graph completion, such as FB15k-237 (Toutanova et al., 2015) and YAGO3-10 (Mahdisoltani et al., 2013) are all small subsets of real-world KGs. These datasets consider the same set of relations during training and testing and often include sufficient training triples for every relation. To construct datasets for one-shot learning, we go back to the original KGs and select those relations that do not have too many triples as one-shot task relations. We refer the rest of the relations as background relations, since their triples provide important background knowledge for us to match entity pairs.

Our first dataset is based on NELL (Mitchell et al., 2018), a system that continuously collects structured knowledge by reading webs. We take the latest dump and remove those inverse relations. We select the relations with less than 500 but more than 50 triples[3] as one-shot tasks. To show that our model is able to operate on large-scale KGs, we follow the similar process to build another larger dataset based on Wikidata (Vrandečić and Krötzsch, 2014). The dataset statistics are shown in Table 1. Note that the Wiki-One dataset is an order of magnitude larger than any other benchmark datasets in terms of the numbers of entities and triples. For NELL-One, we use 51/5/11 task relations for training/validation/testing. For Wiki-One, the division ratio is 133:16:34.

5.2 Implementation Details

In our experiments, we consider the following embedding-based methods: RESCAL (Nickel et al., 2011), TransE (Bordes et al., 2013), DistMult (Yang et al., 2014) and ComplEx (Trouillon et al., 2016). For TransE, we use the code released by Lin et al. (2015b). For the other models, we have tried the code released by Trouillon et al. (2016) but it gives much worse results than TransE on our datasets. Thus we use our own implementations based on PyTorch (Paszke et al., 2017) for comparison. When evaluating existing embedding models, during training, we use not only the triples of background relations but also all the triples of the training relations and the one-shot training triple of those validation/test relations. However, since the proposed metric model does not require the embeddings of query relations, we only include the triples of the background relations for embedding training. As TransE and DistMult use 1-D vectors to represent entities and relations, they can be directly used in our natching model. While for RESCAL, since it uses matrices to represent relations, we employ mean-pooling over these matrices to get 1-D embeddings. For the ComplEx model, we use the concatenation of the real part and imaginary part. The hyperparameters of our model are tuned on the validation task set and can be found in the appendix.

Apart from the above embedding models, a more recent method (Dettmers et al., 2017) applies convolution to model relationships and achieves the best performance on several benchmarks. For every query $(h, r, ?)$, their model enumerates the whole entity set to get positive and negative triples for training. We find that this training paradigm takes lots of computational resources when dealing with large entity sets and cannot scale to real-world KGs such as Wikidata[4] that have millions

---

[3] We want to have enough triples for evaluation.

[4] On a GPU card with 12GB memory, we fail to run their ConvE model on Wiki-One with batch size 1.

	NELL-One				Wiki-One			
Model	MRR	Hits@10	Hits@5	Hits@1	MRR	Hits@10	Hits@5	Hits@1
RESCAL	.071/.140	.100/.229	.082/.186	.048/.089	.119/.072	.167/.082	.132/.062	.093/.051
TransE	.082/.093	.177/.192	.126/.141	.032/.043	.023/.035	.036/.052	.029/.043	.015/.025
DistMult	.075/.102	.128/.177	.093/.126	.045/.066	.042/.048	.086/.101	.055/.070	.017/.019
ComplEx	.072/.131	.128/.223	.041/.086	.041/.086	.079/.069	.148/.121	.106/.092	.046/.040
GMatching (RESCAL)	.144/**.188**	.277/.305	.216/.243	.087/**.133**	.113/.139	.330/.305	.180/.228	.033/.061
GMatching (TransE)	**.168**/.171	.293/.255	**.239**/.210	**.103**/.122	.167/.219	.349/.328	.289/.269	.083/.163
GMatching (DistMult)	.119/.171	.238/.301	.183/.221	.054/.114	.190/**.222**	**.384/.340**	**.291**/.271	.114/**.164**
GMatching (ComplEx)	.132/.185	**.308/.313**	.232/**.260**	.049/.119	**.201**/.200	.350/.336	.231/**.272**	**.141**/.120
GMatching (Random)	.083/.151	.211/.252	.135/.186	.024/.103	.174/.198	.309/.299	.222/.260	.121/.133

Table 2: Link prediction results on validation/test relations. KG embeddings baselines are shown at the top of the table and our one-shot learning (GMatching) results are shown at the bottom. **Bold** numbers denote the best results on meta-validation/meta-test. Underline numbers denote the model selection results from all KG embeddings baselines, or from all one-shot methods, i.e. selecting the method with the best validation score and reporting the corresponding test score.

of entities. For the scalability concern, our experiments only consider models that use negative sampling for training.

5.3 Results

The main results of our methods are shown in Table 2. We denote our method as "GMatching" since our model is trained to match local graph patterns. We use mean reciprocal rank (MRR) and Hits@K to evaluate different models. We can see that our method produces consistent improvements over various embedding models on these one-shot relations. The improvements are even more substantial on the larger Wiki-One dataset. To investigate the learning power of our model, we also try to train our metric model with randomly initialized embeddings. Surprisingly, although the results are worse than the metric models with pre-trained embeddings, they are still superior to the baseline embedding models. This suggests that, by incorporating the neighbor entities into our model, the embeddings of many relations and entities actually get updated in an effective way and provide useful information for our model to make predictions on test data.

It is worth noting that once trained, our model can be used to predict any newly added relations without fine-tuning, while existing models usually need to be re-trained to handle those newly added symbols. On a large real-world KG, this re-training process can be slow and highly computational expensive.

Remark on Model Selection Given the existence of various KG embedding models, one interesting experiment is to incorporate model selection into hyper-parameter tuning and choose the best validation model for testing.

If we think about comparing KG embedding and metric learning as two approaches, the results from the model selection process can then be used as the "final" measurement for comparison. For example, the baseline KG embedding achieves best MRR on Wiki-One with RESCAL (11.9%), so we report the corresponding testing MRR (7.2%) as the final model selection result for KG embedding approach. In this way, at the top half of Table 2, we select the best KG embedding method according to the validation performance. The results are highlighted with underlines. Similarly, we select the best metric learning approach at the bottom.

Our metric-based method outperforms KG embedding by a large margin from this perspective as well. Taking MRR as an example, the selected metric model achieves 17.1% on NELL-One and 20.0% on Wiki-One; while the results of KG embedding are 9.3% and 7.2%. The improvement is 7.8% and 12.8% respectively.

5.4 Analysis on Neighbor-Encoder

As our model leverages entities' local graph structures by encoding the neighbors, here we try to investigate the effect of the neighbor set by restricting the maximum number of neighbors. If the size of the true neighbor set is larger than the maximum limit, the neighbors are then selected by random sampling. Figure 4 shows the learning curves of different settings. These curves are based on the Hits@10 calculated on the validation set. We see that encoding more neighbors

		MRR		Hits@10	
Relations	# Candidates	GMatching	ComplEx	GMatching	ComplEx
sportsGameSport	123	0.424	**0.466**(0.139*)	**1.000**	0.479(0.200*)
athleteInjuredHisBodypart	299	0.025	0.026(**0.330***)	0.015	0.059(**0.444***)
animalSuchAsInvertebrate	786	0.447	0.333(**0.555***)	0.626	0.587(**0.783***)
automobilemakerDealersInCountry	1084	**0.453**	0.245(0.396*)	**0.821**	0.453(0.500*)
sportSchoolIncountry	2100	**0.534**	0.324(0.294*)	**0.745**	0.529(0.571*)
politicianEndorsesPolitician	2160	0.093	0.026(**0.194***)	0.226	0.047(**0.357***)
agriculturalProductFromCountry	2222	**0.120**	0.029(0.042*)	**0.288**	0.058(0.086*)
producedBy	3174	0.085	0.040(**0.165***)	0.179	0.075(**0.241***)
automobilemakerDealersInCity	5716	0.026	0.024(**0.041***)	0.040	0.051(**0.174***)
teamCoach	10569	0.017	0.065(**0.376***)	0.024	0.079(**0.547***)
geopoliticalLocationOfPerson	11618	0.028	0.016(**0.284***)	0.035	0.035(**0.447***)

Table 3: Results decomposed over different relations. "$\star$" denotes the results with standard training settings and "# Candidates" denotes the size of candidate entity set.

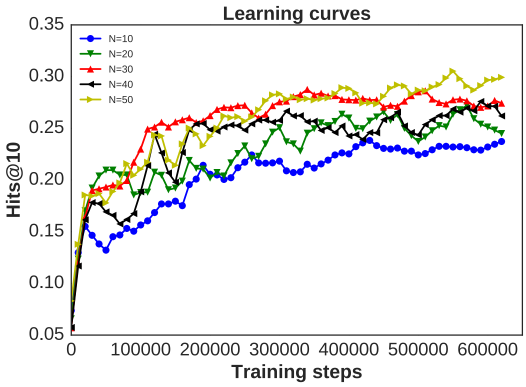

Figure 4: The learning curves on NELL-one. Every run uses different number of neighbors. The y-axis is Hits@10 calculated on all the validation relations.

Configuration	Hits@10
Full Model with ComplEx	**.308/.313**
w/o Matching Processor	.266/.269
w/o Neighbor Encoder	.248/.296
w/o Scaling Factor	.229/.219

Table 4: Ablation on different components.

for every entity generally leads to better performance. We also observe that the model that encodes 40 neighbors in maximum actually yields worse performance than the model that only encodes 30 neighbors. We think the potential reason is that for some entity pairs, there are some local connections that are irrelevant and provide noisy information to the model.

5.5 Ablation Studies

We conduct ablation studies using the model that achieves the best Hits@10 on the NELL-One dataset. The results are shown in Table 4. We use Hits@10 on validation and test set for comparison, as the hyperparameters are selected using this evaluation metric. We can see that both the matching processor[5] and the neighbor encoder play important roles in our model. Another important observation is that the scaling factor $1/\mathcal{N}_e$ turns out to be very essential for the neighbor encoder. Without scaling, the neighbor encoder actually gives worse results compared to the simple embedding-based matching.

5.6 Performance on Different Relations

When testing various models, we observe that the results on different relations are actually of high variance. Table 3 shows the decomposed results on NELL-One generated by our best metric model (GMatching-ComplEx) and its corresponding embedding method. For reference, we also report the embedding model's performance under standard training settings where 75% of the triples (instead of only one) are used for training and the rest are used for testing. We can see that relations with smaller candidate sets are generally easier and our model could even perform better than the embedding model trained under standard settings. For some relations such as *athleteInjuredHisBodypart*, their involved entities have very few connections in KG. It is as expected that one-shot learning on these kinds of relations is quite challenging. Those relations with lots of (>3000) candidates are challenging for all models. Even for embedding model with more training triples, the performance on some relations is still very limited. This suggests that the knowledge graph completion task is still far from being solved.

---

[5] Matching without Matching Processor is equivalent to matching using simple cosine similarity.

6 Conclusion

This paper introduces a one-shot relational learning framework that could be used to predict new facts of long-tail relations in KGs. Our model leverages the local graph structure of entities and learns a differentiable metric to match entity pairs. In contrast to existing methods that usually need finetuning to adapt to new relations, our trained model can be directly used to predict any unseen relation and also achieves much better performance in the one-shot setting. Our future work might consider incorporating external text data and also enhancing our model to make better use of multiple training examples in the few-shot learning case.

Acknowledgments

This research is supported by an IBM Faculty Award. We also thank the anonymous reviewers for their useful feedback.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3. Atlanta.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Wang. 2018. Variational knowledge graph reasoning. *arXiv preprint arXiv:1803.06581*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. *arXiv preprint arXiv:1707.01476*.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. 2017. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Gregory Koch. 2015. *Siamese neural networks for one-shot image recognition*. Ph.D. thesis, University of Toronto.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. 2017. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, B Yang, J Betteridge, A Carlson, B Dalvi, M Gardner, B Kisiel, et al. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. *arXiv preprint arXiv:1703.00837*.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base inference. In *2015 aaai spring symposium series*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, volume 1, page 6.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

Baoxu Shi and Tim Weninger. 2017. Open-world knowledge graph completion. *arXiv preprint arXiv:1711.03438*.

Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. *arXiv preprint arXiv:1805.02473*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*, pages 2659–2665.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. *arXiv preprint arXiv:1805.07513*.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3394–3404.

A Hyperparameters

For the NELL dataset, we set embedding size as 100. For Wikidata, we set the embedding size as 50 for faster training with millions of triples. The embeddings are trained for 1,000 epochs. The other hyperparamters are tuned using the Hits@10 metric[6] on the validation tasks. For matching steps, the optimal setting is 2 for NELL-One and 4 for Wiki-One. For the number of neighbors, we find that the maximum limit 50 works the best for both datasets. For parameter updates, we use Adam (Kingma and Ba, 2014) with the initial learning rate 0.001 and we half the learning rate after 200k update steps. The margin used in our loss function is 5.0. The dimension of LSTM's hidden size is 200.

B Few-Shot Experiments

Metrics	GMatching		ComplEx	
	1-shot	5-shot	1-shot	5-shot
MRR	.132/.185	.178/.201	.072/.131	.113/.200
Hits@10	.308/.313	.307/.311	.128/.223	.221/.325
Hits@5	.232/.260	.241/.264	.041/.086	.160/.269
Hits@1	.049/.119	.109/.143	.041/.086	.113/.133

Table 5: 5-shot experiments on NELL-One.

---

[6] The percentage of correct answer ranks within top10.

Regular Expression Guided Entity Mention Mining from Noisy Web Data

Shanshan Zhang, Lihong He, Slobodan Vucetic, Eduard C. Dragut
Computer and Information Sciences Department
Temple University, PA
`{zhang.shanshan, lihong.he, vucetic, edragut}@temple.edu`

Abstract

Many important entity types in web documents, such as dates, times, email addresses, and course numbers, follow or closely resemble patterns that can be described by Regular Expressions (REs). Due to a vast diversity of web documents and ways in which they are being generated, even seemingly straightforward tasks such as identifying mentions of date in a document become very challenging. It is reasonable to claim that it is impossible to create a RE that is capable of identifying such entities from web documents with perfect precision and recall. Rather than abandoning REs as a go-to approach for entity detection, this paper explores ways to combine the expressive power of REs, ability of deep learning to learn from large data, and human-in-the loop approach into a new integrated framework for entity identification from web data. The framework starts by creating or collecting the existing REs for a particular type of an entity. Those REs are then used over a large document corpus to collect weak labels for the entity mentions and a neural network is trained to predict those RE-generated weak labels. Finally, a human expert is asked to label a small set of documents and the neural network is fine tuned on those documents. The experimental evaluation on several entity identification problems shows that the proposed framework achieves impressive accuracy, while requiring very modest human effort.

1 Introduction

Named Entity Recognition (NER) is the task of automatically locating, extracting, and classifying contiguous pieces of strings, which represent entities of interest, in text. Classification (or typing) seeks to assign pre-defined categories (e.g., person, organization, location, expressions of time, monetary values, and emails) to each extracted piece of text. NER is a subtask of the broader problem of Information Extraction (IE) from text (Chang et al., 2006; Etzioni et al., 2005; Finkel and Manning, 2009; Nadeau and Sekine, 2007; Shen et al., 2015). Named entities usually refer to entity names that describe unique identifiers of people, locations, movies, events, and organizations. There is a large class of entities that are not “named," such as expressions of time, emails, and course identifiers. Their main characteristic is that they often follow an underlying syntactical pattern, which can be fully described or well approximated by Regular Expressions (REs).

Despite being the workhorse of many entity recognition tasks, REs have a number of drawbacks. The construction of highly accurate REs is difficult and requires specific technical skills. For a simple task such as recognizing emails, there are 361 REs proposed in RegExLib.com. Moreover, REs are brittle and difficult to maintain. These obstacles have motivated the work on automatic inference of REs (Banko et al., 2007; Li et al., 2008; Bartoli et al., 2018) where the objective it to develop approaches that are fast and deployable in real time. However, the existing approaches tend to require large number of examples to cover both the alphabet and the possible syntactic patterns. Moreover, they often produce overly complicated or long REs or combinations of REs (Bartoli et al., 2016). One of the most complete REs for emails has nearly 6,500 characters (Millner, 2008)!

Web documents are an important domain for data extraction. Importantly, the Web is not a place where data follows "underlying syntactical pattern" at scale. A datetime RE for New York Times news articles may not work for the articles at Le Figaro or Al Jazeera. Small typos throw off REs and produce nothing. This is a concrete Web example `data-timestamp="Thu Oct 05 2017 10:33:05` **`-0500"`**`>` that contains an unexpected space before "`-0500`". This small typo

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1991–2000
Brussels, Belgium, October 31 - November 4, 2018.

can easily deem a complex and painstakingly constructed datetime RE obsolete. At such scale, any attempt to fully understand an RE and debug it in case it fails is futile. New automated or semi-automated tools are needed to either supersede REs or to work in tandem with REs. In this work, we focus on the later.

In this paper, we target the problem of detecting presence of entity mentions that follow or closely resemble patterns that can be described by REs. Unlike much of the previous body of work on this topic, we do not focus on learning/inferring highly accuracte REs for entity identification (Prasse et al., 2012; Bui and Zeng-Treitler, 2014; Li et al., 2008; Banko et al., 2007; Brauer et al., 2011; Bartoli et al., 2016). We aim instead to show that deep learning can leverage imperfect REs and achieve very high accuracy while requiring only a modest human involvement.

Suppose the goal is to recognize *datetime* string expressions. We use some reasonable REs R for datetime to generate a weakly labeled training dataset from a large corpus of Web documents, e.g., news articles. We train a deep neural network on this data. Denote this model $\boldsymbol{M_{RE}}$. To our surprise, $\boldsymbol{M_{RE}}$ is already capable of recognizing the presence of datetime expressions beyond those recognized by R. Furthermore, with the addition of a very small number of training samples (between 20 - 50 instances) from a human labeler, we obtain a model $\boldsymbol{M_{RE+human}}$ that is superior to $\boldsymbol{M_{RE}}$ by a significant margin. In general, complex systems do not generalize easily with the addition of new data, because the amount of labeled data required to provide a good coverage grows exponentially with the complexity of the problem (Chiu and Nichols, 2015; Lample et al., 2016; Huang et al., 2015; Mahajan et al., 2018). We show that there is an opportunity for faster convergence to a generalized recognizer for this class of entities.

The main contributions in this paper are:

- We show how starting from REs R that recognizes a fraction of entities of a given type $\mathcal{E}$ (say, email) we can pretrain a deep neural network (NN) model which can be a richer recognizer of entities of type $\mathcal{E}$ than R.
- We show that we can fine tune the pretrained model to recognize an even larger set of entities of type $\mathcal{E}$ with the addition of a small number of labeled instances, as small as 20.

The paper is organized as follows. Section 2 gives an overview of the related work. Section 3 describes our method. Section 4 presents our methodology for parameter learning and experimental setup. Section 5 gives the experimental results. Finally, Section 6 concludes the paper.

2 Related Work

This section will mention several lines of research we deem the most related to our work.

The problem of inducing regular expressions has been an active area of work for more than two decades. One line of work focuses on improving the initial REs by identifying the true or false matches (Li et al., 2008; Murthy et al., 2012; Cetinkaya, 2007; Cochran et al., 2015). Another line of work attempts to directly induce REs from positive and negative sample strings (Fernau, 2009; Denis, 2001). The common approaches include generation of prefix and suffix automatons that represent overlapping syntactical features of the entities on character and token level (Brauer et al., 2011) and the automatic creation of REs based on genetic programming (Bartoli et al., 2012, 2014, 2016, 2018).

Constraining NN training to comply with known rules has also been an active research topic. Hu et al. (2016) proposes integration of constraints coming in the form of first order logic rules during training of NNs. Alashkar et al. (2017) trains an NN by minimizing a joint loss based on prediction of labels and adhering to the predefined rules. Locascio et al. (2016) proposed training LSTM NN to generate REs from sample pieces of text. Luo et al. (2018) incorporates knowledge of REs into training of NNs at three different levels: as the input features to NNs, as regularizations of the outputs of NN layers, or as a reward/penalty in the loss functions in NNs.

Unlike the aforementioned work, we do not attempt to learn explicit REs and do not force the outputs of NN layers match predetermined rules. Instead, we leverage REs as a means of generating a large quantity of weak labels from unlabeled data and using such data to pre-train an NN to recognize the provided REs. We fine tune such an NN on human-labeled data to exceed accuracy of the REs. A similar approach is effective in other domains. For example, Felbo et al. (2017) proposed pre-training an NN on millions of tweets labeled by emojis before fine tuning it for sentiment anal-

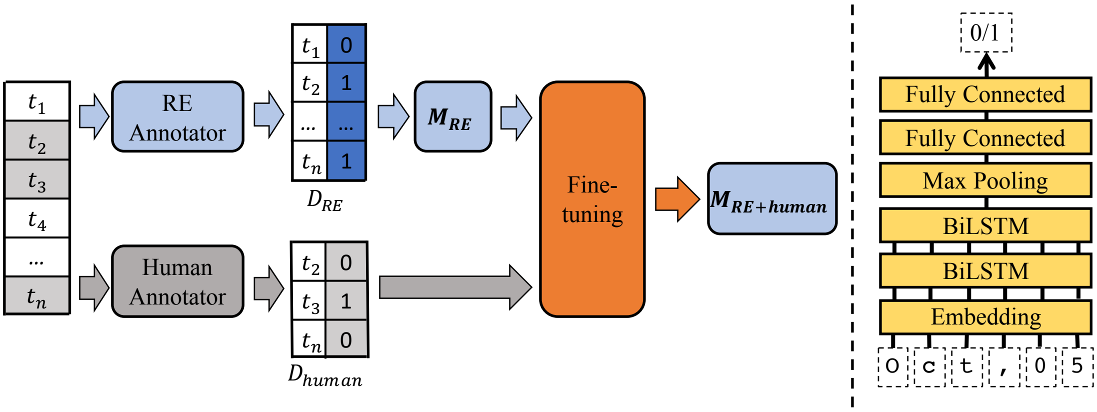

Figure 1: Overview of the solution (left) and deep learning architecture (right) used to train model $\boldsymbol{M_{RE}}$.

ysis. Mahajan et al. (2018) pre-trained an NN on billions of images labeled by hashtags before fine tuning it to various computer vision tasks.

3 Methodology

We describe the proposed framework here.

Problem Definition: Given a text string t and an entity type $\mathcal{E}$, the task is to predict whether t contains an entity mention of type $\mathcal{E}$. We treat this task as binary classification. To build the classifier, we assume that we are given a large corpus of unlabeled text strings $T = \{t_1, t_2, ..., t_n\}$. The challenge is to train the classifier with the minimal human effort. We allow a human expert to help in two ways: (1) construct a new RE or find an RE created by others, and (2) label an unlabeled string. For the purposes of this paper, we assume that one or more REs suitable for entity identification are already available and that human effort refers only to string labeling. The available REs might have an arbitrary precision and recall. We will analyze the impact of the RE quality on classification accuracy in the experimental section.

Solution Framework: An overview of the proposed framework is illustrated in Figure 1.

STEP 1. All of the unlabeled strings from T are fed into an RE annotator. If any of the provided REs t_i recognizes t_i, it is weakly labeled as $y_i = 1$, otherwise it is weakly labeled as $y_i = 0$.

STEP 2. An NN model $\boldsymbol{M_{RE}}$ is trained based on the weakly labeled data $D_{RE} = \{(t_i, y_i)|i = 1, 2, ..., n\}$. Given a large number of sample strings, it is expected that we can train an NN with high accuracy on D_{RE}.

STEP 3. A subset of m strings from T are sampled randomly and a human annotator labels each of the sampled strings. String t_i is labeled as $y_i = 1$ if the annotator recognizes an entity type $\mathcal{E}$ in t_i and as $y_i = 0$, if not. We denote the resulting strongly labeled data set as $D_{human} = \{(t_i, y_i)|i = 1, 2, ..., m\}$, where $m \ll n$.

STEP 4. The pre-trained NN $\boldsymbol{M_{RE}}$ is fine tuned with D_{human} data. We call the resulting NN $\boldsymbol{M_{RE+human}}$. For comparison, we also train a randomly initialized NN directly on D_{human}. We call this NN $\boldsymbol{M_{human}}$. The expectation is that the pre-trained NN captures very useful information about the entity type $\mathcal{E}$ and that fine tuning is more effective than training a new NN from scratch.

RE Annotator: Let us denote the set of REs available for a specific entity type $\mathcal{E}$ as R. R may be either created by human experts or generated automatically by tools like (Li et al., 2008; Bartoli et al., 2018), which require a human-labeled subset of T. Both approaches are human-intensive.

Deep Learning Architectures: We do not have a preference over any deep learning architecture to train $\boldsymbol{M_{RE}}$, as long as it can handle character-level inputs and produce binary outputs. We have no strict assumption about the strings in T. They may contain sentences from a formal news article, pieces of HTML code, or a mixture of formal texts and informal texts. For this reason, we treat t_i as a sequence of characters by default. NN architectures that meet our condition include but are not limited to: CNNs (Kim, 2014), BiLSTMs (Lai et al., 2015), and BiLSTM with self-attentions (Lin et al., 2017). For this paper we implemented a BiLSTM architecture. However, we also tested a CNN architecture, reaching similar conclusions. Our architecture contains an embedding layer to project each character into a vector, 2

Task	Data Size	Example	Labor
Date Time	761,002	data-timestamp="Thu **Oct 05 2017 10:33:05 -0500**"> <script src='/js/next-stories.20170925144113.js'>	2 days
Course Number	44,651	<body><h1>**CS 556** Interactive Software Systems /home.html>Dan R. Olsen Jr. Office: 3360 TM	1 week
Bill Date	49,002	(a) shall terminate on **30-09-2012.** ealth Service Act (42 U.S.C. 254c-15(c)(8)	-
Email Address	29,035	06:13:00 -0700 From:**phillip.allen@enron.com** To: Date:Mon, 23 Oct 2000 06:13:00 -0700 From: tom	1 week

Table 1: Dataset Summary.

BiLSTM layers to encode a sequence of character embeddings into a sequence of hidden vectors, and a max pooling layer followed by 2 fully-connected layers to project the hidden vectors into binary labels (Figure 1, on the right).

Fine-tuning: Once we have a model M_{RE} trained on weak labels, there are multiple ways to improve the weak model with human annotations to get $M_{RE+human}$. One common way is to freeze the parameters of all other layers of M_{RE} and fine-tune the last fully-connected layer (Donahue et al., 2014). Felbo et al. (2017) propose a *'chain-thaw'* strategy, which freezes all layers, then sequentially unfreezes and fine-tunes a single layer at a time. We exploit a less costly strategy as proposed in (Erhan et al., 2010), which uses the weights learned in M_{RE} to initialize $M_{RE+human}$, and start training $M_{RE+human}$ immediately with human annotations.

4 Experiment Design

We aim to answer three research questions in our experiments: **Q1**. Is it possible to train an accurate NN classifier with a limited number of human generated labels? **Q2**. What is the difference between REs and an NN pretrained on those REs? **Q3**. Does the quality of REs matter?

4.1 Data Sets

We use four datasets which are described in Table 1 in our experiments:

- Date Time: We download 6,000 news articles provided in the dataset One Week of Global News Feeds in Kaggle [1]. After chunking each news article, we get 761,002 strings.
- Course Number: The documents to produce chuncked strings are from The 4 Universities Data Set at CMU World Wide Knowledge Base (Web->KB) project[2]. This dataset has 44,651 strings.
- Bill Date: 600 US Congress bills from the THOMAS online database are used to identify the entity mentions of type datetime provided by (Bartoli et al., 2016). We generate 49,002 chuncked strings for this task. Each text instance contains the bill date (and time) and the location (index) of the datetime substring in the text.
- Email Address: The dataset is a collection of publicly available Enron email addresses from (Li et al., 2008; Brauer et al., 2011). It has 29,035 chuncked strings in total.

To answer the above 3 questions, we manually label 6,000 random strings from each data set. We create our training and testing data from this sample. The remaining strings are used to create D_{RE}. We list the human effort spent on labelling each data set in Table 1.

As an example of human annotations is the string `data-timestamp="Thu Oct 05 2017 10:33:05 -0500">Ex-dep`, which contains an entity of type datetime. The string `<scriptsrc='/js/prev-next-stories. 20170925144113.js'defer>` is an example of a negative instance. We give examples of positive and negative instances in Table 1. The presence of an entity mention of a desired type is highlighted in **bold** in positive instances.

Our human annotated data sets are attached as supplementary materials.

4.2 Regular Expression Generation

The number and the source of REs used to train the deep model M_{RE} are listed in Table 2.

For the recognition of Date/Time entity

---

[1] https://www.kaggle.com/therohk/global-news-week

[2] http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/

Task	$\|R\|$	Source	Labor
Date Time	25	Experts	5 hrs
Course Number	4	Li et al. (2008) Murthy et al. (2012)	-
Bill Date	5	Murthy et al. (2012)	-
Email Address	5	RE Library[3]	-

Table 2: Summary of REs.

mentions, we create 25 distinct REs in total (a member of our team who is familiar with REs generated the 25 REs, then the rest of the team checked these REs). For example, the RE `(20[0-9][0-9])(0[1-9]|1[0-2])(\d{2})([01][0-9]|2[0-3])([0-5][0-9])([0-5][0-9])` identifies datetime entities in the format `YYYYMMDDhhmmss`, while the RE `(20[0-9][0-9])-(0[1-9]|1[0-2])-(\d{2})T([01][0-9]|2[0-3]):([0-5][0-9]):([0-5][0-9])Z` identifies datetime entities in the format `YYYY-MM-DDThh:mm:ssZ`.

For the task of course number identification, we used four REs, one of which is borrowed from the results learned by ReLIE (Li et al., 2008), and the remaining three are from (Murthy et al., 2012). The regular expressions to extract the entity mentions of date are all from (Murthy et al., 2012). For email address, we use the top five REs from the RE Library[3] website.

4.3 Experimental Setup

Models in Comparison We compare 5 models on the 4 data sets: (1) ***Naive***, which always predicts 0, because 0 is the majority class on all 4 tasks. (2) ***RE***, which uses the set of REs R designed by experts or tools to weakly label the strings. Although we have multiple REs for each task, R can be any subset of the available REs. (3) M_{RE}, which is the pretrained model of weak labels generated by R in model (2). (4) M_{human}, which is the model trained with human annotations. (5) $M_{RE+human}$, which is the fine tuned model M_{RE}.

Evaluation Metrics For each data set, we divide the 6,000 strings with human annotations into 5 folds. We leave one fold as our test data. The training data is selected from the other 4 folds. We report four scores for each model: Accuracy (ACC), F1, Precision, and Recall. We report the average results over three random repetitions in Section 5.

Hyperparameters We use 100 dimensions in the embedding layer. We set the activation function in the first fully connected layers as `tanh`. The batch size is set to 300. We also add dropout layers after the embedding layer, the max pooling layer, and the first fully-connected layer to avoid overfitting, with drop out rate at 0.5. Our implementation is in PyTorch. We tune the learning rate (`lr`), the hidden units size (`nhidden`) in BiLSTM layers and the output size (`nfc`) of the first fully-connected layer by 5-fold cross validation using a random 6,000 sample from D_{RE}, for the sake of expediency. The ranges of selection are: `lr` $\in [0.0002, 0.0005, 0.001, 0.002, 0.004, 0.008, 0.015]$, `nhidden` $\in [50, 75, 100, 125, 150, 200]$ and `nfc` $\in [20, 50, 100, 200, 500]$. We use the *random search* algorithm proposed in (Bergstra and Bengio, 2012) that has been proved more effective than *grid search*. The hyperparameters used to train M_{human} and $M_{RE+human}$ are identical to those used to train M_{RE}.

We train 2 epochs for M_{RE} on weakly labeled data. M_{human} and $M_{RE+human}$ are trained for 50 epochs on strongly labeled data.

5 Experimental Results

In this section, we evaluate the proposed framework with extensive experiments on the 4 entity recognition tasks. We use the empirical results to understand how the quality of initial REs impacts our conclusions.

5.1 Entity Mention Detection with Limited Human Annotations

We report the comparisons of the 5 models in Table 3 when we only have 20 human annotations. We use all the available REs in each task in this experiment. Comparing the last two rows of each task, $M_{RE+human}$ always outperforms M_{human} by a large margin according to all 4 evaluation metrics on all 4 data sets. The F1, Precision and Recall scores are more than twice larger for all of tasks. The pretraining strategy is quite effective despite the very limited human annotation.

In addition, $M_{RE+human}$ is much better than ***RE*** in the datetime and Course Number tasks. Its Recall scores increase by 19.1% and 14.4%, respectively, in the two tasks. This means the human

---

[3] `http://www.regexlib.com/`

Model Name	Date Time (%)				Course Number (%)			
	ACC	F1	Precision	Recall	ACC	F1	Precision	Recall
Naive	77.34	0.00	0.00	0.00	68.47	0.00	0.00	0.00
RE	90.65	76.77	88.00	68.33	71.64	59.91	54.01	67.28
$\boldsymbol{M_{RE}}$	91.45	79.09	**89.01**	71.39	72.64	61.52	55.21	69.47
$\boldsymbol{M_{human}}$	74.78	40.75	37.25	46.38	62.81	30.74	37.63	28.45
$\boldsymbol{M_{RE+human}}$	**93.50**	**85.44**	87.03	**84.95**	**82.86**	**74.31**	**72.45**	**77.01**
	Bill Date (%)				Email Address (%)			
Naive	92.83	0.00	0.00	0.00	88.75	0.00	0.00	0.00
RE	**94.56**	38.23	**97.92**	24.01	**98.72**	**94.61**	**89.79**	**100.00**
$\boldsymbol{M_{RE}}$	94.53	38.10	96.30	23.96	98.64	94.28	89.19	**100.00**
$\boldsymbol{M_{human}}$	92.56	2.15	6.25	1.30	83.44	42.30	35.99	53.03
$\boldsymbol{M_{RE+human}}$	94.36	**39.95**	87.88	**27.59**	97.75	90.95	83.60	**100.00**

Table 3: The comparisons in the presence of very few human annotations: $|D_{human}| = 20$.

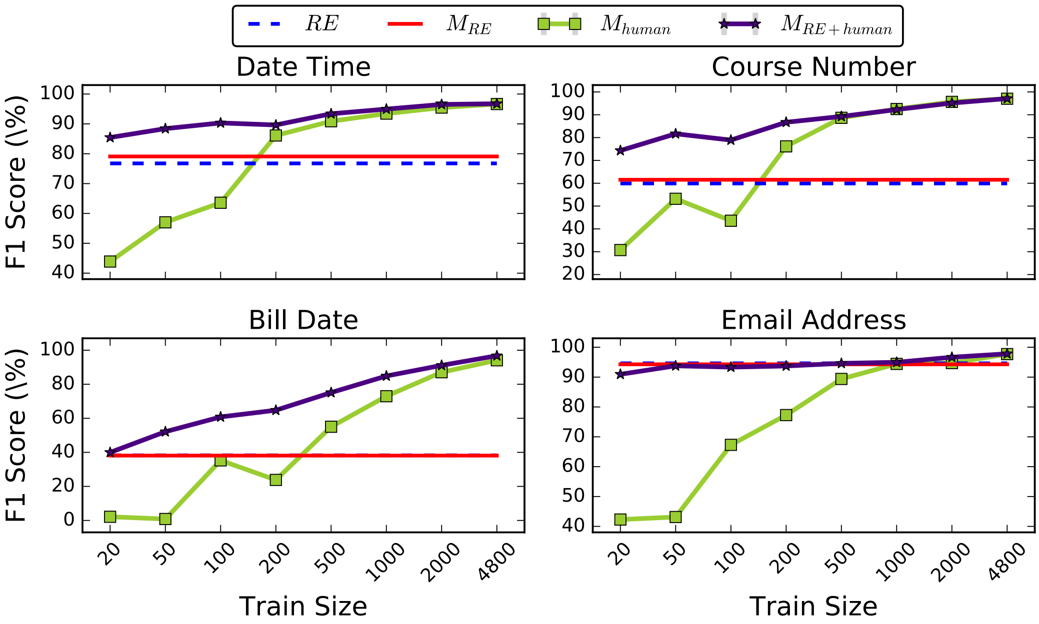

Figure 2: Trend of F1 when varying the number of human annotations.

annotations greatly increase the coverage of the initial REs for entity mentions. The Bill Date task is hard, since the initial REs already achieve Precision = 98% and entity mentions are really rare (ACC = 93% in ***Naive*** model). We still are able to improve the Recall by 15%, but at the expense of reduced Precision. This only gives a slight improvement in F1 (4.5%) and unchanged Accuracy. The Email Address task is even harder, with 90% Precision and 100% Recall from the initial REs. We fail to improve the accuracy with only 20 human annotations in this task.

To summarize, the answer to **Q1**, it is possible to train accurate NNs with limited amount of human generated labels and large amount of weak labels generated automatically.

5.2 Effect of the Number of Human Annotations

In Figure 2 we show how F1 varies with the size of the strongly labeled dataset, $|D_{human}| = [20, 50, 100, 200, 500, 1000, 2000, 4800]$. The observed trend is consistent over all 4 tasks: the larger the set of human annotations the better the performance of both $\boldsymbol{M_{human}}$ and $\boldsymbol{M_{RE+human}}$. For the first 2 tasks, $\boldsymbol{M_{RE+human}}$ surpasses all other competitors at 20; it takes 50 human annotations in Bill Date

	Date Time	
	Example	Labels
Group 1 19 / 11 / 8	"20170825" /><meta name="utime" content=**"20170828042824"**/>	1 / 1 / 0
	{"origin":"mw1273","timestamp":**"20171005170835"**,"ttl":1900800}	1 / 1 / 0
	08/29/2017 12:13 pm ET	1 / 1 / 0
	2017/08/29/1/1700000000AEN20170829007751315F.html	0 / 0 / 1
	11-hristo-dimitrov-2017-10-05-06-57-492-c37d5a31-4ade	0 / 0 / 1
Group 2 48 / 36 / 12	0","dateModified": **"Thu, 05 Oct 2017 17:26:30 +0000"**	1 / 0 / 1
	<meta property="published",content=**"2017-08-28T14:4316-0400"** />	1 / 0 / 1
	published **August 24, 2017 at 6:00 am**</time>	1 / 0 / 1
	http://www.businesswire.com/news/home/20170829005822/en/</p>	0 / 1 / 0
	resources/MWimages/MW-FV607-lava-2-MC-20171004095909.jpg">	0 / 1 / 0

Table 4: Examples of misclassified strings by $\boldsymbol{M_{RE}}$ and ***RE*** for Date Time task. The first columns shows number of misclassified strings, number of positives, and number of negaives. The last column represent human label and classifications by ***RE***, and $\boldsymbol{M_{RE}}$. Datetime is in bold for positive human annotation.

task, and $2,000$ in Email Address task.

5.3 Case Studies

From Table 3, we observe that $\boldsymbol{M_{RE}}$ achieves higher F1 scores than ***RE*** by about 2.3% on Date Time and 1.6% on Course Number tasks. This is a seemingly surprising outcome; NN trained on weak labels does better on human labeled test data than the REs used to generate the weak labels. To provide an insight, in Table 4 we illustrate example strings on which ***RE*** and $\boldsymbol{M_{RE}}$ disagree. *Group 1* consists of 19 strings where ***RE*** is correct and $\boldsymbol{M_{RE}}$ is not. *Group 2* consists of 48 strings where ***RE*** is incorrect and $\boldsymbol{M_{RE}}$ is correct. Looking at the examples in the first and last 2 rows of the table, we find strings that match RE with `YYYYMMDDhhmmss` format. This is an unusual form and it is expected that it might not always correspond to datetime entity. We hypothesize that the neural network encountered many negative strings in the weakly labeled data that have a similar form and learned that this form is not a reliable predictor of datetime.

Rows 6 - 8 in Table 4 illustrate the resilience of $\boldsymbol{M_{RE}}$ to small variations in the original REs. For example, despite *an extra space* in row 6, *a missing colon* in row 7, and *a mixture of spoken language* in row 8, $\boldsymbol{M_{RE}}$ is able to detect those entities, but REs are not. We give cases where $\boldsymbol{M_{RE}}$ makes mistakes in rows 3 - 5, showing that the NN is not able to learn the underlying REs with 100% accuracy.

To summarize, the answer to **Q2**, it appears that NNs are more noise resilient than REs.

5.4 Impact of the Initial REs

In this subsection, we investigate the impact of the choice of REs R on the accuracy of NN models. Since ***Naive*** and $\boldsymbol{M_{human}}$ models are not affected by R, we compare only three models in this subsection: ***RE***, $\boldsymbol{M_{RE}}$ and $\boldsymbol{M_{RE+human}}$. We select 4 out of the 25 REs in the datetime task for this study. The 4 REs are of different quality. We list the 4 REs in Figure 3. An example pattern that RE1 matches is `20180503101212`, for RE2 it is `2018-05-03 10:12:12`, for RE3 it is `2018-05-03T10:12:12Z` in UTC time zone and for RE4 it is `2018-05-03 10:12:12+00:00` or `2018-05-03 10:12:12-00:00`. We also consider the quality of NNs trained on weak labels from the whole set of 25 REs, denoted as All.

In Table 5 we compare the performance of ***RE*** and $\boldsymbol{M_{RE}}$ for the 5 different selections of REs for weak labeling. It can be observed that RE1 is the weakest individual RE in the group with F1 = 4.83, while RE4 is the strongest with F1= 41.86. Using all 25 REs gives the highest accuracy of F1= 76.77. We can see that $\boldsymbol{M_{RE}}$ closely follows the performance of ***RE*** and it is interesting to observe that $\boldsymbol{M_{RE}}$ becomes visibly superior only with good REs.

In Figure 3, we plot the 4 accuracy metrics for model $\boldsymbol{M_{RE+human}}$, which was pretrained on weakly labeled data generated by 5 different choices of REs, with varying initial RE sets and sizes of human annotations. We observe a significant influence of REs on accuracy. We can also observe that as the number of strong labels grows, the impact of RE choice decreases. When the number

Initial Set	***RE***				M_{RE}			
	ACC	F1	Precision	Recall	ACC	F1	Precision	Recall
RE1	75.88	4.83	24.89	2.68	75.71	4.20	23.68	2.32
RE2	79.97	20.72	100.00	11.58	80.00	20.92	100.00	11.70
RE3	79.78	19.34	100.00	10.75	79.83	19.75	100.00	10.99
RE4	83.40	41.86	100.00	26.71	83.73	43.77	100.00	28.19
All	90.65	76.77	88.00	68.33	91.45	79.09	89.01	71.39

Table 5: Comparison between ***RE*** and M_{RE} model for 5 different sets of REs.

```
RE1: (20[0-9][0-9])(0[1-9]|1[0-2])(\d\{2})([01][0-9]|2[0-3])([0-5][0-9])([0-5][0-9])
RE2: (20[0-9][0-9])-(0[1-9]|1[0-2])-(\d{2})T([01][0-9]|2[0-3]):([0-5][0-9]):([0-5][0-9])([+-]([0-9]|0[0-9]|1[0-2]):{0,1}(00|30))
RE3: (20[0-9][0-9])-(0[1-9]|1[0-2])-(\d{2})T([01][0-9]|2[0-3]):([0-5][0-9]):([0-5][0-9])Z
RE4: (20[0-9][0-9])-(0[1-9]|1[0-2])-(\d{2})T([01][0-9]|2[0-3]):([0-5][0-9]):([0-5][0-9])\.([0-9]{1,10})Z
```

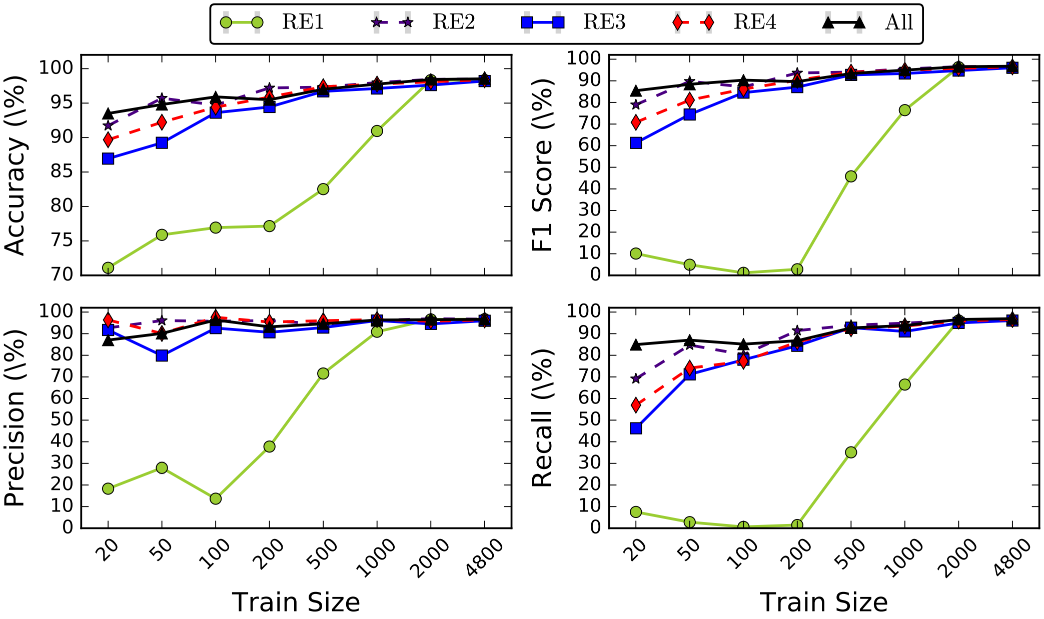

Figure 3: Performance of $M_{RE+human}$ for different initial sets of REs and sizes of D_{human}.

of strong labels exceeds $1,000$, the impact of the RE choice becomes negligible.

In summary, to answer **Q3**, there is a trade-off between creating more REs and creating more strong labels: (1) If designing a comprehensive RE takes a lot of time, a good strategy may be to take some time to construct one moderately good RE and spend more time on data labeling. (2) If the pattern is easy to describe by an RE, it may be a good strategy to spend time on creating a better set of REs and spend less time on labeling.

6 Conclusions

The main premise of this work is that it is practically impossible to create REs capable of identifying entities with perfect precision and recall at web scale. This paper explores ways to combine the expressive power of REs, ability of deep learning, and human-in-the loop into a novel integrated framework for entity recognition in web data. The framework starts by creating or collecting the existing REs for a particular type of an entity type (e.g., emails). Those REs are then used over a large document corpus to collect weak labels for the entity mentions and an NN is trained to predict those RE-generated weak labels. Finally, a human expert is asked to label a small set of documents and the neural network is fine tuned on those documents. The experimental evaluation on several entity identification problems shows that the proposed framework achieves impressive accuracy, while requiring very modest human effort.

Web sources often change in ways that prevent the induced REs from extracting data correctly. At the web scale, we require automated tools to maintain them. One direction of future work is to use our framework to diagnose when a RE is broken over a text stream.

Acknowledgments

This work was supported in part by the following grants: U.S. NSF BigData 1546480 and U.S. NIH R21CA202130.

References

Taleb Alashkar, Songyao Jiang, Shuyang Wang, and Yun Fu. 2017. Examples-rules guided deep neural network for makeup recommendation. In *AAAI*, pages 941–947.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

Alberto Bartoli, Giorgio Davanzo, Andrea De Lorenzo, Marco Mauri, Eric Medvet, and Enrico Sorio. 2012. Automatic generation of regular expressions from examples with genetic programming. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 1477–1478.

Alberto Bartoli, Giorgio Davanzo, Andrea De Lorenzo, Eric Medvet, and Enrico Sorio. 2014. Automatic synthesis of regular expressions from examples. *Computer*, 47(12):72–80.

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. 2016. Inference of regular expressions for text extraction from examples. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1217–1230.

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. 2018. Active learning of regular expressions for entity extraction. *IEEE transactions on cybernetics*, 48(3):1067–1080.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Falk Brauer, Robert Rieger, Adrian Mocan, and Wojciech M Barczynski. 2011. Enabling information extraction by inference of regular expressions from sample entities. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1285–1294. ACM.

Duy Duc An Bui and Qing Zeng-Treitler. 2014. Learning regular expressions for clinical text classification. *Journal of the American Medical Informatics Association*, 21(5):850–857.

Ahmet Cetinkaya. 2007. Regular expression generation through grammatical evolution. In *Proceedings of the 9th annual conference companion on Genetic and evolutionary computation*, pages 2643–2646. ACM.

Chia-Hui Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. 2006. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Robert A Cochran, Loris D'Antoni, Benjamin Livshits, David Molnar, and Margus Veanes. 2015. Program boosting: Program synthesis via crowd-sourcing. In *ACM SIGPLAN Notices*, volume 50, pages 677–688. ACM.

François Denis. 2001. Learning regular languages from simple positive examples. *Machine Learning*, 44(1-2):37–66.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

Henning Fernau. 2009. Algorithms for learning regular expressions from positive data. *Information and Computation*, 207(4):521–541.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *EMNLP*, pages 141–150. Association for Computational Linguistics.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and HV Jagadish. 2008. Regular expression learning for information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 21–30. Association for Computational Linguistics.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. *arXiv preprint arXiv:1608.03000*.

Bingfeng Luo, Yansong Feng, Zheng Wang, Songfang Huang, Rui Yan, and Dongyan Zhao. 2018. Marrying up regular expressions with neural networks: A case study for spoken language understanding. *arXiv preprint arXiv:1805.05588*.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the limits of weakly supervised pretraining. *arXiv preprint arXiv:1805.00932*.

Rachel Millner. 2008. Four regular expressions to check email addresses.

Karin Murthy, P Deepak, and Prasad M Deshpande. 2012. Improving recall of regular expressions for information extraction. In *International Conference on Web Information Systems Engineering*, pages 455–467. Springer.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Paul Prasse, Christoph Sawade, Niels Landwehr, and Tobias Scheffer. 2012. Learning to identify regular expressions that describe email campaigns. In *Int. Conf. Mach. Learn.*, pages 441–448.

W. Shen, J. Wang, and J. Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding

Shib Sankar Dasgupta
Indian Institute of Science
Bangalore, India
`s.s.dasgupta.iisc@gmail.com`

Swayambhu Nath Ray
Indian Institute of Science
Bangalore, India
`swayambhunath93@gmail.com`

Partha Talukdar
Indian Institute of Science
Bangalore, India
`ppt@iisc.ac.in`

Abstract

Knowledge Graph (KG) embedding has emerged as an active area of research resulting in the development of several KG embedding methods. Relational facts in KG often show temporal dynamics, e.g., the fact *(Cristiano_Ronaldo, playsFor, Manchester_United)* is valid only from 2003 to 2009. Most of the existing KG embedding methods ignore this temporal dimension while learning embeddings of the KG elements. In this paper, we propose HyTE, a temporally aware KG embedding method which explicitly incorporates time in the entity-relation space by associating each timestamp with a corresponding hyperplane. HyTE not only performs KG inference using temporal guidance, but also predicts temporal scopes for relational facts with missing time annotations. Through extensive experimentation on temporal datasets extracted from real-world KGs, we demonstrate the effectiveness of our model over both traditional as well as temporal KG embedding methods.

1 Introduction

Knowledge Graphs (KGs) are large multi-relational graphs where nodes correspond to entities, and typed edges represent relationships among them. KGs encode factual beliefs in the form of triple (entity, relation, entity), e.g., *(Brussels, isCapitalOf, Belgium)*. Examples of a few KGs include NELL (Mitchell et al., 2018), YAGO (Suchanek et al., 2007), and Freebase (Bollacker et al., 2008). KGs have been found to be useful for a variety of tasks, viz., Information Retrieval (Kotov and Zhai, 2012; Xiong and Callan, 2015), Question Answering (Dong et al., 2015; Bordes et al., 2015; Yao and Durme, 2014), among others.

KG embedding has emerged as a very active area of research over the last few years, resulting in the development of several techniques (Bordes et al., 2013; Nickel et al., 2016b; Yang et al., 2014; Lin et al., 2015; Trouillon et al., 2016; Dettmers et al., 2018; Guo et al., 2018). These methods learn high-dimensional vectorial representations for nodes and relations in the KG, while preserving various graph and knowledge constraints.

We note that KG beliefs are not universally true, as they tend to be valid only in a specific time period. For example, *(Bill Clinton, presidentOf, USA)* was true only from 1993 to 2001. KG beliefs with such temporal validity marked are called as temporally scoped. These temporal scopes are increasingly available on several large KGs, e.g., YAGO (Suchanek et al., 2007), Wikidata (Erxleben et al., 2014). The mainstream KG embedding methods ignore the availability or importance of such temporal scopes while learning embeddings of nodes and relations in the KGs. These methods treat the KG as a static graph with the assumption that the beliefs contained in them are universally true. This is clearly inadequate and it is quite conceivable that incorporating temporal scopes during representation learning is likely to yield better KG embeddings. In spite of its importance, temporally aware KG embeddings is a relatively unexplored area. Recently, a KG embedding method which utilizes temporal scopes was proposed in (Jiang et al., 2016). However, instead of directly incorporating time in the learned embeddings, the method proposed in (Jiang et al., 2016) first learns temporal order among relations (e.g., *wasBorIn* → *wonPrize* → *diedIn*). These relation orders are then incorporated as constraints during the KG embedding stage. Thus, the embedding learned by (Jiang et al., 2016) is not explicitly temporally aware.

In order to overcome this challenge, in this paper, we propose *Hyperplane-based Temporally aware KG Embedding (HyTE)*, a novel KG em-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2001–2011
Brussels, Belgium, October 31 - November 4, 2018.

bedding technique which directly incorporates temporal information in the learned embeddings. HyTE fragments a temporally-scoped input KG into multiple static subgraphs with each subgraph corresponding to a timestamp. HyTE then projects the entities and the relations of each subgraph onto timestamp specific hyperplanes. We learn the hyperplane (normal) vectors and the representation of the KG elements distributed over time jointly. Our contributions are as follows.

- We draw attention to the important but relatively unexplored problem of temporally aware Knowledge Graph (KG) embedding. In particular, we propose HyTE, a temporally aware method for learning Knowledge Graph (KG) embedding.
- In contrast to previous time-sensitive KG embedding methods, HyTE encodes temporal information directly in the learned embeddings. This enables us to predict temporal scopes for previously unscoped KG beliefs.
- Through extensive experiments on multiple real-world datasets, we demonstrate HyTE's effectiveness.

We have made HyTE's source code and datasets used in the paper available at https://github.com/malllabiisc/HyTE

2 Related Work

Temporal fact and event extraction: Time, apart from being an information, also introduces a separate dimension to knowledge. Thus temporal scoping of relational facts is an imperative part of automatic knowledge graph construction and completion. T-YAGO (Wang et al., 2010) extracts temporal facts from semi-structured data like Wikipedia Infoboxes, and categories using only regular expressions. On the other hand, systems like PRAVDA harvests temporal information from free text sources using label propagation. CoTS (Talukdar et al., 2012b) uses integer linear program based approach to model temporal constraints and proposes joint inference framework with few seed examples.

A method for discovering temporal ordering among factual relations was proposed in (Talukdar et al., 2012a). The task of extracting temporally rich events and time expressions and ordering between them is introduced in TempEval challenge (UzZaman et al., 2013; Verhagen et al., 2010). Various approaches (McDowell et al., 2017; Mirza and Tonelli, 2016) made for solving the task proved to be effective in other temporal reasoning tasks. Although we try to attend to a similar problem, the method proposed in this paper is more related to relational embedding learning paradigm than scoping temporal facts from the web.

Relational embedding learning methods: An enormous amount of research has been done in this field, especially for KG completion or link prediction task (Bordes et al., 2013). (Nickel et al., 2016a) provides a detailed review of the recent KG embedding learning methods. These can be broadly categorized into two different paradigms. TransE(Bordes et al., 2013), TransH(Wang et al., 2014), TransR (Lin et al., 2015), TransD (Ji et al., 2015) are the translational distance-based models. Here the main theme is to minimize the distance between two entity vectors where one of them is translated by a relation vector. The realm of matrix factorization based methods includes bilinear model RESCAL (Nickel et al., 2011), DistMult (Yang et al., 2014), HolE (Nickel et al., 2016b). Some of the other notable models are Neural Tensor Networks(NTN) (Socher et al., 2013). We also provide some background on the traditional methods in section 3. However, the temporal dimension remains silent in all of these inference methods.

Link prediction through embeddings of the graph nodes and edges, are not only useful for inference over KG but also important for predicting incomplete pieces of the KG itself. Learning temporally steered embeddings is an important but proportionately less explored problem. Only some handful of methods have been proposed for this purpose. t-TransE (Jiang et al., 2016) learns time aware embedding by learning relation ordering jointly with TransE. They try to inflict temporal order on time-sensitive relations e.g. $wasBornIn \rightarrow wonPrize \rightarrow diedIn$. t-TransE does not use the time information directly, whereas we incorporate time directly in our learning algorithm. Another approach Know-Evolve (Trivedi et al., 2017) models the non-linear temporal evolution of KG elements using bilinear embedding learning method. They deploy recurrent neural network to capture non-linear dynamical characteristics of the embeddings. However, they restrict their domain to event-based interac-

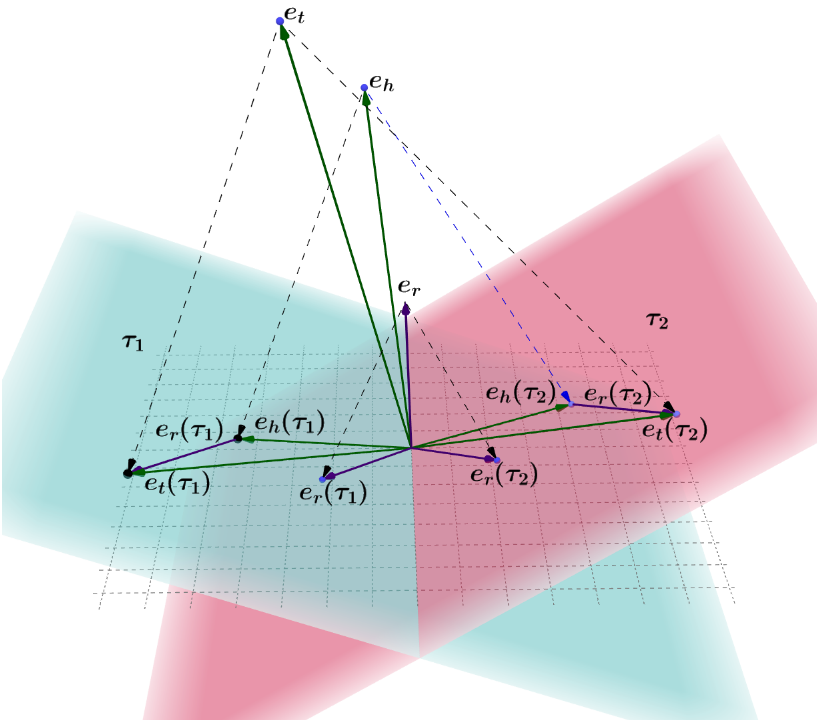

Figure 1: In the figure, the vectors e_h, e_r and e_t correspond to the triple (h, r, t) that is valid at time τ_1 and τ_2. $e_h(\tau_1), e_r(\tau_1)$ and $e_t(\tau_1)$ are the projections of this triple on the hyperplane corresponding to time τ_1 (similarly for time τ_2). Our method HyTE minimizes the translational distance, $\sum_i \|e_t(\tau_i) + e_r(\tau_i) - e_t(\tau_i)\|_1$, in order to learn the temporally aware representations of entities and relations in this triple.

tion type of datasets which are fairly dense in nature. Leblay and Chekol (2018) propose a method for temporal embedding learning using side information from the atemporal part of the graph. However, we use purely temporal KG to learn the temporally aware embedding.

3 Background: KG Embedding

In this section, we provide an overview of the existing methods for knowledge graph representation learning (Bordes et al., 2013), (Wang et al., 2014). Consider a KG $\mathcal{G}$ with a set of entities $\mathcal{E}$. The set of directed edges, $\mathcal{D}^+$ consists of triples (h, r, t), where the edge direction is from h to t and the edge label (also popularly known as relation) is r.

3.1 TransE and TransH

TransE (Bordes et al., 2013) is a simple and efficient translational distance model. It interprets the relation as a translation vector between head and tail entity vectors. Given two entity vectors e_h, e_t $\in \mathbb{R}^n$, it tries to map the relation as a translation vector $e_r \in \mathbb{R}^n$, i.e., $e_h + e_r \approx e_t$ for observed triple (h, r, t). So the distance based scoring function used for plausible triples is hereby,

$$f(h, r, t) = \|e_h + e_r - e_t\|_{l_1/l_2},$$

where, $\|\cdot\|_{l_1/l_2}$ is the l_1 or l_2-norm of the difference vector. $f(h, r, t)$ will be minimized for observed or correct triples. In order to differentiate between correct and incorrect triples, their TransE score difference is minimized using margin based pairwise ranking loss. More formally, we optimize

$$\sum_{x \in \mathcal{D}^+} \sum_{y \in \mathcal{D}^-} \max(0, f(x) - f(y) + \gamma),$$

with respect to the entity and relation vectors. γ is a margin separating correct and incorrect triples. $\mathcal{D}^+$ is the set of all positive triples, i.e., observed triples in KG. The negative samples are drawn randomly from the set -

$$\mathcal{D}^- = \{(h', r, t) | h' \in \mathcal{E}, (h', r, t) \notin \mathcal{D}^+\}$$
$$\cup \{(h, r, t') | t' \in \mathcal{E}, (h, r, t') \notin \mathcal{D}^+\}.$$

TransE fails to model the many-to-one, one-to-many, many-to-many type of relations as it does not learn a distributed representation of entities when it is involved with many relations. In order to tackle these situations, TransH was proposed. TransH (Wang et al., 2014) models a relation r as a vector on a relation specific hyperplane and project entities associated with it on that particular hyperplane in order to learn distributed representation of the entities.

We notice that not only the role of the entities changes with time, but also the relationship between them changes. In this paper, we intend to capture this temporal behavior of the entities and relations and try to learn their embeddings accordingly. As discussed above, TransH (Wang et al., 2014) uses relation specific hyperplanes in order to prevent an entity from exhibiting identical characteristic when it is involved with different relations. Taking inspiration from the objective of TransH, we propose a hyperplane based method for learning KG representation distributed in time.

4 Proposed Method: HyTE

In this section, we present a detailed description of HyTE (Figure 1) which not only exploits the relational properties among entities but also uses the temporal meta-data associated with them.

4.1 Temporal Knowledge Graph

Usually knowledge graphs are treated as a static graph consisting of triples in form of (h, r, t). Adding a separate time dimension to the triple makes the KG dynamic. Consider the quadruple $(h, r, t, [\tau_s, \tau_e])$, where τ_s and τ_e denote the start and end time during which the triple (h, r, t) is valid. Unlike (Jiang et al., 2016), we incorporate this time meta-fact directly into our learning algorithm to learn temporal embeddings of the KG elements. Given the timestamps, the graph can be dismantled into several static graphs consisting of triples that are valid in the respective time steps, e.g., knowledge graph G can be expressed as $\mathbb{G} = \mathbb{G}_{\tau_1} \cup \mathbb{G}_{\tau_2} \cup \cdots \cup \mathbb{G}_{\tau_T}$, where τ_i, $i \in 1, 2, \cdots, T$ are the discrete time points.

We constructed this temporal component-graphs ($\mathbb{G}_\tau$) from the quadruples by considering (h, r, t) to be a positive triple at each time point between τ_s and τ_e. Now, given a quadruple $(h, r, t, [\tau_s, \tau_e])$, we consider it to be a positive triple for each time point between τ_s and τ_e. So, we include (h, r, t) in each $\mathbb{G}_\tau$, where $\tau_s \leq \tau \leq \tau_e$. The set of positive triple corresponding to time τ is denoted as $\mathcal{D}^+_\tau$.

4.2 Projected-Time Translation

TransE considers entity and relation vectors in the same semantic space for a static graph. We observe that time is the main source of different many-to-one, one-to-many or many-to-many relations, e.g., (h, r) pair can be associated with different tail entity t at different points of time. Thus traditional methods fail to disambiguate them directly. In our time guided model, we want the entity to have a distributed representation associated with different time points.

We represent time as a hyperplane i.e., for T number of time steps in the KG, we will have T different hyperplanes represented by normal vectors $w_{t_1}, w_{t_2}, \cdots, w_{t_T}$. Thus, we try to segregate the space into different time zones with the help of the hyperplanes. Now, triples valid at time τ (i.e., the sub graph $\mathbb{G}_\tau$) are projected onto time specific hyperplane w_τ, where their translational distance (TransE Section 3 our case) is minimized. To illustrate, in Figure 1, the triple (h, r, t) is valid for both time frame τ_1 and τ_2. Hence they are projected on hyperplanes corresponding to those times.

Now we compute the projected representation on w_τ as,

$$P_\tau(e_h) = e_h - (w_\tau^\top e_h) w_\tau,$$

$$P_\tau(e_t) = e_t - (w_\tau^\top e_t) w_\tau,$$

$$P_\tau(e_r) = e_r - (w_\tau^\top e_r) w_\tau,$$

where we restrict $\|w_\tau\|_2 = 1$.

We expect that a positive triple, valid at time τ, will have the mapping as $P_\tau(e_h) + P_\tau(e_r) \approx P_\tau(e_t)$. Thus, we use the following scoring function.

$$f_\tau(h, r, t) = \|P_\tau(e_h) + P_\tau(e_r) - P_\tau(e_t)\|_{l_1/l_2}.$$

We learn $\{w_\tau\}_{\tau=1}^T$ for each time stamp τ, along with the entity and relation embeddings. So, by projecting the triple into its time hyperplane, we incorporate temporal knowledge into the relation and entity embeddings, i.e., the same distributed representation will have a different role in different points in time.

Optimization : As mentioned in section 3.1 , we minimize the margin-based ranking loss.

$$\mathcal{L} = \sum_{\tau \in [T]} \sum_{x \in \mathcal{D}^+_\tau} \sum_{y \in \mathcal{D}^-_\tau} \max(0, f_\tau(x) - f_\tau(y) + \gamma),$$

where, $\mathcal{D}_{\tau}^{+}$ is the set of valid triples with timestamp τ. The negative samples are drawn from the set of all negative samples, $\mathcal{D}_{\tau}^{-}$. We explored two different types of negative sampling:

- **Time agnostic negative sampling(TANS)** considers the set of all the triples that does not belong to the KG, irrespective of timestamps. More formally, for time step τ the negative samples are drawn from the set,

$$\mathcal{D}_{\tau}^{-} = \{(h^{'}, r, t, \tau) | h^{'} \in \mathcal{E}, (h^{'}, r, t) \notin \mathcal{D}^{+} \} \cup \{(h, r, t^{'}, \tau) | \, t^{'} \in \mathcal{E}, (h, r, t^{'}) \notin \mathcal{D}^{+} \}. \quad (1)$$

- **Time dependent negative sampling(TDNS)** emphasizes on time. Along with the time agnostic negative sample, we add extra negative samples which are present in the KG but do not exist in the subgraph for a particular time stamp. Thus we draw negative samples from the set,

$$\mathcal{D}_{\tau}^{-} = \{(h^{'}, r, t, \tau) | h^{'} \in \mathcal{E}, (h^{'}, r, t) \in \mathcal{D}^{+}, (h^{'}, r, t, \tau) \notin \mathcal{D}_{\tau}^{+} \} \cup \{(h, r, t^{'}, \tau) | \, t^{'} \in \mathcal{E}, (h, r, t^{'}) \in \mathcal{D}^{+}, (h, r, t^{'}, \tau) \notin \mathcal{D}_{\tau}^{+} \}. \quad (2)$$

The above mentioned loss $\mathcal{L}$ is minimized subjected to the constraints.

$$\|e_p\|_2 \leq 1, \forall \, p \in \mathcal{E}, \quad \|w_\tau\|_2 = 1, \forall \, \tau \in [T]$$

We enforce the first one by adding l_2- regularization of entity vectors with $\mathcal{L}$. We take care of the second constraint by normalizing the time embeddings viz., the hyperplane normal vectors after each update of stochastic gradient descent.

We perform link prediction as well as temporal scoping in order to show the effectiveness of HyTE. For link prediction, we train the model using the optimization procedure as described with time agnostic negative sampling (TANS, Equation 1). The temporal scoping task (Section 5.5) requires the time hyperplanes to be well structured in the embedding space. The time-dependent negative sampling (TDNS Equation 2) is more suitable in case of the temporal scoping problem.

Datasets	#Entity	#Relations	Train/Valid/Test
Wikidata12K	12,554	24	32.5k/4k/4k
YAGO11K	10,623	10	16.4k/2k/2k

Table 1: Details of datasets used. Please see Section 5.1 for details.

5 Experiments

We evaluate our model and compare with different state-of-the-art baselines based on Link prediction(Section 5.3,5.4) and Temporal scoping (Section 5.5). Evaluation metrics used are same as that of the traditional KG embedding method (Bordes et al., 2013) for link prediction task. For temporal scoping task, we present an evaluation criteria as none of the baselines are applicable for this task.

5.1 Datasets

Knowledge Graphs such as Wikidata (Erxleben et al., 2014) and YAGO (Suchanek et al., 2007) have time annotations on a subset of the facts. We extracted the temporally rich subgraph from them for testing our algorithm as well as the baselines.

YAGO11k: In the YAGO3 knowledge graph (Mahdisoltani et al., 2013), some temporally associated facts have meta-facts as *(#factID, occurSince, t_s), (#factID, occurUntil, t_e)*. The total number of time annotated facts containing both occursSince and occursUntil are 722,494. Out of them, we selected top 10 most frequent temporally rich relations. In order to handle sparsity, we recursively remove edges containing entity with only a single mention in the subgraph. This ensures a healthy connectivity within the graph. Finally, we obtain a purely temporal graph of 20.5k triples and 10,623 entities by following this procedure.

Wikidata12k: We extracted this temporal Knowledge Graph from a preprocessed dataset of Wikidata proposed by (Leblay and Chekol, 2018)[1]. We followed a similar procedure as described in YAGO11k. Here also, we distill out the subgraph with time mentions for both start and end. We ensure that no entity has only a single edge connected to it. We select top 24 frequent temporally rich relations for this case, which resulted in 40k triples with 12.5k entities. The dataset is almost double in size with respect to YAGO11k.

---

[1] https://staff.aist.go.jp/julien.leblay/datasets/

5.2 Method Compared

For evaluating the performance of our algorithm, we compare against the following methods:

- **t-TransE** (Jiang et al., 2016): This method uses a temporal ordering of relations to model knowledge evolution in the temporal dimension. They regularize the traditional embedding score function with observed relation ordering with respect to head entities.
- **HolE** (Nickel et al., 2016b): We consider this method as a representative of the state-of-the-art in non-temporal KG representation learning.
- **TransE** (Bordes et al., 2013): This is a simple but effective translation based model. We build HyTE on top of TransE and demonstrate the gains over this method.
- **TransH** (Wang et al., 2014): This method models each relation as different hyperplanes on which the translation operations are carried out. Our proposed method, HyTE also modifies TransE in a similar fashion by treating the timestamps as hyperplanes.
- **HyTE** : Our proposed method. Please see Section 4 for more details.

5.3 Entity Prediction

The task here is to predict the missing entity, given an incomplete relational fact with its time. We experimented with both YAGO11K and Wikidata12k dataset. Training is done in perspective of both head and tail prediction. More formally, for generation of negative sample from a correct triple (h, r, t, τ), we split them in two parts - $(h, r, ?, \tau)$ (for tail entity prediction) and $(?, r, t, \tau)$ (for head entity prediction). In this task, we follow the TANS (Equation 1) procedure for generating negative samples, i.e., for each of tail and head query terms we randomly replace an entity such that newly generated triple is not observed in the graph, for eg, we sample t' such that $t' \in \mathscr{E} \setminus t$ and $(h, r, t^{'}, \tau) \notin D_{\tau}^{+}$.

Ranking Protocol: For a test triple (h, r, t, τ), we generate corrupted triples by replacing tail entity (for tail prediction) or head entity (for head prediction) with all possible entities. Filtered protocol, proposed by (Bordes et al., 2013), says that the corrupted triples must not be a part of the graph itself. To illustrate, given a test triple (h, r, t) for tail prediction task, we compute scores for the candidate set $C(h, r) = \{(h, r, t^{'} : \forall t^{'} \in \mathscr{E})\} \setminus (Train \cup Test \cup Valid) \cup (h, r, t)$. We rank all the triples in $C(h, r)$ in the increasing order of their score and find the rank of the actual triple (h, r, t). We report the mean rank over all the test queries (MR) and proportion of correct entities in top 10 rank (Hits@10).

5.4 Relation Prediction

The aim of this task is to predict the relation between two entities, i.e, for a given time-stamped triple with missing relation $(h, ?, t, \tau)$, we predict the relation r. For evaluation, we corrupt the triples with all possible relations and report the rank of the actual relation. We report Hits@1 for this task as the number of relations are quite less in numbers for both the datasets, 10 and 24 for YAGO11k and Wikidata12k respectively. Please note that we do not train our model separately for this task, rather we report the values obtained by the exact same model used for head and tail entity prediction.

The main motivation for this task is to deal with the relational conflict between two entities at a particular time-scope. For example, given the year 1992, a person 'X' and a city 'Y', one would like to know if he/she was $bornIn$ or $diedIn$ that city in that year. Through the explicit use of temporal information during training, we find that our method HyTE outperforms the baseline methods in both the datasets (as shown in 6.1)

5.5 Temporal Scope Prediction

Given the scarcity of time annotations of the KG facts, predicting time for atemporal part of the KG is an important problem. Unlike the previous baseline methods, our model can predict the time scope for a given triple. In order to perform better in this task, we want the hyperplanes to be well separated even after maintaining consistency with the positive triples. To incorporate this nature during training, we use the time-dependent negative sampling technique (TDNS Equation 2). Rest of the training procedure remains the same as the link prediction task. The model is trained on the same train split used for link prediction. In this task, we predict the time interval or the time instance τ for a given test triple $(h, r, t, ?)$. We project the relation and the entities of the triple on all the time hyperplanes and check the plausibility of that test triple on each

Dataset	YAGO11K				Wikidata12K			
Metric	Mean Rank		Hits@10(%)		Mean Rank		Hits@10(%)	
	tail	head	tail	head	tail	head	tail	head
Trans-E (Bordes et al., 2013)	504	2020	4.4	1.2	520	740	11.0	6.0
TransH (Wang et al., 2014)	354	1808	5.8	1.5	423	648	23.7	11.8
HolE (Nickel et al., 2016b)	1828	1953	29.4	13.7	734	808	25.0	12.3
t-TransE (Jiang et al., 2016)	292	1692	6.2	1.3	283	413	24.5	14.5
HyTE	**107**	**1069**	**38.4**	**16.0**	**179**	**237**	**41.6**	**25.0**

Table 2: Mean Rank (lower the better) and Hits@10(higher the better) for different methods for Entity Prediction task. Proposed method HyTE outperforms all the traditional approaches. The better performance of HyTE over t-TransE can be attributed to the fact that it incorporates time directly. Please see Section 6.1 for details.

Dataset	YAGO11K		Wikidata12K	
Metric	Mean Rank	Hits@1(%)	Mean Rank	Hits@1(%)
Trans-E (Bordes et al., 2013)	1.7	78.4	1.35	88.4
TransH (Wang et al., 2014)	1.53	76.1	1.4	88.1
HolE (Nickel et al., 2016b)	2.57	69.3	2.23	83.96
t-TransE (Jiang et al., 2016)	1.66	75.5	1.97	74.2
HyTE	**1.23**	**81.2**	**1.13**	**92.6**

Table 3: Mean Rank (lower the better) and Hits@1 (higher the better) for different methods for Relation Prediction Task. Proposed method HyTE outperforms all the traditional approaches. Even though t-TransE imposes implicit relation ordering, HyTE beats it with a high margin. Please see Section 6.1 for details.

Negative Sampling	YAGO11K	Wikidata12k
TANS (Equation 1)	14.0	29.3
TDNS (Equation 2)	**9.88**	**17.6**

Table 4: The predicted Mean Rank (lower the better) for temporal Scoping. The number of classes are 61 and 78 for YAGO11K and Wiki-data12k respectively. The results depict the effectiveness of TDNS. Please see Section 6.2

of them. For evaluation, we order the time frames in increasing order of their plausibility score for that particular triple. Now, we select the rank of the time(τ) associated with the test triple. If the associated time is an interval, we consider the lowest rank among the times in between the interval.

6 Results

Implementation details: For all the methods, we have kept batch size b = 50k on both the datasets. The dimensions of the embeddings (d) are varied in the range $\{64, 128, 256\}$. The margins(η) for all the methods are chosen from the set $\{1, 2, 5, 10\}$. Learning rate used for SGD, $lr \in \{0.01, 0.001, 0.0001\}$.

The best configuration is chosen by corresponding lowest MR on the validation set. For both YAGO11k and Wikidata12k, we obtained $d = 128$, $\eta = 10$, $lr = 0.0001$ using ℓ_1-norm in the scoring function.

Both YAGO11k and Wikidata12k contain time annotations to the granularity of days. For the temporal scoping task, we only deal with year level granularity by dropping the month and date information. Timestamps are then treated as 61 and 78 different intervals for YAGO and Wikidata respectively. The main motive behind having time classes is to distribute the time annotations in the KG uniformly. For example, less frequent year mentions are clubbed into same time class but years with high frequency forms individual classes. We take care of the unbalance that may occur in terms of number of triples in a particular interval by applying a minimum threshold of 300 triples per interval during construction. To illustrate, in Wikidata there are classes like 1596-1777, 1791-1815 with a large span as the events occurring on those points of time are quite less in KG. The years like 2013, 2014 being highly frequent are self-contained.

6.1 Performance analysis & comparison

The obtained results for different tasks are based on the above mentioned hyperparameters.

Link Prediction: The results reported in Table 2 demonstrate the efficacy of HyTE. We observe that our model outperforms the traditional state-of-the-art link prediction model HolE (Nickel et al., 2016b) by a significant margin in both the datasets. We also show a large boost in performance over TransE (Bordes et al., 2013). This significant gain empirically validates our claim that including temporal information in a principled

Test quadruples	TransE	HyTE
Gordon_Carroll, ?, Baltimore,[1928, 1928]	diedIn, **wasBornIn**	**wasBornIn**,diedIn
S.Laubenthal, ?, Washington.,[2002, 2002]	wasBornIn,**diedIn**	**diedIn**, wasBornIn
Eugene_Sander, ?, Cornell_Univ.,[1959, 1965]	worksAt,**graduatedFrom**	**graduatedFrom**, isAffiliatedTo
Ernesto_Maceda, ?, Nacionalist_Party, [1971,1987]	isMarriedTo, diedIn	**isAffiliatedTo**,diedIn

Table 5: Example of Qualitative Results on relation prediction. The order of prediction is in descending order. Correct one is in **bold**. Please refer Section 6.2 for details.

fashion helps to learn richer embeddings of the KG elements. We notice that HolE is performing significantly poor in terms of MR but it exceeds the other baselines in Hits@10 by a large margin.

Again, in comparison with the temporal model t-TransE (Jiang et al., 2016), HyTE proves to be effective. t-TransE performs better than TransE and HolE due to its implicit time incorporation through relation ordering. HyTE with its direct inclusion of time in the relation-entity semantic space outperforms all of them.

Relation prediction: Again, in this scenario, we show improvement over baselines. We hypothesize that time scope information helps to disambiguate among relations e.g. traditional methods like TransE or HolE will confuse between relations like wasBornIN, diedIn. Where time information surely helps to resolve that conflict. From Table 3, we validate this claim. In Section 6.2, we also demonstrate some qualitative result in favor of our assertion.

Temporal scoping of facts: We report the rank of correct time instance of the triple. If the triple scope is an interval of time, we consider the lowest rank that corresponds to the time within that interval. The ranks are reported in table 4 for both the datasets. The baseline model t-TransE (Jiang et al., 2016) is not applicable here as it does not use the time meta-facts directly. We also observe that the HyTE hyperplanes form a sequential map in the space. We discuss it with details in subsection 6.2.

6.2 Qualitative results

Table 5 contains some of the qualitative analyses for the relation prediction task. We mention some of the cases where transE is confusing between the temporal relations like *wasBornIn* and *diedIn*. Consider the second example in Table 5, where transE wrongly predicts *wasBornIn* . HyTE predicts *diedIn* as it has a prior knowledge that S.Laubenthal "was born in 1943", "created Excalibur on 1973" from the training data. As the query year is 2002, our method comes to such a conclusion through its relative temporal ordering. We see many examples of this kind, where HyTE is naturally learning some relation ordering in parallel with the temporal direction. We observe many type inconsistency in relation prediction for our model, e.g., for this fact *(Lauren Miller, wasBornIn, Lakeland,Florida,[1982,1982])*, our model predicts *isMarriedTo*. This can be attributed to the fact that we do not impose any type related constraints in our model. We will look forward to incorporating type and temporal constraints within our model as future work.

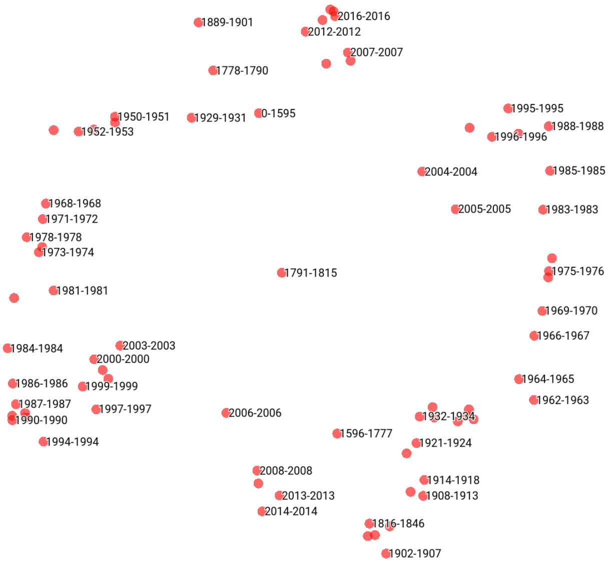

Figure 2: The figure illustrates 2-d PCA projection of the 128 dimensional time embeddings which are obtained after training HyTE for temporal scoping task. We observe that the trained time representations are forming natural clusters and ordering. Please ref Section 6.2

In Figure 2, we show a 2-d PCA projection of the 128-dimensional normal vectors of the hyperplanes. These vectors are trained for the temporal scoping task with time-dependent negative sampling (Equation 2). This figure demonstrates the ability of HyTE to structure the hyperplanes in the entity-relation space according to the data. Also, note that we do not regularize the model with any ordering constraint, but it learns the temporal ordering as well as the clustering from the data itself. We hypothesize that this phenomenon emerges due to TDNS (Equation 2). However, in case of link prediction, we notice that the extra samples are affecting performance as they originate from the KG itself.

7 Conclusion

We propose HyTE, a hyperplane-based method for learning temporally aware knowledge graph embeddings. Our method exploits temporally scoped facts of KG to perform link prediction as well as prediction of time scopes for unannotated temporal facts. Through extensive experiments on real-world datasets, we demonstrate effectiveness of HyTE over both traditional and time aware embedding methods. In future, we would like to incorporate type consistency information to further improve our model and also integrate HyTE with open-world knowledge graph completion (Shi and Weninger, 2018). We are hopeful that our proposed temporal representation learning algorithm will motivate further research on temporal KG embedding learning.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work is supported by the Ministry of Human Resource Development (MHRD), Government of India. We also thank Chandrahas Dewangan for his indispensable suggestions.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 260–269.

Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandeăić. 2014. Introducing wikidata to the linked data web. In *Proceedings of the 13th International Semantic Web Conference - Part I*, ISWC '14, pages 50–65, New York, NY, USA. Springer-Verlag New York, Inc.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 687–696.

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Encoding temporal information for time-aware link prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2350–2354. Association for Computational Linguistics.

Alexander Kotov and ChengXiang Zhai. 2012. Tapping into knowledge base for concept feedback: Leveraging conceptnet to improve search results for difficult queries. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 403–412, New York, NY, USA. ACM.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 1771–1776, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2181–2187. AAAI Press.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Bill McDowell, Nathanael Chambers, Alexander Ororbia II, and David Reitter. 2017. Event ordering with a generalized model for sieve prediction ranking. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 843–853, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 64–75. The COLING 2016 Organizing Committee.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016b. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1955–1961. AAAI Press.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 809–816, USA. Omnipress.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *16th International Conference on the World Wide Web*, pages 697–706.

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012a. Acquiring temporal constraints between relations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 992–1001. ACM.

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012b. Coupled temporal scoping of relational facts. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 73–82, New York, NY, USA. ACM.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, volume 48, pages 2071–2080.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 1–9.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 57–62. Association for Computational Linguistics.

Yafang Wang, Mingjie Zhu, Lizhen Qu, Marc Spaniol, and Gerhard Weikum. 2010. Timely yago: Harvesting, querying, and visualizing temporal knowledge from wikipedia. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 697–700, New York, NY, USA. ACM.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1112–1119. AAAI Press.

Chenyan Xiong and Jamie Callan. 2015. Query expansion with freebase. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, pages 111–120, New York, NY, USA. ACM.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 956–966.

Neural Adaptation Layers for Cross-domain Named Entity Recognition

Bill Yuchen Lin[†] and **Wei Lu**[‡]
[†]University of Southern California
[‡]Singapore University of Technology and Design
yuchen.lin@usc.edu, luwei@sutd.edu.sg

Abstract

Recent research efforts have shown that neural architectures can be effective in conventional information extraction tasks such as named entity recognition, yielding state-of-the-art results on standard newswire datasets. However, despite significant resources required for training such models, the performance of a model trained on one domain typically degrades dramatically when applied to a different domain, yet extracting entities from new emerging domains such as social media can be of significant interest. In this paper, we empirically investigate effective methods for conveniently adapting an existing, well-trained neural NER model for a new domain. Unlike existing approaches, we propose lightweight yet effective methods for performing domain adaptation for neural models. Specifically, we introduce adaptation layers on top of existing neural architectures, where no re-training using the source domain data is required. We conduct extensive empirical studies and show that our approach significantly outperforms state-of-the-art methods.

1 Introduction

Named entity recognition (NER) focuses on extracting named entities in a given text while identifying their underlying semantic types. Most earlier approaches to NER are based on conventional structured prediction models such as conditional random fields (CRF) (Lafferty et al., 2001; Sarawagi and Cohen, 2004), relying on hand-crafted features which can be designed based on domain-specific knowledge (Yang and Cardie, 2012; Passos et al., 2014; Luo et al., 2015). Recently, neural architectures have been shown effective in such a task, whereby minimal feature engineering is required (Lample et al., 2016; Ma and Hovy, 2016; Peters et al., 2017; Liu et al., 2018). Domain adaptation, as a special case for transfer learning, aims to exploit the abundant data of well-studied source domains to improve the performance in target domains of interest (Pan and Yang, 2010; Weiss et al., 2016). There is a growing interest in investigating the transferability of neural models for NLP. Two notable approaches, namely INIT (parameter initialization) and MULT (multitask learning), have been proposed for studying the transferrability of neural networks under tasks such as sentence (pair) classification (Mou et al., 2016) and sequence labeling (Yang et al., 2017b).

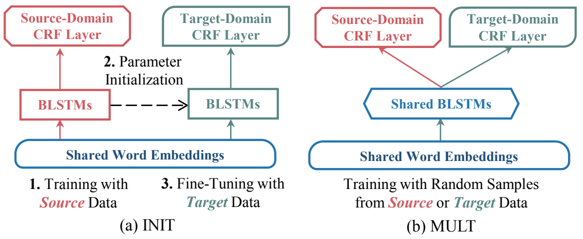

Figure 1: Two existing adaptation approaches for NER.

The INIT method first trains a model using labeled data from the source domain; next, it initializes a target model with the learned parameters; finally, it fine-tunes the initialized target model using labeled data from the target domain. The MULT method, on the other hand, simultaneously trains two models using both source and target data respectively, where some parameters are shared across the two models during the learning process. Figure 1 illustrates the two approaches based on the BLSTM-CRF (bidirectional LSTM augmented with a CRF layer) architecture for NER. While such approaches make intuitive senses, they also come with some limitations.

First, these methods utilize shared domain-general word embeddings when performing learning from both source and target domains. This essentially assumes there is no domain shift of input feature spaces. However, cases when the two domains are distinct (words may contain different se-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2012–2022
Brussels, Belgium, October 31 - November 4, 2018.

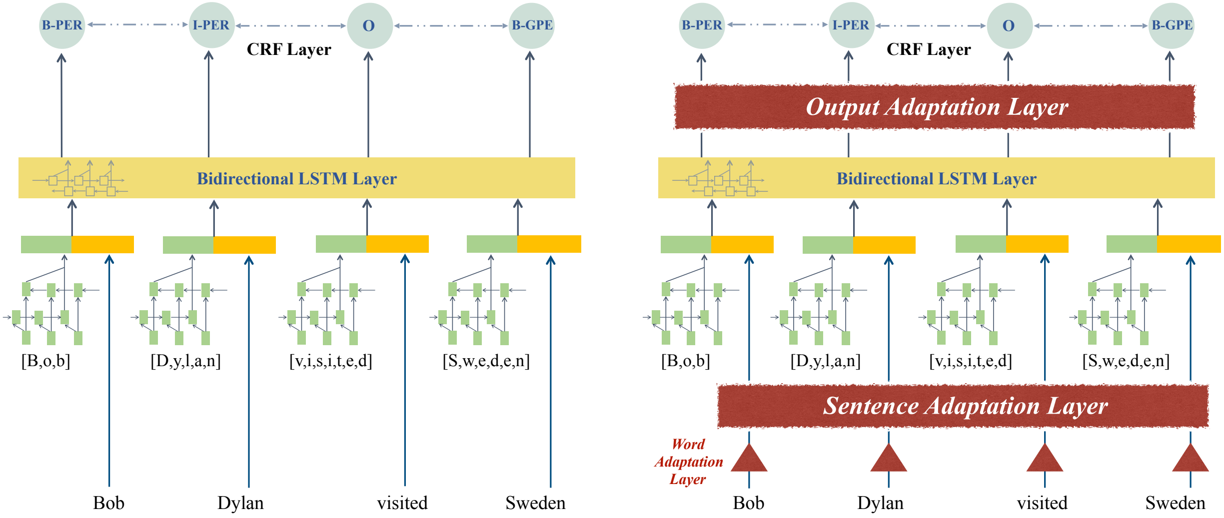

Figure 2: Our base model (left) and target model (right) where we insert new adaptation layers (highlighted in red).

mantics across two domains), we believe such an assumption can be weak.

Second, existing approaches such as INIT directly augment the LSTM layer with a new output CRF layer when learning models for the target domain. One basic assumption involved here is that the model would be able to re-construct a new CRF layer that can capture not only the variation of the input features (or hidden states outputted from LSTM) to the final CRF layer across two domains, but also the structural dependencies between output nodes in the target output space. We believe this overly restrictive assumption may prevent the model from capturing rich, complex cross-domain information due to the inherent linear nature of the CRF model.

Third, most methods involving cross-domain embedding often require highly time-consuming retraining word embeddings on domain-specific corpora. This makes it less realistic in scenarios where source corpora are huge or even inaccessible. Also, MULT-based methods need retraining on the source-domain data for different target domains. We think this disadvantage of existing methods hinders the neural domain adaptation methods for NER to be practical in real scenarios.

In this work, we propose solutions to address the above-mentioned issues. Specifically, we address the first issue at both the word and sentence level by introducing a *word adaptation layer* and *a sentence adaptation layer* respectively, bridging the gap between the two input spaces. Similarly, an *output adaptation layer* is also introduced between the LSTM and the final CRF layer, capturing the variations in the two output spaces. Furthermore, we introduce a single hyper-parameter that controls how much information we would like to preserve from the model trained on the source domain. These approaches are lightweight, without requiring re-training on data from the source domain. We show through extensive empirical analysis as well as ablation study that our proposed approach can significantly improve the performance of cross-domain NER over existing transfer approaches.

2 Base Model

We briefly introduce the BLSTM-CRF architecture for NER, which serves as our *base model* throughout this paper. Our base model is the combination of two recently proposed popular works for named entity recognition by Lample et al. (2016) and Ma and Hovy (2016). Figure 2 illustrates the BLSTM-CRF architecture.

Following Lample et al. (2016), we develop the comprehensive word representations by concatenating pre-trained word embeddings and character-level word representations, which are constructed by running a BLSTM over sequences of character embeddings. The middle BLSTM layer takes a sequence of comprehensive word representations and produces a sequence of hidden states, representing the contextual information of each token. Finally, following Ma and Hovy (2016), we build the final CRF layer by utilizing potential functions describing local structures to define the conditional probabilities of complete predictions for the given input sentence.

This architecture is selected as our base model due to its generality and representativeness. We note that several recently proposed models (Peters et al., 2017; Liu et al., 2018) are built based on it. As our focus is on how to better transfer such architectures for NER, we include further discussions of the model and training details in our supplementary material.

3 Our Approach

We first introduce our proposed three adaptation layers and describe the overall learning process.

3.1 Word Adaptation Layer

Most existing transfer approaches use the same domain-general word embeddings for training both source and target models. Assuming that there is little domain shift of input feature spaces, they simplify the problem as *homogeneous transfer learning* (Weiss et al., 2016). However, this simplified assumption becomes weak when two domains have apparently different language styles and involve considerable domain-specific terms that may not share the same semantics across the domains; for example, the term "*cell*" has different meaning in biomedical articles and product reviews. Furthermore, some important domain-specific words may not be present in the vocabulary of domain-general embeddings. As a result, we have to regard such words as out-of-vacabulary (OOV) words, which may be harmful to the transfer learning process.

Stenetorp et al. (2012) show that domain-specific word embeddings tend to perform better when used in supervised learning tasks.[1] However, maintaining such an improvement in the transfer learning process is very challenging. This is because two domain-specific embeddings are trained separately on source and target datasets, and therefore the two embedding spaces are *heterogeneous*. Thus, model parameters trained in each model are heterogeneous as well, which hinders the transfer process. How can we address such challenges while maintaining the improvement by using domain-specific embeddings?

We address this issue by developing a word adaptation layer, bridging the gap between the source and target embedding spaces, so that both input features and model parameters become homogeneous across domains. Popular pre-trained word embeddings are usually trained on very large corpora, and thus methods requiring re-training them can be extremely costly and time-consuming. We propose a straightforward, lightweight yet effective method to construct the word adaptation layer that projects the learned embeddings from the target embedding space into the source space. This method only requires some corpus-level statistics from the datasets (used for learning embeddings) to build the *pivot lexicon* for constructing the adaptation layer.

3.1.1 Pivot Lexicon

A *pivot word pair* is denoted as (w_s, w_t), where $w_s \in \mathcal{X}_S$ and $w_t \in \mathcal{X}_T$. Here, $\mathcal{X}_S$ and $\mathcal{X}_T$ are source and target vocabularies. A *pivot lexicon* $\mathcal{P}$ is a set of such word pairs.

To construct a pivot lexicon, first, motivated by Tan et al. (2015), we define $\mathcal{P}_1$, which consists of the ordinary words that have high relative frequency in both source and target domain corpora: $\mathcal{P}_1 = \{(w_s, w_t) | w_s = w_t, f(w_s) \geq \phi_s, f(w_t) \geq \phi_t\}$, where $f(w)$ is the frequency function that returns the number of occorrence of the word w in the dataset, and ϕ_s and ϕ_t are word frequency thresholds[2]. Optionally, we can utilize a customized word-pair list $\mathcal{P}_2$, which gives mappings between domain-specific words across domains, such as normalization lexicons (Han and Baldwin, 2011; Liu et al., 2012). The final lexicon is thus defined as $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$.

3.1.2 Projection Learning

Mathematically, given the pre-trained domain-specific word embeddings $\mathbf{V_S}$ and $\mathbf{V_T}$ as well as a pivot lexicon $\mathcal{P}$, we would like to learn a linear projection transforming word vectors from $\mathbf{V_T}$ into $\mathbf{V_S}$. This idea is based on a bilingual word embedding model (Mikolov et al., 2013), but we adapt it to this domain adaptation task.

We first construct two matrices $\mathbf{V_S^*}$ and $\mathbf{V_T^*}$, where the i-th rows of these two matrices are the vector representations for the words in the i-th entry of $\mathcal{P}$: $\mathcal{P}^{(i)} = (w_s^i, w_t^i)$. We use $\mathbf{V_S^{*i}}$ to denote the vector representation of the word w_s^i, and similarly for $\mathbf{V_T^{*i}}$ and w_t^i.

Next, we learn a transformation matrix $\mathbf{Z}$ minimizing the distances between $\mathbf{V_S^*}$ and $\mathbf{V_T^*Z}$ with

---

[1] We also confirm this claim with experiments presented in supplementary materials.

[2] A simple way of setting them is to choose the frequency of the k-th word in the word lists sorted by frequencies.

the following loss function, where c_i is the confidence coefficient for the entry $\mathcal{P}^{(i)}$, highlighting the significance of the entry:

$$\underset{\mathbf{Z}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{P}|} c_i \left\| \mathbf{V}_{\mathbf{T}}^{*\mathbf{i}} \mathbf{Z} - \mathbf{V}_{\mathbf{S}}^{*\mathbf{i}} \right\|^2 ,$$

We use normalized frequency ($\bar{f}$) and Sørensen-Dice coefficient (Sørensen, 1948) to describe the significance of each word pair:

$$\bar{f}(w_s) = \frac{f(w_s)}{\max_{w' \in \mathcal{X}_S} f(w')}, \quad \bar{f}(w_t) = \frac{f(w_t)}{\max_{w' \in \mathcal{X}_T} f(w')}$$

$$c_i = \frac{2 \cdot \bar{f}(w_s^i) \cdot \bar{f}(w_t^i)}{\bar{f}(w_s^i) + \bar{f}(w_t^i)}$$

The intuition behind this scoring method is that a word pair is more important when both words have high relative frequency in both domains. This is because such words are likely more domain-independent, conveying identical or similar semantics across these two different domains.

Now, the matrix $\mathbf{Z}$ exactly gives the weights to the word adaptation layer, which takes in the target domain word embeddings and returns the transformed embeddings in the new space. We learn $\mathbf{Z}$ with stochastic gradient descent. After learning, the projected new embeddings would be $\mathbf{V_T Z}$, which would be used in the subsequent steps as illustrated in Figure 2 and Figure 3. With such a word-level input-space transformation, the parameters of the pre-trained source models based on $\mathbf{V_S}$ can still be relevant, which can be used in subsequent steps.

We would like to highlight that, unlike many previous approaches to learning cross-domain word embeddings (Bollegala et al., 2015; Yang et al., 2017a), the learning of our word adaptation layer involves no modifications to the source-domain embedding spaces. It also requires no re-training of the embeddings based on the target-domain data. Such a distinctive advantage of our approach comes with some important practical implications: it essentially enables the transfer learning process to work directly on top of a well-trained model by performing adaptation without involving significant re-training efforts. For example, the existing model could be one that has already gone through extensive training, tuning and testing for months based on large datasets with embeddings learned from a particular domain (which may be different from the target domain).

3.2 Sentence Adaptation Layer

The word adaptation layer serves as a way to bridge the gap of heterogeneous input spaces, but it does so only at the word level and is context independent. We can still take a step further to address the input space mismatch issue at the sentence level, capturing the contextual information in the learning process of such a mapping based on labeled data from the target domain. To this end, we augment a BLSTM layer right after the word adaptation layer (see Figure 2), and we name it the *sentence adaptation layer*.

This layer pre-encodes the sequence of projected word embeddings for each target instance, before they serve as inputs to the LSTM layer inside the base model. The hidden states for each word generated from this layer can be seen as the further transformed word embeddings capturing target-domain specific contextual information, where such a further transformation is learned in a supervised manner based on target-domain annotations. We also believe that with such a sentence adaptation layer, the OOV issue mentioned above may also be partially alleviated. This is because without such a layer, OOV words would all be mapped to a single fixed vector representation – which is not desirable; whereas with such a sentence adaptation layer, each OOV word would be assigned their "transformed" embeddings based on its respective contextual information.

3.3 Output Adaptation Layer

We focus on the problem of performing domain adaptation for NER under a general setup, where we assume the set of output labels for the source and target domains could be different. Due to the heterogeneousness of output spaces, we have to reconstruct the final CRF layer in the target models.

However, we believe solely doing this may not be enough to address the *labeling difference* problem as highlighted in (Jiang, 2008) as the two output spaces may be very different. For example, in the sentence "***Taylor*** *released her new* ***songs***", "*Taylor*" should be labeled as "MUSIC-ARTIST" instead of "PERSON" in some social media NER datasets; this suggests *re-classifying* with contextual information is necessary. In another example, where we have a tweet "*so...****#kktny*** *in 30 mins?*"; here we should recognize "*#kktny*" as a CREATIVE-WORK entity, but there is little similar instances in newswire data, indicating that

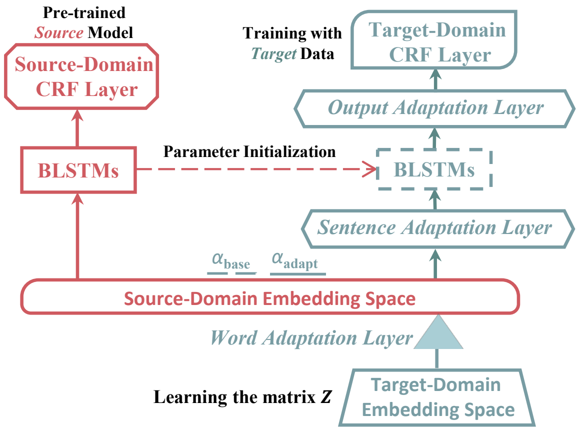

Figure 3: Overview of our transfer learning process.

context-aware *re-recognition* is also needed.

How can we perform re-classification and re-recognition with contextual information in the target model? We answer this question by inserting a BLSTM *output adaptation layer* in the base model, right before the final CRF layer, to capture variations in outputs with contextual information. This output adaption layer takes the output hidden states from the BLSTM layer from the base model as its inputs, producing a sequence of new hidden states for the re-constructed CRF layer. Without this layer, the learning process directly updates the pre-trained parameters of the base model, which may lead to loss of knowledge that can be transferred from the source domain.

3.4 Overall Learning Process

Figure 3 depicts our overall learning process. We initialize the base model with the parameters from the pre-trained source model, and tune the weights for all layers — including layers from the base model, sentence and output adaptation layers, and the CRF layer. We use different learning rates when updating the weights in different layers using Adam (Kingma and Ba, 2014). In all our experiments, we fixed the weights to $\mathbf{Z}$ for the word adaptation layers to avoid over-fitting. This allows us to preserve the knowledge learned from the source domain while effectively leveraging the limited training data from the target domain.

Similar to Yang et al. (2017b), who utilizes a hyper-parameter λ for controlling the transferability, we also introduce a hyper-parameter ψ that serves a similar purpose — it captures the relation between the learning rate used for the base model (α_{base}) and the learning rate used for the adaptation layers plus the final CRF layer $\alpha_{\text{base}} = \psi \cdot \alpha_{\text{adapt}}$.

If $\psi = 0$, we fix the learned parameters (from source domain) of the base model completely (Ours-Frozen). If $\psi = 1$, we treat all the layers equally (Ours-FineTune).

4 Experimental Setup

In this section, we present the setup of our experiments. We show our choice for source and target domains, resources for embeddings, the datasets for evaluation and finally the baseline methods.

4.1 Source and Target Domains

We evaluate our approach with the setting that the source domain is *newswire* and the target domain is *social media*. We designed this experimental setup based on the following considerations:

- *Challenging*: Newswire is a well-studied domain for NER and existing neural models perform very well (around 90.0 F1-score (Ma and Hovy, 2016)). However, the performance drop dramatically in social media data (around 60.0 F-score (Strauss et al., 2016)).
- *Important*: Social media is a rich soil for text mining (Petrovic et al., 2010; Rosenthal and McKeown, 2015; Wang and Yang, 2015), and NER is of significant importance for other information extraction tasks in social media (Ritter et al., 2011a; Peng and Dredze, 2016; Chou et al., 2016).
- *Representative*: The noisy nature of user generated content as well as emerging entities with novel surface forms make the domain shift very salient (Finin et al., 2010; Han et al., 2016).

Nevertheless, the techniques developed in this paper are domain independent and thus can be used for other learning tasks across any two domains so long as we have the necessary resources.

4.2 Resources for Cross-domain Embeddings

We utilizes GloVe (Pennington et al., 2014) to train domain-specific and domain-general word embeddings from different corpora, denoted as follows: 1) *source_emb*, which is trained on the newswire domain corpus (*NewYorkTimes* and *DailyMail* articles); 2) *target_emb*, which is trained on a social media corpus (Archive Team's Twitter stream grab[3]); 3) *general_emb*, which is pre-trained on CommonCrawl containing both formal

---

[3] `https://archive.org/details/twitterstream`

	Co	On	Ri	Wn
#train_token	204,567	848,220	37,098	46,469
#dev_token	51,578	144,319	4,461	16,261
#test_token	46,666	49,235	4,730	61,908
#train_sent.	14,987	33,908	1,915	2,394
#dev_sent.	3,466	5,771	239	1,000
#test_sent.	3,684	1,898	240	3,856
#entity_type	4	11	10	10

Table 1: Statistics of the NER datasets.

and user-generated content.[4] We obtain the intersection of the top 5K words from source and target vocabularies sorted by frequency to build $\mathcal{P}_1$. For $\mathcal{P}_2$, we utilize an existing twitter normalization lexicon containing 3,802 word pairs(Liu et al., 2012). More details are in supplemental material.

4.3 NER Datasets for Evaluation

For the source newswire domain, we use the following two datasets: OntoNotes-nw – the newswire section of OntoNotes 5.0 release dataset (On) (Weischedel et al., 2013) that is publicly available[5], as well as the CoNLL03 NER dataset (Co) (Sang and Meulder, 2003). For the first dataset, we randomly split the dataset into three sets: 80% for training, 15% for development and 5% for testing. For the second dataset, we follow their provided standard train-dev-test split. For the target domain, we consider the following two datasets: Ritter11 (Ri) (Ritter et al., 2011b) and WNUT16 (Wn) (Strauss et al., 2016), both of which are publicly available. The statistics of the four datasets we used in the paper are shown in Table 1.

4.4 Baseline Transfer Approaches

We present the baseline approaches, which were originally investigated by Mou et al. (2016). Lee et al. (2017) explored the INIT method for NER, while Yang et al. (2017b) extended the MULT method for sequence labeling.

INIT: We first train a source model $\mathcal{M}_S$ using the source-domain training data $\mathcal{D}_S$. Next, we construct a target model $\mathcal{M}_T$ and reconstruct the final CRF layer to address the issue of different output spaces. We use the learned parameters of $\mathcal{M}_S$ to initialize $\mathcal{M}_T$ excluding the CRF layer. Finally, INIT-FineTune continues training $\mathcal{M}_T$ with the target-domain training data $\mathcal{D}_T$, while INIT-Frozen instead only updates the parameters of the newly constructed CRF layer.

MULT: Multi-task learning based transfer method simultaneously trains $\mathcal{M}_S$ and $\mathcal{M}_T$ using $\mathcal{D}_S$ and $\mathcal{D}_T$. The parameters of $\mathcal{M}_S$ and $\mathcal{M}_T$ excluding their CRF layers are shared during the training process. Both Mou et al. (2016) and Yang et al. (2017b) follow Collobert and Weston (2008) and use a hyper-parameter λ as the probability of choosing an instance from $\mathcal{D}_S$ instead of $\mathcal{D}_T$ to optimize the model parameters. By selecting the hyper-parameter λ, the multi-task learning process tends to perform better in target domains. Note that this method needs re-training of the source model with $\mathcal{D}_S$ every time we would like to build a new target model, which can be time-consuming especially when $\mathcal{D}_S$ is large.

MULT+INIT: This is a combination of the above two methods. We first use INIT to initialize the target model. Afterwards, we train the two models as what MULT does.

5 Results and Discussion

5.1 Main Results

We primarily focus on the discussion of experiments with a particular setup where $\mathcal{D}_S$ is set to OntoNotes-nw and $\mathcal{D}_T$ is Ritter11. In the experiments, "in-domain" means we only use $\mathcal{D}_T$ to train our base model without any transfer from the source domain. "Δ" represents the amount of improvement we can obtain (in terms of F measure) using transfer learning over "in-domain" for each transfer method. The hyper-parameters ψ and λ are tuned from $\{0.1, 0.2, ..., 1.0\}$ on the development set, and we show the results based on the developed hyper-parameters.

We first conduct the first set of experiments to evaluate performance of different transfer methods under the assumption of homogeneous input spaces. Thus, we utilize the same word embeddings (general_emb) for training both source and target models. Consequently we remove the word adaptation layer (cd_emb) in our approach under this setting. The results are listed in Table 2. As we can observe, the INIT-Frozen method leads to a slight "negative transfer", which is also reported in the experiments of Mou et al. (2016). This indicates that solely updating the parameters of the final CRF layer is not enough for performing re-classification and re-recognition of the named entities for the new target domain. The INIT-FineTune method yields better results for it also updates the parameters of the middle LSTM

---

[4] `https://nlp.stanford.edu/projects/glove/`

[5] `https://catalog.ldc.upenn.edu/ldc2013t19`

Settings	P	R	F	Δ
in-domain	72.73	56.14	63.37	-
INIT-Frozen	72.61	56.11	63.30	-0.07
INIT-FineTune	73.13	56.55	63.78	+0.41
MULT	74.07	57.51	64.75	+1.38
MULT+INIT	74.12	57.57	64.81	+1.44
Ours (*w/o word adapt*)	74.87	57.95	**65.33**	**+1.96**

Table 2: Comparisons of different methods for **homogeneous** input spaces. ($\mathcal{D}_S$ = ON, $\mathcal{D}_T$ = RI)

Settings	P	R	F	Δ
in-domain	72.51	57.11	63.90	-
INIT-Frozen	72.65	55.25	62.77	-1.13
INIT-FineTune	72.83	56.73	63.78	-0.12
MULT	73.11	57.35	64.28	+0.38
MULT+INIT	73.13	57.31	64.26	+0.36
Ours	75.87	59.03	66.40	+2.50
w/ Yang et al. (2017a)	76.12	59.10	66.53	+2.63
w/o word adapt. layer	73.29	57.61	64.51	+0.61

Table 3: Comparisons of different methods for **heterogeneous** input spaces. ($\mathcal{D}_S$ = ON, $\mathcal{D}_T$ = RI)

layers in the base model to mitigate the heterogeneousness. The MULT and MULT+INIT methods yield higher results, partly due to the fact that they can better control the amount of transfer through tuning the hyper-parameter. Our proposed transfer approach outperforms all these baseline approaches. It not only controls the amount of transfer across the two domains but also explicitly captures variations in the input and output spaces when there is significant domain shift.

We use the second set of experiments to understand the effectiveness of each method when dealing with heterogeneous input spaces. We use source_emb for training source models and target_emb for learning the target models. From the results shown in Table 3, we can find that all the baseline methods degrade when the two word embeddings used for training source models and learning target models are different from each other. The heterogeneousness of input feature space hinders them to better use the information from source domains. However, with the help of word and sentence adaptation layers, our method achieves better results. The experiment on learning without the word adaptation layer also confirms the importance of such a layer.[6]

Our results are also comparable to the results when the cross-lingual embedding method of Yang et al. (2017a) is used instead of the word adaptation layer. However, as we mentioned earlier, their method requires re-training the embeddings from target domain, and is more expensive.

5.2 Ablation Test

To investigate the effectiveness of each component of our method, we conduct ablation test based on our full model (F=66.40) reported in Table 3. We use Δ to denote the differences of the performance between each setting and our model. The results of ablation test are shown in Table 4. We first set ψ to 0 and 1 respectively to investigate the two special variant (Ours-Frozen, Ours-FineTune) of our method. We find they both perform worse than using the optimal ψ (0.6).

One natural concern is whether our performance gain is truly caused by the effective approach for cross-domain transfer learning, or is simply because we use a new architecture with more layers (that is built on top of the base model) for learning the target model. To understand this, we carry out an experiment named "w/o transfer" by setting ψ to 1, where we randomly initialize the parameters of the middle BLSTMs in the target model instead of using source model parameters. Results show that such a model does not perform well, confirming the effectiveness of transfer learning with our proposed adaptation layers. Results also confirm the importance of all the three adaptation layers that we introduced. Learning the confidence scores (c_i) and employing the optional $\mathcal{P}_2$ are also helpful but they appear to be playing less significant roles.

5.3 Additional Experiments

As shown in Table 5, we conduct some additional experiments to investigate the significance of our improvements on different source-target domains, and whether the improvement is simply because of the increased model expressiveness due to a larger number of parameters.[7]

We first set the hidden dimension to be the same as the dimension of source-domain word embeddings for the sentence adaptation layer, which is 200 (I/M-200). The dimension used for the output adaptation layer is just half of that of the base BLSTM model. Overall, our model roughly

---

[6] We use the approximate randomization test (Yeh, 2000) for statistical significance of the difference between "Ours" and "MULT+INIT". Our improvements are statistically significant with a p-value of 0.0033.

[7] In this table, I-200/300 and M-200/300 represent the best performance from {INIT-Frozen, INIT-FineTune} and {MULT, MULT+INIT} respectively after tuning; "in-do." here is the best score of our base model without transfer.

Settings		F	Δ
ψ=0.0	Ours-Frozen	63.95	-2.45
ψ=1.0	Ours-FineTune	63.40	-3.00
ψ=1.0	w/o transfer	63.26	-3.14
ψ=0.6	w/o using confidence c_i	66.04	-0.36
ψ=0.6	w/o using $\mathcal{P}_2$	66.11	-0.29
ψ=0.6	w/o word adapt. layer	64.51	-1.89
ψ=0.6	w/o sentence adapt. layer	65.25	-1.15
ψ=0.6	w/o output adapt. layer	64.84	-1.56

Table 4: Comparing different settings of our method.

$\mathcal{D}_S, \mathcal{D}_T$	in-do.	I-200	M-200	I-300	M-300	Ours
(ON, RI)	63.37	+0.41	+1.44	+0.43	+1.48	+3.03
(CO, RI)		+0.23	+0.81	+0.22	+ 0.88	+1.86
(ON, WN)	51.03	+0.89	+1.72	+0.88	+1.77	+3.16
(CO, WN)		+0.69	+1.04	+0.71	+1.13	+2.38

Table 5: Results of transfer learning methods on different datasets with different number of LSTM units in the base model. (I: INIT; M: MULT; 200/300: number of LSTM units).

involves 117.3% more parameters than the base model.[8] To understand the effect of a larger parameter size, we further experimented with hidden unit size as 300 (I/M-300), leading to a parameter size of $213,750$ that is comparable to "Ours" ($203,750$). As we can observe, our approach outperforms these approaches consistently in the four settings. More experiments with other settings can be found in the supplementary material.

5.4 Effect of Target-Domain Data Size

To assess the effectiveness of our approach when we have different amounts of training data from the target domain, we conduct additional experiments by gradually increasing the amount of the target training data from 10% to 100%. We again use the OntoNotes-nw and Ritter11 as $\mathcal{D}_S$ and $\mathcal{D}_T$, respectively. Results are shown in Figure 4. Experiments for INIT and MULT are based on the respective best settings used in Table 5. We find that the improvements of baseline methods tend to be smaller when the target training set becomes larger. This is partly because INIT and MULT do not explicitly preserve the parameters from source models in the constructed target models. Thus, the transferred information is diluted while we train the target model with more data. In contrast, our transfer method explicitly saves the transferred information in the base part of our target model, and we use separate learning rates to help the target model to preserve the transferred knowledge. Similar experiments on other datasets are shown in the supplementary material.

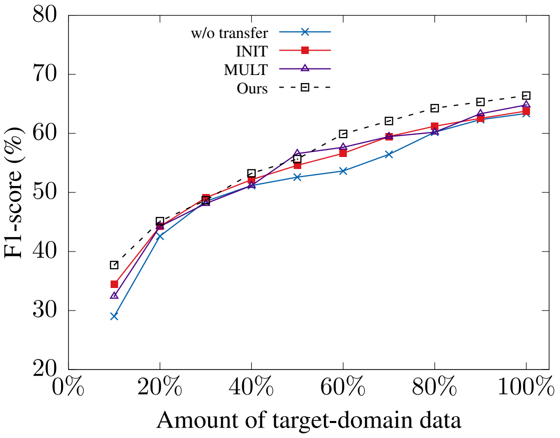

Figure 4: F1-score vs. amount of target-domain data.

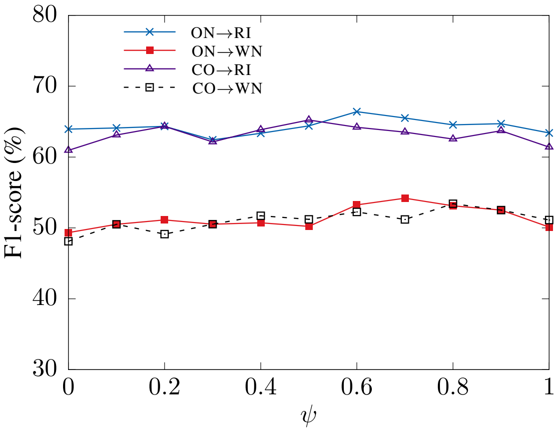

Figure 5: Performance at different ψ of our methods.

5.5 Effect of the Hyper-parameter ψ

We present a set of experiments around the hyper-parameter ψ in Figure 5. Such experiments over different datasets can shed some light on how to select this hyper-parameter. We find that when the target data set is small (Ritter11), the best ψ are 0.5 and 0.6 respectively for the two source domains, whereas when the target data set is larger (WNUT16), the best ψ becomes 0.7 and 0.8. The results suggest that the optimal ψ tends to be relatively larger when the target data set is larger.

6 Related Work

Domain adaptation and transfer learning has been a popular topic that has been extensively studied in the past few years (Pan and Yang, 2010). For well-studied conventional feature-based models in NLP, there are various classic transfer approaches, such as EasyAdapt (Daumé, 2007), instance weighting (Jiang and Zhai, 2007) and

---

[8] Base Model: $(25 \times 50 + 50^2) + (250 \times 200 + 200^2) = 93,750$; Ours: $(25 \times 50 + 50^2) + (200 \times 200 + 200^2) + (250 \times 200 + 200^2) + (200 \times 100 + 100^2) = 203,750$.

structural correspondence learning (Blitzer et al., 2006). Fewer works have been focused on transfer approaches for neural models in NLP. Mou et al. (2016) use intuitive transfer methods (INIT and MULT) to study the transferability of neural network models for the sentence (pair) classification problem; Lee et al. (2017) utilize the INIT method on highly related datasets of electronic health records to study their specific de-identification problem. Yang et al. (2017b) use the MULT approach in sequence tagging tasks including named entity recognition. Following the MULT scheme, Wang et al. (2018) introduce a label-aware mechanism into maximum mean discrepancy (MMD) to explicitly reduce domain shift between the same labels across domains in medical data. Their approach requires the output space to be the same in both source and target domains. Note that the scenario in our paper is that the output spaces are different in two domains.

All these existing works do not use domain-specific embeddings for different domains and they use the same neural model for source and target models. However, with our word adaptation layer, it opens the opportunity to use domain-specific embeddings. Our approach also addresses the domain shift problem at both input and output level by re-constructing target models with our specifically designed adaptation layers. The hyper-parameter ψ in our proposed methods and λ in MULT both control the knowledge transfer from source domain in the transfer learning process. While our method works on top of an existing pre-trained source model directly, MULT needs re-training with source domain data each time they train a target model.

Fang and Cohn (2017) add an "augmented layer" before their final prediction layer for cross-lingual POS tagging — which is a simple multilayer perceptron performing local adaptation for each token separately — ignoring contextual information. In contrast, we employ a BLSTM layer due to its ability in capturing contextual information, which was recently shown to be crucial for sequence labeling tasks such as NER (Ma and Hovy, 2016; Lample et al., 2016). We also notice that a similar idea to ours has been used in the recently proposed Deliberation Network (Xia et al., 2017) for the sequence generation task, where a second-pass decoder is added to a first-pass decoder to polish sequences generated by the latter.

We propose to learn the word adaptation layer in our task inspired by two prior studies. Fang and Cohn (2017) use the cross-lingual word embeddings to obtain distant supervision for target languages. Yang et al. (2017a) propose to re-train word embeddings on target domain by using regularization terms based on the source-domain embeddings, where some hyper-parameter tuning based on down-stream tasks is required. Our word adaptation layer serves as a linear-transformation (Mikolov et al., 2013), which is learned based on corpus level statistics. Although there are alternative methods that also learn a mapping between embeddings learned from different domains (Faruqui and Dyer, 2014; Artetxe et al., 2016; Smith et al., 2017), such methods usually involve modifying source domain embeddings, and thus re-training of the source model based on the modified source embeddings would be required for the subsequent transfer process.

7 Conclusion

We proposed a novel, lightweight transfer learning approach for cross-domain NER with neural networks. Our introduced transfer method performs adaptation across two domains using adaptation layers augmented on top of the existing neural model. Through extensive experiments, we demonstrated the effectiveness of our approach, reporting better results over existing transfer methods. Our approach is general, which can be potentially applied to other cross-domain structured prediction tasks. Future directions include investigations on employing alternative neural architectures such as convolutional neural networks (CNNs) as adaptation layers, as well as on how to learn the optimal value for ψ from the data automatically rather than regarding it as a hyper-parameter. [9]

Acknowledgments

We would like to thank the anonymous reviewers for their thoughtful and constructive comments. Most of this work was done when the first author was visiting Singapore University of Technology and Design (SUTD). This work is supported by DSO grant DSOCL17061, and is partially supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 1 SUTDT12015008.

---

[9] We make our supplementary material and code available at `http://statnlp.org/research/ie`.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proc. of EMNLP*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of ACL*.

Danushka Bollegala, Takanori Maehara, and Ken ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proc. of ACL*.

Chien-Lung Chou, Chia-Hui Chang, and Ya-Yun Huang. 2016. Boosted web named entity recognition via tri-training. *ACM Trans. Asian and Low-Resource Lang. Inf. Process.*

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proc. of ICML*.

Hal Daumé. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proc. of ACL*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proc. of EACL*.

Timothy W. Finin, William Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in twitter data with crowdsourcing. In *Proc. of Mturk@HLT-NAACL*.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proc. of ACL*.

Bo Han, Alan Ritter, Leon Derczynski, Wei Xu, and Tim Baldwin. 2016. Results of the WNUT16 named entity recognition shared task.

Jing Jiang. 2008. *Domain adaptation in natural language processing*. Ph.D. thesis.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proc. of ACL*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL-HLT*.

Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. 2017. Transfer learning for named-entity recognition with neural networks. *arXiv preprint arXiv:1705.06273*.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012. A broad-coverage normalization system for social media language. In *Proc. of ACL*.

Liyuan Liu, Jingbo Shang, Frank F. Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proc. of AAAI*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint named entity recognition and disambiguation. In *Proc. of EMNLP*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of ACL*.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *Proc. of EMNLP*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10).

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proc. of ACL*.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *Proc. of HLT-NAACL*.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011a. Named entity recognition in tweets: an experimental study. In *Proc. of EMNLP*.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011b. Named entity recognition in tweets: An experimental study. In *Proc. of EMNLP*.

Sara Rosenthal and Kathy McKeown. 2015. I couldn't agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *Proc. of SIGDIAL*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*.

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Proc. of NIPS*.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax.

Thorvald Sørensen. 1948. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. Skr.*

Pontus Stenetorp, Hubert Soyer, Sampo Pyysalo, Sophia Ananiadou, and Takashi Chikayama. 2012. Size (and domain) matters: Evaluating semantic word space representations for biomedical text. *Proc. of SMBM*.

Benjamin Strauss, Bethany E. Toma, Alan Ritter, Marie-Catherine de Marneffe, and Wei Xu. 2016. Results of the wnut16 named entity recognition shared task. In *Proc. of W-NUT*.

Luchen Tan, Haotian Zhang, Charles L. A. Clarke, and Mark D. Smucker. 2015. Lexical comparison between wikipedia and twitter corpora by using word embeddings. In *Proc. of ACL*.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proc. of EMNLP*.

Zhenghui Wang, Yanru Qu, Liheng Chen, Jian Shen, Weinan Zhang, Shaodian Zhang, Yimei Gao, G. Gu, Ken Chen, and Yong Yu. 2018. Label-aware double transfer learning for cross-specialty medical named entity recognition. In *Proc. of NAACL-HLT*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium*.

Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big Data*.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Proc. of NIPS*.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *Proc. of EMNLP-CoNLL*.

Wei Yang, Wei Lu, and Vincent W. Zheng. 2017a. A simple regularization-based algorithm for learning cross-domain word embeddings. In *Proc. of EMNLP*.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017b. Transfer learning for sequence tagging with hierarchical recurrent networks. In *Proc. of ICLR*.

Alexander S. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proc. of COLING*.

Entity Linking within a Social Media Platform: A Case Study on Yelp

Hongliang Dai[1], **Yangqiu Song**[1], **Liwei Qiu**[2] **and Rijia Liu**[2]
[1]Department of CSE, HKUST
[2]Tencent Technology (SZ) Co., Ltd.
[1]`{hdai,yqsong}@cse.ust.hk`
[2]`{drolcaqiu,rijialiu}@tencent.com`

Abstract

In this paper, we study a new entity linking problem where both the entity mentions and the target entities are within a same social media platform. Compared with traditional entity linking problems that link mentions to a knowledge base, this new problem have less information about the target entities. However, if we can successfully link mentions to entities within a social media platform, we can improve a lot of applications such as comparative study in business intelligence and opinion leader finding. To study this problem, we constructed a dataset called Yelp-EL, where the business mentions in Yelp reviews are linked to their corresponding businesses on the platform. We conducted comprehensive experiments and analysis on this dataset with a *learning to rank* model that takes different types of features as input, as well as a few state-of-the-art entity linking approaches. Our experimental results show that two types of features that are not available in traditional entity linking: social features and location features, can be very helpful for this task.

1 Introduction

Entity linking is the task of determining the identities of entities mentioned in texts. Most existing studies on entity linking have focused on linking entity mentions to their referred entities in a knowledge base (Cucerzan, 2007; Liu et al., 2013; Ling et al., 2015). However, on social media platforms such as Twitter, Instagram, Yelp, Facebook, etc., the texts produced on them may often mention entities that cannot be found in a knowledge base, but can be found on the platform itself. For example, consider Yelp, a platform where users can write reviews about businesses such as restaurants, hotels, etc., a restaurant review on Yelp may mention another restaurant to compare, which is also likely to be on Yelp but cannot be found in a knowledge base such as Wikipedia. As another example, when people post a photo on a social media platform, their friends may be mentioned in this post if they are also in the photo. Usually, their friends are not included in a knowledge base but may also have accounts on the same platform. Thus for such entity mentions, linking them to an account that is also on the platform is more practical than linking them to a knowledge base.

Performing this kind of entity linking can benefit many applications. For example, on Yelp, we can perform analysis on the comparative sentences in reviews after linking the business mentions in them. The results can be directly used to either provide recommendations for users or suggestions for business owners.

Thus, in this paper, we focus on a new entity linking problem where both the entity mentions and the target entities are within a social media platform. Specifically, the entity mentions are from the texts (which we will refer to as *mention texts*) produced by the users on a social media platform; and these mentions are linked to the accounts on this platform.

It is not straightforward to apply existing entity linking systems that link to a knowledge base to this problem, because they usually take advantage of the rich information knowledge bases provide for the entities. For example, they can use detailed text descriptions, varies kinds of attributes, etc., as features (Francis-Landau et al., 2016; Gupta et al., 2017; Tan et al., 2017), or even additional signals such as the anchor texts in Wikipedia articles (Guo and Barbosa, 2014; Globerson et al., 2016; Ganea et al., 2016). However, on social media platforms, most of these resources or information are either unavailable or of poor quality.

On the other hand, social media platforms also have some unique resources that can be exploited. One that commonly exists on all of them is social

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2023–2032
Brussels, Belgium, October 31 - November 4, 2018.

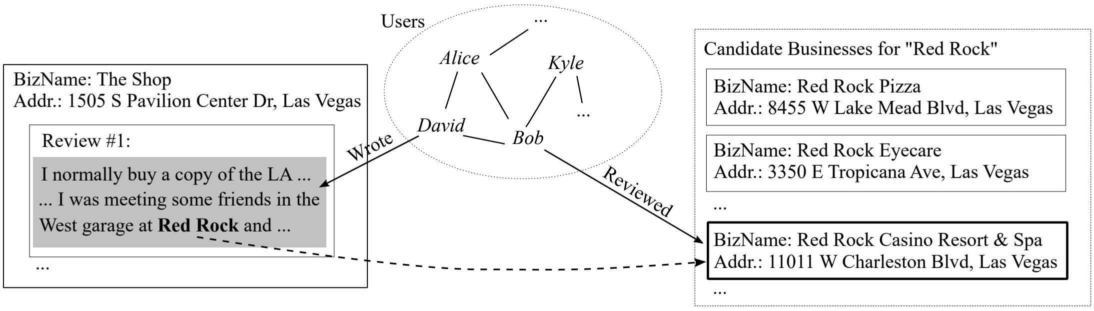

Figure 1: An example of entity linking within the Yelp social media platform. On Yelp, users can have friends which makes it a social network. Users can also write reviews about a business and compare with other businesses.

information, which can be intuitively used in our problem where *mention texts* and target entities may be directly connected by users and their social activities. Other than this, for location-based social media platforms such as Yelp and Foursquare, location information can also be helpful since people are more likely to mention and compare places close to each other.

To study this problem, we construct a dataset based on Yelp, which we name as Yelp-EL. As shown in Figure 1, on Yelp, users can write reviews for businesses and friend other users, and the reviews they write may mention businesses other than the reviewed ones. Thus, reviews, users, and businesses are connected and form a network through users' activities on the platform. In Yelp-EL, we link the business mentions in reviews to their corresponding businesses on the platform. We choose Yelp because other social media platforms such as Facebook and Instagram do not provide open dataset and there can be privacy issues related.

We then study the roles of three types of features in our entity linking problem: social features, location features, as well as conventional features that are also frequently used in traditional entity linking problems. We implemented a *learning to rank* model that takes the above features as input. We conducted comprehensive experiments and analysis on Yelp-EL with this model and also a few state-of-the-art entity linking approaches that we tailored to meet the requirements of Yelp-EL. Experimental results show that both social and location features can improve performance significantly.

Our contributions are summarized as follows.

- We are the first attempt to study the new entity linking problem where both entity mentions and target entities are within a same social media platform.
- We created a dataset based on Yelp to illustrate the usefulness of this problem and use it as a benchmark to compare different approaches.
- We studied both traditional entity linking features and social/location based features that are available from the social media platform, and show that they are indeed helpful for improving the entity linking performance.

The code and data are available at `https://github.com/HKUST-KnowComp/ELWSMP`.

2 Yelp-EL Dataset Construction

In this section we introduce how we create the dataset Yelp-EL based on the Yelp social media platform. We used the Round 9 version of the Yelp challenge dataset[1] to build Yelp-EL. There are 4,153,150 reviews, 144,072 businesses, and 1,029,432 users in this dataset. In order to build Yelp-EL, we first find possible entity mentions in Yelp reviews, and then ask people to manually link these mentions to Yelp businesses if possible.

Ideally, the mentions we need to extract from the reviews should be only those that refer the businesses in Yelp. Unfortunately, there is no existing method or tool that can accomplish this task. In fact, this problem itself is worth studying. Nonetheless, since we focus on entity linking in this paper, we only try to find as many mentions that may refer to Yelp businesses as we can, and then let the annotators decide whether to link this mention to a business. Thus, we use the following two ways to find mentions and then merge their results.

---

[1] https://www.yelp.com/dataset/challenge

#Mentions	#Linked	#NIL	#Disagreement1	#Disagreement2	Agreement%
7,731	1,749	5,117	842	23	88.8%

Table 1: Annotation statistics. “Linked” means the mentions that both annotators link to a same business. “NIL” means the mentions that both annotators think are “unlinkable.” “Disagreement1” means the mentions that are labeled by one annotator as “unlinkable,” but are linked to a business by the other annotator. “Disagreement2” means the mentions that are linked by two annotators to two different businesses.

(1) We use the Standford NER tool (Finkel et al., 2005) to find ordinary entity mentions and filter those that are unlikely to refer to businesses. To do the filtering, we first construct a dictionary which contains entity names that may occur in Yelp reviews frequently but are unlikely to refer to businesses, e.g., city names, country names, etc. Then we run through the mentions found with the NER tool and remove those whose mention strings matches one of the names in the dictionary.

(2) We find all the words/multi-word expressions in reviews that match the name of a business, and output them as mentions.

After extracting the mentions, we obtain the ground-truth by asking annotators to label them. Each time, we show the annotator one review with the mentions in this review highlighted, the annotator then needs to label each of the highlighted mentions. For each mention, we show several candidate businesses whose names match the mention string well. The annotator can also search the business by querying its name and/or location, in case the referred business is not included in the given candidates. We also ask the annotators to label the mention as “unlinkable” when its referred entity is not a Yelp business or it is not an entity mention.

An important issue to note is franchises. There are some mentions that refer to a franchise as a whole, e.g., the mention “Panda Express” in the sentence “If you want something different than the usual Panda Express this is the place to come.” There are also some mentions that refer to a specific location of a franchise. For example, the mention “Best Buy” in “Every store you could possibly need is no further than 3 miles from here, which at that distance is Best Buy” refers to a specific “Best Buy” shop. As a location based social network platform, Yelp only contains businesses for different locations of franchises, not franchises themselves. Thus in these cases, we ask the annotators to link the mentions when they refer to a specific location of a franchise, but label them as “unlinkable” when they refer to a franchise as a whole.

We asked 14 annotators who are all undergraduate or graduate students in an English environment university to perform the annotation. They were given a tutorial before starting to annotate, and the annotation supervisor answered questions during the procedure to ensure the annotation quality. Each review is assigned to two annotators.

The statistics of the annotation results are shown in Table 1. The total agree rate, calculated as $(\#\text{Linked} + \#\text{NIL})/\#\text{Mentions}$, is 88.8%. Most disagreements are on whether to link a mention or not. We checked the data and find that this happens mostly when: they disagree on whether the mention refers to a franchise as a whole or just one specific location; one of the annotators fails to find the referred business. However, when both annotators think the mention should be linked to a business, the disagree rate, calculated as $\#\text{Disagreement2}/(\#\text{Linked} + \#\text{Disagreement2})$, is very low (only 1.3%).

We only use the mentions that both annotators give the same labeling results to build the dataset. As a result, we obtain 1,749 mentions that are linked to a business. These mentions refer to 1,134 different businesses (mentioned businesses) and are from 1,110 reviews. The reviews that contain these mentions are for 967 different businesses (reviewed businesses).

The reviewed businesses are located in 96 different cities and belong to 419 different categories. Note that a business can only locate in one city but may have several different categories. The mentioned businesses are located in 98 different cities and belong to 425 different categories. Figure 2 shows the numbers of reviewed businesses and mentioned businesses in the most popular cities and categories, from where we can see that these mentions have an acceptable level of diversity.

The mentions that can be linked are our focus, but we also include the 5,117 unlinkable mentions in our dataset since they can be helpful for building a complete entity discovery and linking system (Ji et al., 2016).

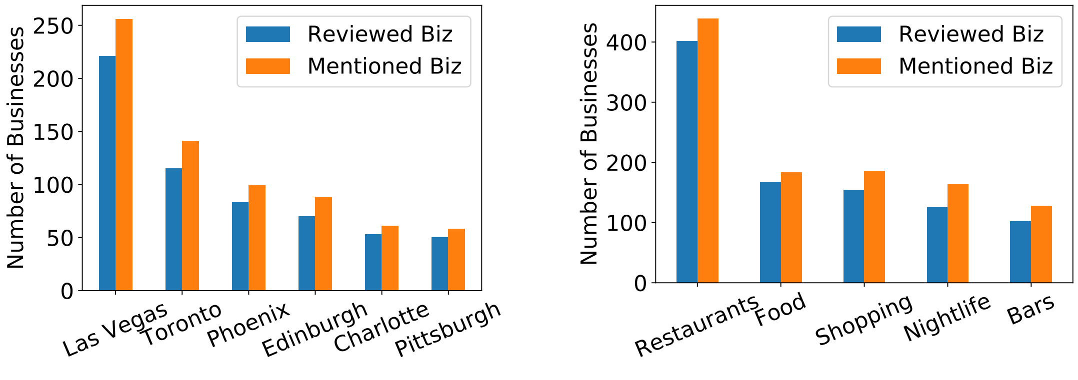

(a) Top Cities

(b) Top Categories

Figure 2: Statistics of the related businesses in Yelp-EL. (a) The number of reviewed businesses in the six most popular cities. (b) The number of mentioned businesses in the five most popular categories. Here, “popular” means having the largest number of businesses in the dataset.

3 Entity Linking Algorithm

In this section, we introduce *LinkYelp*, an entity linking approach we design for Yelp-EL to investigate the new proposed problem. LinkYelp contains two main steps: candidate generation and candidate ranking. The candidate generation step finds a set of businesses that are plausible to be the target of a mention based on the mention string. Afterwards, the candidate ranking step ranks all the candidates and chooses the top ranked one as the target business.

3.1 Candidate Generation

For the first step, candidate generation, we score each business b with $g(m,b) = g_c(m,b) \cdot g_n(s_m, s_b)$ for a mention m, where s_m is the mention string of m, s_b is the name of b. $g_c(m,b)$ equals to a constant value that is larger than 1 (it is set to 1.3 in practice) when the review that contains m is for a business that is located in the same city with b; Otherwise, it equals to 0. g_n is defined as

$$g_n(s_m, s_b) = \begin{cases} 1 & \text{if } s_m \in A(s_b) \\ \text{sim}(s_m, s_b) & \text{Otherwise,} \end{cases} \quad (1)$$

where $A(s_b)$ is the set of possible acronyms for s_b, $\text{sim}(s_m, s_b)$ is the cosine similarity between the TF-IDF representations of s_m and s_b. In practice, $A(s_b)$ is empty when s_b contains less than two words; Otherwise, it contains one string: the concatenation of the first letter of each word in s_b. Then, we find the top 30 highest scored businesses as candidates. This approach has a recall of 0.955 on Yelp-EL.

3.2 Candidate Ranking

Let m be a mention and b be a candidate business of m. We use the following function to score how likely b is the correct business that m refers to:

$$f(m,b) = \boldsymbol{w} \cdot \boldsymbol{\phi}(m,b), \quad (2)$$

where $\boldsymbol{\phi}(m,b)$ is the feature vector for mention-candidate pair m and b, Section 4 describes how to obtain it in detail; $\boldsymbol{w}$ is a parameter vector.

We use a max-margin based loss function to train $\boldsymbol{w}$:

$$J = \frac{1}{|T|} \sum_{<m,b_t,b_c> \in T} \max[0, 1 - f(m,b_t) + f(m,b_c)] + \lambda \|\boldsymbol{w}\|^2, \quad (3)$$

where b_t is the true business mention m refers to; $b_c \neq b_t$ is a corrupted business sample randomly picked from the candidates of m; T is the set of training samples; $\|\cdot\|$ is the l_2-norm; λ is a hyperparameter that controls the regularization strength. We use stochastic gradient descent to train this model.

4 Feature Engineering

We study the effectiveness of three types of features: conventional features, social features, and location features. Among them, conventional features are those that can also be use in traditional entity linking tasks; social features and location features are unique in our problem.

4.1 Conventional Features

Lots of information used in traditional entity linking cannot be found for Yelp businesses, but we try our best to include all such features that can be used in our problem.

For Yelp-EL, we use the following conventional features for a mention m and its candidate business b:

u_1 : The cosine similarity between the TF-IDF representations of the mention string of m and the name of b.

u_2 : Whether the mention string of m is a possible acronym of b's name (i.e., whether it is an element of the set $A(s_b)$ in Equation 1).

u_3 : The popularity of b. Let the number of reviews received by b be n. Then this feature value equals to n/C if n is smaller than a parameter C that's used for normalization, otherwise it equals to 1.

u_4 : The cosine similarity between the TF-IDF representations of the review that contains m and combination of all reviews of b. This feature evaluates how well b fits m semantically.

u_5 : Whether b is the same as the reviewed business. This feature is actually not available in traditional EL, and it is usually not available on other social media platforms either. But it is obviously useful on Yelp-EL. Including it here helps us to see how beneficial social features and location features truly are.

4.2 Social Features

Through the activities of the users on the platform, the users, mentions, reviews and businesses in Yelp-EL form a network where there are different types of nodes and edges. Thus we use Heterogeneous Information Networks (HIN) to model it, and then design meta-path based features to capture the relations between mentions and their candidate businesses. We skip the formal definitions of HIN and meta-path here, readers can refer to (Sun et al., 2011) for detailed introduction. The HIN schema for Yelp-EL is shown in Figure 3.

The following meta-paths are used:

$$\begin{aligned} P1:&\quad M-R-U-R-B\\ P2:&\quad M-R-U-U-R-B\\ P3:&\quad M-R-U-R-B-R-U-R-B \end{aligned}$$

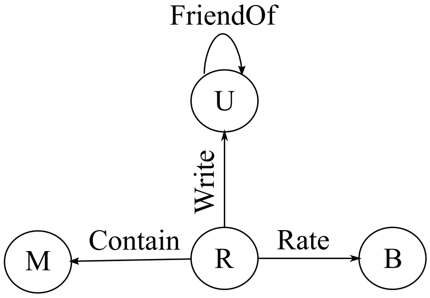

Figure 3: HIN schema of Yelp-EL. M: mention; R: Review; U: user; B: business.

where we denote M for mention, R for Review, U for user, and B for business.

Different meta-paths above capture different kinds of relations between a mention and its candidate entities that are induced by users' social activities. For example, if an instance of $P1$ exists between a mention m and a business b, then m is contained in a review that is written by a user who also reviewed business b. If many such instances of $P1$ exist, then we may assume that m and b are related, which makes it more possible for m to be referring to b.

With the meta-paths above, we use the *Path Count* feature defined in (Sun et al., 2011) to feed into the entity linking model described in Section 3. Given a meta-path P, for mention m and business b, *Path Count* is the number of path instances of P that start from m and end with b. In practice, we normalize this value based on global statistics before feeding it to a model.

4.3 Location Features

Location information commonly exists in location-based social media platforms such as Yelp and Foursquare. Users on platforms such as Twitter and Instagram may also be willing to provide their locations.

Here, we use the following two features for a mention m and its candidate business b:

v_1 : Whether the reviewed business is in the same city as b.

v_2 : The geographical distance between the reviewed business and b. This value is calculated based on the longitude and latitude coordinates of the businesses.

There are still some other location features that can be designed. For example, we can also consider the locations of the other businesses that are reviewed by the user. We only use the above two since we find in our experiments that including them already provides high performance boost.

5 Experiments

5.1 Compared Methods

We compare with a baseline method we name as DirectLink, as well as two existing entity linking methods including the method proposed by (Liu et al., 2013) (which we refer to as ELT) and SSRegu proposed by (Huang et al., 2014).

DirectLink simply links each mention to the corresponding reviewed business. Many business mentions in Yelp reviews actually refer to the business that is being reviewed. This baseline method tells us how many of these mentions there are in Yelp-EL.

ELT collectively links a set of mentions with an objective to maximize local compatibility and global consistence. It achieves this by integrating three types of similarities: mention-entity similarity, entity-entity similarity, and mention-mention similarity. To apply ELT to Yelp-EL, we use the conventional features introduced in Section 4.1 for mention-entity similarities. The path count feature of meta-path B–R–U–R–B is used as entity-entity similarity. For mention-mention similarity, we use two features that are both TF-IDF based cosine similarities, with one between the two mention strings and the other between the reviews that the two mentions belong to.

SSRegu is also a collective approach. It is a graph regularization model that incorporates both local and global evidence through three principals: local compatibility, coreference, and semantic relatedness. SSRegu computes a weight matrix for each of these three principals, and then forms a graph based on the weight matrices and performs graph regularization to rank candidate entities. To apply SSRegu, we need to compute three weight matrices. The weight matrix for local compatibility is based on features extracted from the mention and the candidate entity. In our case, the conventional features are used for computing this matrix. Computing the coreference weight matrix requires to determine whether two corresponding mentions are coreferential. Huang et al. (2014) assume two mentions to be coreferential if their mention strings are the same and there exists at least one meta-path instance of specific patterns between them. In our case, the meta-paths used are M–R–M and M–R–U–R–M. To compute the semantic relatedness weight matrix, we apply the entity-entity similarity used for ELT. Note that SSRegu is a semi-supervised approach and is capable of using unsupervised data, but for fair comparison we do not use this feature here.

Method	Accuracy (mean±std)
DirectLink	0.6684±0.008
ELT	0.8451±0.012
SSRegu	0.7970±0.013
LinkYelp	**0.9034±0.014**

Table 2: Entity linking performance of different methods on Yelp-EL.

ELT and SSRegu are originally proposed to tackle the problem of entity linking for tweets, but their linking target is Wikipedia. Evaluating the performance of these two methods on Yelp-EL shows the difference between their problem and ours.

5.2 Experimental Settings

Throughout our experiments, the hyperparameter λ in Equation (3) is set to 0.001. For each mention, three corrupted samples are random selected for the training with Equation (3). For ELT and SSRegu, the hyperparameters are tuned on the validation set with grid search. The candidate businesses for ELT and SSRegu are also obtained with the method describe in Section 3.1. We run five trials of random split of the linked mentions in the dataset, where each trail uses 60% of the linked mentions as training data and 40% as test data. In each of the training set, we further select 20% as validation set.

Note that we only use linked mentions to evaluate different methods since NIL detection is not our focus, but NIL mentions are utilized in Section 5.6 to build a complete entity linking system.

5.3 Comparison Results

Table 2 shows the entity linking performance of different methods on Yelp-EL. Here, all three types of features described in Section 4 are fed into LinkYelp. Within the compared methods, LinkYelp performs substantially better. This shows that methods carefully designed for traditional entity linking problems may not work so well when applied to entity linking within a social media platform, and this new problem we pro-

Features	All	Restaurants	Nightlife	Shopping	Food	A & E	Bars	E & S	H & T
C	84.05	86.80	79.18	83.15	81.15	71.67	83.49	68.79	63.70
S	79.65	82.92	75.09	82.05	77.69	66.95	80.66	72.25	67.41
L	80.05	85.71	72.70	81.32	85.77	50.21	82.55	49.13	37.78
C+S	86.45	89.29	84.98	85.71	85.00	78.54	87.26	**77.46**	**71.85**
C+L	89.42	93.32	84.98	90.48	88.46	76.39	88.21	72.83	66.67
S+L	85.19	89.29	79.86	87.91	87.31	72.10	85.38	72.25	67.41
C+S+L	**90.34**	**93.79**	**85.67**	**91.94**	**90.00**	**78.97**	**88.68**	76.30	71.11

Table 3: Entity linking accuracy (%) on different categories of businesses with different types of features as input. On the "Features" column, "C," "S," and "L" means conventional, social, and location features respectively. "All" means all the categories combined, i.e., the whole test set; "A & E" means Arts & Entertainment; "E & S" means Event Planning & Services; "H & T" means Hotels & Travel.

pose is worth studying differently from the traditional entity linking problem. The accuracy of DirectLink means that many mentions (about 67%) in Yelp-EL simply refer to the corresponding reviewed businesses. However, this does not mean that our problem is less challenging than traditional entity linking, since simply using the popularity measure of entities can achieve an accuracy of about 82% in the latter task (Pan et al., 2015).

5.4 Ablation Study

We further investigate how the three different types of features described in Section 4 contribute to the final performance of LinkYelp, and how they perform differently in linking mentions that refer to a specific category of Yelp businesses. The results are listed in Table 3. The categories in Table 3 are those that include the largest numbers of businesses in the dataset. Entries in the "All" column in Table 3 are the accuracies on all the categories combined. We can see from this column that both social features and location features are able to improve the performance when combined with conventional features. Location features are relatively more effective than social features, this is because people's activities are mainly restricted to a certain area, so they are more likely to mention businesses that are within this area. But social features are still helpful even when both conventional features and location features are already used, as the best performance is achieved with all the three types of features combined. Moreover, social features can become more important for other social media platforms that do not have location information available.

There are also some interesting findings if we consider the performance on different categories. For example, compared with only using conventional features (row C), incorporating social features (row C+S) provides the largest improvement for Event Planing & Services (e.g., wedding planning, party planning). This matches our intuition because for these kinds of businesses, people tend to be influenced more by their friends and make choices that are socially related. Table 3 also shows that on the categories Event Planning & Services and Hotels & Travel, incorporating location features is not that helpful as it does on other categories. We manually checked the mentions under these two categories that are linked correctly by C+S but incorrectly by C+L. We find that the reasons why incorporating location features fails on these mentions vary from case to case. Two possible reasons are: location information is not helpful to disambiguate a hotel and the shops in this hotel; it also does not work well in disambiguating different locations of a hotel chain that are all not far away from the reviewed business.

5.5 Error Examples

We also manually checked some of the errors made by LinkYelp with all the three types of features as input. A few examples are shown in Table 4. In the first case, since the reviewed business "Jean Philippe Patisserie" is a restaurant, our system tends to find a similar business instead of a hotel. Location features do not help here because Cafe Bellagio has the same location as Bellagio Hotel. The system is also incapable of identifying that "stay at" should be probably followed by a hotel instead of a Cafe. In the second case, the algorithm outputs the reviewed business because it is unable to understand what "the other Second Sole in Rock River" means. The above two examples show that there are still some errors caused by the failure of natural language understanding.

Reviewed Biz:	*Name*: Jean Philippe Patisserie *Addr.*: 3600 S Las Vegas Blvd, Las Vegas
Review:	... Even if you are not staying at the Bellagio, you have to stop by anyway to ...
True Referent:	*Name*: Bellagio Hotel *Addr.*: 3600 S Las Vegas Blvd, Las Vegas
System Prediction:	*Name*: Cafe Bellagio *Addr.*: 3600 S Las Vegas Blvd, Las Vegas
Reviewed Biz:	*Name*: Second Sole Athletic Footwear *Addr.*: 5114 Mayfield Rd, Cleveland
Review:	I did a review of the other Second Sole in Rocky River. This one is in Lyndhurst...
True Referent:	*Name*: Second Sole *Addr.*: 19341 Detroit Rd, Rocky River
System Prediction:	*Name*: Second Sole Athletic Footwear *Addr.*: 5114 Mayfield Rd, Cleveland
Reviewed Biz:	*Name*: Hoot Owl *Addr.*: 4361 W Bell Rd, Phoenix
Review:	This place is a really fun neighborhood bar ... Its tucked away in the Frys parking lot...
True Referent:	*Name*: Fry's Food and Drug *Addr.*: 4315 W Bell Road, Phoenix
System Prediction:	*Name*: Frys *Addr.*: 2626 S 83rd Ave, Phoenix

Table 4: Examples of errors made by LinkYelp. Business mentions are underlined.

Reviewed Biz	Sentence	Mentioned Biz
Name: Burger King *Addr.*: 1194 King St W, Toronto	When you compare it to the McDonald's across the street, the service is way better.	*Name*: McDonald's *Addr.*: 1221 King Street W, Toronto
Name: The Turf Public House *Addr.*: 705 N 1st St, Phoenix	I have to say I like the atmosphere and surroundings of the Turf better than Seamus.	*Name*: Seamus McCaffrey's *Addr.*: 18 W Monroe St, Phoenix

Table 5: Examples of comparative sentences and linked mentions.

Name	City	Stars	#Better
Bacchanal Buffet	Las Vegas	4.0	33
Wicked Spoon	Las Vegas	3.5	24
Cibo	Phoenix	4.5	9
Pizzeria Bianco	Phoenix	4.0	2
XS Nightclub	Las Vegas	4.0	8
Marquee	Las Vegas	3.5	1

Table 6: Comparative study using texts and average ratings. Each row is a pair of two frequently compared businesses. #Better means the number of sentences that claim the corresponding business to be better than the other one.

In the third case, "Fry's Food and Drug" is located at "4315 W Bell Road, Phoenix" which is nearer to the reviewed business "Hoot Owl" located at "4361 W Bell Rd, Phoenix." However, although location information favors the correct business, the others features may contribute more for the system output "Frys" since "Frys" has an exact match of the candidate mention name.

5.6 Comparative Study

In this study, we provide some insight on the possible applications of our task by checking the comparative sentences in Yelp reviews.

First, we find comparative sentences from the whole Yelp review dataset with a simple pattern matching method: we retrieve the sentences that contain one of eight predefined comparison phrases such as "is better," "not as good as," etc. Then we extract the named entity mentions within these sentences and link them to Yelp businesses. A threshold based approach is used to detect NIL mentions (Dalton and Dietz, 2013).

As a result, we get 12,149 comparative sentences from the total 4,153,150 reviews that contains at least one linked mention. Some of the results are shown in Table 5. We can successfully identify both the entity names and their locations on Yelp. We also selected the top three frequently compared pairs and compare with the stars provided by Yelp dataset. From Table 6 we can see that the text comparison is consistent with star ratings.

6 Related Work

The traditional entity linking task of mapping mentions in articles to their corresponding entities in a knowledge base has been studied extensively (Shen et al., 2015; Ling et al., 2015). Various kinds of methods have been studied, e.g., neural network models (Sun et al., 2015; He et al., 2013), generative models (Li et al., 2013), etc. A large group of the existing entity linking approaches are called collective approaches, which are based on the observation that the entities mentioned in a same context are usually related with each other. Thus they usually form entity linking as an optimization problem that tries to maximizes both local mention-entity compatibility and global entity-entity coherence (Han et al., 2011; Nguyen et al., 2016). LinkYelp does not consider global entity-entity coherence as it is not the focus of this paper, but it can be applied to our problem too.

The prevalence of on-line social networks has also motivated researchers to study entity linking in such environments. (Huang et al., 2014; Shen et al., 2013; Liu et al., 2013) proposed methods that are specially designed for linking named entities in tweets. They mainly address the problem that tweets are usually short and informal, while taking advantage of some of the extra information that tweets may provide. For example, (Shen et al., 2013) assumed that each user's tweets have an underlying interest distribution and proposed a graph based interest propagation algorithm to rank the entities. (Huang et al., 2014) also used meta-path on HIN in their entity linking approach, but they only used it to get an indication of whether two mentions are related. Finally, although these studies focused on entity linking for tweets, they still use entities in knowledge bases as the target.

There are a few entity linking studies that do not link mentions to knowledge bases. (Shen et al., 2017) proposed to link entity mentions to an HIN such as DBLP and IMDB. However, their articles are collected from the Internet through searching and thus are not related to the target entities. They also used an HIN based method, but their use is restricted to get the relatedness between different entities. (Lin et al., 2017) studied the entity linking problem where the entities are included in different lists and entities of the same type belong to the same list. They only used this information along with the name of each entity to perform entity linking. Thus their focus is very different from ours.

7 Conclusions

In this paper, we propose a new entity linking problem where both entity mentions and target entities are in a same social media platform. To study this problem, we first create a dataset called Yelp-EL, and then conduct extensive experiments and analysis on it with a learning to rank model that takes three different types of features as input. Through the experimental results, we find that traditional entity linking approaches may not work so well on our problem. The two types of features that are usually not available for traditional entity linking tasks – social features and location features – can both improve the performance significantly on Yelp-EL. Our work can also motivate and enable a lot of downstream applications such as comparative analysis of location based businesses. In the future, we plan to extract more patterns to obtain more comparative sentences, so that we may more accurately demonstrate how useful performing comparative analysis after linking the business mentions can be.

Acknowledgments

This paper was supported by the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong. The experiments and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the affiliated company.

References

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. *Proceedings of EMNLP-CoNLL*, page 708.

Jeffrey Dalton and Laura Dietz. 2013. A neighborhood relevance model for entity linking. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, pages 149–156.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of NAACL*, pages 1256–1261.

Octavian-Eugen Ganea, Marina Ganea, Aurelien Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2016. Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of WWW*, pages 927–938.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of ACL*, volume 1, pages 621–631.

Zhaochen Guo and Denilson Barbosa. 2014. Robust entity linking via random walks. In *Proceedings of CIKM*, pages 499–508.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of EMNLP*, pages 2681–2690.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Proceedings of SIGIR*, pages 765–774.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *Proceedings of ACL*, pages 30–34.

Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji, and Chin-Yew Lin. 2014. Collective tweet wikification based on semi-supervised graph regularization. In *Proceedings of ACL*, pages 380–390.

Heng Ji, Joel Nothman, Hoa Trang Dang, and Sydney Informatics Hub. 2016. Overview of tac-kbp2016 tri-lingual edl and its impact on end-to-end cold-start kbp. *Proceedings of of TAC*.

Yang Li, Chi Wang, Fangqiu Han, Jiawei Han, Dan Roth, and Xifeng Yan. 2013. Mining evidences for named entity disambiguation. In *Proceedings of KDD*, pages 1070–1078.

Ying Lin, Chin-Yew Lin, and Heng Ji. 2017. List-only entity linking. In *Proceedings of ACL*, volume 2, pages 536–541.

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design challenges for entity linking. *TACL*, 3:315–328.

Xiaohua Liu, Yitong Li, Haocheng Wu, Ming Zhou, Furu Wei, and Yi Lu. 2013. Entity linking for tweets. In *Proceedings of ACL*, pages 1304–1311.

Thien Huu Nguyen, Nicolas Fauceglia, Mariano Rodriguez Muro, Oktie Hassanzadeh, Alfio Massimiliano Gliozzo, and Mohammad Sadoghi. 2016. Joint learning of local and global features for entity linking via neural networks. In *Proceedings of COLING*, pages 2310–2320.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of NAACL-HLT*, pages 1130–1139.

Wei Shen, Jiawei Han, Jianyong Wang, Xiaojie Yuan, and Zhenglu Yang. 2017. Shine+: A general framework for domain-specific entity linking with heterogeneous information networks. *IEEE Trans. on Knowl. and Data Eng.*

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.*, 27(2):443–460.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2013. Linking named entities in tweets with knowledge base via user interest modeling. In *Proceedings of KDD*, pages 68–76.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of IJCAI*, pages 1333–1339.

Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB*, 4(11):992–1003.

Chuanqi Tan, Furu Wei, Pengjie Ren, Weifeng Lv, and Ming Zhou. 2017. Entity linking for queries by searching wikipedia sentences. In *Proceedings of EMNLP*, pages 68–77.

Annotation of a Large Clinical Entity Corpus

Pinalkumar Patel
Team Lead, Research
ezDI Inc. Kentucky
`pinal.p@ezdi.us`

Disha Davey
Medical Coding Analyst
ezDI Inc. Kentucky
`disha.d@ezdi.us`

Vishal Panchal
Research Engineer
ezDI Inc. Kentucky
`vishal.p@ezdi.us`

Parth Pathak
Data Scientist
Microsoft
`papatha@microsoft.com`

Abstract

Having an entity annotated corpus of the clinical domain is one of the basic requirements for detection of clinical entities using machine learning (ML) approaches. Past researches have shown the superiority of statistical/ML approaches over the rule based approaches. But in order to take full advantage of the ML approaches, an accurately annotated corpus becomes an essential requirement. Though there are a few annotated corpora available either on a small data set, or covering a narrower domain (like cancer patients records, lab reports), annotation of a large data set representing the entire clinical domain has not been created yet. In this paper, we have described in detail the annotation guidelines, annotation process and our approaches in creating a CER (clinical entity recognition) corpus of 5,160 clinical documents from forty different clinical specialities. The clinical entities range across various types such as diseases, procedures, medications, medical devices and so on. We have classified them into eleven categories for annotation. Our annotation also reflects the relations among the group of entities that constitute larger concepts altogether.

1 Introduction

Corpus annotation is a process of adding interpretive linguistic information to a corpus (Leech, 2004). In the era of increasing trend of machine learning in NLP, annotated data drives the progress of NLP systems in many ways. Fields of NLP like Machine Translation, Information Extraction/Retrieval, Relationship Detection among the entities, depends heavily on an annotated corpus. In the past, many traditional NLP engines have used rule based dictionary lookup methods, with Unified Medical Language System (UMLS) (Lindberg et al., 1993) as a base dictionary, to detect clinical entities. However, these approaches fetch very low recall, due to the fact that the dictionary lookup can never capture all the lexical and linguistic variants of a medical term, and also due to the fact that the clinical documents depend upon a physician's writing style (Pathak et al., 2014).

Over the last few decades, Electronic Medical Records (EMR) have been an integral part of health care. Most of the data consists of a patient's symptoms, procedures being conducted and the medications prescribed to them. This data is mostly available in free text form or semi-structured form and may contain different level of difficulties in parsing this natural text and getting meaningful information. The extensive linguistic study is not available for a clinical domain to the extent it is for the general domain. Therefore, ML approaches are preferred over rule based approaches in the clinical domain. So a resource of the annotated corpus became a necessity to take full advantage of ML approaches. In the recent past, a few attempts have been made to annotate clinical texts. One of the first such attempt was made by (Wang, 2009) on 311 clinical notes from an Intensive Care Unit (ICU) department of the single hospital - Royal Prince Alfred Hospital (RPAH). However, no specific guidelines on how that data were annotated are available. Shared tasks like i2b2 in 2010 (Uzuner et al., 2011) (more than 800 documents), ShARe/CLEF (Suominen et al., 2013) (around 300 documents) in 2013 and SemEval 2014 (Pradhan et al., 2014), 2015 (Elhadad et al., 2015) (around 300 documents) have contributed a lot in increasing the availability of annotated clinical corpora. However, these corpora are focused only on certain type of entities. For example, i2b2 2010 data set has the annotation of three entity types, including test, treatment and disease.

There are other contributions to annotating clin-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2033–2042
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

ical corpora, but depending on the purpose of the clinical research, most of the corpora generated in this domain are very specific to a disease or a disease category or some specialities of hospitals. For example, (Fiszman et al., 2000) annotated chest x-ray reports for automatic identification of acute bacterial pneumonia; (South et al., 2009) manually annotated clinical records to identify phenotypic information for inflammatory bowel disease; (Koeling et al., 2011) have annotated oncology reports on ovarian cancer for symptoms; (Xia and Yetisgen-Yildiz, 2012) have manually annotated the corpus for three different categories - Pneumonia (PNA), Critical Pulmonary Infection Score (CPIS) and critical recommendation on Radiology and ICU Reports; Clinical E-Science Framework (CLEF) corpus (Roberts et al., 2009) have annotated various types of clinical records from a single hospital, Royal Marsden Hospital, Oncology Center. This corpus is restricted to diagnosis, and only considers documents from the patients with neoplasms, that is only a primary diagnosis code in one of the top level sub-categories of ICD-10 Chapter II (neoplasms). As none of the above corpora cover the clinical domain in entirety, we have started building our own CER corpus.

In this paper, we demonstrate our approach of creating the corpus, deciding on annotation guidelines, annotation processes, annotation error analysis and improving the annotation quality with a corpus of 5,160 de-identified clinical documents for 11 different entity types varying from 40 different domains.

2 Corpus Creation

The success of many healthcare IT applications like computer assisted coding (CAC), clinical document improvement (CDI), core/quality measure monitoring are directly proportional to the accuracy of entity recognition and relations among the group of entities. Our aim of creating this corpus was to encapsulate as many entity types as possible, keeping in mind many of such future applications. We have annotated 5,160 de-identified clinical documents for 11 different entity types varying from 40 different domains. All the documents were de-identified using simple rule based approaches which follow safe harbour guidelines before any further use.

Clinical documents are very peculiar. Type of text in the document depends heavily on work types (like admit notes, discharge notes, operative notes, progress notes, etc.), associated medical domains (cardiology, oncology, endocrinology etc.) and varies considerably from physician to physician. So it was very important for us to make sure that we include as many different domains, work types and physicians as we could. So the first step towards corpus creation was to classify documents into different domains. We took around 700,000 documents, from 119 providers (hospitals and specialty clinics), unfortunately not all of these documents had the information regarding its domain and work type. We used a semi-automatic way to find domain related information and were able to classify 236,850 documents from all the documents into 40 different domains. We were also able to capture information regarding sub-specialities and physician expertise for these documents. Table 1 represents a sample domain classification.

Domain	**Docu- -ment Count**	**Sub specialty**	**Physician**
Radiology	84635	2	16
Internal Medicine	15751	3	56
Emergency Medicine	12742	5	28
Cardiology	11555	6	40
Oncology	8325	4	11
Orthopedics	5480	2	19

Table 1: Domain wise classification of the documents

Work Type	**Document Count**
Consultation Reports	18
Operative Report	20
Progress Notes	40
History and Physical	15
Discharge Summary	30
Total	123

Table 2: Sample Document selection of gastroenterology domain

Based on this classification, our domain team prepared a list of important domain and worktype pairs from the whole corpus and on the basis of document distribution over these pairs, we filtered 5,160 documents to annotate which represent 40 different domains and more than 100 worktypes.

For example, Table 2 shows a sample selection of gastroenterology domain.

3 Annotation Guidelines

The first step towards preparing the annotation guideline was to decide on what has to be annotated. The best resource available to make this decision for the clinical domain was UMLS semantic group. The UMLS classifies the bio-medical entities into 133 semantic groups defined as Term Unique Identifiers (TUIs). These semantic groups are very fine-grained. We grouped multiple of these TUIs into different bucket to come up with 11 different types of medical entities. For the purpose of clinical information extraction, it is not necessary to use this much of detail as they would not be of much use in the clinical NLP applications, for example, clinical coding etc.

3.1 Clinical Entity Types

The clinical domain includes medical records like consultation reports, progress notes, history and physical, discharge summary, operative notes. These medical records comprise information about patient's diseases, affected anatomical area, procedures performed to treat the condition, devices used during the procedures and list of medications. It mainly covers the important entity types such as Problem, Anatomical structure, Procedure, Medical device and Medicine respectively. Apart from such information, the medical record also consists of the patient's normal functions, vital signs, lab examination and status of the patient. To cover this information, we have added other entity types Body function, Body measurement, Laboratory data and Finding respectively. Most of the times, entity type like Body measurement is mentioned with its values and to cover these values we have added an entity type named Measurement value which acts like a numerical modifier to add meaning to the entity type. Just a numerical entity has no meaning, so it is necessary that this entity type is always used in relation to an entity type named Body measurement. There are some words used in the clinical domain to add specificity to an entity. Such type of words are mostly adjectives and need a head word without which they have no contextual meaning and cannot be annotated alone. For example the word acute. The word acute alone has no specific meaning, but when it is relates to a head word entity like pain then it adds meaning to the word pain. To cover such information and to maintain the entities as simple and unique as possible, we have added an entity type named Modifier as a separate enhancing entity type. So in all, the gist of the medical record is captured by annotating the documents in our classified entity types.

Problem: The disease conditions which include major problem, disease, symptoms and disorders.

e.g.: Complication of **bleeding, infection, arterial puncture, DVT.**

Finding: Concepts apart from major problem, including abnormal conditions and the minor alteration in the regular condition.

e.g.: This is a 27-year-old female **gravida 4 para 1,** feeling **weak** and **lethargic.**

Procedure: Surgery or other procedures performed to cure or diagnose.

e.g.: This is an 82-year-old female with the history of **appendectomy,** status post **open reduction internal fixation.**

Anatomical Structure: Anatomical sites, cells and organs of the human body.

e.g.: The patient continued to have mild colitis throughout into the **cecum.**

Body Function: Activities carried out by the body to maintain the normal functioning.

e.g.: The patient's **breathing** was normal.

Lab Data: The type of analysis performed on blood, urine, other body substances or tissues to help diagnose or monitor the patient's condition.

e.g.: **AFB test** is negative, **TSH** is 1.1.

Body Measurement: The normal measurement of the body obtained without performing a complex procedure or test.

e.g.: The **weight** is up a couple pounds at 157 pounds, **pulse** is 74.

Measurement Value: Numerical value with its unit, associated with body measurement.

e.g.: The patient's heart rate was **90** and the BP was **140/90.**

Medical Device: Instruments used for the treatment, operation and various medical purposes.

e.g.: **Arterial line catheter** was placed over the **guidewire** without any resistance.

Medicine: A drug used for the treatment or prevention of a disease.

e.g.: Completed antibiotic course of **ceftriaxone.**

Modifier: Any word that adds some specific meaning to an Entity.

e.g.: **Chronic** skin excoriation due to known neurodermatitis.

We have classified our entity types in such a way that the meaning of the medical record is captured properly. Medical terminology is vast in nature, so other entity types except the 11 mentioned types are possible, like gene/variants, lipids, cells, cell lines, steroids, etc. and no doubt they are important entity types, but we do not need to annotate them separately because some of these entities can be classified in the existing entity types and others do not occur frequently in our medical records.

For example, an entity type like Lipid, documented only in laboratory data section is covered by our entity type Laboratory data.

e.g.: Cholesterol levels are high.

In this example, cholesterol is an organic lipid molecule present in the body and generally should be labelled under entity type Lipid. But in the medical records, it is mostly present in the form of a laboratory test. So there is no need to create a new entity type as it would be easily covered under our classified entity type named Laboratory data. We encounter such cases in our data and so entity type like Lipid is not used and if we use, it alters the meaning, which in this case, the system won't understand that it is a lab report and not an organic lipid molecule present in the body.

Another example is an entity type like Steroid, it is documented only in the medicine section which is covered by our entity type Medicine.

e.g.: The patient was prescribed corticosteroids.

"Corticosteroid" is a type of steroid hormone present in the body. But in this case, "corticosteroids" is a medicine which is prescribed and it is annotated under Medicine type. We encounter such cases in our data and entity type like Steroid is covered by our classified Medicine type.

So we have categorized these groups into broader categories of entity types like Problem, Procedure etc. These categories are as mentioned in Table 3 with relevant TUIs from UMLS.

Primary annotation guidelines were prepared by a linguist and an experienced medical coder. Both linguists and medical coder finalized entity types and created its initial descriptive definition with some complex examples. After that, both linguists and coder annotated the same set of documents separately and on the basis of conflicting annotations they sat together and solved the conflicts, then they updated the annotation guidelines to cover these conflict patterns. During this process, we have covered all possible domains and their existing document types in order to cover all the prominent entity patterns. After a few iterations, a well-defined annotation guideline was prepared and we started the actual annotation work. Note that the documents annotated during this process were not included in the corpus.

Entity Type	**Related TUI**
Problem	T046,T047,T048,T049, T050,T191,T037,T019,T184
Finding	T033,T034,T041,T084,T032, T201,T053,T054,T069,T068, T070,T067
Procedure	T059,T060,T061,T065,T058
Medicine	T200,T120,T110,T195,T131
Medical Device	T073,T074,T075,T203,T072, T071
Lab Data	T196,T119
Anatomical Structure	T017,T021,T023,T024, T025,T026,T029,T030
Body Function	T038,T039,T040,T042, T043, T044,T045

Table 3: Entity types with their mappings to TUIs

3.2 Inter-Conceptual Relationships

Relationships may exist between two or more concepts. In such instances, we considered inter-conceptual relationships between limited and frequently used terminologies. Relationships are marked only when relations occur in the same sentence and less frequent relationships like Lab data and problem, Medicine and Problem are rarely mentioned in a single sentence and hence we do not annotate those relations. We have not covered some relations like medicine with its attributes because it unnecessarily increases the complexity of the annotation task and these relations are easy to extract using finite state automata or regex based tools like cTAKES (Savova et al., 2010, 2011). Currently, the relationships are marked for the anatomical structure, measurement value and modifier with their related entities. Anatomical structures get related with three categories like problem, finding and procedure; measurement value gets related to only body measurement and modifier gets related to all other concepts except measurement value.

3.2.1 Relationships with Anatomical Structures

Anatomical structure can be related to problem, procedure and finding. A relationship adds specificity to the concept.

Anatomical structure and problem relationship

This type of relationship helps to understand which part of the anatomical structure is affected by the particular problem. It simplifies the understanding of the problem and its area of effect.

For *e.g.*: she had severe pain in the left ankle.

Here pain is the problem and ankle is the anatomical structure. A relationship can be formed between these two concepts and it makes it easy to understand that pain is in the ankle.

Anatomical structure and finding relationship

This relation helps to understand to which anatomical structure is the finding related.

For *e.g.*: He noted he had felt some tingling and numbness in the left upper chest area.

Here tingling is the finding and chest is the anatomical structure. By linking these two concepts in a relationship we explain that tingling is occurring in the chest.

Anatomical structure and procedure relationship

This relationship explains that the procedure is being done at the given anatomical structure. It is usually used to relate any operation, test, and other such procedures to the organ or body site at which it is being performed.

e.g.: The patient had a CT of the brain.

Here CT is a procedure named Computed Tomography and brain is the anatomical structure. Relating these two concepts simplifies the understanding that CT of the brain was performed.

3.2.2 Relationships of Measurement Values

Measurement values cannot stand alone as only some numerical values do not provide any useful information. So these values need to be related to the concept where they belong to.

Body measurement and Measurement Value relationship

Body measurement is the measurement of basic body parameters like temperature, height, weight, pulse rate, etc. These parameters often have values that can be linked to the measurement in order to provide a complete meaning.

e.g.: Pulse: 80. BP: 110/70. Respirations: 16.

Here, pulse, BP, and respiration are the body measurements and 80, 110/70, and 16 are the measurement values of these respectively. So, forming the relationship between pulse and 80 signifies that the pulse rate is 80 per min, and similarly for all such concepts.

3.2.3 Relationships of Modifier

All the concepts except modifier itself and measurement values can be related to the modifier to add specific meaning to these concepts. Modifiers can never stand alone.

So after classifying the entity types to be annotated and the possible relationships between those entity types as mentioned above, a set of definite principles were documented. Annotators followed these principles while annotating the data. Some directive principles from the guideline are mentioned below.

3.2.4 Directive Principles

Consistency in the annotation is very important to get the good accuracy of systems using ML approaches. Therefore, to achieve good consistency, following simple rules were prescribed for the annotators:

- The entities should be annotated based on the above given entity types and also generate relationship as mentioned above.
- Modifiers and measurement values should not be annotated without any relationship with other entity types.
- Entities from section headers should not be annotated.
- Adjective should be annotated as a modifier only if it is not a part of an abbreviated term instead of the selected text. For example, in “Chronic Hypertension”, ‘Chronic’ should be annotated as Modifier and ‘Hypertension’ should be annotated as Problem but in “Chronic Obstructive Pulmonary Disease” the whole phrase must be marked as Problem, as the phrase has a universally accepted abbreviation COPD. We refer to the UMLS dictionary for the abbreviated terms.
- Mark normal alternation in condition as a finding and abnormal alteration as a problem. i.e. “ST wave changes” is marked as a finding while “ST wave abnormality” as a problem.

- The entities should be annotated depending on the meaning and the context of the sentences because there might be some cases in which the type of an entity changes based on the sentence structure. For example: There was an evidence of “drainage” of pus and “drainage” was carried out. In this case, the word drainage appears twice, in the first half it means the pus is oozing out, so it is tagged as an entity type finding and in the second half it means a procedure named drainage was carried out and so tagged under entity type Procedure.
- There are instances where there are disjoint entities present in the sentence which have a relationship with other entities too. For example “The patient suffered from chest and abdominal pain.” In this case, the word pain is a disjoint entity but it also has a relationship with the chest. So in such cases, we mark chest and abdominal as an anatomical structure respectively and pain as a problem and then generate a relationship between these entities.

4 Annotation Process

There are different ways to annotate a medical corpus, but we embarked on two pre-eminent approaches. The first approach is semi-automated annotation, in this approach, we can choose any pre-existing tools like MetaMap (Aronson, 2001), for CER task and thereafter annotator uses initial annotated results of tool to add, update or delete annotations as required and give the final annotated result. This approach definitely saves some human efforts and takes less time to annotate, however it is difficult to find an efficient and accurate tool. In addition, there is another issue with semi-automated annotation approach, that is, the annotator’s decision may alter after viewing the output of the tool for ambiguous entities. The Second approach is to annotate the document based on just annotation guidelines and based on annotator’s medical knowledge without looking at pre-annotated labels. As quality of annotation is more important than quantity of annotation specifically in medical domain, so we decided to go ahead with second approach.

Protégé (Noy et al., 2001) is an open source annotation and ontology editing tool. This tool was developed by Stanford Center for Biomedical Informatics Research. It allows us to add entity types with a description and assign color. Using this tool we can generate relationships between different entity types. The tool supports plain text, CSV and TSV format and so it is easy to load a medical document. The annotated data can be exported to a structured XML file format which makes the data widely accepted, machine readable and easy to use for any system.

Four annotators with a background in microbiology and biochemistry were involved in the annotation process. The medical document consists of major entities like the problem, procedure, anatomical structure and lab data which the annotators are well aware of because microbiology and biochemistry experts have a good knowledge about the anatomy and physiology as well as information about many ailments and procedures and laboratory tests. The queries were solved using UMLS dictionary and internal discussion with experienced medical experts and AAPC certified coders. The annotators are initially given a brief introduction about the importance of annotation and its applications and the impact of the annotated data on the clinical NLP. After that, the linguist and medical coders walk them through the set of guidelines which should be followed while annotating the corpus and present a demo of the annotating tool so that they get a brief knowledge of the tool. Initially, some documents were annotated in front of them so they become familiar with the process of annotation and tool. For the first few sets of documents (one set contains 30 documents), we divided the team of annotators into two pairs and made them annotate the same documents together. After completion of these sets, we changed the pairs of annotators. After 3 such iterations, each annotator had worked with all other annotators. This process was performed to make sure that all the annotators have the same level of understanding towards the task. The annotators are instructed to keep a note of problems faced while annotating data and discuss it with the medical coders regarding the medical annotation and with engineers regarding the technical difficulties and the functioning of the tool. While annotating ambiguous entity, UMLS metathesaurus was used as a reference. However, annotators found many inconsistencies in the UMLS. So in case of any disagreement among the annotators, the final decision about what to be annotated was taken by a mutual consensus arrived after a discussion with the whole team. Note that the documents annotated during this process were not included in the corpus. After that, all the annotators were given a distinct set of documents in which 20% of the documents were the same between two annotators. We chose these same documents considering the distribution of worktypes over the corpus. These documents were used to calculate the kappa score (inter-annotator agreement). After annotation, the kappa score is calculated. The errors are discussed and solved by the annotators and then the data becomes useful. It took one year for the annotators to complete this task, in which the initial one month was spent on the training process.

4.1 Inter-annotator Agreement Calculation

Annotation is a mentally taxing task, and so annotators occasionally miss to annotate some of the medical entities, especially when a document contains a large number of them. The annotators are free to mark the entities according to their medical knowledge and as a result of that, some disagreement arises. An inter-annotator agreement is an important quality measure. The Cohen's kappa coefficient is used to estimate the agreement between the annotators. The Cohen's kappa equation is calculated with the following formula:

$$K = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e} \tag{1}$$

where, p_o is the relative observed agreement among annotators, and p_e is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly saying each category. If the annotators are in complete agreement then = 1. If there is no agreement among the raters other than what would be expected by chance (as given by pe), $K \leq 0$ (Koeling et al., 2011). The initial kappa score was 68.01% calculated on the set of 30 documents. The aim was to achieve good kappa score up to 95% to minimize the conception gap between the annotators. The kappa score is obtained after each set of annotation and the annotators are made to sit together and review the whole document with their own annotation results. They internally discuss why some entities are annotated and missed or ambiguously annotated. After discussion, they come to a conclusion and make a note of the changes which helps to decrease the conception gap. Using these notes, they are clear about what to annotate and what not to. Using structured guidelines and proper classification of entity types and relations, we achieved a 96.89% kappa score for the entity and relationship annotation.

5 Corpus Statistics

Using the mentioned process, we have created a large annotated corpus of the medical domain for clinical entity and relationships which covers 5,160 clinical documents with 398,568 sentences. Out of these sentences, 190,188 sentences contained one or more medical entities which were annotated as concepts in the corpus, hence concept density over sentences is 47.72%. These 398,568 sentences vary in length from 1 token to 150+ tokens as shown in Figure 3. The corpus has average 9.59 tokens per sentence with total 3,825,465 tokens across the corpus. Out of these 3,825,465 tokens, 600,550 tokens annotated as concepts in the corpus, hence concept density over token is 15.70%. An EMR record consists of a gist of the patient's conditions, procedure and tests carried out, medications prescribed etc. But, along with that, there is a lot of insignificant information present in the record like hospital name and address, patient's demographics which are not healthcare entities. Apart from all these, an EMR contains information documented using definite templates and so a lot of sentence are generally not important and common in every medical record and they reflect in annotated token density. The frequency of annotation for each relation and entity type are detailed in Figure 1 and 2 respectively. There are in total 443,328 annotated concepts in the corpus. These annotated concepts result in an average length of 1.35 tokens per concept. The highest frequency concepts are Problem, Procedure, Medication, Anatomical Structure and Modifier which account for 78.81% of the data. The remaining 21.19% concepts are distributed into 6 rare categories. For the relation annotation, the corpus contains total 119,968 relations, out of which modifier - anatomical structure cover 30,018 relations, modifier with other entity types except anatomical structure cover 49,677 relations, anatomical structure with other entity types except modifier cover 35,401 relations and body measurement - measurement value cover 4,872 relations.

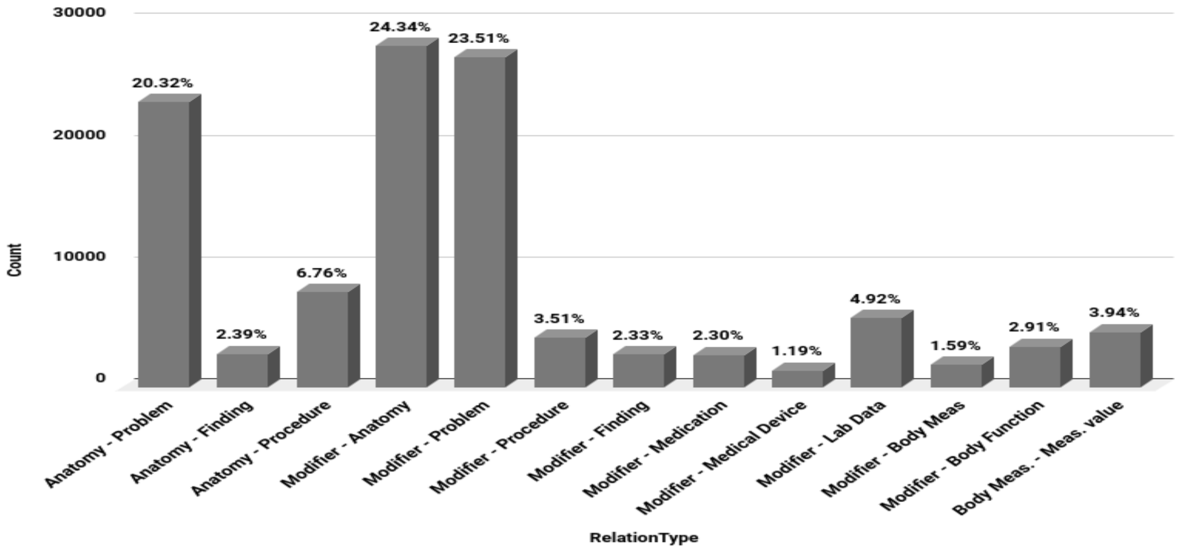

Figure 1: Frequencies of relation types

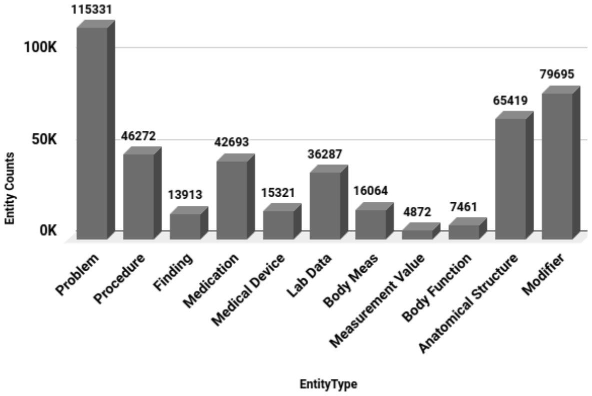

Figure 2: Frequencies of entity types

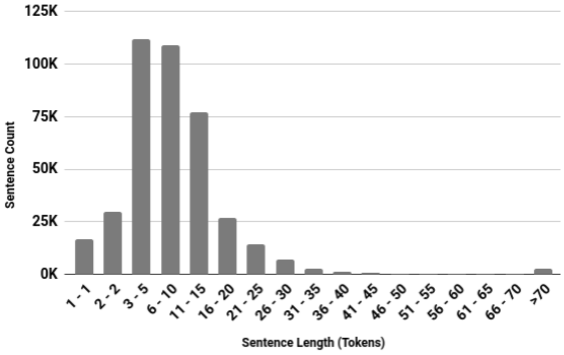

Figure 3: Sentence distribution over token counts

5.1 Initial Experiment with CRF

Initially, we used Conditional Random Fields (CRF) which is a well-known and a proven approach to detect continuous entities using CRF++ (Kudo, 2010) toolkit. Feature selection is the key task for accurate CRF model. We used feature sets as mentioned below (Pathak et al., 2015)

- Bag of words features, prefix, suffix
- Orthographic features, binary (true/false) like whole word capital, first char capital, numeric values, dates, words contains hyphen or slash, medical units (mg, gram, ltr, etc)
- Grammatical features like
  - Parts of Speech (PoS) (Choudhary et al., 2014), chunk, consistency parser (Oinam et al., 2018); all developed and designed in-house for clinical NLP
  - Head of the phrases
  - Stemming of the words
- Dictionary features, binary (true/false) was used to check whether the word is present in the medical dictionary or not.
- Stop words and word embedding id from word2vec trained on 2 lakh clinical documents
- Section header and document type information and sentence cluster id

Table 4 shows the initial accuracy, calculated using the Perl script provided by the CoNLL 2000 task (Tjong Kim Sang and Buchholz, 2000), on CRF using a BIO format with the combination of different features on 5-fold validation. We are able

Feature Set	Precision	Recall	F-Measure
Token	92.32	86.00	89.05
Orthographic	92.22	86.26	89.30
POS and Chunk	91.92	87.94	89.89
Dictionary	91.91	87.43	89.61
Consistency Parse	92.04	87.32	89.62
POS-Chunk and Orthographic	91.91	88.04	89.93
POS-Chunk and Dictionary	91.82	88.77	90.27
POS-Chunk, Consistency Parse and Dictionary	91.65	88.90	90.25
Above + Stemmer, Word Embedding Id	92.00	90.26	91.12
All	92.12	91.05	91.58

Table 4: CER task accuracy using CRF with different feature set

to get 91.58% f-measure with 92.12% precision and 91.05% recall using all features. We obtained a good performance using CRF which shows the quality of the annotation over the definite corpus.

6 Conclusion

In the domain of clinical NLP, there is a paucity of good and large annotated corpus. Research shows that some amount of data has been annotated according to different purposes and applications. Available annotated corpora are not found covering the clinical domain in entirety. In this paper, we have described the creation of a corpus of the clinical domain annotated with clinical entities and their inter-conceptual relationships. We have manually annotated the clinical corpus comprising of 5,160 documents, 398,568 sentences according to the guidelines created in-house.

Acknowledgements

We acknowledge the contribution of former annotators namely Meera Panchal, Namrata Srivastava, Yvonne Christian and Pranita Joshi, who provided insight and expertise that greatly helped in building the corpus. We would also like to thank Sagar Soni for his assistance with the functioning of the protégé tool.

References

Alan R Aronson. 2001. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association.

Narayan Choudhary, Parth Pathak, Pinal Patel, and Vishal Panchal. 2014. Annotating a large representative corpus of clinical notes for parts of speech. In *Proceedings of LAW VIII-The 8th Linguistic Annotation Workshop*, pages 87–92.

Noémie Elhadad, Sameer Pradhan, Sharon Gorman, Suresh Manandhar, Wendy Chapman, and Guergana Savova. 2015. Semeval-2015 task 14: Analysis of clinical text. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 303–310.

Marcelo Fiszman, Wendy W Chapman, Dominik Aronsky, R Scott Evans, and Peter J Haug. 2000. Automatic detection of acute bacterial pneumonia from chest x-ray reports. *Journal of the American Medical Informatics Association*, 7(6):593–604.

Rob Koeling, John Carroll, Rosemary Tate, and Amanda Nicholson. 2011. Annotating a corpus of clinical text records for learning to recognize symptoms automatically.

Taku Kudo. 2010. Crf++: Yet another crf toolkit (2005). *Available under LGPL from the following URL: http://crfpp. sourceforge. net.*

Geoffrey Leech. 2004. Developing linguistic corpora: a guide to good practice adding linguistic annotation. *Lancaster University*.

Donald AB Lindberg, Betsy L Humphreys, and Alexa T McCray. 1993. The unified medical language system. *Methods of information in medicine*, 32(04):281–291.

Natalya F Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W Fergerson, and Mark A Musen. 2001. Creating semantic web contents with protege-2000. *IEEE intelligent systems*, 16(2):60–71.

Nganthoibi Oinam, Diwakar Mishra, Pinal Patel, Narayan Choudhary, and Hitesh Desai. 2018. A treebank for the healthcare domain. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 144–155.

Parth Pathak, Pinal Patel, Vishal Panchal, Narayan Choudhary, Amrish Patel, and Gautam Joshi. 2014. ezdi: A hybrid crf and svm based model for detecting and encoding disorder mentions in clinical notes. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 278–283.

Parth Pathak, Pinal Patel, Vishal Panchal, Sagar Soni, Kinjal Dani, Amrish Patel, and Narayan Choudhary. 2015. ezdi: a supervised nlp system for clinical narrative analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 412–416.

Sameer Pradhan, Noémie Elhadad, Wendy Chapman, Suresh Manandhar, and Guergana Savova. 2014. Semeval-2014 task 7: Analysis of clinical text. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 54–62.

Angus Roberts, Robert Gaizauskas, Mark Hepple, George Demetriou, Yikun Guo, Ian Roberts, and Andrea Setzer. 2009. Building a semantically annotated corpus of clinical texts. *Journal of biomedical informatics*, 42(5):950–966.

Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.

Guergana K Savova, Janet E Olson, Sean P Murphy, Victoria L Cafourek, Fergus J Couch, Matthew P Goetz, James N Ingle, Vera J Suman, Christopher G Chute, and Richard M Weinshilboum. 2011. Automated discovery of drug treatment patterns for endocrine therapy of breast cancer within an electronic medical record. *Journal of the American Medical Informatics Association*, 19(e1):e83–e89.

Brett R South, Shuying Shen, Makoto Jones, Jennifer Garvin, Matthew H Samore, Wendy W Chapman, and Adi V Gundlapalli. 2009. Developing a manually annotated clinical document corpus to identify phenotypic information for inflammatory bowel disease. In *BMC bioinformatics*, volume 10, page S12. BioMed Central.

Hanna Suominen, Sanna Salanterä, Sumithra Velupillai, Wendy W Chapman, Guergana Savova, Noemie Elhadad, Sameer Pradhan, Brett R South, Danielle L Mowery, Gareth JF Jones, et al. 2013. Overview of the share/clef ehealth evaluation lab 2013. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 212–231. Springer.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

Yefeng Wang. 2009. Annotating and recognising named entities in clinical notes. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, pages 18–26. Association for Computational Linguistics.

Fei Xia and Meliha Yetisgen-Yildiz. 2012. Clinical corpus annotation: challenges and strategies. In *Proceedings of the Third Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM'2012) in conjunction with the International Conference on Language Resources and Evaluation (LREC), Istanbul, Turkey*.

Visual Supervision in Bootstrapped Information Extraction

Matthew Berger
Vanderbilt University, University of Arizona
matthew.berger@vanderbilt.edu

Ajay Nagesh
University of Arizona
ajaynagesh@email.arizona.edu

Joshua A. Levine
University of Arizona
josh@email.arizona.edu

Mihai Surdeanu
University of Arizona
msurdeanu@email.arizona.edu

Hao Helen Zhang
University of Arizona
hzhang@math.arizona.edu

Abstract

We challenge a common assumption in active learning, that a list-based interface populated by informative samples provides for efficient and effective data annotation. We show how a *2D scatterplot* populated with *diverse* and *representative* samples can yield improved models given the same time budget. We consider this for bootstrapping-based information extraction, in particular named entity classification, where human and machine jointly label data. To enable effective data annotation in a scatterplot, we have developed an embedding-based bootstrapping model that learns the distributional similarity of entities through the patterns that match them in a large data corpus, while being discriminative with respect to human-labeled and machine-promoted entities. We conducted a user study to assess the effectiveness of these different interfaces, and analyze bootstrapping performance in terms of human labeling accuracy, label quantity, and labeling consensus across multiple users. Our results suggest that supervision acquired from the scatterplot interface, despite being noisier, yields improvements in classification performance compared with the list interface, due to a larger quantity of supervision acquired.

1 Introduction

One strategy for mitigating the cost of supervised learning in information extraction (IE) is to bootstrap extractors with light supervision from a few provided examples (or seeds). Most typical bootstrapping methods (Yarowsky, 1995; Collins and Singer, 1999; Abney, 2007; Carlson et al., 2010; Gupta and Manning, 2014, 2015, inter alia) are iterative in nature, and suffer from semantic drift: as the learning advances, the task often drifts semantically into a related but different space, e.g., from learning women names into learning flower names (Komachi et al., 2008; Yangarber, 2003). In such cases, a human-in-the-loop to help guide bootstrapping through active learning (AL) (Settles, 2012) can be highly beneficial.

In this work, we challenge the common assumption made for AL methods in the context of IE: a visual interface that shows a *list* of samples ranked by their *informativeness* to the classifier is effective for building classifiers that minimize human annotator time (Dalvi et al., 2016; He and Grishman, 2015). We argue that this is an inefficient form of acquiring supervision from humans. Instead, we propose a *two-dimensional (2D) scatterplot* interface (rather than the one-dimensional (1D) list), where the examples to be annotated are selected by their capacity to *cluster together* (rather than by their informativeness to the classifier). We demonstrate that our approach leads to more data being annotated, and better overall performance for the model being learned.

In particular, we focus on the task of bootstrapped named entity classification (NEC), where a classifier is trained to label named entities with their corresponding category. For example, such an algorithm starts with a few examples of labeled names, e.g., "Barack Obama" as `PERSON`, from which it learns representative patterns, e.g., the pattern "`@ENTITY , former president`", which are then used to label other names in future iterations. Unlike traditional bootstrapping, our approach receives supervision in two ways: from the seed examples (as shown above), but also from human labels through an active learning step that is inserted after each bootstrapping iteration. To facilitate the clustering of examples in the annotation interface, we propose a semi-supervised NEC approach that learns custom embeddings for the entities being classified (§4). We select entities that are *diverse* and *representative* of the embedding's data distribution, and project them into a 2D visual encoding of the data via a scatterplot using

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2043–2053
Brussels, Belgium, October 31 - November 4, 2018.

dimensionality reduction (§5).

The resulting scatterplot interface enables the user to label a larger quantity of entities, at the expense of label noise from mixed-category clusters and a potentially less-informative sampling criterion. To better understand this space, we conducted a user study to compare the effectiveness of the scatterplot interface compared to the traditional list that contains examples selected by informativeness (§7). Through our study, we arrive at the following takeaways:

- We observe that in configurations that used the scatterplot interface, annotators indeed labeled considerably more examples, but at an accuracy slightly lower than in the list interface. Despite the lower accuracy, the scatterplot interface generally yields better classifiers. In other words, the volume of annotations matters just as much as quality for the classifier performance.
- We find that a consensus model of users can mitigate noise, but must preserve a certain quantity of annotated data. A consensus model for the list interface that conservatively estimates labels *reduces* performance despite highly accurate labels due to the small amount of annotations. In contrast, the same model for the scatterplot interface yields higher label noise, but more annotations within the same budget of time and gives the best performance.

2 Related Work

Information extraction (IE) techniques commonly assume that the human supervision comes in the form of knowledge bases of facts disconnected from supporting text, as in the case of distant supervision (Mintz et al., 2009), or provides a light amount of supervision up front, as in the case of bootstrapping (Angeli et al., 2015). Common techniques for bootstrapping are to use rules for incrementally classifying entities (Collins and Singer, 1999) or to use syntactic (He and Grishman, 2015) and semantic (Gupta and Manning, 2014, 2015) contextual features. However, such approaches suffer from semantic drift, as previously discussed.

Considering a human-in-the-loop for IE has the potential to mitigate drift and greatly benefit performance, yet the challenge lies in minimizing human effort. The work of Angeli et al. (2014) show how to use active learning to improve distantly supervised relation extraction techniques (Surdeanu et al., 2012) through humans labeling informative relations. Werling et al. (2015) use Bayesian decision theory to minimize human cost and maximize accuracy for named entity recognition. For certain IE tasks, however, human supervision can be very noisy and thus counterproductive, especially from crowds, thus previous work has shown the importance of how to pose tasks for humans in providing labels (Liu et al., 2016), as well as automatically distinguishing simple labeling tasks from expert tasks in crowd-based task assignments (Wang et al., 2017). Our work shares a similar view of human supervision for IE, yet we instead study the impact of the annotation interface on the overall performance.

The role of the human-in-the-loop for topic modeling has also been extensively explored. For instance Smith et al. (2018) consider the types of modifications that one can provide to a built topic model (Hu et al., 2014) to make the topics more meaningful, while also studying the downstream human factor implications. Furthermore, prior work has also considered how different visual representations of topics impact a human's understanding of topic semantics (Smith et al., 2017). Closely related is the technique of Poursabzi-Sangdeh et al. (2016), where they highlight how different visual representations can have an impact on the effectiveness and efficiency of human labeling for document classification, comparing standard list interfaces with topic-grouped lists. Our method instead considers how 2D scatterplot interfaces, via embedding-based techniques, capture semantics for the purposes of providing labels in an IE task.

The visual analytics community has also investigated the role of visual interfaces and interaction tools for annotating data in supervised learning. For instance Heimerl et al. (2012) enable interactive labeling for document classification by visualizing unlabeled documents based on classifier uncertainty and document diversity. The technique of Höferlin et al. (2012) jointly visualizes unlabeled data and the classifier model, and allows the user to both label data points and directly modify model parameters. Closely related to our method is the work of Bernard et al. (2018) which compares active learning, via list interfaces, with in-

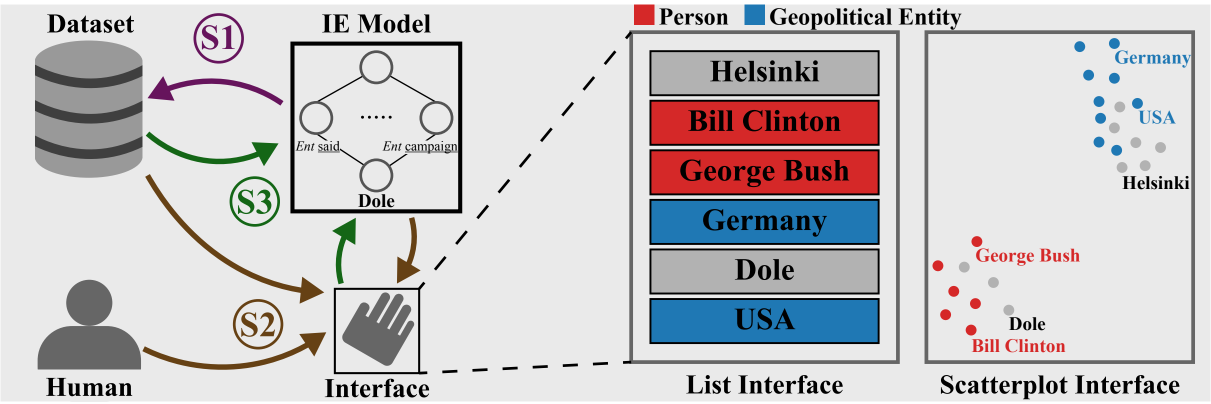

Figure 1: We show the workflow of our study (left): **(S1)** the Information Extraction model first automatically labels entities and updates its parameters, **(S2)** the human then labels entities through a given visual interface, and **(S3)** the model is updated based on the provided set of human labels. We study the effectiveness of two different interfaces (right): a list, and a 2D scatterplot of entities.

teractive visual labeling. However, their method is focused on acquiring labels solely from humans, whereas our method studies the interplay between self-labeling and human labels in a visualization context.

3 Human-Machine Workflow for Bootstrapping

We first describe the general workflow on which our study is based. We consider bootstrapping where both the human and the machine label entities *in tandem*, following the setup described in Fu and Grishman (2013) . More specifically, we consider the following iterative process (c.f. Fig. 1):

- (**S1**) The model automatically classifies entities, adds them as labeled data, and updates its parameters, as detailed in §4.
- (**S2**) The human interacts with a visual interface, driven by the model and the current set of labeled and unlabeled entities. The result of this step is a set of entities labeled by the human. We define this step a **round of labeling**, or just **round**.
- (**S3**) The model then updates its parameters given the human-labeled entities.

In contrast with typical bootstrapping, which interleaves entity promotion with model updates, here the human has the opportunity to label data. The intent is to *guide bootstrapping*: human annotations should steer bootstrapping towards learning the proper concepts, and away from semantic drift.

Yet, in this process we would like to *minimize human effort* while *maximizing bootstrapping performance*, where we define *effort* as the total time spent annotating data. In particular, the primary focus of our study is on step (**S2**), and how the user's interactions with the interface impacts these considerations. We consider two different types of interfaces for this purpose, shown on the right side of Fig. 1: list interfaces, and scatterplot interfaces. In particular, both interfaces utilize the state of the model to decide on what to show to the user. In the case of the list-based interface, we perform *uncertainty sampling* with respect to the model, as a way to maximize the information obtained from the user. For the scatterplot interface, the user interacts with a subset of *diverse* and *representative* entities through a 2D projection. Here we wish to see if more efficient *groupwise labeling* of perceived clusters in the scatterplot results in a more efficient and effective labeling process.

Of course, a critical piece to our study is the bootstrapping model itself. There exists a large body of work in bootstrapping, as previously discussed, and one possibility is to use an existing technique for our work. For our learning scenario, a bootstrapping technique should satisfy several criteria:

- Be efficient to update, in order to minimize user latency with the interface;
- Incorporate user supervision to ensure a discriminative representation;
- Be suitable for visual exploration.

Please note that traditional bootstrapping techniques such as rule-based methods (Collins and Singer, 1999) fail to meet all criteria. In particular, they are not suitable for visual exploration because there is no clear way to represent the semantic proximity of rules or of the concepts being learned. We next describe how to address these challenges through embedding-based bootstrapping.

4 Embedding-Based Bootstrapping

Our bootstrapping technique is based on neural language models (Mikolov et al., 2013), in particular, semi-supervised embedding-based bootstrapping techniques (Valenzuela-Escárcega et al., 2018). Unlike other word embedding algorithms, our approach measures distributional similarity of *entities* (rather than words) with respect to *patterns* (rather than context words). We define a pattern as a small sequence of words that surround an entity, up to ±4 words to the left/right of the entity under consideration. For instance, in the phrases "**John** <u>said</u> <u>on</u> <u>Saturday</u>" and "**John** <u>told</u> <u>reporters</u>" the patterns "<u>said</u> <u>on</u> <u>Saturday</u>" and "<u>told</u> <u>reporters</u>" are suggestive of the category Person for entity **John**.[1] We observe that entities of a given category, e.g. Person, are likely to have common pattern distributions. This observation drives our method for learning entity and pattern embeddings. Furthermore, as we will see, this method permits efficient updates, and it is suitable for visualization due to the geometric representation of entities and patterns.

More specifically, assume that we have extracted a set of entities E and patterns P in a given text corpus C. We associate each entity $e \in E$ and pattern $p \in P$ with *embeddings* $\mathbf{x}_e$ and $\mathbf{x}_p$, respectively, with $\mathbf{x}_e, \mathbf{x}_p \in \mathbb{R}^d$. To satisfy the above form of distributional similarity, we utilize the Skip Gram model (Mikolov et al., 2013) and seek entity embeddings to be close to their embeddings of observed patterns through maximizing:

$$SG = \sum_{(e,p) \in C_p} [\log(\sigma(\mathbf{x}_e^\top \mathbf{x}_p)) + \sum_{n \in N} \log(\sigma(-\mathbf{x}_e^\top \mathbf{x}_n))], \tag{1}$$

where (e, p) corresponds to an entity-pattern occurrence from the corpus set C_p, n represents a *negative* pattern, sampled from the unigram distribution of all patterns N (Levy and Goldberg, 2014) [2], and σ is the sigmoid function. Intuitively, this forces an entity's embedding to be similar to embeddings of its matched patterns from the corpus, and dissimilar to random pattern embeddings.

A disadvantage with the Skip Gram model is that it might fail to be discriminative, since it does not utilize category labels. Thus, we introduce an objective term that seeks to bring entities that belong to the same category to have similar embeddings, and entities that belong to different categories to be far apart in their embeddings. We realize this using large-margin metric learning (Cui et al., 2016; Sohn, 2016), minimizing:

$$LM = \sum_{(a,b,c) \in E_l} \lfloor s(\mathbf{x}_a, \mathbf{x}_c) - s(\mathbf{x}_a, \mathbf{x}_b) + M \rfloor_+, \tag{2}$$

where (a, b, c) represents a triplet of entities, such that a and b belong to the same category, while c belongs to a different category, and the function s is the cosine similarity between entity embeddings. The set E_l is a subset of entities from E that have been assigned categories so far, as provided by the bootstrapper (**S1**) or the human (**S2**). Intuitively, entities from dissimilar categories should be positioned in the embedding space such that their cosine similarity is at least a margin M from any pair of entities of the same category.

We combine the Skip Gram objective with the metric learning objective to obtain:

$$B = LM - SG. \tag{3}$$

The objective B can be viewed as a form of semi-supervised representation learning, where from a sparse set of labeled entities, we wish to learn entity representations that are similar should they have patterns in common (SG), while simultaneously ensuring that embeddings are discriminative with respect to categories (LM). Step (**S3**) minimizes this objective at every round of bootstrapping via stochastic gradient descent, given the current set of labeled entities E_l.

4.1 Promoting Entities

We use the learned embedding to automatically promote unlabeled entities to categories, as discussed in step (**S1**). We use the normalized entity embeddings as features, and build a multinomial

---

[1] In this work we use surface patterns, but the proposed algorithm is agnostic to the types of patterns used.

[2] In initial ablation studies, we found that this strategy was about as effective as negative sampling from *all* patterns, but significantly faster, which is necessary for interactivity.

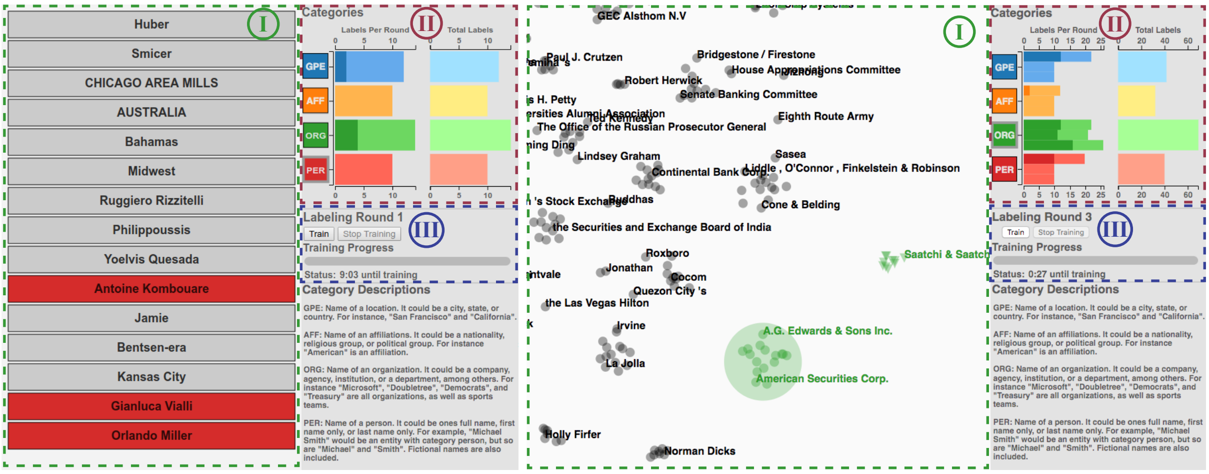

GPE: Name of a location. It could be a city, state, or country. For instance, "San Francisco" and "California".

AFF: Name of an affiliations. It could be a nationality, religious group, or political group. For instance "American" is an affiliation.

ORG: Name of an organization. It could be a company, agency, institution, or a department, among others. For instance "Microsoft", "Doubletree", "Democrats", and "Treasury" are all organizations, as well as sports teams.

PER: Name of a person. It could be ones full name, first name only, or last name only. For example, "Michael Smith" would be an entity with category person, but so are "Michael" and "Smith". Fictional names are also included.

GPE: Name of a location. It could be a city, state, or country. For instance, "San Francisco" and "California".

AFF: Name of an affiliations. It could be a nationality, religious group, or political group. For instance "American" is an affiliation.

ORG: Name of an organization. It could be a company, agency, institution, or a department, among others. For instance "Microsoft", "Doubletree", "Democrats", and "Treasury" are all organizations, as well as sports teams.

PER: Name of a person. It could be ones full name, first name only, or last name only. For example, "Michael Smith" would be an entity with category person, but so are "Michael" and "Smith". Fictional names are also included.

(a) List Interface

(b) Scatterplot Interface

Figure 2: We study two different interfaces for humans to label entities, a list interface (a) and a 2D scatterplot interface (b). In both, the user selects entities in the main view (I), based on an assigned category (II), wherein we also show the total amount of labels the user, and the machine, has thus far labeled. In (III) we show the current status of training/labeling, and the option to initiate/stop training.

logistic regression model over the given set of labeled entities, trained to predict entity categories. For each category, we then promote the most confident entities to the category. To compute confidence, we treat the model's predictions as category probabilities, take the entropy over this distribution as a measure of unconfidence, and promote entities with the lowest entropy.

5 Supervision Interfaces

We now turn to the visual interface through which humans provide supervision (step **S2**). We distinguish the interfaces by how entities are *sampled*, *presented* to the user, and each interface's set of *interactions* for labeling.

5.1 List Interface

Sampling. We sample entities through sorting them by confidence, as defined in §4.1, and sample the most unconfident entities. This form of uncertainty sampling is common in list-based interfaces (Angeli et al., 2014; Poursabzi-Sangdeh et al., 2016), as a measure of informativeness for updating the model (Settles, 2012).

Presentation. We next show the 15 most uncertain entities in a 1D list-based visual interface (Fig. 2a **(I)**).

Interactions. The user labels entities by first selecting their desired category (Fig. 2a **(II)**), followed by clicking on the entity in the main display. We also allow the user to select multiple entities at once, for a given category. As the user labels entities, we repopulate the display with the next set of most unconfident entities.

5.2 Scatterplot Interface

Sampling. For the scatterplot interface, we aim to sample entities that are beneficial for the model, while also ensuring the user can efficiently label entities through groupwise selection. Uncertainty sampling, though potentially informative, can lead to projections that are challenging for the user in performing groupwise labeling, as we experimentally verified that entities with high classification uncertainty are unlikely to group together. To address this, we sample entities based on how they are distributed in the embedding space, to provide us a *diverse* and *representative* sampling (Xu et al., 2007). More specifically, we first perform k-means on the entities' normalized embeddings, with $k = 40$, and sample across clusters to give diversity. To ensure a representative sampling, kernel density estimation is performed within each cluster's entities, and the entity with highest density, along with its nearest neighbors in the embedding space, are selected. The number of neighbors sampled in each cluster is proportional to the cluster size, to ensure balance across clusters.

Presentation. We next perform a 2D projection of 500 sampled entities using t-SNE (Maaten and Hinton, 2008), visually encoding each entity by a filled circle (Fig. 2b **(I)**). To provide context with respect to entities labeled by the human or machine, we jointly project the unlabeled sampled entities, and a subset of the entities previously labeled (drawn as triangles), either by the human annotator or promoted with high-confidence by the machine.

Interactions. The user labels entities by first selecting a category (Fig. 2b **(II)**), followed by using a circular brush to label groups of entities. The user can adjust the brush's radius, as well as change the view through panning and zooming. We dynamically filter the text labels for entities based on the zoom level to reduce clutter, thus the user can observe a high-level overview of the space of entities when zoomed out, and observe more details upon zooming in. This gives the annotator the chance to perform groupwise annotations by jointly considering cluster structure and a sparse set of text labels from the cluster, and labeling all entities at once should the annotator decide the group of entities belongs to a single category.

5.3 Training

Common to both interfaces, user interactions are interleaved with model updates. During training, the list interface is refreshed with the top 15 most informative samples every 3 *training epochs*, or passes over the data, while for the scatterplot the samples and their 2D positions are updated. To ensure temporal coherence for those entities that persist between updates, we employ dynamic t-SNE (Rauber et al., 2016). The user can opt to stop training if they observe little change occurring between snapshots (Fig. 2 **(III)**).

6 User Study

We conducted a user study to investigate the effectiveness of the different interfaces. We recruited 10 participants for our study: the median age was 22, the minimum and maximum respectively 19 and 41, 5 participants self-reported as being "somewhat knowledgeable" of machine learning, 4 with "no knowledge", and 1 identifying as an "expert". We used a within-subject design, where the first interface presented to the participant was selected at random to mitigate potential priming effects. For each interface, a set of instructions was first presented, followed by a brief tutorial where the user must label a small set of seed entities – 10 per category. These seed entities, labeled with ground truth rather than user labels, initialize the bootstrapping model, so that participants start off with identical embeddings. After the instructions, the participant then performs 10 rounds of labeling, where they may label entities for up to 1 minute per round. After each round of labeling, the bootstrapper promotes 10 entities to each category, and then performs 30 epochs of training.

Dataset. We use the Ontonotes dataset (Weischedel et al., 2013), limited to the 4 categories that have the most frequently mentioned entities, in order to make the labeling task manageable, yet still nontrivial, for participants. These categories are people (PER), organizations (ORG), geopolitical entities (GPE), and nationalities as well as religious/political affiliations (AFF), resulting in 6,567, 6,199, 1,617, and 422 entities per category, respectively.

Bootstrapping Details. It is critical to ensure that bootstrapping training minimizes user latency, while not sacrificing performance (c.f. Eq. 3). To strike this balance, we set the number of negative patterns sampled for each entity to 10 (Levy and Goldberg, 2014), the embedding dimension d to 100, perform hard negative triplet mining (Cui et al., 2016) to form the loss on triplets likely to violate the margin, and set the margin M to 0.4. Experimentally, we found these settings allowed training to converge to a good solution after 30 training epochs, and that each epoch took no longer than 1 second on average.

7 Results

We analyze the results in terms of three forms of evaluation: *bootstrapping performance*, as determined by the entities promoted during the course of the user study, *extrapolation*, wherein we let bootstrapping proceed to promote entities after the joint human-machine labeling has completed, and *consensus*, where we combine the set of entities annotated by participants within the different interfaces. We also analyze a typical user's labeling, and corresponding machine performance.

7.1 Bootstrapping Performance

We first look at the effectiveness of promoted entities during the course of each participant's interactions. Fig. 3(a) shows bootstrapping accuracy averaged over all users for the list and scatterplot. We also compare to a baseline of traditional bootstrapping, where no human labels are considered, averaged over 10 trials. Note this is simi-

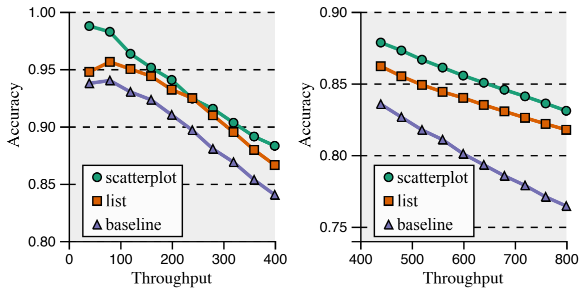

(a) Bootstrapping (b) Extrapolation

Figure 3: Performance averaged over all users for bootstrapping during user study (a) and extrapolation (b), for both interfaces and a baseline that does not use human labels. The green, orange, and purple plots are scatterplot, list, and baseline, respectively.

	scatterplot	list	baseline
bootstrapping	88.8	87.2	85.1
extrapolation	84.7	83.5	78.6

Table 1: We show performance in terms of recall, computed per-category and averaged across categories, for the baseline, list and scatterplot interfaces. This is shown for the last round of bootstrapping (c.f. Fig. 3a) and extrapolation (c.f. Fig. 3b).

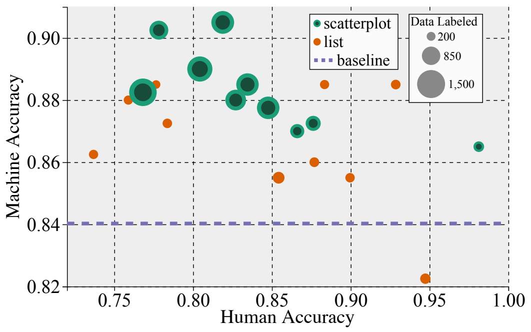

(a) Bootstrapping

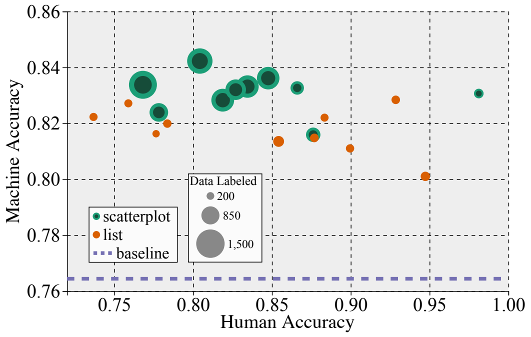

(b) Extrapolation

Figure 4: Bootstrapping performance for user study (a), and extrapolation (b) for individual participants, evaluated at throughputs (number of entities promoted) of 400 and 800, respectively. Each circle is a participant whose color indicates the interface (green, orange, and purple are scatterplot, list, and baseline, respectively); x-axis is participant labeling accuracy; y-axis is machine accuracy; and circle size encodes the number of annotated labels (large circle indicates large number of annotated labels).

lar to Valenzuela-Escárcega et al. (2018), which demonstrates improved performance across a set of bootstrapping techniques, and thus represents a competitive baseline. We find that the performance of both interfaces outperform this baseline, and that the scatterplot outperforms the list: a two-sample Welch's t-test concludes statistical significance (p=0.05), as well as for a paired t-test measured within participants (p=0.03). In Table 1 we also show recall, computed per-category and averaged over categories, at the last round of bootstrapping and similarly find the scatterplot outperforms the list.

Better insight between the interfaces can be gained by looking at individual user performance. Fig. 4(a) shows a plot of each participant's performance (y-axis) for both interfaces, as a function of their labeling accuracy (x-axis) and the total number of labels provided (size of each circle). Note that the 3 best-performing models come from the scatterplot interface, *even at the expense of a lower labeling accuracy*. This suggests that the *number of labels* can counter the *noise in labeling*, compared to a list where we may have potentially more informative entities labeled more accurately, but much fewer entities annotated.

7.2 Extrapolation

We next look at how bootstrapping continues to learn, when starting from all of the annotations provided by the corresponding human annotator as well as the entities promoted by the machine. We term this configuration extrapolation. This analysis indirectly measures how much noise crept into the available annotations, by measuring the performance of the classifier trained on this data.

We repeated this experiment over 10 different trials, promoting 10 entities per category in each round, and take the average accuracy, see Fig. 3(b) for the performance averaged over all users for

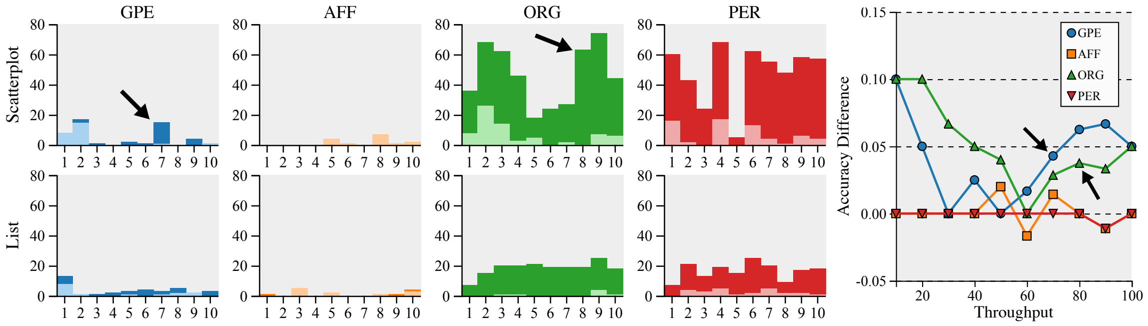

(a) Human Labeling Accuracy

(b) Interface Accuracy

Figure 5: Labeling accuracy and bootstrapping performance for a typical participant. In (a) we show per-category annotations, where the x-axis is the round number and y-axis is the number of entities labeled. Colors of higher lightness indicate mislabeled entities. For example, approximately 40% of the user's annotations in round 2 for ORG in the scatterplot were incorrect. In (b) we show the difference in bootstrapping performance between the scatterplot and list interfaces. For instance, scatterplot yields a model 5% more accurate than list for GPE in round 10. Note the correlation between human labeling (left) and bootstrapping performance (right), highlighted by the arrows.

the interfaces, as well as the aforementioned baseline. Observe that the baseline performs progressively worse as a function of throughput, indicating that labels provided by the human annotators help to prevent drift. We also find the results to be stronger compared with §7.1; namely, a two-sample Welch's t-test concludes statistical significance (p=0.002) and similarly for a paired t-test within participants (p=0.007). The recall performance in Table 1 also confirms the performance gains of the scatterplot. Supporting these results, Fig. 4(b) shows that the scatterplot interface for users that labeled a large amount of entities generally outperform those where fewer entities were labeled. This experiment suggests that the best strategy to control for semantic drift is to aim for an interface that yields *many* annotations at *reasonable* accuracy, rather than few, higher-quality ones.

7.3 Individual User Labeling

We show labeling accuracy and bootstrapping accuracy for a participant that has similar performance to the average (c.f. Fig. 3(a)). In Fig. 5(a) we show the number of entities labeled across categories, where bars of higher color lightness encode the number of incorrect labels, and in (b) we plot the difference in bootstrapping performance between the scatterplot and the list. We observe that the class imbalance of the Ontonotes dataset tends to manifest in labeling as well, where ORG and PER are typically labeled the most. We also see a correlation between accurate human labeling and bootstrapping performance (e.g. the arrows pointing at GPE and ORG for rounds 7 and 8, respectively). This highlights an advantage of exploratory visual interfaces for labeling, such as a scatterplot, where the user can search for clusters of a particular category, e.g., in the case of GPE there may not exist many entities of this type in the list interface.

7.4 Consensus

Given the noise inherent in a single user's annotations, we last analyze whether techniques to *combine* labels across a set of users can help reduce label noise and improve performance. To this end, for the scatterplot and list interface we combine all annotations, and consider two different types of consensus methods: 1) **Union:** we take the union of all annotated entities, choosing an entity's label at random during conflicts; and 2) **Majority Vote (MV):** for conflicting entities we select the most voted category across users, discarding entities that have only been annotated once or have ties across users. We then seed bootstrapping with the resulting set of labels, run 20 rounds of promotion, and take the average performance of 10 trials.

Fig. 6(a-b) compares each method across the different interfaces, while Table 2 shows accuracy and total number of labels for the consensus methods. We also include performance, labeling accuracy, and label totals averaged across individual users, where for each user we seed bootstrapping

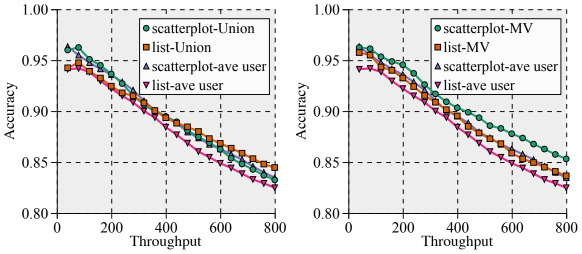

(a) Union (b) Majority Vote

Figure 6: Accuracy of different consensus methods, compared across the two interfaces, as well as average performance across individual users.

with their full set of annotations. For **Union**, we observe that despite the large number of labels, the scatterplot performance is comparable to average individual performance, suggesting that in this case, the volume of labels does not counter the noise in the data. For the list, we can see a gain in performance, which is likely attributed to the cleaner annotations from participants. For **MV**, however, we find that the scatterplot performs the best, whereas the list performs worse compared with its **Union** counterpart, now having approximately 3x less labels. This emphasizes the interplay between *accuracy* and *label quantity*, even despite the more informative labels provided through the list interface, and supports our previous observation that balancing the volume of annotations with their quality yields the best annotation strategy.

8 Discussion

We acknowledge some limitations in our technique and user study. A drawback to our sampling scheme for the scatterplot is that it does not utilize classifier informativeness. In initial experiments, we found that uncertainty sampling led to 2D projections that were challenging to groupwise label. This result is intuitive: data points classified with lowest confidence will naturally have low-quality embeddings, yielding poor clusters. Thus we can not perform a consistent comparison of the same sampling criterion used between the interfaces. For future work we will investigate active learning techniques that are relevant to the learning task, while still permitting efficient labeling.

Interface	Union		MV		Ave. User	
	Acc	Total	Acc	Total	Acc	Total
List	82.7	1463	95.4	543	85.1	217
Scatterplot	77.6	3503	92.5	1959	82.3	788

Table 2: Consensus accuracies and label amounts across the interfaces. The last column is average user accuracy and label amount.

In this work we restricted our study to entity classification, one of many types of IE tasks. Our approach should generalize to other IE tasks, however, provided these tasks exhibit certain structure. Specifically, user annotations in a task should amount to labeling of data instances, and the data instances defined by the task should be able to be perceived in a 2D scatterplot. For instance, coreference resolution and relationship extraction fit both criteria. Assessing the effectiveness of visual supervision for these tasks is outside of the scope of this paper, however, and we will consider these studies in future work.

Overall, our user study clearly highlights the importance of visual interfaces in acquiring supervision for semi-supervised information extraction. We demonstrated that, when compared to the traditional list interface, the scatterplot allows a larger volume of annotations to be created at reasonable accuracy, yielding better classifiers. We believe this finding will influence active learning, in terms of sampling criteria and the interplay between AL and visual interfaces.

9 Acknowledgements

This work was supported by funding from an NSF TRIPODS grant (NSF-1749858), and funding from the Bill and Melinda Gates Foundation HBGDki Initiative. Mihai Surdeanu disclose a financial interest in lum.ai. This interest has been disclosed to the University of Arizona Institutional Review Committee and is being managed in accordance with its conflict of interest policies.

References

Steven Abney. 2007. *Semisupervised Learning for Computational Linguistics*, 1st edition. Chapman & Hall/CRC.

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. 2014. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1556–1567.

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Chaganty, Jason Bolton, Melvin Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D Manning. 2015. Bootstrapped self training for knowledge base population. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*.

Jürgen Bernard, Marco Hutter, Matthias Zeppelzauer, Dieter Fellner, and Michael Sedlmair. 2018. Comparing visual-interactive labeling with active learning: An experimental study. *IEEE transactions on visualization and computer graphics*, 24(1):298–308.

Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Yin Cui, Feng Zhou, Yuanqing Lin, and Serge Belongie. 2016. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1153–1162.

Bhavana Dalvi, Sumithra Bhakthavatsalam, Chris Clark, Peter Clark, Oren Etzioni, Anthony Fader, and Dirk Groeneveld. 2016. IKE - an interactive tool for knowledge extraction. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016*, pages 12–17.

Lisheng Fu and Ralph Grishman. 2013. An efficient active learning framework for new relation types. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 692–698.

Sonal Gupta and Christopher D Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*, pages 98–108.

Sonal Gupta and Christopher D. Manning. 2015. Distributed representations of words to guide bootstrapped entity classifiers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Yifan He and Ralph Grishman. 2015. Ice: Rapid information extraction customization for nlp novices. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 31–35, Denver, Colorado. Association for Computational Linguistics.

Florian Heimerl, Steffen Koch, Harald Bosch, and Thomas Ertl. 2012. Visual classifier training for text document retrieval. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2839–2848.

Benjamin Höferlin, Rudolf Netzel, Markus Höferlin, Daniel Weiskopf, and Gunther Heidemann. 2012. Inter-active learning of ad-hoc classifiers for video visual analytics. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 23–32. IEEE.

Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2014. Interactive topic modeling. *Machine learning*, 95(3):423–469.

Mamoru Komachi, Taku Kudo, Masashi Shimbo, and Yuji Matsumoto. 2008. Graph-based analysis of semantic drift in espresso-like bootstrapping algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1011–1020, Stroudsburg, PA, USA. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Angli Liu, Stephen Soderland, Jonathan Bragg, Christopher H Lin, Xiao Ling, and Daniel S Weld. 2016. Effective crowd annotation for relation extraction. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 897–906.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Forough Poursabzi-Sangdeh, Jordan Boyd-Graber, Leah Findlater, and Kevin Seppi. 2016. Alto: Active learning with topic overviews for speeding label induction and document labeling. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1158–1169.

Paulo E Rauber, Alexandre X Falcão, and Alexandru C Telea. 2016. Visualizing time-dependent data using dynamic t-sne. *Proc. EuroVis Short Papers*, 2(5).

Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.

Alison Smith, Varun Kumar, Jordan Boyd-Graber, Kevin Seppi, and Leah Findlater. 2018. Closing the loop: User-centered design and evaluation of a human-in-the-loop topic modeling system. In *23rd International Conference on Intelligent User Interfaces*, pages 293–304. ACM.

Alison Smith, Tak Yeon Lee, Forough Poursabzi-Sangdeh, Jordan Boyd-Graber, Niklas Elmqvist, and Leah Findlater. 2017. Evaluating visual representations for topic understanding and their effects on manually generated labels. *Transactions of the Association of Computational Linguistics*, 5(1):1–15.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Marco A Valenzuela-Escárcega, Ajay Nagesh, and Mihai Surdeanu. 2018. Lightly-supervised representation learning with global interpretability. *arXiv preprint arXiv:1805.11545*.

Chenguang Wang, Alan Akbik, Yunyao Li, Fei Xia, Anbang Xu, et al. 2017. Crowd-in-the-loop: A hybrid approach for annotating semantic roles. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1913–1922.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Keenon Werling, Arun Tejasvi Chaganty, Percy S Liang, and Christopher D Manning. 2015. On-the-job learning with bayesian decision theory. In *Advances in Neural Information Processing Systems*, pages 3465–3473.

Zuobing Xu, Ram Akella, and Yi Zhang. 2007. Incorporating diversity and density in active learning for relevance feedback. In *European Conference on Information Retrieval*, pages 246–257. Springer.

Roman Yangarber. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Learning Named Entity Tagger using Domain-Specific Dictionary

Jingbo Shang †* **Liyuan Liu** †* **Xiaotao Gu** † **Xiang Ren** ♯ **Teng Ren** ‡ **Jiawei Han** †
† University of Illinois at Urbana-Champaign, Urbana, IL, USA
♯ University of Southern California, Los Angeles, CA, USA
‡ CooTek Inc., Shanghai, China
†`{shang7, ll2, xiaotao2, hanj}@illinois.edu` ♯`xiangren@usc.edu` ‡`teng.ren@cootek.cn`

Abstract

Recent advances in deep neural models allow us to build reliable named entity recognition (NER) systems without handcrafting features. However, such methods require large amounts of manually-labeled training data. There have been efforts on replacing human annotations with distant supervision (in conjunction with external dictionaries), but the generated noisy labels pose significant challenges on learning effective neural models. Here we propose two neural models to suit noisy distant supervision from the dictionary. First, under the traditional sequence labeling framework, we propose a revised fuzzy CRF layer to handle tokens with multiple possible labels. After identifying the nature of noisy labels in distant supervision, we go beyond the traditional framework and propose a novel, more effective neural model AutoNER with a new `Tie or Break` scheme. In addition, we discuss how to refine distant supervision for better NER performance. Extensive experiments on three benchmark datasets demonstrate that AutoNER achieves the best performance when only using dictionaries with no additional human effort, and delivers competitive results with state-of-the-art supervised benchmarks.

1 Introduction

Recently, extensive efforts have been made on building reliable named entity recognition (NER) models without handcrafting features (Liu et al., 2018; Ma and Hovy, 2016; Lample et al., 2016). However, most existing methods require large amounts of manually annotated sentences for training supervised models (e.g., neural sequence models) (Liu et al., 2018; Ma and Hovy, 2016; Lample et al., 2016; Finkel et al., 2005). This is particularly challenging in specific domains, where domain-expert annotation is expensive and/or slow to obtain.

To alleviate human effort, distant supervision has been applied to automatically generate labeled data, and has gained successes in various natural language processing tasks, including phrase mining (Shang et al., 2018), entity recognition (Ren et al., 2015; Fries et al., 2017; He, 2017), aspect term extraction (Giannakopoulos et al., 2017), and relation extraction (Mintz et al., 2009). Meanwhile, open knowledge bases (or dictionaries) are becoming increasingly popular, such as WikiData and YAGO in the general domain, as well as MeSH and CTD in the biomedical domain. The existence of such dictionaries makes it possible to generate training data for NER at a large scale without additional human effort.

Existing distantly supervised NER models usually tackle the entity span detection problem by heuristic matching rules, such as POS tag-based regular expressions (Ren et al., 2015; Fries et al., 2017) and exact string matching (Giannakopoulos et al., 2017; He, 2017). In these models, every unmatched token will be tagged as non-entity. However, as most existing dictionaries have limited coverage on entities, simply ignoring unmatched tokens may introduce false-negative labels (e.g., "prostaglandin synthesis" in Fig. 1). Therefore, we propose to extract high-quality out-of-dictionary phrases from the corpus, and mark them as potential entities with a special "unknown" type. Moreover, every entity span in a sentence can be tagged with multiple types, since two entities of different types may share the same surface name in the dictionary. To address these challenges, we propose and compare two neural architectures with customized tagging schemes.

We start with adjusting models under the traditional sequence labeling framework. Typically, NER models are built upon conditional random

---

[*] Equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2054–2064
Brussels, Belgium, October 31 - November 4, 2018.

fields (CRF) with the `IOB` or `IOBES` tagging scheme (Liu et al., 2018; Ma and Hovy, 2016; Lample et al., 2016; Ratinov and Roth, 2009; Finkel et al., 2005). However, such design cannot deal with multi-label tokens. Therefore, we customize the conventional CRF layer in LSTM-CRF (Lample et al., 2016) into a Fuzzy CRF layer, which allows each token to have multiple labels without sacrificing computing efficiency.

To adapt to imperfect labels generated by distant supervision, we go beyond the traditional sequence labeling framework and propose a new prediction model. Specifically, instead of predicting the label of each single token, we propose to predict whether two adjacent tokens are tied in the same entity mention or not (i.e., broken). The key motivation is that, even the boundaries of an entity mention are mismatched by distant supervision, most of its inner ties are not affected, and thus more robust to noise. Therefore, we design a new `Tie or Break` tagging scheme to better exploit the noisy distant supervision. Accordingly, we design a novel neural architecture that first forms all possible entity spans by detecting such ties, then identifies the entity type for each span. The new scheme and neural architecture form our new model, AutoNER, which proves to work better than the Fuzzy CRF model in our experiments.

We summarize our major contributions as

- We propose AutoNER, a novel neural model with the new `Tie or Break` scheme for the distantly supervised NER task.
- We revise the traditional NER model to the Fuzzy-LSTM-CRF model, which serves as a strong distantly supervised baseline.
- We explore to refine distant supervision for better NER performance, such as incorporating high-quality phrases to reduce false-negative labels, and conduct ablation experiments to verify the effectiveness.
- Experiments on three benchmark datasets demonstrate that AutoNER achieves the best performance when only using dictionaries with no additional human effort and is even competitive with the supervised benchmarks.

We release all code and data for future studies[1]. Related open tools can serve as the NER module of various domain-specific systems in a plug-in-and-play manner.

2 Overview

Our goal, in this paper, is to learn a named entity tagger using, and only using dictionaries. Each dictionary entry consists of 1) the surface names of the entity, including a canonical name and a list of synonyms; and 2) the entity type. Considering the limited coverage of dictionaries, we extend existing dictionaries by adding high-quality phrases as potential entities with unknown type. More details on refining distant supervision for better NER performance will be presented in Sec. 4.

Given a raw corpus and a dictionary, we first generate entity labels (including unknown labels) by exact string matching, where conflicted matches are resolved by maximizing the total number of matched tokens (Etzioni et al., 2005; Hanisch et al., 2005; Lin et al., 2012; He, 2017).

Based on the result of dictionary matching, each token falls into one of three categories: 1) it belongs to an entity mention with one or more known types; 2) it belongs to an entity mention with unknown type; and 3) It is marked as non-entity.

Accordingly, we design and explore two neural models, Fuzzy-LSTM-CRF with the modified `IOBES` scheme and AutoNER with the `Tie or Break` scheme, to learn named entity taggers based on such labels with unknown and multiple types. We will discuss the details in Sec. 3.

3 Neural Models

In this section, we introduce two prediction models for the distantly supervised NER task, one under the traditional sequence labeling framework and another with a new labeling scheme.

3.1 Fuzzy-LSTM-CRF with Modified IOBES

State-of-the-art named entity taggers follow the sequence labeling framework using `IOB` or `IOBES` scheme (Ratinov and Roth, 2009), thus requiring a conditional random field (CRF) layer to capture the dependency between labels. However, both the original scheme and the conventional CRF layer cannot handle multi-typed or unknown-typed tokens. Therefore, we propose the modified `IOBES` scheme and Fuzzy CRF layer accordingly, as illustrated in Figure 1.

Modified IOBES. We define the labels according to the three token categories. 1) For a token

---

[1] `https://github.com/shangjingbo1226/AutoNER`

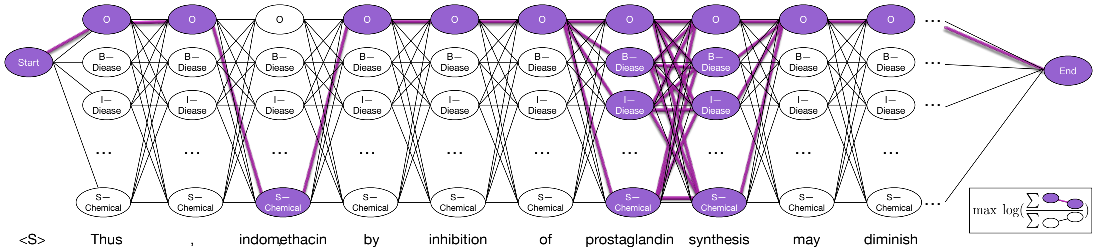

Figure 1: The illustration of the Fuzzy CRF layer with modified `IOBES` tagging scheme. The named entity types are {`Chemical`, `Disease`}. "indomethacin" is a matched `Chemical` entity and "prostaglandin synthesis" is an unknown-typed high-quality phrase. Paths from `Start` to `End` marked as purple form all possible label sequences given the distant supervision.

marked as one or more types, it is labeled with all these types and one of {`I`, `B`, `E`, `S`} according to its positions in the matched entity mention. 2) For a token with unknown type, all five {`I`, `O`, `B`, `E`, `S`} tags are possible. Meanwhile, all available types are assigned. For example, when there are only two available types (e.g., Chemical and Disease), it has nine (i.e., $4 \times 2 + 1$) possible labels in total. 3) For a token that is annotated as non-entity, it is labeled as `O`.

As demonstrated in Fig. 1, based on the dictionary matching results, "indomethacin" is a singleton `Chemical` entity and "prostaglandin synthesis" is an unknown-typed high-quality phrase. Therefore, "indomethacin" is labeled as `S-Chemical`, while both "prostaglandin" and "synthesis" are labeled as `O`, `B-Disease`, `I-Disease`, . . ., and `S-Chemical` because the available entity types are {`Chemical`, `Disease`}. The non-entity tokens, such as "Thus" and "by", are labeled as `O`.

Fuzzy-LSTM-CRF. We revise the LSTM-CRF model (Lample et al., 2016) to the Fuzzy-LSTM-CRF model to support the modified `IOBES` labels.

Given a word sequence $(X_1, X_2, \ldots, X_n)$, it is first passed through a word-level BiLSTM (Hochreiter and Schmidhuber, 1997) (i.e., forward and backward LSTMs). After concatenating the representations from both directions, the model makes independent tagging decisions for each output label. In this step, the model estimates the score P_{i,y_j} for the word X_i being the label y_j.

We follow previous works (Liu et al., 2018; Ma and Hovy, 2016; Lample et al., 2016) to define the score of the predicted sequence, the score of the predicted sequence $(y_1, y_2, \ldots, y_n)$ is defined as:

$$s(X, y) = \sum_{i=0}^{n} \Phi_{y_i,y_{i+1}} + \sum_{i=1}^{n} P_{i,y_i} \qquad (1)$$

where, $\Phi_{y_i,y_{i+1}}$ is the transition probability from a label y_i to its next label y_{i+1}. Φ is a $(k + 2) \times (k + 2)$ matrix, where k is the number of distinct labels. Two additional labels `start` and `end` are used (only used in the CRF layer) to represent the beginning and end of a sequence, respectively.

The conventional CRF layer maximizes the probability of the only valid label sequence. However, in the modified `IOBES` scheme, one sentence may have multiple valid label sequences, as shown in Fig. 1. Therefore, we extend the conventional CRF layer to a fuzzy CRF model. Instead, it maximizes the total probability of all possible label sequences by enumerating both the `IOBES` tags and all matched entity types. Mathematically, we define the optimization goal as Eq. 2.

$$p(y|X) = \frac{\sum_{\tilde{y} \in Y_{possible}} e^{s(X,\tilde{y})}}{\sum_{\tilde{y} \in Y_X} e^{s(X,\tilde{y})}} \qquad (2)$$

where Y_X means all the possible label sequences for sequence X, and $Y_{possible}$ contains all the possible label sequences given the labels of modified `IOBES` scheme. Note that, when all labels and types are known and unique, the fuzzy CRF model is equivalent to the conventional CRF.

During the training process, we maximize the log-likelihood function of Eq. 2. For inference, we apply the Viterbi algorithm to maximize the score of Eq. 1 for each input sequence.

3.2 AutoNER with "Tie or Break"

Identifying the nature of the distant supervision, we go beyond the sequence labeling framework and propose a new tagging scheme, `Tie or Break`. It focuses on the ties between adjacent tokens, i.e., whether they are tied in the same entity mentions or broken into two parts. Accordingly, we design a novel neural model for this scheme.

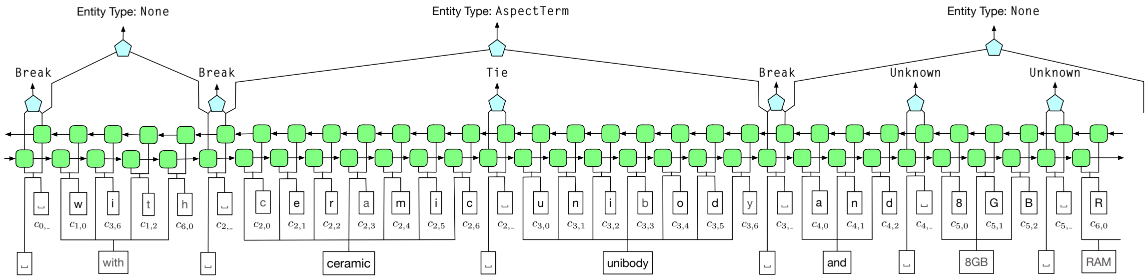

Figure 2: The illustration of AutoNER with `Tie or Break` tagging scheme. The named entity type is {`AspectTerm`}. “ceramic unibody” is a matched `AspectTerm` entity and “8GB RAM” is an unknown-typed high-quality phrase. `Unknown` labels will be skipped during the model training.

“Tie or Break” Tagging Scheme. Specifically, for every two adjacent tokens, the connection between them is labeled as (1) `Tie`, when the two tokens are matched to the same entity; (2) `Unknown`, if at least one of the tokens belongs to an unknown-typed high-quality phrase; (3) `Break`, otherwise.

An example can be found in Fig. 2. The distant supervision shows that “ceramic unibody” is a matched `AspectTerm` and “8GB RAM” is an unknown-typed high-quality phrase. Therefore, a `Tie` is labeled between “ceramic” and “unibody”, while `Unknown` labels are put before “8GB”, between “8GB” and “RAM”, and after “RAM”.

Tokens between every two consecutive `Break` form a token span. Each token span is associated with all its matched types, the same as for the modified `IOBES` scheme. For those token spans without any associated types, such as “with” in the example, we assign them the additional type `None`.

We believe this new scheme can better exploit the knowledge from dictionary according to the following two observations. First, even though the boundaries of an entity mention are mismatched by distant supervision, most of its inner ties are not affected. More interestingly, compared to multi-word entity mentions, matched unigram entity mentions are more likely to be false-positive labels. However, such false-positive labels will not introduce incorrect labels with the `Tie or Break` scheme, since either the unigram is a true entity mention or a false positive, it always brings two `Break` labels around.

AutoNER. In the `Tie or Break` scheme, entity spans and entity types are encoded into two folds. Therefore, we separate the entity span detection and entity type prediction into two steps.

For entity span detection, we build a binary classifier to distinguish `Break` from `Tie`, while `Unknown` positions will be skipped. Specifically, as shown in Fig. 2, for the prediction between i-th token and its previous token, we concatenate the output of the BiLSTM as a new feature vector, $\mathbf{u}_i$. $\mathbf{u}_i$ is then fed into a sigmoid layer, which estimates the probability that there is a `Break` as

$$p(y_i = \texttt{Break}|\mathbf{u}_i) = \sigma(\mathbf{w}^T\mathbf{u}_i)$$

where y_i is the label between the i-th and its previous tokens, σ is the sigmoid function, and $\mathbf{w}$ is the sigmoid layer’s parameter. The entity span detection loss is then computed as follows.

$$\mathcal{L}_{\text{span}} = \sum_{i|y_i \neq \texttt{Unknown}} l\big(y_i, p(y_i = \texttt{Break}|\mathbf{u}_i)\big)$$

Here, $l(\cdot,\cdot)$ is the logistic loss. Note that those `Unknown` positions are skipped.

After obtaining candidate entity spans, we further identify their entity types, including the `None` type for non-entity spans. As shown in Fig. 2, the output of the BiLSTM will be re-aligned to form a new feature vector, which is referred as $\mathbf{v}_i$ for i-th span candidate. $\mathbf{v}_i$ will be further fed into a softmax layer, which estimates the entity type distribution as

$$p(t_j|\mathbf{v}_i) = \frac{e^{\mathbf{t}_j^T\mathbf{v}_i}}{\sum_{t_k \in L} e^{\mathbf{t}_k^T\mathbf{v}_i}}$$

where t_j is an entity type and L is the set of all entity types including `None`.

Since one span can be labeled as multiple types, we mark the possible set of types for i-th entity span candidate as L_i. Accordingly, we modify the cross-entropy loss as follows.

$$\mathcal{L}_{\text{type}} = H(\hat{p}(\cdot|\mathbf{v}_i, L_i), p(\cdot|\mathbf{v}_i))$$

Here, $H(p, q)$ is the cross entropy between p and q, and $\hat{p}(t_j|\mathbf{v}_i, L_i)$ is the soft supervision distribution. Specifically, it is defined as:

$$\hat{p}(t_j|\mathbf{v}_i, L_i) = \frac{\delta(t_j \in L_i) \cdot e^{\mathbf{t}_j^T \mathbf{v}_i}}{\sum_{t_k \in L} \delta(t_k \in L_i) \cdot e^{\mathbf{t}_k^T \mathbf{v}_i}}$$

where $\delta(t_j \in L_i)$ is the boolean function of checking whether the i-th span candidate is labeled as the type t_j in the distant supervision.

It's worth mentioning that AutoNER has no CRF layer and Viterbi decoding, thus being more efficient than Fuzzy-LSTM-CRF for inference.

3.3 Remarks on "Unknown" Entities

"Unknown" entity mentions are not the entities of other types, but the tokens that we are less confident about their boundaries and/or cannot identify their types based on the distant supervision. For example, in Figure 1, "prostaglandin synthesis" is an "unknown" token span. The distant supervision cannot decide whether it is a `Chemical`, a `Disease`, an entity of other types, two separate single-token entities, or (partially) not an entity. Therefore, in the FuzzyCRF model, we assign all possible labels for these tokens.

In our AutoNER model, these "unknown" positions have undefined boundary and type losses, because (1) they make the boundary labels unclear; and (2) they have no type labels. Therefore, they are skipped.

4 Distant Supervision Refinement

In this section, we present two techniques to refine the distant supervision for better named entity taggers. Ablation experiments in Sec. 5.4 verify their effectiveness empirically.

4.1 Corpus-Aware Dictionary Tailoring

In dictionary matching, blindly using the full dictionary may introduce false-positive labels, as there exist many entities beyond the scope of the given corpus but their aliases can be matched. For example, when the dictionary has a non-related character name "Wednesday Addams"[2] and its alias "Wednesday", many Wednesday's will be wrongly marked as persons. In an ideal case, the dictionary should cover, and only cover entities occurring in the given corpus to ensure a high precision while retaining a reasonable coverage.

As an approximation, we tailor the original dictionary to a corpus-related subset by excluding entities whose canonical names never appear in the given corpus. The intuition behind is that to avoid ambiguities, people will likely mention the canonical name of the entity at least once. For example, in the biomedical domain, this is true for 88.12%, 95.07% of entity mentions on the BC5CDR and NCBI datasets respectively. We expect the NER model trained on such tailored dictionary will have a higher precision and a reasonable recall compared to that trained on the original dictionary.

4.2 Unknown-Typed High-Quality Phrases

Another issue of the distant supervision is about the false-negative labels. When a token span cannot be matched to any entity surface names in the dictionary, because of the limited coverage of dictionaries, it is still difficult to claim it as non-entity (i.e., negative labels) for sure. Specifically, some high-quality phrases out of the dictionary may also be potential entities.

We utilize the state-of-the-art distantly supervised phrase mining method, AutoPhrase (Shang et al., 2018), with the corpus and dictionary in the given domain as input. AutoPhrase only requires unlabeled text and a dictionary of high-quality phrases. We obtain quality multi-word and single-word phrases by posing thresholds (e.g., 0.5 and 0.9 respectively). In practice, one can find more unlabeled texts from the same domain (e.g., PubMed papers and Amazon laptop reviews) and use the same domain-specific dictionary for the NER task. In our experiments, for the biomedical domain, we use the titles and abstracts of 686,568 PubMed papers (about 4%) uniformly sampled from the whole PubTator database as the training corpus. For the laptop review domain, we use the Amazon laptop review dataset[3], which is designed for the aspect-based sentiment analysis (Wang et al., 2011).

We treat out-of-dictionary phrases as potential entities with "unknown" type and incorporate them as new dictionary entries. After this, only token spans that cannot be matched in this extended dictionary will be labeled as non-entity. Being aware of these high-quality phrases, we expect the trained NER tagger should be more accurate.

---

[2] `https://en.wikipedia.org/wiki/Wednesday_Addams`

[3] `http://times.cs.uiuc.edu/~wang296/Data/`

Table 1: Dataset Overview.

Dataset	BC5CDR	NCBI-Disease	LaptopReview
Domain	Biomedical	Biomedical	Technical Review
Entity Types	`Disease`, `Chemical`	`Disease`	`AspectTerm`
Dictionary	MeSH + CTD	MeSH + CTD	Computer Terms
Raw Sent. #	20,217	7,286	3,845

5 Experiments

We conduct experiments on three benchmark datasets to evaluate and compare our proposed Fuzzy-LSTM-CRF and AutoNER with many other methods. We further investigate the effectiveness of our proposed refinements for the distant supervision and the impact of the number of distantly supervised sentences.

5.1 Experimental Settings

Datasets are briefly summarized in Table 1. More details as as follows.

- **BC5CDR** is from the most recent BioCreative V Chemical and Disease Mention Recognition task. It has 1,500 articles containing 15,935 `Chemical` and 12,852 `Disease` mentions.
- **NCBI-Disease** focuses on Disease Name Recognition. It contains 793 abstracts and 6,881 `Disease` mentions.
- **LaptopReview** is from the SemEval 2014 Challenge, Task 4 Subtask 1 (Pontiki et al., 2014) focusing on laptop aspect term (e.g., “disk drive”) Recognition. It consists of 3,845 review sentences and 3,012 `AspectTerm` mentions.

All datasets are publicly available. The first two datasets are already partitioned into three subsets: a training set, a development set, and a testing set. For the LaptopReview dataset, we follow (Giannakopoulos et al., 2017) and randomly select 20% from the training set as the development set. Only raw texts are provided as the input of distantly supervised models, while the gold training set is used for supervised models.

Domain-Specific Dictionary. For the biomedical datasets, the dictionary is a combination of both the MeSH database[4] and the CTD Chemical and Disease vocabularies[5]. The dictionary contains 322,882 `Chemical` and `Disease` entity surfaces. For the laptop review dataset, the dictionary has 13,457 computer terms crawled from a public website[6].

Metric. We use the micro-averaged F_1 score as the evaluation metric. Meanwhile, precision and recall are presented. The reported scores are the mean across five different runs.

Parameters and Model Training. Based on the analysis conducted in the development set, we conduct optimization with the stochastic gradient descent with momentum. We set the batch size and the momentum to 10 and 0.9. The learning rate is initially set to 0.05 and will be shrunk by 40% if there is no better development F_1 in the recent 5 rounds. Dropout of a ratio 0.5 is applied in our model. For a better stability, we use gradient clipping of 5.0. Furthermore, we employ the early stopping in the development set.

Pre-trained Word Embeddings. For the biomedical datasets, we use the pre-trained 200-dimension word vectors [7] from (Pyysalo et al., 2013), which are trained on the whole PubMed abstracts, all the full-text articles from PubMed Central (PMC), and English Wikipedia. For the laptop review dataset, we use the GloVe 100-dimension pre-trained word vectors[8] instead, which are trained on the Wikipedia and GigaWord.

5.2 Compared Methods

Dictionary Match is our proposed distant supervision generation method. Specifically, we apply it to the testing set directly to obtain entity mentions with exactly the same surface name as in the dictionary. The type is assigned through a majority voting. By comparing with it, we can check the improvements of neural models over the distant supervision itself.

SwellShark, in the biomedical domain, is arguably the best distantly supervised model, especially on the BC5CDR and NCBI-Disease datasets (Fries et al., 2017). It needs no human annotated data, however, it requires extra expert effort for entity span detection on building POS tagger, designing effective regular expressions, and hand-tuning for special cases.

Distant-LSTM-CRF achieved the best performance on the LaptopReview dataset without annotated training data using a distantly supervised

---

[4] `https://www.nlm.nih.gov/mesh/download_mesh.html`

[5] `http://ctdbase.org/downloads/`

[6] `https://www.computerhope.com/jargon.htm`

[7] `http://bio.nlplab.org/`

[8] `https://nlp.stanford.edu/projects/glove/`

Table 2: [Biomedical Domain] NER Performance Comparison. The supervised benchmarks on the BC5CDR and NCBI-Disease datasets are LM-LSTM-CRF and LSTM-CRF respectively (Wang et al., 2018). SwellShark has no annotated data, but for entity span extraction, it requires pre-trained POS taggers and extra human efforts of designing POS tag-based regular expressions and/or hand-tuning for special cases.

Method	Human Effort other than Dictionary	BC5CDR			NCBI-Disease		
		Pre	Rec	F1	Pre	Rec	F1
Supervised Benchmark	Gold Annotations	88.84	85.16	**86.96**	86.11	85.49	**85.80**
SwellShark	Regex Design + Special Case Tuning	86.11	82.39	84.21	81.6	80.1	**80.8**
	Regex Design	84.98	83.49	**84.23**	64.7	69.7	67.1
Dictionary Match	None	93.93	58.35	71.98	90.59	56.15	69.32
Fuzzy-LSTM-CRF		88.27	76.75	82.11	79.85	67.71	73.28
AutoNER		88.96	81.00	**84.8**	79.42	71.98	**75.52**

Table 3: [Technical Review Domain] NER Performance Comparison. The supervised benchmark refers to the challenge winner.

Method	LaptopReview		
	Pre	Rec	F1
Supervised Benchmark	84.80	66.51	**74.55**
Distant-LSTM-CRF	74.03	31.59	53.93
Dictionary Match	90.68	44.65	59.84
Fuzzy-LSTM-CRF	85.08	47.09	60.63
AutoNER	72.27	59.79	**65.44**

LSTM-CRF model (Giannakopoulos et al., 2017).

Supervised benchmarks on each dataset are listed to check whether AutoNER can deliver competitive performance. On the BC5CDR and NCBI-Disease datasets, LM-LSTM-CRF (Liu et al., 2018) and LSTM-CRF (Lample et al., 2016) achieve the state-of-the-art F_1 scores without external resources, respectively (Wang et al., 2018). On the LaptopReview dataset, we present the scores of the Winner in the SemEval2014 Challenge Task 4 Subtask 1 (Pontiki et al., 2014).

5.3 NER Performance Comparison

We present F_1, precision, and recall scores on all datasets in Table 2 and Table 3. From both tables, one can find the AutoNER achieves the best performance when there is no extra human effort. Fuzzy-LSTM-CRF does have some improvements over the Dictionary Match, but it is always worse than AutoNER.

Even though SwellShark is designed for the biomedical domain and utilizes much more expert effort, AutoNER outperforms it in almost all cases. The only outlier happens on the NCBI-disease dataset when the entity span matcher in SwellShark is carefully tuned by experts for many special cases.

It is worth mentioning that AutoNER beats Distant-LSTM-CRF, which is the previous state-of-the-art distantly supervised model on the LaptopReview dataset.

Moreover, AutoNER's performance is competitive to the supervised benchmarks. For example, on the BC5CDR dataset, its F_1 score is only 2.16% away from the supervised benchmark.

5.4 Distant Supervision Explorations

We investigate the effectiveness of the two techniques that we proposed in Sec. 4 via ablation experiments. As shown in Table 4, using the tailored dictionary always achieves better F_1 scores than using the original dictionary. By using the tailored dictionary, the precision of the AutoNER model will be higher, while the recall will be retained similarly. For example, on the NCBI-Disease dataset, it significantly boosts the precision from 53.14% to 77.30% with an acceptable recall loss from 63.54% to 58.54%. Moreover, incorporating unknown-typed high-quality phrases in the dictionary enhances every score of AutoNER models significantly, especially the recall. These results match our expectations well.

5.5 Test F_1 Scores vs. Size of Raw Corpus

Furthermore, we explore the change of test F_1 scores when we have different sizes of distantly supervised texts. We sample sentences uniformly random from the given raw corpus and then evaluate AutoNER models trained on the selected sentences. We also study what will happen when the gold training set is available. The curves can be found in Figure 3. The X-axis is the number of

Table 4: Ablation Experiments for Dictionary Refinement. The dictionary for the LaptopReview dataset contains no alias, so the corpus-aware dictionary tailoring is not applicable.

Method	BC5CDR			NCBI-Disease			LaptopReview		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
AutoNER w/ Original Dict	82.79	70.40	76.09	53.14	63.54	57.87	69.96	49.85	58.21
AutoNER w/ Tailored Dict	84.57	70.22	76.73	77.30	58.54	66.63	Not Applicable		
AutoNER w/ Tailored Dict & Phrases	88.96	81.00	**84.8**	79.42	71.98	**75.52**	72.27	59.79	**65.44**

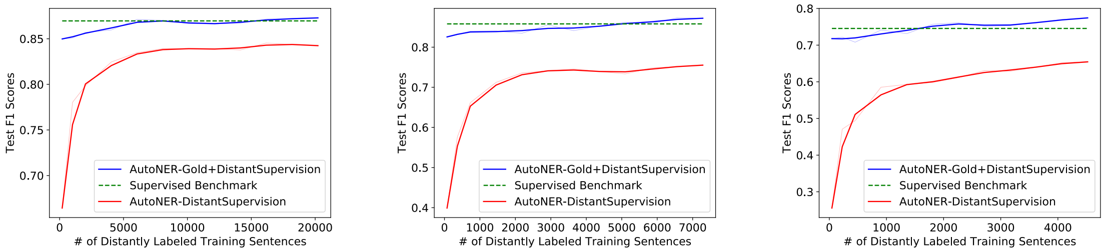

(a) BC5CDR

(b) NCBI

(c) LaptopReview

Figure 3: AutoNER: Test F_1 score vs. the number of distantly supervised sentences.

distantly supervised training sentences while the Y-axis is the F_1 score on the testing set.

When using distant supervision only, one can observe a significant growing trend of test F_1 score in the beginning, but later the increasing rate slows down when there are more and more raw texts.

When the gold training set is available, the distant supervision is still helpful to AutoNER. In the beginning, AutoNER works worse than the supervised benchmarks. Later, with enough distantly supervised sentences, AutoNER outperforms the supervised benchmarks. We think there are two possible reasons: (1) The distant supervision puts emphasis on those matchable entity mentions; and (2) The gold annotation may miss some good but matchable entity mentions. These may guide the training of AutoNER to a more generalized model, and thus have a higher test F_1 score.

5.6 Comparison with Gold Supervision

To demonstrate the effectiveness of distant supervision, we try to compare our method with gold annotations provided by human experts.

Specifically, we conduct experiments on the BC5CDR dataset by sampling different amounts of annotated articles for model training. As shown in Figure 4, we found that our method outperforms the supervised method by a large margin when less training examples are available. For example, when there are only 50 annotated articles available, the test F1 score drops substantially to 74.29%. To achieve a similar test F1 score (e.g., 83.91%) as our AutoNER models (i.e., 84.8%), the supervised benchmark model requires at least 300 annotated articles. Such results indicate the effectiveness and usefulness of AutoNER on the scenario without sufficient human annotations.

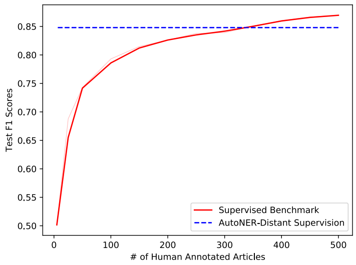

Figure 4: AutoNER: Test F_1 score vs. the number of human annotated articles.

Still, we observe that, when the supervised benchmark is trained with all annotations, it achieves the performance better than AutoNER. We conjugate that this is because AutoNER lacks more advanced techniques to handle distant supervision, and we leave further improvements of AutoNER to the future work.

6 Related Work

The task of supervised named entity recognition (NER) is typically embodied as a sequence labeling problem. Conditional random fields (CRF) models built upon human annotations and handcrafted features are the standard (Finkel et al., 2005; Settles, 2004; Leaman and Gonzalez, 2008). Recent advances in neural models have freed domain experts from handcrafting features for NER tasks. (Lample et al., 2016; Ma and Hovy, 2016; Liu et al., 2018). Such neural models are increasingly common in the domain-specific NER tasks (Sahu and Anand, 2016; Dernoncourt et al., 2017; Wang et al., 2018). Semi-supervised methods have been explored to further improve the accuracy by either augmenting labeled datasets with word embeddings or bootstrapping techniques in tasks like gene name recognition (Kuksa and Qi, 2010; Tang et al., 2014; Vlachos and Gasperin, 2006). Unlike these existing approaches, our study focuses on the distantly supervised setting without any expert-curated training data.

Distant supervision has attracted many attentions to alleviate human efforts. Originally, it was proposed to leverage knowledge bases to supervise relation extraction tasks (Craven et al., 1999; Mintz et al., 2009). AutoPhrase has demonstrated powers in extracting high-quality phrases from domain-specific corpora like scientific papers and business reviews (Shang et al., 2018) but it cannot categorize phrases into typed entities in a context-aware manner. We incorporate the high-quality phrases to enrich the domain-specific dictionary.

There are attempts on the distantly supervised NER task recently (Ren et al., 2015; Fries et al., 2017; He, 2017; Giannakopoulos et al., 2017). For example, SwellShark (Fries et al., 2017), specifically designed for biomedical NER, leverages a generative model to unify and model noise across different supervision sources for named entity typing. However, it leaves the named entity span detection to a heuristic combination of dictionary matching and part-of-speech tag-based regular expressions, which require extensive expert effort to cover many special cases. Other methods (Ren et al., 2015; He, 2017) also utilize similar approaches to extract entity span candidates before entity typing. Distant-LSTM-CRF (Giannakopoulos et al., 2017) has been proposed for the distantly supervised aspect term extraction, which can be viewed as an entity recognition task of a single type for business reviews. As shown in our experiments, our models can outperform Distant-LSTM-CRF significantly on the laptop review dataset.

To the best of our knowledge, AutoNER is the most effective model that can learn NER models by using, and only using dictionaries without any additional human effort.

7 Conclusion and Future Work

In this paper, we explore how to learn an effective NER model by using, and only using dictionaries. We design two neural architectures, Fuzzy-LSTM-CRF model with a modified `IOBES` tagging scheme and AutoNER with a new `Tie or Break` scheme. In experiments on three benchmark datasets, AutoNER achieves the best F_1 scores without additional human efforts. Its performance is even competitive to the supervised benchmarks with full human annotation. In addition, we discuss how to refine the distant supervision for better NER performance, including incorporating high-quality phrases mined from the corpus as well as tailoring dictionary according to the given corpus, and demonstrate their effectiveness in ablation experiments.

In future, we plan to further investigate the power and potentials of the AutoNER model with `Tie or Break` scheme in different languages and domains. Also, the proposed framework can be further extended to other sequence labeling tasks, such as noun phrase chunking. Moreover, going beyond the classical NER setting in this paper, it is interesting to further explore distant supervised methods for the nested and multiple typed entity recognitions in the future.

Acknowledgments

We would like to thank Yu Zhang from University of Illinois at Urbana-Champaign for providing results of supervised benchmark methods on the BC5CDR and NCBI datasets.

Research was sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, DTRA HDTRA11810026, Google Ph.D. Fellowship and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). Any opinions, findings, and conclusions or recommendations expressed in this document are those of the author(s) and should not be interpreted as the views of any U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

Mark Craven, Johan Kumlien, et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86.

Franck Dernoncourt, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits. 2017. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association*, 24(3):596–606.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

Jason Fries, Sen Wu, Alex Ratner, and Christopher Ré. 2017. Swellshark: A generative model for biomedical named entity recognition without labeled data. *arXiv preprint arXiv:1704.06360*.

Athanasios Giannakopoulos, Claudiu Musat, Andreea Hossmann, and Michael Baeriswyl. 2017. Unsupervised aspect term extraction with b-lstm & crf using automatically labelled datasets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 180–188.

Daniel Hanisch, Katrin Fundel, Heinz-Theodor Mevissen, Ralf Zimmer, and Juliane Fluck. 2005. Prominer: rule-based protein and gene entity recognition. *BMC bioinformatics*, 6(1):S14.

Wenqi He. 2017. Autoentity: automated entity detection from massive text corpora. *M.S. Thesis for Computer Science of University of Illinois at Urbana-Champaign*.

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

Pavel P Kuksa and Yanjun Qi. 2010. Semi-supervised bio-named entity recognition with word-codebook learning. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 25–36. SIAM.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Robert Leaman and Graciela Gonzalez. 2008. Banner: an executable survey of advances in biomedical named entity recognition. In *Biocomputing 2008*, pages 652–663. World Scientific.

Thomas Lin, Oren Etzioni, et al. 2012. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 84–88. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. *AAAI*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1064–1074.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Maria Pontiki, Dimitrios Galanis, John Pavlopoulos, Haris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, page 2735.

Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing. In *Proceedings of the 5th International Symposium on Languages in Biology and Medicine, Tokyo, Japan*, pages 39–43.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R Voss, and Jiawei Han. 2015. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 995–1004. ACM.

Sunil Sahu and Ashish Anand. 2016. Recurrent neural network models for disease name recognition using domain invariant features. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2216–2225.

Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 104–107. Association for Computational Linguistics.

Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*.

Buzhou Tang, Hongxin Cao, Xiaolong Wang, Qingcai Chen, and Hua Xu. 2014. Evaluating word representation features in biomedical named entity recognition tasks. *BioMed research international*, 2014.

Andreas Vlachos and Caroline Gasperin. 2006. Bootstrapping and evaluating named entity recognition in the biomedical domain. In *Proceedings of the HLT-NAACL BioNLP Workshop on Linking Natural Language and Biology*, pages 138–145. Association for Computational Linguistics.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 618–626. ACM.

Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. 2018. Cross-type biomedical named entity recognition with deep multi-task learning. *arXiv preprint arXiv:1801.09851*.

Zero-Shot Open Entity Typing as Type-Compatible Grounding

Ben Zhou[1], **Daniel Khashabi**[2], **Chen-Tse Tsai**[3], **Dan Roth**[2]
[1]University of Illinois, Urbana-Champaign, [2]University of Pennsylvania, [3]Bloomberg LP
`xzhou45@illinois.edu, {danielkh,danroth}@cis.upenn.edu, ctsai54@bloomberg.net`

Abstract

The problem of entity-typing has been studied predominantly in supervised learning fashion, mostly with task-specific annotations (for coarse types) and sometimes with distant supervision (for fine types). While such approaches have strong performance within datasets, they often lack the flexibility to transfer across text genres and to generalize to new type taxonomies. In this work we propose a *zero-shot* entity typing approach that requires no annotated data and can flexibly identify newly defined types.

Given a *type taxonomy* defined as Boolean functions of FREEBASE "types", we ground a given mention to a set of *type-compatible* Wikipedia entries and then infer the target mention's types using an inference algorithm that makes use of the types of these entries. We evaluate our system on a broad range of datasets, including standard fine-grained and coarse-grained entity typing datasets, and also a dataset in the biological domain. Our system is shown to be competitive with state-of-the-art supervised NER systems and outperforms them on out-of-domain datasets. We also show that our system significantly outperforms other zero-shot fine typing systems.

1 Introduction

Entity type classification is the task of connecting an entity mention to a given set of semantic types. The commonly used type sets range in size and level of granularity, from a small number of coarse-grained types (Tjong Kim Sang and De Meulder, 2003) to over a hundred fine-grained types (Ling and Weld, 2012). It is understood that semantic typing is a key component in many natural language understanding tasks, including Question Answering (Toral et al., 2005; Li and Roth, 2005) and Textual Entailment (Dagan et al., 2010, 2013). Consequently, the ability to type mentions semantically across domains and text genres, and to use a flexible type hierarchy, is essential for solving many important challenges.

Nevertheless, most commonly used approaches and systems for semantic typing (e.g., CORENLP (Manning et al., 2014), COGCOMPNLP (Khashabi et al., 2018), NLTK (Loper and Bird, 2002), SPACY) are trained in a supervised fashion and rely on high quality, task-specific annotation. Scaling such systems to other domains and to a larger set of entity types faces fundamental restrictions.

Coarse typing systems, which are mostly fully supervised, are known to fit a single dataset very well. However, their performance drops significantly on different text genres and even new data sets. Moreover, adding a new coarse type requires manual annotation and retraining. For fine-typing systems, people have adopted a distant-supervision approach. Nevertheless, the number of types used is small: the distantly-supervised FIGER dataset covers only 113 types, a small fraction of most-conservative estimates of the number of types in the English language (the FREEBASE (Bollacker et al., 2008) and WORDNET (Miller, 1995) hierarchies consist of more than $1k$ and $1.5k$ unique types, respectively). More importantly, adapting these systems, once trained, to new type taxonomies cannot be done flexibly.

As was argued in Roth (2017), there is a need to develop new training paradigms that support scalable semantic processing; specifically, there is a need to scale semantic typing to flexible type taxonomies and to multiple domains.

In this work, we introduce ZOE, a *zero-shot* entity typing system, with *open* type definitions. Given a mention in a sentence and a taxonomy of entity types with their definitions, ZOE identifies a set of types that are appropriate for the mention

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2065–2076
Brussels, Belgium, October 31 - November 4, 2018.

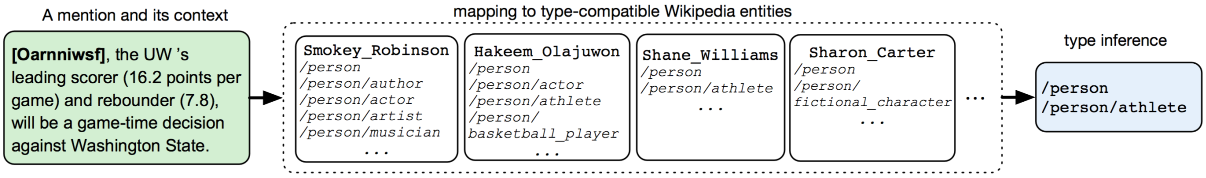

Figure 1: ZOE maps a given mention to its type-compatible entities in Wikipedia and infers a collection of types using this set of entities. While the mention "Oarnniwsf," a football player in the U. of Washington, does not exist in Wikipedia, we ground it to other entities with approximately the same types (§3).

in this context. ZOE does not require any training, and it makes use of existing data resources (e.g., Wikipedia) and tools developed without any task-specific annotation. The key idea is to ground each mention to a set of *type-compatible* Wikipedia entities. The benefit of using a set of Wikipedia titles as an intermediate representation for a mention is that there is much human-curated information in Wikipedia – categories associated with each page, FREEBASE types, and DBpedia types. These were put there independently of the task at hand and can be harnessed for many tasks: in particular, for determining the semantic types of a given mention in its context. In this grounding step, the guiding principle is that type-compatible entities often appear in similar contexts. We rely on contextual signals and, when available, surface forms, to rank Wikipedia titles and choose those that are more compatible with a given mention.

Importantly, our algorithm does not require a given mention to be in Wikipedia; in fact, in many cases (such as nominal mentions) the mentions are not available in Wikipedia. We hypothesize that any entity possible in English corresponds to some type-compatible entities in Wikipedia. We can then rely mostly on the context to reveal a *set* of *compatible titles*, those that are likely to share semantic types with the target mention. The fact that our system is not required to ground to the exact concept is a key difference between our grounding and "standard" Wikification approaches (Mihalcea and Csomai, 2007; Ratinov et al., 2011). As a consequence, while entity linking approaches rely heavily on priors associated with the surface forms and do not consider those that do not link to Wikipedia titles, our system mostly relies on context, regardless of whether the grounding actually exists or not.

Figure 1 shows a high-level visualization of our system. Given a mention, our system grounds it into type-compatible entries in Wikipedia. The target mention "Oarnniwsf," is not in Wikipedia, yet it is grounded to entities with approximately correct types. In addition, while some of the grounded Wikipedia entries are inaccurate in terms of entity types, the resulting aggregated decision is correct.

ZOE is an *open* type system, since it is not restricted to a closed set of types. In our experiments, we build on FREEBASE types as *primitive* types and use them to define types across seven different datasets. Note, however, that our approach is not fundamentally restricted to FREEBASE types; in particular, we allow types to be defined as Boolean formulae over these primitives (considering a type to be a set of entities). Furthermore, we support other primitives, e.g., DBPedia or Wikipedia entries. Consequently, our system can be used across type taxonomies; there is no need to restrict to previously observed types or retrain with annotations of new types. If one wants to use types that are outside our current vocabulary, one only needs to define the target type taxonomy in terms of the primitives used in this work.

In summary, our contributions are as follows:

- We propose a zero-shot open entity typing framework[1] that does not require training on entity-typing-specific supervised data.
- The proposed system outperforms existing zero-shot entity typing systems.
- Our system is competitive with fully-supervised systems in their respective domains across a broad range of coarse- and fine-grained typing datasets, and it outperforms these systems in out-of-domain settings.

2 Related Work

Named Entity Recognition (NER), for which the goal is to discover mention-boundaries in addition to typing, often using a small set of mutu-

---

[1] https://github.com/CogComp/zoe

ally exclusive types, has a considerable amount of work (Grishman and Sundheim, 1996; Mikheev et al., 1999; Tjong Kim Sang and De Meulder, 2003; Florian et al., 2003; Ratinov and Roth, 2009).

There have been many proposals to scale the systems to support a bigger type space (Fleischman and Hovy, 2002; Sekine et al., 2002). This direction was followed by the introduction of datasets with large label-sets, either manually annotated like BBN (Weischedel and Brunstein, 2005) or distantly supervised like FIGER (Ling and Weld, 2012). With larger datasets available, supervised-learning systems were proposed to learn from the data (Yosef et al., 2012; Abhishek et al., 2017; Shimaoka et al., 2017; Xu and Barbosa, 2018; Choi et al., 2018). Such systems have achieved remarkable success, mostly when restricted to their observed domain and labels.

There is a handful of works aiming to pave the road towards zero-shot typing by addressing ways to extract cheap signals, often to help the supervised algorithms: e.g., by generating gazetteers (Nadeau et al., 2006), or using the anchor texts in Wikipedia (Nothman et al., 2008, 2009). Ren et al. (2016) project labels in high-dimensional space and use label correlations to suppress noise and better model their relations. In our work, we choose not to use the supervised-learning paradigm and instead merely rely on a general entity linking corpus and the signals in Wikipedia. Prior work has already shown the importance of Wikipedia information for NER. Tsai et al. (2016a) use a cross-lingual WIKIFIER to facilitate cross-lingual NER. However, they do not explicitly address the case where the target entity does not exist in Wikipedia.

The zero-shot paradigm for entity typing has only recently been studied. Yogatama et al. (2015) proposed an embedding representation for user-defined features and labels, which facilitates information sharing among labels and reduces the dependence on the labels observed in the training set. The work of Yuan and Downey (2018) can also be seen in the same spirit, i.e., systems that rely on a form of representation of the labels. In a broader sense, such works–including ours–are part of a more general line of work on *zero-shot learning* (Chang et al., 2008; Palatucci et al., 2009; Norouzi et al., 2013; Romera-Paredes and Torr, 2015; Song and Roth, 2014). Our work can be thought of as the continuation of the same research direction.

Approach	Zero-shot?	Use labeled data?
ATTENTIVE (Shimaoka et al., 2017)	No	Yes
AAA (Abhishek et al., 2017)	No	Yes
NFETC-HIER(R) (Xu and Barbosa, 2018)	No	Yes
AFET (Ren et al., 2016)	No	Yes (partial)
PROTOLE (Ma et al., 2016)	**Yes** Prototype Embedding	Yes (partial)
OTYPER (Yuan and Downey, 2018)	**Yes** Word Embedding	Yes (partial)
(Huang et al., 2016)	**Yes** Concept-embedding Clustering	**No**
ZOE (ours)	**Yes** Type-Compatible Concepts	**No**

Table 1: Comparison of recent work on entity typing. Our system does not require any labeled data for entity typing; therefore it works on new datasets without retraining.

A critical step in the design of zero-shot systems is the characterization of the output space. For supervised systems, the output representations are trivial, as they are just indices. For zero-shot systems, the output space is often represented in a high-dimensional space that encodes the semantics of the labels. In OTYPER (Yuan and Downey, 2018), each type embedding is computed by averaging the word embeddings of the words comprising the type. The same idea is also used in PROTOLE (Ma et al., 2016), except that averaging is done only for a few prototypical instances of each type. In our work, we choose to define types using information in Wikipedia. This flexibility allows our system to perform well across several datasets without retraining. On a conceptual level, the work of Lin et al. (2012) and Huang et al. (2016) are close to our approach. The governing idea in these works is to cluster mentions, followed by propagating type information from representative mentions.

Table 1 compares our proposed system with several recently proposed models.

3 Zero-Shot Open Entity Typing

Types are conceptual containers that bind entities together to form a coherent group. Among the entities of the same type, *type-compatibility* creates a network of loosely connected entities:

Definition 1 (Weak Type Compatibility) *Two entities are type-compatible if they share at least one type with respect to a type taxonomy and the contexts in which they appear.*

In our approach, given a mention in a sentence, we aim to discover type-compatible entities in Wikipedia and then infer the mention's types using all the type-compatible entities together. The advantage of using Wikipedia entries is that the rich information associated with them allows us to infer the types more easily. Note that this problem is different from the standard entity linking or Wikification problem in which the goal is to find the corresponding entity in Wikipedia. Wikipedia does not contain all entities in the world, but an entity is likely to have at least one type-compatible entity in Wikipedia.

In order to find the type-compatible entities, we use the context of mentions as a proxy. Defining it formally:

Definition 2 (Context Consistency) *A mention m (in a context sentence s) is* context-consistent *with another well-defined mention m', if m can be replaced by m' in the context s, and the new sentence still makes logical sense.*

Hypothesis 1 *Context consistency is a strong proxy for type compatibility.*

Based on this hypothesis, given a mention m in a sentence s, we find other context-compatible mentions in a Wikified corpus. Since the mentions in the Wikified corpus are linked to the corresponding Wikipedia entries, we can infer m's types by aggregating information associated with these Wikipedia entries.

Figure 2 shows the high-level architecture of our proposed system. The inputs to the system are a mention m in a sentence s, and a type definition T. The output of the system is a set of types $\{t_{\text{Target}}\} \subseteq T$ in the target taxonomy that best represents the given mention. The type definitions characterize the target entity-type space. In our experiments, we choose to use FREEBASE types to define the types across 7 datasets; that is, T is a mapping from the set of FREEBASE types to the set of target types: $T : \{t_{\text{FB}}\} \rightarrow \{t_{\text{Target}}\}$. This definition comprises many atomic definitions; for example, we can define the type `location` as the disjunction of FREEBASE types like `FB.location` and `FB.geography`:

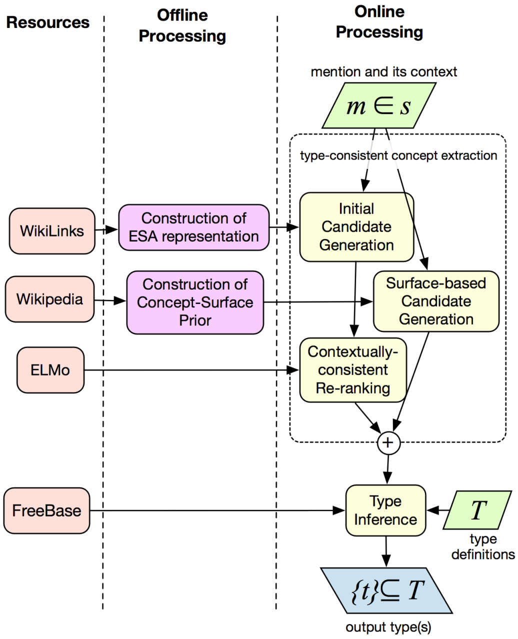

Figure 2: A high-level view of our approach. The inputs to the system are a mention m in a context s, and type definitions T. The output is set of types $\{t\}$ in the type definition. The figure also highlights the input resources, as well as offline and online processes.

```
T (Type Definitions)

- Target.company := FB.business &&
                    !(FB.education || FB.government)
- Target.location := FB.location || FB.geography ||
                     FB.casino || FB.court
- ...
```

The type definitions of a dataset reflect the understanding of a domain expert and the assumptions made in dataset design. Such definitions are often much cheaper to define, than to annotate full-fledged supervised datasets. It is important to emphasize that, to use our system on different datasets, one does not need to retrain it; there is one single system used across different datasets, working with different type definitions.

For notational simplicity, we define a few conventions for the rest of the paper. The notation $t \in T$, simply means t is a member of the image of the map T (i.e., t is a member of the target types). For a fixed concept c, the notation $T(c)$ is the application of $T(.)$ on the FREEBASE types attached to the concept c. For a collection of concepts C, $T(C)$ is defined as $\bigcup_{c \in C} T(c)$. We use $T_{\text{coarse}}(.)$ to refer to the subset of coarse types of $T(.)$, while $T_{\text{fine}}(.)$ defines the fine type subset.

Components in Figure 2 are described in the following sections.

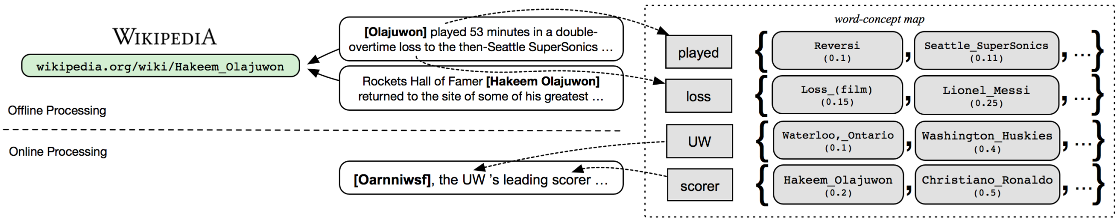

Figure 3: Extraction of topically relevant concepts. Word-concept map is pre-computed using WIKILINKS and used to retrieve the most relevant concepts for a given mention (see §3.1).

3.1 Initial Concept Candidate Generation

Given a mention, the goal of this step is to quickly generate a set of Wikipedia entries based on other words in the sentence. Since there are millions of entries in Wikipedia, it is extremely inefficient to go through all entries for each mention. We adopt ideas from explicit semantic analysis (ESA) (Gabrilovich and Markovitch, 2007), an approach to representing words with a vector of Wikipedia *concepts*, and to providing fast retrieval of the relevant Wikipedia concepts via inverted indexing.

In our construction we use the WIKILINKS (Singh et al., 2012) corpus, which contains a total of 40 million mentions over 3 million concepts. Each mention in WIKILINKS is associated with a Wikipedia concept. To characterize it formally, in the WIKILINKS corpus, for each concept c, there are example sentences $\mathsf{sent}(c) = \{s_i\}$.

Offline computation: The first step is to construct an ESA representation for each word in the WIKILINKS corpus. We create a mapping from each word in the corpus to the relevant concepts associated with it. The result is a map $\mathcal{S}$ from tokens to concepts: $\mathcal{S} : w \to \{c, \mathsf{score}(c|w)\}$ (see Figure 3), where $\mathsf{score}(c|w)$ denotes the association of the word w with concept c, calculated as the sum of the TF-IDF values of the word w in the sentences describing c:

$$\mathsf{score}(c|w) \triangleq \sum_{s \in \mathsf{sent}(c)} \sum_{w \in s} \text{tf-idf}(w, s).$$

That is, we treat each sentence as a document and compute TF-IDF scores for the words in it.

Online computation: For a given mention m and its sentence context s, we use our offline word-concept map $\mathcal{S}$ to find the concepts associated with each word, and aggregate them to create a single list of weighted concepts; i.e., $\sum_{w \in s} \mathcal{S}(w)$. The resulting concepts are sorted by the corresponding weights, and the top ℓ_{ESA} candidates form a set C_{ESA} which is passed to the next step.

3.2 Context-Consistent Re-Ranking

After quick retrieval of the initial concept candidates, we re-rank concepts in C_{ESA} based on context consistency between the input mention and concept mentions in WIKILINKS.

For this step, assume we have a representation that encodes the sentential information anchored on the mention. We denote this mention-aware context representation as $\mathsf{SentRep}(s|m)$. We define a measure of consistency between a concept c and a mention m in a sentence s:

$$\mathsf{Consistency}(c, s, m) = \mathsf{cosine}(\mathsf{SentRep}(s|m), \mathsf{ConceptRep}(c)), \quad (1)$$

where $\mathsf{ConceptRep}(c)$ is representation of a concept:

$$\mathsf{ConceptRep}(c) \triangleq \mathrm{avg}_s \left(\mathsf{SentRep}(s|c) \middle| s \in \text{WIKILINKS}, c \in s \right),$$

which is the average vector of the representation of all the sentences in WIKILINKS that describe the given concept.

We use pre-trained ELMO (Peters et al., 2018), a state-of-the-art contextual and mention-aware word representation. In order to generate $\mathsf{SentRep}(s|m)$, we run ELMO on sentence s, where the tokens of the mention m are concatenated with "_", and retrieve its ELMO vector as $\mathsf{SentRep}(s|m)$.

According to the consistency measure, we select the top ℓ_{ELMO} concepts for each mention. We call this set of concepts C_{ELMO}.

3.3 Surface-Based Concept Generation

While context often is a key signal for typing, one should not ignore the information included in the surface form of the mentions. If the corresponding concept or entity exists in Wikipedia, many mentions can be accurately grounded with only trivial prior probability $\mathbf{Pr}(\text{concept}|\text{surface})$. The prior distribution is pre-computed by calculating the frequency of the times a certain surface string refers to different concepts within Wikipedia.

In the test time, for a given mention, we use the pre-computed probability distribution to obtain the most likely concept, $c_{\text{surf}} = \arg\max_c \mathbf{Pr}(c|m)$, for the given mention m.

3.4 Type Inference

Our inference algorithm starts with selection of concepts, followed by inference of coarse and fine types. Our approach is outlined in Algorithm 1 and explained below.

Concept inference. To integrate surface-based and context-based concepts, we follow a simple rule: if the prior probability of the surface-based concept (c_{surf}) has confidence below a threshold λ, we ignore it; otherwise we include it among the concepts selected from context (C_{ELMo}), and only choose coarse and fine types from c_{surf}.

To map the selected concepts to the target entity types, we retrieve the FREEBASE-types of each concept and then apply the type definition T (defined just before §3.1). In Algorithm 1, the set of target types of a concept c is denoted as $T(c)$. This is followed by an aggregation step for selection of a coarse type $t_{\text{coarse}} \in T_{\text{coarse}}(.)$, and ends with the selection of a set of fine types $\{t_{\text{fine}}\} \subseteq T_{\text{fine}}(.)$.

Coarse type inference. Our type inference algorithm works in a relatively simple confidence analysis procedure. To this end, we define $\text{Count}(t; C)$ to be the number of occurrences of type t in the collection of concepts C:

$$\text{Count}(t; C) := |\{c : c \in C \text{ and } t \in T(c)\}|.$$

In theory, for a sensible type t, the count of context-consistent concepts that have this type should be higher than that of the initial concept candidates. In other words, $\frac{\text{Count}(t; C_{\text{ELMo}})/\ell_{\text{ELMo}}}{\text{Count}(t; C_{\text{ESA}})/\ell_{\text{ESA}}} > 1$. We select the first concept (in the C_{ELMo} ranking) which has some coarse type that matches this criterion. If there is no such concept, we use the coarse types of the highest scoring concept. To select one of the coarse types of the selected concept, we let each concept of C_{ELMo} vote based on its consistency score. We name this voting-based procedure $\text{SelectCoarse}(c)$, which selects one coarse type from a given concept:

$$\text{SelectCoarse}(c) \triangleq$$
$$\operatorname*{argmax}_{t} \sum_{\tilde{c} \in C_{\text{ELMo}}} \sum_{t \in T_{\text{coarse}}(\tilde{c})} \text{Consistency}(\tilde{c}, s, m),$$

where consistency is defined in Equation (1).

Algorithm 1: Type inference algorithm

```
Input mention $m$ in sentence $s$, retrieved concepts
  $C_{\text{ESA}}, C_{\text{ELMo}}, c_{\text{surf}}$, and type definition $T$
Output Inferred types $t_{\text{coarse}}$ and $\{t_{\text{fine}}\}$.
Define, $r(t, t'; C, C') := \frac{\text{Count}(t;C)/|C|}{\text{Count}(t';C')/|C'|}$,
  $r(t; C, C') := r(t, t; C, C')$,
  $r(t, t'; C, ) := r(t, t'; C, C)$.

$\tau_{\text{surf}} \leftarrow \{t | t \in T_{\text{coarse}}(c_{\text{surf}}), r(t; C_{\text{ELMo}}, C_{\text{ESA}}) > 1\}$
if $\mathbf{Pr}(c_{surf}|m) \geq \lambda$ and $\tau_{surf} \neq \emptyset$ then
    $t_{\text{coarse}} \leftarrow \text{SelectCoarse}(c_{\text{surf}})$
    $\tilde{C} \leftarrow \{c_{\text{surf}}\} \cup C_{\text{ELMo}}$
    $\{t_{\text{fine}}\} \leftarrow \left\{ t_f \middle| \begin{array}{l} t_f \in T_{\text{fine}}(c_{\text{surf}}), \\ \text{compatible w/ } t_{\text{coarse}} \text{ and,} \\ r(t_f, t_{\text{coarse}}; \tilde{C}) \geq \eta_s \end{array} \right\}$
else
    $\tilde{C}_{\text{ELMo}} \leftarrow \left\{ c \middle| \begin{array}{l} c \in C_{\text{ELMo}}, \exists t \in T_{\text{coarse}}(c) \\ r(t; C_{\text{ELMo}}, C_{\text{ESA}}) > 1 \end{array} \right\}$
    if $\tilde{C}_{\text{ELMo}} = \varnothing$ then
        $\tilde{c} \leftarrow \text{argmax}_{c \in C_{\text{ELMo}}} \text{Consistency}(c, s, m)$
    else
        $\tilde{c} \leftarrow \text{argmax}_{c \in \tilde{C}_{\text{ELMo}}} \text{Consistency}(c, s, m)$
    end
    $t_{\text{coarse}} \leftarrow \text{SelectCoarse}(\tilde{c})$
    $\{t_{\text{fine}}\} \leftarrow \left\{ t_f \middle| \begin{array}{l} t_f \in T_{\text{fine}}(C_{\text{ELMo}}), \\ \text{compatible w/ } t_{\text{coarse}} \text{ and,} \\ r(t_f, t_{\text{coarse}}; C_{\text{ELMo}}) \geq \eta_c \end{array} \right\}$
end
```

Fine type inference. With the selected coarse type, we take only the fine types that are compatible with the selected coarse type (e.g., the fine type `/people/athlete` and the coarse type `/people` are compatible).

Among the compatible fine types, we further filter the ones that have better support from the context. Therefore, we select the fine types t_f such that $\frac{\text{Count}(t_f; C_{\text{ELMo}})}{\text{Count}(t_c; C_{\text{ELMo}})} \geq \eta$, where t_c is the previously selected coarse type which is compatible with t_f. Intuitively, the fraction filters out the fine-grained candidate types that don't have enough support compared to the selected coarse type.

4 Experiments

Empirically, we study the behavior of our system compared to published results. All the results are reproduced except the ones indicated by $*$, which are directly cited from their corresponding papers.

Datasets. In our experiments, we use a wide range of typing datasets:

- For **coarse** entity typing, we use MUC (Grishman and Sundheim, 1996), CoNLL (Tjong Kim Sang and De Meulder, 2003), and OntoNotes (Hovy et al., 2006).

	Approach	Trained on	FIGER Acc.	FIGER $F1_{\text{ma}}$	FIGER $F1_{\text{mi}}$	BBN Acc.	BBN $F1_{\text{ma}}$	BBN $F1_{\text{mi}}$	$\text{OntoNotes}_{\text{fine}}$ Acc.	$\text{OntoNotes}_{\text{fine}}$ $F1_{\text{ma}}$	$\text{OntoNotes}_{\text{fine}}$ $F1_{\text{mi}}$
Closed Type	AFET* (Ren et al., 2016)	FIGER	53.3	69.3	66.4	-	-	-	-	-	-
	NFETC-HIER(R)*(Xu and Barbosa, 2018)	FIGER	68.9	81.9	79.0	-	-	-	-	-	-
	AFET* (Ren et al., 2016)	BBN	-	-	-	68.3	74.4	74.7	-	-	-
	AAA* (Abhishek et al., 2017)	BBN	-	-	-	73.3	79.1	79.2	-	-	-
	AFET* (Ren et al., 2016)	$\text{OntoNotes}_{\text{fine}}$	-	-	-	-	-	-	55.1	71.1	64.7
	NFETC-HIER(R)* (Xu and Barbosa, 2018)	$\text{OntoNotes}_{\text{fine}}$	-	-	-	-	-	-	60.2	76.4	70.2
Open Type	OTYPER (Yuan and Downey, 2018)	FIGER	47.2	69.1	67.2	27	50.3	49.5	31.6	34.5	32.1
		BBN	5.3	11.5	11.5	29	54.4	48.8	2.5	5.1	5.4
		$\text{OntoNotes}_{\text{fine}}$	0.4	15.6	16.8	23.6	51.1	47.9	31.8	39.1	36
	ELMONN	×	21.5	57.7	53.8	49.3	68.4	66.2	0.5	21.2	21.8
	WIKIFIERTYPER	×	17.2	33.3	46.2	45.8	52.3	66.1	47.8	65.6	58.2
	ZOE (ours)	×	**58.8**	**74.8**	**71.3**	**61.8**	**74.6**	**74.9**	**50.7**	**66.9**	**60.8**

Table 2: Evaluation of fine-grained entity-typing: we compare our system with state-of-the-art systems (§4.1) For each column, the best zero-shot and overall results are **bold-faced** and underlined, respectively. Numbers are $F1$ in percentage. For supervised systems, we report their in-domain performances, since they do not transfer to other datasets with different labels. For OTYPER, cells with gray color indicate *in-domain* evaluation, which is the setting in which it has the best performance. Our system outperforms all the other zero-shot baselines, and achieves competitive results compared to the best supervised systems.

System	Trained on	OntoNotes PER	OntoNotes LOC	OntoNotes ORG	CoNLL PER	CoNLL LOC	CoNLL ORG	MUC PER	MUC LOC	MUC ORG
COGCOMPNLP	OntoNotes	98.4	91.9	97.7	83.7	70.1	68.3	82.5	76.9	86.7
COGCOMPNLP	CoNLL	**94.4**	59.1	**87.8**	95.6	92.9	90.5	**90.8**	90.8	90.9
ZOE (ours)	×	88.4	**70.0**	85.6	**90.1**	**80.1**	**73.9**	87.8	**90.9**	**91.2**

Table 3: Evaluation of coarse entity-typing (§4.2): we compare two supervised entity-typers with our system. For the supervised systems, cells with gray color indicate *in-domain* evaluation. For each column, the best, out-of-domain and overall results are **bold-faced** and underlined, respectively. Numbers are $F1$ in percentage. In most of the out-of-domain settings our system outperforms the supervised system.

- For **fine** typing, we focus on FIGER (Ling and Weld, 2012), BBN (Weischedel and Brunstein, 2005), and $\text{OntoNotes}_{\text{fine}}$ (Gillick et al., 2014).
- In addition to the news NER, we use the BB3 dataset (Delėger et al., 2016), with contain mentions of *bacteria* or other notions, extracted from sentences of scientific papers.

ZOE's parameters. We use different type definitions for each dataset. In order to design type definitions for each dataset, we follow in the footsteps of Abhishek et al. (2017) and randomly sample 10% of the test set. For the experiments, we exclude the sampled set. For completeness, we have included the type definitions of the major experiments in Appendix D.

The parameters are set universally across different experiments. For parameters that determine the number of extracted concepts, we use $\ell_{\text{ESA}} = 300$ and $\ell_{\text{ELMo}} = 20$, which are based on the upper-bound analysis in Appendix A. For other parameters, we set $\lambda = 0.5$, $\eta_s = 0.8$ and $\eta_c = 0.3$, based on the FIGER dev set. We emphasize that these parameters are universal across our evaluations.

Evaluation metrics. Given a collection of mentions M, denote the set of gold types and predicted types of a mention $m \in M$ as $T_g(m)$ and $T_p(m)$ respectively. We define the following metrics for our evaluations:

- Strict Accuracy (Acc.): $\frac{|\{m|T_g(m)=T_p(m)\}|}{|M|}$.
- Macro F1 ($F1_{\text{ma}}$): Macro Precision is defined as $\frac{1}{|M|}\sum_{m\in M}\frac{|T_p(m)\cap T_g(m)|}{|T_p(m)|}$. With this, the definitions of Macro recall and F1 follow.
- Micro F1 ($F1_{\text{mi}}$): The precision is defined as $\frac{\sum_{m\in M}|T_p(m)\cap T_g(m)|}{\sum_{m\in M}|T_p(m)|}$, and the Micro recall and F1 follow the same pattern.

In the experiment in §4.3, to evaluate systems on unseen types we used modified versions of metrics. Let $G(t)$ be the number of mentions with gold type t, $P(t)$ be the number of mentions predicted to have type t, $C(t)$ be the number of mentions correctly predicted to have type t:

- The precision corresponding to $F1_{\text{ma}}^{\text{type}}$ is defined as $\sum_t \frac{C(t)}{P(t)}\frac{G(t)}{\sum_{t'}G(t')}$; recall follows the same pattern.
- The precision corresponding to $F1_{\text{mi}}^{\text{type}}$ is defined as $\frac{\sum_t C(t)}{\sum_t P(t)}$; recall follows the same pattern.

Approach	$F1^{\text{type}}_{\text{ma}}$	$F1^{\text{type}}_{\text{mi}}$
ELMoNN	63.1	53.8
WikifierTyper	53.0	43.9
OTyper (Yuan and Downey, 2018)	50.6	23.4
Zoe (ours)	**71.7**	**71.1**

Table 4: Comparing systems where no labels (types) are seen a priori (§4.3).

System	Bacteria	not-Bacteria	Overall
WikifierTyper	54.8	**86.2**	70.5
ELMoNN	67.6	81.2	74.4
Zoe (ours)	**68.1**	84.2	**76.2**

Table 5: Results of the system classifying mentions to "*bacteria*" or something else (§4.4). Numbers are $F1$ in percentage.

Baselines. To add to the best published results on each dataset, we create two simple and effective baselines. The first baseline, ELMoNN, selects the nearest neighbor types to a given mention, where *mentions* and *types* are represented by ELMo vectors. To create a representation for each type t, we average the representation of the WikiLinks sentences that contain mentions of type t (as explained in §3.2). Our other baseline, WikifierTyper, uses Wikifier (Tsai et al., 2016b) to map the mention to a Wikipedia concept, followed by mapping to FreeBase types, and finally projecting them to the target types, via type definition function $T(.)$. Additionally, to compare with published zero-shot systems, we compare our system to OTyper, a recently published *open*-typing system. Unfortunately, to the best of our knowledge, the systems proposed by Ma et al.; Huang et al. (2016; 2016) are not available online for empirical comparison.

4.1 Fine-Grained Entity Typing

We evaluate our system for *fine-grained* named-entity typing. Table 2 shows the evaluation result for three datasets, FIGER, BBN, and $\text{OntoNotes}_{\text{fine}}$. We report our system's performance, our zero-shot baselines, and two supervised systems (AFET, plus the-state-of-the-art), for each dataset. There is no easy way to transfer the supervised systems across datasets, hence no out-of-domain numbers for such systems. For each dataset, we train OTyper and evaluate on the test sets of all the three datasets. In order to run OTyper on different datasets, we disabled original dataset-specific entity and type features. As a result, among the *open* typing systems, our system has significantly better results. In addition, our system has competitive scores compared to the supervised systems.

4.2 Coarse Entity Typing

In Table 3 we study entity typing for the coarse types on three datasets. We focus on three types that are shared among the datasets: *PER*, *LOC*, *ORG*. In coarse-entity typing, the best available systems are heavily supervised. In this evaluation, we use gold mention spans; i.e., we force the decoding algorithm of the supervised systems to select the best of the three classes for each gold mention. As expected, the supervised systems have strong in-domain performance. However, they suffer a significant drop when evaluated in a different domain. Our system, while not trained on any supervised data, achieves better or comparable performance compared to other supervised baselines in the out-of-domain evaluations.

4.3 Typing of Unseen Types within Domain

We compare the quality of *open* typing, in which the target type(s) have not been seen before. We compare our system to OTyper, which relies on supervised data to create representations for each type; however, it is not restricted to the *observed types*. We follow a similar setting to Yuan and Downey (2018) and split the FIGER test in folds (one fold per type) and do cross-validations. For each fold, mentions of only one type are used for evaluation, and the rest are used for training OTyper. To be able to evaluate on *unseen* types (only for this experiment), we use modified metrics $F1^{\text{type}}_{\text{ma}}$ and $F1^{\text{type}}_{\text{mi}}$ that measure *per type* quality of the system (§4). In this experiment, we focus on a within-domain setting, and show the results of transfer across genres in the next experiments. The results are summarized in Table 4. We observe a significant margin between Zoe and other systems, including OTyper.

4.4 Biology Entity Typing

We go beyond the scope of popular entity-typing tasks, and evaluate the quality of our system on a dataset that contains sentences from scientific papers (Delėger et al., 2016), which makes it different from other entity-typing datasets. The mentions refer to either "*bacteria*", or some miscellaneous class (two class typing). As indicated in Ta-

	FIGER			BBN			OntoNotes$_{\text{fine}}$		
Approach	Acc.	$F1_{\text{ma}}$	$F1_{\text{mi}}$	Acc.	$F1_{\text{ma}}$	$F1_{\text{mi}}$	Acc.	$F1_{\text{ma}}$	$F1_{\text{mi}}$
ZOE (ours)	58.8	74.8	71.3	61.8	74.6	74.9	50.7	66.9	60.8
no surface-based concepts	-8.8	-7.5	-9.2	-12.9	-7.0	-8.6	-1.8	-1.2	-0.1
no context-based concepts	-39.3	-42.1	-25.4	-36.4	-31.0	-13.9	-10.0	-12.3	-7.4

Table 6: Ablation study of different ways in which concepts are generated in our system (§4.5). The first row shows performance of our system on each dataset, followed by the *change* in the performance upon dropping a component. While both signals are crucial, contextual information is playing more important role than the mention-surface signal.

ble 5, our system's overall scores are higher than our baselines.

4.5 Ablation Study

We carry out ablation studies that quantify the contribution of surface information (§3.3) and context information (§3.2). As Table 6 shows, both factors are crucial and complementary for the system. However, the contextual information seems to have a bigger role overall.

We complement our qualitative analysis with the quantitative share of each component. In 69.3%, 54.6%, and 69.7% of mentions, our system uses the context information (and ignores the surface), in FIGER, BBN, and OntoNotes$_{\text{fine}}$ datasets, respectively, underscoring the importance of contextual information.

4.6 Error Analysis

We provide insights into specific reasons for the mistakes made by the system. For our analysis, we use the erroneous decisions in the FIGER dev set. Two independent annotators label the cause(s) of the mistakes, resulting in 83% agreement between the annotators. The disagreements are later reconciled by an adjudication step.

1. *Incompatible concept, due to context information:* Ambiguous contexts, or short ones, often contribute to the inaccurate mapping to concepts. In our manual annotations, 23.3% of errors are caused, at least partly, by this issue.
2. *Incompatible concept, due to surface information:* Although the prior probability is high, the surface-based concept could be wrong. About 26% of the errors are partly due to the surface signal errors.
3. *Incorrect type, due to type inference:* Even when the system is able to find several type-compatible concepts, it can fail due to inference errors. This could happen if the types attached to the type-compatible concepts are not the majority among other types attached to other concepts. This is the major reason behind 56.6% of errors.
4. *Incorrect type, due to type definition:* Some errors are caused by the inaccurate definition of the type mapping function T. About 23% of the mistakes are partly caused by this issue.

Note that each mistake could be caused by multiple factors; in other words, the above categories are not mutually disjoint events. A slightly more detailed analysis is included in Appendix C.

5 Conclusion

Moving beyond a fully supervised paradigm and scaling entity-typing systems to support bigger type sets is a crucial challenge for NLP. In this work, we have presented ZOE, a zero-shot open entity typing framework. The significance of this work is threefold. First, the proposed system does not require task-specific labeled data. Our system relies on type definitions, which are much cheaper to obtain than annotating thousands of examples. Second, our system outperforms existing state-of-the-art zero-shot systems by a significant margin. Third, we show that without reliance on task-specific supervision, one can achieve relatively robust transfer across datasets.

6 Acknowledgement

The authors would like to thank Jennifer Sheffield, Stephen Mayhew, and Qiang Ning for invaluable suggestions. This work was supported by Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Additionally, this research is supported by grants from Google and the Allen Institute for Artificial Intelligence (`allenai.org`).

References

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 797–807, Valencia, Spain. Association for Computational Linguistics.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *Proceedings of The Conference on Artificial Intelligence (AAAI)*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke S. Zettlemoyer. 2018. Ultra-fine entity typing. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. Recognizing textual entailment: Rational, evaluation and approaches–erratum. *Natural Language Engineering*, 16(01):105–105.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzoto. 2013. Recognizing textual entailment: Models and applications.

Louise Delėger, Robert Bossy, Estelle Chaix, Mouhamadou Ba, Arnaud Ferrė, Philippe Bessieres, and Claire Nėdellec. 2016. Overview of the bacteria biotope task at bionlp shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 12–22.

Michael Fleischman and Eduard H. Hovy. 2002. Fine grained classification of named entities. In *COLING*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJcAI*, volume 7, pages 1606–1611.

Daniel Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *CoRR*, abs/1412.1820.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, volume 1.

Eduard H. Hovy, Mitchell P. Marcus, Martha Palmer, Lance A. Ramshaw, and Ralph M. Weischedel. 2006. Ontonotes: The 90% solution. In *HLT-NAACL*.

Lifu Huang, Jonathan May, Xiaoman Pan, and Heng Ji. 2016. Building a fine-grained entity typing system overnight for a new x (x= language, domain, genre). *arXiv preprint arXiv:1603.03112*.

Daniel Khashabi, Mark Sammons, Ben Zhou, Tom Redman, Christos Christodoulopoulos, Vivek Srikumar, Nicholas Rizzolo, Lev Ratinov, Guanheng Luo, Quang Do, Chen-Tse Tsai, Subhro Roy, Stephen Mayhew, Zhilli Feng, John Wieting, Xiaodong Yu, Yangqiu Song, Shashank Gupta, Shyam Upadhyay, Naveen Arivazhagan, Qiang Ning, Shaoshi Ling, and Dan Roth. 2018. CogCompNLP: your swiss army knife for nlp. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*.

Xin Li and Dan Roth. 2005. Learning question classifiers: The role of semantic information. *Journal of Natural Language Engineering*, 11(4).

Thomas Lin, Oren Etzioni, et al. 2012. No noun phrase left behind: detecting and typing unlinkable entities. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 893–903. Association for Computational Linguistics.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of The Conference on Artificial Intelligence (AAAI)*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 171–180.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242. ACM.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 1–8. Association for Computational Linguistics.

George Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

David Nadeau, Peter D Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 266–277. Springer.

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Gregory S. Corrado, and Jeffrey Dean. 2013. Zero-shot learning by convex combination of semantic embeddings. In *International Conference on Learning Representations (ICLR)*.

Joel Nothman, James R Curran, and Tara Murphy. 2008. Transforming wikipedia into named entity training data. In *Proceedings of the Australasian Language Technology Association Workshop 2008*, pages 124–132.

Joel Nothman, Tara Murphy, and James R Curran. 2009. Analysing wikipedia and gold-standard corpora for ner training. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 612–620. Association for Computational Linguistics.

Mark Palatucci, Dean Pomerleau, Geoffrey E. Hinton, and Tom M. Mitchell. 2009. Zero-shot learning with semantic output codes. In *Advances in Neural Information Processing Systems (NIPS)*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*.

Bernardino Romera-Paredes and Philip H. S. Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Dan Roth. 2017. Incidental supervision: Moving beyond supervised learning. In *Proc. of the Conference on Artificial Intelligence (AAAI)*.

Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. 2002. Extended named entity hierarchy. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1271–1280.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to wikipedia. *University of Massachusetts, Amherst, Tech. Rep. UM-CS-2012*, 15.

Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *Proceedings of The Conference on Artificial Intelligence (AAAI)*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Antonio Toral, Elisa Noguera, Fernando Llopis, and Rafael Munoz. 2005. Improving question answering using named entity recognition. In *International Conference on Application of Natural Language to Information Systems*, pages 181–191. Springer.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016a. Cross-lingual named entity recognition via wikification. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016b. Cross-lingual named entity recognition via wikification. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 219–228.

Ralph Weischedel and Ada Brunstein. 2005. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*, 112.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 9pp.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 291–296.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. *Proceedings of COLING 2012: Posters*, pages 1361–1370.

Zheng Yuan and Doug Downey. 2018. Otyper: A neural architecture for open named entity typing. In *Proceedings of The Conference on Artificial Intelligence (AAAI)*.

Attention-Guided Answer Distillation for Machine Reading Comprehension

Minghao Hu[†][*], **Yuxing Peng**[†], **Furu Wei**[§],
Zhen Huang[†], **Dongsheng Li**[†], **Nan Yang**[§], **Ming Zhou**[§]
[†] College of Computer, National University of Defense Technology
[§] Microsoft Research Asia
{huminghao09,pengyuxing,huangzhen,dsli}@nudt.edu.cn
{fuwei,nanya,mingzhou}@microsoft.com

Abstract

Despite that current reading comprehension systems have achieved significant advancements, their promising performances are often obtained at the cost of making an ensemble of numerous models. Besides, existing approaches are also vulnerable to adversarial attacks. This paper tackles these problems by leveraging *knowledge distillation*, which aims to transfer knowledge from an ensemble model to a single model. We first demonstrate that vanilla knowledge distillation applied to answer span prediction is effective for reading comprehension systems. We then propose two novel approaches that not only penalize the prediction on confusing answers but also guide the training with alignment information distilled from the ensemble. Experiments show that our best student model has only a slight drop of 0.4% F1 on the SQuAD test set compared to the ensemble teacher, while running 12× faster during inference. It even outperforms the teacher on adversarial SQuAD datasets and NarrativeQA benchmark.

1 Introduction

Machine reading comprehension (MRC), which aims to answer questions about a given passage or document, is a long-term goal of natural language processing. Recent years have witnessed rapid progress from early cloze-style test (Hermann et al., 2015; Hill et al., 2016) to latest answer extraction test (Rajpurkar et al., 2016; Joshi et al., 2017). Several end-to-end neural networks based approaches even outperform the human performance in terms of exact match accuracy on the SQuAD dataset (Wang et al., 2017, 2018; Yu et al., 2018).

Despite of the advancements, there are still two problems that impedes the deployment of real-world MRC applications. First, although effective, current approaches are *not efficient* because superior performances are usually achieved by ensembling multiple trained models. For example, Seo et al. (2017) submit an ensemble model consisting of 12 training runs and Huang et al. (2018) boost the result with 31 models. The ensemble system, however, has two major drawbacks: the inference time is slow and a huge amount of resource is needed. Second, existing models are *not robust* since they are vulnerable to adversarial attacks. Jia and Liang (2017) show that the models are easily fooled by appending an adversarial sentence into the passage. Such fragility on adversarial examples severely diminishes the practicality of current MRC systems.

One promising direction to address these problems is model compression (Bucilu et al., 2006), which attempts to compress the ensemble model into a single model that has comparable performances. Particularly, the *knowledge distillation* approach (Hinton et al., 2014) has been proposed to train a student model with the supervision of a teacher model. Such idea is further explored to enhance the generalizability and robustness of image recognition systems (Papernot et al., 2016). Some subsequent works attempt to transfer teacher's intermediate representation, such as feature map (Romero et al., 2015; Zagoruyko and Komodakis, 2017), neuron selectivity (Huang and Wang, 2017) and so on, to provide additional supervisions.

In this paper, we present the first work to investigate knowledge distillation in the context of MRC, to improve efficiency and robustness simultaneously. We first apply the standard knowledge disillation to MRC models, by mimicking output distributions of answer boundaries from an ensemble model, and observe consistent improvements upon a strong baseline. We then propose two novel

---

[*] Contribution during internship at Microsoft Research Asia.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2077–2086
Brussels, Belgium, October 31 - November 4, 2018.

Passage: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. The Champ Bowl 40 took place in Chicago.

Question1: Who won Super Bowl 50?
Question2: Where did Super Bowl 50 take place?

(a) Gold answers (red) versus confusing answers (blue) in the SQuAD and adversarial SQuAD datasets.

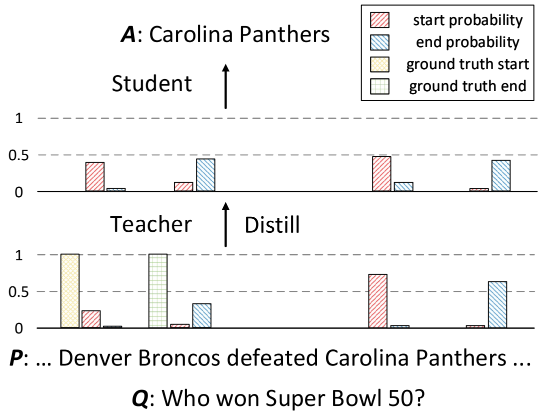

(b) The knowledge biased towards the confusing answer is distilled from the teacher to the student.

Figure 1: An illustration of confusing answer and biased distillation in machine reading comprehension.

distillation approaches to further transfer knowledge between the teacher and the student.

First, we introduce *answer distillation*, which penalizes the most confusing answer with a margin loss, to deal with the problem that biased knowledge misleads the student into incorrect predictions. We find that in MRC datasets there exists many confusing answers (Figure 1(a)), which match the category of true answers but are semantically-contradicted to the question, and an extreme case is the adversarial example. Once the teacher produces biased probabilities towards these distractors, inaccurate distributions will be distilled and later used to supervise the student. As a result, the student could produce over-confident wrong predictions. We refer to this problem as *biased distillation*, and Figure 1(b) gives an example. To address this problem, our approach distilles the boundary of the strongest distractor from the teacher, and explicitly informs the student to decrease its confidence score. This forces the student to produce comparable and unbiased distributions between candidate answers.

Second, we present *attention distillation* that aims to match the attention distribution between the teacher and the student. We notice that neural attention plays an important role in MRC tasks by enabling the model to capture complex interactions between the question and the passage. Compared to other forms of intermediate representation such as feature map and neuron selectivity, attention distribution is more compressed and more informative in reflecting semantic similarities of input text pairs. Hence, by mimicking the word alignments from an ensemble model, we expect that the student can learn to attend more precisely with better compression efficiencies.

We evaluate our approach on the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), the adversarial SQuAD dataset (Jia and Liang, 2017) and the NarrativeQA benchmark (Kočiskỳ et al., 2017). Compared to the ensemble teacher, the single student model trained with our approach has a slight drop of 0.4% F1 on the SQuAD test set, while running 12× faster during inference. The student even outperforms the teacher on the adversarial SQuAD dataset, and surpasses the teacher in terms of Bleu-1 score on the NarrativeQA benchmark.

2 Background

2.1 Machine Reading Comprehension

In the extractive MRC task, the question and the passage are described as sequences of word tokens, denoted as $Q = \{q_i\}_{i=1}^{n}$ and $P = \{p_j\}_{j=1}^{m}$ respectively. The task is to predict an answer A, which is constrained as a segment of text in the passage: $A = \{p_j\}_{j=k}^{l}$.

To model complicated interactions between the question and the passage, the attention mechanism (Bahdanau et al., 2015) has been widely used. Let $V = \{v_i\}_{i=1}^{n}$ and $U = \{u_j\}_{j=1}^{m}$ denote the encoded question/passage representations, where v_i and u_j are both dense vectors. A similarity matrix $E \in \mathbb{R}^{n \times m}$ can be computed as:

$$E_{ij} = f(v_i, u_j)$$

where f is a scalar function producing the similarity between two inputs. Let $\mathrm{softmax}(x)$ denote the softmax function that normalizes the vec-

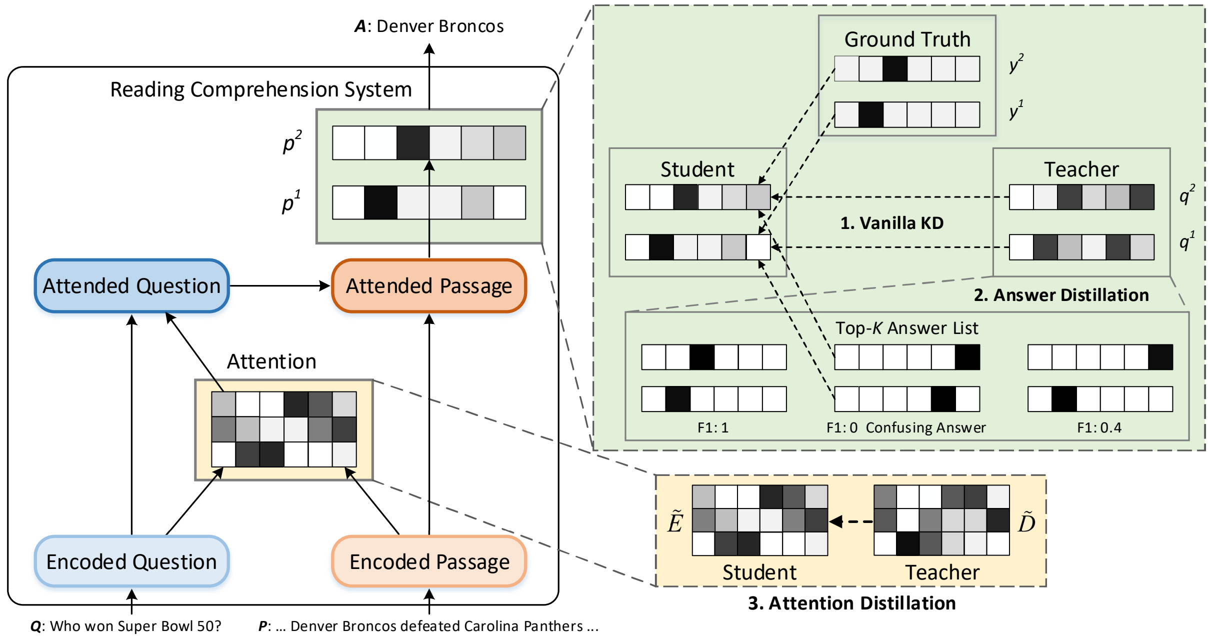

Figure 2: Overview of our approaches. In vanilla knowledge distillation (green), the cross entropy is minimized between the student/teacher distributions of answer positions. In answer distillation (green), the student is trained to penalize the most confusing answer distilled from the teacher. In attention distillation (yellow), mean-squared error is minimized between the student/teacher attention distributions. Darker color denotes higher probability.

tor x. Then the attention distribution across the question for the j-th passage word is computed as $\tilde{E}_j = \text{softmax}(E_{:j})$. This attentive information is commonly used to summarize the entire question as a single vector that is later fused back into the j-th passage word (Seo et al., 2017), resulting in an attended passage representation.

Finally, based on the attended passage, previous works usually apply a pointer network (Vinyals et al., 2015) to produce two probabilities p^1 and p^2 that indicate the answer start and end positions:

$$p(A|Q,P) = p^1(k|Q,P)p^2(l|k,Q,P)$$

where p is the model distribution.

2.2 Knowledge Distillation

The knowledge distillation framework (Hinton et al., 2014) consists of two networks: a *teacher* T, which is a pre-trained large model or an ensemble of multiple models, and a *student* S, which is a smaller network that learns from the teacher. The idea is to supervise the student with not only ground truth labels but also output distributions of the teacher. Concretely, given a data set of examples of the form (X, Y), the cross-entropy loss can be minimized to learn a multi-class classifier:

$$\mathcal{L}_{CE}(\theta) = -\sum_{k=1}^{m} Y_k \log p(k|X;\theta)$$

where p is the model distribution that is parameterized by θ, and m indicates the number of classes. The standard knowledge distillation is to replace ground truth Y with a soft probability distribution q generated by the teacher as:

$$q = \text{softmax}(\frac{\alpha}{\tau}), \quad p = \text{softmax}(\frac{\beta}{\tau})$$

$$\mathcal{L}_{KD}(\theta_S) = -\sum_{k=1}^{m} q(k|X;\theta_T)\log p(k|X;\theta_S)$$

where α and β are pre-softmax logits for teacher and student respectively, and τ is a temperature coefficient that is normally set to 1. A higher τ produces a softer probability distribution, and thus, provides more information about the relative similarity between classes. As Hinton et al. (2014) suggested, the above losses can be jointly optimized as follows:

$$\mathcal{L}(\theta_S) = \mathcal{L}_{CE}(\theta_S) + \lambda\mathcal{L}_{KD}(\theta_S)$$

where λ is usually set as τ^2 since the magnitudes of gradients produced by $\mathcal{L}_{KD}$ scale as $1/\tau^2$.

3 Attention-guided Answer Distillation

Figure 2 gives an overview of our distillation framework, which mainly consists of three approaches including vanilla knowledge distillation, answer distillation and attention distillation. First, we explore standard knowledge distillation in the context of MRC by replacing one-hot labels with soft output distributions extracted from the teacher. We then distill the span of the most confusing answer from the teacher, so that the student can learn to distinguish gold answers from the distractor. Next, we utilize teacher's attention distributions to guide the student's training process for forcing the student to attend more precisely. Finally, the student is jointly trained with the above distilled knowledge.

3.1 Vanilla Knowledge Distillation

The standard training method for MRC models is to minimize the cross entropies on two answer positions (Wang and Jiang, 2017):

$$\mathcal{L}_{CE} = -\sum_{k=1}^{m}\sum_{l=1}^{m} y_k^1 \log p^1(k) + y_l^2 \log p^2(l|k)$$

where y^1 and y^2 are one-hot labels for the answer start and end positions respectively, and m refers to the passage length. We denote $p^1(k|Q,P)$ as $p^1(k)$ and $p^2(l|k,Q,P)$ as $p^2(l|k)$ for abbreviation.

Following the standard procedure of knowledge distillation in Section 2.2, we can replace one-hot labels with output distributions of answer start and end positions predicted by the teacher as:

$$\mathcal{L}_{KD} = -\sum_{k=1}^{m}\sum_{l=1}^{m} q^1(k) \log p^1(k) + q^2(l|k) \log p^2(l|k)$$

Here, we first consider the teacher as a single model, and later extend it to the ensemble scenario in Section 3.4.

3.2 Answer Distillation

Vanilla knowledge distillation allows the student to learn relative similarities between answer candidates at position level. However, the student can suffer from the biased distillation problem once the teacher makes wrong predictions towards confusing answers. As a result, the model may be over-confident on the distractors during inference. Therefore, it is necessary to produce confidence scores that are comparable and unbiased between candidate answers. Since there usually exists one most confusing candidate answer, we hence consider explicitly informing the student about its boundary so as to relatively decrease the corresponding confidence.

Specifically, we perform inference with the teacher network to get a top-K answer list $\mathcal{A}_{\mathcal{K}}$ for each example, and measure the word overlap between the ground truth A^* and each answer candidate A_i from the list. The one that shares no overlap with gold answers and has the highest confidence score is chosen as the *confusing answer*. We use the F1 scoring function $F_1(A^*, A_i)$ as the measurement of word overlap and use the probability $q(A_i)$ as the confidence score. If the F1 scores of all top-K answers are larger than 0, then we argue that the teacher makes a good prediction and therefore do not distill the confusing answer for this example. The above process is performed on the entire training set to annotate all potential confusing answers.

Once the process is finished, we force the student to produce confidence scores of gold answers that are distinguishable from the score of confusing answer, by minimizing a margin ranking loss (Bai et al., 2010) as:

$$\begin{aligned}\mathcal{L}_{ANS} = &\max(0, 1 - \beta_k^1 + \beta_i^1) + \\ &\max(0, 1 - \beta_l^2 + \beta_j^2)\end{aligned}$$

where β is the student's pre-softmax logit, k and l indicate the boundary of gold span while i and j refer to the confusing boundary. With this loss, we penalize the strongest distractor on which the model is over-confident, and encourage the true answer that is underestimated by the student.

3.3 Attention Distillation

The above distillation approaches allow knowledge to be transferred through teacher's outputs. However, we would like to distill not only the final outputs but also some intermediate representations (Romero et al., 2015), in order to provide additional supervised signals for training the student network. The standard approach is to regress the student's intermediate passage representation to the teacher's corresponding representation as a pre-training step. Nevertheless, this approach is cumbersome in that the dimension of passage representation, denoted as $h \times m$ where h is the hidden size, is quite large, leading to a huge amount of resource consumption as we need to distill all training examples. Since neural attention is more compressed and contains rich informantion about where to attend, we hence propose to match the attention distribution between the teacher and the student instead.

Following the notation in Section 2.1, let $\tilde{D}_j$ and $\tilde{E}_j$ denote the attention distribution across the question for the j-th passage word in teacher and student repectively. We define a mean-squared loss as:

$$\mathcal{L}_{ATT} = \frac{1}{2}\sum_{j=1}^{m}||\tilde{D}_j - \tilde{E}_j||^2$$

Compared to regressing the hidden representation, attention distillation brings in two benefits: 1) the dimension of similarity matrix is $n \times m$, where the question length n is significantly smaller than the hidden size h, resulting in a better compression efficiency; 2) the student can learn to attend more precisely as neural attention directly reflects semantic similarities between input text pairs.

3.4 Joint Training

So far, we define three losses to transfer knowledge between MRC models. Next we consider using a joint objective to train the student network as:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda\mathcal{L}_{KD} + \gamma\mathcal{L}_{ANS} + \delta\mathcal{L}_{ATT} \qquad (1)$$

where λ is set as τ^2 to scale gradients, γ and δ are two hyper-parameters that control the task-specific weights.

In addition, since the knowledge is distilled from an ensemble model, we need to integrate multiple outputs into an unified label. For vanilla knowledge distillation and attention distillation, we use an arithmetic mean of their individual predicted distributions as the final soft label. As for answer distillation, we choose the confusing answer that has the highest confidence score among all models.

In summary, the entire training procedure is: 1) train an ensemble teacher model; 2) distill knowledge using the teacher on the training set; 3) integrate multiple types of knowledge to build a new training set; 4) train a single student model with Equation 1 on the new dataset.

4 Experiments

4.1 Datasets and Metrics

We conduct experiments on the following three datasets.

SQuAD (Stanford Question Answering Dataset) (Rajpurkar et al., 2016) is a machine comprehension dataset containing $100,000$+ questions that are annotated by crowd workers on 536 Wikipedia articles. The answer to each question is always a span in the corresponding passage.

Adversarial SQuAD (Jia and Liang, 2017) is a dataset aiming to test whether the model truely understands the text by appending an adversarial sentence to the passage. A strong confusing answer is constructed in the adversarial sentence to distract the answer prediction.

NarrativeQA (Kočiskỳ et al., 2017) is a benchmark proposed for story-based reading comprehension. The answers in this dataset are handwritten by human annotators based on a short summary. Following Tay et al. (2018), we compete on the summary setting to compare with reported baselines. We choose the span that achieves the highest Rouge-L score with respect to the gold answers as labels for training.

We use the official metrics to perform the evaluation. Specifically, for both SQuAD and adversarial SQuAD datasets, we report exact match (EM) and F1 scores. As for the NarrativeQA benchmark, the metrics are Blue-1, Bleu-4 and Rouge-L.

4.2 Implementation Details

We implement the Reinforced Mnemonic Reader (RMR) (Hu et al., 2018) as our base model, which contains the standard attention mechanism and the two-step answer prediction described in Section 2.1. Therefore, our distillation approaches can be seamlessly used, and the improvement of our approaches can be directly compared to the baseline. We use slightly different network architectures and hyper-parameters for different datasets. More specifically, we use the original configuration for SQuAD and adversarial SQuAD datasets. As for the NarrativeQA benchmark, we truncate the passage to the first 800 words and use a batch size of 32. Besides, we also remove ELMo embeddings (Peters et al., 2018) and reduce the number of aligning layer to 2 to avoid out-of-memory problem.

We run 12 single models with the identical architecture but different initial parameters to obtain the ensemble model. The student has the same network architecture as the teacher. We reuse the models trained on the SQuAD dataset to test on

Model	Dev		Test	
	EM	F1	EM	F1
LR Baseline[1]	40.0	51.0	40.4	51.0
FusionNet[2]	75.3	83.6	76.0	83.9
BiSAE[3]	77.9	85.6	78.6	85.8
R-Net+[4]	-	-	79.9	86.5
SLQA+[5]	80.0	87.0	80.4	87.0
QANet[6]	-	-	80.9	87.8
BiSAE (E)	79.6	86.6	81.0	87.4
R-Net+ (E)	-	-	82.6	88.5
SLQA+ (E)	82.0	88.4	82.4	88.6
QANet (E)	-	-	82.7	89.0
RMR[7]	78.9	86.3	79.5	86.6
RMR (E)	**81.2**	**87.9**	**82.3**	**88.5**
RMR + A2D	80.3	87.5	81.5	88.1

Table 1: Comparison of different approaches on the SQuAD test set, extracted on May 9, 2018: Rajpurkar et al. (2016)[1], Huang et al. (2018)[2], Peters et al. (2018)[3], Wang et al. (2017)[4], Wang et al. (2018)[5], Yu et al. (2018)[6] and Hu et al. (2018)[7]. BiSAE refers to BiDAF + Self Attention + ELMo. (E: ensemble model)

adversarial SQuAD datasets, but we retrain another pair of teacher and student models for the NarrativeQA benchmark. The temperature τ is tuned among $[1, 2, 3, 5]$ and is set as 2 by default. The weight γ is set to 0.3 and δ is 0.1. K is set to 4 for generating the top-K answer list. All experiments and runtime benchmarks are tested on a single Nvidia Tesla P100 GPU. Our approach is denoted as A2D for abbreviation.

4.3 Main Results

In this section we report main results on three MRC datasets[1]. Since our main purpose is to compress the ensemble model into a single model that possesses better efficiency and robustness, we focus on comparing our approach with the ensemble baseline.

We present the results on the test set of SQuAD in Table 1. We can see that the student network is able to compete with the ensemble model with only a slight drop of 0.4% on F1. Moreover, nearly 80% of the improvement in terms of F1 achieved by the ensemble model is successfully transferred to the distilled model, which indicates the effectiveness of our approach.

To validate the effect of our approach on enhancing robustness, we show the results on two adversarial SQuAD datasets, namely AddOneSent and AddSent, in Table 2. As we can see, the improvement on adversarial data is much higher than the one on original SQuAD dataset: the student network successfully surpasses the teacher. The significant improvement comes from the fact that there exists much more confusing answers in adversarial datasets, and hence the baseline is more likely to be over-confident on distractors. Our approach, however, explicitly decreases distractor's confidence, thus yielding more robust predictions against adversarial examples.

Model	AddSent		AddOneSent	
	EM	F1	EM	F1
LR Baseline	17.0	23.2	22.3	30.4
BiSAE	38.7	44.4	48.0	54.7
SLQA+	-	52.1	-	62.7
DCN+ (MINI)[1]	52.2	59.7	60.1	67.5
FusionNet (E)	46.2	51.4	54.7	60.7
SLQA+ (E)	-	54.8	-	64.2
RMR	53.0	58.5	60.9	67.0
RMR (E)	56.0	61.1	62.7	68.5
RMR + A2D	**56.0**	**61.3**	**63.3**	**69.3**

Table 2: Comparison of different approaches on two adversarial SQuAD datasets: Min et al. (2018)[1]. (E: ensemble model)

Model	Bleu-1	Bleu-4	Rouge-L
Seq2Seq	15.9	1.3	13.2
AS Reader[1]	23.2	6.4	22.3
BiDAF[2]	33.7	15.5	36.3
BiAttention[3]	36.6	19.8	41.4
RMR	48.4	24.6	51.5
RMR (E)	50.1	**27.5**	**53.9**
RMR + A2D	**50.4**	26.5	53.3

Table 3: Comparison of different approaches on the NarrativeQA test set using summaries: Kadlec et al. (2016)[1], Seo et al. (2017)[2] and Tay et al. (2018)[3]. (E: ensemble model)

To verify the generalizability of our approach among various datasets, we further detail the results on the test set of NarrativeQA in Table 3, Despite that both of our single baseline and ensemble model have already achieved top results, the student's performance can still be boosted using our approach, even outperforming the teacher in terms of Bleu-1 score.

4.4 Speedup over Ensemble Model

Next, in order to show the improvement on efficiency brought by our approach, we compare

---

[1] https://worksheets.codalab.org/worksheets/0xfa2548e70c2540ca8c2d26d7dd402f7f/.

	Params	Time	Speedup
RMR (E)	6.9m×12	118.2	-
RMR + A2D	6.9m	**9.6**	**12.3×**

Table 4: Comparison between the ensemble teacher and the single student on the SQuAD dev set. Time denotes number of minutes needed to perform the entire inference. (E: ensemble model)

the inference speedup of the student against the teacher in Table 4. Since we use 12 single models to make the ensemble, we can easily see that the student is nearly 12× faster than the teacher. Besides, compared to the baseline, the student has nearly the same number of parameters, demonstrating that our approach does not introduce additional computation complexity.

4.5 Ablation Study

To get better insights of our distillation approaches, we conduct in-depth ablation study on both the development set of SQuAD and the AddSent set of adversarial SQuAD. We mainly focus on the F1 score since it is used as the main metric on the SQuAD leaderboard.

Table 5 shows the ablation results. First, we evaluate the vanilla knowledge distillation by removing the KD loss. We find that this loss contributes a lot on both datasets, implying that standard knowledge distillation is helpful for MRC models. Next we ablate the attention loss (ATT), and observe a smaller performance degradation on both datasets, which suggests that matching attention distributions is also beneficial but less effective. We then test the effect of removing answer distillation (ANS), and discover that although the impact on the SQuAD dev set is relatively small, the influence on the AddSent dataset is quite large. We argue that this is because the model trained with answer distillation can better handle the biased distillation problem, which is more severe in the adversarial dataset. Finally, we replace the joint training process with a stage-wise fashion proposed by Romero et al. (2015). Concretely, we first warm up the student by matching the attention distribution as a pre-training step, and then minimize the rest of losses to train the model. The result, however, shows that this strategy does harm to the performance. We think the reason may be that the pre-training step leads the model into a local minima.

	Dev		AddSent	
	F1	ΔF1	F1	ΔF1
RMR + A2D	87.5	-	61.3	-
- Vanilla KD	<u>86.8</u>	<u>-0.7</u>	60.1	-1.2
- ATT	87.0	-0.5	60.4	-0.9
- ANS	87.1	-0.4	<u>59.8</u>	<u>-1.5</u>
- Joint Training	87.3	-0.2	60.8	-0.5

Table 5: Ablation study of different knowledge distillation approaches on SQuAD dev set and AddSent set.

4.6 Analysis and Discussion

We now give a qualitative analysis on our approach by visualizing attention distributions and output probabilities for both of the baseline and the distilled model. From Figure 3(a) we can see that, both models are good at finding candidate answers, such as "*56.2%*" and "*2.8%*", according to the key question word "percentage". Nevertheless, the base model pays more attention on question-passage word pairs around the confusing answer "*56.2%*". As a result, the model is more likely to produce a high confidence score on "*56.2%*", as shown in Figure 3(b). Using our approach, however, leads to less concentrations on superfical clues around the confusing answer. Instead, the distilled model is able to focus more on the critical alignments (e.g., "*Protestant*" to "*Protestants*"), and therefore predicts the correct answer "*2.8%*".

To show on which type of questions our approach is doing better, we report the results in Figure 4. We can see that our approach yields consistent performance gain over the baseline across different question types on various datasets. Particularly, A2D provides an obvious advantage for question types such as "*what*", "*who*", "*which*", "*how*", "*why*" and so on.

5 Related Work

Machine Reading Comprehension. Benefiting from large-scale machine reading comprehension (MRC) datasets (Hermann et al., 2015; Hill et al., 2016; Rajpurkar et al., 2016; Joshi et al., 2017), end-to-end neural networks have achieved promising results (Wang et al., 2018; Yu et al., 2018). Wang and Jiang (2017) combine the match-LSTM with pointer networks to predict the answer boundary. Wang et al. (2017) match the context aginst itself to refine the passage representation. Later, a variety of attention mechanisms have been

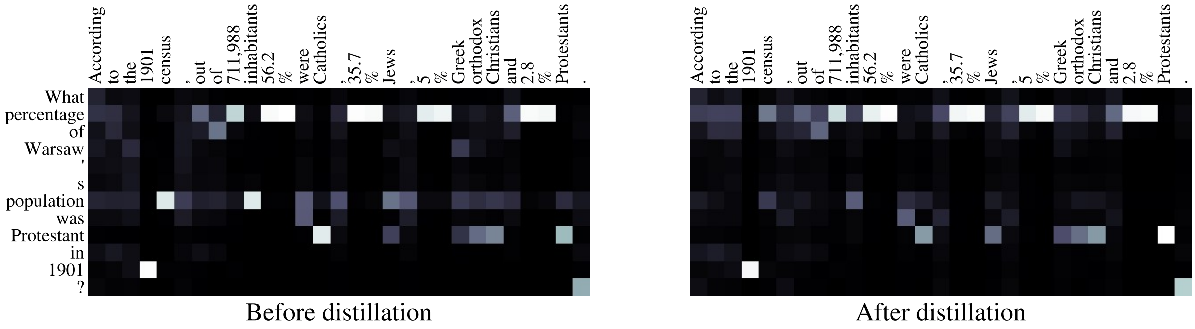

(a) Alignments before/after distillation. Using A2D, the attention distributions concentrate less on superfical clues around the confusing answer "*56.2%*" (e.g., "*population*" to "*inhabitants*", "*Protestant*" to "*Catholics*" and so on.).

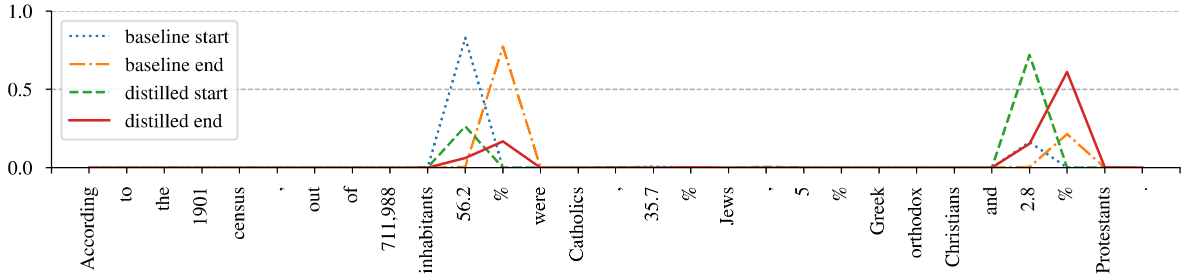

(b) The base model points to the confusing answer ("*56.2%*"), while the distilled model predicts the correct one ("*2.8%*").

Figure 3: A case study between the base model and the distilled model.

proposed, such as bi-attention (Seo et al., 2017), coattention (Xiong et al., 2018), fully-aware attention (Huang et al., 2018) and reattention (Hu et al., 2018). Among these works, two common traits can be summarized as: 1) compute a similary matrix between the question and the passage; 2) sequentially predict the answer start and end positions. Our proposed approach is a simple and effective adaptation to existing models by taking advantage of these traits, and do not complicate previous works more than necessary.

Efficiency and Robustness in MRC. Improving efficiency and robustness for reading comprehension system has attracted a lot of interest in recent years. For efficiency, previous works mostly concentrate on how to scale passage-level models to large corpora such as a document without increasing computation complexity. Existing approaches (Chen et al., 2017; Clark and Gardner, 2018) usually first retrieve relevant passages with a ranking model and then return an answer with a reading model. As for robustness, Wang and Bansal (2018) train the model with an adversarial data augmentation method. Min et al. (2018) propose to selectively read salient sentences rather than the entire passage, so as to avoid looking at the adversarial sentence. Our approach, however, focuses on improving efficiency and robustness by transferring knowledge from a cumbersome ensemble model to a single model.

Knowledge Distillation. Knowledge distillation is first explored by Bucilu et al. (2006) and Hinton et al. (2014), which attempts to transfer knowledge defined as soft output distributions from a teacher to a student. Later works have been proposed to distill not only the final output but also intermediate representation from the teacher (Romero et al., 2015; Zagoruyko and Komodakis, 2017; Huang and Wang, 2017). Papernot et al. (2016) show that knowledge distillation can be used to prevent the network from adversarial attacks in image recognition. Radosavovic et al. (2017) introduce data distillation that annotates large-scale unlabelled data for omni-supervised learning.

In natural language processing (NLP), Mou et al. (2016) distill task-specific knowledge from word embeddings. Kuncoro et al. (2016) propose to learn a single parser from an ensemble of parsers. Kim and Rush (2016) investigate knowledge distillation for neural machine translation by approximately matching the sequence-level distribution of the teacher. Nakashole and Flauger (2017) propose to learn bilingual mapping functions through a distilled training objective. Xu and Yang (2017) distill discriminative knowledge across languages for cross-lingual text classifica-

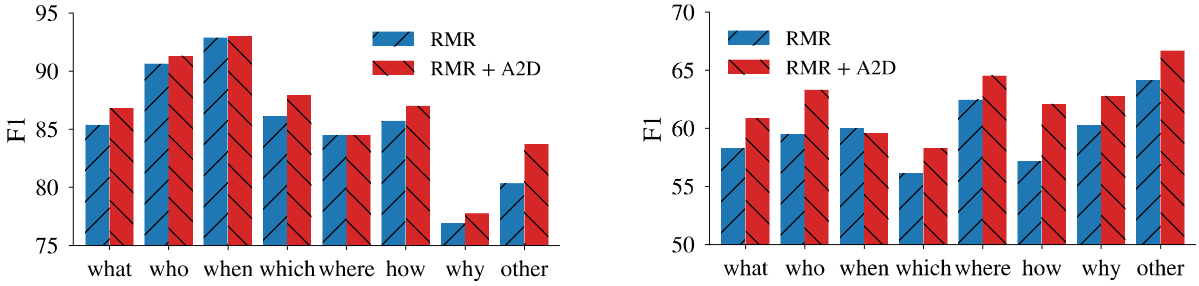

(a) Results on the SQuAD dev set.

(b) Results on the AddSent set.

Figure 4: Comparison of different question types between the base model and the distilled model.

tion. Our work shows that the standard knowledge distillation and its novel variants can be successfully applied to the MRC task.

6 Conclusion

In this paper, we investigate knowledge distillation for machine reading comprehension. We first explore vanilla knowledge distillation to transfer knowledge of answer positions, and then propose two variant approaches including answer distillation for penalizing student's predictions on confusing answer spans, and attention distillation for transferring teacher's attentive information. Experiments show that the ensemble model has been successfully compressed into a single model that possesses better efficiency and robustness.

In future work, we will explore new distillation methods that have better compression capabilities for MRC tasks, such as distilling knowledge from a single model instead of the ensemble without lossing performances, adding weights on knowledge based on the distilling quality and so on. We also plan to further study the biased distillation problem and explore the compatibility of our approach in other NLP tasks such as natural language inference (Bowman et al., 2015), answer sentence selection (Yang et al., 2015) and so on.

Acknowledgments

We would like to thank Pranav Rajpurkar for his help with SQuAD submissions. This work is funded by National Key R&D Program of China (No. 0708063216003).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Olivier Chapelle, and Kilian Weinberger. 2010. Learning to rank with (a lot of) word features. *Information retrieval*, 13(3):291–314.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference systems. In *Proceedings of EMNLP*.

Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of SIGKDD*, pages 535–541. ACM.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of ACL*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of ACL*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading childrens books with explicit memory representations. In *Proceedings of ICLR*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *Proceedings of NIPS Workshop*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of IJCAI*.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2018. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *Proceedings of ICLR*.

Zehao Huang and Naiyan Wang. 2017. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of EMNLP*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of ACL*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of ACL*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of EMNLP*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *arXiv preprint arXiv:1712.07040*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. In *Proceedings of EMNLP*.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents.

Lili Mou, Ran Jia, Yan Xu, Ge Li, Lu Zhang, and Zhi Jin. 2016. Distilling word embeddings: An encoding approach. In *Proceedings of CIKM*, pages 1977–1980. ACM.

Ndapandula Nakashole and Raphael Flauger. 2017. Knowledge distillation for bilingual dictionary induction. In *Proceedings of EMNLP*, pages 2487–2496.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word prepresentations. In *Proceedings of NACCL*.

Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. 2017. Data distillation: Towards omni-supervised learning. *arXiv preprint arXiv:1712.04440*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *Proceedings of ICLR*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-range reasoning for machine comprehension. *arXiv preprint arXiv:1803.09074*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of NIPS*.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *Proceedings of ICLR*.

Wei Wang, Ming Yan, and Chen Wu. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of ACL*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of ACL*.

Yicheng Wang and Mohit Bansal. 2018. Robust machine comprehension models via adversarial training. In *Proceedings of NAACL*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. Dcn+: Mixed objective and deep residual coattention for question answering. In *Proceedings of ICLR*.

Ruochen Xu and Yiming Yang. 2017. Cross-lingual distillation for text classification. In *Proceedings of ACL*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *Proceedings of ICLR*.

Sergey Zagoruyko and Nikos Komodakis. 2017. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Proceedings of ICLR*.

Interpretation of Natural Language Rules in Conversational Machine Reading

Marzieh Saeidi[1*], Max Bartolo[1*], Patrick Lewis[1*], Sameer Singh[1,2],
Tim Rocktäschel[3], Mike Sheldon[1], Guillaume Bouchard[1], and Sebastian Riedel[1,3]

[1]Bloomsbury AI
[2]University of California, Irvine
[3]University College London

`{marzieh.saeidi,maxbartolo,patrick.s.h.lewis}@gmail.com`

Abstract

Most work in machine reading focuses on question answering problems where the answer is directly expressed in the text to read. However, many real-world question answering problems require the reading of text not because it contains the literal answer, but because it contains a recipe to derive an answer together with the reader's background knowledge. One example is the task of interpreting regulations to answer "Can I...?" or "Do I have to...?" questions such as "I am working in Canada. Do I have to carry on paying UK National Insurance?" after reading a UK government website about this topic. This task requires both the interpretation of rules and the application of background knowledge. It is further complicated due to the fact that, in practice, most questions are underspecified, and a human assistant will regularly have to ask clarification questions such as "How long have you been working abroad?" when the answer cannot be directly derived from the question and text. In this paper, we formalise this task and develop a crowd-sourcing strategy to collect 32k task instances based on real-world rules and crowd-generated questions and scenarios. We analyse the challenges of this task and assess its difficulty by evaluating the performance of rule-based and machine-learning baselines. We observe promising results when no background knowledge is necessary, and substantial room for improvement whenever background knowledge is needed.

1 Introduction

There has been significant progress in teaching machines to read text and answer questions when the answer is directly expressed in the text (Rajpurkar et al., 2016; Joshi et al., 2017; Welbl et al., 2018; Hermann et al., 2015). However, in many settings, the text contains rules expressed in natural language that can be used to infer the answer when combined with background knowledge, rather than the literal answer. For example, to answer someone's question "I am working for an employer in Canada. Do I need to carry on paying National Insurance?" with "Yes", one needs to read that "You'll carry on paying National Insurance if you're working for an employer outside the EEA" and understand how the rule and question determine the answer.

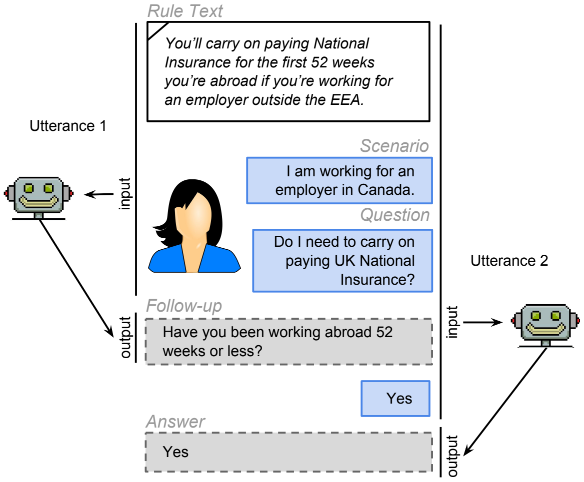

You'll carry on paying National Insurance for the first 52 weeks you're abroad if you're working for an employer outside the EEA.

I am working for an employer in Canada.

Do I need to carry on paying UK National Insurance?

Have you been working abroad 52 weeks or less?

Yes

Yes

Figure 1: An example of two utterances for rule interpretation. In the first utterance, a follow-up question is generated. In the second, the scenario, history and background knowledge (Canada is not in the EEA) is used to arrive at the answer "Yes".

Answering questions that require rule interpretation is often further complicated due to missing information in the question. For example, as illustrated in Figure 1 (Utterance 1), the actual rule also mentions that National Insurance only needs to be paid for the first 52 weeks when abroad. This means that we cannot answer the original question without knowing how long the user has already

---

[*] These three authors contributed equally

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2087–2097
Brussels, Belgium, October 31 - November 4, 2018.

been working abroad. Hence, the correct response in this conversational context is to issue another query such as "Have you been working abroad 52 weeks or less?"

To capture the fact that question answering in the above scenario requires a dialog, we hence consider the following *conversational machine reading* (CMR) problem as displayed in Figure 1: Given an input question, a context scenario of the question, a snippet of supporting rule text containing a rule, and a history of previous follow-up questions and answers, predict the answer to the question ("Yes"or "No") or, if needed, generate a follow-up question whose answer is necessary to answer the original question. Our goal in this paper is to create a corpus for this task, understand its challenges, and develop initial models that can address it.

To collect a dataset for this task, we could give a textual rule to an annotator and ask them to provide an input question, scenario, and dialog in one go. This poses two problems. First, this setup would give us very little control. For example, users would decide which follow-up questions become part of the scenario and which are answered with "Yes" or "No". Ultimately, this can lead to bias because annotators might tend to answer "Yes", or focus on the first condition. Second, the more complex the task, the more likely crowd annotators are to make mistakes. To mitigate these effects, we aim to break up the utterance annotation as much as possible.

We hence develop an annotation protocol in which annotators collaborate with virtual users—agents that give system-produced answers to follow-up questions—to incrementally construct a dialog based on a snippet of rule text and a simple underspecified initial question (e.g., "Do I need to ...?"), and then produce a more elaborate question based on this dialog (e.g., "I am ... Do I need to...?"). By controlling the answers of the virtual user, we control the ratio of "Yes" and "No" answers. And by showing only subsets of the dialog to the annotator that produces the scenario, we can control what the scenario is capturing. The question, rule text and dialogs are then used to produce utterances of the kind we see in Figure 1. Annotators show substantial agreement when constructing dialogs with a three-way annotator agreement at a Fleiss' Kappa level of 0.71.[1] Likewise, we find that our crowd-annotators produce questions that are coherent with the given dialogs with high accuracy.

In theory, the task could be addressed by an end-to-end neural network that encodes the question, history and previous dialog, and then decodes a Yes/No answer or question. In practice, we test this hypothesis using a seq2seq model (Sutskever et al., 2014; Cho et al., 2014), with and without copy mechanisms (Gu et al., 2016) to reflect how follow-up questions often use lexical content from the rule text. We find that despite a training set size of 21,890 training utterances, successful models for this task need a stronger inductive bias due to the inherent challenges of the task: interpreting natural language rules, generating questions, and reasoning with background knowledge. We develop heuristics that can work better in terms of identifying what questions to ask, but they still fail to interpret scenarios correctly. To further motivate the task, we also show in oracle experiments that a CMR system can help humans to answer questions faster and more accurately.

This paper makes the following contributions:

1. We introduce the task of conversational machine reading and provide evaluation metrics.
2. We develop an annotation protocol to collect annotations for conversational machine reading, suitable for use in crowd-sourcing platforms such as Amazon Mechanical Turk.
3. We provide a corpus of over 32k conversational machine reading utterances, from domains such as grant descriptions, traffic laws and benefit programs, and include an analysis of the challenges the corpus poses.
4. We develop and compare several baseline models for the task and subtasks.

2 Task Definition

Figure 1 shows an example of a conversational machine reading problem. A user has a question that relates to a specific rule or part of a regulation, such as "Do I need to carry on paying National Insurance?". In addition, a natural language description of the context or *scenario*, such as "I am working for an employer in Canada", is provided. The question will need to be answered using a small snippet of supporting *rule text*. Akin to machine reading problems in previous work (Rajpurkar et al., 2016; Hermann et al., 2015), we assume that this snippet is pre-identified. We generally assume that the question is *underspecified*, in the sense that the

---

[1] This is well within the range of what is considered as substantial agreement (Artstein and Poesio, 2008).

question often does not provide enough information to be answered directly. However, an agent can use the supporting rule text to infer what needs to be asked in order to determine the final answer. In Figure 1, for example, a reasonable follow-up question is “Have you been working abroad 52 weeks or less?”.

We formalise the above task on a per-utterance basis. A given dialog corresponds to a sequence of prediction problems, one for each utterance the system needs to produce. Let W be a vocabulary. Let $q = w_1 \dots w_{n_q}$ be an *input question* and $r = w_1 \dots w_{n_r}$ an input *support* rule text, where $w_i \in W$ is a word from a vocabulary. Furthermore, let $h = (f_1, a_1) \dots (f_{n_h}, a_{n_h})$ be a dialog *history* where each $f_i \in W^*$ is a *follow-up question*, and each $a_i \in \{\text{YES}, \text{NO}\}$ is a *follow-up answer*. Let s be a scenario describing the context of the question. We will refer to $x = (q, r, h, s)$ as the *input*. Given an input x, our task is to predict an *answer* $y \in \{\text{YES}, \text{NO}, \text{IRRELEVANT}\} \cup W^*$ that specifies whether the answer to the input question, in the context of the rule text and the previous follow-up question dialog, is either YES, NO, IRRELEVANT or another follow-up question in W^*. Here IRRELEVANT is the target answer whenever a rule text is not related to the question q.

3 Annotation Protocol

Our annotation protocol is depicted in Figure 2 and has four high-level stages: Rule Text Extraction, Question Generation, Dialog Generation and Scenario Annotation. We present these stages below, together with discussion of our quality-assurance mechanisms and method to generate negative data. For more details, such as annotation interfaces, we refer the reader to Appendix A.

3.1 Rule Text Extraction Stage

First, we identify the source documents that contain the rules we would like to annotate. Source documents can be found in Appendix C. We then convert each document to a set of *rule texts* using a heuristic which identifies and groups paragraphs and bulleted lists. To preserve readability during the annotation, we also split by a maximum rule text length and a maximum number of bullets.

3.2 Question Generation Stage

For each rule text we ask annotators to come up with an input question. Annotators are instructed to ask questions that cannot be answered directly but instead require follow-up questions. This means that the question should a) match the topic of the support rule text, and b) be underspecified. At present, this part of the annotation is done by expert annotators, but in future work we plan to crowdsource this step as well.

3.3 Dialog Generation Stage

In this stage, we view human annotators as assistants that help users reach the answer to the input question. Because the question was designed to be broad and to omit important information, human annotators will have to ask for this information using the rule text to figure out which question to ask. The follow-up question is then sent to a *virtual user*, *i.e.*, a program that simply generates a random YES or NO answer. If the input question can be answered with this new information, the annotator should enter the respective answer. If not, the annotator should provide the next follow-up question and the process is repeated.

When the virtual user is providing random YES and NO answers in the dialog generation stage, we are traversing a specific branch of a decision tree. We want the corpus to reflect all possible dialogs for each question and rule text. Hence, we ask annotators to label additional branches. For example, if the first annotator received a YES as the answer to the second follow-up question in Figure 3, the second annotator (orange) receives a NO.

3.4 Scenario Annotation Stage

In the final stage, we choose parts of the dialogs created in the previous stage and present this to an annotator. For example, the annotator sees “Are you working or preparing for work?” and NO. They are then asked to write a scenario that is consistent with this dialog such as “I am currently out of work after being laid off from my last job, but am not able to look for any yet.”. The number of questions and answers that the annotator is presented with for generating a scenario can vary from one to the full length of a dialog. Users are encouraged to paraphrase the questions and not to use many words from the dialog.

In an attempt to make these scenarios closer to the real-world situations where a user may provide a lot of unnecessary information to an operator, not only do we present users with one or more questions and answers from a specific dialog but

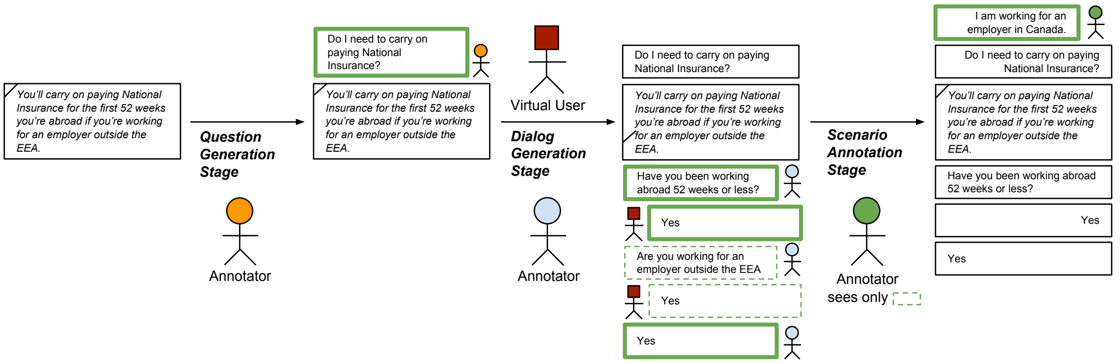

Figure 2: The different stages of the annotation process (excluding the rule text extraction stage). First a human annotator generates an underspecified input question (question generation). Then, a virtual user and a human annotator collaborate to produce a dialog of follow-up questions and answers (dialog generation). Finally, a scenario is generated from parts of the dialog, and these parts are omitted in the final result.

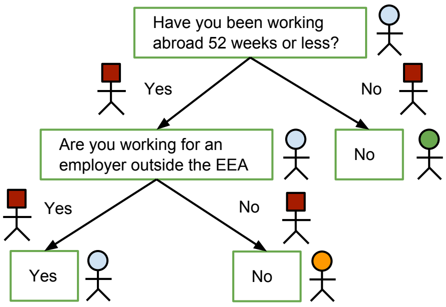

Figure 3: We use different annotators (indicated by different colors) to create the complete dialog tree.

also with one question from a random dialog. The annotators are asked to come up with a scenario that fits all the questions and answers.

Finally, a dialog is produced by combining the scenario with the input question and rule text from the previous stages. In addition, all dialog utterances that were *not* shown to the final annotator are included as well as they complement the information in the scenario. Given a dialog of this form, we can create utterances that are described in Section 2.

As a result of this stage of annotation, we create a corpus of scenarios and questions where the correct answers (YES, NO or IRRELEVANT) to questions can be derived from the related scenarios. This corpus and its challenges will be discussed in Section 4.2.2.

3.5 Negative Examples

To facilitate the future application of the models to large-scale rule-based documents instead of rule text, we deem it to be imperative for the data to contain negative examples of both questions and scenarios.

We define a *negative question* as a question that is not relevant to the rule text. In this case, we expect models to produce the answer IRRELEVANT. For a given rule text and question pair, a negative example is generated by sampling a random question from the set of all possible questions, excluding the question itself and questions sourced from the same document using a methodology similar to the work of Levy et al. (2017).

The data created so far is biased in the sense that when a scenario is given, at least one of the follow-up questions in a dialog can be answered. In practice, we expect users to also provide background scenarios that are completely irrelevant to the input question. Therefore, we sample a *negative scenario* for each input question and rule text pair, (q, r) in our data. We uniformly sample from the scenarios created in Section 3.4 for all question and rule text pairs (q', r') unequal to (q, r). For more details, we point the reader to Appendix D.

3.6 Quality Control

We employ a range of quality control measures throughout the process. In particular, we:

1. Re-annotate pre-terminal nodes in the dialog trees if they have identical YES and NO branches.
2. Ask annotators to validate the previous dialog in case previous utterances where created by different annotators.
3. Assess a sample of annotations for each an-

notator and keep only those annotators with quality scores higher than a certain threshold.

4. We require annotators to pass a qualification test before selecting them for our tasks. We also require high approval rates and restrict location to the UK, US, or Canada.

Further details are provided in Appendix B.

3.7 Cost, Duration and Scalability

The cost of different stages of annotation is as follows. An annotator was paid \$0.15 for an initial question (948 questions), \$0.11 for a dialog part (3000 dialog parts) and \$0.20 for a scenario (6,600 scenarios). It takes in total 2 weeks to complete the annotation process. Considering that all the annotation stages can be done through crowdsourcing and in a relatively short time period and at a reasonable cost using established validation procedures, the dataset can be scaled up without major bottlenecks or an impact on the quality.

4 ShARC

In this section, we present the *SHaping Answers with Rules and Conversation (ShARC)* dataset.[2]

4.1 Dataset Size and Quality

The dataset is built up from of 948 distinct snippets of rule text. Each has an input question and a "dialog tree". At each step in the dialog, there is a followup question posed and the tree branches depending on the answer to the followup question (yes/no). The ShARC dataset is comprised of all individual "utterances" from every tree, i.e. every possible point/node in any dialog tree. There are 6058 of these utterances. In addition, there are 6637 scenarios that provide more information, allowing some questions in the dialog tree to be "skipped" as the answers can be inferred from the scenario. Scenarios therefore modify the dialog trees, which creates new trees. When combined with scenarios and negative sampled scenarios, the total number of distinct utterances became 37087. As a final step, utterances were removed where the scenario referred to a portion of the dialog tree that was unreachable for that utterance, leaving a final dataset size of 32436 utterances.[3]

We break these into train, development and test sets such that each dataset contains approximately the same proportion of sources from each domain, targeting a 70%/10%/20% split.

To evaluate the quality of dialog generation HITs, we sample a subset of 200 rule texts and questions and allow each HIT to be annotated by three distinct workers. In terms of deciding whether the answer is a YES, NO or some follow-up question, the three annotators reach an answer agreement of 72.3%. We also calculate Cohen's Kappa, a measure designed for situations with two annotators. We randomly select two out of the three annotations and compute the unweighted kappa values, repeated for 100 times and averaged to give a value of 0.82.

The above metrics measure whether annotators agree in terms of deciding between YES, NO or some follow-up question, but not whether the follow-up questions are equivalent. To approximate this, we calculate BLEU scores between pairs of annotators when they both predict follow-up questions Generally, we find high agreement: Annotators reach average BLEU scores of 0.71, 0.63, 0.58 and 0.58 for maximum orders of 1, 2, 3 and 4 respectively.

To get an indication of human performance on the sub-task of classifying whether a response should be a YES, NO or FOLLOW-UP QUESTION, we use a similar methodology to (Rajpurkar et al., 2016) by considering the second answer to each question as the human prediction and taking the majority vote as ground truth. The resulting human accuracy is 93.9%.

To evaluate the quality of the scenarios, we sample 100 scenarios randomly and ask two expert annotators to validate them. We perform validation for two cases: 1) scenarios generated by turkers who did not attempt the qualification test and were not filtered by our validation process, 2) scenarios that are generated by turkers who have passed the qualification test and validation process. In the second case, annotators approved an average of 89 of the scenarios whereas in the first case, they only approved an average of 38. This shows that the qualification test and the validation process im-

---

[2] The dataset and its Codalab challenge can be found at `https://sharc-data.github.io`.

[3] One may argue that the the size of the dataset is not sufficient for training end-to-end neural models. While we believe that the availability of large datasets such as SNLI or SQuAD has helped drive the state-of-the-art forward on related tasks, relying solely on large datasets to push the boundaries of AI cannot be as practical as developing better models for incorporating common sense and external knowledge which we believe ShARC is a good test-bed for. Furthermore, the proposed annotation protocol and evaluation procedure can be used to reliably extend the dataset or create datasets for new domains.

proved the quality of the generated scenarios by more than double. In both cases, the annotators agreed on the validity of 91-92 of the scenarios. For further details on dataset quality, the reader if referred to Appendix B.

4.2 Challenges

We analyse the challenges involved in solving conversational machine reading in ShARC. We divide these into two parts: challenges that arise when interpreting rules, and challenges that arise when interpreting scenarios.

4.2.1 Interpreting Rules

When no scenarios are available, the task reduces to a) identifying the follow-up questions within the rule text, b) understanding whether a follow-up question has already been answered in the history, and c) determining the logical structure of the rule (*e.g.*disjunction vs. conjunction vs. conjunction of disjunctions) .

To illustrate the challenges that these sub-tasks involve, we manually categorise a random sample of 100 (q_i, r_i) pairs. We identify 9 phenomena of interest, and estimate their frequency within the corpus. Here we briefly highlight some categories of interest, but full details, including examples, can be found in Appendix G.

A large fraction of problems involve the identification of at least two conditions, and approximately 41% and 27% of the cases involve logical disjunctions and conjunctions respectively. These can appear in linguistic coordination structures as well as bullet points. Often, differentiating between conjunctions and disjunctions is easy when considering bullets—key phrases such as "if all of the following hold" can give this away. However, in 13% of the cases, no such cues are given and we have to rely on language understanding to differentiate. For example:

Q: Do I qualify for Statutory Maternity Leave?
R: You qualify for Statutory Maternity Leave if - you're an employee not a "worker" - you give your employer the correct notice

4.2.2 Interpreting Scenarios

Scenario interpretation can be considered as a multi-sentence entailment task. Given a scenario (premise) of (usually) several sentences, and a question (hypothesis), a system should output YES (ENTAILMENT), NO (CONTRADICTION) or IRRELEVANT (NEUTRAL). In this context, IRRELEVANT indicates that the answer to the question cannot be inferred from the scenario.

Different types of reasoning are required to interpret the scenarios. Examples include numerical reasoning, temporal reasoning and implication (common sense and external knowledge). We manually label 100 scenarios with the type of reasoning required to answer their questions. Table 1 shows examples of different types of reasoning and their percentages. Note that these percentages do not add up to 100% as interpreting a scenario may require more than one type of reasoning.

5 Experiments

To assess the difficulty of ShARC as a machine learning problem, we investigate a set of baseline approaches on the end-to-end task as well as the important sub-tasks we identified. The baselines are chosen to assess and demonstrate both feasibility and difficulty of the tasks.

Metrics For all following classification tasks, we use micro- and macro- averaged accuracies. For the follow-up generation task, we compute the BLEU scores at orders 1, 2, 3 and 4 computed between the gold follow-up questions, y_i and follow-up question $\hat{y}_i = w_{\hat{y}_i,1}, w_{\hat{y}_i,2} \ldots w_{\hat{y}_i,n}$ for all utterances i in the evaluation dataset.

5.1 Classification (excluding Scenarios)

On each turn, a CMR system needs to decide, either explicitly or implicitly, whether the answer is YES or NO, whether the question is not relevant to the rule text (IRRELEVANT), or whether a follow-up question is necessary—an outcome we label as MORE. In the following experiments, we will test whether one can learn to make this decision using the ShARC training data.

When a non-empty scenario is given, this task also requires an understanding of how scenarios answer follow-up questions. In order to focus on the challenges of rule interpretation, here we only consider empty scenarios.

Formally, for an utterance $x = (q, r, h, s)$, we require models to predict an answer y where $y \in \{\text{YES}, \text{NO}, \text{IRRELEVANT}, \text{MORE}\}$. Since we consider only the classification task without scenario influence, we consider the subset of utterances such that $s = NULL$. This data subset consists of 4026 train, 431 dev and 1601 test utterances.

Category	Questions	Scenario	%
Explicit	Has your wife reached state pension age? <u>Yes</u>	My wife just recently reached the age for state pension	25%
Temporal	Did you own it before April 1982? <u>Yes</u>	I purchased the property on June 5, 1980.	10%
Geographic	Do you normally live in the UK? <u>No</u>	I'm a resident of Germany.	7%
Numeric	Do you work less than 24 hours a week between you? <u>No</u>	My wife and I work long hours and get between 90 - 110 hours per week between the two of us.	12%
Paraphrase	Are you working or preparing for work? <u>No</u>	I am currently out of work after being laid off from my last job, but am not able to look for any yet.	19%
Implication	Are you the baby's father? <u>No</u>	My girlfriend is having a baby by her ex.	51%

Table 1: Types of reasoning and their proportions in the dataset based on 100 samples. Implication includes reasoning beyond what is explicitly stated in the text, including common sense reasoning and external knowledge.

Model	Micro Acc.	Macro Acc.
Random	0.254	0.250
Surface LR	0.555	0.511
Heuristic	0.791	0.779
Random Forest	**0.808**	**0.797**
CNN	0.677	0.681

Table 2: Selected Results of the baseline models on the classification sub-task.

Baselines We evaluate various baselines including random, a surface logistic regression applied to a TFIDF representation of the rule text, question and history, a rule-based heuristic which makes predictions depending on the number of overlapping words between the rule text and question, detecting conjunctive or disjunctive rules, detecting negative mismatch between the rule text and the question and what the answer to the last follow-up history was, a feature-engineered Random Forest and a Convolutional Neural Network applied to the tokenised inputs of the concatenated rule text, question and history.

Results We find that, for this classification sub-task, Random Forest slightly outperforms the heuristic. All learnt models considerably outperform the random and majority baselines.

5.2 Follow-up Question Generation without Scenarios

When the target utterance is a follow-up question, we still have to determine what that follow-up question is. For an utterance $x = (q, r, h, s)$, we require models to predict an answer y where y is the next follow-up question, $y = w_{y,1}, w_{y,2} \dots w_{y,n} = f_{m+1}$ if x has history of length m. We therefore consider the subset of utterances such that $s = NULL$ and $y \notin \{\textsc{Yes}, \textsc{No}, \textsc{Irrelevant}\}$. This data subset consists of 1071 train, 112 dev and 424 test utterances.

Baselines We first consider several simple baselines to explore the relationship between our evaluation metric and the task. As annotators are encouraged to re-use the words from rule text when generating follow-up questions, a baseline that simply returns the final sentence of the rule text performs surprisingly well. We also implement a rule-based model that uses several heuristics.

If framed as a seq2seq task, a modified CopyNet is most promising (Gu et al., 2016). We also experiment with span extraction/sequence-tagging approaches to identify relevant spans from the rule text that correspond to the next follow-up questions. We find that Bidirectional Attention Flow (Seo et al., 2017) performed well.[4] Further implementation details can be found in Appendix H.

Results Our results, shown in Table 3 indicate that systems that return contiguous spans from the rule text perform better according to our BLEU metric. We speculate that the logical forms in the data are challenging for existing models to extract and manipulate, which may suggest why the explicit rule-based system performed best. We further note that only the rule-based and NMT-Copy models are capable of generating genuine questions rather than spans or sentences.

5.3 Scenario Interpretation

Many utterances require the interpretation of the scenario associated with a question. If the scenario

---

[4] We use AllenNLP implementations of BiDAF & DAM

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4
First Sent.	0.221	0.144	0.119	0.106
NMT-Copy	0.339	0.206	0.139	0.102
BiDAF	0.450	0.375	0.338	0.312
Rule-based	**0.533**	**0.437**	**0.379**	**0.344**

Table 3: Selected Results of the baseline models on follow-up question generation.

Model	Micro Acc.	Macro Acc.
Random	0.330	0.326
Surface LR	**0.682**	0.333
DAM (SNLI)	0.479	**0.362**
DAM (ShARC)	0.492	0.322

Table 4: Results of entailment models on ShARC.

is understood, certain follow-up questions can be skipped because they are answered within the scenario. In this section, we investigate how difficult scenario interpretation is by training models to answer follow-up questions based on scenarios.

Baselines We use a random baseline and also implement a surface logistic regression applied to a TFIDF representation of the combined scenario and the question. For neural models, we use Decomposed Attention Model (DAM) (Parikh et al., 2016) trained on each the SNLI and ShARC corpora using ELMO embeddings (Peters et al., 2018).[4]

Results Table 4 shows the result of our baseline models on the entailment corpus of ShARC test set. Results show poor performance especially for the macro accuracy metric of both simple baselines and neural state-of-the-art entailment models. This performance highlights the challenges that the scenario interpretation task of ShARC presents, many of which are discussed in Section 4.2.2.

5.4 Conversational Machine Reading

The CMR task requires all of the above abilities. To understand its core challenges, we compare baselines that are trained end-to-end vs. baselines that reuse solutions for the above subtasks.

Baselines We present a Combined Model (CM) which is a pipeline of the best performing Random Forest classification model, rule-based follow-up question generation model and Surface LR entailment model. We first run the classification model to predict YES, NO, MORE or IRRELEVANT. If MORE is predicted, the Follow-up Question Generation model is used to produce a follow-up question, f_1. The rule text and produced follow-up question are then passed as inputs to the Scenario Interpretation model. If the output of this is IRRELEVANT, then the CM predicts f_1, otherwise, these steps are repeated recursively until the classification model no longer predicts MORE or the entailment model predicts IRRELEVANT, in which case the model produces a final answer. We also investigate an extension of the NMT-copy model on the end-to-end task. Input sequences are encoded as a concatenation of the rule text, question, scenario and history. The model consists of a shared encoder LSTM, a 4-class classification head with attention, and a decoder GRU to generate followup questions. The model was trained by alternating training the classifier via standard softmax-cross entropy loss and the followup generator via seq2seq. At test time, the input is first classified, and if the predicted class is MORE, the follow-up generator is used to generate a followup question, f_1. A simpler model without the separate classification head failed to produce predictive results.

Model	**Micro Acc**	**Macro Acc**	**BLEU-1**	**BLEU-4**
CM	0.619	0.689	0.544	0.344
NMT	0.448	0.428	0.340	0.078

Table 5: Results of the models on the CMR task.

Results We find that the combined model outperforms the neural end-to-end model on the CMR task, however, the fact that the neural model has learned to classify better than random and also predict follow-up questions is encouraging for designing more sophisticated neural models for this task.

User Study In order to evaluate the utility of conversational machine reading, we run a user study that compares CMR to when such an agent is not available, i.e. the user has to read the rule text and determine themselves the answer to the question. On the other hand, with the agent, the user does not read the rule text, instead only responds to follow-up questions. Our results show that users using the conversational agent reach conclusions > 2 times faster than ones that are not, but more importantly, they are also much more accurate (93% as compared to 68%). Details of the experiments and the results are included in Appendix I.

6 Related Work

This work relates to several areas of active research.

Machine Reading In our task, systems answer questions about units of texts. In this sense, it is most related to work in Machine Reading (Rajpurkar et al., 2016; Seo et al., 2017; Weissenborn et al., 2017). The core difference lies in the conversational nature of our task: in traditional Machine Reading the questions can be answered right away; in our setting, clarification questions are often needed. The domain of text we consider is also different (regulatory vs Wikipedia, books, newswire).

Dialog The task we propose is, at its heart, about conducting a dialog (Weizenbaum, 1966; Serban et al., 2018; Bordes and Weston, 2016). Within this scope, our work is closest to work in dialog-based QA where complex information needs are addressed using a series of questions. In this space, previous approaches have been looking primarily at QA dialogs about images (Das et al., 2017) and knowledge graphs (Saha et al., 2018; Iyyer et al., 2017). In parallel to our work, both Choi et al. (2018) and Reddy et al. (2018) have to began to investigate QA dialogs with background text. Our work not only differs in the domain covered (regulatory text vs wikipedia), but also in the fact that our task requires the interpretation of complex rules, application of background knowledge, and the formulation of free-form clarification questions. Rao and Daume III (2018) does investigate how to generate clarification questions but this does not require the understanding of explicit natural language rules.

Rule Extraction From Text There is a long line of work in the automatic extraction of rules from text (Silvestro, 1988; Moulin and Rousseau, 1992; Delisle et al., 1994; Hassanpour et al., 2011; Moulin and Rousseau, 1992). The work tackles a similar problem—interpretation of rules and regulatory text—but frames it as a text-to-structure task as opposed to end-to-end question-answering. For example, Delisle et al. (1994) maps text to horn clauses. This can be very effective, and good results are reported, but suffers from the general problem of such approaches: they require careful ontology building, layers of error-prone linguistic preprocessing, and are difficult for non-experts to create annotations for.

Question Generation Our task involves the automatic generation of natural language questions. Previous work in question generation has focussed on producing questions for a given text, such that the questions can be answered using this text (Vanderwende, 2008; M. Olney et al., 2012; Rus et al., 2011). In our case, the questions to generate are *derived* from the background text but cannot be answered by them. Mostafazadeh et al. (2016) investigate how to generate natural follow-up questions based on the content of an image. Besides not working in a visual context, our task is also different because we see question generation as a sub-task of question answering.

7 Conclusion

In this paper we present a new task as well as an annotation protocol, a dataset, and a set of baselines. The task is challenging and requires models to generate language, copy tokens, and make logical inferences. Through the use of an interactive and dialog-based annotation interface, we achieve good agreement rates at a low cost. Initial baseline results suggest that substantial improvements are possible and require sophisticated integration of entailment-like reasoning and question generation.

Acknowledgements

This work was supported by an Allen Distinguished Investigator Award.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC : Question Answering in Context. In *EMNLP*. ArXiv: 1808.07036.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, Jos MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2.

Sylvain Delisle, Ken Barker, Jean franois Delannoy, Stan Matwin, and Stan Szpakowicz. 1994. From text to horn clauses: Combining linguistic analysis and machine learning. In *In 10th Canadian AI Conf*, pages 9–16.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393.

Saeed Hassanpour, Martin O'Connor, and Amar Das. 2011. A framework for the automatic extraction of rules from online text.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based Neural Structured Learning for Sequential Question Answering. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1:1821–1831.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Andrew M. Olney, Arthur Graesser, and Natalie Person. 2012. Question generation from concept maps. 3.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. *CoRR*, abs/1603.06059.

B. Moulin and D. Rousseau. 1992. Automated knowledge acquisition from regulatory texts. *IEEE Expert*, 7(5):27–35.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Sudha Rao and Hal Daume III. 2018. Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2737–2746, Melbourne, Australia. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2018. CoQA: A Conversational Question Answering Challenge. *arXiv:1808.07042 [cs]*. ArXiv: 1808.07042 Citation Key: reddyCoQAConversationalQuestion2018.

Vasile Rus, Paul Piwek, Svetlana Stoyanchev, Brendan Wyse, Mihai Lintean, and Cristian Moldovan. 2011. Question generation shared task and evaluation challenge: Status report. In *Proceedings of the 13th European Workshop on Natural Language Generation*, ENLG '11, pages 318–320, Stroudsburg, PA, USA. Association for Computational Linguistics.

Amrita Saha, Vardaan Pahuja, Mitesh Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex Sequential Question Answering: Towards Learning to Converse Over Linked Question Answer Pairs with a Knowledge Graph.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *The International Conference on Learning Representations (ICLR)*.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2018. A Survey of Available Corpora For Building Data-Driven Dialogue Systems: The Journal Version. *Dialogue & Discourse*, 9(1):1–49.

Kenneth Silvestro. 1988. Using explanations for knowledge-base acquisition. *International Journal of Man-Machine Studies*, 29(2):159 – 169.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Lucy Vanderwende. 2008. The importance of being important: Question generation. In *In Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Fastqa: A simple and efficient neural architecture for question answering. *CoRR*, abs/1703.04816.

Joseph Weizenbaum. 1966. ELIZAa computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of ACL*, abs/1710.06481.

A State-transition Framework to Answer Complex Questions over Knowledge Base

Sen Hu[1], **Lei Zou**[1,2], **Xinbo Zhang**[1]
[1]Peking University, China; [2]Beijing Institute of Big Data Research, China;
{husen,zoulei,zhangxinbo}@pku.edu.cn

Abstract

Although natural language question answering over knowledge graphs have been studied in the literature, existing methods have some limitations in answering complex questions. To address that, in this paper, we propose a *State Transition*-based approach to translate a complex natural language question N to a *semantic query graph* (SQG) Q^S, which is used to match the underlying knowledge graph to find the answers to question N. In order to generate Q^S, we propose four primitive operations (expand, fold, connect and merge) and a learning-based state transition approach. Extensive experiments on several benchmarks (such as QALD, WebQuestions and ComplexQuestions) with two knowledge bases (DBpedia and Freebase) confirm the superiority of our approach compared with state-of-the-arts.

1 Introduction

Complex Question Answering, in which the question corresponds to multiple triples in knowledge base, has attracted researchers' attentions recently (Bao et al., 2014; Xu et al., 2016; Berant and Liang, 2014; Bast and Haussmann, 2015; Yih et al., 2015). However, most of existing solutions employ the predefined patterns or templates in understanding complex questions. For example, (Bao et al., 2014; Xu et al., 2016) use dependency tree patterns to decompose a complex question to several simple questions. (Berant and Liang, 2014; Bast and Haussmann, 2015) rely on templates to translate natural language sentences to predefined logical forms. Obviously, it is impossible to cover all real-world cases using those handcrafted templates. The recent work STAGG (Yih et al., 2015) proposes a state transition strategy to tackle complex questions. However, the generated query graph's structure in STAGG is limited, as shown in Figure 1. Generally, the structure is formed by linking the topic entity e and the answer node t through a core relation. It only allows entity constraints (such as c) and one variable (the answer node t), the expressivity is quite limited. It cannot cover all types of complex questions such as the question in Figure 2, whose query graph contains variable constraint (v_4) and multi-edges (between v_1 and v_4). Therefore, in this paper, we propose a more flexible strategy to answer complex questions without handcrafted templates or restricting the query graph's structure.

1.1 Properties and Challenges of Complex Questions

We first analyze the inherent challenges for complex question answering, which has not been well studied in existing literatures.

Multi or Implicit relations. A complex question has multiple relations and some relations may be multi-hop or implicit. Consider the question N_1 = "*Which Russian astronauts died in the same place they were born in?*". Obviously there are two explicit relations represented by the two phrases "died in" and "born in", but many traditional methods only extract a single relation between topic entity and the answer node. On the other hand, there is an implicit relation between "Russian" and "astronauts" (i.e., ⟨Nationality⟩), which lacks a corresponding relation phrase.

Multi or No entities. Different from simple (one-hop) questions that only contain a single *topic entity* in the question sentence N, complex questions may have multiple entities. For example, "Which Russian astronauts were died in Moscow and born in Soviet Union". There are three entities "Russian", "Moscow" and "Soviet Union". In some cases, the question sentence does not include any entity, such as "Who died in the same place they were born in". Therefore, existing methods that

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2098–2108
Brussels, Belgium, October 31 - November 4, 2018.

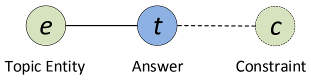

Figure 1: The query graph structure of STAGG. Node *e* is the "topic entity", path *e-t* is the "core inferential chain" and *c* represents "augmenting constraints". Notice we omit the CVT node as it is a virtual node only existed in Freebase.

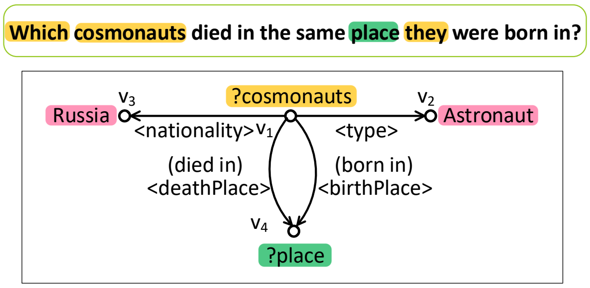

Figure 2: The example of complex questions with semantic query graph

only care about a single topic entity cannot work for such types of complex questions.

Variables and Co-reference. Existing solutions assume that there is a single variable in the question N, i.e., the wh-word, which is focus of the question (representing the answer node). However, complex questions may have other variables. Consider the question in Figure 2, there are four variables "Which", "cosmonauts", "place" and "they" recognized. While the three in yellow grid ("Which", "cosmonaut", "they") refer to the same thing, which is the co-reference phenomenon.

Composition. In the simple questions, the query graph's structure is determinate (one-hop edge). For complex questions, even when all entities, variables and relations are recognized, how to assemble them to logical forms or executable queries (query graphs) is still a challenge. As mentioned earlier, existing solutions that depend on predefined templates (such as (Xu et al., 2016; Bast and Haussmann, 2015)) or query graph structure patterns (such as (Yih et al., 2015)) may not be suitable due to the diversity of complex questions.

1.2 Our Solution

Considering the above challenges, we propose a *state transition* based framework to translate users' question N into a semantic query graph (SQG) Q^S, further the answers of N can be found by matching Q^S over the knowledge graph G. Specifically, we first recognize nodes (such as entities and variables) from N as the initial state (Section 2.1). Then, to enable and speed up the state transition process, we propose four primitive operations with corresponding conditions (Section 2.2). The entity/relation candidates are extracted using existing entity linking algorithms and our proposed MCCNN model (Section 2.3). To guide the state transition, we learn a reward function using SVM ranking algorithm (Section 3) to greedily select a subsequent state.

Compared with existing solutions, our framework has stronger expressivity to solve more types of complex questions (considering the challenges introduced in Section 1.1) and does not rely on handcrafted templates.

2 Semantic Query Graph Generation

This section outlines the whole framework of our system. Given a natural language question N, we aim to translate N into a *semantic query graph* (SQG) (see Definition 1).

Definition 1 ***(Semantic Query Graph).*** *A semantic query graph (denoted as* Q^S*) is a connected graph, in which each vertex* v_i *corresponds a phrase in the question sentence* N *and associated with a list of entity/type/literal candidates; and each edge* $\overline{v_iv_j}$ *is associated with a list of relation candidates,* $1 \leq i, j \leq |V(Q^S)|$.

Given a question sentence N_1 = "Which cosmonauts died in the same place they were born in?", the corresponding semantic query graph Q^S_1 is given in Figure 2. In Q^S_1, the subgraph $\{v_1$, v_2 and $v_3\}$ corresponds "cosmonauts", v_4 corresponds "place". The relation phrases "born in" and "died in" denote two relations between v_1 and v_4. Each edge in Q^S has a list of relation(predicate) candidates with confidence probabilities. In this paper, we only draw one relation candidate of each edge for simplicity. Notice that the edge $\overline{v_1v_3}$ and $\overline{v_1v_2}$ have no relation phrases, however, they still have corresponding relation candidates $\langle$nationality$\rangle$ and $\langle$type$\rangle$.

After obtaining semantic query graph Q^S, we find matches of Q^S over the underlying knowledge graph G to find answers to the original question N. This stage is identical to gAnswer (Zou et al., 2014) and NFF (Hu et al., 2018). Thus, in this paper, we concentrate ourselves on discussing how to generate semantic query graph Q^S, which is different from NFF that relies on the paraphrasing dictionaries and heuristic rules to build Q^S.

2.1 Node Recognition

The first step is to detect the "nodes" (for building SQG Q^S) in users' question sentences. According to Definition 1, each node in the semantic query graph is a phrase corresponding to a subject or object in the SPARQL query. Generally, constant nodes (e.g., *Titanic*) can be divided into three categories according to their roles (entity, type, literal) in knowledge graph. Besides, a variable node (e.g., *?place*) has the potential to map any vertices in knowledge graph.

Generally, the node recognition task is regarded as a sequence labeling problem and we adopt a BLSTM-CRF model (Huang et al., 2015) to realize it. However, this model doesn't work very well on recognizing entity/type nodes as the entity phrases may be too long or too complex. Therefore, we first detect entity and type nodes by utilize existing entity linking algorithms of specific knowledge bases (see section 2.3), and then detect variable/literal nodes by the BLSTM-CRF model. A phrase w_{ij}, which contains the words from i to j in the question, is recognized as entity/type node if we can find entity/type mappings of w_{ij} through the entity linking algorithm. We check all possible phrases and select the longer one if two candidate nodes share any words. The detected entity/type nodes will be replaced by special tokens $\langle E\rangle$ or $\langle T\rangle$ before calling the BLSTM-CRF model.

In practice, we find that some nodes have hidden information, which can be expanded to a subgraph (i.e, one or multiple triples). For example, the hidden information of "?cosmonauts" in question N_1 in Figure 2 can extends two triples: ⟨?cosmonauts rdf:type dbo:Astronaut⟩ and ⟨?cosmonauts dbo:nationality dbr:Russia⟩. Such information can help us to reduce the matching space and make the answers more accurate.

We mine those nodes with hidden information from several QA benchmarks[1] and build a translation dictionary D^T in offline. Given a question N with the gold query graph Q^G, we try to match Q^G to the SQG Q^S generated by our approach. Each unmatched connected subgraph in Q^G is regarded as hidden information of the nearest matched node v. For the benchmarks only have gold question-answers pairs, we utilize the method in (Zhang and Zou, 2017) to mine the gold query graph first.

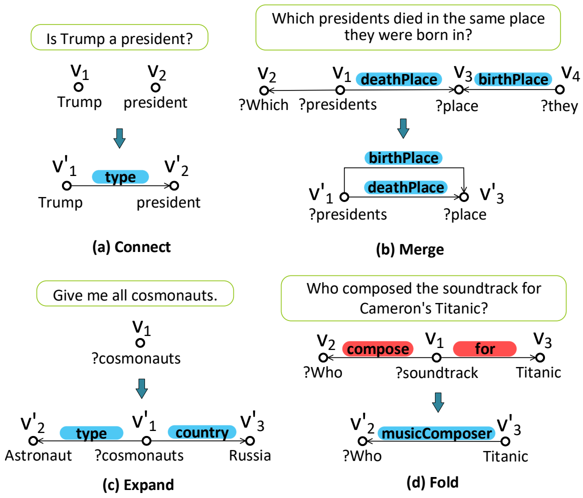

Figure 3: Samples of Operations

2.2 SQG's Structure Construction

As mentioned earlier, our SQG's structure is generated in a state-transition paradigm. Initially, the state only contains several *isolated* nodes that are recognized in the first step. To enable the state transition, we propose four primitive operations as follows.

1. **Connect**. Given two operate nodes u_1 and u_2, we introduce an edge $\overline{u_1u_2}$ between them by the connect operation. Note that the relation mention and candidate relations (predicates) of the edge $\overline{u_1u_2}$ can be found through relation extraction model (see section 2.3).

2. **Merge**. Given a SQG $Q^S = \{V, E\}$ and two operate nodes u,v, this operation is to merge the node u into v. The new SQG $Q'^S = \{V \setminus \{u\}, (E \setminus E^d) \cup E^a\}$, $E^d = \{\overline{uw} \in E\}$ and $E^a = \{\overline{vw} | \overline{uw} \in E \wedge w \neq v\}$. Notice that v also inherits the properties of u. The merge operation is designed to support co-reference resolution. The Figure 3(b) gives an example. For the question "Which presidents died in the same place they were born in?", the node v_2("which") is redundant and can be merged into v_1("presidents"). The node v_4("they") and v_1("presidents") are co-reference, we can merge v_4 to v_1. The edge $\overline{v_3v_4}$(birthPlace) is replaced by edge $\overline{v_3'v_1'}$(birthPlace).

3. **Expand**. Given a SQG $Q^S = \{V, E\}$ and the operate node $u \in V$, this operation is to expand u to a subgraph $Q_u^S = \{V_u, E_u\}$. The new SQG $Q'^S = \{V \cup V_u, E \cup E_u\}$. The expand operation is designed to those

---

[1] QALD, WebQuestions and ComplexQuestions, only the training sets are used to build the translation dictionary D^T.

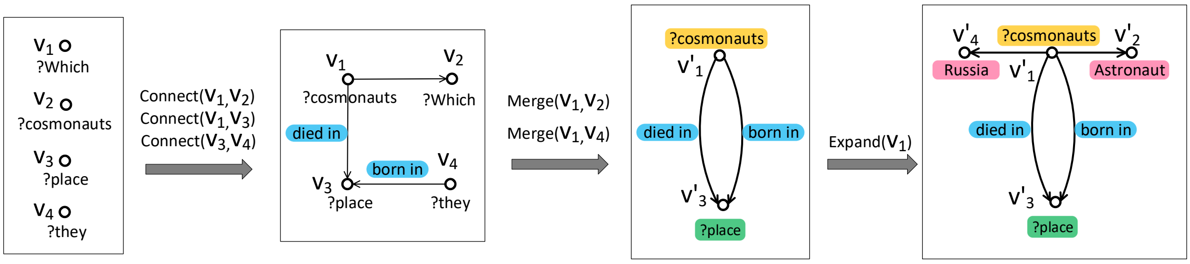

Figure 4: Semantic Query Graph Generation

nodes that have hidden information. Let us see Figure 3(c). The word "cosmonauts"(v_1) means "Russia astronauts", but it cannot be mapped to a certain entity or type in the knowledge base directly. To match "cosmonauts", we can expand it to an equivalent subgraph $\{v_1^{'}, v_2^{'}, v_3^{'}\}$ (according to the transition dictionary D^T) that can match the underlying knowledge base.

4. **Fold**. Given a SQG $Q^S = \{V, E\}$ and an operate node $u \in V$, this operation is to delete u and merge the associated relations. Formally, the new SQG $Q'^S = \{V \setminus \{u\}, (E \setminus E^d) \cup E^a\}$, in which $E^d = \{\overline{uv_i} \in E\}$ and $E^a = \{\overline{v_i v_j} | \overline{uv_i} \in E \wedge \overline{uv_j} \in E \wedge i \neq j\}$. The fold operation is designed to eliminate those nodes which are useless or mis-recognized. Figure 3(d) gives an example. For the question "Who composed the soundtrack for Cameron's Titanic?", SQG $\{v_1, v_2, v_3\}$ cannot find matches in knowledge base because there are no correct relations of $\overline{v_1 v_2}$ and $\overline{v_1 v_3}$. By applying a fold operation, these two edges with the redundant node v_1 are removed, while the new edge $\overline{v'_1 v'_2}$ can be matched to the correct relation $\langle$musicComposer$\rangle$ (according to the relation extraction model).

We focus on generating the correct SQG with the above operations. Obviously, it is impossible to enumerate all possible transitions in each step due to exponential search space. Therefore, we propose a greedy search algorithm[2] inspired by (Yih et al., 2015). The intention is that an better intermediate state should have more opportunities to be visited and produce subsequent states. As each state can be seen as a partial SQG, we propose a reward function using a log-linear model (see Section 3) to estimate the likelihood of each state. A priority queue is utilized to maintain the order of unvisited states. Each turn we choose the state with maximum correctness likelihood and try all possible transitions by each operation. Specifically, the connect and merge operations try every pair of nodes in current state as the two operate nodes, while the expand and fold operations try each node as the operate node. The newly generated states are estimated by the learning model and pushed into the priority queue. Note that reduplicated states will not be added. The algorithm ends when a complete SQG Q^S is found and Q^S can not generates any better subsequent states.

To reduce the search space of the state-transition process, we also propose several constraints for each operation. Only when the conditions are satisfied, the corresponding operation can be executed. Experiments show that those conditions not only speeds up the QA process but also improves the precision (see Section 4.4). The conditions are listed as follows.

Condition 1 (**Condition for Connect**) *Two nodes v_1 and v_2 can be connected if and only if there exists* no *other node v^* that occurs in the shortest path between v_1 and v_2 of the dependency parse tree*[3] *of question sentence N.*

Condition 2 (**Condition for Merge**) *For the merge operation, there should be at least one variable v in the two operate nodes. And v should be a wh-word or pronoun, which may co-reference*[3] *with an other node.*

Condition 3 (**Condition for Expand**) *For the expand operation, the operate node u should be a variable, and we need to be able to find u and its hidden information in the transition dictionary D^T.*

Condition 4 (**Condition for Fold**) *For the fold operation, we require at least one of the connected edges with the operate node have no relation candidates or the confidence probabilities of the corresponding relations are less than a threshold τ.*

---

[2] The formal algorithm can be found in Appendix A.

[3] We utilize the Stanford CoreNLP dependency parser and coreference annotator (Manning et al., 2014)

The following example illustrates the process of SQG generation. Note that it can have other transition sequences to get the final state.

Example 1 *Consider Figure 4. Given a question* N_1 *= "Which cosmonauts died in the same place they were born in?", we first extract four nodes* v_1*,* v_2*,* v_3*,* v_4*, which are all variable nodes. According the condition 1, we connect (*v_1*,*v_2*), (*v_1*,*v_3*), and (*v_3*,*v_4*). The simple path between* v_1 *and* v_3 *in the dependency parse tree can be regarded as the relation phrase of* $\overline{v_1 v_3}$*, which is "died in". Similarly, the relation phrase of* $\overline{v_3 v_4}$ *is "born in". As* v_1 *and* v_2 *are neighbors in the dependency parse tree, the corresponding relation phrase is empty. Then we merge (*v_1*,*v_2*) as they are recognized co-reference. Similarly* v_1 *and* v_4 *are merged. Notice that the associated edge* $\overline{v_3 v_4}$ *have been reserved, then there are two edges between* v_1' *and* v_3' *in the new SQG. In the end, we expand* v_1' *to the subgraph* $\{v_1', v_4', v_2'\}$ *and get the final SQG.*

2.3 Finding entity/relation candidates and matches of SQG

During SQG construction, there are two subtasks: entity extraction and relation extraction. We briefly discuss the two steps as follows.

Given a node phrase in Q^S, we find the candidate entities/types with confidence probabilities by using existing entity linking approaches. Specifically, we use S-MART (Yang and Chang, 2015) for Freebase dataset and DBpeida Lookup[4] for DBpedia dataset.

Given two connected nodes v_i and v_j in Q^S, we need to find the relation candidates between them. Inspired by the recent success of neural relation extraction models in KBQA (Yih et al., 2015; Dong et al., 2015; Xu et al., 2016), we propose a Multi-Channel Convolutional Neural Network (MCCNN) to extract all relations in the question N exploiting both syntactic and sentential information. We use the simple path between v_i and v_j in the dependency tree $Y(N)$ as input to the first channel. The input of second channel is the whole question sentence excluded all nodes, which represents the context of the relation we want to predict. If v_i and v_j are neighbors in $Y(N)$, we need to extract an implicit relation. To tackle this task, we use v_i, v_j and their types as input to the third channel. For v_i = "Chinese" and v_j = "actor", the input is "Chinese-Country-actor-Actor".

Different from (Xu et al., 2016) which can only extract one explicit relation, our model can not only extract multiple relations from N by taking different node pairs, but also extract the implicit relation considering the embedded information in the two nodes v_i and v_j.

After generating the SQG Q^S, we employ the subgraph matching algorithm (Hu et al., 2018) to find matches of Q^S over the underlying RDF knowledge graph G. Once the matches are found, it is straightforward to derive the answers.

3 Learning the reward function

Given an intermediate state s during SQG generation process, we may transit to multiple subsequent states s' by applying different operations. Thus, we greedily select a subsequent state s' with the maximum $\gamma(s')$, where $\gamma()$ is a reward function taking the features and outputting the reward of corresponding state. In this work, the reward function $\gamma()$ is a linear model trained with the SVM ranking loss. Below we describe the features, learning process and how to generate the training data.

3.1 Features

Note that for any intermediate state s, we also regard it as a SQG $Q^S(s)$ even though it is a disconnected graph. For a given SQG Q^S, we extract the following features which are passed as an input vector to the SVM ranker.

Nodes. For each entity/type node v, we utilize entity linking systems to find the candidate entities or types and corresponding confidence probabilities. We use the largest confidence probability as the score of v, and use the average score of all entity nodes as the feature. In addition, the number of constant nodes and variable nodes in current Q^S are considered as features.

Edges. For each edge $\overline{v_i v_j}$, the relation extraction model returns a list of candidate predicates with the corresponding confidence probabilities. We use the largest confidence probability as the score of $\overline{v_i v_j}$, and use the average score of all edges as the feature. We also consider the total number of edges in Q^S, the number of edges which have relation phrase and which have no relation phrase.

Matches. We also consider the matching between Q^S and the knowledge graph G. A new state s would get a low score if its corresponding

---

[4] http://wiki.dbpedia.org/projects/dbpedia-lookup

SQG Q^S could not find any matches in G. We do not discard s directly from the search queue as we allow the fold operation of it. To speed up the matching process, we build the index of all entities and relations in offline. For a given SQG Q^S, we use the offline index to check whether it is valid.

Given a valid state s, if all of its possible subsequent states s' have smaller reward function value than that of s, we will terminate the state transition process and return s as the final state. The corresponding SQG $Q^S(s)$ is used to match the knowledge base to retrieve answers.

3.2 Learning

The task of reward function can be viewed as a ranking problem, in which the appropriate SQGs should be ranked in the front. In this work, we rank them using a SVM rank classifier (Joachims, 2006).

Given a SQG $Q^S = \{V, E\}$, the ranking score that we use to train our reward function $\gamma()$ is calculated by the following function.

$$R(Q^S) = \frac{|P(V)|}{|V'|} + \frac{|P(E)|}{|E'|} + \max(F(A_i, A')) \tag{1}$$

V' and E' are the gold node set and relation set extracted from the gold SPARQL query. $P(V)$ contains the nodes existing in both V and V'. $P(E)$ contains the relations existing in both E and E'. $F(A_i, A')$ is the F-1 measure where A' is the gold answer set and A_i is one of the subgraph matches between Q^S and knowledge graph G obtained by the executing method in (Hu et al., 2018). Notice that for the benchmark without gold SPARQL queries, $R(Q^S) = max(F(A_i, A'))$.

3.3 Generating Training Data

To generate the training data of the reward function, given a question $N = \{w_1, w_2, ..., w_n\}$, we first check all possible phrases w_{ij} by existing entity linking system and get the candidate entity list $C_{v_i} = \{e_1, e_2, ..., e_m\}$ for each entity node v_i. If two entity node v_i and v_j have conflict, i.e, v_i and v_j share one or more words, we reserve the longer one. We replace each entity node v_i with a special token to denote the type of entity e_j, where $e_j \in C_{v_i}$ and has the largest confidence possibility. Then the whole node set V can be predicted by the well-trained node recognition model. $Q^S = \{V, \phi\}$ is the initial state. Further we do state transition by applying the predefined operations using a BFS algorithm. Meanwhile, only the states satisfied with corresponding conditions are considered if enabling the condition mechanism. Once we get a new Q^S we call the well-trained relation extraction model to get the relation candidate list of each new edge $\overline{v_i v_j} \in E$, while the features are collected according Section 3.1 and the ranking score is calculated using Equation 1.

4 Evaluation

We evaluate our system on several benchmarks with two knowledge base DBpedia and Freebase. For DBpedia, we use the QALD-6 benchmark. We compare our method STF (State Transition Framework) with all systems in QALD-6 competition as well as Aqqu (Bast and Haussmann, 2015), gAnswer (Zou et al., 2014) and NFF (Hu et al., 2018). For Freebase, we use WebQuestions (Berant et al., 2013) and ComplexQuestions (Abujabal et al., 2017) as benchmarks and compare our method with corresponding state-of-art systems.

4.1 Setup

4.1.1 Knowledge Bases

DBpedia RDF repository is a knowledge base derived from Wikipedia (Bizer et al., 2009). We use the version of DBpedia 2014 and the statistics are given in Table 1.

Freebase is a collaboratively edited knowledge base. We use the version of Freebase 2013, which is same with (Berant and Liang, 2014). The statistics are given in Table 1.

4.1.2 Benchmarks

QALD is a series of open-domain question answering campaigns, which have several different tasks. We only consider the task of English question answering over DBpedia in QALD-6 (Unger et al., 2016), which has 350 training questions and 100 test questions, with gold SPARQL queries and answer sets.

WebQuestions (Berant et al., 2013) consists of 3778 training questions and 2032 test questions, which are collected using the Google Suggest API and crowdsourcing. It only provides the pairs of question and answer set.

ComplexQuestions (Abujabal et al., 2017) is composed of 150 test questions that exhibit compositionality through multiple clauses, which is constructed using the crawl of WikiAnswers (http://wiki.answers.com) and human annotator. It has no training questions and SPARQL queries.

Table 1: Statistics of Knowledge Bases

	DBpedia	Freebase
Number of Entities	5.4 million	41 million
Number of Triples	110 million	596 million
Number of Predicates	9708	19456
Size of KBs (in GB)	8.7	56.9

Table 2: The average F1 score of WebQuestions and ComplexQuestions benchmark

	WQ	CQ
STF (Our approach)	**53.6%**	**54.3%**
STAGG (Yih et al., 2015)	52.5%	-
QUINT (Abujabal et al., 2017)	51.0%	49.2%
NFF (Hu et al., 2018)	49.6%	-
Aqqu (Bast and Haussmann, 2015)	49.4%	27.8%
Aqqu++ (Bast and Haussmann, 2015)	49.4%	46.7%

4.2 Results on WebQuestions and ComplexQuestions

Table 2 shows the results on the test set of WebQuestions(WQ) and ComplexQuestions(CQ), which are based on the Freebase. We compare our system with STAGG (Yih et al., 2015), QUINT (Abujabal et al., 2017), NFF (Hu et al., 2018) and Aqqu(Bast and Haussmann, 2015). QUINT is the model used in the original ComplexQuestions paper and Aqqu is the best publicly available QA system on WebQuestions. The average F1 of our system (53.6% for WQ and 54.3% for CQ) are better than the other systems.

Aqqu defines three query templates for WebQuestions and try to match test questions to predefined templates. It has a poor performance (27.8%) on ComplexQuestions when answering the test questions directly. Aqqu++ shows the result (46.7%) by taking manually decomposed subquestions as input and getting the intersection of subquestions' answer sets. QUINT is a system that automatically learns utterance-query templates from the pairs of question and answer set. NFF builds a relation paraphrase dictionary and leverages it to extract relations from the questions. STAGG proposes a state-transition based query graph generation method. However, its query graph structure is limited as in the Figure 1, which is very similar with the templates of Aqqu.

To generate the training data of relation extraction model on the WebQuestions, for each two nodes u and v in the training question, we explore all simple relations between u and v in Freebase. Note that if u or v is the answer node, we will anchor the answer entity first and select the relation which can handle most of answer entities as the gold relation. Each entity will be replaced in text by a special token representing its type before training. As there are no training data in ComplexQuestions, we use the models trained on WebQuestions directly.

Table 3: Evaluating QALD-6 Testing Questions

	Processed	Right	Recall	Precision	F-1
Our approach	100	70	0.72	0.89	0.80
CANaLI	100	83	0.89	0.89	**0.89**
UTQA	100	63	0.69	0.82	0.75
KWGAnswer	100	52	0.59	0.85	0.70
SemGraphQA	100	20	0.25	0.70	0.37
UIQA1	44	21	0.63	0.54	0.25
UIQA2	36	14	0.53	0.43	0.17
NFF	100	68	0.70	0.89	0.78
gAnswer	100	40	0.43	0.77	0.55
Aqqu	100	36	0.37	0.39	0.38

Generally, WebQuestions benchmark has low diversity and contains some incorrect or incomplete labeled answers. Most of questions in WebQuestions are simple questions, which can be translated to a "one-triple" query. The results on WebQuestions benchmark prove our approach is able to obtain good performance on simple questions. ComplexQuestions benchmark is more complex than WebQuestions. The experiments results show that the two template based methods QUINT and Aqqu can solve a part of the complex questions. However, it still lack of the capacity to tackle all questions due to the error of dependency parse or the mismatch of the templates. On the other hand, our system outperforms them because we do not rely on templates and have better generalization ability to solve complex questions.

4.3 Results on QALD

Table 3 shows the evaluation results on the test set of QALD-6. We compare our system with all participants of QALD-6 as well as gAnswer (Zou et al., 2014), NFF (Hu et al., 2018) and Aqqu (Bast and Haussmann, 2015). To enable the comparison, we show the experiment results in the QALD competition report format. "Processed" denotes the number of questions can be processed and "Right" refers to the number of questions that were answered correctly. Our approach can answer 70 questions correctly, and get the 0.80 F-1 score.

The experiment results show that we can beat all systems in QALD-6 campaign in F-1 except for CANaLI (Mazzeo and Zaniolo). Note that CANaLI is not an automatic QA system, as it needs users to specify the precise entities and predicates in the question sentences. gAnswer (Zou et al., 2014) proposes a relation first framework to generate the semantic query graph while it can not detect implicit relations. NFF (Hu et al., 2018) proposes a node first framework which performs well on answering complex ques-

Table 4: The performance comparison of state transition with conditions or without conditions in QALD-6 test set

	Right	Recall	Precision	F-1
with condition	70	0.72	0.89	0.80
no condition	62	0.64	0.84	0.72

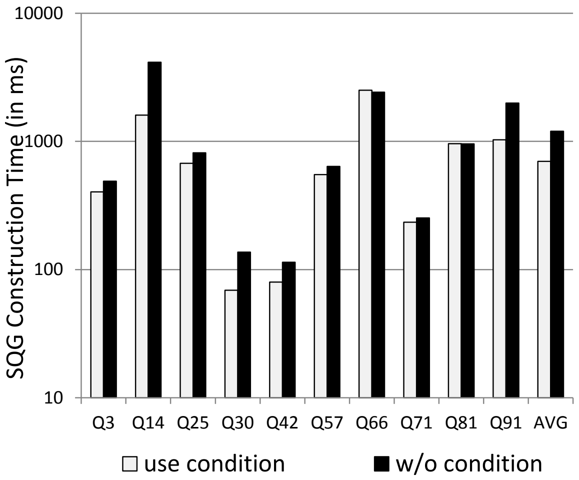

Figure 5: The elapsed time comparison of state transition with conditions or without conditions in QALD-6 test set

tions (0.78 F1 score on QALD-6). However it relies on the predefined paraphrasing dictionaries to extract relations and could not tackle with redundant nodes.

As QALD has the gold SPARQL queries, we can obtain the training data of node recognition and relation extraction model directly. Different from the WebQuestions and ComplexQuestions which have only special questions, the QALD benchmark also has general questions and imperative sentence. Besides, there are some questions that have no entities or multiple entities.

Generally, QALD benchmark has both high diversity and quality. The performance of Aqqu in QALD is worse than its performance in WebQuestions and ComplexQuestions as it has only designed the templates to WebQuestions. In contrast, our approach do not rely specific benchmark or knowledge base. The evaluation results show that our approach performs well on various types of complex questions.

4.4 Comparison experiments of the four conditions

We run experiments on QALD-6 test set to verify the effectiveness of the four conditions proposed in Section 2.2. We first compare the elapsed time of state transition with conditions or without conditions. Figure 5 shows the results of 10 questions selected randomly and AVG means the average time consumption among all 100 test questions. The average elapsed time with conditions is 700 ms while it rises to 1200 ms without the conditions. The results declare that using the conditions in the search process can avoid unnecessary searches and save SQG construction time. In some questions (such as Q66 and Q81), the gap of elapsed time are very little because the number of recognized nodes are less than three. In other words, those simple questions do not need the conditions to speed up the SQG construction process.

Table 5: Failure Analysis on QALD-6

Reason	**# (Ratio)**	**Sample question**
Structure Failure	6 (20%)	Q55: In which countries do people speak Japanese?
Entity Mapping	5 (17%)	Q88: What color expresses loyalty?
Relation Mapping	10 (33%)	Q44: What is the full name of Prince Charles?
Complex Aggregation	9 (30%)	Q80: How short is the shortest active NBA player?

On the other hand, using conditions can improve the evaluation performance (see Table 4). That because the conditions help system to reduce search space and avoid local optimum, especially the condition of connect operation. When the question has multiple nodes, if we do not use the condition of connect operation, it tries to connect each two nodes and enrolls too many mistaken states.

4.5 Error Analysis

We provide the error analysis of our approach on QALD-6 test set (100 questions), which contains gold SPARQL queries and more diversified questions. The ratio of each error type and the corresponding examples are given in Table 5.

There are mainly four reasons for the failure of some questions in our approach. The first reason is the structure failure, which means we generate a wrong SQG. That usually occurred when the correct SQG needs a fold operation. For example, the correct SQG of question "In which countries do people speak Japanese" has only two nodes "countries", "Japanese". However, we recognized the phrase "people" as a variable node by mistake and connected it to both "countries" and "Japanese" to generate a wrong SQG $Q^{S'}$. It should be eliminated by the fold operation to get the correct SQG Q^S. However, the score of $Q^{S'}$ that we get from the reward function is higher than the score of Q^S. The inherent reason is the lack of training data.

The second reason is the failure of entity linking, for example, we could not find the correct entity of the node “loyalty” in the question “What color expresses loyalty?”. It has smallest proportion as the existing entity linking algorithms already have high accuracy on QA task.

The third reason is the failure of relation extraction. For example, the correct relation between “What” and “Prince Charles” in the question ‘What is the full name of Prince Charles?” is ⟨alias⟩. However, we got a wrong relation ⟨name⟩. The relation extraction problem is very important to QA task, which can be improved by designing more elegant models and combining more useful information.

The last one is that our method cannot answer some complex aggregation questions, which need to detect the aggregation constraint first and then infer the related node and corresponding relation. Using predefined templates or utilizing textual evidence maybe works but not good enough.

5 Related Work

Generally, the solutions of KBQA can be divided into *information retrieval-based* and *semantic parsing-based* methods. The general process of *information retrieval-based* methods is to select candidate answers first and then rank them by various methods(Veyseh, 2016; Dong et al., 2015; Xu et al., 2016; Bordes et al., 2014; Yao and Durme, 2014). The main difference among *information retrieval-based* methods is how to rank the candidate answers. (Bordes et al., 2014) utilizes subgraph embedding to predict the confidence of candidate answers. (Dong et al., 2015) maximize the similarity between the distributed representation of a question and its answer candidates using Multi-Column Convolutional Neural Networks (MCCNN), while (Xu et al., 2016) aims to predicate the correct relations between topic entity and answer candidates with text evidence.

The *semantic parsing-based* methods try to translate the natural language question into semantic equivalent logical forms, such as simple λ-DCS (Berant et al., 2013; Berant and Liang, 2014), query graph (Yih et al., 2015; Zou et al., 2014), or executable queries such as SPARQL (Bast and Haussmann, 2015; Unger et al., 2012; Yahya et al., 2012). Then the logical forms are executed by corresponding technique and get the answers from knowledge base. For answering complex questions, the key is how to construct the logical form. Different from existing systems relying on the predefined templates, we build the query graph by state transition, which has stronger representation power and more robustness.

(Dong and Lapata, 2016) trains a sequence to tree model to translate natural language to logical forms directly, which needs a lot of training data. (Yu et al., 2017) uses deep bidirectional LSTMs to learn both relation-level and word-level question/relation representations and match the question to candidate relations. The pipeline of (Yu et al., 2017) is similar with (Yih et al., 2015), which detect topic entity and relation first and then recognize constraints by enumerating all connected entities of answer node or CVT node in knowledge base. The issue is they assume the question N only have one relation, which is from the topic entity to the answer. Different from (Yu et al., 2017), we consider the question can have many nodes (including entities and variables) and recognize them by a BLSTM model. We also allow multi relations between these nodes and extract them by the MCCNN model.

6 Conclusions

In this paper, we propose a state transition framework to utilize neural networks to answer complex questions, which generates semantic query graph based on four primitive operations. We train a BLSTM-CRF model to recognize nodes including entities and variables from the question sentence. To extract the relations between those nodes, we propose a MCCNN model which can tackle both explicit and implicit relations. Comparing with existing solutions, our framework do not rely on handcrafted templates and has the potential to generate all kinds of SQGs given enough training data. Extensive experiments on multi benchmark datasets confirm that our approach have the state-of-art performance.

Acknowledgments

This work was supported by The National Key Research and Development Program of China under grant 2016YFB1000603 and NSFC under grant 61622201 and 61532010. Lei Zou is the corresponding author of this work. This work is also supported by Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1191–1200.

Jun-Wei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 967–976.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1431–1440.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1415–1425.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia - a crystallization point for the web of data. *J. Web Sem.*, 7(3):154–165.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 615–620.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 260–269.

Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2018. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Trans. Knowl. Data Eng.*, 30(5):824–837.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 217–226.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*.

Giuseppe M. Mazzeo and Carlo Zaniolo. Answering controlled natural language questions on RDF knowledge bases. In *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016.*, pages 608–611.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *WWW*, pages 639–648.

Christina Unger, Axel-Cyrille Ngonga Ngomo, and Elena Cabrio. 2016. 6th open challenge on question answering over linked data (QALD-6). In *Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*, pages 171–177.

Amir Pouran Ben Veyseh. 2016. Cross-lingual question answering using common semantic space. In *Proceedings of TextGraphs@NAACL-HLT 2016: the 10th Workshop on Graph-based Methods for Natural Language Processing, June 17, 2016, San Diego, California, USA*, pages 15–19.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *EMNLP-CoNLL*, pages 379–390.

Yi Yang and Ming-Wei Chang. 2015. S-MART: novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 504–513.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 956–966.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1321–1331.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 571–581.

Xinbo Zhang and Lei Zou. 2017. Improving the precision of RDF question/answering systems: A why not approach. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017*, pages 877–878.

Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over RDF: a graph data driven approach. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 313–324.

A Multi-answer Multi-task Framework for Real-world Machine Reading Comprehension

Jiahua Liu[1,2], **Wan Wei**[2], **Maosong Sun**[1], **Hao Chen**[2], **Yantao Du**[2], **Dekang Lin**[2*]
[1]State Key Laboratory of Intelligent Technology and Systems,
Tsinghua National Laboratory for Information Science and Technology,
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Naturali Ltd, Beijing, China

Abstract

The task of machine reading comprehension (MRC) has evolved from answering simple questions from well-edited text to answering real questions from users out of web data. In the real-world setting, full-body text from multiple relevant documents in the top search results are provided as context for questions from user queries, including not only questions with a single, short, and factual answer, but also questions about reasons, procedures, and opinions. In this case, multiple answers could be equally valid for a single question and each answer may occur multiple times in the context, which should be taken into consideration when we build MRC system. We propose a multi-answer multi-task framework, in which different loss functions are used for multiple reference answers. Minimum Risk Training is applied to solve the multi-occurrence problem of a single answer. Combined with a simple heuristic passage extraction strategy for overlong documents, our model increases the ROUGE-L score on the DuReader dataset from 44.18, the previous state-of-the-art, to 51.09.

1 Introduction

Machine reading comprehension (MRC) or question answering (QA) has been a long-standing goal in Natural Language Processing. There is a surge of interest in this area due to new end-to-end modeling techniques and the release of several large-scale, open-domain datasets.

In earlier datasets (Hermann et al., 2015; Hill et al., 2016; Yang et al., 2015; Rajpurkar et al., 2016), the questions did not arise from actual end users. Instead, they were constructed in cloze style or created by crowdworkers given a short passage from well-edited sources such as Wikipedia and CNN/Daily Mail. As a consequence, the questions are usually well-formed and about simple facts, and the answers are guaranteed to exist as short spans in the given candidate passages.

In MS-MARCO (Nguyen et al., 2016), the questions were sampled from actual search queries, which may have typos and may not be phrased as questions.[1] Multiple short passages, which might have the answer to the query, were extracted from webpages by a separate information retrieval system.

He et al. (2017) made the DuReader dataset a more realistic reflection of the real-world problem by including not only questions with relatively short and factual answers, but also questions about complex descriptions, procedures, opinions, etc. which may have multiple, much longer answers, or no answer at all. Furthermore, full-body text from webpages listed in top search results are directly provided as context. These documents tend to be much noiser than Wikipedia and CNN. They are much longer (5 times longer than those in MS-MARCO on average) and contain many paragraphs that are irrelevant to the query.

New problems arise as we now consider the task of machine reading comprehension in a much more challenging real-world setting. First, multiple valid answers to a single question are not only possible but quite common. Figure 1 shows some examples of questions with multiple answers from the DuReader dataset. There could be multiple ways to perform the same task (Question 1), multiple opinions about the same subject (Question 2), or multiple explanations for the same observation (Question 3). However, few works have been done with multiple answers in machine reading comprehension. To address this problem, we propose a multi-answer multi-task scheme which incorporates multiple reference answers in the objective

---

[*] Corresponding author: D. Lin (lindek@naturali.io).

[1] We use these two terms, question and query, interchangeably in the following content

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2109–2118
Brussels, Belgium, October 31 - November 4, 2018.

Question 1:
word字间距
character spacing in Word
Answers:
1. 选择要缩进的内容，点击右键选择字体，打开“字体”对话框，选择“字符间距”选项卡，勾选“为字体调整字间距”并输入数字，最后单击“确定”按钮。
2. 选中你要设置字符间距的文字后，单击鼠标右键，选择字体选项，在切换到字符间距。
3. 点击上部菜单栏上“格式”→“字体”→“字符间距”，调节间距的磅值即可。
1. Select the text you want to indent, and right-click on it to select Font, open the Font dialog box, and then select the Character Spacing tab, select the Kerning for fonts check box, and enter a number, at last click OK.
2. Select the text you want to set character spacing, and right-click on it to select Font, and then switch to Character Spacing.
3. click Format from the upper menu bar, select Font, select Character Spacing, and then change the point size.

Question 2:	
龙珠传奇好看吗	Is LONGZHU worth watching?
Answers:	
1. 好看的、节奏还可以、萌萌的甜甜的。	1. It is, the pace is good and it's adorable and sweet.
2. 不好看，个人不喜欢那种类型的电视剧。	2. It’s not, personally I don’t like that kind of TV series.
3. 还行，不纠结历史问题的话可追。	3. It’s ok, it’s fine to follow if you don’t care about the historical accuracy.

Question 3:	
响一声就说正在通话中	Got “User Busy” message after one ring
Answers:	
1. 别人对你设置了黑名单。	1. Your number is in that person’s blacklist.
2. 别人挂了你的电话。	2. The person you called hung up on you.

Figure 1: Examples of questions with multiple answers from the DuReader dataset

function (but still predicts a single answer in decoding time). We propose three different kinds of multi-answer loss functions and compare their performance through experiment.

Another problem is the multiple occurrences of the same answer. As rich context is provided for a single question, the same answer could occur more than one time in different passages, or even at different places of the same passage. In this case, using only one gold span for the answer could be problematic, as the model is forced to choose one span over others that contain the same content. To solve this problem, we propose to apply Minimum Risk Training (MRT), which uses the expected metric as the loss and gives reward to all spans that are similar with the gold answer.

In this paper, we present a multi-answer, multi-task objective function to train an end-to-end MRC/QA system. We experiment with various alternatives on the DuReader dataset and show that our model out-performs other competing systems and increases the state-of-the-art ROUGE-L score by about 7 points.

2 Related Work

Various datasets have been released in recent years, which fuel the research for reading comprehension and question answering. The CNN/Daily-Mail dataset (Hermann et al., 2015) and the Children’s Book Test (Hill et al., 2016) evaluate comprehension by filling in missing words from well-edited texts. SQuAD (Rajpurkar et al., 2016) is one of the most popular datasets for reading comprehension, where a span in a Wikipedia passage is to be extracted to answer questions generated by annotators. The WikiQA (Yang et al., 2015) is another dataset from Wikipedia, where one single sentence is to be selected to answer questions from search engine logs. Different from the above datasets, the MS-MARCO dataset (Nguyen et al., 2016) was built in a real-world setting. The questions were real anonymized Bing queries and multiple passages are extracted from related web pages by a separate system. The DuReader (He et al., 2017) is a Chinese dataset, similarly constructed from user queries as MS MACRCO, but in a more realistic setting using Baidu Web Search and Baidu Answers (Zhidao) data. While a small proportion of questions were labeled with multiple answers in MS MARCO (9.93%), more than half of the DuReader queries were annotated with multiple answers, which provides the perfect setup for our work.

Great effort has been put into the development of sophisticated neural models for machine reading comprehension. The attention mechanism was first introduced by Hermann et al. (2015) into reading comprehension and soon became the dominating model. Wang and Jiang (2017) proposed to solve machine comprehension using Match-LSTM and answer pointer. Seo et al. (2017) and Xiong et al. (2017) applied different ways to match the question and the context with bi-directional attention. Hu et al. (2017) used iterative aligner to match the question and the passage with feature-rich encoder. Cui et al. (2017) employed one more layer of attention over the bi-directional attention mechanism. Wang et al. (2017) applied a self-matching mechanism to aggregate evidence from the context. Tan et al. (2018) proposed to generate answer from extracted answer span. Yu et al. (2018) proposed to use convolution with self-attention instead of recurrent models in reading comprehension.

Recently there are some emerging works starting to touch the reading comprehension task from the answer side. Wang et al. (2018a) proposed to use evidence aggregation to re-rank answer candidates extracted from different passages, and Wang et al. (2018b) proposed Cross-Passage Answer Verification model for the same purpose. Neither of them involved multiple answers as in this work.

Minimum Risk Training (MRT) has been widely used in various tasks in NLP. Shen et al. (2016) introduced MRT into Neural Machine Translation, and Ayana et al. (2016) applied it in Text Summarization.

3 Our Approach

In this section we describe in details the architecture of our model which is depicted in Figure 2.

3.1 Passage Extraction

Unlike most other datasets where the source of answers is a short passage with a few hundred words, the DuReader dataset provides up to 5 full documents, which may contain up to 100K words. This incurs exorbitant demand on memory and training time. To deal with this issue, previous approaches select a single representative paragraph for each document, on which the answer extraction is performed. The original paper of DuReader (He et al., 2017) employed a simple heuristic strategy, and Wang et al. (2018b) trained a paragraph ranking model, while Clark and Gardner (2017) applied TF-IDF based method for the TriviaQA dataset (Joshi et al., 2017) which was in a similar situation. However, answers could come from more than one paragraph. We apply a simple yet effective method to extract contents from multiple paragraphs of the document, aiming to include as much information for the answer extraction as possible.

We concatenate the title and the whole document as the passage if it is shorter than a predefined maximum length. If not, we employ passage extraction in the following way:

- The title of the document is extracted. Whether a document is relevant to the question could be easily seen from the title.
- We compute BLEU-4 score of each paragraph relative to the question, and select the one that appears first in the document among paragraphs with top-k scores.
- We extract the full body of this selected paragraph and the next paragraph.
- For each of the following paragraphs, the first sentence is extracted as it probably contains the main point.
- We concatenate all the extracted contents to form the extracted passage, and it is truncated to the maximum length if it is longer than the predefined value.

We apply our model on the basis of the extracted passages.

3.2 Representation of Word

Given a word sequence of question $Q = \{w_t^q\}_{t=1}^m$, and a word sequence of extracted passage $P = \{w_t^p\}_{t=1}^n$, we combine different useful information to form the representation of each question word w_t^q and passage word w_t^p:

- Word-level embedding: each word w in the question and passage is mapped to its corresponding n-dimensional embedding we.
- POS tag embedding: we use a POS tagger to tag each word in the question and passage. Each POS tag is mapped to a m-dimensional embedding pe.
- Word-in-question feature: following Chen et al. (2017) and Weissenborn et al. (2017), we use one additional binary feature wiq for each passage word, indicating whether this word occurs in the question.

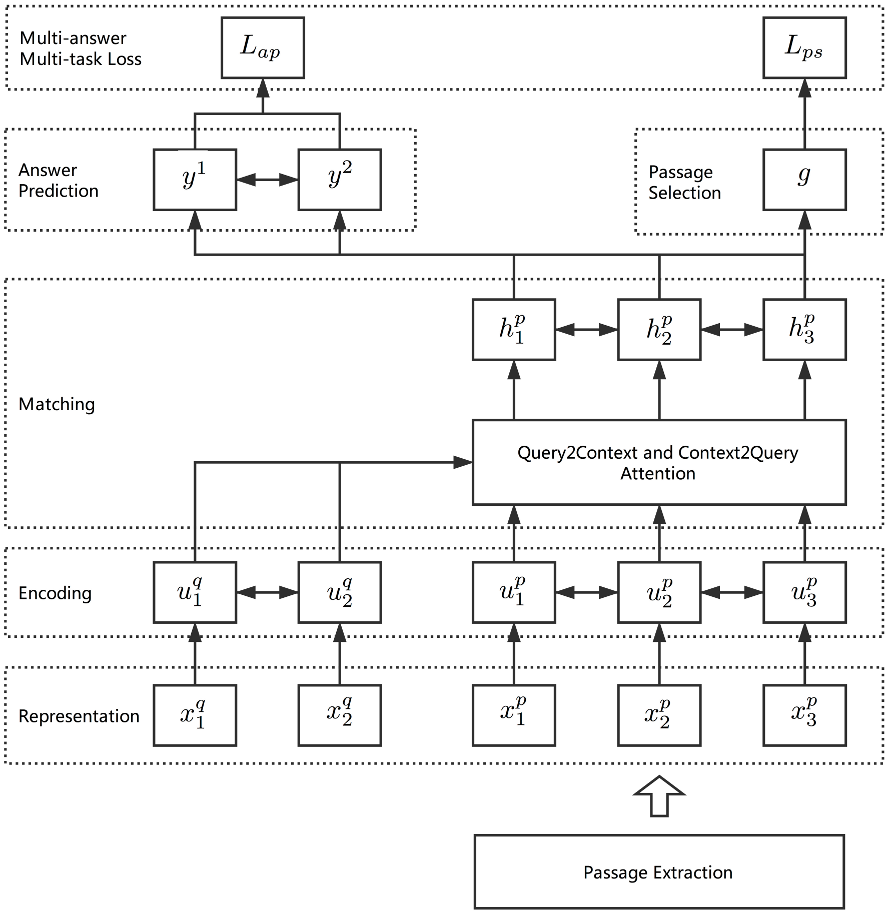

Figure 2: Model Architecture

Each question word is represented as the concatenation of the word embedding we, and the POS tag embedding pe, denoted as $x^q = [we; pe]$. Each passage word is additionally concatenated with the word-in-question feature wiq: $x^p = [we; pe; wiq]$.

It should be noted that, character-level embedding is an important part of word representation in English MRC models (Seo et al., 2017; Weissenborn et al., 2017; Wang et al., 2017; Tan et al., 2018). Character sequence would give information which helps to relieve the OOV problem, as many English words share the same stem and differ only in prefix or suffix. However, this is not the case in Chinese, and we observe no significant improvement incorporating character-level embedding into our system.

3.3 Encoding Layer

Following previous work, we use a bi-directional LSTM to obtain contextual encoding for each word in the question and passage respectively:

$$u_i^q = BiLSTM(u_{i-1}^q, x_i^q) \tag{1}$$

$$u_j^p = BiLSTM(u_{j-1}^p, x_j^p) \tag{2}$$

3.4 Match Layer

To fuse question encoding and passage encoding, we adopt the Attention Flow Layer (Seo et al., 2017) with a simpler similarity function. The similarity score between the contextual encoding for a query word u_i^q and that for a passage word u_j^p is defined as:

$$s_{ij} = u_i^{qT} \cdot u_j^p \tag{3}$$

The context-to-query attention vectors c_j^p are computed from the similarity scores:

$$a_{ij} = \frac{exp(s_{ij})}{\sum_{k=1}^{m} exp(s_{kj})} \tag{4}$$

$$c_j^p = \sum_{i=1}^{m} a_{ij} u_i^q \tag{5}$$

The query-to-context attention vector d^p is computed as:

$$z_j = \max_i s_{ij} \tag{6}$$

$$b_j = \frac{exp(z_j)}{\sum_{k=1}^{n} exp(z_k)} \tag{7}$$

$$d^p = \sum_{j=1}^{n} b_j u_j^p \tag{8}$$

Another BiLSTM is applied on top of them to get the question-aware passage representation:

$$h_j^p = BiLSTM(h_{j-1}^p, [u_j^p; c_j^p; u_j^p \circ c_j^p; u_j^p \circ d^p]) \tag{9}$$

3.5 Multi-answer multi-task loss function

3.5.1 Answer prediction with multi-answer

A reading comprehension model is typically trained as an extractor of an answer span from a candidate passage. In DuReader dataset, multiple reference answers are provided for a single question. For each of the reference answers, we add the span with the highest F1 score to the gold answer spans. For models considering only a single answer span (baseline model), the gold answer span is the one with the highest F1 score relative to any of the reference answers (He et al., 2017; Wang et al., 2018b).

In the boundary model with pointer network (Wang and Jiang, 2017; Wang et al., 2017; Tan et al., 2018), two probability distributions y_j^1 and y_j^2 $(j = 1 \ldots n)$, which denote the probability that position j is the beginning or the end of the answer span respectively, are computed as follows:

$$s_j^t = v^T tanh(W_h^p h_j^p + W_a^P h_{j-1}^a) \tag{10}$$

$$y_j^t = \frac{exp(s_j^t)}{\sum_{k=1}^{n} exp(s_k^t)} \tag{11}$$

$$c_t = \sum_{j=1}^{n} y_j^t h_j^p \tag{12}$$

$$h_t^a = BiLSTM(h_{t-1}^a, c_t) \tag{13}$$

where $t = 1, 2$, and the initial hidden state h_0^a is generated by an attention-pooling over the question representation following Wang et al. (2017):

$$s_i = v^T tanh(W_u^q u_i^q + b) \tag{14}$$

$$a_i = \frac{exp(s_i)}{\sum_{k=1}^{m} exp(s_k)} \tag{15}$$

$$h_0^a = \sum_{i=1}^{m} a_i u_i^q \tag{16}$$

Note that all passages for the same question are concatenated in order to predict one answer span. The loss is defined as the sum of negative log probabilities of the ground truth start and end position based on the predicted distributions:

$$L = -(\log y_{start}^1 + \log y_{end}^2) \tag{17}$$

We propose three different ways to incorporate multiple answers. A simple solution is to compute the average loss for multiple answer spans:

$$L_{avg} = -\frac{1}{A} \sum_{k=1}^{A} (\log y_{start_k}^1 + \log y_{end_k}^2) \tag{18}$$

L_{avg} treats all answer spans as equally good. However, some of them may be closer to human-generated answers than others. We therefore define the weighted average loss as follow:

$$L_{wavg} = -\sum_{k=1}^{A} w_k (\log y_{start_k}^1 + \log y_{end_k}^2) \tag{19}$$

where w_k is the F-score between the answer span and the corresponding human-generated answer, normalized by the sum of the scores of each answer.

Another solution is to use the minimal value of the loss from each span:

$$L_{min} = \min_k (-(\log y_{start_k}^1 + \log y_{end_k}^2)) \tag{20}$$

Instead of predicting all answer spans, this loss will encourage the model to predict only the easiest answer span for it.

The answer span prediction loss L_{ap} is defined as the average of any of the loss functions described above over the training set.

3.5.2 Passage selection with multi-answer

Tan et al. (2018) showed that their single-answer, multi-passage MRC model benefits from using multi-task learning by adding an auxiliary loss to predict the correct passage to extract the answer from. We adapt the idea to compute passage selection loss L_{ps} in multi-answer setting.

We first apply attention-pooling over the passage representation $\{h_j^p\}_{j=1}^n$, and then calculate a matching score g for each passage:

$$s_j = v^T tanh(W_u^p h_j^p + b) \tag{21}$$

$$a_j = \frac{exp(s_j)}{\sum_{k=1}^n exp(s_k)} \tag{22}$$

$$r^p = \sum_{j=1}^n a_j h_j^p \tag{23}$$

$$g = sigmoid(v_{sp}^T r^p) \tag{24}$$

Since multiple answers are provided in the DuReader dataset, multiple passages may contain correct answers. The match score g for different passages are not in competition against one another. We therefore used pointwise sigmoid function instead of the softmax function (as in Tan et al. (2018)) in the passage selection loss L_{ps}:

$$\begin{aligned} L_{ps} = -\frac{1}{K}\sum_{k=1}^K (y_k \log g_k \\ +(1-y_k)\log(1-g_k)) \end{aligned} \tag{25}$$

where $y_k = 1$ if one of the gold span comes from this passage, $y_k = 0$ otherwise.

3.5.3 Joint training

We train our model by jointly optimizing answer span prediction loss and passage selection loss:

$$L = L_{ap} + \lambda_{ps} L_{ps} \tag{26}$$

where λ_{ps} is a hyper-parameter tuned on the dev set.

3.6 Minimum Risk Training

Minimum Risk Training (MRT) has been widely used in various tasks in NLP. The basic idea is to directly optimize the evaluation metric instead of maximizing the log likelihood of training data using Maximum Likelihood Estimation (MLE) as described above. In MRT, the object is to minimize the expected loss with respect to the posterior distribution:

$$J_{MRT}(\theta) = \frac{1}{N}\sum_{i=1}^N \mathbb{E}_{y_i|x_i;\theta}[\Delta(y_i, y_i^*)] \tag{27}$$

where $\Delta(y_i, y_i^*)$ is a function which indicates the difference between the predicted result y_i and the label result y_i^*.

In this work, we apply MRT to solve the problem of multi-occurrence of answer in machine reading comprehension, directly using the metric (ROUGE-L) as Δ. As an answer occurs multiple times in the context, each span in which the answer occurs will have minimum difference with the answer, and is thus given a high score by a model trained with MRT.

In machine translation and many other tasks, to compute the expected metric with respect to the posterior distribution is often intractable. Thus sampling methods are commonly used in MRT training. However, in our span extraction model, we use all spans without sampling.

Formally, the MRT loss in our model is defined as:

$$J_{MRT}(\theta) = \frac{1}{N}\sum_{i=1}^N \sum_{k=1}^{|P|} \sum_{l=k+1}^{|P|} y_k^1 y_l^2 \Delta(P_{k,l}, A) \tag{28}$$

As in Hu et al. (2017), we minimize the linear combination of MLE and MRT loss:

$$J(\theta) = J_{MLE}(\theta) + \lambda J_{MRT}(\theta) \tag{29}$$

where $J_{MLE}(\theta)$ refers to L in equation 26 and λ is a hyper-parameter tuned on the development set.

4 Experiment

We conduct our experiment on the DuReader dataset (He et al., 2017), where multiple passages containing full-body text are provided for each question, and over half of the questions have multiple answers.

4.1 Dataset and Evaluation Metrics

The DuReader dataset consists of 201574 questions in total, with 181574 in the training set, 10000 in the development set, and 10000 in the test set. The questions are sampled from frequently occurring queries from Baidu search engines, and the full-body text of top-5 search results from the web are provided as the context.

BLEU-4 and ROUGE-L are used in evaluation on DuReader. However, the implementations for the two metrics are quite different in the official evaluation tool. As in MS-MARCO, the BLEU-4 score is normalized across all questions, essentially giving different weights to different questions, while the ROUGE-L is averaged across different questions. We mainly focus on ROUGE-L as each question in a reading comprehension

Model	**ROUGE-L**	**BLEU-4**
BiDAF baseline	37.68	35.51
+ passage extraction	44.57	38.03
+ rich feature	**48.93**	**42.17**

Table 1: The influence of passage extraction and rich-feature representation on the development set

Loss	**ROUGE-L**	Δ
single answer	48.93	-
L_{min}	49.05	+ 0.12
L_{avg}	49.67	+ 0.74
L_{wavg}	**49.77**	**+ 0.84**

Table 2: Comparison among different choices for the loss function with multiple answers on the development set

dataset should have equal weight in evaluation (Tan et al., 2018). For a single question with multiple reference answers, the maximum score with any reference answers is used, as implemented in the official tool for ROUGE-L. This is reasonable as providing one valid answer is good enough in many cases.

4.2 Implementation Details

4.2.1 Word and POS Tag Embedding

We train a segmentation model with one-layer BiLSTM using the DuReader dataset, and apply it to a subset of SogouT corpus[2], which contains a large amount of Chinese web pages(Liu et al., 2012). 256-dimension word embeddings are trained on this data with language model task using one-layer BiLSTM model.

As for POS tag, we use a POS tagger trained on the Chinese Treebank (CTB) data to tag each word in questions and passages in the DuReader dataset. 64-dimension POS tag embeddings are trained on this data using one-layer BiLSTM model.

We keep all word and POS tag embeddings fixed during training.

4.2.2 Training and Parameters

The maximum length of each passage is set to be 500. The batch size is set to be 32. The dimension of hidden vector is set to be 150 for all layers. Dropout (Srivastava et al., 2014) is applied between layers, with a dropout rate of 0.15. We use $\lambda_{ps} = 3$ for passage selection loss and $\lambda = 10$ for MRT. All parameters are tuned on the DuReader development set.

As MRT training is more time-consuming than MLE training, our MRT model is initialized with model trained with MLE. It usually obtains the best result in just one epoch, which results in feasible training time.

Our model is optimized with Adam algorithm (Kingma and Ba, 2014), and the learning rate is fixed to 0.001 during training.

Model	**ROUGE-L**	Δ
single-answer baseline	48.93	-
+ multi-answer loss	49.77	+ 0.84
+ passage selection loss	49.96	+ 1.03
+ MRT	**50.62**	**+ 1.69**

Table 3: Results with multi-answer multi-task loss and Minimum Risk Training on the development set

4.3 Results

4.3.1 Single-Answer Baseline

Table 1 shows the results for passage extraction and rich-feature representation (pre-trained word, POS, and word-in-query embeddings) on the development set. Both of them dramatically increase ROUGE-L and BLEU-4 score over the BiDAF baseline from the original DuReader paper. Together they form our single-answer baseline, on which we test the effectiveness of the multi-answer multi-task loss and Minimum Risk Training.

4.3.2 Different loss functions with multi-answer

Table 2 shows the experimental results with three different multi-answer loss functions introduced in Section 3.5.1. All of them offer improvement over the single-answer baseline, which shows the effectiveness of utilizing multiple answers. The average loss performs better than the min loss, which suggests that forcing the model to predict all possible answers is better than asking it to just find the easiest one. Not surprisingly, by taking into account the quality of different answer spans, the weighted average loss outperforms the average loss and achieves the best result among the three. All later experiments are conducted based on the weighted average loss.

---

[2] `http://www.sogou.com/labs/resource/t.php`

Model	ROUGE-L	BLEU-4
BiDAF (He et al., 2017)	39.0	31.8
Match-LSTM (He et al., 2017)	39.2	31.9
PR+BiDAF (Wang et al., 2018b)	41.81	37.55
PE+BiDAF (ours)	45.93	38.86
V-Net (Wang et al., 2018b)	44.18	40.97
Our complete model	**51.09**	**43.76**
Human	57.4	56.1

Table 4: Performance of our model and competing models on the DuReader test set

Model	ROUGE-L	
	Q_s	Q_m
single-answer	38.01	53.8
multi-answer	**38.66**	**54.65**

Table 5: Results on Q_s and Q_m

4.3.3 Multi-task Loss and Minimum Risk Training

As we can see in Table 3, the ROUGE-L score on the DuReader development set increases to 49.77 by incorporating multi-answer into the loss function. Joint learning with passage selection loss yields an increase of 0.19. And with Minimum Risk Training, our model can reach a ROUGE-L score of 50.62, with a further increment of 0.66.

4.3.4 Comparison with State-of-the-art

Table 4 shows the performance of our model and other state-of-the-art models on the DuReader test set. First, we compare our passage extraction method with the paragraph ranking model from Wang et al. (2018b). Based on the same BiDAF model described in Section 3.4, our method (PE+BiDAF) significantly outperforms the trained model from Wang et al. (2018b) (PR+BiDAF) on the DuReader test set. As we can see, our complete model achieves the state-of-the-art performance in both ROUGE-L and BLEU-4, and greatly narrows the performance gap between MRC system and human in the challenging real-world setting.

4.3.5 Further Analysis

For further analysis, we construct two sets from the development set. Q_s contains 2787 questions with a single reference answer, and Q_m contains 6650 questions with more than one reference answer. 563 questions from the development set are labeled with no answer, and thus not included in Q_s or Q_m. Table 5 shows the performance of our model on Q_s and Q_m. It can be seen that even questions with single answer (Q_s) can benefit from using multiple answers in training. The improvement for Q_m is higher than that for Q_s.

5 Conclusion

In this paper, we focus on real-world machine reading comprehension. We propose a multi-answer multi-task framework to tackle the multi-answer problem which is common in everyday world. Minimum Risk Training is applied to solve the multi-occurrence problem of the answer. We also propose a simple method for passage extraction which solves the length issue of the passage. Experimental results indicate that our model achieves state-of-the-art performance in the challenging DuReader dataset.

Despite using multiple answers in training, our system only predicts a single answer in decoding time. However, in some cases (e.g. for questions about opinion), finding all possible answers may be desirable. In the future, we plan to design models which could generate all possible answers for a single question.

References

Ayana, Shiqi Shen, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with minimum risk training. *Computing Research Repository*, arXiv:1604.01904. Version 2.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension.

Computing Research Repository, arXiv:1710.10723. Version 2.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 593–602. Association for Computational Linguistics.

Wei He, Kai Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, et al. 2017. Dureader: a chinese machine reading comprehension dataset from real-world applications. *Computing Research Repository*, arXiv:1711.05073.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of International Conference on Learning Representations*.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *Computing Research Repository*, arXiv:1705.02798. Version 5.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Computing Research Repository*, arXiv:1705.03551.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computing Research Repository*, arXiv:1412.6980.

Yiqun Liu, Fei Chen, Weize Kong, Huijia Yu, Min Zhang, Shaoping Ma, and Liyun Ru. 2012. Identifying web spam with the wisdom of the crowds. *ACM Transactions on the Web (TWEB)*, 6(1):2.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *Computing Research Repository*, arXiv:1611.09268.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of International Conference on Learning Representations*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Chuanqi Tan, Furu Wei, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2018. S-net: From answer extraction to answer synthesis for machine reading comprehension.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *Proceedings of International Conference on Learning Representations*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018a. Evidence aggregation for answer re-ranking in open-domain question answering. In *Proceedings of International Conference on Learning Representations*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198. Association for Computational Linguistics.

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018b. Multi-passage machine reading comprehension with cross-passage answer verification. *Computing Research Repository*, arXiv:1805.02220.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280. Association for Computational Linguistics.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *Proceedings of International Conference on Learning Representations*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018. Association for Computational Linguistics.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *Proceedings of International Conference on Learning Representations*.

Logician and Orator: Learning from the Duality between Language and Knowledge in Open Domain

Mingming Sun[1,2], **Xu Li**[1,2], **Ping Li**[1]
[1]Big Data Lab (BDL-US), Baidu Research
[2]National Engineering Laboratory of Deep Learning Technology and Application, China
{sunmingming01,lixu13,liping11}@baidu.com

Abstract

We propose the task of Open-Domain Information Narration (OIN) as the reverse task of Open Information Extraction (OIE), to implement the dual structure between language and knowledge in the open domain. We then develop an agent, called Orator, to accomplish the OIN task, and assemble the Orator and the recently proposed OIE agent – Logician (Sun et al., 2018) into a dual system to utilize the duality structure with a reinforcement learning paradigm. Experimental results reveal the dual structure between OIE and OIN tasks helps to build better both OIE agents and OIN agents.

1 Introduction

The duality between language and knowledge is natural for human intelligence. The human can extract knowledge from natural language to learn or remember, and then narrate the knowledge back to natural language to communicate. Information extraction (IE) is a task to simulate the first part of the duality, which is a long-term hot spot for NLP research. Recently, the task that fulfills the last part of the duality, that is, assembling a set of relation instances/facts or database records into natural language sentences/documents, has also attracted many interests (Wiseman et al., 2017; Chisholm et al., 2017; Agarwal and Dymetman, 2017; Vougiouklis et al., 2017; Yin et al., 2016). In the literature, this task has been referred to as "data to document generation" (Wiseman et al., 2017) or "knowledge-to-text" (Chisholm et al., 2017). In this paper, we name the task as *information narration (IN)*, to emphasize the reverse relationship to the information extraction (IE) task.

The duality between language and knowledge (and thus between the IE and IN tasks) can be examined in closed-domain or open-domain. For the closed-domain problem, the closed-domain IE (CIE) task is often referred to as "relation extraction" or "relation classification", which identifies instances of a fixed and finite set of relations from natural language corpus, using supervised methods (Kambhatla, 2004; Zelenko et al., 2003; Miwa and Bansal, 2016; Zheng et al., 2017) or weakly supervised methods (Mintz et al., 2009; Lin et al., 2016). In the meantime, the close-domain IN (CIN) task (Wiseman et al., 2017; Chisholm et al., 2017; Agarwal and Dymetman, 2017; Vougiouklis et al., 2017; Yin et al., 2016) transforms a set of facts with a pre-defined schema or relation types (such as facts from Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007), or database tables), into natural language sentences/documents. Furthermore, the dual structure between CIE and CIN tasks has been noticed and utilized in (Chisholm et al., 2017).

For the open-domain problem, the open-domain IE (OIE) task is to investigate how the natural language sentences express the facts, and then use the learned knowledge to extract entity and relation level intermediate structures from open-domain sentences (Christensen et al., 2011; Etzioni et al., 2011; Schmitz et al., 2012; Pal and Mausam, 2016). Although the OIE task has attracted much interests and obtained many applications (Christensen et al., 2013, 2014; Mausam, 2016; Stanovsky et al., 2015; Khot et al., 2017; Fader et al., 2014), the OIN task has not been stated, neither the duality between the language and knowledge in the open domain.

	Open-Domain	Closed-Domain
Extraction	OIE	CIE
Narration	OIN	CIN

Table 1: Taxonomy: Tasks between knowledge and natural language.

The tasks involved in the duality between language and knowledge is shown in Table 1, where

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2119–2130
Brussels, Belgium, October 31 - November 4, 2018.

the OIN task has not been stated. In this paper, we focus on the OIN task and the duality between the OIE and OIN tasks, for following reasons: 1) the OIN task is an essential component for open-domain information processing pipeline. For example, it is helpful for building natural and informative response for open-domain KBQA systems (Khot et al., 2017; Fader et al., 2014). 2) (as the results in this paper will illustrate) the duality between tasks can be valuable for building better agents for both tasks (Xia et al., 2017, 2016).

A major historical obstacle for investigating the duality between OIE and OIN tasks is the absence of parallel corpus between natural language sentence and open-domain facts. Recently, the SAOKE dataset (Sun et al., 2018) was released, which contains more than forty thousand of human-labeled open-domain sentence-facts pairs, and thus essentially eliminates the obstacle for our investigation.

The contribution of this paper lies in following aspects:

- We propose the concept of OIN task, which is potentially an important component for open-domain information pipeline. We develop the Orator agent to fulfill the task;
- We build a multi-agent system with Logician and Orator to exploit the dual structure between language and knowledge in open domain. Experimental results reveal that the dual information is beneficial for improving the performance of both agents.

The paper is organized as follows: Section 2 discusses the related work. Section 3 explains the Orator agent for OIN task. Section 4 describes the multi-agent system with Logician and Oration and its algorithm to learn from the duality between language and sentence. The experimental results of the fine tuned agents are shown and discussed in Section 5. We conclude our work and discuss the future direction in Section 6.

2 Related Work

2.1 Information Narration

The closed-domain information narration (CIN) task has been studied in (Wiseman et al., 2017; Chisholm et al., 2017; Agarwal and Dymetman, 2017; Vougiouklis et al., 2017). These CIN agents face problems from different problem domains, from people biographies to basketball game records, but most of them follow the same sequence-to-sequence pattern. First, the algorithm encodes a sequence of facts into a set of annotations and then decodes the annotations into a natural language text. Mechanisms such as attention (Bahdanau et al., 2014) and copying (Gu et al., 2016) are employed into the decoder to improve the performance. Then, the models are trained on a supervised dataset with back-propagation.

In this work, we adopt a similar sequence-to-sequence architecture to build our baseline Orator agent, but with following differences: 1) the Orator is proposed to narrate open domain facts, where the encoder must encode words rather than the entities and relations in the closed domain; 2) the baseline Orator will be fine tuned using the dual learning algorithm proposed in this paper.

2.2 Dual Learning Systems

For many natural language processing tasks, there exist corresponding reverse/dual tasks. One example of a pair of dual problems is the question answering (QA) and question generation (QG). In (Tang et al., 2017), the duality between QA problem and QG problem was considered as a constraint that both problems must share the same joint probability. Then, a loss function that implemented the constraint was involved in the supervised learning procedures for both agents. Furthermore, researchers (Tang et al., 2017; Duan et al., 2017; Sachan and Xing, 2018) use both the question-answering agent and the question-generation agent to identify extra high-confident question-answering pairs, which are further used to fine tune the pre-trained agents.

Back-and-forth translation (or round-trip translation) [1] is another example of duality, in the field of machine translation. It has been employed to evaluate the quality of machine translation systems (Van Zaanen and Zwarts, 2006), or to test the suitability of text for machine translation (Gaspari, 2006; Shigenobu, 2007). Recently, (Xia et al., 2016) implemented the duality in a neural-based dual learning system, in which the quality of each translation agent was improved on the unlabeled dataset using the rewards provided by its

---

[1] `https://en.wikipedia.org/wiki/Round-trip_translation`

	Chinese	English Translation
Sentence	他发明了有两个固定点（水的沸点和冰点）的100摄氏度计温尺，是世界大多数地区通用的摄氏温度计的前身。	He invented a 100 degree centimeter temperature scale with two fixed points (the boiling point of water and the freezing point), which is the precursor of the Celsius thermometer in most parts of the world.
Facts	(他,发明,100摄氏度计温尺)(100摄氏度计温尺,有,两个固定点)(水的[沸点|冰点],ISA,两个固定点)(100摄氏度计温尺,是X的前身,世界大多数地区通用的摄氏温度计)	(He, invented, the 100 degree Celsius temperature scale) (100 Celsius temperature scale, has, two fixed points) (water [boiling point | freezing point], ISA, two fixed points) (100 degrees Celsius temperature scale, is the predecessor of X, most Celsius thermometers in most parts of the world)

Table 2: An example sentence and the corresponding facts in the SAOKE dataset, where "ISA" is a symbol denoting the "is-a" relationship in the SAOKE format.

corresponding dual agent, using the reinforcement learning technique.

Parsing-reconstruction is also a pattern of duality. (Konstas et al., 2017) considered the AMR (Abstract Meaning Representation)(Banarescu et al., 2013) parsing problem (text to AMR) and AMR generation problem (AMR to text) in one system, in which the AMR parser generated extra text-AMR pair data to fine tune the AMR generator. The AMR generator, however, does not contribute to the performance improvement of the AMR parser. The CIE agent and CIN agent in (Chisholm et al., 2017) also follow this pattern, where both agents help each other to improve by sharing weights. Nevertheless, the sharing weight strategy cannot be applied to agents with different architecture, which is a typical situation in practice.

From these practices, it can be seen that the duality can be implemented with two major approaches: 1) by providing additional labeled samples via bootstrapping, and 2) by adding losses or rewards to the training procedure of the agents. In this paper, we follow the second approach. We design a set of rewards, among which some are related to OIE and OIN tasks respectively, and some are related to the duality of the problems. Then we optimize both agents using the reinforcement learning technique. The learning algorithm is similar to the dual-NMT algorithm described in (Xia et al., 2016), but with adaption for the OIE and OIN tasks, especially on the task related rewards. Compared to the approach of applying the regularization about sharing the same joint probability (Tang et al., 2017), our approach directly optimizes the task objective by introducing task related rewards. Furthermore, our approach is more adaptable than the weight sharing approach adopted in (Chisholm et al., 2017).

3 Orator

3.1 SAOKE Dataset

Symbolic Aided Open Knowledge Expression (SAOKE) is proposed in (Sun et al., 2018) as the form to honestly record the facts that humans can extract from sentences when humans read them. SAOKE uses a unified form - an n-ary tuple:

$$(subject, predicate, object_1, \cdots, object_N),$$

to express four categories of facts: 1) Relation: Verb/preposition-based n-ary relations between entity mentions; 2) Attribute: Nominal attributes for entity mentions; 3) Description: Descriptive phrases of entity mentions; 4) Concept: Hyponymy and synonymy relations among concepts and instances.

Using this SAOKE format, Sun et al. (2018) manually labeled the SAOKE dataset $\mathbb{D}_{\text{SAOKE}}$ by crowdsourcing, which includes more than forty thousand sentence-facts pairs $< \mathcal{S}, \mathcal{F} >$.[2] The labeling procedure is under the supervision of the "Completeness" criterion (Sun et al., 2018), so the facts recorded information in the sentence as much as possible (only auxiliary information and relation between facts are omitted (Sun et al., 2018)). As a result, the SAOKE dataset is a valid open-domain sentence-facts parallel dataset for both OIE and OIN tasks. Table 2 is an example from the SAOKE dataset for an easy understanding of the dual relationship between sentence and facts.

---

[2] `http://ai.baidu.com/broad`

3.2 Model

The Orator is an agent $\mathfrak{O}$ that assembles a set of open-domain facts $\mathcal{F}$ into a sentence $\mathcal{S}$ with probability $P_{\mathfrak{O}}(\mathcal{S}|\mathcal{F},\Theta_{\mathfrak{O}})$, where $\Theta_{\mathfrak{O}}$ is the set of parameters of $\mathfrak{O}$:

$$\text{Orator } \mathfrak{O}: \quad \mathcal{F} \quad \rightarrow \quad P_{\mathfrak{O}}(\mathcal{S}|\mathcal{F},\Theta_{\mathfrak{O}}).$$

For each pair $< \mathcal{S},\mathcal{F} >\in \mathbb{D}_{\text{SAOKE}}$, the set of facts $\mathcal{F}$ is actually expressed as a sequence of facts, in the order of the labeler wrote them. So, the deep sequence to sequence paradigm is suitable to model the Orator. In this work, we build the base Orator model with the attention-based sequence to sequence model, together with copy and coverage mechanism, in a similar way of the implementation of the Logician in (Sun et al., 2018).

3.2.1 Attention based Sequence-to-sequence Learning

The attention-based sequence-to-sequence learning (Bahdanau et al., 2014) first encodes the input fact sequence $\mathcal{F}$ (actually the sequence of N_e-dimensional word embedding vectors) into a N_h-dimensional hidden states $H^{\mathcal{F}} = [h_1^{\mathcal{F}},\cdots,h_{N_S}^{\mathcal{F}}]$ using bi-directional GRU (Gated Recurrent Units) network (Cho et al., 2014). Then, when generating word w_t of the target sentence, the decoder computes the probability of generating w_t by $p(w_t|\{w_1,\cdots,w_{t-1}\},c_t) = g(h_{t-1},s_t,c_t)$, where s_t is the hidden state of the GRU decoder, g is the word generation model, and c_t is the dynamic context vector which focuses attention on specific location l in the input hidden states $H^{\mathcal{F}}$.

For the Orator, we use the copy mechanism to implement the word generation model g and use the coverage mechanism to compute the dynamic context vector c_t.

3.2.2 Copy Mechanism

In the SAOKE dataset, the words in the set of facts (excluding the external symbols) must be in the corresponding sentence, so the problem is suitable to be modeled via the copy mechanism (Gu et al., 2016). In the copy mechanism, when the decoder is considering generating a word w_t, it can either be copied from the source fact sequence $\mathcal{F}$ or select from a vocabulary V:

$$\begin{aligned} p(w_t|w_{t-1},s_t,c_t) \quad = \quad & p_{\mathcal{F}}(w_t|w_{t-1},s_t,c_t) + \\ & p_V(w_t|w_{t-1},s_t,c_t), \end{aligned}$$

where $p_{\mathcal{F}}$ is the probability of copying from $\mathcal{F}$ and p_V is the probability of selecting from V. The details can be found in (Gu et al., 2016).

3.2.3 Coverage Mechanism

To cope with the problem of information lost or redundancy in the generated sentence, the copied histories of previous generated words should be remembered to guide future generation. This could be done through the coverage mechanism (Tu et al., 2016), in which a coverage vector m_j^t is introduced for each word $w_j^{\mathcal{F}}$ in $\mathcal{F}$ and updated at each step t as a gated function of $h_j^{\mathcal{F}},\alpha_{tj},s_{t-1},m_j^{t-1}$. By this means, the coverage vectors remember the historical attentions over source sequence and can be incorporated in the alignment model to generate complete and non-redundant sentences. Detailed formulations can be found in (Tu et al., 2016) and (Sun et al., 2018).

4 Learning the Dual Structure between Knowledge and Natural Language

4.1 Dual Structure between Orator and Logician

In (Sun et al., 2018), an agent $\mathfrak{L}$, called Logician, was trained to convert a sentence $\mathcal{S}$ into a set of facts $\mathcal{F}$ with probability $P_{\mathfrak{L}}(\mathcal{F}|\mathcal{S},\Theta_{\mathfrak{L}})$, where $\Theta_{\mathfrak{L}}$ is the set of parameters of $\mathfrak{L}$:

$$\text{Logician } \mathfrak{L}: \quad \mathcal{S} \quad \rightarrow \quad P_{\mathfrak{L}}(\mathcal{F}|\mathcal{S},\Theta_{\mathfrak{L}}).$$

Obviously, the Logician and Orator can cooperate to supervise each other. Given $< \mathcal{S},\mathcal{F} >\in \mathbb{D}_{\text{SAOKE}}$, the Logician produces a predicted set of facts $\mathcal{F}^*$ for the sentence $\mathcal{S}$, and the Orator can calculate the probability $P_{\mathfrak{O}}(\mathcal{S}|\mathcal{F}^*,\Theta_{\mathfrak{O}})$ of reconstruction $\mathcal{S}$ from $\mathcal{F}^*$. Intuitively, if $\mathcal{F}^*$ loses major information of $\mathcal{S}$, honestly reconstructing $\mathcal{S}$ from $\mathcal{F}^*$ would be impossible, and thus the probability $P_{\mathfrak{O}}(\mathcal{S}|\mathcal{F}^*,\Theta_{\mathfrak{O}})$ would be small. Thus, it is a strong signal to evaluate the quality of $\mathcal{F}^*$. Similarly, when the Orator produces a sentence $\mathcal{S}^*$ for the set of facts $\mathcal{F}$, the probability $P_{\mathfrak{L}}(\mathcal{F}|\mathcal{S}^*,\Theta_{\mathfrak{L}})$ provided by the Logician is a strong signal for evaluating the quality of $\mathcal{S}^*$. These signals are helpful to conquer several problems of the original agents, including information lost, information redundancy, and non-fluency.

Note that the supervision signals $P_{\mathfrak{O}}(\mathcal{S}|\mathcal{F}^*,\Theta_{\mathfrak{O}})$ and $P_{\mathfrak{L}}(\mathcal{F}|\mathcal{S}^*,\Theta_{\mathfrak{L}})$ do not rely on any supervised parallel corpus. Thus, similar to the application of dual learning paradigm on NMT task (Xia et al., 2016), it is theoretically possible to use unparalleled sentences and sets of facts to compute these signals. However, unsupervised collections of fact-groups that can be reasonably narrated in a sentence are not naturally available. Currently, the only available collection is the sets of facts provided by the SAOKE dataset, where the supervised information is available. As a result, we implement our dual learning system in a supervised approach, which uses the reinforcement learning algorithm to optimize the Orator and the Logician. The involved rewards are described in the next subsection, and then the algorithm is detailed in the last subsection.

4.2 Rewards

Given $< \mathcal{S}, \mathcal{F} > \in \mathbb{D}_{\text{SAOKE}}$, we sample a set of facts $\mathcal{F}^*$ from distribution $P_{\mathfrak{L}}(\cdot|\mathcal{S}, \varTheta_{\mathfrak{L}})$ and a sentence $\mathcal{S}^*$from distribution $P_{\mathfrak{O}}(\cdot|\mathcal{F}, \varTheta_{\mathfrak{O}})$. Following rewards are introduced into the proposed dual learning system, and the relationships between them are shown in Figure 1.

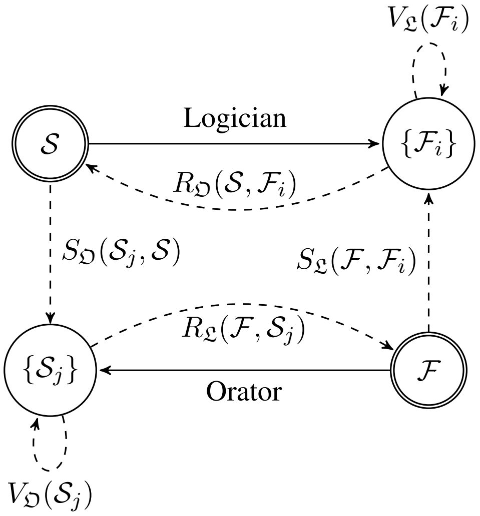

Figure 1: Illustration of the dual learning system of Logician and Orator.

4.2.1 Reconstruction Rewards

Following the idea described in above subsection, we design the reconstruction reward for the Orator as:

$$R_{\mathfrak{O}}(\mathcal{S}^*, \mathcal{F}) = \log P_{\mathfrak{L}}(\mathcal{F}|\mathcal{S}^*, \varTheta_{\mathfrak{L}}),$$

and that for the Logician as:

$$R_{\mathfrak{L}}(\mathcal{F}^*, \mathcal{S}) = \log P_{\mathfrak{O}}(\mathcal{S}|\mathcal{F}^*, \varTheta_{\mathfrak{O}}).$$

4.2.2 Similarity Rewards

Since the SAOKE dataset has label information, the similarities between the predicted results and the ground truths can be used as rewards.

For the Orator, since the $\mathcal{S}^*$ can be viewed as the summarization of $\mathcal{S}$, we use the widely used ROUGE-L (Lin, 2004) measure in the text summarization field to evaluate the quality of $\mathcal{S}^*$:

$$S_{\mathfrak{O}} = \text{ROUGE}_L(\mathcal{S}, \mathcal{S}^*).$$

For the Logician, we use following procedure to calculate the similarity between $\mathcal{F}$ and $\mathcal{F}^*$. First, we compute the similarity between each predicted fact $f^* \in \mathcal{F}^*$ and each ground truth fact $f \in \mathcal{F}$ with following measure:

$$SimFact(f^*, f) = \frac{\sum_{i=1}^{\min(|f^*|,|f|)} SimStr(f_i^*, f_i)}{\max(|f^*|, |f|)},$$

where f_i^* and f_i denote the i-th element of tuples of fact f^* and f, $SimStr(\cdot, \cdot)$ denotes the gestalt pattern matching (Ratcliff and Metzener, 1988) measure for two strings, and $|\cdot|$ is the cardinality function. Then, each predicted fact in $\mathcal{F}^*$ is aligned to its corresponding ground-truth fact in $\mathcal{F}$ by solving a linear assignment problem (Wikipedia, 2017) to maximize the sum of similarities between the aligned facts. Finally, the similarity reward for the Logician is calculated by:

$$S_{\mathfrak{L}}(\mathcal{F}^*, \mathcal{F}) = \frac{\sum SimFact(f*, f)}{\max(|\mathcal{F}^*|, |\mathcal{F}|)},$$

where $f^* \in \mathcal{F}^*, f \in \mathcal{F}$ are aligned pair of facts.

4.2.3 Validity Rewards

For the Orator, the output is expected as a valid natural language sentence, so the validity reward can be defined as:

$$V_{\mathfrak{O}}(\mathcal{S}^*) = LM(\mathcal{S}^*),$$

where the $LM(\cdot)$ is a language model.

For the Logician, the output should represent a valid collection of facts, which means: 1) the output can be parsed into a collection of facts; 2) there is no duplicated fact (identified by the $SimFact$ value larger than 0.85) in the parsed collection. The validity reward for Logician is defined as:

$$V_{\mathfrak{L}}(\mathcal{F}^*) = \begin{cases} 0 & \text{if } \mathcal{F}^* \text{is valid;} \\ -1 & \text{otherwise.} \end{cases}$$

Algorithm 1 A simple dual-learning algorithm for facts extraction and expression

```
Require:
    A set of sentence-facts pairs $\{< \mathcal{S}, \mathcal{F} >\}$;
    An initial Logician $\mathfrak{L}$ and an initial Orator $\mathfrak{O}$;
    Beam size $K$;
repeat
    Sample a sentence-facts pair $< \mathcal{S}, \mathcal{F} >$;
    Logician produces $K$ sets of facts $\mathcal{F}_1, \cdots, \mathcal{F}_K$ from $\mathcal{S}$ via beam search;
    for each set of facts $\mathcal{F}_i$ do
        Compute the reward for $\mathcal{F}_i$ as:
            $r_i^{\mathcal{F}} = \alpha_1 R_{\mathfrak{O}}(\mathcal{S}, \mathcal{F}_i) + \alpha_2 V_F(\mathcal{F}) + \alpha_3 S_F(\mathcal{F}, \mathcal{F}_i).$
    end for
    Compute the total reward $r = \frac{1}{K} \sum_{i=1}^{K} r_i^{\mathcal{F}}$;
    Compute the stochastic gradient of $\Theta_{\mathfrak{L}}$:
        $\nabla_{\Theta_{\mathfrak{L}}} \hat{E}[r] = \frac{1}{K} \sum_{i=1}^{K} r_i^{\mathcal{F}} \mathfrak{D}_{\Theta_{\mathfrak{L}}}(\mathcal{S}, \mathcal{F}_i)$
    Compute the stochastic gradient of $\Theta_{\mathfrak{O}}$:
        $\nabla_{\Theta_{\mathfrak{O}}} \hat{E}[r] = \frac{\alpha_1}{K} \sum_{i=1}^{K} \mathfrak{D}_{\Theta_{\mathfrak{O}}}(\mathcal{F}_i, \mathcal{S})$
    Model updates:
        $\Theta_{\mathfrak{L}} \leftarrow \Theta_{\mathfrak{L}} + \eta_{\mathfrak{L}} \cdot \nabla_{\Theta_{\mathfrak{L}}} \hat{E}[r],$
        $\Theta_{\mathfrak{O}} \leftarrow \Theta_{\mathfrak{O}} + \eta_{\mathfrak{O}} \cdot \nabla_{\Theta_{\mathfrak{O}}} \hat{E}[r].$

    Sample a sentence-facts pair $< \mathcal{S}, \mathcal{F} >$;
    Orator produces $K$ sentences $\mathcal{S}_1, \cdots, \mathcal{S}_K$ from $\mathcal{F}$ via beam search;
    for each sentence $\mathcal{S}_i$ do
        Compute the reward for $\mathcal{S}_i$ as:
            $r_i^{\mathcal{S}} = \beta_1 R_{\mathfrak{L}}(\mathcal{F}, \mathcal{S}_i) + \beta_2 V_S(\mathcal{S}_i) + \beta_3 S_S(\mathcal{S}, \mathcal{S}_i).$
    end for
    Compute the total reward $r = \frac{1}{K} \sum_{i=1}^{K} r_i^{\mathcal{S}}$;
    Compute the stochastic gradient of $\Theta_{\mathfrak{O}}$,
        $\nabla_{\Theta_{\mathfrak{O}}} \hat{E}[r] = \frac{1}{K} \sum_{i=1}^{K} r_i^{\mathcal{S}} \mathfrak{D}_{\Theta_{\mathfrak{O}}}(\mathcal{S}_i, \mathcal{F})$
    Compute the stochastic gradient of $\Theta_{\mathfrak{L}}$:
        $\nabla_{\Theta_{\mathfrak{L}}} \hat{E}[r] = \frac{\beta_1}{K} \sum_{i=1}^{K} \mathfrak{D}_{\Theta_{\mathfrak{L}}}(\mathcal{F}_i, \mathcal{S})$
    Model updates:
        $\Theta_{\mathfrak{O}} \leftarrow \Theta_{\mathfrak{O}} + \eta_{\mathfrak{O}} \cdot \nabla_{\Theta_{\mathfrak{O}}} \hat{E}[r],$
        $\Theta_{\mathfrak{L}} \leftarrow \Theta_{\mathfrak{L}} + \eta_{\mathfrak{L}} \cdot \nabla_{\Theta_{\mathfrak{L}}} \hat{E}[r].$
until convergence
```

4.3 Algorithm

For each pair $< \mathcal{S}, \mathcal{F} > \in \mathbb{D}_{\text{SAOKE}}$, the following procedures are performed respectively (details are shown in Algorithm 1):

4.3.1 Learning from Sentence to Facts

We sample $\mathcal{F}^*$ from the Logician $P_{\mathfrak{L}}(\cdot|\mathcal{S}, \Theta_{\mathfrak{L}})$ and calculate the total reward for $\mathcal{F}^*$ by

$$
\begin{aligned}
r_{\mathfrak{L}} = {} & \alpha_1 \cdot R_{\mathfrak{L}}(\mathcal{F}^*, \mathcal{S}) + \alpha_2 \cdot V_{\mathfrak{L}}(\mathcal{F}^*) + \\
& \alpha_3 \cdot S_{\mathfrak{L}}(\mathcal{F}^*, \mathcal{F}),
\end{aligned}
$$

where $\sum \alpha_i = 1$. The gradients of the expected reward $E[r_{\mathfrak{L}}]$ to the parameters of agents can be computed as follows, according to the policy gradient theorem (Sutton et al., 1999):

$$
\begin{aligned}
\nabla_{\Theta_{\mathfrak{L}}} E[r_{\mathfrak{L}}] &= E[r_{\mathfrak{L}} \mathfrak{D}_{\Theta_{\mathfrak{L}}}(\mathcal{F}^*, \mathcal{S})], \\
\nabla_{\Theta_{\mathfrak{O}}} E[r_{\mathfrak{L}}] &= E[\alpha_1 \mathfrak{D}_{\Theta_{\mathfrak{O}}}(\mathcal{S}, \mathcal{F}^*)].
\end{aligned}
$$

where $\mathfrak{D}_{\Theta_{\mathfrak{L}}}(\mathcal{F}, \mathcal{S}) = \nabla_{\Theta_{\mathfrak{L}}} \log P_{\mathfrak{L}}(\mathcal{F}|\mathcal{S}, \Theta_{\mathfrak{L}})$ and $\mathfrak{D}_{\Theta_{\mathfrak{O}}}(\mathcal{S}, \mathcal{F}) = \nabla_{\Theta_{\mathfrak{O}}} \log P_{\mathfrak{O}}(\mathcal{S}|\mathcal{F}, \Theta_{\mathfrak{O}})$.

4.3.2 Learning from Facts to Sentence

We sample $\mathcal{S}^*$ from the Logician $P_{\mathfrak{O}}(\cdot|\mathcal{F}, \Theta_{\mathfrak{O}})$ and define the total reward for $\mathcal{S}^*$ by:

$$
\begin{aligned}
r_{\mathfrak{O}} = {} & \beta_1 R_{\mathfrak{O}}(\mathcal{S}^*, \mathcal{F}) + \beta_2 V_{\mathfrak{O}}(\mathcal{S}^*) + \\
& \beta_3 S_{\mathfrak{O}}(\mathcal{S}^*, \mathcal{S}),
\end{aligned}
$$

where $\sum \beta_i = 1$. The gradients can be computed as follows:

$$
\begin{aligned}
\nabla_{\Theta_{\mathfrak{L}}} E[r_{\mathfrak{O}}] &= E[\beta_1 \mathfrak{D}_{\Theta_{\mathfrak{L}}}(\mathcal{F}, \mathcal{S}^*)], \\
\nabla_{\Theta_{\mathfrak{O}}} E[r_{\mathfrak{O}}] &= E[r_{\mathfrak{O}} \mathfrak{D}_{\Theta_{\mathfrak{O}}}(\mathcal{S}^*, \mathcal{F})].
\end{aligned}
$$

In practice, we use beam search (Sutskever et al., 2014) to obtain high-quality samples as $\mathcal{F}^*$ and $\mathcal{S}^*$, and estimate the true gradient with the empirical average of gradients over these samples.

5 Experimental Results

5.1 Experimental Design

First, we evaluate the performance of each agent fine-tuned by the dual learning procedure on the

SAOKE dataset. Then we evaluate the Orator on noisy facts, which accords with real OIN application scenarios. Last, we investigate the behavior of agents in the dual system.

In the experiments, the SAOKE dataset is split into the training set, validating set and testing set with ratios of 80%, 10%, 10%, respectively. For each algorithm involved in the experiments, we perform grid search to find the optimal super-parameters, and the model with the best performance on the validating set is chosen as the learnt model to be evaluated on the testing set.

5.1.1 Evaluation Metric

For the Orator, BLEU-4 and ROUGE-L are used to measure how well the output matches the ground truth sentence.

For the Logician, based on the fact-equivalence judgment proposed in (Sun et al., 2018), we compute the Precision(P), Recall (R) and F1-score over the testing set of the SAOKE dataset as the evaluation metric.

5.1.2 Agent Implementation

For the Orator, we make a vocabulary V with size 72,591 by collecting all web pages from Baidu Baike website[3] (a Chinese alternative to Wikipedia) and identifying the words occurred in more than 100 web pages. For the Orator, the dimension of embedding vectors is set to $N_e = 256$, and the dimension of hidden states is set to $N_h = 256$. We use a three-layer bi-directional GRU with dimension 128 as the encoder. All dimensions of hidden states in the decoder are set to 256.

For the Logician, we implement the model described in (Sun et al., 2018), including the shallow tag information and the gated dependency attention mechanism.

Furthermore, to provide an intuitive comprehension of the OIN task, we implement a rule-based method for OIN task. For each sequence of facts in the SAOKE dataset, the method first identifies the subsequences in which the facts share the same subject. Then it preserves the subject of the first fact in each subsequence and removes the subjects of following facts (by replacing it with an empty string). It is necessary since the SAOKE dataset requires the shared subject to be repeated for completeness of the related facts. At last, each fact is formatted into a string by filling the objects into the placeholders of the predicate and these strings are concatenated with commas to form the final sentence.

5.1.3 Reward Implementation

For the validity reward of the Orator, the language model is trained using an RNN based method (Mikolov et al., 2010) with the same vocabulary V and the web pages from Baidu Baike website.

For the reconstruction reward of the Orator, since the Logician needs the shallow tag and dependency information of $\mathcal{S}^*$ as inputs, the information is extracted using the LTP tool-set (Che et al., 2010) and then fed to the Logician.

5.1.4 Training

When training the base model for each agent, the batch size is set to 20. When training two agents in dual learning, the batch size is set to 12, and the beam size is set to 3. Both agents are trained using the stochastic gradient descent (SGD) with RMSPROP strategy (Hinton et al., 2012) and early-stop strategy on the validating set. In dual learning, the super-parameters, including α_i , β_i, is determined by grid-search.

5.2 Evaluation of Agents on the SAOKE dataset

First, we evaluate the performance of the agents optimized by the dual learning method. To identify the contribution of the dual structure, we train another pair of agents with $\alpha_1 = 0$ and $\beta_1 = 0$ in Algorithm 1 to exclude the dual information. Without the dual information, these two agents are trained independently to each other with reinforcement learning on their own supervised information. We name these two agents as R-Logician and R-Orator, where "R" means "Reinforced". In the experimental results of this paper, the symbol at the top mark means that the marked result is significantly different (with $p = 0.05$) with the corresponding result of the agent with the specific mark.

Methods	Precision	Recall	F1
Logician*	0.449	0.400	0.423
R-Logician$^{\mp}$	0.462*	0.432*	0.446*
Logician@Dual	0.494$^{*\mp}$	0.426*	0.457$^{*\mp}$

Table 3: Performance of the Logicians.

The experimental results for the Logician agents are shown in Table 3, from which we can observe a significant performance improvement

---

[3] http://baike.baidu.com

from Logician to R-Logician and also from R-Logician to Logician@Dual. The experimental results for the Orator agents are shown in Table 4. The neural based Orator agents significantly outperform the rule-based agent. For both evaluation metric, the R-Orator and Orator@Dual are both significantly outperform the original Orator. The Orator@Dual significantly outperforms the R-Orator on the BLEU-4 score, but is not significantly different on the ROUGE-L score.

Methods	BLEU-4	ROUGE-L
Rule*	0.257	0.434
Orator*	0.401*	0.556*
R-Orator$^{\mp}$	0.405$^{\star *}$	0.559$^{\star *}$
Orator@Dual	0.419$^{\star * \mp}$	0.559$^{\star *}$

Table 4: Performance of the Orators.

By comparing the performance of R-agents and the agents@Dual, we can observe that agents@Dual generally achieve better performance on precision, but may recall less information, resulting in smaller advances in the balanced evaluation metric (F1 and ROUGE-L). This may imply that the agents tend to provide easy input for each other for higher accuracy, by neglecting some difficult part of the problem which they currently cannot handle properly. This interesting phenomenon is the subject of our future research.

5.3 Evaluation of Orator on Noisy Facts

Experiments in the previous subsection show the performance of Orators to narrate a set of human-labeled facts. In practice, however, the input to the Orator might not be the human-labeled perfect facts, but some noisy facts automatically extracted by OIE algorithms. In this subsection, we make a collection of sets of noisy facts by feeding the sentences in the testing set of the SAOKE dataset to the base Logician model and collecting the outputs. Then we evaluate the series of Orator models on these noisy facts, and report their performance at Table 5, from which we can see the performance improvement from the Orator to Orator@Dual.

Methods	BLEU-4	ROUGE-L
Orator*	0.428	0.565
R-Orator$^{\mp}$	0.431*	0.567*
Orator@Dual	0.458$^{* \mp}$	0.572$^{* \mp}$

Table 5: Performance of the Orators on noisy facts.

5.4 Evaluation of the Dual System

In this section, we investigate the behavior of agents in the dual system. We first examine the procedure $\mathcal{F} \xrightarrow{Orator} \mathcal{S}^{*} \xrightarrow{Logician} \mathcal{F}^{**}$, that is, for each $\mathcal{F}$ in the testing set of the SAOKE dataset, let the Orator narrate it into a sentence $\mathcal{S}^{*}$, and then let the Logician to extract facts $\mathcal{F}^{**}$ from $\mathcal{S}^{*}$. Then the quality of $\mathcal{F}^{**}$ is measured by comparing it with $\mathcal{F}$. Then we examine $\mathcal{S} \xrightarrow{Logician} \mathcal{F}^{*} \xrightarrow{Orator} \mathcal{S}^{**}$, which is the reverse procedure. The comparison is made between the family of base agents and that of the dual-trained agents. The results are shown in Table 6, and two instance of these two experiments are shown in Table 7 and 8 respectively. From these results, we can observe large improvements of reconstruction quality on both directions.

	$\mathcal{F} \xrightarrow{Orator} \mathcal{S}^{*} \xrightarrow{Logician} \mathcal{F}^{**}$		
Methods	Precision	Recall	F1
Base*	0.574	0.488	0.527
Dual	0.657*	0.565*	0.608*

	$\mathcal{S} \xrightarrow{Logician} \mathcal{F}^{*} \xrightarrow{Orator} \mathcal{S}^{**}$	
Methods	BLEU-4	ROUGE-L
Base*	0.428	0.565
Dual	0.635*	0.635*

Table 6: Reconstruction performance for the Logicians and the Orators.

6 Conclusion

In this paper, we investigate the OIN task and its duality to the OIE task. The proposed Orator has shown its ability to fulfill the OIN task, that is, assembling open-domain facts into high quality sentences. Furthermore, our attempt to utilize the duality between OIN and OIE tasks for improving the performances for both OIN and OIE agents accomplishes a preliminary success.

Our work suggests at least three future research topics: Firstly, one can enrich the theoretical study of the duality between the OIE and OIN tasks. Secondly, one can investigate how to conquer the barrier of the absence of an extensive collection of reasonable sets of open-domain facts and incorporate unsupervised information into this Logician-Orator dual learning structures for further improvement. Lastly, one can also interested in developing task-oriented rewards for adapting the agent to a specific task, for example, the answer generation task for open-domain KBQA system.

$\mathcal{S}$	大综货物吞吐量均保持两位数增长，其中铁矿石吞吐量突破4500万吨，木材吞吐量突破600万立方，均创历史新高，成为全国进口木材第一大港。
$\mathcal{S}$ in English	The throughput of all integrated cargoes kept a double-digit growth. Among them, the throughput of iron ore exceeded 45 million tons and the throughput of timber exceeded 6 million cubic meters, all of which hit record highs, and became the country's largest port of timber imports.

	Base Models	Dual Models
$\mathcal{S} \xrightarrow{Logician} \mathcal{F}^*$	(大综货物吞吐量, 保持, 两位数增长) (铁矿石吞吐量, 突破, 4500万吨) (木材吞吐量, 突破, 600万立方) (创历史, DESC, 新高) (_, 成为, 全国进口木材第一大港)	(大综货物吞吐量, 均保持, 两位数增长) (铁矿石吞吐量, 突破, 4500万吨) (木材吞吐量, 突破, 600万立方) (_, 均创, 历史新高) (_, 成为, 全国进口木材第一大港)
$\mathcal{S} \xrightarrow{Logician} \mathcal{F}^*$ in English	(The throughput of all integrated cargoes, kept, double-digits growth) (Throughput of iron ore, exceeded, 45 million tons) (Throughput of wood, breakthrough, 6 million cubic meters) (Hit historical, DESC, new high) (_, become, the country's largest port of timber imports)	(The throughput of integrated cargoes, all kept, double-digit growth) (Throughput of iron ore, exceeded, 45 million tons) (Throughput of timber, exceeded 6, million cubic meters) (_, all hit, record highs) (_, become, the country's largest port of timber imports)
$\mathcal{S} \xrightarrow{Logician}$ $\mathcal{F}^* \xrightarrow{Orator} \mathcal{S}^{**}$	大综货物吞吐量保持两位数增长，突破4500万吨，突破600万立方，新高，成为全国进口木材第一大港。	大综货物吞吐量均保持两位数增长，铁矿石吞吐量突破4500万吨，木材吞吐量突破600万立方，均创历史新高，成为全国进口木材第一大港。
$\mathcal{S} \xrightarrow{Logician}$ $\mathcal{F}^* \xrightarrow{Orator} \mathcal{S}^{**}$ in English	The throughput of all integrated cargoes kept double-digit growth, breaking 45 million tons, breaking 6 million cubic meters, a new high, became the country's largest port of timber imports.	The throughput of all integrated cargoes kept double-digit growth. The throughput of iron ore exceeded 45 million tons and the throughput of timber exceeded 6 million cubic meters, all hit a record high and became the country's largest port of timber imports.

Table 7: Examples for sentence reconstruction.

$\mathcal{F}$	(营业法人, 具备, 中专（高中）以上学历) (营业法人, 有, 一定的[管理|经营]能力) (营业法人, 具备, [较强的服务理念|团队合作精神])
$\mathcal{F}$ in English	(Business legal person, is with, technical secondary school (high school) or above) (Business legal person, has, certain [managerial | operational] ability) (Business legal person, is with, [strong philosophy of service | teamwork spirit])

	Base Models	Dual Models
$\mathcal{F} \xrightarrow{Orator} \mathcal{S}^*$	营业法人具备中专（高中）以上学历，有一定的管理和经营能力，具备较强的服务理念和团队合作精神。	营业法人具备中专（高中）以上学历，营业法人有一定的管理和经营能力，具备较强的服务理念和团队合作精神。
$\mathcal{F} \xrightarrow{Orator} \mathcal{S}^*$ in English	Business legal person is with secondary (high school) or above, has certain managerial and operational capabilities, and is with strong philosophy of service and teamwork spirit.	Business legal person is with secondary (high school) or above, business legal person has certain managerial and operational capabilities, and is with strong philosophy of service and teamwork spirit.
$\mathcal{F} \xrightarrow{Orator}$ $\mathcal{S}^* \xrightarrow{Logician} \mathcal{F}^{**}$	(营业法人具备中专（高中）以上学历, 有, 一定的[管理|经营]能力) (营业法人具备中专的[管理|经营]能力, 具备, 较强的[服务理念|团队合作精神])	(营业法人, 具备, 中专（高中）以上学历) (营业法人,有,[一定的管理|经营能力]) (营业法人, 具备, 较强的[服务理念|团队合作精神])
$\mathcal{F} \xrightarrow{Orator}$ $\mathcal{S}^* \xrightarrow{Logician} \mathcal{F}^{**}$ in English	(Business legal person with technical secondary school (high school) or above, has, certain [managerial | operational] ability) (Business legal person with technical secondary school (high school) or above, is with, [strong philosophy of service | teamwork spirit])	(Business legal person, is with, technical secondary school (high school) or above) (Business legal person, has, [certain managerial | operational ability]) (Business legal person, is with, [strong philosophy of service | teamwork spirit])

Table 8: Examples for fact reconstruction.

References

Shubham Agarwal and Marc Dymetman. 2017. A Surprisingly Effective Out-of-the-Box Char2char Model on the E2E NLG Challenge Dataset. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, August, pages 158–163.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. *The Semantic Web*, 4825 LNCS:722–735.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a Collaboratively Created Graph database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. ACM.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. LTP: A Chinese Language Technology Platform. In *Proceedings of the Coling*, pages 13–16.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to Generate One-sentence Biographies from Wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 1, pages 633–642.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An Analysis of Open Information Extraction based on Semantic Role Labeling. In *Proceedings of the Sixth International Conference on Knowledge Capture*, pages 113–120. ACM Press.

Janara Christensen, Mausam, Stephen Soderland, Oren Etzioni, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards Coherent Multi-Document Summarization. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Section 3, pages 1163–1173.

Janara Christensen, Stephen Soderland, and Gagan Bansal. 2014. Hierarchical Summarization: Scaling Up Multi-Document Summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 902–912.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question Generation for Question Answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open Information Extraction: The Second Generation. In *Proceeding of International Joint Conference on Artificial Intelligence*, pages 3–10.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2014. Open Question Answering Over Curated and Extracted Knowledge Bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1156–1165.

Federico Gaspari. 2006. Look Who's Translating: Impersonations, Chinese Whispers and Fun with Machine Translation on the Internet. In *EAMT-2006: 11th Annual Conference of the European Association for Machine Translation*, pages 19–20.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1631–1640.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Overview of Mini-batch Gradient Descent. Technical report.

Nanda Kambhatla. 2004. Combining Lexical, Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering Complex Questions Using Open Information Extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 311—-316.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 146–157. Association for Computational Linguistics.

C Y Lin. 2004. Rouge: A Package for Automatic Evaluation of Summaries. In *Proceedings of the 2004 ACL Workshop on Text Summarization Branches Out*, 1, pages 25–26.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural Relation Extraction with Selective Attention over Instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2124–2133.

Mausam. 2016. Open Information Extraction Systems and Downstream Applications. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 4074–4077.

T Mikolov, M Karafiat, L Burget, J Cernocky, and S Khudanpur. 2010. Recurrent Neural Network based Language Model. In *Proceedings of Interspeech*, September, pages 1045–1048.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant Supervision for Relation Extraction without Labeled Data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language*, volume 2, page 1003, Morristown, NJ, USA. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1116, Stroudsburg, PA, USA. Association for Computational Linguistics.

Harinder Pal and Mausam. 2016. Demonyms and Compound Relational Nouns in Nominal Open IE. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 35–39.

John W Ratcliff and David E Metzener. 1988. Pattern Matching: The Gestalt Approach. *Dr Dobb's*, 13(7).

Mrinmaya Sachan and Eric Xing. 2018. Self-Training for Jointly Learning to Ask and Answer Questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 629–640.

Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534.

Tomohiro Shigenobu. 2007. Evaluation and Usability of Back Translation for Intercultural Communication. In *International Conference on Usability and Internationalization*, pages 259–265. Springer.

Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. Open IE as an Intermediate Structure for Semantic Tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 303–308.

Mingming Sun, Xu Li, Xin Wang, Miao Fan, Yue Feng, and Ping Li. 2018. Logician: A Unified End-to-End Neural Approach for Open-Domain Information Extraction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, February, pages 556–564, New York, New York, USA. ACM Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, volume 155, page 9.

Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *In Advances in Neural Information Processing Systems 12*, pages 1057–1063.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. *In Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 76–85.

Menno Van Zaanen and Simon Zwarts. 2006. Unsupervised Measurement of Translation Quality using Multi-engine, Bi-directional Translation. In *Australasian Joint Conference on Artificial Intelligence*, pages 1208–1214. Springer.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christoph Gravier, Frederique Laforest, Jonathon Hare, and Elena Simperl. 2017. Neural Wikipedian: Generating Textual Summaries from Knowledge Base Triples. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*.

Wikipedia. 2017. Assignment problem— Wikipedia, The Free Encyclopedia.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in Data-to-Document Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2243–2253.

Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual Learning for Machine Translation. In *Advances in Neural Information Processing Systems 29*, pages 1–9.

Yingce Xia, Tao Qin, Wei Chen, Jiang Bian, Nenghai Yu, and Tie-Yan Liu. 2017. Dual Supervised Learning. In *Proceedings of 34th International Conference on Machine Learning*.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural Generative Question Answering. In *Proceedings of*

2016 NAACL Human-Computer Question Answering Workshop, pages 36–42.

Dmitry Zelenko, Chinatsu Aone, Anthony Richardella, Jaz Kandola, Thomas Hofmann, Tomaso Poggio, and John Shawe-Taylor. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, 3:1083–1106.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1227–1236.

MemoReader: Large-Scale Reading Comprehension through Neural Memory Controller

Seohyun Back[1,2] **Seunghak Yu**[1,*] **Sathish Indurthi**[1] **Jihie Kim**[1] **Jaegul Choo**[2,*]
[1] Samsung Research, Seoul, Korea
[2] Korea University, Seoul, Korea
{scv.back, seunghak.yu, s.indurthi, jihie.kim}@samsung.com
jchoo@korea.ac.kr

Abstract

Machine reading comprehension helps machines learn to utilize most of the human knowledge written in the form of text. Existing approaches made a significant progress comparable to human-level performance, but they are still limited in understanding, up to a few paragraphs, failing to properly comprehend lengthy document. In this paper, we propose a novel deep neural network architecture to handle a long-range dependency in RC tasks. In detail, our method has two novel aspects: (1) an advanced memory-augmented architecture and (2) an expanded gated recurrent unit with dense connections that mitigate potential information distortion occurring in the memory. Our proposed architecture is widely applicable to other models. We have performed extensive experiments with well-known benchmark datasets such as TriviaQA, QUASAR-T, and SQuAD. The experimental results demonstrate that the proposed method outperforms existing methods, especially for lengthy documents.

1 Introduction

Most of the human knowledge has been stored in the form of text. Reading comprehension (RC) to understand this knowledge is a major challenge that can vastly increase the range of knowledge available to the machines. Many neural network-based methods have been proposed, pushing performance close to a human level. Nonetheless, there still exists room to improve the performance especially in comprehending lengthy documents that involve complicated reasoning processes. We identify the main bottleneck as the lack of the long-term memory and its improper controlling mechanism.

Previously, several memory-augmenting methods have been proposed to solve the long-term dependency problem. For example, in relatively simple tasks such as bAbI tasks (Weston et al., 2015), Graves et al. (2014, 2016); Henaff et al. (2017) proposed methods that handle the external memory to address long-term dependency. Inspired by these approaches, we develop a customized memory controller along with an external memory augmentation (Graves et al., 2016) for complicated RC tasks. However, we found that the memory controller is susceptible to information distortion as neural networks become deeper, this distortion can hinder the performance.

To overcome this issue, we propose two novel strategies that improve the memory-handling capability while mitigating the information distortion. We extend the memory controller with a residual connection to alleviate the information distortion occurring in it. We also expand the gated recurrent unit (GRU) (Cho et al., 2014) with a dense connection that conveys enriched features to the next layer containing the original as well as the transformed information. We conducted extensive experiments through several benchmark datasets such as TriviaQA, QUASAR-T, and SQuAD. The results show that the proposed model outperforms all the published results. We also integrated the proposed memory controller and the expanded GRU cell block with other existing methods to ensure that our proposed components are widely applicable. The results show that our components consistently bring performance improvement across various state-of-the-art architectures.

The main contributions of this work include the following: (1) We propose an extended memory controller module for RC tasks. (2) We propose a densely connected encoder block with self attention to provide rich representation of given data, reducing information loss due to deep layers of the network. (3) We present the state-of-the-art results

---

[*] To whom correspondence should be addressed.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2131–2140
Brussels, Belgium, October 31 - November 4, 2018.

in lengthy-document RC tasks such as TriviaQA and QUASAR-T as well as relatively short document RC tasks such as SQuAD.

2 Proposed Method

This section presents two of our proposed components in detail, as depicted in Figure 1.

2.1 Memory Controller

Our first proposed component is an advanced external memory controller module for solving RC tasks. We modified the recently proposed memory controller (Graves et al., 2016) by using our new encoder block and layer-wise residual connections. These modifications enable the memory controller to reason over a lengthy document, leading to the overall performance improvement.

This layer takes input as a sequence of vector representations corresponding to individual tokens, $\mathbf{d}_t \in \mathbb{R}^l$, where l is the given vector dimension. For example, such input can be the output of the co-attention layer in Section 3. The operation of this layer is defined as

$$\mathbf{o}_t, \mathbf{i}_t = \text{Controller}(\mathbf{d}_t, M_{t-1}).$$

That is, at time step t, the controller generates an interface vector $\mathbf{i}_t$ for read and write operations and an output vector $\mathbf{o}_t$ based on the input vector $\mathbf{d}_t$ and the external memory content from the previous time step, $M_{t-1} \in \mathbb{R}^{p \times q}$, where p is the memory size and q is the vector dimension of each memory.

Through this controller, we encode an input $D = \{\mathbf{d}_t\}_{t=1}^n$ to $\{\mathbf{x}_t\}_{t=1}^n$ by using the encoder block, i.e.,

$$\{\mathbf{x}_t\}_{t=1}^n = \text{EncoderBlock}^x(D) \in \mathbb{R}^{n \times k},$$

where k is the output dimension of the encoder block. In general, this block is implemented as a recurrent unit, e.g., GRU (Cho et al., 2014). In our model, we replace it with our dense encoder block with self attention (DEBS), as will be discussed in Section 2.2.

To generate a memory-augmented vector $\mathbf{z}_t$, we concatenate $\mathbf{x}_t$ with the vectors read from the previous time step memory, M_{t-1}, i.e.,

$$\mathbf{z}_t = [\mathbf{x}_t; \mathbf{m}_{t-1}^1; \cdots; \mathbf{m}_{t-1}^s] \in \mathbb{R}^{k+sq},$$

where s represents the number of read heads in the memory interface. We then feed the vector $\mathbf{z}_t$ to the bi-directional GRU (BiGRU) layer and obtain the output vector $\mathbf{h}_t^m$ as

$$\mathbf{h}_t^m = \text{BiGRU}(\mathbf{z}_t, \mathbf{h}_{t-1}^m, \mathbf{h}_{t+1}^m) \in \mathbb{R}^{2l}.$$

Afterwards, we generate output vector $\mathbf{v}_t$ as the weighted sum of the BiGRU output and read vectors from the memory in the current step, i.e.,

$$\mathbf{v}_t = W_h \mathbf{h}_t^m + W_m [\mathbf{m}_t^1; \cdots; \mathbf{m}_t^s] \in \mathbb{R}^{2l}.$$

Finally, we add a residual connection between the input $\mathbf{d}_t$ and the output $\mathbf{v}_t$ to mitigate any possible information distortion that can occur while accessing the memory, resulting in a the output vector that can handle long-term dependency, i.e.,

$$\mathbf{o}_t = \text{ReLU}(W_v \mathbf{v}_t + \mathbf{d}_t) \in \mathbb{R}^l.$$

For further details on how the interface vector works, we refer the readers to Graves et al. (2016) as well as our supplemental material.

2.2 Dense Encoder Block with Self Attention

The second novel component we propose is a dense encoder block with self attention (DEBS), which further improves a GRU cell. Recently, Huang et al. (2017a) proposed that adding a connection between each layer to the other layers in convolution networks can help to properly convey the information across multiple layers. Inspired by this, we add such dense connections that concatenate the input to a particular layer to its output. We also add a self-attention module to this block, to properly address long-term dependency in a length document. In this manner, our encoder block maintains the necessary information not only along the vertical direction (across layers) through dense connections but also along the horizontal direction (across time steps) through self attention.

DEBS takes the input vector sequence with its length as n and transforms each vector to an l-dimensional vector $\mathbf{p}_t$ through the fully connected layer with ReLU as a nonlinear unit and generates a contextually encoded vector $\mathbf{r}_t$ as

$$\mathbf{r}_t = \text{BiGRU}(\mathbf{p}_t, \mathbf{r}_{t-1}, \mathbf{r}_{t+1}) \in \mathbb{R}^{2l}.$$

Then we concatenate each output vector $\mathbf{r}_t$ to the projected input $\mathbf{p}_t$ to obtain $\mathbf{g}_t = [\mathbf{r}_t; \mathbf{p}_t] \in \mathbb{R}^{3l}$ and pass it to the self-attention layer. The self-attention layer then calculates the similarity map $S^g \in \mathbb{R}^{n \times n}$ using the tri-linear function as

$$s_{ij}^g = \mathbf{w}_a \cdot \mathbf{g}_i + \mathbf{w}_b \cdot \mathbf{g}_j + \mathbf{w}_f \cdot (\mathbf{g}_i \odot \mathbf{g}_j),$$

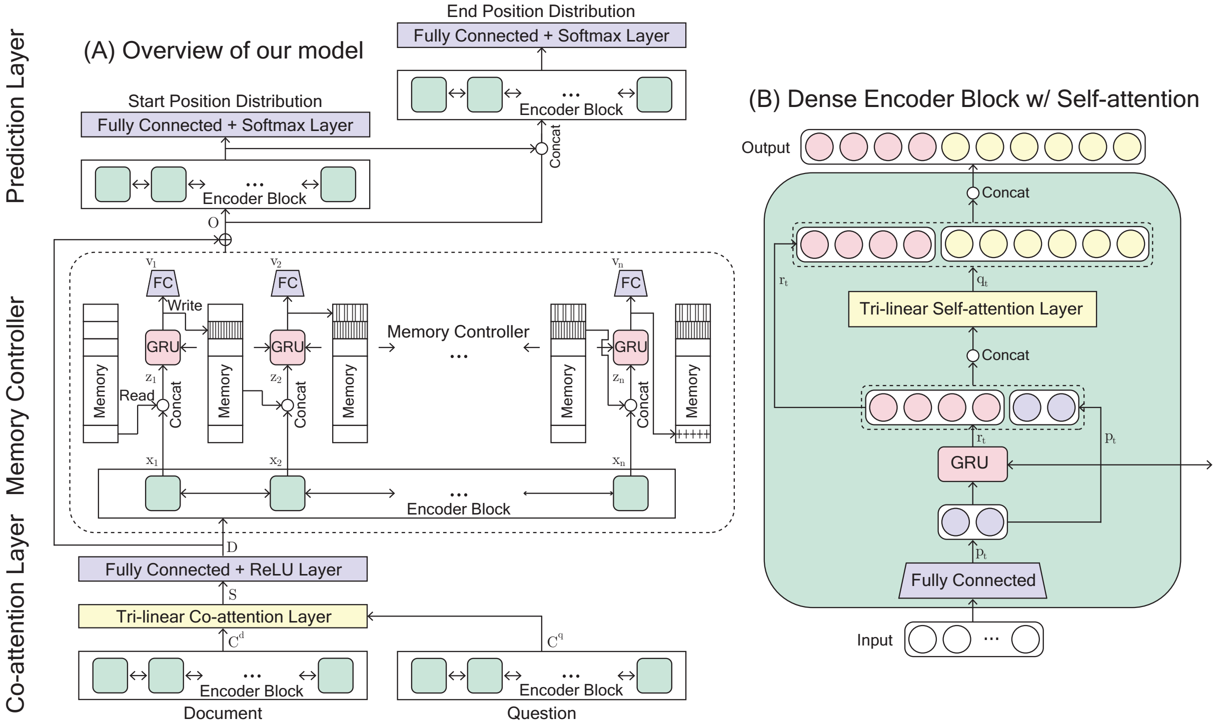

Figure 1: Overview of our model (A) and dense encoder block with self attention (B).

where $i, j = 1, \dots, n$. Finally, the self-attended representation $Q = \{\mathbf{q}_t\}_{t=1}^{n}$ is obtained by performing column-wise softmax on S^g to get the attention matrix A^g, which is further multiplied with $G = \{\mathbf{g}_t\}_{t=1}^{n}$, i.e.,

$$Q = A^g G \in \mathbb{R}^{n \times 3l}.$$

The final output is obtained as the concatenation of outputs from the recurrent layer (BiGRU) and the self-attention layer, i.e., $[\mathbf{r}_t; \mathbf{q}_t] \in \mathbb{R}^{5l}$.

3 Reading Comprehension Model with Proposed Components

We apply the proposed components to our model for RC tasks. As depicted in Figure 1, the model consists of three major layers: the co-attention layer, the memory controller, and the prediction layer. Given the embeddings of a question and a document, the co-attention layer generates query-aware contextual representations. The memory controller further refines these contextual representations using an external memory. Based on such representations, the prediction layer determines the start and the end token indices that form the answer span. In addition, we replace all the encoder block with DEBS in the three major layers.

Embedding. We incorporate both word- and character-level embedding methods to obtain the vector representation of each word in the input data. For word-level embedding $\mathbf{e}_w$, we utilize pre-trained, 300-dimensional embedding vectors from GloVe 6B (Pennington et al., 2014). The character-level word embedding $\mathbf{e}_c$ is obtained as a 100-dimensional vector by first applying a convolution layer with 100 filters to a sequence of 20-dimensional character embeddings learned during training and by further applying global max-pooling over the entire character-level sequence. Then we obtain the embedding vector of a given word token, $\mathbf{e}$, by concatenating these word- and character-level embeddings, i.e., $\mathbf{e} = [\mathbf{e}_w; \mathbf{e}_c] \in \mathbb{R}^{400}$.

Finally, we obtain the two sets of embedding vectors of question and document token sequences as $E^q = \{\mathbf{e}_u^q\}_{u=1}^{m} \in \mathbb{R}^{m \times 400}$ and $E^d = \{\mathbf{e}_t^d\}_{t=1}^{n} \in \mathbb{R}^{n \times 400}$, where m and n represent the sequence length of a question and a document, respectively.

Co-attention layer. Given E^q and E^d, we feed each of them into the encoder block and obtain their contextual representations as

$$C^q = \{\mathbf{c}_u^q\}_{u=1}^{m} = \text{EncoderBlock}^q(E^q) \in \mathbb{R}^{m \times k}$$
$$C^d = \{\mathbf{c}_t^d\}_{t=1}^{n} = \text{EncoderBlock}^d(E^d) \in \mathbb{R}^{n \times k}.$$

These representations are used to calculate the pairwise similarity matrix $S \in \mathbb{R}^{m \times n}$ between tokens in the question and those in the document by a tri-linear function (Seo et al., 2017), i.e.,

$$s_{ij} = \mathbf{w}_q \cdot \mathbf{c}_i^q + \mathbf{w}_d \cdot \mathbf{c}_j^d + \mathbf{w}_c \cdot (\mathbf{c}_i^q \odot \mathbf{c}_j^d),$$

where $i = 1, \ldots, m$, $j = (1, \ldots, n)$, and $\odot$ represents the element-wise multiplication and $\mathbf{w}_q$, $\mathbf{w}_d$, and $\mathbf{w}_c$ are trainable vectors. We apply column-wise softmax to S to obtain the document-to-question attention matrix A. Afterwards, a question-attended document representation $\tilde{C}^q$ is calculate as

$$\tilde{C}^q = \{\tilde{\mathbf{c}}_t^q\}_{t=1}^n = A^T C^q \in \mathbb{R}^{n \times k}.$$

In addition to this, we obtain vector $\tilde{\mathbf{a}} \in \mathbb{R}^n$, corresponding to the attention of a question to document tokens, by applying softmax to the column-wise max values of S. Then document-attended question vector is obtained by

$$\tilde{\mathbf{c}}^d = \sum_{t=1}^{n} \tilde{\mathbf{a}}_t \mathbf{c}_t^d \in \mathbb{R}^k.$$

The final co-attended representations $\{\mathbf{d}_t\}_{t=1}^n$ is obtained by fully connected layer with ReLU as a nonlinear unit, φ, as

$$\mathbf{d}_t = \varphi([\mathbf{c}_t^d; \tilde{\mathbf{c}}_t^q; \mathbf{c}_t^d \odot \tilde{\mathbf{c}}_t^q; \mathbf{c}_t^d \odot \tilde{\mathbf{c}}_t^d]) \in \mathbb{R}^l.$$

Memory controller. This layer takes the output of the co-attention layer $\{\mathbf{d}_t\}_{t=1}^n$ as input and refine their representations using our proposed memory controller (Section 2.1). Afterwards, the resulting output vector $\{\mathbf{o}_t\}_{t=1}^n$ are given as input to the prediction layer.

Prediction layer. We feed the output of the memory controller $\{\mathbf{o}_t\}_{t=1}^n$ to the prediction layer to predict the start and the end token indices of the answer span. First, it goes through the encoder block followed by the fully connected layer with softmax over the entire sequence to compute the probability distribution of a start index. The probability distribution of the end index is calculated by concatenating the output of the encoder block for the start index with the output of the memory controller and then by feeding them as input to another encoder block. These probability distributions are used as part of the negative log-likelihood objective function.

4 Experimental Setup

Datasets and preprocessing. We perform extensive experiments with well-known benchmarks such as TriviaQA, QUASAR-T, and SQuAD, as summarized in Table 1. In most of these datasets, a question q and a document d are represented as a sequence of words, and the answer span has to be selected from the document words based on the question. SQuAD consists of crowd-sourced questions and paragraphs from Wikipedia articles containing the answer to these questions. QUASAR-T is mostly based on factoid questions with their corresponding, large-sized corpus. TriviaQA is composed of question-answer pairs obtained from 14 trivia and quiz-league websites, along with the documents collected later that are likely to contain the answer from either web search or Wikipedia. In TriviaQA dataset, we truncate each document to 1,200 words. Even with such truncation, the average word count per document (AWC) of TriviaQA is approximately four times larger than that of SQuAD. In terms of the AWC, documents in TriviaQA, QUASAR-T, and SQuAD can be viewed as large-, medium-, and small-length documents, respectively.

Dataset	Total Train / Dev / Test	AWC
SQuAD	87,599 / 10,570 / UNK	142
QUASAR-T (Short)	25,465 / 2,043 / 2,068	221
QUASAR-T (Long)	26,318 / 2,129 / 2,102	392
TriviaQA (Web)	528,979 / 68,621 / 65,509	631
TriviaQA (Wikipedia)	110,648 / 14,229 / 13,661	955

Table 1: Statistics of datasets in terms of the average word count per document (AWC). In TriviaQA, AWC was calculated after truncating each document to 1,200 words.

In TriviaQA dataset, because a document is collected separately for an already collected question-answer pair, the document does not sometimes have the information to properly infer the answer to the question. In response, Clark and Gardner (2017) attempted to solve this problem by exposing both relevant and irrelevant paragraphs together separated based on TF-IDF scores. We follow this approach in TriviaQA. In QUASAR-T, we follow the same preprocessing steps done by Dhingra et al. (2017).

Implementation details. We use TensorFlow[1]

---

[1] http://www.tensorflow.org

Domain	Model	Full		Verified		AF
		EM	F1	EM	F1	
Web (AWC=631)	Our model (with DEBS)	**68.21**	**73.26**	**82.57**	**86.05**	
	Our model (without DEBS)	66.82	71.91	81.01	84.12	
	BiDAF + SA + SN (Clark and Gardner, 2017)	66.37	71.32	79.97	83.70	
	Reading Twice for NLU (Weissenborn, 2017)	50.56	56.73	63.20	67.97	✓
	M-Reader (Hu et al., 2017)	46.65	52.89	56.96	61.48	✓
	BiDAF + DNC	42.34	48.65	51.50	57.17	
	MEMEN (Pan et al., 2017)	44.25	48.34	53.27	57.64	✓
	BiDAF (Seo et al., 2017)	40.74	47.05	49.54	55.80	
Wikipedia (AWC=955)	Our model (with DEBS)	64.12	69.44	**71.75**	**76.91**	
	Our model (without DEBS)	**64.41**	**69.60**	70.21	75.49	
	BiDAF + SA + SN (Clark and Gardner, 2017)	63.99	68.93	67.98	72.88	
	QANet (Yu et al., 2018a)	51.10	56.60	53.30	59.20	
	Reading Twice for NLU (Weissenborn, 2017)	48.60	55.10	53.40	59.90	✓
	M-Reader (Hu et al., 2017)	46.94	52.85	54.45	59.46	✓
	BiDAF + DNC	42.57	48.30	46.23	51.61	
	MEMEN (Pan et al., 2017)	43.16	46.90	49.28	55.83	✓
	BiDAF (Seo et al., 2017)	40.32	45.91	44.86	50.71	

Table 2: Single model results on TriviaQA (Web and Wikipedia) dataset. All the results are gathered from their corresponding publications except for our models and 'BiDAF + DNC,' which we implemented on our own. 'Full' represents a complete dataset not guaranteed to contain relevant information to answer the question while 'Verified' corresponds to its subset annotated by humans so that the relevant information for the answer is guaranteed to exist. The last column indicates whether a model uses any additional feature augmentation (AF).

to build the model and Sonnet[2] to implement the memory interface. NLTK (Bird and Loper, 2004) is used for tokenizing words. In the memory controller, we use four read heads and one write head, and the memory size is set to 100×36, with all initialized as 0. The hidden vector dimension l is set to 200. We use AdaDelta (Zeiler, 2012) as an optimizer with a learning rate of 0.5. The batch size is set to 20 for TriviaQA (Joshi et al., 2017) and 30 for SQuAD (Rajpurkar et al., 2016) and QUASAR-T (Dhingra et al., 2017). We use an exponential moving average of weights with a decaying factor of 0.001. Our model does require more memory than existing methods, but a single GPU (e.g., M40 with 12GB memory) was enough to train model within a reasonable amount of time.

5 Quantitative Results

For our quantitative comparisons, we use BiDAF with self attention (Clark and Gardner, 2017) as a baseline, which maintains the best results published on both TriviaQA and SQuAD datasets. In TriviaQA and QUASAR-T dataset, we compare our model with BiDAF (Seo et al., 2017) as well as its variant called 'BiDAF + DNC,' which is augmented with an existing external memory architecture (Graves et al., 2016) just before the answer prediction layer in the BiDAF.

Overall, in lengthy-document cases such as TriviaQA and QUASAR-T, our model outperforms all the published results, as seen in Tables 2 and 3, while in the short-document case such as SQuAD, we mostly achieve the best results, as seen in Table 4. In the following, we present detailed analyses on each dataset.

TriviaQA. As shown in Table 2, our model, even without DEBS, outperforms the existing state-of-the-art method such as 'BiDAF + SA + SN' by a large margin in all the cases. Our model with DEBS, which replaces BiGRU encoder blocks, performs even better than that without it in all the cases except for the combination of the 'full' and 'Wikipedia' case, which involves documents containing no relevant information for the answer. Among those methods shown in Table 2, Reading Twice for NLU (Weissenborn, 2017) uses background knowledge from ConceptNet while both M-Reader (Hu et al., 2017) and MEMEN (Pan et al., 2017) use POS and NER in-

---

[2] https://github.com/deepmind/sonnet

Dataset	Model	Dev set		Test set	
		EM	F1	EM	F1
Short documents (AWC=221)	Our model (with DEBS)	**65.06**	**69.17**	**69.11**	**71.19**
	Our model (without DEBS)	64.87	68.88	68.13	70.32
	BiDAF + DNC	51.18	54.77	54.81	58.24
	BiDAF (Seo et al., 2017)	45.40	50.90	47.60	52.40
Long documents (AWC=392)	Our model (with DEBS)	**62.08**	**65.21**	**63.54**	**66.87**
	Our model (without DEBS)	60.05	63.23	63.44	65.19
	BiDAF + DNC	48.67	52.25	52.15	54.43
	BiDAF (Seo et al., 2017)	37.00	42.50	39.50	44.50

Table 3: Performance results on QUASAR-T dataset.

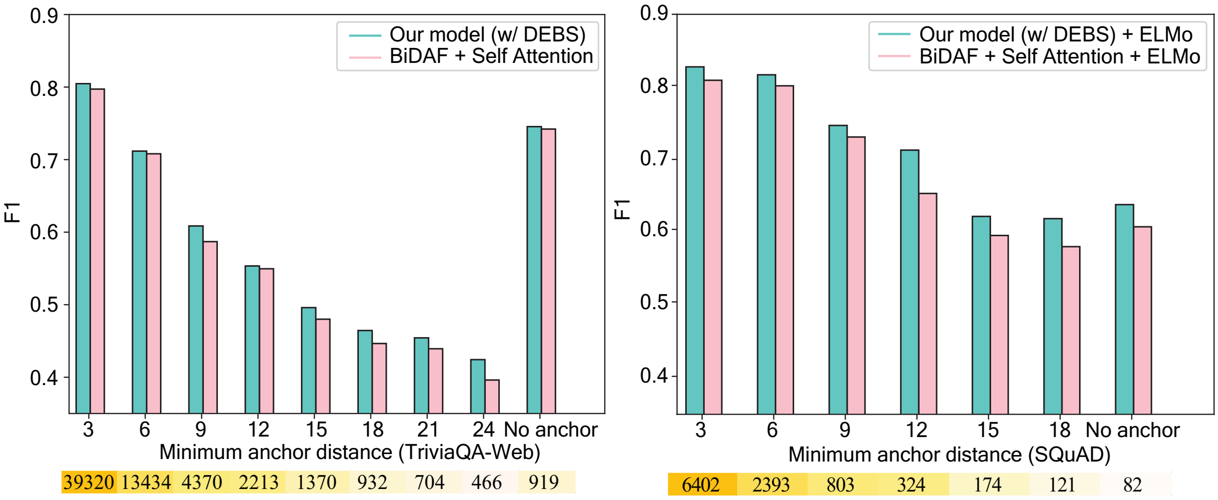

(a) TriviaQA (Web).

(b) SQuAD.

Figure 2: F1 score on the development set in TriviaQA (Web) and SQuAD with respect to the minimum anchor distance.

formation as additional features. We note that our method achieves these outstanding results without any additional features.

QUASAR-T. As shown in Table 3, our simple baseline 'BiDAF + DNC,' which involves an existing memory architecture, improves performance over BiDAF, indicating the efficacy of incorporating an external memory. Moreover, our model with the proposed memory controller achieves significantly better results compared to other models. Furthermore, another proposed component, DEBS, gives an additional performance boost to our model.

SQuAD. As shown in Table 4, most of the models, if not all, use additional features such as ELMo (Peters et al.), and the self-attention mechanism to further improve the performance. We also adopt these mechanisms one by one to show that our model can also benefit from these. First, we adopt ELMo to our model (without DEBS), which uses word embedding as the weighted sum of the hidden layers of a language model with regularization as an additional feature to our word embeddings. This improves the F1 score of our model up to 85.13 and EM to 77.44, showing the highest performances among all the methods without using self attention. Due to the relatively short document length in SQuAD compared to TriviaQA and QUASAR-T, our model without DEBS performs worse than the baseline 'BiDAF + Self Attention + ELMo.' However, after applying DEBS, our model outperforms the baseline, achieving 86.73 F1 and 79.69 EM.

Minimum anchor distance. Rajpurkar et al. (2016) proposed the difficulty measure called syntactic divergence, which is computed as the edit distance between syntactic parse trees of the question and the sentence containing the answer. However, this measure has limitations that the syntactic parser does not work properly on incomplete sentences, which are common in web text. It also becomes difficult to compute this measure if the

Model	Test set		AF	SA
	EM	F1		
Our model (with DEBS) + ELMo	**79.69**	**86.73**	✓	✓
BiDAF + Self Attention + ELMo (Peters et al.)	78.58	85.83	✓	✓
Our model (without DEBS) + ELMo	**77.44**	**85.13**	✓	
RaSoR + TR + LM (Salant and Berant, 2017)	77.58	84.16	✓	
QANet (Yu et al., 2018a)	76.24	84.60	✓	✓
SAN (Liu et al., 2017b)	76.83	84.40	✓	✓
FusionNet (Huang et al., 2017b)	75.97	83.90	✓	✓
RaSoR + TR (Salant and Berant, 2017)	75.79	83.26	✓	
Conducter-net (Liu et al., 2017a)	74.41	82.74	✓	✓
Reinforced Mnemonic Reader (Hu et al., 2017)	73.20	81.80	✓	✓
BiDAF + Self Attention (Clark and Gardner, 2017)	72.14	81.05		✓
MEMEN (Pan et al., 2017)	70.98	80.36	✓	
Our model (without DEBS)	**70.99**	**79.94**		
r-net (Wang et al., 2017)	71.30	79.70		✓
Document Reader (Chen et al., 2017)	70.73	79.35	✓	
FastQAExt (Weissenborn et al., 2017)	70.85	78.86		✓
Human Performance	82.30	91.22		

Table 4: Single model results on SQuAD. All the other results than ours are those reported in their own publications. The last two column indicate whether a model uses any additional feature augmentation (AF) and self attention (SA).

answer requires multi-sentence inference.

Instead, we develop our own metric called a minimum anchor distance, which is simple and robust to noisy text. To compute this metric, we first identify for all the co-occurring words (anchor words) between a document and a question while ignoring stop words. Then, we compute the number of words found between the answer and all the possible anchor words and select the minimum number from these.

In Figure 2, we show F1 scores of our model with DEBS and the baseline with respect to the minimum anchor distance. The scores are obtained from the development set of TriviaQA(Web) and SQuAD. The heat map at the bottom of the figure indicates the number of samples in each interval of the minimum anchor distance. One can see that our model performs increasingly better than the baseline as the minimum anchor distance gets larger. The examples shown in Table 5 indicate that documents with long dependencies tend to have a large minimum anchor distance. These examples show that our model predicts the remotely placed answer from the anchor word relatively well when anaphora resolution and negation are involved.

Ablation study with an encoder block. We assume that the concatenation of the layer outputs in DEBS helps the memory controller store contextual representations clearly. To see how DEBS affects the memory controller depending on different positions in the entire network, we conducted an ablation study by replacing the encoder block with DEBS on SQuAD. As can be seen in Table 6, using DEBS in all the places improves the performance most, and furthermore, the memory controller with DEBS gives the largest performance margin. This implies that DEBS can generally work as a better alternative to a BiGRU module, and DEBS is critical in maintaining the high performance of our proposed memory controller.

Adding our proposed modules to other models. To show the wide effectiveness of our proposed approaches, we choose two well-known baseline models in SQuAD: R-net (Wang et al., 2017) and 'BiDAF + Self Attention' (Clark and Gardner, 2017). These models have similar architectures where the model first pairs a given question and document pair using an attention and afterward applies a self-attention mechanism. We use the publicly available implementation of these models[3,4]. In Table 7, replacing all the recurrent units with DEBS and adding our memory controller between the question-document pairing

---

[3] https://github.com/HKUST-KnowComp/R-Net

[4] https://github.com/allenai/document-qa

Dataset	Example
TriviaQA (Web)	**Question** : What claimed the life of singer Kathleen Ferrier? **Context** : (omit) · · · Kathleen Ferrier (22.III.1912 Higher Walton, Lancashire- 8.X. 1953 London, England) was an English contralto singer* who achieved an international reputation with a repertoire extending from folksong and popular ballads to the classical works. Her death from *cancer*, at the height of her fame, was a shock to the musical world and particularly to the general public, which was kept in ignorance of · · · (omit)
SQuAD	**Question** : What did Mote think the Yuan class system really represented? **Context** : The historian Frederick W.Mote wrote that the usage of the term "social classes" for this system was misleading and that the position of people within the four -class system* was not an indication of their actual social power and wealth, but just entailed *"degrees of privilege"* to which they were entitled institutionally and legally, so a person's standing within the classes was not a guarantee of their standing, · · · (omit)

*A word with an asterisk indicates an anchor word closest to the ground truth answer.

Table 5: Examples in TriviaQA (Web) and SQuAD. *Italic* means the ground truth answer, frame indicates the prediction of our model (with DEBS) and underline shows the prediction of 'BiDAF + Self Attention' model.

Adding DEBS			Dev	
C	M	P	EM	F1
			77.22	85.01
✓			77.31	85.22
✓		✓	77.75	85.34
✓	✓		78.70	86.12
✓	✓	✓	**78.93**	**86.26**

Table 6: Ablation study of replacing an encoder block with DEBS in the co-attention layer (C), the memory controller (M), and the prediction layer (P) in SQuAD. ✓means that DEBS is used. Otherwise, BiGRU is used.

Base model	Adaptation	Dev	
		EM	F1
R-net[3]	-	70.71 (0.07)	79.48 (0.08)
	✓	**71.12** (0.12)	**79.99** (0.11)
BiDAF +SA[4]	-	71.61 (0.07)	80.78 (0.08)
	✓	**72.82** (0.15)	**81.33** (0.09)

Table 7: Effects of our proposed components added to R-net and 'BiDAF + Self Attention (SA)' on SQuAD. The values in parentheses represent the standard deviation from 6 runs. The first row of each base model indicates the result of the original methods. When adding the proposed component, DEBS is used in the place of all the recurrent layers while the memory controller is added between the co-attention and the self-attention layers.

layer and the self-attention layer increases the F1 score by around 0.5 compared to the baseline.

6 Related Work

Numerous neural network-based methods have been proposed, pushing the performance nearly up to a human level. Although slight differences exist, (Wang et al., 2017; Seo et al., 2017; Xiong et al., 2017) mostly leverage the question-document co-attention based on their pairwise similarity of word-level vector representations. These models currently work as the backbone architecture for many other models. Furthermore, Wang et al. (2017) suggest utilizing a self attention mechanism between tokens within a document to refine contextual representations.

Salant and Berant (2017); Chen et al. (2016); Pan et al. (2017); Weissenborn (2017); Peters et al. focus on augmenting feature representations in the word embedding layer to provide rich information. Salant and Berant (2017); Weissenborn (2017); Peters et al. extract and use additional features from other neural models trained for another task or external resources. Chen et al. (2016); Pan et al. (2017) utilize additional syntactic or semantic features through part-of-speech tagging or named-entity recognition, etc.

Enriching the input representation from pre-trained external models has been shown to be useful in improving RC task performances. Yu et al. (2018a) have also improved the performance by leveraging self attention for context encoding based on convolutional neural networks. Hu et al. (2017) refine the contextual representation with multiple hops, and Pan et al. (2017) use the encoded query for refining the answer prediction as a memory, which are different from our work in terms of handling long-range dependency.

7 Conclusion

This paper proposed two novel, crucial components for deep neural network-based RC tasks, (1) an advanced memory controller architecture and (2) a densely connected encoder block with self attention. We showed the effectiveness of these approaches in handling long-range dependencies using three benchmark RC datasets such as TriviaQA, QUASAR-T, and SQuAD. Our proposed modules are widely applicable to other models to improve their performance. Future work includes developing a scalable read/write accessing mechanism to handle a large-scale external memory to reason over multiple documents.

Acknowledgement

We thank to reviewers for helpful feedback. This work is the extended version of Yu et al. (2018b). This work is partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (No. NRF2016R1C1B2015924).

References

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proc. the ACL 2004 Interactive poster and demonstration sessions*, page 31.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL)*, volume 1, pages 2358–2367.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471.

Mikael Henaff, Antoine Bordes Jason Weston, Arthur Szlam, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *arXiv preprint arXiv:1705.02798*.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2017a. Densely connected convolutional networks. In *Proc. the IEEE conference on computer vision and pattern recognition (CVPR)*, volume 1, page 3.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2017b. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL)*, pages 1601–1611.

Rui Liu, Wei Wei, Weiguang Mao, and Maria Chikina. 2017a. Phase conductor on multi-layered attentions for machine comprehension. *arXiv preprint arXiv:1710.10504*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017b. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. the conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (NAACL-HLT)", year = "2018", pages = "2227–2237",.*

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proc. the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Shimi Salant and Jonathan Berant. 2017. Contextualized word representations for reading comprehension. *arXiv preprint arXiv:1712.03609*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proc. the International Conference on Learning Representations (ICLR)*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proc. the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL)*, volume 1, pages 189–198.

Dirk Weissenborn. 2017. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. In *Proc. the Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *Proc. the International Conference on Learning Representations (ICLR)*.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018a. Qanet: Combining local convolution with global self-attention for reading comprehension.

Seunghak Yu, Sathish Reddy Indurthi, Seohyun Back, and Haejun Lee. 2018b. A multi-stage memory augmented neural network for machine reading comprehension. In *Proc. the Workshop on Machine Reading for Question Answering*, pages 21–30. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Multi-Granular Sequence Encoding via Dilated Compositional Units for Reading Comprehension

Yi Tay[†], **Luu Anh Tuan**[ψ], **Siu Cheung Hui**[φ]
[†]ytay017@e.ntu.edu.sg
[ψ]at.luu@i2r.a-star.edu.sg
[φ]asschui@ntu.edu.sg
[†,φ]School of Computer Science and Engineering, Nanyang Technological University
[ψ]A*Star, Institute for Infocomm Research, Singapore

Abstract

Sequence encoders are crucial components in many neural architectures for learning to read and comprehend. This paper presents a new compositional encoder for reading comprehension (RC). Our proposed encoder is not only aimed at being fast but also expressive. Specifically, the key novelty behind our encoder is that it explicitly models across multiple granularities using a new dilated composition mechanism. In our approach, gating functions are learned by modeling relationships and reasoning over multi-granular sequence information, enabling compositional learning that is aware of both long and short term information. We conduct experiments on three RC datasets, showing that our proposed encoder demonstrates very promising results both as a standalone encoder as well as a complementary building block. Empirical results show that simple Bi-Attentive architectures augmented with our proposed encoder not only achieves state-of-the-art / highly competitive results but is also considerably faster than other published works.

1 Introduction

Teaching machines to read, comprehend and reason lives at the heart of reading comprehension (RC) tasks (Rajpurkar et al., 2016; Lai et al., 2017; Dunn et al., 2017; Kočiskỳ et al., 2017). In these tasks, the goal is to answer questions based on a given passage, effectively testing the learner's capability to understand natural language. This has been an extremely productive area of research in the recent years, giving rise to many highly advanced neural network architectures (Xiong et al., 2016; Weissenborn et al., 2017; Seo et al., 2016; Hu et al., 2017; Shen et al., 2017; Wang and Jiang, 2016b; Wang et al., 2018). A common denominator in many of these models is the compositional encoder, i.e., usually a bidirectional recurrent-based (LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014)) encoder that sequentially parses the text sequence word-by-word. This helps to model compositionality of words, capturing rich and complex linguistic and syntactic structure in language.

While the usage of recurrent encoder is often regarded as indispensable in highly complex RC tasks, there are still several challenges and problems pertaining to its usage in modern RC tasks. Firstly, documents can be extremely long to the point where running a BiRNN model across a long document is computationally prohibitive[1]. This is aggravated since RC tasks can be easily extended to reasoning over multiple long documents. Secondly, recurrent encoders have limited access to long term context since each word is sequentially parsed. This restricts any form of multi-sentence and intra-document reasoning from happening within compositional encoder layer.

To this end, we propose a new compositional encoder that can either be used in place of standard RNN encoders or serve as a new module that is complementary to existing neural architectures. Our proposed encoder leverages *dilated compositions* to model relationships across multiple granularities. That is, for a given word in the target sequence, our encoder exploits both long-term (far) and short-term (near) information to decide how much information to retain for it. Intuitively, this can be interpreted as learning to compose based on modeling relationships between word-level, phrase-level, sentence-level, paragraph-level and so on. The output of the dilated composition mechanism acts as gating functions, which are then used to learn compositional representations of the input sequence.

---

[1] Many recent works tackle this issue (Yu et al., 2018; Choi et al., 2017). However, this work presents a complementary/orthogonal approach to many of these works.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2141–2151
Brussels, Belgium, October 31 - November 4, 2018.

A brief high-level overview to our proposed encoder is given as follows: Firstly, sequences are chunked into blocks based on user-defined (hyperparameter) block sizes. Block sizes are often dilated in nature, i.e., $1, 2, 4, 10, 25$, etc., in order to capture more long-term information. Our encoder takes the neural bag-of-words representation of each block size and compresses/folds all words (that reside in each block) into a single summed embedding. All blocks are then passed into fully-connected layers and re-expanded/unfolded to their original sequence lengths. For each word, the gating vectors are then constructed by modeling the relationships between all blocks that this word resides in. As such, this can be interpreted as a divide-and-conquer sequence encoding method.

This has several advantages. Firstly, we enable a major speedup by avoiding either costly step-by-step gate construction while still maintaining interactions between neighboring words. As such, our model belongs to a class of architectures which is inspired by QRNNs (Bradbury et al., 2016) and SRUs (Lei and Zhang, 2017). The key difference is that our gates are not constructed by convolution layers but explicit block-based matching across multiple ranges, both long and short. Secondly, modeling at a long range (e.g., 25 or 50) enables our model to look further ahead as opposed to only one step forward. As such, the learned gates not only possess information about nearby words but also a larger overview of the context. This is in similar[2] spirit to self-attention, albeit occuring within the encoder. Thirdly, the final gates are formed by modeling relationships between multi-range projections (n-gram blocks), allowing for fine-grained intra-document relationships to be captured. The overall contributions of our work are as follows:

- We propose DCU (Dilated Compositional Units[3]), a new compositional encoder for both fast and expressive sequence encoding. We propose an overall architecture that utilizes DCU within a Bi-Attentive framework for both multiple choice and span prediction RC tasks. DCU can be used as a standalone (without RNNs) for fast reading and/or together with RNN models (i.e., DCU-LSTM) for more expressive reading.
- We conduct extensive experiments on three large-scale and challenging RC datasets - RACE (Lai et al., 2017), SearchQA (Dunn et al., 2017) and NarrativeQA (Kočiský et al., 2017). Our model is lightweight, fast and efficient, achieving state-of-the-art or highly competitive performance on three datasets.
- Despite its simplicity, our model outperforms highly complex models such as Dynamic Fusion Networks (DFN) (Xu et al., 2017) on RACE. While DFN takes approximately a week to train, spending at least several hours per epoch, our model converges in less than 12 hours with only $4-5$ minutes per epoch. Moreover, our model outperforms DFN by $2\% - 6\%$ on the RACE benchmark and other strong baselines such as the Gated Attention Reader by 10%. On RACE, we outperform DFN without any recurrent and convolution layers. Ablation studies show an improvement of up to 6% when using DCU over a LSTM/GRU encoder.

2 Dilated Compositional Units (DCU)

In this section, we describe our proposed DCU encoder.

2.1 Dilated Composition Mechanism

The inputs to the DCU encoder are (1) a sequence $\{w_1, w_2 \cdots w_\ell\}$, and (2) list of ranges $\{r_1, r_2 \cdots r_k\}$ where k is the number of times the fold/unfold operation is executed. The final output of the encoder is a sequence of vectors which retains the same dimensionality as its inputs. Figure 1 provides an illustration of the overall encoder architecture.

2.1.1 Fold Operation

This section describes the operation for each r_j. For each r_j and the input sequence, the fold operation performs the summation of every r_j word. This is essentially the NBOW (neural bag-of-words) representation. This reduces the overall document length to ℓ/r_j where each item in the sequence is the sum of every r_j word. Given the new sequence of ℓ/r_j tokens, we then pass each token into a single layered feed-forward neural network:

$$\bar{w_t} = \sigma_r(\mathbf{W}_a(w_t)) + \mathbf{b}_a \qquad (1)$$

---

[2] It is good to note that our approach explicitdly compares across blocks of multi-granularities while self-attention compares on a word-level basis.

[3] This model was originally known as MRU (Multi-Range Reasoning Units) and was published on ArXiv on March 2018.

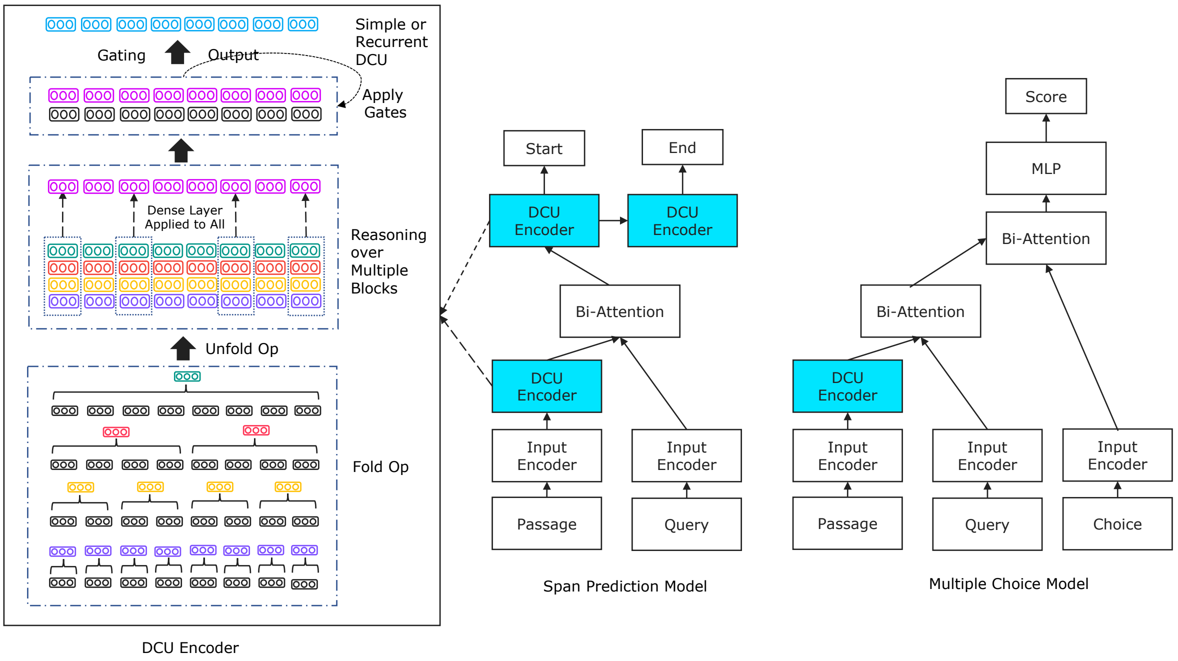

Figure 1: High-level overview of (1) our proposed DCU encoder (left), (2) Span Prediction Architecture (center) and (3) Multiple Choice Architecture (right). In the DCU encoder, blocks are formed at multi-granular levels. A block embedding is learned for each granularity. The composition gates for each word is constructed by modeling the relationships between all NBOW (neural bag-of-words) blocks that it resides in.

where $\mathbf{W}_a \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_a \in \mathbb{R}^d$ are the parameters of the fold layer. σ_r is the ReLU activation function. w_t is the t-th token in the sequence.

2.1.2 Unfold Operation

Given the transformed tokens $\bar{w}_1, \bar{w}_2 \cdots \bar{w}_{\ell/r_j}$, we then expand/unfold them into the original sequence length. Note that for each r_j, the parameters $\mathbf{W}_a, \mathbf{b}_a$ are not shared between blocks. Figure 2 depicts the fold-unfold operation for a single value of r_j.

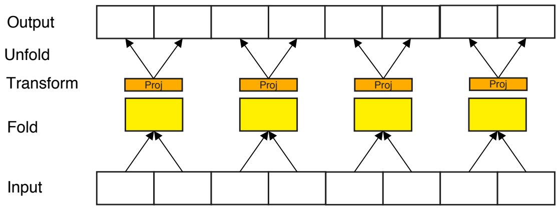

Figure 2: Overview of the Fold-Unfold operation for $r_j = 2$.

Overall, the key intuition of *each* fold-unfold operation is to learn representations of a block of a single granularity (say, blocks of 2). The main rationale for unfolding is to enable reasoning over multiple blocks (or granularities). This is described in the next section.

2.1.3 Multi-Granular Reasoning

From k different calls of the Fold/Unfold operation at different block sizes, we pass the concatenated vector of all transformed tokens into a two layered feed-forward neural network.

$$g_t = \mathbf{F}_2(\mathbf{F}_1([w_t^1; w_t^2; \cdots w_t^k])) \qquad (2)$$

where $\mathbf{F}_1(.), \mathbf{F}_2(.)$ are feed-forward networks with ReLU activations, i.e., $\sigma_r(W_x + b)$. $[;]$ is the concatenation operator. g_t is interpreted as a gating vector learned from multiple granularities and Equation (2) is learning the relationships between a token's representation at multiple hierarchies depending on the values of r_j. Notably, it is easy to see that every n pairs of words will have the same gating vector where n is the lowest value of r_j. As such, the value of the unigram, i.e., $r_j = 1$ (projection of every single token) is critical as it prevents identical gating vectors across the sequence.

2.2 Encoding Operation

To learn the DCU encoded representation of each word, we consider two variations of DCU encoders.

2.2.1 Simple Encoding

In this variation, we use g_t as a gating vector to control the fine-grained balance between the projection of each word w_t in the original input document and the original representation.

$$z_t = \tanh(\mathbf{W}_p\, w_t) + \mathbf{b}_p \quad (3)$$

$$y_t = \sigma(g_t) * w_t + (1 - \sigma(g_t))\, z_t \quad (4)$$

where $\{y_1, y_2, \cdots y_\ell\}$ is the output document representation. σ is the sigmoid function. Note that this formulation is in similar spirit to highway networks (Srivastava et al., 2015). However, since our gating function is learned via reasoning over multi-granular sequence blocks, it captures more compositionality and long range context. Note that an optional and additional projection may be applied to w_t but we found that it did not yield much empirical benefit.

2.2.2 Recurrent Encoding (DCU cell)

In the second variation, we consider a recurrent (sequential) variant. This is in similar spirit to QRNNs (Bradbury et al., 2016) and SRUs (Lei and Zhang, 2017) which reduces computation cost by pre-learning the gating vectors. The following operations describe the operations of the recurrent DCU cell for each time step t.

$$c_t = g_t \odot c_{t-1} + (1 - g_t) \odot z_t \quad (5)$$

$$h_t = o_t \odot c_t \quad (6)$$

where c_t, h_t are the cell and hidden states at time step t. g_t are the gates learned from the output of the multi-range reasoning step. o_t is an additional output gate learned via applying an affine transform on the input vector w_t, i.e., $o_t = W_o(w_t) + b_o$. Similar to RNNs, the Recurrent DCU parses the input sequence word-by-word. However, the cost is significantly reduced because we do not have expensive matrix operations that are executed in an non-parallel fashion. Finally, the outputs of the DCU encoder are a series of hidden vectors $\{h_1, h_2 \cdots h_\ell\}$ for each word in the sequence.

3 Overall Model Architectures

This section describes the overall model architectures that utilize DCU encoders. In our experiments, we focus on both multiple-choice based (RACE) and span prediction RC tasks (SearchQA, NarrativeQA). Since the core focus of this paper is our encoder, we briefly provide the high-level details[4] of our vanilla Bi-Attentive model. The Bi-Attentive models that are used in our experiments act as baselines, often being less complex than current competitive models such as BiDAF (Seo et al., 2016), AMANDA (Kundu and Ng, 2018) or DFN (Xu et al., 2017). Figure 1 (center and right side of the figure) provides a high-level illustration of these architectures.

3.1 Multiple Choice Models

In the Multiple Choice (MCQ) model, there are three types of input sequences, namely Passage (P), Question (Q) and Answers (A_j). The output of the model (for each answer), is a score $s(P, Q, A_j) \in [0, 1]$ denoting the strength of A_j.

Input Encoding Each input sequence is passed into first a projection layer. To enhance the input word representations, we also include the standard EM (exact match) binary feature to each word. In this case, we use a three-way EM adaptation, i.e., $EM(P, Q), EM(Q, A)$ and $EM(P, A)$. The projected embeddings are then passed into a single-layered highway network.

Compositional Encoder In our experiments, we vary the encoder in this layer. Typical choices of encoders in this layer are LSTMs or GRUs. We vary this in our experiments in order to benchmark the effectiveness of our proposed DCU encoder. The output of this layer has the same dimensions as its inputs (typically the hidden states of a RNN model).

Bi-Attention Layer - This layer models the interactions between P, Q and A. Let $B(.)$ be a standard bidirectional attention that utilizes mean-pooling aggregation. The scoring function is the bilinear product of the nonlinearly transformed input, i.e., $F(x)_i^\top \mathbf{M} F(y)_i$. We first apply $B(P, Q)$ to form bi-attentive P^q, Q^p representations. Subsequently, we apply $B(P^q, A_j)$ to learn a vector representation for each answer. A temporal sum pooling is applied on the outputs of P^{qa}, A_j^p and concatenated to form $a_j^f \in \mathbb{R}^{2d}$.

Answer Selection Let $\{a_1, a_2 \cdots a_{N_a}\}$ be the inputs to this layer and N_a is the number of answer candidates. Motivated by the work in retrieval-based QA (Severyn and Moschitti, 2015), we include word overlap features to each answer candidate. This word overlap feature is in similar spirit to the EM feature. Each overlap operation between two sequences returns four features. We convert each answer vector a_j into a scalar via

---

[4] This is primarily due to the lack of space as the main focus of this work is the DCU. Source code will be released at `hhttps://github.com/vanzytay/EMNLP18_DCU`.

$a_j^f = Softmax(\mathbf{W}_2(\sigma_r(\mathbf{W}_1([a_j]) + b_1) + b_2))$ where $\mathbf{W}_*, b_*$ and $* = \{1, 2\}$ are standard dense layer parameters.

Optimization The MCQ-based model minimizes the multi-class cross entropy where the number of classes corresponds to the number of choices.

3.2 Span Prediction Model

In the Span Prediction Model, the goal is to extract (or predict a span s, e) where $P[s : e]$ is the answer to the query. As such, the key interaction in this architecture is between P and Q. For most part, the model architecture remains similar especially for the input encoding layers and compositional encoder layer. The key difference is that we reduce the number of input sequence from three to two.

Input Encoding This follows the same design as the MCQ-based model, albeit for two sequences instead of three. Similarly, the two-way EM feature is added before passing into the highway layer.

Compositional Encoder This layer remains identical to the MCQ-based model.

Bi-Attention Layer We adopt a different bi-attention function for span prediction. More specifically, we use the 'SubMultNN' or the 'Mult' adaptation from (Wang and Jiang, 2016a) (which is tuned) and compare aligned sequences between P and Q to form P^q, the query-dependent passage representation.

Answer Pointer Layer In this layer, we pass P^q through a two layered compositional encoder (which is varied similar to the compositional encoder layer and will be further elaborated in our experiments.). The start pointer and end pointer are determined by $F(H_1), F(H_2)$ where H_1, H_2 are the hidden outputs from the first and second encoders respectively. $F(.)$ is a linear transform, projecting each hidden state to a scalar. We pass both of them into softmax functions to obtain probability distributions.

Optimization Following (Seo et al., 2016; Wang and Jiang, 2016b), we minimize the joint cross entropy loss of the start and end probability distributions. During inference, we follow (Wang and Jiang, 2016b) to find the best answer span.

4 Empirical Evaluation

In this section, we report our experimental results and comparisons against other published works.

4.1 Datasets

For our experiments, we use one challenging multiple choice MC dataset and two span-prediction MC datasets.

RACE (Reading Comprehension from Examinations) (Lai et al., 2017) is a recently proposed dataset that is constructed from real world examinations. Given a passage, there are several questions with four options each. The authors argue that RACE is more challenging compared to popular benchmarks (e.g., SQuAD (Rajpurkar et al., 2016)) as more multi-sentence and compositional reasoning are required. There are two subsets of RACE, namely RACE-M (Middle school) and RACE-H (High school). The latter is considered to be harder than the former.

SearchQA (Dunn et al., 2017) is a recent dataset that emulates a real world QA system. It involves extracting passages from search engine results and requiring models to answer questions by reasoning and reading these search snippets.

NarrativeQA (Kočiskỳ et al., 2017) is a recent benchmark proposed for story-based reading comprehension. Different from many RC datasets, the answers are handwritten by human annotators. We focus on the *summaries* setting instead of reading full stories since our model is targetted at standard RC tasks.

MCQ datasets are evaluated using the standard accuracy metric. For RACE, we train models on the entire dataset, i.e., both RACE-M and RACE-H, and evaluate them separately. For RACE, the model selection is based on each subset's respective development set. For SearchQA, we follow (Kundu and Ng, 2018; Dunn et al., 2017) which evaluates unigram exact match (EM) and n-gram F1 scores. For NarrativeQA, since the answers are human written and not constrained to a span that can be found in the passage, the evaluation metrics are BLEU-1, BLEU-4, Meteor and Rouge-L following (Kočiskỳ et al., 2017).

4.2 Competitor Methods

We describe the key competitors on each dataset.

RACE The key competitors are the Stanford Attention Reader (Stanford AR) (Chen et al., 2016), Gated Attention Reader (GA) (Dhingra et al., 2016), and Dynamic Fusion Networks (DFN) (Xu et al., 2017). GA incorporates a multi-hop attention mechanism that helps to refine the answer representations. DFN is an extremely complex model. It uses (1) BiMPM's matching functions (Wang et al., 2017c) for extensive matching between Q, P and A, (2) multi-hop reasoning powered by ReasoNet (Shen et al., 2017) and (3) employs reinforcement learning techniques for dynamic strategy selection. A leaderboard for this dataset is maintained at `http://www.qizhexie.com/data/RACE_leaderboard`. Note that the current state-of-the-art[5] (Radford et al., 2018), is a generative pre-training model trained on a large external corpus.

SearchQA The main competitor baseline is the AMANDA model proposed by (Kundu and Ng, 2018). AMANDA uses a multi-factor self-attention module, along with a question focused span prediction. AMANDA also uses BiLSTM layers for input encoding and at the span prediction layers. We also compare against the reported ASR (Kadlec et al., 2016) baselines which were reported in (Dunn et al., 2017).

NarrativeQA On this benchmark, we compare with the reported baselines in (Kočiskỳ et al., 2017). We compete on the summaries setting, in which the baselines are a context-less sequence to sequence (seq2seq) model, ASR (Kadlec et al., 2016) and BiDAF (Seo et al., 2016). As per reviewer request, we also benchmark a stronger competitor, namely R-NET (Wang et al., 2017b) on this benchmark. We use the open source implementation[6] at `https://github.com/HKUST-KnowComp/R-Net`.

4.3 Our Methods

Across our experiments, we benchmark several variants of our proposed DCU. The first is denoted as Sim-DCU which corresponds to the Simple DCU model described earlier. The model denoted by DCU (without any prefix) corresponds to the recurrent DCU model. Finally, the final variant is the DCU-LSTM which places a DCU encoder layer on top of a BiLSTM layer. We report the dimensions of the encoder as well as training time (per epoch) for each variant. The encompassing framework for DCU is the Bi-Attentive models described for MCQ-based problems and span prediction problems. Unless stated otherwise, the encoder in the pointer layer for span prediction models also uses DCU. However, for the Hybrid DCU-LSTM models, answer pointer layers use BiLSTMs. For the RACE-dataset, we additionally report scores of an ensemble of nine Sim-DCU models. This is to facilitate comparison against ensemble models of (Xu et al., 2017). We tune the dimensionality of the DCU cell within a range of $100-300$ in denominations of 50. The results reported are the best performing models on the held-out set.

4.4 Implementation Details

We implement all models in TensorFlow (Abadi et al., 2015). Word embeddings are initialized with $300d$ GloVe (Pennington et al., 2014) vectors and are not fine-tuned during training. Dropout rate is tuned amongst $\{0.1, 0.2, 0.3\}$ on all layers including the embedding layer. For our DCU model, we use range values of $\{1, 2, 4, 10, 25\}$. DCU encoders are only applied on the passage and not the query. We adopt the Adam optimizer (Kingma and Ba, 2014) with a learning rate of $0.0003/0.001/0.001$ for RACE/SearchQA/NarrativeQA respectively. The batch size is set to $64/256/32$ accordingly. The maximum sequence lengths are $500/200/1100$ respectively. For NarrativeQA, we use the Rouge-L score to find the best approximate answer relative to the human written answer for training the span model. All models are trained and all runtime benchmarks are based on a TitanXP GPU.

4.5 Experimental Results on RACE

Table 1 reports our results on the RACE benchmark dataset. Our proposed DCU model achieves the best result for both single models and ensemble models. We outperform highly complex models such as DFN. We also pull ahead of other recent baselines such as ElimiNet (Parikh et al., 2018) and GA by at least 5%. The best single model score from RACE-H and RACE-M alternates between Sim-DCU and DCU. Overall, there is a 6% improvement on the RACE-H dataset and

---

[5] This paper was not public at the time of EMNLP 2018 submission.

[6] Note that the authors of this repository state some differences with the original R-NET. However, they get similar scores on the SQuAD benchmark.

Model	RACE-M	RACE-H	RACE	Time
Sliding Window (Lai et al., 2017)	37.3	30.4	32.2	N/A
Stanford AR (Chen et al., 2016)	44.2	43.0	43.3	N/A
GA (Dhingra et al., 2016)	43.7	44.2	44.1	N/A
ElimiNet (Parikh et al., 2018)	N/A	N/A	44.5	N/A
Dynamic Fusion Network (Xu et al., 2017)	51.5	45.7	47.4	≈8 hours (1 week*)
Bi-Attention (No Encoder)	50.6	44.0	44.9	3 min (9 hours)
Bi-Attention (250d GRU)	48.5	42.1	44.0	16 min (2 days)
Bi-Attention (250d LSTM)	50.3	40.9	43.6	18 min (2 days)
Bi-Attention (250d Sim-DCU)	**57.7**	47.4	**50.4**	4 min (12 hours)
Bi-Attention (250d DCU)	56.1	**47.5**	50.0	12 min (20 hours)
GA + ElimiNet (Parikh et al., 2018)	N/A	N/A	47.2	N/A
DFN Ensemble (x9) (Xu et al., 2017)	55.6	49.4	51.2	N/A
Bi-Attention (Sim-DCU) Ensemble (x9)	**60.2**	**50.3**	**53.3**	N/A

Table 1: Comparison against other published models on RACE dataset (Lai et al., 2017). Competitor result are reported from (Lai et al., 2017; Xu et al., 2017). Best result for each category (single and ensemble) is in boldface. Last column reports estimated training time per epoch and total time for convergence. * is an estimated value that we obtain from asking the authors.

	Dev		Test		
Model	Acc	F1	Acc	F1	Time
TF-IDF max (Dunn et al., 2017)	13.0	N/A	12.7	N/A	N/A
ASR (Kadlec et al., 2016)	43.9	24.2	41.3	22.8	N/A
AMANDA (Kundu and Ng, 2018)	48.6	57.7	46.8	56.6	≈8* min
Bi-Attention† (No Encoder)	12.4	20.2	18.9	12.3	≈17 sec
Bi-Attention† (150d BiLSTM)	40.0	51.3	38.6	49.0	≈7 min
Bi-Attention† (300d LSTM)	40.3	48.7	38.2	46.4	≈6 min
Bi-Attention† (300d Sim-DCU)	44.1	45.5	42.9	43.1	≈25 sec
Bi-Attention† (300d DCU)	48.6	54.8	46.8	53.3	≈2 min
Bi-Attention (200d Hybrid DCU-LSTM)	**50.5**	**59.9**	**49.4**	**59.5**	≈7 min

Table 2: Experimental Results on SearchQA dataset. (Dunn et al., 2017). Unigram Accuracy and N-gram F1 are reported following (Kundu and Ng, 2018). All models with † use the same encoder in the answer pointer layer. * is an estimate running a replicated model with same batch size ($b = 256$) as our models.

1.8% improvement on the RACE-M dataset. Our Sim-DCU model also runs at 4 minutes per iteration, which is dramatically faster and simpler than DFN or other recurrent models. We believe that this finding highlights the importance of designing strong and fast baselines for the task at hand.

In general, we also found that the usage of a recurrent cell is not really crucial on this dataset since (1) Sim-DCU and DCU can achieve comparable performance to each other, (2) GRU and LSTM models do not have a competitive edge and (3) using no encoder can achieve comparable[7] performance to DFN. Finally, an ensemble of Sim-DCU models achieve state-of-the-art performance on the RACE dataset, achieving an overall score of 53.3%.

4.6 Experimental Results on SearchQA

Table 2 reports our results on the SearchQA dataset. We draw the reader's attention to the performance of the 300d DCU encoder. We achieve the same accuracy as AMANDA without using any LSTM or GRU encoder. This model runs at 2 minutes per epoch, making it 4 times more efficient than AMANDA (estimated, with identical batch size). While AMANDA also uses multi-factor self-attention, along with character enhanced representations, our simple DCU encoder used within a mere baseline bi-attentive framework comes close in performance. Finally, the hybrid combination, DCU-LSTM significantly outperforms AMANDA by 3%.

Contrary to MCQ-based datasets, we found that

---

[7] Nevertheless, this suggests the importance of benchmarking good and strong baselines since a well-tuned baseline model can outperform DFN, a highly complicated model.

Model	BLEU-1	BLEU-4	Meteor	Rouge-L	Time
Seq2Seq†	15.89	1.26	4.08	13.15	N/A
ASR† (Kadlec et al., 2016)	23.20	6.39	7.77	22.26	N/A
BiDAF† (Seo et al., 2016)	33.72	15.53	15.38	36.30	N/A
R-NET$^\phi$ (Wang et al., 2017b)	34.90	20.30	18.00	36.70	N/A
Bi-Attention (300d LSTM)	31.18	15.34	14.42	32.95	≈1 hour
Bi-Attention (150d BiLSTM)	34.22	18.22	16.19	38.32	≈1 hour
Bi-Attention (300d Sim-DCU)	9.15	1.69	3.95	11.16	1 min
Bi-Attention (300d DCU)	33.28	16.15	15.84	36.65	18 mins
Bi-Attention (150d Hybrid DCU-LSTM)	**36.55**	19.79	17.87	**41.44**	≈1 hour

Table 3: Experimental Results on the NarrativeQA reading comprehension challenge (Kočiský et al., 2017) using summaries. † are baselines reported by (Kočiský et al., 2017). $^\phi$ was obtained by running an open-source implementation of R-NET on the benchmark.

Sim-DCU model could not achieve comparable results to the recurrent DCU. We hypothesize that this is due to the need to predict spans. Nevertheless, the 300d DCU outperforms an LSTM encoder and remains competitive to a BiLSTM of similar dimensionality[8]. We also observe that LSTM and DCU are complementary. This shows that stacking a DCU encoder over standard LSTMs can give a performance boost relative to using each encoder separately.

4.7 Experimental Results on NarrativeQA

Table 3 reports our results on the NarrativeQA benchmark. First, we observe that 300d DCU can achieve comparable performance with BiDAF (Seo et al., 2016). When compared with a BiLSTM of equal output dimensions (150d), we find that our DCU model performs competitively, with less than 1% deprovement across all metrics. However, the time cost required is significantly reduced. The performance of our model is significantly better than 300d LSTM model while also being significantly faster. Here, we note that Sim-DCU does not produce reasonable results at all, which seems to be in similar vein to results on SearchQA, i.e., a recursive cell that processes word-by-word is mandatory for span prediction. However, our results show that it is not necessary to construct gates in a word-by-word fashion. Finally, DCU-LSTM significantly outperforms all models in terms of ROUGE-L, including BiDAF on this dataset. Performance improvement over the vanilla BiLSTM model ranges from $1\% - 3\%$ across all metrics, suggesting that DCU encoders are also effective as a complementary neural building block.

5 Related Work

A diverse collection of MC datasets such as SQuAD (Rajpurkar et al., 2016) and CNN/DailyMail (Hermann et al., 2015) are readily available for benchmarking new deep learning models. New datasets have been recently released (Kočiský et al., 2017; Joshi et al., 2017; Lai et al., 2017; Welbl et al., 2017; Dhingra et al., 2017; Trischler et al., 2016), claiming to involve a greater need for going beyond simple surface-level matching. As such, these datasets often emphasize the extent of compositional and multi-sentence reasoning required to tackle its questions.

In recent years, a wide range of innovative neural solutions have also been proposed, mainly involving bi-attention (Seo et al., 2016; Xiong et al., 2016; Cui et al., 2016) and answer pointers (Wang and Jiang, 2016b). Recent work also investigates the notion of multi-hop reasoning (Dhingra et al., 2016; Shen et al., 2017; Xu et al., 2017), reinforcement learning (Shen et al., 2017; Wang et al., 2017a; Hu et al., 2017), pretraining on auxilliary tasks (Radford et al., 2018; Peters et al., 2018; McCann et al., 2017, 2018) and self-matching / self-attention (Kundu and Ng, 2018; Wang et al., 2017b; Yu et al., 2018). While many of these works use BiLSTMs as standard building blocks, (Yu et al., 2018) proposed a RNN-less model architecture by utilizing components inspired by the Transformer architecture (Vaswani et al., 2017).

Our work is mainly concerned with designing an efficient encoder that is able to capture not only compositional information but also long-range and

---

[8] In terms of representation and parameter size, we consider a 150d BiLSTM to be equivalent to a 300d LSTM for fair comparison.

short-range information. More specifically, our recurrent DCU encoder takes on a similar architecture to Quasi-Recurrent Neural Networks (Bradbury et al., 2016) and Simple Recurrent Units (Lei and Zhang, 2017). In these models, gates are pre-learned and then applied. However, different from existing models such as QRNNs that use convolution layers as gates, we use block-based fold-unfold layers for learning gates. Our model also draws inspiration from dilation, in particular dilated RNNs (Chang et al., 2017) and dilated convolutions (Kalchbrenner et al., 2016), that intuitively help to model long-range dependencies. Notably, our work is orthogonal to recent advances that are targetted at speeding up the reading process. Such works include residual dilated convolutions (Wu et al., 2017), self-attention (Yu et al., 2018) and coarse-to-fine grained paradigm (Choi et al., 2017). However, while speed is one of the clear benefits of this work, our work is the first to introduce the idea of block-based multi-granular reasoning. We believe that this new building block is complementary/useful to the RC task in general.

6 Conclusion and Future Work

We proposed a novel neural architecture, the DCU encoder and an overall bi-attentive model for both MCQ-based and span prediction MC tasks. We apply it to three MC datasets and achieve competitive performance on all without the use of recurrent and convolution layers. Our proposed method outperforms DFN, an extremely complex model, without using any LSTM/GRU layer. We also remain competitive to AMANDA and BiDAF without any LSTM/GRU. While our proposed encoder demonstrates promise on reasoning and understanding natural language, we believe that our encoder is generalizable to other domains beyond reading comprehension. However, we defer this prospect to future work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. http://tensorflow.org/.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *CoRR* abs/1611.01576.

Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. 2017. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 76–86.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858* .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 209–220.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423* .

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* .

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904* .

Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179* .

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

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

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic reader for machine comprehension. *arXiv preprint arXiv:1705.02798* .

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551* .

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547* .

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* .

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *arXiv preprint arXiv:1712.07040* .

Souvik Kundu and Hwee Tou Ng. 2018. A question-focused multi-factor attention network for question answering .

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683* .

Tao Lei and Yu Zhang. 2017. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755* .

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*. pages 6297–6308.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730* .

Soham Parikh, Ananya Sai, Preksha Nema, and Mitesh M Khapra. 2018. Eliminet: A model for eliminating options for reading comprehension with multiple choice questions. https://openreview.net/forum?id=B1bgpzZAZ.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training .

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* .

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. pages 373–382.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 1047–1055.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR* abs/1505.00387.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830* .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 6000–6010.

Shuohang Wang and Jing Jiang. 2016a. A compare-aggregate model for matching text sequences. *CoRR* abs/1611.01747.

Shuohang Wang and Jing Jiang. 2016b. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* .

Shuohang Wang, Mo Yu, Shiyu Chang, and Jing Jiang. 2018. A co-matching model for multi-choice reading comprehension. *arXiv preprint arXiv:1806.04068* .

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2017a. R3: Reinforced reader-ranker for open-domain question answering. *arXiv preprint arXiv:1709.00023* .

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017b. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 189–198.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017c. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. pages 4144–4150.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. *arXiv preprint arXiv:1703.04816* .

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2017. Constructing datasets for multi-hop reading comprehension across documents. *arXiv preprint arXiv:1710.06481* .

Felix Wu, Ni Lao, John Blitzer, Guandao Yang, and Kilian Weinberger. 2017. Fast reading comprehension with convnets. *arXiv preprint arXiv:1711.04352* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR* abs/1611.01604.

Yichong Xu, Jingjing Liu, Jianfeng Gao, Yelong Shen, and Xiaodong Liu. 2017. Towards human-level machine reading comprehension: Reasoning and inference with multiple strategies. *arXiv preprint arXiv:1711.04964* .

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*. https://openreview.net/forum?id=B14TlG-RW.

Neural Compositional Denotational Semantics for Question Answering

Nitish Gupta[*]
University of Pennsylvania
Philadelphia, PA
nitishg@seas.upenn.edu

Mike Lewis
Facebook AI Research
Seattle, WA
mikelewis@fb.com

Abstract

Answering compositional questions requiring multi-step reasoning is challenging. We introduce an end-to-end differentiable model for interpreting questions about a knowledge graph (KG), which is inspired by formal approaches to semantics. Each span of text is represented by a denotation in a KG and a vector that captures ungrounded aspects of meaning. Learned composition modules recursively combine constituent spans, culminating in a grounding for the complete sentence which answers the question. For example, to interpret "*not green*", the model represents "*green*" as a set of KG entities and "*not*" as a trainable ungrounded vector—and then uses this vector to parameterize a composition function that performs a complement operation. For each sentence, we build a parse chart subsuming all possible parses, allowing the model to jointly learn both the composition operators and output structure by gradient descent from end-task supervision. The model learns a variety of challenging semantic operators, such as quantifiers, disjunctions and composed relations, and infers latent syntactic structure. It also generalizes well to longer questions than seen in its training data, in contrast to RNN, its tree-based variants, and semantic parsing baselines.

1 Introduction

Compositionality is a mechanism by which the meanings of complex expressions are systematically determined from the meanings of their parts, and has been widely assumed in the study of both artificial and natural languages (Montague, 1973) as a means for allowing speakers to generalize to understanding an infinite number of sentences. Popular neural network approaches to question answering use a restricted form of compositionality, typically encoding a sentence word-by-word, and then executing the complete sentence encoding against a knowledge source (Perez et al., 2017). Such models can fail to generalize from training data in surprising ways. Inspired by linguistic theories of compositional semantics, we instead build a latent tree of interpretable expressions over a sentence, recursively combining constituents using a small set of neural modules. Our model outperforms RNN encoders, particularly when test questions are longer than training questions.

Our approach resembles Montague semantics, in which a tree of interpretable expressions is built over the sentence, with nodes combined by a small set of composition functions. However, both the structure of the sentence and the composition functions are learned by end-to-end gradient descent. To achieve this, we define the parametric form of small set of composition modules, and then build a parse chart over each sentence subsuming all possible trees. Each node in the chart represents a span of text with a distribution over groundings (in terms of booleans and knowledge base nodes and edges), as well as a vector representing aspects of the meaning that have not yet been grounded. The representation for a node is built by taking a weighted sum over different ways of building the node (similar to Maillard et al. (2017)). The trees induced by our model are linguistically plausible, in contrast to prior work on structure learning from semantic objectives (Williams et al., 2018).

Typical neural approaches to grounded question answering first encode a question with a recurrent neural network (RNN), and then evaluate the encoding against an encoding of the knowledge source (for example, a knowledge graph or image) (Santoro et al., 2017). In contrast to classical approaches to compositionality, constituents of complex expressions are not given explicit interpretations in isolation. For example, in *Which cubes are large or green?*, an RNN encoder will not explic-

---

[*] Work done while interning with Facebook AI Research.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2152–2161
Brussels, Belgium, October 31 - November 4, 2018.

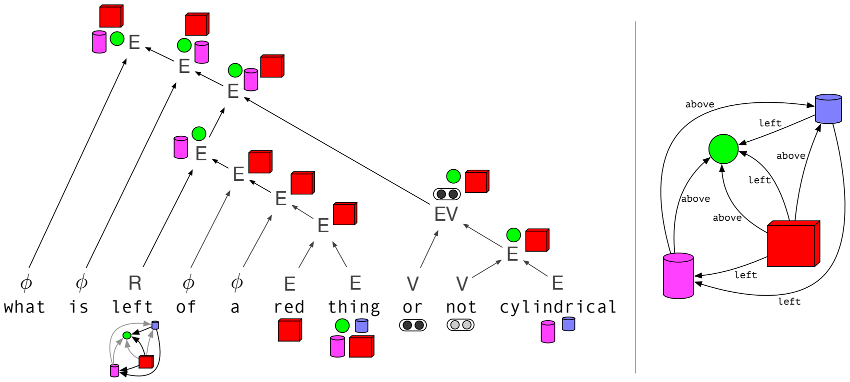

Figure 1: A correct parse for a question given the knowledge graph on the right, using our model. We show the type for each node, and its denotation in terms of the knowledge graph. The words *or* and *not* are represented by vectors, which parameterize composition modules. The denotation for the complete question represents the answer to the question. Nodes here have types E for sets of entities, R for relations, V for ungrounded vectors, EV for a combination of entities and a vector, and ϕ for semantically vacuous nodes. While we show only one parse tree here, our model builds a parse chart subsuming all trees.

itly build an interpretation for the phrase *large or green*. We show that such approaches can generalize poorly when tested on more complex sentences than they were trained on. Our approach instead imposes independence assumptions that give a linguistically motivated inductive bias. In particular, it enforces that phrases are interpreted independently of surrounding words, allowing the model to generalize naturally to interpreting phrases in different contexts. In our model, *large or green* will be represented as a particular set of entities in a knowledge graph, and be intersected with the set of entities represented by the *cubes* node.

Another perspective on our work is as a method for learning layouts of Neural Module Networks (NMNs) (Andreas et al., 2016b). Work on NMNs has focused on construction of the structure of the network, variously using rules, parsers and reinforcement learning (Andreas et al., 2016a; Hu et al., 2017). Our end-to-end differentiable model jointly learns structures and modules by gradient descent.

Our model is a new combination of classical and neural methods, which maintains the interpretability and generalization behaviour of semantic parsing, while being end-to-end differentiable.

2 Model Overview

Our task is to answer a question $q = w_{1..|q|}$, with respect to a Knowledge Graph (KG) consisting of nodes $\mathcal{E}$ (representing entities) and labelled directed edges $\mathcal{R}$ (representing relationship between entities). In our task, answers are either booleans, or specific subsets of nodes from the KG.

Our model builds a parse for the sentence, in which phrases are grounded in the KG, and a small set of composition modules are used to combine phrases, resulting in a grounding for the complete question sentence that answers it. For example, in Figure 1, the phrases *not* and *cylindrical* are interpreted as a function word and an entity set, respectively, and then *not cylindrical* is interpreted by computing the complement of the entity set. The node at the root of the parse tree is the answer to the question. Our model answers questions by:

(a) Grounding individual tokens in a KG, that can either be grounded as particular sets of entities and relations in the KG, as ungrounded vectors, or marked as being semantically vacuous. For each word, we learn parameters that are used to compute a distribution over semantic types and corresponding denotations in a KG (§ 3.1).

(b) Combining representations for adjacent phrases into representations for larger phrases, using trainable neural composition modules (§ 3.2). This produces a denotation for the phrase.

(c) Assigning a binary-tree structure to the question sentence, which determines how words are grounded, and which phrases are combined using which modules. We build a parse chart subsuming all possible structures, and train a parsing model to increase the likelihood of structures leading to the correct answer to questions. Different parses leading to a denotation for a phrase of type t are merged into an expected denotation, allowing dynamic programming (§ 4).

(d) Answering the question, with the most likely grounding of the phrase spanning the sentence.

3 Compositional Semantics

3.1 Semantic Types

Our model classifies spans of text into different semantic types to represent their meaning as explicit denotations, or ungrounded vectors. All phrases are assigned a distribution over semantic types. The semantic type determines how a phrase is grounded, and which composition modules can be used to combine it with other phrases. A phrase spanning $w_{i..j}$ has a denotation $[\![w_{i..j}]\!]^t_{KG}$ for each semantic type t. For example, in Figure 1, *red* corresponds to a set of entities, *left* corresponds to a set of relations, and *not* is treated as an ungrounded vector.

The semantic types we define can be classified into three broad categories.

Grounded Semantic Types: Spans of text that can be fully grounded in the KG.

1. **Entity** (**E**): Spans of text that can be grounded to a set of entities in the KG, for example, *red sphere* or *large cube*. **E**-type span grounding is represented as an attention value for each entity, $[p_{e_1}, \ldots, p_{e_{|\mathcal{E}|}}]$, where $p_{e_i} \in [0, 1]$. This can be viewed as a soft version of a logical set-valued denotation, which we refer to as a soft entity set.
2. **Relation** (**R**): Spans of text that can be grounded to set of relations in the KG, for example, *left of* or *not right of or above*. **R**-type span grounding is represented by a soft adjacency matrix $A \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$ where $A_{ij} = 1$ denotes a directed edge from $e_i \rightarrow e_j$.
3. **Truth** (**T**): Spans of text that can be grounded with a Boolean denotation, for example, *Is anything red?*, *Is one ball green and are no cubes red?*. **T**-type span grounding is represented using a real-value $p_{true} \in [0, 1]$ that denotes the probability of the span being true.

Ungrounded Semantic Types: Spans of text whose meaning cannot be grounded in the KG.

1. **Vector** (**V**): This type is used for spans representing functions that cannot yet be grounded in the KG (e.g. words such as *and* or *every*). These spans are represented using 4 different real-valued vectors v_1-$v_4 \in \mathbb{R}^2$-$\mathbb{R}^5$, that are used to parameterize the composition modules described in §3.2.
2. **Vacuous** (**ϕ**): Spans that are considered semantically vacuous, but are necessary syntactically, e.g. *of* in *left of a cube*. During composition, these nodes act as identity functions.

Partially-Grounded Semantic Types: Spans of text that can only be partially grounded in the knowledge graph, such as *and red* or *are four spheres*. Here, we represent the span by a combination of a grounding and vectors, representing grounded and ungrounded aspects of meaning respectively. The grounded component of the representation will typically combine with another fully grounded representation, and the ungrounded vectors will parameterize the composition module. We define 3 semantic types of this kind: **EV**, **RV** and **TV**, corresponding to the combination of entities, relations and boolean groundings respectively with an ungrounded vector. Here, the word represented by the vectors can be viewed as a binary function, one of whose arguments has been supplied.

3.2 Composition Modules

Next, we describe how we compose phrase representations (from § 3.1) to represent larger phrases. We define a small set of composition modules, that take as input two constituents of text with their corresponding semantic representations (grounded representations and ungrounded vectors), and outputs the semantic type and corresponding representation of the larger constituent. The composition modules are parameterized by the trainable word vectors. These can be divided into several categories:

Composition modules resulting in fully grounded denotations: Described in Figure 2.

Composition with ϕ-typed nodes: Phrases with type ϕ are treated as being semantically transparent identity functions. Phrases of any other type can combined with these nodes, with no change to their type or representation.

Composition modules resulting in partially grounded denotations: We define several modules that combine fully grounded phrases with ungrounded phrases, by deterministically taking the union of the representations, giving phrases with partially grounded representations (§ 3.1). These modules are useful when words act as binary functions; here they combine with their first argument. For example, in Fig. 1, *or* and *not cylindrical* combine to make a phrase containing both the vectors for *or* and the entity set for *not cylindrical*.

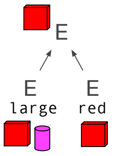

$$p_{e_i} = \sigma\left(\begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} \cdot \begin{bmatrix} p^L_{e_i} \\ p^R_{e_i} \\ 1 \end{bmatrix}\right)$$

E + E → E: This module performs a function on a pair of soft entity sets, parameterized by the model's global parameter vector $[w_1, w_2, b]$ to produce a new soft entity set. The composition function for a single entity's resulting attention value is shown. Such a composition module can be used to interpret compound nouns and entity appositions. For example, the composition module shown above learns to output the intersection of two entity sets.

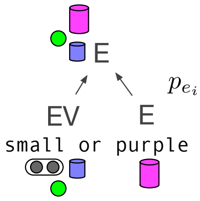

$$p_{e_i} = \sigma\left(\mathbf{v_2} \cdot \begin{bmatrix} p^L_{e_i} \\ p^R_{e_i} \\ 1 \end{bmatrix}\right)$$

EV + E → E: This module combines two soft entity sets into a third set, parameterized by the v_2 word vector. This composition function is similar to a linear threshold unit and is capable of modeling various mathematical operations such as logical conjunctions, disjunctions, differences etc. for different values of v_2. For example, the word *or* learns to model set union.

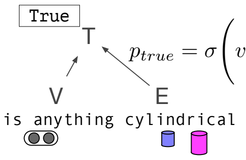

$$p_{true} = \sigma\left(v_3^1\left[\sum_{e_i} \sigma\left(\begin{bmatrix} v_3^3 \\ v_3^4 \end{bmatrix} \cdot \begin{bmatrix} p^R_{e_i} \\ 1 \end{bmatrix}\right)\right] + v_3^2\right)$$

V + E → T: This module maps a soft entity set onto a soft boolean, parameterized by word vector (v_3). The module counts whether a sufficient number of elements are in (or out) of the set. For example, the word *any* should test if a set is non-empty.

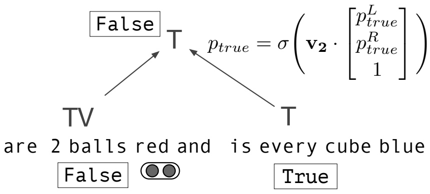

$$p_{true} = \sigma\left(\mathbf{v_2} \cdot \begin{bmatrix} p^L_{true} \\ p^R_{true} \\ 1 \end{bmatrix}\right)$$

TV + T → T: This module maps a pair of soft booleans into a soft boolean using the v_2 word vector to parameterize the composition function. Similar to **EV + E → E**, this module facilitates modeling a range of boolean set operations. Using the same functional form for different composition functions allows our model to use the same ungrounded word vector (v_2) for compositions that are semantically analogous.

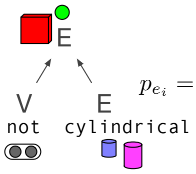

$$p_{e_i} = \sigma\left(\mathbf{v_1} \cdot \begin{bmatrix} p^R_{e_i} \\ 1 \end{bmatrix}\right)$$

V + E → E: This module performs a function on a soft entity set, parameterized by a word vector, to produce a new soft entity set. For example, the word *not* learns to take the complement of a set of entities. The entity attention representation of the resulting span is computed by using the indicated function that takes the $v_1 \in \mathbb{R}^2$ vector of the **V** constituent as a parameter argument and the entity attention vector of the **E** constituent as a function argument.

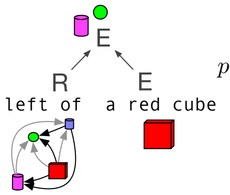

$$p_{e_i} = \max_{e_j} A_{ji} \cdot p^R_{e_j}$$

R + E → E: This module composes a set of relations (represented as a single soft adjacency matrix) and a soft entity set to produce an output soft entity set. The composition function uses the adjacency matrix representation of the **R**-span and the soft entity set representation of the **E**-span.

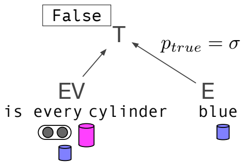

$$p_{true} = \sigma\left(v_4^1\left[\sum_{e_i} \sigma\left(\begin{bmatrix} v_4^3 \\ v_4^4 \\ v_4^5 \end{bmatrix} \cdot \begin{bmatrix} p^L_{e_i} \\ p^R_{e_i} \\ 1 \end{bmatrix}\right)\right] + v_4^2\right)$$

EV + E → T: This module combines two soft entity sets into a soft boolean, which is useful for modelling generalized quantifiers. For example, in *is every cylinder blue*, the module can use the inner sigmoid to test if an element e_i is in the set of cylinders ($p^L_{e_i} \approx 1$) but not in the set of blue things ($p^R_{e_i} \approx 0$), and then use the outer sigmoid to return a value close to 1 if the sum of elements matching this property is close to 0.

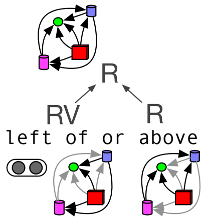

$$A_{ij} = \sigma\left(\mathbf{v_2} \cdot \begin{bmatrix} A^L_{ij} \\ A^R_{ij} \\ 1 \end{bmatrix}\right)$$

RV + R → R: This module composes a pair of soft set of relations to a produce an output soft set of relations. For example, the relations *left* and *above* are composed by the word *or* to produce a set of relations such that entities e_i and e_j are related if either of the two relations exists between them. The functional form for this composition is similar to **EV + E → E** and **TV + T → T** modules.

Figure 2: Composition Modules that compose two constituent span representations into the representation for the combined larger span, using the indicated equations.

4 Parsing Model

Here, we describe how our model classifies question tokens into semantic type spans and computes their representations (§ 4.1), and recursively uses the composition modules defined above to parse the question into a soft latent tree that provides the answer (§ 4.2). The model is trained end-to-end using only question-answer supervision (§ 4.3).

4.1 Lexical Representation Assignment

Each token in the question sentence is assigned a distribution over the semantic types, and a grounded representation for each type. Tokens can only be assigned the **E**, **R**, **V**, and $\boldsymbol{\phi}$ types. For example, the token *cylindrical* in the question in Fig. 1 is assigned a distribution over the 4 semantic types (one shown) and for the **E** type, its representation is the set of *cylindrical* entities.

Semantic Type Distribution for Tokens: To compute the semantic type distribution, our model represents each word w, and each semantic type t using an embedding vector; $v_w, v_t \in \mathbb{R}^d$. The semantic type distribution is assigned with a softmax:

$$p(t|w_i) \propto \exp(v_t \cdot v_{w_i})$$

Grounding for Tokens: For each of the semantic type, we need to compute their representations:

1. **E**-Type Representation: Each entity $e \in \mathcal{E}$, is represented using an embedding vector $v_e \in \mathbb{R}^d$ based on the concatenation of vectors for its properties. For each token w, we use its word vector to find the probability of each entity being part of the **E**-Type grounding:

   $$p^w_{e_i} = \sigma(v_{e_i} \cdot v_w) \;\; \forall\, e_i \in \mathcal{E}$$

   For example, in Fig. 1, the word *red* will be grounded as all the red entities.

2. **R**-Type Representation: Each relation $r \in \mathcal{R}$, is represented using an embedding vector $v_r \in \mathbb{R}^d$. For each token w_i, we compute a distribution over relations, and then use this to compute the *expected* adjacency matrix that forms the **R**-type representation for this token.

   $$p(r|w_i) \propto \exp(v_r \cdot v_{w_i})$$
   $$A^{w_i} = \sum_{r \in \mathcal{R}} p(r|w_i) \cdot A_r$$

   e.g. the word *left* in Fig. 1 is grounded as the subset of edges with label 'left'.

3. **V**-Type Representation: For each word $w \in \mathcal{V}$, we learn four vectors $v_1 \in \mathbb{R}^2, v_2 \in \mathbb{R}^3, v_3 \in \mathbb{R}^4, v_4 \in \mathbb{R}^5$, and use these as the representation for words with the **V**-Type.

4. $\boldsymbol{\phi}$-Type Representation: Semantically vacuous words that do not require a representation.

4.2 Parsing Questions

To learn the correct structure for applying composition modules, we use a simple parsing model. We build a parse-chart over the question encompassing all possible trees by applying all composition modules, similar to a standard CRF-based PCFG parser using the CKY algorithm. Each node in the parse-chart, for each span $w_{i..j}$ of the question, is represented as a distribution over different semantic types with their corresponding representations.

Phrase Semantic Type Potential ($\psi^t_{i,j}$): The model assigns a score, $\psi^t_{i,j}$, to each $w_{i..j}$ span, for each semantic type **t**. This score is computed from all possible ways of forming the span $w_{i..j}$ with type **t**. For a particular composition of span $w_{i..k}$ of type $\mathbf{t_1}$ and $w_{k+1..j}$ of type $\mathbf{t_2}$, using the $\mathbf{t_1} + \mathbf{t_2} \rightarrow \mathbf{t}$ module, the composition score is:

$$\psi^{t_1+t_2\rightarrow t}_{i,k,j} = \psi^{t_1}_{i,k} \cdot \psi^{t_2}_{k+1,j} \cdot e^{\theta \cdot f^{t_1+t_2\rightarrow t}(i,j,k|q)}$$

where θ is a trainable vector and $f^{t_1+t_2\rightarrow t}(i,j,k|q)$ is a simple feature function. Features consist of a conjunction of the composition module type and: the words before (w_{i-1}) and after (w_{j+1}) the span, the first (w_i) and last word (w_k) in the left constituent, and the first (w_{k+1}) and last (w_j) word in the right constituent.
The final **t**-type potential of $w_{i..j}$ is computed by summing scores over all possible compositions:

$$\psi^t_{i,j} = \sum_{k=i}^{j-1} \sum_{\substack{(t_1+t_2\rightarrow t)\\ \in \text{Modules}}} \psi^{t_1+t_2\rightarrow t}_{i,k,j}$$

Combining Phrase Representations ($[\![w_{i..j}]\!]^t_{KG}$): To compute $w_{i..j}$'s **t**-type denotation, $[\![w_{i..j}]\!]^t_{KG}$, we compute an expected output representation from all possible compositions that result in type **t**.

$$[\![w_{i..j}]\!]^t_{KG} = \frac{1}{\psi^t_{i,j}} \sum_{k=i}^{j-1} [\![w_{i..k..j}]\!]^t_{KG}$$

$$[\![w_{i..k..j}]\!]^t_{KG} = \sum_{\substack{(t_1+t_2\rightarrow t)\\ \in \text{Modules}}} \psi^{t_1+t_2\rightarrow t}_{i,k,j} \cdot [\![w_{i..k..j}]\!]^{t_1+t_2\rightarrow t}_{KG}$$

where $[\![w_{i..j}]\!]^t_{KG}$, is the **t**-type representation of the span $w_{i..j}$ and $[\![w_{i..k..j}]\!]^{t_1+t_2\rightarrow t}_{KG}$ is the representation resulting from the composition of $w_{i..k}$ with $w_{k+1..j}$ using the $\mathbf{t_1}+\mathbf{t_2} \rightarrow \mathbf{t}$ composition module.

Answer Grounding: By recursively computing the phrase semantic-type potentials and representations, we can infer the semantic type distribution of the complete question and the resulting grounding for different semantic types t, $[\![w_{1..|q|}]\!]^t_{KG}$.

$$p(t|q) \propto \psi(1, |q|, t) \tag{1}$$

The answer-type (boolean or subset of entities) for the question is computed using:

$$t^* = \underset{t \in \mathbf{T}, \mathbf{E}}{\operatorname{argmax}}\ p(t|q) \tag{2}$$

The corresponding grounding is $[\![w_{1..|q|}]\!]^{t^*}_{KG}$, which answers the question.

4.3 Training Objective

Given a dataset $\mathcal{D}$ of (question, answer, knowledge-graph) tuples, $\{\mathrm{q}^i, \mathrm{a}^i, \mathrm{KG}^i\}_{i=1}^{i=|\mathcal{D}|}$, we train our model to maximize the log-likelihood of the correct answers. We maximize the following objective:

$$\mathcal{L} = \sum_i \log p(\mathrm{a}^i | \mathrm{q}^i, \mathrm{KG}^i) \tag{3}$$

Further details regarding the training objective are given in Appendix A.

5 Dataset

We experiment with two datasets, 1) Questions generated based on the CLEVR (Johnson et al., 2017) dataset, and 2) Referring Expression Generation (GenX) dataset (FitzGerald et al., 2013), both of which feature complex compositional queries.

CLEVRGEN: We generate a dataset of question and answers based on the CLEVR dataset (Johnson et al., 2017), which contains knowledge graphs containing attribute information of objects and relations between them.

We generate a new set of questions as existing questions contain some biases that can be exploited by models.[1] We generate 75K questions for training and 37.5K for validation. Our questions test various challenging semantic operators. These include conjunctions (e.g. *Is anything red and large?*), negations (e.g. *What is not spherical?*), counts (e.g. *Are five spheres green?*), quantifiers (e.g. *Is every red thing cylindrical?*), and relations (e.g. *What is left of and above a cube?*). We create two test sets:

1. **Short Questions:** Drawn from the same distribution as the training data (37.5K).
2. **Complex Questions:** Longer questions than the training data (22.5K). This test set contains the same words and constructions, but chained into longer questions. For example, it contains questions such as *What is a cube that is right of a metallic thing that is beneath a blue sphere?* and *Are two red cylinders that are above a sphere metallic?* Solving these questions require more multi-step reasoning.

REFERRING EXPRESSIONS (GENX) (FitzGerald et al., 2013): This dataset contains human-generated queries, which identify a subset of objects from a larger set (e.g. *all of the red items except for the rectangle*). It tests the ability of models to precisely understand human-generated language, which contains a far greater diversity of syntactic and semantic structures. This dataset does not contain relations between entities, and instead only focuses on entity-set operations. The dataset contains 3920 questions for training, 600 for development and 940 for testing. Our modules and parsing model were designed independently of this dataset, and we re-use hyperparameters from CLEVRGEN.

6 Experiments

Our experiments investigate the ability of our model to understand complex synthetic and natural language queries, learn interpretable structure, and generalize compositionally. We also isolate the effect of learning the syntactic structure and representing sub-phrases using explicit denotations.

6.1 Experimentation Setting

We describe training details, and the baselines.

Training Details: Training the model is challenging since it needs to learn both good syntactic structures and the complex semantics of neural modules—so we use Curriculum Learning (Bengio et al., 2009) to pre-train the model on an easier subset of questions. Appendix B contains the details of curriculum learning and other training details.

---

[1] Johnson et al. (2017) found that many spatial relation questions can be answered only using absolute spatial information, and many long questions can be answered correctly without performing all steps of reasoning. We employ some simple tests to remove trivial biases from our dataset.

Model	Boolean Questions	Entity Set Questions	Relation Questions	Overall
LSTM (No KG)	50.7	14.4	17.5	27.2
LSTM	88.5	99.9	15.7	84.9
Bi-LSTM	85.3	99.6	14.9	83.6
Tree-LSTM	82.2	97.0	15.7	81.2
Tree-LSTM (Unsup.)	85.4	99.4	16.1	83.6
Relation Network	85.6	89.7	97.6	89.4
Our Model (Pre-parsed)	94.8	93.4	70.5	90.8
Our Model	99.9	100	100	99.9

Table 1: **Results for Short Questions** (CLEVRGEN): Performance of our model compared to baseline models on the Short Questions test set. The LSTM (No KG) has accuracy close to chance, showing that the questions lack trivial biases. Our model almost perfectly solves all questions showing its ability to learn challenging semantic operators, and parse questions only using weak end-to-end supervision.

Baseline Models: We compare to the following baselines. **(a)** Models that assume linear structure of language, and encode the question using linear RNNs—LSTM (No KG), LSTM, Bi-LSTM, and a Relation-Network (Santoro et al., 2017) augmented model. [2] **(b)** Models that assume tree-like structure of language. We compare two variants of Tree-structured LSTMs (Zhu et al., 2015; Tai et al., 2015)—Tree-LSTM, that operates on pre-parsed questions, and Tree-LSTM(Unsup.), an unsupervised Tree-LSTM model (Maillard et al., 2017) that learns to jointly parse and represent the sentence. For GenX, we also use an end-to-end semantic parsing model from Pasupat and Liang (2015). Finally, to isolate the contribution of the proposed denotational-semantics model, we train our model on pre-parsed questions. Note that, all LSTM based models only have access to the entities of the KG but not the relationship information between them. See Appendix C for details.

6.2 Experiments

Short Questions Performance: Table 1 shows that our model perfectly answers all test questions, demonstrating that it can learn challenging semantic operators and induce parse trees from end task supervision. Performance drops when using external parser, showing that our model learns an effective syntactic model for this domain. The Relation Network also achieves good performance, particularly on questions involving relations. LSTM baselines work well on questions not involving relations.[3]

Model	Non-relation Questions	Relation Questions	Overall
LSTM (No KG)	46.0	39.6	41.4
LSTM	62.2	49.2	52.2
Bi-LSTM	55.3	47.5	49.2
Tree-LSTM	53.5	46.1	47.8
Tree-LSTM (Unsup.)	64.5	42.6	53.6
Relation Network	51.1	38.9	41.5
Our Model (Pre-parsed)	94.7	74.2	78.8
Our Model	81.8	85.4	84.6

Table 2: **Results for Complex Questions** (CLEVRGEN): All baseline models fail to generalize well to questions requiring longer chains of reasoning than those seen during training. Our model substantially outperforms the baselines, showing its ability to perform complex multi-hop reasoning, and generalize from its training data. Analysis suggests that most errors from our model are due to assigning incorrect structures, rather than mistakes by the composition modules.

Complex Questions Performance: Table 2 shows results on complex questions, which are constructed by combining components of shorter questions. These require complex multi-hop reasoning, and the ability to generalize robustly to new types of questions. We use the same models as in Table 1, which were trained on short questions. All baselines achieve close to random performance, despite high accuracy for shorter questions. This shows the challenges in generalizing RNN encoders beyond their training data. In contrast, the strong inductive bias from our model structure allows it to generalize well to complex questions. Our model outperforms Tree-LSTM (Unsup.) and the version of our model that uses pre-parsed questions, showing the effectiveness of explicit denotations and learning the syntax, respectively.

---

[2] We use this baseline only for CLEVRGEN since GenX does not contain relations.

[3] Relation questions are out of scope for these models.

Model	**Accuracy**
LSTM (NO KG)	0.0
LSTM	64.9
BI-LSTM	64.6
TREE-LSTM	43.5
TREE-LSTM (UNSUP.)	67.7
SEMPRE	48.1
Our Model (Pre-parsed)	67.1
Our Model	73.7

Table 3: **Results for Human Queries** (GENX) Our model outperforms LSTM and semantic parsing models on complex human-generated queries, showing it is robust to work on natural language. Better performance than TREE-LSTM (UNSUP.) shows the efficacy in representing sub-phrases using explicit denotations. Our model also performs better without an external parser, showing the advantages of latent syntax.

Performance on Human-generated Language: Table 3 shows the performance of our model on complex human-generated queries in GENX. Our approach outperforms strong LSTM and semantic parsing baselines, despite the semantic parser's use of hard-coded operators. These results suggest that our method represents an attractive middle ground between minimally structured and highly structured approaches to interpretation. Our model learns to interpret operators such as *except* that were not considered during development. This shows that our model can learn to parse human language, which contains greater lexical and structural diversity than synthetic questions. Trees induced by the model are linguistically plausible (see Appendix D).

Error Analysis: We find that most model errors are due to incorrect assignments of structure, rather than semantic errors from the modules. For example, in the question *Are four red spheres beneath a metallic thing small?*, our model's parse composes *metallic thing small* into a constituent instead of composing *red spheres beneath a metallic thing* into a single node. Future work should explore more sophisticated parsing models.

Discussion: While our model shows promising results, there is significant potential for future work. Performing exact inference over large KGs is likely to be intractable, so approximations such as KNN search, beam search, feature hashing or parallelization may be necessary. To model the large number of entities in KGs such as Freebase, techniques proposed by recent work (Verga et al., 2017; Gupta et al., 2017) that explore representing entities as composition of its properties, such as, types, description etc. could be used. The modules in this work were designed in a way to provide good inductive bias for the kind of composition we expected them to model. For example, **EV** + **E** → **E** is modeled as a linear composition function making it easy to represent words such as *and* and *or*. These modules can be exchanged with any other function with the same 'type signature', with different trade-offs—for example, more general feed-forward networks with greater representation capacity would be needed to represent a linguistic expression equivalent to *xor*. Similarly, more module types would be required to handle certain constructions—for example, a multiword relation such as *much larger than* needs a **V** + **V** → **V** module. This is an exciting direction for future research.

7 Related Work

Many approaches have been proposed to perform question-answering against structured knowledge sources. *Semantic parsing* models have learned structures over pre-defined discrete operators, to produce logical forms that can be executed to answer the question. Early work trained using gold-standard logical forms (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2010), whereas later efforts have only used answers to questions (Clarke et al., 2010; Liang et al., 2011; Krishnamurthy and Kollar, 2013). A key difference is that our model must learn semantic operators from data, which may be necessary to model the fuzzy meanings of function words like *many* or *few*.

Another similar line of work is neural program induction models, such as Neural Programmer (Neelakantan et al., 2017) and Neural Symbolic Machine (Liang et al., 2017). These models learn to produce programs composed of predefined operators using weak supervision to answer questions against semi-structured tables.

Neural module networks have been proposed for learning semantic operators (Andreas et al., 2016b) for question answering. This model assumes that the structure of the semantic parse is given, and must only learn a set of operators. Dynamic Neural Module Networks (D-NMN) extend this approach by selecting from a small set of candidate module structures (Andreas et al., 2016a). We instead learn a model over all possible structures.

Our work is most similar to N2NMN (Hu et al., 2017) model, which learns both semantic operators and the layout in which to compose them. However, optimizing the layouts requires reinforcement learning, which is challenging due to the high variance of policy gradients, whereas our chart-based approach is end-to-end differentiable.

8 Conclusion

We have introduced a model for answering questions requiring compositional reasoning that combines ideas from compositional semantics with end-to-end learning of composition operators and structure. We demonstrated that the model is able to learn a number of complex composition operators from end task supervision, and showed that the linguistically motivated inductive bias imposed by the structure of the model allows it to generalize well beyond its training data. Future work should explore scaling the model to other question answering tasks, using more general composition modules, and introducing additional module types.

Acknowledgement

We would like to thank Shyam Upadhyay and the anonymous reviewers for their helpful suggestions.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural networks for question answering. In *NAACL-HLT*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *CVPR*, pages 39–48.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*.

Nicholas FitzGerald, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Learning distributions over logical forms for referring expression generation. In *EMNLP*.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *EMNLP*.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *ICCV*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *TACL*, 1:193–206.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *EMNLP*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *ACL*.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *CoRR*, abs/1705.09189.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In K. J. J. Hintikka, J. Moravcsic, and P. Suppes, editors, *Approaches to Natural Language*, pages 221–242. Reidel, Dordrecht.

Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. In *ICLR*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *ACL*.

Ethan Perez, Harm de Vries, Florian Strub, Vincent Dumoulin, and Aaron C. Courville. 2017. Learning visual reasoning without strong priors. *CoRR*, abs/1707.03017.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Timothy P. Lillicrap. 2017. A simple neural network module for relational reasoning. In *NIPS*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *ACL*.

Patrick Verga, Arvind Neelakantan, and Andrew McCallum. 2017. Generalizing to unseen entities and entity pairs with row-less universal schema. In *EACL*.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *TACL*, 6:253–267.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI*.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *ICML*.

Cross-Pair Text Representations for Answer Sentence Selection

Kateryna Tymoshenko
DISI, University of Trento (Adeptmind scholar)
38123 Povo (TN), Italy
`kateryna.tymoshenko@unitn.it`

Alessandro Moschitti[*]
Amazon
Manhattan Beach, CA 90266, USA
`amosch@amazon.com`

Abstract

High-level semantics tasks, e.g., paraphrasing, textual entailment or question answering, involve modeling of text pairs. Before the emergence of neural networks, this has been mostly performed using intra-pair features, which incorporate similarity scores or rewrite rules computed between the members within the same pair. In this paper, we compute scalar products between vectors representing similarity between members of different pairs, in place of simply using a single vector for each pair. This allows us to obtain a representation specific to any pair of pairs, which delivers the state of the art in answer sentence selection. Most importantly, our approach can outperform much more complex algorithms based on neural networks.

1 Introduction

Answer sentence selection (AS) is an important subtask of open-domain Question Answering (QA). Its input are a question Q and a set of candidate answer passages $A = \{A_1, A_2, ..., A_N\}$, which may, for example, be the output of a search engine. The objective consists in selecting A_i, $i \in \{1, ..., N\}$ that contain correct answers.

Pre-deep learning renaissance approaches to AS typically addressed the task by modeling Q-to-A (**intra-pair**) similarities (Yih et al., 2013; Wang et al., 2007; Heilman and Smith, 2010; Wang and Manning, 2010). Q-to-A similarity and alignment are indeed crucial, but, in practice, it is very difficult to automatically extract meaningful relations between Q and A. For example, consider two positive Q/A pairs in Table 1. If we want to learn a model based only on the intra-pair Q-to-A matches, simple lexical matching (marked with italics) will not be enough. One would need to conduct more complex processing and identify that *movie* and *film* are synonyms, and that the n-gram *play in the movie* or *be in the movie* can be paraphrased as *star*. While the former can be easily detected using an external lexical resource, e.g., WordNet (Fellbaum, 1998), the latter would require more complex inference.

On the other hand, Q_1 and Q_2 contain the same pattern **who ... in the movie ...**, and their respective answers contain **film ... starring ...**. If we know that $P_1 = (Q_1, A_1)$ is a positive AS example and want to classify $P_2 = (Q_2, A_2)$, then high Q_2-to-Q_1 and A_2-to-A_1 **cross-pair** similarities can suggest that P_1 and P_2 are likely to have the same label. This idea, for example, was exploited by Severyn and Moschitti (2012), whose system measures syntactic-semantic similarities directly between structural syntactic tree representations of Q_1/Q_2 and A_1/A_2. This model still exhibited state-of-the-art performance in 2016 (Tymoshenko et al., 2016a).

Deep neural networks (DNNs) also naturally use such cross-pair similarity when modeling two input texts, and then further combine it with intra-pair similarity, for example, by means of attention mechanisms (Shen et al., 2017), compare-aggregate architectures (Bian et al., 2017; Wang and Jiang, 2017), or fully-connected layers (Severyn and Moschitti, 2015, 2016; Rao et al., 2016).

In this work, we observe that: (i) the high accuracy of the kernel model by Severyn and Moschitti (2012) was due not only to the use of syntactic structures, but also to the use of cross-pair similarities; and (ii) the success of DNNs for QA can be partially attributed to an implicit combination of cross- and intra-pair similarity.

More specifically, we investigate, whether simple similarity metrics, e.g., cosine similarity between standard vector representations, can perform competitively to the state-of-the-art neural models when employed as cross-pair kernels.

---

[*] Professor at the University of Trento.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2162–2173
Brussels, Belgium, October 31 - November 4, 2018.

	Question		Answer	Label
Q_1	**who** plays *mary poppins* **in the movie**?	A_1	*Mary Poppins* is a 1964 musical **film** **starring** Julie Andrews, Dick Van Dyke, David Tomlinson, and Glynis Johns, produced by Walt Disney, and based on the *Mary Poppins* books series by P. L. Travers.	TRUE
Q_2	**WHO** WAS **IN THE MOVIE** *I CONFESS* WITH *MONTGOMERY CLIFT*	A_2	*I Confess* is a 1953 drama **film** directed by Alfred Hitchcock, and **starring** *Montgomery Clift* as Fr. Michael William Logan, a Catholic priest, Anne Baxter as Ruth Grandfort, and Karl Malden as Inspector Larrue .	TRUE

Table 1: Question/Answer Sentence pairs from WikiQA corpus. We use italic font to mark intra-pair lexical matches between Q_1 and A_1, Q_2 and A_2, and bold font to mark the cross-pair matches between Q_1 and Q_2, A_1 and A_2.

To this end, we apply linear and cosine kernels to Q_i/Q_j and A_i/A_j pairs ($i, j = 1, ..., N$) represented as a bag-of-words (BoW) or an averaged sum of their pretrained word embeddings. Then, we combine them with the cross-pair Tree Kernels (TKs) and kernels applied to the traditional Q/A intra-pair similarity feature vector representations in a composite kernel and use it in an SVM model.

We experiment with three reference datasets, WikiQA (Yang et al., 2015), TREC13 (Yao et al., 2013; Wang et al., 2007) and SemEval-2016, Task 3.A (Nakov et al., 2016), using a number of lexical-overlap/syntactic kernels. The latter challenge refers to a community question answering (cQA) task. It consists in reranking the responses to user questions from online forums. It is the same setting as AS, but the text of questions and answer sentences can be ungrammatical due to the nature of the online forum language.

We obtain competitive results on WikiQA and SemEval tasks, showing that: (i) simple BoW representations, when used in cross-pair kernels, perform comparably to and even outperform hand-crafted intra-pair features. (ii) In cQA, simple cross-pair embedding- and BoW-based similarity features outperform domain-specific similarity features, which are hand-crafted from intra-pair members. The simple features also perform comparably to syntactic TKs. (iii) We show that a combination of simple cosine- intra- and cross-pair kernels with TKs can outperform the most recent state-of-the-art DNN architectures.

Assuming the conjecture of our paper correct, *cross-pair modeling is the major neural network contribution*, the last point above is not surprising as on relatively small datasets kernels-based models can exploit syntactic information very effectively while neural models cannot.

The paper is structured as follows. We describe the kernels incorporating intra- and cross-pair matches in Sec. 3.2, list the simple cross- and intra-pair features in Sec. 3.3, describe strong hand-crafted baseline features in Sec. 4, and report the experimental results in Sec. 5.

2 Related work

Early approaches to AS typically focused on modeling intra-pair Q-to-A alignment similarities. For example, Yih et al. (2013) proposed a latent alignment model that employed lexical-semantical Q-to-A alignments, Wang et al. (2007) modeled syntactic alignments with probabilistic quasi-synchronous grammar, and Heilman and Smith (2010); Yao et al. (2013); Wang and Manning (2010) employed Tree Edit Distance-based Q-to-A alignments.

Originally, the idea of cross-pair similarity was proposed by Zanzotto and Moschitti (2006) and applied to the recognizing textual entailment task, which consists in detecting whether a text T entails a hypothesis H. They assumed that if two H/T pairs $\langle H_1, T_1\rangle$ and $\langle H_2, T_2\rangle$ share the same T-to-H "rewrite rules", they are likely to share the same label. Based on this idea, they proposed an algorithm applying TKs to (H_1, H_2) and (T_1, T_2) syntactic tree representations, enriched with H-to-T intra-pair rewrite rule information. More concretely, such algorithm aligns the constituents of H with T and then marks them with symbols directly in the trees. This way the alignment information can be matched by tree kernels applied to cross-pair members.

Then, a line of work on AS, started by Severyn and Moschitti (2012, 2013); Severyn et al. (2013), was inspired by a similar idea of incorporating "rewrite rules" directly into the tree representations of Q_1/A_1 and Q_2/A_2. They represent Q and A as syntactic trees enhanced with Q-to-A relational information, and apply TKs (Moschitti, 2006) to (Q_1, Q_2) and (A_1, A_2). Thus they model **cross-pair** similarity, and learn important patterns occurring in Q and A separately. As shown in (Tymoshenko et al., 2016a), this approach is competitive with convolutional neural networks (CNNs).

In our approach, instead of using only one TK, we employ a number of different word-based kernels, most of which can be computed more efficiently than TKs.

Most recent AS models are based on Deep Neural Networks (DNNs), which learn distributed representations of the input data. DNNs are trained to apply series of non-linear transformations to the input Q and A, represented as compositions of word or character embeddings. DNN architectures learn AS-relevant patterns using intra-pair similarities as well as cross-pair, Q-to-Q and A-to-A, similarities, when modeling the input texts. For example, the CNN network by (Severyn and Moschitti, 2015) has two separate embedding layers for Q and A, which are followed by the respective convolution layers, whose output is concatenated and then passed through the final fully-connected joint layer. The weights in the Q and A convolution layers are learned by means of the backpropagation algorithm on the training Q/A pairs. Thus, obviously, classifying a new Q/A pair is partially equivalent to performing the implicit cross-pair Q-to-Q and A-to-A comparison.

Additionally, the DNN approaches model the *Q-to-A relatedness* explicitly in a variety of ways, e.g., by: (i) using a Q-to-A transformation matrix and simple Q-to-A similarity features (Yu et al., 2014; Severyn and Moschitti, 2015), (ii) relying on RNN and LSTM architectures (Wang and Nyberg, 2015; Shen et al., 2017), (iii) employing attention components (Yin et al., 2016; Shen et al., 2017; Wang et al., 2016a), (iv) decomposing input into similarity and dissimilarity matches (Wang et al., 2016b) or (v) using the compare-aggregate method (Wang and Jiang, 2017; Bian et al., 2017).

We believe that the ability of DNNs to implicitly capture cross-pair relational matching, i.e., the capacity of learning from (Q_1, Q_2) and (A_1, A_2), is a very important factor to their high performance. This is of course paired with their ability to learn non-linear patterns and capture Q-to-A relatedness by means of attention mechanisms. It should be noted that the latter are typically hard-coded in kernel models as lexical matching/similarity (Severyn and Moschitti, 2012). This is effective as much as the attention approach, at least with standard-size dataset, also in neural models (Severyn and Moschitti, 2016).

In our work, we model Q-to-A, Q-to-Q and A-to-A similarities with intra- and cross-pair kernels and show that such combination also exhibits state-of-the-art performance on the reference corpora. In addition, our approach can be applied to smaller datasets as it utilizes less parameters, and can provide insights on future DNN design.

3 Cross-pair similarity kernels for text

3.1 Background on Kernel Machines

Kernel Machines (KMs) allow for replacing the dot product with kernel functions directly applied to examples, i.e., they avoid mapping examples into vectors. The main advantage of KMs is a much lower computational complexity than the dot product as the kernel computation does not depend on the size of the feature space.

KMs are linear classifiers: given a labeled training dataset $S = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$, their classification function can be defined as:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b.$$

where $\mathbf{x}$ is a classification example, $\mathbf{w}$ is the gradient of the separating hyperplane, and b its bias. The equation shows that the gradient is a linear combination of the training points $\mathbf{x}_i \in \mathbb{R}^n$ multiplied by their labels $y_i \in \{-1, 1\}$ and their weights $\alpha_i \in \mathbb{R}^+$. Note that the latter are different from zero only for the support vectors: this reduces the classification complexity, which will be lower than O(n) for each example.

We can replace the scalar product with a kernel function directly defined over a pair of objects, $K(o_i, o) = \phi(o_i)\phi(o)$, where $\phi : O \to \mathbb{R}^n$ maps from objects to vectors of the final feature space. The new classification function becomes: $f(o) = \sum_{i=1}^{n} \alpha_i y_i K(o_i, o) + b$, which only needs the initial input objects.

3.2 Inter- and intra-pair match kernel

We cast AS as a text pair classification task: given a pair, $P = (Q, A)$, constituted by a question (Q) and a candidate answer sentence (A), we classify it as either correct or incorrect. We used KMs, where $K(\cdot, \cdot)$ operates on two pairs, $P_1 = (Q_1, A_1)$ and $P_2 = (Q_2, A_2)$.

3.2.1 Intra-pair similarity

A traditional baseline approach would (i) represent Q/A pairs as feature vectors, where the components are similarity metrics applied to Q and A, e.g., a world overlap-based similarity; and (ii) train a classification model, e.g., an SVM using the following kernel:

$$K_{IP}(P_1, P_2) = K_v \left(V_{\langle Q_1, A_1 \rangle}, V_{\langle Q_2, A_2 \rangle} \right), \quad (1)$$

where K_v can be any kernel operating on the feature vectors, e.g., the polynomial or linear (as in our work) kernel. $V_{\langle T_1, T_2 \rangle}$ is a vector built on N similarity features, $\langle f_1(\cdot,\cdot), f_2(\cdot,\cdot), ..., f_N(\cdot,\cdot) \rangle$, extracted by applying similarity metrics to two texts, T_1 and T_2 (see Sec. 3.3 for the list of the similarity metrics we used). K_{IP} merely uses intra-pair similarities.

3.2.2 Cross-pair similarity

We incorporate the intuition, *similar questions are likely to demand similar answer patterns*, by means of a cross-pair kernel, which measures similarity between questions and answers from P_1 and P_2 as follows:

$$\begin{aligned} K_{CP}(P_1, P_2) &= V_{\langle Q_1, Q_2 \rangle} \cdot V_{\langle A_1, A_2 \rangle} \\ &= \sum_{i=1}^{N} f_i(Q_1, Q_2) \cdot f_i(A_1, A_2) \end{aligned} \quad (2)$$

K_{CP} measures P_1-to-P_2 similarity in terms of a sum of the products of Q_1-to-Q_2 and A_1-to-A_2 similarities. Note, that within K_{IP}, $f_i(Q_i, A_i)$ is merely an i-th feature in the $V_{\langle Q_i, A_i \rangle}$ feature vector. At the same time, within K_{CP}, $f_i(\cdot,\cdot)$ becomes a kernel, which takes the (Q_1, Q_2) or (A_1, A_2) pairs as input. In other words, $V_{\langle Q_1, Q_2 \rangle} \cdot V_{\langle A_1, A_2 \rangle}$ is a sum of products of $f_i(\cdot,\cdot)$ kernels applied to the (Q_1, Q_2) and (A_1, A_2) pairs. K_{CP} is a valid kernel if the similarity metrics used to compute the $f_i(\cdot,\cdot)$ are valid kernel functions.

Finally, combining K_{IP} and K_{CP} enables learning of two different kinds of valuable cross- and intra-pair AS patterns. We combine various K_{IP} and K_{CP} by summing them or by training a meta-classifier on their outputs. See Section 5.4 for more details. Figure 1 summarizes the differences between the K_{IP} and K_{CP} computation processes described above.

3.3 Similarity features

We employ three similarity feature types as $f_i(\cdot,\cdot)$. Two of them are computed using the cosine similarity metrics and differ only in terms of the input texts, T_1 and T_2, representations. The other type is constituted by TKs applied to the structural representations of T_1 and T_2. Note that, since cosine similarity and TKs are valid kernels, K_{CP} is also guaranteed to be a valid kernel when computed using these similarity features.

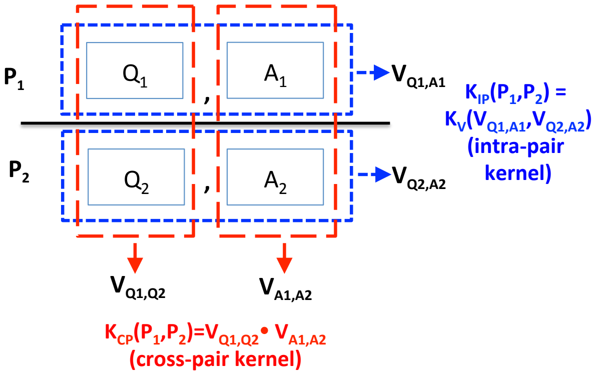

Figure 1: Feature extraction schema for two Q/A pairs, P_1 and P_2. V_{T_1,T_2} is a vector of similarity features extracted for a pair of texts T_1, T_2, the respective dashed boxes show from which pair of input texts they are extracted.

3.3.1 Bag-of-n-grams overlap (B)

$f_B^{t,l,s}(T_1, T_2)$ is a cosine similarity metric applied to the bag-of-n-grams vector representations of T_i, $BOW_{\{t,l,s\}}(T_i)$, $i = 1, 2$. The $\{t, l, s\}$ index describes an n-gram representation configuration: t denotes whether the n-grams are assembled of word lemmas (L), or their part-of-speech tags (POS), or lemmas concatenated with their respective POS-tags (L_{POS}); l is a (n_1, n_2) tuple, with n_1 and n_2 being the minimal and maximal length of n-grams considered, respectively; and s is YES if the representation discards the stopwords and NO, otherwise.

We used $\{t, l, s\}$ configurations from the following set: $C = (\{L, L_{POS}\} \times \{(1,2), (1,3), (1,4), (2,4), (2,3)\} \times \{YES, NO\}) \cup (\{POS\} \times \{(1,4), (2,4)\} \times \{YES\})$. It follows that $|C| = 23$, which means we have 23 similarity features, $f_B^{t,l,s}(T_1, T_2)$, in total, in the intra-pair setting. The respective cross-pair kernels are a composite kernel summing 23 products of cosine kernels applied to 23 different (Q_1, Q_2) and (A_1, A_2) bag-of-ngram representations.

3.3.2 Embedding-based similarities (E).

We represent an input text as an average of embeddings of its lemmas from pre-trained word embedding models. Then, the embedding feature $f_{model}^{E}(T_1, T_2)$ is the cosine kernel applied to the embedding-based representations of T_1 and T_2. We use two pretrained embeddings: Word2Vec (Mikolov et al., 2013) and

GloVe (Pennington et al., 2014), resulting in three[1] embedding-based features (see Sec. 5 for more technical details).

3.3.3 Tree-kernel based similarities

Following the framework defined in (Severyn et al., 2013; Tymoshenko et al., 2016a), we represent T_1 and T_2 as syntactico-semantic structures and use TKs as semantic similarity metrics. When computing K_{CP} with TK as similarities, in Eq.2, we employ summation instead of multiplication[2].

More specifically, we represent T_1 and T_2 as (i) constituency trees and apply subset TK (SST); or (ii) shallow chunk-based trees, similar to the one presented in Figure 2, and apply partial tree (PTK) kernel. In the shallow trees, lemmas are leaves and POS tags are pre-terminals. POS nodes are grouped under chunk nodes, and then under the sentences nodes. These representations encode also some intra-pair similarity information, e.g., prefix `REL` denotes the lexical Q-to-A match. In a structural representation, we prepend it to the parent and grand-parent nodes of lemmas which occur both in Q and A, e.g., "Mary" in the first example of Table 1.

Then, for factoid QA[3], we mark focus words in Q and entities in A, if the answer contains any named entities of types matching the question expected answer type (EAT)[4]. More specifically, we mark the semantic Q-to-A match by prepending the `REL-FOCUS-<EAT>` label to the answer chunk nodes that contain such named entities and also to the question focus word. Here, `<EAT>` stands for the EAT label. For example, in the Q_1/A_1 pair in Table 1, the Q_1 EAT is HUMan, and the matching named entities include "Julie Andrews", "David Tomlinson" and others. Figure 2 depicts Q_1 annotated both with `REL-` and `REL-FOCUS` links. We detect both question focus and EAT automatically. Due to the space limitations, we do not describe the structural representations and matching algorithms in more detail, but refer the reader to the works above.

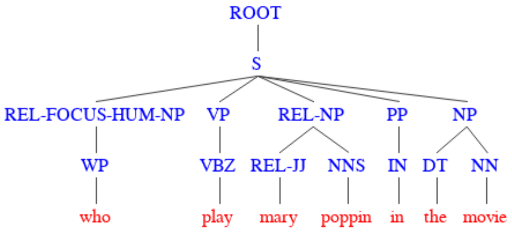

Figure 2: Shallow syntactic representation of A_1 from the running example in Table 1

(i) **cosine similarity** applied to the BoW representations of T_1 and T_2 in terms of word lemmas, bi-, three-, four-grams (computed twice with and without stopwords); POS-tags; dependency triplets;
(ii) **longest common string subsequence** measure w. and w/out stopwords;
(iii) **Jaccard similarity metric** applied to one-, two-, four, three-grams w. and w/out stopwords;
(iv) **word n-gram containment measure** on uni- and bi-grams w. and w/out stopwords (Broder, 1997);
(v) **greedy string tiling** (Wise, 1996) with minimum matching length of 3 ;
(vi) **string kernel similarity** (Lodhi et al., 2002);
(vii) **expected answer type match:** percentage of named entities (NE) in the answer passage compatible with the question class[5];
(viii) **WordNet-based similarity.** WordNet T_1/T_2 common lemma/synonym/hypernym overlap ratio;
(ix) **PTK (Moschitti, 2006) similarity** between constituency or dependency tree representations of input texts;

Table 2: Strong baseline fatures

4 Strong baseline feature vector

As a strong baseline, we use similarity feature vectors and intra-pair K_{IP} kernel.

For the **factoid answer sentence selection** task, we use 47 strong features listed in Table 2. This is a compilation of features used in the top-performing system at SemEval-2012 Semantic Text Similarity workshop (Bär et al., 2012) and earlier factoid QA work (Severyn and Moschitti, 2012), extended with few additional features.

For the **community question answering (cQA)** task, we employ instead a combination of similarity-based and thread-level features shown to be very effective for the cQA task (Nicosia et al., 2015; Barrón-Cedeño et al., 2016). We use the exact feature combination from (Barrón-Cedeño et al., 2016), which includes both lexical and syntactic similarity measures (cosine similarity of bag-of-words, PTK similarity over syntactic tree representations of the input texts) and thread-

---

[1] We use Word2Vec embeddings trained on two different corpora, which result in two features, and GloVe trained on one corpus.

[2] We have opted to use summation in this case to follow the earlier work.

[3] WikiQA and TREC13 are the factoid AS datasets, as their questions ask for a specific fact, e.g. date or a name.

[4] For example, the PERson named entity type matches the HUMan EAT. More specifically, we employ the following NER-to-EAT matching rules: PERson, ORGanization → HUMan; LOCation → LOCation; DATE, TIME, MONEY, PERCENTAGE, DURATION, NUMBER, SET → NUM; ORGanization, PERson, MISCellanious → ENTiTY. We employ the (Li and Roth, 2002) coarse-grained EAT taxonomy and Stanford CoreNLP (Manning et al., 2014) entity types.

MODE	TRAIN		DEV		TEST	
	Q	**A**	**Q**	**A**	**Q**	**A**
raw	2118	20360	296	2733	633	6165
no all$^-$	873	8672	126	1130	243	2351
clean	857	8651	121	1126	237	2341

Table 3: WikiQA corpus statistics

level domain specific features (*are the question and comment authored by the same person?*, *does the comment contain any questions?*, and so on.).

We cannot directly use these feature vectors in the K_{CP} kernels, as not all functions used to compute features are valid kernels, e.g., the longest common string subsequence is not a kernel function. Moreover, some of them can be computed only on the (Q, A) pairs, e.g., the expected type match feature (vii) in Tab. 2, or many of the cQA domain-specific features.

5 Experiments

We conduct experiments on three corpora, namely TREC13, WikiQA and SemEval, and evaluate the results in terms of Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). Our code is available at `https://github.com/iKernels/RelTextRank`.

5.1 Datasets

WikiQA dataset. WikiQA (Yang et al., 2015) is a factoid answer sentence selection dataset with Bing query logs as questions. Candidate answer sentences are extracted from Wikipedia and labeled manually. Some of the questions have no correct answer sentence (all$^-$) or have only correct answer sentences (all$^+$). Table 3 reports the statistics of the WikiQA corpus as distributed (raw), without all$^-$ questions, and without both all$^-$ and all$^+$ questions (clean). We train in the "no all$^-$" mode using 10 answer sentences per question[6] and test in the "clean" mode.

TREC13 dataset. A factoid answer sentence selection dataset originally presented in (Wang et al., 2007)[7], also frequently called **QASent** (Yang et al., 2015). We train on 1,229 automatically labeled TREC8-12 questions. We use only 10 candidate answer sentences per question. We test in the "clean" setting defined in (Rao et al., 2016), i.e., we discard the all$^+$ and all$^-$ questions, resulting in 65 DEV and 68 TEST questions, respectively. DEV and TEST contain 1117 and 1442 candidate associated answer sentences, respectively.

SemEval-2016, Task 3.A dataset. SemEval cQA dataset is a benchmark dataset in the SemEval 2016 Task 3. A question-to-comment similarity competition. It is a collection of user questions and the respective answer comment threads from Qatar Living forum, where the user comments to questions were manually labeled as correct or incorrect. Each question has 10 respective candidate answers. The training, dev and test sets have 1790, 244 and 327 questions, respectively. The AS task consists in reranking comments with respect to the question: most questions are non-factoid and the text is often noisy.

5.2 Models

We used the following notation:

B, **E**. Intra-pair K_{IP} kernels (see Sec. 3.2) using the eponymous similarity features from Sec. 3.3.

V. Linear kernel applied to the strong intra-pair feature vector representation defined in Section 4. Note that, as already mentioned in Sec. 4, due to the slightly different nature of the factoid and community question answering tasks, we used different strong feature groups for WikiQA, TREC13 (Table 2) and SemEval-2016.

$\mathbf{B}_{cr}$, $\mathbf{E}_{cr}$. Cross-pair K_{CP} kernel applied to B and E similarity feature vectors respectively. More specifically, B_{cr} and E_{cr} are a sum of 23 and 3 cross-pair kernel products, respectively (see Eq. 2 and Sec. 3.2).

PTK, **SST** are the cross-pair PTK and SST tree kernels applied to the shallow chunk- and constituency-based representations (see Sec. 3.3).

"+" denotes kernel summation. We use this symbol to denote that we sum the gram-matrices for the distinct standalone kernels, and use the resulting kernel matrix as input to SVMs.

$\mathbf{META}_{BASE;PTK}$, $\mathbf{META}_{BASE;SST}$. Logistic regression metaclassifiers trained on the outputs of two standalone systems, namely (i) V+B_{cr}+E_{cr}+E (we denote it as $BASE$ to simplify the notation), and (ii) PTK or SST, respectively. We ran 10-fold cross-validation on the training set and used the resulting predictions as training data for the ensemble classifier. We did not use the development or training sets for any parameter tuning, thus we report the results both on the DEV and TEST sets.

$\mathbf{SUM}_{BASE;PTK}$, $\mathbf{SUM}_{BASE;SST}$. Simple metaclassifiers, summing the output of the BASE and

---

[6] The 10 answer sentences per question limit speeds up training time without loss in performance

[7] We use the version distributed by (Yao et al., 2013) in `https://code.google.com/p/jacana/`

PTK or SST systems, respectively.

5.3 Toolkits

We trained the models using *scikit-learn*[8] by Pedregosa et al. (2012) using the SVC version of SVM with precomputed K_{IP} and K_{CP} kernel matrices and default parameters. We trained the ensemble model using the scikit *LogisticRegression* classifier implementation with the default parameters. We used spaCy library[9] and scikit to obtain bag-of-n-gram representations for the B similarity features, and to compute B- and E- base gram matrices.

We used the RelTextRank framework[10] (Tymoshenko et al., 2017b) to generate the structural representations for the TK similarity features and to extract the strong baseline feature vectors from Sec. 4. We used KeLP (Filice et al.) to compute the TK gram matrices.

Regarding the Embedding-based similarities (E), we obtain three similarity features by using three word embedding models to generate the representations of the input texts, T_1 and T_2, namely GloVe vectors trained on common crawl data[11], Word2Vec vectors pre-trained on Google News[12], and another Word2Vec vectors model[13] pre-trained on Aquaint[14] plus Wikipedia.

5.4 Results and discussion

Table 4 reports the results obtained with the intra- and cross-pair kernels K_{IP}, K_{CP} and their combinations. In the following, we describe the results according to the model categories above.

Intra-pair kernels. Taking into account intra-pair similarity is the standard approach in the majority of the previous non-DNN work. In our experiments, we implement this approach as K_{IP} using B, E, V groups of similarity features. K_{IP} performs worse than the state-of-art (SoA) DNN systems on all the datasets (see tables 5, 6 and 7, for the SoA systems).

The results on WikiQA are particularly low even when the best K_{IP} system, B+V+E, is used, which scores up to 15 points less than the state of the art. This confirms the Yang et al. (2015) observation on WikiQA, according to which, simple word matching methods are likely to underperform on its data, considering how it was built. Nevertheless, despite its simplicity, B+V+E performs comparably to the Yang et al. (2015) reimplementation of LCLR, the complex latent structured approach employing rich lexical and semantic intra-pair similarity features (Yih et al., 2013). Yang et al. (2015) report that on WikiQA LCLR obtains MRR of 60.86 and MAP of 59.93.

Then, on TREC13 and SemEval-2016, the intra-pair *V*, *V+E* and *B+V+E* kernels exhibit rather high performance, however, they are still significantly below the state of the art, thus confirming our hypothesis that intra-pair similarity alone does not guarantee top results.

Cross-pair kernels. B_{cr} and E_{cr} obtain rather high results on WikiQA and SemEval. On WikiQA, both B_{cr} and $B_{cr} + E_{cr}$ outperform all the intra-pair kernels by a large margin, while, on SemEval, they perform comparably to the manually engineered domain-specific V features of Nicosia et al. (2015). On the contrary, on TREC13, V outperforms both B_{cr} and E_{cr}, thus showing that TREC13 is indeed biased towards intra-pair relatedness features by construction.

More complex PTK and SST cross-pair kernels, both alone and combined with B_{cr}, E_{cr}, typically outperform the standalone B_{cr} and E_{cr} on all the corpora (PTK on TREC13 and WikiQA, and SST and B_{cr}+E_{cr}+PTK on SemEval). This can be explained by the fact that PTK and SST are able to learn complex syntactic patterns and also contain some information about intra-pair relations, namely `REL-` labels described in Sec. 3.2. Thus, it is natural that they outperform simpler cross-pair kernels. Nevertheless, on WikiQA-DEV, B_{cr}+E_{cr} performs very close to PTK. Moreover, on SemEval, B_{cr}+E_{cr} outperforms PTK and is behind SST for less than 1 point in terms of MAP. This can be explained by the fact that Q and A, in SemEval, are frequently ungrammatical as the cQA corpus is collected from online forums.

Finally, note that the B_{cr}+E_{cr}+PTK system, which does not use any cQA domain-specific features, is only 0.56 MAP points behind KeLP, the best-performing system in the SemEval competition (see Line 1 of Table 7).

Kernels combining the intra- and cross-pair similarities. The V+B_{cr}+E_{cr}+E combination (we

---

[8] `http://scikit-learn.org/`

[9] `https://spacy.io/`

[10] This tool employs Stanford CoreNLP 3.6.0 (Manning et al., 2014) for text processing; and DKProSimilarity (Bär et al., 2013) to extract features (ii)-(v).

[11] `http://nlp.stanford.edu/data/glove.42B.300d.zip`

[12] `https://code.google.com/archive/p/word2vec/`

[13] `https://github.com/aseveryn/deep-qa`

[14] `https://catalog.ldc.upenn.edu/LDC2002T31`

		TREC13				WikiQA				Semeval-2016			
		DEV		TEST		DEV		TEST		DEV		TEST	
		MRR	**MAP**	**MRR**	**MAP**	**MRR**	**MAP**	**MRR**	**MAP**	**MRR**	**MAP**	**MRR**	**MAP**
Intra-pair kernel (K_{IP})	B	61.91	57.69	65.68	55.63	51.36	51.41	54.69	53.8	57.90	51.45	65.70	57.28
	V	85.38	78.01	78.82	69.18	47.07	46.53	50.98	50.26	69.75	63.68	81.94	73.89
	E	71.15	67.88	73.27	66.48	48.25	48.49	53.14	52.42	61.50	54.25	67.05	60.25
	V+E	**88.48**	**79.64**	**79.63**	**69.57**	49.24	49.08	53.32	52.87	**70.98**	**64.43**	**82.06**	**74.13**
	B+V+E	84.84	77.31	79.22	68.83	**55.47**	**55.75**	**60.19**	**59.52**	**70.98**	64.37	80.84	74.09
Cross-pair kernel (K_{CP})	B$_{cr}$	72.64	65.81	63.70	56.16	72.71	72.00	68.84	67.67	72.28	65.20	81.63	74.68
	E$_{cr}$	72.22	66.36	76.94	66.14	64.61	64.38	60.77	59.42	70.07	63.92	80.60	73.62
	PTK	**89.49**	**81.02**	84.09	76.06	75.56	74.27	**75.60**	**74.67**	70.45	63.04	81.96	74.30
	SST	77.52	68.8	78.73	70.91	72.38	71.78	69.49	68.1	73.77	66.00	82.37	75.65
	B$_{cr}$+E$_{cr}$	71.52	66.35	69.24	62.16	74.62	74.18	70.03	68.76	71.56	65.26	81.81	75.06
	B$_{cr}$+E$_{cr}$+PTK	84.74	79.24	**85.07**	**77.18**	75.34	74.91	71.01	69.61	**75.99**	**68.82**	**85.26**	**78.63**
	B$_{cr}$+E$_{cr}$+SST	73.80	67.69	72.43	65.75	**75.90**	**75.59**	71.23	69.72	73.38	66.46	82.82	75.91
Combination	V+E$_{cr}$	86.92	78.34	**85.65**	76.72	71.97	71.77	67.63	66.40	75.83	68.75	84.37	77.60
	V+B$_{cr}$	88.85	80.51	81.99	74.46	75.84	75.22	74.59	73.22	75.42	68.06	85.78	78.47
	V+PTK	87.44	80.38	84.25	75.74	75.19	74.81	72.78	71.70	74.15	67.83	**86.68**	79.22
	V+SST	86.41	78.58	84.12	76.86	72.96	72.74	71.24	69.74	75.29	68.71	85.69	79.07
	V+B$_{cr}$+E$_{cr}$+E	**90.51**	**82.94**	82.40	75.98	76.65	76.29	**74.68**	**73.34**	76.07	69.75	85.54	78.83
	V+B$_{cr}$+E$_{cr}$+E+PTK	89.10	81.23	83.14	**76.77**	**76.88**	**76.59**	74.05	72.71	**76.60**	**69.97**	86.52	**79.79**
	V+B$_{cr}$+E$_{cr}$+E+SST	90.56	82.95	84.24	76.63	76.73	76.11	74.08	72.73	76.00	69.89	86.23	79.43
Ensemble	SUM$_{BASE;SST}$	89.41	80.02	84.07	77.35	77.12	76.74	74.07	72.52	76.49	69.50	85.45	78.92
	SUM$_{BASE;PTK}$	88.33	81.37	**86.89**	**77.65**	**77.94**	**77.43**	**77.00**	**75.59**	74.60	67.69	84.94	78.65
	META$_{BASE;PTK}$	88.26	81.26	**86.20**	**77.53**	**77.61**	**77.29**	**76.01**	**74.63**	75.90	69.94	85.34	79.08
	META$_{BASE;SST}$	90.77	**82.51**	83.33	77.05	76.61	76.20	75.01	73.54	**76.80**	**70.49**	**85.78**	**79.46**

Table 4: Results on TREC13, WikiQA and SemEval-2016 datasets. Best results in each feature category are highlighted with bold, overall best results are underlined. TK is SST for Semeval and PTK for WikiQA and TREC13. BASE refers to the V+B$_{cr}$+E$_{cr}$+E configuration.

will refer to it as *BASE*), outperforms the standalone domain-specific handcrafted cQA features, V, and both PTK and SST on SemEval 2016 TEST and DEV by at least 2.3 points in all metrics.

Moreover, V+B$_{cr}$+E$_{cr}$+E is only less than 0.5 points behind the #1 system of the SemEval-2016 competition (see Tab. 7). We recall that V+B$_{cr}$+E$_{cr}$+E only uses basic n-gram overlap-based cross- and intra- similarity features and embedding-based cosine similarities.

Finally, when we add tree kernel models to the combination, i.e., V+B$_{cr}$+E$_{cr}$+E+PTK or V+B$_{cr}$+E$_{cr}$+E+SST, we note improvement for SemEval and TREC13 tasks.

Ensemble models. We ensemble cross- and intra-pair kernels-based models by summing the predictions of the standalone SVM classifiers (SUM models) or by training a logistic regression meta-classifier on them (META models). We build the meta-classifiers on the outputs of the standalone system *BASE* and TKs, namely PTK and SST. The "Ensemble" section of Table 4 shows that meta-system combinations mostly outperform the standalone kernels.

In general, combining cross-pair and intra-pair similarities (with kernel sum or meta-classifiers) provides state-of-the-art results without using deep learning. Additionally, the outcome is deterministic, while the DNN accuracy may vary depending on the type of the hardware used or the random initialization parameters (Crane, 2018).

5.5 Comparison with the state of the art

Tables 5, 6 and 7 report the performance of the most recent state-of-the-art systems on WikiQA, TREC13 and SemEval in comparison with our best results. We discuss them with respect to the different datasets.

WikiQA. As already mentioned earlier, WikiQA contains many questions without correct answer (see Tab. 3). When evaluated on the full data, even the oracle system will achieve at most 38.38 points of MAP. Moreover, as originally observed in (Wang et al., 2007), the questions that do not have either correct answers or incorrect answers are not useful for comparing the performance of different answer sentence selection systems. Therefore, they are typically removed from WikiQA and TREC13 before the evaluation.

There has been some discrepancy in the community when evaluating on WikiQA. The original baselines proposed for the corpus in (Yang et al., 2015) were evaluated in the "clean" setting[15]. We

---

[15] According to the WikiQA gold reference files at `https://www.microsoft.com/en-us/download/details.aspx?id=52419`

	no all$^-$		clean	
	MRR	**MAP**	**MRR**	**MAP**
LCLR (Yang et al., 2015) impl. of (Yih et al., 2013)	*61.83*	*60.92*	60.86	59.93
HybridTK-NN (Tymoshenko et al., 2017a)	*74.72*	*72.88*	74.08	72.19
IWAN-att (Shen et al., 2017)	75.00	73.30	*74.37*	*72.62*
C/A, MULT (Wang and Jiang, 2017)	75.45	74.33	*74.83*	*73.68*
C/A, k-max (Bian et al., 2017)	76.40	75.40	*75.80*	*74.78*
C/A, listwise (Bian et al., 2017)	75.90	74.60	*75.29*	*73.96*
HyperQA (Tay et al., 2018)	72.70	71.20	*72.01*	*70.47*
Our model (PTK)	76.21	75.29	75.60	74.67
Our model (SUM$_{BASE;PTK}$)	77.57	76.19	**77.00**	**75.59**

Table 5: Comparison to the SoA on WikiQA

	MRR	**MAP**
Noise-contrastive estim. (Rao et al., 2016)	87.7	80.1
IWAN-att (Shen et al., 2017)	88.9	**82.2**
BIMPM (Wang et al., 2017)	87.5	80.2
C/A, k-threshold (Bian et al., 2017)	**89.9**	82.1
C/A-listwise (Bian et al., 2017)	**88.9**	81.0
HyperQA (Tay et al., 2018)	86.5	78.4
Our model (B$_{cr}$+E$_{cr}$+PTK)	85.07	77.18
Our model (V+E$_{cr}$)	85.65	76.72
Our model (SUM$_{BASE;PTK}$)	**86.89**	**77.65**

Table 6: Comparison to the SoA on TREC13

	MRR	**MAP**
Kelp [#1] (Filice et al., 2016)	86.42	79.19
Conv-KN [#2] (Barrón-Cedeño et al., 2016)	84.93	77.6
CTK$_C$+V$_{QF}$ (Tymoshenko et al., 2016b)	86.26	78.78
HyperQA (Tay et al., 2018)	n/a	79.5
AI-CNN (Zhang et al., 2017)	n/a	80.14
Our model (V+B$_{cr}$+E$_{cr}$+E+SST)	**86.52**	**79.79**

Table 7: Comparison to the SoA on SemEval-2016

also evaluate in the "clean" setting. However, the performance of the most recent state-of-the-art systems listed in the Tab. 5 is reported in the "no all$^-$" setting, in the respective papers, i.e., they keep the all$^+$ questions[16]. Thus, they have 6 extra questions always answered correctly by default. To account for this discrepancy, in Tab. 5, we report the results in both settings. It is trivial to convert the performance figures from one setting to another. In the table, we mark the conversion results with italic.

Our SUM$_{BASE;PTK}$ system (i) outperforms all the state-of-the-art systems, including the sophisticated architectures with attention, such as IWAN-att (Shen et al., 2017), and compare-aggregate (C/A) frameworks (Wang and Jiang, 2017; Bian et al., 2017) in terms of MRR; and (ii) has the same MAP as (Bian et al., 2017). Obviously, this improvement is not statistically significant with respect to C/A systems by Bian et al. (2017). Nevertheless, ours is a very promising result, considering that we only use linear models with simple kernels and do not tune any learning parameter of such models.

TREC13. As shown in Tab. 6, our models do not outperform the state of the art on TREC13, but they still perform comparably to the recent DNN HyperQA model (Tay et al., 2018). In general, our model is behind the state-of-the-art IWAN-att system by 4.55 points in terms of MAP. Note, however, that TREC13 test set contains only 68 questions, therefore this difference in performance is not likely to be statistically significant[17].

Semeval. Table 7 compares performance of B_{cr} + E_{cr} + V + E + SST system on Semeval to that of KeLP and ConvKN, the two top systems in the SemEval 2016 competition, and also to the performance of the recent DNN-based HyperQA and AI-CNN systems. In the Semeval 2016 competition, our model would have been the first[18], with #1 KeLP system being 0.6 MAP points behind. Then, it would have outperformed the state-of-the-art AI-CNN system by 0.35 MAP points.

6 Conclusions

This work proposes a simple, yet effective approach to the task of answer sentence selection based on the intuition that similar patterns in questions are likely to demand similar patterns in answers. We showed that this hypothesis provides an improvement on three benchmark datasets, WikiQA, TREC13, Semeval-2016, and, moreover, it enables simple features to achieve the state of the art on WikiQA and Semeval-2016, outperforming many of state-of-the-art DNN-based systems. There is significant room for further elaboration of this approach, for example, by expanding feature spaces with more syntactic and semantic features, employing new types of kernels for measuring the inter-question/answer pair similarity or trying to implement the same idea in DNN architectures.

Acknowledgments

The first author is supported by Adeptmind. Many thanks to the Area chairs, PC chairs and anonymous reviewers for their professional work and valuable suggestions.

---

[16] We deduced that from the corpus statistics reported by the authors of the papers. They all report having 243 test questions, which corresponds to the "no all$^-$" setting

[17] We cannot conduct the test since we do not have the outputs of such systems. However, 4 points more correspond to just a couple of correct questions more.

[18] Ranking available in (Nakov et al., 2016)

References

D Bär, C Biemann, I Gurevych, and T Zesch. 2012. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. *First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*, pages 435–440.

Daniel Bär, Torsten Zesch, and Iryna Gurevych. 2013. DKPro Similarity: An Open Source Framework for Text Similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 121–126, Sofia, Bulgaria. Association for Computational Linguistics.

A. Barrón-Cedeño, D. Bonadiman, G. Da San Martino, S. Joty, A. Moschitti, F.A. Al Obaidli, S. Romeo, K. Tymoshenko, and A. Uva. 2016. ConvKN at SemEval-2016 task 3: Answer and question selection for question answering on Arabic and English fora. In *SemEval 2016 - 10th International Workshop on Semantic Evaluation, Proceedings*.

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A Compare-Aggregate Model with Dynamic-Clip Attention for Answer Selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1987–1990, New York, NY, USA. ACM.

Andrei Z Broder. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE.

Matt Crane. 2018. Questionable Answers in Question Answering Research: Reproducibility and Variability of Published Results. *Transactions of the Association for Computational Linguistics*, 6:241–252.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*, volume 71.

Simone Filice, Giuseppe Castellucci, Danilo Croce, Giovanni Da San Martino, Alessandro Moschitti, and Roberto Basili. KeLP: a Kernel-based Learning Platform in Java, year = 2015. In *The workshop on Machine Learning Open Source Software (MLOSS): Open Ecosystems*, Lille, France. International Conference of Machine Learning.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1116–1123.

Michael Heilman and Noah a. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. *The 2010 Annual Conference of the North American Chapter of the ACL*, pages 1011–1019.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text Classification using String Kernels. *Journal of Machine Learning Research*, 2:419–444.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 1–9.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. *Machine Learning ECML 2006*, 4212:318–329.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 Task 3: Community Question Answering. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 525–545.

Massimo Nicosia, Simone Filice, Alberto Barrón-Cedeno, Iman Saleh, Hamdy Mubarak, Wei Gao, Preslav Nakov, Giovanni Da San Martino, Alessandro Moschitti, Kareem Darwish, and Others. 2015. QCRI: Answer selection for community question answering-experiments for Arabic and English. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 203–209.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2012. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-Contrastive Estimation for Answer Selection with Deep Neural Networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM '16*, pages 1913–1916.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '12*, pages 741–750.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic Feature Engineering for Answer Selection and Extraction. In *Proc. Empirical Methods in Natural Language Processing*, October, pages 458–467.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 373–382, New York, NY, USA. ACM.

Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. *CoRR*, abs/1604.01178.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning adaptable patterns for passage reranking. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL)*, pages 75–83. Association for Computational Linguistics.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-Weighted Alignment Network for Sentence Pair Modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189. Association for Computational Linguistics.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Hyperbolic Representation Learning for Fast and Efficient Neural Question Answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 583–591, New York, NY, USA. ACM.

K. Tymoshenko, D. Bonadiman, and A. Moschitti. 2016a. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*.

K. Tymoshenko, D. Bonadiman, and A. Moschitti. 2016b. Learning to rank non-factoid answers: Comment selection in Web forums. In *CIKM*.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2017a. Ranking Kernels for Structures and Embeddings: A Hybrid Preference and Classification Model. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. Association for Computational Linguistics.

Kateryna Tymoshenko, Alessandro Moschitti, Massimo Nicosia, and Aliaksei Severyn. 2017b. RelTextRank: An Open Source Framework for Building Relational Syntactic-Semantic Text Pair Representations. In *Proceedings of ACL 2017, System Demonstrations*, pages 79–84. Association for Computational Linguistics.

Bingning Wang, Kang Liu, and Jun Zhao. 2016a. Inner Attention based Recurrent Neural Networks for Answer Selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1288–1297.

Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. pages 707–712.

Mengqiu Wang and Christopher D Manning. 2010. Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering. *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, (August):1164–1172.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 22–32.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. *ICLR*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 4144–4150.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence Similarity Learning by Lexical Decomposition and Composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1340–1349, Osaka, Japan. The COLING 2016 Organizing Committee.

Michael J Wise. 1996. YAP3: Improved detection of similarities in computer program and other texts. *ACM SIGCSE Bulletin*, 28(1).

Yi Yang, Wen-Tau Yih, and Christopher Meek. 2015. WIKIQA: A Challenge Dataset for Open-Domain Question Answering. *Proceedings of EMNLP 2015*, pages 2013–2018.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer Extraction as Sequence Tagging with Tree Edit Distance.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models. pages 1744–1753.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. *NIPS Deep Learning and Representation Learning Workshop*.

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic Learning of Textual Entailments with Cross-Pair Similarities. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 401–408, Sydney, Australia. Association for Computational Linguistics.

Xiaodong Zhang, Sujian Li, Lei Sha, and Houfeng Wang. 2017. Attentive Interactive Neural Networks for Answer Selection in Community Question Answering. *AAAI*, 24(4):505–509.

🦆 QuAC : Question Answering in Context

Eunsol Choi[★♡] **He He**[★♢] **Mohit Iyyer**[★♣†] **Mark Yatskar**[★†]
Wen-tau Yih[†] **Yejin Choi**[♡†] **Percy Liang**[♢] **Luke Zettlemoyer**[♡]

Allen Institute for Artificial Intelligence[†] University of Washington[♡]
Stanford University[♢] UMass Amherst[♣]

`{eunsol,yejin,lsz}@cs.washington.edu`
`{hehe,pliang}@cs.stanford.edu`
`{mohiti,marky,scottyih}@allenai.org`

Abstract

We present QuAC, a dataset for **Qu**estion **A**nswering in **C**ontext that contains 14K information-seeking QA dialogs (100K questions in total). The dialogs involve two crowd workers: (1) a student who poses a sequence of freeform questions to learn as much as possible about a hidden Wikipedia text, and (2) a teacher who answers the questions by providing short excerpts from the text. QuAC introduces challenges not found in existing machine comprehension datasets: its questions are often more open-ended, unanswerable, or only meaningful within the dialog context, as we show in a detailed qualitative evaluation. We also report results for a number of reference models, including a recently state-of-the-art reading comprehension architecture extended to model dialog context. Our best model underperforms humans by 20 F1, suggesting that there is significant room for future work on this data. Dataset, baseline, and leaderboard available at `http://quac.ai`.

1 Introduction

In information-seeking dialog, students repeatedly ask teachers questions to learn about a topic of interest (Stede and Schlangen, 2004). Modeling such conversations is challenging, as the questions can be highly context-dependent, elliptical, and even unanswerable. To enable learning from rich information-seeking dialog, we present QuAC (henceforth 🦆), a large-scale dataset for **Qu**estion **A**nswering in **C**ontext that contains 14K crowdsourced QA dialogs (100K total QA pairs).[1]

Figure 1 shows an example 🦆 dialog. The interaction is student driven and centered around a short evidence text (a section from Daffy Duck's Wikipedia page), which only the teacher can access. Given just the section's heading, "Origin & History", the student aims to learn as much as possible about its contents by asking questions. The teacher answers these questions with spans from the evidence text, as in existing reading comprehension tasks (Rajpurkar et al., 2016). Additionally, the teacher uses dialog acts to provide the student with feedback (e.g., "ask a follow up ques-

Section: 🦆Daffy Duck, Origin & History

STUDENT: **What is the origin of Daffy Duck?**

TEACHER: ↪ first appeared in Porky's Duck Hunt

STUDENT: **What was he like in that episode?**

TEACHER: ↪ assertive, unrestrained, combative

STUDENT: **Was he the star?**

TEACHER: ⇢ No, barely more than an unnamed bit player in this short

STUDENT: **Who was the star?**

TEACHER: ↛ No answer

STUDENT: **Did he change a lot from that first episode in future episodes?**

TEACHER: ↪ Yes, the only aspects of the character that have remained consistent (...) are his voice characterization by Mel Blanc

STUDENT: **How has he changed?**

TEACHER: ↪ Daffy was less anthropomorphic

STUDENT: **In what other ways did he change?**

TEACHER: ↪ Daffy's slobbery, exaggerated lisp (...) is barely noticeable in the early cartoons.

STUDENT: **Why did they add the lisp?**

TEACHER: ↪ One often-repeated "official" story is that it was modeled after producer Leon Schlesinger's tendency to lisp.

STUDENT: **Is there an "unofficial" story?**

TEACHER: ↪ Yes, Mel Blanc (...) contradicts that conventional belief

...

Figure 1: An example dialog about a Wikipedia section. The student, who does not see the section text, asks questions. The teacher provides a response in the form of a text span (or No answer), optionally yes or no (Yes / No), and encouragement about continuing a line of questioning (should, ↪, could ⇢, or should not ↛ ask a follow-up question).

---

[1] We use "dialog" to refer to a sequence of QA pairs.
★ Authors contributed equally.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2174–2184
Brussels, Belgium, October 31 - November 4, 2018.

Dataset	Multi turn	Text-based	Dialog Acts	Simple Evaluation	Unanswerable Questions	Asker Can't See Evidence
QuAC	✓	✓	✓	✓	✓	✓
CoQA (Reddy et al., 2018)	✓	✓	✗	✓	✓	✗
CSQA (Saha et al., 2018)	✓	✗	✗	✗	✓	✗
CQA (Talmor and Berant, 2018)	✓	✓	✗	✓	✗	✓
SQA (Iyyer et al., 2017)	✓	✗	✗	✓	✗	✗
NarrativeQA (Kociský et al., 2017)	✗	✓	✗	✗	✗	✓
TriviaQA (Joshi et al., 2017)	✗	✓	✗	✓	✗	✓
SQuAD 2.0 (Rajpurkar et al., 2018)	✗	✓	✗	✓	✓	✗
MS Marco (Nguyen et al., 2016)	✗	✓	✗	✗	✓	✓
NewsQA (Trischler et al., 2016)	✗	✓	✗	✓	✓	✓

Table 1: Comparison of the QUAC dataset to other question answering datasets.

tion"), which makes the dialogs more productive.

We collect the dataset in an interactive setting where two crowd workers play the roles of teacher and student. To encourage natural and diverse questions, we do not follow previous dialog-style QA datasets that semi-automatically generate questions (Talmor and Berant, 2018; Saha et al., 2018). Furthermore, unlike QA datasets such as SQuAD and CoQA (Reddy et al., 2018), students in do not know the answers to their questions prior to asking them, which lessens the role of string matching and simple paraphrasing in answering their questions. This property makes similar to datasets that contain real user queries on search engines (Nguyen et al., 2016).

contains many challenging phenomena unique to dialog, such as coreference to previous questions and answers and open-ended questions that must be answered without repeating previous information (Section 3). Additionally, despite lacking access to the section text, we find that students start dialogs by asking questions about the beginning of the section before progressing to asking questions about the end. These observations imply that models built for must incorporate the dialog context to achieve good performance.

We present a strong neural baseline (Clark and Gardner, 2018) that considers both dialog context and section text. While this model achieves within 6 F1 of human performance on SQuAD, it performs 20 F1 points below the human upper bound on , indicating room for future improvement.

2 Dataset collection

This section describes our data collection process, which involves facilitating QA dialogs between crowd workers. Table 1 shows shares many of the same positive characteristics of existing QA datasets while expanding upon the dialog aspect.

	Train	Dev.	Test	Overall
questions	83,568	7,354	7,353	98,407
dialogs	11,567	1,000	1,002	13,594
unique sections	6,843	1,000	1,002	8,854
tokens / section	396.8	440.0	445.8	401.0
tokens / question	6.5	6.5	6.5	6.5
tokens / answer	15.1	12.3	12.3	14.6
questions / dialog	7.2	7.4	7.3	7.2
% yes/no	26.4	22.1	23.4	25.8
% unanswerable	20.2	20.2	20.1	20.2

Table 2: Statistics summarizing the dataset.

2.1 Interactive Task

Our task pairs up two workers, a *teacher* and a *student*, who discuss a section s (e.g., "Origin & History" in the example from Figure 1) from a Wikipedia article about an entity e (Daffy Duck). The student is permitted to see only the section's title t and the first paragraph of the main article b, while the teacher is additionally provided with full access to the section text.

The task begins with the student formulating a free-text question q from the limited information they have been given. The teacher is not allowed to answer with free text; instead, they must select a contiguous span of text defined by indices (i, j) into the section text s.[2] While this decision limits the expressivity of answers, it makes evaluation simpler and more reliable; as such, it has been adopted in other reading comprehension datasets such as SQuAD, TriviaQA (Joshi et al., 2017), and NewsQA (Trischler et al., 2016).

To facilitate more natural interactions, teachers must also provide the student with a list of dialog acts v that indicates the presence of any of n discrete statements. We include three types of di-

---

[2] We set the maximum answer length to 30 tokens to prevent teachers from revealing the full article all at once.

Figure 2: A treemap visualization of the eight most frequent "Wh" words in , where box area is proportional to number of occurrences. Compared to other machine comprehension datasets, we observe increased contextuality and open-endedness, as well as a variety of both general and specific questions.

alog acts: (1) continuation (`follow up`, `maybe follow up`, or `don't follow up`), (2) affirmation (`yes`, `no`, or `neither`) and (3) answerability (`answerable` or `no answer`). The continuation act is crucial for workers to have productive dialogs, as it allows teachers to guide the student's questioning towards aspects of the article that are especially important or interesting. Altogether, a teacher's complete answer to a question q includes a pair of indices and dialog indicators, $a = (i, j, v)$. If a question is marked `no answer`, the indices are ignored.

After receiving an answer from the teacher, the student asks another question. At every turn, the student has more information about the topic than they did previously, which encourages them to ask follow-up questions about what they have just learned. The dialog continues until (1) twelve questions are answered, (2) one of the partners decides to end the interaction, or (3) more than two unanswerable questions were asked.

2.2 Collection Details

We used Amazon Mechanical Turk for collection, restricting the task to workers in English-speaking countries and with more than 1000 HITs with at least a 95% acceptance rate. We paid workers per the number of completed turns in the dialog, which encourages workers to have long dialogs with their partners, and discarded dialogs with less than three QA pairs.[3] To ensure quality, we created a qualification task and allowed workers to report their partner for various problems. More details on data collection can be found in our datasheet.[4]

Article selection Our early pilot studies showed that articles about people generally require less background knowledge to write good questions than other categories. To find articles about people with varied backgrounds, we retrieved articles from a list of category keywords (culture, animal, people associated with event, geography, health, celebrity) using a web interface provided by the Wikimedia foundation.[5] We pruned by popularity by selecting articles with at least 100 incoming links, and we additionally removed non-person entities using YAGO (Suchanek et al., 2007). After article selection, we filtered sections from these articles based on the number of paragraphs, number of tokens, and average words per sentence. [6]

Dataset validation To create our evaluation sets, we collected four additional annotations per question. Workers were presented with questions from a previously collected dialog and asked to

---

[3] On average, we paid $0.33 per question, increasing pay per question as dialogs got longer to encourage completion.

[4] `http://quac.ai/datasheet.pdf`

[5] https://petscan.wmflabs.org/

[6] These filtering steps bias our data towards entertainers; see datasheet for details.

provide answer spans.[7] Acquiring many annotations is important since many questions in 🦆 have multiple valid answers.

Train / Dev / Test Differences Table 2 shows small differences between training, development and testing splits. Sections in the training set are shorter than those in the evaluation folds because we permit multiple dialogs about the same section only in training; since workers preferred reading shorter sections, these were more likely to result in multiple dialogs. Variations in answer span length arise from two sources: (1) having multiple annotations in the validation task and (2) differing incentives between the data collection and validation procedures.[8] An analysis measuring the effect of these variations shows that they result in little difference in evaluation.[9]

3 Dataset Analysis

🦆 differs from other reading comprehension datasets due to our dialog-style collection process and the information asymmetry between teacher and student. In the following sections, we provide a qualitative analysis of the dataset in 🦆 that highlights challenging question types as well as the impact of the dialog context.

Question and answer types Table 2 shows dataset summary statistics. 🦆 has long answers of 15 tokens on average compared to 3 for SQuAD, which is unsurprising as most SQuAD answers are either entities or numerics (Jurczyk et al., 2018) while 🦆 questions can be more open-ended. While the average question length (6.5 tokens) is shorter than that of SQuAD (11 tokens), this does not indicate reduced question complexity, as the student (1) cannot access the section to paraphrase it and (2) can be more concise by coreferencing previous interactions.

Section: Augusto Pinochet : Intellectual life...

STUDENT: **Was he known for being intelligent?**
TEACHER: $\hookrightarrow$ No, Pinochet was publicly known as a man with a lack of culture.
STUDENT: **why did people feel that way?**
TEACHER: $\hookrightarrow$ reinforced by the fact that he also portrayed himself as a common man
STUDENT: **did he have any hobbies?**
TEACHER: $\hookrightarrow$ Yes, Before wresting power from Allende, Pinochet had written two books.
STUDENT: **what is the name of a book written by him?**
TEACHER: $\hookrightarrow$ Geopolitica (1968) and Campana de Tarapaca (1972).
STUDENT: **what were the books about?**
TEACHER: $\hookrightarrow$ Chile's military literature.
STUDENT: **was there anything noteworthy regarding his books?**
TEACHER: $\hookrightarrow$ Yes, In Geopolitica Pinochet plagiarized (...) Gregorio Rodriguez Tascon
STUDENT: **did he deny those allegations?**
TEACHER: $\not\hookrightarrow$ No answer
STUDENT: **what did he plagiarize in Geopolitica?**
TEACHER: $\bar{\hookrightarrow}$ In Geopolitica Pinochet plagiarized (...) paragraphs from a 1949 presentation
. . .

Figure 3: An example successful dialog from 🦆. Questions build on each other and interesting aspects (e.g., plagiarism) are explored as they are discovered.

Figure 2 visualizes the most frequent question types in 🦆 based on "Wh" words.[10] For a more fine-grained analysis, we randomly sampled 100 questions (each from a different dialog) and manually labeled different phenomena in Table 3. Unlike most current QA datasets that focus on factoid questions, our task setup encourages more open-ended questions: about half of 🦆 questions are non-factoid. Furthermore, 86% of questions are contextual, requiring reading the context to resolve coreference; of these, 44% refer to entities or events in the dialog history, while 61% refer to the subject of the article.

The role of context Dialog context is crucial to understanding and answering 🦆 questions. Figure 5a shows that the location of the answer within the text is influenced by the number of questions asked previously. Early questions are mostly answered in the beginning of the section, while later questions tend to focus on the end of the section. Interestingly, text in the middle of the section is not asked about as frequently (Figure 5c). As more questions get asked, the more likely a question is to be unanswerable.

Figure 5b shows how the answers progress through different chunks of the evidence text (where each section is divided into 12 chunks of

---

[7] After submitting an answer, they were shown the original teacher's answer so that they could understand the context of the subsequent questions.

[8] Validation workers did not have to maintain the dialog and so did not include as much information in the response.

[9] More specifically, we analyze whether references from the initial data collection significantly differ from references collected during validation. We observe a difference of less than 1 F1 when using the original answer as system output versus using validation answers.

[10] To more effectively visualize sub-boxes like "what did", we exclude questions from the tail of the distribution.

Section: Gaelic Ireland : Invasion

STUDENT: **What year did the invasion happen?**
TEACHER: ↪ in 1169 the main body of Norman, Welsh and Flemish forces landed in Ireland and quickly retook Leinster and the cities of Waterford and Dublin on behalf of Diarmait.
STUDENT: **Who was Diarmait?**
TEACHER: ⇢ King Diarmait Mac Murchada of Leinster.
STUDENT: **Where is Leinster located?**
TEACHER: ↛ landed in Ireland and quickly retook Leinster.
STUDENT: **Were invasions common?**
TEACHER: ↛ No answer
STUDENT: **Are there any other interesting aspects about this article?**
TEACHER: ⇢ Yes, IPope Adrian IV, the only English pope, had already issued a Papal Bull in 1155 giving Henry II of England authority to invade Ireland.
STUDENT: **Who lead the invasion?**
TEACHER: ↛ No answer
STUDENT: **Did England defeat the Irish armies?**
TEACHER: ↛ No answer

Figure 4: A less successful dialog from . The student struggles to get information despite asking good questions. The teacher attempts to provide extra context to guide the student, but the dialog ultimately ends because of too many unanswerable questions.

Question type	%	Example
Non-factoid	54	Q: **Were** the peace talks **a success?** Q: **What was** her childhood **like?**
Contextual	86	
Coref (article)	61	Title: Paul Cézanne: Early years Q: When did **he** start painting?
Coref (history)	44	Q: What was special about the Harrah's? A: project was built by Trump with financing from the Holiday Corporation. Q: **Which led to what?**
Anything else?	11	Q: What **other acting** did he do? Q: **What else** did he research?

Table 3: An analysis of questions. **Non-factoid** questions do not ask about specific facts, while **contextual** questions require reading the history to resolve coreferences to the dialog history and/or article.

equal size). The answer to the next question is most frequently either in the same chunk as the previous question or an adjacent chunk, and most dialogs in the dataset cover three to six of the chunks (Figure 5d). These observations suggest that models for must take into account the dialog context. However, results in Section 5 show that solely relying on the location of previous answers is not sufficient.

Finally, we examine properties of the questions as a function of the turn position in the dialog (Figure 6). The frequency of yes/no questions increases significantly as the dialogs progress; again, at the beginning of the dialog, students have very little information, so it is harder to formulate a yes/no question. The percentage of questions that have multiple answers declines as the dialog progresses, implying students ask general questions first and specific ones later.

Qualitative examples Figures 3 and 4 contain two representative dialogs from . Longer dialogs sometimes switch topics (such as in Figure 3 about "academic work") and often go from general to specific questions. Students whose questions go unanswered commonly resort to asking their teacher for any interesting content; even if this strategy fails to prolong the dialog as in Figure 4, models can still use the dialog to learn when to give no answer.

4 Experimental Setup

We consider the following QA task: given the first k questions and k ground-truth answers in the dialog, all supporting material (entity e, topic t, background b, and section text s), and question q_{k+1}, we predict the answer span indices i, j in the section text s. Since affirmation questions are incomplete without a yes/no answer and the continuation feedback is important for information-seeking dialog, we predict the dialog acts v, which with the span form the final answer prediction a_{k+1}.

All of our experiments are carried out on a train/dev/test split of 83.5k/7.3k/7.3k questions/answer pairs, where no sections are shared between the different folds. Questions in the training set have one reference answer, while dev and test questions have five references each. For all experiments, we do not evaluate on questions with a human F1 lower than 40, which eliminates roughly 10% of our noisiest annotations.[11]

4.1 Evaluation Metrics

Our core evaluation metric, word-level F1, is implemented similarly to SQuAD (Rajpurkar et al.,

---

[11] A manual inspection of annotations below this threshold revealed many lower quality questions; however, we also report unthresholded F1 in the final column of Table 4.

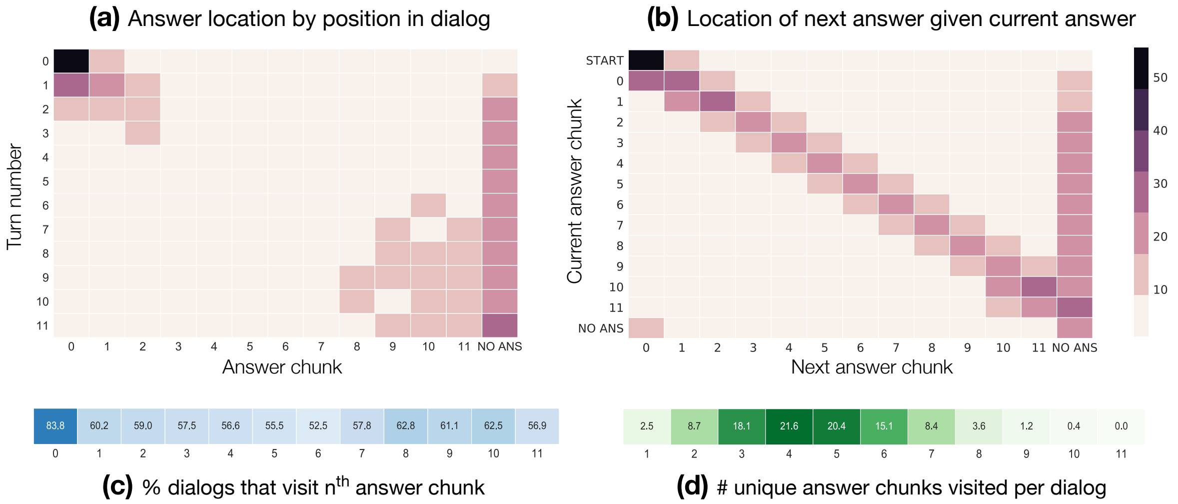

(a) Answer location by position in dialog

(b) Location of next answer given current answer

(c) % dialogs that visit nth answer chunk

(d) # unique answer chunks visited per dialog

Figure 5: Heatmaps depicting the importance of context in 🦆 dialogs, where (a) and (b) share the same color scale. The student's earlier questions are answered mostly by the first few chunks, while the end of the section is covered in later turns (a). The middle is the least covered portion (c), and dialogs cover around five unique chunks of the section on average (d). The transition matrix (b) shows that the answer to the next question is more likely to be located within a chunk adjacent to the current answer than in one farther away.

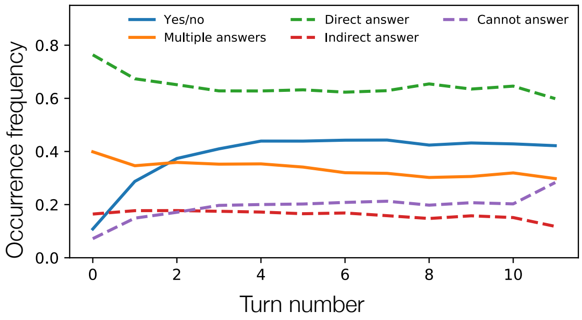

Figure 6: The number of turns in the dialog influences the student's behavior: they start by asking general questions (i.e., easier to answer, with multiple possible answers) and progress to more specific ones.

2016): precision and recall are computed by considering the portion of words in the prediction and references that overlap after removing stopwords.[12] For `no answer` questions, we give the system an F1 of one if it correctly predicts `no answer` and zero otherwise.[13] Like SQuAD, we compute the maximum F1 among all references; however, since many 🦆 questions have multiple valid answers, this metric varies significantly with the number of reference annotations. To make oracle human and system performance comparable, given n references, we report the average of the maximum F1 computed from each $n-1$ subset with respect to the heldout reference.

Additionally, since averaged F1 can be misleading for questions with multiple valid answers, we introduce the human equivalence score (HEQ), a performance measure for judging whether a system's output is as good as that of an average human.[14] HEQ measures the percentage of examples for which system F1 exceeds or matches human F1. We compute two variants: (1) the percentage of questions for which this is true (HEQ-Q), and (2) the percentage of dialogs for which this is true for every question in the dialog (HEQ-D). A system that achieves a value of 100 on HEQ-D can by definition maintain average human quality output over full dialogs.

For dialog acts, we report accuracy with respect to the majority annotation, breaking ties randomly.

5 Experiments

5.1 Sanity checks

Random sentence This baseline selects a random sentence in the section text s as the answer (including `no answer`).

---

[12] Since our answer spans have vaguer boundaries than the shorter ones in SQuAD, exact match is not a useful metric.

[13] Because the validation task was more susceptible to spam by constant annotation of "no-answer," we only allow "no-answer" if the majority of references marked "no-answer", removing other answers. If "no-answer" is not the majority answer, we remove all instances of "no-answer".

[14] In cases with lower human agreement on F1, if a system produces one reference exactly (F1 = 100), it will get points that it can use to offset poor performance on other examples.

Majority The majority answer outputs `no answer` and the majority class for all other dialog acts (`neither` for affirmation and `don't follow up` for continuation).

Transition matrix We divide the supporting text into 12 chunks (with a special chunk for `no answer`) and use the transition matrix (computed from the training set) in Figure 5b to select an answer given the position of the previous answer. This baseline does not output other dialog acts.

5.2 Upper bounds

Gold NA + TM This is the same transition matrix (TM) baseline as before, except that for questions whose gold annotations are `no answer`, we always output `no answer`.

Gold sentence + NA To see if 🦆 can be treated as an answer sentence selection problem, we output the sentence from s with the maximal F1 with respect to references, or `no answer` for unanswerable questions.

Human performance We pick one reference as a system output and compute the F1 with respect to the remaining references using the method described in Section 4.1. By definition, all HEQ measures are 100, and we report agreement for the affirmation dialog act.[15]

5.3 Baselines

Pretrained InferSent To test the importance of lexical matching in our dataset, we output the sentence in s whose pretrained InferSent representation (Conneau et al., 2017) has the highest cosine similarity to that of the question.

Feature-rich logistic regression We train a logistic regression using Vowpal Wabbit (Langford et al., 2007) to select answer sentences. We use simple matching features (e.g., n-gram overlap between questions and candidate answers), bias features (position and length of a candidate), and contextual features (e.g., matching features computed with previous questions / answers, turn number).

BiDAF++ We use a re-implementation of a top-performing SQuAD model (Peters et al., 2018) that augments bidirectional attention flow (Seo et al., 2016, BiDAF) with self-attention (Clark and Gardner, 2018) and contextualized embeddings.[16]

A token for `no answer` is appended to s to enable its prediction following Levy et al. (2017). Additionally, we modify the model for our task to also predict dialog acts, placing a classifier over the same representation used to predict the end position of the predicted span.

BiDAF++ w/ k-ctx As BiDAF++ does not model any dialog context, we modify the passage and question embedding processes to consider the dialog history. We consider context from the previous k QA pairs.[17]

- **Passage embedding** We explicitly identify the previous k answers within the section text by concatenating marker embeddings to the existing word embeddings.

- **Question embedding** Naively prepending the previous k questions to the current question did not show gains in initial experiments. We opt instead to simply encode the dialog turn number within the question embedding.

5.4 Results

Table 4 summarizes our results (each cell displays dev/test scores), where dialog acts are Yes/No (affirmation) and Follow up (continuation). For comparison to other datasets, we report F1 without filtering low-agreement QA pairs (F1').

Sanity check Overall, the poor sanity check results imply that 🦆 is very challenging. Of these, following the transition matrix (TM) gives the best performance, reinforcing the observation that the dialog context plays a significant role in the task.

Upper bounds The human upper bound (80.8 F1) demonstrates high agreement. While Gold sentence + NA does perform well, indicating that significant progress can be made by treating the problem as answer sentence selection, HEQ measures show that span-based approaches will be needed achieve average human equivalence. Finally, the Gold NA + TM shows that 🦆 cannot be solved by ignoring question and answer text.

---

[15] We did not collect multiple annotations for the continuation dialog act and so omit it.

[16] The AllenNLP (Gardner et al., 2017) implementation we use reaches 82.7 on the SQuAD development set, compared to the paper's reported 85.8 on SQuAD; regardless, this implementation would have been state-of-the-art less than a year ago, making it an extremely strong baseline.

[17] Our implementation is available in AllenNLP.

	F1	HEQ-Q	HEQ-D	Yes / No	Follow up	F1 (All)
Random sentence	15.7 / 15.6	6.9 / 6.9	0.0 / 0.1	—	—	16.4 / 16.3
Majority answer	22.7 / 22.5	22.7 / 22.5	0.5 / 0.4	78.8 / 77.6	57.9 / 56.7	20.2 / 20.0
Trans. matrix (TM)	31.8 / 31.5	15.8 / 15.8	0.1 / 0.2	—	—	31.2 / 30.9
Pretrained InferSent	21.4 / 20.8	10.2 / 10.0	0.0 / 0.0	—	—	22.0 / 21.4
Logistic regression	34.3 / 33.9	22.4 / 22.2	0.6 / 0.2	—	—	34.3 / 33.8
BiDAF++ (no ctx)	51.8 / 50.2	45.3 / 43.3	2.0 / 2.2	86.4 / 85.4	59.7 / 59.0	50.1 / 48.2
BiDAF++ (w/ 1-ctx)	59.9 / 59.0	54.9 / 53.6	4.7 / 3.4	86.5 / **86.1**	61.3 / 60.3	57.5 / 56.5
BiDAF++ (w/ 2-ctx)	**60.6 / 60.1**	**55.7 / 54.8**	**5.3** / 4.0	**86.6** / 85.7	**61.6 / 61.3**	**58.3 / 57.8**
BiDAF++ (w/ 3-ctx)	**60.6** / 59.5	55.6 / 54.5	5.0 / **4.1**	86.1 / 85.7	**61.6** / 61.2	58.1 / 57.0
Gold NA + TM	43.0 / 42.6	27.4 / 27.4	1.0 / 0.8	—	—	41.0 / 40.6
Gold sentence + NA	72.4 / 72.7	61.8 / 62.7	9.8 / 9.7	—	—	70.8 / 71.2
Human performance	80.8 / 81.1	100 / 100	100 / 100	89.4 / 89.0	—	74.6 / 74.7

Table 4: Experimental results of sanity checks (top), baselines (middle) and upper bounds (bottom) on . Simple text matching baselines perform poorly, while models that incorporate the dialog context significantly outperform those that do not. Humans outperform our best model by a large margin, indicating room for future improvement.

Baselines Text similarity methods such as bag-of-ngrams overlap and InferSent are largely ineffective on , which shows that questions have little direct overlap with their answers. On the other hand, BiDAF++ models make significant progress, demonstrating that existing models can already capture a significant portion of phenomena in . The addition of information from previous turns (w/ 1-ctx) helps significantly, indicating that integration of context is essential to solving the task. While increasing the context size in BiDAF++ continues to help, we observe saturation using contexts of length 3, suggesting that more sophisticated models are necessary to take full advantage of the context. Finally, even our best model underperforms humans: the system achieves human equivalence on only 60% of questions and 5% of full dialogs.

5.5 Error Analysis

In this section, we analyze the development set performance of our best context-aware model (BiDAF++ w/ 2-ctx), our best context-agnostic model (BiDAF++), and humans. Figure 7 contains three plots showing how F1 scores of baseline models and human agreement vary with (1) turn number, (2) distance from previous answer,[18] and (3) answer length in tokens. Taken as a whole, our analysis reveals significant qualitative differences between our context-aware and context-agnostic models beyond simply F1; additionally, human behavior differs from that of both models.

In the first plot, human agreement is unchanged throughout the dialog while the performance of both models decreases as the number of turns increases, although the context-aware model degrades less. While continuing a dialog for more turns does not affect human agreement, the second plot shows that human disagreement increases as the distance between the current answer's location within the section text and that of the previous answer increases. Larger distances indicate shifts in the student's line of questioning (e.g., if the teacher told the student not to follow up on the previous question). The plot also shows that model performance suffers (significantly more than humans) as distance increases, although the context-aware model can tolerate smaller shifts better than the context-agnostic model. In the last plot, human agreement is higher when the answer span is short; in contrast, our model struggles to pin down short answers compared to longer ones.

The plots demonstrate the increased robustness of the context-aware model compared to BiDAF++. This finding is reinforced by examining the difference in model performance on questions where previously the teacher recommended the student to "follow up" vs. not to follow up. The context-aware baseline performs 6 HEQ-Q higher on the "follow up" questions; in contrast, the context-agnostic baseline shows no HEQ-Q difference between the two types of questions. This discrepancy stems from the context-agnostic

---

[18] We divide the text into 12 equally-sized chunks and compute the difference of the current and previous chunk indices.

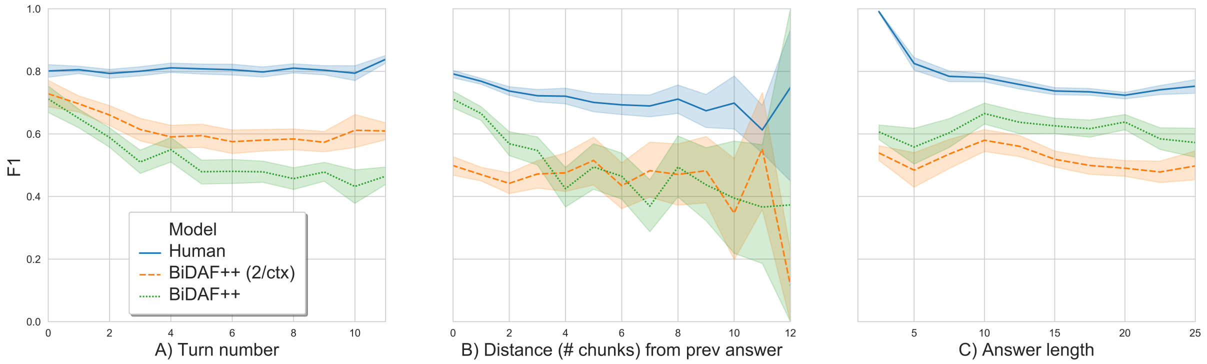

Figure 7: The F1 scores of baseline models and human agreements based on dialog turn number, answer's distance from previous answer, and the answer span token length.

model's inability to take advantage of the location of the previous answer.

6 Related Work

Reading Comprehension Our work builds on span based reading comprehension (Rajpurkar et al., 2016; Joshi et al., 2017; Trischler et al., 2016), while also incorporating innovations such as curating questions independently of supporting text to reduce trivial lexical overlap (Joshi et al., 2017; Kociský et al., 2017) and allowing for unanswerable questions (Trischler et al., 2016; Rajpurkar et al., 2018). We handle open-ended questions like in MSMARCO (Nguyen et al., 2016), with multiple references, but we are the first to incorporate these into information-seeking dialog.

Sequential QA Our work is similar to sequential question answering against knowledge bases (Iyyer et al., 2017) and the web (Talmor and Berant, 2018), but instead of decomposing a single question into smaller questions, we rely on the curiosity of the student to generate a sequence of questions. Such open information seeking was studied in semantic parsing on knowledge bases (Dahl et al., 1994) and more recently with modern approaches (Saha et al., 2018), but with questions paraphrased from templates. Concurrent to our work, Saeidi et al. (2018) proposed a task of generating and answering yes/no questions for rule focused text (such as traffic laws) by interacting with a user through dialog. Also concurrently, Reddy et al. (2018) propose conversational question answering (CoQA) from text but allow both students and questioners to see the evidence. As a result, a large percentage of CoQA answers are named entities or short noun phrases, much like those in SQuAD. In contrast, the asymmetric nature of 🦆 forces students to ask more exploratory questions whose answers can be potentially be followed up on.[19]

Dialog 🦆 fits into an increasing interest in open domain dialog, mostly studied in the context of social chit-chat (Li et al., 2016; Ritter et al., 2011; Fang et al., 2017; Ghazvininejad et al., 2018). Most related to our effort is visual dialog (Das et al., 2017), which relies on images as evidence instead of text. More explicit goal driven scenarios, such as bargaining (Lewis et al., 2017) and item guessing (He et al., 2017) have also been explored, but the language is more constrained than in 🦆. Information-seeking dialog specifically was studied in Stede and Schlangen (2004).

7 Conclusion

In this paper, we introduce 🦆, a large scale dataset of information-seeking dialogs over sections from Wikipedia articles. Our data collection process, which takes the form of a teacher-student interaction between two crowd workers, encourages questions that are highly contextual, open-ended, and even unanswerable from the text. Our baselines, which include top performers on existing machine comprehension datasets, significantly underperform humans on 🦆. We hope this discrepancy will spur the development of machines that can more effectively participate in information seeking dialog.

Acknowledgments

🦆 was jointly funded by the Allen Institute for Artificial Intelligence and the DARPA CwC program through ARO (W911NF-15-1-0543). We would like to thank anonymous reviewers and Hsin-Yuan Huang who helped improve the draft.

---

[19] On average, CoQA answers are 2.7 tokens long, while SQuAD's are 3.2 tokens and 🦆 's are over 14 tokens.

References

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. *Proceedings of the Association for Computational Linguistics*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of Empirical Methods in Natural Language Processing*.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William M. Fisher, Kate Hunicke-Smith, David S. Pallett, Christine Pao, Alexander I. Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Computer Vision and Pattern Recognition*.

Hao Fang, Hao Cheng, Elizabeth Clark, Ariel Holtzman, Maarten Sap, Mari Ostendorf, Yejin Choi, and Noah A Smith. 2017. Sounding board–university of washingtons alexa prize submission. *Alexa Prize Proceedings*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. *Association for the Advancement of Artificial Intelligence*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. *Proceedings of the Association for Computational Linguistics*.

Mohit Iyyer, Wen tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the Association for Computational Linguistics*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the Association for Computational Linguistics*.

Tomasz Jurczyk, Amit Deshmane, and Jinho D. Choi. 2018. Analysis of wikipedia-based corpora for question answering. *arXiv preprint arXiv:abs/1801.02073*.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, abs/1712.07040.

John Langford, Lihong Li, and Alex Strehl. 2007. Vowpal wabbit online learning project.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke S. Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Conference on Computational Natural Language Learning*.

Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning for negotiation dialogues. *Proceedings of Empirical Methods in Natural Language Processing*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *Proceedings of Empirical Methods in Natural Language Processing*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv*, abs/1611.09268.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Conference of the North American Chapter of the Association for Computational Linguistics*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the Association for Computational Linguistics*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2018. Coqa: A conversational question answering challenge. *ArXiv*.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of Empirical Methods in Natural Language Processing*.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Signh, Tim Rocktschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *Proceedings of Empirical Methods in Natural Language Processing*.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Association for the Advancement of Artificial Intelligence*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *Proceedings of the International Conference on Learning Representations*.

Manfred Stede and David Schlangen. 2004. Information-seeking chat: Dialogues driven by topic-structure. In *Eighth workshop on the semantics and pragmatics of dialogue; SemDial*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the World Wide Web Conference*.

A. Talmor and J. Berant. 2018. The web as knowledge-base for answering complex questions. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Knowledge Base Question Answering via Encoding of Complex Query Graphs

Kangqi Luo[1] **Fengli Lin**[1] **Xusheng Luo**[2] **Kenny Q. Zhu**[1]
[1]Shanghai Jiao Tong University, Shanghai, China
[2]Alibaba Group, Hangzhou, China
{luokangqi,fenglilin}@sjtu.edu.cn, lxs140564@alibaba-inc.com, kzhu@cs.sjtu.edu.cn

Abstract

Answering complex questions that involve multiple entities and multiple relations using a standard knowledge base is an open and challenging task. Most existing KBQA approaches focus on simpler questions and do not work very well on complex questions because they were not able to simultaneously represent the question and the corresponding complex query structure. In this work, we encode such complex query structure into a uniform vector representation, and thus successfully capture the interactions between individual semantic components within a complex question. This approach consistently outperforms existing methods on complex questions while staying competitive on simple questions.

1 Introduction

The knowledge-based question answering (KBQA) is a task which takes a natural language question as input and returns a factual answer using structured knowledge bases such as Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and DBpedia (Auer et al., 2007). One simple example is a question like this: "What's the capital of the United States?" A common answer to such question is to identify the focus entity and the main relation predicate (or a sequence) in the question, and map the question to a triple fact query (US, $capital$, ?) over KB. The object answers are returned by executing the query. The mapping above is typically learned from question-answer pairs through distant supervision.

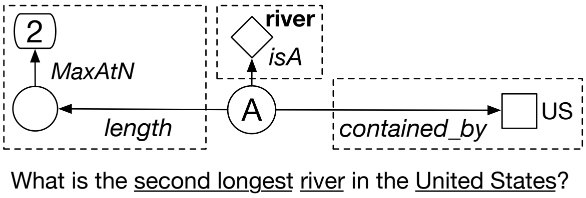

Figure 1: Running example of complex question.

While the above question can be answered by querying a single predicate or predicate sequence in the KB, many other more complex questions cannot, e.g. the question in Figure 1. To answer the question "What is the second longest river in United States", we need to infer several semantic clues: 1) the answer is contained by United States; 2) the answer is a river; 3) the answer ranks second by its length in descending order. Thus, multiple predicates are required to constrain the answer set, and we call such questions "complex questions" throughout this paper.

For answering complex questions, it's more important to understand the compositional semantic meanings of the question. As a classic branch of KBQA solutions, semantic parsing (SP) technique (Berant et al., 2013; Yih et al., 2015; Reddy et al., 2016; Hu et al., 2018) aims at learning semantic parse trees or equivalent query graphs [1] for representing semantic structures of the questions. For example in Figure 1, the query graph forms a tree shape. The answer node A, serving as the root of the tree, is the variable vertex that represents the real answer entities. The focus nodes (*US*, *river*, *2nd*) are extracted from the mentions of the question, and they constrain the answer node via predicate sequences in the knowledge base. Recently, neural network (NN) models have shown great promise in improving the performance of KBQA systems, and SP+NN techniques become the state-of-the-art on several KBQA datasets (Qu et al., 2018; Bao et al., 2016). According to the discussion above, our work extends the current research in the SP+NN direction.

The common step of SP-based approaches

---

[1] The term "query graph" is interchangeable with "query structure" and "semantic parsing tree" throughout this paper.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2185–2194
Brussels, Belgium, October 31 - November 4, 2018.

is to first collect candidate query graphs using bottom up parsing (Berant et al., 2013; Cai and Yates, 2013) or staged query generation methods (Yih et al., 2015; Bao et al., 2016), then predict the best graph mainly based on the semantic similarity with the given question. Existing NN-based methods follow an encode-and-compare framework for answering simple questions, where both the question and the predicate sequence are encoded as semantic vectors in a common embedding space, and the semantic similarity is calculated by the cosine score between vectors. In order to define the similarity function between one question and a complex query graph, an intuitive solution is to split the query graph into multiple semantic components, as the predicate sequences separated by dashed boxes in Figure 1. Then previous methods can be applied for modeling the similarity between the question and each part of the graph. However, such approach faces two limitations. First, each semantic component is not directly comparable with the whole question, since it conveys only partial information of the question. Second, and more importantly, the model encodes different components separately, without learning the representation of the whole graph, hence it's not able to capture the compositional semantics in a global perspective.

In order to attack the above limitations, we propose a neural network based approach to improve the performance of semantic similarity measurement in complex question answering. Given candidate query graphs generated from one question, our model embeds the question surface and predicate sequences into a uniform vector space. The main difference between our approach and previous methods is that we integrate hidden vectors of various semantic components and encode their interaction as the hidden semantics of the entire query graph. In addition, to cope with different semantic components of a query graph, we leverage dependency parsing information as a complementary of sentential information for question encoding, which makes the model better align each component to the question. The contribution of this paper is summarized below.

- We propose a light-weighted and effective neural network model to solve complex KBQA task. To the best of our knowledge, this is the first attempt to explicitly encode the complete semantics of a complex query graph (Section 2.2);
- We leverage dependency parsing to enrich question representation in the NN model, and conduct thorough investigations to verify its effectiveness (Section 2.2.2);
- We propose an ensemble method to enrich entity linking from a state-of-the-art linking tool, which further improves the performance of the overall task (Section 2.3);
- We perform comprehensive experiments on multiple QA datasets, and our proposed method consistently outperforms previous approaches on complex questions, and produces competitive results on datasets made up of simple questions (Section 3).

2 Approach

In this section, we present our approach for solving complex KBQA. First, we generate candidate query graphs by staged generation method (Section 2.1). Second, we measure the semantic similarities between the question and each query graph using deep neural networks (Section 2.2). Then we introduce an ensemble approach for entity linking enrichment (Section 2.3), Finally, we discuss the prediction and parameter learning step of this task (Section 2.4).

2.1 Query Graph Generation

We illustrate our staged candidate generation method in this section. Compared to previous methods, such as Bao et al. (2016), we employ a more effective candidate generation strategy, which takes advantage of implicit type information in query graphs and time interval information in the KB. In our work, we take 4 kinds of semantic constraints into account: **entity**, **type**, **time** and **ordinal** constraints. Figure 2 shows a concrete example of our candidate generation. For simplicity of discussion, we assume Freebase as the KB in this section.

Step 1: Focus linking. We extract possible (mention, focus node) pairs from the question. Focus nodes are the starting points of various semantic constraints, refer to Figure 2(a). For entity linking, we generate (mention, entity) pairs using the state-of-the-art entity linking tool S-MART (Yang and Chang, 2015). For type linking, we brutally combine each type with all uni-, bi-

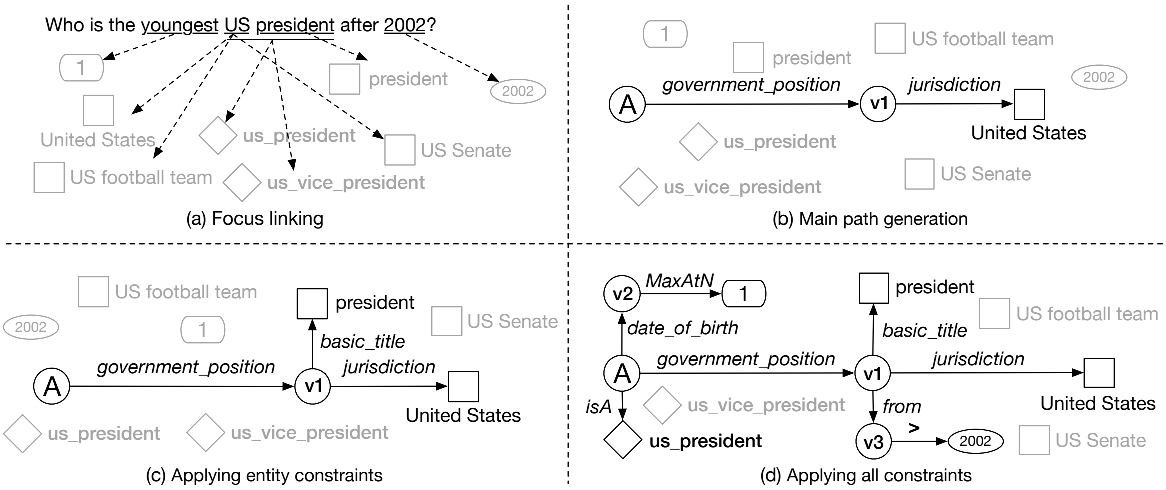

Figure 2: Running example of candidate generation.

and tri-gram mentions in the question, and pick top-10 (mention, type) pairs with the highest word embedding similarities of each pair. For time linking, we extract time mentions by simply matching year regex. For ordinal linking, we leverage a predefined superlative word list[2] and recognize mentions by matching superlative words, or the "ordinal number + superlative" pattern. The ordinal node is an integer representing the ordinal number in the mention.

Step 2: Main path generation. We build different main paths by connecting the answer node to different focus entities using 1-hop or 2-hop-with-mediator[3] predicate sequence. Figure 2(b) shows one of the main paths. Further constraints are attached by connecting an anchor node x to an unused focus node through predicate sequences, where the anchor node x is a non-focus node in the main path (A or v_1 in the example).

Step 3: Attaching entity constraints. We apply a depth-first search to search for combinations of multiple entity constraints to the main path through 1-hop predicate. Figure 2(c) shows a valid entity constraint, $(v_1, basic\_title, president)$. The advantage of depth-first search is that we can involve unlimited number of entities in a query graph, which has a better coverage than template-based methods.

Step 4: Type constraint generation. Type constraints can only be applied at the answer node using *IsA* predicate. Our improvement in this step is to filter type constraints using **implicit types** of the answer, derived from the outgoing predicates of the answer node. For example in Figure 2(c), the domain type of the predicate *government_position* is *politician*, which becomes the implicit type of the answer. Thus we can filter type constraints which are irrelevant to the implicit types, preventing semantic drift and speeding up the generation process. To judge whether two types in Freebase are relevant or not, we adopt the method in Luo et al. (2015) to build a rich type hierarchy of Freebase. Focus types are discarded, if they are not the super- or sub- types of any implicit types of the answer.

Step 5: Time and ordinal constraint generation. As shown in Figure 2(d), the time constraint is represented as a 2-hop predicate sequence, where the second is a virtual predicate determined by the preposition before the focus time, indicating the time comparing operation, like "before", "after" and "in". Similarly, the ordinal constraint also forms a 2-hop predicate sequence, where the second predicate represents descending (*MaxAtN*) or ascending order (*MinAtN*).

For the detail of time constraint, while existing approaches (Yih et al., 2015; Bao et al., 2016) link the focus time with only single time predicate, our improvement is to leverage **paired time predicates** for representing a more accurate time constraint. In Freebase, paired time predicates are used to represent facts within certain time intervals, like $from$ and to[4] in Figure 2(d). For time comparing operation "in", we link the time focus to the starting time predicate, but use both predi-

---

[2] ~20 superlative words, such as largest, highest, latest.

[3] Mediator is a kind of auxiliary nodes in Freebase maintaining N-ary facts.

[4] Short for $governmental\_position\_held.from$ and $governmental\_position\_held.to$ respectively.

cates in SPARQL query, restricting that the focus time lies in the time interval of the paired predicates.

After finishing all these querying stages, we translate candidate graphs into SPARQL query, and produce their final output answers. Finally, we discard query graphs with zero outputs, or using overlapped mentions.

2.2 NN-based Semantic Matching Model

The architecture of the proposed model is shown in Figure 3. We first replace all entity (or time) mentions used in the query graph by dummy tokens $\langle E \rangle$ (or $\langle Tm \rangle$). To encode the complex query structure, we split it into predicate sequences starting from answer to focus nodes, which we call *semantic components*. The predicate sequence doesn't include the information of focus nodes, except for type constraints, where we append the focus type to the *IsA* predicate, resulting in the predicate sequence like $\{$*IsA*, *river*$\}$. We introduce in detail the encoding methods for questions and predicate sequences, and how to calculate the semantic similarity score.

2.2.1 Semantic Component Representation

To encode a semantic component p, we take the sequence of both predicate ids and predicate names into consideration. As the example shown in Figure 3, the id sequence of the first semantic component is $\{$*contained_by*$\}$, and the predicate word sequence is the concatenation of canonical names for each predicate, that is $\{$"contained", "by"$\}$.

Given the word sequence $\{p_1^{(w)}, \dots, p_n^{(w)}\}$, we first use a word embedding matrix $E_w \in \mathbb{R}^{|V_w| \times d}$ to convert the original sequence into word embeddings $\{\boldsymbol{p}_1^{(w)}, \dots, \boldsymbol{p}_n^{(w)}\}$, where $|V_w|$ denotes the vocabulary size of natural language words, and d denotes the embedding dimension. Then we represent the word sequence using word averaging: $\boldsymbol{p}^{(w)} = \frac{1}{n} \sum_i \boldsymbol{p}_i^{(w)}$.

For the id sequence $\{p_1^{(id)}, \dots, p_m^{(id)}\}$, we simply take it as a whole unit, and directly translate it into vector representation using the embedding matrix $E_p \in \mathbb{R}^{|V_p \times d|}$ at path level, where $|V_p|$ is the vocabulary size of predicate sequences. There are two reasons for using such path embedding: 1) the length of id sequence is not larger than two, based on our generation method; 2) the number of distinct predicate sequences is roughly the same as the number of distinct predicates. We get the final vector of the semantic component by element-wise addition: $\boldsymbol{p} = \boldsymbol{p}^{(w)} + \boldsymbol{p}^{(id)}$.

2.2.2 Question Representation

We encode the question in both global and local level, which captures the semantic information with respect to each component p.

The global information takes the token sequence as the input. We use the same word embedding matrix E_w to convert the token sequence into vectors $\{\boldsymbol{q}_1^{(w)}, \dots, \boldsymbol{q}_n^{(w)}\}$. Then we encode the token sequence by applying bidirectional GRU network (Cho et al., 2014). The representation of the token sequence is the concatenation of the last forward and backward hidden states through the Bi-GRU layer, $\boldsymbol{q}^{(tok)} = [\overleftarrow{\boldsymbol{h}}_1^{(w)}; \overrightarrow{\boldsymbol{h}}_n^{(w)}]$.

To encode the question at local level, we leverage dependency parsing to represent long-range dependencies between the answer and the focus node in p. Since the answer is denoted by the wh-word in the question, we extract the dependency path from the answer node to the focus mention in the question. Similar with Xu et al. (2016), we treat the path as the concatenation of words and dependency labels with directions. For example, the dependency path between "what" and "United States" is $\{$what, $\overrightarrow{nsubj}$, is, $\overrightarrow{prep}$, in, $\overrightarrow{pobj}$, $\langle E \rangle\}$. We apply another bidirectional GRU layer to produce the vector representation at dependency level $\boldsymbol{q}_p^{(dep)}$, capturing both syntactic features and local semantic features. Finally we combine global and local representation by element-wise addition, returning the representation of the question with respect to the semantic component, $\boldsymbol{q}_p = \boldsymbol{q}^{(tok)} + \boldsymbol{q}_p^{(dep)}$.

2.2.3 Semantic Similarity Calculation

Given the query graph with multiple semantic components, $G = \{p^{(1)}, \dots, p^{(N)}\}$, now all its semantic components have been projected into a common vector space, representing hidden features in different aspects. We apply max pooling over the hidden vectors of semantic components, and get the compositional semantic representation of the entire query graph. Similarly, we perform max pooling for the question vectors with respect to each semantic component. Finally, we compute the semantic similarity score between the graph and question:

$$S_{sem}(q, G) = cos(\max_i \boldsymbol{p}^{(i)}, \max_i \boldsymbol{q}_p^{(i)}). \qquad (1)$$

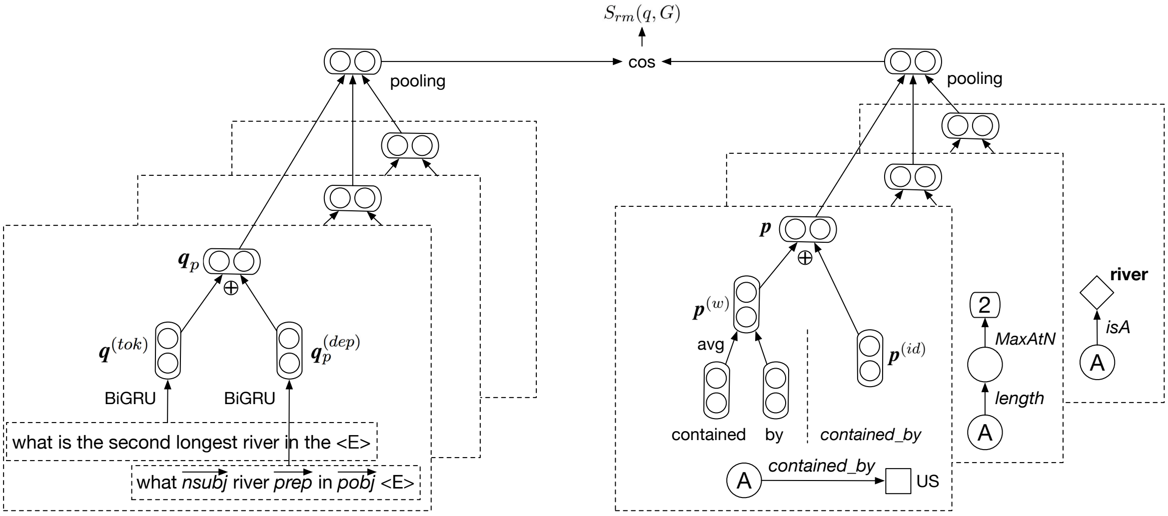

Figure 3: Overview of proposed semantic matching model.

Based on this framework, our proposed method ensures the vector spaces of the question and the entire query graph are comparable, and captures complementary semantic features from different parts of the query graph. It's worth mentioning that the semantic matching model is agnostic to the candidate generation method of the query graphs, hence it can be applied to the other existing semantic parsing frameworks.

2.3 Entity Linking Enrichment

The S-MART linker is a black box for our system, which is not extendable and tend to produce high precision but low recall linking results. To seek a better balance at entity linking, we propose an ensemble approach to enrich linking results. We first build a large lexicon by collecting all (mention, entity) pairs from article titles, anchor texts, redirects and disambiguation pages of Wikipedia. Each pair is associated with statistical features, such as linking probability, letter-tri-gram jaccard similarity and popularity of the entity in Wikipedia. For the pairs found in S-MART results, we take the above features as the input to a 2-layer linear regression model fitting their linking scores. Thus we learn a pseudo linking score for every pair in the lexicon, and for each question, we pick top-K highest pairs to enrich S-MART linking results, where K is a hyperparameter.

2.4 Training and Prediction

To predict the best query graph from candidates, we calculate the overall association score $S(q, G)$ between the question q and each candidate G, which is the weighted sum of features over entity linking, semantic matching and structural level. Table 1 lists the detail features.

During training step, we adopt hinge loss to maximize the margin between positive graphs G^+ and negative graphs G^-:

$$loss = max\{0, \lambda - S(q, G^+) + S(q, G^-)\}. \quad (2)$$

For each question, we pick a candidate graph as positive data, if the F_1 score of its answer is larger than a threshold (set to 0.1 in our work). We randomly sample 20 negative graphs G^- from the candidate set whose F_1 is lower than the corresponding G^+.

Category	Description
Entity	Sum of S-MART scores of all entities; Number of entities from S-MART; Number of entities from enriched lexicon;
Semantics	Semantic similarity score $S_{sem}(q, G)$;
Structural	Number of each kind of constraints in G; Whether a kind of constraints is used in G; Whether the main path is one-hop; Number of output answers.

Table 1: Full set of features.

3 Experiments

In this section, we introduce the QA datasets and state-of-the-art systems that we compare. We show the end-to-end results of the KBQA task, and perform detail analysis to investigate the importance of different modules used in our approach.

3.1 Experimental Setup

QA datasets: We conduct our experiments on ComplexQuestions (Bao et al., 2016), WebQuestions (Berant et al., 2013) and SimpleQuestions (Bordes et al., 2015). We use CompQ, WebQ and SimpQ as abbreviations of the above datasets, respectively. CompQ contains 2,100 complex questions collected from Bing search query log, and the dataset is split into 1,300 training and 800 testing questions. WebQ contains 5,810 questions collected from Google Suggest API, and is split into 3,778 training and 2,032 testing QA pairs. Each question is manually labeled with at least one answer entity in both datasets. SimpQ consists of more than 100K questions, and the gold answer of each question is a gold focus entity paired with a single predicate. This dataset is designed mainly for answering simple questions, and we use it for complementary evaluation.

Knowledge bases: For experiments on both CompQ and WebQ, we follow the settings of Berant et al. (2013) and Xu et al. (2016) to use the full Freebase dump [5] as the knowledge base, which contains 46M entities and 5,323 predicates. We host the knowledge base with Virtuoso engine [6]. For the experiments on SimpQ, the knowledge base we use is FB2M, which is a subset of Freebase provided with the dataset, consisting 2M entities and 10M triple facts.

Implementation detail: For all experiments in this section, we initialize word embeddings using GloVe (Pennington et al., 2014) word vectors with dimensions set to 300, and the size of Bi-GRU hidden layer is also set to 300. We tune the margin λ in $\{0.1, 0.2, 0.5\}$, the ensemble threshold K in $\{1, 2, 3, 5, 10, +\text{INF}\}$, and the batch size B in $\{16, 32, 64\}$. All the source codes, QA datasets, and detail results can be downloaded from `http://202.120.38.146/CompQA/`.

3.2 End-to-End Results

Now we perform KBQA experiments on WebQ and CompQ. We use the average F_1 score over all questions as our evaluation metric. The official evaluation script [7] measures the correctness of output entities at string level. While in CompQ, the annotated names of gold answer entities don't match the case of their names in Freebase, thus we follow Bao et al. (2016) to lowercase both annotated names and the output answer names before calculating the F_1 score. We set $\lambda = 0.5$, $B = 32$, $K = 3$ for WebQ and $K = 5$ for CompQ, as reaching the highest average F_1 on the validation set of each dataset.

We report the experimental results in Table 2. The result of Yih et al. (2015) on CompQ is reported by Bao et al. (2016) as their implemented result. Our approach outperforms existing approaches on CompQ dataset, and ranks 2nd on WebQ among a long list of state-of-the-art works. Jain (2016) achieves highest F_1 score on WebQ using memory networks, which is not semantic parsing based, and thus less interpretable. We point out that Xu et al. (2016) uses Wikipedia texts as the external community knowledge for verifying candidate answers, and achieves a slightly higher F_1 score (53.3) than our model, but the performance decreases to 47.0 if this step is removed. Besides, Yih et al. (2015) and Bao et al. (2016) used ClueWeb dataset for learning more accurate semantics, while based on the ablation test of Yih, the F_1 score of WebQ drops by 0.9 if ClueWeb information is removed.

Method	CompQ	WebQ
Dong et al. (2015)	-	40.8
Yao (2015)	-	44.3
Bast and Haussmann (2015)	-	49.4
Berant and Liang (2015)	-	49.7
Yih et al. (2015)	36.9	52.5
Reddy et al. (2016)	-	50.3
Xu et al. (2016) (w/o text)	-	47.0
Bao et al. (2016)	40.9	52.4
Jain (2016)	-	**55.6**
Abujabal et al. (2017)	-	51.0
Cui et al. (2017)	-	34.0
Hu et al. (2018)	-	49.6
Talmor and Berant (2018)	39.7	-
Ours (w/o linking enrich)	42.0	52.0
Ours (w/ linking enrich)	**42.8**	52.7

Table 2: Average F_1 scores on CompQ and WebQ datasets.

Our results show that entity enrichment method improves the results on both datasets by a large margin (0.8), which is a good help to our approach. We argue that the enriched results are directly comparable with other approaches, as S-MART itself is learned from semi-structured information in Wikipedia, such as anchor texts, redirect links and disambiguation pages, the enrichment step does not bring extra knowledge into our system. In addition, the improvements of the candidate generation step also show a positive effect. If we remove our implicit type filtering in Step 4 and time interval constraints in Step 5, the F_1 of CompQ slightly drops from 42.84 to 42.37. Al-

---

[5] detail information of the Freebase dump is available at https://github.com/syxu828/QuestionAnsweringOverFB/.

[6] http://virtuoso.openlinksw.com/

[7] The evaluation script is available at http://www-nlp.stanford.edu/software/sempre/.

though these improvements mainly concern time-related questions (around 25% in CompQ), we believe these strategies can be useful tricks in the further researches.

As a complementary evaluation, we perform semantic matching experiments on SimpQ. Given the gold entity of each question, we recognize the entity mention in the question, replace it with $\langle E \rangle$, then predict the correct predicate. Table 3 shows the experimental results. The best result is from Qu et al. (2018), which learns the semantic similarity through both attentive RNN and similarity matrix based CNN. Yu et al. (2017) proposed another approach using multi-layer BiLSTM with residual connections. Our semantic matching model performs slightly below these two systems, since answering simple questions is not the main goal of this paper. Comparing with these approaches, our semantic matching model is lightweighted, with a simpler structure and fewer parameters, thus is easier to tune and remains effective.

Method	Relation Inputs	Accuracy
BiLSTM w/ words	words	91.2
BiLSTM w/ rel_name	rel_name	88.9
Yih et al. (2015)	char-3-gram	90.0
Yin et al. (2016)	words	91.3
Yu et al. (2017)	words+rel_name	93.3
Qu et al. (2018)	words+rel_separated	**93.7**
Ours	words+path	93.1

Table 3: Accuracy on the SimpleQuestions dataset.

3.3 Ablation Study

In this section, we explore the contributions of various components in our system.

Semantic component representation: We first evaluate the results on CompQ and WebQ under different path encoding methods. Recap that the encoding result of a semantic component is the summation of its word and id path representations (Section 2.2.1), thus we compare encoding methods by multiple combinations. For encoding predicate word sequence, we use BiGRU (the same setting as encoding question word sequence) as the alternative of average word embedding. For encoding predicate id sequence, we use average predicate embedding as the alternative of the current path-level embedding ($PathEmb$).

The experimental results are shown in Table 4. The encoding method $None$ means that we don't encode the id or word sequence, and simply take the result of the other sequence as the representation of the whole component. we observe that the top three combination settings, ignoring either word or id sequence, perform worse than the bottom three settings. The comparison demonstrates that predicate word and id representation can be complementary to each other. The performance gain is not that large, mainly because predicate id features are largely covered by their word name features.

For the encoding of id sequences, $PathEmb$ works better than average embedding, consistently boosting F_1 by 0.65 on both datasets. The former method treats the whole sequence as a single unit, which is more flexible and can potentially learn diverse representations of id sequences that share the same predicates. For the encoding of word sequences, the average word embedding method outperforms BiGRU on CompQ, and the gap becomes smaller when running on WebQ. This is mainly because the training set of WebQ is about 3 times larger than that of CompQ, making it easier for training a more complex model.

Word repr.	Id repr.	CompQ F_1	WebQ F_1
None	PathEmb	41.11	51.86
Average	None	42.18	51.74
BiGRU	None	41.80	51.87
Average	Average	42.16	52.00
BiGRU	PathEmb	41.52	52.33
Average	PathEmb	**42.84**	**52.66**

Table 4: Ablation results on path representation.

Semantic composition and question representation: To demonstrate the effectiveness of semantic composition, we construct a straightforward baseline, where we remove the max pooling operation in Eq. (2), and instead calculate the semantic similarity score as the summation of individual cosine similarities: $S_{sem}(q, G) = \sum_i cos(\boldsymbol{p}^{(i)}, \boldsymbol{q}_p^{(i)})$. For methods of question encoding, we setup ablations by turning off either sentential encoding or dependency encoding.

Table 5 shows the ablation results on CompQ and WebQ. When dependency path information is augmented with sentential information, the performance boosts by 0.42 on average. Dependency paths focus on hidden features at syntactic and functional perspective, which is a good complementary to sentential encoding results. However, performances drop by 2.17 if only dependency information is used, we find that under certain dependency structures, crucial words (bolded) are not in the path between the answer and the focus mention (underlined), for example, "who did draco malloy end up **marrying**" and "who did the philippines gain **independence** from". While we observe about 5% of such questions in WebQ, it's hard to predict the correct query graph without crucial words.

In terms of semantic composition, Our max pooling based method consistently outperforms the baseline method. The improvement on WebQ is smaller than on CompQ, largely due to the fact that 85% questions in WebQ are simple questions (Bao et al., 2016). As a result of combination, our approach significantly outperforms the vanilla SP+NN approach on CompQ by 1.28, demonstrating the effectiveness of our approach. Theoretically, the pooling outcome may lead to worse end-to-end result when there are too many semantic components in one graph, because the pooling layer takes too many vectors as input, different semantic features between similar query graphs become indistinguishable. In our task, only 0.5% of candidate graphs have more than 3 semantic components, so pooling is a reasonable way to aggregate semantic components in this scenario.

Composition	Q_repr	CompQ F_1	WebQ F_1
Baseline	sentential	41.56	52.14
Baseline	both	42.35	52.39
Ours	dependency	41.48	49.69
Ours	sentential	42.59	52.28
Ours	both	**42.84**	**52.66**

Table 5: Ablation results on question representation and compositional strategy.

To further explain the advantage of semantic composition, we take the following question as an example: "who is gimli's father in the hobbit". Two query graphs are likely to be the final answer: 1) ($?$, $children$, $gimli\_person$); 2) ($?$, $fictional\_children$, $gimli\_character$) $\wedge$ ($?$, $appear\_in$, $hobbit$). If observing semantic components individually, the predicate $children$ is most likely to be the correct one since "'s father" is highly related and with plenty of positive training data. Both $fictional\_children$ and $appear\_in$ get a much lower similarity compared with $children$, hence the baseline method prefer the first query graph. In the meantime, our proposed method learns the hidden semantics of the second candidate by absorbing salient features from both predicates, and such compositional representation is closer to the semantics of the entire question than a simple "children" predicate. That's why our method manages to answer it correctly.

3.4 Error Analysis

We randomly analyzed 100 questions from CompQ where no correct answers are returned. We list the major causes of errors as follows:

Main path error (10%): This type of error occurred when the model failed to understand the main semantics when facing some difficult questions (e.g. "What native american sports heroes earning two gold medals in the 1912 Olympics");

Constraint missing (42%): These types of questions involve implicit constraints, for example, the question "Who was US president when Traicho Kostov was teenager" is difficult to answer because it implies an implicit time constraint "when Traicho Kostov was teenager";

Entity linking error (16%): This error occurs due to the highly ambiguity of mentions. For example, the question "What character did Robert Pattinson play in Harry Potter" expects the film "Harry Potter and the Goblet of Fire" as the focus, while there are 7 movies in Harry Potter series;

Miscellaneous (32%): This error class contains questions with semantic ambiguity or not reasonable. For example, the question "Where is Byron Nelson 2012" is hard to understand, because "Byron Nelson" died in 2006 and maybe this question wants to ask where did he die.

4 Related Work

Knowledge Base Question Answering(KBQA) has been a hot research top in recent years. Generally speaking, the most popular methods for KBQA can be mainly divided into two classes: information retrieval and semantic parsing.

Information retrieval based system tries to obtain target answer directly from question information and KB knowledge without explicit considering interior query structure. There are various methods (Yao and Van Durme, 2014; Bordes et al., 2015; Dong et al., 2015; Xu et al., 2016) to select candidate answers and to rank results.

Semantic parsing based approach focuses on constructing a semantic parsing tree or equivalent query structure that represents the semantic meaning of the question. In terms of logical representation of natural language questions, many methods have been tried, such as query graph (Yih et al., 2014, 2015) or RDF query language (Unger et al., 2012; Cui et al., 2017; Hu et al., 2018).

Recently, as the development of deep learning, NN-based approaches have been combined into the KBQA task (Bordes et al., 2014), showing promising result. These approaches tries to use neural network models to encode both questions and answers (or query structures) into the vector space. Subsequently, similarity functions are used to select the most appropriate query structure to generate the final answer. For example, Bordes et al. (2014) focuses on embedding the subgraph of the candidate answer; Yin et al. (2016) uses character-level CNN and word-level CNN to match different information; Yu et al. (2017) introduces the method of hierarchical residual RNN to compare questions and relation names; Qu et al. (2018) proposes the AR-SMCNN model, which uses RNN to capture semantic-level correlation and employs CNN to extract literal-level words interaction.

Belonging to NN-based semantic parsing category, our approach employs a novel encoding structure method to solve complex questions. Previous works such as Yih et al. (2015) and Bao et al. (2016) require a recognition of a main relation and regard other constraints as variables added to this main relation. Unlike their approaches, our method encodes multiple relations (paths) into a uniform query structure representation (semantic composition), which allows more flexible query structures.

There are also some works can't be simply classified in to IR based methods or SP based methods. Jain (2016) introduces Factual Memory Network, which tries to encode KB and questions in same word vector space, extract a subset of initial candidate facts, then try to employ multi-hop reasoning and refinement to find a path to answer entity. Reddy et al. (2016), Abujabal et al. (2017), and Cui et al. (2017) try to interpret question intention by templates, which learned from KB or QA corpora. Talmor and Berant (2018) attempts to answering complex questions by decomposing them into a sequence of simple questions.

5 Conclusion

To the best of our knowledge, this is the first work to handle complex KBQA task by explicitly encoding the complete semantics of a complex query graph using neural networks. We studied different methods to further improve the performance, mainly leveraging dependency parse and the ensemble method for linking enrichment. Our model becomes the state-of-the-art on ComplexQuestions dataset, and produces competitive results on other simple question based datasets. Possible future work includes supporting more complex semantics like implicit time constraints.

Acknowledgment

Kenny Q. Zhu is the contact author and was supported by NSFC grants 91646205 and 61373031. Thanks to the anonymous reviewers for their valuable feedback.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *WWW*, pages 1191–1200.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.

Jun-Wei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *COLING*, pages 2503–2514.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *CIKM*, pages 1431–1440.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 165–180. Springer.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*, pages 423–433.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Proceedings of SSST-8*, pages 103–111.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. Kbqa: learning question answering over qa corpora and knowledge bases. *Proceedings of the VLDB Endowment*, 10(5):565–576.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *ACL (1)*, pages 260–269.

Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2018. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):824–837.

Sarthak Jain. 2016. Question answering over knowledge base using factual memory networks. In *Proceedings of the NAACL Student Research Workshop*, pages 109–115.

Kangqi Luo, Xusheng Luo, and Kenny Zhu. 2015. Inferring binary relation schemas for open information extraction. In *EMNLP*, pages 555–560.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Yingqi Qu, Jie Liu, Liangyi Kang, Qinfeng Shi, and Dan Ye. 2018. Question answering over freebase via attentive rnn with similarity matrix based cnn. *arXiv preprint arXiv:1804.03317*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*, pages 697–706. ACM.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL-HLT*.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *WWW*, pages 639–648. ACM.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *ACL*, pages 2326–2336.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *ACL-IJCNLP*, pages 504–513.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *NAACL*, pages 66–70.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *ACL*, pages 956–966.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL-IJCNLP*, pages 1321–1331.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *ACL*, volume 2, pages 643–648.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *COLING*, pages 1746–1756.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *ACL*, pages 571–581.

Neural Relation Extraction via Inner-Sentence Noise Reduction and Transfer Learning

Tianyi Liu[1], **Xinsong Zhang**[1], **Wanhao Zhou**[1] **and Weijia Jia**[2,1]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Department of Computer and Information Science, University of Macau
`{liutianyi, xszhang0320, whzhou, jiawj}@sjtu.edu.cn`

Abstract

Extracting relations is critical for knowledge base completion and construction in which distant supervised methods are widely used to extract relational facts automatically with the existing knowledge bases. However, the automatically constructed datasets comprise amounts of low-quality sentences containing noisy words, which is neglected by current distant supervised methods resulting in unacceptable precisions. To mitigate this problem, we propose a novel word-level distant supervised approach for relation extraction. We first build Sub-Tree Parse (STP) to remove noisy words that are irrelevant to relations. Then we construct a neural network inputting the subtree while applying the entity-wise attention to identify the important semantic features of relational words in each instance. To make our model more robust against noisy words, we initialize our network with a priori knowledge learned from the relevant task of entity classification by transfer learning. We conduct extensive experiments using the corpora of New York Times (NYT) and Freebase. Experiments show that our approach is effective and improves the area of Precision/Recall (PR) from 0.35 to 0.39 over the state-of-the-art work.

1 Introduction

Relation extraction aims to extract relations between pairs of marked entities in raw texts. Traditional supervised methods are time-consuming for the requirement of large-scale manually labeled data. Thus, Mintz et al. (2009) propose the distant supervised relation extraction, in which amounts of sentences are crawled from web pages of New York Times (NYT) and labeled with a known knowledge base automatically. The method assumes that if two entities have a relation in a known knowledge base, all instances that mention these two entities will express the same relation. Obviously, this assumption is too strong, since a sentence that mentions the two entities does not necessarily express the relation contained in a known knowledge base. As described in Riedel et al. (2010), the assumption leads to the wrong labeling problem. In order to tackle the wrong labeling problem, various multi-instance learning methods are adopted by mitigating noise between sentences (Hoffmann et al., 2011; Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016). Despite the wrong labeling problem, distant supervised methods may suffer from the low quality of sentences which derive from the large-scale automatically constructed dataset by crawling web pages (Yang et al., 2017). To handle the problem of low-quality sentences, we have to face two major challenges: (1) Reduce word-level noise within sentences; (2) Improve the robustness of relation extraction against noise.

To explain the influence of word-level noise within sentences, we consider the following sentence as an example: *[It is no accident that the main event will feature the junior welterweight champion miguel cotto, a puerto rican, against Paul Malignaggi, an Italian American from Brooklyn.]*, where *Paul Malignaggi* and *Brooklyn* are two corresponding entities. The subsentence *[Paul Malignaggi, an Italian American from Brooklyn.]* keeps enough words to express the relation */people/person/place_of_birth*, and the other words could be regarded as noise that may hamper the extractor's performance. Meanwhile, as shown in Figure 1, half of the original sentences are longer than 40 words, which means that there are many irrelevant words inside sentences. To be more detail, there are about 12 noisy words in each sentence on average, and 99.4% of sentences in the NYT-10 dataset have noise. Although the Shortest Dependency Path (SDP) proposed by Xu et al. (2015) tries to get rid of irrelevant words for relation extraction, it is not suitable to handle such informal sentences. Moreover, word-level attention has been leveraged to alleviate the impact of noisy

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2195–2204
Brussels, Belgium, October 31 - November 4, 2018.

words (Zhou et al., 2016), but it weakens the importance of entity features for relation extraction.

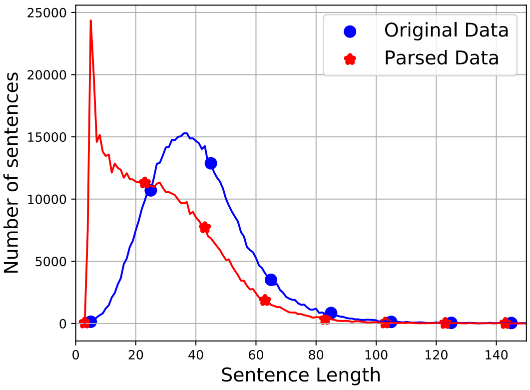

Figure 1: Comparison of sentence length distribution between original data and parsed data.

As for the second challenge, a robust model could extract precise relation features even from low-quality sentences containing noisy words. However, previous neural methods are always lacking in robustness because parameters are initialized randomly and hard to tune with noisy training data, resulting in the poor performance of extractors. Inspired by Kumagai (2016), initializing neural networks with a priori knowledge learned from relevant tasks by transfer learning could improve the robustness of the target task. For the relation extraction, entity type classification can be used as the relevant task since entity types provide abundant background knowledge. For instance, the sentence *[Alfead Kahn, the Cornell-University economist who led the fight to deregulate airplanes.]* has a relation *business/person/company*, which is hard to decide without the information that *Alfead Kahn* is a person and *Cornell-University* is a company. Therefore, type features learned from entity type classification are proper a priori knowledge to initialize the relation extractor.

In this paper, we propose a novel word-level approach for distant supervised relation extraction by reducing inner-sentence noise and improving robustness against noisy words. To reduce inner-sentence noise, we utilize a novel Sub-Tree Parse (STP) method to remove irrelevant words by intercepting a subtree under the parent of entities' lowest common ancestor. As shown in Figure 1, the average length of the parsed sentences is much shorter. Furthermore, the entity-wise attention is adopted to alleviate the influence of noisy words in the subtree and emphasize the task-relevant features. To tackle the second challenge, we initialize our model parameters with a priori knowledge learned from the entity type classification task by transfer learning. The experimental results show that our model can achieve satisfactory performance among the state-of-the-art works. Our contributions are summarized as follows:

- To handle the problem of low-quality sentences, we propose the STP to remove noisy words of sentences and the entity-wise attention mechanism to enhance semantic features of relational words.
- We first propose to initialize the neural relation extractor with a priori knowledge learned from entity type classification, which strengthens its robustness against low-quality corpus.
- Our model achieves significant results for distant supervised relation extraction, which improves the Precision/Recall (PR) curve area from 0.35 to 0.39 and increases top 100 predictions by 6.3% over the state-of-the-art work.

2 Related Work

The distant supervised method plays an increasingly essential role in relation extraction due to its less requirement of human labor (Mintz et al., 2009). However, an evident drawback of the method is the wrong labeling problem. Thus, multi-instance and multi-label learning methods are proposed to address this issue (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). Meanwhile, other researches (Angeli et al., 2014; Han and Sun, 2016) incorporate human-designed features and leverage Natural Language Processing (NLP) tools.

As neural networks have been widely used, an increasing number of researches have been proposed. Zeng et al. (2015) use a piecewise convolutional neural network with multi-instance learning. Furthermore, selective attention over instances with the neural network is proposed (Lin et al., 2016). Making use of entity description, Ji et al. (2017) assign more precise attention weights. Focused on the imbalance of datasets, a soft label method has been proposed by Liu et al. (2017).

Recently, reinforcement learning and adversarial learning are widely used to select the valid instances for relation extraction (Feng et al., 2018; Qin et al., 2018b,a).

However, above methods ignore inner-sentence noise. To better remove irrelevant words, the SDP between entities is proved to be effective (De Marneffe and Manning, 2008; Chen and Manning, 2014; Xu et al., 2015; Miwa and Bansal, 2016). Nevertheless, in our observation, the SDP deals with informal texts difficultly (See Section 3.1 for details). Furthermore, word-level attention is adopted to focus on relational words for relation extraction (Zhou et al., 2016), but it hinders the effect of entity words.

Transfer learning proposed by Pratt (1993) provides a new approach to leverage knowledge extracted by related tasks to enhance the performance of the target task. Furthermore, parameter transfer learning is proved to be effective to improve the stability of models by initializing model parameters reasonably (Pan and Yang, 2010; Kumagai, 2016).

3 Methodology

In this section, we present our methodology for distant supervised relation extraction. Figure 2 shows the overall architecture of our model. Our model is divided into three parts:

Sub-Tree Parser. Input instances are parsed to dependency parse trees by the Stanford parser[1] (Chen and Manning, 2014) at first. Then words in the STP and relative positions are transformed to distributed representations.

Entity-Wise Neural Extractor. Given the representation of each subtree, Bidirectional Gated Recurrent Unit (BGRU) extracts specific features. Then, entity-wise attention combined with word-level attention is applied to reducing irrelevant features for relation extraction. Finally, the sentence-level attention is used to alleviate the influence of wrong labeling sentences.

Parameter-Transfer Initializer. The transfer learning method pre-trains our model parameters from the task of entity type classification aiming at boosting the performance of relation extraction.

3.1 Sub-Tree Parser

Each instance is put into the dependency parse module to build the dependency parse tree in the first place. Then we can tailor the sentences based on the STP method. Finally, we transform word tokens and position tokens of each instance to distributed representations by embedding matrixes.

Sub-Tree Parse

In order to reduce inner-sentence noise and extract relational words, we propose the STP method which intercepts the subtree of each instance under the parent of entities' lowest common ancestor. For instance, in Figure 2(b), *China* and *Shanghai* are entities connected directly with the appositive relation. The instance *[In 1990, he lives in Shanghai, China.]* will be transformed to *[in Shanghai, China.]* on the basis of the STP, where *in* is the parent of *Shanghai* and *China* lowest common ancestor and kept as important information for expressing the relation *location/location/contain*. Words connected by the imaginary line indicating the extracted subtree are reorganized into their original sequence order to form network inputs.

Among the parse tree, the SDP has been widely used by Chen and Manning (2014) and Xu et al. (2015) to help models focus on relational words. However, in our observation, the SDP is not appropriate in the condition that key relation words are not in the SDP. Although additional information (dependency relations between words) is adopted to enhance the performance of SDP, we found they have the minor effect through our experiment. Thus, we do not make use of other types of linguistic information. As Figure 2(b) shows, in the SDP method, the original sentence will be transformed to *[Shanghai China]* because *Shanghai* and *China* are connected with each other directly in the dependency parse tree, which results in deleting the keyword *in* and may confuse the model when extracting relations. Compared with SDP, the STP method is more appropriate to extract useful information in informal sentences where relational words are always not in the SDP.

Word and Position Embeddings

The inputs of the network are word and position tokens, which are transformed to the distributed representations before they are fed into the neural model. We map j^{th} word in the i^{th} instance to a vector of k dimensions denoted as $x_{ij}^{w} \in R^{k}$ through the Skip-Gram model (Mikolov et al., 2013). Like Zeng et al. (2014), we leverage Pos1 and Pos2 to specify entity pairs, which are defined as the relative distances of current word from head

---

[1] https://nlp.stanford.edu/software/lex-parser.shtml

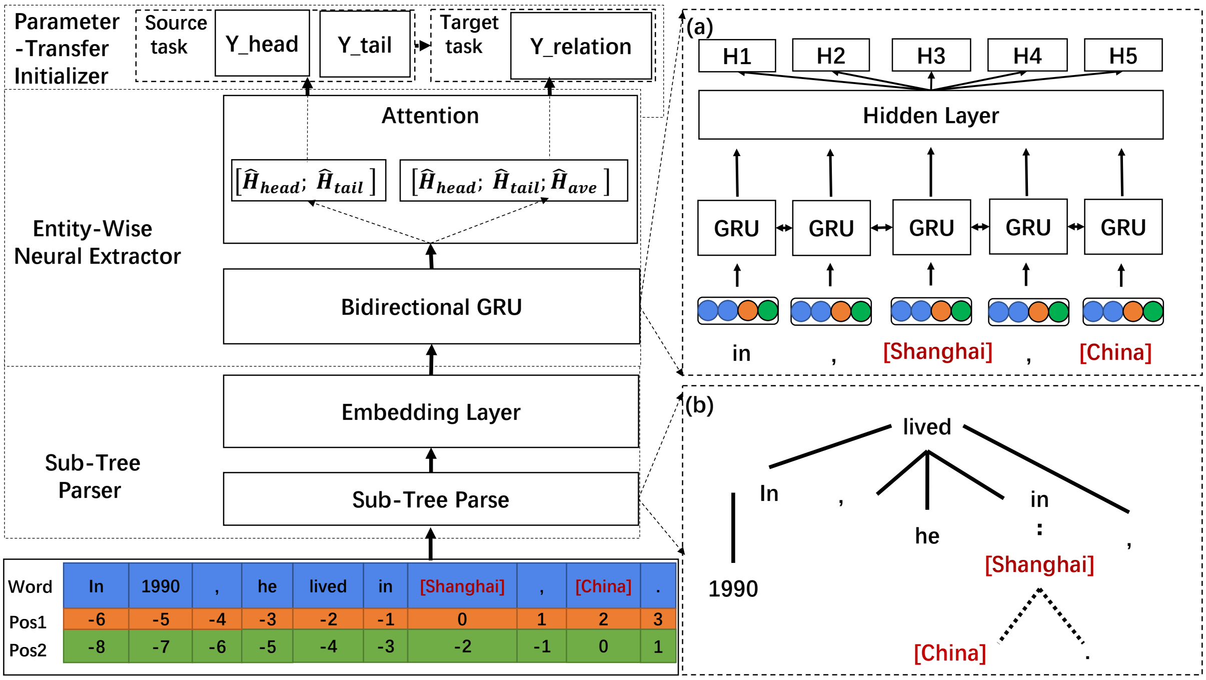

Figure 2: Overall architecture of our model is used for distant supervised relation extraction, expressing the process of handling instances. There are two modules described in detail: (a) One is the BGRU; (b) Another is the STP, where words in the red brackets represent entities (better viewed in color).

entity and tail entity. For instance, in Figure 2 relative distances of *lived* from *Shanghai* and *China* are -2 and -4 respectively. Then, the position token of each word is transformed to a vector in l dimensions. Position embeddings are denoted as $x_{ij}^{p1} \in R^l$ and $x_{ij}^{p2} \in R^l$ respectively. Finally, the input representation for x_{ij} is concatenated by word embedding x_{ij}^{w}, position embeddings x_{ij}^{p1} and x_{ij}^{p2} , which is denoted as $x_{ij} = [x_{ij}^{w}; x_{ij}^{p1}; x_{ij}^{p2}]$ where $x_{ij} \in R^{k+2l}$.

3.2 Entity-Wise Neural Extractor

As shown in Figure 2, we transform the STP into feature vectors by BGRU at first. Next, entity-wise attention combined with the hierarchical-level attention mechanism is applied to enhancing semantic features of each instance.

BGRU over STP

Since the transfer learning and entity-wise attention require the specific features of entities in tree parsed instances as their input, we adopt Gated Recurrent Unit (GRU) (Cho et al., 2014) to be our based relation extractor, which can extract global information of each word by pointing out its corresponding position in the sequence. It can be briefly described as below:

$$h_{it} = GRU(x_{it}) \tag{1}$$

where x_{it} is the t^{th} word representation in the i^{th} parsed instance as described in the input layer, and $h_{it} \in R^m$ is the hidden state of GRU in m dimensions.

Furthermore, BGRU implementing GRU in a different direction can access future as well as past context. Under our circumstance, BGRU combined with the STP can extract semantic and syntactic features adequately. Figure 2(a) shows the processing of BGRU over STP. The following equation defines the operation mathematically.

$$h_{it} = [\overrightarrow{h_{it}} \oplus \overleftarrow{h_{it}}] \tag{2}$$

In above equation, the t^{th} word output $h_{it} \in R^m$ of BGRU is the element-wise addition of the t^{th} hidden states of forward GRU and backward one.

Entity-wise Attention

To reduce noise within sentences, we propose the entity-wise attention mechanism to help our model focus on relational words, especially entity words for relation extraction. Assume that H_i is the i^{th} instance matrix consisting of T word vectors $[h_{i1}, h_{i2}, \cdots, h_{iT}]$ produced by BGRU.

Not all words contribute equally to the representation of the sentence. Entity words are of great importance because they are significantly beneficial to relation extraction. In our model, entity-wise attention assigns the weight α^e_{it} to focus on the target entity and removes noise further. It is defined as follows:

$$\alpha^e_{it} = \begin{cases} 1 & t = head, tail \\ 0 & others \end{cases} \tag{3}$$

In the above equation, $\alpha^e_{it} = 1$ if t^{th} word belongs to the head or tail entity.

Hierarchical-level Attention

To reduce inner-sentence noise further and de-emphasize noisy sentences, we incorporate word-level attention and sentence-level attention as hierarchical-level attention which is introduced in Yang et al. (2016).

Word-level Attention. It assigns an additional weight α^w_{it} to relational word h_{it} due to its relevance to the relation as described by Zhou et al. (2016). It can be described as follows:

$$\alpha^w_{it} = \frac{exp(h_{it}A^w r^w)}{\sum_{t=1}^{T} exp(h_{it}A^w r^w)} \tag{4}$$

where A^w is a weighted matrix, and vector r^w can be seen as a high level representation in a fixed query what is the informative word over the other words.

The i^{th} sentence representation $S_i \in R^m$ is computed as a weighted sum of h_{it}:

$$S_i = \sum_{t=1}^{T} (\alpha^w_{it} + \alpha^e_{it}) h_{it} \tag{5}$$

Sentence-level Attention. After we get the instance representation S_i, we adopt the selective attention mechanism over instances to de-emphasize the noisy sentence (Lin et al., 2016), which is described as follows:

$$S = \sum_i \alpha^s_i S_i \tag{6}$$

$$\alpha^s_i = \frac{exp(S_i A^s r^s)}{\sum_i exp(S_i A^s r^s)} \tag{7}$$

where A^s is a weighted matrix, r^s is the query vector associated with the relation, and $S \in R^m$ is the output of the sentence-level attention layer.

3.3 Parameter-Transfer Initializer

The transfer learning method pre-trains our model parameters in the entity type classification task, which in turn contributes to the relation extraction.

Pre-learn the Entity Type

As entity type information plays a significant role in detecting relation types, the entity type classification task is considered to be the source task, which is learned before the relation extraction task. According to Eq. 6, outputs of the sentence-level attention layer for the head entity and tail entity task are S_{head} and S_{tail} respectively. They are ultimately fed into the softmax layer:

$$\hat{p}^i = softmax(W_i S_i + b_i); \ i \in \{head, tail\} \tag{8}$$

where W_i and b_i are the weight and bias for the entity type classification task respectively, $\hat{p}^i \in R^{z_t}$ is the predicted probability of each class and z_t is the number of entity classes. The loss function of the source task is the negative log-likelihood of the true labels:

$$\begin{aligned} & J_e(\theta_0, \theta_{head}, \theta_{tail}) = \beta \|\theta_0\|^2 \\ & + \sum_t (-\frac{1}{z_t} \lambda_t \sum_{i=1}^{z_t} y^t_i log(\hat{p}^t_i) + \beta \|\theta_t\|^2) \\ & t \in \{head, tail\} \end{aligned} \tag{9}$$

where λ_t is the weight of each task, θ_0 is the shared model parameters, θ_{head} and θ_{tail} are individual parameters for the head and tail entity classification tasks respectively, $y^t \in R^{z_t}$ is the one-hot vector representing ground truth, and β is the hyper-parameter for $L2$ regularization.

Train the Relation Extractor

Based on the pre-trained model in the entity type classification task, the relation extractor initializes shared parameters θ_0 within the best state of the pre-trained model and independent parameters θ_r randomly. Same as the entity type classification task, the output S_r of the attention layer for the relation extraction task is finally fed into the softmax layer and the loss is calculated by cross entropy, which is defined as follows:

$$\hat{p} = softmax(W_r S_r + b_r) \tag{10}$$

$$\begin{aligned} J_r(\theta_0, \theta_r) = & -\frac{1}{z_r} \sum_{i=1}^{z_r} y_i log(\hat{p}_i) \\ & + \beta(\|\theta_0\|^2 + \|\theta_r\|^2) \end{aligned} \tag{11}$$

where W_r, b_r, $y \in R^{z_r}$, $\hat{p} \in R^{z_r}$, θ_r and β are defined similarly in the entity type classification task.

As shown in Figure 2, two tasks share all layers except attention and output layers. Assume that the set of total model parameters is θ. Thus, θ, θ_0, θ_r, θ_{head} and θ_{tail} have a relationship described in the following equations:

$$\theta = \theta_0 \cup \theta_{head} \cup \theta_{tail} \cup \theta_r \tag{12}$$

$$\begin{aligned} \theta_i = \{A_i^w, r_i^w, A_i^s, r_i^s, W_i, b_i\} \\ i \in \{head, tail, r\} \end{aligned} \tag{13}$$

where A_i^w, r_i^w, A_i^s, r_i^s, W_i and b_i are parameters in attention and output layers.

Optimize the Objective Function

At first, we minimize J to obtain θ_0 at the best model state $\hat{\theta}_0$ for entity type classification. Then we minimize J_r for the best performance of relation extraction under the initialization of θ_0 to be $\hat{\theta}_0$. Above process can be summarized as the following equation:

$$\begin{aligned} min \quad J(\theta) = &\lambda J_e(\theta_0, \theta_{head}, \theta_{tail}) + \\ &(1-\lambda) J_r(\theta_0, \theta_r) \end{aligned} \tag{14}$$

where $\lambda \in (0,1)$ is the hyperparameter to determine the importance of each task at different training steps. We use Adam (Kingma and Ba, 2014) optimizer to minimize the objective $J(\theta)$.

4 Experiments

Our experiments are designed to demonstrate that our model alleviates the influence of word-level noise arising from low-quality sentences. In this section, we first introduce the dataset and evaluation metrics. Next, we describe parameter settings. Then we evaluate effects of the STP, entity-wise attention and the parameter-transfer initializer. Finally, we compare our model with the state-of-the-art works by several evaluation metrics.

4.1 Dataset and Evaluation Metrics

To evaluate the performance of our model, we adopt a widely used dataset NYT-10 developed by Riedel et al. (2010). NYT-10 dataset is constructed by aligning relational facts in Freebase (Bollacker et al., 2008) with the NYT corpus, where sentences from 2005-2006 are used as training set, and sentences from 2007 are used for testing. For training data, there are 522,611 sentences, 281,270 entity pairs, and 18,252 relational facts; for testing data, there are 172,448 sentences, 96,678 entity pairs and 1,950 relational facts. There are 53 relations including a special relation NA, which means that there is no relation between the entity pair in the instance. Meanwhile, all relations in Freebase are defined on head types and tail types. Therefore, we can construct datasets for type prediction tasks with the same dataset. The dataset has 29 head types and 26 tail types.

Like previous works, we evaluate our model with the held-out metrics, which compare relations found by models with those in Freebase. The held-out evaluation provides a convenient way to assess models. We report both the PR curve and Precision at top N predictions (P@N) at various numbers of instances under each entity pair:

One: For each entity pair, we randomly select one instance to represent the relation.

Two: For each entity pair, we randomly select two instances to represent the relation.

All: For each entity pair, we select all instances to represent the relation.

4.2 Experimental Settings

In the experiment, we utilize word2vec[2] to train word embeddings on NYT corpus. We use the cross-validation to tune our model and grid search to determine model parameters. The grid search approach is used to select optimal learning rate lr for Adam optimizer among $\{0.1, 0.001, 0.0005, 0.0001\}$, GRU size $m \in \{100, 160, 230, 400\}$, position embedding size $l \in \{5, 10, 15, 20\}$. Table 1 shows all parameters for our task. We follow experienced settings for other parameters because they make little influence to our model performance.

GRU size m	230
Word embedding dimension k	50
POS embedding dimension l	5
Batch size n	50
Entity-Task weights($\lambda_{head}, \lambda_{tail}$)	0.5,0.5
Entity-Relation Task weight λ	0.3
Learning rate lr	0.001
Dropout probability p	0.5
l_2 penalty β	0.0001

Table 1: Parameter Settings

---

[2] https://code.google.com/p/word2vec/

4.3 Effect of Various Model Parts

In this section, we utilize the PR curve to evaluate the effects of three main parts in our model: the STP, entity-wise attention and the parameter-transfer initializer.

Effect of the STP

To demonstrate the effect of the STP, we adopt BGRU with Word-Level Attention (WLA) proposed by Zhou et al. (2016) as our base model. We compare the performance of BGRU, BGRU+STP, and BGRU+SDP.

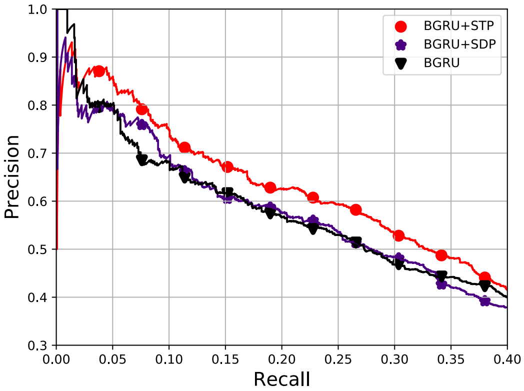

Figure 3: PR curves for BGRU, BGRU+SDP and BGRU+STP.

From Figure 3, we can observe that the model with the STP performs best, and the SDP model obtains an even worse result than the pure one. The PR curve areas of BGRU+SDP and BGRU are about 0.332 and 0.337 respectively, while BGRU+STP increases it to 0.366. The result indicates: (1) Our STP can get rid of irrelevant words in each instance and obtain more precise sentence representation for relation extraction. It proves that our STP module is effective. (2) The SDP method is not appropriate to handle low-quality sentences where key relation words are not in the SDP.

Effect of Entity-wise Attention

Test Settings	PR Curve Area	
Dataset	Original Data	STP Data
BGRU	0.337	0.366
-WLA+EWA	0.365	0.375
+EWA	**0.372**	**0.383**

Table 2: PR curve areas for BGRU, BGRU-WLA+EWA and BGRU+EWA on various datasets.

To evaluate the effect of entity-wise attention combined with word-level attention, we utilize BGRU in three settings on our tree parsed data and original data. One setting is to use WLA mechanism only (BGRU). The second one is to replace WLA with the Entity-Wise Attention (EWA) mechanism (BGRU-WLA+EWA). The third one is to incorporate two mechanisms (BGRU+EWA).

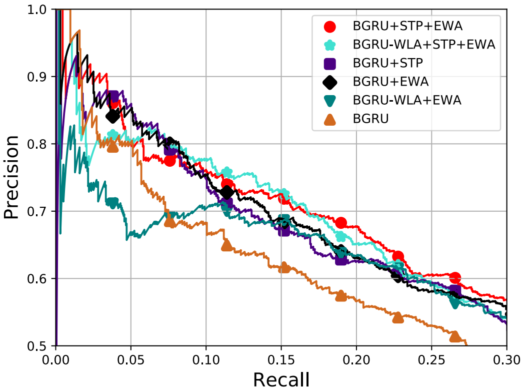

Figure 4: PR curves for BGRU, BGRU-WLA+EWA and BGRU+EWA on various datasets.

From Table 2 and Figure 4, we can obtain: (1) Regardless of the dataset that we employ, BGRU-WLA(+STP)+EWA outperforms BGRU(+STP). To be more specific, the PR curve area has a relative improvement of over 2.3%, which demonstrates that entity-wise hidden states in the BGRU present more precise relational features than other word states. (2) BGRU(+STP)+EWA achieves further improvements and outperforms the baseline by over 4.6%, because it considers more information than entity or relational words alone. Thus, it indicates that entity words are essential for relation extraction, but they can not represent features of the whole sentence without other words.

Effect of Parameter-Transfer Initializer

To evaluate the effect of the parameter-transfer initializer in our model, we leverage BGRU under four circumstances. The first one is to directly apply it on the original dataset. The second one tests BGRU combined with Transfer Learning (TL) on the original dataset. The third one uses BGRU on our STP dataset. The fourth one examines BGRU+TL on our STP dataset.

From Figure 5, we can conclude: (1) Regardless of the dataset that we use, models with TL achieve better performance, which improve the PR curve area by over 4.7%. It demonstrates that

Test Settings	One				Two				All			
P@N	100	200	300	Mean	100	200	300	Mean	100	200	300	Mean
Mintz	35.0	37.5	37.3	36.6	51.0	42.0	43.3	45.4	54.0	50.5	45.3	49.9
MultiR	64.0	61.5	53.7	59.7	62.0	61.5	58.7	61.1	75.0	65.0	62.0	67.3
MIML	62.0	59.0	54.7	58.6	69.0	59.5	59.0	62.5	70.0	64.5	60.3	64.9
PCNN	73.3	64.8	56.8	65.0	70.3	67.2	63.1	66.9	72.3	69.7	64.1	68.7
PCNN+ATT	78.0	68.0	60.7	68.9	75.0	74.0	66.3	71.8	82.0	74.0	69.0	75.0
BGRU	72.0	62.5	59.0	64.5	70.0	64.0	64.7	66.2	74.0	68.0	65.0	69.0
+STP	73.0	63.0	60.7	65.6	83.0	72.5	68.0	74.5	86.0	76.0	70.3	77.4
+EWA	82.0	71.5	66.3	73.3	84.0	79.5	70.3	77.9	86.0	81.5	75.3	80.9
+TL	**83.0**	**75.5**	**67.0**	**75.2**	**85.0**	**81.0**	**72.3**	**79.4**	**87.0**	**83.0**	**78.0**	**82.7**

Table 3: P@N for relation extraction in the entity pairs with different number of sentences

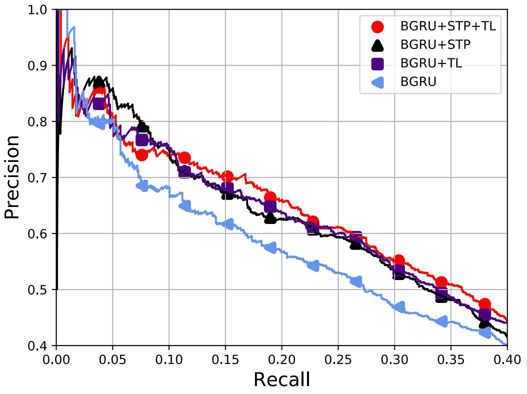

Figure 5: PR curves for BGRU, BGRU+TL, BGRU+STP and BGRU+STP+TL

transfer learning helps our model become more robust against noise. (2) BGRU+STP+TL achieves the best performance and increases the area to 0.383, while areas of BGRU, BGRU+STP and BGRU+TL are 0.337, 0.366 and 0.372 respectively. It means that the TL method works well with the STP and can resist noisy words further.

4.4 Comparison with Baselines

To evaluate our approach, we select the following six methods as our baseline:

Mintz (Mintz et al., 2009) proposes the human-designed feature model.

MultiR (Hoffmann et al., 2011) puts forward a graphical model.

MIML (Surdeanu et al., 2012) proposes a multi-instance multi-label model.

PCNN (Zeng et al., 2015) puts forward a piecewise CNN for relation extraction.

PCNN+ATT (Lin et al., 2016) proposes the selective attention mechanism with PCNN.

BGRU (Zhou et al., 2016) proposes a BGRU with the word-level attention mechanism.

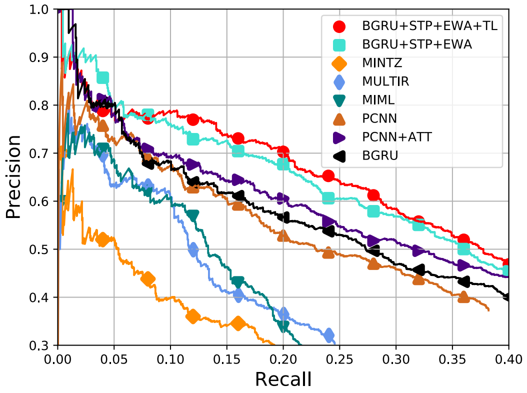

Figure 6: Performance comparison of the proposed method with baselines.

As Figure 6 shows, we can observe: (1) BGRU+STP+EWA achieves the best PR curve over baselines, which improves the area to 0.38 over 0.33 of PCNN, 0.34 of BGRU and 0.35 of PCNN+ATT. At the recall rate of 0.25, our model can still achieve a precision rate above 0.6. It demonstrates that BGRU+STP+EWA is effective because the STP and entity-wise attention combined with word-level attention can reduce inner-sentence noise at a fine-grained level. (2) Integrated with transfer learning, BGRU+STP+EWA+TL performs much better and increases the PR curve area to 0.392. It means that the model is pre-trained for better parameter initialization so the TL model becomes more robust against noisy words. Parameter transfer learning can be applied in better feature extractors for further improvement.

Following previous works, we adopt P@N as a quantitative indicator to compare our model with baselines based on various instances under each relational tuple. In Table 3, we report P@100, P@200, P@300 and the mean of them for each model in the held-out evaluation. We can find: (1) Compared with baselines, BGRU+STP+EWA+TL achieves the best performance in all test settings, which increases the performance of PCNN+ATT in three settings by 6.3%, 7.6%, and 7.7% respectively. It demonstrates that the integrated model is the most effective; (2) Our STP and entity-wise attention combined with word-level attention reduce inner-sentence noise effectively, and outperform baselines by over 5%; (3) Our neural extractor initialized with a priori knowledge learned from entity type classification is more robust against word-level noise where BGRU+STP+EWA+TL has an improvement of 2% over BGRU+STP+EWA.

5 Conclusion

In this paper, we propose a novel word-level approach for distant supervised relation extraction. It aims at tackling the low-quality corpus by reducing inner-sentence noise and improving the robustness against noisy words. To alleviate the influence of word-level noise, we propose the STP. Meanwhile, entity-wise attention combined with word-level attention helps the model focus more on relational words. Furthermore, parameter transfer learning makes our model more robust against noise by reasonable initialization of parameters. The experimental results show that our model significantly and consistently outperforms the state-of-the-art method.

In the future, we will incorporate the SDP and STP to obtain more precise shortened sentences. Furthermore, we will conduct research in how to utilize entity information to assign more appropriate initial parameters of the relation extractor.

Acknowledgments

This work is supported by FDCT 0007/2018/A1, DCT-MoST Joint-project No. (025/2015/AMJ) of SAR Macau; University of Macau Funds Nos: CPG2018-00032-FST & SRG2018-00111-FST; Chinese National Research Fund (NSFC) Key Project No. 61532013 and National China 973 Project No. 2015CB352401.

References

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. 2014. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1556–1567. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. ACM.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 5779–5786.

Xianpei Han and Le Sun. 2016. Global distant supervision for relation extraction. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 2950–2956.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 541–550. Association for Computational Linguistics.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3060–3066.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Wataru Kumagai. 2016. Learning bound for parameter transfer learning. In *Advances in Neural Information Processing Systems*, pages 2721–2729.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133. Association for Computational Linguistics.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1790–1795. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1116. Association for Computational Linguistics.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Lorien Y Pratt. 1993. Discriminability-based transfer between neural networks. In *Advances in Neural Information Processing Systems*, pages 204–211.

Pengda Qin, Weiran XU, and William Yang Wang. 2018a. Dsgan: Generative adversarial training for distant supervision relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505. Association for Computational Linguistics.

Pengda Qin, Weiran XU, and William Yang Wang. 2018b. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2137–2147. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 455–465. Association for Computational Linguistics.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1785–1794. Association for Computational Linguistics.

Wenmian Yang, Na Ruan, Wenyuan Gao, Kun Wang, Wensheng Ran, and Weijia Jia. 2017. Crowdsourced time-sync video tagging using semantic association graph. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 547–552. IEEE.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1753–1762. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212. Association for Computational Linguistics.

Graph Convolution over Pruned Dependency Trees Improves Relation Extraction

Yuhao Zhang,* Peng Qi,* Christopher D. Manning
Stanford University
Stanford, CA 94305
`{yuhaozhang, pengqi, manning}@stanford.edu`

Abstract

Dependency trees help relation extraction models capture long-range relations between words. However, existing dependency-based models either neglect crucial information (e.g., negation) by pruning the dependency trees too aggressively, or are computationally inefficient because it is difficult to parallelize over different tree structures. We propose an extension of graph convolutional networks that is tailored for relation extraction, which pools information over arbitrary dependency structures efficiently in parallel. To incorporate relevant information while maximally removing irrelevant content, we further apply a novel pruning strategy to the input trees by keeping words immediately around the shortest path between the two entities among which a relation might hold. The resulting model achieves state-of-the-art performance on the large-scale TACRED dataset, outperforming existing sequence and dependency-based neural models. We also show through detailed analysis that this model has complementary strengths to sequence models, and combining them further improves the state of the art.

1 Introduction

Relation extraction involves discerning whether a relation exists between two entities in a sentence (often termed *subject* and *object*, respectively). Successful relation extraction is the cornerstone of applications requiring relational understanding of unstructured text on a large scale, such as question answering (Yu et al., 2017), knowledge base population (Zhang et al., 2017), and biomedical knowledge discovery (Quirk and Poon, 2017).

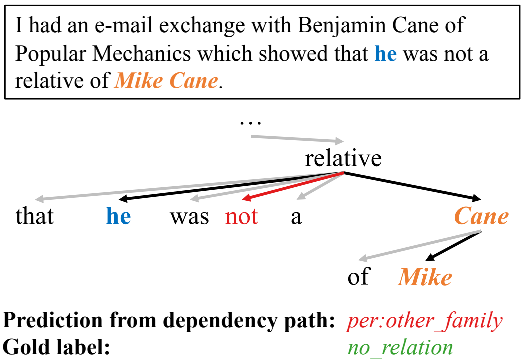

Figure 1: An example modified from the TAC KBP challenge corpus. A subtree of the original UD dependency tree between the subject ("he") and object ("Mike Cane") is also shown, where the shortest dependency path between the entities is highlighted in bold. Note that negation ("not") is off the dependency path.

Models making use of dependency parses of the input sentences, or *dependency-based models*, have proven to be very effective in relation extraction, because they capture long-range syntactic relations that are obscure from the surface form alone (e.g., when long clauses or complex scoping are present). Traditional feature-based models are able to represent dependency information by featurizing dependency trees as overlapping paths along the trees (Kambhatla, 2004). However, these models face the challenge of sparse feature spaces and are brittle to lexical variations. More recent neural models address this problem with distributed representations built from their computation graphs formed along parse trees. One common approach to leverage dependency information is to perform bottom-up or top-down computation along the parse tree or the subtree below the lowest common ancestor (LCA) of the entities (Miwa and Bansal, 2016). Another popular approach, inspired by Bunescu and Mooney (2005), is to reduce the parse tree to the *shortest dependency path* between the entities (Xu et al., 2015a,b).

However, these models suffer from several

---

[*] Equal contribution. The order of authorship was decided by a tossed coin.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2205–2215
Brussels, Belgium, October 31 - November 4, 2018.

drawbacks. Neural models operating directly on parse trees are usually difficult to parallelize and thus computationally inefficient, because aligning trees for efficient batch training is usually nontrivial. Models based on the shortest dependency path between the subject and object are computationally more efficient, but this simplifying assumption has major limitations as well. Figure 1 shows a real-world example where crucial information (i.e., negation) would be excluded when the model is restricted to only considering the dependency path.

In this work, we propose a novel extension of the graph convolutional network (Kipf and Welling, 2017; Marcheggiani and Titov, 2017) that is tailored for relation extraction. Our model encodes the dependency structure over the input sentence with efficient graph convolution operations, then extracts entity-centric representations to make robust relation predictions. We also apply a novel *path-centric pruning* technique to remove irrelevant information from the tree while maximally keeping relevant content, which further improves the performance of several dependency-based models including ours.

We test our model on the popular SemEval 2010 Task 8 dataset and the more recent, larger TACRED dataset. On both datasets, our model not only outperforms existing dependency-based neural models by a significant margin when combined with the new pruning technique, but also achieves a 10–100x speedup over existing tree-based models. On TACRED, our model further achieves the state-of-the-art performance, surpassing a competitive neural sequence model baseline. This model also exhibits complementary strengths to sequence models on TACRED, and combining these two model types through simple prediction interpolation further improves the state of the art.

To recap, our main contributions are: (i) we propose a neural model for relation extraction based on graph convolutional networks, which allows it to efficiently pool information over arbitrary dependency structures; (ii) we present a new path-centric pruning technique to help dependency-based models maximally remove irrelevant information without damaging crucial content to improve their robustness; (iii) we present detailed analysis on the model and the pruning technique, and show that dependency-based models have complementary strengths with sequence models.

2 Models

In this section, we first describe graph convolutional networks (GCNs) over dependency tree structures, and then we introduce an architecture that uses GCNs at its core for relation extraction.

2.1 Graph Convolutional Networks over Dependency Trees

The graph convolutional network (Kipf and Welling, 2017) is an adaptation of the convolutional neural network (LeCun et al., 1998) for encoding graphs. Given a graph with n nodes, we can represent the graph structure with an $n \times n$ adjacency matrix $\mathbf{A}$ where $A_{ij} = 1$ if there is an edge going from node i to node j. In an L-layer GCN, if we denote by $h_i^{(l-1)}$ the input vector and $h_i^{(l)}$ the output vector of node i at the l-th layer, a graph convolution operation can be written as

$$h_i^{(l)} = \sigma\big(\sum_{j=1}^{n} A_{ij} W^{(l)} h_j^{(l-1)} + b^{(l)}\big), \qquad (1)$$

where $W^{(l)}$ is a linear transformation, $b^{(l)}$ a bias term, and σ a nonlinear function (e.g., ReLU). Intuitively, during each graph convolution, each node gathers and summarizes information from its neighboring nodes in the graph.

We adapt the graph convolution operation to model dependency trees by converting each tree into its corresponding adjacency matrix $\mathbf{A}$, where $A_{ij} = 1$ if there is a dependency edge between tokens i and j. However, naively applying the graph convolution operation in Equation (1) could lead to node representations with drastically different magnitudes, since the degree of a token varies a lot. This could bias our sentence representation towards favoring high-degree nodes regardless of the information carried in the node (see details in Section 2.2). Furthermore, the information in $h_i^{(l-1)}$ is never carried over to $h_i^{(l)}$, since nodes never connect to themselves in a dependency tree.

We resolve these issues by normalizing the activations in the graph convolution before feeding it through the nonlinearity, and adding self-loops to each node in the graph:

$$h_i^{(l)} = \sigma\big(\sum_{j=1}^{n} \tilde{A}_{ij} W^{(l)} h_j^{(l-1)} / d_i + b^{(l)}\big), \qquad (2)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ with $\mathbf{I}$ being the $n \times n$ identity matrix, and $d_i = \sum_{j=1}^{n} \tilde{A}_{ij}$ is the degree of token i in the resulting graph.

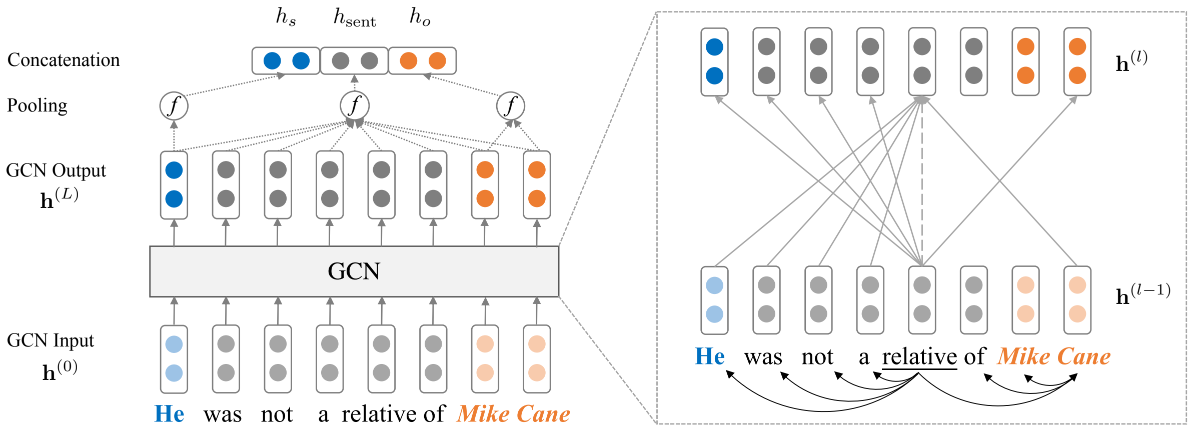

Figure 2: Relation extraction with a graph convolutional network. The left side shows the overall architecture, while on the right side, we only show the detailed graph convolution computation for the word "relative" for clarity. A full unlabeled dependency parse of the sentence is also provided for reference.

Stacking this operation over L layers gives us a deep GCN network, where we set $h_1^{(0)}, \ldots, h_n^{(0)}$ to be input word vectors, and use $h_1^{(L)}, \ldots, h_n^{(L)}$ as output word representations. All operations in this network can be efficiently implemented with matrix multiplications, making it ideal for batching computation over examples and running on GPUs. Moreover, the propagation of information between tokens occurs in parallel, and the runtime does not depend on the depth of the dependency tree.

Note that the GCN model presented above uses the same parameters for all edges in the dependency graph. We also experimented with: (1) using different transformation matrices W for top-down, bottom-up, and self-loop edges; and (2) adding dependency relation-specific parameters for edge-wise gating, similar to (Marcheggiani and Titov, 2017). We found that modeling directions does not lead to improvement,[1] and adding edge-wise gating further hurts performance. We hypothesize that this is because the presented GCN model is usually already able to capture dependency edge patterns that are informative for classifying relations, and modeling edge directions and types does not offer additional discriminative power to the network before it leads to overfitting. For example, the relations entailed by "*A*'s son, *B*" and "*B*'s son, *A*" can be readily distinguished with "'s" attached to different entities, even when edge directionality is not considered.

2.2 Encoding Relations with GCN

We now formally define the task of relation extraction. Let $\mathcal{X} = [x_1, ..., x_n]$ denote a sentence, where x_i is the i^{th} token. A subject entity and an object entity are identified and correspond to two spans in the sentence: $\mathcal{X}_s = [x_{s_1}, \ldots, x_{s_2}]$ and $\mathcal{X}_o = [x_{o_1}, \ldots, x_{o_2}]$. Given $\mathcal{X}$, $\mathcal{X}_s$, and $\mathcal{X}_o$, the goal of relation extraction is to predict a relation $r \in \mathcal{R}$ (a predefined relation set) that holds between the entities or "no relation" otherwise.

After applying an L-layer GCN over word vectors, we obtain hidden representations of each token that are directly influenced by its neighbors no more than L edges apart in the dependency tree. To make use of these word representations for relation extraction, we first obtain a sentence representation as follows (see also Figure 2 left):

$$h_{\text{sent}} = f\big(\mathbf{h}^{(L)}\big) = f\big(\text{GCN}(\mathbf{h}^{(0)})\big), \quad (3)$$

where $\mathbf{h}^{(l)}$ denotes the collective hidden representations at layer l of the GCN, and $f : \mathbb{R}^{d \times n} \to \mathbb{R}^d$ is a max pooling function that maps from n output vectors to the sentence vector.

We also observe that information close to entity tokens in the dependency tree is often central to relation classification. Therefore, we also obtain a subject representation h_s from $\mathbf{h}^{(L)}$ as follows

$$h_s = f\big(\mathbf{h}^{(L)}_{s_1:s_2}\big), \quad (4)$$

as well as an object representation h_o similarly.

Inspired by recent work on relational learning between entities (Santoro et al., 2017; Lee et al., 2017), we obtain the final representation used for classification by concatenating the sentence and the entity representations, and feeding them

---

[1] We therefore treat the dependency graph as undirected, i.e. $\forall i, j, A_{ij} = A_{ji}$.

through a feed-forward neural network (FFNN):

$$h_{\text{final}} = \text{FFNN}\big([h_{\text{sent}}; h_s; h_o]\big). \quad (5)$$

This h_{final} representation is then fed into a linear layer followed by a softmax operation to obtain a probability distribution over relations.

2.3 Contextualized GCN

The network architecture introduced so far learns effective representations for relation extraction, but it also leaves a few issues inadequately addressed. First, the input word vectors do not contain contextual information about word order or disambiguation. Second, the GCN highly depends on a correct parse tree to extract crucial information from the sentence (especially when pruning is performed), while existing parsing algorithms produce imperfect trees in many cases.

To resolve these issues, we further apply a Contextualized GCN (C-GCN) model, where the input word vectors are first fed into a bi-directional long short-term memory (LSTM) network to generate contextualized representations, which are then used as $\mathbf{h}^{(0)}$ in the original model. This BiLSTM contextualization layer is trained jointly with the rest of the network. We show empirically in Section 5 that this augmentation substantially improves the performance over the original model.

We note that this relation extraction model is conceptually similar to graph kernel-based models (Zelenko et al., 2003), in that it aims to utilize local dependency tree patterns to inform relation classification. Our model also incorporates crucial off-path information, which greatly improves its robustness compared to shortest dependency path-based approaches. Compared to tree-structured models (e.g., Tree-LSTM (Tai et al., 2015)), it not only is able to capture more global information through the use of pooling functions, but also achieves substantial speedup by not requiring recursive operations that are difficult to parallelize. For example, we observe that on a Titan Xp GPU, training a Tree-LSTM model over a minibatch of 50 examples takes 6.54 seconds on average, while training the original GCN model takes only 0.07 seconds, and the C-GCN model 0.08 seconds.

3 Incorporating Off-path Information with Path-centric Pruning

Dependency trees provide rich structures that one can exploit in relation extraction, but most of the information pertinent to relations is usually contained within the subtree rooted at the lowest common ancestor (LCA) of the two entities. Previous studies (Xu et al., 2015b; Miwa and Bansal, 2016) have shown that removing tokens outside this scope helps relation extraction by eliminating irrelevant information from the sentence. It is therefore desirable to combine our GCN models with tree pruning strategies to further improve performance. However, pruning too aggressively (e.g., keeping only the dependency path) could lead to loss of crucial information and conversely hurt robustness. For instance, the negation in Figure 1 is neglected when a model is restricted to only looking at the dependency path between the entities. Similarly, in the sentence "*She was diagnosed with cancer last year, and succumbed this June*", the dependency path *She←diagnosed→cancer* is not sufficient to establish that *cancer* is the cause of death for the subject unless the conjunction dependency to *succumbed* is also present.

Motivated by these observations, we propose *path-centric pruning*, a novel technique to incorporate information off the dependency path. This is achieved by including tokens that are up to distance K away from the dependency path in the LCA subtree. $K = 0$, corresponds to pruning the tree down to the path, $K = 1$ keeps all nodes that are directly attached to the path, and $K = \infty$ retains the entire LCA subtree. We combine this pruning strategy with our GCN model, by directly feeding the pruned trees into the graph convolutional layers.[2] We show that pruning with $K = 1$ achieves the best balance between including relevant information (e.g., negation and conjunction) and keeping irrelevant content out of the resulting pruned tree as much as possible.

4 Related Work

At the core of fully-supervised and distantly-supervised relation extraction approaches are statistical classifiers, many of which find syntactic information beneficial. For example, Mintz et al. (2009) explored adding syntactic features to a statistical classifier and found them to be useful when sentences are long. Various kernel-based approaches also leverage syntactic information to measure similarity between training and test examples to predict the relation, finding that tree-

---

[2] For our C-GCN model, the LSTM layer still operates on the full sentence regardless of the pruning.

based kernels (Zelenko et al., 2003) and dependency path-based kernels (Bunescu and Mooney, 2005) are effective for this task.

Recent studies have found neural models effective in relation extraction. Zeng et al. (2014) first applied a one-dimensional convolutional neural network (CNN) with manual features to encode relations. Vu et al. (2016) showed that combining a CNN with a recurrent neural network (RNN) through a voting scheme can further improve performance. Zhou et al. (2016) and Wang et al. (2016) proposed to use attention mechanisms over RNN and CNN architectures for this task.

Apart from neural models over word sequences, incorporating dependency trees into neural models has also been shown to improve relation extraction performance by capturing long-distance relations. Xu et al. (2015b) generalized the idea of dependency path kernels by applying a LSTM network over the shortest dependency path between entities. Liu et al. (2015) first applied a recursive network over the subtrees rooted at the words on the dependency path and then applied a CNN over the path. Miwa and Bansal (2016) applied a Tree-LSTM (Tai et al., 2015), a generalized form of LSTM over dependency trees, in a joint entity and relation extraction setting. They found it to be most effective when applied to the subtree rooted at the LCA of the two entities.

More recently, Adel et al. (2016) and Zhang et al. (2017) have shown that relatively simple neural models (CNN and augmented LSTM, respectively) can achieve comparable or superior performance to dependency-based models when trained on larger datasets. In this paper, we study dependency-based models in depth and show that with a properly designed architecture, they can outperform and have complementary advantages to sequence models, even in a large-scale setting.

Finally, we note that a technique similar to path-centric pruning has been applied to reduce the space of possible arguments in semantic role labeling (He et al., 2018). The authors showed pruning words too far away from the path between the predicate and the root to be beneficial, but reported the best pruning distance to be 10, which almost always retains the entire tree. Our method differs in that it is applied to the shortest dependency path between entities, and we show that in our technique the best pruning distance is 1 for several dependency-based relation extraction models.

5 Experiments

5.1 Baseline Models

We compare our models with several competitive dependency-based and neural sequence models.

Dependency-based models. In our main experiments we compare with three types of dependency-based models. (1) A logistic regression (LR) classifier which combines dependency-based features with other lexical features. (2) Shortest Dependency Path LSTM (SDP-LSTM) (Xu et al., 2015b), which applies a neural sequence model on the shortest path between the subject and object entities in the dependency tree. (3) Tree-LSTM (Tai et al., 2015), which is a recursive model that generalizes the LSTM to arbitrary tree structures. We investigate the child-sum variant of Tree-LSTM, and apply it to the dependency tree (or part of it). In practice, we find that modifying this model by concatenating dependency label embeddings to the input of forget gates improves its performance on relation extraction, and therefore use this variant in our experiments. Earlier, our group compared (1) and (2) with sequence models (Zhang et al., 2017), and we report these results; for (3) we report results with our own implementation.

Neural sequence model. Our group presented a competitive sequence model that employs a position-aware attention mechanism over LSTM outputs (PA-LSTM), and showed that it outperforms several CNN and dependency-based models by a substantial margin (Zhang et al., 2017). We compare with this strong baseline, and use its open implementation in further analysis.[3]

5.2 Experimental Setup

We conduct experiments on two relation extraction datasets: (1) **TACRED**: Introduced in (Zhang et al., 2017), TACRED contains over 106k mention pairs drawn from the yearly TAC KBP[4] challenge. It represents 41 relation types and a special *no_relation* class when the mention pair does not have a relation between them within these categories. Mentions in TACRED are typed, with subjects categorized into person and organization, and objects into 16 fine-grained types (e.g., date and location). We report micro-averaged F_1 scores on this dataset as is conventional. (2) **SemEval**

---

[3] https://github.com/yuhaozhang/tacred-relation

[4] https://tac.nist.gov/2017/KBP/index.html

System	P	R	F_1
LR[†] (Zhang+2017)	<u>73.5</u>	49.9	59.4
SDP-LSTM[†] (Xu+2015b)	66.3	52.7	58.7
Tree-LSTM[‡] (Tai+2015)	66.0	59.2	62.4
PA-LSTM[†] (Zhang+2017)	65.7	<u>64.5</u>	65.1
GCN	69.8	59.0	64.0
C-GCN	69.9	63.3	<u>66.4</u>*
GCN + PA-LSTM	71.7	63.0	67.1*
C-GCN + PA-LSTM	71.3	**65.4**	**68.2***

Table 1: Results on TACRED. Underscore marks highest number among single models; bold marks highest among all. † marks results reported in (Zhang et al., 2017); ‡ marks results produced with our implementation. $*$ marks statistically significant improvements over PA-LSTM with $p < .01$ under a bootstrap test.

System	*with-m*	*mask-m*
SVM[†] (Rink+2010)	82.2	–
SDP-LSTM[†] (Xu+2015b)	83.7	–
SPTree[†] (Miwa+2016)	84.4	–
PA-LSTM[‡] (Zhang+2017)	82.7	75.3
Our Model (C-GCN)	**84.8***	**76.5***

Table 2: F_1 scores on SemEval. † marks results reported in the original papers; ‡ marks results produced by using the open implementation. The last two columns show results from *with-mention* evaluation and *mask-mention* evaluation, respectively. $*$ marks statistically significant improvements over PA-LSTM with $p < .05$ under a bootstrap test.

2010 Task 8: The SemEval dataset is widely used in recent work, but is significantly smaller with 8,000 examples for training and 2,717 for testing. It contains 19 relation classes over untyped mention pairs: 9 directed relations and a special *Other* class. On SemEval, we follow the convention and report the official macro-averaged F_1 scores.

For fair comparisons on the TACRED dataset, we follow the evaluation protocol used in (Zhang et al., 2017) by selecting the model with the median dev F_1 from 5 independent runs and reporting its test F_1. We also use the same "entity mask" strategy where we replace each subject (and object similarly) entity with a special *SUBJ-<NER>* token. For all models, we also adopt the "multi-channel" strategy by concatenating the input word embeddings with POS and NER embeddings.

Traditionally, evaluation on SemEval is conducted without entity mentions masked. However, as we will discuss in Section 6.4, this method encourages models to overfit to these mentions and fails to test their actual ability to generalize. We therefore report results with two evaluation protocols: (1) *with-mention*, where mentions are kept for comparison with previous work; and (2) *mask-mention*, where they are masked to test the generalization of our model in a more realistic setting.

Due to space limitations, we report model training details in the supplementary material.

5.3 Results on the TACRED Dataset

We present our main results on the TACRED test set in Table 1. We observe that our GCN model outperforms all dependency-based models by at least 1.6 F_1. By using contextualized word representations, the C-GCN model further outperforms the strong PA-LSTM model by 1.3 F_1, and achieves a new state of the art. In addition, we find our model improves upon other dependency-based models in both precision and recall. Comparing the C-GCN model with the GCN model, we find that the gain mainly comes from improved recall. We hypothesize that this is because the C-GCN is more robust to parse errors by capturing local word patterns (see also Section 6.2).

As we will show in Section 6.2, we find that our GCN models have complementary strengths when compared to the PA-LSTM. To leverage this result, we experiment with a simple interpolation strategy to combine these models. Given the output probabilities $P_G(r|x)$ from a GCN model and $P_S(r|x)$ from the sequence model for any relation r, we calculate the interpolated probability as

$$P(r|x) = \alpha \cdot P_G(r|x) + (1 - \alpha) \cdot P_S(r|x)$$

where $\alpha \in [0, 1]$ is chosen on the dev set and set to 0.6. This simple interpolation between a GCN and a PA-LSTM achieves an F_1 score of 67.1, outperforming each model alone by at least 2.0 F_1. An interpolation between a C-GCN and a PA-LSTM further improves the result to 68.2.

5.4 Results on the SemEval Dataset

To study the generalizability of our proposed model, we also trained and evaluated our best C-GCN model on the SemEval test set (Table 2). We find that under the conventional *with-entity* evaluation, our C-GCN model outperforms all existing dependency-based neural models on this sep-

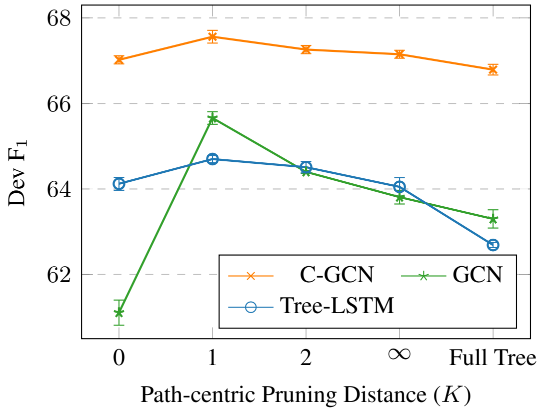

Figure 3: Performance of dependency-based models under different pruning strategies. For each model we show the F_1 score on the TACRED dev set averaged over 5 runs, and error bars indicate standard deviation of the mean estimate. $K = \infty$ is equivalent to using the subtree rooted at the LCA.

arate dataset. Notably, by properly incorporating off-path information, our model outperforms the previous shortest dependency path-based model (SDP-LSTM). Under the *mask-entity* evaluation, our C-GCN model also outperforms PA-LSTM by a substantial margin, suggesting its generalizability even when entities are not seen.

5.5 Effect of Path-centric Pruning

To show the effectiveness of path-centric pruning, we compare the two GCN models and the Tree-LSTM when the pruning distance K is varied. We experimented with $K \in \{0, 1, 2, \infty\}$ on the TACRED dev set, and also include results when the full tree is used. As shown in Figure 3, the performance of all three models peaks when $K = 1$, outperforming their respective dependency path-based counterpart ($K = 0$). This confirms our hypothesis in Section 3 that incorporating off-path information is crucial to relation extraction. Miwa and Bansal (2016) reported that a Tree-LSTM achieves similar performance when the dependency path and the LCA subtree are used respectively. Our experiments confirm this, and further show that the result can be improved by path-centric pruning with $K = 1$.

We find that all three models are less effective when the entire dependency tree is present, indicating that including extra information hurts performance. Finally, we note that contextualizing the GCN makes it less sensitive to changes in the tree structures provided, presumably because the model can use word sequence information in the LSTM layer to recover any off-path information that it needs for correct relation extraction.

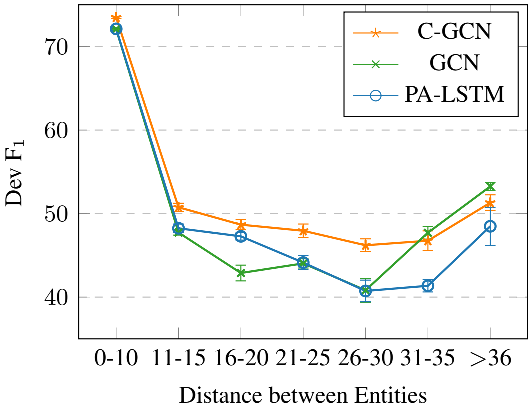

Figure 4: Dev set performance with regard to distance between the entities in the sentence for C-GCN, GCN and PA-LSTM. Error bars indicate standard deviation of the mean estimate over 5 runs.

Model	Dev F_1
Best C-GCN	67.4
– h_s, h_o, and Feedforward (FF)	66.4
– LSTM Layer	65.5
– Dependency tree structure	64.2
– FF, LSTM, and Tree	57.1
– FF, LSTM, Tree, and Pruning	47.4

Table 3: An ablation study of the best C-GCN model. Scores are median of 5 models.

6 Analysis & Discussion

6.1 Ablation Study

To study the contribution of each component in the C-GCN model, we ran an ablation study on the TACRED dev set (Table 3). We find that: (1) The entity representations and feedforward layers contribute 1.0 F_1. (2) When we remove the dependency structure (i.e., setting $\tilde{\mathbf{A}}$ to $\mathbf{I}$), the score drops by 3.2 F_1. (3) F_1 drops by 10.3 when we remove the feedforward layers, the LSTM component and the dependency structure altogether. (4) Removing the pruning (i.e., using full trees as input) further hurts the result by another 9.7 F_1.

6.2 Complementary Strengths of GCNs and PA-LSTMs

To understand what the GCN models are capturing and how they differ from a sequence model such as the PA-LSTM, we compared their performance

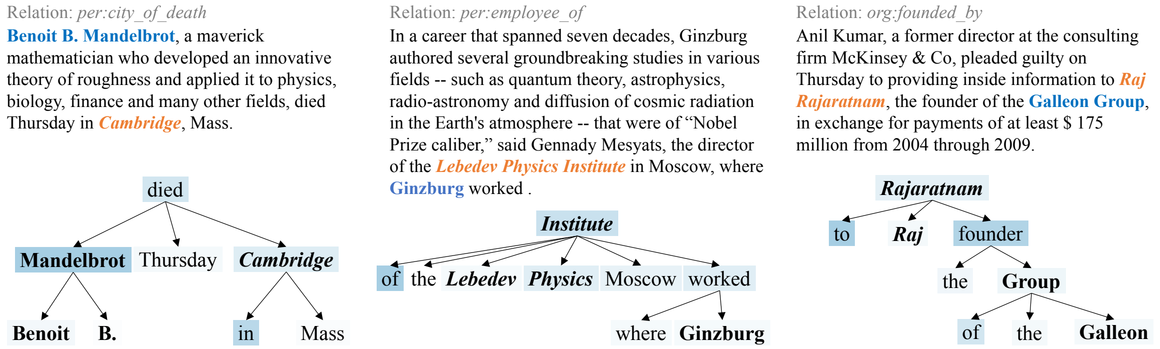

Relation: *per:city_of_death*

Benoit B. Mandelbrot, a maverick mathematician who developed an innovative theory of roughness and applied it to physics, biology, finance and many other fields, died Thursday in ***Cambridge***, Mass.

Relation: *per:employee_of*

In a career that spanned seven decades, Ginzburg authored several groundbreaking studies in various fields -- such as quantum theory, astrophysics, radio-astronomy and diffusion of cosmic radiation in the Earth's atmosphere -- that were of "Nobel Prize caliber," said Gennady Mesyats, the director of the ***Lebedev Physics Institute*** in Moscow, where **Ginzburg** worked .

Relation: *org:founded_by*

Anil Kumar, a former director at the consulting firm McKinsey & Co, pleaded guilty on Thursday to providing inside information to ***Raj Rajaratnam***, the founder of the **Galleon Group**, in exchange for payments of at least $ 175 million from 2004 through 2009.

Figure 5: Examples and the pruned dependency trees where the C-GCN predicted correctly. Words are shaded by the number of dimensions they contributed to h_{sent} in the pooling operation, with punctuation omitted.

Relation	Dependency Tree Edges		
per:children	`S-PER` ← son	son → `O-PER`	`S-PER` ← survived
per:other_family	`S-PER` ← stepson	niece → `O-PER`	`O-PER` ← stepdaughter
per:employee_of	a ← member	`S-PER` ← worked	`S-PER` ← played
per:schools_attended	`S-PER` ← graduated	`S-PER` ← earned	`S-PER` ← attended
org:founded	founded → `O-DATE`	established → `O-DATE`	was ← founded
org:number_of_employees	`S-ORG` ← has	`S-ORG` → employs	`O-NUMBER` ← employees
org:subsidiaries	`S-ORG` ← `O-ORG`	`S-ORG` → 's	`O-ORG` → division
org:shareholders	buffett ← `O-PER`	shareholder → `S-ORG`	largest ← shareholder

Table 4: The three dependency edges that contribute the most to the classification of different relations in the TACRED dev set. For clarity, we removed edges which 1) connect to common punctuation (i.e., commas, periods, and quotation marks), 2) connect to common prepositions (i.e., of, to, by), and 3) connect between tokens within the same entity. We use `PER`, `ORG` for entity types of `PERSON`, `ORGANIZATION`. We use `S-` and `O-` to denote subject and object entities, respectively. We also include edges for more relations in the supplementary material.

over examples in the TACRED dev set. Specifically, for each model, we trained it for 5 independent runs with different seeds, and for each example we evaluated the model's accuracy over these 5 runs. For instance, if a model correctly classifies an example for 3 out of 5 times, it achieves an accuracy of 60% on this example. We observe that on 847 (3.7%) dev examples, our C-GCN model achieves an accuracy at least 60% higher than that of the PA-LSTM, while on 629 (2.8%) examples the PA-LSTM achieves 60% higher. This complementary performance explains the gain we see in Table 1 when the two models are combined.

We further show that this difference is due to each model's competitive advantage (Figure 4): dependency-based models are better at handling sentences with entities farther apart, while sequence models can better leverage local word patterns regardless of parsing quality (see also Figure 6). We include further analysis in the supplementary material.

6.3 Understanding Model Behavior

To gain more insights into the C-GCN model's behavior, we visualized the partial dependency tree it is processing and how much each token's final representation contributed to h_{sent} (Figure 5). We find that the model often focuses on the dependency path, but sometimes also incorporates off-path information to help reinforce its prediction. The model also learns to ignore determiners (e.g., "the") as they rarely affect relation prediction.

To further understand what dependency edges contribute most to the classification of different relations, we scored each dependency edge by summing up the number of dimensions each of its connected nodes contributed to h_{sent}. We present the top scoring edges in Table 4. As can be seen in the table, most of these edges are associated with indicative nouns or verbs of each relation.[5]

6.4 Entity Bias in the SemEval Dataset

In our study, we observed a high correlation between the entity mentions in a sentence and its relation label in the SemEval dataset. We experimented with PA-LSTM models to analyze this

---

[5] We do notice the effect of dataset bias as well: the name "Buffett" is too often associated with contexts where shareholder relations hold, and therefore ranks top in that relation.

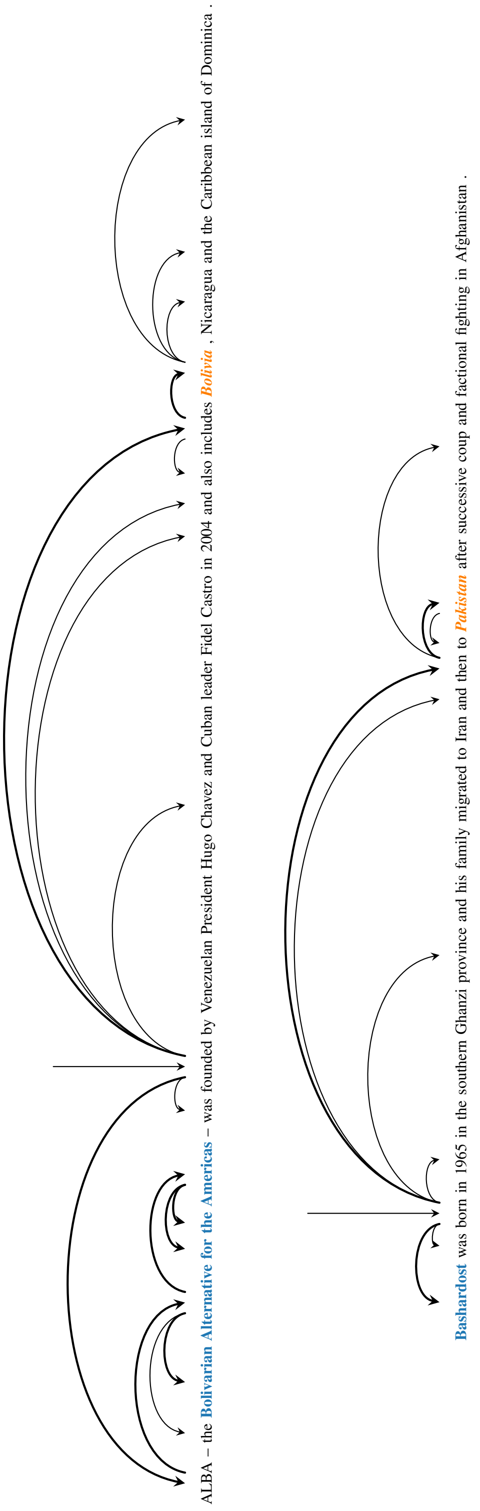

Figure 6: Dev set examples where either the C-GCN (upper) or the PA-LSTM (lower) predicted correctly in five independent runs. For each example, the predicted and pruned dependency tree corresponding to $K = 1$ in path-centric pruning is shown, and the shortest dependency path is thickened. We omit edges to punctuation for clarity. The first example shows that the C-GCN is effective at leveraging long-range dependencies while reducing noise with the help of pruning (while the PA-LSTM predicts *no_relation* twice, *org:alternate_names* twice, and *org:parents* once in this case). The second example shows that the PA-LSTM is better at leveraging the proximity of the word "migrated" regardless of attachment errors in the parse (while the C-GCN is misled to predict *per:country_of_birth* three times, and *no_relation* twice).

phenomenon.[6] We started by simplifying every sentence in the SemEval training and dev sets to "*subject* and *object*", where *subject* and *object* are the actual entities in the sentence. Surprisingly, a trained PA-LSTM model on this data is able to achieve 65.1 F_1 on the dev set if GloVe is used to initialize word vectors, and 47.9 dev F_1 even without GloVe initialization. To further evaluate the model in a more realistic setting, we trained one model with the original SemEval training set (unmasked) and one with mentions masked in the training set, following what we have done for TACRED (masked). While the unmasked model achieves a 83.6 F_1 on the original SemEval dev set, F_1 drops drastically to 62.4 if we replace dev set entity mentions with a special <*UNK*> token to simulate the presence of unseen entities. In contrast, the masked model is unaffected by unseen entity mentions and achieves a stable dev F_1 of 74.7. This suggests that models trained without entities masked generalize poorly to new examples with unseen entities. Our findings call for more careful evaluation that takes dataset biases into account in future relation extraction studies.

7 Conclusion

We showed the success of a neural architecture based on a graph convolutional network for relation extraction. We also proposed path-centric pruning to improve the robustness of dependency-based models by removing irrelevant content without ignoring crucial information. We showed through detailed analysis that our model has complementary strengths to sequence models, and that the proposed pruning technique can be effectively applied to other dependency-based models.

Acknowledgements

We thank Arun Chaganty, Kevin Clark, Sebastian Schuster, Ivan Titov, and the anonymous reviewers for their helpful suggestions. This material is based in part upon work supported by the National Science Foundation under Grant No. IIS-1514268. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

---

[6] We choose the PA-LSTM model because it is more amenable to our experiments with simplified examples.

References

Heike Adel, Benjamin Roth, and Hinrich Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT).*

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (EMNLP 2005)*, pages 724–731.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 Interactive poster and demonstration sessions*. Association for Computational Linguistics.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR 2017).*

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017).*

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL 2015).*

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017).*

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016).*

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, volume 14, pages 1532–1543.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. *Proceedings of the 15th Conference of the European Association for Computational Linguistics (EACL 2017).*

Bryan Rink and Sanda Harabagiu. 2010. UTD: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4974–4983.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL 2015).*

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT).*

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016).*

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1785–1794.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3:1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2014)*, pages 2335–2344.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, page 207.

Multi-Level Structured Self-Attentions for Distantly Supervised Relation Extraction

Jinhua Du[†], **Jingguang Han**[§], **Andy Way**[†], **Dadong Wan**[§]
[†]ADAPT Centre, School of Computing, Dublin City University, Ireland
[§]Accenture Labs Dublin, Ireland
{jinhua.du, andy.way}@adaptcentre.ie
{jingguang.han, dadong.wan}@accenture.com

Abstract

Attention mechanisms are often used in deep neural networks for distantly supervised relation extraction (DS-RE) to distinguish valid from noisy instances. However, traditional 1-D vector attention models are insufficient for the learning of different contexts in the selection of valid instances to predict the relationship for an entity pair. To alleviate this issue, we propose a novel multi-level structured (2-D matrix) self-attention mechanism for DS-RE in a multi-instance learning (MIL) framework using bidirectional recurrent neural networks. In the proposed method, a structured word-level self-attention mechanism learns a 2-D matrix where each row vector represents a weight distribution for different aspects of an instance regarding two entities. Targeting the MIL issue, the structured sentence-level attention learns a 2-D matrix where each row vector represents a weight distribution on selection of different valid instances. Experiments conducted on two publicly available DS-RE datasets show that the proposed framework with a multi-level structured self-attention mechanism significantly outperform state-of-the-art baselines in terms of PR curves, P@N and F1 measures.

1 Introduction

Relation extraction is a fundamental task in information extraction (IE), which studies the issue of predicting semantic relations between pairs of entities in a sentence (Zelenko et al., 2003; Bunescu and Mooney, 2005; Zhou et al., 2005). One crucial problem in RE is the relative lack of large-scale, high-quality labeled data. In recent years, one commonly used and effective technique for dealing with this challenge is the distant supervision method via knowledge bases (KBs) (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011), which assumes that if one entity pair appearing in some sentences can be observed in a KB with a certain relationship, then these sentences will be labeled as the context of this entity pair and this relationship. The distant supervision strategy is an effective and efficient method for automatically labeling large-scale training data. However, it also introduces a severe mislabelling problem due to the fact that a sentence that mentions two entities does not necessarily express their relation in a KB (Surdeanu et al., 2012; Zeng et al., 2015).

Plenty of research work has been proposed to deal with distantly supervised data and has achieved significant progress, especially with the rapid development of deep neural networks (DNN) for relation extraction in recent years (Zeng et al., 2014, 2015; Lin et al., 2016, 2017a; Wang et al., 2016; Zhou et al., 2016; Ji et al., 2017; Yang et al., 2017; Zeng et al., 2017). DNN models under an MIL framework for DS-RE have become state-of-the-art, replacing statistical methods, such as feature-based and graphical models (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). In the MIL framework for distantly supervised RE, each entity pair often has multiple instances where some are noisy and some are valid. The attention mechanism in DNNs, such as convolutional (CNN) and recurrent neural networks (RNN), is an effective way to select valid instances by learning a weight distribution over multiple instances. However, there are two important representation learning problems in DNN-based distantly supervised RE: (1) **Problem I**: entity pair-targeted context representation learning from an instance; and (2) **Problem II**: valid instance selection representation learning over multiple instances. The former can use a word-level attention mechanism to learn a weight distribution on words and then a weighted sentence representation regarding two entities; the latter can employ a sentence-level attention mechanism to

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2216–2225
Brussels, Belgium, October 31 - November 4, 2018.

learn a weight distribution on multiple instances so that valid sentences with higher weights can be focused and selected, and noisy instances with lower weights are suppressed.

Both the word-level and sentence-level attention mechanisms in previous work on the RE task are simple 1-D vectors which are learned using the hidden states of the RNN, or via pooling from either the RNNs' hidden states or convolved n-grams (Zeng et al., 2014, 2015; Zhou et al., 2016; Wang et al., 2016; Ji et al., 2017; Yang et al., 2017). The deficiency of the 1-D attention vector is that it only focuses on one or a small number of aspects of the sentence, or one or a small number of instances (Lin et al., 2017b), with the result that different semantic aspects of the sentence, or different multiple valid sentences are ignored, and cannot be utilised.

Inspired by the structured self-attentive sentence embedding in Lin et al. (2017b), we propose a novel multi-level structured (2-D) self-attention mechanism (MLSSA) in a bidirectional LSTM-based (BiLSTM) (Hochreiter and Schmidhuber, 1997) MIL framework to alleviate two **problems** in the distantly supervised RE. Regarding **Problem I**, we propose a 2-D matrix-based word-level attention mechanism, which contains multiple vectors, each focusing on different aspects of the sentence for better context representation learning. In terms of **Problem II**, we propose a 2-D sentence-level attention mechanism for multiple instance learning, where it contains multiple vectors, each focusing on different valid instances for a better sentence selection. "**structured**" indicates that the weight vectors in the learned 2-D matrix try to construct a structural dependency relationship by learning different weight distributions for different contexts or instances given the entity pair. We can see that our structured attention mechanism is different from that in Kim et al. (2017) which incorporates richer structural distributions and are simple extensions of the basic attention procedure. We verify the proposed framework on two distantly supervised RE datasets, namely the New York Times (NYT) dataset (Riedel et al., 2010) and the DBpedia Portuguese dataset (Batista et al., 2013). Experimental results show that our MLSSA framework significantly outperforms state-of-the-art baseline systems in terms of different evaluation metrics.

The main contributions of this paper include: (1) we propose a novel multi-level structured (2-D) self-attention mechanism for DS-RE which can make full use of input sequences to learn different contexts, without integrating extra resources; (2) we propose a 2-D matrix-based word-level attention for better context representation learning targeting two entities; (3) we propose a 2-D sentence-level attention mechanism over multiple instances to select different valid instances; and (4) we verify the proposed framework on two publicly available distantly supervised datasets.

2 Related Work

Most existing work on distant supervision data mainly focuses on denoising the data under the MIL strategy by learning a valid sentence representation or features, and then selecting one or more valid instances for relation classification (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016, 2017a; Zhou et al., 2016; Ji et al., 2017; Zeng et al., 2017; Yang et al., 2017).

Riedel et al. (2010) and Surdeanu et al. (2012) use a graphical model and MIL to select the valid sentences and classify the relations. However, these models are based on statistical methods and feature engineering, i.e. extracting sentence features using other NLP tools. Zeng et al. (2015) proposed a piece-wise CNN (PCNN) method to automatically learn sentence-level features and select one valid instance for the relation classification. The one-sentence-selection strategy does not make full use of the supervision information among multiple instances.

Lin et al. (2016) and Ji et al. (2017) introduce an attention mechanism to the PCNN-based MIL framework to select informative sentences, which outperforms all baseline systems on the NYT data set. However, their attention mechanism is only a sentence-level model without incorporating word-level attention. Zhou et al. (2016) introduce a word-level attention model to the BiLSTM-based MIL framework and obtain significant improvements on the SemEval2010 (Hendrickx et al., 2010) data set. Wang et al. (2016) extend the single word-level attention model to multiple word levels in CNNs to discern patterns in heterogeneous contexts of the input sentence, and achieve best performance on the SemEval2010 data set. However, these two works were not targeting the distantly supervised RE problem.

Yang et al. (2017) experiment with word-level and sentence-level attention models in the bidirectional RNN on the NYT dataset on the basis of the open source DS-RE system,[1] and verify that a two-level attention mechanism achieves best performance compared to PCNN/CNN models. Both the word-level and sentence-level attention models are 1-D vectors.

From previous work, we can see that the attention mechanism in DNNs has made significant progress on the RE task. However, both word-level and sentence-level attention models are still based on 1-D vectors which have the following insufficiencies: (1) although the 1-D attention model can learn weights for different contexts, it only focuses on one or very few aspects of a single sentence (Lin et al., 2017b), or one or very few instances; (2) in order to allow the attention mechanism to learn more aspects of the sentence, or different instances, extra knowledge needs to be integrated, such as the work in Ji et al. (2017) and Lin et al. (2017a). The former integrates entity descriptions generated from Freebase and Wikipedia as supplementary background knowledge to disambiguate the entity. The latter introduces a multilingual framework which employs a monolingual attention mechanism to utilize the information within monolingual texts, and further uses a cross-lingual attention mechanism to consider the information consistency and complementarity among cross-lingual texts. However, extra resources are difficult to obtain in many practical scenarios.

In order to alleviate the burden of integrating extra knowledge, and make full use of the input sentence (i.e. learning different aspects of context and focusing on different valid instances), we propose a multi-level structured self-attention mechanism in a BiLSTM-based MIL framework without integrating extra resources.

3 Approach

The distantly supervised RE can be formalised as follows: given an entity pair (e_1, e_2), a bag $\mathcal{G}$ containing $\mathcal{J}$ instances, and the relation label r for $\mathcal{G}$, the goal of the training process is to denoise these instances by selecting valid candidates based on r, and the goal of the testing process is to denoise multiple instances by selecting valid candidates to predict the relation r for $\mathcal{G}$.

To alleviate the aforementioned two problems, improving the following two representation learning issues is clearly important for a DNN-based RE classifier:

- *Entity pair-targeted context representation*: The model should have the capability to learn a better context representation from the input sentence targeting the entity pair;
- *Instance selection representation*: The model should have the capability to learn a better weight distribution over multiple instances to select valid instances regarding an entity pair.

Motivated by these two issues, we propose a multi-level structured self-attention framework.

3.1 Architecture

The proposed framework consists of three parts as shown in Figure 1. The first part includes the input layer, embedding layer and BiLSTM layer which transform the input sequence at different time steps to LSTM hidden states.

The second part implements the entity pair-targeted context representation learning, including:

- *a structured word-level self-attention layer*: this generates a set of summation weight vectors (or a 2-D matrix) taking the LSTM hidden states as input. Each vector in the 2-D matrix represents the weights for different aspects of the input sentence.
- *a structured context representation layer*: the weight vectors learned by the 2-D word-level self-attention are dotted with the BiLSTM hidden states. Accordingly, a 2-D matrix or a set of weighted LSTM hidden state vectors, denoted as "M_{L1}" in Figure 1, is obtained. Each weighted vector represents a sentence embedding reflecting a different aspect of the sentence targeting the entity pair. By this means, a dependency parsing-like structure of the input sentence can be constructed, obtaining different semantic representations of the sentence for the two entities in question.
- *a flattened representation layer*: this concatenates each vector in the 2-D matrix of the sentence embedding to one vector. Then, the

---

[1] `https://github.com/frankxu2004/TensorFlow-NRE`

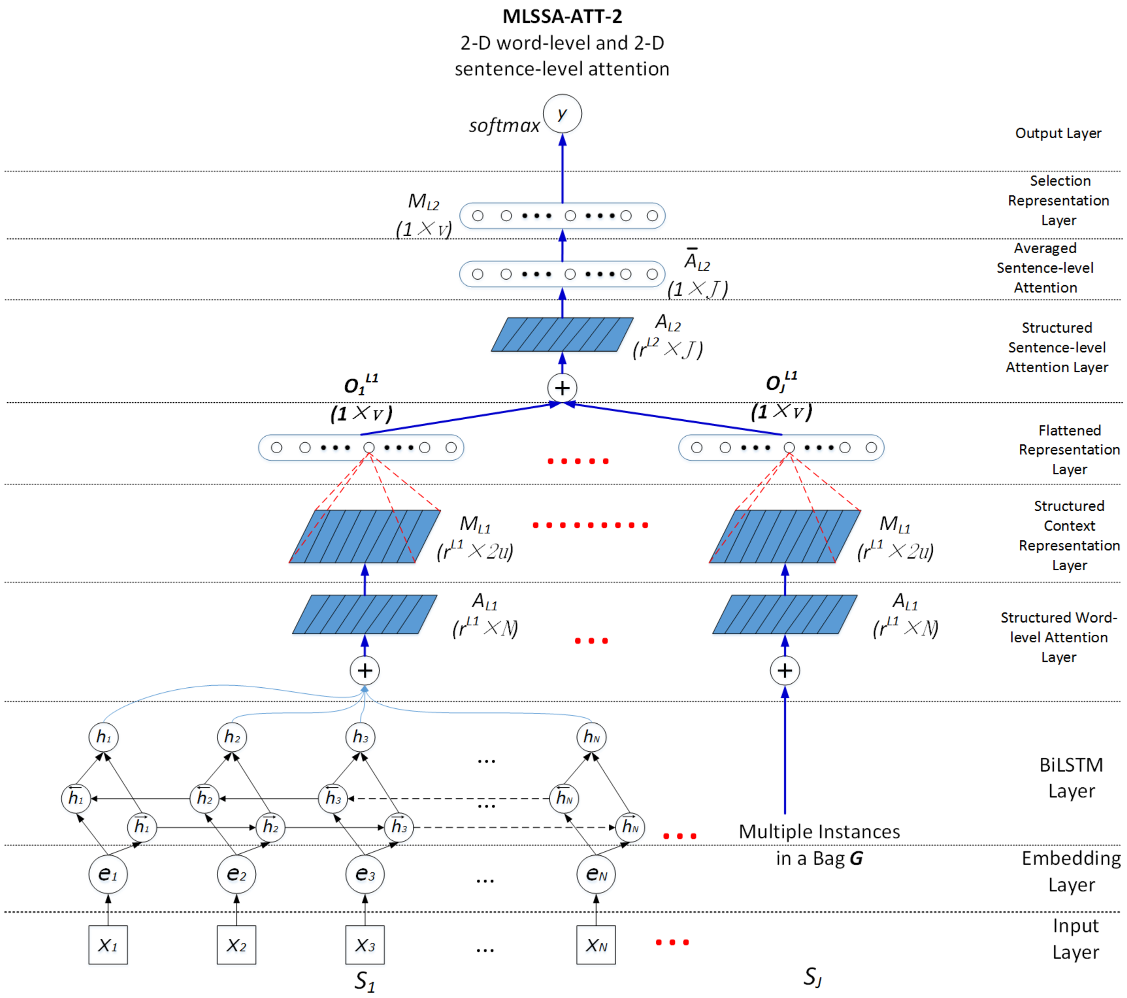

Figure 1: Multi-level structured self-attention framework for distantly supervised RE

flattened vector connects to a 1-layer multi-layer perceptron (MLP) with ReLU activation function, generating an aggregated sentence representation.

The first and second parts operate on the single instance level, i.e. given a bag $\mathcal{G}$ and feeding each instance into the framework, the structured word-level self-attention mechanism will construct J individual structured sentence representations corresponding to J input instances.

The third part targets the instance selection representation learning issue, and operates on the bag level, i.e. considering weighted context representations of all instances in the bag $\mathcal{G}$ and learning probability distributions to distinguish informative from noisy sentences. This part includes:

- *a structured sentence-level attention model*: this has a similar structure to the structured word-level attention mechanism, except that it generates a set of summation weight vectors for all input instances in the same bag $\mathcal{G}$. Each vector is a weight distribution over all instances. Accordingly, the 2-D sentence-level matrix is expected to learn a set of different weight distributions focusing on different informative instances. As a result, informative sentences are expected to contribute more with higher weights, and noisy sentences are expected to contribute less with smaller weights, to the relation classification.

- *an averaged sentence-level attention layer*: the 2-D sentence-level attention matrix is averaged and converted to a 1-D vector.

- *a selection representation layer*: the 1-D averaged attention vector is dotted with the output of the flattened representation layer. Accordingly, a 1-D vector, denoted as “M_{L2}” in Figure 1, is obtained which represents an averaged weighted selection representation of multiple sentences.

- *an output layer*: this connects to a *softmax* layer and produces a probability distribution corresponding to relation classes.

3.2 Structured Word-Level Self-Attention and its Penalisation Function

Given a bag $\mathcal{G} = (S_1, S_2, \ldots, S_J)$ containing J instances, and a sentence S_j in $\mathcal{G}$ consisting of N tokens, S_j can be represented using a sequence of word embeddings, as in (1):

$$S_j = (e_1, e_2, \ldots, e_N) \tag{1}$$

where e_i is a d-dimension vector for the i-th word, and S_j is the j-th instance in $\mathcal{G}$.

We denote the hidden state of the BiLSTM as in (2):

$$H = (\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_N})^T \tag{2}$$

where $\mathbf{h_t}$ is a concatenation of the forward hidden state $\overrightarrow{h}_t$ and the backward hidden state $\overleftarrow{h}_t$ at time step t. T is the transpose operation. If the size of each unidirectional LSTM is u, then H has the size $2u$-by-N.

Then, the structured word-level self-attention mechanism is defined as in (3):

$$A_{L1} = softmax(W_{s2}^{L1} tanh(W_{s1}^{L1} H)) \tag{3}$$

where $L1$ stands for the first-level attention mechanism, i.e. the word-level; W_{s1}^{L1} is a weight matrix of size $d_a^{L1} \times 2u$, where d_a^{L1} is a hyper-parameter for the number of neurons in the attention network; W_{s2}^{L1} is a weight matrix with the shape $r^{L1} \times d_a^{L1}$, where r^{L1} ($r^{L1} > 1$) is the hyper-parameter representing the size of multiple vectors in the 2-D attention matrix. The size of r^{L1} is defined based on how many different aspects of the sentence need to be focused on; A_{L1} is the annotation matrix of size $r^{L1} \times N$. We can see that in A_{L1}, there are r^{L1} attention vectors for the N-token input sentence.

Finally, we compute the r^{L1} weighted sums by multiplying the annotation matrix A_{L1} and BiLSTM hidden states H. The resulting structured sentence representation M_{L1} is (4):

$$M_{L1} = A_{L1} H^T \tag{4}$$

where M_{L1} has the shape $r^{L1} \times 2u$. It can be seen that the traditional 1-D sentence representation is extended to a 2-D representation ($r^{L1} > 1$).

Subsequently, the output of the flattened representation layer for the instance S_j in $\mathcal{G}$ is (5):

$$O_j^{L1} = ReLU(W_o^{L1} M_{L1}^{FT} + b_o^{L1}) \tag{5}$$

where W_o^{L1} is the weight matrix that has the shape v-by-$r^{L1} * 2u$, where v is the amount of neurons in the $ReLU$-based MLP layer; M_{L1}^{FT} is the flattened structured sentence representation which is a concatenated vector of each row in M_{L1} and has the dimension $r^{L1} * 2u$; b_o^{L1} is the bias vector of size v; O_j^{L1} is the aggregated sentence representation of the j-th instance in the bag $\mathcal{G}$ with size v.

Then, the output of all instances in $\mathcal{G}$ from the flattened representation layer is denoted as in (6):

$$O^{L1} = (O_1^{L1}, O_2^{L1}, \ldots, O_J^{L1})^T \tag{6}$$

where O^{L1} has the shape of $v \times J$.

As in Lin et al. (2017b), the penalisation term for the structured word-level attention is as in (7):

$$P_{L1} = ||(A_{L1} A_{L1}^T - I)||_F^2 \tag{7}$$

where $|| \cdot ||_F$ is the Frobenius norm of a matrix. I is an identity matrix. Minimising this penalisation term means that we learn an orthogonal matrix for A_{L1} so that each row in A_{L1} only focuses on a single aspect of semantics.

3.3 Structured Sentence-Level Self-Attention and Averaged Selection Representation

Taking O^{L1} as the input to the structured 2-D sentence-level attention model, the annotation matrix A_{L2} is calculated as in (8):

$$A_{L2} = softmax(W_{s2}^{L2} tanh(W_{s1}^{L2} O^{L1})) \tag{8}$$

where W_{s1}^{L2} is the weight matrix of size $d_a^{L2} \times v$, and d_a^{L2} is the number of neurons in the attention network; W_{s2}^{L2} is the weight matrix of shape $r^{L2} \times d_a^{L2}$, where r^{L2} ($r^{L2} > 1$) is the hyper-parameter representing the size of multiple vectors in the 2-D sentence-level attention matrix. The r^{L2} multiple vectors are expected to focus on different informative instances for the relation classification; A_{L2} is the sentence-level annotation matrix of size $r^{L2} \times J$. We can see that the traditional 1-D sentence-level attention model is expanded to a multi-vector attention ($r^{L2} > 1$).

Then, we average the 2-D A_{L2} to a 1-D vector $\bar{A}_{L2}$ which has the dimension of J.

Accordingly, we calculate the averaged weighted sum by multiplying $\bar{A}_{L2}$ and the aggregated sentence representation O^{L1}, with the resulting instance selection representation M_{L2} being (9):

$$M_{L2} = \bar{A}_{L2} \cdot (O^{L1})^T \tag{9}$$

where M_{L2} has the size of v.

The probability distribution of the predicted relation type, i.e. the final output for relation prediction, can be calculated as in (10):

$$p(\hat{y}|G) = softmax(W_o^{L2} tanh(M_{L2}) + b_o^{L2}) \tag{10}$$

3.4 Loss Function and Optimisation

The total loss of the network is the summation of the penalisation term P_{L1}, *softmax* loss in Eq. (10) and the $L2$ regularisation loss.

We use the ADAM optimiser (Kingma and Ba, 2014) to minimize the loss function on the mini-batch basis which is randomly selected from the training set.

4 Experiments

4.1 Datasets

We use two distantly supervised datasets, namely the NYT corpus (NYT) and the DBpedia Portuguese dataset (PT),[2] to verify our method.

In the NYT dataset, there are 53 relationships including a special relation *NA* which indicates a *None Relation* between two entities. The training set contains 580,888 sentences, 292,484 entity pairs and 19,429 relational facts (Non-NA). The test set contains 172,448 sentences, 96,678 entity pairs and 1,950 relational facts (Non-NA). There are 19.24% and 22.57% entity pairs corresponding to multiple instances in the training set and test set, respectively.

The DBpedia Portuguese dataset is smaller, containing just 10 relationships including a special relation *Other*. After preprocessing the original dataset, we obtain 96,847 sentences, 85,528 entity pairs and 77,321 relational facts (Non-Other). There are 8.61% entity pairs corresponding to multiple instances in the whole dataset. As in Batista et al. (2013), we use two different settings for the training and test sets: (1) a manually reviewed subset that contains 602 sentences (PT-MANUAL) as the test set; and (2) 70%–30% out of the whole data as the training set and test set, respectively (PT-SPLIT).

4.2 Word Embeddings and Relative Position Features

For the NYT dataset, we use the 200-dimensional word vectors pre-trained using the NYT corpus;[3] for the PT dataset, we use a pre-trained 300-dimensional word vector model.[4] For the two-word entities in the data set, we use *underscore* to connect them as one word. The word embeddings of unknown words are intialised using the normal distribution with the standard deviation 0.05. Similar to previous work, we also use position embeddings specified by entity pairs. It is defined as the combination of the relative distances from the current word to head or tail entities (Zeng et al., 2014, 2015; Lin et al., 2016).

4.3 Baselines and Our MLSSA Systems

Neural RE systems have become the state-of-the-art, such as CNN-based (Zeng et al., 2014; Lin et al., 2017a), Piecewise CNN-based (Zeng et al., 2015; Lin et al., 2016; Ji et al., 2017), and BiLSTM-based (Zhou et al., 2016) models with or without an attention mechanism. In order to carry out a fair comparison, we select CNN+ATT, PCNN+ATT, BiGRU+ATT (bidirectional gated recurrent unit) and BiGRU+2ATT models as baselines on the NYT data, PCNN+ATT and BiGRU+2ATT as baselines on the PT data, where ATT indicates that the model has a sentence-level attention mechanism, and 2ATT indicates that the model has a 1-D word-level and a 1-D sentence-level attention.[5]

To show the incremental effectiveness of structured 2-D word-level and 2-D sentence-level self-attention mechanisms, we use two different settings for our MLSSA system: (1) **MLSSA-1**: this has a 2-D word-level self-attention and a 1-D sentence-level attention, i.e. A_{L2} in Figure 1 is a 1-D vector. This system is used to verify the context representation learning targeting **Problem I**;

---

[2] There are several reasons to use the Portuguese dataset: (i) the data sets reported in previous work, such as the KBP data, are not publicly available, or (ii) SemEval data sets which are not distantly supervised data. Google has also released a dataset (https://github.com/google-research-datasets/relation-extraction-corpus), but it is smaller and only has 4 relation types. For all these reasons, the Portuguese data is a better option to verify our method.

[3] `https://catalog.ldc.upenn.edu/ldc2008t19`

[4] `https://s3-us-west-1.amazonaws.com/fasttext-vectors/wiki.pt.vec`

[5] All the baseline systems are obtained from `https://github.com/thunlp/NRE` and `https://github.com/thunlp/TensorFlow-NRE`.

(2) **MLSSA-2**: both the word-level and sentence-level attentions are structured 2-D matrices. This system verifies the instance selection representation learning targeting **Problem II**.

4.4 Experiment Setup and Evaluation Metrics

Following previous work, we use different evaluation metrics on these two datasets. For the NYT dataset:

- Overall evaluation: all training data is used for the model training, and all test data is used for the evaluation in terms of Precision-Recall (PR) curves;
- P@N evaluation: we select those entity pairs that have more than one instance to carry out the comparison in terms of the *precision* at n (P@N) measure.[6] As in Lin et al. (2016), there are three settings: (1) One: for each testing entity pair corresponding to multiple instances, we randomly select one sentence to predict the relation; (2) Two: for each testing entity pair with multiple instances, we randomly select two sentences for the relation extraction; and (3) All: for each entity pair having multiple instances, we use all of them to predict the relation. Note that these three selections are only applied to the test set, and we keep all sentences in the training data for model building.

For the PT dataset, we use Macro F1 to evaluate system performance.[7]

4.5 Hyper-parameter Settings

We use cross-validation to determine the hyper-parameters of our system regarding two different settings and datasets. The in-common and different parameters for our two systems and two datasets are shown in Table 1.

Parameters for MLSSA-1/2	**NYT**	**PT**
Word embedding dimension d	200	300
Position embedding dimension	50	50
Batch size B	64	50
Time steps T	70	70
Learning rate λ	0.001	0.001
Hidden size in BiLSTM u	300	300
d_a^{L1} at word-level attention	300	300
r^{L1} at word-level attention	9	5
MLP size v	1000	1000
Coefficient of the penalisation term	1.0	1.0
Parameters for MLSSA-2 only	**NYT**	**PT**
d_a^{L2} at sentence-level attention	300	300
r^{L2} at sentence-level attention	9	3

Table 1: Hyper-parameter settings

4.6 PR Curves on NYT Dataset

The comparison results for the NYT test set are shown in Figure 2. We have the following observations: (1) BiGRU+ATT outperforms CNN+ATT and PCNN+ATT in terms of the PR curve, showing that it can learn a better semantic representation from the sequential input; (2) BiGRU+2ATT has better overall performance compared to BiGRU+ATT, showing that word-level attention is beneficial to sentence-level attention compared to single-attention models, i.e. the sentence-level attention model can select more informative sentences based on a more reasonable sentence embedding learned by the word-level attention model; (3) MLSSA-1 outperforms all baseline systems in terms of the PR curve, which demonstrates that the structured 2-D word-level attention model can learn a better sentence representation by focusing on different aspects of the sentence, so that the sentence-level attention has a better chance of selecting the most informative sentences; and (4) the PR curve of MLSSA-2 is higher than that of MLSSA-1, demonstrating that the 2-D sentence-level attention model can better select the most informative sentences compared to the 1-D sentence-level attention model targeting those entity pairs with multiple instances.

4.7 P@N Evaluation on NYT Dataset

The results on the NYT dataset regarding P@100, P@200, P@300 and the mean of three settings for each model are shown in Table 2. From the table, we have similar observations to the PR Curves: (1) BiGRU+2ATT outperforms CNN+ATT, PCNN+ATT and BiGRU+ATT in most cases in terms of all P@N scores; and (2) MLSSA-1 and MLSSA-2 significantly outperform all baselines for all measures. We observe that MLSSA-1 performs better than MLSSA-2 on tasks **One** and **Two**, but worse on **All**. We infer that in our 2-D sentence-level attention model, we

---

[6] P@N considers only the topmost results returned by the model.

[7] Regarding the metric, we keep the evaluation consistent with the work in Batista et al. (2013) where they used F1 to measure their RE systems on the Portuguese dataset, in order to maintain a fair comparison with their work using the same metric.

Test Settings	One				Two				All			
P@N(%)	100	200	300	Mean	100	200	300	Mean	100	200	300	Mean
CNN+ATT	72.0	67.0	59.5	66.2	75.5	69.0	63.3	69.3	74.3	71.5	64.5	70.1
PCNN+ATT	73.3	69.2	60.8	67.8	77.2	71.6	66.1	71.6	76.2	73.1	67.4	72.2
BiGRU+ATT	75.0	69.5	64.7	69.7	80.0	72.5	69.3	73.9	82.0	76.5	71.3	76.6
BiGRU+2ATT	81.0	74.0	67.3	74.1	81.0	75.5	70.7	75.7	81.0	76.0	72.7	76.6
MLSSA-1	**88.0**	**77.0**	**70.0**	**78.3**	88.0	**79.0**	**73.3**	**80.1**	87.0	**81.5**	76.0	81.5
MLSSA-2	87.0	76.0	**70.0**	77.7	**89.0**	78.5	72.3	79.9	**90.0**	**81.5**	**77.0**	**82.8**

Table 2: Precision values for the top-100, top-200, and top-300 relation instances that are randomly selected in terms of one, two and all sentences.

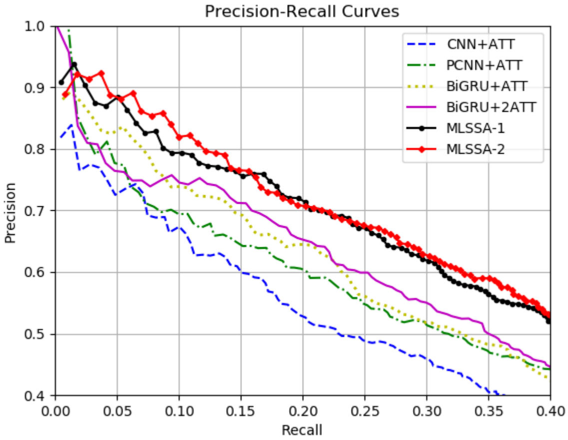

Figure 2: Comparison results of a variety of methods in terms of precision/recall curves.

set r^{L2} to 9, but there are only *one* and *two* instances for selection in tasks **One** and **Two**, so the 2-D matrix cannot demonstrate its full potential. However, in **All**, many entity pairs contain multiple or more than 9 instances, so it can learn a better 2-D matrix to focus on different instances.

4.8 Results on PT Dataset

Based on results from the NYT dataset, we choose PCNN+ATT and BiGRU+2ATT as representative baselines to compare against our MLSSA-1/2 systems on the PT test sets. The results in terms of Macro F1 are shown in Table 3.

It can be seen that on both test sets, our MLSSA-2 model achieved the best performance which shows that the structured 2-D word-level and sentence-level self-attention models can be well applied to datasets of a smaller scale and with a smaller ratio of multiple instances.

4.9 Examples and Analysis

In order to show the effectiveness of structured self-attention mechanisms, we show some examples by visualising the attentions on different aspects of a sentence, and on different sentences comparing with BiLSTM+2ATT model.

SYS	PT-MANUAL (%)	PT-SPLIT (%)
PCNN+ATT	62.3	74.1
BiGRU+2ATT	63.5	75.3
MLSSA-1	66.0	77.2
MLSSA-2	**69.6**	**78.1**

Table 3: Results on the PT test sets

Figure 3 shows the comparison of word-level attention mechanism between BiGRU+2ATT and MLSSA-1 reflecting their capability of context representation learning (*Problem I*). MLSSA-2 has a similar probability distribution to MLSSA-1 in terms of this example.

The *pink* fonts indicate lower probability and *red* indicates higher probability. We observe that: (1) BiGRU+2ATT mainly focuses on one word *baltimore*. We can see that it has little attention on the entity word *maryland*. In this example, the comma implies a semantic relationship *location/location/contains* for the entity pair (*Maryland, Baltimore*). However, BiGRU+2ATT allocates quite a small probability to it; and (2) we can see that our model focuses on different words via different attention vectors (9 in total). Words with a *red* background have a high probability of 0.98 or so. For rows 5, 6, 8 and 9, the focus is on the *BLANK* tokens. In both systems, the maximum time step is set to 70, which indicates that shorter sentences are padded with *BLANK* tokens and longer sentences are cut off. The last row shows the summation of **9** annotation vectors, and it constructs a dependency-like context of the relation for the entity pair. Attentions on different words are attributed to the penalisation P_{L1} which is optimised to learn orthogonal eigenvectors.

Figure 4 shows the comparison of sentence-level attentions between BiGRU+2ATT, MLSSA-1 and MLSSA-2. The first, second and third columns are probability distributions over multi-

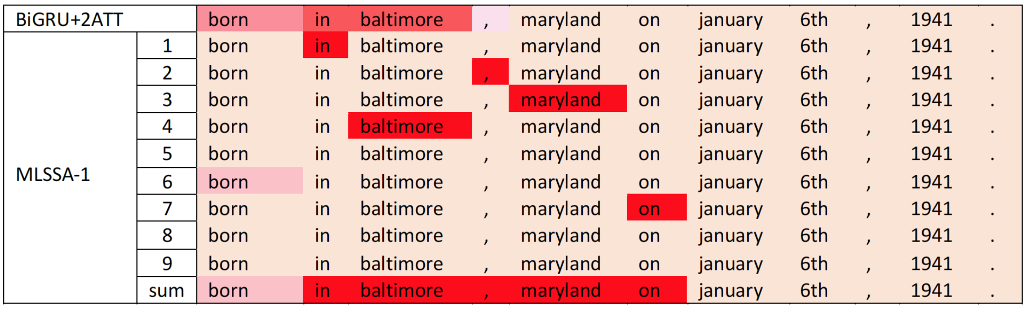

BiGRU+2ATT		born	in	baltimore	,	maryland	on	january	6th	,	1941	.
MLSSA-1	1	born	in	baltimore	,	maryland	on	january	6th	,	1941	.
	2	born	in	baltimore	,	maryland	on	january	6th	,	1941	.
	3	born	in	baltimore	,	maryland	on	january	6th	,	1941	.
	4	born	in	baltimore	,	maryland	on	january	6th	,	1941	.
	5	born	in	baltimore	,	maryland	on	january	6th	,	1941	.
	6	born	in	baltimore	,	maryland	on	january	6th	,	1941	.
	7	born	in	baltimore	,	maryland	on	january	6th	,	1941	.
	8	born	in	baltimore	,	maryland	on	january	6th	,	1941	.
	9	born	in	baltimore	,	maryland	on	january	6th	,	1941	.
	sum	born	in	baltimore	,	maryland	on	january	6th	,	1941	.

Figure 3: Comparison of word-level attentions.

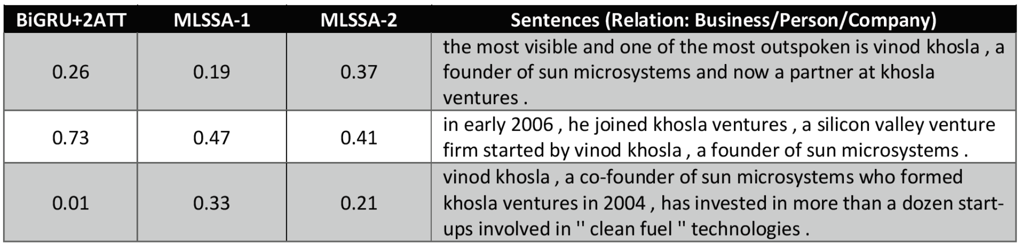

BiGRU+2ATT	MLSSA-1	MLSSA-2	Sentences (Relation: Business/Person/Company)
0.26	0.19	0.37	the most visible and one of the most outspoken is vinod khosla , a founder of sun microsystems and now a partner at khosla ventures .
0.73	0.47	0.41	in early 2006 , he joined khosla ventures , a silicon valley venture firm started by vinod khosla , a founder of sun microsystems .
0.01	0.33	0.21	vinod khosla , a co-founder of sun microsystems who formed khosla ventures in 2004 , has invested in more than a dozen start-ups involved in " clean fuel " technologies .

Figure 4: Comparison of sentence-level attentions.

ple instances. The entity pair is (*vinod khosla, sun microsystems*), and their relation is *Business/Person/Company*. From this figure, we observe that: (1) BiGRU+2ATT allocates high probabilities to **Sentences 1** and **2** by learning the context of "*a founder of*", but does not recognise that "*co-founder*" is semantically the same as "*founder*"; and (2) our two models almost evenly focus on all sentences because they express the same semantic concept of "*a person is a founder of a company*" in terms of the given entity pair. Therefore, the structured self-attention mechanism is helpful to learn a better representation and select informative sentences.

5 Conclusion and Future Work

This paper has proposed a multi-level structured self-attention mechanism for distantly supervised RE. In this framework, the traditional 1-D word-level and sentence-level attentions are extended to 2-D structured matrices which can learn different aspects of a sentence, and different informative instances. Experimental results on two distant supervision data sets show that (1) the structured 2-D word-level attention can learn a better sentence representation; (2) the structured 2-D sentence-level attention and averaged selection can perform better selection from multiple instances for relation classification; (3) the proposed framework significantly outperforms state-of-the-art baseline systems for a range of different measures, which verifies its effectiveness on two representation learning issues. A subsequent manual investigation via examples also show its effectiveness on two representation learning issues.

In future work, we will build a domain-specific distant supervision dataset with a higher ratio of multiple instances and compare our system with others. Furthermore, we will consider not using RNNs or CNNs, but a deeper neural networks with only attentions for distantly supervised RE, similar to the work in Vaswani et al. (2017).

Acknowledgments

We thank Yuyun Huang and Utsab Barman from University College Dublin, and Jer Hayes, Edward Burgin and other colleagues from Accenture Labs Dublin for helpful comments, discussion and facilities. We would like to thank the reviewers for their valuable and constructive comments and suggestions. This research is supported by the ADAPT Centre for Digital Content Technology, funded under the SFI Research Centres Programme (Grant 13/RC/2106), and by SFI Industry Fellowship Programme 2016 (Grant 16/IFB/4490), and is supported by Accenture Labs Dublin.

References

David Soares Batista, David Forte, Rui Silva, Bruno Martins, and Mario J. Silva. 2013. Exploring dbpedia and wikipedia for portuguese semantic relationship extraction. *Linguamatica*, 5(1).

Razvan Bunescu and Raymond J Mooney. 2005. Subsequence kernels for relation extraction. In *In Proceedings of NIPS*, pages 171–178.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th ACL-HLT*, pages 541–550.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the Thirty-First (AAAI) Conference on Artificial Intelligence*, pages 3060–3066.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. *CoRR*, abs/1702.00887.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017a. Neural relation extraction with multi-lingual attention. In *Proceedings of the 55th ACL*, pages 34–43.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th ACL.*

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017b. A structured self-attentive sentence embedding. In *International Conference on Learning Representations 2017.*

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th ACL and the 4th AFNLP*, pages 1003–1011.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*, pages 148–163.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, Jeju Island, Korea.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 6000–6010.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th ACL*, pages 1298–1307.

Linyi Yang, Tin Lok, James Ng, Catherine Mooney, and Ruihai Dong. 2017. Multi-level attention-based neural networks for distant supervised relation extraction. In *Proceedings of Irish Conference on Artificial Intelligence and Cognitive Science (AICS).*

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3:1083–1106.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014*, pages 2335–2344.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Incorporating relation paths in neural relation extraction. In *Proceedings of EMNLP*, pages 1769–1778.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *In Proceedings of the 43rd ACL*, pages 427–434.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th ACL*, pages 207–212.

N-ary Relation Extraction using Graph State LSTM

Linfeng Song[1]*, **Yue Zhang**[3]*, **Zhiguo Wang**[2] and **Daniel Gildea**[1]
[1]Department of Computer Science, University of Rochester, Rochester, NY 14627
[2]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
[3]School of Engineering, Westlake University, China

Abstract

Cross-sentence n-ary relation extraction detects relations among n entities across multiple sentences. Typical methods formulate an input as a *document graph*, integrating various intra-sentential and inter-sentential dependencies. The current state-of-the-art method splits the input graph into two DAGs, adopting a DAG-structured LSTM for each. Though being able to model rich linguistic knowledge by leveraging graph edges, important information can be lost in the splitting procedure. We propose a graph-state LSTM model, which uses a parallel state to model each word, recurrently enriching state values via message passing. Compared with DAG LSTMs, our graph LSTM keeps the original graph structure, and speeds up computation by allowing more parallelization. On a standard benchmark, our model shows the best result in the literature.

1 Introduction

As a central task in natural language processing, relation extraction has been investigated on news, web text and biomedical domains. It has been shown to be useful for detecting explicit facts, such as cause-effect (Hendrickx et al., 2009), and predicting the effectiveness of a medicine on a cancer caused by mutation of a certain gene in the biomedical domain (Quirk and Poon, 2017; Peng et al., 2017). While most existing work extracts relations within a sentence (Zelenko et al., 2003; Palmer et al., 2005; Zhao and Grishman, 2005; Jiang and Zhai, 2007; Plank and Moschitti, 2013; Li and Ji, 2014; Gormley et al., 2015; Miwa and Bansal, 2016; Zhang et al., 2017), the task of cross-sentence relation extraction has received increasing attention (Gerber and Chai, 2010; Yoshikawa et al., 2011). Recently, Peng et al. (2017) extend cross-sentence relation extraction by further detecting relations among several entity mentions (n-ary relation). Table 1 shows an example, which conveys the fact that cancers caused by the *858E* mutation on *EGFR* gene can respond to the *gefitinib* medicine. The three entity mentions form a ternary relation yet appear in distinct sentences.

The deletion mutation on exon-19 of **EGFR** gene was present in 16 patients, while the **858E** point mutation on exon-21 was noted in 10. All patients were treated with **gefitinib** and showed a partial response.

Table 1: An example showing that tumors with *L858E* mutation in *EGFR* gene respond to *gefitinib* treatment.

Peng et al. (2017) proposed a graph-structured LSTM for n-ary relation extraction. As shown in Figure 1 (a), graphs are constructed from input sentences with dependency edges, links between adjacent words, and inter-sentence relations, so that syntactic and discourse information can be used for relation extraction. To calculate a hidden state encoding for each word, Peng et al. (2017) first split the input graph into two directed acyclic graphs (DAGs) by separating left-to-right edges from right-to-left edges (Figure 1 (b)). Then, two separate gated recurrent neural networks, which extend tree LSTM (Tai et al., 2015), were adopted for each single-directional DAG, respectively. Finally, for each word, the hidden states of both directions are concatenated as the final state. The bi-directional DAG LSTM model showed superior performance over several strong baselines, such as tree-structured LSTM (Miwa and Bansal, 2016), on a biomedical-domain benchmark.

However, the bidirectional DAG LSTM model suffers from several limitations. First, important information can be lost when converting a graph

---

*Equal contribution

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2226–2235
Brussels, Belgium, October 31 - November 4, 2018.

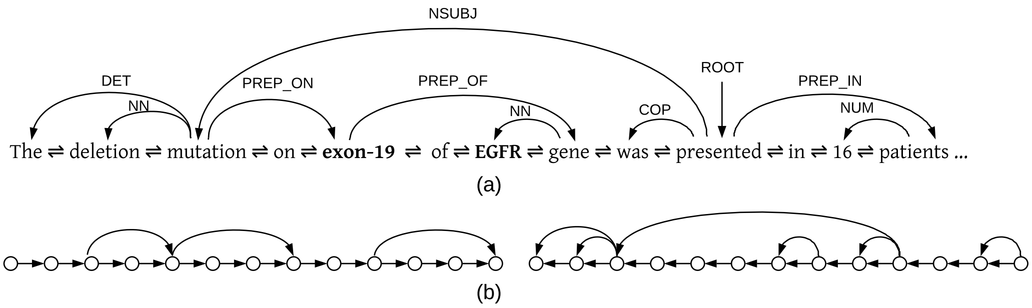

Figure 1: (a) A fraction of the dependency graph of the example in Table 1. For simplicity, we omit edges of discourse relations. (b) Results after splitting the graph into two DAGs.

into two separate DAGs. For the example in Figure 1, the conversion breaks the inner structure of "exon-19 of EGFR gene", where the relation between "exon-19" and "EGFR" via the dependency path "exon-19 $\xrightarrow{\text{PREP\_OF}}$ gene $\xrightarrow{\text{NN}}$ EGFR" is lost from the original subgraph. Second, using LSTMs on both DAGs, information of only ancestors and descendants can be incorporated for each word. Sibling information, which may also be important, is not included.

A potential solution to the problems above is to model a graph as a whole, learning its representation without breaking it into two DAGs. Due to the existence of cycles, naive extension of tree LSTMs cannot serve this goal. Recently, graph convolutional networks (GCN) (Kipf and Welling, 2017; Marcheggiani and Titov, 2017; Bastings et al., 2017) and graph recurrent networks (GRN) (Song et al., 2018; Zhang et al., 2018) have been proposed for representing graph structures for NLP tasks. Such methods encode a given graph by hierarchically learning representations of neighboring nodes in the graphs via their connecting edges. While GCNs use CNN for information exchange, GRNs take gated recurrent steps to this end. For fair comparison with DAG LSTMs, we build a graph LSTM by extending Song et al. (2018), which strictly follow the configurations of Peng et al. (2017) such as the source of features and hyper parameter settings. In particular, the full input graph is modeled as a single state, with words in the graph being its sub states. State transitions are performed on the graph recurrently, allowing word-level states to exchange information through dependency and discourse edges. At each recurrent step, each word advances its current state by receiving information from the current states of its adjacent words. Thus with increasing numbers of recurrent steps each word receives information from a larger context. Figure 2 shows the recurrent transition steps where each node works simultaneously within each transition step.

Compared with bidirectional DAG LSTM, our method has several advantages. First, it keeps the original graph structure, and therefore no information is lost. Second, sibling information can be easily incorporated by passing information up and then down from a parent. Third, information exchange allows more parallelization, and thus can be very efficient in computation.

Results show that our model outperforms a bidirectional DAG LSTM baseline by 5.9% in accuracy, overtaking the state-of-the-art system of Peng et al. (2017) by 1.2%. Our code is available at `https://github.com/freesunshine0316/nary-grn`.

Our contributions are summarized as follows.

- We empirically compared graph LSTM with DAG LSTM for n-ary relation extraction tasks, showing that the former is better by more effective use of structural information;
- To our knowledge, we are the first to investigate a graph recurrent network for modeling dependency and discourse relations.

2 Task Definition

Formally, the input for cross-sentence n-ary relation extraction can be represented as a pair $(\mathcal{E}, \mathcal{T})$, where $\mathcal{E} = (\epsilon_1, \ldots, \epsilon_N)$ is the set of entity mentions, and $\mathcal{T} = [S_1; \ldots; S_M]$ is a text consisting of multiple sentences. Each entity mention ϵ_i belongs to one sentence in $\mathcal{T}$. There is a predefined relation set $\mathcal{R} = (r_1, \ldots, r_L, None)$, where *None* represents that no relation holds for the entities.

This task can be formulated as a binary classification problem of determining whether $\epsilon_1, \ldots, \epsilon_N$ together form a relation (Peng et al., 2017), or a multi-class classification problem of detecting which relation holds for the entity mentions. Take Table 1 as an example. The binary classification task is to determine whether *gefitinib* would have an effect on this type of cancer, given a cancer patient with *858E* mutation on gene *EGFR*. The multi-class classification task is to detect the exact drug effect: response, resistance, sensitivity, etc.

3 Baseline: Bi-directional DAG LSTM

Peng et al. (2017) formulate the task as a graph-structured problem in order to adopt rich dependency and discourse features. In particular, Stanford parser (Manning et al., 2014) is used to assign syntactic structure to input sentences, and heads of two consecutive sentences are connected to represent discourse information, resulting in a graph structure. For each input graph $G = (V, E)$, the nodes V are words within input sentences, and each edge $e \in E$ connects two words that either have a relation or are adjacent to each other. Each edge is denoted as a triple (i, j, l), where i and j are the indices of the source and target words, respectively, and the edge label l indicates either a dependency or discourse relation (such as "nsubj") or a relative position (such as "next_tok" or "prev_tok"). Throughout this paper, we use $E_{in}(j)$ and $E_{out}(j)$ to denote the sets of incoming and outgoing edges for word j.

For a bi-directional DAG LSTM baseline, we follow Peng et al. (2017), splitting each input graph into two separate DAGs by separating left-to-right edges from right-to-left edges (Figure 1). Each DAG is encoded by using a DAG LSTM (Section 3.2), which takes both source words and edge labels as inputs (Section 3.1). Finally, the hidden states of entity mentions from both LSTMs are taken as inputs to a logistic regression classifier to make a prediction:

$$\hat{y} = \text{softmax}(W_0[h_{\epsilon_1}; \ldots; h_{\epsilon_N}] + b_0), \quad (1)$$

where h_{ϵ_j} is the hidden state of entity ϵ_j. W_0 and b_0 are parameters.

3.1 Input Representation

Both nodes and edge labels are useful for modeling a syntactic graph. As the input to our DAG LSTM, we first calculate the representation for each edge (i, j, l) by:

$$x_{i,j}^{l} = W_1\Big([e_l; e_i]\Big) + b_1, \quad (2)$$

where W_1 and b_1 are model parameters, e_i is the embedding of the source word indexed by i, and e_l is the embedding of the edge label l.

3.2 State transition

The baseline LSTM model learns DAG representations sequentially, following word orders. Taking the edge representations (such as $x_{i,j}^{l}$) as input, gated state transition operations are executed on both the forward and backward DAGs. For each word j, the representations of its incoming edges $E_{in}(j)$ are summed up as one vector:

$$x_j^{in} = \sum_{(i,j,l) \in E_{in}(j)} x_{i,j}^{l} \quad (3)$$

Similarly, for each word j, the states of all incoming nodes are summed to a single vector before being passed to the gated operations:

$$h_j^{in} = \sum_{(i,j,l) \in E_{in}(j)} h_i \quad (4)$$

Finally, the gated state transition operation for the hidden state h_j of the j-th word can be defined as:

$$\begin{aligned}
i_j &= \sigma(W_i x_j^{in} + U_i h_j^{in} + b_i) \\
o_j &= \sigma(W_o x_j^{in} + U_o h_j^{in} + b_o) \\
f_{i,j} &= \sigma(W_f x_{i,j}^{l} + U_f h_i + b_f) \\
u_j &= \sigma(W_u x_j^{in} + U_u h_j^{in} + b_u) \\
c_j &= i_j \odot u_j + \sum_{(i,j,l) \in E_{in}(j)} f_{i,j} \odot c_i \\
h_j &= o_j \odot \tanh(c_j),
\end{aligned} \quad (5)$$

where i_j, o_j and $f_{i,j}$ are a set of input, output and forget gates, respectively, and W_x, U_x and b_x ($x \in \{i, o, f, u\}$) are model parameters.

3.3 Comparison with Peng et al. (2017)

Our baseline is computationally similar to Peng et al. (2017), but different on how to utilize edge labels in the gated network. In particular, Peng et al. (2017) make model parameters specific to edge labels. They consider two model variations, namely *Full Parametrization (FULL)* and *Edge-Type Embedding (EMBED)*. *FULL* assigns distinct

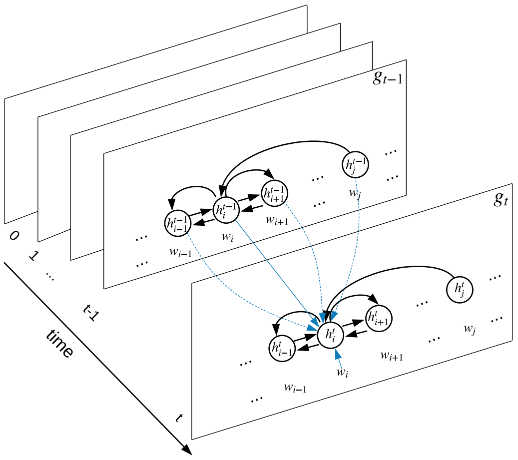

Figure 2: Graph state transitions via message passing, where each w_i is a word.

Us (in Equation 5) to different edge types, so that each edge label is associated with a 2D weight matrix to be tuned in training. On the other hand, *EMBED* assigns each edge label to an embedding vector, but complicates the gated operations by changing the Us to be 3D tensors.[1]

In contrast, we take edge labels as part of the input to the gated network. In general, the edge labels are first represented as embeddings, before being concatenated with the node representation vectors (Equation 2). We choose this setting for both the baseline and our graph state LSTM model in Section 4, since it requires fewer parameters compared with *FULL* and *EMBED*, thus being less exposed to overfitting on small-scaled data.

4 Graph State LSTM

Our input graph formulation strictly follows Section 3. In particular, our model adopts the same methods for calculating input representation (as in Section 3.1) and performing classification as the baseline model. However, different from the baseline bidirectional DAG LSTM model, we leverage a graph-structured LSTM to directly model the input graph, without splitting it into two DAGs.

Figure 2 shows an overview of our model. Formally, given an input graph $G = (V, E)$, we define a state vector h^j for each word $v_j \in V$. The state of the graph consists of all word states, and thus can be represented as:

$$g = \{h^j\}|_{v_j \in V} \tag{6}$$

In order to capture non-local information, our model performs information exchange between words through a recurrent state transition process, resulting in a sequence of graph states $g_0, g_1, \ldots, g_t$, where $g_t = \{h_t^j\}|_{v_j \in V}$. The initial graph state g_0 consists of a set of initial word states $h_0^j = h_0$, where h_0 is a zero vector.

4.1 State transition

Following the approches of Song et al. (2018) and Zhang et al. (2018), a recurrent neural network is utilized to model the state transition process. In particular, the transition from g_{t-1} to g_t consists of hidden state transition for each word, as shown in Figure 2. At each step t, we allow information exchange between a word and all words that are directly connected to the word. To avoid gradient diminishing or bursting, gated LSTM cells are adopted, where a cell c_t^j is taken to record memory for h_t^j. We use an input gate i_t^j, an output gate o_t^j and a forget gate f_t^j to control information flow from the inputs and to h_t^j.

The inputs to a word v_j, include representations of edges that are connected to v_j, where v_j can be either the source or the target of the edge. Similar to Section 3.1, we define each edge as a triple (i, j, l), where i and j are indices of the source and target words, respectively, and l is the edge label. $x_{i,j}^l$ is the representation of edge (i, j, l). The inputs for v_j are distinguished by incoming and outgoing directions, where:

$$\begin{aligned} x_j^i &= \sum_{(i,j,l) \in E_{in}(j)} x_{i,j}^l \\ x_j^o &= \sum_{(j,k,l) \in E_{out}(j)} x_{j,k}^l \end{aligned} \tag{7}$$

Here $E_{in}(j)$ and $E_{out}(j)$ denote the sets of incoming and outgoing edges of v_j, respectively.

In addition to edge inputs, a cell also takes the hidden states of its incoming and outgoing words during a state transition. In particular, the states of all incoming words and outgoing words are summed up, respectively:

$$\begin{aligned} h_j^i &= \sum_{(i,j,l) \in E_{in}(j)} h_{t-1}^i \\ h_j^o &= \sum_{(j,k,l) \in E_{out}(j)} h_{t-1}^k, \end{aligned} \tag{8}$$

Based on the above definitions of x_j^i, x_j^o, h_j^i and h_j^o, the recurrent state transition from g_{t-1} to g_t,

---

[1] For more information please refer Section 3.3 of Peng et al. (2017).

as represented by h_t^j, is defined as:

$$\begin{aligned}
i_t^j &= \sigma(W_i x_j^i + \hat{W}_i x_j^o + U_i h_j^i + \hat{U}_i h_j^o + b_i)\\
o_t^j &= \sigma(W_o x_j^i + \hat{W}_o x_j^o + U_o h_j^i + \hat{U}_o h_j^o + b_o)\\
f_t^j &= \sigma(W_f x_j^i + \hat{W}_f x_j^o + U_f h_j^i + \hat{U}_f h_j^o + b_f)\\
u_t^j &= \sigma(W_u x_j^i + \hat{W}_u x_j^o + U_u h_j^i + \hat{U}_u h_j^o + b_u)\\
c_t^j &= f_t^j \odot c_{t-1}^j + i_t^j \odot u_t^j\\
h_t^j &= o_t^j \odot \tanh(c_t^j),
\end{aligned}$$

where i_t^j, o_t^j and f_t^j are the input, output and forget gates, respectively. W_x, $\hat{W}_x$, U_x, $\hat{U}_x$, b_x ($x \in \{i, o, f, u\}$) are model parameters.

Graph State LSTM vs bidirectional DAG LSTM A contrast between the baseline DAG LSTM and our graph LSTM can be made from the perspective of information flow. For the baseline, information flow follows the natural word order in the input sentence, with the two DAG components propagating information from left to right and from right to left, respectively. In contrast, information flow in our graph state LSTM is relatively more concentrated at individual words, with each word exchanging information with all its graph neighbors simultaneously at each sate transition. As a result, wholistic contextual information can be leveraged for extracting features for each word, as compared to separated handling of bi-directional information flow in DAG LSTM. In addition, arbitrary structures, including arbitrary cyclic graphs, can be handled.

From an initial state with isolated words, information of each word propagates to its graph neighbors after each step. Information exchange between non-neighboring words can be achieved through multiple transition steps. We experiment with different transition step numbers to study the effectiveness of global encoding. Unlike the baseline DAG LSTM encoder, our model allows parallelization in node-state updates, and thus can be highly efficient using a GPU.

5 Training

We train our models with a cross-entropy loss over a set of gold standard data:

$$l = -\log p(y_i | X_i; \theta), \tag{9}$$

where X_i is an input graph, y_i is the gold class label of X_i, and θ is the model parameters. Adam (Kingma and Ba, 2014) with a learning rate of 0.001 is used as the optimizer, and the model that yields the best devset performance is selected to evaluate on the test set. Dropout with rate 0.3 is used during training. Both training and evaluation are conducted using a Tesla K20X GPU.

Data	Avg. Tok.	Avg. Sent.	Cross
TERNARY	73.9	2.0	70.1%
BINARY	61.0	1.8	55.2%

Table 2: Dataset statistics. *Avg. Tok.* and *Avg. Sent.* are the average number of tokens and sentences, respectively. *Cross* is the percentage of instances that contain multiple sentences.

6 Experiments

We conduct experiments for the binary relation detection task and the multi-class relation extraction task discussed in Section 2.

6.1 Data

We use the dataset of Peng et al. (2017), which is a biomedical-domain dataset focusing on drug-gene-mutation ternary relations,[2] extracted from PubMed. It contains 6987 ternary instances about drug-gene-mutation relations, and 6087 binary instances about drug-mutation sub-relations. Table 2 shows statistics of the dataset. Most instances of ternary data contain multiple sentences, and the average number of sentences is around 2. There are five classification labels: “resistance or non-response”, “sensitivity”, “response”, “resistance” and “None”. We follow Peng et al. (2017) and binarize multi-class labels by grouping all relation classes as “Yes” and treat “None” as “No”.

6.2 Settings

Following Peng et al. (2017), five-fold cross-validation is used for evaluating the models,[3] and the final test accuracy is calculated by averaging the test accuracies over all five folds. For each fold, we randomly separate 200 instances from the training set for development. The batch size is set as 8 for all experiments. Word embeddings are initialized with the 100-dimensional GloVe (Pennington et al., 2014) vectors, pretrained on 6 billion words from Wikipedia and web text. The edge label embeddings are 3-dimensional and randomly

---

[2] The dataset is available at `http://hanover.azurewebsites.net`.

[3] The released data has been separated into 5 portions, and we follow the exact split.

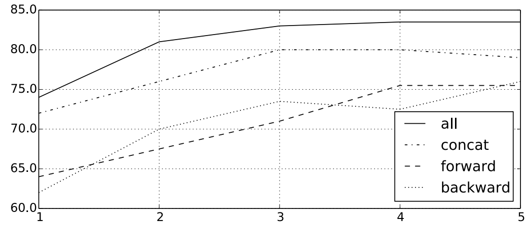

Figure 3: Dev accuracies against transition steps for the graph state LSTM model.

initialized. Pretrained word embeddings are not updated during training. The dimension of hidden vectors in LSTM units is set to 150.

6.3 Development Experiments

We first analyze our model on the drug-gene-mutation ternary relation dataset, taking the first among 5-fold cross validation settings for our data setting. Figure 3 shows the devset accuracies of different state transition numbers, where *forward* and *backward* execute our graph state model only on the forward or backward DAG, respectively. *Concat* concatenates the hidden states of *forward* and *backward*. *All* executes our graph state model on original graphs.

The performance of *forward* and *backward* lag behind *concat*, which is consistent with the intuition that both forward and backward relations are useful (Peng et al., 2017). In addition, *all* gives better accuracies compared with *concat*, demonstrating the advantage of simultaneously considering forward and backward relations during representation learning. For all the models, more state transition steps result in better accuracies, where larger contexts can be integrated in the representations of graphs. The performance of *all* starts to converge after 4 and 5 state transitions, so we set the number of state transitions to 5 in the remaining experiments.

6.4 Final results

Table 3 compares our model with the bidirectional DAG baseline and the state-of-the-art results on this dataset, where *EMBED* and *FULL* have been briefly introduced in Section 3.3. *+multi-task* applies joint training of both ternary (drug-gene-mutation) relations and their binary (drug-mutation) sub-relations. Quirk and Poon (2017) use a statistical method with a logistic regression classifier and features derived from shortest paths between all entity pairs. *Bidir DAG LSTM* is our bidirectional DAG LSTM baseline, and *GS GLSTM* is our graph state LSTM model.

Model	Single	Cross
Quirk and Poon (2017)	74.7	77.7
Peng et al. (2017) - EMBED	76.5	80.6
Peng et al. (2017) - FULL	77.9	80.7
+ multi-task	–	82.0
Bidir DAG LSTM	75.6	77.3
GS GLSTM	**80.3***	**83.2***

Table 3: Average test accuracies for TERNARY drug-gene-mutation interactions. *Single* represents experiments only on instances within single sentences, while *Cross* represents experiments on all instances. *: significant at $p < 0.01$

Using all instances (the *Cross* column in Table 3), our graph state LSTM model shows the highest test accuracy among all methods, which is 5.9% higher than our baseline.[4] The accuracy of our baseline is lower than *EMBED* and *FULL* of Peng et al. (2017), which is likely due to the differences mentioned in Section 3.3. Our final results are better than Peng et al. (2017), despite the fact that we do not use multi-task learning.

We also report accuracies only on instances within single sentences (column *Single* in Table 3), which exhibit similar contrasts. Note that all systems show performance drops when evaluated only on single-sentence relations, which are actually more challenging. One reason may be that some single sentences cannot provide sufficient context for disambiguation, making it necessary to study cross-sentence context. Another reason may be overfitting caused by relatively fewer training instances in this setting, as only 30% instances are within a single sentence. One interesting observation is that our baseline shows the least performance drop of 1.7 points, in contrast to up to 4.1 for other neural systems. This can be a supporting evidence for overfitting, as our baseline has fewer parameters at least than *FULL* and *EMBED*.

6.5 Analysis

Efficiency. Table 4 shows the training and decoding time of both the baseline and our model. Our model is 8 to 10 times faster than the baseline in training and decoding speeds, respectively. By revisiting Table 2, we can see that the average number of tokens for the ternary-relation data is

---

[4] $p < 0.01$ using t-test. For the remaining of this paper, we use the same measure for statistical significance.

Model	Train	Decode
Bidir DAG LSTM	281s	27.3s
GS GLSTM	36.7s	2.7s

Table 4: The average times for training one epoch and decoding (seconds) over five folds on drug-gene-mutation TERNARY cross sentence setting.

74, which means that the baseline model has to execute 74 recurrent transition steps for calculating a hidden state for each input word. On the other hand, our model only performs 5 state transitions, and calculations between each pair of nodes for one transition are parallelizable. This accounts for the better efficiency of our model.

Accuracy against sentence length Figure 5 (a) shows the test accuracies on different sentence lengths. We can see that *GS GLSTM* and *Bidir DAG LSTM* show performance increase along increasing input sentence lengths. This is likely because longer contexts provide richer information for relation disambiguation. *GS GLSTM* is consistently better than *Bidir DAG LSTM*, and the gap is larger on shorter instances. This demonstrates that *GS GLSTM* is more effective in utilizing a smaller context for disambiguation.

Accuracy against the maximal number of neighbors Figure 5 (b) shows the test accuracies against the maximum number of neighbors. Intuitively, it is easier to model graphs containing nodes with more neighbors, because these nodes can serve as a "supernode" that allow more efficient information exchange. The performances of both *GS GLSTM* and *Bidir DAG LSTM* increase with increasing maximal number of neighbors, which coincide with this intuition. In addition, *GS GLSTM* shows more advantage than *Bidir DAG LSTM* under the inputs having lower maximal number of neighbors, which further demonstrates the superiority of *GS GLSTM* over *Bidir DAG LSTM* in utilizing context information.

Case study Figure 4 visualizes the merits of *GS GLSTM* over *Bidir DAG LSTM* using two examples. *GS GLSTM* makes the correct predictions for both cases, while *Bidir DAG LSTM* fails to.

The first case generally mentions that *Gefitinib* does not have an effect on *T790M* mutation on *EGFR* gene. Note that both "However" and "was not" serve as indicators; thus incorporating them into the contextual vectors of these entity mentions is important for making a correct prediction. However, both indicators are leaves of the dependency tree, making it impossible for *Bidir DAG LSTM* to incorporate them into the contextual vectors of entity mentions up the tree through dependency edges.[5] On the other hand, it is easier for *GS GLSTM*. For instance, "was not" can be incorporated into "Gefitinib" through "suppressed $\xrightarrow{\text{agent}}$ treatment $\xrightarrow{\text{nn}}$ Gefitinib".

Model	Single	Cross
Quirk and Poon (2017)	73.9	75.2
Miwa and Bansal (2016)	75.9	75.9
Peng et al. (2017) - EMBED	74.3	76.5
Peng et al. (2017) - FULL	75.6	76.7
+ multi-task	–	78.5
Bidir DAG LSTM	76.9	76.4
GS GLSTM	**83.5***	**83.6***

Table 5: Average test accuracies in five-fold cross-validation for BINARY drug-mutation interactions.

The second case is to detect the relation among "cetuximab" (drug), "EGFR" (gene) and "S492R" (mutation), which does not exist. However, the context introduces further ambiguity by mentioning another drug "Panitumumab", which does have a relation with "EGFR" and "S492R". Being sibling nodes in the dependency tree, "can not" is an indicator for the relation of "cetuximab". *GS GLSTM* is correct, because "can not" can be easily included into the contextual vector of "cetuximab" in two steps via "bind $\xrightarrow{\text{nsubj}}$cetuximab".

6.6 Results on Binary Sub-relations

Following previous work, we also evaluate our model on drug-mutation binary relations. Table 5 shows the results, where Miwa and Bansal (2016) is a state-of-the-art model using sequential and tree-structured LSTMs to jointly capture linear and dependency contexts for relation extraction. Other models have been introduced in Section 6.4.

Similar to the ternary relation extraction experiments, *GS GLSTM* outperforms all the other systems with a large margin, which shows that the message passing graph LSTM is better at encoding rich linguistic knowledge within the input graphs. Binary relations being easier, both *GS GLSTM* and *Bidir DAG LSTM* show increased or similar performances compared with the ternary relation ex-

---

[5] As shown in Figure 1, a directional DAG LSTM propagates information according to the edge directions.

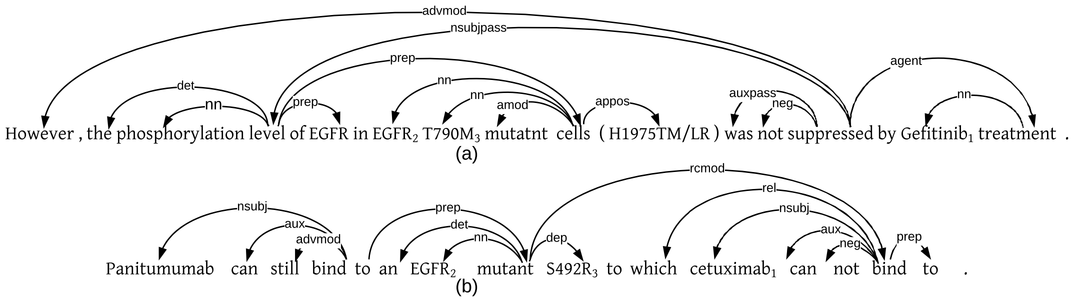

Figure 4: Example cases. Words with subindices 1, 2 and 3 represent drugs, genes and mutations, respectively. References for both cases are “No”. For both cases, *GS GLSTM* makes the correct predictions, while *Bidir DAG LSTM* does incorrectly.

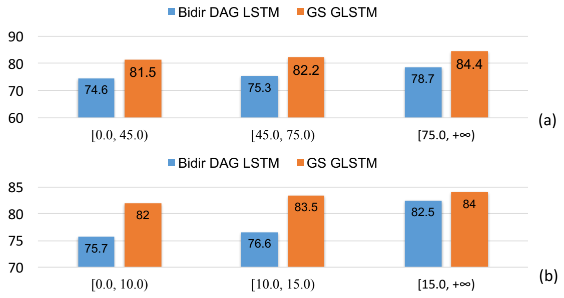

Figure 5: Test set performances on (a) different sentence lengths, and (b) different maximal number of neighbors.

Model	TERNARY	BINARY
Bidir DAG LSTM	51.7	50.7
GS GLSTM	**71.1***	**71.7***

Table 6: Average test accuracies for multi-class relation extraction with all instances (“Cross”).

periments. On this set, our bidirectional DAG LSTM model is comparable to *FULL* using all instances (“Cross”) and slightly better than *FULL* using only single-sentence instances (“Single”).

6.7 Fine-grained Classification

Our dataset contains five classes as mentioned in Section 6.1. However, previous work only investigates binary relation detection. Here we also study the multi-class classification task, which can be more informative for applications.

Table 6 shows accuracies on multi-class relation extraction, which makes the task more ambiguous compared with binary relation extraction. The results show similar comparisons with the binary relation extraction results. However, the performance gaps between *GS GLSTM* and *Bidir DAG LSTM* dramatically increase, showing the superiority of *GS GLSTM* over *Bidir DAG LSTM* in utilizing context information.

7 Related Work

N-ary relation extraction N-ary relation extractions can be traced back to MUC-7 (Chinchor, 1998), which focuses on entity-attribution relations. It has also been studied in biomedical domain (McDonald et al., 2005), but only the instances within a single sentence are considered. Previous work on cross-sentence relation extraction relies on either explicit co-reference annotation (Gerber and Chai, 2010; Yoshikawa et al., 2011), or the assumption that the whole document refers to a single coherent event (Wick et al., 2006; Swampillai and Stevenson, 2011). Both simplify the problem and reduce the need for learning better contextual representation of entity mentions. A notable exception is Quirk and Poon (2017), who adopt distant supervision and integrated contextual evidence of diverse types without relying on these assumptions. However, they only study binary relations. We follow Peng et al. (2017) by studying ternary cross-sentence relations.

Graph encoder Liang et al. (2016) build a graph LSTM model for semantic object parsing, which aims to segment objects within an image into more fine-grained, semantically meaningful parts. The nodes of an input graph come from image superpixels, and the edges are created by connecting spatially neighboring nodes. Their model is similar as Peng et al. (2017) by calculating node states sequentially: for each input graph, a start node and a node sequence are chosen, which determines the order of recurrent state updates. In contrast, our graph LSTM do not need ordering of graph nodes, and is highly parallelizable.

Graph convolutional networks (GCNs) and very recently graph recurrent networks (GRNs) have been used to model graph structures in NLP tasks, such as semantic role labeling (Marcheggiani and Titov, 2017), machine translation (Bastings et al., 2017), text generation (Song et al., 2018), text representation (Zhang et al., 2018) and semantic parsing (Xu et al., 2018b,a). In particular, Zhang et al. (2018) use GRN to represent raw sentences by building a graph structure of neighboring words and a sentence-level node, showing that the encoder outperforms BiLSTMs and Transformer (Vaswani et al., 2017) on classification and sequence labeling tasks; Song et al. (2018) build a GRN for encoding AMR graphs, showing that the representation is superior compared to BiLSTM on serialized AMR. Our work is in line with their work in the investigation of GRN on NLP. To our knowledge, we are the first to use GRN for representing dependency and discourse structures. Under the same recurrent framework, we show that modeling the original graphs with one GRN model is more useful than two DAG LSTMs for our relation extraction task. We choose GRN as our main method because it gives a more fair comparison with DAG LSTM. We leave it to future work to compare GCN and GRN for our task.

8 Conclusion

We explored a graph-state LSTM model for cross-sentence n-ary relation extraction, which uses a recurrent state transition process to incrementally refine a neural graph state representation capturing graph structure contexts. Compared with a bidirectional DAG LSTM baseline, our model has several advantages. First, it does not change the input graph structure, so that no information can be lost. For example, it can easily incorporate sibling information when calculating the contextual vector of a node. Second, it is better parallelizable. Experiments show significant improvements over the previously reported numbers, including that of the bidirectional graph LSTM model.

For future work, we consider adding coreference information as an entity mention can have coreferences, which help on information collection. Another possible direction is including word sense information. Confusing caused by word senses can be a severe problem. Not only content words, but also propositions can introduce word sense problem (Gong et al., 2018).

Acknowledge We thank the anonymized reviewers for their insightful comments, and the Center for Integrated Research Computing (CIRC) of University of Rochester for making special reservations for computation resources.

References

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-17)*.

Nancy A Chinchor. 1998. Overview of muc-7/met-2. Technical report, SCIENCE APPLICATIONS INTERNATIONAL CORP SAN DIEGO CA.

Matthew Gerber and Joyce Chai. 2010. Beyond nombank: A study of implicit arguments for nominal predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*.

Hongyu Gong, Suma Bhat, and Pramod Viswanath. 2018. Embedding syntax and semantics of prepositions via tensor decomposition. In *Proceedings of the 2018 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-18)*.

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Proceedings of the 2015 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-15)*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*.

Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. 2016. Semantic object parsing with graph LSTM. In *European Conference on Computer Vision*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-17)*.

Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. 2005. Simple algorithms for complex relation extraction with applications to biomedical IE. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31(1):71–106.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics* 5:101–115.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the ACL (EACL-17)*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL-18)*.

Kumutha Swampillai and Mark Stevenson. 2011. Extracting relations within and across sentences. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Michael Wick, Aron Culotta, and Andrew McCallum. 2006. Learning field compatibilities to extract database records from unstructured text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-06)*.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018a. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823* .

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018b. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-18)*.

Katsumasa Yoshikawa, Sebastian Riedel, Tsutomu Hirao, Masayuki Asahara, and Yuji Matsumoto. 2011. Coreference based event-argument relation extraction on biomedical text. *Journal of Biomedical Semantics* 2(5):S6.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research* 3(Feb):1083–1106.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-17)*.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state lstm for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL-18)*.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

Hierarchical Relation Extraction with Coarse-to-Fine Grained Attention

Xu Han[1,2,3]*, **Pengfei Yu**[2,3,4]*, **Zhiyuan Liu**[1,2,3]†, **Maosong Sun**[1,2,3], **Peng Li**[5]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Institute for Artificial Intelligence, Tsinghua University, Beijing, China
[3]State Key Lab on Intelligent Technology and Systems, Tsinghua University, Beijing, China
[4]Department of Electronic Engineering, Tsinghua University, Beijing, China
[5]Pattern Recognition Center, WeChat, Tencent, China

Abstract

Distantly supervised relation extraction employs existing knowledge graphs to automatically collect training data. While distant supervision is effective to scale relation extraction up to large-scale corpora, it inevitably suffers from the wrong labeling problem. Many efforts have been devoted to identifying valid instances from noisy data. However, most existing methods handle each relation in isolation, regardless of rich semantic correlations located in relation hierarchies. In this paper, we aim to incorporate the hierarchical information of relations for distantly supervised relation extraction and propose a novel hierarchical attention scheme. The multiple layers of our hierarchical attention scheme provide coarse-to-fine granularity to better identify valid instances, which is especially effective for extracting those long-tail relations. The experimental results on a large-scale benchmark dataset demonstrate that our models are capable of modeling the hierarchical information of relations and significantly outperform other baselines. The source code of this paper can be obtained from `https://github.com/thunlp/HNRE`.

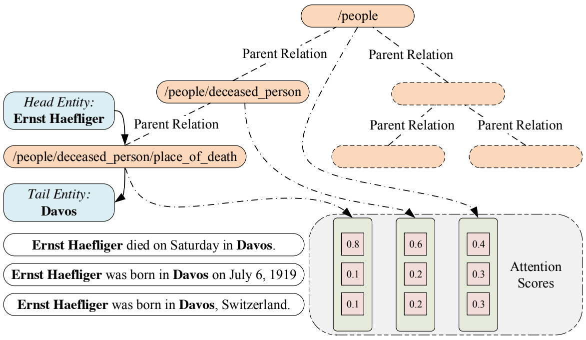

Figure 1: An example of hierarchical relation extraction.

1 Introduction

Relation extraction (RE) aims to predict relational facts from plain text. Conventional supervised RE models (Zelenko et al., 2003; Mooney and Bunescu, 2006) usually suffer from the lack of high-quality training data, because manual labeling of training data is time-consuming and human-intensive. Mintz et al. (2009) propose distant supervision to automatically label training instances by aligning existing knowledge graphs (KGs) and text: For an entity pair in KGs, those sentences containing both the entities will be labeled with the corresponding relation of the entity pair in KGs. RE relies on distant supervision to scale up to large-scale training corpora. However, this automatic mechanism is inevitably accompanied by the wrong labeling problem, because not all sentences containing two entities can exactly express their relations in KGs, e.g., we may mistakenly label "*Bill Gates* retired from *Microsoft*" with the relation `business/company/founders`.

To alleviate the wrong labeling problem, many efforts (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Zeng et al., 2015) have been devoted to identifying valid instances from noisy data, especially the recent state-of-the-art attention-based methods (Lin et al., 2016; Ji et al., 2017; Liu et al., 2017; Wu et al., 2017). Nevertheless, each relation is handled in isolation in most existing methods. For each relation, there is often a separate model (e.g. neural attention scheme) to select relation-related informative instances from noisy data, regardless of rich semantic correlations among relations, typically located in the form of relation hierarchies.

We take the KG Freebase (Bollacker et al., 2008) as an example, in which relations are labeled as hierarchical structures. For example, the

---

* indicates equal contribution

† Corresponding author: Z.Liu(liuzy@tsinghua.edu.cn)

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2236–2245
Brussels, Belgium, October 31 - November 4, 2018.

relation `/location/province/capital` in Freebase indicates the relation between a province and its capital. It is labeled under the `location` branch. Under this branch, there are some other relations `/location/location/contains` and `/location/country/capital`, which are closely correlated to each other. The rich correlations among relations are well revealed by these relation hierarchies. In fact, McCallum et al. (1998) take advantage of hierarchies of classes to improve classification models and inspire many later models (Rousu et al., 2005; Weinberger and Chapelle, 2009). Furthermore, the hierarchical information of entities in KGs has also been utilized and demonstrated to be effective for model enhancement (Hu et al., 2015; Xie et al., 2016).

To take advantage of the rich correlated information among relations, we propose a novel hierarchical attention scheme via utilizing the relation hierarchies, rather than directly utilizing hierarchical information as features for models. Similar to the conventional attention-based method, our method also computes an attention score for each instance according to its significance of expressing the corresponding relation. The key difference is that, as illustrated in Figure 1, our hierarchical attention scheme follows the relation hierarchies to compute scores for those instances containing the same entity pair on the each layer of the hierarchies.

The hierarchical attention scheme provides coarse-to-fine granularity for identifying valid instances. The attention on the bottom layer can capture more specific features of the relation, which has a comparable ability of fine-grained instance selection like conventional attention-based methods. The attention on the top-layer can capture the common features shared by several related subrelations, which provides coarse-grained instance selection. Since there are more sufficient data for training the top-layer attention, the whole hierarchical attention scheme can enhance RE models for solving those long-tail relations.

We conduct experiments on a large-scale benchmark dataset for RE in this paper. The experimental results show that the proposed coarse-to-fine grained attention scheme based on relation hierarchies significantly outperforms other baseline methods, even as compared to the recent state-of-the-art attention-based models, especially for those long-tail relations.

2 Related Works

Supervised models (Zelenko et al., 2003; Zhou et al., 2005; Mooney and Bunescu, 2006) for RE require adequate amounts of annotated data for their training. It is time-consuming to manually label large-scale training data. Hence, Mintz et al. (2009) propose distant supervision to automatically label data. Distant supervision inevitably accompanies with the wrong labeling problem. To alleviate the noise issue caused by distant supervision, Riedel et al. (2010) and Hoffmann et al. (2011) propose multi-instance learning (MIL) mechanisms. Riedel et al. (2013) propose universal schema to transmit information between relations of KGs and textual patterns to enhance extraction performance.

These early RE methods mainly extract semantic features using NLP tools to build relation classifiers. Recently, neural models have been widely used for RE. These neural models can accurately capture textual relations without explicit linguistic analysis (Zeng et al., 2014; Xu et al., 2015; Santos et al., 2015; Zhang and Wang, 2015; Verga et al., 2016; Verga and McCallum, 2016). Zeng et al. (2015) employ the MIL scheme by selecting one most valid instance for distantly supervised neural relation extraction (NRE), whose denoising capability is far from satisfactory because most informative instances are neglected. Lin et al. (2016) and Zhang et al. (2017) propose neural attention schemes to select those informative instances. To further improve the attention performance, some works incorporate knowledge information (Zeng et al., 2017; Ji et al., 2017; Han et al., 2018) and advanced training strategies (Liu et al., 2017; Huang and Wang, 2017). More sophisticated mechanisms, such as reinforcement learning (Feng et al., 2018; Zeng et al., 2018) and adversarial training (Wu et al., 2017), have also been adapted for RE recently.

However, most existing works model each relation in isolation to identify informative instances, neglecting rich correlations among relations, especially the hierarchical information of those relations. Hierarchical information is widely applied for model enhancement, especially for classification models (McCallum et al., 1998; Rousu et al., 2005; Weinberger and Chapelle, 2009; Zhao et al., 2011; Bi and Kwok, 2011; Zhou et al., 2011; Verma et al., 2012). Many efforts are also devoted to utilizing hierarchical information in KGs.

Leacock and Chodorow (1998) and Ponzetto and Strube (2007) adopt hierarchical information derived from KGs to construct concept relatedness. Morin and Bengio (2005) propose a neural language model by utilizing hierarchical information in WordNet. Further, Hu et al. (2015) learn entity representations by considering the whole entity hierarchies of Wikipedia and inspire many works (Krompaß et al., 2015; Xie et al., 2016) to utilize hierarchical type structures to help the representation learning of KGs.

Different from the recent hierarchical models that mainly focus on entity hierarchies and directly utilize hierarchical information as simple features, we incorporate relation hierarchies to build a hierarchical attention scheme with coarse-to-fine granularity to enhance RE performance. As compared with the existing models for RE, our models could take advantage of relation correlations to better identify informative instances, especially for those long-tail relations, by transferring the knowledge from their related relations of high-frequency.

3 Methodology

In this section, we introduce the overall framework of our hierarchical attention for RE, starting with notations and definitions.

3.1 Notations

We denote a KG as $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{F}\}$, where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{F}$ indicate the sets of entities, relations and facts respectively. $(h, r, t) \in \mathcal{F}$ indicates that there is a relation $r \in \mathcal{R}$ between $h \in \mathcal{E}$ and $t \in \mathcal{E}$. We follow the MIL setting and split the entire instances into multiple entity-pair bags $\{\mathcal{S}_{h_1,t_1}, \mathcal{S}_{h_2,t_2}, \ldots\}$. Each bag $\mathcal{S}_{h_i,t_i}$ contains multiple instances $\{s_1, s_2, \ldots\}$ mentioning both the entities h_i and t_i. The distant supervision mechanism will label the bag with the corresponding relation of the mentioned entity pair. Each instance s in these bags is denoted as a word sequence $s = \{w_1, w_2, \ldots\}$.

3.2 Framework

Given an entity pair (h, t) and its entity-pair bag $\mathcal{S}_{h,t}$, we adopt our models to measure the probability of each relation $r \in \mathcal{R}$ holding between the pair. As shown in Figure 2, the overall framework of our models includes a sentence encoder and a coarse-to-fine grained hierarchical attention. The sentence encoder adopts several convolutional neural networks to represent sentence semantics with embeddings, and the hierarchical attention is used to select the most informative instances to exactly express their relations.

For each instance $s_i \in \mathcal{S}_{h,t}$, we use the sentence encoder to represent its semantic information as an embedding $\mathbf{s}_i$. The details of the sentence encoder will be introduced in Section 3.3. Since not all instances in the bag $\mathcal{S}_{h,t}$ are positive to express the relation between h and t, we apply the hierarchical attention to compute an instance weight α_i for each instance s_i. The details of the hierarchical attention will be introduced in Section 3.4. We build the global textual relation representation $\mathbf{r}_{h,t}$ with the weighted sum of instance output embeddings,

$$\mathbf{r}_{h,t} = \sum_{i=1}^{m} \alpha_i \mathbf{s}_i, \quad s_1, \ldots, s_m \in \mathcal{S}_{h,t}. \tag{1}$$

Here α_i is the instance weight for the ith instance output embedding $\mathbf{s}_i$. By taking $\mathbf{r}_{h,t}$ as the textual relation representation of the entity pair (h, t), we estimate its probability over each relation $r \in \mathcal{R}$, i.e., whether there is a specific relation r between h and t. We define the conditional probability $P(r|h, t, \mathcal{S}_{h,t})$,

$$P(r|h, t, \mathcal{S}_{h,t}) = \frac{\exp(\mathbf{o}_r)}{\sum_{\tilde{r}\in\mathcal{R}} \exp(\mathbf{o}_{\tilde{r}})}, \tag{2}$$

where $\mathbf{o}$ is the scores of all relations, which is defined as follows,

$$\mathbf{o} = \mathbf{M}\mathbf{r}_{h,t}, \tag{3}$$

where $\mathbf{M}$ is the representation matrix to calculate the relation scores.

3.3 Sentence Encoder

Given an instance s containing two entities, we apply several neural architectures to encode the instance into its corresponding embeddings $\mathbf{s}$.

Input Layer

The input layer of the sentence encoder aims to embed both semantic information and positional information of words into their input embeddings.

Word Embedding is proposed by Hinton (1986), which aims to transform words into distributed representations to capture syntactic and semantic meanings of words. Given a sentence s consisting of multiple words $s = \{w_1, \ldots, w_n\}$, we adopt Skip-Gram (Mikolov et al., 2013) to

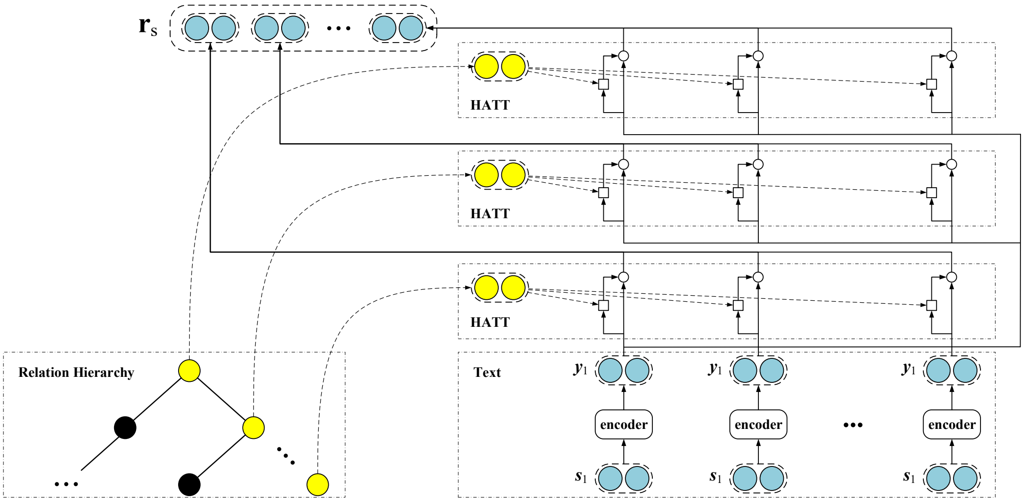

Figure 2: The architecture of hierarchical attention model.

compute all k_w-dimensional word embeddings $\{\mathbf{w}_1, \dots, \mathbf{w}_n\}$.

Position Embedding is proposed by Zeng et al. (2014). Position embedding is used to embed the relative distances of each word to the two entities into two k_p-dimensional vectors. By concatenating the distance embeddings for the current word w_i to the both head and tail entities, we get a unified position embedding $\mathbf{p}_i \in \mathbb{R}^{k_p \times 2}$.

For each word w_i, we concatenate its word embedding $\mathbf{w}_i$ and position embedding $\mathbf{p}_i$ to build its input embedding $\mathbf{x}_i \in \mathbb{R}^{k_i} (k_i = k_w + k_p \times 2)$.

Encoding Layer

The encoding layer aims to compose the input embeddings of the given instance into its corresponding instance embedding. In this paper, we choose two convolutional neural architectures, CNN (Zeng et al., 2014) and PCNN (Zeng et al., 2015), to encode input embeddings into instance embeddings.

Other neural architectures such as recurrent neural architectures (Zhang and Wang, 2015) can also be used as sentence encoders. Since previous works show that both convolutional and recurrent architectures can achieve comparable state-of-the-art performance, we simply select convolutional architectures in this paper. Note that, our hierarchical attention scheme is designed independently to the encoder choices, hence it can be easily adapted to fit other encoder architectures.

CNN slides a convolution kernel with the window size m over the input sequence $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ to get the k_h-dimensional hidden embeddings.

$$\mathbf{h}_i = \text{CNN}\big(\mathbf{x}_{i-\frac{m-1}{2}}, \dots, \mathbf{x}_{i+\frac{m-1}{2}}\big). \tag{4}$$

A max-pooling is then applied over these hidden embeddings to output the final instance embedding $\mathbf{s}$ as follows,

$$[\mathbf{s}]_j = \max_{1 \le i \le n} \{[\mathbf{h}_i]_j\}, \tag{5}$$

where $[\cdot]_j$ is the j-th value of a vector.

PCNN is an extension to CNN, which also adopts a convolution kernel to obtain hidden embeddings. Then, a piecewise max-pooling is applied over the hidden embeddings,

$$\begin{aligned} [\mathbf{s}^{(1)}]_j &= \max_{1 \le i \le i_1} \{[\mathbf{h}_i]_j\}, \\ [\mathbf{s}^{(2)}]_j &= \max_{i_1+1 \le i \le i_2} \{[\mathbf{h}_i]_j\}, \\ [\mathbf{s}^{(3)}]_j &= \max_{i_2+1 \le i \le n} \{[\mathbf{h}_i]_j\}, \end{aligned} \tag{6}$$

where $[\cdot]_j$ is the j-th value of a vector, i_1 and i_2 are entity positions. The final instance embedding $\mathbf{s}$ is achieved by concatenating these three pooling results as follows,

$$\mathbf{s} = [\mathbf{s}^{(1)}; \mathbf{s}^{(2)}; \mathbf{s}^{(3)}]. \tag{7}$$

3.4 Hierarchical Selective Attention

Given the entity pair (h, t) and its bag of instances $S_{h,t} = \{s_1, s_2, \dots, s_m\}$, we achieve the instance embeddings $\{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_m\}$ using the sentence encoder. Afterwards, we apply a hierarchical selective attention over them to get the textual relation representation $\mathbf{r}_{h,t}$ for extracting relations. In this part, we will first introduce a plain selective attention, and then introduce our hierarchical attention.

Plain Selective Attention

The plain selective attention scheme computes the attention score α_i for each instance s_i to indicate how well the instance can express the relation between the two entities. We assign a query vector $\mathbf{q}_r$ to each relation $r \in \mathcal{R}$ and the attention for each sentence in $\mathcal{S}_{h,t} = \{s_1, s_2, \ldots, s_m\}$ is defined as follows,

$$\begin{aligned} e_i &= \mathbf{q}_r^\top \mathbf{W}_s \mathbf{s}_i, \\ \alpha_i &= \frac{\exp(e_i)}{\sum_{j=1}^{m} \exp(e_j)}, \end{aligned} \tag{8}$$

where $\mathbf{W}_s$ is the weight matrix. The attention scores can be used in Eq. 1 to compute textual relation representations. For simplicity, we denote such a plain selective attention operation as the following equation,

$$\mathbf{r}_{h,t} = \mathrm{ATT}(\mathbf{q}_r, \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_m\}). \tag{9}$$

Hierarchical Selective Attention

The inherent hierarchical structure of relations lead us to modeling hierarchical attention. Generally, given a relation set $\mathcal{R}$ of a KG $\mathcal{G}$ (e.g. Freebase), which consists of base-level relations (e.g. `/location/province/capital`), we can generate the corresponding higher-level relation set $\mathcal{R}^H$. The relations in the high-level set (e.g. `location`) are more general and common, which usually contain several sub-relations in the base-level set. We assume that the sub-relations of different relations are disjoint, in other words, the relation hierarchies are tree-structured. The generation process can be done recursively. In practice, we start from $\mathcal{R}^0 = \mathcal{R}$ which is the set of all relations we focus for RE, and generate $k-1$ times to get a total of k-level hierarchical relation sets $\{\mathcal{R}^0, \mathcal{R}^1, \ldots, \mathcal{R}^{k-1}\}$.

As shown in Figure 2, for a relation $r = r^0 \in \mathcal{R} = \mathcal{R}^0$, which is the focus for RE, we construct its hierarchical chain of parent relations by backtracking the relation hierarchy as follows,

$$(r^0, \ldots, r^{k-1}) \in \mathcal{R}^0 \times \ldots \times \mathcal{R}^{k-1}, \tag{10}$$

where r^{i-1} is the sub-relation of r^i.

As with the plain attention, we assign a query vector $\mathbf{q}_r$ to each relation $r \in \bigcup_{i=0}^{k-1} \mathcal{R}^i$. With the hierarchical chain, we compute attention operations on the each layer of the relation hierarchies to obtain corresponding textual relation representations,

$$\mathbf{r}_{h,t}^i = \mathrm{ATT}(\mathbf{q}_{r^i}, \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_m\}). \tag{11}$$

During the training process, those relation query vectors of high-level relations (i.e., $\mathbf{q}_{r^i}$ with larger i) have more instances for training than those query vectors of base-level relations. Hence, the high-level query vectors are more robust for instance selection but with coarse-grained capability. In contrast, the base-level query vectors (i.e., $\mathbf{q}_{r^i}$ with smaller i) always suffer from data sparsity, especially for those long-tail base relations. Hence, the base-level query vectors can perform fine-grained instance selection but the performance is not stable.

Based on the hierarchical selective attention, we can simply concatenate the textual relation representations on different layers as the final representation,

$$\mathbf{r}_{h,t} = [\mathbf{r}_{h,t}^0; \ldots; \mathbf{r}_{h,t}^{k-1}]. \tag{12}$$

The representation $\mathbf{r}_{h,t}$ will be finally fed to compute the conditional probability $P(r|h, t, \mathcal{S}_{h,t})$ in Eq. 2. Note that, those high-level representations (i.e., $\mathbf{r}_{h,t}^i$ with larger i) are coarse-grained, and those base-level representations (i.e., $\mathbf{r}_{h,t}^i$ with smaller i) are fine-grained. These hierarchical representations can provide more informative information than single-layered attention for relation prediction, especially for those long-tail relations.

3.5 Initialization and Implementation Details

Here we introduce the learning and optimization details for our hierarchical attention model. During the training process, we minimize the cross entropy loss function. Given the collection of entity-pair bags $\pi = \{\mathcal{S}_{h_1,t_1}, \mathcal{S}_{h_2,t_2}, \ldots\}$ and corresponding labeled relations $\{r_1, r_2, \ldots\}$, we define the loss function as follows,

$$J(\theta) = -\frac{1}{|\pi|} \sum_{i=1}^{|\pi|} \log P(r_i | h_i, t_i, \mathcal{S}_{h_i,t_i}) + \lambda \|\theta\|_2^2, \tag{13}$$

where λ is a harmonic factor, and $\|\theta\|_2^2$ is the regularizer defined as L_2 normalization. All models are optimized using stochastic gradient descent (SGD).

4 Experiments

4.1 Datasets and Evaluation

We evaluate our models on the New York Times (NYT) dataset developed by Riedel et al. (2010), which is widely used in recent works (Lin et al., 2016; Zeng et al., 2017; Ji et al., 2017; Han et al., 2018; Liu et al., 2017; Wu et al., 2017; Huang and Wang, 2017; Feng et al., 2018; Zeng et al., 2018). The dataset has 53 relations including the `NA` relation which indicates relations of instances are not available. The training set has $522,611$ sentences, $281,270$ entity pairs and $18,252$ relational facts. In the test set, there are $172,448$ sentences, $96,678$ entity pairs and $1,950$ relational facts. In both the training and test set, we truncate the sentences which have more than 120 words into 120 words.

We evaluate all models in the held-out evaluation. It evaluates models by comparing the relational facts discovered from the test articles with those in Freebase and provides an approximate measure of precision without human evaluation. For evaluation, we draw precision-recall curves for all models. Besides precision-recall curves, we also show the precision values at the specific recall rate to conduct a more direct comparison, and calculate the micro and macro average precision scores to show the overall effect of different models. To further verify the effect of our hierarchical attention for few-shot entity pairs, we follow the previous works to report the Precision@N results. The dataset and baseline code can be found from Github [1] (Lin et al., 2016; Wu et al., 2017; Liu et al., 2017).

4.2 Parameter Settings

To fairly compare the results of our hierarchical attention models with those baselines, we also set most of the experimental parameters following Lin et al. (2016). Table 1 shows all experimental parameters used in the experiments. We apply dropout on the output layers of our models to prevent overfitting. For CNN, we set the dropout rate to 0.5. For PCNN, we observe that this model tends to overfit on the training set very quickly, and hence we set the dropout rate to 0.9 to alleviate the overfitting problem. We also pre-train the sentence encoder of PCNN before training our hierarchical attention.

Batch Size B	160
Learning Rate α	0.2
Hidden Layer Dimension k_c for CNNs	230
Word Dimension k_w	50
Position Dimension k_p	5
Convolution Kernel Size m	3

Table 1: Parameter settings.

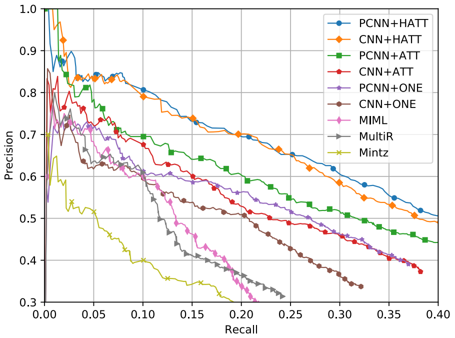

Figure 3: Precision-recall curves for the proposed model and various baseline models.

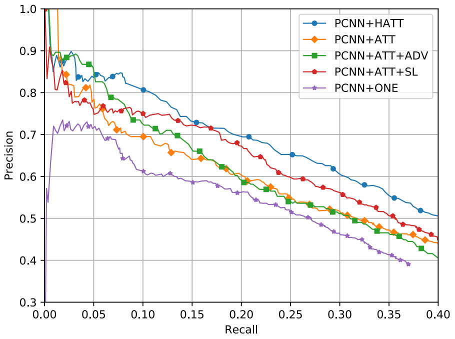

Figure 4: Precision-recall curves for the proposed model and various attention-based neural models.

4.3 Overall Evaluation Results

To evaluate the performance of our proposed hierarchical models, we compare the precision-recall curves of our models with various previous relation extraction models. The evaluation results are shown in Figure 3 and Figure 4. We report the results of the neural architectures including CNN and PCNN with various attention-based methods: **+HATT** is our hierarchical attention method; **+ATT** is the plain selective attention method over instances (Lin et al., 2016); **+ATT+ADV** is the denoising attention method by adding a small adversarial perturbation to instance embeddings (Wu et al., 2017); **+ATT+SL** is the attention-based model using soft-labeling method to mitigate the side effect of the wrong label-

---

[1] NRE, AtNRE and soft-label-RE

ing problem at entity-pair level (Liu et al., 2017); **+ONE** is a vanilla MIL neural model without attention schemes (Zeng et al., 2015). We also compare our method with feature-based models, including **Mintz** (Mintz et al., 2009), **MultiR** (Hoffmann et al., 2011) and **MIML** (Surdeanu et al., 2012).

From the results, we observe that:

(1) All methods have reasonable precision when recall is smaller than 0.05. When the recall gradually grows, the performance of the feature-based methods drops much more faster than those neural models. It shows that human-designed features are very limited as compared to neural models, especially in a noisy data environment. Hence, for simplicity, we mainly show the results of our models and other attention-based neural models in the following experiments.

(2) Both for CNNs and PCNNs, the models with attention schemes outperform the vanilla models without attention schemes. Though vanilla neural models are powerful for relation classification, it is still difficult to address data noise. The attention-based methods apply attentions over multiple instances and dynamically reduce the influence of noisy instances, which can effectively improve the performance of RE and achieve the state-of-the-art results.

(3) As shown in both of the figures, the models using hierarchical attention (HATT) achieve the best results among all the attention-based models. Even when compared with PCNN+ATT+ADV and PCNN+ATT+SL which adopt sophisticated denoising schemes and extra information, our models still keep significant advantages. This indicates that, as compared to the conventional plain attention schemes which handle each relation in isolation, our method can better take advantage of the rich correlations among relations. We believe the performance of our hierarchical attention scheme can be further improved by adopting extra mechanisms like adversarial training, reinforcement learning and soft-labeling at entity-pair level, which will be left as our future work.

Method		0.1	0.2	0.3
CNN	+ATT	67.5	52.8	58.5
	+HATT	**78.9**	**69.9**	**58.5**
PCNN	+ATT	69.4	60.6	51.6
	+HATT	**80.6**	**69.5**	**60.7**
Method		Mean	Micro	Macro
CNN	+ATT	55.4	31.8	8.2
	+HATT	**69.1**	**41.7**	**16.5**
PCNN	+ATT	60.5	38.0	15.1
	+HATT	**70.3**	**42.3**	**17.0**

Table 2: Precision (%) of attention-based models for different recalls.

Training Instances		<100			<200		
Hits@K (Micro)		10	15	20	10	15	20
CNN	+ATT	<5.0	<5.0	21.1	<5.0	30.0	50.0
	+HATT	**5.3**	**36.8**	**52.6**	**40.0**	**60.0**	**70.0**
PCNN	+ATT	<5.0	10.5	47.4	33.3	43.3	66.7
	+HATT	**31.6**	**52.6**	**63.2**	**53.3**	**70.0**	**76.7**
Training Instances		<100			<200		
Hits@K (Macro)		10	15	20	10	15	20
CNN	+ATT	<5.0	<5.0	18.5	<5.0	16.2	33.3
	+HATT	**5.6**	**31.5**	**57.4**	**22.7**	**43.9**	**65.1**
PCNN	+ATT	<5.0	7.4	40.7	17.2	24.2	51.5
	+HATT	**29.6**	**51.9**	**61.1**	**41.4**	**60.6**	**68.2**

Table 3: Accuracy (%) of Hits@K on relations with training instances fewer than 100/200.

4.4 Effect of Hierarchical Attention for Different Relations

To further verify the effectiveness of our hierarchical attention method for different relations, we evaluate the RE performance of our method and conventional attention methods. Since we focus more on the performance of those top-ranked results, we report the precision scores when the recall is 0.1, 0.2, 0.3 and their mean. We also report micro average scores and macro average scores in this experiment. As an approximation of the area under the precision-recall curve, the micro average score gives a more complete view of the model performance. Since the micro average score generally overlooks the influences of those long-tail relations, we use the macro average score to give more emphasis on long-tail relations in test sets, which is often neglected by the previous works.

The evaluation results are shown in Table 2, and from the results we observe that: Our HATT method achieves consistent and significant improvements as compared to the plain ATT method. From the micro and macro average precision scores, we find that our HATT method effectively improves RE performance especially for those long-tail relations. As compared to the plain ATT method, our method can take advantage of correlations among relations to achieve the improvement, especially on the long-tail relations.

To further demonstrate the improvements in performance on long-tail relations after introduc-

Test Mode	ONE				TWO				ALL			
P@N	100	200	300	Mean	100	200	300	Mean	100	200	300	Mean
CNN+ONE	68.3	60.7	53.8	60.9	70.3	62.7	55.8	62.9	67.3	64.7	58.1	63.4
CNN+ATT	76.2	65.2	60.8	67.4	76.2	65.7	62.1	68.0	76.2	68.6	59.8	68.2
CNN+HATT	**88.0**	**74.5**	**68.0**	**76.8**	**85.0**	**76.0**	**73.0**	**78.0**	**88.0**	**79.0**	**77.7**	**81.6**
PCNN+ONE	73.3	64.8	56.8	65.0	70.3	67.2	63.1	66.9	72.3	69.7	64.1	68.7
PCNN+ATT	73.3	69.2	60.8	67.8	77.2	71.6	66.1	71.6	76.2	73.1	67.4	72.2
PCNN+HATT	**84.0**	**76.0**	**69.7**	**76.6**	**85.0**	**76.0**	**72.7**	**77.9**	**88.0**	**79.5**	**75.3**	**80.9**

Table 4: Top-N precision (P@N) for RE on the entity pairs with different number of instances (%).

Relation: `/people/person/children`		
High	Good	**David** and Jody Smith and their son **Nathan** of Ankeny , Iowa , stayed at the hotel, . . .
	Bad	. . . doting grandfather of Amanda, Lindsay, **David**, Alexa, Reese, Paige and **Nathan**.
Base	Good	. . . cherished grandfather of **David**, Michael, Jason, Vicky, Andrew, Sam and **Nathan**
	Bad	**David** and Jody Smith and their son **Nathan** of Ankeny, Iowa, stayed at the hotel . . .

Table 5: Example sentences for case study.

ing relation hierarchies, we extract a subset of the test dataset in which all the relations has fewer than $100/200$ training instances. We employ the Hits@K metric for evaluation. For each entity pair, the evaluation requires its corresponding golden relation in the first K candidate relations recommanded by the models. Because it is difficult for the existing models to extract long-tail relations, we select K from $\{10, 15, 20\}$. We report the both micro and macro average Hits@K accuracies for these subsets. From the evaluation results in Table 3, we observe that:

(1) For both CNN and PCNN models, our hierarchical attention outperforms the plain attention model. By taking advantage of the relation hierarchy, our models can learn better about long-tail relations via correlation information among relations. We also observe that even our hierarchical CNN model presents a better performance than the plain PCNN model. This shows the power of relation hierarchies, which makes our simpler CNN model outperforms the PCNN model on those long-tail relations.

(2) Although our HATT method has achieved obvious progress on the long-tail relations as compared with the plain ATT method, the results of all these methods are still far from satisfactory. This indicates that distantly supervised RE models suffer from not only the wrong labeling problem, but also the long-tail relation problem. We will incorporate more schemes and extra information to solve this problem in the future.

4.5 Effect of Hierarchical Attention with Different Instances

Since our method mainly focuses on modifications over selective attention, we also conduct Precision@N tests on those entity pairs with few instances following (Lin et al., 2016). We use the three test settings for this experiment: the ONE test set where we randomly select one instance for each entity pair for evaluation; the TWO test set where we randomly select two instances for each entity pair; the ALL test set where we use all instances for each remaining entity pair for evaluation. For the ONE and TWO test set, we intend to show that taking correlation information among relations into consideration can lead to a better relation classifier. The ALL test set is designed to show the effect of our attention over multiple instances. We report the precision values of top N triples extracted, where $N \in \{100, 200, 300\}$.

The evaluation results are shown in Table 4, and from the results we observe that:

(1) The performance of all methods is generally improved as the instance number increases. This shows that the selective attention model can effectively take advantage of information from multiple noisy instances by combining useful instances while discarding useless ones.

(2) Our HATT method has higher precision values in the ONE test set. This indicates that even in an insufficient information scenario, correlations among relations can be caught by our hierarchical attention.

4.6 Case Study

We give some examples of how our hierarchical selective attention takes effect in selecting the sentences. In Table 5, we display the sentences that are scored highest ("Good") or lowest ("Bad") by the attention of different hierarchical levels ("High" and "Base").

The relation `/people/person/children`

has fewer than 1000 training instances and it is a long-tail relation. For this relation, the instance recommended by the higher-level attention straightforwardly expresses the relational fact that *Nathan* is the child of *David* by telling that *Nathan* is *David*'s son, while the sentence with the low attention score actually gives the relationship of being at the same generation. On the contrary, the lower-level attention mistakenly assigns high attention to the incorrect sentence. This example shows that our hierarchical attention is beneficial for these long-tail relations.

5 Conclusion and Future Work

In this paper, we take advantage of relation hierarchies and propose a novel hierarchical instance-level attention for relation extraction. As compared with previous attention-based methods, our hierarchical attention provides coarse-to-fine granularity in instance selection and performs better extraction for long-tail relations. We conduct various experiments and the evaluation results show that incorporating the inherent hierarchical structure of relations into attention schemes can take advantage of correlations among relations and improve the performance significantly.

In the future, we plan to explore the following directions: (1) It will be promising to adopt extra information to help train more efficient models for solving the long-tail relation problem. (2) We may also combine our attention method with recent denoising methods to further improve model performance.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC No. 61572273, 61532010). This work is also funded by the Natural Science Foundation of China (NSFC) and the German Research Foundation (DFG) in Project Crossmodal Learning, NSFC 61621136008 / DFC TRR-169.

References

Wei Bi and James T Kwok. 2011. Multi-label classification on tree-and dag-structured hierarchies. In *Proceedings of ICML*, pages 17–24.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of KDD*, pages 1247–1250.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI*.

Geoffrey E Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of COGSCI*, volume 1, page 12.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*, pages 541–550.

Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric P Xing. 2015. Entity hierarchy embedding. In *Proceedings of ACL*, volume 1, pages 1292–1300.

Yi Yao Huang and William Yang Wang. 2017. Deep residual learning for weakly-supervised relation extraction. In *Proceedings of EMNLP*, pages 1803–1807.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

Denis Krompaß, Stephan Baier, and Volker Tresp. 2015. Type-constrained representation learning in knowledge graphs. In *Proceedings of ISWC*.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and wordnet similarity for word sense identification. *Proceedings of WordNet: An electronic lexical database*, 49(2):265–283.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*, pages 2124–2133.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of EMNLP*, pages 1790–1795.

Andrew McCallum, Ronald Rosenfeld, Tom M Mitchell, and Andrew Y Ng. 1998. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of ICML*, volume 98, pages 359–367.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*, pages 1003–1011.

Raymond J Mooney and Razvan C Bunescu. 2006. Subsequence kernels for relation extraction. In *Proceedings of NIPS*, pages 171–178.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of AISTATS*, pages 246–252.

Simone Paolo Ponzetto and Michael Strube. 2007. Knowledge derived from wikipedia for computing semantic relatedness. *Proceedings of JAIR*, 30:181–212.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*, pages 148–163.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL*, pages 74–84.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. 2005. Learning hierarchical multi-category text classification models. In *Proceedings of ICML*, pages 744–751.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL-IJCNLP*, pages 626–634.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*, pages 455–465.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of NAACL*, pages 886–896.

Patrick Verga and Andrew McCallum. 2016. Row-less universal schema. In *Proceedings of ACL*, pages 63–68.

Nakul Verma, Dhruv Mahajan, Sundararajan Sellamanickam, and Vinod Nair. 2012. Learning hierarchical similarity metrics. In *Proceedings of CVPR*, pages 2280–2287.

Kilian Q Weinberger and Olivier Chapelle. 2009. Large margin taxonomy embedding for document categorization. In *Proceedings of NIPS*, pages 1737–1744.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of EMNLP*, pages 1778–1783.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of IJCAI*, pages 2965–2971.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of EMNLP*, pages 1785–1794.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Proceedings of JMLR*, 3(Feb):1083–1106.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Incorporating relation paths in neural relation extraction. In *Proceedings of EMNLP*, pages 1768–1777.

Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Large scaled relation extraction with reinforcement learning. In *Proceedings of AAAI*.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of EMNLP*, pages 35–45.

Bin Zhao, Fei Li, and Eric P Xing. 2011. Large-scale category structure aware image categorization. In *Proceedings of NIPS*, pages 87–96.

Denny Zhou, Lin Xiao, and Mingrui Wu. 2011. Hierarchical classification via orthogonal transfer. In *Proceedings of ICML*.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL*, pages 427–434.

Label-Free Distant Supervision for Relation Extraction via Knowledge Graph Embedding

Guanying Wang[1], Wen Zhang[1], Ruoxu Wang[1], Yalin Zhou[1]
Xi Chen[1], Wei Zhang[23], Hai Zhu[23], and Huajun Chen[1]*
[1]College of Computer Science and Technology, Zhejiang University, China
[2]Alibaba-Zhejiang University Frontier Technology Research Center, China
[3]Alibaba Group, China
`21621253@zju.edu.cn`

Abstract

Distant supervision is an effective method to generate large scale labeled data for relation extraction, which assumes that if a pair of entities appears in some relation of a Knowledge Graph (KG), all sentences containing those entities in a large unlabeled corpus are then labeled with that relation to train a relation classifier. However, when the pair of entities has multiple relationships in the KG, this assumption may produce noisy relation labels. This paper proposes a label-free distant supervision method, which makes no use of the relation labels under this inadequate assumption, but only uses the prior knowledge derived from the KG to supervise the learning of the classifier directly and softly. Specifically, we make use of the type information and the translation law derived from typical KG embedding model to learn embeddings for certain sentence patterns. As the supervision signal is only determined by the two aligned entities, neither hard relation labels nor extra noise-reduction model for the bag of sentences is needed in this way. The experiments show that the approach performs well in current distant supervision dataset.

1 Introduction

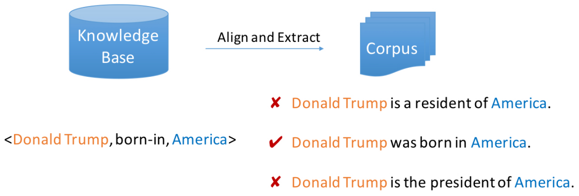

Figure 1: The mislabeled sentences produced by Distant Supervision.

Distant Supervision was first proposed by Mintz (2009), which used seed triples in Freebase instead of manual annotation to supervise text. It marked text as relation r if (h, r, t) can be found in a known KG, where (h, t) is the pair of entities contained in the text. This method can generate large amounts of training data, therefore widely used in recent research. But it can also produce much noise when there are multiple relations between the entities. For instance in Figure 1, we may wrongly mark the sentence "*Donald Trump is the president of America*" as relation *born-in*, with the seed triple (*Donald Trump, born-in, America*).

Previous works have tried different ways to address this issue. One way named Multi-Instance Learning(MIL) divided the sentences into different bags by (h, t), and tried to select well-labeled sentences from each bag (Zeng et al., 2015) or reduced the weight of mislabeled data (Lin et al., 2016). Another way tended to capture the regular pattern of the translation from true label to noise label, and learned the true distribution by modeling the noisy data (Riedel et al., 2010; Luo et al., 2017). Some novel methods like (Feng et al., 2017) used reinforcement learning to train an instance-selector, which will choose true labeled sentences from the whole sentence set. These methods focus on adding an extra model to reduce the noisy label. However, stacking extra model does not fundamentally solve the problem of inadequate supervision signals of distant supervision, and will introduce expensive training costs.

Another solution is to exploit extra supervision signal contained in a KG. Weston (2013) added the confidence of (h, r, t) in the KG as extra supervision signal. Han (2018) used mutual attention of KG and text to calculate a weight distribution of train data. Both of them got a better performance by introducing more information from KG. However, they still used the hard relation label derived from distant supervision, which also brought

---

* Corresponding author.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2246–2255
Brussels, Belgium, October 31 - November 4, 2018.

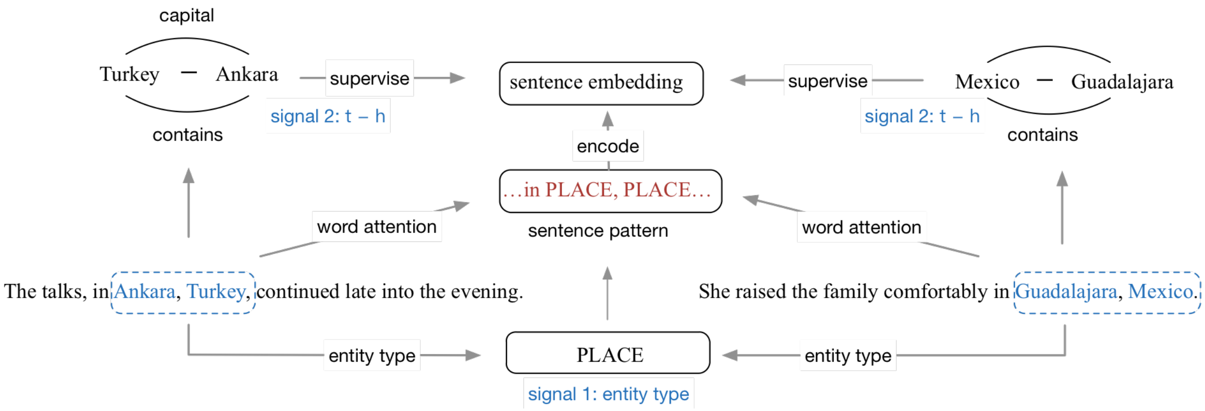

Figure 2: An instance of our label-free distant supervision method.

in much noise.

In this paper, we tend to avoid supervision by hard relation labels, and make full use of prior knowledge from a KG as soft supervision signal. We consider the TransE model proposed by Bordes (2013), which encodes entities and relations of a KG into a continuous low-dimensional space with the translation law $h + r \approx t$, where h, r, t describe the head entity, the relation and the tail entity respectively. Inspired by TransE model, we use $t - h$, instead of a concrete relation label r, as the supervision signal and make the sentence embedding close to $t - h$. Concrete relation labels may introduce mislabeled sentences, while $t - h$ is label-free, which is only determined by the two aligned entities and the the translation law.

Our assumption is that *each relation r in a KG has one or more sentence patterns that can describe the meaning of r*. For the example in Figure 2, we first replace the entity mentions in a sentence with the types of the aligned entities in the KG to form a sentence pattern. For example, "in Guadalajara, Mexico" will be replaced by "in PLACE, PLACE" to form a sentence pattern "in A, B" which conveys the meaning of "B contains A" and indicates the relation $contains$. For this sentence pattern, there may be a group of sentences sharing the same pattern but with different aligned entity pairs. In the first sentence "*The talks, in Ankara, Turkey, continued late into the evening*", $(Turkey - Ankara)$ implies both "*/location/country/capital*" and "*/location/location/contains*" as there are multiple relations between Ankara and Turkey in the KG. But in the similar sentence "*She raised the family comfortably in Guadalajara, Mexico.*", $(Mexico - Guadalajara)$ only implies "*/location/location/contains*" as there is no relation of "*/location/country/capital*" between Mexico and Guadalajara in the KG. As both $(Turkey - Ankara)$ and $(Mexico - Guadalajara)$ will be used to supervise the learning of the encoder for the pattern "in A, B", it makes the embedding of the sentence pattern closer to the correct relation "*/location/location/contains*" instead of the wrong relation "*/location/country/capital*". In this way, we do not need to label the sentences with the hard relation labels anymore.

The main contributions of this paper can be summarized as follows:

- As compared to existing distant supervision for relation extraction, our method makes better use of the prior knowledge derived from KG to address the wrong labeling problem.

- The proposed approach tends to supervise the learning process directly and softly by the type information and translation law, both derived from KG. Neither hard labels nor extra noise-reduction model for the bag of sentences is needed in this way.

- In the experiments, we show that the label-free approach performs well in current distant supervision dataset.

2 Related works

Relation extraction is intended to find the relationship between two entities given an unstructured text. Traditional methods use artificial characteristics or tree kernels to train a classification model (Culotta and Sorensen, 2004; Guodong et al., 2002). Recent works concentrate on deep neural networks to avoid error propagation during generating features (Ebrahimi and Dou, 2010; Zeng et al., 2014; Zhou et al., 2016; Zheng et al., 2017). More complicated models were proposed to learn deeper semantic features, like PCNN (Zeng et al., 2015) and attention pooling CNN (Wang et al., 2016), graph LSTMs (Peng et al., 2017).

Most of the early works were trained on the standard dataset by manual annotation, such as SemEval-2010 Task 8. In actual scenarios, it will cost a lot of manual resources to generate labeled data. Distant supervision (Mintz et al., 2009) aimed to obtain large-scale training data automatically, which becomes the most versatile supervision method. However, it suffers from the noisy label problem. Many works concentrate on dealing with the noise of distant supervision. Multi-instance learning (Riedel et al., 2010; Surdeanu et al., 2012) addresses the problem in bag-level, which divides sentences into different bags by (h, t). Zeng (2015) selects the most correct sentence from each bag. Lin (2016) introduces attention mechanism by distributing different weight to each sentence in the same bag, which reduces the effect of noisy labels and increases utilization of train data. Luo (2017) uses a transition matrix to characterize the inherent noise, convert true distribution to noise distribution. The model is enhanced by curriculum learning. Feng (2017) trains an instance selector to select correct labeled sentences by reinforcement learning.

Most of the above methods introduce a complicated extra model to deal with the noisy label problem. Our work tends to avoid the noisy label from distant supervision, by using entity information and translation law in KG to introduce more supervision signal.

KG is composed of many triples like (*head, relation, tail*), which describe relationships between head entities and tail entities. TransE is first proposed by (Bordes et al., 2013) to encode triples into a continuous low-dimensional space, which based on the translation $h+r \approx t$. Many follow-up works like TransH (Wang et al., 2014), DistMult (Yang et al., 2014), and TransR (Lin et al., 2015), proposed advanced method of translation by introducing different embedding spaces. Some recent works attempt to jointly learn text and KG triples, including (Xie et al., 2016) and (Xiao et al., 2016). These models tend to strengthen the representation of entities and relationships for KG tasks, but not for text representation.

3 Methodology

Here we present LFDS (Label-Free Distant Supervision) that essentially avoids noisy labels introduced by traditional distant supervision. Figure 2 shows an instance of our method. First, we pre-train representations for entities and relations based on the translation law $h + r \approx t$ defined by typical KG embedding models such as TransE. Second, for each sentence in the train sets, we replace the entity mentions with the types of the entities in the KG. An attention mechanism is then applied to calculate the importance of words with regard to the sentence pattern. Third, we train the sentence encoder by the margin loss between $t-h$ and sentence embedding. Note we do not use the noisy relation labels to train the model. Finally, for prediction, we calculate the embedding of test sentences, then compare the sentence embedding with all relation embeddings learned by TransE, and choose the closest relation as our predicted result. We describe these four parts in details as below.

3.1 KG Embedding

We use typical KG embedding models such as TransE to pre-train the embedding of entities and relations. We intend to supervise the learning by $t - h$ instead of hard relation label r. Concretely speaking, given two entities, h and t, we regard the translation based upon TransE between h and t as the target relation representation. TransE interprets relationships as translations operating on low-dimensional embeddings of entities, with the formula $h+r \approx t$, where h, r, t represent head entity, relation, and tail entity separately. The model is proved to perform well in predicting the tail entity when given head entity and relation.

The problem is that there may be multiple relations between t and h. As the example in Figure 2, the vector calculated by $Turkey - Ankara$ contains information for both relations: "*/location/country/capital*" and "*/location/location/contains*". While supervising the learning of the sentence pattern "in PLACE, PLACE", it is difficult to distinguish the two relations by supervision signal from only one sentence. However, other sentences with the similar pattern but different aligned entity pairs can push the embedding of the pattern close to another vector, such as $Mexico - Guadalajara$, which only represents "*/location/location/contains*" relation. As a result, the pattern will be closer to its correct relation "*/location/location/contains*".

Our work chooses TransE instead of other KG embedding models such as TransH or TransR, because TransE builds representations for h and t independent from fixed relation type r as the model assumes we do not know the specific relation r when training the encoder with supervision from $t - h$.

3.2 Sentence Embedding

In order to get a better representation of sentences, we had tried a variety of NRE models, such as Bi-LSTM(Zhou et al., 2016), SDP-LSTM (Yan et al., 2015), and typical CNN models. We chose PCNN (Zeng et al., 2015) to encode the sentence finally, which performs the best in our experiments. The encoder contains three parts as below.

Word Embeddings and Attentions. Instead of encoding sentences directly, we first replace the entity mentions e in the sentences with corresponding entity types $type_e$ in the KG, such as PERSON, PLACE, ORGANIZATION, etc. We then pre-train the word embedding by word2vec.

Attention mechanism is further applied to capture the importance of words with regards to the types information of entities as we assume the words close to the types information are more important.

First, we calculate the similarity between each word w^j and two entity types respectively:

$$A_1^j = f(type_{e_1}, w^j) \tag{1}$$

$$A_2^j = f(type_{e_2}, w^j) \tag{2}$$

$f(type_e, w^j)$ is the similarity function, which is defined as cosine similarity in this paper. $type_{e_1}$ and $type_{e_2}$ are the embeddings of the two entity types. Then the weight distribution for each word can be derived by exponential function:

$$\alpha_1^j = \frac{exp(A_1^j)}{\sum_{i=1}^n exp(A_1^i)} \tag{3}$$

$$\alpha_2^j = \frac{exp(A_2^j)}{\sum_{i=1}^n exp(A_2^i)} \tag{4}$$

We use the average weights of two entities as the attention of word w^j. Finally, the word embedding WF^j is derived as follows:

$$WF^j = \frac{\alpha_1^j + \alpha_2^j}{2} * w^j \tag{5}$$

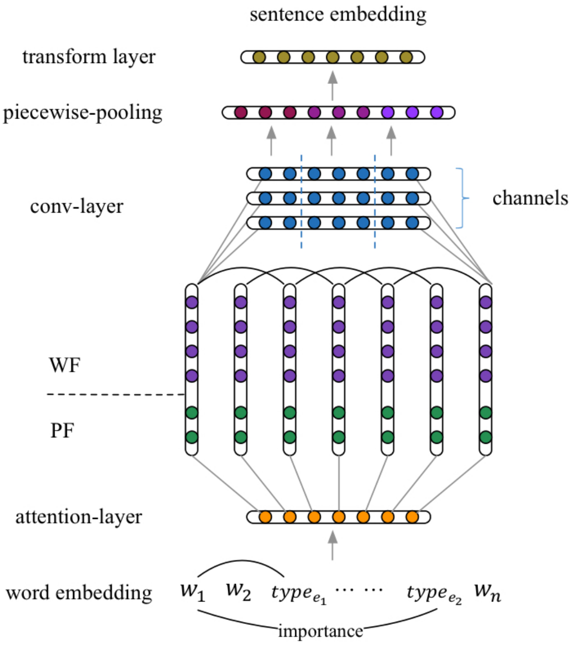

Figure 3: The sentence encoder with word attention and PCNN.

Position embedding. Zeng (2014) first proposed PFs to specify entity pairs. PF is a series of relative distances from current word to the two entities. For instance, for the sentence "*Damascus, the capital of Syria*", the distances from "capital" to the two entities are 3 and -2 respectively. The initial embedding matrix is randomly generated. Then we look up vector in the matrix by the two relative distances. The final position embedding will be the concatenation of $[PF_1, PF_2]$. As a result, we get a representation for each word:

$$w^j = [WF^j, PF_1^j, PF_2^j]$$

Then the input sentence representation will be:

$$x = w^1, w^2, ..., w^n$$

Piecewise-CNN. It was proved by (Zeng et al., 2015) that piecewise max pooling layer performs well in relation extraction, which tends to capture structural information between two entities. For each sentence, we use CNN to obtain a representation, then divide it into three parts by the two entities index. For each part, we perform a max pooling layer, thus we get 3-dimensional vector:

$$p_i = [p_{i_1}, p_{i_2}, p_{i_3}]$$

The shape of final vector will be $(bz, dc * 3)$, where bz represents batch size and dc is the number of channels.

The structure of whole model is shown in Figure 3.

3.3 Margin loss

In order to make the sentence embedding encoded by the PCNN model and relation embedding specified by $t - h$ based on the translation law as close as possible, we use margin loss with linear layer instead of cross-entropy loss with softmax layer. For the sentence embedding via PCNN layer, we perform a linear transformation to make its dimension equal to the relation representation.

$$s_e = \mathbf{W} * PCNN(x) + \mathbf{b} \tag{6}$$

Where **W** is the transformation matrix with shape $(dc*3, embedding\_dim)$. Then we define margin loss between $t - h$ and s_e as follows:

$$L = \sum_{s_e \in S} [(t - h - s_e + \gamma - (rand(t' - h, t - h') - s_e))]_+ \tag{7}$$

Where $rand(a, b)$ means choosing a or b. $t' - h$ is a negative instance of $t - h$, which is generated by randomly replacing t with other entities in KG, so does $t - h'$. For each sentence, we decrease the distance between $t - h$ and s_e, while increase the distance between the negative instance and s_e. γ is the reasonable margin between positive triple and negative triple. If the margin is already larger than γ, the loss of the sentence will be zero.

Another point to note is the special label NA in the dataset, which means there is no relationship between the two entities in the KG. In this case, $t - h$ is pointless and will confuse our encoder. To deal with this issue, we generate a fixed relation for NA, used as the negative relation for those sentences having some relationships. The minimum distance from NA to other relations is forced to be greater than $2 * \gamma$, where γ is the margin in loss function. When the model is used for prediction, the NA is also included.

The training target of our model is shown as Figure 4, including the sentence encoder we introduced above.

3.4 Prediction

We build a sentence encoder which can output a sentence embedding with the same dimension as relation embedding from the KG. For a new test sentence, we first encode it with the model, then calculate the similarity between the sentence embedding and the embeddings of all candidate relations. The most similar relation to the sentence embedding is the predicted category.

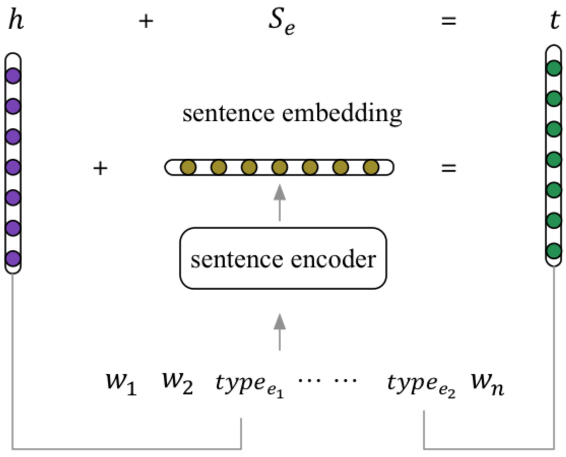

Figure 4: The training target.

$$r = arg\max_i(f(S_e, r_i)) \tag{8}$$

4 Experiments

Our experiments aim to provide positive evidence for the two main questions: (1) Whether or not the sentence pattern can express the essential part of the sentence? (2) Whether the abundant supervision signal in a KG is helpful to predict the true label for those mislabeled sentences?

To this end, we first introduce the widely used dataset for distant supervision, and evaluate our performance on the dataset. To further investigate the effectiveness of our model with noisy data, We divide the sentences in dataset into different categories, and show the study about some specific cases.

4.1 Datasets

The most widely used dataset was generated by Riedel (2010). It aligns the entities in Freebase with the New York Times (NYT) corpus, which contains all the news during 2005-2007. The sentences derived from news in 2005-2006 were used as the training data, while those from year 2007 were used as test data. After the alignment, there are 522,611 training sentences and 172,448 test sentences, labeled by 53 candidate relations in Freebase, and an extra label NA, which means there is no relation between the two entities in Freebase.

According to previous work (Mintz et al., 2009), we evaluate our model in the held-out evaluation and manual evaluation. The held-out evalu-

Parameter	**Settings**
Kernel size k	3
Sentence embedding size	100
Word embedding size	50
Position embedding size	5
Number of Channels	250
Margin	2
Learning rate	0.001
Dropout	0.5
Batch size	128

Table 1: Parameter settings.

ation calculates the precision-recall curves on the whole test set. For the false positives produced by the noisy labels in the test data, the precision will drop rapidly as the recall increases. In order to measure the precision, we need manual evaluation to check misclassified samples.

4.2 Experimental settings

4.2.1 Word Embeddings

In this paper, we use word2vec to train word embeddings on the NYT corpus. The window size of word2vec model is set as 5, and the embedding size is 50. We preserve those words appearing more than 10 times as vocabulary.

4.2.2 KG embeddings

We train the entities and relationships on FB40k[1] (Lin et al., 2015), which is generated for knowledge graph completion, with about 40,000 entities and 1318 relations. We set the embedding size as 100 instead of 50, which performs better in our experiment. Besides, we set the margin as 1 and train with learning rate 0.01. In order to test the performance of the vectors, we evaluate our model in KG completion tasks. The hit@10 of our final TransE model is 0.67, which is evaluated by predicting the closest 10 tail entities with specified head entities and relationships.

4.2.3 Parameter Settings

We use three-fold validation to determine the hyper-parameters. In the network layer, we try {3, 4, 5} for the kernel size, {100, 150, 200, 250, 300} for the number of channels, {5, 10, 15} for the position embedding size. In the update procedure, we use adaptive gradient descent with trying {0.1, 0.05, 0.01, 0.001} for the initial learning rate, and {64, 128, 256} for the mini-batch size. In the dropout operation, we set the probability as 0.5 referring to most of the classical experiments. Table 1 shows our final setting for all hyper-parameters.

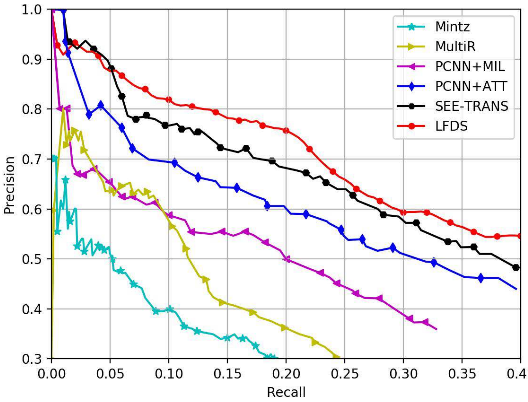

Figure 5: Performance comparison with Traditional methods.

4.3 Comparison with Traditional Methods

4.3.1 Held-out Evaluation

The held-out evaluation is performed directly on the test data. For the labels produced by distant supervision may not be precise, held-out evaluation is an approximate measure of our model, which is usually depicted by the precision-recall curve.

We select six representative models for comparison. *Mintz* (Mintz et al., 2009) proposed a feature-based model that first used distant supervision. *MultiR* (Hoffmann et al., 2011) is a multi-instance learning model under the at-least-one assumption. *PCNN+MIL* (Zeng et al., 2015) proposed the piece-wise pooling method, which is used as the encoder of our works. *PCNN+ATT* (Lin et al., 2016) performed selective attention over instances and got better results in the datasets. *SEE* (He et al., 2018) is a novel work that learned syntax-aware entity embedding for relation extraction and achieved state-of-the-art. The precision-recall curves are shown in Figure 5, where *LFDS* denotes our label-free distant supervision method.

We can observe from the figure that our LFDS method has an overall good performance compared to current works, especially with the growth of recall. It demonstrates that our model has a good classification ability in general, because the sentence pattern can capture the meaning of relations better than a sentence. The result can answer the first question we proposed at section 4.

---

[1] https://github.com/thunlp/KB2E.

Accuracy	Top 100	Top 200	Top 500	Average
Mintz	0.77	0.71	0.55	0.676
MultiR	0.83	0.74	0.59	0.720
PCNN+MIL	0.86	0.80	0.69	0.783
PCNN+ATT	0.86	0.83	0.73	0.807
SEE	0.91	0.87	0.77	0.850
LFDS	0.90	**0.88**	**0.83**	**0.869**

Table 2: Precision values for the top 100, 200 and 500 sentences.

4.3.2 Manual Evaluation

For the wrong labels produced by distant supervision, there will be many false positives in our evaluation inevitably, thus causing a sharp decline in the held-out precision-recall curves. Manual evaluation is necessary to evaluate the model more precisely. Following the previous works, we selected the top 100, top 200, and top 500 sentences, which is ranked by the predicted confidence, then evaluated the precision artificially. The result is shown in table 2.

We can see that the precision is higher than held-out evaluation, because manual evaluation avoid the effect of wrong labels. Our LFDS method achieved a consistently higher precision compared with current works, especially when recall increases. Compared to held-out evaluation, manual evaluation can show our model's ability in differentiating noisy sentence. Detail analysis will be shown in Section 4.4.

In the manual procedure, we found some wrong cases caused by entity types. The entity types in Freebase can be ambiguous, where "ORGANIZATION" may be confused with "PLACE". It causes error propagation in our experiments.

4.4 Case Study

To further prove the effectiveness of our model, especially in distinguishing noisy labels, we select some specific relationships for detail analysis. The noisy labels are produced by the entity pairs which have multiple relationships between them. In this case, different relationships will share the same entity pairs in knowledge graph. We defined this kind of relationships as "overlapping" relationships. The more entity pairs it shares with other relation, the overlapping degree of the relation is higher, which means the relation is harder to distinguish.

Case 1: Non-overlapping Relations. The first case is the non-overlapping relation. For triples of the non-overlapping relation r_1 as (h, r_1, t), there are few triples like (h, r_2, t) in KG, where r_2 is another relation in our candidate relations set. That means for this kind of relation, almost no noisy label will be produced. One of these relation is */business/person/company*. There are near 200 sentences in the test set, with our evaluation of precision achieving 0.98. It proves that our encoder with sentence pattern and label-free supervision is effective in basic classification, which is a convincing answer of the first question we proposed at section 4.

Case 2: Partly-overlapping Relations. The second case is the partly-overlapping relation, in which two relations may share a certain number of entity pairs in Freebase. For instance, the relation */location/country/capital* shares many entity pairs with */location/location/contains* but not all entity pairs in Freebase have both $capital$ and $contain$ relations.

For those entity pairs having both relations, traditional distant supervision would produce two labels for sentences such as:

> "The talks, in Ankara, Turkey, continued late into the evening."

The noisy labels in the train set are hard to differentiate. Recent noise reduction methods commit to improving the distinguishing ability of the model by adding extra models. Our experiment proves that our label-free supervision method not only achieves better differentiation performance but also does not need to train extra noise reduction models. Cases are shown in Table 3.

The prediction results indicate that the model is capable of learning the embedding of the sentence pattern we want. For instance, the model captures the pattern like "in PLACE, PLACE", and tends to predict the sentence with this pattern for */location/location/contains*, while the pattern "PLACE, the capital of PLACE" for */location/country/capital* respectively. When both two relations are labeled for the same sentence in the test set, our model can predict the correct label with the corresponding patterns.

Another similar but more interesting example is */people/person/nationality* and */people/person/place_lived*. In this case, the two relations share a certain number of entity pairs in Freebase like the previous example. But because of the incompleteness of Freebase, many sentences with only one label are actually wrongly labeled.

Sentence	**Label with normal distant supervision**	**Prediction with LFDS**	**Pattern**
The talks, in *Ankara*, *Turkey*, continued late into the evening.	/location/location/contains /location/country/capital	/location/location/contains	in PLACE, PLACE
..., said Mr.Cho, 25, who was born in *Seoul*, *South Korea*, and educated at a boarding school in Scotland.	/location/location/contains /location/country/capital	/location/location/contains	in PLACE, PLACE
On Wednesday, suicide bombings killed 33 people in *Algiers*, the capital of *Algeria*.	/location/location/contains /location/country/capital	/location/country/capital	PLACE, the capital of PLACE
Farah has lived in *India*, Europe and South Africa, and only started revisiting Mogadishu in 1996, after two decades away.	/people/person/nationality	/people/person/place_lived	PERSON lived in PLACE
He was George Mcgovern of South Dakota – not *Frank church* of *Idaho*, who was involved in other antiwar legislation.	/people/person/place_lived	/people/person/nationality	PERSON of PLACE

Table 3: The comparison between labels from normal distant supervision and our label-free relation prediction

For example,the sentence "Farah has lived in India, ..." is labeled with only one relation */people/person/nationality* because there is only one $nationality$ relation in Freebase. But the actual meaning of the sentence is to say $Farah$'s $place\_lived$ is $India$. This type of wrongly labeling problem is caused by incompleteness of Freebase which is very common for many other knowledge graphs.

However, our label-free method can correct this problem because it essentially learns the sentence patterns that are determined only by the sentence itself and the aligned entity pairs. As shown by the last two examples in Table 3, our model successfully learned the patterns "PERSON lived in PLACE" for */people/person/place_lived* and "PERSON of PLACE" for */people/person/nationality* respectively.

These instances show that our model is capable of learning some sentence patterns and mapping them to the corresponding relations in Freebase, which can distinguish noise sentences effectively. It indicates that our label-free supervision with prior knowledge introduced by the translation laws and entity types in KG is effective in avoiding noise, which can answer the second question we proposed at section 4 credibly.

Case 3: Mostly-overlapping Relations. The final case is mostly-overlapping relations, in which the two relations share most entity pairs in Freebase. One example is */people/person/place_of_birth*, which shares most of its entity pairs with */people/person/place_lived* in FB40k, because a person‘s birthplace and residence are likely to be the same. That means in the process of training with TransE, the two relations are updated by similar gradients, which will produce similar representations for $t - h$. In this case, the relations are really hard to differentiate, because there are not enough distinct supervision signals in the KG. We tend to resolve this situation in future work by utilizing prior knowledge derived from relation paths.

5 Conclusion

In this paper, we argue that the noise label problem in distant supervision is mainly caused by the incomplete use of KG information. Thus we propose a label-free distant supervision method, which supervises the learning of the embedding of sentence patterns by $t - h$ and entity types, instead of hard relation labels. We conducted experiments on the widely used relation extraction dataset and showed that with the recall increasing, our model performs better than state-of-the-art results. This demonstrates that our approach can effectively deal with the noise problem and encoding sentence pattern for relation extraction.

In the future, we plan to utilize more information in knowledge graphs to improve the distant supervision signal. For instance, the reasoning path can introduce new prior knowledge, which is a key direction in current works of KG. The path may produce new supervision signals for two entities even there is no direct connection between them. We also plan to apply this method to other datasets.

Acknowledgments

This work is funded by NSFC 61673338/61473260, and supported by Alibaba-Zhejiang University Joint Institute of Frontier Technologies.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *International Conference on Neural Information Processing Systems*, pages 2787–2795.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Meeting on Association for Computational Linguistics*, pages 423–429.

Javid Ebrahimi and Dejing Dou. 2010. Chain based rnn for relation classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1244–1249.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2017. Reinforcement learning for relation classification from noisy data.

Zhou Guodong, Su Jian, Zhang Jie, and Zhang Min. 2002. Exploring various knowledge in relation extraction. In *ACL 2005, Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, Usa*, pages 419–444.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text.

Zhengqiu He, Wenliang Chen, Zhenghua Li, Meishan Zhang, Wei Zhang, and Min Zhang. 2018. See: Syntax-aware entity embedding for neural relation extraction.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550.

Yankai Lin, Zhiyuan Liu, Xuan Zhu, Xuan Zhu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2181–2187.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Meeting of the Association for Computational Linguistics*, pages 2124–2133.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. pages 430–439.

Mintz, Mike, Steven, Jurafsky, and Dan. 2009. Distant supervision for relation extraction without labeled data. In *Joint Conference of the Meeting of the ACL and the International Joint Conference on Natural Language Processing of the Afnlp: Volume*, pages 1003–1011.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen Tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms.

Sebastian Riedel, Limin Yao, and Andrew Mccallum. 2010. Modeling relations and their mentions without labeled text. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465.

Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Meeting of the Association for Computational Linguistics*, pages 1298–1307.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. pages 1134–1137.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. Ssp: Semantic space projection for knowledge graph embedding with text descriptions.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions.

Xu Yan, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency path. *Computer Science*, 42(1):56–61.

Bishan Yang, Wen Tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases.

D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao. 2014. Relation classification via convolutional deep neural network.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Suncong Zheng, Yuexing Hao, Dongyuan Lu, Hongyun Bao, Jiaming Xu, Hongwei Hao, and Bo Xu. 2017. Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing*, 257(000):1–8.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Meeting of the Association for Computational Linguistics*, pages 207–212.

Extracting Entities and Relations with Joint Minimum Risk Training

Changzhi Sun[2], Yuanbin Wu[1, 2], Man Lan[1, 2], Shiliang Sun[2],
Wenting Wang[3], Kuang-Chih Lee[3], and Kewen Wu[3]

[1]Shanghai Key Laboratory of Multidimensional Information Processing
[2]Department of Computer Science and Technology, East China Normal University
[3]Alibaba Group
`{changzhisun}@stu.ecnu.edu.cn`
`{ybwu,mlan,slsun}@cs.ecnu.edu.cn`
`{nentiao.wwt,kuang-chih.lee,kewen.wukw}@alibaba-inc.com`

Abstract

We investigate the task of joint entity relation extraction. Unlike prior efforts, we propose a new lightweight joint learning paradigm based on minimum risk training (MRT). Specifically, our algorithm optimizes a global loss function which is flexible and effective to explore interactions between the entity model and the relation model. We implement a strong and simple neural network where the MRT is executed. Experiment results on the benchmark ACE05 and NYT datasets show that our model is able to achieve state-of-the-art joint extraction performances.

1 Introduction

Detecting entities and relations is usually the first step towards extracting structured knowledge from plain texts. Its goal is to identify text spans representing typed objects (*entities*) and semantic relations among those text spans (*relations*). For example, in the following sentence,

> [Associated Press]$_{\text{ORG}}$ [writer]$_{\text{PER}}$ [Patrick McDowell]$_{\text{PER}}$ in [Kuwait City]$_{\text{GPE}}$.

"Associate Press" is an organization entity (`ORG`), "writer" is a person entity (`PER`), and the two entities have an affiliation relation (`ORG-AFF`).

Two types of models have been applied to the extraction task, the pipeline model and the joint model. In the pipeline setting, the task is broken down into independently learned components (an entity model and a relation model). Despite its flexibility, the pipeline ignores interactions between the two models. For example, the entity model doesn't look at relation annotations which are useful for identifying entities (e.g., if an `ORG-AFF` relation exists, the entity model can only assign `ORG` and `AFF` to its entities). The joint setting, on the other hand, extracts entities and relations in a unified model, which can explore shared information and alleviate error propagations between models. Here we will focus on joint models.

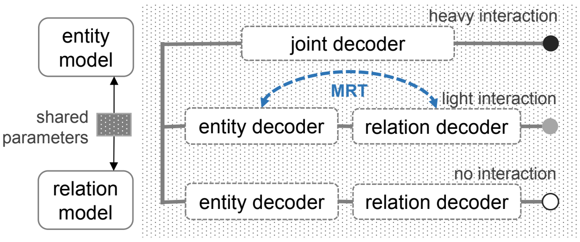

Figure 1: Paradigms of joint entity relation extraction.

One simple joint learning paradigm is through sharing parameters (Miwa and Bansal, 2016; Katiyar and Cardie, 2017). Typically, instead of training two independent models, the entity and relation model can share some input features or internal hidden states. It has an advantage that no additional constraint is required on the two sub-models. But the connections among sub-models are still not fully explored due to independent sub-model decoders. For example, to get signals from relation annotations, the entity model needs to wait for the relation model to update the shared parameters. To further utilize the interaction between decoders, some complex joint decoding algorithms (e.g., simultaneously decoding entities and relations in beam search) have been carefully studied (Li and Ji, 2014; Katiyar and Cardie, 2016; Zhang et al., 2017; Zheng et al., 2017). In this paradigm, it is important (and hard) to make a good balance between the exactness of the joint decoding algorithm and capacities of individual sub-models.

In this work, we propose a joint minimum risk training (MRT) (Och, 2003; Smith and Eisner, 2006) method for the entity and relation extraction

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2256–2265
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

task. It provides a lightweight way to strengthen connections between the entity model and the relation model, and keeps their capacities unaffected. Given an input $\mathbf{x}$ and a loss function $\Delta(\hat{\mathbf{y}}, \mathbf{y})$ (measuring the difference between model output $\hat{\mathbf{y}}$ and the true $\mathbf{y}$), MRT seeks a posterior $P(\hat{\mathbf{y}}|\mathbf{x})$ to minimize the expected loss $\mathbf{E}_{\hat{\mathbf{y}} \sim P(\hat{\mathbf{y}}|\mathbf{x})} \Delta(\hat{\mathbf{y}}, \mathbf{y})$. Comparing with prior joint decoding algorithms, the MRT-based algorithm is simple and can be applied to a broad range of entity relation extraction models without changing the original sub-models and decoders (Figure 1).

One advantage of the MRT-based method is that it can explicitly optimize a global sentence-level loss (e.g., F1 score) rather than local token-level losses. Therefore, it may catch more sentence-level information in the training time and match evaluation metrics better in the testing time. Furthermore, besides the handcrafted losses, we also try to directly learn a loss function from data during the joint MRT process. The automatically obtained loss would help MRT to calibrate its risk estimation with knowledge from the data distribution. On the other hand, comparing with preivous single task MRT, the joint MRT algorithm here will integrate messages from different sub-models, which is the key step for enhancing decoder interactions in the joint learning. As a result, the training of the entity model now can directly acknowledge the loss of the relation model (without waiting for shared parameters) and vice versa.

We compile the proposed joint MRT with a strong neural network-based model which uses recurrent neural networks (RNN) in the entity model and convolutional neural networks (CNN) in the relation model. On benchmark ACE05 and NYT datasets, we show that the new RNN+CNN structure outperforms previous neural network-based models. After adding the joint MRT, our model is able to achieve state-of-the-art performances.

To summarize, our main contributions include

1. proposing a new joint learning paradigm based on minimum risk training for the joint entity relation extraction task.

2. implementing a strong and simple neural-network-based entity relation extraction model which carries the proposed MRT algorithm. [1]

3. achieving state-of-the-art results on two benchmark datasets (ACE05 and NYT).

2 Related Work

In many pipelined entity relation extraction systems, one first learns an entity model, then learns a relation model based on entities generated by the entity model (Miwa et al., 2009; Chan and Roth, 2011; Lin et al., 2016). Such systems are often flexible to incorporate different data sources and different learning algorithms. However, they may also suffer from error propagation and data inefficiency. To tackle the problem, many recent studies try to develop joint extraction algorithms.

Parameter sharing is a basic strategy in joint learning paradigms. For example, in (Miwa and Bansal, 2016), the entity model is a sentence-level RNN, and the relation model is a dependency tree path RNN which takes hidden states of the entity model as features (i.e., the shared parameters). Our basic extraction model is similar to theirs but with a CNN-based relation model. Similarly, Katiyar and Cardie (2017) build a simplified relation model on the entity RNN using the attention mechanism.

To further explore interactions between the entity decoder and the relation decoder, some joint decoding algorithms were studied. For example, Katiyar and Cardie (2016) propose a CRF-based model which conducts joint decoding with augmented transition matrices. Zheng et al. (2017) propose to directly encode relations in the sequential labelling tag set. Both of them are exact decoding algorithms, but they need adding constraints on the relation model (e.g., Zheng et al. (2017) cannot handle entities which appear in multiple relations). On the other side, Li and Ji (2014) develop a joint decoding algorithm based on beam search. Zhang et al. (2017) study a globally normalized joint model. They retain capacities of sub-models, while their decoding algorithms are inexact. Here, we introduce MRT to the task, which is a more lightweight setting of joint learning.

Minimum risk training is a learning framework which tries to handle models with arbitrary discrepancy metrics (i.e., losses of a model output w.r.t. the true answer) (Och, 2003; Smith and Eisner, 2006; Gimpel and Smith, 2010). It has been successfully applied to many NLP tasks. Some recent work include (He and Deng, 2012; Shen et al., 2016) which apply MRT to (neural) ma-

---

[1] Our implementation is available at `https://github.com/changzhisun/entrel-joint-mrt`.

chine translation, (Xu et al., 2016) which develops a shift-reduce CCG parser to directly optimize F1, and (Ayana et al., 2016) which uses a MRT-based model for summarization. We note that most previous applications of MRT focus on a single job, while the joint entity relation extraction consists of two sub-tasks. Investigating MRT in joint learning scenarios is the main topic of this work.

Finally, the sampling algorithm of solving MRT is similar to the policy gradient algorithm in reinforcement learning (RL) (Sutton and Barto, 1998). Some recent NLP applications which share the key idea of MRT but are described with RL language also show promising results (e.g., dialog systems (Li et al., 2016), machine translation (Nguyen et al., 2017)). The idea of learning loss functions from data is similar to inverse reinforcement learning (Abbeel and Ng, 2004; Ratliff et al., 2006).

3 The Approach

We define the joint entity and relation extraction task following the setting of (Miwa and Bansal, 2016). Given an input sentence $s = w_1, \ldots, w_{|s|}$ (w_i is a word), the task is to extract a set of entities $\mathcal{E}$ and a set of relations $\mathcal{R}$. An entity $e \in \mathcal{E}$ is a sequence of words labelling with an entity type (e.g., person (`PER`), organization (`ORG`)). Let $\mathcal{T}_e$ be the set of possible entity types. A relation r is a triple (e_1, e_2, l), where e_1 and e_2 are two entities, l is a relation type describing the semantic relation between e_1 and e_2 (e.g., organization affiliation relation (`ORG-AFF`)). Let $\mathcal{T}_r$ be the set of possible relation types.

In our joint extraction method (Figure 2), we treat entity detection as a sequence labelling task (Section 3.1) and relation detection as a classification task (Section 3.2). Models of the two tasks share parameters and are trained jointly. Departing from previous joint learning algorithms (Miwa and Bansal, 2016; Katiyar and Cardie, 2017; Zhang et al., 2017), we introduce minimum risk training to the joint extraction model. It optimizes a global loss function and bridges the discrepancy between training and testing (Section 3.3).

3.1 Entity Detection

To represent entities in s, we assign a tag t_i to each word w_i following the BILOU tagging scheme: t_i takes a value in $\{\texttt{B-}*, \texttt{I-}*, \texttt{L-}*, \texttt{O}, \texttt{U-}*\}$, where `B`, `I`, `L` and `O` denote the begin, inside, end and outside of an entity, `U` denotes a single word entity and $* \in \mathcal{T}_e$ represents different entity types. For example, for a person (`PER`) entity "Patrick McDowell", we assign `B-PER` to "Patrick" and `L-PER` to "McDowell". Given an input sentence s, the entity model predicts the tags of words $\hat{\mathbf{t}} = \hat{t}_1, \hat{t}_2, \ldots, \hat{t}_{|s|}$ by learning from the true tags $\mathbf{t} = t_1, t_2, \ldots, t_{|s|}$.

We use a bidirectional long short term memory (bi-LSTM) network (Hochreiter and Schmidhuber, 1997) to solve the sequence labelling task. At each sentence position i, a forward LSTM chain computes a hidden state vector $\vec{\mathbf{h}}_i$ by recursively collecting information from the beginning of s to the current position i. Similarly, a backward LSTM chain collects information $\overleftarrow{\mathbf{h}}_i$ from the end of s to the position i.

$$\vec{\mathbf{h}}_i = \mathrm{LSTM}(\mathbf{x}_i, \vec{\mathbf{h}}_{i-1}; \vec{\boldsymbol{\theta}}),$$
$$\overleftarrow{\mathbf{h}}_i = \mathrm{LSTM}(\mathbf{x}_i, \overleftarrow{\mathbf{h}}_{i+1}; \overleftarrow{\boldsymbol{\theta}}).$$

The word representation $\mathbf{x}_i$ of w_i has two parts $\mathbf{x}_i = \mathbf{w}_i \oplus \mathbf{c}_i$ ($\oplus$ is the vector concatenation). $\mathbf{w}_i$ is a word embedding of word w_i (from an embedding look-up table $\mathbf{W}_e$). $\mathbf{c}_i$ is a character-based representation of w_i which is obtained by running a convolution neural network on the character sequence of w_i: $\mathbf{c}_i = \mathrm{CNN}(\mathrm{char}(w_i); \boldsymbol{\theta}_c)$.

To predict the tag $\hat{t}_i$, we combine the forward and the backward hidden vector to $\mathbf{h}_i = \vec{\mathbf{h}}_i \oplus \overleftarrow{\mathbf{h}}_i$, and apply a softmax function on $\mathbf{h}_i$ to get the posterior of $\hat{t}_i$,

$$P_{\text{ent}}(\hat{t}_i | s; \boldsymbol{\theta}_E) = \mathrm{Softmax}(\mathbf{W}_E \cdot \mathbf{h}_i), \quad (1)$$

where $\boldsymbol{\theta}_E = \{\mathbf{W}_e, \boldsymbol{\theta}_c, \vec{\boldsymbol{\theta}}, \overleftarrow{\boldsymbol{\theta}}, \mathbf{W}_E\}$ are parameters of the entity model. Given an input sentence s and its ground truth tag sequence $\mathbf{t}$, the training objective is to minimize $\mathcal{L}_{\text{ent}}$,[2]

$$\mathcal{L}_{\text{ent}}(\boldsymbol{\theta}_E) = -\frac{1}{|s|} \sum_{i=1}^{|s|} \log P_{\text{ent}}(\hat{t}_i = t_i | s; \boldsymbol{\theta}_E).$$

3.2 Relation Detection

Given a set of detected entities $\hat{\mathcal{E}}$ (obtaining from the entity tag sequence $\hat{\mathbf{t}}$), we consider all entity pairs in $\hat{\mathcal{E}}$ as candidate relations. The task of relation detection is to predict a relation type $l \in \mathcal{T}_r$ for each pair,[3] and output a relation set

---

[2] We have also tried biLSTM-CRF (Huang et al., 2015) as an advanced entity model, but performances are nearly the same in our experiments.

[3] We include a `NONE` relation type in $\mathcal{T}_r$ which means that there exists no relation between e_1 and e_2.

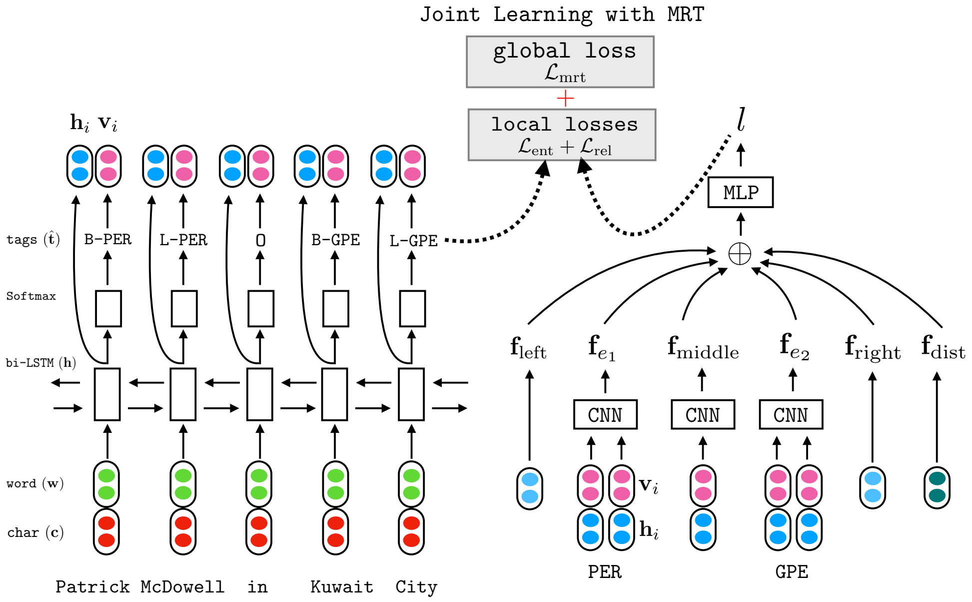

Figure 2: Our network structure for the joint entity and relation extraction.

$\hat{\mathcal{R}} = \{(e_1, e_2, l) | e_1, e_2 \in \hat{\mathcal{E}}, e_1 \neq e_2, l \in \mathcal{T}_r\}$. To build the relation model, we extract two types of features, namely, features regarding words in e_1, e_2 and features regarding contexts of the entity pair (e_1, e_2).

To extract features on words in e_1, e_2, we use two convolutional neural networks. Taking e_1 as an example, for each word w_i in e_1, we first collect w_i's bi-LSTM hidden vector $\mathbf{h}_i$ from the entity model. Then, we concatenate $\mathbf{h}_i$ with a one-hot entity tag representation $\mathbf{v}_i$ of $\hat{t}_i$. We build a feature vector $\mathbf{f}_{e_1}$ for e_1 by running a CNN (a single convolution layer with a max-pooling layer) on vectors $\{\mathbf{h}_i \oplus \mathbf{v}_i | w_i \in e_1\}$. Similarly, we build $\mathbf{f}_{e_2}$ for e_2 with another CNN.

For context features of the entity pair (e_1, e_2), we build three feature vectors by looking at words between e_1 and e_2 ($\mathbf{f}_{\text{middle}}$), words on the left of the pair ($\mathbf{f}_{\text{left}}$) and words on the right of the pair ($\mathbf{f}_{\text{right}}$). For $\mathbf{f}_{\text{middle}}$, we run a CNN on words between e_1 and e_2 like the case of $\mathbf{f}_{e_1}, \mathbf{f}_{e_2}$. For $\mathbf{f}_{\text{left}}$ and $\mathbf{f}_{\text{right}}$, we use the "LSTM-Minus" method as (Wang and Chang, 2016; Zhang et al., 2017). Assume that the left context of (e_1, e_2) is from sentence position 0 to i, then $\mathbf{f}_{\text{left}} = \overrightarrow{\mathbf{h}}_i \oplus (\overleftarrow{\mathbf{h}}_0 - \overleftarrow{\mathbf{h}}_{i+1})$. Similarly, if the right context of (e_1, e_2) is from j to $|s| - 1$, then $\mathbf{f}_{\text{right}} = (\overrightarrow{\mathbf{h}}_{|s|-1} - \overrightarrow{\mathbf{h}}_{j-1}) \oplus \overleftarrow{\mathbf{h}}_j$. We also use a one-hot feature $\mathbf{f}_{\text{dist}}$ to describe the distance between e_1 and e_2 in the sentence.

Finally, $\mathbf{f}_{e_1}$, $\mathbf{f}_{e_2}$, $\mathbf{f}_{\text{middle}}$, $\mathbf{f}_{\text{left}}$, $\mathbf{f}_{\text{right}}$ and $\mathbf{f}_{\text{dist}}$ are concatenated to a single vector $\mathbf{f}_{e_1,e_2}$. To get the posterior of the relation type $\hat{l}$, we apply a multi-layer perceptron with one hidden layer on $\mathbf{f}_{e_1,e_2}$,

$$\begin{aligned} &\mathrm{P}_{\text{rel}}(\hat{l} | s, e_1, e_2; \boldsymbol{\theta}_R) \\ &= \mathrm{Softmax}(\mathbf{W}_2 \cdot \mathrm{ReLU}(\mathbf{W}_1 \cdot \mathbf{f}_{e_1,e_2})), \quad (2) \end{aligned}$$

where $\boldsymbol{\theta}_R = \{\boldsymbol{\theta}_{e_1}, \boldsymbol{\theta}_{e_2}, \boldsymbol{\theta}_{\text{middle}}, \mathbf{W}_1, \mathbf{W}_2\}$ contains parameters of the relation model (shared parameters with the entity model are omitted).

Given an input sentence s, the training objective is to minimize

$$\mathcal{L}_{\text{rel}}(\boldsymbol{\theta}_R) = - \sum_{\substack{e_1, e_2 \in \hat{\mathcal{E}} \\ e_1 \neq e_2}} \frac{\log P_{\text{rel}}(\hat{l} = l | s, e_1, e_2; \boldsymbol{\theta}_R)}{|\hat{\mathcal{E}}|(|\hat{\mathcal{E}}| - 1)},$$

where the true label l of a candidate entity pair (e_1, e_2) can be read from true annotations.

3.3 Joint Minimum Risk Training

To jointly learn the entity model and the relation model, one common strategy is to optimize the combined objective function $\widetilde{\mathcal{L}} = \mathcal{L}_{\text{ent}} + \mathcal{L}_{\text{rel}}$, where the joint learning is accomplished by the shared parameters. However, we would think that $\widetilde{L}$ optimizes a "local" loss by observing that a) in both $\mathcal{L}_{\text{ent}}$ and $\mathcal{L}_{\text{rel}}$, the loss functions are calculated by only looking at local parts. For example, the loss in $\mathcal{L}_{\text{ent}}$ is based on the correctness of local entity tags t_i rather than a global measurement (e.g., F1 score of extracted entities), b) both the entity model and the relation model are unaware of the loss from the other side. For example, the entity model needs to wait for the relation model to update the shared parameters rather than get direct supervision from the loss of the relation model.

Here we introduce the minimum risk training framework to the joint model. Comparing with optimizing the local loss in $\widetilde{\mathcal{L}}$, the joint MRT will optimize a global loss and provide a tighter connection between the entity decoder and the relation decoder. To illustrate the algorithm, we first aggregate some notations.

Let $\mathbf{y} \triangleq (\mathcal{E}, \mathcal{R})$ contain the ground truth entity tag sequence and relations, $\hat{\mathbf{y}} \triangleq (\hat{\mathcal{E}}, \hat{\mathcal{R}})$ contain outputs of the joint extraction model and $\mathcal{Y}(s)$ be the set of all possible outputs of the input sentence s ($\mathbf{y}, \hat{\mathbf{y}} \in \mathcal{Y}(s)$). We define the joint probability,

$$
\begin{aligned}
&P(\hat{\mathbf{y}}|s;\boldsymbol{\theta}) = P(\hat{\mathcal{E}}|s;\boldsymbol{\theta}_E)P(\hat{\mathcal{R}}|s,\hat{\mathcal{E}};\boldsymbol{\theta}_R)\\
&= \prod_i P_{\text{ent}}(\hat{t}_i|s;\boldsymbol{\theta}_E) \prod_{\substack{e_1,e_2\in\hat{\mathcal{E}}\\ e_1\neq e_2}} P_{\text{rel}}(\hat{l}|s,e_1,e_2;\boldsymbol{\theta}_R),
\end{aligned}
$$

where $\boldsymbol{\theta} = \boldsymbol{\theta}_E \bigcup \boldsymbol{\theta}_R$ is the joint model parameter, and $P_{\text{ent}}, P_{\text{rel}}$ are in Equation 1 and 2.

The objective of MRT is to minimize the following expected loss (i.e., *risk*),

$$
\mathbf{E}_{\hat{\mathbf{y}}\sim P(\hat{\mathbf{y}}|s;\boldsymbol{\theta})}\Delta(\hat{\mathbf{y}},\mathbf{y}) = \sum_{\hat{\mathbf{y}}\in\mathcal{Y}(s)} P(\hat{\mathbf{y}}|s;\boldsymbol{\theta})\Delta(\hat{\mathbf{y}},\mathbf{y}), \tag{3}
$$

where $\Delta(\hat{\mathbf{y}}, \mathbf{y})$ is a (arbitrary) loss function describing the difference between $\hat{\mathbf{y}}$ and $\mathbf{y}$.

In our model, the loss function $\Delta(\hat{\mathbf{y}}, \mathbf{y})$ is the key factor to enhance the joint extraction performances. First, in $\Delta(\hat{\mathbf{y}}, \mathbf{y})$, we consider sentence-level F1 scores of entity and relation extraction results (denoted by $\mathrm{F}_{\text{ent}}(\hat{\mathcal{E}}, \mathcal{E})$, $\mathrm{F}_{\text{rel}}(\hat{\mathcal{R}}, \mathcal{R})$). Specifically, we use $1 - \mathrm{F}_{\text{ent}}(\hat{\mathcal{E}}, \mathcal{E})$ and $1 - \mathrm{F}_{\text{rel}}(\hat{\mathcal{R}}, \mathcal{R})$ as the metric of the entity loss and the relation loss respectively. On the one hand, F1 scores characterize the overall performance of the outputs and make the training objective be consistent with the testing time evaluation metric. On the other hand, F1 scores cannot be decomposed onto local predictions of $\hat{\mathcal{E}}$ and $\hat{\mathcal{R}}$ like the log losses in $\mathcal{L}_{\text{ent}}$ and $\mathcal{L}_{\text{rel}}$, thus we need a different training algorithm.

Second, different from previous applications of MRT on single tasks (Xu et al., 2016; Shen et al., 2016), we have two sources of losses in the joint extraction. By integrating losses of individual tasks in the learning algorithm, the entity model could forecast how plausible a candidate entity is according to the relation model, and the relation model could also know the confidence of the entity extraction results. Here, we define a global loss by adding losses of the two models,

$$
\Delta_{E+R}(\hat{\mathbf{y}}, \mathbf{y}) = 1 - \frac{1}{2}[\mathrm{F}_{\text{ent}}(\hat{\mathcal{E}}, \mathcal{E}) + \mathrm{F}_{\text{rel}}(\hat{\mathcal{R}}, \mathcal{R})].
$$

Algorithm 1 The Sampling Algorithm

```
Input: Entity model $\boldsymbol{\theta}_E$, relation model $\boldsymbol{\theta}_R$, sentence $s$, the
    sample size $K$
Output: A subset $\mathcal{Y}'(s)$ of $\mathcal{Y}(s)$
$\mathcal{Y}'(s) \leftarrow \{(\mathcal{E},\mathcal{R})\}$ // add the ground truth
while $|\mathcal{Y}'(s)| \leq K$ do
    $i \leftarrow 1$
    while $i \leq |s|$ do
        with prob. $0.9$, sample $t'_i \sim P_{\text{ent}}(\cdot|s;\boldsymbol{\theta}_E)$
        with prob. $0.1$, sample $t'_i$ uniformly
        $i \leftarrow i + 1$
    end while
    $\mathcal{E}' \leftarrow \mathbf{t}' = t'_1, t'_2, \cdots, t'_{|s|}$
    $\mathcal{R}' \leftarrow \emptyset$
    for $e_1, e_2 \in \hat{\mathcal{E}}, e_1 \neq e_2$ do
        sample $l' \sim P_{\text{rel}}(\cdot|s, e_1, e_2; \boldsymbol{\theta}_R)$
        $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{(e_1, e_2, l')\}$
    end for
    $\mathcal{Y}'(s) \leftarrow \mathcal{Y}'(s) \cup \{(\mathcal{E}', \mathcal{R}')\}$
end while
```

To compare with Δ_{E+R}, we also try two alternatives of $\Delta(\hat{\mathbf{y}}, \mathbf{y})$ in experiments, namely, $\Delta_E(\hat{\mathbf{y}}, \mathbf{y}) = 1 - \mathrm{F}_{\text{ent}}(\hat{\mathcal{E}}, \mathcal{E})$ and $\Delta_R(\hat{\mathbf{y}}, \mathbf{y}) = 1 - \mathrm{F}_{\text{rel}}(\hat{\mathcal{R}}, \mathcal{R})$. They only look one model's loss.

Third, in addition to handcrafted loss functions, we further ask whether the joint MRT model could benefit from automatic "loss engineering". Specifically, let $\Gamma(\hat{\mathbf{y}})$ be the loss learned from the training set, we augment $\Delta(\hat{\mathbf{y}}, \mathbf{y})$ of the MRT objective with $\Gamma(\hat{\mathbf{y}})$, and require the learning process to assign a smaller Γ value (with a margin) to the ground truth output $\mathbf{y}$ than other $\hat{\mathbf{y}} \in Y\backslash\{\mathbf{y}\}$,

$$
\begin{aligned}
&\min. \sum_{\hat{\mathbf{y}}\in\mathcal{Y}(s)} P(\hat{\mathbf{y}}|s;\boldsymbol{\theta}) \left(\Delta(\hat{\mathbf{y}},\mathbf{y}) + \Gamma(\hat{\mathbf{y}})\right) + \xi\\
&\text{s.t.} \quad \Gamma(\mathbf{y}^*) - \Gamma(\mathbf{y}) \geq 1 - \xi,\ \xi \geq 0,
\end{aligned} \tag{4}
$$

where $\mathbf{y}^* = \arg\min_{\hat{y}\in Y(s)} \Gamma(\hat{\mathbf{y}})$. Here, we simply set $\Gamma(\hat{\mathbf{y}}) = 1 - P(\hat{\mathbf{y}}|s;\boldsymbol{\theta})$ [4] and reformulate above objective as

$$
\begin{aligned}
&\sum_{\hat{\mathbf{y}}\in\mathcal{Y}(s)} P(\hat{\mathbf{y}}|s;\boldsymbol{\theta}) \left(\Delta(\hat{\mathbf{y}},\mathbf{y}) - P(\hat{\mathbf{y}}|s;\boldsymbol{\theta})\right)\\
&\qquad + [1 - P(\mathbf{y}|s;\boldsymbol{\theta}) + P(\mathbf{y}^*|s;\boldsymbol{\theta})]_+ .
\end{aligned} \tag{5}
$$

where $[u]_+ = \max(u, 0)$ is the hinge loss.

Optimizing the expected loss is hard since the size of $\mathcal{Y}(s)$ is exponential. In practice, we could

---

[4] Further study on different $\Gamma(\hat{\mathbf{y}})$ is left for future work.

approximate the expectation in Equation 3 by sampling a tractable subset $\mathcal{Y}'(s)$ of $\mathcal{Y}(s)$. Specifically, we first obtain an entity set $\mathcal{E}'$ by sampling (without replacement) an entity tag sequence $\mathbf{t}'$ from P_{ent}. [5] Then based on the sampled entities, we get a relation set $\mathcal{R}'$ by sampling l' from P_{rel} for each entity pairs. Algorithm 1 lists the pseudo code. [6] In experiments, we also try a variant of Algorithm 1 which only samples from the entity model, and selects relation labels with the maximum posterior (i.e., doesn't sample relations).

With the sampled subset $\mathcal{Y}'(s)$, we consider a revised version of the original MRT objective,

$$\mathcal{L}_{\text{mrt}}(\boldsymbol{\theta}) = \sum_{\hat{\mathbf{y}}\in\mathcal{Y}'(s)} Q(\hat{\mathbf{y}}|s;\boldsymbol{\theta},\mu,\alpha)\Delta(\hat{\mathbf{y}},\mathbf{y}), \quad (6)$$

where $Q(\hat{\mathbf{y}}|s;\boldsymbol{\theta},\mu,\alpha)$ is a re-normalization of $P(\hat{\mathbf{y}}|s;\boldsymbol{\theta})$ on the subset $\mathcal{Y}'(s)$, [7]

$$Q(\hat{\mathbf{y}}|s;\boldsymbol{\theta},\mu,\alpha) = \frac{1}{Z}[P(\hat{\mathcal{E}}|s,\boldsymbol{\theta}_E)^{\mu}P(\hat{\mathcal{R}}|s,\hat{\mathcal{E}},\boldsymbol{\theta}_R)^{1-\mu}]^{\alpha}$$

$$Z = \sum_{(\mathcal{E}',\mathcal{R}')\in\mathcal{Y}'(s)} [P(\mathcal{E}'|s,\boldsymbol{\theta}_E)^{\mu}P(\mathcal{R}'|s,\mathcal{E}',\boldsymbol{\theta}_R)^{1-\mu}]^{\alpha}$$

The hyper-parameter α controls the sharpness of the Q distribution (Och, 2003), and μ weights the importance of the entity model and the relation model in Q. Similary, we can rewrite the objective in Equation 5 with $\mathcal{Y}'(s)$ and Q.

Finally, we remark that if we view MRT as a fine tuning step, it can be applied in any joint learning model based on building the joint distribution $P(\hat{\mathbf{y}}|s,\boldsymbol{\theta})$ (e.g., the globally normalized P in (Zhang et al., 2017)). Thus, we would think that MRT is a flexible and lightweight framework for the joint learning.

3.4 Training

To train the joint extraction model, we first pretrain the model with objective $\widetilde{\mathcal{L}}$ (i.e., minimize the local loss), then optimize the local loss and the global loss simultaneously with objective $\widetilde{\mathcal{L}} + \mathcal{L}_{mrt}$. The setting is slightly different from previous work which only optimize $\mathcal{L}_{\text{mrt}}$ in the second step. We find that adding $\widetilde{\mathcal{L}}$ in the experiments could make the training more stable.

When training with $\widetilde{\mathcal{L}}$ in the pre-training step, we apply the scheduled sampling strategy (Bengio et al., 2015) in the entity model as (Miwa and Bansal, 2016). Models are regularized with dropout and trained using Adadelta (Zeiler, 2012). We give the full derivation of Equation 6's gradient in the supplementary. [8]

We select models using development sets: within a fix number of epochs, the model with the best relation extraction performance on the development set is picked out for testing. [9]

4 Experiments

We evaluate the proposed model on two datasets. **ACE05** is a standard corpus for the entity relation extraction task. It is labelled with 7 entity types and 6 relation types. We use the same split of ACE05 documents as previous work (351 training, 80 development, and 80 testing). [10] **NYT** (Riedel et al., 2010) is a larger corpus which is labelled with 3 entity types and 24 relation types. [11] The training set has 353k relation triples which are generated by distant supervision. It also provides another 3880 manually labelled relation triples. Following (Ren et al., 2017; Zheng et al., 2017), we exclude the `None` relation label and randomly select 10% of the labelled data as the development set. We will mainly discuss the results on ACE05 where many previous joint learning models are available for comparison.

We list detailed hyper-parameter settings in the supplementary. Note that, except μ, α, K which are introduced in the joint MRT and selected on the development set, [12] we don't tune hyper-parameters extensively. For example, we use the same setting in both ACE 05 and NYT rather than tune parameters on each of them.

As previous work, we evaluate performances

---

[5] To accelerate sampling, we borrow the idea of ε-greedy in reinforcement learning: with probability 0.9, we sample t_i' from P_{ent}, and with probability 0.1, we sample it uniformly.

[6] The time complexity is $O(K|s|)$ which is the same to the beam search algorithm with beam size K (Zhang et al., 2017).

[7] Here we follow the literature of MRT to apply the renormalization on $\mathcal{Y}'(s)$. Another formulation is the policy gradient framework which sticks to the original probability.

[8] We remark that the MRT objective (Equation 6) is differentiable with respect to model parameters (Shen et al., 2016). The non-decomposability of the F1 score does not make the model non-differentiable. In our implementation, the gradient is automatically calculated using autograd tools. Please see the supplementary for more details.

[9] We focus on the performance of the ent-to-end relation extraction, so we select models by the relation extraction results. It is also possible to consider both the performances of the entity model and the relation model. We leave the study of advanced model selection algorithms for future work.

[10] We use the dataset in https://github.com/tticoin/LSTM-ER, which is from (Miwa and Bansal, 2016).

[11] https://github.com/shanzhenren/CoType.

[12] The default setting is $\alpha = 10^{-4}, \mu = 1.0, K = 3$ in systems without self-learned Γ loss and $\alpha = 1, \mu = 1.0, K = 2$ in systems with Γ loss.

Model	Entity			Relation		
	P	R	F	P	R	F
L&J (2014)	**85.2**	76.9	80.8	**65.4**	39.8	49.5
M&B (2016)	82.9	**83.9**	83.4	57.2	54.0	55.6
Zhang (2017)	-	-	83.5	-	-	57.5
K&C (2017)	84.0	81.3	82.6	55.5	51.8	53.6
NN	84.0	82.9	83.4	59.5	**56.3**	57.8
MRT	83.9	83.2	**83.6**	64.9	55.1	**59.6**

Table 1: Results on the ACE05 test data. (Miwa and Bansal, 2016) and (Katiyar and Cardie, 2017) are joint training systems without joint decoding. (Li and Ji, 2014) and (Zhang et al., 2017) are joint decoding algorithms. NN is our neural network model without minimum risk training. MRT is minimum risk training with loss Γ (Equation 5). We omit pipeline methods which underperform joint models (see (Li and Ji, 2014) for details).

Settings		F1 of Entity	F1 of Relation
Default sampling	Δ_E	**83.8** $_{+0.4}$	57.9 $_{+0.1}$
	Δ_R	83.5 $_{+0.1}$	58.9 $_{+1.1}$
	Δ_{E+R}	83.6 $_{+0.2}$	59.0 $_{+1.2}$
	Γ	83.6 $_{+0.2}$	58.3 $_{+0.5}$
	$\Gamma + \Delta_{E+R}$	83.6 $_{+0.2}$	**59.6** $_{+1.8}$
Only sampling entity	Δ_E	**83.7** $_{+0.3}$	57.4 $_{-0.4}$
	Δ_R	83.5 $_{+0.1}$	59.1 $_{+1.3}$
	Δ_{E+R}	83.6 $_{+0.2}$	57.9 $_{+0.1}$
	Γ	83.6 $_{+0.2}$	58.7 $_{+0.9}$
	$\Gamma + \Delta_{E+R}$	83.3 $_{-0.1}$	**59.2** $_{+1.4}$

Table 2: MRT with different loss functions and sampling methods. The numbers in subscripts indicate improvements over the NN setting in Table 1.

using precision (P), recall (R) and F1 scores. Specifically, an output entity e is correct if its type and the region of its head are correct, and an output relation r is correct if its e_1, e_2, l are correct (i.e., "exact match").

4.1 Results on ACE05

We first compare proposed models with previous work (Table 1). In general, our plain neural network model (NN) is competitive, and after compiling with MRT, it achieves non-negligible improvement over existing state-of-the-art systems. (both on the entity and the relation extraction). [13] We have following two detailed comparisons.

Among systems which only rely on shared parameters ((Miwa and Bansal, 2016; Katiyar and Cardie, 2017) and NN), NN gives the best result (we give detailed results on different relation types in the supplement). One possible reason is that the "RNN+CNN" network structure is not fully explored in previous joint learning models. More importantly, it suggests that how to build powerful sub-models and utilize shared parameters are still among the key problems of the task.

Comparing with the best joint decoding system which adopts global normalization in training (Zhang et al., 2017), MRT mainly improves the relation extraction results. We think that the improvement may come from the sentence-level loss applied in MRT: both systems consider interactions between decoders, and both objectives are approximated by sampling, but MRT optimizes F1 score while Zhang et al. (2017) optimize label accuracy. For the joint decoding system in (Li and Ji, 2014), although it cannot beat recent neural network-based models, it is interesting to compare MRT with a feature-enriched version of (Li and Ji, 2014)'s model in the future work.

Next, we evaluate the joint MRT with different loss functions and sampling methods.

As mentioned in Section 3.3, we have three options (Δ_{E+R}, Δ_E, Δ_R) for $\Delta(\hat{\mathbf{y}}, \mathbf{y})$ and a self-learned loss function Γ. The first five rows of Table 2 show their performances on the test data. We have three observations regarding the results.

1. Δ_R, Δ_{E+R} have higher relation F1 scores than Δ_E and NN. Thus, adding relation loss in $\Delta(\hat{\mathbf{y}}, \mathbf{y})$ is helpful for relation extraction. We think that knowing the relation loss could bias the entity model to highlight the entities appearing in relations, which provides a better candidate relation set for the relation extraction model.

2. Δ_E has the best entity extraction results, which implies that the sentence-level entity loss alone could benefit entity extraction. While after adding relation loss (Δ_{E+R}), the entity performance slightly decreases. One reason might be that our model selection strategy only focuses on the relation part (footnote 9), thus the model with improved entity performances may not be selected.

3. The learned loss Γ can help to improve performances, but only using Γ is not as effective as the handcrafted Δ functions (which are tailored to the evaluation metrics). By combining both the prior knowledge and information from the dataset, $\Gamma + \Delta_{E+R}$ achieves the best results.

Regarding the sampling method, we test a variant of Algorithm 1 which samples entities but not

---

[13] It is worth noting that our models don't access additional linguistic resources such as POS tags and dependency trees. We have tried to add syntactic features in (Zhang et al., 2017), but didn't observe improvements.

relations (the last five rows of Table 2). Comparing with the default sampling algorithm, it has similar entity extraction performances, but its behaviour on the relation extraction is different. Specifically, adding entity loss in $\Delta(\hat{\mathbf{y}}, \mathbf{y})$ (i.e., Δ_E, Δ_{E+R}) now affects relation results negatively. It may suggest that when only exploring the output of entity extraction, the entity loss may dominate the relation loss, and trap the joint model to exploit the entity model only. On the other hand, the performances of self-learned loss Γ are less sensitive to the sampling method. We haven't had a clear understanding of the relationship between sampling algorithms and loss functions, but the above results show that adding data-related loss function could improve the robustness of MRT in practice.

Thirdly, we present influences of hyper-parameters for MRT with Δ_{E+R} on the development set in Figure 3 and 4 (other settings have similar results). We find that, for the parameters examined here, it is hard for the entity model and the relation model to agree with each other: parameters achieving high relation performances usually get low entity performances, and vise versa. Thus, if we perform the model selection by only looking at relation extraction results, the joint model may sacrifice entity extraction performances. For α and μ (Figure 3), we observe that on the ACE05 dataset, the model prefers a small α (which means a sharper Q) and μ at boundary (i.e., Q is either close to the entity model or the relation model). Regarding the sample size K (Figure 4), we don't observe a convergence of performances in a small range of K. Since the computation cost increases rapidly as we increase the sample size ($K = 5$ is about 2x slower than $K = 3$ in our implementation), we stick to a small K.

Finally, due to lack of space, we provide more discussions on model configurations (including results regarding different entity pair distances, additional experiments on tuning hyper parameters etc.), and detailed error analyses on concrete samples in the supplementary.

4.2 Results on NYT

We briefly list results on the NYT dataset in Table 3. The baseline methods are (Ren et al., 2017) which is based on a joint embedding of entities and relations, and (Zheng et al., 2017) which conducts joint decoding with an augmented sequence labelling tag set. Both NN and MRT outperform baseline results. In particular, comparing with the joint tagging scheme in (Zheng et al., 2017), MRT adds no constraint on the relation extraction model and can explore the large NYT training set more effectively. At the same time, since the training set is automatically generated, the global losses observed in MRT are also noisy. Like recent work on bandit structured prediction (Kreutzer et al., 2017; Nguyen et al., 2017), the results here suggest that MRT could be a reasonable choice when the supervision of the joint learning is partial and noisy.

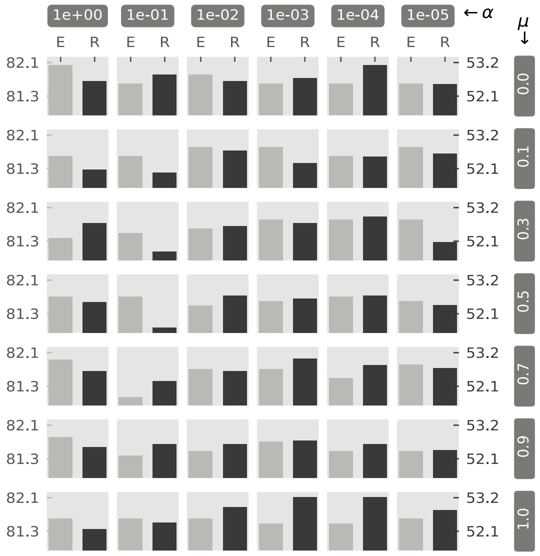

Figure 3: MRT with different Q distributions on the development set. Rows are settings of μ, and columns are settings of α. In each cell, we draw F1 scores of the entity extraction (the left gray bar) and the relation extraction (the right dark bar) under the combination of corresponding α and μ.

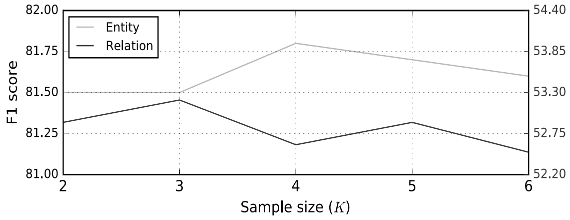

Figure 4: MRT with different sample size K on the development set.

5 Conclusion

We introduced minimum risk training to the task of joint entity and relation extraction. We showed that, with a global loss function, MRT could enhance the connection between the sub-models. Extensive experiments on benchmark datasets witness the effectiveness of the joint MRT.

Model	Relation		
	P	R	F
(Zheng et al., 2017)	61.5	41.4	49.5
NN	61.8	**43.3**	50.9
MRT	**67.4**	42.0	**51.7**
(Ren et al., 2017)	42.3	**51.1**	46.3
NN (exact match)	59.4	41.7	49.0
MRT (exact match)	**65.2**	40.6	**50.0**

Table 3: Results on the NYT dataset. To compare with (Ren et al., 2017), we give results under the "exact match" criterion as ACE05. To compare with (Zheng et al., 2017), we give results which ignore the entity type in the justification of relations. We use $\alpha = 1, \mu = 1, K = 2$ and $\Delta_{E+R} + \Gamma$.

Acknowledgement

The authors wish to thank the reviewers for their helpful comments and suggestions. This research is (partially) supported by STCSM (18ZR1411500), NSFC(61673179), Shanghai Knowledge Service Platform Project (ZF1213), and Shanghai Key Laboratory of Trustworthy Computing (07dz22304201604). The corresponding authors are Yuanbin Wu, Man Lan and Shiliang Sun.

References

Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*.

Shiqi Shen Ayana, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with minimum risk training. *arXiv preprint arXiv:1604.01904*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 551–560, Portland, Oregon, USA. Association for Computational Linguistics.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin crfs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736, Los Angeles, California. Association for Computational Linguistics.

Xiaodong He and Li Deng. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 292–301, Jeju Island, Korea. Association for Computational Linguistics.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 919–929, Berlin, Germany. Association for Computational Linguistics.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 917–928, Vancouver, Canada. Association for Computational Linguistics.

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. 2017. Bandit structured prediction for neural sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1503–1513, Vancouver, Canada. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland. Association for Computational Linguistics.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 121–130, Singapore. Association for Computational Linguistics.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. 2017. Reinforcement learning for bandit neural machine translation with simulated human feedback. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1464–1474, Copenhagen, Denmark. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan. Association for Computational Linguistics.

Nathan D. Ratliff, J. Andrew Bagnell, and Martin Zinkevich. 2006. Maximum margin planning. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 729–736.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1015–1024. International World Wide Web Conferences Steering Committee.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III*, pages 148–163.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 787–794, Sydney, Australia. Association for Computational Linguistics.

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning*, 1st edition. MIT Press, Cambridge, MA, USA.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315, Berlin, Germany. Association for Computational Linguistics.

Wenduan Xu, Michael Auli, and Stephen Clark. 2016. Expected f-measure training for shift-reduce parsing with recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 210–220, San Diego, California. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1731–1741, Copenhagen, Denmark. Association for Computational Linguistics.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, Vancouver, Canada. Association for Computational Linguistics.

Large-scale Exploration of Neural Relation Classification Architectures

Hoang-Quynh Le[1], **Duy-Cat Can**[1†], **Sinh T. Vu**[1†],
Thanh Hai Dang[1*], **Mohammad Taher Pilehvar**[2] **and Nigel Collier**[2]

[1]Faculty of Information Technology, VNU University of Engineering and Technology,
Hanoi, Vietnam
[2]Department of Theoretical and Applied Linguistics, University of Cambridge, UK
`{lhquynh, catcd, sinhvt, hai.dang}@vnu.edu.vn`
`{mp792, nhc30}@cam.ac.uk`

Abstract

Experimental performance on the task of relation classification has generally improved using deep neural network architectures. One major drawback of reported studies is that individual models have been evaluated on a very narrow range of datasets, raising questions about the adaptability of the architectures, while making comparisons between approaches difficult. In this work, we present a systematic large-scale analysis of neural relation classification architectures on six benchmark datasets with widely varying characteristics. We propose a novel multi-channel LSTM model combined with a CNN that takes advantage of all currently popular linguistic and architectural features. Our 'Man for All Seasons' approach achieves state-of-the-art performance on two datasets. More importantly, in our view, the model allowed us to obtain direct insights into the continued challenges faced by neural language models on this task. Example data and source code are available at: `https://github.com/aidantee/MASS`.

1 Introduction

Determining the semantic relation between pairs of named entity mentions, *i.e.* relation classification, is useful in many fact extraction applications, ranging from identifying adverse drug reactions (Gurulingappa et al., 2012; Dandala et al., 2017), extracting drug abuse events (Jenhani et al., 2016), improving the access to scientific literature (Gábor et al., 2018), question answering (Lukovnikov et al., 2017; Das et al., 2017) to major life events extraction (Li et al., 2014; Cavalin et al., 2016). With a multitude of possible relation types, it is critical to understand how systems will behave in a variety of settings (see Table 1 for an example).

(i)	*<e1>**Three-dimensional digital subtraction angiographic**</e1> (<e2>**3D-DSA**</e2>) images from diagnostic cerebral angiography were obtained ...*
(ii)	*The metal <e1>**ball**</e1> makes a ding ding ding <e2>**noise**</e2> when it swings back and hits the metal body of the lamp.*

Table 1: Examples for different relation types: sentence (i) shows a *Synonym-of* relation, represented by an abbreviation pattern, which is very different from the predicate relation *Cause-effect* in (ii).

To the best of our knowledge, almost all relation classification models introduced so far have been experimentally validated on only a few datasets - often only one. This is despite the availability of established benchmarks. The lack of transparency as well as the possibility of having selection bias raise a question about the true capability of state-of-the-art methods for relation classification. In addition, despite such a wealth of studies, it still remains unclear which approach is superior and which factors set the limits on performance. For example, heuristic post-processing rules have been seen to significantly boost relation classification performance on several benchmarks; yet, they cannot be relied upon to generalize across domains. The novel approach we present in this paper draws inspiration from neural hybrid models such as that of Cai et al. (2016). In this work, we present a large-scale analysis of state-of-the-art neural network architectures on six benchmark datasets which represent a variety of language domains and semantic types. As a means of comparison against reported system performance, we propose a novel multi-channel long short term memory (Hochreiter and Schmidhuber, 1997, LSTM) model combined with a Convolutional Neural Network (Kim, 2014, CNN) that takes advantage of all major linguistic and architectural features cur-

---

[†] Contributed equally & Names are in alphabetical order
[*] Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2266–2277
Brussels, Belgium, October 31 - November 4, 2018.

rently employed. We designate this as a 'Man for All SeasonS' (MASS) model because it incorporates many popular elements reported by state of the art systems on individual datasets.

The main contributions of the paper are:

1. We presented a deep neural network model, in which each component is capable of taking advantage of a particular type of major linguistic or architectural feature. The model is robust and adaptable across different relation types in various domains without any architectural changes.

2. We investigated the impact of different components and features on the final performance, therefore, providing insights on which model components and features are useful for future research.

2 Related Works

We focus here on supervised approaches to relation classification. Alternatives include hand built patterns (Aone and Ramos-Santacruz, 2000), unsupervised approaches (Yan et al., 2009) and distantly supervised approaches (Mintz et al., 2009). Traditional supervised and kernel-based approaches have made use of a full range of linguistic features (Miwa et al., 2010) such as orthography, character n-grams, chunking as well as vertex and edge walks over the dependency graph. Hand crafting and modeling with such complex feature sets remains a challenge although performance tends to increase with the amount of syntactic information (Bunescu and Mooney, 2005).

Recent successes in deep learning have stimulated interest in applying neural architectures to the task. Convolutional Neural Networks (CNNs) (Nguyen and Grishman, 2015) were among early approaches to be applied. Following in this direction, (Lee et al., 2017) achieved state of the art performance on the ScienceIE task of SemEval 2017. Other recent variations of CNN architectures include a CNN with an attention mechanism in Shen and Huang (2016) and a CNN combined with maximum entropy in Gu et al. (2017). Various auxiliary information has been reported to improve the performance of CNNs, such as the document graph (Verga et al., 2018) and position embeddings (Shen and Huang, 2016; Lee et al., 2017; Verga et al., 2018). Recurrent Neural Networks (RNNs) are another approach to capturing relations and naturally good at modeling long distance relations within sequential language data. Approaches include Mehryary et al. (2016) with the original RNN and Li et al. (2017); Ammar et al. (2017); Zhou et al. (2018) with RNNs having LSTM units which are used to extend the range of context. Apart from sentences themselves, RNN-based models often take as input information extracted from dependency trees, such as shortest dependency paths (SDP) (Mehryary et al., 2016; Ammar et al., 2017), or even whole trees (Li et al., 2017). Since RNNs and CNNs each have their own distinct advantages, a few models have combined both in a single neural architecture (Cai et al., 2016; Zhang et al., 2018).

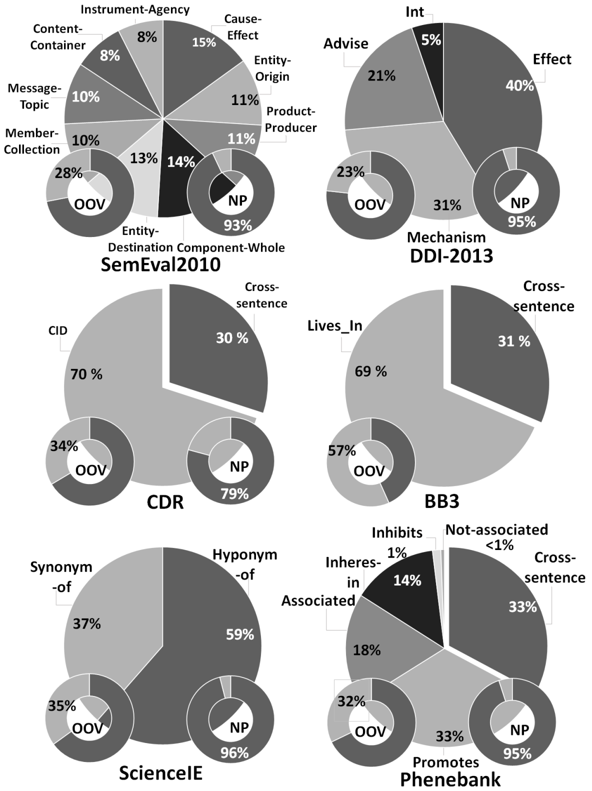

Figure 1: The statistics of corpora used in our experiments. Three aspects are considered: the distribution of relation types, the distribution of Out-Of-Vocabulary (OOV) in the test set and the distribution of new entity pairs (NP) that appeared in the test set but never appeared in the training data.

3 Materials and Methods

3.1 Gold Standard Corpora

As noted above, our experiments used six well-known benchmark corpora from different domains, which have been used to evaluate vari-

#	Corpus	Domain	IAA	Size	Entity	Relation	% of negatives	Cross-sentence	Directed	Undirected	SDP length
1	SemEval *(SemEval 2010 - Task 8)*	Generic	0.74	8000 (2717)	–	9	17.4 %	–	✓	–	3.8 (13)
2	DDI-2013 *(SemEval DDIExtraction 2013)*	Biomedical	D: 0.84 M: 0.62	730 (175)	4	4	85.3 %	–	–	✓	9.0 (66)
3	CDR *(BioCreative5 CDR 2015)*	Biomedical	-	1000 (500)	2	1	61.4 %	✓	✓	–	6.8 (24)
4	BB3 *(BioNLPST BB-Event 2016)*	Biomedical	0.47	95 (51)	3	1	61.4 %	✓	✓	–	7.5 (25)
5	ScienceIE *(SemEval ScienceIE 2017)*	Scientific	0.45-0.85	400 (100)	3	2	88.5 %	–	✓	✓	6.5 (22)
6	Phenebank	Biomedical	0.56	1000 (500)	9	5	77.0 %	✓	✓	✓	6.2 (26)

Table 2: Characteristics of the six corpora used in this study. Domain: the domain of the corpus; IAA: the Inter-annotator Agreement score; Size: training set size (test set size in the brackets) in terms of the number of sentences (SemEval) or documents (all other corpora); Entity: the number of entity types; Relation: the number of relation types; % of negative: the distribution of positive and negative instances; Cross-sentence: if there are cross-sentence relations; Directed: if there are directed relations in the corpus; Undirected: if there are undirected relations in the corpus; SDP length: the averaged (max in brackets) length of the SDPs in the corpus.

ous state-of-the-art relation classification systems. ***SemEval*** is a generic domain benchmark dataset (Hendrickx et al., 2009). The next four chosen corpora are from various biomedical domains: the ***DDI-2013*** corpus (Herrero-Zazo et al., 2013; Segura-Bedmar et al., 2014), the ***CDR*** corpus (Li et al., 2016), the ***BB3*** corpus (Delėger et al., 2016), and the ***Phenebank*** corpus. Finally, ***ScienceIE*** corpus contains scientific journal articles from three sub-domains (Augenstein et al., 2017). Inter-annotator agreement (IAA) as measured with Cohens kappa on these corpora indicates high variability in the range of $[0.45, 0.74]$, *i.e. moderate* to *substantial* agreement (McHugh, 2012).

As shown in Table 2, each of these corpora is distinct in many respects. CDR and BB3 were only annotated with one relation type, whilst other corpora have several relation types. In all corpora except SemEval, negative instances must be automatically generated by pairing all the entities appearing in the same sentences that have not been annotated as positives. As there are a large number of such entities, the number of possible negatives accounts for a large percentage of set of instances, *i.e.* up to 80% of the total in DDI-2013, ScienceIE and Phenebank. Further, the small percentage of positive examples includes several types, causing a severe imbalance in the data (He and Garcia, 2009) (see Figure 1 for further details).

Another challenge for relation classification is in modeling the order of entities in a directed relation type (Lee et al., 2017). In the six corpora, several relations are directed and order-sensitive, such as the *Cause-Effect* relation in SemEval and *Hyponym-of* in ScienceIE. Such relations require the model to predict both relation types and the entity order correctly. In contrast, for undirected relations, such as *Synonym-of* in ScienceIE and *Associated* in Phenebank, both directions can be accepted.

An interesting factor is that the length of the SDP in SemEval is considerably shorter than in the other corpora. The mean and maximum length SDP values for CDR, BB3, ScienceIE and Phenebank are quite similar, *i.e.* ~ 7 and $22 - 26$ tokens. DDI-2013 contains very complex sentences, with an averaged SDP length of 9 and the longest SDP of 66 token.

Figure 1 shows the Out-Of-Vocabulary (OOV) ratios in six corpora, which are quite large, ranging from 23% to 57%. More interesting is the percentage of entities (or nominal) pairs in the test set that have never appeared in the training set (NP: 79% on CDR and more than 93% on SemEval, DDI-2013, ScienceIE and Phenebank). These two characteristics indicate the importance of understanding the mechanisms by which neural networks can generalize, *i.e.* make accurate predictions on novel instances.

3.2 Model Architecture

Our 'Man for All SeasonS' (MASS) model comprises an embeddings layer, multi-channel bidirectional Long Short-Term Memory (BLSTM)

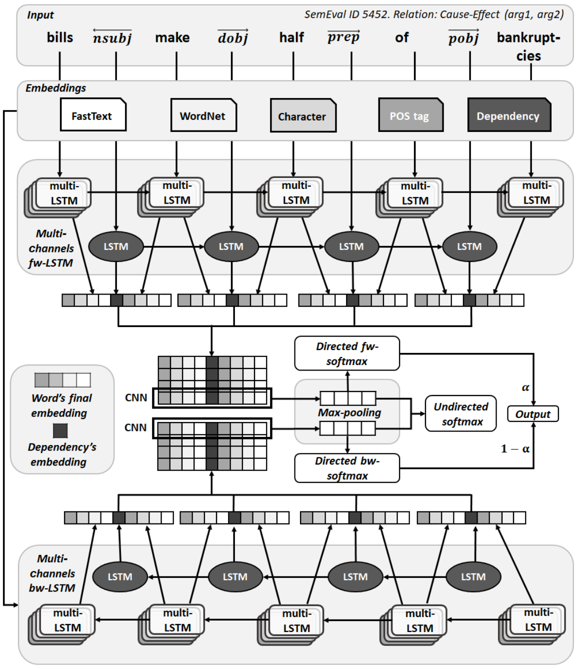

Figure 2: The architecture of MASS model for relation classification. An embeddings layer is followed by multi-channel bi-directional LSTM layers, two parallel CNNs and three *softmax* classifiers. The model's input makes use of words and dependencies along the SDP going from the first entity to the second one using both forwards and backwards sequences.

layers, two parallel Convolutional Neural Network (CNN) layers and three $softmax$ classifiers. The MASS model's architecture is depicted in Figure 2. MASS makes use of words and dependencies along the SDP going from the first entity to the second one using both forwards and backwards sequences. As is standard practice (Xu et al., 2015; Cai et al., 2016; Mehryary et al., 2016; Panyam et al., 2018) an entity pair is classified as having a relation if and only if the SDP between them is classified as having that relation.

3.2.1 Embeddings layer

Despite the presence of inter-sentential relations in the six corpora we make the simplifying assumption that relations occur only between entities (or nominals) in the same sentence. We model each such sentence using a dependency path. In order to classify novel dependency paths we represent a dependency relation d_i as a vector D_i that is the concatenation of two vectors as follow:

$$D_i = Dtyp_i \oplus Ddir_i$$

where $Dtyp$ is the undirected dependency vector, expressing the dependency type among 63 labels and, $Ddir$ is the orientation of the dependency vector *i.e.* from left-to-right or vice versa in the order of the SDP. Both are initialized randomly.

For **word representation**, we take advantage of four types of information, including:

- ***FastText pre-trained embeddings*** (Bojanowski et al., 2017) are the 300-dimensional vectors that represent words as the sum of the skip-gram vector and character n-gram vectors to incorporate sub-word information.

- ***WordNet embeddings*** are in the form of one-hot vectors that determine which sets in the 45 standard WordNet super-senses the tokens belong to.

- ***Character embeddings*** are denoted by $\mathbb{C}$, containing 76 entries for 26 letters in uppercase and lowercase forms, punctuation, and numbers. Each character $c_j \in \mathbb{C}$ is randomly initialized. They will be used to generate the token's character-based embeddings.

- ***POS tag embeddings*** capture (dis)similarities between grammatical properties of words and their lexical-syntactic roles within a sentence. We randomly initialized these vectors values for the 56 POS tags in OntoNotes v5.0.

Note that all initializations are generated by looking up the corresponding lookup table. The character and POS tag embeddings lookup tables were randomly constructed according to the $Glorot$ uniform initializer (Glorot and Bengio, 2010) and then treated as the model's parameters to be learned in the training phase.

3.2.2 Multi-channel Bi-LSTM

For a given linguistic feature type, LSTM networks (Hochreiter and Schmidhuber, 1997) are employed to capture long-distance dependencies along two directions, namely the forward and backward Bi-directional LSTM (BLSTM).

For the **dependencies**, BLSTMs take as input a sequence of dependency embeddings D_i, then gives output are the hidden states for dependencies between adjacent tokens w_i and w_{i+1} as $fwDEP_{ii+1}$ and $bwDEP_{ii+1}$.

Apart from the dependencies between tokens in SDPs, our model exploits four linguistic embeddings relating to words for **representing the**

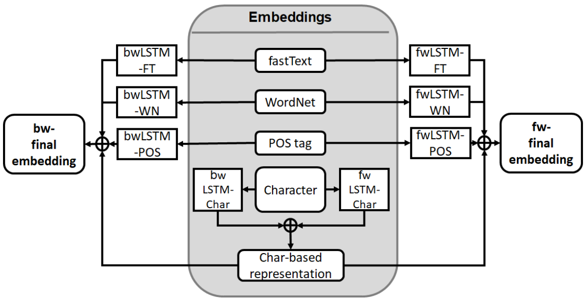

Figure 3: The multi-channel LSTM for word representation. Each token in the SDP is represented by using four word-related embeddings, including FastText word embedding, WordNet embedding, POS tag embedding and the character embedding. These four types of word-related information are fed into eight separate LSTMs, independently from each other during recurrent propagation.

words. These four types of word-related information are fed into eight separate LSTMs (four for each direction) independently from each other during recurrent propagation. These four BLSTM channels are illustrated in Figure 3. The morphological surface information is represented with character-based embedding using a BLSTM, in which the forward and backward LSTM hidden states are jointly concatenated (Ling et al., 2015; Dang et al., 2018). For other layers, the LSTM hidden states are concatenated separately as the forward and the backward vector to form two final embeddings for each token as follows:

$$fwW_i = fwFT_i \oplus fwWN_i \oplus Char_i \oplus fwPOS_i$$

$$bwW_i = bwFT_i \oplus bwWN_i \oplus Char_i \oplus bwPOS_i$$

3.2.3 CNN with dependency unit

Similar to Cai et al. (2016), the Convolutional Neural Networks (CNNs) in our model utilize **Dependency Units (DU)** to model the SDP. DU has the form of $[w_i - d_{ii+1} - w_{i+1}]$, in which w_i, w_{i+1} are two adjacent tokens and d_{ii+1} is the dependency between them. As a result, the low-dimensional forward and backward representation vectors of DU_j are created by concatenating the corresponding final embeddings of tokens w_j, w_{j+1} and the LSTM hidden state of the dependency d_{ii+1}. Formally, we have:

$$fwDU_j = fwW_j \oplus fwDEP_{jj+1} \oplus fwW_{j+1}$$

$$bwDU_j = bwW_j \oplus bwDEP_{jj+1} \oplus bwW_{j+1}$$

The forward and backward SDP representation matrices fwS and bwS are created by stacking the $fwDU$ and $bwDU$ vectors. We then apply two parallel CNNs to fwS and bwS to capture the context features (CF_j) around each dependency unit DU_j in the SDP as follows. These CNNs are designed similarly to the original CNN for sentence classification (Kim, 2014).

$$fwCF_j = f(We_{CNN} \cdot fwDU_j + b_{CNN})$$

$$bwCF_j = f^{'}(We^{'}_{CNN} \cdot bwDU_j + b^{'}_{CNN})$$

where We_{CNN} and $We^{'}_{CNN}$ are the weight matrices for the CNNs, b_{CNN} and $b^{'}_{CNN}$ are the bias terms for the hidden state vectors and f and $f^{'}$ are the non-linear activation functions.

The $n-$max pooling (Boureau et al., 2010) layer gathers the most useful global information G over the whole SDP (Collobert et al., 2011) from the context features of dependency units, which is defined as follows (in this work, we use $1-$max pooling).

$$fwG = \max_{d=1}^{k} fwCF_j$$

$$bwG = \max_{d=1}^{k} bwCF_j$$

where max is an element-wise function, and k is the number of dependency units in the SDP.

3.2.4 Softmax classifiers

Following (Cai et al., 2016), relation classification based on fwS and bwS simultaneously can strengthen the model's ability to judge the direction of relations. We, therefore, use two directed $softmax$ classifiers, one for each direction of the relation, with linear transformation to estimate the probability that each of fwS and bwS belongs to a directed relation (the direction taken into account). Formally we have:

$$p(fw) = softmax(W_f \cdot fwG + b_f)$$

$$p(bw) = softmax(W^{'}_f \cdot bwG + b^{'}_f)$$

where W_f and $W^{'}_f$ are the transformation matrices and b_f and $b^{'}_f$ are the bias vectors.

These two distributions are then combined to get the final distribution with a priority weight α:

$$p = \alpha \cdot p(fw) + (1 - \alpha) \cdot p(bw)$$

We also use the undirected $softmax$ to predict undirected distribution $p(ud)$. This $softmax$ is only used in the training objective function, which is the penalized cross-entropy of three $softmax$

classifiers. Our undirected softmax is quite similar to the idea of coarse-grain softmax used in Cai et al. (2016); Zhou et al. (2018).

$$p(ud) = softmax(W''_f \cdot [fwG \oplus bwG] + b''_f)$$

where W''_f is the transformation matrix and b'_f is the bias vector.

3.3 Additional Techniques

Mehryary et al. (2016) demonstrated that random initialization can, to some extent, have an impact on the model's performance on unseen data, i.e, individual trained models may perform substantially better (or worse) than the averaged results.

Further, an ensemble mechanism, was found to reduce variability whilst yielding better performance than the averaging mechanism. Two simple but effective ensemble methods include strict majority vote (Mehryary et al., 2016) and weighted sum over results (Ammar et al., 2017; Lim et al., 2018; Verga et al., 2018). Since the former brings better results in our experiments, our ensemble system runs the model for 20 times and uses the strict majority vote to obtain the final results.

For dealing with the imbalanced data problem, we apply an under-sampling technique (Yen and Lee, 2006) during pre-processing for the DDI-2013 and Phenebank corpora. For a fair comparison we also apply some simple rules that was used by comparison models as the pre/post-processing step for DDI-2013 (following Zhou et al. (2018)), BCR (following Gu et al. (2017)) and ScienceIE (following Lee et al. (2017)) (for further details, see Appendix A).

Finally, we use several techniques to overcome over-fitting, including: *max-norm regularization* for Gradient descent (Qin et al., 2016); adding *Gaussian noise* (Quan et al., 2016) with mean 0.001 to the input embeddings; applying *dropout* (Srivastava et al., 2014) at 0.5 after all embedding layers, LSTM layers and CNN layers; and using *early stopping* technique (Caruana et al., 2000).

4 Results and Discussion

For each benchmark dataset we adopt the official task evaluations for system with $F1$ score, precision P and recall R. All official evaluations only considered the actual relations (excluding the *Other* relation and negatives) and worked on the abstract level (excepted SemEval). For a clearer comparison, we also report both averaged and ensemble results, in which, the averaged results are calculated over 20 different runs. Both results of the MASS model with and without applying pre/post-processing rules are also reported.

Model	Source of information	F1
SVM (Rink and Harabagiu, 2010)	Rich features	82.2
CNN + Attention Shen and Huang (2016)	Position, WordNet, words around nominals	85.9
BLSTM + CNN (Cai et al., 2016)	NER, WordNet w/o inversed SDP* w/ inversed SDP	 83.8 86.3
BLSTM + CNN + attention (Zhang et al., 2018)	Position embedding	83.7
Baseline model	WordNet, Character embeds	85.0
MASS model	WordNet, Character embeds (+ Inversed SDP) + Ensemble	85.9 85.4 **86.3**

Table 3: Comparison of our system with top performing systems on the SemEval 2010 corpus. The official evaluation is based on the macro-averaged F1. Since most of the comparative models did not report their P and R, we only report our F1 for comparison. All deep learning models use word embedding and POS tag information. *We report results for our implementation of Cai et al.'s system, without using the inversed SDP.

We compare the performance of the MASS model against three types of competitors: (i) A baseline model is used to verify the effectiveness of the multi-channel LSTM, in which we concatenate all embedding vectors used in MASS directly. (ii) The first ranked in the original challenges. (iii) Recent models with state-of-the-art results. The comparative results are shown in Tables 3 - 8.

In all corpora, the MASS model's results are always better than the baseline model. This is because directly concatenating many vectors with various value ranges seems to be causing information interference, and we cannot take advantages of each sequence of information separately anymore.

In **SemEval2010 corpus** (see Table 3), the macro-averaged $F1$ of the original model is 85.9% with the standard deviation of 20 runs is 0.33. This result outperforms all comparative models but Cai et al. (2016) which fed the inversed SDP to enrich the training data (we also tried feeding inversed SDP to the model, but the result became worse since this technique may be unsuitable for our model). Applying ensemble procedure boosts $F1$ for 0.45%, outperforming all comparative models.

Model	Source of information	P	R	F1
2-phase classification Hybrid kernel SVM[1]	Heterogeneous set of feature, rule-based negative filtering	64.6	65.6	65.1
2-phase classification SVM[2]	Rich features	73.6	70.1	71.8
BLSTM + Attention (Zhou et al., 2018)	Position-aware attention + Pre-processing	75.8	70.3	73.0
Baseline model	WordNet, Character embeds	51.6	52.9	52.2
MASS model	WordNet, Character embeds	54.0	56.3	55.1
	+ Ensemble	56.5	57.3	56.0
	+ Pre-processing	57.0	56.5	56.7

Table 4: Results on the DDI-2013 corpus. The official evaluation is the micro-averaged P, R and F1 at abstract-level. Note that all deep learning models use word embedding and POS tag information. [1]Chowdhury and Lavelli (2013). [2]Raihani and Laachfoubi (2017).

For dealing with **DDI-2013** (see Table 4)- an imbalanced data, comparative models often consider it as two sub-tasks, *i.e.* detection and classification. Chowdhury and Lavelli (2013); Raihani and Laachfoubi (2017) applied a two-phrase classification, in which one classifier detects positive instance and the other then classifies them. Zhou et al. (2018) used a binary softmax together with a multi-class softmax. Obviously, our model encounters a serious problem with imbalanced data. Since we treat the RE problem as a multi-class classification, in which, negative is also considered as a class, our results are much lower than comparative models. We applied negative undersampling technique and the pre-processing rules from Zhou et al. (2018) to remove some negatives, however the rules improved performance only slightly (0.3%).

Since our system just extracts the relations within a sentence, for **CDR** (see Table 5)- a corpus where 30% instances are cross-sentence relations, it is reasonable to explain why our recall is much lower than the comparative systems that can extract cross-sentences relations (Gu et al., 2017; Verga et al., 2018). Our results are still extremely encouraging since the $F1$ is better than other models which do not extract cross-sentences relations (Gu et al., 2017; Panyam et al., 2018). For a clearer comparison, we also try applying post-processing rules used by Gu et al. (2017), and they help to increase the $F1$ by 3.3%. Our $F1$ is just a little lower than the combined model of CNN and ME which extracts cross-sentence relations (Gu et al., 2017). The results for BRAN (Verga et al., 2018) however are much better than our MASS model. It is a a strong competitor on this benchmark that is designed to focus on cross-sentence relation classification by creating the document-level graph and is also trained using auxiliary data.

Model	Source of information	P	R	F1
CNN + ME[1] (Gu et al., 2017)	Contextual of whole sentence	59.7	57.5	57.2
	+ Cross-sentence	60.9	59.5	60.2
	+ Post processing	55.7	68.1	**61.3**
ASM[2] (Panyam et al., 2018)	Dependency graph	49.0	67.4	56.8
BRAN[3] (Verga et al., 2018)	Position, multi-head att	55.6	70.8	62.1
	+ Data	64.0	69.2	**66.2**
	+ Ensemble	63.3	67.1	**65.1**
Baseline model	WordNet, character embeds	56.6	54.1	55.3
MASS model	WordNet, character embeds	58.9	54.9	56.9
	+ Ensemble	56.8	57.9	57.3
	+ Post-processing	52.8	71.1	60.6

Table 5: Results on the CDR corpus. The official evaluation is reported at abstract-leve. All deep learning models use word embedding and POS tag information. [1]CNN + Maximum Entropy. [2]Approximate Subgraph Matching. [3]CNN + attention at abstract-level graph.

Model	Source of information	P	R	F1	IntraF
VERSE (SVM)[1]	Rich features	51.0	61.5	55.8	63.4
TurkuNLP (RNN)[2]		62.3	44.8	52.1	62.0
DET-BLSTM (Li et al., 2017)	Dynamic ext dep tree, distance embeddings	56.3	58.0	**57.1**	–
Baseline model	WordNet, Char embds	60.8	47.2	53.1	62.5
MASS model	WordNet, Char embds	59.8	51.3	55.2	64.6
	+ Ensemble	59.2	52.2	55.5	**64.8**

Table 6: Results on the BB3 corpus. The official evaluation is reported at both abstract- and intra sentence levels. All deep learning models use word embedding and POS tag information. [1]Lever and Jones (2016). [2]Mehryary et al. (2016)

In the **BB3** corpus (see Table 6), the original system outperforms all previously reported results at intra-sentence F. Using ensemble procedure, our results increase, but not much and still lower than the DT-BLSTM model, which is based on Dynamic Extended Tree (Li et al., 2017).

In the **ScienceIE** corpus (see Table 7), our results are only outperformed by one competitor. The reason may come from the characteristic of Hyponym-of and Synonym-of relations. Neither of these relations is expressed frequently by the linguistic information of tokens appearing in the SDP. In many cases, they are represented by different patterns with the same SDP. Therefore, our conclusion is that maybe the use of SDP does not match the ScienceIE corpus. The system from

Model	Source of information	F1
NTNU-2 (SVM) (Barik and Marsi, 2017)	Rich features	50.0
MIT (CNN) (Lee et al., 2017)	Relative position, NER + Post-processing	**64.5**
S2_rel (BLSTM) (Ammar et al., 2017)	Semisupervised, language model + Ensemble	54.1 55.2
Baseline model	WordNet, character embeds	48.7
MASS model	WordNet, character embeds + Ensemble + Post- processing (Lee et al., 2017) + Post- processing (rules ++)	54.6 56.4 60.3 **73.0**

Table 7: Results on the ScienceIE corpus. The official evaluation is based on the micro-averaged F1 at abstract-level. Since most of comparative models did not report their P and R, we only report our F1 for comparison. All deep learning models use word embedding and POS tag information.

MIT (Lee et al., 2017) fed the whole sentence with the relative position as input, therefore it may catch many useful patterns which did not appear in the SDP. To test this hypothesis, we apply the post-processing rules used in Lee et al. (2017) and boosted $F1$ by 3.8%. In addition, when we applied some more simple linguistic rules to identify synonyms and hyponyms, the results improved beyond expectations by 16.6%, totally outperformed all other models.

For **Phenebank** (see Table 8), since this new corpus did not have an official evaluation, we report all possible MASS results. The micro-averaged results are much better than the macro-averaged. It is reasonable since Phenebank is an extremely imbalanced corpus, in which we can expect poor accuracy for rare classes, which together account for about 1% of positive data (and positive data only account for 23% of the whole corpus). The micro-averaged and macro-averaged results of the proposed model are always better than the baseline model, in both abstract and sentence-level. Interestingly, the ensemble model boosts the micro-averaged results (1.33% of $F1$ at sentence-level and 0.88% of $F1$ at abstract-level), but brings lower macro-averaged $F1$ (decreased 0.51% and 0.77% of $F1$ at sentence- and abstract-level respectively).

4.1 Components and Information resources

We study the contribution of each model's component and information sources to the system performance by ablating each of them in turn from the model and afterwards evaluating the model on all corpora. We compare these experimental results

			Baseline	Averaged	Ensemble
Sentence level	Macro-averaged	P	45.8	43.6	44.2
		R	39.2	42.6	41.1
		F	42.2	**43.1**	42.6
	Micro-averaged	P	56.5	53.2	55.4
		R	56.2	62.3	62.3
		F	56.4	57.3	**58.7**
Abstract level	Macro-averaged	P	45.8	43.6	44.2
		R	27.3	29.7	28.4
		F	34.3	**35.3**	34.6
	Micro-averaged	P	56.5	53.2	55.5
		R	37.5	41.6	41.6
		F	45.1	46.7	**47.5**

Table 8: Experimental results on the Phenebank corpus for the MASS model.

with the full system's results and then illustrate the changes of F1 in Figure 4. The changes of $F1$ show that all model's components and information sources help the system to boost its performance (in terms of the increments in $F1$) in all corpora. The contribution, however, varies among components, information types and among corpora.

Among information sources, FastText embedding (FT) often has the most important contribution, while using WordNet (WN) brings quite small improvements. Some examples clearly demonstrate that the impact of information sources varies greatly between benchmarks. The dependency embedding (DEP) and type embedding ($Dtyp$) have a very strong influence over the results in DDI-2013 and ScienceIE corpora but not much in other corpora. Furthermore, POS tag information (POS) plays a very important role in the BB3 corpus, surpassing FT, while its contribution in other corpora is not significant.

Also, the impact of model components shows relatively inconsistent across corpora. The baseline models always have lower $F1$ than MASS. This demonstrates the advantage of using a multi-channel LSTM to represent various linguistic information. Furthermore, the contributions of multi-channel LSTM and CNN are quite balanced. Interestingly, the undirected softmax always benefits the result although it was only used to calculate the penalty in the training step.

These experiments prove the effectiveness of using various information as well as architectural components. More importantly, these results show that our proposed MASS model can automatically adjust to each corpus, highlighting the flexibility of the MASS model which is able to adapt to various datasets with many different characteristics.

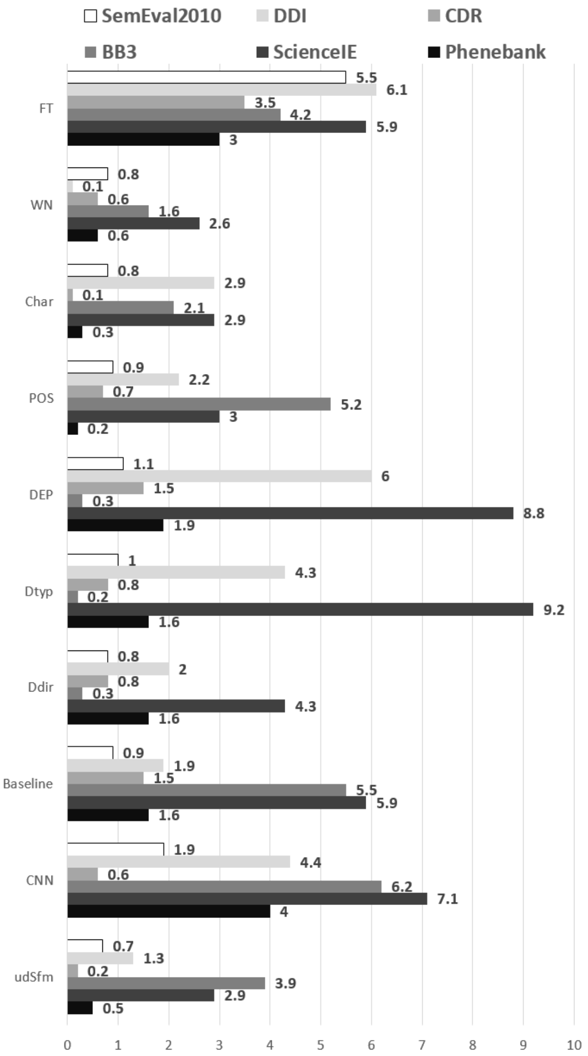

Figure 4: Ablation test results for various components and information sources: FastText (FT), WordNet (WN), Character-based (Char), POS tag, Dependency (DEP), dependency type (Dtyp) and dependency direction embedding (Ddir). Results are calculated based on the averaged F1 over 20 different runs. Baseline: Concatenating all embedding vectors to represent the words instead of using multi-channel LSTM. CNN: Using the final LSTM hidden states instead of CNN. udSfm: Removing the undirected softmax

4.2 Error Analysis

We studied model outputs to analyze system errors that defined the limitations of the model as well as to prioritize future directions. Many errors seem attributable to the parser. In some cases, we cannot generate the SDP, and in some cases where we have the SDP, information on the SDP is still insufficient or redundant to make the correct prediction. The directionality of relations is also challenging; in some cases the relation is predicted correctly but in the wrong direction. Other errors can be attributed to the limitations of our model, including (a) the inability to extract cross-sentence relations (accounting for 30% in CDR, BB3 and Phenebank), (b) the over-fitting problem (leading to wrong prediction - FP) and (c) limited generalisation power in predicting new relations (FN). Finally, we found some errors caused by the imperfect annotation. This problem may come from the different annotations assigned independently by two annotators (see IAA column in Table 2). We illustrate the above issues using realistic examples in Appendix C.

5 Conclusions

In this paper, we have presented a novel well-balanced relation classification model that consists of several deep learning components applied to the Dependency Unit of Shortest Dependency Path. We evaluated our model on six benchmark datasets, comparing the results with 15 recent state-of-the-art models. Experiments were also carried out to verify the rationality and impact of various model components and information sources. Experimental results demonstrated the robustness and adaptability of our system to classify different relation types in various domains without any architectural changes.

One existing issue with our model lies in its sensitiveness to class imbalance. This limitation resulted in significantly low performance on the DDI-2013 corpus (compared to state-of-the-art results). Our experiments also highlighted the existing challenges for neural relation classification models, including cross-sentence relations and imbalanced data. We aim to address these problems in future work.

Acknowledgments

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.052016.14. We also gratefully acknowledge the funding support of the EPSRC (N.Collier - Grant No. EP/M005089/1) and MRC (M. T. Pilehvar - Grant No. MR/M025160/1) for PheneBank. We also thank the anonymous reviewers for their comments and suggestions.

References

Waleed Ammar, Matthew Peters, Chandra Bhagavatula, and Russell Power. 2017. The ai2 system at semeval-2017 task 10 (scienceie): semisupervised end-to-end entity and relation extraction. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 592–596.

Chinatsu Aone and Mila Ramos-Santacruz. 2000. Rees: a large-scale relation and event extraction system. In *Proceedings of the sixth conference on Applied natural language processing*, pages 76–83.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, and Lakshmi Vikraman andAndrew McCallum. 2017. Semeval 2017 task 10: Scienceie - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 546–555. Association for Computational Linguistics.

Biswanath Barik and Erwin Marsi. 2017. Ntnu-2 at scienceie: Identifying synonym and hyponym relations among keyphrases in scientific documents. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 965–968.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 724–731.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 756–765.

Rich Caruana, Steve Lawrence, and C. Lee Giles. 2000. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS)*, pages 402–408.

Paulo R Cavalin, Fillipe Dornelas, and Sérgio MS da Cruz. 2016. Classification of life events on social media. In *29th SIBGRAPI (Conference on Graphics, Patterns and Images)*.

Md. Faisal Mahbub Chowdhury and Alberto Lavelli. 2013. Fbk-irst: A multi-phase kernel based approach for drug-drug interaction detection and classification that exploits linguistic information. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval-2013)*, pages 351–355. Association for Computational Linguistics.

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

Bharath Dandala, Diwakar Mahajan, and Murthy V Devarakonda. 2017. Ibm research system at tac 2017: Adverse drug reactions extraction from drug labels. In *TAC*.

Thanh Hai Dang, Hoang-Quynh Le, Trang M Nguyen, and Sinh T Vu. 2018. D3ner: Biomedical named entity recognition using crf-bilstm improved with finetuned embeddings of various linguistic information. *Bioinformatics*.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 358–365.

Louise Deléger, Robert Bossy, Estelle Chaix, Mouhamadou Ba, Arnaud Ferré, Philippe Bessières, and Claire Nédellec. 2016. Overview of the bacteria biotope task at bionlp shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 12–22. Association for Computational Linguistics.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haifa Zargayouna, and Thierry Charnois. 2018. Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 679–688.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*.

Jinghang Gu, Fuqing Sun, Longhua Qian, and Guodong Zhou. 2017. Chemical-induced disease relation extraction via convolutional neural network. *Database (Oxford)*, 2017:bax024.

Harsha Gurulingappa, Abdul Mateen-Rajpu, and Luca Toldo. 2012. Extraction of potential adverse drug events from medical case reports. *Journal of biomedical semantics*, 3(1):15.

Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó. Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

María Herrero-Zazo, Isabel Segura-Bedmar, Paloma Martínez, and Thierry Declerck. 2013. The ddi corpus: An annotated corpus with pharmacological substances and drugdrug interactions. *Journal of Biomedical Informatics*, 46(5):914–920.

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

Ferdaous Jenhani, Mohamed Salah Gouider, and Lamjed Ben Said. 2016. A hybrid approach for drug abuse events extraction from twitter. *Procedia computer science*, 96:1032–1040.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1746–1751.

Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. 2017. Mit at semeval-2017 task 10: Relation extraction with convolutional neural networks. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 978–984.

Jake Lever and Steven JM. Jones. 2016. Verse: Event and relation extraction in the bionlp 2016 shared task. In *Proceedings of the the 4th BioNLP Shared Task Workshop*, pages 42–49.

Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database Oxford*, 2016:baw068.

Jiwei Li, Alan Ritter, Claire Cardie, and Eduard Hovy. 2014. Major life event extraction from twitter based on congratulations/condolences speech acts. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1997–2007.

Lishuang Li, Jieqiong Zheng, and Jia Wan. 2017. Dynamic extended tree conditioned lstm-based biomedical event extraction. *International Journal of Data Mining and Bioinformatics*, 17(3):266–278.

Sangrak Lim, Kyubum Lee, and Jaewoo Kang. 2018. Drug drug interaction extraction from the literature using a recursive neural network. *PloS One*, 13(1).

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530. Association for Computational Linguistics.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220. International World Wide Web Conferences Steering Committee.

Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica*, 22(3):276–282.

Farrokh Mehryary, Jari Björne, Sampo Pyysalo, Tapio Salakoski, and Filip Ginter. 2016. Deep learning with minimal training data: Turkunlp entry in the bionlp shared task 2016. In *Proceedings of the the 4th BioNLP Shared Task Workshop*, pages 73–81. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011.

Makoto Miwa, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010. Event extraction with complex event classification using rich features. *Journal of bioinformatics and computational biology*, 8(01):131–146.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48.

Nagesh C. Panyam, Karin Verspoor, Trevor Cohn, and Kotagiri Ramamohanarao. 2018. Exploiting graph kernels for high performance biomedical relation extraction. *Journal of biomedical semantics*, 9(1):7.

Pengda Qin, Weiran Xu, and Jun Guo. 2016. An empirical convolutional neural network approach for semantic relation classification. *Neurocomputing*, 190.

Chanqin Quan, Lei Hua, Xiao Sun, and Wenjun Bai. 2016. Multichannel convolutional neural network for biological relation extraction. *BioMed research international*.

Anass Raihani and Nabil Laachfoubi. 2017. A rich feature-based kernel approach for drug-drug interaction extraction. *International journal of advanced computer science and applications*, 8(4):324–3360.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics.

Isabel Segura-Bedmar, Paloma Martínez, and María Herrero Zazo. 2014. Lessons learnt from the ddiextraction-2013 shared task. *Journal of Biomedical Informatics*, 51:152–164.

Yatian Shen and Xuanjing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2526–2536.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540.

Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. 2009. Unsupervised relation extraction by mining wikipedia texts using information from the web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1021–1029.

Show-Jane Yen and Yue-Shi Lee. 2006. Undersampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Proceedings of Intelligent Control and Automation*, pages 731–740.

Xiaobin Zhang, Fucai Chen, and Ruiyang Huang. 2018. A combination of rnn and cnn for attention-based relation classification. *Procedia Computer Science*, 131:911917.

Deyu Zhou, Lei Miao, and Yulan He. 2018. Position-aware deep multi-task learning for drugdrug interaction extraction. *Artificial intelligence in medicine*, In Press.

Possessors Change Over Time: A Case Study with Artworks

Dhivya Chinnappa and **Eduardo Blanco**
Human Intelligence and Language Technologies Lab
University of North Texas
Denton, TX, 76203
dhivyainfantchinnappa@my.unt.edu, eduardo.blanco@unt.edu

Abstract

This paper presents a corpus and experimental results to extract possession relations over time. We work with Wikipedia articles about artworks, and extract possession relations along with temporal information indicating when these relations are true. The annotation scheme yields many possessors over time for a given artwork, and experimental results show that an LSTM ensemble can automate the task.

1 Introduction

All languages have a way to express possessive relationships (Aikhenvald and Dixon, 2012). Possession is an asymmetric semantic relation between two entities, where one entity (the possessee) *belongs* to the other entity (the possessor) (Stassen, 2009). When it comes to defining possession, *belongs* includes a wide range of relationships, including kinship (e.g., *[my]*$_{\text{possessor}}$ *oldest [son]*$_{\text{possessee}}$), part-whole (e.g., *the [car]*$_{\text{possessor}}$*'s [dashboard]*$_{\text{possessee}}$), possession of something intangible (e.g., *[John]*$_{\text{possessor}}$ *got the [flu]*$_{\text{possessee}}$ *last year*), proximity (e.g., *The [shelf]*$_{\text{possessor}}$ *has a [glass sculpture]*$_{\text{possessee}}$), physical and temporary possession (e.g., *[I]*$_{\text{possessor}}$ *have John's [book]*$_{\text{possessee}}$), and ownership (e.g., *[John]*$_{\text{possessor}}$ *bought a [house]*$_{\text{possessee}}$ *last year*).

Possession relations can be divided into alienable (also referred to as acquired, transferable, non-intimate, etc.) and inalienable (also referred to as inherent, inseparable, intimate, etc.). Possessees that can be separated from their possessors are alienable, and possessees that cannot normally be separated from their possessors are inalienable (Heine, 1997). For example, *[John]*$_x$*'s [condo]*$_y$ is alienable, and *[John]*$_x$*'s [arm]*$_y$ is inalienable (some previous works would call the latter a part-whole relation instead). Tham (2004) defines control possession as a relation in which the possessor has temporary control of the possessee, but does not necessarily alienably possess it (e.g., *[John]*$_x$ *borrowed the [car]*$_y$ *for the weekend*). Following the aforecited works, possession goes beyond ownership of property.

In 1530 the painting was inherited by Margaret's niece Mary of Hungary, who [...]. It is clearly described in an inventory taken after her death in 1558, when it was inherited by Philip II of Spain.

Mary of Hungary: 1530–1558
Philip II of Spain: after 1558

Figure 1: Excerpt from the Wikipedia article about *Arnolfini Portrait* and its possessors over time.

Virtually all possessees change possessors over time, especially if possession relationships are understood in a broad sense as outlined above. Consider the excerpt of the Wikipedia article about the Arnolfini Portrait in Figure 1. From this excerpt, we know that the painting had at least two possessors (*Mary of Hungary* and *Philip II of Spain*), and that they were the possessors from 1530 to 1558 and after 1558 respectively.

In this paper, we track possessors of selected possessees over time. Unlike most previous works (Section 3), we (a) start with a document relevant to the possessee of interest, (b) select plausible possessors and years without syntactic restrictions and including inter-sentential pairs, and then (c) determine whether the plausible possessors are actual possessors with respect to the years. The main contributions of this paper are: (a) 88 Wikipedia articles about artworks annotated with their possessors over time;[1] (b) a detailed corpus analysis (e.g., unique possessors, years and possessor-year pairs); and (c) experimental results showing that an LSTM ensemble outperforms SVM.

---

[1] Available at dhivyachinnappa.com

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2278–2287
Brussels, Belgium, October 31 - November 4, 2018.

2 Previous Work

We briefly summarize work on possession relationships from a theoretical perspective, and then move to work in computational linguistics.

2.1 Possession relations

The very definition of possession is not set in stone. Aikhenvald (2013) distinguishes three core meanings for possessive noun phrases that occur across languages: ownership (of property), whole-part (often referred to as part-whole), and kinship. Following a cross-linguistic perspective, she discusses possessions and time (present and former possession relationships, e.g., *my tooth* vs. *my former axe*), temporary and permanent possession (e.g., *borrow* vs. *acquire*) and others. Heine (1997) classifies possession relationships depending on the possessor and possessee. First, he makes a distinction between human (e.g., *[I]*$_x$ *have a [house]*$_y$) and non-human possessors (e.g. *[This house]*$_x$ *has [two bedrooms]*$_y$). Second, he differentiates three kinds of possession depending on the possessee: concrete possession (e.g., *[I]*$_x$ *have [two cats]*$_y$), social possession (e.g., *[I]*$_x$ *have [two sisters]*$_y$), and abstract possession (e.g., *[I]*$_x$ *have [an idea]*$_y$). Miller and Johnson-Laird (1976) differentiate between three kinds of possession: inherent, accidental, and physical; and provide the following example: *He owns an umbrella (inherent), but she's borrowed it (accidental), though she doesn't have it with her (physical).*

Possession relations have also been defined in terms of parameters. For example, Stassen (2009) considers two parameters (permanent contact and control) and Heine (1997) defines five parameters (human possessor, concrete possessee, spatial proximity, temporal permanence, and control).

While we do not closely follow any of these previous works, we borrow from them the broad definition of possession relations, and the motivation to work with possessions over time.

2.2 Computational Linguistics

Within computational linguistics, possession relations have been mostly studied as one of the many relations encoded in a given syntactic construction. For example, Tratz and Hovy (2013) extract semantic relations within English possessives. They propose a set of 18 relations, e.g. *temporal* (e.g., *[today]*$_x$*'s [rates]*$_y$), *extent* (e.g., *[6 hours]*$_y$*' [drive]*$_x$). Their *controller / owner / user* relation (one relation with three aliases) is the closest relation to the possession relations we target in this paper. Extracting semantic relations between noun compounds (Nakov and Hearst, 2013; Tratz and Hovy, 2010) usually includes extracting possession relations, e.g., *[family]*$_x$ *[estate]*$_y$. These previous works extract all semantic relations—including possessions—between arguments that follow a syntactic construction.

In our previous work (Chinnappa and Blanco, 2018), we identify possession relations between a deterministically chosen person (possessor) and a concrete object (possessee) within a sentence. If a possession relation exists, we also identify the possession type (alienable or control). Finally, we temporally anchor the possession relation with respect to the verb of which the possessor is the subject. In this paper, we take a complementary approach. We start with text relevant to the possessee of interest—specifically, its Wikipedia article—and then extract its possessors without any restrictions beyond considering as possessors only named entities. Furthermore, we specify in which years the possessions were true.

To the best of our knowledge, the work by Banea et al. (2016) is the only one on extracting possession relations without imposing syntactic constraints. Banea and Mihalcea (2018) build a corpus working with personal blogs, and present results on automatic extraction of possession using a naive bayes approach. They consider as possessors the author of a blog, and as possessees concrete nouns in blog posts. Regarding time, they annotate possessions at the time of the utterance (when the blog posts were published). Unlike them, we work with one possessee per Wikipedia article (i.e., the artwork the article is about), and then find possessors in the article. Additionally, we extract when a possessor-possessee relation is true with respect to the years in the article, and present results using SVM and end-to-end neural networks.

3 Annotating Possessions Over Time

In this section, we detail the methodology to create a corpus of possession relations over time. We first discuss the selection of source documents and possessees of interest. Then, we detail what is considered as a potential possessor, and how these possessors are paired with years. Finally, we describe

		#	%
x, potential possessor	PERSON	1,152	48.0
	ORG	986	30.5
	GPE, LOC	692	21.5
	All	3,230	100.0
y, year	All	940	100.0
(x, y) pairs	x is PERSON	6,304	48.8
	x is ORG	3,840	29.7
	x is GPE, LOC	2,769	21.4
	All	12,913	100.0

Table 1: Counts of potential possessors (x) and years (y), and (x, y) pairs selected for annotation.

the annotation process (is the potential possessor an actual possessor with respect to the years?) and analyze the resulting corpus.

3.1 Selecting Source Documents

Our goal is to target possessors of a given possessee over time. A natural choice is to work with documents about specific objects, as they are likely to describe the history and key events involving the objects. We decided to work with Wikipedia articles about important artworks. The methodology presented here, however, is not limited to artworks, and we believe it is applicable to any article about an object of interest.

We selected 100 artworks using online content, including Google queries for *famous artwork* and *famous paintings*, and online lists.[2] Then, we downloaded the full content of the corresponding Wikipedia articles. Some of the selected artworks are The Third of May 1808, Philosopher in Meditation, and Saturn Devouring His Son. The final corpus has 88 articles because we discarded articles if we could not select at least three (potential possessor, year) pairs (see below).

3.2 Selecting Potential Possessors and Years

Once possessees and their Wikipedia articles were selected, we identified potential possessors and years following the five steps below for each section in each Wikipedia article:

1. Run the named entity recognizers in spaCy[3] and Stanford CoreNLP (Manning et al., 2014).

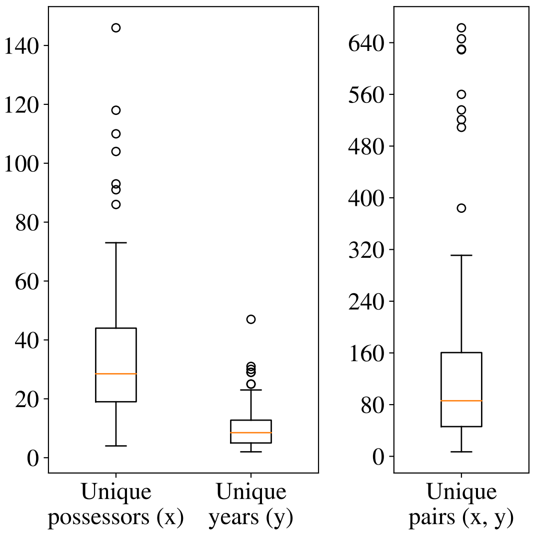

Figure 2: Distribution of unique potential possessors, unique years and unique pairs per article. Each boxplot displays the minimum, first quartile, median, third quartile and maximum.

2. Select as potential possessors all instances of the following named entities: PERSON, ORGANIZATION, LOCATION, and GPE.
3. Select as years all sequences of four digits inside a DATE named entity.
4. Remove all duplicate potential possessors and years from steps 2 and 3.
5. Generate all pairs of potential possessors and years.

Table 1 presents basic counts and percentages of the potential possessors and years after removing duplicates (Step 4), and the pairs generated (Step 5) for all documents. There are 3,230 potential possessors and 940 years, and 12,913 (potential possessor, year) pairs. The most common named entity of potential possessors is PERSON (48%), followed by ORG (30.5%) and GPE / LOC (21.5%). The percentage of (potential possessor, year) pairs depending on the named entity of the potential possessor almost follows an identical distribution (48.8%, 29.7% and 21.4%).

Figure 2 shows the distributions of unique potential possessors, years and (potential possessor, year) pairs generated per article (or equivalently, per possessee). While the distributions are far from uniform, the boxplots show that most articles have a substantial number of potential possessors, years and pairs. The minimum number of potential possessors is 4, of years 2, and of pairs 7. But

---

[2] http://en.most-famous-paintings.com, http://remliel.com/2016/07/08/100-greatest-paintings-of-all-time

[3] https://spacy.io/

	Cohen's κ
Before	0.69
During	0.59
After	0.77
All	0.70

Table 2: Inter-annotator agreement (Cohen's kappa). Values over 0.6 are considered *substantial*, over 0.8 are considered *perfect* (Artstein and Poesio, 2008).

over 75% of articles have at least 19 unique potential possessors, 5 years and 46 pairs; and over 50% of articles have at least 28 unique potential possessors, 8 years and 86 pairs. In other words, our corpus takes into account many potential possessors and years for the vast majority of articles.

3.3 Validating Possessors and Years

After (potential possessor, year) pairs were generated, they were validated manually. To do so, we asked the following questions to annotators:

- Did a possession relation exist between the potential possessor and the possessee at any point of time *before year*?
- Did a possession relation exist between the potential possessor and the possessee at any point of time *during year*?
- Did a possession relation exist between the potential possessor and the possessee at any point of time *after year*?

In all questions, *possessee* refers to the artwork the Wikipedia article is about. Annotators had to choose from two answers: `yes` or `no`, where `no` indicates all cases in which there is not enough information to determine that a possession relation exists at any point of time before / during / after year. In other words, `no` does not mean that the potential possessor did not possess the possessee, and it may mean that there is no information about whether a possession relations exists.

The annotation interface showed the title of the article and the section to which the potential possessor and year belong to (section title + text). Annotators were instructed to first read the section and then answer all questions. Thus, annotators reveal possession information involving possessors and years that are potentially far away (different clauses, sentences, etc.). Recall that all potential possessors and years within a section are paired, thus we allow to cross sentence boundaries.

Annotation Quality. Annotations were done in-house by two graduate students. Both of them annotated 25% of the articles individually. Table 2 shows inter-annotator agreements (Cohen's kappa) for each question. Overall, inter-annotator agreement is 0.70 (values between 0.60 and 0.80 are considered substantial (Artstein and Poesio, 2008)). Agreements are higher for *Before* and *After* than *During* (0.69 and 0.77 vs. 0.59). The remaining articles were annotated once.

Annotation Examples. Figure 3 shows the annotations for one paragraph of the Wikipedia article about *Girl with a Pearl Earring* (more specifically, from the section titled *Ownership and display*). The figure shows the annotations on top of a screenshot of the article for clarity purposes, but the annotation interface only showed one section at a time along with all the generated pairs (Section 3.3, equivalent to pre-drawing edges).

Five potential possessors and two years were selected, thus ten (potential possessor, year) pairs were generated. The annotations reveal the intuitive possession information contained within the paragraph. First, *Victor de Stuers* was an advisor to *Arnoldus Andries des Tombe*, so there is no evidence that he was a possessor at any point of time (missing label edges). Second, *Vermeer* is the artist who made *Girl with a Pearl Earring*, so there are possession relations before *1881* and *1902*. Third, *Arnoldus* purchased the piece in *The Hague* in 1881, and in 1902 it was donated to *Mauritshuis*. So *Arnoldus* was a possessor in 1881 and after 1881 (until 1902), *The Hague* in 1881 (recall that non-humans can be possessors, spatial proximity is also considered possession, Section 2.1), and *Mauritshuis* during and after 1902. We discuss the limitation of the annotation approach in Section 3.5.

3.4 Annotation Analysis.

Counts of `yes` labels for the three questions (before, during and after) are rather low (17%, 9% and 19%, Figure 4). This is not surprising, as any PERSON, ORG, LOC and GPE named entity is considered as potential possessors. We note, however, that we annotated a possession relation (`yes` label) in 35% of (potential possessor, year) pairs generated (either before, during or after *year*).

Figure 5 depicts the distribution of labels per article for (potential possessor, year) pairs generated from the same and different sentences. It is worth noting a couple of interesting patterns. First, the annotations contain many more possessions because we pair potential possessors and years that

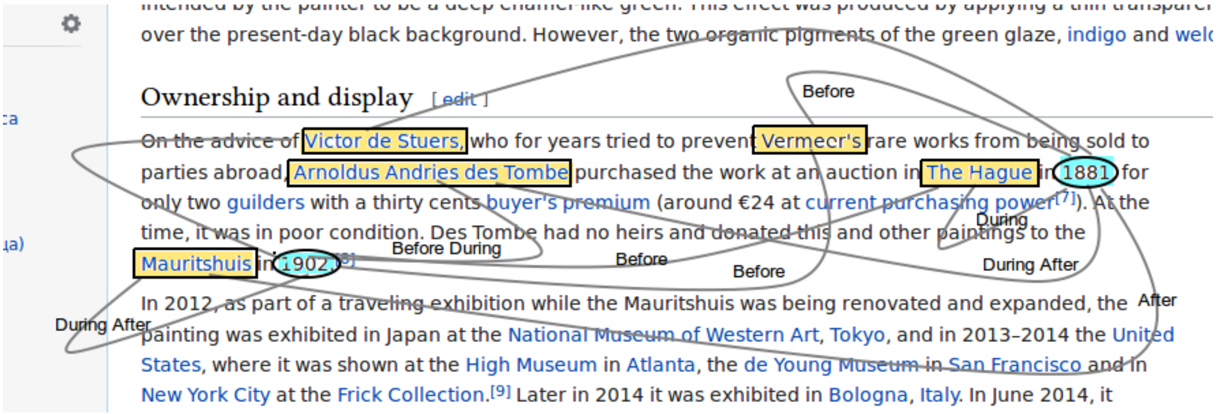
over the present-day black background. However, the two organic pigments of the green glaze, indigo and wel

Ownership and display [edit]

On the advice of Victor de Stuers, who for years tried to prevent Vermeer's rare works from being sold to parties abroad, Arnoldus Andries des Tombe purchased the work at an auction in The Hague in 1881 for only two guilders with a thirty cents buyer's premium (around €24 at current purchasing power[7]). At the time, it was in poor condition. Des Tombe had no heirs and donated this and other paintings to the Mauritshuis in 1902.[8]

In 2012, as part of a traveling exhibition while the Mauritshuis was being renovated and expanded, the painting was exhibited in Japan at the National Museum of Western Art, Tokyo, and in 2013–2014 the United States, where it was shown at the High Museum in Atlanta, the de Young Museum in San Francisco and in New York City at the Frick Collection.[9] Later in 2014 it was exhibited in Bologna, Italy. In June 2014, it

Figure 3: Excerpt of the Wikipedia article of *Girl with a Pearl Earring* and annotations. Edges indicate (potential possessor, year) pairs generated. Edge labels indicate *yes* label for *before*, *during* and *after*; missing edge labels indicates all annotations are *no* (i.e., the potential possessor was invalid).

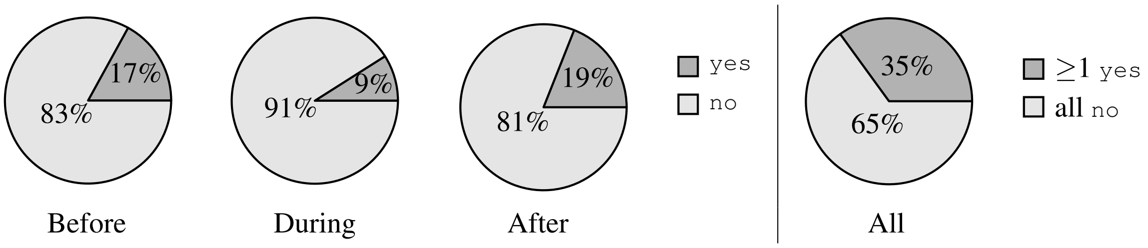

Figure 4: Label percentages for each question (left). While most labels are `no`, one of the three questions (before, during or after) was answered with `yes` in 35% of (potential possessor, year) pairs (right).

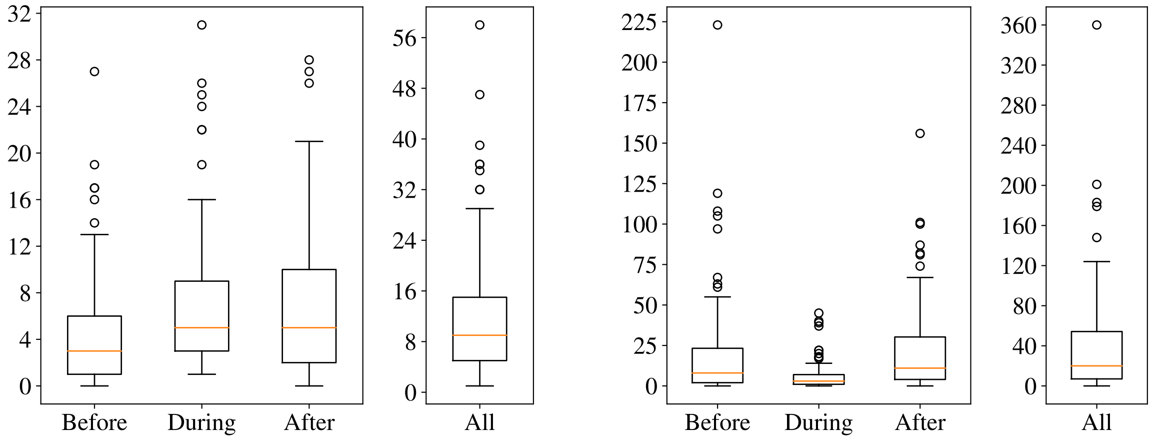

Possessor and year from the same sentence

Possessor and year from different sentences

Figure 5: Distribution of *yes* label per article. We provide distributions for each temporal anchor (at some point of time before, during or after *year*) and for all anchors, and distinguish between possessors and years belonging to the same sentence (left) or different sentences (right). Each boxplot shows the minimum, first quartile, median, third quartile, and maximum.

belong to different sentences (note the different scales in the y-axis). Second, for pairs generated from different sentences, `yes` label for *during* questions is much less likely than for the labels *before* and *after*.

Finally, Figure 6 shows the distribution of `yes` label for all generated pairs. At a minimum, each possessee has at least two possession relations. There are a few outliers articles in which annotators identified over 150 possessors in time. The

From the possessor and year: the concatenation of tokens; binary flags for each token; the syntactic head (token, lemma and part-of-speech tag); and the named entity type.
From the sentences to which the possessor and year belong to: for (a) a window of 4 tokens to the left and right, (b) all the verbs to the left and right, (c) all the verbs that are ancestors or children in the depedency tree, and (d) all the left and right siblings in the dependency tree, the tokens, lemmas and part-of-speech tags.
Other and Wikipedia article: whether the possessor and year belong to the same sentence, whether the possessor appears before or after the year, the Wikipedia article tile (concatenation of tokens and binary flags for each token), and the section title (concatenation of tokens and binary flags for each token).

Table 3: Feature set used with Support Vector Machines.

first quartile is 14, the median 30, and the third quartile 65 (All, right most boxplot). Thus the procedure presented here reveals a substantial amount of possession relations along with temporal information anchored in the form of years.

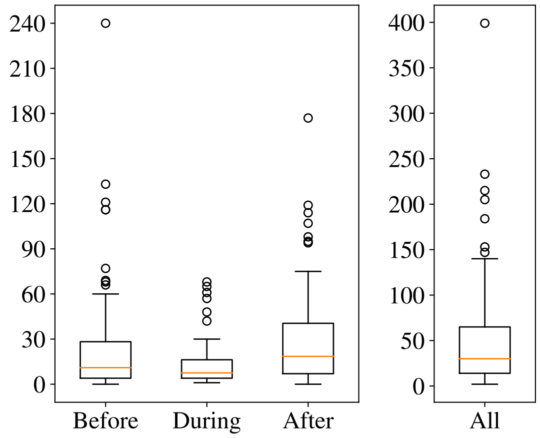

Figure 6: Distribution of *yes* label per article for all possessors and years.

3.5 Limitations

While the proposed procedure successfully identifies possession relations over time, we acknowledge limitations in both the possessors and temporal information considered.

First, we only consider named entities as potential possessors, so it is possible we miss some possessors (e.g., pronouns, *the artist*, *his son*). Because of the source documents we work with (Wikipedia articles about artwork) and the fact that we pair all potential possessors and years within a section, this is not a big issue: most Wikipedia sections do not have mentions that cannot be resolved to a named entity within the same section. We note, however, that coreference resolution (Pradhan et al., 2011) would alleviate this problem.

Second, we only consider four digits within a DATE named entity as temporal information. This means that temporal information encoded in relative dates (e.g., four years later) or historical events (e.g. after World War II) is disregarded. Additionally, we cannot distinguish between several possessors within a year, finer-grained times would be required to do so. To address these issues, temporal parsers (Lee et al., 2014; Strötgen and Gertz, 2015) and anchoring events in time (Reimers et al., 2016) are required.

4 Experiments and Results

We experiment with traditional Support Vector Machines and neural networks. We divided the articles (and the corresponding (potential possessor, year) pairs) into train (80%) and test (20%), and report results obtained with the test split. Note that splitting pairs randomly would be unsound, as possession relations for the same possessee would be in the train and test splits. We build three classifiers with both SVMs and neural networks (one per question: before, during and after), and all of them predict two labels: `yes` or `no`.

4.1 Support Vector Machines

We trained the three classifiers using the SVM implementation in scikit-learn (Pedregosa et al., 2011), and tuned hyper-parameters C and γ using 10-fold cross-validation with the train split. We used features extracted from the possessor, the year, and the sentences they belong to. Additionally, we also included the Wikipedia article title and the section title from which the possessor and year were selected. The full feature set is described in Table 3 and we do not elaborate further. Our motivation to try SVMs is to establish a strong supervised baseline, and to compare with neural networks that take as input only plain text.

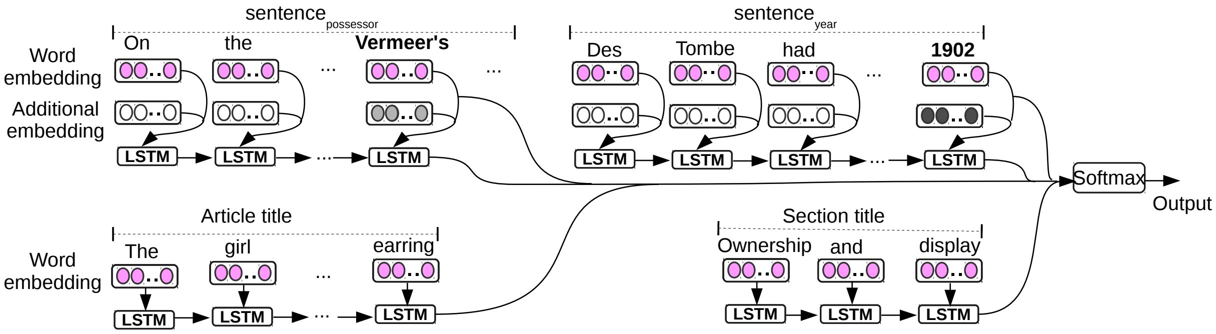

Figure 7: Neural network architecture to predict whether a *potential possessor* is a possessor before, during or after *year*. We exemplify the input to the network with the (*Vermeer*, *1902*) pair from Figure 3.

4.2 Neural Networks

We use the implementations provided by the Keras neural network API (Chollet et al., 2015) with TensorFlow backend (Abadi et al., 2015). Additionally, we use GloVe embeddings with 300 dimensions (Pennington et al., 2014)[4] to transform words into their distributed representations, the Adam optimizer (Kingma and Ba, 2014) and categorical cross entropy as a loss function. We train the network with batch size 16 for up to 200 epochs, but stop earlier if no improvement is observed in the validation set for 5 epochs. We reserve 20% of the train split for validation.

The neural network is composed of four Long Short-Term Memory networks (Hochreiter and Schmidhuber, 1997) with 200 units. The outputs of the LSTMs are concatenated along with the embeddings of the possessor and year, and the final output is calculated with a Softmax layer. Each LSTM has as its input a different chunk of text:

- The first LSTM takes as input the sequence of tokens in the sentence containing the *possessor* (top left in Figure 7). Each token is represented by the corresponding word embedding, and an additional embedding (also with 300 dimensions) for the possessor and all other tokens (there are only two unique additional embeddings, white and light gray in Figure 7). Unlike the word embedding from GloVe, the additional embeddings are initialized randomly and are updated during the training process. Our rationale to add the additional embeddings is to provide the LSTM with information to learn which tokens surrounding the *possessor* are more important.
- The second LSTM takes as input the sentence containing the *year* (top right in Figure 7). The input representation is very similar to the one used in the first LSTM, the only difference is that the additional embeddings (white and dark grey) indicate the *year* and any other token. Again, our rationale for the additional embeddings is to provide the LSTM with information to learn which tokens surrounding the *year* are more important.
- The third LSTM (bottom left in Figure 7) takes as input the Wikipedia article (i.e., the name of the possessee). The input words are represented with their GloVe embeddings.
- The fourth LSTM (bottom right) takes as input the section title from which the *possessor* and *year* were selected. The input words are also represented with their GloVe embeddings and no additional information. Our rationale is that some sections are less likely to contain valid possessors (e.g., *Cultural Impact* (low likelihood) vs. *Ownership and display* (high likelihood).

4.3 Results

Results obtained with the test set are provided in Table 4. F-measures are always higher for `no` than `yes`, but recall that only `yes` label allows us to extract valid possession relations.

Baselines. The majority baseline always predicts `no` label for all temporal tags (before, during and after, see percentages in Figure 4), thus it fails to extract any possession information.

SVMs. SVMs obtain higher-than-chance results, but F-scores with `yes` label are relatively low (before: 0.33, during: 0.31 and after: 0.44).

Neural Networks. The full neural network always outperforms SVMs, but the difference in F-score

---

[4] Available at `https://nlp.stanford.edu/projects/glove/`, file `glove/glove.6B.300.txt`

		Before			During			After		
		P	R	F	P	R	F	P	R	F
SVM	`yes`	0.47	0.26	0.33	0.33	0.29	0.31	0.42	0.46	0.44
	`no`	0.86	0.94	0.90	0.93	0.94	0.93	0.88	0.87	0.87
NN Full	`yes`	0.41	0.38	0.40	0.22	0.56	0.32	0.44	0.65	0.53
	`no`	0.87	0.89	0.88	0.95	0.80	0.87	0.92	0.82	0.87
NN Full - addt_embeds.	`yes`	0.40	0.40	0.40	0.21	0.46	0.29	0.35	0.75	0.47
	`no`	0.88	0.87	0.88	0.94	0.82	0.88	0.93	0.69	0.79
NN Full - LSTM$_{\text{article\_title}}$	`yes`	0.33	0.42	0.37	0.21	0.65	0.31	0.33	0.77	0.46
	`no`	0.87	0.82	0.85	0.95	0.74	0.83	0.93	0.66	0.77
NN Full - LSTM$_{\text{section\_title}}$	`yes`	0.41	0.50	0.45	0.25	0.46	0.33	0.38	0.80	0.48
	`no`	0.89	0.85	0.87	0.94	0.86	0.90	0.94	0.67	0.78

Table 4: Results obtained in the test set using SVMs (all features), and the neural network (the full architecture and after disabling some components). Recall that `yes` is the only label that enables us to extract possession relations, results with `no` are mostly irrelevant.

with `yes` label is minimal for *during* (before: 0.40 vs 0.33, +21.2%; during: 0.32 vs. 0.31, +3.2%; after: 0.53 vs. 0.44, +20.5%).

We also experimented with modifications of the full neural network to provide insights into which components are more useful. Specifically, we report results not using the additional embeddings for the possessor and year, and disabling the LSTMs for the article title and section title. Note that while `yes` F-scores for *during* barely vary regardless of the modifications to the network, we found interesting patterns for *before* and *after*. All F-scores discussed below are for `yes` label, the only label that is useful to extract possession relations.

- First, the additional embeddings for the possessor and year are beneficial for *after* (0.47 vs 0.53, +12.8%) and *during* (0.29 vs 0.32, +10.3%), and neutral for *before*. This leads to the conclusion that the LSTM learns the contexts surrounding the possessor and year successfully only for *after* and *during*. Note that the additional embeddings provides information regarding the position of the possessor and year within their sentences.
- Second, the LSTM that takes as its input the article title is beneficial for *before* (0.37 vs. 0.40, +8.1%) and *after* (0.46 vs 0.53, +15.2%), and barely for *during* (0.31 vs. 0.32, +3.1%). Thus we can conclude that the article title contains useful information to determine the existence of possession relations, and that pretrained word embeddings capture this information.
- Third, the LSTM that takes as its input the section title is beneficial for *after* (0.48 vs 0.53, +10.4%), detrimental for *before* (0.45 vs. 0.40, -11.1%) and barely detrimental for *during* (0.32 vs. 0.33, -3.0%). These results lead to the conclusion that the section title only contains useful information to determine possession relations in future years with respect to the years mentioned in the section.

5 Conclusions

Possession is an asymmetric semantic relation between two entities, where one entity (the possessee) *belongs* to the other entity (the possessor). Following theoretical works, we understand *belongs* in a broad sense, including physical, temporal, and control possessions.

In this paper, we track possession relations over time. Specifically, we work with Wikipedia articles about artworks, and extract their possessors as well as temporal information with respect to the years explicitly mentioned (before, during or after). We have presented an approach to extract potential possessors and pair them with years, and an annotation scheme to validate them. Overall inter-annotator agreement (Cohen's kappa) is 0.70, and the resulting corpus has substantial information regarding possessors over time: in 75% of articles we validate at least 14 (possessor, year) pairs, and in 50%, at least 30 pairs.

Experimental results show that the task can be automated, although we obtain moderate results. We present an LSTM ensemble that outperforms a traditional SVM. Disabling certain components of the full network show that the article title and section title benefit different temporal tags, and that the additional embeddings for the possessor and year are beneficial for *during*.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. https://www.tensorflow.org/.

A.Y Aikhenvald. 2013. Possession and ownership: a cross-linguistic perspective. In A.Y. Aikhenvald and R.M.W. Dixon, editors, *Possession and Ownership: A Cross-Linguistic Typology*, Oxford University Press, Oxford, chapter 1, pages 1–64.

A.Y. Aikhenvald and R.M.W. Dixon. 2012. *Possession and Ownership: A Cross-Linguistic Typology*. Explorations in Linguistic Typology. OUP Oxford.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Comput. Linguist.* 34(4):555–596.

Carmen Banea, Xi Chen, and Rada Mihalcea. 2016. Building a dataset for possessions identification in text. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France.

Carmen Banea and Rada Mihalcea. 2018. Possession identification in text. *Natural Language Engineering* page 122. https://doi.org/10.1017/S1351324918000062.

Dhivya Chinnappa and Eduardo Blanco. 2018. Mining possessions: Existence, type and temporal anchors. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, New Orleans, Louisiana, USA, pages 496–505.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

B. Heine. 1997. *Possession: Cognitive Sources, Forces, and Grammaticalization*. Cambridge Studies in Linguistics. Cambridge University Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1437–1447. http://www.aclweb.org/anthology/P14-1135.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60.

G.A. Miller and P.N. Johnson-Laird. 1976. *Language and perception*. Belknap Press. Belknap Press of Harvard University Press.

Preslav I. Nakov and Marti A. Hearst. 2013. Semantic interpretation of noun compounds using verbal and other paraphrases. *ACM Trans. Speech Lang. Process.* 10(3):13:1–13:51.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Portland, Oregon, USA, pages 1–27.

Nils Reimers, Nazanin Dehghani, and Iryna Gurevych. 2016. Temporal anchoring of events for the timebank corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages

2195–2204. http://www.aclweb.org/anthology/P16-1207.

L. Stassen. 2009. *Predicative Possession*. Oxford Studies in Typology and Linguistic Theory. OUP Oxford.

Jannik Strötgen and Michael Gertz. 2015. A baseline temporal tagger for all languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 541–547. http://aclweb.org/anthology/D15-1063.

Shiao Wei Tham. 2004. *Representing Possessive Predication: Semantic Dimensions and Pragmatic Bases*. Ph.D. thesis, Stanford University.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '10, pages 678–687.

Stephen Tratz and Eduard H. Hovy. 2013. Automatic interpretation of the english possessive. In *ACL (1)*. The Association for Computer Linguistics, pages 372–381.

Using Lexical Alignment and Referring Ability to Address Data Sparsity in Situated Dialog Reference Resolution

Todd Shore [†]
KTH Royal Institute of Technology
Speech, Music and Hearing
Stockholm, Sweden

Gabriel Skantze
KTH Royal Institute of Technology
Speech, Music and Hearing
Stockholm, Sweden
skantze@kth.se

Abstract

Referring to entities in situated dialog is a collaborative process, whereby interlocutors often expand, repair and/or replace referring expressions in an iterative process, converging on conceptual pacts of referring language use in doing so. Nevertheless, much work on exophoric reference resolution (i.e. resolution of references to entities outside of a given text) follows a literary model, whereby individual referring expressions are interpreted as unique identifiers of their referents given the state of the dialog the referring expression is initiated. In this paper, we address this collaborative nature to improve dialogic reference resolution in two ways: First, we trained a words-as-classifiers logistic regression model of word semantics and incrementally adapt the model to idiosyncratic language between dyad partners during evaluation of the dialog. We then used these semantic models to learn the general referring ability of each word, which is independent of referent features. These methods facilitate accurate automatic reference resolution in situated dialog without annotation of referring expressions, even with little background data.

1 Introduction

A crucial part of dialog situated in a physical environment is **exophoric references**, i.e. language used by the participants to make entities in the shared environment salient to each other for the purposes of communication (Poesio and Vieira, 1998). Several studies in exophoric reference resolution have investigated how **referential semantics** can be learned automatically via the relationship of a referent's features to the language referring to it (cf. Kennington et al., 2015; Shore and Skantze, 2017) or the state of the interaction a dialog is situated in (cf. Prasov and Chai, 2008; Iida et al., 2010), inferring a relationship between e.g. the word *red* and the individual features it refers to, e.g. a particular range of hue values.

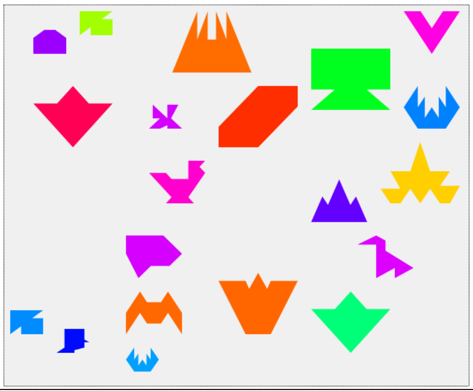

Speaker	Utterance
A	*it's the one to the left*
B	*the purple pinkish uh weird?*
A	*the pinkish one yeah it looks like an asteroid or something big*
B	*the very big?*
A	*yeah*

Figure 1: Collaborative reference in dialog situated in a reference communication task (cf. Krauss and Weinheimer, 1964).

Most works in exophoric reference resolution have assumed the identification of certain subsets of language known as **referring expressions** (REs) that have been either manually or automatically annotated (cf. Schutte et al., 2011; Meena et al., 2012; Zarrieß et al., 2016; Shore and Skantze, 2017). However, discerning REs from non-referring language in dialog is not trivial. For example, Figure 1 illustrates an interaction between two participants in a reference communication task like that of Krauss and Weinheimer (1964), whereby speaker **A** describes a particular

---

[†] Deceased 2 July 2018.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2288–2297
Brussels, Belgium, October 31 - November 4, 2018.

referent which must be resolved by speaker **B**. [1].

While REs are idealized as contiguous, single noun phrases (NPs) such as *the pinkish one*, reference in unrestricted, natural dialog is in fact a collaborative process to which both partners in a dyad contribute (Clark and Wilkes-Gibbs, 1986), and not all referring language (RL) is nominal, e.g. *big* in Figure 1. Both participants contribute RL in a cumulative fashion, but often no complete nominal RE is produced, e.g. *the big pinkish asteroid to the left*. Due to this, it is difficult to infer from syntax alone the **referring ability** (RA) of language, i.e. the overall ability of a subset of language to unambiguously refer to entities in discourse (Ariel, 1988; Reboul, 1997). In context (e.g. given the set of possible abstract shapes to choose from), it is easy to infer which words have the greatest RA, but without context this is more difficult. This makes recognizing "non-ideal" cases of RL difficult, as the boundary between RL and non-RL is often fuzzy.

Moreover, participants in dialog tend to develop so-called **conceptual pacts**, which means that they converge on commonly-used RL for unique referents in dialog (Brennan and Clark, 1996). As an example, they may repeatedly refer to a given entity as e.g. *the asteroid* even though *asteroid* may only rarely be used to refer to similar entities in the general population. Thus, RL varies less within a given dialog than across dialogs, and variation of RL has an inverse relationship with the length of the time two participants interact due to alignment of dialog participants' use of language (Clark and Wilkes-Gibbs, 1986; Garrod and Anderson, 1987; Brennan, 1996).

In this paper, we present two contributions to the automatic learning of referential semantics for reference resolution in situated dialog that address these problems: Firstly, we show the benefits of adapting models of RL semantics to a specific dialog as it progresses to accommodate the dyad's idiosyncratic use of RL. Secondly, we present a method for deriving a gradient (non-binary) measure of RA in situated dialog. Thus, instead of first identifying REs and then resolving which entity they refer to, we treat all language in the dialog as being more or less referential, and use this gradual measure together with the referential semantics to derive which entity is being talked about. Our assumption is that while the exact referential semantics of words vary greatly across dyads, the general ability of a given word to successfully refer to entities varies little across dyads. Thus, it should be possible to statistically measure the ability of a set of language to refer to entities in general, irrespective of the language's semantic content (e.g. the exact hue understood as *pink* by a dyad). This knowledge, combined with dialogic adaptation, facilitates accurate automatic reference resolution in situated dialog without annotation of REs, even with little background data.

2 Background

Both behavioral studies on reference resolution and RL and computational models thereof have illustrated the context-sensitive nature of reference resolution and RL and the gradient nature of RA.

2.1 Collaboration in Reference Resolution

Traditionally, reference resolution in dialog was analyzed using a literary model of reference, whereby individual REs are seen as unique identifiers of a referent as in written discourse, i.e. each RE is assumed to be "atomic" in its reference to a particular entity (Clark and Wilkes-Gibbs, 1986, 3). However, shortcomings in this approach have long since been identified (cf. Olson, 1970): REs often do not unambiguously identify their referent when initiated but rather comprise a larger process of collaborative reference resolution, whereby multiple dialog participants iteratively extend, repair and even replace REs initiated by themselves or others (Clark and Wilkes-Gibbs, 1986; Heeman and Hirst, 1995).

In Figure 1, speaker **A** initiates the RE *the one to the left* and immediately expands it in an episodic manner (Clark and Wilkes-Gibbs, 1986, 4, 17). Given a literary model of reference, they should have supplied exactly enough information to identify the referent and no more (*the big pinkish asteroid to the left*), adhering to Grice's (1975) maxim of quantity. However, RL can undergo not only expansion but also replacement: For color, speaker **B** proposes both *purple* and *pinkish*, but only *pinkish* is then accepted by **A**.

In contrast to a literary model of reference, a collaborative model represents reference resolution as a process of iteratively presenting RL to the other participant(s) in a dialog, which is then either accepted as being sufficient to identify a referent or rejected as insufficient (Clark and Wilkes-

---

[1] Examples are from the dataset of Shore et al. (2018)

Gibbs, 1986, 9). This more accurately models reference observed in spoken dialog.

2.2 Referring Language Syntax

Literary reference models fail to account not only for the collaborative nature of reference resolution but also for the syntactic structure of RL itself: Ideally, a reference is expressed linguistically as an NP, but this ideal does not hold in unrestricted dialog (Clark and Wilkes-Gibbs, 1986).

Since an RE cannot be defined as an atomic referring unit, a model of RL should ideally be able to measure the RA of any given set of language rather than simply classifying language as (part of) an RE in a binary decision, such as by using hand-crafted rules (cf. Shore and Skantze, 2017) or expert annotation (cf. Spanger et al., 2009; Kennington et al., 2015).

2.3 Modeling Referring Language

There have already been some efforts in automatic annotation of REs: For example, Schutte et al. (2011) algorithmically extracted RL as utterance(s) preceding a discrete event in a shared environment within a certain timeframe. However, one drawback to this method is that "the references must be contained in instructions that cause events involving the referents" and "it must be possible to automatically detect these events" (Schutte et al., 2011, 189). Thus, REs not referring to a detectable event cannot be detected in this manner. Moreover, not all language extracted is that of REs: For example, in the instruction *go through that door*, only half of the tokens constitute RL (*that door*). This means that this method must either be supplemented with additional methods to extract RL or tolerate a high noise-to-signal ratio.

Other approaches use language structure to infer RL, namely in parsing said language using a combination of statistical or rule-based methods. However, both entail that a solution be specialized for language specific to a given domain, such as for route-following instructions (Meena et al., 2012) or for a specific instructor-manipulator pair task (Shore and Skantze, 2017): Meena et al. (2012) used the highly-structured nature of route-following instructions to great effect, while Shore and Skantze (2017) used a phrase-structure parser pre-trained on out-of-domain data and supplemented it with hand-crafted rules to extract NPs according to the literary ideal of RL.

Finally, many works simply ignore the distinction between RL and non-RL and focus solely on learning reference resolution as a function of language and extra-linguistic knowledge such as entity features (cf. Kennington et al., 2015; Shore and Skantze, 2017), discourse and action history (cf. Iida et al., 2010), perception (cf. Matuszek et al., 2012) or gesture (cf. Matuszek et al., 2014). Although these methods improve the resolution of what RL *refers to*, they do not resolve what language *is RL*. Moreover, none of these works address the strong dyadic and dialogic entrainment effects on RL which include the formation of CPs, reinforcing the use of RL specific to a given dialog even if it diverges from population RL use.

3 Data Description

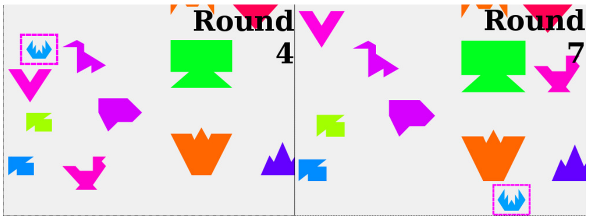

Speaker	Round	Utterance
B	4	*eh it looks like a blue crab sticking up his claws*
		…
A	7	*it's the same the the little crab again*

Figure 2: Example of repeated reference across rounds; the referent the participants have to collaboratively resolve is indicated here with a magenta square.

The data used is that of Shore et al. (2018), a set of $|D| = 42$ task-oriented dialogs (mean duration μ = 15:25 minutes, standard deviation SD = 1:13, total 647:35) in which one participant is an instructor referring to specific pieces on a shared game board which the other participant, the manipulator, must then attempt to resolve by selecting without the aid of extra-linguistic cues (see Figures 1–2): They sit at different locations and communicate solely through an audio channel. Upon successful selection, the piece moves to a random free place on the board and the participants alternate roles. This dataset is somewhat larger than that for similar tasks (cf. Iida et al., 2010; Matuszek et al., 2012; Malinowski and Fritz, 2014; Kennington et al., 2015). However, unlike in many other works, participants were allowed to refer to pieces in any way they wish and both were allowed to speak freely.

Each dialog $d \in D$ has $|R| = 20$ randomly-generated game pieces and is divided into individual game **rounds** $d \triangleq \langle d'_1 \dots d'_n \rangle$, in each of which $d' \triangleq (R, \hat{r}, T)$ a single entity is pre-selected by the game as the entity $\hat{r} \in R$ which must be successfully resolved for that round. Each dialog presents the same 20 referents aside from their changing position, so participants must also refer to pieces which have already been referenced before: After 40 rounds, all pieces are guaranteed to have been referred to and so every reference thereafter is a coreference (see Figure 2).

Each entity $r \in R$ has features representing shape, size, color and position during the given round. A sequence of tokens T was transcribed from the speech of both participants using Penn Treebank tokenization rules (Marcus et al., 1993). See the Supplementary Material for further information on the dataset.

4 Baseline

The reference resolution method used as a baseline was a **words-as-classifiers** (WaC) regression model (cf. Kennington et al., 2015). In this framework, an individual logistic regression model $p_t(r) \triangleq \sigma(w_t^T r + b_t)$ is trained for each token type t, predicting the probability of a given entity r being the token's TRUE referent $\hat{r}$, given the feature vector r representing shape, size, color and position (see the Supplementary Material for details). For example, if trained successfully, the model for the token "red" should be sensitive to the entity's hue, but not to its size. Common non-descriptive words such as "the" should not be sensitive to any of the entity's properties, yielding an output of 0.5 for all entities. The score of a given entity r being the referent $r = \hat{r}$ of a set of RL tokens T is defined as the normalized linear combination of the tokens' corresponding classifiers $p_t(r)$:

$$p'(r = \hat{r}, T) \triangleq \frac{1}{n} \sum_{t \in T}^{n} p_t(r) \tag{1}$$

For training, language in each round $(R, \hat{r}, T)$ is defined as a bag of words T referring to the referent $\hat{r}$. For each token $t \in T$, a training example is defined for the referent $\hat{r}$ (with a target score of 1) as well as for each non-referent entity $r \in R \setminus \hat{r}$ (with a target score of 0). To address model bias, the training example for $\hat{r}$ is weighted by its complement set size, $|R \setminus \hat{r}| = 19$.

Initial experiments showed that lemmatization did not affect the performance on our dataset. Thus, each inflected lexical form is considered a unique **word** (i.e., vocabulary item). Unlike Kennington et al. (2015), no smoothing was used, instead ignoring words of fewer than $\alpha \triangleq 3$ occurrences. The motivation for this is that a general out-of-vocabulary model is not expected to increase the performance, since it basically learns to ignore entity properties, similar to the models for common words such as "the". This was also confirmed in our initial experiments.

Note that all language from both the instructor and the manipulator in each round is used. This is unlike Kennington et al. (2015), who only used language from (manually annotated) REs. As argued above, REs cannot easily be identified in the type of dialog data we are addressing. This of course makes the task much more challenging, and the baseline performance can be expected to be lower than that reported in Kennington et al. (2015).

We did 42-fold cross-validation, in each fold using 40 dialogs for training as **background data**, one for testing and one for use as random data to compare the effects of dialog-specific data to (see Section 5 below). Each round in the test dialog is evaluated by the **reciprocal rank** (RR) of the referent $\hat{r}$ in the set of entities R ordered by their combined score for all word classifiers in the round $\sum_{t \in T} p_t(r)$, and its mean (MRR) is then calculated.

Statistic	Mean	SD	SEM
Rank	2.8060	3.2427	0.0566
RR	0.6892	0.3648	0.0064

Table 1: Baseline results for 42-fold cross-validation.

The cross-validation results for the baseline WaC model are shown in Table 1. As expected, this is indeed worse than e.g. Kennington et al. (2015)'s reported mean rank of 2.16 when only using speech features. The WaC model is nevertheless a simple and effective representation of referential semantics in domains where features for each individual referent can be easily represented (cf. Kennington and Schlangen, 2015). Still, it has two shortcomings: Firstly, it infers a static model of referential semantics which is good across dialogs but is suboptimal for language within dialogs due to effects of language alignment (Garrod and Anderson, 1987; Brennan, 1996; Brennan and Clark, 1996). Secondly, it encodes RA only indirectly: Given a large enough dataset, logistic regression for non-RL such as *okay now I'm ready* should have an even distribution between TRUE and FALSE classes, i.e. these classifiers should decide nothing. Conversely, strong RL such as *red* should entail strong relationships between certain features and decisions. However, due to the effects of idiosyncrasy and alignment on dialogic language, understanding low-frequency words is crucial despite that they cannot be conditioned for as well as can be done for high-frequency ones.

5 Dialogic Model Adaptation

We evaluated the benefits of adapting reference resolution parameters to the language of individual dialogs by initially conditioning WaC models on the training set as background data and then adapting the model during evaluation by retraining using data from previous states in the dialog being evaluated: The RR for the i^{th} round $(R, \hat{r}, T)_i$ is calculated using a model trained on both background data and **interaction data** defined as the rounds observed thus far in the given dialog $(R, \hat{r}, T)_{i'<i}$. The parameters for the logistic regression models representing individual words are optimized using quasi-Newton hybrid conjugate gradient descent from *Weka* v3.8.0 (Dai and Yuan, 2001; Frank et al., 2016). A ridge $\lambda = 100$ was used to avoid over-fitting of models for low-frequency words, tuned using cross-validation over the dataset (le Cessie and van Houwelingen, 1992). The same cross-validation method determined an optimal interaction data weight of 3 relative to background data, i.e. an observation in a given dialog is three times as relevant as one from the background data[2].

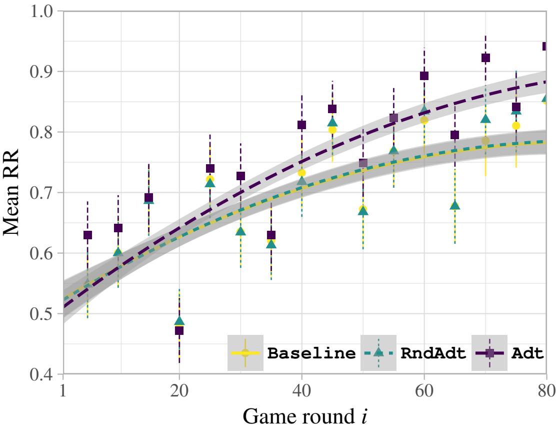

Figure 3: RR as a quadratic function of ROUND $i \cdot i^2$ for model adaptation using interaction data (CI 0.95; error bars are SEM; showing every 5th round).

Figure 3 compares the improvement of RR from adapting model parameters using dialog interaction data (`Adt`) to the `Baseline` as well as effects of adding data from a randomly-chosen round from another unseen dialog (`RndAdt`): The condition `RndAdt` is used to rule out the possibility that model fit improves simply due to more training data in general. We fit a linear mixed model with conditions `Adt`, `RndAdt`, `Wgt` and scaled `Tokens` as linear fixed effects and game round ordinality (ROUND) as a quadratic fixed effect: `Wgt` denotes weighting word classifiers by RA, which will be discussed in Section 6. `Tokens` denotes the number of word tokens produced by both speakers in the given round[3]. DYAD (the pair of participants in a given dialog) was included as a random intercept with a random slope for `Adt` and `Wgt`. We selected the best-fitting model using backwards selection with log-likelihood ratio tests: Starting from the maximally complex model (Barr et al., 2013), we first simplified the random structure and then removed fixed effects not contributing to fit. This showed that including `RndAdt` does not significantly improve fit ($\chi^2 = 0.00003, p = 0.99599$), meaning model fit improves from data specific to the given dialog and not merely from more training data.

We refit the final model using maximum-likelihood estimation with Satterthwaite approximation to degrees of freedom (see the Supplementary Material for details). Despite that RR correlates with ROUND i even for the baseline method due to dialogic lexical alignment (cf. Shore and Skantze, 2017; Shore et al., 2018), there is a significant improvement in RR from `Adt` ($B = 0.04882, t(40) = 7.65, p < 0.001$).

Since adding a small amount of data from a dialog significantly improves reference resolution for that dialog, dialogic reference resolution can be seen as a model adaptation problem, where in-domain data (that from the dialog being evaluated)

---

[2] Interaction data weight values tested were $1, 3, 5, 7, 10$.

[3] Adding the count of coreferences to a given referent as a fixed effect prevented model convergence when included with `Tokens`. Regardless, adding it in lieu of `Tokens` did not significantly improve fit ($\chi^2 = 3.7329,, p = 0.05335$).

is relatively sparse compared to out-of-domain data (that from other dialogs). This suggests that the effect of dyadic alignment on reference resolution is amplified: As ROUND i increases, not only is more data specific to the given dialog available, but the data observed becomes more homogeneous. Thus, the benefit of this method increases with time, as the ratio of interaction to background data increases.

6 Weighting by Referring Ability

While the method above facilitates the adaption of referential semantics models to dyad-specific language, not all language which is rare and/or observed in only one dyad has great RA: For example, in the dataset used, there were 19 observations of the word *awesome* but 15 of those were in a single dialog. Even when evaluating on that dialog, a classifier would be inferred from the $19 - 15 > \alpha$ remaining observations, and even adapting the word models with interaction data as done in Section 5 will only add noise since it only occurs as non-RL (e.g. *awesome good work*). Conversely, a word such as *piece* is semantically heavy in general English but is by itself a poor signifier of referents given the task at hand. So, we evaluated the benefit of weighting word classifiers by their RA in order to mitigate the effects of such spurious observations. To do this, we define the RA of a word t as the mean difference between the probability of the actual referent $\hat{r}$ being TRUE $p_t(\hat{r})$ and the mean probability for all other entities $R \setminus \hat{r}$ for every occurrence of the word in the training data:

$$\begin{aligned} w_t &\triangleq \frac{1}{n} \sum_{\substack{d \in D \\ (R,\hat{r},T) \in d}}^{n} p''_t(R,\hat{r},T) \\ p''_t(R,\hat{r},T) &\triangleq p'_t(\hat{r},T) - \frac{1}{m} \sum_{r \in R \setminus \hat{r}}^{m} p'_t(r,T) \quad (2) \\ p'_t(r,T) &\triangleq p_t(r) \sum_{i=1}^{|T|} [T_i = t] \end{aligned}$$

One alternative to this metric that we considered was the area under the receiver operating characteristic (ROC) curve (AUC). However, the metric above is more conservative in cases of word models with few observations by penalizing their score due to the logistic ridge used, thus putting more "trust" in word models with more observations; The AUC does not account for this directly.

Although this metric is derived from referential semantics learned for a specific domain, the WaC logistic regression model(s) encoding referential semantics are simple and thus can easily be re-trained for other domains. It can also be derived from other models of referential semantics.

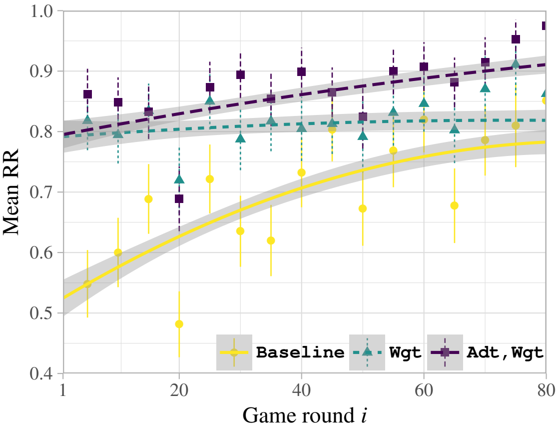

Figure 4: RR as a quadratic function of ROUND $i \cdot i^2$ for weighting model scores by RA $p_t(r) \cdot w_t$ (CI 0.95; error bars are SEM; showing every 5th round).

Figure 4 compares the improvement of RR from weighting each classifier $p_t(r)$ by its RA w_t (`Wgt`) to the `Baseline` and finally to that from combining adaptation from Section 5 with weighting (`Adt,Wgt`); Using the same linear mixed model described in Section 5, a significant improvement in RR was found for `Wgt` over the baseline ($B = 0.1314, t(39) = 11.79, p < 0.001$) although the effect weakens over time. However, this is likely not a weakness of the method but rather an effect of repeated reference on participants' RL use: With repeated reference, the length of RL reduces (Clark and Wilkes-Gibbs, 1986), meaning that the mean RA of each word increases due to fewer tokens of weak RA being uttered. Indeed, a significant interaction between `Round` and `Tokens` was found in their effects on RA (see Supplementary Material). Figure 5 shows a significant correlation of ROUND i and mean RA of all tokens for that round $\frac{1}{n} \sum_{t \in T_i}^{n} w_t$. Additionally, Figure 6 shows a significant inverse relationship with token count $|T_i|$. Since the referent $\hat{r}$ is chosen at random by the game, the amount of references to an entity increases with round ordinality, and so this corresponds with Clark and Wilkes-Gibbs (1986).

A qualitative assessment shows that vocabulary items with the great RA are typically nouns

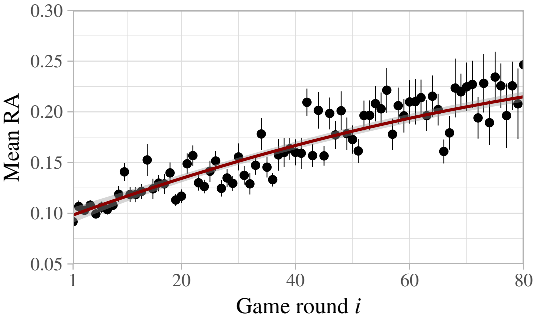

Figure 5: RA w_t for $t \in T_i$ as a quadratic function of ROUND $i \cdot i^2$ (CI 0.95; error bars are SEM).

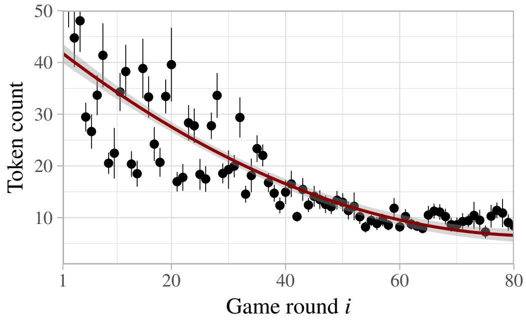

Figure 6: Token count $|T_i|$ as a quadratic function of ROUND $i \cdot i^2$ (CI 0.95; error bars are SEM).

strongly associated with the task at hand: The 31 words with greatest RA are all nouns referring to shapes. Despite this, however, great RA is not exclusive to nouns: In Table 2, *inside*, a preposition, is considered semantically lighter in general English than nouns are (Froud, 2001), but has RA greater than the mean ($\mu = 0.2424$, $SD = 0.1266$). On the other hand, the noun *color* has relatively little RA given the task at hand.

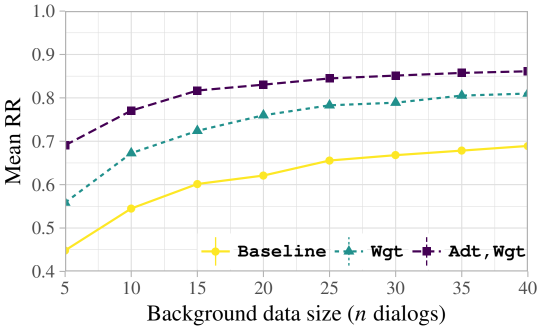

Figure 7: MRR per background training set size.

Word t	RA w_t	Count
K	0.7567	108
bat	0.6382	69
house	0.6374	153
chicken	0.6340	85
computer	0.6181	96
	…	
inside	0.3985	13
	…	
color	0.0291	195
	…	
's	0.0066	1593
it	0.0051	1731
okay	0.0048	1115
the	0.0040	2478

Table 2: Sample vocabulary items ordered by RA.

Moreover, including `Adt` and `Wgt` using dialog interaction data (`Adt,Wgt`) shows significant improvements over either alone: During model selection, including `Wgt` significantly improved fit ($\chi^2 = 61.425, p < 0.001$). This means that both methods can be used together and complement each other: Weighting is particularly beneficial for shorter interactions, where little in-domain interaction data is present, while adaptation provides greater benefit for longer interactions (cf. Figure 3). In fact, Figure 7 shows that `Wgt` has better MRR using only 12 randomly-chosen dialogs as background data than the `Baseline` does with 40, and adaptation and weighting together (`Adt,Wgt`) has better MRR with only 7. Figure 8 illustrates the effects of the two conditions on reference resolution in the task used for evaluation: The baseline classifier has a rank of 10 for the referent $\hat{r}$ out of $|R| = 20$ possible referents. In the baseline ***(A)***, the classifier for e.g. *color* has as much weight as e.g. *rectangle* although the former is not a useful signifier for the given task. When weighting by RA ***(B)***, however, the less-useful words contribute less to the total $\sum_{t \in T} (p_t(\hat{r}) \cdot w_t)$, improving rank to 5. Finally, when adapting the model with interaction data ***(C)***, models for semantically-heavy words like *violet* better fit the dyad's RL use, bringing rank to 1.

When both incrementally adapting semantic models with in-domain dialog data and weighting by RA, MRR for reference resolution was improved by 32.5% over the baseline (see Table 3).

7 Conclusion and Discussion

We have shown that it is possible to improve reference resolution for situated dialog by incrementally adapting word semantic model parameters to

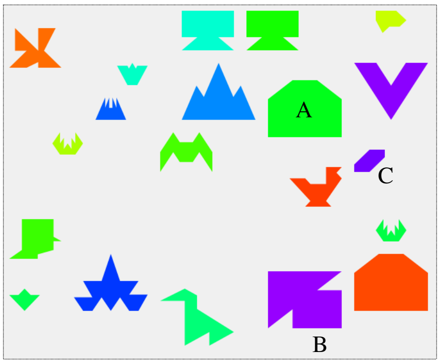

(A) Baseline (Target rank 10)

	the	slanted	rectangle	with	two	triangles	violet	in	color
	the	slanted	rectangle	with	two	triangles	violet	in	color
Score	0.51	0.50	0.74	0.31	0.42	0.23	0.41	0.53	0.31

(B) Weighting by RA (Target rank 5)

	the	slanted	rectangle	with	two	triangles	violet	in	color
			rectangle	with	two	triangles	violet	in	color
Score	0.51	0.50	0.74	0.31	0.42	0.23	0.41	0.53	0.31
RA	0.00	0.00	0.32	0.15	0.19	0.31	0.34	0.02	0.05

(C) Adapting and Weighting by RA (Target rank 1)

	the	slanted	rectangle	with	two	triangles	violet	in	color
		slanted	rectangle	with	two	triangles	violet	in	color
Score	0.51	0.64	0.80	0.31	0.60	0.63	0.70	0.53	0.46
RA	0.00	0.56	0.34	0.15	0.19	0.30	0.34	0.02	0.04

Figure 8: Reference resolution with different conditions: The TRUE referent $\hat{r}$ is labeled *C*. Word hue denotes semantic score $p_t(\hat{r})$ (green = 1.0, yellow = 0.5, red = 0.0). Saturation denotes RA w_t.

a given dialog in order to accommodate idiosyncratic language use by dyad partners, and the effect of the partners' own alignment makes this method even more beneficial over time. Additionally, we have defined a metric of word referring ability which is derived from a word's referential semantics in situated dialog but holds across individual dialogs despite dyadic variation in RL use. We showed that this metric can be used to automatically determine the usefulness of a given word for reference resolution, meaning that RE annotation is not necessary. Both of these aspects are beneficial to natural language understanding (NLU) for situated dialog due to the difficulty of acquiring data domain-appropriate data.

Model adaption using dialogic knowledge can be effective for improving NLU (cf. Riccardi and Gorin, 2000) despite that little work has been done in this regard specifically for reference resolution. Our experiments with model adaptation in Section 5 suggest that it may be beneficial to treat reference resolution in situated dialog as a model adaptation task, where a given dialog being evaluated is considered "in-domain" data and all other dialogs considered "out-of-domain" data. Moreover, due to the fact that dialog participants' use of RL converges over time (Garrod and Anderson, 1987; Brennan, 1996; Brennan and Clark, 1996), the task should adapt a pre-trained reference resolution model not only for a given dialog but also to the given state of that dialog; On the other hand, Iida et al. (2010) incorporate intra-dialogic knowledge but do not adapt to inter-dialogic effects.

Condition	*Rank*	SD$_{Rank}$	SEM$_{Rank}$
`Baseline`	2.8060	3.2427	0.0566
`Adt`	2.4224	2.8614	0.0499
`Wgt`	1.9373	2.3288	0.0406
`Adt,Wgt`	1.5693	1.6685	0.0291
Condition	***RR***	**SD$_{RR}$**	**SEM$_{RR}$**
`Baseline`	0.6892	0.3648	0.0064
`Adt`	0.7372	0.3470	0.0061
`Wgt`	0.8099	0.3116	0.0054
`Adt,Wgt`	0.8613	0.2686	0.0047

Table 3: Overall results for conditions evaluated.

Lastly, weighting by RA w_t as derived from logistic word classifier scores $p_t(r)$ in Section 6 was shown to be effective and can be easily inferred from data. However, this inaccurately assumes inter-word independence, since it does not encode a word's context: For example, the RA of *not* was 0.0638, which is relatively low. While it is a poor signifier in itself, it reverses the polarity of the predicate it modifies. For example, in *it's the baby blue K the light one not the dark one*, the NP *the dark one* should in fact have negative RA: Entities with a low semantic score $\sum_{t \in \langle \mathit{the,dark,one} \rangle} p_t(r)$ should in fact be preferred over those those with a high score. This could be addressed via structural prediction (e.g. conditional random fields or neural networks) or even higher-order n-grams, but these methods cannot be easily utilized given the typically small size of situated dialog datasets.

Acknowledgments

This work is supported by the SSF (Swedish Foundation for Strategic Research) project COIN. Plots were made with *ggplot2* v2.2.1 (Wickham, 2009). The authors are grateful for Zofia Malisz's help with model selection.

References

Mira Ariel. 1988. Referring and accessibility. *Journal of Linguistics*, 24(1):65–87.

Dale J. Barr, Roger Levy, Christoph Scheepers, and Harry J. Tily. 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3):255–278.

Susan E. Brennan. 1996. Lexical entrainment in spontaneous dialog. In *Proceedings of the International Symposium on Spoken Dialogue (ISSD '96)*, pages 41–44, Philadelpha, PA, USA. Acoustical Society of Japan.

Susan E. Brennan and Herbert H. Clark. 1996. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22(6):1482–1493.

S. le Cessie and J. C. van Houwelingen. 1992. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201.

Herbert H. Clark and Deanna Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22(1):1–39.

Yu-Hong Dai and Ya-xiang Yuan. 2001. An efficient hybrid conjugate gradient method for unconstrained optimization. *Annals of Operations Research*, 103(1):33–47.

Eibe Frank, Mark A. Hall, and Ian H. Witten. 2016. The WEKA workbench. Online appendix for "Data mining: Practical machine learning tools and techniques", Morgan Kaufmann, fourth edition, 2016. Last accessed 21 February 2018.

Karen Froud. 2001. Prepositions and the lexical/functional divide: Aphasic evidence. *Lingua*, 111(1):1–28.

Simon Garrod and Anthony Anderson. 1987. Saying what you mean in dialogue: A study in conceptual and semantic co-ordination. *Cognition*, 27(2):181–218.

Herbert Paul Grice. 1975. Logic and conversation. In *Syntax and Semantics, Volume 3: Speech Acts*, pages 41–58. Academic Press, New York, NY, USA.

Peter A. Heeman and Graeme Hirst. 1995. Collaborating on referring expressions. *Computational Linguistics*, 21(3).

Ryu Iida, Syumpei Kobayashi, and Takenobu Tokunaga. 2010. Incorporating extra-linguistic information into reference resolution in collaborative task dialogue. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1259–1267, Uppsala, Sweden. Association for Computational Linguistics.

Casey Kennington, Livia Dia, and David Schlangen. 2015. A discriminative model for perceptually-grounded incremental reference resolution. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 195–205, London, England, UK. Association for Computational Linguistics.

Casey Kennington and David Schlangen. 2015. Simple learning and compositional application of perceptually grounded word meanings for incremental reference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 292–301, Beijing, China. Association for Computational Linguistics.

Robert M. Krauss and Sidney Weinheimer. 1964. Changes in reference phrases as a function of frequency of usage in social interaction: a preliminary study. *Psychonomic Science*, 1(1):113–114.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1682–1690. Curran Associates, Inc.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2).

Cynthia Matuszek, Liefeng Bo, Luke Zettlemoyer, and Dieter Fox. 2014. Learning from unscripted deictic gesture and language for human-robot interactions. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2556–2563, Palo Alto, CA, USA. AAAI Press.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1671–1678, New York, NY, USA. Omnipress.

Raveesh Meena, Gabriel Skantze, and Joakim Gustafson. 2012. A data-driven approach to understanding spoken route directions in human-robot dialogue. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association*, pages 226–229, Portland, OR, USA. International Speech Communication Association.

David Richard Olson. 1970. Language and thought: Aspects of a cognitive theory of semantics. *Psychological Review*, 77(4):257–273.

Massimo Poesio and Renata Vieira. 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216.

Zahar Prasov and Joyce Y. Chai. 2008. What's in a gaze?: The role of eye-gaze in reference resolution in multimodal conversational interfaces. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, IUI '08, pages 20–29, New York, NY, USA. ACM.

Anne Reboul. 1997. What (if anything) is accessibility? a relevance-oriented criticism of Ariel's accessibility theory of referring expressions. In John H. Connolly, Roel M. Vismans, Christopher S. Butler, and Richard A. Gatward, editors, *Discourse and Pragmatics in Functional Grammar*, pages 91–108. De Gruyter Mouton, Berlin, Germany and Boston, MA, USA.

Giuseppe Riccardi and Allen L. Gorin. 2000. Stochastic language adaptation over time and state in natural spoken dialog systems. *IEEE Transactions on Speech and Audio Processing*, 8(1):3–10.

Niels Schutte, John D. Kelleher, and Brian Mac Namee. 2011. Automatic annotation of referring expression in situated dialogues. *International Journal of Computational Linguistics and Applications*, 2(1-2):175–190.

Todd Shore, Theofronia Androulakaki, and Gabriel Skantze. 2018. KTH Tangrams: A dataset for research on alignment and conceptual pacts in task-oriented dialogue. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Todd Shore and Gabriel Skantze. 2017. Enhancing reference resolution in dialogue using participant feedback. In *Proceedings, GLU 2017 International Workshop on Grounding Language Understanding*, pages 78–82, Stockholm, Sweden.

Philipp Spanger, Yasuhara Masaaki, Iida Ryu, and Tokunaga Tokenobu. 2009. Using extra linguistic information for generating demonstrative pronouns in a situated collaboration task. In *Proceedings of the Workshop on the Production of Referring Expressions: Bridging the gap between computational and empirical approaches to reference*, Amsterdam, the Netherlands.

Hadley Wickham. 2009. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, New York, NY, USA.

Sina Zarrieß, Julian Hough, Casey Kennington, Ramesh Manuvinakurike, David DeVault, Raquel Fernández, and David Schlangen. 2016. PentoRef: A corpus of spoken references in task-oriented dialogues. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Subgoal Discovery for Hierarchical Dialogue Policy Learning

Da Tang⋆ **Xiujun Li**† **Jianfeng Gao**† **Chong Wang**‡ **Lihong Li**‡ **Tony Jebara**⋆
†Microsoft Research, Redmond, WA, USA
⋆Columbia University, NY, USA
‡Google Inc., Kirkland, WA, USA
{xiul,jfgao}@microsoft.com
{datang,jebara}@cs.columbia.edu
{chongw,lihong}@google.com

Abstract

Developing agents to engage in complex goal-oriented dialogues is challenging partly because the main learning signals are very sparse in long conversations. In this paper, we propose a divide-and-conquer approach that discovers and exploits the hidden structure of the task to enable efficient policy learning. First, given successful example dialogues, we propose the *Subgoal Discovery Network* (SDN) to divide a complex goal-oriented task into a set of simpler subgoals in an *unsupervised* fashion. We then use these subgoals to learn a multi-level policy by hierarchical reinforcement learning. We demonstrate our method by building a dialogue agent for the composite task of travel planning. Experiments with simulated and real users show that our approach performs competitively against a state-of-the-art method that requires human-defined subgoals. Moreover, we show that the learned subgoals are often human comprehensible.

1 Introduction

Consider we want to plan a trip to a distant city using a dialogue agent. The agent must make choices at each leg, e.g., whether to fly or to drive, whether to book a hotel. Each of these steps in turn involves making a sequence of decisions all the way down to *lower*-level actions. For example, to book a hotel involves identifying the location, specifying the check-in date and time, and negotiating the price etc.

The above process of the agent has a natural hierarchy: a top-level process selects which subgoal to complete, and a low-level process chooses primitive actions to accomplish the selected subgoal. Within the reinforcement learning (RL) paradigm, such a hierarchical decision making process can be formulated in the *options* framework (Sutton et al., 1999), where subgoals with their own reward functions are used to learn policies for achieving these subgoals. These learned policies are then used as temporally extended actions, or options, for solving the entire task.

Based on the options framework, researchers have developed dialogue agents for complex tasks, such as travel planning, using hierarchical reinforcement learning (HRL) (Cuayáhuitl et al., 2010). Recently, Peng et al. (2017b) showed that the use of subgoals mitigates the reward sparsity and leads to more effective exploration for dialogue policy learning. However, these subgoals need to be human-defined which limits the applicability of the approach in practice because the domain knowledge required to properly define subgoals is often not available in many cases.

In this paper, we propose a simple yet effective *Subgoal Discovery Network* (SDN) that discovers useful subgoals automatically for an RL-based dialogue agent. The SDN takes as input a collection of successful conversations, and identifies "hub" states as subgoals. Intuitively, a hub state is a region in the agent's state space that the agent tends to visit frequently on successful paths to a goal but not on unsuccessful paths. Given the discovered subgoals, HRL can be applied to learn a hierarchical dialogue policy which consists of (1) a top-level policy that selects among subgoals, and (2) a low-level policy that chooses primitive actions to achieve selected subgoals.

We present the first study of learning dialogue agents with automatically discovered subgoals. We demonstrate the effectiveness of our approach by building a composite task-completion dialogue agent for travel planning. Experiments with both simulated and real users show that an agent learned with discovered subgoals performs competitively against an agent learned using expert-defined subgoals, and significantly outperforms an agent learned without subgoals. We also find that

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2298–2309
Brussels, Belgium, October 31 - November 4, 2018.

the subgoals discovered by SDN are often human comprehensible.

2 Background

A goal-oriented dialogue can be formulated as a Markov decision process, or MDP (Levin et al., 2000), in which the agent interacts with its environment over a sequence of discrete steps. At each step $t \in \{0, 1, \ldots\}$, the agent observes the current state s_t of the conversation (Henderson, 2015; Mrkšić et al., 2017; Li et al., 2017), and chooses action a_t according to a policy π. Here, the action may be a natural-language sentence or a speech act, among others. Then, the agent receives a numerical reward r_t and switches to next state s_{t+1}. The process repeats until the dialogue terminates. The agent is to learn to choose *optimal* actions $\{a_t\}_{t=1,2,\ldots}$ so as to maximize the total *discounted* reward $r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots$, where $\gamma \in [0, 1]$ is a discount factor. This learning paradigm is known as reinforcement learning, or RL (Sutton and Barto, 1998).

When facing a complex task, it is often more efficient to divide it into multiple simpler subtasks, solve them, and combine the partial solutions into a full solution for the original task. Such an approach may be formalized as hierarchical RL (HRL) in the options framework (Sutton et al., 1999). An option can be understood as a subgoal, which consists of an initiation condition (when the subgoal can be triggered), an option policy to solve the subgoal, and a termination condition (when the subgoal is considered finished).

When subgoals are given, there exist effective RL algorithms to learn a hierarchical policy. A major open challenge is the automatic discovery of subgoals from data, the main innovation of this work is covered in the next section.

3 Subgoal Discovery for HRL

Figure 1 shows the overall workflow of our proposed method of using automatic subgoal discovery for HRL. First a dialogue session is divided into several segments. Then at the end of those segments (subgoals), we equip an intrinsic or extrinsic reward for the HRL algorithm to learn a hierarchical dialogue policy. Note that only the last segment has an extrinsic reward. The details of the segmentation algorithm and how to use subgoals for HRL are presented in Section 3.1 and Section 3.3.

Figure 1: The workflow for HRL with subgoal discovery. In addition to the extrinsic reward at the end of the dialogue session, HRL also uses intrinsic rewards induced by the subgoals (or the ends of dialogue segments). Section 3.2 details the reward design for HRL.

3.1 Subgoal Discovery Network

Assume that we have collected a set of successful state trajectories of a task, as shown in Figure 2. We want to find subgoal states, such as the three red states s_4, s_9 and s_{13}, which form the "hubs" of these trajectories. These hub states indicate the subgoals, and thus divide a state trajectory into several segments, each for an option[1].

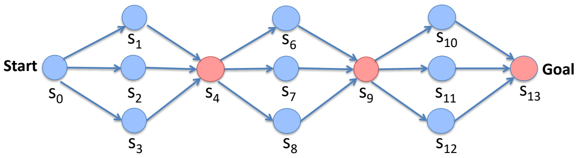

Figure 2: Illustration of "subgoals". Assuming that there are three state trajectories $(s_0, s_1, s_4, s_6, s_9, s_{10}, s_{13})$, $(s_0, s_2, s_4, s_7, s_9, s_{11}, s_{13})$ and $(s_0, s_3, s_4, s_8, s_9, s_{12}, s_{13})$. Then red states s_4, s_9, s_{13} could be good candidates for "subgoals".

Thus, discovering subgoals by identifying hubs in state trajectories is equivalent to segmenting state trajectories into options. In this work, we formulate subgoal discovery as a state trajectory segmentation problem, and address it using the Subgoal Discovery Network (SDN), inspired by the sequence segmentation model (Wang et al., 2017).

The SDN architecture. SDN repeats a two-stage process of generating a state trajectory segment, until a trajectory termination symbol is generated: first it uses an initial segment hidden state

---

[1] There are many ways of creating a new option $\langle I, \pi, \beta \rangle$ for a discovered subgoal state. For example, when a subgoal state is identified at time step t, we add to I the set of states visited by the agent from time $t - n$ to t, where n is a pre-set parameter. I is therefore the union of all such states over all the state trajectories. The termination condition β is set to 1 when the subgoal is reached or when the agent is no longer in I, and to 0 otherwise. In the deep RL setting where states are represented by continuous vectors, β is a probability whose value is proportional to the vector distance e.g., between current state and subgoal state.

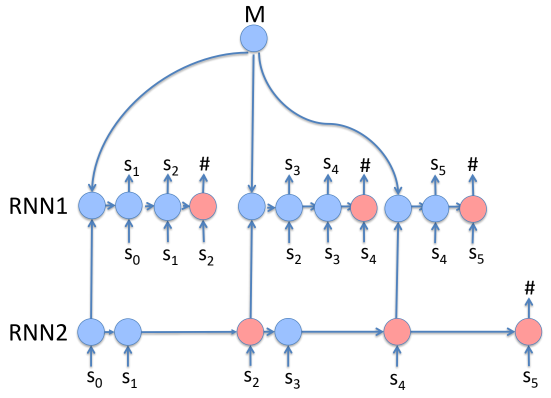

Figure 3: Illustration of SDN for state trajectory $(s_0, \ldots, s_5)$ with s_2, s_4 and s_5 as subgoals. Symbol # is the termination. The top-level RNN (RNN1) models segments and the low-level RNN (RNN2) provides information about previous states from RNN1. The embedding matrix M maps the outputs of RNN2 to low dimensional representations so as to be consistent with the input dimensionality of RNN1. Note that state s_5 is associated with two termination symbols #; one is for the termination of the last segment and the other is for the termination of the entire trajectory.

to start a new segment, or a trajectory termination symbol to terminate the trajectory, given all previous states; if the trajectory is not terminated, then keep generating the next state in this trajectory segment given previous states until a segment termination symbol is generated. We illustrated this process in Figure 3.

We model the likelihood of each segment using an RNN, denoted as RNN1. During the training, at each time step, RNN1 predicts the next state with the current state as input, until it reaches the option termination symbol #. Since different options are under different conditions, it is not plausible to apply a fixed initial input to each segment. Therefore, we use another RNN (RNN2) to encode all previous states to provide relevant information and we transform these information to low dimensional representations as the initial inputs for the RNN1 instances. This is based on the *causality* assumption of the options framework (Sutton et al., 1999) — the agent should be able to determine the next option given all previous information, and this should not depend on information related to any later state. The low dimensional representations are obtained via a global subgoal embedding matrix $M \in \mathbb{R}^{d \times D}$, where d and D are the dimensionality of RNN1's input layer and RNN2's output layer, respectively. Mathematically, if the output of RNN2 at time step t is o_t, then from time t the RNN1 instance has $M \cdot \mathrm{softmax}(o_t)$ as its initial input[2]. D is the number of subgoals we aim to learn. Ideally, the vector $\mathrm{softmax}(o_t)$ in a well-trained SDN is close to an one-hot vector. Therefore, $M \cdot \mathrm{softmax}(o_t)$ should be close to one column in M and we can view that M provides at most D different "embedding vectors" for RNN1 as inputs, indicating at most D different subgoals. Even in the case where $\mathrm{softmax}(o_t)$ is not close to any one-hot vector, choosing a small D helps avoid overfitting.

Segmentation likelihood. Given the state trajectory $(s_0, \ldots, s_5)$, assuming that s_2, s_4 and s_5 are the discovered subgoal states, we model the conditional likelihood of a proposed segmentation $\sigma = ((s_0, s_1, s_2), (s_2, s_3, s_4), (s_4, s_5))$ as $p(\sigma|s_0) = p((s_0, s_1, s_2)|s_0) \cdot p((s_2, s_3, s_4)|s_{0:2}) \cdot p((s_4, s_5)|s_{0:4})$, where each probability term $p(\cdot|s_{0:i})$ is based on an RNN1 instance. And for the whole trajectory $(s_0, \ldots, s_5)$, its likelihood is the sum over all possible segmentations.

Generally, for state trajectory $\boldsymbol{s} = (s_0, \ldots, s_T)$, we model its likelihood as follows[3]:

$$L_S(\boldsymbol{s}) = \sum_{\sigma \subseteq \mathcal{S}(\boldsymbol{s}), \mathrm{length}(\sigma) \leq S} \prod_{i=1}^{\mathrm{length}(\sigma)} p(\sigma_i | \tau(\sigma_{1:i})), \tag{1}$$

where $\mathcal{S}(\boldsymbol{s})$ is the set of all possible segmentations for the trajectory $\boldsymbol{s}$, σ_i denotes the i^{th} segment in the segmentation σ, and τ is the concatenation operator. S is an upper limit on the maximal number of segments. This parameter is important for learning subgoals in our setting since we usually prefer a small number of subgoals. This is different from Wang et al. (2017), where a maximum segment length is enforced.

We use maximum likelihood estimation with Eq. (1) for training. However, the number of possible segmentations is exponential in $\mathcal{S}(\boldsymbol{s})$ and the naive enumeration is intractable. Here, dynamic programming is employed to compute the likelihood in Eq. (1) efficiently: for a trajectory $\boldsymbol{s} = (s_0, \ldots, s_T)$, if we denote the sub-trajectory $(s_i, \ldots, s_t)$ of $\boldsymbol{s}$ as $\boldsymbol{s}_{i:t}$, then its likelihood follows

---

[2] $\mathrm{softmax}(o_t)_i = \exp(o_{t,i}) / \sum_{i'=1}^{D} \exp(o_{t,i'}) \in \mathbb{R}^D$ for $o_t = (o_{t,1}, \ldots, o_{t,D})$.

[3] For notation convenience, we include s_0 into the observational sequence, though s_0 is always conditioned upon.

the below recursion:

$$L_m(\boldsymbol{s}_{0:t}) = \begin{cases} \sum\limits_{i=0}^{t-1} L_{m-1}(\boldsymbol{s}_{0:i}) p(\boldsymbol{s}_{i:t}|\boldsymbol{s}_{0:i}), & m > 0, \\ I[t=0], & m = 0. \end{cases}$$

Here, $L_m(\boldsymbol{s}_{0:t})$ denotes the likelihood of sub-trajectory $\boldsymbol{s}_{0:t}$ with no more than m segments and $I[\cdot]$ is an indicator function. $p(\boldsymbol{s}_{i:t}|\boldsymbol{s}_{0:i})$ is the likelihood segment $\boldsymbol{s}_{i:t}$ given the previous history, where RNN1 models the segment and RNN2 models the history as shown in Figure 3. With this recursion, we can compute the likelihood $L_S(\boldsymbol{s})$ for the trajectory $\boldsymbol{s} = (s_0, \ldots, s_T)$ in $O(ST^2)$ time.

Learning algorithm. We denote θ^s as the model parameter including the parameters of the embedding matrix M, RNN1 and RNN2. We then parameterize the segment likelihood function as $p(\boldsymbol{s}_{i:t}|\boldsymbol{s}_{0:i}) = p(\boldsymbol{s}_{i:t}|\boldsymbol{s}_{0:i}; \theta^s)$, and the trajectory likelihood function as $L_m(\boldsymbol{s}_{0:t}) = L_m(\boldsymbol{s}_{0:t}; \theta^s)$.

Given a set of N state trajectories $(\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(N)})$, we optimize θ^s by minimizing the negative mean log-likelihood with L_2 regularization term $\frac{1}{2}\lambda||\theta^s||^2$ where $\lambda > 0$, using stochastic gradient descent:

$$\mathcal{L}_S(\theta^s, \lambda) = -\frac{1}{N}\sum_{i=1}^{N} \log L_S(\boldsymbol{s}^{(i)}, \theta^s) + \frac{1}{2}\lambda||\theta^s||^2. \quad (2)$$

Algorithm 1 outlines the training procedure for SDN using stochastic gradient descent.

Algorithm 1 Learning SDN

```
Input: A set of state trajectories $(\boldsymbol{s}_1, \ldots \boldsymbol{s}_N)$, the number of segments limit $S$, initial learning rate $\eta > 0$.
Initialize the SDN parameter $\theta^s$.
while not converged do
    Compute the gradient $\nabla_{\theta^s}\mathcal{L}_S(\theta^s, \lambda)$ of the loss $\mathcal{L}_S(\theta^s, \lambda)$ as in Eq. (2).
    Update $\theta^s \leftarrow \theta^s - \eta\nabla_{\theta^s}\mathcal{L}_S(\theta^s, \lambda)$.
    Update the learning rate $\eta$.
end while
```

3.2 Hierarchical Dialogue Policy Learning

Before describing how we use a trained SDN model for HRL, we first present a short review of HRL for a task-oriented dialogue system. Following the *options* framework (Sutton et al., 1999), assume that we have a state set $\mathcal{S}$, an option set $\mathcal{G}$ and a finite primitive action set $\mathcal{A}$.

The HRL approach we take learns two Q-functions (Peng et al., 2017b), parameterized by θ_e and θ_i, respectively:

- The top-level $Q^*(s, g; \theta_e)$ measures the maximum total discounted *extrinsic* reward received by choosing subgoal g in state s and then following an optimal policy. These extrinsic rewards are the objective to be maximized by the entire dialogue policy.
- The low-level $Q^*(s, a, g; \theta_i)$ measures the maximum total discounted intrinsic reward received to achieve a *given* subgoal g, by choosing action a in state s and then following an optimal option policy. These intrinsic rewards are used to learn an option policy to achieve a given subgoal.

Suppose we have a dialogue session of T turns: $\tau = (s_0, a_0, r_0, \ldots, s_T)$, which is segmented into a sequence of subgoals $g_0, g_1, \ldots \in \mathcal{G}$. Consider one of these subgoals g which starts and ends in steps t_0 and t_1, respectively.

The top-level Q-function is learned using Q-learning, by treating subgoals as temporally extended actions:

$$\theta_e \leftarrow \theta_e + \alpha \cdot (q - Q(s_t, g; \theta_e)) \cdot \nabla_{\theta_e} Q(s_t, g; \theta_e)\,,$$

where

$$q = \sum_{t=t_0}^{t_1-1} \gamma^{t-t_0} r_t^e + \gamma^{t_1-t_0} \max_{g' \in \mathcal{G}} Q(s_{t_1}, g'; \theta_e)\,,$$

and α is the step-size parameter, $\gamma \in [0, 1]$ is a discount factor. In the above expression of q, the first term refers to the total discounted reward during fulfillment of subgoal g, and the second to the maximum total discounted after g is fulfilled.

The low-level Q-function is learned in a similar way, and follows the standard Q-learning update, except that intrinsic rewards for subgoal g are used. Specifically, for $t = t_0, t_0 + 1, \ldots, t_1 - 1$:

$$\theta_i \leftarrow \theta_i + \alpha \cdot (q_t - Q(s_t, a_t, g; \theta_e)) \cdot \nabla_{\theta_i} Q(s_t, a_t, g; \theta_i)\,,$$

where

$$q_t = r_t^i + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a', g; \theta_i)\,.$$

Here, the intrinsic reward r_t^i is provided by the internal critic of dialogue manager. More details are in Appendix A.

In hierarchical policy learning, the combination of the extrinsic and intrinsic rewards is expected to help the agent to successfully accomplish a composite task as fast as possible while trying to avoid unnecessary subtask switches. Hence, we define the extrinsic and intrinsic rewards as follows:

Extrinsic Reward. Let L be the maximum number of turns of a dialogue, and K the number of subgoals. At the end of a dialogue, the agent receives a positive extrinsic reward of $2L$ for a success dialogue, or $-L$ for a failure dialogue; for each turn, the agent receives an extrinsic reward of -1 to encourage shorter dialogues.

Intrinsic Reward. When a subgoal terminates, the agent receives a positive intrinsic reward of $2L/K$ if a subgoal is completed successfully, or a negative intrinsic reward of -1 otherwise; for each turn, the agent receives an intrinsic reward -1 to encourage shorter dialogues.

3.3 Hierarchical Policy Learning with SDN

We use a trained SDN in HRL as follows. The agent starts from the initial state s_0, keeps sampling the output from the distribution related to the top-level RNN (RNN1) until a termination symbol # is generated, which indicates the agent reaches a subgoal. In this process, intrinsic rewards are generated as specified in the previous subsection. After # is generated, the agent selects a new option, and repeats this process.

This type of naive sampling may allow the option to terminate at some places with a low probability. To stabilize the HRL training, we introduce a threshold $p \in (0, 1)$, which directs the agent to terminate an option if and only if the probability of outputting # is at least p. We found this modification leads to better behavior of the HRL agent than the naive sampling method, since it normally has a smaller variance.

In the HRL training, the agent only uses the probability of outputting # to decide subgoal termination. Algorithm 2 outlines the full procedure of one episode for hierarchical dialogue policies with a trained SDN in the composite task-completion dialogue system.

4 Experiments and Results

We evaluate the proposed model on a travel planning scenario for composite task-oriented dialogues (Peng et al., 2017b). Over the exchange of a conversation, the agent gathers information about the user's intent before booking a trip. The environment then assesses a binary outcome (success or failure) at the end of the conversation, based on (1) whether a trip is booked, and (2) whether the trip satisfies the user's constraints.

Algorithm 2 HRL episode with a trained SDN

```
Input: A trained SDN $\mathcal{M}$, initial state $s_0$ of an episode, threshold $p$, the HRL agent $\mathcal{A}$.
Initialize an RNN2 instance $R_2$ with parameters from $\mathcal{M}$ and $s_0$ as the initial input.
Initialize an RNN1 instance $R_1$ with parameters from $\mathcal{M}$ and $M \cdot \text{softmax}(o_0^{\text{RNN2}})$ as the initial input, where $M$ is the embedding matrix (from $\mathcal{M}$) and $o_0^{\text{RNN2}}$ is the initial output of $R_2$.
Current state $s \leftarrow s_0$.
Select an option $o$ using the agent $\mathcal{A}$.
while Not reached the final goal do
    Select an action $a$ according to $s$ and $o$ using the agent $\mathcal{A}$. Get the reward $r$ and the next state $s'$ from the environment.
    Place $s'$ to $R_2$, denote $o_t^{\text{RNN2}}$ as $R_2$'s latest output and take $M \cdot \text{softmax}(o_t^{\text{RNN2}})$ as the $R_1$'s new input. Let $p_{s'}$ be the probability of outputting the termination symbol #.
    if $p_{s'} \geq p$ then
        Select a new option $o$ using the agent $\mathcal{A}$.
        Re-initialize $R_1$ using the latest output from $R_2$ and the embedding matrix $M$.
    end if
end while
```

Dataset. The raw dataset in our experiments is from a publicly available multi-domain dialogue corpus (El Asri et al., 2017). Following Peng et al. (2017b), a few changes were made to introduce dependencies among subtasks. For example, the hotel check-in date should be the same with the departure flight arrival date. The data was mainly used to create simulated users, and to build the knowledge bases for the subtasks of booking flights and reserving hotels.

User Simulator. In order to learn good policies, RL algorithms typically need an environment to interact with. In the dialogue research community, it is common to use simulated users for this purpose (Schatzmann et al., 2007; Li et al., 2017; Liu and Lane, 2017). In this work, we adapted a publicly available user simulator (Li et al., 2016) to the composite task-completion dialogue setting with the dataset described above. During training, the simulator provides the agent with an (extrinsic) reward signal at the end of the dialogue. A dialogue is considered to be successful only when a travel plan is booked successfully, and the information provided by the agent satisfies user's constraints.

Baseline Agents. We benchmarked the proposed agent (referred to as the *m-HRL Agent*) against three baseline agents:

- A *Rule Agent* uses a sophisticated, hand-crafted dialogue policy, which requests and informs a hand-picked subset of necessary slots, and then confirms with the user about the reserved trip before booking the flight and hotel.

- A *flat RL Agent* is trained with a standard deep reinforcement learning method, DQN (Mnih et al., 2015), which learns a flat dialogue policy using extrinsic rewards only.
- A *h-HRL Agent* is trained with hierarchical deep reinforcement learning (HDQN), which learns a hierarchical dialogue policy based on human-defined subgoals (Peng et al., 2017b).

Collecting State Trajectories. Recall that our subgoal discovery approach takes as input a set of state trajectories which lead to successful outcomes. In practice, one can collect a large set of successful state trajectories, either by asking human experts to demonstrate (e.g., in a call center), or by rolling out a reasonably good policy (e.g., a policy designed by human experts). In this paper, we obtain dialogue state trajectories from a rule-based agent which is handcrafted by a domain expert, the performance of this rule-based agent can achieve success rate of 32.2% as shown in Figure 4 and Table 1. We only collect the successful dialogue sessions from the roll-outs of the rule-based agent, and try to learn the subgoals from these dialogue state trajectories.

Experiment Settings. To train SDN, we use RMSProp (Tieleman and Hinton, 2012) to optimize the model parameters. For both RNN1 and RNN2, we use LSTM (Hochreiter and Schmidhuber, 1997) as hidden units and set the hidden size to 50. We set embedding matrix M with $D = 4$ columns. As we discussed in Section 3.1, D captures the maximum number of subgoals that the model is expected to learn. Again, to avoid SDN from learning many unnecessary subgoals, we only allow segmentation with at most $S = 4$ segments during subgoal training. The values for D and S are usually set to be a little bit larger than the expected number of subgoals (e.g., 2 or 3 for this task) since we expect a great proportion of the subgoals that SDN learns are useful, but not necessary for all of them. As long as SDN discovers useful subgoals that guide the agent to learn policies faster, it is beneficial for HRL training, even if some non-perfect subgoals are found. During the HRL training, we use the learned SDN to propose subgoal-completion queries. In our experiment, we set the maximum turn $L = 60$.

We collected $N = 1634$ successful, but imperfect, dialogue episodes from the rule-based agent in Table 1 and randomly choose 80% of these dialogue state trajectories for training SDN. The remaining 20% were used as a validation set.

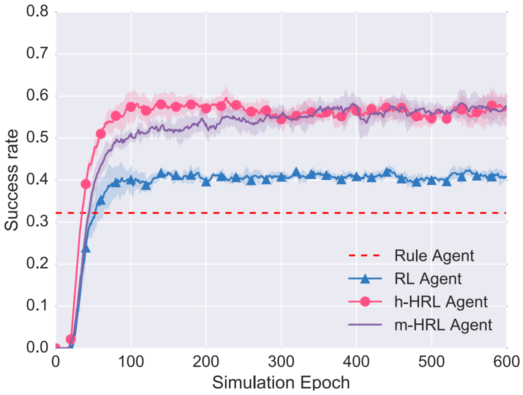

Figure 4: Learning curves of agents under simulation.

Agent	Success Rate	Turns	Reward
Rule	.3220	46.23	-24.02
RL	.4440	45.50	-1.834
h-HRL	.6485	44.23	35.32
m-HRL	.6455	44.85	34.77

Table 1: Performance of agents with simulated user.

As illustrated in Section 3.3, SDN starts a new RNN1 instance and issues a subgoal-completion query when the probability of outputting the termination symbol # is above a certain threshold p (as in Algorithm 2). In our experiment, p is set to be 0.2, which was manually picked according to the termination probability during SDN training.

In dialogue policy learning, for the baseline RL agent, we set the size of the hidden layer to 80. For the HRL agents, both top-level and low-level dialogue policies have a hidden layer size of 80. RMSprop was applied to optimize the parameters. We set the batch size to be 16. During training, we used ϵ-greedy strategy for exploration with annealing and set $\gamma = 0.95$. For each simulation epoch, we simulated 100 dialogues and stored these state transition tuples in the experience replay buffers. At the end of each simulation epoch, the model was updated with all the transition tuples in the buffers in a batch manner.

4.1 Simulated User Evaluation

In the composite task-completion dialogue scenario, we compared the proposed *m-HRL* agent

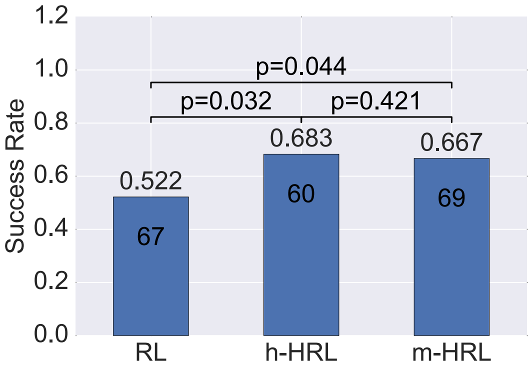

Figure 5: Performance of three agents tested with real users: success rate, number of dialogues and p-value are indicated on each bar (difference in mean is significant with $p < 0.05$).

with three baseline agents in terms of three metrics: success rate[4], average rewards and average turns per dialogue session.

Figure 4 shows the learning curves of all four agents trained against the simulated user. Each learning curve was averaged over 5 runs. Table 1 shows the test performance where each number was averaged over 5 runs and each run generated 2000 simulated dialogues. We find that the HRL agents generated higher success rates and needed fewer conversation turns to achieve the users' goals than the rule-based agent and the flat RL agent. The performance of the m-HRL agent is tied with that of the h-HRL agent, even though the latter requires high-quality subgoals designed by human experts.

4.2 Human Evaluation

We further evaluated the agents that were trained on simulated users against real users, who were recruited from the authors' organization. We conducted a study using the one RL agent and two HRL agents {RL, h-HRL, m-HRL}, and compared two pairs: {RL, m-HRL} and {h-HRL, m-HRL}. In each dialogue session, one agent was randomly selected from the pool to interact with a user. The user was *not* aware of which agent was selected to avoid systematic bias. The user was presented with a goal sampled from a user-goal corpus, then was instructed to converse with the agent to complete the given task. At the end of each dialogue session, the user was asked to give a rating on a scale from 1 to 5 based on the naturalness and coherence of the dialogue; here, 1 is the worst rating and 5 the best. In total, we collected 196 dialogue sessions from 10 human users.

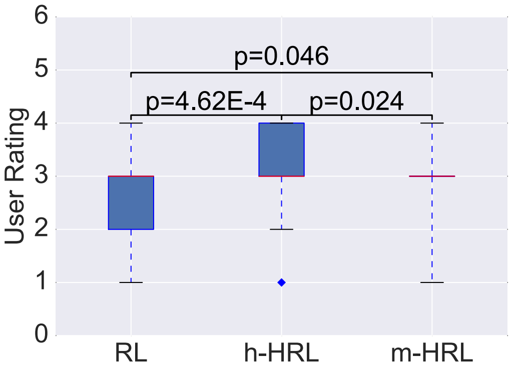

Figure 6: Distribution of user ratings for three agents in human evaluation

Figure 5 summarizes the performances of these agents against real users in terms of success rate. Figure 6 shows the distribution of user ratings for each agent. For these two metrics, both HRL agents were significantly better than the flat RL agent. Another interesting observation is that the m-HRL agent performs similarly to the h-HRL agent in terms of success rate in the real user study as shown on Figure 5. Meanwhile in Figure 6, the h-HRL agent is significantly better than m-HRL agent in terms of real user ratings. This may be caused by the probabilistic termination of subgoals: we used a threshold strategy to decide whether to terminate a subgoal. This could introduce variance so the agent might not behave reasonably compared with human-defined subgoals which terminate deterministically.

4.3 Subgoal Visualization

Table 2 shows the subgoals discovered by SDN in a sample dialogue by a rule-based agent interacting with the simulated user. The rule-based agent is equipped with a human-defined subtask structure, which always solves subtask *flight* (turns 1–15) before *hotel* (turns 16–23), as shown in the first column. At turn 10, the user starts to talk about *hotel* while the rule-based agent is still working on the pre-defined, unfinished *flight* subtask until subtask *flight* is finished at turn 15. At turn 16, the user switches to *hotel*, and so does the rule-based agent until the end of the dialogue. For this rule-based agent, the human-defined subgoal (*flight*) terminates at turn 15. Meanwhile, our SDN model detected two subgoals (except for the final goal):

---

[4] Success rate is the fraction of dialogues which accomplished the task successfully within the maximum turns.

h-Task	ID	Text
Flight	1	*agt*: Hi, may I know your departure city?
	2	*usr*: I depart from Fort Lauderdale.
	3	*agt*: And what is your destination city?
	4	*usr*: I need economy seat to Pittsburgh.
	5	*agt*: What is your departure date?
	6	*usr*: 2016-09-12, what time is available for the flight?
	7	*agt*: There is one flight at 21:00:00.
	8	*usr*: What is the price for that flight?
	9	*agt*: $423.
		Subgoal 1: A detected subgoal terminates!
	10	*usr*: I need to reserve a hotel for five people.
	11	*agt*: What is your return date for the trip?
	12	*usr*: 2016-09-18, I need to book 5 tickets for the flight.
	13	*agt*: What time do you prefer for return flight?
	14	*usr*: What time is available that day?
	15	*agt*: There is one flight departing at 11:00:00AM.
Hotel		Subgoal 2: A detected subgoal terminates!
	16	*usr*: What hotel is available?
	17	*agt*: There is one hotel called Winter Hill Inn available.
	18	*usr*: What is the price for hotel?
	19	*agt*: $410.
	20	*usr*: Great, could you book the trip for me?
	21	*agt*: Okay. Trip tickets are issued.
	22	*usr*: Thanks very much!
	23	*agt*: You are welcome!

Table 2: Discovered subgoals (except for the final goal) in a sample dialogue by a rule-based agent interacting with user simulator. The left column (h-Task) shows the human-defined subtasks for the rule-based agent. SDN detects two subgoals that terminate at turn 9 and 15 respectively. (h-Task: human-defined subtask, ID: turn ID, *agt*: Agent, *usr*: User)

one terminating at turn 9 (Subgoal 1), and another terminating at turn 15 (Subgoal 2). Subgoal 2 is consistent with the human-defined subgoal. Subgoal 1 is also reasonable since the user tries to switch to *hotel* at turn 10. In Appendix B, Table 3 shows a sample dialogue session by m-HRL agent interacting with a real user.

5 Related Work

Task-completion dialogue systems have attracted numerous research efforts, and there is growing interest in leveraging reinforcement learning for policy learning. One line of research is on single-domain task-completion dialogues with flat deep reinforcement learning algorithms such as DQN (Zhao and Eskenazi, 2016; Li et al., 2017; Peng et al., 2018), actor-critic (Peng et al., 2017a; Liu and Lane, 2017) and policy gradients (Williams et al., 2017; Liu et al., 2017). Another line of research addresses multi-domain dialogues where each domain is handled by a separate agent (Gašić et al., 2015; Gašić et al., 2015; Cuayáhuitl et al., 2016). Recently, Peng et al. (2017b) presented a composite task-completion dialogue system. Unlike multi-domain dialogue systems, composite tasks introduce inter-subtask constraints. As a result, the completion of a set of individual subtasks does *not* guarantee the solution of the entire task.

Cuayáhuitl et al. (2010) applied HRL to dialogue policy learning, although they focus on problems with a small state space. Later, Budzianowski et al. (2017) used HRL in multi-domain dialogue systems. Peng et al. (2017b) first presented an HRL agent with a global state tracker to learn the dialogue policy in the composite task-completion dialogue systems. All these works are built based on subgoals that were pre-defined with human domain knowledge for the specific tasks. The only job of the policy learner is to learn a hierarchical dialogue policy, which leaves the subgoal discovery problem unsolved. In addition to the applications in dialogue systems, subgoal is also widely studied in the linguistics research community (Allwood, 2000; Linell, 2009).

In the literature, researchers have proposed algorithms to automatically discovery subgoals for hierarchical RL. One large body of work is based on analyzing the spatial structure of the state transition graphs, by identifying bottleneck states or clusters, among others (Stolle and Precup, 2002; McGovern and Barto, 2001; Mannor et al., 2004; Şimşek et al., 2005; Entezari et al., 2011; Bacon, 2013). Another family of algorithms identifies commonalities of policies and extracts these partial policies as useful skills (Thrun and Schwartz, 1994; Pickett and Barto, 2002; Brunskill and Li, 2014). While similar in spirit to ours, these methods do not easily scale to continuous problems as in dialogue systems. More recently, researchers have proposed deep learning models to discover subgoals in continuous-state MDPs (Bacon et al., 2017; Machado et al., 2017; Vezhnevets et al., 2017). It would be interesting to see how effective they are for dialogue management.

Segmental structures are common in human languages. In the NLP community, some related research on segmentation includes word segmentation (Gao et al., 2005; Zhang et al., 2016) to divide the words into meaningful units. Alternatively, topic detection and tracking (Allan et al., 1998; Sun et al., 2007) segment a stream of data and identify stories or events in news or social text. In this work, we formulate subgoal discovery as a trajectory segmentation problem. Section 3.1 presents our approach to subgoal discovery which is inspired by a probabilistic sequence segmentation model (Wang et al., 2017).

6 Discussion and Conclusion

We have proposed the Subgoal Discovery Network to learn subgoals automatically in an unsupervised fashion without human domain knowledge. Based on the discovered subgoals, we learn the dialogue policy for complex task-completion dialogue agents using HRL. Our experiments with both simulated and real users on a composite task of travel planning, show that an agent trained with automatically discovered subgoals performs competitively against an agent with human-defined subgoals, and significantly outperforms an agent without subgoals. Through visualization, we find that SDN discovers reasonable, comprehensible subgoals given only a small amount of suboptimal but successful dialogue state trajectories.

These promising results suggest several directions for future research. First, we want to integrate subgoal discovery into dialogue policy learning rather than treat them as two separate processes. Second, we would like to extend SDN to identify multi-level hierarchical structures among subgoals so that we can handle more complex tasks than those studied in this paper. Third, we would like to generalize SDN to a wide range of complex goal-oriented tasks beyond dialogue, such as the particularly challenging Atari game of Montezuma's Revenge (Kulkarni et al., 2016).

Acknowledgments

We would like to thank the anonymous reviewers, members of the xlab at the University of Washington, and Chris Brockett, Michel Galley for their insightful comments on the work. Most of this work was done while DT, CW & LL were with Microsoft.

References

James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. 1998. Topic detection and tracking pilot study final report.

Jens Allwood. 2000. An activity based approach to pragmatics. *Abduction, belief and context in dialogue: Studies in computational pragmatics*, pages 47–80.

Pierre-Luc Bacon. 2013. On the bottleneck concept for options discovery: Theoretical underpinnings and extension in continuous state spaces. Master's thesis, McGill University.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 1726–1734.

Emma Brunskill and Lihong Li. 2014. PAC-inspired option discovery in lifelong reinforcement learning. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 316–324.

Pawel Budzianowski, Stefan Ultes, Pei-Hao Su, Nikola Mrksic, Tsung-Hsien Wen, Inigo Casanueva, Lina Rojas-Barahona, and Milica Gasic. 2017. Subdomain modelling for dialogue management with hierarchical reinforcement learning. *arXiv preprint arXiv:1706.06210*.

Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. 2010. Evaluation of a hierarchical reinforcement learning spoken dialogue system. *Computer Speech and Language*, 24(2):395–429.

Heriberto Cuayáhuitl, Seunghak Yu, Ashley Williamson, and Jacob Carse. 2016. Deep reinforcement learning for multi-domain dialogue systems. *arXiv preprint arXiv:1611.08675*.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv:1704.00057*. URL: https://datasets.maluuba.com/Frames.

Negin Entezari, Mohammad Ebrahim Shiri, and Parham Moradi. 2011. Subgoal discovery in reinforcement learning using local graph clustering.

Jianfeng Gao, Mu Li, Andi Wu, and Chang-Ning Huang. 2005. Chinese word segmentation and named entity recognition: A pragmatic approach. *Computational Linguistics*, 31(4):531–574.

Milica Gašić, Dongho Kim, Pirros Tsiakoulis, and Steve Young. 2015. Distributed dialogue policies for multi-domain statistical dialogue management. In *ICASSP 2015*, pages 5371–5375.

Milica Gašić, Nikola Mrkšić, Pei hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2015. Policy committee for adaptation in multi-domain spoken dialogue systems. In *ASRU*, pages 806–812. IEEE.

Matthew Henderson. 2015. Machine learning for dialog state tracking: A review. In *Proceedings of the 1st International Workshop on Machine Learning in Spoken Language Processing*.

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

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 3675–3683.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Xuijun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Per Linell. 2009. *Rethinking Language, Mind, and World Dialogically*. Information Age Publishing.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. *arXiv preprint arXiv:1709.06136*.

Bing Liu, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck. 2017. End-to-end optimization of task-oriented dialogue model with deep reinforcement learning. *arXiv preprint arXiv:1711.10712*.

Marlos C. Machado, Marc G. Bellemare, and Michael H. Bowling. 2017. A Laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2295–2304.

Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. 2004. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the 21st International Conference on Machine learning (ICML)*, pages 560–567. ACM.

Amy McGovern and Andrew G Barto. 2001. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning*, pages 361–368.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1777–1788.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Yun-Nung Chen, and Kam-Fai Wong. 2017a. Adversarial advantage actor-critic model for task-completion dialogue policy learning. *arXiv preprint arXiv:1710.11277*.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018. Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176*.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017b. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2221–2230.

Marc Pickett and Andrew G. Barto. 2002. Policy-Blocks: An algorithm for creating useful macro-actions in reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 506–513.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL 2007; Companion Volume, Short Papers*, pages 149–152. Association for Computational Linguistics.

Özgür Şimşek, Alicia P. Wolfe, and Andrew G. Barto. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 816–823.

Martin Stolle and Doina Precup. 2002. Learning options in reinforcement learning. In *International Symposium on Abstraction, Reformulation, and Approximation*, pages 212–223. Springer.

Bingjun Sun, Prasenjit Mitra, C Lee Giles, John Yen, and Hongyuan Zha. 2007. Topic segmentation with shared topic detection and alignment of multiple documents. In *SIGIR 2007*, pages 199–206. ACM.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.

Sebastian Thrun and Anton Schwartz. 1994. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7 (NIPS)*, pages 385–392.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. FeUdal Networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3540–3549.

Chong Wang, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. 2017. Sequence modeling via segmentations. *arXiv preprint arXiv:1702.07463*.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *ACL (1)*.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*.

A Hierarchical Dialogue Policy Learning

This section provides more algorithmic details for Section 3.2. Again, assume a conversation of length T:

$$\tau = (s_0, a_0, r_0, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)\,.$$

Suppose an HRL agent segments the trajectory into a sequence of subgoals as $g_0, g_1, \ldots \in \mathcal{G}$, and the corresponding subgoal termination time steps as $t_{g_0}, t_{g_1}, \ldots \in \mathbb{N}^*$. Furthermore, denote the intrinsic reward at time step t by r_t^i. The top-level and low-level Q-functions satisfy the following Bellman equations:

$$\begin{aligned} Q^*(s,g) = \mathbb{E}\Big[& \sum_{i=t_{g_j}}^{t_{g_{j+1}}-1} \gamma^{i-t_{g_j}} r_{t+1}^{e} \\ & + \gamma^{t_{g_{j+1}}-t_{g_j}} \cdot \max_{g'\in\mathcal{G}} Q^*(s_{t_{g_{j+1}}}, g') \\ & | s_{t_{g_j}} = s, g_j = g\Big] \end{aligned}$$

and

$$\begin{aligned} Q^*(s,a,g) = \mathbb{E}\Big[& r_t^i + \gamma \cdot \max_{a'\in\mathcal{A}} Q_i^*(s_{t+1}, g_j, a') \\ & | s_t = s, g_j = g, a_t = a, \\ & t \in [t_{g_j}, t_{g_{j+1}})\Big]\,. \end{aligned}$$

Here $\gamma \in [0,1]$ is a discount factor, and the expectations are taken over the randomness of the reward and the state transition,

We use deep neural networks to approximate the two Q-value functions as $Q^*(s,a,g) \approx Q(s,a,g;\theta_i)$ and $Q^*(s,g) \approx Q(s,g;\theta_e)$. The parameters θ_i and θ_e are optimized to minimize the following quadratic loss functions:

$$\begin{aligned} L_i(\theta_i) =& \frac{1}{2|D^i|} \sum_{(s,a,g,s',r^i)\in\mathcal{D}^i} [(y^i - Q(s,a,g;\theta_i))^2] \\ y^i =& r^i + \gamma \cdot \max_{a'\in\mathcal{A}} Q_i(s', a', g; \theta_i) \end{aligned} \tag{3}$$

and

$$\begin{aligned} L_e(\theta_e) =& \frac{1}{2|D^e|} \sum_{(s,g,s',r^e)\in\mathcal{D}^e} [(y^e - Q(s,g;\theta_e))^2] \\ y^e =& r^e + \gamma \cdot \max_{g'\in\mathcal{G}} Q(s', g'; \theta_e)\,. \end{aligned} \tag{4}$$

Here, $\mathcal{D}^e$, $\mathcal{D}^i$ are the replay buffers storing dialogue experience for training top-level and low-level policies.

Optimization of parameters θ_i and θ_e can be done by stochastic gradient descent on the two loss functions in Equations (3) and (4). The gradients of the two loss functions w.r.t their parameters are

$$\begin{aligned} \nabla_{\theta_i} L_i = \frac{1}{|D^i|} \sum_{(s,a,g,s',r^i)\in\mathcal{D}^i} \Big[& \nabla_{\theta_i} Q(s,a,g;\theta_i) \cdot \\ & (y^i - Q_i(s,a,g;\theta_i))\Big] \end{aligned}$$

and

$$\begin{aligned} \nabla_{\theta_e} L_e = \frac{1}{|D^e|} \sum_{(s,g,s',r^e)\in\mathcal{D}^e} \Big[& \nabla_{\theta_e} Q(s,g;\theta_e) \cdot \\ & (y^e - Q_e(s,g;\theta_e))\Big]\,. \end{aligned}$$

To avoid overfitting, we also add L_2-regularization to the objective functions above.

B Sample Dialogue

Table 3: Sample dialogue by the m-HRL agent interacting with real user: bolded slots are the joint constraints between two subtasks. (*agt*: Agent, *usr*: User)

User Goal

reserve-hotel subtask:

{

"request_slots": {	"inform_slots": {
"hotel_price": "?"	**"hotel_date_checkin":"2016-09-22"**
"hotel_date_checkout": "?"	**"hotel_city": "Curitiba"**
"hotel_name": "?"	**"hotel_numberofpeople": "4"**
"hotel_amenity_wifi": "?"	}
}	

}

book-flight-ticket subtask:

{

"request_slots": {	"inform_slots": {
"price": "?"	"or_city": "Lima",
"return_time_dep": "?"	**"dst_city": "Curitiba"**,
"return_date_dep": "?"	**"numberofpeople": "4"**,
"depart_time_dep": "?"	**"depart_date_dep":"2016-09-22"**
"seat": "?"	}
}	

}

m-HRL agent

agt: Hi, may I know your departure city and destination city?
usr: I depart from Lima to Curitiba.
agt: What is your departure date?
usr: I am leaving on 2016-09-22.
agt: There is one flight available at 8:00:00AM.
usr: What date is available for return flight?
agt: There is one return flight on 2016-09-27.
usr: What is the time for the return flight?
agt: It is 00:00:00AM.
usr: What seat is available?
agt: Only Economy seat now.
usr: We need 4 tickets for the flight.
agt: Total is 1201 dollars for flight.

inform(subtask_complete): Subgoal reached!

usr: What hotel is available there?
agt: There is one hotel called Starlight.
usr: Is the wifi available for the hotel?
agt: Yes, it is free.
usr: What's the price for the hotel?
agt: It is 1541 dollars for hotel.
usr: Could you please book the trip for me?
agt: Okay. Trip tickets are issued.
usr: Thanks!
agt: Thanks!

Supervised Clustering of Questions into Intents for Dialog System Applications

Iryna Haponchyk[1], **Antonio Uva**[1]*,
Seunghak Yu[2], **Olga Uryupina**[1] and **Alessandro Moschitti**[3,1]
[1]DISI, University of Trento, Povo (TN), Italy
[2]Samsung Research, Seoul, Korea
[3]Amazon, Manhattan Beach, CA, USA
iryna.haponchyk@unitn.it, antonio.uva@unitn.it
seunghak.yu@samsung.com, olga.uryupina@unitn.it, amosch@amazon.com

Abstract

Modern automated dialog systems require complex dialog managers able to deal with user intent triggered by high-level semantic questions. In this paper, we propose a model for automatically clustering questions into user intents to help the design tasks. Since questions are short texts, uncovering their semantics to group them together can be very challenging. We approach the problem by using powerful semantic classifiers from question duplicate/matching research along with a novel idea of supervised clustering methods based on structured output. We test our approach on two intent clustering corpora, showing an impressive improvement over previous methods for two languages/domains.

1 Introduction

The recent years have seen a resurgent interest for dialog systems, ranging from help desks and more complex task-based dialog to general purpose conversational agents, e.g., Alexa, Cortana or Siri. All these different application scenarios show that users expect to formulate complex information needs in natural language, with no limitation to their expressiveness. In other words, standard shallow semantic parsing based on concept segmentation and labeling is no longer sufficient for dialog modeling in today's applications. In particular, when designing dialog managers, we need to consider *user intents* expressed by articulated natural language text.

Current solutions to the intent detection problem consist in manually analyzing user questions and creating a taxonomy of intents to be attached to the appropriate actions. For example, if several semantically similar/identical questions regard `BookFlight`, the designer will build a category. This is a rather costly, difficult and time consuming task, which can often prevent the fast prototyping of dialog systems even for small domains. Moreover, the effort is extremely task- and system-specific, with very limited possibilities of porting the intent model, at least partially, across platforms, domains and languages. For example, in the recent 2016/2017 Natural Language Understanding Benchmarks (Coucke et al., 2017), the organizers have evaluated built-in intents generated by the major dialog managers (Apple's SiriKit, Amazon's Alexa, Microsoft's Luis, Google's Dialogflow, and Snips.ai) against a rather small set of relatively generic intents (e.g., `GetWeather`). This involved a considerable effort on aligning different system outputs.

To our knowledge, no previous work has been dedicated to automatizing this task, mainly because the underlying problem, semantic question paraphrasing, is very challenging. However, recent initiatives for automatic question duplicate detection[1], question relatedness (Nakov et al., 2016, 2017) and semantic textual similarity (Agirre et al., 2012; Cer et al., 2017) have shown that current technology achieves good accuracy in matching short text expressing similar semantics.

In this paper, we propose a model for automatically clustering questions into user intent categories, which can help the design of dialog systems. Our approach aims at overcoming the difficulty of providing a unique definition of intent from either a theoretical or practical perspective. Collaborating with stakeholders, we observe that it is very challenging to capture the intent property that can optimize dialogue/chatbot engineering work in terms of design and effort. The important aspect of our approach is that given a notion of intent, explicitly provided by annotated data, our

---

* The first two authors contributed equally to this manuscript.

[1] Quora: https://www.kaggle.com/c/quora-question-pairs

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2310–2321
Brussels, Belgium, October 31 - November 4, 2018.

model can create clusterings driven by such intrinsic definition. This is one of our major contributions: providing an effective supervised clustering approach, which can learn definitions from examples.

Regarding the technical solution our approach is bridging (i) the state-of-the-art methods for question similarity/paraphrasing with (ii) powerful supervised clustering algorithms. The former are obtained by exploiting previous research, e.g., on Quora, whereas the latter, are obtained by capitalizing on the structured output machine learning methods used for coreference resolution (CR), e.g., (Yu and Joachims, 2009; Fernandes et al., 2014). The main difference with CR consists in substituting mentions and their vector representation with the one of entire questions. It should be noted that structured output methods require a representation of edges connecting questions, which we obtain from question similarity research.

To train our model, we define a clustering corpus by automatically deriving question clusters from the Quora data, complementing the available question pair annotation with the transitive closure of the semantic matching property. Additionally, since Quora data is noisy, we manually annotate a question sample, defining an intent hierarchical taxonomy. To test the applicability of our methodology across languages and domains, we run evaluation experiments on another intent-based corpus, a collection of FAQs for an Italian online service.

We evaluate our approach comparing it to the classical k-means using (i) tf-idf model, (ii) our learned question similarity, and (iii) the similarity with spectral clustering. The last two methods can be considered another relatively novel contribution. Our results show that our new structured output method for question clustering: (i) highly outperforms all its competitors with respect to the standard clustering accuracy/purity as well as the measures defined in the CR domain, (ii) provides clustering accuracy of 80% with respect to the original Quora annotation and a still valuable accuracy of 65% with respect to the intent classes designed by an expert in Dialog modeling. This is a promising result as our designer can only provide one of the many interpretations of the intent taxonomy, which can be effectively applied.

Finally, a particular strength of our approach lies in identifying new intents (singleton clusters). These unseen intents are clearly problematic for dialogue management and cannot be covered by traditional approaches (e.g., our unsupervised clustering baselines show much lower performance on singleton clusters).

2 Related Work

Intent is a key concept for building dialog systems and is therefore a central research topic in the area. In particular, recent general-purpose dialog systems have to rely on extensive intent modeling to be able to correctly analyze a wide variety of user queries. This has led to a considerable amount of research on data-driven intent modeling.

In particular, Xu et al. (2013) represent query's intent as trees and employ a procedure for mapping an NL query into a tree-structured intent. The problem of this approach is that a new set of intent trees is required for new domains. Kim et al. (2016); Celikyilmaz et al. (2011) use semi-supervised approaches with large amounts of unlabeled data to improve the accuracy in mapping user queries into intents. However, they still require a small amount of labeled data in order to learn a given intent. Chen et al. (2016a) train a Convolutional Deep Neural Network to jointly learn the representations of human intents and associated utterances. Chen et al. (2016b) propose feature-enriched matrix factorization to model open domain intents. This leverages knowledge from Wikipedia and Freebase to acquire information from unexplored domains according to new users' requests. Unfortunately, it also requires external knowledge bases to induce concepts appearing within the intents.

Approaching the same problem from the opposite direction, several studies investigate algorithms for automatic question clustering. Wen et al. (2001) propose to cluster together queries that lead to the same group of web pages that are frequently selected by users. Jeon et al. (2005) use machine translation to estimate word translation probabilities and retrieve similar questions from question archives. Li et al. (2008) try to infer the intent of unlabeled queries according to the proximity with respect to the labeled queries in a click graph. Beitzel et al. (2007) propose to automatically classify web queries from logs into a set of topics by using a combination of different techniques, either supervised or unsupervised. The extracted topics are further used for efficient web search. Deepak (2016) presents MiXKmeans, a variation of k-means algorithm, suited for clustering threads present on forums and Community Question Answering websites. However, most techniques use unsupervised clustering to group similar questions/queries, without modeling intents. In contrast, our study relies on supervised clustering to learn intent-based similarity.

Finally, our work is related to a large body of research on dialog acts (Stolcke et al., 2000; Kim et al., 2010; Chen et al., 2018): our low-level intent labels (Table 1) can be seen as very fine-grained dialog acts (Core and Allen, 1997; Bunt et al., 2010; Oraby et al., 2017).

However, our paper's objective is different as our goal is not to rigidly define intents and then exploit them to derive a semantic interpretation. We focus on two contributions: first, we aim at providing a tool to help implementing dialog managers such that the designer can more easily create categories from precomputed clusters. Note that having in mind hundreds of questions to create intent category from scratch is clearly an exigent task.

Second, our approach can dynamically cluster questions with the same semantics, without any concept annotation. Indeed, the important concepts will be learned from the training data, which constitutes a much simpler annotation task than the creation of ad hoc dialogue acts. In particular, this can help with domain-specific intents: the domain-level semantics will be learned from data with no need for advanced manual engineering.

3 Question Clustering Algorithms

In this paper, we explore techniques for clustering questions into user intent. To this end, an input set of questions Q undergoes splitting into subsets (clusters), $c_i = \{q^i_j\}_{j=1}^{N_i}$, where q^i_j is the j-th question in the cluster i of size N_i and $\bigsqcup_i c_i = Q$. Each c_i is assumed to contain questions with the same intent, i.e., to represent a distinct intent. Performing a clustering of a new question set Q in an unsupervised manner may generally be troublesome due to the lack of any information about the structure of Q and the target number of distinct intents in it. To overcome this issue, we learn a clustering function from data annotated with gold question clusters.

In this work, we pose the task as a supervised clustering problem following the formulation by Finley and Joachims (2005). Having the training examples of the form $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, where each input $\mathbf{x}_i$ is a set of elements of some nature and $\mathbf{y}_i$ – the corresponding gold standard clustering of such a set, the goal is to learn a predictor $h : X \rightarrow Y$, from the space of sets X to the space of clusterings Y. Supervised clustering has been shown particularly effective for the NLP task of coreference resolution (Yu and Joachims, 2009; Fernandes et al., 2014). These models learn to infer an optimal clustering $\mathbf{y}$ of an input set $\mathbf{x}$ in a structured way, i.e., as one output object optimizing a global scoring function $f : X \times Y \rightarrow \mathbb{R}$. This is different from local models, which aggregate multiple clustering decisions taken with respect to pairs of elements.

At the test time, the global models draw predictions by finding

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in Y}{argmax}\ f(\mathbf{x}, \mathbf{y}). \qquad (1)$$

In the following, we give the necessary details of the original approach of Yu and Joachims (2009) and its adaptation for clustering questions.

3.1 Structured Output Clustering

To facilitate the inference of the optimal clustering in Equation 1, Yu and Joachims represent clustering variables $\mathbf{y}$ using graph structures. For an input $\mathbf{x}$, they construct a fully-connected undirected graph G, whose nodes are elements x_i of the input $\mathbf{x}$ and edges are all the pairwise links between them (x_i, x_j).[2] Any spanning forest $\mathbf{h}$ on G straightforwardly translates into a clustering $\mathbf{y}$. The nodes, aka elements represented by them, in each connected component (spanning tree) of $\mathbf{h}$ are considered to belong to the same cluster. The authors incorporate the spanning forest structures $\mathbf{h}$ as latent variables and decompose the feature representation of the input-output pair $(\mathbf{x}, \mathbf{y})$, which is extended with $\mathbf{h}$, over the edges of $\mathbf{h}$:

$$\Phi(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \sum_{(x_i, x_j) \in \mathbf{h}} \phi(x_i, x_j). \qquad (2)$$

They employ Kruskal's spanning algorithm (Kruskal, 1956) to infer the optimal $\mathbf{h}$, and, respectively, $\mathbf{y}$.

A linear model $\mathbf{w}$ is trained using the Latent formulation of the Structural SVM learner (LSSVM),

---

[2] Note that we distinguish between boldface letters which we use to denote structural input/output variables, and normal font letters - for their rather elementary constituents, and simple variables.

to score the output clusterings according to the function $f(\mathbf{x},\mathbf{y},\mathbf{h}) = \mathbf{w} \cdot \Phi(\mathbf{x},\mathbf{y},\mathbf{h})$. In fact, $\mathbf{w}$ learns to score the edges since the structural feature vector decomposes over the edges. However, imposing a structure onto the output is supposed to produce a better $\mathbf{w}$, which we test in our experiments described in Section 5.

Alternatively, we employ the same structural model with another learning algorithm – latent structured perceptron (LSP) by Sun et al. (2009); Fernandes et al. (2014).

3.2 Pairwise question similarity classifier

Our intent clustering algorithm relies on the pairwise similarity between questions (edge score). To provide an accurate estimation of question-question similarity, we build upon an extensive state-of-the-art research in semantic similarity for short text, more specifically, on our previous work (Filice et al., 2016, 2017; Barrón-Cedeño et al., 2016; Da San Martino et al., 2016) solutions/features shown effective in the shared tasks by Nakov et al. (2016, 2017); Agirre et al. (2012, 2013).

In such work, the classifier is trained with SVMs, which learn a classification function $f : Q \times Q \rightarrow \{0,1\}$ on duplicate vs. non-duplicate pairs of questions belonging to the question set Q. The classifier score is used to decide if two questions in the dataset q_i and q_j are duplicate or not. We represent question pairs as vectors of similarity features derived between two questions.

Feature Vectors are built for questions pairs, (q_i, q_j), using a set of *text similarity* features that capture the relations between two questions. More specifically, we compute 20 similarity features $sim(q_i, q_j)$ using word n-grams ($n = [1, \ldots, 4]$), after stopword removal, greedy string tiling (Wise, 1996), longest common subsequences (Allison and Dix, 1986), Jaccard coefficient (Jaccard, 1901), word containment (Lyon et al., 2001), and cosine similarity.

4 Building Intent clusters

In this study, we rely on two datasets, providing some important insights on question semantics. None of them, however, fully annotates intents explicitly. Below we describe our approach for converting these resources into intent corpora, relying on an automatic procedure followed by a manual post-annotation step. Our intent corpora as well as the larger raw question clusters collections are available to the research community.[3]

4.1 Quora corpus

The original Quora task requires detecting whether two questions are semantically duplicate or not. The associated dataset contains over 404, 348 pairs of questions, posted by users on Quora website, labeled as a duplicate pair or not. For example, *How do you start a bakery?* and *How can one start a bakery business?* are duplicate, while *What are natural numbers?* and *What is a least natural number?* are not. Note that the coders label pairs in isolation, only having access to one pair to be labeled at a time on Quora website (answer base). The pairs to be labeled are not selected randomly. To make the task more challenging, as well as more useful for practical applications, the organizers only offer pairs of questions that are somewhat semantically related:

(3) q_1: How does an automobile works?
q_2: How does automobile R&D work?
(4) q_1: Will I lose weight if I fast ?
q_2: Why am I losing weight so fast ?
q_3: How can I lose my weight fast ?

In (3), the lexical material is very similar, yet the questions are rather distinct, as reflected in the Quora annotation. They also express very different user intents: while q_1 is a generic curiosity question about automobiles, q_2 is a practical request for information on R&D in the automotive industry. Example (4) shows why Quora duplicate detection task is very challenging and requires a very good level of NL Understanding: while these three questions are very similar on the surface level, they all convey distinct semantics.

4.1.1 Question clusters from Quora

Differently from the original task, in this work, we are interested in automatically acquiring intents from large question repositories. Given this, we need a corpus that contains clusters of questions annotated by their underlying intent. As a first step in this direction, we approximate intent clusters with the clusters of similar questions from Quora. These can be obtained by exploiting the pairwise annotation and relying on the transitivity property: for each pair q_1, q_2 annotated as a duplicate, we assign q_1 and q_2 to the same cluster;

---

[3] `https://ikernels-portal.disi.unitn.it/repository/intent-qa`

negative pairs (non-duplicate question) do not impact the clustering in any way.

This procedure has obvious limitations by design: (i) it will not give us any intent labels, only the clusters and (ii) it will not provide any hierarchy of intents, or any general/large intent categories. Still, it provides data for a large number of user-generated intents that fully manual annotation initiatives (e.g., Natural Language Understanding Benchmarks) cannot guarantee.

4.1.2 Manually annotating intent clusters

The use of Quora dataset to derive intent raises several potential issues: (i) no consistency is enforced across labels, (ii) duplicate or very similar Quora *answers* potentially pollute the annotation for their corresponding questions, (iii) specific decisions may depend on availability and granularity of the underlying answers, and (iv) the annotation of popular questions might be very spurious since the users have no access to all the other related questions. The first issue has also been noted by the Quora competition[4] participants, who found the same pairs of questions appearing several times in the training set with different labels. Moreover, we found numerous cases where the annotation does not respect the transitivity property:

(5) q_1: What are, if any, the medical benefits of fasting?
q_2: What are the benefits of water fasting?
q_3: What are the health benefits of fasting?

Here, the three independent coders have produced inconsistent labeling: although q_2 and q_3 are explicitly labeled as non-duplicate, they are both considered duplicates of q_1.

The second issue arises when the answer base contains (near-)duplicate entries. For example, the following two very similar questions are considered non-duplicates since they lead to two distinct answers:

(6) q_1: Which is better - DC or Marvel?
q_2: DC VS Marvel: which do you like more? [non-duplicate]

Note that this typically happens for rather popular questions that are therefore important to be analyzed correctly, either manually or automatically.

The third issue is extremely important for areas only partially covered by the answer DB. For example, for the set of questions, *Why is Saltwater Taffy candy imported in LOCATION?*, most LOCATIONs are covered by a generic answer, and all the corresponding questions are judged duplicates. However, some specific LOCATIONs, e.g., Fiji, have a dedicated answer and thus the corresponding questions form singleton clusters.

Finally, the annotation coherence problem arises for very popular areas covering a lot of closely related questions. Thus, more than 100 questions cover different aspects of *Weight Loss*. Since the coders do not have any access to all the questions on the same topic, the individual decisions are not coordinated, which leads to rather arbitrary partitioning of the area into clusters:

(7) **Gold Quora Cluster 1:**
How can I lose weight ?
What is the easiest way to lose weight faster ?
How can you lose weight quickly ?
How do I lose 7kgs in 2 weeks ?
What a great diet to lose weight fast and not make you hungry or keep on measuring portions ?
.. many more
Gold Quora Cluster 2:
How can I lose 3 kg in one week?
Gold Quora Cluster 3:
What are the good diets for weight loss ?
What is the best diet plan for weight loss?

To overcome these issues, we manually re-annotated a portion of the original Quora dataset with intent-based clusters. Starting from automatically induced clusters described above, we first re-assess the partitioning, correcting eventual mistakes and then assign intent-based labels to our clusters. Our labels are hierarchical, thus allowing for a better flexibility when designing dialog managers: the dialog flow/actions can be defined in terms of more generic (e.g., `Advice`) or more specific (e.g., `Advice-WeightLoss-diet`) intents, depending on different implementation considerations (query frequency for specific intents, overall importance for the application, difficulty of processing inter alia). To stay inline with the recent intent-based dialogue research, we also annotate *slots*, where applicable: entities external to the intent, yet playing an essential role for the correct semantic interpretation of the question. Table 1 shows an example of the intent-based cluster annotation.

4.2 FAQ: Hype Intent corpus

Our second corpus, Hype, allows for a more direct evaluation of the intent clustering algorithm. The data come from a set of questions asked by

---

[4] `https://www.kaggle.com/c/quora-question-pairs/discussion/30435`

Intent	Slots	Questions
Recommend-TourismCuisine	streetfood, Delhi	What are the best street food in Delhi ?
Recommend-TourismCuisine	streetfood, Delhi	What is the best street food in Delhi ?
Recommend-TourismCuisine	streetfood, Delhi	What are the best street foods in delhi ?
Recommend-TourismRestaurant	streetfood, Delhi	What are the best street food places of delhi ?

Table 1: Manually annotated intent clusters for Quora

users to a conversational agent, collected and manually processed for constructing a FAQ section for Hype—an online service that offers a credit card, a bank account number and an ibanking app to its customers. Unlike Quora, the questions are explicitly assigned to clusters by human annotators, and these clusters correspond to intents by construction. However, they do not have any informative labels and there is therefore no associated hierarchy. While this corpus provides very valuable data for our study, its main disadvantage is a very limited number of questions. Some questions are reported below:

(8) q_0: Cos'é HYPE? (What is HYPE?)
q_1: Volevo dei chiarimente di cos'é la app hype (I'd like to have more information about the hype app)
q_2: mi puó spiegare cose'é la app hype (could you please explain me what the hype app is?)
q_3: informazione applicazione hype (information about hype app)

At the current stage of our research, we use the FAQ/Hype corpus directly, with no automatic or manual adjustments as we did for Quora. We plan to annotate the FAQ categories with explicit intent labels in future work.

5 Experiments

We describe our comparative experiments on our clustering models on two corpora in two different languages.

5.1 Setup

We used two different corpora, described in Section 4:

The Quora Intent corpus contains 270, 146 and 212 clusters in the training, development and test sets. The clusters contain different numbers of questions, ranging from singletons to groups of 100+ questions. The singletons are the dominant group in the Quora dataset. This is probably due to the inclusion of non-duplicate questions that appear in the original Quora dataset. Overall, there are 1,334 questions distributed in 628 clusters (an average of 2.12 questions/cluster).

The FAQ/Hype corpus contains no small-size (< 3) clusters by construction since smaller clusters are typically not selected as FAQ entries. The largest groups of clusters are those of size 8 and 9. Overall, the FAQ Intent corpus contains 147 questions spread in 28 intent-based clusters, which correspond to an average of 5.25 questions/cluster.

Models To perform supervised clustering, we use: (i) the original implementation of the Latent SVM^{struct} [5] – LSSVM, and (ii) our implementation of the LSP algorithm based on the same clustering inference on undirected graphs using Kruskal's spanning algorithm.

We compare these approaches to two unsupervised clustering baselines: (i) spectral clustering (Ng et al., 2001), for which we employ the implementation from the *smile*[6] library, and (ii) relational k-means (Szalkai, 2013). The former implementation takes a matrix of pairwise similarities between data points as input, whereas the latter approach is a generalization of k-means to an arbitrary matrix of pairwise distances. Thus, they can be run on the scores relative to the question pairs (q_i, q_j).

We provide two variants of such scores: first, we run both the methods on the scores obtained from a binary pairwise similarity classifier, described in Section 3.2. Second, we run the clustering baselines on the tf-idf scores computed for the question pairs. Note that our use of the scores from a trained pairwise classifier introduces some supervision in standard unsupervised clustering approaches, originating new hybrid methods.

Parametrization LSSVM and LSP require the tuning of a regularization parameter, C, and of a specific loss parameter, r (penalty for adding an incorrect edge), which we select on the dev. set. We pick up C from $\{1.0, 10.0, 100.0, 1000.0\}$, and the r values from $\{0.1, 0.5, 1.0\}$. K-means and spectral clustering algorithms require the indication of the number of clusters k. In all of our

---

[5] `www.cs.cornell.edu/~cnyu/latentssvm/`
[6] `http://haifengl.github.io/smile/`

Model	Clustering				Pairwise Classification			
	Precision	Recall	F1	CEAF$_e$	Precision	Recall	F1	Accuracy
LSSVM	**80.16**	77.81	**78.96**	**63.68**	**43.74**	32.00	36.96	88.41
LSP	66.06	**91.64**	76.78	51.50	20.36	**76.85**	32.19	65.62
SVM + spectral clustering	72.06	62.40	66.89	47.04	28.07	3.52	6.26	88.80
SVM + k-means	70.76	66.58	68.60	53.87	31.03	7.92	12.62	88.35
tfidf + spectral clustering	72.06	62.92	67.18	52.96	33.90	4.36	7.72	**88.94**
tfidf + k-means	69.19	65.01	67.04	50.94	29.95	5.33	9.04	88.62
SVM					26.25	72.23	**38.50**	75.50

Table 2: Supervised vs. unsupervised clustering models and pairwise classification baselines on the test set, where the gold labels are from the original Quora annotation. Note that pairwise classification does not provide a good estimation of clustering accuracy.

experiments, we use the gold standard k of each example (clustering) as parameter to run the baseline approaches. This corresponds to comparing with an upperbound of the baselines.

Measures We compare the output clustering $\hat{\mathbf{y}} = \bigsqcup_j \hat{c}_j$ to the ground truth $\mathbf{y}^* = \bigsqcup_i c_i$, where c_i, in our case, are either the clusters obtained with transitive closure from Quora annotation or the manually annotated categories (see Section 4.1).

For evaluation purposes, we assign each cluster $\hat{c}_j$ to the most frequent gold class (cluster), i.e., $argmax_i|c_i \cap \hat{c}_j|$, and compute the average precision over the clustering as:

$$Precision = \frac{1}{N}\sum_{j=1}^{\hat{k}} max_i|c_i \cap \hat{c}_j|, \qquad (9)$$

where N is the number of questions to be clustered, and $\hat{k}$ is the number of output clusters. This number is exactly the standard clustering purity by Zhao and Karypis (2002). Since the purity is known to favor the clustering outputs with the large number of clusters, we interchange the roles of output and gold clusters, which gives us the clustering

$$Recall = \frac{1}{N}\sum_{j=1}^{k} max_i|\hat{c}_i \cap c_j|, \qquad (10)$$

where k is the number of gold standard clusters. We then compute F1 from the above measures.

The defined majority-class based clustering measure allows assigning more than one cluster to the same gold cluster. The coreference resolution metric CEAF (Luo, 2005; Cai and Strube, 2010) solves this issue by finding one-to-one alignment between the clusters in the output and in the ground truth, based on which the final score is computed. We use CEAF$_e$, the variant with the entity-based similarity, as an alternative evaluation measure.[7] Note that, although we split the data into samples, all the clustering measures we use, the majority-based, defined by equations 9 and 10, and CEAF, are computed over the whole test sets (not by averaging scores separately for each sample).

We evaluate the results as well in terms of the classification scores relative to the correctness of the models in detecting the pairs of questions with the same/different intent. This enables the comparison against the pairwise classification approaches and an evaluation of their impact. We compute the Precision, Recall, and Accuracy of question pairs with the same intent.

5.2 Experiments on Quora

Original question label-based evaluation: first, we test the models on the clustering data from the Quora corpus, derived as described in Section 4.1. We train LSSVM, LSP, and the SVM classifier on the training part. The results of all the models on the test set are depicted in Table 2. In terms of clustering accuracy (the left half of the table), the LSSVM approach outperforms all the clustering baselines, improving about 10 points the highest baseline model, i.e., SVM + k-means both in terms of F1 and CEAF. LSP, in this setting, shows a slightly worse F1 than LSSVM, producing a model with high recall.

To study the impact of the pairwise classifier, we consider all the pairs of questions predicted by the clustering approaches as belonging to the same cluster as positive and the rest – as negative. All the clustering models show high accuracy, superior to that shown by the SVM classifier, but this is mainly due to the fact that there are much more negative question pairs. Interestingly, only LSSVM, though, among all the clus-

---

[7] We used the version 8 of the official coreference scorer `conll.cemantix.org/2012/software.html`

Model	**Clustering**				**Pairwise Classification**			
	Precision	Recall	F1	CEAF$_e$	Precision	Recall	F1	Accuracy
LSSVM	**84.92**	51.76	64.32	49.72	33.24	7.86	12.71	78.07
LSP	71.36	**89.45**	**79.38**	**59.99**	37.22	**83.46**	**51.48**	68.03
SVM + spectral clustering	62.31	43.22	51.04	33.04	35.11	2.95	5.44	79.17
SVM + k-means	68.84	47.24	56.03	48.62	42.71	6.48	11.25	**79.23**
tfidf + spectral clustering	65.83	45.23	53.62	35.46	38.18	3.62	6.62	**79.23**
tfidf + k-means	65.83	47.24	55.00	38.49	29.31	9.43	14.26	76.98
SVM					**40.35**	62.33	48.99	73.62

Table 3: Supervised vs. unsupervised clustering models and pairwise classification baselines on the test set, where the gold labels are provided by the intent-based manual annotation on a portion of the test set.

tering model, approaches the classifier in terms of classification F1. However, the cluster accuracy depends on many factors in addition to the pairwise classification accuracy.

Intent-based evaluation In Table 3, we present the results obtained on the portion of the test set which we manually annotated with the intent clusters. Here, we apply the same LSSVM, LSP, and the SVM classifier models trained in the experiments of the previous paragraph. However, we recompute all the four unsupervised clustering baselines supplying them with the new k – the number of gold intent-based annotated clusters.

In spite of being trained on data with different style of annotation (clusters automatically derived form Quora annotation), which is potentially rather noisy, LSSVM is able to recover the new intent categories better than the baseline approaches in terms of all the clustering metrics. The difference from the closest unsupervised clustering approach, which is the same as in the previous experiment, is now reduced in terms of CEAF. However, the information about the number of clusters in the ground truth critically impacts on the accuracy. The LSP model scored the best with respect to the new annotation. The lower classification accuracy of LSSVM with respect to the pairwise classifier is expected as the cluster number changed notably with the new annotation.

5.3 Evaluation on the FAQ dataset

Due to the limited size of the FAQ HYPE dataset, we split it into two parts, each of which forms one sample. We use the one containing 19 out of 28 clusters for training and the other with the remaining 9 clusters – for testing. The training sample comprises 97 questions, while the test sample – 50. The plots in Figure 1 illustrate the performance of LSSVM and LSP in terms of the clustering F1 in confront to the clustering baselines. We run the k-means and the spectral clustering algorithms with k in the range $(1, 50)$, which covers all the possible values for the given test set size.

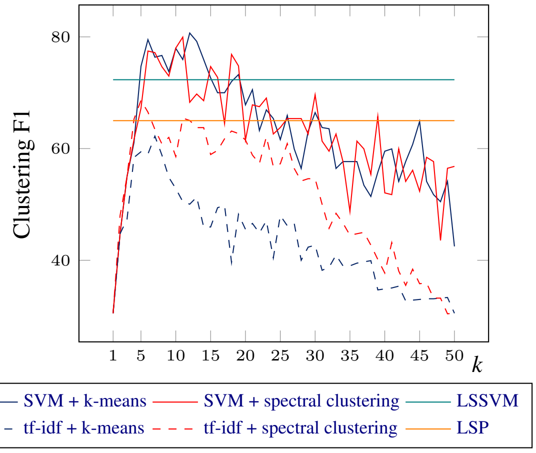

Figure 1: LSSVM and baseline clustering models; the latter vary with the cluster number k, on the FAQ HYPE test set.

LSSVM is better than the spectral clustering models with any k. k-means curves surpass LSSVM only in a narrow interval, showing high unstability. This suggests that guessing the k value in a realistic scenario in the absence of supervision does not seem an easy task. It should be also taken into consideration that we deal with very scarce training data. This also explains slightly insufficient accuracy of LSP compared to the k-means baseline.

5.4 Error Analysis and Discussion

As seen in tables 2 and 3, the structural output model consistently outperforms approaches based on spectral and k-means clustering on the Quora dataset. The most prominent improvement comes from singleton clusters: questions that are not duplicate with any other entries. Recall that the original dataset is constructed in such a way that singleton clusters are somewhat similar or related to existing material, but are still considered distinct by Quora annotators. LSSVM correctly recovers

71% of singleton clusters, whereas other methods perform much worse (5-30%). In the question-answering setting, singleton clusters correspond to novel questions that require setting up of a new entry in the answer base. Accurate recognition of singletons would allow for a timely allocation of resources to keep the answer base up-to-date and in line with incoming user requests.

Larger clusters are problematic for all the compared methods. Still, as evidenced by the CEAF score[8], the structural clustering is doing a better job at recovering non-singleton clusters. This mirrors our observations that even human annotators have difficulties correctly and consistently detecting duplicates in complex over-populated semantic areas (see Example 7) in the absence of the global context (e.g., list of all the related questions).

Finally, the clusters created by the LSSVM approach are more semantically related. Thus, 97% of all the suggested clusters contain questions with the same intent but, possibly, incorrect `slots`. For example, in the following question cluster:

(11) **gold cluster**
`Advice-Weightloss: fast,deadline`
q_1: How do I loose 50 lbs by Dec 2016?
q_2: How do I loose weight fast for operation ?
q_3: How can I lose 20 lbs super fast to audition for a small role in a movie ?
q_4: I want to lose weight for an event coming up in 2 weeks and I really don't care if I gain it back afterwords. What should I do ?

the user wants advice on losing their weight very fast by a specific deadline. LSSVM groups these questions with some others, more generic queries on fast weight loss (*How do I loose weight fast?*). This means that LSSVM captures the intent hierarchy well, providing meaningful clusters, although occasionally missing some important details. Other methods, on the contrary, form more poorly-related clusters (25-42% of clusters suggested by unsupervised approaches contain unrelated intents). Thus, the questions from Example (11) get grouped by other methods with such unrelated queries as *How is it to be in true love?* (spectral clustering over tf-idf).

Note that neither LSSVM nor unsupervised approaches have any access to the cluster labels and their hierarchy: in the training data, we only specify the clustering itself. Yet, by taking into account the global cluster structure, the LSSVM method can uncover the underlying hierarchy.

In the FAQ setting, most clusters are mid-size (5-9 questions). All the methods are doing a moderate job at recuperating the intent structure in this experiment. However, LSSVM shows better performance (cf. Section 5.3). Moreover, structural output is the only method that perfectly recuperates at least some clusters, e.g.:

(12) q_1: Non ricordo piú la password per accedere all'App (I don't remember the password for the App)
q_2: mi sono dimenticato la password (I forgot the password)
q_3: reimpostare la password (reset the password)
q_4: cambio password (change the password)

Here, LSSVM predicts the correct cluster exactly. K-means based approaches put $q_1 - q_4$ into the same cluster, however, they also merge them with *bloccare gli acquisti online (block the online purchases)*. Finally, spectral clustering does a poor job on this particular example, tearing either q_1 (tf-idf based spectral clustering) or q_2 (SVM pairwise classifier-based spectral clustering) apart and introducing a lot of spurious material.

6 Conclusion

We have proposed structured output methods fed with semantic question paraphrasing models to automatically extract user intents from question repositories. Our approach provides clustering accuracy of 80% with respect to the original Quora annotation and still valuable accuracy of 65% with respect one to of the many interpretations of question intent of our dataset, carried out by our expert in dialog modeling. This line of research looks promising as it can potentially simplify and speed up the work of Dialog Manager engineers. Although a deeper study is required to assess the benefits of our approach, the feedback of our designer clearly suggests that automatic clusters, even if were not perfect, simplify the annotation work. Several future research directions are enabled by our study, ranging from the use of neural clustering models to the application of our models to fast and semi-automatic prototyping of Dialog Systems. For this purpose, we make our data and software available to the research community.

Acknowledgements

This research has been supported by a grant from the Samsung Global Research Outreach Program: DiQuaVe Samsung GRO 2017.

---

[8] The reference scorer adopted by the coreference community discards singleton clusters.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43. Association for Computational Linguistics.

Lloyd Allison and Trevor Dix. 1986. A bit-string longest-common-subsequence algorithm. *Information Processing Letters*, 23(6):305–310.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad A. Al Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and question selection for question answering on Arabic and English fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, CA.

Steven M Beitzel, Eric C Jensen, David D Lewis, Abdur Chowdhury, and Ophir Frieder. 2007. Automatic classification of web queries using very large unlabeled query logs. *ACM Transactions on Information Systems (TOIS)*, 25(2):9.

Harry Bunt, Jan Alexandersson, Jean Carletta, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Kiyong Lee, Volha Petukhova, Andrei Popescu-Belis, Laurent Romary, Claudia Soria, and David Traum. 2010. Towards an iso standard for dialogue act annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Jie Cai and Michael Strube. 2010. Evaluation metrics for end-to-end coreference resolution systems. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '10, pages 28–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

Asli Celikyilmaz, Dilek Hakkani-Tür, and Gokhan Tur. 2011. Leveraging web query logs to learn user intent via bayesian latent variable model.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Association for Computational Linguistics.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016a. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6045–6049. IEEE.

Yun-Nung Chen, Ming Sun, Alexander I Rudnicky, and Anatole Gershman. 2016b. Unsupervised user intent modeling by feature-enriched matrix factorization. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6150–6154. IEEE.

Zheqian Chen, Rongqin Yang, Zhou Zhao, Deng Cai, and Xiaofei He. 2018. Dialogue act recognition via crf-attentive structured network. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 225–234, New York, NY, USA. ACM.

Mark G. Core and James F. Allen. 1997. Coding dialogs with the damsl annotation scheme. In *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Cambridge, MA.

Alice Coucke, Adrien Ball, Clement Delpuech, Clement Doumouro, Sylvain Raybaud, Thibault Gisselbrecht, and Joseph Dureau. 2017. Benchmarking natural language understanding systems: Google, Facebook, Microsoft, Amazon, and Snips.

Giovanni Da San Martino, Alberto Barrón-Cedeño, Salvatore Romeo, Antonio Uva, and Alessandro Moschitti. 2016. Learning to re-rank questions in community question answering using advanced features. In *Proceedings of the ACM Conference on Information and Knowledge Management*, CIKM '16, pages 1997–2000, Indianapolis, IN, USA.

P. Deepak. 2016. Mixkmeans: Clustering question-answer archives. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1585. Association for Computational Linguistics.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2014. Latent trees for coreference resolution. *Computational Linguistics*, 40(4):801–835.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning semantic relations between questions and answers. In *Proc. of the 10th Intl. Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, USA.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2017. KeLP at SemEval-2017 task 3: Learning pairwise patterns in community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, pages 327–334, Vancouver, Canada.

Thomas Finley and Thorsten Joachims. 2005. Supervised clustering with support vector machines. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 217–224, New York, NY, USA. ACM.

Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 84–90, New York, NY, USA. ACM.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010. Classifying dialogue acts in one-on-one live chats. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 862–871, Stroudsburg, PA, USA. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Scalable semi-supervised query classification using matrix sketching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 8–13.

Joseph Bernard Kruskal. 1956. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In *Proceedings of the American Mathematical Society, 7*.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2008. Learning query intent from regularized click graphs. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346. ACM.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Caroline Lyon, James Malcolm, and Bob Dickerson. 2001. Detecting short passages of similar text in large document collections. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 118–125, Pittsburgh, PA, USA.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48. Association for Computational Linguistics.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 525–545. Association for Computational Linguistics.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 849–856, Cambridge, MA, USA. MIT Press.

Shereen Oraby, Pritam Gundecha, Jalal Mahmud, Mansurul Bhuiyan, and Rama Akkiraju. 2017. "how may i help you?": Modeling twitter customer serviceconversations using fine-grained dialogue acts. In *Proceedings of the 22Nd International Conference on Intelligent User Interfaces*, IUI '17, pages 343–355, New York, NY, USA. ACM.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Comput. Linguist.*, 26(3):339–373.

Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009. Latent variable perceptron algorithm for structured classification. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1236–1242, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Balázs Szalkai. 2013. An implementation of the relational k-means algorithm. *CoRR*, abs/1304.6899.

Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. 2001. Clustering user queries of a search engine. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 162–168, New York, NY, USA. ACM.

Michael J Wise. 1996. Yap3: Improved detection of similarities in computer program and other texts. In *ACM SIGCSE Bulletin*, volume 28, pages 130–134. ACM.

Juan Xu, Qi Zhang, and Xuanjing Huang. 2013. Understanding the semantic intent of natural language query. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 552–560. Asian Federation of Natural Language Processing.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1169–1176, New York, NY, USA. ACM.

Ying Zhao and George Karypis. 2002. Criterion functions for document clustering: Experiments and analysis. Technical report.

Towards Exploiting Background Knowledge for Building Conversation Systems

Nikita Moghe[1,2], **Siddhartha Arora**[1], **Suman Banerjee**[1], **and Mitesh M. Khapra**[1,2]

[1]Department of Computer Science and Engineering, Indian Institute of Technology Madras
[2]Robert Bosch Centre for Data Science and AI (RBC-DSAI),
Indian Institute of Technology Madras
{nikitavam,sidarora,suman,miteshk}@cse.iitm.ac.in

Abstract

Existing dialog datasets contain a sequence of utterances and responses without any explicit background knowledge associated with them. This has resulted in the development of models which treat conversation as a sequence-to-sequence generation task (*i.e.*, given a sequence of utterances generate the response sequence). This is not only an overly simplistic view of conversation but it is also emphatically different from the way humans converse by heavily relying on their background knowledge about the topic (as opposed to simply relying on the previous sequence of utterances). For example, it is common for humans to (involuntarily) produce utterances which are copied or suitably modified from background articles they have read about the topic. To facilitate the development of such natural conversation models which mimic the human process of conversing, we create a new dataset containing movie chats wherein each response is explicitly generated by copying and/or modifying sentences from unstructured background knowledge such as plots, comments and reviews about the movie. We establish baseline results on this dataset (90K utterances from 9K conversations) using three different models: (i) pure generation based models which ignore the background knowledge (ii) generation based models which learn to copy information from the background knowledge when required and (iii) span prediction based models which predict the appropriate response span in the background knowledge.

1 Introduction

Background knowledge plays a very important role in human conversations. For example, to have a meaningful conversation about a movie, one uses their knowledge about the plot, reviews, comments and facts about the movie. A typical conversation involves recalling important points from this background knowledge and producing them appropriately in the context of the conversation. However, most existing large scale datasets (Lowe et al., 2015b; Ritter et al., 2010; Serban et al., 2016) simply contain a sequence of utterances and responses without any *explicit* background knowledge associated with them. This has led to the development of models which treat conversation as a simple sequence-to-sequence generation task and often produce output which is both syntactically incorrect and incoherent (off topic). To make conversations more coherent, there is an increasing interest in integrating structured and unstructured knowledge sources with neural conversation models. While there are already some works in this direction (Rojas-Barahona et al., 2017; Williams et al., 2016; Lowe et al., 2015a; Ghazvininejad et al., 2017) which try to integrate external knowledge sources with existing datasets, we believe that building new datasets where the utterances are *explicitly* linked to external background knowledge will further facilitate the development of such background aware conversation models.

With this motivation, we built a new background aware conversation dataset using crowdsourcing. Specifically, we asked workers to chat about a movie using structured and unstructured resources about the movie such as plots, reviews, comments, fact tables (see Figure 1). For every even numbered utterance, we asked the workers to consult the available background knowledge and try to construct a sentence which contains information from this background knowledge and is relevant in the current context of the conversation (akin to how humans recall things from their background knowledge and insert them appropriately in the conversation). For example, in Turn 2, Speaker 2 picked a sentence from the plot which is relevant to the current context of the conversation. Similarly, in Turn 3, Speaker 2 picked a

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2322–2332
Brussels, Belgium, October 31 - November 4, 2018.

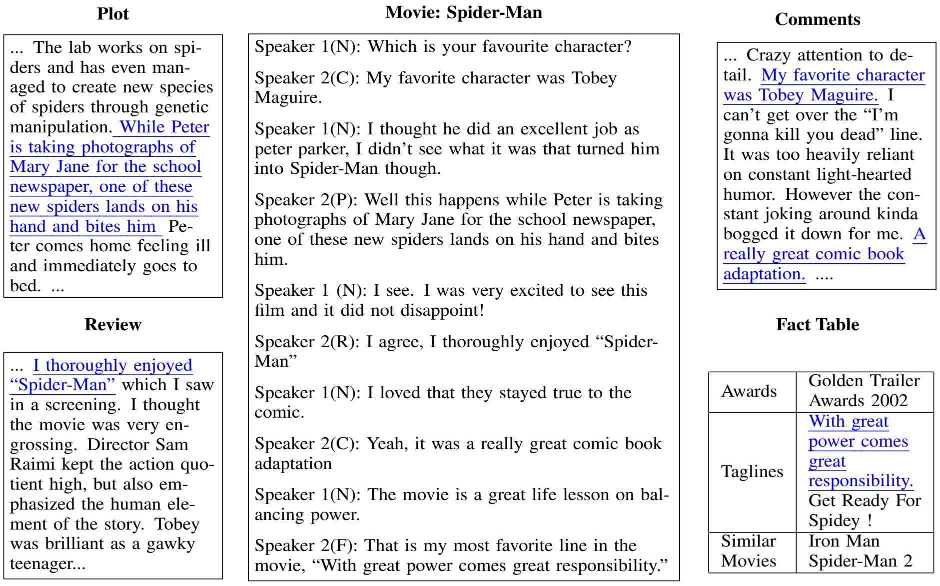

Plot

... The lab works on spiders and has even managed to create new species of spiders through genetic manipulation. While Peter is taking photographs of Mary Jane for the school newspaper, one of these new spiders lands on his hand and bites him Peter comes home feeling ill and immediately goes to bed. ...

Review

... I thoroughly enjoyed "Spider-Man" which I saw in a screening. I thought the movie was very engrossing. Director Sam Raimi kept the action quotient high, but also emphasized the human element of the story. Tobey was brilliant as a gawky teenager...

Movie: Spider-Man

Speaker 1(N): Which is your favourite character?

Speaker 2(C): My favorite character was Tobey Maguire.

Speaker 1(N): I thought he did an excellent job as peter parker, I didn't see what it was that turned him into Spider-Man though.

Speaker 2(P): Well this happens while Peter is taking photographs of Mary Jane for the school newspaper, one of these new spiders lands on his hand and bites him.

Speaker 1 (N): I see. I was very excited to see this film and it did not disappoint!

Speaker 2(R): I agree, I thoroughly enjoyed "Spider-Man"

Speaker 1(N): I loved that they stayed true to the comic.

Speaker 2(C): Yeah, it was a really great comic book adaptation

Speaker 1(N): The movie is a great life lesson on balancing power.

Speaker 2(F): That is my most favorite line in the movie, "With great power comes great responsibility."

Comments

... Crazy attention to detail. My favorite character was Tobey Maguire. I can't get over the "I'm gonna kill you dead" line. It was too heavily reliant on constant light-hearted humor. However the constant joking around kinda bogged it down for me. A really great comic book adaptation. ....

Fact Table

Awards	Golden Trailer Awards 2002
Taglines	With great power comes great responsibility. Get Ready For Spidey !
Similar Movies	Iron Man Spider-Man 2

Figure 1: A sample chat from our dataset which uses background resources. The chosen spans used in the conversation are shown in blue. The letters in the brackets denote the type of resource that was chosen - P, C ,R, F and N indicate **P**lot, **C**omments, **R**eview, **F**act Table and **N**one respectively.

sentence from the movie review. We also asked the workers to suitably modify the content picked from the background knowledge, if needed, so that the conversation remains coherent. We collected around 9K such conversations containing a total of 90K utterances pertaining to about 921 movies. These conversations along with the background resources will be made publicly available[1]. For every utterance, we also provide information about the exact span in the resource from which this utterance was created. Lastly note that unlike existing datasets, our *test set* contains multiple reference responses for each test context thereby facilitating better evaluation of conversation models. We believe that this dataset will allow the community to take a fresh look at conversation modeling and will lead to the development of models which can learn to exploit background knowledge to pick appropriate responses instead of generating responses from scratch. Such a conversation strategy which produces responses from background knowledge would be useful in various domains. For example, a troubleshooting bot could exploit the information available in manuals, reviews and previous bug reports about the software. Similarly, an e-commerce bot could exploit the rich information available in product descriptions, reviews, fact tables, *etc.* about the product. While the proposed dataset is domain specific, it serves as a good benchmark for developing creative background-knowledge-aware models which can then be ported to different domains by building similar datasets for other domains.

We establish some initial baselines using three different paradigms to demonstrate the various models that can be developed and evaluated using this dataset. For the sake of completeness, the first paradigm is a hierarchical variant of the sequence to sequence architecture which does not exploit any background knowledge. The second paradigm is the copy-and-generate paradigm wherein the model tries to copy text from the given resources whenever appropriate and generate it otherwise. The third paradigm borrows from the span prediction based models which are predominantly being used for Question Answering (QA). These baseline results along with the dataset would hopefully shape future research in the area of background aware conversation models.

2 Related Work

There has been an active interest in building datasets (Serban et al., 2015) for training dialog systems. Some of these datasets contain transcripts of human-bot conversations (Williams et al., 2013; Henderson et al., 2014a,b) while others are created using a fixed set of natural language patterns (Bordes and Weston, 2017; Dodge et al., 2016). The advent of deep learning created

---

[1] https://github.com/nikitacs16/Holl-E

interest in the construction of large-scale dialog datasets (Lowe et al., 2015b; Ritter et al., 2010; Sordoni et al., 2015) leading to the development of several end-to-end conversation systems (Shang et al., 2015; Vinyals and Le, 2015; Li et al., 2016; Serban et al., 2016) which treat dialog as a sequence generation task.

To make the output of these models more coherent, there is an increasing effort in integrating external background knowledge with these models. This is because human beings rely on background knowledge for conversations as well as other tasks (Schallert, 2002). There has been considerable work on incorporating background knowledge in the context of goal-oriented dialog datasets even before the advent of large-scale datasets for deep learning (Raux et al., 2005; Seneff et al., 1991) as well as in recent times (Rojas-Barahona et al., 2017; Williams et al., 2016; Eric et al., 2017) where datasets include small sized knowledge graphs as background knowledge. However, the conversations in these datasets are very templated and nowhere close to open conversations in specific domains such as the ones contained in our dataset.

Even in the case of open domain conversations, there are some works which have integrated external knowledge sources. Most of the entries in 2017 Amazon Alexa Prize (Ram et al., 2017) relied on background knowledge for meaningful response generation. Milabot (Serban et al., 2017a) and even the winning entry SoundingBoard (Liu et al., 2018) used Reddit pages, Amazon's Evi Service, and large databases like OMDB, Google Knowledge Graph and Wikidata as external knowledge. The submission named Eigen (Guss et al., 2017) used several dialog datasets and corpora belonging to related Natural Language Processing tasks to make their responses more informative. We refer the reader to (Ram et al., 2017) for detailed analysis of these systems. In the space of academic datasets, Lowe et al. (2015a) report results on the Ubuntu dataset using manpages as external knowledge whereas Ghazvininejad et al. (2017) use Foursquare tips as external knowledge for social media conversations. However, unlike our work both these works do not create a new dataset where the responses are explicitly linked to a knowledge source. The infusion of external knowledge in both these works is post facto (as opposed to our work where we take a bottom-up approach and explicitly create a dataset which allows exploitation of background knowledge). Additionally, existing large-scale datasets are noisy as they are extracted from online forums which are inherently noisy. In contrast, since we use crowdsourcing, the extent of noise is reduced since there are humans in the loop who were explicitly instructed to use only clean sentences from the external knowledge sources.

We would also like to mention some existing works such as (He et al., 2017; Lewis et al., 2017; Krause et al., 2017) which have used crowdsourcing for creating conversation datasets. In fact, our data collection method is inspired by the work of Krause et al. (2017) where the authors use self-dialogs to collect conversation data about movies, music and sports. They are referred to as self-dialogs because the same worker plays the role of both parties in the conversation. However, our work differs from Krause et al. (2017) as we provide explicit background knowledge sources to the workers from where they can copy text with the addition of suitable prefixes and suffixes to generate appropriate coherent responses.

3 Dataset

In the following sub-sections we describe the various stages involved in collecting our dataset.

3.1 Curating a list of popular movies

We created a list of 921 movies containing (i) top 10 popular movies within the past five years, (ii) top 250 movies as per IMDb rankings, (iii) top 10 movies in popular genres, and (iv) other popular movie lists made available elsewhere on the Internet. These movies belonged to 22 different genres such as sci-fi, action, horror, fantasy, adventure, romance, *etc.* thereby ensuring that our dataset is not limited to a specific genre. We considered those movies for which enough background information such as plots, reviews, comments, facts, *etc.* were available on the Internet irrespective of whether they were box-office successes or not. Please find the respective urls in the Appendix.

3.2 Collecting background knowledge

For each movie, we collected the following background knowledge:

1. Review (R): For each movie, we asked some in-house workers to fetch the top 2 most popular reviews for this movie from IMDb using the *sort* *by Total Votes* option. We also instructed them to avoid choosing reviews which were less than 50 words but this was typically never the case with popular reviews. **2. Plot (P):** For each movie, we extracted information about the "Plot" of the movie from the Wikipedia page of the movie. Wikipedia pages of movies have an explicit section on "Plot" making it easy to extract this information using scripts. **3. Comments (C):** Websites like *Reddit* have a segment called "official discussion page about X" (where X is a movie name) containing small comments about various aspects of movie. We identified such pages and extracted the first comment on every thread on this page. We bundled all these comments into a single text file and refer to it as the resource containing "Comments". For a few movies, the official discussion page was not present in which case we used the review titles of all the IMDb reviews of the movie as comments. The difference between Reviews and Comments is that a Review is an opinion piece given by one person thus typically exhibiting one sentiment throughout while Comments include opinions of several people about the same movie ensuring that positive, negative and factual aspects of the movie are captured as well as some banter.

4. Meta data or Fact Table (F): For each movie, we also collected factual details about the movie, *viz.*, box office collection, similar movies (for recommendations), awards and tag-lines from the corresponding IMDb pages and Wikipedia Infoboxes. Such information would be useful for inserting facts in the conversation, for example, *"Did you know that the movie won an Oscar?"*. We included only 4 fields in our fact table instead of showing the entire Wikipedia Infobox to reduce the cognitive load on turkers who already had to read the plot, reviews and comments of the movie.

3.3 Collecting conversation starters

During our initial pilots, we observed that if we asked the workers to converse for at least 8 turns, they used a lot of the initial turns in greetings and general chit-chat before actually chatting about a movie. To avoid this, we collected opening statements using Amazon Mechanical Turk (AMT) where the task for the workers was to answer the following questions "*What is your favorite scene from the movie X ?*", "*What is your favorite character from the movie X ?*" and "*What is your opinion about the movie X?*" (X is the movie name). We paid the workers 0.04$ per movie and showed the same movie to 3 different workers, thereby collecting 9 different opening statements for every movie. By using these statements as conversation starters in our data collection, the workers could now directly start conversing about the movie.

3.4 Collecting background knowledge aware conversations via crowdsourcing

Our aim is to create a conversation dataset wherein every response is explicitly linked to some structured or unstructured background knowledge. Creating such a dataset using dedicated in-house workers would obviously be expensive and time consuming and so we decided to use crowdsourcing. However, unlike other NLP and Vision tasks, where crowdsourcing has been very successful, collecting conversations via crowdsourcing is a bit challenging. The main difficulty arises from the fact that conversation is inherently a task involving two persons but it is hard to get two workers to synchronize and chat on AMT. We did try a few pilot experiments where we setup a server to connect two AMT workers but we found that the probability of two workers simultaneously logging in was very low. Thus, most workers logged in and left in a few seconds because no other worker joined simultaneously. Finally, we took inspiration from the idea of self chats Krause et al. (2017) in which, the same worker plays the role of both Speaker 1 and Speaker 2 to create the chat. In the above self chat setup, we showed every worker 3 to 4 resources related to the movie, *viz.*, plot (P), review (R), comments (C) and fact table (F). We also showed them a randomly selected opening statement from the 9 opening statements that we had collected for each movie and requested them to continue the conversation from that point. The workers were asked to add at least 8 utterances to this initial chat. While playing the role of Speaker 1, the worker was not restricted to copy/modify sentences from the background resources but was given the freedom to create (write) original sentences. However, when playing the role of Speaker 2, the worker was strictly instructed to copy/modify sentences from the shown resources such that they were relevant in the current context of the conversation. The reason for not imposing any restrictions on Speaker 1 was to ensure that the chats look more natural and coherent. Further, Speaker 2 was allowed to add words at the beginning or end of the span selected from the resources to make the chats more coherent and natural (for example, see the prefix in utterance 2 of Speaker 2 in Figure 1). We paid the workers 40 cents for every chat. Please refer to the Appendix for the instruction screen shots.

3.5 Verification of the collected chats

Every chat that was collected by the above process was verified by an in-house evaluator to check if the workers adhered to the instructions and produced coherent chats. Since humans typically tend to paraphrase the background knowledge acquired by reading articles, one could argue that such conversations may not look very natural because of this restriction to copy/modify content from the provided resources. To verify this, we conducted a separate human evaluation wherein we asked 15 in-house evaluators to read conversations (without the background resources) from our dataset and rate them on five different parameters. Specifically, they were asked to check if the conversations were 1) **intelligible:** *i.e.*, an average reader could understand the conversation 2) **coherent:** *i.e.*, there were no abrupt context switches 3) **grammatically correct** 4) **on-topic:** *i.e.*, the chat revolved around the concerned movie with digression limited to related movies/characters/actors and 5) **natural two-person chats:** *i.e.*, the role-play setup does not make the chat look unnatural. These evaluators were post-graduate students who were fluent in English and had watched at least 100 Hollywood movies. We did not give them any information about the data creation process. We used a total of 500 chats for the evaluation and every chat was shown to 3 different evaluators. The evaluators rated the conversations on a scale of 1 (very poor) to 5 (very good). We computed inter-annotator agreement using the mean linearly weighted Cohen's κ (Cohen, 1968) and mean Krippendorff's α (Hayes and Krippendorff, 2007). The average rating for each of the 5 parameters along with the inter annotator agreement are reported in Table 1 and are very encouraging.

Metric	**Rating**	α	κ
Intelligible	4.47 ± 0.52	0.70	0.69
Coherent	4.33 ± 0.93	0.57	0.71
Grammar	4.41 ± 0.56	0.60	0.69
Two-person-chat	4.47 ± 0.46	0.64	0.70
On Topic	4.57 ± 0.43	0.72	0.70

Table 1: Average human evaluation scores with standard deviations for conversations (scale 1-5). We also report mean Krippendorff's α and mean Cohen's κ

#chats	9071
#movies	921
#utterances	90810
Average # of utterances per chat	10.01
Average # of words per utterance	15.29
Average # of words per chat	153.07
Average # of words in Plot	186.10
Average # of words in Review	384.44
Average # of words in Comments	123.81
Average # of words in Fact Table	33.47
# unique Plots	5157
# unique Reviews	1817
# unique Comments	12740

Table 2: Statistics of the dataset

3.6 Statistics

In Table 2, we show different statistics about the dataset collected using the above process. These include average number of utterances per chat, average number of words per utterance, and so on followed by the statistics of the different resources which were used as background knowledge. Please note that the # unique Plots and # unique Reviews correspond to unique paragraphs while the # unique Comments is the count of unique sentences. We observed that 41.2%, 34.6%, 16.1% and 8.1% of Speaker 2 responses came from Reviews, Comments, Plots and Fact Table respectively.

4 Models

We evaluate three different types of models as described below. Since these are popular existing models, we describe them very briefly below and refer the reader to the original papers for more details. Note that in this work we merge the comments, reviews, plots and facts into one single document and refer to it as background knowledge. In the rest of the paper, when we refer to a *resource* we mean this single document which is a merger of all the resources unless specified otherwise.

4.1 Generation based models

We use the standard Hierarchical Recurrent Encoder Decoder model (HRED) (Serban et al., 2016) instead of its variant (Serban et al., 2017b) as the standard model performs only slightly poorly than the variant and is much easier to implement. It decomposes the context of the conversation as two level hierarchy using Recurrent Neural Networks (RNN). The lower RNN encodes individual utterances (sequence of words) which is then fed into the higher level RNN as a sequence of utterances. The decoder RNN then generates the output based on this hierarchical context representation.

4.2 Generate-or-Copy models

Get To The Point (GTTP) (See et al., 2017) proposed a hybrid pointer generator network for abstractive summarization that learns to copy words from the source document when required and otherwise generates a word like any sequence-to-sequence model. In the summarization task, the input is a *document* and the output is a *summary* whereas in our case the input is a {*document, context*} pair and the output is a *response*. Here, the context includes the previous two utterances and the current utterance. We modified the architecture to suit our task. We use an RNN to compute the representation of the document (like the original model) and introduce another RNN to compute a representation of the context by treating it as a single sequence of words. The decoder which is also an RNN then uses the document representation, context representation and its own internal state representation to compute a (i) probability score which indicates whether the next word should be copied or generated (ii) probability distribution over the vocabulary if the next word needs to be generated and (iii) probability distribution over the input words if the next word needs to be copied. These three probability distributions are then combined to produce the next word in the response.

4.3 Span prediction models

Bi-directional Attention Flow Model (BiDAF) (Seo et al., 2017) model is a QA model which was proposed in the context of the SQuAD dataset (Rajpurkar et al., 2016). Given a *document* and a *question*, the model uses a six-layered architecture to predict the span in the document which contains the answer. We can use their model as it is for our task without any modifications by simply treating the *context* as the *question* and the *resource* as the *document*.

We chose to evaluate on the modified generate-or-copy model instead of other variants such as (Ghazvininejad et al., 2017; Lowe et al., 2015a) as the modified model already contains the extra encoder for background model which is present in these models. Moreover, the modified model uses a hybrid copy-or-generate decoder which is well-suited to our task.

5 Experimental Setup

In this section we describe the train-validation-test splits, the process used for creating training instances, the manner in which the models were trained using our data and the evaluation metrics.

5.1 Creating train/valid/test splits

On average we have 9.14 chats per movie. We divide the collected chats into train, validation, and test splits such that all the chats corresponding to a given movie are in exactly one of the splits. This ensures that a movie seen in the test or validation set is never seen at training time. We create the splits such that the percentage of chats in the train-validation-test set is roughly 80%-10%-10%.

5.2 Creating training instances

For each chat in the training data, we construct training instances of the form {*resource, context, response*} where the *context* is taken as previous two utterances and current utterance. We consider only the even numbered utterances as training examples as they are generated from the background resources thus emulating a human-bot setup. If a chat has 10 turns, we will have 5 instances. The task then is to train a model which can predict these even numbered responses. At test time the model is shown {*resource, context*} and predicts the response. Note that, HRED will ignore the *resource* and only use {*context, response*} as input-output pairs. BiDAF and GTTP will use {*resource, context, response*} as training data with relevant *span* instead of *response* for BiDAF.

5.3 Merging resources into a single document

As stated earlier, we simply merge all the background information to create a single document which we collectively refer to as *resource*. For the BiDAF model, we had to restrict the length of the resource to 256 words because we found that even on a K80 GPU with 12GB RAM, this model gives an out of memory error for longer documents. We found this to be a severe limitation of this and other span based models (for example, R-Net (Wang et al., 2017)) . We experimented with three methods of creating this resource. The first method *oracle* uses the actual resource (plot or comments or reviews) from which the next response was generated as a resource. If that resource itself has more than 256 words then we truncate it from the beginning and the end such that the span containing the actual response is contained within the retained 256 words. The number of words that are discarded from the start or the end is chosen at random so that the correct spans do not end up in similar positions throughout the dataset. The next two methods *mixed-short* and *mixed-long* are created by merging the individual resources. We retain each resource in the merged document proportional to its length. (*i.e,*if there are 400 words in the plot, 200 words in the review and 100 in the comments, the merged resource will contain contiguous sentences from these three resources in the ratio of 4:2:1.) Further, we ensure that the merged resource contains the actual response span. In this way, we create *mixed-short* with 256 words and *mixed-long* with 1200 words (the maximum length of the merged resources). We will henceforth denote *oracle*, *mixed-long* and *mixed-short* using '(o) ', '(ms) 'and '(ml) 'respectively. We report results for BiDAF(o), BiDAF (ms), GTTP (o) and GTTP (ml).

5.4 Evaluation metrics

As HRED and GTTP models are generation based models we use BLEU-4, ROUGE-1, ROUGE-2 and ROUGE-L as the evaluation metrics. For BiDAF we use the above metrics by comparing the predicted span with the reference span. For BiDAF, we also report F1 as stated in Rajpurkar et al. (2016).

In addition to the automatic evaluation, we also collected human judgments using 100 test responses generated for every model for every setup (oracle, mixed-short, mixed-long). These evaluators had the same qualifications as the evaluators who earlier helped us evaluate our dataset. They were asked to rate the response on scale of 1 to 5 (with 1 being the least) on the following four metrics: (1) Fluency(Flu), (2) appropriateness/relevance (apt) of the response in the current context language (3) humanness (Hum) of the response, *i.e.,* whether the responses look as if they were generated by a human (4) and specificity (spec) of the response, *i.e.*, whether the model produced movie-specific responses or generic responses such as "This movie is amazing". We report these results in Table 4.

5.5 Collecting multiple reference responses

One common issue with evaluating dialog systems is that existing datasets typically contain only one reference response whereas in practice several responses can be correct in a given context. To solve this to a certain extent, we collected three reference responses for every Speaker 2 utterance in our dataset (note that Speaker 2 is treated as the bot while training/testing our models). We show the previous utterances ending with Speaker 1's response and ask workers to provide three appropriate responses from the given resources. We found that in some cases there was only one appropriate response like factual response and the workers could not provide multiple references . In this way we were able to create a multiple reference test set where 78.04% of the test instances have multiple responses. In Table 3, we report two sets of scores based on single-reference test dataset and multi-reference test dataset. While calculating the scores for multi-reference dataset, we take the maximum score over multiple reference responses.

Please refer to the Appendix section for the details of the model, hyperparameters, example of multiple references in our dataset and sample outputs produced by different models.

6 Results and Discussion

In this section, we discuss the results of our experiments as summarized in Tables 3 and 4.

Generation based models v/s Span prediction models: We compare the generation based models and span prediction models only based on results in the *oracle* setting. Here, the span based model (BiDAF) outperforms the generation based models (HRED and GTTP). This confirms our belief that the natural language generation (NLG) capabilities of current generation based models are far from being acceptable even in case of generate-or-copy modes. This also emphasizes the importance of this data which allows building models which can exploit well-formed sentences in the background knowledge and reproduce them with minor modifications instead of generating them from scratch. While the results for BiDAF are

Model	**F1**		**BLEU**		**Rouge-1**		**Rouge-2**		**Rouge-L**	
HRED	-	-	5.23	5.38	24.55	25.38	7.61	8.35	18.87	19.67
GTTP (o)	-	-	13.92	16.46	30.32	31.6	17.78	21.21	25.67	27.83
GTTP (ms)	-	-	11.05	15.68	29.66	31.71	17.70	19.72	25.13	27.35
GTTP (ml)	-	-	7.51	8.73	23.20	21.55	9.91	10.42	17.35	18.12
BiDAF (o)	39.69	47.18	28.85	34.98	39.68	46.49	33.72	40.58	35.91	42.64
BiDAF (ms)	45.73	51.35	32.95	39.39	45.69	50.73	40.18	45.01	43.46	46.95

Table 3: Performance of the proposed models on our dataset. The figures on the left in each column indicate scores on single-reference test dataset while the figures on the right denote scores on multi-reference dataset.

Model	**Hum**	**Apt**	**Flu**	**Spec**
HRED	3.08	2.49	2.64	2.06
GTTP (o)	4.10	3.73	4.03	3.33
GTTP (ml)	2.93	2.97	3.42	2.60
BiDAF (o)	3.78	3.71	4.05	3.76
BiDAF(ms)	3.41	3.38	3.47	3.30

Table 4: Human evaluation results on the model performances.

encouraging, we reiterate that it does not scale to longer documents (we were not able to run it in the *mixed-long* setting). We still need much better models as BiDAF on SQuAD dataset gives an F1 of 81.52 % which is much higher than the results on our dataset. Further, note that using the predicted span as a response is not natural. This is evident from human likeliness (Hum) score of GTTP (o) being higher than both the BiDAF models. We need models which can suitably alter the span to retain the coherence of the context.

Effect of including background knowledge: We observe that there isn't much difference between the performance of HRED which does not use any background knowledge when compared to GTTP (ml) which actually uses a lot of background knowledge. However, there is a substantial difference between the performance of HRED and GTTP (o) which uses only the relevant background knowledge. Further, without background knowledge, HRED learns to produce very generic responses (Spec score = 2.06). This shows that the background knowledge is important, but the models should learn to focus on the right background knowledge relevant to the current context. Alternately, we can have a two-stage network which first predicts the right resource (plot, review, comments) from which the span should be selected and then selects the span from this chosen resource.

***Oracle* v/s *mixed-short* resource:** We observe that the performance of BiDAF (ms) is actually better than BiDAF (o) even when the resource length for both is 256 words. We would expect a poor performance for BiDAF (ms) as the resource has more noise because of the sentences from irrelevant resources. However, we speculate the model learns to regard irrelevant sentences as noise and learns to focus on sentences corresponding to the correct resource resulting in improved performance (however, this is only a hypothesis and it needs to be verified). We realize that this is clearly a poor baseline and we need better span prediction based models which can work with longer documents. At the same time, GTTP (o) and GTTP (ms) have comparable (yet poor) performance. There is no co-attention mechanism in this model which can effectively filter out noisy sentences.

Observations from the copy-and-gen model: We observed that this model produced sentences where on average of 82.18% (*oracle*) and 71.95% (*mixed-long*) of the tokens were copied. One interesting observation was that it easily learns to copy longer contiguous sequences one word at a time. However, as is evident from the automatic evaluation metrics, in many cases, the 'copied' spans are not relevant to the current context.

Evaluating with multiple references: When considering multiple references, the performance numbers as reported in Table 3 indeed improve. This shows the importance of having multiple references and the need to develop metrics which account for multiple dissimilar references.

7 Conclusion

We introduce a new dataset for building dialog systems which would hopefully allow the community to take a fresh look at this task. Unlike existing datasets which only contain a sequence of utterances, in our dataset each response is explicitly linked to some background knowledge. This mimics how humans converse by recalling information from their background knowledge and use it appropriately in the context of the conversation. Using this dataset, we evaluated models belonging to three different paradigms, *viz.*, generation based models, generate-or-copy models and span prediction models. Our results suggest that the NLG capabilities of existing seq-to-seq models are still far from desirable while span based models which completely bypass the process of NLG show some promise but with clear scope for improvement.

Going forward, we would like to build models which are a hybrid of span prediction models and generation models. Specifically, we would like to build models which can learn to copy a large sequence from the input instead of one word at a time. Another important aspect is to build less complex models which can handle longer documents. For example, the BiDAF model has an expensive outer product between two large matrices which makes it infeasible for long documents (because the size of these matrices grows with the length of the document). Alternately, we would like to build two-stage models which first select the correct resource from which the next response is to be generated and then generate or copy the response from the resource.

Acknowledgements

We would like to thank Department of Computer Science and Engineering, and Robert Bosch Center for Data Sciences and Artificial Intelligence, IIT Madras (RBC-DSAI) for providing us with adequate resources. We also thank Gurneet Singh and Sarath Chandar for helping us in the data collection phase two and three respectively. Lastly, we thank all the AMT workers around the world and our in-house evaluators.

References

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *International Conference on Learning Representations*.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander H. Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. *International Conference on Learning Representations*, abs/1511.06931.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pages 37–49.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A knowledge-grounded neural conversation model. *CoRR*, abs/1702.01932.

William H. Guss, James Bartlett, Phillip Kuznetsov, and Piyush Patil. 2017. Eigen: A step towards conversational ai. *Alexa Prize Proceedings*.

Andrew F Hayes and Klaus Krippendorff. 2007. Answering the call for a standard reliability measure for coding data. *Communication methods and measures*, 1(1):77–89.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1766–1776.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA*, pages 263–272.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014b. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA, December 7-10, 2014*, pages 324–329.

Ben Krause, Marco Damonte, Mihai Dobre, Daniel Duma, Joachim Fainberg, Federico Fancellu, Emmanuel Kahembwe, Jianpeng Cheng, and Bonnie L. Webber. 2017. Edina: Building an open domain socialbot with self-dialogues. *Alexa Prize Proceedings*.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2443–2453.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, pages 994–1003.

Yang Liu, Tim Paek, and Manasi Patwardhan, editors. 2018. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HTL 2018, New Orleans, Louisiana, USA, June 2-4, 2018, Demonstrations*. Association for Computational Linguistics.

Ryan Lowe, Nissan Pow, Iulian Serban, Laurent Charlin, and Joelle Pineau. 2015a. Incorporating unstructured textual knowledge sources into neural dialogue systems. In *Neural Information Processing Systems Workshop on Machine Learning for Spoken Language Understanding*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015b. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic*, pages 285–294.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. 2017. Conversational AI: the science behind the alexa prize. *Alexa Prize Proceedings*.

Antoine Raux, Brian Langner, Dan Bohus, Alan W. Black, and Maxine Eskénazi. 2005. Let's go public! taking a spoken dialog system to the real world. In *INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech Communication and Technology, Lisbon, Portugal, September 4-8, 2005*, pages 885–888.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 172–180.

Lina Maria Rojas-Barahona, Milica Gasic, Nikola Mrksic, Pei-Hao Su, Stefan Ultes, Tsung-Hsien Wen, Steve J. Young, and David Vandyke. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 438–449.

Diane L. Schallert. 2002. Schema theory. *Literacy in America: An encyclopedia of history, theory, and practice Santa Barbara, CA*, pages 556–558.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Stephanie Seneff, James R. Glass, David Goddeau, David Goodine, Lynette Hirschman, Hong C. Leung, Michael S. Phillips, Joseph Polifroni, and Victor Zue. 1991. Development and preliminary evaluation of the MIT ATIS system. In *Speech and Natural Language, Proceedings of a Workshop held at Pacific Grove, California, USA, February 19-22. 1991*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *International Conference on Learning Representations*.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2015. A survey of available corpora for building data-driven dialogue systems. *CoRR*, abs/1512.05742.

Iulian Vlad Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Mudumba, Alexandre de Brébisson, Jose Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau, and Yoshua Bengio. 2017a. The octopus approach to the alexa competition: A deep ensemble-based socialbot. *Alexa Prize Proceedings*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of*

Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pages 1577–1586.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 196–205.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *Proceedings of ICML Deep Learning Workshop, 2015.*

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 189–198.

Jason D. Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *D&D*, 7(3):4–33.

Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan W. Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference, The 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 22-24 August 2013, SUPELEC, Metz, France*, pages 404–413.

Decoupling Strategy and Generation in Negotiation Dialogues

He He and **Derek Chen** and **Anusha Balakrishnan** and **Percy Liang**
Computer Science Department, Stanford University
{hehe,derekchen14,anusha,pliang}@cs.stanford.edu

Abstract

We consider negotiation settings in which two agents use natural language to bargain on goods. Agents need to decide on both high-level strategy (e.g., proposing $50) and the execution of that strategy (e.g., generating *"The bike is brand new. Selling for just $50!"*). Recent work on negotiation trains neural models, but their end-to-end nature makes it hard to control their strategy, and reinforcement learning tends to lead to degenerate solutions. In this paper, we propose a modular approach based on coarse dialogue acts (e.g., propose(price=50)) that decouples strategy and generation. We show that we can flexibly set the strategy using supervised learning, reinforcement learning, or domain-specific knowledge without degeneracy, while our retrieval-based generation can maintain context-awareness and produce diverse utterances. We test our approach on the recently proposed DEALORNODEAL game, and we also collect a richer dataset based on real items on Craigslist. Human evaluation shows that our systems achieve higher task success rate and more human-like negotiation behavior than previous approaches.

1 Introduction

A good negotiator needs to decide on the *strategy* for achieving a certain goal (e.g., proposing $6000) and the realization of that strategy via *generation* of natural language (e.g., *"I really need a car so I can go to work, but all I have is 6000, any more and I won't be able to feed my children."*).

Most past work in NLP on negotiation focuses on strategy (dialogue management) with either no natural language (Cuayáhuitl et al., 2015; Cao et al., 2018) or canned responses (Keizer et al., 2017; Traum et al., 2008). Recently, end-to-end neural models (Lewis et al., 2017; He et al., 2017) are used to simultaneously learn dialogue strategy and language realization from human-human dialogues, following the trend of using neural network models on both goal-oriented dialogue (Wen et al., 2017a; Dhingra et al., 2017) and open-domain dialogue (Sordoni et al., 2015; Li et al., 2017; Lowe et al., 2017). However, these models have two problems: (i) it is hard to control and interpret the strategies, and (ii) directly optimizing the agent's goal through reinforcement learning often leads to degenerate solutions where the utterances become ungrammatical (Lewis et al., 2017) or repetitive (Li et al., 2016).

To alleviate these problems, our key idea is to decouple strategy and generation, which gives us control over the strategy such that we can achieve different negotiation goals (e.g., maximizing utility, achieving a fair deal) with the same language generator. Our framework consists of three components shown in Figure 1: First, the parser identifies keywords and entities to map each utterance to a *coarse dialogue act* capturing the high-level strategic move. Then, the dialogue manager chooses a responding dialogue act based on a sequence-to-sequence model over coarse dialogue acts learned from parsed training dialogues. Finally, the generator produces an utterance given the dialogue act and the utterance history.

Our framework follows that of traditional goal-oriented dialogue systems (Young et al., 2013), with one important difference: coarse dialogue acts are not intended to and cannot capture the full meaning of an utterance. As negotiation dialogues are fairly open-ended, the generator needs to depend on the full utterance history. For example, consider the first turn in Figure 1. We cannot generate a response given only the dialogue act inform; we must also look at the previous question. However, we still optimize the dialogue manager in the coarse dialogue act space using supervised learning, reinforcement learning, or domain-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2333–2343
Brussels, Belgium, October 31 - November 4, 2018.

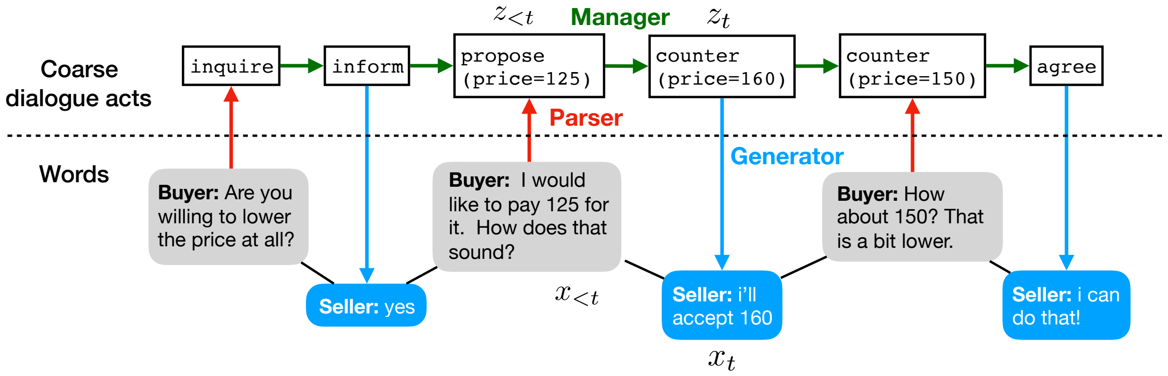

Figure 1: Our modular framework consists of three components similar to traditional goal-oriented dialogue systems. (1) The parser maps received utterances to coarse dialogue acts (an intent and its arguments) that capture the high-level dialogue flow. (2) The manager generates the next coarse dialogue act z_t conditioned on past dialogue acts $z_{<t}$. (3) The generator then produces a response conditioned on both the predicted coarse dialogue act z_t and the dialogue history $x_{<t}$. Importantly, unlike in traditional systems, coarse dialogue acts only capture the rough shape of a dialogue, not the full meaning of its utterances, e.g., `inform` does not specify the answer to the question.

specific knowledge.

Existing human-human negotiation datasets are grounded in closed-domain games with a fixed set of objects such as Settlers of Catan (lumber, coal, brick, wheat, and sheep) (Afantenos et al., 2012; Asher et al., 2016) or item division (book, hat, and ball) (DeVault et al., 2015; Lewis et al., 2017). These objects lack the richness of the real world. To study human negotiation in more open-ended settings that involve real goods, we scraped postings of items for sale from `craigslist.org` as our negotiation scenario. By hiring workers on Amazon Mechanical Turk (AMT) to play the role of buyers and sellers, we collected a new dataset (CRAIGSLISTBARGAIN) of negotiation dialogues.[1] Compared to existing datasets, our more realistic scenario invites richer negotiation behavior involving open-ended aspects such as cheap talk or side offers.

We evaluate two families of systems modeling coarse dialogue acts and words respectively, which are optimized by supervised learning, reinforcement learning, or domain knowledge. Each system is evaluated on our new CRAIGSLISTBARGAIN dataset and the DEALORNODEAL dataset of Lewis et al. (2017) by asking AMT workers to chat with the system in an A/B testing setting. We focus on two metrics: task-specific scores (e.g., utility) and human-likeness. We show that reinforcement learning on coarse dialogue acts avoids degenerate solutions, which was a problem in Li et al. (2016); Lewis et al. (2017). Our modular model maintains reasonable human-like behavior while still optimizes the objective. Furthermore, we find that models trained over coarse dialogue acts are stronger negotiators (even with only supervised learning) and produce more diverse utterances than models trained over words. Finally, the interpretability of coarse dialogue acts allows system developers to combine the learned dialogue policy with hand-coded rules, thus imposing stronger control over the desired strategy.

2 Craigslist Negotiation Dataset

Previous negotiation datasets were collected in the context of games. For example, Asher et al. (2016) collected chat logs from online Settlers of Catan. Lewis et al. (2017) asked two people to divide a set of hats, books, and balls. While such games are convenient for grounding and evaluation, it restricts the dialogue domain and the richness of the language. Most utterances are direct offers such as *"has anyone got wood for me?"* and *"I want the ball."*, whereas real-world negotiation would involve more information gathering and persuasion.

To encourage more open-ended, realistic negotiation, we propose the CRAIGSLISTBARGAIN task. Two agents are assigned the role of a buyer and a seller; they are asked to negotiate the price of an item for sale on Craigslist given a description and photos. As with the real platform, the listing price is shown to both agents. We addition-

---

[1] Available at `https://stanfordnlp.github.io/cocoa`.

JVC HD-ILA 1080P 70 Inch TV

Tv is approximately 10 years old. Just installed new lamp. There are 2 HDMI inputs. Works and looks like new.
Listing price: $275
Buyer's target price: $192

Agent	Utterance	Dialogue Act
Buyer	Hello do you still have the TV?	greet
Seller	Hello, yes the TV is still available	greet
Buyer	What condition is it in? Any scratches or problems? I see it recently got repaired	inquire
Seller	It is in great condition and works like a champ! I just installed a new lamp in it. There aren't any scratches or problems.	inform
Buyer	All right. Well I think 275 is a little high for a 10 year old TV. Can you lower the price some? How about 150?	propose(150)
Seller	I am willing to lower the price, but $150 is a little too low. How about $245 and if you are not too far from me, I will deliver it to you for free?	counter(245)
Buyer	It's still 10 years old and the technology is much older. Will you do 225 and you deliver it. How's that sound?	counter(225)
Seller	Okay, that sounds like a deal!	agree
Buyer	Great thanks!	agree
Seller	OFFER $225.0	offer(225)
Buyer	ACCEPT	accept

Table 1: Example dialogue between two people negotiating the price of a used TV.

ally suggest a private price to the buyer as a target. Agents chat freely in alternating turns. Either agent can enter an offer price at any time, which can be accepted or rejected by the partner. Agents also have the option to quit, in which case the task is completed with no agreement.

To generate the negotiation scenarios, we scraped postings on `sfbay.craigslist.org` from the 6 most popular categories (housing, furniture, cars, bikes, phones, and electronics). Each posting produces three scenarios with the buyer's target prices at 0.5x, 0.7x and 0.9x of the listing price. Statistics of the scenarios are shown in Table 2.

We collected 6682 human-human dialogues on AMT using the interface shown in Appendix A Figure 2. The dataset statistics in Table 3 show that CraigslistBargain has longer dialogues and more diverse utterances compared to prior datasets. Furthermore, workers were encouraged to embellish the item and negotiate side offers such as free delivery or pick-up. This highly relatable scenario leads to richer dialogues such as the one shown in Table 1. We also observed various persuasion techniques listed in Table 4 such as embellishment, side offers, and appeals to sympathy.

# of unique postings	1402
% with images	80.8
Avg # of tokens per description	42.6
Avg # of tokens per title	33.8
Vocab size	12872

Table 2: Statistics of CraigslistBargain scenarios.

	CB	DN	SoC
# of dialogues	6682	5808	1081
Avg # of turns	9.2	6.6	8.5
Avg # of tokens per turn	15.5	7.6	4.2
Vocab size	13928	2719	4921
Vocab size (excl. numbers)	11799	2623	4735

Table 3: Comparison of dataset statistics of CraigslistBargain (CB), DealOrNoDeal (DN), and SettlersOfCatan (SoC). CraigslistBargain contains longer, more diverse dialogues on average.

3 Approach

3.1 Motivation

While end-to-end neural models have made promising progress in dialogue systems (Wen et al., 2017a; Dhingra et al., 2017), we find they struggle to simultaneously learn the strategy and the rich utterances necessary to succeed in the CraigslistBargain domain, e.g., Table 8(a) shows a typical dialogue between a human and a sequence-to-sequence-based bot, where the bot easily agrees. We wish to now separate negotiation strategy and language generation. Suppose the buyer says: *"All right. Well I think 275 is a little high for a 10 year old TV. Can you lower the price some? How about 150?"* We can capture the highest-order bit with a coarse dialogue act propose(price=150). Then, to generate the seller's response, the agent can first focus on this coarse

Phenomenon	Example
Embellishment	It is in great condition and **works like a champ! I just installed a new lamp in it.** There aren't any scratches or problems.
Cheap talk	How about i give you $20 and you keep the helmet. **its for my daughter for her job, she delivers lemonade**.
Side offers	**Throw in a couple of movies** with that DVD player, and you have yourself a deal.
Appeal to sympathy	I would love to have this for my mother, **she is very sick** and this would help her and with me taking care of her and having to take a leave from work I can't pay very much of it
World knowledge	For a **Beemer 5 series** in this condition, I really can't go that low.

Table 4: Rich negotiation language in our CRAIGSLISTBARGAIN dataset.

dialogue act rather than having to ingest the free-form text all at once. Once a counter price is decided, the rest is open-ended justification for the proposed price, e.g., emphasizing the quality of the TV despite its age.

Motivated by these observations, we now describe a modular framework that extracts coarse dialogue acts from utterances, learns to optimize strategy in the dialogue act space, and uses retrieval to fill in the open-ended parts conditioned on the full dialogue history.

3.2 Overview

Our goal is to build a dialogue agent that takes the dialogue history, i.e. a sequence of utterances $x_1, \dots, x_{t-1}$ along with the dialogue scenario c (e.g., item description), and produces a distribution over the responding utterance x_t.

For each utterance x_t (e.g., *"I am willing to pay $15"*), we define a coarse dialogue act z_t (e.g., propose(price=15)); the coarse dialogue act serves as a logical skeleton which does not attempt to capture the full semantics of the utterance. Following the strategy of traditional goal-oriented dialogue systems (Young et al., 2013), we broadly define our model in terms of the following three modules:

1. A **parser** that (deterministically) maps an input utterance x_{t-1} into a coarse dialogue act z_{t-1} given the dialogue history $x_{<t}$ and $z_{<t}$, as well as the scenario c.
2. A **manager** that predicts the responding dialogue act z_t given past coarse dialogue acts $z_{<t}$ and the scenario c.
3. A **generator** that turns the coarse dialogue act z_t to a natural language response x_t given the full dialogue history $x_{<t}$.

Because coarse dialogue acts do not capture the full semantics, the parser and the generator maintains full access to the dialogue history. The main restriction is the manager examining the dialogue acts, which we show will reduce the risk of degeneracy during reinforcement learning Section 4.4. We now describe each module in detail (Figure 1).

3.3 Parser

Our framework is centered around the coarse dialogue act z, which consists of an intent and a set of arguments. For example, *"I am willing to pay $15"* is mapped to propose(price=15). The fact that our coarse dialogue acts do not intend to capture the full semantics of a sentence allows us to use a simple rule-based parser. It detects the intent and its arguments by regular expression matching and a few if-then rules. Our parser starts by detecting entities (e.g., prices, objects) and matching keyword patterns (e.g., *"go lower"*). These signals are checked against an ordered list of rules, where we choose the first matched intent in the case of multiple matches. An unknown act is output if no rule is triggered. The list of intent parsing rules used are shown in Table 5. Please refer to Appendix B for argument parsing based on entity detection.

3.4 Manager

The dialogue manager decides what action z_t the dialogue agent should take at each time step t given the sequence of past coarse dialogue acts $z_{<t}$ and the scenario c. Below, we describe three ways to learn the dialogue manager with increasing controllability: modeling human behavior in the training corpus (supervised learning), explicitly optimizing a reward function (reinforcement learning), and injecting hand-coded rules (hybrid policy).

Supervised learning. Given a parsed training corpus, each training example is a sequence of coarse dialogue acts over one dialogue, $z_1, \dots, z_T$. We learn the transition probabilities

Generic Rules	
Intent	Matching Patterns
greet	*hi, hello, hey, hiya, howdy*
disagree	*no, not, n't, nothing, dont*
agree	not disagree and *ok, okay, great, perfect, deal, that works, i can do that*
insist	the same offer as the previous one is detected
inquire	starts with an interrogative word (e.g., *what, when, where*) or particle (e.g., *do, are*)
CRAIGSLISTBARGAIN Rules	
Intent	Matching Patterns
intro	greet or *how are you, interested*
propose	first price mention
vague-price	no price mention and *come down, highest, lowest, go higher/lower, too high/low*
counter	new price detected
inform	previous coarse dialogue act was inquire
DEALORNODEAL Rules	
Intent	Matching Patterns
propose	items and respective counts are detected

Table 5: Rules for intent detection in the parser.

$p_\theta(z_t \mid z_{<t}, c)$ by maximizing the likelihood of the training data.

We use a standard sequence-to-sequence model with attention. Each coarse dialogue act is represented as a sequence of tokens, i.e. an intent followed by each of its arguments, e.g., "offer 150". During the agent's listening turn, an LSTM encodes the received coarse dialogue act; during its speaking turn, another LSTM decodes the tokens in the coarse dialogue act. The hidden states are carried over the entire dialogue to provide full history.

The vocabulary of coarse dialogue acts is much smaller than the word vocabulary. For example, our implementation includes fewer than 10 intents and argument values are normalized and binned (see Section 4.2).

Reinforcement learning. Supervised learning aims to mimic the average human behavior, but sometimes we want to directly optimize for a particular dialogue goal. In reinforcement learning, we define a reward $R(z_{1:T})$ on the entire sequence of coarse dialogue acts. Specifically, we experiment with three reward functions:

- **Utility** is the objective of a self-interested agent. For CRAIGSLISTBARGAIN, we set the utility function to be a linear function of the final price, such that the buyer has a utility of 1 at their target price, the seller has a utility of 1 at the listing price, and both agents have a utility of zero at the midpoint of the listing price and the buyer's target price, making it a zero-sum game. For DEALORNODEAL, utility is the total value of objects given to the agent.
- **Fairness** aims to achieve equal outcome for both agents, i.e. the difference between two agents' utilities.
- **Length** is the number of utterances in a dialogue, thus encourages agents to chat as long as possible.

The reward is -1 if no agreement is reached.

We use policy gradient (Williams, 1992) for optimization. Given a sampled trajectory $z_{1:T}$ and the final reward r, let a_i be the i-th generated token (i.e. "action" taken by the policy) along the trajectory. We update the parameters θ by

$$\theta \leftarrow \theta - \eta \sum_i \nabla_\theta \log p_\theta(a_i \mid a_{<i}, c)(r - b) \quad (1)$$

where η is the learning rate and b is a baseline estimated by the average return so far for variance reduction.

Hybrid policy. Given the interpretable coarse dialogue acts, a simple option is to write a rule-based manager with domain knowledge, e.g., if z_{t-1} = greet, then z_t = greet. We combine these rules with a learned manager to fine-tune the dialogue policy. Specifically, the dialogue manager predicts the intent from a learned sequence model but fills in the arguments (e.g., price) using rules. For example, given a predicted intent propose, we can set the price to be the average of the buyer's and seller's current proposals (a split-the-difference strategy).

3.5 Generator

We use retrieval-based generation to condition on both the coarse dialogue act and the dialogue history. Each candidate in our database for retrieval is a tuple of an utterance x_t and its dialogue context x_{t-1}, represented by both templates and coarse dialogue acts. i.e. $(d(x_{t-1}), z_{t-1}, d(x_t), z_t)$, where d is the template extractor. Specifically, given a parsed training set, each utterance is converted to a template by delexicalizing arguments in its coarse dialogue act. For example, *"How about $150?"*

becomes “How about [price]?”, where [price] is a placeholder to be filled in at generation time.

At test time, given z_t from the dialogue manager, the generator first retrieves candidates with the same intent as z_t and z_{t-1}. Next, candidates are ranked by similarity between their context templates and the current dialogue context. Specifically, we represent the context $d(x_{t-1})$ as a TF-IDF weighted bag-of-words vector and similarity is computed by a dot product of two context vectors. To encourage diversity, the generator samples an utterance from the top K candidates according to the distribution given by a trigram language model estimated on the training data.

4 Experiments

4.1 Tasks

We test our approach on two negotiation tasks. **CRAIGSLISTBARGAIN** (Section 2) asks a buyer and a seller to negotiate the price of an item for sale given its Craigslist post. **DEALORNODEAL** (Lewis et al., 2017) asks two agents to divide a set of items given their private utility functions.

4.2 Models

We compare two families of models: end-to-end neural models that directly map the input dialogue context to a sequence of output words, and our modular models that use coarse dialogue acts as the intermediate representation.

We start by training the word-based model and the act-based model with supervised learning (SL).

- **SL(word)**: a sequence-to-sequence model with attention over previous utterances and the scenario, both embedded as a continuous Bag-of-Words;
- **SL(act)**: our model described in Section 3 with a rule-based parser, a learned neural dialogue manager, and a retrieval-based generator.

To handle the large range of argument values (prices) in CRAIGSLISTBARGAIN for act-based models, we normalize the prices such that an agent’s target price is 1 and the bottomline price is 0. For the buyer, the target is given and the bottomline is the listing price. For the seller, the target is the listing price and the bottomline is set to 0.7x of the listing price. The prices are then binned according to their approximate values with two digits after the decimal point.

Model	z	Parser	Manager	Generator
SL/RL(word)	vector	learned	learned	generative
SL/RL(act)	logical	rules	learned	retrieval
SL(act)+rule	logical	rules	hybrid	retrieval

Table 6: Comparison of different implementation of the core modules in our framework.

Next, given the pretrained SL models, we fine-tune them with the three reward functions (Section 3.4), producing $\mathbf{RL_{utility}}$, $\mathbf{RL_{fairness}}$, and $\mathbf{RL_{length}}$.

In addition, we compare with the hybrid model, **SL(act)+rule**. It predicts the next intent using a trigram language model learned over intent sequences in the training data, and fills in the arguments with hand-coded rules. For CRAIGSLISTBARGAIN, the only argument is the price. The agent always splits the difference when making counter proposals, rejects an offer if it is worse than its bottomline and accepts otherwise. For DEALORNODEAL, the agent maintains an estimate of the partner’s private utility function. In case of disagreement, it gives up the item with the lowest value of (own utility − partner utility) and takes an item of estimated zero utility to the partner. The agent agrees whenever a proposal is better than the last one or its predefined target. A high-level comparison of all models is shown in Table 6.

4.3 Training Details

CRAIGSLISTBARGAIN For SL(word), we use a sequence-to-sequence model with attention over 3 previous utterances and the negotiation scenario (embedded as a continuous Bag-of-Words). For both SL(word) and SL(act), we use 300-dimensional word vectors initialized by pretrained GloVe word vectors (Pennington et al., 2014), and a two-layer LSTM with 300 hidden units for both the encoder and the decoder. Parameters are initialized by sampling from a uniform distribution between -0.1 and 0.1. For optimization, we use AdaGrad (Duchi et al., 2010) with a learning rate of 0.01 and a mini-batch size of 128. We train the model for 20 epochs and choose the model with the lowest validation loss.

For RL, we first fit a partner model using supervised learning (e.g., SL(word)), then run RL

	CraigslistBargain					DealOrNoDeal				
	Hu	Ut	Fa	Ag	Len	Hu	Ut	Fa	Ag	Len
Human	4.3	-0.07	-0.14	0.91	10.2	4.6	5.5 vs. 5.3	-0.2	0.78	5.8
SL(word)	3.0	-0.32	-0.64	0.75	7.8	**3.8**	4.7 vs. 5.0	-0.3	0.70	5.0
SL(act)	3.3	0.06	-0.12	0.84	14.0	3.2	5.2 vs. 5.0	-0.2	0.67	7.0
SL(act)+rule	**3.6**	0.23	-0.46	0.75	11.4	**4.2**	5.2 vs. 5.2	0	0.72	8.0
RL$_{\text{utility}}$(word)	1.7	1.00	-2.00	0.31	2.5	1.7	2.9 vs. 1.8	-1.1	0.33	10.4
RL$_{\text{utility}}$(act)	**2.8**	1.00	-2.00	0.22	6.7	**2.8**	3.3 vs. 2.3	-1.0	0.38	9.5
RL$_{\text{fairness}}$(word)	1.8	-0.62	-1.24	0.75	9.4	3.2	5.7 vs. 5.9	-0.2	0.79	4.0
RL$_{\text{fairness}}$(act)	**3.0**	-0.28	-0.56	0.68	7.1	3.5	4.2 vs. 5.4	-1.2	0.77	7.6
RL$_{\text{length}}$(word)	1.9	-0.79	-1.58	0.85	13.8	1.6	3.4 vs. 2.9	-0.5	0.48	9.2
RL$_{\text{length}}$(act)	**3.0**	0.89	-1.78	0.40	11.8	**2.5**	2.5 vs. 3.1	-0.6	0.54	11.0

Table 7: Human evaluation results on human-likeness (Hu), agreement rate (Ag), and RL objectives, including agent utility (Ut), deal fairness (Fa), and dialogue length (Len). Results are grouped by the optimization objective. For each group of RL models, the column of the optimization objective is highlighted. For human-likeness, scores that are better than others in the same group with statistical significance ($p < 0.05$ given by paired t-tests) are in **bold**. Overall, with SL, all models are human-like, however, act-based models better matches human statistics across all metrics; with RL, word-based models becomes degenerate, whereas act-based models optimize the reward while maintaining human-likeness.

against it. One agent is updated by policy gradient and the partner model is fixed during training. We use a learning rate of 0.001 and train for 5000 episodes (dialogues). The model with the highest reward on the validation set is chosen.

DealOrNoDeal For act-based models, we use the same parameterization as CraigslistBargain. For word-based models, we use the implementation from Lewis et al. (2017).[2] Note that for fair comparison, we did not apply SL interleaving during RL training and rollouts during inference.

4.4 Human Evaluation

We evaluated each system on two metrics: task-specific scores (e.g., utility) and human-likeness. The scores tell us how well the system is playing the game, and human-likeness tells us whether the bot deviates from human behavior, presumably due to over-optimization.

We put up all 9 systems online and hired workers from AMT to chat with the bots. Each worker was randomly paired with one of the bots or another worker, so as to compare the bots with human performance under the same conditions. At the end of a chat, workers were asked the question *"Do you think your partner demonstrated reasonable human behavior?"*. They provided answers on a Likert scale from 1 (not at all) to 5 (definitely). Table 7 shows the human evaluation results on CraigslistBargain and DealOrNoDeal respectively. We also show example human-bot dialogues in Table 8 and Appendix C.

SL(act) learns more human-like behavior. We first compare performance of SL models over words and coarse dialogue acts. Both SL(word) and SL(act) achieved similar scores on human-likeness (no statistically significant difference). However, SL(word) better matched human statistics such as dialogue length and utility. For instance, SL(word) tended to produce short, generic utterances as shown in Table 8(a); they also agreed on a deal more quickly because utterances such as *"deal"* and *"I can do that"* are frequent in negotiation dialogues. This behavior is reflected by the shorter dialogue length and lower utility of SL(word) models.

RL(word) leads to degeneracy. On CraigslistBargain, all RL(word) models clearly have low scores on human-likeness in Table 7. They merely learned to repeat a few sentences: The three most frequent

---

[2] `https://github.com/facebookresearch/end-to-end-negotiator`

Title: Fitbit Flex + 2 Bands Activity Tracker	**Listing Price:** $60
Description: Hi, I'm selling a Fitbit Activity Tracker. Track your day including steps, distance, calories burned, active minutes. Track your night including hours slept, times to woken up, sleep quality, Wake with a silent vibrating alarm. You can set a ...	
(a) A: SL(word) (Buyer) B: Human (Seller)	(b) A: SL(act) (Buyer) B: Human (Seller)
A: **hi, i 'm interested in your fitbit.** B: Great. It really is a good deal. A: **great.** B: Do you have any questions about it? A: **i 'm interested in your fitbit. i 'm not sure i 'm not sure i 'm interested.** B: It has so many functions that are great. I particularly like the sleep tracking. Plus it is waterproof. How about I will deliver it to you and you give me 60 dollars? A: **i can do $52 if you come pick it up.** B: Well I am selling it. So I will deliver it for 60. Deal? A: **yes, i can do that.** B: OFFER($60) A: **ACCEPT**	A: **how we doing today** B: Hi! A: **yes, i really need one, what color is it?** B: It has a black band and a white band. A: **i see, so you bought an extra one by mistake?** B: No, it comes with 2 bands A: **i would like to make you a offer of 36** B: Ooooh....yeah, I can't. Sorry. It's wireless, has Bluetooth, extra bands, and is water resistant. I could do $50. A: **49?** B: You know..I'd do $49. A: **thank you so much!** B: OFFER($49) A: **ACCEPT**
(c) A: RL$_{\text{length}}$(word) (Buyer) B: Human (Seller)	(d) A: RL$_{\text{length}}$(act) (Buyer) B: Human (Seller)
A: **hello** B: Hi how are you? A: **i am.** B: you are interested in the fitbit flex and the 2 trackers? A: **it's in good condition condition.** B: yes A: **ok, that sounds good.** B: I am selling it at $60? A: **it's in good condition condition.** B: yes A: **ok, that sounds good.** B: $60 A: **i can go that low.**	A: **hi, i saw your ad about the item.** B: Okay great, i'm selling a Fitbit Flex plus 2 bands A: **how old is the item?** B: I've only had it for about a few months. A: **but it does work correct?** B: yes it does work, it's in great condition. A: **excellent. would you consider taking 36 for it?** B: I'm selling for $60 but $36 would be way too low. A: **how about 36?** B: no I cannot accept $36, I'm sorry A: **i'm sorry. would you accept 36?** B: I am not going lower than $50 A: **OFFER($36)** A: REJECT

Table 8: Example human-bot chats on CRAIGSLISTBARGAIN, where bot utterances are in bold. SL(word) produced generic responses, while SL(act) is more human-like. RL$_{\text{length}}$(word) devolved into degenerate behavior repeating itself while RL$_{\text{length}}$(act) maintained coherency. Only the first half of the item description and the RL$_{\text{length}}$(word) chat are shown due to space limit.

sentences of RL$_{\text{utility}}$(word), RL$_{\text{fairness}}$(word), and RL$_{\text{length}}$(word) account for 81.6%, 100% and 100% of all utterances. For example, RL$_{\text{utility}}$(word) almost always opened with *"i can pick it up"*, then offer its target price. RL$_{\text{length}}$(word) repeated generic sentences until the partner submitted a price. While they scored high on the reward being optimized, the conversations are unnatural.

On DEALORNODEAL, we have observed similar patterns. A general strategy learned by RL(word) was to pick an offer depending on its objective, then repeat the same utterance over and over again (e.g., *"i need the ball."*), resulting in low human-likeness scores. One exception is RL$_{\text{fairness}}$(word), since most of its offers were reasonable and agreed on immediately (it has the shorted dialogue length), the conversations are natural.

RL(act) optimizes different negotiation goals while being human-like. On both tasks, RL(act) models optimized their rewards while maintaining reasonable human-likeness scores. We now show that different models demonstrated different negotiation behavior. Two main strategies learned by RL$_{\text{length}}$(act) were to ask questions and to postpone offer submission. On CRAIGSLISTBARGAIN, when acting as a buyer, 42.4% of its utterances were questions, compared to 30.2% for other models. On both tasks, it tended to wait for the partner to submit an offer (even after a deal was agreed on), compared to RL$_{\text{margin}}$(act) which almost always submitted offers first. For RL$_{\text{fairness}}$(act), it aimed to agree on a price in the middle of the listing price and the buyer's target price for CRAIGSLISTBARGAIN. Since the buyer's target was hidden, when the agent was the seller, it tended to wait for the buyer to propose prices first. Similary, on DEALORNoDeal it waited to hear the parter's offer and sometimes changed its offer afterwards, whereas the other models often insisted on one offer.

On both tasks, $\text{RL}_{\text{utility}}$(act) learned to insist on its offer and refuse to budge. This ended up frustrating many people, which is why it has a low agreement rate. The problem is that our human model is simply a SL model trained on human-human dialogues, which may not accurately reflects real human behavior during human-bot chat. For example, the SL model often agrees after a few turns of insistence on a proposal, whereas humans get annoyed if the partner is not willing to make compromises at all. However, by injecting domain knowledge to SL(act)+rule, e.g., making a small compromise is better than stubbornly being fixed on a single price, we were able to achieve high utility and human-likeness on both CraigslistBargain and DealOrNoDeal.

5 Related Work and Discussion

Recent work has explored the space between goal-oriented dialogue and open-domain chit-chat through collaborative or competitive language games, such as collecting cards in a maze (Potts, 2012), finding a mutual friend (He et al., 2017), or splitting a set of items (DeVault et al., 2015; Lewis et al., 2017). Our CraigslistBargain dialogue falls in this category, but exhibits richer and more diverse language than prior datasets. Our dataset calls for systems that can handle both strategic decision-making and open-ended text generation.

Traditional goal-oriented dialogue systems build a pipeline of modules (Young et al., 2013; Williams et al., 2016). Due to the laborious dialogue state design and annotation, recent work has been exploring ways to replace these modules with neural networks and end-to-end training while still having a logical backbone (Wen et al., 2017a; Bordes and Weston, 2017; He et al., 2017). Our work is closely related to the Hybrid Code Network (Williams et al., 2017), but the key difference is that Williams et al. (2017) uses a neural dialogue state, whereas we keep a structured, interpretable dialogue state which allows for stronger top-down control. Another line of work tackles this problem by introducing latent stochastic variables to model the dialogue state (Wen et al., 2017b; Zhao et al., 2017; Cao and Clark, 2017). While the latent discrete variable allows for post-hoc discovery of dialogue acts and increased utterance diversity, it does not provide controllability over the dialogue strategy.

Our work is also related to a large body of literature on dialogue policies in negotiation (English and Heeman, 2005; Efstathiou and Lemon, 2014; Hiraoka et al., 2015; Cao et al., 2018). These work mostly focus on learning good negotiation policies in a domain-specific action space, whereas our model operates in an open-ended space of natural language. An interesting future direction is to connect with game theory (Brams, 2003) for complex multi-issue bargaining. Another direction is learning to generate persuasive utterances, e.g., through framing (Takuya et al., 2014) or accounting for the social and cultural context (Elnaz et al., 2012).

To conclude, we have introduced CraigslistBargain, a rich dataset of human-human negotiation dialogues. We have also presented a modular approach based on coarse dialogue acts that models a rough strategic backbone as well allowing for open-ended generation. We hope this work will spur more research in hybrid approaches that can work in open-ended, goal-oriented settings.

Acknowledgments. This work is supported by DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF-15-1-0462. We thank members of the Stanford NLP group for insightful discussion and the anonymous reviewers for constructive feedback.

Reproducibility. All code, data, and experiments for this paper are available on the CodaLab platform: `https://worksheets.codalab.org/worksheets/0x453913e76b65495d8b9730d41c7e0a0c/`.

References

S. Afantenos, N. Asher, F. Benamara, A. Cadilhac, C. Dégremont, P. Denis, M. Guhe, S. Keizer, A. Lascarides, O. Lemon, et al. 2012. Modelling strategic conversation: Model, annotation design and corpus. In *Proceedings of SemDial 2012: Workshop on the Semantics and Pragmatics of Dialogue*, pages 167–168.

N. Asher, J. Hunter, M. Morey, F. Benamara, and S. Afantenos. 2016. Discourse structure and dialogue acts in multiparty dialogue: the STAC corpus. In *Language Resources and Evaluation Conference (LREC)*.

A. Bordes and J. Weston. 2017. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations (ICLR)*.

S. J. Brams. 2003. *Negotiation Games: Applying Game Theory to Bargaining and Arbitration*. Psychology Press.

K. Cao and S. Clark. 2017. Latent variable dialogue models and their diversity. In *European Association for Computational Linguistics (EACL)*.

K. Cao, A. Lazaridou, M. Lanctot, J. Z. Leibo, K. Tuyls, and S. Clark. 2018. Emergent communication through negotiation. In *International Conference on Learning Representations (ICLR)*.

H. Cuayáhuitl, S. Keizer, and O. Lemon. 2015. Strategic dialogue management via deep reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*.

D. DeVault, J. Mell, and J. Gratch. 2015. Toward natural turn-taking in a virtual human negotiation agent. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

B. Dhingra, L. Li, X. Li, J. Gao, Y. Chen, F. Ahmed, and L. Deng. 2017. End-to-end reinforcement learning of dialogue agents for information access. In *Association for Computational Linguistics (ACL)*.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

I. Efstathiou and O. Lemon. 2014. Learning noncooperative dialogue behaviours. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

N. Elnaz, G. Kallirroi, and T. David. 2012. A cultural decision-making model for negotiation based on inverse reinforcement learning. In *The Annual Meeting of the Cognitive Science Society*.

M. S. English and P. A. Heeman. 2005. Learning mixed initiative dialog strategies by using reinforcement learning on both conversants. In *Empirical Methods in Natural Language Processing (EMNLP)*.

H. He, A. Balakrishnan, M. Eric, and P. Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Association for Computational Linguistics (ACL)*, pages 1766–1776.

T. Hiraoka, K. Georgila, E. Nouri, and D. Traum. 2015. Reinforcement learning in multi-party trading dialog. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

S. Keizer, M. Guhe, H. Cuayahuitl, I. Efstathiou, K. Engelbrecht, M. Dobre, A. Lascarides, and O. Lemon. 2017. Evaluating persuasion strategies and deep reinforcement learning methods for negotiation dialogue agents. In *European Association for Computational Linguistics (EACL)*.

M. Lewis, D. Yarats, Y. N. Dauphin, D. Parikh, and D. Batra. 2017. Deal or no deal? end-to-end learning for negotiation dialogues. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. 2016. Deep reinforcement learning for dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

R. T. Lowe, N. Pow, I. Serban, L. Charlin, C. Liu, and J. Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue and Discourse*, 8.

J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

C. Potts. 2012. Goal-driven answers in the Cards dialogue corpus. In *Proceedings of the 30th West Coast Conference on Formal Linguistics*, pages 1–20.

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J. Nie, J. Gao, and B. Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *North American Association for Computational Linguistics (NAACL)*.

H. Takuya, N. Graham, S. Sakriani, T. Tomoki, and N. Satoshi. 2014. Reinforcement learning of cooperative persuasive dialogue policies using framing. In *International Conference on Computational Linguistics (COLING)*.

D. Traum, S. C. Marsella, J. Gratch, J. Lee, and A. Hartholt. 2008. Multi-party, multi-issue, multi-strategy negotiation for multi-modal virtual agents. In *International Workshop on Intelligent Virtual Agents*, pages 117–130.

T. Wen, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, P. Su, S. Ultes, D. Vandyke, and S. Young. 2017a. A network-based end-to-end trainable task-oriented dialogue system. In *European Association for Computational Linguistics (EACL)*, pages 438–449.

T. Wen, Y. Miao, P. Blunsom, and S. Young. 2017b. Latent intention dialogue models. In *International Conference on Machine Learning (ICML)*.

J. D. Williams, K. Asadi, and G. Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Association for Computational Linguistics (ACL)*.

J. D. Williams, A. Raux, and M. Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue and Discourse*, 7.

R. J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

S. Young, M. Gašić, B. Thomson, and J. D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. In *Proceedings of the IEEE*, 5, pages 1160–1179.

T. Zhao, R. Zhao, and M. Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Association for Computational Linguistics (ACL)*.

Large-scale Cloze Test Dataset Created by Teachers

Qizhe Xie[*], **Guokun Lai**[*], **Zihang Dai, Eduard Hovy**
Language Technologies Institute, Carnegie Melon University
`{qizhex, guokun, dzihang, hovy}@cs.cmu.edu`

Abstract

Cloze tests are widely adopted in language exams to evaluate students' language proficiency. In this paper, we propose the first large-scale human-created cloze test dataset CLOTH [1] [2], containing questions used in middle-school and high-school language exams. With missing blanks carefully created by teachers and candidate choices purposely designed to be nuanced, CLOTH requires a deeper language understanding and a wider attention span than previously automatically-generated cloze datasets. We test the performance of dedicatedly designed baseline models including a language model trained on the One Billion Word Corpus and show humans outperform them by a significant margin. We investigate the source of the performance gap, trace model deficiencies to some distinct properties of CLOTH, and identify the limited ability of comprehending the long-term context to be the key bottleneck.

1 Introduction

Being a classic language exercise, the cloze test (Taylor, 1953) is an accurate assessment of language proficiency (Fotos, 1991; Jonz, 1991; Tremblay, 2011) and has been widely employed in language examinations. Under a typical setting, a cloze test requires examinees to fill in missing words (or sentences) to best fit the surrounding context. To facilitate natural language understanding, automatically-generated cloze datasets are introduced to measure the ability of machines in reading comprehension (Hermann et al., 2015; Hill et al., 2016; Onishi et al., 2016). In these datasets, each cloze question typically consists of a context paragraph and a question sentence. By randomly replacing a particular word in the question sentence with a blank symbol, a single test case is created. For instance, CNN/Daily Mail datasets (Hermann et al., 2015) use news articles as contexts and summary bullet points as the question sentence. Only named entities are removed when creating the blanks. Similarly, in Children's Books test (CBT) (Hill et al., 2016), cloze questions are obtained by removing a word in the last sentence of every consecutive 21 sentences, with the first 20 sentences being the context. Different from CNN/Daily Mail datasets, CBT also provides each question with a candidate answer set, consisting of randomly sampled words with the same part-of-speech tag from the context as that of the correct answer.

Thanks to the automatic generation process, these datasets can be very large in size, leading to significant research progresses. However, compared to how humans would create cloze questions and evaluate reading comprehension ability, the automatic generation process bears some inevitable issues. Firstly, blanks are chosen uniformly without considering which aspect of the language phenomenon that questions will test. Hence, quite a portion of automatically-generated questions can be purposeless or even trivial to answer. Another issue involves the ambiguity of answers. Given a context and a sentence with a blank, there can be multiple words that fit almost equally well into the blank. A possible solution is to include a candidate option set, as done by CBT, to get rid of the ambiguity. However, automatically generating the candidate option set can be problematic since it cannot guarantee the ambiguity is removed. More importantly, automatically-generated candidates can be totally irrelevant or simply grammatically unsuitable for the blank, resulting in again purposeless or trivial questions.

---

[*] Equal contribution.

[1] CLOTH (**CLO**ze test by **T**eac**H**ers) is available at `http://www.cs.cmu.edu/~glai1/data/cloth/`.

[2] The leaderboard is available at `http://www.qizhexie.com/data/CLOTH_leaderboard.html`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2344–2356
Brussels, Belgium, October 31 - November 4, 2018.

Probably due to these unsatisfactory issues, neural models have achieved comparable results to the human-level performance within a very short time (Chen et al., 2016; Dhingra et al., 2016; Seo et al., 2016). While there have been works trying to incorporate human design into cloze question generation (Zweig and Burges, 2011; Paperno et al., 2016), due to the expensive labeling process, the MSR Sentence Completion Challenge created by this effort has $1,040$ questions and the LAMBADA (Paperno et al., 2016) dataset has $10,022$ questions, limiting the possibility of developing powerful neural models on it. As a result of the small size, human-created questions are only used to compose development sets and test sets. Motivated by the aforementioned drawbacks, we propose CLOTH, a large-scale cloze test dataset collected from English exams. Questions in the dataset are designed by middle-school and high-school teachers to prepare Chinese students for entrance exams. To design a cloze test, teachers firstly determine the words that can test students' knowledge of vocabulary, reasoning or grammar; then replace those words with blanks and provide other three candidate options for each blank. If a question does not specifically test grammar usage, all of the candidate options would complete the sentence with correct grammar, leading to highly nuanced questions. As a result, human-created questions are usually harder and are a better assessment of language proficiency. A general cloze test evaluates several aspects of language proficiency including vocabulary, reasoning and grammar, which are key components of comprehending natural language.

To verify if human-created cloze questions are difficult for current models, we train and evaluate the state-of-the-art language model (LM) and machine comprehension models on this dataset, including a language model trained on the One Billion Word Corpus. We find that the state-of-the-art model lags behind human performance even if the model is trained on a large external corpus. We analyze where the model fails compared to humans who perform well. After conducting error analysis, we assume the performance gap results from the model's inability to use a long-term context. To examine this assumption, we evaluate human-level performance when the human subjects are only allowed to see one sentence as the context. Our assumption is confirmed by the matched performances of the models and human when given only one sentence. In addition, we demonstrate that human-created data is more difficult than automatically-generated data. Specifically, it is much easier for the same model to perform well on automatically-generated data.

We hope that CLOTH provides a valuable testbed for both the language modeling community and the machine comprehension community. Specifically, the language modeling community can use CLOTH to evaluate their models' abilities in modeling long contexts, while the machine comprehension community can use CLOTH to test machine's understanding of language phenomena.

2 Related Work

Large-scale automatically-generated cloze tests (Hermann et al., 2015; Hill et al., 2016; Onishi et al., 2016) lead to significant research advancements. However, generated questions do not consider language phenomenon to be tested and are relatively easy to solve. Recently proposed reading comprehension datasets are all labeled by humans to ensure a high quality (Rajpurkar et al., 2016; Joshi et al., 2017; Trischler et al., 2016; Nguyen et al., 2016).

Perhaps the closet work to CLOTH is the LAMBADA dataset (Paperno et al., 2016). LAMBADA also targets at finding challenging words to test LM's ability in comprehending a longer context. However, LAMBADA does not provide a candidate set for each question, which can cause ambiguities when multiple words can fit in. Furthermore, only test set and development set are labeled manually. The provided training set is the unlabeled Book Corpus (Zhu et al., 2015). Such unlabeled data do not emphasize long-dependency questions and have a mismatched distribution with the test set, as showed in Section 5. Further, the Book Corpus is too large to allow rapid algorithm development for researchers who do not have access to a huge amount of computational power.

Aiming to evaluate machines under the same conditions that the humans are evaluated, there is a growing interest in obtaining data from examinations. NTCIR QA Lab (Shibuki et al., 2014) contains a set of real-world college entrance exam questions. The Entrance Exams task at CLEF QA Track (Peñas et al., 2014; Rodrigo et al., 2015) evaluates machine's reading comprehension ability. The AI2 Reasoning Challenge (Clark et al., 2018; Schoenick et al., 2017) contains approximately eight thousand scientific questions used in middle school. Lai et al. (2017) proposes the first large-scale machine comprehension dataset obtained from exams. They show that questions designed by teachers have a significantly larger proportion of reasoning questions. Our dataset focuses on evaluating both language proficiency and reasoning abilities.

3 CLOTH Dataset

In this section, we introduce the CLOTH dataset that is collected from English examinations, and study its abilities of assessment.

3.1 Data Collection and Statistics

We collect the raw data from three free and public websites in China that gather exams created by English teachers to prepare students for college/high school entrance exams[3]. Before cleaning, there are $20,605$ passages and $332,755$ questions. We perform the following processes to ensure the validity of data: Firstly, we remove questions with an inconsistent format such as questions with more than four options. Then we filter all questions whose validity relies on external information such as pictures or tables. Further, we find that half of the total passages are duplicates and we delete those passages. Lastly, on one of the websites, the answers are stored as images. We use two OCR software programs[4] to extract the answers from images. We discard the questions when results from the two software are different. After the cleaning process, we obtain a clean dataset of $7,131$ passages and $99,433$ questions.

Since high school questions are more difficult than middle school questions, we divide the datasets into CLOTH-M and CLOTH-H, which stand for the middle school part and the high school part. We split 11% of the data for both the test set and the development set. The detailed statistics of the whole dataset and two subsets are presented in Table 1. Note that the questions were created to test non-native speakers, hence the vocabulary size is not very large.

3.2 Question Type Analysis

In order to evaluate students' mastery of a language, teachers usually design tests in a way that questions cover different aspects of a language. Specifically, they first identify words in the passage that can examine students' knowledge in vocabulary, logic, or grammar. Then, they replace the words with blanks and prepare three incorrect but nuanced candidate options to make the test non-trivial. A sample passage is presented in Table 2.

To understand the abilities of assessment on this dataset, we divide questions into several types and label the proportion of each type. According to English teachers who regularly create cloze test questions for English exams in China, there are largely three types: grammar, vocabulary and reasoning. Grammar questions are easily differentiated from other two categories. However, the teachers themselves cannot specify a clear distinction between reasoning questions and vocabulary questions since all questions require comprehending the words within the context and conducting some level of reasoning by recognizing incomplete information or conceptual overlap.

Hence, we divided the questions except grammar questions based on the difficulty level for a machine to answer the question, following works on analyzing machine comprehension datasets (Chen et al., 2016; Trischler et al., 2016). In particular, we divide them in terms of their dependency ranges, since questions that only involve a single sentence are easier to answer than questions involving evidence distributed in multiple sentences. Further, we divided questions involving long-term dependency into matching/paraphrasing questions and reasoning questions since matching questions are easier. The four types include:

- Grammar: The question is about grammar usage, involving tense, preposition usage, active/passive voices, subjunctive mood and so on.
- Short-term-reasoning: The question is about content words and can be answered based on the information within the same sentence. Note that the content words can evaluate knowledge of both vocabulary and reasoning.
- Matching/paraphrasing: The question is answered by copying/paraphrasing a word in the context.

---

[3] The three websites include http://www.21cnjy.com/; http://5utk.ks5u.com/; http://zujuan.xkw.com/. We checked that CLOTH does not contain sentence completion example questions from GRE, SAT and PSAT.

[4] tesseract: https://github.com/tesseract-ocr; ABBYY FineReader: https://www.abbyy.com/en-us/finereader/

Dataset	CLOTH-M			CLOTH-H			CLOTH (Total)		
	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
# passages	2,341	355	335	3,172	450	478	5,513	805	813
# questions	22,056	3,273	3,198	54,794	7,794	8,318	76,850	11,067	11,516
Vocab. size	15,096			32,212			37,235		
Avg. # sentence	16.26			18.92			17.79		
Avg. # words	242.88			365.1			313.16		

Table 1: The statistics of the training, development and test sets of CLOTH-M (middle school questions), CLOTH-H (high school questions) and CLOTH

- Long-term-reasoning: The answer must be inferred from synthesizing information distributed across multiple sentences.

We sample 100 passages in the high school category and the middle school category respectively with totally $3,000$ questions. The types of these questions are labeled on Amazon Turk. We pay $1 and $0.5 for high school passages and middle school passages respectively. We refer readers to Appendix A.1 for details of the labeling processes and the labeled sample passage.

The proportion of different questions is shown in Table 3. The majority of questions are short-term-reasoning questions while approximately 22.4% of the data needs long-term information, in which the long-term-reasoning questions constitute a large proportion.

4 Exploring Models' Limits

In this section, we investigate if human-created cloze test is a challenging problem for state-of-the-art models. We find that LM trained on the One Billion Word Corpus can achieve a remarkable score but cannot solve the cloze test. After conducting an error analysis, we hypothesize that the model is not able to deal with long-term dependencies. We verify the hypothesis by comparing the model's performance with the human performance when the information humans obtain is limited to one sentence.

4.1 Human and Model Performance

LSTM To test the performance of RNN-based supervised models, we train a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to predict the missing word given the context with only labeled data. The implementation details are in Appendix A.3.

Attentive Readers To enable the model to gather information from a longer context, we augment the supervised LSTM model with the attention mechanism (Bahdanau et al., 2014), so that the representation at the blank is used as a query to find the relevant context in the document and a blank-specific representation of the document is used to score each candidate answer. Specifically, we adapt the Stanford Attentive Reader (Chen et al., 2016) and the position-aware attention model (Zhang et al., 2017) to the cloze test problem. With the position-aware attention model, the attention scores are based on both the context match and the distance from a context to the blank. Both attention models are trained only with human-created blanks just as the LSTM model.

LM In cloze test, the context on both sides may be enough to determine the correct answer. Suppose x_i is the missing word and $x_1, \cdots, x_{i-1}$, $x_{i+1}, \cdots, x_n$ are the context, we choose x_i that maximizes the joint probability $p(x_1, \cdots, x_n)$, which essentially maximizes the conditional likelihood $p(x_i \mid x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$. Therefore, LM can be naturally adapted to cloze test.

In essence, LM treats each word as a possible blank and learns to predict it. As a result, it receives more supervision than the LSTM trained on human-labeled questions. Besides training a neural LM on our dataset, interested in whether the state-of-the-art LM can solve cloze test, we also test the LM trained on the One Billion Word Benchmark (Chelba et al., 2013) (referred as 1B-LM) that achieves a perplexity of 30.0 (Jozefowicz et al., 2016)[5]. To make the evaluation time tractable, we limit the context length to one sentence or three sentences. Note that the One Billion Word Corpus does not overlap with the CLOTH

---

[5] The pre-trained model is obtained from https://github.com/tensorflow/models/tree/master/research/lm_1b

Passage: Nancy had just got a job as a secretary in a company. Monday was the first day she went to work, so she was very _1_ and arrived early. She _2_ the door open and found nobody there. ”I am the _3_ to arrive.” She thought and came to her desk. She was surprised to find a bunch of _4_ on it. They were fresh. She _5_ them and they were sweet. She looked around for a _6_ to put them in. ”Somebody has sent me flowers the very first day!” she thought _7_ . ” But who could it be?” she began to _8_ . The day passed quickly and Nancy did everything with _9_ interest. For the following days of the _10_ , the first thing Nancy did was to change water for the followers and then set about her work.
Then came another Monday. _11_ she came near her desk she was overjoyed to see a(n) _12_ bunch of flowers there. She quickly put them in the vase, _13_ the old ones. The same thing happened again the next Monday. Nancy began to think of ways to find out the _14_ . On Tuesday afternoon, she was sent to hand in a plan to the _15_ . She waited for his directives at his secretary's _16_ . She happened to see on the desk a half-opened notebook, which _17_ : ”In order to keep the secretaries in high spirits, the company has decided that every Monday morning a bunch of fresh flowers should be put on each secretarys desk.” Later, she was told that their general manager was a business management psychologist.

Questions:

1.	A. depressed	B. encouraged	**C. excited**	D. surprised
2.	A. turned	**B. pushed**	C. knocked	D. forced
3.	A. last	B. second	C. third	**D. first**
4.	A. keys	B. grapes	**C. flowers**	D. bananas
5.	**A. smelled**	B. ate	C. took	D. held
6.	**A. vase**	B. room	C. glass	D. bottle
7.	A. angrily	B. quietly	C. strangely	**D. happily**
8.	A. seek	**B. wonder**	C. work	D. ask
9.	A. low	B. little	**C. great**	D. general
10.	A. month	B. period	C. year	**D. week**
11.	A. Unless	**B. When**	C. Since	D. Before
12.	A. old	B. red	C. blue	**D. new**
13.	A. covering	B. demanding	**C. replacing**	D. forbidding
14.	**A. sender**	B. receiver	C. secretary	D. waiter
15.	A. assistant	B. colleague	C. employee	**D. manager**
16.	A. notebook	**B. desk**	C. office	D. house
17.	**A. said**	B. written	C. printed	D. signed

Table 2: A Sample passage from our dataset. Bold faces highlight the correct answers. There is only one best answer among four candidates, although several candidates may seem correct.

	Short-term		Long-term		
Dataset	GM	STR	MP	LTR	O
CLOTH	0.265	0.503	0.044	0.180	0.007
CLOTH-M	0.330	0.413	0.068	0.174	0.014
CLOTH-H	0.240	0.539	0.035	0.183	0.004

Table 3: The question type statistics of 3000 sampled questions where GM, STR, MP, LTR and O denotes grammar, short-term-reasoning, matching/paraphrasing, long-term-reasoning and others respectively.

Model	CLOTH	CLOTH-M	CLOTH-H
LSTM	0.484	0.518	0.471
Stanford AR	0.487	0.529	0.471
Position-aware AR	0.485	0.523	0.471
LM	0.548	0.646	0.506
1B-LM (one sent.)	0.695	0.723	0.685
1B-LM (three sent.)	0.707	0.745	0.693
Human performance	0.859	0.897	0.845

Table 4: Models' performance and human-level performance on CLOTH. LSTM, Stanford Attentive Reader and Attentive Reader with position-aware attention shown in the top part only use supervised data labelled by human. LM outperforms LSTM since it receives more supervisions in learning to predict each word. Training on large external corpus further significantly enhances LM's accuracy.

corpus.

Human performance We measure the performance of Amazon Mechanical Turkers on $3,000$ sampled questions when the whole passage is given.

Results The comparison is shown in Table 4. Both attentive readers achieve similar accuracy to the LSTM. We hypothesize that the reason of the attention model's unsatisfactory performance is that the evidence of a question cannot be simply found by matching the context. Similarly, on reading comprehension, though attention-based models (Wang et al., 2017; Seo et al., 2016; Dhingra et al., 2016) have reached human performance on the SQuAD dataset (Rajpurkar et al., 2016), their performance is still not comparable to human performance on datasets that focus more on reasoning where the evidence cannot be simply found by a matching behavior (Lai et al., 2017; Xu et al., 2017). Since the focus of this paper is to analyze the proposed dataset, we leave the design of reasoning oriented attention models for future work.

The LM achieves much better performance than LSTM. The gap is larger when the LM is trained on the 1 Billion Word Corpus, indicating that more training data results in a better generalization. Specifically, the accuracy of 1B-LM is 0.695 when one sentence is used as the context. It indicates that LM can learn sophisticated language regularities when given sufficient data. The same conclusion can also be drawn from the success of a concurrent work ELMo which uses LM representations as word vectors and achieves state-of-the-art results on six language tasks (Peters et al., 2018). However, if we increase the context length to three sentences, the accuracy of 1B-LM only has a marginal improvement. In contrast, humans outperform 1B-LM by a significant margin, which demonstrates that deliberately designed questions in CLOTH are not completely solved even for state-of-the-art models.

4.2 Analyzing 1B-LM's Strengths and Weaknesses

In this section, we would like to understand why 1B-LM lags behind human performance. We find that most of the errors involve long-term reasoning. Additionally, in a lot of cases, the dependency is within the context of three sentences. We show several errors made by the 1B-LM in Table 5. In the first example, the model does not know that Nancy found nobody in the company means that Nancy was the first one to arrive at the company. In the second and third example, the model fails probably because of not recognizing "they" referred to "flowers". The dependency in the last case is longer. It depends on the fact that Nancy was alone in the company.

Based on the case study, we hypothesize that the LM is not able to take long-term information into account, although it achieves a surprisingly good overall performance. Additionally, the 1B-LM is trained on the sentence level, which might also result in the inability to track paragraph level information. However, to investigate the differences between training on sentence level and on paragraph level, a prohibitive amount of computational resource is required to train a large model on the 1 Billion Word Corpus.

On the other hand, a practical comparison is to test the model's performance on different types of questions. We find that the model's accuracy is 0.591 on long-term-reasoning questions of CLOTH-H while it achieves 0.693 on short-term-reasoning (a comprehensive type-specific performance is available in Appendix A.3), which partially confirms that long-term-reasoning is harder. However, we could not completely rely on the performance on specific questions types, partly due to a large variance caused by the small sample size. Another reason is that the reliability of question type labels depends on whether turkers are careful enough. For example, in the error analysis shown in Table 5, a careless turker would label the second example as short-term-reasoning without noticing that the meaning of "they" relies on a long context.

To objectively verify if the LM's strengths lie in dealing with short-term information, we obtain the ceiling performance of only utilizing short-term information. Showing only one sentence as the context, we ask the Turkers to select an option based on their best guesses given the insufficient information. By limiting the context span manually, the ceiling performance with the access to only a short context is estimated accurately.

As shown in Table 6, The performance of 1B-LM using one sentence as the context can almost match the human ceiling performance of only using short-term information. Hence we conclude that the LM can almost perfectly solve all short-term cloze questions. However, the performance of LM is not improved significantly when a long-term context is given, indicating that the performance gap is due to the inability of long-term reasoning.

5 Comparing Human-created Data and Automatically-generated Data

In this section, we demonstrate that human-created data is a better testbed than automatically-generated cloze test since it results in a larger gap between model's performance and human performance.

A casual observation is that a cloze test can be created by randomly deleting words and randomly sampling candidate options. In fact, to generate large-scale data, similar generation processes have been introduced and widely used in machine comprehension (Hermann et al., 2015; Hill et al., 2016; Onishi et al., 2016). However, research on cloze test design (Sachs et al., 1997) shows that tests created by deliberately deleting words are more reliable than tests created by randomly or periodically deleting words. To design accurate language proficiency assessment, teachers usually deliberately select words in order to examine students' proficiency in grammar, vocabulary and reasoning. Moreover, in order to make the question non-trivial, three incorrect options provided by teachers are usually grammatically correct and relevant to the context. For instance, in the fourth problem of the sample passage shown in Table 2, "grapes", "flowers" and "bananas" all fit the description of being fresh.

Hence we naturally hypothesize that human-generated data has distinct characteristics when

Context	Options			
She pushed the door open and found nobody there. "I am the __ to arrive." She thought and came to her desk.	*A. last*	B. second	C. third	**D. first**
They were fresh. She __ them and they were sweet. She looked around for a vase to put them in.	**A. smelled**	*B. ate*	C. took	D. held
She smelled them and they were sweet. She looked around for a __ to put them in. "Somebody has sent me flowers the very first day!"	**A. vase**	*B. room*	C. glass	D. bottle
"But who could it be?" she began to __ . The day passed quickly and Nancy did everything with great interest.	A. seek	**B. wonder**	C. work	*D. ask*

Table 5: Error analysis of 1-billion-language-model with three sentences as the context. The questions are sampled from the sample passage shown in Table 2. The correct answer is in bold text. The incorrectly selected options are in italics.

	Model	CLOTH	CLOTH-M	CLOTH-H
Short context	1B-LM	0.695	0.723	0.685
	Human	0.713	0.771	0.691
Long context	1B-LM	0.707	0.745	0.693
	Human	0.859	0.897	0.845

Table 6: Humans' performance compared with 1-billion-language-model. In the short context part, both 1B-LM and humans only use information of one sentence. In the long context part, humans have the whole passage as the context, while 1B-LM uses contexts of three sentences.

compared with automatically-generated data. To verify this assumption, we compare the LSTM model's performance when given different proportions of the two types of data. Specifically, to train a model with α percent of automatically-generated data, we randomly replace a percent blanks with blanks at random positions, while keeping the remaining $1-\alpha$ percent questions the same. The candidate options for the generated blanks are random words sampled from the unigram distribution. We test models obtained with varying α on human-created data and automatically-generated data respectively.

$\alpha\%$ Test	0%	25%	50%	75%	100%
human-created	0.484	0.475	0.469	0.423	0.381
Generated	0.422	0.699	0.757	0.785	0.815

Table 7: The model's performance when trained on α percent of automatically-generated data and $100-\alpha$ percent of human-created data

From the comparison in Table 7, we have the following observations: (1) human-created data leads to a larger gap between model's performance and the ceiling/human performance. The model's performance and human's performance on the human-created data are 0.484 and 0.859 respectively, as shown in Tab. 4, leading to a gap of 0.376. In comparison, the performance gap on the automatically-generated data is at most 0.185 since the model's performance reaches an accuracy of 0.815 when fully trained on generated data. (2) Although human-created data may provide more information in distinguishing similar words, the distributional mismatch between two types of data makes it non-trivial to transfer the knowledge gained from human-created data to tackle automatically-generated data. Specifically, the model's performance on automatically-generated data monotonically decreases when given a higher ratio of human-created data.

6 Combining Human-created Data with Automatically-generated Data

In Section 4.1, we show that LM is able to take advantage of more supervision since it predicts each word based on the context. At the same time, we also show that human-created data and the automatically-generated data are quite different in Section 5. In this section, we propose a model that takes advantage of both sources.

6.1 Representative-based Model

Specifically, for each question, regardless of being human-created or automatically-generated, we can compute the negative log likelihood of the correct answer as the loss function. Suppose J_{H} is the average negative log likelihood loss for human-created questions and J_{R} is the loss function on generated questions, we combine losses on human-created questions and generated questions by simply adding them together, i.e., $J_{\mathrm{R}} + J_{\mathrm{H}}$ is used as the final loss function. We will introduce the definition of J_{R} in the following paragraphs.

Although automatically-generated data has a large quantity and is valuable to the model training, as shown in the previous Section, automatically-generated questions are quite different from human-created questions. Ideally, a large amount of human-created questions is more desirable than a large amount of automatically-generated questions. A possible avenue towards having large-scale human-created data is to automatically pick out a large number of generated questions which are *representative* of or similar to human-created questions. In other words, we train a network to predict whether a question is a generated question or a human-created question. A generated question is representative of human-created questions if it has a high probability of being a human-created question. Then we can give higher weights to questions that resemble human-created question.

We first introduce our method to obtain the representativeness information. Let x denote the passage and z denote whether a word is selected as a question by human, i.e., z is 1 if this word is selected to be filled in the original passage or 0 otherwise. Suppose h_i is the representation of i-th word given by a bidirectional LSTM. The network computes the probability p_i of x_i being a human-created question as follows:

$$l_i = h_i^T w_{x_i}; \quad p_i = \text{Sigmoid}(l_i)$$

where l_i is the logit which will be used as in the final model and w_{x_i} is the the word embedding. We train the network to minimize the binary cross entropy between p and ground-truth labels at each token.

After obtaining the representativeness information, we define the representativeness weighted loss function as

$$J_{\text{R}} = \sum_{i \notin H} \text{Softmax}_i(\frac{l_1}{\alpha}, \cdots, \frac{l_n}{\alpha}) J_i$$

where J_i denotes the negative log likelihood loss for the $i-$th question and let l_i be the output representativeness of the i-th question and H is the set of all human-generated questions and α is the temperature of the Softmax function. The model degenerates into assigning a uniform weight to all questions when the temperature is $+\infty$. We set α to 2 based on the performance on the dev set. [6].

Model	Ex.	CLOTH	CLOTH-M	CLOTH-H
Our model	No	**0.583**	**0.673**	**0.549**
LM		0.548	0.646	0.506
LSTM		0.484	0.518	0.471
Stanford AR		0.487	0.529	0.471
1B-LM	Yes	0.707	0.745	0.693
Human		0.859	0.897	0.845

Table 8: Overall results on CLOTH. Ex. denotes external data.

Model	CLOTH	CLOTH-M	CLOTH-H
Our model	**0.583**	**0.673**	**0.549**
w.o. rep.	0.566	0.662	0.528
w.o. hum.	0.565	0.665	0.526
w.o. rep. or hum.	0.543	0.643	0.505

Table 9: Ablation study on using the representativeness information (denoted as rep.) and the human-created data (denoted as hum.)

6.2 Results

We summarize performances of all models in Table 8. Our representativeness model outperforms all other models that do not use external data on CLOTH, CLOTH-H and CLOTH-M.

6.3 Analysis

In this section, we verify the effectiveness of the representativeness-based averaging by ablation studies. When we remove the representativeness information by setting α to infinity, the accuracy drops from 0.583 to 0.566. When we further remove the human-created data so that only generated data is employed, the accuracy drops to 0.543, similar to the performance of LM. The results further confirm that it is beneficial to incorporate human-created questions into training.

A sample of the predicted representativeness is shown in Figure 1[7]. Clearly, words that are too obvious have low scores, such as punctuation marks, simple words "a" and "the". In contrast, content words whose semantics are directly related to the context have a higher score, e.g., "same", "similar", "difference" have a high score when the difference between two objects is discussed and "secrets" has a high score since it is related to the subsequent sentence "does not want to share with others". Our prediction model achieves an F1 score of 36.5 on the test set, which is understandable since

---

[6] The code is available at https://github.com/qizhex/Large-scale-Cloze-Test-Dataset-Created-by-Teachers

[7] The script to generate the Figure is obtained at `https://gist.github.com/ihsgnef/f13c35cd46624c8f458a4d23589ac768`

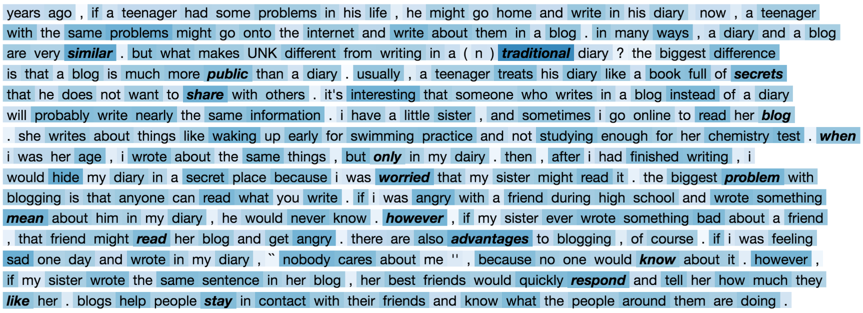

years ago , if a teenager had some problems in his life , he might go home and write in his diary now , a teenager
with the same problems might go onto the internet and write about them in a blog . in many ways , a diary and a blog
are very ***similar*** . but what makes UNK different from writing in a (n) ***traditional*** diary ? the biggest difference
is that a blog is much more ***public*** than a diary . usually , a teenager treats his diary like a book full of ***secrets***
that he does not want to ***share*** with others . it's interesting that someone who writes in a blog instead of a diary
will probably write nearly the same information . i have a little sister , and sometimes i go online to read her ***blog***
. she writes about things like waking up early for swimming practice and not studying enough for her chemistry test . ***when***
i was her age , i wrote about the same things , but ***only*** in my dairy . then , after i had finished writing , i
would hide my diary in a secret place because i was ***worried*** that my sister might read it . the biggest ***problem*** with
blogging is that anyone can read what you write . if i was angry with a friend during high school and wrote something
mean about him in my diary , he would never know . ***however*** , if my sister ever wrote something bad about a friend
, that friend might ***read*** her blog and get angry . there are also ***advantages*** to blogging , of course . if i was feeling
sad one day and wrote in my diary , `` nobody cares about me '' , because no one would ***know*** about it . however ,
if my sister wrote the same sentence in her blog , her best friends would quickly ***respond*** and tell her how much they
like her . blogs help people ***stay*** in contact with their friends and know what the people around them are doing .

Figure 1: Representativeness prediction for each word. Lighter color means less representative. The words deleted by human as blanks are in bold text.

there are many plausible questions within a passage.

It has been shown that features such as morphology information and readability are beneficial in cloze test prediction (Skory and Eskenazi, 2010; Correia et al., 2012, 2010; Kurtasov, 2013). We leave investigating the advanced approaches of automatically designing cloze test to future work.

7 Conclusion and Discussion

In this paper, we propose a large-scale cloze test dataset CLOTH that is designed by teachers. With missing blanks and candidate options carefully created by teachers to test different aspects of language phenomena, CLOTH requires a deep language understanding and better captures the complexity of human language. We find that human outperforms 1B-LM by a significant margin. After detailed analysis, we find that the performance gap is due to the model's inability to understanding a long context. We also show that, compared to automatically-generated questions, human-created questions are more difficult and lead to a larger margin between human performance and the model's performance.

Despite the excellent performance of 1B-LM when compared with models trained only on CLOTH, it is still important to investigate and create more effective models and algorithms which provide complementary advantages to having a large amount of data. For rapid algorithm developments, we suggest training models only on the training set of CLOTH and comparing with models that do not utilize external data.

We hope our dataset provides a valuable testbed to the language modeling community and the machine comprehension community. In particular, the language modeling community can use CLOTH to evaluate their models' abilities in modeling a long context. In addition, the machine comprehension community may also find CLOTH useful in evaluating machine's understanding of language phenomena including vocabulary, reasoning and grammar, which are key components of comprehending natural language.

In our future work, we would like to design algorithms to better model a long context, to utilize external knowledge, and to explore more effective semi-supervised learning approaches. Firstly, we would like to investigate efficient ways of utilizing external knowledge such as paraphrasing and semantic concepts like prior works (Dong et al., 2017; Dasigi et al., 2017). In comparison, training on a large external dataset is actually a time-consuming way of utilizing external knowledge. Secondly, to use the generated questions more effectively, the representative-based semi-supervised approach might be improved by techniques studied in active learning and hard example mining (Settles, 2009; Shrivastava et al., 2016; Chang et al., 2017).

Acknowledgement

We thank Yulun Du, Kaiyu Shi and Zhilin Yang for insightful discussions and suggestions on the draft. We thank Shi Feng for the script to highlight representative words. This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. 2017. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, pages 1003–1013.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical lming. *arXiv preprint arXiv:1312.3005*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Rui Correia, Jorge Baptista, Maxine Eskenazi, and Nuno J Mamede. 2012. Automatic generation of cloze question stems. In *PROPOR*, pages 168–178. Springer.

Rui Correia, Jorge Baptista, Nuno Mamede, Isabel Trancoso, and Maxine Eskenazi. 2010. Automatic generation of cloze question distractors. In *Proceedings of the Interspeech 2010 Satellite Workshop on Second Language Studies: Acquisition, Learning, Education and Technology, Waseda University, Tokyo, Japan*.

Pradeep Dasigi, Waleed Ammar, Chris Dyer, and Eduard Hovy. 2017. Ontology-aware token embeddings for prepositional phrase attachment. *arXiv preprint arXiv:1705.02925*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. *arXiv preprint arXiv:1708.06022*.

Sandra S Fotos. 1991. The cloze test as an integrative measure of efl proficiency: A substitute for essays on college entrance examinations? *Language Learning*, 41(3):313–336.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. *ICLR*.

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

Jon Jonz. 1991. Cloze item types and second language comprehension. *Language testing*, 8(1):1–22.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *ACL*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of lming. *arXiv preprint arXiv:1602.02410*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Andrey Kurtasov. 2013. A system for generating cloze test items from russian-language text. In *Proceedings of the Student Research Workshop associated with RANLP 2013*, pages 107–112.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *EMNLP*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. *arXiv preprint arXiv:1608.05457*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Anselmo Peñas, Yusuke Miyao, Álvaro Rodrigo, Eduard H Hovy, and Noriko Kando. 2014. Overview of clef qa entrance exams task 2014. In *CLEF (Working Notes)*, pages 1194–1200.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Álvaro Rodrigo, Anselmo Peñas, Yusuke Miyao, Eduard H Hovy, and Noriko Kando. 2015. Overview of clef qa entrance exams task 2015. In *CLEF (Working Notes)*.

J Sachs, P Tung, and RYH Lam. 1997. How to construct a cloze test: Lessons from testing measurement theory models. *Perspectives*.

Carissa Schoenick, Peter Clark, Oyvind Tafjord, Peter Turney, and Oren Etzioni. 2017. Moving beyond the turing test with the allen ai science challenge. *Communications of the ACM*, 60(9):60–64.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Burr Settles. 2009. Active learning literature survey.

Hideyuki Shibuki, Kotaro Sakamoto, Yoshinobu Kano, Teruko Mitamura, Madoka Ishioroshi, Kelly Y Itakura, Di Wang, Tatsunori Mori, and Noriko Kando. 2014. Overview of the ntcir-11 qa-lab task. In *NTCIR*.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. 2016. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769.

Adam Skory and Maxine Eskenazi. 2010. Predicting cloze task quality for vocabulary training. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 49–56. Association for Computational Linguistics.

Wilson L Taylor. 1953. cloze procedure: a new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Annie Tremblay. 2011. Proficiency assessment standards in second language acquisition research. *Studies in Second Language Acquisition*, 33(3):339–372.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Yichong Xu, Jingjing Liu, Jianfeng Gao, Yelong Shen, and Xiaodong Liu. 2017. Towards human-level machine reading comprehension: Reasoning and inference with multiple strategies. *arXiv preprint arXiv:1711.04964*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Geoffrey Zweig and Christopher JC Burges. 2011. The microsoft research sentence completion challenge. Technical report, Technical Report MSR-TR-2011-129, Microsoft.

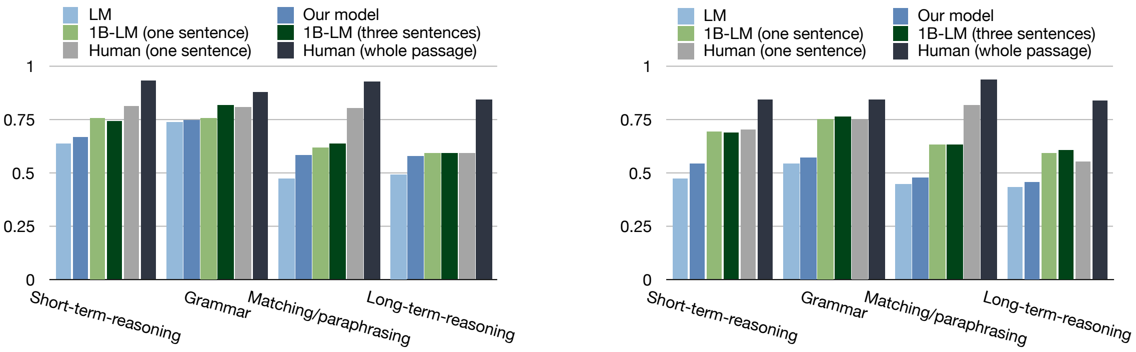

(a) Middle school group (CLOTH-M)

(b) High school group (CLOTH-H)

Figure 2: Model and human's performance on questions with different types. Our model will be introduced in Sec. 6.

A Appendix

A.1 Question Type Labeling

To label the questions, we provided the definition and an example for each question category to the Amazon Mechanical Turkers. To ensure quality, we limited the workers to master Turkers who are experienced and maintain a high acceptance rate. However, we did not restrict the backgrounds of the Turkers since master Turkers should have a reasonable amount of knowledge about English to conduct previous tasks. In addition, the vocabulary used in CLOTH are usually not difficult since they are constructed to test non-native speakers in middle school or high school. To get a concrete idea of the nature of question types, please refer to examples shown in Tab. 10.

A.2 Type-specific Performance Analysis

We can also further verify the strengths and weaknesses of the 1B-LM by studying the performance of models and human on different question categories. Note that the performance presented here may be subject to a high variance due to the limited number of samples in each category. From the comparison shown in Figure 2, we see that 1B-LM is indeed good at short-term questions. Specifically, when the human only has access to the context of one sentence, 1B-LM is close to human's performance on almost all categories. Further, comparing LM and 1B-LM, we find that training on the large corpus leads to improvements on all categories, showing that training on a large amount of data leads to a substantial improvement in learning complex language regularities.

A.3 Implementation Details

We implement our models using PyTorch (Paszke et al., 2017). We train our model on all questions in CLOTH and test it on CLOTH-M and CLOTH-H separately. For our final model, we use Adam (Kingma and Ba, 2014) with the learning rate of 0.001. The hidden dimension is set to 650 and we initialize the word embedding by 300-dimensional Glove word vector (Pennington et al., 2014). The temperature α is set to 2. We tried to increase the dimensionality of the model but do not observe performance improvement.

When we train the small LM on CLOTH, we largely follow the recommended hyperparameters in the Pytorch LM example[8]. Specifically, we employ a 2-layer LSTM with hidden dimension as 1024. The input embedding and output weight matrix are tied. We set the dropout rate to 0.5. The initial learning rate is set to 10 and divided by 4 whenever the PPL stops improving on the dev set.

We predict the answer for each blank independently for all of the models mentioned in this paper, since we do not observe significant performance improvements in our preliminary experiments when an auto-regressive approach is employed, i.e., when we fill all previous blanks with predicted answers. We hypothesize that, regardless of whether there exist inter-blank dependencies, since blanks are usually

---

[8] https://github.com/pytorch/examples/tree/master/word_language_model

distributed far away from each other, LSTM is not able to capture such long dependencies. When testing language models, we use the longest text spans that do not contain blanks.

Passage: Nancy had just got a job as a secretary in a company. Monday was the first day she went to work, so she was very _1_ and arrived early. She _2_ the door open and found nobody there. ”I am the _3_ to arrive.” She thought and came to her desk. She was surprised to find a bunch of _4_ on it. They were fresh. She _5_ them and they were sweet. She looked around for a _6_ to put them in. ”Somebody has sent me flowers the very first day!” she thought _7_ . ” But who could it be?” she began to _8_ . The day passed quickly and Nancy did everything with _9_ interest. For the following days of the _10_ , the first thing Nancy did was to change water for the followers and then set about her work.

Then came another Monday. _11_ she came near her desk she was overjoyed to see a(n) _12_ bunch of flowers there. She quickly put them in the vase, _13_ the old ones. The same thing happened again the next Monday. Nancy began to think of ways to find out the _14_ . On Tuesday afternoon, she was sent to hand in a plan to the _15_ . She waited for his directives at his secretary's _16_ . She happened to see on the desk a half-opened notebook, which _17_ : ”In order to keep the secretaries in high spirits, the company has decided that every Monday morning a bunch of fresh flowers should be put on each secretarys desk.” Later, she was told that their general manager was a business management psychologist.

	Questions				Question type
1.	A. depressed	B. encouraged	**C. excited**	D. surprised	short-term reasoning
2.	A. turned	**B. pushed**	C. knocked	D. forced	short-term reasoning
3.	A. last	B. second	C. third	**D. first**	long-term reasoning
4.	A. keys	B. grapes	**C. flowers**	D. bananas	matching
5.	**A. smelled**	B. ate	C. took	D. held	short-term reasoning
6.	**A. vase**	B. room	C. glass	D. bottle	long-term reasoning
7.	A. angrily	B. quietly	C. strangely	**D. happily**	short-term reasoning
8.	A. seek	**B. wonder**	C. work	D. ask	long-term reasoning
9.	A. low	B. little	**C. great**	D. general	long-term reasoning
10.	A. month	B. period	C. year	**D. week**	long-term reasoning
11.	A. Unless	**B. When**	C. Since	D. Before	grammar
12.	A. old	B. red	C. blue	**D. new**	long-term reasoning
13.	A. covering	B. demanding	**C. replacing**	D. forbidding	long-term reasoning
14.	**A. sender**	B. receiver	C. secretary	D. waiter	long-term reasoning
15.	A. assistant	B. colleague	C. employee	**D. manager**	matching
16.	A. notebook	**B. desk**	C. office	D. house	matching
17.	**A. said**	B. written	C. printed	D. signed	grammar

Table 10: An Amazon Turker's label for the sample passage

emrQA: A Large Corpus for Question Answering on Electronic Medical Records

Anusri Pampari[*] **Preethi Raghavan**[†♣] **Jennifer Liang**[†] **and Jian Peng**[*‡]
[♣]MIT-IBM Watson AI Lab, Cambridge, MA
[†]IBM TJ Watson Research Center, Yorktown Heights, NY
[*]Dept. of Computer Science, University of Illinois Urbana Champaign, IL
[‡]Carle Illinois College of Medicine, University of Illinois Urbana Champaign, IL
[*]`{pampari2,jianpeng}@illinois.edu` [†]`{praghav,jjliang}@us.ibm.com`

Abstract

We propose a novel methodology to generate domain-specific large-scale question answering (QA) datasets by re-purposing existing annotations for other NLP tasks. We demonstrate an instance of this methodology in generating a large-scale QA dataset for electronic medical records by leveraging existing expert annotations on clinical notes for various NLP tasks from the community shared i2b2 datasets[§]. The resulting corpus (emrQA) has 1 million questions-logical form and 400,000+ question-answer evidence pairs. We characterize the dataset and explore its learning potential by training baseline models for question to logical form and question to answer mapping.

1 Introduction

Automatic question answering (QA) has made big strides with several open-domain and machine comprehension systems built using large-scale annotated datasets (Voorhees et al., 1999; Ferrucci et al., 2010; Rajpurkar et al., 2016; Joshi et al., 2017). However, in the clinical domain this problem remains relatively unexplored. Physicians frequently seek answers to questions from unstructured electronic medical records (EMRs) to support clinical decision-making (Demner-Fushman et al., 2009). But in a significant majority of cases, they are unable to unearth the information they want from EMRs (Tang et al., 1994). Moreover to date, there is no general system for answering natural language questions asked by physicians on a patient's EMR (Figure 1) due to lack of large-scale datasets (Raghavan and Patwardhan, 2016).

EMRs are a longitudinal record of a patient's health information in the form of unstructured clinical notes (progress notes, discharge summaries etc.) and structured vocabularies. Physicians wish to answer questions about medical entities and relations from the EMR, requiring a deeper understanding of clinical notes. While this may be likened to machine comprehension, the longitudinal nature of clinical discourse, little to no redundancy in facts, abundant use of domain-specific terminology, temporal narratives with multiple related diseases, symptoms, medications that go back and forth in time, and misspellings, make it complex and difficult to apply existing NLP tools (Demner-Fushman et al., 2009; Raghavan and Patwardhan, 2016). Moreover, answers may be implicit or explicit and may require domain-knowledge and reasoning across clinical notes. Thus, building a credible QA system for patient-specific EMR QA requires large-scale question and answer annotations that sufficiently capture the challenging nature of clinical narratives in the EMR. However, serious privacy concerns about sharing personal health information (Devereaux, 2013; Krumholz et al., 2016), and the tedious nature of assimilating answer annotations from across longitudinal clinical notes, makes this task impractical and possibly erroneous to do manually (Lee et al., 2017).

Record Date: 08/09/98

08/31/96 ascending aortic root replacement with homograft with omentopexy. The patient continued to be hemodynamically stable making good progress. Physical examination: BMI: 33.4 Obese, high risk. Pulse: 60. resp. rate: 18

Question: Has the patient ever had an abnormal BMI?

Answer: BMI: 33.4 Obese, high risk

Question: When did the patient last receive a homograft replacement ?

Answer: 08/31/96 ascending aortic root replacement with homograft with omentopexy.

Figure 1: Question-Answer pairs from emrQA clinical note.

In this work, we address the lack of any publicly available EMR QA corpus by creating a large-scale dataset, emrQA, using a novel gener-

---

[*]This work was conducted during an internship at IBM

[§]https://www.i2b2.org/NLP/DataSets/

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2357–2368
Brussels, Belgium, October 31 - November 4, 2018.

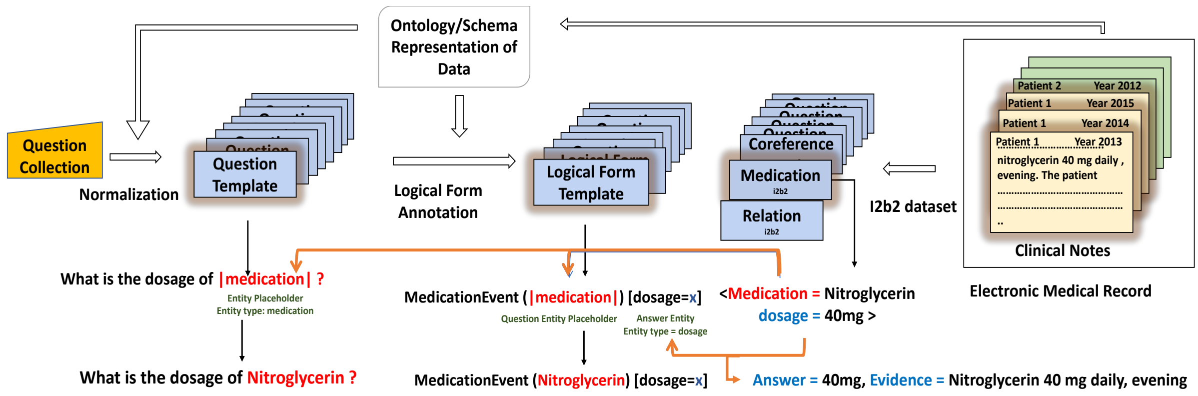

Figure 2: Our QA dataset generation framework using existing i2b2 annotations on a given patient's record to generate a question, its logical form and answer evidence. The highlights in the figure show the annotations being used for this example.

ation framework that allows for minimal expert involvement and re-purposes existing annotations available for other clinical NLP tasks (i2b2 challenge datasets (Guo et al., 2006)). The annotations serve as a proxy-expert in generating questions, answers, and logical forms. Logical forms provide a human-comprehensible symbolic representation, linking questions to answers, and help build interpretable models, critical to the medical domain (Davis et al., 1977; Vellido et al., 2012). We analyze the emrQA dataset in terms of question complexity, relations, and the reasoning required to answer questions, and provide neural and heuristic baselines for learning to predict question-logical forms and question-answers.

The main contributions of this work are as follows:

- A novel framework for systematic generation of domain-specific large-scale QA datasets that can be used in any domain where manual annotations are challenging to obtain but limited annotations may be available for other NLP tasks.

- The first accessible patient-specific EMR QA dataset, emrQA[*], consisting of 400,000 question-answer pairs and 1 million question-logical form pairs. The logical forms will allow users to train and benchmark interpretable models that justify answers with corresponding logical forms.

- Two new reasoning challenges, namely arithmetic and temporal reasoning, that are absent in open-domain datasets like SQuAD (Rajpurkar et al., 2016).

2 Related Work

Question Answering (QA) datasets are classified into two main categories: (1) machine comprehension (MC) using unstructured documents, and (2) QA using Knowledge Bases (KBs).

MC systems aim to answer any question that could be posed against a reference text. Recent advances in crowd-sourcing and search engines have resulted in an explosion of large-scale (100K) MC datasets for factoid QA, having ample redundant evidence in text (Rajpurkar et al., 2016; Trischler et al., 2016; Joshi et al., 2017; Dhingra et al., 2017). On the other hand, complex domain-specific MC datasets such as MCTest (Richardson et al., 2013), biological process modeling (Berant et al., 2014), BioASQ (Tsatsaronis et al., 2015), InsuranceQA (Feng et al., 2015), etc have been limited in scale (500-10K) because of the complexity of the task or the need for expert annotations that cannot be crowd-sourced or gathered from the web. In contrast to the open-domain, EMR data cannot be released publicly due to privacy concerns (Šuster et al., 2017). Also, annotating unstructured EMRs requires a medical expert who can understand and interpret clinical text. Thus, very few datasets like i2b2, MIMIC (Johnson et al., 2016) (developed over several years in collaboration with large medical groups and hospitals), share small-scale annotated clinical notes. In this work, we take advantage of the limited expertly annotated resources to generate emrQA.

KB-based QA datasets, used for semantic parsing, are traditionally limited by the requirement of annotated question and logical form (LF) pairs for supervision where the LF are used to retrieve answers from a schema (Cai and Yates, 2013; Lopez et al., 2013; Bordes et al., 2015). Roberts and Demner-Fushman (2016) generated a corpus by

---

[*] https://github.com/panushri25/emrQA, scripts to generate emrQA from i2b2 data. i2b2 data is accessible by everyone subject to a license agreement.

Datasets	#QA	#QL	#notes	Property	Stats.
Relations	141,243	1,061,710	425	Question len.	8.6
Medications	255,908	198,739	261	Evidence len.	18.7
Heart disease	30,731	36,746	119	LF len.	33
Obesity	23,437	280	1,118	Note len.	3825
Smoking	4,518	6	502	# of evidence	1.5
emrQA	455,837	1,295,814	2,425	# Ques. in note	187

Table 1: (left) i2b2 dataset distribution in emrQA, and (right) emrQA properties with length in tokens, averaged

How was the $|problem|$ managed ?
How was the patient's $|problem|$ treated ?
What was done to correct the patient's $|problem|$?
Has the patient ever been treated for a $|problem|$?
What treatment has the patient had for his $|problem|$?
Has the patient ever received treatment for $|problem|$?
What treatments for $|problem|$ has this patient tried ?

Table 2: Paraphrase templates of a question type in emrQA.

manually annotating LFs on 468 EMR questions (not released publicly), thus limiting its ability to create large scale datasets. In contrast, we only collect LFs for question templates from a domain-expert - the rest of our corpus is automatically generated.

Recent advances in QA combine logic-based and neural MC approaches to build hybrid models (Usbeck et al., 2015; Feng et al., 2016; Palangi et al., 2018). These models are driven to combine the accuracy of neural approaches (Hermann et al., 2015) and the interpretability of the symbolic representations in logic-based methods (Gao et al.; Chabierski et al., 2017). Building interpretable yet accurate models is extremely important in the medical domain (Shickel et al., 2017). We generate large-scale ground truth annotations (questions, logical forms, and answers) that can provide supervision to learn such hybrid models. Our approach to generating emrQA is in the same spirit as Su et al. (2016), who generate graph queries (logical forms) from a structured KB and use them to collect answers. In contrast, our framework can be applied to generate QA dataset in any domain with minimal expert input using annotations from other NLP tasks.

3 QA Dataset Generation Framework

Our general framework for generating a large-scale QA corpus given certain resources consists of three steps: (1) collecting questions to capture domain-specific user needs, followed by normalizing the collected questions to templates by replacing entities (that may be related via binary or composite relations) in the question with placeholders. The entity types replaced in the question are grounded in an ontology like WordNet (Miller, 1995), UMLS (Bodenreider, 2004), or a user-generated schema that defines and relates different entity types. (2) We associate question templates with expert-annotated logical form templates; logical forms are symbolic representations using relations from the ontology/schema to express the relations in the question, and associate the question entity type with an answer entity type. (3) We then proceed to the important step of re-purposing existing NLP annotations to populate question-logical form templates and generate answers. QA is a complex task that requires addressing several fundamental NLP problems before accurately answering a question. Hence, obtaining expert manual annotations in complex domains is infeasible as it is tedious to expert-annotate answers that may be found across long document collections (e.g., longitudinal EMR) (Lee et al., 2017). Thus, we reverse engineer the process where we reuse expert annotations available in NLP tasks such as entity recognition, coreference, and relation learning, based on the information captured in the logical forms to populate entity placeholders in templates and generate answers. Reverse engineering serves as a proxy expert ensuring that the generated QA annotations are credible. The only manual effort is in annotating logical forms, thus significantly reducing expert labor. Moreover, in domain specific instances such as EMRs, manually annotated logical forms allow the experts to express information essential for natural language understanding such as domain knowledge, temporal relations, and negation (Gao et al.; Chabierski et al., 2017). This knowledge, once captured, can be used to generate QA pairs on new documents, making the framework scalable.

4 Generating the emrQA Dataset

We apply the proposed framework to generate the emrQA corpus consisting of questions posed by physicians against longitudinal EMRs of a patient, using annotations provided by i2b2 (Figure 2).

4.1 Question Collection and Normalization

We collect questions for EMR QA by, 1) polling physicians at the Veterans Administration for what they frequently want to know from the EMR (976 questions), 2) using an existing source of 5,696 questions generated by a team of medical experts from 71 patient records (Raghavan et al., 2017) and 3) using 15 prototypical questions from an ob-

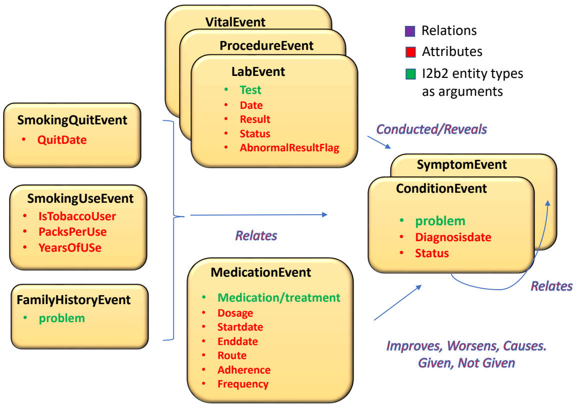

Figure 3: Events, attributes & relations in emrQA's logical forms. Events & attributes accept i2b2 entities as arguments.

servational study done by physicians (Tang et al., 1994). To obtain templates, the questions were automatically normalized by identifying medical entities (using MetaMap (Aronson, 2001)) in questions and replacing them with generic placeholders. The resulting ∼2K noisy templates were expert reviewed and corrected (to account for any entity recognition errors by MetaMap). We align our entity types to those defined in the i2b2 concept extraction tasks (Uzuner et al., 2010a, 2011) - *problem, test, treatment, mode and medication.* E.g., The question *What is the dosage of insulin?* from the collection gets converted to the template *What is the dosage of* $|medication|$*?* as shown in Fig.2. This process resulted in 680 question templates. We do not correct for the usage/spelling errors in these templates, such as usage of "pt" for "patient", or make the templates gender neutral in order to provide a true representation of physicians' questions. Further, analyzing these templates shows that physicians most frequently ask about test results (11%), medications for problem (9%), and problem existence (8%). The long tail following this includes questions about medication dosage, response to treatment, medication duration, prescription date, etiology, etc. Temporal constraints were frequently imposed on questions related to tests, problem diagnosis and medication start/stop.

4.2 Associating Templates w/ Logical Forms

The 680 question templates were annotated by a physician with their corresponding logical form (LF) templates, which resulted in 94 unique LF templates. More than one question template that map to the same LF are considered paraphrases of each other and correspond to a particular question type (Table 2). Logical forms are defined based on an ontology schema designed by medical experts (Figure 3). This schema captures entities in unstructured clinical notes through medical events and their attributes, interconnected through relations. We align the entity and relation types of i2b2 to this schema.

A formal representation of the LF grammar using this schema (Figure 3) is as follows. Medical events are denoted as ME_i (e.g LabEvent, ConditionEvent) and relations are denoted as RE_i (e.g conducted/reveals). Now, $ME[a_1, .., a_j, .., oper(a_n)]$ is a medical event where a_j represents the attribute of the event (such as result in LabEvent). An event may optionally include constraints on attributes captured by an operator ($oper() \in$ sort, range, check for null values, compare). These operators sometimes require values from external medical KB (indicated by *ref*, e.g. lab.*ref*low/lab.*ref*high to indicate range of reference standards considered healthy in lab results) indicating the need for medical knowledge to answer the question. Using these constructs, a LF can be defined using the following rules,

$LF \rightarrow ME_i \mid M_1$ relation M_2
$M_1 \rightarrow ME_i$, $M_2 \rightarrow ME_j$
$M_1 \rightarrow M_1$ relation M_2, $M_2 \rightarrow M_1$ relation M_2
$relation \rightarrow OR \mid AND \mid RE_i$

Advantages of our LF representation include the ability to represent composite relations, define attributes for medical events and constrain the attributes to precisely capture the information need in the question. While these can be achieved using different methods that combine lambda calculus and first order logic (Roberts and Demner-Fushman, 2016), our representation is more human comprehensible. This allows a physician to consider an ontology like Figure 3 and easily define a logical form. Some example question templates with their LF annotations are described in Table 3 using the above notation. The LF representation of the question in Figure 2 is *MedicationEvent(*$|medication|$*) [dosage=x]*. The entities seen in LF are the entities posed in the question and entity marked x indicates the answer entity type.

4.3 Template Filling and Answer Extraction

The next step in the process is to populate the question and logical form (QL) templates with existing annotations in the i2b2 clinical datasets and extract answer evidence for the questions.

Property	Example Annotation	Stats.
Fine grained answer type (attribute entity is answer)	Q: What is the **dosage** of $\vert medication\vert$? LF: MedicationEvent ($\vert medication\vert$) **[dosage=x]**	62.7%
Course grained answer type (event entity is answer)	Q: **What** does the patient take $\vert medication\vert$ for? LF: MedicationEvent($\vert medication\vert$)given{ConditionEvent(**x**) OR SymptomEvent(**x**)}	52.1%
Questions with operators on entities	Q: What are the last set of labs with **elevated** numbers out of range? LF: LabEvent (x) [date=x, (result=x)>lab.refhigh]	25.5%
Questions which require medical KB	Q: What are the last set of labs with **elevated numbers out of range**? LF: LabEvent (x) [date=x, (result=x)>**lab.refhigh**]	11.7%
At least one event relation	What lab results does he have that are pertinent to $\vert problem\vert$ **diagnosis** LF: LabEvent (x) [date=x, result=x] **conducted/reveals** ConditionEvent ($\vert problem\vert$)	46.8%

Table 3: Properties of question templates inferred from the corresponding logical form templates. The boldface words hint at the presence of the corresponding property in both question and the logical form template.

The i2b2 datasets are expert annotated with fine-grained annotations (Guo et al., 2006) that were developed for various shared NLP challenge tasks, including (1) smoking status classification (Uzuner et al., 2008), (2) diagnosis of obesity and its co-morbidities (Uzuner, 2009), extraction of (3) medication concepts (Uzuner et al., 2010a), (4) relations, concepts, assertions (Uzuner et al., 2010b, 2011) (5) co-reference resolution (Uzuner et al., 2012) and (6) heart disease risk factor identification (Stubbs and Uzuner, 2015). In Figure 2, this would correspond to leveraging annotations from medications challenge between medications and their dosages, such as *medication=Nitroglycerin, dosage=40mg*, to populate |medication| and generate several instances of the question "What is the dosage of $|medication|$?" and its corresponding logical form *MedicationEvent(*$|medication|$*)[dosage=x]*. The answer would be derived from the value of the dosage entity in the dataset.

Preprocessing: The i2b2 entities are preprocessed before using them with our templates to ensure syntactic correctness of the generated questions. The pre-processing steps are designed based on the i2b2 annotations syntax guidelines (Guo et al., 2006). To estimate grammatical correctness, we randomly sampled 500 generated questions and found that <5% had errors. These errors include, among others, incorrect usage of article with the entity and incorrect entity phrasing.

Answer Extraction: The final step in the process is generating answer evidence corresponding to each question. The answers in emrQA are defined differently; instead of a single word or phrase we provide the entire i2b2 annotation line from the clinical note as the answer. This is because the context in which the answer entity or phrase is mentioned is extremely important in clinical decision making (Demner-Fushman et al., 2009). Hence, we call them answer evidence instead of just answers. For example, consider the question *Is the patient's* $hypertension$ *controlled?*. The answer to this question is not a simple *yes/no* since the status of the patient's $hypertension$ can change through the course of treatment. The answer evidence to this question in emrQA are multiple lines across the longitudinal notes that reflect this potentially changing status of the patients condition, e.g. *Hypertension-borderline today*. Additionally, for questions seeking specific answers we also provide the corresponding answer entities.

The overall process for answer evidence generation was vetted by a physician. Here is a brief overview of how the different i2b2 datasets were used in generating answers. The *relations challenge* datasets have various event-relation annotations across single/multiple lines in a clinical note. We used a combination of one or more of these, to generate answers for a question; in doing so we used the annotations provided by the *i2b2 co-reference* datasets. Similarly, the *medications challenge* dataset has various event-attribute annotations but since this dataset is not provided with co-reference annotations, it is currently not possible to combine all valid answers. The *heart disease challenge dataset* has longitudinal notes (~5 per patient) with record dates. The events in this dataset are also provided with time annotations and are rich in quantitative entities. This dataset was primarily used to answer questions that require temporal and arithmetic reasoning on events. The patient records in the *smoking and obesity challenge* datasets are categorized into classes with no entity annotations. Thus, for questions generated on these datasets, the entire document acts as evidence and the annotated class information (7 classes) needs to be predicted as the answer.

The total questions, LFs and answers generated using this framework are summarized in Table 1. Consider the question *How much does the patient smoke?* for which we do not have i2b2-annotations to provide an answer. In cases where the answer entity is empty, we only generate the question and LF, resulting in more question types being used for QL than QA pairs: only 53% of question types have answers.

5 emrQA Dataset Analysis

We analyze the complexity of emrQA by considering the LFs for question characteristics, variations in paraphrases, and the type of reasoning required for answering questions (Table 2, 3, 4).

5.1 Question/Logical Form Characteristics

A quantitative and qualitative analysis of emrQA question templates is shown in Table 3, where logical forms help formalize their characteristics (Su et al., 2016). Questions may request specific fine-grained information (attribute values like dosage) or may express a more coarse-grained need (event entities like medications etc), or a combination of both. 25% of questions require complex operators (e.g compare(>)) and 12% of questions express the need for external medical knowledge (e.g. lab.refhigh). The questions in emrQA are highly compositional, where 47% of question templates have at least one event relation.

5.2 Paraphrase Complexity Analysis

Questions templates that map to the same LF are considered paraphrases (e.g, Table 2) and correspond to the same question type. In emrQA, an average of 7 paraphrase templates exist per question type. This is representative of FAQ types that are perhaps more important to the physician. Good paraphrases are lexically dissimilar to each other (Chen and Dolan, 2011). In order to understand the lexical variation within our paraphrases, we randomly select a question from the list of paraphrases as a reference and evaluate the others with respect to the reference, and report the average BLEU (0.74 $\pm$ 0.06) and Jaccard Score (0.72 $\pm$ 0.19). The low BLEU and Jaccard score with large standard deviation indicates the lexical diversity captured by emrQA's paraphrases (Papineni et al., 2002; Niwattanakul et al., 2013).

5.3 Answer Evidence Analysis

33% of the questions in emrQA have more than one answer evidence, with the number ranging from 2 to 61. E.g., the question *Medications Record?* has all medications in the patient's longitudinal record as answer evidence. In order to analyze the reasoning required to answer emrQA questions, we sampled 35 clinical notes from the corpus and analyzed 3 random questions per note by manually labeling them with the categories described in Table 4. Categories are not mutually exclusive: a single example can fall into multiple categories. We compare and contrast this analysis with SQuAD (Rajpurkar et al., 2016), a popular MC dataset generated through crowdsourcing, to show that the framework is capable of generating a corpus as representative and even more complex. Compared to SQuAD, emrQA offers two new reasoning categories, temporal and arithmetic which make up 31% of the dataset. Additionally, over two times as many questions in emrQA require reasoning over multiple sentences. Long and noisy documents make the question answering task more difficult (Joshi et al., 2017). EMRs are inherently noisy and hence 29% have incomplete context and the document length is 27 times more than SQuAD which offers new challenges to existing QA models. Owing to the domain specific nature of the task, 39% of the examples required some form of medical/world knowledge.

As discussed in Section 4.3, 12% of the questions in emrQA corpus require a class category from *i2b2 smoking and obesity datasets* to be predicted. We also found 6% of the questions had other possible answers that were not included by emrQA, this is because of the lack of co-reference annotations for the *medications challenge*.

6 Baseline Methods

We implement baseline models using neural and heuristic methods for question to logical form (Q-L) and question to answer (Q-A) mapping.

6.1 Q-L Mapping

Heuristic Models: We use a template-matching approach where we first split the data into train/test sets, and then normalize questions in the test set into templates by replacing entities with placeholders. The templates are then scored against the ground truth templates of the questions in the train set, to find the best match. The placeholders in the LF template corresponding to the best matched question template is then filled with the normalized entities to obtain the predicted LF. To normalize the test questions we use CLiNER

Reasoning	Description	Example Annotation	emrQA	SQuAD
Lexical Variation (Synonym)	Major correspondence between the question and answer sentence are synonyms.	Q: Has this patient ever been **treated** with insulin? E: Patient sugars were **managed** o/n with sliding scale insulin and diabetic	15.2%	33.3%
Lexical Variation (world/medical knowledge)	Major correspondence between the question and answer sentence requires world/medical knowledge to resolve	Q: Has the patient complained of any **CAD symptoms**? E: 70-year-old female who comes in with ***substernal chest pressure***	39.0%	9.1%
Syntactic Variation	After the question is paraphrased into declarative form, its syntactic dependency structure does not match that of the answer sentence	Q: Has this patient ever been **treated** with ffp? E: attempt to reverse anticoagulation , one unit of **FFP was begun**	60.0%	64.1%
Multiple Sentence	Co-reference and higher level fusion of multiple sentences	Q: What happened when the patient was given **ascending aortic root replacement**? E: The patient tolerated **the procedure** fairly well and was transferred to the ICU with his *chest open*	23.8%	13.6%
Arithmetic	Knowing comparison and subtraction operators.	Q: Show me any **LDL** $>$ **100 mg/dl** in the last 6 years? E: gluc 192, ***LDL 115***, TG 71, HDL 36	13.3%	N.A.
Temporal	Reasoning based on time frame	Q: What were the results of the abnormal A1C on **2115-12-14**? E: HBA1C **12/14/2115** *11.80*	18.1%	N.A.
Incomplete Context	Unstructured clinical text is noisy and may have missing context	Q: What is her current dose of iron? E: Iron *325 mg* p.o. t.i.d.	28.6%	N.A.
Class Prediction	Questions for which a specific predefined class needs to be predicted	Q: Is the patient currently Obese? E: Yes	12.4%	N.A.

Table 4: We manually labeled 105 examples into one or more of the above categories. Words relevant to the corresponding reasoning type are in bold and the answer entity (if any) in the evidence is in *italics*. We compare this analysis with SQuAD.

Dataset	Train/Test	HM-1	HM-2	Neural
GeoQuery	600/280	32.8%	52.1%	74.6%[†]
ATIS	4,473/448	20.8%	52.2%	69.9%[†]
emrQL-1	1M/253K	0.3%	26.3%	22.4%
emrQL-2	1.1M/296K	31.6%	32.0%	42.7%

Table 5: Heuristic (HM) and neural (seq2seq) models performance on question to logical form learning in emrQA.

(Boag et al., 2015) for emrQA and Jia and Liang (2016)'s work for ATIS and GeoQuery. Scoring and matching is done using two heuristics: (1) HM-1, which computes an identical match, and (2) HM-2, which generates a GloVe vector (Arora et al., 2016) representation of the templates using sentence2vec and then computes pairwise cosine similarity.

Neural Model: We train a sequence-to-sequence (seq2seq) (Sutskever et al., 2014) with attention paradigm (Bahdanau et al., 2014; Luong et al., 2017) as our neural baseline (2 layers, each with 64 hidden units). The same setting when used with Geoquery and ATIS gives poor results because the parameters are not appropriate for the nature of that dataset. Hence, for comparison with GeoQuery and ATIS, we use the results of seq2seq model with a single 200 hidden units layer (Jia and Liang, 2016). At test time we automatically balance missing right parentheses.

6.1.1 Experimental Setup

We randomly partition the QL pairs in the dataset in train(80%) and test(20%) sets in two ways. (1) In emrQL-1, we first split the paraphrase templates corresponding to a single LF template into train and test, and then generate the instances of QL pairs. (2) In emrQL-2, we first generate the instances of QL pairs from the templates and then distribute them into train and test sets. As a result, emrQL-1 has more lexical variation between train and test distribution compared to emrQL-2, resulting in increased paraphrase complexity. We use accuracy i.e, the total number of logical forms predicted correctly as a metric to evaluate our model.

6.1.2 Results

The performance of the proposed models is summarized in Table 5. emrQL results are not directly comparable with GeoQuery and ATIS because of the differences in the lexicon and tools available for the domains. However, it helps us establish that QL learning in emrQA is non-trivial and supports significant future work.

Error analysis of heuristic models on emrQL-1 and emrQL-2 showed that 70% of the errors occurred because of incorrect question normalization. In fact, 30% of these questions had not been normalized at all. This shows that the entities

---

[†] results from Jia and Liang (2016)

added to the templates are complex and diverse and make the inverse process of template generation non trivial. This makes a challenging QL corpus that cannot trivially be solved by template matching based approaches.

Errors made by the neural model on both emrQL-1 and emrQL-2 are due to long LFs (20%) and incorrectly identified entities (10%), which are harder for the attention-based model (Jia and Liang, 2016). The increased paraphrase complexity in emrQL-1 compared to emrQL-2 resulted in 20% more structural errors in emrQL-1, where the predicted event/grammar structure deviates significantly from the ground truth. This shows that the model is not adequately capturing the semantics in the questions to generalize to new paraphrases. Therefore, emrQL-1 can be used to benchmark QL models robust to paraphrasing.

6.2 Q-A Mapping

Question-answering on emrQA consists of two different tasks, (1) extraction of answer line from the clinical note (machine comprehension (MC)) and (2) prediction of answer class based on the entire clinical note. We provide baseline models to illustrate the complexity in doing both these tasks.

Machine Comprehension: To do extractive QA on EMRs, we use DrQA's (Chen et al., 2017) document reader which is a multi-layer RNN based MC model. We use their best performing settings trained for SQuAD data using Glove vectors (300 dim-840B).

Class Prediction: We build a multi-class logistic regression model for predicting a class as an answer based on the patient's clinical note. Features input to the classifier are TF-IDF vectors of the question and the clinical notes taken from *i2b2 smoking and obesity datasets*.

6.2.1 Experimental setup

We consider a 80-20 split of the data for train-test. In order to evaluate worst-case performance, we train on question-evidence pairs in a clinical note obtained by using only one random paraphrase for a question instead of all the paraphrases. We use a slightly modified[‡] version of the two popularly reported metrics in MC for evaluation since our evidence span is longer: Exact Match (EM) and F1. Wherever the answer entity in an evidence is explicitly known, EM checks if the answer entity is present within the evidence, otherwise it checks if the predicted evidence span lies within ±20 characters of the ground truth evidence. For F1 we construct a bag of tokens for each evidence string and measure the F1 score of the overlap between the two bags of tokens. Since there may be multiple evidence for a given question, we consider only the top 10 predictions and report an average of EM and F1 over ground truth number of answers. In the class prediction setting, we report the subset accuracy.

Model	Train/Test	Exact Match	F1
DrQA (MC)	47,605/9,966	59.2%	60.6
Class Prediction	1276/320	36.6%	n.a

Table 7: Performance of baseline models on the two QA sub tasks, machine comprehension (MC) and class prediction.

6.2.2 Results

The performance of the proposed models is summarized in Table 7. DrQA is one of the best performing models on SQuAD with an F1 of 78.8 and EM of 69.5. The relatively low performance of the models on emrQA (60.6 F1 and 59.2 EM) shows that QA on EMRs is a complex task and offers new challenges to existing QA models.

To understand model performance, we macro-average the EM across all the questions corresponding to a LF template. We observe that LFs representing temporal and arithmetic[§] needs had $<$ 16% EM. LFs expressing the need for medical KB[§] performed poorly since we used general Glove embeddings. An analysis of LFs which had approximately equal number of QA pair representation in the test set revealed an interesting relation between the model performance and LF complexity, as summarized in Table 6. The trend shows that performance is worse on multiple relation questions as compared to single relation and attribute questions, showing that the LFs sufficiently capture the complexity of the questions and give us an ability to do a qualitative model analysis.

Error analysis on a random sample of 50 questions containing at least one answer entity in an evidence showed that: (1) 38% of the examples required multiple sentence reasoning of which 16% were due to a missing evidence in a multiple evidence question, (2) 14% were due to syntactic variation, (3) 10% required medical reasoning and (4) in 14%, DrQA predicted an incomplete evidence span missing the answer entity in it.

[‡] using the original definitions, the evaluated values were far less than those obtained in Table 7

[§] maximum representation of these templates comes from the i2b2 heart disease risk dataset

Logical Form template	Property	Exact Match
MedicationEvent ($\|medication\|$) **[enddate=x]**	single attribute	55.3%
{LabEvent ($\|test\|$) OR ProcedureEvent ($\|test\|$)} **conducted** {ConditionEvent(x) OR SymptomEvent (x)}	single relation	32.2%
{MedicationEvent($\|treatment\|$)ORProcedureEvent($\|treatment\|$)} **improves/worsens/causes** {ConditionEvent (x) OR SymptomEvent (x)}	multiple relation	12.6%

Table 6: Neural models (DrQA) performance on question-evidence corpus of emrQA stratified according to the logical form templates. Instance showing increasing complexity in the logical forms with decreasing model performance.

7 Discussion

In this section, we describe how our generation framework may also be applied to generate open-domain QA datasets given the availability of other NLP resources. We also discuss possible extensions of the framework to increase the complexity of the generated datasets.

Open domain QA dataset generation: Consider the popularly used SQuAD (Rajpurkar et al., 2016) reading comprehension dataset generated by crowdworkers, where the answer to every question is a segment of text from the corresponding passage in the Wikipedia article. This dataset can easily be generated or extended using our proposed framework with existing NLP annotations on Wikipedia (Auer et al., 2007; Nothman et al., 2008; Ghaddar and Langlais, 2017).

For instance, consider DBPedia (Auer et al., 2007), an existing dataset of entities and their relations extracted from Wikipedia. It also has its own ontology which can serve as the semantic frames schema to define logical forms. Using these resources, our reverse engineering technique for QA dataset generation can be applied as follows. (1) Question templates can be defined for each entity type and relation in DBPedia. For example[¶], consider the relation *[place, country]* field in DBpedia. For this we can define a question template *In what country is* $|place|$ *located?*. (2) Every such question template can be annotated with a logical form template using existing DBPedia ontology. (3) By considering the entity values of DBPedia fields such as *[place=Normandy, dbo:country=France]*, we can automatically generate the question *In what country is Normandy located?* and its corresponding logical form from the templates. The text span of *country=France* from the Wikipedia passage is then used as the answer (Daiber et al., 2013). Currently, this QA pair instance is a part of the SQuAD dev set. Using our framework we can generate many more instances like this example from different Wikipedia passages - without crowdsourcing efforts.

Extensions to the framework: The complexity of the generated dataset can be further extended as follows. (1) We can use a coreferred or a lexical variant of the original entity in the question-logical form generation. This can allow for increased lexical variation between the question and answer line entities in the passage. (2) It is possible to combine two or more question templates to make compositional questions with the answers to these questions similarly combined. This can also result in more multiple sentence reasoning questions. (3) We can generate questions with entities not related to the context in the passage. This can increase empty answer questions in the dataset, resulting in increased negative training examples.

8 Conclusions and Future Work

We propose a novel framework that can generate a large-scale QA dataset using existing resources and minimal expert input. This has the potential to make a huge impact in domains like medicine, where obtaining manual QA annotations is tedious and infeasible. We apply this framework to generate a large scale EMR QA corpus (emrQA), consisting of 400,000 question-answers pairs and 1 million question-logical forms, and analyze the complexity of the dataset to show its non-trivial nature. We show that the logical forms provide a symbolic representation that is very useful for corpus generation and for model analysis. The logical forms also provide an opportunity to build interpretable systems by perhaps jointly (or latently) learning the logical form and answer for a question. In future, this framework may be applied to also re-purpose and integrate other NLP datasets such as MIMIC and generate a more diverse and representative EMR QA corpus (Johnson et al., 2016).

Acknowledgments

This project is partially funded by Sloan Research Fellowship, PhRMA Foundation Award in Informatics, and NSF Career Award (1652815). The authors would like to thank Siddharth Patwardhan for his valuable feedback in formatting the paper.

[¶] example reference: http://dbpedia.org/page/Normandy

References

Alan R Aronson. 2001. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In Proceedings of the AMIA Symposium, page 17. American Medical Informatics Association.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In The semantic web, pages 722–735. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In EMNLP.

William Boag, Kevin Wacome, Tristan Naumann, and Anna Rumshisky. 2015. Cliner: A lightweight tool for clinical named entity recognition. AMIA Joint Summits on Clinical Research Informatics (poster).

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. Nucleic acids research, 32(suppl_1):D267–D270.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In ACL (1), pages 423–433.

Piotr Chabierski, Alessandra Russo, and Mark Law. 2017. Logic-based approach to machine comprehension of text.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pages 190–200. Association for Computational Linguistics.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In Proceedings of the 9th International Conference on Semantic Systems (I-Semantics).

Randall Davis, Bruce Buchanan, and Edward Shortliffe. 1977. Production rules as a representation for a knowledge-based consultation program. Artificial intelligence, 8(1):15–45.

Dina Demner-Fushman, Wendy Webber Chapman, and Clement J. McDonald. 2009. What can natural language processing do for clinical decision support? Journal of Biomedical Informatics, 42(5):760–772.

Mary Devereaux. 2013. The use of patient records (ehr) for research.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. arXiv preprint arXiv:1707.03904.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on, pages 813–820. IEEE.

Yansong Feng, Songfang Huang, Dongyan Zhao, et al. 2016. Hybrid question answering over knowledge base and free text. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2397–2407.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. AI magazine, 31(3):59–79.

Jianfeng Gao, Rangan Majumder, and Bill Dolan. Machine reading for question answering: from symbolic to neural computation.

Abbas Ghaddar and Phillippe Langlais. 2017. Winer: A wikipedia annotated corpus for named entity recognition. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), volume 1, pages 413–422.

Y. Guo, R. Gaizauskas, I. Roberts, G. Demetriou, and M. Hepple. 2006. Identifying personal health information using support vector machines. In i2b2 Workshop on Challenges in Natural Language Processing for Clinical Data, pages 10–11.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In Advances in Neural Information Processing Systems, pages 1693–1701.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. arXiv preprint arXiv:1606.03622.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. Scientific data, 3:160035.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. arXiv preprint arXiv:1705.03551.

Harlan M Krumholz, Sharon F Terry, and Joanne Waldstreicher. 2016. Data acquisition, curation, and use for a continuously learning health system. Jama, 316(16):1669–1670.

Chonho Lee, Zhaojing Luo, Kee Yuan Ngiam, Meihui Zhang, Kaiping Zheng, Gang Chen, Beng Chin Ooi, and Wei Luen James Yip. 2017. Big healthcare data analytics: Challenges and applications. In Handbook of Large-Scale Distributed Computing in Smart Healthcare, pages 11–41. Springer.

Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. 2013. Evaluating question answering over linked data. Web Semantics: Science, Services and Agents on the World Wide Web, 21:3–13.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. https://github.com/tensorflow/nmt.

George A Miller. 1995. Wordnet: a lexical database for english. Communications of the ACM, 38(11):39–41.

Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. 2013. Using of jaccard coefficient for keywords similarity. In Proceedings of the International MultiConference of Engineers and Computer Scientists, volume 1.

Joel Nothman, James R Curran, and Tara Murphy. 2008. Transforming wikipedia into named entity training data. In Proceedings of the Australasian Language Technology Association Workshop 2008, pages 124–132.

Hamid Palangi, Paul Smolensky, Xiaodong He, and Li Deng. 2018. Question-answering with grammatically-interpretable representations. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting on association for computational linguistics, pages 311–318. Association for Computational Linguistics.

Preethi Raghavan and Siddharth Patwardhan. 2016. Question answering on electronic medical records. In Proceedings of the 2016 Summit on Clinical Research Informatics, San Francisco, CA, March 2016.

Preethi Raghavan, Siddharth Patwardhan, Jennifer J. Liang, and Murthy V. Devarakonda. 2017. Annotating electronic medical records for question answering. arXiv:1805.06816.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In EMNLP, volume 3, page 4.

Kirk Roberts and Dina Demner-Fushman. 2016. Annotating logical forms for ehr questions. In LREC... International Conference on Language Resources & Evaluation:[proceedings]. International Conference on Language Resources and Evaluation, volume 2016, page 3772. NIH Public Access.

Benjamin Shickel, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. 2017. Deep ehr: A survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. IEEE Journal of Biomedical and Health Informatics.

Amber Stubbs and Özlem Uzuner. 2015. Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/uthealth corpus. Journal of biomedical informatics, 58:S20–S29.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In EMNLP, pages 562–572.

Simon Šuster, Stéphan Tulkens, and Walter Daelemans. 2017. A short review of ethical challenges in clinical natural language processing. arXiv preprint arXiv:1703.10090.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pages 3104–3112.

Paul C Tang, Danielle Fafchamps, and Edward H Shortliffe. 1994. Traditional medical records as a source of clinical data in the outpatient setting. In Proceedings of the Annual Symposium on Computer Application in Medical Care, page 575. American Medical Informatics Association.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. arXiv preprint arXiv:1611.09830.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia

Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. BMC bioinformatics, 16(1):138.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, and Christina Unger. 2015. Hawk–hybrid question answering using linked data. In European Semantic Web Conference, pages 353–368. Springer.

Özlem Uzuner. 2009. Recognizing obesity and comorbidities in sparse data. Journal of the American Medical Informatics Association, 16(4):561–570.

Ozlem Uzuner, Andreea Bodnari, Shuying Shen, Tyler Forbush, John Pestian, and Brett R South. 2012. Evaluating the state of the art in coreference resolution for electronic medical records. Journal of the American Medical Informatics Association, 19(5):786–791.

Özlem Uzuner, Ira Goldstein, Yuan Luo, and Isaac Kohane. 2008. Identifying patient smoking status from medical discharge records. Journal of the American Medical Informatics Association, 15(1):14–24.

Özlem Uzuner, Imre Solti, and Eithon Cadag. 2010a. Extracting medication information from clinical text. Journal of the American Medical Informatics Association, 17(5):514–518.

Özlem Uzuner, Imre Solti, Fei Xia, and Eithon Cadag. 2010b. Community annotation experiment for ground truth generation for the i2b2 medication challenge. Journal of the American Medical Informatics Association, 17(5):519–523.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. Journal of the American Medical Informatics Association, 18(5):552–556.

Alfredo Vellido, José David Martín-Guerrero, and Paulo JG Lisboa. 2012. Making machine learning models interpretable. In ESANN, volume 12, pages 163–172. Citeseer.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In Trec, volume 99, pages 77–82.

HOTPOTQA: A Dataset for Diverse, Explainable Multi-hop Question Answering

Zhilin Yang[*♠] Peng Qi[*♡] Saizheng Zhang[*♣]
Yoshua Bengio[♣◇] William W. Cohen[†]
Ruslan Salakhutdinov[♠] Christopher D. Manning[♡]
♠ Carnegie Mellon University ♡ Stanford University ♣ Mila, Université de Montréal
◇ CIFAR Senior Fellow † Google AI
`{zhiliny, rsalakhu}@cs.cmu.edu, {pengqi, manning}@cs.stanford.edu`
`saizheng.zhang@umontreal.ca, yoshua.bengio@gmail.com, wcohen@google.com`

Abstract

Existing question answering (QA) datasets fail to train QA systems to perform complex reasoning and provide explanations for answers. We introduce HOTPOTQA, a new dataset with 113k Wikipedia-based question-answer pairs with four key features: (1) the questions require finding and reasoning over multiple supporting documents to answer; (2) the questions are diverse and not constrained to any pre-existing knowledge bases or knowledge schemas; (3) we provide sentence-level supporting facts required for reasoning, allowing QA systems to reason with strong supervision and explain the predictions; (4) we offer a new type of factoid comparison questions to test QA systems' ability to extract relevant facts and perform necessary comparison. We show that HOTPOTQA is challenging for the latest QA systems, and the supporting facts enable models to improve performance and make explainable predictions.

1 Introduction

The ability to perform reasoning and inference over natural language is an important aspect of intelligence. The task of question answering (QA) provides a quantifiable and objective way to test the reasoning ability of intelligent systems. To this end, a few large-scale QA datasets have been proposed, which sparked significant progress in this direction. However, existing datasets have limitations that hinder further advancements of machine reasoning over natural language, especially in testing QA systems' ability to perform *multi-hop reasoning*, where the system has to reason with information taken from more than one document to arrive at the answer.

Paragraph A, Return to Olympus:
[1] *Return to Olympus is the only album by the alternative rock band Malfunkshun.* [2] *It was released after the band had broken up and after lead singer Andrew Wood (later of Mother Love Bone) had died of a drug overdose in 1990.* [3] Stone Gossard, of Pearl Jam, had compiled the songs and released the album on his label, Loosegroove Records.

Paragraph B, Mother Love Bone:
[4] *Mother Love Bone was an American rock band that formed in Seattle, Washington in 1987.* [5] The band was active from 1987 to 1990. [6] *Frontman Andrew Wood's personality and compositions helped to catapult the group to the top of the burgeoning late 1980s/early 1990s Seattle music scene.* [7] *Wood died only days before the scheduled release of the band's debut album, "Apple", thus ending the group's hopes of success.* [8] The album was finally released a few months later.

Q: What was the former band of the member of Mother Love Bone who died just before the release of "Apple"?
A: Malfunkshun
Supporting facts: 1, 2, 4, 6, 7

Figure 1: An example of the multi-hop questions in HOTPOTQA. We also highlight the supporting facts in *blue italics*, which are also part of the dataset.

First, some datasets mainly focus on testing the ability of reasoning within a single paragraph or document, or *single-hop reasoning*. For example, in SQuAD (Rajpurkar et al., 2016) questions are designed to be answered given a single paragraph as the context, and most of the questions can in fact be answered by matching the question with a single sentence in that paragraph. As a result, it has fallen short at testing systems' ability to reason over a larger context. TriviaQA (Joshi et al., 2017) and SearchQA (Dunn et al., 2017) create a more challenging setting by using information retrieval to collect multiple documents to form the context given existing question-answer pairs. Nevertheless, most of the questions can be answered by matching the question with a few nearby sentences in one single paragraph, which is limited as it does not require more complex reasoning (e.g.,

---

[*] These authors contributed equally. The order of authorship is decided through dice rolling.

[†] Work done when WWC was at CMU.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2369–2380
Brussels, Belgium, October 31 - November 4, 2018.

over multiple paragraphs).

Second, existing datasets that target multi-hop reasoning, such as QAngaroo (Welbl et al., 2018) and COMPLEXWEBQUESTIONS (Talmor and Berant, 2018), are constructed using existing knowledge bases (KBs). As a result, these datasets are constrained by the schema of the KBs they use, and therefore the diversity of questions and answers is inherently limited.

Third, all of the above datasets only provide distant supervision; i.e., the systems only know what the answer is, but do not know what supporting facts lead to it. This makes it difficult for models to learn about the underlying reasoning process, as well as to make explainable predictions.

To address the above challenges, we aim at creating a QA dataset that requires reasoning over multiple documents, and does so in natural language, without constraining itself to an existing knowledge base or knowledge schema. We also want it to provide the system with strong supervision about what text the answer is actually derived from, to help guide systems to perform meaningful and explainable reasoning.

We present HOTPOTQA[1], a large-scale dataset that satisfies these desiderata. HOTPOTQA is collected by crowdsourcing based on Wikipedia articles, where crowd workers are shown multiple supporting context documents and asked explicitly to come up with questions requiring reasoning about all of the documents. This ensures it covers multi-hop questions that are more natural, and are not designed with any pre-existing knowledge base schema in mind. Moreover, we also ask the crowd workers to provide the supporting facts they use to answer the question, which we also provide as part of the dataset (see Figure 1 for an example). We have carefully designed a data collection pipeline for HOTPOTQA, since the collection of high-quality multi-hop questions is non-trivial. We hope that this pipeline also sheds light on future work in this direction. Finally, we also collected a novel type of questions—comparison questions—as part of HOTPOTQA, in which we require systems to compare two entities on some shared properties to test their understanding of both language and common concepts such as numerical magnitude. We make HOTPOTQA publicly available at **https://HotpotQA.github.io**.

2 Data Collection

The main goal of our work is to collect a diverse and explainable question answering dataset that requires multi-hop reasoning. One way to do so is to define reasoning chains based on a knowledge base (Welbl et al., 2018; Talmor and Berant, 2018). However, the resulting datasets are limited by the incompleteness of entity relations and the lack of diversity in the question types. Instead, in this work, we focus on text-based question answering in order to diversify the questions and answers. The overall setting is that given some context paragraphs (e.g., a few paragraphs, or the entire Web) and a question, a QA system answers the question by extracting a span of text from the context, similar to Rajpurkar et al. (2016). We additionally ensure that it is necessary to perform multi-hop reasoning to correctly answer the question.

It is non-trivial to collect text-based multi-hop questions. In our pilot studies, we found that simply giving an arbitrary set of paragraphs to crowd workers is counterproductive, because for most paragraph sets, it is difficult to ask a meaningful multi-hop question. To address this challenge, we carefully design a pipeline to collect text-based multi-hop questions. Below, we will highlight the key design choices in our pipeline.

Building a Wikipedia Hyperlink Graph. We use the entire English Wikipedia dump as our corpus.[2] In this corpus, we make two observations: (1) hyper-links in the Wikipedia articles often naturally entail a relation between two (already disambiguated) entities in the context, which could potentially be used to facilitate multi-hop reasoning; (2) the first paragraph of each article often contains much information that could be queried in a meaningful way. Based on these observations, we extract all the hyperlinks from the first paragraphs of all Wikipedia articles. With these hyperlinks, we build a directed graph G, where each edge (a, b) indicates there is a hyperlink from the first paragraph of article a to article b.

Generating Candidate Paragraph Pairs. To generate meaningful pairs of paragraphs for multi-hop question answering with G, we start by considering an example question "when was the singer and songwriter of Radiohead born?" To

---

[1] The name comes from the first three authors' arriving at the main idea during a discussion at a hot pot restaurant.

[2] `https://dumps.wikimedia.org/`

answer this question, one would need to first reason that the "singer and songwriter of Radiohead" is "Thom Yorke", and then figure out his birthday in the text. We call "Thom Yorke" a *bridge entity* in this example. Given an edge (a, b) in the hyperlink graph G, the entity of b can usually be viewed as a bridge entity that connects a and b. As we observe articles b usually determine the theme of the shared context between a and b, but not all articles b are suitable for collecting multi-hop questions. For example, entities like countries are frequently referred to in Wikipedia, but don't necessarily have much in common with all incoming links. It is also difficult, for instance, for the crowd workers to ask meaningful multi-hop questions about highly technical entities like the IPv4 protocol. To alleviate this issue, we constrain the bridge entities to a set of manually curated pages in Wikipedia (see Appendix A). After curating a set of pages B, we create candidate paragraph pairs by sampling edges (a, b) from the hyperlink graph such that $b \in B$.

Comparison Questions. In addition to questions collected using bridge entities, we also collect another type of multi-hop questions—comparison questions. The main idea is that comparing two entities from the same category usually results in interesting multi-hop questions, e.g., "Who has played for more NBA teams, Michael Jordan or Kobe Bryant?" To facilitate collecting this type of question, we manually curate 42 lists of similar entities (denoted as L) from Wikipedia.[3] To generate candidate paragraph pairs, we randomly sample two paragraphs from the same list and present them to the crowd worker.

To increase the diversity of multi-hop questions, we also introduce a subset of yes/no questions in comparison questions. This complements the original scope of comparison questions by offering new ways to require systems to reason over both paragraphs. For example, consider the entities Iron Maiden (from the UK) and AC/DC (from Australia). Questions like "Is Iron Maiden or AC/DC from the UK?" are not ideal, because one would deduce the answer is "Iron Maiden" even if one only had access to that article. With yes/no questions, one may ask "Are Iron Maiden and AC/DC from the same country?", which requires reasoning over both paragraphs.

Algorithm 1 Overall data collection procedure

```
Input: question type ratio $r_1 = 0.75$, yes/no ratio $r_2 = 0.5$
while not finished do
    if $\mathrm{random}() < r_1$ then
        Uniformly sample an entity $b \in B$
        Uniformly sample an edge $(a, b)$
        Workers ask a question about paragraphs $a$ and $b$
    else
        Sample a list from $L$, with probabilities weighted by list sizes
        Uniformly sample two entities $(a, b)$ from the list
        if $\mathrm{random}() < r_2$ then
            Workers ask a yes/no question to compare $a$ and $b$
        else
            Workers ask a question with a span answer to compare $a$ and $b$
        end if
    end if
    Workers provide the supporting facts
end while
```

To the best of our knowledge, text-based comparison questions are a novel type of questions that have not been considered by previous datasets. More importantly, answering these questions usually requires arithmetic comparison, such as comparing ages given birth dates, which presents a new challenge for future model development.

Collecting Supporting Facts. To enhance the explainability of question answering systems, we want them to output a set of *supporting facts* necessary to arrive at the answer, when the answer is generated. To this end, we also collect the sentences that determine the answers from crowd workers. These supporting facts can serve as strong supervision for what sentences to pay attention to. Moreover, we can now test the explainability of a model by comparing the predicted supporting facts to the ground truth ones.

The overall procedure of data collection is illustrated in Algorithm 1.

3 Processing and Benchmark Settings

We collected 112,779 valid examples in total on Amazon Mechanical Turk[4] using the ParlAI interface (Miller et al., 2017) (see Appendix A).To isolate potential single-hop questions from the desired multi-hop ones, we first split out a subset of data called *train-easy*. Specifically, we randomly sampled questions (~3–10 per Turker) from top-contributing turkers, and categorized all

---

[3] This is achieved by manually curating lists from the Wikipedia "List of lists of lists" (`https://wiki.sh/y8qv`). One example is "Highest Mountains on Earth".

[4] `https://www.mturk.com/`

Name	Desc.	Usage	# Examples
train-easy	single-hop	training	18,089
train-medium	multi-hop	training	56,814
train-hard	hard multi-hop	training	15,661
dev	hard multi-hop	dev	7,405
test-distractor	hard multi-hop	test	7,405
test-fullwiki	hard multi-hop	test	7,405
Total			112,779

Table 1: Data split. The splits *train-easy*, *train-medium*, and *train-hard* are combined for training. The distractor and full wiki settings use different test sets so that the gold paragraphs in the full wiki test set remain unknown to any models.

their questions into the *train-easy* set if an overwhelming percentage in the sample only required reasoning over one of the paragraphs. We sampled these turkers because they contributed more than 70% of our data. This *train-easy* set contains 18,089 mostly single-hop examples.

We implemented a question answering model based on the current state-of-the-art architectures, which we discuss in detail in Section 5.1. Based on this model, we performed a three-fold cross validation on the remaining multi-hop examples. Among these examples, the models were able to correctly answer 60% of the questions with high confidence (determined by thresholding the model loss). These correctly-answered questions (56,814 in total, 60% of the multi-hop examples) are split out and marked as the *train-medium* subset, which will also be used as part of our training set.

After splitting out *train-easy* and *train-medium*, we are left with hard examples. As our ultimate goal is to solve multi-hop question answering, we focus on questions that the latest modeling techniques are not able to answer. Thus we constrain our dev and test sets to be hard examples. Specifically, we randomly divide the hard examples into four subsets, *train-hard*, *dev*, *test-distractor*, and *test-fullwiki*. Statistics about the data split can be found in Table 1. In Section 5, we will show that combining *train-easy*, *train-medium*, and *train-hard* to train models yields the best performance, so we use the combined set as our default training set. The two test sets *test-distractor* and *test-fullwiki* are used in two different benchmark settings, which we introduce next.

We create two benchmark settings. In the first setting, to challenge the model to find the true supporting facts in the presence of noise, for each example we employ bigram tf-idf (Chen et al., 2017) to retrieve 8 paragraphs from Wikipedia as *distractors*, using the question as the query. We mix them with the 2 gold paragraphs (the ones used to collect the question and answer) to construct the **distractor** setting. The 2 gold paragraphs and the 8 distractors are shuffled before they are fed to the model. In the second setting, we fully test the model's ability to locate relevant facts as well as reasoning about them by requiring it to answer the question given the first paragraphs of all Wikipedia articles without the gold paragraphs specified. This **full wiki** setting truly tests the performance of the systems' ability at multi-hop reasoning in the wild.[5] The two settings present different levels of difficulty, and would require techniques ranging from reading comprehension to information retrieval. As shown in Table 1, we use separate test sets for the two settings to avoid leaking information, because the gold paragraphs are available to a model in the distractor setting, but should not be accessible in the full wiki setting.

We also try to understand the model's good performance on the *train-medium* split. Manual analysis shows that the ratio of multi-hop questions in *train-medium* is similar to that of the hard examples (93.3% in *train-medium* vs. 92.0% in *dev*), but one of the question types appears more frequently in *train-medium* compared to the hard splits (Type II: 32.0% in *train-medium* vs. 15.0% in *dev*, see Section 4 for the definition of Type II questions). These observations demonstrate that given enough training data, existing neural architectures can be trained to answer certain types and certain subsets of the multi-hop questions. However, *train-medium* remains challenging when not just the gold paragraphs are present—we show in Appendix C that the retrieval problem on these examples are as difficult as that on their hard cousins.

4 Dataset Analysis

In this section, we analyze the types of questions, types of answers, and types of multi-hop reasoning covered in the dataset.

Question Types. We heuristically identified question types for each collected question. To identify the question type, we first locate the *central question word* (CQW) in the question. Since HOTPOTQA contains comparison questions and

---

[5] As we required the crowd workers to use complete entity names in the question, the majority of the questions are unambiguous in the full wiki setting.

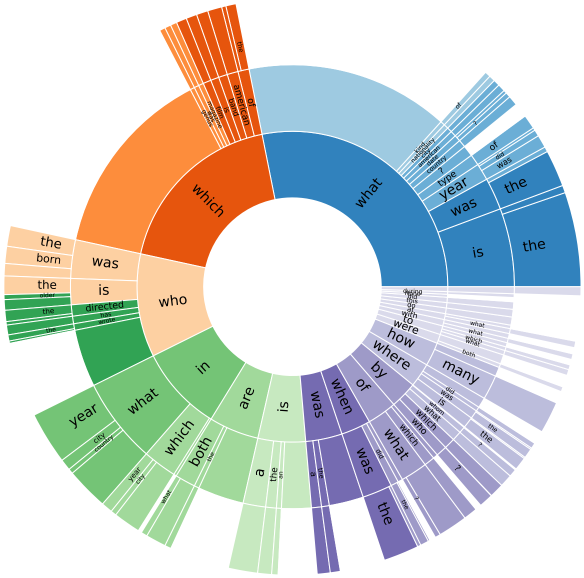

Figure 2: Types of questions covered in HOTPOTQA. Question types are extracted heuristically, starting at question words or prepositions preceding them. Empty colored blocks indicate suffixes that are too rare to show individually. See main text for more details.

yes/no questions, we consider as *question words* WH-words, copulas ("is", "are"), and auxiliary verbs ("does", "did"). Because questions often involve relative clauses beginning with WH-words, we define the CQW as the first question word in the question if it can be found in the first three tokens, or the last question word otherwise. Then, we determine question type by extracting words up to 2 tokens away to the right of the CQW, along with the token to the left if it is one of a few common prepositions (e.g., in the cases of "in which" and "by whom").

We visualize the distribution of question types in Figure 2, and label the ones shared among more than 250 questions. As is shown, our dataset covers a diverse variety of questions centered around entities, locations, events, dates, and numbers, as well as yes/no questions directed at comparing two entities ("Are both A and B ...?"), to name a few.

Answer Types. We further sample 100 examples from the dataset, and present the types of answers in Table 2. As can be seen, HOTPOTQA covers a broad range of answer types, which matches our initial analysis of question types. We find that a majority of the questions are about entities in the articles (68%), and a non-negligible amount of questions also ask about various properties like date (9%) and other descriptive properties such as numbers (8%) and adjectives (4%).

Answer Type	%	Example(s)
Person	30	King Edward II, Rihanna
Group / Org	13	Cartoonito, Apalachee
Location	10	Fort Richardson, California
Date	9	10th or even 13th century
Number	8	79.92 million, 17
Artwork	8	Die schweigsame Frau
Yes/No	6	-
Adjective	4	conservative
Event	1	Prix Benois de la Danse
Other proper noun	6	Cold War, Laban Movement Analysis
Common noun	5	comedy, both men and women

Table 2: Types of answers in HOTPOTQA.

Multi-hop Reasoning Types. We also sampled 100 examples from the dev and test sets and manually classified the types of reasoning required to answer each question. Besides comparing two entities, there are three main types of multi-hop reasoning required to answer these questions, which we show in Table 3 accompanied with examples.

Most of the questions require at least one supporting fact from each paragraph to answer. A majority of sampled questions (42%) require chain reasoning (Type I in the table), where the reader must first identify a bridge entity before the second hop can be answered by filling in the bridge. One strategy to answer these questions would be to decompose them into consecutive single-hop questions. The bridge entity could also be used implicitly to help infer properties of other entities related to it. In some questions (Type III), the entity in question shares certain properties with a bridge entity (e.g., they are collocated), and we can infer its properties through the bridge entity. Another type of question involves locating the answer entity by satisfying multiple properties simultaneously (Type II). Here, to answer the question, one could find the set of all entities that satisfy each of the properties mentioned, and take an intersection to arrive at the final answer. Questions comparing two entities (Comparison) also require the system to understand the properties in question about the two entities (e.g., nationality), and sometimes require arithmetic such as counting (as seen in the table) or comparing numerical values ("Who is older, A or B?"). Finally, we find that sometimes the questions require more than two supporting facts to answer (Other). In our analysis, we also find that for all of the examples shown in the table, the supporting facts provided by the Turkers match exactly with the limited context shown here,

Reasoning Type	%	Example(s)
Inferring the ***bridge entity*** to complete the 2nd-hop question (Type I)	42	**Paragraph A:** The 2015 Diamond Head Classic was a college basketball tournament ... ***Buddy Hield*** *was named the tournament's MVP*. **Paragraph B:** ***Chavano Rainier ”Buddy” Hield*** is a Bahamian professional basketball player for the **Sacramento Kings** of the NBA... **Q:** Which team does the player named 2015 Diamond Head Classic's MVP play for?
Comparing two entities (Comparison)	27	**Paragraph A:** LostAlone were a British rock band ... consisted of *Steven Battelle, Alan Williamson, and Mark Gibson*... **Paragraph B:** Guster is an American alternative rock band ... Founding members *Adam Gardner, Ryan Miller, and Brian Rosenworcel* began... **Q:** Did LostAlone and Guster have the same number of members? (**yes**)
Locating the **answer entity** by checking multiple properties (Type II)	15	**Paragraph A:** Several *current and former members of the Pittsburgh Pirates* – ... John Milner, **Dave Parker**, and Rod Scurry... **Paragraph B:** **David Gene Parker**, *nicknamed ”The Cobra”*, is an American former player in Major League Baseball... **Q:** Which former member of the Pittsburgh Pirates was nicknamed ”The Cobra”?
Inferring about the property of an entity in question through a ***bridge entity*** (Type III)	6	**Paragraph A:** *Marine Tactical Air Command Squadron 28* is a United States Marine Corps aviation command and control unit based at ***Marine Corps Air Station Cherry Point***... **Paragraph B:** ***Marine Corps Air Station Cherry Point*** ... is a United States Marine Corps airfield located in **Havelock, North Carolina**, USA ... **Q:** What city is the Marine Air Control Group 28 located in?
Other types of reasoning that require more than two supporting facts (Other)	2	**Paragraph A:** ... the towns of Yodobashi, **Okubo, Totsuka, and Ochiai town** *were merged into Yodobashi ward*. ... ***Yodobashi Camera*** *is a store with its name taken from the town and ward*. **Paragraph B**: ***Yodobashi Camera*** Co., Ltd. is *a major Japanese retail chain specializing in electronics, PCs, cameras and photographic equipment*. **Q:** Aside from Yodobashi, what other towns were merged into the ward which gave the major Japanese retail chain specializing in electronics, PCs, cameras, and photographic equipment it's name?

Table 3: Types of multi-hop reasoning required to answer questions in the HotpotQA dev and test sets. We show in ***orange bold italics*** bridge entities if applicable, *blue italics* supporting facts from the paragraphs that connect directly to the question, and **green bold** the answer in the paragraph or following the question. The remaining 8% are single-hop (6%) or unanswerable questions (2%) by our judgement.

showing that the supporting facts collected are of high quality.

Aside from the reasoning types mentioned above, we also estimate that about 6% of the sampled questions can be answered with one of the two paragraphs, and 2% of them unanswerable. We also randomly sampled 100 examples from *train-medium* and *train-hard* combined, and the proportions of reasoning types are: Type I 38%, Type II 29%, Comparison 20%, Other 7%, Type III 2%, single-hop 2%, and unanswerable 2%.

5 Experiments

5.1 Model Architecture and Training

To test the performance of leading QA systems on our data, we reimplemented the architecture described in Clark and Gardner (2017) as our baseline model. We note that our implementation without weight averaging achieves performance very close to what the authors reported on SQuAD (about 1 point worse in F_1). Our implemented model subsumes the latest technical advances on question answering, including character-level models, self-attention (Wang et al., 2017), and bi-attention (Seo et al., 2017). Combining these three key components is becoming standard practice, and various state-of-the-art or competitive architectures (Liu et al., 2018; Clark and Gardner, 2017; Wang et al., 2017; Seo et al., 2017; Pan et al., 2017; Salant and Berant, 2018; Xiong et al., 2018) on SQuAD can be viewed as similar to our implemented model. To accommodate yes/no questions, we also add a 3-way classifier after the last recurrent layer to produce the probabilities of "yes", "no", and span-based answers. During decoding, we first use the 3-way output to determine whether the answer is "yes", "no", or a text span. If it is a text span, we further search for the most probable span.

Supporting Facts as Strong Supervision. To evaluate the baseline model's performance in predicting explainable supporting facts, as well as how much they improve QA performance, we additionally design a component to incorporate

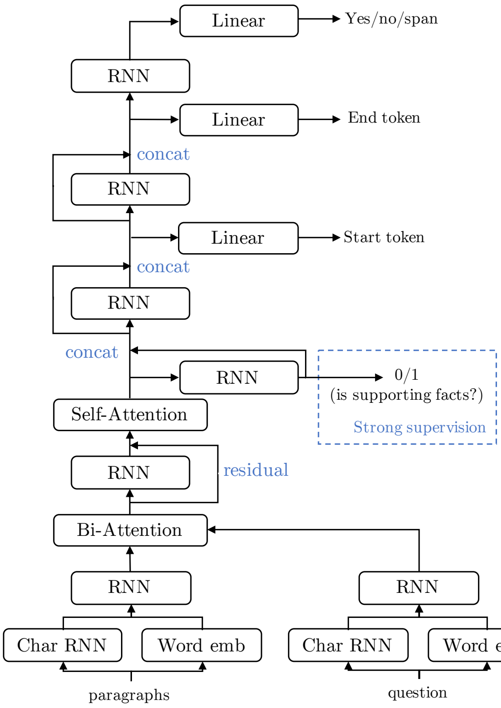

Figure 3: Our model architecture. Strong supervision over supporting facts is used in a multi-task setting.

such strong supervision into our model. For each sentence, we concatenate the output of the self-attention layer at the first and last positions, and use a binary linear classifier to predict the probability that the current sentence is a supporting fact. We minimize a binary cross entropy loss for this classifier. This objective is jointly optimized with the normal question answering objective in a multi-task learning setting, and they share the same low-level representations. With this classifier, the model can also be evaluated on the task of supporting fact prediction to gauge its explainability. Our overall architecture is illustrated in Figure 3. Though it is possible to build a pipeline system, in this work we focus on an end-to-end one, which is easier to tune and faster to train.

5.2 Results

We evaluate our model in the two benchmark settings. In the full wiki setting, to enable efficient tf-idf retrieval among 5,000,000+ wiki paragraphs, given a question we first return a candidate pool of at most 5,000 paragraphs using an inverted-index-based filtering strategy[6] and then select the top 10 paragraphs in the pool as the final candidates using bigram tf-idf.[7] Retrieval performance is shown in Table 5. After retrieving these 10 paragraphs, we then use the model trained in the distractor setting to evaluate its performance on these final candidate paragraphs.

Following previous work (Rajpurkar et al., 2016), we use exact match (EM) and F_1 as two evaluation metrics. To assess the explainability of the models, we further introduce two sets of metrics involving the supporting facts. The first set focuses on evaluating the supporting facts directly, namely EM and F_1 on the set of supporting fact sentences as compared to the gold set. The second set features joint metrics that combine the evaluation of answer spans and supporting facts as follows. For each example, given its precision and recall on the answer span $(P^{(\text{ans})}, R^{(\text{ans})})$ and the supporting facts $(P^{(\text{sup})}, R^{(\text{sup})})$, respectively, we calculate joint F_1 as

$$P^{(\text{joint})} = P^{(\text{ans})} P^{(\text{sup})}, \quad R^{(\text{joint})} = R^{(\text{ans})} R^{(\text{sup})},$$

$$\text{Joint } F_1 = \frac{2P^{(\text{joint})} R^{(\text{joint})}}{P^{(\text{joint})} + R^{(\text{joint})}}.$$

Joint EM is 1 only if both tasks achieve an exact match and otherwise 0. Intuitively, these metrics penalize systems that perform poorly on either task. All metrics are evaluated example-by-example, and then averaged over examples in the evaluation set.

The performance of our model on the benchmark settings is reported in Table 4, where all numbers are obtained with strong supervision over supporting facts. From the distractor setting to the full wiki setting, expanding the scope of the context increases the difficulty of question answering. The performance in the full wiki setting is substantially lower, which poses a challenge to existing techniques on retrieval-based question answering. Overall, model performance in all settings is significantly lower than human performance as shown in Section 5.3, which indicates that more technical advancements are needed in future work.

We also investigate the explainability of our model by measuring supporting fact prediction performance. Our model achieves 60+ supporting fact prediction F_1 and ~40 joint F_1, which indicates there is room for further improvement in terms of explainability.

In Table 6, we break down the performance on different question types. In the distractor setting, comparison questions have lower F_1 scores

---

[6] See Appendix C for details.

[7] We choose the number of final candidates as 10 to stay consistent with the distractor setting where candidates are 2 gold paragraphs plus 8 distractors.

Setting	Split	Answer		Sup Fact		Joint	
		EM	F_1	EM	F_1	EM	F_1
distractor	dev	44.44	58.28	21.95	66.66	11.56	40.86
distractor	test	45.46	58.99	22.24	66.62	12.04	41.37
full wiki	dev	24.68	34.36	5.28	40.98	2.54	17.73
full wiki	test	25.23	34.40	5.07	40.69	2.63	17.85

Table 4: Main results: the performance of question answering and supporting fact prediction in the two benchmark settings. We encourage researchers to report these metrics when evaluating their methods.

Set	MAP	Mean Rank	Hits@2	Hits@10
dev	43.93	314.71	39.43	56.06
test	43.21	314.05	38.67	55.88

Table 5: Retrieval performance in the full wiki setting. Mean Rank is averaged over the ranks of two gold paragraphs.

Setting	Br EM	Br F_1	Cp EM	Cp F_1
distractor	43.41	59.09	48.55	55.05
full wiki	19.76	30.42	43.87	50.70

Table 6: Performance breakdown over different question types on the dev set in the distractor setting. "Br" denotes questions collected using bridge entities, and "Cp" denotes comparison questions.

Setting	EM	F_1
our model	44.44	58.28
– sup fact	42.79	56.19
– sup fact, self attention	41.59	55.19
– sup fact, char model	41.66	55.25
– sup fact, train-easy	41.61	55.12
– sup fact, train-easy, train-medium	31.07	43.61
gold only	48.38	63.58
sup fact only	51.95	66.98

Table 7: Ablation study of question answering performance on the dev set in the distractor setting. "– sup fact" means removing strong supervision over supporting facts from our model. "– train-easy" and "– train-medium" means discarding the according data splits from training. "gold only" and "sup fact only" refer to using the gold paragraphs or the supporting facts as the only context input to the model.

than questions involving bridge entities (as defined in Section 2), which indicates that better modeling this novel question type might need better neural architectures. In the full wiki setting, the performance of bridge entity questions drops significantly while that of comparison questions decreases only marginally. This is because both entities usually appear in the comparison questions, and thus reduces the difficulty of retrieval. Combined with the retrieval performance in Table 5, we believe that the deterioration in the full wiki setting in Table 4 is largely due to the difficulty of retrieving both entities.

We perform an ablation study in the distractor setting, and report the results in Table 7. Both self-attention and character-level models contribute notably to the final performance, which is consistent with prior work. This means that techniques targeted at single-hop QA are still somewhat effective in our setting. Moreover, removing strong supervision over supporting facts decreases performance, which demonstrates the effectiveness of our approach and the usefulness of the supporting facts. We establish an estimate of the upper bound of strong supervision by only considering the supporting facts as the oracle context input to our model, which achieves a 10+ F_1 improvement over not using the supporting facts. Compared with the gain of strong supervision in our model ($\sim$2 points in F_1), our proposed method of incorporating supporting facts supervision is most likely suboptimal, and we leave the challenge of better modeling to future work. At last, we show that combining all data splits (*train-easy*, *train-medium*, and *train-hard*) yields the best performance, which is adopted as the default setting.

5.3 Establishing Human Performance

To establish human performance on our dataset, we randomly sampled 1,000 examples from the dev and test sets, and had at least three additional Turkers provide answers and supporting facts for these examples. As a baseline, we treat the original Turker during data collection as the prediction, and the newly collected answers and supporting facts as references, to evaluate human performance. For each example, we choose the answer and supporting fact reference that maximize the F_1 score to report the final metrics to reduce the effect of ambiguity (Rajpurkar et al., 2016).

Setting	Answer		Sp Fact		Joint	
	EM	F_1	EM	F_1	EM	F_1
gold only	65.87	74.67	59.76	90.41	41.54	68.15
distractor	60.88	68.99	30.99	74.67	20.06	52.37
Human	83.60	91.40	61.50	90.04	52.30	82.55
Human UB	96.80	98.77	87.40	97.56	84.60	96.37

Table 8: Comparing baseline model performance with human performance on 1,000 random samples. "Human UB" stands for the upper bound on annotator performance on HOTPOTQA. For details please refer to the main body.

As can be seen in Table 8, the original crowd worker achieves very high performance in both finding supporting facts, and answering the question correctly. If the baseline model were provided with the correct supporting paragraphs to begin with, it achieves parity with the crowd worker in finding supporting facts, but still falls short at finding the actual answer. When distractor paragraphs are present, the performance gap between the baseline model and the crowd worker on both tasks is enlarged to ∼30% for both EM and F_1.

We further establish the upper bound of human performance in HOTPOTQA, by taking the maximum EM and F_1 for each example. Here, we use each Turker's answer in turn as the prediction, and evaluate it against all other workers' answers. As can be seen in Table 8, most of the metrics are close to 100%, illustrating that on most examples, at least a subset of Turkers agree with each other, showing high inter-annotator agreement. We also note that crowd workers agree less on supporting facts, which could reflect that this task is inherently more subjective than answering the question.

6 Related Work

Various recently-proposed large-scale QA datasets can be categorized in four categories.

Single-document datasets. SQuAD (Rajpurkar et al., 2016, 2018) questions that are relatively simple because they usually require no more than one sentence in the paragraph to answer.

Multi-document datasets. TriviaQA (Joshi et al., 2017) and SearchQA (Dunn et al., 2017) contain question answer pairs that are accompanied with more than one document as the context. This further challenges QA systems' ability to accommodate longer contexts. However, since the supporting documents are collected after the question answer pairs with information retrieval, the questions are not guaranteed to involve interesting reasoning between multiple documents.

KB-based multi-hop datasets. Recent datasets like QAngaroo (Welbl et al., 2018) and COMPLEXWEBQUESTIONS (Talmor and Berant, 2018) explore different approaches of using pre-existing knowledge bases (KB) with pre-defined logic rules to generate valid QA pairs, to test QA models' capability of performing multi-hop reasoning. The diversity of questions and answers is largely limited by the fixed KB schemas or logical forms. Furthermore, some of the questions might be answerable by one text sentence due to the incompleteness of KBs.

Free-form answer-generation datasets. MS MARCO (Nguyen et al., 2016) contains 100k user queries from Bing Search with human generated answers. Systems generate free-form answers and are evaluated by automatic metrics such as ROUGE-L and BLEU-1. However, the reliability of these metrics is questionable because they have been shown to correlate poorly with human judgement (Novikova et al., 2017).

7 Conclusions

We present HOTPOTQA, a large-scale question answering dataset aimed at facilitating the development of QA systems capable of performing explainable, multi-hop reasoning over diverse natural language. We also offer a new type of factoid comparison questions to test systems' ability to extract and compare various entity properties in text.

Acknowledgements

This work is partly funded by the Facebook ParlAI Research Award. ZY, WWC, and RS are supported by a Google grant, the DARPA grant D17AP00001, the ONR grants N000141512791, N000141812861, and the Nvidia NVAIL Award. SZ and YB are supported by Mila, Université de Montréal. PQ and CDM are supported by the National Science Foundation under Grant No. IIS-1514268. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association of Computational Linguistics*.

Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. SearchQA: A new Q&A dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Alexander H Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. 2017. ParlAI: A dialog research software platform. *arXiv preprint arXiv:1705.06476*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS)*.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shimi Salant and Jonathan Berant. 2018. Contextualized word representations for reading comprehension. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the International Conference on Learning Representations*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. DCN+: Mixed objective and deep residual coattention for question answering. In *Proceedings of the International Conference on Learning Representations*.

Zhilin Yang, Saizheng Zhang, Jack Urbanek, Will Feng, Alexander H Miller, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Mastering the dungeon: Grounded language learning by mechanical turker descent. In *Proceedings of the International Conference on Learning Representations*.

A Data Collection Details

A.1 Data Preprocessing

We downloaded the dump of English Wikipedia of October 1, 2017, and extracted text and hyperlinks with WikiExtractor.[8] We use Stanford CoreNLP 3.8.0 (Manning et al., 2014) for word and sentence tokenization. We use the resulting sentence boundaries for collection of supporting facts, and use token boundaries to check whether Turkers are providing answers that cover spans of entire tokens to avoid nonsensical partial-word answers.

A.2 Further Data Collection Details

Details on Curating Wikipedia Pages. To make sure the sampled candidate paragraph pairs are intuitive for crowd workers to ask high-quality multi-hop questions about, we manually curate 591 categories from the lists of popular pages by WikiProject.[9] For each category, we sample (a, b) pairs from the graph G where b is in the considered category, and manually check whether a multi-hop question can be asked given the pair (a, b). Those categories with a high probability of permitting multi-hop questions are selected.

Bonus Structures. To incentivize crowd workers to produce higher-quality data more efficiently, we follow Yang et al. (2018), and employ bonus structures. We mix two settings in our data collection process. In the first setting, we reward the top (in terms of numbers of examples) workers every 200 examples. In the second setting, the workers get bonuses based on their productivity (measured as the number of examples per hour).

A.3 Crowd Worker Interface

Our crowd worker interface is based on ParlAI (Miller et al., 2017), an open-source project that facilitates the development of dialog systems and data collection with a dialog interface. We adapt ParlAI for collecting question answer pairs by converting the collection workflow into a system-oriented dialog. This allows us to have more control over the turkers input, as well as provide turkers with in-the-loop feedbacks or helpful hints to help Turkers finish the task, and therefore speed up the collection process.

Please see Figure 4 for an example of the worker interface during data collection.

Paragraph A, 2015 Miami Dolphins season:

[1] The 2015 Miami Dolphins season was the franchise's 46th season in the National Football League and the 50th overall. [2] The Dolphins look to improve on their 8-8 record from 2014 and return to the playoffs for the first time in seven seasons. [3] However Miami failed to clinch a playoff berth for the seventh consecutive season after a Week 14 loss to the Giants.

Paragraph B, 2008 Miami Dolphins season:

[4] The 2008 Miami Dolphins season was the organization's 39th season in the National Football League and 43rd overall. [5] During the regular season the Dolphins completed the greatest single-season turnaround in NFL history, going from a 1–15 regular season record in 2007 to an 11–5 record in 2008. [6] The previous record for most improved team one year after a 1–15 season belonged to the 1992 Indianapolis Colts, who went 9–7. [7] The 1999 Indianapolis Colts were the only other team to accomplish a 10-game turnaround, winning 13 games after winning 3 in 1998. [8] Additionally, Miami won the AFC East, becoming the first team in NFL history to win their division after only having one win the previous season.

Please type a question given the two paragraphs above, you have 10 min(s) (**0**/1 examples finished so far).

HINT: Maybe ask a question where the answer is 2008 Miami Dolphins season. It should require pie information from both paragraphs.
Example from Tutorial: Which movie starring Ed Harris is based on a French novel? (Info A: starring Harris, Info B: based on a French novel)

Please enter here...

Figure 4: Screenshot of our worker interface on Amazon Mechanical Turk.

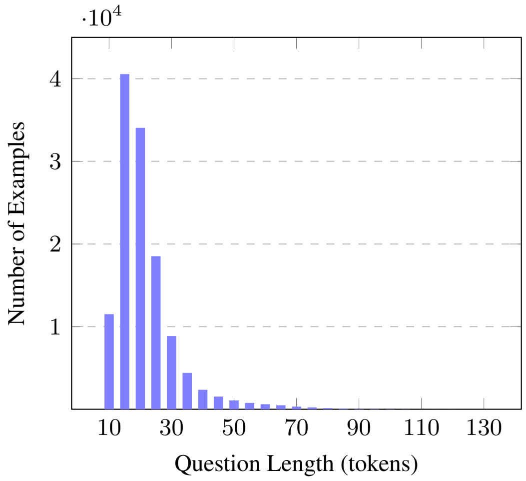

Figure 5: Distribution of lengths of questions in HOTPOTQA.

B Further Data Analysis

To further look into the diversity of the data in HOTPOTQA, we further visualized the distribution of question lengths in the dataset in Figure 5. Besides being diverse in terms of types as is show in the main text, questions also vary greatly in length, indicating different levels of complexity and details covered.

C Full Wiki Setting Details

C.1 The Inverted Index Filtering Strategy

In the full wiki setting, we adopt an efficient inverted-index-based filtering strategy for preliminary candidate paragraph retrieval. We provide details in Algorithm 2, where we set the control threshold $N = 5000$ in our experiments. For some of the question q, its corresponding gold para-

---

[8] https://github.com/attardi/wikiextractor

[9] https://wiki.sh/y8qu

Algorithm 2 Inverted Index Filtering Strategy

```
Input: question text $q$, control threshold $N$, ngram-to-Wikidoc inverted index $\mathcal{D}$
Inintialize:
Extract unigram + bigram set $r_q$ from $q$
$N_{cand} = +\infty$
$C_{gram} = 0$
while $N_{cands} > N$ do
    $C_{gram} = C_{gram} + 1$
    Set $S_{overlap}$ to be an empty dictionary
    for $w \in r_q$ do
        for $d \in \mathcal{D}[w]$ do
            if $d$ not in $S_{overlap}$ then
                $S_{overlap}[d] = 1$
            else
                $S_{overlap}[d] = S_{overlap}[d] + 1$
            end if
        end for
    end for
    $S_{cand} = \emptyset$
    for $d$ in $S_{overlap}$ do
        if $S_{overlap}[d] \geq C_{gram}$ then
            $S_{cand} = S_{cand} \cup \{d\}$
        end if
    end for
    $N_{cands} = |S_{cand}|$
end while
return $S_{cand}$
```

graphs may not be included in the output candidate pool S_{cand}, we set such missing gold paragraph's rank as $|S_{cand}| + 1$ during the evaluation, so MAP and Mean Rank reported in this paper are upper bounds of their true values.

C.2 Compare *train-medium* Split to Hard Ones

Table 9 shows the comparison between *train-medium* split and hard examples like *dev* and *test* under retrieval metrics in full wiki setting. As we can see, the performance gap between *train-medium* split and its *dev*/*test* is close, which implies that *train-medium* split has a similar level of difficulty as hard examples under the full wiki setting in which a retrieval model is necessary as the first processing step.

Set	MAP	Mean Rank	CorAns Rank
train-medium	41.89	288.19	82.76
dev	42.79	304.30	97.93
test	45.92	286.20	74.85

Table 9: Retrieval performance comparison on full wiki setting for *train-medium*, *dev* and *test* with 1,000 random samples each. MAP and are in %. Mean Rank averages over retrieval ranks of two gold paragraphs. CorAns Rank refers to the rank of the gold paragraph containing the answer.

Can a Suit of Armor Conduct Electricity?
A New Dataset for Open Book Question Answering

Todor Mihaylov[‡] and **Peter Clark**[†] and **Tushar Khot**[†] and **Ashish Sabharwal**[†]

[†] Allen Institute for Artificial Intelligence, Seattle, WA, U.S.A.
[‡] Research Training Group AIPHES & Heidelberg University, Heidelberg, Germany
`{peterc,tushark,ashishs}@allenai.org, mihaylov@cl.uni-heidelberg.de`

Abstract

We present a new kind of question answering dataset, OpenBookQA, modeled after open book exams for assessing human understanding of a subject. The open book that comes with our questions is a set of 1326 elementary level science facts. Roughly 6000 questions probe an understanding of these facts and their application to novel situations. This requires combining an open book fact (e.g., metals conduct electricity) with broad common knowledge (e.g., a suit of armor is made of metal) obtained from other sources. While existing QA datasets over documents or knowledge bases, being generally self-contained, focus on linguistic understanding, OpenBookQA probes a deeper understanding of both the topic—in the context of common knowledge—and the language it is expressed in. Human performance on OpenBookQA is close to 92%, but many state-of-the-art pre-trained QA methods perform surprisingly poorly, worse than several simple neural baselines we develop. Our oracle experiments designed to circumvent the knowledge retrieval bottleneck demonstrate the value of both the open book and additional facts. We leave it as a challenge to solve the retrieval problem in this multi-hop setting and to close the large gap to human performance.

1 Introduction

Open book exams are a common mechanism for assessing human understanding of a subject, where test takers are allowed free access to a relevant book, study guide, or class notes when answering questions. In this context, the goal is not to evaluate memorization but a deeper understanding of the material and its application to new situations (Jenkins, 1995; Landsberger, 1996). The application, in turn, often requires combining a fact in the book (e.g., *metals conduct electricity*) with additional common knowledge the test taker is expected to have acquired by this stage (e.g., *a suit of armor is made of metal*).

Question:
Which of these would let the most heat travel through?
A) a new pair of jeans.
B) a steel spoon in a cafeteria.
C) a cotton candy at a store.
D) a calvin klein cotton hat.

Science Fact:
Metal is a thermal conductor.

Common Knowledge:
Steel is made of metal.
Heat travels through a thermal conductor.

Figure 1: An example for a question with a given set of choices and supporting facts.

Motivated by this setting, we present a new kind of question answering dataset, OpenBookQA,[1] that consists of two parts: $\mathcal{Q}$, a set of 5957 multiple-choice questions, and $\mathcal{F}$, a set of 1326 diverse facts about elementary level science. $\mathcal{F}$ has three key characteristics of an 'open book': (a) it forms the basis for generating $\mathcal{Q}$; (b) it has been deemed central to scientific explanations (Jansen et al., 2018); and (c) by itself, $\mathcal{F}$ is generally insufficient to answer questions in $\mathcal{Q}$. Faced with a question $q \in \mathcal{Q}$, a student or system S is expected retrieve a relevant fact $f \in \mathcal{F}$, and appeal to their own common knowledge, $\mathcal{K}_S$, when applying f to answer q.

Figure 1 provides an example. Here, *metals are thermal conductors* is a core scientific fact available in $\mathcal{F}$. One way to apply this fact to decide whether *a steel spoon* would let the *most heat travel through* is to appeal to common knowledge that steel is metallic and heat travels through thermal conductors. In general, the expected common knowledge is relatively simple (taxonomic facts,

---

[1] The dataset and the code for the models are available at `http://data.allenai.org/OpenBookQA`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2381–2391
Brussels, Belgium, October 31 - November 4, 2018.

definitions, object properties, etc.); the difficulty lies in identifying it and meaningfully combining it with a core fact from $\mathcal{F}$ to answer the question.

OpenBookQA questions are challenging as they require *multi-hop reasoning with partial context* provided by $\mathcal{F}$. Specifically, unlike existing datasets for reading comprehension (RC), answering questions on the back of a textbook (TQA),[2] as well as question answering over structured knowledge-bases (KBQA), the open book $\mathcal{F}$ that comes with OpenBookQA is not self-contained. A successful system must therefore go beyond the typical challenges such as paraphrase matching and coreference resolution, without benefiting from the canonicalized and complete information in KBQA.

Generating interesting open book questions is a difficult task. We used a multi-stage process starting with $\mathcal{F}$, using crowd-sourcing to generate (noisy) questions based on $\mathcal{F}$ that probe novel situations, using an automatic filter to ensure hardness for retrieval and association based systems, using a crowd filter to ensure answerability by a lay person, and further using an expert filter to ensure higher quality in Dev and Test sets.

We evaluate a number of existing QA systems for science (without retraining) on OpenBookQA, finding that they perform surprisingly close to the random guessing baseline of 25%. Human performance, on the other hand, is close to 92%.[3]

Motivated by recent findings of gameability of NLP datasets (Gururangan et al., 2018), we also develop and evaluate simple, attention-based, neural baselines including a *plausible answer detector* (which ignores the question text completely) and an *odd-one-out solver*. These highlight inevitable human bias in any crowdsourced dataset, increasing performance on OpenBookQA to 48%.

Building upon a recent neural model for incorporating external knowledge in the story cloze setting (Mihaylov and Frank, 2018), we propose a knowledge-aware neural baseline that can utilize both the open book $\mathcal{F}$ and common knowledge retrieved from sources such as ConceptNet (Speer et al., 2017). While retrieving the most useful pieces of knowledge remains an open challenge, our 'oracle' experiments with the fact f used while generating a question q and an interpretation (by the question author) of the additional knowledge k needed for q, provides valuable insight into the nature of this dataset: Facts from the open book $\mathcal{F}$ are valuable (5% improvement) but not sufficient. Using both f and k increases the accuracy to 76%, but is still far from human level performance, suggesting the need for non-trivial reasoning to combine these facts.

To encourage further research on this new task, for each Train and Dev question q, OpenBookQA also includes f as intermediate supervision signal, which may be viewed as a partial *explanation* for q. We leave closing the large gap to human performance as a challenge for the NLP community.

2 Related Work

By construction, answering OpenBookQA questions requires (i) some base science facts from a provided 'open book', (ii) broader understanding about the world (common or commonsense knowledge), and (iii) an ability to combine these facts (reasoning). This setup differs from several existing QA tasks, as summarized below.

Reading Comprehension (RC) datasets have been proposed as benchmarks to evaluate the ability of systems to understand a document by answering factoid-style questions over this document. These datasets have taken various forms: multiple-choice (Richardson et al., 2013), cloze-style (Hermann et al., 2015; Onishi et al., 2016; Hill et al., 2016), and span prediction (Rajpurkar et al., 2016; Trischler et al., 2017; Joshi et al., 2017) However, analysis (Chen et al., 2016; Sugawara et al., 2017) of these datasets has shown that many of the questions can be solved with context token matching (Chen et al., 2017a; Weissenborn et al., 2017) or relatively simple paraphrasing.

To focus on the more challenging problem of reasoning across sentences, new datasets have been proposed for multi-step RC. **QAngaroo** (Welbl et al., 2018) have used a knowledge-base to identify entity pairs (s, o) with a known relation, r, which is also supported by a multi-hop path in a set of documents. They use structured tuple queries (s, r, ?) and use all the documents along the path as the input passage. **NarrativeQA** (Kociský et al., 2017) is an RC dataset that has been shown to require an iterative reasoning about the narrative of a story. Similar to OpenBookQA, the questions were generated to ensure that the answer is not a direct match or paraphrase

---

[2] Only ∼5% of the TQA questions of Kembhavi et al. (2017) require additional common knowledge.

[3] To avoid ambiguity in the term 'human performance', Section 3.2 describes the specific randomized model we use.

that can be retrieved with an IR approach. Most recently, Khashabi et al. (2018) proposed **MultiRC**, a multiple-choice RC dataset that is designed to require multi-sentence reasoning and can have multiple correct answers. Again, like most RC datasets, it is self-contained.

Tasks with external knowledge. While many of the RC datasets could benefit from commonsense or background knowledge, they are designed to be self-contained, i.e., solvable by the document context alone. Datasets such as the **Story Cloze Test** (Mostafazadeh et al., 2016), **MCScript**,[4] and **ProPara** (Mishra et al., 2018) do require additional domain knowledge about everyday events, scripts, and processes, respectively. However, these datasets need domain-specific modeling of events, whereas OpenBookQA appeals to broad common knowledge cutting across a variety of types and topics.

Stasaski and Hearst (2017) explore the creation of multi-hop questions and propose generating stronger distractors for the multiple-choice setting. Their work, however, starts with structured knowledge, specifically a Biology ontology.

Lastly, many **Science Question Answering** datasets (e.g. Clark et al., 2016, 2018) have been released that need broad external knowledge to answer the questions. However, these questions are not associated with a core set of facts, i.e., an "open book" used to define these questions. As a result, the questions vary widely in style and complexity (Clark et al., 2018). In contrast, OpenBookQA focuses on a more well-defined subset of science QA, appealing to one core fact from the open book and one (or few) relatively simple commonly known supporting facts.

3 OpenBookQA Dataset

The OpenBookQA dataset consists of about 6,000 4-way multiple-choice questions, each associated with one core fact from a "book" $\mathcal{F}$ of 1326 such facts, and an auxiliary set $\mathcal{K}$ of about 6000 additional facts. The questions were created via a multi-stage crowdsourcing and partial expert filtering process, discussed in Section 3.1.

The small "book" $\mathcal{F}$ consists of recurring science themes and principles, each of which can be (and here is) instantiated into multiple questions. For $\mathcal{F}$, we use a subset of the WorldTree corpus which Jansen et al. (2018) have analyzed for sufficiency for elementary level science. The subset we use is taken from the 2287 WorldTree facts that were marked as "central" by the original authors in at least one explanation. We further filter them down to 1326 that appear general enough to be applicable to multiple situations.

OpenBookQA additionally requires broad common knowledge, which is expected to come from large corpora, such as ConceptNet, Wikipedia, or a corpus with 14M science-related sentences used by some existing baselines. The crowdsourcing process below also asks workers to mark a second fact, k, needed for each question q, in addition to f. These second facts, unfortunately, were often incomplete, over-complete, or only distantly related to q. We thus include in OpenBookQA the set $\mathcal{K}$ of such second facts only as *auxiliary data* for optional use. We emphasize that $\mathcal{K}$ should not be viewed as 'gold' additional facts, or as a substitute for broad common knowledge.

3.1 Crowdsourcing Process

The overall question generation and filtering pipeline is summarized in Figure 2. Given the "book" $\mathcal{F}$ of core facts, the process proceeds as follows, starting with an empty question set Qs and an empty 'second facts' set $\mathcal{K}$:

1. A crowd-worker[5] w is shown a random science fact f from the set $\mathcal{F}$.

2. w is asked to think of a second common fact, k, that may be combined with f to derive a new, valid assertion s.

3. w then converts s into a question-answer pair and extends this into a 4-way multiple choice question by adding 3 incorrect answer choices, $q_{\text{mc}} = (q, \{c_1, c_2, c_3, c_4\})$, where one of the c_i's is the unique correct answer.

4. The system verifies q_{mc} passes basic checks such as uniformity of answer choices.[6]

5. w then feeds the multiple-choice question q_{mc} to an information retrieval solver (Clark et al.,

---

[4] SemEval-2018 Task 11: Machine Comprehension using Commonsense Knowledge `https://competitions.codalab.org/competitions/17184`

[5] We used Amazon Mechnical Turk, with workers from North America and with a 'masters' level qualification.

[6] Specifically, it looks for: 1) exactly 4 answer choices; 2) no negation words to trivially fool baselines (no, none, not, isn't, doesn't, aren't, don't, won't, except, can't, shouldn't, wouldn't, couldn't, mustn't); 3) uniform answer choice length: all with at most 3 or at least 4 words.

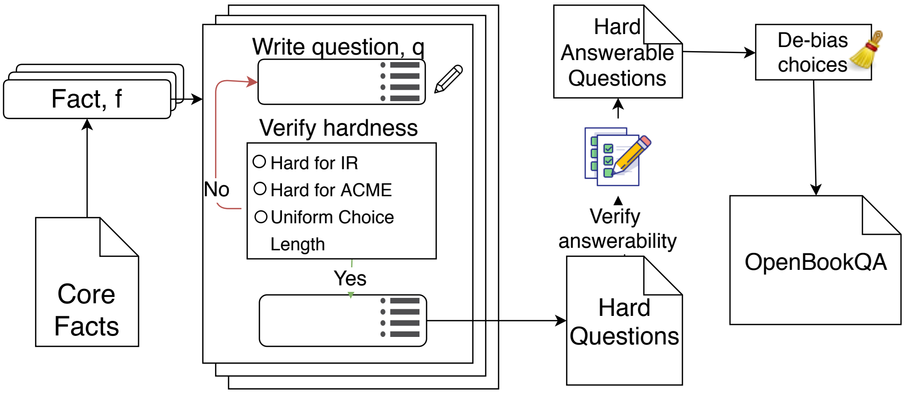

Figure 2: OpenBookQA question generation pipeline

2016) and a word association based solver (Turney, 2017), and verifies that (a) neither of them answers q_{mc} correctly and (b) the top 3 IR retrieved sentences are insufficient to answer q_{mc}; if not, the question is edited and re-tried.

6. Question q_{mc} is then shown to 5 new crowd-workers, who are asked to answer it.

7. If at least 4 out of 5 workers answer q_{mc} correctly, it is deemed answerable and the process continues. If not, q_{mc} is discarded.

8. The answer choices of q_{mc} are randomly shuffled to avoid unintended bias.[7]

9. q_{mc} is associated with f as the core science fact and added to the question set $\mathcal{Q}$. k is added to the set $\mathcal{K}$ of additional (noisy) facts.

The Dev and Test splits were further filtered by an in-house expert to ensure higher quality.

3.2 Human Performance

To assess human accuracy on this dataset, we consider the following model: Each question $q \in \mathcal{Q}$ has some (unknown) human accuracy p_q, defined as the probability that a random human subject, chosen uniformly from a large pool $\mathcal{H}$, would answer q correctly. Thus, we can think of this as defining a Bernoulli random variable, $X_q \sim B(p_q)$, whose mean is (unknown) p_q. The average human accuracy on $\mathcal{Q}$ under this model is:

$$H(\mathcal{Q}) = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} p_q$$

where $\{p_q \mid q \in \mathcal{Q}\}$ are unknown.

With $\mathcal{H}$ as the set of crowd-workers (cf. Footnote 5), step 6 of the above question generation process is equivalent to obtaining 5 independent samples, $X_{q,i}, i \in I, |I| = 5$, from $B(p_q)$. We must, however, be careful when using this data to estimate p_q, as the same 5 samples were used to decide whether q makes it into the question set $\mathcal{Q}$ or not. For instance, if we had kept only those questions that all 5 workers answered correctly, it would clearly be inaccurate to claim that the human accuracy on $\mathcal{Q}$ is 100%. Nevertheless, it is possible to re-use the judgments from Step 6 to approximate $H(\mathcal{Q})$ with high confidence, without posing the questions to new workers.

Intuitively, if all questions in $\mathcal{Q}$ were difficult to answer (i.e., all p_q were small), it would be unlikely that all $|\mathcal{Q}|$ questions would pass the test in Step 6. We can use the contrapositive of this observation to conclude that p_q, on average, must have been high for $q \in \mathcal{Q}$.

Formally, aggregating across all questions gives the following empirical estimate of $H(\mathcal{Q})$:

$$\begin{aligned}\tilde{H}(\mathcal{Q}) &= \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{|I|} \sum_{i \in I} X_{q,i} \\ &= \frac{1}{|\mathcal{Q}||I|} \sum_{q \in \mathcal{Q}, i \in |I|} X_{q,i}\end{aligned}$$

For analysis, we assume all samples $X_{q,i}$ are independent, i.e., every answer is obtained independently.[8] An application of Hoeffding's Inequality (Hoeffding, 1963) shows that $\tilde{H}(\mathcal{Q})$ converges to $H(\mathcal{Q})$ very rapidly as $n = |\mathcal{Q}||I|$ grows; specifically, $\tilde{H}(\mathcal{Q}) \leq H(\mathcal{Q}) + t$ with probability at least $1 - \exp(-2nt^2)$; similarly for $\tilde{H}(\mathcal{Q}) \geq H(\mathcal{Q}) - t$. In our Dev and Test sets, where $|\mathcal{Q}| = 500$ and $|I| = 5$, this translates into $H(Q)$ being at least

---

[7] Choice 'A' was the correct answer in 69% of the questions at the end of Step 4.

[8] Realistically, there is some dependence across questions as a single worker may answer multiple questions. We leave a formal analysis of this setting as future work.

OpenBookQA Statistics	
# of questions	5957
# of choices per question	4
Avg. question sentences	1.08 (6)
Avg. question tokens	11.46 (76)
Avg. choice tokens	2.89 (23)
Avg. science fact tokens	9.38 (28)
Vocabulary size (q+c)	11855
Vocabulary size (q+c+f)	12839
Answer is the longest choice	1108 (18.6%)
Answer is the shortest choice	216 (3.6%)

Table 1: Statistics for full OpenBookQA dataset. Parenthetical numbers next to each average are the *max*.

$\tilde{H}(Q) - 3\%$ with probability over 98.8% and at least $\tilde{H}(Q) - 2.5\%$ with prob 95.6%; we report the former as our conservative estimate on human performance.

3.3 Question Set Analysis

OpenBookQA consists of 5957 questions, with 4957/500/500 in the Train/Dev/Test splits.[9] Table 1 summarizes some statistics about the full dataset. Each question has exactly four answer choices and one associated fact used in the creation process. We report the average length of questions, candidate choices, and associated facts, as well as how often is the longest/shortest choice the correct one.

We analyzed 100 questions in the Train set to capture the kind of common knowledge and reasoning needed. For each, we wrote down the additional common knowledge needed to answer this question *in addition* to the original science fact. In 21% of the cases, the crowdsourced question actually tests for a fact that doesn't necessarily need the original science fact. For example, the question: "On a rainy day the clouds are (A) low (B) white (C) small (D) gray" was written based on the science fact "clouds produce rain" but doesn't need this fact to answer it. We ignore such questions in our analysis. For the remaining questions, we categorized the additional facts into five high-level categories (and collapsed the remaining facts into a catch-all OTHERS category) based on previous approaches on similar science questions (Clark et al., 2018; Jansen et al., 2016):

1. ISA: Basic taxonomic facts such as isa(tree, living thing), isa(granite, rock).
2. PROPERTY: Properties of objects such as madeof(belt buckle, metal), has(mammals, four legs), contains(lemon juice, citric acid).
3. DEFINITION: Definitions of objects that may be based on their appearance (tape is a plastic with markings), working mechanism (telescope is a device that uses mirrors to view objects), etc.
4. CAUSAL: Causal facts such as causes(adding lemon juice to milk, milk to break down).
5. BASIC: General scientific fact that did not fit above, e.g. squirrels eat nuts for food.

Fact Type	% Questions	% Facts
PROPERTY	29.11%	25.81%
ISA	20.25%	17.20%
BASIC	17.72%	19.35%
DEFINITION	17.72%	15.05%
CAUSAL	11.39%	9.68%
OTHERS	13.92%	12.90%

Table 2: Percentage of questions and facts for the five most common type of additional facts. Note that % Questions does not add up to 100% since we count the percentage of questions where at least one such fact is needed.

Table 2 presents the proportions of these facts in our analyzed question set. For each type of fact, we calculate the percentage of questions that need at least one such fact (shown as % Questions). We also calculate the overall percentage of each fact type across all the common knowledge facts (shown as % Facts). Most of our questions need simple facts such as *isa* knowledge and properties of objects, further confirming the need for simple reasoning with common knowledge. Apart from these five major categories of facts, the catch-all OTHERS category contains commonsense facts (e.g., it is dark at night), world knowledge (e.g., Japan is often hit by earthquakes) and lexical rewrites[10] (e.g., *ad infinitum* means over and over).

Most of our questions need simple facts that should be easily retrievable from any knowledge-base/textual corpora. On an average, each question needed 1.16 additional facts ignoring any linguistic variations. Despite the simplicity of the knowledge needed for these questions, as we show

---

[9] Overall, 8140 questions were collected, of which 2183 were discarded in crowdsourcing Step 7.

[10] Of course, every question had lexical variations. We marked it when this was the *only* change to the core fact.

empirically, most baseline approaches achieve a relatively low score on this dataset (even when the core fact is provided). We claim that this is due to the fact that the reasoning needed to answer these questions is non-trivial. Table 3 shows few questions with the associated facts and high-level reasoning needed to answer these questions. Assuming a model can extract the described relations (e.g. defn, contains), the QA system still needs to be able to chain these facts together, identify the resulting relation and verify its expression for each choice. In the extreme case (as shown in the last example), even though only one additional fact is needed to answer the question, it needs a system to *apply* the core "general" science fact to a "specific" situation.

4 Baseline Models

We evaluate the performance of several baselines systems on the Dev and Test subsets of OpenBookQA. For each question, a solver receives 1 point towards this score if it chooses the correct answer, and $1/k$ if it reports a k-way tie that includes the correct answer. The "Guess All" baseline, which always outputs a 4-way tie, thus achieves a score of 25%, same as the expected performance of a uniform random baseline.

4.1 No Training, External Knowledge Only

Since OpenBookQA is a set of elementary level science questions, one natural baseline category is existing systems that have proven to be effective on elementary- and middle-school level science exams. These pre-trained systems, however, rely only on their background knowledge and do not take the set $\mathcal{F}$ of core facts into account. Further, their knowledge sources and retrieval mechanism are close to those used by the IR solver that, by design, is guaranteed to fail on OpenBookQA. These two aspects place a natural limit on the effectiveness of these solvers on OpenBookQA, despite their excellent fit for the domain of multiple-choice science questions. We consider four such solvers.

PMI (Clark et al., 2016) uses pointwise mutual information (PMI) to score each answer choice using statistics based on a corpus of 280 GB of plain text. It extracts unigrams, bigrams, trigrams, and skip-bigrams from the question q and each answer choice c_i. Each answer choice is scored based on the average PMI across all pairs of question and answer n-grams.

TableILP (Khashabi et al., 2016) is an Integer Linear Programming (ILP) based reasoning system designed for science questions. It operates over semi-structured relational tables of knowledge. It scores each answer choice based on the optimal (as defined by the ILP objective) "support graph" connecting the question to that answer through table rows. The small set of these knowledge tables, however, often results in missing knowledge, making TableILP not answer 24% of the OpenBookQA questions at all.

TupleInference (Khot et al., 2017), also an ILP-based QA system, uses Open IE tuples (Banko et al., 2007) as its semi-structured representation. It builds these subject-verb-object tuples *on-the-fly* by retrieving text for each question from a large corpus. It then defines an ILP program to combine evidence from multiple tuples.

DGEM (Khot et al., 2018) is a neural entailment model that also uses Open IE to produce a semi-structured representation. We use the adaptation of this model to multiple-choice question answering proposed by Clark et al. (2018), which works as follows: (1) convert q and each c_i into a hypothesis, h_i, and each retrieved fact into a premise p_j; and (2) return the answer choice with the highest entailment score, $\arg\max_i e(p_j, h_i)$.

4.2 No Training; $\mathcal{F}$ and Extr. Knowledge

We also consider providing the set $\mathcal{F}$ of core facts to two existing solvers: the **IR** solver of Clark et al. (2016) (to assess how far simple word-overlap can get), and the **TupleInference** solver.

4.3 Trained Models, No Knowledge

We consider several neural baseline models that are trained using Train set of OpenBookQA. For ease of explanation, we first define the notation used in our models. For a given question $q_{\mathrm{mc}} = (q, \{c_1, c_2, c_3, c_4\})$, we define the set of token sequences , $\mathcal{S} = \{q, c_1, c_2, c_3, c_4\}$. For each token sequence $s \in \mathcal{S}$, w^s_j is the j^{th} and $e^s_j = \mathrm{Emb}(w^s_j)$ is the embedding for this token. We use n_s to indicate the number of tokens in s and d for the dimensionality of the embeddings.[11] We model multiple-choice QA as multi-class classification: Given q_{mc}, predict one of four class labels $L =$

---

[11] For all experiments we use $d = 300$ *GloVe* (Pennington et al., 2014) embeddings pre-trained on 840B tokens from *Common Crawl* (https://nlp.stanford.edu/projects/glove/).

Question	Science Fact	Common Knowledge (Type)	Reasoning Challenge
What is the most likely to be an effect of acid rain on an aquatic environment? (A) increase in plant growth (B) increase in fish population **(C) decrease in plant life** (D) cleaner and clearer water	acid rain has a negative impact on water quality	decrease in water quality leads to a decrease in aquatic life (CAUSAL)	causes(x, y) ∧ causes(y, z) ⇒ causes(x, z)
The moon's surface (A) is smooth on the entire surface (B) contains an internal core of cheese (C) is filled with lakes **(D) contains large cavities cause by explosions**	the moon's surface contains many craters	Craters are large cavities caused by explosions (DEFINITION)	contains(x, y) ∧ defn(y, z) ⇒ contains(x, z)
As a car approaches you in the night (A) the headlights remain at a constant (B) the headlights turn off **(C) the headlights become more intense** (D) the headlights recede into the dark	as a source of light becomes closer, that source will appear brighter	Headlights of a car are source of light (PROPERTY)	[lhs ⇒ rhs] ⇒ [ground(lhs) ⇒ ground(rhs)]

Table 3: Example training questions (with their correct choices **marked**) along with the facts and reasoning needed. In the last example, the science fact states that lhs="source of light becomes closer" implies rhs="source will appear brighter". Grounding this rule based on the common-knowledge fact, produces a new rule: "As headlights of the car come closer, headlights will appear brighter"

$\{1, 2, 3, 4\}$, where the true label is the correct answer index.

Embeddings + Similarities as Features. We first experiment with a simple logistic regression model (Mihaylov and Nakov, 2016; Mihaylov and Frank, 2016, 2017) that uses centroid vectors r_s^{emb} of the word embeddings of tokens in s, and then computes the cosine similarities between the question and each answer choice, $r_{q,c_i}^{\cos}$:

$$r_s^{\text{emb}} = \frac{1}{n_s}\sum_{j=1}^{n_s} e_{s_j} \in \mathbb{R}^d$$

$$r_{q,c_i}^{\cos} = \cos(r_q^{\text{emb}}, r_{c_i}^{\text{emb}}) \in \mathbb{R}^1$$

For each training instance, we build a feature representations $\vec{f}$ by concatenating these vectors and train an $L2$ logistic regression classifier:

$$\vec{f} = [r_q^{\text{emb}}; r_{c_{1..4}}^{\text{emb}}; r_{q,c_{1..4}}^{\cos}] \in \mathbb{R}^{5d+4}$$

BiLSTM Max-Out Baselines. As a simple neural baseline, we adapt *BiLSTM max-out* model (Conneau et al., 2017) to our QA task. That is, we first encode the question tokens and choice tokens $w_{1 \ldots n_s}^s$, independently with a bi-directional context encoder (LSTM) to obtain a context (ctx) representation $h_{s_{1\ldots n_s}}^{\text{ctx}} = \text{BiLSTM}(e_{1\ldots n_s}^s) \in \mathbb{R}^{n_s \times 2h}$ Next, we perform an element-wise aggregation operation max on the encoded representations $h_{s_{1..n_s}}^{\text{ctx}}$ to construct a single vector:

$$r_s^{\text{ctx}} = \max(h_{s_{1..n_s}}^{\text{ctx}}) \in \mathbb{R}^{2h}. \qquad (1)$$

Given the contextual representations for each token sequence, we experiment with three configurations for using these representations for QA:

(a) Plausible Answer Detector. This baseline goes to the extreme of completely ignoring q and trying to learn how plausible it is for c_i to be the correct answer to *some* question in this domain. This captures the fact that certain choices like 'a magical place' or 'flying cats' are highly unlikely to be the correct answer to a science question without negation (which is the case for OpenBookQA).

We implement a plausible answer detector using a *choice-only* model for predicting the answer by obtaining a score α_{c_i} as: $\alpha_{c_i} = W_c^T r_{c_i}^{\text{ctx}} \in \mathbb{R}^1$, where $W_c^T \in \mathbb{R}^{2h}$ is a weights vector optimized during training, $i = \{1..4\}$ is the index of the choice. To obtain the answer choice from the set of choice scores $\alpha_{c_{1..4}}$ using $\arg\max(\text{softmax}(\alpha_{c_{1..4}}))$, where $\text{softmax}(\alpha_{c_i}) = \frac{\exp(\alpha_{c_i})}{\sum_{j=1}^{4}\exp(\alpha_{c_j})}$ as usual.

(b) Odd-One-Out Solver. It considers all 4 answer options jointly and selects the one that is least similar to the others. This captures bias in human authored questions arising from the fact that creating good quality incorrect answers is difficult. Workers generally start with the correct answer, and then come up with three incorrect ones. The latter often tend to be homogeneous or share other common properties (e.g., non-scientific terms) uncharacteristic of the correct answer.

We implement this using a *choice-to-choices* attention model. For each choice c_i, we calculate the attention to the other choices as α_{c_i,c_j}. We then sum these attention values to compute the attention for c_i to the rest of the choices, $\alpha_{c_i 2 c_{r(est)}}$, and return the choice with the lowest sum. The attention is computed as $\alpha_{c_i,c_j} = \mathrm{Att}(r^{\mathrm{ctx}}_{c_i}, r^{\mathrm{ctx}}_{c_j})$ where

$$\mathrm{Att}(u, v) = W^T([u; v; u \cdot v; |u - v|]) \in \mathbb{R}^1$$

is a linear attention function and $W \in \mathbb{R}^{8h}$ is a weight vector. We then compute $\alpha_{c_i 2c_{r(est)}} = \sum_{j=1}^{4} \alpha_{c_i,c_j}$ $(j \neq i)$ and select the answer with the index $a_{c2c_r} = \arg\min(\mathrm{softmax}(\alpha_{c_{1..4}2c_r}))$.

(c) Question Match. This solver tries to predict which choice best matches the question (Nakov et al., 2016), without relying on external knowledge. To achieve that, we compute an attention score α_{q,c_i} between q and each of the choices q_i as $\alpha_{q,c_i} = \mathrm{Att}(r^{\mathrm{ctx}}_{q}, r^{\mathrm{ctx}}_{c_i})$, and select the one with the highest score. We also experiment with a model where r^{ctx}_{q} and $r^{\mathrm{ctx}}_{c_i}$ are obtained using token-wise interaction proposed in ESIM (Chen et al., 2017b).

4.4 Trained Model with External Knowledge

Lastly, we implement a two stage model for incorporating external common knowledge, K. The first module performs information retrieval on K to select a fixed size subset of potentially relevant facts $K_{Q,C}$ for each instance in the dataset (see Appendix A). The second module is a neural network that takes (Q, C, $K_{Q,C}$) as input to predict the answer $a_{q,c}$ to a question Q from the set of choices C.

Knowledge-Enhanced Reader. As a base knowledge-aware model, we use a variant of the model of Mihaylov and Frank (2018), implemented by extending our **BiLSTM max-out question-match baseline (c)**. For each instance the model reads the question q and answers $c_{1..4}$ independently and attends to the set of retrieved external knowledge facts $\mathrm{K}_{Q,C}$. We encode each fact k_j from $\mathrm{K}_{Q,C} = k_{1..N_k}$ (N_k is the number of facts) with same BiLSTM as used for q and $c_{1..4}$ and construct a single vector $r^{\mathrm{ctx}}_{k_j} \in \mathbb{R}^{2h}$ using Eq. 1. Having such representations for each k_j results in knowledge memory matrix $M_k = r^{\mathrm{ctx}}_{k_{1..N_k}} \in \mathbb{R}^{N_k \times 2h}$. Note that M_k is dynamic memory, specific for each instance in the batch and is encoded in each step during training. This memory is used to calculate a knowledge-aware representation, $r^{\mathrm{kn}}_{s} = \sum((M_k^T r^{\mathrm{ctx}}_{s}).M_k) \in \mathbb{R}^{2h}$. Each context (ctx) representation r^{ctx}_{s} $(s \in \mathcal{S})$ is combined with r^{kn}_{s} to obtain a knowledge-enhanced representation $r^{\mathrm{ctx+kn}}_{s} = (r^{\mathrm{ctx}}_{s} + r^{\mathrm{kn}}_{s})/2$. We then model the knowledge-enhanced attention $\alpha^{\mathrm{kn}}_{q,c_i}$ between q and c_i as a linear combination of the ctx, kn and ctx + kn representations as

$$\begin{aligned}\alpha_{q,c_i} = W^T[&\mathrm{Att}(r^{\mathrm{ctx}}_{s}, r^{\mathrm{ctx}}_{c_i}); \mathrm{Att}(r^{\mathrm{kn}}_{s}, r^{\mathrm{kn}}_{c_i});\\ &\mathrm{Att}(r^{\mathrm{ctx+kn}}_{s}, r^{\mathrm{ctx}}_{c_i}); \mathrm{Att}(r^{\mathrm{ctx}}_{s}, r^{\mathrm{ctx+kn}}_{c_i});\\ &\mathrm{Att}(r^{\mathrm{ctx}}_{s}, r^{\mathrm{kn}}_{c_i}); \mathrm{Att}(r^{\mathrm{kn}}_{s}, r^{\mathrm{ctx}}_{c_i});\\ &\mathrm{Att}(r^{\mathrm{ctx+kn}}_{s}, r^{\mathrm{kn}}_{c_i}); \mathrm{Att}(r^{\mathrm{kn}}_{s}, r^{\mathrm{ctx+kn}}_{c_i});\\ &\mathrm{Att}(r^{\mathrm{ctx+kn}}_{s}, r^{\mathrm{ctx+kn}}_{c_i})],\end{aligned}$$

where $W \in \mathbb{R}^9$ is a weight vector initialized with the *ones* vector and optimized during training. We then select the answer c_i with the highest score.

Solver	Dev	Test
Human solver	89.3*	91.7*
Guess All ("random")	25.0	25.0
NO TRAINING, KB ONLY (§4.1)		
TupleInference	15.9	17.9
PMI (Waterloo corpus)	19.7	21.2
TableILP	20.0	23.4
DGEM	27.4	**24.4**
NO TRAINING, KB + $\mathcal{F}$ (§4.2)		
IR with $\mathcal{F}$	25.5	24.8
TupleInference with $\mathcal{F}$	23.6	**26.6**
DGEM with $\mathcal{F}$	28.2	24.6
TRAINED MODELS, NO $\mathcal{F}$ OR KB (§4.3)		
Embedd+Sim	44.6	41.8
ESIM	53.9±0.4	48.9±1.1
Plausible Answer Detector	54.4±0.7	49.6±0.7
Odd-one-out Solver	56.9±0.5	50.2±1.6
Question Match	54.6±1.2	**50.2±0.9**
ORACLE MODELS, $\mathcal{F}$ AND/OR KB (§4.4)		
f	63.0±2.3	55.8±2.3
f + WordNet	57.6±1.4	56.3±1.3
f + ConceptNet	57.0±1.6	53.7±1.5
f + k	80.2±1.1	76.9±0.7

Table 4: Scores obtained by various solvers on OpenBookQA, reported as a percentage ± the standard deviation across 5 runs with different random seeds. Other baselines are described in the corresponding referenced section. For oracle evaluation, we use the gold science fact f associated with each question, and optionally the additional fact k provided by the question author. Bold denotes the best Test score in each category.

5 Baseline Performance

The results for various baseline models are summarized in Table 4, grouped by method category. We make a few observations:

First, the task is **largely solvable by a lay-person**, as evidenced by the 92% score of crowd-workers. This is measured as described in Section 3.2. We use annotations from Step 6 of the question generation process and report $\tilde{H}(Q) - 3\%$ as a conservative lower estimate. As an additional assessment, we also obtained 5 *new* annotations for 100 randomly chosen questions from each of Train, Dev, and Test sets. The performance remained similar at 88.6%, 90.2%, and 91.6%, resp.

The **second group** shows that pre-trained state-of-the-art solvers for multiple-choice science questions perform poorly. One explanation is their correlation with the the IR method used for question filtering, as mentioned in Section 4.1.

The **third group** of results suggests that adding $\mathcal{F}$ to pre-trained models has a mixed effect, improving TupleInference by 8.7% but not changing DGEM.[12] Unlike DGEM, TupleInference relies on brittle word-overlap similarity measures very similar to the ones used by IR. Since IR (KB) gets 0% by design, TupleInference (KB) also has poor performance and adding $\mathcal{F}$ helps it find better support despite the brittle measures.

The **fourth group** demonstrates that carefully designed trainable neural models—even if simplistic and knowledge-free—can be surprisingly powerful. For example, the "plausible answer detector" can predict the correct answer with 49.6% accuracy without even looking at the question. The "odd-one-out" solver, by considering other answer choices, raises this to 50.2%. The "question match" solver, which simply compares the BiLSTM max-out encoding of the question with that of various answer choices, also achieves 50.2%.[13] Similar findings have been reported for several recent datasets (Gururangan et al., 2018), making it imperative to perform such tests early.

Interestingly, all of these neural knowledge-free baselines simultaneously succeed on 34.4% of the Dev questions, and simultaneously fail on 23.6%. For **Question Match** and **ESIM** we also experiment with ElMo (Peters et al., 2018) which improved their score on Test with 0.4% and 1.8%.

The **final group** demonstrates the need for external knowledge and deeper reasoning. When the "oracle" science fact f used by the question author is provided to the knowledge-enhanced reader, it improves over the knowledge-less models by about 5%. However, there is still a large gap, showing that the core fact is insufficient to answer the question. When we also include facts retrieved from WordNet (Miller et al., 1990), the score improves by about 0.5%. Unlike the WordNet gain, adding ConceptNet (Speer et al., 2017) introduces a distraction and reduces the score. This suggests that ConceptNet is either not a good source of knowledge for our task, or only a subset of its relations should be considered. Overall, external knowledge helps, although retrieving the right bits of knowledge remains difficult. In the last row of Table 4, we use the oracle core fact along with question author's interpretation of the additional fact k. This increases the scores substantially, to about 76%. This big jump shows that improved knowledge retrieval should help on this task. At the same time, we are still not close to the human performance level of 92% due to various reasons: (a) the additional fact needed can be subjective, as hinted at by our earlier analysis; (b) the authored facts $\mathcal{K}$ tend to be noisy (incomplete, over-complete, or only distantly related), also as mentioned earlier; and (b) even given the true gold facts, performing reliable "reasoning" to link them properly remains a challenge.

Sample predictions and analysis of questions from Dev are provided in Appendix D.

6 Conclusion

We present a new dataset, OpenBookQA, of about 6000 questions for open book question answering. The task focuses on the challenge of combining a corpus of provided science facts (open book) with external broad common knowledge. We show that this dataset requires simple common knowledge beyond the provided core facts, as well as multi-hop reasoning combining the two. While simple neural methods are able to achieve an accuracy of about 50%, this is still far from the human performance of 92% on this task. We leave closing this gap for future research, and illustrate, via oracle-style experiments, the potential of better retrieval and reasoning on this task.

Acknowledgments

The authors would like to thank Lane Aasen for helping develop the infrastructure for the crowdsourcing task, and Madeleine van Zuylen for providing expert annotation for the Dev and Test questions.

---

[12] By design, IR with its default corpus gets 0% on OpenBookQA. Hence we don't consider the effect of adding $\mathcal{F}$, which appears artificially magnified.

[13] This model also achieves the current best score, 33.87%, on the ARC Reasoning Challenge (Clark et al., 2018). When adapted for the textual entailment task by comparing BiLSTM max-out encodings of premise and hypothesis, it achieves 85% on the SciTail dataset (Khot et al., 2018).

References

M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. In *IJCAI*.

D. Chen, J. Bolton, and C. D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *ACL*, pages 2358–2367.

D. Chen, A. Fisch, J. Weston, and A. Bordes. 2017a. Reading wikipedia to answer open-domain questions. In *ACL*.

Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen. 2017b. Enhanced lstm for natural language inference. In *ACL*, pages 1657–1668.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

P. Clark, O. Etzioni, T. Khot, A. Sabharwal, O. Tafjord, P. D. Turney, and D. Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *AAAI*, pages 2580–2586.

A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680.

M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L. S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. R. Bowman, and N. A. Smith. 2018. Annotation artifacts in natural language inference data. In *NAACL*.

K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701.

F. Hill, A. Bordes, S. Chopra, and J. Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*.

W. Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

P. Jansen, N. Balasubramanian, M. Surdeanu, and P. Clark. 2016. What's in an explanation? characterizing knowledge and inference requirements for elementary science exams. In *COLING*.

P. A. Jansen, E. Wainwright, S. Marmorstein, and C. T. Morrison. 2018. WorldTree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. In *LREC*.

T. Jenkins. 1995. Open book assessment in computing degree programmes 1. Technical Report 95.28, University of Leeds.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, pages 1601–1611.

A. Kembhavi, M. J. Seo, D. Schwenk, J. Choi, A. Farhadi, and H. Hajishirzi. 2017. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *CVPR*, pages 5376–5384.

D. Khashabi, S. Chaturvedi, M. Roth, S. Upadhyay, and D. Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL*.

D. Khashabi, T. Khot, A. Sabharwal, P. Clark, O. Etzioni, and D. Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *IJCAI*.

T. Khot, A. Sabharwal, and P. Clark. 2017. Answering complex questions using open information extraction. In *ACL*.

T. Khot, A. Sabharwal, and P. Clark. 2018. SciTail: A textual entailment dataset from science question answering. In *AAAI*.

D. P. Kingma and J. L. Ba. 2015. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations 2015*, pages 1–15.

T. Kociský, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette. 2017. The NarrativeQA reading comprehension challenge. *CoRR*, abs/1712.07040.

J. Landsberger. 1996. Study guides and strategies. Http://www.studygs.net/tsttak7.htm.

T. Mihaylov and A. Frank. 2016. Discourse relation sense classification using cross-argument semantic similarity based on word embeddings. In *CoNLL-16 shared task*, pages 100–107.

T. Mihaylov and A. Frank. 2017. Story Cloze Ending Selection Baselines and Data Examination. In *LSD-Sem Shared Task*.

T. Mihaylov and A. Frank. 2018. Knowledgeable Reader: Enhancing Cloze-Style Reading Comprehension with External Commonsense Knowledge. In *ACL*, pages 821–832.

T. Mihaylov and P. Nakov. 2016. SemanticZ at SemEval-2016 Task 3: Ranking relevant answers in community question answering using semantic similarity based on fine-tuned word embeddings. In *SemEval '16*.

G. A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. 1990. Introduction to WordNet: An online lexical database. *International Journal of Lexicography*, 3(4):235–244.

B. D. Mishra, L. Huang, N. Tandon, W. tau Yih, and P. Clark. 2018. Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension. In *NAACL*.

N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, and J. Allen. 2016. A Corpus and Evaluation Framework for Deeper Understanding of Commonsense Stories. In *NAACL*.

P. Nakov, L. Màrquez, A. Moschitti, W. Magdy, H. Mubarak, a. A. Freihat, J. Glass, and B. Randeree. 2016. Semeval-2016 task 3: Community question answering. In *SemEval '16*, pages 525–545.

T. Onishi, H. Wang, M. Bansal, K. Gimpel, and D. McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *EMNLP*, pages 2230–2235, Austin, Texas.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

J. Pennington, R. Socher, and C. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

M. Richardson, C. J. Burges, and E. Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, pages 193–203.

P. Singh, T. Lin, E. Mueller, G. Lim, T. Perkins, and W. Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *Lecture Notes in Computer Science*, volume 2519, pages 1223–1237.

R. Speer, J. Chin, and C. Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

K. Stasaski and M. A. Hearst. 2017. Multiple choice question generation utilizing an ontology. In *BEA@EMNLP, 12th Workshop on Innovative Use of NLP for Building Educational Applications*.

S. Sugawara, H. Yokono, and A. Aizawa. 2017. Prerequisite skills for reading comprehension: Multi-perspective analysis of mctest datasets and systems. In *AAAI*, pages 3089–3096.

A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200.

P. D. Turney. 2017. Leveraging term banks for answering complex questions: A case for sparse vectors. *CoRR*, abs/1704.03543.

D. Weissenborn, G. Wiese, and L. Seiffe. 2017. Making neural qa as simple as possible but not simpler. In *CoNLL*, pages 271–280.

J. Welbl, P. Stenetorp, and S. Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*.

Y. Zhang, H. Dai, K. Toraman, and L. Song. 2018. KGˆ2: Learning to Reason Science Exam Questions with Contextual Knowledge Graph Embeddings. In *arXiv*.

Evaluating Theory of Mind in Question Answering

Aida Nematzadeh
DeepMind
nematzadeh@google.com

Kaylee Burns
UC Berkeley
kayleeburns@berkeley.edu

Erin Grant
UC Berkeley
eringrant@berkeley.edu

Alison Gopnik
UC Berkeley
gopnik@berkeley.edu

Thomas L. Griffiths
Princeton University
tomg@princeton.edu

Abstract

We propose a new dataset for evaluating question answering models with respect to their capacity to reason about beliefs. Our tasks are inspired by theory-of-mind experiments that examine whether children are able to reason about the beliefs of others, in particular when those beliefs differ from reality. We evaluate a number of recent neural models with memory augmentation. We find that all fail on our tasks, which require keeping track of inconsistent states of the world; moreover, the models' accuracy decreases notably when random sentences are introduced to the tasks at test.[1]

1 Reasoning About Beliefs

Possessing a capacity similar to human reasoning has been argued to be necessary for the success of artificial intelligence systems (*e.g.*, Levesque et al., 2011). One well-studied domain that requires reasoning is question answering, where simply memorizing and looking up information is often not enough to correctly answer a question. For example, given the very simple scenario in Table 1, searching for the word "Mary" and returning a nearby word is not a correct strategy; instead, a model needs to recognize that Mary is currently at the second location (office and not the bathroom).

Recent research has focused on developing neural models that succeed in such scenarios (Sukhbaatar et al., 2015; Henaff et al., 2017). As a benchmark to evaluate these models, Weston et al. (2016) released a dataset – Facebook bAbi – that provides a set of toy tasks, each examining a specific type of reasoning. For example, the scenario in Table 1 evaluates the capacity to reason using a single supporting fact. However, the bAbi tasks are already too simple for the current models. Only a few years after their release, existing models fail at only one or two (out of 20) tasks (Rae et al., 2016; Santoro et al., 2017). Moreover, all except two of the reasoning tasks in this dataset only require transitive inference (Lee et al., 2016).

Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A: office

Table 1: A task from the bAbi dataset (Weston et al., 2016).

People reason not just about their own observations and beliefs but also about others' mental states (such as beliefs and intentions). The capacity to recognize that others can have mental states different than one's own – *theory of mind* – marks an important milestone in the development of children and has been extensively studied by psychologists (for a review, see Flavell, 2004). Artificial intelligence (AI) systems will also require a similar reasoning capacity about mental states as they are expected to be able to interact with people (*e.g.*, Chandrasekaran et al., 2017; Grant et al., 2017; Rabinowitz et al., 2018).

However, the bAbi dataset does not include tasks that evaluate a model's ability to reason about beliefs. Grant et al. (2017) created a bAbi-style dataset inspired by an influential experiment on the theory of mind called the Sally-Anne task (*e.g.* Baron-Cohen et al., 1985). Their goal was to examine whether the end-to-end memory network (Sukhbaatar et al., 2015) can answer questions such as "where does Sally think the milk is?" in situations that Sally's belief about the location of milk does not match the reality. For example, Sally thinks that the milk is in the fridge but the milk is actually on the table.

The dataset of Grant et al. (2017) provides a first step in designing benchmarks to evaluate the mental-state reasoning capacity of question-answering models, but it is still limited in the types of reasoning it probes. For example, it

---

[1] Code to generate dataset and replicate results is available at `github.com/kayburns/tom-qa-dataset`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2392–2400
Brussels, Belgium, October 31 - November 4, 2018.

only considered first-order beliefs (*e.g.*, Sally's belief about the location of milk). People also reason about second-order (and higher-order) beliefs (*e.g.*, Anne's belief about Sally's belief about the location of the milk). More importantly, similarly to the bAbi dataset, success in each task is defined as correctly answering one question. This does not guarantee that a model has an understanding of the state of the world; in fact, even in developmental theory-of-mind experiments, children are asked a few questions (*e.g.*, "where is milk really?") to ensure that their correct answer reflects their understanding and is not simply due to chance.

In this paper, we address these shortcomings by designing a new dataset that enables us to evaluate a model's capacity to reason about different types of beliefs as well as whether it maintains a correct understanding of the world. To this end, we evaluate a number of different models that perform well on the bAbi tasks: the end-to-end memory network (Sukhbaatar et al., 2015), the multiple observer model (Grant et al., 2017), the recurrent entity network (Henaff et al., 2017), and Relation-Network (Santoro et al., 2017). We find that none of these models succeed at our tasks, suggesting that they are not able to keep track of inconsistent states of the world, in particular when someone's belief does not match the history or reality of a situation.

2 Theory of Mind Experiments

Behavioral research shows that children gradually develop a theory of mind (for a review, see Gopnik and Astington, 1988). At the age of two, most children have an understanding of others' desires and perceptions – if someone wants something, they will try to get it and if something is in their sight, they can see it. Children begin to understand others' beliefs around the age of three, but this understanding is still limited. For example, they might not be able to reason that someone's actions are a result of their beliefs. By the age of five, most children have a unified theory of mind and are able to represent and reason about others' desires, perceptions, and beliefs. Developmental psychologists have designed various experimental paradigms to examine to what extent children are able to reason about others' mental states. We use these experiments as guidelines for designing tasks to evaluate the reasoning capacity of question-answering models. We first explain these experiments.

2.1 The Sally-Anne Experiment

The Sally-Anne false-belief experiment, proposed by Baron-Cohen et al. (1985), examines children's ability to reason about others' *false beliefs*, *i.e.*, when someone's belief does not match the reality. In this experiment, the participants observe two agents, Sally and Anne, with their containers, a basket and a box. After putting a marble in her basket, Sally leaves the room (and is not able to observe the events anymore). After Sally's departure, Anne moves the marble to her box. Then, Sally returns to the room (see Figure 1). The participants are asked the following questions:

- "Where will Sally look for her marble?" *(belief question)*
- "Where is the marble really?" *(reality question)*
- "Where was the marble in the beginning?" *(memory question)*

The first question tests the participants' ability to reason about Sally's belief about the location of her marble. Interestingly, most children before the age of 3 answer this question incorrectly and say that Sally will look at the box (where the marble really is) instead of the basket (where Sally thinks the marble is). These children are not able to reason about Sally's belief which is different from the reality of the world. The reality and memory questions are used to confirm that children's correct answer to the belief question is not due to chance; but because they have a correct understanding of the state of world and others' beliefs.

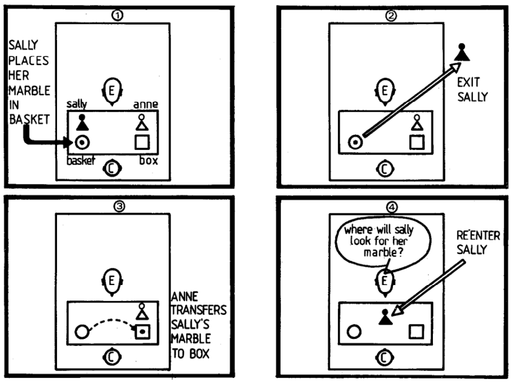

Figure 1: The Sally-Anne experiment setup from Baron-Cohen et al. (1985).

2.2 The Icecream Van Experiment

The Sally-Anne experiment examines the ability to reason about another person's belief or a *first-order belief*. People are also able to reason about beliefs about beliefs, for example, Anne thinks that Sally believes that the marble is in basket. Perner and Wimmer (1985) performed a set of experiments to examine children's reasoning capacity about such higher-order beliefs. In their set-up Mary and John together see an ice cream van in the park, and the icecream man tells them that he will be in the park until later in the afternoon. Mary leaves the park and goes home. A bit after she leaves, the icecream man decides to leave the park and tells John that he is going to the church. On his way to the church he runs into Mary and informs her that he will be selling icecreams close to the church all afternoon. The participants are then asked the following second-order question: "Where does John think Mary goes to get icecream?" Note that John does not know that Mary has been told about the new location of the icecream van; he has a *second-order false belief* about Mary's belief. The participants are also asked a few control questions (*e.g.*, "does Mary know that the van is in the church?") to ensure that they do not correctly answer the second-order question by chance. Perner and Wimmer (1985) found that 6- and 7-year old children are able to answer the second-order questions, suggesting that reasoning about higher-order beliefs (as compared to a first-order belief) is a harder cognitive task.

3 The Theory of Mind Task Dataset

Inspired by the theory-of-mind experiments explained in Section 2 and building on the work of Grant et al. (2017), we created a dataset based on three tasks designed to capture increasingly complex theory-of-mind reasoning: *true-*, *false-*, and *second-order false-belief* tasks. Examples of each task type are given in Figure 2. In the true-belief task, Sally observes the world and as a result she has a first-order *true belief* about the location of the milk – her belief matches reality. In the false-belief task, Sally's first-order belief differs from reality (*i.e.*, she has a *false belief*) because she was absent when the state of the world changed. In the second-order false-belief task, Sally observes the new location of the milk; thus, she has a *true belief* about the milk's location. However, Anne's belief about Sally's mental state does not match reality because Anne does not know that Sally has observed the change in the environment. As a result, Anne has a *false belief* about Sally's beliefs. These tasks are more challenging than the bAbI scenarios, because a model needs to learn whether each agent has a true or false belief about a given world state to succeed, where the world state now includes the mental states of each agent.

Note that we assume all containers are transparent in the underlying world; whenever an agent enters a location, they become aware of the objects true location. We made this decision to keep the tasks as structurally similar as possible. This prevents models from simply learning to produce a specific answer for a task type when a sentence like "Sally looks inside the pantry" is present in the story. The container-transparency property is consistent throughout all task-question pairs.

Question types. To examine the reasoning capacity of each model about beliefs and second-order beliefs, we employ four question types inspired by theory-of-mind experiments discussed in Section 2; see Table 2 for examples of these question types. These questions enable us to test whether a model can reason about first-order and second-order beliefs, and at the same time, knows the initial and current correct location of an object; thus, we can distinguish between when a model answers a question by chance and when it actually understands the entire state of the world.

Table 3 gives the answers for the 12 combinations of task type and question. Given a *true-belief* or *false-belief* task, the answers to the first-order and second-order questions are the same (*e.g.*, "pantry" in the *true-belief* condition and "fridge" in the *false-belief* condition for the tasks in Figure 2). However, they are different in the *second-order false belief* task because Anne has a false belief about Sally's belief.

Dataset variants. We use these tasks to generate two datasets: *ToM* and *ToM-easy*. The primary difference between these two datasets is that, in *ToM-easy*, each story has only one task, while *ToM* can have multiple tasks within a single story. Each dataset contains a training set with 10 000 examples with each of the 12 combinations of task and question types.

In *ToM*, the tasks are randomly grouped into sets of 5 to form stories, which is the same number used in the bAbI dataset. In the test set for *ToM*, each story contains 4 tasks, but there is only one question present at the end. Because questions that come closer to the beginning of a story have

True Belief	**False Belief**	**Second-order False Belief**
Anne entered the kitchen.	Anne entered the kitchen.	Anne entered the kitchen.
Sally entered the kitchen.	Sally entered the kitchen.	Sally entered the kitchen.
The milk is in the fridge.	The milk is in the fridge.	The milk is in the fridge.
Anne moved the milk to the pantry.	*Sally exited the kitchen.*	*Sally exited the kitchen.*
	Anne moved the milk to the pantry.	Anne moved the milk to the pantry.
		Anne exited the kitchen.
		Sally entered the kitchen.

Figure 2: An example story from each of the three task types.

Memory	Where was the milk at the beginning?
Reality	Where is the milk really?
First-order	Where will Sally look for the milk?
Second-order	Where does Anne think that Sally searches for the milk?

Table 2: Examples of the four question types.

	TB	**FB**	**SOFB**
Memory	first	first	first
Reality	second	second	second
First-order	second	first	second
Second-order	second	first	first

Table 3: The correct answer to each question for *true-belief* (TB), *false-belief* (FB), and *second-order false-belief* (SOFB) tasks. Here, “first” and “second” are the initial and actual locations of the object of interest, respectively (*e.g.*, fridge and pantry in Figure 2).

fewer distracting sentences (*i.e.*, potential answer words) that may confound a model, they are easier to answer. We found that this testing procedure gave us a more precise understanding of the performance of the model by separating the difficulty of a question due to its position in a story from the inherent difficulty of the question itself.

Generating the data. Each reasoning task in Weston et al. (2016) can be formalized with a grammar. The training and test data are then the derivations of this grammar. We refer to each derivation as a story (*e.g.*, Figure 2). We follow Grant et al. (2017) in writing grammars for our new tasks. In particular, all the task grammars consist of a set of *entities* (people and objects in the stories) and *predicates* that take entities as subject or object. The grammars also specify the *properties* of entities – which predicates take them as subjects or objects. A predicate can include *actions* that are ways an agent interact with the world (*e.g.*, *place, move, enter, exit*) and *beliefs* that are mental state terms (*e.g.*, *believe, think*). As an example, *Sally* with the property *is agent* can perform the action *displace* on *apple* with the property *is object*. Similar to the previous work, we use a restricted set of *action* and *belief* predicates.

4 The Models

We briefly describe the models that we evaluate in this paper. We chose these models based on the novelty in their architecture or their near state-of-the-art results in the bAbi tasks. More specifically, given 10k examples and joint-training on the bAbi tasks, the best end-to-end memory network (Sukhbaatar et al., 2015) and relation network (Santoro et al., 2017) fail at 6 and 2 tasks, respectively. Given the same training dataset and per-task training, the recurrent entity network succeed at all tasks (but the authors do not report the results of joint-training). Recall that the bAbi tasks are structured as a set of sentences followed by a question about them (*e.g.*, a story from Figure 2 followed by a question from Table 2).

The End-to-End Memory Network. Sukhbaatar et al. (2015) proposed a neural memory-augmented model, the end-to-end memory network (MemN2N), that extends the memory network architecture (Weston et al., 2014). Similarly to its predecessor, MemN2N has a memory component in which sentences are embedded and an attention function that weights the embedded sentences based on their similarity to a given question. The MemN2N model introduces multiple layers of memory (hops) by stacking memory components such that the question embedding at layer $k+1$ is the sum of the output and question embedding of layer k.

The Multiple Observer Model. To perform well on the *false-belief* and *second-order false-belief* conditions, a model needs to identify that agents have experienced different events, and, as a result, have differing knowledge about the state of the world. Although having multiple layers of memory in the MemN2N model enables it to combine attention to different memory slots (*i.e.*, embedded sentences) at each layer, the model does not have access to each agent’s unique perspective. For example, the model is not explicitly told that Sally does not observe the change of the location of the milk in the *false-belief* condition. To address this, Grant et al. (2017) propose the Multiple Observer model that integrates MemN2N with individual memory modules for each agent in the story. An agent's memory only receives the sentences for which the agent is present and observes the world. Their model has an additional attention function that weighs the memory modules based on their relevance to the question. The model is expected to learn to attend to Sally's memory module if the question is about her belief about a state of the world.

The Recurrent Entity Network. Henaff et al. (2017) propose a memory-augmented architecture, EntNet, with two interesting properties; first, their model is a recurrent neural network and thus can capture the sequential nature of the events in a story. Second, instead of keeping a whole sentence embedding in a memory slot, their model can learn the important entities of a story (*e.g.*, a person) and their properties (*e.g.*, location) through a set of gated recurrent units and two weight matrices.

The Relation Network. Santoro et al. (2017) propose a neural model for relational reasoning. Their model consider the possibility of a relation among each two possible pairs of objects. To model the bAbi tasks, they consider each pair of sentences together with the question as inputs to their relation network.

5 Experimental Results

Experiment set-up We train all models jointly over all task types without noise, but evaluate them independently on different task and question pairs. We choose the best-performing models by selecting hyperparameters on the validation set. Similarly to Sukhbaatar et al. (2015), we consider a model successful only when its accuracy exceeds 95% across the entire task suite.

MemN2N and Multiple Observer Models. We first examine how each model performs across a range of parameter and initialization values. MemN2N models are very sensitive to the network initialization and for each set of parameters, the best result out of 10 runs is reported (Sukhbaatar et al., 2015). We first visualize the accuracy of all runs as a box plot to identify how sensitive each model is to random initialization (of parameters and internal states) and thus difficult to train. We also report the results for the best run in each experiment. We use a memory size of 50, the same as experiments of Sukhbaatar et al. (2015), to ensure that the memory contains all sentences of a given story.

EntNet. We report results averaged over 3 initializations because we observed little randomness due to initialization. We selected the learning rate on a held out validation set separately for *ToM-easy* and *ToM*; all otehr the same hyperparameters as Henaff et al. (2017): 20 memory slots and an embedding size of 100 We trained until the training error hit zero, which occurred around 50 epochs for both datasets. .

RelNet. We report results using a single seed because we saw little randomness due to initialization; this is in accordance with the authors' findings (Santoro et al., 2017). We selected model hyperparameters on a held-out validation set separately for each of the *ToM* and *ToM-easy* datasets.

5.1 Overall Performance on *ToM-easy*

We expect the models perform well on this dataset, given that there is only one task in the memory during both training and test and as a result unrelated events do not interfere with a model's reasoning in this condition.

Despite the large variance in accuracy across runs, the MemN2N models often succeed at the memory, reality, and second-order questions (Figure 3a). Note that the median accuracy (the dark blue line in box) is close to 1 for these questions. However, the model often fails (median accuracy around 0.5) given the first-order question ("where will Sally look for the milk") and the *false-belief* and *second-order false-belief* tasks. This pattern is different from the empirical findings on people; the second-order question is harder than the first order one for people. We observe that the performance of the Multiple Observer model is similar to the MemNet for most question-task pairs (expect one) but there is less variation in the accuracy values. Interestingly, the median accuracy is close to one for the first-order question and the *second-order false-belief* task but the Multiple Observer model still performs poorly for this question on the *false-belief* task.

Why is the first-order question harder for the MemN2N model? To investigate this, we look more closely at our task-question pairs. As shown in Table 3, the answers to the first-order question are different for the *false-belief* and *second-order false-belief* tasks but are the same for the

Figure 3: Memory Network and Multiple Observer Model Performance Across Task and Question Types. Pink indicates that the answer to the question is the first container that contained the object in that task. Blue indicates that the answer is the last container that contained the object before the question was asked. Grey indicates that the answer was the first container that contained the object in the entire story which may or may not be the same as the pink.

(a) Memory Network with memory size 50 evaluated on the *ToM-easy* dataset.

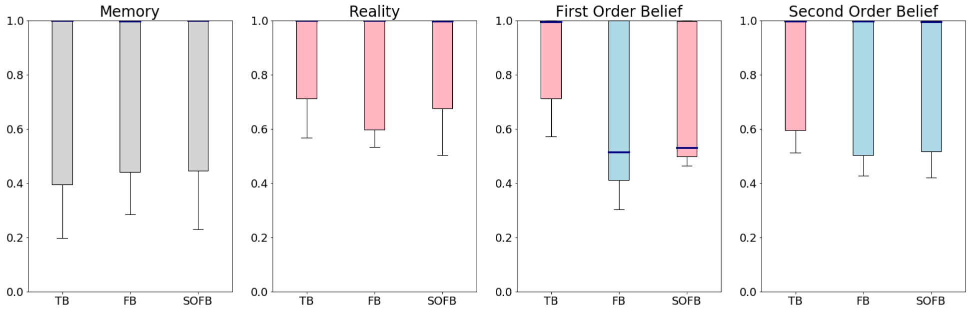

(b) Multiple Observer Model with memory size 50 evaluated on the *ToM-easy* dataset.

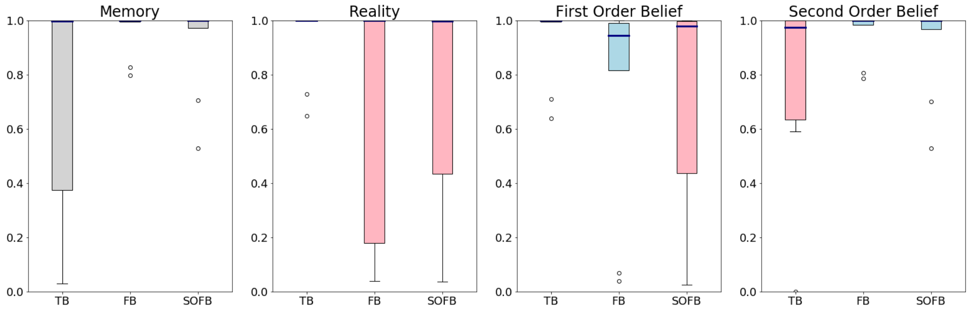

(c) Memory Network with memory size 50 evaluated on the *ToM* dataset.

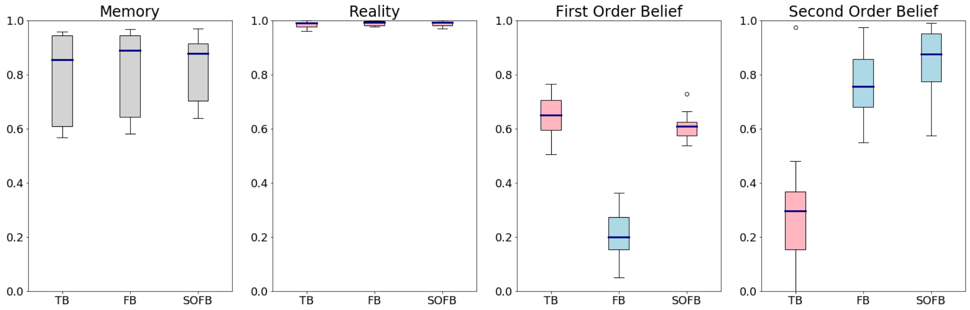

(d) Multiple Observer Model with memory size 50 evaluated on the *ToM* dataset.

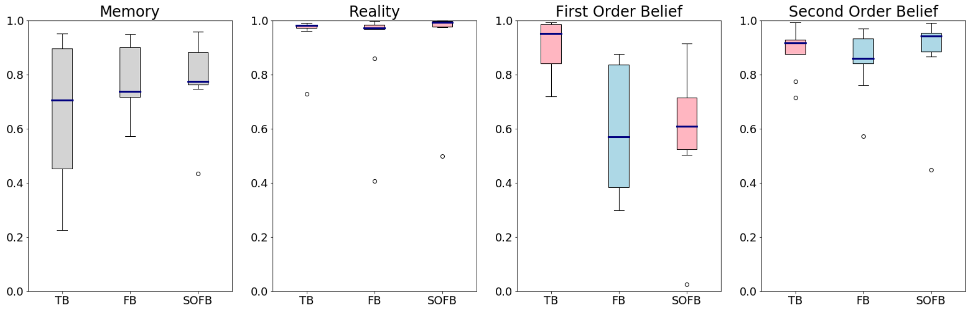

second-order one. We suspect that it is harder for the MemN2N model to learn two distinct answers for the same question given the the similarities of the two false-belief tasks. To test this hypothesis, we altered the data such that the answers to first-order question are (incorrectly) the same for both *false-belief* and *second-order false-belief* tasks (and also the second-order question). We observe that the median accuracy is close to 1 for all conditions suggesting that the model can learn the distinction between two of the tasks but not all three.

We observe that EntNet and RelNet models are not too sensitive to the initialization value, and thus just report the result on best-performing models. Both EntNet and RelNet best models succeed at the *ToM-easy* tasks; their mean error is 0.

5.2 Overall Performance on *ToM*

This dataset is more similar to the bAbi dataset in that, during both training and test, the memory contains a story with multiple tasks; as a result, it is harder for the model to identify the entities relevant to a given question.

As shown in Figure 3c, the MemN2N performs worse on all the questions with the exception of the reality question, where performance is slightly worse but has lower variance. (*cf.* the *ToM-easy* dataset). The first- and second-order questions are, overall, harder for the model. The performance of the Multiple Observer model is better for most of the questions (see Figure 3d), especially the second-order, but it is slightly worse for the memory question. This suggests that adding agent-specific memory modules is not enough to succeed on all the *ToM* tasks, and that the increased complexity of inference with multiple memory modules harms performance on another task.

We also look at the best-performing MemN2N and Multiple Observer models over all question-task pairs. These models are selected based on their performance on the validation dataset. We observe that none of these models succeed on the *ToM* tasks, but, interestingly, the Multiple Observer model performs better overall as compared to the original MemN2N model (see Table 4).

***ToM* and bAbi.** How are *ToM* and bAbi tasks similar? The combination of our *true-belief* task and the reality question is very similar to bAbi task 1 (single supporting fact). To correctly answer the reality question ("where is the milk really?", a model need to use a single fact from the story ("Anne moved the milk to the pantry."). The MemN2N model succeeds at both bAbi task 1 and the reality question given the *true-belief* task. However, the correct answer to the memory question ("where was the milk at the beginning?") for the *true-belief* task also requires a single fact ("the milk is in the fridge."). Interestingly, the error of MemN2N on the memory question is much higher than the reality question. The model (unlike people) cannot learn two representations (initial and current location) for an object. This result demonstrates the importance of representing alternative states of the world, whether it be past states of reality (*i.e.*, where the "milk" used to be) or mental states about the world (*i.e.*, where an agent believes the "milk" to be).

EntNet and RelNet. We also report results of two relevant memory-augmented neural network models on our tasks, EntNet (Henaff et al., 2017) and RelNet (Santoro et al., 2017), in Table 4. Again, because we did not observe sensitivity to initialization for these models, only average performance of their best-performing model is reported. We see that even though these models succeed on the *ToM-easy* dataset, they fail on the *ToM* tasks, suggesting that these models cannot simultaneously deal with inconsistent (*i.e.*, past, present, and mental) states of the world.

We further investigate which questions are the hardest for each best model; see Table 5. We observe that each of the MemN2N, Multiple Observer and RelNet models perform poorly on some combination of the first- and second-order questions, but are successful at answering the reality question. We hypothesize that this phenomenon occurs because the reality question is the most similar to the bAbi tasks. In addition, all models fail the memory question for each task type. While this is to be expected for EntNet due to its recurrent nature and therefore bias towards recency, it is surprising that the other models, which exhibit only a small positional bias, cannot correctly represent a past state of the world in order to answer the memory question correctly.

5.3 Experimenting with Noise

We examine to what extent each model's architecture is sensitive to the position of sentences in each story. We do so by adding a novel sentence at random locations in each story at test time. For any setting of the noise, p, there is a $p\%$ probability of a noise sentence occurring before each sentence in the story. Noise sentences cannot follow other noise sentences in the story. In this paper, we report results with $p = .1$. We observe that the accuracy of all best models decreases notably in the

Model	*ToM-easy*	*ToM-easy* (noised)	*ToM*	*ToM* (noised)
MemN2N (Sukhbaatar et al., 2015)	100.00%	90.28%	82.38%	77.00%
Multiple Observer (Grant et al., 2017)	100.00%	**95.81**%	91.11%	87.43%
EntNet (Henaff et al., 2017)	100.00%	94.61%	93.67%	**88.63**%
RelNet (Santoro et al., 2017)	100.00%	87.82%	**94.31**%	76.84%

Table 4: A comparison of model performance. All models succeed on the *ToM-easy* dataset without noise. Multiple Observer model performs best on *ToM-easy* with noise, RelNet performs best on *ToM*, and EntNet performs best on *ToM* with noise.

Model	**True-belief**	**False-belief**	**Second-order False-belief**
MemN2N (Sukhbaatar et al., 2015)	M (94.4); FOB (78.2); SOB (42.9)	M (94.3); FOB (17.3)	M (92.8); FOB (56.4)
Multiple Observer (Grant et al., 2017)	M (93.2)	M (90.5); FOB (56.4); SOB (92.5)	M (93.0); FOB (90.3); SOB (90.3)
EntNet (Henaff et al., 2017)	M (74.0)	M (76.1)	M (74.25)
RelNet (Santoro et al., 2017)	M (39.7%); FOB (71.37%)	M (78.5%)	M (34.9%); FOB (81.8%)

Table 5: Model accuracy on failed questions given the *ToM* task (without noise); M, R, FOB, and SOB are the memory, reality, first- and second-order questions, respectively. The number in the parentheses is the accuracy for that question on that task.

presence of noise (see Table 4). This result is particularly interesting as it shows that none of the models are able to use the semantics of the sentences in a story in their reasoning – they are all sensitive to the presence of distractor sentences.

Interestingly, the RelNet model is the best performer amongst the models we considered on the *ToM* dataset, yet it is also the most sensitive to noise. Moreover, the Multiple Observer model – with explicit memories for each agent – is the most robust to noise; it has the minimum decrease in accuracy between each dataset and its noised version.

5.4 Experimenting with Memory

In the experiments of Sukhbaatar et al. (2015) the memory size is fixed to 50, which is necessary to capture the entire story in memory (e.g. the answer to the memory question in *ToM* may rely on information at the beginning of a story). We observed that smaller memory sizes artificially improved the performance of the MemN2N and Multiple Observer model on *ToM* tasks. For example, using a memory size of 10, our best MemN2N model performance boosts on the hardest task of *ToM* (FB task with first order belief question) from 5.1% to 97.5% and on the easiest task from 98.3% to 100.0% (SOFB task with reality question). This result is not surprising because given a small memory size, *ToM* and *ToM-easy* are very similar tasks; the memory size of 10 allows for at most two full tasks in memory.

6 Related Work

Recent research has emphasized the importance of modeling and understanding people's mental states for AI systems. Eysenbach et al. (2016) created a dataset of scene-description pairs where each scene is a set of visual frames and some frames include people with mistaken beliefs.[2] The authors build a regression model for identifying a mistaken belief and the person who has such a belief in a given frame. Our work differs with theirs in that we are interested in understanding whether a model can reason about people's true and false beliefs to correctly answer questions as opposed to identifying mistaken beliefs.

Grant et al. (2017) studied whether the end-to-end memory network of Sukhbaatar et al. (2015) can pass a false-belief test – correctly answer where Sally would search for an object in false- and true-belief situations. They created a dataset inspired by the bAbi dataset to examine whether the model can reason about interaction of beliefs and actions in these situations – how actions cause beliefs and vice versa. They show that MemN2N fails at the false-belief test, and their extension of that model with separate memories for each agent and an observer outperforms MemN2N.

Rabinowitz et al. (2018) formulate the capacity to reason about others' beliefs as a meta-learning problem. They propose a neural network, ToMnet, that learns to predict the behavior of different agents given their past and current trajectory. Similarly to Grant et al. (2017), in addition to individual agents, they model an "observer" that has access to states and actions of all agents (though this information can be noisy and partial). Interestingly, their model successfully predicts an agent's behavior in a false-belief situation – the agent's behavior reflects its false-belief as opposed to the reality of the world.

Finally, Chandrasekaran et al. (2017) take a dif-

---

[2] For example, a scene where a person gets sick eating mushrooms is paired with the sentence "the couple mistakenly thinks it's ok to eat the mushrooms".

ferent approach by studying whether people can understand the "beliefs" of a visual-question answering system. More specifically, they examine whether the participants can predict when the model would fail in answering a question as well as if they can predict the model's answer. They find that even with a few examples, people get better at answering these questions.

7 Discussion

We propose a dataset for evaluating question-answering models. Our dataset – inspired by seminal theory-of-mind experiments in children – measures to what extent recently introduced neural models can reason about beliefs and states of the world that are potentially mututally inconsistent. We evaluate three of the recent neural question answering models (and an extension of one) on our tasks. We find that none of the models are able to succeed fully on a suite of tasks that requires keeping track of inconsistent beliefs or states of the world. These inconsistencies arise from differences between the past and the present, as well as the mental states of agents who may have false beliefs about the world or about the mental states of other agents.

The purpose of the dataset introduced in this work is not to test advanced language fluency; instead, consistency in the linguistic structure of the tasks allows us to isolate the performance of the models' reasoning capabilities. Even though the language is simple, the models struggle to achieve good performance. Furthermore, we note that the proposed dataset should be treated as a diagnostic tool and that good performance on similar toy tasks is not sufficient for reasoning capabilities.

References

Simon Baron-Cohen, Alan M Leslie, and Uta Frith. 1985. Does the autistic child have a theory of mind? *Cognition*, 21(1):37–46.

Arjun Chandrasekaran, Deshraj Yadav, Prithvijit Chattopadhyay, Viraj Prabhu, and Devi Parikh. 2017. It takes two to tango: Towards theory of AI's mind. *arXiv preprint arXiv:1704.00717*.

Benjamin Eysenbach, Carl Vondrick, and Antonio Torralba. 2016. Who is mistaken? *arXiv preprint arXiv:1612.01175*.

John H Flavell. 2004. Theory-of-mind development: Retrospect and prospect. *Merrill-Palmer Quarterly*, 50(3):274–290.

Alison Gopnik and Janet W Astington. 1988. Children's understanding of representational change and its relation to the understanding of false belief and the appearance-reality distinction. *Child develop.*, 59:26–37.

Erin Grant, Aida Nematzadeh, and Thomas L. Griffiths. 2017. How can memory-augmented neural networks pass a false-belief task. In *Proceedings of the 39th Annual Conference of the Cognitive Science Society*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *International Conference on Learning Representations*.

Moontae Lee, Xiaodong He, Wen tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. 2016. Reasoning in vector space: An exploratory study of question answering. In *International Conference on Learning Representations*.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47.

Josef Perner and Heinz Wimmer. 1985. "John thinks that Mary thinks that ..." Attribution of second-order beliefs by 5-to 10-year-old children. *Journal of experimental child psychology*, 39(3):437–471.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, S. M. Ali Eslami, and Matthew Botvinick. 2018. Machine theory of mind. In *Proceedings of the 35th International Conference on Machine Learning*.

Jack W. Rae, Jonathan J. Hunt, Tim Harley, Ivo Danihelka, Andrew W. Senior, Greg Wayne, Alex Graves, and Timothy P. Lillicrap. 2016. Scaling memory-augmented neural networks with sparse reads and writes. In *Proceedings of 30th Conference on Neural Information Processing Systems*.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy P. Lillicrap. 2017. A simple neural network module for relational reasoning. In *International Conference on Learning Representations*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Proceedings of 29th Conference on Neural Information Processing Systems*, pages 2440–2448.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *International Conference on Learning Representations*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory Networks. *ArXiv e-prints*.

A Unified Syntax-aware Framework for Semantic Role Labeling

Zuchao Li[1,2,*], **Shexia He**[1,2,*], **Jiaxun Cai**[1,2], **Zhuosheng Zhang**[1,2], **Hai Zhao**[1,2,†],
Gongshen Liu[3], **Linlin Li**[4], **Luo Si**[4]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
[3]School of Cyber Security, Shanghai Jiao Tong University, China
[4]Alibaba Group, Hangzhou, China
{charlee,heshexia,caijiaxun,zhangzs}@sjtu.edu.cn,
zhaohai@cs.sjtu.edu.cn, lgshen@sjtu.edu.cn,
{linyan.lll,luo.si}@alibaba-inc.com

Abstract

Semantic role labeling (SRL) aims to recognize the predicate-argument structure of a sentence. Syntactic information has been paid a great attention over the role of enhancing SRL. However, the latest advance shows that syntax would not be so important for SRL with the emerging much smaller gap between syntax-aware and syntax-agnostic SRL. To comprehensively explore the role of syntax for SRL task, we extend existing models and propose a unified framework to investigate more effective and more diverse ways of incorporating syntax into sequential neural networks. Exploring the effect of syntactic input quality on SRL performance, we confirm that high-quality syntactic parse could still effectively enhance syntactically-driven SRL. Using empirically optimized integration strategy, we even enlarge the gap between syntax-aware and syntax-agnostic SRL. Our framework achieves state-of-the-art results on CoNLL-2009 benchmarks both for English and Chinese, substantially outperforming all previous models.

1 Introduction

The purpose of semantic role labeling (SRL) is to derive the predicate-argument structure of each predicate in a sentence. A popular formalism to represent the semantic predicate-argument structure is based on dependencies, namely dependency SRL, which annotates the heads of arguments rather than phrasal arguments. Given a sentence (in Figure 1), SRL is generally decomposed into multiple subtasks in pipeline framework, consisting of predicate identification (*makes*), predicate disambiguation (*make.02*), argument identification (e.g., *Someone*) and argument classification (*Someone* is A0 for the predicate *makes*). SRL is beneficial to a wide range of natural language processing (NLP) tasks, including machine translation (Shi et al., 2016) and question answering (Berant et al., 2013; Yih et al., 2016).

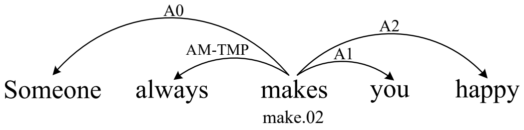

Figure 1: An example of dependency-based SRL.

Most traditional SRL methods rely heavily on feature templates that struggle to capture sufficient discriminative information, while neural models are capable of extracting features automatically. In particular, recent works (Zhou and Xu, 2015; He et al., 2017; Marcheggiani et al., 2017) propose syntax-agnostic models for SRL and achieve favorable results, which seems to be in conflict with the belief that syntactic information is an absolutely necessary prerequisite for high-performance SRL (Gildea and Palmer, 2002). Despite the success of these models, the main reasons for putting syntax aside are two-fold. First, it is still challenging to effectively incorporate syntactic information into neural SRL models, due to the sophisticated tree structure of syntactic relation. Second, the syntactic parsers are unreliable on account of the risk of erroneous syntactic input, which may lead to error propagation and an unsatisfactory SRL performance.

However, syntactic information is considered closely related to semantic relation and plays an essential role in SRL task (Punyakanok et al., 2008). Recently, Marcheggiani and Titov (2017)

---

[*] These authors made equal contribution.[†] Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. 61672343 and No. 61733011), Key Project of National Society Science Foundation of China (No. 15-ZDA041), The Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04) and the joint research project with Youtu Lab of Tencent.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2401–2411
Brussels, Belgium, October 31 - November 4, 2018.

proposed a syntactic graph convolutional networks (GCNs) based SRL model and further improved the SRL performance with relatively better syntactic parser as input. Since syntax can provide rich structure and information for SRL, we seek to effectively model complex syntactic tree structure for incorporating syntax into neural SRL.

In this paper, we present a general framework[1] for SRL, which enables us to integrate syntax into SRL in diverse ways. Following Marcheggiani and Titov (2017), we focus on argument labeling and formulate SRL as sequence labeling problem. However, we differ by (1) leveraging enhanced word representation, (2) applying recent advances in recurrent neural networks (RNNs), such as highway connections (Srivastava et al., 2015), (3) using deep encoder with residual connections (He et al., 2016), (4) further extending Syntax Aware Long Short-Term Memory (SA-LSTM) (Qian et al., 2017) for SRL, and (5) introducing the Tree-Structured Long Short-Term Memory (Tree-LSTM) (Tai et al., 2015) to model syntactic information for SRL.

In addition, as pointed out by He et al. (2017) for span SRL, the worse syntactic input will hurt performance if the syntactically-driven SRL model trusts syntactic information too much, and high-quality syntax can still make a large impact on SRL, which motivates us to investigate the effect of syntactic quality on dependency SRL. In summary, our major contributions are as follows:

- We propose a unified neural framework for dependency SRL to more effectively integrate syntactic information with multiple methods.
- Our SRL framework incorporated with syntax achieves the new state-of-the-art results on both English and Chinese CoNLL-2009 benchmarks.
- We explore the impact of different quality of syntactic input on SRL performance, showing that high quality syntactic parse may indeed improve syntax-aware SRL.

2 A Unified SRL Framework

In order to explore the effectiveness of the syntactic feature from various perspectives, we propose a unified neural framework that is capable of optionally accommodating various types of syntactic encoders for syntax-based SRL.

Since the CoNLL-2009 shared task (Hajič et al., 2009) have beforehand indicated the predicate positions, we need to identify and label all arguments for each predicate, which is a typical sequence tagging problem. In this work, we construct a general SRL framework for argument labeling. As shown in Figure 2, our SRL framework includes three main modules, (1) BiLSTM encoder that directly takes sequential inputs, (2) MLP with highway connections for softmax output layer, and (3) an optional syntactic encoder that receives the outputs of the BiLSTM encoder and then let its own outputs integrate with the BiLSTM outputs through residual connections.

Note that when the syntactic encoder is completely removed, MLP only takes inputs directly from the BiLSTM encoder, which let our framework become a syntax-agnostic labeler.

2.1 Sentence Encoder

Word representation Given a sentence and known predicate, we consider predicate-specific word representation, following previous work (Marcheggiani and Titov, 2017). Specifically, each word embedding representation e_i of input sentence is the concatenation of several features, a randomly initialized word embedding e_i^r, a pre-trained word embedding e_i^p, a randomly initialized lemma embedding e_i^l, a randomly initialized POS tag embedding e_i^{pos}, and a predicate-specific feature e_i^f, which is a binary flag set 0 or 1 indicating whether the current word is the given predicate.

To further enhance the word representation, we leverage an external embedding ELMo (Embeddings from Language Models) proposed by Peters et al. (2018). ELMo is obtained by deep bidirectional language model that takes characters as input, enriching subword information and contextual information, which has expressive representation power. Eventually, the resulting word representation is concatenated as $e_i = [e_i^r, e_i^p, e_i^l, e_i^{pos}, e_i^f, \text{ELMo}_i]$.

BiLSTM encoder We use bi-directional Long Short-term Memory neural network (BiLSTM) (Hochreiter and Schmidhuber, 1997) as the sentence encoder to model sequential inputs. Given an input sequence $(e_1, \ldots, e_n)$, the BiLSTM processes these embedding vectors sequentially from both directions to obtain two separated hidden states, $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$ respectively. By concatenating the two states, we get a contextual representation $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$, which will be taken by the next

---

[1] Our code is available here: `https://github.com/bcmi220/unified_syn_srl`.

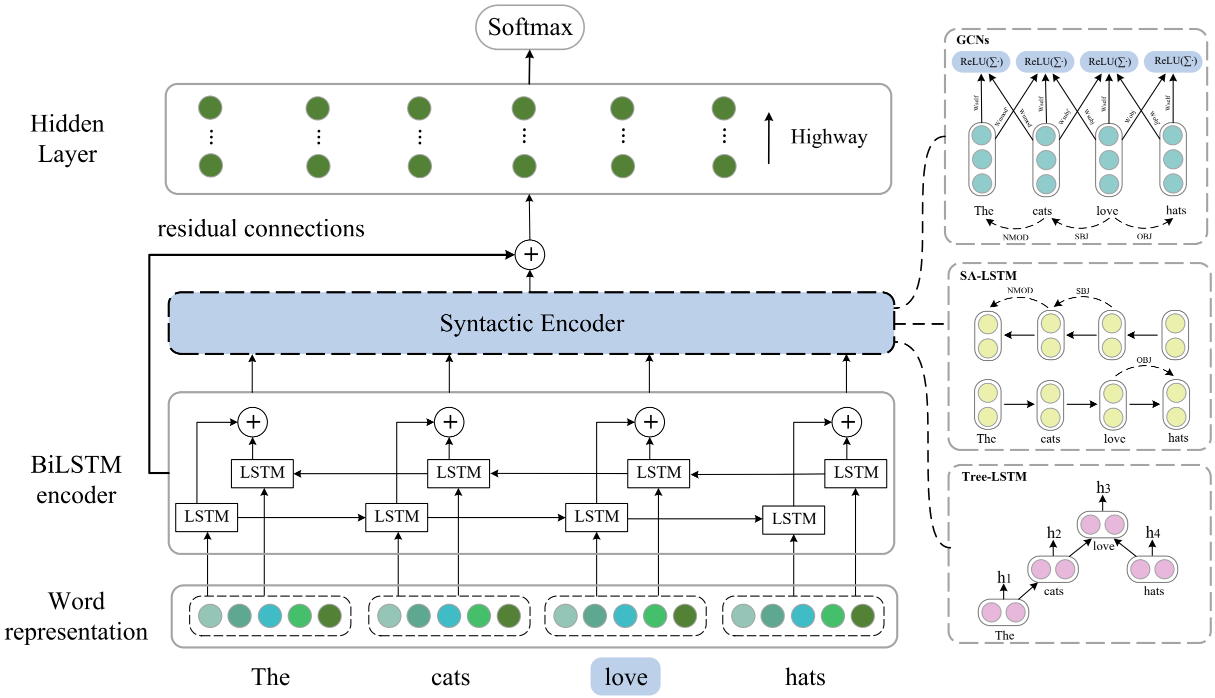

Figure 2: The unified syntax-based SRL framework

BiLSTM layer as input. In this work, we stack four layers of BiLSTM.

2.2 Role Labeler

We adopt a Multi-Layer Perceptron (MLP) with highway connections (Srivastava et al., 2015) on the top of our deep encoder, which takesthe concatenated representation as input. The MLP consists of 10 layers and we employ *ReLU* activations for the hidden layer. To get the final predicted semantic roles, we use a softmax layer over the outputs to maximize the likelihood of labels. The MLP part takes inputs from both the BiLSTM encoder and syntactic encoder, which are joint through a residual connection (He et al., 2016) as shown in Figure 2. It is worth noting that our deep encoder is different from the one of Marcheggiani and Titov (2017), which directly applies a softmax transformation over the syntactic representation and predicts the role label for each word. That is, their syntactic encoder outputs are directly taken as the input of hidden layer.

3 Syntactic Encoder

To integrate the syntactic information into sequential neural networks, we employ a syntactic encoder on top of the BiLSTM encoder.

Specifically, given a syntactic dependency tree T, for each node n_k in T, let $C(k)$ denote the syntactic children set of n_k, $H(k)$ denote the syntactic head of n_k, and $L(k,\cdot)$ be the dependency relation between node n_k and those have a direct arc from or to n_k. Then we formulate the syntactic encoder as a transformation f^{τ} over the node n_k, which may take some of $C(k)$, $H(k)$, or $L(k,\cdot)$ as input, and compute a syntactic representation v_k for node n_k, namely, $v_k = f^{\tau}(C(k), H(k), L(k,\cdot), x_k)$. When not otherwise specified, x_k denotes the input feature representation of n_k which may be either the word representation e_k or the output of BiLSTM h_k, σ denotes the logistic sigmoid function, and $\odot$ denotes the element-wise multiplication.

In practice, the transformation f^{τ} can be any syntax encoding method. In this paper, we will consider three types of syntactic encoders, syntactic graph convolutional network (Syntactic GCN) (in Section 3.1), syntax aware LSTM (SA-LSTM) (in Section 3.2), tree-structured LSTM (Tree-LSTM) (in Section 3.3). Then, we will provide a brief introduction in subsequent subsections.

3.1 Syntactic GCN

GCN (Kipf and Welling, 2017) is proposed to induce the representations of nodes in a graph based on the properties of their neighbors. Given its effectiveness, Marcheggiani and Titov (2017) introduce a generalized version for the SRL task, namely syntactic GCN, and shows that syntactic GCN is effective in incorporating syntactic information into neural models.

Syntactic GCN captures syntactic information flows in two directions, one from heads to dependents (along), the other from dependents to heads (opposite). Besides, it also models the information flows from a node to itself, namely, it assumes that a syntactic graph contains self-loop for each node. Thus, the syntactic GCN transformation of a node n_k is defined on its neighborhood $N(k) = C(k) \cup H(k) \cup \{n_k\}$. For each edge connects n_k and its neighbor n_j, we can compute a vector representation for it,

$$u_{k,j} = W^{dir(k,j)} x_j + b^{L(k,j)},$$

where $dir(k, j)$ denotes the direction type (along, opposite or self-loop) of the edge from n_k to n_j, $W^{dir(k,j)}$ is direction type specific parameter, $b^{L(k,j)}$ is label specific parameter. Considering that syntactic information from all the neighboring nodes may make different contribution to semantic role labeling, syntactic GCN introduces an additional edge-wise gating for each node pair (n_k, n_j) as

$$g_{k,j} = \sigma(W_g^{dir(k,j)} x_k + b_g^{L(k,j)}).$$

The syntactic representation v_k for a node n_k can be then computed as:

$$v_k = \mathit{ReLU}(\sum_{j \in N(k)} g_{k,j} \odot u_{k,j}).$$

3.2 SA-LSTM

SA-LSTM (Qian et al., 2017) is an extension of the standard BiLSTM architecture, which aims to simultaneously encode the syntactic and contextual information for a given word as shown in Figure 2. On one hand, the SA-LSTM calculates the hidden state in sequence timestep order like the standard LSTM,

$$\begin{aligned} i_g &= \sigma(W^{(i)} x_k + U^{(i)} h_{k-1} + b^{(i)}), \\ f_g &= \sigma(W^{(f)} x_k + U^{(f)} h_{k-1} + b^{(f)}), \\ o_g &= \sigma(W^{(o)} x_k + U^{(o)} h_{k-1} + b^{(o)}), \\ u &= f(W^{(u)} x_k + U^{(u)} h_{k-1} + b^{(u)}), \\ c_k &= i_g \odot u + f_g \odot c_{k-1}. \end{aligned}$$

On the other hand, it further incorporates the syntactic information into the representation of each word by introducing an additional gate,

$$\begin{aligned} s_g &= \sigma(W^{(s)} x_k + U^{(s)} h_{k-1} + b^{(s)}), \\ h_k &= o_g \odot f(c_k) + s_g \odot \tilde{h}_k. \end{aligned}$$

where $\tilde{h}_k = f(\sum_{t_j < t_k} \alpha_j \times h_j)$ is the weighted sum of all hidden state vectors h_j which come from previous node (word) n_j, the weight factor α_j is actually a trainable weight related to the dependency relation $L(k, \cdot)$ when there exists a directed edge from n_j to n_k.

Note that $\tilde{h}_k$ is always the hidden state vector of the syntactic head of n_k according to the definition of α_j. Since a word will be assigned a single syntactic head, such a strict constraint prevents the SA-LSTM from incorporating complex syntactic structures. Inspire by the idea of GCN, we relax the directed constraint of α_j, whenever there is an edge between n_j and n_k.

After the SA-LSTM transformation, the outputs of the SA-LSTM layer from both directions are concatenated and taken as the syntactic representation of each word n_k, i.e., $v_k = [\overrightarrow{h_k}, \overleftarrow{h_k}]$. Different from the syntactic GCN, SA-LSTM encoding both syntactic and contextual information in a single vector v_k.

3.3 Tree-LSTM

Tree-LSTM (Tai et al., 2015) can be considered as an extension of the standard LSTM, which aims to model the tree-structured topologies. At each timestep, it composes an input vector and the hidden states from arbitrarily many child units. Specifically, the main difference between Tree-LSTM unit and the standard one is that the memory cell updating and the calculation of gating vectors are depended on multiple child units. A Tree-LSTM unit can be connected to arbitrary number of child units and assigns a single forget gate for each child unit. This provides Tree-LSTM the flexibility to incorporate or drop the information from each child unit.

Given a syntactic tree, the Tree-LSTM transformation is defined on node n_k and its children set $C(k)$, which can be formulated as follows (Tai et al., 2015):

$$\tilde{h}_k = \sum_{j \in C(k)} h_k, \tag{1}$$

$$\begin{aligned} i_g &= \sigma(W^{(i)} x_k + U^{(i)} \tilde{h}_k + b^{(i)}), \\ f_g^{k,j} &= \sigma(W^{(f)} x_k + U^{(f)} h_j + b^{(f)}), \\ o_g &= \sigma(W^{(o)} x_k + U^{(o)} \tilde{h}_k + b^{(o)}), \\ u &= \tanh(W^{(u)} x_k + U^{(u)} \tilde{h}_k + b^{(u)}), \\ c_k &= i_g \odot u + \sum_{j \in C(k)} f_g^{k,j} \odot c_j, \\ h_k &= o_g \odot \tanh(c_k). \end{aligned} \tag{2}$$

where $j \in C(k)$, h_j is the hidden state of the j-th child node, c_k is the memory cell of the head node k, and h_k is the hidden state of node k. Note that in Eq.(2), a single forget gate $f_g^{k,j}$ is computed for each hidden state h_j.

However, the primitive form of Tree-LSTM does not take the dependency relations into consideration. Given the importance of dependency relations in SRL task, we further extend the Tree-LSTM by adding an additional gate r_g and reformulate the Eq. (1),

$$\begin{aligned} r_g^{k,j} &= \sigma(W^{(r)} x_k + U^{(r)} h_j + b^{L(k,j)}), \\ \tilde{h}_k &= \sum_{j \in C(k)} r_g^{k,j} \odot h_j. \end{aligned}$$

where $b^{L(k,j)}$ is a relation label specific bias term. After the Tree-LSTM transformation, the hidden state of each node in dependency tree is taken as its syntactic representation, i.e., $v_k = h_k$.

4 Experiments

We evaluate our models performance of syntactic GCN (henceforth Syn-GCN), SA-LSTM and Tree-LSTM on CoNLL-2009 datasets both for English and Chinese with standard training, development and test splits. For predicate disambiguation, we follow previous work (Marcheggiani and Titov, 2017), using the off-the-shelf disambiguator from Roth and Lapata (2016). For syntactic dependency tree, we parse the corpus with Biaffine Parser (Dozat and Manning, 2017).

4.1 Experimental Settings

In our experiments, the pre-trained word embeddings for English are 100-dimensional GloVe vectors (Pennington et al., 2014). For Chinese, we exploit Wikipedia documents to train the same dimensional Word2Vec embeddings (Mikolov et al., 2013). All other vectors are randomly initialized, the dimension of lemma embeddings is 100, and the dimension of POS tag embedding is 32. In addition, we use 300-dimensional ELMo embedding for English[2].

System	P	R	F_1
Local model			
Lei et al. (2015)	–	–	86.6
FitzGerald et al. (2015)	–	–	86.7
Roth and Lapata (2016)	88.1	85.3	86.7
Marcheggiani et al. (2017)	88.7	86.8	87.7
Marcheggiani and Titov (2017)	89.1	86.8	88.0
He et al. (2018)	89.7	89.3	89.5
Cai et al. (2018)	89.9	89.2	89.6
Ours (Syn-GCN)	90.3	**89.3**	**89.8**
Ours (SA-LSTM)	**90.8**	88.6	89.7
Ours (Tree-LSTM)	90.0	88.8	89.4
Global model			
Björkelund et al. (2010)	88.6	85.2	86.9
FitzGerald et al. (2015)	–	–	87.3
Roth and Lapata (2016)	**90.0**	**85.5**	**87.7**
Ensemble model			
FitzGerald et al. (2015)	–	–	87.7
Roth and Lapata (2016)	90.3	85.7	87.9
Marcheggiani and Titov (2017)	**90.5**	**87.7**	**89.1**

Table 1: Results on the English in-domain test set.

During training, we use the categorical cross-entropy as objective, with Adam optimizer (Kingma and Ba, 2015) the learning rate 0.001, and the batch size is set to 64. The BiLSTM encoder consists of 4-layer BiLSTM with 512-dimensional hidden units. We apply dropout for BiLSTM with a 90% keeping probability between time-steps and layers. We train models for a maximum of 20 epochs and obtain the nearly best model based on English development results.

4.2 Results

We compare our models of Syn-GCN, SA-LSTM and Tree-LSTM with previous approaches for dependency SRL on both English and Chinese. Noteworthily, our model is local (argument identification and classification decisions are conditionally independent) and single without reranking, which neither includes global inference nor com-

---

[2] For Chinese, we do not use pre-trained ELMo whose weights are only available for English.

System	P	R	F_1
Local model			
Zhao et al. (2009a)	80.4	75.2	77.7
Marcheggiani et al. (2017)	83.4	79.1	81.2
Marcheggiani and Titov (2017)	84.6	80.4	82.5
He et al. (2018)	84.2	81.5	82.8
Cai et al. (2018)	84.7	**84.0**	**84.3**
Ours (Syn-GCN)	84.8	81.2	83.0
Ours (SA-LSTM)	**85.2**	80.5	82.8
Ours (Tree-LSTM)	84.5	80.7	82.6
Global model			
Björkelund et al. (2009)	82.4	75.1	78.6
Roth and Lapata (2016)	**83.2**	**75.9**	**79.4**

Table 2: Results on the Chinese test set.

bines multiple models. The experimental results on the in-domain English and Chinese test sets are summarized in Tables 1 and 2, respectively.

For English, our models of Syn-GCN, SA-LSTM and Tree-LSTM overwhelmingly surpass most previously published single models, achieving state-of-the-art results of 89.8%, 89.7% and 89.4% in F_1 scores respectively. In comparison to ensemble models, our Syn-GCN even performs better than the previous model (Marcheggiani and Titov, 2017) with a margin of 0.7% F_1.

From Table 1, we also see that our Syn-GCN model provides the best recall and F_1 score, while our SA-LSTM model yields the competitive performance with higher precision at the expense of recall, which shows that SA-LSTM is better at classifying arguments. Overall, the Tree-LSTM gives slightly weaker performance, which may be attributed to tree-structured network topology. More specifically, Tree-LSTM only considers information from arbitrary child units so that each node lacks of the information from parent. However, our Syn-GCN and SA-LSTM combine bidirectional information, both head-to-dependent and dependent-to-head.

For Chinese (Table 2), even though we use the same parameters as for English, our models are still comparable with the best reported results.

Table 3 presents the results on English out-of-domain test set. Our models outperform the highest records achieved by He et al. (2018), with absolute improvements of 0.2-0.5% in F_1 scores. These favorable results on both in-domain and out-of-domain data demonstrate the effectiveness and robustness of our proposed unified framework.

System	P	R	F_1
Local model			
Lei et al. (2015)	–	–	75.6
FitzGerald et al. (2015)	–	–	75.2
Roth and Lapata (2016)	76.9	73.8	75.3
Marcheggiani et al. (2017)	79.4	76.2	77.7
Marcheggiani and Titov (2017)	78.5	75.9	77.2
He et al. (2018)	81.9	76.9	79.3
Cai et al. (2018)	79.8	78.3	79.0
Ours (Syn-GCN)	80.6	**79.0**	**79.8**
Ours (SA-LSTM)	**81.0**	78.2	79.6
Ours (Tree-LSTM)	80.4	78.7	79.5
Global model			
Björkelund et al. (2010)	77.9	73.6	75.7
FitzGerald et al. (2015)	–	–	75.2
Roth and Lapata (2016)	**78.6**	**73.8**	**76.1**
Ensemble model			
FitzGerald et al. (2015)	–	–	75.5
Roth and Lapata (2016)	79.7	73.6	76.5
Marcheggiani and Titov (2017)	**80.8**	**77.1**	**78.9**

Table 3: Results on the English out-of-domain test set.

Our system	P	R	F_1
Syn-GCN	89.2	87.6	88.4
w/o POS tag	88.5	87.5	88.0
w/o ELMo embedding	87.7	86.7	87.2

Table 4: Ablation on the English development set.

4.3 Ablation and Analysis

To investigate the contributions of word representation and deep encoder in our method, we conduct a series of ablation studies on the English development set, unless otherwise stated.

Effect of word representation In order to better understand how the enhanced word representation influences our model performance, we train our Syn-GCN model with different settings in input word embeddings. Table 4 shows results for our system when we remove POS tag and ELMo embedding respectively. Interestingly, the impact of POS tag embedding (about 0.4% F_1) is less compared to the previous works, which allows us to build an accuracy model even when the POS tag is unavailable. We also observe that effect of ELMo embedding is somewhat surprising (1.2% F_1 performance degradation). Experimental results indicate that a combination of these features could enhance the word representation, leading to SRL performance improvement.

System	syntax-agnostic	syntax-aware	Δ F$_1$
M&T (2017)	87.7	88.0	0.3
He et al. (2018)	88.7	89.5	0.8
Cai et al. (2018)	89.6	89.6	$\approx$0.0
Our model	88.7	89.8	1.1

Table 5: Comparison of our Syn-GCN model with (Marcheggiani and Titov, 2017), (He et al., 2018) and (Cai et al., 2018) on the English test set. Δ F$_1$ shows the absolute performance gap between syntax-agnostic and syntax-aware settings.

Effect of deep encoder Table 5 reports F$_1$ scores of our Syn-GCN model, Marcheggiani and Titov (2017), He et al. (2018) and Cai et al. (2018) on English test set in both syntax-agnostic and syntax-aware settings. The comparison shows that our framework is more effective for incorporating syntactic information by giving more performance improvement through introducing syntax over syntax-agnostic SRL than previous state-of-the-art systems did.

To further investigate the impact of deep encoder, we perform our Syn-GCN, SA-LSTM and Tree-LSTM models with another alternative configuration, using the same encoder as (Marcheggiani and Titov, 2017) (M&T encoder for short), which removes the residual connections from our framework. The corresponding results of our models are also summarized in Table 6 for comparison. Note that the first row is the results of our syntax-agnostic model. Surprisingly, we observe a dramatical performance decline of 1.2% F$_1$ for our Syn-GCN model with M&T encoder. A less significant performance loss for our SA-LSTM (-0.4%) and Tree-LSTM (-0.5%) models shows that the Syn-GCN is more sensitive to contextual information. Nevertheless, the overall results show that applying deep encoder could receive higher gains.

4.4 Syntactic Role

As mentioned before, syntactic parsers are unreliable due to the risk of erroneous syntactic input, especially on out-of-domain data. This section thus attempts to explore the impact of different quality of syntactic input on SRL performance. To this end, we further carry out experiments on English test data with different syntactic inputs based on our Syn-GCN model.

Our system	P	R	F$_1$
Baseline (syntax-agnostic)	89.5	87.9	88.7
Syn-GCN	90.3	89.3	**89.8**
SA-LSTM	90.8	88.6	89.7
Tree-LSTM	90.0	88.8	89.4
Syn-GCN (M&T encoder)	89.2	88.0	88.6
SA-LSTM (M&T encoder)	89.8	88.8	**89.3**
Tree-LSTM (M&T encoder)	90.0	87.8	88.9

Table 6: Comparison of models with deep encoder and M&T encoder (Marcheggiani and Titov, 2017) on the English test set.

Syntactic Input

Four types of syntactic inputs are used to explore the role of syntax in our unified framework, (1) the automatically predicted parse provided by CoNLL-2009 shared task, (2) the parsing results of the CoNLL-2009 data by state-of-the-art syntactic parser, the Biaffine Parser (used in our previous experiments), (3) corresponding results from another parser, the BIST Parser (Kiperwasser and Goldberg, 2016), which is also adopted by Marcheggiani and Titov (2017), (4) the gold syntax available from the official data set.

Evaluation Metric

It is worth noting that for SRL task, the standard evaluation metric is the semantic labeled F$_1$ score (Sem-F$_1$), and we use the labeled attachment score (LAS) to quantify the quality of syntactic input. In addition, the ratio between labeled F$_1$ score for semantic dependencies and the LAS for syntactic dependencies (Sem-F$_1$/LAS) proposed by CoNLL-2008 shared task[3] (Surdeanu et al., 2008), are also given for reference. To a certain extent, the ratio Sem-F$_1$/LAS could normalize the semantic score relative to syntactic parse, impartially estimating the true performance of SRL, independent of the performance of the input syntactic parser.

Comparison and Discussion

Table 7 presents the comprehensive results of our Syn-GCN model on the four syntactic inputs aforementioned of different quality together with previous SRL models. A number of observations can be made from these results. First, our model gives quite stable SRL performance no matter the syntactic input quality varies in a broad range, ob-

---

[3] CoNLL-2008 is an English-only task, while CoNLL-2009 extends to a multilingual one. Their main difference is that predicates have been pre-identified for the latter.

System	LAS	P	R	Sem-F_1	Sem-F_1/LAS
Zhao et al. (2009c) [SRL-only]	86.0	–	–	85.4	99.3
Zhao et al. (2009a) [Joint]	89.2	–	–	86.2	96.6
Björkelund et al. (2010)	89.8	87.1	84.5	85.8	95.6
Lei et al. (2015)	90.4	–	–	86.6	95.8
Roth and Lapata (2016)	89.8	88.1	85.3	86.7	96.5
Marcheggiani and Titov (2017)	90.34*	89.1	86.8	88.0	97.41
He et al. (2018) [CoNLL-2009 predicted]	86.0	89.7	89.3	89.5	104.0
He et al. (2018) [Gold syntax]	100	91.0	89.7	90.3	90.3
Our Syn-GCN (CoNLL-2009 predicted)	86.0	90.5	88.5	89.5	104.07
Our Syn-GCN (Biaffine Parser)	90.22	90.3	89.3	89.8	99.53
Our Syn-GCN (BIST Parser)	90.05	90.3	89.1	89.7	99.61
Our Syn-GCN (Gold syntax)	100.0	91.0	90.0	90.5	90.50

Table 7: Results on English test set, in terms of labeled attachment score for syntactic dependencies (LAS), semantic precision (P), semantic recall (R), semantic labeled F_1 score (Sem-F_1), the ratio Sem-F_1/LAS. All numbers are in percent. A superscript * indicates LAS results from our personal communication with the authors.

taining overall higher scores compared to previous state-of-the-arts. Second, It is interesting to note that the Sem-F_1/LAS score of our model becomes relatively smaller as the syntactic input becomes better. Though not so surprised, these results show that our SRL component is even relatively stronger. Third, when we adopt a syntactic parser with higher parsing accuracy, our SRL system will achieve a better performance. Notably, our model yields a Sem-F_1 of 90.5% taking gold syntax as input. It suggests that high-quality syntactic parse may indeed enhance SRL, which is consistent with the conclusion in (He et al., 2017).

5 Related Work

Semantic role labeling was pioneered by Gildea and Jurafsky (2002), also known as shallow semantic parsing. In early works of SRL, considerable attention has been paid to feature engineering (Pradhan et al., 2005; Zhao and Kit, 2008; Zhao et al., 2009a,b,c; Li et al., 2009; Björkelund et al., 2009; Zhao et al., 2013). Along with the the impressive success of deep neural networks (Zhang et al., 2016; Cai and Zhao, 2016; Qin et al., 2016; Wang et al., 2016b,a; Zhang et al., 2018; Li et al., 2018; Huang et al., 2018), a series of neural SRL systems have been proposed. For instance, Foland and Martin (2015) presented a semantic role labeler using convolutional and time-domain neural networks. FitzGerald et al. (2015) exploited neural network to jointly embed arguments and semantic roles, akin to the work (Lei et al., 2015), which induced a compact feature representation applying tensor-based approach.

Recently, people have attempted to build end-to-end systems for span SRL without syntactic input (Zhou and Xu, 2015; He et al., 2017; Tan et al., 2018). Similarly, Marcheggiani et al. (2017) also proposed a syntax-agnostic model for dependency SRL and obtained favorable results. Despite the success of syntax-agnostic models, there are several works focus on leveraging the advantages of syntax. Roth and Lapata (2016) employed dependency path embedding to model syntactic information and exhibited a notable success. Marcheggiani and Titov (2017) leveraged the graph convolutional network to incorporate syntax into a neural SRL model. Qian et al. (2017) proposed SA-LSTM to model the whole tree structure of dependency relation in an architecture engineering way.

Besides, syntax encoding has also successfully promoted other NLP tasks. Tree-LSTM (Tai et al., 2015) is a variant of the standard LSTM that can encode a dependency tree with arbitrary branching factors, which has shown effectiveness on semantic relatedness and the sentiment classification tasks. In this work, we extend the Tree-LSTM with a relation specific gate and employ it to recursively encode the syntactic dependency tree for SRL. RCNN (Zhu et al., 2015) is an extension of the recursive neural network (Socher et al., 2010) which has been popularly used to encode trees with fixed branching factors. The RCNN is able to encode a tree structure with arbitrary number of factors and is useful in a re-ranking model for dependency parsing (Zhu et al., 2015).

In our experiments, we simplify and reformulate the RCNN model. However, the simplified model performs poorly on the development and the test sets. The reason might be that the RCNN model with a single global composition parameter is too simple to cover all types of syntactic relation in a dependency tree. Because of the poor performance of the modified RCNN, we do not include it in this work. Considering there might be other approach to incorporate the recursive network in SRL model, we leave it as our future work and just provide a brief discussion here.

In this work, we extend existing methods and introduce Tree-LSTM for incorporating syntax into SRL. Rather than proposing completely new model, we synthesize these techniques and present a unified framework to take genuine superiority of syntactic information.

6 Conclusion

This paper presents a unified neural framework for dependency-based SRL, effectively incorporating syntactic information by directly modeling syntax based on syntactic parse tree. Rather than proposing completely new model, we extend existing models and apply tree-structured LSTM for SRL. Our approach significantly outperforms all previous models, achieving state-of-the-art results on the CoNLL-2009 benchmarks for both English and Chinese.

Our experiments specially show that giving an enlarged performance gap from syntax-agnostic to syntax-aware setting, SRL can be further promoted with the help of deep enhanced representation and effective methods of integrating syntax. Furthermore, we explore the impact of the quality of syntactic input. The relevant results indicate that high-quality syntactic parse is more favorable to semantic role labeling.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1533–1544, Seattle, Washington, USA.

Anders Björkelund, Bohnet Bernd, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 33–36, Beijing, China.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 43–48, Boulder, Colorado.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 409–420.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 2753–2765.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Nicholas FitzGerald, Oscar Tckstrm, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 960–970.

William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Joint Conference on Lexical and Computational Semantics*, pages 279–288.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 239–246, Philadelphia, Pennsylvania, USA.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 473–483, Vancouver, Canada.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2061–2071.

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

Yafang Huang, Zuchao Li, Zhuosheng Zhang, and Hai Zhao. 2018. Moon ime: neural-based Chinese pinyin aided input method with customizable association. *Proceedings of ACL 2018, System Demonstrations*, pages 140–145.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*, pages 1150–1160.

Junhui Li, Guodong Zhou, Hai Zhao, Qiaoming Zhu, and Peide Qian. 2009. Improving nominal srl in Chinese language with verbal srl information and automatic predicate recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1280–1288.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1506–1515, Copenhagen, Denmark.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

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

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*, New Orleans, Louisiana.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 581–588, Ann Arbor, Michigan.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Feng Qian, Lei Sha, Baobao Chang, Lu Chen Liu, and Ming Zhang. 2017. Syntax aware LSTM model for semantic role labeling. In *The Workshop on Structured Prediction for Natural Language Processing*, pages 27–32.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. Implicit discourse relation recognition with context-aware character-enhanced embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1914–1924.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1192–1202, Berlin, Germany.

Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun, and Houfeng Wang. 2016. Knowledge-based semantic embedding for machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2245–2254, Berlin, Germany.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1–9.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*, pages 159–177, Manchester, England.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1556–1566, Beijing, China.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2016a. Connecting phrase based statistical machine translation adaptation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3135–3145.

Rui Wang, Hai Zhao, Sabine Ploux, Bao-Liang Lu, and Masao Utiyama. 2016b. A bilingual graph-based semantic model for statistical machine translation. In *IJCAI*, pages 2950–2956.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 201–206, Berlin, Germany.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1382–1392.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2018. Subword-augmented embedding for cloze reading comprehension. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1802–1814.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*, pages 61–66, Boulder, Colorado.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009b. Semantic dependency parsing of nombank and propbank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 30–39.

Hai Zhao, Wenliang Chen, and Guodong Zhou. 2009c. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*, pages 55–60, Boulder, Colorado.

Hai Zhao and Chunyu Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL)*, pages 203–207.

Hai Zhao, Xiaotian Zhang, and Chunyu Kit. 2013. Integrative semantic dependency parsing via efficient large-scale feature selection. *Journal of Artificial Intelligence Research*, 46:203–233.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1127–1137, Beijing, China.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1159–1168.

Semantics as a Foreign Language

Gabriel Stanovsky*[2,3] and Ido Dagan[1]

[1]Bar-Ilan University Computer Science Department, Ramat Gan, Israel
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA
[3]Allen Institute for Artificial Intelligence, Seattle, WA
gabis@cs.washington.edu
dagan@cs.biu.ac.il

Abstract

We propose a novel approach to semantic dependency parsing (SDP) by casting the task as an instance of multi-lingual machine translation, where each semantic representation is a different foreign dialect. To that end, we first generalize syntactic linearization techniques to account for the richer semantic dependency graph structure. Following, we design a neural sequence-to-sequence framework which can effectively recover our graph linearizations, performing almost on-par with previous SDP state-of-the-art while requiring less parallel training annotations. Beyond SDP, our linearization technique opens the door to integration of graph-based semantic representations as features in neural models for downstream applications.

1 Introduction

Many sentence-level representations were developed with the goal of capturing the sentence's proposition structure and making it accessible for downstream applications (Montague, 1973; Carreras and Màrquez, 2005; Banarescu et al., 2013; Abend and Rappoport, 2013). See Abend and Rappoport (2017), for a recent survey.

While syntactic grammars (Marcus et al., 1993; Nivre, 2005) induce a rooted tree structure over the sentence by connecting verbal predicates to their arguments, these semantic representations often take the form of the more general labeled graph structure, and aim to capture a wider notion of propositions (e.g, nominalizations, adjectivals, or appositives). In particular, we will focus on the three graph-based semantic representations collected in the Broad-Coverage Semantic Dependency Parsing SemEval shared task (SDP) (Oepen et al., 2015): (1) DELPH-IN Bi-Lexical Dependencies (DM) (Flickinger, 2000),[1] (2) Enju Predicate-Argument Structures (PAS) (Miyao et al., 2014), and (3) Prague Semantic Dependencies (PSD) (Hajic et al., 2012). These annotations have garnered recent attention (e.g., (Buys and Blunsom, 2017; Peng et al., 2017a)), and were consistently annotated in parallel on over more than 30K sentences of the Wall Street Journal corpus (Charniak et al., 2000).

In this work we take a novel approach to graph parsing, casting sentence-level semantic parsing as a multilingual machine-translation task (MT). We deviate from current graph-parsing approaches to SDP (Peng et al., 2017a) by treating the different semantic formalisms as foreign target dialects, while having English a as a common source language (Section 3). Subsequently, we devise a neural MT sequence-to-sequence framework that is suited for the task.

In order to apply sequence-to-sequence models for structured prediction, a linearization function is required to interpret the model's sequential input and output. Initial work on structured prediction sequence-to-sequence modeling has focused on tree structures (Vinyals et al., 2015; Aharoni and Goldberg, 2017), as these are quite easy to linearize using the bracketed representation (as employed in the Penn TreeBank (Marcus et al., 1993)). Following, various efforts were made to port the attractiveness of sequence-to-sequence modeling to the more general graph structure of semantic representations, such as AMR or MRS (Peng et al., 2017b; Barzdins and Gosko, 2016; Konstas et al., 2017; Buys and Blunsom, 2017). However, to the best of our knowledge, all such current methods actually sidestep the challenge of graph linearization – they reduce the input graph to a tree using lossy heuristics, which are specifi-

---

*Work performed while at Bar-Ilan University.

[1] DM is automatically derived from Minimal Recursion Semantics (MRS) (Copestake et al., 1999).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2412–2421
Brussels, Belgium, October 31 - November 4, 2018.

cally tailored for their target representation.

In contrast, we design a novel deterministic and lossless linearization (Section 4), which is applicable to any graph with ordered nodes (e.g., sentence word order). To that end, we devise solutions for the various obstacles for linearizing a graph structure, such as reentrancies (or multiple heads), non-connected components, and non-projective relations. This lineariztion allows us to follow the spirit of Johnson et al. (2017) in training all source-target combinations in a multi-task approach (Section 5). These combinations include the three traditional text to semantic parsing tasks, as well as six additional inter-representation translation tasks, constituting of all binary combinations of the target representations (e.g., PSD to PAS, or DM to PSD).

Following, we design an encoder-decoder model which has two shared encoders, one for raw English sentences and another for linearized graphs, and a single global graph decoder. Interestingly, we show that training on the auxiliary inter-representation translation tasks greatly improves the performance on the original SDP tasks, without requiring any additional manual annotation effort (Section 6).

Our contributions are two-fold. First, we show that novel sequence-to-sequence models are able to effectively capture and recover general graph structures, making them a viable and easily extensible approach towards the SDP task. Second, beyond SDP, as the inclusion of syntactic linearization was shown beneficial in various tasks (Aharoni and Goldberg, 2017; Le et al., 2017) so does our approach prompt easy integration of graph-based representations as complementary semantic signal in various downstream applications.

2 Background

We begin this section by presenting the corpus we use to train and test our model (the SDP corpus) and the current state-of-the-art in predicting semantic dependencies. Then, we discuss previous work on sequence-to-sequence models for tree prediction, which this work extends to general graph structures. Finally, we briefly describe the multilingual translation approach, which we borrow and adapt to the semantic parsing task.

	DM	**PAS**	**PSD**
#Train sentences	35,657	35,657	35,657
#Test sentences	1,410	1,410	1,410
#Labels	59	42	91
%Trees	2.30	1.22	42.19
%Projective	2.91	1.64	41.92

Table 1: SDP corpus statistics. Numbers taken from Oepen et al. (2015).

2.1 Semantic Dependencies

In general, the development of most semantic formalisms was carried out by disjoint and independent efforts. However, the 2014 and 2015 SemEval shared tasks (Oepen et al., 2014, 2015) have culminated in the Semantic Dependency Parsing (SDP) resource, a consistent and large corpus (roughly 39K sentences), annotated in parallel with three well-established formalisms: DELPH-IN MRS-Derived Bi-Lexical Dependencies (DM) (Flickinger, 2000), Enju Predicate-Argument Structures (PAS) (Miyao et al., 2014), and Prague Semantic Dependencies (PSD) (Hajic et al., 2012). While varying in their labels and annotation guidelines, all three representations induce a graph structure, where each node corresponds to a single word in the sentence. See Table 1 for more details on this corpus, and Figure 1 for examples of the three SDP formalisms. SDP has enabled the application of machine learning models for the task. Peng et al. (2017a) have set the state-of-the-art results on all three tasks, using techniques inspired by graph-based dependency parsing models (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016; Kuncoro et al., 2016). Their best results were obtained by leveraging the fact that SDP was annotated on parallel texts. They reached 88% average labeled F1 score across the SDP representations on an in-domain test set, via joint prediction of the three representations using higher-order cross-representation features. The first row in Table 4 summarizes their performance for the three prediction tasks.

In this work we will take a different approach to structured prediction of the SDP corpus. We will design a novel sequence-to-sequence model, not necessitating parallel annotations, which are often unavailable for multi-task learning.

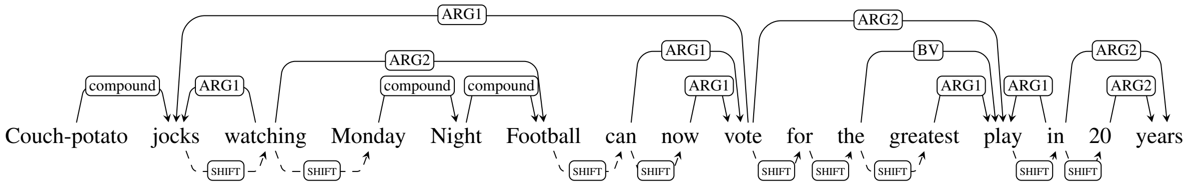

(a) DELPH-IN Minimal Recursion Semantics-derived bi-lexical dependencies (DM).

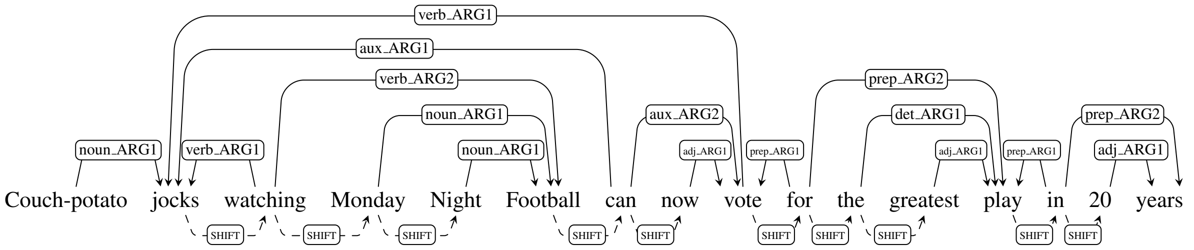

(b) Enju Predicate-Argument Structures (PAS).

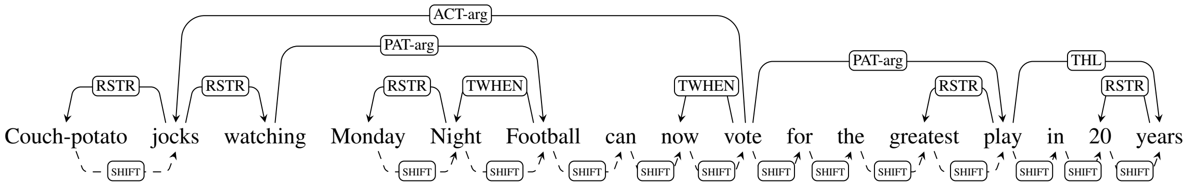

(c) Prague Semantic Dependencies (PSD).

Figure 1: Example of gold annotations for the three sentence-level representations in the SDP corpus (DM, PAS, and PSD) on the same sentence, which was slightly shortened for presentation. Arcs in each of the representations appear above the sentence. Our "SHIFT" edges, which appear dashed below it, were introduced in Section 4 to ensure that all nodes are reachable from the first word.

2.2 Structured Prediction using Sequence-to-Sequence Models

In contrast to the graph-parsing algorithms discussed in Section 2.1, a recent line of work has explored the usage of more general sequence-to-sequence models to perform structured prediction, focusing specifically on predicting tree structures. These approaches devise a task-specific linearization function which converts the structured representation to a sequential string, which is then used to train the recurrent neural network. During inference, the inverted linearization function is applied to the output to recover the desired structure.

Vinyals et al. (2015) showed that sequence-to-sequence phrase-based constituency parsing can be achieved using a tree depth-first search (DFS) traversal as a linearization function.[2] Following this work, several recent efforts have employed a similar DFS approach to AMR and MRS parsing (Barzdins and Gosko, 2016; Konstas et al., 2017; Peng et al., 2017b; Buys and Blunsom, 2017), after reducing AMR to trees by removing re-entrencies.

Several recent works have found syntactic linearization useful outside of neural parsers. For example, in neural machine translation, Aharoni and Goldberg (2017) showed that predicting target-side linearized syntactic trees can improve the quality and grammatically of the predicted translations. In Section 4, we show for the first time that the DFS approach is a viable linearization function also for semantic dependencies, by extending it to account for the challenges introduced by the richer graph structures in SDP.

2.3 Multi-lingual Machine Translation

Multi-Task Learning (MTL) is a modeling approach which shares and tunes the model parameters across several tasks. In some instances of MTL, a subset of the tasks may be defined as the "main tasks", while the other tasks are treated as auxiliaries which improve performance on the main tasks by contributing to their training signal. MTL had regained popularity in recent years thanks to its easy and wide-spread applicability in neural networks (Collobert et al., 2011; Sogaard

---

[2] DFS for rooted trees is equivalent to the bracketed notation of the Penn Treebank.

and Goldberg, 2016).

Perhaps most relevant to this work is Google's neural machine translation system by Johnson et al. (2017), which trained a single sequence-to-sequence model to translate between multiple languages. They introduced the usage of a special tag in the source sentence to specify the desired target language. For example, <*2es*> indicates that the model should translate the input sentence to Spanish.

In Section 4, we adapt the MTL strategy to train a single model for all SDP formalisms. We use a similar "to" and "from" tags to indicate source and desired target representations, and show that introducing auxiliary inter-task translations can improve performance on the main target tasks, namely parsing semantic representations for raw input text.

3 Task Definition

We define the task of **semantic translation**, as converting to, and between, different sentence-level semantic representations. Formally, a sentence-level semantic representation according to formalism R is a tuple, $\mathcal{M}_\mathcal{R} = (S, G)$, where $S = \{w_1, ..., w_n\}$ is a raw sentence, and $G = (V, E \mid V = \{v_1, ..., v_n\}, E \subseteq V^2)$ is a labeled graph whose vertices have a one-to-one correspondence with the words in S,[3] while its edges represent binary semantic relations, adhering to R's specifications.

Using these notations, our input is defined as a triplet $(source, target, \mathcal{M}_{source})$. Preceding the input semantic representation are identifiers for source and target representation schemes (e.g., "PAS", "DM" or "PSD"). The semantic translation task is then to produce $\mathcal{M}_{target}$. I.e., the sentence's representation under the $target$ formalism.

This definition is broad enough to encapsulate many sentence-level representations, and in this work we will use the three SDP representations, as well as an empty "RAW" representation (where $E(G) = \emptyset$ for all sentences) to allow for translations from raw input sentences. We note that future work may extend this framework with other graph-based sentence representations.

4 Graph Linearization

As discussed in Section 2, structured prediction in a sequence-to-sequence framework requires a linearization function, from the desired structure to a linear sequence, and vice versa.

Oftentimes, such linearization consists of node traversal along the edges of the input graph. While previous work have had certain structural constraints on their input (e.g., imposing tree or non-cyclic constructions), in this work, we construct a *lossless* function which allows us to feed the sequence-to-sequence network with a linearized *general* graph representation and expect a linearized graph in its output.

In this section, we describe our linearization traversal order, which generalizes the DFS traversal applied previously only for trees. We do this by converting an SDP graph such that all nodes are reachable from node v_1. We then outline the challenging aspects of graph properties (which do not exist in trees), show that they are prevalent in the SDP corpus, and describe our proposed solutions. To the best of our knowledge, this is the first work which tackles the task of general graph linearization.

While our linearization can be predicted with good accuracy (as we show in following sections), there is ample room to experiment with representational variations, which we start exploring in Section 6. Our conversion code is made publicly available,[4] allowing further experimentation with general graph linearization for SDP and other related tasks.

4.1 Traversing Graphs with Non-Connected Components

The DFS approach is an applicable linearization of trees since a recursive traversal, which starts at the root and explores all outgoing edges, is guaranteed to visit all of the graph's nodes. However, DFS linearization is not directly applicable to SDP, as its graphs often consist of several non-connected components.

For such graphs, there exists no starting node from which all of the nodes are reachable via DFS traversal, and certain nodes are bound to be left out of the traditional DFS encoding. For example, the words "can" and "greatest" in Figure 1a reside in different components, and therefore no single path

---

[3] The one-to-one node-to-word correspondence follows SDP's formulation, but can be relaxed to adjust for other graph structures.

[4] `https://github.com/gabrielStanovsky/semantics-as-foreign-language`

(which traverses along the graph's edge direction) will discover both of them.

To overcome this limitation, we make sure that all nodes are reachable from node v_1, corresponding to the first word in the sentence, from which we start our traversal. This is achieved by introducing an artificial `SHIFT` edge between any two consecutive nodes v_i, v_{i+1} for which there is no directed path already connecting them. Following, it is easy to see, by induction, that all nodes are reachable from v_1, as for every node v_i there exists a directed path $(v_1, v_2, ..., v_{i-1}, v_i)$. For example, revisit the previously mentioned "can" and "greatest" nodes in Figure 1a, which are connected using "SHIFT" edges.

4.2 Linearizing a DFS Graph Traversal

Intuitively, our linearization is a pre-order DFS, generalizing Vinyals et al. (2015)'s approach to syntactic linearization. We start from v_1 and explore all paths from it, in a depth-first manner. Once a path is exhausted, either by reaching a node with no outgoing edges[5] or by reaching an already visited node, we use special backtracking edges to form a path backwards "up" the graph, until we hit a node which still has unexplored outgoing edges.

Formally, our linearization of a given DFS traversal is composed of 3 types of elements (see Figure 2 for example):

First, a **Node reference** identifies a node in the graph, which in turn corresponds to word in the SDP formalism. We identify nodes using two tokens: (1) Their position in the sentence, relative to the previous node in the path (while the first position in the linearization is written in absolute terms, as "0"), and (2) Explicitly writing the word corresponding to the node.

For example, in Figure 2, traversing the *ARG1* edge from "easy" lands at "`ind/2` **`understand`**", whose outgoing *ARG2* edge arrives at "`ind/-4` **`success`**".

Second, an **Edge reference**, identifies an edge label. These are denoted by a single token, composed of 2 parts: (1) The edge's formalism (in our case, the SDP representation to which it pertains), and (2) The edge label. Traversing an edge (u, v) with label L will be encoded by placing the edge reference between the node references of u and v. For instance, in Figure 2, moving from node 0 to node 1 through the edge labeled "poss" is encoded with the following string: "`ind/0` **`Their`** `PAS/poss ind/1` **`success`**".

Finally, **Backtracking edges**, signify a step "backward" in the traversal. These are denoted with a single token, similarly to edge references, with the addition of a "BACK" suffix. For example, in Figure 2, we backtrack from the already visited node "understand" by writing: "*`PAS/ARG1/BACK`*".

This linearization can be deterministically and efficiently inverted back to the graph structure. This is done by building the graph while reading the linearization, adding to it nodes and edges when they first appear, and omitting possible node recurrences in the linearization (due to cycles or backtracking edges), such as "success", which appears twice in the Figure 2.

Redundancy in encoding We note that certain items in our proposed linearization are redundant. First, writing down the explicit word in the traversal is not necessary, as the positional index is sufficient to uniquely identify a node. Second, a single backtracking tag would have been enough to identify the specific edge which is currently being backtracked (e.g., `BACK` instead of `PAS/verb_ARG1/BACK`). The latter is similar to the redundancy in the syntactic linearization of Vinyals et al. (2015), who specify the type of closing bracket, e.g., `NP(...)NP` instead of `NP(...)`.

In Section 6 we show empirically that our model benefits from explicitly generating these redundancies during decoding.

4.3 DFS Traversal Order

A graph DFS traversal does not dictate an order in which to explore the different outgoing paths at each branching point. Consider, as a recurring example, the branching point at the word "vote" in Figure 1c, in which we need to choose an order amongst its four neighbors.

While syntactic linearization conveniently follows the ordering of the words in the sentence, Konstas et al. (2017) have noted that different child visiting linearization orders affect the performance of text generation from AMR. In particular, they found that following the order of annotation of a human expert worked best.

Intuitively, since different graph traversals affect the sequence of encoded nodes during train-

---

[5] Note that after introducing the artificial "SHIFT" edges, only v_n may have no outgoing edges.

DFS order type	Example (*vote*'s PSD neighbors)
Random permutation	*(play, for, jocks, now)**
Sentence order Neighbor's index in the sentence	*(jocks, now, for, play)*
Closest words Neighbor's absolute distance	*(now, for, play, jocks)*
Smaller-first # nodes reachable from the neighbor	*(now, play, for, jocks)*

Table 2: Different neighbor exploration orders. Under the name of each order type, we list the key by which we sort each node's neighbors. The "Example" column shows the corresponding ordering of "vote"'s neighbors in Figure 1c. *An example of one possible random permutation.

ing, the network will inevitably have to learn different weights and attention when presented with different orderings. Therefore, some traversal orderings may be easier to learn than others, leading to better (hopefully more semantic) abstractions.

To the best of our knowledge, the human annotation order is not available for the SDP annotations, and there is no clear a priori optimal ordering. We therefore experiment with several visiting orders, as described in Table 2. Notably, **Sentence order** is equivalent to the ordering used by Vinyals et al. (2015) for syntactic linearization, while **Closest words** orders child nodes from short to longer range-dependencies (commonly associated with syntactic versus semantic relations), and **Smaller first** is motivated by the easy-first approach (Goldberg and Elhadad, 2010), first encoding paths which are shorter (and easier to memorize), before longer, more complicated sequences.

In Section 6 we evaluate the effect of these variations on the SDP parsing task.

5 Model

We start by describing our model architecture, inspired by recent MT architectures, while allowing for different types of inputs, namely English sentences and linearized graphs. Following, we present our methods for training and testing, and specific hyper-parameter configuration and implementation details.

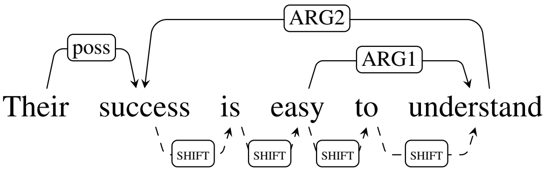

(a) Gold PAS representation from the SDP corpus. Original gold edges appear above the words, while our introduced edges appear below them.

```
ind/0 Their PAS/poss ind/1 success
SHIFT ind/1 is SHIFT ind/1 easy
PAS/ARG1 ind/2 understand PAS/ARG2
ind/-4 success PAS/ARG2/BACK ind/4
understand PAS/ARG1/BACK ind/-2 easy
SHIFT ind/1 to SHIFT ind/1 understand
```

(b) Our linearization scheme for the sentence in 2a. Each node is represented by its relative index and surface form. Backwards traversing edges (marked with *BACK*) appear in italics.

Figure 2: Example of gold PAS representation from the development partition of the SDP corpus (top), and our corresponding linearization (bottom).

5.1 Architecture

Our architecture, depicted in Figure 3, consists of a sequence-to-sequence model using a bi-LSTM encoder-decoder with attention on input and output tokens, similar to that used by Johnson et al. (2017) for multi-lingual MT. As described in Section 3, it is trained on 9 translation tasks in parallel. We split these into two groups, consisting of 3 primary tasks and 6 auxiliary tasks, as follows:

$$\begin{aligned}\texttt{PRIMARY} &= \{(\text{RAW}, tgt) \mid \\ &\qquad tgt \in (\text{DM, PAS, PSD})\} \\ \texttt{AUXILIARY} &= \{(src, tgt) \in \{\text{DM, PAS, PSD}\}^2 \mid \\ &\qquad src \neq tgt\}\end{aligned}$$

The `PRIMARY` tasks deal with converting raw sentences to linearized graph structures, which we can compare to previous published baselines and are therefore our main interest. Conversely, while the `AUXILIARY` tasks provide additional training signal to tune our model, they are also interesting from an analytic point-of-view, which we examine in depth in Section 6.

To allow the model to differentiate between the different tasks, we prefix each input sample with two tags (see example in Figure 3). First, similarly to Johnson et al. (2017), we add a tag indicating the desired target representation, e.g., `<to:DM>`. Second, In contrast to multi-lingual MT which omits the source language (to allow for

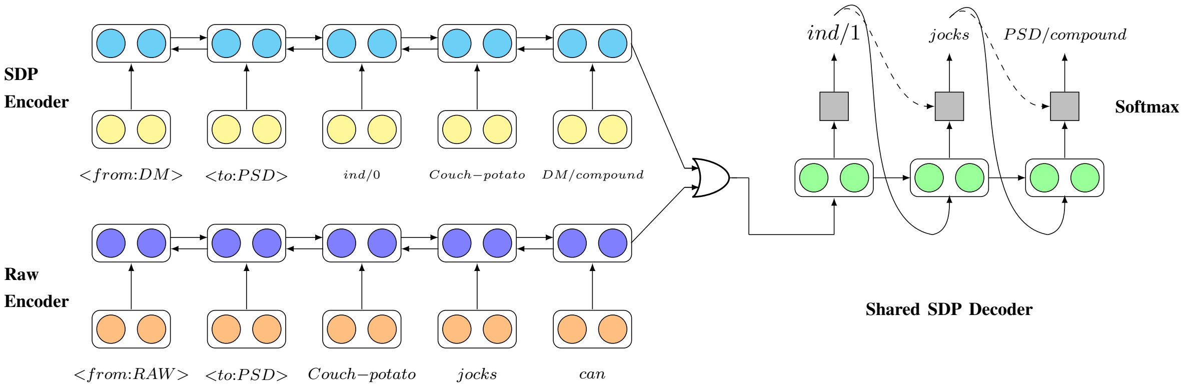

Figure 3: **Simplified sketch of our sequence-to-sequence architecture.** The figure depicts encoding and decoding of two input training samples, one from raw text to PSD (lower left), and the other from DM to PSD (top left). The OR gate denotes choosing only one sample to encode at each training step. While the two samples use different encoders, they share a single global SDP decoder (right) which outputs a the graph structure. As denoted by dashed edges, at every decode step we can deterministically interject and override the softmax probabilities for redundant elements, based on previous predictions. For simplicity sake, a small number of units is showed for encoders and decoder and the attention and deep encoder-decoder layers are omitted.

code switching), we explicitly denote the source representation, e.g., `<from:PSD>`. This addition further strengthens the correlation between inputs from the same representation.[6]

Further deviating from the current practice in MT, our architecture uses two encoders and a single decoder (while common MT regards the encoder-decoder as a single unit). The first shared encoder specializes in encoding raw text for all `PRIMARY` tasks, while a second encodes linearized graph structures for the `AUXILIARY` tasks. Both encoders are linked to a single decoder which converts their output representations to a linearized graph.

Intuitively, the two encoders correspond to the different nature of input to the `PRIMARY` tasks (an English sentence) versus that of the `AUXILIARY` tasks (a linearized graph), while a single decoder allows for a common linearized graphs output format. Since the decoder is trained across all 9 tasks, both encoders are optimized to arrive at similar latent representations which are geared towards graph prediction.

5.2 Training and Inference

The overall size of multi-task training data is $320,913$ samples. This constitutes a 9-fold increase over a single-model for SDP ($35,657$ sentences in the SDP corpus) and a 3-fold increase over a standard MTL approach to SDP (without the `AUXILIARY` tasks). During training, we penalize the model on all predicted elements, including the redundant elements discussed in Section 4.2. During inference, however, these redundancies may cause contradictions leading to incoherent sequences. Namely, a word may not conform to the previous word index, and a backtracking edge may point to a different relation. To overcome this we artificially increase the softmax probabilities (dashed edges in Figure 3) so that they reflect the DFS path decoded up until that point. Specifically, we override the predicted word according to the previous index, and backtrack "up" the corresponding edge.

5.3 Implementation Details

All of our hyper-parameters were tuned on a held out partition of 1000 sentences in the training set. In particular, we use 3 hidden layers for both of the encoders, and 2 hidden layers for the decoder. English word embeddings were fixed with 300-dimensional GloVe embeddings (Pennington et al., 2014), while the graph elements, which consist of a lexicon of roughly 400 tokens across three representations, were randomly initialized. We trained the model until convergence, roughly 20

---

[6] Moreover, "code-switching" between semantic representations is inherently undesired.

	DM	PAS	PSD	Avg.
Random	86.1	87.7	78.4	84.1
Sentence order	87.2	90.3	79.9	85.8
Closest words	87.5	89.8	79.7	85.8
Smaller-first	**87.9**	**90.9**	**80.3**	**86.2**

Table 3: Evaluation of different DFS orderings, in labeled F1 score, across the different tasks.

epochs, in about 12 hours on a GPU (NVIDIA GeForce GTX 1080 Ti), in batches of 50 sentences. All of these sentences belong to the same task, which is chosen at random before each batch.

Finally, our models were developed using the OpenNMT-py library (Klein et al., 2017), and are made available.[7]

6 Evaluation

We perform several evaluations, testing the impact of alternative configurations, including the different DFS traversal orders and MTL versus single-task approach, as well as our model's performance against current state-of-the-art on each of the `PRIMARY` tasks.

6.1 Results

The results of our different analyses are reported in Tables 3-6, as elaborated below. For all evaluations, we use the in-domain test partition of the SDP corpus, containing $1,410$ sentences. Following Peng et al. (2017a) we report performance using labeled F1 scores as well as average scores across representations. We compare the produced graphs, after applying the inverted linearization function, rather than comparing the DFS path directly, as there may be several DFS graph traversals encoding the same relations.

DFS order matters - Table 3 depicts our model's performance when linearizing the graphs according to the different traversal orders discussed and exemplified in Table 2. Overall, we find that the "smaller-first" approach performs best across all datasets, and that imposing one of our orders is always preferable over random permutations. Intuitively, the "smaller-first" approach presents shorter, and likely easier, paths first, thus minimizing the amount of error-propagation for following decoding steps. Due to its better performance, we will report only the smaller-first's performance in all following evaluations.

From English to SDP - Table 4 presents the performance of our complete model ("MTL PRIMARY+AUX") versus Peng et al. (2017a). On average, our model performs within 1% F1 point from the state-of-the art (outperforming it on the harder PSD task), despite using the more general sequence-to-sequence approach instead of a dedicated graph-parsing algorithm. In addition, an ablation study shows that multi-tasking the `PRIMARY` tasks is beneficial over a single task setting, which in turn is outperformed by the inclusion of the `AUXILIARY` tasks.

Simulating disjoint annotations - In contrast with SDP's complete overlap of annotated sentences, multi-task learning often deals with disjoint training data. To simulate such scenario, we retrained the models on a randomly selected set of 33% of the train sentences for each representation ($11,886$ sentences), such that the three representations overlap on only 10% ($3,565$ sentences). The results in Table 5 show that our approach is more resilient to the decrease in annotation overlap, outperforming the state-of-the-art model on the DM and PSD task, as well as on the average score. We hypothesize that this is in part thanks to our ability to use the inter-task translations, even when these exist only for part of the annotations.

6.2 Translating Between Representations

As a byproduct of training on the `AUXILIARY` tasks, our model can also be tested on translating *between* the different representations. This is done by presenting it with a linearized graph of one representation and asking it to translate it to another. To the best of our knowledge, this is the first work which tries to accomplish this.

We report the performance of all source-target combinations in Table 6. These evaluations provide several interesting comparisons between the representations: (1) For all representations, translating from any of the other two is easier than parsing from raw text, (2) The PAS and DM representations can be converted between them with high accuracy (95.7% and 96.1%, respectively). This can be due to their structural resemblance, noted in previous work (Peng et al., 2017a; Oepen et al., 2015), and (3) While PSD serves as a viable input for conversion to DM and PAS (92.1% F1 on

---

[7] `https://github.com/gabrielStanovsky/semantics-as-foreign-language`

	DM	**PAS**	**PSD**	**Avg.**
Peng et al. (2017a)	**90.4**	**92.7**	78.5	**87.2**
Single	70.1	73.6	63.6	69.1
MTL PRIMARY	82.4	87.2	71.4	80.3
MTL PRIMARY+AUX	87.9	90.9	**80.3**	86.2

Table 4: Evaluation of our model (labeled F1 score) versus the current state of the art. "Single" denotes training a different encoder-decoder for each task. "MTL PRIMARY" reports the performance of multi-task learning on only the `PRIMARY` tasks. "MTL PRIMARY+AUX" shows the performance of our full model, including MTL with the `AUXILIARY` tasks.

	DM	**PAS**	**PSD**	**Avg.**
Peng et al. (2017a)	86.8	**90.5**	77.3	84.9
MTL PRIMARY+AUX	**87.1**	89.6	**79.1**	**85.3**

Table 5: Performance (labeled F1 score) of our model versus the state of the art, when reducing the amount of overlap in the training data to 10%.

To \From	**DM**	**PAS**	**PSD**	**Avg.**
DM		96.1	92.4	94.3
PAS	95.7		91.7	93.7
PSD	89.5	87.6		88.6
Avg.	92.6	91.9	92.1	

Table 6: Performance (labeled F1 score) of inter-task translations. Each column depicts the performance converting from a specific source representation, while each row denotes the corresponding target representation.

average), it is relatively harder to convert either of them to PSD (88.6%). This might indicate that PSD subsumes some of the information in DM and PAS.

7 Conclusions and Future Work

We presented a novel sequence-to-sequence approach to the task of semantic dependency parsing, by casting the problem as multi-lingual machine translation. To that end, we introduced a DFS-based graph linearization function which generalizes several previous works on tree linearization. Following, we showed that our model, inspired by neural MT, benefits from the inter-task training signal, reaching performance almost on-par with current state of the art in several scenarios.

Future work can employ this linearization function within downstream applications, as was done with syntactic linearization, or extend this framework with other graph-based representations, such as universal dependencies (Nivre et al., 2016) or AMR (Banarescu et al., 2013).

Acknowledgements

This work was supported in part by grants from the MAGNET program of the Israeli Office of the Chief Scientist (OCS); the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1) and the Israel Science Foundation (grant No. 1157/16).

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). In *Proceedings of the 2013 conference of the Association for Computational Linguistics*.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *ACL*.

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *ACL*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking.

Guntis Barzdins and Didzis Gosko. 2016. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on amr parsing accuracy. In *SemEval@NAACL-HLT*.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *ACL*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CONLL*, pages 152–164.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. Bllip 1987-89 wsj corpus release 1. *Linguistic Data Consortium, Philadelphia*, 36.

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

Ann. Copestake, Dan Flickinger, and Ivan A. Sag. 1999. Minimal recursion semantics: An introduction.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Dan Flickinger. 2000. On building a more effcient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *HLT-NAACL*.

Jan Hajic, Eva Hajicová, Jarmila Panevová, Petr Sgall, Ondrej Bojar, Silvie Cinková, Eva Fucíková, Marie Mikulová, Petr Pajas, Jan Popelka, et al. 2012. Announcing prague czech-english dependency treebank 2.0. In *LREC*, pages 3153–3160.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL*, 5:339–351.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *TACL*, 4:313–327.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke S. Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *ACL*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. In *EMNLP*.

An Nguyen Le, Ander Martinez, Akifumi Yoshimoto, and Yuji Matsumoto. 2017. Improving sequence to sequence neural machine translation by utilizing syntactic dependency information. In *IJCNLP*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Yusuke Miyao, Stephan Oepen, and Daniel Zeman. 2014. In-house: An ensemble of pre-existing off-the-shelf parsers. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 335–340.

Richard Montague. 1973. The proper treatment of quantification in ordinary english. In *Approaches to natural language*, pages 221–242. Springer.

Joakim Nivre. 2005. Dependency grammar and dependency parsing.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresová. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *SemEval@NAACL-HLT*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017a. Deep multitask learning for semantic dependency parsing. In *Proceedings of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017b. Addressing the data sparsity issue in neural amr parsing. In *EACL*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Anders Sogaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *NIPS*.

An AMR Aligner Tuned by Transition-based Parser

Yijia Liu, Wanxiang Che*, Bo Zheng, Bing Qin, Ting Liu
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
`{yjliu,car,bzheng,qinb,tliu}@ir.hit.edu.cn`

Abstract

In this paper, we propose a new rich resource enhanced AMR aligner which produces multiple alignments and a new transition system for AMR parsing along with its oracle parser. Our aligner is further tuned by our oracle parser via picking the alignment that leads to the highest-scored achievable AMR graph. Experimental results show that our aligner outperforms the rule-based aligner in previous work by achieving higher alignment F1 score and consistently improving two open-sourced AMR parsers. Based on our aligner and transition system, we develop a transition-based AMR parser that parses a sentence into its AMR graph directly. An ensemble of our parsers with only words and POS tags as input leads to 68.4 Smatch F1 score, which outperforms the parser of Wang and Xue (2017).

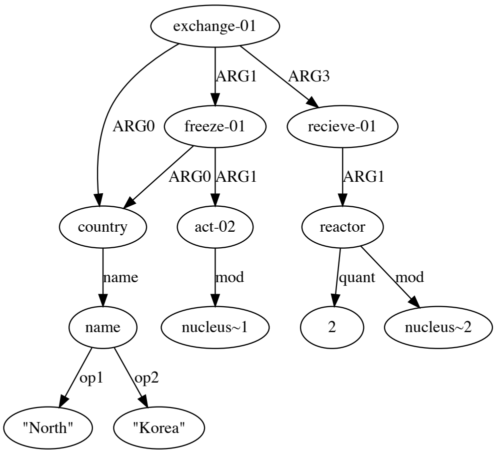

Figure 1: AMR graph for the sentence *"North Korea froze its nuclear actions in exchange for two nuclear reactors."*

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic representation which encodes the meaning of a sentence in a rooted and directed graph, whose nodes are abstract semantic concepts and edges are semantic relations between concepts (see Figure 1 for an example). Parsing a sentence into its AMR graph has drawn a lot of research attention in recent years with a number of parsers being developed (Flanigan et al., 2014; Wang et al., 2015b; Pust et al., 2015; Artzi et al., 2015; Peng et al., 2015; Zhou et al., 2016; Goodman et al., 2016; Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017; Foland and Martin, 2017; Konstas et al., 2017).

The nature of abstracting away the association between a concept and a span of words complicates the training of the AMR parser. A word-concept aligner is required to derive such association from the sentence-AMR-graph pair and the alignment output is then used as reference to train the AMR parser. In previous works, such alignment is extracted by either greedily applying a set of heuristic rules (Flanigan et al., 2014) or adopting the unsupervised word alignment technique from machine translation (Pourdamghani et al., 2014; Wang and Xue, 2017).

The rule-based aligner – JAMR aligner proposed by Flanigan et al. (2014) is widely used in previous works thanks to its flexibility of incorporating additional linguistic resources like WordNet. However, achieving good alignments with the JAMR aligner still faces some difficult challenges. The first challenge is deriving an optimal alignment in ambiguous situations. Taking the sentence-AMR-graph pair in Figure 1 for example, the JAMR aligner doesn't distinguish between the two "*nuclear*"s in the sentence and can yield sub-optimal alignment in which the first "*nuclear*" is aligned to the `nucleus~2` concept. The second challenge is recalling more semantically matched word-concept pair without harming the

---

* Email corresponding.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2422–2430
Brussels, Belgium, October 31 - November 4, 2018.

alignment precision. The JAMR aligner adopts a rule that aligns the word-concept pair which at least have a common longest prefix of 4 characters, but omitting the shorter cases like aligning the word "*actions*" to the concept `act-01` and the semantically matched cases like aligning the word "*example*" to the concept `exemplify-01`. The final challenge which is faced by both the rule-based and unsupervised aligners is tuning the alignment with downstream parser learning. Previous works treated the alignment as a fixed input. Its quality is never evaluated and its alternatives are never explored. All these challenges make the JAMR aligner achieve only an alignment F1 score of about 90% and influence the performance of the trained AMR parsers.

In this paper, we propose a novel method to solve these challenges and improve the word-to-concept alignment, which further improves the AMR parsing performance. A rule-based aligner and a transition-based oracle AMR parser lie in the core of our method. For the aligner part, we incorporate rich semantic resources into the JAMR aligner to recall more word-concept pairs and cancel its greedily aligning process. This leads to multiple alignment outputs with higher recall but lower precision. For the parser part, we propose a new transition system that can parse the raw sentence into AMR graph directly. Meanwhile, a new oracle algorithm is proposed which produces the best achievable AMR graph from an alignment. Our aligner is tuned by our oracle parser by feeding the alignments to the oracle parser and picking the one which leads to the highest Smatch F1 score (Cai and Knight, 2013). The chosen alignment is used in downstream training of the AMR parser. Based on the newly proposed aligner and transition system, we develop a transition-based parser that directly parses a sentence into its AMR graph and it can be easily improved through ensemble thanks to its simplicity.

We conduct experiments on LDC2014T12 dataset.[1] Both intrinsic and extrinsic evaluations are performed on our aligner. In the intrinsic evaluation, our aligner achieves an alignment F1 score of 95.2%. In the extrinsic evaluation, we replace the JAMR aligner with ours in two open-sourced AMR parsers, which leads to consistent improvements on both parsers. We also evaluate our transition-based parser on the same dataset. Using both our aligner and ensemble, a score of 68.1 Smatch F1 is achieved without any additional resources, which is comparable to the parser of Wang and Xue (2017). With additional part-of-speech (POS) tags, our ensemble parser achieves 68.4 Smatch F1 score and outperforms that of Wang and Xue (2017).

The contributions of this paper come in two folds:

- We propose a new AMR aligner (§3) which recalls more semantically matched pairs and produces multiple alignments. We also propose a new transition system for AMR parsing (§4.1) and use its oracle (§4.2) to pick the alignment that leads to the highest-scored achievable AMR graph (§4.3). Both intrinsic and extrinsic evaluations (§5) show the effectiveness of our aligner by achieving higher F1 score and consistently improving two open-sourced AMR parsers.
- We build a new transition-based parser (§4.4) upon our aligner and transition system which directly parses a raw sentence into its AMR graph. Through simple ensemble, our parser achieves 68.4 Smatch F1 score with only words and POS tags as input (§6) and outperforms the parser of Wang and Xue (2017).

Our code and the alignments for LDC2014T12 dataset are publicly available at `https://github.com/Oneplus/tamr`

2 Related Work

AMR Parsers. AMR parsing maps a natural language sentence into its AMR graph. Most current parsers construct the AMR graph in a two-staged manner which first identifies concepts (nodes in the graph) from the input sentence, then identifies relations (edges in the graph) between the identified concepts. Flanigan et al. (2014) and their follow-up works (Flanigan et al., 2016; Zhou et al., 2016) model the parsing problem as finding the maximum spanning connected graph. Wang et al. (2015b) proposes to greedily transduce the dependency tree into AMR graph and a bunch of works (Wang et al., 2015a; Goodman et al., 2016; Wang and Xue, 2017) further improve the transducer's performance with rich features and imitation learning.[2] Transition-based methods

---

[1] `catalog.ldc.upenn.edu/ldc2014t12`

[2] Wang et al. (2015b) and the follow-up works refer their transducing process as "transition-based". However, to dis-

that directly parse an input sentence into its AMR graph have also been studied (Ballesteros and Al-Onaizan, 2017; Damonte et al., 2017). In these works, the concept identification and relation identification are performed jointly.

An aligner which maps a span of words into its concept serves to the generation of training data for the concept identifier, thus is important to the parser training. Missing or incorrect alignments lead to poor concept identification, which then hurt the overall AMR parsing performance. Besides the typical two-staged methods, the aligner also works in some other AMR parsing algorithms like that using syntax-based machine translation (Pust et al., 2015), sequence-to-sequence (Peng et al., 2017; Konstas et al., 2017), Hyperedge Replacement Grammar (Peng et al., 2015) and Combinatory Category Grammar (Artzi et al., 2015).

Previous aligner works solve the alignment problem in two different ways. The rule-based aligner (Flanigan et al., 2014) defines a set of heuristic rules which align a span of words to the graph fragment and greedily applies these rules. The unsupervised aligner (Pourdamghani et al., 2014; Wang and Xue, 2017) uncovers the word-to-concept alignment from the linearized AMR graph through EM. All these approaches yield a single alignment for one sentence and its effect on the downstream parsing is not considered.

JAMR Aligner (Flanigan et al., 2014). Two components exist in the JAMR aligner: 1) a set of heuristic rules and 2) a greedy search process.

The heuristic rules in the JAMR aligner are a set of indicator functions $\rho(c, w_{s,e})$ which take a concept c and a span of words $w_{s,e}$ starting from s and ending with e as input and return whether they should be aligned. These rules can be categorized into *matching rules* and *updating rules*. The matching rules directly compare c with $w_{s,e}$ and determine if they should be aligned. The updating rules first retrieve the concept c' that $w_{s,e}$ aligns, then determine if c and $w_{s,e}$ should be aligned by checking whether c and c' meet some conditions. Here, we illustrate how *update rules* work by applying a rule named **Entity Type** on the AMR graph in Figure 1 as an example. When determining if the entity type concept `country` should be aligned to "*North Korea*", the **Entity Type** rule first retrieve that this span is aligned to the fragment `(name :op1 "North" :op2 "Korea")`, then determine if they are aligned by checking if `name` is the tail concept of `country`.

The greedy search process applies rules in a manually defined order. The results are mutually exclusive which means once a graph fragment is aligned by one rule, it cannot be realigned. By doing so, conflicts between the alignments produced by different rules are resolved. Flanigan et al. (2014) didn't talk about the principle of orders but it generally follows the principle that 1) the matching rules have higher priorities than the updating rules, and 2) exact matching rules have higher priorities than the fuzzy matching rules.

3 Enhanced Rule-based Aligner

3.1 Enhancing Aligner with Rich Semantic Resources

Error propagates in the greedy search process. An alignment error can lead to future errors because of the dependencies and mutual exclusions between rules. In the JAMR aligner, rules that recall more alignments but introduce errors are carefully opted out and it influences the aligner's performance. Our motivation is to use rich semantic resources to recall more alignments. Instead of resolving the resulted conflicts and errors by greedy search, we keep the multiple alignments produced by the aligner and let a parser decide the best alignment.

In this paper, we use two kinds of semantic resources to recall more alignments, which include the similarity drawn from *Glove embedding* (Pennington et al., 2014)[3] and the *morphosemantic* database (Fellbaum et al., 2009) in the WordNet project[4]. Two additional matching schemes **semantic match** and **morphological match** are proposed as:

Semantic Match. *Glove embedding* encodes a word into its vector representation. We define *semantic match* of a concept as a word in the sentence that has a cosine similarity greater than 0.7 in the embedding space with the concept striping off trailing number (e.g. `run-01` → `run`).

Morphological Match. *Morphosemantic* is a database that contains links among derivational

---

tinguish their work with that of Damonte et al. (2017) and Ballesteros and Al-Onaizan (2017), we use the term "transduce" instead.

[3] `nlp.stanford.edu/projects/glove/`

[4] `wordnet.princeton.edu/wordnet/download/standoff/`

(Semantic Named Entity) Applies to `name` concepts and their `opn` children. Matches a span that matches the semantic match of each child in numerical order.

(Morphological Named Entity) Applies to `name` concepts and their `opn` children. Matches a span that matches the morphological match of each child in numerical order.

(Semantic Concept) Applies to any concept. Strips off trailing '-[0-9]+' from the concept, and matches any semantic matching word.

(Morphological Concept) Applies to any concept. Strips off trailing '-[0-9]+' from the concept, and matches any morphological matching word or WordNet lemma.

Table 1: The extended rules.

links connecting noun and verb senses (e.g., "*example*" and `exemplify`). We define morphological match of a concept as a word in the sentence having the (*word*, `concept`) link in the database.

By defining the **semantic match** and **morphological match**, we extend the rules in Flanigan et al. (2014) with four additional matching rules as shown in Table 1. These rules are intended to recall the concepts or entities which either semantically resemble a span of words but differ in the surface form, or match a span of words in their morphological derivation.

3.2 Producing Multiple Alignments

Using the rules in the JAMR aligner along with our four extended matching rules, we propose an algorithm to draw multiple alignments from a pair of sentence and AMR graph and it is shown in Algorithm 1. In this algorithm, A_c denotes the set of candidate alignments for a graph fragment c, in which each alignment is represented as a tuple (s, e, c') where s denotes the starting position, e denotes the ending position, and c' denotes the concept that lead to this alignment. At the beginning, A_c is initialized as an empty set (line 1 to 2). Then all the matching rules are tried to align a span of words to that fragment (line 3 to 7). After applying all the matching rules, all the updating rules are repeatedly applied until no new alignment is generated in one iteration (line 8 to 16). During applying the updating rules, we keep track of the dependencies between fragments. Finally, all the possible combination of the alignments are enumerated without considering the one that violates the fragment dependencies (line 17 to 26).

Algorithm 1: Our alignment algorithm.

```
Input: An AMR graph with a set of graph fragments $C$;
       a sentence $W$; a set of matching rules $\mathrm{P}_M$; and
       a set of updating rules $\mathrm{P}_U$.
Output: a set of alignments $\mathcal{A}$.
1  for $c \in C$ do
2      $A_c \leftarrow \emptyset$;
3  for $\rho_M \in \mathrm{P}_M$ do
4      for $w_{s,e} \leftarrow spans(W)$ do
5          for $c \in C$ do
6              if $\rho_M(c, w_{s,e})$ then
7                  $A_c \leftarrow A_c \cup (s, e, \mathsf{nil})$;
8  updated $\leftarrow$ true ;
9  while updated is true do
10     updated $\leftarrow$ false;
11     for $\rho_U \in \mathrm{P}_U$ do
12         for $c, c' \in C \times C$ do
13             for $(s, e, d) \in A'_c$ do
14                 if $\rho_U(c, w_{s,e}) \wedge (s, e, c') \notin A_c$ then
15                     $A_c \leftarrow A_c \cup (s, e, c')$;
16                     updated $\leftarrow$ true;
17 $\mathcal{A} \leftarrow \emptyset$ ;
18 for $(a_1, ..., a_c) \in CartesianProduct(A_1, ..., A_{|C|})$ do
19     legal $\leftarrow$ true;
20     for $a \in (a_1, ..., a_c)$ do
21         $(s, e, c') \leftarrow a$;
22         $(s', e', d) \leftarrow a_{c'}$;
23         if $s \neq s' \wedge e \neq e'$ then
24             legal $\leftarrow$ false ;
25     if legal then
26         $\mathcal{A} \leftarrow \mathcal{A} \cup (a_1, ..., a_c)$;
```

4 Transition-based AMR Parser

Our enhanced rule-based aligner produces multiple alignments, and we would like to use our parser to evaluate their qualities. A parameterized parser does not accomplish such goal because training its parameters depends on the aligner's outputs. A deterministic parser works in this situation but is required to consider the association between concepts and spans. This stops the deterministic parsers which build AMR graph only from the derived concepts[5] from being used because they do not distinguish alignments that yields to the same set of concepts.[6]

This discussion shows that to evaluate the quality of an alignment, we need a deterministic (oracle) parser which builds the AMR graph from the raw sentence. Ballesteros and Al-Onaizan (2017) presented a transition-based parser that directly parses a sentence into its AMR graph. A transition system which extends the swap-based dependency parsing system to handle AMR non-projectivities (Damonte et al., 2017) was proposed in their work.

---

[5] e.g. the reference relation identifier in Flanigan et al. (2014) and the oracle transducer in Wang et al. (2015b).

[6] recall the "*nuclear*" example in Section 1.

Transition	Current State	Resulting State	Description
Drop	$[\sigma\|\mathtt{s}_0,\ \delta,\ b_0\|\beta,\ A]$	$[\sigma\|\mathtt{s}_0,\ \delta,\ \beta,\ A]$	pops out the word that doesn't convey any semantics (e.g., function words and punctuations).
Merge	$[\sigma\|\mathtt{s}_0,\ \delta,\ b_0\|b_1\|\beta,\ A]$	$[\sigma\|\mathtt{s}_0,\ \delta,\ b_0\_b_1\|\beta,\ A]$	concatenates a sequence of words into a span, which can be derived as a named entity (`name`) or `date-entity`.
Confirm(`c`)	$[\sigma\|\mathtt{s}_0,\ \delta,\ b_0\|\beta,\ A]$	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathtt{c}\|\beta,\ A]$	derives the first element of the buffer (a word or span) into a concept `c`.
Entity(`c`)	$[\sigma\|\mathtt{s}_0,\ \delta,\ b_0\|\beta,\ A]$	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathtt{c}\|\beta,\ A \cup \mathrm{relations}(c)]$	a special form of Confirm that derives the first element into an entity and builds the internal entity AMR fragment.
New(`c`)	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathtt{b}_0\|\beta,\ A]$	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathtt{c}\|\mathrm{b}_0\|\beta,\ A]$	generates a new concept `c` and pushes it to the front of the buffer.
Left(`r`)	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathtt{b}_0\|\beta,\ A]$	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathtt{b}_0\|\beta,\ A \cup \{\mathtt{s}_0 \xleftarrow{\mathtt{r}} \mathtt{b}_0\}]$	links a relation `r` between the top concepts on the stack and the buffer.
Right(`r`)	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathtt{b}_0\|\beta,\ A]$	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathtt{b}_0\|\beta,\ A \cup \{\mathtt{s}_0 \xrightarrow{\mathtt{r}} \mathtt{b}_0\}]$	
Cache	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathrm{b}_0\|\beta,\ A]$	$[\sigma,\ \mathtt{s}_0\|\delta,\ \mathrm{b}_0\|\beta,\ A]$	passes the top concept of the stack onto the deque.
Shift	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathtt{b}_0\|\beta,\ A]$	$[\sigma\|\mathtt{s}_0\|\delta\|\mathtt{b}_0,\ [\,],\ \beta,\ A]$	shifts the first concept of the buffer onto the stack along with those on the deque.
Reduce	$[\sigma\|\mathtt{s}_0,\ \delta,\ \mathrm{b}_0\|\beta,\ A]$	$[\sigma,\ \delta,\ \mathrm{b}_0\|\beta,\ A]$	pops the top concept of the stack.

Table 2: The transition system. The letters in `monospace` font represent the concepts, the *italic* letters represent the word, and the letters in normal font are either concepts or words.

Their work presented the possibility for the oracle parser, but their oracle parser was not touched explicitly. What's more, in the non-projective dependency parsing, Choi and McCallum (2013)'s extension to the list-based system (Nivre, 2008) with caching mechanism achieves expected linear time complexity and requires fewer actions to parse a non-projective tree than the swap-based system. Their extension to transition-based AMR parsing is worth studying.

In this paper, we propose to extend Choi and McCallum (2013)'s transition system to AMR parsing and present the corresponding oracle parser. The oracle parser is used for tuning our aligner and training our parser. We also present a comprehensive comparison of our system with that of Ballesteros and Al-Onaizan (2017) in Section 6.3.

4.1 List-based Extension for AMR Parsing

We follow Choi and McCallum (2013) and define a state in our transition system as a quadruple $s = (\sigma, \delta, \beta, A)$, where σ is a stack holding processed words, δ is a deque holding words popped out of σ that will be pushed back in the future, and β is a buffer holding unprocessed words. A is a set of labeled relations. A set of actions is defined to parse sentence into AMR graph. Table 2 gives a formal illustration of these actions and how they work. The first five actions in Table 2 are our extended actions, and they are used to deriving concepts from the input sentence.

4.2 Oracle Parser

Given an alignment and the gold standard AMR graph, we can build the best AMR graph by repeatedly applying one of these actions and this is what we called *oracle* parser. Before running the oracle parser, we first remove the concepts which aren't aligned with any span of words from the AMR graph. During running the oracle parser, for a state $s = (\sigma|\mathtt{s}_0,\ \delta,\ \mathrm{b}_0|\mathrm{b}_1|\beta,\ A)$, our oracle parser decides which action to apply by checking the following conditions one by one.

1. If b_0 is a word and it doesn't align to any concept, perform Drop.
2. If b_1 is within a span in the alignment, perform Merge.
3. If b_0 is a word or span and it only aligns to one entity concept `c`, perform Entity(`c`).
4. If b_0 is a word or span and it aligns to one or more concepts, perform Confirm(`c`) where `c` is the concept b_0 aligns and has the longest graph distance to the root.
5. If b_0 is a concept and its head concept c has the same alignment as b_0, perform New(`c`).
6. If b_0 is a concept and there is an unprocessed edge `r` between $\mathtt{s}_0$ and $\mathtt{t}_0$, perform Left(`r`) or Right(`r`) according to `r`'s direction.

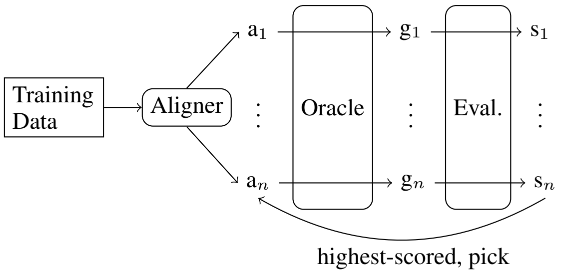

Figure 2: The workflow of tuning the aligner with the oracle parser. a_i denotes the i-th alignment, g_i denotes the i-th AMR graph, and s_i denotes the score of the i-th AMR graph.

7. If s_0 has unprocessed edge, perform CACHE.
8. If s_0 doesn't have unprocessed edge, perform REDUCE.
9. perform SHIFT.

We test our oracle parser on the *hand-align* data created by Flanigan et al. (2014) and it achieves 97.4 Smatch F1 score.[7] Besides the errors resulted from incorrect manual alignments, entity errors made by the limitation of our ENTITY(`c`) action count a lot. Since our ENTITY action directly converts the surface form of a word span into an entity. It cannot correctly generate entity names when they require derivation,[8] or where tokenization errors exist.[9]

4.3 Tune the Aligner with Oracle Parser

Using our oracle parser, we tune the aligner by picking the alignment which leads to the highest-scored AMR graph from the set of candidates (see Figure 2 for the workflow). When more than one alignment achieve the highest score, we choose the one with the smallest number of actions. Intuitively, choosing the one with the smallest number of actions will encourage structurally coherent alignment[10] because coherent alignment requires fewer CACHE actions.

4.4 Parsing Model

Based on our aligner and transition system, we propose a transition-based parser which parse the raw sentence directly into its AMR graph. In this paper, we follow Ballesteros and Al-Onaizan (2017) and use StackLSTM (Dyer et al., 2015) to model the states. The score of a transition action a on state s is calculated as

$$p(a|s) = \frac{\exp\{g_a \cdot \textsc{StackLSTM}(s) + b_a\}}{\sum_{a'} \exp\{g_{a'} \cdot \textsc{StackLSTM}(s) + b_{a'}\}},$$

where $\textsc{StackLSTM}(s)$ encodes the state s into a vector and g_a is the embedding vector of action a. We encourage the reader to refer Ballesteros and Al-Onaizan (2017) for more details.

Ensemble. Ensemble has been shown as an effective way of improving the neural model's performance (He et al., 2017). Since the transition-based parser directly parse a sentence into its AMR graph, ensemble of several parsers is easier compared to the two-staged AMR parsers. In this paper, we ensemble the parsers trained with different initialization by averaging their probability distribution over the actions.

5 Alignment Experiments

5.1 Settings

We evaluate our aligner on the LDC2014T12 dataset. Two kinds of evaluations are carried out including the *intrinsic* and *extrinsic* evaluations.

For the *intrinsic* evaluation, we follow Flanigan et al. (2014) and evaluate the F1 score of the alignments produced by our aligner against the manually aligned data created in their work (*hand-align*). We also use our oracle parser's performance as an *intrinsic* evaluation assuming that better alignment leads to higher scored oracle parser.

For the *extrinsic* evaluation, we plug our alignment into two open-sourced AMR parsers: 1) JAMR (Flanigan et al., 2014, 2016) and 2) CAMR (Wang et al., 2015b,a) and evaluate the final performances of the AMR parsers on both the newswire proportion and the entire dataset of LDC2014T12. We use the configuration in Flanigan et al. (2016) for JAMR and the configuration in Wang et al. (2015a) without semantic role labeling (SRL) features for CAMR.

5.2 Results

Intrinsic Evaluation. Table 3 shows the intrinsic evaluation results, in which our alignment intrinsically outperforms JAMR aligner by achieving better alignment F1 score and leading to a higher scored oracle parser.

---

[7] Since some alignments in *hand-align* were created on incorrect AMR annotations, we filter out them and only use the correct subset which has 136 pairs of alignment and AMR graph. This data is also used in our intrinsic evaluation.

[8] e.g., "*North Koreans*" cannot be parsed into `(name :op1 "North" :op2 "Korea")`

[9] e.g., "*Wi Sung - lac*" cannot be parsed into `(name :op1 "Wi" :op2 "Sung-lac")`

[10] e.g. the first "*nuclear*" aligned to `nucleus~1` in Fig. 1

Aligner	Alignment F1 (on hand-align)	Oracle's Smatch (on dev. dataset)
JAMR	90.6	91.7
Our	95.2	94.7

Table 3: The intrinsic evaluation results.

model	newswire	all
JAMR parser: Word, POS, NER, DEP		
+ JAMR aligner	71.3	65.9
+ Our aligner	73.1	67.6
CAMR parser: Word, POS, NER, DEP		
+ JAMR aligner	68.4	64.6
+ Our aligner	68.8	65.1

Table 4: The parsing results.

Extrinsic Evaluation. Table 4 shows the results. From this table, we can see that our alignment consistently improves all the parsers by a margin ranging from 0.5 to 1.7. Both the intrinsic and the extrinsic evaluations show the effectiveness our aligner.

5.3 Ablation

To have a better understanding of our aligner, we conduct ablation test by removing the *semantic matching* and *oracle parser tuning* respectively and retrain the JAMR parser on the newswire proportion. The results are shown in Table 5. From this table, we can see that removing either of these components harms the performance. Removing *oracle parser tuning* leads to severe performance drop and the score is even lower than that with JAMR aligner. We address this observation to that alignment noise is introduced by the semantic matching especially by the word embedding similarity component. Without filtering the noise by our oracle parser, just introducing more matching rules will harm the performance.

6 Parsing Experiments

6.1 Settings

We use the same settings in our aligner extrinsic evaluation for the experiments on our transition-based parser. For the input to the parser, we tried two settings: 1) using only words as input, and 2) using words and POS tags as input. Automatic POS tags are assigned with Stanford POS tagger (Manning et al., 2014). Word embedding from Ling et al. (2015) is used in the same way with Ballesteros and Al-Onaizan (2017). To opt out the effect of different initialization in training the neural network, we run 10 differently seeded runs and report their average performance following Reimers and Gurevych (2017).

model	newswire
JAMR parser + Our aligner	73.1
- Semantic matching	72.7
- Oracle Parser Tuning	67.6
JAMR parser + JAMR aligner	71.3

Table 5: The ablation test results.

6.2 Results

Table 6 shows the performance of our transition-based parser along with comparison to the parsers in the previous works. When compared with our transition-based counterpart (Ballesteros and Al-Onaizan, 2017), our word-only model outperforms theirs using the same JAMR alignment. The same trend is witnessed using words and POS tags as input. When replacing the JAMR alignments with ours, the parsing performances are improved in the same way as in Table 4, which further confirms the effectiveness of our aligner.

The second block in Table 6 shows the results of our ensemble parser, in which ensemble significantly improves the performance and more parsers ensembled, more improvements are achieved. An ensemble of 10 parsers with only words as input achieves 68.1 Smatch F1 score which is comparable to the AMR parser of Wang and Xue (2017). Using the minimal amount of additional syntactic information – POS tags, the performance of the ensemble of 10 parsers is further pushed to 68.4, which surpasses that of Wang and Xue (2017) which relied on named entity recognition (NER) and dependency parsing (DEP).

A further study on the speed shows that our 10 parser ensemble can parse 43 tokens per second which is faster than JAMR (7 tokens/sec.) and CAMR (24 tokens/sec.) thanks to the simplicity of our model and independence of preprocessing, like NER and DEP.[11]

6.3 Comparison to Ballesteros and Al-Onaizan (2017)

To explain the improved performance against Ballesteros and Al-Onaizan (2017) in Table 6, we

---

[11] In our speed comparison, we also count the time of preprocessing for JAMR and CAMR. All the comparison is performed in the same single-threaded settings.

model	newswire	all
Our single parser: Word only		
+ JAMR aligner	68.6	63.9
+ Our aligner	69.3	64.7
Our single parser: Word, POS		
+ JAMR aligner	68.8	64.6
+ Our aligner	69.8	65.2
Our ensemble: Word only + Our aligner		
x3	71.9	67.4
x10	72.5	68.1
Our ensemble: Word, POS + Our aligner		
x3	72.5	67.7
x10	73.3	**68.4**
BA17: Word only†	68	63
+ POS	68	63
+ POS, DEP	69	64
Damonte et al. (2017)‡	-	66
Artzi et al. (2015)	66.3	-
Wang et al. (2015a)	70	66
Pust et al. (2015)	-	67.1
Zhou et al. (2016)	71	66
Goodman et al. (2016)	70	-
Wang and Xue (2017)	-	68.1

Table 6: The parsing results. xn denotes the ensemble of n differently initialized parsers. The difference in rounding is due to previous works report differently rounded results. † BA17 represents the result of Ballesteros and Al-Onaizan (2017), ‡ Damonte et al. (2017)'s result is drawn from Ballesteros and Al-Onaizan (2017).

give a comprehensive comparison between our transition system and that of Ballesteros and Al-Onaizan (2017).

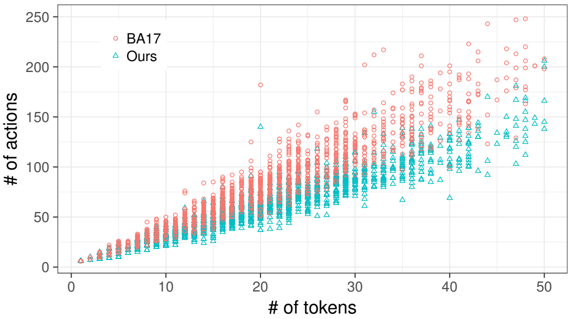

Figure 3: Number of actions required to parse the development set by two systems.

Capability. In both these two systems, a span of words can only be derived into concept for one time. "Patch" actions are required to generate new concepts from the one that is aligned to the same span.[12] Ballesteros and Al-Onaizan (2017) uses a DEPENDENT action to generate one tail concept for one hop and cannot deal with the cases which have a chain of more than two concepts aligned to the same span. Our list-based system differs theirs by using a NEW action to deal these cases. Since the new concept is pushed onto the buffer, NEW action can be repeatedly applied and used to generate arbitrary concepts that aligned to the same span. On the development set of LDC2014T12, our oracle achieves 91.7 Smatch F1 score over the JAMR alignment, which outperforms Ballesteros and Al-Onaizan (2017)'s oracle (89.5 in their paper) on the same alignment. This result confirms that our list-based system is more powerful.

Number of Actions. Our list-based system also differs theirs in the number of oracle actions required to parse the same AMR graphs. We use the oracles from two systems to parse the development set of LDC2014T12 on the same JAMR alignments. Figure 3 shows the comparison in which our system clearly uses fewer actions (the average number of our system is 63.7 and that of Ballesteros and Al-Onaizan (2017) is 86.4). Using fewer actions makes the parser learned from the oracle less prone to error propagation. We attribute the improved performance in Table 6 to this advantage of transition system.

7 Conclusion

In this paper, we propose a new AMR aligner which is tuned by a novel transition-based AMR oracle parser. Our aligner is also enhanced by rich semantic resource and recalls more alignments. Both the intrinsic and extrinsic evaluations show the effectiveness of our aligner by achieving higher alignment F1 score and consistently improving two open-sourced AMR parsers. We also develop transition-based AMR parser based on our aligner and transition system and it achieves a performance of 68.4 Smatch F1 score via ensemble with only words and POS tags as input.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was

---

[12] e.g., three concepts in the fragment `(person :source (country :name (name :op1 "North" :op2 "Korea")))` are aligned to "*North Koreans*".

supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61632011 and 61772153.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proc. of EMNLP*.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using Stack-LSTMs. In *Proc. of EMNLP*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proc. of ACL*.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proc. of ACL*.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proc. of EACL*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Christiane Fellbaum, Anne Osherson, and Peter E. Clark. 2009. Putting semantics into WordNet's "morphosemantic" links. In *Human Language Technology. Challenges of the Information Society*.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proc. of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of ACL*.

William Foland and James H. Martin. 2017. Abstract meaning representation parsing using LSTM recurrent neural networks. In *Proc. of ACL*.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proc. of ACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proc. of ACL*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proc. of ACL*.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proc. of NAACL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4).

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for amr parsing. In *Proc. of CoNLL*.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural amr parsing. In *Proc. of EACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *Proc. of EMNLP*.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into abstract meaning representation using syntax-based machine translation. In *Proc. of EMNLP*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proc. of EMNLP*.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of amr parsing. In *Proc. of EMNLP*.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proc. of ACL*.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for amr parsing. In *Proc. of NAACL*.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proc. of EMNLP*.

Dependency-based Hybrid Trees for Semantic Parsing

Zhanming Jie and **Wei Lu**
Singapore University of Technology and Design
8 Somapah Road, Singapore, 487372
`zhanming_jie@mymail.sutd.edu.sg`, `luwei@sutd.edu.sg`

Abstract

We propose a novel *dependency-based hybrid tree* model for semantic parsing, which converts natural language utterance into machine interpretable meaning representations. Unlike previous state-of-the-art models, the semantic information is interpreted as the latent dependency between the natural language words in our joint representation. Such dependency information can capture the interactions between the semantics and natural language words. We integrate a neural component into our model and propose an efficient dynamic-programming algorithm to perform tractable inference. Through extensive experiments on the standard multilingual GeoQuery dataset with eight languages, we demonstrate that our proposed approach is able to achieve state-of-the-art performance across several languages. Analysis also justifies the effectiveness of using our new dependency-based representation.[1]

1 Introduction

Semantic parsing is a fundamental task within the field of natural language processing (NLP). Consider a natural language (NL) sentence and its corresponding meaning representation (MR) as illustrated in Figure 1. Semantic parsing aims to transform the natural language sentences into machine interpretable meaning representations automatically. The task has been popular for decades and keeps receiving significant attention from the NLP community. Various systems (Zelle and Mooney, 1996; Kate et al., 2005; Zettlemoyer and Collins, 2005; Liang et al., 2011) were proposed over the years to deal with different types of semantic representations. Such models include structure-based models (Wong and Mooney, 2006; Lu et al., 2008; Kwiatkowski et al., 2010; Jones et al., 2012) and neural network based models (Dong and Lapata, 2016; Cheng et al., 2017).

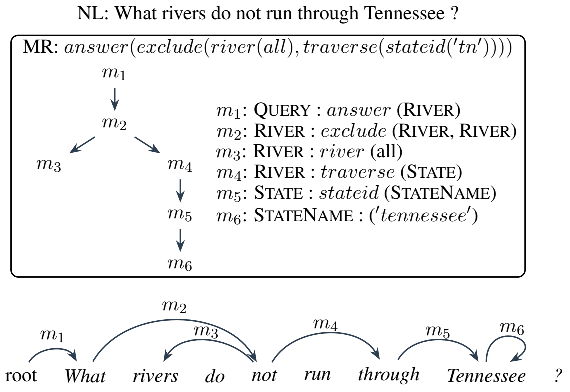

Figure 1: Top: natural language (NL) sentence; middle: meaning representation (MR); bottom: dependency-based hybrid tree representation.

Following various previous research efforts (Wong and Mooney, 2006; Lu et al., 2008; Jones et al., 2012), in this work, we adopt a popular class of semantic formalism – logical forms that can be equivalently represented as tree structures. The tree representation of an example MR is shown in the middle of Figure 1. One challenge associated with building a semantic parser is that the exact correspondence between the words and atomic semantic units are not explicitly given during the training phase. The key to the building of a successful semantic parsing model lies in the identification of a good *joint* latent representation of both the sentence and its corresponding semantics. Example joint representations proposed in the literature include a chart used in phrase-based translation (Wong and Mooney, 2006), a constituency tree-like representation known as *hybrid tree* (Lu et al., 2008), and a CCG-based derivation tree (Kwiatkowski et al., 2010).

Previous research efforts have shown the effec-

---

[1] We make our system and code available at `http://statnlp.org/research/sp`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2431–2441
Brussels, Belgium, October 31 - November 4, 2018.

tiveness of using dependency structures to extract semantic representations (Debusmann et al., 2004; Cimiano, 2009; Bédaride and Gardent, 2011; Stanovsky et al., 2016). Recently, Reddy et al. (2016, 2017) proposed a model to construct logical representations from sentences that are parsed into dependency structures. Their work demonstrates the connection between the dependency structures of a sentence and its underlying semantics. Although their setup and objectives are different from ours where externally trained dependency parsers are assumed available and their system was trained to use the semantics for a specific down-stream task, the success of their work motivates us to propose a novel joint representation that can explicitly capture dependency structures among words for the semantic parsing task.

In this work, we propose a new joint representation for both semantics and words, presenting a new model for semantic parsing. Our main contributions can be summarized as follows:

- We present a novel *dependency-based hybrid tree* representation that captures both words and semantics in a joint manner. Such a dependency tree reveals semantic dependencies between words which are easily interpretable.
- We show that exact dynamic programming algorithms for inference can be designed on top of our new representation. We further show that the model can be integrated with neural networks for improved effectiveness.
- Extensive experiments conducted on the standard multilingual GeoQuery dataset show that our model outperforms the state-of-the-art models on 7 out of 8 languages. Further analysis confirms the effectiveness of our dependency-based representation.

To the best of our knowledge, this is the first work that models the semantics as latent dependencies between words for semantic parsing.

2 Related Work

The literature on semantic parsing has focused on various types of semantic formalisms. The λ-calculus expressions (Zettlemoyer and Collins, 2005) have been popular and widely used in semantic parsing tasks over recent years (Dong and Lapata, 2016; Gardner and Krishnamurthy, 2017; Reddy et al., 2016, 2017; Susanto and Lu, 2017a; Cheng et al., 2017). Dependency-based compositional semantics (DCS)[2] was introduced by Liang et al. (2011), whose extension, λ-DCS, was later proposed by Liang (2013). Various models (Berant et al., 2013; Wang et al., 2015; Jia and Liang, 2016) on semantic parsing with the λ-DCS formalism were proposed. In this work, we focus on the tree-structured semantic formalism which has been examined by various research efforts (Wong and Mooney, 2006; Kate and Mooney, 2006; Lu et al., 2008; Kwiatkowski et al., 2010; Jones et al., 2012; Lu, 2014; Zou and Lu, 2018).

Wong and Mooney (2006) proposed the WASP semantic parser that regards the task as a phrase-based machine translation problem. Lu et al. (2008) proposed a generative process to generate natural language words and semantic units in a joint model. The resulting representation is called *hybrid tree* where both natural language words and semantics are encoded into a joint representation. The UBL-s (Kwiatkowski et al., 2010) parser applied the CCG grammar (Steedman, 1996) to model the joint representation of both semantic units and contiguous word sequences which do not overlap with one another. Jones et al. (2012) applied a generative process with Bayesian tree transducer and their model also simultaneously generates the meaning representations and natural language words. Lu (2014, 2015) proposed a discriminative version of the hybrid tree model of (Lu et al., 2008) where richer features can be captured. Dong and Lapata (2016) proposed a sequence-to-tree model using recurrent neural networks where the decoder can branch out to produce tree structures. Susanto and Lu (2017b) augmented the discriminative hybrid tree model with multilayer perceptron and achieved state-of-the-art performance.

There exists another line of work that applies given syntactic dependency information to semantic parsing. Titov and Klementiev (2011) decomposed a syntactic dependency tree into fragments and modeled the semantics as relations between the fragments. Poon (2013) learned to derive semantic structures based on syntactic dependency trees predicted by the Stanford dependency parser. Reddy et al. (2016, 2017) proposed a linguistically motivated procedure to transform syntactic dependencies into logical forms. Their semantic parsing performance relies on the quality of the syntactic dependencies. Unlike such efforts, we do not re-

---

[2] Unlike ours, their work captures dependencies between semantic units but not natural language words.

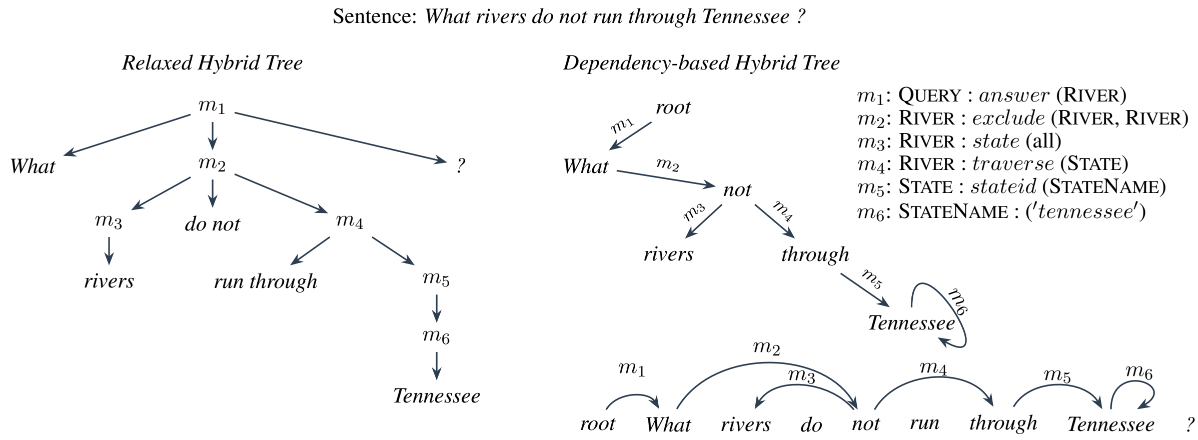

Figure 2: The *relaxed hybrid tree* (left) (Lu, 2014) and our *dependency-based hybrid tree* (right) as well as the flat representation (bottom right) of the example in Figure 1.

quire external syntactic dependencies, but model the semantic units as latent dependencies between natural language words.

3 Approach

3.1 Variable-free Semantics

The variable-free semantic representations in the form of FunQL (Kate et al., 2005) used by the defacto GeoQuery dataset (Zelle and Mooney, 1996) encode semantic compositionality of the logical forms (Cheng et al., 2017). In the tree-structured semantic representations as illustrated in Figure 1, each tree node is a semantic unit of the following form:

$$m_i \equiv \tau_\alpha : p_\alpha(\tau_\beta^*)$$

where m_i denotes the complete semantic unit, which consists of semantic type τ_α, function symbol p_α and an argument list of semantic types τ_β^* (here $*$ denotes that there can be 0, 1, or 2 semantic types in the argument list. This number is known as the *arity* of m_i). Each semantic unit can be regarded as a function that takes in other (partial) semantic representations of certain types as arguments and returns a semantic representation of a specific type. For example in Figure 1, the root unit is represented by m_1, the type of this unit is QUERY, the function name is $answer$ and it has a single argument RIVER which is a semantic type. With recursive function composition, we can obtain a complete MR as shown in Figure 1.

3.2 Dependency-based Hybrid Trees

To jointly encode the tree-structured semantics $\boldsymbol{m}$ and a natural language sentence $\boldsymbol{n}$, we introduce our novel *dependency-based hybrid tree*. Figure 2 (right) shows the two equivalent ways of visualizing the dependency-based hybrid tree based on the example given in Figure 1. In this example, $\boldsymbol{m}$ is the tree-structured semantics $m_1(m_2(m_3, m_4(m_5(m_6))))$ and $\boldsymbol{n}$ is the sentence $\{w_1, w_2, \cdots, w_8\}$[3]. Our dependency-based hybrid tree $\boldsymbol{t}$ consists of a set of dependencies between the natural language words, each of which is labeled with a semantic unit. Formally, a dependency arc is represented as (w_p, w_c, m_i), where w_p is the *parent* of this dependency, w_c is the *child*, and m_i is the semantic unit that serves as the *label* for the dependency arc. A valid dependency-based hybrid tree (with respect to a given semantic representation) allows one to recover the correct semantics from it. Thus, one constraint is that for any two adjacent dependencies (w_p, w_c, m_i) and (w'_p, w'_c, m_j), where $w_c \equiv w'_p$, m_i must be the parent of m_j in the tree-structured representation $\boldsymbol{m}$. For example, in Figure 2, the dependencies (*not*, *through*, m_4) and (*through*, *Tennessee*, m_5) satisfy the above condition. However, we cannot replace (*through*, *Tennessee*, m_5) with, for example, (*through*, *Tennessee*, m_6), since m_6 is not the child of m_4. Furthermore, the number of children for a word in the dependency tree should be consistent with the arity of the corresponding semantic unit that points to it. For example, "*not*" has 2 children in our dependency-based hybrid tree representation because the semantic unit m_2 (i.e., RIVER : $exclude$ (RIVER, RIVER)) has arity 2. Also, "*rivers*" is the leaf as m_3, which points to it, has arity 0. We will discuss in Section 3.3

---

[3] We also introduce a special token "*root*" as w_0.

Abstract Semantic Unit	Arity	Dependency Pattern
A	0	**WW**
B	1	**X**, **WX**, **XW**
C	2	**XY**, **YX**

Table 1: List of dependency patterns.

on how to derive the set of allowable dependency-based hybrid trees for a given $(\boldsymbol{m}, \boldsymbol{n})$ pair.

To understand the potential advantages of our new joint representation, we compare it with the *relaxed hybrid tree* representation (Lu, 2014), which is illustrated on the left of Figure 2. We highlight some similarities and differences between the two representations from the *span level* and *word level* perspectives.

In a relaxed hybrid tree representation, words and semantic units jointly form a constituency tree-like structure, where the former are leaves and the latter are internal nodes of such a joint representation. Such a representation is able to capture alignment between the natural language words and semantics at the span level.[4] For example, m_2 covers the span from "*rivers*" to "*Tennessee*", which allows the interactions between the semantic unit and the span to be captured. Similarly, in our dependency-based hybrid tree, such span level word-semantics correspondence can also be captured. For example, the arc between "*not*" and "*through*" is labeled by the semantic unit m_4. This also allows the interactions between m_4 and words within the span from "*not*" to "*through*" to be captured.

While both models are able to capture the span-level correspondence between words and semantics, we can observe that in the relaxed hybrid tree, some words within the span are more directly related to the semantic unit (e.g., "*do not*" are more related to m_2) and some are not. Specifically, in their representation, the span level information assigned to the parent semantic unit always contains the span level information assigned to all its child semantic units. This may not always be desirable and may lead to irrelevant features. In fact, Lu (2014) also empirically showed that the span-level features may not always be helpful in their representation. In contrast, in our dependency-based hybrid tree, the span covered by m_2 is from "*What*" to "*not*", which only consists of the span level information associated with its first child semantic units. Therefore, our representation is more flexible in capturing the correspondence between words and semantics at the span level, allowing the model to choose the relevant span for features.

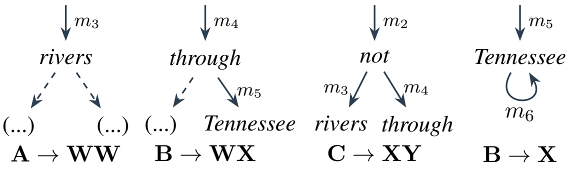

Figure 3: Example dependency patterns used in the dependency-based hybrid tree of Figure 2.

Furthermore, our representation can also capture precise interactions between words through dependency arcs labeled with semantic units. For example, the semantic unit m_4 on the dependency arc from "*not*" to "*through*" in our representation can be used to capture their interactions. However, such information could not be straightforwardly captured in a relaxed hybrid tree, which is essentially a constituency tree-like representation. In the same example, consider the word "*not*" that bridges two arcs labeled by m_2 and m_4. Lexical features defined over such arcs can be used to indirectly capture the interactions between semantic units and guide the tree construction process. We believe such properties can be beneficial in practice, especially for certain languages. We will examine their significance in our experiments later.

3.3 Dependency Patterns

To define the set of allowable dependency-based hybrid tree representation so as to allow us to perform exact inference later, we introduce the *dependency patterns* as shown in Table 1. We use **A**, **B** or **C** to denote the abstract semantic units with arity 0, 1, and 2, respectively. We use **W** to denote a contiguous word span, and **X** and **Y** to denote the first and second child semantic unit, respectively.

We explain these patterns with concrete cases in Figure 3 based on the example in Figure 2. For the first case, the semantic unit m_3 has arity 0, the pattern involved is **WW**, indicating both the left-hand and right-hand sides of "*rivers*" (under the dependency arc with semantic unit m_3) are just word spans (**W**, whose length could be zero). In the second case, the semantic unit m_4 has arity 1, the pattern involved is **WX**, indicating the left-hand side of "*through*" (under the arc of semantic unit m_4) is a word span and the right-hand side should be handled by the first child of m_4 in the

---

[4] We refer readers to (Lu, 2014) for more details.

semantic tree, which is m_5 in this case. In the third case, the semantic unit m_2 has two arguments, and the pattern involved in the example is $\mathbf{XY}$, meaning the left-hand and right-hand sides should be handled by the first and second child semantic units (i.e., m_3 and m_4), respectively.[5] The final case illustrates that we also allow self-loops on our dependency-based hybrid trees, where an arc can be attached to a single word.[6] To avoid an infinite number of self-loops over a word, we set a maximum depth c to restrict the maximum number of recurrences, which is similar to the method introduced in (Lu, 2015).

Based on the dependency patterns, we are able to define the set of all possible allowable *dependency-based hybrid tree* representations. Each representation essentially belongs to a class of *projective* dependency trees where semantic units appear on the dependency arcs and (some of the) words are selected as nodes. The semantic tree can be constructed by following the arcs while referring to the dependency patterns involved.

3.4 Model

Given the natural language words $\boldsymbol{n}$, our task is to predict $\boldsymbol{m}$, which is a tree-structured meaning representation, consisting of a set of semantic units as the nodes in the semantic tree. We use $\boldsymbol{t}$ to denote a dependency-based hybrid tree (as shown in Figure 2), which jointly encodes both natural language words and the gold meaning representation. Let $\mathcal{T}(\boldsymbol{n}, \boldsymbol{m})$ denote all the possible dependency-based hybrid trees that contain the natural language words $\boldsymbol{n}$ and the meaning representation $\boldsymbol{m}$. We adopt the widely-used structured prediction model conditional random fields (CRF) (Lafferty et al., 2001). The probability of a possible meaning representation $\boldsymbol{m}$ and dependency-based hybrid tree $\boldsymbol{t}$ for a sentence $\boldsymbol{n}$ is given by:

$$P_{\mathbf{w}}(\boldsymbol{m}, \boldsymbol{t}|\boldsymbol{n}) = \frac{e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})}}{\sum_{\boldsymbol{m}',\boldsymbol{t}' \in \mathcal{T}(\boldsymbol{n},\boldsymbol{m}')} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m}',\boldsymbol{t}')}}$$

where $\mathbf{f}(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$ is the feature vector defined over the $(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$ tuple, and $\mathbf{w}$ is the parameter vector. Since we do not have the knowledge of the "true" dependencies during training, $\boldsymbol{t}$ is regarded as a latent-variable in our model. We marginalize $\boldsymbol{t}$ in the above equation and the resulting model is a latent-variable CRF (Quattoni et al., 2005):

$$\begin{aligned} P_{\mathbf{w}}(\boldsymbol{m}|\boldsymbol{n}) &= \sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m})} P_{\mathbf{w}}(\boldsymbol{m}, \boldsymbol{t}|\boldsymbol{n}) \\ &= \frac{\sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m})} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})}}{\sum_{\boldsymbol{m}',\boldsymbol{t}' \in \mathcal{T}(\boldsymbol{n},\boldsymbol{m}')} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m}',\boldsymbol{t}')}} \end{aligned} \tag{1}$$

Given a dataset $\mathcal{D}$ of $(\boldsymbol{n}, \boldsymbol{m})$ pairs, our objective is to minimize the negative log-likelihood:[7]

$$\mathcal{L}(\mathbf{w}) = -\sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \log \sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m})} P_{\mathbf{w}}(\boldsymbol{m}, \boldsymbol{t}|\boldsymbol{n}) \tag{2}$$

The gradient for model parameter w_k is:

$$\begin{aligned} \frac{\partial\mathcal{L}(\mathbf{w})}{\partial w_k} =& \sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \sum_{\boldsymbol{m}',\boldsymbol{t}} \mathbf{E}_{P_{\mathbf{w}}(\mathbf{m}',\boldsymbol{t}|\boldsymbol{n})}[f_k(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})] \\ &- \sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \sum_{\boldsymbol{t}} \mathbf{E}_{P_{\mathbf{w}}(\boldsymbol{t}|\boldsymbol{n},\boldsymbol{m})}[f_k(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})] \end{aligned}$$

where $f_k(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$ represents the number of occurrences of the k-th feature. With both the objective and gradient above, we can minimize the objective function with standard optimizers, such as L-BFGS (Liu and Nocedal, 1989) and stochastic gradient descent. Calculation of these expectations involves all possible dependency-based hybrid trees. As there are exponentially many such trees, an efficient inference procedure is required. We will present our efficient algorithm to perform exact inference for learning and decoding in the next section.

3.5 Learning and Decoding

We propose dynamic-programming algorithms to perform efficient and exact inference, which will be used for calculating the objective and gradients discussed in the previous section. The algorithms are inspired by the inside-outside style algorithm (Baker, 1979), graph-based dependency parsing (Eisner, 2000; Koo and Collins, 2010; Shi et al., 2017), and the *relaxed hybrid tree* model (Lu, 2014, 2015). As discussed in Section 3.3, our latent dependency trees are projective as in traditional dependency parsing (Eisner, 1996; Nivre and Scholz, 2004; McDonald et al., 2005) – the dependencies are non-crossing with respect to the word order (see bottom of Figure 1).

---

[5] Analogously, the pattern $\mathbf{YX}$ would mean m_4 handles the left-hand side and m_3 right-hand side.

[6] The limitations associated with disallowing such a pattern have been discussed in the previous work of (Lu, 2015).

[7] We ignore the L_2 regularization term for brevity.

The objective function in Equation 2 can be further decomposed into the following form[8]:

$$\mathcal{L}(\mathbf{w}) = -\sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \log \sum_{\boldsymbol{t}\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m})} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})} + \sum_{(\boldsymbol{n},\boldsymbol{m})\in\mathcal{D}} \log \sum_{\boldsymbol{m}',\boldsymbol{t}'\in\mathcal{T}(\boldsymbol{n},\boldsymbol{m}')} e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m}',\boldsymbol{t}')}$$

We can see the first term is essentially the combined score of all the possible latent structures containing the pair $(\boldsymbol{n}, \boldsymbol{m})$. The second term is the combined score for all the possible latent structures containing $\boldsymbol{n}$. We show how such scores can be calculated in a factorized manner, based on the fact that we can recursively decompose a dependency-based hybrid tree based on the dependency patterns we introduced.

Formally, we introduce two interrelated dynamic-programming structures that are similar to those used in graph-based dependency parsing (Eisner, 2000; Koo and Collins, 2010; Shi et al., 2017), namely *complete span* and *complete arc span*. Figure 4a shows an example of *complete span* (left) and *complete arc span* (right). The *complete span* (over $[i, j]$) consists of a headword (at i) and its descendants on one side (they altogether form a subtree), a dependency pattern and a semantic unit. The *complete arc span* is a span (over $[i, j]$) with a dependency between the headword (at i) and the modifier (at k). We use $C_{i,j,p,m}$ to denote a complete span, where i and j represent the indices of the headword and endpoint, p is the dependency pattern and m is the semantic unit. Analogously, we use $A_{i,k,j,p,m}$ to denote a complete arc span where i and k are used to denote the additional dependency from the word at the i-th position as headword to the word at the k-th position as modifier.

As we can see from the derivation in Figure 4, each type of span can be constructed from smaller spans in a bottom-up manner. Figure 4a shows that a complete span is constructed from a complete arc span following the dependency patterns in Table 1. Figure 4b shows a complete arc span can be simply constructed from two smaller complete spans based on the dependency pattern. In Figure 4c and 4d, we further show how such two complete spans with pattern $\mathbf{X}$ (or $\mathbf{Y}$) and $\mathbf{W}$ can be constructed. Figure 4c illustrates how to model a transition from one semantic unit to another where the parent is m_1 and the child is m_2 in the semantic tree. If m_2 has arity 1, then the pattern is $\mathbf{B}$ following the dependency patterns in Table 1. For spans with a single word, we use the lowercase $\mathbf{w}$ as the pattern to indicate this fact, as shown in Figure 4d. They are the atomic spans used for building larger spans. As the complete span in Figure 4d is associated with pattern $\mathbf{W}$, which means the words within this span are under the semantic unit m_1, we can incrementally construct this span with atomic spans. We illustrate the construction of a complete dependency-based hybrid tree in the supplementary material.

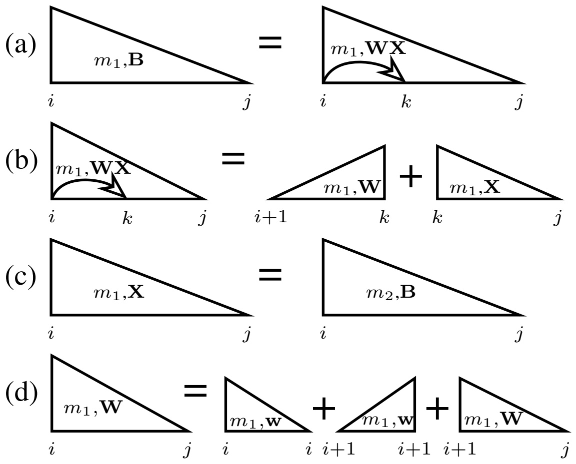

Figure 4: The dynamic-programing structures and derivation of our model. The other direction is symmetric. See supplementary material for the complete structures.

Our final goal during training for a sentence $\boldsymbol{n} = \{w_0, w_1, \cdots, w_N\}$ is to construct all the possible *complete spans* that cover the interval $[0, N]$, which can be represented as $C_{0,N,\cdot,\cdot}$. Similar to the chart-based dependency parsing algorithms (Eisner, 1996, 2000; Koo and Collins, 2010), we can obtain the *inside* and *outside* scores using our dynamic-programming derivation in Figure 4 during the inference process, which can then be used to calculate the objective and feature expectations. Since the spans are defined by at most three free indices, the dependency pattern and the semantic unit, our dynamic-programming algorithm requires $\mathcal{O}(N^3 M)$ time[9] where M is the number of semantic units. The resulting complexity is the same as the relaxed hybrid tree model (Lu, 2014).

During decoding, we can find the optimal (tree-structured) meaning representation $\boldsymbol{m}^*$ for a given

---

[8] Regularization term is excluded for brevity.

[9] We omit a small constant factor associated with patterns.

Feature Type	Examples
Word	"m_4 & *run*", "m_4 & *through*"
Pattern	"m_2 & **XY**", "m_4 & **WX**"
Transition	"m_2 & m_3", "m_2 & m_4"
Head word	"m_2 & *What*", "m_4 & *not*"
Modifier word	"m_2 & *not*", "m_4 & *through*"
Bag of words	"m_4 & *not*", "m_4 & *run*", "m_4 & *through*"

Table 2: Features for the example in Figure 2.

input sentence $\boldsymbol{n}$ by the Viterbi algorithm. This step can also be done efficiently with our dynamic-programming approach, where we switch from marginal inference to MAP inference:

$$\boldsymbol{m}^*, \boldsymbol{t}^* = \underset{\boldsymbol{m},\boldsymbol{t} \in \mathcal{T}(\boldsymbol{n},\boldsymbol{m})}{\arg\max} \; e^{\mathbf{w}\cdot\mathbf{f}(\boldsymbol{n},\boldsymbol{m},\boldsymbol{t})}$$

A similar decoding procedure has been used in previous work (Lu, 2014; Durrett and Klein, 2015) with CKY-based parsing algorithm.

3.6 Features

As shown in Equation 1, the features are defined on the tuple $(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{t})$. With the dynamic-programming procedure, we can define the features over the structures in Figure 2. Our feature design is inspired by the hybrid tree model (Lu, 2015) and graph-based dependency parsing (McDonald et al., 2005). Table 2 shows the feature templates for the example in Figure 2. Specifically, we define simple unigram features (concatenation of a semantic unit and a word that directly appears under the unit), pattern features (concatenation of the semantic unit and the child pattern) and transition features (concatenation of the parent and child semantic units). They form our basic feature set.

Additionally, with the structured properties of dependencies, we can define dependency-related features (McDonald et al., 2005). We use the parent (head) and child (modifier) words of the dependency as features. We also use the bag-of-words covered under a dependency as features. The dependency features are useful in helping improve the performance as we can see in the experiments section.

3.7 Neural Component

Following the approach used in Susanto and Lu (2017b), we could further incorporate neural networks into our latent-variable graphical model. The integration is analogous to the approaches described in the neural CRF models (Do and Artieres, 2010; Durrett and Klein, 2015; Gormley, 2015; Lample et al., 2016), where we use neural networks to learn distributed feature representations within our graphical model.

We employ a neural architecture to calculate the score associated with each dependency arc (w_p, w_c, m) (here w_p and w_c are the parent and child words in the dependency and m is the semantic unit over the arc), where the input to the neural network consists of words (i.e., (w_p, w_c)) associated with this dependency and the neural network will calculate a score for each possible semantic unit, including m. The two words are first mapped to word embeddings $\mathbf{e}_p$ and $\mathbf{e}_c$ (both of dimension d). Next, we use a bilinear layer[10] (Socher et al., 2013; Chen et al., 2016) to capture the interaction between the parent and the child in a dependency:

$$r_i = \mathbf{e}_p^{\mathbf{T}} \mathbf{U}_i \mathbf{e}_c$$

where r_i represents the score for the i-th semantic unit and $\mathbf{U}_i \in \mathbb{R}^{d \times d}$. The scores are then incorporated into the probability expression in Equation 1 during learning and decoding. As a comparison, we also implemented a variant where our model directly takes in the average embedding of $\mathbf{e}_p$ and $\mathbf{e}_c$ as additional features, without using our neural component.

4 Experiments

Data and evaluation methodology We conduct experiments on the publicly available variable-free version of the GeoQuery dataset, which has been widely used for semantic parsing (Wong and Mooney, 2006; Lu et al., 2008; Jones et al., 2012). The dataset consists of 880 pairs of natural language sentences and the corresponding tree-structured semantic representations. This dataset is annotated with eight languages. The original annotation of this dataset is English (Zelle and Mooney, 1996) and Jones et al. (2012) annotated the dataset with three more languages: German, Greek and Thai. Lu and Ng (2011) released the Chinese annotation and Susanto and Lu (2017b) annotated the corpus with three additional languages: Indonesian, Swedish and Farsi. In order to compare with previous work (Jones et al., 2012; Lu, 2015), we follow the standard splits with 600 instances for training and 280 instances for testing. To evaluate the performance, we follow the

---

[10] Empirically, we also tried multilayer perceptron but the bilinear model gives us better results.

Type	System/Model	English (en)		Thai (th)		German (de)		Greek (el)		Chinese (zh)		Indonesian (id)		Swedish (sv)		Farsi (fa)	
		Acc.	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*	*Acc.*	*F.*
Non-Neural	WASP	71.1	77.7	71.4	75.0	65.7	74.9	70.7	78.6	48.2	51.6	74.6	79.8	63.9	71.5	46.8	54.1
	HYBRIDTREE	76.8	81.0	73.6	76.7	62.1	68.5	69.3	74.6	56.1	58.4	66.4	72.8	61.4	70.5	51.8	58.6
	UBL	82.1	82.1	66.4	66.4	75.0	75.0	73.6	73.7	63.8	63.8	73.8	73.8	78.1	78.1	64.4	64.4
	TREETRANS	79.3	79.3	78.2	78.2	74.6	74.6	75.4	75.4	-	-	-	-	-	-	-	-
	RHT	86.8	86.8	80.7	80.7	75.7	75.7	79.3	79.3	76.1	76.1	75.0	75.0	79.3	79.3	73.9	73.9
Neural	SEQ2TREE†	84.5	-	71.9	-	70.3	-	73.1	-	73.3	-	80.7	-	80.8	-	70.5	-
	MSP-SINGLE†	83.5	-	72.1	-	69.3	-	74.2	-	74.9	-	79.8	-	77.5	-	72.2	-
	NEURAL HT (*J*=0)	87.9	87.9	82.1	82.1	75.7	75.7	81.1	81.1	76.8	76.8	76.1	76.1	81.1	81.1	75.0	75.0
	NEURAL HT (*J*=1)	88.6	88.6	84.6	84.6	76.8	76.8	79.6	79.6	75.4	75.4	78.6	78.6	82.9	82.9	76.1	76.1
	NEURAL HT (*J*=2)	**90.0**	**90.0**	82.1	82.1	73.9	73.9	80.7	80.7	81.1	81.1	81.8	81.8	83.9	83.9	74.6	74.6
Non-Neural	(This work) DEPHT	86.8	86.8	81.8	81.8	76.1	76.1	80.4	80.4	81.4	81.4	86.8	86.8	85.4	85.4	73.9	73.9
Non-Neural	(This work) DEPHT + embedding	87.5	87.5	83.9	83.9	75.0	75.0	81.1	81.1	81.4	81.4	87.5	87.5	87.1	87.1	73.6	73.6
Neural	(This work) DEPHT + NN	89.3	89.3	**86.7**	**86.7**	**78.2**	**78.2**	**82.9**	**82.9**	**82.9**	**82.9**	**88.7**	**88.7**	**87.3**	**87.3**	**77.9**	**77.9**

Table 3: Performance comparison with state-of-the-art models on GeoQuery dataset. († represents the system is using lambda-calculus expressions as meaning representations.)

standard evaluation procedure used in various previous works (Wong and Mooney, 2006; Lu et al., 2008; Jones et al., 2012; Lu, 2015) to construct the Prolog query from the tree-structured semantic representation using a standard and publicly available script. The queries are then used to retrieve the answers from the GeoQuery database, and we report accuracy and F_1 scores.

Hyperparameters We set the maximum depth c of the semantic tree to 20, following Lu (2015). The L_2 regularization coefficient is tuned from 0.01 to 0.05 using 5-fold cross-validation on the training set. The Polyglot (Al-Rfou et al., 2013) multilingual word embeddings[11] (with 64 dimensions) are used for all languages. We use L-BFGS (Liu and Nocedal, 1989) to optimize the DEPHT model until convergence and stochastic gradient descent (SGD) with a learning rate of 0.05 to optimize the neural DEPHT model. We implemented our neural component with the Torch7 library (Collobert et al., 2011). Our complete implementation is based on the StatNLP[12] structured prediction framework (Lu, 2017).

4.1 Baseline Systems

We run the released systems of several state-of-the-art semantic parsers, namely the WASP parser (Wong and Mooney, 2006), HYBRIDTREE model (Lu et al., 2008), UBL system (Kwiatkowski et al., 2010), *relaxed hybrid tree* (RHT) (Lu, 2015)[13], the sequence-to-tree (SEQ2TREE) model (Dong and Lapata, 2016), the *neural hybrid tree* (NEURAL HT) model (Susanto and Lu, 2017b), and the multilingual semantic parser (Susanto and Lu, 2017a) with single language (MSP-SINGLE) as input. The results for TREETRANS (Jones et al., 2012) are taken from their paper.

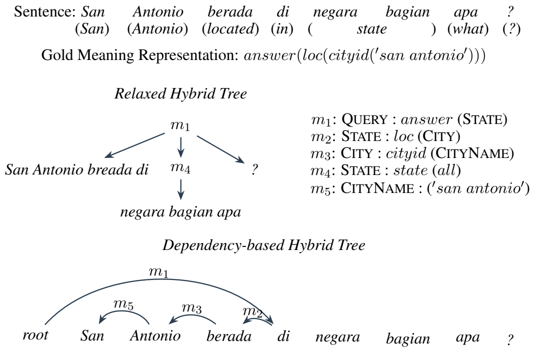

Figure 5: Example results from DEPHT and RHT on Indonesian.

4.2 Results and Discussion

Table 3 (top) shows the results of our dependency-based hybrid tree model compared with non-neural models which achieve state-of-the-art performance on the GeoQuery dataset. Our model DEPHT achieves competitive performance and outperforms the previous best system RHT on 6 languages. Improvements on the Indonesian dataset are particularly striking (+11.8 absolute points in F_1). We further investigated the outputs from both systems on Indonesian by doing error analysis. We found 40 instances that are incorrectly predicted by RHT are correctly predicted by DEPHT. We found that 77.5% of the errors are due to incorrect alignment between words and semantic units. Figure 5 shows an example of such errors where the relaxed hybrid tree fails to capture the correct alignment. We can see the question is asking "*What state is San Antonio located in?*". However, the natural language word order in Indone-

---

[11] The embeddings are fixed to avoid overfitting.

[12] https://gitlab.com/sutd_nlp/statnlp-core

[13] (Lu, 2015) is an extension of the original *relaxed hybrid tree* (Lu, 2014), which reports improved results.

	en	th	de	el	zh	id	sv	fa
DepHT basic	75.0	82.1	70.4	74.6	76.1	71.9	73.9	69.3
BASIC+HM feats.	80.7	**83.9**	**75.7**	79.2	81.1	85.0	81.1	72.5
BASIC+BOW feats.	**86.1**	83.2	73.9	**79.3**	**81.4**	**86.1**	**85.4**	**73.2**
DepHT	86.8	81.8	76.1	80.4	81.4	86.8	85.4	73.9

Table 4: F_1 scores of our model with different dependency features.

sian is different from English, where the phrase "*berada di*" that corresponds to m_2 (i.e., loc) appears between "*San Antonio*" (which corresponds to $m_5 - 'san\ antonio'$) and "*what*" (which corresponds to m_1 – $answer$). Such a structural non isomorphism issue between the sentence and the semantic tree makes the relaxed hybrid tree parser unable to produce a joint representation with valid word-semantics alignment. This issue makes the RHT model unable to predict the semantic unit m_2 (i.e., loc) as RHT has to align the words "*San Antonio*" which should be aligned to m_5 before aligning "*berada di*". However, m_5 has arity 0 and cannot have m_2 as its child. Thus, it would be impossible for the RHT model to predict such a meaning representation as output. In contrast, we can see that our dependency-based hybrid tree representation appears to be more flexible in handling such cases. The dependency between the two words "*di*" (*in*) and "*berada*" (*located*) is also well captured by the arc between them that is labeled with m_2. The error analysis reveals the flexibility of our joint representation in different languages in terms of the word ordering, indicating that the novel dependency-based joint representation is more robust and suffers less from language-specific characteristics associated with the data.

Effectiveness of dependency To investigate the helpfulness of the features defined over latent dependencies, we conduct ablation tests by removing the dependency-related features. Table 4 shows the performance of augmenting different dependency features in our DepHT model with basic features. Specifically, we investigate the performance of head word and modifier word features (HM) and also the bag-of-words features (BOW) that can be extracted based on dependencies. It can be observed that dependency features associated with the words are crucial for all languages, especially the BOW features.

Effectiveness of neural component The bottom part of Table 3 shows the performance comparison among models that involve neural networks. Our DepHT model with embeddings as features can outperform neural baselines across several languages (i.e., Chinese, Indonesian and Swedish). From the table, we can see the neural component is effective, which consistently gives better results than DepHT and the approach that uses word embedding features only. Susanto and Lu (2017b) presented the Neural HT model with different window size J for their multilayer perceptron. Their performance will differ with different window sizes, which need to be tuned for each language. In our neural component, we do not require such a language-specific hyperparameter, yet our neural approach consistently achieves the highest performance on 7 out of 8 languages compared with all previous approaches. As both the embeddings and the neural component are defined on the dependency arcs, the superior results also reveal the effectiveness of our dependency-based hybrid tree representation.

5 Conclusions and Future Work

In this work, we present a novel *dependency-based hybrid tree* model for semantic parsing. The model captures the underlying semantic information of a sentence as latent dependencies between the natural language words. We develop an efficient algorithm for exact inference based on dynamic-programming. Extensive experiments on benchmark dataset across 8 different languages demonstrate the effectiveness of our newly proposed representation for semantic parsing.

Future work includes exploring alternative approaches such as transition-based methods (Nivre et al., 2006; Chen and Manning, 2014) for semantic parsing with latent dependencies, applying our dependency-based hybrid trees on other types of logical representations (e.g., lambda calculus expressions and SQL (Finegan-Dollak et al., 2018)) as well as multilingual semantic parsing (Jie and Lu, 2014; Susanto and Lu, 2017a).

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments on this work. We would also like to thank Yanyan Zou for helping us with running the experiments for baseline systems. This work is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156, and is partially supported by project 61472191 under the National Natural Science Foundation of China.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of CoNLL*.

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

Paul Bédaride and Claire Gardent. 2011. Deep semantics for dependency structures. In *Proceedings of CICLing*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of ACL*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of ACL*.

Philipp Cimiano. 2009. Flexible semantic composition with dudes. In *Proceedings of ICCS*.

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011. Torch7: A matlab-like environment for machine learning. In *Proceedings of BigLearn, NIPS workshop*.

Ralph Debusmann, Denys Duchier, Alexander Koller, Marco Kuhlmann, Gert Smolka, and Stefan Thater. 2004. A relational syntax-semantics interface based on dependency grammar. In *Proceedings of COLING*.

Trinh-Minh-Tri Do and Thierry Artieres. 2010. Neural conditional random fields. In *Proceedings of AISTAT*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of ACL-IJCNLP*.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in probabilistic and other parsing technologies*, pages 29–61. Springer.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of ACL*.

Matt Gardner and Jayant Krishnamurthy. 2017. Open-vocabulary semantic parsing with both distributional statistics and formal knowledge. In *Proceedings of AAAI*.

Matthew R Gormley. 2015. *Graphical Models with Structured Factors, Neural Factors, and Approximation-Aware Training*. Ph.D. thesis.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*.

Zhanming Jie and Wei Lu. 2014. Multilingual semantic parsing: Parsing multiple languages into semantic representations. In *Proceedings of COLING*.

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of ACL*.

Rohit J Kate and Raymond J Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of COLING/ACL*.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of AAAI*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of EMNLP*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *Technical Report arXiv*.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of NNACL*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.

Wei Lu. 2014. Semantic parsing with relaxed hybrid trees. In *Proceedings of EMNLP*.

Wei Lu. 2015. Constrained semantic forests for improved discriminative semantic parsing. In *Proceedings of ACL*.

Wei Lu. 2017. A unified framework for structured prediction: From theory to practice. In *Proceedings of EMNLP (Tutorial)*.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of EMNLP*.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of EMNLP*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of COLING*.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of ACL*.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. 2005. Conditional random fields for object recognition. In *Proceedings of NIPS*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of EMNLP*.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast (er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of EMNLP*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016. Getting more out of syntax with props. *arXiv preprint arXiv:1603.01648*.

Mark Steedman. 1996. Surface structure and interpretation.

Raymond Hendy Susanto and Wei Lu. 2017a. Neural architectures for multilingual semantic parsing. In *Proceedings of ACL*.

Raymond Hendy Susanto and Wei Lu. 2017b. Semantic parsing with neural hybrid trees. In *Proceedings of AAAI*.

Ivan Titov and Alexandre Klementiev. 2011. A bayesian model for unsupervised semantic parsing. In *Proceedings of ACL*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of ACL-IJCNLP*, pages 1332–1342.

Yuk Wah Wong and Raymond J Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of NAACL*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of AAAI*.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.

Yanyan Zou and Wei Lu. 2018. Learning cross-lingual distributed logical representations for semantic parsing. In *Proceedings of ACL*.

Policy Shaping and Generalized Update Equations for Semantic Parsing from Denotations

Dipendra Misra[⋆], **Ming-Wei Chang**[†], **Xiaodong He**[◇], **Wen-tau Yih**[‡]
[⋆]Cornell University, [†]Google AI Language, [◇]JD AI Research
[‡]Allen Institute for Artificial Intelligence
`dkm@cs.cornell.edu`, `mingweichang@google.com`
`xiaodong.he@jd.com`, `scottyih@allenai.org`

Abstract

Semantic parsing from denotations faces two key challenges in model training: (1) given only the denotations (e.g., answers), search for good candidate semantic parses, and (2) choose the best model update algorithm. We propose effective and general solutions to each of them. Using *policy shaping*, we bias the search procedure towards semantic parses that are more compatible to the text, which provide better supervision signals for training. In addition, we propose an update equation that generalizes three different families of learning algorithms, which enables fast model exploration. When experimented on a recently proposed sequential question answering dataset, our framework leads to a new state-of-the-art model that outperforms previous work by 5.0% absolute on exact match accuracy.

1 Introduction

Semantic parsing from denotations (SpFD) is the problem of mapping text to executable formal representations (or *program*) in a situated environment and executing them to generate denotations (or *answer*), in the absence of access to correct representations. Several problems have been handled within this framework, including question answering (Berant et al., 2013; Iyyer et al., 2017) and instructions for robots (Artzi and Zettlemoyer, 2013; Misra et al., 2015).

Consider the example in Figure 1. Given the question and a table environment, a semantic parser maps the question to an executable program, in this case a SQL query, and then executes the query on the environment to generate the answer *England*. In the SpFD setting, the training data does not contain the correct programs. Thus, the existing learning approaches for SpFD perform two steps for every training example, a search step that explores the space of programs and finds suitable candidates, and an update step that uses these programs to update the model. Figure 2 shows the two step training procedure for the above example.

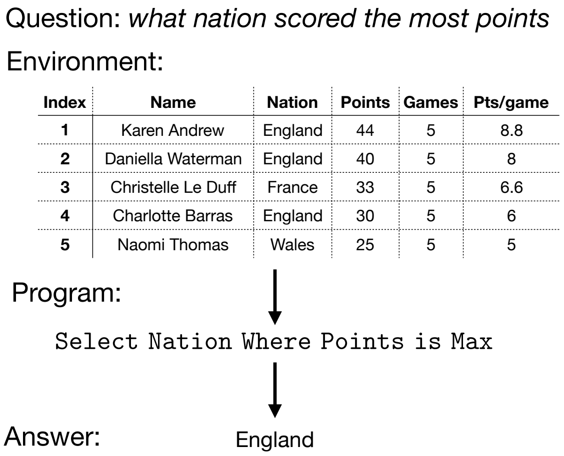

Question: *what nation scored the most points*

Environment:

Index	Name	Nation	Points	Games	Pts/game
1	Karen Andrew	England	44	5	8.8
2	Daniella Waterman	England	40	5	8
3	Christelle Le Duff	France	33	5	6.6
4	Charlotte Barras	England	30	5	6
5	Naomi Thomas	Wales	25	5	5

Program:

`Select Nation Where Points is Max`

Answer: England

Figure 1: An example of semantic parsing from denotations. Given the table environment, map the question to an executable program that evaluates to the answer.

In this paper, we address two key challenges in model training for SpFD by proposing a novel learning framework, improving both the *search* and *update* steps. The first challenge, the existence of *spurious programs*, lies in the search step. More specifically, while the success of the search step relies on its ability to find programs that are semantically correct, we can only verify if the program can generate correct answers, given that no gold programs are presented. The search step is complicated by *spurious programs*, which happen to evaluate to the correct answer but do not represent accurately the meaning of the natural language question. For example, for the environment in Figure 1, the program `Select Nation Where Name = Karen Andrew` is spurious. Selecting spurious programs as positive examples can greatly affect the performance of semantic parsers as these programs generally do not gen-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2442–2452
Brussels, Belgium, October 31 - November 4, 2018.

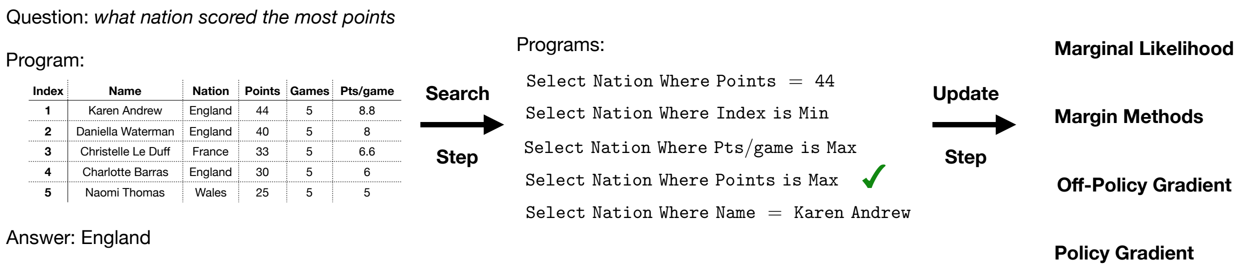

Index	Name	Nation	Points	Games	Pts/game
1	Karen Andrew	England	44	5	8.8
2	Daniella Waterman	England	40	5	8
3	Christelle Le Duff	France	33	5	6.6
4	Charlotte Barras	England	30	5	6
5	Naomi Thomas	Wales	25	5	5

Figure 2: An example of semantic parsing from denotation. Given the question and the table environment, there are several programs which are spurious.

eralize to unseen questions and environments.

The second challenge, *choosing a learning algorithm*, lies in the update step. Because of the unique *indirect supervision* setting of SpFD, the quality of the learned semantic parser is dictated by the choice of how to update the model parameters, often determined empirically. As a result, several families of learning methods, including maximum marginal likelihood, reinforcement learning and margin based methods have been used. How to effectively explore different model choices could be crucial in practice.

Our contributions in this work are twofold. To address the first challenge, we propose a policy shaping (Griffith et al., 2013) method that incorporates simple, lightweight domain knowledge, such as a small set of lexical pairs of tokens in the question and program, in the form of a *critique policy* (§ 3). This helps bias the search towards the correct program, an important step to improve supervision signals, which benefits learning regardless of the choice of algorithm. To address the second challenge, we prove that the parameter update step in several algorithms are similar and can be viewed as special cases of a generalized update equation (§ 4). The equation contains two variable terms that govern the update behavior. Changing these two terms effectively defines an infinite class of learning algorithms where different values lead to significantly different results. We study this effect and propose a novel learning framework that improves over existing methods.

We evaluate our methods using the sequential question answering (SQA) dataset (Iyyer et al., 2017), and show that our proposed improvements to the search and update steps consistently enhance existing approaches. The proposed algorithm achieves new state-of-the-art and outperforms existing parsers by 5.0%.

2 Background

We give a formal problem definition of the semantic parsing task, followed by the general learning framework for solving it.

2.1 The Semantic Parsing Task

The problem discussed in this paper can be formally defined as follows. Let $\mathcal{X}$ be the set of all possible questions, $\mathcal{Y}$ programs (e.g., SQL-like queries), $\mathcal{T}$ tables (i.e., the structured data in this work) and $\mathcal{Z}$ answers. We further assume access to an executor $\Phi : \mathcal{Y} \times \mathcal{T} \to \mathcal{Z}$, that given a program $y \in \mathcal{Y}$ and a table $t \in \mathcal{T}$, generates an answer $\Phi(y, t) \in \mathcal{Z}$. We assume that the executor and all tables are deterministic and the executor can be called as many times as possible. To facilitate discussion in the following sections, we define an environment function $e_t : \mathcal{Y} \to \mathcal{Z}$, by applying the executor to the program as $e_t(y) = \Phi(y, t)$.

Given a question x and an environment e_t, our aim is to *generate* a program $y^* \in \mathcal{Y}$ and then execute it to produce the answer $e_t(y^*)$. Assume that for any $y \in \mathcal{Y}$, the score of y being a correct program for x is $\texttt{score}_\theta(y, x, t)$, parameterized by θ. The *inference* task is thus:

$$y^* = \arg\max_{y \in \mathcal{Y}} \texttt{score}_\theta(y, x, t) \tag{1}$$

As the size of $\mathcal{Y}$ is exponential to the length of the program, a generic *search* procedure is typically employed for Eq. (1), as efficient dynamic algorithms typically do not exist. These search procedures generally maintain a beam of program states sorted according to some scoring function, where each program state represents an incomplete program. The search then generates a new program state from an existing state by performing an action. Each action adds a set of tokens (e.g., `Nation`) and keyword (e.g., `Select`) to a program state. For example, in order to generate the program in Figure 1, the DynSP parser (Iyyer et al., 2017) will take the first action as adding the SQL expression `Select Nation`. Notice that $\texttt{score}_\theta$ can be used in either probabilistic or non-probabilistic models. For probabilistic models, we assume that it is a Boltzmann policy, meaning that $p_\theta(y \mid x, t) \propto \exp\{\texttt{score}_\theta(y, x, t)\}$.

2.2 Learning

Learning a semantic parser is equivalent to learning the parameters θ in the scoring function, which is a structured learning problem, due to the large, structured output space $\mathcal{Y}$. Structured learning algorithms generally consist of two major components: *search* and *update*. When the gold programs are available during training, the search procedure finds a set of high-scoring incorrect programs. These programs are used by the update step to derive loss for updating parameters. For example, these programs are used for approximating the partition-function in maximum-likelihood objective (Liang et al., 2011) and finding set of programs causing margin violation in margin based methods (Daumé III and Marcu, 2005). Depending on the exact algorithm being used, these two components are not necessarily separated into isolated steps. For instance, parameters can be updated in the middle of search (e.g., Huang et al., 2012).

For learning semantic parsers from denotations, where we assume only answers are available in a training set $\{(x_i, t_i, z_i)\}_{i=1}^{N}$ of N examples, the basic construction of the learning algorithms remains the same. However, the problems that search needs to handle in SpFD is more challenging. In addition to finding a set of high-scoring incorrect programs, the search procedure also needs to *guess* the correct program(s) evaluating to the gold answer z_i. This problem is further complicated by the presence of spurious programs, which generate the correct answer but are semantically incompatible with the question. For example, although all programs in Figure 2 evaluate to the same answer, only one of them is correct. The issue of the spurious programs also affects the design of model update. For instance, maximum marginal likelihood methods treat all the programs that evaluate to the gold answer equally, while maximum margin reward networks use model score to break tie and pick one of the programs as the correct reference.

3 Addressing Spurious Programs: Policy Shaping

Given a training example (x, t, z), the aim of the search step is to find a set $\mathcal{K}(\mathtt{x}, \mathtt{t}, \mathtt{z})$ of programs consisting of correct programs that evaluate to z and high-scoring incorrect programs. The search step should avoid picking up spurious programs for learning since such programs typically do not generalize. For example, in Figure 2, the spurious program `Select Nation Where Index is Min` will evaluate to an incorrect answer if the indices of the first two rows are swapped[1]. This problem is challenging since among the programs that evaluate to the correct answer, most of them are spurious.

The search step can be viewed as following an exploration policy $b_\theta(y|x, t, z)$ to explore the set of programs $\mathcal{Y}$. This exploration is often performed by beam search and at each step, we either sample from b_θ or take the top scoring programs. The set $\mathcal{K}(\mathtt{x}, \mathtt{t}, \mathtt{z})$ is then used by the update step for parameter update. Most search strategies use an exploration policy which is based on the score function, for example $b_\theta(y|x, t, z) \propto \exp\{\texttt{score}_\theta(y, t)\}$. However, this approach can suffer from a *divergence* phenomenon whereby the score of spurious programs picked up by the search in the first epoch increases, making it more likely for the search to pick them up in the future. Such *divergence* issues are common with latent-variable learning and often require careful initialization to overcome (Rose, 1998). Unfortunately such initialization schemes are not applicable for deep neural networks which form the model of most successful semantic parsers today (Jia and Liang, 2016; Misra and Artzi, 2016; Iyyer et al., 2017). Prior work, such as ϵ-greedy exploration (Guu et al., 2017), has reduced the severity of this problem by introducing random noise in the search procedure to avoid saturating the search on high-scoring spurious programs. However, random noise need not bias the search towards the correct program(s). In this paper, we introduce a simple policy-shaping method to guide the search. This approach allows incorporating prior knowledge in the exploration policy and can bias the search away from spurious programs.

---

[1] This transformation preserves the answer of the question.

Algorithm 1 Learning a semantic parser from denotation using generalized updates.

```
Input: Training set $\{(x_i, t_i, z_i\}_{i=1}^{N}$ (see Section 2), learning
    rate $\mu$ and stopping epoch $\tilde{T}$(see Section 4).
Definitions: $\texttt{score}_\theta(y, x, t)$ is a semantic parsing model
    parameterized by $\theta$. $p_s(y \mid x, t)$ is the policy used for
    exploration and search$(\theta, x, t, z, p_s)$ generates candi-
    date programs for updating parameters (see Section 3).
    $\Delta$ is the generalized update (see Section 4).
Output: Model parameters $\theta$.
1: » Iterate over the training data.
2: for $t = 1$ to $T$, $i = 1$ to $N$ do
3:     » Find candidate programs using the shaped policy.
4:     $\mathcal{K} = \text{search}(\theta, x_i, t_i, z_i, p_s)$
5:     » Compute generalized gradient updates
6:     $\theta = \theta + \mu\Delta(\mathcal{K})$
7: return $\theta$
```

Policy Shaping Policy shaping is a method to introduce prior knowledge into a policy (Griffith et al., 2013). Formally, let the current behavior policy be $b_\theta(y|x, t, z)$ and a predefined critique policy, the prior knowledge, be $p_c(y|x, t)$. Policy shaping defines a new *shaped behavior policy* $p_b(y|x, t)$ given by:

$$p_b(y|x, t) = \frac{b_\theta(y|x, t, z)p_c(y|x, t)}{\sum_{y' \in \mathcal{Y}} b_\theta(y'|x, t, z)p_c(y'|x, t)}. \quad (2)$$

Using the shaped policy for exploration biases the search towards the critique policy's preference. We next describe a simple critique policy that we use in this paper.

Lexical Policy Shaping We qualitatively observed that correct programs often contains tokens which are also present in the question. For example, the correct program in Figure 2 contains the token *Points*, which is also present in the question. We therefore, define a simple surface form similarity feature $\texttt{match}(x, y)$ that computes the ratio of number of non-keyword tokens in the program y that are also present in the question x.

However, surface-form similarity is often not enough. For example, both the first and fourth program in Figure 2 contain the token *Points* but only the fourth program is correct. Therefore, we also use a simple co-occurrence feature that triggers on frequently co-occurring pairs of tokens in the program and instruction. For example, the token *most* is highly likely to co-occur with a correct program containing the keyword `Max`. This happens for the example in Figure 2. Similarly the token *not* may co-occur with the keyword `NotEqual`. We assume access to a lexicon $\Lambda = \{(w_j, \omega_j)\}_{j=1}^{k}$ containing k lexical pairs of tokens and keywords. Each lexical pair (w, ω) maps the token w in a text to a keyword ω in a program. For a given program y and question x, we define a co-occurrence score as $\texttt{co\_occur}(y, x) = \sum_{(w,\omega)\in\Lambda} \mathbb{1}\{w \in x \wedge \omega \in y\}\}$. We define critique score $\texttt{critique}(y, x)$ as the sum of the `match` and `co_occur` scores. The critique policy is given by:

$$p_c(y|x, t) \propto \exp\left(\eta * \texttt{critique}(y, x)\right), \quad (3)$$

where η is a single scalar hyper-parameter denoting the confidence in the critique policy.

4 Addressing Update Strategy Selection: Generalized Update Equation

Given the set of programs generated by the search step, one can use many objectives to update the parameters. For example, previous work have utilized maximum marginal likelihood (Krishnamurthy et al., 2017; Guu et al., 2017), reinforcement learning (Zhong et al., 2017; Guu et al., 2017) and margin based methods (Iyyer et al., 2017). It could be difficult to choose the suitable algorithm from these options.

In this section, we propose a principle and general update equation such that previous update algorithms can be considered as special cases to this equation. Having a general update is important for the following reasons. First, it allows us to understand existing algorithms better by examining their basic properties. Second, the generalized update equation also makes it easy to implement and experiment with various different algorithms. Moreover, it provides a framework that enables the development of new variations or extensions of existing learning methods.

In the following, we describe how the commonly used algorithms are in fact very similar – their update rules can all be viewed as special cases of the proposed generalized update equation. Algorithm 1 shows the meta-learning framework. For every training example, we first find a set of candidates using an exploration policy (line 4). We use the program candidates to update the parameters (line 6).

4.1 Commonly Used Learning Algorithms

We briefly describe three algorithms: *maximum marginalized likelihood*, *policy gradient* and *maximum margin reward*.

Maximum Marginalized Likelihood The maximum marginalized likelihood method maximizes the log-likelihood of the training data by marginalizing over the set of programs.

$$\begin{aligned} J_{MML} &= \log p(z_i|x_i, t_i) \\ &= \log \sum_{y \in \mathcal{Y}} p(z_i|y, t_i) p(y|x_i, t_i) \end{aligned} \quad (4)$$

Because an answer is deterministically computed given a program and a table, we define $p(z \mid y, t)$ as 1 or 0 depending upon whether the y evaluates to z given t, or not. Let $\texttt{Gen(z,t)} \subseteq \mathcal{Y}$ be the set of compatible programs that evaluate to z given the table t. The objective can then be expressed as:

$$J_{MML} = \log \sum_{y \in \texttt{Gen}(\texttt{z}_\texttt{i}, \texttt{t}_\texttt{i})} p(y|x_i, t_i) \quad (5)$$

In practice, the summation over $\texttt{Gen}(.)$ is approximated by only using the compatible programs in the set $\mathcal{K}$ generated by the search step.

Policy Gradient Methods Most reinforcement learning approaches for semantic parsing assume access to a reward function $R : \mathcal{Y} \times \mathcal{X} \times \mathcal{Z} \rightarrow \mathbb{R}$, giving a scalar reward $R(y, z)$ for a given program y and the correct answer z.[2] We can further assume without loss of generality that the reward is always in $[0, 1]$. Reinforcement learning approaches maximize the expected reward J_{RL}:

$$J_{RL} = \sum_{y \in \mathcal{Y}} p(y|x_i, t_i) R(y, z_i) \quad (6)$$

J_{RL} is hard to approximate using numerical integration since the reward for all programs may not be known a priori. Policy gradient methods solve this by approximating the derivative using a sample from the policy. When the search space is large, the policy may fail to sample a correct program, which can greatly slow down the learning. Therefore, off-policy methods are sometimes introduced to bias the sampling towards high-reward yielding programs. In those methods, an additional exploration policy $u(y|x_i, t_i, z_i)$ is used to improve sampling. Importance weights are used to make the gradient unbiased (see Appendix for derivation).

Maximum Margin Reward For every training example (x_i, t_i, z_i), the maximum margin reward method finds the highest scoring program y_i that evaluates to z_i, as the *reference* program, from the set $\mathcal{K}$ of programs generated by the search. With a margin function $\delta : \mathcal{Y} \times \mathcal{Y} \times \mathcal{Z} \rightarrow \mathbb{R}$ and reference program y, the set of programs $\mathcal{V}$ that violate the margin constraint can thus be defined as:

$$\begin{aligned} \mathcal{V} = \{y' \mid y' \in \mathcal{Y} \textbf{ and } \texttt{score}_\theta(y, x, t) \\ \leq \texttt{score}_\theta(y', x, t) + \delta(y, y', z)\}, \end{aligned} \quad (7)$$

where $\delta(y, y', z) = R(y, z) - R(y', z)$. Similarly, the program that most violates the constraint can be written as:

$$\begin{aligned} \bar{y} = \arg\max_{y' \in \mathcal{Y}} \{\texttt{score}_\theta(y', x, t) + \delta(y, y', z) \\ -\texttt{score}_\theta(y, x, t)\} \end{aligned} \quad (8)$$

The most-violation margin objective (negative margin loss) is thus defined as:

$$\begin{aligned} J_{MMR} = & -\max\{0, \texttt{score}_\theta(\bar{y}, x_i, t_i) \\ & -\texttt{score}_\theta(y_i, x_i, t_i) + \delta(y_i, \bar{y}, z_i)\} \end{aligned}$$

Unlike the previous two learning algorithms, margin methods only update the score of the reference program and the program that violates the margin.

4.2 Generalized Update Equation

Although the algorithms described in §4.1 seem very different on the surface, the gradients of their loss functions can in fact be described in the same generalized form, given in Eq. (9)[3]. In addition to the gradient of the model scoring function, this equation has two variable terms, $w(\cdot)$, $q(\cdot)$. We call the first term $w(y, x, t, z)$ *intensity*, which is a positive scalar value and the second term $q(y|x, t)$ the *competing distribution*, which is a probability distribution over programs. Varying them makes the equation equivalent to the update rule of the algorithms we discussed, as shown in Table 1. We also consider meritocratic update policy which uses a hyperparameter β to sharpen or smooth the intensity of maximum marginal likelihood (Guu et al., 2017).

Intuitively, $w(y, x, t, z)$ defines the positive part of the update equation, which defines how aggressively the update favors program y. Likewise, $q(y|x, t)$ defines the negative part of the learning

---

[2] This is essentially a contextual bandit setting. Guu et al. (2017) also used this setting. A general reinforcement learning setting requires taking a sequence of actions and receiving a reward for each action. For example, a program can be viewed as a sequence of parsing actions, where each action can get a reward. We do not consider the general setting here.

[3] See Appendix for the detailed derivation.

Generalized Update Equation:

$$\Delta(\mathcal{K}) = \sum_{y \in \mathcal{K}} w(y, x, t, z) \left(\nabla_\theta \texttt{score}_\theta(y, x, t) - \sum_{y' \in \mathcal{Y}} q(y'|x, t) \nabla_\theta \texttt{score}_\theta(y', x, t) \right) \tag{9}$$

Learning Algorithm	**Intensity** $w(y, x, t, z)$	**Competing Distribution** $q(y\|x, t)$
Maximum Margin Likelihood	$\frac{p(z\|y)p(y\|x)}{\sum_{y'} p(z\|y')p(y'\|x)}$	$p(y\|x)$
Meritocratic(β)	$\frac{(p(z\|y)p(y\|x))^\beta}{\sum_{y'} (p(z\|y')p(y'\|x))^\beta}$	$p(y\|x)$
REINFORCE	$\mathbb{1}\{y = \hat{y}\}R(y, z)$	$p(y\|x)$
Off-Policy Policy Gradient	$\mathbb{1}\{y = \hat{y}\}\, R(y, z)\, \frac{p(y\|x)}{u(y\|x,z)}$	$p(y\|x)$
Maximum Margin Reward (MMR)	$\mathbb{1}\{y = y^*\}$	$\mathbb{1}\{y = \bar{y}\}$
Maximum Margin Avg. Violation Reward (MAVER)	$\mathbb{1}\{y = y^*\}$	$1/\|\mathcal{V}\|\mathbb{1}\{y \in \mathcal{V}\}$

Table 1: Parameter updates for various learning algorithms are special cases of Eq. (9), with different choices of intensity w and competing distribution q. We do not show dependence upon table t for brevity. For off-policy policy gradient, u is the exploration policy. For margin methods, y^* is the reference program (see §4.1), $\mathcal{V}$ is the set of programs that violate the margin constraint (cf. Eq. (7)) and $\bar{y}$ is the most violating program (cf. Eq. (8)). For REINFORCE, $\hat{y}$ is sampled from $\mathcal{K}$ using $p(.)$ whereas for Off-Policy Policy Gradient, $\hat{y}$ is sampled using $u(.)$.

algorithm, namely how aggressively the update penalizes the members of the program set.

The generalized update equation provides a tool for better understanding individual algorithm, and helps shed some light on when a particular method may perform better.

Intensity versus Search Quality In SpFD, the effectiveness of the algorithms for SpFD is closely related to the quality of the search results given that the gold program is not available. Intuitively, if the search quality is good, the update algorithm could be aggressive on updating the model parameters. When the search quality is poor, the algorithm should be conservative.

The intensity $w(\cdot)$ is closely related to the aggressiveness of the algorithm. For example, the maximum marginal likelihood is less aggressive given that it produces a non-zero intensity over all programs in the program set $\mathcal{K}$ that evaluate to the correct answer. The intensity for a particular correct program y is proportional to its probability $p(y|x, t)$. Further, meritocratic update becomes more aggressive as β becomes larger.

In contrast, REINFORCE and maximum margin reward both have a non-zero intensity only on a single program in $\mathcal{K}$. This value is 1.0 for maximum margin reward, while for reinforcement learning, this value is the reward. Maximum margin reward therefore updates most aggressively in favor of its selection while maximum marginal likelihood tends to hedge its bet. Therefore, the maximum margin methods should benefit the most when the search quality improves.

Stability The general equation also allows us to investigate the stability of a model update algorithm. In general, the variance of update direction can be high, hence less stable, if the model update algorithm has peaky competing distribution, or it puts all of its intensity on a single program. For example, REINFORCE only samples one program and puts non-zero intensity only on that program, so it could be unstable depending on the sampling results.

The competing distribution affects the stability of the algorithm. For example, maximum margin reward penalizes only the most violating program and is benign to other incorrect programs. Therefore, the MMR algorithm could be unstable during training.

New Model Update Algorithm The general equation provides a framework that enables the development of new variations or extensions of existing learning methods. For example, in order to improve the stability of the MMR algorithm, we propose a simple variant of maximum margin reward, which penalizes all violating programs instead of only the most violating one. We call this approach *maximum margin average violation reward* (MAVER), which is included in Table 1 as well. Given that MAVER effectively considers more negative examples during each update, we expect that it is more stable compared to the MMR algorithm.

5 Experiments

We describe the setup in §5.1 and results in §5.2.

5.1 Setup

Dataset We use the sequential question answering (SQA) dataset (Iyyer et al., 2017) for our experiments. SQA contains 6,066 sequences and each sequence contains up to 3 questions, with 17,553 questions in total. The data is partitioned into training (83%) and test (17%) splits. We use 4/5 of the original train split as our training set and the remaining 1/5 as the dev set. We evaluate using exact match on answer. Previous state-of-the-art result on the SQA dataset is 44.7% accuracy, using maximum margin reward learning.

Semantic Parser Our semantic parser is based on DynSP (Iyyer et al., 2017), which contains a set of SQL actions, such as adding a clause (e.g., `Select Column`) or adding an operator (e.g., `Max`). Each action has an associated neural network module that generates the score for the action based on the instruction, the table and the list of past actions. The score of the entire program is given by the sum of scores of all actions.

We modified DynSP to improve its representational capacity. We refer to the new parser as DynSP++. Most notably, we included new features and introduced two additional parser actions. See Appendix 8.2 for more details. While these improvements help us achieve state-of-the-art results, the majority of the gain comes from the learning contributions described in this paper.

Hyperparameters For each experiment, we train the model for 30 epochs. We find the optimal stopping epoch by evaluating the model on the dev set. We then train on train+dev set till the stopping epoch and evaluate the model on the held-out test set. Model parameters are trained using stochastic gradient descent with learning rate of 0.1. We set the hyperparameter η for policy shaping to 5. All hyperparameters were tuned on the dev set. We use 40 lexical pairs for defining the `co-occur` score. We used common English superlatives (e.g., *highest*, *most*) and comparators (e.g., *more*, *larger*) and did not fit the lexical pairs based on the dataset.

Given the model parameter θ, we use a base exploration policy defined in (Iyyer et al., 2017). This exploration policy is given by $b_\theta(y \mid x, t, z) \propto \exp(\lambda \cdot R(y, z) + \mathtt{score}_\theta(y, \theta, z))$. $R(y, z)$ is the reward function of the incomplete program y, given the answer z. We use a reward function $R(y, z)$ given by the Jaccard similarity of the gold answer z and the answer generated by the program y. The value of λ is set to infinity, which essentially is equivalent to sorting the programs based on the reward and using the current model score for tie breaking. Further, we prune all syntactically invalid programs. For more details, we refer the reader to (Iyyer et al., 2017).

5.2 Results

Table 2 contains the dev and test results when using our algorithm on the SQA dataset. We observe that margin based methods perform better than maximum likelihood methods and policy gradient in our experiment. Policy shaping in general improves the performance across different algorithms. Our best test results outperform previous SOTA by 5.0%.

Policy Gradient vs Off-Policy Gradient REINFORCE, a simple policy gradient method, achieved extremely poor performance. This likely due to the problem of exploration and having to sample from a large space of programs. This is further corroborated from observing the much superior performance of off-policy policy gradient methods. Thus, the sampling policy is an important factor to consider for policy gradient methods.

The Effect of Policy Shaping We observe that the improvement due to policy shaping is 6.0% on the SQA dataset for MAVER and only 1.3% for maximum marginal likelihood. We also observe that as β increases, the improvement due to policy shaping for meritocratic update increases. This supports our hypothesis that aggressive updates of margin based methods is beneficial when the search method is more accurate as compared to maximum marginal likelihood which hedges its bet between all programs that evaluate to the right answer.

Stability of MMR In Section 4, the general update equation helps us point out that MMR could be unstable due to the peaky competing distribution. MAVER was proposed to increase the stability of the algorithm. To measure stability, we cal-

Algorithm	Dev		Test	
	w.o. Shaping	w. Shaping	w.o. Shaping	w. Shaping
Maximum Margin Likelihood	33.2	32.5	31.0	32.3
Meritocratic ($\beta = 0$)	27.1	28.1	31.3	30.1
Meritocratic ($\beta = 0.5$)	28.3	28.7	31.7	32.0
Meritocratic ($\beta = \infty$)	39.3	41.6	41.6	45.2
REINFORCE	10.2	11.8	2.4	4.0
Off-Policy Policy Gradient	36.6	38.6	42.6	44.1
MMR	38.4	40.7	43.2	46.9
MAVER	39.6	44.1	43.7	**49.7**

Table 2: Experimental results on different model update algorithms, with and without policy shaping.

w	q	Dev
MMR	MML	**41.9**
Off-Policy Policy Gradient	MMR	37.0
MMR	MMR	40.7

Table 3: The dev set results on the new variations of the update algorithms.

culate the mean absolute difference of the development set accuracy between successive epochs during training, as it indicates how much an algorithm's performance fluctuates during training. With this metric, we found mean difference for MAVER is 0.57% where the mean difference for MMR is 0.9%. This indicates that MAVER is in fact more stable than MMR.

Other variations We also analyze other possible novel learning algorithms that are made possible due to generalized update equations. Table 3 reports development results using these algorithms. By mixing different intensity scalars and competing distribution from different algorithms, we can create new variations of the model update algorithm. In Table 3, we show that by mixing the MMR's intensity and MML's competing distribution, we can create an algorithm that outperform MMR on the development set.

Policy Shaping helps against Spurious Programs In order to better understand if policy shaping helps bias the search away from spurious programs, we analyze 100 training examples. We look at the highest scoring program in the beam at the end of training using MAVER. Without policy shaping, we found that 53 programs were spurious while using policy shaping this number came down to 23. We list few examples of spurious program errors corrected by policy shaping in Table 4.

Policy Shaping vs Model Shaping Critique policy contains useful information that can bias the search away from spurious programs. Therefore, one can also consider making the critique policy as part of the model. We call this model shaping. We define our model to be the shaped policy and train and test using the new model. Using MAVER updates, we found that the dev accuracy dropped to 37.1%. We conjecture that the strong prior in the critique policy can hinder generalization in model shaping.

6 Related Work

Semantic Parsing from Denotation Mapping natural language text to formal meaning representation was first studied by Montague (1970). Early work on learning semantic parsers rely on labeled formal representations as the supervision signals (Zettlemoyer and Collins, 2005, 2007; Zelle and Mooney, 1993). However, because getting access to gold formal representation generally requires expensive annotations by an expert, distant supervision approaches, where semantic parsers are learned from denotation only, have become the main learning paradigm (e.g., Clarke et al., 2010; Liang et al., 2011; Artzi and Zettlemoyer, 2013; Berant et al., 2013; Iyyer et al., 2017; Krishnamurthy et al., 2017). Guu et al. (2017) studied the problem of spurious programs and considered adding noise to diversify the search procedure and introduced meritocratic updates.

Reinforcement Learning Algorithms Reinforcement learning algorithms have been applied to various NLP problems including dialogue (Li et al., 2016), text-based games (Narasimhan et al., 2015), information extraction (Narasimhan et al., 2016), coreference resolution (Clark and Man-

Question	without policy shaping	with policy shaping
"of these teams, which had more than 21 losses?"	`SELECT Club WHERE Losses = ROW 15`	`SELECT Club WHERE Losses > 21`
"of the remaining, which earned the most bronze medals?"	`SELECT Nation WHERE Rank = ROW 1`	`FollowUp WHERE Bronze is Max`
"of those competitors from germany, which was not paul sievert?"	`SELECT Name WHERE Time (hand) = ROW 3`	`FollowUp WHERE Name != ROW 5`

Table 4: Training examples and the highest ranked program in the beam search, scored according to the shaped policy, after training with MAVER. Using policy shaping, we can recover from failures due to spurious programs in the search step for these examples.

ning, 2016), semantic parsing (Guu et al., 2017) and instruction following (Misra et al., 2017). Guu et al. (2017) show that policy gradient methods underperform maximum marginal likelihood approaches. Our result on the SQA dataset supports their observation. However, we show that using off-policy sampling, policy gradient methods can provide superior performance to maximum marginal likelihood methods.

Margin-based Learning Margin-based methods have been considered in the context of SVM learning. In the NLP literature, margin based learning has been applied to parsing (Taskar et al., 2004; McDonald et al., 2005), text classification (Taskar et al., 2003), machine translation (Watanabe et al., 2007) and semantic parsing (Iyyer et al., 2017). Kummerfeld et al. (2015) found that max-margin based methods generally outperform likelihood maximization on a range of tasks. Previous work have studied connections between margin based method and likelihood maximization for supervised learning setting. We show them as special cases of our unified update equation for distant supervision learning. Similar to this work, Lee et al. (2016) also found that in the context of supervised learning, margin-based algorithms which update all violated examples perform better than the one that only updates the most violated example.

Latent Variable Modeling Learning semantic parsers from denotation can be viewed as a latent variable modeling problem, where the program is the latent variable. Probabilistic latent variable models have been studied using EM-algorithm and its variant (Dempster et al., 1977). The graphical model literature has studied latent variable learning on margin-based methods (Yu and Joachims, 2009) and probabilistic models (Quattoni et al., 2007). Samdani et al. (2012) studied various variants of EM algorithm and showed that all of them are special cases of a unified framework. Our generalized update framework is similar in spirit.

7 Conclusion

In this paper, we propose a general update equation from semantic parsing from denotation and propose a policy shaping method for addressing the spurious program challenge. For the future, we plan to apply the proposed learning framework to more semantic parsing tasks and consider new methods for policy shaping.

8 Acknowledgements

We thank Ryan Benmalek, Alane Suhr, Yoav Artzi, Claire Cardie, Chris Quirk, Michel Galley and members of the Cornell NLP group for their valuable comments. We are also grateful to Allen Institute for Artificial Intelligence for the computing resource support. This work was initially started when the first author interned at Microsoft Research.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1:49–62.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kevin Clark and D. Christopher Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 169–176. ACM.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles Lee Isbell, and Andrea Lockerd Thomaz. 2013. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1051–1062. Association for Computational Linguistics.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151. Association for Computational Linguistics.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, and Dan Klein. 2015. An empirical analysis of optimization for max-margin nlp. In *EMNLP*.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2016. Global neural CCG parsing with optimality guarantees. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2366–2376.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Ryan T. McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL*.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. Arxiv preprint: `https://arxiv.org/abs/1704.08795`.

Dipendra K. Misra and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kumar Dipendra Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Richard Montague. 1970. English as a formal language.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Morency Collins, and Trevor Darrell. 2007. Hidden conditional random fields. *IEEE transactions on pattern analysis and machine intelligence*, 29(10).

Kenneth Rose. 1998. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239.

Rajhans Samdani, Ming-Wei Chang, and Dan Roth. 2012. Unified expectation maximization. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, pages 688–698. Association for Computational Linguistics.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin markov networks. In *NIPS*.

Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *EMNLP-CoNLL*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the 26th annual international conference on machine learning*, pages 1169–1176. ACM.

John M Zelle and Raymond J Mooney. 1993. Learning semantic grammars with constructive inductive logic programming. In *AAAI*, pages 817–822.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Sentence Compression for Arbitrary Languages via Multilingual Pivoting

Jonathan Mallinson, Rico Sennrich and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
`J.Mallinson@ed.ac.uk, {rsennric,mlap}@inf.ed.ac.uk`

Abstract

In this paper we advocate the use of bilingual corpora which are abundantly available for training sentence compression models. Our approach borrows much of its machinery from neural machine translation and leverages bilingual pivoting: compressions are obtained by translating a source string into a foreign language and then back-translating it into the source while controlling the translation length. Our model can be trained for any language as long as a bilingual corpus is available and performs arbitrary rewrites without access to compression specific data. We release[1] MOSS, a new parallel **M**ultilingual C**o**mpre**ss**ion dataset for English, German, and French which can be used to evaluate compression models across languages and genres.

1 Introduction

Sentence compression aims to produce a summary of a single sentence that retains the most important information while preserving its fluency. The task has attracted much attention due to its potential for applications such as text summarization (Jing, 2000; Madnani et al., 2007; Woodsend and Lapata, 2010; Berg-Kirkpatrick et al., 2011), subtitle generation (Vandeghinste and Pan, 2004; Luotolahti and Ginter, 2015), and the display of text on small-screens (Corston-Oliver, 2001).

The bulk of research on sentence compression has focused on a simplification of the task involving exclusively word deletion (Knight and Marcu, 2002; Riezler et al., 2003; Turner and Charniak, 2005; McDonald, 2006; Clarke and Lapata, 2008; Cohn and Lapata, 2009), whereas a few approaches view sentence compression as a more general text rewriting problem (Galley and McKeown, 2007; Woodsend and Lapata, 2010; Cohn and Lapata, 2013). Irrespective of how the compression task is formulated, most previous work relies on syntactic information such as parse trees to help decide what to delete from a sentence or which rules to learn in order to rewrite a sentence using less words. More recently, there has been much interest in applying neural network models to natural language generation tasks, including sentence compression (Rush et al., 2015; Filippova et al., 2015; Chopra et al., 2016; Kikuchi et al., 2016). Filippova et al. (2015) focus on deletion-based sentence compression which they model as a sequence labeling problem using a recurrent neural network with long short-term memory units (LSTM; Hochreiter and Schmidhuber 1997). Rush et al. (2015) capture the full gamut of rewrite operations drawing insights from encoder-decoder models recently proposed for machine translation (Bahdanau et al., 2015).

Neural network-based approaches are data-driven, relying on the ability of recurrent architectures to learn continuous features without recourse to preprocessing tools or syntactic information (e.g., part-of-speech tags, parse trees). In order to achieve good performance, they require large amounts of training data, in the region of millions of long-short sentence pairs.[2] Existing compression datasets are several orders of magnitude smaller. For example, the Ziff-Davis corpus (Knight and Marcu, 2002) contains 1,067 sentences and originated from a collection of news articles on computer products. Clarke and Lapata (2008) create two manual corpora sampled from written (1,433 sentences) and spoken sources (1,370 sentences). Cohn and Lapata (2013) elicit manual compressions for 625 sentences taken from newspaper articles. More recently, Toutanova et al. (2016) crowdsource a larger corpus which contains manual compressions for single *and* multiple sentences (about 26,000 pairs of source and compressed texts).

---

[1] Publicly available for download at `https://github.com/Jmallins/MOSS`

[2] Rush et al. (2015) use approximately four million training instances and Filippova et al. (2015) two million.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2453–2464
Brussels, Belgium, October 31 - November 4, 2018.

Since large scale compression datasets do not occur naturally, they must be somehow approximated, e.g., by pairing headlines with the first sentence of a news article (Filippova and Altun, 2013; Rush et al., 2015). As a result, the training corpus construction process must be repeated and reconfigured for new languages and domains (e.g., many headline-first sentence pairs are spurious and need to be filtered using language and domain specific heuristics). And although it may be easy to automatically obtain large scale training data in the news domain, it is not clear how such data can be sourced for many other genres with different writing conventions.

Our work addresses the paucity of data for sentence compression models. We argue that *multilingual* corpora are a rich source for learning a variety of rewrite rules across languages and that existing neural machine translation (NMT) models (Sutskever et al. 2014; Bahdanau et al. 2015) can be easily adapted to the compression task through bilingual pivoting (Mallinson et al., 2017) coupled with methods which decode the output sequence to a desired length (e.g., subject to language and genre requirements). We obtain compressions by translating a source string into a foreign language and then back-translating it into the source while controlling the translation length (Kikuchi et al., 2016). Our model can be trained for any language as long as a bilingual corpus is available, and can perform arbitrary rewrites while taking advantage of multiple pivots if these exist.We also demonstrate that models trained on multilingual data perform well out-of-domain.

Although our approach does not employ compression corpora for training, for evaluation purposes, we create MOSS, a new **M**ultilingual C**o**mpre**ss**ion dataset for English, French, and German. MOSS is a *parallel* corpus containing documents from the European parliament proceedings, TED talks, news commentaries, and the EU bookshop. Each document is written in English, French, and German, and compressed by native speakers of the respective language who process a document at a time. We obtain five compressions per document leading to 2,000 long-short sentence pairs per language. Like previous related resources (Clarke and Lapata, 2008; Cohn and Lapata, 2013; de Loupy et al., 2010) our corpus is curated manually, however it differs from Toutanova et al. (2016) in that it contains compressions for individual sentences, not documents.

There has been relatively little interest in compressing languages other than English. A few models have been proposed for Japanese (Hori and Furui, 2004; Hirao et al., 2009; Harashima and Kurohashi, 2012), including a neural network model (Hasegawa et al., 2017) which repurposes Filippova and Altun's (2013) data construction method for Japanese. There is a compression corpus available for French (de Loupy et al., 2010), however, we are not aware of any modeling work on this language. Overall, there are no standardized datasets in languages other than English, either for training or testing.

Our contributions in this work are three-fold: a novel application of bilingual pivoting to sentence compression; corroborated by empirical results showing that our model scales across languages and text genres without additional supervision over and above what is available in the bilingual parallel data; and the release of a multilingual, multi-reference compression corpus which can be effectively used to gain insight in the compression task and facilitate further research in compression modeling.

2 Pivot-based Neural Compression

In our pivot-based sentence compression model an input sequence is first translated into a foreign language, and then back into the source language. Unlike previous paraphrasing pivoting models (Mallinson et al., 2017), we parameterize our translation models with a length feature, which allows us to produce compressed output. We define two models, performing compression in one step or alternatively in two steps which affords more flexibility in model output.

2.1 NMT Background

In the neural encoder-decoder framework for MT (Bahdanau et al., 2015; Sutskever et al., 2014), an encoder takes in a source $X = (x_1, \ldots, x_{T_x})$ of length T_x and the decoder generates a target sequence $(y_1, \ldots, y_{T_y})$ of length T_y. Let h_i be the hidden state of the source symbol at position i, obtained by concatenating the forward and backward encoder RNN hidden states, $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$. We deviate from previous work (Bahdanau et al., 2015; Sutskever et al., 2014) in that we initialize the decoder with the average of the hidden states, following Sennrich et al. (2017):

$$s_0 = tanh(W_{init} \frac{\sum_{i=1}^{T_x} h_i}{T_x}) \qquad (1)$$

where W_{init} is a learnt parameter. Our decoder is a conditional recurrent neural network, specifically a gated recurrent unit (GRU, Cho et al., 2014) with attention, which we denote as $cGRU_{att}$. $cGRU_{att}$ takes as input the previous hidden state s_{j-1}, the source annotations $C = h_1, ..., h_{T_x}$, and the previously decoded symbol y_{j-1} in order to update its hidden state s_j, which is used to decode symbol y_j at position j:

$$s_j = cGRU_{att}(s_{j-1}, y_{j-1}, C) \tag{2}$$

$cGRU_{att}$ consists of three components. The first combines the previously decoded symbol y_{j-1} and the previous hidden state s_{j-1} to generate an intermediate representation s'_j. The attention mechanism, ATT, inputs the entire context set C along with intermediate hidden state s'_j in order to compute the context vector c_j:

$$c_j = ATT(C, s'_j) = \sum_i^{T_x} \alpha_{ij} h_i \tag{3}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{kj})} \tag{4}$$

$$e_{ij} = f(s'_j, h_i) \tag{5}$$

Where α_{ij} is the normalized alignment weight between the source symbol at position i and the target symbol at position j, and f is a feedfoward neural network.

Finally, we generate s_j, the hidden state of $cGRU_{att}$, by using the intermediate representation s'_j and the context vector c_j. Given s_j, y_{j-1}, and c_j the output probability $p(y_j | s_j, y_{j-1}, c_j)$ is computed using a feedforward neural network with a softmax activation. We define the probability of sequence y as:

$$P(y|x; \theta) = \prod_{j=1}^{T_y} p(y_j | s_j, y_{j-1}, c_j) \tag{6}$$

2.2 Length Control

To be able to produce compressed sentences, we parameterize our model with a length vector which allows to control the output length. Our approach is similar to the *LenInit* model of Kikuchi et al. (2016), however we use a GRU instead of an LSTM. The hidden state of the decoder consists of the average of the encoder's hidden states but also a length vector LV, a learnt parameter, which is scaled by the desired target length $T_{y'}$. We therefore rewrite Equation (1) as follows:

$$s'_0 = tanh\left(W_{init}\left[\frac{\sum_{i=1}^{T_x} h_i}{T_x}; LV \cdot T_{y'}\right]\right) \tag{7}$$

As such we now define our model as:

$$P(y|x, T_{y'}; \theta) \tag{8}$$

During training, the target length is set to $T_{y'} = T_y$. However, at test time, the target length generally varies according to the domain, genre, and language at hand. We determine the target length experimentally based on a small validation set.

2.3 Pivoting

Pivoting is often used in machine translation to overcome the shortage of parallel data, i,e., when there is no translation path from the source language to the target by taking advantage of paths through an intermediate language. The idea dates back at least to Kay (1997), who observed that ambiguities in translating from one language onto another may be resolved if a translation into some third language is available, and has met with success in phrase-based SMT (Wu and Wang, 2007; Utiyama and Isahara, 2007) and more recently in neural MT systems (Firat et al., 2016).

We use pivoting to provide a path from a source English sentence, via an intermediate foreign language, to English in a compressed form. We propose to extend Mallinson et al.'s (2017) approach to multi-pivoting, where a sentence x is translated to K-best foreign pivots, $F_x = \{f_1, ..., f_K\}$. The probability of generating compression $y = y_1...y_{T_y}$ is decomposed as:

$$P(y|x) = \sum_f^{F_x} P(y|f; \overrightarrow{\theta}) \cdot P(f|x; \overleftarrow{\theta}) \tag{9}$$

which we approximate as the tokenwise weighted average of the pivots:

$$P(y|x) \approx \prod_{j=1}^{T_y} \sum_f^{F_x} P(y_j | y_{<j}, f) P(f|x) \tag{10}$$

where $y_{<j} = y_1, ...y_j$. To ensure a probability distribution, we normalize the K-best list F_x, such that the translation probabilities sum to one. We use beam search to decode tokens by conditioning on multiple pivoting sentences. The results with the best decoding scores are considered candidate compressions.

To ensure the model produces compressed output, we extend the pivoting approach in two ways. In *single step* compression, one of the translation

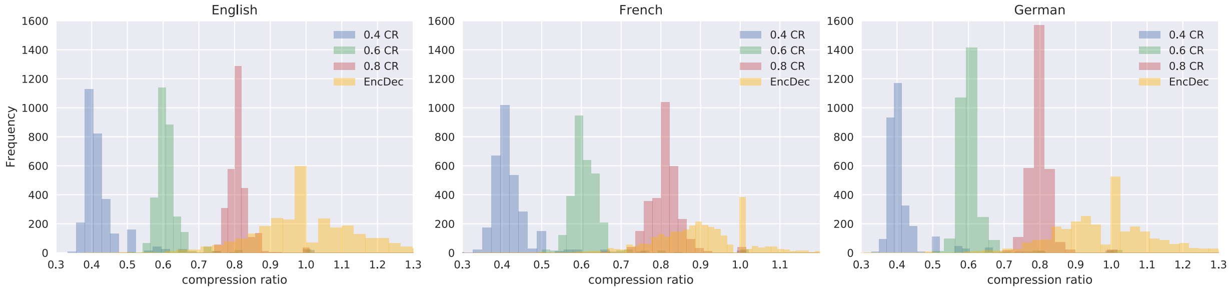

Figure 1: Histograms of output lengths at three compression rates (CR) compared to a vanilla encoder-decoder system which does not manipulate output length. German is used as pivot for English, and English as pivot for French and German.

models is parameterized with length information:

$$P(y|x,T_{y'}) \approx \sum_{f}^{F} P(y|f,T_{y'};\overrightarrow{\theta}) \cdot P(f|x;\overleftarrow{\theta})$$

In *dual-step* compression, we parameterize both translation models with length information:

$$P(y|x,T_{y'},T_{y''}) \approx \sum_{f}^{F} P(y|f,T_{y'};\overrightarrow{\theta}) \cdot P(f|x,T_{y''};\overleftarrow{\theta})$$

We find that dual-compression performs better when the system is expected to drastically compress the source sentence (e.g., in a headline generation task). Imposing a high compression ratio from the start tends to produce unintelligible text. The model attempts to reduce the length of the source at all costs, even at the expense of being semantically faithful to the input. Performing two moderate compressions in succession reduces both length and content conservatively and as a result produces more meaningful text.

In Figure 1 we illustrate how the pivot-based model sketched above can successfully control the output of the generated compressions. We show the output of a single-step compression model on three languages initialized with varying compression rates[3] (see Section 4 for details on how the models were trained and tested). The compression rate (CR) is used to determine length parameter of Equation (8):

$$T_{y'} = T_x \cdot CR \tag{11}$$

The figure shows how the output length varies compared to a vanilla encoder-decoder system which uses pivoting to backtranslate the source language (Mallinson et al., 2017). We can see that the majority of sentences are generated with length close to the desired compression rate.

3 The Moss Dataset

For evaluation purposes, we created a multilingual sentence compression corpus in English, German, and French. The corpus was collated from existing document and sentence aligned multilingual datasets which vary both in terms of topic and genre. We sampled five documents each from:

1. Europarl, the European Parliament Proceedings Parallel Corpus (Koehn, 2005), has been used extensively in machine translation research; it contains the minutes of the European parliament and is a spoken corpus of formulaic nature; speakers take part in debating various issues concerning EU policy (e.g., taxation, environment).
2. The TED parallel Corpus (Cettolo et al., 2012) contains transcripts in multiple languages of short talks devoted to spreading powerful ideas on a variety of topics ranging from science to business and global issues.
3. The EU bookshop corpus (Skadiņš et al., 2014) contains publications from European institutions covering a variety of topics such as refugees, gender equality, and travel.
4. The News Commentary Parallel Corpus contains articles downloaded from Project Syndicate, an international media organization that publishes commentary on global topics (e.g., economics, world affairs).

We obtained compressions using the Crowdflower platform. Crowdworkers were given instructions that explained the task and defined sentence compression with the aid of examples. They

---

[3] The term refers to the percentage of words retained from the source sentence in the compression.

English	French	German
On the very day that the earthquake struck, the European Council asked the High Representative and the Commission to mobilise all appropriate assistance. *Assistance was mobilized on the very day of the earthquake.*	Le jour même du tremblement de terre, le Conseil européen a demandé à la haute représentante et à la Commission de mobiliser toute l'aide appropriée. *Le Conseil européen a demandé à la haute reprsentante et à la Commission de mobiliser l'aide.*	Am gleichen Tag, an dem das Erdbeben ausbrach, ersuchte der Europäische Rat die Hohe Vertreterin und die Kommission um die Mobilisierung aller angemessenen Hilfe. *Europa erbrachte Hilfe noch am selben Tag.*
We're at a tipping point in human history, a species poised between gaining the stars and losing the planet we call home. *We're at tipping point in human history, poised between gaining the stars and losing the Earth.*	L'histoire humaine est à un tournant. Notre espèce hésite à toucher les étoiles ou à perdre la planète qui est la sienne. *L'humanité est à un tourt. Notre espèce a envie des étoiles ou perdre sa planète.*	Wir stehen vor einem historischen Wendepunkt: zwischen dem Griff nach den Sternen und dem Verlust unseres Heimatplaneten. *Wir sind vor einem historischen Wendepunkt: zwischen dem Griff nach Sternen und Verlust unseres Planeten.*
Surveys undertaken by the World Bank in developing countries show that when poor people are asked to name the three most important concerns they face good health is always mentioned. *World Bank surveys in developing countries show poor people always name good health as an important concern.*	Les enquêtes menées par la Banque mondiale dans les pays en développement montrent que, quand on demande aux populations pauvres de nommer les trois défis les plus importants qu'ils rencontrent, leur "bonne santé" fait toujours partie de cette liste. *Quand on demande aux populations pauvres de nommer les trois défis les plus importants qu'ils rencontrent, leur "bonne santé" fait toujours partie de la liste.*	Umfragen der Weltbank in Entwicklungsländern zeigen, wenn man Arme nach den drei wichtigsten Anliegen fragt, die sie beschäftigen, wird "Gesundheit" immer genannt. *Umfragen in Entwicklungslndern zeigen, dass bei Armen das wichtigste Anliegen Gesundheit ist.*

Table 1: Examples of crowdsourced compressions (in italics) from the MOSS corpus. Sentences shown (in order of appearance) from Europarl, TED, and News Commentary corpora.

were asked to compress while preserving the most important information, ensuring the sentences remained grammatical and meaning preserving. Annotators were encouraged to use any rewriting operations that seemed appropriate, e.g., to delete words, add new words, substitute them, or reorder them. Annotation proceeded on a document-by-document basis, line-by-line. Crowdworkers compressed the first twenty lines of each document and we elicited five compression per document. Example compressions are shown in Table 1.

Table 2 presents various statistics on our corpus. As can be seen, Europarl contains the longest sentences across languages (see column SL), TED contains the shortest sentences, while the other two corpora are somewhere in-between. We also observe that crowdworkers compress the least when it comes to TED (see column CR), which is not surprising given the brevity of the utterances. Overall, French speakers seem more conservative when shortening sentences compared to English and German. In general, compression rates are genre dependent, they range from 0.58 (for English Europarl) to 0.84 (for German TED). We also examined the degree to which crowdworkers paraphrase the source sentence using Translation Edit Rate (TER; Snover et al., 2006), a measure commonly used to automatically evaluate the quality of machine translation output. We used TER to compute the (average) number of edits required to change a long sentence to shorter output. We also report the number of edits by type, i.e., the number of insertions, substitutions, deletions, and shifts needed (on average) to convert long to short sentences. We observe that crowdworkers perform a fair amount of rewriting across corpora and languages. The most frequent rewrite operations are deletions followed by substitutions, shifts, and insertions.

4 Experimental Setup

Neural Machine Translation Training Nematus (Sennrich et al., 2017) was used as the machine translation system for all our experiments. We generally used the default settings and training procedures as specified within Nematus. All networks have a hidden layer size of 1,000, and an embedding layer size of 512. In addition, layer normalization (Ba et al., 2016) was used. During training, we used ADAM (Kingma and Ba, 2014), a minibatch size of 80, and the training set was reshuffled between epochs. We also employed early stopping.

English	SL	TL	CR	TER	Ins	Del	Sub	Shft
EUPar	27.29	15.89	0.58	0.47	0.07	11.29	1.57	0.50
TED	10.64	7.70	0.72	0.40	0.11	2.89	1.33	0.23
News	20.32	15.25	0.75	0.42	0.21	5.58	2.64	0.55
Books	20.52	15.23	0.78	0.38	0.14	5.43	1.86	0.46
All	19.69	13.51	0.71	0.42	0.13	6.30	1.85	0.44

French	SL	TL	CR	TER	Ins	Del	Sub	Shft
EUPar	29.40	23.00	0.78	0.29	0.06	6.39	1.89	0.20
TED	6.16	4.88	0.79	0.46	0.03	1.63	1.35	0.04
News	27.52	21.03	0.76	0.37	0.11	6.56	2.92	0.54
Books	21.97	17.93	0.82	0.24	0.08	4.12	1.25	0.16
All	21.26	16.71	0.79	0.34	0.07	4.66	1.85	0.24

German	SL	TL	CR	TER	Ins	Del	Sub	Shft
EUPar	24.78	15.52	0.63	0.41	0.02	9.28	1.08	0.22
TED	5.44	4.54	0.84	0.25	0.02	0.84	0.49	0.07
News	23.35	14.68	0.63	0.47	0.08	9.42	2.11	0.51
Books	19.35	14.37	0.74	0.37	0.09	5.07	1.94	0.31
All	18.23	12.28	0.71	0.38	0.05	6.15	1.41	0.28

Table 2: MOSS statistics across corpora and languages: length of source (SL) and target sentence (TL), compression rate (CR), TER scores, and number of insertions (Ins), deletions (Del), substitutions (Sub), and shifts (Shft).

We used up to four encoder-decoder NMT models in our experiments (BLEU scores[4] shown in parentheses): English→French (27.03), French→English (29.14), English→German (28.3), and German→English (31.19). German training/test data was taken from the WMT16 shared task and French from the WMT14 shared task. The training data was 4.2 million and 39 million sentence pairs for en-de, and en-fr, respectively. We also used back-translated monolingual training data, from the news domain, (Sennrich et al., 2016a) in training for the German systems. The data was pre-processed using standard scripts found in MOSES (Koehn et al., 2007). Rare words were split into sub-word units, using byte pair encoding (BPE; Sennrich et al. 2016b). The BPE operations are shared between language directions.

We experimented with various model variants using one or multiple pivots. The compression rate (see Equation (8)) was tuned experimentally on the validation set which consists of one document from each domain (20 source sentences; 100 compression-pairs). Compression rates varied from 0.55 to 0.85 and were broadly comparable to those shown in Table 2.

Comparison Systems We compared our model against *ABS*, a sequence-to-sequence attention-based model, developed by Rush et al. (2015). This model was trained on a monolingual dataset extracted from the Annotated English Gigaword corpus (Napoles et al., 2011). The dataset consists of approximately 4 million pairs of the first sentence from each source document and its headline. We also trained *LenInit* (Kikuchi et al., 2016) on the same corpus which is conceptually similar to ABS but additionally controls the output length using a length embedding vector (as described in Section 2.2).[5] Unfortunately, we could not train these models for French or German, since there are no monolingual sentence compression datasets available at a similar scale. An obvious workaround is to translate Gigaword to French and German and then train compression models on the translated data. As the quality of the translation is relatively poor, we also translated German or French into English, compressed it with ABS and LenInit trained on the Gigaword corpus, and then translated the compressions back to French or German. Finally, we include a prefix (*Pfix*) baseline which does not perform any rewriting but simply truncates the source sentence so that it matches the compression ratio of the validation set.

5 Results

MOSS Evaluation We assessed model performance using three automatic metrics which represent different aspects of the compression task and have been found to correlate well with human judgments (Toutanova et al., 2016; Clarke and Lapata, 2006). These include a recall metric based on skip bi-grams, any pair of words in a sequence allowing for gaps of size four[6] (RS-R); a recall metric based on bi-grams of dependency tree triples (D2-R); and bi-gram ROUGE (R2-F1). We used the Stanford neural network parser (Chen and Manning, 2014) to obtain dependency triples.

Table 3(a) reports results on English with a model which controls the output length ($\mathcal{L}$) and uses either a single pivot (SP; $K = 1$) or multiple pivots (MP; $K = 10$). We experimented with French (fr) or German (de) as pivot languages. All pivot-based models perform compression in a single step (see Section 2.3). Dual-step compres-

---

[4] BLEU scores were calculated using mteval-v13a.pl.

[5] We used our own implementation of ABS and LenInit which on DUC-2004 obtained ROUGE scores similar to those published in Rush et al. (2015) and Kikuchi et al. (2016).

[6] We add a begin-of-sentence marker at the start of the candidate and reference sentences.

English	RS-R	D2-R	R2-F1
Pfix	45.38	47.57	33.67
ABS	18.29	23.55	15.60
LenInit	17.90	19.64	11.18
SP$_{\mathcal{L},de}$	**34.60**	**37.97**	**22.67**
SP$_{\mathcal{L},fr}$	27.42	32.34	19.29
MP$_{\mathcal{L},de}$	28.71	34.70	19.06
MP$_{\mathcal{L},fr}$	20.74	27.50	13.89
Gold	76.60	71.68	42.89

(a)

French	RS-R	D2-R	R2-F1
Pfix	60.33	62.44	53.37
ABS	13.84	18.00	9.74
ABS$_{en}$	16.39	22.08	13.17
LenInit	9.91	14.52	8.08
LenInit$_{en}$	20.08	24.41	13.06
SP$_{\mathcal{L},en}$	**43.38**	**46.17**	**35.07**
MP$_{\mathcal{L},en}$	31.55	37.88	26.59
Gold	74.42	80.00	52.13

(b)

German	RS-R	D2-R	R2-F1
Pfix	56.28	50.78	45.84
ABS	5.72	12.95	5.21
ABS$_{en}$	9.43	14.78	6.79
LenInit	4.91	11.77	2.87
LenInit$_{en}$	13.19	18.67	7.65
SP$_{\mathcal{L},en}$	**38.19**	**38.54**	**31.15**
MP$_{\mathcal{L},en}$	23.62	29.13	17.36
Gold	76.01	77.48	48.36

(c)

Table 3: Automatic evaluation on MOSS; S/MP: single/multiple pivot models; $\mathcal{L}$: length parameter; pivot languages: English (en), French (fr), German (de); ABS (Rush et al., 2015) and LenInit (Kikuchi et al., 2016) are sequence-to-sequence models trained on Gigaword; Gold is inter-annotator agreement.

	English	**French**	**German**
ABS	Europe urged to help quake victims.	Le Conseil Européen demande une aide pour les victimes du tremblement de terre.	Europäischer Rat sucht Hilfen für Quiz-Opfer.
SP	The European Council called on the High Representative and the Commission to mobilise all appropriate assistance.	Le Conseil Européen a demandé au Haut Représentant et à la Commission de mobiliser l'assistance.	Am selben Tag forderte der Europäische Rat die Hohe Vertreterin und die Kommission auf, jede Hilfe.
ABS	Advance for Sunday July a new look at the world.	Un tournant pour le tournant.	Die Stars der Stars und die Stars.
SP	We are at a turning point in human history and losing the planet we call home.	L'histoire de l'humanité est à la croisée des chemins et de l'histoire.	Zwischen dem Griff der Sterne und dem Verlust unseres Planeten stehen wir vor.
ABS	Poor people ask to name the three most important concerns.	Les enquêtes de la Banque mondiale révèlent que la santé fait toujours partie de la liste.	Weltbank-Umfragen zeigen arme Menschen in Entwicklungsländern.
SP	Polls conducted by the World Bank show that when poor people are asked to mention the three main concerns.	Les enquêtes menées par la Banque mondiale dans les pays en développement montrent que, lorsqu'on demande aux pauvres de nommer les trois plus grands éfis.	Wenn man die Armen nach den drei Hauptanliegen fragt, werden sie gefordert.

Table 4: System output for the example source sentences in Table 1.

sion obtained inferior results which we omit for the sake of brevity. As can be seen, models which use a single pivot are better than those using multiple ones (German is better than French; see SP$_{de}$ vs SP$_{fr}$). More pivots might introduce noise at the expense of translation quality.

Overall, pivot-based models outperform ABS and LenInit. This is perhaps to be expected since these models are tested on out of domain data with different vocabulary and writing conventions; MOSS does not contain any newspaper articles. Unfortunately, it is not possible to train ABS and LenInt on in-domain data as compression data only exists for the headlines-first sentences pairs. As an upper bound, we also report how well humans agree with each other, treating one (randomly selected) reference as system output and computing how it agrees with the rest (row Gold in Table 3). All models lag significantly behind human performance on this task.

Tables 3(b) and 3(c) report results on French and German, respectively. For these languages, we obtained best results with English as pivot, using a single-step compression model. ABS and LenInit perform poorly when trained directly on translations of Gigaword into French and German; their performance improves considerably when they are trained on the Gigaword and used to compress English translations of French or German (ABS$_{en}$, LenInit$_{en}$). Again, we observe that our models (SP$_{\mathcal{L},en}$, MP$_{\mathcal{L},en}$) outperform the comparison systems across all metrics and that using a single pivot yields better compressions. Example compressions are given in Table 4 where we show output produced by ABS and SP for each language (see the supplementary material for more examples). Finally, notice that automatic scores for the prefix baseline across languages are misleadingly high, since it simply repeats the source sentence up to a fixed length without performing any rewriting.

	English			French			German		
Models	Imp	Gram	Avg	Imp	Gram	Avg	Imp	Gram	Avg
Pfix	2.72	2.98	2.85	2.73	2.89	2.80	3.17	2.96	3.06
LenInit	2.51	3.0	2.75	1.82	2.62	2.22	2.10	3.25	2.67
SP$_{\mathcal{L}}$	**3.27**	**3.69**	**3.48**	**3.48**	**3.60**	**3.54**	**3.30**	**3.87**	**3.59**
Ref	3.47	3.80	3.63	4.05	4.14	4.10	3.97	4.26	4.10

Table 5: Mean ratings elicited by humans on MOSS; Avg is the average rating of grammaticality and importance.

	SL	TL	CR	TER	Ins	Del	Sub	Shft
English	19.69	12.31	0.63	0.65	0.10	6.68	2.14	0.44
French	21.26	14.98	0.70	0.67	0.29	5.71	3.36	0.61
German	18.23	12.51	0.68	0.67	0.16	6.38	2.94	0.50

Table 6: Statistics of model output (SP$_{\mathcal{L}}$) on MOSS (aggregated across domains): length of source (SL) and target (TL), compression rate (CR), TER scores, and number of insertions (Ins), deletions (Del), substitutions (Sub), and shifts (Shft).

We also elicited human judgments through the Crowdflower platform. We asked crowdworkers to rate the grammaticality of the target compressions and whether they preserved the most important information from the source. In both cases, they used a five-point rating scale where a high number indicates better performance. We randomly selected 25 sentences from each corpus from the test portion of MOSS, i.e., 100 long-short sentence pairs per language. We compared compressions generated by our model (SP$_{\mathcal{L}}$), with ABS models for the three languages, the prefix baseline, and (randomly selected) gold-standard reference (Ref) compressions from MOSS. All systems used the length parameter to allow comparisons with approximately the *same* compression rates. We collected five ratings per compression. Our results are summarized in Table 5. We show mean ratings for grammaticality (Gram), importance (Imp) and their combination (column Avg). Across languages our model (SP$_{\mathcal{L}}$) significantly ($p < 0.05$) outperforms comparison systems (Pfix, ABS) on both dimensions of grammaticality and importance (significance tests were performed using a student t-test). All systems are significantly worse ($p < 0.05$) than the human reference compressions.

Models	RS-R	D2-R	R2-F1	R1-R	R2-R	RL-R
Pfix	15.25	15.59	5.38	20.42	5.86	18.07
SP$_{\mathcal{L},de}$	**12.93**	**13.89**	**4.97**	**20.70**	**5.35**	**18.35**
SP$_{\mathcal{L},fr}$	12.06	12.18	4.42	19.77	4.75	17.40
MP$_{\mathcal{L},fr}$	10.38	11.85	3.70	18.67	4.03	16.20
MP$_{\mathcal{L},de}$	11.06	13.26	4.30	19.10	4.69	16.84
Gold	16.41	18.12	7.72	26.95	7.72	22.79
ABS[7] (Rush et al., 2015)				26.55	7.06	22.05
ABS+ (Rush et al., 2015)				28.18	8.49	23.81
RAS (Chopra et al., 2016)				28.97	8.26	24.06
LenInit [8] (Kikuchi et al., 2016)				25.87	8.27	23.24
LenEmb (Kikuchi et al., 2016)				26.73	8.40	23.88

Table 7: DUC-2004 results (75 char length cap); results for comparison systems are taken from their respective papers.

Finally, in Table 6 we analyze the output of our best model (SP$_{\mathcal{L}}$) using the same statistics we applied to the human compressions (see Table 2). As can be seen, the model generally compressess more aggressively and applies more edits than the crowdworkers (both compression rates and TER scores are higher for all three languages). Although the rate of deletions is similar to humans, insertions, substitutions and shifts happen to a greater extent for our model, indicating that it performs a good amount of paraphrasing.

DUC-2004 Evaluation Besides MOSS, we evaluated our model on the benchmark DUC-2004 task-1 dataset. In this task, the aim is to create a very short summary (75 bytes) for a document. The evaluation set consists of 500 source documents (from the New York Times and Associated Press Wire services) each paired with four human-written (reference) summaries. We follow previous work (Rush et al., 2015; Chopra et al., 2016) in compressing the first sentence of the document and presenting this as the summary. To make the evaluation unbiased to length, the output of all systems is cut-off after 75-characters and no bonus is given for shorter summaries.

Our results are shown in Table 7. To compare with existing methods, we also report ROUGE (Lin, 2004) unigram and bigram overlap (Lin, 2004) and the longest common subsequence (ROUGE-L).[9] We employed a dual step compression model (see Section 2) as preliminary experiments showed that it was superior to single-stage variants. We compared single and multiple pivot models against existing ABS and ABS+ (Rush et al., 2015), two encoder-decoder models trained on the English Gigaword. ABS+ applies minimum error rate (MERT) training as a copy-

---

[7] Our ABS implementation obtains R1-R 25.03, R2-R 8.40, and RL-R: 22.35

[8] Our LenInit implementation obtains R1-R 29.26, R2-R 9.56, and RL-R 25.70

[9] We used ROUGE version 1.5.5 with the original DUC-2004 ROUGE parameters.

Source	King Norodom Sihanouk has declined requests to chair a summit of Cambodia's top political leaders, saying the meeting would not bring any progress in deadlocked negotiations to form a government.
SP$_{\mathcal{L},de}$	King Norodom Sihanouk has refused to chair Cambodia summit.
Gold	Sihanouk refuses to chair Cambodian political summit at home or abroad.
Source	Cambodia's ruling party responded Tuesday to criticisms of its leader in the U.S. Congress with a lengthy defense of strongman Hun Sen's human rights record.
SP$_{\mathcal{L},de}$	Cambodia's ruling party responded Tuesday to criticism of its leader in the US.
Gold	Cambodian party defends leader Hun Sen against criticism of U.S. House.
Source	The Swiss government has ordered no investigation of possible bank accounts belonging to former Chilean dictator Augusto Pinochet, a spokesman said Wednesday.
SP$_{\mathcal{L},de}$	Swiss government ordered no inquiry into possible bank accounts of former Chilean dictator Augusto.
Gold	Switzerland joins charges against Pinochet but avoids bank probe.

Table 8: System output for DUC-2004.

ing mechanism. LenEmb and LenInit include a length parameter (Kikuchi et al., 2016), whereas RAS uses a specialized recurrent neural network architecture (Elman, 1990). We also report how well DUC-2004 abstractors agree with each other (row Gold in Table 7). Example compressions are given in Table 8, where we show output produced by SP$_{\mathcal{L},de}$ and a human reference (see the supplementary material for further examples).

Using automatic metrics we see that our model generally performs worse compared to these systems and that German is the best pivot for English. Although the objective of this paper is not to obtain state-of-the-art scores on this evaluation set, it is interesting to see that our model is able to compress out-of-domain. We do not have access to headline-first sentence pairs, while all comparison systems do. We also elicited human judgments on the compressions of 100 lead sentences whose documents were randomly selected from the DUC-2004 test set. We compared the prefix baseline, our model (SP$_{\mathcal{L},de}$), ABS+ (Rush et al., 2015), LenEmb (Kikuchi et al., 2016), Topiary (Zajic et al., 2004), and a randomly selected reference. Topiary came top in almost all measures in the DUC-2004 evaluation; it first compresses the lead sentence using linguistically motivated heuristics and then enhances it with topic keywords. Crowdworkers rated grammaticality and importance, using a five-point scale; we collected five ratings per compression.

As shown in Table 9 ABS+ has the lead with our system following suit. In terms of grammaticality, ABS+ and SP$_{\mathcal{L},de}$ are not significantly different from the gold standard or from each other (Pfix, Topiary, and LenEmb are significantly worse than Gold; $p < 0.05$). In terms of importance, pairwise differences between systems and the gold standard are not significant. Overall, we observe that SP$_{\mathcal{L},de}$ performs comparably to ABS+ even though it was not trained on any compression specific data. Inspection of system output reveals that our model performs more paraphrasing than comparison systems (a conclusion also confirmed by the statistics in Table 6).

Models	Gram	Imp	Avg
Pfix	3.03	2.93	2.98
SP$_{\mathcal{L},de}$	3.37	3.22	3.29
Topiary	3.05	3.15	3.10
ABS+	**3.67**	**3.23**	**3.45**
LenEmb	3.14	3.08	3.11
Ref	3.62	3.27	3.45

Table 9: Mean ratings elicited by humans on DUC-2004; Avg is the average rating of grammaticality and importance.

6 Conclusions

In this paper we have shown that multilingual corpora can be used to bootstrap compression models across languages and text genres. Our approach adapts existing neural machine translation machinery to the compression task coupled with methods which decode the output to a desired length. An interesting direction for future work would be to train our model using reinforcement learning (Ranzato et al., 2016; Zhang and Lapata, 2017) in order to control the compression output more directly. Moreover, although we do not use any direct supervision in our experiments, it would be interesting to incorporate it as a means of domain adaptation (Cheng et al., 2016).

Acknowledgments The authors gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1; Mallinson) and the European Research Council (award number 681760; Lapata).

References

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, San Diego, California.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490, Portland, Oregon.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750, Doha, Qatar.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1965–1974, Berlin, Germany.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 377–384.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:273–381.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34:637–674.

Trevor Cohn and Mirella Lapata. 2013. An abstractive approach to sentence compression. *ACM Transactions on Intelligent Systems and Technology*, 4(3):1–35.

Simon Corston-Oliver. 2001. Text Compaction for Display on Very Small Screens. In *Proceedings of the NAACL Workshop on Automatic Summarization*, pages 89–98, Pittsburgh, Pennsylvania.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368, Lisbon, Portugal.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491, Seattle, Washington, USA.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277. Association for Computational Linguistics.

Michel Galley and Kathleen McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 180–187, Rochester, New York.

Jun Harashima and Sadao Kurohashi. 2012. Flexible Japanese sentence compression by relaxing unit constraints. In *Proceedings of COLING 2012*, pages 1097–1112, Mumbai, India.

Shun Hasegawa, Yuta Kikuchi, Hiroya Takamura, and Manabu Okumura. 2017. Japanese sentence compression with a large training dataset. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 281–286, Vancouver, Canada.

Tsutomu Hirao, Jun Suzuki, and Hideki Isozaki. 2009. A syntax-free approach to japanese sentence compression. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 826–833, Suntec, Singapore.

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

Chiori Hori and Sadaoki Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems*, E87-D(1):15–25.

Hongyan Jing. 2000. Sentence Reduction for Automatic Text Summarization. In *Proceedings of the 6th ANLP*, pages 310–315, Seattle,WA.

Martin Kay. 1997. The proper place of men and machines in language translation. *Machine Translation*, 12(1-2):3–23.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit*, pages 70–86, Phuket, Thailand.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain.

Claude de Loupy, Marie Guégan, Christelle Ayache, Somara Seng, and Juan-Manuel Torres Moreno. 2010. A french human reference corpus for multi-document summarization and sentence compression. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

Juhani Luotolahti and Filip Ginter. 2015. Sentence compression for automatic subtitling. In *Proceedings of the 20th Nordic Conference for Computational Linguistics*, pages 135–143, Vilnius, Lithuania.

Nitin Madnani, David Zajic, Bonnie Dorr, Necip Fazil Ayan, and Jimmy Lin. 2007. Mutiple alternative sentence compressions for automatic text summarization. In *Proceedings of the 2007 Document Understanding Conference*, Rochester, NY, USA.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893, Valencia, Spain.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic constraints. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 297–304, Trento, Italy.

Courtney Napoles, Chris Callison-Burch, Juri Ganitkevitch, and Benjamin Van Durme. 2011. Paraphrastic sentence compression with a character-based metric: Tightening without deletion. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 84–90, Portland, Oregon.

MarcAurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations*, San Juan, Puerto Rico.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of the HLT/NAACL*, pages 118–125, Edmonton, Canada.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany.

Raivis Skadiņš, Jörg Tiedemann, Roberts Rozis, and Daiga Deksne. 2014. Billions of parallel words for free: Building and using the EU bookshop corpus. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation of the Americas*, pages 223–231, Cambridge, Massachusetts.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 340–350, Austin, Texas.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 290–297, Ann Arbor, Michigan.

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 484–491, Rochester, New York.

Vincent Vandeghinste and Yi Pan. 2004. Sentence compression for automated subtitling: A hybrid approach. In *Proceedings of the ACL Workshop on Text Summarization*, pages 89–95, Barcelona, Spain.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 565–574, Uppsala, Sweden.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *In Proceedings of the NAACL Workshop on Document Understanding*, pages 112–119, Boston, Massachusetts.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Copenhagen, Denmark.

Unsupervised Cross-lingual Transfer of Word Embedding Spaces

Ruochen Xu, Yiming Yang, Naoki Otani, Yuexin Wu
Carnegie Mellon University
{ruochenx, yiming, notani, yuexinw }@cs.cmu.edu

Abstract

Cross-lingual transfer of word embeddings aims to establish the semantic mappings among words in different languages by learning the transformation functions over the corresponding word embedding spaces. Successfully solving this problem would benefit many downstream tasks such as to translate text classification models from resource-rich languages (e.g. English) to low-resource languages. Supervised methods for this problem rely on the availability of cross-lingual supervision, either using parallel corpora or bilingual lexicons as the labeled data for training, which may not be available for many low resource languages. This paper proposes an unsupervised learning approach that does not require any cross-lingual labeled data. Given two monolingual word embedding spaces for any language pair, our algorithm optimizes the transformation functions in both directions simultaneously based on distributional matching as well as minimizing the back-translation losses. We use a neural network implementation to calculate the Sinkhorn distance, a well-defined distributional similarity measure, and optimize our objective through back-propagation. Our evaluation on benchmark datasets for bilingual lexicon induction and cross-lingual word similarity prediction shows stronger or competitive performance of the proposed method compared to other state-of-the-art supervised and unsupervised baseline methods over many language pairs.

1 Introduction

Word embeddings are well known to capture meaningful representations of words based on large text corpora (Mikolov et al., 2013; Pennington et al., 2014). Training word vectors using monolingual corpora is a common practice in various NLP tasks. However, how to establish cross-lingual semantic mapping among monolingual embeddings remain an open challenge as the availability of resources and benchmarks are highly imbalanced across languages.

Recently, increasing effort of research has been motivated to address this challenge. Successful cross-lingual word mapping will benefit many cross-lingual learning tasks, such as transforming text classification models trained in resource-rich languages to low-resource languages. Downstream applications include word alignment, text classification, named entity recognition, dependency parsing, POS-tagging, and more (Søgaard et al., 2015). Most methods for cross-lingual transfer of word embeddings are based on supervised or semi-supervised learning, i.e., they require cross-lingual supervision such as human-annotated bilingual lexicons and parallel corpora (Lu et al., 2015; Smith et al., 2017; Artetxe et al., 2016). Such a requirement may not be met for many language pairs in the real world.

This paper proposes an unsupervised approach to the cross-lingual transfer of monolingual word embeddings, which requires zero cross-lingual supervision. The key idea is to optimize the mapping in both directions for each language pair (say A and B), in the way that the word embedding translated from language A to language B will match the distribution of word embedding in language B. And when translated back from B to A, the word embedding after two steps of transfer will be maximally close to the original word embedding. A similar property holds for the other direction of the loop (from B to A and then from A back to B). Specifically, we use the Sinkhorn distance (Cuturi, 2013) to capture the distributional similarity between two set of embeddings after transformation, which we found empirically superior to the KL-divergence (Zhang et al., 2017a) and distance to nearest neighbor (Artetxe et al., 2017; Conneau et al., 2017) with regards to the quality of learned

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2465–2474
Brussels, Belgium, October 31 - November 4, 2018.

transformation as well as the robustness under different training conditions.

Our novel contributions in the proposed work include:

- We propose an unsupervised learning framework which incorporates the Sinkhorn distance as a distributional similarity measure in the back-translation loss function.
- We use a neural network to optimize our model, especially to implement the Sinkhorn distance whose calculation itself is an optimization problem.
- Unlike previous models which only consider cross-lingual transformation in a single direction, our model jointly learns the word embedding transfer in both directions for each language pair.
- We present an intensive comparative evaluation where our model achieved the state-of-the-art performance for many language pairs in cross-lingual tasks.

2 Related Work

We divide the related work into supervised and unsupervised categories. Representative methods in both categories are included in our comparative evaluation (Section 3.4). We also discuss some related work in unsupervised domain transfer in addition.

Supervised Methods: There is a rich body of supervised methods for learning cross-lingual transfer of word embeddings based on bilingual dictionaries (Mikolov et al., 2013; Faruqui and Dyer, 2014; Artetxe et al., 2016; Xing et al., 2015; Duong et al., 2016; Gouws and Søgaard, 2015), sentence-aligned corpora (Kočiskỳ et al., 2014; Hermann and Blunsom, 2014; Gouws et al., 2015) and document-aligned corpora (Vulić and Moens, 2016; Søgaard et al., 2015). The most relevant line of work is that by Mikolov et al. (2013) where they showed monolingual word embeddings are likely to share similar geometric properties across languages although they are trained separately and hence cross-lingual mapping can be captured by a linear transformation across embedding spaces. Several follow-up studies tried to improve the cross-lingual transformation in various ways (Faruqui and Dyer, 2014; Artetxe et al., 2016; Xing et al., 2015; Duong et al., 2016; Ammar et al., 2016; Artetxe et al., 2016; Zhang et al., 2016; Shigeto et al., 2015). Nevertheless, all these methods require bilingual lexicons for supervised learning. Vulić and Korhonen (2016) showed that 5000 high-quality bilingual lexicons are sufficient for learning a reasonable cross-lingual mapping.

Unsupervised Methods have been studied to establish cross-lingual mapping without any human-annotated supervision. Earlier work simply relied on word occurrence information only (Rapp, 1995; Fung, 1996) while later efforts have considered more sophisticated statistics in addition (Haghighi et al., 2008). The main difficulty in unsupervised learning of cross-lingual mapping is the formulation of the objective function, i.e., how to measure the goodness of an induced mapping without any supervision is a non-trivial question. Cao et al. (2016) tried to match the mean and standard deviation of the embedded word vectors in two different languages after mapping the words in the source language to the target language. However, such an approach has shown to be sub-optimal because the objective function only carries the first and second order statistics of the mapping. Artetxe et al. (2017) tried to impose an orthogonal constraint to their linear transformation model and minimize the distance between the transferred source-word embedding and its nearest neighbor in the target embedding space. Their method, however, requires a seed bilingual dictionary as the labeled training data and hence is not fully unsupervised. (Zhang et al., 2017a; Barone, 2016) adapted a generative adversarial network (GAN) to make the transferred embedding of each source-language word indistinguishable from its true translation in the target embedding space (Goodfellow et al., 2014). The adversarial model could be optimized in a purely unsupervised manner but is often suffered from unstable training, i.e. the adversarial learning does not always improve the performance over simpler baselines. Zhang et al. (2017b), Conneau et al. (2017) and Artetxe et al. (2017) also tried adversarial approaches for the induction of seed bilingual dictionaries, as a sub-problem in the cross-lingual transfer of word embedding.

Unsupervised Domain Transfer: Generally speaking, learning the cross-lingual transfer of word embedding can be viewed as a domain transfer problem, where the domains are word sets in different languages. Thus various work in the field of *unsupervised domain adaptation* or *unsupervised transfer learning* can shed light on our problem. For example, He et al. (2016) proposed a semi-supervised method for machine translation to utilize large monolingual corpora. Shen et al. (2017) used unsupervised learning to transfer sentences of different sentiments. Recent work in computer vision addresses the problem of image style transfer without any annotated training data (Zhu et al., 2017; Taigman et al., 2016; Yi et al., 2017). Among those, our work is mostly inspired by the work on CycleGAN (Zhu et al., 2017), and we adopt their cycled consistent loss over images into our back-translation loss. One key difference of our method from CycleGAN is that they used the training loss of an adversarial classifier as an indicator of the distributional distance, but instead, we introduce the Sinkhorn distance in our objective function and demonstrate its superiority over the representative method using adversarial loss (Zhang et al., 2017a).

3 Proposed Method

Our system takes two sets of monolingual word embeddings of dimension d as input, which are trained separately on two languages. We denote them as $X = \{x_i\}_{i=1}^n, Y = \{y_j\}_{j=1}^m, x_i, y_j \in \mathbb{R}^d$. During the training of monolingual word embedding for X and Y, we also have the access to the word frequencies, represented by vectors $r \in \mathbb{N}^n$ and $c \in \mathbb{N}^m$ for X and Y, respectively. Specifically, r_i is the frequency for word (embedding) x_i and similarly for c_j of y_j. As illustrated in Figure 3, our model has two mappings: $G : X \rightarrow Y$ and $F : Y \rightarrow X$. We further denote transferred embedding from X as $G(X) := \{G(x_i)\}_{i=1}^n$ and correspondingly for $F(Y)$.

In the unsupervised setting, the goal is to learn the mapping G and F without any paired word translation. To achieve this, our loss function consists of two parts: Sinkhorn distance (Cuturi, 2013) for matching the distribution of transferred embedding to its target embedding distribution; and a back-translation loss for preventing degenerated transformation.

3.1 Sinkhorn Distance

3.1.1 Definition

Sinkhorn distance is a recently proposed distance between probability distributions. We use the Sinkhorn distance to measure the closeness between $G(X)$ and Y, and also between $F(Y)$ and X. During the training, our model optimizes G and F for lower Sinkhorn distance to make the transferred embeddings match the distribution of the target embeddings. Here we only illustrate the Sinkhorn distance between $G(X)$ and Y, the derivation for $F(Y)$ and X is very similar. Although the vocabulary sizes of two languages could be different, we are able to sample mini-batches of equal size from $G(X)$ and Y. therefore we assume $n = m$ in the following derivation.

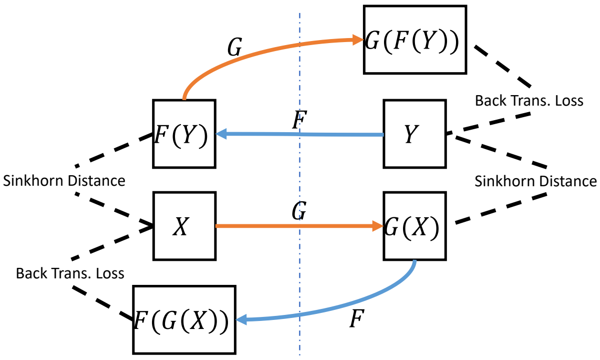

Figure 1: The model takes monolingual word embedding X and Y as input. G and F are embedding transfer functions parameterized by a neural network, which are represented by solid arrows. The dashed lines indicate the input for our objective losses, namely the Sinkhorn distance and back-translation loss

.

To compute Sinkhorn distance, we firstly compute a distance matrix $M^{(G)} \in \mathbb{R}^{n \times m}$ between $G(X)$ and Y where $M_{ij}^{(G)}$ is the distance measure between $G(x_i)$ and y_j. The superscript on $M^{(G)}$ indicates the distance that depends on a parameterized transformation G. For instance, if we choose Euclidean distance as a measure (see Section 3.1.3 for more discussions), we will have

$$M_{ij}^{(G)} = \|G(x_i) - y_j\|_2.$$

Given the distance matrix, the Sinkhorn distance between $P_{G(X)}$ and P_Y is defined as:

$$d_{sh}(G) := \min_{P \in U_\alpha(r,c)} \langle P, M^{(G)} \rangle \qquad (1)$$

where $\langle \cdot, \cdot \rangle$ is the Forbenius dot-product and $U_\alpha(r, c)$ is an entropy constrained transport poly-

Algorithm 1 Computation of Sinkhorn Distance $d_{sh}(G)$

```
procedure SINKHORN($M^{(G)}, r, c, \lambda, I$)
    $K^{(G)} := e^{-\lambda M^{(G)}}$
    $v = \mathbb{1}_m / m$            ▷ normalized one vector
    $i = 0$
    while $i < I$ do                  ▷ iterate for $I$ times
        $u = r./K^{(G)} v$
        $v = c./{K^{(G)}}^T u$
        $i = i + 1$
    $d_{sh}(G) = u^T((K^{(G)} \otimes M^{(G)})v)$
    return $d_{sh}(G)$                ▷ The Sinkhorn distance
```

tope, defined as

$$U_\alpha(r,c) = \{P \in {\mathbb{R}_+}^{n \times m} | P\mathbb{1}_m = r, P^T \mathbb{1}_n = c, \\ h(P) \le h(r) + h(c) - \alpha\} \qquad (2)$$

Note that P is non-negative and the first two constraints make its element-wise sum be 1. Therefore, P can be seen as a set of probability distributions. The same applies for r and c since they are frequencies. h is the entropy function defined on any probability distributions and α is a hyperparameter to choose. For any probabilistic matrix $P \in U_\alpha(r,c)$, it can be viewed as the joint probability of $(G(X), Y)$. The first two constraints ensure that P has marginal distribution on $G(X)$ as $P_{G(X)}$ and on Y as P_Y. We can also view P_{ij} as the evidence for establishing a translation between word vector x_i and word vector y_j.

An intuitive interpretation of equation (1) is that we are trying to find the optimal transport probability P under the entropy constraint such that the total distance to transport from $G(X)$ to Y is minimized.

3.1.2 Computing Sinkhorn Distance $d_{sh}(G)$

Cuturi (2013) showed that the optimal solution of formula (1) has the form $P^* = \mathbf{diag}(u) K \mathbf{diag}(v)$, where u and v are some non-negative vectors and $K^{(G)} := e^{-\lambda M^{(G)}}$; λ is the Lagrange multiplier for the entropic constraint in 2 and each α in Equation (1) has one corresponding λ. The Sinkhorn distance can be efficiently computed by a matrix scaling algorithm. We present the pseudo code in Algorithm 1. Note that the computation of $d_{sh}(G)$ only requires matrix-vector multiplication. Therefore, we can compute and back propagate the gradient of $d_{sh}(G)$ with regards to the parameters in G using standard deep learning libraries. We show our implementation details in Section 3.4 and supplementary material.

3.1.3 Choice of the Distance Metric

In Section 3.1.1, we used the Euclidean distance of vector pairs to define $M^{(G)}$ and Sinkhorn distance $d_{sh}(G)$. However, in our preliminary experiment, we found that Euclidean distance of unnormalized vectors gave poor performance. Therefore, following the common practice, we normalize all word embedding vectors to have a unit L2 norm in the construction of $M^{(G)}$.

As pointed out in Theorem 1 of Cuturi (2013), $M^{(G)}$ must be a valid metric in order to make $d_{sh}(G)$ a valid metric. For example, the commonly used cosine distance, which is defined as $CosDist(a,b) = 1 - cos(a,b)$, is not a valid metric because it does not satisfy triangle inequality [1]. Thus, for constructing $M^{(G)}$, we propose the square root cosine distance ($SqrtCosDist$) below:

$$SqrtCosDist(a,b) := \sqrt{2 - 2cos(a,b)} \qquad (3)$$

$$M_{ij}^{(G)} = SqrtCosDist(G(x_i), y_j) \qquad (4)$$

Theorem 1. *$SqrtCosDist$ is a valid metric.*

Proof. $\forall a, b \in \mathbb{R}^d$, let $\hat{a} = \frac{a}{\|a\|}$, $\hat{b} = \frac{b}{\|b\|}$. We have $cos(a,b) = \langle \hat{a}, \hat{b} \rangle$ and $\langle \hat{a}, \hat{a} \rangle = \langle \hat{b}, \hat{b} \rangle = 1$. Then

$$\begin{aligned} SqrtCosDist(a,b) &= \sqrt{2 - 2cos(a,b)} \\ &= \sqrt{\langle \hat{a}, \hat{a} \rangle + \langle \hat{b}, \hat{b} \rangle - 2\langle \hat{a}, \hat{b} \rangle} \\ &= \sqrt{\langle \hat{a} - \hat{b}, \hat{a} - \hat{b} \rangle} \\ &= \|\hat{a} - \hat{b}\| \end{aligned}$$

Obviously, the last term is the Euclidean distance between normalized input vectors $\hat{a}$ and $\hat{b}$. Since Euclidean distance is a valid metric, it follows that $SqrtCosDist$ satisfies all the axioms for a valid metric. $\square$

3.2 Objective Function

Given enough capacity, G is capable to transfer X to Y for arbitrary word-to-word mappings. To ensure that, we learn a meaningful translation and also to regularize the search space of possible transformations, we enforce the word embedding after the forward and the backward transformation

---

[1] If we select $a = [1,0], b = [\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}], c = [0,1]$ We have $CosDist(a,c) \geq CosDist(a,b) + CosDist(b,c)$, which violates the triangle inequality.

should not diverge much from its original direction. We simply choose the back-translation loss based on the cosine similarity:

$$d_{bt}(G, F) = \sum_i 1 - cos(x_i, F(G(x_i))) + \sum_j 1 - cos(y_i, G(F(y_i))) \quad (5)$$

where cos is the cosine similarity.

Putting everything together, we minimize the following objective function.

$$L_{X,Y,r,c}(G, F) = d_{sh}(G) + d_{sh}(F) + \beta d_{bt}(G, F) \quad (6)$$

where hyper-parameter β controls the relative weight of the last term against the first two terms in the objective function. By definition, computation of $d_{sh}(G)$ or $d_{sh}(F)$ involves another minimization problem as shown in Equation (1). We solve it using the matrix scaling algorithm in Section 3.1.2, and treat $d_{sh}(G)$ as a deterministic and differentiable function of parameters in G. The same holds for $d_{sh}(F)$ and F.

3.3 Wasserstein GAN Training for Good Initial Point

In preliminary experiments, we found that our objective 6 is sensitive to the initialization of the weight in G and F in the purely unsupervised setting. It requires a good initial setting of the parameters to avoid getting stuck in the poor local minimal. To address this sensitivity issue, we employed a similar approach as in (Zhang et al., 2017b; Aldarmaki et al., 2018) to firstly used an adversarial training approach to learn G and F and use them as the initial point for training our full objective 6. More specifically, we choose to minimize the optimal transport distance below.

$$d_{ot}(G) := \min_{P \in U(r,c)} \langle P, M^{(G)} \rangle \quad (7)$$

U is the transport polytope without entropy constraint, defined as follows.

$$U = \{P \in {\mathbb{R}_+}^{n \times m} | P\mathbb{1}_m = r, P^T\mathbb{1}_n = c\} \quad (8)$$

We optimize the distance above by its dual form and through adversarial training, which is also known as Wasserstein GAN (WGAN) (Arjovsky et al., 2017). We applied the optimization trick proposed by Gulrajani et al. (2017).

Although the first phase of adversarial training could be unstable, and the performance is lower than using the Sinkhorn distance, the adversarial training narrows down the search space of model parameters and boosting the training of our proposed model.

3.4 Implementation

We implemented transformation G and F by a linear transformation. The dimension of the input and output are the same with the word embedding dimension d.[2] For all the experiments in the subsequent section, the β in (6) was set to be 0.1. For hyper-parameters from the computation of Sinkhorn distance, we choose $\lambda = 10$ and run the matrix scaling algorithm for 20 iterations. Due to the space constraint, a detailed implementation description is presented in the supplementary material. The code of our implementation is publicly available [3].

4 Experiments

We conducted an evaluation of our approach in comparison with state-of-the-art supervised/unsupervised methods on several evaluation benchmarks for bilingual lexicon induction (Task 1) and word similarity prediction (Task 2). We include our main results in this section and report the ablation study in the supplementary material.

4.1 Data

4.1.1 Monolingual Word Embedding Data

All the methods being evaluated in both tasks take monolingual word embedding in each language as the input data. We use publicly available pretrained word embeddings trained on Wikipedia articles: (1) a smaller set of word embeddings of dimension 50 trained on comparable Wikipedia dump in five languages (Zhang et al., 2017a)[4] and (2) a larger set of word embeddings of dimension 300 trained on Wikipedia dump in 294 languages (Bojanowski et al., 2016)[5]. For conve-

---

[2] We tried more complex non-linear transformations for G and F. The performance is slightly worse than the linear case.

[3] Our implementation https://github.com/xrc10/unsup-cross-lingual-embedding-transfer

[4] Available at http://nlp.csai.tsinghua.edu.cn/~zm/UBiLexAT

[5] Available at https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

nience, we name the two sets **WE-Z** and **WE-C**, respectively.

4.1.2 Bilingual Lexicon Data

We need true translation pairs of words for evaluating methods in bilingual lexicon induction (Task 1). We followed previous studies and prepared two datasets below.

LEX-Z: Zhang et al. (2017a) constructed the bilingual lexicons from various resources. Since their ground truth word pairs are not released, we followed their procedure, crawled bilingual dictionaries and randomly separated them into the training and testing set of equal size.[6] Note that our proposed method did not utilize the training set. It was only used by supervised baseline methods described in Section 4.2. There are eight language pairs (order counted); the corresponding dataset statistics are summarized in Table 1. We use WE-Z embeddings in this dataset.

LEX-C: This lexicon was constructed by Conneau et al. (2017) and contains more translation pairs than LEX-Z. They divided them into training and testing set. We run our model and the baseline methods on 16 language pairs. For each language pair, the training set contains $5,000$ unique query words and the testing set has $1,500$ query words. We followed Conneau et al. (2017) and set the search space of candidate translations to be the $200,000$ most frequent words in each target language. We use WE-C embeddings in this dataset.

4.1.3 Bilingual Word Similarity Data

For bilingual word similarity prediction (Task 2) we need the true labels for evaluation. Following Conneau et al. (2017), we used the SemEval 2017 competition dataset, where human annotators measured the cross-lingual similarity of nominal word pairs according to the five-point Likert scale. This dataset contains word pairs across five languages: English (en), German (de), Spanish (es), Italian (it), and Farsi (fa). Each language pair has about 1,000 word pairs annotated with a real similarity score ranging from 0 to 4.

4.2 Baseline Methods

We evaluated the same set of supervised and unsupervised baselines for comparative evaluation in both Task 1 and Task 2. The supervised baselines include the methods of Shigeto et al. (2015); Zhang et al. (2016); Artetxe et al. (2016); Xing et al. (2015); Mikolov et al. (2013); Artetxe et al. (2017).[7] We fed all the supervised methods with the bilingual dictionaries in the training portions of the LEX-Z and LEX-C datasets, respectively.

		# tokens	vocab. size	bi. lex. size
tr-en	tr	6m	7,482	18,404
	en	28m	13,220	27,327
es-en	es	61m	4,774	3,482
	en	95m	6,637	10,772
zh-en	zh	21m	3,349	54,170
	en	53m	5,154	51,375
it-en	it	73m	8,490	4,999
	en	93m	6,597	11,812

Table 1: The statistics of LEX-Z. The languages are Spanish (es), French (fr), Chinese (zh), Turkish (tr) and English (en). Number of tokens is the size of training corpus of WE-Z. The bilingual lexicon size means the number of unique words of a language in the gold bilingual lexicons.

For unsupervised baselines we include the methods of Zhang et al. (2017a) and Conneau et al. (2017), whose source code is publicly available as provided by the authors.[8]

4.3 Results in Bilingual Lexicons Induction (Task 1)

Bilingual lexicon induction is a task to induce a translation in the target language for each query word in the source language. After the query word and the target-language words are represented in the same embedding space (or after our system maps the query word from the source embedding space to the target embedding space), the k nearest target words are retrieved based on their cosine similarity scores with respect to the query vector. If the k retrieved target words contain any valid translation according to the gold bilingual lexicon, the translation (retrieval) is considered successful. The fraction of the correctly translated source words in the test set is defined as $accuracy@k$,

---

[6] The bilingual dictionaries we crawled are submitted as supplementary material.

[7] The implementations are available from `https://github.com/artetxem/vecmap`.

[8] We used implementation by Zhang et al. (2017a) from `http://nlp.csai.tsinghua.edu.cn/~zm/UBiLexAT` and that of Conneau et al. (2017) from `https://github.com/facebookresearch/MUSE`

	Methods	tr-en	en-tr	es-en	en-es	zh-en	en-zh	it-en	en-it
Supervised	Mikolov et al. (2013)	19.41	10.81	68.73	41.19	45.88	**45.37**	59.83	**41.26**
	Zhang et al. (2016)	23.39	**11.07**	72.36	41.19	48.01	42.66	**63.19**	40.37
	Xing et al. (2015)	24.00	10.78	71.92	41.02	48.10	42.90	62.81	40.43
	Shigeto et al. (2015)	**26.56**	8.52	72.23	37.80	**49.95**	38.15	63.14	35.63
	Artetxe et al. (2016)	23.49	10.74	71.98	41.12	48.01	42.66	63.14	40.28
	Artetxe et al. (2017)	22.88	10.78	**72.61**	**41.62**	47.54	42.82	61.32	39.63
Unsupervised	Conneau et al. (2017)	4.09	1.41	60.16	33.58	41.98	34.70	26.98	15.47
	Zhang et al. (2017a)	15.83	7.41	63.41	37.73	42.08	41.26	54.75	37.17
	Ours	**23.29**	**9.96**	**73.05**	**41.95**	**49.03**	**44.63**	**61.42**	**39.63**

Table 2: The accuracy@k scores of all methods in bilingual lexicon induction on **LEX-Z**. The best score for each language pair is bold-faced for the supervised and unsupervised categories, respectively. Language pair "A-B" means query words are in language A and the search space of word translations is in language B. Languages are paired among **English(en), Turkish (tr), Spanish (es), Chinese (zh) and Italian (it)**.

	Methods	bg-en	en-bg	ca-en	en-ca	sv-en	en-sv	lv-en	en-lv
Supervised	Mikolov et al. (2013)	44.80	**48.47**	57.73	**66.20**	43.73	**63.73**	26.53	**28.93**
	Zhang et al. (2016)	50.60	39.73	63.40	58.73	50.87	53.93	34.53	22.87
	Xing et al. (2015)	50.33	40.00	63.40	58.53	51.13	53.73	34.27	21.60
	Shigeto et al. (2015)	**61.00**	33.80	**69.33**	53.60	**61.27**	41.67	**42.20**	13.87
	Artetxe et al. (2016)	53.27	43.40	65.27	60.87	54.07	55.93	35.80	26.47
	Artetxe et al. (2017)	47.27	34.40	61.27	56.73	38.07	44.20	24.07	12.20
Unsupervised	Conneau et al. (2017)	26.47	13.87	41.00	33.07	24.27	24.47	-	-
	Zhang et al. (2017a)	-	-	-	-	-	-	-	-
	Ours	**50.33**	**34.27**	**58.60**	**54.60**	**48.13**	**50.47**	**27.73**	**13.53**

Table 3: The accuracy@k scores of all methods in bilingual lexicon induction on **LEX-C**. The best score for each language pair is bold-faced for the supervised and unsupervised categories, respectively. Languages are paired among **English(en), Bulgarian(bg), Catalan(ca), Swedish(sv) and Latvian(lv)**. "-" means that during the training time, the model failed to converge to reasonable local minimal and hence the result is omitted in the table.

which is conventional metric in benchmark evaluations.

Table 2 shows the accuracy@1 for all the methods on LEX-Z in our evaluation. We can see that our method outperformed the other unsupervised baselines by a large margin on all the eight language pairs. Compared with the supervised methods, our method is still competitive (the best or the second-best scores on four out of eight language pairs), even ours does not require cross-lingual supervision. Also, we notice the performance variance over different language pairs. Our method outperforms all the methods (supervised and unsupervised combined) on the English-Spanish (en-es) pair, perhaps for the reasons that these two languages are most similar to each other, and that the monolingual word embeddings for this pair in the comparable corpus are better aligned than the other language pairs. On the other hand, all the methods including ours have the worst performance on the English-Turkish (en-tr) pair. Another observation is the performance differences in the two directions of the language pair. For example, the performance of it-en is better than en-it for all methods in table 2. A part of the reason is that there are more unique English words than non-English words in the evaluation set. This would cause direction "xx-en" to be easier than "en-xx" because there are often multiple valid ground truth English translations for each query in "xx". But the same may not hold for the opposite direction of "en-xx". Nevertheless, the relative performance of our method compared to others is quite robust over different language pairs and different directions of

	Methods	de-en	en-de	es-en	en-es	fr-en	en-fr	it-en	en-it
Supervised	Mikolov et al. (2013)	61.93	**73.07**	74.00	**80.73**	71.33	**82.20**	68.93	**77.60**
	Zhang et al. (2016)	67.67	69.87	77.27	78.53	76.07	78.20	72.40	73.40
	Xing et al. (2015)	67.73	69.53	77.20	78.60	76.33	78.67	72.00	73.33
	Shigeto et al. (2015)	**71.07**	63.73	**81.07**	74.53	**79.93**	73.13	**76.47**	68.13
	Artetxe et al. (2016)	69.13	72.13	78.27	80.07	77.73	79.20	73.60	74.47
	Artetxe et al. (2017)	68.07	69.20	75.60	78.20	74.47	77.67	70.53	71.67
Unsupervised	Conneau et al. (2017)	**69.87**	**71.53**	**78.53**	79.40	**77.67**	**78.33**	**74.60**	**75.80**
	Zhang et al. (2017a)	-	-	-	-	-	-	-	-
	Ours	67.00	69.33	77.80	**79.53**	75.47	77.93	72.60	73.47

Table 4: The accuracy@k scores of all methods in bilingual lexicon induction on **LEX-C**. The best score for each language pair is bold-faced for the supervised and unsupervised categories, respectively. Languages are paired among **English (en), German (de), Spanish (es), French (fr) and Italian (it)**. ”-” means that during the training time, the model failed to converge to reasonable local minimal and hence the result is omitted in the table.

translation.

Table 3 and Table 4 summarize the results of all the methods on the LEX-C dataset. Several points may be worth noticing. Firstly, the performance scores on LEX-C are not necessarily consistent with those on LEX-Z (Table 2) even if the methods and the language pairs are the same; this is not surprising as the two datasets differ in query words, word embedding quality, and training-set sizes. Secondly, the performance gap between the best supervised methods and the best unsupervised methods in both Table 3 and Table 4 are larger than that in Table 2. This is attributed to the large amount of good-quality supervision in LEX-C (5,000 human-annotated word pairs) and the larger candidate size in WE-C (200, 000 candidates). Thirdly, the average performance in Table 3 is lower than that in Table 4, indicating that the language pairs in the former are more difficult than that in the latter. Nevertheless, we can see that our method has much stronger performance than other unsupervised methods in Table 3, i.e., on the harder language pairs, and that it performed comparably with the model by Conneau et al. (2017) in Table 4 on the easier language pairs. Combining all these observations, we see that our method is highly robust for various language pairs and under different training conditions.

4.4 Results in Cross-lingual Word Similarity Prediction (Task 2)

We evaluate models on cross-lingual word similarity prediction (Task 2) to measure how much the predicted cross-language word similarities match the ground truth annotated by humans. Following the convention in benchmark evaluations for this task, we compute the Pearson correlation between the model-induced similarity scores and the human-annotated similarity scores over testing word pairs for each language pair. A higher correlation score with the ground truth represents the better quality of induced embeddings. All systems use the cosine similarity between the transformed embedding of each query and the word embedding of its paired translation as the predicted similarity score.

	Methods	de-en	es-en	fa-en	it-en
Supervised	Mikolov et al. (2013)	0.71	0.72	0.68	0.71
	Zhang et al. (2016)	0.71	0.71	0.69	0.71
	Xing et al. (2015)	0.72	0.71	0.69	0.72
	Shigeto et al. (2015)	0.72	**0.72**	0.69	0.71
	Artetxe et al. (2016)	**0.73**	**0.72**	**0.70**	**0.73**
	Artetxe et al. (2017)	0.70	0.70	0.67	0.71
Unsupervised	Conneau et al. (2017)	**0.71**	**0.71**	**0.68**	**0.71**
	Zhang et al. (2017a)	-	-	-	-
	Ours	**0.71**	**0.71**	0.67	**0.71**

Table 5: Performance (measured using Pearson correlation) of all the methods in cross-lingual semantic word similarity prediction on the benchmark data from Conneau et al. (2017). The best score in the supervised and unsupervised category is bold-faced, respectively. The languages include English (en), German (de), Spanish (es), Persian (fa) and Italian (it). ”-” means that the model failed to converge to reasonable local minimal during the training process.

Table 5 summarizes the performance of all the methods in cross-lingual word similarity prediction. We can see that the unsupervised methods, including ours, perform equally well as the supervised methods, which is highly encouraging.

5 Conclusion

In this paper, we presented a novel method for cross-lingual transformation of monolingual embeddings in an unsupervised manner. By simultaneously optimizing the bi-directional mappings w.r.t. Sinkhorn distances and back-translation losses on both ends, our model enjoys its prediction power as well as robustness, with the impressive performance on multiple evaluation benchmarks. For future work, we would like to extend this work in the semi-supervised setting where insufficient bilingual dictionaries are available.

Acknowledgments

We thank the reviewers for their helpful comments. This work is supported in part by Defense Advanced Research Projects Agency Information Innovation Office (I2O), the Low Resource Languages for Emergent Incidents (LORELEI) Program, Issued by DARPA/I2O under Contract No. HR0011-15-C-0114, and in part by the National Science Foundation (NSF) under grant IIS-1546329.

References

Aldarmaki, H., Mohan, M., and Diab, M. (2018). Unsupervised word mapping using structural similarities in monolingual embeddings. *Transactions of the Association for Computational Linguistics*, 6:185–196.

Ammar, W., Mulcaire, G., Tsvetkov, Y., Lample, G., Dyer, C., and Smith, N. A. (2016). Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Artetxe, M., Labaka, G., and Agirre, E. (2016). Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Artetxe, M., Labaka, G., and Agirre, E. (2017). Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 451–462.

Barone, A. V. M. (2016). Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. *arXiv preprint arXiv:1608.02996*.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Cao, H., Zhao, T., Zhang, S., and Meng, Y. (2016). In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 1818–1827.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2017). Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300.

Duong, L., Kanayama, H., Ma, T., Bird, S., and Cohn, T. (2016). Learning crosslingual word embeddings without bilingual corpora. *arXiv preprint arXiv:1606.09403*.

Faruqui, M. and Dyer, C. (2014). Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

Fung, P. (1996). Compiling Bilingual Lexicon Entries From a Non-Parallel English-Chinese Corpus A Non-parallel Corpus of Chinese and English. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 173–183.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Gouws, S., Bengio, Y., and Corrado, G. (2015). Bilbowa: Fast bilingual distributed representations without word alignments. In *International Conference on Machine Learning*, pages 748–756.

Gouws, S. and Søgaard, A. (2015). Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779.

Haghighi, A., Liang, P., Berg-Kirkpatrick, T., and Klein, D. (2008). Learning bilingual lexicons from monolingual corpora. *Proceedings of ACL-08: Hlt*, pages 771–779.

He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T., and Ma, W.-Y. (2016). Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Hermann, K. M. and Blunsom, P. (2014). Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*.

Kočiskỳ, T., Hermann, K. M., and Blunsom, P. (2014). Learning bilingual word representations by marginalizing alignments. *arXiv preprint arXiv:1405.0947*.

Lu, A., Wang, W., Bansal, M., Gimpel, K., and Livescu, K. (2015). Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Rapp, R. (1995). Identifying Word Translations in Non-Parallel Texts. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 320–322.

Shen, T., Lei, T., Barzilay, R., and Jaakkola, T. (2017). Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Shigeto, Y., Suzuki, I., Hara, K., Shimbo, M., and Matsumoto, Y. (2015). Ridge regression, hubness, and zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–151. Springer.

Smith, S. L., Turban, D. H., Hamblin, S., and Hammerla, N. Y. (2017). Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Søgaard, A., Agić, Ž., Alonso, H. M., Plank, B., Bohnet, B., and Johannsen, A. (2015). Inverted indexing for cross-lingual nlp. In *The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*.

Taigman, Y., Polyak, A., and Wolf, L. (2016). Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*.

Vulić, I. and Korhonen, A. (2016). On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 247–257, Berlin, Germany. Association for Computational Linguistics.

Vulić, I. and Moens, M.-F. (2016). Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994.

Xing, C., Wang, D., Liu, C., and Lin, Y. (2015). Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Yi, Z., Zhang, H., Tan, P., and Gong, M. (2017). Dualgan: Unsupervised dual learning for image-to-image translation. *arXiv preprint*.

Zhang, M., Liu, Y., Luan, H., and Sun, M. (2017a). Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1959–1970.

Zhang, M., Liu, Y., Luan, H., and Sun, M. (2017b). Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945.

Zhang, Y., Gaddy, D., Barzilay, R., and Jaakkola, T. (2016). Ten pairs to tag-multilingual pos tagging via coarse mapping between embeddings. Association for Computational Linguistics.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*.

XNLI: Evaluating Cross-lingual Sentence Representations

Alexis Conneau, Ruty Rinott, Guillaume Lample, Holger Schwenk, Ves Stoyanov
Facebook AI
aconneau,ruty,glample,schwenk,ves@fb.com

Adina Williams, Samuel R. Bowman
New York University
adinawilliams,bowman@nyu.edu

Abstract

State-of-the-art natural language processing systems rely on supervision in the form of annotated data to learn competent models. These models are generally trained on data in a single language (usually English), and cannot be directly used beyond that language. Since collecting data in every language is not realistic, there has been a growing interest in cross-lingual language understanding (XLU) and low-resource cross-language transfer. In this work, we construct an evaluation set for XLU by extending the development and test sets of the Multi-Genre Natural Language Inference Corpus (MultiNLI) to 15 languages, including low-resource languages such as Swahili and Urdu. We hope that our dataset, dubbed XNLI, will catalyze research in cross-lingual sentence understanding by providing an informative standard evaluation task. In addition, we provide several baselines for multilingual sentence understanding, including two based on machine translation systems, and two that use parallel data to train aligned multilingual bag-of-words and LSTM encoders. We find that XNLI represents a practical and challenging evaluation suite, and that directly translating the test data yields the best performance among available baselines.

1 Introduction

Contemporary natural language processing systems typically rely on annotated data to learn how to perform a task (e.g., classification, sequence tagging, natural language inference). Most commonly the available training data is in a single language (e.g., English or Chinese) and the resulting system can perform the task only in the training language. In practice, however, systems used in major international products need to handle inputs in many languages. In these settings, it is nearly impossible to annotate data in all languages that a system might encounter during operation.

A scalable way to build multilingual systems is through cross-lingual language understanding (XLU), in which a system is trained primarily on data in one language and evaluated on data in others. While XLU shows promising results for tasks such as cross-lingual document classification (Klementiev et al., 2012; Schwenk and Li, 2018), there are very few, if any, XLU benchmarks for more difficult language understanding tasks like natural language inference. Large-scale natural language inference (NLI), also known as recognizing textual entailment (RTE), has emerged as a practical test bed for work on sentence understanding. In NLI, a system is tasked with reading two sentences and determining whether one entails the other, contradicts it, or neither (*neutral*). Recent crowdsourced annotation efforts have yielded datasets for NLI in English (Bowman et al., 2015; Williams et al., 2017) with nearly a million examples, and these have been widely used to evaluate neural network architectures and training strategies (Rocktäschel et al., 2016; Gong et al., 2018; Peters et al., 2018; Wang et al., 2018), as well as to train effective, reusable sentence representations (Conneau et al., 2017; Subramanian et al., 2018; Cer et al., 2018).

In this work, we introduce a benchmark that we call the Cross-lingual Natural Language Inference corpus, or XNLI, by extending these NLI corpora to 15 languages. XNLI consists of 7500 human-annotated development and test examples in NLI three-way classification format in English, French, Spanish, German, Greek, Bulgarian, Russian, Turkish, Arabic, Vietnamese, Thai, Chinese, Hindi, Swahili and Urdu, making a total of 112,500 annotated pairs. These languages span several language families, and with the inclusion of Swahili and Urdu, include two lower-resource languages as well.

Because of its focus on development and test

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2475–2485
Brussels, Belgium, October 31 - November 4, 2018.

Language	Premise / Hypothesis	Genre	Label
English	You don't have to stay there. You can leave.	Face-To-Face	Entailment
French	La figure 4 montre la courbe d'offre des services de partage de travaux. Les services de partage de travaux ont une offre variable.	Government	Entailment
Spanish	Y se estremeció con el recuerdo. El pensamiento sobre el acontecimiento hizo su estremecimiento.	Fiction	Entailment
German	Während der Depression war es die ärmste Gegend, kurz vor dem Hungertod. Die Weltwirtschaftskrise dauerte mehr als zehn Jahre an.	Travel	Neutral
Swahili	Ni silaha ya plastiki ya moja kwa moja inayopiga risasi. Inadumu zaidi kuliko silaha ya chuma.	Telephone	Neutral
Russian	И мы занимаемся этим уже на протяжении 85 лет. Мы только начали этим заниматься.	Letters	Contradiction
Chinese	让我告诉你，美国人最终如何看待你作为独立顾问的表现。 美国人完全不知道您是独立律师。	Slate	Contradiction
Arabic	تحتاج الوكالات لأن تكون قادرة على قياس مستويات النجاح. لا يمكن للوكالات أن تعرف ما إذا كانت ناجحة أم لا.	Nine-Eleven	Contradiction

Table 1: Examples (premise and hypothesis) from various languages and genres from the XNLI corpus.

data, this corpus is designed to evaluate *cross-lingual* sentence understanding, where models have to be trained in one language and tested in different ones.

We evaluate several approaches to cross-lingual learning of natural language inference that leverage parallel data from publicly available corpora at training time. We show that parallel data can help align sentence encoders in multiple languages such that a classifier trained with English NLI data can correctly classify pairs of sentences in other languages. While outperformed by our machine translation baselines, we show that this alignment mechanism gives very competitive results.

A second practical use of XNLI is the evaluation of pretrained general-purpose language-universal sentence encoders. We hope that this benchmark will help the research community build multilingual text embedding spaces. Such embeddings spaces will facilitate the creation of multilingual systems that can transfer across languages with little or no extra supervision.

The paper is organized as follows: We next survey the related literature on cross-lingual language understanding. We then describe our data collection methods and the resulting corpus in Section 3. We describe our baselines in Section 4, and finally present and discuss results in Section 5.

2 Related Work

Multilingual Word Embeddings Much of the work on multilinguality in language understanding has been at the word level. Several approaches have been proposed to learn cross-lingual word representations, i.e., word representations where translations are close in the embedding space. Many of these methods require some form of supervision (typically in the form of a small bilingual lexicon) to align two sets of source and target embeddings to the same space (Mikolov et al., 2013a; Kociský et al., 2014; Faruqui and Dyer, 2014; Ammar et al., 2016). More recent studies have showed that cross-lingual word embeddings can be generated with no supervision whatsoever (Artetxe et al., 2017; Conneau et al., 2018).

Sentence Representation Learning Many approaches have been proposed to extend word embeddings to sentence or paragraph representations (Le and Mikolov, 2014; Wieting et al., 2016; Arora et al., 2017). The most straightforward way to generate sentence embeddings is to consider an average or weighted average of word representations, usually referred to as continuous bag-of-words (CBOW). Although naïve, this method often provides a strong baseline. More sophisticated approaches—such as the unsupervised SkipThought model of Kiros et al. (2015) that extends the skip-gram model of Mikolov et al. (2013b) to the sentence level—have been proposed to capture syntactic and semantic dependencies inside sentence representations. While these fixed-size sentence embedding methods have been outperformed by their supervised counterparts (Conneau et al., 2017; Subramanian et al., 2018), some recent developments have shown that pretrained language models can also transfer very well, either when the hidden states of the model are used as contextualized word vectors (Peters et al., 2018), or when the full model is fine-tuned on transfer tasks (Radford et al.; Howard and Ruder, 2018).

Multilingual Sentence Representations There has been some effort on developing multilingual sentence embeddings. For example, Chandar et al. (2013) train bilingual autoencoders with the objective of minimizing reconstruction error between two languages. Schwenk et al. (2017) and España-Bonet et al. (2017) jointly train a sequence-to-sequence MT system on multiple languages to learn a shared multilingual sentence embedding space. Hermann and Blunsom (2014) propose a compositional vector model involving unigrams and bigrams to learn document level representations. Pham et al. (2015) directly train embedding representations for sentences with no attempt at compositionality. Zhou et al. (2016) learn bilingual document representations by minimizing the Euclidean distance between document representations and their translations.

Cross-lingual Evaluation Benchmarks The lack of evaluation benchmark has hindered the development of such multilingual representations. Most previous approaches use the Reuters cross-lingual document classification corpus Klementiev et al. (2012) for evaluation. However, the classification in this corpus is done at document level, and, as there are many ways to aggregate sentence embeddings, the comparison between different sentence embeddings is difficult. Moreover, the distribution of classes in the Reuters corpus is highly unbalanced, and the dataset does not provide a development set in the target language, further complicating experimental comparisons.

In addition to the Reuters corpus, Cer et al. (2017) propose sentence-level multilingual training and evaluation datasets for semantic textual similarity in four languages. There have also been efforts to build multilingual RTE datasets, either through translating English data (Mehdad et al., 2011), or annotating sentences from a parallel corpora (Negri et al., 2011). More recently, Agic and Schluter (2017) provide a corpus, that is very complementary to our work, of human translations for 1332 pairs of the SNLI data into Arabic, French, Russian, and Spanish. Among all these benchmarks, XNLI is the first large-scale corpus for evaluating sentence-level representations on that many languages.

In practice, cross-lingual sentence understanding goes beyond translation. For instance, Mohammad et al. (2016) analyze the differences in human sentiment annotations of Arabic sentences and their English translations, and conclude that most of them come from cultural differences. Similarly, Smith et al. (2016) show that most of the degradation in performance when applying a classification model trained in English to Spanish data translated to English is due to cultural differences. One of the limitations of the XNLI corpus is that it does not capture these differences, since it was obtained by translation. We see the XNLI evaluation as a necessary step for multilingual NLP before tackling the even more complex problem of domain-adaptation that occurs when handling this the change in style from one language to another.

3 The XNLI Corpus

Because the test portion of the Multi-Genre NLI data is private, the Cross-lingual NLI Corpus (XNLI) is based on new English NLI data. To collect the core English portion, we follow precisely the same crowdsourcing-based procedure used for the existing Multi-Genre NLI corpus, and collect and validate 750 new examples from each of the ten text sources used in that corpus for a total of 7500 examples. With that portion in place, we create the full XNLI corpus by employing professional translators to translate it into our ten target languages. This section describes this process and the resulting corpus.

Translating, rather than generating new hypothesis sentences in each language separately, has multiple advantages. First, it ensures that the data distributions are maximally similar across languages. As speakers of different languages may have slightly different intuitions about how to fill in the supplied prompt, this allows us to avoid adding this unwanted degree of freedom. Second, it allows us to use the same trusted pool of workers as was used prior NLI crowdsourcing efforts, without the need for training a new pool of workers in each language. Third, for any premise, this process allows us to have a corresponding hypothesis in any language.

This translation approach carries with it the risk that the semantic relations between the two sentences in each pair might not be reliably preserved in translation, as Mohammad et al. (2016) observed for sentiment. We investigate this potential issue in our corpus and find that, while it does occur, it only concerns a negligible number of sentences (see Section 3.2).

3.1 Data Collection

The English Corpus Our collection procedure for the English portion of the XNLI corpus follows the same procedure as the MultiNLI corpus. We sample 250 sentences from each of the ten sources that were used in that corpus, ensuring that none of those selected sentences overlap with the distributed corpus. Nine of the ten text sources are drawn from the second release of the Open American National Corpus[1]: *Face-To-Face*, *Telephone*, *Government*, *9/11*, *Letters*, *Oxford University Press* (*OUP*), *Slate*, *Verbatim*, and *Government*. The tenth, *Fiction*, is drawn from the novel *Captain Blood* (Sabatini, 1922). We refer the reader to Williams et al. (2017) for more details on each genre.

Given these sentences, we ask the same MultiNLI worker pool from a crowdsourcing platform to produce three hypotheses for each premise, one for each possible label.

We present premise sentences to workers using the same templates as were used in MultiNLI. We also follow that work in pursuing a second validation phase of data collection in which each pair of sentences is relabeled by four other workers. For each validated sentence pair, we assign a gold label representing a majority vote between the initial label assigned to the pair by the original annotator, and the four additional labels assigned by validation annotators. We obtained a three-vote consensus for 93% of the data. In our experiments, we kept the 7% additional ones, but we mark these ones with a special label '-'.

Translating the Corpus Finally, we hire translators to translate the resulting sentences into 15 languages using the *One Hour Translation* platform. We translate the premises and hypotheses separately, to ensure that no context is added to the hypothesis that was not there originally, and simply copy the labels from the English source text. Some examples are shown in Table 1.

3.2 The Resulting Corpus

One main concern in studying the resulting corpus is to determine whether the gold label for some of the sentence pairs changes as a result of information added or removed in the translation process.

Investigating the data manually, we find an example in the Chinese translation where an entailment relation becomes a contradictory relation, while the entailment is preserved in other languages. Specifically, the term *upright* which was used in English as entailment of *standing*, was translated into Chinese as *sitting upright* thus creating a contradiction. However, the difficulty of finding such an example in the data suggests its rarity.

To quantify this observation, we recruit two bilingual annotators to re-annotate 100 examples each in both English and French following our standard validation procedure. The examples are drawn from two non-overlapping random subsets of the development data to prevent the annotators from seeing the source English text for any translated text they annotate. With no training or burn-in period, these annotators recover the English consensus label 85% of the time on the original English data and 83% of the time on the translated French, suggesting that the overall semantic relationship between the two languages has been preserved. As most sentences are relatively easy to translate, in particular the hypotheses generated by the workers, there seems to be little ambiguity added by the translator.

More broadly, we find that the resulting corpus has similar properties to the MultiNLI corpus. For all languages, on average, the premises are twice as long as the hypotheses (See Table 2). The top hypothesis words indicative of the class label – scored using the mutual information between each word and class in the corpus – are similar across languages, and overlap those of the MultiNLI corpus (Gururangan et al., 2018). For example, a translation of at least one of the words *no*, *not* or *never* is among the top two cues for contradiction in all languages.

As in the original MultiNLI corpus, we expect that cues like these ('artifacts', in Guru-

---

[1] `http://www.anc.org/`

	en	fr	es	de	el	bg	ru	tr	ar	vi	th	zh	hi	sw	ur
Premise	21.7	24.1	22.1	21.1	21.0	20.9	19.6	16.8	20.7	27.6	22.1	21.8	23.2	18.7	24.1
Hypothesis	10.7	12.4	10.9	10.8	10.6	10.4	9.7	8.4	10.2	13.5	10.4	10.8	11.9	9.0	12.3

Table 2: Average number of tokens per sentence in the XNLI corpus for each language.

rangan's terms, also observed by Poliak et al., 2018; Tsuchiya, 2018) allow a baseline system to achieve better-than-random accuracy with access only to the premise sentences. We accept this as an unavoidable property of the NLI task over naturalistic sentence pairs, and see no reason to expect that this baseline would achieve better accuracy than the relatively poor 53% seen in Gururangan et al. (2018).

4 Cross-Lingual NLI

In this section we present results with XLU systems that can serve as baselines for future work.

4.1 Translation-Based Approaches

The most straightforward techniques for XLU rely on translation systems. There are two natural ways to use a translation system: TRANSLATE TRAIN, where the training data is translated into each target language to provide data to train each classifier, and TRANSLATE TEST, where a translation system is used at test time to translate input sentences to the training language. These two methods provide strong baselines, but both present practical challenges. The former requires training and maintaining as many classifiers as there are languages, while the latter relies on computationally-intensive translation at test time. Both approaches are limited by the quality of the translation system, which itself varies with the quantity of available training data and the similarity of the language pair involved.

4.2 Multilingual Sentence Encoders

An alternative to translation is to rely on language-universal embeddings of text and build multilingual classifiers on top of these representations. If an encoder produces an embedding of an English sentence close to the embedding of its translation in another language, then a classifier learned on top of English sentence embeddings will be able to classify sentences from different languages without needing a translation system at inference time.

We evaluate two types of cross-lingual sentence encoders: (i) pretrained universal multilingual sentence embeddings based on the average of word embeddings (X-CBOW), (ii) bidirectional-LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997) sentence encoders trained on the MultiNLI training data (X-BILSTM). The former evaluates transfer learning while the latter evaluates NLI-specific encoders trained on in-domain data. Both approaches use the same alignment loss for aligning sentence embedding spaces from multiple languages which is present below. We consider two ways of extracting feature vectors from the BiLSTM: either using the initial and final hidden states (Sutskever et al., 2014), or using the element-wise max over all states (Collobert and Weston, 2008).

The first approach is commonly used as a strong baseline for monolingual sentence embeddings (Arora et al., 2017; Conneau and Kiela, 2018; Gouews et al., 2014). Concretely, we consider the English fastText word embedding space as being fixed, and fine-tune embeddings in other languages so that the average of the word vectors in a sentence is close to the average of the word vectors in its English translation. The second approach consists in learning an English sentence encoder on the MultiNLI training data along with an encoder on the target language, with the objective that the representations of two translations are nearby in the embedding space. In both approaches, an English encoder is fixed, and we train target language encoders to match the output of this encoder. This allows us to build sentence representations that belong to the same space. Joint training of encoders and parameter sharing are also promising directions to improve and simplify the alignment of sentence embedding spaces. We leave this for future work.

In all experiments, we consider encoders that output a vector of fixed size as a sentence representation. While previous work shows that performance on the NLI task can be improved by using cross-sentence attention between the premise and hypothesis (Rocktäschel et al., 2016; Gong et al., 2018), we focus on methods with fixed-size sentence embeddings.

4.2.1 Aligning Word Embeddings

Multilingual word embeddings are an efficient way to transfer knowledge from one language to another. For instance, Zhang et al. (2016) show that cross-lingual embeddings can be used to extend an English part-of-speech tagger to the cross-lingual setting, and Xiao and Guo (2014) achieve similar results in dependency parsing. Cross-lingual embeddings also provide an efficient mechanism to bootstrap neural machine translation (NMT) systems for low-resource language pairs, which is critical in the case of unsupervised machine translation (Lample et al., 2018; Artetxe et al., 2018). In that case, the use cross-lingual embeddings directly helps the alignment of sentence-level encoders. Cross-lingual embeddings can be generated efficiently using a very small amount of supervision. By using a small parallel dictionary with $n = 5000$ word pairs, it is possible to learn a linear mapping to minimize

$$W^\star = \underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_{\mathrm{F}} = UV^T,$$

where d is the dimension of the embeddings, and X and Y are two matrices of shape (d, n) that correspond to the aligned word embeddings that appear in the parallel dictionary, $O_d(\mathbb{R})$ is the group of orthogonal matrices of dimension d, and U and V are obtained from the singular value decomposition (SVD) of YX^T: $U\Sigma V^T = \mathrm{SVD}(YX^T)$. Xing et al. (2015) show that enforcing the orthogonality constraint on the linear mapping leads to better results on the word translation task.

In this paper, we use common-crawl word embeddings (Grave et al., 2018) aligned with the MUSE library of Conneau et al. (2018).

4.2.2 Universal Multilingual Sentence Embeddings

Most of the successful recent approaches for learning universal sentence representations have relied on English (Kiros et al., 2015; Arora et al., 2017; Conneau et al., 2017; Subramanian et al., 2018; Cer et al., 2018). While notable recent approaches have considered building a shared sentence encoder for multiple languages using publicly available parallel corpora (Johnson et al., 2016; Schwenk et al., 2017; España-Bonet et al., 2017), the lack of a large-scale, sentence-level semantic evaluation has limited their adoption by the community. In particular, these works do not cover the scale of languages considered in XNLI, and are limited to high-resource languages. As a baseline for the evaluation of multilingual sentence representations in the 15 languages of XNLI, we consider state-of-the-art common-crawl embeddings with a CBOW encoder. Our approach, dubbed X-CBOW, consists in fixing the English pretrained word embeddings, and fine-tuning the target (e.g., French) word embeddings so that the CBOW representations of two translations are close in embedding space. In that case, we consider our multilingual sentence embeddings as being pretrained and only learn a classifier on top of them to evaluate their quality, similar to so-called "transfer" tasks in (Kiros et al., 2015; Conneau et al., 2017) but in the multilingual setting.

4.2.3 Aligning Sentence Embeddings

Training for similarity of source and target sentences in an embedding space is conceptually and computationally simpler than generating a translation in the target language from a source sentence. We propose a method for training for cross-lingual similarity and evaluate approaches based on the simpler task of aligning sentence representations. Under our objective, the embeddings of two parallel sentences need not be identical, but only close enough in the embedding space that the decision boundary of the English classifier captures the similarity.

We propose a simple alignment loss function to align the embedding spaces of two different languages. Specifically, we train an English encoder on NLI, and train a target encoder by minimizing the loss:

$$\mathcal{L}_{\text{align}}(x, y) = sim(x, y) - \lambda(sim(x_c, y) + sim(x, y_c))$$

where (x, y) corresponds to the source and target sentence embeddings, (x_c, y_c) is a contrastive term (i.e. negative sampling), λ controls the weight of the negative examples in the loss. For a similarity measure, we use the L2 norm with $sim(x, y) = -\|x - y\|_2$. We obtain similar results using the cosine similarity. A ranking loss (Weston et al., 2011) of the form

$$\begin{aligned}\mathcal{L}_{\text{rank}}(x, y) = &\max(0, \alpha - sim(x, y) + s(x, y_c)) \; + \\ &\max(0, \alpha - sim(x, y) + s(x_c, y))\end{aligned}$$

	fr	es	de	el	bg	ru	tr	ar	vi	th	zh	hi	sw	ur
XX-En BLEU	41.2	45.8	39.3	42.1	38.7	27.1	29.9	35.2	23.6	22.6	24.6	27.3	21.3	24.4
En-XX BLEU	49.3	48.5	38.8	42.4	34.2	24.9	21.9	15.8	39.9	21.4	23.2	37.5	24.6	24.1
Word translation P@1	73.7	73.9	65.9	61.1	61.9	60.6	55.0	51.9	35.8	25.4	48.6	48.2	-	-

Table 3: BLEU scores of our translation models (XX-En) P@1 for multilingual word embeddings.

that pushes the sentence embeddings of a translation pair to be closer than the ones of negative pairs leads to very poor results in this particular case. As opposed to $\mathcal{L}_{\text{align}}$, $\mathcal{L}_{\text{rank}}$ does not encourage the embeddings of sentence pairs to be close enough so that the shared classifier can understand that these sentences have the same meaning.

We use $\mathcal{L}_{\text{align}}$ in the cross-lingual embeddings baselines X-CBOW, X-BILSTM-LAST and X-BILSTM-MAX. For X-CBOW, the encoder is pretrained (transfer-learning), while the English X-BiLSTMs are trained on NLI (in-domain). For the three methods, the English encoder is then fixed. Each of the 14 other languages have their own encoders with same architecture. These encoders are trained to "copy" the English encoder using the $\mathcal{L}_{\text{align}}$ loss and the parallel data described in section 5.2.

We only back-propagate through the target encoder when optimizing $\mathcal{L}_{\text{align}}$ such that all 14 encoders live in the same English embedding space. In these experiments, we initialize lookup tables of the LSTMs with pretrained cross-lingual embeddings discussed in Section 4.2.1.

5 Experiments and Results

5.1 Training details

We use internal translation systems to translate data between English and the 10 other languages. For TRANSLATE TEST (see Table 4), we translate each test set into English, while for the TRANSLATE TRAIN, we translate the English training data of MultiNLI[2]. To give an idea of the translation quality, we give BLEU scores of the automatic translation from the foreign language into English of the XNLI test set in Table 3.

We use pretrained 300D word embeddings and only consider the most 500,000 frequent words in the dictionary, which generally covers more than 98% of the words found in XNLI corpora. We set the number of hidden units of the BiLSTMs to 512, and use the Adam optimizer (Kingma and Ba, 2014) with default parameters. For the alignment loss, setting λ to 0.25 worked best in our experiments, and we found that the trade-off between the importance of the positive and the negative pairs was particularly important (see Table 5). When fitting the target BiLSTM encoder to the English encoder, we fine-tune the lookup table associated to the target encoder, but keep the source word embeddings fixed. The classifier is a feed-forward neural network with one hidden layer of 128 hidden units, regularized with dropout (Srivastava et al., 2014) at a rate of 0.1. For X-BiLSTMs, we perform model selection on the XNLI validation set in each target language. For X-CBOW, we keep a validation set of parallel sentences to evaluate our alignment loss. The alignment loss requires a parallel dataset of sentences for each pair of languages, which we describe next.

5.2 Parallel Datasets

We use publicly available parallel datasets to learn the alignment between English and target encoders. For French, Spanish, Russian, Arabic and Chinese, we use the United Nation corpora (Ziemski et al., 2016), for German, Greek and Bulgarian, the Europarl corpora (Koehn, 2005), for Turkish, Vietnamese and Thai, the OpenSubtitles 2018 corpus (Tiedemann, 2012), and for Hindi, the IIT Bombay corpus (Anoop et al., 2018). For all the above language pairs, we were able to gather more than 500,000 parallel sentences, and we set the maximum number of parallel sentences to 2 million. For the lower-resource languages Urdu and Swahili, the number of parallel sentences is an order of magnitude smaller than for the other languages we consider. For Urdu, we used the Bible and Quran transcriptions (Tiedemann, 2012), the OpenSubtitles 2016 and 2018 corpora (Tiedemann, 2012) and LDC2010T21, LDC2010T23 LDC corpora, and obtained a total of 64k parallel sentences. For Swahili, we were

---

[2] To allow replication of results, we share the MT translations of XNLI training and test sets.

	en	fr	es	de	el	bg	ru	tr	ar	vi	th	zh	hi	sw	ur
*Machine translation baselines (*TRANSLATE TRAIN*)*															
BiLSTM-last	71.0	66.7	67.0	65.7	65.3	65.6	65.1	61.9	63.9	63.1	61.3	65.7	61.3	55.2	55.2
BiLSTM-max	**73.7**	68.3	68.8	66.5	66.4	67.4	66.5	64.5	65.8	66.0	62.8	67.0	62.1	58.2	56.6
*Machine translation baselines (*TRANSLATE TEST*)*															
BiLSTM-last	71.0	68.3	68.7	66.9	67.3	68.1	66.2	64.9	65.8	64.3	63.2	66.5	61.8	60.1	58 .1
BiLSTM-max	**73.7**	**70.4**	**70.7**	**68.7**	**69.1**	**70.4**	**67.8**	**66.3**	**66.8**	**66.5**	**64.4**	**68.3**	**64.2**	**61.8**	**59.3**
Evaluation of XNLI multilingual sentence encoders (in-domain)															
X-BiLSTM-last	71.0	65.2	67.8	66.6	66.3	65.7	63.7	64.2	62.7	65.6	62.7	63.7	62.8	54.1	56.4
X-BiLSTM-max	**73.7**	67.7	68.7	67.7	68.9	67.9	65.4	64.2	64.8	66.4	64.1	65.8	64.1	55.7	58.4
Evaluation of pretrained multilingual sentence encoders (transfer learning)															
X-CBOW	64.5	60.3	60.7	61.0	60.5	60.4	57.8	58.7	57.5	58.8	56.9	58.8	56.3	50.4	52.2

Table 4: Cross-lingual natural language inference (XNLI) test accuracy for the 15 languages.

only able to gather 42k sentences using the Global Voices corpus and the Tanzil Quran transcription corpus[3].

5.3 Analysis

Comparing in-language performance in Table 4, we observe that, when using BiLSTMs, results are consistently better when we take the dimension-wise maximum over all hidden states (BiLSTM-max) compared to taking the last hidden state (BiLSTM-last). Unsuprisingly, BiLSTM results are better than the pretrained CBOW approach for all languages. As in Bowman et al. (2015), we also observe the superiority of BiLSTM encoders over CBOW, even when fine-tuning the word embeddings of the latter on the MultiNLI training set, thereby again confirming that the NLI task requires more than just word information. Both of these findings confirm previously published results (Conneau et al., 2017).

Table 4 shows that translation offers a strong baseline for XLU. Within translation, TRANSLATE TEST appears to perform consistently better than TRANSLATE TRAIN for all languages. The best cross-lingual results in our evaluation are obtained by the TRANSLATE TEST approach for all cross-lingual directions. Within the translation approaches, as expected, we observe that cross-lingual performance depends on the quality of the translation system. In fact, translation-based results are very well-correlated with the BLEU scores for the translation systems; XNLI performance for three of the four languages with the best translation systems (comparing absolute BLEU, Table 3) is above 70%. This performance is still about three points below the English NLI performance of 73.7%. This slight drop in performance may be related to translation error, changes in style, or artifacts introduced by the machine translation systems that result in discrepancies between the training and test data.

For cross-lingual performance, we observe a healthy gap between the English results and the results obtained on other languages. For instance, for French, we obtain 67.7% accuracy when classifying French pairs using our English classifier and multilingual sentence encoder. When using our alignment process, our method is competitive with the TRANSLATE TRAIN baseline, suggesting that it might be possible to encode similarity between languages directly in the embedding spaces generated by the encoders. However, these methods are still below the other machine translation baseline TRANSLATE TEST, which significantly outperforms the multilingual sentence encoder approach by up to 6% (Swahili). These production systems have been trained on much larger training data than the ones used for the alignment loss (section 5.2), which can partly explain the superiority of this method over the baseline. Interestingly, the two points difference in accuracy between X-BiLSTM-last and X-BiLSTM-max is maintained across languages, which suggests that having a stronger encoder in English also positively impacts the transfer results on other languages.

In Table 5, we report the validation accuracy using BiLSTM-max on three languages with different training hyper-parameters. Fine-tuning the

---

[3] `http://opus.nlpl.eu/`

	fr	ru	zh
$ft = 1, \lambda = 0.25$ [default]	68.9	66.4	67.9
$ft = 1, \lambda = 0.0$ (no negatives)	67.8	66.2	66.3
$ft = 1, \lambda = 0.5$	64.5	61.3	63.7
$ft = 0, \lambda = 0.25$	68.5	66.3	67.7

Table 5: **Validation accuracy using BiLSTM-max.** Default setting corresponds to $\lambda = 0.25$ (importance of the negative terms) and uses fine-tuning of the target lookup table (ft =1).

embeddings does not significantly impact the results, suggesting that the LSTM alone is ensuring alignment of parallel sentence embeddings. We also observe that the negative term is not critical to the performance of the model, but can lead to slight improvement in Chinese (up to 1.6%).

6 Conclusion

A typical problem in industrial applications is the lack of supervised data for languages other than English, and particularly for low-resource languages. Since annotating data in every language is not a realistic approach, there has been a growing interest in cross-lingual understanding and low-resource transfer in multilingual scenarios. In this work, we extend the development and test sets of the Multi-Genre Natural Language Inference Corpus to 15 languages, including low-resource languages such as Swahili and Urdu. Our dataset, dubbed XNLI, is designed to address the lack of standardized evaluation protocols in cross-lingual understanding, and will hopefully help the community make further strides in this area. We present several approaches based on cross-lingual sentence encoders and machine translation systems. While machine translation baselines obtained the best results in our experiments, these approaches rely on computationally-intensive translation models either at training or at test time. We found that cross-lingual encoder baselines provide an encouraging and efficient alternative, and that further work is required to match the performance of translation based methods.

Acknowledgments

This project has benefited from financial support to Samuel R. Bowman by Google, Tencent Holdings, and Samsung Research.

References

Zeljko Agic and Natalie Schluter. 2017. Baselines and test data for cross-lingual inference. CoRR, abs/1704.05347.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. In `https://arxiv.org/abs/1602.01925`.

Kunchukuttan Anoop, Mehta Pratik, and Bhattacharyya Pushpak. 2018. The iit bombay english-hindi parallel corpus. In LREC.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. ICLR.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In ACL, pages 451–462.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In ICLR.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In EMNLP.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. arXiv preprint arXiv:1803.11175.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. CoRR, abs/1708.00055.

Sarath Chandar, Mitesh M. Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. 2013. Multilingual deep learning. In NIPS, Workshop Track.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In ICML, pages 160–167. ACM.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. CoRR, abs/1803.05449.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In EMNLP, pages 670–680.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jegou. 2018. Word translation without parallel data. In ICLR.

Cristina España-Bonet, Ádám Csaba Varga, Alberto Barrón-Cedeño, and Josef van Genabith. 2017. An empirical analysis of nmt-derived interlingual embeddings and their use in parallel sentence identification. IEEE Journal of Selected Topics in Signal Processing, pages 1340–1348.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using monolingual correlation. In EACL.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. ICLR.

S. Gouews, Y. Bengio, and G. Corrado. 2014. Bilbowa: Fast bilingual distributed representations without word alignments. In ICML.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. LREC.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. CoRR, abs/1803.02324.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In ACL, pages 58–68.

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

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. arXiv preprint arXiv:1801.06146.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. arXiv preprint arXiv:1611.04558.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In Advances in neural information processing systems, pages 3294–3302.

A. Klementiev, I. Titov, and B. Bhattarai. 2012. Inducing crosslingual distributed representations of words. In Coling.

T. Kociský, K.M. Hermann, and P. Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. In ACL, pages 224–229.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In MT summit, volume 5, pages 79–86.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In ICLR.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

Yashar Mehdad, Matteo Negri, and Marcello Federico. 2011. Using bilingual parallel corpora for cross-lingual textual entailment. In ACL, HLT '11, pages 1336–1345, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. arXiv preprint arXiv:1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In NIPS, pages 3111–3119.

Saif M. Mohammad, Mohammad Salameh, and Svetlana Kiritchenko. 2016. How translation alters sentiment. J. Artif. Int. Res., 55(1):95–130.

Matteo Negri, Luisa Bentivogli, Yashar Mehdad, and Danilo Giampiccolo. 2011. Divide and conquer: Crowdsourcing the creation of cross-lingual textual entailment corpora.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In NAACL, volume 1, pages 2227–2237.

Hieu Pham, Minh-Thang Luong, and Christopher D. Manning. 2015. Learning distributed representations for multilingual text sequences. In Workshop on Vector Space Modeling for NLP.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In Proceedings of *SEM.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. ICLR.

Rafael Sabatini. 1922. Captain Blood. Houghton Mifflin Company.

Holger Schwenk and Xian Li. 2018. A corpus for multilingual document classification in eight languages. In LREC, pages 3548–3551.

Holger Schwenk, Ke Tran, Orhan Firat, and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. In ACL workshop, Repl4NLP.

Laura Smith, Salvatore Giorgi, Rishi Solanki, Johannes C. Eichstaedt, H. Andrew Schwartz, Muhammad Abdul-Mageed, Anneke Buffone, and Lyle H. Ungar. 2016. Does 'well-being' translate on twitter? In EMNLP, pages 2042–2047.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In ICLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In NIPS, pages 3104–3112.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In LREC, Istanbul, Turkey. European Language Resources Association (ELRA).

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In LREC.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In IJCAI.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. ICLR.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. In NAACL.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In CoNLL, pages 119–129.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. NAACL.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag-multilingual pos tagging via coarse mapping between embeddings. Association for Computational Linguistics.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Cross-lingual sentiment classification with bilingual document representation learning. In ACL.

Michal Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1. 0. In LREC.

Joint Multilingual Supervision for Cross-lingual Entity Linking

Shyam Upadhyay
University of Pennsylvania
Philadelphia, PA
shyamupa@seas.upenn.edu

Nitish Gupta
University of Pennsylvania
Philadelphia, PA
nitishg@seas.upenn.edu

Dan Roth
University of Pennsylvania
Philadelphia, PA
danroth@seas.upenn.edu

Abstract

Cross-lingual Entity Linking (XEL) aims to ground entity mentions written in *any* language to an English Knowledge Base (KB), such as Wikipedia. XEL for most languages is challenging, owing to limited availability of resources as supervision. We address this challenge by developing the first XEL approach that combines supervision from multiple languages *jointly*. This enables our approach to: **(a)** augment the limited supervision in the target language with additional supervision from a high-resource language (like English), and **(b)** train a *single* entity linking model for multiple languages, improving upon individually trained models for each language. Extensive evaluation on three benchmark datasets across 8 languages shows that our approach significantly improves over the current state-of-the-art. We also provide analyses in two limited resource settings: **(a)** *zero-shot setting*, when no supervision in the target language is available, and in **(b)** *low-resource setting*, when some supervision in the target language is available. Our analysis provides insights into the limitations of zero-shot XEL approaches in realistic scenarios, and shows the value of joint supervision in low-resource settings.[1]

1 Introduction

Entity Linking (EL) systems ground entity mentions in text to entries in Knowledge Bases (KB), such as Wikipedia (Mihalcea and Csomai, 2007). Recently, the task of Cross-lingual Entity Linking (XEL) has gained attention (McNamee et al., 2011; Ji et al., 2015; Tsai and Roth, 2016) with the goal of grounding entity mentions written in *any* language to the English Wikipedia. For instance, Figure 1 shows a Tamil (a language with >70 million speakers) and an English mention (shown [**enclosed**]) and their mention contexts. XEL involves grounding the Tamil mention (which translates to 'Liverpool') to the football club Liverpool_F.C., and not the city or the university. XEL enables knowledge acquisition directly from documents in any language, without resorting to machine translation.

சுரேஸ் **[லிவர்பூல்]** மற்றும் உருகுவே விளையாடுகிறார்.

Everton won against **[Liverpool]** in an FA Cup match.

Figure 1: Tamil and English mention contexts containing [**mentions**] of the entity Liverpool_F.C. from the respective Wikipedias. Tamil Wikipedia only has 9 mentions referring to Liverpool_F.C., whereas English Wikipedia has 5303 such mentions. Clearly, there is a need to augment the limited contextual evidence in low-resource languages with evidence from high-resource languages like English. Tamil sentence translates to "Suarez plays for [**Liverpool**] and Uruguay."

Training an EL model requires grounded mentions, i.e. mentions of entities that are grounded to a Knowledge Base (KB), as supervision (Figure 1). While millions of such mentions are available in English, by virtue of hyperlinks in the English Wikipedia, this is not the case for most languages. This makes learning XEL models challenging, especially for languages with limited resources (e.g., the Tamil Wikipedia is only 1% of the English Wikipedia in size). To overcome this challenge, it is desirable to augment the limited contextual evidence available in the target language with evidence from high-resource languages like English.

We propose XELMS (XEL with Multilingual Supervision) (§2), the first approach that fulfills the above desiderata by using multilingual supervision to train an XEL model. XELMS represents the mention contexts of the same entity from different languages in the same semantic space using a single context encoder (§2.1). Language-agnostic entity representations are jointly learned with the relevant mention context representations, so that an entity and its context share similar representations.

---

[1] Code at www.github.com/shyamupa/xelms

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2486–2495
Brussels, Belgium, October 31 - November 4, 2018.

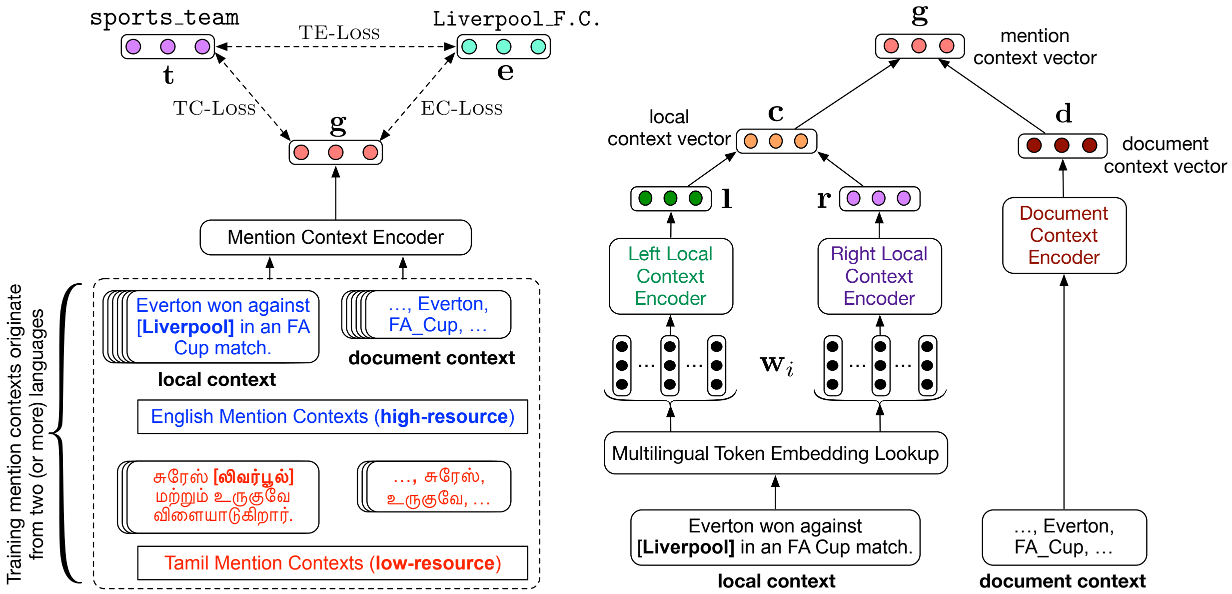

(a) Overview of XELMS. Mentions are shown [**enclosed**].

(b) Mention Context Encoder.

Figure 2: (**a**) Grounded mentions from two or more languages (English and Tamil shown) can be used to supervise XELMS. The context $\mathbf{g}$, entity $\mathbf{e}$ and type $\mathbf{t}$ vectors interact through Entity-Context loss (EC-LOSS), Type-Context loss (TC-LOSS) and Type-Entity loss (TE-LOSS). The Tamil sentence is the same as in Figure 1, and other mentions in it translate to [**Suarez**] and [**Uruguay**]. (**b**) The Mention Context Encoder (§2.1) encodes the local context (neighboring words) and the document context (surfaces of other mentions in the document) of the mention into $\mathbf{g}$. Internal view of local context encoder is in Figure 3.

Additionally, by encoding freely available structured knowledge, like fine-grained entity types, the entity and context representations can be further improved (§2.2).

The ability to use multilingual supervision enables XELMS to learn XEL models for target languages with limited resources by exploiting freely available supervision from high resource languages (like English). We show that XELMS outperforms existing state-of-the-art approaches that only use target language supervision, across 3 benchmark datasets in 8 languages (§5.1). Moreover, while previous XEL models (McNamee et al., 2011; Tsai and Roth, 2016) train separate models for different languages, XELMS can train a *single* model for performing XEL in multiple languages (§5.2).

One of the goals of XEL is to enable understanding of languages with limited resources. We provide experimental analyses in two such settings. In the *zero-shot setting* (§6.1), where *no* supervision is available in the target language, we show that the good performance of zero-shot XEL approaches (Sil et al., 2018) can be attributed to the use of prior probabilities. These probabilities are computed from large amount of grounded mentions, which are not available in realistic zero-shot settings. In the *low-resource setting* (§6.2), where some supervision is available in the target language, we show that even when only a fraction of the available supervision in the target language is provided, XELMS can achieve competitive performance by exploiting supervision from English.

The contributions of our work are,

- A new XEL approach, XELMS, that learns a XEL model for a language with limited resources by exploiting additional supervision from a high-resource language like English.
- XELMS can also train a *single* XEL model for multiple languages jointly, which we show improves on separately trained models.
- Analysis of XEL approaches in the zero-shot and low-resource settings. Our analysis reveals that in realistic scenarios, zero-shot XEL is not as effective as previously shown. We also show that in low-resource settings jointly training with English leads to better utilization of target language supervision.

2 Cross-lingual EL with XELMS

Given a mention m in a document $\mathcal{D}$ written in any language, XEL involves linking m to its gold entity e^* in a KB, $\mathcal{K} = \{e_1, \cdots, e_n\}$.

An overview of XELMS is shown in Figure 2a. XELMS computes the probability, $\mathrm{P}_{\text{context}}(e \mid m)$, of a mention m referring to entity $e \in \mathcal{K}$ using a mention context vector $\mathbf{g} \in \mathbb{R}^h$ representing

m's context, and an entity vector $\mathbf{e} \in \mathbb{R}^h$, representing the entity $e \in \mathcal{K}$ (one vector per entity). XELMS can also incorporate structured knowledge like fine-grained entity types (§2.2) using a multi-task learning approach (Caruana, 1998), by learning a type vector $\mathbf{t} \in \mathbb{R}^h$ for each possible type t (e.g., sports_team) associated with the entity e. The entity vector $\mathbf{e}$, context vector $\mathbf{g}$ and the type vector $\mathbf{t}$ are jointly trained, and interact through appropriately defined pairwise loss terms – an Entity-Context loss (EC-LOSS), Type-Entity loss (TE-LOSS) and a Type-Context loss (TC-LOSS).

The mention context vector $\mathbf{g}$ is generated by a mention context encoder (§2.1), shown in Figure 2b. The *mention context* of m in a document $\mathcal{D}$ consists of: **(a)** neighboring words around the mention, which we refer to as its *local context* and, **(b)** surfaces of other mentions appearing in $\mathcal{D}$, which we refer as its *document context*.

XELMS is trained using grounded mentions in multiple languages (English and Tamil in Figure 2a), which can be derived from Wikipedia (§4.1).

2.1 Mention Context Representation

To learn from mention contexts in multiple languages, we generate mention context representations using a language-agnostic mention context encoder. An overview of the mention context encoder is shown in Figure 2b. Below we describe the components of the mention context encoder, namely multilingual word embeddings and local and document context encoders.

Multilingual Word Embeddings (Ammar et al., 2016b; Smith et al., 2017; Duong et al., 2017) jointly encode words in multiple ($\geq$2) languages in the same vector space such that semantically similar words in the same language, and translationally equivalent words in different languages are close (per cosine similarity). Multilingual embeddings generalize bilingual embeddings, which do the same for two languages *only*.

We use FASTTEXT (Bojanowski et al., 2017; Smith et al., 2017), which aligns monolingual embeddings of multiple languages in the same space using a small dictionary (~2500 pairs) from each language to English. Both monolingual embeddings and the dictionary can be easily obtained for languages with limited resources. We denote the multilingual word embeddings for a set of tokens $\{w_1, w_2, \cdots, w_n\}$ by $\mathbf{w}_{1:n} = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}$, where each $\mathbf{w}_i \in \mathbb{R}^d$.

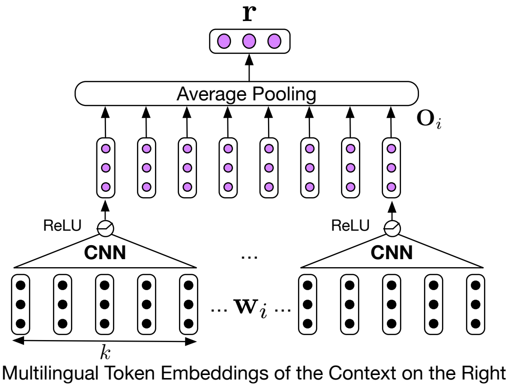

Figure 3: Local Context Encoder, for the right context. Figure 2b shows how it fits inside Mention Context Encoder.

Local Context Representation The local context of a mention m, spanning tokens i to j, consists of left context (tokens $i - W$ to j) and right context (tokens i to $j + W$). For example, for the mention [**Liverpool**] in Figure 2b, the left and right contexts are "Everton won against Liverpool" and "Liverpool in a FA Cup match" respectively. The local context encoder (Figure 3) encodes the left and the right contexts into vectors $\mathbf{l} \in \mathbb{R}^h$ and $\mathbf{r} \in \mathbb{R}^h$ using a convolutional neural network (CNN). These two vectors are then combined to generate the local context vector $\mathbf{c} \in \mathbb{R}^h$ (Figure 2b).

The CNN convolves continuous spans of k tokens using a filter matrix $\mathbf{F} \in \mathbb{R}^{kd \times h}$ to project the concatenation ($\oplus$ operator) of the token embeddings in the span. The resulting vector is passed through a ReLU unit to generate convolutional output $\mathbf{O}_i$. The outputs $\{\mathbf{O}_i\}$ are pooled by averaging,

$$\mathbf{O}_i = \text{RELU}(\mathbf{F}^T(\mathbf{w}_i \oplus \cdots \oplus \mathbf{w}_{i+k-1})) \quad (1)$$

$$\text{ENC}(\mathbf{w}_{1:n}) = \text{AVG}(\mathbf{O}_1, \cdots, \mathbf{O}_{n-k+1}) \quad (2)$$

Left and right context vectors $\mathbf{l}$ and $\mathbf{r}$ are computed using respective ENC(.) layers,

$$\mathbf{l} = \text{ENC}_{\text{left}}(\mathbf{w}_{i-W} \cdots \mathbf{w}_j) \quad (3)$$

$$\mathbf{r} = \text{ENC}_{\text{right}}(\mathbf{w}_i \cdots \mathbf{w}_{j+W}) \quad (4)$$

These vectors together generate the local context vector $\mathbf{c} = \mathcal{F}_{2h,h}(\mathbf{l} \oplus \mathbf{r})$. Here $\mathcal{F}_{d_i,d_o} : \mathbf{v}_i \rightarrow \mathbf{v}_o$ denotes a feed-forward layer that takes $\mathbf{v}_i \in \mathbb{R}^{d_i}$ as input, and outputs $\mathbf{v}_o \in \mathbb{R}^{d_o}$.

Document Context Representation Presence of certain mentions in a document can help disambiguate other mentions. For example, "Suarez", "Everton" in a document can help disambiguate "Liverpool". To incorporate this, we define the

document context d_m of a mention m appearing in document $\mathcal{D}$ to be the bag of all other mentions in $\mathcal{D}$. We encode d_m into a dense document context vector $\mathbf{d} \in \mathbb{R}^h$ by a feed-forward layer $\mathbf{d} = \mathcal{F}_{|V|,h}(d_m)$. Here V is the set containing all mention surfaces seen during training. When training jointly over multiple languages, V consists of mention surfaces seen in all languages (e.g. all English and Tamil mention surfaces) during training. This enables parameter sharing by embedding mention surfaces in different languages in the same low-dimensional space.
The local and document context vectors $\mathbf{c}$ and $\mathbf{d}$ are combined to get the mention context vector $\mathbf{g} = \mathcal{F}_{2h,h}(\mathbf{c} \oplus \mathbf{d})$.

Context Conditional Probability We compute the probability of a mention m linking to entity e using its context vector $\mathbf{g}$ and the entity vector $\mathbf{e}$,

$$\mathrm{P}_{\text{context}}(e \mid m) = \frac{\exp(\mathbf{g}^T\mathbf{e})}{\sum\limits_{e' \in C(m)} \exp(\mathbf{g}^T\mathbf{e}')} \quad (5)$$

where $C(m)$ denotes all candidate entities of the mention m (§3.1 explains how $C(m)$ is generated). We minimize the negative log-likelihood of $\mathrm{P}_{\text{context}}(e \mid m)$ with respect to the gold entity e^* against the candidate entities $C(m)$, and call it the Entity-Context loss (EC-LOSS),

$$\text{EC-LOSS} = -\log \frac{\mathrm{P}_{\text{context}}(e^* \mid m)}{\sum\limits_{e' \in C(m)} \mathrm{P}_{\text{context}}(e' \mid m)} \quad (6)$$

2.2 Including Type Information

Incorporating the fine-grained types of a mention m can help rank entities of the appropriate type higher than others (Ling et al., 2015; Gupta et al., 2017; Raiman and Raiman, 2018). For instance, knowing the correct type of mention **[Liverpool]** as sports_team and constraining linking to entities with the relevant type, encourages disambiguation to the correct entity.

To make the mention context representation $\mathbf{g}$ type-aware, we predict the set of fine-grained types of m, $\mathbf{T}(m) = \{t_1, ..., t_{|\mathbf{T}(m)|}\}$ using $\mathbf{g}$. Each t_i belongs to a pre-defined type vocabulary Γ.[2] The probability of a type t belonging to $\mathbf{T}(m)$ given the mention context is defined as $\mathrm{P}(t \mid m) = \sigma(\mathbf{t}^T\mathbf{g})$, where σ is the sigmoid function and $\mathbf{t}$ is the learnable embedding for type t.

We define a Type-Context loss (TC-LOSS) as,

$$\text{TC-LOSS} = \mathrm{BCE}(\mathbf{T}(m), \mathrm{P}(t \mid m)) \quad (7)$$

where BCE is the Binary Cross-Entropy Loss,

$$-\sum_{t \in \mathbf{T}(m)} \log \mathrm{P}(t \mid m) - \sum_{t \notin \mathbf{T}(m)} \log(1 - \mathrm{P}(t \mid m))$$

We also incorporate the entity-type information in the entity representations, and define a similar Type-Entity loss (TE-LOSS).

To identify the gold types $\mathbf{T}(m)$ of a mention m, we make the distant supervision assumption (same as Ling et al. (2015)) and assign the types of the gold entity e^* to be the types of the mention. Gold fine-grained types of the entities can be acquired from resources like Freebase (Bollacker et al., 2008) or YAGO (Hoffart et al., 2013).

3 Training and Inference

We explain how XELMS generates candidate entities, performs inference, and combines the different training losses.

3.1 Candidate Generation

Candidate generation identifies a small number of plausible entities for a mention m to avoid brute force comparison with all KB entities. Given m, candidate generation outputs a list of candidate entities $C(m) = \{e_1, e_2, \cdots, e_K\}$ of size at most K (we use K=20), each associated with a prior probability $\mathrm{P}_{\text{prior}}(e_i \mid m)$ indicating the probability of m referring to e_i, given only m's surface. $\mathrm{P}_{\text{prior}}$ is estimated from counts over the training mentions.

We adopt Tsai and Roth (2016)'s candidate generation strategy with some minor modifications (Appendix A). Using other approaches like CrossWikis (Spitkovsky and Chang, 2012), lead to consistently worse recall. We note that transliteration based candidate generation (McNamee et al., 2011; Pan et al., 2017; Tsai and Roth, 2018; Upadhyay et al., 2018) can further improve recall.

3.2 Inference

We combine the context conditional entity probability $\mathrm{P}_{\text{context}}(e \mid m)$ (eq. 5) and prior probability $\mathrm{P}_{\text{prior}}(e \mid m)$ by taking their union:

$$\begin{aligned}\mathrm{P}_{\text{model}}(e \mid m) &= \mathrm{P}_{\text{prior}}(e \mid m) + \mathrm{P}_{\text{context}}(e \mid m) \\ &\quad - \mathrm{P}_{\text{prior}}(e \mid m) \times \mathrm{P}_{\text{context}}(e \mid m)\end{aligned}$$

Inference for the mention m picks the entity,

$$\hat{e} = \underset{e \in C(m)}{\arg\max}\, \mathrm{P}_{\text{model}}(e \mid m) \quad (8)$$

---

[2] We use the type vocabulary Γ from Ling and Weld (2012), which contains 112 fine-grained types ($|\Gamma| = 112$)

3.3 Training Objective

When only training the mention context encoder and entity vectors, we minimize the EC-Loss averaged over all training mentions. When using the two type-aware losses, we minimize a weighted sum of EC-Loss, TE-Loss, and TC-Loss, using the weighing scheme of Kendall et al. (2018),

$$\frac{\text{EC-Loss}}{2\lambda_{\text{EC}}^2} + \frac{\text{TE-Loss}}{2\lambda_{\text{TE}}^2} + \frac{\text{TC-Loss}}{2\lambda_{\text{TC}}^2} \\ + \log \lambda_{\text{EC}}^2 + \log \lambda_{\text{TE}}^2 + \log \lambda_{\text{TC}}^2 \quad (9)$$

Here λ_i are learnable scalar weighing parameters, and the respective $\frac{1}{2\lambda_i^2}$ and $\log \lambda_i^2$ term ensure that λ_i^2 does not grow unboundedly. This way, the model learns the relative weight for each loss term.

During training, mentions from different languages are mixed using *inverse-ratio mini-batch mixing* strategy. That is, if two languages have training data sizes proportional to $\alpha : \beta$, at any time during training, mini-batches seen from them are in the ratio $\frac{1}{\alpha} : \frac{1}{\beta}$. This strategy prevents languages with more training data from overwhelming languages with less training data. Though simple, we found this strategy yielded good results.

4 Experimental Setup

We briefly describe the training and evaluation datasets, and the previous XEL approaches from the literature used in our comparison.

4.1 Training Mentions

Following previous work, we use hyperlinks from Wikipedia (dumps dated 05/20/2017) as our source of grounded mentions for supervision. Wikipedias in different languages have different pages for the same entity, which are resolved by using interlanguage links (e.g., page 利物浦 in Chinese Wikipedia resolves to Liverpool in English). Training mentions statistics are shown in Table 1.

Lang.	# Train Mentions	Size Relative to # English Mentions
German (de)	22.6M	43.7%
Spanish (es)	13.8M	26.7%
French (fr)	16.2M	31.3%
Italian (it)	11.5M	22.2%
Chinese (zh)	5.9M	11.4%
Arabic (ar)	3.1M	6.0%
Turkish (tr)	1.8M	3.5%
Tamil (ta)	473k	0.9%

Table 1: Number of train mentions (from Wikipedia) in each language, with % size relative to English (51.7M mentions). Train mentions from Wikipedias like Arabic, Turkish and Tamil are <10% the size of those from the English Wikipedia.

We evaluate on 8 languages – German (de), Spanish (es), Italian (it), French (fr), Chinese (zh), Arabic (ar), Turkish (tr) and Tamil (ta), each of which has varying amount of grounded mentions from the respective Wikipedia (Table 1). We note that our method is applicable to any of the 293 Wikipedia languages as a target language.

4.2 Evaluation Datasets

We evaluate XELMS on the following benchmark datasets, spanning 8 different languages, thus providing an extensive evaluation.

McN-Test dataset from (McNamee et al., 2011). The test set was collected by using parallel document collections, and then crowd-sourcing the ground truths. All the test mentions in this dataset consists of person-names only.

TH-Test A subset of the dataset used in (Tsai and Roth, 2016), derived from Wikipedia.[3] The mentions in the dataset fall in two categories – *easy* and *hard*, where hard mentions are those for which the most likely candidate according to the prior probability (i.e., $\arg\max \mathrm{P}_{\text{prior}}(e \mid m)$) is *not* the correct title. Indeed, most Wikipedia mentions can be correctly linked by selecting the most likely candidate (Ratinov et al., 2011). We use all the hard mentions from Tsai and Roth (2016)'s test splits for each language, and collectively call this subset TH-Test.

TAC15-Test TAC 2015 (Ji et al., 2015) dataset for Chinese and Spanish. It contains documents from discussion forum articles and news.

We evaluate all models using linking accuracy on gold mentions, and assume gold mentions are provided at test time. Table 2 summarizes the different domains of the evaluation datasets.

Tuning We avoid any dataset-specific tuning, instead tuning on a development set and applying the same parameters across all datasets. All tunable parameters were tuned on a development set containing the hard mentions from the train split released by Tsai and Roth (2016). We refer the reader to Appendix B for details on tuning.

---

[3] Pan et al. (2017) also created a dataset using Wikipedia, but did not categorize mentions like Tsai and Roth (2016). Preliminary experiments on their dataset showed XELMS consistently beat Pan et al. (2017)'s model. We chose TH-Test for more controlled experiments.

Dataset	Lang.	Source
TH-Test	de, es, fr, it, zh, ar, tr, ta	Wikipedia
McN-Test	de, es, fr, it, zh, ar, tr	News, Parliament Proceedings
TAC15-Test	es, zh	News, Discussion Forums

Table 2: Evaluation datasets used in our experiments.

4.3 Comparative Approaches

We compare against the following state-of-the-art (SoTA) approaches, described with the language from which they use mention contexts in **(.)**,

Tsai and Roth (2016) (Target Only) trains a separate XEL model for each language using mention contexts from the target language Wikipedia only. Current SoTA on TH-Test.

Pan et al. (2017) (English Only) uses entity coherence statistics from English Wikipedia and the document context of a mention for XEL. Current SoTA on McN-Test, except for Italian and Turkish, for which it's McNamee et al. (2011).

Sil et al. (2018) (English Only) uses multilingual embeddings to transfer a pre-trained English entity linking model to perform XEL for Spanish and Chinese. Prior probabilities P_{prior} are used as a feature. Current SoTA on TAC15-Test.

5 Experiments

We show that: **(a)** XELMS can train a better entity linking model for a target language on various benchmark datasets by exploiting additional data from a high resource language like English (§5.1). **(b)** XELMS can train a *single* XEL model for multiple related languages and improve upon separately trained models (§5.2). **(c)** Adding additional type information as multi-task loss to XELMS further improves performance (§5.3).

In all tables, we report the linking accuracy of XELMS, averaged over 5 different runs, and mark with * the statistical significance ($p < 0.01$) of the best result (shown **bold**) against the state-of-the-art (SoTA) using Student's one-sample t-test.

5.1 Monolingual and Joint Models

In Table 3 and 4 we compare XELMS(mono), which uses monolingual supervision in the target language only, and XELMS(joint), which uses supervision from English in addition to the monolingual supervision, with the state-of-the-art approaches.

Dataset →	TH-Test			McN-Test		
		XELMS			XELMS	
Lang ↓	SoTA	mono	joint	SoTA	mono	joint
de	53.3	53.7	**55.6***	89.7	90.9	**91.5**
es	54.5	54.9	**56.6***	**91.5**	91.2	91.4
fr	47.5	48.5	**49.9***	92.1	92.6	**92.7**
it	48.3	48.4	**51.9***	85.9	87.0	**87.8***
zh	57.6	58.1	**61.3***	**91.2***	87.4	88.2
ar	62.1	62.6	**63.8***	80.2	80.3	**83.1***
tr	60.2	61.0	**61.7***	**95.3***	91.0	91.9
ta	54.1	54.7	**59.7***	n/a	n/a	n/a
avg.	54.7	55.2	**57.6**	89.4	88.6	**89.5**

Table 3: XELMS(joint) improves upon XELMS(mono) and the current State-of-The-Art (SoTA) on TH-Test and McN-Test, showing the benefit of using additional supervision from English. The best score is shown **bold** and * marks statistical significance of best against SoTA. Refer §4.3 for details on SoTA.

	Model ↓ Lang. →	es	zh
	(Tsai and Roth, 2016)	82.4	85.1
	(Sil et al., 2018) (SoTA)	83.9	85.9
XELMS	mono	83.3	84.4
	mono$^{\text{+type}}$	83.5	84.8
	joint	84.1	85.5
	joint$^{\text{+type}}$	**84.4***	**86.0**
	multi	83.9	n/a
	multi$^{\text{+type}}$	**84.4***	n/a

Table 4: Linking accuracy on TAC15-Test. Numbers for Sil et al. (2018) from personal communication.

We see that XELMS(mono) achieves similar or slightly better scores than respective SoTA on all datasets. The SoTA for McN-Test in Turkish and Chinese enhances the model by using transliteration for candidate generation, explaining their superior performance. XELMS(joint) performs substantially better than XELMS(mono) on all datasets (Table 3 and 4), proving that using additional supervision from a high resource language like English leads to better linking performance. In particular, XELMS(joint) outperforms the SoTA on all languages in TH-Test, on Spanish in TAC15-Test, and on 4 of the 7 languages in McN-Test.

5.2 Multilingual Training

XELMS is the first approach that can train a *single* XEL model for multiple languages. To demonstrate this capability, we train a model, henceforth referred as XELMS(multi), *jointly* on 5 related languages – Spanish, German, French, Italian and En-

Dataset →	TH-TEST			MCN-TEST		
		XELMS			XELMS	
Lang ↓	SoTA	joint	multi	SoTA	joint	multi
de	53.3	**55.6***	55.2	89.7	**91.5**	91.4
es	54.5	56.6	**56.8***	**91.5**	91.4	91.4
fr	47.5	49.9	**51.0***	92.1	**92.7**	92.6
it	48.3	51.9	**52.3***	85.9	87.8	**87.9***
avg.	50.9	53.5	**53.8**	89.8	**90.8**	**90.8**

Table 5: Linking accuracy of a *single* XELMS(multi) model for four languages – German, Spanish, French and Italian. Individually trained XELMS(joint) scores are also shown. The best score is shown **bold** and * marks statistical significance of **best** against SoTA. Refer §4.3 for details on SoTA.

glish. We compare XELMS(multi) to the respective XELMS(joint) model for each language.

Table 4 and 5, show that XELMS(multi) is better (or at par) than XELMS(joint) on all datasets. This shows that XELMS(multi) can making more efficient use of available supervision in related languages than previous approaches which trained separate models per language.

5.3 Adding Fine-grained Type Information

To study the effect of adding fine-grained type information, in Table 4 we compare XELMS(mono) and XELMS(joint) to XELMS(mono^{+type}) and XELMS(joint^{+type}) respectively, which are versions of XELMS(mono) and XELMS(joint) trained using the two type-aware losses.

XELMS(mono^{+type}) and XELMS(joint^{+type}) both improve compared to XELMS(mono) and XELMS(joint) on MCN-TEST and TH-TEST (Table 6 vs Table 3), showing the benefit of using structured knowledge in the form of fine-grained types. Similar trends are also seen on TAC15-TEST (Table 4), where XELMS(joint^{+type}) improves on the SoTA for Spanish and Chinese.

6 Experiments with Limited Resources

The key motivation of XELMS is to exploit supervision from high-resource languages like English to aid XEL for languages with limited resources. In this section, we examine two such scenarios,
(a) *Zero-shot setting* i.e., no supervision available in the target language. Our analysis reveals the limitations of zero-shot XEL approaches and finds that the prior probabilities play an important role in achieving good performance (§6.1), which are unavailable in realistic zero-shot scenarios.
(b) *Low-resource setting* i.e., some supervision available in the target language. We show that by combining supervision from a high-resource language, like English, XELMS can achieve competitive performance with a fraction of available supervision in the target language (§6.2).

Dataset →	TH-TEST			MCN-TEST		
		XELMS			XELMS	
Lang ↓	SoTA	mono^{+type}	joint^{+type}	SoTA	mono^{+type}	joint^{+type}
de	53.3	54.0	**55.9***	89.7	91.2	**91.5**
es	54.5	55.1	**57.2***	**91.5**	91.0	91.2
fr	47.5	49.0	**50.6***	92.1	92.6	**92.7**
it	48.3	49.2	**52.2***	85.9	87.4	**87.9***
zh	57.6	58.9	**61.5***	**91.2***	87.6	88.4
ar	62.1	63.0	**64.0***	80.2	81.1	**84.0***
tr	60.2	61.5	**62.0***	**95.3***	91.2	92.1
ta	54.1	56.0	**59.9***	n/a	n/a	n/a
avg.	54.7	55.8	**57.9**	89.4	88.9	**89.7**

Table 6: Adding fine-grained type information further improves linking accuracy (compare to Table 3). The best score is shown **bold** and * marks statistical significance of best against SoTA. Refer §4.3 for details on SoTA.

6.1 Zero-shot Setting

We first explain how XELMS can perform zero-shot XEL, the implications of our zero-shot setting, and how it is more realistic than previous work.

Zero-shot XEL with XELMS XELMS performs zero-shot XEL by training a model using English supervision and multilingual embeddings for English, and directly applying it to the test data in another language using the respective multilingual word embedding instead of English embeddings.

No Prior Probabilities Prior probabilities (or prior), i.e., P_{prior} have been shown to be a reliable indicator of the correct disambiguation in entity linking (Ratinov et al., 2011; Tsai and Roth, 2016). These probabilities are estimated from counts over the training mentions in the target language. In the absence of training data for the target language, as in the zero-shot setting, these prior probabilities are not available to an XEL model.

Comparison to Previous Work The only other model capable of zero-shot XEL is that of Sil et al. (2018). However, Sil et al. (2018) use prior probabilities and coreference chains for the target language in their zero-shot experiments, both of which will not be available in a realistic zero-shot scenario. Compared to Sil et al. (2018), we evaluate the performance of zero-shot XEL in more realistic setting, and show it is adversely affected by absence of prior probabilities.

Dataset → **Approach** ↓	TAC15-Test (es)	(zh)	TH-Test (avg)	McN-Test (avg)
XELMS (Z-S w/ prior)	80.3	83.9	43.5	88.1
XELMS (Z-S w/o prior)	53.5	55.9	41.1	86.0
SoTA	83.9	85.9	54.7	89.4

Table 7: Linking accuracy of the zero-shot (Z-S) approach on different datasets. Zero-shot (w/ prior) is close to SoTA for datasets like TAC15-Test, but performance drops in the more realistic setting of zero-shot (w/o prior) (§6.1) on all datasets, indicating most of the performance can be attributed to the presence of prior probabilities. The slight drop in McN-Test is due to trivial mentions, which only have a single candidate.

Is zero-shot XEL really effective? To evaluate the effectiveness of the zero-shot XEL approach, we perform zero-shot XEL using XELMS on all datasets. Table 7 shows zero-shot XEL results on all datasets, both with and without using the prior during inference. Note that zero-shot XEL (with prior) is close to SoTA (Sil et al. (2018)) on TAC15-Test, which also uses the prior for zero-shot XEL. However, for zero-shot XEL (without prior) performance drops by more than 20% for TAC15-Test, 2.4% for TH-Test and by 2.1% for McN-Test. This indicates that zero-shot XEL is not effective in a realistic zero-shot setting (i.e., when the prior is unavailable for inference).

We found that the prior is indeed a strong indicator of the correct disambiguation. For instance, simply selecting the the most likely candidate using the prior for TAC15-Test achieved 77.2% and 78.8% for Spanish and Chinese respectively. It is interesting to note that both zero-shot XEL (with or without prior) perform worse than the best possible model on TH-Test, because TH-Test was constructed to ensure prior probabilities are not strong indicators (Tsai and Roth, 2016). On McN-Test, we found that an average of 75.9% mentions have only one (the correct) candidate, making them trivial to link, regardless of the absence of priors.

The results show that most of the XEL performance in zero-shot settings can be attributed to availability of prior probabilities for the candidates. It is evident that zero-shot XEL in a realistic setting (i.e., when prior probabilities are not available) is still a challenging problem.

6.2 Low-resource Setting

We analyze the behavior of XELMS in a low-resource setting, i.e. when some supervision is available in the target language. The aim of this setting is to estimate how much supervision from the target language is needed to get reasonable performance when using it jointly with supervision from English. To discount the effect of prior probabilities, we report all results without the prior.

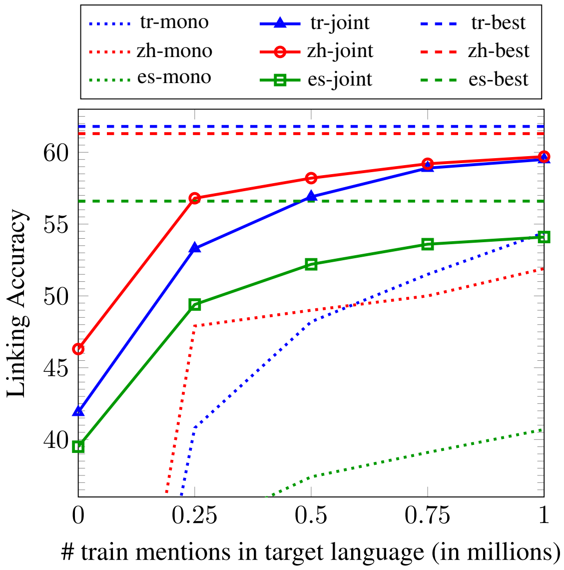

Figure 4: Linking accuracy vs. the number of train mentions in the target language L (= Turkish (tr), Chinese (zh) and Spanish (es)). We compare both XELMS(mono) and XELMS(joint) to the best results using all available supervision, denoted by L-best. To discount the effect of the prior, all results above are without it. For number of train mentions = 0, XELMS(joint) is equivalent to zero-shot without prior. Best viewed in color.

Figure 4 plots results on the TH-Test dataset when training a XELMS(joint) model by gradually increasing the number of mention contexts for target language L (= Spanish, Chinese and Turkish) that are available for supervision. Figure 4 also shows the best results achieved using all available target language supervision (denoted by L-best). For comparison with the mono-lingually supervised model, we also plot the performance of XELMS(mono), which only uses the target language supervision.

Figure 4 shows that after training on 0.75M mentions from Turkish and Chinese (and 1.0M mentions from Spanish), the XELMS(joint) model is within 2-3% of the respective L-best model which uses all training mentions in the target language, indicating that XELMS(joint) can reach competitive performance even with a fraction of the full target language supervision. For comparison, a XELMS(mono) model trained on the same number of training mentions is 5-10% behind the respective XELMS(joint) model, showing better utilization of target language supervision by XELMS(joint).

7 Related Work

Existing approaches have taken two main directions to obtain supervision for learning XEL models — **(a)** using mention contexts appearing in the target language (McNamee et al., 2011; Tsai and Roth, 2016), or **(b)** using mention contexts appearing only in English (Pan et al., 2017; Sil et al., 2018). We describe these directions and their limitations below, and explain how XELMS overcomes these limitations.

McNamee et al. (2011) use annotation projection via parallel corpora to generate mention contexts in the target language, while Tsai and Roth (2016) learns separate XEL models for each language and only use mention contexts in the target language. Both these approach have scalability issues for languages with limited resources. Another limitation of these approaches is that they train separate models for each language, which is inefficient when working with multiple languages. XELMS overcomes these limitations as it can use mention context from multiple languages simultaneously, and train a single model.

Other approaches only use mention contexts from English. While Pan et al. (2017) compute entity coherence statistics from English Wikipedia, Sil et al. (2018) perform zero-shot XEL for Chinese and Spanish by using multilingual embeddings to transfer a pre-trained English EL model. However, our work suggests that mention contexts in the target language should also be used, if available. Indeed, a recent study (Lewoniewski et al., 2017) found that for language sensitive topics, the quality of information can be better in the relevant language version of Wikipedia than the English version. Our analysis also shows that zero-shot XEL approaches like that of Sil et al. (2018) are not effective in realistic zero-shot scenarios where good prior probabilities are unlikely to be available. In such cases, we showed that combining supervision available in the target language with supervision from a high-resource language like English can yield significant performance improvements.

The architecture of XELMS is inspired by several monolingual entity linking systems (Francis-Landau et al., 2016; Nguyen et al., 2016; Gupta et al., 2017), approaches that use type information to aid entity linking (Ling et al., 2015; Gupta et al., 2017; Raiman and Raiman, 2018), and the recent success of multilingual embeddings for several tasks (Ammar et al., 2016a; Duong et al., 2017).

8 Conclusion

We introduced XELMS, an approach that can combine supervision from multiple languages to train an XEL model. We illustrate its benefits through extensive evaluation on different benchmarks. XELMS is also the first approach that can train a *single* model for multiple languages, making more efficient use of available supervision than previous approaches which trained separate models.

Our analysis sheds light on the poor performance of zero-shot XEL in realistic scenarios where the prior probabilities for candidates are unlikely to exist, in contrast to findings in previous work that focused on high-resource languages. We also show how in low-resource settings, XELMS makes it possible to achieve competitive performance even when only a fraction of the available supervision in the target language is provided.

Several future research directions remain open. For all XEL approaches, the task of candidate generation is currently limited by existence of a target language Wikipedia and remains a key challenge. A joint inference framework which enforces coherent predictions (Cheng and Roth, 2013; Globerson et al., 2016; Ganea and Hofmann, 2017) could also lead to further improvements for XEL. Similar techniques can be applied to other information extraction tasks like relation extraction to extend them to multilingual settings.

Acknowledgments

The authors would like to thank Snigdha Chaturvedi, Anne Cocos, Stephen Mayhew, Chen-Tse Tsai, Qiang Ning, Jordan Kodner, Dan Deutsch, John Hewitt and the anonymous EMNLP reviewers for their useful comments and suggestions.

This work was supported by Contract HR0011-15-2-0025 and Agreement HR0011-15-2-0023 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016a. Many Languages, One Parser. In *Transactions of the Association for Computational Linguistics*, volume 4.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016b. Massively Multilingual Word Embeddings. *arXiv preprint arXiv:1602.01925*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. In *Transactions of the Association for Computational Linguistics*, volume 5.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proc. of ACM SIGMOD*.

Rich Caruana. 1998. Multitask Learning. In *Learning to Learn*, pages 95–133. Springer.

Xiao Cheng and Dan Roth. 2013. Relational Inference for Wikification. In *Proc. of EMNLP*.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2017. Multilingual Training of Crosslingual Word Embeddings. In *Proc. of EACL*.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing Semantic Similarity for Entity Linking with Convolutional Neural Networks. In *Proc. of NAACL-HLT*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep Joint Entity Disambiguation with Local Neural Attention. In *Proc. of EMNLP*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective Entity Resolution with Multi-Focal Attention. In *Proc. of ACL*.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity Linking via Joint Encoding of Types, Descriptions, and Context. In *Proc. of EMNLP*.

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. In *Proc. of IJCAI*.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of TAC-KBP2015 Tri-lingual Entity Discovery and Linking. In *Text Analysis Conference*.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *Proc. of CVPR*.

Włodzimierz Lewoniewski, Krzysztof Wecel, and Witold Abramowicz. 2017. Relative Quality and Popularity Evaluation of Multilingual Wikipedia Articles. In *Journal of Informatics*.

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design Challenges for Entity Linking. In *Transactions of the Association for Computational Linguistics*, volume 3.

Xiao Ling and Daniel S Weld. 2012. Fine-grained Entity Recognition. In *Proc. of AAAI*.

Paul McNamee, James Mayfield, Dawn Lawrie, Douglas W Oard, and David S Doermann. 2011. Cross-Language Entity Linking. In *Proc. of IJCNLP*.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proc. of CIKM*.

Thien Huu Nguyen, Nicolas Fauceglia, Mariano Rodriguez Muro, Oktie Hassanzadeh, Alfio Massimiliano Gliozzo, and Mohammad Sadoghi. 2016. Joint Learning of Local and Global Features for Entity Linking via Neural Networks. In *Proc. of COLING*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual Name Tagging and Linking for 282 Languages. In *Proc. of ACL*.

Jonathan Raiman and Olivier Raiman. 2018. Deep-Type: Multilingual Entity Linking by Neural Type System Evolution. In *Proc. of AAAI*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proc. of ACL-HLT*.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural Cross-lingual Entity Linking. In *Proc. of AAAI*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline Bilingual Word Vectors, Orthogonal Transformations, and the Inverted Softmax. In *Proc. of ICLR*.

Valentin I. Spitkovsky and Angel X. Chang. 2012. A Cross-Lingual Dictionary for English Wikipedia Concepts. In *Proc. of LREC*.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual Wikification Using Multilingual Embeddings. In *Proc. of NAACL*.

Chen-Tse Tsai and Dan Roth. 2018. Learning Better Name Translation for Cross-Lingual Wikification. In *Proc. of AAAI*.

Shyam Upadhyay, Jordan Kodner, and Dan Roth. 2018. Bootstrapping Transliteration with Constrained Discovery for Low-Resource Languages. In *Proc. of EMNLP*.

Fine-grained Coordinated Cross-lingual Text Stream Alignment for Endless Language Knowledge Acquisition

Tao Ge[1,2], **Qing Dou**[3], **Heng Ji**[4], **Lei Cui**[2], **Baobao Chang**[1], **Zhifang Sui**[1],
Furu Wei[2] **and Ming Zhou**[2]
[1]MOE Key Laboratory of Computational Linguistics, Peking University, Beijing, 100871, China
[2]Microsoft Research Asia, Beijing, 100080, China
[3]Facebook, CA, 94025, USA
[4]Rensselaer Polytechnic Institute, NY, 12180, USA
`{tage,lecu,fuwei,mingzhou}@microsoft.com`
`douqing@gmail.com, jih@rpi.edu, {chbb,szf}@pku.edu.cn`

Abstract

This paper proposes to study fine-grained coordinated cross-lingual text stream alignment through a novel information network decipherment paradigm. We use Burst Information Networks as media to represent text streams and present a simple yet effective network decipherment algorithm with diverse clues to decipher the networks for accurate text stream alignment. Experiments on Chinese-English news streams show our approach not only outperforms previous approaches on bilingual lexicon extraction from coordinated text streams but also can harvest high-quality alignments from large amounts of streaming data for endless language knowledge mining, which makes it promising to be a new paradigm for automatic language knowledge acquisition.

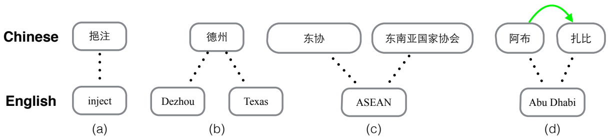

Figure 1: Knowledge derived from fine-grained cross-lingual text stream alignments: (a) word translations; (b) polysemy/multi-references; (c) synonym/co-reference; (d) entity phrases

1 Introduction

Coordinated text streams (Wang et al., 2007) refer to the text streams that are topically related and indexed by the same set of time points. Previous studies (Wang et al., 2007; Hu et al., 2012) on coordinated text stream focus on discovering and aligning common topic patterns across languages. Despite their contributions to applications like cross-lingual information retrieval and topic analysis, such a coarse-grained topic-level alignment framework inevitably overlooks many useful fine-grained alignment knowledge. For example, Figure 1 shows typical knowledge that can be derived from fine-grained Chinese-English text stream alignments. In addition to (a) bi-lingual word translations, we can also discover (b) polysemous and multi-referential words if one Chinese word is aligned to multiple English words, (c) synonymous and co-referential word pairs if two Chinese words are aligned to the same English word, and (d) entity phrases (e.g., 阿布扎比 in Figure 1) if adjacent Chinese words in text are aligned to the same English named entity.

In order to acquire language knowledge for Natural Language Processing (NLP) applications, we study fine-grained cross-lingual text stream alignment. Instead of directly turning massive, unstructured data streams into structured knowledge (D2K), we adopt a new Data-to-Network-to-Knowledge (D2N2K) paradigm, based on the following observations: (i) most information units are not independent, instead they are interconnected or interacting, forming massive networks; (ii) if information networks can be constructed across multiple languages, they may bring tremendous power to make knowledge mining algorithms more scalable and effective because we can employ the graph structures to acquire and propagate knowledge.

Based on the motivations, we employ a promising text stream representation – *Burst Information Networks (BINets)* (Ge et al., 2016a), which can be easily constructed without rich language resources, as media to display the most important information units and illustrate their connections in the text streams. With the BINet representation, we propose a simple yet effective network decipherment algorithm for aligning cross-lingual text streams, which can take advantage of the co-burst characteristic of cross-lingual text streams and easily incorporate prior knowledge and rich clues for fast and accurate network decipherment.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2496–2506
Brussels, Belgium, October 31 - November 4, 2018.

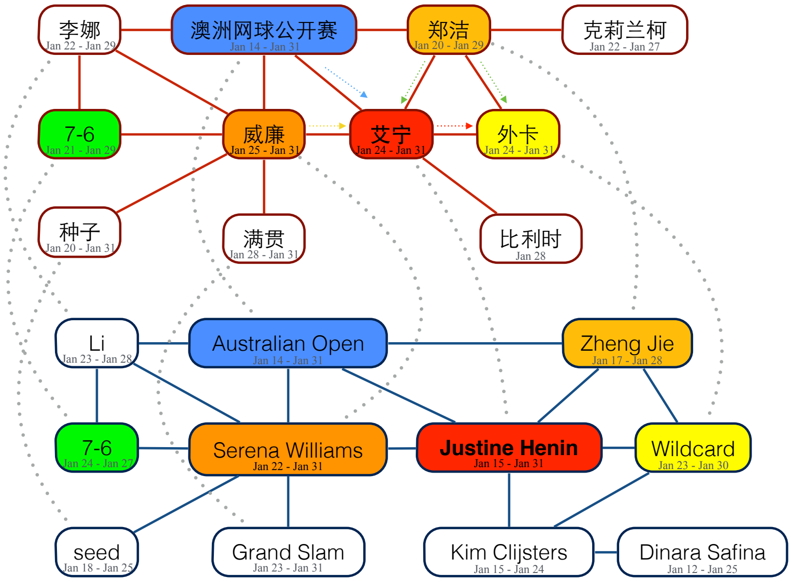

Figure 2: (A part of) Burst Information Networks built from Chinese and English news streams.

For example, in Figure 2, each node in a BINet is a bursty word with one of its burst periods, representing an important information unit in a text stream. To decipher the Chinese BINet, our approach first focuses on the nodes in the English BINet in Figure 2 as the candidates because they co-burst with the Chinese nodes. Then, we decipher some nodes based on prior knowledge (the green node), the pronunciation similarity clue (the orange nodes) or literal translation similarity clue (the blue node). These deciphered nodes will serve as neighbor clues to decipher their adjacent nodes (the red node) which will then be used for further decipherment (e.g., decipher the yellow node) through knowledge propagation across the network, as the dashed arrows in Figure 2 show.

Experiments on Chinese-English coordinated news streams show our approach can accurately align nodes across the cross-lingual BINets and derive various knowledge, and that with more streaming data provided, we can harvest more high-quality alignments and thus derive more knowledge. By aligning endless text streams, it is promising for never-ending language knowledge mining, which can not only complement language resources but also benefit some NLP applications.

The main contributions of this paper are:

- We propose a promising framework to mine knowledge from inexhaustible coordinated cross-lingual text streams through fine-grained alignment, exploring a paradigm for language knowledge acquisition.
- We propose a network decipherment approach for text stream alignment, which can work in both low and rich resource settings and outperform previous approaches.
- We release our data (annotations) and systems to guarantee the reproducibility and help future work improve on this task.

2 Burst Information Network

A Burst Information Network (BINet) is a graph-based text stream representation and has proven effective for multiple text stream mining tasks (Ge et al., 2016a,b,c). In contrast to many information networks (e.g., (Ji, 2009; Li et al., 2014)), BINets are specially for text streams. They focus on the burst information units which are usually related to important events or trending topics in text streams and illustrate their connections.

A BINet is originally defined as $G = \langle V, E, \boldsymbol{\omega} \rangle$ in (Ge et al., 2016a). Each node $v \in V$ is a burst element defined as a burst word[1] during one of its burst periods $\langle w, \mathcal{P} \rangle$ where w denotes a word and $\mathcal{P}$ denotes one consecutive burst period of w, as Figure 2 shows. Each edge $\epsilon \in E$ indicates the connection between two burst elements with the weight ω which is defined as the number of documents where these two burst elements co-occur in the text stream. In this paper, we extend the BINet definition to $G = \langle V, E, \boldsymbol{\omega}, \boldsymbol{\pi} \rangle$ by adding a binary

---

[1] Burst words and their corresponding burst periods can be detected based on Kleinberg burst detection algorithm (Kleinberg, 2003), as (Ge et al., 2016a) did.

indicator π to indicate if two nodes (i.e., burst elements) are frequently (more than 5 times) adjacent (as a bigram) in text, for mining knowledge such as entity phrases in Figure 1(d).

3 Decipherment

After constructing a BINet from a foreign language (we use Chinese as a foreign language in this paper), we can decipher it by consulting an Engish BINet constructed from its coordinated English text stream. We define $G_c = \langle V_c, E_c, \boldsymbol{\omega_c}, \boldsymbol{\pi_c} \rangle$ and $G_e = \langle V_e, E_e, \boldsymbol{\omega_e}, \boldsymbol{\pi_e} \rangle$ as the Chinese BINet and English BINet respectively. For people who do not know Chinese, G_c is a network of ciphers. We design a novel BINet decipherment procedure to decipher G_c by aligning as many nodes in G_c as possible to G_e. The decipherment process is defined to find $e \in V_e$ for a node $c \in V_c$ so that e is c's counterpart in the English text stream.[2]

3.1 Starting Point

To decipher the Chinese BINet, we need a few seeds based on prior knowledge as a starting point. Inspired by previous work on bi-lingual lexicon induction, decipherment and name translation mining, we utilize a few linguistic resources - a bilingual lexicon and language-universal representations such as time/calendar date, number, website URL, currency and emoticons to decipher a subset of Chinese nodes. For the example shown in Figure 2, we can decipher some nodes in the Chinese BINet such as “7-6” (to “7-6”) and “种子” (to “seed”).

3.2 Candidate Generation

For the nodes that cannot be deciphered by the prior knowledge, we first need to discover their possible candidates. For a node c in the Chinese BINet, its counterpart e can be any node in the English BINet or does not exist in the English BINet, resulting in an extremely large search space. Fortunately, burst information that refers to a hot topic usually co-bursts across languages. Based on this characteristic, for a node in the Chinese BINet, its counterpart is likely to be a node with the same burst period in the English BINet. For example, the node “威廉(Williams)” in the Chinese BINet in Figure 2 bursts between January 25 and January 31, 2010. We only need to look for its counterpart from the nodes in the English BINet whose burst period overlaps with this period. Formally, for a node $c \in V_c$ in the Chinese BINet, its candidate nodes in the English BINet can be derived as:

$$Cand(c) = \{e | \mathcal{P}(e) \cap \mathcal{P}(c) \neq \emptyset\}$$

where $e \in V_e$, and $\mathcal{P}(c)$ and $\mathcal{P}(e)$ are the burst periods of c and e respectively.

3.3 Candidate Verification

For the candidate list for c (i.e., $Cand(c)$), we need to verify each node $e \in Cand(c)$ and choose the most probable one as c's counterpart. Formally, we define $Score(c, e)$ as the credibility score of e being the correct counterpart of c and propose the following novel clues for verification.

Pronunciation

Inspired by previous work on name translation mining (e.g., (Schafer III, 2006; Sproat et al., 2006; Ji, 2009)), for a node $e \in Cand(c)$, if its pronunciation is similar to c, then e is likely to be the translation of c. For a Chinese node c and an English node e, we define S_p as its scaled pronunciation score to measure their pronunciation similarity whose range is $[0, 1]$:

$$S_p \in [0, 1] \propto \frac{1}{LD}$$

where LD is the normalized (by e's length) Levenshtein edit distance between c's pinyin[3] string and e's word string.

Translation

For a node $e \in Cand(c)$, it is possible that e's word exists or partially exists in the bi-lingual lexicon. We can exploit the translation clue to verify if e is c's counterpart. For example, “Australian Open” is a candidate of “澳洲网球公开赛(Australian Open)” as shown in Figure 2. Even though “澳洲网球公开赛(Australian Open)” is not in the bi-lingual lexicon, “Australian” and “open” are in the lexicon and their Chinese translations are “澳洲的(Australian)” and “公开(open)” respectively. If we literally translate “Australian Open” word by word, we will get “澳洲的公开” which has long common subsequences with the Chinese node “澳洲网球公开赛(Australian Open)”, inferring that “Australian Open” is likely to be the translation of “澳洲网球公开赛”.

---

[2] c and e are burst elements (i.e., nodes in the BINets). Sometimes, we also use c and e to denote the nodes' word if that does not lead to misunderstanding.

[3] Pinyin is the official romanization system for Chinese. We use pinyin instead of IPA because romanization is usually more easily available than IPA for a language.

Motivated by this observation, for a candidate $e \in Cand(c)$, we first extract its possible Chinese translations $C(e)$ from the bilingual lexicon. Note that if e is a multiword, we concatenate translations of its components. Then, for $\langle c, e \rangle$, we define S_t as its scaled translation similarity score whose range is $[0, 1]$:

$$S_t \in [0,1] \propto \max_{c' \in C(e)} LCS(c, c')$$

where $\max_{c' \in C(e)} LCS(c, c')$ is maximum length of the longest common subsequence between c and $c' \in C(e)$.

Neighbor

The graph topological structure of a BINet is also an important clue for decipherment. By analyzing a node's neighbors, we can learn useful topic-level knowledge to decipher the node. For the example in Figure 2, "艾宁(Henin)" in the Chinese BINet has neighbors such as "威廉(Williams)", "澳洲网球公开赛(Australian Open)" and "郑洁(Zheng Jie)" while "Justine Henin" in the English BINet is connected with "Serena Williams", "Australian Open" and "Zheng Jie". If we know "Serena Williams", "Australian Open" and "Zheng Jie" are the counterpart of '威廉", "澳洲网球公开赛" and "郑洁" respectively, we can infer "Justine Henin" is likely to be the counterpart of "艾宁", which can be further used as a clue to decipher its neighbors such as "外卡(wildcard)" through knowledge propagation.

We define $N(c)$ and $N(e)$ as the set of adjacent nodes of c in the Chinese BINet and the adjacent nodes of e in the English BINet respectively. The neighbor clue score S_n of $\langle c, e \rangle$ is defined as:

$$S_n = \sum_{c' \in N(c)} \hat{\omega}_{c,c'} \max_{e' \in N(e)} Score(c', e') \qquad (1)$$

where $Score(c', e')$ is the overall score of e' being the counterpart of c', as defined at the beginning of this section, $\hat{\omega}_{c,c'} = \frac{\omega_{c,c'}}{\sum_{c'' \in N(c')} \omega_{c',c''}}$ is the normalized weight of the edge between c and c'.

Correlation of burst

If the word of $e \in Cand(c)$ frequently co-bursts with the word of c, then e is likely to be the counterpart of c. For example, "Serena Williams" in the English stream usually co-bursts with "小威" in the Chinese stream, as shown in Figure 3, which is a useful clue to infer that "Serena Williams" is the counterpart of "小威".

We define S_b as the burst correlation score:

$$S_b = \frac{\boldsymbol{s}_{w(c)} \cdot \boldsymbol{s}_{w(e)}}{\|\boldsymbol{s}_{w(c)}\|_1 + \|\boldsymbol{s}_{w(e)}\|_1 - \boldsymbol{s}_{w(c)} \cdot \boldsymbol{s}_{w(e)}} \qquad (2)$$

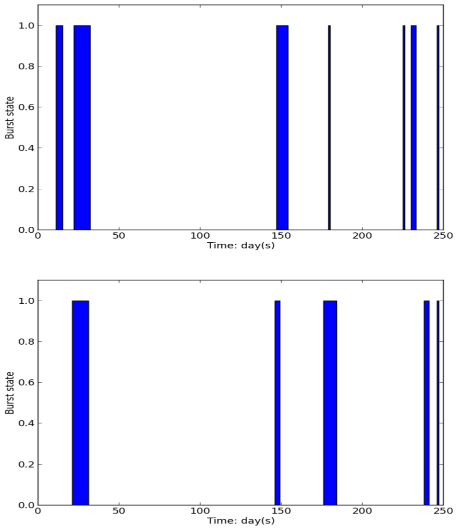

Figure 3: Burst states of "小威" (the upper) and "Serena Williams" (the lower) are correlated.

where $w(v)$ denotes the word of the node v and $\boldsymbol{s}_w$ denotes the burst sequence of the word w in which each entry is a binary variable indicating if w bursts at a moment throughout the time frame. Note that in the above equation, we regard $\boldsymbol{s}_{\boldsymbol{w}}$ as a vector. The numerator is the number of days when $w(c)$ and $w(e)$ co-burst and the denominator is the number of days when either $w(c)$ or $w(e)$ bursts.

3.4 Graph-based Decipherment

We define the overall (credibility) score as the linear combination of the clues introduced above:

$$Score(c, e) = \eta S_p + \lambda S_t + \gamma S_n + \delta S_b \qquad (3)$$

where S_p, S_t, S_n and S_b are the scores that measure the value/reliability of the pronunciation, translation, neighbor and burst correlation clues respectively, and η, λ, γ and δ are hyperparameters for adjusting their weights.

Based on Eq (3), we can now compute the score of any candidate pair $\langle c, e \rangle$. For pairs that are known to be correct alignments according to prior knowledge, their overall scores will be fixed to 1.0. For other possible candidate pairs, we simply initialize their scores as follows:

$$Score(c, e) = \frac{1.0}{|Cand(c)|} \qquad (4)$$

where $Cand(c)$ is the set of c's candidate nodes in the English BINet.

Given that $Score(c, e)$ is influenced by other pairs' scores, we design an iterative algorithm to compute and update the scores to decipher the entire Chinese BINet through propagation. This process is elaborated in Algorithm 1.

Algorithm 1 Graph-based Decipherment

```
1:  For the determined pair $\langle c, e\rangle$ based on the prior knowledge, $Score(c, e) \leftarrow 1.0$
2:  For other undermined pairs $\langle c, e\rangle$, initialize $Score(c, e)$ according to Eq (4);
3:  while $True$ (until $\Delta Conf(G_c, G_e) \leq 0.0001$) do
4:      for each undetermined pair $\langle c, e\rangle$ do
5:          Compute $new\_score$ according to Eq (3);
6:          $update(c, e) = \min(1.0, new\_score)$
7:      end for
8:      for each undetermined pair $\langle c, e\rangle$ do
9:          $Score(c, e) \leftarrow update(c, e)$
10:     end for
11: end while
```

$\Delta Conf(G_c, G_e)$ in the 3rd line of Algorithm 1 is the difference between the network decipherment confidence score at the current iteration and that at its previous iteration. $Conf(G_c, G_e)$ is defined as follows, reflecting how much confidence we have in our network decipherment result:

$$Conf(G_c, G_e) = \sum_{c \in V_c} \max_{e \in Cand(c)} Score(c, e) \qquad (5)$$

In practice, propagation of prior knowledge and clues makes the confidence score increase because it helps us know more about the network (as illustrated by Figure 2). When the confidence score stops increasing or increases marginally (≤ 0.0001) after several iterations, the algorithm terminates[4].

4 Experiments

We first evaluate our approach on aligning nodes in the cross-lingual BINets for fine-grained cross-lingual stream alignment in Section 4.1. Then, we show the value of derived alignments for endless language knowledge acquisition in Section 4.2.

4.1 Stream alignment

4.1.1 Data

We used the public 2010 Agence France Presse (AFP) news in Chinese (Graff and Chen, 2005) and English Gigaword (Graff et al., 2003) as our cross-lingual text streams. The Chinese stream has 17,327 while the English one contains 186,737 documents.

We removed stopwords, conducted lemmatization and name tagging for the English stream, and did word segmentation and name tagging for the Chinese stream using the Stanford CoreNLP toolkit (Manning et al., 2014).

We detected bursts and constructed the BINets[5] for the Chinese and English stream based on (Ge et al., 2016a). The constructed Chinese BINet has 7,360 nodes and 33,892 edges while the English one has 8,852 nodes and 85,125 edges. Our seed bi-lingual lexicon is released by (Zens and Ney, 2004), containing 81,990 Chinese word entries, each of which has an English translation. Among the 7,360 nodes in the Chinese BINet, 2,281 nodes need to be deciphered since their words are not in the bi-lingual lexicon.

4.1.2 Evaluation Setting

We evaluate our approach in an end-to-end fashion. For a node c in the Chinese BINet, we choose the node e^* which has the highest score as c's counterpart in the English BINet:

$$e^* = \arg \max_{e \in Cand(c)} Score(c, e)$$

We rank the aligned node pairs by the score and manually evaluate the quality of the top K pairs. A pair $\langle c,e \rangle$ is annotated as correct if e is a correct translation of c or e refers to an entity that c refers to. The annotation assignment is done by three human judges with 89.4% agreement. The disagreement mainly arises from the ambiguity of some named entities. In the evaluation, we consider $\langle c,e \rangle$ correct if more than two judges annotate it as correct.

We compare our approach to the following baselines that use various combinations of clues to verify candidates for decipherment as well as the state-of-the-art algorithm for language decipherment from non-parallel corpora:

- Pronunciation verification (*pv*): Use the pronunciation clue only
- Translation verification (*tv*): Use the translation clue only
- Neighbor verification (*nv*): Use the neighbor clue only to decipher the BINet through propagation.
- Correlation of burst verification: (*cv*): Use the burst correlation clue only
- *pv*+*tv* and *pv*+*tv*+*nv*
- *Bayesian Inference*: Bayesian inference based decipherment approach (Dou and Knight, 2012) based on the alignment of bigram language

---

[4] Due to the upper bound of $Conf(G_c, G_e)$, the algorithm must terminate after several iterations.

[5] We discarded the edges whose weight is smaller than a threshold (5 for Chinese and 20 for English BINet given the difference of their data size) for removing trivial connections.

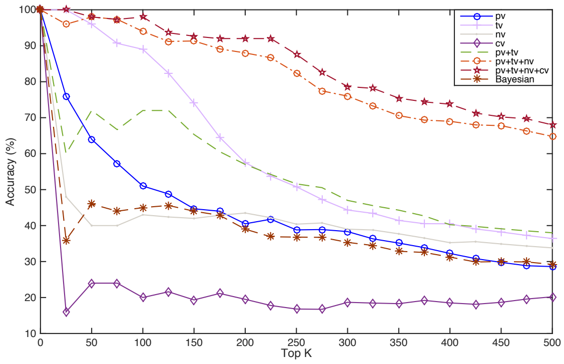

Figure 4: Accuracy curves of various approaches. Note that *pv+tv+nv+cv* is our final approach.

models across languages. We adapt it to our experiment setting by considering adjacent nodes in a BINet as bigrams for decipherment.

We used 2009 AFP Chinese/English news in Gigaword as our development set to tune hyperparameters. Since our approach has only 4 parameters (i.e., η, λ, γ, δ in Eq (3)), it is easy to tune the parameters using grid search (from 0.0 to 1.0 with a step 0.2) on the development set. For baselines except Bayesian inference, the score computation function is almost identical to Eq (3) except that the weights of the clues which are not used are set to 0.

4.1.3 Results

We present the results in Figure 4. Our approach outperforms all the baselines because it considers various clues for decipherment. Among the baselines, accuracy scores of *pv* and *tv* drop dramatically with K increasing because a single clue can only decipher a limited number of nodes effectively. *pv+tv* seems to alleviate the problem to some extent: its accuracy does not drop so drastically as *pv* or *tv* because multiple clues allow us to decipher more nodes but its accuracy is still not desirable. Among the clues, *cv* performs worst, demonstrating that the burst correlation clue alone is far from enough for decipherment. Compared with *pv*, *tv* and *cv*, *nv* deciphers the nodes in the Chinese BINet through propagation but the neighbor clue alone is not sufficient for accurate decipherment. It is notable that *nv* achieves comparable performance to the Bayesian inference method which uses similar clues, demonstrating the effectiveness of our decipherment framework despite its simplicity. Moreover, our graph-based decipherment approach is more flexible to incorporate a variety of clues. When it is combined with *pv+tv*, the performance shows a significant boost and achieves approximately 90% accuracy in the top 200 results though it is slightly inferior to our final approach due to the lack of awareness of burst correlation.

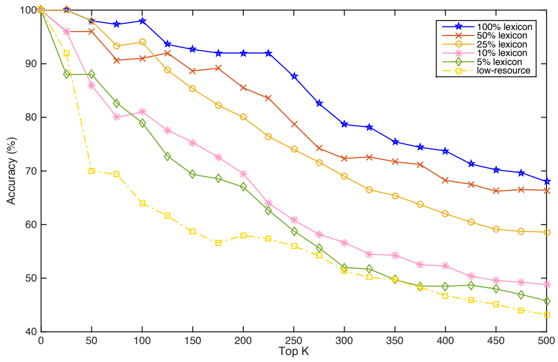

Figure 5: Accuracy curves of our approach with different resource settings.

Another interesting observation from Figure 4 is that our approach clearly know the confidence of its predictions. For top 100 mined pairs with the highest confidence scores (i.e., the score in Eq (3)), **the accuracy is 98%**. Therefore, it is easy to control the quality of mined pairs, which is important for a text mining algorithm.

We also study the effect of language resources on the performance. We first randomly sample different sizes of entries from the original bi-lingual lexicon as new bi-lingual lexicons. The results[6] in Figure 5 show the accuracy improves as the size of bi-lingual lexicon grows because more prior knowledge benefits deciphering the BINet. In addition, we test our approach in a low-resource setting where there is no knowledge of the romanization system (i.e., pinyin) and no pre-trained word segmentation and name tagging tools are available. The only available resource is a very small bi-lingual lexicon with 1,000 most common Chinese words[7] and their corresponding English translations. In this setting, we use an unsupervised Chinese word segmentation approach combining a Hierarchical Dirichlet Process (HDP) model with a Bayesian HMM model (Chen et al., 2014) to segment Chinese text instead of the preprocessing steps mentioned in Section 4.1.1. According to Figure 5, our approach still performs well in the low-resource setting although its accuracy curve is lower than that in rich-resource settings, demonstrating it can work in both rich- and low-resource settings.

---

[6] The sample processes are repeated for 3 times and the results are the averaged accuracy.

[7] We sample these Chinese words based on IDF.

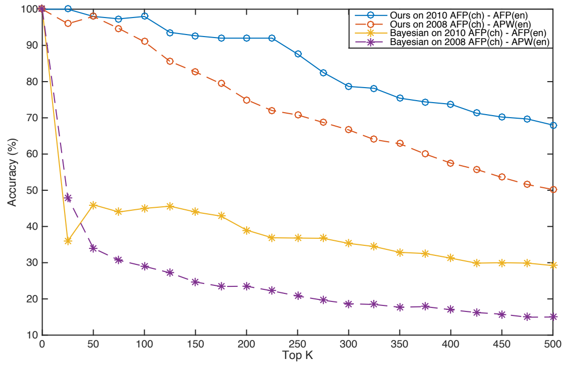

Figure 6: Accuracy curves on multiple datasets.

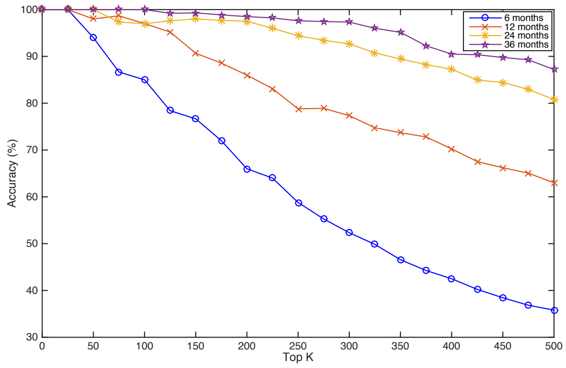

Figure 7: Stream alignments derived from 2008 on the Chinese and English AFP news streams. With more data (i.e., streams within a longer time frame) being aligned, our approach can harvest more high-quality alignments.

In order to test the generalization ability, we evaluate our approach using the same hyperparameters on another coordinated text streams – AFP Chinese and APW English news stream in 2008. The results in Figure 6 show that our decipherment approach consistently outperforms the other baseline and still deciphers the top 100 nodes in high accuracy even though the curve in 2008 is lower than that in 2010. The performance difference in 2008 and 2010 mainly arises from the difference on topic overlaps. In the streams of 2010, the Chinese and English news are from the same news agency (i.e., AFP). Therefore, the topic overlaps of the cross-lingual streams are larger than 2008, allowing more nodes to be deciphered correctly.

Finally, we investigated the performance of our approach under various sizes of data provided, as shown in Figure 7. As observed, when the data size is small (e.g., 6-month coordinated text streams), the approach works poorly because there are very few nodes in BINets that can be aligned. As the data size increases, our approach can efficiently[8] harvest a growing number of high-quality alignments, as reflected by the higher curves in Figure 7. Considering massive coordinated text streams generated every day, if the approach can be applied to the endless streams, it is possible to monitor the streaming data and derive countless alignments for never-ending language knowledge acquisition.

Node	Chinese (burst period)	English (burst period)
1	贝鲁斯柯尼 (346-348)	Berlusconi (344-348)
2	卫报 (332-337)	Guardian (334-336)
3	曼城 (360-363)	Manchester City (358-361)
4	鸠山 (147-158)	Hatoyama (147-158)
5	空域 (106-111)	airspace (104-111)
6	纾困 (119-130)	bailout (112-129)
7	国际奥林匹克委员会 (38-45)	IOC (36-44)
8	恋童癖 (102-104)	paedophilia (90-108)
9	小威 (147-153)	Serena Williams (146-148)
10	东协 (200-203)	ASEAN (198-203)
11	东南亚国家协会 (299-302)	ASEAN (299-303)
12	央行 (129-130)	European Central Bank (125-129)
13	央行 (75-76)	Bank of Japan (73-76)
14	翁山 (310-311)	Aung San Suu Kyi (308-313)
15	苏姬 (310-311)	Aung San Suu Kyi (308-313)

Table 1: Alignment examples. The numbers of burst periods denote the number of days after Jan 1, 2010.

4.2 Endless language knowledge mining

Table 1 shows the stream alignment result of our approach. As demonstrated above, we can derive a variety of language knowledge from the fine-grained cross-lingual alignments.

Word/entity translations are the main knowledge that can be derived from our alignment results by extracting word pairs from the aligned cross-lingual node pairs. Formally, we find a Chinese word w's English translation w^* as follows:

$$w^* = w(e^*)$$

$$e^* = arg\,\max\nolimits_{e \in V_e} \max\nolimits_{c \in V_c(w)} Score(c, e)$$

where $V_c(w)$ is the set of Chinese nodes whose word is w, and $w(e)$ denotes the word of node e.

We evaluate our approach on mining translations of bursty Chinese words, based on the evaluation criteria of bilingual lexicon extraction. Specifically, we test how many out-of-vocabulary (OOV) words appearing in the Chinese BINet are correctly translated. The datasets used for evaluation are the 2010 and 2008 streams in Figure 6. In total, there are 1,226 and 1,082 distinct Chinese OOV words (excluding incorrectly segmented words) in the corresponding Chinese BINets. Accuracy is used to measure the proportion of the words being correctly translated, as (Tamura et al., 2012) did.

Table 2 compares our approach to representative bilingual lexicon extraction approaches. CONTEXT is one of the earliest approaches for extracting word translations from comparable cor-

---

[8] Efficiency is reported in the supplementary notes.

Model	Acc_1(2010)	Acc_1(2008)
Context (Fung and Yee, 1998)	0.32%	0.37%
CoLP (Tamura et al., 2012)	0.32%	0.46%
SimLP (Tamura et al., 2012)	0.49%	0.46%
Diverse (Schafer and Yarowsky, 2002)	5.22%	4.25%
DiverseSP (Sproat et al., 2006)	5.46%	4.44 %
Bayesian(LM) (Dou and Knight, 2012)	0.57%	0.55%
Bayesian(BINet)	11.17%	4.81%
Ours	**28.38%**	**19.78%**

Table 2: Performance of translating bursty words.

pora based on context similarity. CoLP and SimLP are label propagation models on word co-occurrence and similarity graphs for bilingual lexicon extraction. Diverse is a variant of Context by adding various information (e.g., pronunciation and temporality) and DiverseSP is the approach using phonetic and frequency correlation with a score propagation strategy. Bayesian is the Bayesian decipherment approach which has been introduced in the previous section, and it is evaluated in two settings (i.e., based on traditional bigram language models and BINets). According to Table 2, our approach substantially outperforms the other approaches on both datasets, showing its advantages for mining translation of bursty words in coordinated text streams. It is also notable that the BINet-based Bayesian improves the LM-based counterpart, demonstrating the advantage of burst-level alignment for this task.

In addition to the comparisons to the classical baselines, we also test the latest representative unsupervised bi-lingual lexicon extraction approaches (Zhang et al., 2017a,b) based on word embedding and generative adversarial nets (GANs). Unfortunately, these approaches do not perform well in our setting. For example, the approach in (Zhang et al., 2017a) achieved $<1\%$ accuracy[9]. One reason is that the topic overlap of coordinated cross-lingual text streams is not so significant as the Wikipedia data used for their experiments, and the other reason is that their approaches focus on common fundamental words like “城市(city)” while our targets are OOVs like “东协(ASEAN)” which do not frequently appear in a corpus. In contrast, our approach is more practical: it not only works well in easily available and endless coordinated text streams without high content overlap requirement, but also can accurately mine translations of many OOVs which do not appear frequently and really need mining their translations.

Knowledge	Derived	Correct	Acc
Word/entity translation pair	500	416	0.83
Polysemy/multi-referential word	11	8	0.73
Synonym/Co-referential word pair	72	49	0.68
Entity phrase	99	84	0.85

Table 3: Knowledge derived from top 500 alignments obtained by aligning AFP Chinese and English text streams from 2002 to 2010.

As illustrated in Figure 1, besides word/entity translations, various types of knowledge can also be derived from the BINet alignment results as by-products. For example, for node 9 in Table 1, deciphering the nickname “小威” into *Serena Williams* can benefit cross-lingual entity linking. Nodes 10-11 also demonstrate the potential effect on synonym detection, entity linking and coreference resolution, like the case of Figure 1(b). Nodes 12-13 show that the deciphered BINets can detect polysemous/multi-referential word like “央行(Central bank)” which may refer to different entities during different burst periods, like Figure 1(c). Moreover, the deciphered BINets can also help entity phrase extraction based on the idea of Figure 1(d). For example, in nodes 14-15, 翁山苏姬(Aung San Suu Kyi) is not recognized as a person name by the Chinese name tagger; instead, it is mistakenly separated into two words – 翁山(Aung San) and 苏姬(Suu Kyi). However, since 翁山(Aung San) and 苏姬(Suu Kyi) are deciphered into the same English named entity – Aung San Suu Kyi, we can merge them back to form the correct entity.

For evaluating our approach's performance on language knowledge acquisition, we align the AFP Chinese-English text streams from 2002 to 2010. The Chinese stream has 119,196 documents and the English one contains 1,608,636 documents. Our approach obtained 7,211 node alignments[10]. Among them, we focus on the top 500 alignments to guarantee their quality and use the aforementioned idea for deriving language knowledge.

Table 3 shows the result of deriving knowledge from the alignments. Among top 500 alignments, we derived 416 correct word/entity translation pairs with 83% accuracy. Also, we correctly derived 8 polysemous/multi-referential words, 49 synonymous/co-referential word pairs and 84 entity phrases as byproducts. It is notable that the data size of coordinated cross-lingual text streams available on the web is much larger than that used in our experiment and they are endlessly updated.

---

[9] We implement this approach using the codes released by the authors. Their reported accuracy for the common words with over 1,000 occurrences is 2.53% on Gigaword corpus.

[10] The alignments with a low score (< 0.05) are discarded.

That means it is promising to endlessly derive language knowledge by applying our approach to the huge size of endless cross-lingual text streams, which may benefit NLP applications like machine translation, entity linking and name tagging.

5 Related Work

Previous studies on cross-lingual text stream alignment tend to focus on coarse-grained (i.e., topic-level) alignment for finding common patterns (Wang et al., 2007; De Smet and Moens, 2009; Wang et al., 2009; Zhang et al., 2010; Hu et al., 2012) and discovering parallel sentences and documents (Munteanu and Marcu, 2005; Enright and Kondrak, 2007; Uszkoreit et al., 2010; Smith et al., 2010; Krstovski and Smith, 2011, 2016) across languages. Studies on fine-grained cross-lingual alignment are mainly for bilingual lexicon induction (e.g., (Fung and Yee, 1998; Rapp, 1999; Koehn and Knight, 2002; Schafer and Yarowsky, 2002; Shao and Ng, 2004; Schafer III, 2006; Hassan et al., 2007; Haghighi et al., 2008; Udupa et al., 2009; Klementiev and Callison-Burch, 2010; Tamura et al., 2012; Irvine and Callison-Burch, 2013, 2015b; Kiela et al., 2015; Irvine and Callison-Burch, 2015a; Vulic and Moens, 2015; Cao et al., 2016; Zhang et al., 2017b,a)) and name translation mining (e.g., (Sproat et al., 2006; Klementiev and Roth, 2006; Udupa et al., 2008; Ji, 2009; won You et al., 2010; Kotov et al., 2011; Lin et al., 2011; Sellami et al., 2014)) from non-parallel corpora. However, these approaches are mainly developed for general comparable corpora, not specially for cross-lingual text streams; thus many of them did not use the powerful stream-level information (e.g., co-burst across languages). In contrast to the word-level alignment methods, we attempt to mine burst-level alignment to largely narrow down candidates, and introduce powerful clues for improving accuracy and discovering various language knowledge.

In contrast to previous cross-lingual projection work like data transfer (Pado and Lapata, 2009) and model transfer (McDonald et al., 2011), we do not require any parallel data. Moreover, our BINets are cheap to construct, which can be easily extended to other languages. This is also the first attempt to apply the decipherment idea (e.g., (Ravi and Knight, 2011; Dou and Knight, 2012; Dou et al., 2014)) to graph structures instead of sequence data.

6 Conclusions and Future Work

This paper proposes an approach to deciphering the Burst Information Network constructed from foreign languages as a novel way to align cross-lingual text streams. For the first time we propose to model stream alignment as a network decipherment problem. By leveraging the network structures with stream-level burst features as well as various clues, our approach can accurately align the important information units across languages and derive a variety of knowledge. Given that our approach is unsupervised, effective, intuitive, interpretable, and easily implementable, it is promising to use it as a framework for never-ending language knowledge mining from big data, which might benefit NLP applications such as machine translation and cross-lingual information access.

For future work, we plan to 1) conduct more experiments and analyses following this preliminary study to verify our approach's effectiveness for more languages and domains (e.g., social stream VS news stream); 2) attempt to use word embedding (e.g., word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018)) for local context encoding and use it as a clue for decipherment; 3) apply our approach to real-time coordinated text streams for never-ending knowledge mining and use the mined knowledge to improve the downstream applications.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. We also want to thank Xiaoman Pan, Dr. Taylor Cassidy, Dr. Clare R. Voss, Prof. Jiawei Han, Prof. Sujian Li and Prof. Yu Hong for their helpful comments and discussions. This work is supported by NSFC project 61772040 and 61751201. Heng Ji's work has been supported by the U.S. DARPA AIDA Program No. FA8750-18-2-0014, Air Force No. FA8650-17-C-7715, ARL NS-CTA No. W911NF-09-2-0053 and NSF Awards IIS-0953149 and IIS-1523198. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. The contact author is Zhifang Sui.

References

Hailong Cao, Tiejun Zhao, Shu Zhang, and Yao Meng. 2016. A distribution-based model to learn bilingual word embeddings. In *COLING*.

Miaohong Chen, Baobao Chang, and Wenzhe Pei. 2014. A joint model for unsupervised chinese word segmentation. In *EMNLP*.

Wim De Smet and Marie-Francine Moens. 2009. Cross-language linking of news stories on the web using interlingual topic modelling. In *ACM workshop on Social web search and mining*.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *EMNLP*.

Qing Dou, Ashish Vaswani, and Kevin Knight. 2014. Beyond parallel data: Joint word alignment and decipherment improves machine translation. In *EMNLP*.

Jessica Enright and Grzegorz Kondrak. 2007. A fast method for parallel document identification. In *NAACL*.

Pascale Fung and Lo Yuen Yee. 1998. An ir approach for translating new words from nonparallel and comparable texts. In *COLING-ACL*.

Tao Ge, Lei Cui, Baobao Chang, Sujian Li, Ming Zhou, and Zhifang Sui. 2016a. News stream summarization using burst information networks. In *EMNLP*.

Tao Ge, Lei Cui, Baobao Chang, Zhifang Sui, and Ming Zhou. 2016b. Event detection with burst information networks. In *COLING*.

Tao Ge, Lei Cui, Heng Ji, Baobao Chang, and Zhifang Sui. 2016c. Discovering concept-level event associations from a text stream. In *NLPCC*.

David Graff and Ke Chen. 2005. Chinese gigaword. *LDC Catalog No.: LDC2003T09, ISBN*, 1:58563–58230.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

A. Haghighi, P. Liang, T. Berg-Kirkpatrick, and D. Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*.

Ahmed Hassan, Haytham Fahmy, and Hany Hassan. 2007. Improving named entity translation by exploiting comparable and parallel corpora. In *RANLP*.

Shuo Hu, Takahashi Yusuke, Liyi Zheng, Takehito Utsuro, Masaharu Yoshioka, Noriko Kando, Tomohiro Fukuhara, Hiroshi Nakagawa, and Yoji Kiyota. 2012. Cross-lingual topic alignment in time series japanese/chinese news. In *PACLIC*.

A. Irvine and C. Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *NAACL*.

Ann Irvine and Chris Callison-Burch. 2015a. A comprehensive analysis of bilingual lexicon induction. *Computational Linguistics*, 1(1).

Ann Irvine and Chris Callison-Burch. 2015b. Discriminative bilingual lexicon induction. *Computational Linguistics*, 1(1).

Heng Ji. 2009. Mining name translations from comparable corpora by creating bilingual information networks. In *the 2nd Workshop on Building and Using Comparable Corpora: from Parallel to Non-parallel Corpora*.

Douwe Kiela, Ivan Vulic, and Stephen Clark. 2015. Visual bilingual lexicon induction with transferred convnet features. In *EMNLP*.

Jon Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397.

Alexandre Klementiev and Chris Callison-Burch. 2010. Bilingual lexicon induction for low-resource languages. In *JHU Technical Report*.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *COLING-ACL*.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *ACL workshop on Unsupervised lexical acquisition*.

Alexander Kotov, ChengXiang Zhai, and Richard Sproat. 2011. Mining named entities with temporally correlated bursts from multilingual web news streams. In *WSDM*.

Kriste Krstovski and David A Smith. 2011. A minimally supervised approach for detecting and ranking document translation pairs. In *WMT*.

Kriste Krstovski and David A Smith. 2016. Bootstrapping translation detection and sentence extraction from comparable corpora. In *NAACL*.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *EMNLP*.

Wen-Pin Lin, Matthew Snover, and Heng Ji. 2011. Unsupervised language-independent name translation mining from wikipedia infoboxes. In *EMNLP Workshop on Unsupervised Learning for NLP*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language ssing toolkit. In *ACL: System Demonstrations*.

R. McDonald, S. Petrov, and K. Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *EMNLP*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

S. Pado and M. Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *ACL*.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *ACL*.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *CoNLL*.

Charles F Schafer III. 2006. *Translation discovery using diverse similarity measures*. Johns Hopkins University.

Rahma Sellami, Fatiha Sadat, and Lamia Belguith Hadrich. 2014. Mining named entity translation from non parallel corpora. In *FLAIRS*.

Li Shao and Hwee Tou Ng. 2004. Mining new word translations from comparable corpora. In *COLING*.

Jason R Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *NAACL*.

Richard Sproat, Tao Tao, and ChengXiang Zhai. 2006. Named entity transliteration with comparable corpora. In *ACL*.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2012. Bilingual lexicon extraction from comparable corpora using label propagation. In *EMNLP*.

Raghavendra Udupa, K Saravanan, A Kumaran, and Jagadeesh Jagarlamudi. 2008. Mining named entity transliteration equivalents from comparable corpora. In *CIKM*.

Raghavendra Udupa, K. Saravanan, A. Kumaran, and Jagadeesh Jagarlamudi. 2009. Mint: A method for effective and scalable mining of named entity transliterations from large comparable corpora. In *EACL*.

Jakob Uszkoreit, Jay M Ponte, Ashok C Popat, and Moshe Dubiner. 2010. Large scale parallel document mining for machine translation. In *COLING*.

Ivan Vulic and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *ACL*.

X. Wang, C. Zhai, X. Hu, and R. Sproat. 2007. Mining correlated bursty topic patterns from coordinated text streams. In *KDD*.

Xiang Wang, Kai Zhang, Xiaoming Jin, and Dou Shen. 2009. Mining common topics from multiple asynchronous text streams. In *WSDM*.

Gae won You, Seung won Hwang, Young-In Song, Long Jiang, and Zaiqing Nie. 2010. Mining name translations from entity graph mapping. In *EMNLP*.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT-NAACL*.

Duo Zhang, Qiaozhu Mei, and ChengXiang Zhai. 2010. Cross-lingual latent topic extraction. In *ACL*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017a. Adversarial training for unsupervised bilingual lexicon induction. In *ACL*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *EMNLP*.

WECA: A WordNet-Encoded Collocation-Attention Network for Homographic Pun Recognition

Yufeng Diao[1,2], **Hongfei Lin**[1]*, **Di Wu**[1], **Liang Yang**[1], **Kan Xu**[1],
Zhihao Yang[1], Jian Wang[1], Shaowu Zhang[1], Bo Xu[1], DongyuZhang[1]
[1]DaLian University of Technology, Da Lian, China
[2]Inner Mongolia University for Nationalities, Tong Liao, China
diaoyufeng@mail.dlut.edu.cn, hflin@dlut.edu.cn
wudi@dlut.edu.cn, liang@dlut.edu.cn
xukan@dlut.edu.cn, yangzh@dlut.edu.cn
wangjian@dlut.edu.cn, zhangsw@dlut.edu.cn
xubo2011@mail.dlut.edu.cn, zhangdongyu@dlut.edu.cn

Abstract

Homographic puns have a long history in human writing, widely used in written and spoken literature, which usually occur in a certain syntactic or stylistic structure. How to recognize homographic puns is an important research. However, homographic pun recognition does not solve very well in existing work. In this work, we first use WordNet to understand and expand word embedding for settling the polysemy of homographic puns, and then propose a WordNet-Encoded Collocation-Attention network model (WECA) which combined with the context weights for recognizing the puns. Our experiments on the SemEval2017 Task7 and Pun of the Day demonstrate that the proposed model is able to distinguish between homographic pun and non-homographic pun texts. We show the effectiveness of the model to present the capability of choosing qualitatively informative words. The results show that our model achieves the state-of-the-art performance on homographic puns recognition.

1 Introduction

A pun is a writers use of a word in an ambiguous and inconsistent way in language, often to play on the different meanings of the word or utilize similarly pronounced sounds for a common humorous effect. Puns are widely used in written and spoken literature, which intended as jokes. For example, Tom Swifty by(Lippman and Dunn, 2000), in which puns usually occur in a certain syntactic or stylistic structure. From literature, speeches and oral storytelling, puns are also a standard rhetorical device, which also can be applied non-humorously. For instance, Shakespeare is well known for his puns, which continually appeared in his non-comedic works by (Tanaka, 1992). Both humorous and non-humorous puns have been the theme of extensive and attractive works that has led to discernment for the nature of puns with double meaning.

There are many relevant studies on pun recognition in natural language processing. Many scholars attempted to classify puns according to the similar relationship between the pronunciations and double meanings of the words. For example, (Pafford, 1987) categorizes pun into homophonic puns and homographic puns, which used homonyms and polysemy of words respectively. The research on pun recognition has carried out according to this classification system of Redfern. Our work also considers that puns consist of homophonic and homographic puns.

Type of Pun	Example	Pun Word
Homographic	I used to be a banker but I lost interest.	interest
Homophonic	When the church bought gas for their annual barbecue, proceeds went from sacred to the propane.	propane

Table 1: Pun Examples

Both homographic puns and homophonic puns have double meanings to increase deep impression in a certain environment. However, two types of puns have their own features, respectively. Homographic puns, as an important class of puns, which the words for two senses of puns share the same orthographic form. While homophonic puns have the similarity in pronunciations with double senses that distinguished from homographic puns. The former one mainly settles synonyms, while the latter one solves homonyms. Because of the difference, we can not use the unified model to distinguish. Table 1 illustrates the examples of homographic pun and homophonic pun.

In this study, we mainly focus on homographic

---

* Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2507–2516
Brussels, Belgium, October 31 - November 4, 2018.

puns since they widely used everywhere (Miller, Tristan and Turković, Mladen, 2016) and easily obtain in existing corpus. However, homographic puns recognition in the current works does not solve very well because of their confused double meanings.

To solve the mentioned problem, we propose a computational WordNet-Encoded Collocation-Attention network model (WECA) to recognize homographic puns. Our model takes semantic word embedding and collocation into account for homographic puns recognition. Based on the experiments, the results show that our work will improve the performance of homographic puns recognition. This work is the first to recognize homographic puns with improved word representation and attention mechanism to the best of knowledge. Here, our contributions are as follows.

- The paper applies the lexical ontology WordNet to understand and extend the word embedding for solving the polysemy of homographic puns.
- The paper proposes a neural attention mechanism to extract the collocation for homographic puns classification, which combined with Bi-LSTM to obtain the context weights.
- Experimental results on the datasets of Semeval2017 Task7 and Pun of the Day demonstrate our method outperforms several baselines for recognition homographic puns. Furthermore, visualization of selected examples show the reasons that this model works well.

The rest of this paper is structured in the following. Section 2 mainly reviews the related work on word representation and puns classification. Section 3 presents our proposed word embedding and collocation attention-based network model. Section 4 shows our experiments and discusses evaluation results. Finally, Section 5 concludes our research contributions and offers the future work.

2 Related Work

In this section, we will review related works on word representation and homographic pun recognition for homographic puns classification briefly.

2.1 Word Representation

In recent years, word representation has the great improvement because it solves data sparsity problem and obtain more semantic relations between words compared with one-hot representation.

(Rumelhart et al., 1986) proposed the idea of word distributed representation, which converts all the words into a low-dimensional continuous semantic space. This space took each word as a vector. These distributed low-dimensional word representation have been widely applied in many NLP tasks, including machine translation(Sutskever et al., 2014; Bahdanau et al., 2014), text classification (Niu et al., 2017; Du et al., 2017), neural language models (Mikolov et al., 2010, 2013) and parsing (Chen and Manning, 2014; Chen et al., 2015). Word embedding is taken as the essential and available inputs for NLP tasks, which enables encoding semantic representation in meaningful vector space.

The studies show that word representations are useful to achieve a good balance between effectiveness and efficiency, such as Word2Vec (Mikolov et al., 2013) and GloVe(Pennington et al., 2014). Therefore, the semantic meanings of words can reflect in the contexts according to these distributed representation models.

However, homographic puns always have multiple meanings. The word representation, considering as only one vector for each word, which puzzled by the understanding for polysemy of puns. This paper combines the representations of lemmas, synsets and words from WordNet[1](Miller, 2002) to understand multiple meanings of homographic puns. The lemma and synset annotation in WordNet provide helpful semantic information for detecting homographic puns.

2.2 Homographic Pun Recognition

In recent years, homographic puns have increasingly become a respectable research topic, which widely appears in rhetoric and literary criticism. However, there were little related works in the fields of computational linguistics and natural language processing by (Miller, Tristan and Turković, Mladen, 2016). In this subsection, we mainly introduce some puns detecting methods.

There are many useful methods to classify the puns in NLP. For example, (Kao et al., 2016; Huang et al., 2017) used a probability statistical model to capture the latent semantic information between words for detecting homographic puns. (Jaech et al., 2016) proposed a new prob-

---

[1] WordNet: http://wordnet.princeton.edu/

ability model to learn phoneme edit probabilities for classifying the homophonic puns. The system ECNU(Xiu et al., 2017) applied a supervised training classifier, which helpful features derived from WordNet and Word2Vec embeddings to distinguish between homographic puns. The system Fermi (Indurthi and Oota, 2017) employed a supervised approach for the detection of homographic puns. It used a bi-directional RNN for a classification model and adopted the distributed semantic word embeddings as input features. These methods do not consider the collocation between words in homographic puns.

The attention mechanism proposed by (Bahdanau et al., 2014) to settle machine translation problem, which was used to select the reference words for words before translation. (Xu et al., 2015) used attention model for image generation to select the similar image regions. For text classification, (Yang et al., 2016) applied attention mechanism into solving document-level classification. Many other tasks in NLP used this mechanism, including natural language question answering (Kumar et al., 2015), parsing (Vinyals et al., 2014), image question answering(Yang et al., 2015), and classification(Shen et al., 2018; Tan et al., 2018). Therefore, this model is capable of discovering the important and semantic information. Meanwhile, attention mechanism can also improve the performance of classification tasks.

Hence, we explore an attention mechanism for collocation to mine the latent semantic information between the part of speech words to achieve good result for homographic puns recognition.

3 Methods

Homographic puns recognition could influence by considering both the semantic word embedding and collocation with the context weights for homographic puns. In this section, we propose our model as WordNet-Encoded Collocation-Attention network (WECA). Figure 1 demonstrates the overall structure of our model.

It consists of three main components: an improved word embedding with WordNet-Encoded as inputs, a Bidirectional Long Short-Term Memory (Bi-LSTM) as context weights in a sentence for homographic puns and a fully-connected network as the collocation-attention mechanism. The attention networks combined by a concatenate operation to discover the collocation. Then the context weights and attention networks combined by an element-wise multiplication operation in the classification layer. We describe the details of three components as follows.

3.1 WordNet-Encoded Word Embedding

The homographic pun is a clever trick to let one word relate to two aspects or multiple meanings. For example, “Before he sold Christmas trees, he got himself spruced up” The pun word spruced has two meaning: one meaning is spruce tree, while the other is making oneself or something look neater and tidier. We find this word spruced means the last meaning in this situation. Therefore, the polysemy of the ambiguity from homographic puns need additional large lexical ontology. Thus, we apply WordNet for computational linguistics and natural language processing.

Polysemy is critical factor for recognizing homographic puns. To combine the information of multiple meanings, we propose giving a WordNet-Encoded model (WE) to obtain the word embedding for each word. WordNet is a lexical ontology of words. Each word has multiple semantics corresponding with respect to different senses and each sense corresponds to multiple words.

We introduce lemmas, synsets (senses) and words in WordNet. For example, the word is “interest”. The word “interest” has three main synsets: sake (a reason for wanting something done), pastime (a diversion that occupies one’s time and thoughts) and interest (a sense of concern with and curiosity about someone or something). The lemmas eliminate the ambiguity of each sense. For instance, the synset pastime represents a diversion that occupies one’s time and thoughts, which contains lemmas pastime, interest and pursuit. Then we propose two strategies to generate the Word-Net-Encoded embedding based on the information of lemmas and synsets information, Average Lemma Aggregation Model (ALA) and Weighted Lemma Aggregation Model (WLA).

Average Lemma Aggregation Model (ALA) adopts a strategy of equal weight according to meanings of homographic puns. ALA model mixes all the lemmas of all the senses of a word together for each word. Hence, it represents the target word by using the average of its whole lemma embedding and puts this together on the original

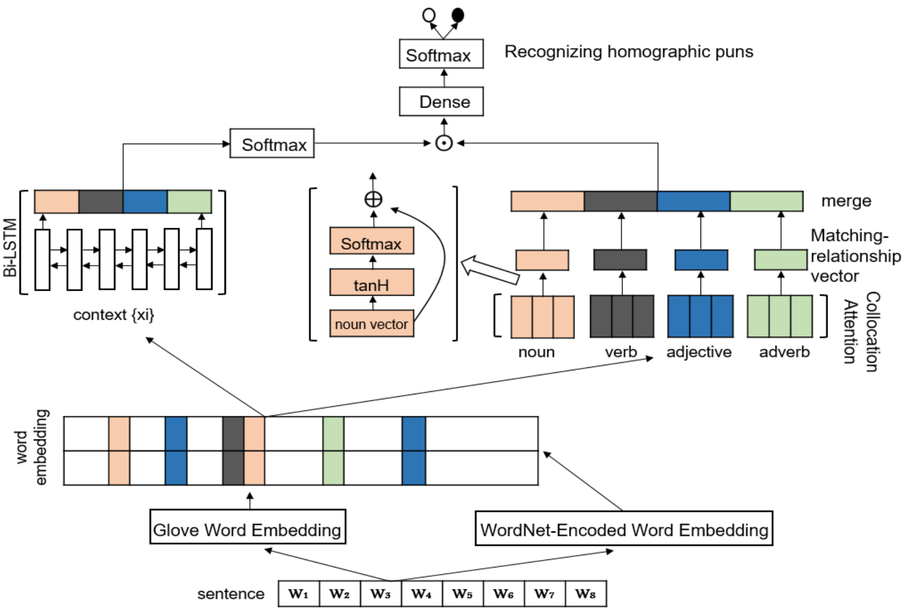

Figure 1: WordNet-Encoded Collocation-Attention network model(WECA)

vector of target word. The formula is as follows:

$$w = \frac{1}{m} \sum_{{s_i}^{(w)} \in S^{(w)}} \sum_{{l_i}^{(s_j)} \in {L_i}^{(s_j)}} w^{{l_j}^{s_j}} \qquad (1)$$

which means the new embedding vector of w is determined by the average of all its lemma embedding. Here, m represents the number of lemmas with overlapping senses with respect to the word w, s_i is the sense i, l_j is the lemma j. Finally, word embedding of w is the concatenation of the original vector and above new vector.

ALA model can apply lemmas to encoding latent semantic relationship because lemmas share the information by multiple words and senses. Therefore, words sharing the same lemmas are likely to obtain the similar representations.

Weighted Lemma Aggregation Model (WLA) The ALA Model takes the lemma embedding to encode lemma information for word representation. Although ALA model represents the average of all the lemma, which does not consider the importance of certain lemmas. Hence, we construct embedding for a target word with the help of word senses and lemmas in WordNet that we called WLA model. The formula is as follows:

$$w = \sum_{{s_i}^{(w)} \in S^{(w)}} \frac{|{L_i}^{(s_i)}|}{m} \sum_{{l_i}^{(s_j)} \in {L_i}^{(s_j)}} w^{{l_j}^{s_j}} \qquad (2)$$

where m represents for all the number of lemmas with overlapping senses with respect to the word w, s_i is the sense i, l_j is the lemma j, ${l_i}^{(s_j)}$ is the number of lemmas in each sense with target word, w is the new embedding considering the weighted lemma information. Then, the target word embedding concatenates new vector to original vector.

The weighted lemma strategy assumes one sense of word obtains more attention if this sense of word has more lemmas. We can show each word as a special distribution on the sense. From the results, WLA model is the best representation.

3.2 Bidirectional Long Short Term Memory(Bi-LSTM) for Recognizing Homographic Puns

Long Short Term Memory (LSTM) was proposed by Hochreiter and Schmiduber (1997), which has been widely adopted for text processing. There are three gates and one cell in LSTM: an input gate i_t, a forget gate f_t, an output gate o_t and a memory cell c_t. They are all vector in R^d. The equations of transition are:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1}) \qquad (3)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1}) \qquad (4)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_{t-1}) \qquad (5)$$

$$\tilde{c_t} = tanh(W_c x_t + U_c h_{t-1}) \qquad (6)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{7}$$

$$h_t = o_t \odot tanh(c_t) \tag{8}$$

where x_t is an input vector at the current time step, σ is the sigmoid function and $\odot$ is the element-wise multiplication operation, $W_{\{i,f,o,c\}}$,$U_{\{i,f,o,c\}}$,$V_{\{i,f,o,c\}}$ are learned weight parameters, h_t is the hidden state vector. In LSTM, the hidden state h_t only encodes the front context in a forward direction but not consider the backward context.

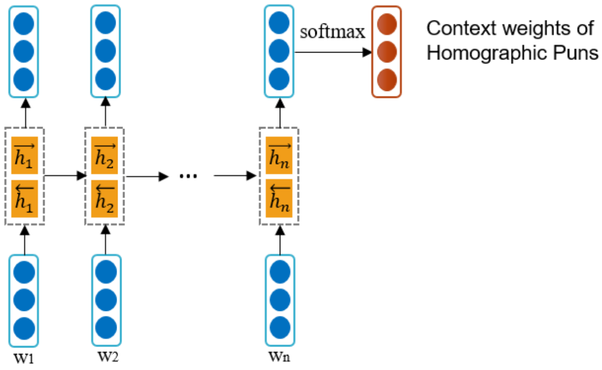

Figure 2: The Structure of Bi-LSTM

In this study, we apply Bi-LSTM model(Graves, 2012) to capture the latent semantic information of homographic puns for obtaining the context weights. For each sentence, it has a forward LSTM $\overrightarrow{h}$ and a backward LSTM $\overleftarrow{h}$ to concatenate the hidden states of two LSTMs as the representation of corresponding word. Figure 2 illustrates the architecture of Bi-LSTM model. $\{w_1, w_2, \cdots, w_n\}$ represent the word vector in a sentence whose length is N. Then, the forward and backward contexts can take into account simultaneously. The equations of transition are:

$$\overrightarrow{h}_t = H(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \tag{9}$$

$$\overleftarrow{h}_t = H(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}) \tag{10}$$

$$h_{out} = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y \tag{11}$$

where $\overrightarrow{h}_t$ is a forward LSTM, $\overleftarrow{h}_t$ is a backward LSTM, h_{out} is the output of Bi-LSTM.

3.3 Collocation-Attention Mechansim

It proposes that not all the words provide the same contribution of word representation for the sentence. Especially for homographic puns recognition, the collocation between candidate pun words in a sentence offers more clues for getting the collocational word weights. Miller points out that the candidate pun words mainly consist of nouns, verbs, adjectives and adverbs in each pun. For example, “The money doesn’t grow on the tree, but it can grow on the branch.” The word “branch” is the pun word. From this example, we know that the collocation of candidate pun words {money, grow, tree, branch}, which should be more important for recognizing homographic puns.

Therefore, it is necessary to learn about the latent relationship in collocation of words. We design an attention mechanism to obtain the collocational weights by extracting such words of collocation from nouns, verbs, adjectives and adverbs, respectively. Then we concentrate on the four parts in sentences with pun to aggregate the informative words for classifying the homographic puns. This model uses an attention network taking word embedding with WordNet-Encoded as input then to extract polysemy attention signal, which made use of polysemy to understand ambiguity of homographic puns. The formula is as follows:

$$u_{ijt} = V \cdot tanh(W_u h_{ijt} + b_w) \tag{12}$$

$$\alpha_{ijt} = \frac{exp(u_{ijt})}{\sum_{t=1}^{T_x} exp(u_{ijt})} \tag{13}$$

$$c_{ij} = \sum_{t=1}^{T_x} \alpha_{ijt} h_{ijt} \tag{14}$$

where h_{ijt} is a hidden state at each time step for each part of speech, $j \in \{nouns, verbs, adjectives, adverbs\}$, u_{ijt} is a hidden representation of h_{ijt} through a one-layer MLP, α_{ijt} is a normalized importance weight through a softmax function with each part of speech, c_{ij} is a context vector as a high level representation over the words from attention-based model by the weighted mean of the hidden state sequence h_{ijt} for each part of speech.

After combining the attention networks with the context weights in a sentence, we merge all the c_{ij} vectors from collocation attention model and take the uniformed context weights in a sentence.

Then we mix the two parts results with element wise multiplication operation to recognize the homographic puns. The formula is as follows:

$$c_i = [c_{inouns}; c_{iverbs}; c_{iadjectives}; c_{iadverbs}] \quad (15)$$

$$l_{out} = Softmax(h_{out}) \quad (16)$$

$$s_i = c_i \cdot l_{out} \quad (17)$$

where c_i is merged by c_{ij}, $j \in \{nouns, verbs, adjectives, adverbs\}$, l_{out} is the softmax function of h_{out}, s_i is the result with the multiplication operation of c_i and l_{out}.

The model can be trained in an end-to-end way by backpropagation, where objective function is the cross-entropy loss. Let y be the target distribution and $\hat{y}$ be the predicted distribution. The goal of training is to minimize the cross-entropy error between y and $\hat{y}$ for all sentences.

$$loss = -\sum_i \sum_j y_i^j log\hat{y_i}^j + \lambda\|\theta\|^2 \quad (18)$$

where i is the index of sentence, j is the index of class. Our classification is two way. λ is the L2 regularization term. θ is the parameter set.

4 Experiments and Evaluation

In this section, we first evaluate the effectiveness of our WordNet-Encode model (WE) on two tasks to detect the polysemy of homographic puns. Then, we examine the performance of our WECA model compared with existing methods. Finally, we show the effectiveness of our model.

4.1 Experimental Setting

In this section, we introduce datasets, evaluation metrics, baseline methods, and present the details of the training process of our model.

Datasets To verify the effectiveness of our proposed model, we use two datasets: SemEval2017 Task7[2] and Pun of the Day[3] .

SemEval2017 Task7. This dataset is composed of homographic and heterographic puns for recognizing and interpreting puns. We focus on homographic puns detection in semantic rather than phonology. The homographic pun word will have at least two words sense in the WordNet(Miller and Gurevych, 2015). Table 1 shows a detailed statistical distribution of our datasets.

Pun of the Day. This dataset only includes pun content in the beginning. Then it collects the negative samples from Yahoo! Answer[4] , AP News[5] , Proverb, and New York Times in order to balance the distribution of positive and negative examples, which adapt to decrease the domain discrepancy. Table 2 provides a complete statistical description of our dataset.

Dataset	Positive	Negative	Average Length
Task7	1607	643	13.1
Pun of the Day	2423	2403	13.5

Table 2: Statistics of Datasets

Metrics We apply the standard measures precision, recall, accuracy and F1-score to evaluate the effectiveness for homographic puns recognition, which also adopted as metrics in SemEval 2017 Task7 evaluation.

Baselines We compare several strong baselines as follows.

LSTM: LSTM without WordNet-Encoded embedding and Collocation-Attention mechanism.

Bi-LSTM: Bi-LSTM without WordNet-Encoded embedding and Collocation-Attention.

Bi-LSTM$_E$: Bi-LSTM with WordNet-Encoded embedding used the WLA model.

Bi-LSTM-Attention: Bi-LSTM with single attention mechanism.

Fermi and N-Hance are the good performing model in the SemEval2017 task7.

Top1 Fermi: Fermi took a supervised approach for homographic puns detection. It did not construct own train data set, but rather split the shared task data set into train sets and test sets(Miller, Tristan and Hempelmann, Christian and Gurevych, Iryna, 2017). It used a Bi-RNN to learn a classification model and treat the word embedding as the input features.

Top2 N-Hance: It assumed every pun had a particularly strong association with exactly one other word in context(Miller, Tristan and Hempelmann, Christian and Gurevych, Iryna, 2017). Then it calculated PMI between words in context to detect and locate puns. If the score exceeded

---

[2] SemEval2017 Task7: htto://alt.qcri.org/semeval2017/task7/

[3] Pun of the Day: htto://www.punoftheday.com/

[4] htto://answers.yahoo.com/

[5] htto://hosted.ap.org/dynamic/fronts/HOME?SITE=AP

a certain threshold, the text assumed to contain a pun. Otherwise, the text assumed to have no pun.

WECA: Here, we use Bi-LSTM with WordNet-Encoded embedding with WLA model and Collocation-Attention mechanism.

Training Details In experiments, our model is tuned with 5-fold cross validation. All word vectors are initialized by GloVe. We use 50, 100, 200 and 300 dimension to verify the performance, respectively. Here, 200 dimension is the best performance. Therefore, we set the dimensions of word, synset and lemma embedding to be 200. The size of units in LSTM is 800. RMSprop is used for our optimization method. We use learning rate decay and early stop in the training process. All models are trained by mini-batch of 64 instances.

4.2 The Effectiveness of WordNet-Encoded Word Embedding

Comparing the GloVe model with our ALA model and WLA model, we evaluate the quality of our improved word representations to detect the homographic puns. In this experiment, we use the same classifier Bi-LSTM and parameters to verify the effectiveness of our word embedding.

Figure 3 and Figure 4 show the results of different word embedding for detecting homographic puns. From the results we can observe that:

(1) Our models ALA and WLA, which outperform the original vector GloVe on both two datasets. It indicates that our model can better capture the semantic relations of words by utilizing lemma annotation properly based on the WordNet.

(2) The ALA model represents each word with the average of its lemma embedding. In general, the ALA model performs better than GloVe, showed that lemma and synset of WordNet is very useful. The reason is the words sharing mutual lemma representation are helpful with each other.

(3) The WLA model mostly performs better than GloVe and ALA model. This model can obtain a weighted distribution according to the senses and lemmas. The results show that their different senses are commonly different from others, but share certain representation.

4.3 Pun Recognition with WordNet-Encoded Collocation-Attention Network (WECA)

Our model WECA, combination of WordNet-Encoded word embedding and Collocation-Attention network with context weight, which performs compared with the suggested baselines.

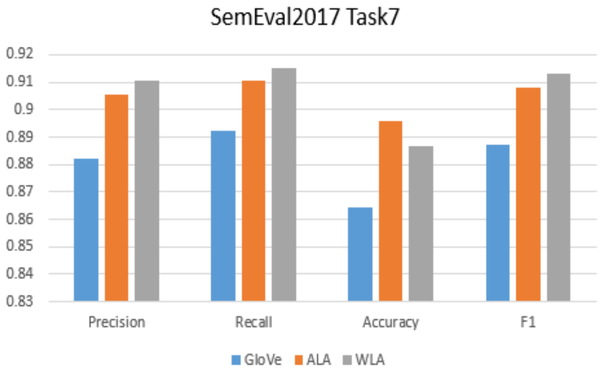

Figure 3: Comparison of Different Word Embedding on SemEval2017 Task7

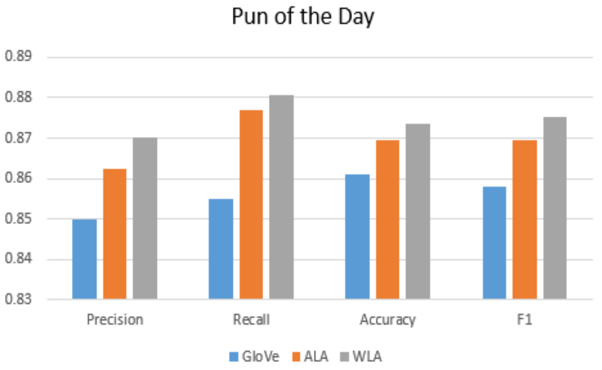

Figure 4: Comparison of Different Word Embedding on Pun of the Day

Here, we use Pun of the Day as the training set to obtain all the parameters, and test the results of homographic pun recognition in SemEval2017 task7. The results are shown in Table 3.

	Precision	Recall	F1
LSTM	81.80	83.7	82.43
Bi-LSTM	85.40	83.64	84.51
Bi-LSTM$_E$	85.87	85.07	85.46
Bi-LSTM-Attention	84.92	85.62	85.26
N-Hance	75.53	**93.34**	83.50
WECA	**89.19**	90.64	**89.21**

Table 3: Comparison of Different Models of Homographic Puns Recognition

(1) Bi-LSTM has the better performance for homographic puns detection compared with LSTM (84.51% vs.82.43%). It shows that Bi-LSTM exploits two parallels to discover more context information. At the same time, Bi-LSTM$_E$ outperforms Bi-LSTM (85.46% vs.84.51%), which demonstrating the effectiveness of the WordNet-Encoded word embedding.

(2) Bi-LSTM-Attention performs slightly better

	Precision	Recall	F1
Fermi	90.24	89.70	89.97
WECA	**91.43**	**90.53**	**90.98**

Table 4: Comparison of WECA and Fermi of Homographic Puns Recognition

than Bi-LSTM and LSTM (85.26% vs. 84.51%, 82.43%). The reason is that the attention mechanism can assign the weight to the whole words according to the context information.

(3) Our model WECA has a better performance compared with Bi-LSTM$_E$, Bi-LSTM-Attention and N-Hance (87.45% vs. 85.46%, 85.26%, 83.50%). N-Hance is the second place in SemEval2017 task7. It shows the WordNet-Encoded word embedding can capture more semantic information between words with the help of lemma and synsets in WordNet. Meanwhile, it presents that the attention network mechanism combined with collocation of the specific part of speech of puns, which capture the characteristic information to recognize the homographic puns.

The best perform of SemEval2017 task7 is Fermi. However, Fermi only evaluates on 675 of 2250 homographic contexts(Miller, Tristan and Hempelmann, Christian and Gurevych, Iryna, 2017) in SemEval2017 task7. Thus, our model uses 675 as a test set and rest of data as a training set. The results are shown in Table 4. Experiment results present our model outperforms Fermi under the same data distribution. It shows the effectiveness of our model again.

4.4 Visualization of Model

In order to verify our model is enable to select the valuable information of words that reflected the collocation, we visualize the attention layers for several sentences in Pun of the Day and SemEval2017 Task7 data sets whose labels are correctly predicted by our model in Figure 5. We choose two examples. One presents collocation between nouns, the other presents collocation between verbs.

Each line is a sentence. Blue denotes word weight. If color of word is darker, the word is more important. Figure 5 shows our WECA model can select words carrying ambiguous meanings from the collocation of homographic puns. For example, in the first sentence, it highlights "interest", which is worthy attracting more attention because of multiple meanings for the pun word. In the second sentence, "sleep" is selected word by our attention model as related to homographic puns. Therefore, attention networks of collocation is effective for recognizing homographic puns.

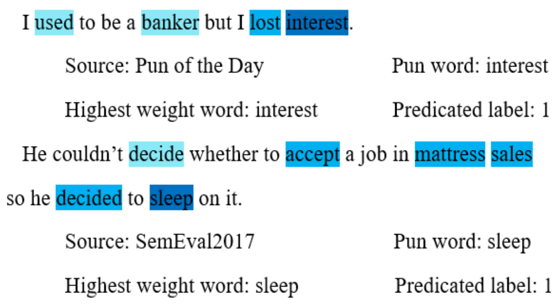

I used to be a banker but I lost interest.

Source: Pun of the Day
Pun word: interest
Highest weight word: interest
Predicated label: 1

He couldn't decide whether to accept a job in mattress sales so he decided to sleep on it.

Source: SemEval2017
Pun word: sleep
Highest weight word: sleep
Predicated label: 1

Figure 5: Visualization of attention layers

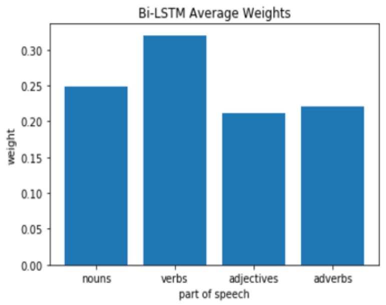

Figure 6: Average context weight of all sentences by Bi-LSTM

We apply the Bi-LSTM to capture latent semantic context for weighting part of speech which included nouns, verbs, adjectives and adverbs from forward and backward direction. Figure 6 shows Bi-LSTM model distributes weights to the four part of speeches. The weights of verbs occupy the first place, then second one is nouns, adjectives and adverbs are lower. Meanwhile, it demonstrates the importance of part of speech.

Thus, we choose two examples to illustrate weights with the four part of speeches according to context information by Bi-LSTM in Figure7. For the first example, the word "cured", as a verb, which is a pun word. It shows weights of verbs are highest allocation by Bi-LSTM. For the second example, the word "cinch", as a noun, which is a pun word. It illustrates that Bi-LSTM distributes the higher weights to nouns, which presents the importance of nouns from the context semantic information. Hence, context weights providing by Bi-LSTM are helpful of the collocation for recognizing the homographic puns.

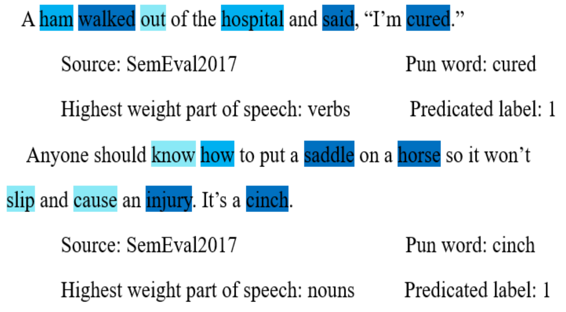

A ham walked out of the hospital and said, “I’m cured.”

Source: SemEval2017 Pun word: cured

Highest weight part of speech: verbs Predicated label: 1

Anyone should know how to put a saddle on a horse so it won’t slip and cause an injury. It’s a cinch.

Source: SemEval2017 Pun word: cinch

Highest weight part of speech: nouns Predicated label: 1

Figure 7: Visualization of Bi-LSTM

5 Conclusion and Future Work

In this study, we propose a computational model WECA combined with WordNet-Encoded word embedding and Collocation-Attention network. We extend the semantic information of word embedding by lemma and synset according to WordNet. We also apply a neural attention network, combined with Bi-LSTM, which captures the collocation of homographic puns. Experimental results show our model achieves the best performance and outperforms several baselines.

In future work, we would like to find an appropriate way in incorporating the external linguistic knowledge to improve the performance of homographic puns recognition. We also focus on automatically generating homographic puns. Those are all promising jobs we can pursue in the future.

Acknowledgements

This work is partially supported by grant from the Natural Science Foundation of China (No.61632011, 61702080, 61572102, 61602079, 61602078), the Fundamental Research Funds for the Central Universities (No.DUT18ZD102, DUT17RC(3)016).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computer Science*.

D. Chen and C. D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Conference on Empirical Methods in Natural Language Processing*, pages 740–750.

Wenliang Chen, Min Zhang, and Yue Zhang. 2015. *Distributed feature representations for dependency parsing*. IEEE Press.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3988–3994.

Alex Graves. 2012. *Long Short-Term Memory*. Springer Berlin Heidelberg.

Yu Hsiang Huang, Hen Hsen Huang, and Hsin Hsi Chen. 2017. Identification of homographic pun location for pun understanding. In *International Conference on World Wide Web Companion*, pages 797–798.

Vijayasaradhi Indurthi and Subba Reddy Oota. 2017. Fermi at semeval-2017 task 7: Detection and interpretation of homographic puns in english language. In *International Workshop on Semantic Evaluation*, pages 457–460.

Aaron Jaech, Rik Koncel-Kedziorski, and Mari Ostendorf. 2016. Phonological pun-derstanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 654–663.

Justine T. Kao, Roger Levy, and Noah D. Goodman. 2016. A computational model of linguistic humor in puns. *Cognitive Science*, 40(5):1270–1285.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2015. Ask me anything: dynamic memory networks for natural language processing. pages 1378–1387.

L. G. Lippman and M. L. Dunn. 2000. Contextual connections within puns: effects on perceived humor and memory. *Journal of General Psychology*, 127(2):185–197.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computer Science*.

Tomas Mikolov, Martin Karafit, Lukas Burget, Jan Cernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September*, pages 1045–1048.

G. A. Miller. 2002. Wordnet: A lexical database for the english language. *Contemporary Review*, 241(1):206–208.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of english puns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 719–729.

Miller, Tristan and Hempelmann, Christian and Gurevych, Iryna. 2017. Semeval-2017 task 7: Detection and interpretation of english puns. In *International Workshop on Semantic Evaluation*, pages 58–68.

Miller, Tristan and Turković, Mladen. 2016. Towards the automatic detection and identification of english puns. *The European Journal of Humour Research*, 4(1):59–75.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In *Meeting of the Association for Computational Linguistics*, pages 2049–2058.

J. H. P. Pafford. 1987. Redfern, w., puns. *Notes Queries*, (3).

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

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Parallel Distributed Processing Explorations in the Microstructure of Cognition*, 323(6088):399–421.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. 4:3104–3112.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *AAAI*.

Keiko Tanaka. 1992. The pun in advertising: A pragmatic approach. *Lingua*, 87(1-2):91–102.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2014. Grammar as a foreign language. *Eprint Arxiv*, pages 2773–2781.

Yuhuan Xiu, Man Lan, and Yuanbin Wu. 2017. Ecnu at semeval-2017 task 7: Using supervised and unsupervised methods to detect and locate english puns. In *International Workshop on Semantic Evaluation*, pages 453–456.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *Computer Science*, pages 2048–2057.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2015. Stacked attention networks for image question answering. pages 21–29.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

A Hybrid Approach to Automatic Corpus Generation for Chinese Spelling Check

Dingmin Wang[♣][*], **Yan Song**[♠], **Jing Li**[♠], **Jialong Han**[♠], **Haisong Zhang**[♠]
[♣]Tencent Inc.
dimmywang@tencent.com
[♠]Tencent AI Lab
{clksong, ameliajli, jialonghan, hansonzhang}@tencent.com

Abstract

Chinese spelling check (CSC) is a challenging yet meaningful task, which not only serves as a preprocessing in many natural language processing (NLP) applications, but also facilitates reading and understanding of running texts in peoples' daily lives. However, to utilize data-driven approaches for CSC, there is one major limitation that annotated corpora are not enough in applying algorithms and building models. In this paper, we propose a novel approach of constructing CSC corpus with automatically generated spelling errors, which are either visually or phonologically resembled characters, corresponding to the OCR- and ASR-based methods, respectively. Upon the constructed corpus, different models are trained and evaluated for CSC with respect to three standard test sets. Experimental results demonstrate the effectiveness of the corpus, therefore confirm the validity of our approach.

Sentence	Correction
我们应该认真对待这些 $\boxed{己}$ (ji2) 经发生的事	已 (yi3)
在我们班上，$\boxed{她}$ (ta1)是一个很聰明的男孩	他 (ta1)

Table 1: Two examples of Chinese sentences containing spelling errors. Spelling errors are marked in red. The first sentence contains a visually similar spelling error, i.e., 已 (yi3) is misspelled as 己 (ji2). The second sentence contains a phonologically similar spelling error, i.e., 他 (ta1) is misspelled as 她 (ta1).

1 Introduction

Spelling check is a crucial task to detect and correct human spelling errors in running texts (Yu and Li, 2014). This task is vital for NLP applications such as search engine (Martins and Silva, 2004; Gao et al., 2010) and automatic essay scoring (Burstein and Chodorow, 1999; Lonsdale and Strong-Krause, 2003), for the reason that spelling errors not only affect reading but also sometimes completely alter the meaning delivered in a text fragment. Especially, in Chinese language processing, spelling errors can be more serious since they may affect fundamental tasks such as word segmentation (Xue, 2003; Song and Xia, 2012) and part-of-speech tagging (Chang et al., 1993; Jiang et al., 2008; Sun, 2011), etc. Of all causes lead to spelling errors, a major one comes from the misuse of Chinese input methods on daily texts, e.g., emails and social media posts. Table 1 illustrates two examples of such Chinese spelling errors. The first incorrect sentence contains a misused character, 己 (ji2)[1], which has a similar shape to its corresponding correct character, i.e., 已 (yi3). In the second incorrect sentence, the boxed spelling error 她 (ta1) is phonetically identical to its corresponding correct one 他 (ta1).

Owing to the limited number of available datasets, many state-of-the-art supervised models are seldom employed in this field, which hinders the development of CSC. Currently, some mainstream approaches still focus on using unsupervised methods, i.e., language model based ones (Chen et al., 2013; Yu et al., 2014; Tseng et al., 2015; Lee et al., 2016). As a result, the development of CSC techniques are restricted and thus CSC performance is not satisfied so far (Fung et al., 2017). To enhance CSC performance, the biggest challenge is the unavailability of large scale corpora with labeled spelling errors, which is of high value for training and applying supervised models. Such issue of data absence is mainly caused by the fact that annotating spelling errors is an expensive and challenging task.

To address the data unavailability issue so that

---

[*] This work was conducted during Dingmin Wang's internship in Tencent AI Lab.

[1] We use Chinese pinyin to identify the pronunciation for each character, where the last number represents different tones (ranging from 1 to 4) in Pinyin, same below.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2517–2527
Brussels, Belgium, October 31 - November 4, 2018.

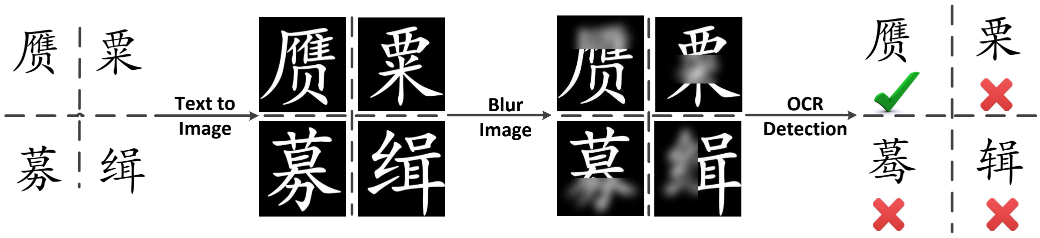

Figure 1: An example process of generating V-style errors by the OCR-based method. In the OCR detection results, except for 赝 (yan4), the other three characters, i.e., 粟 (li4), 募 (mu4), and 缉 (ji1), are incorrectly recognized as 粟 (su4), 蓦 (mo4), and 辑 (ji2), respectively. All the three incorrect characters have similar shapes with their corresponding correct references.

to facilitate CSC with data-driven approaches, in this paper, we propose a novel approach that automatically constructs Chinese corpora with labeled spelling errors. Specifically, given that Chinese spelling errors mainly result from the misuse of visually and phonologically similar characters (Chang, 1995; Liu et al., 2011; Yu and Li, 2014), we propose OCR-based and ASR-based methods to produce the aforementioned two types of misused characters. Note that, different from detecting spelling errors from incorrect sentences, our proposed approach aims at automatically generating texts with annotated spelling errors like those in Table 1. With the help of OCR- and ASR-based methods, CSC corpora are constructed with annotated visual and phonological spelling errors.

In our experiments, qualitative analysis illustrates that incorrectly recognized Chinese characters by OCR or ASR toolkits are not trivial for human to detect, while interestingly, human are likely to make such spelling errors in their daily writing. In the quantitative comparison, we cast Chinese spelling check into a sequence labeling problem and implement a supervised benchmark model, i.e., bidirectional LSTM (BiLSTM), to evaluate the performance of CSC on three standard testing datasets. Experimental results show that the BiLSTM models trained on our generated corpus yield better performance than their counterparts trained on the training dataset provided in the standard testing datasets. To further facilitating the CSC task, we construct confusion sets by collecting all incorrect variants for each character and their corresponding correct references. The effectiveness of the confusion set is confirmed in the error correction task, indicating that the constructed confusion sets are highly useful in many existing Chinese spelling check schemes (Chang, 1995; Wu et al., 2010; Dong et al., 2016).

2 Automatic Data Generation

Spelling errors in Chinese are mainly caused by the misuse of visually or phonologically similar characters (Chang, 1995; Liu et al., 2011; Yu and Li, 2014). Errors of visually similar characters (henceforth V-style errors) are due to the prominence of character pairs visually similar to each other. The reason is that, Chinese, as a hieroglyph language, consists of more than sixty thousand characters[2]. They are constructed by a limited number of radicals and components[3]. As for errors caused by the misuse of phonologically similar characters (henceforth P-style errors), we note that pronunciations of Chinese characters are usually defined by Pinyin, which consists of initials, finals, and tones[4]. According to Yang et al. (2012), there are only 398 syllables for thousands of characters in modern Chinese. As a result, there are many Chinese characters sharing similar pronunciation, which further leads to the prominence of P-style errors. In the rest of this section, we describe how we generate these two types of errors in Section 2.1 and 2.2, respectively.

2.1 OCR-based Generation

Inspired by the observation that optical character recognition (OCR) tools are likely to misidentify characters with those visually similar ones (Tong and Evans, 1996), we intentionally blur images with correct characters, and apply OCR tools on them to produce V-style spelling errors.

In detail, we use Google Tesseract (Smith, 2007) as the OCR toolkit and the generation process is illustrated in Figure 1. Given a sentence,

---

[2] `http://www.hanzizidian.com`.

[3] There are less than three hundred radicals in total. `https://en.wikipedia.org/wiki/Radical_(Chinese_characters)`

[4] `https://en.wikipedia.org/wiki/Pinyin`

as the first step, we randomly select $1 \sim 2$ character(s) from it as our target characters to be detected by Tesseract, denoted as $C_{targets}$. Specifically, except for Chinese characters, other characters like punctuations and foreign alphabets are excluded and we also filter those Chinese characters of low frequency[5] based on the statistics of the Chinese Wikipedia Corpus[6]. Second, we transfer $C_{targets}$ from text to image with 100×100 pixels, namely, each generated image has a same size. Third, we randomly blur a region[7] in the produced images using Gaussian blurring (Bradski, 2000), which aims at leading the OCR toolkit to make mistakes. Finally, we use Google Tesseract to recognize the blurred images. Once the recognized result does not match to the original one, a V-style error is generated, which is used to replace the original character in the sentence, resulting in a sentence with V-style spelling error(s). After the aforementioned steps, we obtain the spelling errors for each sentence with their correct references.

The raw texts used for OCR-based method are mainly from newspaper articles, which are crawled from People's Daily, an official newspaper website[8], of which articles are reported to undergo a strict edition process and assumed to be all correct. We divide these texts into sentences using clause-ending punctuations such as periods (。), question mark (？), and exclamation mark (！) (Chang et al., 1993). In total, we obtain $50,000$ sentences, each of which contains 8 to 85 characters, including punctuations. These sentences are then handled by the OCR-based method as we describe before, resulting in an annotated corpus containing about $40,000$ annotated sentences with $56,857$ spelling errors. Note that in our experiment, we find that the OCR toolkit can still correctly detect the characters in the produced images even we blur part of the images. This explains why the size of generated annotated sentences is smaller than that of the original sentences. We denote the dataset by D-ocr, and show its statistics in the D-ocr column in Table 3.

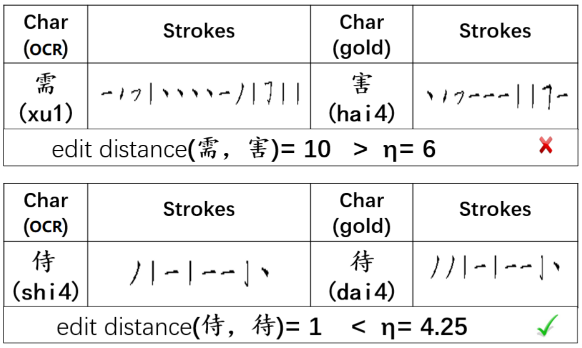

Figure 2: Two cases from the OCR-based method. Char (OCR): the character recognized by the OCR-based method; Char (gold): the corresponding correct character.

Bad Cases and the Solution Although the OCR-based method work smoothly, there are still some cases worth further investigation. By analyzing the generated spelling errors by this method, we find that in terms of shape, there exist some incorrectly recognized characters by the OCR toolkit that greatly differ from their corresponding correct characters. For example, for the blurred image containing the character 领 (ling3), the Tesseract incorrectly recognizes it as 铈 (shi4), which is totally different from 领 (ling3) in shape. Therefore, these cases should be excluded because human are less likely to make such mistakes. In solving this problem, we propose a novel approach to judge whether two characters are visually similar by calculating the edit distance based on the strokes of Chinese characters. Similar to English words consisting of alphabets, a Chinese character can be split into different strokes[9]. To this end, we obtain the strokes of Chinese character from the Online dictionary[10]. Empirically, given two Chinese characters, c_1 and c_2, we set $0.25 * (len(c_1) + len(c_2))$ as the threshold, η, where $len(c)$ denotes the number of strokes for the Chinese character c. If the edit distance of two characters is more than a threshold η, we consider them not to be similar in shape. To better clarify it, a bad case and a good case are shown in Figure 2.

---

[5] In our setting, Chinese characters occurring less than five times in the corpus are considered as low-frequency ones.

[6] https://dumps.wikimedia.org/zhwiki/

[7] A Chinese character may be misspelled to other different characters, resulting in different spelling errors. For example, the Chinese character 缉 (ji1) could be misspelled as 揖 (ji1), 辑 (ji2), or 楫 (ji1), such spelling errors could be obtained by blurring different locations of the given character, which makes it possible for the OCR toolkit to give different recognized results for a same character. According to our experiment, if we blur the entire character, we found that the incorrectly recognized results are almost the same. As a result, we could not obtain different spelling errors for a same character. Specially, the blurred location may be different, but the size of the region to be blurred is the same for all images.

[8] http://www.people.com.cn/

[9] https://en.wikipedia.org/wiki/Stroke_(CJKV_character)

[10] https://bihua.51240.com/

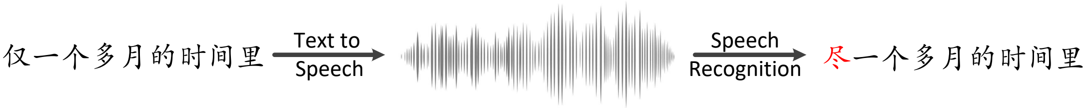

Figure 3: The full pipeline of generating P-style errors by ASR-based method. In this example, 仅 (jin3) is incorrectly recognized as 尽 (jin4) marked in red, both of which have a similar pronunciation.

Case	Gold Sentences	Incorrectly-recognized results	Type
1	而对楼市成交抑制作用最大的限购	而对面楼市成交抑制作用最大的限购	D
2	围绕亚运会进行的城市资金投入	没让亚运会进行的城市改造的资金投入	N
3	与院方协商赔偿问题	与岳风学生赔偿问题	T
4	但是不幸最终还是发生了	但是不行最终还是发生了	C

Table 2: Four incorrectly-recognized cases with different types of errors. D: Different lengths with that of the original one. N: Not a P-style error. T: Too many errors. C: Correct case to collect.

2.2 ASR-based Generation

Similar to OCR tools, automatic speech recognition (ASR) tools may also mistake characters for others with similar pronunciations (Hartley and Reich, 2005). To build an annotated corpus of P-style errors, we follow the similar inspiration with those for V-style errors and OCR tools, and adopted a pipeline as shown in Figure 3. However, given the availability of various speech recognition datasets, we employ a simpler approach. We exploit a publicly available Mandarin speech corpus, AIShell (Bu et al., 2017), which contains around 140,000 sentences with utterances[11]. We use Kaldi (Povey et al., 2011), a speech recognition toolkit, to transcribe the utterances into recognized sentences. Finally, by comparing the recognized sentences with the original ones, we can identify whether the recognition results are correct. If not, they can serve as incorrectly-recognized results and be used to build a corpus with P-style spelling errors.

Bad Cases and the Solution For generated P-style errors, we also identify some bad cases, which potentially introduce much noise. To improve the quality of the generated corpus, a solution is thus needed to remove them. Table 2 gives three types of bad cases with a good one. We describe the solution to deal with them as follows.

First, we discard all incorrectly recognized results similar to Case 1, which has different lengths comparing to the corresponding reference sentence. Second, the incorrect characters in Case 2 have totally different pronunciations with their corresponding characters in the gold sentence. Such cases do not satisfy our requirement in generating P-style errors. To this end, we obtain the pronunciation by pinyin[12] of Chinese characters from an online Chinese lexicon[13]. Then it is easy to identify whether the incorrectly-recognized characters have similar or same pronunciation with their corresponding characters in the gold sentence. Specifically, in terms of Pinyin, two characters have similar pronunciation when they have the same initials and finals but different tones, i.e., *da2* and *da1*. Third, according to Chen et al. (2011), there may have two errors per student essay on average, which reflects the fact that that each sentence will not contain more than two spelling errors on average. Therefore, we remove those incorrectly-recognized results that contains more than two incorrect characters as shown in Case 3. After the aforementioned steps, we generate a corpus with more than 7K P-style spelling errors in total. We denote it D-asr and show its statistics in the D-asr column in Table 3.

3 Evaluation

3.1 Benchmark Data

To evaluate the quality of the generated corpora, we adopt three benchmark datasets from 2013–2015 shared tasks (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015)[14] on CSC. Table 4 reports the

---

[11] Collected from 400 people from different dialect areas in China. `http://www.openslr.org/resources/33/data_aishell.tgz`

[12] Pinyin is the standard method to define the pronunciation of a Chinese character. `https://en.wikipedia.org/wiki/Pinyin`

[13] `http://www.zdic.net`

[14] These datasets are written in traditional Chinese, so to keep the consistency with our generated corpus of simplified

	D-ocr	D-asr	D
Sentences #	40,000	40,000	80,000
Characters #	915,949	716,509	1,632,458
Errors #	56,857	75,667	132,524

Table 3: Statistics of our generated corpus. D-ocr denotes the corpus generated by OCR-based method, D-asr denotes the corpus generated by ASR-based method and D is the combination of D-ocr and D-asr.

statistics of the three standard datasets, including training and test parts. Consider that the quality of the training dataset has a significant impact on models' performance on the test datasets. Thus in this paper, we propose a metric, $\mathrm{C}_{\mathrm{train:test}}$, to measure the correlation degree between the training and test dataset by calculating the number of spelling errors that occur in them:

$$C_{train:test} = \frac{\left|E_{train} \cap E_{test}\right|}{\left|E_{test}\right|} \tag{1}$$

where $\mathrm{E}_{\mathrm{train}}$ and $\mathrm{E}_{\mathrm{test}}$ refer to the set of spelling errors in the training dataset and the test dataset, respectively. Each element in the set is a pair containing a correct character and a misspelled character, e.g., (撼 (han4), 憾 (han4)). Table 5 illustrates that, for three different testing datasets, the entire generated corpus D achieves 74.1%, 80.6% and 84.2% on $\mathrm{C}_{\mathrm{train:test}}$, respectively, which are much higher than that of Trn_{13}, Trn_{14} and Trn_{15}. This difference may denote the validity of the generated corpus, with adequate spelling errors.

3.2 Qualitative Analysis

Setup To evaluate whether the generated corpus contains errors that are easily made by human, we randomly select 300 sentences from it for manual evaluation, with 150 from D-ocr and 150 from D-asr. Three native Chinese speakers, who are college students, are invited to read and annotate errors in these sentences. Then, we analyze the annotated results by three college students from two levels: sentence-level and error-level. On the sentence-level, we consider a sentence to be correctly annotated only if all errors in the sentence are recognized. On the error-level, we calculate the percentage of the number of correctly annotated errors out of the total number of errors.

		Error #	Char #	Sent #
2013[15]	Trn_{13}	324	17,611	350
	Tst_{13}	966	75,328	1,000
2014[16]	Trn_{14}	5,224	330,656	6,527
	Tst_{14}	510	54,176	1,063
2015[17]	Trn_{15}	3,101	95,114	3,174
	Tst_{15}	531	34811	1,100

Table 4: Statistics of three standard datasets. Error # denotes the number of spelling errors, Char # represents the number of Chinese character and Sent # refers to the number of sentences.

Train:Test	C (%)	Train:Test	C (%)
Trn_{13} : Tst_{13}	16.2	D : Tst_{13}	74.1
Trn_{14} : Tst_{14}	53.9	D : Tst_{14}	80.6
Trn_{15} : Tst_{15}	46.7	D : Tst_{15}	84.2

Table 5: Correlation results of Trn_{13}, Trn_{14}, Trn_{15} and D with Tst_{13}, Tst_{14}, Tst_{15}.

Results Table 6 shows the information of 300 sentences and the annotation results on them. The average recall in the table illustrates that three students have a higher recognition rate for errors from D-asr than that from D-ocr, which, to some extent, indicates that P-style errors are easier to be detected than V-style ones. Besides, we observe that three volunteers fail to identify around 36.9% errors on average, which may indicate that our generated sentences include some challenging errors which are likely to be made by human. Such errors are valuable for CSC since they are potential real cases in people's writing or typing.

Case Study To qualitatively analyze why some spelling errors are not detected by human, we conduct a case study on an example sentence containing some spelling errors that are not found by the three students. The sentence is 政企部分是一种痼疾 (translation: politics and industry parts are a kind of a chronic disease), in which the third character 部 (bu4) is a spelling error and should be corrected as 不 (bu4). In this example, 部分 (translation: a part of) and 不分 (translation: equally treat) are two highly common Chinese words and

---

Chinese, we convert these datasets from tradition Chinese to simplified Chinese using the OpenCC, an Open-source Chinese Converter. https://github.com/BYVoid/OpenCC

[15] http://ir.itc.ntnu.edu.tw/lre/sighan7csc.html

[16] http://ir.itc.ntnu.edu.tw/lre/clp14csc.html

[17] http://ir.itc.ntnu.edu.tw/lre/sighan8csc.html

	Stu1	Stu2	Stu3	R
S-ocr	**84**/150	**100**/150	**75**/150	57.3
E-ocr	**104**/170	**121**/170	**100**/170	72.0
S-asr	**95**/150	**79**/150	**106**/150	62.0
E-asr	**341**/393	**179**/393	**356**/393	74.3

Table 6: Human evaluation results. S-ocr refer to 150 sentences from the D-ocr and E-ocr represents the errors in S-ocr, similar for S-asr and E-asr. R denotes the average recall of three students. Numbers in bold denotes the correctly-annotated results by students.

easy to be considered as correct. However, considering the preceding word 政企 (translation: politics and industry) and the subsequent words 是一种痼疾 (translation: a kind of chronic disease), we can see that 部分 does not fit the current context and should be corrected as 不分. This case study confirms that our generated corpus contains some spelling errors like human made ones in their writing or typing, which further demonstrates its quality and the effectiveness of our approach.

3.3 Quantitative Comparison

3.3.1 Chinese Spelling Error Detection

In this section, we evaluate the quality of our generated corpus through Chinese spelling detection. We firstly explore how different proportions of P-style and V-style errors affect the quality of the corpus. Then we compare the detection performance of the generated corpus with that of training datasets provided in the three shared tasks.

Setup We cast Chinese spelling error detection into a sequence tagging problem on characters, in which the correct and incorrect characters are tagged as 1 and 0, respectively[18]. We then implement a supervised sequence tagging model, i.e., bidirectional LSTM (BiLSTM) (Graves and Schmidhuber, 2005), as our baseline to evaluate the quality of different corpus. The hidden size of the BiLSTM is set to 150 and the other hyper-parameters are tuned on a development set consisting of 10% randomly selected sentences from the training data. We minimize categorical cross-entropy loss for the model, with RMSprop (Graves, 2013) as the optimizer.

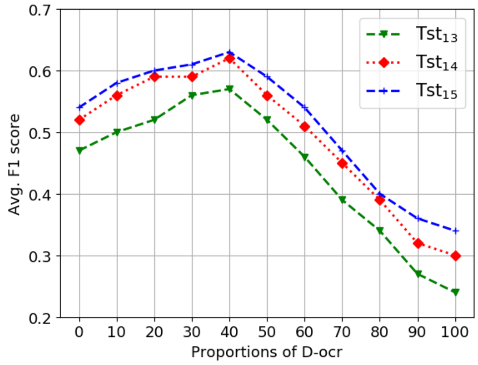

Figure 4: F1 scores achieved by different proportions of D-ocr and D-asr on three testing dataset (Tst_{13},Tst_{14} and Tst_{15}). The total size of the corpus is 40k, the percentage of which the x-axis represents from D-ocr.

Results The performance of BiLSTM trained on different proportions of D-ocr and D-asr aims at exploring the quality of the generated corpus influenced by the distribution of P-style and V-style spelling errors. Figure 4 shows that when the size of training dataset is fixed (=40k), different proportions of P-style and V-style errors achieve different F1 scores, denoting that the proportion of P-style and V-style spelling errors affects the quality of the generated corpus. Specifically, it is observed that a 4:6 proportion of D-ocr and D-asr achieves the best performance on three testing datasets when compared with other proportion ratios. In addition, for the two special proportions (0% and 100%), it is seen that with the same size of corpus, the BiLSTM model trained on D-asr achieves better performance than that on D-ocr, which indicates that P-style spelling error contributes more to the quality of the corpus. This experimental result complies with the previous conclusion (Liu et al., 2009, 2011) that most of spelling errors are related to the pronunciations.

Furthermore, to better illustrate the quality of the generated corpus, we compare it with some training datasets, which are manually annotated (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015). According to previous analyses on the experimental results shown in Figure 4, we choose the 4:6 proportion of P-style and V-style spelling errors to construct the training data in the following experiments. Specifically, we build five different sizes of datasets: D-10k[19], D-20k, D-30k, D-40k and D-50k, which are extracted from

---

[18] The detection process can be considered as a problem of binary classification.

[19] D-10k denotes the corpus contains 10k sentences, similar for D-20k, D-30k, D-40k and D-50k.

	Tst_{13}			Tst_{14}			Tst_{15}		
	P	R	F1	P	R	F1	P	R	F1
Trn	24.4	27.3	25.8	49.8	51.5	50.6	40.1	43.2	41.6
D-10k	33.3	39.6	36.1	31.1	35.1	32.9	31.0	37.0	33.7
D-20k	41.1	50.2	45.2	41.1	50.2	45.2	43.0	54.9	48.2
D-30k	47.2	59.1	52.5	40.9	48.0	44.2	50.3	62.3	55.7
D-40k	53.4	65.0	58.6	**52.3**	64.3	57.7	**56.6**	66.5	61.2
D-50k	**54.0**	**69.3**	**60.7**	51.9	**66.2**	**58.2**	**56.6**	**69.4**	**62.3**

Table 7: The performance of Chinese spelling error detection with BiLSTM on Tst_{13},Tst_{14},Tst_{15} (%). Best results are in **bold**. Trn represents the training dataset provided in the corresponding shared task, e.g., Trn denotes Trn_{13} in Tst_{13}.

D-ocr and D-asr following the proportion of 4:6. Then, we train the BiLSTM model on these training datasets and evaluate error detection performance on Tst_{13}, Tst_{14} and Tst_{15}. Table 7 shows the detection performance on three different testing datasets. We have the following observations.

The size of training dataset is important for the model training. For Tst_{13}, D-10k achieves a better F1 score than Trn_{13}. A major reason may be the size of Trn_{13} (=350, see in Table 3), which is much smaller than the testing dataset. In this situation, the model can not learn enough information, resulting in being unable to detect unseen spelling errors. Besides, we can see that the detection performance shows a stable improvement as the size of our generated corpus is continuously enlarged. Therefore, for data-driven approaches, it is of great importance to train our model with enough instances having different spelling errors.

The precision may be compromised if the training dataset contains too many "noisy" spelling errors From Table 7, although the overall performance (F1 score) keeps improving as the size of our generated corpus increases, the precision and the recall demonstrate different changing trends. It is observed that as the size of training dataset increases, the model achieves a better performance in terms of the recall. An possible reason is that with more instances containing different spelling error including in the training dataset, the number of unseen spelling error in the testing dataset is reduced, thus facilitating the model to detect more spelling errors. However, the improvement of the precision is not so obvious as that of the recall. Specifically, in Tst_{14} and Tst_{15}, D-50k does not achieve a higher precision than D-40k. A possible explanation is that with a larger training dataset containing more spelling error instances, it may lead the model to misidentify some more correct characters, resulting in a lower precision.

Compared with the limited training dataset manually annotated by human, our generated large-scale corpus can achieves a better performance From Table 7, we can see that with a certain size of our generated corpus, it can train a model that achieve better detection performance than with the manually annotated datasets provided in the corresponding shared tasks. To some extent, this demonstrates the effectiveness of our generated corpus, thus confirms the validity of our approach.

3.3.2 Chinese Spelling Error Correction

Once the Chinese spelling errors are detected, we test on correcting them.[20] from Section 3.3.1.

Following previous studies (Chang, 1995; Huang et al., 2007; Wu et al., 2010; Chen et al., 2013; Dong et al., 2016), we adopt confusion sets[21] and a language model to handle the taks for Chinese spelling error correction. In particular, by collecting all incorrect variants for each correct character, we construct a confusion set for all involved correct characters, denoted as Ours. In addition, to illustrate its effectiveness, we compare it with two publicly available confusion sets (Liu et al., 2009), Con1 and Con2. Specifically, Con1 consists of SC (similar Cangjie), SSST (same-sound-same-tone) and SSDT (same-sound-different-tone), while Con2 consists of all sets, SC, SSST, SSDT, MSST (similar-sound-

---

[20] In particular, we choose the best detected results achieved by D-50k as shown in Table 7.

[21] A confusion set refers to a set that contains confusion characters for a given character. For example, a Chinese character 快 (kuai4) may have a confusion set, {筷 (kuai4),块 (kuai4),怏 (yang1),$\cdots$ }, each of which has a similar pronunciation or a similar shape with 快 (kuai4).

Metrics	Tst_{13}			Tst_{14}			Tst_{15}		
	Con1	Con2	Ours	Con1	Con2	Ours	Con1	Con2	Ours
F1 (%)	47.6	**52.1**	50.3	52.6	**56.1**	53.0	55.6	**57.1**	56.3
Time(s)	290.3	679.9	**101.2**	310.2	792.1	**132.4**	267.6	622.5	**119.2**

Table 8: Error correction results on Tst_{13}, Tst_{14}, Tst_{15}, using Con1, Con2 and Ours, respectively.

Name	# of Char	Min.	Max.	Avg.
Ours	4,676	1	53	5.6
Con1	5,207	8	196	50.6
Con2	5,207	21	323	86.0

Table 9: Statistics of different confusions sets.

same-tone) and MSDT (similar-sound-different-tone). Table 9 shows the statistics information of the three confusion sets.

Setup Similar to Dong et al. (2016), we adopt a tri-gram language model to calculate the probability of a given sentence. Based on the detected results by the sequence labeling models, we choose as the error correction the character from the corresponding confusion set that has the highest probability. For a given sentence containing n words, $W = w_1, w_2, \cdots, w_n$, where w_i represents the i^{th} character in the sentence, E is a set containing the indexes of detected incorrect characters, $W(w_i, c)$ denotes the new generated sentence after the i^{th} character is replaced with c. The process of error correction can be formulated as follows:

$$\forall i \in E: \quad \underset{c \in C(w_i)}{\arg\max} \quad P(W(w_i, c)) \tag{2}$$

where $P(W)$ is the probability of a sentence W approximated by the product of a series of conditional probabilities as described in (Jelinek, 1997) and $C(w_i)$ refers to the confusion set of w_i, one of which with the maximum conditional probability is selected as the correction character.[22]

Results Table 9 shows the running time[23] and the F1 scores achieved by different confusion sets based on a tri-gram language model. We can observe that our constructed confusion sets achieve a better correction performance than that of Con1 while a little lower than Con2. However, from Table 9, we can see that Con2 has a much larger size of confusion characters than Ours; and for the same testing, Con2 needs much more time to finish the task while Ours always uses the least time. These observations indicate that our constructed confusions sets are more efficient in containing fewer redundant confusion characters that seldom serve as correction characters.

Error Analysis We conduct an error analysis on two types of incorrect cases, namely, the false positive and the false negative case, which affect the precision and recall in CSC, respectively.

For the false positive case, we find that one common issue is that for some fixed usages, such as idioms, phrases, and poems, our model often gives incorrect results. For example, in 风雨送春归 (translation: wind and rain escort Spring's departure), a line of a Chinese poem, 送 is incorrectly recognized as an irrelevant character. By checking the annotated corpus generated by the proposed methods, we observe that in most cases, 迎春 is a more common match, and 送 is annotated as an spelling error when it co-occurs with 春 in the generated corpus. To improve the precision, a possible approach to handle such cases is utilizing some external knowledge, such as building a collection of special Chinese usages.

For the false negative case, taking 想想健康，你就会知道应该要禁烟了 (translation: you will realize that you should give up smoking when your consider of your health) as an example, in which 禁 should be corrected as 戒. However, since 戒 and 禁 are neither visually nor phonologically similar, the proposed corpus generation approach is unable to construct such spelling errors, so it is understandable that the trained model can not detect such spelling errors. In addition, on the word-level, 禁烟 (translation: forbid smoking) and 戒烟 (translation: give up smoking) are two related common Chinese words; it needs to incorporate more context in order to improve the recall performance. Similar to our study on character-based corpus generation, one potential solution is to construct a word-level annotated corpus in order to better detect such spelling errors.

---

[22] If more than one character from the confusion set have the same maximum probability, we randomly select one of them as the correction character.

[23] We run the experiments on a computer with Intel Core i5-7500 CPU.

4 Related Work

In the line of research on spelling error detection and correction, most previous efforts focus on designing different models to improve the performance of CSC (Chang, 1995; Huang et al., 2007, 2008; Chang et al., 2013). Different from them, this work contributes to the generation of training datasets, which are important resources and can be used for improving many existing CSC models. Currently, the limited training datasets have set a high barrier for many data-driven approaches (Wang et al., 2013; Wang and Liao, 2015; Zheng et al., 2016). To the best of our knowledge, up to date, there is no large quantities of annotated data sets commonly available for CSC.

Some previous work (Liu et al., 2009; Chang et al., 2013) pointed out that visually and phonologically similar characters are major contributing factors for errors in Chinese texts, where the number of phonologically similar spelling errors is about two times than that of visually similar spelling errors. Vision- and speech-related technologies are then adopted in our approach. As a technology to extract text information from images, optical character recognition recognizes the shapes and assigns characters. According to Nagy (1988); McBride-Chang et al. (2003), incorrectly recognized results are mainly due to the visual similarities among some different Chinese characters. On the other side, automatic speech recognition is an acoustics-based recognition process for handling audio stream, where phonologically similar characters are usually confused (Kaki et al., 1998; Voll et al., 2008; Braho et al., 2014).

5 Conclusion and Future Work

In this paper, we proposed a hybrid approach to automatic generating Chinese corpus for spelling check with labeled spelling errors. Specifically, OCR- and ASR-based methods were used to generate labeled spelling errors by replacing visually and phonologically resembled characters. Human evaluation confirmed that our proposed method can produce common errors that are likely to be made by human and such errors can serve as effective annotated spelling errors for CSC. In our experiment, a neural tagging model was trained on the generated corpus and the results tested on benchmark datasets confirmed the quality of the corpus, which further demonstrated the effectiveness of our corpus generation approach for CSC.

The large scale annotated dataset generated by the proposed approach can potentially serve as useful resources in helping improving the performance of data-driven models for CSC, because the availability of large scale annotated data is the first vital step before applying any algorithms or models. In this paper, we do not put too many efforts into model design for CSC, which we leave as potential future work. To facilitate related research in the community and benefit other researchers, we make our code and data in this work publicly available on: `https://github.com/wdimmy/Automatic-Corpus-Generation`.

Acknowledgements

The authors want to express special thanks to Xixin Wu for his suggestions and help in the experiment of ASR. Besides, the authors would like to thank Li Zhong, Shuming Shi, Garbriel Fung, Kam-Fai Wong, and three anonymous reviewers for their help and insightful comments on various aspects of this work.

References

Gary Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123.

Keith P Braho, Jeffrey P Pike, and Lori A Pike. 2014. Methods And Systems for Identifying Errors in A Speech Recognition System. US Patent 8,868,421.

Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. 2017. AIShell-1: An Open-source Mandarin Speech Corpus and A Speech Recognition Baseline. *arXiv preprint arXiv:1709.05522*.

Jill Burstein and Martin Chodorow. 1999. Automated Essay Scoring for Nonnative English Speakers. In *Proceedings of a Symposium on Computer Mediated Language Assessment and Evaluation in Natural Language Processing*, pages 68–75. Association for Computational Linguistics.

Chao-Huang Chang. 1995. A New Approach for Automatic Chinese Spelling Correction. In *Proceedings of Natural Language Processing Pacific Rim Symposium*, volume 95, pages 278–283.

Chao-Huang Chang et al. 1993. HMM-based Part-of-speech Tagging for Chinese Corpora. *Very Large Corpora: Academic and Industrial Perspective*.

Tao-Hsing Chang, Hsueh-Chih Chen, Yuen-Hsien Tseng, and Jian-Liang Zheng. 2013. Automatic Detection and Correction for Chinese Misspelled Words Using Phonological and Orthographic Similarities. In *Proceedings of the Seventh SIGHAN*

Workshop on Chinese Language Processing, pages 97–101.

Kuan-Yu Chen, Hung-Shin Lee, Chung-Han Lee, Hsin-Min Wang, and Hsin-Hsi Chen. 2013. A study of Language Modeling for Chinese Spelling Check. In *Proceedings of the 7th SIGHAN Workshop on Chinese Language Processing (SIGHAN-13)*, pages 79–83.

Yong-Zhi Chen, Shih-Hung Wu, Ping-Che Yang, Tsun Ku, and Gwo-Dong Chen. 2011. Improve the Detection of Improperly Used Chinese Characters in Students' Essays with Error Model. *International Journal of Continuing Engineering Education and Life Long Learning*, 21(1):103–116.

Shichao Dong, Gabriel Pui Cheong Fung, Binyang Li, Baolin Peng, Ming Liao, Jia Zhu, and Kam-Fai Wong. 2016. ACE: Automatic Colloquialism, Typographical and Orthographic Errors Detection for Chinese Language. In *COLING (Demos)*, pages 194–197.

Gabriel Fung, Maxime Debosschere, Dingmin Wang, Bo Li, Jia Zhu, and Kam-Fai Wong. 2017. NLPTEA 2017 Shared Task–Chinese Spelling Check. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 29–34.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A Large Scale Ranker-based System for Search Query Spelling Correction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 358–366. Association for Computational Linguistics.

Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *arXiv pre-print*, abs/1308.0850.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, 18(5):602–610.

Matthew W Hartley and David E Reich. 2005. Method and System for Speech Recognition Using Phonetically Similar Word Alternatives. US Patent 6,910,012.

Chuen-Min Huang, Mei-Chen Wu, and Ching-Che Chang. 2007. Error Detection and Correction Based on Chinese Phonemic Alphabet in Chinese Text. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 463–476. Springer.

Chuen-Min Huang, Mei-Chen Wu, and Ching-Che Chang. 2008. Error Detection and Correction Based on Chinese Phonemic Alphabet in Chinese Text. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 16(supp01):89–105.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT press.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-speech Tagging. *Proceedings of ACL-08: HLT*, pages 897–904.

Satoshi Kaki, Eiichiro Sumita, and Hitoshi Iida. 1998. A Method for Correcting Errors in Speech Recognition Using the Statistical Features of Character Co-occurrence. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 653–657. Association for Computational Linguistics.

Lung-Hao Lee, RAO Gaoqi, Liang-Chih Yu, XUN Endong, Baolin Zhang, and Li-Ping Chang. 2016. Overview of nlp-tea 2016 Shared task for Chinese Grammatical Error Diagnosis. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 40–48.

C-L Liu, M-H Lai, K-W Tien, Y-H Chuang, S-H Wu, and C-Y Lee. 2011. Visually and Phonologically Similar Characters in Incorrect Chinese Words: Analyses, Identification, and Applications. *ACM Transactions on Asian Language Information Processing (TALIP)*, 10(2):10.

Chao-Lin Liu, Kan-Wen Tien, Min-Hua Lai, Yi-Hsuan Chuang, and Shih-Hung Wu. 2009. Phonological and Logographic Influences on Errors in Written Chinese Words. In *Proceedings of the 7th Workshop on Asian Language Resources*, pages 84–91. Association for Computational Linguistics.

Deryle Lonsdale and Diane Strong-Krause. 2003. Automated Rating of ESL Essays. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2*, pages 61–67. Association for Computational Linguistics.

Bruno Martins and Mário J Silva. 2004. Spelling correction for search engine queries. In *Advances in Natural Language Processing*, pages 372–383. Springer.

Catherine McBride-Chang, Hua Shu, Aibao Zhou, Chun Pong Wat, and Richard K Wagner. 2003. Morphological Awareness Uniquely Predicts Young Children's Chinese Character Recognition. *Journal of educational psychology*, 95(4):743.

G Nagy. 1988. Chinese Character Recognition: A Twenty-five-year Retrospective. In *Pattern Recognition, 1988., 9th International Conference on*, pages 163–167. IEEE.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The Kaldi Speech Recognition Toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, EPFL-CONF-192584. IEEE Signal Processing Society.

Ray Smith. 2007. An Overview of the Tesseract OCR Engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 629–633. IEEE.

Yan Song and Fei Xia. 2012. Using a Goodness Measurement for Domain Adaptation: A Case Study on Chinese Word Segmentation. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3853–3860, Istanbul, Turkey.

Weiwei Sun. 2011. A Stacked Sub-Word Model for Joint Chinese Word Segmentation and Part-of-speech Tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1385–1394. Association for Computational Linguistics.

Xiang Tong and David A Evans. 1996. A Statistical Approach to Automatic OCR Error Correction in Context. In *Fourth Workshop on Very Large Corpora*.

Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. 2015. Introduction to Sighan 2015 Bake-off for Chinese Spelling Check. *ACL-IJCNLP 2015*, page 32.

Kimberly Voll, Stella Atkins, and Bruce Forster. 2008. Improving the Utility of Speech Recognition Through Error Detection. *Journal of digital imaging*, 21(4):371.

Yih-Ru Wang and Yuan-Fu Liao. 2015. Word vector/conditional random field-based chinese spelling error detection for sighan-2015 evaluation. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 46–49.

Yih-Ru Wang, Yuan-Fu Liao, Yeh-Kuang Wu, and Liang-Chun Chang. 2013. Conditional Random Field-based Parser and Language Model for Traditional Chinese Spelling Checker. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 69–73.

Shih-Hung Wu, Yong-Zhi Chen, Ping-Che Yang, Tsun Ku, and Chao-Lin Liu. 2010. Reducing the False Alarm Rate of Chinese Character Error Detection and Correction. In *CIPS-SIGHAN Joint Conference on Chinese Language Processing*.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese Spelling Check Evaluation at SIGHAN Bake-off 2013.

Nianwen Xue. 2003. Chinese Word Segmentation as Character Tagging. *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing*, 8(1):29–48.

Shaohua Yang, Hai Zhao, Xiaolin Wang, and Bao-liang Lu. 2012. Spell Checking for Chinese. In *LREC*, pages 730–736.

Junjie Yu and Zhenghua Li. 2014. Chinese Spelling Error Detection and Correction Based on Language Model, Pronunciation, and shape. *CLP 2014*, page 220.

Liang-Chih Yu, Lung-Hao Lee, Yuen-Hsien Tseng, Hsin-Hsi Chen, et al. 2014. Overview of SIGHAN 2014 Bake-off for Chinese Spelling Check. In *Proceedings of the 3rd CIPSSIGHAN Joint Conference on Chinese Language Processing (CLP'14)*, pages 126–132.

Bo Zheng, Wanxiang Che, Jiang Guo, and Ting Liu. 2016. Chinese Grammatical Error Diagnosis with Long Short-Term Memory Networks. *NLPTEA 2016*, page 49.

Neural Quality Estimation of Grammatical Error Correction

Shamil Chollampatt[1,2] and **Hwee Tou Ng**[1,2]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]Department of Computer Science, School of Computing
National University of Singapore
shamil@u.nus.edu, nght@comp.nus.edu.sg

Abstract

Grammatical error correction (GEC) systems deployed in language learning environments are expected to accurately correct errors in learners' writing. However, in practice, they often produce spurious corrections and fail to correct many errors, thereby misleading learners. This necessitates the estimation of the quality of output sentences produced by GEC systems so that instructors can selectively intervene and re-correct the sentences which are poorly corrected by the system and ensure that learners get accurate feedback. We propose the first neural approach to automatic quality estimation of GEC output sentences that does not employ any hand-crafted features. Our system is trained in a supervised manner on learner sentences and corresponding GEC system outputs with quality score labels computed using human-annotated references. Our neural quality estimation models for GEC show significant improvements over a strong feature-based baseline. We also show that a state-of-the-art GEC system can be improved when quality scores are used as features for re-ranking the N-best candidates.

1 Introduction

The task of automatically correcting various kinds of errors in written text, termed as grammatical error correction (GEC), is primarily aimed at assisting language learning and providing corrective feedback to second-language learners. GEC systems are expected to give precise corrections and have the ability to correct most learner mistakes. In reality, however, this is not the case. State-of-the-art GEC systems (Junczys-Dowmunt et al., 2018; Grundkiewicz and Junczys-Dowmunt, 2018; Chollampatt and Ng, 2018) have a precision below 70% and a recall around 40% when evaluated on benchmark datasets. This level of performance is impressive since GEC is a difficult task given the diversity and complexity of language errors. However, in real-world use cases such as language learning, erroneous feedback from automatic GEC systems can potentially mislead language learners. To prevent this, the instructor can intervene and re-correct the system's corrections when necessary, before they are provided as feedback to learners. Having quality estimates for the system's output sentences can help instructors to decide whether to check and fix the system's corrections (for higher quality corrections) or to ignore the system's corrections altogether and re-correct the original learner-written sentences (for lower quality ones) instead. This can significantly make the process of post-editing easier and faster. Such quality estimates can also directly help end users — the language learners — to decide on the extent to which the system's corrections can be trusted and seek assistance from instructors and other sources to get better corrective feedback if needed. In this paper, we propose a neural approach to automatic quality estimation of GEC output.

Quality of language output applications can refer to several aspects such as fluency, grammaticality, adequacy, and post-editing effort. While reference-based metrics such as MaxMatch or M^2 (Dahlmeier and Ng, 2012) and GLEU (Napoles et al., 2016a, 2015) are used to evaluate GEC systems with human-annotated references, a few reference-less GEC metrics have been proposed to evaluate fluency, grammaticality, and adequacy (Napoles et al., 2016b; Asano et al., 2017; Choshen and Abend, 2018b). However, there has been no work in GEC addressing the estimation of post-editing effort. Also, to our knowledge, this is the first supervised approach to quality estimation (QE) for GEC system outputs, similar to the supervised QE task in machine translation (MT) (Specia et al., 2009). Our neural models for GEC QE are

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2528–2539
Brussels, Belgium, October 31 - November 4, 2018.

based on variants of the predictor-estimator architecture (Kim et al., 2017a), where knowledge from a pre-trained network for a word-prediction task is transferred to another network that estimates the quality score. Apart from re-implementing the recurrent predictor-estimator models, we propose convolutional variants that are faster to train and run. We release our source code[1] publicly.

In summary, the contributions of this paper are: (1) we propose the first supervised approach to QE of GEC system outputs, (2) we present neural QE models that outperform a strong feature-based baseline for estimating post-editing effort and an automatic GEC evaluation metric, (3) we propose new convolutional neural architectures for QE that can be potentially utilized for QE tasks in other language applications, and (4) we show that the performance of a state-of-the-art GEC system can be improved by adding QE scores as features in re-ranking the N-best candidates.

2 Related Work

The task of quality estimation became popular in machine translation (MT) through the studies by Blatz et al. (2004) and Specia et al. (2009). Much of the later work in QE of MT was through the shared tasks in Workshop on Machine Translation (WMT) campaigns (Bojar et al., 2016b) from 2012 onwards (Callison-Burch et al., 2012). Supervised methods of quality assessment have been applied to other natural language processing tasks such as text simplification (Štajner et al., 2016), language generation (Dušek et al., 2017), and in assisting interpreters (Stewart et al., 2018).

In the context of GEC, Heilman et al. (2014) attempted to predict grammaticality of learner sentences using regression with a variety of linguistic features such as the number of misspellings, language model scores, etc. They use a dataset of learner sentences manually annotated with subjective scores of grammaticality. However, their method was to assess learner writing and not for system evaluation. To evaluate GEC systems, Napoles et al. (2015) developed reference-less metrics known as *grammaticality-based metrics* or GBMs. GBM scores are based on the number of errors detected using third-party tools or determined by a grammaticality prediction model (Heilman et al., 2014). Their method ignores the source sentence completely and judges the system outputs independently for grammaticality. Asano et al. (2017) improved their method to account for fluency as well as faithfulness to the source sentence. Choshen and Abend (2018b) provide another measurement for meaning preservation using a semantic annotation scheme. Contrary to prior work in GEC reference-less evaluation, our work is aimed at estimating post-editing effort in terms of translation error rate (Snover et al., 2006) and an automatic evaluation metric, Max-Match (Dahlmeier and Ng, 2012), in a supervised approach. We propose variants of the predictor-estimator architecture (Kim and Lee, 2016b; Kim et al., 2017a) and compare them to a competitive feature-based baseline, QuEst (Specia et al., 2013, 2015) that has been successfully used for a number of language pairs in MT QE and for other applications (Stewart et al., 2018). It has also been the baseline for WMT QE tasks.

3 Quality Estimation of GEC

Quality estimation (QE) of GEC can be defined as the task of estimating a quality score $\hat{q}$ given a source sentence S and its corresponding GEC system-corrected hypothesis, H. We formulate the GEC QE task as a supervised regression task to predict the quality scores, following the MT QE approach (Specia et al., 2009). The score is estimated using a trained regression model f with parameters θ, such that $\hat{q} = f(S, H, \theta)$. The model f is trained and evaluated by utilizing a set of learner-written sentences and their corresponding corrected hypotheses produced by a "black-box" GEC system, i.e., neither the GEC system's model scores nor internal states will be known to the QE system. The gold-standard quality scores are obtained by comparing the system-corrected sentences and human-corrected references. We are primarily interested in estimating the post-editing effort for correcting the output sentences. Similar to MT QE, we assess GEC post-editing effort scores using human-targeted translation error rate or HTER (Snover et al., 2006). HTER is the minimum number of edit operations (insertions, deletions, substitutions, or shifts of word sequences) needed to transform the hypothesis sentence to the reference sentence, normalized by the length of the reference. A low HTER score indicates less post-editing effort.

$$\text{HTER} = \frac{\text{number of edits}}{\text{number of reference tokens}}$$

---

[1] `https://github.com/nusnlp/neuqe`

In MT, the reference translations for HTER are *targeted*, i.e., they are created by post-editing system translated sentences. However, in GEC, high-quality datasets annotated by experts with minimal edits are available (Dahlmeier et al., 2013; Yannakoudakis et al., 2011) and GEC systems are typically trained to make minimal changes to input sentences. Hence, the actual human annotated references can be substituted for post-edited references of output sentences[2]. We also experiment with estimating an automatic GEC evaluation metric, MaxMatch or M^2 (Dahlmeier and Ng, 2012) as the quality score. M^2 is the most widely used GEC evaluation metric that computes the F$_{0.5}$-score of phrase-level edits made by a system.

4 Neural Quality Estimation Model

Our neural quality estimation (NQE) model uses the predictor-estimator architecture (Kim et al., 2017a) to model the regression function f. Recurrent variants of the model have achieved the first and second places for WMT 2017 and 2016 sentence-level QE tasks, respectively (Kim and Lee, 2016a; Kim et al., 2017b). The key idea behind the model is to employ a preliminary *predictor* neural network that is trained for the "word prediction" task, i.e., to predict the probabilities of the words in the target sentence given the source sentence and the remaining target context (the words in the target sentence other than the predicted word). The predictor networks are trained using large parallel texts (potentially erroneous learner sentences and their corresponding human-corrected sentences). The knowledge from the predictor network is transferred to the *estimator* network that is trained to estimate the quality score $\hat{q}$ given the source sentence S and its corresponding system hypothesis H. Specifically, a pre-trained predictor network takes as input S and H (in place of the target sentence) and predicts probability scores for words in H. The intuition is that hypothesis words that are likely to match the reference sentence will be assigned higher probabilities. The hidden representations from the predictor network, called *quality vectors*, having information about the quality of the hypothesis words, become the input to the estimator network that estimates the quality score. The estimator networks are trained using learner sentences and their corresponding GEC system-corrected hypotheses. The gold quality score is obtained by comparing a hypothesis and the corresponding human-corrected reference. We use high-quality datasets annotated minimally to train the estimator networks. Apart from re-implementing the recurrent neural network (RNN)-based predictor-estimator model, we build fully convolutional neural network (CNN)-based variants for both the predictor and the estimator.

4.1 Predictor Network

The inputs to the predictor network are the source sentence S with source tokens $s_1, ..., s_m$ and its corresponding target sentence T with target tokens $t_1, ..., t_n$. The predictor networks are trained to predict each target token t_j given the source sentence S and the remaining target tokens excluding the predicted target token, denoted by T_{-j}. The output of the predictor network is a softmax probability score normalized across the target vocabulary, V_t:

$$p(t|S, T_{-j}) = \frac{\exp(o_{j,t})}{\sum_{t' \in V_t} \exp(o_{j,t'})}$$

where $o_{j,t}$ is the node corresponding to the word t in the predictor's output vectors $\mathbf{o}_j \in \mathbb{R}^{|V_t|}$, when t_j is predicted (see Figure 1). Predictor networks estimate the output probability by an architecture that extends the encoder-decoder neural network for sequence-to-sequence translation (Bahdanau et al., 2015). Traditional encoder-decoder models use a bidirectional RNN on the source sentence and a forward RNN on the target side to capture the target context preceding the predicted target word. The predictor network additionally employs another backward RNN in the decoder to capture the target context following the predicted word (Kim and Lee, 2016b). In the case of QE, the entire target sentence is available as input, unlike the case of sequence-to-sequence translation. Predictor networks are originally based on RNNs with gated recurrent units or GRUs (Cho et al., 2014) and a soft-attention mechanism similar to that in (Bahdanau et al., 2015). We use separate attention mechanisms for the forward and backward RNNs of the decoder in our implementation of the predictor network.

Due to the recent success of multilayer convolutional encoder-decoder neural networks for MT

---

[2] Since we use the original annotations instead of human post-edits as reference corrections, our HTER scores are the same as TER scores. Nevertheless, we use the term HTER for consistency with the QE task in MT.

(Gehring et al., 2017) and subsequently for GEC (Chollampatt and Ng, 2018) that enables better capturing of the local context, we create a multilayer convolutional variant of the predictor network. Using CNNs also helps in efficient parallelization and improves training and inference speed as shown in (Gehring et al., 2017). We use a similar architecture, which is explained in detail in (Chollampatt and Ng, 2018). In addition, analogous to the backward RNN in the decoder, we use a secondary CNN mechanism in each decoder layer for the convolutional predictor to capture the target words following the predicted target word. The first CNN uses $k-1$ pre-paddings (paddings at the beginning), where k is the convolutional kernel width. This ensures that the decoder state corresponding to the previous target word does not include the target word to be predicted in its computation. For the same reason, the secondary CNN uses $k-1$ post-paddings (paddings at the end). The convolutional predictor uses separate multi-step attention mechanisms (Gehring et al., 2017) for both the CNNs in each decoder layer. Additionally, for the prediction of the target word t_j, the nearby target embeddings $\mathbf{t}_{j-1}$ and $\mathbf{t}_{j+1}$ are also used with a maxout non-linearity (Bahdanau et al., 2015) as done in the RNN-based predictor. The predictor network is trained to minimize the negative log-likelihood loss of the target words, similar to the neural language modeling objective (Bengio et al., 2003). The overall architecture of a predictor model is shown in Figure 1.

Quality Vectors: While training and testing the estimator, the internal hidden representations from the predictor for every hypothesis word, termed as *quality vectors*, are used as inputs to the estimator network. Specifically, we use the "pre-prediction" quality vectors in (Kim et al., 2017a), which performed the best for our GEC QE task. The quality vector $\mathbf{q}_j \in \mathbb{R}^h$ corresponding to the hypothesis word t_j is given by $\mathbf{q}_j = \mathbf{h}_j \circ \mathbf{w}_{t_j}^\top$ where $\mathbf{h}_j \in \mathbb{R}^h$ is the final hidden vector after the maxout layer (Figure 1), $\mathbf{w}_{t_j}$ is the column vector corresponding to the target word t_j in the final linear transformation matrix $\mathbf{W} \in \mathbb{R}^{h\times|V_t|}$ (Figure 1) that projects $\mathbf{h}_j$ to the size of the target vocabulary V_t, and $\circ$ represents element-wise multiplication.

4.2 Estimator Network

The estimator network takes the quality vectors (§4.1) as input and quality scores as labels during training. Our re-implementation of the estimator network in Kim et al. (2017b) uses a bidirectional recurrent network with GRU cells to aggregate the quality vectors. The concatenated final states of the forward and the backward RNNs (with GRU cells) are used as the aggregated summary vector, which is projected to a scalar value using an affine transformation and clipped to the range between 0 and 1 using a sigmoid function.

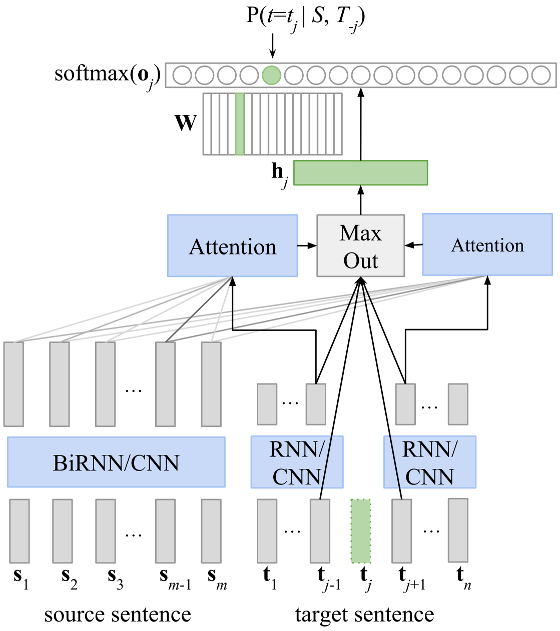

Figure 1: Architecture of a predictor model. For our proposed CNN-based variant multiple layers of CNNs are stacked (only one is shown in the figure).

We also propose a variant of the estimator networks using CNNs that achieves faster training and inference, and performs competitively. CNNs help to aggregate local quality statistics around quality vectors, thereby identifying sequences of words that have a higher or lower quality. In our proposed convolutional estimator model (Figure 2), the quality vectors $\mathbf{q}_1, ..., \mathbf{q}_n$ are transformed to $\mathbf{q}'_1, ..., \mathbf{q}'_n$ where $\mathbf{q}'_j \in \mathbb{R}^{h'}$ and h' is the size of the hidden layer of the estimator. The transformed quality vectors are fed to a convolutional neural network with kernel width k_q and h' filters, followed by the rectified linear units or ReLU (Nair and Hinton, 2010) operation. Sufficient paddings are added on the left and right to retrieve back the same number of output vectors as the input vectors. The input vectors are added to the output vectors as residual connections. The resulting vector

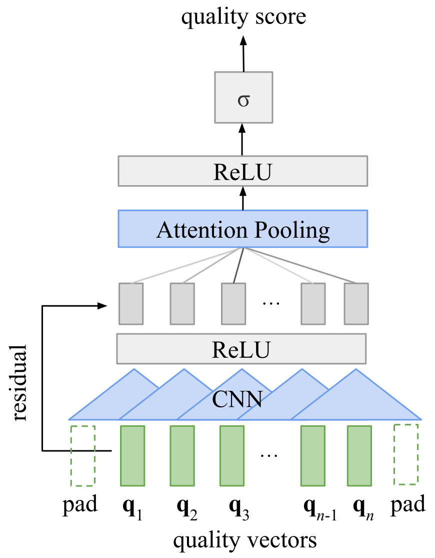

Figure 2: The proposed convolutional estimator neural network with attention-based pooling and residual connections.

after these operations over a single convolutional window around $\mathbf{q}'_j$, denoted by $\mathbf{u}_j \in \mathbb{R}^{h'}$, is given by

$$\mathbf{u}_j = \mathrm{ReLU}\left(\mathrm{Conv}(\mathbf{q}'_{j-\lfloor\frac{k_q}{2}\rfloor}, \cdots, \mathbf{q}'_{j+\lfloor\frac{k_q}{2}\rfloor})\right) + \mathbf{q}'_j$$

$\mathbf{u}_1, ..., \mathbf{u}_j, ..., \mathbf{u}_n$ are aggregated into a summary vector $\mathbf{u} \in \mathbb{R}^{h'}$ using a weighted pooling based on attention weights α_j for each $\mathbf{u}_j$:

$$\alpha_j = \frac{\exp(\mathbf{v}_e \mathbf{u}_j^\top)}{\sum_{k=1}^{n} \exp(\mathbf{v}_e \mathbf{u}_k^\top)}$$
$$\mathbf{u} = \textstyle\sum_{j=1}^{n} \alpha_j \mathbf{u}_j$$

where $\mathbf{v}_e \in \mathbb{R}^{h'}$ is a trainable parameter. The summary vector $\mathbf{u}$ is then fed through another affine transformation with weights $\mathbf{W}_u \in \mathbb{R}^{h' \times h'}$ and biases $\mathbf{b}_u \in \mathbb{R}^{h'}$ followed by ReLU resulting in the output vector $\mathbf{u}'$. The quality score $\hat{q}$ is computed by projecting $\mathbf{u}'$ to a scalar value using an affine transformation with weights $\mathbf{W}_q \in \mathbb{R}^{h' \times 1}$ and bias $b_q \in \mathbb{R}$ followed by a sigmoid operation σ to limit the score to between 0 and 1.

$$\hat{q} = \sigma(\mathbf{u}'\mathbf{W}_q + b_q)$$

The network is trained using mean square error (MSE) as the loss function. We use the pre-trained predictor model only to generate input vectors to the estimator. The predictor parameters are not updated while training the estimator. We apply dropout (Srivastava et al., 2014) to both the RNN-based and CNN-based estimator networks on the inputs to each layer during training. For the CNN-based predictor and estimator, learning is stabilized using strategies in (Gehring et al., 2017) such as the initialization and weight normalization of CNNs and controlling the variance of activations after residual connections.

5 Experimental Setup

5.1 GEC System

The data to train and evaluate the NQE models require GEC system-generated hypotheses. For this, we train a single multilayer convolutional neural network GEC model initialized with pre-trained embeddings following (Chollampatt and Ng, 2018) on Lang-8 corpus only, following the same pre-processing method with 5,000 sentence pairs set aside for validation. The remaining data consists of 2.15M sentence pairs (25.47M source tokens and 28.94M target tokens). For training the model, we use only the annotated sentence pairs after sub-word segmentation (1.28M sentence pairs with 18.50M source sub-words and 21.88 target sub-words). During decoding, a beam width of 12 is used and the top candidate is chosen without any re-scoring.

5.2 Datasets

For training the QE models, we use sentences from the NUS Corpus of Learner English or NUCLE (Dahlmeier et al., 2013) and sentences from the training scripts of the Cambridge Learner Corpus-First Certificate Examination[3] (FCE) (Yannakoudakis et al., 2011) and their corresponding hypotheses generated by the GEC system described in §5.1. We use sentences from the FCE development set and CoNLL-2013 (Ng et al., 2013) test set and their GEC system-generated hypotheses as our development data. We separately test on two datasets, the FCE test set and the CoNLL-2014 test set (Ng et al., 2014). The statistics of the datasets are given in Table 1. The gold-standard scores are obtained by computing HTER using TERp (Snover et al., 2009) and sentence-level M^2 using the MaxMatch scorer with the GEC hypotheses and the reference

---

[3] The file IDs of training, development, and testing scripts of FCE are obtained from the FCE dataset for error detection at https://www.ilexir.co.uk/datasets/index.html

	sentences	src. words	hyp. words
Train	86,293	1,614,120	1,620,399
Development	3,635	63,782	63,890
FCE (test)	2,769	41,457	41,531
CoNLL-2014	1,312	30,144	30,109

Table 1: Statistics of the datasets used for QE.

sentences. When multiple references are available, the gold-standard score is chosen to be the best score (lowest HTER or highest M^2) among the scores computed against each reference separately.

5.3 NQE Models and Training

We build variants of the NQE models denoted by NQE_{XY}, where X indicates the predictor architecture and Y indicates the estimator architecture. X and Y can be recurrent (R) or convolutional (C), of which NQE_{RR} is our replication of (Kim et al., 2017a). The predictor models are trained and validated using parallel data from Lang-8 with 2.15M and 5000 sentence pairs, respectively (described in §5.1). For the predictor models, we use a source and target vocabulary size of 30,000 words, with 500-dimensional source and target embeddings, and 700-dimensional hidden layer. For our convolutional predictor model, a kernel width of 3 and 7 encoder and decoder layers are used. We train the predictor network with ADADELTA optimizer (Zeiler, 2012) with a batch size of 64. We clip gradients by their ℓ^2-norm with a threshold of 5.0 (Pascanu et al., 2013). The estimator networks, both recurrent and convolutional variants, use a hidden layer dimension of 100. They are trained with the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 0.0005 and batch size of 32. We use dropout with a probability of 0.5 during training. Our final model is NQE_{ALL}, which averages the estimated scores of all variants, namely, NQE_{RR}, NQE_{CR}, NQE_{CC}, and NQE_{RC}.

5.4 Baselines

We use two non-neural baselines for comparison to our neural QE models.

AVERAGE: The average score of training sentences is used as the estimated score for all test sentences.

QUEST: We use the standard 17 sentence-level features in QuEst++ (Specia et al., 2015) which has been used as the baseline for WMT QE tasks from 2012 to 2017. The features are based on word-level statistics from the source-hypothesis sentence pairs, and statistics and features from language and lexical translation models trained using a parallel corpus. The descriptions of the 17 features can be found in (Bojar et al., 2017). We use the Lang-8 corpus to train the language and translation models for QuEst.

5.5 Evaluation

We evaluate primarily using the Pearson's correlation coefficient (PCC) metric following the recommendations in (Graham, 2015) and the recent WMT shared tasks (Bojar et al., 2016a, 2017). It is shown in (Graham, 2015) that aggregates of gold score distributions are easier to predict and metrics such as mean absolute error (MAE) and root mean square error (RMSE) over-estimate systems that predict the aggregates accurately despite these systems performing poorly on tail ends of the distribution (higher quality and lower quality samples). PCC does not suffer from this weakness. We use William's Test (Williams, 1959) following (Graham, 2015) to assess the significance of the improvements. However, we also report the root mean square error (RMSE) which reflects the estimator's loss and shows the deviation from the AVERAGE baseline.

6 Experiments and Results

6.1 Estimating Post-Editing Effort

We compare the QE models in terms of their performance in estimating the post-editing effort (HTER). The results, including those of the significance tests, are shown in Table 2. On the FCE test set, all NQE models significantly outperform the baseline QuEst ($p < 0.001$), with the best single model being our NQE_{CR} model (78.28% PCC). The NQE models achieve high PCC ($\geq$ 75%) compared to QuEst (35%). On the CoNLL-2014 dataset, however, only the NQE_{RC} model with our proposed convolutional estimator significantly outperforms all other single models including QuEst. The overall best performance on FCE is achieved by the average model, NQE_{ALL}, significantly outperforming NQE_{CR} ($p < 0.05$) and all other systems ($p < 0.001$). On CoNLL-2014, NQE_{ALL} and NQE_{RC} achieve statistically similar results and outperform all other systems significantly ($p < 0.01$).

	FCE		CoNLL-2014		Dev.	
	PCC↑	RMSE↓	PCC↑	RMSE↓	PCC↑	RMSE↓
Average	-	0.2082	-	0.0972	-	0.1755
QuEst	0.3539	0.1935	0.4113	0.0865	0.5021	0.1514
NQE$_{\text{RR}}$	0.7753	0.1283	0.4334	0.0954	0.6581	0.1309
NQE$_{\text{CR}}$	0.7828	0.1263	0.3096	0.0917	0.6730	0.1283
NQE$_{\text{CC}}$	0.7495	0.1351	0.3888	0.0916	0.6653	0.1293
NQE$_{\text{RC}}$	0.7644	0.1361	**0.4787**	**0.0806**	0.6415	0.1383
NQE$_{\text{ALL}}$	**0.7881**	**0.1258**	0.4687	0.0858	**0.6811**	**0.1270**

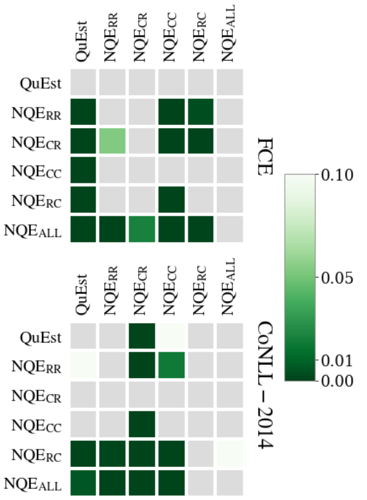

Table 2: Results of the NQE models in estimating post-editing effort (HTER). The matrices on the right show the results of the William's significance tests. A dark green cell indicates the system labeled on the row significantly outperforms the system labeled on the column (p-values shown in the color bar).

	FCE		CoNLL-2014	
	PCC↑	RMSE↓	PCC↑	RMSE↓
Average	-	0.4529	-	0.4302
QuEst	0.2506	0.4585	0.2182	0.4129
NQE$_{\text{RR}}$	0.3594	0.4235	0.2100	**0.3970**
NQE$_{\text{CR}}$	0.4066	0.4153	0.1992	0.4104
NQE$_{\text{CC}}$	0.4129	0.4123	0.2017	0.4038
NQE$_{\text{RC}}$	0.4028	0.4158	0.2104	0.4014
NQE$_{\text{ALL}}$	**0.4186**	**0.4106**	**0.2210**	0.3999

Table 3: Results of the NQE models for sentence-level M^2 estimation.

6.2 Estimating M^2 Score

We use NQE models to estimate the MaxMatch (M^2) GEC evaluation metric at the sentence-level, which computes F$_{0.5}$ based on phrase-level edits. Results are shown in Table 3. All models significantly outperform the baseline QuEst on FCE ($p < 0.01$) test set. NQE$_{\text{ALL}}$ is significantly better than all other systems except NQE$_{\text{CC}}$ on FCE ($p < 0.01$). The PCC on CoNLL-2014 turns out to be much lower for all systems with the NQE models not significantly better than the baseline. Estimating M^2 appears to be more difficult compared to estimating post-editing effort with HTER scores. This could be because M^2 is a phrase-level measure with phrase-boundaries determined by matching with gold annotations, unlike HTER which is a token-level evaluation measure.

6.3 Improving GEC Performance

We use the estimated sentence-level M^2 scores as features to improve the performance of downstream GEC by using them as an additional feature during re-scoring the N-best candidates from a high-performing GEC baseline. Our GEC baseline is built on the multilayer convolutional architecture initialized with pre-trained embeddings and re-scoring (Chollampatt and Ng, 2018). We use the same hyper-parameter settings. The baseline GEC system consists of an ensemble of 3 sets of 4 models each. The first set consists of the 4 models released by Chollampatt and Ng (2018). The second set of 4 models is trained using a label-smoothed cross entropy loss function (Szegedy et al., 2016) which has been found to be effective in neural machine translation (Vaswani et al., 2017; Edunov et al., 2018). We use a smoothing parameter of 0.1 following Vaswani et al. (2017). The third set of 4 models consists of high-recall models that make use of three techniques proposed by Junczys-Dowmunt et al. (2018): (1) pre-training decoder parameters (2) source word dropout, and (3) edit-weighted negative log-likelihood. The parameters of the decoder are initialized using the parameters from a pre-trained neural language model (NLM) of the same architecture as our decoder except for the attention mechanism. We train this NLM using 100 million sentences (1.42 billion words) from the Common Crawl corpora released by Buck et al. (2014) for one epoch. We use the reported hyper-parameters

(Junczys-Dowmunt et al., 2018) for the other two techniques. The 12-best candidates produced by this ensemble are then re-scored using edit operations and language model features following Chollampatt and Ng (2018).

The performance of the baseline system (Base GEC) in terms of document-level $F_{0.5}$ computed by M^2scorer on the FCE and CoNLL-2014 test sets is reported in Table 4. When we use the development set used by Chollampatt and Ng (2018) consisting of 5.4k sentences from NUCLE to train the re-scorer, Base GEC system achieves a competitive performance compared to the top GEC systems with the best-published results on CoNLL-2014 test set: G&J (Grundkiewicz and Junczys-Dowmunt, 2018), JGGH (Junczys-Dowmunt et al., 2018), and C&N (Chollampatt and Ng, 2018). When we make use of the spelling error correction system (+SpellCheck) proposed by Chollampatt and Ng (2017), which is also used by G&J and C&N, our baseline achieves the highest reported $F_{0.5}$ on the CoNLL-2014 test set (56.43) when trained on public corpora alone. To the Base GEC system, we add the sentence-level M^2 scores estimated by the final NQE model (NQE_{ALL}) as a feature in the re-scorer. Since our NQE models use NUCLE during training, we use our development set consisting of 3.6k sentences from FCE and CoNLL-2013 to re-train the re-scorer instead of sentences from NUCLE so that the feature weights will not be biased. We observe a slight drop in performance upon retraining, potentially due to the fewer number of sentences and error annotations in this new development set. The added feature scores are also in the logarithmic scale, similar to LM and the encoder-decoder model score. When the estimated M^2 score is added, we find a significant improvement of 1.18 $F_{0.5}$ on the FCE test set and a significant improvement of 0.25 $F_{0.5}$ score on the CoNLL-2014 test set ($p < 0.001$). Significance testing is done using sign test by bootstrap re-sampling (Koehn, 2004) with 100 samples. The smaller margin of improvement on CoNLL-2014 is expected due to the low PCC values (Table 3). When we add spelling error correction, the results reach 48.70 $F_{0.5}$ score on FCE and 56.52 $F_{0.5}$ score on CoNLL-2014. However, the results obtained by training the re-scorer with our development set (FCE+CoNLL) and adding the NQE models should not be directly compared to the top systems (G&J, JGGH, and C&N) as they do not make use of the FCE data. We also re-score using oracle sentence-level M^2 scores instead of the NQE estimated scores. We find that GEC performance can reach up to 80.74 $F_{0.5}$ for CoNLL-2014 and 76.70 $F_{0.5}$ on FCE. This shows that improving automatic QE can substantially improve downstream GEC simply via re-scoring.

	FCE	CoNLL-2014
Best published results		
G&J (2018) w/ SpellCheck	–	56.25
JGGH (2018)	–	55.8
C&N (2018) w/ SpellCheck	–	54.79
Re-scorer trained with 5.4k sents. from NUCLE		
Base GEC	47.53	55.86
+ SpellCheck	47.79	56.43
Re-scorer trained with FCE+CoNLL dev set		
Base GEC	47.29	55.72
+ M^2 (NQE_{ALL})	**48.47***	**55.97***
+ SpellCheck	48.70	56.52
Base GEC + M^2 (Oracle)	76.70	80.74

Table 4: Performance (in terms of $F_{0.5}$ in %) when NQE-estimated sentence-level M^2 scores are used as features in re-scoring. * indicates statistically significant improvement compared to Base GEC ($p < 0.001$).

7 Discussion and Analysis

Our results show that the NQE models perform better than feature-based baselines for QE of GEC. The crucial component of the NQE model that enables it to make better score estimates is the predictor network whose internal representations (quality vectors) are used as input to the estimator. The sum of nodes of a quality vector corresponds to the output node of the predictor network for a particular target word, and a softmax operation across all vocabulary words results in the predicted probability value. In Figure 3, we analyze the probability outputs by our convolutional predictor network for four GEC hypotheses for a source sentence '*We are all looking forward for you answer .*'. Hypothesis 1 is the source sentence itself and the predictor has rightly identified the location of error by giving a low probability

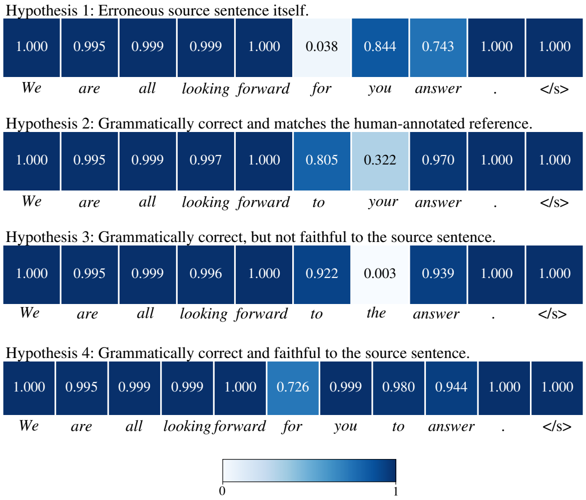

Figure 3: Probabilities predicted by the convolutional predictor for different GEC hypotheses.

score to the erroneous preposition '*for*' (0.038). In Hypothesis 2, which also matches the actual human-annotated reference, the phrase '*for you*' is replaced with '*to your*'. The correct preposition '*to*' gets a higher probability score. In Hypothesis 3, where a less suitable word '*the*' is used, a lower probability score (0.003) is assigned compared to the word '*your*' (0.322) in Hypothesis 2, despite Hypothesis 3 being grammatically correct. This indicates that the predictor rightly considers the faithfulness to the source sentence as well. When we analyze Hypothesis 4, which is grammatically correct and also faithful to the source, the probabilities of all words are much higher. Note that this hypothesis does not match the human annotated reference (Hypothesis 2). It is impractical to have human-annotated references that cover all possible corrections for all source sentences. This issue of reference-coverage has been noted previously in GEC literature (Bryant and Ng, 2015; Napoles et al., 2016b; Choshen and Abend, 2018a). This example shows that QE systems can potentially address this issue, similar to the reference-less evaluation measures for GEC.

We study if the estimator networks are able to count edits, which is the basis of estimating HTER. To do this, we take an example sentence of 14 tokens: '*It will be incredible if we have a chance to watch the show .*' as the source and the hypothesis as well as the reference. We substitute tokens one by one with an arbitrary token '*X*', thereby increasing HTER linearly. Figure 4 shows the performance of the NQE models compared to true HTER scores (straight line). We find that with increasing substitutions, all NQE models show an increasing trend, with the models having convolutional components NQE_{CR}, NQE_{CC}, and NQE_{RC} having Pearson's correlation coefficients (PCC) of 0.982, 0.976, and 0.986, respectively, and 0.939 PCC for the recurrent variant NQE_{RR} compared to true HTER scores. The average model NQE_{ALL} has the highest PCC value of 0.995. This example illustrates that the NQE models are able to capture edit counts for estimating HTER scores.

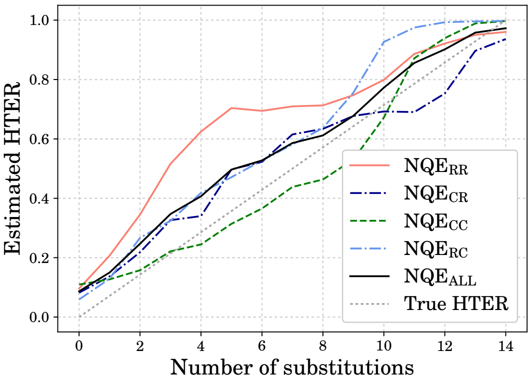

Figure 4: Estimated HTER scores of NQE models increasing with the number of incorrect substitutions.

8 Conclusion and Future Work

We propose the first supervised approach to quality estimation (QE) for GEC system outputs. We propose several neural QE model variants that perform significantly better than feature-based baselines in estimating the post-editing effort of GEC output sentences. We also show that the QE variants perform reasonably well on a more difficult task of estimating quality in terms of a GEC evaluation metric, M^2, by showing that the estimated scores are useful in improving GEC performance via N-best re-scoring. In future, the general framework of QE for GEC can be used to train on subjective human rankings of hypotheses as well, so that the system can learn the intuitions underlying human judgments of quality instead of estimating a pre-defined measure such as HTER or M^2.

References

Hiroki Asano, Tomoya Mizumoto, and Kentaro Inui. 2017. Reference-based metrics can be replaced with reference-less metrics in evaluating grammatical error correction systems. In *Proceedings of the*

Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016a. Findings of the 2016 conference on machine translation (WMT16). In *Proceedings of the First Conference on Machine Translation*.

Ondřej Bojar, Christian Federmann, Barry Haddow, Philipp Koehn, Matt Post, and Lucia Specia. 2016b. Ten years of WMT evaluation campaigns: Lessons learnt. In *Proceedings of LREC Workshop Translation Evaluation: From Fragmented Tools and Data Sets to an Integrated Ecosystem*.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the Common Crawl. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: Towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Leshem Choshen and Omri Abend. 2018a. Inherent biases in reference-based evaluation for grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Leshem Choshen and Omri Abend. 2018b. Referenceless measure of faithfulness for grammatical error correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2017. Referenceless quality estimation for natural language generation. In *Proceedings of the First Workshop on Learning to Generate Natural Language*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*.

Yvette Graham. 2015. Improving evaluation of machine translation quality estimation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Michael Heilman, Aoife Cahill, Nitin Madnani, Melissa Lopez, Matthew Mulholland, and Joel Tetreault. 2014. Predicting grammaticality on an ordinal scale. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Hyun Kim, Hun-Young Jung, Hongseok Kwon, Jong-Hyeok Lee, and Seung-Hoon Na. 2017a. Predictor-estimator: Neural quality estimation based on target word prediction for machine translation. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 17(1):3:1–3:22.

Hyun Kim and Jong-Hyeok Lee. 2016a. Recurrent neural network based translation quality estimation. In *Proceedings of the First Conference on Machine Translation*.

Hyun Kim and Jong-Hyeok Lee. 2016b. A recurrent neural networks approach for estimating the quality of machine translation output. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. 2017b. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proceedings of the Second Conference on Machine Translation*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the Third International Conference on Learning Representations*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016a. GLEU without tuning. *arXiv preprint arXiv:1605.02592*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2016b. There's no comparison: Reference-less evaluation metrics in grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

Matthew Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*.

Lucia Specia, Gustavo Paetzold, and Carolina Scarton. 2015. Multi-level translation quality prediction with QuEst++. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*.

Lucia Specia, Kashif Shah, Jose G.C. de Souza, and Trevor Cohn. 2013. QuEst - a translation quality estimation framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Lucia Specia, Marco Turchi, Nicola Cancedda, Marc Dymetman, and Nello Cristianini. 2009. Estimating the sentence-level quality of machine translation systems. In *Proceedings of the 13th Conference of European Association for Machine Translation*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Sanja Štajner, Maja Popović, and Hanna Béchara. 2016. Quality estimation for text simplification. In *Proceedings of the LREC 2016 Workshop and Shared Task on Quality Assessment for Text Simplification*.

Craig Stewart, Nikolai Vogler, Junjie Hu, Jordan Boyd-Graber, and Graham Neubig. 2018. Automatic estimation of simultaneous interpreter performance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Evan James Williams. 1959. *Regression Analysis*. Wiley.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Matthew D Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Transferring from Formal Newswire Domain with Hypernet for Twitter POS Tagging

Tao Gui[1], **Qi Zhang**[1], **Jingjing Gong**[1],
Minlong Peng[1], **Di Liang**[1], **Keyu Ding**[2], **Xuanjing Huang**[1]
[1]Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
[2]iFlytek Co., Ltd.
[1]`{tgui16,qz,jjgong15,mlpeng16,liangd17,xjhuang}@fudan.edu.cn`
[2]`kyding@iflytek.com`

Abstract

Part-of-Speech (POS) tagging for Twitter has received considerable attention in recent years. Because most POS tagging methods are based on supervised models, they usually require a large amount of labeled data for training. However, the existing labeled datasets for Twitter are much smaller than those for newswire text. Hence, to help POS tagging for Twitter, most domain adaptation methods try to leverage newswire datasets by learning the shared features between the two domains. However, from a linguistic perspective, Twitter users not only tend to mimic the formal expressions of traditional media, like news, but they also appear to be developing linguistically informal styles. Therefore, POS tagging for the formal Twitter context can be learned together with the newswire dataset, while POS tagging for the informal Twitter context should be learned separately. To achieve this task, in this work, we propose a hypernetwork-based method to generate different parameters to separately model contexts with different expression styles. Experimental results on three different datasets show that our approach achieves better performance than state-of-the-art methods in most cases.

1 Introduction

With the continuous growth of online communication, hundreds of millions of online conversational messages have become important resources for various applications such as real-time event detection (Sakaki et al., 2010), stock prediction (Bollen et al., 2011) and public health analysis (Wilson and Brownstein, 2009). Because these applications need to process natural language text, POS tagging, which is one of the fundamental natural language processing tasks, has become one of the basic pre-processing components of such applications. The performance of POS tagging may highly impact the results of these applications.

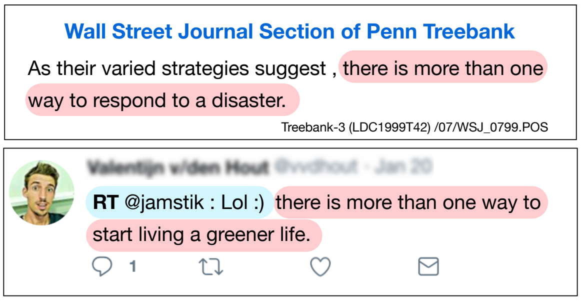

Wall Street Journal Section of Penn Treebank

As their varied strategies suggest , there is more than one way to respond to a disaster.

Treebank-3 (LDC1999T42) /07/WSJ_0799.POS

[redacted]

RT @jamstik : Lol :) there is more than one way to start living a greener life.

Figure 1: Examples of WSJ and tweets. Segments with red highlights can be regarded as the similar expressions. Segments with blue highlights correspond to expressions that cannot be learned from WSJ.

Most of the POS tagging methods that can achieve state-of-the-art performance are based on supervised learning algorithms (Gimpel et al., 2011). Although these methods can achieve good performance for in-domain data, their performance usually drops quickly when processing data from a domain that is different from that of the training data (Caruana and Niculescu-Mizil, 2006). To achieve better performance, we usually need to manually label a large amount of in-domain data. However, the task of constructing labeled data is time-consuming and tedious. Currently, various methods have been proposed to solve this problem using out-of-domain data, including domain adaptation (Daumé III, 2009; Gui et al., 2017), multi-task learning (Ben-David et al., 2007), and dual learning (Chandrasekaran et al., 2014).

Most existing methods aim to learn the shared representations or parameters, which can reduce the classification or regression model errors of each task/domain. However, these methods usually ignore the fact that each domain has domain-specific features that should not be shared. From

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2540–2549
Brussels, Belgium, October 31 - November 4, 2018.

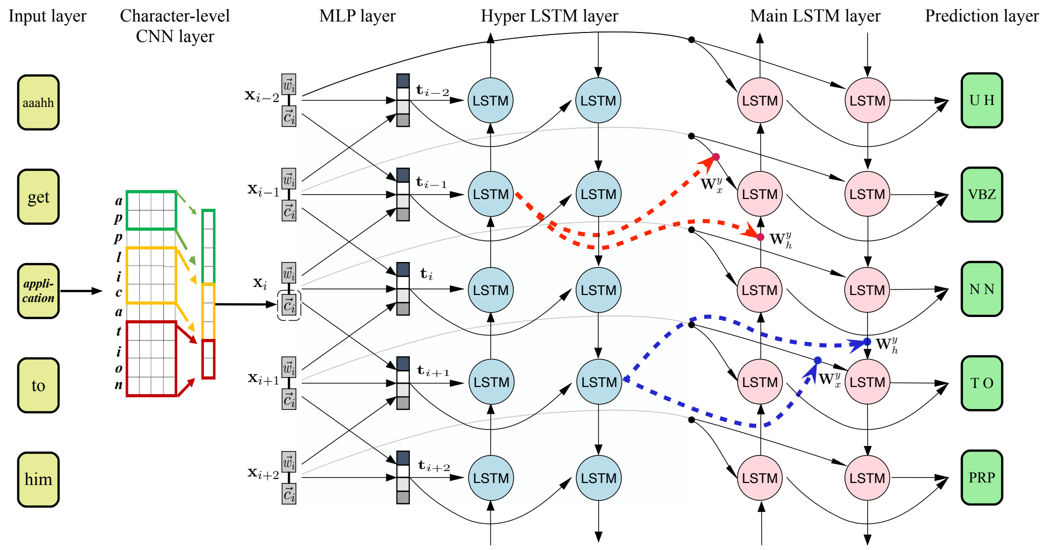

Figure 2: Model architecture of the proposed dynamic conversion neural networks.

the language characteristics, in spite of the expressions that are similar to the newswire text, Twitter has some informal expressions that cannot be shared, as shown in Figure 1. Hu et al. (2013) investigated the characteristics of language on Twitter and found that Twitter users not only tended to mimic the linguistic practices of traditional media, like news, they also appeared to be developing linguistically unique styles. We also believe that the tweets simply follow the standard language rules, and at some point eventually deviate from those. Thus, tweets are a combination of formal expressions and informal expressions with conversions between different expression styles.

Based on the above observations, we believe that annotated sentences of newswire text can be selectively used to help tag contextual segments of tweets, especially for formal expressions. To achieve this goal, in this work we adopt bidirectional long short-term memory (bi-LSTM) networks (Schuster and Paliwal, 1997), which have been successfully used for various sequence tagging problems. However, different from previous methods, the formal expressions and informal expressions in a sentence should be separately modeled. Inspired by recent work on dynamic parameter prediction (Ha et al., 2016), we propose a method to generate the context-specific parameters of the bi-LSTM based on different styles of context for POS tagging. We evaluated our models on three different corpora. The results demonstrated that the proposed method can benefit from annotated newswire text data and achieve competitive performance. In addition, we visualized the context distributions and the change of parameters. The visualization results verified the fact that different contexts produce different parameters for POS tagging.

The main contributions of the paper can be summarized as follows.

1. We study problems in the segment modeling method to apply domain adaptation to the POS tagging task. Based on the observations from a linguistic perspective, we found that there are many shared expressions between the newswire and tweets, and some expressions cannot be shared.

2. A novel neural network architecture based on the bi-LSTM was proposed to perform the task. Different parameters were applied in different contexts.

3. Experimental results demonstrated that the proposed method can benefit from different domains. We also conducted qualitative and quantitative analyses to show why our model can achieve better performance.

2 Approach

Twitter is responsible for colorful linguistic expressions, and full of the conversions between formal expressions and informal expressions. To address this problem, we propose the Dynamic Conversion Neural Networks (DCNN), which can dynamically generate different parameters for POS tagging based on different contexts as shown in Figure 2. Our model mainly consists of four parts: (1) a CNN layer for extracting word representations $\mathbf{x}_i$, (2) an MLP layer for producing low-dimensional context representations $\mathbf{t}_i$, (3) a hyper LSTM layer for generating the weights $\mathbf{W}$ of a main LSTM, and (4) the main LSTM layer with dynamic parameters for POS tagging.

The architecture of the main network is the same with any sequence labeling model, which learns to map the word representations to the corresponding labels. However, the parameters of the main network can be modified according to our purpose. we use a low-dimensional context distribution vector $\mathbf{t}_i$ as the input of the hyper LSTM to generate the weights of the main LSTM. Thus, the weights of the main LSTM will be subject to a change in context vector. Therefore, the main LSTM can predict the POS tag based on the different parameters.

Ha et al. (2016) also proposed a HyperRNN network, in which the hyper net is influenced by the main net. This is inconsistent with our motivation. Different from (Ha et al., 2016), to make the parameters of the main LSTM totally controlled by the context distribution, we changed the architecture by cutting off the data path from the main LSTM to the hyper LSTM. This method can prevent the hidden states of the main LSTM from influencing the hyper LSTM. In addition, we add an extra layer of learning the context representations based on features returned by a CNN.

2.1 Word and Context Representations

Out-of-vocabulary words are frequently used in Twitter. Moreover, new symbols, abbreviations, and words are constantly being created. These make word representations difficult to address. Thus, robust methods should be used to extract the morphological and shape information from words.

Inspired by (Santos and Zadrozny, 2014), we adopted character-level convolutional neural networks (CNN layer) to tackle this problem, which can take all of the characters of the word into consideration and output important orthographic features (Santos and Zadrozny, 2014). Suppose that we are given the sentence $X = \{w_1, w_2, \dots\}$ with vocabulary $\mathcal{V}$ of words. We use multiple filters with varying widths to obtain the orthographic feature vector $\vec{c_i}$ for word w_i. Then, the orthographic feature vector $\vec{c_i}$ is concatenated to the word embedding $\vec{w_i}$ to form word representation $\mathbf{x}_i$ as the input of the main LSTM. Utilizing a bi-LSTM to model sentences, the model can extract the sequential relations and contextual information.

The context is a fixed window of words around target word. The context representations are learned by the MLP later. To do this, we apply a fully connected neural network, which takes sequential word representations in a fixed window as input to generate a low-dimensional vector $\mathbf{t}_i$ as follows:

$$\mathbf{t}_i = softmax(MLP[\mathbf{x}_{i-r}...\mathbf{x}_{i-1} \oplus \mathbf{x}_i \oplus \mathbf{x}_{i+1}...\mathbf{x}_{i+r}]), \quad (1)$$

where $[\cdot \oplus \cdot]$ represents concatenation operation. r represents the length from central word x_i to the edge of the window. MLP is the multilayer perceptrons function, which transfers the context matrix to a low-dimensional vector. We apply MLP to every window of contexts. $softmax$ denotes the softmax function that converts the context vector into a probability distribution. The goal is to learn an MLP layer that, given sequential word representations, estimates a distribution over the contexts.

2.2 Adaptive Weight Generation

The identical weights at each time step will limit the expressiveness of recurrent neural network (RNN) (Ha et al., 2016). To overcome the limitation, our model uses a small network (hyper network) taking low-dimensional context representations as inputs to dynamically generate the parameters of a large network (main network) for POS tagging. Different with (Ha et al., 2016), at every time step, the hyper LSTM only takes the context representation $\mathbf{t}_i$ as an input and generates the hidden state $\hat{\mathbf{h}}_i$ as an output. This hidden state $\hat{\mathbf{h}}_i$ is used to generate the weights for the main LSTM at the same time step. The hyper LSTM and main LSTM are jointly trained with backpropagation and gradient descent. Next, we will give a more formal description of the weight generation.

The hyper LSTM is a standard LSTM (Hochreiter and Schmidhuber, 1997), which takes context vectors as inputs and outputs hidden states. The hyper LSTM is defined as follows:

$$\begin{bmatrix}\hat{\mathbf{g}}_i\\ \hat{\mathbf{i}}_i\\ \hat{\mathbf{f}}_i\\ \hat{\mathbf{o}}_i\end{bmatrix} = \left(\begin{bmatrix}\mathbf{W}_t^{\hat{g}}, \mathbf{W}_{\hat{h}}^{\hat{g}}\\ \mathbf{W}_t^{\hat{i}}, \mathbf{W}_{\hat{h}}^{\hat{i}}\\ \mathbf{W}_t^{\hat{f}}, \mathbf{W}_{\hat{h}}^{\hat{f}}\\ \mathbf{W}_t^{\hat{o}}, \mathbf{W}_{\hat{h}}^{\hat{o}}\end{bmatrix} \bullet \begin{bmatrix}\mathbf{t}_i\\ \hat{\mathbf{h}}_{i-1}\end{bmatrix} + \begin{bmatrix}\hat{\mathbf{b}}^{\hat{g}}\\ \hat{\mathbf{b}}^{\hat{i}}\\ \hat{\mathbf{b}}^{\hat{f}}\\ \hat{\mathbf{b}}^{\hat{o}}\end{bmatrix}\right) \tag{2}$$

$$\begin{aligned}\hat{\mathbf{c}}_i &= \phi(\hat{\mathbf{g}}_i) \odot \sigma(\hat{\mathbf{i}}_i) + \hat{\mathbf{c}}_{i-1} \odot \sigma(\hat{\mathbf{f}}_i)\\ \hat{\mathbf{h}}_i &= \sigma(\hat{\mathbf{o}}_i) \odot \phi\left(\hat{\mathbf{c}}_i\right),\end{aligned}$$

where ϕ denotes the $tanh$ function, and σ is the $sigmoid$ function. $\odot$ and $\bullet$ represent the Hadamard product and matrix product, respectively. We assume that $\hat{y}$ is one of $\{\hat{g}, \hat{i}, \hat{f}, \hat{o}\}$. The hyper LSTM has $N_{\hat{h}}$ hidden units, and N_t is the dimensionality of $\mathbf{t}_i$. Then, $\mathbf{W}_t^{\hat{y}} \in \mathbb{R}^{N_{\hat{h}} \times N_t}$, $\mathbf{W}_{\hat{h}}^{\hat{y}} \in \mathbb{R}^{N_{\hat{h}} \times N_{\hat{h}}}$, $\hat{\mathbf{b}}^{\hat{y}} \in \mathbb{R}^{N_{\hat{h}}}$ are the parameters of the hyper LSTM and stay invariable during one sentential sequence.

Inspired by (Ha et al., 2016), we adopted a *weight scaling vector* $\mathbf{d}$ which is a linear projection of $\hat{\mathbf{h}}_i$. $\mathbf{d}$ is used to linearly scale each row of the weight matrix in the standard LSTM. Because the context vector $\mathbf{t}$ is produced by the different contexts at each time step, the hidden state $\mathbf{h}_i$ and $\mathbf{d}_i$ will change corresponding to $\mathbf{t}_i$. Thus, the main LSTM can be modified as follows:

$$\begin{bmatrix}\mathbf{d}_{i,x}^{y}\\ \mathbf{d}_{i,h}^{y}\\ \mathbf{d}_{i,b}^{y}\end{bmatrix} = \begin{bmatrix}MLP_x^y(\hat{\mathbf{h}}_i)\\ MLP_h^y(\hat{\mathbf{h}}_i)\\ MLP_b^y(\hat{\mathbf{h}}_i)\end{bmatrix}$$

$$\begin{aligned}\mathbf{y}_i = &\left[\mathbf{d}_{i,x}^{y} \otimes \mathbf{W}_x^y, \mathbf{d}_{i,h}^{y} \otimes \mathbf{W}_h^y\right] \bullet \begin{bmatrix}\mathbf{x}_i\\ \mathbf{h}_{i-1}\end{bmatrix} \\ &+ \mathbf{d}_{i,b}^{y} \otimes \mathbf{b}^y\\ \mathbf{c}_i =&\phi(\mathbf{g}_i) \odot \sigma(\mathbf{i}_i) + \mathbf{c}_{i-1} \odot \sigma(\mathbf{f}_i)\\ \mathbf{h}_i =&\sigma(\mathbf{o}_i) \odot \phi\left(\mathbf{c}_i\right),\end{aligned} \tag{3}$$

where $\otimes$ represents the element-wise product with broadcasting. $\mathbf{y}$ is one of $\{\mathbf{g}, \mathbf{i}, \mathbf{f}, \mathbf{o}\}$. Generally, $N_{\hat{h}}$ and N_t are much smaller than N_h and N_x, respectively. Thus, the size of the parameters in the hyper LSTM is hundreds of times less than that of the standard LSTM.

According to the above functions, if the model is given contexts with different styles, it will generate different parameters for all kinds of gates in the main LSTM. Then, the outputs of the main LSTM are used to predict the POS tags of the central words with the cross entropy loss:

$$\mathcal{L}_{POS} = -\sum_i z_i * \log \hat{z}_i, \tag{4}$$

where z_i is the one-hot vector of the POS tagging label corresponding to $\mathbf{x}_i$. $\hat{z}_i$ is the output of the top softmax layer: $\hat{z}_i = softmax(MLP(\mathbf{h}_i))$.

3 Experimental Setup

In this section, we will first detail the datasets we used. Then, we will describe several baseline methods, including a number of classic taggers and a series of deep learning sequence labeling methods.

3.1 Datasets

Following (Derczynski et al., 2013), we use RIT-Twitter (Ritter et al., 2011) as our main dataset. The RIT-Twitter was split into training, development and evaluation sets (RIT-Train, RIT-Dev, RIT-Test). The splitting method was shown in (Derczynski et al., 2013). In order to verify the validity of our model, we also tested it on two more datasets, NPSChat (Forsythand and Martell, 2007), and ARK-Twitter (Gimpel et al., 2011) using standard splits. The tag-sets of the RIT-Twitter and NPSChat are PTB-like, while that of the ARK-Twitter is specific. In order to use WSJ labeled data in experiments on ARK-Twitter, we performed the mapping from PTB tag-sets to ARK tag-sets, according to the PTB POS Tagging Guidelines (Santorini, 1990) and ARK Guidelines[1]. The mapping proceeded from fine to coarse.

For pretraining the word embedding, we constructed a dataset containing 30 million tweets, from Twitter using its API. We introduced a newswire dataset containing 1173K tokens as the written language dataset, namely the Wall Street Journal (WSJ) from the Penn TreeBank v3 (Marcus et al., 1993). During training, we mixed each of RIT-Twitter, NPSChat and ARK-Twitter with WSJ into three kinds of training data.

The detailed data statistics of the above datasets used in this work are listed in Table 1.

Dataset	#Train	#Dev	#Test
RIT-Twitter	10652	2242	2291
NPSChat	40497	-	4500
ARK-Twitter	26594	-	7707

Table 1: The statistics of the datasets used in our experiments, where # represents the number of tokens in datasets.

---

[1] http://www.ark.cs.cmu.edu/TweetNLP/

3.2 Competitor Methods

We applied several classic and state-of-the-art methods for comparison. In addition, we used a series of deep learning sequence labeling methods as baselines for comparison, as follows:

Stanford POS Tagger is a widely used part-of-speech taggers described in (Toutanova et al., 2003). It demonstrates the broad use of features and appropriate model regularization, which produces a superior level of performance (97.24%). In this work, we trained it using two different sets: sections 0-18 of the WSJ (**Stanford-WSJ**) and a mixed corpus of WSJ, IRC, and Twitter (**Stanford-MIX**).

T-POS (Ritter et al., 2011) adopts hierarchical clustering and Brown clustering methods to address the issue of OOV words and lexical variations. It also uses conditional random fields and other standard sets of features to perform the task. In this work, we trained it using three different sets: the WSJ (**T-POS-WSJ**), RIT-Train (**T-POS-RIT**) and a mixed corpus of WSJ, IRC, and RIT-Train (**T-POS-MIX**).

GATE Tagger (Derczynski et al., 2013) uses an approach that combines the available taggers for different tagsets. The tagger adopts a vote-constrained bootstrapping method with unlabeled data and assigns prior probabilities to handle of unknown words and slang.

ARK Tagger (Owoputi et al., 2013) is a system that reports the best accuracy on ARK-Twitter. It uses unsupervised word clustering and a variety of lexical features.

TPANN (Gui et al., 2017) applies adversarial networks and autoencoder to model labeled out-of-domain data, unlabeled in-domain data and labeled in-domain data and achieved the best performance on RIT-Twitter.

Bidirectional LSTM (Bi-LSTM) (Wang et al., 2015) has been widely used in a variety of sequence labeling tasks. In this work, we also evaluated it as a baseline.

Bi-HyperLSTM (Ha et al., 2016) was used as a substitute for the standard Bi-LSTM. What makes the Bi-HyperLSTM model different from the proposed model is that we used context distribution to generate the parameters of main LSTM.

3.3 Initialization and Hyperparameter

The word embeddings for all the models were initialized with the word2vec tool (Mikolov et al., 2013) on 30 million tweets. The other parameters excluding the word embeddings, such as the parameters in LSTM and MLP, were initialized by randomly sampling from a uniform distribution in [-0.05, 0.05].

The dimensionality of the word embedding was set at 200. The dimensionality for the randomly initialized character embedding was set at 25. We adopted a hyper LSTM with 160 hidden neurons to produce the weights of each gates of the main LSTM with 250 hidden neurons. The dimensionality of the context vector was set at 10.

Our DCNN could be trained end-to-end with backpropagation and gradient-based optimization was performed using the Adam update rule (Kingma and Ba, 2014) with learning rate 0.0001.

4 Results and Analysis

In this section, we will report the experimental results and a detailed analysis of the results for the three different datasets.

4.1 Evaluation on RIT-Twitter

The RIT-Twitter was introduced in (Ritter et al., 2011). This dataset uses a tagset based on the Penn Treebank tagset with several Twitter-specific tags: *retweets, @usernames, hashtags, and urls.*

Table 2 lists the results of our method compared with other methods on this dataset. The first part shows the results of the classic methods. From the result of Stanford-WSJ, we can see that although it can achieve a superior level of performance (97.24%) on the WSJ dataset, the accuracy drops significantly to 73.37% when applied to the Twitter dataset. If we add some in-domain data to the training set, the Stanford-MIX can improve by 10% compared to the Stanford-WSJ. The same phenomenon can be observed from T-POS tagger. If we apply more features, like clustering, bootstrapping and lexical features, the T-POS, GATE tagger and ARK tagger can achieve better performances. Although TPANN achieve accuracy of 90.92%, it incorporates additional a large amount of in-domain unlabeled data. Our method is more competitive because of the use of much fewer data sets.

Methods	**Training Set**	**RIT-Test**	**RIT-Dev**
Stanford-WSJ (Toutanova et al., 2003)		-	73.37%
Stanford-MIX		-	83.14%
T-POS-WSJ (Ritter et al., 2011)		81.30%	
T-POS-RIT		84.55%	84.83%
T-POS-MIX		88.30%	-
GATE Tagger (Derczynski et al., 2013)		88.69%	89.37%
ARK Tagger (Owoputi et al., 2013)		90.40%	-
TPANN (Gui et al., 2017)		90.92%	91.08%
Bi-LSTM	RIT-Train	89.48%	89.30%
Bi-LSTM*		89.31%	90.37%
Bi-HyperLSTM		88.65%	89.16%
Bi-HyperLSTM*		88.30%	89.56%
DCNN		89.87%	90.50%
Bi-LSTM	WSJ + RIT-Train	90.09%	90.37%
Bi-LSTM*		90.31%	90.81%
Bi-HyperLSTM		90.57%	90.41%
Bi-HyperLSTM*		90.44%	90.54%
DCNN		**91.18%**	**91.17%**

Table 2: Token level accuracies of different methods on RIT-Test and RIT-Dev. The first part demonstrates the results of classic methods. The second part demonstrates a series of deep learning methods trained on RIT-train. The third part demonstrates the same deep learning methods train on the mixed dataset of RIT-train and WSJ. DCNN refers to our dynamic conversion neural network. Other models are described in the Section 3.2. The symbol * represents the model concatenates the context vectors with the word representations as inputs.

The second part shows the results of the deep learning methods trained on the RIT-Train dataset. We can see that if the sequence labeling methods are just trained on the RIT-Train dataset, their accuracies can exceed those of most conventional taggers. Thus, the deep learning methods are competitive and avoid feature engineering. Compared with other models, the DCNN achieved best performance among the models just trained on RIT-Train dataset.

The third part shows the results of the deep learning methods trained on the mixed dataset of the RIT-Train and WSJ. As observed, when we added the WSJ data to train the models, all of them could obtain different degrees of improvement. Moreover, our model could make better use of the out-of-domain data and obtained the best result. Compared with the ARK tagger, which achieved the previous best result in conventional methods, our model was almost 0.78% better. The error reduction rate was more than 8%. Our model also outperformed the TPANN, which incorporated additional unlabeled in-domain data.

From the perspective of utilizing a low-dimensional context vector, we provided the same information (word information and context information) for all of the deep learning models as shown in Table 2. However, except for the DCNN, the other models were incapable of utilizing the context information. Most of the models could not obtain obvious improvement. In contrast, our DCNN could make better use of the context information to generate more appropriate parameters for POS tagging. Next, we will analyze the behavior how the DCNN changes parameters when encountering different context vectors.

Intuitively, contexts with different language expression styles should be transformed into different vectors. Figure 3 visualizes the context distribution. Subfigure (a) shows the context vector extracted from WSJ. We can see that the formal expressions are mainly concentrated in the middle of the four dimensions. This phenomenon can be observed in the subfigure (b), where the formal expressions in the Twitter are concentrated in the middle of the same dimensions and the informal expressions are concentrated in another three dimensions. Notice that in our experimental setup, the dimensionality of the context vector is

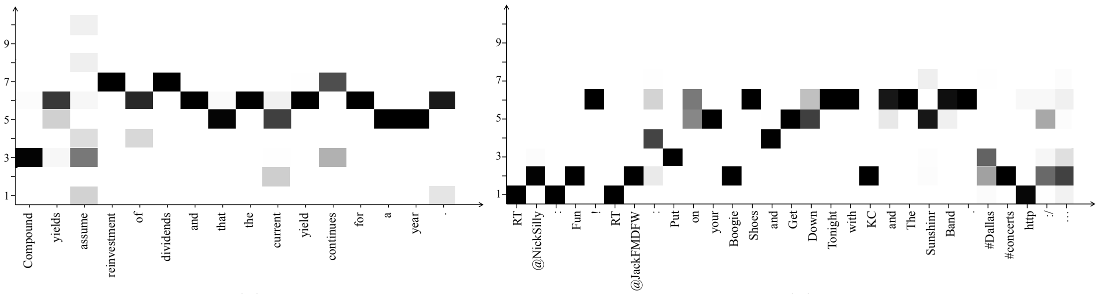

(a) (b)

Figure 3: Visualization of context distribution. The left sentence comes from WSJ, and the right one comes from Twitter.

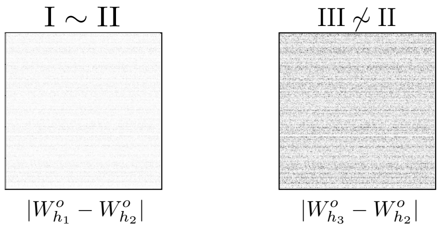

I ➤ *Compound yields assume reinvestment of dividends and that the current yield continues for a year.* (WSJ Formal)

II ➤ *Very grateful for another Monday and a new week full of endless possibilities ! Go World!* (Twitter Formal)

III ➤ *RT @NickSilly : Fun ! RT @JackFMDFW : Put on your Boogie Shoes and Get Down Tonight with KC and The Sunshine Band . #Dallas #concerts http :/ ...* (Twitter Informal)

Figure 4: Visualization of the comparison of weight matrices. We denote the weight matrix of output gate as W^o_h. Each subgraph represents the result of element-wise subtraction $|W^o_{h_i} - W^o_{h_j}|$, where $|\cdot|$ means absolute value, i and j are the sequence numbers of sentences. If the two sentences have a similar expression style, then the absolute value would be close to zero represented by white color. We use $i \sim j$ to represent it. On the contrary, We use $i \not\sim j$ to represent the different expression styles. We only visualize the weights on the last time step.

set to 10. However, the values of the last three dimensions are close to zero. Consequently, we set this hyperparameter to 7 and we achieve a higher accuracy of 91.27%.

Figure 4 shows how the weight matrix W^o_h in output gate gets changed when the model inputs different kinds of contexts. Through making a comparison among the sentences I, II and III. we can find that although the sentences II and III are both from Twitter, whereas the sentence I is from WSJ, If the style of sentence II is close to that of sentence I, then the model will produce similar weight values to achieve the task. If the style of sentence I is different from that of sentence III, then the model will produce different parameters more suitable for Twitter-specific sentences. The similar phenomena can be found in other gates.

Methods	**Acc.**
Forsyth (2007)	90.8%
ARK Tagger	93.4% ± 0.3%
Gui et al. (2017)	**94.1%**
Bi-LSTM(IRC)	90.3%
Bi-LSTM(WSJ + IRC)	93.2%
DCNN	**94.0%**

Table 3: Accuracy comparison of different methods on NPSChat Corpus.

4.2 Evaluation on NPSChat

The NPSChat Corpus (Forsythand and Martell, 2007) is a PTB-POS annotated dataset of Internet Relay Chat (IRC) room messages from 2006. The corpus consists of 10,567 posts out of approximately 500,000 posts gathered from various online chat services in accordance with their terms of service. The authors of the corpus made several decisions during the process that were unique to the chat domain regarding some abbreviations, emotions and misspelled words. For example, LOL and :-) were frequently encountered in the chat messages. Because these expressions conveyed emotion, they were treated as individual tokens and tagged as interjections (UH).

Table 3 lists the results of different taggers evaluated on NPSChat. Our method was tested using the same setup as the experiments in (Forsyth, 2007). The training part contained 90% of the data. The testing part contained the remaining 10%. Based on the results, we can see that our method could achieve the best accuracy (94.0%), which was significantly better than 90.8% (Forsyth, 2007). They trained the tagger on a mix of several corpora tagged with the Penn Treebank tag set. Our method also outperformed

Methods	Acc.
Gimpel et al. (2011)	89.17%
Gui et al. (2017)	92.8%
ARK Tagger	**93.2%**
ARK Tagger†	92.38%
Bi-LSTM(OCT27)	90.59%
Bi-LSTM(WSJ + IRC + OCT27)	91.57%
DCNN(WSJ + IRC + OCT27)	**92.42%**

Table 4: Accuracy comparison of different methods on ARK-Twitter Corpus. The symbol † represents ARK Tagger trained without tagdicts and namelists.

the ARK tagger, which applies various external corpus and features, e.g., Brown clustering, PTB, Freebase lists of celebrities, and video games.

4.3 Evaluation on ARK-Twitter

The ARK-Twitter that contains 34K tokens uses a novel tagset. The training set (OCT27) is provided in (Gimpel et al., 2011). It is a dataset of POS-tagged tweets consisting almost entirely of tweets sampled from one particular day (October 27, 2010). However, the test set was introduced in (Owoputi et al., 2013), and contains 574 tweets (DAILY547). The DAILY547 consists of one random English tweet from every day between January 1, 2011 and June 30, 2012. Thus, the distribution between the training set and test set may be slightly different. For example, a substantial fraction of the messages in the training data are about a basketball game that occurred on that day.

The results of the ARK tagger and TweetNLP Tagger in Table 4 are reported in (Owoputi et al., 2013). We can see that our method could significantly outperform the TweetNLP Tagger. However, our method was worse than the ARK tagger. By analyzing the incorrect results, we found that 20.3% of the errors occurred between nouns and proper nouns. Because our model does not incorporate any knowledge of proper nouns, it is difficult for it to recognize proper nouns from datasets. As reported in (Owoputi et al., 2013), if ARK-tagger does not add tag dictionary features and name list features, its performance will drop to 92.38%, which is lower than that of the DCNN. Thus, our model is also competitive when lacking of knowledge.

5 Related Work

At a very early time, Schmidhuber (1992) began to explore the concept of fast weights, in which one network can produce context-dependent weight changes for a second network (Schmidhuber, 1992, 1993). Moreover, they provided the theoretical possibility of a recurrent network version. Recently, numerous studies have been conducted in this field (Moczulski et al., 2015; Fernando et al., 2016). De Brabandere et al. (2016) introduced a new framework called the dynamic filter network where the filters in the CNN are generated dynamically. Ha et al. (2016) explored the use of this approach in recurrent networks. Our work uses a different mechanism to generate parameters, which can make the parameters subject to a change in context representations. we cut off the data path from the main LSTM to the hyper LSTM. This method can prevent the hidden states of the main LSTM from influencing the hyper LSTM.

Recently, deep learning has achieved promising results on POS tagging. Santos and Zadrozny (2014) used a CNN to construct a character-based model for English (PTB) and Portuguese. Wang et al. (2015) used the bi-LSTM on WSJ and reported a state-of-the-art performance. However, because of a lack of training data and an unconstrained writing style, these models encountered resistance in the implementation process on Twitter. In this work, we focused on the linguistic correlation between Twitter and newswire and took the linguistic characteristics into consideration. To selectively utilize out-of-domain data, we used a low-dimensional context vector to generate different parameters for text with different expression styles and obtained better results.

6 Conclusion

In this work, we study the problem of incorporating labeled newswire texts for Twitter POS tagging tasks. From a linguistic perspective, we find that Twitter users not only tend to mimic the formal expressions of traditional media, like news, but they also appear to be developing linguistically informal styles. Hence, we predict that labeled data from the newswire should selectively be used to help tag contextual segments of tweets. To achieve this task, we introduce a novel deep neural network architecture that can dynamically generate different parameters based on different expression styles for POS tagging. To evaluate the performance of the proposed method, we compare the method with previous state-of-the-art methods on three different datasets. Experimental results demonstrate that the proposed method can achieve better performance in most cases. We also visualize some parameters learned for the proposed method to demonstrate the motivation for this work.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by China National Key R&D Program (No.2017YFB1002104), National Natural Science Foundation of China (No.61751201, 61532011, 61473092, and 61472088), and STCSM (No.16JC1420401, 17JC1420200).

References

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *NIPS*, pages 137–144.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Rich Caruana and Alexandru Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd ICML*, pages 161–168. ACM.

Bharath Chandrasekaran, Han-Gyol Yi, and W Todd Maddox. 2014. Dual-learning systems during speech category learning. *Psychonomic bulletin & review*, 21(2):488.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. 2016. Dynamic filter networks. In *NIPS*.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *RANLP*, pages 198–206.

Chrisantha Fernando, Dylan Banarse, Malcolm Reynolds, Frederic Besse, David Pfau, Max Jaderberg, Marc Lanctot, and Daan Wierstra. 2016. Convolution by evolution: Differentiable pattern producing networks. In *GECCO*, pages 109–116. ACM.

Eric N Forsyth. 2007. Improving automated lexical and discourse analysis of online chat dialog.

Eric N Forsythand and Craig H Martell. 2007. Lexical and discourse analysis of online chat dialog. In *ICSC 2007.*, pages 19–26. IEEE.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *ACL: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *EMNLP*.

David Ha, Andrew Dai, and Quoc Le. 2016. Hypernetworks. In *ICLR 2016*.

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

Yuheng Hu, Kartik Talamadupula, Subbarao Kambhampati, et al. 2013. Dude, srsly?: The surprisingly formal nature of twitter's language. In *ICWSM*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Marcin Moczulski, Misha Denil, Jeremy Appleyard, and Nando de Freitas. 2015. Acdc: A structured efficient linear layer. *arXiv preprint arXiv:1511.05946*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. ACL.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *EMNLP*.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of WWW 2010*, pages 851–860. ACM.

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). *Technical Reports (CIS)*, page 570.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML-14*, pages 1818–1826.

Jürgen Schmidhuber. 1992. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139.

Jürgen Schmidhuber. 1993. A self-referential weight matrix. In *Proceedings of the International Conference on Artificial Neural Networks, Amsterdam*, pages 446–451.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings NAACL*, pages 173–180.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168*.

Kumanan Wilson and John S Brownstein. 2009. Early detection of disease outbreaks using the internet. *CMAJ*, 180(8):829–831.

Free as in Free Word Order: An Energy Based Model for Word Segmentation and Morphological Tagging in Sanskrit

Amrith Krishna[1], Bishal Santra[1], Sasi Prasanth Bandaru[2]*, Gaurav Sahu[3]*
Vishnu Dutt Sharma[4]†, Pavankumar Satuluri[5]† and Pawan Goyal[1]
[1]Dept. of Computer Science and Engineering, IIT Kharagpur
[2]Flipkart, India [3]School of Computer Science, University of Waterloo
[4]American Express India Pvt Ltd [5]Chinmaya Vishwavidyapeeth
`amrith@iitkgp.ac.in, bsantraigi@gmail.com,`
`pawang@cse.iitkgp.ernet.in`

Abstract

The configurational information in sentences of a free word order language such as Sanskrit is of limited use. Thus, the context of the entire sentence will be desirable even for basic processing tasks such as word segmentation. We propose a structured prediction framework that jointly solves the word segmentation and morphological tagging tasks in Sanskrit. We build an energy based model where we adopt approaches generally employed in graph based parsing techniques (McDonald et al., 2005a; Carreras, 2007). Our model outperforms the state of the art with an F-Score of 96.92 (percentage improvement of 7.06%) while using less than one tenth of the task-specific training data. We find that the use of a graph based approach instead of a traditional lattice-based sequential labelling approach leads to a percentage gain of 12.6% in F-Score for the segmentation task.[1]

1 Introduction

Sanskrit, a morphologically-rich and a free word order language (Kulkarni et al., 2015), poses a series of challenges even for automation of basic processing tasks such as word segmentation. The recent surge in the digitisation efforts for archiving the works ranging from the pre-classical to modern times (Hellwig, 2010-2016) has led to a growing demand for such tools (Goyal et al., 2012; Huet, 2006). We propose a structured prediction approach that jointly solves the word segmentation and morphological tagging tasks for Sanskrit.

The computational problems arising from the mechanical treatment of Sanskrit fall somewhere between speech recognition and the analysis of written text (Huet, 2005). For instance, consider Figure 1a which shows all the phonetically valid word splits for a Sanskrit poetic verse[2]. The written representation in Sanskrit is actually a phonemic stream (Huet, 2005). The constructions often undergo phonetic transformations at the juncture of successive words, similar to what one observes in connected speech (Morris et al., 2004; Shieber and Tao, 2003). These transformations obscure the word boundaries and often modify the phones at these word boundaries. In Sanskrit, these transformations get reflected in writing as well. This is primarily due to the presence of an advanced discipline of phonetics in Sanskrit which explicitly described euphonic assimilation as *sandhi* (Goyal and Huet, 2016). For instance, words prefixed with numbers 14 and 15 in Figure 1a are valid candidates in spite of the phonetic differences they posses from that of the original sentence.

Sanskrit is rich with syncretisms (Crystal, 2011) and homonyms. For example, the surface form '*satī*', prefixed with numbers 6 and 10, are homonyms, while the root '*satya*' generates identical surface form for three different morphological classes leading to syncretism (1 to 3 in Figure 1a). Hence, in addition to segmentation, the morphological analysis of the segmented word forms will be critical for reducing the ambiguity in further downstream tasks such as syntactic analysis. The sentence construction in the language follows weak non-projectivity (Havelka, 2007) permitting the words to have a relatively free word order structure (Kulkarni et al., 2015). The language is all the more lenient for poetic constructions (Scharf et al., 2015; Kulkarni et al., 2015), where arranging the words to adhere to metri-

---

[*] Work done while the authors were at IIT Kharagpur

[†] Part of the work was done while the authors were at IIT Kharagpur

[1] The code and the pretrained edge vectors (§3) used in this work are available at `https://zenodo.org/record/1035413#.W35s8hjhUUs`

[2] A saying from *subhāṣitam* text: One should tell the truth, one should say kind words; one should neither tell harsh truths, nor flattering lies; this is a rule for all times.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2550–2561
Brussels, Belgium, October 31 - November 4, 2018.

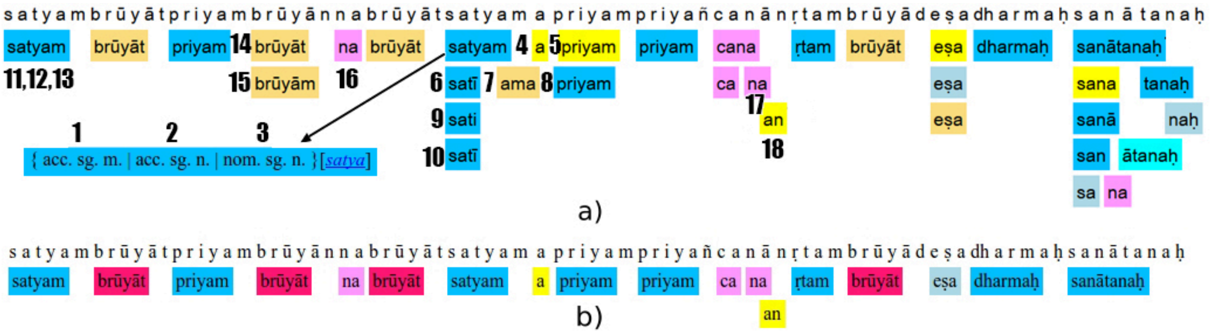

Figure 1: a) All the phonetically valid segmentations (link) for '*satyaṃbrūyātpriyaṃbrūyānnabrūyātsatyamapriyaṃpriyaṃcanānṛtambrūyādeṣadharmaḥsanātanaḥ*' from (*subhāṣitam*) as output by Sanskrit Heritage Reader (SHR) and b) correct segmentation selected from the candidate space.

cal constraints is a bigger concern (Melnad et al., 2013). Hence, the whole input context is desirable when making each prediction at the output (Bahdanau et al., 2015), even for preliminary tasks such as segmentation in Sanskrit (Reddy et al., 2018).

The word splits in Figure 1 are based on the analysis by a lexicon driven analyser, Sanskrit Heritage Reader (SHR)[3]. A total of 1,056 combinations can be formed from the word splits, such that each of those combinations is a solution which covers the entire input. We call such a solution as an '*exhaustive segmentation*'. Out task is to find an '*exhaustive segmentation*', which is also semantically valid. Figure 1b shows the semantically valid solution for the sentence.

We propose our structured prediction framework as an energy based model (LeCun et al., 2006). Considering the free word-orderness, morphological richness and the phenomena of *Sandhi* in Sanskrit, we adopt a graph based treatment for a given input sentence as shown in Figure 2. All the word splits, as predicted by SHR, are treated as the nodes in the graph. Every pair of nodes that can co-occur in at least one '*exhaustive segmentation*'[4] forms directed edges in both the directions. By construction, any subset of nodes that forms a maximal clique will be an '*exhaustive segmentation*'. We formalise our task as the search for a maximal clique. The graph structure eliminates the sequential nature of the input, while the greedy maximal clique selection inference policy of ours can take the entire input context into consideration. We hypothesise that both of these will be beneficial for processing constructions in Sanskrit.

The major contributions of our work are:

1. We propose the first model that performs both word segmentation and morphological tagging for Sanskrit as suggested by Krishna et al. (2017); the combined task reports an F-Score of 90.45.
2. We obtain an F-Score of 96.92 for the word segmentation task, an improvement of 7.06% over the state of the art, a seq2seq model with attention (Reddy et al., 2018).
3. We achieve the results with less than one-tenth of the training data that Reddy et al. (2018) uses, a desirable outcome for a low resource language such as Sanskrit. The pre-training in the form of morphological constraints to form edge vectors enables this.
4. We propose a scheme that uses the Path Ranking Algorithm (Lao and Cohen, 2010) to automate the feature selection and the feature vector generation for the edges. This eliminates the need for manual feature engineering.

2 Proposed Architecture

Given an input construction, we obtain our search space of possible word splits using SHR as shown in Figure 1. The search space represents all the possible exhaustive segmentations with possible gaps and overlaps between the word splits in each of the exhaustive segmentation (Kudo, 2006; Oerder and Ney, 1993; Wolf and Woods, 1977).[5] In such a setting, representing the search space as a lattice (Kudo, 2006; Smith et al., 2005) has been

---

[3] Available at http://sanskrit.inria.fr/, SHR is a lexicon-driven segmenter which produces all the valid word splits. An interface is provided for manual selection of a solution (Goyal and Huet, 2016)

[4] For instance, segments 6 and 7 in Figure 1a are connected, while 6 and 9 are not.

[5] The word splits in an exhaustive segmentation often overlap and sometimes leave gaps by virtue of Sandhi. For examples, please refer the §1 supplementary material.

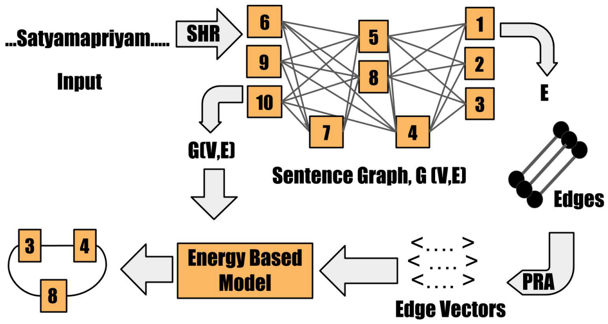

Figure 2: Architecture for the proposed model. Word-splits for '*satyamapriyam*', a sub-sequence of the sentence in Figure 1a are considered here. The nodes are numbered from 1 to 10 and are marked with the same in Figure 1a. For simplicity, we assume that words in nodes 4 to 10 have only one morphological analysis each.

a popular approach for fusional languages (Goldberg and Tsarfaty, 2008; Cohen and Smith, 2007; Hatori et al., 2012). In a lattice, there will be edges only between the adjacent word splits of an exhaustive segmentation. We deviate from this norm in a minor yet fundamental way. In the search space, we choose to add edges between every pair of word splits that are part of a single exhaustive segmentation. Henceforth, we will refer this structure as the sentence graph G. We then employ our minimum cost maximal clique finding energy based model on the sentence graph G. Figure 2 shows the proposed architecture of the model. It shows the sentence graph G for '*satyamapriyam*', a sub-sequence of the sentence in Figure 1a.

Our current design choice results in a denser graph structure as input and a computationally expensive inference. Such a choice requires justification. Currently, there exist digitised versions of texts which spans over a period of 3000 years categorised into pre-classical literature (1500 BCE - 100 BCE), classical literature (300 CE - 800 CE) and modern literature (900 CE to now). Hellwig (2009) assert that the assumption that Sanskrit syntax has remained unchanged over an interval of over 3000 years is not valid. Kulkarni et al. (2015) notes that the constructions in prose generally follow weak non-projectivity (Havelka, 2007; Maier and Lichte, 2011). Kulkarni et al. (2015) also observes that constructions in verses violate weak non-projectivity especially with the adjectival and genitive relations. A large number of texts are written in verses or to complicate things further, they are written as a combination of prose and verse. A lack of consensus among the experts on a common set of rules for works across the different time spans, and the enormous effort in categorising constructions based on their writing styles, motivated us to use this graph construction scheme which is agnostic to word order.

Graph Formation:[6] In Figure 1a, identical surface-forms with the same root are grouped together and displayed as a single entity. But we consider, every unique combination of root, morphological class and the word position in SHR as a separate node in $G(V,E)$. Hence the surface-from *satyam*, appears as 6 separate nodes numbered from 1-3 and 11-13 in Figure 1a. Here the nodes 1-3 differ from each other in terms of their morphological classes. The nodes 1 and 11 differ only in terms of their position owing to its repeated occurrence in the input. The position information is opaque to our proposed system and is used only in forming the nodes for the sentence graph. During the inference, we consider all the pairwise potentials as contexts for each of the prediction made in the search space. The edges in our model should capture the likeliness of two nodes to co-occur in the final solution. Hence, every pair of nodes that can co-occur in an 'exhaustive segmentation' forms two directed edges, one each at either of the directions.

Energy Based Model (EBM) Architecture: Our approach is inspired from the graph based parsing approaches employed generally for dependency parsing (McDonald et al., 2005b; Carreras, 2007) and follows a likewise structured prediction paradigm (Taskar et al., 2005). Specifically, we use an EBM where we model our joint task as search for a maximal clique with minimum energy. Learning consists of finding an energy function that associates lower energies to cliques with increasing similarity to the correct clique. The correct clique configuration will have the lowest energy (LeCun et al., 2006). The inference policy, a maximal clique selection algorithm, is used to make the predictions.

We follow an approach similar to the arc-factored approaches in graphs (McDonald et al., 2005b; Carreras, 2007), where the total energy of a maximal clique, $T_i = (V_{T_i}, E_{T_i})$, is decomposed as the summation of energies of its

---

[6] Our graph construction approach is explained using finite state methods in §1 of the supplementary material

edges (Ishikawa, 2011).

$$\mathcal{S}(T_i) = \sum_{e_{pq}\in E_{T_i}} \mathcal{S}(\vec{e}_{pq})$$

where, $V_{T_i} \subseteq V, E_{T_i} \subseteq E$. The edges are featurised. For an edge $e_{pq} \in E$, the features are represented as a vector, denoted by $\vec{e}_{pq}$. For a given edge, the energy function, $\mathcal{S}(\cdot) : [0,\infty)^{|\vec{e}|} \rightarrow (-\infty,\infty)$, takes the edge feature vector and produces a scalar value as the energy assignment.

Loss Function and Training: We use Hinge Loss (Altun et al., 2003; Taskar et al., 2003) as our loss function. The hinge loss is formulated such that it increasingly penalises those cliques, sampled by our inference algorithm, with increasingly more number of wrong segmentation candidates. We minimise the hinge loss L which is defined as

$$L = max(0, \mathcal{S}(T_{GT}) - \operatorname*{argmin}_{T_i\in A^Q}(\mathcal{S}(T_i) - \Delta(T_i))$$

Here, A^Q denotes the set of all the unique maximal cliques and T_{GT} denotes the maximal clique corresponding to the ground truth.

The margin $\Delta(T_i)$ is defined as $\Delta(T_i) = |V_{T_i} - V_{GT}|^2$. We minimise the given loss function using gradient descent method. The network parameters are updated per sentence using back-propagation. The hinge loss function is not differentiable at the origin. Hence, we use the subgradient method to update the network parameters (Socher et al., 2010; Ratliff et al., 2007). We use a multi-layer perceptron network with a single hidden layer and a leaky ReLU activation function at the hidden layer for the training.

Inference Policy: For the maximal clique selection, we use a greedy heuristic approach inspired from Prim's algorithm (Prim, 1957). The policy is described in Algorithm 1.

In Algorithm 1, we start the clique selection with a single node. At any given instance, we loop through the nodes in the graph which are not yet part of the clique. We add a vertex v to the clique if the cumulative score of all the edges from v to every vertex that is already in the clique is the minimum. We discard all the nodes which are conflicting with vertex v. "Conflicting" nodes are any pair of nodes which are not connected by an edge between them. This follows from the construction of the graph G, as the non-connectivity between the nodes implies that they are proposed as alternative word suggestions in G. As guaranteed by our sentence graph construction, we obtain the maximal clique (*exhaustive segmentation*) when there exist no more vertices to loop through. We perform this for every node in the graph G. From all the cliques so obtained we select the maximal clique with the least score. The approach does not guarantee enumeration of all the cliques, but it is guaranteed that every node will be covered by at least one maximal clique. The heuristic can be seen as a means of sampling some potential minimum energy maximal cliques for the learning task. Energy based models do not require proper normalisation of the solution space (LeCun et al., 2006), a choice that enables the use of the heuristic.

Algorithm 1: Greedy maximal clique selection heuristic

```
1 for each node $v_i$ in $V$ do
2     Initialize a graph $K_i(V_{K_i}, E_{K_i})$ with
        $K_i = G$ such that $V_{K_i} = V$ and
        $E_{K_i} = E$. Initialise a vertex set $V_{T_i}$ with
        $v_i$ as the only element in it. Remove all
        the vertices which are conflicting with $v_i$
        from $K_i$.
3     Add the vertex $v_j \in (V_{K_i} - V_{T_i})$ to $V_{T_i}$,
        such that in $K_i$, the sum of edge weights
        for the edges starting from $v_j$ to all other
        vertices in $V_{T_i}$ is minimum.
4     Remove all the vertexes which are
        conflicting with $v_j$ from $V_{K_i}$.
5     Repeat steps 3 - 4 till $V_{K_i} - V_{T_i} = \varnothing$
6 end
```

During inference, the greedy clique selection heuristic is performed for every node in G. Though the run-time for this inference is polynomial, it can still be computationally expensive. But, in practice we find that our inference procedure results in faster output for graphs with > 19 nodes in comparison to the exponential time Bron-Kerbosch algorithm (Tomita et al., 2006; Bron and Kerbosch, 1973) for clique enumeration (McDonald et al., 2005a). We further improve the run time of our inference procedure by paralleling the clique selection procedure for each node on a separate thread.

3 Feature Generation for the Edges

Given two non-conflicting nodes in G, there exists a pair of directed edges, one each in either direction. For every edge in the sentence graph G, we need to generate features that capture the distributional information between the candidate nodes that the edge connects. Similar in spirit to Bilmes and Kirchhoff (2003) and Krishna et al. (2016), we condition the distributional information based on different morphological constraints to enrich the context. The distributional information is obtained from a morphologically tagged corpus $\mathcal{C}$. Let $\mathcal{V}_w$, $\mathcal{V}_m$ and $\mathcal{V}_r$ be the vocabulary of the inflected word forms, morphological classes and the roots respectively in the corpus. Let $\mathcal{V} = \mathcal{V}_w \cup \mathcal{V}_m \cup \mathcal{V}_r$. For each $n_i, n_j \in \mathcal{V}$, the conditional probability is calculated as $P_{co}(n_j|n_i) = \frac{count(n_j,n_i)}{count(n_i)}$. Here $count(\cdot)$ represents the count of co-occurrence between the entries in the corpus. Also, let MC be the set of morphological constraints used for conditioning the distributional information. Now, for each $n_i, n_j \in \mathcal{V}$ and each $g_k \in MC$, we can obtain the feature value as follows:

$$P_{g_k}(n_j|n_i) = -log(P_{co}(n_j|g_k) \times P_{co}(g_k|n_i))$$

We use the following scheme for feature generation. A node in G essentially contains three attributes, namely, the root, the morphological class and the inflected word form. A feature uses corpus evidence of exactly one of the three attributes. For instance, consider two candidate nodes o_1, o_2 in G with (o_{1w}, o_{1m}, o_{1r}) and (o_{2w}, o_{2m}, o_{2r}) as the respective 3-tuple attributes. Now, one such possible feature value for the edge from o_1 to o_2 can be calculated as $P_{g_1}(o_{1r}|o_{2w})$ where $g_1 \in MC$ and $o_{1r}, o_{2w} \in \mathcal{V}$. Hence, features for a directed edge connecting two different nodes in G can be generated in $3 \times |MC| \times 3$ ways.

We automate the process of feature generation and feature selection using the framework of Path Ranking Algorithm (Lao and Cohen, 2010). Formalising our approach using PRA leads to an efficient and scalable implementation of our scheme. In PRA, enumerating and generating all the possible features needs to be performed only for a sampled set of data pairs from the corpus. By using a supervised feature selection approach, a relevant sub-set of features is filtered. This is a one-time process (Gardner and Mitchell, 2015). During inference, the feature values are computed only for the filtered features. [7]

Morphological Constraints: MC is defined as the set of grammatical category combinations, each combination falling into one of the following two descriptions. a) *Complete combination*, i.e., a morphological class – In Sanskrit, similar to Czech (Smith et al., 2005), a morphological class represents a certain combination of grammatical categories. For instance, a noun is represented by case, gender and number. Hence, the combination 'genitive-masculine-singular' forms a morphological class. b) *Partial combination* - A combination of grammatical categories, which can form a morphological class by adding one or more categories to it. For instance, 'genitive-masculine' is a partial combination that denotes all the possible (three) morphological classes which differ from each other only in terms of the category 'number'. However, 'genitive-present tense' is not a 'valid' combination as it can never form a valid morphological class. The evidence for a partial combination in the corpus $\mathcal{C}$ can be obtained by summing the evidence of all the morphological classes which it denotes. We obtain a total of 528 morphological constraints. A filtered set of 1500 features (out of 4752) is used in our model. Mutual Information Regression (Kraskov et al., 2004) with the word to word co-occurrence probability as label is used for feature selection.[8]

4 Experiments

Dataset: We use the Digital Corpus of Sanskrit (DCS) (Hellwig, 2010-2016), a morphologically tagged corpus of Sanskrit, for all our experiments. DCS contains digitised works from periods that span over 3000 years and contain constructions written in prose or poetry. Identifying sentence boundaries in Sanskrit constructions is a challenge of its own (Hellwig, 2016). DCS currently has split the corpus into more than 560,000 text lines, all of which need not be following explicit sentence boundaries. Krishna et al. (2016) identify 350,000 constructions from the DCS fit for the segmentation task. They use a separate set of 9,577 constructions from the DCS, called as 'DCS10k', and use it as the test set. They ignore the remaining text lines from DCS due to ambiguities either in the provided tagging or alignment with SHR (Krishna et al., 2017). We use the 350,000 constructions used in Krishna et al.

---

[7] The edge vector formation is explained in terms of Metapaths (Sun, 2010) in §3 of the Supplementary.

[8] For different settings we experimented with, for the vector generation, refer to §4 of the supplementary material.

(2016) as the corpus $\mathcal{C}$ (§3) for the generation of our edge vectors. 'DCS10k' was neither used in the training of our model, nor in the generation of edge vectors. Reddy et al. (2018) report their results on a subset of DCS10k containing 4,200 sentences, which we will refer to as 'DCS4k'.

We experiment with the following systems:

SupervisedPCRW: Proposed in Krishna et al. (2016), this model also uses the graph output from SHR. Using PCRW (Lao and Cohen, 2010; Meng et al., 2015), feature vectors for edges are generated using hand-crafted morphological constraints. Starting with the longest word in the graph, the prediction is performed by greedily selecting the candidates as per the edge weights.

EdgeGraphCRF: This is a second order CRF Model (Müller and Behnke, 2014; Ishikawa, 2011) which uses the sentence graph structure G as the input to the system. Every node is represented with fastText (Bojanowski et al., 2017) word embeddings trained under default settings. The edges are featurised with the PRA vectors (§3). We used 1-slack structured SVM for training. For the binary class problem, QPBO (Rother et al., 2007) inference approach provided the best results.

Seq2Seq - Reddy et al. (2018) uses an Encoder-Decoder framework with LSTM cells for the segmentation task. We consider two models from the work, namely, '*segSeq2Seq*' and '*attnSegSeq2seq*' as our baselines. The later which uses attention (Bahdanau et al., 2015) is the current state of the art in Sanskrit word segmentation.

Lattice-EBM: An energy based sequence labelling model, where the input is a lattice (Wolf and Woods, 1977) similar to that of Kudo (2006). The model can be seen as a special case of Graph Transformer Networks (LeCun et al., 1998, 2007). In the lattice structure, the candidate links only to its adjacent nodes in an exhaustive segmentation. We also generate edge vectors for the dummy nodes that act as the start and end markers in the lattice. During prediction, we have to find the best path from the lattice which minimises the sentence score. Here, we consider two variants of Lattice-EBM. *L-EBM-Vanilla* uses the discriminative forward training approach (Collobert et al., 2011) with the standard hinge loss. The second variant *L-EBM-Beam*, uses multi-margin loss (Edunov et al., 2018), instead of the hinge loss. Here, we employ beam search to generate multiple candidates as required by the loss.

Tree-EBM: The baseline model works exactly the same as the proposed model where the only difference between both is in the inference algorithm used. Tree-EBM has an inference that searches for a Steiner Tree (Takahashi, 1980) from the input graph $G(V, E)$, the structure of which is described in §2.[9] The inference procedure outputs a spanning tree that covers a subset of the nodes from G. This raises a challenge while estimating the loss as, unlike in the case of a clique, there can be multiple rooted tress that spans the subset of nodes in the ground truth. In this model, we augment the loss L_{tree} such that the Steiner tree with the least energy that spans the nodes in ground truth is chosen.

$$\begin{aligned} L_{tree} = max(0, & \operatorname*{argmin}_{T_m \in A^G} \mathcal{S}(T_m) \\ & - \operatorname*{argmin}_{T_i \in A^Q} (\mathcal{S}(T_i) - \Delta(T_i)) \end{aligned}$$

Here A^G represents set of all the trees that spans the nodes in the ground truth.

Clique-EBM: The proposed model. The EBM model uses the maximal clique selection heuristic for the inference.

Tasks and Evaluation Measures: We use the macro-averaged Precision (P), Recall (R), F1-score (F) and also the percentage of sentences with perfect matching (PM) as our evaluation metrics. We evaluate the competing systems on the following two different tasks.

*Word Prediction Task **(WPT)*** - The word segmentation task is evaluated based on the correctness of the inflected word forms predicted. This was used in Krishna et al. (2016).

*Word++ Prediction Task **(WP3T)*** - This is a stricter metric for the evaluation of the joint task of segmentation and morphological tag prediction. It evaluates the correctness of each of the inflected word form, lemma and its morphological tag.

4.1 Results

Table 1 provides the results for the best performing configurations for each of the systems. The results for WPT on DCS4k and WP3T on DCS10k for each of the systems are shown in Tables 1.A and 1.B. The proposed model, Clique-EBM (System 8), outperforms all the other models across all the 4 metrics on both the tasks. Clique-EBM shows an

---

[9] The inference procedure is given in §2 of the supplementary material

No:	System	P	R	F	PM
1	segSeq2Seq	73.44	73.04	73.24	29.2
2	SupervisedPCRW	76.30	79.47	77.85	38.64
3	EdgeGraphCRF	79.27	81.6	80.42	35.91
4	L-EBM-Vanilla	86.38	85.49	84.29	53.62
5	L-EBM-Beam	86.38	85.77	86.07	60.32
6	AttnsegSeq2Seq	90.77	90.3	90.53	55.99
7	Tree-EBM	89.44	92.35	90.87	61.72
8	**Clique-EBM**	**96.18**	**97.67**	**96.92**	**78.83**

Table 1.A: WPT on DCS4k

System	P	R	F	PM
EdgeGraphCRF	76.69	78.74	77.7	31.82
LatticeEBM-Vanilla	76.88	74.76	75.8	27.49
LatticeEBM-Beam	79.41	77.98	78.69	31.57
Tree-EBM	82.35	79.74	81.02	32.88
Clique-EBM	**91.35**	**89.57**	**90.45**	**55.78**

Table 1.B: WP3T on DCS10k

Table 1: Performance evaluation of the competing systems in ascending order of their F-Score.

improvement of 7.06% and 40.79% in terms of F-score and *perfect matching* from the current state of the art (System 6) in WPT. Currently there exists no system that predicts the morphological tags for a given word in Sanskrit. For WP3T, Clique-EBM has shown a percentage increase of 11.64% and 69.65% in F-Score and the perfect matching score from Tree-EBM, the next best system.

All the systems except 1 and 6 in Table 1.A use the linguistically refined output from SHR as their search space to predict the final solution. Out of which 3, 4, 5, 7 and 8 use the edge vectors, which encodes the morphological constraints refined using PRA (Lao and Cohen, 2010), generated by the method discussed in §3. As a result these systems require <10% training data than required by system 6. System 3 was trained on 10,000 sentences, while systems 4 and 5 were trained on 9,000 sentences after which the models got saturated. Systems 7 and 8 were trained on 8,200 sentences which is 7.66% of the training data (107,000) used in System 6. In terms of training time, Reddy et al. (2018) reports a training time of 12 hours on a GPU machine, while systems 7 and 8 take a training time of 8 hours on an Intel Xeon CPU based machine with 24 cores and 256 GB RAM.[10] For systems 4 and 5 it takes roughly 4 hours to train on the same machine. There was no training involved for the prediction of segmentations in system 2.

In systems 4 and 5, the inference is performed sequentially from left to right[11]. The use of beam search with multi margin in System 5 resulted in marginal improvements (<2) in terms of F-Score to that of system 4. Further, the improvement in the results saturated after a beam size of 128. System 3 being a second order CRF model (Ishikawa, 2011), does not take the entire sentence context into account. In fact, about 85.16% of the sentences predicted by the model from DCS10K do not correspond to an '*exhaustive segmentation*'. Prediction of an '*exhaustive segmentation*' is guaranteed in all our EBM models 4, 5, 7 and 8 (also in system 2) by virtue of the inference procedure we use. Both systems 7 and 8, along with System 6 which uses attention, consider the entire input context when making each prediction. System 8 considers all the pairwise potentials between the nodes while making a prediction, but System 7 does not (Steiner tree vs. maximal clique).

Figure 3 reports the performance of the systems 2, 5, 6 and 8 where the sentences in DCS4k[12] are categorised based on the number of words presented in the segmented ground-truth solution. Our proposed system Clique-EBM performs the best across all the lengths with an exception towards shorter constructions of 2 words or less. Interestingly, both the sequential models (systems 5 and 6) show a decreasing trend as the number of words increases, while the Clique-EBM model shows an increasing trend with a larger length, which might indicate that more context helps the model. In fact, the greedy yet non-sequential approach used in Krishna et al. (2016) outperforms both the sequential models at longer lengths. The average length of a sentence in DCS is 6.7 (Krishna et al., 2016), the share of sentences with seven or more words is 62.78%.

4.2 Fine Grained Analysis on Clique-EBM[13]

Pruning the edges in sentence graph G: Constructions in Sanskrit follow weak non-projectivity (with exception to verses), implying that they adhere to the principle of proximity (Kulkarni et al., 2015). By proximity we expect that the words in a phrase go together, without being interrupted by a syntactically unrelated word.

---

[10] Please refer §4 of the supplementary for wall time analysis. System 6, when trained on this CPU based system, did not converge even after 15 hours of training.

[11] We also tried reversing the input effectively enabling the right to left direction but the results were worse than the reported system by an F-Score of 3.

[12] sentences with length more than 12 words are not shown as such sentences appear less than 10 times in DCS4k.

[13] For hyper-parameter settings, and other fine-grained analysis refer to §4 of the supplementary material.

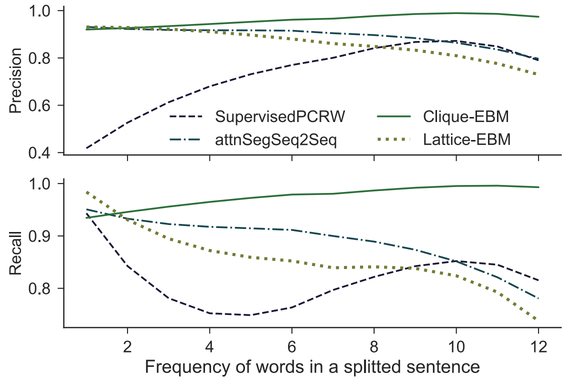

Figure 3: Performance of the competing systems for DCS4k grouped on the word counts in the ground-truth

k	WPT			WP3T		
	P	R	F	P	R	F
5	90.46	92.27	91.36	83.52	80.48	81.97
10	92.92	95.07	93.98	85.32	84.4	84.86
15	94.85	96.14	95.49	87.67	86.38	87.02
20	**95.23**	**96.49**	**95.86**	**89.25**	**88.62**	**88.93**

Table 2: Performance of Clique-EBM with pruned edges in G.

Type	WPT Recall		WP3T Recall	
	T-EBM	C-EBM	T-EBM	C-EBM
Noun	93.06	96.87	86.14	89.0
Verb	89.14	95.91	87.38	94.42
Compound	89.35	93.52	86.01	91.07
Indeclinable	95.07	97.09	94.93	96.47

Table 3: System performance on the coarse level POS for the competing systems Clique-EBM (C-EBM) and Tree-EBM (T-EBM)

But the relative ordering between the phrases in a construction and the order of words within a phrase can be free.

For any two words appearing in an exhaustive segmentation, we keep an edge only if both the words overlap within a distance of k characters. We experiment with k = 5, 10, 15 and 20. Hence, for $K = 20$, a word will form edges with all the words that fall within 20 characters to left and 20 characters to right. The average length of an input sequence in DCS10K is 40.79 characters. We do not modify our inference procedure in system 8 other than to take care of the possibility that a clique need not always be returned. Table 2 shows the results for different values of k. Interestingly, the results show a monotonic increase with the increase in context window, and the results with the entire context are still better than those with $k = 20$, even though only marginally. It is interesting to note that, keeping the entire context does not adversely affect the predictions as none of the pruned models outperforms System 8.

The lattice structure can be seen as an extreme case of pruning. We modify System 4 to use a non-sequential inference procedure, adapted from System 7. Here, the start and end markers were removed. Additionally, a given connected node pair has 2 edges, each in either of the directions. We find that the model gives an F-Score of 87.4 which outperforms System 4 by more than three points.

Out of Vocabulary words: As described in §3, the distributional information from DCS is used as the corpus $\mathcal{C}$ for the feature generation. For the case of OOV in roots ($\mathcal{V}_r$), we use add-1 smoothing. But, for the case of OOV in inflections ($\mathcal{V}_w$) we find that using the evidence from corresponding root of the candidate is beneficial. DCS10k has 8,007 roots of which 514 are OOV and 833 occur only 1 to 5 times. The micro-averaged F-Score for these are 57.98 and 72.87, respectively.

Morphological class specific assessment : Table 3 presents the micro-averaged recall for the words grouped based on their parts of speech (POS) for Clique-EBM and Tree-EBM. Both the systems follow similar trends and the morphological class misprediction is highest among the nouns and compounds (WP3T Recall). It also needs to be noted that the improvement made by Clique-EBM in comparison to Tree-EBM for WP3T was also on prediction of noun and compound morphological classes. Also in Tree-EBM, the mispredictions in compounds were mostly cases of the system getting the compound components confused to one of the morphological classes in nouns.

We find that considering the pairwise potential between all the words in a sentence in Clique-EBM led to improved morphological agreement between the words in comparison to Tree-EBM. In Tree-EBM, the top 5 cases of mispredictions from one morphological class to a particular wrong class were between those classes of nouns that differed in exactly one of the three possible grammatical categories, namely gender, number or declension, that makes up a noun. In Clique-EBM such patterns were not anymore present and more importantly the skewedness in such mispredictions were considerably reduced.[14]

Summarily, our non-sequential method of inference results in better performance in comparison

---

[14] Please refer to Tables 6 and 7 in the supplementary material for details

to the sequential models. We also find that the sequential models see a drop in their performances when the number of words in a sentence increases. Leveraging the pairwise potentials between every connected nodes while making a prediction improves the performance. The performance gain of Clique-EBM over Tree-EBM illustrates the effectiveness of this approach.

5 Discussion

In Sanskrit, syncretism (Crystal, 2011) leads to ambiguity during morphological analysis. It can further be observed that such common root identical surface forms often have one or more common grammatical categories in their morphological classes (Goyal and Huet, 2016). We find that the first three models in Table 1.B often end up predicting an identical surface-form with an incorrect morphological tag, thus affecting the WP3T scores.[15]. The grammatical categories in a morphological class are indicative of the syntactic roles and the morphological agreement between the words in a construction. We empirically observe that the inference procedure for clique-EBM, which considers the entire input context and pairwise potentials between the candidates, helps in improving the performance of the model. A similar observation regarding incompatible morphological agreements between predicted words was made by Hassan et al. (2018) for their NMT model. The authors introduced an elaborate 2 phase decoder and a KL-Divergence based regularisation to combat the issue.

The energy based model (EBM) we propose is a general framework for structured prediction in Sanskrit. EBMs are widely used for various structured prediction tasks (LeCun et al., 2007; Belanger and McCallum, 2016). Belanger (2017) states that, "CRFs are typically attributed to Lafferty et al. (2001), but many of the core technical contributions of the paper appeared earlier in LeCun et al. (1998)." GTNs (LeCun et al., 1998), in fact, work on a graph based input very similar to that of a lattice, a variant of which we use in L-EBM-Vanilla. For dependency parsing, use of a word-level lattice structure similar to Seeker and Çetinoğlu (2015), where all the homonyms and syncertisms of a given surface-form form a lattice, will potentially result in a reduced candidate space than ours. Additionally, our model currently does not take into account the phrasal nature of compounds in Sanskrit (Lowe, 2015). This can further reduce the edge density in our current graph construction. But, this needs further exploration, as current edge vectors may not be suitable for the task. To generate the possible candidates, we rely completely on the SHR. In case of words not recognised by the lexicon driven SHR, analysis of sentences with a partially recognised portion is still possible. Once a root is added, all its inflections can be generated by the SHR automatically.

6 Conclusion

We proposed a novel approach to tackle word segmentation and morphological tagging problem in Sanskrit. Our model outperforms Reddy et al. (2018), the current state of the art, with an F-Score of 96.92. Reddy et al. (2018) report that the extension of their model to perform morphological tagging is not straightforward, as they learn a new sub-word vocabulary using the sentencepiece model (Schuster and Nakajima, 2012).

The free word order nature of the language motivated us to consider the input to be a graph so as to avoid the sequential processing of input. For the EBM we use, there is no requirement of proper normalisation (LeCun et al., 2006). We benefit from this as we perform a search in the space of complete outputs and there is a combinatorial explosion in the output space for a linear increase in the input space (Doppa et al., 2014). The pretraining of the edge vectors with external knowledge in the form of morphological constraints is effective in reducing the task specific training size (Yang et al., 2017; Andor et al., 2016).

Acknowledgements

We are grateful to Oliver Hellwig for providing the DCS Corpus and Gérard Huet for providing the Sanskrit Heritage Engine, to be installed at local systems. We extend our gratitude to Amba Kulkarni and Rogers Mathew, along with Gérard for helpful comments and discussions regarding the work. We thank the anonymous reviewers for their constructive and helpful comments, which greatly improved the paper. We are indebted to Unni Krishnan T A for his contributions towards implementation of the framework.

---

[15] Details in §4 of the Supplementary material

References

Yasemin Altun, Mark Johnson, and Thomas Hofmann. 2003. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Sapporo, Japan. Association for Computational Linguistics.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representation (ICLR)*, San Diego, US.

David Belanger. 2017. *Deep Energy-Based Models for Structured Prediction*. Ph.D. thesis, University of Massachusetts Amherst.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 983–992, New York, New York, USA. PMLR.

Jeff A. Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Companion Volume of the Proceedings of HLT-NAACL 2003 - Short Papers*, Edmonton, Canada. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Coen Bron and Joep Kerbosch. 1973. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 957–961, Prague, Czech Republic. Association for Computational Linguistics.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 208–217, Prague, Czech Republic. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

David Crystal. 2011. *A dictionary of linguistics and phonetics*, volume 30. John Wiley & Sons.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2014. Hc-search: a learning framework for search-based structured prediction. *Journal of Artificial Intelligence Research*, 50(1):369–407.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364. Association for Computational Linguistics.

Matt Gardner and Tom Mitchell. 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, Lisbon, Portugal. Association for Computational Linguistics.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL-08: HLT*, pages 371–379, Columbus, Ohio. Association for Computational Linguistics.

Pawan Goyal and Gerard Huet. 2016. Design and analysis of a lean interface for sanskrit corpus annotation. *Journal of Language Modelling*, 4(2):145–182.

Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. 2012. A distributed platform for sanskrit processing. In *Proceedings of COLING 2012*, pages 1011–1028. The COLING 2012 Organizing Committee.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1045–1053, Jeju Island, Korea. Association for Computational Linguistics.

Jiri Havelka. 2007. Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures. In *Proceedings of the 45th Annual Meeting of the Association of Computational*

Linguistics, pages 608–615, Prague, Czech Republic. Association for Computational Linguistics.

Oliver Hellwig. 2009. Extracting dependency trees from sanskrit texts. In *Sanskrit Computational Linguistics*, pages 106–115, Berlin, Heidelberg. Springer Berlin Heidelberg.

Oliver Hellwig. 2010-2016. *DCS - The Digital Corpus of Sanskrit*. Berlin.

Oliver Hellwig. 2016. Detecting sentence boundaries in sanskrit texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 288–297. The COLING 2016 Organizing Committee.

Gérard Huet. 2005. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *Journal of Functional Programming*, 15(4):573–614.

Gérard Huet. 2006. *Themes and Tasks in Old and Middle Indo-Aryan Linguistics*, chapter Lexicon-directed Segmentation and Tagging of Sanskrit. Motilal Banarsidass, Delhi.

Hiroshi Ishikawa. 2011. Transformation of general binary mrf minimization to the first-order case. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(6):1234–1249.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. 2004. Estimating mutual information. *Physical review E*, 69(6):066138.

Amrith Krishna, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu, Yajuvendra Singh, and Pawan Goyal. 2016. Word segmentation in sanskrit using path constrained random walks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 494–504. The COLING 2016 Organizing Committee.

Amrith Krishna, Pavan Kumar Satuluri, and Pawan Goyal. 2017. A dataset for sanskrit word segmentation. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 105–114, Vancouver, Canada. Association for Computational Linguistics.

Taku Kudo. 2006. Mecab: Yet another part-of-speech and morphological analyzer. *http://mecab. sourceforge. jp*.

Amba Kulkarni, Preethi Shukla, Pavankumar Satuluri, and Devanand Shukl. 2015. How Free is free Word Order in Sanskrit. In *The Sanskrit Library, USA*, pages 269–304.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Mach. Learn.*, 81(1):53–67.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. 2006. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press.

Yann LeCun, Sumit Chopra, Marc'Aurelio Ranzato, and Fu-Jie Huang. 2007. Energy-based models in document recognition and computer vision. In *Proceedings of International Conference on Document Analysis and Recognition (ICDAR)*, pages 337–341, Curitiba, Paraná, Brazil. IEEE Computer Society.

John J. Lowe. 2015. The syntax of sanskrit compounds. *Journal of Historical Syntax*, 91(3):71–115.

Wolfgang Maier and Timm Lichte. 2011. Characterizing discontinuity in constituent treebanks. In *Formal Grammar*, pages 167–182, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. 2005a. Simple algorithms for complex relation extraction with applications to biomedical ie. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 491–498. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Keshav S Melnad, Pawan Goyal, and Peter Scharf. 2013. Meter identification of sanskrit verse. *The Sanskrit Library, USA*.

Changping Meng, Reynold Cheng, Silviu Maniu, Pierre Senellart, and Wangda Zhang. 2015. Discovering meta-paths in large heterogeneous information networks. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 754–764, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Andrew Cameron Morris, Viktoria Maier, and Phil Green. 2004. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition. In *Eighth International Conference on Spoken Language Processing*.

Andreas C Müller and Sven Behnke. 2014. Pystruct: learning structured prediction in python. *Journal of Machine Learning Research*, 15(1):2055–2060.

M. Oerder and H. Ney. 1993. Word graphs: an efficient interface between continuous-speech recognition and language understanding. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 119–122.

Robert Clay Prim. 1957. Shortest connection networks and some generalizations. *Bell Labs Technical Journal*, 36(6):1389–1401.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2007. (online) subgradient methods for structured prediction. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*, pages 380–387, San Juan, Puerto Rico. JMLR.org.

Vikas Reddy, Amrith Krishna, Vishnu Sharma, Prateek Gupta, Vineeth M R, and Pawan Goyal. 2018. Building a Word Segmenter for Sanskrit Overnight. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. 2007. Optimizing binary mrfs via extended roof duality. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, pages 1–8, Minneapolis, Minnesota, USA. IEEE Computer Society.

Peter Scharf, Anuja Ajotikar, Sampada Savardekar, and Pawan Goyal. 2015. Distinctive features of poetic syntax preliminary results. *Sanskrit syntax*, pages 305–324.

M. Schuster and K. Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Wolfgang Seeker and Özlem Çetinoğlu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *Transactions of the Association for Computational Linguistics*, 3:359–373.

Stuart M Shieber and Xiaopeng Tao. 2003. Comma restoration using constituency information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 142–148. Association for Computational Linguistics.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 475–482, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *Coling 2010: Posters*, pages 1211–1219, Beijing, China. Coling 2010 Organizing Committee.

Hiromitsu Takahashi. 1980. An approximate solution for steiner problem in graphs. *Math. Japonica*, 24(6):573–577.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 896–903, New York, NY, USA. ACM.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin markov networks. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 25–32, Cambridge, MA, USA. MIT Press.

Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. 2006. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42.

J. Wolf and W. Woods. 1977. The hwim speech understanding system. In *ICASSP '77. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 784–787.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849, Vancouver, Canada. Association for Computational Linguistics.

A Challenge Set and Methods for Noun-Verb Ambiguity

Ali Elkahky, Kellie Webster, Daniel Andor, and Emily Pitler
Google AI Language
`{alielkahky, websterk, andor, epitler}@google.com`

Abstract

English part-of-speech taggers regularly make egregious errors related to noun-verb ambiguity, despite having achieved 97%+ accuracy on the WSJ Penn Treebank since 2002. These mistakes have been difficult to quantify and make taggers less useful to downstream tasks such as translation and text-to-speech synthesis. This paper creates a new dataset of over 30,000 naturally-occurring non-trivial examples of noun-verb ambiguity. Taggers within 1% of each other when measured on the WSJ have accuracies ranging from 57% to 75% accuracy on this challenge set. Enhancing the strongest existing tagger with contextual word embeddings and targeted training data improves its accuracy to 89%, a 14% absolute (52% relative) improvement. Downstream, using just this enhanced tagger yields a 28% reduction in error over the prior best learned model for homograph disambiguation for text-to-speech synthesis.

1 Introduction

Whether a word is functioning as a noun or a verb in a particular linguistic context critically affects the output of tasks including translation and text-to-speech synthesis. The English word *close* may be translated as either *nah* (adjective/non-verb) or *schließen* (verb) (example from Sennrich and Haddow (2016)). In text-to-speech, the homograph *lives* is pronounced /laɪvz/ (noun) or /lɪvz/ (verb; example from Sproat et al. (1992)).

While downstream applications require taggers be sensitive to non-local linguistic context, it is difficult to measure such sensitivity with current tagging evaluation. In the past 15 years since Collins (2002), many models have accuracy exceeding 97% when measured on the WSJ Penn Treebank, which is within the level of human inter-annotator agreement for the corpus. Incorporating non-local context via sentence-based representations (Collobert et al., 2011) or state-of-the-art contextual representations of tokens (ELMo, Peters et al. (2018)) yields the same tagging accuracy as Collobert et al.'s limited window-based representation (97.3%). However, existing local models "regularly make egregious errors" (Manning, 2011), notably on imperative detection[1]. That is, the applicability of the part-of-speech labeling task is limited by its standard evaluation not reflecting difficult cases which require contextual reasoning to resolve ambiguity.

In this paper, we address this mismatch by creating a targeted intrinsic evaluation: a challenge dataset of over 30,000 naturally-occurring non-trivial examples of noun-verb ambiguity spanning multiple domains and containing many imperatives that non-expert humans can annotate with high agreement (Section 2). We will publicly release both the training and evaluation data[2].

We further contribute a series of modeling experiments on this data. We first show that state-of-the-art taggers perform poorly on this challenge (Table 1) and then investigate two simple and orthogonal approaches to enhancing a state-of-the-art tagger: incorporating generic contextual embeddings trained on billions of words, and incorporating thousands of examples of training data targeted for this task. Both of these approaches yield large and complementary improvements: the combined methods give an accuracy of 89.1%, a 14% absolute improvement over a state-of-the-art tagger and a 31% absolute improvement over the widely used Stanford tagger. Section 3 provides an overview of the investigated taggers, experiments, and results.

---

[1] In experiments with recipe data in Kiddon et al. (2015), an unsupervised system had an F1 score over 20% higher in absolute terms than supervised taggers.

[2] `http://goo.gl/language/noun-verb`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2562–2572
Brussels, Belgium, October 31 - November 4, 2018.

Model	WSJ	NV
Existing Taggers		
Toutanova et al. (2003)	97.24	57.6
Choi (2016)	97.64	71.2
Dozat et al. (2017)	97.33	70.4
Bohnet et al. (2018)	**98.00**±.12	74.0±1.2
Enhancements		
+ELMo	**97.94**±.08	82.1±0.9
+NV Data	**97.98**±.11	86.4±0.4
+ELMo+NV Data	**97.97**±.09	**88.9**±0.3

Table 1: Empirical Results. All investigated new and existing taggers are within 1% of each other when measured on the WSJ test set. When evaluated on the Noun-Verb dataset, however, existing taggers range from 57% to 74%. Adding enhancements to the Bohnet et al. (2018) tagger gives over 14% absolute improvement. Best results and results insignificantly different from the best are bolded (two-tailed t-test).

Finally, we demonstrate that these tagging improvements make a positive impact on the downstream task of homograph disambiguation for text-to-speech (Section 4).

2 Noun-Verb Dataset

Consider the ambiguous examples below:

(1) Certain insects can damage plumerias, such as mites, **flies**, or aphids. **NOUN**

(2) **Mark** which area you want to distress. **VERB**

All tested existing part-of-speech taggers (Table 1) mistag both of these examples, tagging *flies* as a verb and *Mark* as a noun[3]. Looking at only the WSJ Penn Treebank, all occurrences of *Mark* are nouns, so a part-of-speech tagger that ignores context completely could appear to do quite well on this word type. Similarly, all occurrences of the word type *share* in the WSJ development set are noun instances.

A baseline of selecting the most frequent tag per word type (ignoring all context) achieves 93.0% accuracy on the ambiguous tokens in the WSJ (Table 2). A simple tagger based on a single hidden layer feed-forward neural network with 128 units that uses a three word window around the focus token as features achieves an accuracy of 97.0% on the WSJ ambiguous words (WSJ:NV).

Train	Dev	Type Majority	NN ±3 Words
WSJ:NV	WSJ:NV	93.0	97.0
NV	NV	70.1	77.6

Table 2: Taggers that use no context (Type Majority) or very little context (NN ±3 words) can achieve high accuracies on the ambiguous tokens in the WSJ (WSJ:NV), but would fare much worse on the Noun-Verb dataset.

We therefore aim to create a dataset in which taggers would have to take into account the surrounding context in order to correctly tag ambiguous words, rather than relying on skewed priors per word type. We design a methodology for identifying and labeling hard cases of noun-verb ambiguity. The result is a dataset of over 30,000 hand-labeled, natural, and non-trivial examples of noun-verb ambiguity, which we will make publicly available to facilitate research on modeling for this task.

2.1 Collection Methodology

Our goal is to build a resource which captures a wide range of challenges that a part-of-speech tagger needs to handle in the wild. To produce this resource, we find large sources of *naturally occurring* examples with a diversity of challenges, identify *noun-verb ambiguity*, find the *non-trivial* examples, and finally acquire *high-precision labels* from humans.

2.1.1 Naturally Occurring Sources

All examples come from naturally occurring English web text from three distinct genres. Typical examples from each are shown in Table 3. These genres present a diverse range of challenges: genre 1 has long well-edited sentences, genre 2 makes heavy use of imperative verbs, and genre 3 contains largely headline style short sentences.

2.1.2 Ambiguous Token Detection

We used an online dictionary to identify ambiguous word *types* (such as *play*) that can be either a noun or a verb.[4] To find ambiguous *instances* of these types, we ran a CRF-based tagger similar to Toutanova et al. (2003) over the input sen-

---

[3] The enhanced tagger that uses both contextual word embeddings and data augmentation (+ELMo+NV Data in Table 1) gets both Example (1) and Example (2) correct.

[4] We exclude a short stop list (*do*, *name*, *state*); the final list contains 24,170 word types.

Representative Examples	Label
Genre 1	
“**Man** With a Vision” peaked at #91 in the UK, spending two weeks on the chart.	**NOUN**
40.7% of the population **benefit** from public assistance as of 2004, up from 23.0% in 2000.	**VERB**
Genre 2	
Your doctor may recommend a diet or **exercise** routine.	**NOUN**
Use within 3 days of cooking.	**VERB**
Genre 3	
Safeguard Infrastructure From Electrical **Surges** & Limit Downtime.	**NOUN**
Stop In Today Or Shop Online!	**VERB**

Table 3: Noun and Verb examples from each genre. All examples are taken from the development set.

tences. We selected tokens tagged as either a noun or a verb[5] and for which the k-best list for that token contained both noun and verb[6] tags with close scores. We used a heuristic that the lower scoring tag had to have a score within 20% of the score of the higher.

2.1.3 Filtering Trivial Examples

Part-of-speech tagging is already a well-established task with plenty of existing labeled examples. Adding more examples similar to *John watched a play* would not affect the output predictions of taggers, which already tend to correctly label tokens as nouns if they follow determiners. Inspired by work on active learning (Tomanek and Hahn, 2009; Small and Roth, 2010), we focused our data collection efforts on difficult examples. To remove easy contexts, we excluded tokens preceded by a determiner or modal verb. Tokens[7] were additionally restricted to be neither adjectival modifiers[8] nor components of noun-compounds[9].

2.1.4 Diversification

Noun-verb disambiguation is a challenge for modern POS taggers both because words can look simultaneously noun- and verb-like to a model, but also because verbs (nouns) can falsely present as nouns (verbs). Our extraction methodology is well-designed to identify the former. To identify tokens on which models are falsely confident, we manually reviewed a sample of tokens discarded in extraction. We found that sentence-initial imperative verbs were very likely to be confidently tagged as nouns. To ensure that this important class of ambiguous tokens was included in our dataset, we made it a special extraction case and did not apply the above filters for trivial examples.

Agreement Type	#	%
Unanimous	23,908	71.4%
Majority	9,122	27.3%
Disagreement	432	1.3%

Table 4: Inter-annotator agreement rates. Unanimous examples had 3/3 agreement, while *majority* examples had 2/3, 3/5, or 4/5 in agreement.

2.1.5 Crowdsourced Annotation

We presented annotators with the extracted tokens in their full sentence context. Annotators were asked to select whether the target word was a “Noun”, a “Verb”, “Ambiguous”, or “Neither” (a noun or a verb). Full annotation guidelines will accompany the dataset release. Each example was annotated by at least three annotators for quality assurance. For batches with larger than average proportions of non-unanimous annotations, the non-unanimous examples were sent to an additional two annotators for a total of five annotations. Table 4 shows that annotators generally had a high level of agreement with each other, with unanimous agreement on 71.4% of the examples and majority agreement on 98.7% of the examples. Annotators achieved an average pace of 40 seconds per sentence.

---

[5] Nouns and verbs were identified by mapping the fine-grained part-of-speech tag to its coarse-grained category (Petrov et al., 2012): `https://github.com/slavpetrov/universal-pos-tags/blob/master/en-ptb.map`

[6] We excluded VBN from the set of verb tags, as it often functions more similarly to non-verbs

[7] Specifically, non-sentence initial tokens

[8] Labeled as *amod* according to a dependency parser

[9] Labeled as *nn* according to a dependency parser

Genre	Train	Dev	Test
1	8621	1081	2711
2	6160	919	2289
3	9473	400	1000
All	24254	2400	6000

Table 5: Noun-Verb dataset statistics.

2.2 Final Dataset

To compile the final dataset, we rejected examples in which there was no majority agreement or in which the majority label was "Ambiguous" or "Neither". This excluded 808 sentences and yielded a final dataset size of 32,654. We divided this into training, development, and test sets. Table 5 shows the dataset sizes and genre distributions. The genre distribution of the training set is intentionally different from that of the development and test sets, as realistically one will often have different distributions at training and test time, and future work may want to model this difference (Donmez et al., 2010; Steinhardt and Liang, 2016).

We asked a professional linguist to independently label 200 examples and adjudicate any differences from the crowd-sourced labels with other professional linguists. The linguists found only 7 actual mistakes (3.5% of examples). Of the remaining 96.5% plausible annotations, the linguist agreed with the crowd in 167 cases (83.5%), and found 26 disparities between PTB-style guidelines and plausible intuitive judgments (13%). All but one of the disparities involved a word ending in *"ing"* inside a noun phrase, such as "*Manufacturing* defects"). Also, all but two of the disparities were cases which the crowd source annotators labeled as nouns while the PTB-style guidelines labeled as verbs.

While humans can do well on these instances, Table 2 shows that baseline taggers that use little or no context have high error rates on this dataset, in contrast to the WSJ.

3 Empirical Evaluation of Taggers

In this section, we demonstrate empirically the limitations of several existing taggers on the new challenge dataset. We then take the most accurate, Bohnet et al. (2018), and investigate how it can be enhanced to be much more discriminative in ambiguous contexts. We finish with some error analysis to inspire future work.

3.1 Experimental Setup

Training All experiments used the standard splits of the WSJ Penn Treebank and the new Noun-Verb dataset. Specifically, WSJ Sections 2-21 were used to train all models; where indicated, this was augmented with the training portion of the Noun-Verb dataset. Neural models (Dozat et al. (2017), Bohnet et al. (2018), and extensions) used WSJ Section 22 for early stopping, and were run with $n = 10$ random restarts to compute standard deviations.

Evaluation Models are evaluated on the Noun-Verb test set. The development set was used for developing the proposed enhancements, as well as to do error analysis. To verify performance on the standard task, we also evaluate accuracy on WSJ Section 23, cf. Table 1 first column.

Our evaluation metric is VERB/NON-VERB classification accuracy over tokens which have gold annotations. To evaluate the taggers we map the fine-grained tag output using Petrov et al. (2012): tags with a coarse-grained VERB category map to the VERB label, and all other tags to the NON-VERB label.

3.2 Existing Taggers

We evaluated four commonly used and/or state-of-the-art taggers on our task. The first investigated tagger is the Stanford POS tagger[10] (Toutanova et al., 2003), part of the Stanford CoreNLP Toolkit (Manning et al., 2014) and widely used. This pre-trained model is a log-linear model with features over the surrounding words and tags in a local window around the focus word.

The second investigated tagger is the publicly available *NLP4J*, a pre-trained tagging model (Choi, 2016)[11]. It used feature induction to expand the feature set during training by adding combinations of low-dimensional features. The approach achieved 97.64% on WSJ evaluation. It is worth noting that this model used a large automatically tagged corpus to get ambiguity classes for each word and Choi (2016) showed that this extra piece of information was responsible for the largest part of the improvement.

The third tagger is Dozat et al. (2017), which won the UPOS portion of the CoNLL 2017 Shared Task on Universal Dependencies (Zeman et al.,

---

[10] `https://nlp.stanford.edu/software/tagger.shtml`

[11] `https://github.com/emorynlp/nlp4j`

2017) by a wide margin. It represents each word by a sum of its pretrained word embedding (glove Pennington et al. (2014)), trained word embedding, and the output from an LSTM runs over word's characters. Those representations are supplied to a deep BiLSTM followed by a Multi-Layer Perceptron (MLP) layer. The output from the MLP layer is multiplied by a learned embedding for tags and the tag with the highest score is selected as the output.

Finally the fourth existing tagger is the Meta-BiLSTM (Bohnet et al., 2018) which is the current state of the art on both WSJ and CoNLL 2017 POS tagging evaluation. This model consists of three components, all of which run over the entire input sentence: a word-BiLSTM that takes a sum of pretrained (GloVe (Pennington et al., 2014)) and trained word embeddings, a char-BiLSTM that consumes trained characters embedding and a Meta component that takes a concatenation of word and character representations (at word boundaries) and feeds it to a Bi-LSTM followed by a MLP layer. The final output is computed using softmax over the Meta-MLP representation but a multi-loss is also optimized at the char and word representations level.

For Dozat et al. (2017) and Bohnet et al. (2018), we trained the model on WSJ PTB training data to get comparable models to the two previous systems. For Dozat et al. (2017) we used the default hyperparameters. For Bohnet et al. (2018), the hyperparameters used are almost identical to the original paper.[12]

The first two taggers are linear models (with feature combinations) while the second two are neural models. Both Dozat et al. (2017) and Bohnet et al. (2018) take non-local context into account through BiLSTMs over the full sentence. However, these models might not use this modeling power when trained on the WSJ, since local context is usually sufficient (Table 2).

3.3 Enhancements

We take the best existing tagger (Bohnet et al., 2018) as our starting point to investigate the efficacy of two simple enhancements and their combination for improving noun-verb disambiguation.

The first enhancement is to add generic, contextual word embeddings trained on a billion words of language modeling data (Peters et al., 2018). The second enhancement is to add task-specific targeted training data, with thousands of examples derived from the Noun-Verb training set.

Contextual Word Embeddings (ELMo) The statistics of the new dataset, shown in Table 2, suggest that this dataset might benefit from more contextual modeling. Although the basic Meta-BiLSTM model is already contextual, one can suspect based on the first row in Table 2 that WSJ training might lead the model to ignore wider context. One way to make the model use more contextual information is to replace the word embedding layer with a contextual embedding. We used ELMo embeddings (Peters et al., 2018), which are generated by training a bi-directional language model on a large corpus of unlabeled data. The aim of using ELMo here is that we expect to get different embeddings for a word like "play" when it is used as a verb, as in "I will come and play", versus when it is used as a noun, as in "I liked the two-act play".

We replaced the word embedding layer in the Word component with ELMo.[13] As in Peters et al. (2018), we trained a task specific weighting of the three ELMo layers:

$$v_i^{(\text{word})} = \gamma \sum_{j=0}^{2} s_j \mathbf{h}_{i,j}^{\text{ELMo}}, \qquad (1)$$

where $\mathbf{h}_{i,j}^{\text{ELMo}}$ is the j-th layer ELMo embedding of word i, s_j are softmax-normalized weights over the layers, and γ is a scalar parameter. We trained this model on the WSJ training data only.

Targeted Data Augmentation (NV Data) Our Noun-Verb training data comes with gold binary labels ("Noun" or "Verb"). To add them to our current model, we took a simple approach to map the Noun-Verb labels into the fine-grained POS tagset used in the WSJ dataset. To do that, we ran the baseline tagger used to extract the annotated examples in §2.1 over the Noun-Verb training data, and extracted all possible tags for the annotated words, sorted by their score. We then assigned to that word the highest scoring tag consistent with the coarse-grained tags. This resulted in a silver training dataset containing partially labeled sentences, each with one word tagged by its

---

[12] Two hyperparameter differences: we used two layers instead of three for the word component and a learning rate decay of 0.99994 instead of 0.999994. These were fixed early on and not tuned.

[13] We used the "Original" model from `https://allennlp.org/elmo`.

Model	SI	¬ SI
Majority class per word type using Noun-Verb training set	74.6	69.3
Existing Taggers		
Toutanova et al. (2003)	47.4	59.6
Choi (2016)	67.8	71.0
Dozat et al. (2017)	68.3	70.7
Bohnet et al. (2018)	68.4±4.0	74.4±0.9
Enhancements		
+ELMo	73.4±2.2	82.1±1.0
+NV Data	**89.3**±0.5	85.4±0.5
+ELMo+NV Data	**90.0**±0.8	**87.6**±0.6

Table 6: Development set accuracies on sentence initial (SI) tokens compared with non-sentence-initial (¬SI) tokens.

fine-grained POS tag. We used this dataset to augment the WSJ training data. Since the Noun-Verb examples only contain one labeled token per sentence, we assigned the unlabeled tokens a cost of zero in the cost function at training time.

ELMo and Data Augmentation Together Finally, we experimented with using both enhancements together. We trained the ELMo-enhanced model on the dataset augmented with the Noun-Verb training set examples. The motivating intuition for combining them is that the inclusion of the difficult Noun-Verb training set examples could encourage the model to make more use of ELMo embeddings than the model trained on the WSJ only. Another possibility is that these two types of enhancements are redundant and that one dominates the other.

3.4 Results

Table 1 shows the main results of both existing taggers and the enhanced models on both WSJ and the Noun-Verb Challenge Set.

Existing Taggers While all four selected taggers achieve accuracies above 97% on WSJ, they all struggle on our noun-verb challenge (Table 1). The widely used tagger of Toutanova et al. (2003) has an accuracy of just 57.6%, below the 70.1% accuracy of a per-word type majority class baseline (Table 2). The best performing tagger (Bohnet et al., 2018) was 3.9% above the next best model. However it still has an error rate of 25%.

The ranking of the four taggers stays the same whether one uses the WSJ or the Noun-Verb Challenge Set for evaluation. However, the magnitude of differences changes drastically. For example, on the WSJ test set, the differences between Dozat et al. (2017) and Toutanova et al. (2003) appear insignificant: Dozat et al. (2017) improves over Toutanova et al. (2003) by 0.09% absolute (3% relative reduction in error). When measured on the Noun-Verb Challenge Set, the differences are stark: the tagger of Dozat et al. (2017) is 12.8% absolute more accurate, which is a 30% relative reduction in error.

Enhancements Experimental results in Table 1 show that ELMo gave 7.2% absolute improvement and did not significantly affect the WSJ results[14]. This is further evidence that WSJ evaluation does not model ambiguities in cases where context matters. Adding the silver Noun-Verb data to the baseline model gave 10% absolute improvement over the baseline. This is significant given that the model capacity remained unchanged. By contrast, hooking up ELMo added a very large multi-layer BiLSTM language model to the parameters.

The best model was the model which used both ELMo embeddings and data augmentation. It achieved 13.1% absolute improvement over the state-of-the-art baseline of Bohnet et al. (2018), equivalent to over a 52% error reduction. This demonstrates that the improvement from ELMo is complementary to that from the additional Noun-Verb data.

Sentence-Initial Examples The trend in Table 1 is magnified in Table 6, which shows development set accuracies separately for tokens that are sentence-initial (SI), which are often imperatives, and for tokens that are not SI.

On SI accuracy, none of the WSJ-trained baselines could beat the most-frequent-tag baseline from the Noun-Verb training data. This shows that these sorts of examples, which are mostly imperatives, are underrepresented in the WSJ corpus. ELMo embeddings were able to improve both SI and non-SI accuracies by roughly the same amount, but again, not as much as adding the Noun-Verb data, which gave a 21.7% boost to SI accuracy. The efficacy of the Noun-Verb data in this case shows that directed training examples can

---

[14] We also ran the experiment using the "Original (5.5B)" ELMo model, trained on a larger and more diverse corpus. We did not find any significant difference between the two.

Model	Tuning Set WSJ	 NV
WSJ Test Set		
Bohnet et al. (2018)	98.00±0.12	97.98±0.13
+ELMo	97.94±0.08	97.85±0.16
+NV Data	97.98±0.11	97.94±0.14
+ELMo+NV Data	97.97±0.09	97.94±0.13
Noun-Verb Test Set		
Bohnet et al. (2018)	74.0±1.2	76.9±0.6 †
+ELMo	82.1±0.9	83.4±0.5 †
+NV Data	86.4±0.4	86.8±0.4
+ELMo+NV Data	88.9±0.3	89.3±0.2 ‡

Table 7: Effect of using different tuning sets. As usual with early stopping, the best tuning set performance was used to evaluate the test set. Here, we evaluated the same experimental runs at two points: when the performance was best on the WSJ development set, and again when the performance was best on the Noun-Verb development set. The increase in Noun-Verb results is significant at the $p < 0.001(\dagger)$ and $p < 0.01(\ddagger)$ levels.

be especially beneficial for fixing some common error patterns.

Impact of Tuning Set Table 7 compares performance of the same experiments on the WSJ and Noun-Verb Challenge test sets, tuned either using the WSJ or the Noun-Verb development set. The only effect of the change in tuning set was for the Noun-Verb tuning to cause the early stopping to sometimes be a little earlier. When we tuned on the Noun-Verb development set, the WSJ results remained almost unchanged, while the Noun-Verb test set results increased significantly. We see that the performance on each dataset is best when matched with its tuning data. The effect was greatest on the unenhanced model, which improved 2.9% absolute on the Noun-Verb evaluation. The best overall Noun-Verb test set result was **89.3**±0.2 when tuned this way.

3.5 Error Analysis

Table 8 shows representative examples that the best baseline run got wrong, along with the predictions from the best runs for each of the different enhancements. While each enhancement reduces all error types, adding Noun-Verb data improves imperatives in particular when compared with adding ELMo. This holds true even when imperatives are not sentence-initial, like the *practice* example in Table 8.

Of the errors made by our best model, roughly a quarter occurred when the focus word was a conjunction. This provides additional evidence for the importance of modeling non-local context in this dataset.

4 Homograph Disambiguation

To show the impact of our best models on a downstream task, we used the text-to-speech homograph disambiguation task described in Gorman et al. (2018). The dataset contains 161 word types, each of which has up to three possible pronunciations. In that work, the authors built a linear model that used lexical features of the focus word and its surrounding words, POS tags, and capitalization, to achieve 95.4% on this task. Here, we want to see the effectiveness of our taggers by using just the POS tag of each word to determine its pronunciation category. To do this, we annotated the homograph disambiguation train and test data with with POS tags using each of our taggers. We collected counts from the training corpus of the form <*word, POS tag, word_sense, Count*>. These counts show how many times a given word got assigned to a certain word sense when it has a certain POS tag. We used those counts to select the most frequent pronunciation for each <*word, POS tag*> pair on the test data. Note that this approach will miss some word senses that cannot be determined from the word and POS tag only, like the difference in pronunciation of the word "jesus" between English: /ˈdʒiːzəs/ and Spanish: /heɪˈsuːs/.

Table 9 shows results for the micro and macro accrucies among different word types in the same way (Gorman et al., 2018) reported their results. The overall results show similar trend to what is observed in the Noun-Verb evaluation results. The Choi (2016), and Bohnet et al. (2018) baseline taggers perform close to the full model in Gorman et al. (2018), which uses a wider context and more features. This is probably due to having a stronger POS tagger than the one used in that model. It is also interesting to see the gap between Toutanova et al. (2003) and the rest of baseline taggers which was measured only on the Noun-Verb evaluation and not in WSJ evaluation. The rest of the results show that using either ELMo achieves a 1.3% absolute improvement over the baseline. while adding data augmentation achieves 0.3% absolute improvement over the baseline. Using both ELMo

Example	Gold	Base	+ELMo	+Data	+ELMo +Data
Will gets his revenge by masquerading as Sue's hairdresser and forcibly shaving her head bald.	**NOUN**	*MD*	**NNP**	*MD*	**NNP**
Will putting a patch over my eye **help** to get the object out of it?	**VERB**	*NN*	**VB**	*NN*	**VB**
If you don't have a table, you can mount the frame on a desk, **stand**, or other structure that will hold the bike off the ground.	**NOUN**	*VB*	*VB*	**NN**	**NN**
For best results, **practice** hitting one note higher than your standard range.	**VERB**	*NN*	*NN*	**VB**	**VB**
Spirit actually suggests unpacking their smokes by rolling the cigarette between your fingers, filter to **end**, so that a pinch or so of tobacco comes out.	**NOUN**	*VB*	*VB*	*VB*	**NN**
Choose the highest combat level and **duel**.	**VERB**	*NN*	*NN*	*NN*	**VB**

Table 8: Development set examples that reflect the types of errors the enhancements address. Base is the tagger of Bohnet et al. (2018), while the remaining columns show the impact of the enhancements. Tags consistent with the gold annotations are in bold and inconsistent are in italics.

Model	Micro	Macro
Best ML system		
Gorman et al. (2018)	95.4	95.1
Existing Taggers		
Toutanova et al. (2003)	91.1	91.5
Choi (2016)	95.8	95.8
Dozat et al. (2017)	94.6	94.7
Bohnet et al. (2018)	95.9±0.2	95.9±0.2
Enhancements		
+ELMo	**96.7**±0.2	**96.7**±0.2
+NV Data	96.2±0.2	96.2±0.2
+ELMo+NV Data	**96.7**±0.3	**96.7**±0.3

Table 9: Accurcies of different models on the homograph disambiguation test set. All enhancements' improvements over (Bohnet et al., 2018) baseline are statistically significant $p < 0.008$. Standard deviations are estimated from $n = 10$ random restarts, and p-values were computed using a heteroscedastic two-tailed t-test.

and data augmentation was not better than just using ELMo. Those improvements correspond to a 28% error reduction compared to the machine-learned model in Gorman et al. (2018).

5 Discussion and Related Work

Dataset Creation Prior work in crowd-sourcing syntactic annotations and using them in models motivated the dataset creation portion of this work. Jha et al. (2010) showed that non-linguists could reliably do aspects of syntactic annotation, and Hovy et al. (2014) showed that non-experts could annotate universal part-of-speech tags (Petrov et al., 2012) almost as well as experts. He et al. (2016) then showed that incorporating crowd-sourced annotations improves parsing by a noticeable margin on the *subset* of sentences in which the human judgments affected the parser's output. Inspired by this result, we focused our efforts on collecting annotations that were likely to change a tagger's predictions and humans can annotate reliably.

This work filtered out trivial examples via hand-written heuristics targeted towards examples that taggers generally get correct (Section 2.1). One interesting direction for future work would be to eliminate this manual step. One option could be to instead use automatically produced high-precision interpretable rules to filter out these examples, such as the Anchor explanations output by Ribeiro et al. (2018). Table 1 in that paper shows how the system can automatically induce that a part-of-speech tagging system will tag the word *play* as a NOUN in the sentence *I went to a play yesterday* because the previous word is a determiner.

Measurement Manning (2011) performed an error analysis for WSJ and discovered that 19% of the errors fall under "Difficult linguistics" which need non-local context modeling to be able to solve them. The negative results of Kiddon et al. (2015) on using existing supervised part-of-speech taggers for imperative detection provided motivation for focusing on noun-verb confusion. However we are not aware of any prior work on trying to measure part-of-speech-tagging accuracy on hard ambiguities that are easily recognized by human using diverse corpora.

6 Conclusion and Future Work

This paper proposes a challenge set approach to evaluating part-of-speech taggers, and builds a new resource for doing so. We show that a part-of-speech tagger can be trained to be better at noun-verb ambiguity by using extra Noun-Verb targeted training data or by adding contextual word embedding. We also show that our evaluation data can measure improvements in Noun-Verb disambiguation that standard evaluation dataset was not able to capture. Those previously unmeasured improvements in the Noun-Verb disambiguation are shown to lead to improvements in a downstream task. Improvements were especially large on sentence-initial tokens, which are often imperatives. Even with these improvements, there is still a large gap between the noun-verb accuracies and overall WSJ tagging accuracy. We expect that closing this gap will make incorporating syntax more useful across natural language understanding applications.

Future work can include exploring ways to incorporate more context into the tagger, possibly by using information from dependency tree. Also investigating more downstream tasks and explore if this dataset can be used directly in downstream tasks in a way similar to what have been done in Swayamdipta et al. (2017) and (Eriguchi et al., 2017; Niehues and Cho, 2017; Kiperwasser and Ballesteros, 2018) for injecting syntax in semantic role labeling and translation tasks. A third direction for research would be using this dataset to evaluate different contextual modeling approaches and investigate the creation and using such context sensitive dataset to create simpler and smaller models that can capture a lot of contextual word representation.

Future work on dataset creation can include generating similar challenge datasets for different key ambiguities in NLP. A collection of such datasets could be one way to cover hard examples that models do not get right but humans are good at. Such targeted datasets can complement the use of large unsupervised contextual embedding models. This can open an avenue to improve core NLP tasks on hard relevant ambiguities that allows making progress on downstream tasks.

Acknowledgments

The authors gratefully acknowledge their anonymous reviewers for their insightful questions and feedback. We are grateful to Alexander Clines, Kazoo Sone, Ashwin Kakarla, Daphne Luong, and Austin Tarango for their assistance in the creation of the dataset. Thanks also go to members of the Google AI Language group for their input to this work, including Slav Petrov, Michael Collins, and all who provided feedback in reading group sessions.

References

Bernd Bohnet, Ryan McDonald, Gon¸calo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. 2018. Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings. In *(To Appear) Proceedings of ACL*.

Jinho D. Choi. 2016. Dynamic feature induction: The last gist to the state-of-the-art. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 271–281, San Diego, California. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

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

Pinar Donmez, Guy Lebanon, and Krishnakumar Balasubramanian. 2010. Unsupervised supervised learning i: Estimating classification and regression errors without labels. *Journal of Machine Learning Research*, 11(Apr):1323–1351.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78, Vancouver, Canada. Association for Computational Linguistics.

Kyle Gorman, Gleb Mazovetskiy, and Vitaly Nikolaev. 2018. Improving homograph disambiguation with supervised machine learning. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2337–2342.

Dirk Hovy, Barbara Plank, and Anders Søgaard. 2014. Experiments with crowdsourced re-annotation of a pos tagging data set. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (volume 2: Short Papers)*, volume 2, pages 377–382.

Mukund Jha, Jacob Andreas, Kapil Thadani, Sara Rosenthal, and Kathleen McKeown. 2010. Corpus creation for new genres: A crowdsourced approach to pp attachment. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 13–20. Association for Computational Linguistics.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. In *Transactions of the Association for Computational Linguistics*, volume 6, page 225–240.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 171–189. Springer.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. *WMT 2017*, pages 80–89.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of AAAI*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Kevin Small and Dan Roth. 2010. Margin-based active learning for structured predictions. *International Journal of Machine Learning and Cybernetics*, 1(1-4):3–25.

Richard Sproat, Julia Hirschberg, and David Yarowsky. 1992. A corpus-based synthesizer. In *Second International Conference on Spoken Language Processing*.

Jacob Steinhardt and Percy S Liang. 2016. Unsupervised risk estimation using only conditional independence structure. In *Advances in Neural Information Processing Systems*, pages 3657–3665.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. *arXiv preprint arXiv:1706.09528*.

Katrin Tomanek and Udo Hahn. 2009. Semi-supervised active learning for sequence labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1039–1047. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de-Paiva, Kira Droganova, Héctor Martínez Alonso, ÇağrÄ Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

What do character-level models learn about morphology? The case of dependency parsing

Clara Vania **Andreas Grivas**[*] **Adam Lopez**
Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh
c.vania@ed.ac.uk, andreasgrv@gmail.com, alopez@inf.ed.ac.uk

Abstract

When parsing morphologically-rich languages with neural models, it is beneficial to model input at the character level, and it has been claimed that this is because character-level models learn morphology. We test these claims by comparing character-level models to an oracle with access to explicit morphological analysis on twelve languages with varying morphological typologies. Our results highlight many strengths of character-level models, but also show that they are poor at disambiguating some words, particularly in the face of case syncretism. We then demonstrate that explicitly modeling morphological case improves our best model, showing that character-level models can benefit from targeted forms of explicit morphological modeling.

1 Introduction

Modeling language input at the character level (Ling et al., 2015; Kim et al., 2016) is effective for many NLP tasks, and often produces better results than modeling at the word level. For parsing, Ballesteros et al. (2015) have shown that character-level input modeling is highly effective on morphologically-rich languages, and the three best systems on the 45 languages of the CoNLL 2017 shared task on universal dependency parsing all use character-level models (Dozat et al., 2017; Shi et al., 2017; Björkelund et al., 2017; Zeman et al., 2017), showing that they are effective across many typologies.

The effectiveness of character-level models in morphologically-rich languages has raised a question and indeed debate about explicit modeling of morphology in NLP. Ling et al. (2015) propose that "prior information regarding morphology ... among others, should be incorporated" into character-level models, while Chung et al. (2016) counter that it is "unnecessary to consider these prior information" when modeling characters. Whether we need to explicitly model morphology is a question whose answer has a real cost: as Ballesteros et al. (2015) note, morphological annotation is expensive, and this expense could be reinvested elsewhere if the predictive aspects of morphology are learnable from strings.

Do character-level models learn morphology? We view this as an empirical claim requiring empirical evidence. The claim has been tested implicitly by comparing character-level models to word lookup models (Qian et al., 2016; Belinkov et al., 2017). In this paper, we test it explicitly, asking how character-level models compare with an oracle model with access to morphological annotations. This extends experiments showing that character-aware language models in Czech and Russian benefit substantially from oracle morphology (Vania and Lopez, 2017), but here we focus on dependency parsing (§2)—a task that benefits substantially from morphological knowledge—and we experiment with twelve languages using a variety of techniques to probe our models.

Our summary finding is that character-level models lag the oracle in nearly all languages (§3). The difference is small, but suggests that there is value in modeling morphology. When we tease apart the results by part of speech and dependency type, we trace the difference back to the character-level model's inability to disambiguate words even when encoded with arbitrary context (§4). Specifically, it struggles with *case syncretism*, in which noun case—and thus syntactic function—is ambiguous. We show that the oracle relies on morphological case, and that a character-level model provided *only* with morphological case rivals the oracle, even when case is provided by another predictive model (§5). Finally, we show that the crucial morphological features vary by language (§6).

---

[*] Work done while at the University of Edinburgh.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2573–2583
Brussels, Belgium, October 31 - November 4, 2018.

2 Dependency parsing model

We use a neural graph-based dependency parser combining elements of two recent models (Kiperwasser and Goldberg, 2016; Zhang et al., 2017). Let $w = w_1, \ldots, w_{|w|}$ be an input sentence of length $|w|$ and let w_0 denote an artificial ROOT token. We represent the ith input token w_i by concatenating its *word representation* (§2.3), $\mathbf{e}(w_i)$ and part-of-speech (POS) representation, $\mathbf{p}_i$.[1] Using a semicolon $(;)$ to denote vector concatenation, we have:

$$\mathbf{x}_i = [\mathbf{e}(w_i); \mathbf{p}_i] \tag{1}$$

We call $\mathbf{x}_i$ the *embedding* of w_i since it depends on context-independent word and POS representations. We obtain a context-sensitive *encoding* $\mathbf{h}_i$ with a bidirectional LSTM (bi-LSTM), which concatenates the hidden states of a forward and backward LSTM at position i. Using $\mathbf{h}_i^f$ and $\mathbf{h}_i^b$ respectively to denote these hidden states, we have:

$$\mathbf{h}_i = [\mathbf{h}_i^f; \mathbf{h}_i^b] \tag{2}$$

We use $\mathbf{h}_i$ as the final input representation of w_i.

2.1 Head prediction

For each word w_i, we compute a distribution over all other word positions $j \in \{0, ..., |w|\}/i$ denoting the probability that w_j is the headword of w_i.

$$P_{head}(w_j \mid w_i, w) = \frac{\exp(a(\mathbf{h}_i, \mathbf{h}_j))}{\sum_{j'=0}^{|w|} \exp(a(\mathbf{h}_i, \mathbf{h}_{j'}))} \tag{3}$$

Here, a is a neural network that computes an association between w_i and w_j using model parameters $\mathbf{U}_a$, $\mathbf{W}_a$, and $\mathbf{v}_a$.

$$a(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{v}_a \tanh(\mathbf{U}_a \mathbf{h}_i + \mathbf{W}_a \mathbf{h}_j) \tag{4}$$

2.2 Label prediction

Given a head prediction for word w_i, we predict its syntactic label $\ell_k \in L$ using a similar network.

$$P_{label}(\ell_k \mid w_i, w_j, w) = \frac{\exp(f(\mathbf{h}_i, \mathbf{h}_j)[k])}{\sum_{k'=1}^{|L|} \exp(f(\mathbf{h}_i, \mathbf{h}_j)[k'])} \tag{5}$$

where L is the set of output labels and f is a function that computes label score using model parameters $\mathbf{U}_\ell$, $\mathbf{W}_\ell$, and $\mathbf{V}_\ell$:

$$f(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{V}_\ell \tanh(\mathbf{U}_\ell \mathbf{h}_i + \mathbf{W}_\ell \mathbf{h}_j) \tag{6}$$

The model is trained to minimize the summed cross-entropy losses of both head and label prediction. At test time, we use the Chu-Liu-Edmonds (Chu and Liu, 1965; Edmonds, 1967) algorithm to ensure well-formed, possibly non-projective trees.

2.3 Computing word representations

We consider several ways to compute the word representation $\mathbf{e}(w_i)$ in Eq. 1:

word. Every word type has its own learned vector representation.

char-lstm. Characters are composed using a bi-LSTM (Ling et al., 2015), and the final states of the forward and backward LSTMs are concatenated to yield the word representation.

char-cnn. Characters are composed using a convolutional neural network (Kim et al., 2016).

trigram-lstm. Character trigrams are composed using a bi-LSTM, an approach that we previously found to be effective across typologies (Vania and Lopez, 2017).

oracle. We treat the morphemes of a morphological annotation as a sequence and compose them using a bi-LSTM. We only use universal inflectional features defined in the UD annotation guidelines. For example, the morphological annotation of "chases" is ⟨`chase, person=3rd, num-SG, tense=Pres`⟩.

For the remainder of the paper, we use the name of model as shorthand for the dependency parser that uses that model as input (Eq. 1).

3 Experiments

Data We experiment on twelve languages with varying morphological typologies (Table 1) in the Universal Dependencies (UD) treebanks version 2.0 (Nivre et al., 2017).[2] Note that while Arabic and Hebrew follow a root & pattern typology, their datasets are unvocalized, which might reduce the observed effects of this typology. Following common practice, we remove language-specific dependency relations and multiword token annotations. We use gold sentence segmentation, tokenization, universal POS (UPOS), and morphological (XFEATS) annotations provided in UD.

Implementation and training Our Chainer (Tokui et al., 2015) implementation encodes words (Eq. 2) in two-layer bi-LSTMs with 200 hidden units, and uses 100 hidden units for head and label

---

[1] This combination yields the best labeled accuracy according to Ballesteros et al. (2015).

[2] For Russian we use the UD_Russian_SynTagRus treebank, and for all other languages we use the default treebank.

Languages	#sents (K)	#tokens (K)	type/token ratio (%)
Finnish	12.2	162.6	28.5
Turkish	3.7	38.1	33.6
Czech	68.5	1173.3	9.5
English	12.5	204.6	8.1
German	14.1	269.6	17.7
Hindi	13.3	281.1	6
Portuguese	8.3	206.7	11.7
Russian	48.8	870	11.4
Spanish	14.2	382.4	11.1
Urdu	4.0	108.7	8.8
Arabic	6.1	223.9	10.3
Hebrew	5.2	137.7	11.7

Table 1: Training data statistics. Languages are grouped by their dominant morphological processes, from top to bottom: agglutinative, fusional, and root & pattern.

predictions (output of Eqs. 4 and 6). We set batch size to 16 for char-cnn and 32 for other models following a grid search. We apply dropout to the embeddings (Eq. 1) and the input of the head prediction. We use Adam optimizer with initial learning rate 0.001 and clip gradients to 5, and train all models for 50 epochs with early stopping. For the word model, we limit our vocabulary to the 20K most frequent words, replacing less frequent words with an unknown word token. The char-lstm, trigram-lstm, and oracle models use a one-layer bi-LSTM with 200 hidden units to compose subwords. For char-cnn, we use the small model setup of Kim et al. (2016).

Parsing Results Table 2 presents test results for every model on every language, establishing three results. First, they support previous findings that character-level models outperform word-based models—indeed, the char-lstm model outperforms the word model on LAS for all languages except Hindi and Urdu for which the results are identical.[3] Second, they establish strong baselines for the character-level models: the char-lstm generally obtains the best parsing accuracy, closely followed by char-cnn. Third, they demonstrate that character-level models rarely match the accuracy of an oracle model with access to explicit morphology. This reinforces a finding of Vania and Lopez (2017): character-level models are effective tools, but they do not learn everything about morphology, and they seem to be closer to oracle accuracy in agglutinative rather than in fusional languages.

4 Analysis

4.1 Why do characters beat words?

In character-level models, orthographically similar words share many parameters, so we would expect these models to produce good representations of OOV words that are morphological variants of training words. Does this effect explain why they are better than word-level models?

Sharing parameters helps with both seen and unseen words Table 3 shows how the character model improves over the word model for both non-OOV and OOV words. On the agglutinative languages Finnish and Turkish, where the OOV rates are 23% and 24% respectively, we see the highest LAS improvements, and we see especially large improvements in accuracy of OOV words. However, the effects are more mixed in other languages, even with relatively high OOV rates. In particular, languages with rich morphology like Czech, Russian, and (unvocalised) Arabic see more improvement than languages with moderately rich morphology and high OOV rates like Portuguese or Spanish. This pattern suggests that parameter sharing between pairs of *observed* training words can also improve parsing performance. For example, if "dog" and "dogs" are observed in the training data, they will share activations in their context *and* on their common prefix.

4.2 Why do morphemes beat characters?

Let's turn to our main question: what do character-level models learn about morphology? To answer it, we compare the oracle model to char-lstm, our best character-level model.

Morphological analysis disambiguates words In the oracle, morphological annotations disambiguate some words that the char-lstm must disambiguate from context. Consider these Russian sentences from Baerman et al. (2005):

(1) Maša čitaet <u>pis′mo</u>
Masha reads <u>letter</u>
'*Masha reads a <u>letter</u>.*'

---

[3] Note that Hindi and Urdu are mutually intelligible.

Model →	word		char-lstm		char-cnn		trigram-lstm		oracle		o/c
↓ Language	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	LAS
Finnish	85.7	80.8	**90.6**	**88.4**	89.9	87.5	89.7	87.0	90.6	88.8	+0.4
Turkish	71.4	61.6	**74.7**	**68.6**	74.4	67.9	73.2	65.9	75.3	69.5	+0.9
Czech	92.6	89.3	**93.5**	**90.6**	**93.5**	**90.6**	92.7	89.2	94.3	92.0	+1.4
English	90.6	88.9	91.3	89.4	**91.7**	**90.0**	90.4	88.5	91.7	89.9	+0.5
German	**88.1**	**84.5**	88.0	**84.5**	87.8	84.4	87.1	83.5	88.8	86.5	+2.0
Hindi	**95.8**	93.1	95.7	**93.3**	95.7	93.2	93.4	89.8	95.9	93.3	-
Portuguese	87.4	85.5	**87.8**	**86.0**	87.7	**86.0**	86.7	84.8	88.0	86.5	+0.5
Russian	92.4	90.1	**94.0**	**92.4**	93.8	92.1	92.0	89.5	94.4	93.3	+0.9
Spanish	89.4	86.9	89.8	**87.4**	**90.0**	87.3	88.6	85.5	90.0	87.7	+0.3
Urdu	91.1	87.0	91.2	87.1	**91.3**	**87.2**	88.6	83.5	90.9	87.0	-0.1
Arabic	75.5	70.9	**76.7**	72.1	76.6	**72.2**	74.6	68.9	76.7	72.7	+0.6
Hebrew	**73.5**	**69.8**	73.4	**69.8**	73.3	**69.8**	71.3	67.1	73.3	70.0	+0.2

Table 2: Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS) on **test set**. The best accuracy for each language is highlighted in **bold** for all models, and for all non-oracle models. **o/c:** LAS improvement from char-lstm to oracle.

Language	dev	LAS improvement	
	%OOV	non-OOV	OOV
Finnish	23.0	6.8	17.5
Turkish	24.0	4.6	13.5
Czech	5.8	1.4	3.9
English	6.8	0.7	5.2
German	9.7	0.9	0.7
Hindi	4.3	0.2	0.0
Portuguese	8.1	0.3	1.3
Russian	8.4	2.1	6.9
Spanish	7.0	0.4	0.7
Arabic	8.0	1.2	7.3
Hebrew	9.0	0.2	1.3

Table 3: LAS improvements (char-lstm − word) for non-OOV and OOV words on development set.

(2) Na stole ležit pis′mo
on table lies letter
'*There's a letter on the table.*'

Pis′mo ("letter") acts as the subject in (1), and as object in (2). This knowledge is available to the oracle via morphological case: in (1), the case of *pis′mo* is nominative and in (2) it is accusative. Could this explain why the oracle outperforms the character model?

To test this, we look at accuracy for word types

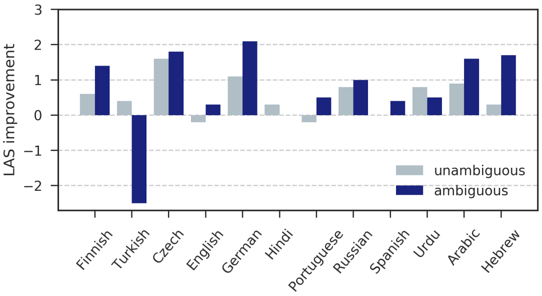

Figure 1: LAS improvements (oracle − char-lstm) for ambiguous and unambiguous words on development set.

that are *empirically* ambiguous—those that have more than one morphological analysis in the training data. Note that by this definition, some ambiguous words will be seen as unambiguous, since they were seen with only one analysis. To make the comparison as fair as possible, we consider only words that were observed in the training data. Figure 1 compares the improvement of the oracle on ambiguous and seen unambiguous words, and as expected we find that handling of ambiguous words improves with the oracle in almost all languages. The only exception is Turkish, which has the least training data.

Morphology helps for nouns Now we turn to a more fine-grained analysis conditioned on the annotated part-of-speech (POS) of the *depen-*

Language	Model	ADJ	NOUN	PRON	PROPN	VERB	Overall
Finnish	%tokens	8.1	32.5	8.2	6.7	16.1	-
	char-lstm	89.2	82.1	88.1	84.5	88.4	87.7
	oracle	90.3	83.3	89.5	86.2	89.3	88.5
	diff	+1.1	+1.2	+1.4	+1.7	+0.9	+0.8
Czech	%tokens	14.9	28.7	3.6	6.3	10.7	-
	char-lstm	94.2	83.6	85.3	84.3	90.7	91.2
	oracle	94.8	87.5	88.5	86.8	91.1	92.5
	diff	+0.6	+3.9	+3.2	+2.5	+0.4	+1.3
German	%tokens	7.6	20.4	9.5	5.6	12.1	-
	char-lstm	88.4	81.4	86.0	82.4	85.2	87.5
	oracle	89.1	87.1	93.2	84.4	86.3	89.7
	diff	+0.7	+5.7	+7.2	+2.0	+1.1	+2.2
Russian	%tokens	12.2	29.3	6.1	4.6	13.7	-
	char-lstm	93.2	86.7	92.0	80.2	88.5	91.6
	oracle	93.7	88.8	93.3	86.4	88.9	92.6
	diff	+0.5	+2.1	+1.3	+6.2	+0.4	+1.0

Table 4: Labeled accuracy for different parts of speech on development set.

dent. We focus on four languages where the oracle strongly outperforms the best character-level model on the development set: Finnish, Czech, German, and Russian.[4] We consider five POS categories that are frequent in all languages and consistently annotated for morphology in our data: adjective (ADJ), noun (NOUN), pronoun (PRON), proper noun (PROPN), and verb (VERB).

Table 4 shows that the three noun categories—ADJ, PRON, and PROPN—benefit substantially from oracle morphology, especially for the three fusional languages: Czech, German, and Russian.

Morphology helps for subjects and objects We analyze results by the dependency type of the *dependent*, focusing on types that interact with morphology: *root*, nominal subjects (*nsubj*), objects (*obj*), indirect objects (*iobj*), nominal modifiers (*nmod*), adjectival modifier (*amod*), obliques (*obl*), and (syntactic) case markings (*case*).

Figure 2 shows the differences in the confusion matrices of the char-lstm and oracle for those words on which both models correctly predict the head. The differences on Finnish are small, which we expect from the similar overall LAS of both models. But for the fusional languages, a pattern emerges: the char-lstm consistently underperforms the oracle on nominal subject, object, and indirect object dependencies—labels closely associated with noun categories. From inspection, it appears to frequently mislabel objects as nominal subjects when the dependent noun is morphologically ambiguous. For example, in the sentence of Figure 3, *Gelände* ("terrain") is an object, but the char-lstm incorrectly predicts that it is a nominal subject. In the training data, *Gelände* is ambiguous: it can be accusative, nominative, or dative.

In German, the char-lstm frequently confuses objects and indirect objects. By inspection, we found 21 mislabeled cases, where 20 of them would likely be correct if the model had access to morphological case (usually dative). In Czech and Russian, the results are more varied: indirect objects are frequently mislabeled as objects, obliques, nominal modifiers, and nominal subjects. We note that indirect objects are relatively rare in these data, which may partly explain their frequent mislabeling.

5 Characters and case syncretism

So far, we've seen that for our three fusional languages—German, Czech, and Russian—the or-

---

[4] This is slightly different than on the test set, where the effect was stronger in Turkish than in Finnish. In general, we found it difficult to draw conclusions from Turkish, possibly due to the small size of the data.

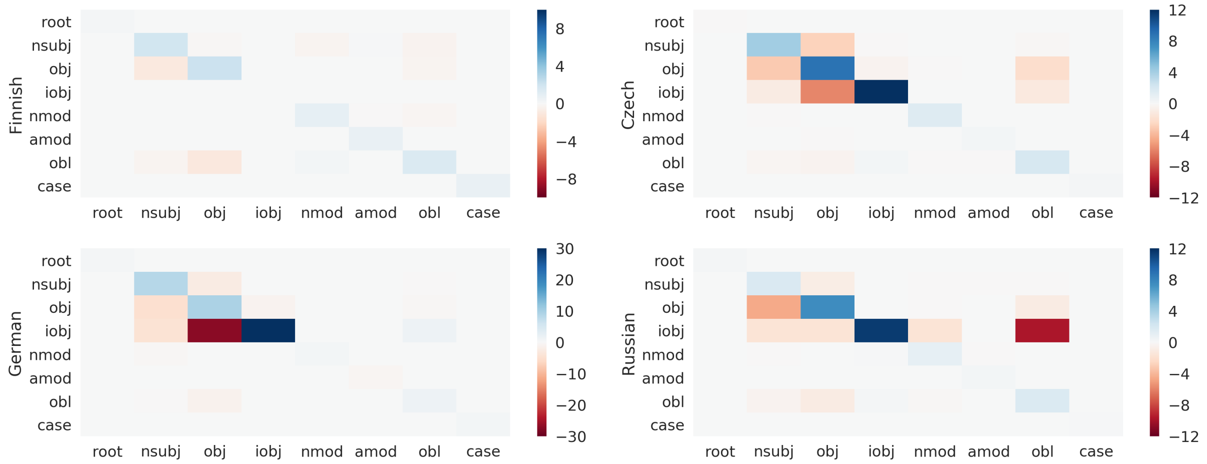

Figure 2: Heatmaps of the difference between oracle vs. char-lstm confusion matrices for label prediction when both head predictions are correct (**x-axis**: predicted labels; **y-axis**: gold labels). Blue cells have higher oracle values, red cells have higher char-lstm values.

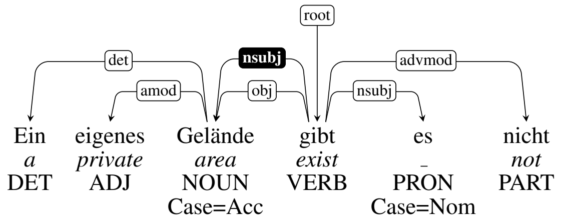

Figure 3: A sentence which the oracle parses perfectly (shown in white) and the char-lstm predicts an incorrect label (shown in black).

acle strongly outperforms a character model on nouns with ambiguous morphological analyses, particularly on core dependencies: nominal subjects, objects and indirect objects. Since the nominative, accusative, and dative morphological cases are strongly (though not perfectly) correlated with these dependencies, it is easy to see why the morphologically-aware oracle is able to predict them so well. We hypothesized that these cases are more challenging for the character model because these languages feature a high degree of *syncretism*—functionally distinct words that have the same form—and in particular case syncretism. For example, referring back to examples (1) and (2), the character model must disambiguate *pis′mo* from its context, whereas the oracle can directly disambiguate it from a feature of the word itself.[5]

To understand this, we first designed an experiment to see whether the char-lstm could successfully disambiguate *noun* case, using a method similar to (Belinkov et al., 2017). We train a neural classifier that takes as input a word representation from the trained parser and predicts a morphological feature of that word—for example that its case is nominative (`Case=Nom`). The classifier is a feedforward neural network with one hidden layer, followed by a ReLU non-linearity. We consider two representations of each word: its embedding ($\mathbf{x}_i$; Eq. 1) and its encoding ($\mathbf{h}_i$; Eq. 2). To understand the importance of case, we consider it alongside number and gender features as well as whole feature bundles.

The oracle relies on case Table 5 shows the results of morphological feature classification on Czech; we found very similar results in German and Russian (Appendix A.2). The oracle embeddings have almost perfect accuracy—and this is just what we expect, since the representation only needs to preserve information from its input. The char-lstm embeddings perform well on number and gender, but less well on case. This results suggest that the character-level models still struggle to learn case when given only the input text. Comparing the char-lstm with a baseline model which predicts the most frequent feature for each type in the training data, we observe that both of them show similar trends even though character models slightly outperforms the baseline model.

The classification results from the encoding are particularly interesting: the oracle still performs very well on morphological case, but less well on other features, even though they appear in

---

[5] We are far from first to observe that morphological case is important to parsing: Seeker and Kuhn (2013) observe the same for non-neural parsers.

Feature	baseline	embedding		encoder	
		char	oracle	char	oracle
Case	71.1	74.4	**100**	86.5	**98.6**
Gender	92.9	98.1	**100**	**71.2**	58.6
Number	88.9	94.7	**100**	84.2	**84.8**
All	70.4	72.5	**99.9**	**58.1**	50.2

Table 5: Morphological tagging accuracy from representations using the char-lstm and oracle embedding and encoder representations in Czech. Baseline simply chooses the most frequent tag. *All* means we concatenate all annotated features in UD as one tag.

the input. In the character model, the accuracy in morphological prediction also degrades in the encoding—*except* for case, where accuracy on case improves by 12%.

These results make intuitive sense: representations learn to preserve information from their input that is useful for subsequent predictions. In our parsing model, morphological case is very useful for predicting dependency labels, and since it is present in the oracle's input, it is passed almost completely intact through each representation layer. The character model, which must disambiguate case from context, draws as much additional information as it can from surrounding words through the LSTM encoder. But other features, and particularly whole feature bundles, are presumably less useful for parsing, so neither model preserves them with the same fidelity.[6]

Explicitly modeling case improves parsing accuracy Our analysis indicates that case is important for parsing, so it is natural to ask: Can we improve the neural model by explicitly modeling case? To answer this question, we ran a set of experiments, considering two ways to augment the char-lstm with case information: multitask learning (MTL; Caruana, 1997) and a pipeline model in which we augment the char-lstm model with either predicted or gold case. For example, we use ⟨`p`, `i`, `z`, `z`, `a`, `Nom`⟩ to represent *pizza* with nominative case. For MTL, we follow the setup of Søgaard and Goldberg (2016) and Coavoux and Crabbé (2017). We increase the biLSTMs layers from two to four and use the first two layers to predict morphological case, leaving out the other two layers specific only for parser. For the pipeline model, we train a morphological tagger to predict morphological case (Appendix A.1). This tagger does not share parameters with the parser.

Language	Input	Dev	Test
Czech	char	91.2	90.6
	char (multi-task)	91.6	91.0
	char + predicted case	**92.2**	**91.8**
	char + gold case	92.3	91.9
	oracle	92.5	92.0
German	char	87.5	84.5
	char (multi-task)	**87.9**	84.4
	char + predicted case	87.8	**86.4**
	char + gold case	90.2	86.9
	oracle	89.7	86.5
Russian	char	91.6	92.4
	char (multi-task)	92.2	92.6
	char + predicted case	**92.5**	**93.3**
	char + gold case	92.8	93.5
	oracle	92.6	93.3

Table 6: LAS results when case information is added. We use **bold** to highlight the best results for models without explicit access to gold annotations.

Table 6 summarizes the results on Czech, German, and Russian. We find augmenting the char-lstm model with either oracle or predicted case improve its accuracy, although the effect is different across languages. The improvements from predicted case results are interesting, since in non-neural parsers, predicted case usually harms accuracy (Tsarfaty et al., 2010). However, we note that our taggers use gold POS, which might help. The MTL models achieve similar or slightly better performance than the character-only models, suggesting that supplying case in this way is beneficial. Curiously, the MTL parser is worse than the the pipeline parser, but the MTL case tagger is better than the pipeline case tagger (Table 7). This indicates that the MTL model must learn to encode case in the model's representation, but must not learn to effectively use it for parsing. Finally, we

---

[6] This finding is consistent with Ballesteros (2013) which performed careful feature analysis on morphologically rich languages and found that lemma and case features provide the highest improvement in a non-neural transition based parser compared to other features.

Language	%case	Dev		Test	
		PL	MT	PL	MT
Czech	66.5	95.4	**96.7**	95.2	**96.6**
German	36.2	**92.6**	92.0	90.8	**91.4**
Russian	55.8	95.8	**96.5**	95.9	**96.5**

Table 7: Case accuracy for case-annotated tokens, for pipeline (**PL**) vs. multitask (**MT**) setup. **%case** shows percentage of training tokens annotated with case.

observe that augmenting the char-lstm with either gold or predicted case improves the parsing performance for all languages, and indeed closes the performance gap with the full oracle, which has access to *all* morphological features. This is especially interesting, because it shows using carefully targeted linguistic analyses can improve accuracy as much as wholesale linguistic analysis.

6 Understanding head selection

The previous experiments condition their analysis on the *dependent*, but dependency is a relationship between dependents and heads. We also want to understand the importance of morphological features to the *head*. Which morphological features of the head are important to the oracle?

Composing features in the oracle To see which morphological features the oracle depends on when making predictions, we augmented our model with a **gated attention mechanism** following Kuncoro et al. (2017). Our new model attends to the morphological features of candidate head w_j when computing its association with dependent w_i (Eq. 3), and morpheme representations are then scaled by their *attention weights* to produce a final representation.

Let $f_{i1}, \cdots, f_{ik}$ be the k morphological features of w_i, and denote by $\mathbf{f}_{i1}, \cdots, \mathbf{f}_{ik}$ their corresponding *feature embeddings*. As in §2, $\mathbf{h}_i$ and $\mathbf{h}_j$ are the encodings of w_i and w_j, respectively. The morphological representation $\mathbf{m}_j$ of w_j is:

$$\mathbf{m}_j = [\mathbf{f}_{j1}, \cdots, \mathbf{f}_{jk}]^\top \mathbf{k} \tag{7}$$

where $\mathbf{k}$ is a vector of attention weights:

$$\mathbf{k} = \mathrm{softmax}([\mathbf{f}_{j1}, \cdots, \mathbf{f}_{jk}]^\top \mathbf{V}\mathbf{h}_i) \tag{8}$$

The intuition is that dependent w_i can choose which morphological features of w_j are most important when deciding whether w_j is its head. Note that this model is asymmetric: a word only attends to the morphological features of its (single) parent, and not its (many) children, which may have different functions. [7]

We combine the morphological representation with the word's encoding via a sigmoid gating mechanism.

$$\mathbf{z}_j = \mathbf{g} \odot \mathbf{h}_j + (1 - \mathbf{g}) \odot \mathbf{m}_j \tag{9}$$

$$\mathbf{g} = \sigma(\mathbf{W}_1 \mathbf{h}_j + \mathbf{W}_2 \mathbf{m}_j) \tag{10}$$

where $\odot$ denotes element-wise multiplication. The gating mechanism allows the model to choose between the computed word representation and the weighted morphological representations, since for some dependencies, morphological features of the head might not be important. In the final model, we replace Eq. 3 and Eq. 4 with the following:

$$P_{head}(w_j | w_i, w) = \frac{\exp(a(\mathbf{h}_i, \mathbf{z}_j))}{\sum_{j'=0}^{N} \exp a(\mathbf{h}_i, \mathbf{z}_{j'})} \tag{11}$$

$$a(\mathbf{h}_i, \mathbf{z}_j) = \mathbf{v}_a \tanh(\mathbf{U}_a \mathbf{h}_i + \mathbf{W}_a \mathbf{z}_j) \tag{12}$$

The modified label prediction is:

$$P_{label}(\ell_k | w_i, w_j, w) = \frac{\exp(f(\mathbf{h}_i, \mathbf{z}_j)[k])}{\sum_{k'=0}^{|L|} \exp(f(\mathbf{h}_i, \mathbf{z}_j)[k'])} \tag{13}$$

where f is again a function to compute label score:

$$f(\mathbf{h}_i, \mathbf{z}_j) = \mathbf{V}_\ell \tanh(\mathbf{U}_\ell \mathbf{h}_i + \mathbf{W}_\ell \mathbf{z}_j) \tag{14}$$

Attention to headword morphological features We trained our augmented model (**oracle-attn**) on Finnish, German, Czech, and Russian. Its accuracy is very similar to the oracle model (Table 8), so we obtain a more interpretable model with no change to our main results.

Next, we look at the learned attention vectors to understand which morphological features are important, focusing on the core arguments: nominal subjects, objects, and indirect objects. Since our model knows the case of each dependent, this enables us to understand what features it seeks in potential heads for each case. For simplicity, we only report results for words where both head and label predictions are correct.

---

[7] This is a simple and much less computationally demanding variant of the model of Dozat et al. (2017), which uses different views for each head/dependent role.

Language	oracle		oracle-attn	
	UAS	LAS	UAS	LAS
Finnish	**89.2**	**87.3**	88.9	86.9
Czech	93.4	91.3	**93.5**	**91.3**
German	90.4	88.7	**90.7**	**89.1**
Russian	**93.9**	**92.8**	93.8	92.7

Table 8: Our attention experiment results on development set.

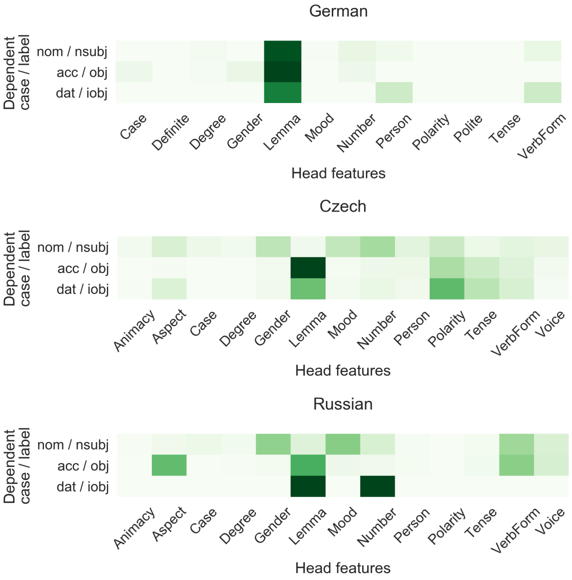

Figure 4: The importance of morphological features of the head for subject and object predictions.

Figure 4 shows how attention is distributed across multiple features of the head word. In Czech and Russian, we observe that the model attends to *Gender* and *Number* when the noun is in nominative case. This makes intuitive sense since these features often signal subject-verb agreement. As we saw in earlier experiments, these are features for which a character model can learn reliably good representations. For most other dependencies (and all dependencies in German), *Lemma* is the most important feature, suggesting a strong reliance on lexical semantics of nouns and verbs. However, we also notice that the model sometimes attends to features like *Aspect*, *Polarity*, and *VerbForm*—since these features are present only on verbs, we suspect that the model may simply use them as convenient signals that a word is verb, and thus a likely head for a given noun.

7 Conclusion

Character-level models are effective because they can represent OOV words and orthographic regularities of words that are consistent with morphology. But they depend on context to disambiguate words, and for some words this context is insufficient. Case syncretism is a specific example that our analysis identified, but the main results in Table 2 hint at the possibility that different phenomena are at play in different languages.

While our results show that prior knowledge of morphology is important, they also show that it can be used in a targeted way: our character-level models improved markedly when we augmented them only with case. This suggests a pragmatic reality in the middle of the wide spectrum between pure machine learning from raw text input and linguistically-intensive modeling: our new models don't need all prior linguistic knowledge, but they clearly benefit from some knowledge in addition to raw input. While we used a data-driven analysis to identify case syncretism as a problem for neural parsers, this result is consistent with previous linguistically-informed analyses (Seeker and Kuhn, 2013; Tsarfaty et al., 2010). We conclude that neural models can still benefit from linguistic analyses that target specific phenomena where annotation is likely to be useful.

Acknowledgments

Clara Vania is supported by the Indonesian Endowment Fund for Education (LPDP), the Centre for Doctoral Training in Data Science, funded by the UK EPSRC (grant EP/L016427/1), and the University of Edinburgh. We would like to thank Yonatan Belinkov for the helpful discussion regarding morphological tagging experiments. We thank Sameer Bansal, Marco Damonte, Denis Emelin, Federico Fancellu, Sorcha Gilroy, Jonathan Mallinson, Joana Ribeiro, Naomi Saphra, Ida Szubert, Sabine Weber, and the anonymous reviewers for helpful discussion of this work and comments on previous drafts of the paper.

References

Matthew Baerman, Dunstan Brown, and Greville G. Corbett. 2005. *The Syntax-Morphology Interface: A Study of Syncretism*. Cambridge Studies in Linguistics. Cambridge University Press.

Miguel Ballesteros. 2013. Effective morphological feature selection with MaltOptimizer at the

SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 63–70, Seattle, Washington, USA. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal. Association for Computational Linguistics.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Anders Björkelund, Agnieszka Falenska, Xiang Yu, and Jonas Kuhn. 2017. IMS at the CoNLL 2017 UD shared task: CRFs and perceptrons meet neural networks. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 40–51, Vancouver, Canada. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany. Association for Computational Linguistics.

Maximin Coavoux and Benoit Crabbé. 2017. Multilingual lexicalized constituency parsing with word-level auxiliary tasks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 331–336. Association for Computational Linguistics.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. 2016. Character-aware neural language models. In *Proceedings of the 2016 Conference on Artificial Intelligence (AAAI)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258, Valencia, Spain. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Joakim Nivre et al. 2017. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague, `http://hdl.handle.net/11234/1-1983`.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016. Investigating language universal and specific properties in word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1478–1488, Berlin, Germany. Association for Computational Linguistics.

Wolfgang Seeker and Jonas Kuhn. 2013. Morphological and syntactic case in statistical dependency parsing. *Computational Linguistics*, 39(1):23–55.

Tianze Shi, Felix G. Wu, Xilun Chen, and Yao Cheng. 2017. Combining global models for parsing universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 31–39, Vancouver, Canada. Association for Computational Linguistics.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235. Association for Computational Linguistics.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings* *of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (SPMRL): What, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, SPMRL '10, pages 1–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2016–2027. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de-Paiva, Kira Droganova, Héctor Martínez Alonso, Çağr Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.

Learning Better Internal Structure of Words for Sequence Labeling

Yingwei Xin, Ethan Hart, Vibhuti Mahajan, Jean-David Ruvini
eBay Research
2025 Hamilton Ave, San Jose, CA 95125, USA
`{yixin,ejhart,vibmahajan,jean-david.ruvini}@ebay.com`

Abstract

Character-based neural models have recently proven very useful for many NLP tasks. However, there is a gap of sophistication between methods for learning representations of sentences and words. While, most character models for learning representations of sentences are deep and complex, models for learning representations of words are shallow and simple. Also, in spite of considerable research on learning character embeddings, it is still not clear which kind of architecture is the best for capturing character-to-word representations. To address these questions, we first investigate the gaps between methods for learning word and sentence representations. We conduct detailed experiments and comparisons on different state-of-the-art convolutional models, and also investigate the advantages and disadvantages of their constituents. Furthermore, we propose IntNet, a funnel-shaped wide convolutional neural architecture with no down-sampling for learning representations of the internal structure of words by composing their characters from limited, supervised training corpora. We evaluate our proposed model on six sequence labeling datasets, including named entity recognition, part-of-speech tagging, and syntactic chunking. Our in-depth analysis shows that IntNet significantly outperforms other character embedding models and obtains new state-of-the-art performance without relying on any external knowledge or resources.

1 Introduction

Sequence labeling is the task of assigning a label or class to each element of a sequence of data, and is one of the first stages in many natural language processing (NLP) tasks. For example, named entity recognition (NER) aims to classify words in a sentence into several predefined categories of interest such as person, organization, location, etc. Part-of-speech (POS) tagging assigns a part of speech to each word in an input sentence. Syntactic chunking divides text into syntactically related, non-overlapping groups of words. Sequence labeling is a challenging problem because human annotation is very expensive and typically only a small amount of tagging data is available.

Most traditional sequence labeling systems have been dominated by linear statistical models which heavily rely on feature engineering. As a result, carefully constructed hand-crafted features and domain-specific knowledge are widely used for solving these tasks. Unfortunately, it is costly to develop domain specific knowledge and hand-crafted features. Recently, neural networks using character-level information have been used successfully for minimizing the need of feature engineering. There are basically two threads of character-based modeling, one focuses on learning representations of sentences for semantics and syntax (Zhang et al., 2015; Conneau et al., 2017); the other focuses on learning representations of words for the purpose of eliminating hand-crafted features for word shape information (Lample et al., 2016; Ma and Hovy, 2016).

Two main state-of-the-art approaches of learning character representations for sequence labeling emerged from the latter thread. One is based on RNNs and uses bidirectional LSTMs or GRUs to learn forward and backward character information (Ling et al., 2015; Lample et al., 2016; Yang et al., 2017). The other approach is based on CNNs with a fixed-size window around each word to create character-level representations (Santos and Zadrozny, 2014; Chiu and Nichols, 2016; Ma and Hovy, 2016). However, there is a gap in the sophistication between character-based methods for learning representations of sentences compared to that of words. We found that most of the state-of-the-art character-based CNN models for words

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2584–2593
Brussels, Belgium, October 31 - November 4, 2018.

use a convolution followed by max pooling as a shallow feature extractor, which is very different from the CNN models with deep and complex architecture for sentences. In spite of considerable research on learning character embeddings, it is still not clear which kind of architecture is the best for capturing character-to-word representations.

Therefore, a number of questions remain open:

- Why is there a gap between methods for learning representations of sentences and words? How can this gap be bridged?
- How do state-of-the-art character embedding models differ in term of performance?
- What kind of neural network architecture is better for learning the internal structure of a word? Deep or shallow? Narrow or wide?

To answer these questions, we first investigate the gap between learning word representations and sentence representations for convolutional architectures. The most straightforward idea is to add more convolutional layers which follows the approaches from learning representations of sentences. Interestingly, we observe the accuracy does not increase much and found that accuracy drops when we increased the depth of the network. This observation shows that learning character representations for the internal structure of words is very different than sentences, and also might explain one of the reasons there has been a gap in character-based CNN models for representing words and sentences.

In this paper, we present detailed experiments and comparisons across different state-of-the-art convolutional models from natural language processing and computer vision. We also investigate the advantages and disadvantages of some of their constituents on different convolutional architectures. Furthermore, we propose IntNet, a funnel-shaped wide convolutional neural network for learning the internal structure of words by composing their characters. Unlike previous CNN-based approaches, our funnel-shaped IntNet explores deeper and wider architecture with no down-sampling for learning character-to-word representations from limited supervised training corpora. Lastly, we combine our IntNet model with LSTM-CRF, which captures both word shape and context information, and jointly decode tags for sequence labeling.

The main contributions of this paper are the following:

- We conduct detailed studies on investigating the gap between learning word representations and sentence representations.
- We provide in-depth experiments and empirical comparisons of different convolutional models and explore the advantages and disadvantages of their components for learning character-to-word representations.
- We propose a funnel-shaped wide convolutional neural architecture with no down-sampling that focuses on learning a better internal structure of words.
- Our proposed compositional character-to-word model combined with LSTM-CRF achieves state-of-the-art performance for various sequence labeling tasks.

This paper is organized as follows: Section 2 describes multiple threads of related work. Section 3 presents the whole architecture of the neural network. Section 4 provides details about experimental settings and compared methods. Section 5 reports model results on different benchmarks with detailed analyses and discussion.

2 Related Work

There exist three threads of related work regarding the topic of this paper: (i) different convolutional architectures from different domains; (ii) character embedding models for words; (iii) sequence labeling with deep neural network.

CNN models across domains. Convolutional neural networks (CNNs) are very useful in extracting information from raw signals. In the area of NLP, Kim (2014) was the first to propose shallow CNN with word embeddings for sentence classification. Zhang et al. (2015) proposed CNN with 6 convolutional layers by directly extracting character level information for learning representations of semantic structure on sentences. Recently, Conneau et al. (2017) proposed a VDCNN architecture with 29 convolutional layers using residual connections for text classification. However, one study on randomly dropping layers for training deep residual networks, (Huang et al., 2016), has shown that not all layers may be needed and highlighted there is some amount of redundancy in

ResNet (He et al., 2016). Also, some research has shown promising results with wide architectures, for example, wide ResNet (Zagoruyko and Komodakis, 2016), Inception-ResNet (Szegedy et al., 2017) and DenseNet (Huang et al., 2017). These models use character-level information to learn representations are for sentences, not words.

Character embedding models. Santos and Zadrozny (2014) proposed a CNN model to learn character representations of words to replace hand-crafted features for part-of-speech tagging. Ling et al. (2015) proposed a bidirectional LSTM over characters to use as input for learning character-to-word representations. Chiu and Nichols (2016) proposed a bidirectional LSTM-CNN with lexicons for named entity recognition by applying the CNN-based character embedding model from Santos and Zadrozny (2014). Plank et al. (2016) proposed a bi-LSTM model with auxiliary loss for multilingual part-of-speech tagging by following the LSTM-based character embedding model from Ling et al. (2015). Cotterell and Heigold (2017) proposed a character-level transfer learning model for neural morphological tagging.

Sequence labeling. Collobert et al. (2011) first proposed a method based on CNN-CRF that learns important features from words and requires few hand-crafted features. Huang et al. (2015) proposed a bidirectional LSTM-CRF model by using word embeddings and hand-crafted features for sequence tagging. Lample et al. (2016) applied the LSTM-based character embedding model from Ling et al. (2015) with bidirectional LSTM-CRF and obtained best results on NER for Spanish, Dutch, and German. Ma and Hovy (2016) applied the CNN-based character embedding model from Chiu and Nichols (2016), but without using any data preprocessing or external knowledge and achieved the best result on NER for English and part-of-speech tagging. Also, there have been some joint models which use additional knowledge, like transfer learning (Yang et al., 2017), pre-trained language models (Peters et al., 2017), language model joint training (Rei, 2017), and multi-task learning (Liu et al., 2018). Without any additional supervision or extra resources, LSTM-CRF (Lample et al., 2016) and LSTM-CNN-CRF (Ma and Hovy, 2016) are current state-of-the-art methods. To test the effectiveness of our proposed model, we use these two models as our baselines in the latter sections.

3 Neural Network Architecture

3.1 IntNet

Character embeddings. The first step is to initialize the character embeddings for each word w in the input sequence. We define the finite set of characters V^{char}. This vocabulary contains all the variations of the raw text, including uppercase and lowercase letters, numbers, punctuation marks, and symbols. Unlike some character-based approaches, we do not use any character-level prepossessing which enables our model to learn and capture regularities from prefixes to suffixes to construct character-to-word representations. The input word w is decomposed into a sequence of characters $\{c_1, ..., c_n\}$, where n is the length of w. Character embeddings are encoded by column vectors in the embedding matrix $W^{char} \in \mathbb{R}^{d^{char} \times |V^{char}|}$, where d^{char} is the number of parameters for each character in V^{char}. Given a character c_i, its embedding r_i^{char} is obtained by the matrix-vector product:

$$r_i^{char} = W^{char} v_i^{char}, \tag{1}$$

where v_i^{char} is defined as a one-hot vector for c_i. We randomly initialize a look-up table with values drawn from a uniform distribution with range $[-\sqrt{\frac{3}{d^{char}}}, +\sqrt{\frac{3}{d^{char}}}]$, where d^{char} is empirically chosen by users. The character set includes all unique characters and the special tokens `PADDING` and `UNKNOWN`. We do not perform any character-level preprocessing, including case normalization, digit replacement (e.g. replacing all sequences of digits 0-9 with a single “0”), nor do we use any capitalization features (e.g. `allCaps`, `upperInitial`, `lowercase`, `mixedCaps`, `noinfo`).

Convolutional blocks. The input for the IntNet is the sequence of character embeddings $\{r_1^{char}, ..., r_n^{char}\}$. First is the initial convolutional layer, which is a temporal convolutional module that computes 1-D convolutions. Let $\mathbf{x}_i \in \mathbb{R}^{d^{char} \times r^{char}}$ be the concatenation of the character embeddings for each w. The initial convolutional layer applies a matrix-vector operation to each successive window of size k^{char}. An input k-grams $\mathbf{x}_{i:i+k-1}$ is transformed through a convolution filter $\mathbf{w}_c$:

$$\mathbf{c}_i = f(\mathbf{w}_c \cdot \mathbf{x}_{i:i+k-1} + b_c), \tag{2}$$

where $\mathbf{c}_i$ is the feature map of 1-D convolution, f

is the non-linear ReLU function, and b_c is a bias term. Equation 2 produces m filters with different kernel sizes. The filters are computed with different kernels by the initial convolutional layers are concatenated:

$$\mathbf{g}_0 = [c_1^{k_1} \dots c_m^{k_1}; c_1^{k_2} \dots c_m^{k_2}; c_1^{k_h} \dots c_m^{k_h}], \qquad (3)$$

where h is the number of kernels, $\mathbf{g}_0$ is the output for the initial convolutional layer which feeds into the next convolutional block.

We define $\mathcal{F}(\cdot)$ as a function of several consecutive operations within a convolutional block. Firstly, a N$\times$1 convolution transforms the input. The output size is $4 \times m \times h$ feature maps, like a bottleneck layer. The next step consists of multiple 1-D convolutions with kernels of different sizes. Lastly, we concatenate all the feature maps from kernels of different size. In each convolution, we use a batch normalization, followed by a ReLU activation and N$\times$k temporal convolution.

Funnel-shaped wide architecture. The network comprises of L convolutional layers, which implies $(\frac{L-1}{2})$ convolutional blocks. We use direct connections from every other layer to all subsequent layers, inspired by dense connection. Therefore, the l^{th} layer has access to the feature maps of all the alternate layers:

$$\mathbf{g}_l = \mathcal{F}_l([\mathbf{g}_0, \mathbf{g}_2, \dots, \mathbf{g}_{l-2}]). \qquad (4)$$

Equation 4 ensures maximum information flow between blocks in the network. Compared to residual connection $\mathcal{F}_l(\mathbf{g}_{l-1}) + \mathbf{g}_{l-1}$, it can be viewed as an extreme case of residual connection and makes feature reuse possible. Unlike DenseNet and ResNet, we concatenate feature maps by different kernels in every other convolutional layers, which captures different levels of features and makes our wide architecture possible, inspired by Inception. Different levels of concatenation can help IntNet to learn different patterns of word shape information. We compare our architecture to residual connection and dense connection for learning character-to-word representations in Section 5.

Without down-sampling. Compared to other CNN models like ResNet and DenseNet, our model does not contain any halve down-sampling layer or average pooling to reduce resolution. We did not find these operations to be helpful and, in some cases, found them to be detrimental to performance. These operations are useful for sentences and images, but might break the internal structure of words, like the sequential patterns for prefixes and suffixes.

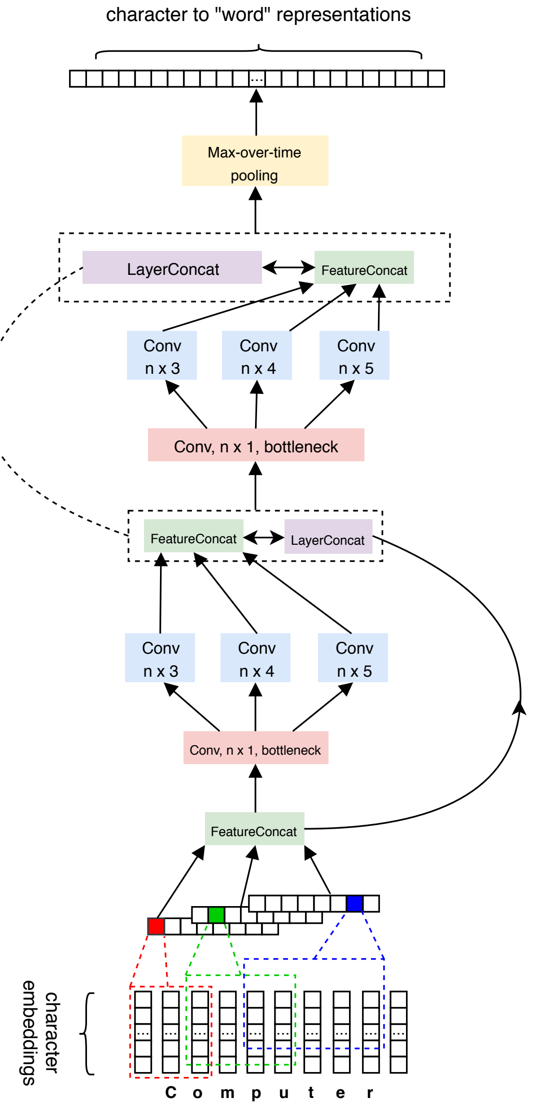

Figure 1: The main architecture of IntNet.

Character-to-word representations. In the last layer, we use a max-over-time pooling operation:

$$\hat{\mathbf{c}}_i = max(\mathbf{c}_i), \qquad (5)$$

which takes the maximum value corresponding to a particular filter. The idea is to capture the most important feature with the highest value for each feature map. Finally, we concatenate all of salient features together as a representation for this word:

$$\mathbf{z} = [\hat{\mathbf{c}_0}, \hat{\mathbf{c}_1}, \dots \hat{\mathbf{c}_u}], \qquad (6)$$

where u is the number of salient features which is equal to the total number of output feature maps in the last layer. If each function $\mathcal{F}_l$ produces p feature maps, we obtain $(p_0 + p \times \frac{L-1}{2})$ representations, where p_0 is the number of output feature maps in the initial convolution layer.

3.2 Bi-directional RNN

Given the character-to-word representations are computed by IntNet in Equation 6, we denote the input vector $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n)$ for a sentence. LSTM (Hochreiter and Schmidhuber, 1997) returns the sequence $(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$ that represents the sequential information at every step. We use the following implementation:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{zi}\mathbf{z}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{zf}\mathbf{z}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\widetilde{\mathbf{c}_t} &= tanh(\mathbf{W}_{zc}\mathbf{z}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \widetilde{\mathbf{c}_t} \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{zo}\mathbf{z}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t),
\end{aligned}$$

where σ is the element-wise sigmoid function and $\odot$ is the element-wise product. $\mathbf{z}_t$ is the input vector at time t and $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$ are the input gate, forget gate, output gate, and cell vectors, all of which are the same size as the hidden vector $\mathbf{h}_t$. $\mathbf{W}_{zi}$, $\mathbf{W}_{zf}$, $\mathbf{W}_{zo}$, $\mathbf{W}_{zc}$ denote the weight matrices of different gates for input $\mathbf{z}_t$; $\mathbf{W}_{hi}$, $\mathbf{W}_{hf}$, $\mathbf{W}_{ho}$, $\mathbf{W}_{hc}$ are the weight matrices for hidden state $\mathbf{h}_t$, and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$, $\mathbf{b}_c$ denote the bias vectors. Forward LSTM and backward LSTM compute the representations of $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$ for left and right context of the sentence, respectively. We concatenate two hidden states to form the output of bi-directional LSTM $[\overrightarrow{\mathbf{h}_t}, \overleftarrow{\mathbf{h}_t}]$ for capturing context information from both sides.

3.3 Scoring Function

Instead of predicting each label independently, we consider the correlations between labels in neighborhoods and jointly decode the best chain of labels for a given input sentence by leveraging a conditional random field (Lafferty et al., 2001). Formally, the sequence of labels is defined as:

$$\mathbf{y} = (y_1, y_2, ..., y_T). \tag{7}$$

To define the scoring function $f(\mathbf{h}, \mathbf{y})$ for each position t, we multiply the hidden state $\mathbf{h}_t^w$ with a parameter vector $\mathbf{w}_{y_t}$ that is indexed by the tag y_t to obtain the matrix of scores output by the bidirectional LSTM network. Therefore, the function f can be written as:

$$f(\mathbf{h}, \mathbf{y}) = \sum_{t=1}^{T} \mathbf{w}_{y_t}\mathbf{h}_t^w + \sum_{t=1}^{T} \mathbf{A}_{y_{t-1},y_t}. \tag{8}$$

In Equation 8, $\mathbf{A}$ is a matrix of transition scores, $\mathbf{A}_{i,j}$ represents the score of a transition from the tag i to tag j, y_1 is the start tag of a sentence. Let $\mathcal{Y}(\mathbf{h})$ denote the set of possible label sequences for $\mathbf{h}$. A probabilistic model for a sequence defines a family of conditional probabilities $p(\mathbf{y}|\mathbf{h})$ over all possible label sequences $\mathbf{y}$ given $\mathbf{h}$ with the following form:

$$p(\mathbf{y}|\mathbf{h}) = \frac{e^{f(\mathbf{h},\mathbf{y})}}{\sum_{y' \in \mathcal{Y}(\mathbf{h})} e^{f(\mathbf{h},\mathbf{y}')}}. \tag{9}$$

3.4 Objective Function and Inference

For end-to-end network training, we use maximum conditional likelihood estimation to maximize the log probability of the correct tag sequence:

$$log(p(\mathbf{y}|\mathbf{h})) = f(\mathbf{h}, \mathbf{y}) - log\left(\sum_{y' \in \mathcal{Y}(\mathbf{h})} e^{f(\mathbf{h},\mathbf{y}')}\right).$$

While decoding, we predict the label sequence that obtains the highest score given by:

$$\mathbf{y}^* = \arg \max_{y' \in \mathcal{Y}(\mathbf{h})} f(\mathbf{h}, \mathbf{y}'). \tag{10}$$

The objective function and its gradients can be efficiently computed by dynamic programming; for inference, we use the Viterbi algorithm to find the best tag path which maximizes the score.

4 Experiments

4.1 Datasets

We performed experiments on six standard datasets for sequence labeling tasks, i.e. named entity recognition, part-of-speech tagging, and syntactic chunking. To test the effectiveness of our proposed model, we do not use language-specific resources (such as gazetteers), external knowledge

Model	Spanish NER	Dutch NER	English NER	German NER	Chunking	PTB POS
Baseline	70.73±0.42	63.49±0.42	77.51±0.39	54.07±0.42	91.97±0.21	95.76±0.13
+ char-LSTM	79.93±0.43	77.16±0.47	83.98±0.46	64.29±0.47	93.31±0.23	97.14±0.11
+ char-CNN	79.78±0.41	76.43±0.48	83.85±0.38	63.53±0.41	92.67±0.24	97.02±0.12
+ char-CNN-5	79.63±0.38	**76.92**±0.42	83.60±0.39	**64.26**±0.42	**93.11**±0.26	**97.15**±0.12
+ char-CNN-9	79.25±0.56	74.82±0.46	83.31±0.47	63.97±0.46	**92.92**±0.27	**97.13**±0.13
+ char-ResNet-9	74.34±0.45	**76.54**±0.39	**83.91**±0.42	**66.15**±0.44	**93.85**±0.24	96.99±0.15
+ char-DenseNet-9	78.25±0.52	**76.71**±0.53	**84.16**±0.41	**67.54**±0.46	**93.82**±0.25	**97.13**±0.11
+ char-IntNet-9	78.53±0.44	**76.93**±0.47	83.83±0.44	**70.11**±0.41	**93.94**±0.26	**97.19**±0.12
+ char-IntNet-5	**80.44**±0.43	**78.06**±0.45	**85.34**±0.39	**69.48**±0.42	**94.27**±0.23	**97.23**±0.11

Table 1: F1 score of different character-to-word models.

(such as transfer learning, joint training), hand-crafted features, or any character preprocessing, we do not replace any rare words into `UNKNOWN`.

Named entity recognition. CoNLL-2002 and CoNLL2003 datasets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) contain named entity labels for Spanish, Dutch, English and German as separate datasets. These four datasets contain different types of named entities: locations, persons, organizations, and miscellaneous entities. Unlike some approaches, we do not combine the validation set with the training set. Although POS tags were made available for these datasets, we do not leverage those as additional information which sets our approach apart from that of transfer learning.

Part-of-speech tagging. The Wall Street Journal (WSJ) portion of Penn Treebank (PTB) (Marcus et al., 1993) contains 25 sections and categorizes each word into one out of 45 POS tags. We adopt the standard split and use sections 0-18 as training data, sections 19-21 as development data, and sections 22-24 as test data.

Syntactic chunking. The CoNLL 2000 chunking task (Tjong Kim Sang and Buchholz, 2000) uses sections 15-18 from the Wall Street Journal corpus for training and section 20 for testing. It defines 11 syntactic chunk types (e.g., NP, VP, ADJP), we adopt the standard split and sample 1000 sentences from the training set as the development set.

4.2 Training Settings

Initialization. The size of the dimensions of character embeddings is 32 which are randomly initialized using a uniform distribution. We adopt the same initialization method for randomly initialized word embeddings that are updated during training. For IntNet, the filter size of the initial convolution is 32 and that of other convolutions is 16. We have used filters of size $[3, 4, 5]$ for all the kernels. The number of convolutional layers are 5 and 9 for IntNet-5 and IntNet-9, respectively, and we have adopted the same weight initialization as that of ResNet. We use pre-trained word embeddings for initialization, GloVe (Pennington et al., 2014) 100-dimension word embeddings for English, and fastText (Bojanowski et al., 2017) 300-dimension word embeddings for Spanish, Dutch, and German. The state size of the bi-directional LSTMs is set to 256. We adopt standard BIOES tagging scheme for NER and Chunking.

Optimization. We employ mini-batch stochastic gradient descent with momentum. The batch size, momentum and learning rate are set to 10, 0.9 and $\eta_t = \frac{\eta_0}{1+\rho t}$, where η_0 is the initial learning rate 0.01 and $\rho = 0.05$ is the decay ratio, the value of gradient clipping is 5. Dropout is applied on the input of IntNet, LSTMs, and CRF, and its ratio 0.5 is fixed, but with no dropout inside of IntNet.

4.3 Compared Methods

To address those open questions in Section 1, we conduct detailed experiments and empirical comparisons on different state-of-the-art character embedding models across different domains. Firstly, we use LSTM-CRF with randomly initialized word embeddings as our initial baseline. We adopt two state-of-the-art methods in sequence labeling, denoted as char-LSTM (Lample et al., 2016) and char-CNN (Ma and Hovy, 2016). We add more layers to the char-CNN model and refer to that as char-CNN-5 and char-CNN-9, respectively for 5 and 9 convolutional layers. Furthermore, we add residual connections to the char-CNN-9 and refer it as char-ResNet. Also, we apply 3 dense blocks based on char-ResNet which we refer to as char-DenseNet, to compare the difference between residual connection and dense connection. Lastly, we refer to our proposed

Model	Spanish	Dutch	English	German	Chunking	POS
Conv-CRF+Lexicon (Collobert et al., 2011)	-	-	89.59	-	94.32	97.29
LSTM-CRF+Lexicon (Huang et al., 2015)	-	-	90.10	-	94.46	97.43
LSTM-CRF+Lexicon+char-CNN (Chiu and Nichols, 2016)	-	-	90.77	-	-	-
LSTM-Softmax+char-LSTM (Ling et al., 2015)	-	-	-	-	-	97.55
LSTM-CRF+char-LSTM (Lample et al., 2016)	85.75	81.74	90.94	78.76	-	-
LSTM-CRF+char-CNN (Ma and Hovy, 2016)	-	-	91.21	-	-	97.55
GRM-CRF+char-GRU (Yang et al., 2017)	84.69	85.00	91.20	-	94.66	97.55
LSTM-CRF	80.33±0.37	79.87±0.28	88.41±0.22	73.42±0.39	94.29±0.11	96.63±0.08
LSTM-CRF+char-LSTM	86.12±0.34	87.13±0.25	91.13±0.15	78.31±0.35	94.97±0.09	97.49±0.04
LSTM-CRF+char-CNN	85.91±0.38	86.69±0.22	91.11±0.14	78.15±0.31	94.91±0.08	97.45±0.03
LSTM-CRF+char-IntNet-9	85.71±0.39	**87.38**±0.27	**91.39**±0.16	**79.43**±0.33	**95.08**±0.07	97.51±0.04
LSTM-CRF+char-IntNet-5	**86.68**±0.35	**87.81**±0.24	**91.64**±0.17	78.58±0.32	**95.29**±0.08	**97.58**±0.02

Table 2: F1 score of our proposed models in comparison with state-of-the-art results.

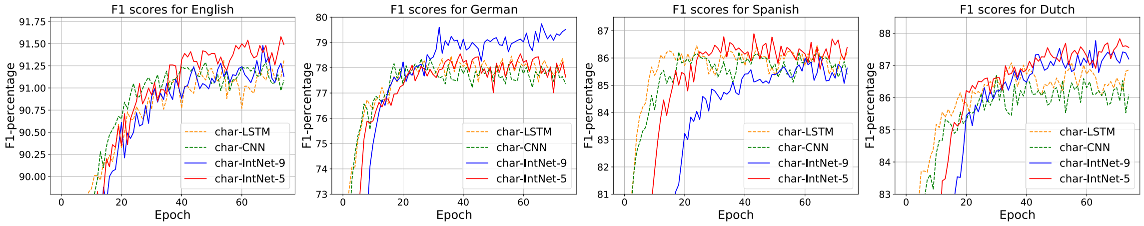

Figure 2: Training details of different models for English, German, Spanish, and Dutch.

model, which uses different convolution layers, as char-IntNet-5 and char-IntNet-9.

5 Results and Analysis

5.1 Character-to-word Models

Table 1 presents the performance of different character-to-word models on six benchmark datasets. For sequence labeling, char-LSTM and char-CNN are current state-of-the-art character embedding models for learning character-to-word representations. We observe that char-LSTM performs better than char-CNN in most cases, however, char-CNN uses a convolution layer followed by max pooling as a shallow feature extractor, that does not explore the full potential of CNNs.

Therefore, we implement two variations based on char-CNN, referred to as char-CNN-5 and char-CNN-9. The result shows that for most of the datasets, the F1 score does not improve much when we directly add more layers. We also observe some accuracy drop when we continuously increase the depth. This confirms why most CNN-based approaches for learning representations on words are shallow, which is very different from learning representations for sentences. Furthermore, we add residual connections to char-CNN-9 as char-ResNet-9, which confirms that residual connections can help train deep layers. We further improve char-ResNet-9 by changing residual connections into dense connection blocks as char-DenseNet-9, which shows that the dense connections are better than residual connections for learning word shape information.

Our proposed character-to-word model, char-IntNet-5 and char-IntNet-9 generally improves the results across all datasets. Our IntNet significantly outperforms other character embedding models, for example, the improvement is more than 2% in terms of F1 score for German and Dutch. Also, we observe that char-IntNet-5 is more effective for learning character-to-word representations than char-IntNet-9 in most of the cases. The only exception is German which seems to require a deeper and wider model for learning better representations.

5.2 State-of-the-art Results

Table 2 presents our proposed model in comparison with state-of-the-art results. LSTM-CRF is our baseline which uses fine-tuned pre-trained word embeddings. Its comparison with LSTM-CRF using random initializations for word embeddings, as shown in Table 1, confirms that pre-trained word embeddings are useful for sequence labeling. Since the training corpus for sequence labeling is relatively small, pre-trained embeddings learned from a huge unlabeled corpus can help to enhance word semantics. Furthermore, we adopt and re-implement two state-of-the-art character models, char-LSTM and char-CNN, by combining with LSTM-CRF, which we

	Model	English				German				Spanish				Dutch			
		IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
Dev	char-LSTM	97.15	89.87	89.41	87.07	86.97	85.80	68.35	64.76	89.63	89.06	78.14	74.13	94.50	87.98	80.00	72.37
	char-CNN	97.10	90.04	95.45	88.02	87.45	86.13	57.14	63.28	88.93	88.85	72.90	71.96	94.54	87.27	74.55	68.77
	char-IntNet-9	96.86	**90.52**	**91.95**	**90.16**	**87.92**	85.29	**76.07**	**67.98**	88.43	**88.58**	74.53	**72.09**	93.68	87.49	**89.09**	**75.58**
	char-IntNet-5	96.65	**90.14**	88.10	**88.31**	87.21	85.00	67.10	64.17	88.56	88.47	**78.90**	70.23	**94.63**	**88.56**	**89.09**	**74.40**
Test	char-LSTM	93.68	92.48	100.00	82.64	86.97	83.95	69.67	62.74	87.19	87.79	95.29	76.01	95.13	83.00	78.26	72.34
	char-CNN	93.85	92.65	100.00	84.09	64.72	83.67	69.67	58.19	87.81	88.46	87.96	73.68	94.25	82.50	73.27	73.37
	char-IntNet-9	93.79	**94.94**	100.00	82.31	**87.56**	**83.85**	**74.33**	**65.75**	87.08	87.98	**95.29**	**77.16**	94.42	**83.85**	**85.02**	**75.46**
	char-IntNet-5	**93.94**	**92.72**	100.00	**83.91**	**87.11**	83.60	67.22	60.92	87.19	**88.42**	**97.38**	**78.02**	94.71	**84.84**	**82.13**	**76.99**

Table 3: F1 score of different models for IV, OOTV, OOEV and OOBV.

Model	Frequent Words			Rare Words			OOV Words		
	newspapers	*slipped*	*world*	*Commerce*	*youthful*	*sessions*	11-*month*	*Thursdays*	*undetermined*
char-LSTM	enclosures	stirred	wolrd	Committee	luthier	cessions	19-month	Thousands	undereducated
	nelsonville	clipped	worde	Computer	loughmoe	sensible	10-month	Tunbridge	underpinned
	entrances	snipped	lowed	Comments	wrathful	stepanos	12-month	Standings	undermined
	newpapers	striped	wowed	Corrects	slothful	stefanos	14-month	Torrance	underlined
	necklaces	stifled	crowd	Clippers	ephorus	constans	11-inch	Phillies	underprepared
char-CNN	newspaper	slipper	worli	Committee	mouthful	suppressions	31-month	Thursday	determined
	newspapermen	slippy	worle	Community	eeyou	oppressions	51-month	Wednesday	overdetermined
	newpapers	stripped	worse	Commodities	mouthfeel	digressions	1-month	Tuesday	determinist
	nitrification	shipped	werle	Communist	motul	confessions	21-month	Ecuador	determiners
	megaphones	stopped	wereld	Comments	yourself	fissions	41-month	Windass	determiner
char-IntNet	newspapers	blipped	eworld	Commissioner	mouthful	recessions	55-month	Thursday	undermined
	wallpapers	unclipped	offworld	Commodities	mirthful	accessions	51-month	Saturday	determined
	escapers	tripped	homeworld	Clarence	mouthfuls	missions	22-month	thursdays	overdetermined
	carcases	dripped	linuxworld	Commission	youths	conversions	25-month	Tuesday	unexamined
	spacers	slopped	westworld	Commons	slothful	possessions	12-month	tuesdays	predetermined

Table 4: Nearest neighbours of different models for frequent words, rare words and OOV words.

refer to as LSTM-CRF-char-LSTM and LSTM-CRF-char-CNN. Lastly, we combine our proposed model with LSTM-CRF which we refer to as LSTM-CRF-char-IntNet-9 and LSTM-CRF-char-IntNet-5.

These experiments show that our char-IntNet generally improves results across different models and datasets. The improvement is more pronounced for non-English datasets, for example, IntNet improves the F-1 score over the state-of-the-art results by more than 2% for Dutch and Spanish. It also shows that the results of LSTM-CRF are significantly improved after adding character-to-word models, which confirms that word shape information is very important for sequence labeling. Figure 2 presents the details of training epochs in comparison with other state-of-the-art character models for different languages. It shows that char-CNN and char-LSTM converge early whereas char-IntNet takes more epochs to converge and generally performs better. It alludes to the fact that IntNet is suitable for reducing over-fitting, since we have used early stopping while training.

5.3 Rare and OOV Words Analysis

Another advantage of learning internal structure of words is that it can capture representations for out-of-vocabulary (OOV) words. To better understand the behavior of IntNet, Table 3 presents error analysis on in-vocabulary words (IV), out-of-training-vocabulary words (OOTV), out-of-embedding-vocabulary words (OOEV), and out-of-both-vocabulary words (OOBV) compared to different character models. The result shows that our proposed model significantly outperforms other character models on OOV words including OOTV, OOEV, and OOBV. For example, in OOBV category, our IntNet outperforms other models by more than 3% in terms of F1 score for Dutch and German datasets.

Furthermore, we present comparisons of nearest neighbors with different models for frequent words, rare words, and OOV words. Table 4 shows the results of nearest neighbors for learning word shape information, which gives insights on what kind of character-to-word representations can be learned by different models. For example, in OOV words, our IntNet model learns a better *xx-month* shape pattern when matching 11-*month* compared to other models.

5.4 Discussion

In many situations, learning character-to-word representations of subword sequences that exceed the typical length of word shape pattern or morpheme sequences might result in noise. RNNs can capture longer sequences in theory, however, longer sequences do not guarantee better results when learning prefixes and suffixes. The funnel-shaped wide architecture of IntNet, uses different kernels with different levels of concatenation to capture patterns of different subword lengths and that is flexible than char-LSTM and char-CNN. For example, Table 4 shows $Thursday$ in OOV words, our model learns a better word-shape structure for character-to-word representations compared to other methods.

When considering training time, IntNet is only 20% slower than char-CNN for the whole training process. Also, learning word representations use fewer parameters than learning sentence representations. Therefore, the impact of training speed for sequence labeling is limited. The inference time of IntNet is almost the same as char-CNN.

6 Conclusion

We presented empirical comparisons of different character embedding models for learning character-to-word representations and investigated the gaps between methods for learning representations of words and sentences. We conducted detailed experiments of different state-of-the-art convolutional models, and explored the advantages and disadvantages of their components for learning word shape information. Furthermore, we presented IntNet, a funnel-shaped wide convolutional neural architecture with no down-sampling that focuses on learning better internal structure of words by composing their characters from limited supervised training corpora. Our in-depth analysis showed that a shallow wide architecture is better than a narrow deep architecture for learning character-to-word representations. Omitting down-sampling operations is useful for capturing the sequential patterns of prefixes and suffixes. Our proposed compositional character-to-word model does not leverage any external resources, hand-crafted features, additional knowledge, joint training, or character-level preprocessing, and achieves new state-of-the-art performance for various sequence labeling tasks, including named entity recognition, part-of-speech tagging and syntactic chunking. In the future, we would like to explore using the IntNet model for other NLP tasks.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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

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

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1107–1116, Valencia, Spain.

Ryan Cotterell and Georg Heigold. 2017. Cross-lingual character-level neural morphological tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 748–759. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

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

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. 2016. Deep networks with stochastic depth. In *Proceedings of European Conference on Computer Vision*.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, volume 1, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT 2016*, pages 260–270.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank F. Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1064–1074, Berlin, Germany.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, volume 14, pages 1532–1543. Association for Computational Linguistics.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1756–1765.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 412–418.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 2121–2130.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1818–1826.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex A. Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Erik F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *Proceedings of the International Conference on Learning Representations*.

Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, pages 87.1–87.12.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28*, pages 649–657.

ICON: Interactive Conversational Memory Network for Multimodal Emotion Detection

Devamanyu Hazarika[1], **Soujanya Poria**[2] , **Rada Mihalcea**[3],
Erik Cambria[2] and **Roger Zimmermann**[1]
[1] School of Computing, National University of Singapore
[2] School of Computer Science and Engineering, NTU, Singapore
[3] Computer Science & Engineering, University of Michigan, Ann Arbor
`{hazarika|rogerz}@comp.nus.edu.sg, {sporia|cambria}@ntu.edu.sg,`
`mihalcea@umich.edu`

Abstract

Emotion recognition in conversations is crucial for building empathetic machines. Current work in this domain do not explicitly consider the inter-personal influences that thrive in the emotional dynamics of dialogues. To this end, we propose Interactive COnversational memory Network (ICON), a multimodal emotion detection framework that extracts multimodal features from conversational videos and hierarchically models the *self*- and *inter-speaker* emotional influences into global memories. Such memories generate contextual summaries which aid in predicting the emotional orientation of utterance-videos. Our model outperforms state-of-the-art networks on multiple classification and regression tasks in two benchmark datasets.

1 Introduction

Emotions play an important role in our daily life. A long-standing goal of AI has been to create affective agents that can detect and comprehend emotions. Research in affective computing has mainly focused on understanding affect (emotions and sentiment) in monologues. However, with increasing interactions of humans with machines, researchers now aim at building agents that can seamlessly analyze affective content in conversations. This can help in creating empathetic dialogue systems, thus improving the overall human-computer interaction experience (Young et al., 2018).

Analyzing emotional dynamics in conversations, however, poses complex challenges. This is due to the presence of intricate dependencies between the affective states of speakers participating in the dialogue. In this paper, we address the problem of emotion recognition in conversational videos. We specifically focus on dyadic conversations where two entities participate in a dialogue.

We propose Interactive COnversational memory Network (ICON), a multimodal network for identifying emotions in utterance-videos. Here, utterances are units of speech bounded by breaths or pauses of the speaker. Emotional dynamics in conversations consist of two important properties: *self* and *inter-personal dependencies* (Morris and Keltner, 2000). Self-dependencies, also known as *emotional inertia*, deal with the aspect of emotional influence that speakers have on themselves during conversations (Kuppens et al., 2010). On the other hand, inter-personal dependencies relate to the emotional influences that the counterparts induce into a speaker. Conversely, during the course of a dialogue, speakers also tend to mirror their counterparts to build rapport (Navarretta et al., 2016).

Figure 1 demonstrates a sample conversation from the dataset involving both self and inter-personal dependencies. While most conversational frameworks only focus on self dependencies, ICON leverages both such dependencies to generate affective summaries of conversations. First, it extracts multimodal features from all utterance-videos. Next, given a test utterance to be classified, ICON considers the preceding utterances of both speakers falling within a context-window and models their self-emotional influences using local gated recurrent units (GRUs).

Furthermore, to incorporate inter-speaker influences, a global representation is generated using a GRU that intakes output of the local GRUs. For each instance in the context-window, the output of this global GRU is stored as a memory cell. These memories are then subjected to multiple read/write cycles that include attention mechanism for generating contextual summaries of the conversational history. At each iteration, the representation of the test utterance is improved with this summary representation and finally used for prediction.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2594–2604
Brussels, Belgium, October 31 - November 4, 2018.

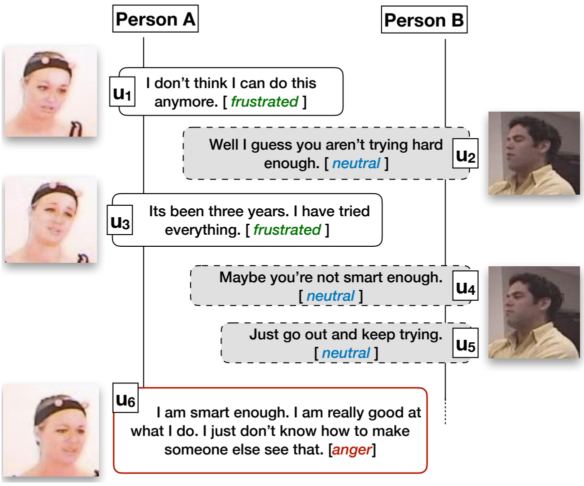

Figure 1: An abridged dialogue from the IEMOCAP dataset. P_a is frustrated over her long term unemployment and seeks encouragement (u_1, u_3). P_b, however, is pre-occupied and replies sarcastically (u_4). This enrages P_a to appropriate an angry response (u_6). In this dialogue, *emotional inertia* is evident in P_b who does not deviate from his nonchalant behavior. P_a, however, gets emotionally influenced by her counterpart. This influence is content-based, not label-based.

The contributions of this paper are as follows:

- We propose ICON, a novel model for emotion recognition that incorporates self and inter-speaker influences in a dialogue. Memory networks are used to model contextual summaries for prediction.
- We introduce a multimodal approach that provides comprehensive features from modalities such as language, visual, and audio in utterance-videos.
- ICON can be considered as a generic framework for conversational modeling that can be extended to multi-party conversations.
- Experiments on two benchmark datasets show that ICON significantly outperforms existing models on multiple discrete and continuous emotional categories.

The remainder of the paper is organized as follows: Section 2 presents related works; Section 3 formalizes the problem statement and Section 4 describes our proposed approach; Section 5 provides details on experimental setup; Section 6 reports the results and related analysis; finally, Section 7 concludes the paper.

2 Related Works

Emotion recognition is an interdisciplinary field of research with contributions from psychology, cognitive science, machine learning, natural language processing, and others (Picard, 2010).

Initial research in this area primarily involved visual and audio processing (Ekman, 1993; Datcu and Rothkrantz, 2008). The role of text in emotional analysis became evident with later research such as Alm et al. (2005); Strapparava and Mihalcea (2010). Current research in this domain is mainly performed from a multimodal learning perspective (Poria et al., 2017a; Baltrušaitis et al., 2018). Numerous previous approaches have relied on fusion techniques that leverage multiple modalities for affect recognition (Soleymani et al., 2012; Zadeh et al., 2017; Chen et al., 2017; Tzirakis et al., 2017; Zadeh et al., 2018b).

Understanding conversations is crucial for machines to replicate human language and discourse. Emotions play an important role in shaping such social interactions (Ruusuvuori, 2013). Richards et al. (2003) attribute emotional dynamics to be an interactive phenomena, rather than being within-person. We utilize this trait in the design of our model that accommodates inter-personal dynamics. Being a temporal event, context also plays an important role in conversational analysis. Poria et al. (2017b) use contextual information from neighboring utterances of the same speaker to predict emotions. However, there is no provision to model interactive influences. Work by Yang et al., 2011; Xiaolan et al., 2013 stresses the study of patterns for emotion transitions. In contrast, we posit the use of utterance content to model context with multimodal features.

In the literature, memory networks have been successfully applied in many areas, including question-answering (Weston et al., 2014; Sukhbaatar et al., 2015; Kumar et al., 2016), machine translation (Bahdanau et al., 2014), speech recognition (Graves et al., 2014), and others. In emotional analysis, Zadeh et al. (2018a) propose a memory-based sequential learning for multi-view signals. Although we utilize memory networks, our work is different as we use memories to encode whole utterances. Also, each memory cell in our network is processed using GRUs to capture temporal dependencies. This technique deviates from the traditional use of embedding matrices to encode information into memory cells.

ICON builds on our previous research (Hazarika et al., 2018) that used separate memory networks for both interlocutors participating in a dyadic conversation. In contrast, ICON adopts an interactive scheme that actively models inter-speaker emotional dynamics with fewer trainable parameters.

3 Problem Setting

Let us define a conversation U to be a set of asynchronous exchange of utterances between two persons P_a and P_b over time. With T utterances, $U = \{u_1, u_2, ..., u_T\}$ is a totally ordered set which can be arranged as a sequence $(u_1, ..., u_T)$ based on temporal occurrence. Here, each utterance u_i is spoken by either P_a or P_b. Furthermore, for each $\lambda \in \{a, b\}$, U_λ denotes person P_λ's individual utterances in U, i.e., $U_\lambda = \{u_i \mid u_i \in U \text{ and } u_i \text{ spoken by } P_\lambda, \ \forall i \in [1, |U|]\}$. This provides two sets of utterances for both the respective speakers, such that $U = U_a \cup U_b$.

Our aim is to identify the emotions of utterances in conversational videos. At each time step $t \in [1, T]$ of video U, our model is provided with the utterance spoken at that time, i.e. u_t, and tasked to predict its emotion. Moreover, we also utilize the previous utterances within U spoken by both persons. Considering a context-window of size K, the preceding utterances of P_a and P_b (starting with the most recent) within this context-window can be represented by H_a and H_b, respectively. Formally, for each $\lambda \in \{a, b\}$, H_λ is created as,

$$H_\lambda = \{u_i \mid i \in [t - K, t - 1] \text{ and } u_i \in U_\lambda\} \quad (1)$$

$$\text{and} \quad |H_a| + |H_b| \leq K \quad (2)$$

Table 1 provides a sample conversation with a context-window of size $K = 5$.

U	$\{ u_1^a, u_2^a, u_3^b, u_4^a, u_5^a, u_6^b \}$
U_a, U_b	$\{ u_1, u_2, u_4, u_5 \}, \{ u_3, u_6 \}$
test utterance	u_7^a
H_a, H_b	$\{ u_2, u_4, u_5 \}, \{ u_3, u_6 \}$

Table 1: Sample conversation U with test utterance u_7. Context-window $K = 5$. Here, $u_i^\lambda = i^{th}$ utterance by P_λ.

4 Methodology

ICON has been designed as a generic framework for affective modeling of conversations. Its computations can be categorized as a sequence of four successive modules: Multimodal Feature Extraction, Self-Influence Module, Dynamic Global-Influence Module, and Multi-hop Memory. Figure 2 illustrates the overall model.

4.1 Multimodal Feature Extraction

ICON adopts a multimodal framework and performs feature extraction from three modalities, i.e., language (transcripts), audio and visual.

These features are extracted for each utterance in the conversation and their concatenated vectors serve as the utterance representations. The motivation of this setup derives from previous works that demonstrate the effectiveness of multimodal features in creating rich feature representations (D'mello and Kory, 2015). These features provide complementary information from heterogeneous sources which helps to accumulate comprehensive features. Its need is particularly pronounced in videos as they are often plagued with noisy signals and missing-information within individual modalities (e.g., facial occlusion, loud background music, imperfect transcriptions).

4.1.1 Textual Features

We employ a convolutional neural network (CNN) to extract textual features from the transcript of each utterance. CNNs are capable of learning abstract semantic representations of a sentence based on its words and n-grams (Kalchbrenner et al., 2014). For our purpose, we utilize a simple CNN with a single convolutional layer followed by max-pooling (Kim, 2014). The input to this network consists of pre-trained word embeddings extracted from the 300-dimensional FastText embeddings (Bojanowski et al., 2016). The convolution layer consists of three filters with sizes f_t^1, f_t^2, f_t^3 with f_{out} feature maps each. We perform 1D convolutions using these filters followed by max-pooling on its output. The pooled features are finally projected onto a dense layer with dimension d_t and its activations are used as the textual representation $\boldsymbol{t_u} \in \mathbb{R}^{d_t}$.

4.1.2 Audio Features

Audio plays a significant role in determining the emotional states of a speaker (De Silva and Ng, 2000; Song et al., 2004). To extract audio features, we first format the audio of each utterance-video as a 16-bit PCM WAV file and use the open-sourced software openSMILE (Eyben et al., 2010). This tool provides high dimensional vectors for audio files that summarizes important statistical descriptors such as loudness, pitch, Mel-spectra, MFCC, etc. Specifically, we use the *IS13_ComParE*[1] extractor which provides 6373 features for each utterance. The features are then normalized using Min-Max scaling followed by L2-based feature selection. This selection provides low-dimensional audio features $\boldsymbol{a_u} \in \mathbb{R}^{d_a}$ of dimensions d_a.

---

[1] `http://audeering.com/technology/opensmile`

4.1.3 Visual Features

Visual indicators such as facial expressions are key to understand emotions. In our work, we use a deep 3D-CNN to model spatiotemporal features of each utterance video (Tran et al., 2015). 3D-CNN helps to understand emotional concepts such as smiling or frowning that are often spread across multiple frames of a video with no predefined spatial location. The input to this network is a video with dimensions (c, h, w, f), where c is the number of channels, h, w are the height and width of each frame, with a total of f frames per video.

The network contains three blocks of convolution where each block contains two convolutional layers followed by max-pooling. For the convolution, 3D filters are employed having dimensions $(f_{out}, f_{in}, f_h, f_w, f_d)$, where, $f_{[out/in/h/w/d]}$ represents the number of feature maps, input channels, height, width, and depth of the filter, respectively. After a non-linear reLU activation (LeCun et al., 2015), max-pooling is performed using a sliding window of dimensions (m_p, m_p, m_p). For an input utterance video, the final features of the third convolutional block is mapped onto a dense layer of dimension d_v whose activations are used as the visual features $\boldsymbol{v_u} \in \mathbb{R}^{d_v}$.

4.1.4 Fusion

We generate the final representation of an utterance u by concatenating all three multimodal features:

$$\boldsymbol{u} = tanh((W^f[\boldsymbol{t_u}; \boldsymbol{a_u}; \boldsymbol{v_u}]) + \boldsymbol{b}^f) \qquad (3)$$

Concatenation is one of the most common fusion methods (Shwartz et al., 2016). Its simplicity also allows us to emphasize the contribution of the remaining components of ICON.

4.2 SIM: Self-Influence Module

Given a test utterance u_t to be classified, this module independently processes the histories of both speakers. SIM consists of two GRUs, GRU_a^s and GRU_b^s, for H_a and H_b, respectively. For each $\lambda \in \{a, b\}$, GRU_λ^s attempts to model the *emotional inertia* of speaker P_λ which represents the emotional dependency of a speaker with their own previous states. In particular, for each historical utterance $\boldsymbol{u}_{i<t} \in H_\lambda$, an internal memory state $\boldsymbol{h}_\lambda^{(j)}$ is computed by GRU_λ^s conditioned on utterance $\boldsymbol{u}_i$ and previous memory state $\boldsymbol{h}_\lambda^{(j-1)}$. This can be abbreviated as $\boldsymbol{h}_\lambda^{(j)} = GRU_\lambda^s(\boldsymbol{u}_i, \boldsymbol{h}_\lambda^{(j-1)})$.

Gated Recurrent Unit: GRUs are gated recurrent cells introduced by Cho et al. (2014). At time step j, GRU computes hidden state $s_j \in \mathbb{R}^{d_{em}}$ by calculating two gates, r_j (*reset gate*) and z_j (*update gate*) with j^{th} input $\boldsymbol{x}_j$ and previous state $\boldsymbol{s}_{j-1}$. The computations are:

$$\begin{aligned}
\boldsymbol{z}_j &= \sigma(V^z\boldsymbol{x}_j + W^z\boldsymbol{s}_{j-1} + \boldsymbol{b}^z) \\
\boldsymbol{r}_j &= \sigma(V^r\boldsymbol{x}_j + W^r\boldsymbol{s}_{j-1} + \boldsymbol{b}^r) \\
\boldsymbol{v}_j &= tanh(V^h\boldsymbol{x}_j + W^h(\boldsymbol{s}_{j-1} \otimes \boldsymbol{r}_j) + \boldsymbol{b}^h) \\
\boldsymbol{s}_j &= (1 - \boldsymbol{z}_j) \otimes \boldsymbol{v}_j + \boldsymbol{z}_j \otimes \boldsymbol{s}_{j-1}
\end{aligned}$$

In this work, input $\boldsymbol{x}_j = \boldsymbol{u}_i$ and $\boldsymbol{s}_j = \boldsymbol{h}_\lambda^{(j)}$. SIM computes both sequences $H_a^* \in \mathbb{R}^{d_{em} \times |H_a|}$ and $H_b^* \in \mathbb{R}^{d_{em} \times |H_b|}$ using the respective GRUs,

$$H_\lambda^* = [\boldsymbol{h}_\lambda^{(j)}]_{j=1}^{|H_\lambda|} = GRU_\lambda^s(H_\lambda)\ ,\ \lambda \in \{a, b\} \qquad (4)$$

4.3 DGIM: Dynamic Global Influence Module

Emotions are not only regarded as internal-psychological phenomena but also interpreted and processed communicatively through social interactions (Fiehler, 2002). Conversations exemplify such a scenario where inter-personal emotional influence persists. Theories in cognitive science also suggest the existence of *emotional contagion* that causes humans to mirror their counterpart's gesture, posture and emotional state (Chartrand and Bargh, 1999; Navarretta et al., 2016). Additionally, these interactions occur dynamically through the discourse of a dialogue.

While modeling the contextual history, we incorporate such properties using a dynamic influence module. This module maintains a global representation of the conversation and updates it recurrently at each time step of the K-length conversation history. For any $k \in [1, K]$, the global state is updated using a GRU operation on the previous state $\boldsymbol{s}_{k-1}$ and current speaker P_λ's SIM memory $\boldsymbol{h}_\lambda^{(j)}$ for the corresponding spoken utterance $\boldsymbol{u}_{(t-K+k-1)}$, i.e., $\boldsymbol{h}_\lambda^{(j)} = GRU_\lambda^s(\boldsymbol{u}_{(t-K+k-1)})$.

Formally, DGIM consists of a GRU network, GRU^g, where the k^{th} global state $\boldsymbol{s}_k$ is computed as:

$$\boldsymbol{s}_k = \begin{cases} GRU^g(\boldsymbol{h}_a^{(j)}, \boldsymbol{s}_{k-1}), & \text{if } \boldsymbol{u}_{(t-K+k-1)} \in H_a \\ GRU^g(\boldsymbol{h}_b^{(j)}, \boldsymbol{s}_{k-1}), & \text{if } \boldsymbol{u}_{(t-K+k-1)} \in H_b \end{cases} \qquad (5)$$

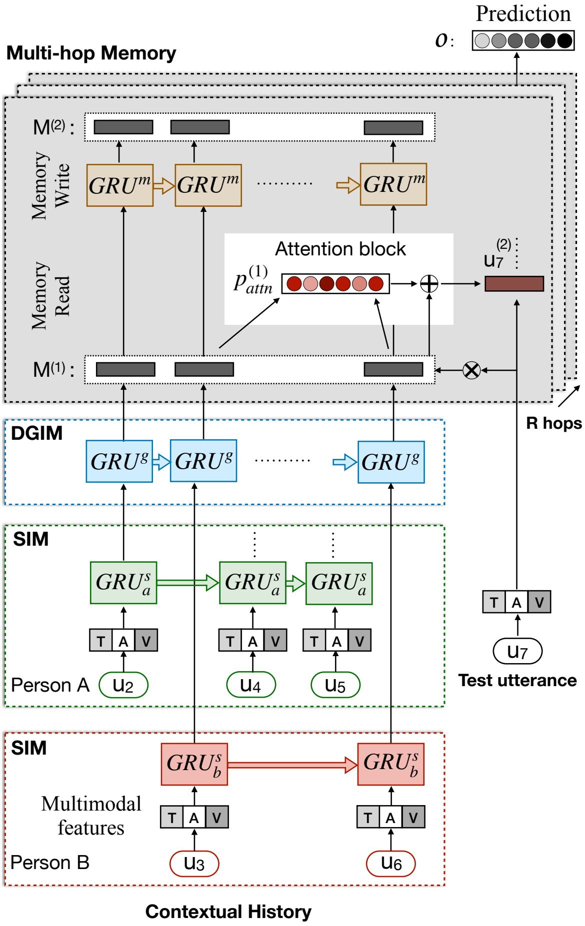

Figure 2: Illustration of ICON. Input conversation is as presented in Table 1.

4.4 Multi-hop Memory

The overall operation of the GRU^g produces a sequence of memories $M = [\boldsymbol{s}_1, ..., \boldsymbol{s}_K] \in \mathbb{R}^{d_{em} \times K}$. These memories incorporate dynamic influences from each of the K utterances spoken in the history. They serve as a contextual memory bank from which selective person-specific information can be incorporated into test utterance u_t to get discriminative features. To achieve this, a series of R memory read/write cycles are performed that are coupled with soft attention for refinement of u_t into a context-aware representation.

The need for multiple hops is inspired by recent works on memory networks (Kumar et al., 2016; Weston et al., 2014), which suggests the importance of multiple read/write iterations for performing transitive inference. Multiple hops also help in improving the focus of attention heads which might miss essential memories in a single hop. At the r^{th} hop, the computations are as follows:

- **Memory Read:** An attention mechanism is used to read the memories from r^{th} memory bank $M^{(r)}$ (Weston et al., 2014). First, each memory $\boldsymbol{m}_{\boldsymbol{k}}^{(r)} \in M^{(r)}$ is matched with test utterance $\boldsymbol{u}_t^{(r)}$ (initially, $\boldsymbol{u}_t^{(1)} = \boldsymbol{u}_t$ and $M^{(1)} = M$).

  This matching generates an attention vector $\boldsymbol{p}_{attn}^{(r)} \in \mathcal{R}^K$ whose k^{th} normalized score represents the relevance of k^{th} memory cell with respect to the test utterance. Inner product is used for the matching as follows:

  $$\boldsymbol{p}_{attn}^{(r)} = softmax(\ (M^{(r)})^T \boldsymbol{u}_t^{(r)}\ ) \tag{6}$$

  Where, $softmax(x_i) = e^{x_i} / \sum_j e^{x_j}$. These scores are then used to find a weighted representation of the memories as

  $$\boldsymbol{m}^{(r)} = \sum_{k=1}^{K} (\boldsymbol{p}_{attn}^{(r)})^k . (\boldsymbol{m}_k) = M^{(r)} \boldsymbol{p}_{attn}^{(r)} \tag{7}$$

  This vector denotes the summary of the context that is person-specific and based on the test utterance. Finally, the representation of the test utterance is updated by consolidating itself with the weighted memory $\boldsymbol{m}$ as:

  $$\boldsymbol{u}_t^{(r+1)} = tanh(\boldsymbol{m}^{(r)} + \boldsymbol{u}_t^{(r)}) \tag{8}$$

- **Memory Write:** After the read operation at each hop, memories are updated for the next hop. For this purpose, a GRU network, GRU^m, takes the r^{th} memory cells $M^{(r)}$ as input and reprocesses this sequence to generate memories $M^{(r+1)}$, i.e., $M^{(r+1)} = GRU^m(M^{(r)})$. Across all hops, this write operation can be viewed as that of a stacked recurrent neural network (RNN) where each level (or hop) improves the representational output of the RNN. The parameters of GRU^m are shared across all hops.

Final Prediction: We use the $(R+1)^{th}$ test utterance vector $\boldsymbol{u}_t^{(R+1)}$ and get the final prediction vector through its affine transformation,

$$\boldsymbol{o} = softmax(W^o \boldsymbol{u}_t^{(R+1)} + \boldsymbol{b}^o) \tag{9}$$

For classification, dimensions of vector $\boldsymbol{o}$ is the number of classes C, i.e., $\boldsymbol{o} \in \mathbb{R}^C$ and *categorical cross-entropy* loss is used as the cost measure for training. For regression, $\boldsymbol{o}$ is a scalar (without softmax normalization) whose scores are used to calculate the *mean squared error* cost metric.

Dataset	Fold	No. of Utterances	No. of Videos	Avg. history length
IEMOCAP	train/val	5810	120	36.54
	test	1623	31	39.00
SEMAINE	train/val	4368	63	43.61
	test	1430	32	45.61

* val = validation set.

Table 2: Summary of datasets. Note: Avg. history length represents the expected number of historical utterances available for any utterance in the dataset.

5 Experiments

5.1 Datasets

We perform experiments on two benchmark datasets in dialogue-based emotion detection: *IEMOCAP*[2] (Busso et al., 2008) and *SEMAINE*[3] (McKeown et al., 2012).

IEMOCAP is a database consisting of videos of dyadic conversations between pairs of 10 speakers. Grouped into five sessions, each pair is assigned with diverse scenarios for dialogues. Videos are segmented into utterances with annotations of fine-grained emotion categories. We consider six such categories for the classification task: *anger, happiness, sadness, neutral, excitement,* and *frustration*. The training set is curated using the first 8 speakers from session 1-4 while session 5 is used for testing.

SEMAINE is a video database of human-agent interactions. Here, users interact with characters whose responses are based on users' emotional state. Specifically, we utilize the AVEC 2012's *fully continuous sub-challenge* (Schuller et al., 2012) that requires predictions of four continuous affective dimensions: arousal, expectancy, power, and valence. The gold annotations are available for every 0.2 seconds in each video (Nicolle et al., 2012). However, to align with our problem statement, we approximate the utterance-level annotation as the mean of the continuous values within the spoken utterance. The sub-challenge provides standard training and testing splits which has been summarized in Table 2.

5.2 Training Details

20% of the training set is used as validation set for hyper-parameter tuning. We use the Adam optimizer (Kingma and Ba, 2014) for training the parameters starting with an initial learning rate of 0.001. Termination of the training-phase is decided by early-stopping with a patience of 10 epochs. The network is subjected to regularization in the form of Dropout (Srivastava et al., 2014) and Gradient-clipping for a norm of 40. Finally, the best hyper-parameters are decided using a grid-search. Their values are summarized in Table 3.

$(f_t^1, f_t^2, f_t^3) = (3, 4, 5)$	$f_{[h,w,d]} = 3$	$f_{out} = 64$
$d_{[t,a]} = 100$	$d_v = 512$	$d_{em} = 100$
$K = 40$	$R = 3$	

Table 3: Hyper-parameter values for the best model.

For multimodal feature extraction, we explore different designs for the employed CNNs. For text, we find the single layer CNN to perform at par with deeper variants. For visual features, however, a deeper CNN provides better representations. We also find that contextually conditioned features perform better than context-less features. Thus, in our experiments, we extract video-level contextual features for utterances from each modality using the network proposed by Poria et al. 2017b. These modified features are then used to form the multimodal utterance representations using equation 3.

5.3 Baselines

We compare our proposed model with multiple state-of-the-art networks in multimodal utterance-level emotion detection.

- *memnet* (Sukhbaatar et al., 2015) is an end-to-end memory network. For comparison, we modify our network to adopt their embedding-based memory-encoding in the multi-hop stage.
- *cLSTM*[4] (Poria et al., 2017b) classifies utterances using neighboring utterances (of same speaker) as context. LSTM is used for this purpose.
- *TFN*[5] (Zadeh et al., 2017) models intra- and inter-modality dynamics by explicitly aggregating uni-, bi- and trimodal interactions. Unlike cLSTM, contextual utterances are not considered.
- *MFN* (Zadeh et al., 2018a) performs multi-view learning by using Delta-memory Attention Network, a fusion mechanism to learn cross-view interactions. Similar to TFN, the modeling is performed within utterances.
- *CMN* (Hazarika et al., 2018) models separate contexts for both speaker and listener to an utterance. These contexts are stored as memories and combined with test utterance using attention mechanism.

---

[2] `http://sail.usc.edu/iemocap/`

[3] `http://sspnet.eu/avec2012/`

[4] `http://github.com/senticnet/contextual-sentiment-analysis`

[5] `http://github.com/A2Zadeh/TensorFusionNetwork`

Models	IEMOCAP: Emotion Categories													
	Happy		*Sad*		*Neutral*		*Angry*		*Excited*		*Frustrated*		***Avg.***	
	acc.	F1	acc.	F1	acc.	F1	acc.	F1	acc.	F1	acc.	F1	acc.	F1
memnet	24.4	33.0	60.4	69.3	56.8	55.0	67.1	66.1	65.2	62.3	68.4	63.0	59.9	59.5
cLSTM	25.5	**35.6**	58.6	69.2	56.5	53.5	70.0	66.3	58.8	61.1	67.4	62.4	59.8	59.0
TFN	23.2	33.7	58.0	68.6	56.6	55.1	69.1	64.2	63.1	62.4	65.5	61.2	58.8	58.5
MFN	24.0	34.1	65.6	70.5	55.5	52.1	**72.3**†	66.8	64.3	62.1	67.9	62.5	60.1	59.9
CMN	**25.7**	32.6	66.5	72.9	53.9	56.2	67.6	64.6	69.9	67.9	71.7	63.1	61.9	61.4
ICON	23.6	32.8	**70.6**†	**74.4**†	**59.9**	**60.6**†	68.2	**68.2**	**72.2**†	**68.4**	**71.9**	**66.2**†	**64.0**†	**63.5**†

Table 4: Performance of ICON on the IEMOCAP dataset. † represents statistical significance over state-of-the-art scores under the paired-t test ($p < 0.05$).

Models	SEMAINE							
	DV		DA		DP		DE	
	MAE	r	MAE	r	MAE	r	MAE	r
memnet	.20	.16	.21	.24	.21	.23	8.97	.05
cLSTM	**.18**	.14	.21	.23	.20	.25	8.90	-.04
TFN	.21	.01	.22	.10	.21	.12	9.19	.12
MFN	.19	.14	.20	.25	**.18**	.26	8.60	.15
CMN	**.18**	.23	.20	.30	**.18**	.26	8.89	-.02
ICON	**.18**	**.24**	**.19**	**.31**	**.18**	**.27**	**8.45**	**0.24**

Table 5: Performance on the SEMAINE dataset. Note: MAE = Mean Absolute Error, r = Pearson's correlation coefficient, DV = Valence, DA = Activation/Arousal, DP = Power, DE = Anticipation/Expectation.

Modality	IEMOCAP		SEMAINE			
	Emotions		DV	DA	DP	DE
	acc.	F1	r	r	r	r
T	58.3	57.9	.237	.297	.260	.225
A	50.7	50.9	.021	.082	.250	.035
V	41.2	39.8	.001	.068	.251	.001
A+V	52.0	51.2	.031	.122	.283	.050
T+A	63.8	63.2	.237	**.310**	.272	**.242**
T+V	61.4	61.2	.238	.293	.268	.239
T+A+V	**64.0**	**63.5**	**.243**	**.312**	**.279**	**.244**

Table 6: Comparison of the performance of ICON on both IEMOCAP and SEMAINE considering different modality combinations. Note: T=Text, A=Audio, V=Video

6 Results

Tables 4 and 5 present the results on the IEMOCAP and SEMAINE testing sets, respectively. In Table 4, we evaluate the mean classification performance using Weighted Accuracy (*acc.*) and F1-Score (*F1*) on the discrete emotion categories. ICON performs better than the compared models with significant performance increase in emotions (~2.1% *acc.*). For each emotion, ICON outperforms all the compared models except for *happiness* emotion. However, its performance is still at par with *cLSTM* without a significant gap. Also, ICON manages to correctly identify the relatively similar *excitement* emotion by a large margin.

In Table 5, evaluations of the four continuous labels from SEMAINE are performed using Mean Absolute Error (*MAE*) and Pearson's Correlation Coefficient (*r*). In all the labels, ICON attains improved performance over its counterparts, suggesting the efficacy of its context-modeling scheme.

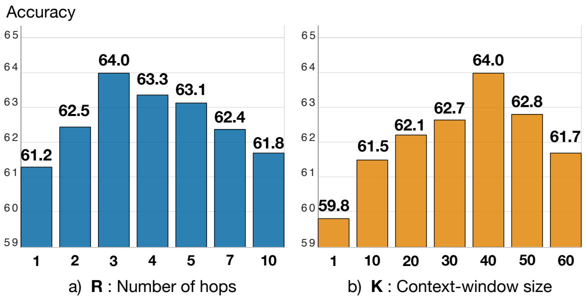

a) **R** : Number of hops

b) **K** : Context-window size

Figure 3: Trends in the performance of ICON on IEMOCAP dataset with varying R (hops) and K (Context-window size).

Hyperparameters: We plot the performance trends of ICON on the IEMOCAP dataset concerning the two main hyperparameters, R (number of hops) and K (context-window size). For R, the performance initially improves showing the importance of multiple hops in the memories. However, with a further increase, the hopping recurrence deepens and causes the *vanishing gradient* problem. This leads to decrease in performance. The best performance is obtained at $R = 3$. For K, similar trends are observed where performance improvement is seen by increasing the number of historical utterances. The best results are obtained for $K = 40$ which also aligns with the average number of historical utterances in the dataset (Table 2). Further increase in context does not provide relevant information and rather leads to performance degradation due to model confusion.

Multimodality: We investigate the importance of multimodal features for our task. Table 6 presents the results for different combinations of modes used by ICON on IEMOCAP. As seen, the trimodal network provides the best performance which is preceded by the bimodal variants. Among

ICON variants				IEMOCAP		SEMAINE			
				Emotions		DV	DA	DP	DE
	history	DGIM	hop	acc.	F1	r	r	r	r
1.	-	-	-	58.0	57.6	.10	.14	.10	.01
2.	self	✓	-	60.7	60.2	.17	.23	.15	.13
3.	dual	✓	-	61.2	60.7	.19	.24	.19	.20
4.	self	-	✓	61.9	61.3	.19	.23	.22	.20
5.	dual	-	✓	63.1	62.4	.21	.25	.26	.22
6.	self	✓	✓	62.2	61.7	.20	.28	.21	.22
7.	dual	✓	✓	**64.0**	**63.5**	**.24**	**.31**	**.27**	**.24**

Table 7: Ablation study for components of ICON.

unimodals, language modality performs the best, reaffirming its significance in multimodal systems. Interestingly, the audio and visual modality, on their own, do not provide good performance, but when used with text, complementary data is shared to improve overall performance.

6.1 Ablation Study

To check the importance of the modules present in ICON, we perform an ablation study where we remove constituent components and evaluate the model's performance. Table 7 provides the results on this study. In the first variant, none of the histories and the associated context-modeling is used. This provides the worst relative performance.

Self vs Dual History: We evaluate the scenarios where only self-history of the speaker is considered (variants 2, 4, and 6). Compared to the dual-history variants (variants 3, 5, and 7), these models provide lesser performance. Reasons involve the provision of partial information from the conversational histories. Similar trends can be seen for the *cLSTM* model in Table 4 which works in the same regime.

DGIM vs no-DGIM: Variants 4 and 5 do not contain the DGIM. In variant 5, separate memory banks are created for both histories ($M_a = H_a^*$ and $M_b = H_b^*$). Memory hops are also separately performed without parameter sharing. Absence of DGIM prevents the storage of dynamic influences between speakers at each historical time step and leads to performance deterioration.

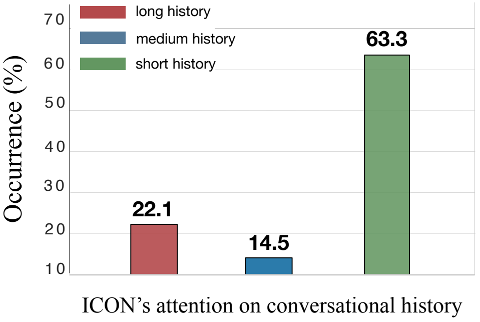

Figure 4: Distribution of top-attention by ICON on correctly classified instances in the testing set.

Multi-hop vs No-hop: Variants 2 and 3 represent cases where multi-hop is omitted, i.e., $R = 1$. Performance for them are poorer than variants having multi-hop mechanism (variants 4-7). Also, removal of multi-hop leads to worse performance than the removal of DGIM. This suggests that multi-hop is more crucial than the latter. However, best performance is achieved by variant 6 which contains all the proposed modules in its pipeline.

6.2 Dependency on distant history

For all the test utterances of IEMOCAP correctly classified by ICON, we analyze the global memories receiving the highest attention. First, we divide the conversational history (context-length $K = 40$) into three regions: *long*, *short*, and *medium*. Figure 4 provides a summary of how much the model attends each of these regions. The short region (labeled green) covering 10 utterances, corresponds to conversational history just preceding the test utterance. Utterances which occur more than 30 time steps behind the current test utterance are considered part of the long region (labeled red). Remaining utterances in between fall on the medium region (labeled blue).

The distribution of top-valued attention scores across the histories reveal interesting insights. Most of the correctly classified instances focus on the immediate or short history. In other words, 63% of the time, at least one of the top-5 attention value belongs to a memory in the short-history range. A significant share is also present for distant history (22%). This result indicates the presence of long-term emotional dependencies and the need to consider histories far away from the current test utterance.

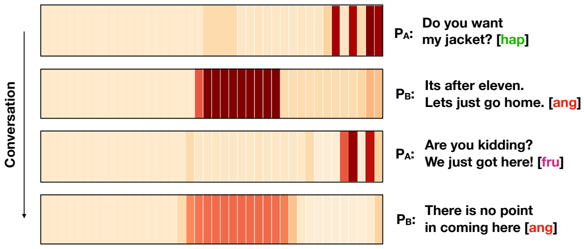

Figure 5: As a conversation develops, different speakers induce different affective bias which reflects in the memory selection for generation of the summaries.

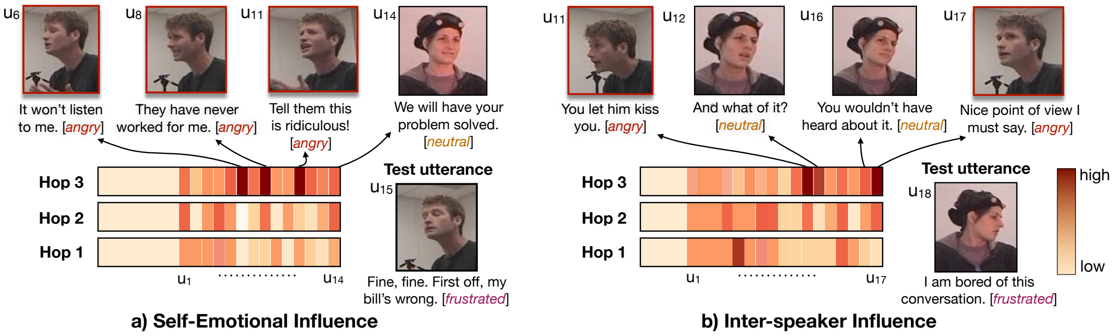

Figure 6: Case studies for emotional influence. 20 memories in the history which are nearest to test utterance, i.e. $k \in [21, 40]$ are visualized from the trained ICON.

6.3 Dynamic Modeling of Global Memories:

ICON holds the capability to model dynamic interactions between speakers. The memories by its DGIM (§4.3) are used to create summaries conditioned on the test utterance. Consequently, these summaries contain characteristics that are specific to the affective state of the current speaker (of the test utterance).

Figure 5 presents a sample slice of conversation from the dataset. As seen, summary selection for Person A varies from Person B. Such differences arise due to person-specific characteristics and unique affective interpretations of the conversation. Apart from the inter-speaker variance, the emotional state of a speaker also varies across turns.

6.4 Case Studies

To understand ICON's behavior while processing the global memories through multi-hop, we manually explore the utterances in the testing set of IEMOCAP. Figure 6 presents two cases which provide traces of self and inter-personal emotional influences and were correctly classified by ICON. Both the figures show the trend where multiple hops gradually improve the focus of attention mechanism on relevant memories.

In Figure 6a, person P_a registers a complaint to an operator P_b. Throughout the dialogue, P_a maintains an angry demeanor while P_b remains calm and neutral (u_{14}). While classifying utterance u_{15}, ICON focuses more on the histories uttered by P_a (u_6, u_8, and u_{11}). This demonstrates ICON's ability to model self-emotional influences. It should be noted that emotion of P_a here also depends on the utterances of P_b but compared to self-utterances, this dependency is much less. Figure 6b presents another scenario where a couple argue over an alleged affair.

A man (P_b) is angry over this fact and questions his partner (P_a) asking for details. The woman tries to behave unperturbed by providing neutral responses (u_{12}, u_{16}) but is eventually affected by P_b's continuous anger and expresses a frustrated response (u_{18}). These characteristics are captured by the attention mechanism applied on the global memories (generated by DGIM), which finds contextual information from histories that are relevant to the test utterance u_{18}. This example displays the role of inter-speaker influences and how ICON processes such dependencies.

7 Conclusion

In this paper, we presented ICON, a multimodal framework for emotion detection in conversations. ICON capitalizes on modeling contextual information that incorporates self and inter-speaker influences. We accomplish this by using an RNN-based memory network with multi-hop attention modeling. Experiments show that ICON outperforms state-of-the-art models on multiple benchmark datasets. Extensive evaluations and case studies demonstrate the effectiveness of our proposed model. Additionally, the ability to visualize the attentions brings a sense of interpretability to the model, as it allows us to investigate which utterances in the conversational history provide important emotional cues for the current emotional state of the speaker.

In the future, we plan to test ICON on other relevant dialogue-based applications and also use it for empathetic dialogue generation.

Acknowledgments

This research has been supported in part by Singapore's Ministry of Education (MOE) Academic Research Fund Tier 1, grant number T1 251RES1713.

References

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 579–586. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2018. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335.

Tanya L Chartrand and John A Bargh. 1999. The chameleon effect: the perception–behavior link and social interaction. *Journal of personality and social psychology*, 76(6):893.

Minghai Chen, Sen Wang, Paul Pu Liang, Tadas Baltrušaitis, Amir Zadeh, and Louis-Philippe Morency. 2017. Multimodal sentiment analysis with word-level fusion and reinforcement learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 163–171. ACM.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Dragos Datcu and L Rothkrantz. 2008. Semantic audio-visual data fusion for automatic emotion recognition. *Euromedia'2008*.

Liyanage C De Silva and Pei Chi Ng. 2000. Bimodal emotion recognition. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 332–335. IEEE.

Sidney K D'mello and Jacqueline Kory. 2015. A review and meta-analysis of multimodal affect detection systems. *ACM Computing Surveys (CSUR)*, 47(3):43.

Paul Ekman. 1993. Facial expression and emotion. *American psychologist*, 48(4):384.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1459–1462. ACM.

Reinhard Fiehler. 2002. How to do emotions with words: Emotionality in conversations. *The verbal communication of emotions*, pages 79–106.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018. Conversational memory network for emotion recognition in dyadic dialogue videos. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2122–2132.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL 2014*, volume 1, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP 2014*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Peter Kuppens, Nicholas B Allen, and Lisa B Sheeber. 2010. Emotional inertia and psychological maladjustment. *Psychological Science*, 21(7):984–991.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436.

Gary McKeown, Michel Valstar, Roddy Cowie, Maja Pantic, and Marc Schroder. 2012. The semaine database: Annotated multimodal records of emotionally colored conversations between a person and a limited agent. *IEEE Transactions on Affective Computing*, 3(1):5–17.

Michael W Morris and Dacher Keltner. 2000. How emotions work: The social functions of emotional expression in negotiations. *Research in organizational behavior*, 22:1–50.

Costanza Navarretta, K Choukri, T Declerck, S Goggi, M Grobelnik, and B Maegaard. 2016. Mirroring facial expressions and emotions in dyadic conversations. In *LREC*.

Jérémie Nicolle, Vincent Rapp, Kévin Bailly, Lionel Prevost, and Mohamed Chetouani. 2012. Robust continuous prediction of human emotions using multiscale dynamic cues. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 501–508. ACM.

Rosalind W Picard. 2010. Affective computing: from laughter to ieee. *IEEE Transactions on Affective Computing*, 1(1):11–17.

Soujanya Poria, Erik Cambria, Rajiv Bajpai, and Amir Hussain. 2017a. A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion*, 37:98–125.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017b. Context-dependent sentiment analysis in user-generated videos. In *ACL 2017*, volume 1, pages 873–883.

Jane M Richards, Emily A Butler, and James J Gross. 2003. Emotion regulation in romantic relationships: The cognitive consequences of concealing feelings. *Journal of Social and Personal Relationships*, 20(5):599–620.

Johanna Ruusuvuori. 2013. Emotion, affect and conversation. *The handbook of conversation analysis*, pages 330–349.

Björn Schuller, Michel Valster, Florian Eyben, Roddy Cowie, and Maja Pantic. 2012. Avec 2012: the continuous audio/visual emotion challenge. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 449–456. ACM.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. *arXiv preprint arXiv:1603.06076*.

Mohammad Soleymani, Maja Pantic, and Thierry Pun. 2012. Multimodal emotion recognition in response to videos. *IEEE transactions on affective computing*, 3(2):211–223.

Mingli Song, Jiajun Bu, Chun Chen, and Nan Li. 2004. Audio-visual based emotion recognition-a new approach. In *CVPR 2004*, volume 2, pages II–II. IEEE.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Carlo Strapparava and Rada Mihalcea. 2010. Annotating and identifying emotions in text. In *Intelligent Information Access*, pages 21–38. Springer.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS 2015*, pages 2440–2448.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV 2015*, pages 4489–4497.

Panagiotis Tzirakis, George Trigeorgis, Mihalis A Nicolaou, Björn W Schuller, and Stefanos Zafeiriou. 2017. End-to-end multimodal emotion recognition using deep neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1301–1309.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Peng Xiaolan, Xie Lun, Liu Xin, and Wang Zhiliang. 2013. Emotional state transition model based on stimulus and personality characteristics. *China Communications*, 10(6):146–155.

Liang Yang, Hong-fei LIN, and Wei GUO. 2011. Text-based emotion transformation analysis. *Computer Engineering & Science*, 9:026.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *AAAI*, pages 4970–4977.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1114.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018a. Memory fusion network for multiview sequential learning. In *AAAI*, pages 5634–5641.

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. 2018b. Multi-attention recurrent network for human communication comprehension. In *AAAI*, pages 5642–5649.

Discriminative Learning of Open-Vocabulary Object Retrieval and Localization by Negative Phrase Augmentation

Ryota Hinami[1,2] **and Shin'ichi Satoh**[2,1]
[1]The University of Tokyo, [2]National Institute of Infomatics
`hinami@nii.ac.jp, satoh@nii.ac.jp`

Abstract

Thanks to the success of object detection technology, we can retrieve objects of the specified classes even from huge image collections. However, the current state-of-the-art object detectors (such as Faster R-CNN) can only handle pre-specified classes. In addition, large amounts of positive and negative visual samples are required for training. In this paper, we address the problem of open-vocabulary object retrieval and localization, where the target object is specified by a textual query (e.g., a word or phrase). We first propose Query-Adaptive R-CNN, a simple extension of Faster R-CNN adapted to open-vocabulary queries, by transforming the text embedding vector into an object classifier and localization regressor. Then, for discriminative training, we then propose negative phrase augmentation (NPA) to mine hard negative samples which are visually similar to the query and at the same time semantically mutually exclusive of the query. The proposed method can retrieve and localize objects specified by a textual query from one million images in only 0.5 seconds with high precision.

1 Introduction

Our goal is to retrieve objects from large-scale image database and localize their spatial locations given a textual query. The task of object retrieval and localization has many applications such as spatial position-aware image searches (Hinami et al., 2017) and it recently has gathered much attention from researchers. While much of the previous work mainly focused on object instance retrieval wherein the query is an image (Shen et al., 2012; Tao et al., 2014; Tolias et al., 2016), recent approaches (Aytar and Zisserman, 2014; Hinami and Satoh, 2016) enable retrieval of more generic concepts such as an object category. Although such approaches are built on the recent successes of object detection including that of R-CNN (Girshick et al., 2014), object detection methods can generally handle only closed sets of categories (e.g., PASCAL 20 classes), which severely limits the variety of queries when they are used as retrieval systems. Open-vocabulary object localization is also a hot topic and many approaches are proposed to solve this problem (Plummer et al., 2015; Chen et al., 2017). However, most of them are not scalable to make them useful for large-scale retrieval.

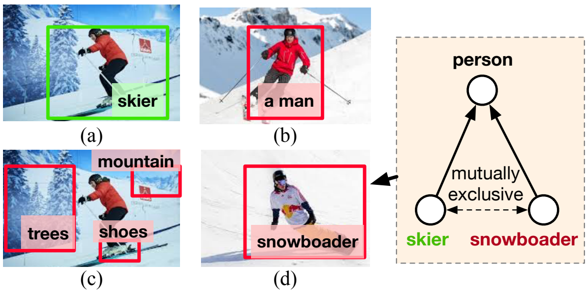

(a) (b) (c) (d)

Figure 1: Training examples in open-vocabulary object detection. (a) positive example of skier classifier. (b) examples without positive annotation, which can be positive. (c) examples without positive annotation from an image that contains a positive example. (d) proposed approach to select hard and true negative examples by using linguistics knowledge.

We first describe *Query-Adaptive* R-CNN as an extension of the Faster R-CNN (Ren et al., 2015) object detection framework to open-vocabulary object detection simply by adding a component called a *detector generator*. While Faster R-CNN learns the class-specific linear classifier as learnable parameters of the neural network, we generate the weight of the classifier adaptively from text descriptions by learning the detector generator (Fig. 2b). All of its components can be trained in an end-to-end manner. In spite of its simple archi-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2605–2615
Brussels, Belgium, October 31 - November 4, 2018.

tecture, it outperforms all state-of-the-art methods in the Flickr30k Entities phrase localization task. It can also be used for large-scale retrievals in the manner presented in (Hinami and Satoh, 2016).

However, training a discriminative classifier is harder in the open-vocabulary setting. Closed-vocabulary object detection models such as Faster R-CNN are trained using many negative examples, where a sufficient amount of good-quality negative examples is shown to be important for learning a discriminative classifier (Felzenszwalb et al., 2010; Shrivastava et al., 2016). While closed-vocabulary object detection can use all regions without positive labels as negative data, in open-vocabulary detection, it is not guaranteed that a region without a positive label is negative. For example, as shown in Fig. 1b, a region with the annotation `a man` is not always negative for `skier`. Since training data for open-vocabulary object detection is generally composed of images, each having region annotations with free descriptions, it is nearly impossible to do an exhaustive annotation throughout the dataset for all possible descriptions. Another possible approach is to use the regions without positive labels in the image that contains positive examples, as shown in Fig. 1c. Although they can be guaranteed to be positive by carefully annotating the datasets, negative examples are only limited to the objects that cooccur with the learned class.

To exploit negative data in open-vocabulary object detection, we use mutually exclusive relationships between categories. For example, an object with a label `dog` is guaranteed to be negative for the `cat` class because `dog` and `cat` are mutually exclusive. In addition, we propose an approach to select *hard negative* phrases that are difficult to discriminate (e.g., selecting `zebra` for `horse`). This approach, called *negative phrase augmentation (NPA)*, significantly improves the discriminative ability of the classifier and improves the retrieval performance by a large margin.

Our contributions are as follows. 1) We propose Query-Adaptive R-CNN, an extension of Faster R-CNN to open vocabulary, that is a simple yet strong method of open-vocabulary object detection and that outperforms all state-of-the-art methods in the phrase localization task. 2) We propose negative phrase augmentation (NPA) to exploit hard negative examples when training for open-vocabulary object detection, which makes the classifier more discriminative and robust to distractors in retrieval. Our method can accurately find objects amidst one million images in 0.5 second.

2 Related work

Phrase localization. Object grounding with natural language descriptions has recently drawn much attention and several tasks and approaches have been proposed for it (Guadarrama et al., 2014; Hu et al., 2016; Kazemzadeh et al., 2014; Mao et al., 2016; Plummer et al., 2015). The most related task to ours is the phrase localization introduced by Plummer et al. (Plummer et al., 2015), whose goal is to localize objects that corresponds to noun phrases in textual descriptions from an image. Chen et al. (Chen et al., 2017) is the closest to our work in terms of learning region proposals and performing regression conditioned upon a query. However, most phrase localization methods are not scalable and cannot be used for retrieval tasks. Some approaches (Plummer et al., 2017b; Wang et al., 2016a) learn a common subspace between the text and image for phrase localization. Instead of learning the subspace between the image and sentence as in standard cross-modal searches, they learn the subspace between a region and a phrase. In particular, Wang et al. (Wang et al., 2016a) use a deep neural network to learn the joint embedding of images and text; their training uses structure-preserving constraints based on structured matching. Although these approaches can be used for large-scale retrieval, their accuracy is not as good as recent state-of-the-art methods.

Object retrieval and localization. Object retrieval and localization have been researched in the context of particular object retrieval (Shen et al., 2012; Tao et al., 2014; Tolias et al., 2016), where a query is given as an image. Aytar et al. (Aytar and Zisserman, 2014) proposed retrieval and localization of generic category objects by extending the object detection technique to large-scale retrieval. Hinami and Satoh (Hinami and Satoh, 2016) extended the R-CNN to large-scale retrieval by using approximate nearest neighbor search techniques. However, they assumed that the detector of the category is given as a query and require many sample images with bounding box annotations in order to learn the detector. Several other approaches have used the external search engines (e.g., Google image search) to get training images from textual queries (Arandjelovi et al., 2012; Chatfield et al., 2015). Instead, we generate an object detector directly from the given textual query by using a neural network.

Parameter prediction by neural network. Query-Adaptive R-CNN generates the weights of the detector from the query instead of learning them by backpropagation. The dynamic filter network (De Brabandere et al., 2016) is one of the first methods that generate neural network parameters dynamically conditioned on an input. Several subsequent approaches use this idea in zero-shot learning (Ba et al., 2016) and visual question answering (Noh et al., 2016). Zhang et al. (Zhang et al., 2017) integrates this idea into the Fast R-CNN framework by dynamically generating the classifier from the text in a similar manner to (Ba et al., 2016). We extend this work to the case of large-scale retrieval. The proposed Query-Adaptive R-CNN generates the regressor weights and learn the region proposal network following Faster R-CNN. It enables precise localization with fewer proposals, which makes the retrieval system more memory efficient. In addition, we propose a novel hard negative mining approach, called negative phrase augmentation, which makes the generated classifier more discriminative.

3 Query-Adaptive R-CNN

Query-adaptive R-CNN is a simple extension of Faster R-CNN to open-vocabulary object detection. While Faster R-CNN detects objects of fixed categories, Query-Adaptive R-CNN detects any objects specified by a textual phrase. Figure 2 illustrates the difference between Faster R-CNN and Query-Adaptive R-CNN. While Faster R-CNN learns a class-specific classifier and regressor as parameters of the neural networks, Query-Adaptive R-CNN generates them from the query text by using a detector generator. Query-Adaptive R-CNN is a simple but effective method that surpasses state-of-the-art phrase localization methods and can be easily extended to the case of large-scale retrieval. Furthermore, its retrieval accuracy is significantly improved by a novel training strategy called negative phrase augmentation (Sec. 3.2).

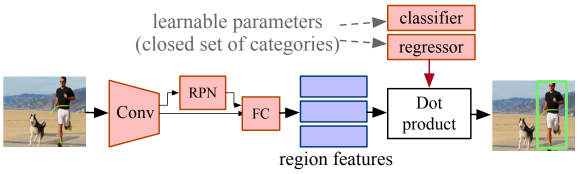

(a) Faster R-CNN

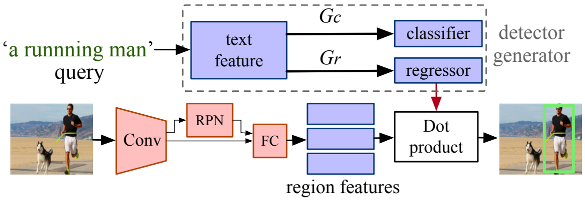

(b) Query-Adaptive R-CNN

Figure 2: Difference in network architecture between (a) Faster R-CNN and (b) Query-Adaptive R-CNN. While Faster R-CNN learns the classifier of a closed set of categories as learnable parameters of neural networks, Query-Adaptive R-CNN generates a classifier and regressor adaptively from a query text by learning a detector generator that transforms the text into a classifier and regressor.

3.1 Architecture

The network is composed of two subnetworks: a *region feature extractor* and *detector generator*, both of which are trained in an end-to-end manner. The region feature extractor takes an image as input and outputs features extracted from sub-regions that are candidate objects. Following Faster R-CNN (Ren et al., 2015), regions are detected using a region proposal network (RPN) and the features of the last layer (e.g., fc7 in VGG network) are used as region features. The detector generator takes a text description as an input and outputs a linear classifier and regressor for the description (e.g., if `a dog` is given, `a dog` classifier and regressor are output). Finally, a confidence and a regressed bounding box are predicted for each region by applying the classifier and regressor to the region features.

Detector generator. The detector generator transforms the given text t into a classifier $\mathbf{w}^c$ and regressor $(\mathbf{w}^r_x, \mathbf{w}^r_y, \mathbf{w}^r_w, \mathbf{w}^r_h)$, where $\mathbf{w}_c$ is the weight of a linear classifier and $(\mathbf{w}^r_x, \mathbf{w}^r_y, \mathbf{w}^r_w, \mathbf{w}^r_h)$ is the weight of a linear regressor in terms of x, y, width w, and height h, following (Girshick et al., 2014). We first transform a text t of variable length into a text embedding vector $\mathbf{v}$. Other phrase localization approaches uses the Fisher vector encoding of word2vec (Klein et al., 2015; Plummer et al., 2015) or long-short term memory

(LSTM) (Chen et al., 2017) for the phrase embedding. However, we found that the simple mean pooling of word2vec (Mikolov et al., 2013) performs better than these methods for our model (comparisons given in the supplemental material). The text embedding is then transformed into a detector, i.e., $\mathbf{w}_c = G_c(\mathbf{v})$ and $(\mathbf{w}^r_x, \mathbf{w}^r_y, \mathbf{w}^r_w, \mathbf{w}^r_h) = G_r(\mathbf{v})$. Here, we use a linear transformation for G_c (i.e., $\mathbf{w}_c = \mathbf{W}\mathbf{v}$, where $\mathbf{W}$ is a projection matrix). For the regressor, we use a multi-layer perceptron with one hidden layer to predict each of $(\mathbf{w}^r_x, \mathbf{w}^r_y, \mathbf{w}^r_w, \mathbf{w}^r_h) = G_r(\mathbf{v})$. We tested various architectures for G_r and found that sharing the hidden layer and reducing the dimension of the hidden layer (up to 16) does not adversely affect the performance, while at the same time it significantly reduces the number of parameters (see Sec. 5.2 for details).

3.2 Training with Negative Phrase Augmentation

All components of Query-Adaptive R-CNN can be jointly trained in an end-to-end manner. The training strategy basically follows that of Faster R-CNN. The differences are shown in Figure 3. Faster R-CNN is trained with the fixed closed set of categories (Fig. 3a), where all regions without a positive label can be used as negative examples. On the other hand, Query-Adaptive R-CNN is trained using the open-vocabulary phrases annotated to the regions (Fig. 3b), where sufficient negative examples cannot be used for each phrase compared to Faster R-CNN because a region without a positive label is not guaranteed to be negative in open-vocabulary object detection. We solve this problem by proposing negative phrase augmentation (NPA), which enables us to use good quality negative examples by using the linguistic relationship (e.g., mutually exclusiveness) and the confusion between the categories (Fig. 3c). It significantly improves the discriminative ability of the generated classifiers.

3.2.1 Basic Training

First, we describe the basic training strategy without NPA (Fig. 3b). Training a Query-Adaptive R-CNN requires the phrases and their corresponding bounding boxes to be annotated. For the ith image (we use one image as a minibatch), let us assume that C_i phrases are associated with the image. The C_i phrases can be considered as the classes to train in the minibatch. The labels $\mathbf{L}_i \in \{0,1\}^{C_i \times n_r}$ are assigned to the region proposals generated by RPN (each of the dotted rectangles in Fig 3b); a positive label is assigned if the box overlaps the ground truth box by more than 0.5 in IoU and negative labels are assigned to other RoIs under the assumption that all positive objects of C_i classes are annotated (i.e., regions without annotations are negative within the image).[1] We then compute the classification loss by using the training labels and classification scores.[2] The loss in terms of RPN and bounding box regression is computed in the

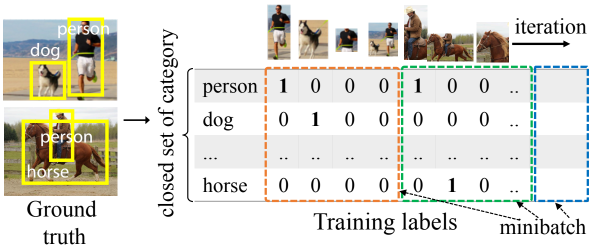

(a) Faster R-CNN (closed-vocabulary)

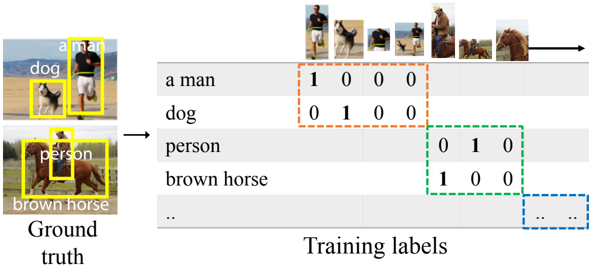

(b) Query-Adaptive R-CNN (open-vocabulary)

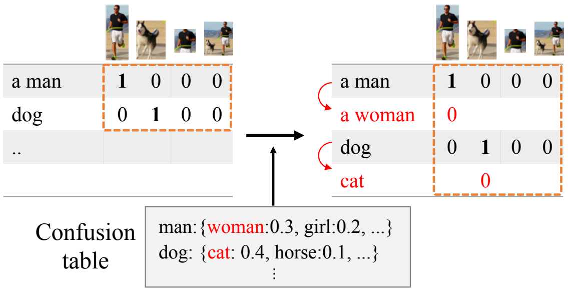

(c) Negative phrase augmentation

Figure 3: Difference in training between (a) closed-vocabulary and (b) open-vocabulary object detection. The approach of NPA is illustrated in (c).

---

[1] Although this assumption is not always true for datasets such as Flickr30k Entities, it nonetheless works well for them because exceptions are rare.

[2] Whereas Faster R-CNN uses the softmax cross entropy over the $C + 1$ (background) classes, where C is the number of closed sets of a category, we use the sigmoid cross entropy because the C_i classes are not always mutually exclusive and a background class cannot be defined in the context of open-vocabulary object detection.

same way as Faster R-CNN (Ren et al., 2015).

3.2.2 Negative Phrase Augmentation

Here, we address the difficulty of using negative examples in the training of open-vocabulary object detection. As shown in Fig. 1b, our generated classifier is not discriminative enough. The reason is the scarcity of negative examples when using the training strategy described in Sec. 3.2.1; e.g., the `horse` classifier is not learned with the `zebra` as a negative example except for the rare case that both a `zebra` and a `horse` are in the same image. Using hard negative examples has proven to be effective in the object detection to train a discriminative detector (Felzenszwalb et al., 2010; Girshick et al., 2014; Shrivastava et al., 2016). However, adding negative examples is usually not easy in the open-vocabulary setting, because it is not guaranteed that a region without a positive label is negative. For example, an object with the label `man` is not a negative of `person` even though `person` is not annotated. There are an infinite number of categories in open-vocabulary settings, which makes it difficult to exhaustively annotate all categories throughout the dataset.

How can we exploit hard examples that are guaranteed to be negative? We can make use of the mutually exclusive relationship between categories: e.g., an object with a `dog` label is negative for `cat` because `dog` and `cat` are mutually exclusive. There are two ways we can add to a minibatch: add negative images (regions) or negative phrases. Adding negative phrases (as in Fig. 3c) is generally better because it involves a much smaller additional training cost than adding images in terms of the both computational cost and GPU memory usage. In addition, to improve the discriminative ability of the classifier, we select only hard negative phrases by mining the confusing categories. This approach, called *negative phrase augmentation (NPA)*, is a generic way of exploiting hard negative examples in open-vocabulary object detection and leads to large improvements in accuracy, as we show in Sec. 5.3.

Confusion table. We create a confusion table that associates a category with its hard negative categories, from which negative phrases are picked as illustrated in Fig. 3c. To create the entry for category c, we first generate the candidate list of hard negative categories by retrieving the top 500 scored objects from all objects in the validation set of Visual Genome (Krishna et al., 2016) (using c as a query). After that, we remove the mutually non-exclusive category relative to c from the list. Finally, we aggregate the list by category and assign a weight to each category. Each of the registered entries becomes like `dog:{cat:0.5, horse:0.3, cow:0.2}`. The weight corresponds to the probability of selecting the category in NPA, which is computed based on the number of appearances and their ranks in the candidate list.[3]

Removal of mutually non-exclusive phrases. To remove non-mutually exclusive phrases from the confusion table, we use two approaches that estimate whether the two categories are mutually exclusive or not. 1) The first approach uses the *WordNet hierarchy*: if two categories have parent-child relationships in WordNet (Miller, 1995), they are not mutually exclusive. However, the converse is not necessarily true; e.g., `man` and `skier` are not mutually exclusive but do not have the parent-child relationship in the WordNet hierarchy. 2) As an alternative approach, we propose to use *Visual Genome annotation*: if two categories co-occur more often in the Visual Genome dataset (Krishna et al., 2016), these categories are considered to be not mutually exclusive.[4] These two approaches are complementary, and they improve detection performance by removing the mutually non-exclusive words (see Sec. 5.3).

The training pipeline with NPA is as follows:

(1) **Update the confusion table:** The confusion table is updated periodically (after every 10k iterations in our study). Entries were created for categories that frequently appeared in 10k successive batches (or the whole training set if the size of the dataset is not large).

(2) **Add hard negative phrases:** Negative phrases are added to each of the C_i phrases in a minibatch. We replace the name of the category in each phrase with its hard negative category (e.g., generate `a running woman` for `a running man`), where the category name is obtained by extracting nouns. A negative phrase is randomly selected from the confusion table on the basis of the assigned probability.

---

[3] We compute the weight of each category as the sum of 500 minus the rank for all ranked results in the candidate lists normalized over all categories in order to sum to one.

[4] We set the ratio at 1% of objects in either category. For example, if there are 1000 objects with the `skier` label and 20 of those objects are also annotated with `man` (20/1000=2%), we consider that `skier` and `man` are not mutually exclusive.

(3) **Add losses:** As illustrated in Fig. 3c, we only add negative labels to the regions where a positive label is assigned to the original phrase. The classification loss is computed only for the regions, which is added to the original loss.

4 Large-Scale Object Retrieval

Query-Adaptive R-CNN can be used for large-scale object retrieval and localization, because it can be decomposed into a query-independent part and a query-dependent part, i.e., a region feature extractor and detector generator. We follow the approach used in large-scale R-CNN (Hinami and Satoh, 2016), but we overcome its two critical drawbacks. First, a large-scale R-CNN can only predict boxes included in the region proposals; these are detected offline even though the query is unknown at the time; therefore, to get high recall, a large number of object proposals should be used, which is memory inefficient. Instead, we generate a regressor as well as a classifier, which enables more accurate localization with fewer proposals. Second, a large-scale R-CNN assumes that the classifier is given as a query, and learning a classifier requires many samples with bounding annotations. We generate the classifier from a text query directly by using the detector generator of Query-Adaptive R-CNN. The resulting system is able to retrieve and localize objects from a database with *one million images* in *less than one second.*

Database indexing. For each image in the database, the region feature extractor extracts region proposals and corresponding features. We create an index for the region features in order to speed up the search. For this, we use the IVFADC system (Jégou et al., 2011) in the manner described in (Hinami and Satoh, 2016).

Searching. Given a text query, the detector generator generates a linear classifier and bounding box regressor. The regions with high classification scores are then retrieved from the database by making an IVFADC-based search. Finally, the regressor is applied to the retrieved regions to obtain the accurately localized bounding boxes.

5 Experiments

5.1 Experimental Setup

Model: Query-Adaptive R-CNN is based on VGG16 (Simonyan and Zisserman, 2015), as in other work on phrase localization. We first initialized the weights of the VGG and RPN by using Faster R-CNN trained on Microsoft COCO (Lin et al., 2014); the weights were then fine-tuned for each dataset of the evaluation. In the training using Flickr30k Entities, we first pretrained the model on the Visual Genome dataset using the object name annotations. We used Adam (Kingma and Ba, 2015) with a learning rate starting from 1e-5 and ran it for 200k iterations.

Tasks and datasets: We evaluated our approaches on two tasks: phrase localization and open-vocabulary object detection and retrieval. The **phrase localization task** was performed on the Flickr30k Entities dataset (Plummer et al., 2015). Given an image and a sentence that describes the image, the task was to localize region that corresponds to the phrase in a sentence. Flickr30k datasets contain 44,518 unique phrases, where the number of words of each phrase is 1–8 (2.1 words on average). We followed the evaluation protocol of (Plummer et al., 2015). We did not use Flickr30k Entities for the retrieval task because the dataset is not exhaustively annotated (e.g., not all men appearing in the dataset are annotated with `man`), which makes it difficult to evaluate with a retrieval metric such as AP, as discussed in Plummer et al. (Plummer et al., 2017b). Although we cannot evaluate the retrieval performance directly on the phrase localization task, we can make comparisons with other approaches and show that our method can handle a wide variety of phrases.

The **open-vocabulary object detection and retrieval task** was evaluated in the same way as the standard object detection task. The difference was the assumption that we do not know the target category at training time in open-vocabulary settings; i.e., the method does not tune in to a specific category, unlike the standard object detection task. We used the Visual Genome dataset (Krishna et al., 2016) and selected the 100 most frequently object categories as queries among its 100k or so categories.[5] [6] We split the dataset into training, validation, and test sets following (Johnson et al., 2016). We also evaluated our approaches on the PASCAL VOC 2007 dataset, which is a widely used dataset

---

[5] Since the WordNet synset ID is assigned to each object, we add objects with labels of hyponyms as positives (e.g., `man` is positive for the `person` category).

[6] We exclude the background (e.g., `grass`, `sky`, `field`), multiple objects (e.g., `people`, `leaves`), and ambiguous categories (e.g, `top`, `line`).

Approach	People	Clothing	Body	Animals	Vehicles	Instruments	Scene	Other	All
Non-scalable methods									
GroundeR (Rohrbach et al., 2016)	61.00	38.12	10.33	62.55	68.75	36.42	58.18	29.08	47.81
Multimodal compact bilinear (Fukui et al., 2016)	-	-	-	-	-	-	-	-	48.69
PGN+QRN (Chen et al., 2017)	75.08	55.90	20.27	73.36	68.95	45.68	65.27	38.80	60.21
Non-scalable and joint localization methods									
Structured matching (Wang et al., 2016b)	57.89	34.61	15.87	55.98	52.25	23.46	34.22	26.23	42.08
SPC+PPC (Plummer et al., 2017a)	71.69	50.95	25.24	76.25	66.50	35.80	51.51	35.98	55.85
QRC net (Chen et al., 2017)	76.32	59.58	25.24	**80.50**	**78.25**	50.62	67.12	43.60	65.14
Scalable methods									
Structure-preserving embedding (Wang et al., 2016a)	-	-	-	-	-	-	-	-	43.89
CCA+Detector+Size+Color (Plummer et al., 2017b)	64.73	46.88	17.21	65.83	68.75	37.65	51.39	31.77	50.89
Query-Adaptive R-CNN (proposed)	**78.17**	**61.99**	**35.25**	74.41	76.16	**56.69**	**68.07**	**47.42**	**65.21**

Table 1: **Phrase localization** accuracy on Flickr30k Entities dataset.

		IoU				
Architecture	Params	0.5	0.6	0.7	0.8	0.9
w/o regression	-	**65.21**	53.19	35.70	14.32	1.88
300–16(–4096)	**0.3M**	64.14	57.66	48.22	33.04	9.29
300–64(–4096)	1.1M	63.87	57.43	**49.05**	33.84	**10.55**
300–256(–4096)	4.3M	63.84	57.70	48.71	33.87	10.05
300–1024(–4096)	17M	64.29	**58.05**	48.49	**33.94**	10.09
300(–256–4096)	4.5M	62.82	56.28	48.02	32.71	9.89
300–4096	1.2M	63.23	56.92	48.17	32.66	9.20

Table 2: Comparison of various **bounding box regressors** on Flickr30k Entities for different IoU thresholds. The number of parameters in G_r is also shown.

for object detection.[7] As metrics, we used top-k precision and average precision (AP), computed from the region-level ranked list as in the standard object detection task.[8]

5.2 Phrase localization

Comparison with state-of-the-art. We compared our method with state-of-the-art methods on the Flickr30k Entities phrase localization task. We categorized the methods into two types, i.e., non-scalable and scalable methods (Tab. 1). 1) *Non-scalable methods* cannot be used for large-scale retrieval because their query-dependent components are too complex to process a large amount of images online, and 2) *Scalable methods* can be used for large-scale retrieval because their query-dependent components are easy to scale up (e.g., the L_2 distance computation); these include common subspace-based approaches such as CCA. Our method also belongs to the scalable category. We used a simple model without a regressor and NPA in the experiments.

Table 1 compares Query-Adaptive R-CNN with the state-of-the-art methods. Our model achieved *65.21%* in accuracy and outperformed all of the previous state-of-the-art models including the non-scalable or joint localization methods. Moreover, it significantly outperformed the scalable methods, which suggests the approach of predicting the classifier is better than learning a common subspace for the open-vocabulary detection problem.

Bounding box regressor. To demonstrate the effectiveness of the bounding box regressor for precise localization, we conducted evaluations with the regressor at different IoU thresholds. As explained in Sec. 3.1, the regressor was generated using G_r, which transformed 300-d text embeddings x into 4096-d regressor weights $\mathbf{w}_x^r$, $\mathbf{w}_y^r$, $\mathbf{w}_w^r$, and $\mathbf{w}_h^r$. We compared three network architectures for G_r: 1) `300-n(-4096)` MLP having a hidden layer with n units that is shared across the four outputs, 2) `300(-n-4096)` MLP having a hidden layer that is not shared, and 3) `300(-4096)` linear transformation (without a hidden layer).

Table 2 shows the results with and without regressor. The regressor significantly improved the accuracy with high IoU thresholds, which demonstrates that the regressor improved the localization accuracy. In addition, the accuracy did not decrease as a result of sharing the hidden layer or reducing the number of units in the hidden layer. This suggests that the regressor lies in a very low-dimensional manifold because the regressor for one concept can be shared by many concepts (e.g., the `person` regressor can be used for `man`, `woman`, `girl`, `boy`, etc.). The number of parameters was significantly reduced by these tricks,

---

[7] We used the model trained on Visual Genome even for the evaluation on the PASCAL dataset because of the assumption that the target category is unknown.

[8] We did not separately evaluate the detection and retrieval tasks because both can be evaluated with the same metric.

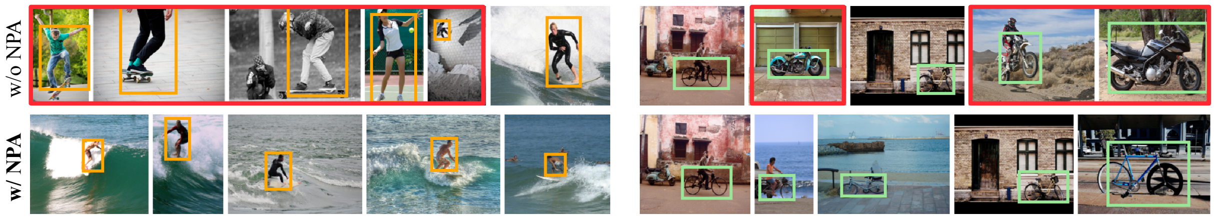

Query: surfer

Query: bicycle

Figure 4: Qualitative results with and without NPA. Top-k retrieved results for two queries are shown (sorted by rank) and false alarms are depicted with a red border.

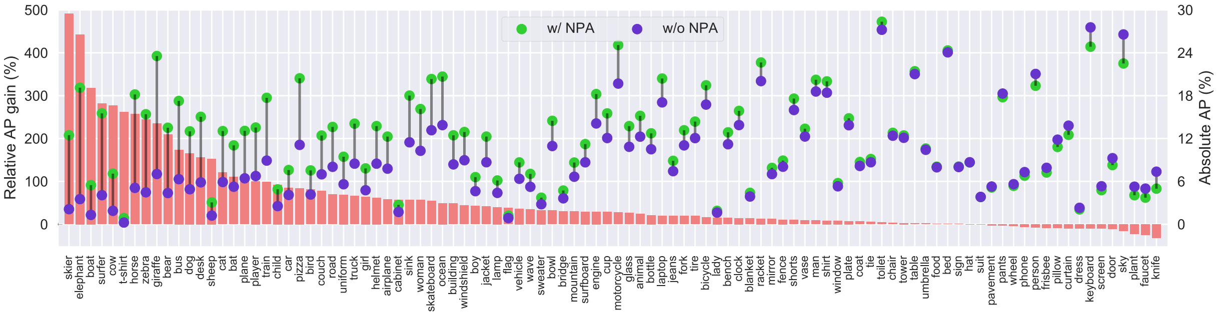

Figure 5: AP gain by **negative phrase augmentation** (NPA) for individual queries. The bars show the relative AP gain and points shows the absolute AP with and without NPA.

				Visual Genome			VOC
	NPA	WN	VG	mAP	PR@10	PR@100	mAP
CCA				3.18	20.40	15.64	28.23
Query-Adaptive R-CNN				9.15	52.60	36.85	29.14
	✓			10.90	60.10	43.21	36.74
	✓	✓		11.53	61.80	45.91	37.07
	✓		✓	11.65	65.40	46.85	41.32
	✓	✓	✓	**12.19**	**65.70**	**48.45**	**42.81**

Table 3: **Open-vocabulary object detection** performance on Visual Genome and PASCAL VOC 2007 datasets. WN and VG are the strategies to remove mutually non-exclusive phrases.

Query	Most confusing class				2nd most confusing class			
girl	man	19	→	3	boy	4	→	2
skateboard	surfboard	12	→	0	snowboard	11	→	0
train	bus	17	→	1	oven	3	→	0
helmet	hat	18	→	1	cap	6	→	4
elephant	bear	14	→	0	horse	6	→	0

Table 4: Number of false alarms in top 100 results for five queries (**w/o NPA** → **w/ NPA**). The top 2 confusing categories are shown for each query.

to even fewer than in the linear transformation. The accuracy slightly decreased with a threshold of 0.5, because the regressor was not learned properly for the categories that did not frequently appear in the training data.

5.3 Open-Vocabulary Object Retrieval

Main comparison. Open-vocabulary object detection and retrieval is a much more difficult task than phrase localization, because we do not know how many objects are present in an image. We used NPA to train our model. As explained in Sec. 3.2.2, we used two strategies, *Visual Genome annotation (VG)* and *WordNet hierarchy (WN)*, to remove mutually non-exclusive phrases from the confusion table. As a baseline, we compared with region-based CCA (Plummer et al., 2017b), which is scalable and shown to be effective for phrase localization; for a fair comparison, the subspace was learned using the same dataset as ours. An approximate search was not used to evaluate the actual performance at open-vocabulary object detection.

Table 3 compares different training strategies. NPA significantly improved the performance: *more than 25% relative improvement* for all metrics. Removing mutually non-exclusive words also contributed the performance: WN and VG both improved performance (5.8% and 6.9% relative AP gain, respectively). Performance improved even further by combining them (11.8% relative AP gain), which shows they are complementary. AP was much improved by NPA for the PASCAL dataset as well (47% relative gain). However, the performance was still much poorer than those of the state-of-the-art object detection

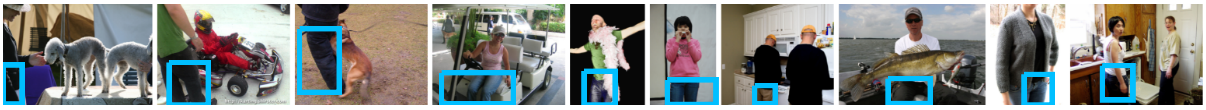

Query: blue jeans

Query: shark

Query: black dog

Figure 6: Retrievals from one million images. Top-k results for three queries are shown.

methods (Redmon and Farhadi, 2017; Ren et al., 2015), which suggests that there is a large gap between open-vocabulary and closed-vocabulary object detection.

Detailed results of NPA. To investigate the effect of NPA, we show the AP with and without NPA for individual categories in Figure 5, which are sorted by relative AP improvement. It shows that AP improved especially for animals (`elephant`, `cow`, `horse`, etc.) and person (`skier`, `surfer`, `girl`), which are visually similar within the same upper category. Table 4 shows the most confused category and its total count in the top 100 search results for each query, which shows what concept is confusing for each query and how much the confusion is reduced by NPA.[9] This shows that visually similar categories resulted in false positive without NPA, while their number was suppressed by training with NPA. The reason is that these confusing categories were added for negative phrases in NPA, and the network learned to reject them. Figure 4 shows the qualitative search results for each query with and without NPA (and CCA as a baseline), which also showed that NPA can discriminate confusing categories (e.g., `horse` and `zebra`). These results clearly demonstrate that NPA significantly improves the discriminative ability of classifiers by adding hard negative categories.

Large-scale experiments. Finally, we evaluated the scalability of our method on a large image database. We used one million images from the ILSVRC 2012 training set for this evaluation. Table 5 show the speed and memory. The mean and standard deviation of speed are computed over 20 queries in PASCAL VOC dataset. Our system could retrieve objects from one million images in around 0.5 seconds. We did not evaluate accuracy because there is no such large dataset with bounding box annotations.[10] Figure 6 shows the retrieval results from one million images, which demonstrates that our system can accurately retrieve and localize objects from a very large-scale database.

Database size	10K	50K	100K	500K	1M
Time (ms)	183±16	196±21	242±28	314±90	484±165
Memory (GB)	0.46	1.23	2.19	9.87	19.47

Table 5: Speed/memory in **large-scale** experiments.

6 Conclusion

Query-Adaptive R-CNN is a simple yet strong framework for open-vocabulary object detection and retrieval. It achieves state-of-the-art performance on the Flickr30k phrase localization benchmark and it can be used for large-scale object retrieval by textual query. In addition, its retrieval accuracy can be further increased by using a novel training strategy called negative phrase augmentation (NPA) that appropriately selects hard negative examples by using their linguistic relationship and confusion between categories. This simple and generic approach significantly improves the discriminative ability of the generated classifier.

Acknowledgements: This work was supported by JST CREST JPMJCR1686 and JSPS KAKENHI 17J08378.

---

[9] For each query, we scored all the objects in the Visual Genome testing set and counted the false alarms in the top 100 scored objects.

[10] adding distractors would also be difficult, because we cannot guarantee that relevant objects are not in the images.

References

Relja Arandjelovi, Andrew Zisserman, Relja Arandjelovic, Andrew Zisserman, Relja Arandjelovi, Andrew Zisserman, Relja Arandjelovic, and Andrew Zisserman. 2012. Multiple queries for large scale specific object retrieval. In *BMVC*.

Yusuf Aytar and Andrew Zisserman. 2014. Immediate, scalable object category detection. In *CVPR*.

Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. 2016. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *ICCV*.

Ken Chatfield, Relja Arandjelovi, Andrew Zisserman, Relja Arandjelović, Omkar Parkhi, and Andrew Zisserman. 2015. On-the-fly learning for visual search of large-scale image and video datasets. *International Journal of Multimedia Information Retrieval*, 4(2):75–93.

Kan Chen, Rama Kovvuri, and Ram Nevatia. 2017. Query-guided Regression Network with Context Policy for Phrase Grounding. In *ICCV*.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. 2016. Dynamic filter networks. In *NIPS*.

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, Deva Ramanan, and David Forsyth. 2010. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–45.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*.

Ross Girshick, Jeff Donahue, Trevor Darrell, U C Berkeley, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Sergio Guadarrama, Erik Rodner, Kate Saenko, Ning Zhang, Ryan Farrell, Jeff Donahue, and Trevor Darrell. 2014. Open-vocabulary object retrieval. *Robotics: Science and Systems*, 2(5):1–9.

Ryota Hinami, Yusuke Matsui, and Shin'ichi Satoh. 2017. Region-based image retrieval revisited. In *ACMMM*.

Ryota Hinami and Shin'ichi Satoh. 2016. Large-scale r-cnn with classifier adaptive quantization. In *ECCV*.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016. Natural language object retrieval. In *CVPR*.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *PAMI*, 33(1):117–128.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. Densecap: fully convolutional localization networks for dense captioning. In *CVPR*.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. Referitgame: referring to objects in photographs of natural scenes. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization. In *ICLR*.

Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. 2015. Fisher vectors derived from hybrid gaussian-laplacian mixture models for image annotation. In *CVPR*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalanditis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, Li Fei-Fei, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. 2016. Visual genome: connecting language and vision using crowdsourced dense image annotations. *IJCV*, page 44.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. 2014. Microsoft coco: common objects in context. In *ECCV*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *CVPR*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A. Miller. 1995. Wordnet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. 2016. Image question answering using convolutional neural network with dynamic parameter prediction. In *CVPR*.

Bryan A Plummer, Christopher M Cervantes, and C V Aug. 2017a. Phrase localization and visual relationship Detection with Comprehensive Image-Language Cues. In *ICCV*.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2017b. Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models. *International Journal of Computer Vision*, 123(1):74–93.

Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *CVPR*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: towards real-time object detection with region proposal networks. In *NIPS*.

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *ECCV*.

Xiaohui Shen, Zhe Lin, Jonathan Brandt, Shai Avidan, and Ying Wu. 2012. Object retrieval and localization with spatially-constrained similarity measure and k-nn re-ranking. In *CVPR*.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. 2016. Training region-based object detectors with online hard example mining. In *CVPR*.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Ran Tao, Efstratios Gavves, Cees G M Snoek, and Arnold W M Smeulders. 2014. Locality in generic instance search from one example. In *CVPR*.

Giorgos Tolias, Ronan Sicre, and Hervé Jégou. 2016. Particular object retrieval with integral max-pooling of cnn activations. In *ICLR*.

Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016a. Learning deep structure-preserving image-text embeddings. In *CVPR*.

Mingzhe Wang, Mahmoud Azab, Noriyuki Kojima, Rada Mihalcea, and Jia Deng. 2016b. Structured matching for phrase localization. In *ECCV*.

Yuting Zhang, Luyao Yuan, Yijie Guo, Zhiyuan He, I-An Huang, and Honglak Lee. 2017. Discriminative Bimodal Networks for Visual Localization and Detection with Natural Language Queries. *CVPR*.

Grounding Semantic Roles in Images

Carina Silberer[†♣]
carina.silberer@upf.edu

Manfred Pinkal[†]
pinkal@coli.uni-saarland.de

[†]Department of Computational Linguistics
Saarland University, Saarbrücken, Germany
[♣]Universitat Pompeu Fabra
Barcelona, Spain

Abstract

We address the task of visual semantic role labeling (vSRL), the identification of the participants of a situation or event in a visual scene, and their labeling with their semantic relations to the event or situation. We render candidate participants as image regions of objects, and train a model which learns to ground roles in the regions which depict the corresponding participant. Experimental results demonstrate that we can train a vSRL model without reliance on prohibitive image-based role annotations, by utilizing noisy data which we extract automatically from image captions using a linguistic SRL system. Furthermore, our model induces frame–semantic visual representations, and their comparison to previous work on supervised visual verb sense disambiguation yields overall better results.

1 Introduction

Images of everyday scenes can be interpreted and described in many ways, depending on the perceiver and the context in which the image is presented. The latter may be natural language data or a visual sequence. As an example, consider the two scenes in Figure 1 and the question *What is the man doing?* The interpretation of the first target image (left) in isolation would allow many answers. Taking into account the visual context, however, may disprove many of those answers (e.g., *He is questioning the women.*). For the target image on the right, the reason for *Why there is so much food on the table?* can be inferred from its textual context.

As the examples illustrate, the interpretation of a (visual) scene is related to the determination of its events, their participants and the roles they play therein (i.e., distill *who did what to whom, where, why and how*), and this may require a joint processing or reasoning with possibly multiple (extra-)linguistic information sources (e.g., text, images). In NLP, the well-established and studied task of semantic role labeling (SRL) aims to extract such knowledge in the form of shallow semantic structures from natural language texts (e.g., *questioning(Agent:man, Theme:women)*); see, e.g., Gildea and Jurafsky (2002); Palmer et al. (2010), for an overview). It is considered an essential task towards text understanding, and was shown to be beneficial for applications such as information extraction (see Roth and Lapata (2016) and the references therein) and question answering (Shen and Lapata, 2007). In computer vision research, recent efforts have been made on *visual* SRL or *situation recognition*, a task coined by transferring the use of semantic roles to produce similar structured meaning descriptions for visual scenes (e.g., Yang et al. (2016); Yatskar et al. (2016)). To facilitate the endeavor of joint processing over multiple sources, it is desirable to induce representations of texts and visual scenes which do encode this kind of information, and in, essentially, a congruent and generic way. The latter would furthermore support the induction of a desired level of abstraction as needed.

In this paper we propose an approach towards this goal: We address the task of visual SRL (vSRL) and learn frame–semantic representations of images. Specifically, we present a model that learns to ground the semantic roles of a *semantic frame* in image *regions*, which may be crucial for, e.g., human-robot interaction or surveillance (e.g., *Who/Where* is the robber?). For example, the image shown in Figure 2 evokes the ARREST frame, and its semantic roles `Authorities`, `Suspect`, and `Place` are grounded in the image regions (delineated by bounding boxes) which depict their corresponding fillers. While being trained on this task, our model learns distributed *situation representations* (for images and frames), and *participant representations* (for image regions and roles) which

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2616–2626
Brussels, Belgium, October 31 - November 4, 2018.

Well, the fridge broke, so I had to eat everything.

TARGET IMAGE CONTEXT TARGET IMAGE CONTEXT

Figure 1: Example images along with their visual (left) or textual (right) contexts.

capture the visual–frame-semantic features of situations and participants, respectively.

We train our model on data that we automatically extract by running a linguistic SRL system on image captions—human produced data that is abundant and requires less time and expertise than frame-semantic annotations. Supervised SRL has suffered from data sparsity since it relies on labor-intensive human annotations. Analogous issues on manually annotated images have been addressed by Yatskar et al. (2017). By leveraging existing efforts made in NLP, we explore whether we can alleviate the supervision bottleneck in visual SRL

Our experiments yield promising results, and our models are even able to make correct predictions for erroneous data points. Furthermore, we evaluate the induced situation representations on the task of supervised visual verb sense disambiguation, where it outperforms or is comparable to previous work (on motion or non-motion verbs, respectively).

2 Related Work

Yatskar et al. (2016) introduced the *ImSitu* dataset for the task of *situation recognition*, i.e., the problem of, given an image, predicting a structured output which specifies the depicted activity (e.g., *jumping*) and its associated semantic roles paired with their nominal fillers (e.g., {*(agent, bear), (obstacle, water)*}. To address the task, Yatskar et al. (2016, 2017) train conditional random field (CRF) models on ImSitu (Yatskar et al., 2016) and on additional training data for rarely occurring noun-role combinations which they source from the web (Yatskar et al., 2017). Mallya and Lazebnik (2017) assume that the roles associated with each activity are in a fixed order, and treat the above task as one of recognizing activities and generating a sequence of nouns, for which they use a recurrent neural network. They show how hereby learned features can be transferred to tackle image caption generation. Li et al. (2017) explicitly model role dependencies through a gated graph neural network. Given an image, they instantiate a fully connected graph with a verb and its roles as nodes. Each node's hidden state vector is initialized with image features from two CNNs, which were pre-trained for the prediction of verbs and nouns, respectively. Using a softmax layer augmented with hidden state vectors, they predict the verb and the nominal fillers of its roles.

In contrast to above works on ImSitu, we do not link the roles of a verb to their lexical fillers. We address the related task of explicitly grounding roles in the corresponding image regions, since our focus is on the relation between semantic roles and the typical visual features of their fillers (e.g., a `Body_part` is typically not a *bike* but *arms*). Gupta and Malik (2015) introduced this task as *visual semantic role labeling*. Similarly, Yang et al. (2016) formulate a CRF that jointly processes a cooking video and its natural language descriptions in order to ground the semantic roles associated with the verbs in corresponding object tracks. Both of these studies are limited to a small number of activities performed by people and a few semantic roles (26 and 11 verbs, 3 and 6 roles, respectively).

Unlike related work, our approach does not rely on manual role annotations of images, but exploits a linguistic SRL system for data creation. With more than 1k frame-specific roles, our data is of a larger scope than Gupta and Malik (2015) and Yang et al. (2016). Further, unlike the CRF-based approaches, our model induces frame-semantic representations during training.

ARREST		PLACING	
r_1, r_2	`Authorities`	r_1	`Agent`
r_5	`Suspect`	r_5	`Theme`
r_3	`Place`	r_3	`Place`
		r_4	`Goal`

target outputs

Figure 2: Example image (from *Flickr30k Entities*) augmented with frame-semantic annotations. Top: Image with objects rendered by bounding boxes. Bottom: Annotations which show the frames which the image evokes, and their roles, linked to their filler objects.

3 Grounding Semantic Roles in Images

We first define the task of vSRL and then present our model and our approach for data creation.

3.1 Task Definition: vSRL

Our approach is based on the linguistic theory of frame-semantics (Fillmore, 1982), which underlies the idea that words evoke semantic frames. Frames describe prototypical situations or events and contain semantic roles. For example, in the sentence *They arrested him for assault*, the argument *they* fills the `Authorities` role, *him* is the `Suspect`, and *assault* the `Charges` of the ARREST frame, which was evoked by the verb *arrest*.

Let F be a set of frames, E be the set of all semantic role labels, and E_f be the inventory of roles associated with the frame f (e.g., E_{ARREST} =\{`Authorities`, `Suspect`, `Charges`, `Offense`, `Place`\})[1]. Assume we are given an image i, which evokes a frame f, and a set of image regions R_i, which render one or several objects in i. The task of vSRL is to link each role $e \in E_f$ to the object $r \in R_i$ that fills role e in the situation or event which f describes. We call a role e to be *realized* in an image, if it can be grounded in an image (region). The object r shown in the image region is called the *filler* or *realization* of e. The structure $A_f = \{(r, e) | r \in R_i, e \in E_f\}$ overall represents the frame f in the image i.

In SRL, the task of identifying the frame which a *predicate* evokes is a prerequisite, but it is usually treated as a subtask of SRL. We follow this approach and consider the identification of the frames evoked by an image as a subtask of vSRL. We formulate two further subtasks for vSRL, namely *role prediction*—determining the correct role for a relevant image region, and *role grounding*—linking a realized role to its filler.

Note that not all roles of a frame may be realized in an image, and not all objects may play a role in an evoked frame. Figure 2, for instance, shows an image with some of its objects delineated by six bounding boxes $R_i = \{r_1, r_2, r_3, r_4, r_5, r_6\}$. The target outputs (bottom, Fig. 2) are the frames ARREST and PLACING, as well as their realized roles which are aligned with their fillers (marked by colors). The FrameNet roles `Charges` and `Offense` are not realized in the image, i.e., they cannot be grounded. The vehicle, box r_4, in turn, does not participate in the ARREST frame.

3.2 Model: Visual-Frame–Semantic Embedder

Our model, illustrated in Figure 3, is formulated as a neural network architecture. Its input is a tuple $q = (i, r, f, e) \in Q$ of an image i, an object which is delineated by bounding box r, a frame $f \in F$, and a role label $e \in E_f$ (e.g., q = (img_1, r_5, ARREST, `Suspect`); cf. Fig. 2). The model output is a score $s(q) \in [-1, 1]$ which quantifies the visual–frame-semantic correspondence between the box r and the role e of f (Fig. 3, right).

More specifically, the model maps visual encodings of i and r (e.g., vectors of a pre-trained CNN), and frame-semantic representations of f and e (randomly initialized embeddings) to common visual–frame-semantic spaces (*cross-modal layers* in Fig. 3).

We assume that images capture different frame-semantic features than image regions—an image encodes the whole scene and its participants and thus evokes a frame, while individual image regions of participants capture the participant-specific features of the semantic roles they fill. We therefore distinguish between two different cross-modal spaces: a situation space for images and frames, and a participant space for regions and roles. Using the respective representations in these spaces, the model then estimates the situation similar-

---

[1] We use FrameNet 1.5 (Ruppenhofer et al., 2006).

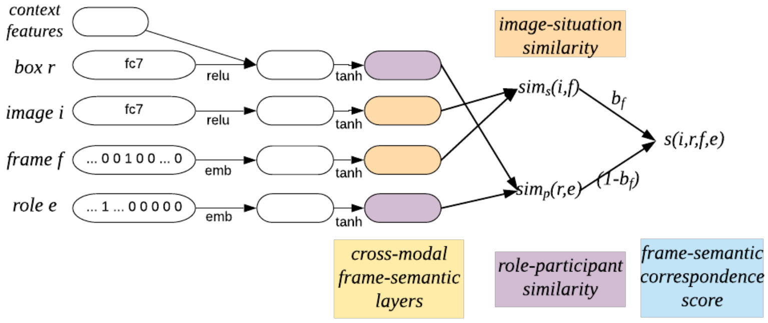

Figure 3: The **ImgObjLoc** model which scores the correspondence between a semantic role and its frame, respectively, and a candidate role filler (an image region) and the whole image, respectively.

ity, $\mathrm{sim}_s(i, f)$, between the image and the frame, and the participant-role similarity, $\mathrm{sim}_p(r, e)$, between the box and the role. Finally, the overall frame-semantic score $s(q)$ is the aggregation of sim_s and sim_p:

$$\mathrm{s}(q) = b_f\, \mathrm{sim}_s(i, f) + (1 - b_f)\, \mathrm{sim}_p(r, e), \quad (1)$$

where parameter $b_f \in \theta$ weights the contribution of the situation and participant scores to the overall score and is learned along all model parameters θ.

By definition of the output function s (Equ. 1), each role-object pair is scored independently of the decisions made for the other roles and regions of the same frame and image, respectively. Technically, this allows for the use of partially labeled training data, where not every realized role of a frame has been linked to its filler, as we will explain in Section 3.3.

Below we describe how we use our model to address the subtasks of role prediction and grounding (Section 3.1), respectively, for which we will report experimental results in Section 5.[2] In any case, the method is based on the visual-frame–semantic correspondence $\mathrm{s}(q)$ (Equ. 1), where we discard all candidates of role-filler pairings with a score less than zero.

Role Prediction Given an image i, we formulate the role prediction problem as a mapping L:

$$\begin{aligned} L :& \{i\} \times R_i \rightarrow F \times E \\ L(i, r) =& \operatorname*{arg\,max}_{(f,e), f \in F, e \in E_f} s(i, r, f, e) \end{aligned} \quad (2)$$

That is, the predicted role (and the frame it is associated with) which an image region $r \in R_i$ of i fills is that $e \in E$ to which r is most similar in the visual–frame-semantic space.

Role Grounding is the equivalent to linguistic semantic role labeling.[3] Given a frame f realized in i, we ground each role $e \in E_f$ in the region $r \in R_i$ with the highest visual–frame-semantic similarity to e:

$$\begin{aligned} G :& \{i\} \times \{f\} \times E_f \rightarrow R_i \\ G(i, f, e) =& \operatorname*{arg\,max}_{r \in R_i} s(i, r, f, e) \end{aligned} \quad (3)$$

Training We train the model by using a ranking criterion designed to give higher scores to true cross-modal frame-semantic combinations (i, r, f, e) than to mismatches, by a margin M. To this end, for each positive example $q = (i, r, f, e)$ of a training set Q, we sample K negative examples $q'_k = (i, r, f', e')$ of a frame f' and role $e' \in E_{f'}$ not true for image i and box r,[4] and learn model parameters θ by minimizing the maximum margin hinge loss function on the tuples (q, q') (Equ. 4) . Ideally, using this loss function would guide the parameter learning towards mapping images and the frames they evoke, and regions and the roles they fill, respectively, nearby each other in the cross-modal spaces.

$$\theta = \operatorname*{arg\,min}_{\theta} \sum_{q \in Q} \frac{1}{K} \sum_{k=1}^{K} \max(0, M - \mathrm{s}(q) + \mathrm{s}(q'_k)) \quad (4)$$

Margin M is found during hyperparameter optimization on a validation set.

---

[2] We refer to Appendix A.2 in the supplemental material for the production of the structure $A_{i,f}$ of all role–filler pairs for a frame f evoked by image i.

[3] More formally, the task of SRL is the determination of the arguments and their semantic roles of a predicate in a sentence.

[4] We could extend the model to also sample a negative image and box for f and e, and a negative role r' for f that is filled by another box in the image. We refrain from this since we create our training data from automatically labeled data, which hence could contain erroneous role-filler pairs.

(1a) [r5 A man] is being placed in [r4 a police car] by [r1 a uniformed officer].

(1b) [r1,r2 The police] arresting [r5 someone] on [r3 a busy city street].

(1c) [r5 A young guy] is getting arrested.

(2a) PLACING (`Theme`:r5/A man,
`Goal`:r4/a police car,
`Agent`:r1/a uniformed officer)

(2b) ARREST (`Authorities`:r1,r2/The police,
`Suspect`:r5/someone,
`Place`:r3/on a busy city street)

(2c) ARREST (`Suspect`:r5/A young guy)

Figure 4: Flickr30k captions for the image in Fig. 2. Left: Flickr30k Entities annotations of the mentioned objects with unique entity ids. Right: Frame-semantic annotations of the sentences, output by PathLSTM (Roth, 2016; Roth and Lapata, 2016).

3.3 Using Linguistic Knowledge for Data Creation

SRL systems in NLP research use training data which have been carefully created by linguistic experts (e.g., Ruppenhofer et al. (2006); Palmer et al. (2005)) for many years. To train our model on the *visual* SRL task, we build upon the annotation efforts made in NLP. The exploitation of existing resources which were developed for the analogous goal means to get around the time-consuming and costly annotation effort involved in the creation of training data. Moreover, adopting an established framework in NLP for shallow semantic representations (FrameNet, Ruppenhofer et al. (2006), in our case), including the therein defined frame and role labels, could facilitate cross-modal interactions—advances in vSRL can help to improve SRL and vice versa, or jointly draw inferences from both modalities (e.g., a text and its illustration).

Our data creation approach is to use a (linguistic) SRL system to extract frame-semantic annotations from a corpus of images paired with captions. We use the Flickr30k Entities dataset (Plummer et al., 2015)[5] which contains 30k images and five captions per image. We chose this dataset since its captions are augmented with entity mention annotations, associating them with the 276k manually annotated bounding boxes (i.e., entities are grounded in the image). To create the set $Q = \{(i^{(j)}, r^{(k_j)}, f^{(l_j)}, e^{(l_j,k_j)}) | j \in \{1, \ldots, 30k\}\}$ of training instances, we run PathLSTM (Roth, 2016; Roth and Lapata, 2016) on all captions, and extract all semantic frame annotations whose roles are filled by a grounded entity. As a result, our training corpus comprises images, the frames they evoke, and the associated semantic roles paired with their grounded fillers (i.e., bounding boxes).

Sentences (1a)–(1c) in Figure 4 (left), for example, are three human produced captions for the image in Figure 2, in which entity mentions are linked to their image regions (indicated by colors). Using PathLSTM, we extract the grounded frame-semantic annotations (2a)–(2c) (Fig. 4, right), which results in the following six instances of our corpus Q:

$(\text{img}_1, r_5, \text{PLACING}, \texttt{Theme})$
$(\text{img}_1, r_1, \text{PLACING}, \texttt{Agent})$
$(\text{img}_1, r_4, \text{PLACING}, \texttt{Goal})$

$(\text{img}_1, r_1\_r_2, \text{ARREST}, \texttt{Authorities})$
$(\text{img}_1, r_5, \text{ARREST}, \texttt{Suspect})$
$(\text{img}_1, r_3, \text{ARREST}, \texttt{Place})$

4 Data

Training Data We adopt the training, validation and test splits provided in the Flickr30k Entities dataset (Plummer et al., 2015) and create our dataset Q with the method described above. Some verbs and the frame types which they evoke occur very frequently in the set of annotations (e.g., BEING_LOCATED) and therefore allow the induction of a finer-grained frame inventory. Specifically, we transform each frame which is evoked by an individual *verb* (e.g., *stand* or *sit*) for at least 100 images (as obtained from the captions) in the Flickr30k Entities training split to a finer-grained frame type by concatenating it with the verb (e.g., BEING_LOCATED-*sit*). Finally, we keep all *frame types* (fine-grained or coarse) which had been assigned to at least 100 different images. This amounts to an inventory of 252 frame types (102 coarse types, e.g., STATEMENT), 1,409 frame-specific role types (e.g., STATEMENT.`speaker`), 169 role labels (e.g., `Speaker`) and 76,939 training instances. We derive our validation and test splits from the original splits on the basis of above modifications. See Table 1 for the quantitative details on the dataset, which we henceforth call *Flickr30k Roles*.

---

[5] See `web.engr.illinois.edu/~bplumme2/Flickr30kEntities`

	# inst.	**# types**				
	frame.role	frame.role		frames		roles
		fine	coarse	fine	coarse	
train	76,939	1,409	426	252	102	169
val	7,171	755	421	239	102	143
test	7,229	756	426	242	102	146

Table 1: Overview of our *Flickr30K Roles* dataset.

Reference Data *Flickr30k Roles* may contain false instances due to its creation on the basis of automatic frame–semantic annotations. ImSitu (Yatskar et al., 2016) is, to the best of our knowledge, the only existing benchmark dataset for vSRL. As explained in Section 2, however, it is image-based, and does not provide explicit links between roles and the regions which depict their fillers. It cannot be used for the evaluation of role prediction and grounding without additional annotations.

We therefore created a set of reference instances by presenting a subset of the *Flickr30k Roles* test data to two human subjects (both students of computational linguistics) for annotation. We chose all instances which agree in their frame label with instances extracted from at least two other captions of the underlying image. This amounts to 201 images and 715 instances. The annotators were presented with an image with relevant objects rendered by bounding boxes, along with the automatically grounded semantic frame annotations. Figure 5 gives an example image along with the 4 automatically obtained instances. They were asked to judge the correctness of the frame (e.g., INGESTION, Fig. 5), the verb (in the case of a fine-grained frame type; e.g., *eat*) and each of the role–filler links (e.g., `Ingestor`–226403). They further linked wrong role assignments to their correct fillers when possible. We created the reference set as the intersection of all correct instances of the two annotators (frame and role–filler linkings), which amounts to 554 instances.

Visual Representations We use high-dimensional distributed vectors to represent images and regions (bounding boxes), and represent the latter by additional contextual features. These encode a region's relative location and size with respect to the whole image (cf. (Mao et al., 2016)):

$$\left[\frac{x_{tl}}{W}, \frac{y_{tl}}{H}, \frac{x_{br}}{W}, \frac{y_{br}}{H}, \frac{w \cdot h}{W \cdot H}\right], \tag{5}$$

where (x_{tl}, y_{tl}) and (x_{br}, y_{br}) are the coordinates of the top left and bottom right corners of the bounding box, H and W are the height and width of the image, and h and w the height and width of the box. These features have been found useful for referring expression generation/interpretation for objects in images (Mao et al., 2016). We hypothesize that the relative position and size of an object can be likewise informative for the roles it can(not) realize. For example, an object that is located at the bottom of an image is probably rather the `Patient` of a KICKing event than the `Agent`.

INGESTION-***eat***			1–1
Role	Entity ID	(obj. name)	
`Ingestor`	226403	woman	1
`Ingestibles`	226404	lunch/hotdog	1
`Place`	226408	at_diner	1
`Ingestor`	226409	table/at_table	0

instances *label*

Figure 5: Automatically derived instances in *Flickr30k Roles* (colored, left columns) and the human correctness judgments of the frame, verb, and role fillers (right-most column; 1 is *correct*, 0 *wrong*). The object names were presented to facilitate the annotation, but are not part of the instance.

5 Experiments

We first evaluate our model in terms of different aspects related to visual SRL on the two subtasks role prediction and grounding (see Section 3).

Our second experiment assesses the usefulness of the learned frame-semantic image representations on the task of visual verb disambiguation: given an image and a verb, assign the correct sense of the verb, i.e., the one that describes the action depicted in the image (e.g., *play* an instrument; *play* sport). This task is different from visual SRL, but forms a prerequisite for it, since in frame semantics, roles are defined on the basis of frames evoked by verb *senses*.

Model Details For each bounding box and image, we use the VGG16 network (Simonyan and Zisserman, 2014), trained on ImageNet (Deng et al., 2009), to extract a $4,096$-dimensional feature vec-

		Fine-grained frame types							Coarse frame types			
		top-1-pred.			top-5 preds.			gt fr.	top-1-pred.		top-5 preds.	
		frame	fr.role	role	frame	fr.role	role	role	frame	fr.role	frame	fr.role
test set	Image-only	19.0	9.4	16.7	44.1	28.6	52.3	47.9	23.7	12.0	55.8	36.3
	ImgObject	18.7	12.8	24.1	44.9	33.8	61.2	64.3	22.6	15.5	55.5	41.4
	ImgObjLoc	18.6	13.5	25.9	46.8	35.7	62.2	65.7	23.0	16.7	56.5	43.2
reference	Image-only	**27.8**	13.2	17.2	55.2	39.3	57.3	50.2	**30.8**	14.6	67.8	46.6
	ImgObject	22.6	15.7	22.4	59.6	44.3	66.9	69.0	25.1	16.7	**68.8**	51.0
	ImgObjLoc	24.9	**17.4**	**23.6**	**60.2**	**47.3**	**68.6**	**70.3**	28.4	**19.7**	67.4	**53.3**

Table 2: Role prediction accuracy on the *Flickr30k Roles* test data and on its human corrected subset.

tor from the fully connected fc7 layer. To transform the feature vectors into the visual–frame-semantic embedding space, we use two two-layer networks which are composed of a layer with rectified linear activation units (relu) followed by a layer with tanh activations (see Fig. 3, top left). We furthermore concatenate the first hidden layer (relu layer) of each image region (i.e., box) with a vector of contextual features (relative box size and location, Equ. 5).

Frames and roles, in turn, are encoded as one-hot vectors and mapped to randomly initialized embedding layers, which are then transformed into the visual–frame-semantic representations using tanh activation layers (Fig. 3, bottom). We use the cosine similarity to quantify visual-frame–semantic correspondences in the cross-modal space (Equ. 1).

Throughout our experiments we compare our model (**ImgObjLoc**), which takes into account the contextual features (Equ. 5), to a model that does not use contextual box features (**ImgObject**), and one that only uses the image as visual input (**Image-only**). **Image-only** derives its cross-modal *role* representation by augmenting both, the image *and* the box input layers with the image's fc7 feature vector.

The network parameters were optimized using AdaGrad (Duchi et al., 2011) with a learning rate of 0.003. We monitored the role prediction performance on the validation set of Flickr30k Roles and kept the best performing model. See Appendix A.1 for further details on the model hyperparameters.

5.1 Exp.1: Semantic Role Prediction and Local Grounding

In the role prediction evaluation, the model is given an image and a bounding box, which represents a candidate role filler, and needs to predict the frame and role which the entity (or entities) in the box fills.

In the grounding experiment, the model is given an image, a frame and an associated role which is realized in the image, and needs to determine the correct role filler from a list of boxes. We report results on using ground truth boxes as well as box proposals, extracted with selective search (Uijlings et al., 2013). Regarding the latter, we apply the intersection over union (IoU) metric (e.g., Everingham et al. (2010)), and consider a role to be grounded in the correct box proposal $\tilde{r}$ if the area of overlap between $\tilde{r}$ and the reference box, divided by the area of their union, exceeds 50%.

Results We report top-1 and top-k accuracy (i.e., the frame and role is among the top-k scored predictions) on the *Flickr30k Roles* test and reference sets for both subtasks (recall that Flickr30k Entities provides ground truth alignments between entity mentions and objects).

Table 2 gives the results on role *prediction* with ground truth bounding boxes (i.e., for all entities which fill at least one semantic role). We report the accuracy for predicting the correct frame and role (columns fr.role), for predicting the correct frame (columns frame), and the correct role regardless of its frame (columns role; e.g., a prediction of STATEMENT.`Speaker` would be considered correct even if the reference was SPEAK_ON_TOPIC.`Speaker`). We further give results for the coarse frame types, where verbs are stripped off the frame labels (i.e., STATEMENT-*speak* is STATEMENT). Since the role prediction performance is equal for both frame types, we report the results for the fine-grained frames only.

As Table 2 shows, the models which use participant representations extracted from the relevant image regions (**ImgObject** and **ImgObjLoc**) perform better than **Image-only** which considers the

			Fine-grained frame types							Fine-grained frame types					
			top-1 pred. filler			top-3 pred. fillers				top-1 pred. filler			top-3 pred. fillers		
			frame	fr.role	role	frame	fr.role	role		frame	fr.role	role	frame	fr.role	role
test set	Random	gt	37.7	23.6	25.3	70.8	56.5	59.4	props	5.5	3.7	4.1	15.7	10.6	11.6
	ImgObject		55.9	55.1	58.0	83.2	84.0	78.7		10.5	11.3	11.7	21.8	21.4	21.2
	ImgObjLoc		56.6	56.6	59.4	83.1	85.1	79.7		11.5	12.8	13.3	22.3	22.6	22.5
reference	Random	gt	54.7	25.7	25.7	91.7	65.5	65.5	props	8.1	3.8	3.8	22.9	11.8	11.8
	ImgObject		78.9	62.1	62.1	95.8	88.2	83.6		13.7	12.8	12.8	39.6	30.9	28.2
	ImgObjLoc		**80.8**	**63.9**	**63.9**	**97.9**	**91.8**	**86.4**		**18.6**	**16.9**	**16.9**	**43.8**	**35.5**	**34.6**

Table 3: Role grounding accuracy on the *Flickr30k Roles* test data and on its human corrected subset. Instances with less than 2 (for top-1) or 3 (for top-3) gt filler candidates were discarded.

PathLSTM	Roles		
high prec.	`Ingestor`	`Source`	
	`Carrier`	`Speaker`	
low prec.	`Body_part`	`Activity`	`Seller`
	`Buyer`	`Manner`	`Purpose`

Table 4: Roles which were most difficult to predict by **ImgObjLoc**, in the order of their total frequency in the reference set (top left to bottom right), distinguished by the prediction precision of PathLSTM.

global image only, except for the top-1 frame prediction. This indicates that the two models are able to learn useful role-specific visual representations. Contextual features in the form of the relative size and location of a region (cf. Equ. 5) seems to be also beneficial, due to **ImgObjLoc** yielding the overall best results.

These features are furthermore beneficial for role *grounding* in automatically selected bounding boxes: When using automatically selected boxes, **ImgObjLoc** is significantly more effective than **ImgObject** in all settings (rows props, right block in Table 3). The Random baseline, which assigns each role randomly to a box in the image, performs unsurprisingly worst.

Interestingly, the models perform substantially better on the reference set than on the noisy test set (top and bottom blocks in Tables 2,3).[6] This indicates that they were able to generalize over wrong role-filler pairs in the training data, and are able to make correct predictions even for erroneous instances (see the qualitative analysis below). When assuming that the correct frame has been identified (columns gt fr.), the best role prediction accuracy reaches 70.3% on the reference set, and grounding accuracy with box proposals is at 35.5% (**ImgObjLoc**, Tables 2,3, respectively).

Finally, frame prediction proves to be a difficult task, especially for fine-grained frame types (e.g., BEING_LOCATED-*sit*; left block in Table 2).

Qualitative Analysis Notably, our analysis revealed that **ImgObjLoc** could correctly predict roles for cases in which PathLSTM failed, especially for highly visual entities (e.g., `performance` vs. `location`, `goal` vs. `path`). Overall, **ImgObjLoc** was often able to identify location roles which PathLSTM had missed, but may confuse the specific labels (e.g., `area` vs. `path` or `location`) for reasons discussed below. See Figure 6 for the recall of **ImgObjLoc** on the reference set for individual roles (top-20).

In an error analysis of the predictions of **ImgObjLoc** we identified several classes of errors. Typical errors in *role* prediction were in cases in which an image region contained multiple objects, and the system predicted a label for an object which was occluded by the target or vice versa (e.g., `ingestibles` vs. `source`; `clothing` vs. `wearer` or `body_part`; `path` vs. `area`). We found that this error was propagated from noise in the training data. Table 4 shows the roles which were most difficult to predict by **ImgObjLoc**, and which the textual SRL system (PathLSTM) could predict with a high precision (top; as calculated from the human annotations, cf. Section 4), or with a low precision (bottom), respectively. As may be expected, among these are also highly non-visual roles, such as `manner` and `purpose`.

Other noise propagated from the training data was caused by wrong frame predictions of PathLSTM (e.g., TRAVERSE-*pass* instead of BRINGING-*carry*; CONTAINING-*hold*.`contents` vs. IN-

---

[6] The accuracy scores on the uncorrected instances in the reference set yield comparable or worse accuracy scores than those on the test set, except for the top-5 predicted frames.

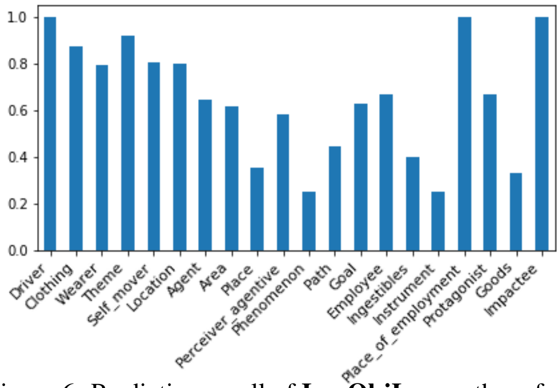

Figure 6: Prediction recall of **ImgObjLoc** on the reference set for the top-20 roles, ordered by their frequency.

Features	Motion	Non-motion
Random	76.7 ± 0.86	78.5 ± 0.39
MFS$^+$	76.1	80.0
CNN$^+$	82.3	80.0
Gella–CNN+O$^+$	83.0	80.0
Gella–CNN+C$^+$	82.3	80.3
CNN (reproduced)	83.1	79.8 ± 0.53
ImgObjLoc	**84.8 ± 0.69**	**80.4 ± 0.57**

Table 5: Sense prediction accuracy for motion (left) and non-motion verbs (right) using different image representations. $^+$ marks results taken from Gella et al. (2018). MFS is the most frequent sense heuristic.

GESTION.ingestibles). Frequent patterns of incorrect frame predictions were furthermore a failure of the system to distinguish between fine-grained frames (e.g., BEING_LOCATED-*sit* vs. *-lie* or SELF_MOTION-*walk* vs. *-run*), or between motion and non-motion actions (e.g., POSTURE vs. SELF_MOTION).

Finally, we observed that often the reference did not contain an actually valid frame which had been predicted by the system for an image, due to different levels of frame specificity, i.e., the output of **ImgObjLoc** was more specific (e.g., ASSISTANCE-*help*.helper vs. WEARING.wearer; OPERATE_VEHICLE-*ride*.vehicle vs. PERCEPTION_ACTIVE-*look*.location_of_perceiver) or it was more general (e.g., WEARING.wearer vs. IMPACT-*hit*.impactor).

5.2 Exp.2: Visual Verb Sense Disambiguation

We evaluate the effectiveness of the frame-semantic *image* representations that can be extracted with our **ImgObjLoc** model on the VerSe (visual Verb Sense disambiguation) dataset (Gella et al., 2018). It covers 90 verbs and 163 senses used to annotate $3,510$ images. We follow the supervised method applied in (Gella et al., 2018), divide VerSe into training and test data, and train logistic regression classifiers for sense prediction on 19 motion verbs and 19 non-motion verbs (those which have at least 20 images and at least 2 senses). Input to the sense classifiers are the frame-semantic image representations (second top cross-modal layer in Fig. 3) of the VerSe images, which we extract with the **ImgObjLoc** model, trained on Flickr30k Roles.

Table 5 gives the mean accuracy obtained on the test data (of 100 runs). Our **ImgObjLoc** vectors outperform all comparison models on motion verbs, including CNN-based image features and the best-performing models of (Gella et al., 2018), namely Gella–CNN+O and Gella–CNN+C (CNN features concatenated with predicted object labels and image captions, respectively). On non-motion verbs, the best models, including our own, perform only comparably to the most frequent sense heuristic. Note that we examine the simplest representation **ImgObjLoc** can yield, i.e., frame-semantic representations for individual images. More complex representations are left for future work. See Appendix A.3 for examples.

6 Conclusions

We addressed the task of grounding semantic roles of frames which an image evokes in the corresponding image regions of its fillers. We found that our model can be trained without the need of manual role annotations of image data, and that the frame-semantic image representations it learns can be used for related tasks. Encouraged by our findings, future work includes the exploration of the model and its learned frame-semantic representations for tasks such as the interpretation of multimodal scenes and stories and referring expressions.

Acknowledgments

We thank the anonymous reviewers, Leonie Harter, Christine Schäfer, Michael Roth, Gemma Boleda, Anna Rohrbach and Bernt Schiele. This research was supported by the German Research Foundation (DFG EXC 285), by the European Research Council (ERC Horizon 2020 grant agreement No 715154), and the Spanish Ramon y Cajal programme (grant RYC-2015-18907). This paper reflects the authors' view only, and the EU is not responsible for any use that may be made of the information it contains.

References

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A Large-scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338.

Charles J. Fillmore. 1982. *Frame Semantics*. Hanshin Publishing Co., Seoul, South Korea.

S. Gella, F. Keller, and M. Lapata. 2018. Disambiguating Visual Verbs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. In press.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

Saurabh Gupta and Jitendra Malik. 2015. Visual Semantic Role Labeling. *CoRR*, abs/1505.04474.

Ruiyu Li, Makarand Tapaswi, Renjie Liao, Jiaya Jia, Raquel Urtasun, and Sanja Fidler. 2017. Situation Recognition With Graph Neural Networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

Arun Mallya and Svetlana Lazebnik. 2017. Recurrent Models for Situation Recognition. *arXiv preprint arXiv:1703.06233*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2016. Generation and Comprehension of Unambiguous Object Descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. Semantic Role Labeling. volume 3, pages 1–103. Morgan & Claypool Publishers.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *CoRR*, abs/1505.04870.

Michael Roth. 2016. Improving Frame Semantic Parsing via Dependency Path Embeddings. In *Book of Abstracts of the 9th International Conference on Construction Grammar*, pages 165–167, Juiz de Fora, Brazil.

Michael Roth and Mirella Lapata. 2016. Neural Semantic Role Labeling with Dependency Path Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany.

Josef Ruppenhofer, Michael Ellsworth, Miriam R.L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2006. *FrameNet II: Extended Theory and Practice*. International Computer Science Institute, Berkeley, California. Distributed with the FrameNet data.

Dan Shen and Mirella Lapata. 2007. Using Semantic Roles to Improve Question Answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556.

Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. 2013. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171.

Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y. Chai. 2016. Grounded Semantic Role Labeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 149–159, San Diego, California. Association for Computational Linguistics.

Mark Yatskar, Vicente Ordonez, Luke Zettlemoyer, and Ali Farhadi. 2017. Commonly Uncommon: Semantic Sparsity in Situation Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mark Yatskar, Luke Zettlemoyer, and Ali Farhadi. 2016. Situation Recognition: Visual Semantic Role Labeling for Image Understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

A Supplemental Material

A.1 Model Parameters

We optimized the network parameters using AdaGrad (Duchi et al., 2011) with a learning rate of 0.003. Dropout with rate 0.25 was added on top of the visual–frame-semantic layers. We monitored the role prediction performance on the validation

Algorithm 1 vSRL algorithm which grounds each semantic role e of a frame f^* in at most one region $r \in R_i$ of image i. $s(.)$ denotes the visual-frame–semantic correspondence score (Equation (1)).

```
Require: a frame $f^*$ for image $i$, $R_i$    // Equ. (6)
  $S_{i,f^*} \leftarrow ((s(i,r,f^*,e), r, e) : r \in R_i, e \in E_{f^*})$,
  sorted in descending order of $s(.)$
  $A_{i,f^*} \leftarrow \emptyset$, grounded realization of $f^*$
  for all $(s,r,e) \in S_{i,f^*}$ do
      if $(r,.) \notin A_{i,f^*} \wedge (.,e) \notin A_{i,f^*}$ then
          $A_{i,f^*} \leftarrow A_{i,f^*} \cup \{(r,e)\}$
      end if
  end for
  return $A_{i,f^*}$
```

set of Flickr30k Roles and kept the best performing model, which was obtained after about 20 epochs for each model. For all models, the first visual hidden layer has 1000 dimensions, and all other layers have 250 units. The margin M (Equation 4 in the main paper) was set to 0.3, and $K = 10$ negative frame-role examples were sampled for each training instance.

A.2 Model: vSRL

The full vSRL task requires, given an image i, the computation of the set $A_{i,f}$ of role-object pairs which comprises the semantic roles of a frame f grounded to their fillers, i.e., $A_{i,f} = \{(r,e) | r \in R_i, e \in E_f\}$. Using our model, we first determine the frame f^* which image i evokes on the basis of all role-filler predictions, i.e.,

$$f^* = \arg\max_{f \in F} \sum_{\{r \in R_i, e \in E_f\}} s(i,r,f,e) \tag{6}$$

We then apply a simple algorithm (Algorithm 1) which chooses the filler–role pairs with maximum similarity from the set S_{i,f^*} of all scored frame-specific filler-role pairings for image i given frame f^* (cf. Equation 1 in the main paper), such that every role is grounded in at most one region, and every region fills at most one role (line 6, Algorithm 1).

A.3 Examples for VSD

Figure 7 shows example images of non-motion verbs for which ImgObjLoc achieved a high (*serve*, 95%) and a low accuracy (*reach*, 50%), their ground truth senses (gt) and the predictions of **ImgObjLoc** (sys).

serve (95%):
dish out, hand out something, often food
sys: ✓

serve (95%):
put a ball into play
sys: ✓

reach (50%)
extend physically/by influence
sys: ✓

reach (50%)
extend physically/by influence
sys: ✓

reach (50%):
gt: *pass or transfer something*
✗sys: *attain/arrive at a state, real or abstract*

reach (50%):
gt: *pass or transfer something*
✗sys: *extend physically/by influence*

Figure 7: VSD: Example images of non-motion verbs, their verb senses (gt) and our system's predictions (sys).

Commonsense Justification for Action Explanation

Shaohua Yang, Qiaozi Gao, Sari Saba-Sadiya, Joyce Y. Chai
Department of Computer Science and Engineering,
Michigan State University, East Lansing, MI 48824
{yangshao,gaoqiaoz,sadiyasa,jchai}@cse.msu.edu

Abstract

To enable collaboration and communication between humans and agents, this paper investigates learning to acquire commonsense evidence for action justification. In particular, we have developed an approach based on the generative Conditional Variational Autoencoder (`CVAE`) that models object relations/attributes of the world as latent variables and jointly learns a performer that predicts actions and an explainer that gathers commonsense evidence to justify the action. Our empirical results have shown that, compared to a typical attention-based model, `CVAE` achieves significantly higher performance in both action prediction and justification. A human subject study further shows that the commonsense evidence gathered by `CVAE` can be communicated to humans to achieve a significantly higher common ground between humans and agents.

1 Introduction

To make AI more accessible, transparent, and trustworthy, recent years have seen an increasing effort on Explainable AI (XAI) which develops explainable models that attempts to explain the agent's decision making behaviors while maintaining a high-level of performance. Two types of explanation have been explored by the research community: *introspective explanation* which addresses the process of decision making (e.g., how a decision is made) and *justification explanation* which gathers evidence to support a certain decision (Park et al., 2018; Biran and McKeown, 2017). In this paper we focus on justification explanation, particularly identifying ***commonsense evidence*** to justify the prediction of an action. The key question we are addressing is: when an AI agent makes a prediction about an action in the world, how can the system justify its prediction that makes sense to the human?

Humans have tremendous commonsense knowledge about actions in the world (e.g., key constituents of an action) which allows them to quickly recognize and infer actions in the environment from millions of available features (Rensink, 2000). As a first step in our investigation, we initiated a human study to observe the kind of commonsense reasoning used by humans to justify the prediction of an action. From this study, we identified several dimensions of commonsense evidence which is commonly used to explain an action. Motivated by this study, we frame our task as follows: given all the symbolic descriptions of the perceived physical world (e.g., object relations and attributes as a result of vision or other processing), the goal is to identify a small set of descriptions which can justify an action prediction in line with humans' commonsense knowledge about that action. The lack of commonsense knowledge is a major bottleneck in artificial agents which jeopardizes the common ground between humans and agents for successful communication. If artificial agents ever become partners with humans in joint tasks, the ability to learn and acquire commonsense evidence for action justification is crucial.

To address this problem, we developed an approach based on the generative Conditional Variational Autoencoder (`CVAE`). This approach models the perceived attributes/relations as latent variables and jointly learns a ***performer*** which predicts actions based on attributes/relations and a ***explainer*** which selects a subset of attributes/relations as commonsense evidence to justify the action prediction. Our empirical results on a subset of the Visual Genome data (Krishna et al., 2016) have shown that, compared to a typical attention-based model, `CVAE` has a significantly higher explanation ability in terms of identifying correct commonsense evidence to justify

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2627–2637
Brussels, Belgium, October 31 - November 4, 2018.

the predicted action. When adding the supervision of commonsense evidence during training, both the explainability and the performance (i.e., action prediction) are further improved.

As commonsense evidence is intuitive to humans, the agent's ability to select the right kind of commonsense evidence will allow the human and the agent to come to a common understanding of actions and their justifications, in other words, *common ground*. To evaluate the role of commonsense evidence in facilitating common ground, we conducted additional human subject studies. In these experiments, the agent is given a set of images and applies our models to predict actions and select commonsense evidence to justify the prediction. For each image, the agent communicates the selected commonsense evidence to the human. The human, who does not have access to the original image, makes a guess on the action only based on the communicated evidence. The agreement between the action guessed by the human and the action predicted by the agent is used to measure how well the selected commonsense evidence serves to bring the human and the agent to a common ground of perceived actions. Our experimental results have shown that the commonsense evidence selected by `CVAE` leads to a significantly higher common ground.

The contributions of this paper are three folds. First we identified several key dimensions of commonsense evidence, from a human's perspective, to justify concrete actions in the physical environment. These dimensions provide a basis for justification explanation that is aligned with human's commonsense knowledge about the action. Second we proposed a method using `CVAE` to jointly learn to predict actions and select commonsense evidence as action justification. `CVAE` naturally models the generation process of both actions and commonsense evidence. Inferring commonsense evidence is equivalent to the posterior inference of the `CVAE` model, which is flexible and powerful by incorporating actions as context. Our experimental results have shown a higher explainability of `CVAE` in action justification without sacrificing performance. Finally our dataset of commonsense evidence for action explanation is available to the community[1]. It can serve as a benchmark for future work on this topic.

2 Related Work

Advanced machine learning such as deep learning approaches have shown effectiveness in many applications, however, they often lack transparency and interpretability. This makes it difficult for humans to understand the agent's capabilities and limitations. To address this problem, there is a growing interest in Explainable AI. For example, previous work has applied high-precision rules to explain classifiers' decisions (Ribeiro et al., 2016, 2018). For Convolutional Neural Networks (CNNs), recent work attempts to explain model behaviors by mining semantic meanings of filters (Zhang et al., 2017a,b) or by generating language explanations (Hendricks et al., 2016; Park et al., 2018). An increasing amount of work on the Visual Question Answering (VQA) task (Antol et al., 2015; Lu et al., 2016) has also looked into more interpretable approaches, for example, by utilizing attention-based models (Fukui et al., 2016) or reasoning based on explicit evidence (Wang et al., 2017).

Specifically for action understanding, recent work explicitly models commonsense knowledge including causal relations (Gao et al., 2016; Forbes and Choi, 2017; Zellers and Choi, 2017; Gao et al., 2018) related to concrete actions, which can facilitate action explanation. Commonsense knowledge can be acquired from image annotations (Yatskar et al., 2016) or learned from visual abstraction (Vedantam et al., 2015). Different from the above work, our work here focuses on learning to acquire commonsense evidence for action justification.

3 A Study on Justification Explanation

While there is a rich literature on explanations in Psychology, Philosophy, and Linguistics, particularly for higher-level events and decision making (Thagard, 2000; Lombrozo, 2012; Dennett, 1987), explanations for recognition of lower-level concrete physical actions (e.g., drink, brush, cook, etc.) occurred in our daily life are rarely studied. One possible reason is that we humans are so intuitive in recognizing these actions, which are often taken for granted without the need for any further explanation. However, despite recent advances, the ability to recognize and understand actions in the real world is extremely challenging for artificial agents. Thus it becomes important for the agent to have an ability to explain and justify its

---

[1] The dataset is available at `https://github.com/yangshao/Commonsense4Action`

action prediction. What can be used to justify an action prediction, and more importantly, in a human understandable way? To address this question, we initiated a human study to examine what kind of evidence humans would gather in justifying their recognition of an action perceived from the physical world.

More specifically, we selected a set of 12 short video clips (each about 14 seconds) from the Microsoft Research Video to Text dataset (Xu et al., 2016). For each video clip, we asked human subjects to explain why they think a certain action is happening in the video. The answers were collected via an online interface. A total of about 140 responses from 67 Michigan State University engineering students were collected. From the data, we identified the following categories of evidence commonly used by the subjects in their justifications. Most responses contain multiple categories of explanation.

- **Transitive-Relation**. This kind of explanation does not directly focus on the structural relations between an action and its participants, but rather transits to the relation between the participant of the action and other related evidence. For example, using *a woman wears an apron* to justify the *cook* action. In the collected responses, 64% of them used transitive relations.
- **Subaction-Relation**. Lower-level subactions are used to justify a higher-level action. For example, the action is *cook* because there are sub-actions like *cutting* and *heating meat*. Almost 75% of the responses used subactions.
- **Spatial-Relation**. Spatial relations between the participants of the action play an important role. For example, *the knife is on the cutting board* is used to explain *cooking*; and *the water is in the bottle* to explain *drinking*. Around 15% of responses are in this category.
- **Effect-Attribute**. A change in the state of an object, in other words the effect state after the action, is often used as evidence. For example, *cucumber in small pieces* is used as the evidence for *chop*. Over 28% of the responses are in this category.
- **Associated-Attribute**. Other attributes associated with the participants of the action, but not the effect state of the participants as a result of the action (20%). While these attributes are not directly related to the action, they are linked to the action by association. For example, *banana is sliced* is used as evidence to justify *blend*.
- **Other**. Participants have also cited other commonsense such as the “definition” of the action (5%), or the manner associated with different sub actions(12%).

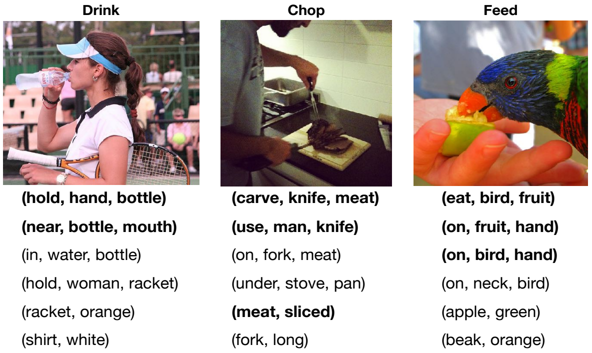

Figure 1: Examples of commonsense evidence selected by the crowd (in bold) from the list of relations and attributes.

Most of the above categories can be potentially perceived and represented through symbolic descriptions such as logic predicates to capture object attributes and relations between objects. This study has motivated us to collect additional data (Section 4) and formulate the task of commonsense justification as described in Section 5.

4 Data Collection

Motivated by the human study described above, we created a dataset based on the Visual Genome (VG) data (Krishna et al., 2016) for our investigation. Each image in the VG dataset is annotated with bounding boxes, attributes of the bounding boxes, and relations between the bounding boxes. The available annotation provides an ideal setup for us to focus on commonsense justification.

In this work, we are interested in the concrete physical actions that involve physical objects that can be perceived. We selected ten frequently occurred concrete actions: *feed, pull, ride, drink, chop, brush, fry, bake, blend, eat* and manually identified a set of images from the VG dataset depicting these actions. This has led to a dataset of 853 images with annotated ground-truth actions.

We conducted a crowd-source study to collect responses from the crowd in terms of common-

Table 1: The average number of available relations/attributes and the average number of annotated commonsense evidence relations/attributes across the corresponding images for each verb in the dataset.

	feed	**pull**	**ride**	**drink**	**chop**	**brush**	**fry**	**bake**	**blend**	**eat**
Rel#	15.49 ± 7.55	14.62 ± 9.36	12.42 ± 7.18	15.16 ± 9.89	12.00 ± 7.22	15.40 ± 8.93	14.02 ± 7.02	13.31 ± 7.27	14.37 ± 6.37	15.08 ± 6.87
Gold_Rel#	2.79 ± 1.28	1.86 ± 0.84	1.69 ± 0.83	2.41 ± 1.14	2.41 ± 1.66	2.26 ± 1.08	2.72 ± 2.06	2.25 ± 1.69	2.56 ± 1.84	2.52 ± 1.08
Att#	12.48 ± 7.11	13.60 ± 7.52	12.20 ± 7.13	10.86 ± 6.52	15.09 ± 6.82	12.31 ± 8.91	15.31 ± 7.16	13.44 ± 6.84	15.22 ± 7.18	11.98 ± 6.50
Gold_Att#	0.26 ± 0.48	0.20 ± 0.45	0.13 ± 0.40	0.30 ± 0.56	1.60 ± 1.33	0.22 ± 0.49	0.91 ± 1.26	0.93 ± 1.06	0.15 ± 0.40	0.41 ± 0.70

Table 2: Distributions of the categories of commonsense evidence relations/attributes for each verb.

	feed	**pull**	**ride**	**drink**	**chop**	**brush**	**fry**	**bake**	**blend**	**eat**
Transitive-Relation	0.10	0.14	0.15	0.11	0.11	0.13	0.12	0.18	0.15	0.09
Subaction-Relation	0.45	0.46	0.13	0.32	0.29	0.39	0.17	0.11	0.09	0.43
Spatial-Relation	0.45	0.40	0.72	0.57	0.60	0.48	0.71	0.71	0.76	0.48
Effect-Attribute	0.0	0.0	0.0	0.14	0.82	0.05	0.53	0.34	0.22	0.27
Associated-Attribute	1.0	1.0	1.0	0.86	0.18	0.95	0.47	0.66	0.78	0.73

sense evidence for action justification. As shown in Figure 1, for each image, we showed to the crowd (through Amazon Mechanical Turk) the image itself, the ground-truth action, and a list of relations/attributes. The workers were instructed to select the relations/attributes that were deemed to justify the corresponding action. We randomly assigned three workers to each image. The relations or attributes that were selected by the majority (two or more) workers were considered *gold commonsense evidence* for action justification.

Table 1 shows the average number of relations/attributes available (i.e., `Rel#` and `Att#`) for the corresponding images for each verb. It also shows the number of relations/attributes selected by the workers as commonsense evidence (i.e., `Gold_Rel#` and `Gold_Att#`). The average number of relations and attributes in each image for different actions varies slightly. However, only a small percentage of them are considered commonsense evidence. What's interesting is that the percentage of attributes considered good evidence is significantly less than the percentage of the relations. The sparsity of gold relations/attributes shows that it's a challenging task to learn an explainer for a target action.

We further inspected the selected gold commonsense relations and attributes. As shown in Table 2, they nicely fall into the categories of commonsense evidence discussed in Section 3. The ratios of `Transitive-Relation` are similar across different actions. The ratios of `Subaction-Relation` and `Spatial-Relation` vary for different verbs. For instance, *ride, bake, blend* tend to be justified by spatial relations more often than sub-actions. In addition, *feed, pull, ride* are rarely justified by `Effect-Attribute` while *chop* is mainly explained by the effect state of its direct object. These results will provide insight for generating justification explanations for a variety of verbs in the future.

5 Method

Before we formulate the problem, we will first give some formal definitions. The set of relations $\mathbf{R}$ is defined as $\{r_1, r_2, ..., r_m\}$ where each r_i is a tuple (r_i^p, r_i^s, r_i^o) corresponding to the *predicate*, *subject*, and *object*; and the set of attributes $\mathbf{E}$ is represented as $\{e_1, e_2, ..., e_n\}$ where each e_i is a tuple (e_i^o, e_i^p) corresponding to the *object* and *attribute*. We introduce $\mathbf{z}$ as a discrete vector $(\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{m+n})$ where $\mathbf{z}_i \in \{0, 1\}$ represents the hidden explainable variable. $\mathbf{z}$ is interpreted as an evidence selector: $\mathbf{z}_i = 1$ means the corresponding relation/attribute justifies the target action $\mathbf{a}$. We define $\mathbf{A}$ as the vocabulary of target actions. Based on all these definitions, our goal is to jointly select evidence $\mathbf{z}$ and predict target action $\mathbf{a} \in \mathbf{A}$. In other words, to learn the probability $\mathbf{p}(\mathbf{a}, \mathbf{z}|\mathbf{R}, \mathbf{E})$.

5.1 Conditional Variational Autoencoder

The varational autoencoder(`VAE`) (Kingma and Welling, 2013) is proposed as a generative model to combine the power of both directed continuous

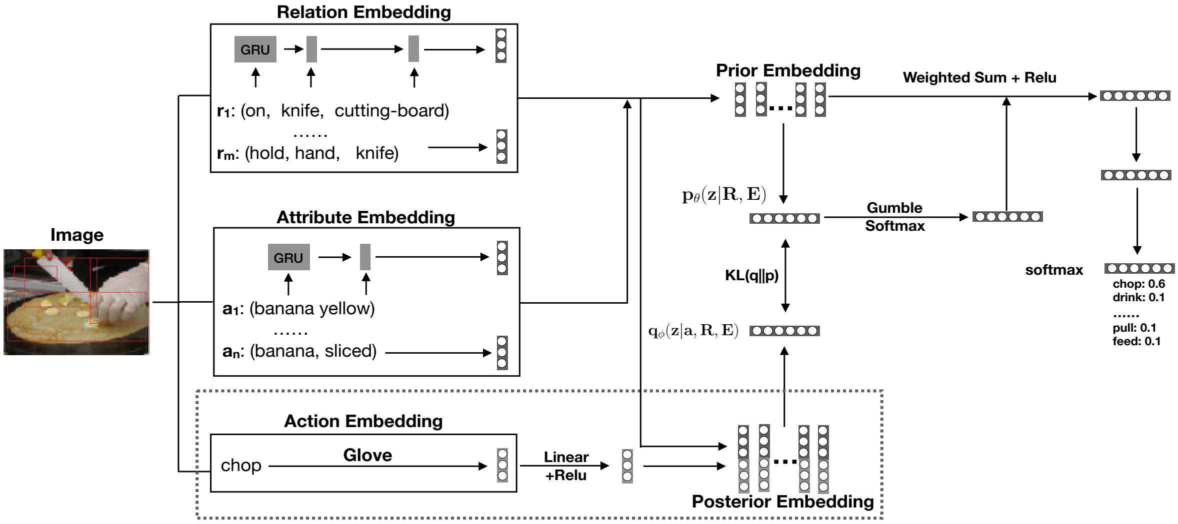

Figure 2: System architecture for the `CVAE` model. The dotted region is only used during the model training process.

or discrete graphical models and neural network with latent variables. The `VAE` models the generative process of a random variable $\mathbf{x}$ as following: first the latent variable $\mathbf{z}$ is generated from a prior probability distribution $\mathbf{p}(\mathbf{z})$, then a data sample $\mathbf{x}$ is generated from a conditional probability distribution $\mathbf{p}(\mathbf{x}|\mathbf{z})$. The `CVAE` (Zhao et al., 2017) is a natural extension of `VAE`: Both the prior distribution and conditional distribution now are conditioned on an additional context $\mathbf{c}$: $\mathbf{p}(\mathbf{z}|\mathbf{c})$ and $\mathbf{p}(\mathbf{x}|\mathbf{z},\mathbf{c})$.

In our task, we decompose the inference problem $\mathbf{p}(\mathbf{a},\mathbf{z}|\mathbf{R},\mathbf{E})$ into two smaller problems. The first sub-problem is to infer $\mathbf{p}(\mathbf{a}|\mathbf{R},\mathbf{E})$, which is a **performer**. The second problem is to infer $\mathbf{p}(\mathbf{z}|\mathbf{a},\mathbf{R},\mathbf{E})$ which is an **explainer**. These two problems are closely coupled, hence we model them jointly. The probability distribution $\mathbf{p}(\mathbf{a}|\mathbf{R},\mathbf{E})$ can be written as :

$$\mathbf{p}(\mathbf{a}|\mathbf{R},\mathbf{E}) = \sum_{\mathbf{z}} \mathbf{p}_\theta(\mathbf{a}|\mathbf{z},\mathbf{R},\mathbf{E})\mathbf{p}(\mathbf{z}|\mathbf{R},\mathbf{E})$$

Directly optimizing this conditional probability is not feasible. Usually the Evidence Lower Bound (`ELBO`) (Sohn et al., 2015) is optimized, which can be derived as the following:

$$\begin{aligned}
&\mathrm{ELBO}(\mathbf{a},\mathbf{R},\mathbf{E};\theta,\phi) \\
&= -\,\mathrm{KL}(\mathbf{q}_\phi(\mathbf{z}|\mathbf{a},\mathbf{R},\mathbf{E})||\mathbf{p}_\theta(\mathbf{z}|\mathbf{R},\mathbf{E})) \\
&+ \mathbf{E}_{\mathbf{q}_\phi(\mathbf{z}|\mathbf{a},\mathbf{R},\mathbf{E})}[\log p_\theta(\mathbf{a}|\mathbf{z},\mathbf{R},\mathbf{E})] \\
&\leq \log \mathbf{p}(\mathbf{a}|\mathbf{R},\mathbf{E})
\end{aligned} \tag{1}$$

The first KL divergence term is to minimize the distance between the posterior distribution and the prior distribution. The second term is to maximize the expectation of the target action based on the posterior latent distribution.

In most previous work using `VAE`, there is no explicit meaning for the hidden representation $\mathbf{z}$, thus it's hard for humans to interpret. For example, $\mathbf{z}$ is simply assumed as a Gaussian distribution or a categorical distribution. In order to have a more explicit representation for the purpose of explanation, our latent discrete variable $\mathbf{z}$ is used to indicate whether the corresponding relation or attribute can be used for justifying the action.

The whole system architecture is shown in Figure 2. From an image, we first extract a candidate relation set $\mathbf{R}$ and an attribute set $\mathbf{E}$. Every relation $\mathbf{r}$ and attribute $\mathbf{e}$ are embedded using a Gated Recurrent Neural Network (Chung et al., 2014).

$$\mathbf{r}^{emb} = \mathrm{GRU}([r^p, r^s, r^o])$$

$$\mathbf{e}^{emb} = \mathrm{GRU}([e^o, e^p])$$

The action $\mathbf{a}$ is represented by a GloVe embedding (Pennington et al., 2014), followed by another non-linear layer:

$$\mathbf{a}^{emb} = \mathrm{ReLU}(\mathbf{W}_i\mathbf{a}^{glove} + \mathbf{b}_i)$$

where $\mathbf{a}^{glove} \in \mathbb{R}^k$ is the pre-trained GloVe embedding. Then the latent variable $\mathbf{z}$ can be calculated as:

$$\mathbf{q}_\phi(\mathbf{z}|\mathbf{a},\mathbf{R},\mathbf{E}) = \mathrm{softmax}(\mathbf{W}_\mathbf{z}[\mathbf{U};\mathbf{a}^{emb}] + \mathbf{b}_\mathbf{z})$$

where $\mathbf{U} = [\mathbf{r}_1^{emb}, ..., \mathbf{r}_m^{emb}, \mathbf{e}_1^{emb}, ..., \mathbf{e}_n^{emb}]$ and $[\mathbf{U}, \mathbf{a}^{emb}]$ means the concatenation of $\mathbf{U}$ and $\mathbf{a}^{emb}$. and $W_{\mathbf{z}} \in \mathbb{R}^{2\times 2k}$ as we assume each $\mathbf{z}_i$ belongs to one of the two classes $\{0, 1\}$.

The prior distribution can be calculated as:

$$\mathbf{p}_\theta(\mathbf{z}|\mathbf{R}, \mathbf{E}) = \text{softmax}(\mathbf{W}_{\mathbf{z}}^{'}\mathbf{U} + \mathbf{b}_{\mathbf{z}}^{'})$$

The KL divergence between the prior random variable $\mathbf{z}_{prior}$ from $\mathbf{p}_\theta(\mathbf{z}|\mathbf{R}, \mathbf{E})$ and the posterior random variable $\mathbf{z}_{posterior}$ from $\mathbf{q}_\phi(\mathbf{z}|\mathbf{a}, \mathbf{R}, \mathbf{E})$ is:

$$\text{KL}(\mathbf{z}_{prior}, \mathbf{z}_{posterior}) = -p_i \log \frac{p_i}{p_i^{'}} - (1 - p_i) \log \frac{1 - p_i}{1 - p_i^{'}}$$

here $\mathbf{z}_{prior} \sim \text{Bern}\,(p_i)$, $\mathbf{z}_{posterior} \sim \text{Bern}\left(p_i^{'}\right)$.

Another challenge is that $\mathbf{z}$ is a discrete variable which blocks the gradient and makes the end-to-end training infeasible. Gumbel-Softmax (Jang et al., 2016) is a re-parameterization trick to deal with the discrete variables in the neural network. We use this trick to sample discrete $\mathbf{z}$. Then we do a weighted sum pooling between discretized $\mathbf{z}$ and $\mathbf{U}$:

$$\mathbf{h}_z = \text{ReLU}(\sum_i \mathbf{z}_i * \mathbf{U}_i)$$

$$\mathbf{h} = \text{ReLU}(\mathbf{W}_h\mathbf{h}_z + \mathbf{b}_h)$$

$$\mathbf{p}_\theta(\mathbf{a}|\mathbf{z}, \mathbf{R}, \mathbf{E}) = \text{softmax}(\mathbf{W}\mathbf{h} + \mathbf{b})$$

During training, we also add a sparsity regularization on the latent variable $\mathbf{z}$ besides the ELBO. So our final training objective is

$$\begin{aligned} \mathcal{L}_{CVAE} = &-\text{ELBO}(\mathbf{a}, \mathbf{R}, \mathbf{E}; \theta, \phi) \\ &+ \beta\, \text{KL}(\mathbf{q}_\phi(\mathbf{z}|\mathbf{a}, \mathbf{R}, \mathbf{E}) || \,\text{Bern}(0)) \end{aligned} \quad (2)$$

During testing, we have two objectives. First we want to infer the target action $\mathbf{a}$, which can be computed through sampling:

$$\begin{aligned} \mathbf{p}(\mathbf{a}|\mathbf{R}, \mathbf{E}) &= \sum_{\mathbf{z}} \mathbf{p}_\theta(\mathbf{z}|\mathbf{R}, \mathbf{E})\mathbf{p}_\theta(\mathbf{a}|\mathbf{z}, \mathbf{R}, \mathbf{E}) \\ &\approx \frac{1}{S}\sum_{s=1}^{S} \mathbf{p}_\theta(\mathbf{a}|\mathbf{z}_s, \mathbf{R}, \mathbf{E}) \end{aligned} \quad (3)$$

where $\mathbf{z}_s \sim \mathbf{p}(\mathbf{z}|\mathbf{R}, \mathbf{E})$ and S is the number of samples. After obtaining the predicted action $\hat{\mathbf{a}}$, the posterior explanation is inferred as $\mathbf{q}_\phi(\mathbf{z}|\hat{\mathbf{a}}, \mathbf{R}, \mathbf{E})$.

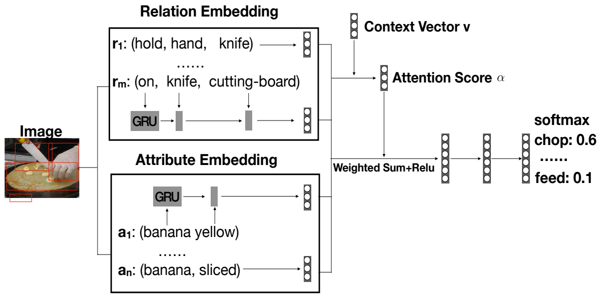

Figure 3: The system architecture for attention-based method.

5.2 Conditional Variational Autoencoder with Supervision (`CVAE+SV`)

In this setting, we assume we have the supervision for the discrete latent variable z, which is more like a multi-task setting. We optimize both the action prediction loss and the evidence selection loss. The final loss function is defined as:

$$\mathcal{L}_{SV} = \lambda\mathcal{L}_{CVAE} + (1 - \lambda)\mathcal{L}_{evidence}$$

where

$$\mathcal{L}_{evidence} = -\sum_k (\mathbf{z}_k \log \mathbf{p}(\hat{\mathbf{z}}_k) + (1 - \mathbf{z}_k)\log(1 - \mathbf{p}(\hat{\mathbf{z}}_k)))$$

in which $\mathbf{z}_k \in \{0, 1\}$ is the ground truth label, $\hat{\mathbf{z}}_k$ is the predicted label and λ is a hyper-parameter.

6 Evaluation on Action Explanation

To evaluate our model, we randomly split our dataset (853 images) into 60% for training, 20% for validation, and 20% for test. For all the models we use the Adam optimizer (Kingma and Ba, 2014) with a starting learning rate 1e-4. All other hyperparameters are tuned on the validation set.

6.1 Baseline: Attention Model

We use an attention-based model as a baseline, which is similar to the model originally proposed for document classification (Yang et al., 2016). The architecture is shown in Figure 3. Different from the CVAE-based method, this model directly learns a context parameter instead of learning from the posterior action context. The attention is calculated as:

$$\alpha_i = \frac{\exp(\mathbf{u}_i^T\mathbf{v})}{\sum_j \exp(\mathbf{u}_i^T\mathbf{v})}$$

where $\mathbf{v}$ is the context parameter, and $\mathbf{u}_i$ is the GRU embedding of the corresponding relation/attribute. The learned attention weights are used for the selection of commonsense evidence.

Table 3: Action prediction accuracy and evidence selection MAP.

	Action Accuracy	Evidence MAP
Attention	0.789	0.442
CVAE	0.835	0.572
CVAE+SV	0.871	0.690
Upper Bound	0.918	1.0

6.2 Evaluation Metrics and Comparison

Our evaluation compares the performance from the following models:

- `Baseline`. The attention model presented in Section 6.1.
- `CVAE`. The conditional variational autoencoder model presented in Section 5.1.
- `CVAE+SV`. The `CVAE` model with supervision as presented in Section 5.2.
- `Upper Bound`. We also calculate the upper bound of the `CVAE` model using the human annotated gold evidence.

For each of the above model, evaluate model performance on both action prediction (i.e., performer) and action justification (i.e., explainer)

- **Performer**: Accuracy is used to measure the percentage of actions that are correctly predicted by the model.
- **Explainer**: As discussed in Section 5, the binary random variable $\mathbf{z}$ is used to capture commonsense evidence. The probability of each $\mathbf{z}$ represents the model's belief that the corresponding evidence supports the action decision. As we hope to rank the gold evidence higher, the Mean Average Precision (MAP) metric is calculated for evaluating evidence selection.

6.3 Evaluation Results

The results are shown in Table 3. Since the `Upper Bound` method directly uses the human annotated gold evidence, its MAP for selecting evidence is always 1.0.

The `CVAE` model outperforms the attention-based model in both action prediction and evidence selection. This indicates that the `CVAE` model can incorporate a better guidance for evidence selection during the training process. One possible explanation is that the CVAE model incorporates the target action as the context during learning instead of directly learning a context parameter. Furthermore, after adding the evidence supervision, the `CVAE+SV` model gives even better performance in both action prediction and evidence selection. We notice that for the `CVAE+SV` model, its action prediction accuracy is approaching the upper bound 91.8%, however the evidence selection MAP is still far from the upper bound even with supervision.

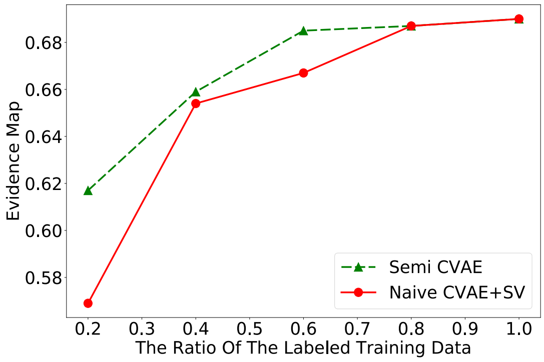

Figure 4: Evidence selection MAP for semi-supervised learning.

6.4 Semi-Supervised Learning

Although we have shown that adding supervision on the latent variable $\mathbf{z}$ improves the model performance, collecting this label information through human annotation is usually time consuming and expensive. In this section, we explore how semi-supervised learning can help to alleviate this difficulty.

As a generative model, `VAE` has shown its advantage on semi-supervised learning (Kingma et al., 2014). Following the method in (Kingma et al., 2014), our semi-supervised learning loss function is defined as:

$$\mathcal{L} = \sum_{(\mathbf{a},\mathbf{R},\mathbf{E},\mathbf{z})\sim\mathbf{p}_l} \mathcal{L}_{SV} + \sum_{(\mathbf{a},\mathbf{R},\mathbf{E})\sim\mathbf{p}_u} \mathcal{L}_{CVAE}$$

where $\mathcal{L}_{SV}$ is defined in section 5.2 and $\mathcal{L}_{CVAE}$ is detailed in section 5.1. In other words, the data sample with evidence label is fed to $\mathcal{L}_{SV}$, otherwise is fed to $\mathcal{L}_{CVAE}$.

The results are shown in Figure 4 where the x-axis shows the ratio of labeled examples. The incremental `Naive CVAE+SV` model only uses the labeled evidence examples while the `Semi CVAE` model also uses unlabeled evidence examples. The figure shows that the `Semi CVAE`

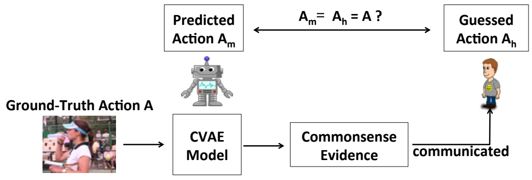

Figure 5: The experimental setup for the human subject study examining the role of commonsense justification towards common ground.

model outperforms the `Naive CVAE+SV` model. This indicates that the semi-supervised method can improve the evidence selection by making use of unlabeled examples.

7 Commonsense Justification towards Common Ground

In human-agent communication, the success of communication is largely dependent on common ground which captures shared knowledge, beliefs, or past experience (Clark, 1996). As commonsense evidence what humans use to justify actions, To validate this hypothesis, we conducted a human-subject experiment to examine the role of commonsense justification in facilitating common ground.

7.1 Experiment Setup

Figure 5 shows the setup of our experiment. The agent is provided with an image and applies various models (e.g., `CVAE`) to jointly predict the action and identify commonsense evidence. The human is provided with a list of six action choices and does not have access to the image. The agent communicates to the human only the identified commonsense evidence and the human makes a guess on the action from the candidate list purely based on the communicated evidence. The idea is that, if the human and the agent share the same beliefs about evidence to justify an action, then the action guessed by the human should be the same as the action predicted by the agent.

Generating Distracting Verbs. For each image, the human is provided with a list of six action/verb candidates. To generate this list, we mix four distracting verbs with the ground-truth action verb plus a default `Other`. Most of the distracting verbs come from the concrete action verbs made available by (Gao et al., 2018). We first manually filtered out the verbs which have the same meaning with the ground-truth verb. We then selected two groups of distracting verbs: an *easy* group (where the distracting verbs have larger distance from the ground-truth verb in the embedding space, with an average similarity of 0.284) and a *hard* group (more close to the ground-truth verbs with an average similarity of 0.479). The temperature based softmax distribution (Chorowski and Jaitly, 2016) was used to sample the easy and the hard distracting verbs based on the pre-trained GloVe (Pennington et al., 2014) embedding cosine similarity.

Process. A total of 170 images were used in this experiment, and 24 workers from AMT participated in our study. For each image, we applied three different models: `Attention` baseline, `CVAE`, and `CVAE+SV` to generate the commonsense evidence. An upper bound based on gold commonsense evidence was also measured. Note that, the agent has no knowledge of the human's action choices when generating the commonsense evidence. Theory of mind is an important aspect in human-agent communication. Incorporating human's action choices in justifying action is an interesting however a different problem which requires different solutions. In this paper, we only focus on the situation where the mind of the human is opaque to the agent.

For each model and each image under the easy or hard configurations, the top five predicted commonsense evidence (associated with the predicted action) were shown to a worker. The the worker was requested to select the most probable action from the distracting list only based on these five pieces of evidence. We randomly assigned three workers to each image. The majority of three selections was considered as the final answer. If all three selections disagreed, one worker's choice was randomly selected as the final answer.

Metrics for Common Ground. We use the agreement between the action guessed by the human and the action predicted by the agent to measure how well the selected commonsense evidence serves to bring the human and the agent to a common ground of perceived actions. More formally, as shown in Figure 5, given an image, suppose its ground-truth action is A, the action predicted by the agent/machine is A_m, and the action guessed by the human is A_h, the *Common Ground* is defined as: $A_m = A_h = A$. Here we also enforce that the predicted action should be the same as the ground-truth action. The percentage of trials based on different models that have led to a com-

	Attention	CVAE	CVAE+SV	Gold
Gold Action: Bake	A_m: Eat A_h: Bake	A_m: Bake A_h: Bake	A_m: Bake A_h: Bake	A_m: Bake A_h: Bake
	• The bread is **next to** the bread. • The bread is **on** the rack. • The bread is **on** the pan. • The man **has** keys. • The man **has** the band.	• The bread is **on** the rack. • The bread is **on** the pan. • The bread is **on** the tray. • The bread is **next to** the bread. • The bread is **baked**.	• The bread is **baked**. • The bread is **next to** the bread. • The person is **pushing** the tray. • The bread is **on** the pan. • The bread is **on** the rack.	• The bread is **on** the tray. • The person is **pushing** the tray. • The bread is **baked**.
Gold Action: Brush	A_m: Brush A_h: Skin	A_m: Brush A_h: Brush	A_m: Brush A_h: Brush	A_m: Brush A_h: Brush
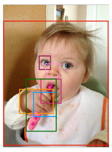	• The baby **has** a mouth. • The baby **has** a hand. • The baby **has** eyeballs. • The baby **has** fingers. • The baby **has** a nose.	• The hand **holds** the toothbrush. • The toothbrush **is** in the mouth. • The baby **has** a mouth. • The baby **has** fingers. • The baby **has** a nose.	• The hand **holds** the toothbrush. • The toothbrush **is** in the mouth. • The baby **has** eyeballs. • The baby **has** a mouth. • The baby **has** a hand.	• The toothbrush **is** in the mouth. • The hand **holds** the toothbrush.

Figure 6: Two examples of the common ground study based on different models. In each example, a ranked list of commonsense evidence generated by different models is shown. A_m captures the action predicted by the agent. A_h captures the action guessed by the human based on the selected commonsense evidence.

Table 4: Results from the human subject study on common ground.

	Attenton	CVAE	CVAE+SV	Gold
Easy	0.665	0.776	0.818	0.888
Hard	0.576	0.718	0.788	0.841

mon ground is measured and compared.

7.2 Experimental Results

Table 4 shows the comparison results among various models and the upper bound where the gold commonsense evidence provided to the human. It's not surprising that performance on common ground is worse in the *hard* configuration as the distracting verbs are more similar to the target action. The `CVAE`-based method is better than the attention-based method in facilitating common ground.

Figure 6 shows two examples of the top five predicted evidence under different models. For each model, it also shows the agent predicted action (A_m) and the human guessed action (A_h). In both examples, all models were able to establish a common ground except for the attention-based model. The evidence selected by the `CVAE+SV` model is clearly more accurate than the `CVAE` model and is more close to the ground-truth evidence. The second example shows that although the attention-based model predicts a correct target action, it fails to convey correct commonsense evidence to establish a common ground with the human.

8 Conclusion

This paper describes an approach for action justification using commonsense evidence. As demonstrated in our experiments, commonsense evidence is selected to align with humans' justification of an action and is therefore critical in establishing a common ground between humans and agents.

For all experiments in this paper, we use the annotated relations/attributes from the original Visual Genome data. As the state-of-the-art recall@50 on the relation detection with a limited vocabulary is only around 20% (Liang et al., 2018). Using annotated relations and attributes allows us to focus on the study of commonsense evidence and its role in action justification and common ground. Nevertheless, our proposed method has the potential to handle the erroneous relations/entities, e.g., as a result of vision processing, for example, by avoiding to select erroneous relations as they do not correlate with actions and other indicative relations/attributes. Our future work will extend the model and findings from this work to vision processing that will not only identify commonsense evidence but also explain where and how in the perceived environment the evidence is gathered.

Acknowledgement

This work was supported by IIS-1617682 from the National Science Foundation and the DARPA XAI program under a subcontract from UCLA (N66001-17-2-4029).

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Or Biran and Kathleen McKeown. 2017. Human-centric justification of machine learning predictions. *IJCAI, Melbourne, Australia*.

Jan Chorowski and Navdeep Jaitly. 2016. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Herbert H Clark. 1996. Using language. 1996. *Cambridge University Press: Cambridge*, 952:274–296.

D. Dennett. 1987. *The intentional Stance*. MIT Press.

Maxwell Forbes and Yejin Choi. 2017. Verb physics: Relative physical knowledge of actions and objects. *arXiv preprint arXiv:1706.03799*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Chai. 2016. Physical causality of action verbs in grounded language understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1814–1824.

Qiaozi Gao, Shaohua Yang, Joyce Chai, and Lucy Vanderwende. 2018. What action causes this? towards naive physical action-effect prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations.

Kongming Liang, Yuhong Guo, Hong Chang, and Xilin Chen. 2018. Visual relationship detection with deep structural ranking. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

Tania Lombrozo. 2012. Explanation and abductive inference. *Oxford handbook of thinking and reasoning*, pages 260–276.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. 2018. Multimodal explanations: Justifying decisions and pointing to the evidence. *arXiv preprint arXiv:1802.08129*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ronald A Rensink. 2000. The dynamic representation of scenes. *Visual cognition*, 7(1-3):17–42.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Knowledge Discovery and Data Mining (KDD)*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491.

Thagard. 2000. Probabilistic networks and explanatory coherence. *Cognitive Science Quarterly*.

Ramakrishna Vedantam, Xiao Lin, Tanmay Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Learning common sense through visual abstraction. In *Proceedings of the IEEE international conference on computer vision*, pages 2542–2550.

Peng Wang, Qi Wu, Chunhua Shen, and Anton van den Hengel. 2017. The vqa-machine: Learning how to use existing vision algorithms to answer new questions. In *Proc. CVPR*.

J. Xu, T. Mei, T. Yao, and Y. Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5288–5296.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Mark Yatskar, Vicente Ordonez, and Ali Farhadi. 2016. Stating the obvious: Extracting visual common sense knowledge. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 193–198.

Rowan Zellers and Yejin Choi. 2017. Zero-shot activity recognition with verb attribute induction. *arXiv preprint arXiv:1707.09468*.

Quanshi Zhang, Ruiming Cao, Feng Shi, Ying Nian Wu, and Song-Chun Zhu. 2017a. Interpreting cnn knowledge via an explanatory graph. *arXiv preprint arXiv:1708.01785*.

Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. 2017b. Interpretable convolutional neural networks. *arXiv preprint arXiv:1710.00935*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*.

Learning Personas from Dialogue with Attentive Memory Networks

Eric Chu *
MIT Media Lab
echu@mit.edu

Prashanth Vijayaraghavan *
MIT Media Lab
pralav@mit.edu

Deb Roy
MIT Media Lab
dkroy@media.mit.edu

Abstract

The ability to infer persona from dialogue can have applications in areas ranging from computational narrative analysis to personalized dialogue generation. We introduce neural models to learn persona embeddings in a supervised character trope classification task. The models encode dialogue snippets from IMDB into representations that can capture the various categories of film characters. The best-performing models use a multi-level attention mechanism over a set of utterances. We also utilize prior knowledge in the form of textual descriptions of the different tropes. We apply the learned embeddings to find similar characters across different movies, and cluster movies according to the distribution of the embeddings. The use of short conversational text as input, and the ability to learn from prior knowledge using memory, suggests these methods could be applied to other domains.

1 Introduction

Individual personality plays a deep and pervasive role in shaping social life. Research indicates that it can relate to the professional and personal relationships we develop (Barrick and Mount, 1993), (Shaver and Brennan, 1992), the technological interfaces we prefer (Nass and Lee, 2000), the behavior we exhibit on social media networks (Selfhout et al., 2010), and the political stances we take (Jost et al., 2009).

With increasing advances in human-machine dialogue systems, and widespread use of social media in which people express themselves via short text messages, there is growing interest in systems that have an ability to understand different personality types. Automated personality analysis based on short text analysis could open up a range of potential applications, such as dialogue agents that sense personality in order to generate more interesting and varied conversations.

We define persona as a person's social role, which can be categorized according to their conversations, beliefs, and actions. To learn personas, we start with the character tropes data provided in the CMU Movie Summary Corpus by (Bamman et al., 2014). It consists of 72 manually identified commonly occurring character archetypes and examples of each. In the character trope classification task, we predict the character trope based on a batch of dialogue snippets.

In their original work, the authors use Wikipedia plot summaries to learn latent variable models that provide a clustering from words to topics and topics to personas – their persona clusterings were then evaluated by measuring similarity to the ground-truth character trope clusters. We asked the question – could personas also be inferred through dialogue? Because we use quotes as a primary input and not plot summaries, we believe our model is extensible to areas such as dialogue generation and conversational analysis.

Our contributions are:

1. Data collection of IMDB quotes and character trope descriptions for characters from the CMU Movie Summary Corpus.
2. Models that greatly outperform the baseline model in the character trope classification task. Our experiments show the importance of multi-level attention over words in dialogue, and over a set of dialogue snippets.
3. We also examine how prior knowledge in the form of textual descriptions of the persona categories may be used. We find that a 'Knowledge-Store' memory initialized with descriptions of the tropes is particularly useful. This ability may allow these models to be used more flexibly in new domains and with

---

*The first two authors contributed equally to this work.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2638–2646
Brussels, Belgium, October 31 - November 4, 2018.

Character Trope	Character	Movie
Corrupt corporate executive	Les Grossman	Tropic Thunder
Retired outlaw	Butch Cassidy	Butch Cassidy and the Sundance Kid
Lovable rogue	Wolverine	X-Men

Table 1: Example tropes and characters

different persona categories.

2 Related Work

Prior to data-driven approaches, personalities were largely measured by asking people questions and assigning traits according to some fixed set of dimensions, such as the Big Five traits of openness, conscientiousness, extraversion, agreeability, and neuroticism (Tupes and Christal, 1992). Computational approaches have since advanced to infer these personalities based on observable behaviors such as the actions people take and the language they use (Golbeck et al., 2011).

Our work builds on recent advances in neural networks that have been used for natural language processing tasks such as reading comprehension (Sukhbaatar et al., 2015) and dialogue modeling and generation (Vinyals and Le, 2015; Li et al., 2016; Shang et al., 2015). This includes the growing literature in attention mechanisms and memory networks (Bahdanau et al., 2014; Sukhbaatar et al., 2015; Kumar et al., 2016).

The ability to infer and model personality has applications in storytelling agents, dialogue systems, and psychometric analysis. In particular, personality-infused agents can help "chit-chat" bots avoid repetitive and uninteresting utterances (Walker et al., 1997; Mairesse and Walker, 2007; Li et al., 2016; Zhang et al., 2018). The more recent neural models do so by conditioning on a 'persona' embedding – our model could help produce those embeddings.

Finally, in the field of literary analysis, graphical models have been proposed for learning character personas in novels (Flekova and Gurevych, 2015; Srivastava et al., 2016), folktales (Valls-Vargas et al., 2014), and movies (Bamman et al., 2014). However, these models often use more structured inputs than dialogue to learn personas.

3 Datasets

Characters in movies can often be categorized into archetypal roles and personalities. To understand the relationship between dialogue and personas, we utilized three different datasets for our models: (a) the Movie Character Trope dataset, (b) the IMDB Dialogue Dataset, and (c) the Character Trope Description Dataset. We collected the IMDB Dialogue and Trope Description datasets, and these datasets are made publicly available [1].

3.1 Character Tropes Dataset

The CMU Movie Summary dataset provides tropes commonly occurring in stories and media (Bamman et al., 2014). There are a total of 72 tropes, which span 433 characters and 384 movies. Each trope contains between 1 and 25 characters, with a median of 6 characters per trope. Tropes and canonical examples are shown in Table 1.

3.2 IMDB Dialogue Snippet Dataset

To obtain the utterances spoken by the characters, we crawled the IMDB Quotes page for each movie. Though not every single utterance spoken by the character may be available, as the quotes are submitted by IMDB users, many quotes from most of the characters are typically found, especially for the famous characters found in the Character Tropes dataset. The distribution of quotes per trope is displayed in Figure 1. Our models were trained on 13,874 quotes and validated and tested on a set of 1,734 quotes each.

We refer to each IMDB quote as a (contextualized) dialogue snippet, as each quote can contain several lines between multiple characters, as well as italicized text giving context to what might be happening when the quote took place. Figure 2 show a typical dialogue snippet. 70.3% of the quotes are multi-turn exchanges, with a mean of 3.34 turns per multi-turn exchange. While the character's own lines alone can be highly indicative of the trope, our models show that accounting for context and the other characters' lines and context improves performance. The context, for instance, can give clues to typical scenes and actions that are associated with certain tropes, while the other characters' lines give further detail into

---

[1] `https://pralav.github.io/emnlp_personas/`

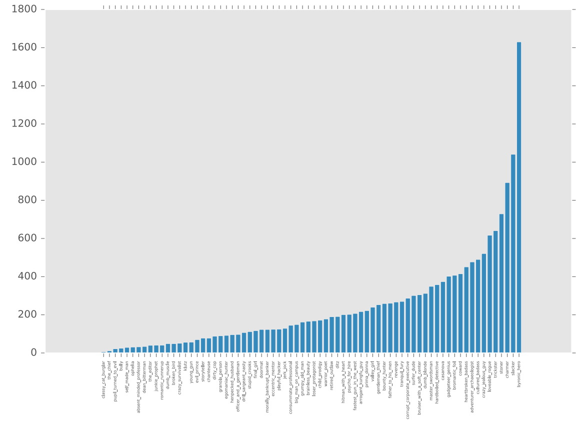

Figure 1: Number of IMDB dialogue snippets per trope

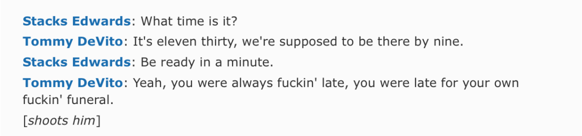
Stacks Edwards: What time is it?
Tommy DeVito: It's eleven thirty, we're supposed to be there by nine.
Stacks Edwards: Be ready in a minute.
Tommy DeVito: Yeah, you were always fuckin' late, you were late for your own fuckin' funeral.
[*shoots him*]

Figure 2: Example IMDB dialogue snippet containing multiple characters and context.

the relationship between the character and his or her environment.

3.3 Character Trope Description Dataset

We also incorporate descriptions of each of the character tropes by using the corresponding descriptions scraped from TVTropes[2]. Each description contains several paragraphs describing typical characteristics, actions, personalities, etc. about the trope. As we demonstrate in our experiments, the use of these descriptions improves classification performance. This could allow our model to be applied more flexibly beyond the movie character tropes – as one example, we could store descriptions of personalities based on the Big Five traits in our Knowledge-Store memory.

4 Problem Formulation

Our goal is to train a model that can take a batch of dialogue snippets from the IMDB dataset and predict the character trope.

Formally, let N_P be the total number of character tropes in the character tropes dataset. Each character C is associated with a corresponding ground-truth trope category P. Let $S = (D, E, O)$ be a dialog snippet associated with a character C, where $D = [w_{D_1}, w_{D_2} ..., w_{D_T}]$ refers to the character's own lines, $E = [w_{E_1}, w_{E_2} ..., w_{E_T}]$ is the contextual information and $O = [w_{O_1}, w_{O_2} ..., w_{O_T}]$ denotes the other characters' lines. We define all three components of S to have fixed sequence length T and pad when necessary. Let N_S be the total number of dialogue snippets for a trope. We sample a set of N_{diag} (where $N_{diag} \ll N_S$) snippets from N_S snippets related to the trope as inputs to our model.

5 Attentive Memory Network

The Attentive Memory Network consists of two major components: (a) Attentive Encoders, and (b) a Knowledge-Store Memory Module. Figure 3 outlines the overall model. We describe the components in the following sections.

5.1 Attentive Encoders

Not every piece of dialogue may be reflective of a latent persona. In order to learn to ignore words and dialogue snippets that are not informative about the trope we use a multi-level attentive encoder that operates at (a) the individual snippet level, and (b) across multiple snippets.

Attentive Snippet Encoder

The snippet encoder extracts features from a single dialogue snippet S, with attention over the words in the snippet. A snippet $S = (D, E, O)$ is fed to the encoder to extract features from each of these textual inputs and encode them into an embedding space. We use a recurrent neural network as our encoder, explained in detail in Section 5.1.1. In order to capture the trope-reflective words from the input text, we augment our model with a self-attention layer which scores each word in the given text for its relevance. Section 5.1.2 explains how the attention weights are computed. The output of this encoder is an encoded snippet embedding $S^e = (D^e, E^e, O^e)$.

Attentive Inter-Snippet Encoder

As shown in Figure 3, the N_{diag} snippet embeddings S^e from the snippet encoder are fed to our inter-snippet encoder. This encoder captures inter-snippet relationship using recurrence over the N_{diag} snippet embeddings for a given trope and determines their importance. Some of the dialogue snippets may not be informative about the trope, and the model learns to assign low attention scores to such snippets. The resulting attended

---

[2] http://tvtropes.org

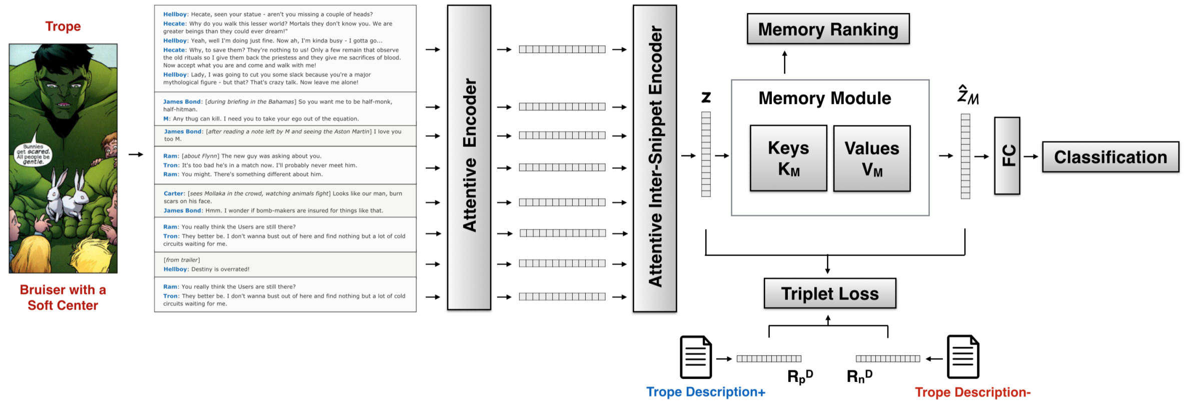

Hellboy: Hecate, seen your statue - aren't you missing a couple of heads?
Hecate: Why do you walk this lesser world? Mortals they don't know you. We are greater beings than they could ever dream!"
Hellboy: Yeah, well I'm doing just fine. Now ah, I'm kinda busy - I gotta go...
Hecate: Why, to save them? They're nothing to us! Only a few remain that observe the old rituals so I give them back the priestess and they give me sacrifices of blood. Now accept what you are and come and walk with me!
Hellboy: Lady, I was going to cut you some slack because you're a major mythological figure - but that? That's crazy talk. Now leave me alone!

James Bond: [*during briefing in the Bahamas*] So you want me to be half-monk, half-hitman.
M: Any thug can kill. I need you to take your ego out of the equation.

James Bond: [*after reading a note left by M and seeing the Aston Martin*] I love you too M.

Ram: [*about Flynn*] The new guy was asking about you.
Tron: It's too bad he's in a match now. I'll probably never meet him.
Ram: You might. There's something different about him.

Carter: [*sees Mollaka in the crowd, watching animals fight*] Looks like our man, burn scars on his face.
James Bond: Hmm. I wonder if bomb-makers are insured for things like that.

Ram: You really think the Users are still there?
Tron: They better be. I don't wanna bust out of here and find nothing but a lot of cold circuits waiting for me.

[*from trailer*]
Hellboy: Destiny is overrated!

Ram: You really think the Users are still there?
Tron: They better be. I don't wanna bust out of here and find nothing but a lot of cold circuits waiting for me.

Figure 3: Illustration of the Attentive Memory Network. The network takes dialogue snippets as input and predicts its associated character trope. In this example, dialogue snippets associated with the character trope "Bruiser with a Soft Corner" is given as input to the model.

summary vector from this phase is the persona representation z, defined as:

$$\begin{gathered} z = \gamma_D^s D^s + \gamma_E^s E^s + \gamma_O^s O^s \\ \gamma_D^s + \gamma_E^s + \gamma_O^s = 1 \end{gathered} \tag{1}$$

where $\gamma_D^s, \gamma_E^s, \gamma_O^s$ are learnable weight parameters. D^s, E^s, O^s refers to summary vectors of the N_{diag} character's lines, contextual information, and other characters' lines, respectively. In Section 7, we experiment with models that have γ_E^s and γ_O^s set to 0 to understand how the contextual information and other characters' lines contribute to the overall performance.

5.1.1 Encoder

Given an input sequence $(x_1, x_2, ..., x_T)$, we use a recurrent neural network to encode the sequence into hidden states $(h_1, h_2, ..., h_T)$. In our experiments, we use a gated recurrent network (GRU) (Chung et al., 2014) over LSTMs (Hochreiter and Schmidhuber, 1997) because the latter is more computationally expensive. We use bidirectional GRUs and concatenate our forward and backwards hidden states to get $\overleftrightarrow{h_t}$ for $t = 1, ..., T$.

5.1.2 Attention

We define an attention mechanism $Attn$ that computes s from the resultant hidden states $\overleftrightarrow{h_t}$ of a GRU by learning to generate weights α_t. This can be interpreted as the relative importance given to a hidden state h_t to form an overall summary vector for the sequence. Formally, we define it as:

$$a_t = f_{attn}(h_t) \tag{2}$$

$$\alpha_t = softmax(a_t) \tag{3}$$

$$s = \sum_{t=1}^{T} \alpha_t h_t \tag{4}$$

where f_{attn} is a two layer fully connected network in which the first layer projects $h_t \in \mathbb{R}^{d_h}$ to an attention hidden space $g_t \in \mathbb{R}^{d_a}$, and the second layer produces a relevance score for every hidden state at timestep t.

5.2 Memory Modules

Our model consists of a read-only 'Knowledge-Store' memory, and we also test a recent read-write memory. External memories have been shown to help on natural language processing tasks (Sukhbaatar et al., 2015; Kumar et al., 2016; Kaiser and Nachum, 2017), and we find similar improvements in learning capability.

5.2.1 Knowledge-Store Memory

The main motivation behind the Knowledge-Store memory module is to incorporate prior domain knowledge. In our work, this knowledge refers to the trope descriptions described in Section 3.3.

Related works have initialized their memory networks with positional encoding using word embeddings (Sukhbaatar et al., 2015; Kumar et al., 2016; Miller et al., 2016). To incorporate the descriptions, we represent them with skip thought vectors (Kiros et al., 2015) and use them to initialize the memory keys $K_M \in \mathbb{R}^{N_P \times d_K}$, where N_P is the number of tropes, and d_K is set to the size of embedded trope description R^D, i.e. $d_K = ||R^D||$.

The values in the memory represent learnable embeddings of corresponding trope categories $V_M \in \mathbb{R}^{N_P \times d_V}$, where d_V is the size of the trope category embeddings. The network learns to use the persona representation z from the encoder phase to find relevant matches in the memory. This corresponds to calculating similarities between z and the keys K_M. Formally, this is calculated as:

$$z_M = f_z(z) \tag{5}$$

$$p_M^{(i)} = softmax(z_M \cdot K_M[i]) \quad \forall i \in \{1, .., N_P\} \tag{6}$$

where $f_z : \mathbb{R}^{d_h} \mapsto \mathbb{R}^{d_K}$ is a fully-connected layer that projects the persona representation in the space of memory keys K_M. Based on the match probabilities $p_M^{(i)}$, the values V_M are weighted and cumulatively added to the original persona representation as:

$$r_{out} = \sum_{i=1}^{N_P} p_M^{(i)} \cdot V_M[i] \tag{7}$$

We iteratively combine our mapped persona representation z_M with information from the memory r_{out}. The above process is repeated n_{hop} times. The memory mapped persona representation z_M is updated as follows:

$$z_M^{hop} = f_r(z_M^{hop-1}) + r_{out} \tag{8}$$

where $z_M^0 = z_M$, and $f_r : \mathbb{R}^{d_V} \mapsto \mathbb{R}^{d_K}$ is a fully-connected layer. Finally, we transform the resulting $z_M^{n_{hop}}$ using another fully-connected layer, $f_{out} \in \mathbb{R}^{d_K} \mapsto \mathbb{R}^{d_h}$, via:

$$\hat{z}_M = f_{out}(z_M^{n_{hop}}) \tag{9}$$

5.2.2 Read-Write Memory

We also tested a Read-Write Memory following Kaiser et. al (Kaiser and Nachum, 2017), which was originally designed to remember rare events. In our case, these 'rare' events might be key dialogue snippets that are particularly indicative of latent persona. It consists of keys, which are activations of a specific layer of model, i.e. the persona representation z, and values, which are the ground-truth labels, i.e. the trope categories. Over time, it is able to facilitate predictions based on past data with similar activations stored in the memory. For every new example, the network writes to memory for future look up. A memory with memory size N_M is defined as:

$$M = (K_{N_M \times d_H}, V_{N_M}, A_{N_M}) \tag{10}$$

Memory Read We use the persona embedding z as a query to the memory. We calculate the cosine similarities between z and the keys in M, take the softmax on the top-k neighbors, and compute a weighted embedding $\hat{z}_M$ using those scores.

Memory Write We update the memory in a similar fashion to the original work by (Kaiser and Nachum, 2017), which takes into account the maximum age of items as stored in A_{N_M}.

6 Objective Losses

To train our model, we utilize the different objective losses described below.

6.1 Classification Loss

We calculate the probability of a character belonging to a particular trope category P through Equation 11, where $f_P : \mathbb{R}^{d_h} \mapsto \mathbb{R}^{N_P}$ is a fully-connected layer, and z is the persona representation produced by the multi-level attentive encoders described in Equation 1. We then optimize the categorical cross-entropy loss between the predicted and true tropes as in Equation 12, where N_P is the total number of tropes, q_j is the predicted distribution that the input character fulls under trope j, and $p_j \in \{0, 1\}$ denotes the ground-truth of whether the input snippets come from characters from the j^{th} trope.

$$q = softmax(f_P(z)) \tag{11}$$

$$J_{CE} = \sum_{j=1}^{N_P} -p_j \log(q_j) \tag{12}$$

6.2 Trope Description Triplet Loss

In addition to using trope descriptions to initialize the Knowledge-Store Memory, we also test learning from the trope descriptions through a triplet loss (Hoffer and Ailon, 2015). We again use the skip thought vectors to represent the descriptions. Specifically, we want to maximize the similarity of representations obtained from dialogue snippets with their corresponding description, and minimize their similarity with negative examples. We implement this as:

$$R^P = f_D(z) \tag{13}$$

$$J_T = max(0, s(R^P, R_n^D) - s(R^P, R_p^D) + \alpha_T) \tag{14}$$

where $f_D : \mathbb{R}^{d_h} \mapsto \mathbb{R}^{||R^D||}$ is a fully-connected layer. The triplet ranking loss is then Equation 14, where α_T is a learnable margin parameter and $s(\cdot, \cdot)$ denotes the similarity between trope embeddings (R^P), positive (R_p^D) and negative (R_n^D) trope descriptions.

Trope Description Triplet Loss with Memory Module

If a memory module is used, we compute a new triplet loss in place of the one described in Equation 14. Models that use a memory module should learn a representation $\hat{z}_M$, based on either the prior knowledge stored in the memory (as in Knowledge-Store memory) or the top-k key matches (as in Read-Write memory), that is similar to the representation of the trope descriptions.

This is achieved by replacing the persona embedding z in Equation 13 with the memory output $\hat{z}_M$ as shown in Equation 15, where $f_{D_M} : \mathbb{R}^{d_h} \mapsto \mathbb{R}^{||R^D||}$ is a fully-connected layer. To compute the new loss, we combine the representations obtained from Equations 13 and 15 through a learnable parameter γ that determines the importance of each representation. Finally, we utilize this combined representation $\hat{R}^P$ to calculate the loss as shown in Equation 17.

$$R^P_M = f_{D_M}(\hat{z}_M) \tag{15}$$

$$\hat{R}^P = \gamma R^P + (1 - \gamma) R^P_M \tag{16}$$

$$J_{MT} = max(0, s(\hat{R}^P, R^D_n) - s(\hat{R}^P, R^D_p) + \alpha_{MT}) \tag{17}$$

6.3 Read-Write Memory Losses

When the Read-Write memory is used, we use two extra loss functions. The first is a Memory Ranking Loss J_{MR} as done in (Kaiser and Nachum, 2017), which learns based on whether a query with the persona embedding z returns nearest neighbors with the correct trope. The second is a Memory Classification Loss J_{MCE} that uses the values returned by the memory to predict the trope. The full details for both are found in Supplementary Section A.

6.4 Overall Loss

We combine the above losses through:

$$\begin{gathered} J = \beta_{CE} \cdot J_{CE} \\ + \beta_T \cdot \hat{J}_T \\ + \beta_{MR} \cdot J_{MR} + \beta_{MCE} \cdot J_{MCE} \end{gathered} \tag{18}$$

$$\hat{J}_T = \begin{cases} J_{MT} & \text{if memory module is used} \\ J_T & \text{otherwise.} \end{cases}$$

where $\beta = [\beta_{CE}, \beta_{MCE}, , \beta_T, \beta_{MR}]$ are learnable weights such that $\sum_i \beta_i = 1$. Depending on which variant of the model is being used, the list β is modified to contain only relevant losses. For example, when the Knowledge-Store memory is used, we set $\beta_{MR} = \beta_{MCE} = 0$ and β is modified to $\beta = [\beta_{CE}, \beta_T]$. We discuss different variants of our model in the next section.

7 Experiments

We experimented with combinations of our various modules and losses. The experimental results and ablation studies are described in the following sections, and the experimental details are described in Supplementary Section B. The different model permutation names in Table 2, e.g. "attn_3_tropetrip_ks-mem_ndialog16", are defined as follows:

- *baseline vs attn*: The 'baseline' model uses only one dialogue snippet S to predict the trope, i.e. $N_{diag} = 1$. Hence, the inter-snippet encoder is not used. The 'attn' model operates on N_{diag} dialogue snippets using the inter-snippet encoder to assign an attention score for each snippet S_i.
- *char vs. 3*: To measure the importance of context and other characters' lines, we have two variants – 'char' uses only the character's lines, while '3' uses the character's lines, other character's lines, and all context lines. Formally, in 'char' mode, we set γ^s_E and γ^s_O to 0 in Equation 1. In 'attn' mode, $(\gamma^s_E, \gamma^s_O, \gamma^s_D)$ are learned by the model.
- *tropetrip*: The presence of 'tropetrip' indicates that the triplet loss on the trope descriptions was used. If '-500' is appended to 'tropetrip', then the 4800-dimensional skip embeddings representing the descriptions in Equations 15 and 17 are projected to 500 dimensions using a fully connected layer.
- *ks-mem vs. rw-mem*: 'ks-mem' refers to the Knowledge-Store memory, and 'rw-mem' refers to the Read-Write memory.
- *ndialog*: The number of dialogue snippets N_{diag} used as input for the attention models. Any attention model without the explicit N_{diag} listed uses $N_{diag} = 8$.

7.1 Ablation Results

Baseline vs. Attention Model. The attention model shows a large improvement over the baseline models. This matches our intuition that not every quote is strongly indicative of character trope. Some may be largely expository or 'chit-chat' pieces of dialogue. Example attention scores are shown in Section 7.2.

Model	Accuracy	Precision	Recall	F1
baseline_char	0.286	0.538	0.286	0.339
baseline_3	0.287	0.552	0.288	0.349
attn_char	**0.630**	0.586	0.628	0.600
attn_3	0.630	**0.590**	**0.630**	**0.603**
attn_3_tropetrip	0.615	0.566	0.615	0.583
attn_3_tropetrip-500	**0.644**	**0.601**	**0.642**	**0.615**
attn_3_ks-mem	**0.678**	**0.648**	**0.676**	**0.657**
attn_3_rw-mem	0.635	0.598	0.635	0.611
attn_3_tropetrip_ks-mem	**0.663**	**0.628**	**0.662**	**0.639**
attn_3_tropetrip-500_ks-mem	0.654	0.618	0.652	0.629
attn_3_tropetrip_rw-mem	0.649	0.602	0.648	0.617
attn_3_tropetrip-500_rw-mem	0.644	0.608	0.644	0.619
attn_3_tropetrip-500_ks-mem_ndialog16	0.740	0.707	0.741	0.718
attn_3_tropetrip-500_ks-mem_ndialog32	**0.750**	**0.750**	**0.750**	**0.750**
attn_3_tropetrip_ks-mem_ndialog16	0.740	0.722	0.741	0.728
attn_3_tropetrip_ks-mem_ndialog32	0.731	0.712	0.731	0.718

Table 2: Experimental results. Details and analysis are given in Section 7.1. The best performing results in each block are bolded. The first block examines the baseline model vs. the attention model, as well as use of different inputs. The second block uses the triplet loss, and the third block uses our memory modules. The fourth block combines the triplet loss and memory module, which the fifth block extends to larger N_{diag}.

Rumors of my death have been greatly exaggerated.

What does it matter? Nothing's real down there. Our life is here.

I don't give a shit who he's connected to. … I want you vacate this guy off the premises, and I want you to exit him off his feet and use his head to open the fucking door.

I'm the man.

I had to see you. Stacy, I swear... I- I don't blame you for hating me. Or for wanting to break up. I just- Let me explain. About my family. I ju- I didn't want you to know. See, my dad's a drunk. Alright? … Cause when you love somebody, you never give up on them.

Not so bad... Oh! 'Ello, beastie.

I heard things.

Fuck! Fuck! That's it, I'm screwed. It's over.

Figure 4: Attention scores for a batch of dialogues for the "byronic hero" trope [3].

Though our experiments showed marginal improvement between using the 'char' data and the '3' data, we found that using all 3 inputs had greater performance for models with the triplet loss and read-only memory. This is likely because the others' lines and context capture more of the social dynamics and situations that are described in the trope descriptions. Subsequent results are shown only for the 'attn_3' models.

Trope Description Triplet Loss. Adding the trope description loss alone provided relatively small gains in performance, though we see greater gains when combined with memory. While both use the descriptions, perhaps the Knowledge Store memory matches an embedding against all the tropes, whereas the trope triplet loss is only provided information from one positive and one negative example.

Memory Modules. The Knowledge-Store memory in particular was helpful. Initialized with the trope descriptions, this memory can 'sharpen' queries toward one of the tropes. The Read-Write memory had smaller gains in performance. It may be that more data is required to take advantage of the write capabilities.

Combined Trope Description Triplet Loss and Memory Modules. Using the triplet loss with memory modules led to greater performance when compared to the $attn\_3$ model, but the performance sits around the use of either triplet only or memory only. However, when we increase the N_{diag} to 16 or 32, we find a jump in performance. This is likely the case because the model has both increased learning capacity and a larger sample of data at every batch, which means at least some of the N_{diag} quotes should be informative about the trope.

7.2 Attention Scores

Because the inter-snippet encoder provides such a large gain in performance compared to the baseline model, we provide an example illustrating the weights placed on a batch of N_{diag} snippets. Figure 4 shows the attention scores for the character's lines in the "byronic hero" trope. Matching what we might expect for an antihero personality, we find the top weighted line to be full of confidence and heroic bluster, while the middle lines

---

[3] TVtropes.org defines a byronic hero as "Sometimes an Anti-Hero, others an Anti-Villain, or even Just a Villain, Byronic heroes are charismatic characters with strong passions and ideals, but who are nonetheless deeply flawed individuals...."

hint at the characters' personal turmoil. We also find the lowly weighted sixth and seventh lines to be largely uninformative (e.g. "I heard things."), and the last line to be perhaps too pessimistic and negative for a hero, even a byronic one.

7.3 Purity scores of character clusters

Finally, we measure our ability to recover the trope 'clusters' (with one trope being a cluster of its characters) with our embeddings through the purity score used in (Bamman et al., 2014). Equation 19 measures the amount of overlap between two clusterings, where N is the total number of characters, g_i is the i-th ground truth cluster, and c_j is the j-th predicted cluster.

$$\text{Purity} = \frac{1}{N} \sum_i \max_j |g_i \cap c_j| \tag{19}$$

We use a simple agglomerative clustering method on our embeddings with a parameter k for the number of clusters. The methods in (Bamman et al., 2014) contain a similar hyper-parameter for the number of persona clusters. We note that the metrics are not completely comparable because not every character in the original dataset was found on IMDB. The results are shown in Table 3. It might be expected that our model perform better because we use the character tropes themselves as training data. However, dialogue may be noisier than the movie summary data; their better performing Persona Regression (PR) model also uses useful metadata features such as the movie genre and character gender. We simply note that our scores are comparable or higher.

k	PR	DP	AMN
25	42.9	39.63	**48.4**
50	36.5	31.0	**48.1**
100	30.3	24.4	**45.2**

Table 3: Cluster purity scores. k is the number of clusters, PR and DP are the Persona Regression and Dirichlet Persona models from (Bamman et al., 2014), and AMN is our attention memory network.

8 Application: Narrative Analysis

We collected IMDB quotes for the top 250 movies on IMDB. For every character, we calculated a character embedding by taking the average embedding produced by passing all the dialogues through our model. We then calculated movie embeddings by taking the weighted sum of all the character embeddings in the movie, with the weight as the percentage of quotes they had in the movie. By computing distances between pairs of character or movie embeddings, we could potentially unearth notable similarities. We note some of the interesting clusters below.

8.1 Clustering Characters

- Grumpy old men: Carl Fredricksen (Up); Walk Kowalski (Gran Torino)
- Shady tricksters, crooks, well versed in deceit: Ugarte (Casablanca); Eames (Inception)
- Intrepid heroes, adventurers: Indiana Jones (Indiana Jones and the Last Crusade); Nemo (Finding Nemo); Murph (Interstellar)

8.2 Clustering Movies

- Epics, historical tales: Amadeus, Ben-Hur
- Tortured individuals, dark, violent: Donnie Darko, Taxi Driver, Inception, The Prestige
- Gangsters, excess: Scarface, Goodfellas, Reservoir Dogs, The Departed, Wolf of Wall Street

9 Conclusion

We used the character trope classification task as a test bed for learning personas from dialogue. Our experiments demonstrate that the use of a multi-level attention mechanism greatly outperforms a baseline GRU model. We were also able to leverage prior knowledge in the form of textual descriptions of the trope. In particular, using these descriptions to initialize our Knowledge-Store memory helped improved performance. Because we use short text and can leverage domain knowledge, we believe future work could use our models for applications such as personalized dialogue systems.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

David Bamman, Brendan O'Connor, and Noah A Smith. 2014. Learning latent personas of film characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, page 352.

Murray R Barrick and Michael K Mount. 1993. Autonomy as a moderator of the relationships between the big five personality dimensions and job performance. *Journal of applied Psychology*, 78(1):111.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Lucie Flekova and Iryna Gurevych. 2015. Personality profiling of fictional characters using sense-level links between lexical resources. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1805–1816.

Jennifer Golbeck, Cristina Robles, Michon Edmondson, and Karen Turner. 2011. Predicting personality from twitter. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 149–156. IEEE.

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

Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer.

John T Jost, Tessa V West, and Samuel D Gosling. 2009. Personality and ideology as determinants of candidate preferences and obama conversion in the 2008 us presidential election. *Du Bois Review: Social Science Research on Race*, 6(1):103–124.

Łukasz Kaiser and Ofir Nachum. 2017. Aurko roy, and samy bengio. *Learning to remember rare events. arXiv preprint*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

François Mairesse and Marilyn Walker. 2007. Personage: Personality generation for dialogue. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 496–503.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Clifford Nass and Kwan Min Lee. 2000. Does computer-generated speech manifest personality? an experimental test of similarity-attraction. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 329–336. ACM.

Maarten Selfhout, William Burk, Susan Branje, Jaap Denissen, Marcel Van Aken, and Wim Meeus. 2010. Emerging late adolescent friendship networks and big five personality traits: A social network approach. *Journal of personality*, 78(2):509–538.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Phillip R Shaver and Kelly A Brennan. 1992. Attachment styles and the" big five" personality traits: Their connections with each other and with romantic relationship outcomes. *Personality and Social Psychology Bulletin*, 18(5):536–545.

Shashank Srivastava, Snigdha Chaturvedi, and Tom M Mitchell. 2016. Inferring interpersonal relations in narrative summaries. In *AAAI*, pages 2807–2813.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ernest C Tupes and Raymond E Christal. 1992. Recurrent personality factors based on trait ratings. *Journal of personality*, 60(2):225–251.

Josep Valls-Vargas, Jichen Zhu, and Santiago Ontanón. 2014. Toward automatic role identification in unannotated folk tales. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Marilyn A Walker, Janet E Cahn, and Stephen J Whittaker. 1997. Improvising linguistic style: Social and affective bases for agent personality. In *Proceedings of the first international conference on Autonomous agents*, pages 96–105. ACM.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

Grounding language acquisition by training semantic parsers using captioned videos

Candace Ross
CSAIL, MIT
`ccross@mit.edu`

Andrei Barbu
CSAIL, MIT
`abarbu@mit.edu`

Yevgeni Berzak
BCS, MIT
`berzak@mit.edu`

Battushig Myanganbayar
CSAIL, MIT
`btushig@mit.edu`

Boris Katz
CSAIL, MIT
`boris@mit.edu`

Abstract

We develop a semantic parser that is trained in a grounded setting using pairs of videos captioned with sentences. This setting is both data-efficient, requiring little annotation, and similar to the experience of children where they observe their environment and listen to speakers. The semantic parser recovers the meaning of English sentences despite not having access to any annotated sentences. It does so despite the ambiguity inherent in vision where a sentence may refer to any combination of objects, object properties, relations or actions taken by any agent in a video. For this task, we collected a new dataset for grounded language acquisition. Learning a grounded semantic parser — turning sentences into logical forms using captioned videos — can significantly expand the range of data that parsers can be trained on, lower the effort of training a semantic parser, and ultimately lead to a better understanding of child language acquisition.

1 Introduction

Children learn language from observations that are very different in nature from what parsers are trained on today. Most of the time, rather than receiving direct feedback such as annotated sentences or answers to direct questions, children observe and occasionally interact with their environment. They must use these observations to learn the structure of the speaker's language despite never seeing that structure overtly. This weak and indirect supervision where most of the information is obtained through passive observation poses a difficult disambiguation problem for learners: how do you know what the speaker is referring to in the environment, i.e., what does the speaker mean? Speakers can refer to actions, objects, the properties of actions and objects, relations between those actions and objects, as well as other features in the environment and generally do so by combining multiple features into complex sentences. Moreover, speakers need not refer to the most visually salient parts of a visual scene. Here, we induce a semantic parser by simultaneously resolving visual ambiguities and grounding the semantics of language using a corpus of sentences paired with videos without other annotations.

The woman walks by the table with a yellow cup.
λxyz.woman x, walk x, near x y, table y,
hold x z, yellow z, cup z

Figure 1: We develop a semantic parser trained on video-sentence pairs, *without parses*. At inference time a sentence, *without a video*, is presented and a logical form is produced.

The goal of semantic parsing is to convert a natural-language sentence into a representation that encodes its meaning. The parser takes sentences as input and produces these representations – a lambda-calculus expression in our case – that can be used for a variety of tasks such as querying databases, understanding references in images and videos, and answering questions. To train the parser presented here we collected a video dataset, balanced such that the raw statistics of the co-occurrences of objects and events are not informative, and asked annotators on Mechanical Turk to produce sentences that are true of those videos. The parser is presented with pairs of short clips and sentences. It hypothesizes potential meanings for those sentences as lambda-calculus expressions. Each hypothesized expression serves as input for a modular vision system that constructs a specific detector for that lambda-calculus expression and determines the likelihood of the parse being true of the video. The likelihood of the parse with respect to the video is used as supervision for the parser. To

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2647–2656
Brussels, Belgium, October 31 - November 4, 2018.

test the parser, we annotated each sentence with its ground-truth semantic parse, but this information is not available at training time.

This process introduces ambiguity. For example, Figure 1 shows a frame from a video annotated with the sentence *"The woman walks by the table with a yellow cup."*, yet the parse, $\lambda x.\ \text{object}(x)$, corresponding to a sentence like *"There exists an object."*, is also true of that video. For a single video there exists an infinite number of true parses that have high likelihood with respect to the vision system because they are indeed indicative of something that is occurring in the video. We demonstrate how to construct a semantic parser that resolves this ambiguity and acquires language from captioned videos by learning to tune the amount of polysemy in the induced lexicon.

This work makes several contributions: We show how to construct a semantic parser that learns language in a setting closer to that of children. We demonstrate how to jointly resolve linguistic and visual ambiguities at training time in a way that can be adapted to other semantic parsing approaches. We demonstrate how such an approach can be used to augment data where a small number of directly annotated sentences can be combined with a large number of videos paired with sentences in order to improve performance. We release a dataset systematically constructed and annotated on Mechanical Turk for joint visual and linguistic learning tasks.

2 Prior work

Learning to understand language in a multimodal environment is a well-developed task. For example, visual question answering (VQA) datasets have led to a number of systems capable of answering complex questions about scenes (Antol et al., 2015). The goal of our work is not to produce answers for any one set of questions, although it is possible to do so from our results; it is instead to learn to predict the structure of the sentences and their meaning. This is a more general and difficult problem, in particular because at test time we do not receive any visual input, only the sentence. The resulting approach is reusable, generic and more similar to the kind of general-purpose linguistic knowledge that humans have. For example, one could use it to guide robotic actions. Al-Omari et al. (2017) acquire a grammar for a fragment of English and Arabic from videos paired with sentences. They learn a small number of grammar rules for a language restricted to robotic commands. Learning occurs mostly in simulation and with little visual ambiguity, and the resulting model is not a parser but a means of associating n-grams with visual concepts.

Siddharth et al. (2014) and Yu et al. (2015) acquire the meaning of a lexicon from videos paired with sentences but assume a fully-trained parser. Matuszek et al. (2012) similarly present a model to learn the meanings and referents of words restricted to attributes and static scenes. Hermann et al. (2017) extend these notions to train agents that learn to carry out instructions in simulated environments without the need for a parser, but do so using simple adjective-noun-relation utterances. Kollar et al. (2013) learn to parse similar utterances in an interactive setting. Wang et al. (2016) create a language game to learn a parser but do not incorporate visual ambiguity or fallible perception.

Berant et al. (2013) describe semantic parsing with execution by annotating answers to database queries. This learning mechanism provides the same results as the one described here: a parser produces the meanings of sentences at inference time without requiring the database, or in our case a video. Databases have far less ambiguity than videos; there is not a temporal aspect to their contents and there is not a notion of unreliable perception. Berant and Liang (2014) learn to parse sentences from paraphrases; one might consider the work here as concerned with visual and not just linguistic paraphrases. Artzi and Zettlemoyer (2013) consider a setting where a validation function involves the dynamic actions of a simulated robot while sentences describe its actions.

3 Task

Given a dataset of captioned videos, D, we train the parameters and lexicon, θ and Λ, of a semantic parser. At training time, we perform gradient descent over the parameters θ and employ *GENLEX* (Zettlemoyer and Collins, 2005) to augment the lexicon Λ. The objective function of the semantic parser is written in terms of a visual-linguistic compatibility between a hypothesized parse p and video v. This compatibility computes the likelihood of the parse being true of the video, $P(v|p)$. At test time, we take as input a sentence without an associated video and produce a semantic parse. We could in principle also take as input the video and produce a targeted parse for that visual scenario. This is a problem similar to that considered by Berzak et al. (2015), but we do not do so here.

We create a CCG-based (Combinatory Categorical Grammar; Steedman (1996)) semantic parser capable of being trained in this setting. To do so, we adapt the objective function, training procedure, and feature set to this new scenario. The visual-linguistic compatibility function is similar to the Sentence Tracker developed in Siddharth et al. (2014) and Yu et al. (2015). Given a parse, the Sentence Tracker produces a targeted detector that determines if the parse is true of a video, which provides a weak supervision signal for the parser.

Parses are represented as lambda-calculus expressions consisting of a set of binders and a conjunction of literal expressions referring to those binders. The domain of the variables are the potential object locations, or object tracks, in the videos. For example, in the parse presented in Figure 1, three potential object track slots are available, represented by the binders x, y, and z. Because of perceptual ambiguities and the large number of possible referents in any one video, we do not explicitly enumerate the space of object tracks. Instead, we rely on a joint-inference process between the parser and the Sentence Tracker. Intuitively, each literal expression of the parse asserts a constraint; for example, if an expression conveys that one object is approaching another, the Sentence Tracker will search the space of object tracks and attempt to satisfy these constraints. In Figure 1, for instance, there is a constraint that for whichever objects are bound to x and z, x must be near y, x must be walking, x must be a person, etc.

4 Model

We develop an approach that combines a semantic parser with a vision system at training time, but does not require the vision system at test time.

4.1 Semantic Parsing

We adopt a semantic parsing framework similar to that of Artzi and Zettlemoyer (2013), although the general approach of using vision as weak supervision for semantic parsing generalizes to other parsers. CCG-based parsing employs a small number of fixed unary and binary derivation rules (Steedman, 2000) while learning a lexicon. In CCG-based parsing, a parser takes as input a sequence of tokens and a lexicon that maps tokens to potential syntactic types and derives parse trees by creating and ranking multiple hypotheses that combine those types together. The syntactic types are richer than other approaches and include forward and backward function application (the forward and backward slash) in addition to the standard syntactic categories. Each derivation has a current syntactic type that is the result of the application of a sequence of rules. To create a derivation, at each step the parser applies each rule to either an individual subderivation or to a pair of subderivations. This process produces multiple hypotheses. Parsing rules are generic, polymorphic, and language-neutral and include concepts like function application and type raising (Carpenter, 1997). The parser accepts a derivation when the tree reaches a single node. We refer to the single node of the parse tree as the logical form. Figure 2 shows a parse starting with tokens and their syntactic types along with each rule being applied.

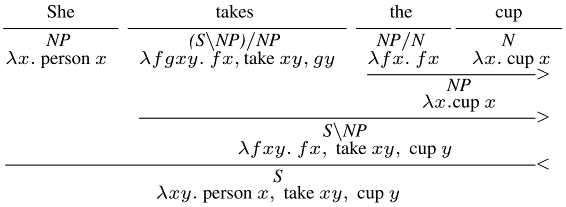

Figure 2: A simple sentence parsed into a lambda-calculus expression using a CCG-based grammar. The parse is determined by the lexicon that associates tokens with syntactic and semantic types as well as the order of function applications. Here, we acquire this lexicon and a means to score derivations.

Semantic parsing with CCGs extends this framework to simultaneously derive a logical form while performing syntactic parsing. Each syntactic rule includes a simple semantic component that manipulates the logical form of its arguments. For example, the forward application rule reduces the syntactic type by applying the syntactic type of the right argument to that of the left, while at the same time performing a lambda-calculus reduction of the semantic types of those same arguments. Concretely, consider a case from Figure 2 where a determiner is attached to a noun, *the cup*. The tokens the and cup are hypothesized to have syntactic types NP/N and N (a function returning NP given an argument on the right side and a noun) and semantic type $\lambda fx.fx$ and $\lambda x.\text{cup}(x)$ (the identity function and a function that adds a cup constraint). These two derivations can be reduced by forward application, denoted by $>$. Both the syntactic and semantic types are applied and reduced, which means the semantics helps guide the syntax. Derivations that produce illegal operations, such as applying an argument to a constant, are forbidden.

Following Zettlemoyer and Collins (2005) and Curran et al. (2007), we adopt a weighted linear semantic parser. For each sentence paired with its hypothesized derivation, this approach computes a feature vector ϕ and a parameter vector θ. Given a sentence s, a parse p, a lexicon Λ, the set of all possible parses for that sentence with that lexicon, $P(s, \Lambda)$, and an n-dimensional feature vector computed for that sentence and parse, $\phi(s, p)$, the parser optimizes

$$\underset{p \in P}{\operatorname{argmax}} \; \theta \cdot \phi(s, p) \tag{1}$$

to find the best parse p^*. Using a fixed-width beam search, the parser enumerates derivations by choosing a potential syntactic and semantic type for each token from the lexicon and choosing a set of derivation rules to apply. For the i-th training sample d_i, consisting of a sentence d_i^s and a video d_i^v in dataset D and the feature function, the parser finds margin-violating positive, E^+, and negative, E^-, parses, and then uses

$$\theta + \frac{1}{|E_i^+|} \sum_{e \in E_i^+} \phi_i(e, d_i^v) - \frac{1}{|E_i^-|} \sum_{e \in E_i^-} \phi_i(e, d_i^v) \tag{2}$$

to update the parameter θ. After each sweep through the dataset, the lexicon Λ is augmented using the modified *GENLEX* from Artzi and Zettlemoyer (2013), which does not require the ground-truth logical form. At no point is the logical form needed for updating the lexicon or parameters; we rely instead on a visual validation function to compute the margin-violating examples.

Rather than attempting to learn a fixed lexicon that directly maps tokens to semantic and syntactic parses, we use a factored lexicon like that of Kwiatkowski et al. (2011). This represents tokens and any associated constants separately from potential syntactic and semantic types. For example, the token *chair* is associated with a single constant `chair`; *chair* $\vdash$ [`chair`]. In addition to the token-constants pairs, there exists a list of pairs of syntactic and semantic types along with placeholders for constants; in the case for *chair*, a useful type might be $\lambda v.[N : \lambda x.\texttt{placeholder}(x)]$. When parsing, each token is applied to a potential syntactic and semantic type and the derivation proceeds from there. The factored lexical entries allow for far greater reuse; the model learns a small number of constants that a word can imply separately from a small number of syntactic and semantic types for any word. The weighted linear CCG-based parser searches over potential lexical entries, applying the token to different syntactic and semantic types and over multiple hypotheses for which rule should be applied. At training time, in order to learn a reasonable lexicon and set of parameters, a supervision signal is required to validate candidates. We provide that supervision using the vision system described below.

4.2 Sentence Tracking

To score a video-parse pair, we employ a framework similar to that of Yu et al. (2015). This approach constructs a parse-specific model by extracting the number of participants in the scene described by a caption as well as the relationships and properties of those participants. It builds a graphical model where each participant is localized by an object tracker and each relationship is encoded by temporal models that express the properties of the trackers that those models refer to. The parser's output representation is chosen to make building the vision system possible. Each target logical form is a lambda expression with a set of binders, whose domain are objects, and a conjunction of constraints that refer to those binders. In essence, this notes which objects should be present in a scene and what static and changing properties and relationships those objects should have with respect to one another.

The Sentence Tracker creates one Viterbi-based tracker for each participant and, given a mapping from constraints to Hidden Markov models (HMMs), connects each tracker and each constraint together. Given a video v and a parse p, first a large number of object detections are computed for the video by using a low confidence threshold of an object detector. Trackers weave these bounding-box detections into high-scoring object tracks and use constraints to verify if the tracks have the desired properties and relations. Inference proceeds jointly between vision and the parse to allow the parse to focus the vision component on events and properties that might otherwise be missed.

Understanding the relationship between a sentence and a video requires finding the objects that the sentence refers to and determining if those objects follow the behavior implied by the sentence. We carry out a joint optimization that finds objects whose behavior follows certain rules. For clarity, the two steps are presented separately, while we find the global optimum for a linear combination of Equation (3) and Equation (4). Object trackers are a maximum-entropy Markov model with a per-frame score f, the likelihood that any one object detection is true, as well as a motion-coherence score g, the likelihood that the bounding boxes selected between frames refer to the same object instance. Given a parse p with L participants and a video v of length T, Equation (3) shows the optimization where J is a set of L candidate tracks ranging over every hypothesis from the object detector and b is a candidate object detection.

$$\max_{J} \sum_{l=1}^{L} \left(\sum_{t=1}^{T} f(b^{t}_{j^{t}_{l}}) + \sum_{t=1}^{T} g(b^{t-1}_{j^{t-1}_{l}}, b^{t}_{j^{t}_{l}}) \right) \tag{3}$$

Determining if an object track follows a set of behaviors implied by a sentence is done using a collection of HMMs. Each has a per-frame score h that observes one or more objects tracks, depending on the number of participants in the behavior being modeled, and a transition function a that determines the temporal sequence of the behavior. Given a parse p with C behaviors, also termed constraints, along with a video v of length T, Equation (4) shows the optimization where K is a set of states, one for each constraint, and γ is a linking function.

$$\max_{J,K} \sum_{c=1}^{C} \left(\sum_{t=1}^{T} h_c(b^{t-1}_{j^{t-1}_{\gamma^1_c}}, b^{t}_{j^{t}_{\gamma^2_c}}, k^t_c) + \sum_{t=1}^{T} a_c(k^{t-1}_c, k^t_c) \right) \tag{4}$$

The linking function is an indicator variable that encodes the structure of the logical form thereby filling in the correct trackers as arguments for the corresponding constraints. The exposition above presents a variant using binary constraints that is trivially generalized to n-ary constraints by extending γ and adding arguments to the appropriate constraint observation functions h_c. The domain of the optimization problem is the combination of all objects at all timesteps that the logical form can refer to as well as every state of each constraint. The Viterbi algorithm carries out this optimization in time linear in the length of the video and quadratic in the number of detections per frame. The result is a likelihood of the parse being true of a video. This is used to create the joint model that supervises the parser with vision. The tracker can also produce a time series of bounding boxes that make explicit the groundings of the sentences, though we do not use these directly here.

4.3 Joint Model

At training time, we jointly learn using both the semantic parser and the language-vision component. At test time, only the parser is used. Two parameters are learned, a set of weights θ and the lexicon Λ. For both the parser and the associated language-vision component, Λ is used to structure inference. To induce new lexical entries, we employ a variant of *GENLEX* (Artzi and Zettlemoyer, 2013) that takes as input a validation function — the compatibility between a parse and the video. This *GENLEX* uses an ontology of predicates, a validation function, and templates from the current lexicon to construct new syntactic and semantic forms. A ground-truth logical form is not required or used.

The joint model must learn these parameters despite three sources of noise. First, the vision-language component may simply fail to produce the correct likelihood because machine vision is far from perfect. Overcoming this requires large beam widths to avoid falling into local minima due to these errors.

Second, an infinite number of possibly-erroneous parses are true of a video. When children learn language, they face this same challenge as they do not have access to bounding boxes or to logical forms. The parse $\lambda x.person(x)$ as well as many other seemingly reasonable parses are true and cannot be distinguished from the ground-truth parse — which is not available — by the vision component. This is a far less constrained environment than other approaches to semantic parsing. It is easy to be misguided by a loss function that is often true when it should not be and thus create many special-purpose definitions of words that happen to fit the peculiarities of any video. This results in two different problems: assigning empty semantics to many words since the likelihood of a subset of a parse is always the same or higher than the whole parse and excessive polysemy where the meaning of a word is highly specific to some irrelevant feature in a video. We introduce two features to the parser that bias it against empty semantics and against excessive polysemy. Models of communication such as the Rational Speech Acts model (Frank and Goodman, 2012) predict that speakers will avoid inserting meaningless words. One feature counts the number of predicates mapped onto semantic forms which are empty that occur in each parse. The other feature attempts to prevent excessive polysemy by counting how many new semantic forms are introduced for existing tokens by the generated entries from each parse. As the parser becomes more capable of handling sentences in the training set, these features begin to bias it against adding empty semantics and new semantic forms.

Third, models in computer vision are computationally expensive while many evaluations of parse-video pairs are required to train a parser. To overcome this, we construct a provably-correct cache that keeps track of failing subexpressions. This is possible because of a feature of this particular vision-language scoring function: the score decreases monotonically with the number of constraints. With these improvements, the modified semantic parser employing vision-language-based validation learns to map sentences into semantic parses despite facing a challenging setting with few examples and much ambiguity.

5 Dataset

We collected and annotated a dataset of captioned videos with fully annotated semantic parses of the captions. The videos contain people carrying out one of 15 actions, such as picking things up and putting things down, with one of 20 objects spanning 10 different colors. We control for 11 spatial relations between objects and actors. Many videos depict multiple agents performing actions leading to additional ambiguity. Videos were filmed in multiple locations with multiple agents but care was taken to ensure that the background and agents are not informative of the events depicted.

On Mechanical Turk we asked participants to provide sentences that describe something about the video. We did not specify what participants should describe to avoid biasing them and to add richness to the dataset. This sometimes led to sentences that referred to properties of the video that are well beyond the capacities of the vision system, e.g., descriptions of an agent being lazy or references to the camera's movement. We removed such sentences. At training time, the parser receives captioned videos but no annotations about which objects those captions refer to. Each sentence was annotated with a ground-truth semantic form by two trained annotators using a set of 34 predicates. Each sentence was then reviewed and corrected by one other annotator.

To detect the objects in the videos, we used two off-the-shelf detectors, OpenPose (Cao et al., 2017) for person detection and YOLO version 3 (Redmon and Farhadi, 2018) for the remaining objects. In each case we significantly lowered the confidence threshold to avoid false negatives. Many objects in this dataset are small and are handled by humans, which leads to regular object detector failures that are only partially compensated for by lowering the detection threshold at the cost of a large number of false positives. We rely on the inference mechanism of the grounded parser to automatically eliminate these numerous false positives as candidates when grounding sentences due to their low likelihoods. False negatives are much more misleading and difficult to overcome than false positives. It is harder to read in where an unseen object might be than to eliminate a low-confidence detection.

In total, the dataset contains 1200 captions from 401 videos, which selected out of a larger body of sentences collected and pruned as described above. This is comparable to the size of other datasets used for semantic parsing such as two datasets from Tang and Mooney (2001) with 880 and 640 examples respectively and the navigation instruction dataset (Chen and Mooney, 2011) with 706 examples (containing 3236 single sentences). The sentences comprising our dataset contain 169 unique tokens with an average of 7.93 tokens per caption. There are an average of 2.31 objects per caption.

6 Evaluation

6.1 Experimental Setup

We adapted the Cornell SFP (Semantic Parsing Framework) developed by Artzi (2016) to jointly reason about sentences and videos. We selected 720 examples for training and used 120 examples for the validation set to fine-tune the model parameters. We used the remaining 360 examples for the test set. This split was fixed and used in all experiments below. No sentences or videos occurred in both the training and test sets. During training, each hypothesized parse for each sentence is marked as either correct or incorrect, using either direct supervision with the target parse or compatibility with the video, depending on the experiment.

We use beams of 80 for the CKY-parser and *GENLEX*. CCG-based semantic parsers are seeded with a small number of generic combinations of syntactic and semantic types. For example, Artzi (2016) seed with 141 lexical entries; we provide 98. *GENLEX* uses these entries along with an ontology to form new syntactic and semantic types.

Condition	Precision		Recall		F1	
Direct supervision	0.851	*0.946*	0.84	*0.933*	0.846	*0.939*
Noisy supervision (60%)	0.235	*0.423*	0.201	*0.362*	0.217	*0.390*
Shuffled labels (direct supervision)	0.147	*0.384*	0.122	*0.321*	0.136	*0.349*
Shuffled videos (weak supervision)	0.000	*0.106*	0.000	*0.103*	0.000	*0.104*
Object-only vision	0.051	*0.387*	0.042	*0.349*	0.046	*0.367*
Vision-language	0.223	*0.663*	0.183	*0.553*	0.201	*0.591*

Figure 3: Pairs of results for each condition. On the left, we show exact match results and on the right, in *italics*, results for the near miss metric. In the case of *direct supervision*, we train with the target parses. In the case of *noisy supervision*, a percentage of the time (60% here) the parser randomly accepts or rejects a parse. In the case of *shuffled labels*, the target logical forms are assigned to random sentences. For *shuffled videos* the sentences are assigned to random videos. The likelihood of any sentence being true of a random video is low. In the case of *object-only vision*, the vision system consists solely of an object detector discarding any other predicates. The full *vision-language* approach learns to parse a significant fraction of sentences, far outperforming the object-only approach, and usually being within one predicate of the correct answer.

6.2 Results

Figures 3 and 4 summarize the experiments and ablation studies performed. The metrics we use when reporting results are *exact* matches, where the predicted parses must perfectly match the target parses, and *near misses*, where a single predicate in the semantic parse is allowed to differ from the target. Experiments were averaged across 5 runs.

To establish chance-level performance, we trained the directly supervised approach on shuffled labels, assigning random correct parses to random sentences. This is more powerful than a simple chance-level performance calculation as the parser can still take advantage of any dataset biases. Even with the ability to exploit potential biases, performance is very low with F1 scores of 0.136 and 0.349 for the exact and near miss metrics. Both metrics pose a challenging learning problem.

As a baseline, we directly supervised the parser with the target logical forms. When doing so, it achieved high performance with F1 scores of 0.841 and 0.911 for the exact match and near miss cases. Figure 4 shows performance of direct supervision as a function of training set size.

We then added noise to the directly supervised parser. Doing so simulates the unreliable nature of vision and, to an extent, the ambiguities inherent in vision. Noise was introduced by modifying the compatibility function which determines if a parse is correct. A certain percentage of the time, that function returned true or false randomly when given a hypothesized logical form. With around 60% noise, performance was 0.22 and 0.39 F1 for the noisy and near miss cases. Figure 4 shows performance of the noisy baseline as a function of how much noise was introduced.

The fully grounded parser produced 0.2 and 0.6 F1 scores for the exact and near miss metrics. This is far beyond chance performance and corresponds to direct supervision with around 55% noise. There are a number of reasons for why performance is not perfect. First, the evaluation metrics cannot consider equivalences in meaning, just form. A hypothesized parse may carry the same meaning as the target logical form yet it will be considered incorrect. This is less of a problem with direct supervision where the preferences that annotators have for a particular way of encoding the meaning of a sentence can be learned. In the grounded case, this cannot be learned; visually equivalent parses are equally likely. Second, computer vision is unreliable, i.e., object detectors fail. We find that in many of our videos while person detection is fairly reliable, object detection is unreliable. Third, vision in the real world is very ambiguous. Predicates like *hold* are true in almost every interaction. This makes learning the meanings of words much more difficult resulting in the grounded parser often adding useless entries into the predicted logical forms or substituted one predicate for a similar one. The near miss metric shows that overall the parser learned reasonable logical forms. Figure 5 shows six examples from our dataset along with expected and predicted parses, both correct and incorrect.

To understand how much of the performance of the grounded parser comes from visual correlations, like the presence or absence of particular objects, as opposed to more complex and cognitively relevant spatio-temporal relations like actions, we ablated the parser. We removed all features other than objects. The resulting grounded parser accepts any hypothesized parse as long as the objects mentioned in that parse are present in the video. This led to a significant performance drop, near-chance level performance on the exact metric, F1 0.05, and nearly half the F1 score on the near miss metric, 0.37. Having a sophisticated vision system to infer about agents and interactions is crucial for learning.

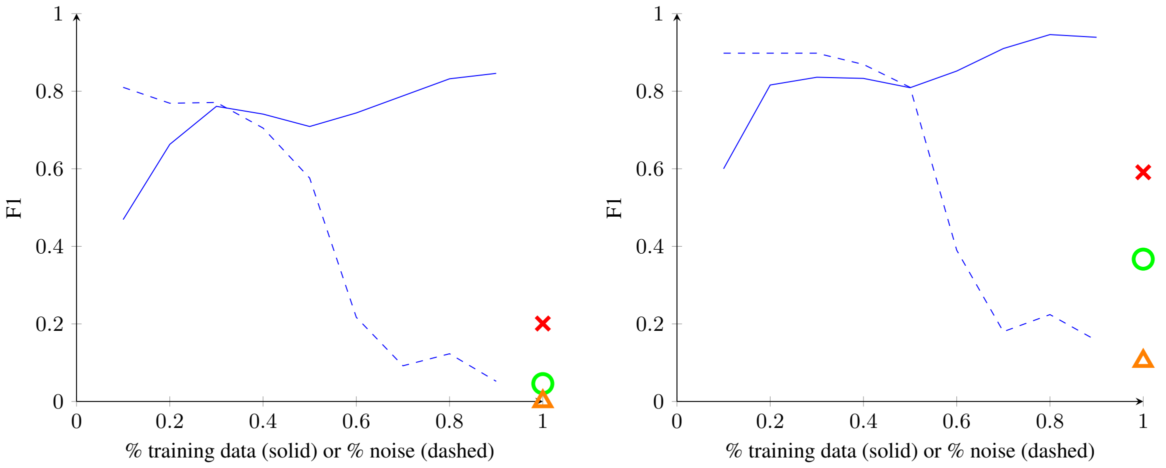

(a) Exact match

(b) Near miss

Figure 4: Results from training the grounded semantic parser. In blue, *direct supervision* as a function of the amount of training data. In dashed blue, *noisy supervision* uses the whole training set but accepts and rejects parses at random for a given fraction of the time. The red cross is the full vision system while the green o is the object detector ablation. The orange triangle represents *shuffled videos* and shows chance performance. While direct supervision outperforms vision-only supervision, the grounded parser closes the gap and operates like noisy direct supervision with roughly 55% noise.

7 Discussion

We present a semantic parser that learns the structure of language using weak supervision from vision. At test time, the model parses sentences without the need for visual input. Learning by passive observation in this way extends the capabilities of semantic parsers and points the way to a more cognitively plausible model of language acquisition. Several limits remain. Evaluating parses as correct or incorrect depending on a match to a human-annotated logical form is an overly strict criterion and is a problem that also plagues fully-supervised syntactic parsing (Berzak et al., 2016). Since two logical forms may express the same meaning, it is not yet clear what an effective evaluation metric is for these grounded scenarios. In addition, learning in such a passive scenario is hard as correlations between events, e.g., every *pick up* event involves a *touch* event, are very difficult to disentangle.

An interesting source of error in the experimental results comes from visual ambiguities. At the level of relative motions of labeled bounding boxes, the analysis performed by the language-vision system we employed here has difficulty distinguishing certain parts of actions. For example, carrying a shirt and wearing a shirt appear very similar to one another as they are actions that mostly involve moving alongside a person detection. Moreover, since every agent is wearing a shirt it becomes more difficult to learn to distinguish the two actions using positive evidence alone, i.e., a maximum likelihood approach. A more robust vision system, perhaps including object segmentations, person pose, and weak negative evidence for the occurrence of actions, would likely significantly improve the results presented.

In the future, we intend to add a generative model along with a physical simulation allowing the learner to imagine scenarios where a predicate might not hold. This would help mitigate systematic correlations between sentences and videos. The sentences selected here were all chosen such that they are true of the video being shown, yet much of what people discuss is ungrounded, or at least not grounded in the current visual scene. We intend to combine the weakly supervised parser with an unsupervised parser and learn to determine whether a sentence should be grounded visually during training. We hope this work will find applications in robotics where learning to adapt to the specific language of a user while engaging with them is of utmost importance when deploying robots in users' homes.

Acknowledgments

This work was supported, in part, by the Center for Brains, Minds and Machines, NSF STC award 1231216, NSF Graduate Research Fellowship, Ford Foundation Graduate Research Fellowship, Toyota Research Institute, AFRL contract No. FA8750-15-C-0010, and MIT-IBM Brain-Inspired Multimedia Comprehension project.

(i)

Annotated sentence: *The woman is picking up an apple.*
Ground-truth parse: λxy.woman x, pick_up x y, apple y
Predicted parse: λxy.woman x, pick_up x y, apple y

(ii)

Annotated sentence: *A man walks across the hall holding a chair.*
Ground-truth parse: λxyz.person x, walk x, across x y, hallway y, hold x z chair z
Predicted parse: λxyz.person x, from x y, person y, hold x z chair z

(iii)

Annotated sentence: *A man is walking toward a chair.*
Ground-truth parse: λxy.person x, walk x, toward x y, chair y
Predicted parse: λxy.person x, walk x, toward x y, chair y

(v)

Annotated sentence: *She places the toy car down on the table.*
Ground-truth parse: λxyz.person x, put_down x y, toy y, car y, on y z table z
Predicted parse: λxyz.person x, in x y, toy y, car y, on y z table z

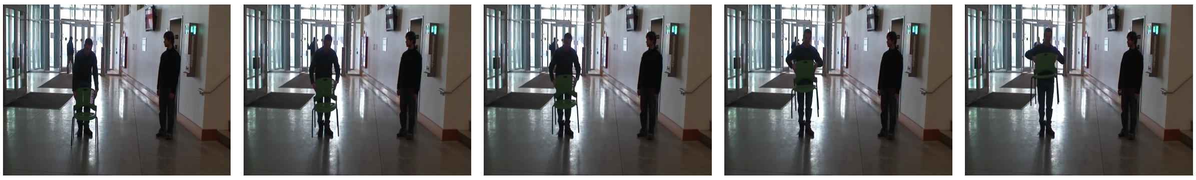

(iv)

Annotated sentence: *A man is lifting the chair.*
Ground-truth parse: λxy.person x, pick_up x y, chair y
Predicted parse: λxy.person x, pick_up x y, chair y

(vi)

Annotated sentence: *A woman reaches for a book on the table.*
Ground-truth parse: λxyz.person x, pick_up x y, book y, on y z table z
Predicted parse: λxyz.person x, stand x, in x y, book y, on y z table z

Figure 5: Six examples of frames from videos in the dataset along with target and predicted logical forms showing both successes and failures. Failures are highlighted in red. Note how incorrect parses are usually similar to the correct semantic forms. The intended meaning is often preserved even in these cases.

References

Muhannad Al-Omari, Paul Duckworth, David C Hogg, and Anthony G Cohn. 2017. Natural language acquisition and grounding for embodied robotic systems. In *AAAI*. pages 4349–4356.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2425–2433.

Yoav Artzi. 2016. Cornell SPF: Cornell semantic parsing framework.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics* pages 49–62.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. *Empirical Methods for Natural Language Processing* .

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Annual Meeting of the Association for Computational Linguistics*.

Yevgeni Berzak, Andrei Barbu, Daniel Harari, and Boris Katz. 2015. Do You See What I Mean ? Visual Resolution of Linguistic Ambiguities. *Conference on Empirical Methods on Natural Language Processing* (September):1477–1487.

Yevgeni Berzak, Yan Huang, Andrei Barbu, Anna Korhonen, and Boris Katz. 2016. Anchoring and agreement in syntactic annotations. *Conference on Empirical Methods in Natural Language Processing* .

Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*.

Bob Carpenter. 1997. *Type-logical semantics*. MIT press.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions fro mobservations. In *AAAI Conference on Artificial Intelligence*. San Francisco, CA, USA.

James R Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, pages 33–36.

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science* 336(6084):998–998.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojtek Czarnecki, Max Jaderberg, Denis Teplyashin, et al. 2017. Grounded language learning in a simulated 3D world. *arXiv preprint arXiv:1706.06551* .

Thomas Kollar, Jayant Krishnamurthy, and Grant P Strimel. 2013. Toward interactive grounded language acqusition. In *Robotics: Science and systems*. volume 1, pages 721–732.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical Generalization in CCG Grammar Induction for Semantic Parsing. *Conference on Empirical Methods in Natural Language Processing* pages 1512–1523.

Cynthia Matuszek, Nicholas Fitzgerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning*. ACM, pages 1671–1678.

Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv* .

Narayanaswamy Siddharth, Andrei Barbu, and Jeffrey Mark Siskind. 2014. Seeing what you're told: Sentence-guided activity recognition in video. In *The IEEE Conference on Computer Vision and Pattern Recognition*.

Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*. pages 466–477.

Sida I Wang, Percy Liang, and Christopher D Manning. 2016. Learning language games through interaction. *Meeting of the Association for Computational Linguistics* .

Haonan Yu, N. Siddharth, Andrei Barbu, and Jeffrey Mark Siskind. 2015. A compositional framework for grounding language inference, generation, and acquisition in video. *Journal of Artificial Intelligence Research* .

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*. AUAI Press, Arlington, Virginia, pages 658–666.

Translating Navigation Instructions in Natural Language to a High-Level Plan for Behavioral Robot Navigation

Xiaoxue Zang[1]*, **Ashwini Pokle**[1]*, **Marynel Vázquez**[2], **Kevin Chen**[1],
Juan Carlos Niebles[1], **Alvaro Soto**[3], **Silvio Savarese**[1]
[1] Stanford University, [2] Yale University, [3] P. Universidad Católica de Chile
[1]{xzang, ashwinipokle, kchen92, jniebles, ssilvio}@stanford.edu,
[2]marynel.vazquez@yale.edu, [3]asoto@ing.puc.cl

Abstract

We propose an end-to-end deep learning model for translating free-form natural language instructions to a high-level plan for behavioral robot navigation. The proposed model uses attention mechanisms to connect information from user instructions with a topological representation of the environment. To evaluate this model, we collected a new dataset for the translation problem containing 11,051 pairs of user instructions and navigation plans. Our results show that the proposed model outperforms baseline approaches on the new dataset. Overall, our work suggests that a topological map of the environment can serve as a relevant knowledge base for translating natural language instructions into a sequence of navigation behaviors.

1 Introduction

Enabling robots to follow navigation instructions in natural language can facilitate human-robot interaction across a variety of applications. For instance, within the service robotics domain, robots can follow navigation instructions to help with mobile manipulation (Tellex et al., 2011) and delivery tasks (Veloso et al., 2015).

Interpreting navigation instructions in natural language is difficult due to the high variability in the way people describe routes (Chen and Mooney, 2011). For example, there are a variety of ways to describe the route in Fig. 1(a):

- *"Exit the room, turn right, follow the corridor until you pass a vase on your left, and enter the next room on your left"*; or
- *"Turn right after you exit the room, and enter the room on the left right before the end of the corridor"*; or
- *"Advance forward to the right after going out of the door. Enter the room which is in the middle of two vases on your left."*

Each fragment of a sentence within these instructions can be mapped to one or more than one navigation behaviors. For instance, assume that a robot counts with a number of primitive, navigation behaviors, such as *"enter the room on the left (or on right)"* , *"follow the corridor"*, *"cross the intersection"*, etc. Then, the fragment *"advance forward"* in a navigation instruction could be interpreted as a *"follow the corridor"* behavior, or as a sequence of *"follow the corridor"* interspersed with *"cross the intersection"* behaviors depending on the topology of the environment. Resolving such ambiguities often requires reasoning about "common-sense" concepts, as well as interpreting spatial information and landmarks, e.g., in sentences such as *"the room on the left right before the end of the corridor"* and *"the room which is in the middle of two vases"*.

In this work, we pose the problem of interpreting navigation instructions as finding a mapping (or grounding) of the commands into an executable navigation plan. While the plan is typically modeled as a formal specification of low-level motions (Chen and Mooney, 2011) or a grammar (Artzi and Zettlemoyer, 2013; Matuszek et al., 2010), we focus specifically on translating instructions to a high-level navigation plan based on a topological representation of the environment. This representation is a *behavioral navigation graph*, as recently proposed by (Sepúlveda et al., 2018), designed to take advantage of the semantic structure typical of human environments. The nodes of the graph correspond to semantically meaningful locations for the navigation task, such as kitchens or entrances to rooms in corridors. The edges are parameterized, visuo-motor behaviors that allow a robot to navigate between neighboring nodes, as illustrated in Fig. 1(b). Under this framework, complex navigation routes can be achieved by sequencing behaviors without an explicit metric representation of the world.

---

[*] Both authors contributed equally to this work.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2657–2666
Brussels, Belgium, October 31 - November 4, 2018.

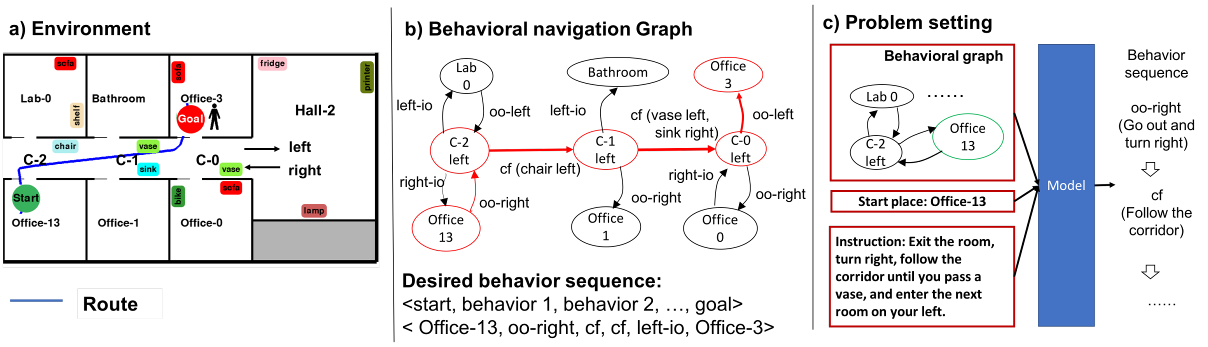

Figure 1: Map of an environment (a), its (partial) behavioral navigation graph (b), and the problem setting of interest (c). The red part of (b) corresponds to the representation of the route highlighted in blue in (a). The codes "oo-left", "oo-right", "cf", "left-io", and "right-io" correspond to the behaviors "go out and turn left", "go out and turn right", "follow the corridor", "enter the room on left", and "enter office on right", respectively.

We formulate the problem of following instructions under the framework of (Sepúlveda et al., 2018) as finding a path in the behavioral navigation graph that follows the desired route, given a known starting location. The edges (behaviors) along this path serve to reach the – sometimes implicit – destination requested by the user. As in (Zang et al., 2018), our focus is on the problem of interpreting navigation directions. We assume that a robot can realize valid navigation plans according to the graph.

We contribute a new end-to-end model for following directions in natural language under the behavioral navigation framework. Inspired by the information retrieval and question answering literature (Lewis and Jones, 1996; Seo et al., 2017; Xiong et al., 2016; Palangi et al., 2016), we propose to leverage the behavioral graph as a knowledge base to facilitate the interpretation of navigation commands. More specifically, the proposed model takes as input user directions in text form, the behavioral graph of the environment encoded as ⟨`node; edge; node`⟩ triplets, and the initial location of the robot in the graph. The model then predicts a set of behaviors to reach the desired destination according to the instructions and the map (Fig. 1(c)). Our main insight is that using attention mechanisms to correlate navigation instructions with the topological map of the environment can facilitate predicting correct navigation plans.

This work also contributes a new dataset of $11,050$ pairs of free-form natural language instructions and high-level navigation plans. This dataset was collected through Mechanical Turk using 100 simulated environments with a corresponding topological map and, to the best of our knowledge, it is the first of its kind for behavioral navigation. The dataset opens up opportunities to explore data-driven methods for grounding navigation commands into high-level motion plans.

We conduct extensive experiments to study the generalization capabilities of the proposed model for following natural language instructions. We investigate both generalization to new instructions in known and in new environments. We conclude this paper by discussing the benefits of the proposed approach as well as opportunities for future research based on our findings.

2 Related work

This section reviews relevant prior work on following navigation instructions. Readers interested in an in-depth review of methods to interpret spatial natural language for robotics are encouraged to refer to (Landsiedel et al., 2017).

Typical approaches to follow navigation commands deal with the complexity of natural language by manually parsing commands, constraining language descriptions, or using statistical machine translation methods. While manually parsing commands is often impractical, the first type of approaches are foundational: they showed that it is possible to leverage the compositionality of semantic units to interpret spatial language (Bugmann et al., 2004; Levit and Roy, 2007).

Constraining language descriptions can reduce the size of the input space to facilitate the interpretation of user commands. For example, (Talbot et al., 2016) explored using structured, symbolic language phrases for navigation. As in this earlier work, we are also interested in navigation with a topological map of the environment. However, we do not process symbolic phrases. Our aim is to translate free-form natural language instructions to a navigation plan using information from a high-level representation of the environment. This translation problem requires dealing with missing actions in navigation instructions and actions with preconditions, such as *"at the end of the corridor, turn right"* (MacMahon et al., 2006).

Statistical machine translation (Koehn, 2009) is at the core of recent approaches to enable robots to follow navigation instructions. These methods aim to automatically discover translation rules from a corpus of data, and often leverage the fact that navigation directions are composed of sequential commands. For instance, (Wong and Mooney, 2006; Matuszek et al., 2010; Chen and Mooney, 2011) used statistical machine translation to map instructions to a formal language defined by a grammar. Likewise, (Kollar et al., 2010; Tellex et al., 2011) mapped commands to spatial description clauses based on the hierarchical structure of language in the navigation problem. Our approach to machine translation builds on insights from these prior efforts. In particular, we focus on end-to-end learning for statistical machine translation due to the recent success of Neural Networks in Natural Language Processing (Goodfellow et al., 2016).

Our work is inspired by methods that reduce the task of interpreting user commands to a sequential prediction problem (Shimizu and Haas, 2009; Mei et al., 2016; Anderson et al., 2018). Similar to Mei et al. and Anderson et al., we use a *sequence-to-sequence* model to enable a mobile agent to follow routes. But instead leveraging visual information to output low-level navigation commands, we focus on using a topological map of the environment to output a high-level navigation plan. This plan is a sequence of behaviors that can be executed by a robot to reach a desired destination (Sepúlveda et al., 2018; Zang et al., 2018).

We explore machine translation from the perspective of automatic question answering. Following (Seo et al., 2017; Xiong et al., 2016), our approach uses attention mechanisms to learn alignments between different input modalities. In our case, the inputs to our model are navigation instructions, a topological environment map, and the start location of the robot (Fig. 1(c)). Our results show that the map can serve as an effective source of contextual information for the translation task. Additionally, it is possible to leverage this kind of information in an end-to-end fashion.

3 Problem Formulation

Our goal is to translate navigation instructions in text form into a sequence of behaviors that a robot can execute to reach a desired destination from a known start location. We frame this problem under a behavioral approach to indoor autonomous navigation (Sepúlveda et al., 2018) and assume that prior knowledge about the environment is available for the translation task. This prior knowledge is a topological map, in the form of a behavioral navigation graph (Fig. 1(b)). The nodes of the graph correspond to semantically-meaningful locations for the navigation task, and its directed edges are visuo-motor behaviors that a robot can use to move between nodes. This formulation takes advantage of the rich semantic structure behind man-made environments, resulting in a compact route representation for robot navigation.

Fig. 1(c) provides a schematic view of the problem setting. The inputs are: (1) a navigation graph m, (2) the starting node s of the robot in m, and (3) a set of free-form navigation instructions I in natural language. The instructions describe a path in the graph to reach from s to a – potentially implicit – destination node g. Using this information, the objective is to predict a suitable sequence of robot behaviors $b_1, \ldots, b_T$ to navigate from s to g according to I. From a supervised learning perspective, the goal is then to estimate:

$$\underset{b_1,\ldots,b_T}{\operatorname{argmax}} P(b_1, \ldots, b_T | m, s, I) \tag{1}$$

based on a dataset of input-target pairs $\{(\mathbf{x}_i, \mathbf{y}_i) \,|\, 0 \leq i \leq N\}$, where $\mathbf{x}_i = (m, s, I)_i$ and $\mathbf{y}_i = (b_1, \ldots, b_T)_i$, respectively. The sequential execution of the behaviors $b_1, \ldots, b_T$ should replicate the route intended by the instructions I.

We assume no prior linguistic knowledge. Thus, translation approaches have to cope with the semantics and syntax of the language by discovering corresponding patterns in the data.

3.1 The Behavioral Graph: A Knowledge Base For Navigation

We view the behavioral graph m as a knowledge base that encodes a set of navigational rules as triplets $\langle p_i; b_l[attr]; p_j\rangle$, where p_i and p_j are adjacent nodes in the graph, and the edge b_l is an executable behavior to navigate from p_i to p_j. In general, each behaviors includes a list of relevant navigational attributes $attr$ that the robot might encounter when moving between nodes.

Behavior	Description
oo<d>	Go out of the current place and turn <d>
io<d>	Turn <d> and enter the place straight ahead
oio	Exit current place and enter straight ahead
<d>t	Turn <d> at the intersection
cf	Follow (or go straight down) the corridor
sp	Go straight at a T intersection
ch<d>	Cross the hall and turn <d>

Table 1: Behaviors (edges) of the navigation graphs considered in this work. The direction <d> can be left or right.

We consider 7 types of semantic locations, 11 types of behaviors, and 20 different types of landmarks. A location in the navigation graph can be a room, a lab, an office, a kitchen, a hall, a corridor, or a bathroom. These places are labeled with unique tags, such as "room-1" or "lab-2", except for bathrooms and kitchens which people do not typically refer to by unique names when describing navigation routes.

Table 1 lists the navigation behaviors that we consider in this work. These behaviors can be described in reference to visual landmarks or objects, such as paintings, book shelfs, tables, etc. As in Fig. 1, maps might contain multiple landmarks of the same type. Please see the supplementary material (Appendix A) for more details.

4 Approach

We leverage recent advances in deep learning to translate natural language instructions to a sequence of navigation behaviors in an end-to-end fashion. Our proposed model builds on the sequence-to-sequence translation model of (Bahdanau et al., 2015), which computes a soft-alignment between a source sequence (natural language instructions in our case) and the corresponding target sequence (navigation behaviors).

As one of our main contributions, we augment the neural machine translation approach of Bahdanau et al. to take as input not only natural language instructions, but also the corresponding behavioral navigation graph m of the environment where navigation should take place. Specifically, at each step, the graph m operates as a knowledge base that the model can access to obtain information about path connectivity, facilitating the grounding of navigation commands.

Figure 2 shows the structure of the proposed model for interpreting navigation instructions. The model consists of six layers:

Embed layer: The model first encodes each word and symbol in the input sequences I and m into fixed-length representations. The instructions I are embedded into a 100-dimensional pre-trained GloVe vector (Pennington et al., 2014). Each of the triplet components, p_i, $b_l[attr]$, and p_j of the graph m, are one-hot encoded into vectors of dimensionality $2N+E$, where N and E are the number of nodes and edges in m, respectively.

Encoder layer: The model then uses two bidirectional Gated Recurrent Units (GRUs) (Cho et al., 2014) to independently process the information from I and m, and incorporate contextual cues from the surrounding embeddings in each sequence. The outputs of the encoder layer are the matrix $\bar{I} \in \mathbb{R}^{T\times 2H}$ for the navigational commands and the matrix $\bar{G} \in \mathbb{R}^{L\times 2H}$ for the behavioral graph, where H is the hidden size of each GRU, T is the number of words in the instruction I, and L is the number of triplets in the graph m.

Attention layer: Matrices $\bar{I}$ and $\bar{G}$ generated by the encoder layer are combined using an attention mechanism. We use one-way attention because the graph contains information about the whole environment, while the instruction has (potentially incomplete) local information about the route of interest. The use of attention provides our model with a two-step strategy to interpret commands. This resembles the way people find paths on a map: first, relevant parts on the map are selected according to their affinity to each of the words in the input instruction (attention layer); second, the selected parts are connected to assemble a valid path (decoder layer). More formally, let $\bar{G}_i$ ($i \in [1, L]$) be the i-th row of $\bar{G}$, and $\bar{I}_j$ ($j \in [1, T]$) the j-th row of $\bar{I}$. We use each encoded triplet $\bar{G}_i$ in $\bar{G}$ to calculate its associated attention distribution $a_i \in \mathbb{R}^T$ over all the atomic instructions $\bar{I}_j$:

$$e_i = [\bar{G}_i W \bar{I}_1^{\intercal}, \ldots, \bar{G}_i W \bar{I}_T^{\intercal}] \quad (2)$$

$$a_i = softmax(e_i) \quad (3)$$

where the matrix $W \in \mathbb{R}^{2H\times 2H}$ serves to combine the different sources of information $\bar{G}$ and $\bar{I}$. Each component a_{ij} of the attention distributions a_i quantifies the affinity between the i-th triplet in $\bar{G}$ and the j-th word in the corresponding input I.

The model then uses each attention distribution a_i to obtain a weighted sum of the encodings of the words in $\bar{I}$, according to their relevance to the corresponding triplet $\bar{G}_i$. This results in L attention vectors $R_i \in \mathbb{R}^{2H}$, $R_i = \sum_{j=1}^{T} a_{ij} I_j$.

The final step in the attention layer concatenates each R_i with $\bar{G}_i$ to generate the outputs

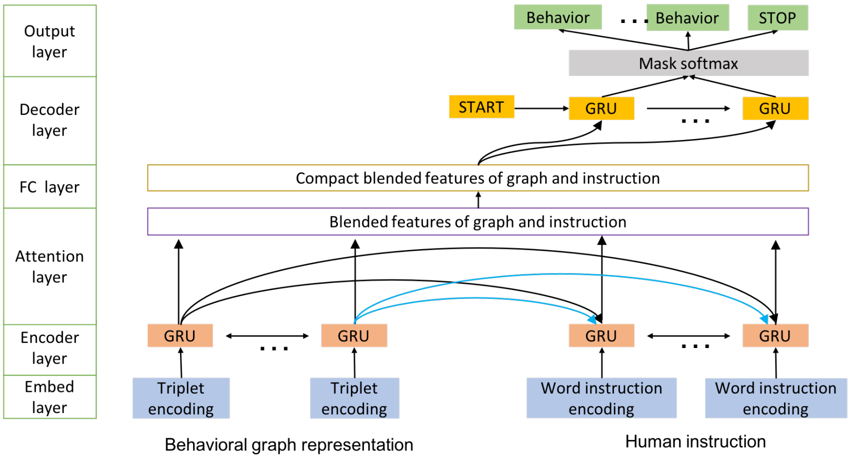

Figure 2: Model overview. The model contains six layers, takes the input of behavioral graph representation, free-form instruction, and the start location (yellow block marked as START in the decoder layer) and outputs a sequence of behaviors.

$F_i = [R_i; \bar{G}_i]$, $i \in [1, L]$. Following (Seo et al., 2017), we include the encoded triplet $\bar{G}_i$ in the output tensor F_i of this layer to prevent early summaries of relevant map information.

FC layer: The model reduces the dimensionality of each individual vector F_i from $4H$ to H with a fully-connected (FC) layer. The resulting L vectors are output to the next layer as columns of a context matrix $C \in \mathbb{R}^{H \times L}$.

Decoder layer: After the FC layer, the model predicts likelihoods over the sequence of behaviors that correspond to the input instructions with a GRU network. Without loss of generality, consider the t-th recurrent cell in the GRU network. This cell takes two inputs: a hidden state vector h_{t-1} from the prior cell, and a one-hot embedding of the previous behavior b_{t-1} that was predicted by the model. Based on these inputs, the GRU cell outputs a new hidden state h_t to compute likelihoods for the next behavior. These likelihoods are estimated by combining the output state h_t with relevant information from the context C:

$$\hat{d}_{ts} = v_a^\intercal \tanh(W_1 h_t + W_2 C_s) \quad (4)$$

$$d_t = softmax(\hat{d}_{t1}, \ldots, \hat{d}_{tL}) \quad (5)$$

where W_1, W_2, and v_a are trainable parameters. The attention vector $d_t \in \mathbb{R}^L$ in Eq. (5) quantifies the affinity of h_t with respect to each of the columns C_s of C, where $s \in [1, L]$. The attention vector also helps to estimate a dynamic contextual vector $S_t = \sum_{s=1}^{L} d_{ts} C_s$ that the t-th GRU cell uses to compute logits for the next behavior:

$$o_t = W_3 [S_t; h_t] \quad (6)$$

with W_3 trainable parameters. Note that o_t includes a value for each of the pre-defined behaviors in the graph m, as well as for a special "*stop*" symbol to identify the end of the output sequence.

Output layer: The final layer of the model searches for a valid sequence of robot behaviors based on the robot's initial node, the connectivity of the graph m, and the output logits from the previous decoder layer. Again, without loss of generality, consider the t-th behavior b_t that is finally predicted by the model. The search for this behavior is implemented as:

$$b_t = argmax(softmax(o_t + mask(m, n_t))) \quad (7)$$

with $mask(m, n_t)$ a masking function that takes as input the graph m and the node n_t that the robot reaches after following the sequence of behaviors $b_1, \ldots, b_{t-1}$ previously predicted by the model. The $mask$ function returns a vector of the same dimensionality as the logits o_t, but with zeros for the valid behaviors after the last location n_t and for the special stop symbol, and $-\inf$ for any invalid predictions according to the connectivity of the behavioral navigation graph.

5 Dataset

We created a new dataset for the problem of following navigation instructions under the behavioral navigation framework of (Sepúlveda et al., 2018).[1] This dataset was created using Amazon Mechanical Turk and 100 maps of simulated indoor environments, each with 6 to 65 rooms. To the best of our knowledge, this is the first bench-

---

[1] The dataset is publicly available through the website: `follow-nav-directions.stanford.edu`.

Dataset	# Single	# Double	Total
Training	4062	2002	8066
Test-Repeated	944	34	1012
Test-New	962	0	962

Table 2: Dataset statistics. "# Single" indicates the number of navigation plans with a single natural language instruction. "# Double" is the number of plans with two different instructions. The total number of plans is (# Single) × 2(# Double).

mark for comparing translation models in the context of behavioral robot navigation.

As shown in Table 2, the dataset consists of 8066 pairs of free-form natural language instructions and navigation plans for training. This training data was collected from 88 unique simulated environments, totaling 6064 distinct navigation plans (2002 plans have two different navigation instructions each; the rest has one). The dataset contains two test set variants:

1) Test-Repeated: Contains 1012 pairs of instructions and navigation plans. These routes are not part of the training set; however, they are collected using environments that are part of the training set.

2) Test-New: Contains 962 pairs of instructions and navigation plans. This test set is more challenging than the Test-Repeated dataset because it contains new routes on 12 new indoor environments not included in the training set.

While the dataset was collected with simulated environments, no structure was imposed on the navigation instructions while crowd-sourcing data. Thus, many instructions in our dataset are ambiguous. Moreover, the order of the behaviors in the instructions is not always the same. For instance, a person said *"turn right and advance"* to describe part of a route, while another person said *"go straight after turning right"* in a similar situation. The high variability present in the natural language descriptions of our dataset makes the problem of decoding instructions into behaviors not trivial. See Appendix A of the supplementary material for additional details on our data collection effort.

6 Experiments

This section describes our evaluation of the proposed approach for interpreting navigation commands in natural language. We provide both quantitative and qualitative results.

6.1 Evaluation Metrics

While computing evaluation metrics, we only consider the behaviors present in the route because they are sufficient to recover the high-level navigation plan from the graph. Our metrics treat each behavior as a single token. For example, the sample plan "R-1 oor C-1 cf C-1 lt C-0 cf C-0 iol O-3" is considered to have 5 tokens, each corresponding to one of its behaviors ("oor", "cf", "lt", "cf", "iol"). In this plan, "R-1","C-1", "C-0", and "O-3" are symbols for locations (nodes) in the graph.

We compare the performance of translation approaches based on four metrics:

- Exact Match (EM). As in (Shimizu and Haas, 2009), EM is 1 if a predicted plan matches exactly the ground truth; otherwise it is 0.

- F1 score (F1). The harmonic average of the precision and recall over all the test set (Chinchor and Sundheim, 1993).

- Edit Distance (ED). The minimum number of insertions, deletions or swap operations required to transform a predicted sequence of behaviors into the ground truth sequence (Navarro, 2001).

- Goal Match (GM). GM is 1 if a predicted plan reaches the ground truth destination (even if the full sequence of behaviors does not match exactly the ground truth). Otherwise, GM is 0.

6.2 Models Used in the Evaluation

We compare the proposed approach for translating natural language instructions into a navigation plan against alternative deep-learning models:

Baseline model. The baseline approach is based on (Shimizu and Haas, 2009). It divides the task of interpreting commands for behavioral navigation into two steps: path generation, and path verification. For path generation, this baseline uses a standard sequence-to-sequence model augmented with an attention mechanism, similar to (Bahdanau et al., 2015; Zang et al., 2018). For path verification, the baseline uses depth-first search to find a route in the graph that matches the sequence of predicted behaviors. If no route matches perfectly, the baseline changes up to three behaviors in the predicted sequence to try to turn it into a valid path.

Ablation model. To test the impact of using the behavioral graphs as an extra input to our translation model, we implemented a version of our approach that only takes natural language instructions as input. In this ablation model, the output of the bidirectional GRU that encodes the input instruction I is directly fed to the decoder layer. This model does not have the attention and FC layers described in Sec. 4, nor uses the masking function in the output layer.

Ablation with mask model. This model is the same as the previous Ablation model, but with the masking function in the output layer.

6.3 Implementation Details

We pre-processed the inputs to the various models that are considered in our experiment. In particular, we lowercased, tokenized, spell-checked and lemmatized the input instructions in text-form using WordNet (Miller, 1995). We also truncated the graphs to a maximum of 300 triplets, and the navigational instructions to a maximum of 150 words. Only 6.4% (5.4%) of the unique graphs in the training (validation) set had more than 300 triplets, and less than 0.15% of the natural language instructions in these sets had more than 150 tokens.

The dimensionality of the hidden state of the GRU networks was set to 128 in all the experiments. In general, we used 12.5% of the training set as validation for choosing models' hyperparameters. In particular, we used dropout after the encoder and the fully-connected layers of the proposed model to reduce overfitting. Best performance was achieved with a dropout rate of 0.5 and batch size equal to 256. We also used scheduled sampling (Bengio et al., 2015) at training time for all models except the baseline.

We input the triplets from the graph to our proposed model in alphabetical order, and consider a modification where the triplets that surround the start location of the robot are provided first in the input graph sequence. We hypothesized that such rearrangement would help identify the starting location (node) of the robot in the graph. In turn, this could facilitate the prediction of correct output sequences. In the remaining of the paper, we refer to models that were provided a rearranged graph, beginning with the starting location of the robot, as models with "Ordered Triplets".

6.4 Quantitative Evaluation

Table 3 shows the performance of the models considered in our evaluation on both test sets. The next two sections discuss the results in detail.

6.4.1 Performance in the Test-Repeated Set

First, we can observe that the final model "Ours with Mask and Ordered Triplets" outperforms the Baseline and Ablation models on all metrics in previously seen environments. The difference in performance is particularly evident for the Exact Match and Goal Match metrics, with our model increasing accuracy by 35% and 25% in comparison to the Baseline and Ablation models, respectively. These results suggest that providing the behavioral navigation graph to the model and allowing it to process this information as a knowledge base in an end-to-end fashion is beneficial.

We can also observe from Table 3 that the masking function of Eq. (7) tends to increase performance in the Test-Repeated Set by constraining the output sequence to a valid set of navigation behaviors. For the Ablation model, using the masking function leads to about 10% increase in EM and GM accuracy. For the proposed model (with or without reordering the graph triplets), the increase in accuracy is around 4%. Note that the impact of the masking function is less evident in terms of the F1 score because this metric considers if a predicted behavior exists in the ground truth navigation plan, irrespective of its specific position in the output sequence.

The results in the last four rows of Table 3 suggest that ordering the graph triplets can facilitate predicting correct navigation plans in previously seen environments. Providing the triplets that surround the starting location of the robot first to the model leads to a boost of 4% in EM and GM performance. The rearrangement of the graph triplets also helps to reduce ED and increase F1.

Lastly, it is worth noting that our proposed model (last row of Table 3) outperforms all other models in previously seen environments. In particular, we obtain over 4% increase in EM and GM between our model and the next best two models.

6.4.2 Performance in the Test-New Set

The previous section evaluated model performance on new instructions (and corresponding navigation plans) for environments that were previously seen at training time. Here, we examine whether the trained models succeed on environments that are completely new.

The evaluation on the Test-New Set helps understand the generalization capabilities of the models under consideration. This experiment is more challenging than the one in the previous sec-

Model	Test-Repeated Set				Test-New Set			
	EM ↑	F1 ↑	ED ↓	GM ↑	EM ↑	F1↑	ED ↓	GM ↑
Baseline	25.30	79.83	2.53	26.28	25.44	81.38	2.39	25.44
Ablation	36.36	90.28	1.36	36.36	24.82	88.65	1.71	24.92
Ablation with Mask	45.95	90.08	1.20	46.05	36.45	88.31	1.45	36.56
Ours without Mask	52.47	91.74	0.95	53.95	21.94	87.50	1.78	22.65
Ours with Mask	57.31	91.91	0.91	57.31	38.52	88.98	1.32	38.52
Ours without Mask and with Ordered Triplets	57.21	93.37	0.79	57.71	33.36	**91.02**	1.37	33.78
Ours with Mask and Ordered Triplets	**61.17**	**93.54**	**0.75**	**61.36**	**41.71**	90.22	**1.22**	**41.81**

Table 3: Performance of different models on the test datasets. EM and GM report percentages, and ED corresponds to average edit distance. The symbol ↑ indicates that higher results are better in the corresponding column; ↓ indicates that lower is better.

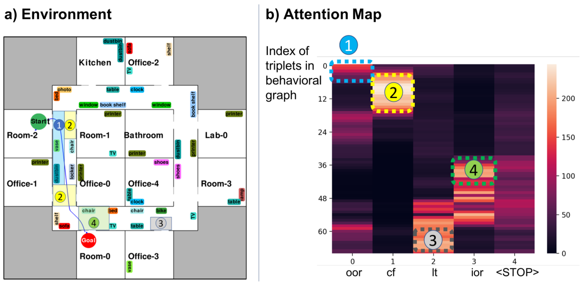

Figure 3: Visualization of the attention weights of the decoder layer. The color-coded and numbered regions on the map (left) correspond to the triplets that are highlighted with the corresponding color in the attention map (right).

tion, as can be seen in performance drops in Table 3 for the new environments. Nonetheless, the insights from the previous section still hold: masking in the output layer and reordering the graph triplets tend to increase performance.

Even though the results in Table 3 suggest that there is room for future work on decoding natural language instructions, our model still outperforms the baselines by a clear margin in new environments. For instance, the difference between our model and the second best model in the Test-New set is about 3% EM and GM. Note that the average number of actions in the ground truth output sequences is 7.07 for the Test-New set. Our model's predictions are just 1.22 edits off on average from the correct navigation plans.

6.5 Qualitative Evaluation

This section discusses qualitative results to better understand how the proposed model uses the navigation graph.

6.5.1 Attention Visualization

We analyze the evolution of the attention weights d_t in Eq. (5) to assess if the decoder layer of the proposed model is attending to the correct parts of the behavioral graph when making predictions. Fig 3(b) shows an example of the resulting attention map for the case of a correct prediction. In the Figure, the attention map is depicted as a scaled and normalized 2D array of color codes. Each column in the array shows the attention distribution d_t used to generate the predicted output at step t. Consequently, each row in the array represents a triplet in the corresponding behavioral graph. This graph consists of 72 triplets for Fig 3(b).

We observe a locality effect associated to the attention coefficients corresponding to high values (bright areas) in each column of Fig 3(b). This suggests that the decoder is paying attention to graph triplets associated to particular neighborhoods of the environment in each prediction step. We include additional attention visualizations in the supplementary Appendix, including cases where the dynamics of the attention distribution are harder to interpret.

6.5.2 Experiments with Sub-Optimal Paths

All the routes in our dataset are the shortest paths from a start location to a given destination. Thus, we collected a few additional natural language instructions to check if our model was able to follow navigation instructions describing sub-optimal paths. One such example is shown in Fig. 4, where the blue route (shortest path) and the red route (alternative path) are described by:

- **Blue route:** "*Go out the office and make a left. Turn right at the corner and go down the hall. Make a right at the next corner and enter the kitchen in front of table.*"
- **Red route:** "*Exit the room 0 and turn right, go to the end of the corridor and turn left, go straight to the end of the corridor and turn left again. After passing bookshelf on your left and table on your right, Enter the kitchen on your right.*"

For both routes, the proposed model was able to predict the correct sequence of navigation behaviors. This result suggests that the model is indeed using the input instructions and is not just approximating shortest paths in the behavioral graph.

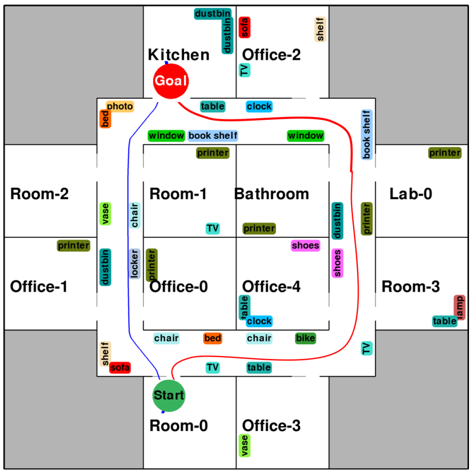

Figure 4: An example of two different navigation paths between the same pair of start and goal locations.

Other examples on the prediction of sub-optimal paths are described in the Appendix.

7 Conclusion

This work introduced behavioral navigation through free-form natural language instructions as a challenging and a novel task that falls at the intersection of natural language processing and robotics. This problem has a range of interesting cross-domain applications, including information retrieval.

We proposed an end-to-end system to translate user instructions to a high-level navigation plan. Our model utilized an attention mechanism to merge relevant information from the navigation instructions with a behavioral graph of the environment. The model then used a decoder to predict a sequence of navigation behaviors that matched the input commands.

As part of this effort, we contributed a new dataset of 11,051 pairs of user instructions and navigation plans from 100 different environments. Our model achieved the best performance in this dataset in comparison to a two-step baseline approach for interpreting navigation instructions, and a sequence-to-sequence model that does not consider the behavioral graph. Our quantitative and qualitative results suggest that attention mechanisms can help leverage the behavioral graph as a relevant knowledge base to facilitate the translation of free-form navigation instructions. Overall, our approach demonstrated practical form of learning for a complex and useful task.

In future work, we are interested in investigating mechanisms to improve generalization to new environments. For example, pointer and graph networks (Vinyals et al., 2015; Defferrard et al., 2016) are a promising direction to help supervise translation models and predict motion behaviors.

Acknowledgments

The Toyota Research Institute (TRI) provided funds to assist with this research, but this paper solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. This work is also partially funded by Fondecyt grant 1181739, Conicyt, Chile. The authors would also like to thank Gabriel Sepúlveda for his assistance with parts of this project.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics (TACL)*, 1:49–62.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Guido Bugmann, Ewan Klein, Stanislao Lauria, and Theocharis Kyriacou. 2004. Corpus-based robotics: A route instruction example. In *Proceedings of Intelligent Autonomous Systems (IAS)*, pages 96–103. Citeseer.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *AAAI Conference on Artificial Intelligence*, pages 859–865.

Nancy Chinchor and Beth Sundheim. 1993. Muc-5 evaluation metrics. In *Proceedings of the 5th conference on Message understanding*, pages 69–78. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, aglar Gülehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

P. Koehn. 2009. *Statistical Machine Translation*. Cambridge University Press.

T. Kollar, S. Tellex, D. Roy, and N. Roy. 2010. Toward understanding natural language directions. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 259–266.

Christian Landsiedel, Verena Rieser, Matthew Walter, and Dirk Wollherr. 2017. A review of spatial reasoning and interaction for real-world robotics. *Advanced Robotics*, 31(5):222–242.

Michael Levit and Deb Roy. 2007. Interpretation of spatial language in a map navigation task. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(3):667–679.

David D Lewis and Karen Spärck Jones. 1996. Natural language processing for information retrieval. *Communications of the ACM*, 39(1):92–101.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *National Conference on Artificial Intelligence (AAAI)*.

C. Matuszek, D. Fox, and K. Koscher. 2010. Following directions using statistical machine translation. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 251–258.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *National Conference on Artificial Intelligence (AAAI)*, pages 2772–2778.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations ICLR*.

G. Sepúlveda, JC. Niebles, and A. Soto. 2018. A deep learning based behavioral approach to indoor autonomous navigation. In *International Conference on Learning Representations (ICRA)*.

Nobuyuki Shimizu and Andrew R. Haas. 2009. Learning to follow navigational route instructions. In *International Joint Conferences on Artificial Intelligence (IJCAI)*.

Ben Talbot, Obadiah Lam, Ruth Schulz, Feras Dayoub, Ben Upcroft, and Gordon Wyeth. 2016. Find my office: Navigating real space from semantic descriptions. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5782–5787.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *National Conference on Artificial Intelligence (AAAI)*, volume 1, page 2.

Manuela M Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. 2015. Cobots: Robust symbiotic autonomous mobile service robots. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, page 4423.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700.

Yuk Wah Wong and Raymond J Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proc. of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 439–446.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning (ICML)*, pages 2397–2406.

Xiaoxue Zang, Marynel Vázquez, Juan Carlos Niebles, Alvaro Soto, and Silvio Savarese. 2018. Behavioral indoor navigation with natural language directions. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 283–284.

Mapping Instructions to Actions in 3D Environments with Visual Goal Prediction

Dipendra Misra **Andrew Bennett** **Valts Blukis**
Eyvind Niklasson **Max Shatkhin** **Yoav Artzi**

Department of Computer Science and Cornell Tech, Cornell University, New York, NY, 10044
`{dkm, awbennett, valts, yoav}@cs.cornell.edu`
`{een7, ms3448}@cornell.edu`

Abstract

We propose to decompose instruction execution to goal prediction and action generation. We design a model that maps raw visual observations to goals using LINGUNET, a language-conditioned image generation network, and then generates the actions required to complete them. Our model is trained from demonstration only without external resources. To evaluate our approach, we introduce two benchmarks for instruction following: LANI, a navigation task; and CHAI, where an agent executes household instructions. Our evaluation demonstrates the advantages of our model decomposition, and illustrates the challenges posed by our new benchmarks.

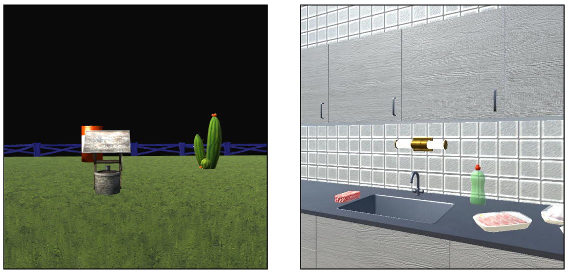

After reaching the hydrant head towards the blue fence and pass towards the right side of the well.

Put the cereal, the sponge, and the dishwashing soap into the cupboard above the sink.

Figure 1: Example instructions from our two tasks: LANI (left) and CHAI (right). LANI is a landmark navigation task, and CHAI is a corpus of instructions in the CHALET environment.

1 Introduction

Executing instructions in interactive environments requires mapping natural language and observations to actions. Recent approaches propose learning to directly map from inputs to actions, for example given language and either structured observations (Mei et al., 2016; Suhr and Artzi, 2018) or raw visual observations (Misra et al., 2017; Xiong et al., 2018). Rather than using a combination of models, these approaches learn a single model to solve language, perception, and planning challenges. This reduces the amount of engineering required and eliminates the need for hand-crafted meaning representations. At each step, the agent maps its current inputs to the next action using a single learned function that is executed repeatedly until task completion.

Although executing the same computation at each step simplifies modeling, it exemplifies certain inefficiencies; while the agent needs to decide what action to take at each step, identifying its goal is only required once every several steps or even once per execution. The left instruction in Figure 1 illustrates this. The agent can compute its goal once given the initial observation, and given this goal can then generate the actions required. In this paper, we study a new model that explicitly distinguishes between goal selection and action generation, and introduce two instruction following benchmark tasks to evaluate it.

Our model decomposes into goal prediction and action generation. Given a natural language instruction and system observations, the model predicts the goal to complete. Given the goal, the model generates a sequence of actions.

The key challenge we address is designing the goal representation. We avoid manually designing a meaning representation, and predict the goal in the agent's observation space. Given the image of the environment the agent observes, we generate a probability distribution over the image to highlight the goal location. We treat this prediction as image generation, and develop LINGUNET, a language conditioned variant of the U-NET image-to-image architecture (Ronneberger et al., 2015). Given the visual goal prediction, we generate actions using a recurrent neural network (RNN).

Our model decomposition offers two key advantages. First, we can use different learning methods as appropriate for the goal prediction and action

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2667–2678
Brussels, Belgium, October 31 - November 4, 2018.

generation problems. We find supervised learning more effective for goal prediction, where only a limited amount of natural language data is available. For action generation, where exploration is critical, we use policy gradient in a contextual bandit setting (Misra et al., 2017). Second, the goal distribution is easily interpretable by overlaying it on the agent observations. This can be used to increase the safety of physical systems by letting the user verify the goal before any action is executed. Despite the decomposition, our approach retains the advantages of the single-model approach. It does not require designing intermediate representations, and training does not rely on external resources, such as pre-trained parsers or object detectors, instead using demonstrations only.

We introduce two new benchmark tasks with different levels of complexity of goal prediction and action generation. LANI is a 3D navigation environment and corpus, where an agent navigates between landmarks. The corpus includes 6,000 sequences of natural language instructions, each containing on average 4.7 instructions. CHAI is a corpus of 1,596 instruction sequences, each including 7.7 instructions on average, for CHALET, a 3D house environment (Yan et al., 2018). Instructions combine navigation and simple manipulation, including moving objects and opening containers. Both tasks require solving language challenges, including spatial and temporal reasoning, as well as complex perception and planning problems. While LANI provides a task where most instructions include a single goal, the CHAI instructions often require multiple intermediate goals. For example, the household instruction in Figure 1 can be decomposed to eight goals: opening the cupboard, picking each item and moving it to the cupboard, and closing the cupboard. Achieving each goal requires multiple actions of different types, including moving and acting on objects. This allows us to experiment with a simple variation of our model to generate intermediate goals.

We compare our approach to multiple recent methods. Experiments on the LANI navigation task indicate that decomposing goal prediction and action generation significantly improves instruction execution performance. While we observe similar trends on the CHAI instructions, results are overall weaker, illustrating the complexity of the task. We also observe that inherent ambiguities in instruction following make exact goal identification difficult, as demonstrated by imperfect human performance. However, the gap to human-level performance still remains large across both tasks. Our code and data are available at `github.com/clic-lab/ciff`.

2 Technical Overview

Task Let $\mathcal{X}$ be the set of all *instructions*, $\mathcal{S}$ the set of all *world states*, and $\mathcal{A}$ the set of all *actions*. An instruction $\bar{x} \in \mathcal{X}$ is a sequence $\langle x_1, \ldots, x_n \rangle$, where each x_i is a token. The agent executes instructions by generating a sequence of actions, and indicates execution completion with the special action `STOP`.

The sets of actions $\mathcal{A}$ and states $\mathcal{S}$ are domain specific. In the navigation domain LANI, the actions include moving the agent and changing its orientation. The state information includes the position and orientation of the agent and the different landmarks. The agent actions in the CHALET house environment include moving and changing the agent orientation, as well as an object interaction action. The state encodes the position and orientation of the agent and all objects in the house. For interactive objects, the state also includes their status, for example if a drawer is open or closed. In both domains, the actions are discrete. The domains are described in Section 6.

Model The agent does not observe the world state directly, but instead observes its pose and an RGB image of the environment from its point of view. We define these observations as the agent context $\tilde{s}$. An agent model is a function from an agent context $\tilde{s}$ to an action $a \in \mathcal{A}$. We model goal prediction as predicting a probability distribution over the agent visual observations, representing the likelihood of locations or objects in the environment being target positions or objects to be acted on. Our model is described in Section 4.

Learning We assume access to training data with N examples $\{(\bar{x}^{(i)}, s_1^{(i)}, s_g^{(i)})\}_{i=1}^N$, where $\bar{x}^{(i)}$ is an instruction, $s_1^{(i)}$ is a start state, and $s_g^{(i)}$ is the goal state. We decompose learning; training goal prediction using supervised learning, and action generation using oracle goals with policy gradient in a contextual bandit setting. We assume an instrumented environment with access to the world state, which is used to compute rewards during training only. Learning is described in Section 5.

Evaluation We evaluate task performance on a test set $\{(\bar{x}^{(i)}, s_1^{(i)}, s_g^{(i)})\}_{i=1}^M$, where $\bar{x}^{(i)}$ is an instruction, $s_1^{(i)}$ is a start state, and $s_g^{(i)}$ is the goal state. We evaluate task completion accuracy and the distance of the agent's final state to $s_g^{(i)}$.

3 Related Work

Mapping instruction to action has been studied extensively with intermediate symbolic representations (e.g., Chen and Mooney, 2011; Kim and Mooney, 2012; Artzi and Zettlemoyer, 2013; Artzi et al., 2014; Misra et al., 2015, 2016). Recently, there has been growing interest in direct mapping from raw visual observations to actions (Misra et al., 2017; Xiong et al., 2018; Anderson et al., 2018; Fried et al., 2018). We propose a model that enjoys the benefits of such direct mapping, but explicitly decomposes that task to interpretable goal prediction and action generation. While we focus on natural language, the problem has also been studied using synthetic language (Chaplot et al., 2018; Hermann et al., 2017).

Our model design is related to hierarchical reinforcement learning, where sub-policies at different levels of the hierarchy are used at different frequencies (Sutton et al., 1998). Oh et al. (2017) uses a two-level hierarchy for mapping synthetic language to actions. Unlike our visual goal representation, they use an opaque vector representation. Also, instead of reinforcement learning, our methods emphasize sample efficiency.

Goal prediction is related to referring expression interpretation (Matuszek et al., 2012a; Krishnamurthy and Kollar, 2013; Kazemzadeh et al., 2014; Kong et al., 2014; Yu et al., 2016; Mao et al., 2016; Kitaev and Klein, 2017). While our model solves a similar problem for goal prediction, we focus on detecting visual goals for actions, including both navigation and manipulation, as part of an instruction following model. Using formal goal representation for instruction following was studied by MacGlashan et al. (2015). In contrast, our model generates a probability distribution over images, and does not require an ontology.

Our data collection is related to existing work. LANI is inspired by the HCRC Map Task (Anderson et al., 1991), where a leader directs a follower to navigate between landmarks on a map. We use a similar task, but our scalable data collection process allows for a significantly larger corpus. We also provide an interactive navigation environment, instead of only map diagrams. Unlike Map Task, our leaders and followers do not interact in real time. This abstracts away interaction challenges, similar to how the SAIL navigation corpus was collected (MacMahon et al., 2006). CHAI instructions were collected using scenarios given to workers, similar to the ATIS collection process (Hemphill et al., 1990; Dahl et al., 1994). Recently, multiple 3D research environments were released. LANI has a significantly larger state space than existing navigation environments (Hermann et al., 2017; Chaplot et al., 2018), and CHALET, the environment used for CHAI, is larger and has more complex manipulation compared to similar environments (Gordon et al., 2018; Das et al., 2018). In addition, only synthetic language data has been released for these environment. An exception is the Room-to-Room dataset (Anderson et al., 2018) that makes use of an environment of connected panoramas of house settings. Although it provides a realistic vision challenge, unlike our environments, the state space is limited to a small number of panoramas and manipulation is not possible.

4 Model

We model the agent policy as a neural network. The agent observes the world state s_t at time t as an RGB image $\mathbf{I}_t$. The agent context $\tilde{s}_t$, the information available to the agent to select the next action a_t, is a tuple $(\bar{x}, \mathbf{I}_P, \langle(\mathbf{I}_1, p_1), \ldots, (\mathbf{I}_t, p_t)\rangle)$, where $\bar{x}$ is the natural language instructions, $\mathbf{I}_P$ is a panoramic view of the environment from the starting position at time $t = 1$, and $\langle(\mathbf{I}_1, p_1), \ldots, (\mathbf{I}_t, p_t)\rangle$ is the sequence of observations $\mathbf{I}_t$ and poses p_t up to time t. The panorama $\mathbf{I}_P$ is generated through deterministic exploration by rotating 360° to observe the environment at the beginning of the execution.[1]

The model includes two main components: goal prediction and action generation. The agent uses the panorama $\mathbf{I}_P$ to predict the goal location l_g. At each time step t, a projection of the goal location into the agent's current view $\mathbf{M}_t$ is given as input to an RNN to generate actions. The probability of an action a_t at time t decomposes to:

$$P(a_t \mid \tilde{s}_t) = \sum_{l_g} \Big(P(l_g \mid \bar{x}, \mathbf{I}_P) \\ P(a_t \mid l_g, (\mathbf{I}_1, p_1), \ldots, (\mathbf{I}_t, p_t)) \Big) \ ,$$

where the first term puts the complete distribution mass on a single location (i.e., a delta function). Figure 2 illustrates the model.

---

[1] The panorama is a concatenation of deterministic observations along the width dimension. For simplicity, we do not include these deterministic steps in the execution.

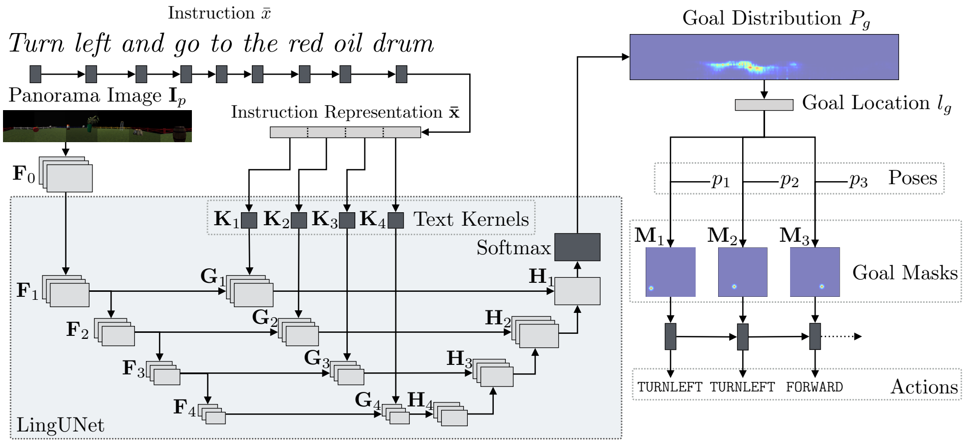

Figure 2: An illustration for our architecture (Section 4) for the instruction *turn left and go to the red oil drum* with a LINGUNET depth of $m = 4$. The instruction $\bar{x}$ is mapped to $\bar{\mathbf{x}}$ with an RNN, and the initial panorama observation $\mathbf{I}_P$ to $\mathbf{F}_0$ with a CNN. LINGUNET generates $\mathbf{H}_1$, a visual representation of the goal. First, a sequence of convolutions maps the image features $\mathbf{F}_0$ to feature maps $\mathbf{F}_1,\ldots,\mathbf{F}_4$. The text representation $\bar{\mathbf{x}}$ is used to generate the kernels $\mathbf{K}_1,\ldots,\mathbf{K}_4$, which are convolved to generate the text-conditioned feature maps $\mathbf{G}_1,\ldots,\mathbf{G}_4$. These feature maps are de-convolved to $\mathbf{H}_1,\ldots,\mathbf{H}_4$. The goal probability distribution P_g is computed from $\mathbf{H}_1$. The goal location is the inferred from the max of P_g. Given l_g and p_t, the pose at step t, the goal mask $\mathbf{M}_t$ is computed and passed into an RNN that outputs the action to execute.

Goal Prediction To predict the goal location, we generate a probability distribution P_g over a feature map $\mathbf{F}_0$ generated using convolutions from the initial panorama observation $\mathbf{I}_P$. Each element in the probability distribution P_g corresponds to an area in $\mathbf{I}_P$. Given the instruction $\bar{x}$ and panorama $\mathbf{I}_P$, we first generate their representations. From the panorama $\mathbf{I}_P$, we generate a feature map $\mathbf{F}_0 = [\text{CNN}_0(\mathbf{I}_P); \mathbf{F}^p]$, where CNN_0 is a two-layer convolutional neural network (CNN; LeCun et al., 1998) with rectified linear units (ReLU; Nair and Hinton, 2010) and $\mathbf{F}^p$ are positional embeddings.[2] The concatenation is along the channel dimension. The instruction $\bar{x} = \langle x_1, \cdots x_n \rangle$ is mapped to a sequence of hidden states $\mathbf{l}_i = \text{LSTM}_x(\psi_x(x_i), \mathbf{l}_{i-1})$, $i = 1, \ldots, n$ using a learned embedding function ψ_x and a long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997) RNN LSTM_x. The instruction representation is $\bar{\mathbf{x}} = \mathbf{l}_n$.

We generate the probability distribution P_g over pixels in $\mathbf{F}_0$ using LINGUNET. The architecture of LINGUNET is inspired by the U-NET image generation method (Ronneberger et al., 2015), except that the reconstruction phase is conditioned on the natural language instruction. LINGUNET first applies m convolutional layers to generate a sequence of feature maps $\mathbf{F}_j = \text{CNN}_j(\mathbf{F}_{j-1})$, $j = 1 \ldots m$, where each CNN_j is a convolutional layer with leaky ReLU non-linearities (Maas et al., 2013) and instance normalization (Ulyanov et al., 2016). The instruction representation $\bar{\mathbf{x}}$ is split evenly into m vectors $\{\bar{\mathbf{x}}_j\}_{j=1}^{m}$, each is used to create a 1×1 kernel $\mathbf{K}_j = \text{AFFINE}_j(\bar{\mathbf{x}}_j)$, where each AFFINE_j is an affine transformation followed by normalizing and reshaping. For each $\mathbf{F}_j$, we apply a 2D 1×1 convolution using the text kernel $\mathbf{K}_j$ to generate a text-conditioned feature map $\mathbf{G}_j = \text{CONVOLVE}(\mathbf{K}_j, \mathbf{F}_j)$, where CONVOLVE convolves the kernel over the feature map. We then perform m deconvolutions to generate a sequence of feature maps $\mathbf{H}_m,\ldots,\mathbf{H}_1$:

$$\begin{aligned}\mathbf{H}_m &= \text{DECONV}_m(\text{DROPOUT}(\mathbf{G}_m))\\ \mathbf{H}_j &= \text{DECONV}_j([\mathbf{H}_{j+1}; \mathbf{G}_j])\ .\end{aligned}$$

DROPOUT is dropout regularization (Srivastava et al., 2014) and each DECONV_j is a deconvolution operation followed a leaky ReLU nonlinearity and instance norm.[3] Finally, we generate P_g by applying a softmax to $\mathbf{H}_1$ and an additional learned scalar bias term b_g to represent events where the goal is out of sight. For example, when the agent already stands in the goal position and therefore the panorama does not show it.

We use P_g to predict the goal position in the environment. We first select the goal pixel in $\mathbf{F}_0$ as the pixel corresponding to the highest probability element in P_g. We then identify the corresponding 3D location l_g in the environment using backward camera projection, which is computed given the

---

[2] We generate $\mathbf{F}^p$ by creating a channel for each deterministic observation used to create the panorama, and setting all the pixels corresponding to that observation location in the panorama to 1 and all others to 0. The number of observations depends on the agent's camera angle.

[3] DECONV_1 does deconvolution only.

camera parameters and p_1, the agent pose at the beginning of the execution.

Action Generation Given the predicted goal l_g, we generate actions using an RNN. At each time step t, given p_t, we generate the goal mask $\mathbf{M}_t$, which has the same shape as the observed image $\mathbf{I}_t$. The goal mask $\mathbf{M}_t$ has a value of 1 for each element that corresponds to the goal location l_g in $\mathbf{I}_t$. We do not distinguish between visible or occluded locations. All other elements are set to 0. We also maintain an out-of-sight flag o_t that is set to 1 if (a) l_g is not within the agent's view; or (b) the max scoring element in P_g corresponds to b_g, the term for events when the goal is not visible in $\mathbf{I}_P$. Otherwise, o_t is set to 0. We compute an action generation hidden state y_t with an RNN:

$$y_t = \mathrm{LSTM}_A\left(\textsc{Affine}_A([\textsc{Flat}(\mathbf{M}_t); o_t]), y_{t-1}\right) \quad ,$$

where $\textsc{Flat}$ flattens $\mathbf{M}_t$ into a vector, $\textsc{Affine}_A$ is a learned affine transformation with ReLU, and LSTM_A is an LSTM RNN. The previous hidden state y_{t-1} was computed when generating the previous action, and the RNN is extended gradually during execution. Finally, we compute a probability distribution over actions:

$$\begin{aligned} P(a_t \mid l_g, (\mathbf{I}_1, p_1), \ldots, (\mathbf{I}_t, p_t)) &= \\ \textsc{Softmax}(\textsc{Affine}_p([y_t; \psi_T(t)])) & \quad , \end{aligned}$$

where ψ_T is a learned embedding lookup table for the current time (Chaplot et al., 2018) and $\textsc{Affine}_p$ is a learned affine transformation.

Model Parameters The model parameters θ include the parameters of the convolutions CNN_0 and the components of $\textsc{LingUNet}$: CNN_j, $\textsc{Affine}_j$, and $\textsc{Deconv}_j$ for $j = 1, \ldots, m$. In addition we learn two affine transformations $\textsc{Affine}_A$ and $\textsc{Affine}_p$, two RNNs LSTM_x and LSTM_A, two embedding functions ψ_x and ψ_T, and the goal distribution bias term b_g. In our experiments (Section 7), all parameters are learned without external resources.

5 Learning

Our modeling decomposition enables us to choose different learning algorithms for the two parts. While reinforcement learning is commonly deployed for tasks that benefit from exploration (e.g., Peters and Schaal, 2008; Mnih et al., 2013), these methods require many samples due to their high sample complexity. However, when learning with natural language, only a relatively small number of samples is realistically available. This problem was addressed in prior work by learning in a contextual bandit setting (Misra et al., 2017) or mixing reinforcement and supervised learning (Xiong et al., 2018). Our decomposition uniquely offers to tease apart the language understanding problem and address it with supervised learning, which generally has lower sample complexity. For action generation though, where exploration can be autonomous, we use policy gradient in a contextual bandit setting (Misra et al., 2017).

We assume access to training data with N examples $\{(\bar{x}^{(i)}, s_1^{(i)}, s_g^{(i)})\}_{i=1}^N$, where $\bar{x}^{(i)}$ is an instruction, $s_1^{(i)}$ is a start state, and $s_g^{(i)}$ is the goal state. We train the goal prediction component by minimizing the cross-entropy of the predicted distribution with the gold-standard goal distribution. The gold-standard goal distribution is a deterministic distribution with probability one at the pixel corresponding to the goal location if the goal is in the field of view, or probability one at the extra out-of-sight position otherwise. The gold location is the agent's location in $s_g^{(i)}$. We update the model parameters using Adam (Kingma and Ba, 2014).

We train action generation by maximizing the expected immediate reward the agent observes while exploring the environment. The objective for a single example i and time stamp t is:

$$J = \sum_{a \in \mathcal{A}} \pi(a \mid \tilde{s}_t) R^{(i)}(s_t, a) + \lambda H(\pi(. \mid \tilde{s}_t)) \quad ,$$

where $R^{(i)} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is an example-specific reward function, $H(\cdot)$ is an entropy regularization term, and λ is the regularization coefficient. The reward function $R^{(i)}$ details are described in details in Appendix B. Roughly speaking, the reward function includes two additive components: a problem reward and a shaping term (Ng et al., 1999). The problem reward provides a positive reward for successful task completion, and a negative reward for incorrect completion or collision. The shaping term is positive when the agent gets closer to the goal position, and negative if it is moving away. The gradient of the objective is:

$$\begin{aligned} \nabla J &= \sum_{a \in \mathcal{A}} \pi(a \mid \tilde{s}_t) \nabla \log \pi(a \mid \tilde{s}_t) R(s_t, a) \\ &\quad + \lambda \nabla H(\pi(. \mid \tilde{s}_t) \quad . \end{aligned}$$

We approximate the gradient by sampling an action using the policy (Williams, 1992), and use the gold goal location computed from $s_g^{(i)}$. We perform several parallel rollouts to compute gradients and update the parameters using Hogwild! (Recht et al., 2011) and Adam learning rates.

Dataset Statistic	Lani	Chai
Number paragraphs	6,000	1,596
Mean instructions per paragraph	4.7	7.70
Mean actions per instruction	24.6	54.5
Mean tokens per instruction	12.1	8.4
Vocabulary size	2,292	1,018

Table 1: Summary statistics of the two corpora.

6 Tasks and Data

6.1 Lani

The goal of Lani is to evaluate how well an agent can follow navigation instructions. The agent task is to follow a sequence of instructions that specify a path in an environment with multiple landmarks. Figure 1 (left) shows an example instruction.

The environment is a fenced, square, grass field. Each instance of the environment contains between 6–13 randomly placed landmarks, sampled from 63 unique landmarks. The agent can take four types of discrete actions: `FORWARD`, `TURNRIGHT`, `TURNLEFT`, and `STOP`. The field is of size 50×50, the distance of the `FORWARD` action is 1.5, and the turn angle is 15°. The environment simulator is implemented in Unity3D. At each time step, the agent performs an action, observes a first person view of the environment as an RGB image, and receives a scalar reward. The simulator provides a socket API to control the agent and the environment.

Agent performance is evaluated using two metrics: task completion accuracy, and stop distance error. A task is completed correctly if the agent stops within an aerial distance of 5 from the goal.

We collect a corpus of navigation instructions using crowdsourcing. We randomly generate environments, and generate one reference path for each environment. To elicit linguistically interesting instructions, reference paths are generated to pass near landmarks. We use Amazon Mechanical Turk, and split the annotation process to two tasks. First, given an environment and a reference path, a worker writes an instruction paragraph for following the path. The second task requires another worker to control the agent to perform the instructions and simultaneously mark at each point what part of the instruction was executed. The recording of the second worker creates the final data of segmented instructions and demonstrations. The generated reference path is displayed in both tasks. The second worker could also mark the paragraph as invalid. Both tasks are done from an overhead view of the environment, but workers are instructed to provide instructions for a robot that observes the environment from a first person view. Figure 3 shows a reference path and the written instruction. This data can be used for evaluating both executing sequences of instructions and single instructions in isolation.

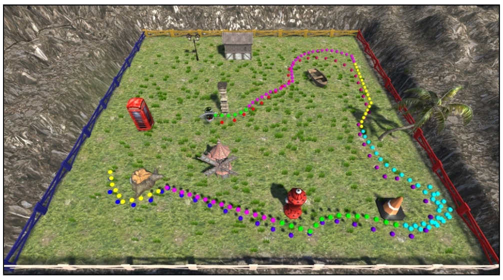

[*Go around the pillar on the right hand side*] [*and head towards the boat, circling around it clockwise.*] [*When you are facing the tree, walk towards it, and the pass on the right hand side,*] [*and the left hand side of the cone. Circle around the cone,*] [*and then walk past the hydrant on your right,*] [*and the the tree stump.*] [*Circle around the stump and then stop right behind it.*]

Figure 3: Segmented instructions in the Lani domain. The original reference path is marked in red (start) and blue (end). The agent, using a drone icon, is placed at the beginning of the path. The follower path is coded in colors to align to the segmented instruction paragraph.

Table 1 shows the corpus statistics.[4] Each paragraph corresponds to a single unique instance of the environment. The paragraphs are split into train, test, and development, with a 70% / 15% / 15% split. Finally, we sample 200 single development instructions for qualitative analysis of the language challenge the corpus presents (Table 2).

6.2 Chai

The Chai corpus combines both navigation and simple manipulation in a complex, simulated household environment. We use the Chalet simulator (Yan et al., 2018), a 3D house simulator that provides multiple houses, each with multiple rooms. The environment supports moving between rooms, picking and placing objects, and opening and closing cabinets and similar containers. Objects can be moved between rooms and in and out of containers. The agent observes the world in first-person view, and can take five actions: `FORWARD`, `TURNLEFT`, `TURNRIGHT`, `STOP`, and `INTERACT`. The `INTERACT` action acts on objects. It takes as argument a 2D position in the agent's view. Agent performance is evaluated with two metrics: (a) stop distance, which measures the distance of the agent's final state to the final annotated position; and (b) manipulation accuracy, which compares the set of manipulation actions

---

[4] Appendix A provides statistics for related datasets.

Category	Count		Example
	LANI	CHAI	
Spatial relations between locations	123	52	LANI: *go to the* ***right side of the*** *rock* CHAI: *pick up the cup* ***next to the*** *bathtub and place it on . . .*
Conjunctions of two more locations	36	5	LANI: *fly between* ***the mushroom and the yellow cone*** CHAI: *. . . set it on the table next to* ***the juice and milk****.*
Temporal coordination of sub-goals	65	68	LANI: *at the mushroom* ***turn right and move forward towards the statue*** CHAI: ***go back to*** *the kitchen* ***and put the glass in the sink****.*
Constraints on the shape of trajectory	94	0	LANI: *go past the house* ***by the right side of the apple***
Co-reference	32	18	LANI: *turn around* ***it*** *and move in front of fern plant* CHAI: *turn left, towards the kitchen door and move through* ***it****.*
Comparatives	2	0	LANI: *. . . the small stone* ***closest to the blue and white fences*** *stop*

Table 2: Qualitative analysis of the LANI and CHAI corpora. We sample 200 single development instructions from each corpora. For each category, we count how many examples of the 200 contained it and show an example.

Scenario

You have several hours before guests begin to arrive for a dinner party. You are preparing a wide variety of meat dishes, and need to put them in the sink. In addition, you want to remove things in the kitchen, and bathroom which you don't want your guests seeing, like the soaps in the bathroom, and the dish cleaning items. You can put these in the cupboards. Finally, put the dirty dishes around the house in the dishwasher and close it.

Written Instructions

[*In the kitchen, open the cupboard above the sink.*] [*Put the cereal, the sponge, and the dishwashing soap into the cupboard above the sink.*] [*Close the cupboard.*] [*Pick up the meats and put them into the sink.*] [*Open the dishwasher, grab the dirty dishes on the counter, and put the dishes into the dishwasher.*]

Figure 4: Scenario and segmented instruction from the CHAI corpus.

to a reference set. When measuring distance, to consider the house plan, we compute the minimal aerial distance for each room that must be visited. Yan et al. (2018) provides the full details of the simulator and evaluation. We use five different houses, each with up to six rooms. Each room contains on average 30 objects. A typical room is of size 6×6. We set the distance of `FORWARD` to 0.1, the turn angle to 90°, and divide the agent's view to a 32×32 grid for the `INTERACT` action.

We collected a corpus of navigation and manipulation instructions using Amazon Mechanical Turk. We created 36 common household scenarios to provide a familiar context to the task.[5] We use two crowdsourcing tasks. First, we provide workers with a scenario and ask them to write instructions. The workers are encouraged to explore the environment and interact with it. We then segment the instructions to sentences automatically. In the second task, workers are presented with the segmented sentences in order and asked to execute them. After finishing a sentence, the workers request the next sentence. The workers do not see the original scenario. Figure 4 shows a scenario and the written segmented paragraph. Similar to LANI, CHAI data can be used for studying complete paragraphs and single instructions.

Table 1 shows the corpus statistics.[6] The paragraphs are split into train, test, and development, with a 70% / 15% / 15% split. Table 2 shows qualitative analysis of a sample of 200 instructions.

7 Experimental Setup

Method Adaptations for CHAI We apply two modifications to our model to support intermediate goal for the CHAI instructions. First, we train an additional RNN to predict the sequence of intermediate goals given the instruction only. There are two types of goals: `NAVIGATION`, for action sequences requiring movement only and ending with the `STOP` action; and `INTERACTION`, for sequence of movement actions that end with an `INTERACT` action. For example, for the instruction *pick up the red book and go to the kitchen*, the sequence of goals will be $\langle$`INTERACTION`, `NAVIGATION`, `NAVIGATION`$\rangle$. This indicates the agent must first move to the object to pick it up via interaction, move to the kitchen door, and finally move within the kitchen. The process of executing an instruction starts with predicting the sequence of goal types. We call our model (Section 4) separately for each goal type. The execution concludes when the final goal is completed. For learning, we create a separate example for each intermediate goal and train the additional RNN separately. The second modification is replacing the backward camera projection for inferring the goal location with ray casting to iden-

---

[5] We observed that asking workers to simply write instructions without providing a scenario leads to combinations of repetitive instructions unlikely to occur in reality.

[6] The number of actions per instruction is given in the more fine-grained action space used during collection. To make the required number of actions smaller, we use the more coarse action space specified.

Method	Lani		Chai	
	SD	TC	SD	MA
Stop	15.37	8.20	2.99	37.53
RandomWalk	14.80	9.66	2.99	28.96
MostFrequent	19.31	2.94	3.80	37.53
Misra17	10.54	22.9	2.99	32.25
Chaplot18	9.05	31.0	2.99	37.53
Our Approach (OA)	**8.65**	**35.72**	**2.75**	37.53
OA w/o RNN	9.21	31.30	3.75	37.43
OA w/o Language	10.65	23.02	3.22	37.53
OA w/joint	11.54	21.76	2.99	36.90
OA w/oracle goals	2.13	94.60	2.19	41.07

Table 3: Performance on the development data.

Method	Lani		Chai	
	SD	TC	SD	MA
Stop	15.18	8.29	3.59	39.77
RandomWalk	14.63	9.76	3.59	33.29
MostFrequent	19.14	3.15	4.36	39.77
Misra17	10.23	23.2	3.59	36.84
Chaplot18	8.78	31.9	3.59	39.76
Our Approach	**8.43**	**36.9**	**3.34**	**39.97**

Table 4: Performance on the held-out test dataset.

Method	Lani		Chai	
	Dist	Acc	Dist	Acc
Center	12.0	19.51	3.41	19.0
Janner et al. (2018)	9.61	30.26	2.81	28.3
Our Approach	8.67	35.83	2.12	40.3

Table 5: Development goal prediction performance. We measure distance (Dist) and accuracy (Acc).

tify `INTERACTION` goals, which are often objects that are not located on the ground.

Baselines We compare our approach against the following baselines: (a) Stop: Agent stops immediately; (b) RandomWalk: Agent samples actions uniformly until it exhausts the horizon or stops; (c) MostFrequent: Agent takes the most frequent action in the data, `FORWARD` for both datasets, until it exhausts the horizon; (d) Misra17: the approach of Misra et al. (2017); and (e) Chaplot18: the approach of Chaplot et al. (2018). We also evaluate goal prediction and compare to the method of Janner et al. (2018) and a Center baseline, which always predict the center pixel. Appendix C provides baseline details.

Evaluation Metrics We evaluate using the metrics described in Section 6: stop distance (SD) and task completion (TC) for Lani, and stop distance (SD) and manipulation accuracy (MA) for Chai. To evaluate the goal prediction, we report the real distance of the predicted goal from the annotated goal and the percentage of correct predictions. We consider a goal correct if it is within a distance of 5.0 for Lani and 1.0 for Chai. We also report human evaluation for Lani by asking raters if the generated path follows the instruction on a Likert-type scale of 1–5. Raters were shown the generated path, the reference path, and the instruction.

Parameters We use a horizon of 40 for both domains. During training, we allow additional 5 steps to encourage learning even after errors. When using intermediate goals in Chai, the horizon is used for each intermediate goal separately. All other parameters and detailed in Appendix D.

8 Results

Tables 3 and 4 show development and test results. Both sets of experiments demonstrate similar trends. The low performance of Stop, RandomWalk, and MostFrequent demonstrates the challenges of both tasks, and shows the tasks are robust to simple biases. On Lani, our approach outperforms Chaplot18, improving task completion (TC) accuracy by 5%, and both methods outperform Misra17. On Chai, Chaplot18 and Misra17 both fail to learn, while our approach shows an improvement on stop distance (SD). However, all models perform poorly on Chai, especially on manipulation (MA).

To isolate navigation performance on Chai, we limit our train and test data to instructions that include navigation actions only. The Stop baseline on these instructions gives a stop distance (SD) of 3.91, higher than the average for the entire data as these instructions require more movement. Our approach gives a stop distance (SD) of 3.24, a 17% reduction of error, significantly better than the 8% reduction of error over the entire corpus.

We also measure human performance on a sample of 100 development examples for both tasks. On Lani, we observe a stop distance error (SD) of 5.2 and successful task completion (TC) 63% of the time. On Chai, the human distance error (SD) is 1.34 and the manipulation accuracy is 100%. The imperfect performance demonstrates the inherent ambiguity of the tasks. The gap to human performance is still large though, demonstrating that both tasks are largely open problems.

The imperfect human performance raises questions about automated evaluation. In general, we observe that often measuring execution quality with rigid goals is insufficient. We conduct a human evaluation with 50 development examples from Lani rating human performance and our approach. Figure 5 shows a histogram of the ratings. The mean rating for human followers is 4.38, while our approach's is 3.78; we observe a similar trend to before with this metric. Using

Category	Present	Absent	p-value
Spatial relations	8.75	10.09	.262
Location conjunction	10.19	9.05	.327
Temporal coordination	11.38	8.24	.015
Trajectory constraints	9.56	8.99	.607
Co-reference	12.88	8.59	.016
Comparatives	10.22	9.25	.906

Table 6: Mean goal prediction error for LANI instructions with and without the analysis categories we used in Table 2. The p-values are from two-sided t-tests comparing the means in each row.

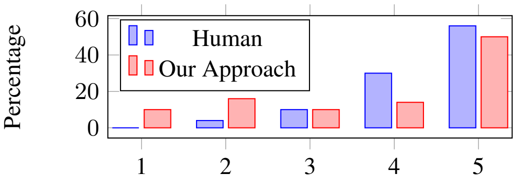

Figure 5: Likert rating histogram for expert human follower and our approach for LANI.

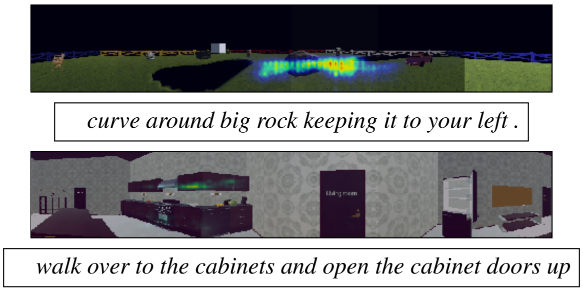
curve around big rock keeping it to your left .

walk over to the cabinets and open the cabinet doors up

Figure 6: Goal prediction probability maps P_g overlaid on the corresponding observed panoramas $\mathbf{I}_P$. The top example shows a result on LANI, the bottom on CHAI.

judgements on our approach, we correlate the human metric with the SD measure. We observe a Pearson correlation -0.65 (p=5e-7), indicating that our automated metric correlates well with human judgment.[7] This initial study suggests that our automated evaluation is appropriate for this task.

Our ablations (Table 3) demonstrate the importance of each of the components of the model. We ablate the action generation RNN (w/o RNN), completely remove the language input (w/o Language), and train the model jointly (w/joint Learning).[8] On CHAI especially, ablations results in models that display ineffective behavior. Of the ablations, we observe the largest benefit from decomposing the learning and using supervised learning for the language problem.

We also evaluate our approach with access to oracle goals (Table 3). We observe this improves navigation performance significantly on both tasks. However, the model completely fails to learn a reasonable manipulation behavior for CHAI. This illustrates the planning complexity of this domain. A large part of the improvement in measured navigation behavior is likely due to eliminating much of the ambiguity the automated metric often fails to capture.

Finally, on goal prediction (Table 5), our approach outperforms the method of Janner et al. (2018). Figure 6 and Appendix Figure 7 show example goal predictions. In Table 6, we break down LANI goal prediction results for the analysis categories we used in Table 2 using the same sample of the data. Appendix E includes a similar table for CHAI. We observe that our approach finds instructions with temporal coordination or co-reference challenging. Co-reference is an expected limitation; with single instructions, the model can not resolve references to previous instructions.

9 Discussion

We propose a model for instruction following with explicit separation of goal prediction and action generation. Our representation of goal prediction is easily interpretable, while not requiring the design of logical ontologies and symbolic representations. A potential limitation of our approach is cascading errors. Action generation relies completely on the predicted goal and is not exposed to the language otherwise. This also suggests a second related limitation: the model is unlikely to successfully reason about instructions that include constraints on the execution itself. While the model may reach the final goal correctly, it is unlikely to account for the intermediate trajectory constraints. As we show (Table 2), such instructions are common in our data. These two limitations may be addressed by allowing action generation access to the instruction. Achieving this while retaining an interpretable goal representation that clearly determines the execution is an important direction for future work. Another important open question concerns automated evaluation, which remains especially challenging when instructions do not only specify goals, but also constraints on how to achieve them. Our resources provide the platform and data to conduct this research.

Acknowledgments

This research was supported by NSF (CRII-1656998), Schmidt Sciences, and cloud computing credits from Microsoft. We thank John Langford, Claudia Yan, Bharath Hariharan, Noah Snavely, the Cornell NLP group, and the anonymous reviewers for their advice.

---

[7] We did not observe this kind of clear anti-correlation comparing the two results for human performance (Pearson correlation of 0.09 and p=0.52). The limited variance in human performance makes correlation harder to test.

[8] Appendix C provides the details of joint learning.

References

Anne H Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, Catherine Sotillo, Henry S. Thompson, and Regina Weinert. 1991. The HCRC map task corpus. *Language and Speech*, 34.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning compact lexicons for CCG semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1.

Yonatan Bisk, Daniel Marcu, and William Wong. 2016. Towards a dataset for human computer communication via grounded language acquisition. In *Proceedings of the AAAI Workshop on Symbiotic Cognitive Systems*.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. 2018. Gated-attention architectures for task-oriented language grounding.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the workshop on Human Language Technology*.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. *CoRR*, abs/1806.02724.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. 2017. Grounded language learning in a simulated 3D world. *CoRR*, abs/1706.06551.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9.

Michael Janner, Karthik Narasimhan, and Regina Barzilay. 2018. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics*, 6.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L. Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Joohyun Kim and Raymond Mooney. 2012. Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Nikita Kitaev and Dan Klein. 2017. Where is misty? interpreting spatial descriptors by modeling regions in space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. 2014. What are you talking about? text-to-image coreference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Jayant Krishnamurthy and T. Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the international conference on machine learning*.

James MacGlashan, Monica Babes-Vroman, Marie desJardins, Michael L. Littman, Smaranda Muresan, S Bertel Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. 2015. Grounding english commands to reward functions. In *Robotics: Science and Systems*.

Matthew MacMahon, Brian Stankiewics, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. 2016. Generation and Comprehension of Unambiguous Object Descriptions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012a. A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference on Machine Learning*.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2012b. Learning to parse natural language commands to a robot control system. In *Proceedings of the International Symposium on Experimental Robotics*.

Hongyuan Mei, Mohit Bansal, and R. Matthew Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Dipendra K. Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2016. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35.

Kumar Dipendra Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the international conference on machine learning*.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*.

Junhyuk Oh, Satinder P. Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the international conference on machine learning*.

Jan Peters and Stefan Schaal. 2008. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15.

Alane Suhr and Yoav Artzi. 2018. Situated mapping of sequential instructions to actions with single-step reward observation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. 1998. Intra-option learning about temporally abstract actions. In *Proceedings of the international conference on machine learning*.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8.

Wenhan Xiong, Xiaoxiao Guo, Mo Yu, Shiyu Chang, Bowen Zhou, and William Yang Wang. 2018. Scheduled policy optimization for natural language communication with intelligent agents. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.

Claudia Yan, Dipendra Kumar Misra, Andrew Bennett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. 2018. Chalet: Cornell house agent learning environment. *CoRR*, abs/1801.07357.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. 2016. Modeling context in referring expressions. In *Proceedings of the European Conference on Computer Vision*.

Deconvolutional Time Series Regression: A Technique for Modeling Temporally Diffuse Effects

Cory Shain
Department of Linguistics
The Ohio State University
`shain.3@osu.edu`

William Schuler
Department of Linguistics
The Ohio State University
`schuler.77@osu.edu`

Abstract

Researchers in computational psycholinguistics frequently use linear models to study time series data generated by human subjects. However, time series may violate the assumptions of these models through temporal diffusion, where stimulus presentation has a lingering influence on the response as the rest of the experiment unfolds. This paper proposes a new statistical model that borrows from digital signal processing by recasting the predictors and response as convolutionally-related signals, using recent advances in machine learning to fit latent impulse response functions (IRFs) of arbitrary shape. A synthetic experiment shows successful recovery of true latent IRFs, and psycholinguistic experiments reveal plausible, replicable, and fine-grained estimates of latent temporal dynamics, with comparable or improved prediction quality to widely-used alternatives.

1 Introduction

Time series are abundant in many naturally-occurring phenomena of interest to science, and they frequently violate the assumptions of linear modeling and its generalizations. One confound that may be widespread in psycholinguistic data is *temporal diffusion*: the dependent variable may evolve slowly in response to its inputs, with the result that a particular predictor observed at a particular time may continue to exert an influence on the response as the rest of the process unfolds. If not properly controlled for, such a confound could have a detrimental impact on parameter estimation, model interpretation, and hypothesis testing.

The problem of temporal diffusion remains largely unsolved in the general case.[1] A standard approach for handling the possibility of temporally diffuse relationships between the predictors and the response is to use spillover or lag regressors, where the observed predictor value is used to predict subsequent observations of the response (Erlich and Rayner, 1983). But this strategy has several undesirable properties. First, the choice of spillover position(s) for a given predictor is difficult to motivate empirically. Second, in experiments with variably long trials the use of relative event indices obscures potentially important details about the actual amount of time that passed between events. And third, including multiple spillover positions per predictor quickly leads to parametric explosion on realistically complex models over realistically sized data sets, especially if random effects structures are included.

As a solution to the problem of temporal diffusion, this paper proposes deconvolutional time series regression (DTSR), a technique that directly models diffusion by learning parametric *impulse response functions* (IRFs) of the predictors that mediate their relationship to the response variable over time. Parametric deconvolution is difficult in the general case because the likelihood surface depends on the choice IRF kernel, requiring the user to re-derive estimators for each unique model structure. Furthermore, arbitrary IRF kernels are not guaranteed to afford analytical estimator functions or unique real-valued solutions. However, recent advances in machine learning have led to libraries like Tensorflow (Abadi et al., 2015) — which uses auto-differentiation to support optimization of arbitrary computation graphs — and Edward (Tran et al., 2016) — which enables black box variational inference (BBVI) on Tensorflow graphs. While these libraries are typically used to build and train deep networks, DTSR uses them

---

[1] Although recent work in computational psycholinguistics has begun to address separate but related problems in time series modeling (auto-correlation and non-stationarity) using generalized additive models (GAM) with a particular structure (Baayen et al., 2017, 2018).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2679–2689
Brussels, Belgium, October 31 - November 4, 2018.

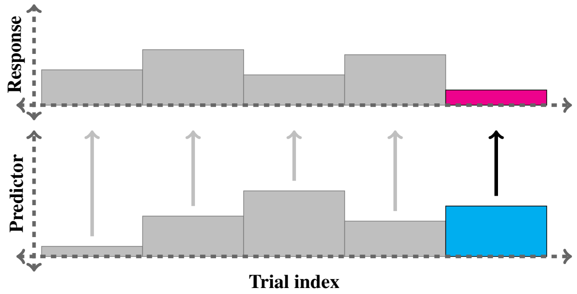

Figure 1: Effects in a linear time series model

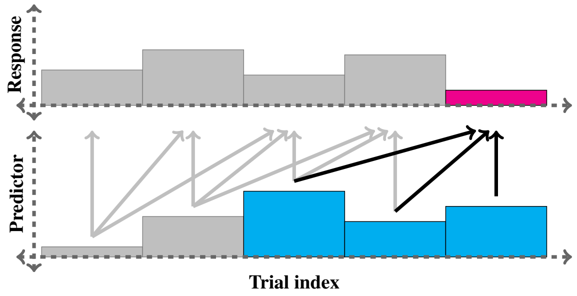

Figure 2: Linear time series model with spillover

to overcome the aforementioned difficulties with general-purpose temporal deconvolution by eliminating the need for hand-derivation of estimators and sampling distributions for each model.

The IRFs learned by DTSR are interpretable as estimates of the temporal shape of predictors' influence on the response variable. By convolving predictors with their IRFs, DTSR is able to consider arbitrarily long histories of independent variable observations in generating a given prediction, and (in contrast to spillover) model complexity is constant on the length of the history window. DTSR is thus a parsimonious technique for directly measuring temporal diffusion.

Figures 1–3 illustrate the present proposal and how it differs from linear time series models. As shown in Figure 1, a standard linear model assumes conditional independence of the response from all preceding observations of the predictor. This independence assumption can be weakened by including additional spillover predictors (Figure 2), at a cost of requiring additional parameters. In both cases, only the relative order of events is considered, not their actual distance in time. By contrast, DTSR recasts the predictor and response vectors as streams of impulses and responses (respectively) localized in time. It then fits latent IRFs that govern the influence of each predictor value on the response as a function of time (Figure 3).

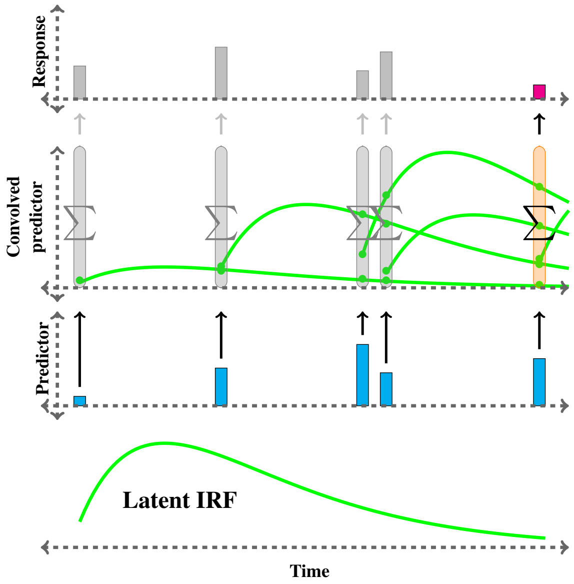

Figure 3: Effects of predictors in DTSR

This paper presents evidence that DTSR can (1) recover known underlying IRFs from synthetic data, (2) discover previously unknown temporal structure in human data (psycholinguistic reading time experiments), (3) provide support for the *absence* of temporal diffusion in settings where it might exist in principle, and (4) provide comparable (or in some cases improved) prediction quality to standard linear mixed-effects (LME) and generalized additive (GAM) models.

2 Related work

2.1 Non-deconvolutional time series modeling

The two most widely used tools for analyzing psycholinguistic time series are linear mixed effects regression (LME) (Bates et al., 2015) and generalized additive models (GAM) (Hastie and Tibshirani, 1986; Wood, 2006). LME learns a linear combination of the predictors that generates a given response variable. GAM generalizes linear models by allowing the response variable to be computed as the sum of smooth functions of one or more predictors.

In both approaches, responses are modeled as conditionally independent of preceding observations of predictors unless spillover terms are added, with the attendant drawbacks discussed in Section 1. To make this point more forcefully, take for example Shain et al. (2016), who find significant effects of constituent wrap-up ($p = 2.33\text{e-}14$) and dependency locality ($p = 4.87\text{e-}10$) in the

Natural Stories self-paced reading corpus (Futrell et al., 2018). They argue that this constitutes the first strong evidence of memory effects in broad-coverage sentence processing. However, it turns out that when one baseline predictor — probabilistic context free grammar (PCFG) surprisal — is spilled over one position, the reported effects disappear: p = 0.816 for constituent wrap-up and p = 0.370 for dependency locality. Thus, a reasonable but ultimately inaccurate assumption about baseline effect timecourses can have a dramatic impact on the conclusions supported by the statistical model. DTSR offers a way forward by bringing temporal diffusion under direct statistical control.

2.2 Deconvolutional time series modeling

Deconvolutional modeling has long been used in a variety of scientific fields, including economics (Ramey, 2016), epidemiology (Goldstein et al., 2011), and neuroimaging (Friston et al., 1998). Non-parametric deconvolutional models quantize the time series and fit estimates for each time point within some window, similarly to the spillover approach discussed above. These estimates can be unconstrained, as in finite impulse response models (FIR) (H. Glover, 1999; Ward, 2006), or smoothed with some form of regularization (Goutte et al., 2000; Pedregosa et al., 2014). Additional post-hoc interpolation is necessary in order to obtain a closed-form continuous IRF. These non-parametric approaches are prone to parametric explosion as well as sparsity problems when trials are variably spaced in time.

Parametric deconvolutional approaches (i.e. specific instantiations of DTSR) have evolved in certain fields (e.g. fMRI modeling) to solve particular problems, generally with some independently-motivated IRF kernel like the hemodynamic response function (HRF) (Friston et al., 1998; Lindquist and Wager, 2007; Lindquist et al., 2009). However, to our knowledge DTSR constitutes the first mathematical formulation and software implementation of general-purpose mixed effects parametric deconvolutional regression for arbitrary impulse response kernels. DTSR also supports Bayesian inference, enabling quantification of uncertainty in the absence of analytic formulae for standard errors. With these properties, DTSR expands the range of possible applications of parametric deconvolution beyond those fields for which appropriate formulations have already been developed.

3 Model definition

This section presents the mathematical definition of DTSR. For readability, only a fixed effects model is presented below, since mixed modeling substantially complicates the equations. The full model definition is provided in Appendix A. Note that the full definition is used to construct all reading time models reported in subsequent sections, since they contain random effects.

Let $\mathbf{X} \in \mathbb{R}^{M \times K}$ be a design matrix of M observations for K predictor variables and $\mathbf{y} \in \mathbb{R}^N$ be a vector of N responses, both of which contain contiguous temporally-sorted time series. DTSR models the relationship between $\mathbf{X}$ and $\mathbf{y}$ using parameters consisting of:

- a scalar intercept $\mu \in \mathbb{R}$
- a vector $\mathbf{u} \in \mathbb{R}^K$ of K coefficients
- a matrix $\mathbf{A} \in \mathbb{R}^{R \times K}$ of R IRF kernel parameters for K fixed impulse vectors
- a scalar variance $\sigma^2 \in \mathbb{R}$ of the response

To define the convolution step, let g_k for $k \in \{1, 2, \ldots, K\}$ be a set of parametric IRF kernels, one for each predictor; let $\mathbf{a} \in \mathbb{R}^M$ and $\mathbf{b} \in \mathbb{R}^N$ be vectors of timestamps associated with each observation in $\mathbf{X}$ and $\mathbf{y}$, respectively; and let $\mathbf{c} \in \mathbb{N}^M$ and $\mathbf{d} \in \mathbb{N}^N$ be vectors of series ID's associated with each observation in $\mathbf{X}$ and $\mathbf{y}$, respectively. A filter $\mathbf{F} \in \mathbb{R}^{N \times M}$ admits only those observations in $\mathbf{X}$ that precede $\mathbf{y}_{[n]}$ in the same time series:

$$\mathbf{F}_{[n,m]} \overset{\text{def}}{=} \begin{cases} 1 & \mathbf{c}_{[m]} = \mathbf{d}_{[n]} \wedge \mathbf{a}_{[m]} \leq \mathbf{b}_{[n]} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The inputs $\mathbf{X}$ can be convolved with each IRF g_k by premultiplication with sparse matrix $\mathbf{G}_k \in \mathbb{R}^{N \times M}$ for $k \in \{1, 2, ..., K\}$ as defined below:

$$\mathbf{G}_k = g_k \left(\mathbf{b}\mathbf{1}^\top - \mathbf{1}\mathbf{a}^\top ; \mathbf{A}_{[*,k]} \right) \odot \mathbf{F} \tag{2}$$

The convolution that yields the design matrix of convolved predictors $\mathbf{X}' \in \mathbb{R}^{N \times K}$ is then defined using products of the $\mathbf{G}$ matrices and the design matrix $\mathbf{X}$:[2]

$$\mathbf{X}'_{[*,k]} \overset{\text{def}}{=} \mathbf{G}_k \, \mathbf{X}_{[*,k]} \tag{3}$$

---

[2] This implementation of convolution is only exact when the predictors fully describe a discrete impulse signal. Exact convolution of samples from continuous signals is generally not possible because the signal is generally not analytically integrable. For continuous signals, DTSR can approximate the convolution as long as the predictor is interpolated between sample points at a fixed frequency prior to fitting.

Following convolution, DTSR is simply a linear model. The full model mean is the sum of (1) the intercept μ and (2) the product of the convolved predictor matrix $\mathbf{X}'$ and the coefficient vector $\mathbf{u}$:

$$\mathbf{y} \sim \mathcal{N}\left(\mu + \mathbf{X}'\mathbf{u}, \sigma^2\right) \tag{4}$$

4 Implementation

The present implementation defines the aforementioned equations[3] as a Bayesian computation graph in Tensorflow and Edward and trains it with black box variation inference (BBVI) using the Nadam optimizer (Dozat, 2016)[4] with a constant learning rate of 0.01 and minibatches of size 1024. For computational efficiency, histories are truncated at 128 timesteps. Prediction from the network uses an exponential moving average of parameter iterates with a decay rate of 0.998. Convergence was visually diagnosed.

The present experiments use a ShiftedGamma IRF kernel:

$$f(x;\alpha,\beta,\delta) = \frac{\beta^{\alpha}(x-\delta)^{\alpha-1}e^{-\beta(x-\delta)}}{\Gamma(\alpha)} \tag{5}$$

This is simply the PDF of the Gamma distribution augmented with a shift parameter δ allowing the lower bound of the support of the distribution to deviate from 0. We constrain δ to be strictly negative, thereby allowing the model to find a non-zero instantaneous response. We also constrain k to be strictly greater than 1, which deconfounds the shape and shift parameters. All bounded variables are constrained using the softplus bijection:

$$\text{softplus}(x) = \log(e^x + 1) \tag{6}$$

The ShiftedGamma kernel is used here because it can fit a wide range of response shapes and has precedent in the fMRI literature, where HRF kernels are often assumed to be Gamma-shaped (Lindquist et al., 2009).[5]

All parameters are given normal priors with unit variance. Prior means for the fixed IRF kernel parameters are domain-specific and discussed in the experiments sections below. To center the prior at an intercept-only model,[6] prior means for the intercept μ and variance σ^2 are set (respectively) to the empirical mean and variance of the response, and prior means for both fixed coefficients and random effects[7] are set to 0. Although the Bayesian implementation of DTSR is used for this study because it provides quantification of uncertainty, placing priors on the IRF kernel parameters is not crucial to the success of the system. In all experiments reported below, the MLE implementation arrives at similar solutions and achieves slightly better error.

In the interests of enabling the use of DTSR by the scientific community, the implementation of DTSR used here is offered as a documented open-source Python package with support for (1) Bayesian, variational Bayesian, and MLE inferences and (2) a variety model structures and impulse response kernels. The Tensorflow backend also enables GPU acceleration where available. Source code and links to documentation are available at `https://github.com/coryshain/dtsr`.

5 Experiment 1: Synthetic data

An initial experiment fits DTSR estimates to synthetic data to determine whether the model can recover known ground truth IRFs. Synthetic data were synthesized using the following procedure. First, 20 input vectors of size 10,000 were drawn from a standard normal distribution. These values synthesize an impulse stream containing 20 covariates, each with 10,000 observations. A ShiftedGamma IRF was then drawn for each of the 20 covariates. Coefficients were drawn from a uniform distribution $\mathcal{U}(-50, 50)$, and IRF parameters were drawn from the following distributions: $\alpha \sim \mathcal{U}(1, 6)$, $\beta \sim \mathcal{U}(0, 5)$, $\delta \sim \mathcal{U}(-1, 0)$. The prior means for the corresponding IRF kernel parameters are placed at the centers of these ranges. The stream of responses was generated by convolving the covariates with their corresponding IRFs. Gaussian noise with standard deviation 20 was injected into the response following generation. The 10,000 trials were spaced 100ms apart. As shown in Figure 4, the DTSR estimates for the

---

[3] As noted above, for expository purposes the definition in Section 3 only supports fixed-effects models. The full definition for mixed-effects DTSR models is provided in Appendix A. Mixed models are used throughout the experiments reported below.

[4] The Adam optimizer (Kingma and Ba, 2014) with Nesterov momentum (Nesterov, 1983)

[5] Other IRF kernels, including spline functions and composition of convolutions, are supported by the current implementation of DTSR but are not explored in these experiments. More details are provided in the software documentation.

[6] A model in which the response is insensitive to the model structure.

[7] See Appendix A for the definition of the mixed-effects DTSR model, which includes random effects.

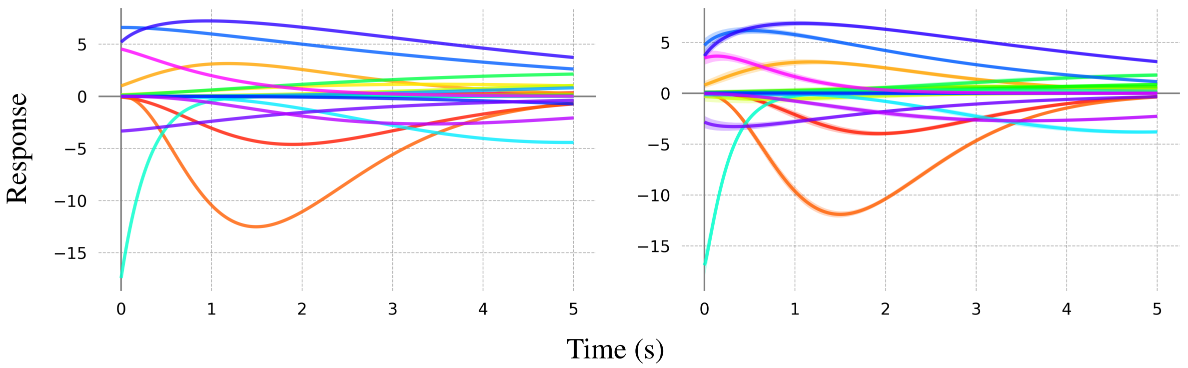

Figure 4: *Synthetic data*. True IRFs (left) and estimated IRFs with 95% credible intervals (right).

synthetic data are very similar to the ground truth, confirming that when the data-generating model matches the assumptions of DTSR, DTSR can recover its latent structure with high fidelity.

6 Experiment 2: Human reading times

6.1 Background and experimental design

The main interest of DTSR is the potential to better understand real-world dynamical systems like the human sentence processing response. Therefore, Experiment 2 applies DTSR to three existing datasets of naturalistic reading: Natural Stories (Futrell et al., 2018), Dundee (Kennedy et al., 2003), and UCL (Frank et al., 2013).

Natural Stories is a self-paced reading (SPR) corpus consisting of narratives designed to provide context-rich, fluent-sounding stimuli that nonetheless contain many grammatical constructions that rarely occur naturally in texts. The public release of the corpus contains data collected from 181 subjects. The stimulus set contains 10 stories with a total of 485 sentences and 10,245 tokens, for a total 848,768 fixation events.

Dundee is an eye-tracking (ET) corpus containing newspaper editorials read by 10 subjects, with incremental eye fixation data recorded during reading. The stimulus set contains 20 editorials with a total of 2,368 sentences and 51,502 tokens, for a total of 260,065 fixation events.

UCL is a reading corpus containing individual sentences that were extracted from novels written by amateur authors. The sentences were shuffled and presented in isolation to 42 subjects. The eye-tracking portion of the UCL corpus used in these experiments contains 205 sentences with a total of 1,931 tokens, for a total of 53,070 fixation events.

In all experiments, the response variable is log fixation duration (go-past duration for ET). Models use the following set of predictor variables in common use in psycholinguistics: *Sentence position* (index of word in sentence), *Trial* (index of trial in series),[8] *Saccade Length* (in words, ET only), *Word Length* (in characters), *Unigram Logprob*, and *5-gram Surprisal*. *Unigram Logprob* and *5-gram Surprisal* are computed by the KenLM toolkit (Heafield et al., 2013) trained on Gigaword 4 (Parker et al., 2009). In addition, DTSR enables fitting of a *Rate* predictor, which is simply a vector of ones, one for each observation, that is convolved using a latent IRF. *Rate* thus measures the response to density of stimulus presentation in the recent past. Since without deconvolution *Rate* is identical to the intercept, it is excluded from non-deconvolutional baseline models. Following standard practice in psycholinguistics, by-subject random coefficients for each of these predictors are included in all models (baseline and DTSR).[9]

ShiftedGamma IRFs are fitted to all predictors except *Sentence Position*, which is assigned a Dirac delta IRF (i.e. a linear coefficient) since it increases linearly within the sentence and is not expected to have a diffuse response. In plots, the *Sentence Position* estimate is shown as a stick function at time 0s. Prior means used for the IRF kernel parameters are $\alpha = 2$, $\beta = 5$, and $\delta = -0.5$. Together, these priors define an expected exponential-like IRF which decays to near-zero in about 1s, which seems plausible for human reading times. In practice they do not appear to be very constraining, since posterior means of fitted

---

[8] Except UCL, which contains isolated sentences, in which case *Trial* is identical to *Sentence Position*.

[9] By-subject IRF parameters were not used for this study because they substantially complicate the model and initial experiments using them showed little benefit on training data.

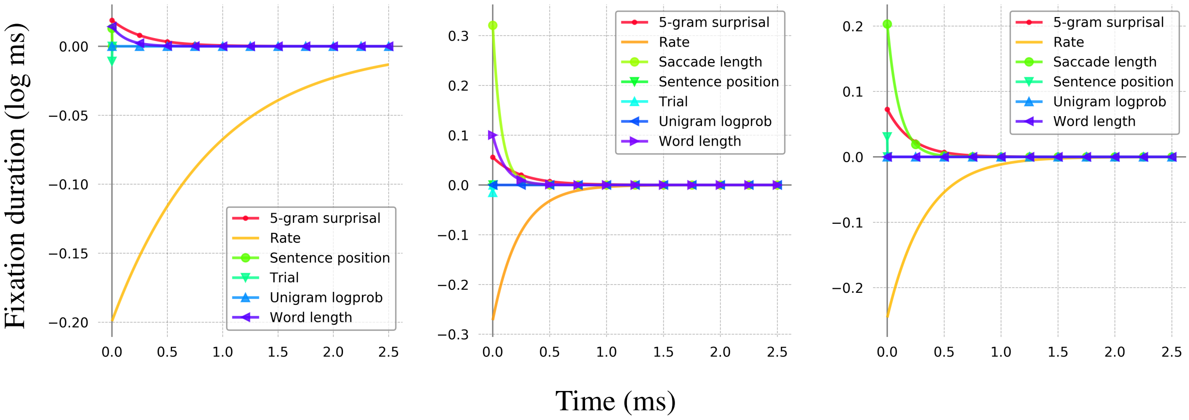

Figure 5: *Human data.* Estimated IRFs with 95% credible intervals for Natural Stories (left), Dundee (center) and UCL (right). Intervals are too tight to be seen.

models often deviate quite far from these values.

Existing work provides some expectations about the relationships of these variables to reading time. Processing difficulty is expected to increase with *Saccade Length*, *Word Length*, and *5-gram Surprisal*, and positive linear relationships have been shown experimentally (Demberg and Keller, 2008). *Unigram Logprob* is expected to be negatively correlated with reading times, since more frequent words are expected to be easier to process. *Sentence Position*, *Trial*, and *Rate* index different kinds of change in the response over time and their relationship has not been carefully studied, in part for lack of deconvolutional regression tools. Although reading times tend to decrease over the course of the experiment (Baayen et al., 2018), suggesting an expected negative effect of *Trial*, this may be partially explained by temporal diffusion. For the present study, all predictors are rescaled by their standard deviations.[10]

In all reading experiments, data are partitioned into training (50%), development (25%) and test (25%) sets. Outlier filtering is also performed. For Natural Stories, following Shain et al. (2016), items are excluded if they have fixations shorter than 100ms or longer than 3000ms, if they start or end a sentence, or if subjects missed 4 or more subsequent comprehension questions. For Dundee, following van Schijndel and Schuler (2015), unfixated items are excluded as well as (1) items following saccades longer than 4 words and (2) starts and ends of sentences, screens, documents, and lines. For UCL, unfixated items are excluded as well as (1) items following saccades longer than 4 words and (2) sentence starts and ends. Partitioning and filtering are applied only to the response series. The entire predictor history remains visible to the model.

From a modeling perspective, the primary results of interest in Experiment 2 are the IRFs themselves and the insights they provide into human sentence processing. However, to check the reliability of the DTSR estimates, prediction quality on unseen data is compared to that of non-deconvolutional baseline models fitted with LME and GAM.[11] Both baselines are fitted with and without three preceding spillover positions for each predictor (baselines with spillover are designated throughout this paper with the suffix -*S*).[12]

6.2 Results

The fitted IRFs for Natural Stories, Dundee, and UCL are shown in Figure 5. Effect sizes by corpus — computed here as the integral of each IRF over the first 10s — are shown in Table 1, along with

---

[10] Except *Rate*, which has no variance and therefore cannot be scaled by its standard deviation of 0.

[11] Formulae used to construct each model reported in this study are available in the associated code repository.

[12] This number of spillover positions is among the largest attested in the psycholinguistic literature because model complexity in LME and GAM increases substantially with each spillover position added, especially when by-subject random slopes are included for each spillover position for each variable. Indeed, many of the baseline models run for these experiments are already at the limits of tractability, as shown by the non-convergence reported in certain cells of Table 2. An advantage of the DTSR approach is that it can consider arbitrarily long histories at no cost to model complexity. While this permits DTSR to consider longer histories than its competitors (in these experiments, 128 timepoints vs. 4), DTSR is more constrained in its use of history since it must apply the same set of IRFs to all datapoints, while the baselines essentially fit separate models for each spillover position.

	Natural Stories			Dundee			UCL		
Predictor	Mean	2.5%	97.5%	Mean	2.5%	97.5%	Mean	2.5%	97.5%
Trial	-0.0053	-0.0057	-0.0049	-0.0085	-0.0010	-0.0071	—	—	—
Sent pos	0.0154	0.0148	0.0160	0.0004	-0.0013	0.0022	0.0340	0.0301	0.0379
Rate	-0.1853	-0.1858	-0.1848	-0.0649	-0.0659	-0.0640	-0.0806	-0.0832	-0.0781
Sac len	—	—	—	0.0249	0.0216	0.0207	0.0217	0.0209	0.0225
Word len	0.0020	0.0019	0.0021	0.0107	0.0105	0.0109	-8e-07	-1.7e-5	1.4e-5
Unigram	2.6e-6	-5e-6	2.2e-5	-2.0e-6	-3.9e-5	2.8e-5	1e-06	-4e-6	1.2e-5
5-gram	0.0057	0.0056	0.0059	0.0139	0.0134	0.0145	0.0159	0.0148	0.0171

Table 1: Effect sizes by corpus with 95% credible intervals based on 1024 posterior samples

95% credible intervals (CI). The IRFs (curves) in these plots represent the expected change in the response over time from observing a unit impulse of the predictor. For example, the Dundee model estimates that observing a standard deviation of *5-gram surprisal* engenders a slowdown of about 0.05 log ms instantaneously and a slowdown of about 0.03 log ms 250 ms after stimulus presentation. Because the response is reading time, positive IRFs represent inhibition and negative IRFs represent facilitation. Detailed interpretation of these curves is provided below in Section 6.3.

Table 2 shows prediction error from DTSR vs. baselines fitted to the same feature set. As shown, DTSR provides comparable or improved prediction performance to the baselines, even against the *-S* models which are more heavily parameterized. DTSR outperforms LME models on unseen data across all corpora and generally improves upon or closely matches the performance of GAM (with no spillover). Compared to GAM-S (with three additional spillover positions), there is a clear advantage of DTSR for Natural Stories but not for the eye-tracking (ET) datasets. This is likely due to more pronounced temporal confounds in Natural Stories (especially of *Rate*, which the baseline models cannot estimate) compared to the other corpora.[13] However, even in the absence of sufficiently diffuse effects to afford prediction improvements, the ability to measure diffusion directly is a major advantage of the DTSR model, since it can be used to detect the *absence* of diffusion in settings where it might in principle exist. Further discussion of the DTSR IRF estimates themselves is provided in Section 6.3.

As shown in Table 3, pooling across corpora, permutation testing reveals a significant improvement in MSE on test data of DTSR over each baseline system ($p = 0.0001$ for all comparisons).[14]

6.3 Discussion

Some key generalizations emerge from the DTSR estimates shown in Figure 5. The first is the pronounced facilitative role of *Rate* in all three models, but especially in Natural Stories. This means that fast reading in the recent past engenders fast reading in the present, because (1) observing a stimulus exerts a large-magnitude, diffuse, and negative (facilitative) influence on the subsequent response, and (2) the *Rate* contributions of the stimuli are additive. This result demonstrates an important pre-linguistic influence of *inertia* — a tendency toward slow overall change in base response rate. This effect is especially large-magnitude and diffuse in Natural Stories, which is self-paced reading and therefore differs in modality from the other datasets (which are eye-tracking). This suggests that SPR participants strongly habituate to repeated button pressing and stresses the importance of deconvolutional regression for bringing this low-level confound under control in analyzing SPR data, since it appears to have a large influence on the response and might otherwise confound model interpretation.

Second, effects are generally consistent with expectations: positive effects for *Saccade Length*, *Word Length*, and *5-gram Surprisal*, and a negative effect of *Trial*. The null influence of *Unigram Logprob* is likely due to the presence in the model of both *5-gram Surprisal* (which interpolates unigram probabilities) and *Word Length* (which is inversely correlated with *Unigram Logprob*). The biggest departure from prior expectations is the null estimate for *Word Length* in UCL. It appears

---

[13] Note that GAM-S is more heavily parameterized than DTSR in that it fits multidimensional spline functions of each spillover position of each predictor. This makes it difficult to generalize information about effect timecourses from GAM fits, motivating the use of DTSR for studies in which timecourses are a quantity of interest.

[14] To ensure comparability across corpora with different error variances, per-datum errors were first scaled by their standard deviations within each corpus. Standard deviations were computed over the joint set of error values in each pair of DTSR and baseline models.

	Natural Stories			Dundee			UCL		
System	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
LME	0.0803	0.0818	0.0815	0.2135	0.2133	0.2128	0.2613	0.2776	0.2561
LME-S	0.0789†	0.0807†	0.0804†	0.2099†	0.2103†	0.2095†	0.2509†	0.2754†	0.2557†
GAM	0.0798	0.0814	0.081	0.212	0.2116	0.2111	0.2576	0.2741	0.2538
GAM-S	0.0784	0.0802	0.0799	**0.2083**	**0.2085**	**0.2078**	**0.2440**	**0.2661**	**0.2457**
DTSR	**0.0648**	**0.0655**	**0.0650**	0.2100	0.2094	0.2088	0.2590	0.2752	0.2543

Table 2: Mean squared prediction error by system (daggers indicate convergence warnings)

Baseline	DTSR improvement (z-units)	p-value
LME	0.059	0.0001***
LME-S	0.054	0.0001***
GAM	0.057	0.0001***
GAM-S	0.051	0.0001***

Table 3: Overall pairwise significance of prediction improvement from DTSR vs. baselines

that the contribution of *Word Length* in this corpus can be effectively explained by other variables.

Third, the response estimates for Dundee and UCL (both of which are eye-tracking) are very similar, which suggests that DTSR is discovering replicable population-level features of the temporal profile for eye-tracking data.

Fourth, there is a general asymmetry in degree of diffusion between low-level perceptual-motor variables like *Saccade Length* and *Word Length*, whose responses tend to decay quickly, and the high-level *5-gram Surprisal* variable, whose response tends to decay more slowly. This is consistent with expectations from the sentence processing literature. Perceptual-motor variables involve rapid bottom-up computation (e.g. visual processing or motor planning/execution) and are therefore not expected to have a diffuse response, while surprisal involves top-down computation of future words given context, which might be more computationally expensive and therefore engender a slower response. While this outcome is suggested e.g. by the aforementioned finding that spillover 1 winds up being a stronger position for a surprisal predictor in the Shain et al. (2016) models, DTSR permits direct investigation of these dynamics.

7 A note on hypothesis testing

As a Bayesian model, DTSR supports hypothesis testing by querying the variational posterior. For example, as shown in Table 1, the credible interval (CI) for *5-gram Surprisal* in Natural Stories does not include zero (rejecting the null hypothesis of no effect), while the CI for *Unigram logprob* does (failing to reject). To control for effects of multicolinearity, one could perform ablative tests of fitted null and alternative models using (1) likelihood comparison or (2) predictive performance on unseen data.

However, DTSR estimates are obtained through non-convex stochastic optimization, which complicates hypothesis testing because of possible *estimation noise* due to (1) convergence to a local but not global optimum, (2) imperfect convergence to the local optimum, and/or (3) Monte Carlo estimation of the test statistic via posterior sampling. It cannot therefore be guaranteed that hypothesis testing results are due to differences in model structure rather than differences in relative amounts of estimation noise introduced by the fitting procedure. Thus, p-values (and, consequently, hypothesis tests) based on direct comparison of DTSR models should be considered approximate.

However, even in situations where such uncertainty in hypothesis testing is not acceptable, DTSR is appropriate for certain important use cases. First, DTSR can be used for *exploratory data analysis* in order to empirically motivate the spillover structure of the linear model. Spillover variables can be excluded or included based on the degree of temporal diffusion revealed by DTSR, permitting construction of linear models that are both parsimonious and effective for controlling temporal diffusion. Second, DTSR can be used to fit a *data transform* which is then applied to the data prior to statistical analysis. This approach is identical in spirit to e.g. the use of the canonical HRF to convolve predictors in fMRI models prior to linear regression. However, since DTSR is domain-general, it can be a valuable component in any analysis toolchain for time series.

8 Conclusion

This paper presented a variational Bayesian deconvolutional time series regression method as a solution to the problem of temporal diffusion in psycholinguistic time series data and applied it to both synthetic and human responses in order to better understand and control for latent temporal dynamics. Results showed that DTSR can yield a plausible, replicable, parsimonious, insightful, and predictive model of a complex dynamical system like the human sentence processing response and therefore support the use of DTSR for psycholinguistic time series modeling. While the present study explored the use of DTSR to understand human reading times, DTSR can in principle also be used to deconvolve other kinds of response variables, such as the HRF in fMRI modeling or the power/coherence response in oscillatory measures like electroencephalography, suggesting a rich array of potential applications of DTSR in computational psycholinguistics.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported by National Science Foundation grants #1551313 and #1816891. All views expressed are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vigas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. Tensorflow: Large-scale machine learning on heterogeneous distributed systems.

Harald Baayen, Shravan Vasishth, Reinhold Kliegl, and Douglas Bates. 2017. The cave of shadows: Addressing the human factor with generalized additive mixed models. *Journal of Memory and Language*, 94(Supplement C):206 – 234.

R. Harald Baayen, Jacolien van Rij, Cecile de Cat, and Simon Wood. 2018. Autocorrelated errors in experimental data in the language sciences: Some solutions offered by Generalized Additive Mixed Models. In Dirk Speelman, Kris Heylen, and Dirk Geeraerts, editors, *Mixed Effects Regression Models in Linguistics*. Springer, Berlin.

Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. 2015. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48.

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

Timothy Dozat. 2016. Incorporating Nesterov momentum into Adam. In *ICLR Workshop*.

Kate Erlich and Keith Rayner. 1983. Pronoun assignment and semantic integration during reading: Eye movements and immediacy of processing. *Journal of Verbal Learning & Verbal Behavior*, 22:75–87.

Stefan L. Frank, Irene Fernandez Monsalve, Robin L. Thompson, and Gabriella Vigliocco. 2013. Reading time data for evaluating broad-coverage models of English sentence processing. *Behavior Research Methods*, 45:1182–1190.

Karl J. Friston, Oliver Josephs, Geraint Rees, and Robert Turner. 1998. Nonlinear event-related responses in fMRI. *Magn. Reson. Med*, pages 41–52.

Richard Futrell, Edward Gibson, Hal Tily, Anastasia Vishnevetsky, Steve Piantadosi, and Evelina Fedorenko. 2018. The Natural Stories corpus. In *LREC 2018*.

Edward Goldstein, Benjamin J. Cowling, Allison E. Aiello, Saki Takahashi, Gary King, Ying Lu, and Marc Lipsitch. 2011. Estimating incidence curves of several infections using symptom surveillance data. *PLOS ONE*, 6(8):1–8.

C. Goutte, F. A. Nielsen, and K. H. Hansen. 2000. Modeling the hemodynamic response in fMRI using smooth FIR filters. *IEEE Transactions on Medical Imaging*, 19(12):1188–1201.

Gary H. Glover. 1999. Deconvolution of impulse response in event-related BOLD fMRI. 9:416–29.

Trevor Hastie and Robert Tibshirani. 1986. Generalized additive models. *Statist. Sci.*, 1(3):297–310.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Alan Kennedy, James Pynte, and Robin Hill. 2003. The Dundee corpus. In *Proceedings of the 12th European conference on eye movement*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Martin Lindquist and Tor Wager. 2007. Validity and power in hemodynamic response modeling: A comparison study and a new approach. 28:764–84.

Martin A. Lindquist, Ji Meng Loh, Lauren Y. Atlas, and Tor D. Wager. 2009. Modeling the hemodynamic response function in fmri: Efficiency, bias and mis-modeling. *NeuroImage*, 45(1, Supplement 1):S187 – S198. Mathematics in Brain Imaging.

Yurii E Nesterov. 1983. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2009. *English Gigaword LDC2009T13*.

Fabian Pedregosa, Michael Eickenberg, Philippe Ciuciu, Alexandre Gramfort, and Bertrand Thirion. 2014. Data-driven hrf estimation for encoding and decoding models. 104.

V.A. Ramey. 2016. Macroeconomic shocks and their propagation. volume 2 of *Handbook of Macroeconomics*, pages 71 – 162. Elsevier.

Cory Shain, Marten van Schijndel, Richard Futrell, Edward Gibson, and William Schuler. 2016. Memory access during incremental sentence processing causes reading time latency. In *Proceedings of the Computational Linguistics for Linguistic Complexity Workshop*, pages 49–58. Association for Computational Linguistics.

Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. 2016. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*.

Marten van Schijndel and William Schuler. 2015. Hierarchic syntax improves reading time prediction. In *Proceedings of NAACL-HLT 2015*. Association for Computational Linguistics.

B. Douglas Ward. 2006. Deconvolution analysis of fmri time series data.

Simon N. Wood. 2006. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, Boca Raton.

A Definition of mixed effects DTSR

For expository purposes, in Section 3 the DTSR model was defined only for fixed effects. However, DTSR is compatible with mixed modeling and the implementation used here supports random effects in the model intercepts, coefficients, and IRF parameters. The full mixed-effects DTSR equations are presented below.

The definitions of $\mathbf{X}$, $\mathbf{y}$, μ, σ^2, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$, $\mathbf{F}$, M, N, K, and R presented in Section 3 are retained for the mixed model definition. The remaining variables and equations must be redefined to some extent. Mixed-effects DTSR models additionally contain the following parameters:

- a vector $\mathbf{o} \in \mathbb{R}^O$ of O random intercepts
- a vector $\mathbf{u} \in \mathbb{R}^U$ of U fixed coefficients
- a vector $\mathbf{v} \in \mathbb{R}^V$ of V random coefficients
- a matrix $\mathbf{A} \in \mathbb{R}^{R\times L}$ of R fixed IRF kernel parameters for L fixed impulse vectors
- a matrix $\mathbf{B} \in \mathbb{R}^{R\times W}$ of R random IRF kernel parameters for W random impulse vectors

Random parameters $\mathbf{o}$, $\mathbf{v}$, and $\mathbf{B}$ are constrained to be zero-centered within each random grouping factor.

To support mixed modeling, the fixed and random effects must first be combined using additional utility matrices. Let $\mathbf{O} \in \{0,1\}^{N\times O}$ be a mask matrix for random intercepts. A vector $\mathbf{q} \in \mathbb{R}^N$ of intercepts is:

$$\mathbf{q} \overset{\text{def}}{=} \mu + \mathbf{O}\,\mathbf{o} \tag{7}$$

Let $\mathbf{U} \in \{0,1\}^{L\times U}$ be an indicator matrix for fixed coefficients, $\mathbf{V} \in \{0,1\}^{L\times V}$ be an indicator matrix for random coefficients, and $\mathbf{V}' \in \{0,1\}^{N\times V}$ be a mask matrix for random coefficients. A matrix $\mathbf{Q} \in \mathbb{R}^{N\times L}$ of coefficients is:

$$\mathbf{Q} \overset{\text{def}}{=} \mathbf{1}\,(\mathbf{U}\,\mathbf{u})^\top + \mathbf{V}'\,\mathrm{diag}(\mathbf{v})\,\mathbf{V}^\top \tag{8}$$

Let $\mathbf{W} \in \{0,1\}^{L\times W}$ be an indicator matrix for random IRF parameters and $\mathbf{W}'_1, \dots, \mathbf{W}'_n \in \{0,1\}^{R\times W}$ be mask matrices for random IRF parameters. Then matrices $\mathbf{P}_n \in \mathbb{R}^{R\times L}$ for $n \in \{1,2,\dots,N\}$ are:

$$\mathbf{P}_n \overset{\text{def}}{=} \mathbf{A} + (\mathbf{W}'_n \odot \mathbf{B})\,\mathbf{W}^\top \tag{9}$$

In each equation above, the random effects parameters are masked using the random effects filter associated with each data point. $\mathbf{Q}$ and $\mathbf{P}_n$ are then transformed into the impulse vector space using the indicator matrices $\mathbf{V}$ and $\mathbf{W}$, respectively. This procedure sums the random effects associated with each data point and adds them to the population-level parameters.

To define the convolution step, let g_l for $l \in \{1,2,\dots,L\}$ be parametric IRF kernels, one for each impulse. Convolution of $\mathbf{X}$ with each IRF kernel is performed by premultiplying the inputs

$\mathbf{X}$ with sparse matrix $\mathbf{G}_l \in \mathbb{R}^{N \times M}$ for $l \in \{1, 2, ..., L\}$:

$$(\mathbf{G}_l)_{[n,*]} \stackrel{\text{def}}{=} g_l \left(\mathbf{b}_{[n]} - \mathbf{a}^\top ; (\mathbf{P}_n)_{[*,l]} \right) \odot \mathbf{F}_{[n,*]} \tag{10}$$

Finally, let $\mathbf{L} \in \{0, 1\}^{K \times L}$ be an indicator matrix mapping the K predictors of $\mathbf{X}$ to the corresponding L impulse vectors of the model.[15] The convolution that yields the design matrix of convolved predictors $\mathbf{X}' \in \mathbb{R}^{N \times L}$ is then defined using a product of the convolution matrices $\mathbf{G}$, the design matrix $\mathbf{X}$, and the impulse indicator $\mathbf{L}$:

$$\mathbf{X}'_{[*,l]} \stackrel{\text{def}}{=} \mathbf{G}_l \, \mathbf{X} \, \mathbf{L}_{[*,l]} \tag{11}$$

The full model mean is the sum of (1) the intercepts and (2) the sum-product of the convolved predictors $\mathbf{X}'$ with the coefficient parameters $\mathbf{Q}$:

$$\mathbf{y} \sim \mathcal{N} \left(\mathbf{q} + (\mathbf{X}' \odot \mathbf{Q}) \, \mathbf{1}, \sigma^2 \right) \tag{12}$$

---

[15] Predictors and impulse vectors are distinguished because in principle multiple IRFs can be applied to the same predictor. In the usual case where this distinction is not needed, $\mathbf{L}$ is identity and $K = L$.

Is this Sentence Difficult? Do you Agree?

Dominique Brunato[◇], **Lorenzo De Mattei**[•]
Felice Dell'Orletta[◇], **Benedetta Iavarone**[⋆], **Giulia Venturi**[◇]
[•]Dipartimento di Informatica, Università di Pisa
[⋆]Scuola Normale Superiore, Pisa
[◇]Istituto di Linguistica Computazionale "Antonio Zampolli" (ILC-CNR), Pisa
ItaliaNLP Lab - *www.italianlp.it*
`{dominique.brunato,felice.dellorletta,giulia.venturi}@ilc.cnr.it`
`lorenzo.demattei@di.unipi.it` `beneiavarone@gmail.com`

Abstract

In this paper, we present a crowdsourcing-based approach to model the human perception of sentence complexity. We collect a large corpus of sentences rated with judgments of complexity for two typologically-different languages, Italian and English. We test our approach in two experimental scenarios aimed to investigate the contribution of a wide set of lexical, morpho-syntactic and syntactic phenomena in predicting i) the degree of agreement among annotators independently from the assigned judgment and ii) the perception of sentence complexity.

1 Introduction

Linguistic complexity is a well-studied and multifaceted notion for which several measures have been proposed in different frameworks ranging from First and Second Language Acquisition, language typology and readability assessment. Such measures depend on the perspective from which linguistic complexity is considered. According to one established distinction, linguistic complexity should be divided into an *absolute* vs a *relative* notion (Miestamo, 2008). While the former is driven by theory and aims at assessing the complexity of a language according to some formal properties of the linguistic system, the latter defines complexity in relation to the language user (e.g. speaker, listener or learner) thus considering complexity in terms of processing difficulty. From this second perspective, sentence complexity is analyzed in terms of cognitive load, which can be inferred using both off-line (e.g. complexity judgments, error rates on comprehension test, preference for a structure over a meaning-equivalent one in elicited production tasks) and online processing measures (e.g. eye-tracking data such as total gaze time, fixation duration and pupil dilation). To operationalize factors underlying sentence processing performance, several complexity metrics have been proposed which consider properties of single word and sentence, as well as experience-based expectations. Word-level predictors shown to correlate with greater processing difficulties are e.g. word frequency, age of acquisition, root frequency effect, orthographic neighbourhood frequency. At syntactic level, a well-studied measure of sentence complexity takes into account dependency length (Gibson, 1998, 2000), which has been used to explain a wide range of psycholinguistic phenomena, such as the subject/object relative clauses asymmetry or the garden path effect in main verb/reduced-relative ambiguities (Gordon et al., 2001; Staub et al., 2010), as well as variations in word order patterns (Gildea and Temperley, 2010), also in a diachronic perspective (Gulordava and Merlo, 2015). Alternatively, processing difficulty has been explained in terms of surprisal (Hale, 2001). Computational models to calculate lexical and syntactic surprisal have been developed by e.g. Roark et al. (2009) using a broad-coverage probabilistic PCFG parser and Demberg and Keller (2009), who introduced Prediction Theory, which aims at unifying Dependency Length Theory with syntactic surprisal, by making use of a psycholinguistically-motivated version of tree-adjoining grammar.

Unlike more conventional studies on human sentence processing carried out in experimental settings, in this study we rely on crowdsourcing methods to investigate how people perceive sentence complexity. The reliability of crowdsourced data for linguistics and computational linguistics research is well acknowledged as shown in the survey by Munro et al. (2010) proving that the quality of findings obtained from the crowd is comparable, if not higher, to controlled laboratory experiments. In addition, crowdsourcing reaches a

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2690–2699
Brussels, Belgium, October 31 - November 4, 2018.

broader population, in terms of age, education, profession etc. and it is thus more suitable to catch the "layman" intuition of sentence complexity. For these reasons, this method has been used in recent works in the field of readability and text simplification; it is the case of Lasecki et al. (2015); Clercq et al. (2013); Brunato et al. (2016) where the crowd was asked to evaluate the level of complexity or the degree of informativeness of simplified sentences compared to the original one.

In our study, we adopted a similar perspective relying on a crowdsourcing approach to collect a wide resource containing multiple annotations of sentence complexity given by humans. Unlike traditional studies which typically assess either lexical or structural complexity phenomena, we focused on the analysis of a wide set of linguistic features to investigate how all contribute to human perception of sentence complexity. This choice is also motivated by previous studies focused on the "form" of a text all related to the assessment of complexity, e.g. readability assessment (Collins-Thompson, 2015), first language acquisition (Sagae et al., 2005) and Native Language Identification (Malmasi et al., 2017).

2 Our Contributions

Our contribution to the study of sentence complexity is multiple:

- we address two research questions aimed to investigate the role played by a set of linguistic phenomena in characterizing a) the agreement among annotators when they rated the sentences independently from the assigned score and b) the human perception of complexity.
- we introduce a new crowdsourcing-based method to assess how people perceive sentence complexity and we test it for two languages;
- we collect two corpora of sentences annotated by humans with a judgment of complexity;

The two research questions refer to two phenomena that are by definition highly subjective and difficult to define. Our study intends to address this vagueness providing the following main contributions: i) detecting the main linguistic phenomena involved in the prediction of agreement and ii) which phenomena characterize a sentence that is perceived complex by a high number of human subjects.

All the data discussed here are made available at www.italianlp.it/resources/.

3 Approach

We collected a dataset of rated sentences through a crowdsourcing task in which annotators were asked to give a score of complexity to a sentence. The task was carried out on two languages, Italian and English, which have different morpho-syntactic and syntactic properties such as morphological richness and word order freedom. This choice was aimed to investigate whether there are linguistic complexity parameters shared by typologically different languages. Starting from the collected rated sentences, we automatically extracted a wide set of features spanning across multiple levels of linguistic description, which have been acknowledged in the literature on human sentence processing to be involved in sentence complexity. The contribution of these features in modeling the perception of sentence complexity was tested in two different scenarios: i) a classification experiment to assess which features contribute more in the automatic prediction of the degree of agreement among annotators and which features vary in a statistically significant way between agreed and not-agreed sentences; ii) a regression experiment to evaluate if the considered features allow predicting the complexity judgment assigned by humans and how they contribute to the prediction.

In what follows, we introduce the three main ingredients of our approach, i.e. the set of linguistic features (Section 3.1), the datasets of sentences (Section 3.2) and the crowdsourcing task (Section 3.3). In the rest of the paper, we describe the experimental scenarios raised by our two research questions and discuss the results (Sections 4 and 5).

3.1 Linguistic Features

The set of features considered in this study captures different aspects of sentence complexity.

Raw text features:

word length, i.e. average number of characters per words (*char_tok* in all tables and figures that follow) and **sentence length**, i.e. average number of words per sentence (*n_tokens*), which are typically used as a proxy of lexical and syntactic complexity in traditional readability metrics (Collins-Thompson, 2015);

Morpho-syntactic features:

distribution of part-of-speech types; **type/token ratio**, calculated as the ratio between the number of lexical types, the number of tokens, in terms of both lemma and forms (*ttr_form*, *ttr_lemma*); **verbal features**, i.e. the distribution of verbs according to mood (*verbs_mood*), tense (*verbs_tense*) and persons (*verbs_num_per*), and **lexical density** (*lex_density*), calculated as the ratio of content words (verbs, nouns, adjectives and adverbs) to the total tokens in a text. Psycholinguistic studies highlight that higher lexical density implies greater cognitive load (Gibson, 1993);

Syntactic features:

probability of syntactic dependency types e.g. subject, direct object, modifier, etc., calculated as the *distribution* of each type out of the total dependency types. Some syntactic relations have been shown to be harder to process, e.g. object-relative clauses and prepositional-phrase attachments (Gibson and Pearlmutter, 1994; Gibson, 2000), or the subject and object relations especially in free word-order languages;

distribution of verbal roots, i.e. the *distribution of verbal roots* out of the total of sentence roots. A lower percentage of verbal roots implies a higher number of nominal sentences which have a less-standard structure due to verb ellipsis thus possibly causing processing ambiguity;

parse tree depth features: the *depth of the whole parse tree* (*max_depth*), calculated in terms of the longest path from the root of the dependency tree to some leaf; the *depth of embedded complement chains* governed by a nominal head and including either prepositional complements or nominal and adjectival modifiers, calculated as the *total* number of prepositional chains (*n_prep_chains*) and the *average* depth of chains (*prep_chain_l*); the *distribution of embedded complement chains by depth*, calculated as the number of chains out of the total number of chains in a sentence (*prep_depth*). All these features are related to length factors and correlate with processing difficulty (Frazier, 1985), as in the case of long sequences of embedded prepositional complements;

verbal predicate features: the distribution of verbal head (*verb_head*); the *arity of verbs*, meant as the average number of instantiated dependency links sharing the same verbal head covering both arguments and modifiers *verb_arity*); the *distribution of verbal head by arity*, calculated as the total number of verbal heads with the same arity in a sentence (*verb_head_arity*); the *relative ordering of subject and object with respect to the verbal head* (*order_subj* and *order_obj*);

subordination features include the *distribution of main vs. subordinate clauses* (*n_subord_clauses* and *n_princ_clauses*; the *average depth of chains of embedded subordinate clauses*, calculated as the *total* number of subordinate chains (*n_subord_chain*) and the *average* depth of subordinate chains (*subord_chain_l*); the *distribution of embedded subordinate clauses chains by depth*, calculated as the number of chains out of the total number of chains in a sentence (*subord_depth*). We also calculated the order of the subordinate clause with respect to the main clause (*order_subord*), since according to e.g. (Miller and Weinert, 1998), sentences containing subordinate clauses in postverbal than in preverbal position are easier to process;

length of dependency links calculated as the number of words between the syntactic head and the dependent: the feature includes the *length of all dependency links* (*links_len)* and of the *maximum dependency links* (*max_links_l*). It is widely known that long-distance constructions cause cognitive load (Gibson, 1998; Gildea and Temperley, 2010);

clause length measured as the number of tokens occurring within a clause (*token_clause*). Syntactic metrics relying on this feature, such as the T-Unit (Hunt, 1966), are widely used e.g. in first and second language acquisition to assess the development of syntactic competence.

3.2 Data

The experiments were carried out on a subset of sentences extracted from two manually revised treebanks. We chose this kind of data in order to prevent possible errors produced by the automatic annotation of sentences. Specifically, we considered the newspaper section of the Italian Universal Dependency Treebank (UDT) (Simi et al., 2014) and the automatically converted Wall Street Journal section of the Penn Treebank (McDonald et al., 2013). Since we wanted to investigate the human perception of complexity with respect to standard language, we didn't use the English version of the UDT containing different genres of web media (e.g. blogs, emails). Although the two selected treebanks have different annotation schemes, the annotation scheme of the UDT project (McDonald et al., 2013) is based on an evolution of (universal) Stanford dependencies (de Marneffe et al., 2006). This allowed us to compare linguistic phenomena correlated with sentence complexity minimizing possible cross-linguistic differences due to not uniform principles of sentence structure representation. In order to reduce the influence of lexicon on the study of sentence complexity we pruned from the two treebanks those sentences containing low-frequency lemmas with respect to a lemma frequency list that we automatically extracted from a large reference corpus, excluding numerals and proper nouns. For what concerns Italian, we used as a reference corpus PAISÁ (Lyding et al., 2014), which is one of the biggest corpus of authentic contemporary Italian texts. For English, we selected a large corpus of sentences from the Wall Street Journal (Nivre et al., 2007). For both languages, all the sentences contained in the two treebanks were grouped into 6 bins based on a different sentence length, i.e. 10, 15, 20, 25, 30, 35 tokens (only for Italian with a range of +/- 1 tokens each). This was meant to investigate if some linguistic features that are known to correlate with sentence length (e.g. parse tree depth features and dependency links) still play an influence on sentence complexity judgments when sentence length is controlled. Sentences in each subset were then ranked according to the sum of the average frequency of their lemmas. We extracted for each bin the first 200-top ranked sentences, with the exception of Italian for which the last bin contains 123 sentences. As a result of the whole selection process, we obtained 1,200 sentences for English and 1,123 for Italian used for experiments.

3.3 Collection of Judgments of Complexity

To collect human complexity judgments, we administered a crowdsourcing task through the platform CrowdFlower[1]. For each language we recruited 20 native speakers who were asked to read a sentence and rate how difficult it was on a 7-point scale where 1 means "very easy" and 7 "very difficult". Sentences were randomly ordered and presented on distinct pages containing five sentences each. To improve the quality of the collected annotations we chose workers with a "high quality" level assigned by the platform on the basis of their performance in previous tasks and we set a minimum of ten seconds to complete a page. We computed the Krippendorff's alpha reliability corresponding to the number of annotators who assigned the same judgment. We obtained a reliability of 26% for Italian and 24% for English.

4 Studying the Agreement between Human Judgments

Our first research question concerned the investigation of linguistic phenomena characterizing the agreement among annotators in assigning the same judgment of complexity to a sentence. To this end, we split the whole set of rated sentences into ten sets corresponding to the number of annotators giving a judgment of complexity within a same range, hereafter referred to *degrees of agreement*[2]. Figure 1 reports the number of sentences for each degree of agreement. For both languages, if we consider a minimum number of 10 agreeing annotators, very few sentences were discarded (∼50 for Italian and 70 for English). As the number of agreeing annotators increases, the number of sentences progressively decreases but we still have a considerable number of sentences (∼600) when 14 annotators agree.

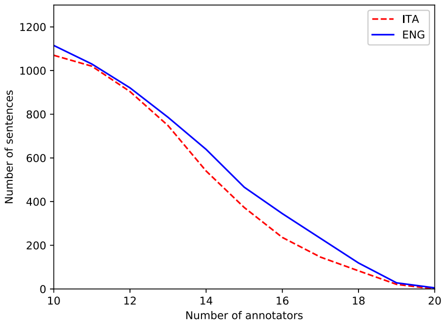

Figure 1: Number of sentences at different degrees of agreement.

To study the linguistic phenomena characterizing the agreement, we firstly extracted the features described in Section 3.1 from sentences on which annotators agreed (*agreed sentences*) and from the rest of sentences (*not-agreed sentences*); we assessed if the difference is statistically significant

---

[1] www.crowdflower.com

[2] Each range was calculated in terms of standard deviation from the mean judgment values given to each sentence.

using Wilcoxon Rank-sum test. This was done for each agreement threshold.

We then performed a feature selection process to identify the features that maximize the accuracies of a classifier in predicting *agreed* vs *not-agreed* sentences. To create a ranking of feature relevance, we used the Recursive Feature Elimination (RFE) algorithm implemented in the Scikit-learn library (Pedregosa et al., 2011), using Linear SVM as estimator algorithm, and we dropped 1 feature in each iteration. We evaluated the classifier performance using a 3-fold cross validation method. At the end of this process we selected the top ranked features. This procedure was iterated 10 times for each degree of agreement.

In order to evaluate the accuracy of the SVM classifier we computed a baseline corresponding to the performance of the classifier using a most-likely class classification method, where each sentence is always classified into the most likely class.

Table 1 reports the features that vary in a statistically significant way (✓in table) and the ones selected in classification (marked with $\star$) for both languages and degrees of agreement levels. As it can be seen, there is an opposite trend between the statistically significant features and those selected by the classifier as the degree of agreement increases. For what concerns the Wilcoxon test, very few features have significantly different values at lower degrees of agreement. That is to say that very few features are involved in discriminating the *agreed* vs *not-agreed* sentences, especially when the agreement is lower than 14.

For both languages, raw text features (*n_tokens* and *char_tok*) vary significantly at all degrees of agreement. Interestingly, these two features are not considered by the classifier which uses more complex syntactic features, such as features related to subordination (e.g. *subord_depth*) and nominal modification (e.g. *prep_chain_l*). Syntactic features start to vary significantly as the agreement increases, e.g. parse tree depth features such as the depth of the whole parse tree (*max_depth*) and the complement chains (*dep_mark*), and features related to the use of subordination. Comparing the two languages we also found a number of differences. For example, at the lowest agreement (degree 10), features of all types turned out to vary significantly for English, while the Italian *agreed* and *not-agreed* sentences do not vary for any features. At higher agreement, Italian *agreed* sentences are characterized by the variation of two language-specific features: the position of the object with respect to the verb head (*order_obj*) and some verbal morphological features (*verbs_num_pers*, *verbs_tense*), which also contributes to the classification only for Italian.

Table 2 reports the accuracy of SVM classifier for each degree of agreement[3] and the baseline. At lower degrees of agreement (i.e. $<$14) the classifier achieves lower accuracy compared to the baseline showing that the selected features do not contribute to discriminate *agreed* vs *not-agreed* sentences. Instead, these features start to have a greater impact for the classification of sentences at degrees 14, 15, 16, 17. This means that at these degrees of agreement the values of the features characterizing the *agreed* sentences are significantly different from those of the *not-agreed* sentences. In addition, even though for these sentences a very high number of features are considered statistically significant by the Wilcoxon test the classifier needs less features to assign the correct class (as shown in Table 1).

5 Correlation of Linguistic Features with Sentence Complexity

The second research question aims to model the human perception of complexity studying the correlation between the set of linguistic features extracted from sentences and the judgments of complexity assigned to each sentence. We firstly calculated the average complexity judgments for the six bins of sentences of the same length (i.e. 10, 15, 20, 25, 30, 35 tokens). As expected, long sentences were judged as more complex for both languages even though all sentences were always rated as more complex for Italian (see Figure 2).

We then calculated the Spearman's rank correlation coefficient between the values of each feature and the average judgments of complexity thus obtaining a ranking of features. The correlation was computed at two distinct degrees of agreement, i.e. 10 and 14. We chose these two thresholds since at 10 the *agreed* sentences correspond to almost all the rated sentences and at 14 the SVM classifier starts to outperform the baseline (see Table 2). Besides, at 14 we still have a quite large set of *agreed* sentences allowing a reliable statistical study of the features (see Figure 1). Only at threshold 10

---

[3] The accuracy was computed as the average classification score of the 10 best results of the feature selection process.

Feature	Agreement															
	10		11		12		13		14		15		16		17	
	IT	EN	IT	EN	IT	EN	IT	EN	IT	EN	IT	EN	IT	EN	IT	EN
char_tok	⋆	⋆	⋆	⋆	⋆	-	-	✓	✓	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆
cpos_ADJ	⋆	⋆	⋆	⋆	⋆	-	-	-	✓⋆	-	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓
cpos_ADP	⋆	⋆	⋆	⋆	⋆	-	-	⋆	-	-	-	-	✓	✓	✓	✓
cpos_ADV	⋆	-	⋆	-	⋆	-	-	-	-	-	⋆	-	⋆	-	⋆	-
cpos_AUX	⋆	-	⋆	-	⋆	-	✓	-	-	-	-	-	✓⋆	-	✓	-
cpos_CONJ	⋆	⋆	⋆	⋆	⋆	-	-	⋆	-	✓	✓	✓⋆	✓⋆	✓	✓	✓
cpos_PRON	⋆	-	⋆	-	⋆	-	-	-	✓	-	✓⋆	-	✓	-	✓	-
cpos_DET	-	⋆	-	⋆	-	-	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆
cpos_NUM	-	⋆	-	✓⋆	-	✓	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆
cpos_PROPN	⋆	-	⋆	-	⋆	-	-	-	✓	-	⋆	-	✓⋆	-		-
cpos_PUNCT	⋆	-	⋆	-	⋆	-	✓	-	-	-	✓⋆	-	✓	-	✓⋆	-
cpos_SCONJ	⋆	-	⋆	-	⋆	-	-	-	-	-	✓⋆	-	✓	-	✓	-
cpos_VERB	-	⋆	-	⋆	-	✓	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆	-	✓
dep_acl	-	-	⋆	-	⋆	-	✓	-	✓	-	✓	-	✓	-	✓	-
dep_acl:relcl	-	-	⋆	-	⋆	-	-	-	⋆	-	✓	-	✓⋆	-	✓	-
dep_adpobj	-	⋆	-	⋆	-	-	-	⋆	-	-	-	-	-	-	-	✓
dep_advcl	⋆	-	⋆	-	⋆	-	-	-	✓	-	✓	-	✓⋆	-	✓	-
dep_amod	⋆	✓⋆	⋆	⋆	⋆	✓	-	✓⋆	✓	✓	✓	✓⋆	✓⋆	✓⋆	✓	✓⋆
dep_appos	-	⋆	-	⋆	-	-	-	-	-	⋆	-	-	-	-	-	-
dep_attr	-	⋆	-	⋆	-	-	-	-	-	-	-	✓⋆	-	✓⋆	-	✓
dep_aux	-	-	⋆	-	⋆	-	✓	-	✓	-	-	-	✓⋆	-	✓	-
dep_case	⋆	-	⋆	-	⋆	-	-	-	⋆	-	-	-	✓	-	✓	-
dep_cc	⋆	⋆	⋆	⋆	⋆	-	-	-	-	✓⋆	✓	✓⋆	✓⋆	✓⋆	✓	✓
dep_ccomp	-	⋆	-	⋆	-	-	-	-	-	✓	-	✓	-	✓	-	✓
dep_compmod	-	⋆	-	⋆	-	-	-	-	-	✓⋆	-	⋆	-	✓⋆	-	✓⋆
dep_conj	⋆	⋆	⋆	⋆	⋆	-	-	✓⋆	-	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓	✓⋆
dep_det	-	⋆	-	⋆	-	-	-	⋆	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆
dep_dobj	⋆	-	⋆	-	⋆	-	-	-	-	-	✓	-	✓⋆	-	✓	-
dep_mark	⋆	⋆	⋆	⋆	⋆	-	✓	⋆	✓	⋆	✓⋆	⋆	✓⋆	✓	✓	✓
dep_nmod	⋆	⋆	⋆	⋆	✓⋆	-	✓	-	✓	-	-	✓⋆	✓⋆	✓	✓	✓
dep_nsubj	-	✓⋆	-	✓⋆	-	✓	-	✓	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆
dep_num	-	⋆	-	⋆	-	✓	-	✓	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆
dep_partmod	-	⋆	-	⋆	-	-	-	-	-	-	-	✓	-	✓	-	✓
dep_poss	-	⋆	-	⋆	-	-	-	-	-	✓	-	✓	-	✓	-	✓
dep_punct	⋆	-	⋆	-	⋆	-	✓	-	-	-	✓⋆	-	✓	-	✓	-
dep_rcmod	-	⋆	-	⋆	-	-	-	⋆	-	-	-	✓⋆	-	✓⋆	-	✓
dep_xcomp	⋆	-	⋆	-	⋆	-	-	-	-	-	-	-	✓	-	✓	-
lex_density	-	⋆	-	⋆	-	-	-	✓⋆	-	✓⋆	-	✓	-	✓⋆	-	✓⋆
links_len	-	✓⋆	⋆	⋆	✓⋆	✓	✓	✓	✓⋆	✓	✓	✓⋆	✓	✓	✓	✓
max_depth	-	⋆	⋆	⋆	✓⋆	-	✓	✓	✓	✓	✓	✓	✓	✓⋆	✓	✓
max_links_l	-	✓⋆	⋆	✓⋆	✓⋆	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
n_prep_chains	⋆	✓⋆	✓⋆	⋆	✓⋆	✓	✓	✓	✓	✓	✓	✓	✓	✓⋆	✓	✓
n_principal_clauses	-	⋆	⋆	⋆	⋆	-	✓	✓	✓	✓⋆	✓	✓	✓	✓	✓	✓
n_subord_chain	⋆	⋆	⋆	⋆	⋆	✓	✓	-	✓⋆	✓⋆	✓⋆	✓	✓⋆	✓	✓	✓
n_subord_clauses	⋆	-	⋆	-	⋆	-	✓	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆	-
n_tokens	-	✓⋆	✓⋆	✓⋆	✓⋆	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
order_obj	-	-	⋆	-	⋆	-	-	-	-	-	✓	-	✓	-	✓	-
order_subj	-	-	⋆	-	⋆	-	-	-	⋆	-	-	-	✓	-	✓	-
order_subord	⋆	⋆	⋆	⋆	⋆	-	✓	✓	✓	✓⋆	✓	✓	✓	✓	✓	✓
prep_chain_l	-	⋆	⋆	⋆	⋆	-	✓	-	✓	✓	✓⋆	✓	✓⋆	✓	✓	✓
prep_depth	-	✓⋆	⋆	⋆	✓	✓	✓	✓	✓	✓⋆	✓	✓⋆	✓	✓⋆	✓	✓⋆
subord_depth	⋆	⋆	⋆	⋆	⋆	-	✓⋆	⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆
token_clause	-	⋆	⋆	⋆	⋆	-	-	-	-	-	-	-	✓	✓	✓	✓
ttr_form	-	✓⋆	⋆	⋆	✓⋆	✓	✓	✓	✓	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓	✓⋆
ttr_lemma	⋆	✓⋆	⋆	✓⋆	⋆	✓	✓	✓⋆	✓	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆
verb_arity	⋆	⋆	⋆	⋆	✓⋆	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
verb_head_arity	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	✓⋆	⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆	✓⋆
verb_head	⋆	⋆	⋆	⋆	✓⋆	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
verbs_num_pers	⋆	-	⋆	-	⋆	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆	-
verbs_tense	⋆	-	⋆	-	✓⋆	-	✓	-	✓⋆	-	✓⋆	-	✓⋆	-	✓⋆	-

Table 1: Linguistic features that vary statistically (✓) and the ones selected by the SVM classifier in at least 50% of the 10 runs (⋆) for Italian and English at different degrees of agreement.

	Baseline Accuracy (%) – SVM Classifier Accuracy (%)							
	10	11	12	13	14	15	16	17
Italian	95.4-95.4	91-90.8	80.6-80.5	66.7-66	51.9-**59.1**	66.8-**68.8**	79-**80.7**	87-**87.1**
English	94-94	86.8-86.8	83.6-77.4	66.3-66.1	53.9-**60**	60.7-**71.8**	70.9-**79.3**	80.4-**84.6**

Table 2: Baseline and SVM classifier accuracy at different degrees of human agreement.

we also calculated the ranking of the features with respect to the six bins of sentences of the same length (L10, L15, L20, L25, L30, L35). Figure 3 reports the ranking of features with p <0.05. Positive numbers mean that the higher the feature value the more complex the sentence was judged (i.e. the feature ranked +1 is the top-ranked one since it is the most positively correlated). Instead, negative numbers mean that the lower the feature value, the more complex the sentence was judged (i.e. the feature ranked -1 is the highest negatively correlated). In both languages, the correlation between the top 20 ranked features and the complexity judgment is extremely high, ranging from 0.30 to 0.85 when we consider sentences at agreement 14. At the two agreement thresholds, for all lengths (columns *T10*, *T14*), they concern not only sentence length but also deep syntactic

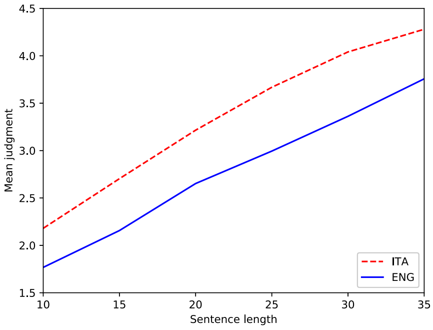

Figure 2: Mean complexity judgment at different sentence length.

features, in terms of e.g. the depth of the whole parse tree (*max_depth*), the length of dependency links (*links_len*), and features related to subordination (e.g. *n_subord_clauses*). Specifically, the 1st-ranked feature in Italian (parse tree depth) and the one in English (sentence length) have a correlation of 0.64 and 0.84 respectively. Nominal modification (*n_prep_chains*) is also highly correlated (Italian r_s=0.59, English 0.54) and similarly ranked in the two languages at 3rd position. The distribution of *verbs_num_pers* makes the sentence harder only for Italian; this is possibly related to the higher complexity of verbal morphology since the 3rd person verb in impersonal structures might increase the ambiguity of the sentence with respect to the referent. Only in English, sentence complexity is affected by the distribution of cardinal numbers (*cpos_NUM*) and the dependency type "numeric modifier" (*dep_num*), in line with the difficulty of numerical information shown in readability studies (Bautista and Saggion, 2014). Conversely, the verbal arity and the relative ordering of subjects with respect to the verb have a lower position in the negative ranking, suggesting that these features make a sentence easier: this might be due to a more fixed predicate-argument structure and word order in this language.

If we focus on sentences of the same length, features considered as a proxy of lexical complexity are in the top positions in both languages. It is the case of the average word length (*char_tok*) and the lexical density (*lex_density*) only for English. Interestingly, while for English the majority of features are similarly ranked in all bins of sentences of the same length, for Italian we observe differences between the rankings of features extracted from sentences $\leq$ and $\geq$20 token long. Namely, when the average sentence length is $\geq$20 tokens, features related to subordination make the sentence more complex.

5.1 Predicting Human Complexity Judgments

To asses the contribution of the linguistic features to predict the judgment of sentence complexity we trained a linear SVM regression model with default parameters. We performed a 3-fold cross validation over each subset of *agreed* sentences at agreement 10 and 14. We measured two performance metrics: the *mean absolute error* to evaluate the accuracy of the model to predict the same complexity judgment assigned by humans; the *Spearman correlation* to evaluate the correlation between the ranking of features produced by the regression model with the ranking produced by the human judgments. Table 3 reports the results and the average score of the two metrics. As it can be seen, the model is very accurate and achieves a very high correlation (>0.56 with p <0.001) with an average error difference (*avg mean abs err*) below 1. In particular, the model obtained higher performance in predicting the ranking of features extracted from sentences at agreement 14. This might be due to the fact these sentences are characterized by a more uniform distribution of linguistic phenomena and that these phenomena contribute to predict the same judgment of complexity. This is in line with the results obtained by the SVM classifier in predicting agreement (Table 2). This is particularly the case of English and it possibly suggests that the set of sentences similarly judged by humans are characterized by a lower variability of the values of the features.

	IT-10	**IT-14**	**EN-10**	**EN-14**
mean abs err 1	0.77	0.78	0.71	0.68
Spearman 1	0.57	0.64	0.68	0.64
mean abs err 2	0.79	0.80	0.70	0.70
Spearman 2	0.55	0.63	0.67	0.73
mean abs err 3	0.85	0.75	0.77	0.60
Spearman 3	0.55	0.64	0.61	0.71
avg mean abs err	**0.80**	**0.78**	**0.72**	**0.66**
avg Spearman	**0.56**	**0.63**	**0.65**	**0.69**

Table 3: Performance of the linear SVM regression model and the avg score at different agreements.

6 Discussion and Conclusion

In this paper, we introduced a method to model the human perception of sentence complexity relying on a new corpus of Italian and English sen-

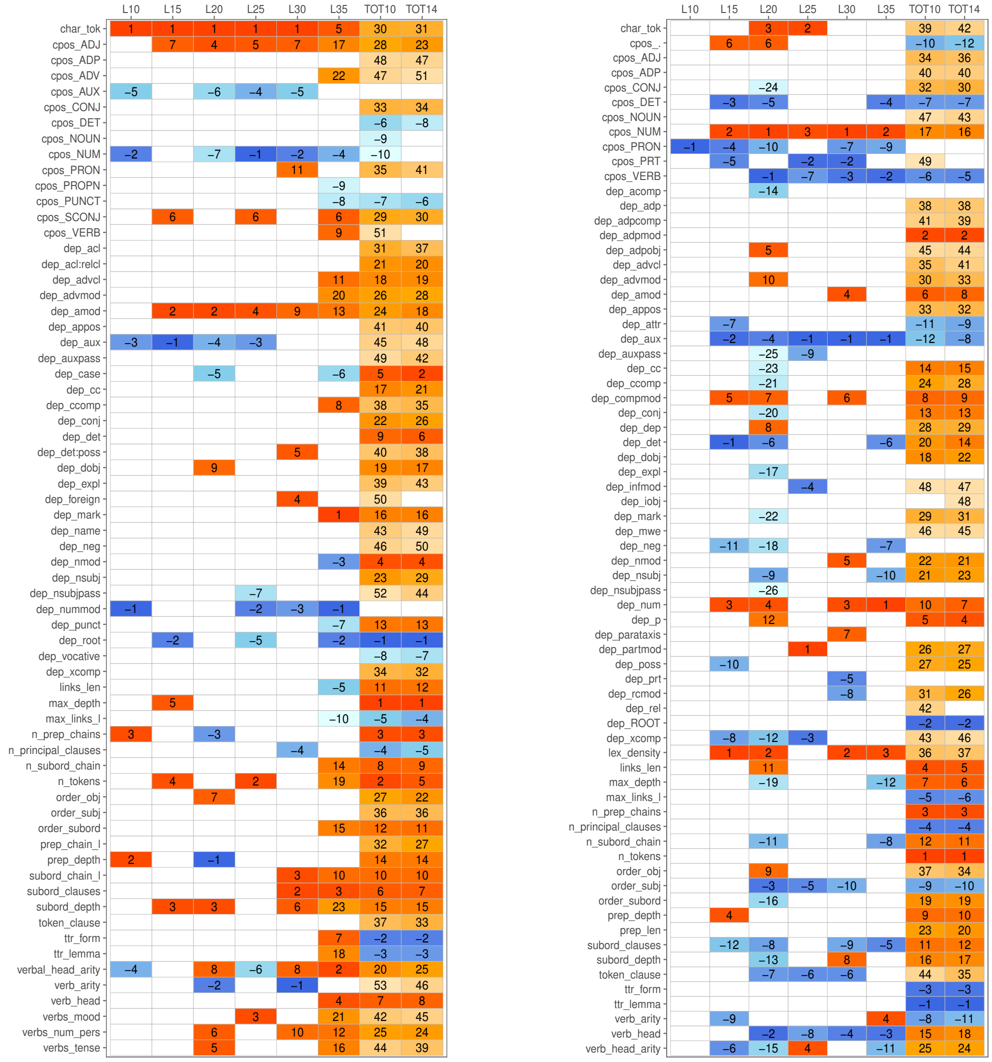

(a) Italian.

	L10	L15	L20	L25	L30	L35	TOT10	TOT14
char_tok	1	1	1	1	1	5	30	31
cpos_ADJ		7	4	5	7	17	28	23
cpos_ADP							48	47
cpos_ADV						22	47	51
cpos_AUX	−5		−6	−4	−5			
cpos_CONJ							33	34
cpos_DET							−6	−8
cpos_NOUN							−9	
cpos_NUM	−2		−7	−1	−2	−4	−10	
cpos_PRON					11		35	41
cpos_PROPN						−9		
cpos_PUNCT						−8	−7	−6
cpos_SCONJ		6		6		6	29	30
cpos_VERB						9	51	
dep_acl							31	37
dep_acl:relcl							21	20
dep_advcl						11	18	19
dep_advmod						20	26	28
dep_amod		2	2	4	9	13	24	18
dep_appos							41	40
dep_aux	−3	−1	−4	−3			45	48
dep_auxpass							49	42
dep_case			−5			−6	5	2
dep_cc							17	21
dep_ccomp						8	38	35
dep_conj							22	26
dep_det							9	6
dep_det:poss					5		40	38
dep_dobj			9				19	17
dep_expl							39	43
dep_foreign					4		50	
dep_mark						1	16	16
dep_name							43	49
dep_neg							46	50
dep_nmod						−3	4	4
dep_nsubj							23	29
dep_nsubjpass				−7			52	44
dep_nummod	−1			−2	−3	−1		
dep_punct						−7	13	13
dep_root		−2		−5		−2	−1	−1
dep_vocative							−8	−7
dep_xcomp							34	32
links_len						−5	11	12
max_depth		5					1	1
max_links_l						−10	−5	−4
n_prep_chains	3		−3				3	3
n_principal_clauses					−4		−4	−5
n_subord_chain						14	8	9
n_tokens		4		2		19	2	5
order_obj			7				27	22
order_subj							36	36
order_subord						15	12	11
prep_chain_l							32	27
prep_depth	2		−1				14	14
subord_chain_l					3	10	10	10
subord_clauses					2	3	6	7
subord_depth		3	3		6	23	15	15
token_clause							37	33
ttr_form						7	−2	−2
ttr_lemma						18	−3	−3
verbal_head_arity	−4		8	−6	8	2	20	25
verb_arity			−2		−1		53	46
verb_head						4	7	8
verbs_mood				3		21	42	45
verbs_num_pers			6		10	12	25	24
verbs_tense			5			16	44	39

(b) English.

	L10	L15	L20	L25	L30	L35	TOT10	TOT14
char_tok			3	2			39	42
cpos_.		6	6				−10	−12
cpos_ADJ							34	36
cpos_ADP							40	40
cpos_CONJ			−24				32	30
cpos_DET		−3	−5			−4	−7	−7
cpos_NOUN							47	43
cpos_NUM		2	1	3	1	2	17	16
cpos_PRON	−1	−4	−10		−7	−9		
cpos_PRT		−5		−2	−2		49	
cpos_VERB			−1	−7	−3	−2	−6	−5
dep_acomp			−14					
dep_adp							38	38
dep_adpcomp							41	39
dep_adpmod							2	2
dep_adpobj			5				45	44
dep_advcl							35	41
dep_advmod			10				30	33
dep_amod					4		6	8
dep_appos							33	32
dep_attr		−7					−11	−9
dep_aux		−2	−4	−1	−1	−1	−12	−8
dep_auxpass			−25	−9				
dep_cc			−23				14	15
dep_ccomp			−21				24	28
dep_compmod		5	7		6		8	9
dep_conj			−20				13	13
dep_dep			8				28	29
dep_det		−1	−6			−6	20	14
dep_dobj							18	22
dep_expl			−17					
dep_infmod				−4			48	47
dep_iobj								48
dep_mark			−22				29	31
dep_mwe							46	45
dep_neg		−11	−18			−7		
dep_nmod					5		22	21
dep_nsubj			−9			−10	21	23
dep_nsubjpass			−26					
dep_num		3	4		3	1	10	7
dep_p			12				5	4
dep_parataxis					7			
dep_partmod				1			26	27
dep_poss		−10					27	25
dep_prt					−5			
dep_rcmod					−8		31	26
dep_rel							42	
dep_ROOT							−2	−2
dep_xcomp		−8	−12	−3			43	46
lex_density		1	2		2	3	36	37
links_len			11				4	5
max_depth			−19			−12	7	6
max_links_l							−5	−6
n_prep_chains							3	3
n_principal_clauses							−4	−4
n_subord_chain			−11			−8	12	11
n_tokens							1	1
order_obj			9				37	34
order_subj			−3	−5	−10		−9	−10
order_subord			−16				19	19
prep_depth		4					9	10
prep_len							23	20
subord_clauses		−12	−8		−9	−5	11	12
subord_depth			−13		8		16	17
token_clause			−7	−6	−6		44	35
ttr_form							−3	−3
ttr_lemma							−1	−1
verb_arity		−9				4	−8	−11
verb_head			−2	−8	−4	−3	15	18
verb_head_arity		−6	−15	4		−11	25	24

Figure 3: Features correlating with human judgments at different sentence lengths and with respect to the sentences at agreement 10 (*TOT 10*) and 14 (*TOT 14*).

tences rated with human complexity judgments. We tested the contribution of a wide set of linguistic features automatically extracted from these sentences in two experimental scenarios. The first one highlighted that we can reliably predict the degree of agreement between human annotators, independently from the assigned judgment of complexity: given the high subjectivity of the task, this is a quite notable result that to our knowledge has never been reported. We observed in particular that deep syntactic features related to e.g. the use of subordination and nominal modification play a main role in the automatic prediction of human agreement. This is true for the two languages even though we found that some features resulted to be more relevant in the classification of *agreed* Italian sentences, e.g. the relative ordering of the object. Interestingly, we also noticed that the classifier needs few features to predict agreed sentences when more than half of annotators shares the same judgment.

In the second experiment, we studied the correlation between linguistic features and complexity judgments. The resulting ranking highlighted the key role played by syntactic phenomena: features related to sentence structure are among the top-ranked features characterizing sentences that were rated highly complex by a given number of agreeing annotators. When sentence length was controlled, the relevance of the considered features changes in particular for Italian: e.g. features concerning the use of subordination make the sentence more complex when sentence length is ≥20 tokens. As showed by the results of the regression model, the set of studied features contribute significantly to automatically predict the human judgment of sentence complexity.

In addition, the presented corpus can be useful for different applications. From a psycholinguistic perspective, it can be used for comparison with data collected through controlled experimental scenarios assessing sentence complexity in terms of cognitive measures (offline and online), which are also more constrained and costly to acquire in large-dimensions. The corpus also allows to study whether features of linguistic complexity are implied in modeling other properties of texts, such as the level of engagement or subjectivity. From a NLP perspective, the corpus can be exploited to train systems able to predict people's perception of complexity. For example, it can support a range of related tasks, such as the development of linguistically-informed algorithms for the automatic assessment of text difficulty, as well as in Natural Language Generation tasks, going from text simplification to the automatic generation/evaluation of highly-engaging texts.

References

Susana Bautista and Horacio Saggion. 2014. Can numerical expressions be simpler? Implementation and demostration of a numerical simplification system for spanish. In *Proceedings of LREC, the 9th International Conference on Language Resources and Evaluation*.

D. Brunato, A. Cimino, F. Dell'Orletta F., and G. Venturi. 2016. PaCCSS–IT: A parallel corpus of complex–simple sentences for automatic text simplification. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 10–18.

O. De Clercq, V. Hoste, B. Desmet, and P. Van Oosten. 2013. Using the crowd for readability prediction. *Natural Language Engineering*, pages 1–33.

Kevyn Collins-Thompson. 2015. Computational assessment of text readability: A survey of current and future research. *Recent Advances in Automatic Readability Assessment and Text Simplification. Special issue of International Journal of Applied Linguistics*, 165(2):97–135.

Vera Demberg and Frank Keller. 2009. A computational model of prediction in human parsing: Unifying locality and surprisal effects. In *Proceedings of CogSci 2009, the 31st Annual Conference of the Cognitive Science Society*, pages 1888–1893.

Lin Frazier. 1985. Syntactic complexity. *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, Cambridge: Cambridge University Press.

Edward Gibson. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 24(11):1–76.

Edward Gibson. 2000. The dependency locality theory: A distance–based theory of linguistic complexity. *W.O.A. Marants and Y. Miyashita (Eds.), Image, Language and Brain*, Cambridge, MA: MIT Press.

Edward Gibson and Neal J. Pearlmutter. 1994. A corpus-based analysis of psycholinguistic constraints on prepositional-phrase attachment. *Perspectives on sentence processing*.

Timothy R. Gibson. 1993. Towards a discourse theory of abstracts and abstracting. *Monographs in Systemic Linguistics*, Nottingham: Department of English Studies, University of Nottingham.

Daniel Gildea and David Temperley. 2010. Do grammars minimize dependency length? *Cognitive Science*, 34(2):286–310.

Peter C. Gordon, Randall Hendrick, and Marcus Johnson. 2001. Memory interference during language processing. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 27(6):1411–1423.

Kristina Gulordava and Paola Merlo. 2015. Diachronic trends in word order freedom and dependency length in dependency-annotated corpora of latin and ancient greek. In *Proceedings of Depling 2015, the Third International Conference on Dependency Linguistics*.

John Hale. 2001. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the NAACL*, pages 159–166.

Kellogg W. Hunt. 1966. Recent measures in syntactic development. *Elementary English*, 43(7):732–739.

W. S. Lasecki, L. Rello, and J. P. Bigham. 2015. Measuring text simplification with the crowd. In *Proceedings of the 12th Web for All Conference (W4A 15)*.

V. Lyding, E. Stemle, C. Borghetti, M. Brunello, S. Castagnoli, F. Dell'Orletta, H. Dittmann, A. Lenci, and V. Pirrelli. 2014. The PAISÀ corpus of Italian web texts. In *Proceedings of the 9th Web as Corpus Workshop (WAC-9) EACL*, pages 36–43.

S. Malmasi, E. Keelan, A. Cahill, J. Tetreault, R. Pugh, C. Hamill, D. Napolitano, and Y. Qian. 2017. A report on the 2017 native language identification shared task. In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*, pages 449–454.

R. McDonald, J. Nivre, Y. Quirmbach-brundage, Y. Goldberg, D. Das, K. Ganchev, K. Hall, S. Petrov, H. Zhang, O. Tackstrom, C. Bedini, N. B. Castelló, and J. Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 92–97.

M. Miestamo. 2008. Grammatical complexity in a cross–linguistic perspective. *M. Miestamo, K. Sinnemaki and F. Karlsson (eds.), Language Complexity: Typology, Contact, Change*, Amsterdam: Benjamins:23–41.

Jim Miller and Regina Weinert. 1998. Spontaneous spoken language. *Syntax and discourse. Oxford, Clarendon Press.*

R. Munro, S. Bethard, V. Kuperman, V.T. Lai, R. Melnick, C. Potts, T. Schnoebelen, and H. Tily. 2010. Crowdsourcing and language studies: The new generation of linguistic data. In *Proceedings of the Workshop on Creating Speech and Language Data with Amazons Mechanical Turk*, pages 122–130.

J. Nivre, J. Hall, S. Kubler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the EMNLP-CoNLL*, pages 915–932.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Brian Roark, A. Bachrach, C. Cardenas, and C. Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top–down parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 324–333.

K. Sagae, A. Lavie, and B. MacWhinne. 2005. Automatic measurement of syntactic development in child language. In *Proceedings of the 43rd Annual Meeting of the ACL.*

Maria Simi, Cristina Bosco, and Simonetta Montemagni. 2014. Less is more? Towards a reduced inventory of categories for training a parser for the italian stanford dependencies. In *Proceedings of the 9th International Conference on Language Resources and Evaluation, (LREC'14)*, pages 83–90.

A. Staub, S.J. White, D. Drieghe, E.C. Hollway, and K. Rayner. 2010. Distributional effects of word frequency on eye fixation durations. *Journal of Experimental Psychology: Human Perception and Performance.*

Neural Transition Based Parsing of Web Queries: An Entity Based Approach

Rivka Malca and **Roi Reichart**
Technion, Israel Institute of Technology
srikim@st.technion.ac.il, roiri@technion.ac.il

Abstract

Web queries with question intent manifest a complex syntactic structure and the processing of this structure is important for their interpretation. Pinter et al. (2016) has formalized the grammar of these queries and proposed semi-supervised algorithms for the adaptation of parsers originally designed to parse according to the standard dependency grammar, so that they can account for the unique forest grammar of queries. However, their algorithms rely on resources typically not available outside of big web corporates. We propose a new BiLSTM query parser that: (1) Explicitly accounts for the unique grammar of web queries; and (2) Utilizes named entity (NE) information from a BiLSTM NE tagger, that can be jointly trained with the parser. In order to train our model we annotate the query treebank of Pinter et al. (2016) with NEs. When trained on 2500 annotated queries our parser achieves UAS of 83.5% and segmentation F1-score of 84.5, substantially outperforming existing state-of-the-art parsers.[1]

1 Introduction

Web queries, authored by users in order to search for information, form a major gate to the Web, and their correct interpretation is hence invaluable. While earlier research (Bergsma and Wang, 2007; Barr et al., 2008) suggested that many queries are trivial in structure, Pinter et al. (2016) (henceforth PRS16) demonstrated that this is often not the case. Particularly, they demonstrated that queries related to questions that are answered in Community Question Answering (CQA) sites (social QA forums such as Yahoo Answers), follow a complex dependency grammar. As such queries are quite frequent (e.g. an early study (White et al., 2015) showed that they constitute ∼10% of all queries issued to search engines) and their interpretation can benefit from structural analysis (e.g. (Tsur et al., 2016)), effective query parsing is of importance.[2]

In order to properly describe the syntactic structure of queries, PRS16 extended the standard dependency grammar so that it accounts for dependency forests that consist of syntactically independent segments, each of which has its own internal dependency tree. Additionally, they constructed a query treebank consisting of 4,000 CQA queries, manually annotated according to the query grammar. Examples of annotated queries from the treebank are given in Figures 2 and 4.

PRS16 presented two algorithms that can adapt off-the-shelf dependency parsers trained on standard edited text, so that they can produce syntactic structures that conform to the query grammar. Importantly, their methods do not contain any change to the states and transitions of the parser. Instead, they require millions of unannotated queries each paired with the title (usually a grammatical question) of the Yahoo Answers question page that was clicked by the user who initiated the query; it is the alignment between the query and the title that provides the training signal for their algorithms. Unfortunately, millions of (query, title) pairs are typically not available outside of big web corporates, and the practical value of the PRS16 algorithms is hence limited. Moreover, despite the unique supervision signal, their parsers achieve a segmentation F1-score of up to 70.4, which leaves a large room for improvement.

In this paper we present a transition-based BiLSTM query parser that requires no distant supervision. Our parser is based on two ideas: (a) We change the standard transition system so that the parser explicitly supports the PRS16 query gram-

---

[1] Our code and data are available at: `https://bitbucket.org/riki_malca/ptsparser/src/master`.

[2] In the rest of the paper we will refer to queries with question intent, like those addressed in PRS16, simply as queries.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2700–2710
Brussels, Belgium, October 31 - November 4, 2018.

mar (§ 3.2); and (b) Observing that entities are very frequent in CQA queries and provide a strong structural signal (§ 4), we extend our parser to consider information from a named entity (NE) tagger. We explore both sequential and joint training of the NE tagger and the parser and demonstrate the superiority of the joint approach.

As another contribution of this paper, we annotate the dataset of PRS16 with NEs (§ 4). We use this data to establish our observation about the importance of NEs for query parsing, and in order to train and test our entity-aware parser.

We split the PRS16 corpus to train (2500 queries), development (750) and test (750) sections (§ 6). In this training setup our segmentation and entity-aware parser achieves a segmentation F1-score of 84.5 (100 on single-segment queries, 60.7 on multi-segment queries) and a dependency parsing UAS of 83.5. Our model outperforms its simpler variants that do not utilize segmentation and/or NE information. For example, the BiLSTM parser of Kiperwasser and Goldberg (2016), which forms the basis for our model, scores 67.7 in segmentation F1 and 77.0 in UAS.

We note that our training setup is very different than that of PRS16. They trained their parser on edited text from the OntoNotes 5 corpus (Weischedel et al., 2013) augmented with millions of (query, title) pairs, and their test set consists of the 4000 queries of the query treebank, as they do not train on queries.[3] While our work is not directly comparable to theirs, it is worth mentioning that their best model scores 70.4 in segmentation F1 and 76.4 in UAS, much lower than the numbers we report here for our best models.

2 Previous Work

We divide this section to two: We start with works that analyze the structure of queries and the role of NEs in their processing, and then discuss work on parsing of user generated content on the web.

Query structure and entity analysis As noted in PRS16, web queries differ from standard sentences, as they tend to be shorter and have a unique dependency structure. Hence, prior to PRS16 several works have addressed the syntactic structure of web queries. However, all these works were restricted to tasks that are much simpler than full dependency parsing, including POS tagging (Bendersky et al., 2010; Ganchev et al., 2012), phrase chunking (Bendersky et al., 2011), semantic tagging (Manshadi and Li, 2009; Li, 2010) and classification of queries into syntactic classes (Allan and Raghavan, 2002; Barr et al., 2008).

NER has been recognized as a fundamental problem in query processing by Guo et al. (2009), and many works since (e.g. (Alasiry et al., 2012; Eiselt and Figueroa, 2013; Zhai et al., 2016)) explored various models and features for the task. Differently from those works, our goal is to design a BiLSTM model that can be easily integrated with modern BiLSTM parsers. We hence use simple input features and a simple NE scheme (e.g. see (Guo et al., 2009) for more fine-grained distinctions). More sophisticated features, entity schemes and deep learning architectures (e.g. (Lample et al., 2016)) are left for the future.

Syntactic parsing of Web data Only a handful of papers aimed to parse web data. One important example is the shared task of Petrov and McDonald (2012) on parsing web data from the Google Web Treebank, consisting of texts from the email, weblog, CQA, newsgroup, and review domains. Other relevant works are the tweet parsers of Foster et al. (2011), Kong et al. (2014) and Liu et al. (2018). However, all these works did not address the unique properties of web queries with question intent that express information needs in a concise manner (e.g. with one or more phrases or sentence fragments) and follow a forest-based grammar.

PRS16 were the first, and to the best of our knowledge the only work to address the parsing of web queries with question intent. However, as noted in § 1 their algorithms rely on millions of (query, title) pairs, which deems their algorithm impractical for most users. In practice, they started with a query log of 60M Yahoo Answers pages and ended up using 7.5M queries as distant supervision. In this paper we aim to overcome this limitation by introducing a high quality query parser that can train on several thousands annotated queries to provide higher UAS and segmentation F1 figures compared to those reported in PRS16 (see footnote 3 for their training protocol and data).

We finally note that joint parsing and NER was explored in past (Reichart et al., 2008; Finkel and Manning, 2009, 2010), but for edited text, standard grammar and different modeling techniques. Our work re-emphasizes the strong ties between

---

[3] In some setups they used the segmentation signal from the queries and experimented with a five fold cross-validation over the 4000 queries

NER and parsing, in the context of query analysis.

3 Segmentation-Aware Parsing

In this section we present a parser that explicitly accounts for the query dependency grammar of PRS16. We start (§ 3.1) with a brief description of the BiLSTM parser of Kiperwasser and Goldberg (2016) (henceforth KG16), that forms the basis for our parser, and then describe our query parser.

3.1 The KG16 BiLSTM Parser

KG16 presented a BiLSTM model for transition based dependency parsing (Figure 1). Given a sentence s with words $w_1, ..., w_n$ and corresponding POS tags $p_1, ..., p_n$, the word w_i is represented as:

$$x_i = e(w_i) \circ e(p_i) \tag{1}$$

where $e(w_i)$ and $e(p_i)$ are the embeddings of w_i and p_i, respectively, and $\circ$ is the vector concatenation operator. [4]

The BiLSTM consists of two LSTMs: $LSTM_{forward}$ and $LSTM_{backward}$. Given an input vector x_i, $LSTM_{forward}(x_i)$ captures the past context, and is calculated using the information in the input vectors $x_1, \ldots, x_{i-1}$. Similarly, $LSTM_{backward}(x_i)$ captures the future context and is calculated using the information in the input vectors $x_n, \ldots, x_{i+1}$. v_i, the resulting representation of x_i, is given by:

$$\begin{aligned} v_i &= BiLSTM(x_{1..n}, i) \\ &= LSTM_{forward}(x_i) \circ LSTM_{backward}(x_i) \end{aligned} \tag{2}$$

The parser implements the arc-hybrid system (Kuhlmann et al., 2011) which uses a configuration $c = (\sigma, \beta, A)$ where $\sigma = [s_0, s_1, ..]$ is a stack, $\beta = [b_0, b_1..]$ is a buffer and A is a set of dependency arcs. The arc-hybrid system allows three transitions: $SHIFT$, $LEFT_{arc}$ and $RIGHT_{arc}$. At each step the parser scores the possible transitions and selects the highest scoring one.

The parser represents each configuration by the concatenation of the BiLSTM embeddings of the first word in the buffer (b_0) and the three words at the top of the stack (s_0, s_1 and s_2). Then, a multi-layer perceptron (MLP) with one hidden layer scores the possible transitions given the current configuration:

$$MLP_\theta(c) = W^2 \cdot tanh(W^1 \cdot c + b^1) + b^2 \tag{3}$$

where $\theta = W^1, W^2, b^1, b^2$ are the MLP parameters and $c = v_{s_0} \circ v_{s_1} \circ v_{s_2} \circ v_{b_0}$.

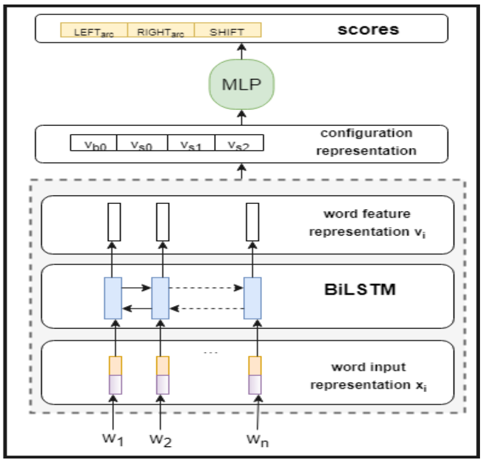

Figure 1: A sketch of the KG16 arc-hybrid parser.

The parser employs a margin-based loss (MBL) function at each step:

$$\begin{aligned} MBL = max(0, 1 - \max_{t_o \in G} MLP_\theta(c)[t_o] \\ + \max_{t_p \in T \setminus G} MLP_\theta(c)[t_p]) \end{aligned} \tag{4}$$

where T is the set of possible transitions and G is the set of correct transitions. The losses are summed throughout the parsing of a sentence and the parameters are updated accordingly.

The parser employs a *dynamic oracle* (Goldberg and Nivre, 2013), which enables exploration in training. We next describe our modification of the KG16 parser. Specifically, we change the transition logic so that it can directly account for the query grammar defined in PRS16.

3.2 A Segmentation-Aware BiLSTM Parser

In order for our parser to directly account for the forest-based query grammar of PRS16, we follow previous work (e.g. Nivre (2009)) and modify its set of actions and transition logic. Before we do that, we start with a more standard modification.

An arc-eager KG16 parser The first step in the design of our parser is changing the arc-hybrid system of the KG16 parser to an arc-eager system (Nivre, 2008). To do that we change the definitions of the $RIGHT_{arc}$ and $LEFT_{arc}$ transitions and add a *REDUCE* transition. The (original) arc-hybrid and the (modified) arc-eager KG16 parsers are denoted with P^H and P^E, respectively.

The motivation for this change is the addition of the $REDUCE$ transition that explicitly facili-

---

[4] The embedding vectors are initialized using the Xavier initialization (Glorot and Bengio, 2010) and trained as part of the BiLSTM.

tates segmentation. After the words in the stack σ are reduced, they cannot be connected to the unprocessed words in the buffer β. This state constitutes a *segmentation point*. We use this connection between the $REDUCE$ transition and the segmentation operation as an integral part of our new segmentation-aware parser, and hence all the following parsers extend this arc-eager parser.

A segmentation-aware parser In our parser, denoted as P^B (for BASIC), a configuration $c = (\sigma, \beta_1, \beta_2, A)$ consists of a stack $\sigma = [s_0, s_1, ..]$, two buffers: $\beta_1 = [b_{10}, .., b_{1last}]$ and $\beta_2 = [b_{20}, .., b_{2last}]$, and a dependency arcs set A. The buffers β_1, β_2 contain the unprocessed tokens, and the words within β_2 form the current segment. We expand the configuration representation to include not only the representations of the first token in the buffer β_1 (b_{10}) and the three tokens at the top of the stack σ (s_0, s_1, s_2) but also the representation of the last token in the buffer β_2 (b_{2last}).

Given a sentence $s = w_1, \ldots, w_n$, the initial configuration is $([ROOT], [w_i \mid i > 1], [w_1], \emptyset)$. In the final configuration the stack σ contains only the $ROOT$ token (we refer to this as an empty stack) and both buffers, β_1 and β_2 are empty. The new transition set, described in Table 1, includes a new transition: *PushToSeg*. This transition adds a new token to the current segment by pushing the top token of β_1 to the end of β_2. The $REDUCE$ transition preconditions have also been modified: this transition is only allowed if β_2 is empty or there is more then one word in the stack.

This new parser performs a two-step process that repeats until convergence, to induce a parse forest. The first step is segment allocation, consisting of a sequence of PushToSeg transitions. When the parser reaches a configuration in which β_2 and σ are empty, only this transition is allowed. There can be one or more consecutive PushToSeg transitions, each pushes a new token from β_1 to β_2. This step ends once the parser selects any other transition, to form a segmentation point. In the second step the allocated segment is parsed. In this step the parser acts as an arc-eager parser with β_2 as the main buffer, and the PushToSeg transition and the β_1 buffer being ignored until the segment is completely parsed (the PushToSeg transition is forbidden while the stack is not empty).

An example of the parsing process is provided in Figure 2. Appendix A provides a proof that the parser is complete and sound as required in Nivre (2008) from any dependency parser.

We next describe two auxiliary segmentation models that can be integrated with our parser.

3.3 Auxiliary Segmentation Models

We consider two models: one is independent of the parser while the other is added as a component to the parser.

Independent segmentation model Similarly to our parser, this model, denoted as SEG, is a BiLSTM that feeds an MLP classifier which predicts for every input word whether it is a segmentation point or not. The loss is a sum of word level MBL functions (equation 4). The input word representation, x_i, and the definition of the hidden word vector, v_i, are as in equations 1 and 2, respectively; the output scores vector $o^{seg}(w_i)$ is derived from the hidden vector, h^{seg}, as in equation 3:

$$\begin{aligned} h^{seg}(v_i) &= tanh(W_s^1 \cdot v_i + b_s^1) \\ o^{seg}(v_i) &= W_s^2 \cdot h^{seg}(v_i) + b_s^2 \end{aligned} \tag{5}$$

where $W_s^1, W_s^2, b_s^1, b_s^2$ are parameters.

We consider two ways through which our P^B parser uses the information from the SEG model. The parser we denote with P^S concatenates the SEG hidden vector of the top word of the stack to the configuration representation:

$$c_{seg} = v_{s_0} \circ v_{s_1} \circ v_{s_2} \circ v_{b_{10}} \circ v_{b_{2last}} \circ h^{seg}(s_0)$$

Alternatively, the parser we denote with P^{FS} (for $FULL\_SEG$) concatenates the hidden vectors of all the configuration elements to the configuration representation:

$$\begin{aligned} c_{full\_seg} =& v_{s_0} \circ v_{s_1} \circ v_{s_2} \circ v_{b_{10}} \circ v_{b_{2last}} \\ &\circ h^{seg}(s_0) \circ h^{seg}(s_1) \circ h^{seg}(s_2) \\ &\circ h^{seg}(b_{10}) \circ h^{seg}(b_{2last}) \end{aligned}$$

This segmentation model can be trained independently of the parser or jointly with it. In development data experiments independent training was superior so we report results with this option.

Configuration-based segmentation model This model, denoted as *FLAG*, predicts whether a given parser configuration is a segmentation point. The answer is positive if the processed words (words in the stack σ) are in the same segment and the unprocessed words (words in the buffer β_2 or β_1) are not in this segment.

Transition		**Precondition**
$LEFT_{arc}$	$(\sigma\mid i, \beta_1, j\mid\beta_2, A) \Rightarrow (\sigma, \beta_1, j\mid\beta_2, A \cup (j, l, i))$	$\neg[i = 0] \wedge \neg\exists k \exists l'[(k, l', i) \in A]$
$RIGHT_{arc}$	$(\sigma\mid i, \beta_1, j\mid\beta_2, A) \Rightarrow (\sigma\mid i\mid j, \beta_1, \beta_2, A \cup (i, l, j))$	
$REDUCE$	$(\sigma\mid i, \beta_1, \beta_2, A) \Rightarrow (\sigma, \beta_1, \beta_2, A)$	$[\exists k \exists l[(k, l, i) \in A]]$ $\wedge\neg[\boldsymbol{size(\beta_2)! = 0 \wedge (size(\sigma) == 1)}]$
$SHIFT$	$(\sigma, \beta_1, i\mid\beta_2, A) \Rightarrow (\sigma\mid i, \beta_1, \beta_2, A)$	
$\boldsymbol{PushToSeg}$	$(\sigma, i\mid\beta_1, \beta_2, A) \Rightarrow (\sigma, \beta_1, \beta_2\mid i, A)$	$size(\sigma) == 1$

Table 1: The transition logic of the segmentation-aware parser. The bold items are the new transitions and conditions of our parser, compared to a standard arc-eager parser.

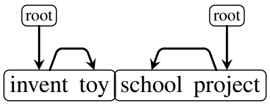

	Action	$\boldsymbol{\beta_2}$	$\boldsymbol{\beta_1}$	$\boldsymbol{\sigma}$	**Arcs**
1	$PushToSeg$	[invent]	[toy, school, project]	[ROOT]	$\emptyset$
2	$PushToSeg$	[invent, toy]	[school, project]	[ROOT]	$\emptyset$
3	$RIGHT_{arc}$	[toy]	[school, project]	[ROOT, invent]	A = (ROOT, invent)
4	$RIGHT_{arc}$	[]	[school, project]	[ROOT, invent, toy]	$A = A \cup (invent, toy)$
5	$REDUCE$	[]	[school, project]	[ROOT, invent]	A
6	$REDUCE$	[]	[school, project]	[ROOT]	A
7	$PushToSeg$	[school]	[project]	[ROOT]	A
8	$PushToSeg$	[school, project]	[]	[ROOT]	A
9	$SHIFT$	[project]	[]	[ROOT, school]	A
10	$LEFT_{arc}$	[project]	[]	[ROOT]	$A = A \cup (project, school)$
11	$RIGHT_{arc}$	[]	[]	[ROOT, project]	$A = A \cup (ROOT, project)$
12	$REDUCE$	[]	[]	[ROOT]	A

Figure 2: Example of the application of our segmentation-aware parser to the multi-segment query *invent toy school project* (borrowed from PRS16). First, a sequence of PushToSeg transitions is performed in order to insert the first segment *invent toy* to the buffer β_2 (transitions 1-2). Then, the segment is parsed until the buffer β_2 and the stack σ are empty (3-6). Similarly, the second segment *school project* is pushed to the buffer β_2 through a sequence of PushToSeg transitions (7-8) and then parsed (9-12).

The model is a simple MLP that receives a parser configuration as input and produces a hidden vector (denoted with $h^{flag}(c)$) and a scores vector (denoted with $o^{flag}(c)$). The equations for $h^{flag}(c)$ and $o^{flag}(c)$ are similar to equation 5, and the loss is an MBL loss as in equation 4.

Information from this model is integrated into the parser configuration representation through:

$$c_{flag}(c) = c \circ h^{flag}(c)$$

We refer to this parser as P^{Fl} (for FLAG). The $FLAG$ model must be trained jointly with the parser as its input is a parser configuration.

As shown in § 7, adapting the KG16 parser to explicitly account for multiple segments improves over the KG16 parser in the task of query parsing. We next show how additional gains can be achieved when recognizing the role of NEs.

4 Entities in Query Parsing

In this section we explore the role of NEs in the syntactic structure of queries. We first describe our NE annotation process, and then qualitatively demonstrate the valuable structural cues they provide. In § 5 we will describe extensions of the segmentation-aware BiLSTM parser (§ 3.2) that integrate information from a BiLSTM NE tagger.

Data We consider five entity types: Location (e.g ”London”), Person (e.g ”Marilyn Monroe”), Organization (e.g ”Google”), Product (e.g ”Iphone 4”) and Other (e.g. see Figure 4 for NEs such as ”song name” and ”computer game”). Two human annotators annotated the dataset. Of the 4000 queries, 400 were randomly selected for initial tagging by both annotators, so that they could discuss ambiguous cases and resolve conflicts (the labeled micro-F1 score between the annotators at this stage was 85.6). Then, the remaining 3600 queries were equally split between the two annotators, who again consulted each other in ambiguous cases (the inter-annotator micro-F1, measured on a randomly sampled set of 100 queries, was 92.0).

NEs as a dependency parsing signal The dataset consists of 3010 single-segment (henceforth *SSG*) and 990 multi-segment (henceforth *MSG*) queries. 62.5% of the queries (59% of the SSG and 72% of the MSG) contain at least one NE. Figure 3 (top) provides segment and query level NE statistics. The middle part of the figure shows the proportion of segments and queries that start with an NE. Finally, the bottom part of the table provides word level statistics. The figures clearly demonstrate the prominence of NEs

Figure 3: Top: Query and segment level NE distribution (numbers are averages across the queries in the relevant subset). Middle: NE type distribution (the ”only names” column refers to queries that consist of named entities only, and the ”any name” column refers to queries that contain at least one named entity). Bottom: The percentage of queries and segments that start with an NE.

in queries and the segmentation signal they provide. For example, as many as 35.4% of segments within the MSG queries start with an NE (42% of the first segments and 30% of the other segments). Finally, Figure 4 presents three example queries where NEs provide invaluable cues about the syntactic structure.

Now that we have established the importance of NEs for query parsing, we are ready to describe our entity-aware query parser.

5 A Segmentation and Entity-Aware BiLSTM Parser

Here we describe the integration of NE signals in our segmentation-aware query parser (§ 3.2).

A BiLSTM NE tagger Our NE tagger is a BiLSTM with an MLP classification layer, very similar to our independent segmentation model (SEG, § 3.3). We denote the MLP's hidden state with $h^{ne}(w_i)$ and its output scores vector with $o^{ne}(w_i)$. The model equations are:

$$\begin{aligned} h^{ne}(w_i) &= tanh(W_n^1 \cdot v_i + b_n^1) \\ o^{ne}(w_i) &= W_n^2 \cdot h^{ne}(w_i) + b_n^2 \end{aligned} \tag{6}$$

The margin-based loss (MBL) function we use in this model is:

$$\begin{aligned} MBL = max(0, 1 - MLP_\theta(w_i)[ne_{correct}] + \\ MLP_\theta(w_i)[ne_{predicted}]) \end{aligned} \tag{7}$$

where $\theta = (W_n^1, b_n^1, W_n^2, b_n^2)$ are the model parameters, $ne_{predicted}$ is the named entity type predicted by model and $ne_{correct}$ is the gold named entity type. $MLP_\theta(w_i)[ne_i]$ is the score given by the MLP to the ne_i named entity type.

NE-aware parsing We consider two methods for integrating information from the NE tagger into the parser. In both methods, we construct a new feature representation for each input word, denoted with v_i^M, where M stands for the integration method. The new word representations are then used in the configuration representation (§ 3).

The first method, denoted with *Hi* (for Hidden), uses the hidden vector $h^{ne}(w_i)$:

$$v_i^{Hi} = v_i \circ h^{ne}(w_i)$$

The second method, denoted with *Fi* (for Final), uses the NE embeddings:

$$v_i^{Fi} = v_i \circ e(ne_{predicted}(w_i))$$

For both methods v_i is the word representation generated by the parser (equation 2). For v_i^{Fi}, $ne_{predicted}(w_i)$ is the named entity type predicted by the tagger for the word w_i and $e(ne_{predicted}(w_i))$ is its embedding (part of the parser parametrization).

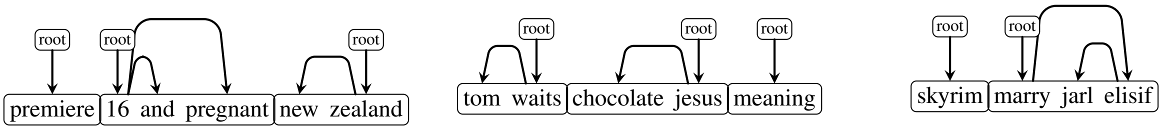

(a) "16 and pregnant" is a name of a television series; "new zealand" is a name of a place.

(b) "tom waits" is a name of a person; "chocolate jesus" is a name of a song.

(c) "skyrim" is a name of a computer game; "jarl elisif" is a name of a character in the game.

Figure 4: Three example queries from the PRS16 dataset, along with their parse trees. NEs provide an important signal about the structure of the queries.

Tagger and parser training We consider two approaches:

(a) Independent training: First, the tagger is trained with the gold NEs of the training set (§ 4), then the tagger is applied to the training set, and finally the parser is trained with the gold parse trees and the tagger's NE tagging of the training set.

(b) Joint training: The parameters of both models are updated together, each update is taking place after observing a single input sentence. In this joint model, the parser and the tagger are both using the same BiLSTM to learn the word representation v_i. The loss function of this model is the sum of the losses of the parser (sum of the stepwise losses of Eq. 4) and the tagger (Eq. 7):

$$MBL_{joint} = MBL_{parser} + MBL_{ner} \quad (8)$$

6 Experiments

Task and data Our task is the query parsing task of PRS16, but unlike them we do not use millions of unannotated queries. Instead, we experiment with a supervised setup were the parser is trained on parsed web queries and no unannotated queries are used. For our experiments we randomly split the PRS16 dataset of 4000 queries annotated with dependency structures and POS tags,[5] into train, development and test sections. This split is done so that: (a) The train set consists of 2500 queries while the dev and the test sets consist of 750 queries each; (b) For each $k \geq 1$, k-segment queries are split between the three sets so that to keep the 2500:750:750 proportion. As a result, the train, dev and test sets contain 618, 185 and 186 MSG queries, respectively, for the total of 989 MSG queries.

We consider the evaluation measures of PRS16: (a) The standard dependency parsing Unlabeled Attachment Score (UAS); and (b) Segmentation F1-score, where a segment is considered correct if both its start and end point are correctly identified.

Hyper-parameter	Value
Word embedding dim.	100, 200, 300
POS tag embedding dim.	25, 50, 100
Named entity embedding dim.	6
Hidden units in MLP	100
BiLSTM hidden dim.	50,125,200
BiLSTM output dim.	125
α (for word dropout)	0.25
p_{agg} (for exploration training)	1

Table 2: The hyper-parameters considered in our development data experiments.

Models and baselines We experiment with three model families: (a) The baseline KG16 parser and our arc-eager variant of the parser (§ 3.1, § 3.2); (b) Our segmentation-aware parsers (§ 3.2, § 3.3); and (c) Our segmentation and entity-aware parsers (§ 5) where the parser and the NE tagger are trained either jointly or independently (we also consider the integration of NE information into the original (arc-hybrid) KG16 parser).[6]

Hyper-parameter tuning Following (Kiperwasser and Goldberg, 2016) and due to the large number of models we experiment with, we consider a relatively small grid of hyper-parameter values, focused around the values chosen by these authors, as described in Table 2.

To avoid a very large number of experiments, we tune the parameters for the original KG16 arc-hybrid model (P^H) and for our arc-eager version of the parser (P^E). We then report test-set results for the P^H model with its tuned hyper-parameters, and for all the other models with the hyper-parameters that were estimated for the P^E model. While this setup gives an unfair advantage

---

[5] `webscope.sandbox.yahoo.com` (dataset L-28)

[6] A comment about naming conventions: unless H (for the arc-hybrid KG16 parser) or E (for the arc-eager KG16 parser) is part of the model name, a model is an extension of our P^B parser.

for the baseline P^H model, it helps us avoid an expensive model-specific tuning process. The auxiliary segmentation models (§ 3.3) and the NE tagger (§ 5) use the hyper-parameters of Table 2, but for these models we do not perform any tuning – for each hyper-parameter with more than one option, we use the leftmost number from the table.

7 Results

Our results are presented in Table 3. We focus on selected members of each model family, and within each model family we focus on the simplest models (P^H, P^E and P^B, with segmentation and entity information when appropriate), and on the most complex ones. We make sure to include the best performing model of each category, which happens to be one of the most complex models for all families, emphasizing the quality of our modeling choices. The results for the full list of models are in the spp. material.

The *Baseline parsers* section of the table demonstrates the impact of moving from the arc-hybrid variant to an arc-eager variant of KG16, to better support segmentation. While the P^H model performs slightly better than P^E in terms of UAS (78.3 vs. 77.0), the segmentation F1-score of P^E is 4.3 points better (72.0 vs. 67.7). Interestingly, this improvement is not achieved through better segmentation of MSG queries, but by avoiding unnecessary segmentation decisions on SSG queries.

The *segmentation-aware parsers* section of the table shows that further extending the arc-eager KG16 parser to explicitly account for segmentation results in substantial segmentation improvements. Particularly, the overall F1-score of P^B – our segmentation-aware parser that does not use information from any auxiliary segmentation models – is as high as 77.4. This amounts to 5.4 and 9.7 additional F1 points compared to the arc-eager and the original arc-hybrid KG16 parsers, respectively. Information from auxiliary segmentation models (P^{S+Fl} and P^{FS+Fl}) does not substantially increase performance in this family.

When considering entity information, the performance of our models and of the P^E baseline substantially improves. However, while the UAS of the P^E baseline increases to 80.2 (P^{E+Fi} with independent training), its segmentation F1-score does not cross the 66.9 bound (P^{E+Hi} with joint training).[7] The gain of the segmentation-aware parser from entity information is much more substantial. First, regardless of how the entity information is integrated into the model (Hi vs. Fi) and of whether the auxiliary segmentation models (S, FS and Fl) are used or not, the model perfectly segments the SSG queries. Moreover, it demonstrates substantial performance boosts with respect to all measures. Our best performing model, $P^{S+Fl+Fi}$ with joint training (bold result in the bottom model section of the table) improves the original KG16 parser (P^H, top row of the table) by 6.5 UAS points (83.5 vs. 77.0) and by 16.8 segmentation F1 scores (84.5 vs. 67.7).

Model	UAS			Seg. F1-score		
	all	msg	ssg	all	msg	ssg
Baseline parsers						
P^H	**77.0**	**74.9**	**78.3**	67.7	**51.5**	77.9
P^E	75.4	70.5	77.0	**72.0**	47.6	**87.6**
Seg. aware parsers - no entity information						
P^B	76.7	73.4	77.8	**77.4**	48.7	**95.6**
P^{S+Fl}	**77.1**	**74.1**	**78.1**	76.0	**53.3**	90.8
P^{FS+Fl}	76.6	72.1	78.0	**77.4**	52.1	93.8
Seg. and entity parsers - independent training						
P^{E+Hi}	80.2	70.3	83.3	63.3	34.3	80.3
P^{E+Fi}	80.2	70.8	83.2	61.4	42.0	73.2
P^{Hi}	79.3	68.3	82.8	63.3	40.5	**100**
P^{Fi}	80.9	69.5	84.5	79.1	45.2	**100**
$P^{S+Fl+Hi}$	80.6	70.5	83.8	77.0	48.3	**100**
$P^{S+Fl+Fi}$	81.2	71.5	84.2	80.6	49.0	**100**
$P^{FS+Fl+Hi}$	**81.6**	**72.3**	**84.6**	80.1	**54.1**	**100**
$P^{FS+Fl+Fi}$	80.6	70.4	83.9	**81.7**	52.8	**100**
Seg. and entity parsers - joint training						
P^{E+Hi}	79.3	66.7	83.4	66.9	26.2	90.2
P^{E+Fi}	79.8	67.7	83.6	66.1	36.9	83.6
P^{Hi}	79.7	66.5	83.4	75.1	33.1	**100**
P^{Fi}	81.6	70.5	85.1	78.6	43.7	**100**
$P^{S+Fl+Hi}$	81.6	71.9	84.6	82.0	53.4	**100**
$P^{S+Fl+Fi}$	**83.5**	**73.2**	**86.7**	**84.5**	**60.7**	**100**
$P^{FS+Fl+Hi}$	79.6	67.3	83.5	76.0	36.1	**100**
$P^{FS+Fl+Fi}$	82.4	72.1	85.7	81.2	51.4	**100**

Table 3: Results. Best numbers within each model section are highlighted in bold. H and E stand for the arc-hybrid and arc-eager KG16 parser, while B is our segmentation aware parser without any auxiliary segmentation model or NE information. All models where these letters do not appear refer to extensions of our segmentation-aware parser (B). S: independent segmentation model. FS: full independent segmentation model. Fl: configuration based segmentation model. Hi: NE aware parser (Hidden). Fi: NE aware parser (Final).

Overall, joint training of the parsing and NER

---

[7] The UAS numbers of the p^H models with entity information are similar to those of the p^E models, but their segmentation quality is lower. Due to space limitations, we do not provide these numbers.

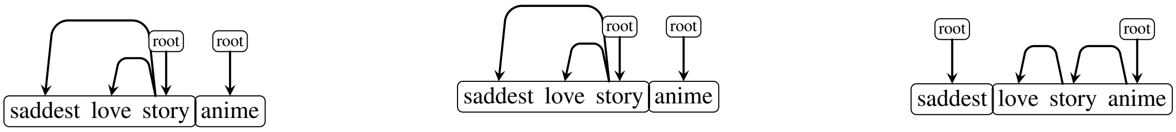

(a) Gold tree.

(b) The P^B parser (perfect) tree.

(c) The P^H parse tree.

Figure 5: Example multi-segment query where P^B succeeds and P^H fails.

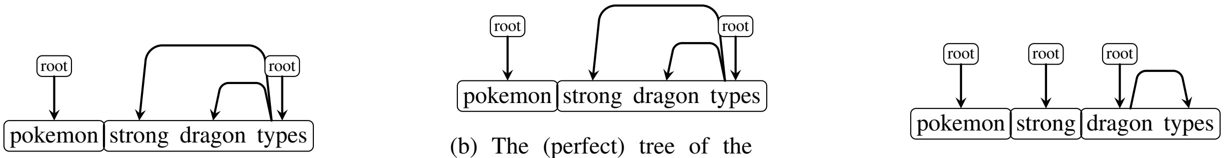

(a) Gold tree.

(b) The (perfect) tree of the $P^{S+Fl+Fi}$ parser when trained jointly with the named entity tagger.

(c) The P^{S+Fl} parse tree.

Figure 6: Example multi-segment query where the $P^{S+Fl+Fi}$ parser succeeds when trained jointly with the named entity tagger, and P^{S+Fl} fails.

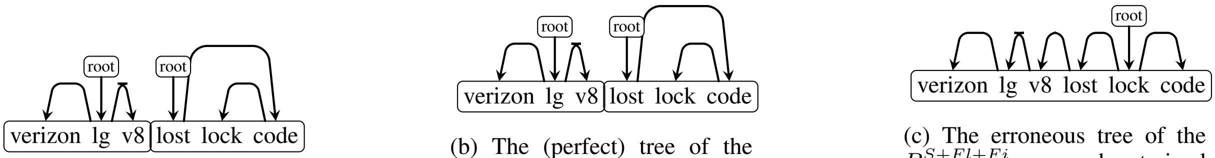

(a) Gold tree.

(b) The (perfect) tree of the $P^{S+Fl+Fi}$ parser when trained jointly with the named entity tagger.

(c) The erroneous tree of the $P^{S+Fl+Fi}$ parser when trained independently of the named entity tagger.

Figure 7: Example multi-segment query where the $P^{S+Fl+Fi}$ parser succeeds when trained jointly with the named entity tagger, and fails when using a pre-trained named entity tagger.

models improves over independent training for both segmentation F1 and parsing UAS (in 5 of 8 cases for each measure, two bottom model sections of the table). This improvement comes mostly from SSG queries (7 of 8 UAS cases and the 2 cases where segmentation F1 could improve), but at the cost of some degradation on MSG queries. Moreover, our best model is a jointly trained one.

While the goal of this paper is mostly to improve the syntactic analysis of web queries, our simple NE tagger provides decent results. When trained independently of the parser its test-set (labeled) micro-F1 score is 85.6. When trained jointly with the parser it achieves similar scores in some cases: e.g. when jointly trained with the best performing parsing model, $P^{S+Fl+Fi}$, it achieves a micro-F1 of 85.2. Yet, in other cases such as joint training with P^{S+H} and P^{S+Fi} its micro-F1 drops to 78.0 and 78.5, respectively.

Finally, figure 5-7 provide some qualitative analysis of our models and baselines.

8 Conclusions

We presented a new BiLSTM transition-based parser for web queries. Our parser is the first that explicitly accounts for the forest-based query grammar of PRS16. Moreover, we demonstrated the importance of NEs for understanding the syntactic structure of web queries, annotated the Query Treebank of PRS16 with NEs, and demonstrated how to effectively use NE information in the syntactic parsing of web queries.

In future work we intend to explore methods for closing the performance gap our algorithms still have for MSG queries (both UAS and segmentation F1) and for SSG queries (UAS only). Relevant directions include improving the transition logic of our parser, the BiLSTM NE model and the interactions between the two models.

Acknowledgements

We would like to thank Yuval Pinter, Daniel Hershcovich and the Technion NLP group members for their valuable feedback.

References

Areej Alasiry, Mark Levene, and Alexandra Poulovassilis. 2012. Detecting candidate named entities in search queries. In *Proceedings of SIGIR*.

James Allan and Hema Raghavan. 2002. Using part-of-speech patterns to reduce query ambiguity. In *Proceedings of SIGIR*.

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of english web-search queries. In *Proceedings of EMNLP*.

Michael Bendersky, W Bruce Croft, and David A Smith. 2010. Structural annotation of search queries using pseudo-relevance feedback. In *Proceedings of CIKM*.

Michael Bendersky, W Bruce Croft, and David A Smith. 2011. Joint annotation of search queries. In *Proceedings of ACL*.

Shane Bergsma and Qin Iris Wang. 2007. Learning noun phrase query segmentation. In *Proceedings of EMNLP-CoNLL*.

Andreas Eiselt and Alejandro Figueroa. 2013. A two-step named entity recognizer for open-domain search queries. In *Proceedings of IJCNLP*.

Jenny Rose Finkel and Christopher D Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of NAACL-HLT*.

Jenny Rose Finkel and Christopher D Manning. 2010. Hierarchical joint learning: Improving joint parsing and named entity recognition with non-jointly labeled data. In *Proceedings of ACL*.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, Josef Van Genabith, et al. 2011. # hardtoparse: Pos tagging and parsing the twitterverse. In *proceedings of the Workshop On Analyzing Microtext (AAAI 2011)*.

Kuzman Ganchev, Keith Hall, Ryan McDonald, and Slav Petrov. 2012. Using search-logs to improve query tagging. In *Proceedings of ACL (Volume 2: Short Papers)*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTAT*.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association of Computational Linguistics* 1:403–414.

Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *Proceedings of SIGIR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of EMNLP*.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of ACL*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*.

Xiao Li. 2010. Understanding the semantic structure of noun phrase queries. In *Proceedings of ACL*.

Yijia Liu, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider, and Noah A. Smith. 2018. Parsing tweets into universal dependencies. In *Proceedings of NAACL*.

Mehdi Manshadi and Xiao Li. 2009. Semantic tagging of web search queries. In *Proceedings of the ACL-IJCNLP*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics* 34(4):513–553.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of ACL*.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Yuval Pinter, Roi Reichart, and Idan Szpektor. 2016. Syntactic parsing of web queries with question intent. In *Proceedings of NAACL*.

Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *Proceedings of ACL-08: HLT*.

Gilad Tsur, Yuval Pinter, Idan Szpektor, and David Carmel. 2016. Identifying web queries with question intent. In *Proceedings of WWW*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. *OntoNotes Release 5.0*. https://catalog.ldc.upenn.edu/LDC2013T19.

Ryen W. White, Matthew Richardson, and Wen-tau Yih. 2015. Questions vs. queries in informational search tasks. In *Proceedings of WWW*.

Ke Zhai, Zornitsa Kozareva, Yuening Hu, Qi Li, and Weiwei Guo. 2016. Query to knowledge: Unsupervised entity extraction from shopping queries using adaptor grammars. In *Proceedings SIGIR*.

A Parser correctness

In this appendix we discuss the correctness of our segmentation-aware dependency parser, as described in §3.2 of the main paper.

According to Nivre (2008), a parsing system is correct for a class $\mathbb{G}$ of dependency graphs if and only if it is sound and complete for $\mathbb{G}$. We prove correctness for the set $\mathbb{G}$ of all possible dependency trees. The system is *sound* if and only if for every sentence x and every transition sequence $C_{0,m}$ for x, it holds that $G_{C_{0,m}} \in \mathbb{G}$. The system is *complete* for the class $\mathbb{G}$ of all possible dependency trees if and only if for every sentence x and every dependency graph $G_x \in \mathbb{G}$, there is a transition sequence $C_{0,m}$ for x such that $G_{C_{0,m}} = G_x$.

We prove that our segmentation-aware parser is sound by showing that its resulting dependency graph is necessarily a valid parse tree. In order to do that we show that in every possible output graph of the parser each word has exactly one head.

We start by showing that a word must have at least one head. This stems from the fact that a word processing is completed once it is reduced. This can be done using the $REDUCE$ transition or the $LEFT_{arc}$ transition. The preconditions of the $REDUCE$ transition prevent a node from being reduced without a head. The $LEFT_{arc}$ transition, by definition, reduces the node after assigning it with a head. There can be no more than one head to each node as the arcs are created using the $RIGHT_{arc}$ and $LEFT_{arc}$ transitions. If the $RIGHT_{arc}$ transition is applied to a word, it inserts the node to the stack. Once a node is in the stack, $RIGHT_{arc}$ cannot be applied to it and the $LEFT_{arc}$ preconditions prevent the generation of a second head. If the $LEFT_{arc}$ transition applies to the word, the word is automatically reduced and cannot have additional heads, as discussed above.

To prove completeness we develop a valid transition sequence that produces any given dependency tree G_x for a sentence x. We denote the transition sequence that produces a dependency graph G_x for a sentence x in the arc-eager system as $C^{eager}_{0,m}(x, G_x)$. $C^{eager}_{0,m}(x, G_x)$ exists according to the completeness of the arc-eager system. We also denote the sub-graph of a segment s within a dependency tree G_x as G^s_x. An input sentence contains one or more segments. We construct the transition sequence by performing $|s|$ $PushToSeg$ transitions for every segment s in x followed by performing $C^{eager}_{0,m'}(s, G^s_x)$ with the stack σ and the buffer β_2. This transition sequence produces a valid tree G_x for sentence x.

An Investigation of the Interactions Between Pre-Trained Word Embeddings, Character Models and POS Tags in Dependency Parsing

Aaron Smith **Miryam de Lhoneux** **Sara Stymne** **Joakim Nivre**
Department of Linguistics and Philology, Uppsala University

Abstract

We provide a comprehensive analysis of the interactions between pre-trained word embeddings, character models and POS tags in a transition-based dependency parser. While previous studies have shown POS information to be less important in the presence of character models, we show that in fact there are complex interactions between all three techniques. In isolation each produces large improvements over a baseline system using randomly initialised word embeddings only, but combining them quickly leads to diminishing returns. We categorise words by frequency, POS tag and language in order to systematically investigate how each of the techniques affects parsing quality. For many word categories, applying any two of the three techniques is almost as good as the full combined system. Character models tend to be more important for low-frequency open-class words, especially in morphologically rich languages, while POS tags can help disambiguate high-frequency function words. We also show that large character embedding sizes help even for languages with small character sets, especially in morphologically rich languages.

1 Introduction

The last few years of research in natural language processing (NLP) have witnessed an explosion in the application of neural networks and word embeddings. In tasks ranging from POS tagging to reading comprehension to machine translation, a unique dense vector is learned for each word type in the training data. These word embeddings have been shown to capture essential semantic and morphological relationships between words (Mikolov et al., 2013), and have precipitated the enormous success of neural network-based architectures across a wide variety of NLP tasks (Plank et al., 2016; Dhingra et al., 2017b; Vaswani et al., 2017).

When task-specific training data is scarce or the morphological complexity of a language leads to sparsity at the word-type level, word embeddings often need to be augmented with sub-word or part-of-speech (POS) tag information in order to release their full power (Kim et al., 2016; Sennrich et al., 2016; Chen and Manning, 2014). Initialising vectors with embeddings trained for a different task, typically language modelling, on huge unlabelled corpora has also been shown to improve results significantly (Dhingra et al., 2017a). In dependency parsing, the use of character (Ballesteros et al., 2015) and POS (Dyer et al., 2015) models is widespread, and the majority of parsers make use of pre-trained word embeddings (Zeman et al., 2017).

While previous research has examined in detail the benefits of character and POS models in dependency parsing and their interactions (Ballesteros et al., 2015; Dozat et al., 2017), there has been no systematic investigation into the way these techniques combine with the use of pre-trained embeddings. Our results suggest a large amount of redundancy between all three techniques: in isolation, each gives large improvements over a simple baseline model, but these improvements are not additive. In fact combining any two of the three methods gives similar results, close to the performance of the fully combined system.

We set out to systematically investigate the ways in which pre-trained embeddings, character and POS models contribute to improving parser quality. We break down results along three dimensions—word frequency, POS tag, and language—in order to tease out the complex interactions between the three techniques. Our main findings can be summarized as follows:

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2711–2720
Brussels, Belgium, October 31 - November 4, 2018.

- For all techniques, improvements are largest for low-frequency and open-class words and for morphologically rich languages.
- These improvements are largely redundant when the techniques are used together.
- Character-based models are the most effective technique for low-frequency words.
- Part-of-speech tags are potentially very effective for high-frequency function words, but current state-of-the-art taggers are not accurate enough to take full advantage of this.
- Large character embeddings are helpful for morphologically rich languages, regardless of character set size.

2 Related Work

Chen and Manning (2014) introduced POS tag embeddings: a learned dense representation of each tag designed to exploit semantic similarities between tags. In their greedy transition-based parser, the inclusion of these POS tag embeddings improved labelled attachment score (LAS) by 1.7 on the English Penn Treebank (ETB) and almost 10 on the Chinese Penn Treebank (CTB). They also tested the use of pre-trained word embeddings for initialisation of word vectors, finding gains of 0.7 for PTB and 1.7 for CTB.

Dyer et al. (2015) in their Stack Long Short-Term Memory (LSTM) dependency parser, show that POS tag embeddings in their architecture improve LAS by 0.6 for English and 6.6 for Chinese. Unlike Chen and Manning (2014), they do not use pre-trained word embeddings for initialisation, instead concatenating them as a fixed vector representation to a separate randomly-initialised learned representation. This leads to improvements in LAS of 0.9 and 1.6 of English and Chinese, respectively.

Following on from the work of Dyer et al. (2015), Ballesteros et al. (2015) introduced the first character-based parsing model. They found that a model based purely on character information performed at the same level as a model using a combination of word embeddings and POS tags. Combining character and POS models produced even better results, but they conclude that POS tags are less important for character-based parsers. They also showed that character models are particularly effective for morphologically rich languages, but that performance remains good in languages with little morphology, and that character models help substantially with out-of-vocabulary (OOV) words, but that this does not fully explain the improvements they bring. The use of pre-trained embeddings was not considered in their work.

Kiperwasser and Goldberg (2016), in the transition-based version of their parser based on BiLSTM feature extractors, found that POS tags improved performance by 0.3 LAS for English and 4.4 LAS for Chinese. Like Dyer et al. (2015), they concatenate a randomly-initialised word embeddings to a pre-trained word vector; however in this case the pre-trained vector is also updated during training. They find that this helps LAS by 0.5–0.7 for English and 0.9–1.2 for Chinese, depending on the specific architecture of their system.

Dozat et al. (2017), building on the graph-based version of Kiperwasser and Goldberg (2016), confirmed the relationship between character models and morphological complexity, both for POS tagging and parsing. They also examined the importance of the quality of POS tags on parsing, showing that their own tagger led to better parsing results than a baseline provided by UDPipe v1.1 (Straka et al., 2016).

3 The Parser

We use and extend UUParser[1] (de Lhoneux et al., 2017a; Smith et al., 2018), a variation of the transition-based parser of Kiperwasser and Goldberg (2016) (K&G). The K&G architecture can be adapted to both transition- and graph-based dependency parsing, and has quickly become a *de facto* standard in the field (Zeman et al., 2017). In a K&G parser, BiLSTMs (Hochreiter and Schmidhuber, 1997; Graves, 2008) are employed to learn useful representations of tokens in context. A multi-layer perceptron (MLP) is trained to predict transitions and possible arc labels, taking as input the BiLSTM vectors of a few tokens at a time. Crucially, the BiLSTMs and MLP are trained together, enabling the parser to learn very effective token representations for parsing. For further details we refer the reader to Nivre (2008) and Kiperwasser and Goldberg (2016), for transition-based parsing and BiLSTM feature extractors, respectively.

Our version of the K&G parser is extended with a SWAP transition to facilitate the construction

---

[1] `https://github.com/UppsalaNLP/uuparser`

of non-projective dependency trees (Nivre, 2009). We use a static-dynamic oracle to allow the parser to learn from non-optimal configurations at training time in order to recover better from mistakes at test time, as described in de Lhoneux et al. (2017b).

In this paper we experiment with a total of eight variations of the parser, where the difference between each version resides in the vector representations x_i of word types w_i before they are passed to the BiLSTM feature extractors (see Section 3 of Kiperwasser and Goldberg (2016)). In the simplest case, we set x_i equal to the word embedding $e^r(w_i)$:

$$x_i = e^r(w_i)$$

The superscript r refers to the fact that the word embeddings are initialised randomly at training time. This is the setup in our BASELINE system.

For our +CHAR system, the word embedding $e^r(w_i)$ is concatenated to a character-based vector, obtained by running a BiLSTM over the characters $ch_{1:m}$ of w_i:

$$x_i = e^r(w_i) \circ \text{BiLSTM}(ch_{1:m})$$

In the +POS setting, the word embedding is instead concatenated to an embedding $p(w_i)$ of the word's universal POS tag (Nivre et al., 2016):

$$x_i = e^r(w_i) \circ p(w_i)$$

This scenario necessitates knowledge of the POS tag of w_i; at test time, we therefore need a POS tagger to provide predicted tags.

In another version of our parser (+EXT), pre-trained embeddings are used to initialise the word embeddings.[2] We use the superscript t to distinguish these from randomly initialised vectors:

$$x_i = e^t(w_i)$$

We use the embeddings that were released as part of the 2017 CoNLL Shared Task on Universal Dependency Parsing (CoNLL-ST-17) (Zeman et al., 2017). Words in the training data that do not have pre-trained embeddings are initialised randomly. At test time, we look up the updated embeddings for all words seen in the training data; OOV words are assigned their un-updated pre-trained embedding where it exists, otherwise a learnt OOV vector.

In our COMBINED setup, we include pre-trained embeddings along with the character vector and POS tag embedding:

$$x_i = e^t(w_i) \circ \text{BiLSTM}(ch_{1:m}) \circ p(w_i)$$

The three remaining versions of the vector x_i constitute all possible combinations of two techniques of pre-trained embeddings, the character model and POS tags. We refer to these versions of the parser as −EXT, −CHAR, and −POS, respectively.

4 Experimental setup

4.1 Data

We ran our experiments on nine treebanks from Universal Dependencies (Nivre et al., 2016) (v2.0): Ancient Greek PROIEL, Arabic, Chinese, English, Finnish, Hebrew, Korean, Russian and Swedish. Inspired partially by de Lhoneux et al. (2017c), these treebanks were chosen to reflect a diversity of writing systems, character set sizes, and morphological complexity. As error analysis is carried out on the results, we perform all experiments on the *dev* data sets.

Table 1 shows some statistics of each treebank. Of particular note are the large character set sizes in Chinese and Korean, an order of magnitude bigger than those of all other treebanks. The high type-token ratio for Finnish, Russian and Korean also stands out; this is likely due to the high morphological complexity of these languages.

Treebank	Sentences		TTR	Chars
Ancient Greek	14864	1019	0.15	179
Arabic	6075	909	0.10	105
Chinese	3997	500	0.16	3571
English	12534	2002	0.07	108
Finnish	12217	1364	0.26	244
Hebrew	5241	484	0.11	53
Korean	4400	950	0.46	1730
Russian	3850	579	0.30	189
Swedish	4303	504	0.16	86

Table 1: Treebank statistics. Number of sentences in *train* and *dev* sets, type-token ratio (TTR), and character set size.

4.2 Parser settings

The parser is trained three times for each language with different random seeds for 30 epochs each. At the end of each epoch we parse the *dev* data

---

[2] This strategy proved more successful in preliminary experiments than others for incorporating pre-trained embeddings discussed in Section 2.

Word embedding size	100
Character embedding size	500
Character BiLSTM output size	100
POS tag embedding size	20

Table 2: Embedding sizes.

and calculate LAS. For each training run, results are averaged over the five best epochs for each language. In this way, we attempt to make our results more robust to variance due to randomness in the training procedure.[3] Our macro-averaged scores are based on a total of 135 different epochs (3 random seeds × 5 best epochs × 9 languages).

Table 2 shows the embedding sizes we found to produce best results in preliminary work and which we use in all experiments in Section 5. Note our unusually large character embedding size; we will discuss this in more detail in Section 6. We use predicted UPOS tags from the system of Dozat et al. (2017) for experiments with POS tags,[4] other than in Section 7 where we compare results with different taggers and gold POS tags, in order to set a ceiling on the potential gains from a perfect POS tagger. For all other hyperparameters we use default values (Smith et al., 2018).

4.3 Analysis

The hypothesis underlying our choice of analysis is that the three techniques under study here—pretrained embeddings, character vectors and POS tag embeddings—affect words differently depending on their frequencies, POS tags, and the language of the sentence. We do not claim this to be an exhaustive list; many other dimensions of analysis are clearly possible (dependency relation would be another obvious choice for example), but we believe that these are likely to be three of the most informative factors. In the frequency and POS tag cases, we want to examine the overall contribution to LAS of words from each category. We expect changing the representation of a token to affect how likely it is to be assigned the correct head in the dependency tree, but also how likely it is to be assigned correctly as the head of other words. We thus introduce a new metric for this part of the analysis: the head and dependents labelled attachment score, which we refer to as HDLAS.

When calculating HDLAS, the dependency analysis for a given token is only considered correct if the token has the correct labelled head *and* the complete set of correctly labelled dependents. This is a harsher metric than LAS, which only considers whether a token has the correct labelled head. Note that when calculating HDLAS for all tokens in a sentence, each dependency relation is counted twice, once for the head word and once for the dependent. It only makes sense to use this metric when analysing individual tokens in a sentence, or when grouping tokens into different categories across multiple sentences.

4.3.1 Frequency

In this analysis, we first label each token in the *dev* data for each language by its relative frequency in the *train* data, with add-one smoothing.[5] Frequency categories are created by rounding the log relative frequency down to the nearest integer. We calculate the HDLAS for each frequency category for each language, before macro-averaging the results across the nine languages to produce a final score for each frequency class.

4.3.2 POS tag

In this case, we label each word from the *dev* data by its gold POS tag, before calculating HDLAS for each category and taking the macro average across languages. Here the total number of tokens in each category varies across several orders of magnitude: the most common category NOUNs make up 26.0% of all words, while the smallest class SYM represents just 0.1%. For this reason, and to make our graphs more readable, we do not show results for the six smallest categories: INTJ, NUM, PART, SCONJ, SYM, and X.

4.3.3 Language

Here we consider LAS directly for each language; the HDLAS metric used in the previous two sections is not relevant as all tokens in a given sentence are assigned to the same category determined by the language of the sentence.

5 Results

Table 3 gives the LAS for each of the eight systems described in Section 3. We observe that pre-

---

[3] Changing the random seed has been shown to produce results that appear statistically significant different in neural systems (Reimers and Gurevych, 2017).

[4] Available at `https://web.stanford.edu/~tdozat/`.

[5] The smoothing ensures that OOV tokens, those that appear in *dev* but not *train*, are not assigned zero frequency; this alleviates the problem of taking log(0) in the subsequent conversion to log relative frequency.

BASELINE	67.7	COMBINED	81.0
+EXT	76.1	−EXT	79.9
+CHAR	78.3	−CHAR	79.2
+POS	75.9	−POS	80.3

Table 3: Mean LAS across nine languages for a baseline system employing randomly-initialised word embeddings only, compared to three separate systems using pre-trained word embeddings (+EXT), a character model (+CHAR), and POS tags (+POS). Scores are also shown for a combined system that utilises all three techniques and corresponding systems where one of the three techniques is ablated (−EXT, −CHAR and −POS).

trained embeddings (+8.4), the character model (+10.6) and POS tags (+8.2) all give large improvements in LAS over the baseline system. The combined system is the best overall, but the improvement of 13.3 LAS is far from the sum of its components. Employing two of the techniques at a time reduces LAS by only 0.7–1.8 compared to the combined system.

5.1 Frequency

Fig. 1 and Fig. 2 compare systems by word frequency. As expected, accuracy improves with frequency for all systems: the parser does better with words it sees more often during training. There is a levelling off for the highest frequency words, probably due to the fact that these categories contain a small number of highly polysemous word types.

Fig. 1 demonstrates a clear trend in the improvement achieved by each of the individual techniques over the baseline, with larger gains for lower frequency words. This confirms a result from Ballesteros et al. (2015), who found that character models help substantially with OOV words. We can generalise this to say that character models improve parsing quality most for low frequency words (including OOV words), and that this is also true, albeit to a slightly lesser effect, of POS tags and pre-trained word embeddings. It is notable however that HDLAS increases universally across all frequency classes: even the highest frequency words benefit from enhancements to the basic word representation.

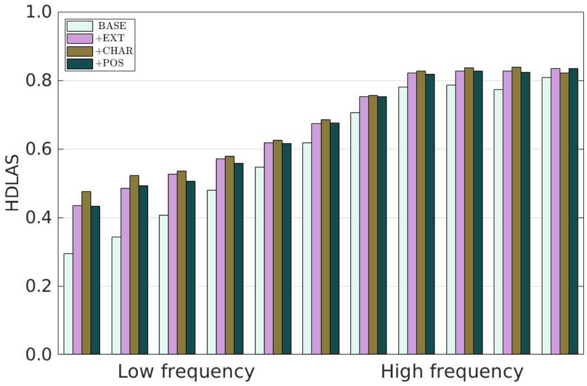

Figure 1: BASELINE system compared to pre-trained embeddings (+EXT), character model (+CHAR) and POS tags (+POS).

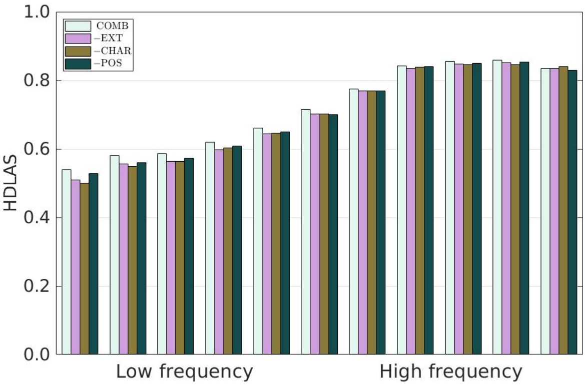

Figure 2: COMBINED system compared to ablated systems where pre-trained embeddings (−EXT), character models (−CHAR) and POS tags (−POS) are removed.

What immediately stands out in Fig. 2 is that for mid- and high frequency words, there is little difference in HDLAS between different combinations of two of the three techniques, and for the highest frequency words this is at a level almost indistinguishable from the full COMBINED system. The slight improvements we see for COMBINED in Table 3 compared to the three ablated systems thus principally also come from the low-frequency range.

5.2 POS tags

In Fig. 3 systems are compared by POS tag. We observe a universal improvement across all POS tags for each of the three variations of the system compared to the baseline. However, it is notable that the biggest gains in HDLAS are for open word classes: NOUNs, VERBs and ADJs. As these make up a large overall proportion of words, these differences have an overall relatively large impact on LAS.

For the most frequent POS categories NOUN and VERB we again see a clear victory for the character model (note that while these POS categories are frequent, they contain a large number of low-frequency words). Overall the character model succeeds best for the open-class POS categories, while having the right POS tag is marginally better for closed-class categories such as DET, CCONJ, and AUX. It is interesting that the character model is not as strong for PROPN, despite the fact that these are open-class low-frequency words; for these words pre-trained embeddings are the best single technique. This may be due to the fact that the rules governing the composition of names at the character level are different from other words in the language.

It is perhaps surprising that the advantage of POS tag embeddings is not greater when it comes to auxiliary verbs, for example, where the distinction from main verbs can be difficult and crucial for a correct syntactic analysis. The reason probably lies in the fact that this distinction is equally difficult for the POS tagger. We will investigate this further in Section 7.

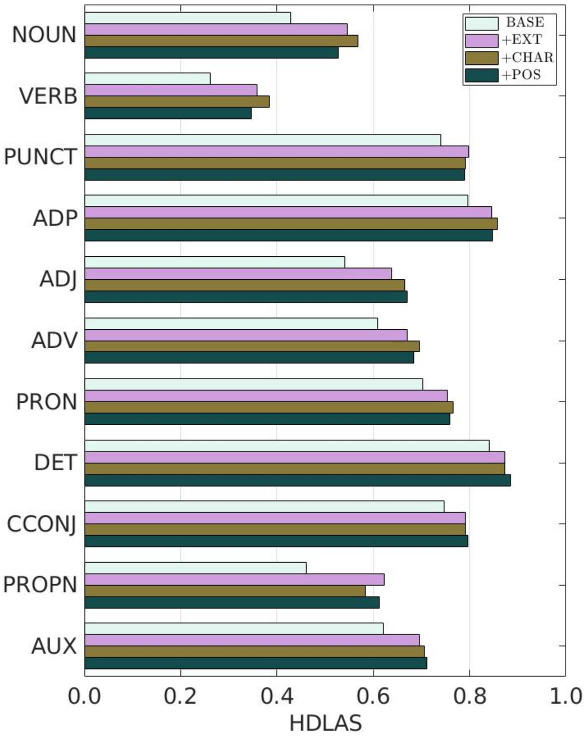

Figure 3: Comparison by POS tag of BASELINE system to +EXT, +CHAR, and +POS. Tags are sorted by frequency.

5.3 Language

Fig. 4 compares the systems by language. Once again improvement is universal for each system compared to the baseline. There are however substantial differences between languages. The three biggest overall improvements are for Finnish, Korean and Russian, with a particularly notable increase in the Korean case. This suggests that the baseline model struggles to learn adequate representations for each word type in these languages. These are the three languages we identified in Section 4.1 as having high type-token ratios in their training data. It is also notable that the character model becomes more important compared to other methods for these three languages. In fact, despite the overall superiority of the character model (see Table 3), it is only the best single technique for 4 of the 9 languages, the three already mentioned plus Ancient Greek.

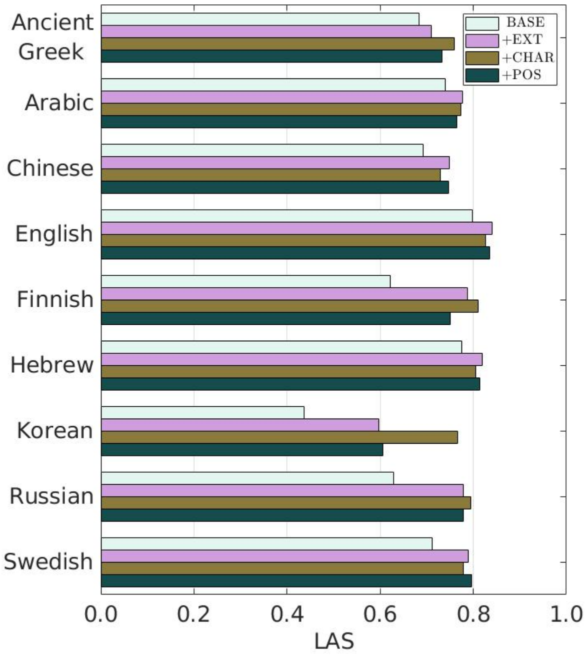

Figure 4: Comparison by language of BASELINE system to +EXT, +CHAR, and +POS.

6 Character Embedding Size

All results with character models observed thus far make use of a character embedding of dimension 500. This value is large compared to typical sizes used for character models (Kim et al., 2016; Ballesteros et al., 2015). A common belief is that larger character embedding sizes are justified for languages with larger character set sizes such as Chinese: in other words, the embedding size should be related to the number of entities being embedded (Shao, 2018).

In Table 4, we show how LAS varies with a few values of this hyperparameter when averaged across our nine-language sample. We see a steady

BASELINE	67.7	−CHAR	79.2
+CH-24	76.8	+CH-24	80.5
+CH-100	77.7	+CH-100	80.6
+CH-500	78.3	+CH-500	81.0

Table 4: Mean LAS across nine languages for BASELINE system compared to systems with character vectors of different sizes. Comparison also shown for systems employing pre-trained word vectors and POS tag embeddings.

improvement in LAS as the character embedding size increases, both when compared to a baseline with randomly initialised word embeddings only and when compared to a system that also employs pre-trained word vectors and POS tag embeddings.[6]

It is particularly interesting to break down the effects here by language. In Table 5 we show results for Chinese, Finnish, Korean and Russian. It is particularly striking that the larger character embeddings do not help for Chinese; the score for the largest character embedding size is actually marginally lower than a baseline without a character model at all. This is despite the fact that a small character embedding improves LAS, albeit marginally, suggesting that there is some useful information in the characters even when pre-trained embeddings and POS tags are present. Conversely, the large character models are very effective for Finnish, a treebank with a character set less than a tenth of the size of Chinese (see Table 1).

	−CHAR	+CH-24	+CH-100	+CH-500
Chinese	76.0	76.1	75.9	75.8
Finnish	81.9	83.7	83.8	84.7
Korean	70.1	78.0	78.2	79.4
Russian	82.0	81.4	81.5	82.5

Table 5: Comparison by language of different character embedding sizes.

We claim therefore that character set size is not in fact a good metric to use in determining character embedding sizes. Our tentative explanation is that while languages like Finnish have relatively small character sets, those characters interact with each other in much more complex ways, thus requiring larger embeddings to store all the necessary information. While there are many characters in Chinese, the entropy in the interactions between characters appears to be smaller, enabling smaller character embeddings to do just as good a job.

It is also worth noting from Tables 4 and 5 that, in the presence of POS tags and pre-trained embeddings, the improvement gained from increasing the character embedding size from 24 to 100 is small (0.1 LAS for Finnish, 0.2 for Korean, 0.1 for Russian; 0.1 on average across the nine treebanks). This perhaps gives the impression of diminishing returns; that going even larger is likely to lead to ever smaller improvements. This may be the reason that smaller character embeddings have generally been preferred previously. However, we in fact observe a much *greater* gain when increasing from 100 to 500 (0.9 for Finnish, 1.2 for Korean, 1.0 for Russian; 0.4 on average across the nine treebanks), suggesting that very large character embeddings are effective, and particularly useful for morphologically rich languages.

7 POS tagger

In this section we apply our POS tag analysis to the effect of the POS tagger used to produce tags at test time. We compare three setups: firstly using tags predicted by UDPipe (Straka and Straková, 2017), which was the baseline model for CoNLL-ST-2017, secondly using tags predicted by the winning Stanford system (Dozat et al., 2017), and thirdly using gold tags. Note that for the Stanford system, we train on gold tags and use predicted tags at test time, while for UDPipe we train on a jackknifed version of the train data with predicted tags that was released as part of CoNLL-ST-2017.

BASELINE	67.7	−POS	80.3
UDPipe	73.4	UDPipe	80.2
Stanford	75.9	Stanford	81.0
Gold	78.4	Gold	83.8

Table 6: Mean LAS across nine languages for BASELINE system compared to systems with POS tags predicted by different systems. Comparison also shown for systems employing pre-trained word vectors and a character vector.

Table 6 shows how LAS varies with the different POS taggers when averaged across the nine-language sample. We see a clear improvement

---

[6] Note that for character embeddings of dimension 24, we use an output size for the character BiLSTM of 50, for character embeddings of dimension 100, we use an output size of 75, and for character embeddings of dimension 500, we use an output size of 100. We checked in separate experiments that the improvements are not simply due to the increase in output size.

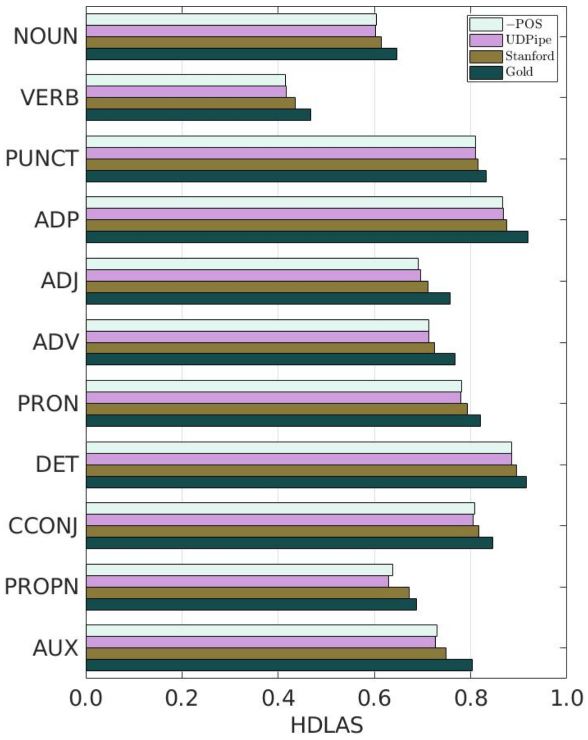

Figure 5: Comparison by POS tag of POS taggers.

from UDPipe to Stanford and then from Stanford to gold tags over the baseline system. This partially confirms results from Dozat et al. (2017), where the Stanford tagger was found to improve parsing results significantly over the UDPipe baseline. More surprising perhaps is the result when comparing to the −POS system, which also makes use of pre-trained word embeddings and a character model. Here, results do not improve at all by adding predicted tags from UDPipe. Stanford tags do give an improvement of 0.7 LAS over −POS, but this is a long way from the improvement of 8.2 LAS we see when adding them on top of BASELINE. Gold tags do however still give a big improvement over −POS (3.5 LAS), suggesting strongly that both UDPipe and Stanford struggle with the decisions that would be most beneficial to parsing accuracy.

In Fig. 5 we present the parsing results broken down by POS tag for the various POS taggers. It is particularly notable that results when tagging with UDPipe are no better than for −POS, which does not use POS tags at all, across most categories, and particularly for the closed-classes ADP, PRON, DET, CCONJ and AUX. Stanford tags do marginally better, but access to gold tags is particularly important in these cases; we see a particularly striking improvement when ADPs and AUXs are correctly tagged over an already strong baseline.

8 Parser speed

It should be noted that increasing the character embedding size and character BiLSTM output dimension as in Section 6 slows down the parser during training and at test time. We found no noticeable difference in speed between the baseline system and versions of the parser with smaller character embedding sizes (24/100), with approximately 20 sentences per second being processed on average during training and 65 sentences per second parsed at test time on the Taito super cluster.[7] There was however a discernible difference when the character embedding size was increased to 500, with only 12 sentences processed per second during training and 44 during testing.

Adding a POS tag embedding makes no appreciable difference to parser speed,[8] but necessitates a pipeline system that first predicts POS tags (assuming gold tags are unavailable). The application of pre-trained embeddings, meanwhile, requires expensive pre-training on large unlabelled corpora. Loading these embeddings into the parser takes time and can occupy large amounts of memory, but does not directly impact the time it takes to process a sentence during training or parsing.

9 Conclusions and Future Work

In this article we examined the complex interactions between pre-trained word vectors, character models and POS tags in neural transition-based dependency parsing. While previous work had shown that POS tags are not as important in the presence of character models, we extend that conclusion to say that in the presence of two of the three techniques, the third is never as important. The best system, however, is always a combination of all three techniques.

We introduced the HDLAS metric to capture the overall effect on parsing quality of changes to the representation of a particular word. We found that all three techniques produce substantial improvements across a range of frequency classes, POS tags, and languages, but the biggest improvements for all techniques were for low-frequency, open-class words. We suggest that this goes some way

---

[7] `https://research.csc.fi/taito-supercluster`

[8] Note that the POS tag embedding we use is small relative to the other components of the word type representation (see Table 2).

to explaining the redundancy between the three techniques: they target the same weaknesses in the baseline word-type level embedding.

We confirmed a previous result that the character model is particularly important for morphologically rich languages with high type-token ratios, and went on to show that these languages also benefit from larger character embedding sizes, whereas morphologically simpler languages make do with small character embeddings, even if the character set size is large.

POS tag embeddings can improve results for difficult closed-class categories, but our current best POS taggers are not capable of making the distinctions necessary to really take advantage of this. The strength of pre-trained embeddings is that they are trained on much larger corpora than the task-specific data; the use of character models and POS tag embeddings however seems to allow us to generalise much better from smaller data sets, as each character and each POS tag is normally seen many times, even if each word type is rare.

We saw that increasing the character embedding size slows the parser down; whether this tradeoff is worthwhile will depend on the application in question. If accuracy is all that matters, we recommend using a fully combined system with large character embeddings in tandem with POS tags and pre-trained embeddings. Where speed is more important, it may be worth considering a system that employs a smaller character embedding and does without POS tags, using just pre-trained embeddings.

In future work it would be interesting to investigate whether the patterns observed here also hold true for other types of models in dependency parsing; possible variations to examine include alternative character models such as convolutional neural networks, joint tagging-parsing models, and graph-based parsers.

Acknowledgments

We acknowledge the computational resources provided by CSC in Helsinki and Sigma2 in Oslo through NeIC-NLPL (www.nlpl.eu). Aaron Smith was supported by the Swedish Research Council. We would like to thank Sujoung Baeck for a valuable discussion regarding Korean morphology.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359.

Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750.

Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W Cohen. 2017a. A Comparative Study of Word Embeddings for Reading Comprehension. *arXiv preprint arXiv:1703.00993.*

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017b. Gated-Attention Readers for Text Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's Graph-based Neural Dependency Parser at the CoNLL 2017 Shared Task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343.

Alex Graves. 2008. *Supervised Sequence Labelling with Recurrent Neural Networks.* Ph.D. thesis, Technical University Munich.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware Neural Language Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association of Computational Linguistics*, 4:313–327.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and

Joakim Nivre. 2017a. From Raw Text to Universal Dependencies - Look, No Tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 207–217.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017b. Arc-Hybrid Non-Projective Dependency Parsing with a Static-Dynamic Oracle. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 99–104.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017c. Old School vs. New School: Comparing Transition-Based Parsers with and without Neural Network Enhancement. In *Proceedings of the 15th Treebanks and Linguistic Theories Workshop*, pages 99–110.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Joakim Nivre. 2008. Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics*, 34(4):513–553.

Joakim Nivre. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Yan Shao. 2018. *Segmenting and Tagging Text with Neural Networks*. Ph.D. thesis, Uppsala University.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 Treebanks, 34 Models: Universal Dependency Parsing with Multi-Treebank Models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Milan Straka, Jan Hajic, and Jana Straková. 2016. UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Daniel Zeman, Martin Popel, Milan Straka, et al. 2017. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19.

Depth-bounding is effective: Improvements and evaluation of unsupervised PCFG induction

Lifeng Jin
Department of Linguistics
The Ohio State University
`jin.544@osu.edu`

Finale Doshi-Velez
Harvard University
`finale@seas.harvard.edu`

Timothy Miller
Boston Children's Hospital &
Harvard Medical School
`timothy.miller@childrens.harvard.edu`

William Schuler
Department of Linguistics
The Ohio State University
`schuler@ling.osu.edu`

Lane Schwartz
Department of Linguistics
University of Illinois at Urbana-Champaign
`lanes@illinois.edu`

Abstract

There have been several recent attempts to improve the accuracy of grammar induction systems by bounding the recursive complexity of the induction model (Ponvert et al., 2011; Noji and Johnson, 2016; Shain et al., 2016; Jin et al., 2018). Modern depth-bounded grammar inducers have been shown to be more accurate than early unbounded PCFG inducers, but this technique has never been compared against unbounded induction within the same system, in part because most previous depth-bounding models are built around sequence models, the complexity of which grows exponentially with the maximum allowed depth. The present work instead applies depth bounds within a chart-based Bayesian PCFG inducer (Johnson et al., 2007b), where bounding can be switched on and off, and then samples trees with and without bounding.[1] Results show that depth-bounding is indeed significantly effective in limiting the search space of the inducer and thereby increasing the accuracy of the resulting parsing model. Moreover, parsing results on English, Chinese and German show that this bounded model with a new inference technique is able to produce parse trees more accurately than or competitively with state-of-the-art constituency-based grammar induction models.

1 Introduction

Unsupervised grammar inducers hypothesize hierarchical structures for strings of words. Using context-free grammars (CFGs) to define these structures, previous attempts at either CFG parameter estimation (Carroll and Charniak, 1992; Schabes and Pereira, 1992; Johnson et al., 2007b) or directly inducing a CFG as well as its probabilities (Liang et al., 2009; Tu, 2012) have not achieved as much success as experiments with other kinds of formalisms (Klein and Manning, 2004; Seginer, 2007; Ponvert et al., 2011). The assumption has been made that the space of grammars is so big that constraints must be applied to the learning process to reduce the burden of the learner (Gold, 1967; Cramer, 2007; Liang et al., 2009).

One constraint that has been applied is recursion depth (Schuler et al., 2010; Ponvert et al., 2011; Shain et al., 2016; Noji and Johnson, 2016; Jin et al., 2018), motivated by human cognitive constraints on memory capacity (Chomsky and Miller, 1963). Recursion depth can be defined in a left-corner parsing paradigm (Rosenkrantz and Lewis, 1970; Johnson-Laird, 1983). Left-corner parsers require only minimal stack memory to process left-branching and right-branching structures, but require an extra stack element to process each center embedding in a structure. For example, a left-corner parser must add a stack element for each of the first three words in the sentence, *For parts the plant built to fail was awful,* shown in Figure 1. These kinds of depth bounds in sentence processing have been used to explain the relative difficulty of center-embedded sentences compared to more right-branching paraphrases like *It was awful for the plant's parts to fail.*

However, depth-bounded grammar induction has never been compared against unbounded induction in the same system, in part because most previous depth-bounding models are built around sequence models, the complexity of which grows exponentially with the maximum allowed depth. In order to compare the effects of depth-bounding more directly, this work extends a chart-based Bayesian PCFG induction model (Johnson et al., 2007b) to include depth bounding, which allows both bounded and unbounded PCFGs to be induced from unannotated text.

Experiments reported in this paper confirm that

---

[1] The public repository can be found at `https://github.com/lifengjin/dimi_emnlp18`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2721–2731
Brussels, Belgium, October 31 - November 4, 2018.

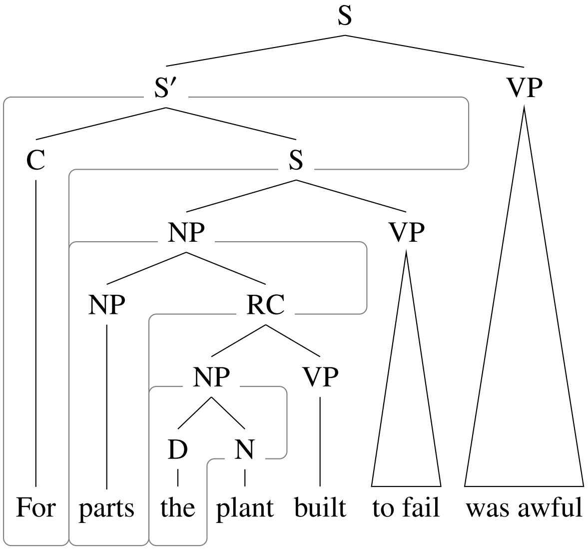

Figure 1: Stack elements after the word *the* in a left-corner parse of the sentence *For parts the plant built to fail was awful.*

depth-bounding does empirically have the effect of significantly limiting the search space of the inducer. Analyses of this model also show that the posterior samples are indicative of implicit depth limits in the data. This work also shows for the first time that it is possible to induce an accurate unbounded PCFG from raw text with no strong linguistic constraints. With a novel grammar-level marginalization in posterior inference, comparisons of the accuracy of bounded grammar induction using this model against other recent constituency grammar inducers show that this model is able to achieve state-of-the-art or competitive results on datasets in multiple languages.

2 Related work

Induction of PCFGs has long been considered a difficult problem (Carroll and Charniak, 1992; Johnson et al., 2007b; Liang et al., 2009; Tu, 2012). Lack of success for direct estimation was attributed either to a lack of correlation between the linguistic accuracy and the optimization objective (Johnson et al., 2007b), or the likelihood function or the posterior being filled with weak local optima (Smith, 2006; Liang et al., 2009). Much of this grammar induction work used strong linguistically motivated constraints or direct linguistic annotation to help the inducer eliminate some local optima. Schabes and Pereira (1992) use bracketed corpora to provide extra structural information to the inducer. Use of part-of-speech (POS) sequences in place of word strings is popular in the dependency grammar induction literature (Klein and Manning, 2002, 2004; Berg-Kirkpatrick et al., 2010; Jiang et al., 2016; Noji and Johnson, 2016). Combinatory Categorial Grammar (CCG) induction also relies on POS tags to assign basic categories to words (Bisk and Hockenmaier, 2012, 2013), among other constraints such as CCG combinators. Other linguistic constraints such as constraints of root nodes (Noji and Johnson, 2016), attachment rules (Naseem et al., 2010) or acoustic cues (Pate, 2013) have also been used in induction.

Depth-like constraints have been applied in work by Seginer (2007) and Ponvert et al. (2011) to help with the search. Both of these systems are successful in inducing phrase structure trees from only words, but only generate unlabeled constituents.

Depth-bounds are directly used by induction models in work by Noji and Johnson (2016), Shain et al. (2016) and Jin et al. (2018), and are shown to be beneficial to induction. Noji and Johnson (2016) apply depth-bounding to dependency grammar induction with POS tags. However the constituency parsing evaluation scores they report are low compared to other induction systems. The model in Shain et al. (2016) is a hierarchical sequence model instead of a PCFG. Although depth-bounding limits the search space, the sequence model has more parameters than a PCFG, therefore benefits brought by depth-bounding may be offset by this larger parameter space.

Jin et al. (2018) also apply depth-bounding to a grammar inducer and induce depth-bounded PCFGs and show that the depth-bounded grammar inducer can learn labeled PCFGs competitive with state-of-the-art grammar inducers that only produce unlabeled trees. However, because of the cognitively motivated left-corner HMM sampler used in the model, its state space grows exponentially with the maximum depth and polynomially with the number of categories. This renders the transition matrix and the trellis of the inducer too big to be practical in exploring models with higher depth limits, let alone unbounded models. By using Gibbs sampling for PCFGs (Goodman, 1998; Johnson et al., 2007b), here described as the inside-sampling algorithm, the state space of the model proposed in this work grows only polynomially with both the maximum depth and the number of categories. This allows experiments with more complex models and also achieves a faster processing speed due to an overall smaller state space.

3 Proposed model

The model described in this paper follows Jin et al. (2018) to induce a depth-bounded PCFG by first inducing an unbounded PCFG and then deterministically deriving the parameters of a depth-bounded PCFG from it. The main difference between this model and the model in Jin et al. (2018) is that they use the bounded PCFG to derive parameters for a factored HMM sequence model, where a forward-filtering backward-sampling algorithm (Carter and Kohn, 1996) can be used in inference. In contrast, the model described in this paper transforms the unbounded PCFG into a bounded PCFG, and then uses the inside-sampling algorithm (Goodman, 1998) to sample from the posterior of the parse trees given the bounded PCFG in inference. This section first gives an overview of the model, then briefly reviews the depth-bounding algorithm for PCFGs (van Schijndel et al., 2013; Jin et al., 2018), and finally describes the inference.

As defined in Jin et al. (2018), a Chomsky normal form (CNF) unbounded PCFG is a matrix $\mathbf{G}$ of binary rule probabilities with one row for each of C parent symbols c and one column for each of $C^2{+}W$ combinations of left and right child symbols a and b, which can be pairs of nonterminals or observed words from vocabulary W followed by null symbols $\perp$:

$$\mathbf{G} = \sum_{a,b,c} \mathsf{P}(c \rightarrow a\ b \mid c)\ \delta_c\ (\delta_a \otimes \delta_b)^\top \tag{1}$$

where δ_c is a Kronecker delta (a vector with value one at index c and zeros elsewhere) and $\otimes$ is a Kronecker product (multiplying two matrices[2] of dimension $m \times n$ and $o \times p$ into a matrix of dimension $mo \times np$ composed of products of all pairs of elements in the operands). A deterministic depth-bounding transform ϕ is then applied to $\mathbf{G}$ to create a depth-bounded version $\mathbf{G}_D$. A depth-bounded grammar is composed of a set of side- and depth-specific distributions $\mathbf{G}_{s,d}$:

$$\mathbf{G}_D = \sum_{s \in \{1,2\}} \sum_{d \in \{1..D\}} \mathbf{D}_{s,d}\, \mathbf{G}_{s,d}\, \mathbf{E}_{s,d}{}^\top \tag{2}$$

where side $s \in \{1,2\}$ indicates left (1) or right (2) child. Categories in $\mathbf{G}_D$ are made to be side- and depth-specific using transforms $\mathbf{D}_{s,d}$ and $\mathbf{E}_{s,d}$:[3]

$$\mathbf{D}_{s,d} = \delta_s \otimes \delta_d \otimes \mathbf{I} \tag{3a}$$

$$\mathbf{E}_{1,d} = \delta_1 \otimes \delta_d \otimes \mathbf{I} \otimes \delta_2 \otimes \delta_d \otimes \mathbf{I} \tag{3b}$$

$$\mathbf{E}_{2,d} = \delta_1 \otimes \delta_{d+1} \otimes \mathbf{I} \otimes \delta_2 \otimes \delta_d \otimes \mathbf{I} \tag{3c}$$

The generative story of this model is as follows. The model first generates an unbounded grammar $\mathbf{G}$ from the Dirichlet prior. Distributions over expansions $\mathsf{P}(c \rightarrow a\ b \mid c)$ of each category c in this model are drawn from a Dirichlet with symmetric parameter β:

$$\mathbf{G} \sim \text{Dirichlet}(\beta) \tag{4}$$

Trees for sentences $1..N$ are each drawn from a PCFG given parameters $\mathbf{G}_D = \phi(\mathbf{G})$:

$$\tau_{1..N} \sim \text{PCFG}(\mathbf{G}_D) \tag{5}$$

Each tree τ is a set $\{\tau_\epsilon, \tau_1, \tau_2, \tau_{11}, \tau_{12}, \tau_{21}, ...\}$ of category labels τ_η where $\eta \in \{1,2\}^*$ is a Gorn address specifying a path of left or right branches from the root. Categories of every pair of left and right children $\tau_{\eta 1}, \tau_{\eta 2}$ are drawn from a multinomial defined by the grammar $\mathbf{G}_D$ and the category of the parent τ_η:

$$\tau_{\eta 1}, \tau_{\eta 2} \sim \text{Multinomial}(\delta_{\tau_\eta}{}^\top \mathbf{G}_D) \tag{6}$$

where $\mathsf{P}_{\mathbf{G}_D}(a\ b \mid w) = \mathsf{P}_{\mathbf{G}_D}(a\ b \mid \perp) = [\![a, b{=}\perp, \perp]\!]$ for $w \in W$, and $[\![\cdot]\!]$ is an indicator function.

In inference, a Gibbs sampler can be used to iteratively draw samples from the conditional posteriors of the unbounded grammar and the parse trees. For example, at iteration t:

$$\mathbf{G}^t \sim \mathsf{P}(\mathbf{G}^t \mid \tau^{t-1}_{1..N}, \sigma_{\tau^{t-1}_{1..N}}, \beta) \tag{7}$$

$$\tau^t_{1...N} \sim \mathsf{P}(\tau^t_{1..N} \mid \mathbf{G}^t_D, \sigma_{\tau^t_{1..N}}) \tag{8}$$

where σ_τ denotes the terminals in τ. These distributions will be defined in Section 3.2.

3.1 Depth-bounding a PCFG

This section summarizes the depth-bounding function ϕ for PCFGs described in van Schijndel et al. (2013) and Jin et al. (2018). Depth-bounding essentially creates a set of PCFGs with depth- and side-specific categories where no tree that exceeds

---

[2] or vectors in case n and p equal one

[3] Note that this correctly stipulates depth increases for left children of right children.

its depth bound can be generated by the bounded grammar. Because depth increases when a left child of a right child of some parent category performs non-terminal expansion, the probability of such expansions at the maximum depth limit as well as non-depth-increasing expansions beyond the maximum depth limit must be removed from the unbounded grammar. Following van Schijndel et al. (2013) and Jin et al. (2018), this can be done by iteratively defining a side- and depth-specific containment likelihood $\mathbf{h}^{(i)}_{s,d}$ for left- or right-side siblings $s \in \{1, 2\}$ at depth $d \in \{1..D\}$ at each iteration $i \in \{1..I\}$, as a vector with one row for each nonterminal or terminal symbol (or null symbol $\perp$) in $\mathbf{G}$, containing the probability of each symbol generating a complete yield within depth d as an s-side sibling:

$$\mathbf{h}^{(0)}_{s,d} = \mathbf{0} \tag{9a}$$

$$\mathbf{h}^{(i)}_{1,d} = \begin{cases} \mathbf{G}\,(\mathbf{1} \otimes \delta_\perp \ + \ \mathbf{h}^{(i-1)}_{1,d} \otimes \mathbf{h}^{(i-1)}_{2,d}) & \text{if } d \leq D+1 \\ \mathbf{0} & \text{if } d > D+1 \end{cases} \tag{9b}$$

$$\mathbf{h}^{(i)}_{2,d} = \begin{cases} \delta_\mathrm{T} & \text{if } d = 0 \\ \mathbf{G}\,(\mathbf{1} \otimes \delta_\perp \ + \ \mathbf{h}^{(i-1)}_{1,d+1} \otimes \mathbf{h}^{(i-1)}_{2,d}) & \text{if } 0 < d \leq D \\ \mathbf{0} & \text{if } d > D \end{cases} \tag{9c}$$

where ‘T’ is a top-level category label at depth zero. Following previous work, experiments described in this paper use $I = 20$.

A depth-bounded grammar $\mathbf{G}_{s,d}$ can then be defined to be the original grammar $\mathbf{G}$ reweighted and renormalized by this containment likelihood:

$$\mathbf{G}_{1,d} = \frac{\mathbf{G}\,\mathrm{diag}(\mathbf{1} \otimes \delta_\perp \ + \ \mathbf{h}^{(I)}_{1,d} \otimes \mathbf{h}^{(I)}_{2,d})}{\mathbf{h}^{(I)}_{1,d}} \tag{10a}$$

$$\mathbf{G}_{2,d} = \frac{\mathbf{G}\,\mathrm{diag}(\mathbf{1} \otimes \delta_\perp \ + \ \mathbf{h}^{(I)}_{1,d+1} \otimes \mathbf{h}^{(I)}_{2,d})}{\mathbf{h}^{(I)}_{2,d}} \tag{10b}$$

3.2 Gibbs sampling of unbounded grammars and bounded trees

As defined above, this model samples iteratively from the conditional posteriors of $\mathsf{P}(\mathbf{G} \mid \tau_{0..N}, \sigma_{\tau_{0..N}}, \beta)$ and $\mathsf{P}(\tau_{0..N} \mid \mathbf{G}_D, \sigma_{\tau_{0..N}})$ in inference, extending the Gibbs sampling algorithm for PCFG induction introduced in Johnson et al. (2007b) to depth-bounded grammars. The below equations will omit the superscript t for the iteration number of inference for clarity.

To sample from the conditional posterior of $\mathbf{G}$, it is necessary to first sum over all rule applications in all sampled trees:

$$\mathbf{C}_D = \sum_{\tau \in \tau_{1..N}} \sum_{\tau_\eta \in \tau} \delta_{\tau_\eta}\,(\delta_{\tau_{\eta 1}} \otimes \delta_{\tau_{\eta 2}})^\top \tag{11}$$

then remove side- and depth-specificity from category labels:

$$\mathbf{C} = \sum_s \sum_d \mathbf{D}_{s,d}{}^\top \mathbf{C}_D\, \mathbf{E}_{s,d} \tag{12}$$

A side- and depth-independent grammar is then sampled from these counts, plus the pseudocount β:

$$\mathbf{G} \sim \mathrm{Dirichlet}(\beta + \mathbf{C}) \tag{13}$$

Inside-sampling (Goodman, 1998; Johnson et al., 2007b) is then used to sample from the posterior of trees $\mathsf{P}(\tau_{0..N} \mid \mathbf{G}_D, \sigma_{\tau_{0...N}})$. Given a depth-bounded grammar and a sentence, this algorithm first constructs the inside chart $\mathbf{V} \in \mathbb{R}^{L \times L \times C}$, where L is the length of the sentence. A chart vector $\mathbf{V}_{[i,j,1..C]}$ for the span i, j where $i < j \leq L$ in some sentence $w_{1..L}$ is the likelihood $\mathsf{P}_{\mathbf{G}_D}(w_{i..j} \mid c)$ of the span for all side- and depth-specific categories c:

$$\mathbf{V}_{[i,j,1..C]} = \begin{cases} \mathbf{G}_D\,(\delta_{w_i} \otimes \delta_\perp) & \text{if } j{-}i = 1 \\ \sum_k \mathbf{G}_D\,(\mathbf{V}_{[i,k,1..C]} \otimes \mathbf{V}_{[k,j,1..C]}) & \text{if } j{-}i > 1 \end{cases} \tag{14}$$

Trees are sampled iteratively from the top down by first choosing a split point $k_{i,j}$ for the current span i, j such that $i < k_{i,j} < j$:

$$k_{i,j} \sim \mathrm{Mul}\left(\sum_k \delta_k\, \delta^\top_{c_{i,j}} \mathbf{G}_D\,(\mathbf{V}_{[i,k,1..C]} \otimes \mathbf{V}_{[k,j,1..C]}) \right) \tag{15}$$

The algorithm then samples pairs of category labels $c_{i,k_{i,j}}$ and $c_{k_{i,j},j}$ adjacent at this split point $k_{i,j}$:

$$c_{i,k}, c_{k,j} \sim \mathrm{Mul}\left(\delta^\top_{c_{i,j}} \mathbf{G}_D\, \mathrm{diag}(\mathbf{V}_{[i,k,1..C]} \otimes \mathbf{V}_{[k,j,1..C]}) \right) \tag{16}$$

Empirically the sampler spends most of its time constructing the inside chart. The model described in this paper therefore efficiently computes the inside chart using matrix multiplication, which is able to exploit GPU optimization.

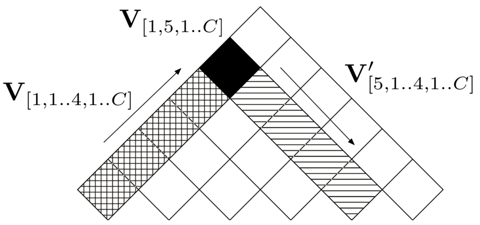

Figure 2: Example of matrix multiplication in place of looping over break points for the span (0,5). Each chart cell represents a likelihood vector for the span between i and j where i is the leftmost delimiting index of the span and j the rightmost. The arrows represent the order in which the cells are stored in the chart matrices $\mathbf{V}$ and $\mathbf{V}'$.

Efficient inside score calculation

The complexity of the inside algorithm is cubic on the length of the sentence because it has to iterate over all start points i, all end points j and all split points k of a span. For a dense PCFG with a large number of states, the explicit looping is undesirable, especially when it can be formulated as matrix multiplication. The split point loop is therefore replaced with a matrix multiplication in order to take advantage of highly optimized GPU linear algebra packages like cuBLAS and cuSPARSE, whereas previous work explores how to parse efficiently on GPUs (Johnson, 2011; Canny et al., 2013; Hall et al., 2014).

Inside likelihoods are propagated using a copy $\mathbf{V}'$ of the inside likelihood tensor $\mathbf{V}$ with the first and second indices reversed:

$$\mathbf{V}'_{[j,i,c]} = \mathbf{V}_{[i,j,c]} \tag{17}$$

This reversal allows the sum over split points $k \in \{i{+}1, ..., j{-}1\}$ to be calculated as a product of contiguous matrices, which can be efficiently implemented on a GPU:

$$\mathbf{V}_{[i,j,1..C]} = \mathbf{G}_D \, \mathrm{vec}(\mathbf{V}_{[i,i+1..j-1,1..C]}{}^{\top}\mathbf{V}'_{[j,i+1..j-1,1..C]}) \tag{18}$$

where vec($\mathbf{M}$) flattens a matrix $\mathbf{M}$ into a vector.

3.3 Posterior inference on constituents

Prior work (Johnson et al., 2007a) shows that using EM-like algorithms, which seek to maximize the likelihood of data marginalizing out the latent trees, does not yield good performance. Because trees are the main target for evaluation, it may be preferable to find the most probable tree structures given the marginal posterior of tree structures compared to finding the most probable grammar. Some recent work (McClosky and Charniak, 2015; Keith et al., 2018) explores how to use marginal distributions of tree structures from supervised parsers to create more accurate parse trees. Based on these arguments, this model performs maximum a posteriori (MAP) inference on constituents (PIoC) using approximate conditional posteriors of spans to create final parses for evaluation.

Formally, let $\sigma^{\star}_{i,j}$ be an MAP unlabeled span of words in a sentence from a corpus σ, with start point i and end point j, and $\sigma_{i,k}, \sigma_{k,j}$ its possible children. This algorithm iteratively looks for the best pair of children $\sigma^{\star}_{i,k}, \sigma^{\star}_{k,j}$ according to the posterior of the children, using all posterior samples. The spans are sentence-specific, but the below equations omit the sentence index for brevity:

$$\begin{aligned} \sigma^{\star}_{i,k}, \sigma^{\star}_{k,j} &= \operatorname*{arg\,max}_{\sigma_{i,k},\sigma_{k,j}} \mathsf{P}(\sigma_{i,k}, \sigma_{k,j} \mid \sigma^{\star}_{i,j}, \sigma) \\ &= \operatorname*{arg\,max}_{\sigma_{i,k},\sigma_{k,j}} \int \mathsf{P}(\sigma_{i,k}, \sigma_{k,j}, \mathbf{G} \mid \sigma^{\star}_{i,j}, \sigma) \, d\mathbf{G} \\ &\approx \operatorname*{arg\,max}_{\sigma_{i,k},\sigma_{k,j}} \sum_{\hat{\mathbf{G}} \sim \mathsf{P}(\mathbf{G}|\sigma)} \mathsf{P}(\sigma_{i,k}, \sigma_{k,j}, \hat{\mathbf{G}} \mid \sigma^{\star}_{i,j}, \sigma) \end{aligned} \tag{19}$$

where σ is the training corpus. Starting from the whole sentence $\sigma_{0,N}$, this algorithm finds the best children for a span from the Monte Carlo estimation of the marginal posterior distribution of children for the span, and then continues to split the found children spans. Because samples from different runs at different iterations can be used to approximate the span posteriors, the process marginalizes out sampled grammars, whole-sentence parse trees and constituent labels to only consider split points for spans. In terms of input and output, the PIoC algorithm takes in posterior samples of trees for a sentence, and outputs an unlabeled binary-branching tree.

There are a few benefits of doing posterior inference on constituents. First, the distribution $\mathsf{P}(\sigma_{i,k}, \sigma_{k,j} \mid \sigma^{\star}_{i,j}, \sigma)$ quantifies how much uncertainty there is in splitting a span $\sigma_{i,j}$ at all possible k's. One way of using this uncertainty information is to merge spans where uncertainty is high, effectively weakening or removing the constraint of binary-branching from the grammar inducer. Second, this algorithm produces trees that may not be seen in the samples, potentially helping aggregate evidence across different iterations within a run and across runs. Third, the multimodal nature of the joint posterior of grammars and trees often makes the sampler get stuck at local modes, but doing MAP on constituents may allow information about trees from different modes to come together. If different grammars all consider certain children for a span to be highly likely, then these children should be in the final parse output. Finally, it is a nonparametric way of doing model selection. As will be shown, model selection relies on the log likelihood of the data, but the log likelihood of the data is only weakly correlated with parsing accuracy. Performing PIoC with multiple runs can increase accuracy without depending too heavily on log likelihood for model selection.

4 Model analysis and evaluation

The model described above has hyperparameters for maximum depth D, number of categories C and the symmetric Dirichlet prior β. Following Jin et al. (2018), this evaluation uses the first half of the WSJ20 corpus as the development set (WSJ20dev) for all experiments. However instead of using the development set only to set the hyperparameters of the model, this evaluation also uses it to explore interactions among parsing accuracy, model fit, depth limit and category domain. The first set of experiments explores various settings of D in the hope of acquiring a better picture of how depth-bounding affects the inducer. The second set of experiments uses the value of D tuned in the first experiments, and does PIoC on different sets of samples to examine the effect it has on parse quality. Optimal parameter values from these first two experiments are then applied in experiments on English (The Penn Treebank; Marcus et al., 1993), Chinese (The Chinese Treebank 5.0; Xia et al., 2000) and German (NEGRA 2.0; Skut et al., 1998) data to show how the model performs compared with competing systems.

Each run in evaluation uses one sample of parse trees from the posterior samples after convergence. Preliminary experiments show that the samples after convergence are very similar within a run and their parsing accuracies differ very little. This evaluation follows Seginer (2007) by running unlabeled PARSEVAL on parse trees collected from each run. Punctuation is retained in the raw text in induction, and removed in evaluation, also following Seginer (2007).

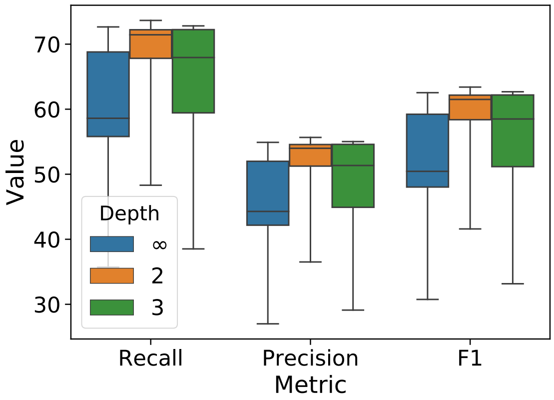

Figure 3: PARSEVAL scores for runs with different depth limits. The difference of all PARSEVAL scores between depth ∞ and depth 2 is significant (p=0.017, Student's t test).

4.1 Analysis of model behavior

The first experiment explores the effects of depth-bounding on linguistic grammar quality. The hypothesis is that depth-bounding limits the search space of possible grammars, so the inducer will be less likely to find low-quality local optima where cognitively implausible parse trees are assigned non-zero probabilities, because such local optima would be removed from the posterior by limiting the maximum depth of parse trees to a small number d.

The effect of depth-bounding

Figure 3 shows the effect of depth bounding using 60 data points of unlabeled PARSEVAL scores from 20 different runs for each of three different depth bounds: 2, 3, and ∞ (unbounded). The range of possible parsing accuracy scores is very wide, as mapped out by the runs. Although the unbounded model is able to reach the performance upper bound seen from the figure, most of the time its results are in the middle of the range. By bounding the maximum depth to 2, the sampler is able to stay in the region of high parsing accuracy. This may be because the majority of the modes in the region of low parsing accuracy require higher depth limits, and humans who produce the sentences do not have access to those higher depth limits. The difference between depth ∞ and depth 2 is significant (p=0.017, Student's t test), showing that depth-bounding does have a positive effect on the linguistic grammar quality of the induced grammars. Data from depth 3 also shows a positive trend of inducing better grammars than

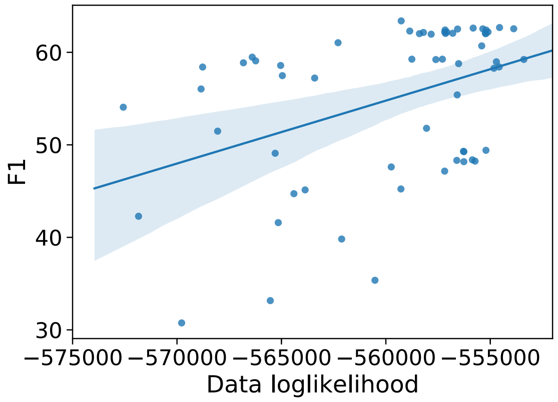

Figure 4: The correlation between data likelihood and parsing accuracy of all 60 runs. Calculations show that there is a significant ($p = 0.007$) positive correlation (Pearson's r=0.39) between data likelihood and parsing accuracy at convergence for our model.

unbounded.

A purely right-branching baseline achieves an F1 score of 48 on the WSJ20 development dataset. A majority of induction runs perform better than this baseline, which indicates that the PCFG induction model with the inside-sampling algorithm is able to find good solutions, most of the time much better than the right-branching baseline. This is especially interesting when the grammar is unbounded with almost no other constraint, which had previously been shown to converge to weak local optima.

Correlation of model fit and parsing accuracy

Model fit, or data likelihood, has been reported not to be correlated or to be correlated only weakly with parsing accuracy for some unsupervised grammar induction models (Smith, 2006; Johnson et al., 2007b; Liang et al., 2009) when the model has converged to a local maximum. Figure 4 shows the correlation between data likelihood and parsing accuracy at convergence for all the runs. There is a significant ($p = 0.007$) positive correlation (Pearson's r=0.39) between data likelihood and parsing accuracy at convergence for our model. This indicates that although noisy and unreliable, the data likelihood can be used as a metric to do preliminary model selection.

The bounded unbounded PCFG

We also examine the distribution of tree depths in unbounded runs. For a run, we compute the percentage of parse trees with a certain depth, and then examine how these percentages vary across

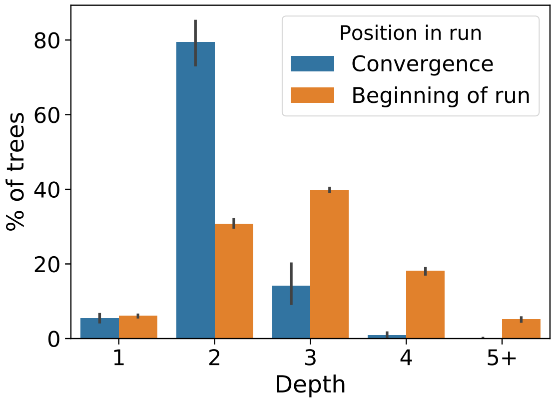

Figure 5: The usage of different depths for parse trees in the samples from 20 runs with the unbounded grammar.

different runs. Theoretically the possible maximum depth of a parse for a sentence is the sentence length divided by 2. For example, a 20-word sentence can have a parse of depth 10 because at least two words are needed to create a new depth with a center embedded phrase, but under most PCFGs this maximally center embedded configuration is not very likely. Figure 5 shows the percentage of tree depths from samples in the beginning of each unbounded run and at convergence. It shows that at the beginning of the sampling process with a random model sampled from the prior, the distribution of parse tree depths seems to be centered around depth 2 and 3, with non-negligible probability mass at other depth levels. At convergence, the distribution of parse tree depths is very peaked with a large portion of the probability mass concentrated at depth 2. Given that an unbounded PCFG has no constraint on depth, this convergence of the marginal posterior distribution of parse tree depth shows that the depth limit seems to be a natural tendency in the data, rather than an arbitrary preference of corpus annotators.

4.2 Posterior uncertainty of constituents

Experiments were also conducted to determine whether posterior inference on constituents (PIoC) has any effect on parsing accuracy. These experiments use 10 runs on WSJ20dev with depth 2 that have the highest log-likelihoods for exploration. In this data, some spans have a strikingly higher degree of uncertainty than other spans. For example, the posterior probability of splitting the phrase *the old story*, into *the old* and *story* is 0.55, and the

System	Rec	Prec	F1
Best	73.65	55.66	63.40
Best w/ PIoC	73.59	56.41	63.87
All w/ PIoC	72.99	59.21	65.38
All w/ PIoC w/o best	73.00	59.06	65.29

Table 1: Development results for different systems using posterior inference on constituents (PIoC).

probability of splitting it into *the* and *old story* is 0.45. Some other spans like *use old tools* have virtually no uncertainty in how the inducer evaluates the splits. Many such spans with high uncertainty are noun phrases, which are not annotated with subconstituents in the Penn Treebank annotation. The parser can therefore avoid precision losses by not splitting constituents with 3 or 4 words if there is large uncertainty in this posterior.[4] This experiment only merges spans that would cover 3 or 4 words and leave merging spans with larger coverage to future work.

Table 1 shows parsing results on the WSJ20dev dataset. The *Best* result is from an arbitrary sample at convergence of the oracle best run. The *Best with PIoC* is the same run, but with PIoC to aggregate 100 posterior samples at convergence. *All with PIoC* uses 100 posterior samples from all of the 10 chosen runs, and finally *All with PIoC without best* excludes the best run in PIoC calculation.

There is almost a point of gain in precision going from *Best* to *Best with PIoC* with virtually no recall loss, showing that the posterior uncertainty is helpful in flattening binary trees. As more samples from the posterior are collected, as shown in *All with PIoC without best*, the precision gain is even more substantial. This shows that with PIoC there is no need to know which sample from which run is the best. Model selection in this case is only needed to weed out the runs with very low likelihood.

4.3 Multilingual PARSEVAL

A final set of experiments compare the proposed model with several state-of-the-art constituency grammar induction systems on three different languages. The competing systems are CCL (Seginer, 2007)[5] and UPPARSE (Ponvert et al., 2011).[6] We also include the published results of DB-PCFG (Jin et al., 2018) on English for comparison.[7] The corpora used are the WSJ20test dataset used in Jin et al. (2018), the CTB20 (sentences with 20 words or fewer from the Chinese Treebank) and NEGRA20 (sentences with 20 words or fewer from the German NEGRA Treebank) datasets used in Seginer (2007). All systems are trained and evaluated on the same datasets to ensure fair and direct comparison. Five different induction runs were run on each dataset with the same hyperparameters D=2, C=15, β=0.2 as tuned on the development set, and three runs with the highest likelihood at convergence were chosen for comparison with other models. Parse trees were then calculated using PIoC as previously described, removing punctuation to calculate the unlabeled PARSEVAL scores with EVALB. Multiple runs of CCL and UPPARSE on the same data yield the same results.

Table 2 shows the unlabeled PARSEVAL scores for the competing systems. The model described in this paper shows strong performance in all languages. On English and Chinese, this model achieves the new state-of-the-art recall and F1 numbers. On German, this model also achieves the best recall scores among all models, showing that more constituents found in the gold annotation are discovered. It is worth noting that the CCL and UPPARSE models do take advantage of additional linguistic constraints, e.g. using punctuation as delimiters of constituents. Experiments described in this paper show that this system can perform better than or competitive with these existing models without similar heuristics and constraints.

The model described in this paper performs relatively poorly on precision due to the fact that trees produced by this system are mostly binary-branching with some constituents flattened by PIoC. This issue is most evident on Negra, where fully binary-branching trees have nearly twice as many constituents as are annotated in gold. This puts any system that produces binary-branching trees under a precision celling of 0.51, and F1 celling of 0.675.

5 Conclusion

Experiments in this work confirm that depth-bounding does empirically have the effect of limiting the search space of an unsupervised PCFG in-

---

[4] I.e. if the difference between the first and the second highest posterior probabilities is smaller than 0.3.

[5] `https://github.com/DrDub/cclparser`

[6] `https://github.com/eponvert/upparse`

[7] We are not able to run DB-PCFG on the other languages due to its substantial resource requirements.

System	WSJ20test			CTB20			NEGRA20		
	Rec	Prec	F1	Rec	Prec	F1	Rec	Prec	F1
CCL	61.7	**60.1**	60.9	35.3	39.2	37.1	44.4	27.2	33.7
UPPARSE	40.5	47.8	43.9	33.8	**44.0**	38.2	55.5	**41.9**	**47.7**
DB-PCFG	70.5	53.0	60.5	-	-	-	-	-	-
this work	**73.1**	55.6	**63.1**	**43.8**	35.1	**38.9**	**59.1**	31.2	40.8

Table 2: PARSEVAL scores for different constituency grammar induction systems.

ducer. Analysis of a depth-bounded model demonstrates desirable engineering properties, including a significant correlation between parsing accuracy and data likelihood, and interesting linguistic properties such as implicit bounding for unbounded grammars. This paper also introduces the Posterior Inference on Constituents technique for model selection and shows for the first time that it is possible to accurately induce a PCFG with no strong universal linguistic constraints. Comparisons of the proposed model with other state-of-the-art constituency grammar inducers show that this model is able to achieve state-of-the-art or competitive results on datasets in multiple languages.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. Computations for this project were partly run on the Ohio Supercomputer Center (1987). This research was funded by the Defense Advanced Research Projects Agency award HR0011-15-2-0022. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 582–590.

Yonatan Bisk and Julia Hockenmaier. 2012. Simple Robust Grammar Induction with Combinatory Categorial Grammars. *AAAI*, pages 1643–1649.

Yonatan Bisk and Julia Hockenmaier. 2013. An HDP Model for Inducing Combinatory Categorial Grammars. In *Transactions Of The Association For Computational Linguistics*, pages 75–88.

John Canny, David Hall, and Dan Klein. 2013. A multi-Teraflop Constituency Parser using GPUs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1898–1907.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. *Working Notes of the Workshop on Statistically-Based NLP Techniques*, (March):1–13.

C. K. Carter and R. Kohn. 1996. Markov chain Monte Carlo in conditionally Gaussian state space models. *Biometrika*, 83(3):589–601.

Noam Chomsky and George A Miller. 1963. Introduction to the formal analysis of natural languages. In *Handbook of Mathematical Psychology*, pages 269–321. Wiley, New York, NY.

Bart Cramer. 2007. Limitations of current grammar induction algorithms. *Proceedings of the 45th Annual Meeting of the ACL: Student Research Workshop on - ACL '07*, (June):43.

Mark E. Gold. 1967. Language Identification in the Limit. *Information and Control*, (10):447–474.

Joshua Goodman. 1998. Parsing Inside-Out.

David Hall, Taylor Berg-Kirkpatrick, Canny John, and Dan Klein. 2014. Sparser, Better, Faster GPU Parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2007, page 208217.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 61503248, pages 763–771.

Lifeng Jin, William Schuler, Finale Doshi-Velez, Timothy A Miller, and Lane Schwartz. 2018. Unsupervised Grammar Induction with Depth-bounded PCFG. *Transactions of the Association for Computational Linguistics*.

Mark Johnson. 2011. Parsing in parallel on multiple cores and gpus. In *In Proceedings of Australasian Language Technology Association Workshop*, pages 29–37.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007a. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems*, volume 19, page 641.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007b. Bayesian Inference for PCFGs via Markov chain Monte Carlo. *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference (ACL)*, pages 139–146.

Philip N Johnson-Laird. 1983. *Mental models: Towards a cognitive science of language, inference, and consciousness*. Harvard University Press, Cambridge, MA, USA.

Katherine A Keith, Su Lin Blodgett, and Brendan O'Connor. 2018. Monte Carlo Syntax Marginals for Exploring and Using Dependency Parses. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 128–135.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*, volume 1, pages 478–485.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Probabilistic Grammars and Hierarchical Dirichlet Processes. *The Handbook of Applied Bayesian Analysis*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky and Eugene Charniak. 2015. Syntactic Parse Fusion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, September, pages 1360–1366.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using Universal Linguistic Knowledge to Guide Grammar Induction. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, (October):1234–1244.

Hiroshi Noji and Mark Johnson. 2016. Using Left-corner Parsing to Encode Universal Structural Constraints in Grammar Induction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 33–43.

The Ohio Supercomputer Center. 1987. Ohio Supercomputer Center. \url{http://osc.edu/ark:/19495/f5s1ph73}.

John K Pate. 2013. Unsupervised Dependency Parsing with Acoustic Cues. *Transactions of the Association for Computational Linguistics*, 1:63–74.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086.

D J Rosenkrantz and P M Lewis. 1970. Deterministic left corner parsing. In *11th Annual Symposium on Switching and Automata Theory (swat 1970)*, pages 139–152.

Fernando Schabes and Yves Pereira. 1992. Inside-Outside Reestimation From Partially Bracketed Corpora. *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 128–135.

Marten van Schijndel, Andy Exley, and William Schuler. 2013. A Model of language processing as hierarchic sequential prediction. *Topics in Cognitive Science*, 5(3):522–540.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage parsing using human-Like memory constraints. *Computational Linguistics*, 36(1):1–30.

Yoav Seginer. 2007. Fast Unsupervised Incremental Parsing. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics*, pages 384–391.

Cory Shain, William Bryce, Lifeng Jin, Victoria Krakovna, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2016. Memory-bounded left-corner unsupervised grammar induction on child-directed input. In *Proceedings of the International Conference on Computational Linguistics*, pages 964–975.

Wojciech Skut, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. 1998. A Linguistically Interpreted Corpus of German Newspaper Text. In *Proceedings of the ESSLLI Workshop on Recent Advances in Corpus Annotation.*, page 7.

Noah Ashton Smith. 2006. Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text. *PhD Thesis*, pages 1–228.

Kewei Tu. 2012. *Unsupervised learning of probabilistic grammars*. Ph.D. thesis.

Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Ocurowski, John Kovarik, Fu-Dong Chiou, Shizhe Huang, Tony Kroch, and Mitch Marcus. 2000. Developing Guidelines and Ensuring Consistency for

Chinese Text Annotation. *Proceedings of the Second Language Resources and Evaluation Conference*, (Section 3).

Incremental Computation of Infix Probabilities for Probabilistic Finite Automata

Marco Cognetta, Yo-Sub Han, and **Soon Chan Kwon**
Department of Computer Science
Yonsei University, Seoul, Republic of Korea
`{mcognetta, emmous, soon--chan}@yonsei.ac.kr`

Abstract

In natural language processing, a common task is to compute the probability of a given phrase appearing or to calculate the probability of all phrases matching a given pattern. For instance, one computes affix (prefix, suffix, infix, etc.) probabilities of a string or a set of strings with respect to a probability distribution of patterns.

The problem of computing infix probabilities of strings when the pattern distribution is given by a probabilistic context-free grammar or by a probabilistic finite automaton is already solved, yet it was open to compute the infix probabilities in an incremental manner. The incremental computation is crucial when a new query is built from a previous query. We tackle this problem and suggest a method that computes infix probabilities incrementally for probabilistic finite automata by representing all the probabilities of matching strings as a series of transition matrix calculations. We show that the proposed approach is theoretically faster than the previous method and, using real world data, demonstrate that our approach has vastly better performance in practice.

1 Introduction

Probabilistic grammars and finite automata are commonly used to model distributions in natural language processing. Among language models, probabilistic finite automata (PFAs) provide a simple, yet powerful and well-understood representation of many probabilistic language phenomena. Numerous speech processing tasks rely on PFAs in practice (Gyawali et al., 2013; Mohri et al., 2002; Wilson and Raaijmakers, 2008; Ng et al., 2000).

An important problem regarding PFAs is to calculate the probability of some affix (prefix, suffix, infix, etc.) of a string with respect to a given distribution. That is, given a PFA $\mathcal{P}$ and a string w, one might ask the probability of w appearing as a prefix, suffix, or infix in the distribution modeled by $\mathcal{P}$—in other words, the sum of the probabilities of all strings in the form of wx, xw, or xwy with respect to $\mathcal{P}$, for some strings x and y. A more general problem is to compute the sum of the probabilities of all strings in a regular language with respect to a PFA. Computing affix probabilities in probabilistic models is an important problem in natural language processing. For probabilistic context-free grammars (PCFGs) and PFAs, the problem of calculating the prefix or suffix probability of a string can be efficiently solved (Fred, 2000; Corazza et al., 1991). However, calculating the infix probability of a string or the weight of a regular language is not as straightforward (Corazza et al., 1991). Additionally, computing affix probabilities for more general probabilistic language models has proven to be quite difficult. Nevertheless, there are some approaches for computing the exact affix probabilities over a variety of models. Lattice posterior probabilities for n-grams, which are a restricted form of PFA (Vidal et al., 2005b), have a variety of uses in speech processing and can be computed efficiently (de Gispert et al., 2013; Can and Narayanan, 2015). For several affixes, Corazza et al. (1991) described algorithms to determine the probability of a string appearing as that affix in a PCFG. They made an important note that, unlike computing prefix or suffix probabilities, infix probability calculations are prone to double counting in the event of the infix appearing multiple times in a string. Then, they provided an algorithm for computing the infix probability of a string in restricted cases. Stolcke (1995) described a series of recurrences that can be used to compute affix probabilities and variations of the most probable parse of a string for PCFGs. For the general class of linear context-free rewriting systems, a method

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2732–2741
Brussels, Belgium, October 31 - November 4, 2018.

to compute prefix probabilities is known (Nederhof and Satta, 2011b). Fred (2000) also considered these problems and described a method to compute the infix probabilities under the condition that the infix appears at most once in any nonzero probability string when the language model is a stochastic regular grammar, which is equivalent in power to a PFA. These assumptions are rather strict and led researchers to consider a more general problem. Nederhof and Satta (2011a) solved the general infix probability problem for PCFGs. In fact, their method can be used to compute the weight of any regular language with respect to a PCFG. They also proposed an open problem of *incrementally* computing the infix probability of a string—using the numerical result of one infix computation to speed up the evaluation of another.

We design a new method for solving the infix probability problem for PFAs incrementally. Unlike the previous methods involving recurrence calculations or intersection constructions, our method is based on evaluating a series of matrices formed from regular expressions. Additionally, our method has no constraints on the input string. We show that our method is both theoretically and practically more performant than the previous algorithms. Our experimental results show a greater than 80% performance improvement. In Section 2, we review PFAs and other necessary formalisms. We recall how to obtain unambiguous regular expressions in Section 3, and introduce a matrix representation for computing the weight of a regular language in Section 4. In Section 5, we propose an algorithm to incrementally compute the infix probability of a given string. We validate the practical performance of the incremental method using a test set of PFAs obtained from real life data in Section 6 and conclude with a discussion and some open problems in Section 7.

2 Preliminaries

2.1 Finite Automata and PFAs

Following standard notation in automata theory, we let Σ be a set of characters and Σ^* be the set of all strings. For a string $w = w_1 w_2 \dots w_n \in \Sigma^*$, we write $|w| = n$ as its length. The empty string is written as λ.

A deterministic finite automaton (DFA) is a 5-tuple $\mathcal{D} = (Q, \Sigma, \delta, s, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $s \in Q$ is the initial state, and $F \subset Q$ is the set of final states. A language is a set of strings, and $\mathcal{D}$ recognizes a regular language denoted by $L(\mathcal{D})$. For a summary of automata theory (including regular expressions and formal language theory), we direct the reader to Hopcroft and Ullman (1979).

A PFA is a weighted finite automaton that computes a function $\mathcal{P} : \Sigma^* \to [0, 1]$. We abuse function notation and write $\mathcal{P}(w)$ or $\mathcal{P}(\pi)$ to denote the weight of a word or a path, respectively. These values are defined later. A PFA $\mathcal{P}$ is specified by a 5-tuple $\mathcal{P} = (Q, \Sigma, \delta, I, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma \times Q \to [0, 1]$ is the transition function, $I : Q \to [0, 1]$ and $F : Q \to [0, 1]$ are the initial and final functions, respectively. The transition function is assumed to have a default value of 0, in other words, if a transition does not exist it can be considered as having weight 0. A PFA has three additional requirements:

1. $\sum\limits_{q \in Q} I(q) = 1$
2. $\forall q \in Q,\ F(q) + \sum\limits_{q' \in Q, c \in \Sigma} \delta(q, c, q') = 1$
3. All states are both accessible and co-accessible[1].

If these conditions hold, then for all strings w, $0 \leq \mathcal{P}(w) \leq 1$ and $\sum_{w \in \Sigma^*} \mathcal{P}(w) = 1$. A PFA can be represented in the form of transition matrices, which simplifies several computations. We denote the matrix formulation of a PFA by $\mathcal{P} = (Q, \Sigma, \{\mathbb{M}(c)\}_{c \in \Sigma}, \mathbb{I}, \mathbb{F})$ where $\{\mathbb{M}(c)\}_{c \in \Sigma}$ is a set of $|Q| \times |Q|$ transition matrices with $\mathbb{M}(c)_{i,j} = \delta(q_i, c, q_j)$. Likewise, $\mathbb{I}$ and $\mathbb{F}$ are $1 \times |Q|$ and $|Q| \times 1$ vectors with $\mathbb{I}_i = I(q_i)$ and $\mathbb{F}_j = F(q_j)$.

Consider a string $w = w_1 w_2 \cdots w_n \in \Sigma^*$ and a corresponding labeled path

$$\pi = (q_0, w_1, q_1), (q_1, w_2, q_2), \dots, (q_{n-1}, w_n, q_n)$$

in $\mathcal{P}$. Then the probability of a path π in $\mathcal{P}$ is

$$\mathcal{P}(\pi) = I(q_0) \left(\prod_{i=1}^{n} \delta(q_{i-1}, w_i, q_i) \right) F(q_n).$$

Let Φ_w be the set of all labeled paths corresponding to w. The probability of w is now

---

[1] Accessible states are states reachable from a state with non-zero initial weight and co-accessible states are those that can reach a state with non-zero final weight.

$\sum_{\pi\in\Phi_w}\mathcal{P}(\pi)$. There exist two equivalent dynamic programming methods—the forwards and backwards algorithms—to compute the probability of a given string (Vidal et al., 2005a). Using the matrix formulation, the probability of a string is given succinctly as

$$\mathbb{I}\prod_{i=1}^{|w|}\mathbb{M}(w_i)\mathbb{F}.$$

For brevity, we write $\mathbb{M}(\Sigma) = \sum_{c\in\Sigma}\mathbb{M}(c)$ and $\mathbf{0}$ and $\mathbf{1}$ for the zero and identity matrices when the dimensions are clear. Further, we compute

$$\sum_{i=0}^{\infty}\mathbb{M}(\Sigma)^i = (\mathbf{1} - \mathbb{M}(\Sigma))^{-1},$$

which we denote $\mathbb{M}(\Sigma^*)$. We can compute the prefix and suffix probabilities of a string w as

$$\mathcal{P}(w\Sigma^*) = \mathbb{I}\left(\prod_{i=1}^{|w|}\mathbb{M}(w_i)\right)\mathbb{M}(\Sigma^*)\mathbb{F}$$

and

$$\mathcal{P}(\Sigma^* w) = \mathbb{I}\mathbb{M}(\Sigma^*)\left(\prod_{i=1}^{|w|}\mathbb{M}(w_i)\right)\mathbb{F},$$

respectively. If an automaton $\mathcal{M}$ satisfies all of the requirements for a PFA except that $\sum_{q\in Q} I(q) \leq 1$ or $\forall\ q \in Q$, $F(q) + \sum_{q'\in Q, c\in\Sigma}\delta(q,c,q') \leq 1$, then we call $\mathcal{M}$ a sub-PFA. These machines have the property that for any string w over Σ, $0 \leq \mathcal{M}(w) \leq 1$ and $\sum_{w\in\Sigma^*}\mathcal{M}(w) \leq 1$. Since a sub-PFA $\mathcal{M}$ may not describe a probability distribution over strings, we call $\mathcal{M}(w)$ the *weight* of w instead of probability. A stochastic language over an alphabet Σ is a set $\mathcal{S} \subseteq \Sigma^*$ where each string in $\mathcal{S}$ has an associated probability, $0 \leq Pr_{\mathcal{S}}(w) \leq 1$, such that $\sum_{w\in\Sigma^*} Pr_{\mathcal{S}}(w) = 1$. Given a stochastic language $\mathcal{S}$, if there exists a PFA $\mathcal{P}$ such that $\forall w \in \Sigma^*$, $\mathcal{P}(w) = Pr_{\mathcal{S}}(w)$, we call $\mathcal{S}$ a *regular stochastic language*.

2.2 Unambiguous Regular Expressions

Regular expressions are a common representation of regular languages. A string that can be matched with a regular expression is said to be in the language of the regular expression. A match occurs when there is a valid assignment of symbols in the regular expression to the queried string. We denote the set of all strings that match a regular expression as $L(R)$. An *unambiguous* regular expression has only one valid assignment for any string in the language. For example, consider the regular expression $(a \cup b)^*aa(a \cup b)^*$, which accepts the set of strings over $\Sigma = \{a, b\}$ containing aa as an infix. We assign different subscripts to symbols that appear in more than one position to form $(a_1 \cup b_1)^*a_2a_3(a_4 \cup b_2)^*$. Given the string $baaa$, we find that $b_1a_1a_2a_3$ and $b_1a_2a_3a_4$ are both valid assignments, hence $baaa$ is in the language. However, since there are two valid assignments, we say that this regular expression is ambiguous (Book et al., 1971). An unambiguous expression for the same language is $b^*a(bb^*a)^*a(a\cup b)^*$.

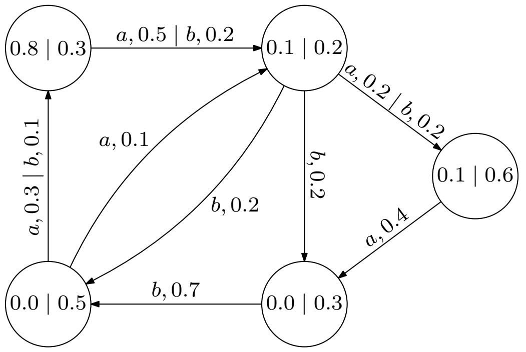

Figure 1: An example PFA over $\Sigma = \{a, b\}$. Each state has an initial and final probability, separated by a bar. The edges hold a character and the probability of the corresponding transition.

2.3 Intersecting a DFA and a PFA

The standard method to compute the weight of a regular language with respect to a PFA is to intersect its DFA representation with the PFA. This construction was already discussed in (Vidal et al., 2005a). Nederhof and Satta (2011a) considered a similar construction for DFAs and PCFGs and solved the infix problem for that class of machine. The construction creates a new sub-PFA $[\mathcal{D} \cap \mathcal{P}]$ such that:

$$[\mathcal{D} \cap \mathcal{P}](w) = \begin{cases}\mathcal{P}(w), & w \in L(\mathcal{D});\\ 0, & \text{otherwise.}\end{cases}$$

It follows that

$$\sum_{w\in\Sigma^*}[\mathcal{D} \cap \mathcal{P}](w) = \sum_{w\in L(\mathcal{D})}\mathcal{P}(w),$$

as desired.

The intersection algorithm is similar to that of the cross product of two automata from classical automata theory. Given a DFA $\mathcal{D}$ and a PFA $\mathcal{P}$, we construct a new machine $\mathcal{W}$ with states $Q_{\mathcal{W}} = Q_{\mathcal{D}} \times Q_{\mathcal{P}}$. For two states (x, y), (x', y') and a character $c \in \Sigma$, $\delta_{\mathcal{W}}((x, y), c, (x', y')) = \delta_{\mathcal{P}}(y, c, y')$ if $\delta_{\mathcal{D}}(x, c) = x'$ and 0 otherwise. Likewise, $I_{\mathcal{W}}((p, q)) = I_{\mathcal{P}}(q)$ if p is the initial state of $\mathcal{D}$ and $F_{\mathcal{W}}((p', q')) = F_{\mathcal{P}}(q')$ if p' is a final state of $\mathcal{D}$ and are 0 otherwise.

Using Algorithm 1, we can now efficiently compute the sum of the weight of all strings in $\mathcal{L}(\mathcal{D})$ by evaluating

$$\mathbb{I}_{[\mathcal{D}\,\cap\,\mathcal{P}]}(\mathbf{1} - \mathbb{M}_{[\mathcal{D}\,\cap\,\mathcal{P}]}(\Sigma))^{-1}\mathbb{F}_{[\mathcal{D}\,\cap\,\mathcal{P}]}.$$

Algorithm 1 DFA/PFA intersection

```
procedure INTERSECT(DFA $\mathcal{D}$, PFA $\mathcal{P}$)
    $Q' = Q_{\mathcal{D}} \times Q_{\mathcal{P}}$
    for $(d, p) \in Q'$ do
        if $d$ is $q_0$ then
            $I'((d, p)) = I(p)$
        else
            $I'((d, p)) = 0$
        end if
        if $d \in F_{\mathcal{D}}$ then
            $F'((d, p)) = F(p)$
        else
            $F'((d, p)) = 0$
        end if
        for $c \in \Sigma$, $(d', p') \in Q'$ do
            if $\delta_{\mathcal{D}}(d, c) = d'$ then
                $\delta'((d, p), c, (d', p')) = \delta_{\mathcal{P}}(p, c, p')$
            else
                $\delta'((d, p), c, (d', p')) = 0$
            end if
        end for
    end for
    return $[\mathcal{D} \cap \mathcal{P}] = (Q', \Sigma, \delta', I', F')$
end procedure
```

To find the infix probability of a given string $w = w_1 w_2 \cdots w_n$, we take $\mathcal{D}$ to be a DFA that recognizes the language of all strings containing w as an infix. The Knuth-Morris-Pratt (KMP) algorithm produces such a DFA (with $O(n)$ states) in $O(n)$ time (Knuth et al., 1977). Thus, computing the infix probabilities of each of w's prefixes takes $O(n(n|Q_{\mathcal{P}}|)^m)$ time, where m is the matrix multiplication constant[2], as at each step one needs to rebuild the entire intersection automaton for the current prefix and compute an inverse on its transition matrices.

Nederhof and Satta (2011a) demonstrated that a similar method can be used to compute the infix probability of a given string in PCFGs.

3 Generating Unambiguous Regular Expressions

Book et al. (1971) showed that, given a regular language (in the form of an automaton or regular expression), one can find an equivalent unambiguous regular expression by constructing an equivalent DFA and using state elimination on the resulting DFA.

Let $\mathcal{D} = (Q, \Sigma, \delta, q_1, F)$ be a DFA with $|Q| = n$ and the states being ordered from 1 to n. We add two new states, q_0 and q_{n+1} such that q_0 is the new start state and q_{n+1} is the only final state, and add the following transitions: $\delta(q_0, \lambda) = q_1$ and $\forall q \in F,\ \delta(q, \lambda) = q_{n+1}$. We then dynamically eliminate states using the following recurrence:

$$\alpha^k_{i,j} = \alpha^{k-1}_{i,j} + \alpha^{k-1}_{i,k}(\alpha^{k-1}_{k,k})^*\alpha^{k-1}_{k,j}$$

with the base cases:

$$\alpha^0_{i,j} = \begin{cases} \lambda, & i = 0, j = 1; \\ \lambda, & q_i \in F \wedge j = n + 1; \\ \{c \mid \delta(q_i, c) = q_j\}, & \text{otherwise.} \end{cases}$$

The equations follow the general concatenation, union, and Kleene star rules for regular expressions. In addition, we have:

- $\emptyset + c = c + \emptyset = c$, for $c \in \Sigma$
- $\emptyset c = c\emptyset = \emptyset$, for $c \in \Sigma$
- $\lambda c = c\lambda = c$, for $c \in \Sigma$
- $\emptyset^* = \lambda$.

The term $\alpha^{k-1}_{i,j}$ corresponds to the set of strings for which, starting from state q_i, describe a path to q_j where all intermediate states are of the form q_ℓ, where $\ell < k$ (the terminal state in the path has no such restriction). Similarly, $\alpha^{k-1}_{i,k}(\alpha^{k-1}_{k,k})^*\alpha^{k-1}_{k,j}$ corresponds to all of the strings which, beginning

---

[2] The matrix multiplication constant, m, is the order of the polynomial for the runtime of multiplying two $n \times n$ matrices together, i.e. a function in $O(n^m)$. In practice, Strassen's algorithm is often used, yielding $m \approx 2.81$ (Strassen, 1969).

at state q_i, end at q_j going through q_k, where all intermediate states have label at most k (McNaughton and Yamada, 1960).

We extract $\alpha^n_{0,n+1}$, which is an unambiguous regular expression for the language recognized by $\mathcal{D}$ (Book et al., 1971). Note that the requirement of an input automaton being deterministic is not strict. In fact, state elimination generates unambiguous regular expressions from any unambiguous automaton.

From now on, we only consider unambiguous regular expressions for any regular languages.

4 The Weight of a Regular Language

In Section 2.3, we reviewed the classical way of computing the weight of a regular language with respect to a PFA using an intersection construction. Here, we present a new method based on unambiguous regular expressions. We describe a simple transformation to convert regular expressions into operations on transition matrices. Consider the following mapping from regular expressions to matrices:

- $\emptyset \rightarrow \mathbf{0}$
- $\lambda \rightarrow \mathbf{1}$
- $c \in \Sigma \rightarrow \mathbb{M}(c)$.

Now, let R and S be regular expressions with $\mathbb{M}(R)$ and $\mathbb{M}(S)$ being their corresponding matrices:

- $R \cup S \rightarrow \mathbb{M}(R) + \mathbb{M}(S)$
- $RS \rightarrow \mathbb{M}(R)\mathbb{M}(S)$
- $R^* \rightarrow (\mathbf{1} - \mathbb{M}(R))^{-1}$.

Using these definitions, we can build a matrix calculation out of a given regular expression. We then obtain the weight of a regular expression R with respect to some PFA $\mathcal{P}$ by evaluating $\mathbb{I}_{\mathcal{P}}\mathbb{M}_{\mathcal{P}}(R)\mathbb{F}_{\mathcal{P}}$. However, the straightforward application of this method is prone to overcounting when there are many ways for a string to be matched to the expression.

We present a simple example where an ambiguous expression overcounts whereas an unambiguous expression returns the correct result when transformed into matrix calculations:

Let $R = b^*a(bb^*a)^*a(a \cup b)^*$ and $S = (a \cup b)^*aa(a \cup b)^*$, which are both regular expressions for all strings containing aa as an infix. Using the PFA in Figure 1, we have:

$$\mathbb{IM}(b^*a(bb^*a)^*a(a \cup b)^*)\mathbb{F} \approx 0.153,$$

$$\mathbb{IM}((a \cup b)^*aa(a \cup b)^*)\mathbb{F} \approx 0.198.$$

This gap can be made arbitrarily large by adding ambiguity to the regular expression without changing the described language.

We now show that, given a PFA $\mathcal{P}$ and a DFA $\mathcal{D}$, we can compute the weight of $L(\mathcal{D})$ with respect to $\mathcal{P}$.

Lemma 1. *Let $\mathcal{P}$ be a PFA, $\mathcal{D}$ be a DFA and R be an unambiguous regular expression for $L(\mathcal{D})$ generated by state elimination. Then* $\mathbb{I}_{\mathcal{P}}\mathbb{M}_{\mathcal{P}}(R)\mathbb{F}_{\mathcal{P}} = \sum_{w \in L(\mathcal{D})} \mathcal{P}(w)$.

Proof. Let $\mathcal{D}$ have n states, labeled q_1 to q_n. We proceed as in (Book et al., 1971) by adding two new states, q_0 and q_{n+1}. We fill out the base case table α^0. We then construct a new table, β where $\beta^k_{i,j} = \mathbb{M}_{\mathcal{P}}(\alpha^k_{i,j})$. In other words, α holds the regular expressions generated during state elimination while β holds the corresponding matrices. Since α^0 contains only unambiguous regular expressions, $\mathbb{M}_{\mathcal{P}}(\alpha^0_{i,j})$ is the matrix corresponding to the sum of $\mathbb{M}_P(w)$ for all w that, starting from state q_i travel to state q_j without passing through any states with label greater than 0. We continue the elimination process until we reach α^n and β^n. The regular expression in $\alpha^n_{0,n+1}$ corresponds to the unambiguous regular expression containing all strings accepted by $\mathcal{D}$. Thus, the matrix stored in $\beta^n_{0,n+1}$ is the matrix such that

$$\mathbb{I}_{\mathcal{P}}\mathbb{M}_{\mathcal{P}}(\alpha^n_{0,n+1})\mathbb{F}_{\mathcal{P}} = \mathbb{I}_{\mathcal{P}}\beta^n_{0,n+1}\mathbb{F}_{\mathcal{P}} = \sum_{w \in L(\mathcal{D})} \mathcal{P}(w).$$

□

5 Incremental Infix Calculation

We now tackle the open problem of incrementally computing the infix probability of a string with respect to a PFA. Suppose we have computed the infix probability of a string w. We want to use the result of that computation to compute the infix probability of wa without simply starting the computation again from scratch. Such a calculation is relatively easy for other affixes. For the prefix probability of a string represented by the unambiguous regular expression $w\Sigma^*$, one can simply compute $\mathbb{IM}(w)\mathbb{M}(\Sigma^*)\mathbb{F}$ and save the vector $\mathbb{IM}(w)$. When the prefix is extended to wa, the saved vector can be multiplied by $\mathbb{M}(a)\mathbb{M}(\Sigma^*)\mathbb{F}$ and obtain the result (we can save the vector $\mathbb{IM}(w)\mathbb{M}(a)$ to use in future incremental calculations). A similar process works for the incremental suffix probability of a string ($\Sigma^* w$ to $\Sigma^* aw$). These incremental approaches are due to the inherent unambiguity of regular expressions for strings appearing as a prefix or suffix.

Unfortunately, the analogous method for infix probabilities is not as straightforward. We cannot simply append (or prepend) the desired character to a precomputed regular expression because the resulting expression may not be unambiguous or may represent the a different language. To tackle this problem, we first define the language $\mathcal{F}(w)$ to be the set of strings that end in the first occurrence of w. In other words,

$$\mathcal{F}(w) = \{x \mid w \text{ appears only as a suffix of } x\}.$$

It follows that $\mathcal{F}(w) \cdot \Sigma^*$ is exactly the set of strings containing w as an infix. Thus, given an unambiguous regular expression for $\mathcal{F}(w)$, we can build an unambiguous regular expression for the infix of w by concatenating with Σ^*.

Next, we find a regular language $\mathcal{L}$ such that $\mathcal{F}(wa) = \mathcal{F}(w) \cdot \mathcal{L}$ for a character $a \in \Sigma$, which gives rise to an incremental computation using the previous result $\mathcal{F}(w)$. Given two languages R and S, we define the *left quotient* $R \backslash S$ to be:

$$R \backslash S = \{y \mid \exists x \in R \text{ such that } xy \in S\}.$$

It is known that regular languages are closed under the left quotient operation (Hopcroft and Ullman, 1979).

Corollary 2 follows from the definition of the left quotient and describes the desired $\mathcal{L}$.

Corollary 2. *Given* $\mathcal{F}(w)$ *and* $\mathcal{F}(w) \backslash \mathcal{F}(wa)$*,* $\mathcal{F}(wa) = \mathcal{F}(w) \cdot \mathcal{F}(w) \backslash \mathcal{F}(wa)$*.*

We use this characteristic and compute $\mathcal{F}(w) \backslash \mathcal{F}(wa)$ without explicitly computing $\mathcal{F}(wa)$ based on state elimination—the procedure is detailed later in this section. Figure 2 is an annotated example of unambiguous regular expressions for $\mathcal{F}$ of a, aa, and aab.

$$\mathcal{F}(a) = b^*a$$

$$\mathcal{F}(aa) = \underbrace{b^*a}_{\mathcal{F}(a)} \cdot \underbrace{(bb^*a)^*a}_{\mathcal{F}(a)\backslash\mathcal{F}(aa)}$$

$$\mathcal{F}(aab) = \underbrace{b^*a(bb^*a)^*a}_{\mathcal{F}(aa)} \cdot \underbrace{a^*b}_{\mathcal{F}(aa)\backslash\mathcal{F}(aab)}$$

Figure 2: An example of $\mathcal{F}$ for a, aa, aab. We annotate them to show how $\mathcal{F}$ can be built up incrementally using previously computed regular expressions.

Let $\mathcal{D}$ be the DFA for $\mathcal{F}(w_1 w_2 \cdots w_n)$ generated by the KMP algorithm (Knuth et al., 1977). Two examples are depicted in Figure 3. $\mathcal{D}$ has $n + 1$ states, with state q_{n+1} being final and having no outgoing transitions. Furthermore, each state q_i with $i < n + 1$ has exactly one outgoing transition to state q_{i+1} and all other transitions are to states with labels at most i. Thus, removing state q_{n+1} and all incoming and outgoing transitions and making q_n the only final state results in the DFA for $\mathcal{F}(w_1 w_2 \cdots w_{n-1})$. This process can be repeated until the empty string is reached. This leads to the observation that, when constructing an unambiguous expression for $\mathcal{F}(w_1 w_2 \cdots w_n)$, we can recover the expression for $\mathcal{F}(w_1 w_2 \cdots w_{n-1})$ extracting the regular expression at $\alpha_{0,n}^{n-1}$. Similarly, the expression for $\mathcal{F}(w_1 w_2 \cdots w_k)$ is stored at $\alpha_{0,k+1}^{k}$.

At stage k of the state elimination procedure on $\mathcal{D}$, state q_0 is only connected to states up to label $k - 1$, thus $\alpha_{0,k+1}^{k-1} = \emptyset$. Since

$$\alpha_{0,k+1}^{k} = \alpha_{0,k+1}^{k-1} + \alpha_{0,k}^{k-1}(\alpha_{k,k}^{k-1})^* \alpha_{k,k+1}^{k-1},$$

we can simplify the expression as

$$\alpha_{0,k+1}^{k} = \alpha_{0,k}^{k-1}(\alpha_{k,k}^{k-1})^* \alpha_{k,k+1}^{k-1}.$$

In addition, we recall

$$\alpha_{0,k}^{k-1} = \mathcal{F}(w_1 w_2 \cdots w_{k-1}).$$

Therefore,

$$\mathcal{F}(w_1 \cdots w_k) = \mathcal{F}(w_1 \cdots w_{k-1})(\alpha_{k,k}^{k-1})^* \alpha_{k,k+1}^{k-1}$$

and

$$(\alpha_{k,k}^{k-1})^* \alpha_{k,k+1}^{k-1} = \mathcal{F}(w_1 \cdots w_{k-1}) \backslash \mathcal{F}(w_1 \cdots w_k).$$

Algorithm 2 is for the offline setting—we know the entire string ahead of time and compute the infix probabilities of each of its prefixes incrementally. In Algorithm 2, following Section 3 and the relationship between the tables α and β in the proof of Lemma 1, we run the matrix evaluations of each regular expression instead of the

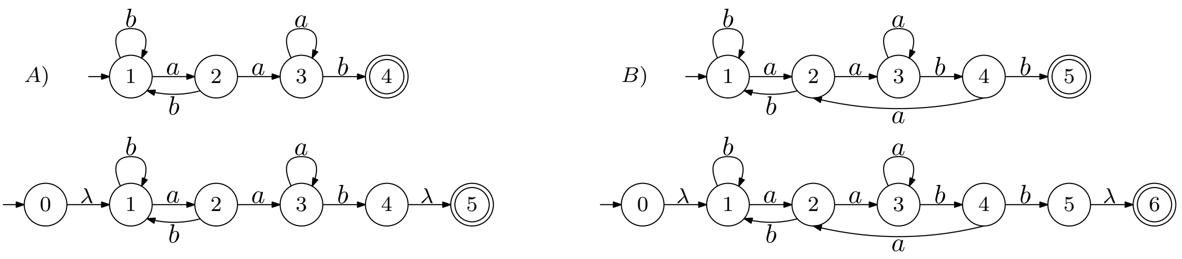

Figure 3: DFAs generated by the KMP algorithm. DFA A) accepts $\mathcal{F}(aab)$ and B) accepts $\mathcal{F}(aabb)$. The automata below are the new finite automata after adding the new initial and final state for the state elimination procedure, as described in Section 3.

regular expressions directly. At any step k, we only consider tables α^k and α^{k-1} and, thus, we can employ a standard sliding window technique for dynamic programming to reduce the required space complexity. In this scheme, instead of holding all tables up to α^k, we simply hold the two most previous ones, reducing the space complexity required by a factor of $O(k)$. We call our two tables T and T' and, to simplify the pseudocode, make elements of these two tables behave as both matrices and regular expressions. For example, $(T_{i,j})^*$ simultaneously corresponds to the regular expression $(\alpha_{i,j})^*$ and the matrix $\mathbb{M}((\alpha_{i,j})^*) = (\mathbf{1} - \mathbb{M}(\alpha_{i,j}))^{-1}$.

Algorithm 2 Offline Incremental Infix

```
 1: procedure INFIX($w = w_1 w_2 \cdots w_n \in \Sigma^*$)
 2:     $\mathcal{D} \leftarrow$ DFA accepting $\mathcal{F}(w)$
 3:     $T \leftarrow (n+3) \times (n+3)$ table
 4:     $T_{0,1} \leftarrow \mathbf{1}$
 5:     $T_{n+1,n+2} \leftarrow \mathbf{1}$
 6:     for $i \in [1, n+2]; j \in [1, n+2]; c \in \Sigma$ do
 7:         if $\delta(q_i, c) = q_j$ then
 8:             $T_{i,j} \leftarrow T_{i,j} + \mathbb{M}(c)$
 9:         end if
10:     end for
11:     $\mathbb{V} \leftarrow \mathbb{I}$
12:     for $k \in [0, n+1]$ do
13:         $\mathbb{V} \leftarrow \mathbb{V}(T_{k,k})^* T_{k,k+1}$
14:         yield $\mathbb{V}\mathbb{M}(\Sigma^*)\mathbb{F}$
15:         $T' \leftarrow (n+3) \times (n+3)$ table
16:         for $i \in [0, n+2]; j \in [0, n+2]$ do
17:             $T'_{i,j} \leftarrow T_{i,j} + T_{i,k}(T_{k,k})^* T_{k,j}$
18:         end for
19:         $T \leftarrow T'$
20:     end for
21: end procedure
```

Algorithm 2 begins by constructing DFA for $\mathcal{F}(w)$, which takes linear time in the length of w (Knuth et al., 1977) on Line 2. This DFA is modified to contain the new start and end state as described in Section 3 for a total of $n + 3$ states. We then begin to step through the state elimination algorithm. After constructing the initial table based on the base cases described in Section 3 from Line 3 to 10, we prepare our vector $\mathbb{V}$ that will record results from the previous infix calculation. At the beginning, $\mathbb{V}$ should hold $\mathbb{IM}(\mathcal{F}(\lambda)) = \mathbb{I}$. In the first step, we eliminate state q_1 of the automaton. Since state q_0 leads to state q_1 with a λ-transition, $\mathcal{F}$ at this step corresponds to $\mathcal{F}(\lambda)$ and $\mathcal{F}(\lambda) \cdot \Sigma^* = \Sigma^*$, which trivially has infix probability 1. At stage k of the for loop, we compute $\mathcal{F}(w_1 w_2 \cdots w_{k-1}) \backslash \mathcal{F}(w_1 w_2 \cdots w_k)$ on Line 13. We multiply this by $\mathbb{V}$, which holds $\mathcal{F}(w_1 w_2 \cdots w_{k-1})$, which makes $\mathbb{V}$ now store $\mathcal{F}(w_1 w_2 \cdots w_k)$. On Line 14, we emit the infix probability of $w_1 w_2 \cdots w_k$ by evaluating $\mathbb{V}\mathbb{M}_{\mathcal{P}}(\Sigma^*)\mathbb{F}_{\mathcal{P}}$. From Line 15 to 18, we update our state elimination table. On Line 19, we use the sliding window technique and copy our updated table into T which allows for T' to be safely overwritten in the next iteration. This process repeats until we have removed all states q_i for $1 \leq i \leq n+1$, and therefore have computed the infix probability of each prefix of w.

The loop starting on Line 12 executes $O(|w|)$ times since the automaton has $|w| + 3$ states. At each iteration, we recompute the state-elimination table, which has size $O(|w|^2)$. For each element of the table we must perform one matrix addition, two matrix multiplications, and one matrix inverse on the matrices from the PFA, which has overall time complexity $O(|Q|^m)$. Thus, the runtime of Algorithm 2 is $O(|w|(|w|^2|Q|^m)) = O(|w|^3|Q|^m)$, where m is the matrix multiplication constant. Throughout the computa-

States	$\lvert Q\rvert = 614$		$\lvert Q\rvert = 1028$		$\lvert Q\rvert = 1455$	
Infix Length	Incremental	Intersection	Incremental	Intersection	Incremental	Intersection
1	0.226	0.147	0.857	0.468	2.383	1.079
2	0.272	0.316	1.072	1.235	3.000	3.112
3	0.334	0.637	1.327	2.634	3.693	6.997
4	0.399	1.133	1.586	4.864	4.442	13.250
5	0.465	1.934	1.855	8.104	5.124	22.357
6	0.527	3.375	2.088	12.562	5.815	35.065
7	0.584	4.129	2.347	18.414	6.593	51.709
8	0.649	5.791	2.591	25.614	7.224	72.512
9	0.711	7.879	2.851	34.959	7.950	99.347
Total	4.169	25.342	16.574	108.853	46.224	305.428

Table 1: Experimental results (in seconds) for the incremental regular expression method and the intersection method. The average of 10 runs is reported. For the incremental test, we measure the time for the step of building the state elimination table and outputting the current infix probability.

tion, we generate a series of tables of size $O(|w| \times |w|)$ with elements of size $O(|Q_{\mathcal{P}}| \times |Q_{\mathcal{P}}|)$. However, since we use the sliding window technique, we only save $O(1)$ of these tables, leading to an overall space complexity of $O((|w||Q_{\mathcal{P}}|)^2)$. Note that the space complexity matches that of the intersection method, but the time complexity of the incremental method is asymptotically faster; $O(|w|^3|Q_{\mathcal{P}}|^m)$ compared to $O(|w|(|w||Q_{\mathcal{P}}|)^m) = O(|w|^{m+1}|Q_{\mathcal{P}}|^m)$.

6 Experimental Results

We verify the effectiveness of our method using real-world data. We present 3 n-grams[3] (two 2-grams and one 3-gram) and their 3 PFAs with 614, 1028, and 1455 states extracted from the Brown Corpus tagged with the Penn Tree Bank tagset using the NLTK library (Bird and Loper, 2004). As a preprocessing step, we convert all punctuation to the `PUNC` tag for a total of 35 tags. The n-grams are built by computing the probability of reading a given tag under the condition that we have just seen a specific sequence of n tags. We then randomly select a sequence of 9 tags as our string across all experiments. For each of the n-grams, we calculate the infix probability of each prefix of our input string using the incremental regular expression method and the intersection method. We record the average of 10 runs per infix calculation. All experiments were written in Python 3.5 and the automata and matrices were implemented using NumPy. The experiments were run on an AMD Ryzen 7 1700 (3.0 GHz) 8-Core Processor with 16GB of RAM.

In the worst-case (the case when $|Q| = 1028$ in Table 1), the cumulative time for the incremental shows an 556.77% speed-up. The largest individual infix probability calculation is in the experiment with $|Q| = 1455$ when calculating the infix of length 9, where the incremental method obtains a 560.76% speed-up. These results show that the incremental method is not only theoretically faster, but also much faster in practice when compared to the intersection method.

The runtime gap is quite large compared to what is expected from the theoretical analysis of the two algorithms. Additionally, in theory, the time for each step of the incremental calculation should remain essentially constant, but the experimental results show that it grows at a slow rate. These are most likely implementation issues stemming from the large memory requirements involved in the calculations.

The experimental results show that the incremental infix method vastly outperforms the naive intersection method. The ability to memoize previous calculations allows the incremental approach to calculate the next infix probability much faster than simply restarting the entire calculation. This empirically verifies the asymptotic analysis of the two approaches.

7 Conclusions

We have presented a new method for solving the open problem of incrementally computing the infix probabilities of a given string with respect to a PFA. Our method utilizes unambiguous regular expressions and is distinguished from the previous

---

[3] We have a similar performance improvement for all other test cases that we extract from the dataset.

methods in that it does not alter the structure of the PFA during the evaluation. Similarly to the algorithm presented in (Nederhof and Satta, 2011a), this method imposes no restrictions on the input string.

We have showed that our method is asymptotically faster than the previously best-known method. Furthermore, we have experimentally evaluated the performance of our algorithm on a real life dataset and have observed that the proposed algorithm performs significantly better than the intersection method in all cases.

Future directions of this line of research are to determine a method for two sided incremental infix computation—that is, given a computation for the infix of w, compute the infix of wa or aw at will. Currently, it is only possible to do one sided incremental infix calculations. Computing the infix incrementally in an online fashion—in which we do not know the entire string ahead of time and receive new characters in a stream—would be another improvement. We believe that the current method can be modified to work in an online setting, possibly with an increase in runtime. Furthermore, finding new classes of problems that can benefit from a similar incremental calculation is also interesting. Finally, extending this method to work for PCFGs and more complex probabilistic models is an important open problem.

Acknowledgements

The authors thank the reviewers for their detailed and constructive comments.

This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (2018-0-00247, 2018-0-00276).

References

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*.

Ronald Book, Shimon Even, Sheila Greiback, and Gene Ott. 1971. Ambiguity in graphs and expressions. *IEEE Transactions on Computers*, 20:149–153.

Dogan Can and Shrikanth Narayanan. 2015. A dynamic programming algorithm for computing n-gram posteriors from lattices. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 2388–2397.

Anna Corazza, Renato De Mori, Roberto Gretter, and Giorgio Satta. 1991. Computation of probabilities for an island-driven parser. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):936–950.

Ana L. N. Fred. 2000. Computation of substring probabilities in stochastic grammars. In *Grammatical Inference: Algorithms and Applications*, pages 103–114.

Adrià de Gispert, Graeme Blackwood, Gonzalo Iglesias, and William Byrne. 2013. N-gram posterior probability confidence measures for statistical machine translation: an empirical study. *Machine Translation*, 2:85–114.

Binod Gyawali, Gabriela Ramírez-de-la-Rosa, and Thamar Solorio. 2013. Native language identification: a simple n-gram based approach. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications, BEA@NAACL-HLT*, pages 224–231.

John E. Hopcroft and Jeff D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company.

Donald E. Knuth, Jr. James H. Morris, and Vaughan R. Pratt. 1977. Fast pattern matching in strings. *SIAM Journal on Computing*, 6:323–350.

Robert McNaughton and Hisao Yamada. 1960. Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers*, 9:39–47.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Mark-Jan Nederhof and Giorgio Satta. 2011a. Computation of infix probabilities for probabilistic context-free grammars. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1213–1221.

Mark-Jan Nederhof and Giorgio Satta. 2011b. Prefix probabilities for linear context-free rewriting systems. In *Proceedings of the 12th International Conference on Parsing*, pages 151–162.

Corinna Ng, Ross Wilkinson, and Justin Zobel. 2000. Experiments in spoken document retrieval using phoneme n-grams. *Speech Communication*, 32(1-2):61–77.

Andreas Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

Volker Strassen. 1969. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356.

Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. 2005a. Probabilistic finite-state machines–part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1013–1025.

Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. 2005b. Probabilistic finite-state machines–part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1026–1039.

Theresa Wilson and Stephan Raaijmakers. 2008. Comparing word, character, and phoneme n-grams for subjective utterance recognition. In *INTERSPEECH 2008, 9th Annual Conference of the International Speech Communication Association*, pages 1614–1617.

Syntax Encoding with Application in Authorship Attribution

Richong Zhang
Beihang University, Beijing, China
zhangrc@act.buaa.edu.cn

Zhiyuan Hu
Beijing University of Chemical Technology
zhiyuan.hu.bj@gmail.com

Hongyu Guo
National Research Council Canada
hongyu.guo@nrc-cnrc.gc.ca

Yongyi Mao
University of Ottawa, Ottawa, Ontario
ymao@uottawa.ca

Abstract

We propose a novel strategy to encode the syntax parse tree of sentence into a learnable distributed representation. The proposed syntax encoding scheme is provably information-lossless. In specific, an embedding vector is constructed for each word in the sentence, encoding the path in the syntax tree corresponding to the word. The one-to-one correspondence between these "syntax-embedding" vectors and the words (hence their embedding vectors) in the sentence makes it easy to integrate such a representation with all word-level NLP models. We empirically show the benefits of the syntax embeddings on the Authorship Attribution domain, where our approach improves upon the prior art and achieves new performance records on five benchmarking data sets.

1 Introduction

Syntactic parse information plays an essential role in interpreting natural languages because natural language sentences are typically structured in a linguistic grammar. As such, in many NLP applications it is desirable to extract syntactic features from text or sentences, for which there exist a rich body of literature (Baayen et al., 1996; Hirst, 2007; Massung et al., 2013; Wang et al., 2015; Socher et al., 2011; Zhu et al., 2015b; Tai et al., 2015; Zhu et al., 2015a)

To date, existing approaches to exploit syntactic information can be categorized into two categories. The first category may be regarded as "syntactic feature engineering". In such approaches, certain properties or statistics are extracted from the syntax parse tree of a sentence as the syntactical feature. For example, the extracted feature may include the depths of the tree, frequency of certain structural patterns in the tree and so on (Massung et al., 2013; Wang et al., 2015). The advantage of such a method is that the extracted feature can be used for any kind of classifier, if the feature is deemed relevant to the classification task. The limitation of such an approach is however that rich structural information contained in the syntax tree is lost in the feature extraction process. Additionally, with such a strategy, the model designer is often required to design syntactic feature extractors specific for his tasks.

The second category may be regarded as "syntax-assisted sentence coding". This category of approaches build upon neural network models. Examples of such approaches include Tree-LSTM (Tai et al., 2015; Zhu et al., 2015b) and Recursive Neural Networks (Socher et al., 2011), in which the networks are structured according to the syntax tree of the input sentence. The network, after being trained, is capable of encoding a sequence of word embeddings, in a bottom-up manner, to a vector representing the entire sentence. It is worth noting that with these approaches, the encoded feature vector, although containing syntactical information, mainly serves as a *semantic* representation of the input sentence, and the syntactic information exploited therein primarily serves to assist the semantic representation. Additionally, such an approach is not flexible enough to be integrated with another popular class of NLP models, CNN.

One motivation of this work is to develop a generic representation of the parse structure of sentences. Ideally, we would like the representation to maximally preserve the syntactical information and can be integrated easily with any neural network NLP models. This latter requirement is particularly desirable, in light of the competitive performance of CNN and their advantages in training efficiency.

Another motivation of this work is the application of Authorship Attribution (AA) (Hirst, 2007; Stamatatos, 2009; Ouamour and Sayoud, 2012; Stamatatos, 2011; De Vel et al., 2001; Rocha et al.,

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2742–2753
Brussels, Belgium, October 31 - November 4, 2018.

2017b; Binongo, 2003; Sohn et al., 2015; Shrestha et al., 2017b). In this application, one is to extract information from a document so as to infer the document's author, from a given list of candidates. There have been a variety of use cases of AA in practice, which include, amongst many others, plagiarism detection (Stamatatos and Koppel, 2011), forensics (Rocha et al., 2017a), suicidal note verification (Chaski, 2005), and intelligent question answering (Stamatatos, 2006). In this application, previous works have mostly focused on building a classifier based on content-level features. The hypothesis underlying our approach to AA in this paper is that syntactic information is a useful additional feature that can characterize an author. This is not only because that syntactical information complements the document content in language understanding, it is also due to the fact that different authors may construct sentences with varying distributions of syntactical structures. In a sense, the "syntactical pattern" of an author may characterize in part the "writing style" of the author, which is independent of the content he writes and the semantics of the document.

To that end, we propose a novel strategy to encode the syntax parse tree of sentence into a learnable distributed representation. Briefly, in this representation, an embedding vector is constructed for each word in the sentence, encoding the path in the syntax tree corresponding to the word. The one-to-one correspondence between these "syntax-embedding" vectors and the words (hence their embedding vectors) in the sentence makes it easy to integrate such a representation with all word-level NLP models.

The proposed syntax encoding scheme has a remarkable property, namely that it is provably information-lossless, as long as the syntax embedding space has adequate dimension. This property also distinguishes the proposed scheme from other distributed representations of syntax.

We apply the proposed syntax encoding approach to the state-of-the-art CNN-based AA model and evaluate its performance over five benchmarking datasets. Experimental study verifies the effectiveness of the proposed syntax encoding scheme. In fact, over all five datasets, syntax-augmented CNN model demonstrates new state-of-the-art performance.

2 Related Work

The aim of the related work here is twofold: syntax encoding and authorship attribution.

2.1 Syntax Encoding

Including syntactic parse information to benefit NLP models have been actively investigated in decades. Syntactic feature engineering refers to efforts to statically extract domain specific features from syntax parse tree of the given text (Massung et al., 2013; Wang et al., 2015). Recent attempts also include leveraging syntactic parse tree structure to recursively generate sentence representations bottom-up (Socher et al., 2011; Zhu et al., 2015b; Tai et al., 2015; Zhu et al., 2015a).

Both the above two categories of methods have severe limitations. The former parse representation typically fails to encode the parse tree structure, and the latter is constrained by the tree structures favored by the parser. Furthermore, the recent distributed word embedding techniques, such as Glove (Pennington et al., 2014) and W2V (Mikolov et al., 2013), have been shown to encode limited syntax knowledge of the given corpus (Andreas and Klein, 2014). This shortcoming has also promoted recent research on creating syntax-aware word embedding, which enhances the distributed embedding vectors with position information of the word within its surrounding context (Cheng and Kartsaklis, 2015), which again encodes limited syntax information.

Our syntax embedding method overcomes the above mentioned limitations, as previously discussed in Section 1.

2.2 Authorship Attribution

Various AA models make use of SVM classifiers on some carefully engineered lexical or syntactic feature. These works include (Pillay and Solorio, 2010; Varela et al., 2011; Segarra et al., 2013; Seker et al., 2013; Seroussi et al., 2010; Castillo et al., 2015). In many of these models and among many others (e.g., (Koppel et al., 2009; Peng et al., 2003; Apté et al., 1994)), character n-grams are chosen as an important feature.

Recently, researchers have relied on CNN to extract features automatically. In (Ferracane et al., 2017), for instance, a CNN is used on 2-gram embeddings to learn the discourse information for the AA task. In (Shrestha et al., 2017b), a CNN is employed on n-gram embeddings to learn richer

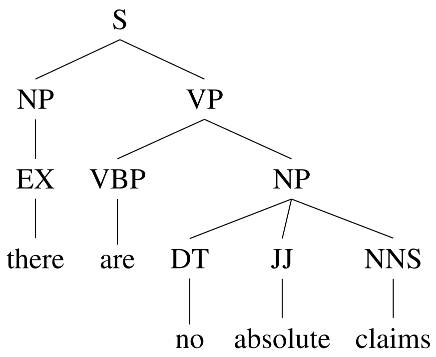

Figure 1: Syntax Tree Example

Words	Syntax Path
there	S→NP→EX
are	S→VP→VBP
no	S→VP→NP→DT
absolute	S→VP→NP→JJ
claims	S→VP→NP→NNS

Table 1: Syntax Path Example

features. A study by (Tang, 2013) considers using SVM as an activation layer for the CNN, replacing the soft-max layer. Nevertheless, a small performance advantage is demonstrated by the method at a high computation cost.

Our proposed model, apart from syntax encoding, is the most close to the CNN architecture of (Shrestha et al., 2017b) where n-grams are used to extract global content information representing content features. The detailed model architecture will be given in a later section.

3 Syntax Encoding

The syntactical structure of a given sentence can be uniquely represented by a tree, which we refer to as the *syntax tree*. An example of such a syntax tree is given in Figure 1. As seen in the example, a syntax tree has labeled nodes. Specifically, the label of each node is a "syntax token", such as `S`, `NP`, `VP`, etc., representing the grammatical property of the word sequences covered by tree branches underneath the node. For example, the root of the tree is always labeled by `S` ("sentence"), and the branches underneath the tree cover the entire sentence. On the other hand, the labels for terminals or leafs of the tree, such as `EX`, `VBP`, `JJ` etc., correspond to "part-of-speech" tags of each word in this sentence. We will denote by $\mathcal{T}$ the set of all syntax tokens.

Given this syntactic tree structure for a sentence s, each word w in sentence s has a unique path in the tree leaving the root and arriving at a terminal. Such a "syntax path" for the word w can then be represented by a sequence of node labels along the path. Some examples of syntax paths are given in Table 1. The following lemma is easy to verify.

Lemma 1 *Let a sentence s be written as a sequence of words $(w_1, w_2, \ldots, w_n)$. For each word position $i = 1, 2, \ldots, n$, let $r(w_i)$ denote the syntax path of word w_i. Let $\mathcal{R} := \{(i, r(w_i)) : i = 1, 2, \ldots, n\}$ be an (unordered) set containing precisely all syntax paths for the words in s. Then the syntax tree of s can be uniquely recovered by $\mathcal{R}$.*

In the lemma, we note that $\mathcal{R}$ is an unordered set. That is, regardless of the ordering of the paths in $\mathcal{R}$, one can always recover the syntax tree from $\mathcal{R}$.

Let $r(w)$ be the syntax path of a word w in the sentence s of interest. Specifically, $r(w)$ can be written as the sequence $(t_1, t_2, \ldots, t_L)$, where L is the number of nodes in the path r, and each t_i is a syntax token.

Let the Euclidean space $\mathbb{R}^K$ be the embedding space which we will use to encode syntax. We now describe a method that encodes the path $r(w)$ into a vector $\overline{r(w)} \in \mathbb{R}^K$.

Let $L_{\max}$ be maximum depth of the syntax trees in the corpus. For each $i = 1, 2, \ldots, L_{\max}$, let $\overline{p}_i$ be a vector in $\mathbb{R}^K$ serving as the embedding for integer i. Here integer i is meant to indicate the location of a token in a syntax path. For each $t \in \mathcal{T}$, let $\overline{t}$ also be a vector in $\mathbb{R}^K$, serving as the embedding for syntax token t. Let vector $\overline{r(w)} \in \mathbb{R}^K$, the embedding of path $r(w)$, be defined by

$$\overline{r(w)} := \sum_{t_j \in r(w)} \overline{t}_j \circ \overline{p}_j \tag{1}$$

where $\circ$ is the element-wise product operation. For example in Table 1, the syntax path for word "there" will have embedding $\overline{\mathrm{NP}} \circ \overline{p}_1 + \overline{\mathrm{EX}} \circ \overline{p}_2$; the syntax path for word "no" will have embedding $\overline{\mathrm{VP}} \circ \overline{p}_1 + \overline{\mathrm{NP}} \circ \overline{p}_2 + \overline{\mathrm{DT}} \circ \overline{p}_3$. Note that when embedding a syntax path, the beginning token S is removed from the path since it exists in every path.

Lemma 2 *There exists a random assignment of the vectors $\{\overline{t} : t \in \mathcal{T}\}$ and $\{\overline{p}_i : i = 1, 2, \ldots, L\}$ such that the syntax path $r(w)$ can be recovered from its embedding $\overline{r(w)}$ almost surely for sufficiently large K.*

This lemma (proof given in Appendix) suggests that as long as we choose a sufficiently large embedding dimension K, the above introduced encoding for syntax paths is essentially lossless. Thus, for a given n-word sentence $s = (w_1, w_2, \dots, w_n)$, if we collect the embedding vectors $\overline{r(w_1)}, \overline{r(w_2)}, \dots, \overline{r(w_n)}$ of all syntax paths and list them as the columns of a $K \times n$ matrix, then by Lemmas 1 and 2, the syntax tree of s can be recovered from the matrix. Such a matrix is then an information lossless encoding of the syntax tree.

We note however that in practice, when the tokens embeddings and the position (integer) embeddings are learned, there is no longer a guaranty that a syntax path can be recovered from its embedding. This is particularly the case for supervised tasks. During the training for such tasks, the information irrelevant to the training objective is necessarily "squeezed out", and the representations of those syntax paths that provide no distinguishing features are "pulled closer". This will cause these paths non-distinguishable (and hence non-recoverable) from their embeddings. This is also the reason that in practice there is no need to have very large embedding dimension K.

Nonetheless, since different supervised tasks may have distinct training objectives, a "lossy" syntax encoding suitable for one task may prove ineffective for other tasks. Thus it is still essential to adopt an information-lossless encoding framework, as we propose in this paper, that is universally applicable.

Next, we will discuss the application of our syntax encoding approach to AA models.

4 Authorship Attribution Model

4.1 Problem Definition

The objective of Authorship Attribution is to develop a classifier that predicts the authors of unseen documents based on a given set of documents and their corresponding authors. Despite the previous successes in solving the AA problem discussed in a previous section, we argue that the syntactic parse information in an author's writing can characterize in part the "writing style" of the author. Specifically, even when writing the same content, two authors may prefer using different syntax structure in constructing their sentences. Consider the following two sentences.

A1: Take a left at the end of the street, you will see the house.

A2: You will see the house, if you go down to the end of the street and take a left.

The two sentences have the same meaning and yet two different authors may favor different ones over the other. This provides opportunity for leveraging such syntactical information as an additional feature, beyond the lexical and semantic features, to distinguish the two authors.

Detecting the syntactical differences among authors suits well the application scope of the proposed syntax encoding scheme. As such, in this work, we will use the AA problem as an test bed to examine the effectiveness of our scheme.

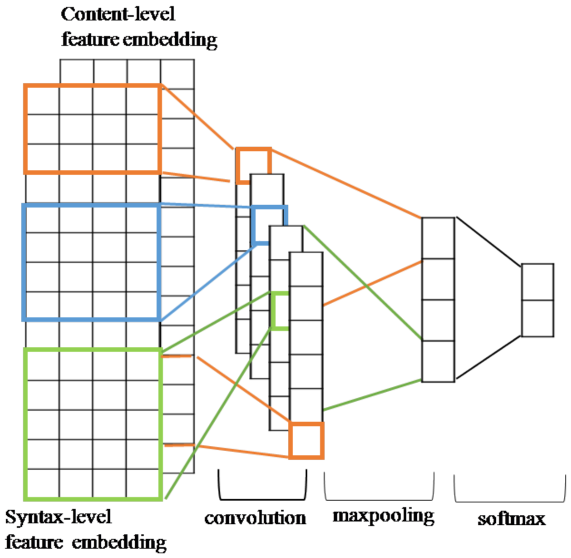

Figure 2: Single Layer CNN Architecture

4.2 Syntax-augmented CNN Model

The overall architecture of our model is shown in Figure 2. Our model takes advantage of the expressive power of convolutional neural networks(CNNs) to learn the embeddings for both of the content-level features and the syntax-level features. The model consists of 5 types of layers: *Syntax-level feature embedding, Content-level feature embedding, Convolution, Max-pooling* and *Softmax*.

4.2.1 Overall Structure

The model takes both the context-level features and the syntax-level features as the input.

As the character n-gram features have been used successfully in both of the text classification tasks (Kim, 2014; Chen et al., 2017) and the AA tasks (Shrestha et al., 2017b; Sapkota et al., 2015; Shrestha et al., 2017a; Sari et al., 2017; Ruder et al., 2016), in this study, we use the character n-grams to represent the content-level features.

We align in tandem the embedding vectors of consecutive n-grams of a document to form a content-level embedding matrix, and the consecutive syntax-path embeddings of each word to form the syntax-level embedding matrix. These embedding matrices are individually passed to 2 parallel CNNs having different filter lengths. The convolutional layer outputs feature maps via convolutional filtering. The max-pool layer is then applied across different feature maps to form the content-level and syntax-level representation of a document respectively. These two representations are then concatenated to form the final feature vector characterizing the author of the document. Finally, this feature vector is passed to a learnable softmax layer. The number of outputs of the softmax layer is the number of authors, which the i^{th} output is the probability that the document written by author i.

4.2.2 Convolution and Max-pool

In the convolutional layer, we capture local contextual information using kernels, which combine the vectors within it's window as it slides over the embedding matrix. A linear transformation processes the output of the kernel.

We present a clean description for this operation. Let $E \in \mathbb{R}^{k \times d}$ be the embedding matrix containing embeddings $e_{1:k}$, where $e_i \in \mathbb{R}^d$ is the i-th embedding in E, l is the number of filters and w is the filter length or window size. Let $F \in \mathbb{R}^{w \times d}$ be the filter matrix and a bias vector b. We define the vector $g_i \in \mathbb{R}^{w \times d}$ as the concatenation of w embeddings in the i-th window, where

$$g_i = e_{i-w+1:i} \quad 1 \leq i \leq k+w-1 \tag{2}$$

The result of filter F across embedding matrix E outputs a feature map $f_j \in \mathbb{R}^d$ for the j-th kernel where the i-th value of f_j is computed as

$$f_{ji} = F \otimes g_i + b \tag{3}$$

where $\otimes$ denotes the convolution operator. Let $f \in \mathbb{R}^{l \times d}$ denote the concatenation of the l feature maps for compactness. Further to this, we apply a ReLU activation function before passing the output to the max-pool layer. Given l filters of different dimensions, we obtain l feature maps. Each feature map encodes different types of abstract contextual information globally for the matrix E. To obtain the most relevant features from each dimension of the feature maps, we max-over feature map dimensions expressed as:

$$m_i = max(f(\cdot, i)), \quad 1 \leq i \leq d \tag{4}$$

where m_i represents the maximum value for dimension i across the l feature maps. The feature vector m from the max-pool operation is the concatenation of all m_i. For our 2 CNN's we obtain a content-level vector representation and a syntax-level representation for an author's text. We denote $\bar{m} \in \mathbb{R}^t$ as the concatenation of these 2 vectors representing an author's style. We obtain the confidence of an author's attribution to a text by feeding this final vector to a softmax layer. In the next section, we give a brief description of the softmax layer.

4.2.3 Softmax Layer

Given a text input x and CNN parameters θ, this layer outputs a score for all n authors. The output of the softmax layer is a vector with dimension equal to the number of author labels. To compute the confidence of author's attribution, the author's style representation $\bar{m}$ is transformed by a transformation matrix $W \in \mathbb{R}^{n \times t}$.

$$o = W\bar{m} \tag{5}$$

where the i-th component of o corresponds to the confidence score of author i. A softmax operation is called on o to compute the conditional probability for each dimension. This is calculated by:

$$p(i|x, \theta) = \frac{e^{o_i}}{\sum_{j=1}^{n} e^{o_j}} \tag{6}$$

To train our model, the log-likelihood of the probability should be maximized. To predict the attribution of authors, the label with the highest probability is selected. For the optimization of our model, we use SGD algorithm to solve the optimization problem.

We denote our syntax augmented CNN model as Syntax-CNN.

5 Experimental Studies

We first empirically show the predictive performance of the Syntax-CNN strategy, which establishes new state-of-the-art accuracy for several

Description	**CCAT10**	**CCAT50**	**IMDB62**	**blogs10**	**blogs50**
Avg # of words per doc	580	584	345	108.6	117.1
Avg # of chars per doc	3,089	3,010	1742	502.3	541.5
Docs per author	100	100	1000	2353.4	1470.1
Max # of chars	8,716	8,716	11617	30712	30712
Min # of chars	483	345	82	3	3

Table 2: Statistics of Datasets

benchmarking data sets. We then provide ablation studies, aiming at better understanding the contributions of the syntax embeddings to the Syntax-CNN method.

5.1 Experimental Setup

5.1.1 Datasets

We test the Syntax-CNN approach on several benchmarking datasets. Summary statistics of the datasets are in Table 2.

CCAT10: 100 newswire stories, written by 10 authors, in English taken from Reuters Corpus Volume 1 (RCV1) (Stamatatos, 2008).

CCAT50: Same as CCAT10 but with 100 news article written by 50 authors.

IMDB62: 62,000 movie reviews and 17,550 message posts from 62 prolific authors obtained from Internet Movie Database(IMDb) (Seroussi et al., 2010).

Blogs10: The original data set contains 681,288 blog posts by 19,320 bloggers for blogger.com. Posts written by the top ten bloggers are selected for the Blogs10 data set (Schler et al., 2006).

Blogs50: Same as Blogs50 but with the posts written by the top 50 bloggers.

5.1.2 Compared Prior Art

We compare our method with various baseline approaches, which represent the current art in the AA problem. They include SCAP (Frantzeskou et al., 2007), SVM with 2,500 most frequent 3-grams (Plakias and Stamatatos, 2008), SVM with bag of local histogram (Escalante et al., 2011), Imposters (Koppel et al., 2011), LDAH-S (Seroussi et al., 2011),SVM with affix and punctuation 3-grams (Sapkota et al., 2015), CNN-char (Ruder et al., 2016), Continuous n-gram representation (Sari et al., 2017), and N-gram CNN (Shrestha et al., 2017a). Except the last two methods, all the results reported in this paper were obtained from their respective papers. In all our experiments, we partitioned the datasets into train/dev/test in the same way as are used in the literature in order for fair comparison.

5.1.3 Hyperparameters and Training

Our experimental setup follows that of the current state-of-the-art AA method (Shrestha et al., 2017b). In specific, the networks are trained using mini-batches with size of either 16 (for IMDB62,Blogs10 and Blogs50) or 32 (for CCAT10 and CCAT50). We use 3-gram with embedding size of 300 for character, and embedding size of 60 for syntax vector; these embeddings are randomly initialized. We apply Adagrad (Duchi et al., 2011) with initial learning rate of 1e-4. For the CNN, we use filter sizes of 3, 4, and 5 with 50 feature maps for syntax embedding; 500 dimensional character embedding for CCAT, IMDB, and Blogs50, and 200 for Blog10. We train for at most 300 epochs, with a dropout rate of 0.25. We deploy early stop strategy for the training using a validation data set, which contain 10% of the randomly selected samples from the training set. We stop the training when the validation loss goes up.

In our experiments, we use the Stanford CoreNLP parser. For documents contain more than one sentences, each sentence is parsed separately. The syntax encoding for each word is done according to its syntax path in syntax tree containing the word. The syntax embeddings of all words in the document form the input matrix to the CNN responsible for extracting syntactical features.

5.2 Predictive Performance

Table 3 presents the accuracy obtained by various testing methods on the five benchmarking datasets, where the best result of each data set is highlighted in bold.

5.2.1 Accuracy Obtained by Syntax-CNN

Results in Table 3 indicate that the Syntax-CNN outperformed all the testing methods on the five benchmarking datasets, beating the current best records on these testing AA tasks. Our further

Model	**CCAT10**	**CCAT50**	**IMDB62**	**Blogs10**	**Blogs50**
SCAP$^{(2007)}$	#	#	94.8	48.6	41.6
SVM with most frequent 3-grams$^{(2008)}$	80.8	67	81.4	#	#
SVM with bag of local histogram$^{(2011)}$	86.4	#	#	#	#
Imposters$^{(2011)}$	#	#	76.9	35.4	22.6
LDAH-S$^{(2011)}$	#	#	72.0	52.5	18.3
SVM with affix+punctuation 3-grams$^{(2015)}$	78.8	69.3	#	#	#
CNN-char$^{(2016)}$	#	#	91.7	61.2	49.4
Continuous n-gram representations$^{(2017)}$	74.8	72.6	95.12	61.34	52.82
N-gram CNN$^{(2017)}$	86.8	76.5	95.21	63.74	53.09
Syntax	22.8	10.08	83.48	48.64	42.91
Syntax-CNN	**88.20**	**81.00**	**96.16**	**64.10**	**56.73**

Table 3: Accuracy obtained by the testing methods; best result for each data set is in **bold**.

Quality metrics	**CCAT10**	**CCAT50**	**IMDB62**	**Blogs10**	**Blogs50**
normal	18.58%	18.79%	55.81%	82.15%	82.13%
hard	5.96%	6.3%	19.05%	8.61%	8.9%
very hard	75.46%	74.91%	25.14%	9.24%	8.97%

Table 4: Syntax correctness as measured by the Heminway Editor tool

analysis also shows that, the predictive improvement on some datasets is large. For example, against the CCAT50 and Blogs50 datasets, the relative error reductions over the current state-of-the-art result are 5.5% and 3.64% , respectively.

5.2.2 Contribution of Syntax Encoding

We also include the accuracy obtained by using the syntax embeddings alone on the second last row in Table 3. These results indicate that using only the syntax style embeddings achieved very low accuracy on some of the datasets, for example, with only 10.08% on the CCAT50 dataset. As shown in Table 4, the CCAT50 is, indeed, one of hardest datasets, as judged by the Hemingway Editor tool [1], which aims to measure the syntax correction of a given piece of text. Nevertheless, such syntax style embeddings can bring significant accuracy gain to the CNN strategies. As shown in Table 3, for the CCAT50 dataset, a relative error reduction of 5.5% (which represents the largest error reduction of the five testing datasets) was achieved by Syntax-CNN over the best performed CNN model, i.e., the N-gram CNN. Similar behavior can also been observed for the CCAT10 dataset as shown in Tables 4 and 3. These results suggest that the Syntax-CNN may favor sentences with syntax difficulties.

We also provide, in Table 5, further information about the percentage of classifications corrected or mislabeled when enabling the syntax style embeddings in the Syntax-CNN method on both the CCAT10 and CCAT50 datasets. Table 5 clearly indicates that with the syntax style embeddings deployed, the Syntax-CNN was able to, respectively for the CCAT10 and CCAT50 datasets, correct 43.48% and 37.7% of the testing examples which were mislabeled when the syntax embeddings was disabled in the Syntax-CNN.

data set	**CCAT10**	**CCAT50**
wrong-to-correct	43.48%	37.7%
correct-to-wrong	4.87%	6.1%

Table 5: Percentage of classifications corrected (wrong-to-correct) or mislabeled (correct-to-wrong) when using the syntax style embeddings.

5.2.3 Syntactic Style Examples

The contributions of the syntax information is further justified by examining documents whose classifications are corrected when enabling the syntax embedding of the Syntax-CNN. For example, the following two excerpts, extracted from documents written by two different authors, share great semantic similarity. But they show a difference in

---

[1] http://www.hemingwayapp.com

terms of syntax information contained. In particular, the former is constructed by deep parse trees, but the later has a succinct parse tree structure.

> author1:
>
> China's long-delayed bid to enter the World Trade Organization will fall under the spotlight when the world's richest nations meet to discuss its application this week. United States says Beijing must comply with a "road map" to open its markets and eliminate trade and non-trade barriers before it can win U.S. support for its entry.
>
> author2:
>
> China must make real changes to its economy if it wants to join the World Trade Organization and should replace anti-U.S. rhetoric with international cooperation.

Using the two documents containing these two excerpts, we perform a separate small experiment. When we remove syntax encoding and its corresponding feature from Syntax-CNN, the model fails to classify the authors of the two documents. But when syntax encoding is included, the model can distinguish the two authors. Additionally, on the syntax encoding side, when we only keep the syntax path encoding for these two excerpts in their respective documents and remove all other sentences from the two documents, Syntax-CNN is still able to classify the two authors correctly.

This suggests that syntactic information is indeed useful for authorship attribution, and Syntax-CNN is able to extract such information effectively.

5.3 Model Behaviors

This section aims to further evaluate the behaviors of the Syntax-CNN model.

5.3.1 Sensitivity to Syntax Embedding Dimension

To understand the impact of the embedding size of the syntax tree in the Syntax-CNN method, we conduct further experiments on the Blogs50 dataset. We vary the dimension of the syntax tree embedding, from 5 up to 150. The experimental results are reported in Table 6.

Table 6 shows that Syntax-CNN is robust to the dimension size of syntax style embedding. The lowest accuracy was obtained by Syntax-CNN with an embedding dimension of 150. Although we have noted earlier that using a higher syntax embedding dimension allows the syntactical information to be better preserved. A downside of such a choice is however the risk of overfitting, which is expectedly the cause of the degradation of the performance at dimension 150. Note that even in this case, the performance of Syntax-CNN is still higher than the current state-of-the-art accuracy of 53.09% achieved by the N-gram CNN.

dimension	**Syntax-CNN**
5	54.96
30	55.86
60	56.82
100	56.42
150	53.34

Table 6: Accuracy obtained by Syntax-CNN on the Blogs50 while varying the syntax style embedding dimension.

data set	**with**	**without**
Blogs50	56.73	55.05
Blogs10	64.10	62.70

Table 7: Accuracy obtained by Syntax-CNN with and without position vector.

5.3.2 Impact of Position Vector

We further evaluate the impact of the position embedding vectors (i.e., the $\overline{p}_j$ vectors in Section 3) in syntax encoding. We conduct experiments on both the Blogs10 and Blogs50 datasets with and without the position embedding. In particular, by "without position", we mean that the element-wise multiplication with the $\overline{p}_j$ is removed from equation (1) in Section 3. The results are reported in Table 7.

Table 7 indicates that the position vectors play an important role in capturing syntactical information. In fact, it is possible to prove that without multiplying with the position vectors, in general the syntax tree is no longer recoverable from the encoding $\overline{r(w)}$ of syntax path $r(w)$. Additionally, the results in Table 7 suggest that depth information in the syntax tree is an important feature for distinguishing the writing styles of the authors.

6 Conclusion

We propose a novel strategy to encode a syntax tree into a learnable distributed representation.

This representation can be easily integrated into any NLP neural network model and entails no loss of information. Using this representation as an additional input, we extend the CNN architecture introduced in (Shrestha et al., 2017b) for the authorship attribution problem. Experimental evaluation of the extended model on the standard author attribution datasets demonstrates the effectiveness of the proposed syntax encoding approach. Record-breaking performances are obtained.

Acknowledgments

This work is supported partly by China 973 program (2015CB358700), by the National Natural Science Foundation of China (61772059,61421003), and by the Beijing Advanced Innovation Center for Big Data and Brain Computing and State Key Laboratory of Software Development Environment.

Appendix: Proof of Lemma 2

We first establish some elementary results.

Lemma 3 *Suppose that random variables* X, Y, Z, U *are independent standard normal random variables. Then*

1. $\mathrm{E}(XYZU) = 0$ *and* $\mathrm{VAR}(XYZU) = 1$.
2. $\mathrm{E}(X^2YZ) = 0$ *and* $\mathrm{VAR}(X^2YZ) = 3$.
3. $\mathrm{E}(X^2Y^2) = 1$ *and* $\mathrm{VAR}(X^2Y^2) = 8$.

These results follow from the independence among (X, Y, Z, U) and the fact that the square of a standard normal random variable is a Chi-Square random variable.

We now construct a random assignment scheme and a recovery scheme.

Random Assignment: Generate each vector in $\{\overline{t} : t \in \mathcal{T}\}$ and in $\{\overline{p}_i : i = 1, 2, \ldots, L_{\max}\}$ by assigning independent values drawn from a standard normal distribution.

Recovery Scheme: Let a path embedding $\overline{r(w)}$ be given. For each token-integer pair $(u, i) \in \mathcal{T} \times \{1, 2, \ldots, L_{\max}\}$, compute vector $\mathbf{a} \in \mathbb{R}^K$ and scalar b by

$$\mathbf{a} := \overline{r(w)} \circ \overline{u} \circ \overline{p}_i, \quad b := \frac{1}{K}\sum_{k=1}^{K} \mathbf{a}[k]$$

where $\mathbf{a}[k]$ is the k^{th} element of $\mathbf{a}$. Choose an arbitrary positive value $\epsilon < 1/2$. If $|b - 1| < \epsilon$, claim the i^{th} token on path $r(w)$ is u.

Now we prove that using this scheme, when the embedding dimension K is sufficiently large, one can recover the path $r(w)$ with probability arbitrarily close to 1.

First

$$\mathbf{a} = \sum_{t_j \in r(w)} \overline{t}_j \circ \overline{p_j} \circ \overline{u} \circ \overline{p}_i$$

Note that this summation contains summing of L K-vectors, which we will re-denote by $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_L$. Also we denote

$$C_l := \frac{1}{K}\sum_{k=1}^{K} \mathbf{c}_l[k].$$

Then we have

$$\mathbf{a} = \sum_{l=1}^{L} \mathbf{c}_l \tag{7}$$

and

$$b = \sum_{l=1}^{L} C_l \tag{8}$$

Each of $\mathbf{c}_l$ terms in Equation (7) corresponds to one of the four cases below.

Case 1: $i = j$ and $u = t_j$. This corresponds to case 3 of Lemma 3, and we have $\mathrm{E}(\mathbf{c}_l[k]) = 1$ and $\mathrm{VAR}(\mathbf{c}[k]) = 8$. It follows that $\mathrm{E}(C_l) = 1$ and $\mathrm{VAR}(C_l) = 8/K$.

Case 2: $i = j$ and $u \neq t_j$. This corresponds to case 2 of Lemma 3, and we have $\mathrm{E}(\mathbf{c}_l[k]) = 0$ and $\mathrm{VAR}(\mathbf{c}[k]) = 3$. It follows that $\mathrm{E}(C_l) = 0$ and $\mathrm{VAR}(C_l) = 3/K$.

Case 3: $i \neq j$ and $u = t_j$. This also corresponds to case 2 of Lemma 3, and we also have $\mathrm{E}(C_l) = 0$ and $\mathrm{VAR}(C_l) = 3/K$.

Case 4: $i \neq j$ and $u \neq t_j$. This corresponds to case 1 of Lemma 3, and we have $\mathrm{E}(\mathbf{c}_l[k]) = 0$ and $\mathrm{VAR}(\mathbf{c}[k]) = 1$. It follows that $\mathrm{E}(C_l) = 0$ and $\mathrm{VAR}(C_l) = 1/K$.

For each of these cases, the distribution of C_l can be made concentrated at its mean when K is made large enough (by invoking either the Weak Law of Large Number or the Central Limit Theorem). Therefore, if there exists a token in the path $r(w)$ that makes Case 1 satisfied, the distribution of b is concentrated at 1, and our recovery scheme will detect, with probability close to 1, the token and its position in the path. On the other hand, if there is no such token, the distribution of b is concentrated at 0, and with probability close to 0 our scheme will make a false detection. $\square$

References

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *ACL*, pages 822–827.

Chidanand Apté, Fred Damerau, and Sholom M Weiss. 1994. Towards language independent automated learning of text categorization models. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 23–30. Springer-Verlag New York, Inc.

H Baayen, H van Halteren, and F Tweedie. 1996. Outside the cave of shadows: using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing*, 11(3):121–132.

José Nilo G Binongo. 2003. Who wrote the 15th book of oz? an application of multivariate analysis to authorship attribution. *Chance*, 16(2):9–17.

Esteban Castillo, Darnes Vilarino, Ofelia Cervantes, and David Pinto. 2015. Author attribution using a graph based representation. In *2015 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 135–142.

Carole E. Chaski. 2005. Who's at the keyboard? authorship attribution in digital evidence investigations. *IJDE*, 4(1).

Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2017. Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, 72:221–230.

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In *EMNLP*, pages 1531–1542.

Olivier De Vel, Alison Anderson, Malcolm Corney, and George Mohay. 2001. Mining email content for author identification forensics. *ACM Sigmod Record*, 30(4):55–64.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Hugo Jair Escalante, Thamar Solorio, and Manuel Montes-y Gómez. 2011. Local histograms of character n-grams for authorship attribution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 288–298. Association for Computational Linguistics.

Elisa Ferracane, Su Wang, and Raymond Mooney. 2017. Leveraging discourse information effectively for authorship attribution. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 584–593.

Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, Carole E Chaski, and Blake Stephen Howald. 2007. Identifying authorship by byte-level n-grams: The source code author profile (scap) method. *International Journal of Digital Evidence*, 6(1):1–18.

Graeme Hirst. 2007. Bigrams of syntactic labels for authorship discrimination of short texts. In *Literary and Linguistic Computing*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the Association for Information Science and Technology*, 60(1):9–26.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011. Authorship attribution in the wild. *Language Resources and Evaluation*, 45(1):83–94.

Sean Massung, Chengxiang Zhai, and Julia Hockenmaier. 2013. Structural parse tree features for text representation. In *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, pages 9–16.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Siham Ouamour and Halim Sayoud. 2012. Authorship attribution of ancient texts written by ten arabic travelers using a smo-svm classifier. In *Communications and Information Technology (ICCIT), 2012 International Conference on*, pages 44–47. IEEE.

Fuchim Peng, Dale Schuurmans, Vlado Keselj, and Shaojun Wang. 2003. Automated authorship attribution with character level language models. In *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*, pages 267–274.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Sangita R Pillay and Thamar Solorio. 2010. Authorship attribution of web forum posts. In *eCrime Researchers Summit (ECrime), 2010*, pages 1–7. IEEE.

Spyridon Plakias and Efstathios Stamatatos. 2008. Tensor space models for authorship identification. *Artificial Intelligence: Theories, Models and Applications*, pages 239–249.

A. Rocha, W. J. Scheirer, C. W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A. R. B. Carvalho, and E. Stamatatos. 2017a. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security*, 12(1):5–33.

Anderson Rocha, Walter J Scheirer, Christopher W Forstall, Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne RB Carvalho, and Efstathios Stamatatos. 2017b. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security*, 12(1):5–33.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *arXiv preprint arXiv:1609.06686*.

Upendra Sapkota, Steven Bethard, Manuel Montes-y-Gmez, and Thamar Solorio. 2015. Not all character n -grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–102.

Yunita Sari, Andreas Vlachos, and Mark Stevenson. 2017. Continuous n-gram representations for authorship attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 267–273.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 6, pages 199–205.

Santiago Segarra, Mark Eisen, and Alejandro Ribeiro. 2013. Authorship attribution using function words adjacency networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5563–5567.

Sadi Evren Seker, Khaled Al-Naami, and Latifur Khan. 2013. Author attribution on streaming data. In *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on*, pages 497–503. IEEE.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2010. Collaborative inference of sentiments from texts. In *UMAP*, pages 195–206. Springer.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2011. Authorship attribution with latent dirichlet allocation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning, CoNLL 2011, Portland, Oregon, USA, June 23-24, 2011*, pages 181–189.

Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017a. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 669–674.

Prasha Shrestha, Sebastian Sierra, Fabio A González, Paolo Rosso, Manuel Montes-y Gómez, and Thamar Solorio. 2017b. Convolutional neural networks for authorship attribution of short texts. *EACL 2017*, page 669.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, pages 129–136.

Kyung-Ah Sohn, Tae-Sun Chung, et al. 2015. A graph model based author attribution technique for single-class e-mail classification. In *Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on*, pages 191–196. IEEE.

Efstathios Stamatatos. 2006. Authorship attribution based on feature set subspacing ensembles. 15:823–838.

Efstathios Stamatatos. 2008. Author identification: Using text sampling to handle the class imbalance problem. *Information Processing & Management*, 44(2):790–799.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the Association for Information Science and Technology*, 60(3):538–556.

Efstathios Stamatatos. 2011. Plagiarism detection using stopword n-grams. *Journal of the Association for Information Science and Technology*, 62(12):2512–2527.

Efstathios Stamatatos and Moshe Koppel. 2011. Plagiarism and authorship analysis: introduction to the special issue. *Language Resources and Evaluation*, 45(1):1–4.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, pages 1556–1566.

Yichuan Tang. 2013. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*.

Paulo Varela, Edson Justino, and Luiz S Oliveira. 2011. Selecting syntactic attributes for authorship attribution. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 167–172. IEEE.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *ACL*, pages 700–706.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015a. A re-ranking model for dependency parser with recursive convolutional neural network. In *ACL*, pages 1159–1168.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015b. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1604–1612.

Sanskrit Word Segmentation Using Character-level Recurrent and Convolutional Neural Networks

Oliver Hellwig
SFB 991, University of Düsseldorf
IVS, University of Zurich
hellwig7@gmx.de

Sebastian Nehrdich
Center for Buddhist Studies
University of Hamburg
nehrdbsd@googlemail.com

Abstract

The paper introduces end-to-end neural network models that tokenize Sanskrit by jointly splitting compounds and resolving phonetic merges (Sandhi). Tokenization of Sanskrit depends on local phonetic and distant semantic features that are incorporated using convolutional and recurrent elements. Contrary to most previous systems, our models do not require feature engineering or extern linguistic resources, but operate solely on parallel versions of raw and segmented text. The models discussed in this paper clearly improve over previous approaches to Sanskrit word segmentation. As they are language agnostic, we will demonstrate that they also outperform the state of the art for the related task of German compound splitting.

1 Introduction

Sanskrit is an Indo-Aryan language that served as lingua franca for the religious, scientific and literary communities of ancient India. Text production in Sanskrit started in the 2. millenium BCE and has continued until today.[1] A 19th century cataloguing project recorded more than 40,000 Sanskrit texts known at that time (Aufrecht, 1891–1903), which covers only a small part of the extant Sanskrit literature. Apart from the oldest Vedic texts, Sanskrit has little diachronic variation on the morphological level, because it was regularized by the grammarian Pāṇini in the 3rd c. BCE.

NLP of Sanskrit is challenging due to compounding (see Ex. 1) and the phonetic processes called Sandhi (‘connection’; see Ex. 2–5). Compounding is widely used in other languages, and NLP has developed methods for analyzing compounds (Macherey et al., 2011; Ma et al., 2016). In Sanskrit, however, syntactic co- and subordination tend to be diachronically replaced by compounding (Lowe 2015; see also Sec. 3), so that many sentences in later literature consist only of a few long compounds that are loosely connected by a semantically light verb or an (optional) copula, as shown in this example:

(1) *āśrayabhūtakhādikathanena*
foundation-become-air-etc.-mentioning
“(Something is described) by mentioning air etc. that have become [its] foundations.”

The term Sandhi denotes a set of phonetic processes by which the contact phonemes of neighboring word tokens are changed and merged, and which create unseparated strings spanning multiple tokens (Whitney, 1879). Sandhi occurs between adjacent vowels (vocalic Sandhi; Ex. 2), between consonants and vowels (Ex. 4) and between adjacent consonants (Ex. 5):

(2) *rājā+uvāca* ‘the king said’ $\xrightarrow{\text{ā+u=o}}$ *rāj*$\underline{o}$*vāca*

(3) **rāja+uvāca* ‘O king, he said’ $\xrightarrow{\text{a+u=o}}$ *rāj*$\underline{o}$*vāca*

(4) *prāc+eva* ‘before indeed’ $\xrightarrow{\text{c+e=ge}}$ *prā*$\underline{g}$*eva*

(5) *tad+hi* ‘because this ...’ $\xrightarrow{\text{d+h=ddh}}$ *tad*$\underline{dh}$*i*

In addition, Sandhi occurs between independent inflected words (Ex. 2–5) as well as between members of compounds.[2] Because different combinations of unsandhied phonemes can result in the same surface phoneme, Sandhi resolution is non-deterministic and depends on the semantic context of the sentence (see Ex. 3 for a morphologically and lexically valid, but semantically dispreferred reading of the string *rājovāca*).

---

[1] Text production was oral until the first centuries BCE (Falk, 1993). The texts were transmitted by memorization in this period, making them less (!) prone to transmission errors than in written form.

[2] The compound in Ex. 1 is split as *āśraya-bhūta-kha-ādi-kathanena*, and *kha +ādi = khādi* is a Sandhi phenomenon.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2754–2763
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

Scriptorial and editorial conventions further complicate the analysis of compounds and Sandhi. While most Indian manuscripts don't insert spaces between strings, modern editors use spaces a gusto. Moreover, the (correct) application of Sandhi is not followed by all authors and editors to the same extent, so that the unsandhied tokens *tat hi asti* ('as this is ...') can occur as *taddhyasti*, *taddhy asti*, *tad dhy asti*, *tad dhyasti* or even *tat hi asti* (unchanged).

Our models aim at transforming a given sentence into a sequence of unsandhied tokens. We refer to this task as *Sanskrit word splitting* (SWS), and subsume Sandhi and compounding phenomena under the common term *splits*. We address SWS by using a combination of convolutional and recurrent elements. The recurrent elements integrate sentence level information that leads to qualified decisions about the semantic meaningfulness of possible compound and Sandhi splits (see Ex. 2 and 3), while the convolutional elements are meant to replace n-gram extraction, which is frequently used in word segmentation architectures. As our models operate on the character level, SWS can be formulated in a sequence labeling framework.

Consequently, this paper has three main contributions:

1. We introduce novel character-based models for SWS that beat state of the art models by large margins.
2. We compare against sequence-to-sequence models and demonstrate that our models work on par with them, but need significantly less time for training and inference.
3. We publish a new dataset for Sanskrit word splitting that consists of more than 560,000 sentences with manually validated splits. The dataset and the code are released at `https://github.com/OliverHellwig/sanskrit/papers/2018emnlp`.

In the rest of this paper, we use the following terminology. A *token* is an unsandhied word that is not itself a compound. A *string* is a sequence of characters that is delimited by a space or a *daṇḍa*. Each string contains at least one token, at least one compound (that itself consists of at least two tokens) or a Sandhied mixture of both. A *sentence* is a piece of Sanskrit text that is terminated by the punctuation mark called *daṇḍa* "stick" (|) and consists of at least one string. Any sentence can consist of multiple independent clauses, which are not demarcated by punctuation in Sanskrit, or consist of a part of a larger clause only.

The paper proceeds as follows: Section 2 gives an overview of related work in NLP. Section 3 introduces our SWS dataset. Section 4 describes the sequence labeling models developed for this paper and three baseline systems, whose evalution is presented in Sec. 5. Section 6 summarizes the paper.

2 Related Research

Most NLP systems for SWS combine Pāṇinis phonetic and morphological rules with a lexical resource, either by using formal (Huet, 2005; Goyal et al., 2009; Kulkarni and Shukla, 2009) or statistical methods, including Dirichlet processes (Natarajan and Charniak, 2011), finite state methods (Mittal, 2010), graph queries (Krishna et al., 2016) and hybrid systems (Hellwig, 2015a).

A number of recent papers approaches SWS with deep learning models. Hellwig (2015b) splits isolated strings by applying a one-layer bidirectional LSTM to two parallel character based representations of a string. The restriction to isolated strings is problematic, because SWS relies on the grammatical and semantic context of the full sentence in many cases. Restricting a model to isolated strings ignores these linguistic clues.

Reddy et al. (2018) formulate SWS as a translation task on the sentence level. They transform surface and unsandhied sentences using the *sentencepiece* model and "translate" the surface into the unsandhied sentence using a seq2seq model with attention. Gantayat et al. (2018) use an encoder-decoder architecture with a global attention mechanism and apply their model to isolated strings from a small dataset (Bhardwaj et al., 2018). So far, no direct comparison of deep learning models for SWS has been done, because the authors used different, partly unpublished datasets and reported performance on different linguistic levels (sentence, string) and with different evaluation methods. We will therefore try to make a fair and comprehensive comparison with the state of the art in Sec. 5.

SWS is closely related to word segmentation for other Asian languages such as Thai (Haruechaiyasak et al., 2008), Chinese or Japanese

(Kanji), with most research being done for Chinese and Japanese. Contrary to Sanskrit, Chinese and Japanese don't exhibit Sandhi phenomena and their logographic scripts condense information, making it possible to use "word-level" CRFs on the output, for example. Chen et al. (2015) interpret Chinese word segmentation (CWS) as a sequence labeling task and evaluate a range of (stacked) bidirectional recurrent architectures that are combined with a final sentence level likelihood layer (Collobert et al., 2011) maximizing the transition score of the BMES encoded target sequence. Their best model uses a single layer bidirectional LSTM with bigrams of pre-trained character embeddings as inputs. Cai and Zhao (2016) deal with CWS by first forming word hypotheses from characters using a gated unit and then processing the word hypotheses with an LSTM-based language model. They minimize the combined word and sentence level scores using a structured margin loss and achieve better performance than Chen et al. (2015) on standard CWS datasets. Kitagawa and Komachi (2017) adapt the model proposed by Chen et al. (2015) for Japanese word splitting, but use characters, character n-grams and lexicon-based word boundary features as inputs. The authors report state of the art performance, but observe a clear drop in the F score of their model, when texts contain a high proportion of Hiragana characters and thus come closer to syllabic or alphabetic scripts.

3 Data

Several datasets for SWS have been published in the last years. While the dataset of Bhardwaj et al. (2018) may be too small and unvaried for training deep learning models, Krishna et al. (2017) re-analyze 560,000 sentences from the Digital Corpus of Sanskrit (DCS)[3] using the Sanskrit Heritage Reader (Goyal and Huet, 2016). Re-analysis is necessary, because the DCS stores the morpho-lexical analysis of strings, but does not record split points and Sandhi rules applied. Due to different linguistic choices (Pāṇinian vs. corpus-oriented) and to different ideas about the (non-)compositional meanings of compounds their final dataset contains only 115,000 sentences (see the discussion in Krishna et al. 2017 and the analysis in Sec. 3.1). As the size of the dataset is crucial for most deep learning methods, we decided to release a new dataset along with this paper. Each sentence contained in the DCS is re-analyzed using the SanskritTagger software (Hellwig, 2009). Our dataset contains the surface forms of sentences in the DCS and the split points and Sandhi rules that the tagger proposes for their morpho-lexical gold analyses stored in the DCS. We didn't differentiate between compound and inter-word splits, as this distinction introduces morphological categories into the dataset. Table 1 shows an example of the annotation format.

Surface	r	ā	j	o	v	ā	c	a
Unsandhied	r	ā	j	ā-u	v	ā	c	a

Table 1: Data extracted from the string *rājovāca*, which is split into the two tokens *rājā* ("king") and *uvāca* ("(he, she) said"; see Ex. 2).

Table 2 shows the statistics of our dataset, split by text genres (first column). The dataset contains 2,978,509 strings and 4,171,682 tokens in 561,596 sentences. Most sentences come from the Epic and scientific (medicine, alchemy, astronomy) domain. While Epic texts are mostly written in easy, plain Sanskrit, the scientific works use many uncommon terms (likely to reoccur in the lexicographic domain) and long compounds. Sentence length is higher in the prose subcorpora (Buddhist, Vedic prose, ritualistic texts).

The fourth column shows that split phenomena are frequent in Sanskrit, occurring for more than 8% of all characters. Columns 5 and 6 report the proportions of complicated splits in relation to all splits. While 15% of all splits are resolved into a vocalic Sandhi, compound breaks are the dominant split type, which is also responsible for the majority of errors and ambiguities (see Sections 3.1 and 5.2). The last column also reflects the diachronic development from earlier texts with limited compounding (Vedic, ritualistic and Dharma texts) towards classical Sanskrit, which shows a strong preference for compounding. We use a fixed split of 90% of the sentences for training, a development set of 5% for parameter optimization and 5% for testing.

---

[3] `http://kjc-sv013.kjc.uni-heidelberg.de/dcs/`

Genre	#sen	$\lvert\bar{S}\rvert$	$\frac{s\bar{p}l}{S}$	$\frac{v\bar{o}c}{spl}$	$\frac{\bar{cp}}{spl}$
Epos	322811	42	0.081	0.144	0.355
Science	105597	50	0.096	0.148	0.478
Literature	36989	50	0.085	0.173	0.382
Religious	24055	48	0.091	0.174	0.413
Dharma	18506	42	0.08	0.162	0.276
Buddhist	13739	78	0.083	0.143	0.442
Lexicogragphy	13015	44	0.077	0.146	0.376
Vedic prose	11425	58	0.081	0.181	0.097
Philosopy	7277	50	0.088	0.185	0.379
Vedic poetry	4355	46	0.071	0.106	0.079
Ritual	3222	71	0.069	0.183	0.215
Grammar	605	32	0.054	0.194	0.344
Overall	561596	46	0.084	0.150	0.375

Table 2: Statistics of the full dataset; $|\bar{S}|$: average sentence length in characters; columns 4-6 give the average proportion of splits/string $\frac{s\bar{p}l}{S}$, of vocalic Sandhis/split $\frac{v\bar{o}c}{spl}$ and of compound splits/split $\frac{\bar{cp}}{spl}$

3.1 Quality of the training data

The dataset released by Krishna et al. (2017) and the one released with this paper both build on the DCS as gold standard. As this corpus was curated by a single user and the project never released a proper annotation guideline, one may suspect that it contains a certain level of inconsistencies and errors that influence the quality of the models and impose an upper limit for the model accuracy. In order to estimate the size of these effects, the authors of this paper independently corrected the analyses of 50 sentences randomly drawn from the training set (250 words, 2,354 characters including spaces). The corrections made by the authors differed at 23 character positions, corresponding to 20 strings in 15 sentences. 16 of these differences concerned compound splits, where the authors disagreed about the (non-)compositional meaning of compounds. A good example for such a disagreement is the string *rājayoga*, which was split as *rāja-yoga* “king-Yoga” = “Yoga of a king” by one author (compositional reading), but left unchanged as the name of a school of Yoga by the other one (non-compositional reading). After adjudicating these disagreements, there remain 5 of 250 strings with annotation errors in the training data, which corresponds to an error level of 2% of all strings and 0.2% of all characters for this sample.

We further explored the effect of compositionality by independently splitting 56 sentences of the Buddhist treatise Triṃśikāvijñaptibhāṣya, which is not part of the DCS. As the text uses highly technical terminology, the degree of disagreement can be expected to be higher than for plain narrative texts. We adjudicated our Sandhi annotations, but kept conflicts in compound splitting unresolved. 94.5% of all strings (394 of 417) and 69.7% of all sentences obtained the same compound analysis by both authors. Again, the majority of differences (11 of 23) showed up when a compound can have a non-compositional meaning that is closely connected with its compositional reading. Evaluation will show that these cases are responsible for a large parts of the model errors.

4 Models

4.1 Input Features

The character based models are trained with embeddings of the indidual surface characters, which are initialized with uniform random values from $[-1, +1]$ and updated during training. Following Kitagawa and Komachi (2017), the input can be enriched with multinomial *split probabilities* that are built from the training data. When the training data contain a split rule for surface character t_i at position i, we extract left ($g^L_{i,n}$) and right ($g^R_{i,n}$) character n-grams with lengths $n \in [2, 7]$[4] that end/start at position i, so that $g^L_{i,n} = \{t_{i-n+1}, t_{i-n+2}, \ldots t_i\}$ and $g^R_{i,n} = \{t_i, t_{i+1}, \ldots t_{i+n-1}\}$. Counts $\#(.)$ for individual n-grams are accumulated over the whole training set. At training and test time, a vector $\mathbf{v}_p \in \mathbb{R}^{2\cdot(7-2+1)=12}$ is assigned to each character position. Its element corresponding to the left n-gram of length 2, for example, is calculated as

$$\mathbf{v}_p(L, 2) = \frac{\#(g^L_{i,2})}{\max \#(g^L_{*,2})} \qquad (6)$$

We evaluate the influence of split probabilities in the ablation study (Sec. 5.2).

4.2 Extern Models for Comparison

We compare our models against the following baselines:

Bidirectional RNN We re-implement the model described in Hellwig (2015b), but apply it to full sentences instead of isolated strings. Character embeddings are fed into a bidirectional recurrent

---

[4] Longer n-grams did not produce improvements on the dev set.

layer with LSTM units. The output of the recurrent layer is additionally regularized by using dropout (Srivastava et al., 2014), and classification is performed using softmax with cross-entropy loss. We decode the output of the softmax in a greedy fashion without considering interactions between adjacent output classes.

seq2seq We retrain the model described in Reddy et al. (2018) with our data after pre-processing them with the unsupervised text tokenizer *sentencepiece* (Schuster and Nakajima, 2012).[5]

Transformer As an alternative to recurrency based seq2seq, we apply the model described in Vaswani et al. (2017) to the input pre-processed with *sentencepiece*. This model relies entirely on an attention mechanism to draw global dependencies between input and output. To our best knowledge, this is the first time that this model has been used for SWS. We use the publicy available implementation tensor2tensor.[6].

4.3 Models Combining RNN and CNN

Convolutional Element Combinations of recurrent and convolutional (LeCun et al., 1998) elements are effective for tasks where complex local features are extracted by the convolutional element and then considered in larger contexts by the recurrent element (and vice versa; see Bjerva et al. 2016 or Ma and Hovy 2016). We use convolutional features c_i as proposed by Kim (2014). Let w denote the width of the input matrix $\mathbf{X}$ of the convolution (= number of time steps), h its height, n the width of the convolutional filter $f^n \in \mathbb{R}^{n \times h}$, $\sigma(.)$ a non-linearity (Rectified Linear Units (Nair and Hinton, 2010) in this paper) and b a bias. A convolutional feature at character position i and for filter j is defined as:

$$c^n_{ij} = \sigma(f^n_j \cdot \mathbf{X}_{[i:i+n-1,*]} + b) \tag{7}$$

The feature map $\mathbf{c}^n_i$ for m different filters is formed by concatenating the convolutional features ($\mathbf{c}^n_i = [c^n_{i1}, c^n_{i2}, \ldots c^n_{im}]$) and the output $\mathbf{c}$ of the convolutional element is formed by concatenating the feature maps ($\mathbf{c}_i = c^1_i \oplus c^3_i \oplus \ldots$). We use use odd filter widths only to avoid problems with patch alignment. We tested convolution with small quadratic filters as used in image convolution as well as other methods for combining the learned filter such as averaging, addition or max-pooling of the stacked filters, but did not observe improved performance on the dev set.

Model 1: Convolution → Recurrency (crNN) As an alternative to n-gram extraction (Chen et al., 2015; Kitagawa and Komachi, 2017), a convolutional element is applied to the character embeddings (see Fig. 1a). Its outputs (Eq. 7) are fed into a bidirectional recurrent layer (Schuster and Paliwal, 1997). As in the baseline RNN (Sec. 4.2), dropout is inserted after the recurrent layer, and classification is performed using softmax with cross-entropy loss and greedy decoding.

Model 2: Recurrency → Convolution (rcNN) The order of convolutional and recurrent elements is switched (see Fig. 1b), so that the convolutional operation replaces additive n-gram formation before the classification layer. The remaining architecture is identical to that of crNN

Model 3: rcNN with Shortcuts ($\mathbf{rcNN_{short}}$) This model extends rcNN by adding shortcut connections (Bishop, 2000) that concatenate the character embeddings and the RNN outputs with the concatenated feature maps $\mathbf{c}$ (see Fig. 1c). When $\mathbf{e}_i$ denotes the embedding of character i and $\mathbf{r}_i$ the output of the recurrent layer at position i, the input to the classification layer is defined as $\mathbf{e}_i \oplus \mathbf{r}_i \oplus \mathbf{c}_i$. Shortcuts are evaluated because we hypothesized that the access to unconvolved information about the input sequence and the output of the recurrent layer would facilitate the exact prediction of split locations. For a better control of information flow, we also experimented with residual (He et al., 2016) and highway (Srivastava et al., 2015) instead of shortcut layers, but could not observe improvements on the dev set, most probably because our models are not deep enough for these layer types to show effects.

5 Evaluation

5.1 Evaluation Settings

We use the following settings found on the dev set for the character based models: embedding size: 128; 200 hidden recurrent units; 100 convolutional feature maps with filter widths of 3,5 and 7. We use regularized (Zaremba et al., 2014) instead of

---

[5] Code for the model: https://github.com/cvikasreddy/skt; for the tokenizer: https://github.com/google/sentencepiece

[6] https://github.com/tensorflow/tensor2tensor

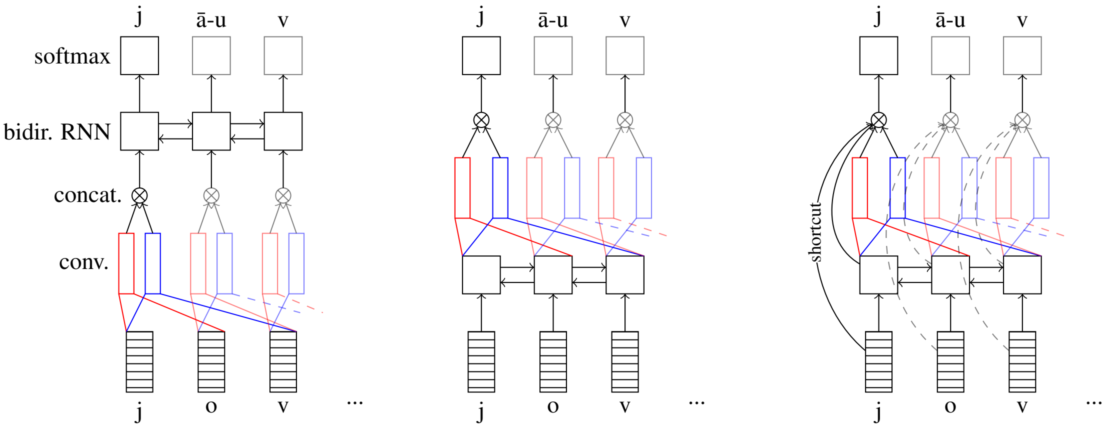

(a) crNN (b) rcNN (c) rcNN$_{\text{short}}$

Figure 1: Character based models, unfolded for the sequence labeling task $j{+}o{+}v \rightarrow j{+}a\text{-}u{+}v$.

vanilla LSTM units. All models are trained with the Adam optimizer (Kingma and Ba, 2015), an initial learning rate of 0.005 and batch size of 100. Gradients with a magnitude higher than 5.0 are cut. The models used for model selection (Sec. 5.2) are trained for 5, the other character-based models for 10 iterations. We train the Transformer in its default configuration as described in Vaswani et al. (2017) with a vocabulary size 5k[7] and report performance on the test set based on evaluations on the dev set. The model of Reddy et al. (2018) is trained for 80 epochs with our training data and the same parameters as described in the original paper. All calculations are run on a Maxwell Titan X GPU. We compare the models using sentence accuracy ($\frac{\text{\#sens. with errors}}{\text{\#all sens.}}$) and string based P(recision), R(ecall) and F score (Ma et al., 2016), where P and R are equivalent to the measures used in the CWS bakeoffs (Sproat and Emerson, 2003).

5.2 Model Selection

The upper half of Tab. 3 compares the evaluation metrics for the three character based models introduced in this paper trained with and without split probabilities (Sec. 4.1). We test differences in string accuracy using the McNemar test.[8] In general, all models that use recurrency *before* convolution (rcNN*) have string accuracy rates that are significantly higher at the 0.001 level than for models that use convolution before recurrency (crNN).

Table 3 shows that the differences in the performance of crNN and rcNN* are almost as large as between the RNN baseline and the best model from this paper (lower half of Tab. 3), although crNN and rcNN* differ only by the switched order of recurrent and convolutional elements. We found this result surprising, because applying convolution to the character embeddings appeared like a good parametrized alternative to n-gram extraction, which is often the first step in architectures for Chinese and Japanese word segmentation.

To further investigate this phenomenon, we evaluated 60 randomly chosen strings from the test set in which either crNN$^{\text{split}}$ or (XOR) rcNN$^{\text{split}}_{\text{short}}$ made an error. 45 of the errors relate to compound splitting, partly combined with vocalic Sandhi, either by missing a split (rcNN$^{\text{split}}_{\text{short}}$: 11, crNN$^{\text{split}}$: 15) or by oversegmenting compounds (rcNN$^{\text{split}}_{\text{short}}$: 13, crNN$^{\text{split}}$: 6). Most notably, rcNN$^{\text{split}}_{\text{short}}$ tends to insert more splits than crNN$^{\text{split}}$. This behavior can be observed for missing splits and especially for oversegmentations. A more detailed inspection shows that 11 of 13 oversegmentations actually induce a compositional reading of a compound. *saralāṅga* “name of a pine resin”, for example, is oversegmented into *sarala-aṅga* “pine-limb”, which is the etymological derivation of this compound. In contrast, crNN creates oversegmentations such as *śṛṅgavanti-aḥ*, where *śṛṅgavanti* “having horns” (nom. pl. neutre) is a valid form, while *aḥ* is not an independent word form in Sanskrit. Interestingly, rcNN$^{\text{split}}_{\text{short}}$ mis-segments the same string into *śṛṅga-vantyaḥ* in another sentence of the test set. Though differing from the

---

[7] Larger vocabulary sizes did not improve on the dev set, but performance gains by further decreasing the vocabulary size appear to be possible.

[8] Testing the sentence accuracy produced highly correlated test statistics. Results are therefore not discussed.

Model	sp?	Sen. A	String P	String R	String A
Model selection					
crNN		75.7	90.8	90.8	94.2
crNN	✓	75.4	90.6	90.9	94.2
rcNN		81.9	93.2	93.6	95.8
rcNN	✓	*82.2*	*93.3*	*93.8*	*95.9*
rcNN$_{\text{short}}$		81.7	93.1	93.5	95.8
rcNN$_{\text{short}}$	✓	**82.6**	**93.6**	**93.8**	**96**
Comparison with other models					
Hellwig (2015b)		77.7	91.8	91.8	94.8
Reddy et al. (2018)		72.3	90.2	88.4	93.3
Transformer 5K		84.9	**94.9**	94.5	96.5
rcNN$^{\text{split}}_{\text{short}}$	✓	**85.2**	94.6	**94.8**	**96.7**

Table 3: Upper half: Results for model selection (Sec. 5.2); lower half: Comparison with baseline models (Sec. 4.2 and 5.3)

Rule	Hellwig (2015b) P	R	F	rcNN$^{\text{split}}_{\text{short}}$ (this paper) P	R	F
ā	98.09	97.82	97.95	99.24	99.1	99.17
ā-	84.6	87.72	86.13	89.81	93.27	91.5
a-a	89.08	92.67	90.84	94.36	96.34	95.34
a-ā	88.26	86.48	87.36	91.5	95.6	93.51
ā-a	83.59	75	79.06	92.29	84.76	88.36
ā-ā	72.45	58.97	65.02	90.76	72.24	80.45
āḥ	73.13	77.66	75.33	91.48	89.44	90.45

Table 4: P, R and F for rules that produce the surface phoneme *ā*. Data in the left half are from the original publication. As all metrics are consistently better for this paper, we refrain from highlighting the best results in the right half of the table.

gold analysis, this segmentation gives the correct derivational analysis of the adjective (noun *śṛṅga* "horn" + inflected form of the adjectivizing possessive affix *-vat*). The results of rcNN$^{\text{split}}_{\text{short}}$ thus reflect the inherent inconsistencies of the dataset on the level of compound splitting (see Sec. 3.1), and their erroneous splits are frequently semantically meaningful while glossing over minute semantic distinctions. Errors of crNN, in contrast, tend to be real mis-segmentations, indicating that its ability to reflect the semantic level is underdeveloped.

Split probabilities (Sec. 4.1) have a small, but positive effect on string accuracy of the rcNN* models. When the same model with and without split probabilities is compared using the McNemar test, split probabilities significantly increase string accuracy at the 0.1 level for rcNN and at the 0.001 level for rcNN$_{\text{short}}$, while they don't result in significantly better performance for crNN.

5.3 Comparison with Baseline Models

The lower half of Tab. 3 compares the best model introduced in this paper (rcNN$^{\text{split}}_{\text{short}}$) with baselines proposed for SWS in previous research. rcNN$^{\text{split}}_{\text{short}}$ outperforms the character based RNN described in Hellwig (2015b) by a wide margin. While Tab. 3 shows differences of almost 8% in sentence and 3% in string accuracy, Tab. 4 presents the improvements for the single surface character *ā*, which can correspond to a compound split (*ā-*) or to various vocalic Sandhis (*a-a* etc.). For this complicated character, rcNN$^{\text{split}}_{\text{short}}$ achieves consistent improvements of up to 15% on all metrics. We found it especially relevant to observe that rcNN$^{\text{split}}_{\text{short}}$ made large progress for rare rule types such as *ā-ā* or *ā-a*, indicating its increased ability for semantic generalization.

The seq2seq model (Reddy et al., 2018) performs on a similar level of accuracy as the one proposed in Hellwig (2015b). Similar to crNN (Sec. 5.2) it tends to miss splits and to insert faulty ones (e.g., *dānādānaratiḥ*, should: *dāna-ādānaratiḥ* "pleasure in giving and taking", is: *dānāt-ānaratiḥ* "from giving … UNK").

Gantayat et al. (2018) evaluate their model using location and split[9] prediction accuracy. The authors report 95.0 location and 79.5 split accuracy, but don't specify how they calculated these values. For this reason and because they evaluate on isolated strings only, we cannot compare directly against their work, but would like to report the following measures for rcNN$^{\text{split}}_{\text{short}}$:

- P, R and F for location prediction[10]: 97.64/98.19/97.91
- Micro-averaged P, R and F for individual rule types such as vocalic Sandhi or compound

---

[9] This seems to mean prediction of the correct Sandhi rule; see Gantayat et al. (2018, 4.2).

[10] $P = \frac{TP}{TP+FP}$, $R = \frac{TP}{TP+FN}$, TP: number of characters for which gold and model both record a split (though not necessarily of the same type), FP: number of characters at which the model over-segmentates, FN: number of character where the model fails to detect a valid split.

Model	S/A	P	R	A
Triṃśikāvijñaptibhāṣya				
rcNN$^{\text{split}}_{\text{short}}$	73.9	78.7	68.6	87.1
Transformer (5K)	72.6	77.5	66.9	86.8
Nyāyamañjarī				
rcNN$^{\text{split}}_{\text{short}}$	60.2	66.4	63.3	84.8
Transformer (5K)	62.1	68.6	63.5	85.2

Table 5: Results for evaluation on the Triṃśikāvijñaptibhāṣya and the Nyāyamañjarī; S/A: sentence accuracy

split: 95.12/95.12/95.12

The Transformer performs almost on par with rcNN$^{\text{split}}_{\text{short}}$, and the differences in string accuracy are not statistically significant, although rcNN$^{\text{split}}_{\text{short}}$ takes less time for training (2 h vs. 55 h) and inference (less than 1 min vs. 30 min when analyzing the test set). To better understand if the systems make orthogonal errors and could therefore be used in a mixture of experts, we performed a domain-specific evaluation with 73 sentences from the Buddhist treatise *Triṃśikāvijñaptibhāṣya* and 104 sentences from the philosophical text *Nyāyamañjarī*. We preserved the non-standard orthography of both texts in order to simulate the application of the models to real-world data. This includes the presence of typos, unsolved textual problems and erratic (non)-application of Sandhi.

Both models show a significant drop in overall performance when applied to these data (see Tab. 5). This is not surprising, because the input conventions of these files do not match the conventions of the training-data. Most errors arise again from disagreement about the (non-)compositional reading of technical compounds such as *sarvajña-tva* "all-knowing-ness" (see Sec. 3.1). It has to be noted that both models agree well in their correct decisions and in the type of errors they produce on these data. This indicates that the discrepancy in the orthographical conventions is indeed responsible for a large part of the drop in performance. Given the fact that both texts exhibit a lot of special vocabulary that is not present or used in a very different way in the training set, both models perform surprisingly well. Typical errors common to both models are for example *svalpam* instead of *su-alpam* "very small". Both models have difficulties to seperate Sandhi in passages that do not adhere to the common practice for typesetting of Indian texts in Latin transliteration. *ayaṃpariṇāmaḥ*, for example, was not separated into the usual form *ayaṃ pariṇāmaḥ*. There are certain cases of disagreement between both models that are noteworthy. While Transformer has changed the misspelled word *abhupagamyate* to the correct form *abhyupagamyate* in one case (overlooked by rcNN$^{\text{split}}_{\text{short}}$), rcNN$^{\text{split}}_{\text{short}}$ correctly identified the verbal form *upacaryante iti*, where Transformer inserted the semantically dispreferred, but grammatically possible present participle *upacaryantaḥ iti*. Overall, none of the models shows a generally better or worse performance in these cases of disagreement.

Model	P	R	A
Ma et al. (2016)	0.955	0.941	0.943
rcNN$_{\text{short}}$, no sp	0.958	0.958	0.955

Table 6: Results for splitting German compounds; evaluation metrics according to Koehn and Knight (2003)

5.4 Application to German Compounds

In order to test if the character based models generalize well to other languages with limited training resources, we applied rcNN$_{\text{short}}$ with split probabilities and the same settings as for SWS to the task of splitting German compounds. The current state of the art is set with a CRF operating on n-grams of characters (Ma et al., 2016). Table 6 shows that our model achieves an improvement of about 1% for recall and accuracy when trained with the training set of Ma et al. (2016) only. We sampled 20 examples for the three error classes "wrong split", "wrong faulty split" and "wrong non-split" (Ma et al., 2016, 78). While our model failed to detect splits for all 20 examples of the type "wrong non-split", the type "wrong split" contained 10 cases, where the split(s) proposed by the model make(s) good sense for us, but are not recorded in the test set (e.g. "Viermaster" 'four-master', "Viermaster" in test; already remarked by Ma et al. 2016). We observed a similar level of inconsistencies for the "wrong faulty split" type (8 instances), where, for example, our model analyzed "Bundes-tags-vize-präsident" 'vice president of the Federal Parliament', while the test set had "Bundes-tags-vizepräsident".

6 Conclusion

While the models discussed in this paper have produced clear performance gains when compared with previous research on SWS, we expect that future research will improve over our results, but it will be difficult to approach error-free performance. The reservation is due to the errors in the training data and especially the question of (non-)compositional readings of compounds, which seems to produce related levels of confusion for human annotators and ML models. While following this track of research, we would like to expand its scope to joint learning of splits, lexical and morphological annotations. Here, we expect that especially lexical and morphological analysis will benefit from a joint model. We hypothesize that CTC (Graves, 2012) trained as a co-task or segmental NNs (Lu et al., 2016) with a modified objective (including split probabilities) may be suitable for this task.

References

Theodor Aufrecht. 1891–1903. *Catalogus Catalogorum. An Alphabetical Register of Sanskrit Works and Authors*. Otto Harrassowitz, Leipzig.

Shubham Bhardwaj, Neelamadhav Gantayat, Rahul Garg, and Sumeet Agarwal. 2018. SandhiKosh: A benchmark corpus for evaluating Sanskrit Sandhi tools. In *Proceedings of the LREC*.

Christopher M. Bishop. 2000. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. *arXiv preprint arXiv:1609.07053*.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for Chinese. In *Proceedings of the 54th Annual Meeting of the ACL*, volume 1, pages 409–420.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long Short-Term Memory neural networks for Chinese word segmentation. In *Proceedings of the 2015 Conference on EMNLP*, pages 1197–1206.

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

Harry Falk. 1993. *Schrift im alten Indien: Ein Forschungsbericht mit Anmerkungen*. Gunter Narr Verlag, Tübingen.

N. Gantayat, R. Aralikatte, N. Panwar, A. Sankaran, and S. Mani. 2018. Sanskrit sandhi splitting using $seq2(seq)^2$. *ArXiv e-prints*.

Pawan Goyal, Vipul Arora, and Laxmidhar Behera. 2009. Analysis of Sanskrit text: Parsing and semantic relations. In *Sanskrit Computational Linguistics*, pages 200–218. Springer.

Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for Sanskrit corpus annotation. *Journal of Language Modelling*, 4(2):145–182.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Verlag, Heidelberg.

Choochart Haruechaiyasak, Sarawoot Kongyoung, and Matthew Dailey. 2008. A comparative study on Thai word segmentation approaches. In *Proceedings of ECTI-CON*, volume 1, pages 125–128.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778.

Oliver Hellwig. 2009. SanskritTagger, a stochastic lexical and POS tagger for Sanskrit. In *Sanskrit Computational Linguistics. First and Second International Symposia*, Lecture Notes in Artificial Intelligence, 5402, pages 266–277, Berlin. Springer Verlag.

Oliver Hellwig. 2015a. Morphological disambiguation of Classical Sanskrit. In *Systems and Frameworks for Computational Morphology*, pages 41–59, Cham. Springer.

Oliver Hellwig. 2015b. Using Recurrent Neural Networks for joint compound splitting and Sandhi resolution in Sanskrit. In *Proceedings of the 7th LTC*, pages 289–293.

Gérard Huet. 2005. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *Journal of Functional Programming*, 15(04):573–614.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on EMNLP*, pages 1746–1751.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the ICLR*.

Yoshiaki Kitagawa and Mamoru Komachi. 2017. Long Short-Term Memory for Japanese word segmentation. *ArXiv e-prints*.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the Tenth Conference of EACL, Volume 1*, pages 187–193.

Amrith Krishna, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu, Yajuvendra Singh, and Pawan Goyal. 2016. Word segmentation in Sanskrit using path constrained random walks. In *Proceedings of the COLING*, pages 494–504.

Amrith Krishna, Pavan Kumar Satuluri, and Pawan Goyal. 2017. A dataset for Sanskrit word segmentation. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 105–114.

Amba Kulkarni and Devanand Shukla. 2009. Sanskrit morphological analyser: Some issues. *Indian Linguistics*, 70(1-4):169–177.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

John J. Lowe. 2015. The syntax of Sanskrit compounds. *Language*, 91(3):71–115.

Liang Lu, Lingpeng Kong, Chris Dyer, Noah A Smith, and Steve Renals. 2016. Segmental recurrent neural networks for end-to-end speech recognition. *arXiv preprint arXiv:1603.00223*.

Jianqiang Ma, Verena Henrich, and Erhard Hinrichs. 2016. Letter sequence labeling for compound splitting. In *Proceedings of the 14th SIGMORPHON Workshop*, pages 76–81.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the ACL*, volume 1, pages 1064–1074.

Klaus Macherey, Andrew M Dai, David Talbot, Ashok C. Popat, and Franz Och. 2011. Language-independent compound splitting with morphological operations. In *Proceedings of the 49th Annual Meeting of the ACL*, pages 1395–1404.

Vipul Mittal. 2010. Automatic Sanskrit segmentizer using finite state transducers. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 85–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve Restricted Boltzmann Machines. In *Proceedings of the 27th ICML*, pages 807–814.

Abhiram Natarajan and Eugene Charniak. 2011. Sˆ3-statistical Sandhi splitting. In *IJCNLP*, pages 301–308.

Vikas Reddy, Amrith Krishna, Vishnu Dutt Sharma, Prateek Gupta, M R Vineeth, and Pawan Goyal. 2018. Building a word segmenter for Sanskrit overnight. In *Proceedings of the LREC*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. In *Proceedings of the ICASSP*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Richard Sproat and Thomas Emerson. 2003. The first international Chinese word segmentation bakeoff. In *Proceedings of the second SIGHAN Workshop on Chinese Language Processing*, pages 133–143.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. In *Proceedings of the Deep Learning Workshop at ICML*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

William Dwight Whitney. 1879. *A Sanskrit Grammar*. Breitkopf and Härtel, Leipzig.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Session-level Language Modeling for Conversational Speech

Wayne Xiong **Lingfeng Wu** **Jun Zhang**
Microsoft, Bellevue, WA, USA
{weixi,lingfw,junzh}@microsoft.com

Andreas Stolcke
Microsoft, Sunnyvale, CA, USA
anstolck@microsoft.com

Abstract

We propose to generalize language models for conversational speech recognition to allow them to operate across utterance boundaries and speaker changes, thereby capturing conversation-level phenomena such as adjacency pairs, lexical entrainment, and topical coherence. The model consists of a long-short-term memory (LSTM) recurrent network that reads the entire word-level history of a conversation, as well as information about turn taking and speaker overlap, in order to predict each next word. The model is applied in a rescoring framework, where the word history prior to the current utterance is approximated with preliminary recognition results. In experiments in the conversational telephone speech domain (Switchboard) we find that such a model gives substantial perplexity reductions over a standard LSTM-LM with utterance scope, as well as improvements in word error rate.

1 Introduction

Over the past decade the state of the art in language modeling has shifted from N-gram models to feed-forward networks (Bengio et al., 2006), and then to recurrent neural networks (RNNs) that read a list of words sequentially and predict the next word at each position. Starting with standard recurrent networks (Mikolov et al., 2010) the sequential modeling approach was later improved using the long-short-term memory (LSTM) architecture of (Hochreiter and Schmidhuber, 1997) for further gains (Sundermeyer et al., 2012; Medennikov et al., 2016; Xiong et al., 2017). RNN models give two fundamental advantages over the old N-gram framework. First, the continuous-space embedding of word identities allows word similarities to be exploited for generalization (Bengio et al., 2006; Mikolov et al., 2013). Second, the recurrent architecture allows, in principle at least, an unlimited history to condition the prediction of next words.

The potential advantage of unlimited history, however, is not commonly used to its full benefit, since the language model (LM) is typically “reset” at the start of each utterance in current state-of-the-art recognition systems (Saon et al., 2017; Xiong et al., 2018). This presumes that each utterance is independent of the others, and clearly violates what we know about how language and conversation works, as discussed in the next section. Consequently, there have been many proposals to inject information from a longer context into standard LM architectures, going back to N-gram models (Bellegarda, 2004), or to generalize N-grams LMs to operate across utterance boundaries and speakers (Ji and Bilmes, 2004). Based on the RNN framework, (Mikolov and Zweig, 2012) proposed augmenting network inputs with a more slowly varying context vector that would encode longer-range properties of the history, such as a latent semantic indexing vector. The problem with these approaches is that the modeler has to make design decisions about how to encapsulate contextual information as network inputs. Therefore, our approach here is to simply provide the entire conversation history as input to a standard LSTM-LM, and let the network learn the information that is relevant to next-word prediction.

We start by discussing linguistic phenomena that could potentially help in conversational LM (Section 2), followed by a description of the LSTM model we propose to capture them (Section 3). Section 4 describes the data and recognition system we used to test our models, with results reported in Section 5. We end with conclusions and future directions.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2764–2768
Brussels, Belgium, October 31 - November 4, 2018.

2 Conversation-level Phenomena

Here we review a few of the conversation-level phenomena that could be used for predicting words from longer context. Perhaps the most widely studied effect is topical coherence, or the tendency of words that are semantically related to one or more underlying topics to appear together in the conversation. Consequently, topic-related words are bound to re-occur across utterances, or certain related words appear to trigger one another (such as “children” and “school”). This should be especially true for conversations in the Switchboard (and Fischer) corpora, which were collected by pairing up strangers to talk about a mutually agreeable topic.

Another phenomenon that could lead to words reoccurring is lexical entrainment (Brennan and Clark, 1996), or the tendency of conversants to adopt the same words and phrases. Entrainment can also apply to speaking style, so the use of common discourse particles, syntactic patterns (like question tags), or even disfluencies could be triggered across speakers.

Other phenomena operate more locally, but across speaker turn boundaries. Linguistic conversation analysis has long noted that utterance types come in *adjacency pairs* (Schegloff, 1968), with preferences for certain pairs over others (like a statement is preferentially followed by agreement rather than disagreement). Therefore, words in an utterance should be more predicable based on the previous utterance. In the past, this has been modeled by conditioning utterance words on an underlying dialog act label, which in turn is conditioned on adjacent dialog act labels via a dialog act grammar (Stolcke et al., 2000).

A good part of conversational behavior has to do with how turn-taking is negotiated (Sacks et al., 1974). Speakers use special discourse devices, such as backchannel words and pause fillers, to signal when they want to take the floor, or to signal that the other party should keep the floor. Conversants also anticipate the ends of turns and jump in before the other speaker is completely done, making for very efficient use of time. As a result of all of these mechanisms, a good portion of conversations consists of overlapping (simultaneous) speaking. It was shown (Shriberg et al., 2001) that such overlap locations can be partly predicted by word-based language models. This suggests reversing the modeling and using overlap (the timing of utterances) to help predict the words.

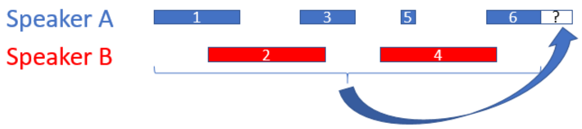

Figure 1: Use of conversation-level context in session-based LM. The utterance numbering shows how overlapping utterances are serialized (according to onset times).

3 Models

Our baseline language model is a standard LSTM that models utterances independently from one another, i.e., the history at the onset of each utterance is the start-of-sentence token. In fact, we used two version of this basic LSTM-LM:

- Word inputs encoded with one-hot vectors, combined with a jointly trained embedding layer
- Words encoded by multiple-hot vectors corresponding to the letter trigrams making up the words.

Both types of LSTM-LMs use three 1000-dimensional hidden layers with recurrence. The word embedding layer is also of size 1000, and the letter-trigram encoding has size 7190 (the number of unique trigrams in our vocabulary).

The main addition for session-level modeling is that the LSTM history consists of all the utterances preceding the current utterance, followed by all words in the current utterance preceding the word to be predicted. The preceding utterances are serialized in the order of their onset times, so that the flow of words within an utterance is not disrupted. The resulting total word history and next-word prediction is depicted in Figure 1. Information about utterance boundaries is encoded using a boundary tag, similar to the start-of-sentence token that is commonly used in LMs.

Several of the conversational phenomena described in Section 2 refer to turn-taking between speakers; to capture this in the model we augment the word input encoding with an extra bit that indicates whether a speaker change occurred. This bit is turned on only for the start-of-utterance token.

We also want to capture some information about utterance overlap, since, as described earlier, speech overlap interacts with word choice. Possible events to model would be overlap (exceedings a time threshold) at the starts and ends of utterances, or maybe a continuous measure of such overlaps. As a first proof of concept we chose to encode only one type of overlap, i.e., when the utterance in question is completely overlapped temporally by the other speaker's turn. This is typical of backchannel acknowledgments ("uh-huh") and short utterances that attempt to grab the floor ("um", "but"). Complete utterance overlap is also encoded by an additional input bit that is turned on for the start-of-utterance token.

4 Experiments

4.1 Recognition system

We used a single bidirectional LSTM acoustic model in experiments reported here, trained on the commonly used conversational telephone speech corpora (Switchboard, Fisher, CallHome English), estimating frame-level posterior probabilities for 9000 context-dependent phone units. The system decodes speech utterances using a 4-gram language model, generating lattices. These are then expanded to 500-best lists, which in turn are rescored using the various LMs.

The recognition system and the N-gram LM used in decoding have a vocabulary of 165k words, but the LSTM-LMs are trained on only the 38k words occurring at least twice in the in-domain conversational training data. Words outside of the LSTM-LM vocabulary are penalized in rescoring with a constant weight that is empirically optimized on the development set.

4.2 Data

Language model training uses the Switchboard-1, BBN Switchboard-2, Fisher, and English CallHome transcripts (about 23 million words in total) as well as the UW conversational Web corpus (Bulyko et al., 2003) for pre-training (see below). The N-gram LM used for N-best generation also includes the LDC Hub4 (Broadcast News) corpus. The Switchboard-1 and Switchboard-2 portions of the NIST 2002 CTS test set were used for tuning and development. Evaluation is carried out on the NIST 2000 CTS test set, consisting of Switchboard (SWB) and CallHome (CH) subsets.

As an expedient, we refrained from resegmenting utterances based on forced alignments of words, and instead use utterance boundaries as given in the available transcripts (corresponding to the audio segments used in acoustic training). Similarly, in testing, we use the presegmented utterances provided by NIST. No doubt there are inconsistencies in how the different corpora define utterance units, and a consistent, alignment-based resegmentation of all training and test data based on the durations nonspeech regions and/or lexical tagging might give improved results.

Table 1: Perplexities with session-based LSTM-LMs. The last two lines reflect use of errorful recognition output for preceding utterances.

Model inputs	devset SWB	test SWB	test CH
Utterance words, letter-3grams	48.90	44.56	54.57
+ session history words	38.86	36.81	44.31
+ speaker change	37.25	35.33	42.23
+ speaker overlap	37.09	35.12	42.02
Using recognized word histories			
single system	39.55	37.45	46.49
full system (Xiong et al., 2018)	39.41	37.29	45.99

4.3 Model training

All LSTM-LMs are trained using the Microsoft Cognitive Toolkit, or CNTK (Yu et al., 2014; Microsoft Research, 2016) on a Linux-based multi-GPU server farm. Training is parallelized using CNTK's distributed stochastic gradient descent (SGD) with 1-bit gradient quantization (Seide et al., 2014). We use the CNTK "FsAdaGrad" learning algorithm, which is an implementation of Adam (Kingma and Ba, 2015).

All LSTM-LMs are pretrained for one or two epochs on a large corpus of "conversational Web" data (Bulyko et al., 2003), followed by normal training to convergence on the in-domain data. Each utterance in the Web data is treated as a single session for purposes of session-based LM, i.e., the extra bits for speaker change and overlap are never turned on.

5 Results

When evaluating the session-based LMs on speech test data, the true utterance contexts are not known, and we must use hypothesized words for word histories preceding the current utterance. In our case, the histories were obtained using the output of our best recognition system, which uses a combination of acoustic models (Xiong et al., 2018), but excluding the session-based LM.[1] Per-

---

[1] We also omitted the final confusion network rescoring stage described in (Xiong et al., 2018).

Table 2: Recognition results with standard and session-based LSTM-LMs, measured by word error rates (WER).

Word encoding	Model	WER devset	WER test SWB	CH
Letter 3gram	LSTM-LM	10.01	6.88	12.79
	Session LSTM-LM	9.67	6.81	12.54
	Session LSTM-LM, 2nd iteration	9.66	6.77	12.56
One-hot	LSTM-LM	9.81	6.89	13.02
	Session LSTM-LM	9.47	6.81	12.60
	Session LSTM-LM, 2nd iteration	9.50	6.83	12.73
Letter 3gram + One-hot	LSTM-LM	9.66	6.63	12.77
	Session LSTM-LM	9.28	6.52	12.34
	LSTM-LM + Session LSTM-LM	9.22	6.45	12.11

plexity was evaluated on reference transcripts, as is customary.

Table 1 shows the effect of session-level modeling and of optional model elements on perplexity, based on LSTMs using letter-trigram encoding. Baseline is the standard utterance-scope LSTM-LM. We see a large perplexity reduction of 17-21% by conditioning on session history words, with smaller incremental reductions from adding speaker change and overlap information.

The last two table rows show that some of the perplexity gain over the baseline is negated by the use of errorful recognition output for the conversation history. It does not make much difference whether the recognized word history is generated by just the subsystem being rescored ("single system", with 6% word error on SWB) or the full recognition system using multiple acoustic models ("full system", with about 5% word error rate on SWB and 10% on CH). Using recognition output as history, the perplexity degrades about 6% relative for SWB, and 11% on CH, relative to using the true word histories. Even with the more errorful recognition on CH, the session-based LM still gives a perplexity reduction of 14% relative to the baseline.

Table 2 presents recognition results, comparing baseline LSTM-LMs to the full session-based LSTM-LMs. Both the letter-trigram and one-word word encoding versions are reported. The different models may also be used jointly, using log-linear score combination in rescoring, shown in the third section of the table. We also tried iterating the session LM rescoring, after the recognized word histories were updated from the first rescoring pass (shown as "2nd iteration" in the table).

Results show that the session-based LM yields between 1% and 4% relative word error reduction for the two word encodings, and test sets. When the two word encoding types are combined by log-linear combination of model scores, the gain from session-based modeling is preserved. Iterating the session LM rescoring to improve the word histories did not give consistent gains.

Even though the session-based LSTM subsumes all the information used in the standard LSTM, there is an additional gain to be had from combining those two model types (last row in the table). Thus, the overall gain from adding the session-based models to the two baseline models is 3-5% relative word error reduction.

6 Conclusion and Future Work

We have proposed a simple generalization of utterance-level LSTM language models aimed at capturing conversational phenomena that operate across utterances and speakers, such as lexical entrainment, adjacency pairs, speech overlap, and topical coherence. To capture non-local conditioning information, the LSTM-LM is trained to read the entire sequence of utterances making up a conversation, along with side information encoding speaker changes and overlap of utterances. This is found to reduce perplexity by about 25%, most of which is retained when errorful recognition output is used to represent the word history in previous utterances. The session-based LM yields up to 5% relative reduction in word error when the utterance- and session-based LMs are combined.

It would be worthwhile to investigate which conversational phenomena are actually being exploited by the session LSTM model. The ease with which additional information can be input to the LSTM-LM also suggests encoding other conditioning information, such a more details about utterance timing, as well as semantic features that capture topical coherence.

References

Jerome R. Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42:93–108.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Studies in Fuzziness and Soft Computing*, volume 194, pages 137–186.

Susan E. Brennan and Herbert H. Clark. 1996. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22(6):1482–1493.

Ivan Bulyko, Mari Ostendorf, and Andreas Stolcke. 2003. Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures. In *Proceedings of HLT-NAACL 2003, Conference of the North American Chapter of the Association of Computational Linguistics*, volume 2, pages 7–9, Edmonton, Alberta, Canada. Association for Computational Linguistics.

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

Gang Ji and Jeffrey Bilmes. 2004. Multi-speaker language modeling. In *Proceedings of HLT-NAACL 2004, Conference of the North American Chapter of the Association of Computational Linguistics*, volume Short Papers, pages 133–136, Boston. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. Proceedings 3rd International Conference for Learning Representations, arXiv preprint arXiv:1703.02136.

Ivan Medennikov, Alexey Prudnikov, and Alexander Zatvornitskiy. 2016. Improving English conversational telephone speech recognition. In *Proc. Interspeech*, pages 2–6.

Microsoft Research. 2016. The Microsoft Cognition Toolkit (CNTK). https://cntk.ai.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. Interspeech*, pages 1045–1048.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, volume 13, pages 746–751.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Proc. Interspeech*, pages 901–904.

H. Sacks, E. A. Schegloff, and G. Jefferson. 1974. A simplest semantics for the organization of turn-taking in conversation. *Language*, 50(4):696–735.

George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, Bergul Roomi, and Phil Hall. 2017. English conversational telephone speech recognition by humans and machines. In *Proc. Interspeech*, pages 132–136, Stockholm.

E. A. Schegloff. 1968. Sequencing in conversational openings. *American Anthropologist*, 70:1075–1095.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 2014. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Proc. Interspeech*, pages 1058–1062.

Elizabeth Shriberg, Andreas Stolcke, and Don Baron. 2001. Observations on overlap: Findings and implications for automatic processing of multi-party conversation. In *Proceedings of the 7th European Conference on Speech Communication and Technology*, volume 2, pages 1359–1362, Aalborg, Denmark.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Proc. Interspeech*, pages 194–197.

W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke. 2018. The Microsoft 2017 conversational speech recognition system. In *Proc. IEEE ICASSP*, pages 5934–5938, Calgary, Alberta.

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. 2017. Toward human parity in conversational speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 25(12):2410–2423.

D. Yu et al. 2014. An introduction to computational networks and the Computational Network Toolkit. Technical Report MSR-TR-2014-112, Microsoft Research. Https://github.com/Microsoft/CNTK.

Towards Less Generic Responses in Neural Conversation Models: A Statistical Re-weighting Method

Yahui Liu[1]*, Victoria Bi[2]†, Jun Gao[3], Xiaojiang Liu[2], Jian Yao[1], Shuming Shi[2]
[1]Wuhan University, Wuhan, China
[2]Tencent AI Lab, Shenzhen, China
[3]Soochow University, Suzhou, China
{liuyahui, jian.yao}@whu.edu.cn, imgaojun@gmail.com,
{victoriabi, kieranliu, shumingshi}@tencent.com

Abstract

Sequence-to-sequence neural generation models have achieved promising performance on short text conversation tasks. However, they tend to generate generic/dull responses, leading to unsatisfying dialogue experience. We observe that in conversation tasks, each query could have multiple responses, which forms a *1-to-n* or *m-to-n* relationship in the view of the total corpus. The objective function used in standard sequence-to-sequence models will be dominated by loss terms with generic patterns. Inspired by this observation, we introduce a statistical re-weighting method that assigns different weights for the multiple responses of the same query, and trains the standard neural generation model with the weights. Experimental results on a large Chinese dialogue corpus show that our method improves the acceptance rate of generated responses compared with several baseline models and significantly reduces the number of generated generic responses.

1 Introduction

Many recent works have been proposed to use neural networks to generate responses for open-domain dialogue systems (Shang et al., 2015; Sordoni et al., 2015; Vinyals and Le, 2015; Li et al., 2016a,c; Serban et al., 2017; Shen et al., 2017; Li et al., 2017; Yu et al., 2017; Xu et al., 2017). These methods are inspired by the sequence-to-sequence (Seq2Seq) framework (Sutskever et al., 2014), which is originally applied for Neural Machine Translation (NMT). They aim at maximizing the probability of generating a response given an input query, and generally use the *maximum likelihood estimation* (MLE) as their objective function. However, various problems occur when Seq2Seq models are used for dialogue generation tasks. One of the most important problems is that such models are inclined to generate generic and dull responses (e.g., I don't know), rather than meaningful and specific answers (Sordoni et al., 2015; Serban et al., 2016; Li et al., 2016a,c; Kannan et al., 2016; Li et al., 2017; Xie, 2017; Wei et al., 2017; Mou et al., 2017).

Until now, it has attracted increasing studies to address the issue of generating generic response. For example, Li et al. (2016a) used the mutual information theory to reconstruct MLE, but this model is easy to generate ungrammatical outputs. They further proposed a fast diverse decoding approach (Li et al., 2016b), which modifies the beam search to re-rank meaningful responses into higher positions. Similar works explore different ways to encourage response diversity for picking less generic responses in the decoding search (Vijayakumar et al., 2016; Li and Jurafsky, 2016). In the reinforcement learning framework (Li et al., 2016c), the reward function used in the decoding considers the ease of answering, which is measured by a distance towards a set of 8 generic responses. Thus, it can also alleviate the problem of generating generic responses to some extent. Lison and Bibauw (2017) proposed to add a weighting model to learn the "quality" of the query and response pair, but it relies heavily on additional inputs. All these works tried to add extra optimized terms in the encoding or decoding modules in Seq2Seq, making the training or prediction more complicated.

In this work, we consider the reason why Seq2Seq often generates generic responses by analyzing the MLE objective function directly. We notice that multiple responses are often associated with one single input query. As shown in Figure 1, the relationship between queries and responses is much looser in conversation models than that in NMT, since the space of possible responses is much

---

[*] This work was done while Yahui Liu was with Tencent AI Lab.

[†] Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2769–2774
Brussels, Belgium, October 31 - November 4, 2018.

larger than the space of possible translations for a given sentence. On one hand, the information of these responses is only required to be relevant to the input query but usually differs from it. On the other hand, a query accepts large semantic diversity among its responses. Hence, it is a *1*-to-*n* relationship between a query and its responses (Vinyals and Le, 2015; Zhou et al., 2017). Meanwhile, we can see there is a *m*-to-*n* relationship between all queries and responses in the training corpus. Then, we find that MLE, which learns a *1*-to-*1* mapping in response generation, naturally puts more emphasis on optimizing the frequent patterns. Thus, the converged local optimum is easy to output these patterns or their combinations, leading to generic responses.

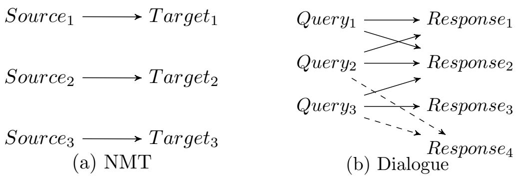

(a) NMT (b) Dialogue

Figure 1: An illustration of the differences between NMT and dialogue generation. *Response*$_4$ is the potential cases that are not collected in corpus.

Inspired by this observation, we propose a statistical re-weighting method which modifies MLE by re-weighting the multiple responses for each query such that MLE will not be dominated by the frequent patterns or their combinations. The proposed method calculates the weights of a response with the consideration of two statistical features: similarity frequency and sentence length. Our model is simple and efficient to optimize without adding additional terms into the original Seq2Seq objective function. We validate the performance of our proposed method on a large Chinese dialogue corpus. Results show that it can improve the acceptance rate of the generated responses and significantly suppress the number of generic responses.

2 Proposed Method

Standard Seq2Seq models for NMT and dialogue generation aim at estimating the conditional probability $p(\mathbf{y}|\mathbf{x})$ where $\mathbf{x} = (x_1, \ldots, x_T)$ is an input sequence and $\mathbf{y} = (y_1, \ldots, y_{T'})$ is its corresponding output sequence whose length $T^{'}$ may differ from T. During training, we learn all the model parameters $\boldsymbol{\theta}$ by summing the negative log likelihood of each sample pair $(\mathbf{x}, \mathbf{y})$ in the training corpus $\mathbb{C}$:

$$\begin{aligned} \ell(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) &= -\sum_{t=1}^{T'} \log p(y_t|\mathbf{x}, \mathbf{y}_{[t-1]}; \boldsymbol{\theta}), && (1) \\ \mathcal{L}(\mathbb{C}, \boldsymbol{\theta}) &= \sum_{(\mathbf{x},\mathbf{y})\in\mathbb{C}} \ell(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}). && (2) \end{aligned}$$

Recall that generic responses are those that are safe and universal for many queries and thus frequently appear in the training corpus. Hence, if we have two responses of $\mathbf{x}$ in which one is generic and the other one contains more meaningful content, using $\mathcal{L}(\mathbb{C}, \boldsymbol{\theta})$ in Eq. 1 will put the same emphasis on optimizing each of their loss terms. Therefore, $\mathcal{L}(\mathbb{C}, \boldsymbol{\theta})$ contains a large amount of patterns from the generic responses, thus it is not surprised to see that the trained models are stuck into local optimum that are inclined to generate these patterns or their combinations.

Based on this observation, we argue that a good loss function of Seq2Seq for dialogue generation should not be dominated by the patterns from generic responses. Here, we propose a re-weighting method for responses of a query $\mathbf{x}$. Specifically, $\ell(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})$ in Eq. 1 is modified to be:

$$\ell_w(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = w(\mathbf{y}|\mathbf{x})\ell(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}), \qquad (3)$$

where $w(\mathbf{y}|\mathbf{x}) \in (0, 1]$ is a soft weight for a response $\mathbf{y}$ of a query $\mathbf{x}$. In the implementation, we make the normalization of this loss at the mini-batch level for better computational efficiency. Hence, the loss of Eq. 2 for a mini-batch $\mathcal{L}(\mathbb{B}, \boldsymbol{\theta})$ takes the form:

$$\mathcal{L}(\mathbb{B}, \boldsymbol{\theta}) = \frac{\sum_{\mathbf{x},\mathbf{y}\in\mathbb{B}} \ell_w(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})}{\sum_{\mathbf{x},\mathbf{y}\in\mathbb{B}} w(\mathbf{y}|\mathbf{x})}. \qquad (4)$$

We summarize two common properties for the responses:

- Responses with the patterns of frequently appearing in the training corpus tend to be generic. Here, the patterns refer to both the whole sentence or n-grams which can be described by similarities among responses.
- Very short and long responses should be avoid. Owing to the MLE objective function, the Seq2Seq frameworks are inclined to generate short responses that are universal replies. While long responses usually contain more specific information which may not be generalized to most conversation scenarios. Hence,

high-quality responses tend to be with moderate length.

We propose an estimator by considering these two properties:

$$w(\mathbf{y}|\mathbf{x}, \mathbb{R}, \mathbb{C}) = \frac{\Phi(\mathbf{y})}{\max_{\mathbf{r}\in\mathbb{R}}\{\Phi(\mathbf{r})\}}, \tag{5}$$

where $\mathbb{R}$ denotes all collected responses of $\mathbf{x}$ in $\mathbb{C}$. For each response, the estimator gives a weight by:

$$\Phi(\mathbf{y}) = \alpha\mathcal{E}(\mathbf{y}) + \beta\mathcal{F}(\mathbf{y}). \tag{6}$$

Here, $\mathcal{E}(\mathbf{y})$ and $\mathcal{F}(\mathbf{y})$ correspond to the mentioned two properties respectively:

- $\mathcal{E}(\mathbf{y}) = e^{-af(\mathbf{y})}$, where $f(\mathbf{y})$ is a function related to the frequency of response $\mathbf{y}$. It could be formulated as

  $$f(\mathbf{y}) = \max\{0, \text{Count}(D(\mathbf{y}, \mathbf{y}_j) \geq \tau) - b\} \quad \forall j \in |\mathbb{C}|,$$

  where $D(\cdot)$ refers to the similarity between two sentences, a is a scale factor, b is bias and $\tau \in [0, 1]$ is a threshold specifying the similarity that two responses will be considered identical. For instance, it could be the simplest strictly matching, which is used in our experiments. Other methods like cosine distance of TF-IDF (token or n-grams) can also be applied, but may encounter computational issues for large corpus. A response with a higher frequency will be assigned with a smaller $\mathcal{E}(\mathbf{y})$.

- $\mathcal{F}(\mathbf{y}) = e^{-c||\mathbf{y}|-|\hat{\mathbf{y}}||}$, where $|\mathbf{y}|$ denotes the number of tokens in $\mathbf{y}$, $|\hat{\mathbf{y}}| = \frac{1}{|\mathbb{C}|}\sum_{\mathbf{r}\in\mathbb{C}}|\mathbf{r}|$ refers to the average length of responses in the total training corpus, and c is a scale factor. Here, the "moderate length" is set to the average length of responses of the total training corpus. In practice, we have tried to use long responses (longer than average length) to fine-tune the Seq2Seq model. Though it slightly increases the average length of generated responses, the generated responses suffer from more ungrammatical and influent issues. Hence, if a response is too short or long, it will receive a low score of $\mathcal{F}(\mathbf{y})$.

Mentioned hyper-parameters $\{\alpha, \beta, a, b, \tau, c\}$ are constant values in the following experiments, which are set to $\{0.5, 0.5, 0.33, 3, 1.0, 0.33\}$. When we performed our experiments, we tried several hyper-parameter settings and found that our method is not sensitive to different hyper-parameters and achieves stable results in general. Hence, we do not spend many efforts to specifically tune these hyper-parameters.

Response (frequency/length)	$w(\mathbf{y}\|\mathbf{x})$	$\mathcal{E}(\mathbf{y})$	$\mathcal{F}(\mathbf{y})$
一直单身 (70/2) Single till now.	0.047	0.014	0.066
求交往 (173/2) Would you like to have a date with me?	0.039	0.000	0.066
等你看到别人甜蜜就知道一个人的痛苦了 (2/13) Once seeing the happy lovers, you will feel sad.	0.769	1.000	0.560
应该还在哭吧 (1/5) You may be crying now.	0.665	1.000	0.176
单身狗的自我安慰 (2/4) Morale-boosting of singles.	0.622	1.000	0.128
我也是这么觉得 (30/6) I think so.	0.149	0.052	0.248
单身不好,单身23年,孤单太久了 (1/10) Given being single for 23 years, it's not so good.	1.000	1.000	0.924
多么心塞的领悟 (1/4) What a painful understanding!	0.623	1.000	0.128

Table 1: Weights of the responses for a query "其实单身也挺好的 (It's pretty good to be single)".

To validate that our design function in Eq. 5 and Eq. 6 are effective to weight the responses, Table 1 shows the weights of 8 responses for a query "其实单身也挺好的 (It's pretty good to be single)". As can be seen, the weights are reasonable, in which the higher-ranked responses are more informative ones with low similarity frequency and moderate length.

3 Experiments

3.1 Corpus and Evaluation

We crawl conversation pairs from some popular Chinese social media websites[1], and select 7M high-quality pairs as our training corpus. Conventional metrics such as BLEU (Papineni et al., 2002) and perplexity, are improper to be used for response generation tasks. Following previous works (Li et al., 2016c, 2017), we apply human annotations. We randomly sample 500 queries (not used in training) as our test samples, and recruit 3 annotators to evaluate each generated response from two aspects:

- *Fluency*: 0 (unreadable), 1 (readable but with some grammar mistakes), 2 (fluent);
- *Relevance*: 0 (not relevant at all), 1 (relevant at a distant level), 2 (relevant, including the

---

[1] Weibo: `www.weibo.com`, Baidu Tieba: `tieba.baidu.com`, and Zhihu: `www.zhihu.com`

generic responses), 3 (relevant as well as interesting).

Acceptance is then automatically calculated as a metric reflecting whether the response is acceptable to real users. A response will be assigned 1 when it gets *Fluency*$\geq$1 and *Relevance*$\geq$2, otherwise it will be assigned 0.

We implement our baseline Seq2Seq model using its standard objective function in Eq. 1 with two LSTM layers for encoding/decoding and a standard beam search with a beam size of 5 (the best setting), termed as *Seq2Seq*. We also compare several Seq2Seq variants:

- *Seq2Seq-RS*: training with a subset by randomly sampling only one from the multiple responses for each query;
- *Seq2Seq-MMI*: applying the maximum mutual information (Li et al., 2016a) (only the MMI-bidi);
- *Seq2Seq-DD*: applying the diverse decoding algorithm (Li et al., 2016b);
- *Ours-RW*: calculating weights via our reweighting method proposed in Section 2. Without applying any other tricks, we implement three versions of our method by using $\mathcal{E}(\cdot)$ only, $\mathcal{F}(\cdot)$ only, a linear combination of $\mathcal{E}(\cdot)$ and $\mathcal{F}(\cdot)$ in Eq 6, termed as *Ours-RW*$_{\{\text{E,F,EF}\}}$.

3.2 Results and Discussion

Human annotation results are shown in Table 2. Several observations can be made. First, *Seq2Seq-RS* performs slightly worse than the baseline model. This means that it does not work to simply discard a large amount of training data to construct a 1-to-1 query-response subset for training. Second, *Seq2Seq-MMI* not only provides no improvement for the baseline but also inclines to generate generic response. Third, *Seq2Seq-DD* obtains higher relevance and acceptance scores than the baseline, which shows its effectiveness by re-ranking more meaningful responses into higher positions in beam search. Fourth, our method achieves the best performance on almost all metrics. When we use strictly matched frequency of each response, *Ours-RW*$_{\text{E}}$ does not perform better than the baseline model because that the percentage of responses with frequency higher than 3 is about 0.5% in our training corpus. However, it still enhances the performance in *Ours-RW*$_{\text{EF}}$, which performs the best and increases the acceptance of the baseline model from 0.42 to 0.55. This validates that the properties about similarity frequency and sentence length play important roles in generating better responses.

Model	Evaluation Metrics		
	Fluency	Relevance	Acceptance
Seq2Seq	1.96±3.8e-5	1.31±5.3e-3	0.42±4.7e-4
Seq2Seq-RS	**1.97**±8.1e-5	1.30±2.1e-3	0.42±9.9e-4
Seq2Seq-MMI	1.94±7.2e-5	1.19±4.0e-3	0.41±1.9e-4
Seq2Seq-DD	1.86±2.8e-3	1.40±1.5e-2	0.49±2.4e-3
Ours-RW$_{\text{E}}$	1.95±1.9e-4	1.30±5.1e-3	0.42±4.2e-4
Ours-RW$_{\text{F}}$	**1.97**±1.5e-4	1.47±2.1e-3	0.51±6.3e-4
Ours-RW$_{\text{EF}}$	1.96±8.3e-5	**1.59**±1.9e-2	**0.55**±4.4e-3

Table 2: Human annotation results.

Specifically, the average percentage of the generated responses that are assigned to relevance rating 2 (relevant, including the generic responses) and 3 (relevant as well as interesting) are presented in Table 4. It shows that our method achieves higher relevance score owing to generating more high-quality responses with rating 3.

To validate that our method is effective to reduce the number of generated generic responses, we calculated the *distinct-1* and *distinct-2* (Li et al., 2016a) for the compared methods respectively, which are the number of distinct unigrams and bigrams divided by total number of generated words respectively. As shown in Table 5, *Ours-RW*$_{\text{EF}}$ achieves the best performance on the two metrics. This indicates that our model often outputs more meaningful and relevant responses than the other compared methods.

We further randomly sample another 100K queries (not used in training) and use the various models to generate responses. We compare the frequencies of several common generic responses appearing in the generated results, as shown in Table 6. It shows that our method can significantly reduce the number of generic responses. For instance, we reduce about 75% of the case “我也不知道 (I don’t know, either.)” and 77% of the case “我也想知道 (I want to know, too)” to be generated.

4 Conclusion

In this paper, we propose a statistical re-weighting method to weight multiple responses differently and optimize the MLE objective function. The weight of each response is calculated based on

Query	Response			
	Seq2Seq	Seq2Seq-MMI	Seq2Seq-DD	Ours-RW$_{EF}$
我要生气了 I'm going to be angry.	我也生气了 So do I.	生气了 I'm angry.	我也生气了 So do I.	不要生气 Don't be angry.
最近有什么好书推荐 Would you like to recommend me some good books?	看你喜欢什么类型的 It depends on what kind of books you like?	我也想去看电影 I want to see the movie, too.	你喜欢什么类型的 What kind of books do you like?	《红楼梦》 Dream of the Red Chamber
做人不一定要风风光光 It's not necessary to be a successful and grand man.	一定要 Must be	我也不知道 I don't know, either.	是啊。。。 Yes...	做人一定要善良 Everyone should try to be a kind man.
这几天嗓子疼 I have a sore throat recently.	我也不知道 I don't know, either.	我也想去 I want to go, too	我也是。。。 So do I...	我这两天也是 So do I these days .
二号煤矿放几天假 How many days did people in No.2 coal mine take off?	假的 False	我也想放假 I want to have a holiday, too	假的，假的 False, False	我放了3天 I took three days off.
孟记普洱茶专营店一帖 An advertising of Mengji's Pu'er tea shop.	我也想吃 I want to eat, too.	我也想喝 I want to drink, too.	铁观音，茶，茶，茶 Tat-Kuan-Yin Tea, tea, tea, tea	我喜欢喝茶 I like drinking tea.

Table 3: Examples of responses generated by Seq2Seq, Seq2Seq-MMI, Seq2Seq-DD and Ours-RW$_{EF}$.

Model	rating 2(%)	rating 3(%)
Seq2Seq	31.3	10.6
Seq2Seq-MMI	35.8	4.4
Seq2Seq-DD	**37.1**	11.7
Ours-RW$_{EF}$	36.8	**18.1**

Table 4: Comparisons on the average percentage of the generated responses that are assigned to relevance rating 2 and 3.

Model	distinct-1	distinct-2
Seq2Seq	0.170	0.307
Seq2Seq-MMI	0.140	0.259
Seq2Seq-DD	0.131	0.170
Ours-RW$_{EF}$	**0.173**	**0.359**

Table 5: Performances on the metrics *distinct-1* and *distinct-2*.

Response	Model			
	Seq2Seq	MMI	DD	RW$_{EF}$
我也不知道 I don't know, either.	3296	4404	941	847
我也想知道 I want to know, too.	3211	13764	795	738
我也觉得 I think so.	883	632	1505	59
我也想问 I want to ask, too.	515	51	441	12
不喜欢 I don't like it.	337	131	214	95
看什么 What do you look at?	254	52	377	87
不会 Will/can not	109	92	54	24

Table 6: Frequencies of several generic responses.

two terms according to the similarity frequency and its length. Experiments show that our approach improves the performance over the baseline models and reduces the number of generated generic responses significantly. It indicates that mismatching issue of objective function can be alleviated through such similar re-weighting methods, by which current encoder-decoder architectures can take full use of the *m*-to-*n* training corpus and model the dialogue generation tasks better.

References

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, et al. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. *The North American Chapter of the Association for Computational Linguistics (NAACL)*.

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Conference on Empirical Methods in Neural Language Processing (EMNLP)*.

Pierre Lison and Serge Bibauw. 2017. Not all dialogues are created equal: instance weighing for neural conversational models. In *Conference on Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2017. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. *arXiv preprint arXiv:1705.00316*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Bolin Wei, Shuai Lu, Lili Mou, Hao Zhou, Pascal Poupart, Ge Li, and Zhi Jin. 2017. Why do neural dialog systems generate short and meaningless replies? a comparison between dialog and translation. *arXiv preprint arXiv:1712.02250*.

Ziang Xie. 2017. Neural text generation: A practical guide. *arXiv preprint arXiv:1711.09534*.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, Xiaolong Wang, Zhouran Wang, and Chao Qi. 2017. Neural response generation via gan with an approximate embedding layer. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Training Millions of Personalized Dialogue Agents

Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, Antoine Bordes
Facebook
{pem, samuelhumeau, raison, abordes}@fb.com

Abstract

Current dialogue systems are not very engaging for users, especially when trained end-to-end without relying on proactive reengaging scripted strategies. Zhang et al. (2018) showed that the engagement level of end-to-end dialogue models increases when conditioning them on text personas providing some personalized back-story to the model. However, the dataset used in (Zhang et al., 2018) is synthetic and of limited size as it contains around 1k different personas. In this paper we introduce a new dataset providing 5 million personas and 700 million persona-based dialogues. Our experiments show that, at this scale, training using personas still improves the performance of end-to-end systems. In addition, we show that other tasks benefit from the wide coverage of our dataset by fine-tuning our model on the data from (Zhang et al., 2018) and achieving state-of-the-art results.

1 Introduction

End-to-end dialogue systems, based on neural architectures like bidirectional LSTMs or Memory Networks (Sukhbaatar et al., 2015) trained directly by gradient descent on dialogue logs, have been showing promising performance in multiple contexts (Wen et al., 2016; Serban et al., 2016; Bordes et al., 2016). One of their main advantages is that they can rely on large data sources of existing dialogues to learn to cover various domains without requiring any expert knowledge. However, the flip side is that they also exhibit limited engagement, especially in chit-chat settings: they lack consistency and do not leverage proactive engagement strategies as (even partially) scripted chatbots do.

Zhang et al. (2018) introduced the PERSONA-CHAT dataset as a solution to cope with this issue. This dataset consists of dialogues between pairs of agents with text profiles, or *personas*, attached to each of them. As shown in their paper, conditioning an end-to-end system on a given persona improves the engagement of a dialogue agent. This paves the way to potentially end-to-end personalized chatbots because the personas of the bots, by being short texts, could be easily edited by most users. However, the PERSONA-CHAT dataset was created using an artificial data collection mechanism based on Mechanical Turk. As a result, neither dialogs nor personas can be fully representative of real user-bot interactions and the dataset coverage remains limited, containing a bit more than 1k different personas.

In this paper, we build a very large-scale persona-based dialogue dataset using conversations previously extracted from REDDIT[1]. With simple heuristics, we create a corpus of over 5 million personas spanning more than 700 million conversations. We train persona-based end-to-end dialogue models on this dataset. These models outperform their counterparts that do not have access to personas, confirming results of Zhang et al. (2018). In addition, the coverage of our dataset seems very good since pre-training on it also leads to state-of-the-art results on the PERSONA-CHAT dataset.

2 Related work

With the rise of end-to-end dialogue systems, personalized trained systems have started to appear. Li et al. (2016) proposed to learn latent variables representing each speaker's bias/personality in a dialogue model. Other classic strategies include extracting explicit variables from structured knowledge bases or other symbolic sources as in (Ghazvininejad et al., 2017; Joshi et al., 2017; Young et al., 2017). Still, in the context of per-

---

[1] https://www.reddit.com/r/datasets/comments/3bxlg7/

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2775–2779
Brussels, Belgium, October 31 - November 4, 2018.

sonal chatbots, it might be more desirable to condition on data that can be generated and interpreted by the user itself such as text rather than relying on some knowledge base facts that might not exist for everyone or a great variety of situations. PERSONA-CHAT (Zhang et al., 2018) recently introduced a dataset of conversations revolving around human habits and preferences. In their experiments, they showed that conditioning on a text description of each speaker's habits, their *persona*, improved dialogue modeling.

In this paper, we use a pre-existing REDDIT data dump as data source. REDDIT is a massive online message board. Dodge et al. (2015) used it to assess chit-chat qualities of generic dialogue models. Yang et al. (2018) used response prediction on REDDIT as an auxiliary task in order to improve prediction performance on natural language inference problems.

3 Building a dataset of millions of persona-based dialogues

Our goal is to learn to predict responses based on a persona for a large variety of personas. To that end, we build a dataset of examples of the following form using data from REDDIT:

- Persona: [*"I like sport"*, *"I work a lot"*]
- Context: *"I love running."*
- Response: *"Me too! But only on weekends."*

The persona is a set of sentences representing the personality of the responding agent, the context is the utterance that it responds to, and the response is the answer to be predicted.

3.1 Preprocessing

As in (Dodge et al., 2015), we use a preexisting dump of REDDIT that consists of 1.7 billion comments. We tokenize sentences by padding all special characters with a space and splitting on whitespace characters. We create a dictionary containing the 250k most frequent tokens. We truncate comments that are longer than 100 tokens.

3.2 Persona extraction

We construct the persona of a user by gathering all the comments they wrote, splitting them into sentences, and selecting the sentences that satisfy the following rules: (i) each sentence must contain between 4 and 20 words or punctuation marks, (ii) it contains either the word *I* or *my*, (iii) at least one verb, and (iv) at least one noun, pronoun or adjective.

To handle the quantity of data involved, we limit the size of a persona to N sentences for each user. We compare four different setups for persona creation. In the *rules* setup, we select up to N random sentences that satisfy the rules above. In the *rules + classifier* setup, we filter with the rules then score the resulting sentences using a bag-of-words classifier that is trained to discriminate PERSONA-CHAT persona sentences from random comments. We manually tune a threshold on the score in order to select sentences. If there are more than N eligible persona sentences for a given user, we keep the highest-scored ones. In the *random from user* setup, we randomly select sentences uttered by the user while keeping the sentence length requirement above (we ignore the other rules). The *random from dataset* baseline refers to random sentences from the dataset. They do not necessarily come from the same user. This last setup serves as a control mechanism to verify that the gains in prediction accuracy are due to the user-specific information contained in personas.

In the example at the beginning of this section, the response is clearly consistent with the persona. There may not always be such an obvious relationship between the two: the discussion topic may not be covered by the persona, a single user may write contradictory statements, and due to errors in the extraction process, some persona sentences may not represent a general trait of the user (e.g. *I am feeling happy today*).

3.3 Dataset creation

We take each pair of successive comments in a thread to form the context and response of an example. The persona corresponding to the response is extracted using one of the methods of Section 3.2. We split the dataset randomly between training, validation and test. Validation and test sets contain 50k examples each. We extract personas using training data only: test set responses cannot be contained explicitly in the persona.

In total, we select personas covering 4.6m users in the rule-based setups and 7.2m users in the random setups. This is a sizable fraction of the total 13.2m users of the dataset; depending on the persona selection setup, between 97 and 99.4 % of the training set examples are linked to a persona.

4 End-to-end dialogue models

We model dialogue by next utterance retrieval (Lowe et al., 2016), where a response is picked among a set of candidates and not generated.

4.1 Architecture

The overall architecture is depicted in Fig. 1. We encode the persona and the context using separate modules. As in Zhang et al. (2018), we combine the encoded context and persona using a 1-hop memory network with a residual connection, using the context as query and the set of persona sentences as memory. We also encode all candidate responses and compute the dot-product between all those candidate representations and the joint representation of the context and the persona. The predicted response is the candidate that maximizes the dot product.

We train by passing all the dot products through a softmax and maximizing the log-likelihood of the correct responses. We use mini-batches of training examples and, for each example therein, all the responses of the other examples of the same batch are used as negative responses.

4.2 Context and response encoders

Both context and response encoders share the same architecture and word embeddings but have different weights in the subsequent layers. We train three different encoder architectures.

Bag-of-words applies two linear projections separated by a $\tanh$ non-linearity to the word embeddings. We then sum the resulting sentence representation across all positions in the sentence and divide the result by $\sqrt{n}$ where n is the length of the sequence.

LSTM applies a 2-layer bidirectional LSTM. We use the last hidden state as encoded sentence.

Transformer is a variation of an End-to-end Memory Network (Sukhbaatar et al., 2015) introduced by Vaswani et al. (2017). Based solely on attention mechanisms, it exhibited state-of-the-art performance on next utterance retrieval tasks in dialogues (Yang et al., 2018). Here we use only its encoding module. We subsequently average the resulting representation across all positions in the sentence, yielding a fixed-size representation.

		hits@k		
	Persona	k=1	k=3	k=10
IR Baseline	No	5.6	9.9	19.5
BOW	No	51.7	64.7	77.9
BOW	Yes	53.9	67.9	81.9
LSTM	No	63.1	75.6	87.3
LSTM	Yes	66.3	79.5	90.6
Transformer	No	69.1	80.7	90.7
Transformer	Yes	**74.4**	**85.6**	**94.2**

Table 1: Test results when classifying the correct answer among a total of 100 possible answers.

4.3 Persona encoder

The persona encoder encodes each persona sentence separately. It relies on the same word embeddings as the context encoder and applies a linear layer on top of them. We then sum the representations across the sentence.

We deliberately choose a simpler architecture than the other encoders for performance reasons as the number of personas encoded for each batch is an order of magnitude greater than the number of training examples. Most personas are short sentences; we therefore expect a bag-of-words representation to encode them well.

5 Experiments

We train models on the persona-based dialogue dataset described in Section 3.3 and we evaluate its accuracy both on the original task and when transferring onto PERSONA-CHAT.

5.1 Experimental details

We optimize network parameters using Adamax with a learning rate of $8e{-}4$ on mini-batches of size 512. We initialize embeddings with FastText word vectors and optimize them during learning.

REDDIT LSTMs use a hidden size of 150; we concatenate the last hidden states for both directions and layers, resulting in a final representation of size 600. Transformer architectures on reddit use 4 layers with a hidden size of 300 and 6 attention heads, resulting in a final representation of size 300. We use Spacy for part-of-speech tagging in order to verify the persona extraction rules. We distribute the training by splitting each batch across 8 GPUs; we stop training after 1 full epoch, which takes about 3 days.

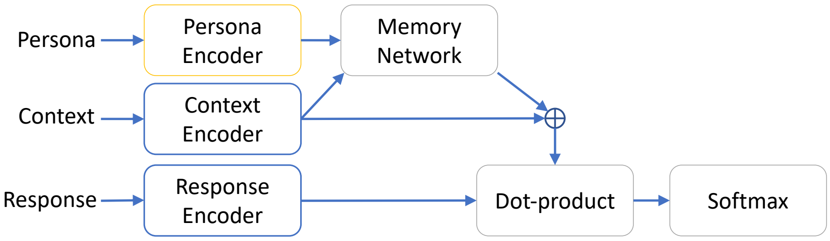

Figure 1: Persona-based network architecture.

Context (Persona)	*Predicted Answer*
Where do you come from?	
(I was born in London.)	*I'm from London, studying in Scotland.*
(I was born in New York.)	*I'm from New York.*
What do you do?	
(I am a doctor.)	*I am a sleep and respiratory therapist.*
(I am an engineer.)	*I am a software developer.*

Table 2: Sample predictions from the best model. In all selected cases the persona consists of a single sentence. The answer is constrained to be at most 10 tokens and is retrieved among 1M candidates sampled randomly from the training set.

PERSONA-CHAT We used the revised version of the dataset where the personas have been rephrased, making it a harder task. The dataset being only a few thousands samples, we had to reduce the architecture to avoid overfitting for the models trained purely on PERSONA-CHAT. 2 layers, 2 attention heads, a dropout of 0.2 and keeping the size of the word embeddings to 300 units yield the highest accuracy on the validation set.

IR Baseline As basic baseline, we use an information retrieval (IR) system that ranks candidate responses according to a TF-IDF weighted exact-match similarity with the context alone.

5.2 Results

Impact of personas We report the accuracy of the different architectures on the reddit task in Table 1. Conditioning on personas improves the prediction performance regardless of the encoder architecture. Table 2 gives some examples of how the persona affects the predicted answer.

Influence of the persona extraction In Table 3, we report precision results for several persona extraction setups. The *rules* setup improves the results somewhat, however adding the persona classifier actually degrades the results. A possible interpretation is that the persona classifier is trained only on the PERSONA-CHAT revised personas, and that this selection might be too narrow and lack diversity. Increasing the maximum persona size also improves the prediction performance.

N	Persona selection	hits@1
0	–	69.1
20	rules + classifier	70.7
20	rules	71.3
100	rules + classifier	72.5
100	rules	**74.4**
100	random from user	73.8
100	random from dataset	66.9

Table 3: Retrieval precision on the REDDIT test set using a Transformer and different persona selection systems. N: maximum number of sentences per persona.

Transfer learning We compare the performance of transformer models trained on REDDIT and on PERSONA-CHAT on both datasets. We report results in Table 4. This architecture provides a strong improvement over the results of (Zhang et al., 2018), jumping from 35.4% hits@1 to 42.1%. Pretraining the model on REDDIT and then fine-tuning on PERSONA-CHAT pushes this score to 60.7%, largely improving the state of the art. As expected, fine-tuning on PERSONA-CHAT reduces the performance on REDDIT. However, directly testing on PERSONA-CHAT the model trained on REDDIT without fine-tuning yields a very low result. This could be a consequence of a discrepancy

	Validation set	
Training set	PERSONA-CHAT	REDDIT
PERSONA-CHAT	42.1	3.04
REDDIT	25.6	**74.4**
FT-PC	**60.7**	65.5
IR Baseline	20.7	5.6
(Zhang et al., 2018)	35.4	–

Table 4: hits@1 results for the best found Transformer architecture on different test sets. FT-PC: REDDIT-trained model fine-tuned on the PERSONA-CHAT training set. To be comparable to the state of the art on each dataset, results on PERSONA-CHAT are computed using 20 candidates, while results on REDDIT use 100.

between the style of personas of the two datasets.

6 Conclusion

This paper shows how to create a very large dataset for persona-based dialogue. We show that training models to align answers both with the persona of their author and the context improves the predicting performance. The trained models show promising coverage as exhibited by the state-of-the-art transfer results on the PERSONA-CHAT dataset. As pretraining leads to a considerable improvement in performance, future work could be done fine-tuning this model for various dialog systems. Future work may also entail building more advanced strategies to select a limited number of personas for each user while maximizing the prediction performance.

References

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2015. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A knowledge-grounded neural conversation model. *CoRR*, abs/1702.01932.

Chaitanya K Joshi, Fei Mi, and Boi Faltings. 2017. Personalization in goal-oriented dialog. *arXiv preprint arXiv:1706.07503*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *CoRR*, abs/1603.06155.

Ryan Lowe, Iulian V Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. On the evaluation of dialogue systems with next utterance classification. *arXiv preprint arXiv:1605.05414*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *Proceedings of NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning semantic textual similarity from conversations. *CoRR*, abs/1804.07754.

Tom Young, Erik Cambria, Iti Chaturvedi, Minlie Huang, Hao Zhou, and Subham Biswas. 2017. Augmenting end-to-end dialog systems with commonsense knowledge. *arXiv preprint arXiv:1709.05453*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *CoRR*, abs/1801.07243.

Towards Universal Dialogue State Tracking

Liliang Ren, Kaige Xie, Lu Chen and Kai Yu
Key Lab. of Shanghai Education Commission for Intelligent Interaction and Cognitive Eng.
SpeechLab, Department of Computer Science and Engineering
Brain Science and Technology Research Center
Shanghai Jiao Tong University, Shanghai, China
`{renll204,lightyear0117,chenlusz,kai.yu}@sjtu.edu.cn`

Abstract

Dialogue state tracking is the core part of a spoken dialogue system. It estimates the beliefs of possible user's goals at every dialogue turn. However, for most current approaches, it's difficult to scale to large dialogue domains. They have one or more of following limitations: (a) Some models don't work in the situation where slot values in ontology changes dynamically; (b) The number of model parameters is proportional to the number of slots; (c) Some models extract features based on hand-crafted lexicons. To tackle these challenges, we propose StateNet, a *universal* dialogue state tracker. It is independent of the number of values, shares parameters across all slots, and uses pre-trained word vectors instead of explicit semantic dictionaries. Our experiments on two datasets show that our approach not only overcomes the limitations, but also significantly outperforms the performance of state-of-the-art approaches.

1 Introduction

A task-oriented spoken dialogue system (SDS) is a system that can continuously interact with a human to accomplish a predefined task through speech. It usually consists of three modules: input, output, and control. The control module is also referred to as *dialogue management* (Young et al., 2010; Yu et al., 2014). It has two missions: dialogue state tracking (DST) and decision making. At each dialogue turn, a state tracker maintains the internal state of the system based on the information received from the input module. Then a machine action is chosen based on the dialogue state according to a dialogue policy to direct the dialogue (Chen et al., 2018).

The dialogue state is an encoding of the machine's understanding of the whole conversation. Traditionally, it is usually factorized into three distinct components (Young et al., 2013): the user's *goal*, the user's action, and the dialogue history. Among them, the user's goal is most important, which is often simply represented by *slot-value* pairs. In this paper, we focus on the tracking of the user's goal.

Recently, the dialogue state tracking challenges (DSTCs) (Williams et al., 2013; Henderson et al., 2014a,d) are organized to provide shared tasks for comparing DST algorithms. A various of models are proposed, e.g. rule-based models (Wang and Lemon, 2013; Sun et al., 2014a; Yu et al., 2015, 2016; Sun et al., 2016b), generative statistical models (Thomson and Young, 2010; Young et al., 2010, 2013), and discriminative statistical models (Lee and Eskenazi, 2013; Lee, 2013; Sun et al., 2014b; Xie et al., 2015; Sun et al., 2016a; Xie et al., 2018). And the state-of-the-art one is the deep learning-based approach. However, most of these models have some limitations. First, some models can only work on a fixed domain *ontology*, i.e. the slots and values are defined in advance, and can't change dynamically. However, this is not flexible in practice (Xu and Hu, 2018). For example, in the tourist information domain, new restaurants or hotels are often added, which results in the change of the ontology. Second, in many approaches the models for every slot are different. Therefore, the number of parameters is proportional to the number of slots. Third, some models extract features based on text *delexicalisation* (Henderson et al., 2014b), which depends on predefined semantic dictionaries. In large scale domains, it's hard to manually construct the semantic dictionaries for all slots and values (Mrkšić et al., 2017).

To tackle these challenges, here we propose a *universal* dialogue state tracker, StateNet. For each state slot, StateNet generates a fixed-length representation of the dialogue history, and then compares the distances between this representation and the value vectors in the candidate set for making prediction. The set of candidate values

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2780–2786
Brussels, Belgium, October 31 - November 4, 2018.

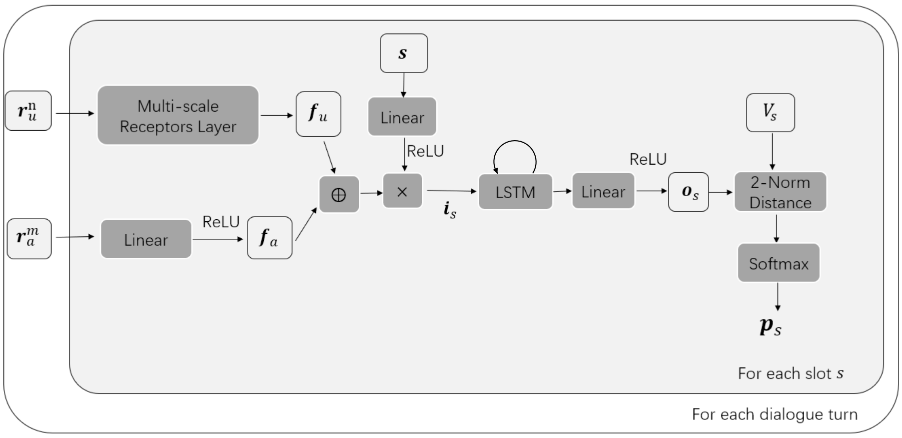

Figure 1: General model architecture of StateNet.

can change dynamically. StateNet only needs the following three parts of the data: (1) the original ASR information (or the transcript) of the user utterance; (2) the information of the machine act; (3) the literal names of the slots and the values. The manually-tagging of the user utterance is not needed as a part of the data. StateNet shares parameters among all slots, through which we can not only transfer knowledge among slots but also reduce the number of parameters.

2 StateNet: A Universal Dialogue State Tracker

For each dialogue turn, StateNet takes the multiple n-gram user utterance representation, $\mathbf{r}_u^n$, the n-gram machine act representation, $\mathbf{r}_a^n$, the value set, $\mathcal{V}_s$, and the word vector of the slot, $\mathbf{s}$, as the input. Then StateNet applies the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to track the inner dialogue states among the dialogue turns. And for each slot, StateNet outputs a corresponding probability distribution, $\mathbf{p}_s$, over the set of possible values, $\mathcal{V}_s$, at each of the dialogue turn,

$$\mathbf{p}_s = \text{StateNet}(\mathbf{r}_u^n, \mathbf{r}_a^n, \mathbf{s}, \mathcal{V}_s).$$

The general model architecture is shown in Figure 1.

2.1 User Utterance Representation

At the t-th dialogue turn, the user utterance, U_t, may consist of l number of words, u_i, with their corresponding word vectors, $\mathbf{u}_i$, $(1 \le i \le l)$. The user utterance may also have its corresponding m-best ASR hypotheses with the normalized confidence scores (Chen et al., 2017), q_j,$(1 \le j \le m)$. In this case, we can calculate the weighted word vectors, $\mathbf{u}'_i$,

$$\mathbf{u}'_i = \sum_{j=1}^{m} q_j \mathbf{u}_{i,j},$$

where $\mathbf{u}_{i,j}$ represents the word vector $\mathbf{u}_i$ presented at the j-th ASR hypothesis, and the zero vectors are padded at the end of all the hypotheses that are shorter than the longest one to have a same length of the utterance.

Based on the weighted word vectors generalizing the information from the ASR hypothesis, we can then construct the n-gram weighted word vectors, as proposed by Mrkšić et al. (2017),

$$\mathbf{u'}_i^n = \mathbf{u}'_i \oplus ... \oplus \mathbf{u}'_{i+n-1},$$

where $\oplus$ is the concatenation operator between the word vectors.

An n-gram user utterance representation is then constructed through a sum of the n-gram weighted word vectors,

$$\mathbf{r}_u^n = \sum_{i=1}^{l-n+1} \mathbf{u'}_i^n.$$

2.2 Multi-scale Receptors Layer

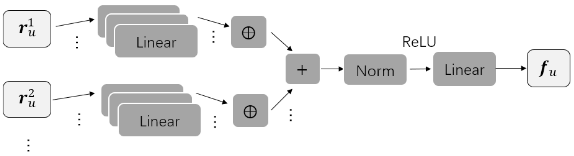

Figure 2: Multi-scale Receptors Layer.

For each gram k of the user utterance representation, $\mathbf{r}_u^k$, $(1 \le k \le n)$, the Multi-scale Receptors Layer has c number of linear neural networks (with the same number of neurons, N_c). Each of them takes the representation as input and is expected to work as the specialized receptor to amplify the signals from some of the word vectors in the utterance representation,

$$\hat{\mathbf{r}}_u^k = \oplus_{j=1}^{c}(\mathbf{W}_k^j \mathbf{r}_u^k + \mathbf{b}_k^j),$$

where $\mathbf{W}_k^j$ means the weight of the j-th linear layer, $\mathbf{b}_k^j$ means the corresponding bias, and $\oplus$ is the concatenation operator between the neurons of these linear layers. Note that each receptor does not necessarily has to be a single linear neural network and can be sophisticated with multiple layers and non-linearity for better detection performance. Here we only use the linear layer to provide a baseline of this kind of structure design.

These c number of linear layers (or receptors) for different grams (or scales) of the representation $\hat{\mathbf{r}}_u^k$ is then summed together to be layer-normalized (Ba et al., 2016). After that, the ReLU activation function is applied, followed by a linear layer with the size N_c that maps all the receptors to a user feature vector, $\mathbf{f}_u$,

$$\mathbf{f}_u = \text{Linear}(\text{ReLU}(\text{LayerNorm}(\sum_{k=1}^{n} \hat{\mathbf{r}}_u^k))).$$

2.3 Machine Act Representation

We represent the machine act in the m order n-gram of bag of words, $\mathbf{r}_a^m$, based on the vocabularies generalized from the machine acts in the training set of a given data set. The machine act feature, $\mathbf{f}_a$, is then simply generated through a linear layer of size N_c with the ReLU activation function,

$$\mathbf{f}_a = \text{ReLU}(\text{Linear}(\mathbf{r}_a^m)).$$

2.4 Slot Information Decoding

Since a slot, e.g. *area* or *food*, is usually indicated as a word or a short word group, then it can be represented as a single word vector (with multiple word vectors summed together), $\mathbf{s}$. A single linear layer with the size $2N_c$ is applied to the word vector $\mathbf{s}$, followed by the ReLU non-linear layer,

$$\mathbf{f}_s = \text{ReLU}(\text{Linear}(\mathbf{s})).$$

The turn-level feature vector, $\mathbf{i}_s$, is then generated through a point-wise multiplication $\otimes$ between the slot feature and the concatenation of the user feature and the machine act feature,

$$\mathbf{i}_s = \mathbf{f}_s \otimes (\mathbf{f}_u \oplus \mathbf{f}_a).$$

In this way, the turn-level feature vector is intended to amplify the large magnitude signals that are from both the user and machine act feature vector and the slot feature vector.

2.5 Fixed-length Value Prediction

Given the turn-level feature vector, $\mathbf{i}_s$, we can now track the dialogue state throughout the dialogue turns by LSTM. For the current turn t, the LSTM takes the $\mathbf{i}_s$ and the previous hidden state, $\mathbf{q}_{t-1}$, as the input. We can then obtain a fixed-length value prediction vector, $\mathbf{o}_s$, whose length is equal to N_w, i.e. the dimension of the word vectors which are fed into the model,

$$\mathbf{o}_s = \text{ReLU}(\text{Linear}(\text{LSTM}(\mathbf{i}_s, \mathbf{q}_{t-1}))),$$

where the linear layer has N_w neurons. In this way, the prediction of the model is independent of the number of the given values, so it is possible for the model to perform parameter sharing among each of the slots. The fixed-length prediction can somehow be interpreted as a *word vector* that is ready for the calculation of the similarity between the prediction and the true value label.

2.6 2-Norm Distance

For a specific semantic slot, since there may be no corresponding value in a given dialogue turn, thus we always add a literally "none" value to the value set for the model to track this state. For the evaluation of the similarity between the prediction and the value, we calculate the 2-Norm distance between the prediction vector and each of the word vectors of the values in the value set. Softmax function is performed with respect to all the negative relative distances to give a distribution of probabilities for the values, $v_i \in \mathcal{V}_s$,

$$p_s(v_i) = \text{Softmax}(-||\mathbf{o}_s - \mathbf{v}_i||),$$

where $\mathbf{v}_i$ is the representation vector of v_i. If the slot value v_i consists of more than one word, $\mathbf{v}_i$ will then be the summation of all corresponding word vectors. When training the model, we minimize the Cross-Entropy (CE) loss between the output probabilities and the given label.

StateNet requires the user utterance, the semantic slots, and slot values to be able to be expressed in words and have their corresponding word vectors. We use the fixed word embedding for every word, and do not fine-tune the word embeddings in the model. Since the word embeddings are distributed on a fixed-dimension vector space and hold rich semantic information, StateNet may have the ability to track the dialogue state for any new slot or value, as long as the corresponding word embedding can be found. This is the reason why we call the StateNet a *universal* dialogue state tracker.

3 Experiments

Experiments are conducted to assess the performance on joint goal. Two datasets are used by us for training and evaluation. One is the second Dialogue State Tracking Challenge (**DSTC2**) dataset (Henderson et al., 2014a), and the other is the second version of Wizard-of-Oz (**WOZ 2.0**) dataset (Wen et al., 2017). Both of them are the conversations between users and a machine system. The user's goal is to find a suitable restaurant around Cambridge. The ontology of these two datasets is identical, which is composed of three informable slots: *food, pricerange* and *area*. The main difference between them is that in WOZ 2.0, users typed instead of using speech directly. This means the users can use far more sophisticated language than they can in the DSTC2, which is a big challenge for the language understanding ability of the model. Thus, it allows WOZ 2.0 to be more indicative of the model's actual performance since it is immune to ASR errors.

Based on the model structure as described in Section 2, we implement three kinds of dialogue state tracker. The difference among them lies in the utilization of parameter sharing and parameter initialization.

- `StateNet`: It doesn't have shared parameters among different slots. In other words, three models for three slots are trained separately using RMSProp optimizer, learning rate set to 0.0005. And its parameters are not initialized with any pre-trained model.
- `StateNet_PS`: Parameter sharing is conducted among three slots. For each slot in a batch, we infer the model with the slot information and the same dialogue information. The losses are calculated based on the corresponding value set. After each slot is inferred, we back-propagate all the losses and do the optimization. So we just train one model in total using RMSProp optimizer, learning rate set to 0.0005. As a result, the amount of model parameters is one third of that of `StateNet`, which means `StateNet_PS` can significantly save the memory usage during inferring.
- `StateNet_PSI`: Parameter sharing is conducted within this model, same as `StateNet_PS`, but its parameters are initialized with a pre-trained model. For pre-training, we only allow the model to track one single slot and make predictions on its value set. After the training ends, we save the model parameters and use them to initialize the model parameters for the training of the multi-slot tracking. The pre-trained model with the best performance on the validation set is selected for initialization. Here, we choose the *food* slot for pre-training since `StateNet` has the lowest prediction accuracy on the *food* slot. `StateNet_PSI` is trained using Adam optimizer and learning rate is set to 0.001. Since the model has obtained the basic knowledge from the pre-trained model, then a more aggressive learning process is preferred. Adam with a higher learning rate can help a lot compared to RMSProp optimizer.

The hyperparameters are identical for all three models, $N_c = 128, N_w = 300, n = 2, m = 3$. We use $c = 4$ for the number of the receptors for each slot, where the number is determined through the grid search. The word embeddings used by us is the *semantically specialised* Paragram-SL999 vectors (Wieting et al., 2015) with the dimension of 300, which contain richer semantic contents compared to other kinds of word embeddings. Implemented with the MXNet deep learning framework of Version 1.1.0, the model is trained with a batch size of 32 for 150 epochs on a single NVIDIA GTX 1080Ti GPU.

The results in Table 1 show the effectiveness of parameter sharing and initialization. `StateNet_PS` outperforms `StateNet`, and `StateNet_PSI` performs best among all 3 models. It is because the parameter sharing can not only prevent the model diverging from the right learning process but also transfer necessary knowledge among different slots. And the parameter initialization provides the model with the opportunity to gain some basic while essential semantic information at the very beginning since the *food* slot is the most important and difficult one. Besides, `StateNet_PSI` beats all the mod-

DST Models	Joint Acc. DSTC2	Joint Acc. WOZ 2.0
Delexicalisation-Based (DB) Model (Mrkšić et al., 2017)	69.1	70.8
DB Model + Semantic Dictionary (Mrkšić et al., 2017)	72.9	83.7
Scalable Multi-domain DST (Rastogi et al., 2017)	70.3	-
MemN2N (Perez and Liu, 2017)	74.0	-
PtrNet (Xu and Hu, 2018)	72.1	-
Neural Belief Tracker: NBT-DNN (Mrkšić et al., 2017)	72.6	84.4
Neural Belief Tracker: NBT-CNN (Mrkšić et al., 2017)	73.4	84.2
Belief Tracking: Bi-LSTM (Ramadan et al., 2018)	-	85.1
Belief Tracking: CNN (Ramadan et al., 2018)	-	85.5
GLAD (Zhong et al., 2018)	74.5	88.1
StateNet	74.1	87.8
StateNet_PS	74.5	88.2
StateNet_PSI	**75.5**	**88.9**

Table 1: Joint goal accuracy on DSTC2 and WOZ 2.0 test set vs. various approaches as reported in the literature.

els reported in the previous literature, whether the model with delexicalisation (Henderson et al., 2014b,c; Rastogi et al., 2017) or not (Mrkšić et al., 2017; Perez and Liu, 2017; Xu and Hu, 2018; Ramadan et al., 2018; Zhong et al., 2018).

Initialization	Joint Acc. DSTC2	Joint Acc. WOZ 2.0
food	**75.5**	**88.9**
pricerange	73.6	88.2
area	73.5	87.8

Table 2: Joint goal accuracy on DSTC2 and WOZ 2.0 of `StateNet_PSI` using different pre-trained models based on different single slot.

We also test `StateNet_PSI` with different pre-trained models, as shown in Table 2. The fact that the *food* initialization has the best performance verifies our selection of the slot with the worst performance for pre-training. This is because the good performance on joint goal requires a model to make correct predictions on all of the slots. A slot on which the model has the worst accuracy, i.e. the most difficult slot, will dramatically limit the overall model performance on the metric of the joint goal accuracy. Thus, the initialization with a model pre-trained on the most difficult slot can improve the performance of the model on its weakness slot and boost the joint goal accuracy, while the initialization of a strength slot may not help much for the overall accuracy but in turn causes the over-fitting problem of the slot itself.

4 Conclusion

In this paper, we propose a novel dialogue state tracker that has the state-of-the-art accuracy as well as the following three advantages: 1) the model does not need manually-tagged user utterance; 2) the model is scalable for the slots that need tracking, and the number of the model parameters will not increase as the number of the slots increases, because the model can share parameters among different slots; 3) the model is independent of the number of slot values, which means for a given slot, the model can make the prediction on a new value as long as we have the corresponding word vector of this new value. If there are a great number of values for a certain slot, to reduce the computational complexity, we can utilize a fixed-size candidate set (Rastogi et al., 2017), which dynamically changes as the dialogue goes on. Experiment results demonstrate the effectiveness of parameter sharing & initialization.

Our future work is to evaluate the performance of our models in the scenario where there are new slots and more unobserved slot values, and to evaluate the domain-transferring ability of our models.

Acknowledgments

The corresponding author is Kai Yu. This work has been supported by the National Key Research and Development Program of China (Grant No. 2017YFB1002102) and the China NSFC project (No. 61573241). Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Lu Chen, Bowen Tan, Sishan Long, and Kai Yu. 2018. Structured dialogue policy with graph neural networks. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1257–1268.

Zhehuai Chen, Yimeng Zhuang, and Kai Yu. 2017. Confidence measures for ctc-based phone synchronous decoding. In *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, pages 4850–4854.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014c. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Mettew Henderson, Blaise Thomson, and Jason D. Williams. 2014d. The third dialog state tracking challenge. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*.

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

Sungjin Lee. 2013. Structured discriminative model for dialog state tracking. In *Proceedings of the SIGDIAL 2013 Conference*, pages 442–451, Metz, France. Association for Computational Linguistics.

Sungjin Lee and Maxine Eskenazi. 2013. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *Proceedings of the SIGDIAL 2013 Conference*, pages 414–422, Metz, France. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1777–1788.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using memory network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 305–314.

Osman Ramadan, Paweł Budzianowski, and Milica Gasic. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 432–437.

Abhinav Rastogi, Dilek Hakkani-Tur, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. *arXiv preprint arXiv:1712.10224*.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014a. A generalized rule based tracker for dialogue state tracking. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014b. The SJTU system for dialog state tracking challenge 2. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 318–326, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Kai Sun, Qizhe Xie, and Kai Yu. 2016a. Recurrent polynomial network for dialogue state tracking. *Dialogue & Discourse*, 7(3):65–88.

Kai Sun, Su Zhu, Lu Chen, Siqiu Yao, Xueyang Wu, and Kai Yu. 2016b. Hybrid dialogue state tracking for real world human-to-human dialogues. In *Proc. InterSpeech*, pages 2060–2064, San Francisco, America.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.

Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432, Metz, France. Association for Computational Linguistics.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, pages 438–449, Valencia, Spain. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France. Association for Computational Linguistics.

Kaige Xie, Cheng Chang, Liliang Ren, Lu Chen, and Kai Yu. 2018. Cost-sensitive active learning for dialogue state tracking. In *Proceedings of the 19th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 209–213, Melbourne, Australia. Association for Computational Linguistics.

Qizhe Xie, Kai Sun, Su Zhu, Lu Chen, and Kai Yu. 2015. Recurrent polynomial network for dialogue state tracking with mismatched semantic parsers. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 295–304, Prague, Czech Republic. Association for Computational Linguistics.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

Steve Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Kai Yu, Lu Chen, Bo Chen, Kai Sun, and Su Zhu. 2014. Cognitive technology in task-oriented dialogue systems: Concepts, advances and future. *Chinese Journal of Computers*, 37(18):1–17.

Kai Yu, Lu Chen, Kai Sun, Qizhe Xie, and Su Zhu. 2016. Evolvable dialogue state tracking for statistical dialogue management. *Frontiers of Computer Science*, 10(2):201–215.

Kai Yu, Kai Sun, Lu Chen, and Su Zhu. 2015. Constrained markov bayesian polynomial for efficient dialogue state tracking. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(12):2177–2188.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1458–1467.

Semantic Parsing for Task Oriented Dialog using Hierarchical Representations

Sonal Gupta **Rushin Shah** **Mrinal Mohit** **Anuj Kumar**
Facebook Conversational AI

Michael Lewis
Facebook AI Research
{sonalgupta, rushinshah, mrinalmohit, anujk, mikelewis}@fb.com

Abstract

Task oriented dialog systems typically first parse user utterances to semantic frames comprised of intents and slots. Previous work on task oriented intent and slot-filling work has been restricted to one intent per query and one slot label per token, and thus cannot model complex compositional requests. Alternative semantic parsing systems have represented queries as logical forms, but these are challenging to annotate and parse. We propose a hierarchical annotation scheme for semantic parsing that allows the representation of compositional queries, and can be efficiently and accurately parsed by standard constituency parsing models. We release a dataset of 44k annotated queries [1], and show that parsing models outperform sequence-to-sequence approaches on this dataset.

1 Introduction

Intelligent personal assistants are now ubiquitous, but modeling the semantics of complex compositional natural language queries remains challenging. Typical systems classify the *intent* of a query (e.g. `GET_DIRECTIONS`) and tag the necessary slots (e.g. `San Francisco`) (Mesnil et al., 2013; Liu and Lane, 2016). It is difficult for such representations to adequately represent nested queries such as *"Driving directions to the Eagles game"*, which is composed of `GET_DIRECTIONS` and `GET_EVENT` intents. We explore a hierarchical representation for such queries, which dramatically improves the expressive power while remaining accurate and efficient to annotate and parse (see Figure 1).

We introduce a Task Oriented Parsing (TOP) representation for intent-slot based dialog systems. This hierarchical representation is expressive enough to capture the semantics of complex nested queries, but is easier to annotate and parse than alternative representations such as logical forms or dependency graphs. We show empirically that our representation is expressive enough to model the vast majority of human-generated requests in two domains.

A key advantage of our representation is that it has a structure similar to standard constituency parses, allowing us to easily adapt algorithms developed for phrase structure parsing for inference. In particular, we use linear-time Recurrent Neural Network Grammars (RNNG) (Dyer et al., 2016) and show that the inductive bias provided by this model significantly improves the accuracy compared to strong sequence-to-sequence (seq2seq) models based on CNNs, LSTMs and Transformers.

Our contributions in this paper are:

1. A hierarchical semantic representation for task oriented dialog systems that can model compositional and nested queries.
2. A publicly available dataset of 44k requests annotated with our representation. We show that our representation has very high coverage of these requests, and that inter-annotator agreement is high.
3. We show that the representation is learnable by standard algorithms. In particular, we show that the RNNG parsing model outperforms seq2seq baselines.

2 Representation

Designing semantic annotation schemes requires trade-offs between how expressive the representation is on one hand, and how easily can it be annotated, parsed, and executed on the other. Most existing annotations for task oriented dialog systems have fallen on the extremes of non-recursive intent and slot tagging, such as in the ATIS dataset (Mes-

---

[1] http://fb.me/semanticparsingdialog

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2787–2792
Brussels, Belgium, October 31 - November 4, 2018.

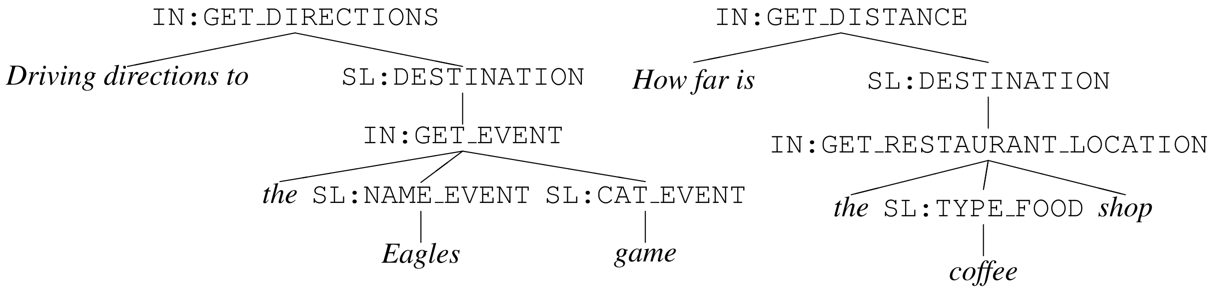

Figure 1: Example TOP annotations of utterances. Intents are prefixed with `IN:` and slots with `SL:`. In a traditional intent-slot system, the `SL:DESTINATION` could not have an intent nested inside it.

nil et al., 2013; Liu and Lane, 2016), and full logical forms (Zettlemoyer and Collins, 2012).

We introduce a hierarchical representation, similar to a constituency syntax tree, with words as terminals. Non-terminals are either *intents* or *slots*, and the root node is an intent. We allow intents to be nested inside a slot, resulting in the ability to compose requests and call multiple APIs. Using this compositional tree representation, we can enable answering compositional queries over multiple domains.

We introduce the following constraints in our representation: 1. The top level node must be an intent, 2. An intent can have tokens and/or slots as children, 3. A slot can have either tokens as children or one intent as a child.

Executing queries such as those in Figure 1 is straightforward because of the explicit tagging of the outer location slot: first we fetch 'the Eagles game' event (or the relevant coffee shop), extract the location, and pass it as the destination slot to the navigation domain intent.

Compositional queries are frequent. In our dataset of crowd-sourced utterances, we found that 30% could not be adequately represented with traditional intent-slot tagging.[2] This shows that more expressive representations are often necessary.

While our representation is capable of modeling many complex queries, some utterances are beyond its scope. For example, in *Set an alarm at 8 am for Monday and Wednesday*, *8 am* needs to be associated with both *Monday* and *Wednesday* which would require graph-structured representations. However, we found that just 0.3% of our dataset would require a more expressive representation to model adequately.[3] More expressive representations, such as dependency graphs or logical forms, would only bring marginal gains but would add significant challenges for annotation and learning.

Together, these results suggest that our tree-structured approach offers a useful compromise between traditional intent-slot tagging and logical forms, by providing very high coverage of queries while avoiding the complexities of annotating and learning more expressive representations.

In summary, our representation has the following attractive properties:

- **Expressiveness** Compared to traditional intent-slot annotations, it can express complex hierarchical queries, improving coverage of queries by 30%. We found that more general representations are required for only 0.3% of queries.
- **Easy Annotation** Annotating our representation simply requires labeling spans of a sentence, which is much more straightforward than alternatives such as creating a logical form for the sentence, or an arbitrary dependency graph. This fact allows us to quickly create a large dataset.
- **Efficient and Accurate Parsing** Since our representation closely resembles syntactic trees, we can easily re-use models from the large literature on constituency parsing.

---

[2] In our dataset, 35% of queries have depth >2, which means that the traditional intent-slot tagging systems would not have been able to annotate or predict these annotations. In addition, to avoid the depth-based statistic being influenced by our label set specification, we manually performed analysis of 100 samples that showed that 30% of the queries required compositional representation.

[3] Utterances that could not be annotated with the label sets and corresponding instructions were marked as 'unsupported' (9.14% of the dataset), including the ones that needed more expressive representation than a tree. Analysis of a random sample of 120 unsupported utterances shows that only four instances cannot be supported because of the tree representation constraint, estimating that there are only 0.3% queries that would require a more general (e.g graph-based) representation.

- **Execution** Our approach can be seen as a simple generalization of traditional dialog systems, meaning that existing infrastructure can easily be adapted to execute the intents.

3 Dataset

We asked crowdsourced workers to generate natural language sentences that they would ask a system that could assist in navigation and event queries. These requests were then labeled by two annotators. If these annotations weren't identical then we adjudicated with a third annotator. If all three annotators disagreed then we discarded the utterance and its annotations. 63.40% of utterances were resolved with 2 annotations and 94.09% were resolved after getting 3 annotations. We also compared percentage of utterances that were resolved after 2 annotations for depth ≤ 2 (traditional slot filling) and for depth > 2 (compositional): 68.87% vs 62.03%, noting that the agreement rate is similar.

We collected a total of 44783 annotations with 25 intents and 36 slots, randomly split into 31279 training, 4462 validation and 9042 test utterances. The dataset has utterances that are focused on navigation, events, and navigation to events.

The median (mean) depth of the trees is 2 (2.54), and the median (mean) length of the utterances is 8 (8.93) tokens. 35% of trees have depth more than 2. The dataset has 4646 utterances that contain both navigation and event intents. Figure 2 shows the distribution of instances in the full dataset over the utterance length and tree depth.

4 Models

We experiment with two types of models: standard sequence-to-sequence learning models, and a model adapted from syntactic parsing, Recurrent Neural Network Grammars (Dyer et al., 2016) (RNNG). RNNG is a top-down transition-based parser and was originally proposed for parsing syntax trees and language modeling. We trained the RNNG parser discriminatively and not generatively to reduce training time of the model. While sequence-to-sequence learning can model arbitrary sequence transduction, we hypothesize that parsing models like RNNG, which can only output well-formed trees, will give better inductive bias and flexibility for predicting compositional and cross-domains scenarios on the fly, particularly for domains with less training data available.

We briefly review the RNNG model – The parse tree is constructed using a sequence of transitions, or 'actions'. The transitions are defined as a set of SHIFT, REDUCE, and the generation of intent and slot labels. SHIFT action *consumes* an input token (that is, adds the token as a child of the right most 'open' sub-tree node) and REDUCE *closes* a sub-tree. The third set of actions is generating non-terminals – the slot and intent labels. Note that at each step, there are only a subset of valid actions. For examples, if all the input tokens have been added to the tree, the only valid action is REDUCE.

5 Experiments and Results

Baselines

We compared RNNG with implementations of sequence-to-sequence models using CNNs (Gehring et al., 2017), LSTMs (Wiseman and Rush, 2016) and Transformer networks (Vaswani et al., 2017) in fairseq[4].

Metrics

We used three metrics to evaluate the systems. **Exact match** accuracy is defined as the number of utterances whose full trees are correctly predicted. The second metric is a commonly used scoring method for syntactic parsing – **labeled bracketing** F_1 scores (Black et al., 1991) (called F_1 henceforth). We used the pre-terminals as well in the calculations. One downside of this metric is that it only considers the token spans for a given non-terminal but not the internal structure. Tree substructures are rather important for task completion. Thus, we introduce a third metric, **Tree-Labeled** (TL) F_1, which compares the sub-tree structure for a non-terminal, instead of just the token span. Exact match accuracy is the strictest metric and F_1 is the least strict metric. **Tree Validity** is the percentage of predictions which formed valid trees (via bracket matching).

Preprocessing and Hyperparameters

We used the same preprocessing of unknown words as used in (Dyer et al., 2016) and mapped numbers to a constant. We used pre-trained GloVe embeddings (Pennington et al., 2014) and tuned hyperparameters on the validation set.

- **RNNG** - We used 2-layer 164-unit LSTM with a bidirectional LSTM compositional

---

[4] https://github.com/pytorch/fairseq

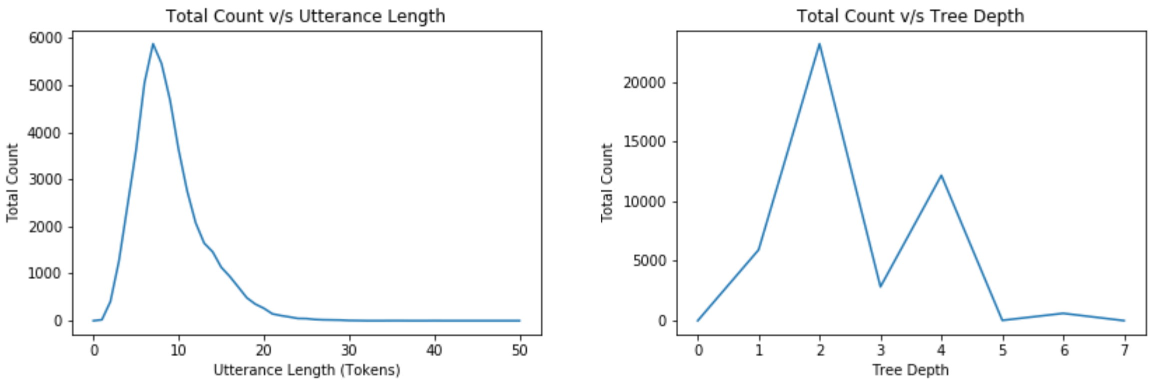

Figure 2: Count statistics of the full dataset.

Model	Exact match	F_1	Precision	Recall	TL-F_1	TL-Precision	TL-Recall	Tree Validity
RNNG	**78.51**	**90.23**	**90.62**	**89.84**	**84.27**	**84.64**	**83.91**	**100.00**
seq2seq-CNN (LOTV)	75.87	88.56	89.25	87.88	82.31	82.92	81.72	99.75
seq2seq-LSTM (LOTV)	75.31	87.69	88.35	87.03	81.15	81.72	80.58	99.94
seq2seq-Transformer	72.20	86.60	87.09	86.11	78.54	78.99	78.19	99.55

Table 1: Performance (in percentage) of RNNG and seq2seq models based on LSTMs, CNNs, and Transformer networks. The CNN and LSTM models were trained with a Limited Output Token Vocabulary (LOTV) of just a single element.

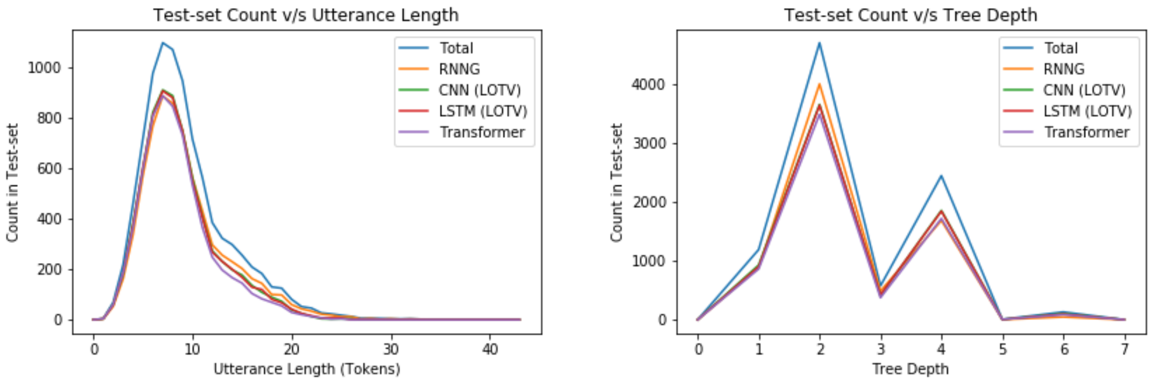

Figure 3: Exact match statistics of the different models on the testset

function, trained with a learning rate of 0.0004, weight decay of 0.00004, dropout of 0.34 with Adam optimizer for 1 epoch with 16 workers using Hogwild updates.

- **seq2seq CNN** - We used 3x3 convolutions (9 layers of 512 units, 4 units of 1024 units), followed by 1x1 convolutions (2 layers of 2048 units) with attention, trained with initial learning rate of 0.9, dropout of 0.2, gradient clipping of 0.1 with NAG optimizer for 30 epochs, inferred with beam size 5.
- **seq2seq Transformer** - We used 3 layers of 4 FFN attention heads of embedding dimension 512, trained with initial learning rate of 0.01, dropout of 0.2, gradient clipping of 5 with Adam optimizer for 50 epochs, inferred with beam size 5.
- **seq2seq LSTM** - We used 1-layer 256 unit LSTMs with attention, trained with initial learning rate of 0.7, dropout of 0.2, gradient clipping of 0.1 with NAG optimizer for 40 epochs, inferred with beam size 5.

Results

The experimental results in Table 1 and Figure 3 show that existing approaches for syntactic parsing, such as RNNG, perform well for this task, achieving perfect outputs on over 75% of queries. RNNG performs better than sequence-to-sequence models, especially in predicting exact trees, which is important for task completion. We present results for the CNN and LSTM models with an output terminal token vocabulary of just a single element (LOTV), which performed better than the regular token vocabulary (exact match accuracy of 75.63% and 68.39%, respectively). We believe LOTV makes the models focus on learning to predict the tree structure rather than to reproduce the input tokens. But we observed that this vocabulary reduction resulted in significantly poorer performance for the Transformer model.

Below we discuss how varying the beam size during inference affects accuracy. The seq2seq results in Table 1 are accuracy for the top prediction when a beam of size 5 was run and the RNNG results are for greedy inference. For RNNG, the accuracy of top prediction did not change much when a beam of size 5 was run. We also measured how often the correct tree annotation was in the top k predictions for the models when a beam of size k was run during inference, called as *Top-k*. For RNNG, *Top-3* was 90.21 and *Top-5* was 92.48, compared to 78.51 for k=1. For seq2seq-CNN, the *Top-3* score was 88.08 and *Top-5* score was 90.21. For seq2seq-LSTM, *Top-3* was 86.55 and *Top-5* was 88.76. We note that *Top-5* is substantially higher than the accuracy of the top prediction. These top-k predictions could be used by a hypothesis ranker downstream, which can take into account agent capabilities.

We also experimented with more minimal representations of the RNNG model (Kuncoro et al., 2017). Removing the actions LSTM dropped Exact match score slightly to 78.08. Separately, removing the stack LSTM dropped it to 75.31. Removing the buffer LSTM caused a unusable decrease to 13.78.

While sequence-to-sequence models have shown strong parsing performance when trained on very large amounts of data (Vinyals et al., 2015); in our setting the inductive bias provided by the RNNG model is crucial to achieving high performance. The model has several useful biases, such as guaranteeing a well-formed output tree, and shortening the dependencies between intents and their slots.

A further advantage of RNNG is that inference has linear time complexity, whereas seq2seq models are quadratic because attention is recomputed at every time step.

6 Related Work

Many annotation schemes have previously been proposed for representing the semantics of natural language. We briefly compare our method with these.

Most work on task oriented dialog systems has focused on identifying a single user intent and then filling the relevant slots – for example, the representations used on the ATIS dataset (Mesnil et al., 2013; Liu and Lane, 2016; Zhu and Yu, 2017) and in the Dialog State Tracking Challenge (Williams et al., 2016). We showed that hierarchical representations with nested intents can improve coverage of requests in our domains.

The semantic parsing literature has focused on representing language with logical forms (Liang, 2016; Zettlemoyer and Collins, 2012; Kwiatkowski et al., 2010). Logical forms are more expressive than our representation, as they are less tightly coupled to the input query, and can be executed directly. However, logical forms are difficult to annotate and no large-scale datasets are available.

While we used a tree-structured representation, others have used arbitrary graphs, such as Abstract Meaning Representation (Banarescu et al., 2013) and Alexa Meaning Representation (Fan et al., 2017). These approaches can represent complex constructions that are beyond the scope of our approach, but with significantly challenging parsing (Artzi et al., 2015). We showed that such cases are very rare in our data.

7 Conclusions

Drawing on ideas from slot-filling and semantic parsing, we introduce a hierarchical generalization of traditional intents and slots that allows the representation of complex nested queries, leading to 30% higher coverage of user requests. We show that the representation can be annotated with high agreement. We are releasing a large dataset of annotated utterances at `http://fb.me/semanticparsingdialog`. The representation allows the use of existing constituency parsing algorithms, resulting in higher accuracy than sequence-to-sequence models.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with AMR.

In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Ezra W. Black, Steven Abney, Daniel P. Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert J. P. Ingria, Frederick Jelinek, Judith L. Klavans, Mark Y. Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proc. DARPA Speech and Natural Language Workshop, pp. 306311.*

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, ACL*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. *ArXiv e-prints*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233. Association for Computational Linguistics.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Commun. ACM*, 59(9):68–76.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *INTERSPEECH*.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2773–2781, Cambridge, MA, USA. MIT Press.

Jason Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.

S. Zhu and K. Yu. 2017. Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5675–5679.

The glass ceiling in NLP

Natalie Schluter
Department of Computer Science
IT University
Copenhagen, Denmark
natschluter@itu.dk

Abstract

In this paper, we provide empirical evidence based on a rigourously studied mathematical model for bi-populated networks, that a glass ceiling within the field of NLP has developed since the mid 2000s.

1 Introduction

The *glass ceiling* is a powerful metaphor for the unethical, invisible, and yet virtually impenetrable barrier that prevents highly achieving women and minorities from obtaining equal access to senior career opportunities. The existence of a glass ceiling is well documented both in STEM[1] and specifically in Computer Science (Moss-Racusin et al., 2012; Shen, 2013; Larivière et al., 2013; Van der Lee and Ellemers, 2015; Way et al., 2016, for example). To date there has been no published study on this topic for the field of NLP.

In most countries, Computer Science has long been struggling to support female researchers sufficiently: female representation in Computer Science is not only disproportional to the population, but it is lower than the average STEM field. Moreover, as opposed to STEM fields in general, the proportion of women in Computer Science has been on a marked decline for the past two decades (Sax et al., 2017; Williams et al., 2017), placing the entire the tech field in a diversity crisis today.

The discussion of gender representation or even the existence of a glass ceiling is rather more complex for NLP due to its fundamental interdisciplinarity especially across the fields of Linguistics, Computer Science, and Statistics. That is, much mainstream research in NLP follows trends that are heavily situated in one of the main subdisciplines. Can we witness any emergent glass ceiling for female researchers in the wake of an increasing concentration on deep learning engineering techniques applied to NLP problems? What about the preceding Machine Learning wave from the mid 2000s? In this paper we answer this question in the affirmative.

We acquired a gender-annotated co-author dataset covering arguably the most central ACL publication venues for the past 52 years. We carry out basic data analysis over this dataset and the bi-populated (female and male researcher) mentor-mentee network derived from it. We make the following concerning empirical observations:

1. **There is a growing mentor gender gap.** There is a growing disparity between the proportions of female and male NLP researchers who achieve mentor status, with a higher proportion of male researchers becoming mentors, especially since the mid 2000s.
2. **There is a significant time gap to mentor status across genders.** Female NLP researchers must wait a considerable time longer to achieve mentor status than their male colleagues.
3. **In-gender mentorship correlates with future success.** Female NLP researchers who take a male supervisor will have greater difficulty in becoming a mentor than if they take a female supervisor, on average.
4. **Homophily is on the rise.** There is consistently increasing homophily in our field–the preference to establish in-gender mentor-mentee relationships.

Following this analysis, we employ Avin et al. (2015)'s rigorously studied conditions for power inequality and the glass-ceiling effect for complex systems data structured like ours to show that these empirical observations indicate quite precisely **the existence of a glass ceiling effect for the field of NLP**.

---

[1] Science, technology, engineering and mathematics fields.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2793–2798
Brussels, Belgium, October 31 - November 4, 2018.

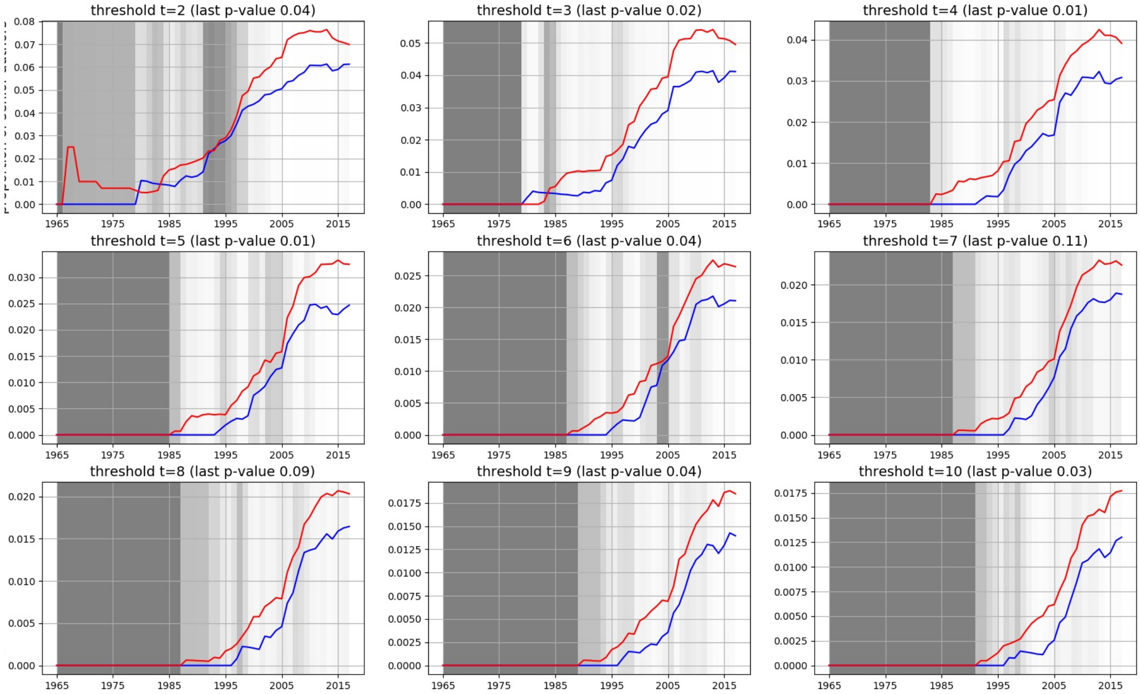

Figure 1: Proportion of male (red line) and female (blue line) mentors from 1966 to 2017, for different thresholds of "mentor seniority". A disparity in these proportions has been increasing since the 1990s, which follows the general field of Computer Science. The whiter the background the more significant the difference in proportions for the corresponding year, with the p-value for 2017 in the title.

2 Acquiring a gender-annotated mentor-mentee network

We scraped all meta-information available from the ACL Anthology[2] for arguably the most central publication venues in NLP. This includes all papers from CoNLL, EACL, TACL, CL, ACL, EMNLP, COLING, ANLP, NAACL, *Sem/SemEval from 1965 to 2017: 19,552 papers in total.

We carried out some normalisation of the author names scraped by lower-casing, normalising for order (first name then last name), removing middle initials and title abbreviations, and removing accents and punctuation, collapsing the extracted list of 18,437 author names to a list of 17,232 author names. Following this, we applied several gendered first-name lists to automatically annotate a large portion of the author names with gender.[3] This resulted in 13,435 automatically annotated author names. Of the remaining 3797 unannotated names, we automatically label as 'unknown' all author names with only an initial standing for the first name, effectively filtering out a further 565 author names. The remaining 3232 author names were annotated by the current authors by manually inspecting the results of Google Image queries for the full name.

The resulting dataset spans 52 years and includes 17,232 authors, of which we labeled 10,382 as male, 5,227 as female and 1,623 whose gender we could not identify. In what remains of our study, we discard these latter authors. This leaves a total of 15,609 researchers.

Power in academia. In our study, we need to account for mentor status–a type of seniority and power. As in many other fields, in NLP it is customary for mentors to take the last-authorship position of papers. Though there can be exceptions to this custom, the assumption of mentor last-authorship is simple, and with this large dataset, we believe it provides a robust approximation of mentorship in the absence of other more precise indications like centralised supervision logs. This method was also adopted by Avin et al. (2015).

We use the assumption of mentor last-authorship to provide an empirical definition of a mentor in our dataset. We say that after t last-authored papers for some threshold $t \in \{1, \ldots, 10\}$, and excluding all sole-author papers,

---

[2] `http://www.aclweb.org/anthology`

[3] The lists are discussed in the appendix.

a researcher is considered to hold *mentor standing with seniority threshold* t.

We model the interactions between researchers by creating the bi-populated (for female and male populations) mentor-mentee network. The network's nodes therefore are researchers and there is an edge between two co-authors of a paper in our dataset if and only if one of the co-authors is the last author. This leaves a mentor-mentee network with 14248 nodes, 25211 edges, and average degree 3.539. This network allows us to observe whether the current system of mentor-mentee relationships entails a glass ceiling effect in the modeled community. We now present the results of this analysis.

3 Evidence of a rising gender gap

We provide some basic empirical evidence which could be indicative of the presence of a glass ceiling effect in NLP. In Section 4, we then prove that there is indeed a glass ceiling.

3.1 Growing mentor standing disparity

A researcher who has achieved some seniority is generally eligible to become a mentor and supervise students. As such, the rise to becoming a mentor is a measurable criterion of success for a researcher in academia. Concretely, in some countries, the mentor role is reserved for permanent/tenured faculty (for example, in Denmark). Therefore a barrier to mentor standing for females can lead to an important under-representation of women. This under-representation in turn may perpetuate itself through the lower availability of same-gender advisors for female students, which we show to be of central importance for rising NLP researchers (in Section 3.3).

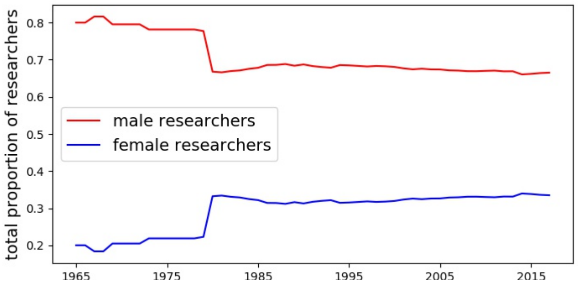

Figure 2: The proportion of female researchers (blue line) in NLP has been gradually increasing since 1965, but seems to be leveling off at around 33.5%, with a very slight decrease since 2014. A corresponding decrease is observed for male NLP researchers (red line).

For thresholds of "mentor seniority" $t \in \{2, \ldots, 10\}$ we examined the proportion of mentors with respect to the pool of researchers of the same gender over time from 1966 to today. Figure 1 shows the resulting time series. Across all thresholds, we observe that the proportion of male supervisors with respect to the total number of male researchers is increasing faster than the proportion of female supervisors within the general pool of female researchers. In fact, the discrepancy between these two proportions seems to slowly close until the early-to-mid 2000s after which it steadily increases again. And in almost all cases this difference in proportions develops into a statistically significant difference (with a 1-sided z-test for proportions, and p-value 0.05). This is despite there being no corresponding development in mentor-mentee proportions as shown in Figure 2.

3.2 Time to seniority gap

We further investigate the subset of female researchers who achieved mentor standing, and compare their difficulty in doing so with that of the respective pool of male researchers. One measurable factor from our dataset is time. Isolating a substantially larger delay to achieving mentor standing for female researchers is one way to use our dataset to measure the difficulty in transitioning female researchers from mentee to mentor standing. We consider the average time it takes to achieve mentor standing between the two populations. For consecutive periods of two years, we compute the average number of years for researchers to achieve mentor standing at threshold $t \in \{2, \ldots, 10\}$. We provide a visualisation of the results in Figure 3. We do a two sample t-test to expose the statistical significance in the non-equality of the respective means. We observe that across all thresholds for mentor standing, female researchers are substantially more delayed than male researchers in becoming mentors. For the most recent numbers, the result is most significant where there is the most data, at seniority $t = 3$, with p-level 0.04. However we note the general whitening of the plots (indicating statistical significance) after the mid-2000s.

3.3 The effects of in-gender supervision

The availability of female mentors been has shown to correlate with mentees' future success–in particular, females in Chemistry who are mentored by

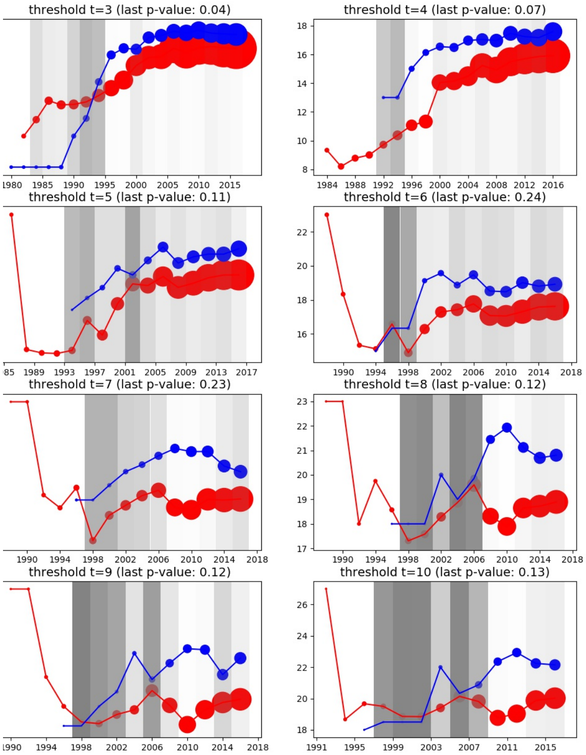

Figure 3: The average time it takes in years, across the past 52 years, for female researchers (blue line) and male researchers (red line) to achieve mentor status in the NLP field. The more significant the difference in means, the whiter the background. The relative number of data points for each year and gender are indicated by the size of the scatter points (blue for females and red for males). The last recorded p-level for each seniority threshold is provided in the corresponding plot's title.

	mentor					
	male	female	male	female	male	female
mentee	$t = 2$		$t = 3$		$t = 4$	
male	12.26	5.69	6.98	4.21	4.95	3.29
female	7.29	10.73	4.96	6.12	3.69	4.11
	$t = 5$		$t = 6$		$t = 7$	
male	3.91	2.72	3.25	2.4	2.64	1.99
female	2.83	3.08	2.31	2.47	2.01	2.1
	$t = 8$		$t = 9$		$t = 10$	
male	2.26	1.74	2.03	1.58	1.85	1.46
female	1.8	1.87	1.61	1.65	1.38	1.4

Table 1: Probabilities (as % here) that a mentee of the row gender, supervised by a mentor of the column gender will achieve mentor standing, for various thresholds t.

female supervisors are considerably more likely to become faculty themselves (Gaule and Piacentini, 2018). In Table 1 we observe a similar trend for in-gender mentorship. In particular, female researchers who have female mentors are much more likely to become mentors themselves. This is a particular problem if, as Sections 3.1 and 3.2 show, the proportion of female NLP mentors is not increasing at the same rate as that of male NLP researchers, possibly due in part to the added delay in achieving mentor status for women. Indeed this delay in access, perpetuated due to the lack of in-gender supervision, can be the result of a glass ceiling in NLP. In the next section we investigate the likelihood of such a glass ceiling.

4 The glass ceiling effect in NLP

In order to understand better how the population of female researchers in NLP can be increasing, but the growth level of seniority/mentor standing still falls significantly below that of the male population and that this gap is widening, we turn to an investigation of power inequality and the glass ceiling effect.

First three key observations can be made of the mentor-mentee network introduced in Section 2 vis-à-vis three well-accepted mechanisms of observed human behavior.

(O1) **Minority-majority partition.** Figure 2 shows the resulting proportion of male and female researchers in NLP through the 52 years. Our network displays a minority-majority partition: the proportion of females has hovered around 33.5% for the past decade now.

(O2) **Homophily** is the is the tendency of individuals to associate with people similar to them.

Easley and Kleinberg (2010) provide the following test for homophily. Given the proportions of male and female ended edges in the network, we should be able to calculate the approximate proportion of mixed edges (the probability that we select a mixed-gendered edge at random). If the true fraction is significantly below the expected amount, the network is exhibiting homophily. Figure 4 shows that homophily is a consistently worsening problem in the NLP community. All numbers are significant with p-value virtually 0 (1-sided z-test for proportions). Note that the plot includes error bars, which are so small they are not visible.

(O3) **The "rich-get-richer" feedback mechanism** describes and explains the process of wealth concentration, by which the future distribution of wealth is predictable from empirical data based on the current wealth distribution.

In our network, the degree of a node captures its level of social wealth: people may try to connect more often to people who already have many connections, either in order to profit from their social wealth or because they are more visible in the network. In our NLP mentor-mentee network, the average degree for male researcher nodes is 3.356, while for females it is 3.186. Hence our mentor-mentee network exhibits a "rich-get-richer" mechanism in favour of male researchers.

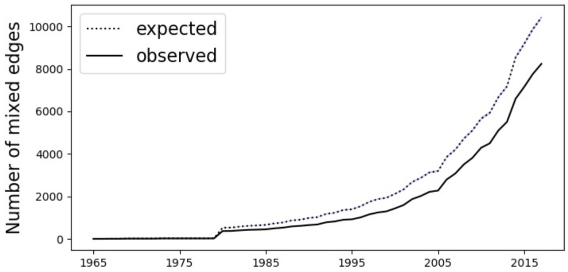

Figure 4: Evidence of ever-increasing homophily. The expected number of mixed-gender interactions is higher than the observed.

The biased preferential attachment model. Avin et al. (2015) extend Barabási and Albert (1999)'s preferential attachment model that was originally based on the "rich-get-richer" feedback mechanism to a *biased preferential attachment model* of mentor-mentee dynamics, $G(n, f, p)$, where there further is (1) a minority-majority partition (the proportion of female nodes is less than half, $f < \frac{1}{2}$) and (2) homophily. The model works as follows, instantiated to our context. Over time, a sequence of bi-populated mentor-mentee networks is constructed, $G_t = (V_t, E_t)$, like the one described in Section 2. $V_t = F_t \cup M_t$ is the set of G_t's nodes, and E_t its edges, where $F_t(M_t)$ is the set of female (male) nodes. G_0 is the empty graph. At each time $t > 0$ a mentee enters the network. The mentee is a female with probability f and a male with probability $m = 1 - f$. Assuming a rich-get-richer mechanism, the mentee chooses a potential mentor according to that mentor's importance in the network: with probability $\frac{\delta_t(u)}{\sum_{v \in V_t} \delta_t(v)}$ where $\delta_t(v)$ is the degree of $v \in V_t$. If this supervision is in-gender, then a relation (edge) is established. However if genders differ, then the relation (edge) is established according to the probability of homophily (p); otherwise (with probability $(1 - p)$) it is rejected and the mentee must restart the process of finding a mentor. Once an advisor for the mentee has been found, t increments to the next time step.

We now introduce definitions and the main theorem established by (Avin et al., 2015) for conditions of the existence of power inequality and a glass ceiling effect in bi-populated networks. Then we empirically check for these conditions in our NLP mentor-mentee network for the main result.

Power inequality definition. The sequence of mentor-mentee networks G_t is said to exhibit a *power inequality effect* for females if the average power of a female node is strictly bounded by the power of a male node: i.e., $\lim_{t \to \infty} \frac{\frac{1}{|F_t|} \sum_{v \in F_t} \delta_t(v)}{\frac{1}{|M_t|} \sum_{v \in M_t} \delta_t(v)} < 1$.

Tail and moment glass ceiling definitions. Let $\text{top}_k(F_t)$ ($\text{top}_k(M_t)$) denote the number of female (male) nodes that have degree of at least k in G_t– this is the group of scholars whose wealth in relations in the network is at level at least k; this wealth of relations is a form of power. The glass ceiling effect for the minority of females describes a process by which the proportion of access to this wealth of relations is limited for females but not for males. Formally, the sequence G_t is said to exhibit a *tail glass ceiling effect* for the female nodes (the minority) if there exists an increasing sequence k_t such that $\lim_{t \to \infty} \text{top}_{k_t}(M_t) = \infty$ and $\lim_{t \to \infty} \frac{\text{top}_{k_t}(F_t)}{\text{top}_{k_t}(M_t)} = 0$. G_t exhibits a *moment glass ceiling* g for the female nodes, if $g = \lim_{t \to \infty} \frac{\frac{1}{|F_t|} \sum_{v \in F_t} \delta_t(v)^2}{\frac{1}{|M_t|} \sum_{v \in M_t} \delta_t(v)^2}$. And if $g = 0$, G_t has a *strong glass ceiling effect*.

The main result: Power inequality and glass ceiling. Avin et al. (2015) proved that if $0 < f < \frac{1}{2}$ and $0 < p < 1$, then for $G(n, f, p)$ produced by the biased preferential attachment model, $G(n, f, p)$ exhibits both power inequality and a tail and strong glass ceiling effects. In observations (O2) and (O3), we identified the conditions $f = 0.335 < 0.5$ and the existence of homophily (i.e., $0 < p < 1$) in our NLP mentor-mentee network. We have therefore shown there to exist power inequality and a glass ceiling in NLP.

5 Concluding remarks

Given our study of the mentee-mentor network for NLP, we have shown that there is a glass ceiling for female researchers in NLP that has taken a hold of the field since the mid-2000s.

References

Chen Avin, Barbara Keller, Zvi Lotker, Claire Mathieu, David Peleg, and Yvonne-Anne Pignolet. 2015. Homophily and the glass ceiling effect in social networks. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*. ACM, ITCS ’15, pages 41–50.

Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286(5439):509–512.

David Easley and Jon Kleinberg. 2010. *Networks, Crowds and Markets: Reasoning about a Highly Connected World*. Cambridge University Press.

Patrick Gaule and Mario Piacentini. 2018. An advisor like me? advisor gender and post-graduate careers in science. *Research Policy* (47):805–813.

Vincent Larivière, Chaoqun Ni, Yves Gingras, Blaise Cronin, and Cassidy R Sugimoto. 2013. Bibliometrics: Global gender disparities in science. *Nature News* 504(7479):211.

Corinne A Moss-Racusin, John F Dovidio, Victoria L Brescoll, Mark J Graham, and Jo Handelsman. 2012. Science facultys subtle gender biases favor male students. *Proceedings of the National Academy of Sciences* 109(41):16474–16479.

Linda J. Sax, Kathleen J. Lehman, Jerry A. Jacobs, M. Allison Kanny, Gloria Lim, Laura Monje-Paulson, and Hilary B. Zimmerman. 2017. Anatomy of an enduring gender gap: The evolution of womens participation in computer science. *The Journal of Higher Education* 88(2):258–293. https://doi.org/10.1080/00221546.2016.1257306.

Helen Shen. 2013. Mind the gender gap. *Nature* 495(7439):22.

Romy Van der Lee and Naomi Ellemers. 2015. Gender contributes to personal research funding success in the netherlands. *Proceedings of the National Academy of Sciences* 112(40):12349–12353.

Samuel F Way, Daniel B Larremore, and Aaron Clauset. 2016. Gender, productivity, and prestige in computer science faculty hiring networks. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 1169–1179.

Wendy M. Williams, Agrima Mahajan, Felix Thoemmes, Susan M. Barnett, Francoise Vermeylen, Brian M. Cash, and Stephen J. Ceci. 2017. Does gender of administrator matter? national study explores u.s. university administrators’ attitudes about retaining women professors in stem. *Frontiers in Psychology* 8:15.

Reducing Gender Bias in Abusive Language Detection

Ji Ho Park, Jamin Shin, Pascale Fung
Centre for Artificial Intelligence Research (CAiRE)
Hong Kong University of Science and Technology
`{jhpark, jmshinaa}@connect.ust.hk`, `pascale@ece.ust.hk`

Abstract

Abusive language detection models tend to have a problem of being biased toward identity words of a certain group of people because of imbalanced training datasets. For example, "You are a good *woman*" was considered "sexist" when trained on an existing dataset. Such model bias is an obstacle for models to be robust enough for practical use. In this work, we measure gender biases on models trained with different abusive language datasets, while analyzing the effect of different pre-trained word embeddings and model architectures. We also experiment with three bias mitigation methods: (1) debiased word embeddings, (2) gender swap data augmentation, and (3) fine-tuning with a larger corpus. These methods can effectively reduce gender bias by 90-98% and can be extended to correct model bias in other scenarios.

1 Introduction

Automatic detection of abusive language is an important task since such language in online space can lead to personal trauma, cyber-bullying, hate crime, and discrimination. As more and more people freely express their opinions in social media, the amount of textual contents produced every day grows almost exponentially, rendering it difficult to effectively moderate user content. For this reason, using machine learning and natural language processing (NLP) systems to automatically detect abusive language is useful for many websites or social media services.

Although many works already tackled on training machine learning models to automatically detect abusive language, recent works have raised concerns about the robustness of those systems. Hosseini et al. (2017) have shown how to easily cause false predictions with adversarial examples in Google's API, and Dixon et al. (2017) show that classifiers can have unfair biases toward certain groups of people.

We focus on the fact that the representations of abusive language learned in only supervised learning setting may not be able to generalize well enough for practical use since they tend to overfit to certain words that are neutral but occur frequently in the training samples. To such classifiers, sentences like "You are a good woman" are considered "sexist" probably because of the word "woman."

This phenomenon, called *false positive bias*, has been reported by Dixon et al. (2017). They further defined this model bias as unintended, "a model contains unintended bias if it performs better for comments containing some particular identity terms than for comments containing others."

Such model bias is important but often unmeasurable in the usual experiment settings since the validation/test sets we use for evaluation are already biased. For this reason, we tackle the issue of measuring and mitigating unintended bias. Without achieving certain level of generalization ability, abusive language detection models may not be suitable for real-life situations.

In this work, we address model biases specific to gender identities (gender bias) existing in abusive language datasets by measuring them with a generated unbiased test set and propose three reduction methods: (1) debiased word embedding, (2) gender swap data augmentation, (3) fine-tuning with a larger corpus. Moreover, we compare the effects of different pre-trained word embeddings and model architectures on gender bias.

2 Related Work

So far, many efforts were put into defining and constructing abusive language datasets from different sources and labeling them through crowd-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2799–2804
Brussels, Belgium, October 31 - November 4, 2018.

sourcing or user moderation (Waseem and Hovy, 2016; Waseem, 2016; Founta et al., 2018; Wulczyn et al., 2017). Many deep learning approaches have been explored to train a classifier with those datasets to develop an automatic abusive language detection system (Badjatiya et al., 2017; Park and Fung, 2017; Pavlopoulos et al., 2017). However, these works do not explicitly address any model bias in their models.

Addressing biases in NLP models/systems have recently started to gain more interest in the research community, not only because fairness in AI is important but also because bias correction can improve the robustness of the models. Bolukbasi et al. (2016) is one of the first works to point out the gender stereotypes inside word2vec (Mikolov et al., 2013) and propose an algorithm to correct them. Caliskan et al. (2017) also propose a method called Word Embedding Association Test (WEAT) to measure model bias inside word embeddings and finds that many of those pretrained embeddings contain problematic bias toward gender or race. Dixon et al. (2017) is one of the first works that point out existing "unintended" bias in abusive language detection models. Kiritchenko and Mohammad (2018) compare 219 sentiment analysis systems participating in SemEval competition with their proposed dataset, which can be used for evaluating racial and gender bias of those systems. Zhao et al. (2018) shows the effectiveness of measuring and correcting gender biases in coreference resolution tasks. We later show how we extend a few of these works into ours.

3 Datasets

3.1 Sexist Tweets (`st`)

This dataset consists of tweets with sexist tweets collected from Twitter by searching for tweets that contain common terms pertaining to sexism such as "feminazi." The tweets were then annotated by experts based on criteria founded in critical race theory. The original dataset also contained a relatively small number of "racist" label tweets, but we only retain "sexist" samples to focus on gender biases. Waseem and Hovy (2016); Waseem (2016), the creators of the dataset, describe "sexist" and "racist" languages as specific subsets of abusive language.

Name	Size	Positives (%)	μ	σ	max
`st`	18K	33%	15.6	6.8	39
`abt`	60K	18.5%	17.9	4.6	65

Table 1: Dataset statistics. μ, σ, max are mean, std.dev, and maximum of sentence lengths

3.2 Abusive Tweets (`abt`)

Recently, Founta et al. (2018) has published a large scale crowdsourced abusive tweet dataset with 60K tweets. Their work incrementally and iteratively investigated methods such as boosted sampling and exploratory rounds, to effectively annotate tweets through crowdsourcing. Through such systematic processes, they identify the most relevant label set in identifying abusive behaviors in Twitter as $\{None, Spam, Abusive, Hateful\}$ resulting in 11% as 'Abusive,' 7.5% as 'Hateful', 22.5% as 'Spam', and 59% as 'None'. We transform this dataset for a binary classification problem by concatenating 'None'/'Spam' together, and 'Abusive'/'Hateful' together.

4 Measuring Gender Biases

4.1 Methodology

Gender bias cannot be measured when evaluated on the original dataset as the test sets will follow the same biased distribution, so normal evaluation set will not suffice. Therefore, we generate a separate *unbiased test set* for each gender, male and female, using the identity term template method proposed in Dixon et al. (2017).

The intuition of this template method is that given a pair of sentences with only the identity terms different (ex. "He is happy" & "She is happy"), the model should be able to generalize well and output same prediction for abusive language. This kind of evaluation has also been performed in *SemEval 2018: Task 1 Affect In Tweets* (Kiritchenko and Mohammad, 2018) to measure the gender and race bias among the competing systems for sentiment/emotion analysis.

Using the released code[1] of Dixon et al. (2017), we generated 1,152 samples (576 pairs) by filling the templates with common gender identity pairs (ex. male/female, man/woman, etc.). We created templates (Table 2) that contained both neutral and offensive nouns and adjectives inside the vocabu-

---

[1] `https://github.com/conversationai/unintended-ml-bias-analysis`

Example Templates
You are a *(adjective) (identity term).*
(verb) (identity term).
Being *(identity term)* is *(adjective)*
I am *(identity term)*
I hate *(identity term)*

Table 2: Example of templates used to generated an unbiased test set.

Type	Example Words
Offensive	disgusting, filthy, nasty, rude, horrible, terrible, awful, worst, idiotic, stupid, dumb, ugly, etc.
Non-offensive	help, love, respect, believe, congrats, hi, like, great, fun, nice, neat, happy, good, best, etc.

Table 3: Example of offensive and non-offensive verbs & adjectives used for generating the unbiased test set.

lary (See Table 3) to retain balance in neutral and abusive samples.

For the evaluation metric, we use 1) AUC scores on the original test set (Orig. AUC), 2) AUC scores on the unbiased generated test set (Gen. AUC), and 3) the false positive/negative equality differences proposed in Dixon et al. (2017) which aggregates the difference between the overall false positive/negative rate and gender-specific false positive/negative rate. False Positive Equality Difference (FPED) and False Negative Equality Difference (FNED) are defined as below, where $T = \{male, female\}$.

$$FPED = \sum_{t \in T} |FPR - FPR_t|$$

$$FNED = \sum_{t \in T} |FNR - FNR_t|$$

Since the classifiers output probabilities, equal error rate thresholds are used for prediction decision.

While the two AUC scores show the performances of the models in terms of accuracy, the equality difference scores show them in terms of fairness, which we believe is another dimension for evaluating the model's generalization ability.

4.2 Experimental Setup

We first measure gender biases in `st` and `abt` datasets. We explore three neural models used in previous works on abusive language classification: Convolutional Neural Network (CNN) (Park and Fung, 2017), Gated Recurrent Unit (GRU) (Cho et al., 2014), and Bidirectional GRU with self-attention (α-GRU) (Pavlopoulos et al., 2017), but with a simpler mechanism used in Felbo et al. (2017). Hyperparameters are found using the validation set by finding the best performing ones in terms of original AUC scores. These are the used hyperparameters:

1. CNN: Convolution layers with 3 filters with the size of [3,4,5], feature map size=100, Embedding Size=300, Max-pooling, Dropout=0.5
2. GRU: hidden dimension=512, Maximum Sequence Length=100, Embedding Size=300, Dropout=0.3
3. α-GRU: hidden dimension=256 (bidirectional, so 512 in total), Maximum Sequence Length=100, Attention Size=512, Embedding Size=300, Dropout=0.3

We also compare different pre-trained embeddings, `word2vec` (Mikolov et al., 2013) trained on Google News corpus, `FastText` (Bojanowski et al., 2017)) trained on Wikipedia corpus, and randomly initialized embeddings (*random*) to analyze their effects on the biases. Experiments were run 10 times and averaged.

Model	Embed.	Orig. AUC	Gen. AUC	FNED	FPED
CNN	random	.881	.572	.261	.249
	fasttext	**.906**	.620	.323	.327
	word2vec	**.906**	.635	.305	.263
GRU	random	.854	.536	**.132**	**.136**
	fasttext	.887	**.661**	.312	.284
	word2vec	.887	.633	.301	.254
α-GRU	random	.868	.586	.236	.219
	fasttext	.891	.639	.324	.365
	word2vec	.890	.631	.315	.306

Table 4: Results on `st`. False negative/positive equality differences are larger when pre-trained embedding is used and CNN or α-RNN is trained

4.3 Results & Discussions

Tables 4 and 5 show the bias measurement experiment results for `st` and `abt`, respectively. As expected, pre-trained embeddings improved task performance. The score on the unbiased generated test set (Gen. ROC) also improved since word embeddings can provide prior knowledge of words.

However, the equality difference scores tended to be larger when pre-trained embeddings were

Model	Embed.	Orig. AUC	Gen. AUC	FNED	FPED
CNN	random	.926	.893	.013	.045
	fasttext	.955	.995	.004	**.001**
	word2vec	**.956**	**.999**	**.002**	.021
GRU	random	.919	.850	.036	.010
	fasttext	.951	.997	.014	.018
	word2vec	.952	.997	.017	.037
α-GRU	random	.927	.914	.008	.039
	fasttext	**.956**	.998	.014	.005
	word2vec	.955	**.999**	.012	.026

Table 5: Results on `abt`. The false negative/positive equality difference is significantly smaller than the `st`

used, especially in the `st` dataset. This confirms the result of Bolukbasi et al. (2016). In all experiments, direction of the gender bias was towards female identity words. We can infer that this is due to the more frequent appearances of female identities in "sexist" tweets and lack of negative samples, similar to the reports of Dixon et al. (2017). This is problematic since not many NLP datasets are large enough to reflect the true data distribution, more prominent in tasks like abusive language where data collection and annotation are difficult.

On the other hand, `abt` dataset showed significantly better results on the two equality difference scores, of at most 0.04. Performance in the generated test set was better because the models successfully classify abusive samples regardless of the gender identity terms used. Hence, we can assume that `abt` dataset is less gender-biased than the `st` dataset, presumably due to its larger size, balance in classes, and systematic collection method.

Interestingly, the architecture of the models also influenced the biases. Models that "attend" to certain words, such as CNN's max-pooling or α-GRU's self-attention, tended to result in higher false positive equality difference scores in `st` dataset. These models show effectiveness in catching not only the discriminative features for classification, but also the "unintended" ones causing the model biases.

5 Reducing Gender Biases

We experiment and discuss various methods to reduce gender biases identified in Section 4.3.

Model	DE	GS	FT	Orig. AUC	Gen. AUC	FNED	FPED
CNN	.	.	.	**.906**	.635	.305	.263
	O	.	.	.902	.627	.333	.337
	.	O	.	.898	.676	.164	.104
	O	O	.	.895	.647	.157	.096
	.	.	O	.896	.650	.302	.240
	.	O	O	.889	.671	.163	.122
	O	O	O	.884	.703	.135	.095
GRU	.	.	.	.887	.633	.301	.254
	O	.	.	.882	.658	.274	.270
	.	O	.	.879	.657	.044	.040
	O	O	.	.873	.667	**.006**	**.027**
	.	.	O	.874	.761	.241	.181
	.	O	O	.862	.768	.141	.095
	O	O	O	.854	.854	.081	.059
α-GRU	.	.	.	.890	.631	.315	.306
	O	.	.	.885	.656	.291	.330
	.	O	.	.879	.667	.114	.098
	O	O	.	.877	.689	.067	.059
	.	.	O	.874	.756	.310	.212
	.	O	O	.866	.814	.185	.065
	O	O	O	.855	**.912**	.055	.030

Table 6: Results of bias mitigation methods on `st` dataset. 'O' indicates that the corresponding method is applied. See Section 5.3 for more analysis.

5.1 Methodology

Debiased Word Embeddings (DE) (Bolukbasi et al., 2016) proposed an algorithm to correct word embeddings by removing gender stereotypical information. All the other experiments used pretrained word2vec to initialized the embedding layer but we substitute the pretrained word2vec with their published embeddings to verify their effectiveness in our task.

Gender Swap (GS) We augment the training data by identifying male entities and swapping them with equivalent female entities and vice-versa. This simple method removes correlation between gender and classification decision and has proven to be effective for correcting gender biases in coreference resolution task (Zhao et al., 2018).

Bias fine-tuning (FT) We propose a method to use transfer learning from a less biased corpus to reduce the bias. A model is initially trained with a larger, less-biased source corpus with a same or similar task, and fine-tuned with a target corpus with a larger bias. This method is inspired by the fact that model bias mainly rises from the imbalance of labels and the limited size of data samples. Training the model with a larger and less biased dataset may regularize and prevent the model from over-fitting to the small, biased dataset.

5.2 Experimental Setup

Debiased word2vec Bolukbasi et al. (2016) is compared with the original *word2vec* (Mikolov et al., 2013) for evaluation. For gender swapping data augmentation, we use pairs identified through crowd-sourcing by Zhao et al. (2018).

After identifying the degree of gender bias of each dataset, we select a source with less bias and a target with more bias. Vocabulary is extracted from training split of both sets. The model is first trained by the source dataset. We then remove final softmax layer and attach a new one initialized for training the target. The target is trained with a slower learning rate. Early stopping is decided by the valid set of the respective dataset.

Based on this criterion and results from Section 4.3, we choose the `abt` dataset as source and `st` dataset as target for bias fine-tuning experiments.

5.3 Results & Discussion

Table 6 shows the results of experiments using the three methods proposed. The first rows are the baselines without any method applied. We can see from the second rows of each section that debiased word embeddings alone do not effectively correct the bias of the whole system that well, while gender swapping significantly reduced both the equality difference scores. Meanwhile, fine-tuning bias with a larger, less biased source dataset helped to decrease the equality difference scores and greatly improve the AUC scores from the generated unbiased test set. The latter improvement shows that the model significantly reduced errors on the unbiased set in general.

To our surprise, the most effective method was applying both debiased embedding and gender swap to GRU, which reduced the equality differences by 98% & 89% while losing only 1.5% of the original performance. We assume that this may be related to the influence of "attending" model architectures on biases as discussed in Section 4.3. On the other hand, using the three methods together improved both generated unbiased set performance and equality differences, but had the largest decrease in the original performance.

All methods involved some performance loss when gender biases were reduced. Especially, fine-tuning had the largest decrease in original test set performance. This could be attributed to the difference in the source and target tasks (abusive & sexist). However, the decrease was marginal (less than 4%), while the drop in bias was significant. We assume the performance loss happens because mitigation methods modify the data or the model in a way that sometimes deters the models from discriminating important "unbiased" features.

6 Conclusion & Future Work

We discussed model biases, especially toward gender identity terms, in abusive language detection. We found out that pre-trained word embeddings, model architecture, and different datasets all can have influence. Also, we found our proposed methods can reduce gender biases up to 90-98%, improving the robustness of the models.

As shown in Section 4.3, some classification performance drop happens when mitigation methods. We believe that a meaningful extension of our work can be developing bias mitigation methods that maintain (or even increase) the classification performance and reduce the bias at the same time. Some previous works (Beutel et al.; Zhang et al., 2018) employ adversarial training methods to make the classifiers unbiased toward certain variables. However, those works do not deal with natural language where features like gender and race are latent variables inside the language. Although those approaches are not directly comparable to our methods, it would be interesting to explore adversarial training to tackle this problem in the future.

Although our work is preliminary, we hope that our work can further develop the discussion of evaluating NLP systems in different directions, not merely focusing on performance metrics like accuracy or AUC. The idea of improving models by measuring and correcting gender bias is still unfamiliar but we argue that they can be crucial in building systems that are not only ethical but also practical. Although this work focuses on gender terms, the methods we proposed can easily be extended to other identity problems like racial and to different tasks like sentiment analysis by following similar steps, and we hope to work on this in the future.

Acknowledgments

This work is partially funded by ITS/319/16FP of Innovation Technology Commission, HKUST, and 16248016 of Hong Kong Research Grants Council.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. Data decisions and theoretical implications when adversarially learning fair representations. *FAT/ML 2018: 5th Workshop on Fairness, Accountability, and Transparency in Machine Learning*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics Volume 5, Issue 1*.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP2014*.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2017. Measuring and mitigating unintended bias in text classification. In *AAAI*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *EMNLP2017*.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. *AAAI*.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

Svetlana Kiritchenko and Saif M Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. *Proceedings of the 7th Joint Conference on Lexical and Computational Semantics(*SEM), New Orleans, USA*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. *ALW1: 1st Workshop on Abusive Language Online to be held at the annual meeting of the Association of Computational Linguistics (ACL) 2017*.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deeper attention to abusive user content moderation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on NLP and computational social science*, pages 138–142.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399. International World Wide Web Conferences Steering Committee.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. *Proceedings of AAAI/ACM Conference on Ethics and Society(AIES) 2018*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. *NAACL 2018*.

SafeCity: Understanding Diverse Forms of Sexual Harassment Personal Stories

Sweta Karlekar **Mohit Bansal**
UNC Chapel Hill
{swetakar, mbansal}@cs.unc.edu

Abstract

With the recent rise of #MeToo, an increasing number of personal stories about sexual harassment and sexual abuse have been shared online. In order to push forward the fight against such harassment and abuse, we present the task of automatically categorizing and analyzing various forms of sexual harassment, based on stories shared on the online forum SafeCity. For the labels of groping, ogling, and commenting, our single-label CNN-RNN model achieves an accuracy of 86.5%, and our multi-label model achieves a Hamming score of 82.5%. Furthermore, we present analysis using LIME, first-derivative saliency heatmaps, activation clustering, and embedding visualization to interpret neural model predictions and demonstrate how this helps extract features that can help automatically fill out incident reports, identify unsafe areas, avoid unsafe practices, and 'pin the creeps'.

1 Introduction

The hashtag #MeToo[1] has been prevalent on various social media platforms as a campaign centered around sharing stories of sexual harassment in an act of solidarity with other victims and spreading awareness of a widespread and endemic issue. With vast amounts of personal stories on the internet, it is important that we make scientific use of this data to push these movements forward and enable real-world change. Manually sorting and comprehending the information shared in these stories is an arduous task, and the power of natural language processing (NLP) can serve as the missing link between online activism and real change.

We present several neural NLP models that allow us to automatically classify, aggregate, and analyze vast amounts of harassment data found on social media, becoming an effective tool for spreading awareness, increasing understanding, and allowing faster action. This large-scale automatic categorization, summarization, and analysis of personal abuse stories can help activist groups enlighten the public and advocate for social change in a timely manner.

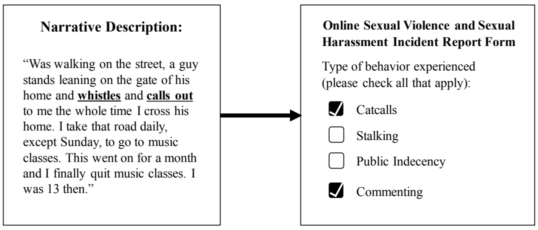

Narrative Description:

"Was walking on the street, a guy stands leaning on the gate of his home and **whistles** and **calls out** to me the whole time I cross his home. I take that road daily, except Sunday, to go to music classes. This went on for a month and I finally quit music classes. I was 13 then."

Online Sexual Violence and Sexual Harassment Incident Report Form

Type of behavior experienced (please check all that apply):

☑ Catcalls
☐ Stalking
☐ Public Indecency
☑ Commenting

Figure 1: Task of sexual harassment story classification to help fill online incident reports.

We present single-label and multi-label classification of diverse forms of sexual harassment present in abuse stories shared online through the forum SafeCity, a crowd-sourcing platform for personal stories of sexual harassment and abuse. Each story includes one or more tagged forms of sexual harassment, along with a description of the occurrence. For example, the description "*My college was nearby. This happened all the time. Guys passing comments, staring, trying to touch. Frustrating*" is positive for three classes: commenting, ogling/staring, and touching/groping.

We use CNN-RNN architectures (with character-level CNN embeddings and bidirectional RNNs) to classify the three forms of sexual harassment mentioned above using both single- and multi-label setups. Our models achieve strong performances of 80-86% on these setups. This automatic classification of different forms of sexual harassment can help victims and authorities to partially automate and speed up the process of filling online sexual violence reporting forms (see Figure 1), which usually requires the victim to detail each form of sexual harassment that took place. The act of partially filling out the report

---

[1] https://metoomvmt.org

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2805–2811
Brussels, Belgium, October 31 - November 4, 2018.

(by our classifier) in itself makes it more likely for the victim to file a report. A study by the Bureau of Justice found that victims who report sexual assault are more likely to seek medical treatment for injuries, which also allows for more immediate prosecution and a better chance of finding DNA evidence to convict the offender (Rennison, 2002). Further, it can also be used to fulfill the need to automatically categorize and summarize large numbers of online testimonials describing or reporting sexual harassment.

Next, in order to further utilize these stories as an important tool for harassment understanding and to help prevent similar situations from happening to others, we present interpretability analysis of our neural classification results in the forms of LIME analysis, first-derivative saliency heatmaps, activation clustering, and t-SNE embedding visualization. We show how these analysis techniques hold promise as avenues for future work and can potentially provide insightful clues towards building (1) a tool to analyze the most common circumstances around each distinct form of harassment to provide more detailed and accurate safety advice, (2) a map of unsafe areas to help others avoid dangerous spaces, and 3) an unofficial sex offender registry that marks frequently-mentioned offenders to warn potential victims. This paper seeks to provide an avenue to utilize the millions of stories shared on social media describing instances of sexual harassment, including #MeToo, #WhyILeft, and #YesAllWomen. With this task and analysis, we hope that these stories can be used to prevent future sexual harassment.

2 Related Work

Analyzing personal sexual harassment stories from online social forums is fairly unexplored, to the best of our knowledge. However, recent works in a similar vein include detecting the presence of domestic abuse stories on social media sites (Schrading et al., 2015a; Schrading, 2015; Schrading et al., 2015b). In more distantly related work, NLP has been used for various socially-driven tasks, such as detecting the presence of cyberbullying or incivility (Ziegele et al., 2018; Founta et al., 2018; Chen et al., 2012; Zhao et al., 2016; Agrawal and Awekar, 2018; Van Hee et al., 2018), and detecting and providing aid for signs of depression or suicidal thoughts (Pestian et al., 2010; Yazdavar et al., 2017; Stepanov et al., 2017; Fitzpatrick et al., 2017).

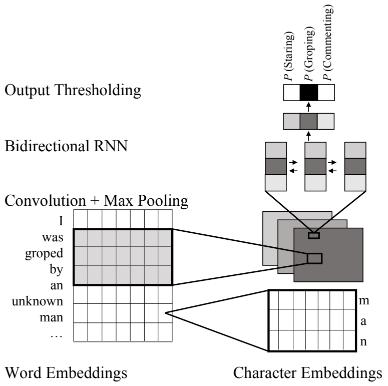

Figure 2: Multi-label CNN-RNN model with CNN-based character embeddings and bidirectional RNNs.

3 Classification Models

For our single-label binary classification task, the two output classes can be [commenting, non-commenting], [ogling, non-ogling], or [groping, non-groping]. For our multi-label scenario, there are a total of 8 combinations (true or false for three types of sexual harassment), including a label for none of the three classes present in the description.

CNN: For each input description, an embedding and convolutional layer are applied. This is followed by a max-pooling layer (Collobert et al., 2011). Filters of varying window sizes are applied to each window of word vectors, the result of which is then passed through a softmax layer to produce probabilities over the output classes.

LSTM-RNN: As CNNs are not designed to capture sequential relationships (Pascanu et al., 2014), we adopted an RNN model that consisted of word vectors fed into LSTM layer, the final state of which was fed into a fully-connected layer. The result is passed through a softmax layer to output the probability over all output classes.

CNN-RNN: As both models have strengths and weaknesses, we experimented with a hybrid architecture in which our LSTM-RNN model after the embedding layer is laid on top of our CNN model before the max-pooling (related to Zhou et al. (2015)). For single-label models, the final fully-connected layer is fed into a softmax to give final output probabilities.

Multi-Label Classification We also present multi-label classification (Boutell et al., 2004; Tsoumakas and Katakis, 2006; Katakis et al., 2008), which allows for models to predict multiple categories simultaneously for the same input. We further utilized CNN-based character embeddings in addition to word embeddings, and also employed bidirectional RNNs (see Figure 2). The outputs of the final fully-connected layer (F) are fed into a sigmoid function. The classification for each category (C) are seen as positive (1) if the output is above threshold *t* and negative (0) if the output is below threshold *t*, a hyperparameter, giving the equation: $C = \mathbb{1}(\sigma(F) \geq t)$.

4 Experimental Setup

4.1 Dataset

SafeCity[2] is, to the best of our knowledge, the largest publicly-available online forum for reporting sexual harassment. Its motto is "pin the creeps". Victims of sexual harassment share personal stories, with the objective of spreading awareness of ongoing sexual harassment and showcasing location-based trends. The language styles of SafeCity forums are very diverse, and therefore can potentially be used for a variety of test cases, such as emails or tweets.

Each of the 9,892 stories includes a description of the incident, the location, and tagged forms of harassment, with all identifying information removed. SafeCity has explicitly given us permission to use this data. The dataset[3] contains descriptions of text submitted by forum users, along with tags of 13 forms of sexual harassment. We chose the top three most dense categories—groping/touching, staring/ogling, and commenting—to use as our dataset, as the others were more sparse. Each description may fall into none, some, or all of the categories.

4.2 Evaluation

The single-label models were evaluated using accuracy. The multi-label models were evaluated using exact match ratio and Hamming score (calculated as the complement of Hamming loss). Hamming loss was used as detailed by Tsoumakas and Katakis (2006). Hamming loss (y) is equal to 1 over $|D|$ (number of multi-label samples), multiplied by the sum of the symmetric differences between the predictions (Z) and the true labels (Y), divided by the number of labels (L), giving $y = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \Delta Z_i|}{|L|}$.

Model	Commenting	Ogling	Groping
Linear SVM	42.2	35.0	55.8
Gaussian NB	46.8	74.7	66.0
Logistic Reg.	61.4	78.0	69.1
SVM	65.5	79.0	70.3
CNN	80.9	82.2	86.0
RNN	81.0	82.2	86.2
CNN-RNN	**81.6**	**84.1**	**86.5**

Table 1: Single-label classification (accuracy) results.

Model	Exact Match	Hamming
Random Forest	35.0	70.2
CNN	53.7	80.2
RNN	57.1	81.5
CNN-RNN	59.2	82.3
CNN-RNN (bidirec + char)	**62.0**	**82.5**

Table 2: Multi-label classification results.

4.3 Training Details

All models have vocabulary size of $10,000$, and use AdamOptimizer (Kingma and Ba, 2015) with a learning rate of $1e^{-4}$. All gradient norms are clipped to 2.0 (Pascanu et al., 2013; Graves, 2013). For each model, the hyperparameters are tuned using the development set.

CNN We use a 2-D CNN. Filter sizes of [3, 4, 5] are used with 128 filters per filter size. Batch size is set to 128, and a dropout (Srivastava et al., 2014) of 0.80 is applied.

LSTM Our LSTM has 2 layers with 60 hidden units. Batch size is 64 with a dropout of 0.75.

CNN-LSTM Our CNN-LSTM model consists of an LSTM on top of a CNN. The CNN has 100 filters per filter size of [3, 4, 5]. Embedding dimensions of 300 are used. An LSTM with 300 hidden units is used. For the character level embeddings, we use an additional CNN with 100 filters per filter size of [3, 4, 5]. Bidirectional RNNs of 300 units are used.

5 Results

See Table 1 for single-label results on the selected harassment categories, where CNN-RNN was the best performing model compared to several non-neural and neural baselines. See Table 2 for multi-label classification results, where the Hamming score for the multi-label CNN-RNN model is 82.5%, showing potential for real-world use as well as substantial future research scope.

6 Analysis

We provide various visualization techniques to analyze our models. Each of these techniques employs a different approach and offers new information or supports previous findings.

---

[2] `http://safecity.in`

[3] We release our dataset splits at `https://github.com/swkarlekar/safecity`. Please follow SafeCity guidelines for usage.

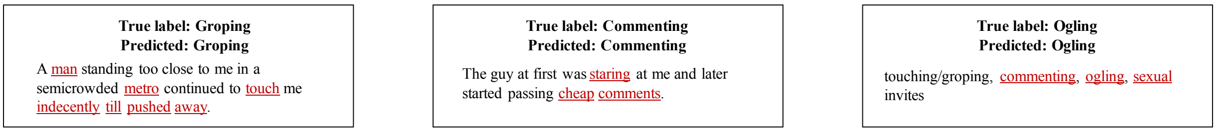

True label: Groping
Predicted: Groping

A man standing too close to me in a semicrowded metro continued to touch me indecently till pushed away.

True label: Commenting
Predicted: Commenting

The guy at first was staring at me and later started passing cheap comments.

True label: Ogling
Predicted: Ogling

touching/groping, commenting, ogling, sexual invites

Figure 3: LIME analysis. **Left** : Correctly-classified example of groping. **Middle**: Correctly-classified example of commenting. **Right**: Correctly-classified example of ogling/staring.

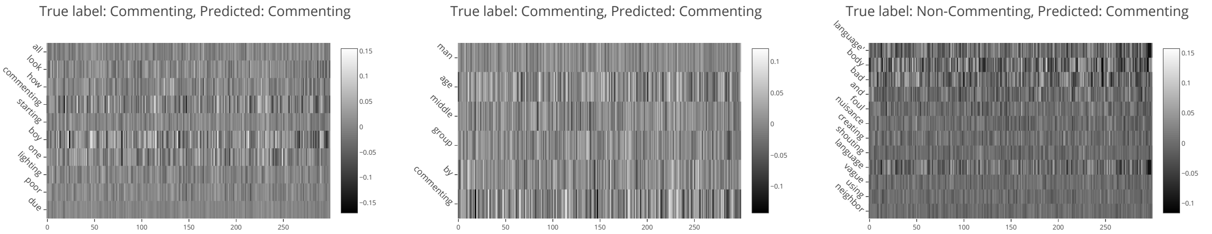

Figure 4: First derivative saliency heatmaps. **Left**: Correctly-classified example of commenting. **Middle**: Correctly-classified example of commenting. **Right**: Incorrectly-classified example of commenting.

Harassment Type	Observed t-SNE Cluster
Groping	abusively, encounter, talk, underage, surrounded, embarrassed
Ogling	boobs, leering, disturbing, gestures, voyeur, visually, gestures
Commenting	shameful, vulgar, inappropriately, indecent, invites, stalked, strong

Table 3: Relevant word clusters found via t-SNE word embedding clustering.

6.1 Word Embedding Visualization

We selected seed words that corresponded to class labels and found the nearest neighbors of each seed word's vector by reducing the dimensionality of the word embeddings using t-SNE (see Table 3) (Maaten and Hinton, 2008). This form of visualization not only ensures that our model has learned appropriate word embeddings, but also demonstrates that each form of sexual harassment has a unique and distinct context. Furthermore, this shows that our model learns related words and concepts for each type of harassment.

6.2 LIME Analysis

LIME analysis (Ribeiro et al., 2016), or Local Interpretable Model-Agnostic Explanation, interprets the local reasoning of a model around an instance. Results of LIME (ξ) are found by taking the minimum of $\mathcal{L}$, which is the measure of how unfaithful the interpretable model (g) is to approximating the probability that an input (x) belongs to a certain class (f) in the locally defined area (π_x) summed with complexity measures Ω, giving $\xi(x) = argmin\ \mathcal{L}(f, g, \pi_x) + \Omega(g)$. In Figure 3 (left), the words "touch", "man", and the collective words "indecently till pushed away" are the most important to the local classification of "groping". Furthermore, the word "metro" has importance in the classification, suggesting that this may be a frequent location in which groping takes place. In Figure 3 (middle), the words with the most importance are "comments" and "staring", indicating that ogling may coincide with commenting very frequently. In Figure 3 (right), the words "ogling", "sexual", and "commenting" had the most importance, which further supports the notion that ogling and commenting often occur together. As verified by the data, ogling and commenting together is more common than ogling alone.

6.3 First Derivative Saliency

Saliency heatmaps (Simonyan et al., 2014; Li et al., 2016) illustrate which words of an input have the biggest impact on the final classification by taking the gradient of the final scores outputted by the neural network (S) with respect to the embedding (E), given the true label (L), giving $\frac{\partial S_L(E)}{\partial E}$. While LIME analysis and first derivative saliency are both used to find word-level contributions, first derivative saliency is model-dependent and gives reasoning behind classification based on the whole model, in contrast to the locally-faithful, model-agnostic LIME analysis technique.

In Figure 4 (left), the word "commenting" and the words "one boy" have the most influence on the classification. The influence of the word "lighting" indicates poor lighting is often present in situations where sexual harassment takes place. In Figure 4 (middle), the classification of "commenting" was most influenced by the word "commenting", followed by the word "age". This suggests the possibility of using descriptors of offenders as a classification tool. Figure 4 (right) is an incorrectly classified example. We see that the word "body", followed by "language", had the most influence on the classification of this example as “commenting”. Our model identifies synonyms and hyponyms like the word “language” in relation to the category of commenting. However, the true label was “non-commenting”, as the word was not used in a context of sexual language, but rather as “vague language” and “body language”.

6.4 Activation Clustering

Activation clustering (Girshick et al., 2014; Aubakirova and Bansal, 2016) accesses the activation values of all n neurons and treats the activation values per input as coordinates in n-dimensional space. K-means clustering was performed to group activation clusters and find common themes in these reports. Activation clustering is distinct from both LIME analysis and first derivative saliency in that it finds patterns and clusterings at a description-level.

Circumstances of Harassment: One of the clusters was classified as “ogling”: {*‘a group of boys was standing near us and were making weird expressions and as we moved away they started following’; ‘a group of guys lurking around the theater...’*}. Another cluster was classified as “commenting”: {*’a group of men were standing who commented on every girl who passed by the’, ’a group of boys were standing there... as we started moving one of them commented on us’*} Both of these clusters contained examples describing circumstances of the harassment, following the pattern of “a group of boys/men were standing/lurking and...” It can be inferred that certain forms of sexual harassment are more likely to happen with large groups of men. Activation clustering can identify the circumstances of harassment, helping potential victims to be better prepared.

Location and Time of Harassment: Some clusters contain examples that point to specific locations of harassment, e.g., a groping cluster: {*‘i was in the bus and there was this man who purposely fell on me and touched me inappropriately’; ‘while traveling in a crowded bus most of the time men try to grind their intimate part over my body’; ‘i was in the bus when a man standing tried to put his d**k on my hand’*}. Specific locations can also be found: {*‘the gurgaon sohna road is very unsafe at night especially if you are alone with no street lights’; ‘kurla station really gets scary at night once i was trying to get a train from kurla station around 10’; ‘mathura highway , not enough lights on the way during nights so is not safe for a individual to journey’*}. Notice that the second cluster examples also contain the word “night”. With data that contains more specific locations or times of day, activation clusters can serve as an automatic way to map out unsafe areas based on location and time of day.

Identifying Offenders: Examples from another groping cluster include: {*‘...her step father abused her physically for a year’; ‘one of the girl of about 6 years got raped by her own father’; ‘it happened at my house my brother harassed me and also misbehaved with me one night its been six months’*}. This shows that clusters can point to common relationships or titles for offenders. This phenomenon can be presumed to happen with names of offenders as well. If many reports have been filed around this offender, clusters will form around his/her name. Instead of a case of “he said, she said”, activation clustering provides an avenue towards “he said, they said”, as clusters form when multiple reports have been filed around the same name.

The main purpose of our visualization techniques is to explain what the black-box deep learning models are learning, such as locations, offenders, or times of day. With more detailed data in the future, we may be able to uncover more nuanced circumstances behind harassment.

7 Conclusion

We presented the novel task of identifying various forms of sexual harassment in personal stories. Our accurate multi-label classification models illustrate the plausibility of automatically filling out incident reports. Using visualization techniques, we found circumstances surrounding forms of harassment and the possibility of automatically identifying safe areas and repeat offenders. In future work, we hope to experiment with the transferability of our model to other datasets to encompass the diverse mediums through which these personal stories are shared. Honoring the courage that these victims demonstrated in sharing their stories online, we use these descriptions not only to help summarize online testimonials and provide more detailed safety advice, but also to help others report similar occurrences to hopefully prevent future sexual harassment from occurring.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported by a Google Faculty Research Award, a Bloomberg Data Science Research Grant, an IBM Faculty Award, and NVidia GPU awards.

References

Sweta Agrawal and Amit Awekar. 2018. Deep learning for detecting cyberbullying across multiple social media platforms. *European Conference on Information Retrieval*.

Malika Aubakirova and Mohit Bansal. 2016. Interpreting neural networks to improve politeness comprehension. *Proceedings of EMNLP*.

Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Conference on Social Computing (SocialCom)*, pages 71–80. IEEE.

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

Kathleen Kara Fitzpatrick, Alison Darcy, and Molly Vierhile. 2017. Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): a randomized controlled trial. *JMIR Mental Health*, 4(2).

Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2018. A unified deep learning architecture for abuse detection. *CoRR*.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*, volume 18.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. *Proceedings of NAACL*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. *Proceedings of ICLR*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

John Pestian, Henry Nasrallah, Pawel Matykiewicz, Aurora Bennett, and Antoon Leenaars. 2010. Suicide note classification using natural language processing: A content analysis. *Biomedical Informatics Insights*, 3:BII–S4706.

Callie Marie Rennison. 2002. *Rape and sexual assault: Reporting to police and medical attention, 1992-2000*. US Department of Justice, Office of Justice Programs Washington, DC.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

J Nicolas Schrading. 2015. *Analyzing domestic abuse using natural language processing on social media data*. Rochester Institute of Technology.

Nicolas Schrading, Cecilia Ovesdotter Alm, Ray Ptucha, and Christopher Homan. 2015a. An analysis of domestic abuse discourse on reddit. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2577–2583.

Nicolas Schrading, Cecilia Ovesdotter Alm, Raymond Ptucha, and Christopher Homan. 2015b. # whyistayed,# whyileft: Microblogging to make sense of domestic abuse. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1281–1286.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. *ICLR Workshop*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Evgeny Stepanov, Stephane Lathuiliere, Shammur Absar Chowdhury, Arindam Ghosh, Radu-Laurentiu Vieriu, Nicu Sebe, and Giuseppe Riccardi. 2017. Depression severity estimation from multiple modalities. *arXiv preprint arXiv:1711.06095*.

Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3).

Cynthia Van Hee, Gilles Jacobs, Chris Emmery, Bart Desmet, Els Lefever, Ben Verhoeven, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2018. Automatic detection of cyberbullying in social media text. *arXiv preprint arXiv:1801.05617*.

Amir Hossein Yazdavar, Hussein S Al-Olimat, Monireh Ebrahimi, Goonmeet Bajaj, Tanvi Banerjee, Krishnaprasad Thirunarayan, Jyotishman Pathak, and Amit Sheth. 2017. Semi-supervised approach to monitoring clinical depressive symptoms in social media. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 1191–1198. ACM.

Rui Zhao, Anna Zhou, and Kezhi Mao. 2016. Automatic detection of cyberbullying on social networks based on bullying features. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 43. ACM.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Marc Ziegele, Johannes Daxenberger, Oliver Quiring, and Iryna Gurevych. 2018. Developing automated measures to predict incivility in public online discussions on the facebook sites of established news media. In *Proceedings of the 68th Annual Conference of the International Communication Association (ICA)*.

Learning multiview embeddings for assessing dementia

Chloé Pou-Prom [1,2,3], **Frank Rudzicz**[1,2,3]
[1] Toronto Rehabilitation Institute - UHN, Toronto, Canada
[2]Vector Institute, Toronto, Canada
[3] Department of Computer Science, University of Toronto, Canada
`{ chloe, frank }@cs.toronto.edu`

Abstract

As the incidence of Alzheimer's Disease (AD) increases, early detection becomes crucial. Unfortunately, datasets for AD assessment are often sparse and incomplete. In this work, we leverage the multiview nature of a small AD dataset, DementiaBank, to learn an embedding that captures different modes of cognitive impairment. We apply generalized canonical correlation analysis (GCCA) to our dataset and demonstrate the added benefit of using multiview embeddings in two downstream tasks: identifying AD and predicting clinical scores. By including multiview embeddings, we obtain an F1 score of 0.82 in the classification task and a mean absolute error of 3.42 in the regression task. Furthermore, we show that multiview embeddings can be obtained from other datasets as well.

1 Introduction

Alzheimer's disease (AD) is a neurodegenerative progressive disease whose symptoms include memory loss, disorientation, and behavioral issues (Ballard et al., 2011). In 2017, 5.7 million Americans were living with AD, and the disease accounted for $11.4 billion in healthcare costs in the United States (Alzheimer's Association, 2018). AD is diagnosed through clinician-administered questionnaires, such as the Mini-Mental State Examination (MMSE), which assigns a score between 0 and 30 based on responses to questions testing memory, recall, and orientation (Folstein et al., 1975). For context, a MMSE score of 23 and below is associated with cognitive decline.

AD affects language and some of its symptoms include difficulties in word-finding and changes in the voice. Detecting these subtle changes can help identify AD at an early stage. Indeed, many studies have applied a combination of natural language processing and machine learning techniques to detect AD. On the DementiaBank (DB) dataset, which includes audio files and corresponding transcripts of participants completing a picture description task, Wankerl et al. (2017) employed an n-gram based approach to classify between participants with and without AD. On the same dataset, Fraser et al. (2015) extracted an extensive list of lexicosyntactic features from the transcripts and identified participants with AD with an accuracy of 81%. More recently, Hernández-Domínguez et al. (2018) looked at the information content units of the pictures and compared them to healthy population-specific references to achieve an F-score of 0.81.

Predicting clinical scores is a harder task and is more common in image processing, where researchers make use of brain scans. For example, Huang et al. (2016) used MRI scans from 805 subjects and relied on the longitudinal aspect of their dataset to predict MMSE scores. Specific to the DB dataset, Yancheva et al. (2015) extracted linguistic features and used a bivariate dynamic Bayes net to represent the longitudinal nature of the data, and obtained a mean absolute error (MAE) of 3.83. Focusing on subjects with larger samples of data yielded a MAE of 2.91.

In instances where multiple views of the same data are available, it makes sense to learn a vector representation (an embedding) that encapsulates the different sources of information. Benton et al. (2016) used different representations of their data (e.g., bag-of-words, word vectors) to learn multiview embeddings for Twitter users, and obtained promising results when evaluating their embeddings in downstream prediction tasks.

In this work, we leverage the multiview nature of DB to learn an embedding for each user. We evaluate the utility of the multiview embedding in two downstream tasks: classification of AD vs non-AD participants, and clinical score prediction.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2812–2817
Brussels, Belgium, October 31 - November 4, 2018.

2 Methods

2.1 Dataset

We use the DementiaBank (DB) corpus (Becker et al., 1994), which consists of adults aged 44 and older, assigned to either the *'Dementia'* ($N = 167$) or *'Healthy'* ($N = 97$) group based on a battery of neuropsychological tests and on their medical histories. In DB, participants performed the "Cookie Theft" picture description task from the Boston Diagnostic Aphasia Examination (Goodglass and Kaplan, 1983), in which they verbally describe the contents of a picture. Additionally, participants in the *'Dementia'* group completed the category fluency (i.e., naming words belonging to a given category), letter fluency (i.e., naming words that start with a given letter), sentence construction, and story recall tasks. The picture description and both fluency tasks were professionally transcribed and annotated with instances of filled pauses. Previous experiments in the literature on DB have been limited to the picture description task, most likely because the other tasks are not available for all participants.

2.2 Linguistic features

From transcripts of the picture description, category fluency and letter fluency tasks, we extract 565 linguistic features[1]. We compute lexical features (e.g., the mean number of syllables per word, the vocabulary richness as measured by the type-token-ratio[2]), semantic features (e.g., the mean specificity of words as measured by their depth in WordNet[3]), and syntactic features (e.g., the proportion of various parts-of-speech tags, such as nouns and adjectives). We also automatically extract various subjective measures, such as the mean imageability (i.e., a word's ability to evoke a mental image) and the mean age-of-acquisition of words using norms derived from the Bristol (Stadthagen-Gonzalez and Davis, 2006) and Gilhoolie-Logie (Gilhooly and Logie, 1980) norms. Finally, we train an LDA model of 100 topics (Blei et al., 2003) using a Wikipedia snapshot, and compute the topic probabilities for each transcript.

2.3 Learning a multiview embedding

We apply generalized canonical correlation analysis (GCCA) to our dataset to obtain a multiview embedding. We use GCCA as described by Benton et al. 2016 to learn linear transformations U_j which project different views of our data into the embedding G. In our experiments, we consider the following views of DB: linguistic features of the picture description, category fluency and letter fluency tasks, and demographic information.

Given $X \in \mathbb{R}^{d \times N}$, $X' \in \mathbb{R}^{d' \times N'}$, where N is the total number of data points, N' is the total number of data points for which all views J are available, and d and d' are the dimensions of X and X'; let X_j and X'_j denote views j of X and X'. Here, $j \in \{PD, CAT, LET, DEM\}$, which correspond to the picture description, category fluency, and letter fluency linguistic features, and demographic information, respectively.

1. We use GCCA to learn U_j from X'_{PD}, X'_{CAT}, X'_{LET}, X'_{DEM}, such that:
$$\underset{U_j, G'}{\text{minimize}} \sum_{j \in J} ||G' - U_j^T X'_j||_F^2$$
$U_j \in \mathbb{R}^{d_j \times k}$, $G' \in \mathbb{R}^{k \times N'}$.
2. We compute $G = U_{PD}^T X_{PD}$. Since $U_{PD} \in \mathbb{R}^{d_{PD} \times k}$ and $X_{PD} \in \mathbb{R}^{d_{PD} \times N}$, then $G \in \mathbb{R}^{k \times N}$.
3. We concatenate G to a subset of the picture description linguistic features, X^*_{PD}, to obtain $C = (X^*_{PD}, G)$, where $C \in \mathbb{R}^{(k + d^*_{PD}) \times N}$.
4. We use the augmented set of features C for two downstream tasks: AD classification and clinical score prediction.

3 Results

We run all experiments with 10-fold cross validation and test various settings of k, the dimension of the multiview embedding.

3.1 GCCA and classification

We select the top n linguistic features, ordered through a one-way ANOVA and concatenate them with multiview embeddings of size k. The $n + k$ features are then given as input to a random forest

---

[1] The code to extract these is being made available at `https://github.com/SPOClab-ca/COVFEFE`.

[2] The type-token ratio is obtained by dividing the number of types (i.e., the total number of *different* words) by the number of tokens (i.e., the total number of words).

[3] WordNet (Miller, 1995) is a lexical database which groups English words into collections of synonyms. The database is ordered from most generic (e.g., "plant") to most specific (e.g., "rose").

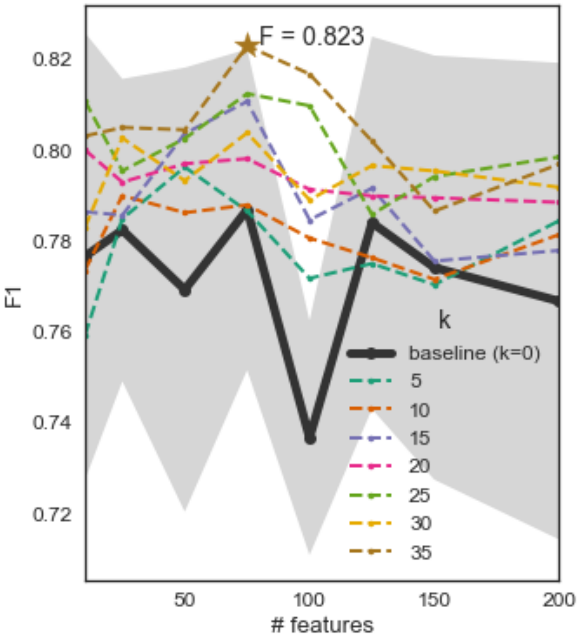

Figure 1: Classification of *'Dementia'* vs *'Healthy'* participants in DB. We report the F1 score as we increase the number of significant features used in our random forest classifier. The dark black line denotes the baseline (i.e., no GCCA) with the shaded grey region corresponding to the standard deviations, and the colored dotted lines denote experiments with multiview embeddings of size k.

classifier with 100 decision trees, and we report the F1 scores in Figure 1. Our best classification result ($F1 = 0.823 \pm 0.032$) is achieved with a multiview embedding of size $k = 35$ using the best $n = 75$ linguistic features. Adding GCCA embeddings improves classification results: an ANOVA test reveals a significant difference between F1 results with and without GCCA ($F = 15.85$, $p = 0.00018$), and a post-hoc Tukey's honest significant difference test reveals that F1 scores are significantly higher in experiments using GCCA embeddings ($p = 0.00018$).

Next, we look at multiview embeddings generated from different combinations of DB views, and report our F1 scores in Table 1 for embeddings of size $k = 35$ and using the top $n = 75$ features. Adding multiview embeddings always improves classification, and we obtain our best results by learning an embedding from the picture description and category fluency views.

3.2 GCCA and regression

Views	**F1**
None (baseline)	*0.782* (0.042)
$X'_{PD}, X'_{CAT}, X'_{LET}, X'_{DEM}$	0.811 (0.045)
$X'_{PD}, X'_{CAT}, X'_{LET}$	0.817 (0.037)
$X'_{PD}, X'_{LET}, X'_{DEM}$	0.815 (0.043)
$X'_{PD}, X'_{CAT}, X'_{DEM}$	0.818 (0.042)
X'_{PD}, X'_{CAT}	**0.824** (0.052)
X'_{PD}, X'_{LET}	0.805 (0.055)
X'_{PD}, X'_{DEM}	0.816 (0.057)

Table 1: Classification results with GCCA applied on different views of DB. We report the different views used to learn our multiview embedding and the resulting F1 scores (with standard deviation in parenthesis) on 10-fold cross-validation experiments. X' denotes the data points for which all views are present (i.e., the data used to learn multiview embeddings), and the subscripts PD, CAT, LET, DEM are used to represent the following views: picture description text features, category fluency text features, letter fluency text features, and demographic information.

To predict MMSE scores, we select the top n best features, ordered through a continuous one-way ANOVA, and concatenate them with our multiview embedding of size k. The $n + k$ features are then given as input to a linear regression model and we report the mean absolute error (MAE). 10-fold cross-validation results are given in Figure 2. Our lowest MAE of 3.412 ± 0.300 was obtained using a GCCA embedding of size $k = 5$ and retaining the top $n = 75$ linguistics features. Adding multiview embeddings yields the best results, but an ANOVA test reveals no significant difference ($F = 0.41$, $p = 0.53$).

3.3 Learning a multiview embedding from another dataset

We then perform the same experiments as described in sections 3.1 and 3.2, but we learn our U_{PD} linear projection matrix with a different dataset. We use *Talk2Me*[4], an online language assessment from the University of Toronto, in which participants use the web to complete a variety of language tasks, including the picture description task, the vocabulary task, the Winograd task (Levesque et al., 2011), and the word fluency task (including both category and letter fluency). For all tasks in *Talk2Me*, we transcribe the audio recordings using the Kaldi open-source automatic speech recognition engine (Povey et al., 2011), and extract the same set of text features as in Section 2.2. Next, we apply GCCA to learn a

---

[4] `https://www.cs.toronto.edu/talk2me`

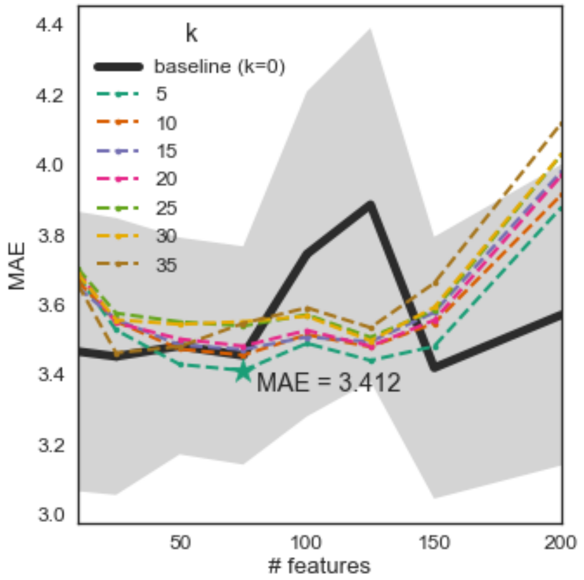

Figure 2: Clinical score prediction. We report the mean absolute error (lower is better) in predicting MMSE score as we increase the number of significant features used in our linear regression model. The black line denotes the baseline (i.e., no GCCA) with the shaded grey region corresponding to the standard deviations, and the colored dotted lines denote experiments with multiview embeddings of size k.

multiview embedding from the following views: picture description, story recall, vocabulary, fluency, and image naming tasks, and demographics.

As in previous experiments, we concatenate the multiview embedding with the DB picture description linguistic features, and use these to classify AD participants and to predict MMSE scores. In the regression task, the GCCA features from *Talk2Me* greatly hinder performance. The best result we obtain with *Talk2Me* multiview embeddings is an MAE of 3.929 ± 1.37. In classification, we observe improvements, as shown in Figure 3, and obtain an F1 of 0.793 ± 0.052. However, an ANOVA test reveals no significant difference with multiview embeddings ($F = 0.45$, $p = 0.50$).

4 Discussion

In our experiments, we use GCCA to learn a multiview embedding and augment our existing set of features. The multiview embedding consists of a vector representation which encapsulates information from various sources of information (i.e., the picture description task, the category and letter fluency tasks, and demographic data). We hypothesize that the additional information contained in this embedding would be useful in downstream

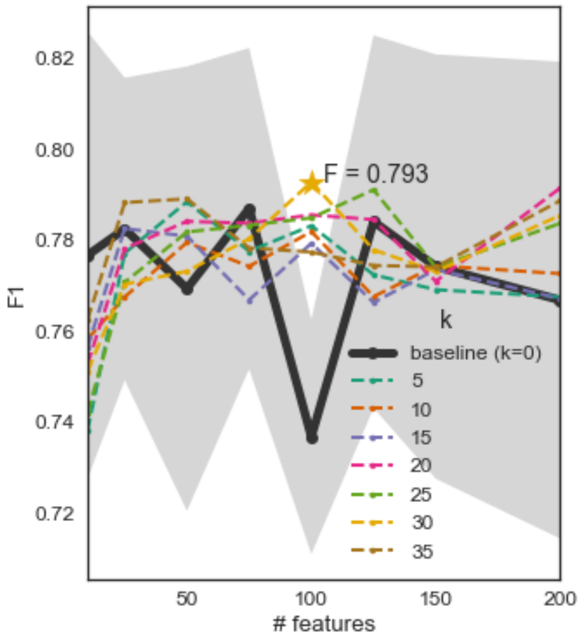

Figure 3: Classification task using multiview embeddings learned from the *Talk2Me* dataset. We report the F1 scores as we increase the number of significant features. The black line denotes the baseline (i.e., no GCCA) with the shaded grey region corresponding to the standard deviations, and the dotted colored lines denote experiments with multiview embeddings of size k obtained through GCCA on *Talk2Me* views.

tasks such as classification and regression. Indeed, the multiview embedding obtained from DB improves AD detection and clinical score prediction. Similarly, we also observe an improvement in classification when using a multiview embedding learned from a normative dataset.

Our results are better in the classification task than in the regression task, since the MMSE score is mainly used as a screening tool (i.e., determining if a person has AD or not) and has restricted sensitivity, especially for identifying milder stages of AD (Trzepacz et al., 2015).

5 Conclusion

We have shown that we can make use of the multiview aspect of a small dataset such as DB to learn a multiview embedding. This embedding can subsequently be used to improve models for classification and regression. In our experiments, multiview embeddings allowed the use of both the category and letter fluency data in DB, even though they were only available for the *'Dementia'* participants. Benefits are also possible using secondary datasets to learn multiview embeddings.

Extracting acoustic features – such as pause ratio, pitch, and Mel-frequency cepstral coefficients (MFCCs) – and treating them as an additional view is part of our future work. Furthermore, we will look into other secondary datasets as well as different approaches of obtaining multiview embeddings. While GCCA allows for an arbitrary number of views, it is limited in that it only learns linear projections to the embedding space. A possible alternative is deep generalized canonical correlation analysis (DGCCA), which makes use of neural networks to learn non-linear mappings to the embedding space (Benton et al., 2017).

Acknowledgments

This work was partially funded by an NSERC Discovery grant (RGPIN 435874) and by a Young Investigator award by the Alzheimer Society of Canada, both held by Rudzicz.

References

Alzheimer's Association. 2018. 2018 Alzheimer's disease facts and figures. *Alzheimer's & dementia : the journal of the Alzheimer's Association*, 13(4):325–373.

Clive Ballard, Serge Gauthier, Anne Corbett, Carol Brayne, Dag Aarsland, and Emma Jones. 2011. Alzheimer's disease. *Lancet*, 377(9770):1019–31.

James T. Becker, Francois Boller, Oscar I. Lopez, Judith Saxton, and Karen L. McGonigle. 1994. The Natural History of Alzheimer's Disease: Description of Study Cohort and Accuracy of Diagnosis. *Archives of Neurology*, 51(6):585–594.

Adrian Benton, Raman Arora, and Mark Dredze. 2016. Learning Multiview Embeddings of Twitter Users. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 14–19, Stroudsburg, PA, USA. Association for Computational Linguistics.

Adrian Benton, Huda Khayrallah, Biman Gujral, Dee Ann Reisinger, Sheng Zhang, and Raman Arora. 2017. Deep Generalized Canonical Correlation Analysis. *arXiv preprint, arXiv:1702.02519*.

David M. Blei, Andrew Y. Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Marshal F. Folstein, Susan E. Folstein, and Paul R. McHugh. 1975. "Mini-mental state". A practical method for grading the cognitive state of patients for the clinician. *Journal of Psychiatric research*, 12(3):189–198.

Kathleen C. Fraser, Jed A. Meltzer, and Frank Rudzicz. 2015. Linguistic features identify Alzheimer's disease in narrative speech. *Journal of Alzheimer's Disease*, 49(2):407–422.

K. J. Gilhooly and R. H. Logie. 1980. Age-of-acquisition, imagery, concreteness, familiarity, and ambiguity measures for 1,944 words. *Behavior Research Methods & Instrumentation*, 12(4):395–427.

Harold Goodglass and Edith Kaplan. 1983. Boston Diagnostic Aphasia Examination.

Laura Hernández-Domínguez, Sylvie Ratté, Gerardo Sierra-Martínez, and Andrés Roche-Bergua. 2018. Computer-based evaluation of Alzheimer's disease and mild cognitive impairment patients during a picture description task. *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring*, 10:260–268.

Lei Huang, Yan Jin, Yaozong Gao, Kim-Han Thung, and Dinggang Shen. 2016. Longitudinal clinical score prediction in Alzheimer's disease with soft-split sparse regression based random forest. *Neurobiology of Aging*, 46:180–191.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge. In *AAAI spring symposium: Logical formalizations of commonsense reasoning*, volume 46, page 47. Association for the Advancement of Artificial Intelligence (AAAI).

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the Association for Computing Machinery*, 38(11):39–41.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The Kaldi Speech Recognition Toolkit. In *Proceedings of the IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, Waikoloa, Hawaii, USA. Institute of Electrical and Electronics Engineers (IEEE) Signal Processing Society.

Hans Stadthagen-Gonzalez and Colin J. Davis. 2006. The Bristol norms for age of acquisition, imageability, and familiarity. *Behavior Research Methods*, 38(4):598–605.

Paula T Trzepacz, Helen Hochstetler, Shufang Wang, Brett Walker, and Andrew J. Saykin. 2015. Relationship between the Montreal Cognitive Assessment and Mini-mental State Examination for assessment of mild cognitive impairment in older adults. *BioMed Central Geriatrics*, 15(1):107.

Sebastian Wankerl, Elmar Nöth, and Stefan Evert. 2017. An N-Gram Based Approach to the Automatic Diagnosis of Alzheimer's Disease from Spoken Language. In *Proceedings of Interspeech 2017*, pages 3162–3166, Stockholm, Sweden. International Speech and Communication Association.

Maria Yancheva, Kathleen C. Fraser, and Frank Rudzicz. 2015. Using linguistic features longitudinally to predict clinical scores for Alzheimer's disease and related dementias. In *Proceedings of the 6th Workshop on Speech and Language Processing for Assistive Technologies*, page 134, Dresden, Germany. Speech and Language Processing and Assistive Technologies (SLPAT).

WikiConv: A Corpus of the Complete Conversational History of a Large Online Collaborative Community

Yiqing Hua
Cornell Tech
yiqing@cs.cornell.edu

Cristian Danescu-Niculescu-Mizil
Cornell University
cristian@cs.cornell.edu

Dario Taraborelli
Wikimedia Foundation
dtaraborelli@wikimedia.org

Nithum Thain and **Jeffery Sorensen** and **Lucas Dixon**
Jigsaw, Google
nthain, sorenj, ldixon@google.com

Abstract

We present a corpus that encompasses the complete history of conversations between contributors to Wikipedia, one of the largest online collaborative communities. By recording the intermediate states of conversations—including not only comments and replies, but also their modifications, deletions and restorations—this data offers an unprecedented view of online conversation. This level of detail supports new research questions pertaining to the process (and challenges) of large-scale online collaboration. We illustrate the corpus' potential with two case studies that highlight new perspectives on earlier work. First, we explore how a person's conversational behavior depends on how they relate to the discussion's venue. Second, we show that community moderation of toxic behavior happens at a higher rate than previously estimated. Finally the reconstruction framework is designed to be language agnostic, and we show that it can extract high quality conversational data in both Chinese and English.

1 Introduction

Compared to large-scale collections of conversations from social media (Felbo et al., 2017; Luo et al., 2012; Zhang et al., 2017; Tan et al., 2016) or news comments (Napoles et al., 2017), Wikipedia talk pages offer a unique perspective into goal-oriented discussions between thousands of volunteer contributors coordinating to write the largest online encyclopedia. Talk page data already underpins research on social phenomena such as conversational behavior (Danescu-Niculescu-Mizil et al., 2012, 2013), disputes (Wang and Cardie, 2014b), antisocial behavior (Wulczyn et al., 2017; Zhang et al., 2018) and collaboration (Kittur et al., 2007; Halfaker et al., 2009). However, the scope of such studies has so far been limited by a view of the conversation that is incomplete in two crucial ways: first, it only captures a subset of all discussions; and second, it only accounts for the final form of each conversation, which frequently differs from the interlocutors experience as the conversation develops.

In this paper, we undertake the challenge of reconstructing a complete and structured history of the conversational process in Wikipedia talk pages, containing detailed information about all the interlocutors' actions, such as adding and replying to comments, modifying or deleting them. To this end, we devise a methodology for identifying and structuring these actions, while also addressing the challenges spurring from the inconsistent formatting and the raw scale of existing records. This results in the largest public dataset of goal-oriented conversations, *WikiConv*, spanning five languages. The largest component of this dataset is based on the English Wikipedia, and contains roughly 91 million conversations consisting of 212 million conversational actions taking place in 24 million talk pages.

By including details about how each conversation evolved, this corpus provides an unprecedented view into the conversational process, as ex-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2818–2823
Brussels, Belgium, October 31 - November 4, 2018.

perienced by the interlocutors. In fact, we find that about 40% of discussion activity would be missed by approaches that do not consider comment modifications and deletions, and even more is missed when only considering the (final) static snapshots of conversations. Furthermore, a manual review of the English Wikipedia portion of the dataset reveals that 98% of the reply structure is recovered correctly and 98% of the interlocutor's actions are categorized correctly.

Since the reconstruction pipeline does not rely on any language specific heuristics, we also apply it to Chinese, German, Greek and Russian Wikipedia Talk page archives, in addition to those from English Wikipadia. A manual review of the conversations obtained from the Chinese Wikipedia Talk pages shows a similarly high reconstruction accuracy with that obtained from the English Wikipedia, suggesting that it is reasonable to apply the reconstruction pipeline to different languages. To encourage further validation, refinements and updates, we have open sourced the code and published the datasets.[1]

Finally, we present two case studies illustrating how the corpus can bring new insights into previously observed phenomena. We first analyze the conversational behavior of a subset of English Wikipedia contributors across the entire range of talk pages, and show that their levels of linguistic coordination vary according to where the conversation takes place. Second, we investigate the toxicity of deleted comments, and show that community moderation of undesired behavior takes place at a much higher rate than previously estimated.

2 Further Related Work

Past efforts aimed at characterizing conversations on Wikipedia talk pages have either focused on snapshots of discussion threads (Danescu-Niculescu-Mizil et al., 2012; Prabhakaran and Rambow, 2016; Wang and Cardie, 2014b,a), or have considered text segments in talk page history as incremental comments, ignoring conversational turns and reply structures within these conversations (Wulczyn et al., 2017). The limitations of these approaches can be seen in Figure 2, where, if we limit our analysis to only a snapshot of the final state of the conversation, we miss the abusive comment introduced in revision 3 and removed in revision 4, and thus miss an important part of the experience of the participants. In fact, this "hidden" activity accounts for one third of all actions taken on talk pages in English Wikipedia.

The closest dataset to our work is Bender et al. (2011) which introduces the Authority and Alignment in Wikipedia discussions corpus (AAWD), containing 365 talk page discussions. While acknowledging the complexity of conversational behaviors on Wikipedia talk pages, the AAWD work falls short of providing data on the deletions and follow-up changes to existing comments. Beyond addressing this shortcoming, the dataset we introduce in this paper is many orders of magnitude larger, containing 91 million conversations in English Wikipedia alone.

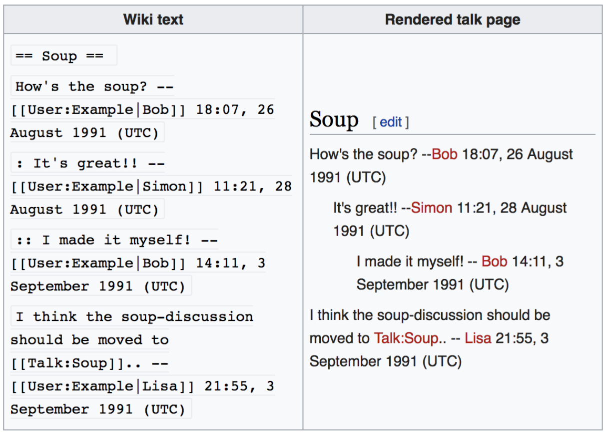

<table>
<tr><th>Wiki text</th><th>Rendered talk page</th></tr>
<tr><td><code>== Soup ==</code></td><td>Soup [edit]</td></tr>
<tr><td><code>How's the soup? -- [[User:Example|Bob]] 18:07, 26 August 1991 (UTC)</code></td><td>How's the soup? --Bob 18:07, 26 August 1991 (UTC)</td></tr>
<tr><td><code>: It's great!! -- [[User:Example|Simon]] 11:21, 28 August 1991 (UTC)</code></td><td>It's great!! --Simon 11:21, 28 August 1991 (UTC)</td></tr>
<tr><td><code>:: I made it myself! -- [[User:Example|Bob]] 14:11, 3 September 1991 (UTC)</code></td><td>I made it myself! -- Bob 14:11, 3 September 1991 (UTC)</td></tr>
<tr><td><code>I think the soup-discussion should be moved to [[Talk:Soup]].. -- [[User:Example|Lisa]] 21:55, 3 September 1991 (UTC)</code></td><td>I think the soup-discussion should be moved to Talk:Soup.. -- Lisa 21:55, 3 September 1991 (UTC)</td></tr>
</table>

Figure 1: An example Wiki markdown and its rendered form from Wikipedia Talk Page Help[2].

3 Conversation Reconstruction

Technically, comments are added to Wikipedia talk pages the same way content is added to article pages: contributors simply edit the markdown of any part of the talk page without relying on any functionality specialized for structuring the conversations. Figure 1 gives an example of the discussion interface and the resulting rendered conversation. Each edit results in a revision of the whole page that is permanently stored in a public historical record.[3] Because conversations on Wikipedia have no 'official' underlying structure, and instead are organized using indentation markup and other ad hoc visual cues, computational heuristics are necessary to interpret conversational structure.

Actions. We model the conversational structure of

---

[1] `github.com/conversationai/wikidetox`

[2] `mediawiki.org/wiki/Help:Talk_pages`

[3] In some rare cases revisions are deleted, for example, if personal information is accidentally written into a page.

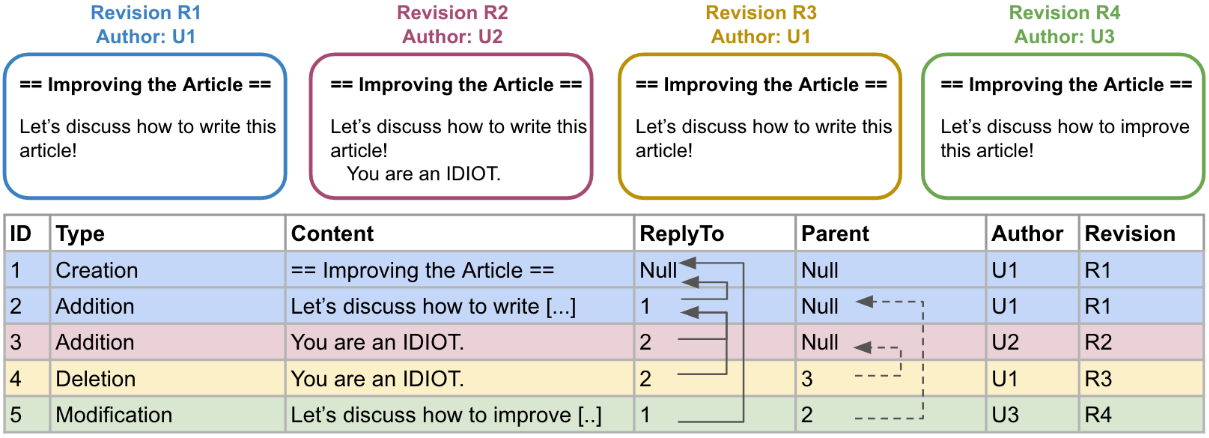

ID	Type	Content	ReplyTo	Parent	Author	Revision
1	Creation	== Improving the Article ==	Null	Null	U1	R1
2	Addition	Let's discuss how to write [...]	1	Null	U1	R1
3	Addition	You are an IDIOT.	2	Null	U2	R2
4	Deletion	You are an IDIOT.	2	3	U1	R3
5	Modification	Let's discuss how to improve [..]	1	2	U3	R4

Figure 2: **Example conversation reconstruction.** The action id in the ReplyTo column defines the conversation's structure; The Parent column indicates history, showing how actions change earlier actions. Note that each revision (color-coded) can introduce multiple actions.

English Wikipedia				**Reconstruction Accuracy by Action Type**			
Number of		**Action Type Breakdown**		*Boundary*	*Type*	*ReplyTo*	*Parent*
Distinct users	4.4M	Creation	21%	100%	100%	100%	100%
Talk Pages	24M	Addition	39%	96%	100%	95%	100%
Revisions	120M	Modification	13%	97%	95%	97%	95%
Conversations	91M	Deletion	24%	94%	96%	100%	100%
Actions	212M	Restoration	3%	84%	98%	100%	99%
		All actions:		96%	98%	98%	99%
Chinese Wikipedia				**Reconstruction Accuracy by Action Type**			
Number of		**Action Type Breakdown**		*Boundary*	*Type*	*ReplyTo*	*Parent*
Distinct users	87K	Creation	22%	100%	100%	100%	100%
Talk Pages	2.2M	Addition	50%	96%	100%	100%	100%
Revisions	4.6M	Modification	9%	84%	94%	99%	97%
Conversations	4.4M	Deletion	16%	99%	90%	100%	98%
Actions	6.4M	Restoration	3%	97%	98%	100%	98%
		All actions:		96%	98%	99%	99%

Table 1: **Summary statistics and reconstruction accuracy for the English and Chinese Wikipedia talk page corpora.** These statistics exclude actions that result in empty content after markup cleaning (e.g., purely formatting edits).

interactions as a graph of *actions*, as illustrated in Figure 2. Actions are categorized into five *types*:

- *Creation*: the start of a conversation thread based on a markup section heading being added (e.g. Action 1 in Figure 2).
- *Addition*: the addition of a new comment to a thread (e.g. Actions 2 and 3).
- *Modification*: modification of an existing comment (e.g. Action 5); with the original specified as the Parent-id.
- *Deletion*: the removal of a comment or heading (e.g. Action 4) where the Parent-id specifies the comment or heading's most recent action.
- *Restoration*: a revert specifies the deleted action being undone as the Parent-id.

All action types except thread creations, thread deletions and thread restorations also include a ReplyTo-id indicating the target of the reply.

From Page Revisions to Actions. Our reconstruction pipeline is a Python program written for Google Cloud Dataflow (also known as Apache Beam)[4] that operates on pages in parallel and on the revisions of each page sequentially in temporal order.

Due to the large scale of Wikipedia data, we use external sorting for pages that contains too many revisions to fit in a Dataflow worker's memory.

---

[4] `cloud.google.com/dataflow/`

When the number of revisions is too large for a Dataflow worker's local disk, the computation is performed in stages, a few years at a time.

Given the sorted set of a page-revisions, token-level diffs between sequential revisions are computed using a longest common sequence (LCS) algorithm.[5] Each sequential diff is then decomposed into the set of atomic conversation *actions* attributed to the user who submitted the page revision. During the sequential processing of a page's revisions, two data structures are maintained: each comment's current character offset, and a list of deleted comments. The comment offsets are used to interpret the difference between modification actions (edits within the bounds of an existing action) and additions; the deleted comments are used to identify restoration of comments.

We store the most recent 100 deleted comments between 10 to 1000 characters long, for each page. This is used to compute when a comment is restored by looking up deleted comments in a trie. The token length lower bound parameter avoids short commonly added comments—like "Thanks!"—from being interpreted as restorations. The upper bound ensures that occasional very long deleted comments are skipped, to bound Dataflow workers' memory usage.

Finally, reconstructed actions are processed using *mwparserfromhell* [6] to clean the MediaWiki formating. Note that, since arbitrary page changes are allowed, some actions cannot be processed by the parser (about 1 in 200,000); in such cases, an action's raw MediaWiki markup is stored.

Table 1 shows summary statistics of the final dataset on English and Chinese Wikipedia. The version of the raw data dumps processed were retrieved on July 1st 2018.

4 Evaluation of Reconstruction Quality

We evaluate the quality of the automatic reconstruction by manually verifying a randomly drawn subset of (at least) 100 examples from each action category. For each action we verify the accuracy of (1) the assigned action type, (2) the token-level boundary of the comment, (3) the ReplyTo relation and (4) the action's Parent relation.

We conduct the evaluation for both English and Chinese data (Table 1). With over 98% of actions classified correctly in both languages, the dataset exhibits a high annotation quality given its scale and detail. From the error cases in the English data, 10% result from limitations in the current technologies for HTML parsing and LCS matching. User behavior that we could interpret but is not yet captured by our algorithm, such as moving ongoing conversations to another talk pages accounts for another 24%. The remaining errors were from edits that we were unable to interpret. By open sourcing the reconstruction code, we encourage further refinements.

5 Case Studies

We now briefly present two studies on English Wikipedia that highlight the importance of (1) collecting the full history of Wikipedia across all pages and (2) capturing the various types of interactions.

Linguistic Coordination. Danescu-Niculescu-Mizil et al. (2012) studied language coordinations (i.e., in a conversation between a and b, to what degree is b systematically adopting a's language patterns when replying to a) on a conversational corpus derived from $5,657$ *User Talk pages*: those associated with, and managed by, a specific user. The study showed that social status mediates the amount of linguistic coordination, with contributors imitating more the linguistic style of those with higher status in the community.

We now show that the coordination pattern of the page owners in the previous dataset differs significantly based on where the conversation takes place. We compare each contributor's coordination patterns on their own user talk page to patterns exhibited on talk pages of other contributors, as well as to those on article talk pages—talk pages associated with a Wikipedia article. To avoid confounding different populations (and fall into the trap of Simpson's paradox), we only include in the comparison users that had a sufficient amount of contributions across all three venues. Figure 3 shows the three aggregated coordination values computed by applying the methodology of the original paper on 4 million addition actions that occurred before 2012.

Our results show with significant difference ($p < 0.001$ calculated by one-way ANOVA) that contributors coordinate the least when replying on other users' talk pages, and most on their own talk page. This leads us to speculate a new hypothesis: contributors have a different perception of status

---

[5] `github.com/google/diff-match-patch`

[6] `github.com/earwig/mwparserfromhell`

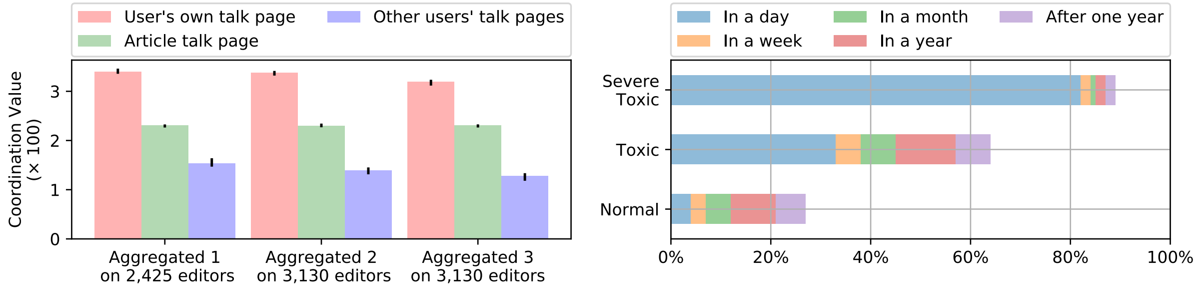

Figure 3: **(Left) Linguistic coordination depends on the discussion's venue.** Error bars are estimated by bootstrap resampling. **(Right) Deletion rate of content over varying time periods.**

or respect on their own page than on others. Such questions, which require more thorough investigation that depends on observing how contributors interact across different discussion venues, can be studied using the WikiConv corpus.

Moderation of toxic behavior. Wulczyn et al. (2017) measured prevalence of personal attacks in a Wikipedia talk page corpus, and evaluated the fraction of attacks that moderators follow up on with a block or warning (17.9%). However, because there was no structured history of comment deletion, the authors were unable to measure the rate at which toxic comments are moderated through deletion. Using the more complete datasets provided by WikiConv we show that the fraction of problematic comments moderated by Wikipedians is significantly higher than their initial estimate suggests.

We used the Perspective API[7] to score the toxicity of all addition and creation actions (which we refer to as "comments" here).[8] Each comment is further classified as toxic or non-toxic according to the equal error rate threshold, following the methodology of (Wulczyn et al., 2017), where false positives are offset by false negatives. The threshold is calculated by on the human labels in the Kaggle Toxicity dataset of Wikipedia comments [9]. Classification at this threshold yields 86% precision and 84% recall.

We used the same method to labeled comments with the *severe toxic* model. Figure 3 shows the fraction of comments deleted by Wikipedians who are not the author of the comment for different lengths of time; distinguishing between comments labeled as toxic, severely toxic, and the background distribution. The key results here are that nearly 33% of toxic comments are removed within a day; And over 82% of severely toxic comments are deleted within a day. This complements results previously reported by Wulczyn et al. (2017), accounting for an additional type of community moderation that is revealed using the detailed information about the history of the conversation provided by our corpus.

6 Conclusion and Future Work

We introduced a pipeline that extracts the complete conversational history of Wikipedia talk pages at a level of detail that was not previously available. We applied this pipeline to Wikipedia in multiple languages and evaluated its quality on the English and Chinese Talk page corpora, obtaining a high reconstruction accuracy for both the Chinese and English datasets (9̃8%). This level of detail and completeness opens avenues for new research, as well as revisiting and extending existing work on online conversational and collaboration behavior. For example, while in our use cases we have focused on contributors deleting toxic comments, one could seek to understand why and when an editor is deleting or rewording their own comments. Beyond refining the heuristics and parsing methods used in our reconstruction pipeline, and reducing the time needed to update the corpus, a significant remaining challenge is to capture conversations that happen across page boundaries.

7 Acknowledgement

We thank Thomas Ristenpart, Andreas Veit for proof reading; Ben Vitale for many helpful discussions on building the pipeline; and all those who helped us find bugs and discussed the challenges throughout. This project is supported in part by NSF grant CNS-1558500.

---

[7] `www.perspectiveapi.com`

[8] We release the scores with the dataset.

[9] The Jigsaw Toxicity Kaggle Competition: `goo.gl/N6UGPK`

References

Emily M Bender, Jonathan T Morgan, Meghan Oxley, Mark Zachry, Brian Hutchinson, Alex Marin, Bin Zhang, and Mari Ostendorf. 2011. Annotating social acts: Authority claims and alignment moves in Wikipedia talk pages. In *Proceedings of the Workshop on Languages in Social Media*.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of WWW*.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of ACL*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of EMNLP*.

Aaron Halfaker, Aniket Kittur, Robert Kraut, and John Riedl. 2009. A jury of your peers: quality, experience and ownership in Wikipedia. In *Proceedings of the 5th International Symposium on Wikis and Open Collaboration*.

Aniket Kittur, Bongwon Suh, Bryan A Pendleton, and Ed H Chi. 2007. He says, she says: Conflict and coordination in Wikipedia. In *Proceedings of the SIGCHI conference on Human factors in computing systems*.

Zhunchen Luo, Miles Osborne, and Ting Wang. 2012. Opinion retrieval in Twitter. In *Proceedings of ICWSM*.

Courtney Napoles, Joel Tetreault, Aasish Pappu, Enrica Rosato, and Brian Provenzale. 2017. Finding good conversations online: The Yahoo news annotated comments corpus. In *Proceedings of the 11th Linguistic Annotation Workshop*.

Vinodkumar Prabhakaran and Owen Rambow. 2016. A corpus of wikipedia discussions: Over the years, with topic, power and gender labels. In *Proceedings of LREC*.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of WWW*.

Lu Wang and Claire Cardie. 2014a. Improving agreement and disagreement identification in online discussions with a socially-tuned sentiment lexicon. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

Lu Wang and Claire Cardie. 2014b. A piece of my mind: A sentiment analysis approach for online dispute detection. In *Proceedings of ACL*.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of WWW*.

Amy X Zhang, Bryan Culbertson, and Praveen Paritosh. 2017. Characterizing online discussion using coarse discourse sequences. In *Proceedings of ICWSM*.

Justine Zhang, Jonathan P Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Nithum Thain, and Dario Taraborelli. 2018. Conversations gone awry: Detecting early signs of conversational failure. In *Proceedings of ACL*.

Marginal Likelihood Training of BiLSTM-CRF for Biomedical Named Entity Recognition from Disjoint Label Sets

Nathan Greenberg, Trapit Bansal, Patrick Verga, and Andrew McCallum
College of Information and Computer Sciences
University of Massachusetts Amherst
{ngreenberg, tbansal, pat, mccallum}@cs.umass.edu

Abstract

Extracting typed entity mentions from text is a fundamental component to language understanding and reasoning. While there exist substantial labeled text datasets for multiple *subsets* of biomedical entity types—such as genes and proteins, or chemicals and diseases—it is rare to find large labeled datasets containing labels for all desired entity types together. This paper presents a method for training a single CRF extractor from multiple datasets with disjoint or partially overlapping sets of entity types. Our approach employs marginal likelihood training to insist on labels that are present in the data, while filling in "missing labels". This allows us to leverage all the available data within a single model. In experimental results on the Biocreative V CDR (chemicals/diseases), Biocreative VI ChemProt (chemicals/proteins) and MedMentions (19 entity types) datasets, we show that joint training on multiple datasets improves NER F1 over training in isolation, and our methods achieve state-of-the-art results.

1 Introduction

Identifying entities in text is a vital component in language understanding, facilitating knowledge base construction (Riedel et al., 2013), question answering (Bordes et al., 2015), and search. Identifying these entities are particularly important in biomedical data. While large scale Named Entity Recognition (NER) datasets exist in news and web data (Tjong Kim Sang and De Meulder, 2003; Hovy et al., 2006), biomedical NER datasets are typically smaller and contain only one or two types per dataset. Ultimately, we would like to identify all entity types present across the union of the label sets during inference while leveraging all the available annotations to train our models.

While one may train a single model across the union of all the datasets available, this training procedure assumes that all labels (from the union of the tag set) are correctly annotated in every training instance – which is incorrect. On the other hand, training separate models on each available dataset does not take advantage of shared statistical strength from the multiple sources of information, and requires resolution of the conflicting predictions output by the different models.

To remedy these problems, we propose methods to train a joint model across the multiple tag-sets of the different datasets, sharing statistical strength by using a single feature encoder across datasets while respecting the incompleteness of the labels during training. Thus, our single model can take full advantage of all the available annotated resources and predict the full set of relevant types given a piece of text.

In experiments on three datasets, we show our methods outperform models that do not consider the incomplete annotations. We also show that jointly training on multiple datasets improves performance further and achieves state-of-the-art performance on the Biocreative V CDR dataset.

2 Model

Our models build on state-of-the-art NER systems (Lample et al., 2016) based on bi-directional Long Short Term Memory (BiLSTM) feature extractors fed into a conditional random field (CRF).

The data consists of input sequence of tokens $\mathbf{x} = \{x_1, \ldots, x_T\}$ where each token is a sequence of characters $x_t = \{c_1, \ldots, c_{K_t}\}$. The output consists of labels for each token in the sequence $\mathbf{y} = \{y_1, \ldots, y_T\}$. Labeling is done using the BILOU tagging scheme, following previous observations that it outperforms the BIO tagging scheme (Ratinov and Roth, 2009). We have D such datasets of input tokens and output labels.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2824–2829
Brussels, Belgium, October 31 - November 4, 2018.

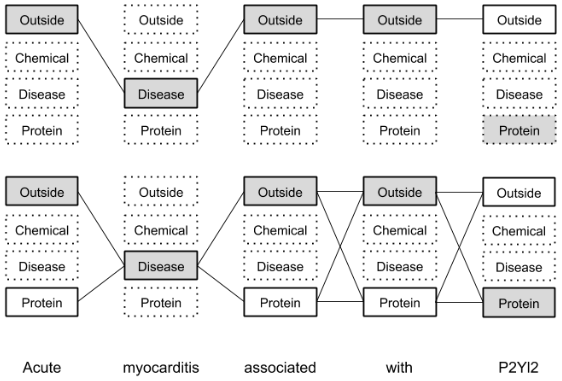

Figure 1: Training example where one label set contains Chemical/Protein and the other contains Chemical/Disease. Here Chemical and Disease annotations are given and Outside is ambiguous. Tokens labeled as Outside could potentially be either Outside or Protein (top). The shaded labels are the gold labels. The EM CRF marginalizes over all potential sequences (bot).

2.1 Feature Encoder BiLSTM

Our model takes a sequence of tokens from a single abstract as input. Tokens are generated using byte-pair encodings (BPE) (Gage, 1994; Sennrich et al., 2016), which have recently been shown to be effective for tokenization of biological texts by addressing the issue of rare or out-of-vocabulary tokens (Verga et al., 2018). BPE starts from white space tokenization and breaks down the tokens further. Because all of the evaluations are on the span level rather than the token level, the use of BPE does not impact any numerical performance. Each token t produced from BPE is mapped to a d dimensional word embedding w.

Character level features have been shown to improve NER accuracy (Lafferty et al., 2001; Lample et al., 2016; Passos et al., 2014). We encode characters in a word using another BiLSTM, similar to Lample et al. (2016), and obtain a character based embedding for every word by concatenating the last hidden state of the forward and backward character LSTM. We concatenate this character based embedding with the d-dimensional word embedding and input it to the word-level BiLSTM. This feature representation is then projected to the label dimension L using a linear layer, giving a matrix of scores $[f_{il}]$ where f_{il} is the score for predicting label $l \in [L]$ for token $i \in [T]$.

2.2 Conditional Random Field (CRF)

BiLSTM-CRF models used for named entity recognition add a CRF layer (Lafferty et al., 2001) on the output representations from the BiLSTM model described. The CRF layer scores all possible labelings to give a probability of the correct label sequence under the model. Given an input sequence of tokens $\mathbf{x} = \{x_1, \ldots, x_T\}$ and the output matrix of scores $[f_{il}]$, the score for an output labeling $\mathbf{y} = \{y_1, \ldots, y_T\}$ is given by: $s(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} \left(A_{y_{t-1}, y_t} + f_{t, y_t}\right)$, where A is an $L \times L$ matrix of parameters for transitioning between output labels. The CRF then generates the likelihood for the correct labeling by normalizing this score over all possible output labelings:

$$\log P(\mathbf{y}|\mathbf{x}) = s(\mathbf{x}, \mathbf{y}) - \underset{y'}{\text{logsumexp}}\, s(\mathbf{x}, \mathbf{y}') \quad (1)$$

The log normalization term here is: $\underset{y'}{\text{logsumexp}}\, s(\mathbf{x}, \mathbf{y}') = \log \sum_{y'} \exp s(\mathbf{x}, \mathbf{y}')$ where the sum goes over all possible labelings y' of the sequence and is computed efficiently using dynamic programming (Lafferty et al., 2001).

2.3 Tagging Multiple Datasets

One way to tag multiple datasets is to concatenate all the datasets with all the output labels and train a single BiLSTM-CRF model. However, this assumes that each text snippet is completely annotated across the label sets, which is not true. We now discuss two models which do not make this assumption.

2.3.1 Multiple CRFs

We first propose one simple method to get around the assumption of complete annotation – train separate CRFs for the label set of each dataset. In particular, to share statistical strengths on the input tokens, we share the BiLSTM feature encoder across the datasets but use separate CRF layers for each of the datasets. This is a multi-task learning model (Caruana, 1998) and is expected to perform better than the naive model as it no longer makes the strict assumption of complete annotation (by using separate CRFs), and shares statistical strength across datasets. However, given a new abstract to tag, this model will generate multiple possible labelings from the different CRFs. Moreover, the labelings output by the different CRFs may be inconsistent, and how to combine these multiple labelings is not obvious. We propose and evaluate a simple heuristic procedure for merging the outputs of the different CRF predictions. Whenever the different CRF predictions disagree on a span of tokens, we choose the prediction from the CRF that has higher marginal probability of predicting that span of tokens (Alg. 1 in supplementary).

2.3.2 EM Marginal CRF

We also propose an alternative principled approach that does not require a heuristic merging process. In order to label D datasets with some disjoint labels, we only consider the probability of the "observed labels" and allow the "unobserved" tokens to be free. Thus, when tagging dataset $i \in [D]$, we treat the non-entity tokens as potentially taking any entity type label from any of the other datasets as well as the 'O' label.

For a particular input $\mathbf{x}$ of length T from a dataset $i \in [D]$ with label set S_i, let $\mathbf{y}$ be the gold output label. Let $E \subset [T]$ be the index of tokens with any entity type label in S_i and $N \subset [T]$ be the index of tokens with 'O' label, and let $\mathbf{y}_E$ be the output sequence corresponding to indices in E, and similarly $\mathbf{y}_N$ be the output sequence for indices in N. Then, from (1), we get the likelihood $P_i(\mathbf{y_E} \cup \mathbf{y_N}|\mathbf{x})$, and a naive CRF trained on the concatenation of all the data will maximize this probability. However, since we cannot make the complete annotation assumption, we should instead maximize only the marginal probability of the observed entities on the dataset i, $P_i(\mathbf{y_E}|\mathbf{x})$, allowing $\mathbf{y_N}$ to take any values from the labels of the *other* datasets: $\cup_{j \neq i}^{D} S_j$. Thus,

$$\log P_i(\mathbf{y_E}|\mathbf{x}) = \log \sum_{\mathbf{y_N} \in \cup_{j \neq i} S_j} P_i(\mathbf{y_E}, \mathbf{y_N}|\mathbf{x})$$

$$\log P_i(\mathbf{y_E}|\mathbf{x}) = \underset{\mathbf{y_N} \in \cup_{j \neq i} S_j}{\text{logsumexp}}\, s(\mathbf{x}, \mathbf{y}_E, \mathbf{y}_N) - \log Z$$

where $\log Z$ is the log normalization term which is the same as in (1). Note that since the normalization term is the same here as for a standard CRF, we can still use the same dynamic programming algorithm as for a regular CRF to compute this $\log Z$. Now, in order to compute the first term, we note that it is similar to the computation required to compute $\log Z$ – whereas $\log Z$ is obtained by summing over all possible output sequences, this term is obtained by summing over all possible output sequences which have indices in E fixed to the correct label and indices in N taking values from $\cup_{j \neq i} S_j$. Thus, this can be computed using the same dynamic programming algorithm (Tsuboi et al., 2008), and the implementation of training this model is compatible with modern automatic differentiation libraries.

3 Experimental Results

We perform experiments on two benchmark Biocreative datasets as well as the recently introduced MedMentions data (Murty et al., 2018). Our experiments consider three types of models. The **single CRF** model naively concatenates all training datasets together and assumes complete labeling, **multi CRF** has a single Bi-LSTM feature encoder with a separate CRF for each dataset (Section 2.3.1), and **EM CRF** has a single feature encoder and a single CRF trained with EM marginalization (Section 2.3.2). For full dataset statistics and specific implementation details see supplementary material.

3.1 Biocreative V / VI

Biocreative V Chemical Disease Relation (CDR): consists of 1,500 titles and abstracts from PubMed, human annotated with chemical and disease mentions (Li et al., 2016), and has been used in previous NER evluations (Fries et al., 2017; Leaman and Lu, 2016). **Biocreative VI ChemProt (CP)**: consists of 2,432 PubMed titles and abstracts, and contains human annotated mentions of both chemicals and proteins (Krallinger et al., 2017)[1].

Our results are shown in Table 1. The top portion of the table shows models trained on single datasets, and the bottom portion shows models trained on both CDR and CP. Comparing the top and bottom portions of the table, we can see that models trained on both CP and CDR outperform training on either in isolation. Further, we see in the bottom section that our EM CRF outperforms the single CRF model and is generally better than the multi CRF model.

3.2 Adding Additional Data

Weakly Labeled data The addition of weakly labeled data has been used recently to improve the performance of relation extraction systems (Peng et al., 2016; Verga et al., 2018). In these approaches, titles and abstracts from PubMed are annotated using Pubtator, a state of the art entity tagging and linking/normalization system (Wei et al., 2013). We use the same weakly labeled data from Verga et al. (2018).

Results when adding in the additional weakly labeled data is shown in Table 2. Our models

---

[1] To the best of our knowledge, there is no benchmark result for this dataset

	CDR						ChemProt					
	Chemical			Disease			Chemical			Protein		
Model	P	R	F1	P	R	F1	P	R	F1	P	R	F1
CDR single CRF	91.8	84.7	88.1	80.7	75.0	77.8	-	-	-	-	-	-
CP single CRF	-	-	-	-	-	-	85.9	84.2	85.0	83.3	81.3	82.3
CDR+CP												
single CRF	93.3	91.9	92.6	82.0	75.5	78.6	87.8	86.8	**87.3**	82.9	83.2	83.0
multi CRF	94.1	91.8	**92.9**	82.7	76.8	**79.6**	84.8	88.4	86.6	83.7	81.6	82.6
EM CRF	94.0	91.8	**92.9**	81.1	77.7	79.4	87.1	87.6	**87.3**	83.2	83.6	**83.4**

Table 1: Precision, recall, and F1 for Biocreative V CDR and Biocreative VI ChemProt(CP) Datasets. The top portion of the table shows models trained on single datasets, the bottom portion trains on both CDR and CP, and the bottom portion trains on CDR and CP. Highest F1 scores in each section are bolded.

	CDR						ChemProt					
	Chemical			Disease			Chemical			Protein		
Model	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TaggerOne	92.4	84.7	88.4	83.1	76.4	79.6	-	-	-	-	-	-
TaggerOne †	94.2	88.8	91.4	85.2	80.2	82.6	-	-	-	-	-	-
WLD single CRF	97.5	85.2	91.0	84.4	83.0	83.7	86.3	77.5	81.6	80.0	63.7	70.9
CDR+CP+WLD												
single CRF	95.7	92.4	94.0	84.5	82.9	83.7	87.6	87.2	87.4	82.0	84.7	83.3
multi CRF	95.6	93.2	**94.4**	85.7	84.0	**84.8**	85.6	90.0	87.7	87.3	81.8	84.5
EM CRF	96.6	92.1	94.3	84.9	83.6	84.2	88.9	87.5	**88.2**	84.0	86.1	**85.0**

Table 2: WLD trains with weakly labeled data. Highest F1 scores in each section are bolded. † jointly performs NER and entity linking.

improve further, outperforming the state-of-the-art TaggerOne model (Leaman and Lu, 2016).

3.3 MedMentions

MedMentions (Murty et al., 2018) is a recently introduced large dataset of PubMed abstracts containing entity linked mentions of many different semantic types. We used this data to create an artificially extreme example where two training sets contain 9 and 10 entity types each. The two type sets are fully disjoint (further details in supplementary).

In Table 3, we see that the single CRF model performs very poorly in this extreme setting due to the large amount of missing annotations. The multi CRF and EM CRF both perform well and come close to the performance of a single CRF trained on the full data, which is approximately twice as much annotated data.

Model	P	R	F1
single CRF	65.0	24.3	35.3
multi CRF	62.5	50.9	56.1
EM CRF	59.7	54.2	**56.8**
Full single CRF	60.5	58.3	59.4

Table 3: MedMentions results. Full single CRF is trained on the full set of annotations. Other models are trained on the two disjoint training sets.

4 Related Work

Until recently, feature engineered machine learning models were the highest performing approaches to NER (Ratinov and Roth, 2009; Passos et al., 2014). More recently, neural network based approaches have become state-of-the-art (Lample et al., 2016; Strubell et al., 2017; Peters et al., 2017). In BioNLP, many highest performing systems still use engineered features fed into a CRF (Wei et al., 2015; Leaman et al., 2015; Leaman and Lu, 2016). In addition to the two datasets we explored in this work, there are several other popular bio NER datasets for chemicals (Krallinger et al., 2015), species (Wang et al., 2010), diseases

(Doğan et al., 2014), and genes (Tanabe et al., 2005).

In concurrent work, Wang et al. (2018) train a model very similar to our multi-CRF model on multiple biological NER datasets with non-fully overlapping labels. Additionally, they experiment with different ways of sharing the parameters of the BiLSTM encoder. We believe this work is complementary to ours, and in many ways deals with a simpler subset of the tasks we address. Wang et al. assumes complete labeling in each of their datasets, and does not attempt to merge the final results of the multiple CRFS. On the other hand, we focus on the problem of cohesively labeling a dataset with the joint set of the different label sets, either directly through the EM model or by the merging process of the multi-CRF model.

Our method of training via marginal likelihood is the same as Tsuboi et al. (2008), who trained CRF models for Japanese word segmentation and POS tagging where only partial annotations of sentences are available. In comparison, we use the marginal likelihood training in conjunction with state-of-the art deep learning models for NER and use it to tag across multiple disjoint labels sets.

5 Conclusions and Future Work

We've introduced a method for training NER models on multiple datasets containing disjoint label sets. We show experimentally that this joint training improves performance and that our EM CRF methods outperform models using a single CRF.

One interesting problem that our models do not account for is the existence of overlapping and non-continuous entity spans. Particularly when annotating using disjoint label sets, a token could belong to multiple entity spans from different label sets. We are interested in investigating this problem in future work.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and the Center for Data Science, in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction., and in part by the National Science Foundation under Grant No. IIS-1514053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Jason Fries, Sen Wu, Alex Ratner, and Christopher Ré. 2017. Swellshark: A generative model for biomedical named entity recognition without labeled data. *arXiv preprint arXiv:1704.06360*.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Martin Krallinger, Obdulia Rabal, Saber A. Akhondi, Martn Prez Prez, Jess Santamara, Prez Gael Rodrguez, Georgios Tsatsaronis, Ander Intxaurrondo, Jos Antonio Lpez, Umesh Nandal, Erin Van Buel, Akileshwari Chandrasekhar, Marleen Rodenburg, Astrid Laegreid, Marius Doornenbal, Julen Oyarzabal, Analia Loureno, and Alfonso Valencia. 2017. Overview of the biocreative vi chemical-protein interaction track. *Proceedings of the BioCreative VI Workshop*, page 140.

Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. 2015. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7(S1):S2.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Robert Leaman and Zhiyong Lu. 2016. Taggerone: joint named entity recognition and normalization with semi-markov models. *Bioinformatics*, 32(18):2839–2846.

Robert Leaman, Chih-Hsuan Wei, and Zhiyong Lu. 2015. tmchem: a high performance approach for chemical named entity recognition and normalization. *Journal of cheminformatics*, 7(1):S3.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. 2018. Hierarchical Losses and New Resources for Fine-grained Entity Typing and Linking. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *CoNLL-2014*, page 78.

Yifan Peng, Chih-Hsuan Wei, and Zhiyong Lu. 2016. Improving chemical disease relation extraction with rich features and weakly labeled data. *Journal of cheminformatics*, 8(1):53.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark.

Lorraine Tanabe, Natalie Xie, Lynne H Thom, Wayne Matten, and W John Wilbur. 2005. Genetag: a tagged corpus for gene/protein named entity recognition. *BMC bioinformatics*, 6(1):S3.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Yuta Tsuboi, Hisashi Kashima, Hiroki Oda, Shinsuke Mori, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 897–904. Association for Computational Linguistics.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously Self-attending to All Mentions for Full-Abstract Biological Relation Extraction. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, New Orleans, Louisiana.

Xinglong Wang, Jun'ichi Tsujii, and Sophia Ananiadou. 2010. Disambiguating the species of biomedical named entities using natural language parsers. *Bioinformatics*, 26(5):661–667.

Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. 2018. Cross-type biomedical named entity recognition with deep multi-task learning. *arXiv preprint arXiv:1801.09851*.

Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. 2013. Pubtator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Research*, 41.

Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. 2015. Gnormplus: an integrative approach for tagging genes, gene families, and protein domains. *BioMed research international*, 2015.

Adversarial training for multi-context joint entity and relation extraction

Giannis Bekoulis **Johannes Deleu** **Thomas Demeester** **Chris Develder**
Ghent University - imec, IDLab
Department of Information Technology
`{firstname.lastname}@ugent.be`

Abstract

Adversarial training (AT) is a regularization method that can be used to improve the robustness of neural network methods by adding small perturbations in the training data. We show how to use AT for the tasks of entity recognition and relation extraction. In particular, we demonstrate that applying AT to a general purpose baseline model for jointly extracting entities and relations, allows improving the state-of-the-art effectiveness on several datasets in different contexts (i.e., news, biomedical, and real estate data) and for different languages (English and Dutch).

1 Introduction

Many neural network methods have recently been exploited in various natural language processing (NLP) tasks, such as parsing (Zhang et al., 2017), POS tagging (Lample et al., 2016), relation extraction (dos Santos et al., 2015), translation (Bahdanau et al., 2015), and joint tasks (Miwa and Bansal, 2016). However, Szegedy et al. (2014) observed that intentional small scale perturbations (i.e., adversarial examples) to the input of such models may lead to incorrect decisions (with high confidence). Goodfellow et al. (2015) proposed adversarial training (AT) (for image recognition) as a regularization method which uses a mixture of clean and adversarial examples to enhance the robustness of the model. Although AT has recently been applied in NLP tasks (e.g., text classification (Miyato et al., 2017)), this paper — to the best of our knowledge — is the first attempt investigating regularization effects of AT in a joint setting for two related tasks.

We start from a baseline joint model that performs the tasks of named entity recognition (NER) and relation extraction at once. Previously proposed models (summarized in Section 2) exhibit several issues that the neural network-based baseline approach (detailed in Section 3.1) overcomes: (i) our model uses automatically extracted features without the need of external parsers nor manually extracted features (see Gupta et al. (2016); Miwa and Bansal (2016); Li et al. (2017)), (ii) all entities and the corresponding relations within the sentence are extracted at once, instead of examining one pair of entities at a time (see Adel and Schütze (2017)), and (iii) we model relation extraction in a multi-label setting, allowing multiple relations per entity (see Katiyar and Cardie (2017); Bekoulis et al. (2018a)). The core contribution of the paper is the use of AT as an extension in the training procedure for the joint extraction task (Section 3.2).

To evaluate the proposed AT method, we perform a large scale experimental study in this joint task (see Section 4), using datasets from different contexts (i.e., news, biomedical, real estate) and languages (i.e., English, Dutch). We use a strong baseline that outperforms all previous models that rely on automatically extracted features, achieving state-of-the-art performance (Section 5). Compared to the baseline model, applying AT during training leads to a consistent additional increase in joint extraction effectiveness.

2 Related work

Joint entity and relation extraction: Joint models (Li and Ji, 2014; Miwa and Sasaki, 2014) that are based on manually extracted features have been proposed for performing both the named entity recognition (NER) and relation extraction subtasks at once. These methods rely on the availability of NLP tools (e.g., POS taggers) or manually designed features leading to additional complexity. Neural network methods have been exploited to overcome this feature design issue and usually involve RNNs and CNNs (Miwa and Bansal,

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2830–2836
Brussels, Belgium, October 31 - November 4, 2018.

2016; Zheng et al., 2017). Specifically, Miwa and Bansal (2016) as well as Li et al. (2017) apply bidirectional tree-structured RNNs for different contexts (i.e., news, biomedical) to capture syntactic information (using external dependency parsers). Gupta et al. (2016) propose the use of various manually extracted features along with RNNs. Adel and Schütze (2017) solve the simpler problem of entity classification (EC, assuming entity boundaries are given), instead of NER, and they replicate the context around the entities, feeding entity pairs to the relation extraction layer. Katiyar and Cardie (2017) investigate RNNs with attention without taking into account that relation labels are not mutually exclusive. Finally, Bekoulis et al. (2018a) use LSTMs in a joint model for extracting just one relation at a time, but increase the complexity of the NER part. Our baseline model enables simultaneous extraction of multiple relations from the same input. Then, we further extend this strong baseline using adversarial training.

Adversarial training (AT) (Goodfellow et al., 2015) has been proposed to make classifiers more robust to input perturbations in the context of image recognition. In the context of NLP, several variants have been proposed for different tasks such as text classification (Miyato et al., 2017), relation extraction (Wu et al., 2017) and POS tagging (Yasunaga et al., 2018). AT is considered as a regularization method. Unlike other regularization methods (i.e., dropout (Srivastava et al., 2014), word dropout (Iyyer et al., 2015)) that introduce random noise, AT generates perturbations that are variations of examples easily misclassified by the model.

3 Model

3.1 Joint learning as head selection

The baseline model, described in detail in Bekoulis et al. (2018b), is illustrated in Fig. 1. It aims to detect (i) the type and the boundaries of the entities and (ii) the relations between them. The input is a sequence of tokens (i.e., sentence) $w = w_1, ..., w_n$. We use character level embeddings to implicitly capture morphological features (e.g., prefixes and suffixes), representing each character by a vector (embedding). The character embeddings are fed to a bidirectional LSTM (BiLSTM) to obtain the character-based representation of the word. We also use pre-trained word embeddings.

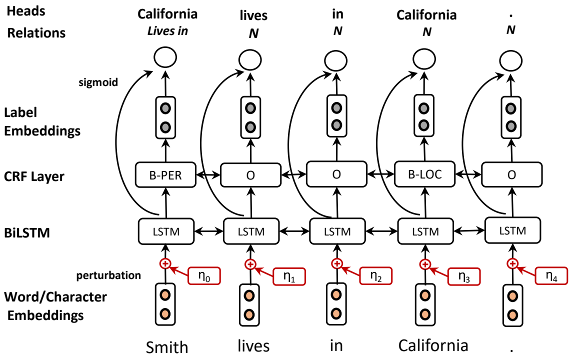

Figure 1: Our model for joint entity and relation extraction with adversarial training (AT) comprises (i) a word and character embedding layer, (ii) a BiLSTM layer, (iii) a CRF layer and (iv) a relation extraction layer. In AT, we compute the worst-case perturbations η of the input embeddings.

Word and character embeddings are concatenated to form the final token representation, which is then fed to a BiLSTM layer to extract sequential information.

For the **NER task**, we adopt the BIO (Beginning, Inside, Outside) encoding scheme. In Fig. 1, the B-*PER* tag is assigned to the beginning token of a 'person' (*PER*) entity. For the prediction of the entity tags, we use: (i) a softmax approach for the entity classification (EC) task (assuming entity boundaries given) or (ii) a CRF approach where we identify both the *type* and the boundaries for each entity. During decoding, in the softmax setting, we greedily detect the entity *types* of the tokens (i.e., independent prediction). Although independent distribution of *types* is reasonable for EC tasks, this is not the case when there are strong correlations between neighboring tags. For instance, the BIO encoding scheme imposes several constraints in the NER task (e.g., the B-*PER* and I-*LOC* tags cannot be sequential). Motivated by this intuition, we use a linear-chain CRF for the NER task (Lample et al., 2016). For decoding, in the CRF setting, we use the Viterbi algorithm. During training, for both EC (softmax) and NER tasks (CRF), we minimize the cross-entropy loss $\mathcal{L}_{\text{NER}}$. The entity tags are later fed into the relation extraction layer as label embeddings (see Fig. 1), assuming that knowledge of the entity *types* is beneficial in predicting the relations between the involved entities.

We model the **relation extraction task** as a multi-label head selection problem (Bekoulis et al., 2018b; Zhang et al., 2017). In our model, each word w_i can be involved in multiple relations with other words. For instance, in the example illustrated in Fig. 1, "Smith" could be involved not only in a *Lives in* relation with the token "California" (head) but also in other relations simultaneously (e.g., *Works for, Born In* with some corresponding tokens). The goal of the task is to predict for each word w_i, a vector of heads $\hat{y}_i$ and the vector of corresponding relations $\hat{r}_i$. We compute the score $s(w_j, w_i, r_k)$ of word w_j to be the head of w_i given a relation label r_k using a single layer neural network. The corresponding probability is defined as: $\mathbb{P}(w_j, r_k \mid w_i; \theta) = \sigma(s(w_j, w_i, r_k))$, where $\sigma(.)$ is the sigmoid function. During training, we minimize the cross-entropy loss $\mathcal{L}_{\text{rel}}$ as:

$$\sum_{i=0}^{n}\sum_{j=0}^{m} -\log \mathbb{P}(y_{i,j},\, r_{i,j} \mid w_i; \theta) \tag{1}$$

where m is the number of associated heads (and thus relations) per word w_i. During decoding, the most probable heads and relations are selected using threshold-based prediction. The final objective for the joint task is computed as $\mathcal{L}_{\text{JOINT}}(w;\theta) = \mathcal{L}_{\text{NER}} + \mathcal{L}_{\text{rel}}$ where θ is a set of parameters. In the case of multi-token entities, only the last token of the entity can serve as head of another token, to eliminate redundant relations. If an entity is not involved in any relation, we predict the auxiliary "N" relation label and the token itself as head.

3.2 Adversarial training (AT)

We exploit the idea of AT (Goodfellow et al., 2015) as a regularization method to make our model robust to input perturbations. Specifically, we generate examples which are variations of the original ones by adding some noise at the level of the concatenated word representation (Miyato et al., 2017). This is similar to the concept introduced by Goodfellow et al. (2015) to improve the robustness of image recognition classifiers. We generate an adversarial example by adding the worst-case perturbation η_{adv} to the original embedding w that maximizes the loss function:

$$\eta_{adv} = \underset{\|\eta\|\leq\epsilon}{\operatorname{argmax}}\, \mathcal{L}_{\text{JOINT}}(w+\eta; \hat{\theta}) \tag{2}$$

where $\hat{\theta}$ is a copy of the current model parameters. Since Eq. (2) is intractable in neural networks, we use the approximation proposed in Goodfellow et al. (2015) defined as: $\eta_{adv} = \epsilon g / \|g\|$, with $g = \nabla_w \mathcal{L}_{\text{JOINT}}(w;\hat{\theta})$, where ϵ is a small bounded norm treated as a hyperparameter. Similar to Yasunaga et al. (2018), we set ϵ to be $\alpha\sqrt{D}$ (where D is the dimension of the embeddings). We train on the mixture of original and adversarial examples, so the final loss is computed as: $\mathcal{L}_{\text{JOINT}}(w;\hat{\theta}) + \mathcal{L}_{\text{JOINT}}(w+\eta_{adv};\hat{\theta})$.

4 Experimental setup

We evaluate our models on four datasets, using the code as available from our github codebase.[1] Specifically, we follow the 5-fold cross-validation defined by Miwa and Bansal (2016) for the ACE04 (Doddington et al., 2004) dataset. For the CoNLL04 (Roth and Yih, 2004) EC task (assuming boundaries are given), we use the same splits as in Gupta et al. (2016); Adel and Schütze (2017). We also evaluate our models on the NER task similar to Miwa and Sasaki (2014) in the same dataset using 10-fold cross validation. For the Dutch Real Estate Classifieds, DREC (Bekoulis et al., 2017) dataset, we use train-test splits as in Bekoulis et al. (2018a). For the Adverse Drug Events, ADE (Gurulingappa et al., 2012), we perform 10-fold cross-validation similar to Li et al. (2017). To obtain comparable results that are not affected by the input embeddings, we use the embeddings of the previous works. We employ early stopping in all of the experiments. We use the Adam optimizer (Kingma and Ba, 2015) and we fix the hyperparameters (i.e., α, dropout values, best epoch, learning rate) on the validation sets. The scaling parameter α is selected from $\{5\text{e}{-2}, 1\text{e}{-2}, 1\text{e}{-3}, 1\text{e}{-4}\}$. Larger values of α (i.e., larger perturbations) lead to consistent performance decrease in our early experiments. This can be explained from the fact that adding more noise can change the content of the sentence as also reported by Wu et al. (2017).

We use three types of evaluation, namely: (i) *S(trict)*: we score an entity as correct if both the entity boundaries and the entity *type* are correct (ACE04, ADE, CoNLL04, DREC), (ii) *B(oundaries)*: we score an entity as correct if only the entity boundaries are correct while the entity *type* is not taken into account (DREC) and (iii) *R(elaxed)*: a multi-token entity is considered correct if at least one correct *type* is assigned to the tokens comprising the entity, assuming that the

---

[1] `https://github.com/bekou/multihead_joint_entity_relation_extraction`

	Settings	Features	Eval.	Entity	Relation	Overall
ACE 04	Miwa and Bansal (2016)	✓	*S*	81.80	48.40	65.10
	Katiyar and Cardie (2017)	✗	*S*	79.60	45.70	62.65
	baseline	✗	*S*	81.16	47.14	64.15
	baseline + AT	✗	*S*	**81.64**	**47.45**	**64.54**
CoNLL 04	Gupta et al. (2016)	✓	*R*	92.40	69.90	81.15
	Gupta et al. (2016)	✗	*R*	88.80	58.30	73.60
	Adel and Schütze (2017)	✗	*R*	82.10	62.50	72.30
	baseline EC	✗	*R*	**93.26**	67.01	80.14
	baseline EC + AT	✗	*R*	93.04	**67.99**	**80.51**
	Miwa and Sasaki (2014)	✓	*S*	80.70	61.00	70.85
	baseline	✗	*S*	83.04	61.04	72.04
	baseline + AT	✗	*S*	**83.61**	**61.95**	**72.78**
DREC	Bekoulis et al. (2018a)	✗	*B*	79.11	49.70	64.41
	baseline	✗	*B*	82.30	52.81	67.56
	baseline + AT	✗	*B*	**82.96**	**53.87**	**68.42**
	baseline	✗	*S*	81.39	52.26	66.83
	baseline + AT	✗	*S*	**82.04**	**53.12**	**67.58**
ADE	Li et al. (2016)	✓	*S*	79.50	63.40	71.45
	Li et al. (2017)	✓	*S*	84.60	71.40	78.00
	baseline	✗	*S*	86.40	74.58	80.49
	baseline + AT	✗	*S*	**86.73**	**75.52**	**81.13**

Table 1: Comparison of our method with the state-of-the-art in terms of F_1 score. The proposed models are: (i) baseline, (ii) baseline EC (predicts only entity classes) and (iii) baseline (EC) + AT (regularized by AT). The ✓and ✗ symbols indicate whether the models rely on external NLP tools. We include different evaluation types (*S*, *R* and *B*).

boundaries are known (CoNLL04), to compare to previous works. In all cases, a relation is considered as correct when both the relation *type* and the argument entities are correct.

5 Results

Table 1 shows our experimental results. The name of the dataset is presented in the first column while the models are listed in the second column. The proposed models are the following: (i) *baseline*: the baseline model shown in Fig. 1 with the CRF layer and the sigmoid loss, (ii) *baseline EC*: the proposed model with the softmax layer for EC, (iii) *baseline (EC) + AT*: the baseline regularized using AT. The final three columns present the F_1 results for the two subtasks and their average performance. Bold values indicate the best results among models that use only automatically extracted features.

For ACE04, the baseline outperforms Katiyar and Cardie (2017) by ∼2% in both tasks. This improvement can be explained by the use of: (i) multi-label head selection, (ii) CRF-layer and (iii) character level embeddings. Compared to Miwa and Bansal (2016), who rely on NLP tools, the baseline performs within a reasonable margin (less than 1%) on the joint task. On the other hand, Li et al. (2017) use the same model for the ADE biomedical dataset, where we report a 2.5% overall improvement. This indicates that NLP tools are not always accurate for various contexts. For the CoNLL04 dataset, we use two evaluation settings. We use the *relaxed* evaluation similar to Gupta et al. (2016); Adel and Schütze (2017) on the EC task. The baseline model outperforms the state-of-the-art models that do not rely on manually extracted features (>4% improvement for both tasks), since we directly model the whole sentence, instead of just considering pairs of entities. Moreover, compared to the model of Gupta et al. (2016) that relies on complex features, the baseline model performs within a margin of 1% in terms of overall F_1 score. We also report NER results on the same dataset and improve overall F_1 score with ∼1% compared to Miwa and Sasaki (2014), indicating that our automatically extracted features are more informative than the hand-crafted ones. These automatically extracted features exhibit their performance improvement mainly due to the shared LSTM layer that learns to automatically generate feature representations of entities and their corresponding relations within a single model. For the DREC dataset, we use two evaluation methods. In the *boundaries* evaluation, the baseline has an improvement of ∼3% on both tasks compared to Bekoulis et al. (2018a), whose quadratic scoring layer complicates NER.

Table 1 and Fig. 2 show the effectiveness of the adversarial training on top of the baseline model. In all of the experiments, AT improves the predictive performance of the baseline model in the joint setting. Moreover, as seen in Fig. 2, the performance of the models using AT is closer to maximum even from the early training epochs. Specifically, for ACE04, there is an improvement in both tasks as well as in the overall F_1 performance (0.4%). For CoNLL04, we note an improvement in the overall F_1 of 0.4% for the EC and 0.8% for the NER tasks, respectively. For the DREC dataset, in both settings, there is an overall improvement of ∼1%. Figure 2 shows that from the first epochs, the model obtains its maximum performance on the DREC validation set. Finally, for ADE, our AT model beats the baseline F_1 by 0.7%.

Our results demonstrate that AT outperforms the neural baseline model consistently, considering our experiments across multiple and more diverse datasets than typical related works. The im-

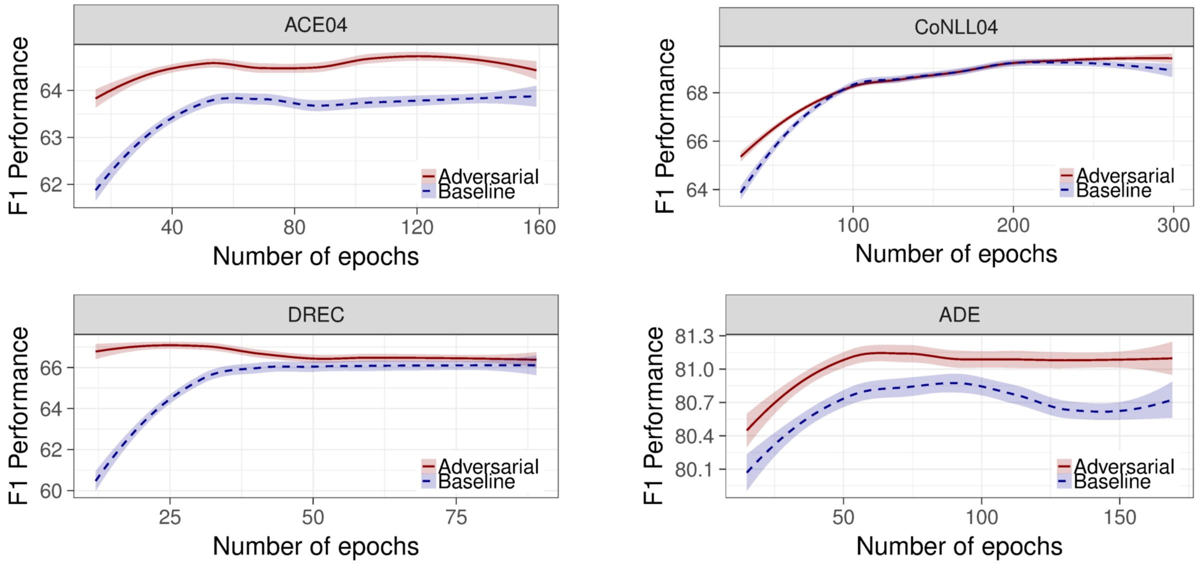

Figure 2: F_1 performance of the baseline and the AT models on the validation sets from 10-30 epochs onwards depending on the dataset. The smoothed lines (obtained by LOWESS smoothing) model the trends and the 95% confidence intervals.

provement of AT over our baseline (depending on the dataset) ranges from ∼0.4% to ∼0.9% in terms of overall F_1 score. This seemingly small performance increase is mainly due to the limited performance benefit for the NER component, which is in accordance with the recent advances in NER using neural networks that report similarly small gains (e.g., the performance improvement in Ma and Hovy (2016) and Lample et al. (2016) on the CoNLL-2003 test set is 0.01% and 0.17% F_1 percentage points, while in the work of Yasunaga et al. (2018), a 0.07% F_1 improvement on CoNLL-2000 using AT for NER is reported). However, the relation extraction performance increases by ∼1% F_1 scoring points, except for the ACE04 dataset. Further, as seen in Fig. 2, the improvement for CoNLL04 is particularly small on the evaluation set. This may indicate a correlation between the dataset size and the benefit of adversarial training in the context of joint models, but this needs further investigation in future work.

6 Conclusion

We proposed to use adversarial training (AT) for the joint task of entity recognition and relation extraction. The contribution of this study is twofold: (i) investigation of the consistent effectiveness of AT as a regularization method over a multi-context baseline joint model, with (ii) a large scale experimental evaluation. Experiments show that AT improves the results for each task separately, as well as the overall performance of the baseline joint model, while reaching high performance already during the first epochs of the training procedure.

Acknowledgments

We would like to thank the anonymous reviewers for the time and effort they spent in reviewing our work, and for their valuable feedback.

References

Heike Adel and Hinrich Schütze. 2017. Global normalization of convolutional neural networks for joint entity and relation classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference for Learning Representations*, San Diego, USA.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2017. Reconstructing the house from the ad: Structured prediction on real estate classifieds. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 2, Short Papers)*, pages 274–279, Valencia, Spain.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018a. An attentive neural architecture for joint segmentation and parsing and its application to real estate ads. *Expert Systems with Applications*, 102:100–112.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018b. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, volume 2, page 1, Lisbon, Portugal.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*, San Diego, USA.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of Biomedical Informatics*, 45(5):885–892.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, San Diego, USA.

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

Fei Li, Meishan Zhang, Guohong Fu, and Donghong Ji. 2017. A neural joint model for entity and relation extraction from biomedical text. *BMC Bioinformatics*, 18(1):1–11.

Fei Li, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Joint models for extracting adverse drug events from biomedical text. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2838–2844, New York, USA. IJCAI/AAAI Press.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, USA.

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

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1858–1869, Doha, Qatar. Association for Computational Linguistics.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *Proceedings of the International Conference on Learning Representations*, Toulon, France.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 1–8, Boston, USA. Association for Computational Linguistics.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634, Beijing, China.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*, Banff, Canada.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1778–1783, Copenhagen, Denmark. Association for Computational Linguistics.

Michihiro Yasunaga, Jungo Kasai, and Dragomir Radev. 2018. Robust multilingual part-of-speech tagging via adversarial training. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, USA.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 1, Long Papers)*, pages 665–676, Valencia, Spain.

Suncong Zheng, Yuexing Hao, Dongyuan Lu, Hongyun Bao, Jiaming Xu, Hongwei Hao, and Bo Xu. 2017. Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing*, 257:59–66.

Structured Multi-Label Biomedical Text Tagging via Attentive Neural Tree Decoding

Gaurav Singh
University College London
gaurav.singh.15@ucl.ac.uk

James Thomas
University College London
james.thomas@ucl.ac.uk

Iain J. Marshall
King's College London
iain.marshall@kcl.ac.uk

John Shawe-Taylor
University College London
j.shawe-taylor@ucl.ac.uk

Byron C. Wallace
Northeastern University
b.wallace@northeastern.edu

Abstract

We propose a model for tagging unstructured texts with an arbitrary number of terms drawn from a tree-structured vocabulary (i.e., an ontology). We treat this as a special case of sequence-to-sequence learning in which the decoder begins at the root node of an ontological tree and recursively elects to expand child nodes as a function of the input text, the current node, and the latent decoder state. In our experiments the proposed method outperforms state-of-the-art approaches on the important task of automatically assigning MeSH terms to biomedical abstracts.

1 Introduction

We consider the task of multilabel text annotation, where labels are drawn from an ontology. We are motivated by problems in biomedical NLP (Zweigenbaum et al., 2007; Demner-Fushman et al., 2016). Specifically, scientific abstracts in this domain are typically associated with multiple Medical Subject Heading (MeSH) terms. MeSH is a controlled, hierarchically structured vocabulary that facilitates semantic labeling of texts at varying levels of granularity. This in turn supports semantic indexing of biomedical literature, thus facilitating improved search and retrieval.[1]

At present, MeSH annotation is largely performed manually by highly skilled annotators employed by the National Library of Medicine (NLM). Automating this annotation task is thus highly desirable, and there have been considerable efforts to do so. The BIOASQ[2] challenge, in particular, concerns MeSH annotation, and competitive systems have emerged from this in past years (Liu et al., 2014; Tsoumakas et al., 2013); these constitute baseline approaches in the present work.

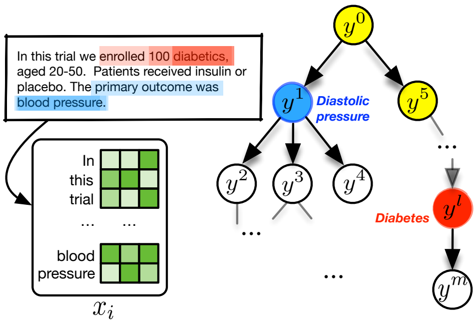

Figure 1: Illustration of the proposed Neural Tree Decoding (NTD) model. Input text is encoded, and a decoder then conditionally traverses the label tree to select all relevant nodes to apply, with node-wise attention induced over the input text.

More generally, MeSH annotation is a specific instance of multi-label classification, which has received substantial attention in general (Elisseeff and Weston, 2002; Fürnkranz et al., 2008; Read et al., 2011; Bhatia et al., 2015; Daumé III et al., 2017; Chen et al., 2017; Jernite et al., 2016). Our work differs from these prior efforts in that MeSH tagging involves *structured* multi-label classification: the label space is a tree[3] in which nodes represent nested semantic concepts, and the specificity of these increases with depth.

Past efforts in multi-label classification have considered hierarchical and tree-based approaches for tagging (Jernite et al., 2016; Beygelzimer et al., 2009; Daumé III et al., 2017), but these have not assumed a *given* structured label space; instead, these efforts have attempted to *induce* trees to improve inference efficiency. By contrast, we propose to explicitly capitalize on a known output structure codified here by the target ontology from which tags are drawn. We realize this by recursively traversing the tree to make (conditional) bi-

---

[1] This problem also resembles tagging clinical notes with ICD codes (Mullenbach et al., 2018).

[2] `http://bioasq.org/`

[3] Technically, MeSH comprises multiple trees, but we join these by insertion of an overarching root node.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2837–2842
Brussels, Belgium, October 31 - November 4, 2018.

nary tag application predictions.

The contribution of this work is a neural sequence-to-sequence (*seq2seq*) model (Bahdanau et al., 2014) for structured multi-label classification. Our approach entails encoding the input text to be tagged using an RNN, and then *decoding into the ontological output space*. This involves a tree traversal beginning at the root of the tree. At each step, the decoder decides whether to 'expand' children as a function of a hidden state vector, node embeddings, and induced attention weights over the input text. This approach is schematized in Figure 1. Expanded nodes are added to the predicted tag set. This process is repeated recursively until either leaf nodes are reached or no children are selected for expansion. This neural tree decoding (NTD) model outperforms state-of-the-art models for MeSH tagging.

2 Model

Overview. Our model is an instance of an encoder-decoder architecture. For the encoder, we adopt a standard Gated Recurrent Unit (GRU) network (Cho et al., 2014a), which yields hidden states for the tokens comprising an input document. The decoder network consumes these outputs and begins at the root of the ontological tree. It induces an attention distribution over encoder states, which is used together with the current decoder state vector to inform which (if any) of its immediate children are applicable to the input text (Figure 1). This decoding process proceeds recursively for all children deemed relevant. Below we provide more in-depth technical detail regarding the constituent modules.

The encoder (ENC) consumes as input a raw sequence of words, here composing an abstract. These are passed through an embedding layer, producing a sequence of word embeddings x (for clarity we omit a document index here), which are then passed through a GRU (Cho et al., 2014b) to obtain a sequence of hidden vectors $h = \{h_0, \cdots, h_{|x|-1}\}$, where $h_t = \text{GRU}(x_t, h_{t-1})$.

These are then passed to our neural tree decoder, which is responsible for tagging the encoded text with an arbitrary number of terms from the label tree, i.e., sequences in the structured output space. This module traverses the label space top-down, beginning at the root, thus exploiting the concept hierarchy codified by the tree structure.

At each step in the decoding process, the decoder will be positioned at a particular node in the tree n. Children — immediate descendents — of this node are then considered for expansion in turn, based on a hidden state vector s_n, and a context vector c_n. Both of these are initialized to zero vectors and recursively updated during traversal, i.e., as nodes are selected for expansion (and hence added to the predicted tag set). More specifically, the context vector that informs the decision to expand node v in the label hierarchy from its parent node n is a weighted sum of the encoder hidden states h, where weights reflect induced attention over inputs, conditioned on n. That is:

$$c_n = \sum_j \alpha_{nj} h_j \tag{1}$$

where

$$\alpha_{nj} = \frac{exp\{a(s_n, h_j)|\theta_n\}}{\sum_l exp\{a(s_n, h_l)|\theta_n\}} \tag{2}$$

and a is a simple multi-layer perceptron (MLP), with node-specific parameters θ_n. Here both sums range over the length of the input text.

Given c_n, we then estimate the probability that child label v is applicable to the current input text as a function of the decoder state vector (s_n), the current context vector (c_n) and the decoder parameters. In particular, this is realized via a standard linear layer with sigmoid activations, parameterized by a weight matrix W comprising independent weight vectors for each output node v. Thus the score for a particular output node v is $\sigma(W_v \cdot [s_n, c_n])$, where W_v denotes the weight vector for output node v.

Pseudocode for the training and decoding procedures are presented in Algorithm 1. In the NODELOSS function, n denotes a particular node. The set of hidden vectors induced by the encoder (corresponding to the inputs) are denoted by h, s is the hidden state of the decoder, and y is the reference label (this encodes a path in the output tree). We assume the decoder, DEC, consumes input representations, a node index and a hidden state and yields a context vector for n, c_n and an updated state vector s_n; in our case the latter is implemented via a GRU. The advantage of using an RNN during decoding is that this allows the exploitation of learned, distributed hidden representations of partial tree paths, which inform node-wise attention and subsequent predictions.

Algorithm 1 RecursiveTreeDecoding

```
function NodeLoss($n$, $h$, $s$, $y$)
    $l_n \leftarrow 0$
    $c_n, s_n \leftarrow$ DEC($h$, $n$, $s$)
    for each child $v \in$ children($n$) do
        $\hat{y}_v \leftarrow \sigma(W_v \cdot [s_n, c_n])$
        $p_v \leftarrow \propto$ depth in tree
        $B_v \sim$ Ber($p_v$)
        if $B_v$ then
            $l_n \leftarrow l_n + \mathcal{L}(\hat{y}_v, y)$
        if $\hat{y}_v > \tau$ then
            $l_n \leftarrow l_n$ + NodeLoss($v$, $h$, $s_n$, $y$)
    return $l_n$

function Train($x$, $y$, $\alpha$, epochs)
    $\theta \leftarrow$ INIT($\theta$)
    $e \leftarrow 0$
    while $e <$ epochs do
        for each instance $x_i \in x$ do
            $h_i \leftarrow$ ENC($x_i$)
            $s_0 \leftarrow \mathbf{0}$
            $l_i \leftarrow$ NodeLoss(ROOT, $h_i$, $s_0$, $y_i$)
            $\Delta\theta \leftarrow$ BACKPROP($l_i$)
            $\theta \leftarrow \theta + \alpha\Delta\theta$
        $e \leftarrow e + 1$
    return $\theta$
```

Incurring loss for all nodes along the path specified by y would place a disproportionate amount of emphasis on correctly applying terms that are 'higher' in the ontology, as loss will be propagated for the initial predictions concerning the application of these and then also, due to recursive application, for all of their children (and so on). Thus we only incur (and hence backpropagate) loss for a node v stochastically, according to a Bernoulli distribution B with parameter p_v. We set p_v to be proportional to the depth of node v in the tree such that we are likely to incur larger loss for deeper (rarely occurring) nodes. We operationalize this as: $p_v = \mathit{min}(1, 0.5 + \frac{m}{f_v})$, where m is the count corresponding to the least frequently observed node in the training corpus and f_v is the count for node v. In Section 4 we demonstrate the benefit of this approach.

At train time we use *teacher forcing* (Williams and Zipser, 1989) during decoding. That is, we revert the model back to the correct (training) tree subsequence when it goes off-course, and continue decoding from there. We have elided this detail from the pseudocode for clarity.

3 Experimental setup

Below we describe experimental details concerning our implementation, datasets and baselines. Code and data to reproduce our results is available at `https://github.com/gauravsc/NTD`.

3.1 Implementation Details

We limited the vocabulary to the $50,000$ most frequent words. Word embeddings were initialized to pre-trained vectors induced via word2vec, trained over a large set of abstracts indexed on PubMed.[4] Ontology node embeddings were pre-trained using DeepWalk (Perozzi et al., 2014), fit over PubMed.

3.2 Dataset

Our dataset comprises abstracts of articles describing randomized controlled trials (RCTs) from PubMed along with their MeSH terms. The MeSH annotations were manually applied by professionals at the National Library of Medicine (NLM). The label space underlying MeSH terms is codified by a publicly available ontology.[5]

We split this dataset into disjoint sets for training/development and final evaluation (Table 1). We further separated the former into train, validation and development test subsets, to refine our approach. For our final evaluation we used a held-out set of 10,000 abstracts that were not seen in any way during model development and/or hyper-parameter tuning. We performed extensive hyper-parameter tuning for the baseline models to ensure fair comparison; details regarding this tuning are provided in the Appendix.

3.3 Baselines

We compare our proposed approach to three baselines, including two prior winners of the annual BioASQ challenge, which includes an automated MeSH annotation task. However, it is important to note that we used a different (and considerably smaller) dataset in the current work, as compared to the corpus used in the BioASQ challenge.

LSSI (Tsoumakas et al., 2013) use an approach that involves predicting both the number of terms and which to apply to a given abstract. They use linear models for both tasks, which operate over TF-IDF representations of abstracts. Specifically, they train a regressor to predict k, the number of MeSH terms to be applied to an abstract. Simultaneously, a binary linear SVM is trained independently for each MeSH term appearing in the train set. At test time, these SVMs provide scores for each term and the top $\hat{k}$ terms are applied, where $\hat{k}$ is the estimate from the aforementioned regressor.

---

[4] A repository of biomedical literature.

[5] `https://meshb-prev.nlm.nih.gov/treeView`

Train	20000
Validation	4000
Dev test	18884
Test (held-out)	10000
Mean MeSH terms per article	15.33
Total unique MeSH terms	27892
Unique MeSH terms in dataset	3781

Table 1: Dataset statistics.

Method	Precision	Recall	F1	SD
LSSI	0.326	0.293	0.309	1.518
UIUC	0.236	**0.388**	0.291	1.433
DML	0.378	0.223	0.275	1.516
NTD-d	**0.434**	0.235	0.299	1.209
NTD-s	0.425	0.265	**0.327**	**1.130**

Table 2: Results on the held-out test dataset. SD refers to *semantic distance*, defined in Eq. 3.

UIUC (Liu et al., 2014) uses a learning-to-rank model to identify the top MeSH terms for an abstract from a candidate set of terms, which is obtained from the nearest neighbours of the abstract. Additionally, one SVM classifier is trained for each of the MeSH terms (similar to the above approach), and scores for each are used to obtain additional terms to be added to the candidate set. In the end, a threshold (tuned on the validation set) is used to select the final set of terms to be assigned.

Finally, we consider a deep multilabel classification model **DML** (Rios and Kavuluru, 2015) that takes as input unstructured abstracts and activates the output nodes corresponding to the relevant MeSH terms. In brief, embedded tokens are fed through a CNN to induce a vector representation, which is then passed on to the dense output layer. Finally, this is passed through a sigmoid activation function. Note that this model exploits the same pre-trained word embeddings as our model does.

3.4 Evaluation metrics

We first evaluate model performance via output node-wise precision, recall and F1 measure. However, these metrics are overly strict in the sense that a model will be penalized equally for all mistakes, regardless of whether they are nearby or far from the target in the label tree. This is problematic because whether to apply a specific MeSH term or its immediate parent may be somewhat subjective in practice. To quantify this, and to explore the extent to which explicitly decoding into the target label space yields improved predictions, we also consider a measure that we refer to as *semantic distance* (SD):

$$\text{SD} = \frac{1}{|\mathcal{Y}|} \sum_{u \in \mathcal{Y}} \min_{v \in \hat{\mathcal{Y}}} \mathit{dist}(u, v) \qquad (3)$$

where $\mathcal{Y}$ and $\hat{\mathcal{Y}}$ are the sets of target and predicted terms respectively, and *dist* is a function that returns the shortest distance between two nodes in the label ontology tree. The idea is that this penalizes less for 'near misses'. Thus if a model fails to apply a particular tag t, but does apply one near to t in the label tree, then it is penalized less.[6] We hypothesize that our model will improve results markedly with respect to this metric, given our exploitation of the tree structure.

As in the case of recall, SD can be 'gamed': one can achieve a perfect score by predicting that all nodes apply to a given abstract. Thus this is only meaningful alongside complementary metrics like F1.

4 Results

Results on the test set (which was completely held out during development) are reported in Table 2. The proposed Neural Tree Decoding model with stochastic backpropagation (NTD-s) bests the most competitive baseline (LSSI) in F1 score by over 2 points.

To explore the effect of backpropagating loss from nodes in proportion to their depth in the ontology, we also include results for a deterministic variant that does not do this, NTD-d. This version does not perform as well, demonstrating the utility of the proposed training approach.

The metrics reported thus far do not account of the structure in the output space. We thus additionally report results with respect to the the semantic distance (SD) metric (Eq. 3). We observe a marked performance increase of $\sim$21% over the best performing baseline. This is intuitive given that we are explicitly decoding into the label tree structure, and demonstrates the ability of our model to learn the ontological structure, thereby predicting semantically appropriate terms.

---

[6] This metric is equivalent to the sum of two metrics ("divergent path to gold standard" and "divergent path to prediction") defined in (Perotte et al., 2013).

5 Conclusions, Discussion & Limitations

We developed a neural attentive sequence tree decoding model for structured multilabel classification where labels are drawn from a known ontology. The proposed method can decode an input text into a tree of labels, effectively using the structure in the output space. We demonstrated that this model outperformed SOTA approaches for the important task of tagging biomedical abstracts with Medical Subject Heading (MeSH) terms on a modestly sized training corpus. Code and data to reproduce these results are available at `https://github.com/gauravsc/NTD`.

One limitation of our model is that it is comparatively slow, due to having to traverse the tree structure during decoding. Prediction speed may not be a major issue in practice, as articles on PubMed could be batch tagged nightly as they arrive. However, slow decoding also means lengthy training (see Appendix, section A.2 for details). For this reason we have here used a modest training set of ∼20k abstracts, which is smaller than corpora used in prior work on this task. Given the relative expressiveness of our model, we expect it to benefit substantially from additional training data, moreso than the simpler baseline architectures. But at present this is only a conjecture.

In future work we thus hope to apply this model to larger datasets, and to address the efficiency issue. Concerning the latter, sibling subtrees may be traversed in parallel, conditioned on the hidden state of their parent. Another promising direction would be to move to convolutional encoder and decoder architectures, designing the latter in a way similarly capitalizes on the label space tree structure.

6 Acknowledgements

JT and GS acknowledge support from Cochrane via the Transform project. BCW was supported by the National Library of Medicine (NLM) of the National Institutes of Health (NIH), grant R01LM012086. IJM acknowledges support from the MRC (UK), through its grant MR/N015185/1.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Alina Beygelzimer, John Langford, Yuri Lifshits, Gregory Sorkin, and Alex Strehl. 2009. Conditional probability tree estimation analysis and algorithms. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*.

Sheng Chen, Akshay Soni, Aasish Pappu, and Yashar Mehdad. 2017. Doctag2vec: An embedding based multi-label learning approach for document tagging. In *Proceedings of the Workshop on Representation Learning for NLP*. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014a. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014b. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Hal Daumé III, Nikos Karampatziakis, John Langford, and Paul Mineiro. 2017. Logarithmic time one-against-some. In *International Conference on Machine Learning*.

D Demner-Fushman, N Elhadad, et al. 2016. Aspiring to unintended consequences of natural language processing: A review of recent developments in clinical and consumer-generated text processing. *IMIA Yearbook*.

André Elisseeff and Jason Weston. 2002. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems*.

Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. 2008. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2).

Yacine Jernite, Anna Choromanska, and David Sontag. 2016. Simultaneous learning of trees and representations for extreme classification, with application to language modeling. *arXiv preprint arXiv:1610.04658*.

Ke Liu, Junqiu Wu, Shengwen Peng, Chengxiang Zhai, and Shanfeng Zhu. 2014. The fudan-uiuc participation in the bioasq challenge task 2a: The antinomyra system. *Risk*, 129816.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *North American Chapter of the Association for Computational Linguistics*.

Adler Perotte, Rimma Pivovarov, Karthik Natarajan, Nicole Weiskopf, Frank Wood, and Noémie Elhadad. 2013. Diagnosis code assignment: models and evaluation metrics. *Journal of the American Medical Informatics Association*, 21(2).

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3).

Anthony Rios and Ramakanth Kavuluru. 2015. Convolutional neural networks for biomedical text classification: application in indexing biomedical articles. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Health Informatics*. ACM.

Grigorios Tsoumakas, Manos Laliotis, Nikos Markantonatos, and Ioannis Vlahavas. 2013. Large-scale semantic indexing of biomedical publications at BioASQ. In *BioASQ Workshop*.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2).

Pierre Zweigenbaum, Dina Demner-Fushman, Hong Yu, and Kevin B Cohen. 2007. Frontiers of biomedical text mining: current progress. *Briefings in bioinformatics*, 8(5).

Deep Exhaustive Model for Nested Named Entity Recognition

Mohammad Golam Sohrab† and **Makoto Miwa**†, ‡
†Artificial Intelligence Research Center (AIRC),
National Institute of Advanced Industrial Science and Technology (AIST), Japan
‡Toyota Technological Institute, Japan
sohrab.mohammad@aist.go.jp, makoto-miwa@toyota-ti.ac.jp

Abstract

We propose a simple deep neural model for nested named entity recognition (NER). Most NER models focused on flat entities and ignored nested entities, which failed to fully capture underlying semantic information in texts. The key idea of our model is to enumerate all possible regions or spans as potential entity mentions and classify them with deep neural networks. To reduce the computational costs and capture the information of the contexts around the regions, the model represents the regions using the outputs of shared underlying bidirectional long short-term memory. We evaluate our exhaustive model on the GENIA and JNLPBA corpora in biomedical domain, and the results show that our model outperforms state-of-the-art models on nested and flat NER, achieving 77.1% and 78.4% respectively in terms of F-score, without any external knowledge resources.

1 Introduction

Named entity recognition (NER) is a task of finding entities with specific semantic types such as *Protein*, *Cell*, and *RNA* in text. NER is generally treated as a sequential labeling task, where each token is tagged with a label that corresponds to its surrounding entity. However, when entities overlap or are nested within one another, treating the task as a sequential labeling task becomes difficult because an individual token can be included in several entities and defining a label for each token can be difficult. For example, in the following phrase from the GENIA corpus (Kim et al., 2004), four levels of nested entities occur and the token "*IL-2*" is a *Protein* on its own, and it is also a part of two other *Protein*s and one *DNA*.

[[[[*IL-2*]^{Protein} *receptor*]^{Protein} *(IL-2R) alpha chain*]^{Protein} *gene*]^{DNA}

NER has drawn considerable attention as the first step towards many natural language processing (NLP) applications including relation extraction (Miwa and Bansal, 2016), event extraction (Feng et al., 2016), co-reference resolution (Fragkou, 2017; Stone and Arora, 2017), and entity linking (Gupta et al., 2017). Much work on NER, however, has ignored nested entities and instead chosen to focus on the non-nested entities, which are also referred to as flat entities. Only a few studies target the nested named entity recognition (Muis and Lu, 2017; Lu and Roth, 2015; Finkel and Manning, 2009).

Recent successes in neural networks have shown impressive performance gains on flat named entity recognition in several domains (Lample et al., 2016; Ma and Hovy, 2016; Gridach, 2017; Strubell et al., 2017). Such models achieve state-of-the-art results without requiring any hand crafted features or external knowledge resources. In contrast, fewer approaches have emphasized the nested entity recognition problem. Existing approaches to nested NER (Shen et al., 2003; Alex et al., 2007; Finkel and Manning, 2009; Lu and Roth, 2015; Xu et al., 2017; Muis and Lu, 2017) are mostly feature-based and thus suffer from heavy feature engineering. In this paper, we present a novel neural exhaustive model that reasons over all the regions within a specified maximum size. The model represents each region using the outputs of bidirectional long short-term memory (LSTM) by combining the boundary representation of a region and inside representation that simply treats all the tokens in a region equally by taking the average of LSTM outputs corresponding to tokens inside the region. It then classifies regions into their entity types or non-entity. Unlike the existing model that relies on token-level labels, our model directly employs an entity type as the label of a region. The model

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2843–2849
Brussels, Belgium, October 31 - November 4, 2018.

does not rely on any external knowledge resources or NLP tools like part-of-speech taggers. We evaluated our model on the GENIA and JNLPBA corpora in the biomedical domain and the model achieved 77.1% and 78.4% respectively in terms of F-score, which are the new state-of-the-art performances on the corpora.

2 Neural Exhaustive Model

The proposed model exhaustively considers all possible regions in a sentence using a single neural network; we thus call the model neural exhaustive model. Our model is built upon a shared bidirectional LSTM layer. The model enumerates all possible regions or spans that can include all the nested entities. It then represent the regions by using the outputs of the LSTM layer and detect the entities from the regions. The number of possible regions depend on the predefined maximum size. In this section, we describe the architecture of our neural exhaustive model in detail, which is summarized in Figure 1.

2.1 Word Representation

We represent each word by concatenating word embeddings and character-based word representations. Pre-trained word embeddings are used to initialize word embeddings (Chiu et al., 2016). For the character-based word representations, we encode the character-level information of each word following the successes of Ma and Hovy (2016) and Lample et al. (2016) that utilized character embeddings for the flat NER task. The embedding of each character in a word is randomly initialized. We feed the sequence of character embeddings comprising a word to a bidirectional LSTM layer and concatenate the forward and backward output representations to obtain the word representations.

2.2 Exhaustive Combination using LSTM

Given an input sentence sequence $X = \{x_1, x_2, ...x_n\}$, where x_i denotes the i-th word and n denotes the number of words in the sentence sequence, the distributed embeddings of words and characters are fed into a bidirectional LSTM layer that computes the hidden vector sequence in forward $\overrightarrow{\mathbf{h}} = \left\{\overrightarrow{\mathbf{h}_1}, \overrightarrow{\mathbf{h}_2}, \ldots, \overrightarrow{\mathbf{h}_n}\right\}$ and backward $\overleftarrow{\mathbf{h}} = \left\{\overleftarrow{\mathbf{h}_1}, \overleftarrow{\mathbf{h}_2}, \ldots, \overleftarrow{\mathbf{h}_n}\right\}$ manners. We concatenate the forward and backward outputs as $\mathbf{h}_i = \left[\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}\right]$, where $[;]$ denotes concatenation.

With the LSTM output $\mathbf{h}_i$, our exhaustive model shares the underlying representations of all possible regions by exhaustive combination. We generate all possible regions with the sizes less than or equal to the maximum region size L. We use a $region(i, j)$ to represent the region from i to j inclusive, where $1 \leq i < j \leq n$ and $j - i < L$.

2.3 Region Representation and Classification

We represent the region by separating the region into the boundary and inside representations. The boundary representation is important to capture the contexts surrounding the region. We simply rely on the outputs of the bidirectional LSTM layer corresponding to the boundary words of a target region for this purpose. For the inside representation, we simply average the outputs of the bidirectional LSTM layer in the region to treat them equally. We include the outputs for the boundary words to guarantee that the inside representation has corresponding outputs. In summary, we obtain the representation $\mathbf{R}(i, j)$ of the $region(i, j)$ as follows:

$$\mathbf{R}(i, j) = \left[\mathbf{h}_i; \frac{1}{j - i + 1} \sum_{k=i}^{j} \mathbf{h}_k; \mathbf{h}_j\right]. \qquad (1)$$

We then feed the representation of each segmented region to a rectified linear unit (ReLU) as an activation function. Finally, the output of the activation layer is passed to a softmax output layer to classify the region into a specific entity type or non-entity.

The exhaustive model represents all possible regions based on maximum entity length and classify all of them. The overall number of classifications for each sentence in the exhaustive model is in $O(lmn)$, where l is a total number of words in the sentence, m is the maximum entity length and n is the total number of possible entity types. Finkel and Manning (2009) and Alex et al. (2007) proposed featured-based approaches for handling nested NER. The time complexity of their models are expensive, i.e., cubic in the number of the words in the sentence. The exhaustive approach is fast since we run the LSTM once and the classifications can be performed in parallel on the combinations created from the LSTM outputs.

The exhaustive model classify each region independently unlike word-level taggers. This makes the model flexible so that it can incorporate

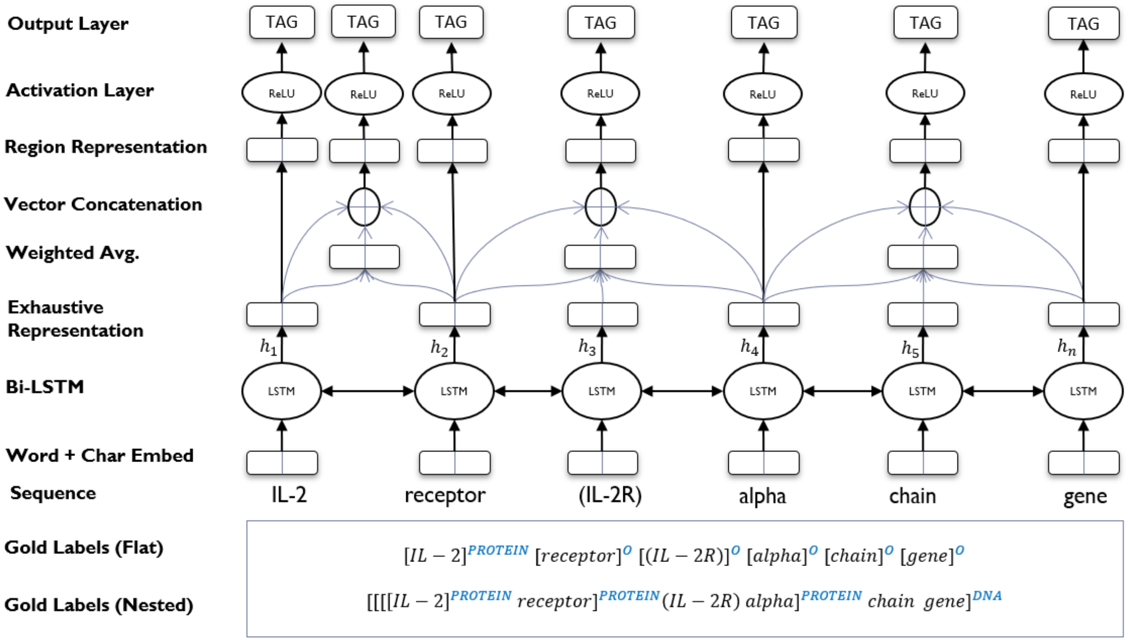

Figure 1: Architecture of the proposed neural exhaustive model. The model considers all possible regions up to a maximum size, but we depict here only a small subset for brevity. "*IL-2*", "*IL-2 receptor*", "*IL-2 receptor (IL-2R) alpha*", and "*IL-2 receptor (IL-2R) alpha chain gene*" are nested entities.

phrase-level dictionary information directly and we can tune biases for each type unlike CRF. We leave this evaluation to our future work.

3 Experimental Settings

We evaluated our exhaustive model on GENIA[1] (Kim et al., 2003) and JNLPBA[2] (Kim et al., 2004) datasets to provide empirical evidence for the effectiveness of our model both in nested and flat NER. Table 1 shows the statistics of GENIA dataset.

Our model was implemented in Chainer[3] deep learning framework. We employed pre-trained word embeddings that were trained on MEDLINE abstracts (Chiu et al., 2016), which included 200-dimensional embeddings of 2,231,686 vocabulary. We used ADAM (Kingma and Ba., 2015) for learning with a mini-batch size of 100. We used the same hyper-parameters in all the experiments; we set the dimension of word embedding to 200, the dimension of character embedding to 25, the hidden layer size to 200, the gradient clipping to 5, and the ADAM hyper-parameters to its default values (Kingma and Ba., 2015).

Item	Train	Dev	Test
Documents	1,599	189	212
Sentences	15,022	1,669	1,855
Split(%)	81	9	10
DNA	7,921	1061	1,283
RNA	730	140	117-
protein	29,032	2,338	3,098
cell line	3,149	340	460
cell type	6,021	563	617
outermost entity	42,462	4,020	4,942
nested level	4	3	3
entity avg. length	2.87	3.13	2.93
multi-token entity	33951	3554	4203
overall nested entity	8301	803	1202
overall entity	46,853	4,442	5,575

Table 1: Statistics of GENIA

To deeply understand the model parameters, we compared the models in different regions. We chose the maximum region size from 3, 6, 8 and 10. We also employed different region representation. We tried only the boundary representation (boundary), only the inside representation (inside), and our region representation (boundary+inside).

We employed precision, recall, and F-score to

---

[1] `http://www.geniaproject.org/genia-corpus/term-corpus`

[2] `http://www.nactem.ac.uk/tsujii/GENIA/ERtask/report.html`

[3] `https://chainer.org/`

Model	P(%)	R(%)	F(%)
Exhaustive Model	**93.2**	64.0	**77.1**
Ju et al. (2018)	78.5	**71.3**	74.7
Katiyar and Cardie	76.7	71.1	73.8
Muis and Lu (2017)	75.4	66.8	70.8
Lu and Roth (2015)	72.5	65.2	68.7
Finkel and Manning	75.4	65.9	70.3

Table 2: Performance comparison of the state-of-the-art nested NER models on the test dataset.

Entity Level	P(%)	R(%)	F(%)
Single-token	91.6	58.4	69.9
Multi-token	95.9	65.8	77.9
Top Level	92.7	69.8	79.3
Nested	94.3	59.3	72.7
All entities	93.2	64.0	77.1

Table 3: Performances of our model on different entity level on the test dataset.

evaluate our model. We also compared the performances for single-token v.s. multi-token entities and top-level v.s. nested entities.

4 Results and Discussions

4.1 Nested NER

Table 2 shows the comparison of our model with several previous state-of-the nested NER models on the test dataset. Our model outperforms the state-of-the-art models in terms of F-score. Our results on Table 2 is based on bidirectional LSTM with character embeddings and the maximum region size is 10.

Table 3 describes the performances of our model on different entity levels on the test dataset. The model performs well on multi-token and top-level entities. This is interesting because they are often considered difficult for sequential labeling models.

Table 4 shows the performances on the five entity types on the test dataset. We here show the performance by Finkel and Manning (2009) (F&M) for the reference. Our system performs better than their model except for the *RNA* type.

4.2 Ablation Tests

We show the differences in the performance on the development dataset to compare the possible scenarios of the proposed approach and to report the importance of each component in our exhaustive model.

Label	P(%)	R(%)	F(%)	F&M F(%)
DNA	92.6	58.7	**71.8**	65.2
RNA	98.8	57.1	72.4	**74.7**
cell line	94.6	53.1	**67.9**	64.0
cell type	88.4	70.0	**78.1**	67.1
protein	94.1	70.8	**80.8**	73.8

Table 4: Categorical performances on the GENIA test dataset.

Region	Ratio(%)	P(%)	R(%)	F(%)
size = 3	89.6	92.9	69.8	79.5
size = 6	98.9	93.6	66.7	77.5
size = 8	99.4	93.7	66.5	77.6
size = 10	100	93.5	67.6	78.2

Table 5: Performance of our model with different maximum region sizes on the development dataset. Ratio refers to the coverage ratio of entity mentions.

Setting	P(%)	R(%)	F(%)
Bi-LSTM	94.1	65.7	77.1
Bi-LSTM + Character*	93.5	67.6	78.2
Boundary*	94.1	54.3	68.5
Inside*	93.2	46.4	61.2
Boundary+Inside*	93.5	67.6	78.2

Table 6: Performance of our model with different model architectures on the development dataset. $\star$ indicates results using character embeddings.

Label	P(%)	R(%)	F(%)
DNA	95.2	56.8	71.4
RNA	96.1	61.4	75.2
cell line	86.2	44.1	58.8
cell type	96.7	61.5	75.3
protein	97.1	72.2	82.6
overall	96.4	66.8	78.4

Table 7: Categorical and overall performances of the JNLPBA test dataset.

Table 5 shows the coverage ratio and the performance with different maximum region sizes. Since the average entity mention length of GENIA dataset is less than 4, the system can cover almost all the entities for the maximum sizes of 6 or more. The longer maximum region size is desirable to cover all the mentions, but it requires more computational costs. Fortunately, the performance did not degrade with the long maximum region size, despite the fact that it introduces more out-of-entity regions.

Ablations on character embeddings in Table 6 also show the importance of character embeddings. It also shows that both the boundary information and the inside information, i.e., average of the embeddings in a region, are necessary to improve the performance.

4.3 Flat NER

We evaluated our model on JNLPBA as a flat dataset, where nested and discontinuous entities are removed. Table 7 shows the performances of our model on JNLPBA dataset. We compared our result with the state-of-the-art result of Gridach (2017) which achieved 75.8% in F-score, where our model obtained 78.4% in terms of F-score.

5 Related Work

Interests in nested NER detection have increased in recent years, but it is still the case that NER models deals with only one flat level at a time. Zhou et al. (2004) detected nested entities in a bottom-up way. They detected the innermost flat entities and then found other NEs containing the flat entities as substrings using rules derived from the detected entities. The authors reported an improvement of around 3% in the F-score under certain conditions on the GENIA corpus (Collier et al., 1999). Katiyar and Cardie (2018) proposed a neural network-based approach that learns hypergraph representation for nested entities using features extracted from a recurrent neural network (RNN). The authors reported that the model outperformed the existing state-of-the-art feature-based approaches.

Recent studies show that the conditional random fields (CRFs) can significantly produce higher tagging accuracy in flat (Athavale et al., 2016) or nested (stacking flat NER to nested representation) (Son and Minh, 2017) NERs. Ju et al. (2018) proposed a novel neural model to address nested entities by dynamically stacking flat NER layers until no outer entities are extracted. A cascaded CRF layer is used after the LSTM output in each flat layer. The authors reported that the model outperforms state-of-the-art results by achieving 74.5% in terms of F-score. Finkel and Manning (2009) proposed a tree-based representation to represent each sentence as a constituency tree of nested entities. All entities were treated as phrases and represented as subtrees following the whole tree structure and used a CRF-based approach driven by entity-level features to detect nested entities. We demonstrate that the performance can be improved significantly without CRFs, by training an exhaustive neural model that learns which regions are entity mentions and how to best classify the regions.

6 Conclusion

This paper presented a neural exhaustive model that considers all possible regions exhaustively for nested NER. The model obtains the representation of each region from an underlying shared LSTM layer, and it represents the region by concatenating boundary representations of the region and inside representation that averages embeddings of words in the region. It then classifies the region into its entity type or non-entity. The model does not depend on any external NLP tools. In the experiment, we show that our model learns to detect nested named entities from the generated mention candidates of all possible regions. Our exhaustive model outperformed existing models with a significant margin in terms of F-score in both flat and nested NER.

For future work, we would like to investigate the use of region-level information. We also consider modeling the dependencies between regions.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising Nested Named Entities in Biomedical Text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 65–72, Stroudsburg, PA, USA. ACL.

Vinayak Athavale, Shreenivas Bharadwaj, Monik Pamecha, Ameya Prabhu, and Manish Shrivastava. 2016. *Towards Deep Learning in Hindi NER: An approach to tackle the Labelled Data Scarcity*. Cornell University Library.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to Train good Word Embeddings for Biomedical NLP. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 166–174.

N. Collier, H. S. Park, N. Ogata, Y. Tateisi, C. Nobata, T. Ohta, T. Sekimizu, H. Imai, K. Ibushi, and Jun'ichi Tsujii. 1999. The GENIA Project: Corpus-based Knowledge Acquisition and Information Extraction from Genome Research Papers. In *Proceedings of EACL*, pages 171–172. ACL.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A Language-Independent Neural Network for Event Detection. In *Proceedings of the 54th Annual Meeting of the ACL (Volume 2: Short Papers)*, pages 66–71, Berlin, Germany.

Jenny Rose Finkel and Manning Manning, Christopher D. 2009. Nested Named Entity Recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 141–150, Singapore. ACL.

Pavlina Fragkou. 2017. Applying named entity recognition and co-reference resolution for segmenting english texts. *Progress in Artificial Intelligence*, 6(4):325–346.

Mourad Gridach. 2017. Character-level neural network for biomedical named entity recognition. *Journal of biomedical informatics*, 70:85–91.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity Linking via Joint Encoding of Types, Descriptions, and Context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2671–2680, Copenhagen, Denmark. ACL.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A Neural Layered Model for Nested Named Entity Recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459, New Orleans, Louisiana. ACL.

Arzoo Katiyar and Claire Cardie. 2018. Nested Named Entity Recognition Revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871, New Orleans, Louisiana. ACL.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the Bio-Entity Recognition Task at JNLPBA. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. ACL.

Jin-Dong Kim, Yuka Ohta, Tateisi, and Junichi Tsujii. 2003. GENIA corpus– a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl. 1):i180–i182.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the ACL: Human Language Technologies. ACL*, volume 1, pages 260–270, San Diego, California. ACL.

Wei Lu and Dan Roth. 2015. Joint Mention Extraction and Classification with Mention Hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867. ACL.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. pages 1064–1074.

Makoto Miwa and Mohit Bansal. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 1105–1116, Berlin, Germany. ACL.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling Gaps Between Words: Recognizing Overlapping Mentions with Mention Separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2598–2608, Copenhagen, Denmark. ACL.

Dan Shen, Jie Zhang, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2003. Effective Adaptation of a Hidden Markov Model-based Named Entity Recognizer for Biomedical Domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*, pages 49–56, Sapporo, Japan. ACL.

Nguyen Truong Son and Nguyen Le Minh. 2017. Nested Named Entity Recognition Using Multilayer Recurrent Neural Networks. In *Proceedings of PACLING 2017*, pages 16–18, Sedona Hotel, Yangon, Myanmar.

M. Stone and R. Arora. 2017. *Identifying nominals with no head match co-references using deep learning*. CoRR abs/1710.00936.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680, Copenhagen, Denmark. ACL.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A Local Detection Approach for Named Entity Recognition and Mention Detection. In *Proceedings of the 55th Annual Meeting of the ACL (Volume 1: Long Papers)*, volume 1, pages 1237–1247.

Guodong Zhou, Jie Zhang, Jian Su, Dan Shen, and Chewlim Tan. 2004. Recognizing Names in Biomedical Texts: a Machine Learning Approach. *Bioinformatics*, 20(7):1178–1190.

Evaluating the Utility of Hand-crafted Features in Sequence Labelling[*]

Minghao Wu[♠♡†] **Fei Liu**[♠] **Trevor Cohn**[♠]
[♠]The University of Melbourne, Victoria, Australia
[♡]JD AI Research, Beijing, China
`wuminghao@jd.com`
`fliu3@student.unimelb.edu.au`
`t.cohn@unimelb.edu.au`

Abstract

Conventional wisdom is that hand-crafted features are redundant for deep learning models, as they already learn adequate representations of text automatically from corpora. In this work, we test this claim by proposing a new method for exploiting handcrafted features as part of a novel hybrid learning approach, incorporating a feature auto-encoder loss component. We evaluate on the task of named entity recognition (NER), where we show that including manual features for part-of-speech, word shapes and gazetteers can improve the performance of a neural CRF model. We obtain a F_1 of 91.89 for the CoNLL-2003 English shared task, which significantly outperforms a collection of highly competitive baseline models. We also present an ablation study showing the importance of auto-encoding, over using features as either inputs or outputs alone, and moreover, show including the autoencoder components reduces training requirements to 60%, while retaining the same predictive accuracy.

1 Introduction

Deep neural networks have been proven to be a powerful framework for natural language processing, and have demonstrated strong performance on a number of challenging tasks, ranging from machine translation (Cho et al., 2014b,a), to text categorisation (Zhang et al., 2015; Joulin et al., 2017; Liu et al., 2018b). Not only do such deep models outperform traditional machine learning methods, they also come with the benefit of not requiring difficult feature engineering. For instance, both Lample et al. (2016) and Ma and Hovy (2016) propose end-to-end models for sequence labelling task and achieve state-of-the-art results.

Orthogonal to the advances in deep learning is the effort spent on feature engineering. A representative example is the task of named entity recognition (NER), one that requires both lexical and syntactic knowledge, where, until recently, most models heavily rely on statistical sequential labelling models taking in manually engineered features (Florian et al., 2003; Chieu and Ng, 2002; Ando and Zhang, 2005). Typical features include POS and chunk tags, prefixes and suffixes, and external gazetteers, all of which represent years of accumulated knowledge in the field of computational linguistics.

The work of Collobert et al. (2011) started the trend of feature engineering-free modelling by learning internal representations of compositional components of text (e.g., word embeddings). Subsequent work has shown impressive progress through capturing syntactic and semantic knowledge with dense real-valued vectors trained on large unannotated corpora (Mikolov et al., 2013a,b; Pennington et al., 2014). Enabled by the powerful representational capacity of such embeddings and neural networks, feature engineering has largely been replaced with taking off-the-shelf pre-trained word embeddings as input, thereby making models fully end-to-end and the research focus has shifted to neural network architecture engineering.

More recently, there has been increasing recognition of the utility of linguistic features (Li et al., 2017; Chen et al., 2017; Wu et al., 2017; Liu et al., 2018a) where such features are integrated to improve model performance. Inspired by this, taking NER as a case study, we investigate the utility of hand-crafted features in deep learning models, challenging conventional wisdom in an attempt to refute the utility of manually-engineered features. Of particular interest to this paper is the work by Ma and Hovy (2016) where they

---

[*] `https://github.com/minghao-wu/CRF-AE`

[†] Work carried out at The University of Melbourne

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2850–2856
Brussels, Belgium, October 31 - November 4, 2018.

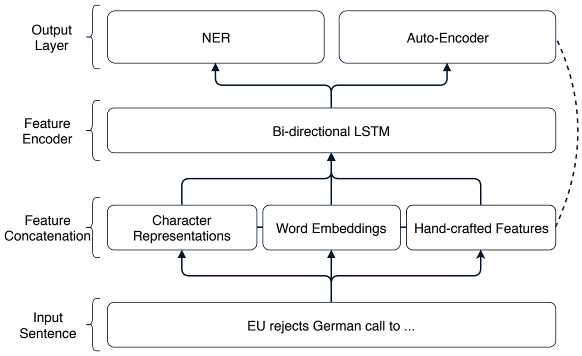

Figure 1: Main architecture of our neural network. Character representations are extracted by a character-level CNN. The dash line indicates we use an auto-encoder loss to reconstruct hand-crafted features.

introduce a strong end-to-end model combining a bi-directional Long Short-Term Memory (Bi-LSTM) network with Convolutional Neural Network (CNN) character encoding in a Conditional Random Field (CRF). Their model is highly capable of capturing not only word- but also character-level features. We extend this model by integrating an auto-encoder loss, allowing the model to take hand-crafted features as input and re-construct them as output, and show that, even with such a highly competitive model, incorporating linguistic features is still beneficial. Perhaps the closest to this study is the works by Ammar et al. (2014) and Zhang et al. (2017), who show how CRFs can be framed as auto-encoders in unsupervised or semi-supervised settings.

With our proposed model, we achieve strong performance on the CoNLL 2003 English NER shared task with an F_1 of 91.89, significantly outperforming an array of competitive baselines. We conduct an ablation study to better understand the impacts of each manually-crafted feature. Finally, we further provide an in-depth analysis of model performance when trained with varying amount of data and show that the proposed model is highly competent with only 60% of the training set.

2 Methodology

In this section, we first outline the model architecture, then the manually crafted features, and finally how they are incorporated into the model.

2.1 Model Architecture

We build on a highly competitive sequence labelling model, namely Bi-LSTM-CNN-CRF, first introduced by Ma and Hovy (2016). Given an input sequence of $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ of length T, the model is capable of tagging each input with a predicted label $\hat{y}$, resulting in a sequence of $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T\}$ closely matching the gold label sequence $\mathbf{y} = \{y_1, y_2, \ldots, y_T\}$. Here, we extend the model by incorporating an auto-encoder loss taking hand-crafted features as in/output, thereby forcing the model to preserve crucial information stored in such features and allowing us to evaluate the impacts of each feature on model performance. Specifically, our model, referred to as Neural-CRF+AE, consists of four major components: (1) a character-level CNN (char-CNN); (2) a word-level bi-directional LSTM (Bi-LSTM); (3) a conditional random field (CRF); and (4) an auto-encoder auxiliary loss. An illustration of the model architecture is presented in Figure 1.

Char-CNN. Previous studies (Santos and Zadrozny, 2014; Chiu and Nichols, 2016; Ma and Hovy, 2016) have demonstrated that CNNs are highly capable of capturing character-level features. Here, our character-level CNN is similar to that used in Ma and Hovy (2016) but differs in that we use a ReLU activation (Nair and Hinton, 2010).[1]

Bi-LSTM. We use a Bi-LSTM to learn contextual information of a sequence of words. As inputs to the Bi-LSTM, we first concatenate the pre-trained embedding of each word $\boldsymbol{w}_i$ with its character-level representation $\boldsymbol{c}_{w_i}$ (the output of the char-CNN) and a vector of manually crafted features $\boldsymbol{f}_i$ (described in Section 2.2):

$$\overrightarrow{\boldsymbol{h}}_i = \overrightarrow{\mathrm{LSTM}}(\overrightarrow{\boldsymbol{h}}_{i-1}, [\boldsymbol{w}_i; \boldsymbol{c}_{w_i}; \boldsymbol{f}_i]) \qquad (1)$$

$$\overleftarrow{\boldsymbol{h}}_i = \overleftarrow{\mathrm{LSTM}}(\overleftarrow{\boldsymbol{h}}_{i+1}, [\boldsymbol{w}_i; \boldsymbol{c}_{w_i}; \boldsymbol{f}_i])\,, \qquad (2)$$

where $[;]$ denotes concatenation. The outputs of the forward and backward pass of the Bi-LSTM is then concatenated $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i]$ to form the output of the Bi-LSTM, where dropout is also applied.

CRF. For sequence labelling tasks, it is intuitive and beneficial to utilise information carried between neighbouring labels to predict the best sequence of labels for a given sentence. Therefore,

---

[1] While the hyperbolic tangent activation function results in comparable performance, the choice of ReLU is mainly due to faster convergence.

x	*U.N.*	*official*	*Ekeus*	*heads*	*for*	*Baghdad*	.
POS	NNP	NN	NNP	VBZ	IN	NNP	.
Word shape	X.X.	xxxx	Xxxxx	xxxx	xxx	Xxxxx	.
Dependency tags	compound	compound	compound	ROOT	prep	pobj	punct
Gazetteer	O	O	PER	O	O	LOC	O
y	B-ORG	O	B-PER	O	O	B-LOC	O

Table 1: Example sentence (top), showing the different types of linguistic features used in this work as additional inputs and auxiliary outputs (middle), and its labelling (bottom).

we employ a conditional random field layer (Lafferty et al., 2001) taking as input the output of the Bi-LSTM $\boldsymbol{h}_i$. Training is carried out by maximising the log probability of the gold sequence: $\mathcal{L}_{CRF} = \log p(\mathbf{y}|\mathbf{x})$ while decoding can be efficiently performed with the Viterbi algorithm.

Auto-encoder loss. Alongside sequence labelling as the primary task, we also deploy, as auxiliary tasks, three auto-encoders for reconstructing the hand-engineered feature vectors. To this end, we add multiple independent fully-connected dense layers, all taking as input the Bi-LSTM output $\boldsymbol{h}_i$ with each responsible for reconstructing a particular type of feature: $\hat{\boldsymbol{f}}_i^t = \sigma(\boldsymbol{W}^t\boldsymbol{h}_i)$ where σ is the sigmoid activation function, t denotes the type of feature, and $\boldsymbol{W}^t$ is a trainable parameter matrix. More formally, we define the auto-encoder loss as:

$$\mathcal{L}_{AE}^t = \sum_{i=0}^{T} \text{XEntropy}(\boldsymbol{f}_i^t, \hat{\boldsymbol{f}}_i^t)\,. \tag{3}$$

Model training. Training is carried out by optimising the joint loss:

$$\mathcal{L} = \mathcal{L}_{CRF} + \sum_t \lambda^t \mathcal{L}_{AE}^t\,, \tag{4}$$

where, in addition to $\mathcal{L}_{CRF}$, we also add the auto-encoder loss, weighted by λ^t. In all our experiments, we set λ^t to 1 for all ts.

2.2 Hand-crafted Features

We consider three categories of widely used features: (1) POS tags; (2) word shape; and (3) gazetteers and present an example in Table 1. While POS tags carry syntactic information regarding sentence structure, the word shape feature focuses on a more fine-grained level, encoding character-level knowledge to complement the loss of information caused by embedding lookup, such as capitalisation. Both features are based on the implementation of spaCy.[2] For the gazetteer feature, we focus on *PERSON* and *LOCATION* and compile a list for each. The *PERSON* gazetteer is collected from U.S. census 2000, U.S. census 2010 and DBpedia whereas GeoNames is the main source for *LOCATION*, taking in both official and alternative names. All the tokens on both lists are then filtered to exclude frequently occurring common words.[3] Each category is converted into a one-hot sparse feature vector $\boldsymbol{f}_i^t$ and then concatenated to form a multi-hot vector $\boldsymbol{f}_i = [\boldsymbol{f}_i^{\text{POS}}; \boldsymbol{f}_i^{\text{shape}}; \boldsymbol{f}_i^{\text{gazetteer}}]$ for the i-th word. In addition, we also experimented with including the label of the incoming dependency edge to each word as a feature, but observed performance deterioration on the development set. While we still study and analyse the impacts of this feature in Table 3 and Section 3.2, it is excluded from our model configuration (not considered as part of $\boldsymbol{f}_i$ unless indicated otherwise).

3 Experiments

In this section, we present our experimental setup and results for name entity recognition over the CoNLL 2003 English NER shared task dataset (Tjong Kim Sang and De Meulder, 2003).

3.1 Experimental Setup

Dataset. We use the CoNLL 2003 NER shared task dataset, consisting of 14,041/3,250/3,453 sentences in the training/development/test set respectively, all extracted from Reuters news articles during the period from 1996 to 1997. The dataset is annotated with four categories of name entities: *PERSON, LOCATION, ORGANIZATION* and *MISC*. We use the `IOBES` tagging scheme, as previous study have shown that this scheme provides a modest improvement to the model performance (Ratinov and Roth, 2009; Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016).

---

[2] `https://spacy.io/`

[3] Gazetteer data is included in the code release.

Model configuration. Following the work of Ma and Hovy (2016), we initialise word embeddings with GloVe (Pennington et al., 2014) (300-dimensional, trained on a 6B-token corpus). Character embeddings are 30-dimensional and randomly initialised with a uniform distribution in the range $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$. Parameters are optimised with stochastic gradient descent (SGD) with an initial learning rate of $\eta = 0.015$ and momentum of 0.9. Exponential learning rate decay is applied every 5 epochs with a factor of 0.8. To reduce the impact of exploding gradients, we employ gradient clipping at 5.0 (Pascanu et al., 2013).

We train our models on a single GeForce GTX TITAN X GPU. With the above hyper-parameter setting, training takes approximately 8 hours for a full run of 40 epochs.

Evaluation. We measure model performance with the official CoNLL evaluation script and report span-level named entity F-score on the test set using early stopping based on the performance on the validation set. We report average F-scores and standard deviation over 5 runs for our model.

Baseline. In addition to reporting a number of prior results of competitive baseline models, as listed in Table 2, we also re-implement the Bi-LSTM-CNN-CRF model by Ma and Hovy (2016) (referred to as Neural-CRF in Table 2) and report its average performance.

3.2 Results

The experimental results are presented in Table 2. Observe that Neural-CRF+AE, trained either on the training set only or with the addition of the development set, achieves substantial improvements in F-score in both settings, superior to all but one of the benchmark models, highlighting the utility of hand-crafted features incorporated with the proposed auto-encoder loss. Compared against the Neural-CRF, a very strong model in itself, our model significantly improves performance, showing the positive impact of our technique for exploiting manually-engineered features. Although Peters et al. (2018) report a higher F-score using their ELMo embedding technique, our approach here is orthogonal, and accordingly we would expect a performance increase if we were to incorporate their ELMo representations into our model.

Model	F_1
Chieu and Ng (2002)	88.31
Florian et al. (2003)	88.76
Ando and Zhang (2005)	89.31
Collobert et al. (2011)	89.59
Huang et al. (2015)	90.10
Passos et al. (2014)	90.90
Lample et al. (2016)	90.94
Luo et al. (2015)	91.20
Ma and Hovy (2016)	91.21
Yang et al. (2017)	91.62
Peters et al. (2018)	90.15
Peters et al. (2018)+ELMo	**92.22** (± 0.10)
Neural-CRF‡	91.06 (± 0.18)
Neural-CRF+AE‡∗	91.89 (± 0.23)
Ratinov and Roth (2009)†	90.80
Chiu and Nichols (2016)†	91.62
Neural-CRF+AE† ‡	**92.29** (± 0.20)

Table 2: NER Performance on the CoNLL 2003 English NER shared task test set. **Bold** highlights best performance. † marks models trained on both the training and development sets. ‡ indicates average performance over 5 runs. ∗ indicates statistical significance on the test set against Neural-CRF by two-sample Student's t-test at level $\alpha = 0.05$.

Ablation Study To gain a better understanding of the impacts of each feature, we perform an ablation study and present the results in Table 3. We observe performance degradation when eliminating POS, word shape and gazetteer features, showing that each feature contributes to NER performance beyond what is learned through deep learning alone. Interestingly, the contribution of gazetteers is much less than that of the other features, which is likely due to the noise introduced in the matching process, with many incorrectly identified false positives.

Including features based on dependency tags into our model decreases the performance slightly. This might be a result of our simple implementation (as illustrated in Table 1), which does not include dependency direction, nor parent-child relationships.

Next, we investigate the impact of different means of incorporating manually-engineered features into the model. To this end, we experiment with three configurations with features as: (1) input only; (2) output only (equivalent to multi-task learning); and (3) both input and output (Neural-CRF+AE) and present the results in Table 4. Simply using features as either input or output only improves model performance slightly, but insignificantly so. It is only when features are incorporated with the proposed auto-encoder loss do we observe a significant performance boost.

Model	Dev F_1	Test F_1
Neural-CRF+AE	**94.87** (± 0.21)	**91.89** (± 0.23)
− POS tagging∗	94.78 (± 0.17)	91.30 (± 0.28)
− word shape∗	94.83 (± 0.31)	91.36 (± 0.30)
− gazetteer	94.85 (± 0.20)	91.80 (± 0.19)
+ dependencies	94.74 (± 0.16)	91.66 (± 0.18)

Table 3: Ablation study. Average performance over 5 runs with standard deviation. + and − denote adding and removing a particular feature (to/from Neural-CRF+AE trained on the training set only with POS tagging, word shape and gazetteer features). ∗ indicates statistical significance on the test set against Neural-CRF+AE by two-sample Student's t-test at level $\alpha = 0.05$. Note that in this table, ∗ measures the drop in performance.

Model	Dev F_1	Test F_1
Neural-CRF	94.53 (± 0.21)	91.06 (± 0.18)
+ input	94.63 (± 0.23)	91.17 (± 0.25)
+ output	94.69 (± 0.22)	91.23 (± 0.19)
+ input & output∗	**94.87** (± 0.21)	**91.89** (± 0.23)

Table 4: Average performance of Neural-CRF with different features configurations over 5 runs with standard deviation. Note that + input & output = Neural-CRF+AE. ∗ indicates statistical significance on the test set against Neural-CRF by two-sample Student's t-test at level $\alpha = 0.05$.

Training Requirements Neural systems typically require a large amount of annotated data. Here we measure the impact of training with varying amount of annotated data, as shown in Figure 2. Wtih the proposed model architecture, the amount of labelled training data can be drastically reduced: our model, achieves comparable performance against the baseline Neural-CRF, with as little as 60% of the training data. Moreover, as we increase the amount of training text, the performance of Neural-CRF+AE continues to improve.

Hyperparameters Three extra hyperparameters are introduced into our model, controlling the weight of the autoencoder loss relative to the CRF loss, for each feature type. Figure 3 shows the effect of each hyperparameter on test performance. Observe that setting $\lambda_i = 1$ gives strong performance, and that the impact of the gazetteer is less marked than the other two feature types. While increasing λ is mostly beneficial, performance drops if the λs are overly large, that is, the auto-encoder loss overwhelms the main prediction task.

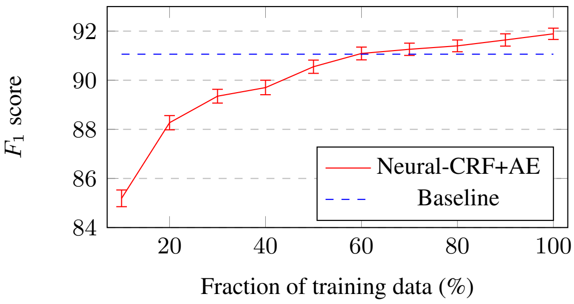

Figure 2: Comparing the Neural-CRF+AE (red solid line) trained with varying amounts of data vs. a Neural-CRF baseline (blue dashed line), trained on the full training set. Performance averaged over 5 runs, and error bars show ± 1 std.dev.

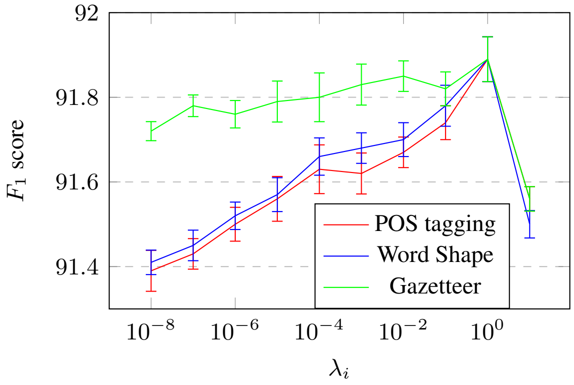

Figure 3: Effect of hyperparameter values on model performance. Each curve shows the effect of λ_i, for feature type i, with all other $\lambda_j = 1$, $j \neq i$. Performance averaged over 5 runs, and error bars show ± 1 variance.

4 Conclusion

In this paper, we set out to investigate the utility of hand-crafted features. To this end, we have presented a hybrid neural architecture to validate this hypothesis extending a Bi-LSTM-CNN-CRF by incorporating an auto-encoder loss to take manual features as input and then reconstruct them. On the task of named entity recognition, we show significant improvements over a collection of competitive baselines, verifying the value of such features. Lastly, the method presented in this work can also be easily applied to other tasks and models, where hand-engineered features provide key insights about the data.

References

Waleed Ammar, Chris Dyer, and Noah A Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In *Proceedings of the 27th International Conference on Neural Informa-*

tion Processing Systems (NIPS 2014), pages 3311–3319.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1936–1945.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: A maximum entropy approach using global information. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 1–7.

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

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8 2014)*, pages 103–111.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734.

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

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2003)*, pages 168–171.

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

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 427–431.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 282–289.

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

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 688–697.

Fei Liu, Trevor Cohn, and Timothy Baldwin. 2018a. Narrative modeling with memory chains and semantic supervision. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 278–284.

Fei Liu, Trevor Cohn, and Timothy Baldwin. 2018b. Recurrent entity networks with delayed memory update for targeted aspect-based sentiment analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2018)*, pages 278–283.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 879–888.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1064–1074.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations (ICLR 2013)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS 2013)*, pages 3111–3119.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML 2010)*, pages 807–814.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, pages 1310–1318.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning (CoNLL 2014)*, pages 78–86.

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

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2018)*, pages 2227–2237.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009)*, pages 147–155.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, pages 1818–1826.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2003)*, pages 142–147.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 698–707.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural reranking for named entity recognition. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP 2017)*, pages 784–792.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS 2015)*, pages 649–657.

Xiao Zhang, Yong Jiang, Hao Peng, Kewei Tu, and Dan Goldwasser. 2017. Semi-supervised structured prediction with neural crf autoencoder. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 1701–1711.

Improved Dependency Parsing using Implicit Word Connections Learned from Unlabeled Data

Wenhui Wang[†] **Baobao Chang**[†] **Mairgup Mansur**[‡]
[†]Key Laboratory of Computational Linguistics, Peking University, MOE, China
[‡]Sogou Technology Inc., Beijing, China
{wenwan,chbb}@pku.edu.cn
{maerhufu}@sogou-inc.com

Abstract

Pre-trained word embeddings and language model have been shown useful in a lot of tasks. However, both of them cannot directly capture word connections in a sentence, which is important for dependency parsing given its goal is to establish dependency relations between words. In this paper, we propose to implicitly capture word connections from unlabeled data by a word ordering model with self-attention mechanism. Experiments show that these implicit word connections do improve our parsing model. Furthermore, by combining with a pre-trained language model, our model gets state-of-the-art performance on the English PTB dataset, achieving 96.35% UAS and 95.25% LAS.

1 Introduction

Dependency parsing is a fundamental task for language processing which aims to establish syntactic relations between words in a sentence. Graph-based models (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010) and transition-based models (Nivre, 2008; Zhang and Nivre, 2011) are the most successful solutions to the challenge.

Recently, neural network methods have been successfully introduced into dependency parsing. Deep feed-forward neural network models (Chen and Manning, 2014; Pei et al., 2015; Weiss et al., 2015) are proposed firstly. It alleviates the heavy burden of feature engineering. LSTM networks (Hochreiter and Schmidhuber, 1997) are then applied to dependency parsing (Dyer et al., 2015; Cross and Huang, 2016; Wang and Chang, 2016; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016) due to its ability to capture contextual information. Generative neural network models (Dyer et al., 2016; Smith et al., 2017; Choe and Charniak, 2016) also show promising parsing performance.

Different from Machine Translation task where massive sets of labeled data could be easily obtained, parsing performance is limited by the relatively small size of available treebank. Vinyals et al. (2015) and Weiss et al. (2015) adopt an approach of tri-training to augment the labeled data. They generate large quantities of parse trees by parsing unlabeled data with two existing parsers and selecting only the sentences for which the two parsers produced the same trees. However, the trees produced this way have noise[1] and tend to be short sentences, since it is easier for different parsers to get consistent results.

Pre-trained neural networks are another methods to take advantage of unlabeled data. Pre-trained word embeddings (Mikolov et al., 2013) and language model (Józefowicz et al., 2016; Peters et al., 2017, 2018) have been shown useful in modelling NLP tasks since word embeddings could capture word semantic information and language model could capture contextual information at the sentence level. However, connections between words in the sentence cannot be directly captured by word embeddings or language model, which are crucial for dependency parsing given its goal is to establish dependency relations between words. In this paper, we propose to implicitly model word connections by a word ordering model. The purpose of word ordering model is to generate a well-formed sentence given a bag of words. We human could make sentences easily from unordered words since we have syntactic knowledge, thus a model generating well-formed sentences from the bag of words encodes syntactic information. In addition, word ordering task allows us to use self-attention mechanism

---

[1] The tri-training approach accuracy on the tune set is 97.26% UAS (Weiss et al., 2015).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2857–2863
Brussels, Belgium, October 31 - November 4, 2018.

to model connections between words in the sentence. Different from the tri-training approach, our approach takes advantage of implicit word connections learned by self-attended word ordering model in an unsupervised way.

Experiments show that pre-trained word ordering model significantly improves our dependency parsing model. Ablation tests also show self-attention mechanism is critical. Moreover, by combining word ordering model and language model, our graph-based dependency parsing model achieves SOTA performance on the English Penn Treebank (Marcus et al., 1993) with 96.35% UAS and 95.25% LAS.

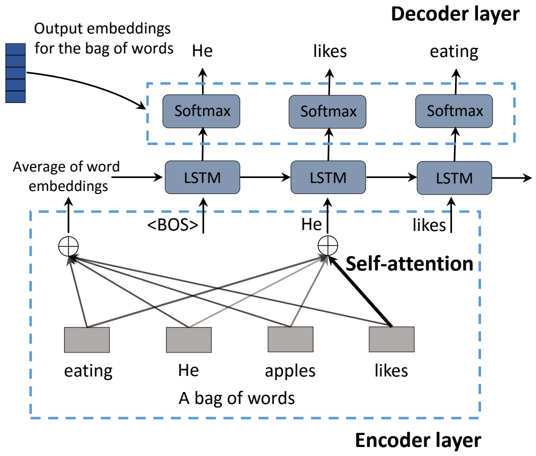

Figure 1: Overview of our word ordering model.

2 Neural Word Ordering Model

The target of word ordering is to generate a well-formed sentence given a bag of words. To capture word connections implicated in the sentence, an LSTM-based word ordering model with self-attention is proposed. Self-attention mechanism effectively decides which words in the word bag are more important in generating the next word. It improves the ability of our model to capture word connections. As illustrated in Figure 1, the proposed word ordering model consists of two layers:

Encoder Layer

Given a bag of words $w_1, w_2, ..., w_n$, we encode each word by a character-level BiLSTM ($c_{w_{1:n}}^{wo}$), which could reduce the parameters used in our model compared with word embeddings. For the input word of current time-step (w_i), a self-attention layer is utilized to align the word with its related words, producing its self-attended vector ($sa_{w_i}^{wo}$) as following:

$$s_j^i = \mathrm{v}^{\mathrm{T}}\mathrm{ReLU}(W^{sa}[c_{w_i}^{wo}; c_{w_j}^{wo}]) \quad (1)$$

$$a_t^i = \exp(s_t^i)/\Sigma_{j=1}^{n}\exp(s_j^i) \quad (2)$$

$$sa_{w_i}^{wo} = \Sigma_{j=1}^{n} a_j^i c_{w_j}^{wo} \quad (3)$$

The scores $s_{1:n}^i$ in self-attention explicitly represent the connections between words.

We then concatenate the character-level word embedding ($c_{w_i}^{wo}$) and its self-attended vector ($sa_{w_i}^{wo}$):

$$x_i^{wo} = [c_{w_i}^{wo}; sa_{w_i}^{wo}] \quad (4)$$

x_i^{wo} is fed into the decoder layer to generate the next word.

Decoder Layer

Given the current input vector (x_i^{wo}), which contains current word information and weighted information of related words, a forward LSTM is used to generate the next word. We initialize the forward LSTM with an average of the input word embeddings ($c_{w_{i:n}}^{wo}$). A virtual token $\langle$BOS$\rangle$ is added as the input of the first LSTM time-step:

$$\overrightarrow{h}_i^{wo} = \mathrm{LSTM}(x_i^{wo}, \overrightarrow{h}_{i-1}^{wo}) \quad (5)$$

At each time-step, the hidden state $\overrightarrow{h}_i^{wo}$ is utilized to predict the next word. Due to the output vocabulary is limited in the bag of words, we just compute scores for the given words ($w_{1:n}$):

$$so_j^i = \mathrm{v}^{\mathrm{T}}\mathrm{ReLU}(W^o[\overrightarrow{h}_i^{wo}; cd_{w_j}^{wo}]) + {\overrightarrow{h}_i^{wo}}^{\mathrm{T}} M wd_{w_j}^{wo} \quad (6)$$

To reduce the parameters, each output word is represented by a character-level BiLSTM embedding ($cd_{w_j}^{wo}$) and a low-dimensional word embedding[2] ($wd_{w_j}^{wo}$). M is a matrix projecting a low-dimensional embedding back up to the dimensionality of LSTM hidden states. The scores $so_{1:n}^i$ are then normalized with Softmax, and the word with max probability is chosen as the next token.

The word ordering model could be trained easily in an unsupervised manner. Given a large set of unlabeled sentences, we can just ignore the word

---

[2] Character-level representations lack the capacity to differentiate between words that have very different meanings but that are spelled similarly. Low-dimensional word embeddings are added to improve the ability.

order of sentence and train the model to generate the corresponding well-formed sentence in the training set. To be specific, we minimize the sum of negative log probabilities of the ground truth words on the unlabeled data set. Different from language model, the choice for each decoder step is limited in the bag of words. Moreover, self-attention can be introduced into the word ordering model since we have known the bag of words, which could capture the dependency connections between words. We also pre-train a backward word ordering model to generate sentences in reverse order. The forward and backward models share character-level BiLSTM embeddings, self-attention layer, and Softmax layer.

Different from previous word ordering models (Liu et al., 2015; Schmaltz et al., 2016), self-attention mechanism is introduced into our model to capture word connections. Moreover, our more important goal is to implicitly utilize large-scale unlabeled data to help dependency parsing.

3 Neural Graph-based Parsing Model

We implement an LSTM-based neural network model as our graph-based dependency parsing baseline, which is similar to (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016). As shown in the Figure 2, it consists of three layers:

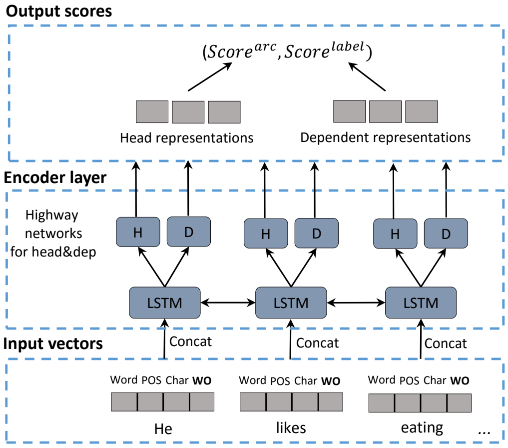

Figure 2: Overview of neural graph-based dependency parsing model. WO represents pre-trained vectors from word ordering model.

Input Layer

Given a n-words input sentence s with words $w_1, w_2, ..., w_n$ and its POS tags $p_1, p_2, ..., p_n$. The input layer creates a sequence of input vectors $x_{1:n}$ in which each x_i is a concatenation of its word embedding (e_{w_i}), POS tag embedding (e_{p_i}), character-level BiLSTM embedding (c_{w_i}), and word ordering model pre-trained vector (wo_{w_i}):

$$x_i = [e_{w_i}; e_{p_i}; c_{w_i}; wo_{w_i}] \tag{7}$$

To get the word ordering model pre-trained vector (wo_{w_i}), the sentence s is fed into the pre-trained word ordering model. Following Peters et al. (2018), we then combine the input vector ($x_i^{wo} = [c_{w_i}^{wo}; sa_{w_i}^{wo}]$) and L-layer BiLSTM vectors ($h_{i,j}^{wo} = [\overrightarrow{h}_{i,j}^{wo}; \overleftarrow{h}_{i,j}^{wo}] \mid j=1,2,...,L$) by a Softmax-normalized weight (W^{woc}) and a scalar parameter (γ):

$$wo_{w_i} = \gamma(W_0^{woc} x_i^{wo} + \Sigma_{j=1}^{L} W_j^{woc} h_{i,j}^{wo}) \tag{8}$$

The parameters of word ordering model are fixed during the training of parsing model. However, the weight and scalar parameters are tuned to better adapt to it. The combined output wo_{w_i} contains word connections from self-attention, word information from character-level embedding and contextual information from LSTM.

Encoder & Output Layer

To introduce more contextual information, we encode each input element by deep BiLSTMs:

$$v_i = \textbf{BiLSTM}(x_{1:n}, i) \tag{9}$$

Two different highway networks (Srivastava et al., 2015) are then used to encode head word representations ($v_{1:n}^{head}$) and dependent word representations ($v_{1:n}^{dep}$). For a head-dependent dependency pair (w_h, w_d), the dependency arc and label score are computed by two MLP networks:

$$i_{h,d} = [v_h^{head}; v_d^{dep}; v_h^{head} \odot v_d^{dep}] \tag{10}$$

$$s_{h,d}^{arc} = W_1^{arc}\mathrm{ReLU}(W_2^{arc} i_{h,d}) \tag{11}$$

$$s_{h,d}^{label} = W_1^{label}\mathrm{ReLU}(W_2^{label} i_{h,d}) \tag{12}$$

We use the Max-Margin criterion to train our parsing model, which is the same as (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016).

4 Experiments

4.1 Datasets

We conduct experiments on the English Penn Treebank and the CoNLL 09 English dataset. For

Method	PTB UAS / LAS	CONLL 09 UAS / LAS
Bohnet2012	- / -	92.87 / 90.60
Weiss2015	94.26 / 92.41	- / -
Alberti2015	94.23 / 92.36	92.7 / 90.56
Kiperwasser2016	93.9 / 91.9	- / -
Wang2016	94.08 / 91.82	- / -
Andor2016	94.61 / 92.79	93.22 / 91.23
Dozat2016	95.74 / 94.08	**95.21 / 93.20**
Smith2017	95.8 / 94.6	- / -
Choe2016	95.9 / 94.1	- / -
Baseline	95.12 / 93.98	93.61 / 91.70
Baseline+WO	95.66 / 94.54	94.21 / 92.27
Baseline+LM	96.05 / 94.88	94.50 / 92.70
Baseline+LM&WO	**96.35 / 95.25**	94.83 / 93.05

Table 1: Results on the English PTB dataset and CoNLL 09 English dataset. WO represents the pre-trained word ordering model. LM represents the pre-trained language model from Peters et al. (2018).

PTB dataset, we follow the standard splits. Using section 2-21 for training, section 22 as development set and 23 as test set. The treebank is converted to Stanford Basic Dependencies (Marneffe et al., 2006) by version 3.3.0[3] of the Stanford parser. The Stanford POS Tagger (Toutanova et al., 2003) is used for assigning POS tags. Following previous work, UAS (unlabeled attachment scores) and LAS (labeled attachment scores) are calculated by excluding punctuation. For the CoNLL 09 English dataset, we follow the standard practice and include all punctuation in the evaluation. We pre-train our word ordering model on the 1 billion word benchmark (Chelba et al., 2014).

4.2 Implementation Details

The graph-based dependency parsing model and word ordering model are optimized with Adam with an initial learning rate of $2e^{-3}$. The β_1 and β_2 used in Adam are 0.9 and 0.999 respectively.

The following hyper-parameters are used in all graph-based dependency parsing models: word embedding size = 300, POS tag embedding size = 32, character embedding size = 50, word-level LSTM hidden vector size = 200, word-level BiLSTM layer number = 3, character-level LSTM hidden vector size = 50, character-level BiLSTM layer number = 2, batch size = 32. We also apply dropout for the input and each layer with dropout rate of 0.3. We use pre-trained case-sensitive GloVe embeddings[4] to initialize word embeddings. These word embeddings are fine tuned with the graph-based dependency parsing model. The parameters of pre-trained word ordering model are fixed during the training of dependency parsing model. For deep BiLSTM, we concatenate the outputs of each layer as its final outputs.

For our word ordering model: input character-level LSTM hidden vector size = 512, input character-level BiLSTM layer number = 1, word-level LSTM hidden vector size = 1024, word-level LSTM layer number = 2, output character-level LSTM hidden vector size = 512, output character-level BiLSTM layer number = 1, output low-dimensional word embedding size = 64, batch size = 32, dropout for the input and each layer = 0.5.

4.3 Main Results & Ablation Study

Table 1 shows the performance of our model and previous work on two English benchmarks. Our model achieves promising results on both datasets. Two sets of experiments are provided to show the effectiveness of pre-trained word ordering model. Although our baseline system is similar to (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016) but with subtle differences in architecture, the baseline could perform much better to our surprise and thus constitutes a very strong baseline. Compared with this baseline, introducing the pre-trained word ordering model achieves a significant improvement (almost 0.6% UAS gains for both datasets, $p < 0.001$). To further show the effectiveness of word ordering model, we also implement an even stronger baseline with pre-trained language model[5]. Compared with this much stronger baseline, incorporating pre-trained word ordering model still achieves a significant improvement (0.3% UAS gains for both datasets, $p < 0.01$). We attribute the improvement to the ability of word ordering model to capture word connections, which cannot be directly captured by language model. Moreover, by combining with a pre-trained language model, our model outperforms current SOTA model from 95.9% UAS to 96.35% UAS on the PTB dataset. The introduction

---

[3] `http://nlp.stanford.edu/software/lex-parser.shtml`

[4] Downloaded from `http://nlp.stanford.edu/data/glove.840B.300d.zip`.

[5] We use the pre-trained language model provided by Peters et al. (2018), which can be downloaded at `http://allennlp.org/elmo`. The pre-trained language model vectors are added in the input layer, which are included in the same way as word ordering model. Peters et al. (2018) found the pre-trained language model works extremely well in six NLP tasks including QA, SRL, and others, we confirm its effectiveness in parsing task.

Model	UAS / LAS
Baseline+WO	**95.66 / 94.54**
-Self-attention vectors	95.38 / 94.27
Baseline+LM&WO	**96.35 / 95.25**
-Self-attention vectors	96.13 / 94.98

Table 2: Ablation tests of self-attention mechanism on the PTB dataset.

of POS tag features could contribute about 0.2% improvement in our experiments. The word ordering model could be more helpful without POS tag features and seem to compensate for the lack of POS tag features.

To show the importance of self-attention mechanism, we do ablation tests on the models with pre-trained word ordering model vectors. We remove self-attention vectors by replacing it with the character-level representations. As shown in table 2, self-attention further improves dependency parsing. Word connections modeled by self-attention are important for dependency parsing.

Figure 3 shows an example of word connections learned by the model, where we use the solid line to indicate the word connections learned by the word-ordering model and dashed line to the expected dependencies. We can see meaningful overlap could be observed in the example. The percentage of overlap between connections and dependency arcs is over 40% for the sentences less than 10 words. The differences between connections and dependency arcs are because that our word ordering model trained without any supervised dependency information. The connections are actually built to increase the likelihood.

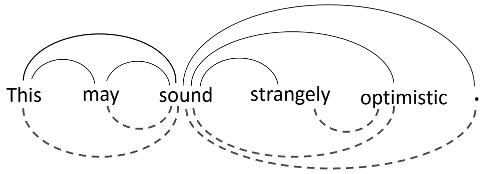

Figure 3: An example of self-attention. The solid line denotes the mostly attended word when generating the next word, dashed line denotes the correct dependency.

5 Conclusion

In this paper, we propose to implicitly capture word connections from large-scale unlabeled data by a word ordering model with self-attention. Experiments show these features are helpful for dependency parsing. Moreover, with the help of word ordering model and language model, our model achieves SOTA results on the PTB dataset. As for future work, we are testing on languages other than English.

Acknowledgement

We thank all the anonymous reviewers for their helpful comments. This work is supported by National Natural Science Foundation of China under Grant No.61876004 and No.61751201. The corresponding author of this paper is Baobao Chang.

References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1354–1359.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1455–1465.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *EMNLP-CoNLL*, pages 957–961.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural*

Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, pages 2331–2336.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 334–343.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 199–209.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics.

Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 113–122.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Marie Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. *Lrec*, pages 449–454.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98. Association for Computational Linguistics.

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*. Citeseer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 313–322.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1756–1765.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Allen Schmaltz, Alexander M. Rush, and Stuart M. Shieber. 2016. Word ordering without syntax. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2319–2324.

Noah A. Smith, Chris Dyer, Miguel Ballesteros, Graham Neubig, Lingpeng Kong, and Adhiguna Kuncoro. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1249–1258.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2773–2781.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193.

A Framework for Understanding the Role of Morphology in Universal Dependency Parsing

Mathieu Dehouck
Univ. Lille, CNRS, UMR 9189 - CRIStAL
Magnet Team, Inria Lille
59650 Villeneuve d'Ascq, France
mathieu.dehouck@inria.fr

Pascal Denis
Magnet Team, Inria Lille
59650 Villeneuve d'Ascq, France
pascal.denis@inria.fr

Abstract

This paper presents a simple framework for characterizing morphological complexity and how it encodes syntactic information. In particular, we propose a new measure of morpho-syntactic complexity in terms of governor-dependent preferential attachment that explains parsing performance. Through experiments on dependency parsing with data from Universal Dependencies (UD), we show that representations derived from morphological attributes deliver important parsing performance improvements over standard word form embeddings when trained on the same datasets. We also show that the new morpho-syntactic complexity measure is predictive of the gains provided by using morphological attributes over plain forms on parsing scores, making it a tool to distinguish languages using morphology as a syntactic marker from others.

1 Introduction

While word embedding has proven a good solution to reduce data sparsity in parsing (Koo et al., 2008), treating word forms as atomic units is at odds with the fact that words have a potentially complex internal structure. Furthermore, it makes parameters estimation difficult for morphologically rich languages (MRL) in which the number of possible forms a word can take can be very large[1].

Recently, researchers have started to work on morphologically informed word embeddings (Cao and Rei, 2016; Botha and Blunsom, 2014), aiming at better capturing both lexical, syntactic and morphological information. But encoding lexicon and morphology in the same space makes it difficult to distinguish the role of each in syntactic tasks such as dependency parsing. Furthermore, morphologically rich languages for which we hope to see a real impact from those morphologically aware representations, might not all rely to the same extent on morphology for syntax encoding. Some might benefit mostly from reducing data sparsity while others, for which paradigm richness correlate with freer word order (Comrie, 1981), will also benefit from morphological information encoding.

This paper aims at characterizing the role of morphology as a syntax encoding device for various languages. Using simple word representations, we measure the impact of morphological information on dependency parsing and relate it to two measures of language morphological complexity: the basic form per lemma ratio and a new measure (*HPE*) defined in terms of head attachment preference encoded by its morphological attributes. We show that this new measure is predictive of parsing result differences observed when using different word representations and that it allows one to distinguish amongst morphologically rich languages, those that use morphology for syntactic purpose from those using morphology as a more semantic marker. To the best of our knowledge, this work is the first attempt at systematically measuring the syntactic content of morphology in a multi-lingual environment.

Section 2 presents the representation learning method and the dependency parsing model. It also defines two measures of morphological complexity. Section 3 describes the experimental setting and analyses parsing results in terms of the previously defined morphological complexity measures. Section 4 gives some conclusions and future work perspectives.

---

[1] A typical English noun has 2 forms while a Finnish one may have more than 30. This shows in data as English lemmas have 1.39 forms on average while Finnish ones have 2.19, as measured on UD data (Nivre et al., 2016).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2864–2870
Brussels, Belgium, October 31 - November 4, 2018.

2 Framework

This section details: (i) our method for learning lexical and morphological representations, (ii) how these can be used for graph-based dependency parsing, and (iii) how to measure morphological complexity. Our representation learning and parsing techniques are purposely very simple in order to let us separate lexical and morphological information and weight the role of morphology in dependency parsing of MRL.

2.1 Word Representation

We construct separate vectorial representations for lemmas, forms and morphological attributes, either learned via dimension reduction of their own cooccurrence count matrices or represented as raw one-hot vectors.

Let $\mathcal{V}$ be a vocabulary (it can be lemmas or forms or morphological attributes (incl. values for POS, number, case, tense, mood...)) for a given language. Correspondingly, let $\mathcal{C}$ be the set of contexts defined over elements of $\mathcal{V}$. That is, lemmas appear in the context of other lemmas, forms in the context of forms, and attributes in the context of attributes. Then, given a corpus annotated with lemmas and morphological information, we can gather the cooccurrence counts in the matrix $\boldsymbol{M} \in \mathbb{N}^{|\mathcal{V}| \times |\mathcal{C}|}$, such that $\boldsymbol{M}_{ij}$ is the frequency of lemma (form or morphological attributes) $\mathcal{V}_i$ appearing in context $\mathcal{C}_j$ in the corpus. Here, we consider plain sequential contexts (i.e. surrounding bag of "words") of length 1, although we could extend them to more structured contexts (Bansal et al., 2014). Those cooccurrence matrices are then reweighted by unshifted Positive Point-wise Mutual Information (PPMI) and reduced via Singular Value Decomposition (SVD). For more information on word embedding via matrix factorization, please refer to (Levy et al., 2015).

Despite its apparent simplicity, this model is as expressive as more popular state of the art embedding techniques. Indeed, Goldberg and Levy (2014) have shown that the SkipGram objective with negative sampling of Mikolov's Word2vec (2013) can be framed as the factorization of a shifted PMI weighted cooccurrence matrix.

This matrix reduction procedure gives us vectors for lemmas, forms and morphological attributes, noted $\boldsymbol{R}$. Note that while a word has only one lemma and one form, it will often realize several morphological attributes. We tackle this issue by simply summing over all the attributes of a word (noted $Morph(w)$). If we note $\boldsymbol{r}_w$ the vectorial representation of word w we have:

$$\boldsymbol{r}_w = \sum_{a \in Morph(w)} \boldsymbol{R}_a.$$

Simple additive models have been shown to be very efficient for compositionally derived embeddings (Arora et al., 2017).

2.2 Dependency Parsing

We work with graph-based dependency parsing, which offers very competitive parsing models as recently re-emphasized by Dozat et al. (2017) in the CONLL 2017 shared-task on dependency parsing (Zeman et al., 2017).

Let $x = (w_1, w_2, ..., w_n)$ be a sentence, $\mathcal{T}_x$ be the set of all possible trees over it, $\hat{y}$ the tree that we predict for x, and $Score(\bullet, \bullet)$ a scoring function over sentence-tree pairs :

$$\hat{y} = \underset{t \in \mathcal{T}_x}{\text{argmax}}\ Score(x, t).$$

We use edge factorization to make the inference problem tractable. A tree score is thus the sum of its edges scores. We use a simple linear model:

$$Score(x, t) = \sum_{e \in t} \boldsymbol{\theta}^\top \cdot \boldsymbol{\phi}(x, e),$$

where $\boldsymbol{\phi}(x, e)$ is a feature vector representing edge e in sentence x, and $\boldsymbol{\theta} \in \mathbb{R}^m$ is a parameter vector to be learned.

The vector representation of an edge e_{ij} whose governor is the i-th word w_i and dependent is the j-th word w_j, is defined by the outer product of their respective representations in context. Let $\oplus$ note vector concatenation, $\otimes$ the outer product and $w_{k\pm1}$ be the word just before/after w_k, then: $\boldsymbol{v}_i = \boldsymbol{w}_{i-1} \oplus \boldsymbol{w}_i \oplus \boldsymbol{w}_{i+1}$, $\boldsymbol{v}_j = \boldsymbol{w}_{j-1} \oplus \boldsymbol{w}_j \oplus \boldsymbol{w}_{j+1}$ and

$$\boldsymbol{\phi}(x, e_{ij}) = \text{vec}(\boldsymbol{v}_i \otimes \boldsymbol{v}_j) \ \in \mathbb{R}^{9d^2}.$$

Recall that $\boldsymbol{w}_i$ of length $d \ll \mathcal{V}$ is a vector from $\boldsymbol{R}$.

We use the averaged Passive-Aggressive online algorithm for structured prediction (Crammer et al., 2006) for learning the model $\boldsymbol{\theta}$. Given a score for each edge, we use Eisner algorithm (Eisner, 1996) to retrieve the best projective spanning tree. Even though some languages display a fair amount of non-projective edges, on average Eisner algorithm scores higher than Chu-Liu-Edmonds algorithm (Chu and Liu, 1965) in our setting.

2.3 Measuring Morpho-Syntactic Complexity

Some languages use morphological cues to encode syntactic information while other encode more semantic information with them. For example, the Case feature (especially core cases) is of prime syntactic importance, for it encodes the type of relation words have with each other. On the contrary, the Possessor feature (in Hungarian for example) is more semantic in nature and need not impact sentence structure. This remark would support different treatment for each language. However, those languages tend to be treated equally in works dealing with MRL.

Form to Lemma Ratio A basic measure of morphological complexity is the form per lemma ratio, we note it F/L. It captures the tendency of words to inflect in a given language. Because some word classes tend not to inflect and not all forms are equally productive, we note F/iL the ratio of form per inflected lemma. Given a language l with a lemma vocabulary $\mathcal{V}^l$ and a form counting function $c : \mathcal{V}^l \to \mathbb{N}$ that returns the number of forms a lemma can take, we have:

$$F/L(l) = \frac{1}{|\mathcal{V}^l|} \sum_{w \in \mathcal{V}^l} c(w),$$

$$F/iL(l) = \frac{1}{|\mathcal{V}^l_i|} \sum_{w \in \mathcal{V}^l_i} c(w), \mathcal{V}^l_i = \{w \in \mathcal{V}^l | c(w) > 1\}$$

F/L and F/iL do not measure the informative content of morphology, but simply its productivity. Bentz et al. (2016) compared five different measures of morphological complexity amongst which word entropy and the micro-averaged version of F/L (they call it TTR) and showed that they all have high positive correlation given enough data.

Head POS Entropy In order to compare the morpho-syntactic complexity of different languages, we introduce a new measure called *Head Part-of-speech Entropy* or *HPE*. The *HPE* of a token t represents the amount of information t has about the part-of-speech of its governor. More formally, let $POS(Gov(t))$ be the set of parts-of-speech that t can depend on, and let $\pi_t(p)$ be the probability of t actually depending on part-of-speech p, then the *HPE* is defined as:

$$HPE(t) = \sum_{p \in POS(Gov(t))} -\pi_t(p) log_2(\pi_t(p)).$$

This is a measure of a token preferencial attachment to its head. A token with a low *HPE* tends to attach often to the same part-of-speech, while a token with a high *HPE* will attach to many different parts-of-speech. Thus a language with a low *HPE* will tend to encode a lot of syntactic information in the morphology, rather than in word order say.

For example, a noun can attach to another noun like a genitive, or to a verb as a subject or object, or even to an adjective in the case of transitive adjective. French nouns do not inflect for case, thus attachment to another noun or verb can only be infered from words relative positions. On the contrary, Gothic nouns do inflect for case, thus making verb or noun attachment clear directly from the morphological analysis.

We compute the *HPE* of a language as the averaged *HPE* of its attributes sets over a given corpus. Likewise, we use the empirical counts as a surrogate for c in F/L and F/iL.

3 Experiments

In order to test the hypothesis that morphological representations contain syntactic information crucial for dependency parsing of morphologically rich languages, but that this information is not equally distributed across MRL, we run experiments on data from the Universal Dependencies (Nivre et al., 2016) project.

Data Description For conciseness, we focused on eleven languages that display varying degrees of morphological complexity and belong to four different language families. Basque (eu) is an isolate and it is an ergative language. English (en), Gothic (got), Danish (da) and Swedish (sv) are Germanic languages, and French (fr) and Romanian (ro) are Romance languages (Indo-European). Finnish (fi), Estonian (et) and Hungarian (hu) are Finno-Ugric languages. Hebrew (he) is a Semitic language. Basic statistics are provided in Table 1.

Experimental Settings For the experiments we use the train/dev/test data provided by UD 2.0. Basic statistics about the data are reported in the appendix. Lemmas and forms are embedded in 150 dimensions, while Morphological attributes are embedded in 50 dimensions, because they are much less numerous (less than 100). All embeddings are induced on their language respective train set only using a context window of size 1 (i.e. the

	da	en	et	eu	fi	fr	got	he	hu	ro	sv
Train	4383	12543	2263	5396	12217	14553	3387	5241	910	8043	4303
POS	17	17	16	16	15	17	14	16	16	17	16
Feats	44	35	58	69	88	36	40	48	73	59	39
F/L	1.44	1.39	1.60	2.32	2.19	1.38	2.44	1.83	1.46	2.03	1.59
F/iL	2.80	2.76	3.35	4.29	4.68	3.15	4.20	3.39	3.03	3.76	2.91
HPE	1.07	1.12	0.55	0.51	0.57	0.87	0.60	1.01	0.60	0.71	0.84

Table 1: Basic datasets statistics. The first line gives the number of train sentences for each language. The second and third give the number of part-of-speech and of morphological attribute values for each language. The fourth and fifth lines reports forms per lemma ratios. Last line gives the HPE.

		da	en	et	eu	fi	fr	got	he	hu	ro	sv
Lem	OH	58.47	67.05	44.96	60.27	56.06	70.93	60.63	65.70	47.32	68.69	61.30
	Emb	68.33	75.21	59.23	69.59	66.90	71.90	71.14	72.52	51.04	72.83	73.27
Form	OH	56.92	65.83	41.36	57.67	51.50	70.35	58.68	67.08	44.35	67.2	58.54
	Emb	70.64	75.13	57.36	65.64	60.38	76.05	68.72	72.68	55.06	73.21	72.72
Morph	OH	**73.76**	**76.27**	**71.21**	**73.81**	**75.58**	78.67	**76.88**	**77.65**	**69.67**	**76.57**	**76.42**
	Emb	73.27	76.17	70.20	73.21	73.33	**78.79**	76.37	76.95	69.38	76.32	76.02

Table 2: UAS scores for parsers using lemmas (Lem), word forms (Form) or morphological attributes (Morph) representations as features. For each type, we report results using one-hot representation (OH) and results using embeddings (Emb).

directly preceding and following words).

Parsers are trained for 10 iterations using either lemma, form or morphological representations, and we pick the best iteration on the basis of UAS on the development set.

While we used gold lemmas as provided in the corpora, we ran two experiments for morphological attributes, one with gold attributes and one with predicted attributes. Morphological attributes are predicted with a simple multinomial logistic regression per attribute (POS, Tense, Case, Gender...), where we add a special *undef* value (except for POS) to represent the lack of an attribute (e.g., nouns have no Tense in English). The models predict attribute values for the center word of trigrams represented by feature vectors encoding word prefixes and suffixes of length 1, 2 and 3, word length and capitalization. We used the logistic regression implemented in the Scikit-Learn (Pedregosa et al., 2011) library with the default settings. It can output an argmaxed decision or a softmaxed decision, thus we tried both as input to the parser. The argmaxed decision gives a vector of zeros and ones, while the softmaxed decision gives a continuous vector with each each attributes summing to one (the probability assigned to each possible value for Gender like Masculine, Feminine, Neuter and Undef must sum to one). Then those vectors are used unchanged for the one-hot representation or passed through an embedding matrix for the embedding representation.

Results For clarity, we focus on comparing results using form embeddings and gold morphological representations. They are given in Table 2. Because the analysis carries to the labeled case, we stick to unlabeled scores (UAS) for the analysis. A more complete table is provided in the appendix as well as a complete labeled accuracy score (LAS) table. Morphological complexity measures are also reported.

One-hot gold morphological attributes consistently outerperform form embeddings. This is expected since forms embedding were trained on much fewer data than usually considered necessary. However, improvements are not consistent across languages, ranging from 1.14 point for English to 15.20 points for Finnish. While those differences are not explained by morphological productivity alone (Figure 1a), a measure of preferential attachment gives a good account of them (Figure 1b). Those inconsistencies become even more striking, considering results using predicted attributes. We notice that despite a general drop of performance of 5-12 points, predicted attributes

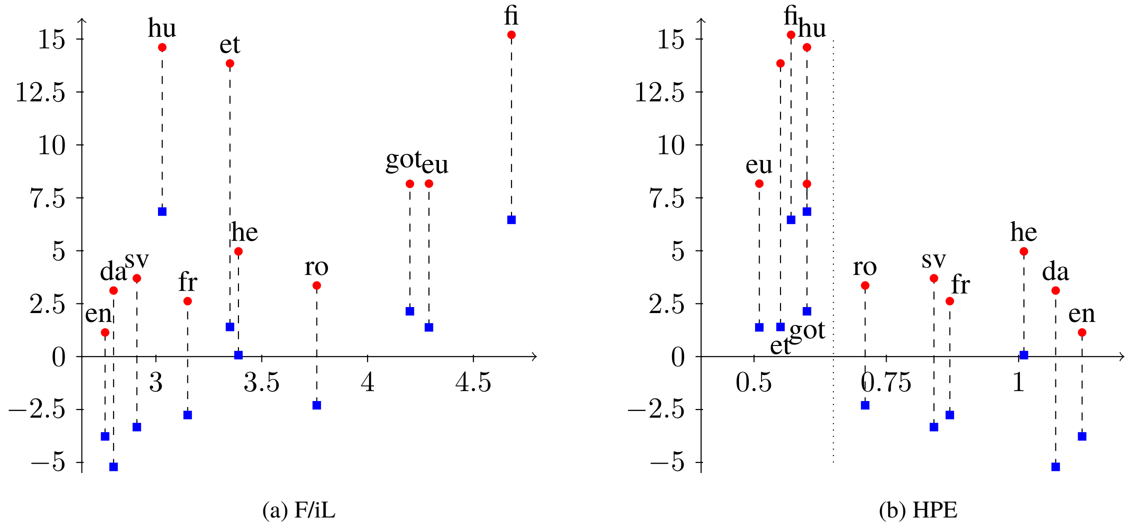

Figure 1: Accuracy differences (y-axis) between parsers using form embeddings and parsers using one-hot attributes, with respect to morphological complexity (x-axis). Red dots represent the gold attributes scores and blue squares the predicted attributes scores.

still perform significantly better than form embeddings for those morphologically rich languages that have an *HPE* lower than 0.65 as depicted on Figure 1b.

Figures 1a and 1b plot the differences in parsing scores. For each language, the red dot corresponds to the score difference between using form embeddings and gold attributes one-hot representations, and the blue square corresponds to the score difference between using the same form embeddings and predicted attributes softmax representations (the complete scores are given in the appendix). Figure 1a plots those differences with regard to the form per inflected lemma ratio (F/iL) and Figure 1b plots those differences with regard to the head POS entropy (*HPE*).

Both Figures show trends. Score differences seem to increase with F/iL and decrease with *HPE*. But while the F/iL plot suffers outliers (Hungarian, Estonian and Romanian), the *HPE* plot shows a clear boundary between languages benefiting fully from morphological information (even predicted) and those benefiting primarily from reducing data sparsity. While Hebrew seems to be an outlier, it might be due to its annotation style, where attached prepositions, articles and possessive markers are treated as independent words rather than morphological inflection as other languages do, thus artificially increasing the parsing accuracy with a lot of trivial dependencies.

This shows that indeed, *HPE* is a good measure of the syntactic informativeness of a language morphology, and that it can help deciding between encoding morphological information or just reducing data sparsity. Furthermore, it seems to be link to the distinction that Kibort and Corbett (2010) do between morphosyntax and morphosemantic.

4 Conclusion

We have contributed a new measure of morpho-syntactic complexity (*HPE*) that helps distinguishing languages that use morphology for syntactic purpose from languages that use morphology to encode more semantic information. We showed that this measure correlates much more with differences in parsing results using morphological representations than the simple form per lemma ratio. It could thus be used to help designing language specific word representations.

It is worth mentioning that we focused here on dependent marked head selection. It would be interesting to have a similar measure for head-marking situations with dependencies marked on the governor. We leave it for future work.

5 Acknowledgement

This work was supported by ANR Grant GRASP No. ANR-16-CE33-0011-01 and Grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020. We also thank the reviewers for their valuable feedback.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *International Conference on Learning Representations 2017*.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815. Association for Computational Linguistics.

Christian Bentz, Tatyana Ruzsics, Alexander Koplenig, and Tanja Samardzic. 2016. A comparison between morphological complexity measures: typological data vs. language corpora. In *Proceedings of the workshop on computational linguistics for linguistic complexity (cl4lc)*, pages 142–153.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. *CoRR*, abs/1405.4273.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. *CoRR*, abs/1606.02601.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14.

Bernard Comrie. 1981. *Language universals and linguistic typology : syntax and morphology / Bernard Comrie*. Blackwell, Oxford :.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

A Kibort and GG Corbett. 2010. *Features: perspective on a key notion in linguistics.* Oxford University Press, Oxford.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

		da	en	et	eu	fi	fr	got	he	hu	ro	sv
	POS	87.37	87.24	84.49	86.09	86.14	90.60	90.44	90.93	88.43	90.64	89.23
	Attributes	98.02	97.74	96.20	97.56	97.09	97.60	96.00	97.84	92.76	97.76	96.88
	Form Emb	**70.64**	**75.13**	57.36	65.64	60.38	**76.05**	68.72	72.68	55.06	**73.21**	**72.72**
Morph	Hard OH	64.69	69.32	57.16	64.51	64.33	72.82	69.94	71.60	61.80	71.36	67.99
	Soft OH	65.43	71.36	**58.76**	**67.02**	**66.84**	73.29	**70.86**	**72.75**	**61.91**	72.03	69.48
	Hard Emb	64.19	69.51	55.53	64.10	62.58	72.28	69.29	71.51	59.62	70.86	67.82
	Soft Emb	65.33	70.75	57.24	66.18	65.04	73.18	70.46	71.85	60.64	71.34	68.60

Table 3: UAS scores for parsers using predicted morphological attributes. The two first rows are POS and averaged attributes prediction accuracy. The third row reports UAS using form representations for comparison purpose. Rows 4 to 7 give UAS using morphological representations, either one-hot or embedding. Regressors output a probability distribution per morphological feature, we either use those soft decision as input for the parser (Soft) or apply argmax first (Hard).

		da	en	et	eu	fi	fr	got	he	hu	ro	sv
Lem	OH	48.09	57.09	25.30	45.96	40.78	64.88	46.85	54.91	27.80	56.89	48.61
	Emb	62.47	70.95	48.17	62.52	59.34	65.62	61.37	64.41	41.59	64.76	65.70
Form	OH	45.12	54.97	21.29	40.53	34.59	61.95	45.19	55.82	25.60	53.83	45.00
	Emb	65.09	71.20	45.79	57.42	52.67	70.81	59.35	66.92	44.30	65.13	64.93
Morph	OH	**69.19**	**72.32**	**64.06**	**68.19**	**71.00**	73.92	**71.04**	**72.66**	**64.31**	**68.94**	**69.97**
	Emb	68.71	72.22	62.81	67.30	68.70	**73.96**	70.41	71.77	63.45	68.76	69.69

Table 4: LAS scores for parsers using lemmas (Lem), forms (Form) or morphosyntactic attributes (Morph) representations as features. Representations are either embeddings or one-hot.

		da	en	et	eu	fi	fr	got	he	hu	ro	sv
	Form Emb	**65.09**	**71.20**	45.79	57.42	52.67	**70.81**	59.35	**66.92**	44.30	**64.13**	**65.93**
Morph	hard OH	58.33	62.64	43.80	55.81	54.42	66.66	59.73	63.74	52.41	62.10	60.29
	soft OH	59.68	65.59	**47.05**	**59.43**	**58.74**	67.39	**62.36**	66.25	**53.63**	63.26	62.44
	hard Emb	57.72	62.73	42.22	55.06	52.79	66.25	59.14	63.57	49.99	61.67	60.03
	soft Emb	59.13	64.97	45.64	58.25	56.51	67.00	62.02	65.33	52.61	62.65	61.47

Table 5: LAS scores for parsers using predicted morpho-syntactic attributes. First row is LAS using form representation. Rows 2 to 5 are LAS using morphological representation, either one-hot or embedding and either hard decisions or soft decisions.

Appendix A: Supplementary Tables

Table 3 reports results for the predicted attributes experiment. The POS and averaged attributes prediction accuracies are given. Are also reported, scores for the four representation regimes of predicted attributes. Predictions can be either probability distributions (Soft) or argmax (Hard) and either used as such (OH) or passed through an embedding (Emb).

Table 4 reports all the labeled accuracy scores for parsers using either gold lemmas, forms or gold attributes, either as one-hot vectors or as dense embeddings.

Table 5 reports results for the predicted attributes experiment. Are also reported, scores for the four representation regimes of predicted attributes as in table 4. Predictions can be either probability distributions (Soft) or argmax (Hard) and either used as such (OH) or passed through an embedding (Emb).

The Lazy Encoder: A Fine-Grained Analysis of the Role of Morphology in Neural Machine Translation

Arianna Bisazza
Leiden University
The Netherlands
a.bisazza@liacs.leidenuniv.nl

Clara Tump
KTH Royal Institute of Technology
Sweden
clara.tump@hotmail.com

Abstract

Neural sequence-to-sequence models have proven very effective for machine translation, but at the expense of model interpretability. To shed more light into the role played by linguistic structure in the process of neural machine translation, we perform a fine-grained analysis of how various source-side morphological features are captured at different levels of the NMT encoder while varying the target language. Differently from previous work, we find no correlation between the accuracy of source morphology encoding and translation quality. We do find that morphological features are only captured in context and only to the extent that they are directly transferable to the target words.

1 Introduction

The advent of Neural Machine Translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2014) has led to remarkable improvements in machine translation quality (Bentivogli et al., 2016) but has also produced models that are much less interpretable. In particular, the role played by linguistic features in the process of understanding the source text and rendering it in the target language remains hard to gauge. Acquiring this knowledge is important to inform future research in NMT, especially regarding the usefulness of injecting linguistic information into the NMT model, e.g. by using supervised annotation (Sennrich and Haddow, 2016).

Hill et al. (2014) gave a first answer to this question, reporting high accuracies by source-side NMT word embeddings on the well-known analogy task by Mikolov et al. (2013) which also includes a number of derivational and inflectional transformations in the morphologically poor English language. More recent work (Shi et al., 2016) has shown that source *sentence* representations produced by NMT encoders contain a great deal of syntactic information. Belinkov et al. (2017a) focused on the *word* level and examined to what extent part-of-speech and morphological information can be extracted from various NMT word representations. The latter study found that source-side morphology is captured slightly better by the first recurrent layer than by the word embedding and the final recurrent layer. Another, somewhat surprising finding was that source-side morphology is learned better when translating into an 'easier' target language than into a related one, even if the 'easier' language is morphologically poor.

In this paper, we also focus on source-side morphology but perform a finer-grained analysis of how morphological features are captured by different components of the NMT encoder while varying the target language. We argue that predicting generic morphological tags where all features are mixed, as done by Belinkov et al. (2017a), can only give us a limited insight into the linguistic competence of the model. Hence, we predict morphological features independently from one another and ask the following questions:

- Are different morphological features captured by the NMT encoder to substantially different extents and, if yes, why?
- Are morphological features captured as a word type property (i.e. at the word embedding level) or are they mostly computed in context (i.e. at the recurrent state level)?
- How does source-target language relatedness affect the morphological competence of the NMT encoder?

More specifically, we look at whether the NMT encoder only learns those morphological features that can be directly transferred to the target words

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2871–2876
Brussels, Belgium, October 31 - November 4, 2018.

FR: Les **arbres**$_{pl}$ sont hauts.
IT: Gli **alberi**$_{pl}$ sono alti.
DE: Die **Bäume**$_{pl}$ sind hoch.
EN: The **trees**$_{pl}$ are high.

FR: La **pomme**$_{f}$ est petite.
IT: La **mela**$_{f}$ è piccola.
DE: Der **Apfel**$_{m}$ ist klein.
EN: The **apple** is small.

Figure 1: Two example French sentences translated to Italian, German, and English. Number (top) is usually carried over to the three target languages, while gender (bottom) is less predictable. Colored fonts mark agreement with the noun in boldface.

(such as *number*) or whether it also learns features that are not directly transferable but can still be useful to correctly parse and infer the meaning of a sentence (such as *gender*). See example in Fig. 1.

We focus on French and similarly to previous work (Shi et al., 2016; Belinkov et al., 2017a) we use the continuous word representations produced by a trained NMT system to build and evaluate a number of linguistic feature classifiers. Classifier accuracy represents the extent to which a given feature is captured by the NMT encoder.

2 Methods

We train NMT systems on the following language pairs: French-Italian (FR_{IT}), French-German (FR_{DE}), and French-English (FR_{EN}). We chose these language pairs for their different levels of language relatedness and morphological feature correspondence. Grammatical gender is especially interesting as it is marked in French, Italian and German, but not in English (except for a few pronouns). The gender of Italian nouns often corresponds to that of French because of their common language ancestor, whereas German gender is mostly unrelated from French gender (see example in Fig. 1).

The continuous word representations produced by the three NMT systems while encoding a corpus of French sentences are used to build and evaluate several specialized classifiers: one per morphological feature. If a classifier significantly outperforms the majority baseline, we conclude that the corresponding feature is captured by the NMT encoder. While this methodology is similar to that of previous work (Köhn, 2015; Belinkov et al., 2017a,b; Dalvi et al., 2017) we make sure that our results are not affected by overfitting by eliminating any vocabulary overlap between the classifier's training and test sets. We find this step crucial to ensure that the redundancy in this type of data does not lead to over-optimistic conclusions. We now provide more details on the experimental setup.

Parallel corpora. For a fair comparison among target languages, we extract the intersection of the Europarl corpus (Koehn, 2005) in our three language pairs so that the source side data is identical for all NMT systems. Sentences longer than 50 tokens are ignored. This data is then split into an NMT training, validation, and test set of 1.3M, 2.5K, and 2.5K sentence pairs respectively.

NMT model. The NMT architecture is an attentional encoder-decoder model similar to (Luong et al., 2015) and uses a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) as the recurrent cell. The models have 3 stacked LSTM layers and are trained for 15 epochs. Embedding and hidden state sizes are set to 1000. Source and target vocabularies are limited to the 30,000 most frequent words on each side of the training data.[1] The NMT models achieve a test BLEU score of **32.6**, **25.4** and **39.4** for French-Italian, French-German and French-English respectively.

Continuous word representations. Given a source sentence, the NMT system first encodes it into a sequence of word embeddings (context-independent representations), and then into a sequence of recurrent states (context-dependent representations). As we are mostly interested in the impact of context on word representations, we compare the word embeddings against the final layer of the stacked LSTMs (corresponding to layers 0 and 3 in Belinkov et al. (2017a)'s terms) while disregarding the intermediate layers.

Morphological classification. The continuous word representations are used to train a logistic regression classifier[2] for each morphological feature: *gender* and *number* for noun and adjectives; *tense* for verbs (with labels: present, fu-

---

[1] Subword/character-level representations are not included in this study since we are interested in the models' ability to learn morphology from word usage, rather than word form.

[2] We use linear classifiers since their accuracies can be interpreted as a measure of supervised clustering accuracy, which gives a better insight on the structure of the vector space (Köhn, 2015). Results with a simple multi-layer perceptron were consistent with the findings by the linear classifier, with slightly better performance overall.

ture, imperfect, or simple past). Word labels are taken from the *Lefff* French morphological lexicon (Sagot, 2010)[3]. To ensure a fair comparison between context-independent and context-dependent embedding classification, words that are ambiguous with respect to a given feature are excluded from the respective classifier's training and test data.

Classifiers' training/test data. The classifiers are trained on a 50K-sentence subset of the NMT training data and tested on the NMT test sets (2.5K). For each experiment, we extract *one vector per token* from the NMT encoder. While this is the only possible setup for context-dependent representations, it leads to a problematic training/test overlap in the word embedding experiment because all occurrences of the same word are associated to exactly the same vector. We find that, due to this overlap, a dummy binary feature assigned to a random half of the vocabulary can be predicted from the word embeddings with very high accuracy (86% for a linear, 98% for a non-linear classifier) leading to over-optimistic conclusions on the linguistic regularities of these representations. To avoid this, we split the vocabulary in two parts of 15K types: the first is used to filter the training samples and the second to filter the test samples. We repeat each experiment five times using five different random vocabulary splits and report mean accuracies. This process is applied to all experiments (including those on hidden states) to allow for a fair comparison of the results.

3 Results and Discussion

This section presents our results along three dimensions: context-dependency of the word representations (§3.1), different morphological features (§3.2), and target language impact (§3.3). Unless explicitly stated, all discussed results are statistically significant (computed using a t-test for a one-tailed hypothesis and independent means).

3.1 Word embeddings *vs* recurrent states

One of our goals was to discover whether morphological features are captured as a word type property or in context. Fig. 2 shows the extent to which the NMT encoder captures different features at the word level (word embeddings) compared to the recurrent state level (LSTM state), averaged over all target languages. We can see that each feature is clearly captured at the recurrent state level, confirming that source-side morphology is indeed successfully exploited by NMT models. However, at the word embedding level, accuracies are comparable to the majority class baseline (these differences are not significant), which implies that the source-side lexicon of our NMT systems does not encode morphology in a systematic way. This might be partly explained by the fact that learning morphological features at the word level is difficult due to data sparsity – indeed the rarest French words in our dataset are observed only 10 times in the training data. However, additional experiments showed that our finding is consistent across different word frequency bins: that is, even the embeddings of frequent words do not encode morphological features better than the majority baseline.

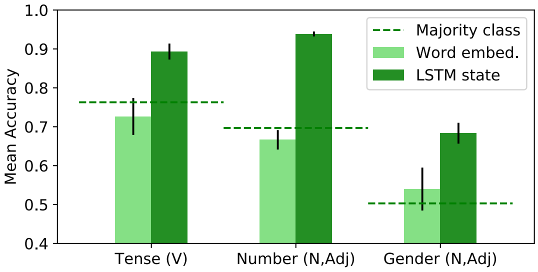

Figure 2: Classifier accuracy for different morphological features, averaged over target languages.

This result is surprising, considering that our morphological features are usually easy to infer from the immediate context of French words (see examples in Fig.1) and that morphology was shown to be well captured by monolingual word embeddings in various European languages including French (Köhn, 2015). By contrast, our NMT encoders choose not to store morphology at the word type level, perhaps in order to allocate more capacity to semantic information.

3.2 Different morphological features

Secondly, we asked whether the NMT encoder captured different morphological features to different extents. For this question, we disregard the word embedding results because none of the features are significantly captured at this level.

Fig. 2 shows that the mean accuracy of *number* is the highest, followed by *tense* and then by *gender*. However, it should be noted that the majority baselines for number and tense are much higher than the one for gender. In both absolute and rela-

---

[3] Lexique des Formes Fléchies du Français: `http://alpage.inria.fr/~sagot/lefff-en.html`

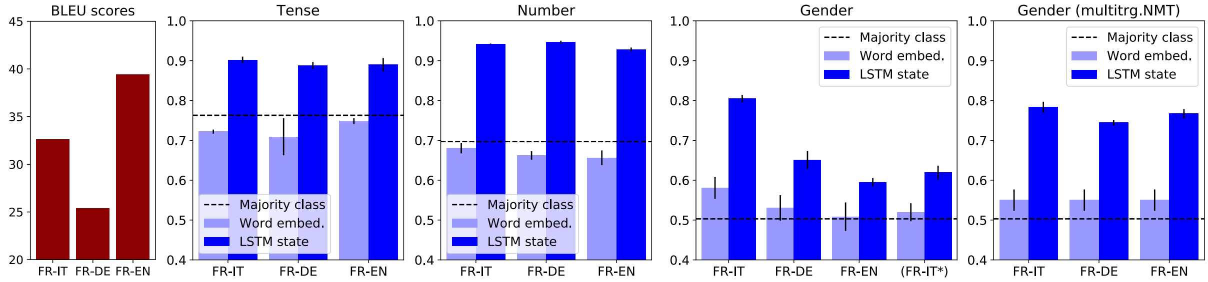

Figure 3: Translation quality (left) and classifier accuracy for each morphological feature in the three different language pairs. IT* denotes a modified Italian language where gender marking is removed.

tive terms, the best performing feature is number. This can be explained by the fact that number remains most often unchanged through translation, and is marked in all target languages – albeit to different extents. On the other hand, tense is determined by the semantics but also by language-specific usage, while gender has little semantic value and is mostly assigned to nouns arbitrarily.

The fact that the results of different morphological features are so variable confirms the setup of examining each feature independently.

3.3 Source-target language relatedness

Fig. 3 shows the impact of the target language on the encoded morphology accuracy. We again focus our analysis on the LSTM state level since embedding level results are mostly near the baseline.

Differently from Belinkov et al. (2017a) we do not find that source-side morphology is captured better when translating into the 'easiest' language, which in our case is English, both in terms of morphological complexity and BLEU performance. We note that their findings were based on very small, possibly not significant differences, and on the prediction of all morphological features simultaneously. By contrast, our fine-grained analysis reveals that the impact of target language is significant and even major on only one feature, namely *gender*, where it agrees with our linguistic intuition. Indeed this feature differs from the others because it varies largely among languages and, when present, is semantically determined only to a very limited extent. FR_{IT}, where source gender is a good predictor of target gender, shows the highest accuracy; FR_{EN}, where target gender is not marked, shows the lowest; FR_{DE}, where source gender is often confounding for target gender, lies in-between.

Is language relatedness the main explaining variable? To find that out, we experiment with a modified Italian target language without gender marking, i.e. all gender-marked words are replaced by their masculine form (FR_{IT*}). This language pair achieves a slightly higher BLEU score than FR_{IT} (33.2 *vs* 32.6), which can be attributed to the smaller target vocabulary. However its source gender accuracy is much worse (see Fig. 3), which indicates that the high performance of the FR_{IT} encoder is mostly due to the ubiquitous gender marking in the target language, rather than to language relatedness. All this suggests that source morphological features contribute to sentence understanding to some degree, but the incentive to learn them mostly depends on how directly they can be transferred to the target sentence.

Finally, we look at what happens when a single NMT system is trained in a multitarget fashion on our three language pairs. Following the setup of Johnson et al. (2017), we prepend a *to-target-language* tag $\{2it, 2de, 2en\}$ to the source side of each sentence pair and mix all language pairs in the NMT training data. Results are presented for gender in Fig. 3 (right).[4] Note that, while word embeddings are identical for the three language pairs, recurrent states change according to the language tag. In this setup the target language impact is less visible and gender accuracy at the LSTM state level is overall much higher than that of the mono-target systems (0.77 *vs* 0.68 on average) whereas BLEU scores are slightly lower (-0.9% on average). While this is only an initial exploration of multilingual NMT systems, our results suggest that this kind of multi-task objective pushes the model to learn linguistic features in a more consistent way (Bjerva, 2017; Enguehard et al., 2017).

---

[4] Multitarget results for tense and number did not differ significantly from the corresponding monotarget results.

4 Conclusion

We have confirmed previous findings that morphological features are significantly captured by word-level NMT encoders. However, the features are not captured at the word type level but only at the recurrent state level where word representations are context-dependent. Secondly, there is a visible difference in the extent to which different morphological features are learned: Semantic categories like number and verb tense are well captured in all language pairs, whereas grammatical gender with its only agreement-triggering function, is dramatically affected by the target language. Source-side gender is encoded well only when it is a good predictor of target gender and when target-side marking is extensive, i.e. when translating from French to Italian.

Our findings indicate that the importance of linguistic structure for the neural translation process is very variable and language-dependent. They also suggest that the NMT encoder is rather 'lazy' when it comes to learning grammatical features of the source words, unless these are directly transferable to their target equivalents.

Acknowledgments

This research was partly funded by the Netherlands Organization for Scientific Research (NWO) under project number 639.021.646. Part of the work was carried out on the DAS computing system (Bal et al., 2016) while the authors were affiliated at the Informatics Institute of the University of Amsterdam. We thank Ke Tran for providing feedback on the early stages of this research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Henri Bal, Dick Epema, Cees de Laat, Rob van Nieuwpoort, John Romein, Frank Seinstra, Cees Snoek, and Harry Wijshoff. 2016. A medium-scale distributed system for computer science research: Infrastructure for the long term. *Computer*, 49(5):54–63.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas. Association for Computational Linguistics.

Johannes Bjerva. 2017. *One Model to Rule them All: Multitask and Multilingual Modelling for Lexical Analysis*. Ph.D. thesis, University of Groningen.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 142–151. Asian Federation of Natural Language Processing.

Émile Enguehard, Yoav Goldberg, and Tal Linzen. 2017. Exploring the syntactic abilities of rnns with multi-task learning. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 3–14, Vancouver, Canada. Association for Computational Linguistics.

Felix Hill, KyungHyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014. Not all neural embeddings are born equal. In *NIPS 2014 Workshop on Learning Semantics*.

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

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernand a Vigas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Arne Köhn. 2015. What's in an embedding? analyzing word embeddings through multilingual evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages

2067–2073, Lisbon, Portugal. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Benoît Sagot. 2010. The lefff, a freely available and large-coverage morphological and syntactic lexicon for french. In *7th international conference on Language Resources and Evaluation (LREC 2010)*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Imitation Learning for Neural Morphological String Transduction

Peter Makarov **Simon Clematide**
Institute of Computational Linguistics
University of Zurich, Switzerland
makarov@cl.uzh.ch simon.clematide@cl.uzh.ch

Abstract

We employ imitation learning to train a neural transition-based string transducer for morphological tasks such as inflection generation and lemmatization. Previous approaches to training this type of model either rely on an external character aligner for the production of gold action sequences, which results in a suboptimal model due to the unwarranted dependence on a single gold action sequence despite spurious ambiguity, or require warm starting with an MLE model. Our approach only requires a simple expert policy, eliminating the need for a character aligner or warm start. It also addresses familiar MLE training biases and leads to strong and state-of-the-art performance on several benchmarks.

1 Introduction

Recently, morphological tasks such as inflection generation and lemmatization (Figure 1) have been successfully tackled with neural transition-based models over edit actions (Aharoni and Goldberg, 2017; Robertson and Goldwater, 2018; Makarov and Clematide, 2018; Cotterell et al., 2017b). The model, introduced in Aharoni and Goldberg (2017), uses familiar inductive biases about morphological string transduction such as conditioning on a single input character and monotonic character-to-character alignment. Due to this, the model achieves lower time complexity (compared to soft-attentional seq2seq models) and strong performance on several datasets.

하다 {V, Past, Formal Polite, Interrogative, Honorific} ▷ 하셨습니까

하셨습니까 ▷ 하다

Figure 1: Morphological tasks with examples in Korean: inflection generation (top) and lemmatization (bottom). (McCune-Reischauer: 하다=hada, 하셨습니까=hasyŏssŭmnikka).

Aharoni and Goldberg train the model by maximizing the conditional log-likelihood (MLE) of gold edit actions derived by an independent character-pair aligner. The MLE training procedure is therefore a pipeline, and the aligner is completely uninformed of the end task. This results in error propagation and the unwarranted dependence of the transducer on a single gold action sequence—in contrast to weighted finite-state transducers (WFST) that take into account all permitted action sequences. Although these problems—as well as the exposure bias and the loss-metric mismatch arising from this MLE training (Wiseman and Rush, 2016)—can be addressed by reinforcement learning-style methods (Ranzato et al., 2016; Bahdanau et al., 2017; Shen et al., 2016, RL), for an effective performance, all these approaches require warm-start initialization with an MLE-pretrained model. Another shortcoming of the RL-style methods is delayed punishment: For many NLP problems, including morphological string transduction, one can pinpoint actions that adversely affect the global score. For example, it is easy to tell if inserting some character c at step t would render the entire output incorrect. Assigning individual blame to single actions directly—as opposed to scoring the entire sequence via a sequence-level objective—simplifies the learning problem.

Faced with problems similar to those arising in transition-based dependency parsing with static oracles (Goldberg and Nivre, 2012), we train this model in the imitation learning (IL) framework (Daumé III et al., 2009; Ross et al., 2011; Chang et al., 2015), using a simple expert policy. Our approach eliminates both all dependency on an external character aligner and the need for MLE pre-training. By making use of exploration of past

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2877–2882
Brussels, Belgium, October 31 - November 4, 2018.

and future actions and having a global objective, it addresses the MLE training biases, while providing relevant action-level training signal. The approach leads to strong and state-of-the-art results on a number of morphological datasets, outperforming models trained with minimum risk training (MRT).

2 Model Description

We use a variant of the seq2seq state-transition system by Aharoni and Goldberg (2017). The model transduces the input string into the output string by performing single-character edits (insertions, deletions). The encoder RNN computes context-enriched representations of input characters, which are pushed onto the buffer at the beginning of transduction. The decoder RNN keeps track of the history of edits. Transitions—edits—are scored based on the output of the decoder state and can write a character or pop the representation of a character from the top of the buffer. We choose the model variant of Makarov and Clematide (2018), who add the copy edit, which results in strong performance gains in low-resource settings.

Let $\mathbf{x} = x_1 \dots x_n$, $x_i \in \Sigma_x$ be an input sequence, $\mathbf{y} = y_1 \dots y_p$, $y_j \in \Sigma_y$ an output sequence, and $\mathbf{a} = a_1 \dots a_m$, $a_t \in \Sigma_a$ an action sequence. Let $\{f_h\}_{h=1}^{H}$ be the set of all features. The morpho-syntactic description of a transduction is then an n-hot vector $\mathbf{e} \in \{0,1\}^H$.

The model employs a bidirectional long short-term memory (LSTM) encoder (Graves and Schmidhuber, 2005) to produce representations for each character of the input $\mathbf{x}$:

$$\mathbf{h}_1, \dots, \mathbf{h}_n = \text{BiLSTM}(E(x_1), \dots, E(x_n)), \quad (1)$$

where E returns the embedding for x_i. We push $\mathbf{h}_1, \dots, \mathbf{h}_n$ in reversed order onto the buffer. The transduction begins with the full buffer and the empty decoder state.

Transitions are scored based on the output of the LSTM decoder state (Hochreiter and Schmidhuber, 1997):

$$\mathbf{s}_t = \text{LSTM}(\mathbf{c}_{t-1}, [A(a_{t-1})\ ;\ \mathbf{h}_i]), \quad (2)$$

where $\mathbf{c}_{t-1}$ is the previous decoder state, $A(a_{t-1})$ is the embedding of the previous edit action, and $\mathbf{h}_i$ is the input character representation at the top of the buffer. If features are part of the input in the task, then the input to the decoder also contains the representation of morpho-syntactic description $\mathbf{e}$, $[F(f_1)\,; \dots ; F(f_H)]$, which is a concatenation of the embedded features and a designated embedding $F(0)$ is used instead of $F(f_h)$ if $e_h = 0$.

The probabilities of transitions are computed with a softmax classifier:

$$P(a_t = k \mid \mathbf{a}_{<t}, \mathbf{x}, \Theta) = \text{softmax}_k(\mathbf{W} \cdot \mathbf{s}_t + \mathbf{b}) \quad (3)$$

Model parameters Θ include $\mathbf{W}$, $\mathbf{b}$, the embeddings, and the parameters of the LSTMs.

The alphabet of edit actions Σ_a contains INSERT(c) for each $c \in \Sigma_y$, DELETE, and COPY. An INSERT(c) action outputs c; DELETE pops $\mathbf{h}_i$ from the top of the buffer; COPY pops $\mathbf{h}_i$ from the top of the buffer and outputs x_i. The system exhibits spurious ambiguity: Multiple action sequences lead to the same output string.

2.1 MLE Training

Aharoni and Goldberg train their model by minimizing the negative conditional log-likelihood of the data $D = \{(\mathbf{x}^{(l)}, \mathbf{a}^{(l)})\}_{l=1}^{N}$:

$$\mathcal{L}(D, \Theta) = -\sum_{l=1}^{N} \sum_{t=1}^{m} \log P(a_t^{(l)} \mid \mathbf{a}_{<t}^{(l)}, \mathbf{x}^{(l)}, \Theta), \quad (4)$$

where gold action sequences $\mathbf{a}^{(l)}$ are deterministically computed from a character-pair alignment of the input and output sequences $(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})$. The character-pair aligner is trained separately to optimize the likelihood of the actual training data $T = \{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\}_{l=1}^{N}$. For the details, we refer the reader to Aharoni and Goldberg (2017).

3 IL Training

One problem with the MLE approach is that the aligner is trained in a disconnect from the end task. As a result, alignment errors lead to the learning of a suboptimal transducer. Switching to a different aligner can dramatically improve performance (Makarov and Clematide, 2018). More fundamentally, in the face of the vast spurious ambiguity, the transducer is forced to adhere to a single gold action sequence whereas typically, legitimate and equally likely alternative edit sequences exist. This uncertainty is not accessible to the transducer, but could be profitably leveraged by it.

We address this problem within the IL framework and train the model to imitate an expert policy (dynamic oracle), which is a map—on the training data—from configurations to sets of optimal actions. Actions are optimal if they lead to the lowest sequence-level loss, under the assumption that all future actions are also optimal (Daumé III et al., 2009). In the *roll-in* stage, we run the model on a training sample and follow actions either returned by the expert policy (as in teacher forcing) or sampled from the model (which itself is a stochastic policy). In this way, we obtain a sequence of configurations summarized as decoder outputs $\mathbf{s}_1, \ldots, \mathbf{s}_m$. In the *roll-out* stage, we compute the sequence-level loss for every valid action a in each configuration $\mathbf{s}_t$. To this end, we execute a and then either query the expert to obtain the loss for the optimal action sequence following a or run the model for the rest of the input and evaluate the loss of the resulting action sequence. Finally, the sequence-level losses obtained in this way for all actions a enter the action-level loss for configuration $\mathbf{s}_t$ that we minimize with respect to Θ.

Sequence-level loss We define the loss in terms of the *Levenshtein distance* (Levenshtein, 1966) between the prediction and the target and the *edit cost* of the action sequence. Given input $\mathbf{x}^{(l)}$ with target $\mathbf{y}^{(l)}$, the loss from producing an action sequence $\mathbf{a}$ is:

$$\ell(\mathbf{a}, \mathbf{x}^{(l)}, \mathbf{y}^{(l)}) = \beta\, \text{distance}(\mathbf{y}, \mathbf{y}^{(l)}) + \text{cost}(\mathbf{a}), \quad (5)$$

where $\mathbf{y}$ is computed from $\mathbf{a}$ and $\mathbf{x}^{(l)}$ and $\beta \geq 1$ is some penalty for unit distance.[1] The first term represents the task objective. The second term enforces that the task objective is reached with a minimum number of edits.

The second term is crucial as it takes over the role of the character aligner. Initially, we also experimented with only Levenshtein distance as loss, similar to previous work on character-level problems (Leblond et al., 2018; Bahdanau et al., 2017). However, models did not learn much, which we attribute to sparse training signal as all action sequences producing the same $\mathbf{y}$ would incur the same sequence-level loss, including intuitively very wasteful ones, e.g. first deleting all of $\mathbf{x}^{(l)}$ and then inserting of all of $\mathbf{y}^{(l)}$.

Expert The expert policy keeps track of the prefix of the target $\mathbf{y}^{(l)}$ in the predicted sequence $\mathbf{y}_{<t}$ and returns actions that lead to the completion of the suffix of $\mathbf{y}^{(l)}$ using an action sequence with the lowest edit cost. The resulting prediction $\mathbf{y}$ attains the minimum edit distance from $\mathbf{y}^{(l)}$. For example, if $\mathbf{x}^{(l)} = walk$ and $\mathbf{y}^{(l)} = walked$, the top of the buffer is $\mathbf{h}_3$ representing $x_3 = l$, and $\mathbf{y}_{<3} = wad$ due to a sampling error from a roll-in with the model, the expert returns $\{\text{COPY}\}$.

Action-level loss Given sequence-level losses, we compute the regret for each action a:

$$r_t(a) = \ell(\mathbf{a}, \mathbf{x}^{(l)}, \mathbf{y}^{(l)}) - \min_{a' \in A(\mathbf{s}_t)} \ell(\mathbf{a}', \mathbf{x}^{(l)}, \mathbf{y}^{(l)}), \quad (6)$$

where $\mathbf{a}$ (or $\mathbf{a}'$) is the action sequence resulting from taking a (or a') at $\mathbf{s}_t$ and $A(\mathbf{s}_t)$ is the set of valid actions. Thus, $r_t(a)$, which quantifies how much we suffer from taking action a relative to the optimal action under the current policy, constitutes the direct blame of a in the sequence-level loss.

Classic IL employs cost-sensitive classification, with regrets making up costs (Daumé III et al., 2009; Chang et al., 2015). Our initial experiments with cost-sensitive classification resulted in rather inefficient training and not very effective models. Instead, we choose to minimize the negative marginal log-likelihood of all optimal actions (Riezler et al., 2000; Goldberg, 2013; Ballesteros et al., 2016). Given the training data $T = \{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\}_{l=1}^{N}$, the action-level loss is:

$$\mathcal{L}(T, \Theta) = -\sum_{l=1}^{N} \sum_{t=1}^{m} \log \sum_{a \in A_t} P(a \,|\, \mathbf{a}_{<t}, \mathbf{x}^{(l)}, \Theta), \quad (7)$$

where $A_t = \{a \in A(\mathbf{s}_t) : r_t(a) = 0\}$, the set of optimal actions under the current policy. Depending on the roll-in schedule, the next edit a_{t+1} is sampled either uniformly at random from A_t or from the distribution of valid edits. To include all the computed regrets into the loss, we also experiment with the cost-augmented version of this objective (Gimpel and Smith, 2010), where regrets function as costs.

The downside of IL is that roll-outs are costly. We avoid computing most of the roll-outs by checking if an action increases the edit distance from $\mathbf{y}^{(l)}$. If it does, we heuristically assign this action a regret of β. We use this heuristic in both expert and model roll-outs.

4 Experiments

We demonstrate the effectiveness of our approach on three tasks: inflection generation (using the typologically diverse SIGMORPHON 2016 and SIGMORPHON 2017 datasets of Cotterell et al. (2016, 2017a)), reinflection (the small-sized German CELEX dataset of Dreyer et al. (2008)), and

---

[1] We use unit costs to compute edit cost and distance.

Model	RU	DE	ES	KA	FI	TR	HU	NV	AR	MT	Avg.
MED	91.5	95.8	98.8	98.5	95.5	98.9	96.8	91.5	**99.3**	89.0	95.6
SOFT	92.2	96.5	98.9	**98.9**	97.0	**99.4**	97.0	**95.4**	**99.3**	88.9	**96.3**
HA	92.2	96.6	98.9	98.1	95.9	98.0	96.2	93.0	98.8	88.3	95.6
HA*	92.0	96.3	98.9	97.9	95.8	97.6	98.8	92.1	95.1	87.8	95.2
CA	91.9	96.4	98.8	98.3	96.5	97.7	98.9	92.1	94.6	87.7	95.3
CA-D	**92.4**	**96.6**	**98.9**	98.7	97.2	98.5	99.3	95.2	96.5	**89.2**	96.2
CA-R	92.3	96.5	98.9	**98.9**	**97.3**	98.9	**99.4**	95.2	96.1	88.8	96.2

Table 1: Results on SIGMORPHON 2016 data.[2]

Model	L	M
SGM17TOP	50.6	82.8
HA*	31.5	80.2
CA	48.8	81.0
HA*-MRT	33.1	81.5
CA-MRT	49.9	82.9
CA-MRT-A	49.9	82.7
CA-D	50.3	82.6
CA-R	**51.6**	83.8
CA-RM	50.6	**84.0**

Table 4: Results on **L**ow and **M**edium settings of SIGMORPHON 2017 data (averaged over 52 languages).

-MRT: minimum risk training; **-MRT-A**: MRT with action cost in the loss; **-D**: only expert roll-outs; **-R**: expert and model roll-outs; **-RM**: softmax-margin, expert and model roll-outs

Model	13SIA	2PIE	2PKE	rP	Avg.
LAT	**87.5**	93.4	87.4	84.9	88.3
NWFST	85.1	94.4	85.5	83.0	87.0
HA*	84.6	93.9	88.1	85.1	87.9
CA	85.0	94.5	88.0	84.9	88.1
HA*-MRT	84.8	94.0	88.1	85.2	88.0
CA-MRT	85.6	**94.6**	88.0	85.3	88.4
CA-D	85.7	94.4	**88.4**	85.1	88.4
CA-R	85.6	94.4	88.3	**85.3**	**88.4**
CA-RM	84.9	94.1	88.3	85.0	88.1

Table 2: Results on CELEX data.

Model	EU	EN	GA	TL	Avg.
LAT	93.6	96.9	97.9	88.6	94.2
NWFST	91.5	94.5	97.9	97.4	95.3
LEM[3]	96.5	96.3	**98.7**	**98.8**	97.6
HA*	**97.0**	**97.5**	97.9	98.3	**97.7**
CA	96.3	96.9	97.7	98.3	97.3
CA-D	96.1	97.0	97.7	98.4	97.3
CA-R	96.6	97.2	97.5	98.3	97.4
CA-RM	96.5	97.0	97.8	98.3	97.4

Table 3: Lemmatization results.

Experimental results. Soft-attention seq2seq models: MED=Kann and Schütze (2016) (cited from Aharoni and Goldberg (2017)), SOFT=Aharoni and Goldberg (2017), LEM=Bergmanis and Goldwater (2018). WFSTs: LAT=Dreyer et al. (2008), NWFST=Rastogi et al. (2016). Transition-based models: HA=Aharoni and Goldberg (2017), SGM17TOP=Makarov et al. (2017), and from Makarov and Clematide (2018): HA*=reimplementation of HA, CA=model in §2, and HA*-MRT, CA-MRT (risk=normalized edit distance). We report exact-match accuracies for ensembles of 5 models (SIGM. 2016 and 2017) and single-model averages over 5 folds (CELEX) and 10 folds (lemmatization).

lemmatization (the standard subset of the Wicentowski (2002) dataset).

We use character and feature embeddings of size 100 and 20, respectively, and one-layer LSTMs with hidden-state size 200. Following Aharoni and Goldberg, for every character $c \in \Sigma_x \cap \Sigma_y$, we let $A(\text{INSERT}(c)) := E(c)$, i.e. the same embedding represents both c and the insertion of c. We optimize with ADADELTA (Zeiler, 2012), use early stopping and batches of size 1. We set the penalty for unit distance $\beta = 5$ and roll in with an inverse-sigmoid decay schedule as in Bengio et al. (2015). CA-D models are trained with expert roll-outs only (as is often the case in dynamic-oracle parsing). CA-R and CA-RM models mix expert and learned roll-outs with probability 0.5 as in Chang et al. (2015). CA-RM models optimize softmax-margin.

For comparison, we also train models with MRT (CA-MRT-A) as in Shen et al. (2016), using a global objective similar to our sequence-level loss (Eq. 5). We use batches of at most 20 unique samples per training example. The risk is a convex combination of normalized Levenshtein distance and the action sequence cost, which we min-max scale, within a batch, to the [0, 1] interval.

We decode all our models using beam search with beam width 4.

Our approach performs best on most languages of the SIGMORPHON 2016 data (Table 1) and both limited-resource settings of SIGMORPHON 2017 (Table 4). It achieves marginal improvement over an MRT model on the reinflection task (Table 2) with consistent gains on the 2PKE$\mapsto$z transformation (Dreyer et al., 2008), that involves infixation. Using mixed roll-outs (CA-R, CA-RM) improves performance on the SIGMORPHON 2017 inflection data (Table 4), otherwise the results are close to CA-D. We also note strong gains over CA-MRT-A trained with a similar global loss (Table 4). Generally, improvements are most pronounced in inflection generation, the only task where the model could profit from adjusting alignment to available feature information (cf. Table 3).

---

[2] Language codes: RU=Russian, DE=German, ES=Spanish, KA=Georgian, FI=Finnish, TR=Turkish, HU=Hungarian, NV=Navajo, AR=Arabic, MT=Maltese, EU=Basque, EN=English, GA=Irish, TL=Tagalog.

[3] Personal communication.

We take a closer look at the results in the SIGMORPHON 2017 medium data-size setting (1,000 training examples per language). CA-RM makes the largest performance gains on languages with complex morphological phenomena (Semitic and Uralic languages, Navajo) and an above average number of unique morpho-syntactic descriptions. Khaling and Basque, outliers with 367 and 740 unique morpho-syntactic descriptions in the training data, are among the top five languages with the largest gains. The lowest gains and rare losses are made for Romance and Germanic languages and languages with many unique morpho-syntactic descriptions but regular morphologies (Quechua, Urdu/Hindi).

5 Related work

Traditionally, morphological string transduction has been approached with discriminative weighted finite-state transducers (Rastogi et al., 2016; Cotterell et al., 2014; Dreyer et al., 2008; Eisner, 2002). Yu et al. (2016) and Graves (2012) tackle the modeling of unbounded dependencies in the output, while preserving latent monotonic hard character alignment. Faruqui et al. (2016); Kann and Schütze (2016) successfully apply seq2seq modeling to the task. Aharoni and Goldberg (2017) introduce a neural version of the transition-based model over edits. Makarov and Clematide (2018) show gains from using the copy edit and address the MLE training biases with MRT.

The limitations of teacher forcing have recently been the focus of intense research (Edunov et al., 2017; Wiseman and Rush, 2016; Shen et al., 2016), including the adaptation of RL methods (Ranzato et al., 2016; Bahdanau et al., 2017). Most of these approaches require warm start with an MLE model and themselves introduce discrepancies between training with sampling and search-based decoding. Such biases do not arise from IL, which has recently been proposed for seq2seq models (Leblond et al., 2018). Our approach, related to Leblond et al. (2018), additionally addresses the problem of spurious ambiguity, which is not present in seq2seq models.

6 Conclusion

We show that training to imitate a simple expert policy results in an effective neural transition-based model for morphological string transduction. The fully end-to-end approach addresses various shortcomings of previous training regimes (the need for an external character aligner, warm-start initialization, and MLE training biases), and leads to strong empirical results. We make our code and predictions publicly available.[4]

Acknowledgments

We thank Mathias Müller for help with the manuscript, Toms Bergmanis for sharing with us the results of his system, and the reviewers for interesting and helpful comments. We also thank the anonymous COLING reviewer who suggested that we should look at spurious ambiguity. Peter Makarov has been supported by European Research Council Grant No. 338875, Simon Clematide by the Swiss National Science Foundation under Grant No. CRSII5_173719.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *ACL*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A Smith. 2016. Training with exploration improves a greedy stack-LSTM parser. In *EMNLP*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with Lematus. In *NAACL*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume III, and John Langford. 2015. Learning to search better than your teacher. In *ICML*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017a. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*.

---

[4] https://github.com/ZurichNLP/emnlp2018-imitation-learning-for-neural-morphology

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 Shared Task—Morphological Reinflection. In *Proceedings of the 2016 Meeting of SIGMORPHON*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic FSTs. In *ACL*.

Ryan Cotterell, John Sylak-Glassman, and Christo Kirov. 2017b. Neural graphical models over strings for principal parts morphological paradigm completion. In *EACL*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *ACL*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2017. Classical structured prediction losses for sequence to sequence learning. In *NAACL*.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *ACL*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *NAACL-HLT*.

Kevin Gimpel and Noah A Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *NAACL-HLT*.

Yoav Goldberg. 2013. Dynamic-oracle transition-based parsing with calibrated probabilistic output. In *IWPT*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *COLING*.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*.

Rémi Leblond, Jean-Baptiste Alayrac, Anton Osokin, and Simon Lacoste-Julien. 2018. SEARNN: Training RNNs with global-local losses. In *ICLR*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10.

Peter Makarov and Simon Clematide. 2018. Neural transition-based string transduction for limited-resource setting in morphology. In *COLING*.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *NAACL-HLT*.

Stefan Riezler, Jonas Kuhn, Detlef Prescher, and Mark Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *ACL*.

Alexander Robertson and Sharon Goldwater. 2018. Evaluating historical text normalization systems: How well do they generalize? In *NAACL*.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *ACL*.

Richard Wicentowski. 2002. *Modeling and learning multilingual inflectional morphology in a minimally supervised framework*. Ph.D. thesis, Johns Hopkins University.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *EMNLP*.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. arXiv:1212.5701.

An Encoder-Decoder Approach to the Paradigm Cell Filling Problem

Miikka Silfverberg
Department of Linguistics
University of Colorado / University of Helsinki
`first.last@colorado.edu`

Mans Hulden
Department of Linguistics
University of Colorado
`first.last@colorado.edu`

Abstract

The Paradigm Cell Filling Problem in morphology asks to complete word inflection tables from partial ones. We implement novel neural models for this task, evaluating them on 18 data sets in 8 languages, showing performance that is comparable with previous work with far less training data. We also publish a new dataset for this task and code implementing the system described in this paper.[1]

1 Introduction

An important learning question in morphology—both for NLP and models of language acquisition—is the so-called Paradigm Cell Filling Problem (PCFP). So dubbed by Ackerman et al. (2009), this problem asks how it is that speakers of a language can reliably produce inflectional forms of most lexemes without ever witnessing those forms before. For example, a Finnish noun or adjective can be inflected in 2,263 ways if one includes case forms, number, and clitics (Karlsson, 2008). However, it is unlikely that a Finnish speaker would have heard all forms for even a single, highly frequent lexical item. It is also unlikely that all 2,263 forms are found in the aggregate of all the witnessed inflected forms over different lexemes and speakers must be able to assess the felicity of, and possibly produce such inflectional combinations they have never witnessed for any noun or adjective. Figure 1 illustrates the PCFP.

This paper investigates PCFP in three different settings: (1) when we know $n > 1$ randomly selected forms in each of a number of inflection tables, (2) when we know a set of frequent word forms in each table (this most closely resembles an L1 language learning setting), and finally (3) when we know exactly $n = 1$ word form from each table.

We treat settings (1) and (2) as traditional morphological reinflection tasks (Cotterell et al., 2016) as explained in Section 2. In contrast, setting (3) is substantially more challenging because it cannot be handled using a traditional reinflection approach. To overcome this problem, we utilize an adaptive dropout mechanism which will be discussed in Section 2. This allows us to train the reinflection system in a manner reminiscent of denoising autoencoders (Vincent et al., 2008).

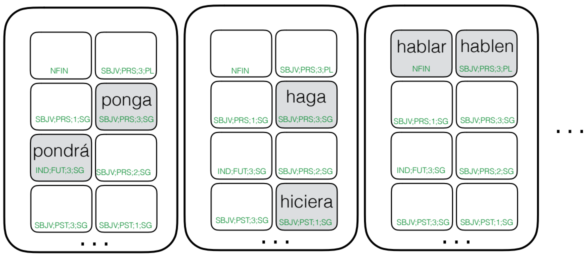

Figure 1: Illustration of the PCFP using a fraction of Spanish verb tables: given such partially filled paradigms, the task is to fill in all the missing forms.

Related Work Neural models have recently been shown to be highly competitive in many different tasks of learning supervised morphological inflection (Faruqui et al., 2016; Kann and Schütze, 2016; Makarov et al., 2017; Aharoni and Goldberg, 2017) and derivation (Cotterell et al., 2017b). Most current architectures are based on encoder-decoder models (Sutskever et al., 2014), and usually contain an attention component (Bahdanau et al., 2015).

The SIGMORPHON (Cotterell et al., 2016) and CoNLL-SIGMORPHON (Cotterell et al., 2017a, 2018) shared tasks in recent years have explored morphological inflection but not explicitly the PCFP. In the 2017 task, participants were given full paradigms—i.e. a listing of all forms—of

---

[1] `https://github.com/mpsilfve/pcfp-data`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2883–2889
Brussels, Belgium, October 31 - November 4, 2018.

lexemes during training after which they were given incomplete paradigms which had to be completed at test time. This is a slightly unrealistic setting in an L1-style learning scenario (Blevins and Blevins, 2009) where arguably very few full paradigms are ever witnessed and where generalization has to proceed on a number of very gappy paradigms. Of course, such gaps form a distribution where frequently used lexemes have fewer gaps than infrequent ones, which we will attempt to model in this work.

Silfverberg et al. (2018) evaluate an extension to a linguistically informed symbolic paradigm model based on stem extraction from the longest common subsequence (LCS) shared among related forms (Ahlberg et al., 2014, 2015). While the original LCS paradigm extraction method was intended to learn from complete inflection tables (Hulden, 2014), Silfverberg et al. (2018) present modifications to allow learning from incomplete paradigms as well, and apply it to the PCFP. Comparing against their results, shows that our neural model consistently outperforms such a subsequence-based learning model.

Kann et al. (2017) report results on so-called multi-source reinflection in which several input forms are used to generate one output form. This task is related to the PCFP; however, Kann et al. (2017) use full inflection tables for training. Moreover, their approach is applicable for PCFP only when 3 or more forms are given in the input tables. Since this mostly excludes our experimental settings, we do not compare to their system. Malouf (2016, 2017) documents an experiment with a generator LSTM in completing inflection tables in up to seven languages with either 10% or 40% of table entries missing. Our work differs from this in that Malouf gives as input a two-hot encoding of both the lexeme and the desired slot during training and testing for which an inflection table is to be completed, which means the system cannot complete paradigms which it has not seen examples of in the training data. By contrast, our system has no notion of lexeme and we simply work from the symbol strings which are collections of inflected forms of a lexeme given in the test data which may in principle be completely disjoint from training data lexemes. We use the Malouf system as a baseline to compare against.

2 Encoder-Decoder Models for PCFP

We explore two different models for paradigm filling. The first model is applicable when $n > 1$ forms are given in each inflection table. When exactly one ($n = 1$) form is given, we use another model.

Case n>1 When more than one form is given in training tables, PCFP can be treated as a morphological reinflection task (Cotterell et al., 2016), where the aim is to translate inflected word forms and their tags into target word forms. For example, a model would translate **tried+PAST** into the present participle (**PRES,PCPLE**) form **trying**. We adopt a common approach employed by Kann and Schütze (2016) and many others: we build a model which translates an input word form, its tag and a target tag, for example **tried+PAST+PRES,PCPLE**, into the target word form **trying**.

Our model closely follows the formulation of the encoder-decoder LSTM model for morphological reinflection proposed by Kann and Schütze (2016). We use a 1-layer bidirectional LSTM encoder for encoding the input word form into a sequence of state vectors and a 1-layer LSTM decoder with an attention mechanism over encoder states for generating the output word form.

We form training pairs by using the given forms in each table, i.e. take the cross-product of the given forms and learn to reinflect each given form in a table to another given form in the same table as demonstrated in Figure 2.[2] During test time, we predict forms for missing slots based on each of the given forms in the table and take a majority vote of the results.[3]

Case n=1 When only one form is given in each inflection table, we cannot train the model as a traditional reinflection model. The best we can do is to train a model to reinflect forms into the same form **walked+PAST+PAST** ↦ **walked** and then try to apply this model for reinflection to fill in missing forms **walked+PAST+PRES,PCPLE** ↦ **walking**. According to preliminary experiments, this however leads to massive over-fitting and the model simply learns to only copy input forms.

---

[2] Note that the CoNLL-SIGMORPHON data provides a 'citation form' that identifies each table; we do not use this form and the model has no knowledge of it.

[3] When only two forms are given in the partial inflection table, we randomly choose one of the resulting output forms since the vote is always tied.

```
augšanai    N,DAT,SG
            N,LOC,SG
            N,NOM,SG
            N,VOC,SG
            N,GEN,SG
augšanu     N,ACC,SG
            N,INST,SG
                ⇩
augšanu+N,ACC,SG+N,DAT,SG  → augšanai
augšanai+N,DAT,SG+N,ACC,SG → augšanu
```

Figure 2: Partial inflection table for the Latvian noun **augana** 'growth'. From a partial inflection table with two given forms, we get two training examples. With n given forms in a table, we hence produce $n(n-1)$ training examples.

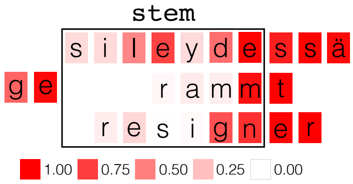

Figure 3: Language model confidences for a Finnish noun (singular inessive of **sileys** 'smoothness'), a German past participle (of **rammen** 'to ram') and a French verb (infinitive form of **resigner** 'to resign'). The figure demonstrates that confidence is higher in the inflectional affixes than in the stem in general. It is also high at the stem-affix boundary.

The idea for our approach in case $n = 1$ is to first learn to segment word forms into a stem and an affix, for example **walk+ed**. We then hide the affix in the input form and learn to inflect. In other words, we map the word form **walked** into **walk$$** and then learn a mapping **walk$$+PAST** $\mapsto$ **walked**. This model suffers less from overfitting and we can use it to find missing forms in partial inflection tables.

Since we do not have access to segmented training data, we cannot directly train a segmentation model. Instead, we use the forms in the training data to train an LSTM language model conditioned on morphological tags. We then use the language model for identifying which characters belong to stems and which characters belong to affixes.

As shown in Figure 3, the language model in general gives higher confidence for predictions of characters in the affix than in the word stem. Nevertheless, it only gives a probabilistic segmentation into a stem and affix(es). Therefore, we do not perform a deterministic segmentation. Instead we use the language model to guide a character dropout mechanism in our word inflection model. When the language model is very confident, as in the case of affix characters, we frequently drop characters. In contrast, when the language model is less confident, as in the case of stem characters, we typically keep the character. Apart from this adaptive dropout applied during training, our inflection system in case $n = 1$ is exactly the same as in case $n > 1$.

	Our baseline	Malouf (2017)
FINNISH NOUNS	99.50	99.27 ±0.09
FRENCH VERBS	99.88	99.92 ± 0.02
IRISH NOUNS	85.11	85.69 ±1.71
KHALING VERBS	99.66	99.29 ±0.08
MALTESE VERBS	98.65	98.93 ±0.32
P. CHINANTEC VERBS	91.16	91.20 ±0.97
RUSSIAN NOUNS	95.90	96.34 ±0.96

Table 1: We reproduce experiments in Malouf (2017) using our own implementation of the model. In contrast to Malouf (2017), who used cross-validation, we train one system for each language. Therefore, we only report standard deviation for the results in Column 2.

More precisely, given an input word form, which is a sequence of characters $x = x_1, ..., x_T$, the LSTM language model emits a probability $p(x_{t+1}, \mathbf{h_t}, \mathrm{E}_{x_t}, \mathrm{E}_y)$ for the next character x_{t+1} based on the entire previous input sequence $x_1, ..., x_t$. Here $\mathbf{h_t}$ is the hidden state vector of the language model at position t, E a joint tag and character embedding and y the morphological tag of the input word form. The embedding vector E_y is in fact a sum of sub-tag embeddings. For example, $\mathrm{E}_{\mathrm{PAST+PCPLE}}$ denotes $\mathrm{E}_{\mathrm{PAST}} + \mathrm{E}_{\mathrm{PCPLE}}$. This allows us to handle combinations of sub-tags which we have not seen in the training data. Guided by the language model, we replace input characters x_{t+1} during training of the reinflection system with a dropout character $ with probability equal to language model confidence $p(x_{t+1}, \mathbf{h_t}, \mathrm{E}_{x_t}, \mathrm{E}_y)$.[4]

Baseline Model As a baseline model, we use the neural system presented by Malouf (2016, 2017) for solving PCFP. It is an LSTM generator which is conditioned on the table number of the partial inflection tables and the morphological tag index. The model is trained to generate training word forms in inflection tables. During testing, it can then generate missing forms by conditioning on morphological tags for the missing forms.

In order to assure fair comparison, we perform the paradigm completion experiment described in Malouf (2017), where 90% of the word forms in the data set is used for training and the remaining 10% for testing. [5] As the results in Table 1 show,

---

[4] In practice, we pad input forms with end-of-sequence characters in order to be able to drop x_1 if needed.

[5] We perform the the experiments on the original data sets,

our results very closely replicate those reported by Malouf (2017).

Implementation details We use 1-layer bidirectional LSTM encoders, decoders and generators with embeddings and hidden states of size 100. We train the language model for case $n > 1$ for 20 epochs and all other models for 60 epochs without batching. We train 10 models for every language and part-of-speech and apply majority voting to get the final output forms. All models were implemented using DyNet (Neubig et al., 2017).

3 Data

We use UniMorph morphological paradigm data in our experiments (Kirov et al., 2018). Unimorph data sets are crowd-sourced collections of morphological inflection tables based on Wiktionary. We conduct experiments on noun and verb paradigms from eight languages.[6] Not all languages have 1,000 noun and verb tables. Hence, our selection is not complete as seen in Table 3.

We conduct experiments on two different sets of tables: (1) we randomly sample 1,000 tables for each language and part-of-speech, and (2) we select Unimorph tables including some of the 10,000 most common word forms according to Wikipedia frequency. The Wikipedia word frequencies are based on plain Wikipedia text dumps from the Polyglot project (Al-Rfou et al., 2013). Georgian and Latin did not have a Polyglot Wikipedia so we excluded those. Moreover, we excluded Latvian verbs because there was very little overlap between the most frequent Wikipedia word forms and Unimorph table entries (< 200 forms occurred in both). Details for both types of data sets are given in Tables 3 and 2.

	# Tables	Table Size
Finnish Nouns	1,335	27.3
Finnish Verbs	513	38.9
French Verbs	1,131	47.8
German Verbs	657	24.9
Latvian Nouns	802	12.8
Spanish Verbs	1,067	62.8
Turkish Nouns	884	78.5

Table 2: Details for inflection tables chosen according to Wikipedia word frequency.

	Table Size	Unique Forms per Table
Fin N	27.7	25.7
Fin V	39.0	37.6
Fre V	47.5	36.1
Geo N	19.0	16.9
Ger V	28.9	12.3
Lat N	11.9	7.2
Lat V	99.8	94.8
Lav N	11.6	7.6
Spa V	62.5	52.1
Tur N	74.4	54.8

Table 3: Details for randomly sampled inflection tables. The data for each language and part-of-speech consist of 1,000 tables.

	Our system	Baseline
Finnish Nouns	**63.64** ± 3.24	25.63 ± 1.63
Finnish Verbs	**24.82** ± 1.13	16.14 ± 1.14
French Verbs	**31.34** ± 1.18	14.34 ± 0.87
German Nouns	18.73 ± 1.26	**67.16** ± 3.20
German Verbs	**61.21** ± 1.85	50.18 ± 2.58
Latvian Nouns	**76.90** ± 5.30	57.28 ± 2.05
Spanish Verbs	**27.27** ± 0.72	16.61 ± 0.70
Turkish Nouns	**33.87** ± 2.03	25.00 ± 2.52

Table 4: Overall results for filling in missing forms when the 10,000 most frequent forms are given in the inflection tables. We give the 0.99 confidence intervals as given by a one-sided t-test. Figures where one system significantly outperforms the other one are in boldface.

4 Experiments and Results

We perform two experiments. In the first one, we take the set of 1,000 randomly sampled inflection tables for each language and part-of-speech and then randomly select n=1, 2 or 3 training forms from each table. We then train a reinflection system on these forms and use the resulting system to predict the missing forms. We report accuracy on correctly predicted missing forms and on reconstructing the entire paradigm correctly. In our second experiment, we consider Unimorph tables which contain entries from a list of 10,000 most common word tokens compiled using a Wikipedia dump of the language as explained above. We take the forms in the top-10,000 list as given and train a model which is used to reconstruct the remaining forms in each table. We train an identical model as in the case $n > 1$ on tables with more than one given form. As in the first task, we evaluate with regard to accuracy for reconstructed forms and full tables. Results are presented in Tables 4 and 5, and Figure 4.

5 Discussion and Conclusions

Table 4 shows results for completing tables for common lexemes. Our system significantly out-

---

however, we did not have access to the exact splits into training and test data used by Malouf (2017). This may influence results.

[6] Finnish (fin), French (fre), Georgian (geo), German (ger), Latin (lat), Latvian (lav), Spanish (spa) and Turkish (tur).

	Our System			Baseline		
	1 form	2 forms	3 forms	1 form	2 forms	3 forms
FIN N	**18.87** ± 0.41 (0.00 ± 0.00)	**81.72** ± 0.78 (**16.50** ± 3.76)	**93.07** ± 0.71 (**54.80** ± 4.27)	6.07 ± 0.29 (0.00 ± 0.00)	46.64 ± 0.97 (0.00 ± 0.00)	65.60 ± 1.25 (0.80 ± 0.65)
FIN V	**19.88** ± 0.69 (0.00 ± 0.00)	**87.73** ± 0.36 (**59.20** ± 5.73)	**94.63** ± 0.41 (**75.50** ± 3.82)	12.35 ± 0.58 (0.00 ± 0.00)	63.56 ± 0.89 (0.90 ± 0.90)	81.49 ± 0.42 (17.20 ± 2.21)
FRE V	**15.66** ± 0.65 (0.00 ± 0.00)	**78.30** ± 0.66 (**23.50** ± 4.06)	**83.64** ± 0.72 (**35.60** ± 4.63)	11.46 ± 0.33 (0.00 ± 0.00)	61.01 ± 0.79 (0.40 ± 0.53)	74.07 ± 1.00 (7.60 ± 2.53)
GEO N	**28.66** ± 1.12 (0.00 ± 0.00)	**90.53** ± 0.48 (**53.20** ± 6.03)	**96.02** ± 0.48 (**84.80** ± 3.28)	21.14 ± 0.84 (0.00 ± 0.00)	78.91 ± 0.56 (23.50 ± 4.03)	90.61 ± 0.76 (51.30 ± 6.19)
GER N	39.46 ± 2.18 (2.50 ± 1.83)	**84.65** ± 2.00 (**61.30** ± 4.33)	**93.38** ± 0.86 (**78.30** ± 3.14)	40.25 ± 2.09 (4.40 ± 1.83)	72.26 ± 1.70 (32.70 ± 4.70)	86.49 ± 2.09 (57.30 ± 5.11)
GER V	**43.38** ± 0.68 (0.00 ± 0.00)	**92.73** ± 0.41 (**54.70** ± 3.75)	**95.83** ± 0.38 (**70.00** ± 4.99)	33.97 ± 1.13 (0.00 ± 0.00)	83.32 ± 0.48 (17.10 ± 3.27)	90.51 ± 0.62 (34.90 ± 3.92)
LAT N	16.89 ± 1.20 (0.00 ± 0.00)	**83.59** ± 1.20 (**49.50** ± 5.13)	**91.02** ± 0.76 (**68.70** ± 4.47)	**23.62** ± 1.19 (0.10 ± 0.32)	63.27 ± 1.25 (17.40 ± 4.09)	77.96 ± 1.34 (32.90 ± 5.60)
LAT V	**17.34** ± 0.37 (0.00 ± 0.00)	**83.01** ± 0.30 (**27.00** ± 2.65)	**89.66** ± 0.44 (**2.80** ± 1.35)	5.96 ± 0.21 (0.00 ± 0.00)	52.68 ± 0.43 (0.00 ± 0.00)	68.95 ± 0.47 (0.00 ± 0.00)
LAV N	**30.11** ± 1.27 (2.00 ± 1.37)	**85.41** ± 1.07 (**48.50** ± 4.00)	**94.83** ± 0.53 (**83.40** ± 4.37)	22.35 ± 0.88 (2.60 ± 1.39)	64.76 ± 1.28 (22.20 ± 3.17)	79.21 ± 1.13 (40.80 ± 4.41)
SPA V	**27.78** ± 0.69 (0.00 ± 0.00)	**87.44** ± 0.34 (**32.20** ± 5.71)	**94.81** ± 0.25 (**59.00** ± 6.71)	10.88 ± 0.35 (0.00 ± 0.00)	70.67 ± 0.27 (0.40 ± 0.53)	84.08 ± 0.38 (11.60 ± 2.91)
TUR N	**15.70** ± 0.44 (0.00 ± 0.00)	**88.90** ± 0.60 (**19.20** ± 4.89)	**92.07** ± 0.37 (**22.80** ± 4.22)	7.94 ± 0.39 (0.00 ± 0.00)	61.95 ± 0.72 (5.60 ± 2.95)	77.02 ± 0.49 (11.40 ± 3.82)

Table 5: Accuracy for filling in missing forms when n=1,2 or 3 forms are given in the inflection table (accuracy for complete paradigms in parentheses). We give the 0.99 confidence intervals as given by a one-sided t-test. Figures where one system significantly outperforms the other one are in boldface.

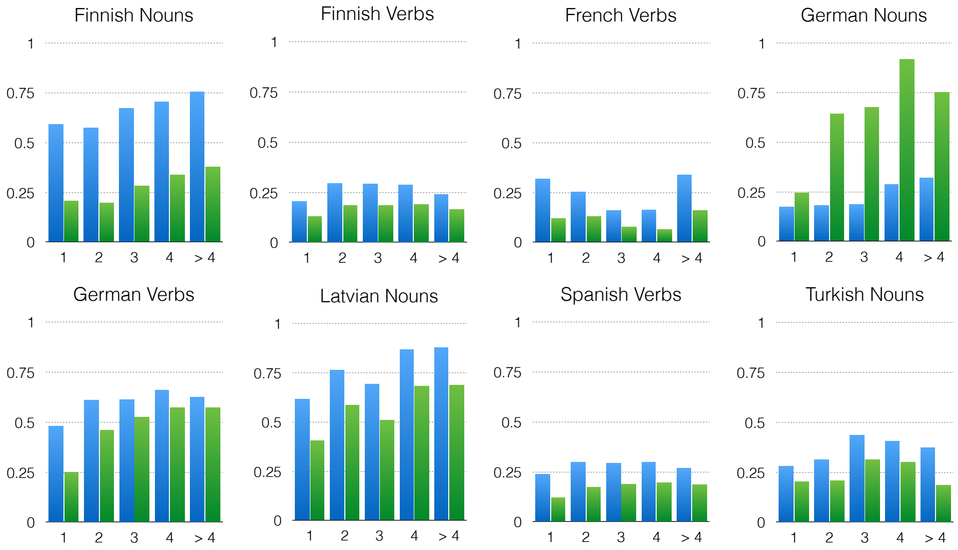

Figure 4: Detailed results for filling in missing forms when the 10,000 most frequent forms are given in the inflection tables. The blue bars (on the left) denote accuracy for our system and green bars (on the right) accuracy for the baseline system. The graphs show accuracy separately for tables where 1, 2, 3, 4, and $>$ 4 forms are given.

performs the baseline on all other datasets apart from German nouns. We believe that the reason for the German outlier is the high degree of syncretism in German noun tables. To see why syncretism is harmful, consider the German noun **Gräben**. Its paradigm consists of eight forms but four of those are identical: **Gräben**. Only this form is observed among the top 10,000 forms in the German Wikipedia. Following Section 2, this gives rise to 12 training examples where both the input and output form are **Gräben**. This strongly biases the system to copying input forms into the output. However, this will never give the correct output because, by design, missing forms cannot be **Gräben**.[7] This can be seen as a problem with our datasets rather than the model itself. Consequently, an important future work in addressing the PCFP from an acquisition perspective is to create realistic and accurate data sets that model learner exposure both in word types and frequencies to enable assessment of the true difficulty of the PCFP.

There is a notable transition from witnessing one form in each inflection table to witnessing two forms. With only two forms given, we already approach accuracies reported in earlier work (Malouf, 2016, 2017) that used almost complete tables to train—only 10% of the forms were missing. Additionally, our encoder-decoder model strongly outperforms that generator model designed for the same task with the same amount of training data on nearly all of our datasets.

Acknowledgements

The first author was supported by The Society of Swedish Literature in Finland (SLS). NVIDIA Corp. donated the Titan Xp GPU used for this research.

---

[7] If the same word form occurs in multiple slots, all of them are considered known.

References

Farrell Ackerman, James P. Blevins, and Robert Malouf. 2009. Parts and wholes: Implicative patterns in inflectional paradigms. pages 54–82. Oxford University Press.

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 569–578, Gothenburg, Sweden. Association for Computational Linguistics.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 1024–1029, Denver, CO. Association for Computational Linguistics.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

James P. Blevins and Juliette Blevins, editors. 2009. *Analogy in Grammar*. Oxford University Press.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, Brussels, Belgium. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017a. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Ryan Cotterell, Ekaterina Vylomova, Huda Khayrallah, Christo Kirov, and David Yarowsky. 2017b. Paradigm completion for derivational morphology. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 714–720, Copenhagen, Denmark. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California. Association for Computational Linguistics.

Mans Hulden. 2014. Generalizing inflection tables into paradigms with finite state operations. In *Proceedings of the 2014 Joint Meeting of SIGMORPHON and SIGFSM*, pages 29–36. Association for Computational Linguistics.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. Neural multi-source morphological reinflection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 514–524, Valencia, Spain. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 555–560, Berlin, Germany. Association for Computational Linguistics.

Fred Karlsson. 2008. *Finnish: An essential grammar*. Routledge.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Graldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian J. Mielke, Arya McCarthy, Sandra Kb̈ler, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. Unimorph 2.0: Universal morphology. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017*

Shared Task: Universal Morphological Reinflection, pages 49–57, Vancouver. Association for Computational Linguistics.

Robert Malouf. 2016. Generating morphological paradigms with a recurrent neural network. *San Diego Linguistic Papers*.

Robert Malouf. 2017. Abstractive morphological learning with a recurrent neural network. *Morphology*, 27(4):431–458.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Miikka Silfverberg, Ling Liu, and Mans Hulden. 2018. A computational model for the linguistic notion of morphological paradigm. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1615–1626. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Generating Natural Language Adversarial Examples

Moustafa Alzantot[1]*, Yash Sharma[2]*, Ahmed Elgohary[3],
Bo-Jhang Ho[1], Mani B. Srivastava[1], Kai-Wei Chang[1]

[1]Department of Computer Science, University of California, Los Angeles (UCLA)
{malzantot, bojhang, mbs, kwchang}@ucla.edu
[2]Cooper Union sharma2@cooper.edu
[3]Computer Science Department, University of Maryland elgohary@cs.umd.edu

Abstract

Deep neural networks (DNNs) are vulnerable to adversarial examples, perturbations to correctly classified examples which can cause the model to misclassify. In the image domain, these perturbations are often virtually indistinguishable to human perception, causing humans and state-of-the-art models to disagree. However, in the natural language domain, small perturbations are clearly perceptible, and the replacement of a single word can drastically alter the semantics of the document. Given these challenges, we use a black-box population-based optimization algorithm to generate semantically and syntactically similar adversarial examples that fool well-trained sentiment analysis and textual entailment models with success rates of 97% and 70%, respectively. We additionally demonstrate that 92.3% of the successful sentiment analysis adversarial examples are classified to their original label by 20 human annotators, and that the examples are perceptibly quite similar. Finally, we discuss an attempt to use adversarial training as a defense, but fail to yield improvement, demonstrating the strength and diversity of our adversarial examples. We hope our findings encourage researchers to pursue improving the robustness of DNNs in the natural language domain.

1 Introduction

Recent research has found that deep neural networks (DNNs) are vulnerable to *adversarial examples* (Goodfellow et al., 2015; Szegedy et al., 2014). The existence of adversarial examples has been shown in image classification (Szegedy et al., 2014) and speech recognition (Carlini and Wagner, 2018). In this work, we demonstrate that adversarial examples can be constructed in the context of natural language. Using a black-box population-based optimization algorithm, we successfully generate both semantically and syntactically similar adversarial examples against models trained on both the IMDB (Maas et al., 2011) sentiment analysis task and the Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) textual entailment task. In addition, we validate that the examples are both correctly classified by human evaluators and similar to the original via a human study. Finally, we attempt to defend against said adversarial attack using adversarial training, but fail to yield any robustness, demonstrating the strength and diversity of the generated adversarial examples.

Our results show that by minimizing the semantic and syntactic dissimilarity, an attacker can perturb examples such that humans correctly classify, but high-performing models misclassify. We are open-sourcing our attack[1] to encourage research in training DNNs robust to adversarial attacks in the natural language domain.

2 Natural Language Adversarial Examples

Adversarial examples have been explored primarily in the image recognition domain. Examples have been generated through solving an optimization problem, attempting to induce misclassification while minimizing the perceptual distortion (Szegedy et al., 2014; Carlini and Wagner, 2017; Chen et al., 2018; Sharma and Chen, 2018). Due to the computational cost of such approaches, fast methods were introduced which, either in one-step or iteratively, shift all pixels simultaneously until a distortion constraint is reached (Goodfellow et al., 2015; Kurakin et al., 2017; Madry et al., 2018). Nearly all popular methods are gradient-based.

---

* Moustafa Alzantot and Yash Sharma contribute equally to this work.

[1] https://github.com/nesl/nlp_adversarial_examples

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2890–2896
Brussels, Belgium, October 31 - November 4, 2018.

Such methods, however, rely on the fact that adding small perturbations to many pixels in the image will not have a noticeable effect on a human viewer. This approach obviously does not transfer to the natural language domain, as all changes are perceptible. Furthermore, unlike continuous image pixel values, words in a sentence are discrete tokens. Therefore, it is not possible to compute the gradient of the network loss function with respect to the input words. A straightforward workaround is to project input sentences into a continuous space (e.g. word embeddings) and consider this as the model input. However, this approach also fails because it still assumes that replacing *every* word with words nearby in the embedding space will not be noticeable. Replacing words without accounting for syntactic coherence will certainly lead to improperly constructed sentences which will look odd to the reader.

Relative to the image domain, little work has been pursued for generating natural language adversarial examples. Given the difficulty in generating semantics-preserving perturbations, distracting sentences have been added to the input document in order to induce misclassification (Jia and Liang, 2017). In our work, we attempt to generate semantically and syntactically similar adversarial examples, via word replacements, resolving the aforementioned issues. Minimizing the number of word replacements necessary to induce misclassification has been studied in previous work (Papernot et al., 2016), however without consideration given to semantics or syntactics, yielding incoherent generated examples. In recent work, there have been a few attempts at generating adversarial examples for language tasks by using back-translation (Iyyer et al., 2018), exploiting machine-generated rules (Ribeiro et al., 2018), and searching in underlying semantic space (Zhao et al., 2018). In addition, while preparing our submission, we became aware of recent work which target a similar contribution (Kuleshov et al., 2018; Ebrahimi et al., 2018). We treat these contributions as parallel work.

3 Attack Design

3.1 Threat model

We assume the attacker has black-box access to the target model; the attacker is not aware of the model architecture, parameters, or training data, and is only capable of querying the target model with supplied inputs and obtaining the output predictions and their confidence scores. This setting has been extensively studied in the image domain (Papernot et al., 2017; Chen et al., 2017a; Alzantot et al., 2018), but has yet to be explored in the context of natural language.

3.2 Algorithm

To avoid the limitations of gradient-based attack methods, we design an algorithm for constructing adversarial examples with the following goals in mind. We aim to minimize the number of modified words between the original and adversarial examples, but only perform modifications which retain semantic similarity with the original and syntactic coherence. To achieve these goals, instead of relying on gradient-based optimization, we developed an attack algorithm that exploits population-based gradient-free optimization via genetic algorithms.

An added benefit of using gradient-free optimization is enabling use in the black-box case; gradient-reliant algorithms are inapplicable in this case, as they are dependent on the model being differentiable and the internals being accessible (Papernot et al., 2016; Ebrahimi et al., 2018).

Genetic algorithms are inspired by the process of natural selection, iteratively evolving a population of candidate solutions towards better solutions. The population of each iteration is a called a *generation*. In each generation, the quality of population members is evaluated using a *fitness* function. “Fitter” solutions are more likely to be selected for breeding the next generation. The next generation is generated through a combination of *crossover* and *mutation*. Crossover is the process of taking more than one parent solution and producing a child solution from them; it is analogous to reproduction and biological crossover. Mutation is done in order to increase the diversity of population members and provide better exploration of the search space. Genetic algorithms are known to perform well in solving combinatorial optimization problems (Anderson and Ferris, 1994; Mühlenbein, 1989), and due to employing a population of candidate solutions, these algorithms can find successful adversarial examples with fewer modifications.

`Perturb` Subroutine: In order to explain our algorithm, we first introduce the subroutine `Perturb`. This subroutine accepts an input sentence $\mathbf{x}_{cur}$ which can be either a modified sentence or the same as $\mathbf{x}_{orig}$. It randomly selects a word w

in the sentence $\mathbf{x}_{cur}$ and then selects a suitable replacement word that has similar semantic meaning, fits within the surrounding context, and increases the $target$ label prediction score.

In order to select the best replacement word, `Perturb` applies the following steps:

- Computes the N nearest neighbors of the selected word according to the distance in the GloVe embedding space (Pennington et al., 2014). We used euclidean distance, as we did not see noticeable improvement using cosine. We filter out candidates with distance to the selected word greater than δ. We use the counter-fitting method presented in (Mrkšić et al., 2016) to post-process the adversary's GloVe vectors to ensure that the nearest neighbors are synonyms. The resulting embedding is independent of the embeddings used by victim models.
- Second, we use the Google 1 billion words language model (Chelba et al., 2013) to filter out words that do not fit within the context surrounding the word w in $\mathbf{x}_{cur}$. We do so by ranking the candidate words based on their language model scores when fit within the replacement context, and keeping only the top K words with the highest scores.
- From the remaining set of words, we pick the one that will maximize the target label prediction probability when it replaces the word w in $\mathbf{x}_{cur}$.
- Finally, the selected word is inserted in place of w, and `Perturb` returns the resulting sentence.

The selection of which word to replace in the input sentence is done by random sampling with probabilities proportional to the number of neighbors each word has within Euclidean distance δ in the counter-fitted embedding space, encouraging the solution set to be large enough for the algorithm to make appropriate modifications. We exclude common articles and prepositions (e.g. a, to) from being selected for replacement.

Optimization Procedure: The optimization algorithm can be seen in Algorithm 1. The algorithm starts by creating the initial generation $\mathcal{P}^0$ of size S by calling the `Perturb` subroutine S times to create a set of distinct modifications to the original sentence. Then, the fitness of each population member in the current generation is computed as the target label prediction probability, found by querying the victim model function f. If a population member's predicted label is equal to the target label, the optimization is complete. Otherwise, pairs of population members from the current generation are randomly sampled with probability proportional to their fitness values. A new *child* sentence is then synthesized from a pair of parent sentences by independently sampling from the two using a uniform distribution. Finally, the `Perturb` subroutine is applied to the resulting children.

Algorithm 1 Finding adversarial examples

```
for $i = 1, ..., S$ in population do
    $\mathcal{P}_i^0 \leftarrow \texttt{Perturb}(\mathbf{x}_{orig}, target)$
for $g = 1, 2...G$ generations do
    for $i = 1, ..., S$ in population do
        $F_i^{g-1} = f(\mathcal{P}_i^{g-1})_{target}$
    $\mathbf{x}_{adv} = \mathcal{P}^{g-1}_{\arg\max_j F_j^{g-1}}$
    if $\arg\max_c f(\mathbf{x}_{adv})_c == t$ then
        return $\mathbf{x}_{adv}$ $\triangleright$ {Found successful attack}
    else
        $\mathcal{P}_1^g = \{\mathbf{x}_{adv}\}$
        $p = Normalize(F^{g-1})$
        for $i = 2, ..., S$ in population do
            Sample $parent_1$ from $\mathcal{P}^{g-1}$ with probs $p$
            Sample $parent_2$ from $\mathcal{P}^{g-1}$ with probs $p$
            $child = Crossover(parent_1, parent_2)$
            $child_{mut} = \texttt{Perturb}(child, target)$
            $\mathcal{P}_i^g = \{child_{mut}\}$
```

4 Experiments

To evaluate our attack method, we trained models for the sentiment analysis and textual entailment classification tasks. For both models, each word in the input sentence is first projected into a fixed 300-dimensional vector space using GloVe (Pennington et al., 2014). Each of the models used are based on popular open-source benchmarks, and can be found in the following repositories[2][3]. Model descriptions are given below.

Sentiment Analysis: We trained a sentiment analysis model using the IMDB dataset of movie reviews (Maas et al., 2011). The IMDB dataset consists of 25,000 training examples and 25,000 test examples. The LSTM model is composed of 128 units, and the outputs across all time steps are

---

[2] `https://github.com/keras-team/keras/blob/master/examples/imdb_lstm.py`

[3] `https://github.com/Smerity/keras_snli/blob/master/snli_rnn.py`

Original Text Prediction = **Negative**. (Confidence = 78.0%)
*This movie had **terrible** acting, **terrible** plot, and **terrible** choice of actors. (Leslie Nielsen ...come on!!!) the one part I **considered** slightly funny was the battling FBI/CIA agents, but because the audience was mainly **kids** they didn't understand that theme.*

Adversarial Text Prediction = **Positive**. (Confidence = 59.8%)
*This movie had **horrific** acting, **horrific** plot, and **horrifying** choice of actors. (Leslie Nielsen ...come on!!!) the one part I **regarded** slightly funny was the battling FBI/CIA agents, but because the audience was mainly **youngsters** they didn't understand that theme.*

Table 1: Example of attack results for the sentiment analysis task. Modified words are highlighted in green and red for the original and adversarial texts, respectively.

Original Text Prediction: **Entailment** (Confidence = 86%)
Premise: *A runner wearing purple strives for the finish line.* **Hypothesis:** *A **runner** wants to head for the finish line.*

Adversarial Text Prediction: **Contradiction** (Confidence = 43%)
Premise: *A runner wearing purple strives for the finish line.* **Hypothesis:** *A **racer** wants to head for the finish line.*

Table 2: Example of attack results for the textual entailment task. Modified words are highlighted in green and red for the original and adversarial texts, respectively.

	Sentiment Analysis		Textual Entailment	
	% success	% modified	% success	% modified
`Perturb` baseline	52%	19%	–	–
Genetic attack	97%	14.7%	70%	23%

Table 3: Comparison between the attack success rate and mean percentage of modifications required by the genetic attack and perturb baseline for the two tasks.

averaged and fed to the output layer. The test accuracy of the model is 90%, which is relatively close to the state-of-the-art results on this dataset.

Textual Entailment: We trained a textual entailment model using the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015). The model passes the input through a ReLU "translation" layer (Bowman et al., 2015), which encodes the premise and hypothesis sentences by performing a summation over the word embeddings, concatenates the two sentence embeddings, and finally passes the output through 3 600-dimensional ReLU layers before feeding it to a 3-way softmax. The model predicts whether the premise sentence entails, contradicts or is neutral to the hypothesis sentence. The test accuracy of the model is 83% which is also relatively close to the state-of-the-art (Chen et al., 2017b).

4.1 Attack Evaluation Results

We randomly sampled 1000, and 500 *correctly classified* examples from the test sets of the two tasks to evaluate our algorithm. Correctly classified examples were chosen to limit the accuracy levels of the victim models from confounding our results. For the sentiment analysis task, the attacker aims to divert the prediction result from positive to negative, and vice versa. For the textual entailment task, the attacker is only allowed to modify the hypothesis, and aims to divert the prediction result from 'entailment' to 'contradiction', and vice versa. We limit the attacker to maximum $G = 20$ iterations, and fix the hyperparameter values to $S = 60$, $N = 8$, $K = 4$, and $\delta = 0.5$. We also fixed the maximum percentage of allowed changes to the document to be 20% and 25% for the two tasks, respectively. If increased, the success rate would increase but the mean quality would decrease. If the attack does not succeed within the iterations limit or exceeds the specified threshold, it is counted as a failure.

Sample outputs produced by our attack are shown in Tables 1 and 2. Additional outputs can be found in the supplementary material. Table 3 shows the attack success rate and mean percentage of modified words on each task. We compare to the `Perturb` baseline, which greedily applies the `Perturb` subroutine, to validate the use of population-based optimization. As can be seen from our results, we are able to achieve high success rate with a limited number of modifications on both tasks. In addition, the genetic algorithm significantly outperformed the `Perturb` baseline in both success rate and percentage of words modified, demonstrating the additional benefit yielded by using population-based optimization. Testing using a single TitanX GPU, for sentiment analysis and textual entailment, we measured average runtimes on success to be 43.5 and 5 seconds per example, respectively. The high success rate and reasonable runtimes demonstrate the practicality of our approach, even when scaling to long sentences, such as those found in the IMDB dataset.

Speaking of which, our success rate on textual entailment is lower due to the large disparity in sentence length. On average, hypothesis sentences in the SNLI corpus are 9 words long, which is very short compared to IMDB (229 words, limited to 100 for experiments). With sentences that short, applying successful perturbations becomes much harder, however we were still able to achieve a success rate of 70%. For the same reason, we didn't apply the `Perturb` baseline on the textual entailment task, as the `Perturb` baseline fails to achieve any success under the limits of the maximum allowed changes constraint.

4.2 User study

We performed a user study on the sentiment analysis task with 20 volunteers to evaluate how perceptible our adversarial perturbations are. Note that the number of participating volunteers is significantly larger than used in previous studies (Jia and Liang, 2017; Ebrahimi et al., 2018). The user study was composed of two parts. First, we presented 100 adversarial examples to the participants and asked them to label the sentiment of the text (i.e., positive or negative.) 92.3% of the responses matched the original text sentiment, indicating that our modification did not significantly affect human judgment on the text sentiment. Second, we prepared 100 questions, each question includes the original example and the corresponding adversarial example in a pair. Participants were asked to judge the similarity of each pair on a scale from 1 (very similar) to 4 (very different). The average rating is 2.23 ± 0.25, which shows the perceived difference is also small.

4.3 Adversarial Training

The results demonstrated in section 4.1 raise the following question: How can we defend against these attacks? We performed a preliminary experiment to see if adversarial training (Madry et al., 2018), the only effective defense in the image domain, can be used to lower the attack success rate. We generated 1000 adversarial examples on the cleanly trained sentiment analysis model using the IMDB training set, appended them to the existing training set, and used the updated dataset to adversarially train a model from scratch. We found that adversarial training provided no additional robustness benefit in our experiments using the test set, despite the fact that the model achieves near 100% accuracy classifying adversarial examples included in the training set. These results demonstrate the diversity in the perturbations generated by our attack algorithm, and illustrates the difficulty in defending against adversarial attacks. We hope these results inspire further work in increasing the robustness of natural language models.

5 Conclusion

We demonstrate that despite the difficulties in generating imperceptible adversarial examples in the natural language domain, semantically and syntactically similar adversarial examples can be crafted using a black-box population-based optimization algorithm, yielding success on both the sentiment analysis and textual entailment tasks. Our human study validated that the generated examples were indeed adversarial and perceptibly quite similar. We hope our work encourages researchers to pursue improving the robustness of DNNs in the natural language domain.

Acknowledgement

This research was supported in part by the U.S. Army Research Laboratory and the UK Ministry of Defence under Agreement Number W911NF-16-3-0001, the National Science Foundation under award # CNS-1705135, OAC-1640813, and IIS-1760523, and the NIH Center of Excellence for Mobile Sensor Data-to-Knowledge (MD2K) under award 1-U54EB020404-01. Ahmed Elgohary is funded by an IBM PhD Fellowship. Any findings in this material are those of the author(s) and do not reflect the views of any of the above funding agencies. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, and Mani Srivastava. 2018. Genattack: Practical black-box attacks with gradient-free optimization. *arXiv preprint arXiv:1805.11090*.

Edward J Anderson and Michael C Ferris. 1994. Genetic algorithms for combinatorial optimization: the assemble line balancing problem. *ORSA Journal on Computing*, 6.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing*.

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*.

Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *Advances in neural information processing systems: Deep Learning and Security Workshop*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 2018. EAD: Elastic-net attacks to deep neural networks via adversarial examples. In *AAAI Conference on Artificial Intelligence*.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017a. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *ACM Workshop on Artificial Intelligence and Security*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Enhanced lstm for natural language inference. In *Association for Computational Linguistics*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Association for Computational Linguistics*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *North American Association for Computational Linguistics*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing*.

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems. *arXiv preprint arXiv:1602.02697*.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. In *International Conference on Learning Representations*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Association for Computational Linguistics: Human Language Technologies*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *North American Chapter of the Association for Computational Linguistics*.

Heinz Mühlenbein. 1989. Parallel genetic algorithms, population genetics and combinatorial optimization. In *Workshop on Parallel Processing: Logic, Organization, and Technology*.

N. Papernot, P. McDaniel, A. Swami, and R. Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. *arXiv preprint arXiv:1604.08275*.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *ACM on Asia Conference on Computer and Communications Security*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Association for Computational Linguistics*.

Yash Sharma and Pin-Yu Chen. 2018. Attacking the madry defense model with l1-based adversarial examples. In *International Conference on Learning Representations: Workshops*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *International Conference on Learning Representations*.

Multi-Head Attention with Disagreement Regularization

Jian Li[1,2] **Zhaopeng Tu**[3*] **Baosong Yang**[4] **Michael R. Lyu**[1,2] **Tong Zhang**[3]
[1]Department of Computer Science and Engineering, The Chinese University of Hong Kong
[2]Shenzhen Research Institute, The Chinese University of Hong Kong
[1,2]`{jianli,lyu}@cse.cuhk.edu.hk`
[3]Tencent AI Lab
[3]`{zptu,bradymzhang}@tencent.com`
[4]University of Macau
[4]`nlp2ct.baosong@gmail.com`

Abstract

Multi-head attention is appealing for the ability to jointly attend to information from different representation subspaces at different positions. In this work, we introduce a disagreement regularization to explicitly encourage the diversity among multiple attention heads. Specifically, we propose three types of disagreement regularization, which respectively encourage the subspace, the attended positions, and the output representation associated with each attention head to be different from other heads. Experimental results on widely-used WMT14 English⇒German and WMT17 Chinese⇒English translation tasks demonstrate the effectiveness and universality of the proposed approach.

1 Introduction

Attention model is now a standard component of the deep learning networks, contributing to impressive results in neural machine translation (Bahdanau et al., 2015; Luong et al., 2015), image captioning (Xu et al., 2015), speech recognition (Chorowski et al., 2015), among many other applications. Recently, Vaswani et al. (2017) introduced a multi-head attention mechanism to capture different context with multiple individual attention functions.

One strong point of multi-head attention is the ability to jointly attend to information from *different* representation subspaces at *different* positions. However, there is no mechanism to guarantee that different attention heads indeed capture distinct features. In response to this problem, we introduce a disagreement regularization term to explicitly encourage the diversity among multiple attention heads. The disagreement regularization serves as an auxiliary objective to guide the training of the related attention component.

Specifically, we propose three types of disagreement regularization, which are applied to the three key components that refer to the calculation of feature vector using multi-head attention. Two regularization terms are respectively to maximize cosine distances of the input subspaces and output representations, while the last one is to disperse the positions attended by multiple heads with element-wise multiplication of the corresponding attention matrices. The three regularization terms can be either used individually or in combination.

We validate our approach on top of advanced TRANSFORMER model (Vaswani et al., 2017) for both English⇒German and Chinese⇒English translation tasks. Experimental results show that our approach consistently improves translation performance across language pairs. One encouraging finding is that TRANSFORMER-BASE with disagreement regularization achieves comparable performance with TRANSFORMER-BIG, while the training speed is nearly twice faster.

2 Background: Multi-Head Attention

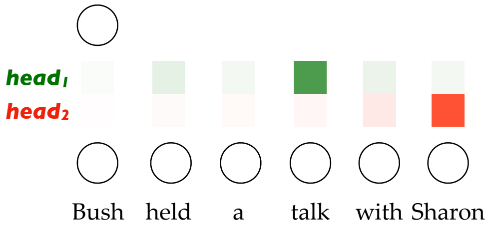

Figure 1: Illustration of the multi-head attention, which jointly attends to different representation subspaces (colored boxes) at different positions (darker color denotes higher attention probability).

Attention mechanism aims at modeling the strength of relevance between representation pairs,

---

[*] Zhaopeng Tu is the corresponding author of the paper. This work was mainly conducted when Jian Li and Baosong Yang were interning at Tencent AI Lab.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2897–2903
Brussels, Belgium, October 31 - November 4, 2018.

such that a representation is allowed to build a direct relation with another representation. Instead of performing a single attention function, Vaswani et al. (2017) found it is beneficial to capture different context with multiple individual attention functions. Figure 1 shows an example of a two-head attention model. For the query word "Bush", green and red head pay attention to different positions of "talk" and "Sharon" respectively.

Attention function softly maps a sequence of query $Q = \{Q_1, \dots, Q_N\}$ and a set of key-value pairs $\{K, V\} = \{(K_1, V_1), \dots, (K_M, V_M)\}$ to outputs. More specifically, multi-head attention model first transforms Q, K, and V into H subspaces, with different, learnable linear projections, namely:

$$Q^h, K^h, V^h = QW_h^Q, KW_h^K, VW_h^V,$$

where $\{Q^h, K^h, V^h\}$ are respective the query, key, and value representations of the h-th head. $\{W_h^Q, W_h^K, W_h^V\} \in \mathbb{R}^{d \times d_k}$ denote parameter matrices, d and d_k represent the dimensionality of the model and its subspace. Furthermore, H attention functions are applied in parallel to produce the output states $\{O^1, \dots, O^H\}$, among them:

$$O^h = A^h V^h \;\; with \;\; A^h = softmax(\frac{Q^h {K^h}^T}{\sqrt{d_k}}).$$

Here A^h is the attention distribution produced by the h-th attention head. Finally, the output states are concatenated to produce the final state.

3 Approach

Multi-head attention allows the model to jointly attend to information from *different* representation subspaces at *different* positions. To further guarantee the diversity, we enlarge the distances among multiple attention heads with disagreement regularization (Section 3.1). Specifically, we propose three types of disagreement regularization to encourage each head vector O^i to be different from other heads (Section 3.2).

3.1 Framework

In this work, we take the machine translation tasks as application. Given a source sentence $\mathbf{x}$ and its translation $\mathbf{y}$, a neural machine translation model is trained to maximize the conditional translation probability over a parallel training corpus.

We introduce an auxiliary regularization term in order to encourage the diversity among multiple attention heads. Formally, the training objective is revised as:

$$J(\theta) = \arg\max_{\theta} \{ \underbrace{L(\mathbf{y}|\mathbf{x};\theta)}_{likelihood} + \lambda * \underbrace{D(\mathbf{a}|\mathbf{x},\mathbf{y};\theta)}_{disagreement} \},$$

where $\mathbf{a}$ is the referred attention matrices, λ is a hyper-parameter and is empirically set to 1.0 in this paper. The auxiliary regularization term $D(\cdot)$ guides the related attention component to capture different features from the corresponding projected subspaces.

Note that the introduced regularization term works like $L1$ and $L2$ terms, which do not introduce any new parameters and only influence the training of the standard model parameters.

3.2 Disagreement Regularization

Three types of regularization term, which are applied to three parts of the original multi-head attention, are introduced in this section.

Disagreement on Subspaces (Sub.) This disagreement is designed to maximize the cosine distance between the projected values. Specifically, we first calculate the cosine similarity $\cos(\cdot)$ between the vector pair V^i and V^j in different value subspaces, through the dot product of the normalized vectors[1], which measures the cosine of the angle between V^i and V^j. Thus, the cosine distance is defined as negative similarity, i.e, $-\cos(\cdot)$. Our training objective is to enlarge the average cosine distance among all head pairs. The regularization term is formally expressed as:

$$D_{subpace} = -\frac{1}{H^2} \sum_{i=1}^{H} \sum_{j=1}^{H} \frac{V^i \cdot V^j}{\|V^i\| \|V^j\|}. \qquad (1)$$

Disagreement on Attended Positions (Pos.) Another strategy is to disperse the attended positions predicted by multiple heads. Inspired by the agreement regularization (Liang et al., 2006; Cheng et al., 2016) which encourages multiple alignments to be similar, in this work, we deploy a variant of the original term by introducing an alignment disagreement regularization. Formally, we employ the sum of element-wise multiplication of corresponding matrix cells[2], to measure the

---

[1] We did not employ the Euler Distance between vectors since we do not care the absolute value in each vector.

[2] We also used the squared element-wise subtraction of two matrices in our preliminary experiments, and found it underperforms its multiplication counterpart, which is consistent with the results in (Cheng et al., 2016).

similarity between two matrices A^i and A^j of two heads:

$$D_{position} = -\frac{1}{H^2}\sum_{i=1}^{H}\sum_{j=1}^{H}|A^i \odot A^j|. \quad (2)$$

Disagreement on Outputs (Out.) This disagreement directly applies regularization on the outputs of each attention head, by maximizing the difference among them. Similar to the *subspace* strategy, we employ negative cosine similarity to measure the distance:

$$D_{output} = -\frac{1}{H^2}\sum_{i=1}^{H}\sum_{j=1}^{H}\frac{O^i \cdot O^j}{\|O^i\|\|O^j\|}. \quad (3)$$

4 Related Work

The regularization on attended positions is inspired by agreement learning in prior works, which encourages alignments or hidden variables of multiple models to be similar. Liang et al. (2006) first assigned agreement terms for jointly training word alignment in phrase-based statistic machine translation (Koehn et al., 2003). The idea was further extended into other natural language processing tasks such as grammar induction (Liang et al., 2008). Levinboim et al. (2015) extended the agreement for general bidirectional sequence alignment models with model inevitability regularization. Cheng et al. (2016) further explored the agreement on modeling the source-target and target-source alignments in neural machine translation model. In contrast to the mentioned approaches which assigned agreement terms into loss function, we deploy an alignment disagreement regularization by maximizing the distance among multiple attention heads.

As standard multi-head attention model lacks effective control on the influence of different attention heads, Ahmed et al. (2017) used a weighted mechanism to combine them rather than simple concatenation. As an alternative approach to multi-head attention, Shen et al. (2018a) and Shen et al. (2018b) extended the single relevance score to multi-dimensional attention weights, demonstrating the effectiveness of modeling multiple features for attention networks. Our approach is complementary to theirs: our model encourages the diversity among multiple heads, while theirs enhance the power of each head.

#	**Regularization**			**Speed**	**BLEU**
	Sub.	*Pos.*	*Out.*		
1	×	×	×	1.21	24.13
2	✓	×	×	1.15	24.64
3	×	✓	×	1.14	24.42
4	×	×	✓	1.15	**24.78**
5	✓	×	✓	1.12	24.73
6	✓	✓	×	1.11	24.38
7	✓	✓	✓	1.05	24.60

Table 1: Effect of regularization terms, which are applied to the encoder self-attention only. "Speed" denotes the training speed (steps/second).

5 Experiments

5.1 Setup

To compare with the results reported by previous work (Gehring et al., 2017; Vaswani et al., 2017; Hassan et al., 2018), we conduct experiments on both WMT2017 Chinese⇒English (Zh⇒En) and WMT2014 English⇒German (En⇒De) translation tasks. The Zh⇒En corpus consists of 20M sentence pairs, and the En⇒De corpus consists of 4M sentence pairs. We follow previous work to select the validation and test sets. Byte-pair encoding (BPE) is employed to alleviate the Out-of-Vocabulary problem (Sennrich et al., 2016) with 32K merge operations for both language pairs. We use the case-sensitive 4-gram NIST BLEU score (Papineni et al., 2002) as evaluation metric, and *sign-test* (Collins et al., 2005) for statistical significance test.

We evaluate the proposed approaches on the advanced TRANSFORMER model (Vaswani et al., 2017), and implement on top of an open-source toolkit – THUMT (Zhang et al., 2017). We follow Vaswani et al. (2017) to set the configurations and have reproduced their reported results on the En⇒De task. All the evaluations are conducted on the test sets. We have tested both *Base* and *Big* models, which differ at hidden size (512 vs. 1024) and number of attention heads (8 vs. 16). We study model variations with *Base* model on the Zh⇒En task (Section 5.2 and 5.3), and evaluate overall performance with *Big* model on both Zh⇒En and En⇒De tasks (Section 5.4).

5.2 Effect of Regularization Terms

In this section, we evaluate the impact of different regularization terms on the Zh⇒En task us-

System	**Architecture**	**Zh⇒En**		**En⇒De**	
		Speed	BLEU	Speed	BLEU
Existing NMT systems					
(Wu et al., 2016)	GNMT	n/a	n/a	n/a	26.30
(Gehring et al., 2017)	CONVS2S	n/a	n/a	n/a	26.36
(Vaswani et al., 2017)	TRANSFORMER-BASE	n/a	n/a	n/a	27.3
	TRANSFORMER-BIG	n/a	n/a	n/a	28.4
(Hassan et al., 2018)	TRANSFORMER-BIG	n/a	24.2	n/a	n/a
Our NMT systems					
this work	TRANSFORMER-BASE	1.21	24.13	1.28	27.64
	+ Disagreement	1.06	$24.85^{\Uparrow}$	1.10	$28.51^{\Uparrow}$
	TRANSFORMER-BIG	0.58	24.56	0.61	28.58
	+ Disagreement	0.47	$25.08^{\Uparrow}$	0.51	$29.28^{\Uparrow}$

Table 3: Comparing with existing NMT systems on WMT17 Chinese⇒English and WMT14 English⇒German translation tasks. "$\Uparrow$" indicates that the model is significantly better than its baseline counterpart ($p < 0.01$).

Applying to			**Speed**	**BLEU**
Enc	*E-D*	*Dec*		
×	×	×	1.21	24.13
✓	×	×	1.15	24.78
✓	✓	×	1.10	24.67
✓	×	✓	1.11	24.69
✓	✓	✓	1.06	**24.85**

Table 2: Effect of regularization on different attention networks, i.e., encoder self-attention ("*Enc*"), encoder-decoder attention ("*E-D*"), and decoder self-attention ("*Dec*").

ing TRANSFORMER-BASE. For simplicity and efficiency, here we only apply regularizations on the encoder side. As shown in Table 1, all the models with the proposed disagreement regularizations (Rows 2-4) consistently outperform the vanilla TRANSFORMER (Row 1). Among them, the *Output* term performs best which is +0.65 BLEU score better than the baseline model, the *Position* term is less effective than the other two. In terms of training speed, we do not observe obvious decrease, which in turn demonstrates the advantage of our disagreement regularizations.

However, the combinations of different disagreement regularizations fail to further improve translation performance (Rows 5-7). One possible reason is that different regularization terms have overlapped guidance, and thus combining them does not introduce too much new information while makes training more difficult.

5.3 Effect on Different Attention Networks

The TRANSFORMER consists of three attention networks, including encoder self-attention, decoder self-attention, and encoder-decoder attention. In this experiment, we investigate how each attention network benefits from the disagreement regularization. As seen from Table 2, all models consistently improve upon the baseline model. When applying disagreement regularization to all three attention networks, we achieve the best performance, which is +0.72 BLEU score better than the baseline model. The training speed decreases by 12%, which is acceptable considering the performance improvement.

5.4 Main Results

Finally, we validate the proposed disagreement regularization on both WMT17 Chinese-to-English and WMT14 English-to-German translation tasks. Specifically, we adopt the *Output* disagreement regularization, which is applied to all three attention networks. The results are concluded in Table 3. We can see that our implementation of TRANSFORMER outperforms all existing NMT systems, and matches the results of TRANSFORMER reported in previous works. Incorporating disagreement regularization consistently improves translation performance for both base and big TRANSFORMER models across language pairs, demonstrating the effectiveness of the proposed approach. It is encouraging to see that TRANSFORMER-BASE with disagreement regu-

Regularization on	Disagreement on		
	Sub.	*Pos.*	*Out.*
n/a	0.882	0.007	0.881
Subspace	**0.999**	0.006	0.935
Position	0.882	**0.219**	0.882
Output	0.989	0.006	**0.997**

Table 4: Effect of different regularization terms on the three disagreement measurements. "n/a" denotes the baseline model without any regularization term. Larger value denotes more disagreement (at most 1.0).

Reg.	Layer					
	1	2	3	4	5	6
n/a	0.040	0.009	0.002	0.003	0.008	0.006
Sub.	0.039	0.009	0.001	0.003	0.006	0.005
Pos.	0.217	0.167	0.219	0.242	0.233	0.249
Out.	0.048	0.009	0.002	0.003	0.008	0.006

Table 5: Disagreement on attended positions with respect to the levels of the encoder layers.

larization achieves comparable performance with TRANSFORMER-BIG, while the training speed is nearly twice faster.

5.5 Quantitative Analysis of Regularization

In this section, we empirically investigate how the regularization terms affect the multi-head attention. To this end, we compare the disagreement scores on subspaces ("Sub."), attended positions ("Pos."), and outputs ("Out."). Since the scores are negative values, we list $\exp(D)$ for readability, which has a maximum value of 1.0. Table 4 lists the results of encoder-side multi-head attention on the Zh⇒En validation set. As seen, the disagreement score on the individual component indeed increases with the corresponding regularization term. For example, the disagreement of outputs increases to almost 1.0 by using the *Output* regularization, which means that the output vectors are almost perpendicular to each other as we measure the cosine distance as the disagreement.

One interesting finding is that attending to different positions may not be the essential strength of multi-head attention on the translation task. As seen, the disagreement score on the attended positions for the standard multi-head attention is only 0.007, which indicates that almost all the heads attend to the same positions. Table 5 shows the disagreement scores on attended positions across encoder layers. Except for the 1^{st} layer that attends to the input word embeddings, the disagreement scores on other layers (i.e. ranging from the 2^{nd} to 6^{th} layer) are very low, which confirms out above hypothesis.

Concerning the regularization terms, except that on position, the other two regularization terms (i.e. "Sub." and "Out.") do not increase the disagreement score on the attended positions. This can explain why positional regularization term does not work well with the other two terms, as shown in Table 1. This is also consistent with the finding in (Tu et al., 2016), which indicates that neural networks can model linguistic information in their own way. In contrast to attended positions, it seems that the multi-head attention prefer to encoding the differences among multiple heads in the learned representations.

6 Conclusion

In this work, we propose several disagreement regularizations to augment the multi-head attention model, which encourage the diversity among attention heads so that different head can learn distinct features. Experimental results across language pairs validate the effectiveness of the proposed approaches.

The models also suggest a wide range of potential advantages and extensions, from being able to improve the performance of multi-head attention in other tasks such as reading comprehension and language inference, to being able to combine with other techniques (Shaw et al., 2018; Shen et al., 2018a; Dou et al., 2018; Yang et al., 2018) to further improve performance.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (Project No. 61332010 and No. 61472338), the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14234416 of the General Research Fund), and Microsoft Research Asia (2018 Microsoft Research Asia Collaborative Research Award). We thank the anonymous reviewers for their insightful comments.

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2017. Weighted transformer network for machine translation. *arXiv preprint arXiv:1711.02132*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Yong Cheng, Shiqi Shen, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Agreement-based joint training for bidirectional attention-based neural machine translation. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based Models for Speech Recognition. In *Advances in neural information processing systems (NIPS)*.

M. Collins, P. Koehn, and I. Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting Deep Representations for Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Tomer Levinboim, Ashish Vaswani, and David Chiang. 2015. Model invertibility regularization: Sequence alignment with or without parallel data. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Percy Liang, Dan Klein, and Michael I Jordan. 2008. Agreement-based Learning. In *Advances in neural information processing systems (NIPS)*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. DiSAN: Directional Self-Attention Network for RNN/CNN-free Language Understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018b. Bi-directional Block Self-Attention for Fast and Memory-Efficient Sequence Modeling. In *International Conference on Learning Representations (ICLR)*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems (NIPS)*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling Localness for Self-Attention Networks. In

Conference on Empirical Methods in Natural Language Processing (EMNLP).

Jiacheng Zhang, Yanzhuo Ding, Shiqi Shen, Yong Cheng, Maosong Sun, Huanbo Luan, and Yang Liu. 2017. THUMT: An Open Source Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1706.06415*.

Deep Bayesian Active Learning for Natural Language Processing: Results of a Large-Scale Empirical Study

Aditya Siddhant
Carnegie Mellon University
`asiddhan@cs.cmu.edu`

Zachary C. Lipton
Carnegie Mellon University
`zlipton@cmu.edu`

Abstract

Several recent papers investigate Active Learning (AL) for mitigating the data-dependence of deep learning for natural language processing. However, the applicability of AL to real-world problems remains an open question. While in supervised learning, practitioners can try many different methods, evaluating each against a validation set before selecting a model, AL affords no such luxury. Over the course of one AL run, an agent annotates its dataset exhausting its labeling budget. Thus, given a new task, an active learner has no opportunity to compare models and acquisition functions. This paper provides a large-scale empirical study of deep active learning, addressing multiple tasks and, for each, multiple datasets, multiple models, and a full suite of acquisition functions. We find that across all settings, *Bayesian active learning by disagreement*, using uncertainty estimates provided either by Dropout or Bayes-by-Backprop significantly improves over i.i.d. baselines and usually outperforms classic uncertainty sampling.

1 Introduction

While over the past several years, deep learning has pushed the state of the art on numerous tasks, its extreme data-dependence presents a formidable obstacle under restricted annotation budgets. Active Learning (AL) presents one promising approach to reduce deep learning's data requirements (Cohn et al., 1996). Strategically selecting points to annotate over alternating rounds of labeling and learning, an active learner is hoped to outperform budget-matched i.i.d. labeling. Typical *acquisition functions* select examples for which the current predictor is most uncertain. However, how precisely to quantify uncertainty, especially for neural networks, remains an open question.

Classical approaches interpret either the entropy or the negative argmax of the predictive (e.g. softmax) distribution as the model's uncertainty, yielding the *maximum entropy* and *least confidence* heuristics, respectively. These approaches account for aleatoric but not epistemic uncertainty (Kendall and Gal, 2017). Several recent Bayesian formulations of deep learning provide alternative techniques for extracting uncertainty estimates from deep networks, including a *dropout*-based approach (Gal and Ghahramani, 2016b), previously employed in Deep Active Learning (DAL) for image classification (Gal et al., 2017) and named entity recognition (Shen et al., 2018), and Bayes-by-Backprop (Blundell et al., 2015). To our knowledge, our paper is the first to apply Bayes-by-Backprop in the context of DAL.

While the results in recent papers hint at DAL's potential, its suitability in practice has yet to be proven. That's because papers often address just a single task, just a single model, and sometimes just one or two datasets. However, it's not enough to look back retrospectively after a final round of experiments and declare that one acquisition function outperforms an i.i.d. baseline. To apply DAL in practice, we must be confident that the technique will work correctly—*the first time*—on a dataset that we have never seen before. Otherwise, we might exhaust the annotation budget while performing worse than an i.i.d. baseline. Once we've exhausted our resources for labeling, there's no going back. Moreover, many DAL papers suffer from implicit target leaks. The architectures and hyper-parameters are often tuned using the full dataset, before concealing the labels and *simulating* AL.

In this paper, we present a large-scale study[1], comparing various acquisition functions across multiple tasks: Sentiment Classification (SC),

---

[1] Code for all of our models and for running active learning experiments can be found at `https://github.com/asiddhant/Active-NLP`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2904–2909
Brussels, Belgium, October 31 - November 4, 2018.

Named Entity Recognition (NER), and Semantic Role Labeling (SRL). For each task we consider, with multiple datasets, multiple models, and multiple acquisition functions. Moreover, in all experiments, we set hyper-parameters on warm-start data, allowing for a more honest assessment. This paper does not seek to champion any one approach but instead to ask, *is there any single method that we can reliably expect to work out-of-the-box on a new problem?*

To our surprise, we find that BALD (Houlsby et al., 2011), which measures uncertainty by the frequency over multiple Monte Carlo draws from a stochastic model with which the drawn models disagree with the plurality, proved effective across all combinations of task, dataset, and model. Moreover both variants of the approach, drawing samples according to the dropout method (Gal et al., 2017) and from a Bayes-by-Backprop network (Blundell et al., 2015), performed similarly well across most tasks, datasets, and models.

Related Work Only a few papers have addressed DAL for NLP, notably Shen et al. (2018) for NER and Zhang et al. (2017) who address text classification, proposing to select examples according to the expected magnitude of updates to word embeddings. In this paper, we do not consider the latter heuristic because we address sequence tagging tasks, where the difficulty of marginalizing over all possible labels blows up exponentially with sequence length. While both previous papers do conduct experiments on multiple datasets (2 and 3, respectively) they each consider just one task and just one model.

Gal et al. (2017) apply the dropout-based uncertainty estimates due to (Gal and Ghahramani, 2016a) together with the BALD framework due to (Houlsby et al., 2011) for image classification with convolutional neural networks. They obtain significant improvement over classic uncertainty-based acquisition functions on the MNIST dataset and for diagnosing skin cancer from lesion images (ISIC2016 task). Our work builds on theirs, both by offering a large-scale evaluation of BALD for NLP tasks and models, and by exploring BALD with another method for estimating uncertainty: the uncertainty of the weights as modeled by a Bayes-by-Backprop network.

2 Bayesian Deep Learning

While space constraints preclude an extensive discussion of the various Bayesian formulations of neural nets, we briefly summarize the methods compared in this paper, pointing out various design decisions that are important for reproducing our results.

Monte Carlo Dropout According to (Gal and Ghahramani, 2016b), the dropout regularization techniques for neural networks can be interpreted as a Bayesian approximation to Gaussian processes (Rasmussen, 2004). Here, unlike standard uses of dropout, we apply it at prediction time. Uncertainty estimates are produced by comparing the output of a trained neural network using T different stochastic passes through the neural network. The extension to CNNs is straightforward. To apply dropout to RNNs, we follow the approach due to (Gal and Ghahramani, 2016c), who extended their variational analysis to RNNs, arguing that dropout ought to be applied to the recurrent layers (and not just the synchronous connections, per previous standard practice (Zaremba et al., 2014)) by applying identical dropout masks at each sequence step.

Bayes by Backprop In this approach due to Blundell et al. (2015), instead of maintaining a point estimate for each weight, we maintain a probability distribution over the weights. A standard L-layer MLP model $P(y|x, w)$ is parametrized by weights $w = \{W_l, b_l\}_{l=1}^{L} \in \mathbb{R}^d$. Then, $\hat{y} = \phi(W_L \cdot ... \cdot +\phi(W_1 \cdot x + b_1) + .. + b_L)$ where ϕ is an activation function such as tanh or ReLU. Bayes-by-Backprop represents imposes a prior over the weights, $p(w)$ and seeks to learn the posterior distribution $p(w|D)$ given training data $D = \{x^i, y^i\}_{i=1}^{N}$. To deal with intractability, Bayes-by-Backprop approximates $p(w|D)$ by a variational distribution $q(w|\theta)$, typically choosing q to be a Gaussian with diagonal covariance and each weight sampled from $\mathcal{N}(\mu_i, \sigma_i^2)$. To enforce non-negativity, the σ_i are further parametrized via the *softplus* function $\sigma_i = log(1 + exp(\rho_i))$ giving variational parameters $\theta = \{\mu_i, \rho_i\}_{i=1}^{d}$.

Our objective in optimizing the variational parameters is to minimize the KL divergence between $q(\theta)$ and $p(w|D)$. Some simplification of the objective gives $\mathcal{L}(D, \theta) = \sum_{j=1}^{N} \left[\log q(w^j|\theta) - \log p(w^j) - \log p(D|w^j)\right]$, where w^j denotes the j-th Monte Carlo sample drawn from $q(w|\theta)$ (we use $N = 1$). In Bayes-by-Backprop, the parameters are optimized by stochastic gradient descent, using the re-parameterization trick popularized by Kingma and Welling (2014). Extending Bayes-by-Backprop to CNNs and RNNs is straightforward with the latter requiring minor adjustments for truncated back-propagation through time (Fortunato et al., 2017). Uncertainty estimates calculated via Bayes-by-Backprop have been shown to be useful for efficient exploration in reinforcement learning (Lipton et al., 2018).

3 Experimental Setup

3.1 Acquisition functions

In this work, we consider only uncertainty-based acquisition. In particular, we consider least confidence (LC) for classification and maximum length-normalized log probability (MNLP) for sequence labeling tasks (Shen et al., 2018). LC chooses that example with for which the prediction has lowest predicted probability. MNLP extends this to sequences, selecting by log probability normalized by length, removing the bias for the model to preferentially select longer sequences.

BALD We briefly articulate the details of the Bayesian Active Learning by Disagreement (BALD) approach due to Houlsby et al. (2011), upon which both our Bayesian approaches are based. We denote Monte Carlo Dropout Disagreement by DO-BALD and its Bayes-by-Backprop counterpart as BB-BALD. BALD originally selects samples that maximise the information gained about the model parameters. This boils down to choosing data points which each stochastic forward pass through the model would have the highest probability assigned to a different class (Gal et al., 2017). Our measure of uncertainty is the fraction of models, across MC samples from the network, that that disagree with most popular choice. This can be mathematically represented as

$$\arg\max_j \left(1 - \frac{count(mode(\tilde{y}_j^{(1)}, ..., \tilde{y}_j^{(T)}))}{T}\right)$$

Here $\tilde{y}_j^{(t)}$ represents the prediction (argmax) applied to the tth forward pass on jth sample $\tilde{y}_j^{(t)} = \texttt{argmax}(\hat{y}_j^{(t)})$. We resolve ties by choosing the least confident predictions as determined by the mean probability assigned to the consensus class. For sequences, we look at agreement on the entire sequence tag, noting that this may exhibit a bias to preferentially sample longer sentences. Because we measure the budget at each round in words (not sentences), while this constitutes a bias, it does not constitute an unfair advantage. Moreover, we note that all AL necessarily consists of biased sampling.

3.2 Training details

The active learning process begins with a random acquisition of 2% *warmstart* samples from the dataset. We train an initial model on this data. Then based on this model's uncertainty estimates, we apply our chosen acquisition function to sample an additional 2% of examples and train a new model based on this data In each round, we train from scratch to avoid badly overfitting the data collected in earlier rounds per observations by Hu et al. (2018). We continue with alternating rounds of labeling and training until we have annotated 50% of the dataset. For classification tasks, the we measure the budget in sentences while for sequence labeling, we measure the budget by the number of words because the annotator must provide one tag per word.

In each iteration, we train each model to convergence, decided based on early stopping with a patience of 1 epoch, or 25 epochs (whichever comes earlier). For datasets with fixed validation sets such as Conll 2003, instead of using the entire validation set for early stopping, we use the percentage of validation data equivalent to that in our current training pool. Our motivation here is to keep the simulation realistic. Essentially, we assume that given a large annotation budget, one will collect both a larger training set and a larger validation set. As a motivating example, it seems unreasonable that a practitioner might have only 500 training examples but 10,000 examples available for early stopping. Our reported results are averaged over 3 runs with different warmstart samples.

3.3 Sentence Classification

We use two datasets for simulation: one question classification dataset TrecQA (Roth et al., 2002) and one sentiment analysis dataset (Pang and Lee, 2005) and two architectures for training: CNNs and BiLSTMs. For implementation of the CNNs on both these datasets, we follow the setup of Kim (2014) and for BiLSTMs, we use a single layer model with 300 hidden units for both datasets. We

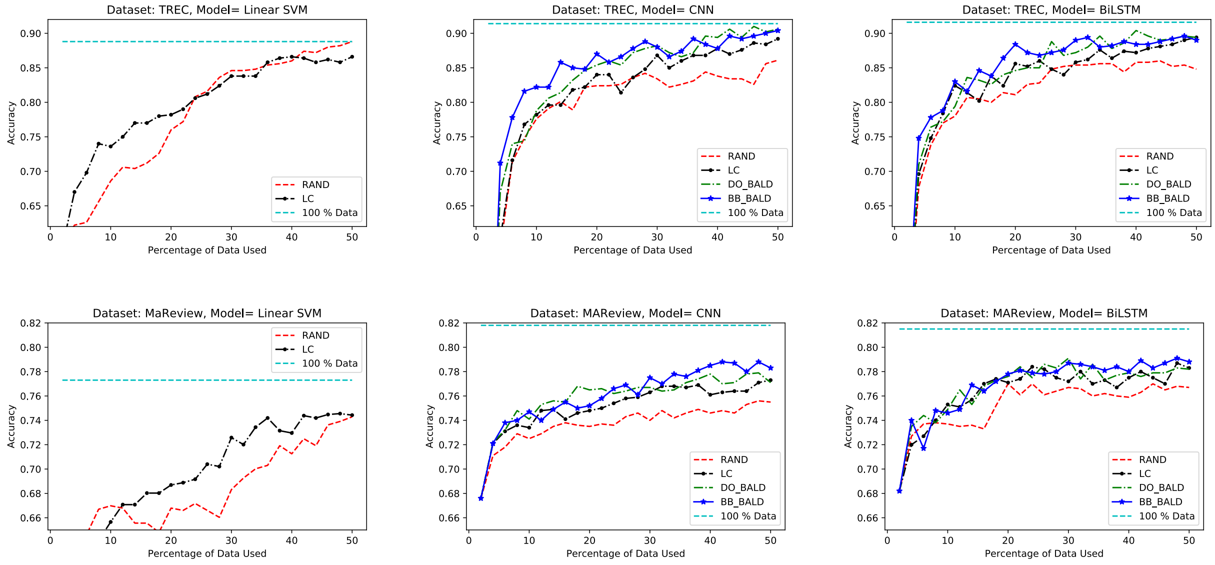
Dataset: TREC, Model= Linear SVM
Dataset: TREC, Model= CNN
Dataset: TREC, Model= BiLSTM
Dataset: MaReview, Model= Linear SVM
Dataset: MAReview, Model= CNN
Dataset: MAReview, Model= BiLSTM

Figure 1: Performance of various models and acquisition functions for two SC datasets

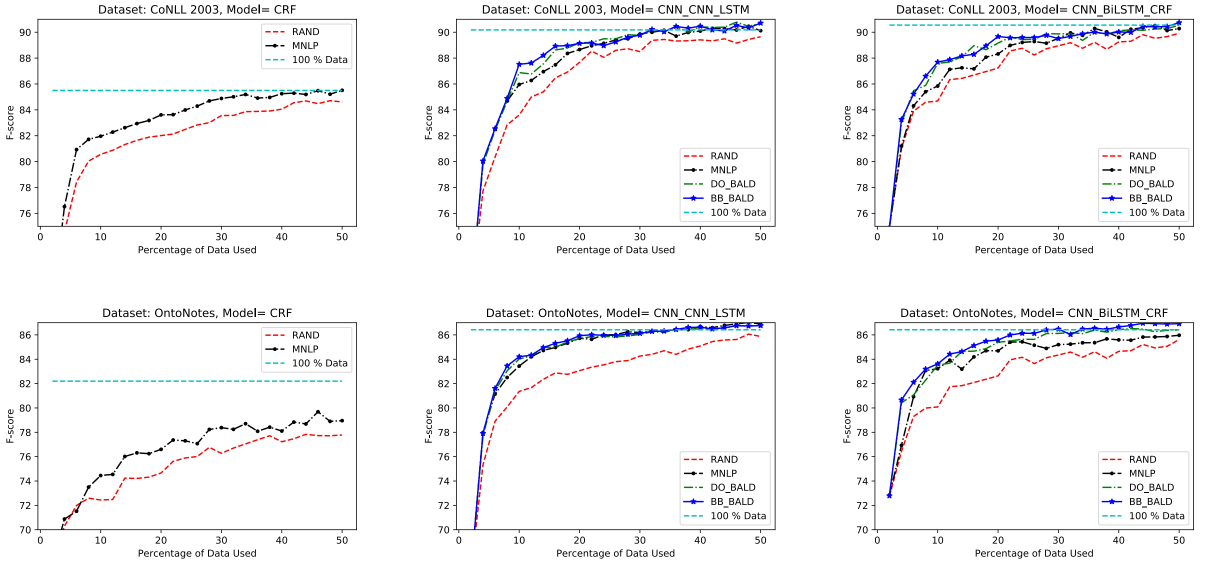
Dataset: CoNLL 2003, Model= CRF
Dataset: CoNLL 2003, Model= CNN_CNN_LSTM
Dataset: CoNLL 2003, Model= CNN_BiLSTM_CRF
Dataset: OntoNotes, Model= CRF
Dataset: OntoNotes, Model= CNN_CNN_LSTM
Dataset: OntoNotes, Model= CNN_BiLSTM_CRF

Figure 2: Performance of various models and acquisition functions for two NER datasets

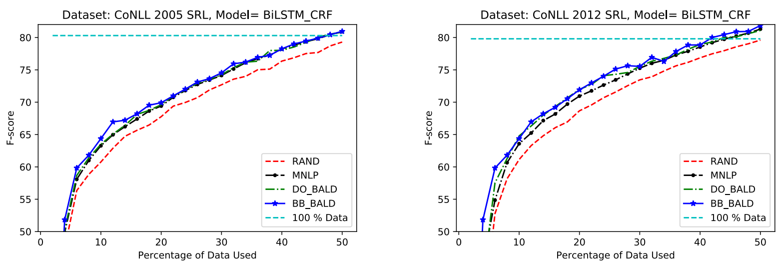
Dataset: CoNLL 2005 SRL, Model= BiLSTM_CRF
Dataset: CoNLL 2012 SRL, Model= BiLSTM_CRF

Figure 3: Performance of different acquisition functions on SRL task for two datasets

use 300-dimensional glove embeddings (Pennington et al., 2014) pretrained on 6B tokens for all 4 settings, a dropout rate of 0.5, and the Adam optimizer (Kinga and Adam, 2015) with initial learning rate 1e-3. We use a batch size set to be either 50 or the number required for at least 10 updates whichever is lower. This is done to ensure that when the training pool is small, the batch size is not too large and models get sufficient number of updates in an epoch. We also train a Unigram + Bigram + Linear SVM model with LC acquisition as a shallow AL baseline.

3.4 Named Entity Recognition

Again, we use two datasets: CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) and OntoNotes 5.0. The two architectures used for training are CNN-BiLSTM-CRF (CNN for character-level encoding, BiLSTM for word-level encoding, and CRF for decoding) (Ma and Hovy, 2016) and CNN-CNN-LSTM (CNN for character-level encoding, CNN for word-level encoding, and LSTM for decoding) (Shen et al., 2018). We follow the exact experimental settings of these papers except that batch size is 16 for CoNLL and 80 for OntoNotes (minimum 10 updates heuristic is followed here too).

We note that our NER models consist of multiple modular components, and that we only train a subset of those units in a Bayesian fashion. In both DO-BALD and BB-BALD, we apply dropout/stochastic weights on the word-level layers, but not on the character-level encoders or decoding layers. For example, with DO-BALD, we apply recurrent dropout in the BiLSTM word-level component of CNN-BiLSTM-CRF and we apply normal dropout in the word-level (middle) CNN layer of the CNN-CNN-LSTM. For NER, as a shallow AL baseline, we have a linear chain CRF model with MNLP acquisition.

3.5 Semantic Role Labeling

We consider two datasets: CoNLL 2005 (Carreras and Màrquez, 2005) and CoNLL 2012, focusing only on an LSTM-based model this time. Our model resembles He et al. (2017), but instead of using contained A* decoding, we use a CRF decoder, noting that while this causes a 2% drop in performance (at 100% annotation), our goal is to compare acquisition functions, not achieve record-setting performance. We follow the experimental setup of the paper but use a higher dropout rate of 0.25, adjusting the batch size according to the minimum update heuristic.

3.6 Results

We plot the performance for various annotation budgets for all combinations of dataset, model, and acquisition function, for the SC, NER, and SRL tasks in Figures 1, 2, and 3, respectively. In all cases, the active learning methods perform better than random i.i.d. baseline. We note that across the board, DAL methods show significant improvement over shallow baselines. The Bayesian acquisition functions, DO-BALD and BB-BALD consistently outperform classic uncertainly sampling, although in a few cases including the setting considered by Shen et al. (2018), the improvement is only marginal. This finding underscores the importance of examining proposed AL methods on a broad set of representative tasks and with a broad set of representative models.

In general, we find that the advantages of DAL can be substantial. For example, on NER tasks, we achieve roughly 98-99% of the full-dataset performance while labeling only 20% of the samples for both CNN-BiLSTM-CRF and CNN-CNN-LSTM models. By comparison, the i.i.d. baseline requires 50% of the data to achieve comparable F score. While the reduction in the percentage of data required is not as dramatic in the classification datasets (possibly owing to their comparatively small size), the relative improvement over i.i.d. baselines remains significant.

4 Conclusion

This paper set out to investigate the practical utility of DAL for NLP. Our study consisted of over 40 experiments, each repeated for 3 times to average results and consisting of roughly 25 rounds of retraining, adding up to 3000 training runs to completion. Our goal was not to champion any one approach, but to ask if there was any consistent story at all: *can active learning be applied on a new dataset with an arbitrarily architecture, without peeking at the labels to perform hyperparameter tuning?* To our surprise, we found that across many tasks, both classic uncertainty sampling and Bayesian approaches outperform i.i.d. baselines and that DO-BALD and BB-BALD consistently perform best.

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. In *International Conference on Machine Learning (ICML)*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the and shared task: Semantic role labeling. In *Computational Natural Language Learning (CoNLL)*.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. 2017. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*.

Yarin Gal and Zoubin Ghahramani. 2016a. Bayesian convolutional neural networks with Bernoulli approximate variational inference. *International Conference on Learning Representations (ICLR) Workshop Track*.

Yarin Gal and Zoubin Ghahramani. 2016b. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning ICML)*.

Yarin Gal and Zoubin Ghahramani. 2016c. A theoretically grounded application of dropout in recurrent neural networks. In *Neural information processing systems (NIPS)*.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep Bayesian active learning with image data. In *International Conference on Machine Learning (ICML)*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Association for Computational Linguistics (ACL)*.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

Peiyun Hu, Zachary C Lipton, Anima Anandkumar, and Deva Ramanan. 2018. Active learning with partial feedback. *arXiv preprint arXiv:1802.07427*.

Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Neural Information Processing Systems (NIPS)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Empirical Methods in Natural Language Processing (EMNLP)*.

D Kinga and J Ba Adam. 2015. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. 2018. BBQ-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. *Association for the Advancement of Artificial Intelligence (AAAI)*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. *arXiv preprint arXiv:1603.01354*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Carl Edward Rasmussen. 2004. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer.

Dan Roth, Chad M Cumby, Xin Li, Paul Morie, Ramya Nagarajan, Nick Rizzolo, Kevin Small, and Wen-tau Yih. 2002. Question-answering via enhanced understanding of questions. In *Text Retrieval Conference (TREC)*.

Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. 2018. Deep active learning for named entity recognition. In *International Conference on Learning Representations (ICLR)*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Computational Natural Language Learning (CoNLL)*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Ye Zhang, Matthew Lease, and Byron C Wallace. 2017. Active discriminative text representation learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Bayesian Compression for Natural Language Processing

Nadezhda Chirkova[1*], **Ekaterina Lobacheva**[1*], **Dmitry Vetrov**[1,2]
[1]Samsung-HSE Laboratory, National Research University Higher School of Economics
[2]Samsung AI Center
Moscow, Russia
`{nchirkova,elobacheva,dvetrov}@hse.ru`

Abstract

In natural language processing, a lot of the tasks are successfully solved with recurrent neural networks, but such models have a huge number of parameters. The majority of these parameters are often concentrated in the embedding layer, which size grows proportionally to the vocabulary length. We propose a Bayesian sparsification technique for RNNs which allows compressing the RNN dozens or hundreds of times without time-consuming hyperparameters tuning. We also generalize the model for vocabulary sparsification to filter out unnecessary words and compress the RNN even further. We show that the choice of the kept words is interpretable.

1 Introduction

Recurrent neural networks (RNNs) are among the most powerful models for natural language processing, speech recognition, question-answering systems (Chan et al., 2016; Ha et al., 2017; Wu et al., 2016; Ren et al., 2015). For complex tasks such as machine translation (Wu et al., 2016) modern RNN architectures incorporate a huge number of parameters. To use these models on portable devices with limited memory the model compression is desired.

There are a lot of RNNs compression methods based on specific weight matrix representations (Tjandra et al., 2017; Le et al., 2015) or sparsification (Narang et al., 2017; Wen et al., 2018). In this paper we focus on RNNs compression via sparsification. One way to sparsify RNN is pruning where the weights with a small absolute value are eliminated from the model. Such methods are heuristic and require time-consuming hyperparameters tuning. There is another group of sparsification techniques based on Bayesian approach. Molchanov et al. (2017) describe a model called SparseVD in which parameters controlling sparsity are tuned automatically during neural network training. However, this technique was not previously investigated for RNNs. In this paper, we apply Sparse VD to RNNs taking into account the specifics of recurrent network structure (Section 3.2). More precisely, we use the insight about using the same sample of weights for all timesteps in the sequence (Gal and Ghahramani, 2016; Fortunato et al., 2017). This modification makes local reparametrization trick (Kingma et al., 2015; Molchanov et al., 2017) not applicable and changes SparseVD training procedure.

In natural language processing tasks the majority of weights in RNNs are often concentrated in the first layer that is connected to the vocabulary, for example in embedding layer. However, for some tasks the most of the words are unnecessary for accurate predictions. In our model we introduce multiplicative weights for the words to perform vocabulary sparsification (Section 3.3). These multiplicative weights are zeroing out during training causing filtering corresponding unnecessary words out of the model. It allows to boost RNN sparsification level even further.

To sum up, our contributions are as follows: (i) we adapt SparseVD to RNNs explaining the specifics of the resulting model and (ii) we generalize this model by introducing multiplicative weights for words to purposefully sparsify the vocabulary. Our results show that Sparse Variational Dropout leads to a very high level of sparsity in recurrent models without a significant quality drop. Models with additional vocabulary sparsification boost compression rate on text classification tasks but do not help that much on language modeling tasks. In classification tasks the vocabulary is compressed dozens of times, and the choice of words is interpretable.

---

[*] Equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2910–2915
Brussels, Belgium, October 31 - November 4, 2018.

2 Related work

Reducing RNN size is an important and rapidly developing area of research. There are three research directions: approximation of weight matries (Tjandra et al., 2017; Le et al., 2015), reducing the precision of the weights (Hubara et al., 2016) and sparsification of the weight matrices (Narang et al., 2017; Wen et al., 2018). We focus on the last one. The most popular approach here is pruning: the weights of the RNN are cut off on some threshold. Narang et al. (2017) choose threshold using several hyperparameters that control the frequency, the rate and the duration of the weights eliminating. Wen et al. (2018) propose to prune the weights in LSTM by groups corresponding to each neuron, this allows to accelerate forward pass through the network.

Another group of sparsification methods relies on Bayesian neural networks (Molchanov et al., 2017; Neklyudov et al., 2017; Louizos et al., 2017). In Bayesian NNs the weights are treated as random variables, and our desire about sparse weights is expressed in a prior distribution over them. During training, the prior distribution is transformed into the posterior distribution over the weights, used to make predictions on testing phase. Neklyudov et al. (2017) and Louizos et al. (2017) also introduce group Bayesian sparsification techniques that allow to eliminate neurons from the model.

The main advantage of the Bayesian sparsification techniques is that they have a small number of hyperparameters compared to pruning-based methods. Also, they lead to a higher sparsity level (Molchanov et al., 2017; Neklyudov et al., 2017; Louizos et al., 2017).

There are several works on Bayesian recurrent neural networks (Gal and Ghahramani, 2016; Fortunato et al., 2017), but these methods are hard to extend to achieve sparsification. We apply sparse variational dropout to RNNs taking into account its recurrent specifics, including some insights highlighted by Gal and Ghahramani (2016), Fortunato et al. (2017).

3 Proposed method

3.1 Notations

In the rest of the paper $x = [x_0, \dots, x_T]$ is an input sequence, y is a true output and $\hat{y}$ is an output predicted by the RNN (y and $\hat{y}$ may be single vectors, sequences, etc.), X, Y denotes a training set $\{(x^1, y^1), \dots, (x^N, y^N)\}$. All weights of the RNN except biases are denoted by ω, while a single weight (an element of any weight matrix) is denoted by w_{ij}. Note that we detach biases and denote them by B because we do not sparsify them.

For definiteness, we will illustrate our model on an example architecture for the language modeling task, where $y = [x_1, \dots, x_T]$:

$$\begin{aligned}
&\text{embedding} : \tilde{x}_t = w^e_{x_t};\\
&\text{recurrent} : h_{t+1} = \sigma(W^h h_t + W^x \tilde{x}_{t+1} + b^r);\\
&\text{fully-connected} : \hat{y}_t = \text{softmax}(W^d h_t + b^d).
\end{aligned}$$

In this example $\omega = \{W^e, W^x, W^h, W^d\}$, $B = \{b^r, b^d\}$. However, the model may be directly applied to any recurrent architecture.

3.2 Sparse variational dropout for RNNs

Following Kingma et al. (2015), Molchanov et al. (2017), we put a fully-factorized log-uniform prior over the weights:

$$p(\omega) = \prod_{w_{ij} \in \omega} p(w_{ij}), \quad p(w_{ij}) \propto \frac{1}{|w_{ij}|}$$

and approximate the posterior with a fully factorized normal distribution:

$$q(w|\theta, \sigma) = \prod_{w_{ij} \in \omega} \mathcal{N}\big(w_{ij}|\theta_{ij}, \sigma^2_{ij}\big).$$

The task of posterior approximation $\min_{\theta,\sigma,B} KL(q(\omega|\theta,\sigma)||p(\omega|X,Y,B))$ is equivalent to variational lower bound optimization (Molchanov et al., 2017):

$$-\sum_{i=1}^{N} \int q(\omega|\theta,\sigma) \log p(y^i|x^i_0, \dots, x^i_T, \omega, B) d\omega + \\ + \sum_{w_{ij} \in \omega} KL(q(w_{ij}|\theta_{ij}, \sigma_{ij})||p(w_{ij})) \to \min_{\theta,\sigma,B}. \tag{1}$$

Here the first term, a task-specific loss, is approximated with one sample from $q(\omega|\theta,\sigma)$. The second term is a regularizer that moves posterior closer to prior and induces sparsity. This regularizer can be very closely approximated analytically (Molchanov et al., 2017):

$$KL(q(w_{ij}|\theta_{ij}, \sigma_{ij})||p(w_{ij})) \approx k\left(\frac{\sigma^2_{ij}}{\theta^2_{ij}}\right), \tag{2}$$

$$k(\alpha) \approx 0.64\sigma(1.87 + 1.49\log\alpha) - \frac{1}{2}\log\Bigl(1 + \frac{1}{\alpha}\Bigr).$$

To make integral estimation unbiased, sampling from the posterior is performed with the use of reparametrization trick (Kingma and Welling, 2014):

$$w_{ij} = \theta_{ij} + \sigma_{ij}\epsilon_{ij}, \quad \epsilon_{ij} \sim \mathcal{N}(\epsilon_{ij}|0, 1) \qquad (3)$$

The important difference of RNNs compared to feed-forward networks consists in sharing the same weight variable between different timesteps. Thus, we should use the same sample of weights for each timestep t while computing the likelihood $p(y^i|x_0^i, \ldots, x_T^i, \omega, B)$ (Gal and Ghahramani, 2016; Fortunato et al., 2017).

Kingma et al. (2015), Molchanov et al. (2017) also use local reparametrization trick (LRT) that is sampling preactivation instead of individual weights. For example,

$$(W^x x_t)_i = \sum_j \theta_{ij}^x x_{tj} + \epsilon_i \sum_j (\sigma_{ij}^x)^2 x_{tj}^2.$$

Tied weight sampling makes LRT not applicable to weight matrices that are used in more than one timestep in the RNN.

For the hidden-to-hidden matrix W^h the linear combination $(W^h h_t)$ is not normally distributed because h_t depends on W^h from the previous timestep. As a result, the rule about the sum of independent normal distributions with constant coefficients is not applicable. In practice, network with LRT on hidden-to-hidden weights cannot be trained properly.

For the input-to-hidden matrix W^x the linear combination $(W^x x_t)$ is normally distributed. However, sampling the same W^x for all timesteps and sampling the same noise ϵ_i for preactivations for all timesteps are not equivalent. The same sample of W^x corresponds to different samples of noise ϵ_i at different timesteps because of the different x_t. Hence theoretically LRT is not applicable here. In practice, networks with LRT on input-to-hidden weights may give the same results and in some experiments, they even converge a little bit faster.

Since the training procedure is effective only with 2D noise tensor, we propose to sample the noise on the weights per mini-batch, not per individual object.

To sum up, the training procedure is as follows. To perform forward pass for a mini-batch, we firstly sample all weights ω following (3) and then apply RNN as usual. Then the gradients of (1) are computed w.r.t $\theta, \log\sigma, B$.

During the testing stage, we use the mean weights θ (Molchanov et al., 2017). Regularizer (2) causes the majority of θ components approach 0, and the weights are sparsified. More precisely, we eliminate weights with low signal-to-noise ratio $\frac{\theta_{ij}^2}{\sigma_{ij}^2} < \tau$ (Molchanov et al., 2017).

3.3 Multiplicative weights for vocabulary sparsification

One of the advantages of Bayesian sparsification is an easy generalization for the sparsification of any groups of the weights that doesn't complicate the training procedure (Louizos et al., 2017). To do so, one should introduce shared multiplicative weight per each group, and elimination of this multiplicative weight will mean the elimination of the corresponding group. In our work we utilize this approach to achieve vocabulary sparsification.

Precisely, we introduce multiplicative probabilistic weights $z \in \mathbb{R}^V$ for words in the vocabulary (here V is the size of the vocabulary). The forward pass with z looks as follows:

1. sample vector z^i from the current approximation of the posterior for each input sequence x^i from the mini-batch;
2. multiply each one-hot encoded token x_t^i from the sequence x^i by z^i (here both x_t^i and z^i are V-dimensional);
3. continue the forward pass as usual.

We work with z in the same way as with other weights W: we use a log-uniform prior and approximate the posterior with a fully-factorized normal distribution with trainable mean and variance. However, since z is a one-dimensional vector, we can sample it individually for each object in a mini-batch to reduce the variance of the gradients. After training, we prune elements of z with a low signal-to-noise ratio and subsequently, we do not use the corresponding words from the vocabulary and drop columns of weights from the embedding or input-to-hidden weight matrices.

4 Experiments

We perform experiments with LSTM architecture on two types of problems: text classification and language modeling. Three models are compared here: baseline model without any regularization, SparseVD model and SparseVD model with multiplicative weights for vocabulary sparsification (SparseVD-Voc).

To measure the sparsity level of our models we calculate the compression rate of individual weights as follows: $|w|/|w \neq 0|$. The sparsification of weights may lead not only to the compression but also to the acceleration of RNNs through group sparsity. Hence, we report the number of remaining neurons in all layers: input (vocabulary), embedding and recurrent. To compute this number for vocabulary layer in SparseVD-Voc we use introduced variables z_v. For all other layers in SparseVD and SparseVD-Voc, we drop a neuron if all weights connected to this neuron are eliminated.

We optimize our networks using Adam (Kingma and Ba, 2015). Baseline networks overfit for all our tasks, therefore, we present results for them with early stopping. For all weights that we sparsify, we initialize $\log \sigma$ with -3. We eliminate weights with signal-to-noise ratio less then $\tau = 0.05$. More details about experiment setup are presented in Appendix A.

4.1 Text Classification

We evaluated our approach on two standard datasets for text classification: IMDb dataset (Maas et al., 2011) for binary classification and AGNews dataset (Zhang et al., 2015) for four-class classification. We set aside 15% and 5% of training data for validation purposes respectively. For both datasets, we use the vocabulary of 20,000 most frequent words.

We use networks with one embedding layer of 300 units, one LSTM layer of 128 / 512 hidden units for IMDb / AGNews, and finally, a fully connected layer applied to the last output of the LSTM. Embedding layer is initialized with word2vec (Mikolov et al., 2013) / GloVe (Pennington et al., 2014) and SparseVD and SparseVD-Voc models are trained for 800 / 150 epochs on IMDb / AGNews.

The results are shown in Table 1. SparseVD leads to a very high compression rate without a significant quality drop. SparseVD-Voc boosts the compression rate even further while still preserving the accuracy. Such high compression rates are achieved mostly because of the sparsification of the vocabulary: to classify texts we need to read only some important words from them. The remaining words in our models are mostly interpretable for the task (see Appendix B for the list of remaining words for IMBb). Figure 1 shows the only kept embedding component for remaining words on IMDb. This component reflects the sentiment score of the words.

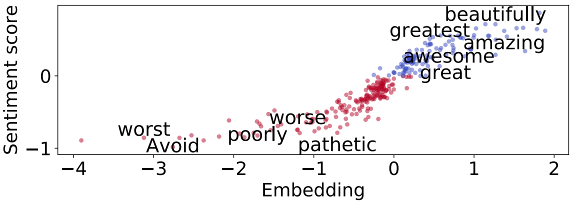

Figure 1: IMDB: remained embedding component vs sentiment score ((#pos. - #neg.) / #all texts with the word).

4.2 Language Modeling

We evaluate our models on the task of character-level and word-level language modeling on the Penn Treebank corpus (Marcus et al., 1993) according to the train/valid/test partition of Mikolov et al. (2011). The dataset has a vocabulary of 50 characters or 10,000 words.

To solve character / word-level tasks we use networks with one LSTM layer of 1000 / 256 hidden units and fully-connected layer with softmax activation to predict next character or word. We train SparseVD and SparseVD-Voc models for 250 / 150 epochs on character-level / word-level tasks.

The results are shown in Table 2. To obtain these results we employ LRT on the last fully-connected layer. In our experiments with language modeling LRT on the last layer accelerate the training without harming the final result. Here we do not get such extreme compression rates as in the previous experiment but still, we are able to compress the models several times while achieving better quality w.r.t. the baseline because of the regularization effect of SparseVD. Vocabulary is not sparsified in the character-level task because there are only 50 characters and all of them matter. In the word-level task more than a half of the words are dropped. However, since in language modeling almost all words are important, the sparsification of the vocabulary makes the task more difficult to the network and leads to the drop in quality and the overall compression (network needs more difficult dynamic in the recurrent layer).

Task	Method	Accuracy %	Compression	Vocabulary	Neurons $\tilde{x}$ - h
	Original	84.1	1x	20000	$300-128$
IMDb	SparseVD	**85.1**	1135x	4611	$16-17$
	SparseVD-Voc	83.6	**12985x**	**292**	$\mathbf{1-8}$
	Original	**90.6**	1x	20000	$300-512$
AGNews	SparseVD	88.8	322x	5727	$179-56$
	SparseVD-Voc	89.2	**469x**	**2444**	$\mathbf{127-32}$

Table 1: Results on text classification tasks. Compression is equal to $|w|/|w \neq 0|$. In last two columns number of remaining neurons in the input, embedding and recurrent layers are reported.

Task	Method	Valid	Test	Compression	Vocabulary	Neurons h
	Original	1.498	1.454	1x	50	1000
Char PTB	SparseVD	1.472	1.429	**4.2x**	50	**431**
Bits-per-char	SparseVD-Voc	**1.4584**	**1.4165**	3.53x	**48**	510
	Original	135.6	129.5	1x	10000	256
Word PTB	SparseVD	**115.0**	**109.0**	**14.0x**	9985	**153**
Perplexity	SparseVD-Voc	126.3	120.6	11.1x	**4353**	207

Table 2: Results on language modeling tasks. Compression is equal to $|w|/|w \neq 0|$. In last two columns number of remaining neurons in input and recurrent layers are reported.

Acknowledgments

Results on SparseVD for RNNs shown in Section 3.2 have been supported by Russian Science Foundation (grant 17-71-20072). Results on multiplicative weights for vocabulary sparsification shown in Section 3.3 have been supported by Samsung Research, Samsung Electronics.

References

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. 2017. Bayesian recurrent neural networks. *Computing Research Repository*, arXiv:1704.02798.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. Proceedings of Machine Learning Research.

David Ha, Andrew Dai, and Quoc V. Le. 2017. Hypernetworks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Quantized neural networks: Training neural networks with low precision weights and activations. *Computing Research Repository*, arXiv:1609.07061.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*.

Diederik P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28*, pages 2575–2583.

Diederik P Kingma and Max Welling. 2014. Autoencoding variational bayes. In *Proceedings of the International Conference for Learning Representations (ICLR)*.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *Computing Research Repository*, arXiv:1504.00941.

Christos Louizos, Karen Ullrich, and Max Welling. 2017. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems 30*, pages 3288–3298.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of* *the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur. 2011. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. 2017. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*.

Sharan Narang, Gregory F. Diamos, Shubho Sengupta, and Erich Elsen. 2017. Exploring sparsity in recurrent neural networks. In *Proceedings of the International Conference for Learning Representations (ICLR)*.

Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P Vetrov. 2017. Structured bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems 30*, pages 6778–6787.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, volume 14, pages 1532–1543.

Mengye Ren, Ryan Kiros, and Richard S. Zemel. 2015. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2953–2961.

Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2017. Compressing recurrent neural network with tensor train. *Computing Research Repository*, arXiv:1705.08052.

Wei Wen, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li. 2018. Learning intrinsic sparse structures within long short-term memory. In *International Conference on Learning Representations*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *Computing Research Repository*, arXiv:1609.08144.

X. Zhang, J. Zhao, and Y. LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems (NIPS)*.

Multimodal neural pronunciation modeling for spoken languages with logographic origin

Minh Nguyen
National University of Singapore
elenguy@nus.edu.sg

Gia H. Ngo
National University of Singapore
ngohgia@u.nus.edu

Nancy F. Chen
Institute for Infocomm Research
Singapore
nancychen@alum.mit.edu

Abstract

Graphemes of most languages encode pronunciation, though some are more explicit than others. Languages like Spanish have a straightforward mapping between its graphemes and phonemes, while this mapping is more convoluted for languages like English. Spoken languages such as Cantonese present even more challenges in pronunciation modeling: (1) they do not have a standard written form, (2) the closest graphemic origins are logographic Han characters, of which only a subset of these logographic characters implicitly encodes pronunciation. In this work, we propose a multimodal approach to predict the pronunciation of Cantonese logographic characters, using neural networks with a geometric representation of logographs and pronunciation of cognates in historically related languages. The proposed framework improves performance by 18.1% and 25.0% respective to unimodal and multimodal baselines.

1 Introduction

In phonographic languages, there is a direct correspondence between graphemes and phonemes (Defrancis, 1996), though this correspondence is not always one-to-one. For example, in English, the word *table* corresponds to the pronunciation [“teɪ.bl], in which each alphabetic character corresponds to one phoneme, and the character *e* is mapped to silence. However, in logographic languages, the correspondence between graphemes and phonemes is more ambiguous (Defrancis, 1996), as only some sub-units in a grapheme are indicative of its phonemes. Korean[1], Vietnamese[2] and Chinese languages (e.g. Cantonese) are examples of logographic languages, all belonging to the Han logographic family. Similar to pronunciation modeling in phonographic languages, in which words are broken down into characters and modeling is done at the character level, pronunciation modeling in logographic languages requires decomposing logographs into sub-units and extracting only sub-units carrying pronunciation hints. As the correspondence of Han logograph to phoneme is intricately complex with many sub-rules or exceptions (Hashimoto, 1978), it is challenging to computationally model these correspondences using white box approaches (e.g. graphical model). Instead, we exploit neural networks, as they (1) can flexibly model the implicit similarity of grapheme-phoneme relationships across languages with Han origin, (2) can automatically learn the most relevant knowledge representation with minimal feature engineering (LeCun et al., 2015), such as extracting pronunciation hints from logographic representations.

Due to historical contact, there is much lexical overlap across Han logographic languages, as they borrowed words from one another (Rokuro, 1969; Miyake, 1997; Loveday, 1996; Sohn, 2001; Alves, 1999). As a result, cognates in different languages are written using identical graphemes but pronounced differently. For example, [she] in Mandarin and [sip] in Cantonese are cognates; their pronunciations are different yet they are written using the same logograph (懾), which represents “admire”. Though Han logographic languages are mutually unintelligible (Tang and Van Heuven, 2009; Handel, 2015), the correspondence of Han logographic graphemes to phonemes across languages is often similar in systematic ways (Cai et al., 2011; Frellesvig and Whitman, 2008; Miyake, 1997). The shared characteristics in pronunciation of cognates could be leveraged in deciphering the pronunciation of Han logographs. In this work, we proposed a neural pronuncia-

---

[1] A large portion of Korean vocabulary are Sino-Korean written in Hanja (Korean logographs) (Sohn, 2001)

[2] Traditional Vietnamese vocabulary comprises of Sino-Vietnamese words written by Chinese logographs and locally-invented Nom logographs (Alves, 1999).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2916–2922
Brussels, Belgium, October 31 - November 4, 2018.

tion model that exploits both embeddings of logographs and cognates' phonemes. The proposed model significantly improves pronunciation prediction of logographs in Cantonese.

2 Related Work

The basic units in writing (graphemes) of Han logographic languages are logographs. A word contains one or more logographs and a logograph consists of one or more radicals. The pronunciation of a logograph corresponds to a syllable which has three phonemes: onset, nucleus and coda.

Grapheme-to-phoneme (G2P) approaches such as (Xu et al., 2004; Chen et al., 2016) predicted a Han logograph's pronunciation from its local context in a phrase. This was similar to predicting a Latin word's pronunciation from its surrounding words, essentially treated individual logographs as the basic units of the model and did not delve further into the logographic sub-units (the radicals).

While we are unaware of any work that derives features for pronunciation prediction from logographs, there are recent work in deriving representation of logographs for various semantic tasks. Some methods (Shi et al., 2015; Ke and Hagiwara, 2017; Nguyen et al., 2017; Zhuang et al., 2017) decomposed logographs into subunits using expert-defined rules and then extracted the relevant semantic features. Other methods use convolutional neural network to extract features from the images of logographs (Dai and Cai, 2017; Liu et al., 2017; Toyama et al., 2017). Other works combined multiple level of information for feature extraction, using both logograph and sub-units obtained from logograph decomposition (Dong et al., 2016; Han et al., 2017; Peng et al., 2017; Yu et al., 2017; Yin et al., 2016).

In this work, we explicitly looked at the relationship between a logograph's constituent radicals and its pronunciation. Among Han logographs, 81% of frequently used logographs are semantic-phonetic compounds (Li and Kang, 1993) which consist of radicals that might contain phonetic or semantic hints (Hsiao and Shillcock, 2006). The pronunciation of a logograph could conceivably be predicted from the phonetic radicals. Furthermore, the relative position of radicals in the logograph might also offer clues about it pronunciation. Table 1 shows an example of such intricate relationships between a logograph's pronunciation and its constituent radicals. All Han logographs in the table have a common phonetic radical (in red), which offers an inkling of the pronunciation of these logographs. For instance, logographs that have the phonetic radical on the left (剖 and 部) share a similar pronunciation in Korean (in blue) while logographs that have the phonetic radical on the right (陪, 賠, and 蓓) share a similar pronunciation in Mandarin, Cantonese and Vietnamese. Note that for each logograph, their pronunciations across the different languages share similarities: when the phonetic radical is on the left, the nucleus ends in a back vowel like *u* or *o*, whereas when the phonetic radical is on the right, the nucleus ends in a front vowel like *i*.

Position of 音					
Logograph	剖	部	陪	賠	蓓
Mandarin	pou	bu	**pei**	**pei**	bei
Cantonese	fau	bou	**pui**	**pui**	bui
Korean	**pwu**	**pwu**	**pay**	**pay**	**pay**
Vietnamese	phau	bo	**boi**	**boi**	bui

Table 1: The position of radicals affects pronunciations. All logographs share a common radical in red. Similar pronunciations for 剖 and 部 are bolded in blue. Similar pronunciations for 陪, 賠, and 蓓 are bolded in green. The pronunciation of a logograph in Mandarin, Cantonese, Korean and Vietnamese are represented by Pinyin, Jyutping, Yale, and Vietnamese alphabet symbols respectively.

The example in Table 1 explains the motivation for our proposed approach to predict a logograph's pronunciation by modelling both the constituent radicals and their geometric positions. Furthermore, the proposed approach can generalize to unseen logographs if the co-occurrence patterns of their constituent radicals have been learnt.

3 Model

We first describe a geometric decomposition of logographs and then different neural pronunciation models for logographs. Finally, we present a multimodal neural model that incorporates both logographic input and the cognates' phonemes in predicting pronunciation of logographs.

Representation of Han logographs

The majority of logographs (characters) in Han logographic language family comprise of a radical that indicates its nominal semantic category and a phonetic radical that gives an inkling of the pronunciation (Defrancis, 1996). Thus, patterns of co-occurrence of radicals across logographs might

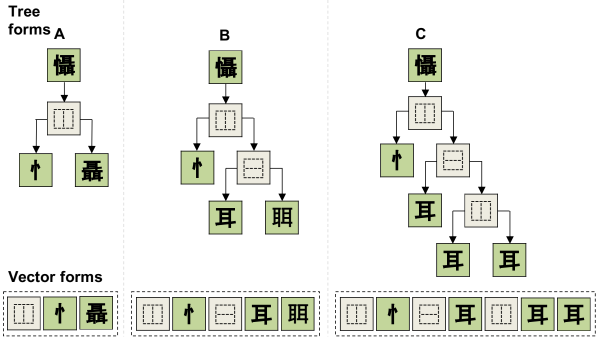

*Figure 1: Geometric representation of the logograph "admire". **A**, **B** and **C** are equivalent decomposition of the same logograph but with different levels of granularity. The geometric representation comprises of both the radicals and geometric operators, which can be used to reconstruct the original logograph.*

be exploited to find the phonetic radicals, which in turn can suggest the corresponding pronunciation of a logograph. Using this intuition, we model the pronunciation of logographs at the radical level.

We investigated two representations of radicals in a logograph. In the first approach, a logograph is represented as a bag of its unordered constituent radicals (BoR), encoded as a vector of radical counts. The second approach is to use a decomposition of radicals in the logograph that retains the original geometric organization of the radicals. The geometric decomposition (GeoD) approach preserves important cues about the word's pronunciation in the relative position of the radicals. For example, differentiating the left radical from the right radical in a left-right semantic-phonetic compound allows more effective extraction of pronunciation hints. In addition, radicals that should be interpreted together are closer spatially in the GeoD representation, making the knowledge representation easier to learn. Note that the GeoD representation is lossless as the original logograph can be reconstructed perfectly (details in Appendix A). Figure 1 shows the geometric decomposition of the Han logograph "admire" at three levels of granularity.

Neural pronunciation prediction models

Figure 2 and Figure 3 show two neural pronunciation prediction models of logographs. In Figure 2, each logograph is treated as an ordered "bag of radicals" (BoR). For example, assume the vocabulary of radicals in the whole dataset is [忄, 氵, 耳, 灬], the word 懾 ("admire" - see Figure 1) is represented by a vector of counts $[1, 0, 3, 0]$, corresponding to one radical 忄 and three radicals 耳. The BoR is input to a multilayer perceptron (MLP) with three layers of size 750, 500, 250. L2 regularization of 1e-4 is applied to the hidden layers. The three dropout layers have dropout probabilities of 0.5, 0.5, and 0.2, respectively. As the output variables are categorical, cross-entropy loss was used.

We investigated two structures for predicting output phonemes (i.e. onset, nucleus, coda). In the first structure, output phonemes were predicted independently using the last hidden layer. The second structure made a sequential prediction (1) the coda was first predicted using the last hidden layer (2) the nucleus was predicted using both the final hidden layer and the predicted coda, and (3) the onset was predicted using the last hidden layer together with the predicted coda and nucleus. The second structure was motivated by a stronger dependency between the nuclues and coda. For example, the nucleus and coda are often grouped together as a single unit (rime/final) in the syllabic structure of most languages (Kessler and Treiman, 2002). In our experiments, the sequential structure yielded lower error rates so it is used in all neural network models.

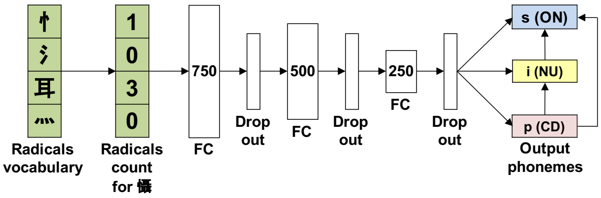

Figure 2: Pronunciation model of logographs using multilayer perceptron (MLP). FC: Fully connected.

In Figure 3, each logograph is represented by its geometric decomposition (GeoD). For example, the logograph 懾 is represented by a sequence of radicals and geometric operators shown in Figure 1C. The neural prediction model consists of two LSTM layers with 256 memory cells each. Input and recurrent dropout (Gal and Ghahramani, 2016) of 0.2 and 0.5 are applied to the LSTM layers to prevent overfitting.

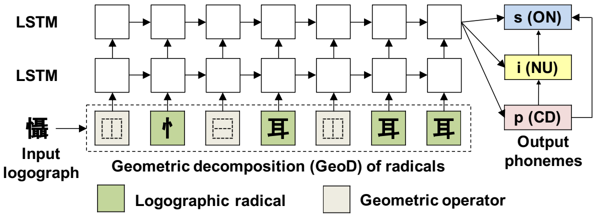

Figure 3: Neural pronunciation model with geometric decomposition of logographs.

Multimodal neural pronunciation model of logographs

In this section, we want to model the pronunciations of a logograph in the target language Cantonese using multimodal information from both the logograph and phonemes of the cognates, as shown in Figure 4. Given a vocabulary of phonemes in the source languages related to Cantonese (Mandarin, Korean, Vietnamese), the cognates' phonemes are encoded as an indicator vector, with an element equals 1 if the corresponding phoneme in the vocabulary appears in a cognate's pronunciation, and 0 otherwise.

The geometric decomposition (GeoD) of the logograph is fed to two LSTM layers. The output at the last time step is concatenated together with the multilingual phonemic vector and used as input for a multi-layer perceptron (MLP). The MLP and LSTM setups are the same as those in Figure 2 and Figure 3 respectively. Deep supervision (Szegedy et al., 2015) was applied by using the output of the LSTM to make auxiliary prediction of the output phonemes. Note that the auxiliary prediction should be identical to the main prediction. While predicting the same target, the main prediction used both cognate phonemes and the logograph while the auxiliary prediction used only the logograph. This was to ensure features extracted from the logographs are useful for pronunciation prediction and are complementary to the features extracted from the multilingual phonemes.

4 Experiments

We investigate whether Cantonese phonemes could be predicted using Han logographs and the cognates' phonemes from Mandarin, Korean, and Vietnamese. The prediction output are Cantonese onsets, nuclei and codas. The experimental design is motivated by the nature of Han-logographic languages. A Chinese logograph (character) is phonologically equivalent to a syllable in English while the constituent radicals are analogous to alphabet letters (with far less phonetic information). While in most languages, a syllable's pronunciation is influenced by neighboring syllables, most Han-logographic languages are monosyllabic and a logograph's pronunciation is rarely affected by neighboring logographs. Therefore, pronunciation prediction at the logograph (character) level for Han logographs is more appropriate. We use string error rate (SER) and token error rate (TER) as evaluation metrics. A wrongly predicted phoneme (onset, nucleus or coda) is counted as one token error. A syllable containing token error(s) is counted as one string error. All the neural networks were trained using Adam (Kingma and Ba, 2014).

Data

The dataset is extracted from the UniHan database,[3] which is a pronunciation database of logographs from Han logographic languages and maintained by the Unicode consortium. For each entry in the dataset, a logograph corresponds to phonemes in Cantonese, Mandarin, Korean and Vietnamese, represented by Jyutping,[4] Pinyin,[5] Yale,[6] and Vietnamese alphabet symbols respectively.[7] We randomly partition the dataset into two sets, with 80% for training and the other 20% for testing. Overall, there are 16,011 entries in the training set and 4,002 entries in the test set. 1000 entries of the training set are used as the development set for hyper-parameters fine-tuning.

In the test set, only 16% of logographs have pronunciations in all non-target languages, while 6% of logographs have no non-target language pronunciation. The availability of pronunciations in non-target languages differs from logograph to logograph. For example, some logographs have Mandarin and Korean pronunciations, while others only have Mandarin pronunciations.

Predicting pronunciation using logograph input

We compared the neural networks against a decision tree baseline. The decision tree baseline was implemented using scikit-learn (Pedregosa et al., 2011). The input of the decision tree (DT) model is the BoR representation of the logograph, while the input of neural networks can be either BoR or GeoD. The MLP network in Figure 2 uses BoR, while the LSTM in Figure 3 uses GeoD as input. All models output phonemes in Cantonese.

From Table 2, the neural network (MLP) outperforms decision tree when using BoR input. Both the SER and TER of the MLP model are lower than those of the decision tree. The LSTM model using GeoD leads to the lowest SER and TER, suggesting the benefits of relative positional

---

[3] https://www.unicode.org/charts/unihan.html

[4] https://en.wikipedia.org/wiki/Jyutping

[5] https://en.wikipedia.org/wiki/Pinyin

[6] https://en.wikipedia.org/wiki/Yale_romanization_of_Korean

[7] https://en.wikipedia.org/wiki/Vietnamese_alphabet

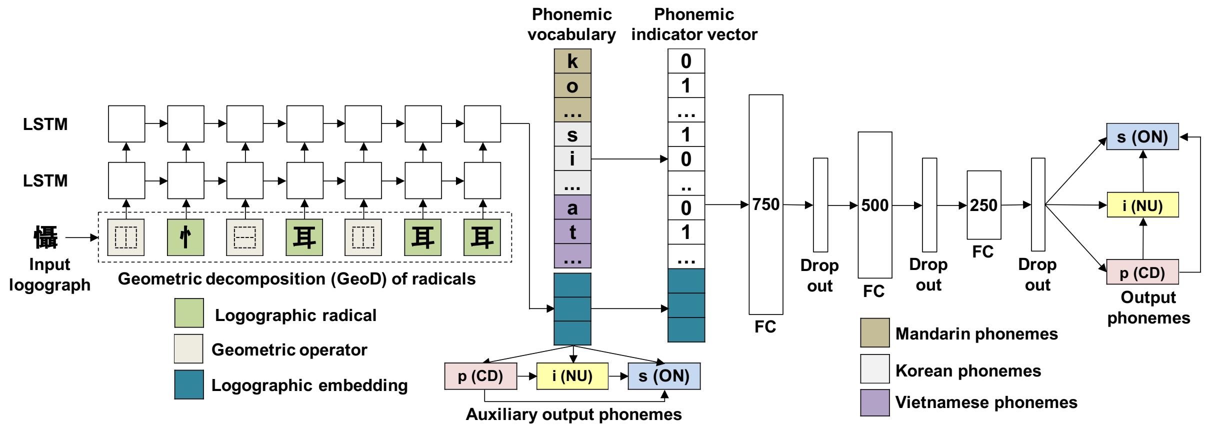

Figure 4: Multimodal neural pronunciation prediction model using logographs' geometric representation and cognates' phonemes.

information of radicals in predicting pronunciation. The trends of onset, nucleus and coda error rates are similar to those of TER and SER. However, as the gap of of error rate between MLP (BoR) and LSTM (GeoD) for TER and SER are quite small, using BoR instead of GeoD can be a good computation-accuracy trade-off.

Method	SER	TER	On.	Nu.	Cd.
DT (BoR)	63.8	39.8	50.7	45.7	22.9
MLP (BoR)	59.2	33.6	44.5	38.6	17.8
LSTM (GeoD)	**58.4**	**32.6**	**43.3**	**37.4**	**17.1**

Table 2: Prediction error rates of Cantonese phonemes by decision tree (DT), MLP and LSTM using only logographic input. Best results are in bold.

Predicting pronunciation using multimodal input

The input of the models are logographs and cognate phonemes from Mandarin, Korean and Vietnamese. Table 3 shows that the proposed multimodal neural network exploits multimodal and geometric information effectively. The relative improvement reaches 18.2% and 33.3% for SER and TER respectively. The last rows in Table 2 and Table 3 show that by combining Korean, Mandarin and Vietnamese phonemes input with GeoD, the prediction performance improves by 54.1% relative in TER and by 65.5% relative in SER. Moreover, using solely logograph input resulted in higher onset error (43.3%) than nucleus error (37.4%) while using both logographs and multilingual phonemes improves the onset error (23.5%) to be lower than nucleus error (24.6%). This suggests that logographs and phonemes of cognates provide complementary information about the pronunciation of a logograph, which in this case, most notably at the onset position. While logographs usually carry hints about phonemes at the nucleus and coda position but not at the onset position, multilingual phonemes input might carry hints about pronunciation at all three positions.

Method	SER	TER	On.	Nu.	Cd.
DT (BoR, ph)	44.0	24.8	29.8	29.9	14.7
MLP (BoR, ph)	38.5	19.6	23.4	24.8	10.5
LSTM (GeoD, ph)	**37.2**	**18.6**	**22.6**	**23.4**	**9.8**

Table 3: Prediction error rates of Cantonese phonemes by multimodal models; BoR: Bag of Radicals; GeoD: Geometric Decomposition; ph:phonemes. Best results are in bold.

5 Discussion

We have empirically shown that the systematic yet tenuous correspondence between pronunciations of cognates in Han logographic languages can be exploited for pronunciation modeling using neural networks. Moreover, combining logograph with cognate pronunciations further improves pronunciation prediction. These results could be potentially applied to speech processing tasks such as speech synthesis, where the construction of pronunciation dictionaries are expert labor-intensive, especially for under-resourced spoken languages.

For future work, recursive neural network (Tai et al., 2015) can be used as it is better suited for the hierarchical logographic decomposition. Besides, incorporating more detailed relationship between radicals (e.g. (Zhuang et al., 2017)) can help improve the model. The proposed approaches can also be applied to other languages such as Min Nan or Hakka, which are spoken languages that are even less well-documented than Cantonese.

References

Mark J Alves. 1999. What's so Chinese about Vietnamese. In *Papers from the ninth annual meeting of the Southeast Asian Linguistics Society*, pages 221–242.

Zhenguang G Cai, Martin J Pickering, Hao Yan, and Holly P Branigan. 2011. Lexical and syntactic representations in closely related languages: Evidence from Cantonese–Mandarin bilinguals. *Journal of Memory and Language*, 65(4):431–445.

Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2016. Acoustic data-driven pronunciation lexicon generation for logographic languages. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5350–5354. IEEE.

Falcon Z Dai and Zheng Cai. 2017. Glyph-aware Embedding of Chinese Characters. *EMNLP 2017*, page 64.

John Defrancis. 1996. Graphemic indeterminacy in writing systems. *Word*, 47(3):365–377.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based LSTM-CRF with radical-level features for Chinese named entity recognition. In *Natural Language Understanding and Intelligent Applications*, pages 239–250. Springer.

Bjarke Frellesvig and John Whitman. 2008. The Japanese-Korean vowel correspondences. *Japanese/Korean Linguistics*, 13:15–28.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

He Han, Yang Xiaokun, Wu Lei, Yan Hua, Gao Zhimin, Feng Yi, and Townsend George. 2017. Dual long short-term memory networks for sub-character representation learning. *arXiv preprint arXiv:1712.08841*.

Zev Handel. 2015. The classification of Chinese: sinitic (the Chinese language family). In *The Oxford handbook of Chinese linguistics*, pages 34–44. Oxford University Press.

Mantaro J Hashimoto. 1978. Current developments in Sino—Vietnamese studies. *Journal of Chinese Linguistics*, pages 1–26.

Janet Hui-wen Hsiao and Richard Shillcock. 2006. Analysis of a chinese phonetic compound database: Implications for orthographic processing. *Journal of psycholinguistic research*, 35(5):405–426.

Yuanzhi Ke and Masafumi Hagiwara. 2017. Radical-level Ideograph Encoder for RNN-based Sentiment Analysis of Chinese and Japanese. *arXiv preprint arXiv:1708.03312*.

Brett Kessler and Rebecca Treiman. 2002. Syllable structure and the distribution of phonemes in english syllables.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*. ArXiv: 1412.6980.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Y Li and JS Kang. 1993. Analysis of phonetics of the ideophonetic characters in Modern Chinese. *Information analysis of usage of characters in modern Chinese*, pages 84–98.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning character-level compositionality with visual features. *arXiv preprint arXiv:1704.04859*.

Leo J Loveday. 1996. *Language contact in Japan: A sociolinguistic history*. Clarendon Press.

Marc Hideo Miyake. 1997. Pre-Sino-Korean and Pre-Sino-Japanese: reexamining an old Problem from a modern perspective. *Japanese/Korean Linguistics*, 6:179–211.

Viet Nguyen, Julian Brooke, and Timothy Baldwin. 2017. Sub-character Neural Language Modelling in Japanese. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 148–153.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Haiyun Peng, Erik Cambria, and Xiaomei Zou. 2017. Radical-based hierarchical embeddings for Chinese sentiment analysis at sentence level. In *The 30th International FLAIRS conference. Marco Island*.

Kono Rokuro. 1969. The Chinese writing and its influence on the Scripts of the Neighbouring Peoples with special reference to Korea and Japan. *Memoirs of the Research Department of the Toyo Bunko (The Oriental Library) No*, 27:117–123.

Xinlei Shi, Junjie Zhai, Xudong Yang, Zehua Xie, and Chao Liu. 2015. Radical embedding: Delving deeper to Chinese radicals. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 594–598.

Ho-Min Sohn. 2001. *The Korean Language*. Cambridge University Press.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Chaoju Tang and Vincent J Van Heuven. 2009. Mutual intelligibility of Chinese dialects experimentally tested. *Lingua*, 119(5):709–732.

Yota Toyama, Makoto Miwa, and Yutaka Sasaki. 2017. Utilizing Visual Forms of Japanese Characters for Neural Review Classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 378–382.

Jun Xu, Guohong Fu, and Haizhou Li. 2004. Grapheme-to-phoneme conversion for chinese text-to-speech. In *Eighth International Conference on Spoken Language Processing*.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity Chinese word embedding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 981–986.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 2017. Joint Embeddings of Chinese Words, Characters, and Fine-grained Subcharacter Components. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 286–291.

Hang Zhuang, Chao Wang, Changlong Li, Qingfeng Wang, and Xuehai Zhou. 2017. Natural Language Processing Service Based on Stroke-Level Convolutional Networks for Chinese Text Classification. In *Web Services (ICWS), 2017 IEEE International Conference on*, pages 404–411. IEEE.

Chinese Pinyin Aided IME, Input What You Have Not Keystroked Yet

Yafang Huang^{1,2}, **Hai Zhao**^{1,2,*},
^{1}Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China
^{2}Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
huangyafang@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn *

Abstract

Chinese pinyin input method engine (IME) converts pinyin into character so that Chinese characters can be conveniently inputted into computer through common keyboard. IMEs work relying on its core component, pinyin-to-character conversion (P2C). Usually Chinese IMEs simply predict a list of character sequences for user choice only according to user pinyin input at each turn. However, Chinese inputting is a multi-turn online procedure, which can be supposed to be exploited for further user experience promoting. This paper thus for the first time introduces a sequence-to-sequence model with gated-attention mechanism for the core task in IMEs. The proposed neural P2C model is learned by encoding previous input utterance as extra context to enable our IME capable of predicting character sequence with incomplete pinyin input. Our model is evaluated in different benchmark datasets showing great user experience improvement compared to traditional models, which demonstrates the first engineering practice of building Chinese aided IME.

1 Introduction

Pinyin is the official romanization representation for Chinese and the P2C converting the inputted pinyin sequence to Chinese character sequence is the most basic module of all pinyin based IMEs.

Most of the previous research (Chen, 2003; Zhang et al., 2006; Lin and Zhang, 2008; Chen and Lee, 2000; Jiang et al., 2007; Cai et al., 2017a) for IME focused on the matching correspondence between pinyin syllables and Chinese characters. (Huang et al., 2018; Yang et al., 2012; Jia and Zhao, 2014; Chen et al., 2015) regarded the P2C as a translation between two languages and solved it in statistical or neural machine translation framework. The fundamental difference between (Chen et al., 2015) work and ours is that our work is a fully end-to-end neural IME model with extra attention enhancement, while the former still works on traditional IME only with converted neural network language model enhancement. (Zhang et al., 2017) introduced an online algorithm to construct appropriate dictionary for P2C. All the above mentioned work, however, still rely on a complete input pattern, and IME users have to input very long pinyin sequence to guarantee the accuracy of P2C module as longer pinyin sequence may receive less decoding ambiguity.

The Chinese IME is supposed to let user input Chinese characters with least inputting cost, i.e., keystroking, which indicates extra content predication from incomplete inputting will be extremely welcomed by all IME users. (Huang et al., 2015) partially realized such an extra predication using a maximum suffix matching postprocessing in vocabulary after SMT based P2C to predict longer words than the inputted pinyin.

To facilitate the most convenience for such an IME, in terms of a sequence to sequence model as neural machine translation (NMT) between pinyin sequence and character sequence, we propose a P2C model with the entire previous inputted utterance confirmed by IME users being used as a part of the source input. When learning the type of the previous utterance varies from the previous sentence in the same article to the previous turn of utterance in a conversation, the resulting IME will make amazing predication far more than what the pinyin IME users actually input.

In this paper, we adopt the attention-based NMT

---

* Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. 61672343 and No. 61733011), Key Project of National Society Science Foundation of China (No. 15-ZDA041), The Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04), and the joint research project with Youtu Lab of Tencent.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2923–2929
Brussels, Belgium, October 31 - November 4, 2018.

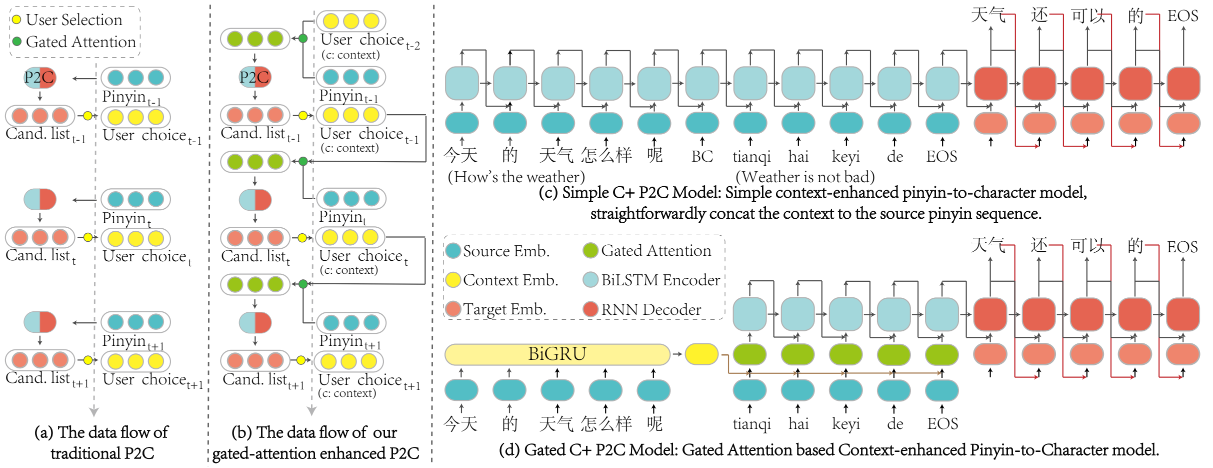

(a) The data flow of traditional P2C

(b) The data flow of our gated-attention enhanced P2C

(c) Simple C+ P2C Model: Simple context-enhanced pinyin-to-character model, straightforwardly concat the context to the source pinyin sequence.

(d) Gated C+ P2C Model: Gated Attention based Context-enhanced Pinyin-to-Character model.

Figure 1: Architecture of the proposed model.

framework in (Luong et al., 2015) for the P2C task. In contrast to related work that simply extended the source side with different sized context window to improve of translation quality (Tiedemann and Scherrer, 2017), we add the entire input utterance according to IME user choice at previous time (referred to the *context* hereafter). Hence the resulting IME may effectively improve P2C quality with the help of extra information offered by context and support incomplete pinyin input but predict complete, extra, and corrected character output. The evaluation and analysis will be performed on two Chinese datasets, include a Chinese open domain conversations dataset for verifying the effectiveness of the proposed method.

2 Model

As illustrated in Figure **1**, the core of our P2C is based the attention-based neural machine translation model that converts at word level. Still, we formulize P2C as a translation between pinyin and character sequences as shown in a traditional model in Figure 1(a). However, there comes a key difference from any previous work that our source language side includes two types of inputs, the current source pinyin sequence (noted as P) as usual, and the extended context, i.e., target character sequence inputted by IME user last time (noted as C). As IME works dynamically, every time IME makes a predication according to a source pinyin input, user has to indicate the 'right answer' to output target character sequence for P2C model learning. This online work mode of IMEs can be fully exploited by our model whose work flow is shown in Figure 1(b).

As introduced a hybrid source side input, our model has to handle document-wide translation by considering discourse relationship between two consecutive sentences. The most straightforward modeling is to simply concatenate two types of source inputs with a special token 'BC' as separator. Such a model is in Figure 1(c). However, the significant drawback of the model is that there are a slew of unnecessary words in the extended context (previous utterance) playing a noisy role in the source side representation.

To alleviate the noise issue introduced by the extra part in the source side, inspired by the work of (Dhingra et al., 2016; Pang et al., 2016; Zhang et al., 2018c,a,b; Cai et al., 2017b), our model adopts a gated-attention (GA) mechanism that performs multiple hops over the pinyin with the extended context as shown in Figure 1(d). In order to ensure the correlation between each other, we build a parallel bilingual training corpus and use it to train the pinyin embeddings and the Chinese embeddings at once. We use two Bidirectional gated recurrent unit (BiGRU) (Cho et al., 2014) to get contextual representations of the source pinyin and context respectively, $H_p = \mathrm{BiGRU}(P), H_c = \mathrm{BiGRU}(C)$, where the representation of each word is formed by concatenating the forward and backward hidden states.

For each pinyin p_i in H_p, the GA module forms a word-specific representation of the context $c_i \in H_c$ using soft attention, and then adopts element-wise product to multiply the context representation with the pinyin representation. $\alpha_i = softmax(H_c^{\mathrm{T}} p_i), \beta_i = C\alpha_i, x_i = p_i \odot \beta_i$, where $\odot$ is multiplication operator.

The pinyin representation $\tilde{H}_p = x_1, x_2, ..., x_k$

is augmented by context representation and then sent into the encoder-decoder framework. The encoder is a bi-directional long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997). The vectorized inputs are fed to forward and backward LSTMs to obtain the internal representation of two directions. The output for each input is the concatenation of the two vectors from both directions. Our decoder based on the global attentional models proposed by (Luong et al., 2015) to consider the hidden states of the encoder when deriving the context vector. The probability is conditioned on a distinct context vector for each target word. The context vector is computed as a weighted sum of previous hidden states. The probability of each candidate word as being the recommended one is predicted using a softmax layer over the inner-product between source embeddings and candidate target characters.

This work belongs to one of the first line which fully introduces end-to-end deep learning solution to the IME implementation following a series of our previous work (Zhu et al., 2018; Wu and Zhao, 2018; Zhang and Zhao, 2018; Bai and Zhao, 2018; Cai et al., 2018; He et al., 2018; Qin et al., 2018; Li et al., 2018; Jia et al., 2013).

3 Experiment

3.1 Definition of Incomplete Input for IME

The completeness in IME is actually uneasily well-defined as it is a relative concept for inputting procedure. Note that not until user types the return key *enter*, user will not (usually) really make the input choice. Meanwhile, even though the entire/complete input can be strictly defined by the time when user types *enter*, user still can make decision at any time and such incompleteness cannot be well evaluated by all the current IME metrics. As the incomplete from is hard to simulate and it is diverse in types, we have to partially evaluate it in the following two ways[1],

The incomplete pinyin as abbreviation pinyin To compare with previous work directly, we followed (Huang et al., 2015) and focused on the abbreviated pinyin (the consonant letter only) to perform evaluation (i.e., tian_qi to t_q).

Take incomplete user input as the incomplete As IME works as an interactive system, it will always give prediction only if users keep typing. If user's input does not end with typing *enter*, we can regard the current input pinyin sequence is an incomplete one.

3.2 Datasets and Settings

	PD		DC	
	Train	Test	Train	Test
# Sentence	5.0M	2.0K	1.0M	2.0K
$L < 10$ %	88.7	89.5	43.0	54.0
$L < 50$ %	11.3	10.5	47.0	42.0
$L > 50$ %	0.0	0.0	4.0	2.0
Relativity %	18.0	21.1	65.8	53.4

Table 1: Data statistics for the sentence number and sentence length (L) of two corpora.

Our model is evaluated on two datasets, namely the People's Daily (PD) corpus and Douban conversation (DC) corpus. The former is extracted from the People's Daily from 1992 to 1998 that has word segmentation annotations by Peking University. The DC corpus is created by (Wu et al., 2017) from Chinese open domain conversations. One sentence of the DC corpus contains one complete utterance in a continuous dialogue situation. The statistics of two datasets is shown in Table 1. The relativity refers to total proportion of sentences that associate with contextual history at word level. For example, there are 65.8% of sentences of DC corpus have words appearing in the context. With character text available, the needed parallel corpus between pinyin and character texts is automatically created following the approach proposed by (Yang et al., 2012).

Our model was implemented using the PyTorch[2] library, here is the hyperparameters we used: (a) the RNNs used are deep LSTM models, 3 layers, 500 cells, (c) 13 epoch training with plain SGD and a simple learning rate schedule - start with a learning rate of 1.0; after 9 epochs, halve the learning rate every epoch, (d) mini-batches are of size 64 and shuffled, (e) dropout is 0.3. Word embeddings are pre-trained by word2vec (Mikolov et al., 2013) toolkit on the adopted corpus and unseen words are assigned unique random vectors. (f) the gated attention layers size is 3, the hidden units number of BiGRU is 100.

Two metrics are used: Maximum Input Unit (MIU) accuracy (Zhang et al., 2017) and KeyStroke Score (KySS) (Jia and Zhao, 2013).

---

[1] Our code is at https://github.com/YvonneHuang/gaIME

[2] https://github.com/pytorch/pytorch

	DC		PD	
	Top-1	KySS	Top-1	KySS
CoCat⋆	49.72	0.7393	48.78	0.7868
Basic P2C⋆	52.31	0.7305	47.95	0.7879
Simple C+ P2C⋆	23.83	0.5431	43.95	0.7495
Gated C+ P2C⋆	53.76	0.8896	48.31	0.7931
CoCat	59.15	0.7651	61.42	0.7933
Basic P2C	71.31	0.8845	70.5	0.8301
Simple C+ P2C	61.28	0.7963	60.87	0.7883
Gated C+ P2C	**73.89**	**0.8935**	**70.98**	**0.8407**

Table 2: The effect of P2C modules with Different Input Forms. ⋆ means that the input is incomplete.

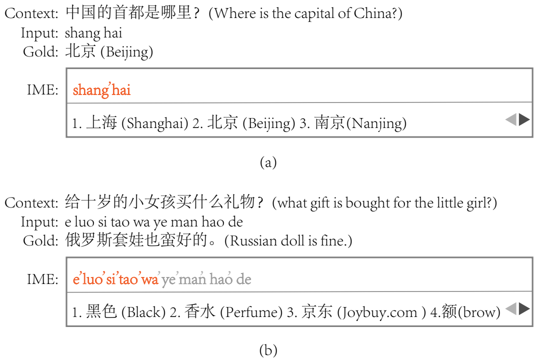
Context: 中国的首都是哪里？(Where is the capital of China?)
Input: shang hai
Gold: 北京 (Beijing)
IME: shang'hai
1. 上海 (Shanghai) 2. 北京 (Beijing) 3. 南京(Nanjing)

(a)

Context: 给十岁的小女孩买什么礼物？(what gift is bought for the little girl?)
Input: e luo si tao wa ye man hao de
Gold: 俄罗斯套娃也蛮好的。(Russian doll is fine.)
IME: e'luo'si'tao'wa'ye'man hao de
1. 黑色 (Black) 2. 香水 (Perfume) 3. 京东 (Joybuy.com) 4.额(brow)

(b)

Figure 2: Examples of the candidates list given by the proposed IMEs.

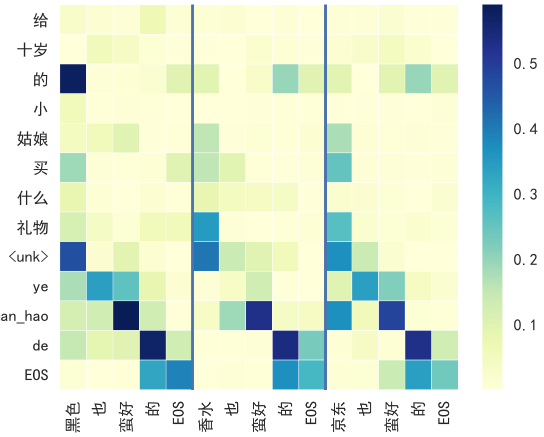

Figure 3: Attention visualization. Deeper color mean larger value.

The former measures the conversion accuracy of MIU, whose definition is the longest uninterrupted Chinese character sequence during inputting. As the P2C conversion aims to output a rank list of the corresponding character sequences candidates, the Top-K MIU accuracy means the possibility of hitting the target in the first K predicted items. The KySS quantifies user experience by using keystroke count. For an ideal IME with complete input, we have KySS $= 1$. An IME with higher KySS is supposed to perform better.

3.3 Model Definition

We considered the following baselines: (a) Google IME: the only commercial Chinese IME providing a debuggable API in the market now; (b) OMWA: online model for word acquisition proposed by (Zhang et al., 2017); (c) CoCat: an SMT based input method proposed by (Huang et al., 2015) that supports incomplete pinyin inputs.

Three models with incomplete or complete inputs will be evaluated: (a) Basic P2C, the basic P2C based on attention-NMT model; (b) Basic C2C, the basic C to C model based on Seq2Seq model; (b) Simple C+ P2C, the simple concatenated P2C conversion model that concatenate context to pinyin representation; (c) Gated C+ P2C, our gated attention based context-enhanced pinyin-to-character model. Pinyin in model * has been actually set to abbreviation form when we say it goes to (Huang et al., 2015) incomplete definition.

3.4 Result and Analysis

Effect of Gated Attention Mechanism Table 3 shows the Effect of gated attention mechanism. We compared models with Gated C+ P2C and Simple C+ P2C. The MIU accuracy of the P2C model has over 10% improvement when changing the operate pattern of the extra information proves the effect of GA mechanism. The Gated C+ P2C achieves the best in DC corpus, suggesting that the gated-attention works extremely well for handling long and diverse context.

Effect of P2C modules with Different Input Forms Table 2 shows the evaluation results of P2C modules with different input forms. It should

	DC				PD			
	Top-1	Top-5	Top-10	KySS	Top-1	Top-5	Top-10	KySS
Google IME	62.13	72.17	74.72	0.6731	70.93	80.32	**82.23**	0.7535
CoCat	59.15	71.85	76.78	0.7651	61.42	73.08	78.33	0.7933
OMWA	57.14	72.32	80.21	0.7389	64.42	72.91	77.93	0.7115
basic P2C	71.31	89.12	90.17	0.8845	70.5	79.8	80.1	0.8301
Simple C+ P2C	61.28	71.88	73.74	0.7963	60.87	71.23	75.33	0.7883
Gated C+ P2C	**73.89**	**90.14**	**91.22**	**0.8935**	**70.98**	**80.79**	81.37	**0.8407**

Table 3: Comparison with previous state-of-the-art models.

not surprise that straightforward concatenation strategy for source inputs performs poorly when the input pinyin is incomplete in DC corpus, due to obvious noise in too long context. The relatively small gap between the results of CoCat⋆ and CoCat indicate that statistical learning model may be helpful in obtaining some useful patterns from limited input. When the input statement contains adequacy information, the MIU accuracy of Gated C+ P2C system achieves more than 20% improvement in both corpora. However, we find that the KySS scores are much more close even with different pinyin integrity, which indicates that user experience in terms of KySS are more hard improved.

Instance Analysis We input a dialogue in wonder to how much of the contextual information is used when P2C module find the input pinyin is unknown. Figure 2 demonstrates the effect of the gated attention mechanism on candidates offering and unknown word replacement. As shown in Figure 2(a), we find that our IME suggests a more suitable candidates to the user when user is obviously not consistent with what the model has learned previously, which shows our model exceeds the Simple C+ P2C learning for maximally matching the inputted pinyin, but become capable of effectively resisting user pinyin input noise, and turns to learn potential language knowledge in previous input history[3].

As the ability predict user input from incomplete pinyin cannot be covered by any current IME performance metrics, thus the reported results yielded by our model actually underestimate our model performance to some extent. We illustrate the empirical discoveries of Figure 2(b) to demonstrate the extra effect of our P2C system on such situation, which indicates that the gated-attention pattern has taken great advantage of contextual information when given an unknown word. Or, namely, our model enables the incomplete input prediction though has to let it outside the current IME performance measurement. We display the attention visualization of Figure 2(b) in Figure 3 for better reference to explain the effect extended context plays on the generation of target characters.

Main Result Our model is compared to other models in Table 3. So far, (Huang et al., 2015) and (Zhang et al., 2017) reported the state-of-the-art results among statistical models. We list the top-5 accuracy contrast to all baselines with top-10 results, and the comparison indicates the noticeable advancement of our P2C model. To our surprise, the top-5 result on PD of our best Gated C+ P2C system approaches the top-10 accuracy of Google IME. On DC corpus, our Gated C+ P2C model with the best setting achieves 90.14% accuracy, surpassing all the baselines. The comparison shows our gated-attention system outperforms all state-of-the-art baselines with better user experience.

4 Conclusion

For the first time, this work makes an attempt to introduce additional context in neural pinyin-to-character converter for pinyin-based Chinese IME as to our best knowledge. We propose a gated-attention enhanced model for digging significant context information to improve conversion quality. More importantly, the resulting IME supports incomplete user pinyin input but returns complete, extra and even corrected character outputs, which brings about a story-telling mode change for all existing IMEs.

---

[3] Note as we evaluate our model only on two available corpora, but not the real world case from true user inputting history, which makes the instance situation limit to the domain feature of the given corpora.

References

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *CoLING*, pages 571–583.

Deng Cai, Hai Zhao, xin Yang, Wang Yuzhu, and Jia Zhongye. 2017a. A hybrid model for Chinese spelling check. In *ACM Transactions on Asian Low-Resource Language Information Process*.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjiang Wu, and Feiyue Huang. 2017b. Fast and accurate neural word segmentation for Chinese. In *ACL*, pages 608–615.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntax-agnostic or syntax-aware? In *CoLING*, pages 2753–2765.

Shenyuan Chen, Rui Wang, and Hai Zhao. 2015. Neural network language model for Chinese pinyin input method engine. In *PACLIC-29*, pages 455–461.

Stanley F. Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Eurospeech 2003 - INTERSPEECH 2003*, pages 2033–2036.

Zheng Chen and Kai Fu Lee. 2000. A new statistical approach to Chinese pinyin input. In *ACL*, pages 241–247.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1832–1846.

Shexia He, Zuchao Li, Hai Zhao, Hongxiao Bai, and Gongshen Liu. 2018. Syntax for semantic role labeling, to be, or not to be. In *ACL*, pages 2061–2071.

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

Guoping Huang, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2015. A new input method for human translators: integrating machine translation effectively and imperceptibly. In *IJCAI*, pages 1163–1169.

Yafang Huang, Zuchao Li, Zhuosheng Zhang, and Hai Zhao. 2018. Moon ime: Neural-based Chinese pinyin aided input method with customizable association. In *ACL: System Demonstrations*, pages 140–145.

Zhongye Jia, Peilu Wang, and Hai Zhao. 2013. Graph model for Chinese spell checking. In *SIGHAN-7*, pages 88–92.

Zhongye Jia and Hai Zhao. 2013. Kyss 1.0: a framework for automatic evaluation of Chinese input method engines. In *IJCNLP*, pages 1195–1201.

Zhongye Jia and Hai Zhao. 2014. A joint graph model for pinyin-to-Chinese conversion with typo correction. In *ACL*, pages 1512–1523.

Wei Jiang, Yi Guan, Xiao Long Wang, and Bing Quan Liu. 2007. Pinyin to character conversion model based on support vector machines. *Journal of Chinese Information Processing*, 21(2):100–105.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *CoLING*, pages 3203–3214.

Bo Lin and Jun Zhang. 2008. A novel statistical Chinese language model and its application in pinyin-to-character conversion. In *ACM Conference on Information and Knowledge Management*, pages 1433–1434.

Minh Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Chenxi Pang, Hai Zhao, and Zhongyi Li. 2016. I can guess what you mean: A monolingual query enhancement for machine translation. In *CCL*, volume 10035, pages 50–63.

Lianhui Qin, Lemao Liu, Victoria Bi, Yan Wang, Xiaojiang Liu, Zhiting Hu, Hai Zhao, and Shuming Shi. 2018. Automatic article commenting: the task and dataset. In *ACL*, pages 151–156.

Jorg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92.

Yingting Wu and Hai Zhao. 2018. Finding better subword segmentation for neural machine translation. In *CCL*.

Yu Wu, Wei Wu, Chen Xing, Zhoujun Li, and Ming Zhou. 2017. Lsequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *ACL*, pages 496–505.

Shaohua Yang, Hai Zhao, and Bao-liang Lu. 2012. A machine translation approach for Chinese whole-sentence pinyin-to-character conversion. In *Proceedings of the 26th Asian Pacific conference on language and information and computation*, pages 333–342.

Shenyuan Chen, Rui Wang, and Hai Zhao. 2015. Neural network language model for Chinese pinyin input method engine. In *PACLIC-29*, pages 455–461.

Xihu Zhang, Chu Wei, and Hai Zhao. 2017. Tracing a loose wordhood for Chinese input method engine. *arXiv preprint arXiv:1712.04158*.

Yan Zhang, Bo Xu, and Chengqing Zong. 2006. Rule-based post-processing of pin to Chinese characters conversion system. In *International Symposium on Chinese Spoken Language Processing*, pages 1–4.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2018a. Subword-augmented embedding for cloze reading comprehension. In *CoLING*, pages 3740–3752.

Zhuosheng Zhang, Yafang Huang, Pengfei Zhu, and Hai Zhao. 2018b. Effective character-augmentedword embedding for machine reading comprehension. In *NLPCC*.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, and Hai Zhao. 2018c. Modeling multi-turn conversation with deep utterance aggregation. In *CoLING*, pages 1802–1814.

Zhuosheng Zhang and Hai Zhao. 2018. One-shot learning for question-answering in gaokao history challenge. In *CoLING*, pages 449–4612.

Pengfei Zhu, Zhuosheng Zhang, Jiangtong Li, Yafang Huang, and Hai Zhao. 2018. Lingke: A fine-grained multi-turn chatbot for customer service. In *CoLING: System Demonstrations*, pages 108–112.

Appendices

A Architecture of the Evaluation Models

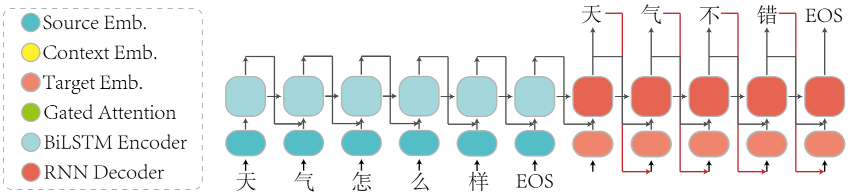

(a) Basic C2C Model: basic context-to-character model, straightforwardly generate the response candidate list from the context utterance.

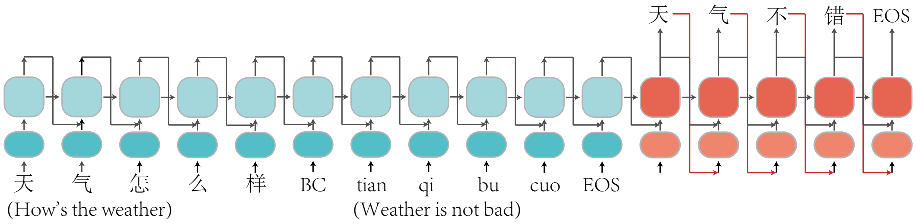

(b)Simple C+ P2C Model: Simple context-enhanced pinyin-to-character model, straightforwardly concat the context to the source pinyin sequence.

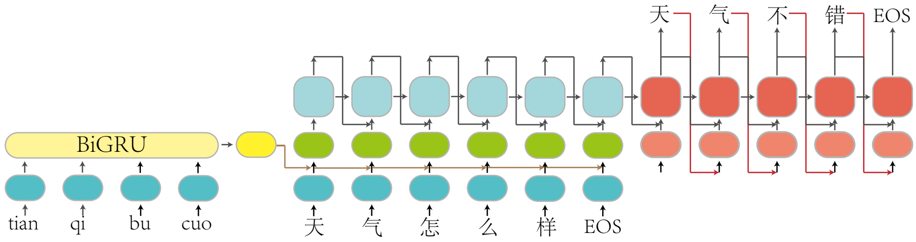

(c) Gated P+ C2C Model: Gated Attention based Pinyin-enhanced Context-to-Character model.

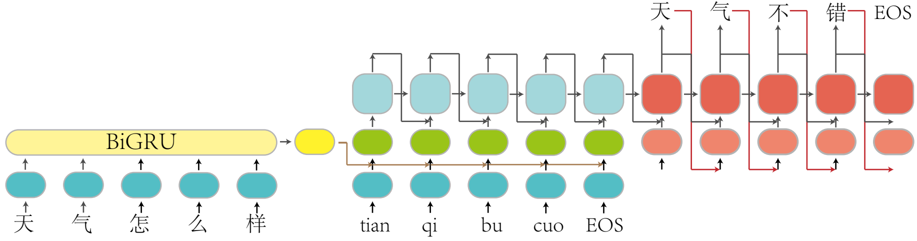

(d) Gated C+ P2C Model: Gated Attention based Context-enhanced Pinyin-to-Character model.

Estimating Marginal Probabilities of n-grams for Recurrent Neural Language Models

Thanapon Noraset[*]
Mahidol University
Nakhon Pathom, Thailand

Doug Downey[†]
Northwestern University
Evanston, IL, USA

Lidong Bing[‡]
Tencent AI Lab
Shenzhen, China

Abstract

Recurrent neural network language models (RNNLMs) are the current standard-bearer for statistical language modeling. However, RNNLMs only estimate probabilities for complete sequences of text, whereas some applications require context-independent phrase probabilities instead. In this paper, we study how to compute an RNNLM's *marginal* probability: the probability that the model assigns to a short sequence of text when the preceding context is not known. We introduce a simple method of altering the RNNLM training to make the model more accurate at marginal estimation. Our experiments demonstrate that the technique is effective compared to baselines including the traditional RNNLM probability and an importance sampling approach. Finally, we show how we can use the marginal estimation to improve an RNNLM by training the marginals to match n-gram probabilities from a larger corpus.

1 Introduction

Recurrent neural networks (RNNs) are the state-of-the-art architecture for statistical language modeling (Jozefowicz et al., 2016; Melis et al., 2018), the task of assigning a probability distribution to a sequence of words. The relative likelihoods of the sequences are useful in applications such as speech recognition, machine translation, automated conversation, and summarization (Mikolov et al., 2010; Bahdanau et al., 2014; See et al., 2017; Wen et al., 2017). Typically, RNN language models (RNNLMs) are trained on complete sequences (e.g., a sentence or an utterance), or long sequences (e.g. several documents), and used in the same fashion in applications or testing. A question arises when we want to compute the probability of a short sequence without the preceding context. For instance, we may wish to query for how likely the RNNLM is to generate a particular phase aggregated over *all contexts*. We refer to this context-independent probability of a short phrase as a *marginal probability*, or marginal.

These marginal probabilities are useful in three board categories of applications. First, they allow us to inspect the behavior of a given RNNLM. We could check, for example, whether an RNNLM-based generator might ever output a given offensive phrase. Second, the marginals could be used in phrase-based information extraction, such as extracting cities by finding high-probability x's in the phrase "cities such as x" (Soderland et al., 2004; Bhagavatula et al., 2014). Finally, we can use the phrase probabilities to train an RNNLM itself, e.g. updating the RNNLM according to n-gram statistics instead of running text (Chelba et al., 2017; Noraset et al., 2018). In our experiments, we show an example of the last application.

Estimating marginals from an RNNLM is challenging. Unlike an n-gram language model (Chen and Goodman, 1996), an RNNLM does not explicitly store marginal probabilities as its parameters. Instead, previous words are recurrently combined with the RNN's hidden state to produce a new state, which is used to compute a probability distribution of the next word (Elman, 1990; Mikolov et al., 2010). When the preceding context is absent, however, the starting state is also missing. In order to compute the marginal probability, in principle we must marginalize over all possible previous contexts or all continuous-vector states. Both options pose a severe computational challenge.

In this paper, we study how to efficiently approximate marginal probabilities from an RNNLM, without generating a large amount of text. Given an RNNLM and a phrase, our goal is

---

[*] `thanapon.nor@mahidol.edu`
[†] `d-downey@northwestern.edu`
[‡] `lyndonbig@tencent.com`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2930–2935
Brussels, Belgium, October 31 - November 4, 2018.

to estimate how frequently the phrase will occur in text generated by the RNNLM. We present two approaches that can be used to estimate the marginal probabilities: sampling for the starting state, and using a single starting state with altered RNNLM training. We show empirically that we can use a zero vector as a starting state of an RNNLM to compute accurate marginal estimates, but we must randomly reset the RNNLM state to zero during training and add a unigram likelihood term to the RNNLM training objective. Finally, we demonstrate that we can use marginal estimation to incorporate n-gram statistics from a larger corpus to improve the perplexity of an RNNLM trained on a similar, but smaller corpus.

2 Marginal Estimation

The goal of marginal estimation is to determine the likelihood of a short phrase where the preceding context is not known; we refer to this likelihood as a *marginal probability*. In other words, the marginal probability of a query refers to how likely a language model will generate a query regardless of context.

2.1 Problem settings

An RNNLM (Mikolov et al., 2010) defines a probability distribution over words conditioned on previous words as the following:

$$\begin{aligned} P(w_{1:T}) &= \prod_{t=1}^{T} P(w_t | w_{1:t-1}) \\ P(w_t | w_{1:t-1}) &= P(w_t | h_t) \propto \exp(\theta_o^{(w)} h_t) \\ h_t &= g(h_{t-1}, w_{t-1}) \end{aligned}$$

where $w_{1:t-1}$ is a sequence of previous words, θ_o^w denotes the output weights of a word w, and $g(\cdot)$ is a recurrent function such as an LSTM (Hochreiter and Schmidhuber, 1997) or GRU unit (Cho et al., 2014).

An initial state, h_1 is needed to start the recurrent function $g(h_1, w_1)$, and also defines the probability distribution of the first word $P(w_1|h_1)$. In the standard language model setting, we compute h_1 using a start-of-sentence symbol for w_0 ("`<s>`"), and a special starting state h_0 (usually set to be a vector of zeros $\vec{0}$). This initialization approach works fine for long sequences, because it is only utilized once and its effect is quickly swamped by the recurrent steps of the network. However, it is not effective for estimating marginal probabilities of a short phrase. For example, if we naively apply the standard approach to compute the probability of the phrase "`of the`", we would obtain:

$$\begin{aligned} P(\texttt{of the}) = &P(\texttt{of} | h_1 = g(\vec{0}, \texttt{<s>})) \times \\ &P(\texttt{the} | h_2 = g(h_1, \texttt{the})) \end{aligned}$$

The initial setting of the network results in low likelihoods of the first few tokens in the evaluation. For instance, the probability $P(\texttt{of the})$ computed in the above fashion will likely be a bad underestimate, because "`of the`" does not usually start a sentence.

We would like to compute the likelihood of standalone phrases, where rather than assuming the starting state we instead marginalize out the preceding context. Let the RNN's state prior to our query sequence be $z \in \mathbb{R}^d$, a vector-valued random variable representing the RNN initial state, and let $w_{1:T}$ be a short sequence of text. The marginal probability is defined as:

$$P(w_{1:T}) = \int P(w_{1:T}|z) P(z) dz \qquad (1)$$

The integral form of the marginal probability is intractable and requires an unknown density estimator of the state, $P(z)$.

2.2 Trace-based approaches

The integral form of the marginal probability in Eq 1 can be approximated by sampling for z. In this approach, we assume that there is a source of samples which asymptotically approaches the true distribution of the RNN states as the number of samples grows. In this work, we use a collection of RNN states generated in an evaluation, called a *trace*.

Given a corpus of text, a trace of an RNNLM is the corresponding list of RNN states, $H^{(tr)} = (h_1^{(tr)}, h_2^{(tr)}, ..., h_M^{(tr)})$, produced when evaluating the corpus. We can estimate the marginal probability by sampling the initial state z from H as follows:

$$P(w_{1:T}) = \mathbb{E}_{z \sim H^{(tr)}} \left[P(w_1|z) \prod_{t=2}^{T} p(w_t|h_t) \right] \quad (2)$$

where $h_2 = g(z_\psi, w_1)$ and $h_t = g(h_{t-1}, w_{t-1})$ for $t > 2$ (i.e. the following states are the deterministic output of the RNN function). Given a large trace this may produce accurate estimates, but it is intractably expensive and also wasteful, since in general there are very few states in the trace that yield a high likelihood for a sequence.

To reduce the number of times we run the model on the query, we use importance sampling over the trace. We train an encoder to output for a given n-gram query a state "near" the starting state(s) of the query, $z_\chi = q_\chi(w_{1:T})$. We define a sampling weight for a state in the trace, $h^{(tr)}$, proportional to the dot product of the state and the output of the encoder, z_χ , as the following:

$$P(h^{(tr)}|w_{1:T}) = \frac{exp(z_\chi h^{(tr)})}{\sum\limits_{h'^{(tr)}\in H^{(tr)}} exp(z_\chi h'^{(tr)})}$$

This distribution is biased to toward states that are likely to precede the query $w_{1:T}$. We can estimate the marginal probability as the following:

$$P(w_{1:T}) = \mathop{\mathbb{E}}_{z\sim P(h^{(tr)}|w_{1:T})}\left[\frac{P(z)}{P(z|w_{1:T})}P(w_1|z)\prod_{t=2}^{T} p(w_t|h_t)\right]$$

Here the choice of the prior $P(z)$ is a uniform distribution over the states in the trace. The encoder, $q_\chi(w_{1:T})$, is a trained RNN with its input reversed, and z_χ is the final output state of q_χ. To train the encoder, we randomly draw sub-strings $w_{i:i+n}$ of random length from the text used to produce the trace, and minimize the mean-squared difference between z_χ and $h_i^{(tr)}$.

2.3 Fixed-point approaches

While the trace-based approaches work on an existing (already trained) RNNLM, they might take several samples to accurately estimate the marginal probability. We would like to have a single point as the starting state, named z_ψ. We can either train this vector or simply set it to a zero vector. Then the marginal probability in Eq 1 can be estimated with a single run i.e. $p(z_\psi) = 1.0$ and $p(z) = 0.0$ if $z \neq z_\psi$. The computation is reduced to:

$$P(w_{1:T}) = P(w_1|z_\psi)\prod_{t=2}^{T} P(w_t|h_t) \qquad (3)$$

where $h_2 = g(z_\psi, w_1)$ and the rest of the state process is as usual, $h_t = g(h_{t-1}, w_{t-1})$. In this paper, we set z_ψ to be a zero vector, and call this method *Zero*.

As we previously discussed, our fixed-point state, z_ψ, is not a suitable starting state of all n-grams for any given RNNLM, so we need to train an RNNLM to adapt to this state. To achieve this, we use a slight modification of the RNN's truncated back-propagation through time training algorithm. We randomly *reset* the states to z_ψ when computing a new state during the training of the RNNLM (a similar reset was used for a different purpose—regularization—in Melis et al. (2018)). This implies that z_ψ is trained to maximize the likelihood of different subsequent texts of different lengths, and thus is an approximately good starting point for any sequence. Specifically, a new state is computed during the training as follows:

$$h_t = rz_\psi + (1-r)g(h_{t-1}, w_{t-1})$$

where $r \sim Bern(\rho)$ and ρ is a hyper-parameter for the probability of resetting a state. Larger ρ means more training with z_ψ, but it could disrupt the long-term dependency information captured in the state. We keep ρ relatively small at 0.05.

In addition to the *state reset*, we introduce a *unigram* regularization to improve the accuracy of the marginal estimation. From Eq 3, z_ψ is used to predict the probability distribution of the first token, which should be the unigram distribution. To get this desired behavior, we employ regularization to maximize the likelihood of each token in the training data *independently* (as if reset every step). We call this a unigram regularizer:

$$\mathcal{L}_U = -\sum_{t=1}^{T} \text{log}P(w_t|z_\psi)$$

and we add it to the training objective: $\mathcal{L}_{text} = -\sum_{t=1}^{T} \text{log}P(w_t|h_t)$. Thus, the overall training loss is: $\mathcal{L} = \mathcal{L}_{text} + \mathcal{L}_U$.

3 Experiments and Results

3.1 Experimental Settings

We experiment with a standard medium-size LSTM language model (Zaremba et al., 2014) over 2 datasets: Penn Treebank (PTB) (Mikolov et al., 2010) and WikiText-2 (WT-2) (Merity et al., 2017). We use weight tying (Inan et al., 2017) and train all models with Adam (Kingma and Ba, 2014) for 40 epochs with learning rate starting from 0.003 and decaying every epoch at the rate of 0.85. We use a batch size of 64 and truncated backpropagation with 35 time steps, and

	PTB		WT-2	
	$E(\cdot)$	PPL	$E(\cdot)$	PPL
Zero	3.828	90.18	4.432	104.08
Zero$^{(RU)}$	**0.425**	91.92	**0.801**	106.23
Trace-IW	0.661	-	0.862	-
Zero$^{(R)}$	2.968	93.70	3.519	109.02
Zero$^{(U)}$	1.007	**90.15**	1.713	**102.56**
Trace-Rand	1.105	-	1.742	-
n-grams	26,070		47,130	

Table 1: The average error of different marginal estimation approaches and the testing perplexity. The RNNLM trained with *state-reset (R)* and *unigram (U)* regularization has the lowest error when using a zero starting state to estimate the marginals.

employ variational dropout (Gal and Ghahramani, 2016). The final parameter set is chosen as the one minimizing validation loss. For the query model $q_X(w_{1:T})$ used in importance sampling, we use the same settings for the model and the training procedure as the above.

3.2 Marginal Estimation

In this subsection, we evaluate the accuracy of each approach at estimating marginal probabilities. Given a model and a phrase, we first obtain a target marginal probability, $P_{text}(w_{1:T})$, from a frequency of the phrase occurring in a text generated by the model. Then, we use each approach to estimate the marginal probability of the phrase, $P_{est}(w_{1:T})$. To measure the performance, we compute the absolute value of the log ratio (lower is better) between the target marginal probability and the estimated marginal probability (P_{est}):

$$E(w_{1:T}) = \left|\log(P_{text}(w_{1:T})/P_{est}(w_{1:T}))\right| \quad (4)$$

This evaluation measure gives equal importance to every n-gram regardless of its frequency.

In the following experiments, we generate approximately 2 million and 4 million tokens for PTB and WT-2 models respectively. The probability of phrases occurring in the generated text serves as our target marginal. We form a test set consisting of all n-grams in the generated text for $n \leq 5$ words, excluding n-grams with frequency less than 20 to reduce noise from the generation. For the trace-based estimations, we average the marginal probabilities from 100 samples.

Table 1 shows the average discrepancy between marginal probabilities estimated by generated-text statistics and the methods discussed in Section 2 (Eq 4). From the table, the RNNLM trained with the *state-reset* and the *unigram* regularization (*Zero*$^{(RU)}$) performs better than both zero-start and trace-based approaches on the traditional model. The importance sampling method (*Trace-IW*) has the second lowest error and performs better than random sampling (*Trace-Rand*). Ablation analysis shows that both *state-reset* and the *unigram* regularization contribute to the accuracy. Note that the trace-based methods use the same model as *Zero*.

	1	2	3	4	5
Zero	1.02	4.18	6.15	8.78	10.8
Zero$^{(RU)}$	**0.38**	**0.72**	0.95	1.54	2.10
Trace-IW	0.70	0.81	**0.92**	**1.22**	**1.48**
n-grams	10.7	20.7	10.5	3.7	1.5

Table 2: The error aggregate by n-gram lengths. This shows the same trend as in Table 1, but *Trace-IW* performs better for longer n-grams.

To show how performance varies depending on the query, we present results aggregated by n-gram lengths. Table 2 shows the errors of the WT-2 dataset. When the n-gram length is greater than 2, *Trace-IW* has better accuracy. This makes sense because the encoder has more evidence to use when inferring the likely start state.

3.3 Training with marginal probabilities

We now turn to an application of the marginal estimation. One way that we can apply our marginal estimation techniques is to train an RNNLM with n-gram probabilities in addition to running text. This is helpful when we want to efficiently incorporate data from a much larger corpus without training the RNNLM on it directly (Chelba et al., 2017; Noraset et al., 2018). In this work, we frame the problem as a regression and use a loss equal to the squared difference of log probabilities:

$$\mathcal{L}_N = \frac{\alpha}{2K}\sum_{k}^{K}(\log P_{text}(x_{1:T}^{(k)}) - \log P_{est}(x_{1:T}^{(k)}))^2$$

where α is a hyper-parameter and set to 0.1. Following the result in Table 1, we use the *Zero* method to estimate $P_{est}(x_{1:T}^{(k)})$ as in Eq 3, and add $\mathcal{L}_N$ to the training losses that use the running text corpus.

To evaluate the approach, we follow the Noraset et al. (2018) experiment in which bi-gram statistics from the training text of WT-103 are used to

Loss	PPL
$\mathcal{L}_{text}^{(2)}\ (Zero)$	104.08
$\mathcal{L}_{text}^{(2)} + \mathcal{L}_{U}^{(2)}\ (Zero^{(U)})$	102.56
$\mathcal{L}_{text}^{(2)} + \mathcal{L}_{U}^{(2)} + \mathcal{L}_{N}^{(103)}$	**93.95**

Table 3: Test perplexities of RNNLMs trained on different loss functions. Using n-gram probabilities from a larger corpus (WT-103) improves perplexities.

improve an RNNLM trained using WT-2. In our experiment, we use n-grams up to $n = 5$ with frequency greater than 50. We ignore n-gram containing `<unk>`, because the vocabulary sets are different. Table 3 shows the result. Since we do not use the same setting as in the original work, we cannot directly compare to that work – they use different optimization settings, more expensive n-gram loss, and Kneser-Ney bi-gram language model. However, we see that the proposed n-gram loss is beneficial when combined with the unigram loss. Importantly, unlike the approach in Noraset et al. (2018), our approach requires no sampling which makes it several times faster.

In addition, we present our preliminary result comparing training with the marginal probability of n-grams to training with the complete data. Given a limited budget of optimization steps, we ask whether training on n-grams is more valuable than training on the full corpus. To keep the results compatible, we use the vocabulary set of WikiText-2 and convert all OOV tokens in the training data of WikiText-103 to the "`<unk>`" token. Figure 1 shows the loss (average negative log-likelihood) of the validation data as the number of optimization steps increases.

We can see that training with the marginals does not perform as well as training with WikiText-103 training data, but outperforms the model trained only with WikiText-2 training data. This might be due to our choice of n-grams and optimization settings such as a number of n-grams per batch, weight of the n-gram loss, and the learning rate decay rate. We leave exploring these hyperparameters as an item of future work.

4 Conclusion

We investigated how to estimate marginal probabilities of n-grams from an RNNLM, when the preceding context is absent. We presented a simple method to train an RNNLM in which we occasionally reset the RNN's state and also maximize

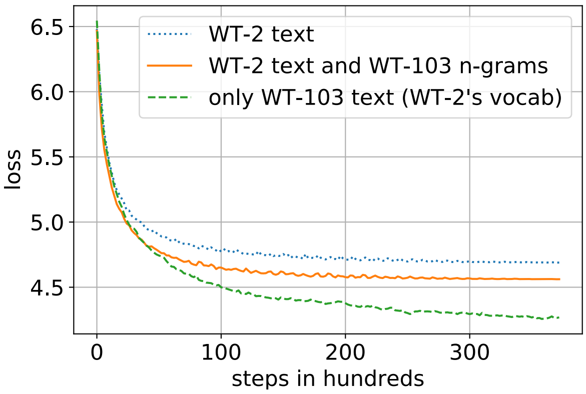

Figure 1: Loss in negative log-likelihood over steps in training. The loss computed using the valid data from WikiText-2 corpus. Training with n-grams from a larger corpus is helpful, but not as well as training with the running text from a larger corpus itself.

unigram likelihood along with the traditional objective. Our experiments showed that an RNNLM trained with our method outperformed other baselines on the marginal estimation task. Finally, we showed how to improve RNNLM perplexity by efficiently using additional n-gram probabilities from a larger corpus.

For future work, we would like to evaluate our approaches in more applications. For example, we can use the marginal statistics for information extraction, or to detect and remove abnormal phrases in text generation. In addition, we would like to continue improving the marginal estimation by experimenting with recent density estimation techniques such as NADE (Uria et al., 2016).

Acknowledgements

This work was supported in part by NSF grant IIS-1351029, the Tencent AI Lab Rhino-Bird Gift Fund, and Faculty of ICT, Mahidol University. We thank the reviewers for their valuable input.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR2014)*. ArXiv e-Prints. 1409.0473.

Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2014. Textjoiner: On-demand information extraction with multi-pattern queries. In *4th Workshop on Automated Knowledge Base Construction at NIPS 2014*. AKBC.

Ciprian Chelba, Mohammad Norouzi, and Samy Bengio. 2017. N-gram Language Modeling using Recurrent Neural Network Estimation. *ArXiv e-prints*, arXiv:1703.10724v2 [cs.CL].

Stanley F. Chen and Joshua Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc.

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

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. In *International Conference on Learning Representations (ICLR2017)*. ArXiv e-Prints. 1611.01462.

Refal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the Limits of Language Modeling. *ArXiv e-prints*, arXiv:1602.02410v2 [cs.CL].

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*.

Gbor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations (ICLR2018)*. ArXiv e-Prints. 1707.05589.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *International Conference on Learning Representations (ICLR2017)*. ArXiv e-Prints. 1609.07843.

Tom Mikolov, Martin Karafit, Luk Burget, Jan ernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, pages 1045–1048.

Thanapon Noraset, David Demeter, and Doug Downey. 2018. Controlling global statistics in recurrent neural network text generation. In *The Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Stephen Soderland, Oren Etzioni, Tal Shaked, and Daniel Weld. 2004. The use of web-based statistics to validate information extraction. In *AAAI-04 Workshop on Adaptive Text Extraction and Mining*, pages 21–26.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. 2016. Neural autoregressive distribution estimation. *J. Mach. Learn. Res.*, 17(1):7184–7220.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3732–3741, International Convention Centre, Sydney, Australia. PMLR.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent Neural Network Regularization. *ArXiv e-prints*, arXiv:1409.2329v5 [cs.NE].

How to represent a word and predict it, too: Improving tied architectures for language modelling

Kristina Gulordava **Laura Aina** **Gemma Boleda**
Universitat Pompeu Fabra
Barcelona, Spain
{firstname.lastname}@upf.edu

Abstract

Recent state-of-the-art neural language models share the representations of words given by the input and output mappings. We propose a simple modification to these architectures that decouples the hidden state from the word embedding prediction. Our architecture leads to comparable or better results compared to previous tied models and models without tying, with a much smaller number of parameters. We also extend our proposal to word2vec models, showing that tying is appropriate for general word prediction tasks.

1 Introduction

In neural models, reusing representations of the same type of data (e.g., sentences or words) in different parts of the architecture can be a powerful way to aid learning: it reduces parameters, enabling more compact models and faster learning. Recurrent neural network (RNN) language models (Mikolov et al., 2010; Sundermeyer et al., 2012; Zaremba et al., 2014) have two word mappings: From the input word onto its embedding representation, and from the internal representation of the network (the hidden layer) to the weights for the prediction of the next word. In standard models, these representations are different. Recently, Inan et al. (2017) and Press and Wolf (2017) proposed to instead use a single word representation, *tying* the input and output mappings. Intuitively, both are representations of the same type of data (words), and information learnt when observing a word as input can be reused when predicting this word as output. Tied language models obtained better perplexity and better word similarity scores of embedding matrices while reducing the number of parameters. The models that achieve the latest state-of-the art results incorporate this technique (see, e.g., Merity et al., 2018).

However, note that, by tying the output mapping to the input mapping, the hidden layer of the network is optimised to match the representation of the predicted word. We suggest that this introduces a constraint that conflicts with the function of the hidden layer in language models to represent the previous context and transmit information to the next timestep. In this paper, we propose a minimal modification to tied LM architectures to address this issue: we add a linear transformation between the hidden layer and the word embedding prediction, partially decoupling the two. This has an important advantage. Standard tied architectures require the hidden layer to have the same dimensionality as the word embeddings. We lift this constraint in our architecture: by separating the hidden layer and the word mapping, we can choose a large hidden layer dimensionality while keeping the embedding dimensionality and, consequently, the size of the embedding matrix small.

In a set of experiments on LM, we show that our tied models achieve results similar to or better than models with standard or no tying, with much smaller embedding sizes and a reduction of 30-60% in the overall number of parameters. Notably, the word embeddings obtained with the modified model have a higher quality than double-sized embeddings obtained with standard tied models, as measured on word similarity.

We further extend this idea to word representation learning models (in particular, word2vec), which have a similar architecture and objective function to language models. For instance, the standard skipgram model (Mikolov et al., 2013) has two mappings, one for the context words and one for the target word. While tying these two matrices directly constraints learning too strongly, an additional linear mapping adds the sufficient capacity to learn embeddings of the same quality as the standard model using only half the parameters.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2936–2941
Brussels, Belgium, October 31 - November 4, 2018.

2 Tied LM architectures

2.1 Previous work

Equations (1) through (4) define a standard RNN language model (Mikolov et al., 2010):

$$\mathbf{\hat{x}_t} = \mathbf{x_t} E \tag{1}$$

$$\mathbf{h_t} = LSTM(\mathbf{\hat{x}_t}, \mathbf{h_{t-1}}) \tag{2}$$

$$\mathbf{o} = \mathbf{h_t} W^T \tag{3}$$

$$\mathbf{p} = softmax(\mathbf{o}), \tag{4}$$

where $\mathbf{x_t}$ is the one-hot encoding of the input word at time t, $\mathbf{\hat{x}_t}$ is its embedded representation and E is the embedding matrix (input mapping). We follow the majority of previous work in adopting an LSTM (Hochreiter and Schmidhuber, 1997) as the recurrent unit, as it was shown to outperform other recurrent architectures for LM (Jozefowicz et al., 2015). The hidden vector $\mathbf{h_t}$ is used as input to the LSTM for the next time step and also as input to a linear transformation W (output mapping) which produces the output weights. These weights are normalised into probability scores using the softmax function.

The tied models proposed by Inan et al. (2017) and Press and Wolf (2017) set W to be equal to E (Figure 1b). Note that since E is of size $|V| \times m$, where m is the embedding size, and W is of size $|V| \times n$, where n is the hidden state size, the hidden and embedding dimensions must be equal, that is, $m = n$.

Press and Wolf (2017) observe that the output matrix W represents an embedding matrix since two similar words with indices i and j are learnt to receive similar probabilities given a context and hence the rows W_i and W_j should also be similar. Indeed, they show that on some word similarity evaluation tasks the output matrix W outperforms significantly the input matrix E. Tying E and W makes the model share the representations for the input and output vocabularies.

Inan et al. (2017) suggest a theoretical motivation for the tying technique and derive it as an instance of a more general approach of augmenting the cross-entropy loss. They show that a loss that takes into account not only the target word (i.e., $\log p_i$ for cross-entropy) but the scores for all words in the vocabulary according to their similarity to the target (computed as dot-product of embeddings in E) improves performance on LM.

(a) Non-tied model

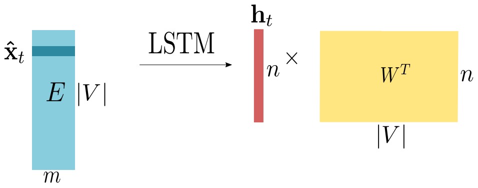

(b) Tied model

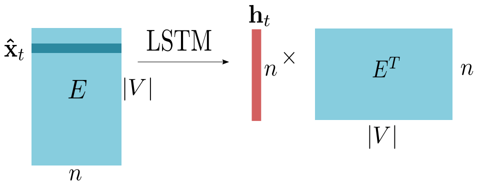

(c) Tied+L model

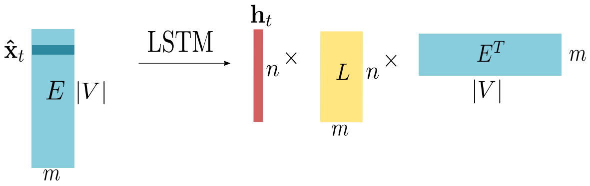

Figure 1: Effect of the three architectures on mapping sizes. Note that the actual difference between vocabulary size $|V|$ and n, m is of 2 to 3 orders of magnitude.

One important practical advantage of tying input and output matrices is the reduction in number of parameters with respect to a standard model with the same hidden and embedding dimensions, since instead of two matrices E and W of size $|V| \times m$ we have only one matrix of size $|V| \times m$.

2.2 Proposed modification

One potential problem with the tied model, where $E = W$, is that the hidden state $\mathbf{h_t}$ is optimised to be close to the embedding of the target word. To see this, consider that $o_j = \mathbf{e_j} \cdot \mathbf{h_t}, \forall j$. The cross-entropy objective is to maximise $\log p_i$ for the target word with index i and consequently to minimise $\log p_j \forall j \neq i$. Hence, $o_i = \mathbf{e_i} \cdot \mathbf{h_t}$ will be increased by the gradient descent update and $\mathbf{h_t}$ will be aligned closer with $\mathbf{e_i}$ (for each dimension k of the two vectors, $h_{tk} \cdot e_{ik}$ will be increased). This association between $\mathbf{h_t}$ and word embedding space could prevent efficient retention of LSTM history in $\mathbf{h_t}$, which is used as input for the following time step.

To address this issue, we propose a simple modification to the standard tied model, replacing the output transformation (3) as follows:

$$\begin{aligned}\hat{\mathbf{h}}_{\mathbf{t}} &= \mathbf{h}_{\mathbf{t}} L \\ \mathbf{o} &= \hat{\mathbf{h}}_{\mathbf{t}} E^T.\end{aligned} \tag{5}$$

L is an additional linear transformation that decouples the hidden state $\mathbf{h}_{\mathbf{t}}$, which is passed to the next time step and represents the previous, possibly long-term, linguistic context, from $\hat{\mathbf{h}}_{\mathbf{t}}$, which is instead optimised to match the embedding of the output word.

As Figure 1 illustrates, an important advantage of the additional transformation L is that a model can have different dimensions for the hidden vector (n) and the embedding vectors (m). The embedding matrix is the largest part of the model when the vocabulary is large ($|V| \gg n$). Reducing the size of the embedding m leads to a significant reduction in the number of parameters, proportional to $|V|$, and the acceleration of softmax computation. On the other hand, the size of the additional matrix L is only $n \times m$ and contributes very little to the overall size of the model. We test empirically how reducing the embedding size affects the performance of language models by varying hidden and embedding sizes in our experiments and evaluating embedding matrices on word similarity tasks.

Standardly used LM models often have two layers of LSTM cells. Thus, the issue we identifed might be mitigated in practice, since the hidden state of the first layer is not directly affected by tying the output and input transformation matrices. Moreover, an LSTM cell carries over information both through the hidden state and a memory state; the latter is affected by tying only indirectly (see Hochreiter and Schmidhuber (1997) for details on LSTM architectures). However, our experiments show that, in practice, two-layer LSTM LMs are still affected by tying despite these caveats.

2.3 Extending the tied technique to word2vec

The tied technique, as formulated above, can be in principle applied to any model which has the same general objective of LM: predicting a target word given context words. The CBOW word2vec model for word representation learning (Mikolov et al., 2013) is the primary candidate for testing the applicability of the tied technique beyond LM, since we can see it as substituting the LSTM function (2) with a simple sum of context word embeddings $\mathbf{h} = \sum_i \hat{\mathbf{x}}_{\mathbf{i}}$ (where x_i are words in the context window, e.g., of size 5). Similarly then, the equations in (5) describe the tied version of CBOW model. The linear step here provides the capacity to learn a transformation from the sum of embeddings to the predicted embedding $\hat{\mathbf{h}}$. Without such transformation, the tying model would assume that the sum function is always a good approximation of the output embedding.

The skipgram word2vec model (Mikolov et al., 2013) employs a variant of the LM objective: It is trained to predict context words given a word, instead of the opposite. As in CBOW and neural language models, words are both inputs and targets, making the use of tying an option also for this architecture. Press and Wolf (2017) apply direct tying to this architecture and report that the quality of the obtained embeddings is below the quality of non-tied skipgram embeddings. Unlike CBOW or LSTM, the input-to-hidden state function of the skipgram model is identity, reducing the tying model objective to $\hat{\mathbf{x}}_{\mathbf{i}} = \hat{\mathbf{x}}_{\mathbf{j}}$ for every pair of input-output words i, j. It is thus not surprising that enforcing the tying constraint leads to poor empirical results. We test whether adding an additional linear transformation improves performance of the tied technique also for a skipgram model.

3 Experiments and results

3.1 Evaluation data

We use two corpora for the evaluation of language models. First, we employ a medium-sized corpus of approximately 100M tokens with a relatively large vocabulary, 50K words, created from a Wikipedia dump (henceforth, Wiki).[1] To allow comparison with previous work, we also evaluate on the Penn Treebank (PTB), which is small but has been used as a benchmark for LM since Mikolov et al. (2011). The PTB has approximately 1M tokens and is preprocessed to have 10K vocabulary words; we use the standard train-validation-test split.

Furthermore, to evaluate the quality of the embeddings induced by the language models, as well as for the word representation experiments in Section 3.4, we use three standard word similarity datasets: SimLex-999 (Hill et al., 2015; SimLex), MEN (Bruni et al., 2014), and RareWords (Luong et al., 2013; RW). The performance on these datasets is evaluated in terms of Spearman corre-

---

[1] `https://dumps.wikimedia.org/enwiki/20180301/`

Hid	Emb	Model	Valid	Test	Δ	Size
200	200	non-tied	95.0	91.1		4.7M
		tied	90.8	86.6	-4.5	2.7M
		tied+L	89.8	85.8	-5.3	2.7M
400	200	non-tied	89.4	85.3		8.3M
		tied+L	83.4	80.3	-5.0	4.3M
	400	non-tied	87.2	83.5		10.6M
		tied	82.0	78.2	-5.3	6.6M
		tied+L	81.9	78.0	-5.5	6.7M
600	400	non-tied	85.8	82.4		15.3M
		tied+L	79.0	76.0	-6.4	9.5M
	600	non-tied	84.3	81.3		17.8M
		tied	79.7	76.1	-5.2	11.8M
		tied+L	78.7	75.5	-5.8	12.1M
Inan2017 VD tied 650			77.1	73.9		-
Zaremba2014 1500			82.2	78.4		66M
P&W2016 tied 1500			77.7	74.3		51M

Table 1: LM perplexity results on PTB. Δ: difference in test perplexity of the tied models with respect to the non-tied model with the same number of hidden units.

lation between the cosine similarity of word pairs and human judgments.[2]

3.2 Training setup

As our base language models, we adopt the ones proposed in Zaremba et al. (2014). We use 2-layer LSTMs with dropout applied to the input embedding, to the output of the first LSTM layer and to the output of the second layer. We used the PyTorch implementation[3] and modified it to include the additional linear layer for our tied models. We report the best model after the hyperparameter search for dropout and learning rate (see the details in Appendix A).

3.3 Language modelling results

We present the LM results for the standard non-tied model, the tied model as in Inan et al. (2017) and Press and Wolf (2017), and our tied model with an additional linear transformation (*tied+L*) in Tables 1 (PTB) and 2 (Wiki).

Table 1 confirms that tying generally brings gains with respect to not tying. This is also true for the cases when the hidden and embedding sizes are different (e.g. 400/200 and 600/400), where our tied+L model outperforms the non-tied model by 5 to 6.4 points having around 40% less parameters. Furthermore, our decoupled model slightly but consistently improves results with respect to standard tying, confirming our intuition that the coupling of the hidden state to the embedding representation is a limiting constraint. Smaller tied+L models perform well compared to larger tied models. In particular, the tied+L model with 600/400 units has perplexity of 76.0, compared to 76.1 of the tied 600/600 model, with 55% the number of parametres. Note that our results are comparable to previously reported perplexity values on PTB for similar models. Our best results of 75.5 test perplexity is only 1.2 points behind the large tied model with 1500 units reported in Press and Wolf (2017) and is only 1.6 points behind the medium tied model with 650 units and variational dropout (Gal and Ghahramani, 2016) reported in Inan et al. (2017).

On the Wiki corpus with larger vocabulary (Table 2), we find that tied models achieve slightly lower perplexity than non-tied models with half the number of parameters, and our proposed tied+L model achieves lower perplexity than the tied model. The most relevant result of the present experiment, however, is that the tied+L model with 300 embedding units is actually better than the tied model with 600 units (38.5 vs 39.7 points; the tied+L model has 20M parameters compared to 36M of the tied model) – that is, a smaller model outperforms a larger model. Thus, our decoupling mechanism not only allows models to have better perplexity, but also more compact word embeddings, which are of a higher quality also as measured on word similarity: .42/.61/.68 for the tied+L embeddings of size 300, compared to .39/.55/.64 for the tied embeddings of size 600.

3.4 Experiments on word2vec models

Table 3 presents the evaluation of word2vec models on the three word similarity datasets. We ran the experiments only on the Wiki corpus due to its higher coverage (50K vocabulary), and used embeddings of size 300.

Our results on CBOW show that the tied+L architecture obtains comparable results to the non-

---

[2] We computed correlation on the word pairs covered by the Wiki corpus, namely 98%, 88% and 31% (973, 2648 and 623 datapoints) for SimLex, MEN, and RW, respectively.

[3] `https://github.com/pytorch/examples/tree/master/word_language_model`

				Perplexity		Spearman's ρ		
Hidden	Emb	Model	Size	Valid	Test	SimLex	RW	MEN
600	300	non-tied	50M	40.0	39.2	.33/.34	.52/.50	.55/.60
		tied+L	20M	39.3	38.5	.42	.61	.68
	600	non-tied	66M	40.8	40.0	.33/.34	.52/.50	.55/.60
		tied	36M	40.5	39.7	.39	.55	.64
		tied+L	36M	38.6	37.9	.42	.59	.70

Table 2: Results on Wiki on LM (perplexity; lower is better) and word similarity (Spearman's ρ; for non-tied models, results for input and output matrix are reported).

Model	Type	SimLex	RW	MEN
CBOW	non-tied	.38	.51	.63
	tied+L	.38	.50	.65
skipgram	non-tied	.39	.52	.74
	tied	.18	.25	.50
	tied+L	.35	.51	.72

Table 3: Results (ρ) for word2vec models.

tied architecture with almost half the parameters (15.1M vs 30M). This confirms that tying with an additional linear transformation is appropriate not only for language models but for word learning models more generally.

The skipgram algorithm shows a small degradation of performance for the tied+L architecture with respect to the non-tied one; note that, as explained in Section 2.3, tying makes the most sense for CBOW. However, the fact that standard tying obtains much worse results (similarly to the results of Press and Wolf, 2017) shows that the linear mapping substantially relaxes the tying constraint.

4 Conclusions

Overall, our simple modification to tied language modelling architectures generalises previous work by allowing tying without imposing constraints on the number of hidden and embedding dimensions. This leads to flexible architectures with a more efficient use of both hidden states and embeddings. For word representation learning models, having an additional linear transformation reduces the number of parameters while maintaining learning capacity. In general, reducing model size without harming performance is a desirable feature in practice, for example in the case of language models running on mobile devices, and it is also desirable on theoretical grounds, since it is a better use of the learning capacity of neural networks.

Acknowledgements

We thank Germán Kruszewski and the AMORE team for the helpful discussions. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 715154), and from the Ramón y Cajal programme (grant RYC-2015-18907) and the Catalan government (SGR 2017 1575). We gratefully acknowledge the support of NVIDIA Corporation with the donation of GPUs used for this research. This paper reflects the authors' view only, and the EU is not responsible for any use that may be made of the information it contains.

References

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating Semantic Models with Genuine Similarity Estimation. *Comput. Linguist.*, 41(4):665–695.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9 8:1735–80.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In ICLR'17. arXiv preprint arXiv:1611.01462.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2342–2350. JMLR.org.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113. Association for Computational Linguistics.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In ICLR'18. arXiv preprint arXiv:1708.02182.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. pages 1–12. ArXiv preprint arXiv:1301.3781.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. Association for Computational Linguistics.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM Neural Networks for Language Modeling. In *INTERSPEECH*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization.

The Importance of Generation Order in Language Modeling

Nicolas Ford[*] **Daniel Duckworth** **Mohammad Norouzi** **George E. Dahl**
Google Brain
{nicf,duckworthd,mnorouzi,gdahl}@google.com

Abstract

Neural language models are a critical component of state-of-the-art systems for machine translation, summarization, audio transcription, and other tasks. These language models are almost universally autoregressive in nature, generating sentences one token at a time from left to right. This paper studies the influence of token generation order on model quality via a novel *two-pass language model* that produces partially-filled sentence "templates" and then fills in missing tokens. We compare various strategies for structuring these two passes and observe a surprisingly large variation in model quality. We find the most effective strategy generates *function* words in the first pass followed by *content* words in the second. We believe these experimental results justify a more extensive investigation of generation order for neural language models.

1 Introduction

Neural networks have been extremely successful statistical models of text in language modeling and machine translation. Despite differences in model architectures, state of the art neural nets generate sequences from left to right (Vaswani et al., 2017; Jozefowicz et al., 2016; Wu et al., 2016). Although in some sense humans produce and consume language from left to right as well, there are many other intuitively appealing ways to generate text. For instance, language is slow enough on a neurological time scale for multiple passes of generation that incorporate feedback to occur. Linguistic intuition might suggest that we should first generate some abstract representation of what we want to say and then serialize it, a process that seems more universally appropriate given the existence of languages with freer word order such as Czech and Polish.

There has been interest in moving beyond the left-to-right generation order by developing alternative multi-stage strategies such as syntax-aware neural language models (Bowman et al., 2016) and latent variable models of text (Wood et al., 2011). Before embarking on a long-term research program to find better generation strategies that improve modern neural networks, one needs evidence that the generation strategy can make a large difference. This paper presents one way of isolating the generation strategy from the general neural network design problem. Our key technical contribution involves developing a flexible and *tractable* architecture that incorporates different generation orders, while enabling *exact* computation of the log-probabilities of a sentence. Our experiments demonstrate that even when using a few simple two-pass generation orders, the differences between good and bad orderings are substantial.

We consider ways of reordering the tokens within a sequence based on their identities. The best ordering we tried generates function words first and content words last, which cuts against the idea of committing to the general topic of a sentence first and only then deciding exactly how to phrase it. We offer some possible explanations in Section 3, and we conclude that our experimental results justify a more extensive investigation of the generation order for language and translation models.

2 Two-pass Language Models

We develop a family of *two-pass language models* that depend on a partitioning of the vocabulary into a set of *first-pass* and *second-pass* tokens to generate sentences. We perform a preprocessing step on each sequence $\mathbf{y}$, creating two new sequences $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$. The sequence $\mathbf{y}^{(1)}$, which we call the *template*, has the same length as $\mathbf{y}$, and consists of the first-pass tokens from $\mathbf{y}$ together with a special *placeholder* token wherever

---

[*] Work done as a member of the Google AI Residency program (g.co/airesidency)

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2942–2946
Brussels, Belgium, October 31 - November 4, 2018.

sentence	common first	rare first	function first	content first	odd first
” all you need to do if you want the nation ’s press camped on your doorstep is to say you once had a [UNK] in 1947 , ” he noted memorably in his diary . [EOS]	” all you __ to __ if you __ the __ ’s __ __ on __ __ is to __ you __ had a [UNK] in __ , ” he __ __ in his __ . [EOS]	__ __ __ need __ do __ __ want __ nation __ press camped __ your doorstep __ __ say __ once __ __ __ __ 1947 __ __ __ noted memorably __ __ diary __ [EOS]	” all you __ to __ if you __ the __ ’s __ __ on your __ is to __ you __ __ a __ in __ , ” he __ __ in his __ . [EOS]	__ __ __ need __ do __ __ want __ nation __ press camped __ __ doorstep __ __ say __ once had __ [UNK] __ 1947 __ __ __ noted memorably __ __ diary __ [EOS]	” all you need __ __ __ you __ the nation ’s press camped on your doorstep __ __ say you once had __ __ __ __ __ ” __ noted __ __ his __ . [EOS]
the team announced thursday that the 6-foot-1 , [UNK] starter will remain in detroit through the 2013 season . [EOS]	the __ __ __ that the __ , [UNK] __ will __ in __ __ the __ __ . [EOS]	__ team announced thursday __ __ 6-foot-1 __ __ starter __ remain __ detroit through __ 2013 season __ [EOS]	the __ __ __ that the __ , __ __ will __ in __ through the __ __ . [EOS]	__ team announced thursday __ __ 6-foot-1 __ [UNK] starter __ remain __ detroit __ __ 2013 season __ [EOS]	the team announced __ __ the 6-foot-1 __ __ __ will remain __ __ through the 2013 __ . [EOS]
scotland ’s next game is a friendly against the czech republic at hampden on 3 march . [EOS]	__ ’s __ __ is a __ __ the __ __ at __ on __ __ . [EOS]	scotland __ next game __ __ friendly against __ czech republic __ hampden __ 3 march __ [EOS]	__ ’s __ __ is a __ against the __ __ at __ on __ __ . [EOS]	scotland __ next game __ __ friendly __ __ czech republic __ hampden __ 3 march __ [EOS]	__ ’s next game __ __ __ __ the czech republic at hampden on 3 march . [EOS]
of course , millions of additional homeowners did make a big mistake : they took advantage of ” liar loans ” and other [UNK] deals to buy homes they couldn ’t afford . [EOS]	of __ , __ of __ __ __ __ a __ __ : they __ __ of ” __ __ ” and __ [UNK] __ to __ __ they __ ’t __ . [EOS]	__ course __ millions __ additional homeowners did make __ big mistake __ __ took advantage __ __ liar loans __ __ other __ deals __ buy homes __ couldn __ afford __ [EOS]	of __ , __ of __ __ __ __ a __ __ : they __ __ of ” __ __ ” and __ __ __ to __ __ they __ __ __ . [EOS]	__ course __ millions __ additional homeowners did make __ big mistake __ __ took advantage __ __ liar loans __ __ other [UNK] deals __ buy homes __ couldn ’t afford __ [EOS]	of __ __ __ of additional __ __ __ __ big __ __ they __ advantage of ” liar __ ” and other __ deals __ buy homes they couldn __ afford . [EOS]

Table 1: Some example sentences from the dataset and their corresponding templates. The placeholder token is indicated by “__”.

$\mathbf{y}$ had a second-pass token. The sequence $\mathbf{y}^{(2)}$ has length equal to the number of these placeholders, and consists of the second-pass tokens from $\mathbf{y}$ in order.

We use a neural language model p_1 to generate $\mathbf{y}^{(1)}$, and then a conditional translation model p_2 to generate $\mathbf{y}^{(2)}$ given $\mathbf{y}^{(1)}$. Note that, since the division of the vocabulary into first- and second-pass tokens is decided in advance, there is a one-to-one correspondence between sequences $\mathbf{y}$ and pairs $(\mathbf{y}^{(1)}, \mathbf{y}^{(2)})$. The total probability of $\mathbf{y}$ is then

$$p(\mathbf{y}) \;=\; p_1(\mathbf{y}^{(1)})\, p_2(\mathbf{y}^{(2)} \mid \mathbf{y}^{(1)})\,. \qquad (1)$$

Two-pass language models present a unique opportunity to study the importance of generation order because, since the template is a deterministic function of $\mathbf{y}$, the probability of $\mathbf{y}$ can be computed exactly. This is in contrast to a language model using a *latent* generation order, which requires a prohibitive marginalization over permutations to compute the exact probabilities. Given the tractable nature of the model, exact learning based on log-likelihood is possible, and we can compare different vocabulary partitioning strategies both against each other and against a single-pass language model.

Our implementation consists of two copies of the Transformer model from Vaswani et al. (2017). The first copy just generates the template, so it has no encoder. The second copy is a sequence-to-sequence model that translates the template into the complete sentence. There are three places in this model where word embeddings appear — the first-phase decoder, the second-phase encoder, and the second-phase decoder — and all three sets of parameters are shared. The output layer also shares the embedding parameters.[1]

For the second pass, we include the entire target sentence, not just the second-pass tokens, on the output side. In this way, when generating a token, the decoder is allowed to examine all tokens to the left of its position. However, only the second-pass tokens count toward the loss, since in the other positions the correct token is already known. Our loss function is then the sum of all of these numbers (from both copies) divided by the length of the original sentence, which is the log-perplexity that our model assigns to the sentence.

We tried five different ways of splitting the vocabulary:

Common First and Rare First: The vocabulary was sorted by frequency and then a cutoff was chosen, splitting the vocabulary into “common” and “rare” tokens. The location of the cutoff[2] was chosen so that the number of common tokens and the number of rare tokens in the average sentence were approximately the same. In “common first”

---

[1] This behavior is enabled in the publicly available implementation of Transformer using the hyperparameter called `shared_embedding_and_softmax_weights`.

[2] In our experiments on LM1B, this is at index 78.

we place the common tokens in the first pass, and in "rare first" we start with the rare tokens.

Function First and Content First: We parsed about 1% of LM1B's training set using Parsey McParseface (Andor et al., 2016) and assigned each token in the vocabulary to the grammatical role it was assigned most frequently by the parser. We used this data to divide the vocabulary into "function" words and "content" words; punctuation, adpositions, conjunctions, determiners, pronouns, particles, modal verbs, "wh-adverbs" (Penn part-of-speech tag `WRB`), and conjugations of "be" were chosen to be function words. In "function first" we place the function words in the first phase and in "content first" we start with the content words.

Odd First: As a control, we also used a linguistically meaningless split where tokens at an odd index in the frequency-sorted vocabulary list were assigned to the first pass and tokens with an even index were assigned to the second pass.

A few sentences from the dataset are shown in Table 1 together with their templates. Note that the common and function tokens are very similar; the main differences are the "unknown" token, conjugations of "have," and some prepositions.

3 Experimental Results and Discussion

We ran experiments with several different ways of splitting the vocabulary into first-pass and second-pass tokens. We trained all of these models on the One Billion Word Language Modeling benchmark (LM1B) dataset (Chelba et al., 2013). One sixth of the training data was used as a validation set. We used a vocabulary of size 65,536 consisting of whole words (rather than word pieces) converted to lower-case.

We compared the two-pass generation strategies to a baseline version of Transformer without an encoder, which was trained to unconditionally predict the target sentences in the ordinary way. Because the two-pass models contain slightly more trainable parameters than this baseline, we also compare to an "enhanced baseline" in which the size of Transformer's hidden space was increased to make the number of parameters match the two-pass models.

Both the two-pass models and the baselines used the hyperparameters referred to as `base` in the publicly available implementation of Transformer,[3] which has a hidden size of 512, a filter size of 2048, and 8 attention heads, except that the enhanced baseline used a hidden size of 704. We used a batch size of 4096. All models were trained using ADAM (Kingma and Ba, 2014), with $\beta_1 = 0.85$, $\beta_2 = 0.997$, and $\epsilon = 10^{-6}$. The learning rate was tuned by hand separately for each experiment and the experiments that produced the best results on the validation set are reported. Dropout was disabled after some initial experimentation found it to be detrimental to the final validation loss.

Table 2 shows the results for all the two-pass generation strategies we tried as well as the baselines, sorted from worst to best on the validation set. Strikingly, the linguistically meaningless odd first generation strategy that splits words arbitrarily between the two phases is far worse than the baseline, showing that the two-pass setup on its own provides no inherent advantage over a single phase. The common first and closely related function first strategies perform the best of all the two-pass strategies, whereas the rare first and closely related content first strategies are much worse. Since the control, rare first, and content first orderings are all worse than the baseline, the gains seen by the other two orderings cannot be explained by the increase in the number of trainable parameters alone.

The enhanced version of the baseline achieved slightly better perplexity than the best of the two-pass models we trained. Given that state-of-the-art results with Transformer require models larger than the ones we trained, we should expect growing the embedding and hidden size to produce large benefits. However, the two-pass model we proposed in this work is primarily a tool to understand the importance of sequence generation order and was not designed to be parameter efficient. Thus, as these results indicate, increasing the embedding size in Transformer is a more effective use of trainable parameters than having extra copies of the other model parameters for the second pass (recall that the embeddings are shared across both passes).

One potential explanation for why the function first split performed the best is that, in order to generate a sentence, it is easier to first decide something about its syntactic structure. If this is the primary explanation for the observed results, then common first's success can be attributed to how many function words are also common. However, an alternative explanation might

---

[3] github.com/tensorflow/tensor2tensor

Model	Train	Validation	Test
odd first	39.925	45.377	45.196
rare first	38.283	43.293	43.077
content first	38.321	42.564	42.394
common first	36.525	41.018	40.895
function first	36.126	40.246	40.085
baseline	38.668	41.888	41.721
enhanced baseline	35.945	39.845	39.726

Table 2: The perplexities achieved by the best version of each of our models.

simply be that it is preferable to delay committing to a rare token for as long as possible as all subsequent decisions will then be conditioning on a low-probability event. This is particularly problematic in language modeling where datasets are too small to cover the space of all utterances. We lack sufficient evidence to decide between these hypotheses and believe further investigation is necessary.

Ultimately, our results show that content-dependent generation orders can have a surprisingly large effect on model quality. Moreover, the gaps between different generation strategies can be quite large.

4 Related Work

For tasks conditioning on sequences and sets, it is well known that order significantly affects model quality in applications such as machine translation (Sutskever et al., 2014), program synthesis (Vinyals et al., 2016), and text classification (Yogatama et al., 2016). Experimentally, Khandelwal et al. (2018) show that recurrent neural networks have a memory that degrades with time. Techniques such as attention (Bahdanau et al., 2014) can be seen as augmenting that memory.

Text generation via neural networks, as in language models and machine translation, proceeds almost universally left-to-right (Jozefowicz et al., 2016; Sutskever et al., 2014). This is in stark contrast to phrase-based machine translation systems (Charniak et al., 2003) which traditionally split token translation and "editing" (typically via reordering) into separate stages. This line of work is carried forward in Post-Editing Models (Junczys-Dowmunt and Grundkiewicz, 2016), Deliberation Networks (Xia et al., 2017), and Review Network (Yang et al., 2016) which produce a "draft" decoding that is further edited. As any valid sequence may be used in a draft, calculating perplexity in these models is unfortunately intractable, and model quality can only be evaluated via external tasks.

In addition to surface-form intermediate representation, syntax-based representations have a rich history in text modeling. Chelba and Jelinek (1998); Yamada and Knight (2001); Graham and Genabith (2010); Shen et al. (2018) integrate parse structures, explicitly designed or automatically learned, into the decoding process.

Similar to the second phase of this work's proposed model, (Fedus et al., 2018) directly tackles the problem of filling in the blank, akin to the second stage of our proposed model. The Multi-Scale version of PixelRNN in (Van Oord et al., 2016) was also an inspiration for the two-pass setup we used here.

5 Conclusion and Future Work

To investigate the question of generation order in language modeling, we proposed a model that generates a sentence in two passes, first generating tokens from left to right while skipping over some positions and then filling in the positions that it skipped. We found that the decision of which tokens to place in the first pass had a strong effect.

Given the success of our function word first generation procedure, we could imagine taking this idea beyond splitting the vocabulary. One could run a parser on each sentence and use the resulting tree to decide on the generation order. Such a scheme might shed light on which aspect of this split was most helpful. Finally, filling in a template with missing words is a task that might be interesting in its own right. One might want to provide partial information about the target sentence as part of scripting flexible responses for a dialogue agent, question answering system, or other system that mixes a hand-designed grammar with learned responses.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1466–1477.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of MT Summit IX*, pages 40–46.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 225–231. Association for Computational Linguistics.

Ciprian Chelba, Tomáš Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

William Fedus, Ian Goodfellow, and Andrew M. Dai. 2018. MaskGAN: Better text generation via filling in the ________. In *International Conference on Learning Representations (ICLR)*.

Yvette Graham and Josef Genabith. 2010. Deep syntax language models and statistical machine translation. In *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation*, pages 118–126.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. *arXiv preprint arXiv:1605.04800*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Yikang Shen, Zhouhan Lin, Chin-wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*.

Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. 2011. The sequence memoizer. *Communications of the ACM*, 54(2):91–98.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Advances in Neural Information Processing Systems*, pages 1782–1792.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

Zhilin Yang, Ye Yuan, Yuexin Wu, William W Cohen, and Ruslan R Salakhutdinov. 2016. Review networks for caption generation. In *Advances in Neural Information Processing Systems*, pages 2361–2369.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to compose words into sentences with reinforcement learning. *arXiv preprint arXiv:1611.09100*.

Document-Level Neural Machine Translation with Hierarchical Attention Networks

Lesly Miculicich[†] [‡] **Dhananjay Ram**[†] [‡] **Nikolaos Pappas**[†] **James Henderson**[†]
[†] Idiap Research Institute, Switzerland
[‡]École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
{lmiculicich, dram, npappas, jhenderson}@idiap.ch

Abstract

Neural Machine Translation (NMT) can be improved by including document-level contextual information. For this purpose, we propose a hierarchical attention model to capture the context in a structured and dynamic manner. The model is integrated in the original NMT architecture as another level of abstraction, conditioning on the NMT model's own previous hidden states. Experiments show that hierarchical attention significantly improves the BLEU score over a strong NMT baseline with the state-of-the-art in context-aware methods, and that both the encoder and decoder benefit from context in complementary ways.

1 Introduction

Neural machine translation (NMT) (Bahdanau et al., 2015; Wu et al., 2016; Vaswani et al., 2017) trains an encoder-decoder network on sentence pairs to maximize the likelihood of predicting a target-language sentence given the corresponding source-language sentence, without considering the document context. By ignoring discourse connections between sentences and other valuable contextual information, this simplification potentially degrades the coherence and cohesion of a translated document (Hardmeier, 2012; Meyer and Webber, 2013; Sim Smith, 2017). Recent studies (Tiedemann and Scherrer, 2017; Jean et al., 2017; Wang et al., 2017; Tu et al., 2018) have demonstrated that adding contextual information to the NMT model improves the general translation performance, and more importantly, improves the coherence and cohesion of the translated text (Bawden et al., 2018; Lapshinova-Koltunski and Hardmeier, 2017). Most of these methods use an additional encoder (Jean et al., 2017; Wang et al., 2017) to extract contextual information from previous source-side sentences. However, this requires additional parameters and it does not exploit the representations already learned by the NMT encoder. More recently, Tu et al. (2018) have shown that a cache-based memory network performs better than the above encoder-based methods. The cache-based memory keeps past context as a set of words, where each cell corresponds to one unique word keeping the hidden representations learned by the NMT while translating it. However, in this method, the word representations are stored irrespective of the sentences where they occur, and those vector representations are disconnected from the original NMT network.

We propose to use a hierarchical attention network (HAN) (Yang et al., 2016) to model the contextual information in a structured manner using word-level and sentence-level abstractions. In contrast to the hierarchical recurrent neural network (HRNN) used by (Wang et al., 2017), here the attention allows dynamic access to the context by selectively focusing on different sentences and words for each predicted word. In addition, we integrate two HANs in the NMT model to account for target and source context. The HAN encoder helps in the disambiguation of source-word representations, while the HAN decoder improves the target-side lexical cohesion and coherence. The integration is done by (i) re-using the hidden representations from both the encoder and decoder of previous sentence translations and (ii) providing input to both the encoder and decoder for the current translation. This integration method enables it to jointly optimize for multiple-sentences. Furthermore, we extend the original HAN with a multi-head attention (Vaswani et al., 2017) to capture different types of discourse phenomena.

Our main contributions are the following: (i) We propose a HAN framework for translation to capture context and inter-sentence connections in a structured and dynamic manner. (ii) We integrate the HAN in a very competitive NMT ar-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2947–2954
Brussels, Belgium, October 31 - November 4, 2018.

chitecture (Vaswani et al., 2017) and show significant improvement over two strong baselines on multiple data sets. (iii) We perform an ablation study of the contribution of each HAN configuration, showing that contextual information obtained from source and target sides are complementary.

2 The Proposed Approach

The goal of NMT is to maximize the likelihood of a set of sentences in a target language represented as sequences of words $\mathbf{y} = (y_1, ..., y_t)$ given a set of input sentences in a source language $\mathbf{x} = (x_1, ..., x_m)$ as:

$$\max_{\Theta} \frac{1}{N} \sum_{n=1}^{N} log(P_{\Theta}(\mathbf{y^n}|\mathbf{x^n})) \tag{1}$$

so, the translation of a document $\mathbf{D}$ is made by translating each of its sentences independently. In this study, we introduce dependencies on the previous sentences from the source and target sides:

$$\max_{\Theta} \frac{1}{N} \sum_{n=1}^{N} log(P_{\Theta}(\mathbf{y^n}|\mathbf{x^n}, \mathbf{D_{x^n}}, \mathbf{D_{y^n}})) \tag{2}$$

where $\mathbf{D_{x^n}} = (\mathbf{x^{n-k}}, ..., \mathbf{x^{n-1}})$ and $\mathbf{D_{y^n}} = (\mathbf{y^{n-k}}, ..., \mathbf{y^{n-1}})$ denote the previous k sentences from source and target sides respectively. The contexts $\mathbf{D_{x^n}}$ and $\mathbf{D_{y^n}}$ are modeled with HANs.

2.1 Hierarchical Attention Network

The proposed HAN has two levels of abstraction. The word-level abstraction summarizes information from each previous sentence j into a vector s^j as:

$$q_w = f_w(h_t) \tag{3}$$

$$s^j = \underset{i}{\text{MultiHead}}(q_w, h_i^j) \tag{4}$$

where h denotes a hidden state of the NMT network. In particular, h_t is the last hidden state of the word to be encoded, or decoded at time step t, and h_i^j is the last hidden state of the i-th word of the j-th sentence of the context. The function f_w is a linear transformation to obtain the *query* q_w. We used the MultiHead attention function proposed by (Vaswani et al., 2017) to capture different types of relations among words. It matches the *query* against each of the hidden representations h_i^j (used as *value* and *key* for the attention).

The sentence-level abstraction summarizes the contextual information required at time t in d_t as:

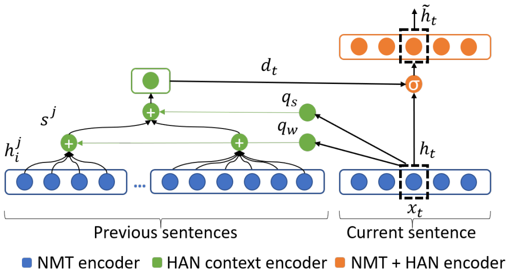

Figure 1: Integration of HAN during encoding at time step t, $\tilde{h}_t$ is the context-aware hidden state of the word x_t. Similar architecture is used during decoding.

$$q_s = f_s(h_t) \tag{5}$$

$$d_t = \text{FFN}(\underset{j}{\text{MultiHead}}(q_s, s^j)) \tag{6}$$

where f_s is a linear transformation, q_s is the query for the attention function, FFN is a position-wise feed-forward layer (Vaswani et al., 2017). Each layer is followed by a normalization layer (Lei Ba et al., 2016).

2.2 Context Gating

We use a gate (Tu et al., 2018, 2017) to regulate the information at sentence-level h_t and the contextual information at document-level d_t. The intuition is that different words require different amount of context for translation:

$$\lambda_t = \sigma(W_h h_t + W_d d_t) \tag{7}$$

$$\tilde{h}_t = \lambda_t h_t + (1 - \lambda_t) d_t \tag{8}$$

where W_h, W_p are parameter matrices, and $\tilde{h}_t$ is the final hidden representation for a word x_t or y_t.

2.3 Integrated Model

The context can be used during encoding or decoding a word, and it can be taken from previously encoded source sentences, previously decoded target sentences, or from previous alignment vectors (i.e. context vectors (Bahdanau et al., 2015)). The different configurations will define the input *query* and *values* of the attention function. In this work we experiment with five of them: one at encoding time, three at decoding time, and one combining both. At encoding time the *query* is a function of the hidden state h_{x_t} of the current word to be encoded x_t, and the *values* are the encoded states of previous sentences $h_{x_i}^j$ (HAN encoder). At decoding time, the *query* is a function of the hidden state h_{y_t} of the current word to be decoded y_t, and the *values* can be (a) the encoded states of previous sentences $h_{x_i}^j$ (HAN decoder *source*),

(b) the decoded states of previous sentences $h^j_{y_i}$ (HAN decoder), and (c) the alignment vectors c^j_i (HAN decoder *alignment*). Finally, we combine complementary target-source sides of the context by joining HAN encoder and HAN decoder. Figure 1 shows the integration of the HAN encoder with the NMT model; a similar architecture is applied to the decoder. The output $\tilde{h}_t$ is used by the NMT model as replacement of h_t during the final classification layer.

3 Experimental Setup

3.1 Datasets and Evaluation Metrics

We carry out experiments with Chinese-to-English (Zh-En) and Spanish-to-English (Es-En) sets on three different domains: talks, subtitles, and news.

TED Talks is part of the IWSLT 2014 and 2015 (Cettolo et al., 2012, 2015) evaluation campaigns[1]. We use *dev2010* for development; and *tst2010-2012* (Es-En), *tst2010-2013* (Zh-En) for testing. The Zh-En subtitles corpus is a compilation of TV subtitles designed for research on context (Wang et al., 2018). In contrast to the other sets, it has three references to compare. The Es-En corpus is a subset of OpenSubtitles2018 (Lison and Tiedemann, 2016)[2]. We randomly select two episodes for development and testing each. Finally, we use the Es-En News-Commentaries11[3] corpus which has document-level delimitation. We evaluate on WMT sets (Bojar et al., 2013): *newstest2008* for development, and *newstest2009-2013* for testing. A similar corpus for Zh-En is too small to be comparable. Table 2 shows the corpus statistics.

For evaluation, we use BLEU score (Papineni et al., 2002) (*multi-blue*) on *tokenized* text, and we measure significance with the paired bootstrap resampling method proposed by Koehn (2004) (implementations by Koehn et al. (2007)).

3.2 Model Configuration and Training

As baselines, we use a NMT transformer, and a context-aware NMT transformer with cache memory which we implemented for comparison following the best model described by Tu et al. (2018), with memory size of 25 words. We used the OpenNMT (Klein et al., 2017) implementation of the transformer network. The configuration is the same as the model called "base model" in the original paper (Vaswani et al., 2017). The encoder and decoder are composed of 6 hidden layers each. All hidden states have dimension of 512, dropout of 0.1, and 8 heads for the multi-head attention. The target and source vocabulary size is 30K. The optimization and regularization methods were the same as proposed by Vaswani et al. (2017). Inspired by Tu et al. (2018) we trained the models in two stages. First we optimize the parameters for the NMT without the HAN, then we proceed to optimize the parameters of the whole network. We use $k = 3$ previous sentences, which gave the best performance on the development set.

4 Experimental Results

4.1 Translation Performance

Table 1 shows the BLEU scores for different models. The baseline NMT transformer already has better performance than previously published results on these datasets, and we replicate previous previous improvements from the cache method over the this stronger baseline. All of our proposed HAN models perform at least as well as the cache method. The best scores are obtained by the combined encoder and decoder HAN model, which is significantly better than the cache method on all datasets without compromising training speed (2.3K vs 2.6K tok/sec). An important portion of the improvement comes from the HAN encoder, which can be attributed to the fact that the source-side always contains correct information, while the target-side may contain erroneous predictions at testing time. But combining HAN decoder with HAN encoder further improves translation performance, showing that they contribute complementary information. The three ways of incorporating information into the decoder all perform similarly.

Table 3 shows the performance of our best HAN model with a varying number k of previous sentences in the test-set. We can see that the best performance for TED talks and news is archived with 3, while for subtitles it is similar between 3 and 7.

4.2 Accuracy of Pronoun/Noun Translations

We evaluate coreference and anaphora using the reference-based metric: accuracy of pronoun translation (Miculicich Werlen and Popescu-Belis, 2017b), which can be extended for nouns. The list of evaluated pronouns is predefined in the metric, while the list of nouns was extracted using NLTK POS tagging (Bird, 2006). The upper part

---

[1] `https://wit3.fbk.eu`

[2] `http://www.opensubtitles.org`

[3] `http://opus.nlpl.eu/News-Commentary11.php`

	TED Talks				**Subtitles**				**News**	
	Zh–En		**Es–En**		**Zh–En** [4]		**Es–En**		**Es–En**	
Models	BLEU	Δ	BLEU	Δ	BLEU	Δ	BLEU	Δ	BLEU	Δ
NMT transformer	16.87		35.44		28.60		35.20		21.36	
+ cache (Tu et al., 2018)	17.32	$(+0.45)^{***}$	36.46	$(+1.02)^{***}$	28.86	$(+0.26)$	35.49	$(+0.29)$	22.36	$(+1.00)^{***}$
+ HAN encoder	17.61	$(+0.74)^{***}_{\dagger\dagger}$	36.91	$(+1.47)^{***}_{\dagger\dagger}$	29.35	$(+0.75)^{*}_{\dagger}$	35.96	$(+0.76)^{*}_{\dagger}$	22.36	$(+1.00)^{***}$
+ HAN decoder	17.39	$(+0.52)^{***}$	37.01	$(+1.57)^{***}_{\dagger\dagger\dagger}$	29.21	$(+0.61)^{*}$	35.50	$(+0.30)$	22.62	$(+1.26)^{***}_{\dagger\dagger\dagger}$
+ HAN decoder *source*	17.56	$(+0.69)^{***}_{\dagger\dagger}$	36.94	$(+1.50)^{***}_{\dagger\dagger}$	28.92	$(+0.32)$	35.71	$(+0.51)^{*}$	22.68	$(+1.32)^{***}_{\dagger\dagger\dagger}$
+ HAN decoder *alignment*	17.48	$(+0.61)^{***}_{\dagger}$	37.03	$(+1.60)^{***}_{\dagger\dagger\dagger}$	28.87	$(+0.27)$	35.63	$(+0.43)$	22.59	$(+1.23)^{***}_{\dagger\dagger\dagger}$
+ HAN encoder + HAN decoder	**17.79**	$(+0.92)^{***}_{\dagger\dagger\dagger}$	**37.24**	$(+1.80)^{***}_{\dagger\dagger\dagger}$	**29.67**	$(+1.07)^{**}_{\dagger}$	**36.23**	$(+1.03)^{**}_{\dagger\dagger}$	**22.76**	$(+1.40)^{***}_{\dagger\dagger\dagger}$

Table 1: BLEU score for the different configurations of the HAN model, and two baselines. The highest score per dataset is marked in bold. Δ denotes the difference in BLEU score with respect of the NMT transformer. The significance values with respect to the NMT and the cache method are denoted by $*$, and † respectively. The repetitions correspond to the p-values: $^{*}_{\dagger} < .05, ^{**}_{\dagger\dagger} < .01, ^{***}_{\dagger\dagger\dagger} < .001$.

	TED Talks		**Subtitles**		**News**
	Zh–En	Es–En	Zh–En	Es–En	Es–En
Training	0.2M	0.2M	2.2M	4.0M	0.2M
Development	0.8K	0.8K	1.1K	1.0K	1.9K
Test	5.5K	4.7K	1.2K	1.0K	13.5K

Table 2: Dataset statistics in # sentence pairs.

of Table 4 shows the results. For nouns, the joint HAN achieves the best accuracy with a significant improvement compared to other models, showing that target and source contextual information are complementary. Similarity for pronouns, the joint model has the best result for TED talks and news. However, HAN encoder alone is better in the case of subtitles. Here HAN decoder produces mistakes by repeating past translated personal pronouns. Subtitles is a challenging corpus for personal pronoun disambiguation because it usually involves dialogue between multiple speakers.

4.3 Cohesion and Coherence Evaluation

We use the metric proposed by Wong and Kit (2012) to evaluate lexical cohesion. It is defined as the ratio between the number of repeated and lexically similar content words over the total number of content words in a target document. The lexical similarity is obtained using WordNet. Table 4 (bottom-left) displays the average ratio per tested document. In some cases, HAN decoder achieves the best score because it produces a larger quantity of repetitions than other models. However, as previously demonstrated in 4.2, repetitions do not always make the translation better. Although HAN boosts lexical cohesion, the scores are still far from the human reference, so there is room for improvement in this aspect.

For coherence, we use a metric based on Latent Semantic Analysis (LSA) (Foltz et al., 1998). LSA is used to obtain sentence representations, then cosine similarity is calculated from one sentence to the next, and the results are averaged to get a document score. We employed the pre-trained LSA model *Wiki-6* from (Stefanescu et al., 2014). Table 4 (bottom-right) shows the average coherence score of documents. The joint HAN model consistently obtains the best coherence score, but close to other HAN models. Most of the improvement comes from the HAN decoder.

	TED Talks		**Subtitles**		**News**
k	Zh–En	Es–En	Zh–En	Es–En	Es–En
1	17.70	37.20	29.35	36.20	22.46
3	**17.79**	**37.24**	29.67	**36.23**	**22.76**
5	17.49	37.11	**29.69**	36.22	22.54
7	17.00	37.22	29.64	36.21	22.64

Table 3: Performance for variable context sizes k with the HAN encoder + HAN decoder.

4.4 Qualitative Analysis

Table 5 shows an example where HAN helped to generate the correct translation. The first box shows the current sentence with the analyzed word in bold; and the second, the past context at source and target. For the context visualization we use the toolkit provided by Pappas and Popescu-Belis (2017). Red corresponds to sentences, and blue to words. The intensity of color is proportional to the weight. We see that HAN correctly translates the ambiguous Spanish pronoun "*su*" into the English "*his*". The HAN decoder highlighted a previous mention of "*his*", and the HAN encoder highlighted the antecedent "*Nathaniel*". This shows that HAN can capture interpretable inter-sentence connections. More samples with different attention heads are shown in the Appendix **??**.

5 Related Work

Statistical Machine Translation (SMT) Initial studies were based on cache memories (Tiede-

---

[4] *NIST BLEU*: NMT transformer 35.99, cache 36.52, and HAN 37.15.

	Noun Translation					Pronoun Translation				
	TED Talks		Subtitles		News	TED Talks		Subtitles		News
Model	Zh–En	Es–En	Zh–En	Es–En	Zh–En	Es–En	Zh–En	Es–En	Zh–En	Es–En
NMT Transformer	40.16	65.97	46.65	61.79	47.94	63.44	68.00	69.71	65.83	47.22
+ cache	40.87	66.75	46.00	61.87	49.91	63.53	68.66	69.97	66.27	49.34
+ HAN encoder	41.93	67.75	46.78	61.52	50.06	64.05	69.17	**71.04**	**68.56**	49.57
+ HAN decoder	41.61	67.35	46.78	61.99	50.03	64.02	69.36	70.50	67.03	49.33
+ HAN encoder + HAN decoder	**42.99**	**67.81**	**47.43**	**62.30**	**50.40**	**64.35**	**69.60**	70.60	67.47	**49.59**
	Lexical cohesion					**Coherence**				
NMT Transformer	54.26	51.98	51.87	51.77	30.06	0.298	0.299	0.283	0.262	0.279
+ HAN encoder	54.87	52.35	51.89	52.33	30.34	0.304	0.299	0.285	0.262	0.280
+ HAN decoder	54.95	**52.43**	**52.33**	52.43	30.41	0.302	0.301	0.287	0.265	0.282
+ HAN enc. + HAN dec.	**55.40**	52.36	51.94	**52.75**	**30.58**	**0.305**	**0.302**	**0.287**	**0.265**	**0.282**
Human reference	56.08	57.02	54.81	58.19	35.12	0.310	0.314	0.296	0.270	0.298

Table 4: Evaluation on discourse phenomena. Noun and pronoun translation: Accuracy with respect to a human reference. Lexical cohesion: Ratio of repeated and lexically similar words over the number of content words. Coherence: Average cosine similarity of consecutive sentences (i.e. average of LSA word-vectors)

Currently Translated Sentence

Src.:	y esto es un escape de **su** estado atormentado .
Ref.:	and that is an escape from **his** tormented state .
Base:	and this is an escape from ***its*** $< unk >$ state .
Cache:	and this is an escape from ***their*** state .
HAN:	and this is an escape from **his** $< unk >$ state .

Context from Previous Sentences

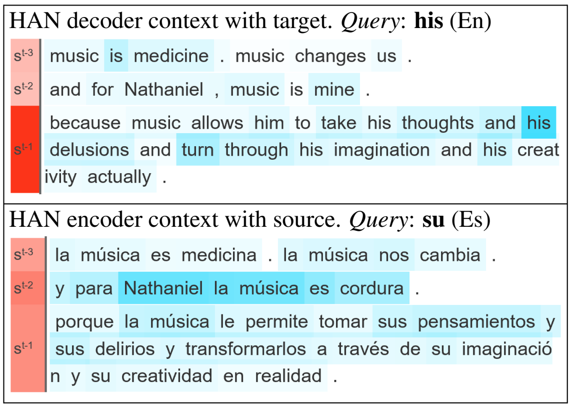

HAN decoder context with target. *Query*: **his** (En)

s^{t-3} music is medicine . music changes us .

s^{t-2} and for Nathaniel , music is mine .

s^{t-1} because music allows him to take his thoughts and his delusions and turn through his imagination and his creativity actually .

HAN encoder context with source. *Query*: **su** (Es)

s^{t-3} la música es medicina . la música nos cambia .

s^{t-2} y para Nathaniel la música es cordura .

s^{t-1} porque la música le permite tomar sus pensamientos y sus delirios y transformarlos a través de su imaginación y su creatividad en realidad .

Table 5: Example of pronoun disambiguation using HAN (TED Talks Es-En).

mann, 2010; Gong et al., 2011). However, most of the work explicitly models discourse phenomena (Sim Smith, 2017) such as lexical cohesion (Meyer and Popescu-Belis, 2012; Xiong et al., 2013; Loáiciga and Grisot, 2016; Pu et al., 2017; Mascarell, 2017), coherence (Born et al., 2017), and coreference (Rios Gonzales and Tuggener, 2017; Miculicich Werlen and Popescu-Belis, 2017a). Hardmeier et al. (2013) introduced the document-level SMT paradigm.

Sentence-level NMT Initial studies on NMT enhanced the sentence-level context by using memory networks (Wang et al., 2016), self-attention (Miculicich Werlen et al., 2018; Zhang et al., 2016), and latent variables (Yang et al., 2017).

Document-level NMT Tiedemann and Scherrer (2017) use the concatenation of multiple sentences as NMT's input/output, Jean et al. (2017) add a context encoder for the previous source sentence, Wang et al. (2017) includes a HRNN to summarize source-side context, and Tu et al. (2018) use a dynamic cache memory to store representations of previously translated words. Recently, Bawden et al. (2018) proposed test-sets for evaluating discourse in NMT, Voita et al. (2018) shows that context-aware NMT improves the of anaphoric pronouns, and Maruf and Haffari (2018) proposed a document-level NMT using memory-networks.

6 Conclusion

We proposed a hierarchical multi-head HAN NMT model[5] to capture inter-sentence connections. We integrated context from source and target sides by directly connecting representations from previous sentence translations into the current sentence translation. The model significantly outperforms two competitive baselines, and the ablation study shows that target and source context is complementary. It also improves lexical cohesion and coherence, and the translation of nouns and pronouns. The qualitative analysis shows that the model is able to identify important previous sentences and words for the correct prediction. In future work, we plan to explicitly model discourse connections with the help of annotated data, which may further improve translation quality.

Acknowledgments

We are grateful for the support of the European Union under the Horizon 2020 SUMMA project n. 688139, see `www.summa-project.eu`.

---

[5] Code available at `https://github.com/idiap/HAN_NMT`. Project Towards Document-Level NMT (Miculicich Werlen, 2017)

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, San Diego, USA.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating discourse phenomena in neural machine translation. In *Proceedings of the 16th Annual Conference of North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, USA. Association for Computational Linguistics.

Steven Bird. 2006. Nltk: The natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Leo Born, Mohsen Mesgar, and Michael Strube. 2017. Using a graph-based coherence model in document-level machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 26–35, Copenhagen, Denmark. Association for Computational Linguistics.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The IWSLT 2015 evaluation campaign. In *In proceedins of the International Workshop on Spoken Language Translation*.

Peter W Foltz, Walter Kintsch, and Thomas K Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse processes*, 25(2-3):285–307.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 909–919, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Christian Hardmeier. 2012. Discourse in statistical machine translation. a survey and a case study. *Discours. Revue de linguistique, psycholinguistique et informatique. A journal of linguistics, psycholinguistics and computational linguistics*, (11).

Christian Hardmeier, Sara Stymne, Jörg Tiedemann, and Joakim Nivre. 2013. cder for phrase-based statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 193–198, Sofia, Bulgaria. Association for Computational Linguistics.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? *arXiv preprint arXiv:1704.05135*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Ekaterina Lapshinova-Koltunski and Christian Hardmeier. 2017. Discovery of discourse-related language contrasts through alignment discrepancies in english-german translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 73–81, Copenhagen, Denmark. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. In *NISP 2016 - Deep Learning Symposium paper*.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *In Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association.

Sharid Loáiciga and Cristina Grisot. 2016. Predicting and using a pragmatic component of lexical aspect. *LiLT (Linguistic Issues in Language Technology)*, 13.

Sameen Maruf and Gholamreza Haffari. 2018. Document context neural machine translation with memory networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1275–1284. Association for Computational Linguistics.

Laura Mascarell. 2017. Lexical chains meet word embeddings in document-level statistical machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 99–109, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Meyer and Andrei Popescu-Belis. 2012. Using sense-labeled discourse connectives for statistical machine translation. In *Proceedings of the Joint Workshop on Exploiting Synergies between Information Retrieval and Machine Translation (ESIRMT) and Hybrid Approaches to Machine Translation (HyTra)*, pages 129–138. Association for Computational Linguistics.

Thomas Meyer and Bonnie Webber. 2013. Implicitation of discourse connectives in (machine) translation. In *Proceedings of the Workshop on Discourse in Machine Translation*, pages 19–26, Sofia, Bulgaria. Association for Computational Linguistics.

Lesly Miculicich Werlen. 2017. Towards document-level neural machine translation. Technical report, Idiap.

Lesly Miculicich Werlen, Nikolaos Pappas, Dhananjay Ram, and Andrei Popescu-Belis. 2018. Self-attentive residual decoder for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1366–1379. Association for Computational Linguistics.

Lesly Miculicich Werlen and Andrei Popescu-Belis. 2017a. Using coreference links to improve spanish-to-english machine translation. In *Proceedings of the 2nd Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2017)*, pages 30–40, Valencia, Spain. Association for Computational Linguistics.

Lesly Miculicich Werlen and Andrei Popescu-Belis. 2017b. Validation of an automatic metric for the accuracy of pronoun translation (APT). In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 17–25, Copenhagen, Denmark. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Nikolaos Pappas and Andrei Popescu-Belis. 2017. Multilingual hierarchical attention networks for document classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1015–1025, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Xiao Pu, Laura Mascarell, and Andrei Popescu-Belis. 2017. Consistent translation of repeated nouns using syntactic and semantic cues. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 948–957. Association for Computational Linguistics.

Annette Rios Gonzales and Don Tuggener. 2017. Coreference resolution of elided subjects and possessive pronouns in spanish-english statistical machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 657–662. Association for Computational Linguistics.

Karin Sim Smith. 2017. On integrating discourse in machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 110–121, Copenhagen, Denmark. Association for Computational Linguistics.

Dan Stefanescu, Rajendra Banjade, and Vasile Rus. 2014. Latent semantic analysis models on wikipedia and tasa. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Jörg Tiedemann. 2010. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 8–15, Uppsala, Sweden. Association for Computational Linguistics.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92, Copenhagen, Denmark. Association for Computational Linguistics.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics*, 5:87–99.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *Transactions of the Association for Computational Linguistics*, 6.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 1264–1274. Association for Computational Linguistics.

Longyue Wang, Zhaopeng Tu, Shuming Shi, Tong Zhang, Yvette Graham, and Qun Liu. 2018. Translating pro-drop languages with reconstruction models. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1–9, New Orleans, Louisiana, USA. AAAI Press.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2826–2831, Copenhagen, Denmark. Association for Computational Linguistics.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 278–286, Austin, Texas. Association for Computational Linguistics.

Billy T. M. Wong and Chunyu Kit. 2012. Extending machine translation evaluation metrics with lexical cohesion to document level. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1060–1068, Jeju Island, Korea. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Deyi Xiong, Yang Ding, Min Zhang, and Chew Lim Tan. 2013. Lexical chain based cohesion models for document-level statistical machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1563–1573, Seattle, Washington, USA. Association for Computational Linguistics.

Zichao Yang, Zhiting Hu, Yuntian Deng, Chris Dyer, and Alex Smola. 2017. Neural machine translation with recurrent attention modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 383–387, Valencia, Spain. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, jinsong su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530, Austin, Texas. Association for Computational Linguistics.

Three Strategies to Improve One-to-Many Multilingual Translation

Yining Wang[2,3] , **Jiajun Zhang**[1,2,3]*, **Feifei Zhai**[5], **Jingfang Xu**[5] and **Chengqing Zong**[2,3,4]
[1]Beijing Advanced Innovation Center for Language Resources, Beijing, China
[2]National Laboratory of Pattern Recognition, CASIA, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
[4]CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
[5]Sogou Inc., Beijing, China
{yining.wang, jjzhang, cqzong}@nlpr.ia.ac.cn
{zhaifeifei,xujingfang}@sogou-inc.com

Abstract

Due to the benefits of model compactness, multilingual translation (including many-to-one, many-to-many and one-to-many) based on a universal encoder-decoder architecture attracts more and more attention. However, previous studies show that one-to-many translation based on this framework cannot perform on par with the individually trained models. In this work, we introduce three strategies to improve one-to-many multilingual translation by balancing the shared and unique features. Within the architecture of one decoder for all target languages, we first exploit the use of unique initial states for different target languages. Then, we employ language-dependent positional embeddings. Finally and especially, we propose to divide the hidden cells of the decoder into shared and language-dependent ones. The extensive experiments demonstrate that our proposed methods can obtain remarkable improvements over the strong baselines. Moreover, our strategies can achieve comparable or even better performance than the individually trained translation models.

1 Introduction

Encoder-decoder based neural machine translation (NMT) has achieved the new state-of-the-art due to powerful end-to-end modeling (Sutskever et al., 2014; Bahdanau et al., 2015; Wu et al., 2016; Hassan et al., 2018). Under this end-to-end framework, many researchers attempt to improve the translation quality between two languages by exploiting monolingual data (Sennrich et al., 2016; Zhang and Zong, 2016), taking advantage of both NMT and statistical machine translation (Wang et al., 2017a; Tang et al., 2016; Zhao et al., 2018; Zhou et al., 2017) and so on.

Another research direction about how to perform multilingual translation within this encoder-decoder architecture has recently drawn more and more attention (Zoph and Knight, 2016; Dong et al., 2015; Luong et al., 2016; Johnson et al., 2017; Firat et al., 2016b).

In multilingual translation scenarios, one can employ multi-task learning framework to perform many-to-one or one-to-many translation using multiple encoders or multiple decoders (Luong et al., 2016; Dong et al., 2015). Firat et al. (2016a) and Lu et al. (2018) further propose to share a universal attention mechanism for many-to-many translations. In these methods, encoder or decoder is language dependent and network parameters increase linearly with the number of languages.

Johnson et al. (2017) and Ha et al. (2016) present an appealing approach in which a universal encoder-decoder framework is designed for many-to-one, many-to-many and one-to-many multilingual translation tasks. The network model is compact and the model size does not grow as the number of languages increases. However, Johnson et al. (2017) observe that only the many-to-one paradigm can achieve better translation results than the individually trained models. For the other two paradigms, there are various degrees of quality degradation. In this work, we focus on one-to-many multilingual translation under the universal encoder-decoder framework and attempt to boost its performance while maintaining the model compactness.

To this end, we propose three strategies which exploit the unique features of each target language and keep as many parameters shared as possible. First, we design two special labels at the tail of encoder and the head of decoder to mark the target language and guide the generation of different target languages. Then, we introduce language-dependent positional embeddings into the bottom

---

[*] Jiajun Zhang is the corresponding author and the work is done while Yining Wang is doing research intern at Sogou Inc.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2955–2960
Brussels, Belgium, October 31 - November 4, 2018.

layer of the decoder network and correspondingly the structural difference between target languages can be well captured. Finally and especially, we propose a new parameter-sharing mechanism in which we divide the hidden units of each decoder layer into shared and language-dependent ones.

We verify the effectiveness of our proposed methods on two one-to-many tasks: Chinese-to-English/Japanese translation and English-to-German/French translation. The experimental results demonstrate that the three strategies can significantly outperform the baseline multilingual models and they can achieve comparable or even better performance than the individually trained translation models.

Specifically, our contributions in this paper are two-fold:

- The proposed three strategies can take advantage of unique features of each target language while sharing the network parameters as many as possible.
- The extensive experiments on multiple translation tasks show that the three proposed strategies improve the translation quality. Moreover, the effects of the strategies are complementary and the combined one can perform on par with or better than the individually optimized translation models.

2 Background

Our proposed approach can be applied to any encoder-decoder architecture. Considering the excellent translation performance of Transformer network (Vaswani et al., 2017), we implement our method entirely based on it in this work. Transformer consists of stacked encoder and decoder layers. The encoder maps an input sequence $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, z_2, \cdots, z_n)$ whose size varies with respect to the source sentence length. The decoder generates an output sequence $\mathbf{y} = (y_1, y_2, \cdots, y_m)$ from the continuous representations $\mathbf{z}$. Since the Transformer network contains no recurrence, positional embeddings are used in model to make use of sequence order. The encoder and decoder are trained to maximize the conditional probability of target sequence given a source sequence:

$$L(\theta) = \sum_{t=1}^{N} \log P(y_t | y_{<t}, x; \theta) \qquad (1)$$

For the sake of brevity, we refer the reader to Vaswani et al. (2017) for more details regarding the architecture.

3 Method Description

In this section, we introduce our general strategies for extending the transformer network to one-to-many translation task. We decompose the probability of the target sequences into the products of per token probabilities in all translation forms:

$$L(\theta) = \sum_{t=1}^{M} \sum_{l=1}^{N_l} \log(P(y_t^l | x, y_{<t}^l; \theta)) \qquad (2)$$

where M is number of target languages, and $P(y_t^l | x, y_{<t}^l; \theta)$ denotes the translation probability of t-th word of the l-th target language. Note that the translation process for all target languages uses the same parameter set θ.

Our methods mainly concentrate on improving one-to-many multilingual translation by designing new decoder structure under the universal encoder-decoder framework. The idea is to exploit the shared and unique features of different target languages, and we respectively propose three strategies including special label initialization, language-dependent positional embedding and a new parameter-sharing mechanism.

3.1 Special Label Initialization

In the universal encoder-decoder network for one-to-many multilingual translation (Johnson et al., 2017), a special token (e.g. **en2fr**) is added at the end of the source sentence to indicate the translation direction. Although it is an effective mechanism, we find that the initial states of the decoder are very important to guide the generation process for different target languages. In order to enhance the model, we utilize another special language-dependent label at the beginning of the decoder and we regard it as the first generated token of the target language (e.g. **2fr**).

3.2 Language-dependent Positional Embedding

Positional embeddings give the model the sense of which part of the sequence is currently being dealt with. Intuitively, different target languages should have different positional embeddings to distinguish the structural difference between multiple target languages. Therefore, we design language-dependent positional embeddings in the universal

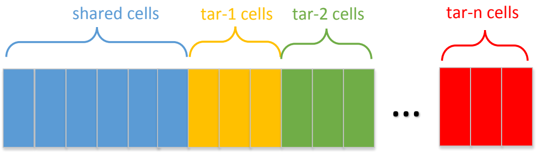

Figure 1: The hidden units of decoder network. Blue part represents the shared units, and yellow, green and red parts denote different language-dependent units respectively.

encoder-decoder multilingual translation. For the fixed embedding method (Vaswani et al., 2017), $sine(x)$ and $cosine(x)$ functions are used to generate positional embeddings. In this case, we introduce trigonometric functions with different orders or offsets on the decoder to distinguish different target languages. For the dynamic embedding method (Gehring et al., 2017), we equip the target inputs by embedding the absolute position of different languages separately.

3.3 Shared and Language-dependent Hidden Units per Layer

In the universal encoder-decoder multilingual translation, the hidden layers of the decoder are responsible for generating different target language sentences. As a result, the hidden layers should embody some language-dependent information.

In this work, we propose to divide the hidden units of each decoder layer into shared units and language-dependent ones. On the one hand, shared units can learn the commonality of languages and enable one-to-many translation to share the network parameters as many as possible. On the other hand, language-dependent units are capable of capturing the characteristic of each specific language.

Figure 1 gives a brief description of our proposed strategy. For instance, in training step for one target language (**tar-1**), we tune the shared units and the language-dependent units of **tar-1**, and mask out other parts. In decoding step, we only use the shared and language-dependent hidden units of target language **tar-1** to predict translation results.

4 Experiments Settings

In this section, we test the proposed methods on two one-to-many translation tasks, including (i) Chinese→English/Japanese in general domain, and (ii) English→French/German in WMT14 task.

Chinese→English/Japanese For this translation task, the training sets of Chinese-to-English (briefly, Zh→En) and Chinese-to-Japanese (briefly, Zh→Ja) both contain about 10 million parallel corpora. We evaluate our methods on NIST03-06 (MT03-06) for Zh→En translation and 400 sentences extracted from our general corpus for Zh→Ja translation.

English→French/German The training set consists of about 4.5 million bilingual sentence pairs in WMT14 English-German (briefly, En→De) task and about 36 million sentence pairs in WMT14 English-French (briefly, En→Fr) task[1]. We use the combination of *newstest2012* and *newstest2013* as our validation set, and we use *newstest2014* as our test set on En→De and En→Fr tasks.

We adopt the tensor2tensor[2] library for training and evaluating our basic Transformer translation model. We use wordpiece method (Wu et al., 2016; Schuster and Nakajima, 2012) to encode source side sentences and the combination of target side sentences. The vocabulary size is 37,000 for both sides. We train our models using configuration *transformer_big* adopted by Vaswani et al. (2017), which contains a 6-layer encoder and a 6-layer decoder with 1024-dimensional hidden representations. During training, each mini-batch on one GPU contains a set of sentence pairs with roughly 3,072 source and 3,072 target tokens. We use Adam optimizer (Kingma and Ba, 2014) with β_1=0.9, β_2=0.98, and ϵ=10^{-9}. For our model, we train for 400,000 steps on one machine with 8 NVIDIA Tesla M40 GPUs.

5 Results and Analysis

We show the results of one-to-many translation experiments using our proposed strategies. The translation performance is evaluated by case-insensitive BLEU4 for Zh→En translation, character-level BLEU5 for Zh→Ja translation, and case-sensitive BLEU4 (Papineni et al., 2002) for En→De/Fr translation task.

5.1 Our Strategies vs. Baseline

Table 1 reports the main translation results of Zh→En/Ja and En→De/Fr translation tasks. We conduct universal one-to-many translation using

---

[1] http://www.statmt.org/wmt14/translation-task.html

[2] https://github.com/tensorflow/tensor2tensor

Methods	Zh→En					Zh→Ja	En→De	En→Fr
	MT03	MT04	MT05	MT06	**Ave**	**test**	**test**	**test**
Indiv	43.59	43.95	45.34	44.05	44.23	40.71	**27.84**	41.50
O2M	43.20	43.55	44.68	43.93	43.84	42.09	26.42	41.32
O2M + ①	43.91	44.01	45.12	44.14	44.30	42.54	26.78	41.56
O2M + ① + ② (Dyn)	44.24	44.45	45.43	44.51	44.66	42.77	26.98	41.78
O2M + ① + ② (Fixed)	44.13	44.57	45.22	44.68	44.65	42.70	26.90	41.75
O2M + ① + ③	44.78	45.23	45.78	45.22	45.25	42.97	27.11	**41.98**
O2M + ① + ② (Dyn)+ ③	**44.85**	**45.51**	**45.91**	**45.38**	**45.41**	**43.03**	27.23	41.92

Table 1: Translation performance of our methods on Zh→En/Ja and En→De/Fr tasks. Indiv means translation model of individual pair. O2M is the our baseline system. ① , ② and ③ denote our proposed three strategies of special label initialization, language-dependent positional embedding and the new parameter-sharing mechanism separately. ② (Dyn) and ② (Fixed) represent the two ways of language-dependent positional embedding method. For shared and language-dependent method, we set one-half of hidden units as shared units, and for another half, we use a quarter hidden units to denote two output languages respectively.

Johnson et al. (2017) method on Transformer framework as our baseline system (briefly, O2M method). From the first two lines, we can see that the O2M method cannot perform on par with the individually trained systems in most cases.

We mentioned before that our goal is to improve the universal one-to-many multilingual translation framework while maintaining the parameter sharing property. We can observe from the table that all our proposed strategies (last part in Table 1) improve the translation performance compared to the baseline (O2M). Specifically, the combined use of three strategies performs best and it can achieve the improvements up to 1.96 BLEU points (45.51 vs. 43.55 on Zh→En MT04). As for language-dependent positional embedding, we find that both fixed and dynamic styles perform similarly.

Our ultimate goal is to make the universal one-to-many framework as good as or better than the individually trained systems. Table 1 demonstrates some encouraging results. It is shown in the table that the universal one-to-many architecture enhanced with our strategies can outperform the individually trained models on three out of four language translations (Zh→En, Zh→Ja, En→Fr). The results verify the effectiveness of our proposed methods.

5.2 Comparison of Shared Unit Size

For the new parameter-sharing mechanism, it is an open question to decide how many hidden units should be shared and how many ones should be language dependent. To figure out this question, we further conduct an experiment to investigate different settings. For example, we keep a quarter of the hidden units of each decoder layer as shared and make the left three quarters evenly distributed to different target languages.

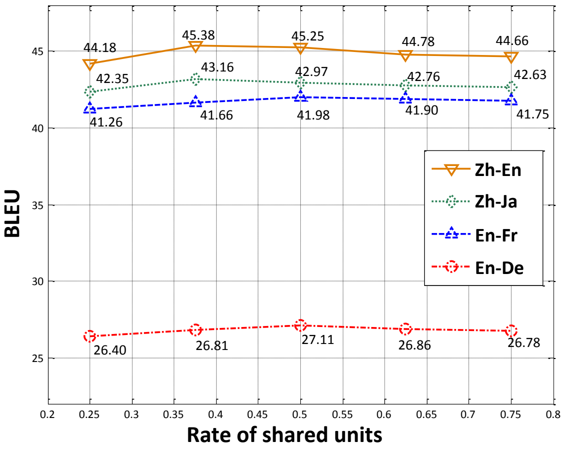

Figure 2: The comparison of different shared units.

Figure 2 reports the results. We can observe different trends for different language pairs. On the En→De/Fr translation task, the performance is best when we share one-half of the hidden units. In contrast, it obtains the best results when we share only 37.5% of hidden units on Zh→En/Ja translation. It indicates that similar languages (De/Fr) can share more hidden units and languages with a great difference (En/Ja) may share less hidden units.

6 Related Work

In this work, we explore the balancing problem of shared and unique parameters, and attempt to incorporate the language-dependent presentation features to distinguish different target languages under the scenario of one-to-many multilingual translation.

Multilingual translation has been extensively studied in Dong et al. (2015), Firat et al. (2016a), Luong et al. (2016) and Johnson et al. (2017). Owing to excellent translation performance and ease of use, many researchers (Blackwood et al., 2018; Lakew et al., 2018) have conduct translation of multiple languages based on the framework of Johnson et al. (2017) and Ha et al. (2016). As for low-resource translation scenario (Zoph et al., 2016; Chen et al., 2017; Wang et al., 2017b), similar to above method, Gu et al. (2018) enable sharing of lexical and sentence representation across multiple languages especially for low-resource multilingual NMT. Different from previous methods, our work mainly focuses on improving the one-to-many multilingual translation framework while sharing as many parameters as possible.

7 Conclusion

In this paper, we have proposed three effective strategies to improve the universal one-to-many multilingual translation, including special label initialization, language-dependent positional embedding and a new parameter-sharing mechanism. The empirical experiments on four language pairs demonstrate that our strategies can obtain significant improvement over the strong baseline, and can achieve comparable or even better results than the individually trained models.

For future work, we plan to extend our strategies on many-to-many multilingual translation scenarios, and explore other effective strategies to balance parameter sharing.

Acknowledgments

The research work described in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2016QY02D0303 and the Natural Science Foundation of China under Grant No. 61673380. The research work in this paper has also been supported by Beijing Advanced Innovation Center for Language Resources and Sogou Inc.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In Proceedings of ICLR 2015*.

Graeme Blackwood, Miguel Ballesteros, and Todd Ward. 2018. Multilingual neural machine translation with task-specific attention. *In Proceedings of COLING 2018*, pages 3112–3122.

Yun Chen, Yong Cheng, Yang Liu, and Li Victor, O.K. 2017. A teacher-student framework for zero-resource neural machine translation. *In Proceedings of ACL 2017*, pages 1925–1935.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. *In Proceedings of ACL 2015*, pages 1723–1732.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. *In Proceedings of NAACL-HLT 2016*, pages 866–875.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. *In Proceedings of EMNLP 2016*, pages 268–277.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1601.03317*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. 2018. Universal neural machine translation for extremely low resource languages. *In Proceedings of NAACL-HLT 2018*, pages 344–354.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *In Proceedings of IWSLT 2016*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic Chinese to English news translation. *arXiv preprint arXiv:1803.05567*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Surafel Melaku Lakew, Mauro Cettolo, and Marcello Federico. 2018. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. *In Proceedings of COLING 2018*, pages 641–652.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. *arXiv preprint arXiv:1804.08198*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. *In Proceedings of ICLR 2016*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *In Proceedings of ACL*, pages 311–318.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. *In Proceedings of ICASSP 2012*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. *In Proceedings of ACL 2016*, pages 86–96.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *In Proceedings of NIPS*, pages 3104–3112.

Yaohua Tang, Fandong Meng, Zhengdong Lu, Hang Li, and Philip LH Yu. 2016. Neural machine translation with external phrase memory. *arXiv preprint arXiv:1606.01792*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and Łukasz Kaiser. 2017. Attention is all you need. *In Proceedings of NIPS 2017*, pages 30–34.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017a. Neural machine translation advised by statistical machine translation. *In Proceedings of AAAI 2017*.

Yining Wang, Yang Zhao, Jiajun Zhang, Chengqing Zong, and Zhengshan Xue. 2017b. Towards neural machine translation with partially aligned corpora. *In Proceedings of IJCNLP 2017*, pages 384–393.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. *In Proceedings of EMNLP 2016*, pages 1535–1545.

Yang Zhao, Yining Wang, Jiajun Zhang, and Chengqing Zong. 2018. Phrase table as recommendation memory for neural machine translation. *In Proceedings of IJCAI 2018*, pages 4609–4615.

Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. 2017. Neural system combination for machine translation. *In Proceedings of ACL 2017*, pages 378–384.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *In Proceedings of NAACL-HLT 2016*, pages 30–34.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *In Proceedings of EMNLP 2016*, pages 1568–1575.

Multi-Source Syntactic Neural Machine Translation

Anna Currey
University of Edinburgh
a.currey@sms.ed.ac.uk

Kenneth Heafield
University of Edinburgh
kheafiel@inf.ed.ac.uk

Abstract

We introduce a novel multi-source technique for incorporating source syntax into neural machine translation using linearized parses. This is achieved by employing separate encoders for the sequential and parsed versions of the same source sentence; the resulting representations are then combined using a hierarchical attention mechanism. The proposed model improves over both seq2seq and parsed baselines by over 1 BLEU on the WMT17 English→German task. Further analysis shows that our multi-source syntactic model is able to translate successfully without any parsed input, unlike standard parsed methods. In addition, performance does not deteriorate as much on long sentences as for the baselines.

1 Introduction

Neural machine translation (NMT) typically makes use of a recurrent neural network (RNN) -based encoder and decoder, along with an attention mechanism (Bahdanau et al., 2015; Cho et al., 2014; Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014). However, it has been shown that RNNs require some supervision to learn syntax (Bentivogli et al., 2016; Linzen et al., 2016; Shi et al., 2016). Therefore, explicitly incorporating syntactic information into NMT has the potential to improve performance. This is particularly true for source syntax, which can improve the model's representation of the source language.

Recently, there have been a number of proposals for using linearized representations of parses within standard NMT (Aharoni and Goldberg, 2017; Li et al., 2017; Nadejde et al., 2017). Linearized parses are advantageous because they can inject syntactic information into the models without significant changes to the architecture. However, using linearized parses in a sequence-to-sequence (seq2seq) framework creates some challenges, particularly when using source parses. First, the parsed sequences are significantly longer than standard sentences, since they contain node labels as well as words. Second, these systems often fail when the source sentence is not parsed. This can be a problem for inference, since the external parser may fail on an input sentence at test time. We propose a method for incorporating linearized source parses into NMT that addresses these challenges by taking both the sequential source sentence and its linearized parse simultaneously as input in a multi-source framework. Thus, the model is able to use the syntactic information encoded in the parse while falling back to the sequential sentence when necessary. Our proposed model improves over both standard and parsed NMT baselines.

2 Related Work

2.1 Seq2seq Neural Parsing

Using linearized parse trees within sequential frameworks was first done in the context of neural parsing. Vinyals et al. (2015) parsed using an attentional seq2seq model; they used linearized, unlexicalized parse trees on the target side and sentences on the source side. In addition, as in this work, they used an external parser to create synthetic parsed training data, resulting in improved parsing performance. Choe and Charniak (2016) adopted a similar strategy, using linearized parses in an RNN language modeling framework.

2.2 NMT with Source Syntax

Among the first proposals for using source syntax in NMT was that of Luong et al. (2016), who introduced a multi-task system in which the source data was parsed and translated using a shared encoder and two decoders. More radical changes to

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2961–2966
Brussels, Belgium, October 31 - November 4, 2018.

the standard NMT paradigm have also been proposed. Eriguchi et al. (2016) introduced tree-to-sequence NMT; this model took parse trees as input using a tree-LSTM (Tai et al., 2015) encoder. Bastings et al. (2017) used a graph convolutional encoder in order to take labeled dependency parses of the source sentences into account. Hashimoto and Tsuruoka (2017) added a latent graph parser to the encoder, allowing it to learn soft dependency parses while simultaneously learning to translate.

2.3 Linearized Parse Trees in NMT

The idea of incorporating linearized parses into seq2seq has been adapted to NMT as a means of injecting syntax. Aharoni and Goldberg (2017) first did this by parsing the target side of the training data and training the system to generate parsed translations of the source input; this is the inverse of our parse2seq baseline. Similarly, Nadejde et al. (2017) interleaved CCG supertags with words on the target side, finding that this improved translation despite requiring longer sequences.

Most similar to our multi-source model is the parallel RNN model proposed by Li et al. (2017). Like multi-source, the parallel RNN used two encoders, one for words and the other for syntax. However, they combined these representations at the word level, whereas we combine them on the sentence level. Their mixed RNN model is also similar to our parse2seq baseline, although the mixed RNN decoder attended only to words. As the mixed RNN model outperformed the parallel RNN model, we do not attempt to compare our model to parallel RNN. These models are similar to ours in that they incorporate linearized parses into NMT; here, we utilize a multi-source framework.

2.4 Multi-Source NMT

Multi-source methods in neural machine translation were first introduced by Zoph and Knight (2016) for multilingual translation. They used one encoder per source language, and combined the resulting sentence representations before feeding them into the decoder. Firat et al. (2016) expanded on this by creating a multilingual NMT system with multiple encoders and decoders. Libovický and Helcl (2017) applied multi-source NMT to multimodal translation and automatic post-editing and explored different strategies for combining attention over the two sources. In this paper, we apply the multi-source framework to a novel task, syntactic neural machine translation.

3 NMT with Linearized Source Parses

We propose a multi-source method for incorporating source syntax into NMT. This method makes use of linearized source parses; we describe these parses in section 3.1. Throughout this paper, we refer to standard sentences that do not contain any explicit syntactic information as *sequential*; see Table 1 for an example.

3.1 Linearized Source Parses

We use an off-the-shelf parser, in this case Stanford CoreNLP (Manning et al., 2014), to create binary constituency parses. These parses are linearized as shown in Table 1. We tokenize the opening parentheses with the node label (so each node label begins with a parenthesis) but keep the closing parentheses separate from the words they follow. For our task, the parser failed on one training sentence of 5.9 million, which we discarded, and succeeded on all test sentences. It took roughly 16 hours to parse the 5.9 million training sentences.

Following Sennrich et al. (2016b), our networks operate at the subword level using byte pair encoding (BPE) with a shared vocabulary on the source and target sides. However, the parser operates at the word level. Therefore, we parse then break into subwords, so a leaf may have multiple tokens without internal structure.

The proposed method is tested using both *lexicalized* and *unlexicalized* parses. In *unlexicalized* parses, we remove the words, keeping only the node labels and the parentheses. In *lexicalized* parses, the words are included. Table 1 shows an example of the three source sentence formats: sequential, lexicalized parse, and unlexicalized parse. Note that the lexicalized parse is significantly longer than the other versions.

3.2 Multi-Source

We propose a multi-source framework for injecting linearized source parses into NMT. This model consists of two identical RNN encoders with no shared parameters, as well as a standard RNN decoder. For each target sentence, two versions of the source sentence are used: the sequential (standard) version and the linearized parse (lexicalized or unlexicalized). Each of these is encoded simul-

	Example Sentence
sequential	history is a great teacher .
lexicalized parse	(ROOT (S (NP (NN history)) (VP (VBZ is) (NP (DT a) (JJ great) (NN teacher))) (. .)))
unlexicalized parse	(ROOT (S (NP (NN)) (VP (VBZ) (NP (DT) (JJ) (NN))) (. .)))
target sentence	die Geschichte ist ein großartiger Lehrmeister .

Table 1: Example source training sentence with sequential, lexicalized parse, and unlexicalized parse versions. We include the corresponding target sentence for reference.

taneously using the encoders; the encodings are then combined and used as input to the decoder. We combine the source encodings using the hierarchical attention combination proposed by Libovický and Helcl (2017). This consists of a separate attention mechanism for each encoder; these are then combined using an additional attention mechanism over the two separate context vectors. This multi-source method is thus able to combine the advantages of both standard RNN-based encodings and syntactic encodings.

4 Experimental Setup

4.1 Data

We base our experiments on the WMT17 (Bojar et al., 2017) English (EN) → German (DE) news translation task. All 5.9 million parallel training sentences are used, but no monolingual data. Validation is done on newstest2015, while newstest2016 and newstest2017 are used for testing.

We train a shared BPE vocabulary with 60k merge operations on the parallel training data. For the parsed data, we break words into subwords after applying the Stanford parser. We tokenize and truecase the data using the Moses tokenizer and truecaser (Koehn et al., 2007).

4.2 Implementation

The models are implemented in Neural Monkey (Helcl and Libovický, 2017). They are trained using Adam (Kingma and Ba, 2015) and have minibatch size 40, RNN size 512, and dropout probability 0.2 (Gal and Ghahramani, 2016). We train to convergence on the validation set, using BLEU (Papineni et al., 2002) as the metric.

For sequential inputs and outputs, the maximum sentence length is 50 subwords. For parsed inputs, we increase maximum sentence length to 150 subwords to account for the increased length due to the parsing labels; we still use a maximum output length of 50 subwords for these systems.

	System	**2016**	**2017**
baseline	seq2seq	25.0	20.8
	parse2seq	25.4	20.9
proposed	multi-source lex	**26.5**	**21.9**
	multi-source unlex	26.4	21.7

Table 2: BLEU scores on newstest2016 and newstest2017 datasets for the baselines, unlexicalized (unlex), and lexicalized (lex) systems.

4.3 Baselines

Seq2seq

The proposed models are compared against two baselines. The first, referred to here as *seq2seq*, is the standard RNN-based neural machine translation system with attention (Bahdanau et al., 2015). This baseline does not use the parsed data.

Parse2seq

The second baseline we consider is a slight modification of the mixed RNN model proposed by Li et al. (2017). This uses an identical architecture to the seq2seq baseline (except for a longer maximum sentence length in the encoder). Instead of using sequential data on the source side, the linearized parses are used. We allow the system to attend equally to words and node labels on the source side, rather than restricting the attention to words. We refer to this baseline as *parse2seq*.

5 Results

Table 2 shows the performance on EN→DE translation for each of the proposed systems and the baselines, as approximated by BLEU score.

The multi-source systems improve strongly over both baselines, with improvements of up to 1.5 BLEU over the seq2seq baseline and up to 1.1 BLEU over the parse2seq baseline. In addition, the lexicalized multi-source systems yields slightly higher BLEU scores than the unlexicalized multi-source systems; this is surprising because the lexicalized systems have significantly longer sequences than the unlexicalized ones. Finally, it is interesting to compare the seq2seq and parse2seq baselines. Parse2seq outperforms

System	Source Data	2016	2017
parse2seq	seq	0.6	0.5
multi-source lex	seq + seq	23.6	20.0
	seq + null	23.1	19.3
multi-source unlex	seq + seq	**23.7**	19.9
	seq + null	23.6	**20.9**

Table 3: BLEU scores on newstest2016 and newstest2017 when no parsed data is used during inference.

seq2seq by only a small amount compared to multi-source; thus, while adding syntax to NMT can be helpful, some ways of doing so are more effective than others.

6 Analysis

6.1 Inference Without Parsed Sentences

The parse2seq and multi-source systems require parsed source data at inference time. However, the parser may fail on an input sentence. Therefore, we examine how well these systems do when given only unparsed source sentences at test time.

Table 3 displays the results of these experiments. For the parse2seq baseline, we use only sequential (*seq*) data as input. For the lexicalized and unlexicalized multi-source systems, two options are considered: *seq* + *seq* uses identical sequential data as input to both encoders, while *seq* + *null* uses null input for the parsed encoder, where every source sentence is “( )”.

The parse2seq system fails when given only sequential source data. On the other hand, both multi-source systems perform reasonably well without parsed data, although the BLEU scores are worse than multi-source with parsed data.

6.2 BLEU by Sentence Length

For models that use source-side linearized parses (multi-source and parse2seq), the source sequences are significantly longer than for the seq2seq baseline. Since NMT already performs relatively poorly on long sentences (Bahdanau et al., 2015), adding linearized source parses may exacerbate this issue. To detect whether this occurs, we calculate BLEU by sentence length.

We bucket the sentences in newstest2017 by source sentence length. We then compute BLEU scores for each bucket for the seq2seq and parse2seq baselines and the lexicalized multi-source system. The results are in Figure 1.

In line with previous work on NMT on long sentences (Bahdanau et al., 2015; Li et al., 2017), we see a significant deterioration in BLEU for longer sentences for all systems. In particular, although the parse2seq model outperformed the seq2seq model overall, it does worse than seq2seq for sentences containing more than 30 words. This indicates that parse2seq performance does indeed suffer due to its long sequences. On the other hand, the multi-source system outperforms the seq2seq baseline for all sentence lengths and does particularly well for sentences with over 50 words. This may be because the multi-source system has both sequential and parsed input, so it can rely more on sequential input for very long sentences.

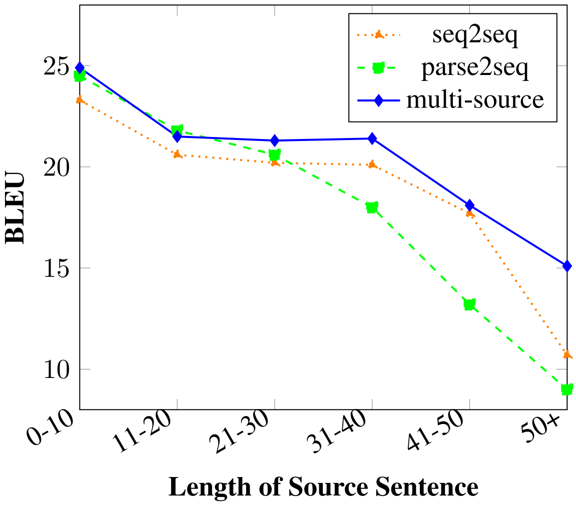

Figure 1: BLEU by sentence length on newstest2017 for baselines and lexicalized multi-source.

7 Conclusion

In this paper, we presented a multi-source method for effectively incorporating linearized parses of the source data into neural machine translation. This method, in which the parsed and sequential versions of the sentence were both taken as input during training and inference, resulted in gains of up to 1.5 BLEU on EN→DE translation. In addition, unlike parse2seq, the multi-source model translated reasonably well even when the source sentence was not parsed.

In the future, we will explore adding back-translated (Sennrich et al., 2016a) or copied (Currey et al., 2017) target data to our multi-source system. The multi-source model does not require all training data to be parsed; thus, monolingual data can be used even if the parser is unreliable for the synthetic or copied source sentences.

Acknowledgments

This work was funded by the Amazon Academic Research Awards program.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 132–140. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967. Association for Computational Linguistics.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: A case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734. Association for Computational Linguistics.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336. Association for Computational Linguistics.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156. Association for Computational Linguistics.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 823–833. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of NAACL-HLT*, pages 866–875. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29*, pages 1019–1027.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 125–135. Association for Computational Linguistics.

Jindřich Helcl and Jindřich Libovickỳ. 2017. Neural Monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, 107(1):5–17.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 177–180. Association for Computational Linguistics.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 688–697. Association for Computational Linguistics.

Jindřich Libovickỳ and Jindřich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 196–202. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *4th International Conference on Learning Representations*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the ACL*, pages 55–60. Association for Computational Linguistics.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Predicting target language CCG supertags improves neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 68–79. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 86–96. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 1715–1725. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566. Association for Computational Linguistics.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*, pages 2773–2781.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of NAACL-HLT*, pages 30–34. Association for Computational Linguistics.

Fixing Translation Divergences in Parallel Corpora for Neural MT

MinhQuang Pham[†‡], **Josep Crego**[†], **Jean Senellart**[†] , **François Yvon**[‡]
[†]SYSTRAN / 5 rue Feydeau, 75002 Paris, France
`firstname.lastname@systrangroup.com`
[‡]LIMSI, CNRS, Université Paris-Saclay 91405 Orsay, France
`firstname.lastname@limsi.fr`

Abstract

Corpus-based approaches to machine translation rely on the availability of clean parallel corpora. Such resources are scarce, and because of the automatic processes involved in their preparation, they are often noisy. This paper describes an unsupervised method for detecting translation divergences in parallel sentences. We rely on a neural network that computes cross-lingual sentence similarity scores, which are then used to effectively filter out divergent translations. Furthermore, similarity scores predicted by the network are used to identify and fix some partial divergences, yielding additional parallel segments. We evaluate these methods for English-French and English-German machine translation tasks, and show that using filtered/corrected corpora actually improves MT performance.

1 Introduction

Parallel sentence pairs are the only necessary resource to build Machine Translation (MT) systems. In the case of neural MT, a large neural network is trained through maximising a proxy of translation performance on a parallel corpus. Therefore, the quality of MT engines is heavily dependent on the amount but also the quality of available parallel sentences.[1]

Parallel texts are unfortunately, scarce resources: There are relatively few language pairs for which parallel corpora of large sizes exist, and even for those pairs, available corpora only concern few restricted domains. To alleviate the lack of parallel data, several approaches have been developed over the years. They range from methods using non-parallel, or comparable data (Zhao and Vogel, 2002; Fung and Cheung, 2004; Munteanu and Marcu, 2005; Grégoire and Langlais, 2018; Grover and Mitra, 2017; Schwenk, 2018) to techniques that produce synthetic parallel data from monolingual corpora (Sennrich et al., 2016a; Chinea-Rios et al., 2017), using automated alignment/translation engines that are prone to the introduction of noise in the resulting parallel sentences. Mismatches in parallel sentences extracted from translated texts are also reported (Tiedemann, 2011; Xu and Yvon, 2016). This problem is mostly ignored in MT, where parallel sentences are considered to convey the exact same meaning; yet it seems particularly important for neural MT engines (Chen et al., 2016).

`en`	*What do you feel*, **Spock**?
`fr`	*Que ressentez-vous?*
`gl`	*What do you feel?*
`en`	*How much do you get paid?*
`fr`	*T'es payé combien* **de l'heure**?
`gl`	*How much do you get paid per hour?*
`en`	**That seems a lot.**
`fr`	**40 livres?**
`gl`	*40 pounds?*
`en`	*I brought you* **french fries**!
`fr`	*Je t'ai rapporté des* **saucisses**!
`gl`	*I brought you sausage!*

Table 1: Examples of semantically divergent parallel sentences. English (`en`), French (`fr`) and gloss of French (`gl`). Divergences are in bold letters.

Table 1 gives some examples of English-French parallel sentences that are not completely semantically equivalent, extracted from the OpenSubtitles corpus (Lison and Tiedemann, 2016).

Multiples types of translation divergences are found in parallel corpora: Additional segments are included on either side of the parallel sentences (first and second rows) most likely due to errors in sentence segmentation; Some translations may be completely uncorrelated (third row); Inaccurate translations also exist (fourth row). Note that divergent translations can be due various reasons (Li et al., 2014), the study of which is beyond the

---

[1] Recent work on neural MT (Lample et al., 2018; Artetxe et al., 2018) completely dispenses with parallel data, using unsupervised methods to obtain performance improvements over word-by-word statistical MT. These systems however lag far behind supervised systems, as considered in this work.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2967–2973
Brussels, Belgium, October 31 - November 4, 2018.

scope of this paper.

In this work, we present an unsupervised method for building cross-lingual sentence embeddings based on modelling word similarity, relying on a neural architecture (see § 3) that is able to identify several types of common cross-lingual divergences. The resulting embeddings are then used to measure semantic equivalence between sentences. To evaluate our method, we show in § 4 that translation accuracy can be improved after filtering out divergent sentence pairs in an English-to-French and an English-to-German translation tasks. We also show that in some cases, divergent sentences can be fixed by removing divergent segments, further increasing translation quality. All the code used in this paper is freely available.[2]

2 Related Work

Attempts to measure the impact of translation divergences in MT have focused on the introduction of noise in sentence alignments (Goutte et al., 2012), showing that statistical MT is highly robust to noise, and that performance only degrades seriously at very high noise levels. In contrast, neural MTs seem to be more sensitive to noise (Chen et al., 2016), as they tend to assign high probabilities to rare events (Hassan et al., 2018).

Efforts devoted to characterising the degree of semantic equivalence between two snippets of texts in the same or different languages are presented (Agirre et al., 2016). In (Mueller and Thyagarajan, 2016), a monolingual sentence similarity network is proposed, making use of a simple LSTM layer to compute sentence representations. The authors show that a simple SVM classifier exploiting such sentence representations achieves state-of-the-art results in a textual entailment task. With the same objective, the system of He and Lin (2016) uses multiple convolutional layers and models pairwise word interactions.

Our work is inspired by Carpuat et al. (2017), who train a SVM-based cross-lingual divergence detector using word alignments and sentence length features. Their work shows that an NMT system trained only on non-divergent sentences yields slightly better translation scores, while requiring less training time. A follow-up study by the same authors (Vyas et al., 2018) achieves even better results, using the neural architecture of He and Lin (2016). Our work differs from theirs as we make use of a network with a different, arguably simpler, topology. We model sentence similarity by means of optimising a loss function based on word alignments. Furthermore, the network predicts word similarity scores that we further use to correct divergent sentences.

3 Neural Divergence Classifier

The architecture of our network is inspired by the work on word alignment of Legrand et al. (2016), using however contextual, rather than fixed, word embeddings (see Figure 1).

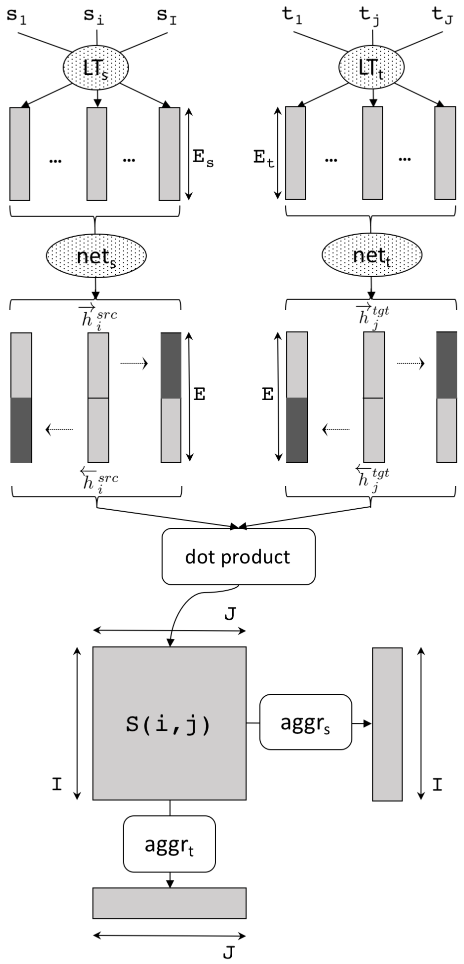

Figure 1: Illustration of the model.

It computes the similarity of any source-target sentence pair (s, t), where $s = (s_1, ..., s_I)$ and $t = (t_1, ..., t_J)$. The model is composed of 2 bi-directional LSTM subnetworks, net_s and net_t, which respectively encode source and target sentences. Since both net_s and net_t take the same form, we only describe the former network: it outputs forward and backward hidden states, $\overrightarrow{h}_i^{src}$ and $\overleftarrow{h}_i^{src}$, which are then concate-

---

[2] https://github.com/jmcrego/similarity

nated into a vector encoding the i^{th} source word as $h_i^{src} = [\overrightarrow{h}_i^{src}; \overleftarrow{h}_i^{src}]$. In addition, the last forward/backward hidden states (in dark grey on Figure 1) are also concatenated to represent whole sentences $h_{src} = [\overrightarrow{h}_I^{src}; \overleftarrow{h}_1^{src}]$. The similarity between sentence pairs can then be obtained using eg. the cosine similarity:

$$sim(h_{src}, h_{tgt}) = \frac{h_{src} \cdot h_{tgt}}{||h_{src}|| * ||h_{tgt}||} \tag{1}$$

Our model is trained to maximize word alignment scores between words in both sentences, using aggregation functions that summarise the alignment scores for each source/target word. Similar to (Legrand et al., 2016), alignment scores $S(i,j)$ are given by the dot-product $S(i,j) = h_i^{src} \cdot h_j^{tgt}$, further aggregated as follows:

$$\begin{aligned} aggr_s(i, S) &= \frac{1}{r}\, log \left(\sum_{j=1}^{J} e^{r * S(i,j)} \right) \\ aggr_t(j, S) &= \frac{1}{r}\, log \left(\sum_{i=1}^{I} e^{r * S(i,j)} \right) \end{aligned} \tag{2}$$

The training loss function is then defined as:

$$\begin{aligned} &\mathcal{L}(src, tgt) = \\ &\quad \sum_{i=1}^{I} log(1 + e^{aggr_s(i,S) * sign_i}) \ + \\ &\quad + \sum_{j=1}^{J} log(1 + e^{aggr_t(j,S) * sign_j}) \end{aligned} \tag{3}$$

3.1 Training with Negative Examples

Training is performed by minimizing Eq. (3), for which annotated examples of source (sign_i) and target (sign_j) words are needed. As positive examples, we use paired sentences of a parallel corpus; all words in such sentences are labelled as parallel ($\forall i, j, \text{sign}_i = \text{sign}_j = -1$). We consider three types of negative instances: the basic case uses random unpaired sentences; in this case, all words are labelled as divergent ($\forall i, j, \text{sign}_i = \text{sign}_j = +1.$). Since negative pairs may be very easy to classify and we want our network to detect less obvious divergences, we further create more difficult negative examples as follows.

We first replace random sequences of words in source or target by a sequence of words with the same part-of-speeches.[3] Words that are not replaced are deemed parallel ($\text{sign}_i = -1$) while those replaced are annotated as $\text{sign}_i = +1$. Words aligned to some replaced words are also assigned the divergent label ($\text{sign}_i = +1$). For instance, given the original sentence pair:

src: `What do you feel ?`
tgt: `Que ressentez-vous ?` ,

we may replace '`you feel`', with part-of-speech tags '`PRP VB`', by another sequence with same tags (i.e. '`we want`'), yielding a new negative instance (divergent words are in bold):

src:	`What`	`do`	**we**	**want**	`?`
$\mathcal{Y}^{src}$:	`-1`	`-1`	**+1**	**+1**	`-1`
tgt:	`Que`	**ressentez-vous**			`?`
$\mathcal{Y}^{tgt}$:	`-1`	**+1**			`-1`

Note that we need word alignments to identify as divergent the sequence '`ressentez-vous`', which was aligned to '`you feel`' in the original sentence. Finally, motivated by sentence segmentation errors observed in many corpora, we also build negative examples by inserting a sentence at the beginning (or end) of the source (or target) sentence. Words in the original sentence pair are annotated $\text{sign}_i = -1$, while the new words inserted are considered divergent ($\text{sign}_i = +1$). Given the same sentence pair as above, a negative example is created by inserting the sentence '`Not .`' at the end of the original source:

src:	`What`	`do`	`you`	`want`	`?`	**Not**	**.**
$\mathcal{Y}^{src}$:	`-1`	`-1`	`-1`	`-1`	`-1`	**+1**	**+1**
tgt:	`Que`	`ressentez-vous`			`?`		
$\mathcal{Y}^{tgt}$:	`-1`	`-1`			`-1`		

To finally avoid the generation of easy negative sentence pairs having a large difference in sentence length, we restrict negative examples to have a length ratio < 2.0 (3.0 for shortest sentences).

3.2 Divergence Correction

Our training corpora contains many divergent sentences that follow a common pattern, consisting of adding some extra leading/trailing words. Accordingly, we implemented a simple algorithm that discards sequences of leading/trailing words on both

---

[3] The rationale is to try to keep the generated sentences as grammatical as possible; Otherwise, the network could learn to flag non-grammatical sentences as non-parallel.

sides. To find optimal source (u, v) and target (x, y) indices that enclose parallel segments within the original sentence, we compute:

$$\underset{u,v,x,y}{\arg\max}\left\{ \sum_{u \le I \le v} \max_{x \le j \le y} \{S(i,j)\} \right\}$$

The $\mathcal{N}$-best sequences (s_u^v, t_x^y) are considered as likely corrections, in which we use the one having the highest similarity score to replace the original (s_1^I, t_1^J). Note that short sentences are not considered and we enforce $v - u > \tau$ and $y - x > \tau$. Figure 2 (left) displays an example of an alignment matrix $S(i, j)$. An acceptable correction is: *Que ressentez-vous ? ⇔ What do you feel ?*. corresponding to $u = 1$, $v = 5$, $x = 1$ and $y = 3$.

4 Experiments

4.1 Corpora

We filter out divergences from the English-French OpenSubtitles corpus (Lison and Tiedemann, 2016), which consists of a collection of movie and TV subtitles. We also use the very noisy English-German Paracrawl[4] corpus. Both corpora present many potential divergences. To evaluate English-French translation performance, we use the En-Fr Microsoft Spoken Language Translation corpus, created from actual Skype conversations (Federmann and Lewis, 2016). English-German performance is evaluated on the publicly available Newstest-2017 (Bojar et al., 2017), corresponding to news stories selected from online sources.

In order to better assess the quality of our classifier when facing different word divergences, we also collected from the original OpenSubtitles corpus 500 sentences containing different types of examples: 200 paired sentences; 100 unpaired sentences; 100 sentences with replace examples; and 100 sentences with insert examples (see § 3.1). All data is preprocessed with `OpenNMT`[5], performing minimal tokenisation. After tokenisation, each out-of-vocabulary word is mapped to a special UNK token, assuming a vocabulary containing the $50,000$ more frequent words.

4.2 Neural Divergence

Word embeddings of $E_s = E_t = 256$ cells are initialised using `fastText`,[6] further aligned by means of `MUSE`[7] following the unsupervised method of Lample et al. (2018). Both bi-LSTMs use 256-dimensional hidden representations ($E = 512$). Network optimization is done using SGD with gradient clipping (Pascanu et al., 2013). For each epoch, we randomly select 1 million sentence pairs that we place in batches of 32 examples. We run 10 epochs and start decaying at each epoch by 0.8 when the loss on validation set increases. Divergence is computed as in equation (1) and setting $r = 1.0$; For divergence correction, we use $\mathcal{N} = 20$ and $\tau = 3$. The same number of examples are always generated for each type of example (`Paired`, `Unpaired`, `Replace` and `Insert`). Alignments needed for `Replace` and `Insert` methods are performed using `fast_align`[8].

4.3 Neural Translation

In addition to the basic tokenisation detailed above, we perform Byte-Pair Encoding (Sennrich et al., 2016b) with 30000 merge operations learned by joining both language sides. Neural systems are based on the open-source project `OpenNMT`; using a Transformer model similar to the model of Vaswani et al. (2017): both encoder and decoder have 6 layers; Multi-head attention is performed over 8 head; the hidden layer size is 512; and the inner layer of feed forward network is of size 2048. Word embeddings have 512 cells. We set the dropout probability to 0.1 and the batch size to 3072. The optimiser is Lazy Adam with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$, $warmup\_steps = 4000$. Training stops after 30 epochs.

5 Results

We first evaluate the ability of our divergence classifier to predict different types of divergences at the level of words. We use the test set manually annotated for that purpose and train our model on the OpenSubtitles corpus. A word is considered divergent when associated to a negative aggregation score (see Equation (2)). Accuracies obtained for various combinations of negative examples, where we see that non-divergent words in parallel and unpaired sentences (columns `P` and `U`) are easy to spot, as long as the model has seen these types of examples in training. However, the accuracy drops dramatically when the model is not trained with unpaired sentences (rows `PR`, `PI` and `PRI`). Regarding columns `R` and `I`, accuracies are lower

---

[4] http://paracrawl.eu/

[5] http://opennmt.net

[6] https://github.com/facebookresearch/fastText

[7] https://github.com/facebookresearch/MUSE

[8] https://github.com/clab/fast_align

since these sentences contain a mix of divergent and non-divergent words.

	Accuracy	Test examples				
		`P`	`U`	`R`	`I`	`PURI`
Train examples	`PU`	**0.996**	**0.994**	0.671	0.673	0.874
	`PR`	**0.995**	0.033	**0.951**	0.689	0.746
	`PI`	**0.998**	0.071	0.697	**0.725**	0.705
	`PUR`	**0.994**	**0.989**	**0.919**	0.710	0.932
	`PUI`	**0.995**	**0.996**	0.662	**0.769**	0.887
	`PRI`	**0.991**	0.161	**0.924**	**0.719**	0.768
	`PURI`	**0.995**	**0.980**	**0.916**	**0.788**	**0.942**

Table 2: Word divergence accuracies according to different type of examples used in train/test.

Models that were trained with the matching examples (`R` and `I`) obtain the highest accuracies (in bold letters). Column `PURI` gives results for the complete test set, mixing all type of examples. As expected, the best accuracy is also obtained when training on all types of examples.

Figure 2 illustrates the output of our network when trained using `PU` examples (right) and `PURI` examples (left). The former (right) fails to predict some divergences, most likely because its training set does not contain sentences mixing divergent and non-divergent words. Furthermore, the network trained with `PURI` examples correctly assigns a lower similarity score to this pair, as both sentences do not convey the exact same meaning.

	Que	ressentez-vous	?		
	+6.8	+6.4	+5.1		
What	+6.0	+2.1	-0.5	+6.0	What
do	+5.5	+4.1	+1.3	+5.7	do
you	+5.2	+5.2	+2.3	+5.9	you
feel	+4.1	+5.9	+3.1	+6.1	feel
?	-0.6	+1.3	+4.9	+4.9	?
Not	-3.4	-2.5	-0.2	-0.1	Not
.	-3.1	-2.9	-0.3	-0.2	.

+0.1201

	Que	ressentez-vous	?		
	+3.9	+3.9	+4.4		
What	+3.2	+0.9	-1.1	+3.3	What
do	+2.5	+2.4	+0.8	+3.2	do
you	+2.1	+2.8	+1.9	+3.4	you
feel	+1.0	+3.0	+2.7	+3.6	feel
?	-1.2	+0.2	+3.1	+3.1	?
Not	-1.7	-0.5	+3.1	+3.1	Not
.	-2.0	-1.4	+2.7	+2.7	.

+0.2036

Figure 2: Sentence pair with similarity scores produced by our model when trained with `PU` examples (right) and over `PURI` examples (left). Aggregation scores (Eq. (2)) are shown next to words. Matrices contain alignment scores. Sentence similarities (Eq. (1)) are below matrices.

Finally, BLEU scores obtained with varying training data configurations are in Table 3: The entire[9] data sets (`all`); The most similar pairs after optimizing Eq. (3) (`sim`); After applying the correction algorithm of § 3.2 (`sim+fix`). Columns Ref and Fix indicate the number of original and corrected sentences (in millions) used in training.

Data	**Ref (M)**	**Fix (M)**	**Test (BLEU)**
OpenSubtitles English-French			
`all`	27.2	-	42.18
`sim`	15.5	-	43.12 (+0.94)
`sim`	18.0	-	43.19 (+1.01)
`sim+fix`	15.5	2.5	**44.19** (+2.01)
Paracrawl English-German			
`all`	22.2	-	19.27
`sim`	15.0	-	21.52 (+2.25)
`sim`	17.5	-	21.97 (+2.70)
`sim+fix`	15.0	2.5	**22.42** (+3.15)

Table 3: BLEU scores obtained by neural MT using different subsets of the OpenSubtitles and Paracrawl corpora.

Results obtained after filtering sentence pairs (`sim`) clearly outperform the baseline (`all`) by +0.94 and +2.25 BLEU respectively. Regarding OpenSubtitles, when fixing 2.5M sentences (4^{th} row) the accuracy is further boosted to +2.01, whereas the same sentence pairs do not show any improvement when added in their original form (3^{rd} row). Similar results are obtained for the Paracrawl corpus. Results after fixing 2.5M sentences ($4^{th} row$) outperform those obtained with their original form ($3^{rd} row$).

6 Conclusions and outlook

We presented an unsupervised method based on deep neural networks for detecting translation divergences in parallel corpora. Our model optimizes word alignments, and computes a fine grained divergence prediction at the level of words. Misaligned/divergent words can then be filtered out, yielding larger and better training sets. Our experiments on two machine translation tasks show significant improvements in comparison to training with the entire data set.

We plan to use our model to predict sentence embeddings over monolingual corpora, allowing to collect parallel pairs through vector similarity measures. In addition, we would like to measure the performance of our model after applying subword tokenisation, as well as using multiple LSTM layers, a technique well known to capture hierarchical structure in the context of MT.

Acknowledgements

We are grateful to J. Legrand for his fruitful comments regarding the neural divergence classifier.

---

[9] Paracrawl contains more than 100M sentences. We reduced its size to 22.2M using standard filtering techniques.

References

Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval@NAACL-HLT*, pages 497–511. The Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of the Sixth International Conference on Learning Representations*, Vancouver, Canada.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Marine Carpuat, Yogarshi Vyas, and Xing Niu. 2017. Detecting cross-lingual semantic divergence for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 69–79, Vancouver. Association for Computational Linguistics.

Boxing Chen, Roland Kuhn, George Foster, Colin Cherry, and Fei Huang. 2016. Bilingual methods for adaptive training data selection for machine translation. In *Proceedings of the 12th biennial conference of the Association for the Machine Translation in Americas (AMTA2016)*, Austin, TX.

Mara Chinea-Rios, Álvaro Peris, and Francisco Casacuberta. 2017. Adapting neural machine translation with parallel synthetic data. In *Proceedings of the Second Conference on Machine Translation*, pages 138–147, Copenhagen, Denmark. Association for Computational Linguistics.

Christian Federmann and Will Lewis. 2016. Microsoft speech language translation (MSLT) corpus: The IWSLT 2016 release for English, French and German. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT2016)*, Seattle, WA.

Pascale Fung and Percy Cheung. 2004. Mining very-non-parallel corpora: Parallel sentence and lexicon extraction via bootstrapping and EM. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, EMNLP.

Cyril Goutte, marine carpuat, and Georges Foster. 2012. The impact of sentence alignment errors on phrase-based machine translation performance. In *The Tenth Biennial Conference of the Association for Machine Translation in the Americas*, San Diego, California.

Francis Grégoire and Philippe Langlais. 2018. Extracting parallel sentences with bidirectional recurrent neural networks to improve machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1442–1453.

Jeenu Grover and Pabitra Mitra. 2017. Bilingual word embeddings with bucketed CNN for parallel sentence extraction. In *Proceedings of ACL 2017, Student Research Workshop*, pages 11–16, Vancouver, Canada. Association for Computational Linguistics.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic Chinese to English news translation. *CoRR*, abs/1803.05567.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, San Diego, California. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*, Long Beach, CA.

Joël Legrand, Michael Auli, and Ronan Collobert. 2016. Neural network-based word alignment through score aggregation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 66–73, Berlin, Germany. Association for Computational Linguistics.

Junyi Jessy Li, Marine Carpuat, and Ani Nenkova. 2014. Cross-lingual discourse relation analysis: A corpus study and a semi-supervised classification system. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 577–587, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *10th International Conference on Language Resources and Evaluation (LREC 2016)*, Portoroz, Slovenia. European Language Resources Association (ELRA).

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2786–2792, Phoenix, Arizona. AAAI Press.

Dragos S. Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–III–1318, Atlanta, GA, USA. JMLR.org.

Holger Schwenk. 2018. Filtering and mining parallel data in a joint multilingual space. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–234, Melbourne, Australia. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Jörg Tiedemann. 2011. *Bitext Alignment*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, Long Beach, CA.

Yogarshi Vyas, Xing Niu, and Marine Carpuat. 2018. Identifying semantic divergences in parallel text without annotations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1503–1515, New Orleans, Louisiana. Association for Computational Linguistics.

Yong Xu and François Yvon. 2016. Novel elicitation and annotation schemes for sentential and sub-sentential alignments of bitexts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portoroz, Slovenia. European Language Resources Association (ELRA).

Bing Zhao and Stephan Vogel. 2002. Adaptive parallel sentences mining from web bilingual news collection. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, Washington, DC, USA. IEEE Computer Society.

Adversarial Evaluation of Multimodal Machine Translation

Desmond Elliott[*]
Department of Computer Science
University of Copenhagen
`de@di.ku.dk`

Abstract

The promise of combining vision and language in multimodal machine translation is that systems will produce better translations by leveraging the image data. However, inconsistent results have lead to uncertainty about whether the images actually improve translation quality. We present an adversarial evaluation method to directly examine the utility of the image data in this task. Our evaluation measures whether multimodal translation systems perform better given either the congruent image or a random incongruent image, in addition to the correct source language sentence. We find that two out of three publicly available systems are sensitive to this perturbation of the data, and recommend that all systems pass this evaluation in the future.

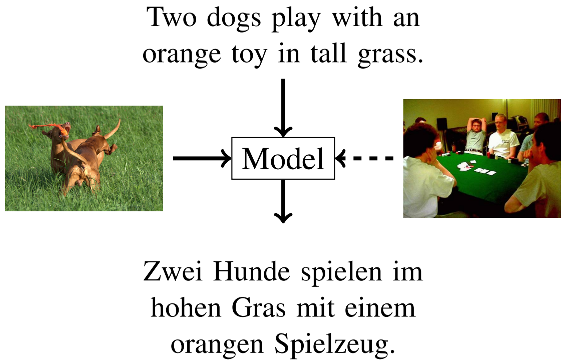

Figure 1: An adversarial evaluation for multimodal translation. We measure the difference in performance when a model sees a congruent image (left) or an incongruent image (right).

1 Introduction

Multimodal machine translation is the task of translating sentences situated in a visual context, such as captioned images on social media. The core argument of this area of research is that we can produce better translations by exploiting both the source language sentence and the visual context (Elliott et al., 2015; Hitschler et al., 2016). There is some evidence to support this argument for human translation: Frank et al. (2018) found that 13% of the German evaluation data in the Multi30K dataset (Elliott et al., 2016) needed at least one post-edit to reflect the joint meaning of the visual and linguistic context. However, the evidence that visual context helps computational models is less clear. Consider the three teams that submitted contrastive multimodal and text-only variants of their systems to the 2017 Multimodal Translation Shared Task (Elliott et al., 2017): the University of Le Mans' multimodal system outperformed their text-only variant (Caglayan et al., 2017); the Oregon State University text-only system outperformed their multimodal variant (Ma et al., 2017); and the performance of the Charles University systems depended on the language pair (Libovický and Helcl, 2017). In light of these results, we need a better understanding of the role of visual context in multimodal translation systems.

We propose an adversarial evaluation Method to determine whether multimodal translation systems are aware of the visual context. We introduce a measure of *image awareness* to quantify the difference in performance in two settings: (i) when a system is presented with congruent visual data; (ii) when it is presented with incongruent visual data. In both settings, a system is presented with the correct source language sentence. See Figure 1 for an illustration of our evaluation. We hypothesise that if a system is aware of the visual context, i.e. it is actually using the image for translation, then the system will perform better when it is presented with the congruent visual data than incongruent visual data. Our evaluation is related

---

[*] Work carried out at the University of Edinburgh.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2974–2978
Brussels, Belgium, October 31 - November 4, 2018.

to the foiled image captions evaluation, in which the performance of an image captioning system is measured when a single word is replaced with an incorrect, but similar word (Shekhar et al., 2017); the main difference is that we *replace* the visual data instead of manipulating the text. Our work is also related to a study of question-answering systems, in which additional text was appended to the end of a document (Jia and Liang, 2017). They found that these additional text segments distracted QA systems from producing the correct answer. In contrast, our evaluation does not manipulate the textual data, instead we replace the original visual input with a random distractor.

We evaluate three publicly available multimodal translation systems with our adversarial evaluation. The main finding of this paper is that one publicly available multimodal translation system is not aware of the congruent image data. This finding raises doubts about whether state-of-the-art multimodal translation systems actually use the visual context to produce better translations. We conclude this paper by discussing whether this is likely to be due to problems with the data or with the model architectures.

2 Adversarial Evaluation

2.1 Image Awareness

We propose an adversarial evaluation method for multimodal machine translation. This method measures how a system performs when it is presented with the correct text data and either the congruent image or with an incongruent image. In this section we define two image awareness functions to measure whether a multimodal translation system is aware of the congruent visual data.

Let x be a source language sentence, y be a target language sentence, v be the congruent image, and $\bar{v}$ be an incongruent image. Image awareness is calculated using an evaluable performance measure $\mathcal{E}$. The overall image awareness of a model $\mathcal{M}$ on an evaluation dataset $\mathcal{D}$ is:

$$\Delta\text{-Awareness} = \frac{1}{|\mathcal{D}|} \sum_{i}^{|\mathcal{D}|} a_{\mathcal{M}}(x_i, y_i, v_i, \bar{v}_i) \quad (1)$$

The image awareness of a model $\mathcal{M}$ for a single instance $a_{\mathcal{M}}(x_i, y_i, v_i, \bar{v}_i)$ is given by:

$$a_{\mathcal{M}}(x_i, y_i, v_i, \bar{v}_i) = \mathcal{E}(x_i, y_i, v_i) - \mathcal{E}(x_i, y_i, \bar{v}_i) \quad (2)$$

Under this definition, the output of the evaluable performance measure should be higher in the presence of the congruent data than the incongruent data, i.e. $\mathcal{E}(x_i, y_i, v_i) > \mathcal{E}(x_i, y_i, \bar{v}_i)$.[1] If this is the case, on average, then the overall image awareness of a model Δ-Awareness is positive. This can only happen when model outputs are evaluated more favourably in the presence of the the congruent image data than the incongruent image data.

2.2 Model-internal awareness Δ_I

A model-internal image measure of awareness is the difference in the probability assigned to the target language sentence y in the congruent and incongruent conditions. This is model-internal because it has the same form as the maximum-likelihood objective used to train the translation model. In this case, $\mathcal{E} = p(y|x, \cdot)$, and the difference in performance for a single instance is:

$$a_{\mathcal{M}} = \Delta_I = p(y_i|x_i, v_i) - p(y_i|x_i, \bar{v}_i) \quad (3)$$

2.3 Model-external awareness Δ_E

A model-external awareness measure could be a text-similarity evaluation or human judgement. In this paper, we use the Meteor text-similarity score (Denkowski and Lavie, 2014) because it naturally decomposes to the sentence level, and it is already the de-facto evaluation metric for multimodal machine translation (Specia et al., 2016). Let $\mathcal{E}$ be any text-similarity scoring function T that decomposes to the sentence level. The difference in performance for a single instance is defined as:

$$a_{\mathcal{M}} = \Delta_E = T(x_i, y_i, v_i) - T(x_i, y_i, \bar{v}_i) \quad (4)$$

3 Systems Evaluation

We evaluate the image awareness of three pretrained multimodal translations systems that we received by direct correspondence:

decinit: The initial state of the decoder network is set with a learned transformation of the visual data (Caglayan et al., 2017).

trgmul: The target language word embeddings are modulated by an element-wise multiplication with a learned transformation of the visual data (Caglayan et al., 2017).

---

[1] This assumes that a higher score means better performance for the performance measure $\mathcal{E}$. Swap the order of the operands if lower performance means better.

	C	**I**	Δ_E-Awareness
trgmul	57.3	57.3 $\pm$ 0.2	-0.001 $\pm$ 0.002
decinit	57.0	56.8 $\pm$ 0.1	0.003 $\pm$ 0.001
hierattn	55.0	53.3 $\pm$ 0.3	0.019 $\pm$ 0.003

Table 1: Corpus-level Meteor scores in the **C**ongruent and **I**ncongruent settings, along with the Meteor-awareness results. Incongruent and Δ_E-Awareness scores are the mean and standard deviation of five permutations of the visual data.

hierattn: The decoder network learns to selectively attend to a combination of the source language and the visual data (Libovický and Helcl, 2017).

Each system was trained on the 29,000 English–German–image tuples in the Multi30K dataset (Elliott et al., 2016). We evaluate the image awareness of these systems using the 1,014 tuples in the validation data, which is typically used for model selection. We select the incongruent images $\bar{v}$ by randomly shuffling the order in which the images v are associated with the source language text x. In our evaluation, we report the mean and standard deviation of randomly shuffling the image data five times. The code to evaluate your own system is publicly available.[2]

3.1 Statistical test

To determine if a model passes the proposed evaluation, we conduct a non-parametric Wilcoxon signed-rank test of the following hypothesis:

> H_1: Congruent images *improve* the quality of multimodal translation compared to incongruent images.
>
> H_0: Congruent images *make no difference* to the quality of multimodal translation compared to incongruent images.

We conduct this statistical test using the pairs of values that are calculated in the process of computing the the image awareness scores (Eq. 2), i.e. $\mathcal{E}(x_i, y_i, v_i)$ and $\mathcal{E}(x_i, y_i, \bar{v}_i)$.

We combine the k=5 separate p values from each test using Fisher's method and reject the null hypothesis H_0 if the result of the χ^2 test with $2k$ degrees of freedom is $p \leq 0.005$.[3]

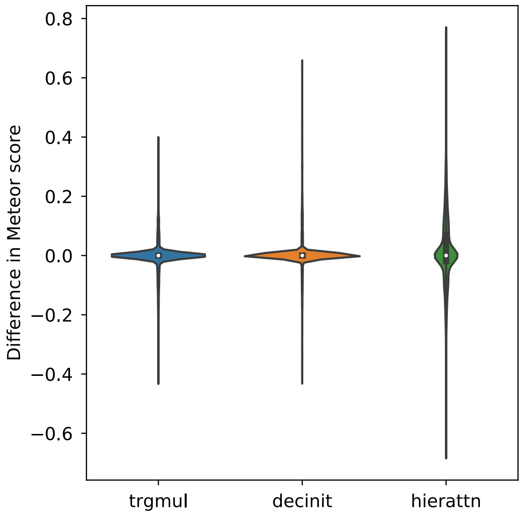

Figure 2: Violin plots of the Meteor-awareness scores for evaluated models. The white dot marks the median value, the thick gray bar shows the interquartile range, and the thin gray bar is the 95% confidence interval. The width of the plots show the kernel density estimate of the distributions.

3.2 Results

Table 1 shows the corpus-level results of a Meteor-based evaluation and the Meteor-awareness evaluation. We find that images improve the quality of the **hierattn** system (χ^2 = 136.74, p < 0.0001), and images also improve the quality of the **decinit** system (χ^2 = 32.79, $p = 0.0003$). Images make no difference to the quality of the translations generated by the **trgmul** system (χ^2 = 8.98, $p = 0.533$). To complement these tests, Figure 2 shows violin plots of the Meteor-awareness scores. These show that the translations generated by the **trgmul** and **decinit** systems are most likely to result in no difference in Meteor score between the congruent and incongruent conditions.

We now turn our attention to the results of the probability-awareness evaluation. Images improve the quality of the **trgmul** system (χ^2 = 52.55, p < 0.0001), and images also improve the quality of the **hierattn** system (χ^2 = 622.03, p < 0.0001). Images make no difference to the quality of **decinit** system (χ^2 = 6.49, p = 0.772).

Figure 3 shows examples of translations pro-

---

[2] `http://github.com/elliottd/awareness`

[3] We use a stricter threshold to reduce the chance of false positive findings (Benjamin et al., 2018).

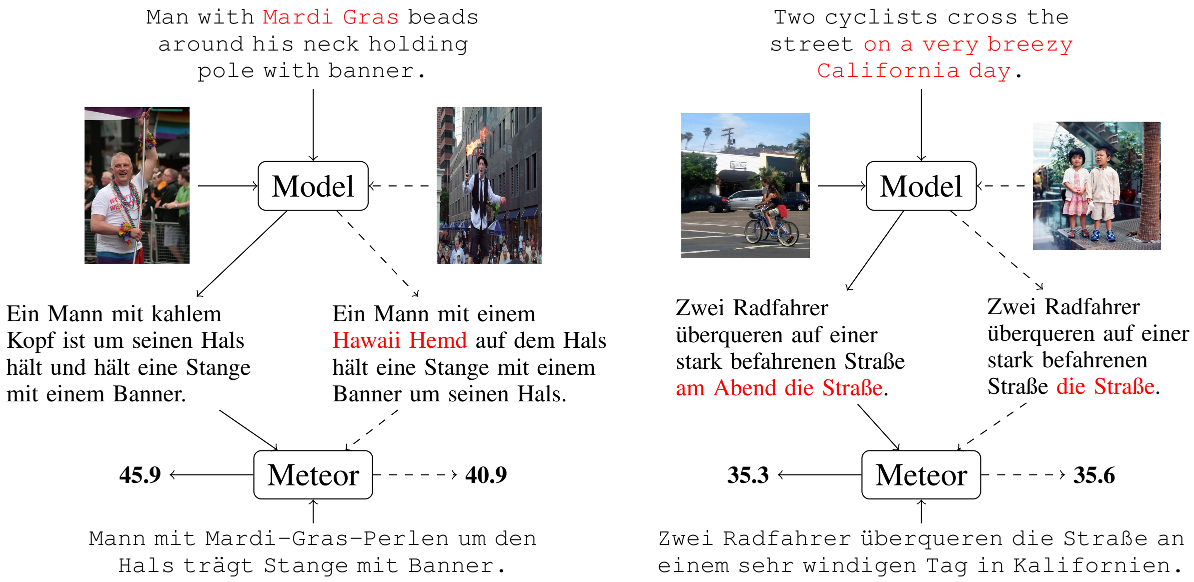

(a) Congruent is better than Incongruent

(b) Incongruent is better than Congruent

Figure 3: Examples of the difference in Meteor awareness for the **hierattn** system. In each example, the source sentence is shown at the top and the reference sentence is shown at the bottom, both in `Typewriter` font. The congruent image is on the left, and the incongruent image is on the right.

duced by the **hierattn** system for sentences paired with congruent / incongruent images. Figure 3 (a) shows an example with high positive difference in Meteor score. The incongruent image causes the translation system to refer to an unseen Hawaiian shirt. In neither setting does the system translate the phrase “Mardi Gras”. Figure 3 (b) shows an example with a negative difference in Meteor score. The congruent image results in a long translation with poor coverage of the reference, which Meteor punishes more severely than the shorter translation arising from the incongruent image. In neither setting does the model translate the prepositional phrase “on a very breezy California day”.

4 Discussion

4.1 Data problems

We posit that the current Multi30K training data does not necessarily require systems to use the visual context to solve the translation task. Elliott et al. (2016) note that the German translation data was produced without showing the translators the images, and Frank et al. (2018) found that 13% of the Multi30K test data needed to be post-edited to reflect the joint semantics of both modalities. We recommend that entirety of the German Multi30K training data should be post-edited so that future systems are more likely to require a joint understanding of the visual and linguistic context.[4] We note that a similar issue was found in a visual question answering dataset, resulting in the creation of a new “balanced” dataset (Goyal et al., 2017).

4.2 The role of model architectures

The key difference between the systems evaluated in this paper is how they use the visual context. The **hierattn** system learns a timestep-dependent context vector over a location-preserving 3D volume of image features, whereas the **trgmul** and **decinit** systems use an average-pool of the 3D location-preserving features. In our evaluation, the only system that is aware of the congruent image data for both types of image-awareness is the **hierattn** system that learns a spatial context over the image. Learning to attend to specific regions of the image may prove to be crucial to improving translations with visual context.

5 Conclusion

We proposed an adversarial evaluation method to determine whether multimodal translation systems are aware of the visual context. This evaluation method measures the difference in the perfor-

---

[4] We planned to repeat the adversarial evaluation with the Multi30K French data, which *was* created by showing the annotators the images. However, we did not receive pre-trained models for all the systems for English–French translation.

mance of a system given the congruent or an incongruent image as additional context. We found that two out of three publicly available multimodal translation systems were improved by the congruent visual context, when compared to the incongruent visual context. We encourage researchers to use this method to evaluate their own systems. Future work includes augment existing multimodal translation models with an additional adversarial objective that forces the model to perform better in the presence of the congruent image than a random incongruent image. We will also apply this evaluation method to other tasks that use additional context, e.g. images in visual-question answering, or part-of-speech tags in neural machine translation.

Acknowledgements

Barry Haddow asked if we knew whether the additional image data actually improved the quality of multimodal machine translation. I claimed that there was indisputable evidence, given the results of the human evaluation at the 2017 Multimodal Translation Shared Task. I am not sure if I agree with myself any more.

Desmond Elliott is supported by an Amazon Research Award. Stella Frank, Ákos Kádár, Emiel van Miltenburg, and the Edinburgh NLP group provided valuable feedback on this paper.

References

Daniel J Benjamin, James O Berger, Magnus Johannesson, Brian A Nosek, E-J Wagenmakers, Richard Berk, Kenneth A Bollen, Björn Brembs, Lawrence Brown, Colin Camerer, et al. 2018. Redefine statistical significance. *Nature Human Behaviour*, 2(1):6.

Ozan Caglayan, Walid Aransa, Adrien Bardet, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, Marc Masana, Luis Herranz, and Joost van de Weijer. 2017. LIUM-CVC Submissions for WMT17 Multimodal Translation Task. In *Proceedings of the Second Conference on Machine Translation*, pages 432–439, Copenhagen, Denmark.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA.

Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the Second Shared Task on Multimodal Machine Translation and Multilingual Image Description. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 215–233, Copenhagen, Denmark.

Desmond Elliott, Stella Frank, and Eva Hasler. 2015. Multi-Language Image Description with Neural Sequence Models. *CoRR*, abs/1510.04709.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual English-German Image Descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74.

Stella Frank, Desmond Elliott, and Lucia Specia. 2018. Assessing multilingual multimodal image description: Studies of native speaker preferences and translator choices. *Natural Language Engineering*, 24(3):393–413.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition*.

Julian Hitschler, Shigehiko Schamoni, and Stefan Riezler. 2016. Multimodal Pivots for Image Caption Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2399–2409.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Jindřich Libovický and Jindřich Helcl. 2017. Attention Strategies for Multi-Source Sequence-to-Sequence Learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 196–202, Vancouver, Canada.

Mingbo Ma, Dapeng Li, Kai Zhao, and Liang Huang. 2017. OSU Multimodal Machine Translation System Report. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 465–469, Copenhagen, Denmark.

Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich, Aurélie Herbelot, Moin Nabi, Enver Sangineto, and Raffaella Bernardi. 2017. FOIL it! Find One mismatch between Image and Language caption. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 255–265, Vancouver, Canada.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A Shared Task on Multimodal Machine Translation and Crosslingual Image Description. In *Proceedings of the First Conference on Machine Translation*, pages 543–553, Berlin, Germany.

Loss in Translation:
Learning Bilingual Word Mapping with a Retrieval Criterion

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou and Edouard Grave
Facebook AI Research
`{ajoulin,bojanowski,tmikolov,rvj,egrave}@fb.com`

Abstract

Continuous word representations learned separately on distinct languages can be aligned so that their words become comparable in a common space. Existing works typically solve a quadratic problem to learn a orthogonal matrix aligning a bilingual lexicon, and use a retrieval criterion for inference. In this paper, we propose an unified formulation that directly optimizes a retrieval criterion in an end-to-end fashion. Our experiments on standard benchmarks show that our approach outperforms the state of the art on word translation, with the biggest improvements observed for distant language pairs such as English-Chinese.

1 Introduction

Previous work has proposed to learn a linear mapping between continuous representations of words by employing a small bilingual lexicon as supervision. The transformation generalizes well to words that are not observed during training, making it possible to extend the lexicon. Another application is to transfer predictive models between languages (Klementiev et al., 2012).

The first simple method proposed by Mikolov *et al.* (2013b) has been subsequently improved by changing the problem parametrization. One successful suggestion is to ℓ_2–normalize the word vectors and to constrain the linear mapping to be orthogonal (Xing et al., 2015). An alignment is then efficiently found using orthogonal Procrustes (Artetxe et al., 2016; Smith et al., 2017), improving the accuracy on standard benchmarks.

Yet, the resulting models suffer from the so-called “hubness problem”: some word vectors tend to be the nearest neighbors of an abnormally high number of other words. This limitation is now addressed by applying a corrective metric at inference time, such as the inverted softmax (ISF) (Smith et al., 2017) or the cross-domain similarity local scaling (CSLS) (Conneau et al., 2017). This is not fully satisfactory because the loss used for inference is not consistent with that employed for training. This observation suggests that the square loss is suboptimal and could advantageously be replaced by a loss adapted to retrieval.

In this paper, we propose a training objective inspired by the CSLS retrieval criterion. We introduce convex relaxations of the corresponding objective function, which are efficiently optimized with projected subgradient descent. This loss can advantageously include unsupervised information and therefore leverage the representations of words not occurring in the training lexicon.

Our contributions are as follows. First we introduce our approach and empirically evaluate it on standard benchmarks for word translation. We obtain state-of-the-art bilingual mappings for more than 25 language pairs. Second, we specifically show the benefit of our alternative loss function and of leveraging unsupervised information. Finally, we show that with our end-to-end formulation, a non-orthogonal mapping achieves better results. The code for our approach is a part of the fastText library[1] and the aligned vectors are available on `https://fasttext.cc/`.

2 Preliminaries on bilingual mappings

This section introduces pre-requisites and prior works to learn a mapping between two languages, using a small bilingual lexicon as supervision.

We start from two sets of continuous representations in two languages, each learned on monolingual data. Let us introduce some notation. Each word $i \in \{1, \dots, N\}$ in the source language (respectively target language) is associated with a vector $\mathbf{x}_i \in \mathbb{R}^d$ (respectively $\mathbf{y}_i \in \mathbb{R}^d$). For simplicity,

---

[1] `https://github.com/facebookresearch/fastText/tree/master/alignment/`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2979–2984
Brussels, Belgium, October 31 - November 4, 2018.

we assume that our initial lexicon, or seeds, corresponds to the first n pairs $(\mathbf{x}_i, \mathbf{y}_i)_{i\in\{1,\ldots,n\}}$. The goal is to extend the lexicon to all source words $i \in \{n+1, \ldots, N\}$ that are not seeds. Mikolov *et al.* (2013b) learn a linear mapping $\mathbf{W} \in \mathbb{R}^{d\times d}$ between the word vectors of the seed lexicon that minimizes a measure of discrepancy between mapped word vectors of the source language and word vectors of the target language:

$$\min_{\mathbf{W}\in\mathbb{R}^{d\times d}} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{W}\mathbf{x}_i, \mathbf{y}_i), \tag{1}$$

where ℓ is a loss function, typically the square loss $\ell_2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2$. This leads to a least squares problem, which is solved in closed form.

Orthogonality. The linear mapping $\mathbf{W}$ is constrained to be orthogonal, *i.e.* such that $\mathbf{W}^\top\mathbf{W} = \mathbf{I}_d$, where $\mathbf{I}_d$ is the d-dimensional identity matrix. This choice preserves distances between word vectors, and likewise word similarities. Previous works (Xing et al., 2015; Artetxe et al., 2016; Smith et al., 2017) experimentally observed that constraining the mapping in such a way improves the quality of the inferred lexicon. With the square loss and by enforcing an orthogonal mapping $\mathbf{W}$, Eq. (1) admits a closed form solution (Gower and Dijksterhuis, 2004): $\mathbf{W}^* = \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U}\mathbf{D}\mathbf{V}^\top$ is the singular value decomposition of the matrix $\mathbf{Y}^\top\mathbf{X}$.

Inference. Once a mapping $\mathbf{W}$ is learned, one can infer word correspondences for words that are not in the initial lexicon. The translation $t(i)$ of a source word i is obtained as

$$t(i) \in \operatorname*{arg\,min}_{j\in\{1,\ldots,N\}} \ \ell(\mathbf{W}\mathbf{x}_i, \mathbf{y}_j). \tag{2}$$

When the squared loss is used, this amounts to computing $\mathbf{W}\mathbf{x}_i$ and to performing a nearest neighbor search with respect to the Euclidean distance:

$$t(i) \in \operatorname*{arg\,min}_{j\in\{1,\ldots,N\}} \|\mathbf{W}\mathbf{x}_i - \mathbf{y}_j\|_2^2. \tag{3}$$

Hubness. A common observation is that nearest neighbor search for bilingual lexicon inference suffers from the “hubness problem” (Doddington et al., 1998; Dinu et al., 2014). Hubs are words that appear too frequently in the neighborhoods of other words. To mitigate this effect, a simple solution is to replace, at inference time, the square ℓ_2-norm in Eq. (3) by another criterion, such as ISF (Smith et al., 2017) or CSLS (Conneau et al., 2017).

This solution, both with ISF and CSLS criteria, is applied with a transformation $\mathbf{W}$ learned using the square loss. However, replacing the loss in Eq. (3) creates a discrepancy between the learning of the translation model and the inference.

3 Word translation as a retrieval task

In this section, we propose to directly include the CSLS criterion in the model in order to make learning and inference consistent. We also show how to incorporate unsupervised information..

The CSLS criterion is a similarity measure between the vectors $\mathbf{x}$ and $\mathbf{y}$ defined as:

$$\begin{aligned}&\mathrm{CSLS}(\mathbf{x}, \mathbf{y}) = -2\cos(\mathbf{x}, \mathbf{y}) \\ &+\frac{1}{k}\sum_{\mathbf{y}'\in\mathcal{N}_Y(\mathbf{x})} \cos(\mathbf{x}, \mathbf{y}') + \frac{1}{k}\sum_{\mathbf{x}'\in\mathcal{N}_X(\mathbf{y})} \cos(\mathbf{x}', \mathbf{y}),\end{aligned}$$

where $\mathcal{N}_Y(\mathbf{x})$ is the set of k nearest neighbors of the point $\mathbf{x}$ in the set of target word vectors $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, and cos is the cosine similarity. Note, the second term in the expression of the CSLS loss does not change the neighbors of $\mathbf{x}$. However, it gives a loss function that is symmetrical with respect to its two arguments, which is a desirable property for training.

Objective function. Let us now write the optimization problem for learning the bilingual mapping with CSLS. At this stage, we follow previous work and constrain the linear mapping $\mathbf{W}$ to belong to the set of orthogonal matrices $\mathcal{O}_d$. Here, we also assume that word vectors are ℓ_2-normalized. Under these assumptions, we have $\cos(\mathbf{W}\mathbf{x}_i, \mathbf{y}_i) = \mathbf{x}_i^\top\mathbf{W}^\top\mathbf{y}_i$. Similarly, we have $\|\mathbf{y}_j - \mathbf{W}\mathbf{x}_i\|_2^2 = 2 - 2\mathbf{x}_i^\top\mathbf{W}^\top\mathbf{y}_j$. Therefore, finding the k nearest neighbors of $\mathbf{W}\mathbf{x}_i$ among the elements of Y is equivalent to finding the k elements of Y which have the largest dot product with $\mathbf{W}\mathbf{x}_i$. We adopt this equivalent formulation because it leads to a convex formulation when relaxing the orthogonality constraint on $\mathbf{W}$. In summary, our optimization problem with the Relaxed CSLS loss (RCSLS) is written as:

$$\begin{aligned}\min_{\mathbf{W}\in\mathcal{O}_d} \frac{1}{n}\sum_{i=1}^{n} & -2\mathbf{x}_i^\top\mathbf{W}^\top\mathbf{y}_i \\ &+\frac{1}{k}\sum_{\mathbf{y}_j\in\mathcal{N}_Y(\mathbf{W}\mathbf{x}_i)} \mathbf{x}_i^\top\mathbf{W}^\top\mathbf{y}_j \\ &+\frac{1}{k}\sum_{\mathbf{W}\mathbf{x}_j\in\mathcal{N}_X(\mathbf{y}_i)} \mathbf{x}_j^\top\mathbf{W}^\top\mathbf{y}_i.\end{aligned} \tag{4}$$

Convex relaxation. Eq. (4) involves the minimization of a non-smooth cost function over the manifold of orthogonal matrices $\mathcal{O}_d$. As such, it can be solved using manifold optimization tools (Boumal et al., 2014). In this work, we consider as an alternative to the set $\mathcal{O}_d$, its convex hull $\mathcal{C}_d$, *i.e.*, the unit ball of the spectral norm. We refer to this projection as the "Spectral" model. We also consider the case where these constraints on the alignment matrix are simply removed.

Having a convex domain allows us to reason about the convexity of the cost function. We observe that the second and third terms in the CSLS loss can be rewritten as follows:

$$\sum_{\mathbf{y}_j \in \mathcal{N}_k(\mathbf{W}\mathbf{x}_i)} \mathbf{x}_i^\top \mathbf{W}^\top \mathbf{y}_j = \max_{S \in \mathcal{S}_k(n)} \sum_{j \in S} \mathbf{x}_i^\top \mathbf{W}^\top \mathbf{y}_j,$$

where $\mathcal{S}_k(n)$ denotes the set of all subsets of $\{1, \dots, n\}$ of size k. This term, seen as a function of $\mathbf{W}$, is a maximum of linear functions of $\mathbf{W}$, which is convex (Boyd and Vandenberghe, 2004). This shows that our objective function is convex with respect to the mapping $\mathbf{W}$ and piecewise linear (hence non-smooth). Note, our approach could be generalized to other loss functions by replacing the term $\mathbf{x}_i^\top \mathbf{W}^\top \mathbf{y}_j$ by any function convex in $\mathbf{W}$. We minimize this objective function over the convex set $\mathcal{C}_d$ by using the projected subgradient descent algorithm.

The projection onto the set $\mathcal{C}_d$ is solved by taking the singular value decomposition (SVD) of the matrix, and thresholding the singular values to one.

Extended Normalization. Usually, the number of word pairs in the seed lexicon n is small with respect to the size of the dictionaries N. To benefit from unlabeled data, it is common to add an iterative "refinement procedure" (Artetxe et al., 2017) when learning the translation model $\mathbf{W}$. Given a model $\mathbf{W}_t$, this procedure iterates over two steps. First it augments the training lexicon by keeping the best-inferred translation in Eq. (3). Second it learns a new mapping $\mathbf{W}_{t+1}$ by solving the problem in Eq. (1). This strategy is similar to standard semi-supervised approaches where the training set is augmented over time. In this work, we propose to use the unpaired words in the dictionaries as "negatives" in the RCSLS loss: instead of computing the k-nearest neighbors $\mathcal{N}_Y(\mathbf{W}\mathbf{x}_i)$ amongst the annotated words $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$, we do it over the whole dictionary $\{\mathbf{y}_1, \dots, \mathbf{y}_N\}$.

4 Experiments

This section reports the main results obtained with our method. We provide complementary results and an ablation study in the appendix. We refer to our method without constraints as RCSLS and as RCSLS+spectral if the spectral constraints are used.

4.1 Implementation details

We choose a learning rate in $\{1,\ 10,\ 25,\ 50\}$ and a number of epochs in $\{10,\ 20\}$ on the validation set. For the unconstrained RCSLS, a small ℓ_2 regularization can be added to prevent the norm of $\mathbf{W}$ to diverge. In practice, we do not use any regularization. For the English-Chinese pairs (en-zh), we center the word vectors. The number of nearest neighbors in the CSLS loss is 10. We use the ℓ_2-normalized fastText word vectors by Bojanowski *et al.* (2017) trained on Wikipedia.

4.2 The MUSE benchmark

Table 1 reports the comparison of RCSLS with standard supervised and unsupervised approaches on 5 language pairs (in both directions) of the MUSE benchmark (Conneau et al., 2017). Every approach uses the Wikipedia fastText vectors and supervision comes in the form of a lexicon composed of 5k words and their translations. Regardless of the relaxation, RCSLS outperforms the state of the art by, on average, 3 to 4% in accuracy. This shows the importance of using the same criterion during training and inference. Note that the refinement step ("refine") also uses CSLS to finetune the alignments but leads to a marginal gain for supervised methods.

Interestingly, RCSLS achieves a better performance without constraints (+0.8%) for all pairs. Contrary to observations made in previous works, this result suggests that preserving the distance between word vectors is not essential for word translation. Indeed, previous works used a ℓ_2 loss where, indeed, orthogonal constraints lead to an improvement of +5.3% (Procrustes versus Least Square Error). This suggests that a linear mapping $\mathbf{W}$ with no constraints works well only if it is learned with a proper criterion.

Impact of extended normalization. Table 2 reports the gain brought by including words not in the lexicon (unannotated words) to the performance of RCSLS. Extending the dictionary significantly improves the performance on all language pairs.

Method	en-es	es-en	en-fr	fr-en	en-de	de-en	en-ru	ru-en	en-zh	zh-en	avg.
Adversarial + refine	81.7	83.3	82.3	82.1	74.0	72.2	44.0	59.1	32.5	31.4	64.3
ICP + refine	82.2	83.8	82.5	82.5	74.8	73.1	46.3	61.6	-	-	-
Wass. Proc. + refine	82.8	84.1	82.6	82.9	75.4	73.3	43.7	59.1	-	-	-
Least Square Error	78.9	80.7	79.3	80.7	71.5	70.1	47.2	60.2	42.3	4.0	61.5
Procrustes	81.4	82.9	81.1	82.4	73.5	72.4	51.7	63.7	42.7	36.7	66.8
Procrustes + refine	82.4	83.9	82.3	83.2	75.3	73.2	50.1	63.5	40.3	35.5	66.9
RCSLS + spectral	83.5	85.7	82.3	**84.1**	78.2	75.8	56.1	66.5	44.9	45.7	70.2
RCSLS	**84.1**	**86.3**	**83.3**	**84.1**	**79.1**	**76.3**	**57.9**	**67.2**	**45.9**	**46.4**	**71.0**

Table 1: Comparison between RCSLS, Least Square Error, Procrustes and unsupervised approaches in the setting of Conneau et al. (2017). All the methods use the CSLS criterion for retrieval. "Refine" is the refinement step of Conneau et al. (2017). Adversarial, ICP and Wassertsein Proc. are unsupervised (Conneau et al., 2017; Hoshen and Wolf, 2018; Grave et al., 2018).

	en-es	en-fr	en-de	en-ru	avg.
Train	80.7	82.3	74.8	51.9	72.4
Ext.	**84.1**	**83.3**	**79.1**	**57.9**	**76.1**

Table 2: Accuracy with and without an extended normalization for RCSLS. "Ext." uses the full 200k vocabulary and "Train" only uses the pairs from the training lexicon.

	en-it	it-en
Adversarial + refine + CSLS	45.1	38.3
Mikolov et al. (2013b)	33.8	24.9
Dinu et al. (2014)	38.5	24.6
Artetxe et al. (2016)	39.7	33.8
Smith et al. (2017)	43.1	38.0
Procrustes + CSLS	44.9	**38.5**
RCSLS	**45.5**	38.0

Table 3: Accuracy on English and Italian with the setting of Dinu et al. (2014). "Adversarial" is an unsupervised technique. The adversarial and Procrustes results are from Conneau et al. (2017). We use a CSLS criterion for retrieval.

4.3 The WaCky dataset

Dinu et al. (2014) introduce a setting where word vectors are learned on the WaCky datasets (Baroni et al., 2009) and aligned with a noisy bilingual lexicon. We select the number of epochs within $\{1, 2, 5, 10\}$ on a validation set. Table 3 shows that RCSLS is on par with the state of the art. RCSLS is thus robust to relatively poor word vectors and noisy lexicons.

	Original			Aligned		
	Sem.	Synt.	Tot.	Sem.	Synt.	Tot.
Cs	26.4	76.7	63.7	**27.3**	**77.7**	**64.6**
De	**62.2**	56.9	**59.5**	61.4	**57.1**	59.3
Es	54.5	59.4	56.8	**55.1**	**61.1**	**57.9**
Fr	**76.0**	54.7	**68.5**	75.2	**55.1**	68.1
It	51.8	62.0	56.9	**52.7**	**63.8**	**58.2**
Pl	49.7	62.4	53.4	**50.9**	**63.2**	**54.5**
Zh	42.6	-	42.6	**47.2**	-	**47.2**
Avg.	51.9	62.0	57.3	**52.8**	**58.5**	**58.5**

Table 4: Performance on word analogies for different languages. We compare the original embeddings to their mapping to English. The mappings are learned using the full MUSE bilingual lexicons. We use the fastText vectors of Bojanowski et al. (2017).

4.4 Comparison with existing aligned vectors

Recently, word vectors based on fastText have been aligned and released by Smith *et al.* (2017, BabylonPartners, BP) and Conneau *et al.* (2017, MUSE). Both use a variation of Procrustes to align word vectors in the same space.

We compare these methods to RCSLS and report results in Table 5. RCSLS improves the performance by $+3.5\%$ over MUSE vectors when trained with the same lexicon (Original). Training RSCSL on the full training lexicon (Full) brings an additional improvement of $+2.9\%$ on average with a CSLS criterion, and $+6.1\%$ with a NN criterion. For reference, the performance of Procrustes only improves by $+1.4\%$ with CSLS and even degrades with a NN criterion. RCSLS benefits more from additional supervision than Procrustes. Finally,

	BP*	MUSE	Proc.	RCSLS	
		Orig.	Full	Orig.	Full
Bg	55.7	57.5	58.1	63.9	**65.2**
Ca	66.5	70.9	70.5	73.8	**75.0**
Cs	63.9	64.5	66.3	68.2	**71.1**
Da	66.8	67.4	68.3	71.1	**72.9**
De	68.9	72.7	73.5	76.9	**77.6**
El	54.9	58.5	60.1	62.7	**64.5**
Es	82.1	83.5	84.5	86.4	**87.1**
Et	41.5	45.7	47.3	49.5	**53.7**
Fi	56.7	59.5	61.9	65.8	**69.9**
Fr	81.7	82.4	82.5	**84.7**	**84.7**
He	51.5	54.1	55.4	57.8	**60.0**
Hr	48.9	52.2	53.4	55.6	**60.2**
Hu	61.9	64.9	66.1	69.3	**73.1**
Id	62.8	67.9	67.9	69.7	**72.9**
It	75.3	77.9	78.5	81.5	**82.8**
Mk	53.9	54.6	55.4	59.9	**60.4**
Nl	72.0	75.3	76.1	79.7	**80.5**
No	65.3	67.4	68.3	71.2	**73.3**
Pl	63.3	66.9	68.1	70.5	**73.5**
Pt	77.7	80.3	80.4	82.9	**84.6**
Ro	66.3	68.1	67.6	**74.0**	73.9
Ru	61.3	63.7	64.3	67.1	**70.3**
Sk	55.1	55.3	57.9	59.0	**61.7**
Sl	51.1	50.4	52.5	54.2	**58.2**
Sv	55.9	60.0	64.0	63.7	**69.5**
Tr	57.4	59.2	61.4	61.9	**65.8**
Uk	48.7	49.3	51.3	51.5	**55.5**
Vi	35.0	55.8	63.0	55.8	**66.9**
CSLS	60.8	63.8	65.2	67.4	**70.2**
NN	54.6	57.4	57.5	62.4	**68.5**

Table 5: Comparison with publicly available aligned vectors over 28 languages. All use supervision. Alignements are learned either on the "Original" or "Full" MUSE training. We report the detailed performance with a CSLS criterion and the average for both NN and CSLS criteria.

*BP uses a different training set of comparable size.

the gap between RCSLS and the other methods is higher with a NN criterion, suggesting that RCSLS imports some of the properties of CSLS to the dot product between aligned vectors.

4.5 Impact on word vectors

Non-orthogonal mapping of word vectors changes their dot products. We evaluate the impact of this mapping on word analogy tasks (Mikolov et al., 2013a). In Table 4, we report the accuracy on analogies for raw word vectors and our vectors mapped to English with an alignement trained on the full MUSE training set. Regardless of the source language, the mapping does not negatively impact the word vectors. Similarly, our alignement has also little impact on word similarity, as shown in Table 6.

		Original	Aligned
De	Gur350	72	**74**
	Ws350	68	**70**
	Zg222	**46**	44
Es	Ws353	59	**61**
It	Ws350	**64**	**64**
Pt	Ws353	**60**	58
Ro	Ws353	**58**	55
	Hj	**67**	65
	Ws350	**59**	58
Zh	Sim	35	**44**
Avg.		58.8	**59.2**

Table 6: Performance on word similarities for different languages. We compare the original embeddings to their mapping to English. The mappings are learned with the full MUSE bilingual lexicons over the fastText vectors of Bojanowski et al. (2017).

We confirm this observation by running the reverse mapping, i.e., by mapping the English word vectors of Mikolov et al. (2018) to Spanish. It leads to an improvement of 1% both for vectors trained on Common Crawl (85% to 86%) and Wikipedia + News (87% to 88%).

5 Conclusion

This paper shows that minimizing a convex relaxation of the CSLS loss significantly improves the quality of bilingual word vector alignment. We use a reformulation of CSLS that generalizes to convex functions beyond dot products and provides a single end-to-end training that is consistent with the inference stage. Finally, we show that removing the orthogonality constraint does not degrade the quality of the aligned vectors.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 451–462.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146. `https://fasttext.cc/docs/en/pretrained-vectors.html`.

Nicolas Boumal, Bamdev Mishra, P.-A. Absil, and Rodolphe Sepulchre. 2014. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*. `http://github.com/facebookresearch/MUSE`.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*.

George Doddington, Walter Liggett, Alvin Martin, Mark Przybocki, and Douglas Reynolds. 1998. Sheep, goats, lambs and wolves: A statistical analysis of speaker performance in the NIST 1998 speaker recognition evaluation. Technical report.

John C Gower and Garmt B Dijksterhuis. 2004. *Procrustes problems*, volume 30. Oxford University Press on Demand.

Edouard Grave, Armand Joulin, and Quentin Berthet. 2018. Unsupervised alignment of embeddings with wasserstein procrustes. *arXiv preprint arXiv:1805.11222*.

Yedid Hoshen and Lior Wolf. 2018. An iterative closest point method for unsupervised word translation. *arXiv preprint arXiv:1801.06126*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. *Proceedings of COLING 2012*, pages 1459–1474.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*. `https://github.com/Babylonpartners/fastText_multilingual`.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Learning When to Concentrate or Divert Attention: Self-Adaptive Attention Temperature for Neural Machine Translation

Junyang Lin[1,2], **Xu Sun**[2], **Xuancheng Ren**[2], **Muyu Li**[2], **Qi Su**[1]
School of Foreign Languages, Peking University[1]
MOE Key Lab of Computational Linguistics, School of EECS, Peking University[2]
`{linjunyang, xusun, renxc, limuyu0110, sukia}@pku.edu.cn`

Abstract

Most of the Neural Machine Translation (NMT) models are based on the sequence-to-sequence (Seq2Seq) model with an encoder-decoder framework equipped with the attention mechanism. However, the conventional attention mechanism treats the decoding at each time step equally with the same matrix, which is problematic since the softness of the attention for different types of words (e.g. content words and function words) should differ. Therefore, we propose a new model with a mechanism called Self-Adaptive Control of Temperature (SACT) to control the softness of attention by means of an attention temperature. Experimental results on the Chinese-English translation and English-Vietnamese translation demonstrate that our model outperforms the baseline models, and the analysis and the case study show that our model can attend to the most relevant elements in the source-side contexts and generate the translation of high quality.

1 Introduction

In recent years, Neural Machine Translation (NMT) has become the mainstream method of machine translation as it, in a great number of cases, outperforms most models based on Statistical Machine Translation (SMT), let alone the linguistics-based methods. One of the most popular baseline models is the sequence-to-sequence (Seq2Seq) model (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014) with attention mechanism (Bahdanau et al., 2014; Luong et al., 2015). However, the conventional attention mechanism is problematic in real practice. The same weight matrix for attention is applied to all decoder outputs at all time steps, which, however, can cause inaccuracy. Take a typical example from the perspective of linguistics. Words can be categorized into two types, function word, and content word. Function words and content words execute different functions in the construction of a sentence, which is relevant to syntactic structure and semantic meaning respectively. Our motivation is that the attention mechanism for different types of words, especially function word and content word, should be different. When decoding a content word, the attention scores on the source-side contexts should be harder so that the decoding can be more focused on the concrete word that is semantic referent in the source text. But when decoding a function word, the attention scores should be softer so that the decoding can pay attention to its syntactic constituents in the source text that may be several words instead of one word.

To tackle the problem mentioned above, we propose a mechanism called Self-Adaptive Control of Temperature (SACT) to control the softness of attention for the RNN-based Seq2Seq model[1]. We set a temperature parameter, which can be learned by the model based on the attention in the previous decoding time steps as well as the output of the decoder at the current time step. With the temperature parameter, the model is able to automatically tune the degree of softness of the distribution of the attention scores. To be specific, the model can learn a soft distribution of attention which is more uniform for generating function word and a hard distribution which is sparser for generating content words.

Our contributions in this study are in the following: (1). We propose a new model for NMT, which contains a mechanism called Self-Adaptive Control of Temperature (SACT) to control the softness of the attention score distribution. (2). Experimental results demonstrate that our model outperforms the attention-based Seq2Seq model in both Chinese-English and English-Vietnamese translation, with a 2.94 BLEU point and 2.19 BLEU

---

[1] The code is available at `https://github.com/lancopku/SACT`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2985–2990
Brussels, Belgium, October 31 - November 4, 2018.

score advantage respectively[2]. (3). The analysis shows that our model is more capable of translating long texts, compared with the baseline models.

2 Our Model

As is mentioned above, our model is substantially a Seq2Seq framework improved by the SACT mechanism. In this section, we first briefly describe the Seq2Seq model, then introduce the SACT mechanism in detail.

2.1 Seq2Seq Model

We implement the encoder with bidirectional Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), where the encoder outputs from two directions at each time step are concatenated, and we implement the decoder with unidirectional LSTM. We train our model with the Cross-Entropy Loss, which is equivalent to the maximum likelihood estimation. In the following, we introduce the details of our proposed attention mechanism.

2.2 Self-Adaptive Control of Temperature

In our assumption, due to the various functions of words, decoding at each time step should not use the identical attention mechanism to extract the required information from the source-side contexts. Therefore, we propose our Self-Adaptive Control of Temperature (SACT) to improve the conventional attention mechanism, so that the model can learn to control the scale of the softness of attention for the decoding of different words. In the following, we present the details of our design of the mechanism.

We set a temperature parameter τ to control the softness of the attention at each time step. The temperature parameter τ can be learned by the model itself. In our assumption, the temperature parameter is learned based on the information of the decoding at the current time step as well as the attention in the previous time steps, referring to the information about what has been translated and what is going to be translated. Specifically, it is defined as below:

$$\tau_t = \lambda^{\beta_t} \tag{1}$$

$$\beta_t = \tanh(W_c\tilde{c}_{t-1} + U_s s_t) \tag{2}$$

where s_t is the output of the LSTM decoder as mentioned above, $\tilde{c}_{t-1}$ is the context vector generated by our attention mechanism at the last time step (initialized with the initial state of the decoder for the decoding at the first time step), and λ is a hyper-parameter, which decides the upper bound and the lower bound of the scale for the softness of attention. To be specific, λ should be a number larger than 1[3]. The range of the output value of $\tanh$ function is $(-1, 1)$, so the range of the τ is $(\frac{1}{\lambda}, \lambda)$. Furthermore, the temperature parameter is applied to the conventional attention mechanism.

Different from the conventional attention mechanism, the temperature parameter is applied to the computation of attention score α so that the scale of the softness of attention can be changed. We define the new attention score and context vector as $\tilde{\alpha}$ and $\tilde{c}$, which are computed as:

$$\tilde{c}_t = \sum_{i=1}^{n} \tilde{\alpha}_{t,i} h_i \tag{3}$$

$$\tilde{\alpha}_{t,i} = \frac{exp(\tau_t^{-1} e_{t,i})}{\sum_{j=1}^{n} exp(\tau_t^{-1} e_{t,j})} \tag{4}$$

From the definition above, it can be inferred that when the temperature increases, the distribution of the attention score α is smoother, meaning that softer attention is required, and when the temperature is low, the distribution is sparser, meaning that harder attention is required. Therefore, the model can tune the softness of the attention distribution self-adaptively based on the current output for the decoder and the history of attention, and learns when to attend to only corresponding words and when to attend to more relevant words for further syntactic and semantic information.

3 Experiment

In the following, we introduce the experimental details, including the datasets and the experiment setting.

---

[2] What should be mentioned is that though the "Transformer" model is recently regarded as the best, the model architecture is not the focus of our study. Furthermore, our proposed mechanism can also be applied to the aforementioned model, which will be a part of our future study.

[3] In our experiments, we use λ of different values, ranging from 2 to 10. The performance differences of models with different λ values are not significant, and we report the results of the model with 4 as the value of λ as it achieves the best performance.

Model	MT-03	MT-04	MT-05	MT-06	Ave.
Moses	32.43	34.14	31.47	30.81	32.21
RNNSearch	33.08	35.32	31.42	31.61	32.86
Coverage	34.49	38.34	34.91	34.25	35.49
MemDec	36.16	39.81	35.91	**35.98**	36.97
Seq2Seq	35.32	37.25	33.52	33.54	34.91
+SACT	**38.16**	**40.48**	**36.81**	35.95	**37.85**

Table 1: **Results of the models on the Chinese-English translation**

3.1 Datasets

Chinese-English Translation We train our model on 1.25M sentence pairs[4] with 27.9M Chinese words and 34.5M English words, and we validate our model on the dataset for the NIST 2002 translation task and test our model on the datasets for the NIST 2003, 2004, 2005, 2006 translation tasks. We use the most frequent 30K words for the Chinese vocabulary and the English vocabulary respectively, covering about 97.4% and 99.7% of the corpora. The evaluation metric is case-insensitive BLEU score computed by `mteval-13a.perl` (Papineni et al., 2002).

English-Vietnamese Translation The training data is from the translated TED talks, containing 133K training sentence pairs provided by the IWSLT 2015 Evaluation Campaign (Cettolo et al., 2015). The validation set is the TED tst2012 with 1553 sentences and the test set is the TED tst2013 with 1268 sentences. The English vocabulary is 17.7K words and the Vietnamese vocabulary is 7K words. The evaluation metric is also BLEU as mentioned above[5].

3.2 Setting

Our model is implemented with PyTorch on an NVIDIA 1080Ti GPU. Both the size of word embedding and the size of the hidden layers in the encoder and decoder are 512. Gradient clipping for the gradients is applied with the largest gradient norm 10 in our experiments. Dropout is used with the dropout rate set to 0.3 for the Chinese-English translation and 0.4 for the English-Vietnamese translation, in accordance with the evaluation on the development set. Batch size is set to 64. We use Adam optimizer (Kingma and Ba, 2014) to train the model[6].

3.3 Baselines

In the following, we introduce our baseline models for the Chinese-English translation and the English-Vietnamese translation respectively.

For the Chinese-English translation, we compare our model with the most recent NMT systems, illustrated in the following. **Moses** is an open source phrase-based translation system with default configurations and a 4-gram language model trained on the training data for the target language; **RNNSearch** is an attention-based Seq2Seq with fine-tuned hyperparameters; **Coverage** is the attention-based Seq2Seq model with a coverage model (Tu et al., 2016); **MemDec** is the attention-based Seq2Seq model with the external memory (Wang et al., 2016).

For the English-Vietnamese translation, the models to be compared are presented below. **RNNSearch** The attention-based Seq2Seq model as mentioned above, and we present the results of (Luong and Manning, 2015); **NPMT** is the Neural Phrase-based Machine Translation model by Huang et al. (2017).

4 Results and Analysis

In the following, we present the experimental results as well as our analysis of temperature and case study.

4.1 Results

We present the performance of the baseline models and our model on the Chinese-English translation in Table 1. As to the recent models on the same task with the same training data, we extract their results from their original articles. Compared with the baseline models, our model with the SACT for the softness of attention achieves

---

[4] The dataset is extracted from LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

[5] For comparison with the existing system, we use `multi-bleu.perl` instead.

[6] $\alpha = 0.0003$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$

Model	BLEU
RNNSearch	26.10
NPMT	27.69
Seq2Seq	26.93
+SACT	**29.12**

Table 2: **Results of the models on the English-Vietnamese translation**

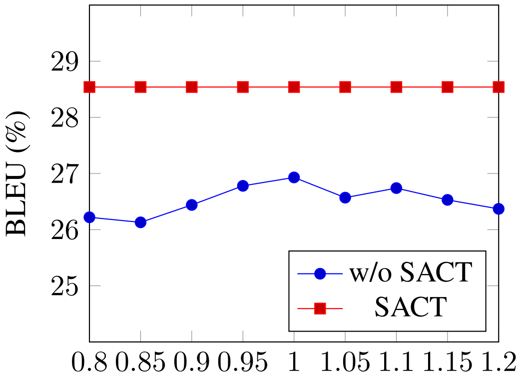

Figure 1: BLEU scores of the Seq2Seq models with fixed values for the temperature parameter. Models are tested on the test set of the English-Vietnamese translation.

better performance, with the advantages of BLEU score 2.94 over the conventional attention-based Seq2Seq model. The SACT effectively learns the temperature to control the softness of attention so that the model can utilize the information from the source-side contexts more efficiently.

We present the results of the models on the English-Vietnamese translation in Table 2. Compared with the attention-based Seq2Seq model, our model with the SACT can outperform it with a clear advantage of 2.17 BLEU score. We also display the most recent model NPMT (Huang et al., 2017) trained and tested on the dataset. Compared with NPMT, our model has an advantage of BLEU score of 1.43. It can be indicated that for low-resource translation, the information from the deconvolution-based decoder is important, which brings significant improvement to the conventional attention-based Seq2Seq model.

4.2 Analysis

In order to verify whether the automatically changing temperature can positively impact the performance of the model, we implement a series of models with fixed values, ranging from 0.8 to 1.2, for the temperature parameter. From the results shown in Figure1, it can be found that the automatically changing temperature can encourage the model to outperform those with fixed temperature parameter.

Furthermore, as our model generates a temperature parameter at each time step of decoding, we present the heatmaps of two translations from the testing on the NIST 2003 for the Chinese-English translation on Figure 2. From the heatmaps, it can be found that the model can adapt the temperature parameter to the generation at the current time step. In Figure 2(a), when translating words such as "to" and "from", which are syntactic-relevant prepositions and both lack direct corresponding words in the source text or pronoun such as "they", whose corresponding word "tamen" in the source may be a part of the possessive case or the objective case, the temperature parameter increases to soften the attention distribution so that the model can attend to more relevant elements for accurate extraction of the information from the source-side contexts. On the contrary, when translating content words or phrases such as "pay attention" and "nuclear", where there are direct corresponding words "zhuyi" and "hezi" in the source text, the temperature decreases to harden the attention distribution so that the model can focus on the corresponding information in the source text for accurate translation. In Figure 2(b), the temperature parameters for the punctuations are high as they are highly connected to the syntactic structure and those for the content words with concrete correspondences such as location "paris", name of organization "xinhua", name of person "wang" and nationality "french".

4.3 Case Study

We present two examples of the translation of our model in comparison with the translation of the conventional attention-based Seq2Seq model and the golden translation. In Table 3(a), it can be found that the translation of the conventional Seq2Seq model does not give enough credit to the word "chengzhang" (meaning "growth"), while our model can not only concentrate on the word but also recognize the word as a noun ("chengzhang" in Chinese can be both noun and verb). Even compared with the golden translation, the translation of our model seems better, which is a grammatical and coherent sentence. In Table 3(b), although the Seq2Seq model can generate the

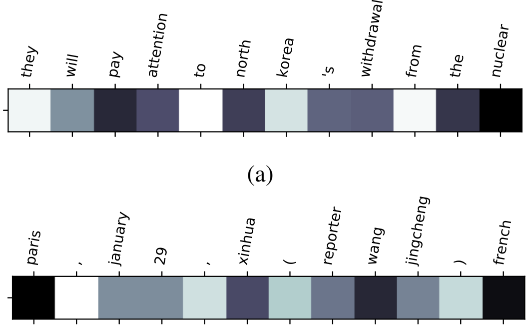

(a)

(b)

Figure 2: **Examples of the heatmaps of temperature parameter** The dark color refers to low temperature, while the light color refers to high temperature.

Source:	中国 大陆 手机 用户 成长 将 减缓
Gold:	growth of mobile phone users in mainland china to slow down
Seq2Seq:	mainland cell phone users slow down
SACT:	the growth of cell phone users in chinese mainland will slow down

(a)

Source:	自 去年 12 以来, 受 委内瑞拉 国内 大罢工 和 伊拉克 战争 的影响, 国际 市场 原油 价格 持续 上涨 。
Gold:	since december last year , the price of crude oil on the international market has kept rising due to the general strike in venezuela and the threat of war in iraq .
Seq2Seq:	since december last year , the international market has continued to rise in the international market and the threat of the iraqi war has continued to rise .
SACT:	since december last year , the international market of crude oil has continued to rise because of the strike in venezuela and the war in iraq .

(b)

Table 3: Two examples of the translation on the NIST 2003 Chinese-English translation task. The difference between Seq2Seq and SACT is shown in color.

translation about the increase in the crude oil, it wrongly connects the increase with the threat of war in Iraq. In contrast, as our model has more capability of analyzing the syntactic structure by softening the attention distribution in the generation of syntax-relevant words, it extracts the causal relationship in the source text and generates the correct translation.

5 Related Work

Most systems for Neural Machine Translation are based on the sequence-to-sequence model (Seq2Seq) (Sutskever et al., 2014), which is an encoder-decoder framework (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014). To improve NMT, a significant mechanism for the Seq2Seq model is the attention mechanism (Bahdanau et al., 2014). Two types of attention are the most common, which are proposed by Bahdanau et al. (2014) and Luong et al. (2015) respectively.

Though the attention mechanism is powerful for the requirements of alignment in NMT, some prominent problems still exist. To tackle the impact of the attention historyTu et al. (2016); Mi et al. (2016); Meng et al. (2016); Wang et al. (2016); Lin et al. (2018a) take the attention history into consideration. An important breakthrough in NMT is that Vaswani et al. (2017) applied the fully-attention-based model to NMT and achieved the state-of-the-art performance. To further evaluate the effect of our attention temperature mechanism, we will implement it to the “Transformer” model in the future. Besides, the studies on the attention mechanism have also contributed to some other tasks (Lin et al., 2018b; Liu et al., 2018)

Beyond the attention mechanism, there are also important methods for the Seq2Seq that contribute to the improvement of NMT. Ma et al. (2018) incorporates the information about the bag-of-words of the target for adapting to multiple translations, and Lin et al. (2018c) takes the target context into consideration.

6 Conclusion and Future Work

In this paper, we propose a novel mechanism for the control over the scope of attention so that the softness of the attention distribution can be changed adaptively. Experimental results demonstrate that the model outperforms the baseline models, and the analysis shows that our temperature parameter can change automatically when decoding diverse words. In the future, we hope to find out more patterns and generalized rules to explain the model’s learning of the temperature.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028) and the National Thousand Young Talents Program. Qi Su is the corresponding author of this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The iwslt 2015 evaluation campaign. *Proc. of IWSLT, Da Nang, Vietnam*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP 2014*, pages 1724–1734.

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

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Neural phrase-based machine translation. *CoRR*, abs/1706.05565.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP 2013*, pages 1700–1709.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Junyang Lin, Shuming Ma, Qi Su, and Xu Sun. 2018a. Decoding-history-based adaptive control of attention for neural machine translation. *CoRR*, abs/1802.01812.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018b. Global encoding for abstractive summarization. In *ACL 2018*, pages 163–169.

Junyang Lin, Xu Sun, Xuancheng Ren, Shuming Ma, Jinsong Su, and Qi Su. 2018c. Deconvolution-based global decoding for neural machine translation. In *COLING 2018*, pages 3260–3271.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *AAAI 2018*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP 2015*, pages 1412–1421.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *ACL 2018*, pages 332–338.

Fandong Meng, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Interactive attention for neural machine translation. In *COLING 2016*, pages 2174–2185.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *EMNLP 2016*, pages 955–960.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL, 2002*, pages 311–318.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS, 2014*, pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL 2016*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS 2017*, pages 6000–6010.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. In *EMNLP 2016*, pages 278–286.

Accelerating Asynchronous Stochastic Gradient Descent for Neural Machine Translation

Nikolay Bogoychev[★] **Marcin Junczys-Dowmunt**[✈] **Kenneth Heafield**[★] **Alham Fikri Aji**[★]
★ School of Informatics, University of Edinburgh, Edinburgh, United Kingdom
✈ Microsoft, 1 Microsoft Way, Redmond, WA 98121, USA
`{n.bogoych, a.fikri, kheafiel}@ed.ac.uk, marcinjd@microsoft.com`

Abstract

In order to extract the best possible performance from asynchronous stochastic gradient descent one must increase the mini-batch size and scale the learning rate accordingly. In order to achieve further speedup we introduce a technique that delays gradient updates effectively increasing the mini-batch size. Unfortunately with the increase of mini-batch size we worsen the stale gradient problem in asynchronous stochastic gradient descent (SGD) which makes the model convergence poor. We introduce local optimizers which mitigate the stale gradient problem and together with fine tuning our momentum we are able to train a shallow machine translation system 27% faster than an optimized baseline with negligible penalty in BLEU.

1 Introduction

With training times measured in days, parallelizing stochastic gradient descent (SGD) is valuable for making experimental progress and scaling data sizes. Synchronous SGD sums gradients computed by multiple GPUs into one update, equivalent to a larger batch size. But GPUs sit idle unless workloads are balanced, which is difficult in machine translation and other natural language tasks because sentences have different lengths. Asynchronous SGD avoids waiting, which is faster in terms of words processed per second. However asynchronous SGD suffers from stale gradients (Abadi et al., 2016) that degrade convergence, resulting in an almost no improvement in time to convergence (Hadjis et al., 2016). This paper makes asynchronous SGD even faster and deploys a series of convergence optimizations.

In order to achieve fastest training (and inspired by Goyal et al. (2017) we increase the mini-batch size, making the matrix operations more efficient and reducing the frequency of gradient communication for the optimizer step. Unlike their task (image classification), text training consumes a lot of GPU Memory (Table 1) for word embedding activations making it impossible to fit mini-batches of similar magnitude as Goyal et al. (2017).

Our main contributions are as follows:

1. We introduce a delayed gradient updates which allow us to work with much larger mini-batches which would otherwise not be possible due to limited GPU memory.
2. We introduce local optimizers which run on each worker to mitigate the extra staleness and convergence issues (Dekel et al., 2010; Keskar et al., 2017) caused by large mini-batches.
3. We highlight the importance of tuning the optimizer momentum and show how it can be used as a cooldown strategy.

VRAM	τ	**Words**	**WPS**
3 GB	1	3080	19.5k
7 GB	1	7310	36.6k
10 GB	1	10448	40.2k
20* GB	2	20897	44.2k
30* GB	2	31345	46.0k
40* GB	4	41794	47.6k

Table 1: Relationship between the GPU Memory (VRAM) budget for batches (* means emulated by summing τ smaller batches), number of source words processed in each batch and words-per-second (WPS) measured on a shallow model.

2 Experiments

This section introduces each optimization along with an intrinsic experiment on the WMT 2016 Romanian→English task (Bojar et al., 2016).

The translation system is equivalent to Sennrich et al. (2016), which was the first place constrained system (and tied for first overall in the WMT16

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2991–2996
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

shared task.). The model is a shallow bidirectional GRU (Bahdanau et al., 2014) encoder-decoder trained on 2.6 million parallel sentences. Due to variable-length sentences, machine translation systems commonly fix a memory budget then pack as many sentences as possible into a dynamically-sized batch. The memory allowance for mini-batches in our system is 3 GB (for an average batch size of 2633 words). Adam (Kingma and Ba, 2015) is used to perform asynchronous SGD with learning rate of 0.0001. This is our baseline system. We also compare with a synchronous baseline which uses modified Adam parameters, warmup of 16000 mini-batches and inverse square root cooldown following Vaswani et al. (2017). We used 4 Tesla P 100 GPUs in a single node with the Marian NMT framework for training (Junczys-Dowmunt et al., 2018). Since we apply optimizations over asynchronous SGD we performed a learning rate and mini-batch-size parameter sweep over the baseline system and settled on a learning rate of 0.00045 and 10 GB memory allowance for mini-batches (average batch size of 10449 words). This is the fastest system we could train without sacrificing performance before adding our improvements. In our experiments on Table 2 we refer to this system as ”Optimized asynchronous”. All systems were trained until 5 consecutive stalls in the cross-entropy metric of the validation set. Note that some systems require more epochs to reach this criteria which indicates poor model convergence.

2.1 Larger Batches and delayed updates

This experiment aims to increase speed, in words-per-second (WPS), by increasing the batch size. Larger batches have two well-known impacts on speed: making more use of GPU parallelism and communicating less often.

After raising the batch size to the maximum that fits on the GPU,[1] we emulate even larger batches by processing multiple mini-batches and summing their gradients locally without sending them to the optimizer. This still increases speed because communication is reduced (Table 1). We introduce parameter τ, which is the number of iterations a GPU performs locally before communicating externally as if it had run one large batch. The Words-per-second (WPS) column on Table 1 shows the effect on corpora processing speed when applying delayed gradients updates for different values of τ. While we reduce the overall training time if we just apply delayed gradient updates we worsen the overall convergence (Table 2).

When increasing the mini-batch size τ times without touching the learning rate, we effectively do τ times less updates per epoch. On the surface, it might seem that these less frequent updates are counterbalanced by the fact that each update is accumulated over a larger batch. But practical optimization heuristics like gradient clipping mean that in effect we end up updating the model less often, resulting in slower convergence. Goyal et al. (2017) recommend scaling the learning rate linearly with the mini-batch size in order to maintain convergence speed.

2.2 Warmup

Goyal et al. (2017) point out that just increasing the learning rate performs poorly for very large batch sizes, because when the model is initialized at a random point, the training error is large. Large error and large learning rate result in bad ”jerky” updates to the model and it can’t recover from those. Goyal et al. (2017) suggest that initially model updates should be small so that the model will not be pushed in a suboptimal state. Afterwards we no longer need to be so careful with our updates.

2.2.1 Lowering initial learning rate

Goyal et al. (2017) lower the initial learning rate and gradually increase it over a number of mini-batches until it reaches a predefined maximum. This technique is also adopted in the work of Vaswani et al. (2017). This is the canonical way to perform warmup for neural network training.

2.2.2 Local optimizers

We propose an alternative warm up strategy and compare it with the canonical method. Since we emulate large batches by running multiple smaller batches, it makes sense to consider whether to optimize locally between each batch by adapting the concept of local per-worker optimizers from Zhang et al. (2014). In asynchronous SGD setting each GPU has a full copy of the model as well as the master copy of $1/N$th of the parameters in its capacity as parameter server. We use the local optimizers to update the local model shard in between delayed gradient updates, which helps mitigate staleness.

---

[1] The Tesla P100 has 16 GB of GPU memory and we opt to use 10 GBs of mini-batches and the rest is used to store model parameters, shards, optimizers and additional system specific elements such as the cache vectors for gradient dropping (Aji and Heafield, 2017).

Unlike prior work, we also update the shard of the global model that happens to be on the same GPU. Local updates are almost free because we avoid remote device communication.

Updating the parameter shard of the global model bears some resemblance to the Hogwild method (Recht et al., 2011) as we don't synchronize the updates to the shard, however, global updates are still synchronised. As before, once every τ iterations we run a global optimizer that updates the sharded parameter set and then distributes the updated model across all devices. Any local model divergences are lost at this point. We found that this strategy improves model convergence in the early epochs but tends to be harmful later on. We hypothesize that initially partial model updates reduce staleness, but when the model starts to converge, local optimizers introduce extra noise in the training, which is harmful. We use local optimizers purely as a warmup strategy, turning them off after the initial phase of the training. Empirically, we found that we can get the best convergence by using them for the first 4000 mini-batches that each device sees. On Table 2 we compare and contrast the two warmup strategies. By itself learning-rate warmup offers slower convergence but to a better point compared to local optimizers. The reader may notice that if we apply delayed gradient updates, the effective batch size that the global optimizer deals with is τ times larger than the mini-batch size on which the local optimizers runs. Therefore we use τ times lower learning rate for the local optimizers compared to the global optimizers.

2.3 Momentum cooldown and tuning

Goyal et al. (2017) and Vaswani et al. (2017) both employ cooldown strategies that lower the learning rate towards the end of training. Inspired by the work of Hadjis et al. (2016) however we decided to pursue a different cooldown strategy by modifying the momentum inside Adam's parameters.

Momentum tuning is not a well explored area in deep learning. Most researchers simply use the default values for momentum for a chosen optimizer (Hadjis et al., 2016) (in the case of NMT, this is usually Adam). Hadjis et al. (2016) argue that this is an oversight especially when it comes to asynchronous SGD, because the asynchronisity adds extra implicit momentum to the training which is not accounted for. Because of this, asynchronous SGD has been deemed ineffective, as without momentum tuning, the observed increase in training speed is negated by the lower convergence rate, resulting in near-zero net gain (Abadi et al., 2016). However, Hadjis et al. (2016) show that after performing a grid search over momentum values, it is possible to achieve convergence rates typical for synchronous SGD even when working with many asynchronous workers. The downside of momentum tuning is that we can't offer rule-of-thumb values, as they are individually dependent on the optimizer used, the neural model, the number of workers and the batch size. In our experiments, we lowered the overall momentum and in addition performed momentum cooldown where we reduced the momentum of our optimizer (Adam) after the first few thousand batches.

2.4 Results

Table 2 shows the effect of modifying momentum values. When using just delayed gradient updates, training is noticeably faster, but there are significant regressions in BLEU and CE (system 2). In order to mitigate those, when using delayed gradient updates, we tune the momentum and apply momentum cooldown on top of either of our warmup strategies. By doing this we not only further reduce training time, but also recover the loss of accuracy. Compared to the optimized baseline system (1), our best system (4) reduces the training time by 27%. Progression of the training can be seen on figures 1 and 2. Our system starts poorly compared to the baselines in terms of epoch-for-epoch convergence, but catches up in the later epochs. Due to faster training speed however, the desired BLEU score is achieved faster (Figure 2).

Local optimizers as a warmup strategy show faster convergence compared to learning rate warmup at almost no penalty to BLEU or cross-entropy (System 4 vs system 6). Against the system used in WMT 16 (Sennrich et al., 2016), we achieve nearly 4 times faster training time with no discernible penalty in BLEU or CE. In contrast, the other communication reducing method tested, the work of Aji and Heafield (2017), is slower than our work and achieves worse BLEU and CE.

2.5 Using even larger mini-batches

We can achieve even greater processing speed by further increasing τ but we were unable to maintain the same convergence with the Romanian-English shallow model. We found that larger τ values are useful when dealing with the larger deep RNN mod-

System	Time (hours)	Epochs	BLEU	CE	WPS
synchronous	14.3	11	35.3	50.63	15.7k
asynchronous (0)	12.2	13	35.61	50.47	19.5k
(0) + Aji and Heafield (2017)	6.23	12	35.16	50.86	26.9k
optimized asynchronous (1)	4.97	10	35.56	50.90	40.2k
(1) + Aji and Heafield (2017)	4.32	11	35.16	52.02	41.5k
(1) + delayed updates $\tau = 2$ (2)	4.20	11	34.82	51.68	44.2k
(2) + local optimizers (3)	3.66	10	35.45	51.32	44.2k
(3) + momentum tuning (4)	**3.66**	**10**	**35.48**	**50.87**	**44.2k**
(2) + warmup (5)	4.87	13	35.29	50.78	44.2k
(5) + momentum tuning (6)	3.98	11	35.76	50.73	44.2k

Table 2: Romanian-English results from our exploration and optimizations. We also compare our methods against the work of Aji and Heafield (2017) which also reduces communication. We use system (1) as our reference baseline upon which we improve. The system that achieved the best training time is bolded.

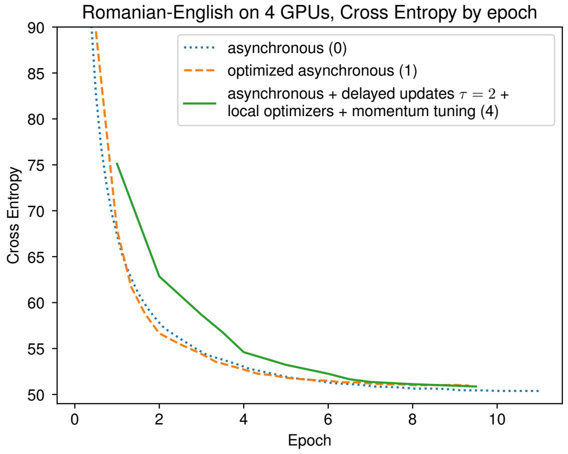

Figure 1: Cross-entropy training progression per epoch for our ro-en systems.

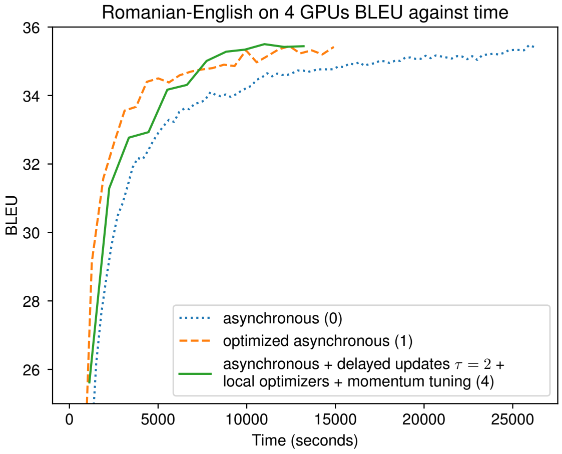

Figure 2: BLEU scores for our ro-en systems.

els. With deep RNN models the parameters take the majority of the available VRAM leaving very little for mini-batches. In this scenario we can apply $\tau = 4$ without negative effect towards convergence. We demonstrate the effectiveness of larger τ on Table 3. The baseline system is equivalent to the winning system for English-German at the WMT 2017 competition (Sennrich et al., 2017). The baseline is trained with synchronous SGD and our system uses asynchronous SGD, delayed gradient updates by a factor of 4, local optimizers and the momentum is tuned and further reduced after the first 16000 mini-batches. We found learning rate of 0.0007 to work the best. We do not report the numbers for asynchronous baseline because we were unable to achieve competitive BLEU scores without using delayed gradient updates. We speculate this is because with this type of deep model, our mini-batch size is very small leading to very jerky and unstable training updates. Larger mini-batches ensure the gradients produced by different workers are going to be closer to one another. Our training progression can be seen on figures 3 and 4. We show that even though we use 4 times larger mini-batches we actually manage to get lower Cross-Entropy epoch for epoch compared to the baseline (Figure 3). This coupled with out higher training speed makes our method reach the best BLEU score 1.6 times faster than the baseline (Figure 4).

3 Related work

We use larger mini-batches and delay gradient updates in order to increase the speed at which the dataset is processed. The principal reason why this works is because when mini-batch size is increased n (also includes delayed updates) times, communication is reduced by the same amount. This as-

System	**Time (h)**	**BLEU**	**CE**
Baseline	51.3	25.1	47.31
Async (4) + $\tau = 4$	39.7	25.07	46.59

Table 3: Training times for English-German deep RNN system trained on WMT17 data. Our asynchronous system includes the optimizations of system (4) from Table 2.

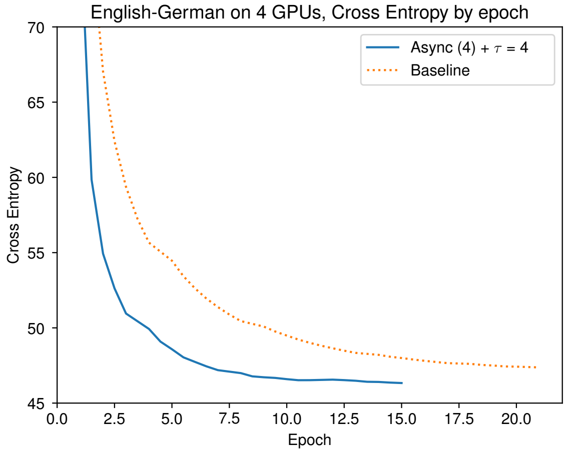

Figure 3: CE scores for our en-de systems.

pect of our work is similar to the work of Aji and Heafield (2017) where they drop the lower 99% of the gradient updates based on absolute value thus reducing the memory traffic. Compared with them we achieve faster dataset processing speed and also better model convergence as shown on Table 2.

Independently from us Mao et al. (2018) extend the work of Aji and Heafield (2017) aiming to reduce gradient communication without suffering any of the negative effects we have noted. In process they independently arrive to some of the methods that we use, notably tuning the momentum and applying warmup to achieve better convergence.

Independently from us Shazeer and Stern (2018) have done further exploratory work on ADAM's momentum parameters using the Transformer model (Vaswani et al., 2017) as a case study and have offered a mathematical explanation about why different stages of the training require different momentum values.[2]

Independently from us Saunders et al. (2018) have employed delayed gradient updates in syntax NMT setting, where the sequences are much longer due to the syntax annotation and delayed updates are necessary because video RAM is limited.[3]

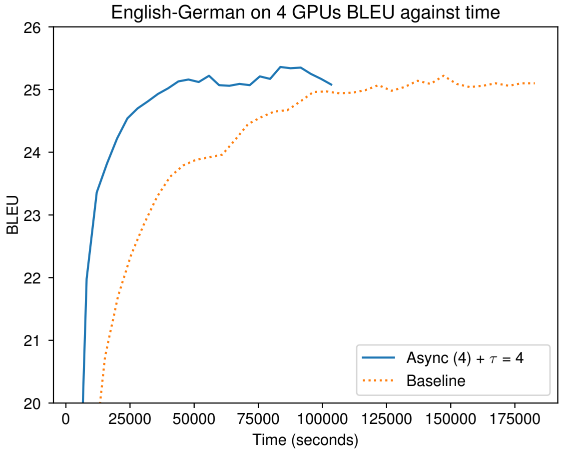

Figure 4: BLEU scores for our en-de systems.

Independently from us Lin et al. (2018) have developed their own local optimizer solution as an alternative to increasing mini-batch sizes.[4]

4 Conclusion and Future work

We show that we can increase speed and maintain convergence rate for very large mini-batch asynchronous SGD by carefully adjusting momentum and applying warmup and cooldown strategies. While we have demonstrated our methods on GPUs, they are hardware agnostic and can be applied to neural network training on any multi-device hardware such as TPUs or Xeon Phis. We were able to achieve end-to-end training on multiple tasks a lot faster than the baseline systems. For our Romanian-English model, we train nearly 3X faster than the commonly used baseline and 1.5X faster over a specifically optimised baseline. When experimenting with English-German we are able to train our model 1.3X faster than the baseline model, achieving practically the same BLEU score and much better model cross-entropy.

In the future we would like to apply local optimizers in distributed setting where the communication latency between local and remote devices varies significantly we could use local optimizers to synchronize remote models less often.

Acknowledgments

We thank Adam Lopez, Sameer Bansal and Naomi Saphra for their help and comments on the paper. We thank our reviewers for their comments and suggestions. Nikolay Bogoychev was funded by an Amazon faculty research award to Adam Lopez.

---

[2] This work was published on 11.04.2018.

[3] This work was published on 01.05.2018.

[4] This work was published on 22.08.2018

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zhang. 2016. Tensorflow: A system for large-scale machine learning. *CoRR*, abs/1605.08695.

Alham Fikri Aji and Kenneth Heafield. 2017. Sparse communication for distributed gradient descent. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. 2010. Optimal distributed online prediction using mini-batches. Technical report.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677.

S. Hadjis, C. Zhang, I. Mitliagkas, and C. Re. 2016. Omnivore: An optimizer for multi-device deep learning on cpus and gpus.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *Proceedings of ICLR 2017*, ICLR '17.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Tao Lin, Sebastian U. Stich, and Martin Jaggi. 2018. Don't use large mini-batches, use local sgd. *CoRR*, abs/1808.07217.

Huizi Mao, Song Han, Wang Yu, William Dally, and Yujun Lin. 2018. Deep gradient compression: Reducing the communication bandwidth for distributed training.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 693–701. Curran Associates, Inc.

Danielle Saunders, Felix Stahlberg, Adrià de Gispert, and Bill Byrne. 2018. Multi-representation ensembles and delayed sgd updates improve syntax-based nmt. *CoRR*, abs/1805.00456.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The University of Edinburgh's Neural MT Systems for WMT17. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, Copenhagen, Denmark.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Sixin Zhang, Anna Choromanska, and Yann LeCun. 2014. Deep learning with elastic averaging SGD. *CoRR*, abs/1412.6651.

Learning to Jointly Translate and Predict Dropped Pronouns with a Shared Reconstruction Mechanism

Longyue Wang
Tencent AI Lab
`vinnylywang@tencent.com`

Zhaopeng Tu[*]
Tencent AI Lab
`zptu@tencent.com`

Andy Way
Dublin City University
`andy.way@adaptcentre.ie`

Qun Liu
Huawei Noah's Ark Lab
`qun.liu@huawei.com`

Abstract

Pronouns are frequently omitted in pro-drop languages, such as Chinese, generally leading to significant challenges with respect to the production of complete translations. Recently, Wang et al. (2018) proposed a novel reconstruction-based approach to alleviating dropped pronoun (DP) translation problems for neural machine translation models. In this work, we improve the original model from two perspectives. First, we employ a shared reconstructor to better exploit encoder and decoder representations. Second, we jointly learn to translate and predict DPs in an end-to-end manner, to avoid the errors propagated from an external DP prediction model. Experimental results show that our approach significantly improves both translation performance and DP prediction accuracy.

1 Introduction

Pronouns are important in natural languages as they imply rich discourse information. However, in pro-drop languages such as Chinese and Japanese, pronouns are frequently omitted when their referents can be pragmatically inferred from the context. When translating sentences from a pro-drop language into a non-pro-drop language (*e.g.* Chinese-to-English), translation models generally fail to translate invisible dropped pronouns (DPs). This phenomenon leads to various translation problems in terms of completeness, syntax and even semantics of translations. A number of approaches have been investigated for DP translation (Le Nagard and Koehn, 2010; Xiang et al., 2013; Wang et al., 2016, 2018).

Wang et al. (2018) is a pioneering work to model DP translation for neural machine translation (NMT) models. They employ two *separate* reconstructors (Tu et al., 2017) to respectively reconstruct encoder and decoder representations back to the DP-annotated source sentence. The annotation of DP is provided by an *external* prediction model, which is trained on the parallel corpus using automatically learned alignment information (Wang et al., 2016). Although this model achieved significant improvements, there nonetheless exist two drawbacks: 1) there is no interaction between the two separate reconstructors, which misses the opportunity to exploit useful relations between encoder and decoder representations; and 2) the external DP prediction model only has an accuracy of 66% in F1-score, which propagates numerous errors to the translation model.

In this work, we propose to improve the original model from two perspectives. First, we use a *shared* reconstructor to read hidden states from both encoder and decoder. Second, we integrate a DP predictor into NMT to *jointly* learn to translate and predict DPs. Incorporating these as two auxiliary loss terms can guide both the encoder and decoder states to learn critical information relevant to DPs. Experimental results on a large-scale Chinese–English subtitle corpus show that the two modifications can accumulatively improve translation performance, and the best result is +1.5 BLEU points better than that reported by Wang et al. (2018). In addition, the jointly learned DP prediction model significantly outperforms its external counterpart by 9% in F1-score.

2 Background

As shown in Figure 1, Wang et al. (2018) introduced two independent reconstructors with their own parameters, which reconstruct the DP-annotated source sentence from the encoder and decoder hidden states, respectively. The central

---

[*] Zhaopeng Tu is the corresponding author of the paper. This work was conducted when Longyue Wang was studying and Qun Liu was working at the ADAPT Centre in the School of Computing at Dublin City University.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2997–3002
Brussels, Belgium, October 31 - November 4, 2018.

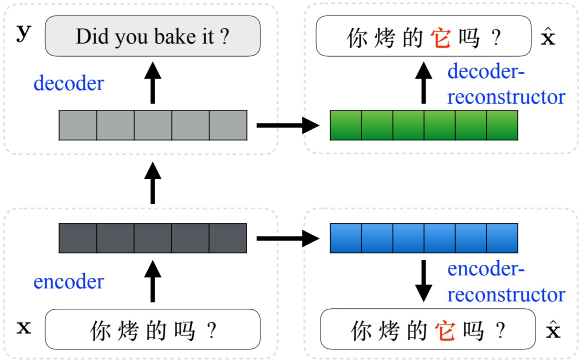

Figure 1: Architecture of separate reconstructors.

Prediction	F1-score	Example
DP Position	88%	你 烤 的 #DP# 吗 ?
DP Words	66%	你 烤 的 它 吗 ?

Table 1: Evaluation of external models on predicting the positions of DPs ("DP Position") and the exact words of DP ("DP Words").

idea underpinning their approach is to guide the corresponding hidden states to embed the recalled source-side DP information and subsequently to help the NMT model generate the missing pronouns with these enhanced hidden representations.

The DPs can be automatically annotated for training and test data using two different strategies (Wang et al., 2016). In the *training phase*, where the target sentence is available, we annotate DPs for the source sentence using alignment information. These annotated source sentences can be used to build a neural-based DP predictor, which can be used to annotate test sentences since the target sentence is not available during the *testing phase*. As shown in Table 1, Wang et al. (2016, 2018) explored to predict the exact DP words[1], the accuracy of which is only 66% in F1-score. By analyzing the translation outputs, we found that 16.2% of errors are newly introduced and caused by errors from the DP predictor. Fortunately, the accuracy of predicting DP positions (DPPs) is much higher, which provides the chance to alleviate the error propagation problem. Intuitively, we can learn to generate DPs at the predicted positions using a jointly trained DP predictor, which is fed with informative representations in the reconstructor.

3 Approach

3.1 Shared Reconstructor

Recent work shows that NMT models can benefit from sharing a component across different tasks and languages. Taking multi-language translation as an example, Firat et al. (2016) share an attention model across languages while Dong et al. (2015) share an encoder. Our work is most similar to the work of Zoph and Knight (2016) and Anastasopoulos and Chiang (2018), which share a decoder and two separate attention models to read from two different sources. In contrast, we share information at the level of reconstructed frames.

The architectures of our proposed shared reconstruction model are shown in Figure 2(a). Formally, the reconstructor reads from both the encoder and decoder hidden states, as well as the DP-annotated source sentence, and outputs a reconstruction score. It uses two separate attention models to reconstruct the annotated source sentence $\hat{\mathbf{x}} = \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_T\}$ word by word, and the reconstruction score is computed by

$$R(\hat{\mathbf{x}}|\mathbf{h}^{enc}, \mathbf{h}^{dec}) = \prod_{t=1}^{T} g_r(\hat{x}_{t-1}, \mathbf{h}_t^{rec}, \hat{\mathbf{c}}_t^{enc}, \hat{\mathbf{c}}_t^{dec})$$

where $\mathbf{h}_t^{rec}$ is the hidden state in the reconstructor, and computed by Equation (1):

$$\mathbf{h}_t^{rec} = f_r(\hat{x}_{t-1}, \mathbf{h}_{t-1}^{rec}, \hat{\mathbf{c}}_t^{enc}, \hat{\mathbf{c}}_t^{dec}) \quad (1)$$

Here $g_r(\cdot)$ and $f_r(\cdot)$ are respectively softmax and activation functions for the reconstructor. The context vectors $\hat{\mathbf{c}}_t^{enc}$ and $\hat{\mathbf{c}}_t^{dec}$ are the weighted sum of $\mathbf{h}^{enc}$ and $\mathbf{h}^{dec}$, respectively, as in Equation (2) and (3):

$$\hat{\mathbf{c}}_t^{enc} = \textstyle\sum_{j=1}^{J} \hat{\alpha}_{t,j}^{enc} \cdot \mathbf{h}_j^{enc} \quad (2)$$

$$\hat{\mathbf{c}}_t^{dec} = \textstyle\sum_{i=1}^{I} \hat{\alpha}_{t,i}^{dec} \cdot \mathbf{h}_i^{dec} \quad (3)$$

Note that the weights $\hat{\alpha}^{enc}$ and $\hat{\alpha}^{dec}$ are calculated by two separate attention models. We propose two attention strategies which differ as to whether the two attention models have interactions or not.

Independent Attention calculates the two weight matrices independently, as in Equation (4) and (5):

$$\hat{\alpha}^{enc} = \text{ATT}_{enc}(\hat{x}_{t-1}, \mathbf{h}_{t-1}^{rec}, \mathbf{h}^{enc}) \quad (4)$$

$$\hat{\alpha}^{dec} = \text{ATT}_{dec}(\hat{x}_{t-1}, \mathbf{h}_{t-1}^{rec}, \mathbf{h}^{dec}) \quad (5)$$

where $\text{ATT}_{enc}(\cdot)$ and $\text{ATT}_{dec}(\cdot)$ are two separate attention models with their own parameters.

---

[1] Unless otherwise indicated, in the paper, the terms "DP" and "DP word" are identical.

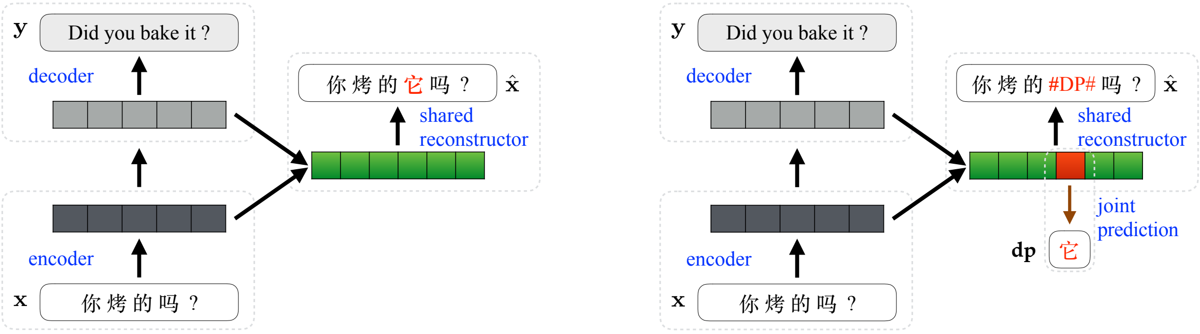

(a) Shared reconstructor.

(b) Shared reconstructor with joint prediction.

Figure 2: Model architectures in which the words in red are automatically annotated DPs and DPPs.

Interactive Attention feeds the context vector produced by one attention model to another attention model. The intuition behind this is that the interaction between two attention models can lead to a better exploitation of the encoder and decoder representations. As the interactive attention is directional, we have two options (Equation (6) and (7)) which modify either $\text{ATT}_{enc}(\cdot)$ or $\text{ATT}_{dec}(\cdot)$ while leaving the other one unchanged:

- *enc→dec*:

$$\hat{\alpha}^{dec} = \text{ATT}_{dec}(\hat{x}_{t-1}, \mathbf{h}_{t-1}^{rec}, \mathbf{h}^{dec}, \hat{\mathbf{c}}_t^{enc}) \quad (6)$$

- *dec→enc*:

$$\hat{\alpha}^{enc} = \text{ATT}_{enc}(\hat{x}_{t-1}, \mathbf{h}_{t-1}^{rec}, \mathbf{h}^{enc}, \hat{\mathbf{c}}_t^{dec}) \quad (7)$$

3.2 Joint Prediction of Dropped Pronouns

Inspired by recent successes of multi-task learning (Dong et al., 2015; Luong et al., 2016), we propose to jointly learn to translate and predict DPs (as shown in Figure 2(b)). To ease the learning difficulty, we leverage the information of DPPs predicted by an external model, which can achieve an accuracy of 88% in F1-score. Accordingly, we transform the original DP prediction problem to DP word generation given the pre-predicted DP positions. Since the DPP-annotated source sentence serves as the reconstructed input, we introduce an additional *DP-generation loss*, which measures how well the DP is generated from the corresponding hidden state in the reconstructor.

Let $\mathbf{dp} = \{dp_1, dp_2, \dots, dp_D\}$ be the list of DPs in the annotated source sentence, and $\mathbf{h}^{rec} = \{\mathbf{h}_1^{rec}, \mathbf{h}_2^{rec}, \dots, \mathbf{h}_D^{rec}\}$ be the corresponding hidden states in the reconstructor. The generation probability is computed by

$$\begin{aligned} P(\mathbf{dp}|\mathbf{h}^{rec}) &= \prod_{d=1}^{D} P(dp_d|\mathbf{h}_d^{rec}) \\ &= \prod_{d=1}^{D} g_p(dp_d|\mathbf{h}_d^{rec}) \end{aligned} \quad (8)$$

where $g_p(\cdot)$ is softmax for the DP predictor.

3.3 Training and Testing

We train both the encoder-decoder and the shared reconstructors together in a single end-to-end process, and the training objective is

$$\begin{aligned} J(\theta, \gamma, \psi) = \arg\max_{\theta,\gamma,\psi} \Big\{ & \underbrace{\log L(\mathbf{y}|\mathbf{x}; \theta)}_{likelihood} \\ & + \underbrace{\log R(\hat{\mathbf{x}}|\mathbf{h}^{enc}, \mathbf{h}^{dec}; \theta, \gamma)}_{reconstruction} \\ & + \underbrace{\log P(\mathbf{dp}|\hat{\mathbf{h}}^{rec}; \theta, \gamma, \psi)}_{prediction} \Big\} \end{aligned} \quad (9)$$

where $\{\theta, \gamma, \psi\}$ are respectively the parameters associated with the encoder-decoder, shared reconstructor and the DP prediction model. The auxiliary reconstruction objective $R(\cdot)$ guides the related part of the parameter matrix θ to learn better latent representations, which are used to reconstruct the DPP-annotated source sentence. The auxiliary prediction loss $P(\cdot)$ guides the related part of both the encoder-decoder and the reconstructor to learn better latent representations, which are used to predict the DPs in the source sentence.

Following Tu et al. (2017) and Wang et al. (2018), we use the reconstruction score

#	Model	#Params	Speed		BLEU
			Train	Decode	
	Existing system (Wang et al., 2018)				
1	Baseline	86.7M	1.60K	15.23	31.80
2	Baseline (+DPs)	86.7M	1.59K	15.20	32.67
3	Separate-Recs⇒(+DPs)	+73.8M	0.57K	12.00	35.08
	Our system				
4	Baseline (+DPPs)	86.7M	1.54K	15.19	33.18
5	Shared-Rec$_{independent}$⇒(+DPPs)	+86.6M	0.52K	11.87	35.27†‡
6	Shared-Rec$_{independent}$⇒(+DPPs) + joint prediction	+87.9M	0.51K	11.88	35.88†‡
7	Shared-Rec$_{enc\rightarrow dec}$⇒(+DPPs) + joint prediction	+91.9M	0.48K	11.84	**36.53**†‡
8	Shared-Rec$_{dec\rightarrow enc}$⇒(+DPPs) + joint prediction	+89.9M	0.49K	11.85	35.99†‡

Table 2: Evaluation of translation performance for Chinese–English. "Baseline" is trained and evaluated on the original data, while "Baseline (+DPs)" and "Baseline (+DPPs)" are trained on the data annotated with DPs and DPPs, respectively. Training and decoding (beam size is 10) speeds are measured in words/second. "†" and "‡" indicate statistically significant difference ($p < 0.01$) from "Baseline (+DDPs)" and "Separate-Recs⇒(+DPs)", respectively.

as a reranking technique to select the best translation candidate from the generated n-best list at testing time. Different from Wang et al. (2018), we reconstruct DPP-annotated source sentence, which is predicted by an external model.

4 Experiment

4.1 Setup

To compare our work with the results reported by previous work (Wang et al., 2018), we conducted experiments on their released Chinese⇒English TV Subtitle corpus.[2] The training, validation, and test sets contain 2.15M, 1.09K, and 1.15K sentence pairs, respectively. We used case-insensitive 4-gram NIST BLEU metrics (Papineni et al., 2002) for evaluation, and *sign-test* (Collins et al., 2005) to test for statistical significance.

We implemented our models on the code repository released by Wang et al. (2018).[3] We used the same configurations (*e.g.* vocabulary size = 30K, hidden size = 1000) and reproduced their reported results. It should be emphasized that we did not use the pre-train strategy as done in Wang et al. (2018), since we found training from scratch achieved a better performance in the shared reconstructor setting.

4.2 Results

Table 2 shows the translation results. It is clear that the proposed models significantly outperform the baselines in all cases, although there are considerable differences among different variations.

Baselines (Rows 1-4): The three baselines (Rows 1, 2, and 4) differ regarding the training data used. "Separate-Recs⇒(+DPs)" (Row 3) is the best model reported in Wang et al. (2018), which we employed as another strong baseline. The baseline trained on the DPP-annotated data ("Baseline (+DPPs)", Row 4) outperforms the other two counterparts, indicating that the error propagation problem does affect the performance of translating DPs. It suggests the necessity of jointly learning to translate and predict DPs.

Our Models (Rows 5-8): Using our shared reconstructor (Row 5) not only outperforms the corresponding baseline (Row 4), but also surpasses its separate reconstructor counterpart (Row 3). Introducing a joint prediction objective (Row 6) can achieve a further improvement of +0.61 BLEU points. These results verify that shared reconstructor and jointly predicting DPs can accumulatively improve translation performance.

Among the variations of shared reconstructors (Rows 6-8), we found that an interaction attention from encoder to decoder (Row 7) achieves the best performance, which is +3.45 BLEU points better than our baseline (Row 4) and +1.45 BLEU points

---

[2] `https://github.com/longyuewangdcu/tvsub`

[3] `https://github.com/tuzhaopeng/nmt`

better than the best result reported by Wang et al. (2018) (Row 3). We attribute the superior performance of "Shared-Rec$_{enc \rightarrow dec}$" to the fact that the attention context over encoder representations embeds useful DP information, which can help to better attend to the representations of the corresponding pronouns in the decoder side. Similar to Wang et al. (2018), the proposed approach improves BLEU scores at the cost of decreased training and decoding speed, which is due to the large number of newly introduced parameters resulting from the incorporation of reconstructors into the NMT model.

4.3 Analysis

Models	Precision	Recall	F1-score
External	0.67	0.65	0.66
Joint	0.74	0.76	0.75

Table 3: Evaluation of DP prediction accuracy. "External" model is *separately* trained on DP-annotated data with external neural methods (Wang et al., 2016), while "Joint" model is *jointly* trained with the NMT model (Section 3.2).

DP Prediction Accuracy As shown in Table 3, the jointly learned model significantly outperforms the external one by 9% in F1-score. We attribute this to the useful contextual information embedded in the reconstructor representations, which are used to generate the exact DP words.

Model	Test	△
Baseline (+DPPs)	33.18	–
Separate-Recs (+DPs)	34.02	+0.84
Shared-Rec (+DPPs)	**34.80**	**+1.62**

Table 4: Translation results when *reconstruction is used in training only while not used in testing*.

Contribution Analysis Table 4 lists translation results when the reconstruction model is used in training only. We can see that the proposed model outperforms both the strong baseline and the best model reported in Wang et al. (2018). This is encouraging since no extra resources and computation are introduced to online decoding, which makes the approach highly practical, for example for translation in industry applications.

Model	Auto.	Man.	△
Seperate-Recs (+DPs)	35.08	38.38	+3.30
Shared-Rec (+DPPs)	36.53	38.94	+2.41

Table 5: Translation performance gap ("△") between manually ("Man.") and automatically ("Auto.") labelling DPs/DPPs for input sentences in testing.

Effect of DPP Labelling Accuracy For each sentence in testing, the DPs and DPPs are labelled automatically by two separate external prediction models, the accuracy of which are respectively 66% and 88% measured in F1 score. We investigate the best performance the models can achieve with manual labelling, which can be regarded as an "Oracle", as shown in Table 5. As seen, there still exists a significant gap in performance, and this could be improved by improving the accuracy of our DPP generator. In addition, our models show a relatively smaller distance in performance from the oracle performance ("Man"), indicating that the error propagation problem is alleviated to some extent.

5 Conclusion

In this paper, we proposed effective approaches of translating DPs with NMT models: *shared* reconstructor and *jointly* learning to translate and predict DPs. Through experiments we verified that 1) shared reconstruction is helpful to share knowledge between the encoder and decoder; and 2) joint learning of the DP prediction model indeed alleviates the error propagation problem by improving prediction accuracy. The two approaches accumulatively improve translation performance. The method is not restricted to the DP translation task and could potentially be applied to other sequence generation problems where additional source-side information could be incorporated.

In future work we plan to: 1) build a fully end-to-end NMT model for DP translation, which does not depend on any external component (*i.e.* DPP predictor); 2) exploit cross-sentence context (Wang et al., 2017) to further improve DP translation; 3) investigate a new research strand that adapts our model in an inverse translation direction by learning to drop pronouns instead of recovering DPs.

Acknowledgments

The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund. We thank the anonymous reviewers for their insightful comments.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 82–91, New Orleans, Louisiana, USA.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 531–540, Ann Arbor, Michigan, USA.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1723–1732, Beijing, China.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California, USA.

Ronan Le Nagard and Philipp Koehn. 2010. Aiding pronoun translation with co-reference resolution. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR*, pages 252–261, Uppsala, Sweden.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2965–2977, Santa Fe, New Mexico, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3097–3103, San Francisco, California, USA.

Longyue Wang, Zhaopeng Tu, Shuming Shi, Tong Zhang, Yvette Graham, and Qun Liu. 2018. Translating pro-drop languages with reconstruction models. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4937–4945, New Orleans, Louisiana, USA.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2816–2821, Copenhagen, Denmark.

Longyue Wang, Zhaopeng Tu, Xiaojun Zhang, Hang Li, Andy Way, and Qun Liu. 2016. A novel approach for dropped pronoun translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 983–993, San Diego, California, USA.

Bing Xiang, Xiaoqiang Luo, and Bowen Zhou. 2013. Enlisting the ghost: Modeling empty categories for machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 822–831, Sofia, Bulgaria.

Barret Zoph and Knight Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California.

Getting Gender Right in Neural Machine Translation

Eva Vanmassenhove$^{\alpha}$ **Christian Hardmeier**$^{\beta}$ **Andy Way**$^{\alpha}$

$^{\alpha}$ ADAPT, School of Computing, Dublin City University, Dublin, Ireland
`firstname.lastname@adaptcentre.ie`

$^{\beta}$ Department of Linguistics and Philology, Uppsala University, Uppsala, Sweden
`christian.hardmeier@lingfil.uu.se`

Abstract

Speakers of different languages must attend to and encode strikingly different aspects of the world in order to use their language correctly (Sapir, 1921; Slobin, 1996). One such difference is related to the way gender is expressed in a language. Saying "I am happy" in English, does not encode any additional knowledge of the speaker that uttered the sentence. However, many other languages do have grammatical gender systems and so such knowledge would be encoded. In order to correctly translate such a sentence into, say, French, the inherent gender information needs to be retained/recovered. The same sentence would become either "Je suis heureux", for a male speaker or "Je suis heureuse" for a female one. Apart from morphological agreement, demographic factors (gender, age, etc.) also influence our use of language in terms of word choices or even on the level of syntactic constructions (Tannen, 1991; Pennebaker et al., 2003). We integrate gender information into NMT systems. Our contribution is twofold: (1) the compilation of large datasets with speaker information for 20 language pairs, and (2) a simple set of experiments that incorporate gender information into NMT for multiple language pairs. Our experiments show that adding a gender feature to an NMT system significantly improves the translation quality for some language pairs.

1 Introduction

In the field of linguistics, the differences between male and female traits within spoken and written language have been studied both empirically and theoretically, revealing that the language used by males and females differs in terms of style and syntax (Coates, 2015). The increasing amount of work on automatic author classification (or 'author profiling') reaching relatively high accuracies on domain-specific data corroborates these findings (Rangel et al., 2013; Santosh et al., 2013). However, determining the gender of an author based solely on text is not a solved issue. Likewise, the selection of the most informative features for gender classification remains a difficult task (Litvinova et al., 2016).

When translating from one language into another, original author traits are partially lost, both in human and machine translations (Mirkin et al., 2015; Rabinovich et al., 2017). However, in the field of Machine Translation (MT) one of the most observable consequences of this missing information are morphologically incorrect variants due to a lack of agreement in number and gender with the subject. Such errors harm the overall fluency and adequacy of the translated sentence. Furthermore, gender-related errors are not just harming the quality of the translation as getting the gender right is also a matter of basic politeness. Current systems have a tendency to perpetuate a male bias which amounts to negative discrimination against half the population and this has been picked up by the media.[1]

Human translators rely on contextual information to infer the gender of the speaker in order to make the correct morphological agreement. However, most current MT systems do not; they simply exploit statistical dependencies on the sentence level that have been learned from large amounts of parallel data. Furthermore, sentences are translated in isolation. As a consequence, pieces of information necessary to determine the gender of the speakers, might get lost. The MT system will, in such cases, opt for the statistically most likely variant, which depending on the training data, will

---

[1] `https://www.theguardian.com/technology/2017/apr/13/ai-programs-exhibit-racist-and-sexist-biases-research`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3003–3008
Brussels, Belgium, October 31 - November 4, 2018.

be either the male or the female form. Additionally, in the field of MT, training data often consists of both original and translated parallel texts: large parts of the texts have already been translated, which, as studied by Mirkin et al. (2015), does not preserve the original demographic and psychometric traits of the author, making it very hard for a Neural MT (NMT) system to determine the gender of the author.

With this in mind, a first step towards the preservation of author traits would be their integration into an NMT system. As ‘gender’ manifests itself not only in the agreement with other words in a sentence, but also in the choice of context-based words or on the level of syntactic constructions, the sets of experiments conducted in this paper focus on the integration of a gender feature into NMT for multiple language pairs.

The structure of the paper is the following: related work is described in Section 2; Section 3 describes and analyses the datasets that were compiled; the experimental setup is discussed in Section 4; the results are presented in Section 5; finally, we conclude and provide some ideas for future work in Section 6.

2 Related Work

Differences in the language between male and female speakers have been studied within various fields related to linguistics, including Natural Language Processing (NLP) for author profiling, conversational agents, recommendation systems etc. Mirkin et al. (2015) motivated the need for more personalized MT. Their experiments show that MT is detrimental to the automatic recognition of linguistic signals of traits of the original author/speaker. Their work suggests using domain-adaptation techniques to make MT more personalized but does not include any actual experiments on the inclusion of author traits in MT.

Rabinovich et al. (2017) conducted a series of experiments on preserving original author traits, focusing particularly on gender. As suggested by Mirkin et al. (2015), they treat the personalization of Statistical MT (SMT) systems as a domain-adaptation task treating the female and male gender as two different domains. They applied two common simple domain-adaptation techniques in order to create personalized SMT: (1) using gender-specific phrase-tables and language models, and (2) using a gender-specific tuning set. Although their models did not improve over the baseline, their work provides a detailed analysis of gender traits in human and machine translation.

Our work is, to the best of our knowledge, the first to attempt building a speaker-informed NMT system. Our approach is similar to the work of Sennrich et al. (2016) on controlling politeness, where some sentence of the training data are followed with an ‘informal’ or ‘polite’ tag indicating the level of politeness expressed.

3 Compilation of Datasets

One of the main obstacles for more personalized MT systems is finding large enough annotated parallel datasets with speaker information. Rabinovich et al. (2017) published an annotated parallel dataset for EN–FR and EN–DE. However, for many other language pairs no sufficiently large annotated datasets are available.

To address the aforementioned problem, we published online a collection of parallel corpora licensed under the Creative Commons Attribution 4.0 International License for 20 language pairs (Vanmassenhove and Hardmeier, 2018).[2]

We followed the approach described by Rabinovich et al. (2017) and tagged parallel sentences from Europarl (Koehn, 2005) with speaker information (name, gender, age, date of birth, euroID and date of the session) by retrieving speaker information provided by tags in the Europarl source files. The Europarl source files contain information about the speaker on the paragraph level and the filenames contain the data of the session. By retrieving the names of the speakers together with meta-information on the members of the European Parliament (MEPs) released by Rabinovich et al. (2017) (which includes among others name, country, date of birth and gender predictions per MEP), we were able to retrieve demographic annotations (gender, age, etc.). An overview of the language pairs as well as the amount of annotated parallel sentences per language pair is given in Table 1.

3.1 Analysis of the EN–FR Annotated Dataset

We first analysed the distribution of male and female sentence in our data. In the 10 different

---

[2] `https://github.com/evavnmssnhv/Europarl-Speaker-Information`

Languages	**# sents**	**Languages**	**# sents**
EN–BG	306,380	**EN–IT**	1,297,635
EN–CS	491,848	**EN–LT**	481,570
EN–DA	1,421,197	**EN–LV**	487,287
EN–DE	1,296,843	**EN–NL**	1,419,359
EN–EL	921,540	**EN–PL**	478,008
EN–ES	1,419,507	**EN–PT**	1,426,043
EN–ET	494,645	**EN–RO**	303,396
EN–FI	1,393,572	**EN–SK**	488,351
EN–FR	1,440,620	**EN–SL**	479,313
EN–HU	251,833	**EN–SV**	1,349,472

Table 1: Overview of annotated parallel sentences per language pair

datasets we experimented with, the percentage of sentences uttered by female speakers is very similar, ranging between 32% and 33%. This similarity can be explained by the fact that Europarl is largely a multilingual corpus with a big overlap between the different language pairs.

We conducted a more focused analysis on one of the subcorpora (EN–FR) with respect to the percentage of sentences uttered by males/females for various age groups to obtain a better grasp of what kind of data we are using for training. As can be seen from Figure 1, with the exception of the youngest age group (20–30), which represents only a very small percentage of the total amount of sentences (0.71%), more male data is available in all age groups. Furthermore, when looking at the entire dataset, 67.39% of the sentences are produced by male speakers. Moreover, almost half of the total number of sentences are uttered by the 50–60 age group (43.76%).

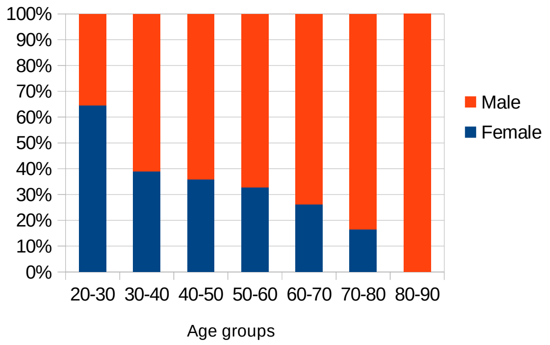

Figure 1: Percentage of female and male speakers per age group

The analysis shows that indeed, there is a gender unbalance in the Europarl dataset, which will be reflected in the translations that MT systems trained on this data produce.

4 Experimental Setup

4.1 Datasets

We carried out a set of experiments on 10 language pairs (the ones for which we compiled more than 500k annotated Europarl parallel sentences): EN–DE, EN–FR, EN–ES, EN–EL, EN–PT, EN–FI, EN–IT, EN–SV, EN–NL and EN–DA. We augmented every sentence with a tag on the English source side, identifying the gender of the speaker, as illustrated in (1). This approach for encoding sentence-specific information for NMT has been successfully exploited to tackle other types of issues, multilingual NMT systems (e.g., Zero Shot Translation (Johnson et al., 2017)), domain adaptation (Sennrich et al., 2016), etc.

(1) “FEMALE Madam President, as a...”

For each of these language pairs we trained two NMT systems: a baseline and a tagged one. We evaluated the performance of all our systems on a randomly selected 2K general test set. Moreover, we further evaluated the EN–FR systems on 2K male-only and female-only test sets to have a look at the system performance with respect to gender-related issues. We also looked at two additional male and female test sets in which the first person singular pronoun appeared.

4.2 Description of the NMT Systems

We used the OpenNMT-py toolkit (Klein et al., 2017) to train the NMT models. The models are sequence-to-sequence encoder-decoders with LSTMs as the recurrent unit (Bahdanau et al., 2014; Cho et al., 2014; Sutskever et al., 2014) trained with the default parameters. In order to bypass the OOV problem and reduce the number of dictionary entries, we use word-segmentation with BPE (Sennrich, 2015). We ran the BPE algorithm with 89,500 operations (Sennrich, 2015). All systems are trained for 13 epochs and the best model is selected for evaluation.

5 Results

In this section we discuss some of the results obtained. We hypothesized that the male/female tags would be particularly helpful for French, Portuguese, Italian, Spanish and Greek, where adjectives and even verb forms can be marked by the gender of the speaker. Since, according to the literature, women and men also make use of different syntactic constructions and make different word choices, we also tested the approach on other languages that do not have morphological agreement with the gender of the speaker such as Danish (DA), Dutch (NL), Finnish (FI), German (DE) and Swedish (SV).

First, we wanted to see how our tagged systems performed on the general test set compared to the baseline. In Table 2, the BLEU scores for 10 baseline and 10 gender-enhanced NMT systems are presented.

Systems	**EN**	**EN-TAG**
FR	37.82	**39.26***
ES	42.47	42.28
EL	31.38	**31.54**
IT	31.46	**31.75***
PT	36.11	**36.33**
DA	36.69	**37.00***
DE	28.28	28.05
FI	21.82	21.35*
SV	35.42	35.19
NL	28.35	28.22

Table 2: BLEU scores for the 10 baseline (denoted with **EN**) and the 10 gender-enhanced NMT (denoted with **EN-TAG**) systems. Entries labeled with * present statistically significant differences ($p < 0.05$). Statistical significance was computed with the MultEval tool (Clark et al., 2011).

While most of the BLEU-scores (Papineni et al., 2002) in Table 2 are consistent with our hypothesis, showing (significant) improvements for the NMT systems enriched with a gender tag (EN-TAG) over the baseline systems (EN) for French, Italian, Portuguese and Greek, the Spanish enriched system surprisingly does not (–0.19 BLEU). As hypothesized, the Dutch, German, Finnish and Swedish systems do not improve. However, the Danish (EN–DA) enriched NMT system does achieve a significant +0.31 BLEU improvement.

We expected to see the strongest improvements in sentences uttered by female speakers as, according to our initial analysis, the male data was overrepresented in the training. To test this hypothesis, we evaluated all systems on a male-only and female-only test set. Furthermore, we also experimented on test sets containing the pronoun of the first person singular as this form is used when a speaker refers to himself/herself. The results on the specific test set for the EN–FR dataset are presented in Table 3. As hypothesized, the biggest BLEU score improvement is observed on the female test set, particularly, the test sets containing first person singular pronouns (F1).

Test Sets	**EN**	**EN-TAG**
FR (M)	37.58	**38.71***
FR (F)	37.75	**38.97***
FR (M1)	39.00	**39.66***
FR (F1)	37.32	**38.57***

Table 3: BLEU-scores on EN–FR comparing the baseline (EN) and the tagged systems (EN–TAG) on 4 different test sets: a test set containing only male data (M), only female data (F), 1st person male data (M1) and first person female data (F1). All the improvements of the EN-TAG system are statistically significant ($p < 0.5$), as indicated by *.

We had a closer look at some of the translations.[3] There are cases where the gender-informed (TAG) system improves over the baseline (BASE) due to better agreement. Interestingly, in (2) the French female form of vice-president (vice-présidente) appears in the translation produced by the BASE system while the male form is the correct one. The gender-informed system does make the correct agreement by using the female variant. In (3) the speaker is female but the baseline system outputs a male form of the adjective 'happy' ('heureux').

(2) (Ref) En tant que *vice-président*...
(BASE) En tant que *vice-présidente*...
(TAG) En tant que *vice-président*...

(3) (Ref) ... je suis *heureuse* que...
(BASE) ... je suis *heureux* que...
(TAG) ... je suis *heureuse* que...

However, we also encountered cases where the gender-informed system fails to produce the correct agreement, as in (4), where both the BASE and the TAG system produce a male form ('embarassé') instead of the correct female one ('embarassée' or 'gênée').

(4) (Ref) je suis *gênée* que...
(BASE) je suis *embarassé* que...
(TAG) je suis *embarassé* que...

For some language pairs the gender-informed system leads to a significant improvement even on a general test set. This implies that the improvement is not merely because of better morphological agreement, as these kinds of improvements are very hard to measure with BLEU, especially given the fact that Europarl consists of formal spoken language and does not contain many sentences using the first person singular pronoun. From our analysis, we observe that in many cases the gender-informed systems have a higher BLEU

---

[3] We used the tool provided by Tilde `https://www.letsmt.eu/Bleu.aspx` to see where the BLEU score between the baseline and our tagged systems varied the most.

score than the baseline system due to differences in word choices as in (5) and (6), where both translations are correct, but the gender-informed system picks the preferred variant.

The observations with respect to differences in word preferences between male and female speakers are in accordance with corpus linguistic studies, which have shown that gender does not only have an effect on morphological agreement, but also manifests itself in other ways as males and females have different preferences when it comes to different types of constructions, word choices etc. (Newman et al., 2008; Coates, 2015). This also implies that, even for languages that do not mark gender overtly (i.e. grammatically), it can still be beneficial to take the gender of the author/speaker into account.

(5) (Ref) Je pense que ...
(BASE) Je crois que...
(TAG) Je pense que...

Although more research is required in order to draw general conclusions on this matter, from other linguistic studies, it appears that it is indeed the case that there is a relation between the use of the word "pense" ("think") / "crois" ("believe") and the gender of the speaker. To see whether there is a difference in word choice and whether this is reflected in our data, we compiled a list of the most frequent French words for the male data and the female data. Our analysis reveals that "crois" is, in general, used more by males (having position 303 in the most frequent words for males, but only position 373 for females), while "pense" is found at a similar position in both lists (position 151 and 153). These findings are in accordance with other linguistic corpus studies on language and gender stating that women use less assertive speech (Newman et al., 2008). "Croire" and "penser" are both verbs of cognition but there is a difference in the degree of confidence in the truth value predicated: the verb "croire" denotes more confidence in the truth of the complement clause than the verb "penser" does. In the future, we would like to perform a more detailed analysis of other specific differences in lexical choices between males and females on multiple language pairs.

(6) (Ref) J' ai plusieurs remarques...
(BASE) J' ai un nombre de commentaires...
(TAG) J' ai plusieurs remarques...

6 Conclusions and Future Work

In this work, we experimented with the incorporation of speaker-gender tags during the training of NMT systems in order to improve morphological agreement. We focused particularly on language pairs that express grammatical gender but included other language pairs as well, as linguistic studies have shown that the style and syntax of language used by males and females differs (Coates, 2015).

From the experiments, we see that informing the NMT system by providing tags indicating the gender of the speaker can indeed lead to significant improvements over state-of-the-art baseline systems, especially for those languages expressing grammatical gender agreement. However, while analyzing the EN–FR translations, we observed that the improvements are not always consistent and that, apart from morphological agreement, the gender-aware NMT system differs from the baseline in terms of word choices.

In the future, we would like to conduct further manual evaluation on the translations to further analyze the differences with the baseline system. Furthermore, we aim to experiment with other ways of integrating speaker information. We envisage working on gender classification techniques in order to work on other types (more informal) of corpora that are more likely to express speaker characteristics.

Acknowledgements

This work has been supported by COST action IS1312, the Dublin City University Faculty of Engineering & Computing under the Daniel O'Hare Research Scholarship scheme and by the ADAPT Centre for Digital Content Technology, which is funded under the SFI Research Centres Programme (Grant 13/RC/2106). Christian Hardmeier was supported by the Swedish Research Council under grant 2017-930.

We would also like to thank the anonymous reviewers for their insightful comments and feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*, Banff, Canada.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of EMNLP 2014*, pages 1724–1734, Doha, Qatar.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181. Association for Computational Linguistics.

Jennifer Coates. 2015. *Women, Men and Language: A Sociolinguistic Account of Gender Differences in Language*. Routledge, London.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association of Computational Linguistics*, 5(1):339–351.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proceeding of ACL, Vancouver, Canada*.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *MT Summit*, volume 5, pages 79–86, Phuket, Thailand.

Tatiana Litvinova, Pavel Seredin, Olga Litvinova, Olga Zagorovskaya, Aleksandr Sboev, Dmitry Gudovskih, Ivan Moloshnikov, and Roman Rybka. 2016. Gender Prediction for Authors of Russian Texts Using Regression And Classification Techniques. In *Proceedings of the Third Workshop on Concept Discovery in Unstructured Data co-located with the 13th International Conference on Concept Lattices and Their Applications (CDUD@ CLA)*, pages 44–53, Moscow, Russia.

Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating Personality-Aware Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1102–1108, Lisbon, Portugal.

Matthew L Newman, Carla J Groom, Lori D Handelman, and James W Pennebaker. 2008. Gender Differences in Language Use: An Analysis of 14,000 Text Samples. *Discourse Processes*, 45(3):211–236.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

James W. Pennebaker, Matthias R. Mehl, and Kate G. Niederhoffer. 2003. Psychological Aspects of Natural Language Use: Our words, Our Selves. *Annual review of psychology*, 54(1):547–577.

Ella Rabinovich, Raj Nath Patel, Shachar Mirkin, Lucia Specia, and Shuly Wintner. 2017. Personalized Machine Translation: Preserving Original Author Traits. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1074–1084, Valencia, Spain.

Francisco Rangel, Paolo Rosso, Moshe Koppel, Efstathios Stamatatos, and Giacomo Inches. 2013. Overview of The Author Profiling Task at PAN 2013. In *CLEF Conference on Multilingual and Multimodal Information Access Evaluation*, pages 352–365.

K Santosh, Romil Bansal, Mihir Shekhar, and Vasudeva Varma. 2013. Author profiling: Predicting age and gender from blogs. *Notebook for PAN at CLEF*, pages 119–124.

Edward Sapir. 1921. Language: An Introduction to the Study of Speech. *NewYork: Harcourt Brace & Company*.

Rico Sennrich. 2015. Modelling and Optimizing on Syntactic N-grams for Statistical Machine Translation. *Transactions of the Association for Computational Linguistics*, 3:169–182.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling Politeness in Neural Machine Translation via Side Constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, Berlin, Germany.

Dan I. Slobin. 1996. From Thought and Language to Thinking for Speaking. In J. Gumperz and S. Levinson, editors, *Rethinking Linguistic Relativity*, pages 70–96. Cambridge University Press, Cambridge.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, pages 3104–3112, Montreal, Quebec, Canada.

Deborah Tannen. 1991. *You Just Don't Understand*. Ballantine Books, New York, USA.

Eva Vanmassenhove and Christian Hardmeier. 2018. Europarl Datasets with Demographic Speaker Information. In *EAMT*, Alicante, Spain.

Towards Two-Dimensional Sequence to Sequence Model in Neural Machine Translation

Parnia Bahar, Christopher Brix and Hermann Ney
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
`<surname>@cs.rwth-aachen.de`

Abstract

This work investigates an alternative model for neural machine translation (NMT) and proposes a novel architecture, where we employ a multi-dimensional long short-term memory (MDLSTM) for translation modeling. In the state-of-the-art methods, source and target sentences are treated as one-dimensional sequences over time, while we view translation as a two-dimensional (2D) mapping using an MDLSTM layer to define the correspondence between source and target words. We extend beyond the current sequence to sequence backbone NMT models to a 2D structure in which the source and target sentences are aligned with each other in a 2D grid. Our proposed topology shows consistent improvements over attention-based sequence to sequence model on two WMT 2017 tasks, German↔English.

1 Introduction

The widely used state-of-the-art neural machine translation (NMT) systems are based on an encoder-decoder architecture equipped with attention layer(s). The encoder and the decoder can be constructed using recurrent neural networks (RNNs), especially long-short term memory (LSTM) (Bahdanau et al., 2014; Wu et al., 2016), convolutional neural networks (CNNs) (Gehring et al., 2017), self-attention units (Vaswani et al., 2017), or a combination of them (Chen et al., 2018). In all these architectures, source and target sentences are handled separately as a one-dimensional sequence over time. Then, an attention mechanism (additive, multiplicative or multihead) is incorporated into the decoder to selectively focus on individual parts of the source sentence.

One of the weaknesses of such models is that the encoder states are computed only once at the beginning and are left untouched with respect to the target histories. In this case, at every decoding step, the same set of vectors are read repeatedly. Hence, the attention mechanism is limited in its ability to effectively model the coverage of the source sentence. By providing the encoder states with the greater capacity to remember what has been generated and what needs to be translated, we believe that we can alleviate the coverage problems such as over- and under-translation.

One solution is to assimilate the context from both source and target sentences jointly and to align them in a two-dimensional grid. Two-dimensional LSTM (2DLSTM) is able to process data with complex interdependencies in a 2D space (Graves, 2012).

To incorporate the solution, in this work, we propose a novel architecture based on the 2DLSTM unit, which enables the computation of the encoding of the source sentence as a function of the previously generated target words. We treat translation as a 2D mapping. One dimension processes the source sentence, and the other dimension generates the target words. Each time a target word is generated, its representation is used to compute a hidden state sequence that models the source sentence encoding. In principle, by updating the encoder states across the second dimension using the target history, the 2DLSTM captures the coverage concepts internally by its cell states.

2 Related Works

MDLSTM (Graves, 2008, 2012) has been successfully used in handwriting recognition (HWR) to automatically extract features from raw images which are inherently two-dimensional (Graves and Schmidhuber, 2008; Leifert et al., 2016a; Voigtlaender et al., 2016). Voigtlaender et al. (2016) explore a larger MDLSTM for deeper and wider architectures using an implementation for the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3009–3015
Brussels, Belgium, October 31 - November 4, 2018.

graphical processing unit (GPU). It has also been applied to automatic speech recognition (ASR) where a 2DLSTM scans the input over both time and frequency jointly (Li et al., 2016; Sainath and Li, 2016). As an alternative architecture to the concept of MDLSTM, Kalchbrenner et al. (2015) propose a grid LSTM that is a network of LSTM cells arranged in a multidimensional grid, in which the cells are communicating between layers as well as time recurrences. Li et al. (2017) also apply the grid LSTM architecture for the endpoint detection task in ASR.

This work, for the first time, presents an end-to-end 2D neural model where we process the source and the target words jointly by a 2DLSTM layer.

3 Two-Dimensional LSTM

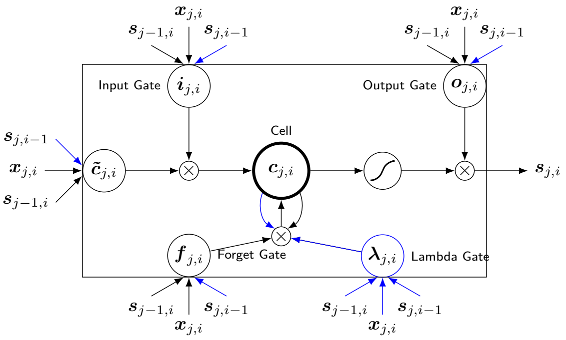

Figure 1: 2DLSTM unit. The additional links vs. standard LSTM are marked in blue.

The 2DLSTM has been introduced by (Graves, 2008) as a generalization of standard LSTM. Figure 1 illustrates one of the stable variants proposed by (Leifert et al., 2016b). A 2DLSTM unit processes a 2D sequential data $x \in \mathbb{R}^{J \times I}$ of arbitrary lengths, J and I. At time step (j, i), the computation of its cell depends on both vertical $s_{j,i-1}$ and horizontal hidden states $s_{j-1,i}$ (see Equations (1)–(5)). Similar to the LSTM cell, it maintains some state information in an internal cell state $c_{j,i}$. Besides the input $i_{j,i}$, the forget $f_{j,i}$ and the output $o_{j,i}$ gates that all control information flows, 2DLSTM employs an extra `lambda gate` $\lambda_{j,i}$. As written in Equ. 5, its activation is computed analogously to the other gates. The lambda gate is used to weight the two predecessor cells $c_{j-1,i}$ and $c_{j,i-1}$ before passing them through the forget gate (Equation 6). g and σ are the tanh and the sigmoid functions. Vs, Ws and Us are the weight matrices.

In order to train a 2DLSTM unit, back-propagation through time (BPTT) is performed over two dimensions (Graves, 2008, 2012). Thus, the gradient is passed backwards from the time step (J, I) to $(1, 1)$, the origin. More details, as well as the derivations of the gradients, can be found in (Graves, 2008).

$$i_{j,i} = \sigma\Big(W_1 x_{j,i} + U_1 l s_{j-1,i} + V_1 s_{j,i-1}\Big) \tag{1}$$

$$f_{j,i} = \sigma\Big(W_2 x_{j,i} + U_2 s_{j-1,i} + V_2 s_{j,i-1}\Big) \tag{2}$$

$$o_{j,i} = \sigma\Big(W_3 x_{j,i} + U_3 s_{j-1,i} + V_3 s_{j,i-1}\Big) \tag{3}$$

$$\tilde{c}_{j,i} = g\Big(W_4 x_{j,i} + U_4 s_{j-1,i} + V_4 s_{j,i-1}\Big) \tag{4}$$

$$\lambda_{j,i} = \sigma\Big(W_5 x_{j,i} + U_5 s_{j-1,i} + V_5 s_{j,i-1}\Big) \tag{5}$$

$$\begin{aligned} c_{j,i} = {} & f_{j,i} \circ \big[l\lambda_{j,i} \circ c_{j-1,i} + (1 - \lambda_{j,i}) \circ c_{j,i-1}\big] \\ & + \tilde{c}_{j,i} \circ i_{j,i} \end{aligned} \tag{6}$$

$$s_{j,i} = g\left(c_{j,i}\right) \circ o_{j,i} \tag{7}$$

4 Two-Dimensional Sequence to Sequence Model

We aim to apply a 2DLSTM to map the source and the target sequences into a 2D space as shown in Figure 2. We call this architecture, the two-dimensional sequence to sequence (`2D-seq2seq`) model.

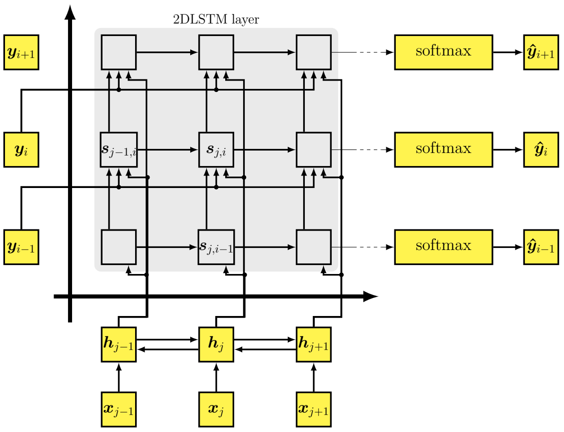

Figure 2: Two-dimensional sequence to sequence model (2D-seq2seq).

Given a source sequence $x_1^J = x_1, \ldots, x_J$ and a target sequence $y_1^I = y_1, \ldots, y_I$, we scan the source sequence from left to right and the target sequence from bottom to top as shown in Figure 2. In the 2D-seq2seq model, one dimension of the 2DLSTM (horizontal-axis in the figure) serves as the encoder and another (vertical axis) plays the role of the decoder. As a pre-step before the 2DLSTM, in order to have the whole source context, a bidirectional LSTM scans the input words once from left to right and once from right to left to compute a sequence of encoder states $h_1^J = h_1, \ldots, h_J$. At time step (j, i), the 2DLSTM receives both encoder state, h_j, and the last target embedding vector, y_{i-1}, as an input. It repeatedly updates the source information, h_1^J, while generating new target word, y_i. The state of the 2DLSTM is computed as follows.

$$s_{j,i} = \psi\Big(W\big[h_j; y_{i-1}\big], U s_{j-1,i}, V s_{j,i-1}\Big) \quad (8)$$

where ψ stands for the 2DLSTM as a function. At each decoder step, once the whole source sequence is processed from 1 to J, the last hidden state of the 2DLSTM, $s_{J,i}$, is used as the context vector. It means, at time step i, $t_i = s_{J,i}$. In order to generate the next target word, y_i, a transformation followed by a softmax operation is applied. Therefore:

$$p_i(y_i = w | y_1^{i-1}, x_1^J) = \frac{\exp(W_o t_{iw})}{\sum_{v=1}^{|V_t|} \exp(W_o t_{iv})} \quad (9)$$

where W_o and $|V_t|$ are the weight matrix and the target vocabulary respectively.

4.1 Training versus Decoding

One practical concern that should be noticed is the difference between the training and the decoding. Since the whole target sequence is known during training, all states of the 2DLSTM can be computed once at the beginning. Slices of it can then be used during the forward and backward training passes. In theory, the complexity of training is $\mathcal{O}(JI)$. But, in practice, the training computation can be optimally parallelized to take linear time (Voigtlaender et al., 2016). During the decoding, only the already generated target words are available. Thus, either all 2DLSTM states have to be recomputed, or it has to be extended by an additional row at every time step i that cause higher complexity.

5 Experiments

We have done the experiments on the WMT 2017 German→English and English→German news tasks consisting of 4.6M training samples collected from the well-known data sets `Europarl-v7`, `News-Commentary-v10` and `Common-Crawl`. We use `newstest2015` as our development set and `newstest2016` and `-2017` as our test sets, which contain 2169, 2999 and 3004 sentences respectively. No synthetic data and no additional features are used. Our goal is to keep the baseline model simple and standard to compare methods rather that advancing the state-of-the-art systems.

After tokenization and true-casing using `Moses` toolkit (Koehn et al., 2007), byte pair encoding (BPE) (Sennrich et al., 2016) is used jointly with 20k merge operations. We remove sentences longer than 50 subwords and batch them together with a batch size of 50. All models are trained from scratch by the Adam optimizer (Kingma and Ba, 2014), dropout of 30% (Srivastava et al., 2014) and the norm of the gradient is clipped with the threshold of 1. The final models are the average of the 4 best checkpoints of a single run based on the perplexity on the development set (Junczys-Dowmunt et al., 2016). Decoding is performed using beam search of size 12, without ensemble of various networks.

We have used our in-house implementation of the NMT system which relies on Theano (Bastien et al., 2012) and Blocks (Merriënboer et al., 2015). Our implementation of 2DLSTM is based on CUDA code adapted from (Voigtlaender et al., 2016; Zeyer et al., 2018), leveraging some speedup.

The models are evaluated using case-sensitive BLEU (Papineni et al., 2002) computed by `mteval-v13a`[1] and case-sensitive TER (Snover et al., 2006) using `tercom`[2]. We also report perplexities on the development set.

Attention Model: the attention based sequence to sequence model (Bahdanau et al., 2014) is selected as our baseline that performs quite well. The model consists of one layer bidirectional encoder and a unidirectional decoder with an additive attention mechanism. All words are projected into a 500-dimensional embedding on both sides. To explore the performance of the models with respect to hidden size, we try LSTMs (Hochreiter and Schmidhuber, 1997) with both 500 and 1000 nodes.

2D-Seq2Seq Model: we apply the same embedding size of that of the attention model. The 2DLSTM, as well as the bidirectional LSTM

---

[1] ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v13a.pl

[2] http://www.cs.umd.edu/ snover/tercom/

	Models	Hidden Size	De→En					En→De				
			devset	**newstest2016**		**newstest2017**		**devset**	**newstest2016**		**newstest2017**	
			PPL	BLEU	TER	BLEU	TER	PPL	BLEU	TER	BLEU	TER
1	attention	n=500	7.3	31.9	48.6	27.5	53.1	7.0	27.0	53.9	22.1	60.5
2	2D-seq2seq		6.5	**32.6**	47.8	**28.2**	52.7	6.1	**27.5**	53.8	**22.4**	60.6
3	+ weighting		6.5	32.3	**47.1**	27.9	**51.7**	6.3	**27.5**	**53.3**	**22.4**	**60.0**
1	attention	n=1000	6.4	33.1	47.5	29.0	**51.9**	6.5	27.4	53.9	22.9	60.2
2	2D-seq2seq		5.7	**33.7**	**46.9**	**29.3**	**51.9**	5.3	**28.9**	**52.6**	**23.2**	**59.5**
3	+ weighting		6.1	32.7	47.1	28.0	**51.9**	5.7	27.8	53.0	22.7	60.0
4	coverage	n=1000	6.3	33.1	47.5	28.7	51.9	5.8	28.6	52.4	23.0	59.4
5	fertility		6.2	33.4	46.9	28.9	51.6	5.8	28.4	52.1	23.2	59.1

Table 1: BLEU [%] and TER [%] on the test sets and perplexity (PPL) on the development set.

layer, are structured using the same number of nodes (500 or 1000). The 2D-seq2seq model is trained with the learning rate of 0.0005 vs. 0.001 for the attention model.

Translation Performance: in the first set of experiments, we compare the 2D-seq2seq model with the attention sequence to sequence model. The results are shown in Table 1 in the rows 1 and 2. As it is seen, for size $n = 500$, the 2D-seq2seq model outperforms the standard attention model on average by 0.7% BLEU and 0.6% TER on De→En, 0.4% BLEU and no improvements in TER on En→De. The model is also superior for larger hidden size ($n = 1000$) on average by 0.5% BLEU and 0.3% TER on De→En, 0.9% BLEU and 1.0% TER on En→De. In both cases, the perplexity of the 2D-seq2seq model is lower compared to that of the attention model.

The 2D-seq2seq topology is analogous to the bidirectional encoder-decoder model without attention. To examine whether the 2DLSTM reduces the need of attention, in the second set of experiments, we equip our model with a weighted sum of 2DLSTM states, t_i, over j positions to dynamically select the most relevant information. In other words:

$$\gamma_{j,i} = \underset{j}{softmax}\Big(v^T \tanh\big(W s_{j,i}\big)\Big) \tag{10}$$

$$t_i = \sum_{j=1}^{J} \gamma_{j,i} s_{j,i} \tag{11}$$

In these equations, $\gamma_{j,i}$ is the normalized weight over source positions, $s_{j,i}$ is the 2DLSTM states and W and v are weight matrices. As the results shown in the Table 1 in the rows 2 and 3, adding an additional weighting layer on top of the 2DLSTM layer does not help in terms of BLEU and rarely helps in TER.

By updating the encoder states across the second dimension with respect to the target history, the 2D-seq2seq model can internally indicate which source words have already been translated and where it should focus next. Therefore, it reduces the risk of over- and under-translation. To examine our assumption, we compare the 2D-seq2seq model with two NMT models where the concepts such as fertility and coverage have been addressed (Tu et al., 2016; Cohn et al., 2016).

Coverage Model: in the coverage model, we feed back the last alignments from the time step $i-1$ to compute the attention weight at time step i. Therefore, in the coverage model, we redefine the attention weight, $\alpha_{i,j}$, as:

$$\alpha_{i,j} = a\big(s_{i-1}, h_j, \alpha_{i-1,j}\big) \tag{12}$$

where a is an attention function followed by the softmax. h_j and s_{i-1} are the the encoder and the previous decoder states respectively. In our experiments, we use additive attention similar to (Bahdanau et al., 2014).

Fertility Model: in the fertility model, we feed back the sum of the alignments over the past decoder steps to indicate how much attention has been given to the source position j up to step i and divide it over the fertility of source word at position j. This term depends on the encoder states and it varies if the word is used in a different context (Tu et al., 2016).

$$\beta_{i,j} = \frac{1}{N \cdot \sigma(v_\phi^\top \cdot h_j)} \sum_{k=1}^{i-1} \alpha_{i,j} \tag{13}$$

$$\alpha_{i,j} = a\big(s_{i-1}, h_j, \beta_{i,j}\big) \tag{14}$$

where N specifies the maximum value for the fertility which set to 2 in our experiments. v_ϕ is a weight vector.

source	HP beschäftigte zum Ende des Geschäftsjahres 2013/14 noch rund 302.000 Mitarbeiter.
reference	At the end of the 2013/14 business year HP still employed around 302,000 staff.
attention	At the end of the financial year, HP employed some 302,000 employees at the end of the financial year of 2013/14.
2D-seq2seq	HP still employs about 302,000 people at the end of the financial year 2013/14.
coverage	HP employed around 302,000 employees at the end of the fiscal year 2013/14.
fertility	HP employed some 302,000 people at the end of the fiscal year 2013/14.

Table 2: An example of over-translation.

As it is seen in Table 1, rows 2, 4 and 5, our proposed model is 0.3% BLEU ahead and 0.3% TER worse compared to the fertility approach and slightly better compared to the coverage one. We note, the fertility and coverage models were trained using embedding size of 620.

We have also qualitatively verified the coverage issue in Table 2 by showing an example from the test set. Without the knowledge of which source words have already been translated, the attention layer is at risk of attending to the same positions multiple times. This could lead to over-translation. Similarly, under-translation could be occur when the attention model rarely focusing at the corresponding source positions. As shown in the example, the 2DLSTM can internally track which source positions have already contributed to the target generation.

Speed: we have also compared the models in terms of speed on a single GPU training. In general, the training and decoding speed of the 2D-seq2seq model is 791 and 0.7 words/s respectively compared to those of standard attention model which is 2944 and 48 words/s. The computation of the added weighting mechanism is negligible in this case. This is still an initial architecture which indicates the necessity of multi-GPU usage. We also expect to speedup the decoding phase by avoiding the unnecessary recomputation of previous 2DLSTM states. In the current implementation, at each target step, we re-compute the 2DLSTM states from time step 0 to $i-1$, while we only need to store the states from the last step $i-1$. This does not influence our results, as it is purely an implementation issue, not algorithm. However, decoding will still be slower than the training. One suggestion for further speedup of training phase is applying truncated BPTT on both directions to reduce the number of updates.

The 2DLSTM can be simply combined with self-attention layers (Vaswani et al., 2017) in the encoder and the decoder for better context representation as well as RNMT+ (Chen et al., 2018) that is composed of standard LSTMs. We believe that 2D-seq2seq model can be potentially applied to the other applications where sequence to sequence modeling is helpful.

6 Conclusion and Future Works

We have introduced a novel 2D sequence to sequence model (2D-seq2seq), a network that applies a 2DLSTM unit to read both the source and the target sentences jointly. Hence, in each decoding step, the network implicitly updates the source representation conditioned on the generated target words so far. The experimental results show that we outperform the attention model on two WMT 2017 translation tasks. We have also shown that our model implicitly handles the coverage issue.

As future work, we aim to develop a bidirectional 2DLSTM and consider stacking up 2DLSTMs for a deeper model. We consider the results promising and try more language pairs and fine-tune the hyperparameters.

Acknowledgements

This work has received funding from the European Research Council (ERC) (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 694537, project "SEQCLAS") and the Deutsche Forschungsgemeinschaft (DFG; grant agreement NE 572/8-1, project "CoreTec"). The GPU computing cluster was supported by DFG (Deutsche Forschungsgemeinschaft) under grant INST 222/1168-1 FUGG. The work reflects only the authors' views and none of the funding agencies is responsible for any use that may be made of the information it contains.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. *CoRR*, abs/1804.09849.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 876–885.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1243–1252.

Alex Graves. 2008. *Supervised sequence labelling with recurrent neural networks*. Ph.D. thesis, Technical University Munich.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer.

Alex Graves and Jürgen Schmidhuber. 2008. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 545–552.

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

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. 2016. The AMU-UEDIN submission to the WMT16 news translation task: Attention-based NMT models as feature functions in phrase-based SMT. In *Proceedings of the First Conference on Machine Translation, WMT 2016, Germany*, pages 319–325.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *CoRR*, abs/1507.01526.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*.

Gundram Leifert, Tobias Strauß, Tobias Grüning, and Roger Labahn. 2016a. Citlab ARGUS for historical handwritten documents. *CoRR*, abs/1605.08412.

Gundram Leifert, Tobias Strauß, Tobias Grüning, Welf Wustlich, and Roger Labahn. 2016b. Cells in multidimensional recurrent neural networks. *The Journal of Machine Learning Research*, 17(1):3313–3349.

Bo Li, Carolina Parada, Gabor Simko, Shuo yiin Chang, and Tara Sainath. 2017. Endpoint detection using grid long short-term memory networks for streaming speech recognition. In *In Proc. Interspeech 2017*.

Jinyu Li, Abdelrahman Mohamed, Geoffrey Zweig, and Yifan Gong. 2016. Exploring multidimensional lstms for large vocabulary ASR. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 4940–4944.

Bart Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: Frameworks for deep learning.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Tara N. Sainath and Bo Li. 2016. Modeling time-frequency patterns with LSTM vs. convolutional architectures for LVCSR tasks. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 813–817.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Paul Voigtlaender, Patrick Doetsch, and Hermann Ney. 2016. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, pages 228–233.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Albert Zeyer, Tamer Alkhouli, and Hermann Ney. 2018. RETURNN as a generic flexible neural toolkit with application to translation and speech recognition. In *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations*, pages 128–133.

End-to-End Non-Autoregressive Neural Machine Translation with Connectionist Temporal Classification

Jindřich Libovický and **Jindřich Helcl**
Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, 118 00 Prague, Czech Republic
{libovicky, helcl}@ufal.mff.cuni.cz

Abstract

Autoregressive decoding is the only part of sequence-to-sequence models that prevents them from massive parallelization at inference time. Non-autoregressive models enable the decoder to generate all output symbols independently in parallel. We present a novel non-autoregressive architecture based on connectionist temporal classification and evaluate it on the task of neural machine translation. Unlike other non-autoregressive methods which operate in several steps, our model can be trained end-to-end. We conduct experiments on the WMT English-Romanian and English-German datasets. Our models achieve a significant speedup over the autoregressive models, keeping the translation quality comparable to other non-autoregressive models.

1 Introduction

Parallelization is the key ingredient for making deep learning models computationally tractable. While the advantages of parallelization are exploited on many levels during training and inference, autoregressive decoders require sequential execution.

Training and inference algorithms in sequence-to-sequence tasks with recurrent neural networks (RNNs) such as neural machine translation (NMT) have linear time complexity w.r.t. the target sequence length, even when parallelized (Sutskever et al., 2014; Bahdanau et al., 2014).

Recent approaches such as convolutional sequence-to-sequence learning (Gehring et al., 2017) or self-attentive networks a.k.a. the Transformer (Vaswani et al., 2017) replace RNNs with parallelizable components in order to reduce the time complexity of the training. In these models, the decoding is still sequential, because the probability of emitting a symbol is conditioned on the previously decoded symbols.

In non-autoregressive decoders, the inference algorithm can be parallelized because the decoder does not depend on its previous outputs. The apparent advantage of this approach is the near-constant time complexity achieved by the parallelization. On the other hand, the drawback is that the model needs to explicitly determine the target sentence length and reorder the state sequence before it starts generating the output. In the current research contributions on this topic, these parts are trained separately and the inference is done in several steps.

In this paper, we propose an end-to-end non-autoregressive model for NMT using Connectionist Temporal Classification (CTC; Graves et al. 2006). The proposed technique achieves promising results on translation between English-Romanian and English-German on the WMT News task datasets.

The paper is organized as follows. In Section 2, we summarize the related work on non-autoregressive NMT. Section 3 describes the architecture of our proposed model. Section 4 presents details of the conducted experiments. The results are discussed in Section 5. We conclude and present ideas for future work in Section 6.

2 Non-Autoregressive NMT

In this section, we describe two methods for non-autoregressive decoding in NMT. Both of them are based on the Transformer architecture (Vaswani et al., 2017), with the encoder part unchanged.

Gu et al. (2017) use a latent fertility model to copy the sequence of source embeddings which is then used for the target sentence generation. The fertility (i.e. the number of target words for each source word) is estimated using a softmax on the encoder states. In the decoder, the input embeddings are repeated based on their fertility.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3016–3021
Brussels, Belgium, October 31 - November 4, 2018.

The decoder has the same architecture as the encoder plus the encoder attention. The best results were achieved by sampling fertilities from the model and then rescoring the output sentences using an autoregressive model. The reported inference speed of this method is 2–15 times faster than of a comparable autoregressive model, depending on the number of fertility samples.

Lee et al. (2018) propose an architecture with two decoders. The first decoder generates a candidate translation from a source sentence padded to an estimated target length. The explicit length estimate is done with a softmax over possible sentence lengths (up to a fixed maximum). The output of the first decoder is then fed as an input to the second decoder. The second decoder is used as a denoising auto-encoder and can be applied iteratively. Both decoders have the same architecture as in Gu et al. (2017). They achieved a speedup of 16 times over the autoregressive model with a single denoising iteration. They report the best result in terms of BLEU (Papineni et al., 2002) after 20 iterations with almost no inference speedup compared to their autoregressive baseline.

3 Proposed Architecture

Similar to the previous work (Gu et al., 2017; Lee et al., 2018), our models are based on the Transformer architecture as described by Vaswani et al. (2017), keeping the encoder part unchanged. Figure 1 illustrates our method and highlights the differences from the Transformer model.

In order to generate output words in parallel, we formulate the translation as a sequence labeling problem. Neural architectures used for encoding input in NLP tasks usually generate sequences of hidden states of the same or shorter length as the input sequence. For this reason, we cannot apply the sequence labeling directly over the states because the target sentence might be longer than the source sentence.

To enable the labeler to generate sentences that are longer than the source sentence, we project the encoder output states $\mathbf{h}$ into a k-times longer sequence $\mathbf{s}$, such that:

$$s_{ci+b} = \left(W_{\text{spl}} h_c + b_{\text{spl}}\right)_{bd:(b+1)d} \qquad (1)$$

for $b = 0 \dots k-1$, and $c = 0 \dots T_x$ where d is the Transformer model dimension, T_x is the length of the source sentence, and $W_{\text{spl}} \in \mathbb{R}^{d \times kd}$ and $b_{\text{spl}} \in \mathbb{R}^{kd}$ are trainable projection parameters. In other words, after a linear projection, each state is sliced to k vectors, creating a sequence of length kT_x.

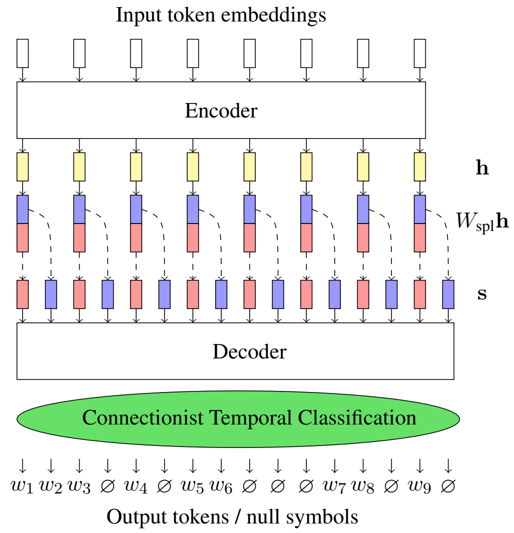

Figure 1: Scheme of the proposed architecture. The part between the encoder and the decoder is expressed by Equation 1.

In the next step, we process the sequence $\mathbf{s}$ with a decoder. Unlike the Transformer architecture, our decoder does not use the temporal mask in the self-attention step.

Finally, the decoder states are labeled either with an output token or a null symbol. The number of combinations of the possible positions of the null symbols in the output sequence given reference sequence length T_y is $\binom{kT_x}{T_y}$. Because there is no prior alignment between the input and output symbols, we consider all output sequences that yield the correct output in the loss function. Because summing the exponential number of combinations directly is not tractable, we we use the CTC loss (Graves et al., 2006) which employs dynamic programming to compute the negative log-likelihood of the output sequence, summed over all the combinations.

The loss can be computed using a linear algorithm similar to training Hidden Markov Models (Rabiner, 1989). The algorithm computes and stores partial log-probabilities sums for all prefixes and suffixes of the output symbol sequence using dynamic programming. The table of precomputed log-probablities allows us to compute the probability of being a part of a correct output sequence by combining the log-probabilities of its prefix and suffix.

An appealing property of training using the

CTC loss is that the models support left-to-right beam search decoding by recombining prefixes that yield the same output. Unlike the greedy decoding this can no longer be done in parallel. However, the linear computation is in theory still faster than autoregressive decoding.

4 Experiments

We experiment with three variants of this architecture. All of them have the same total number of layers. First, the *deep encoder* uses a stack of self-attentive layers only. We apply the state splitting and the labeler on the output of the last encoder layer. In contrast to Figure 1, this variant omits the decoder part. Second, the *encoder-decoder* consists of two stacks of self-attentive layers – encoder and decoder. The outputs of the encoder are transformed using Equation 1 and processed by the decoder. In each layer, the decoder part attends to the encoder output. Third, we extend the encoder-decoder variant with positional encoding (Vaswani et al., 2017). The positional encoding vectors are added to the decoder input $\mathbf{s}$.

In all the experiments, we used the same hyperparameters. We set the model dimension to 512 and the feed-forward layer dimension to 4096. We use multi-head attention with 16 heads. In the deep encoder setup, we use 12 layers in the encoder, in the encoder-decoder setup, we use 6 layers for the encoder and 6 layers for the decoder. We set the split factor k to 3, so the encoder states are projected to vectors of 1536 units.

We conduct our experiments on English-Romanian and English-German translation. These language pairs were selected by the authors of the previous work because the training datasets for these language pairs are of considerably different sizes. We follow these choices in order to present comparable results.

For English-Romanian experiments, we used the WMT16 (Bojar et al., 2016) news dataset. The training data consists of 613k sentence pairs, validation 2k and test 2k. We used a shared vocabulary of 38k wordpieces (Wu et al., 2016; Johnson et al., 2017).

The English-German dataset consists of 4.6M training sentence pairs from WMT competitions. As a validation set, we used the test set from WMT13 (Bojar et al., 2013), which contains 3k sentence pairs. To enable comparison to other non-autoregressive approaches, we evaluate our models on the test sets from WMT14 (Bojar et al., 2014) with 3k sentence pairs and WMT15 (Bojar et al., 2015) with 2.1k sentence pairs. As in the previous case, we used shared vocabulary for both languages which contained 41k wordpieces.

The experiments were conducted using Neural Monkey[1] (Helcl and Libovický, 2017). We evaluate the models using BLEU score (Papineni et al., 2002) as implemented in SacreBLEU,[2] originally a part of the Sockeye toolkit (Hieber et al., 2017).

5 Results

Quantitative results are tabulated in Table 1. In general, our models achieve a similar performance to other non-autoregressive models. In case of English-German, our results in both directions are comparable on the WMT 14 test set and slightly better on the WMT 15 test set. This might be given by the fact that our autoregressive baseline performs better for this language pair than for English-Romanian.

The encoder-decoder setup outperforms the deep encoder setup. Including positional encoding seems beneficial when translating into German. Weight averaging from the 5 models with the highest validation score during the training improves the performance consistently.

We performed a manual evaluation on 100 randomly sampled sentences from the English-German test sets in both directions. The results of the analysis are summarized in Table 2.

Non-autoregressive translations of sentences that had errors in the autoregressive translation were often incomprehensible. In general, less than a quarter of the sentences was completely correct and over two thirds (one half in the de→en direction) were comprehensible. The most frequent errors include omitting verbs at the end of German sentences and corruption of named entities and infrequent words that are represented by more wordpieces. Most of these errors can be attributed to insufficient language-modeling capabilities of the model. The results suggest that integrating an external language model into an efficient beam search implementation could boost the translation quality while preserving the speedup over the auto-regressive models.

We also evaluated the translations using sentence-level BLEU score (Chen and Cherry,

---

[1] `https://github.com/ufal/neuralmonkey`

[2] `https://github.com/mjpost/sacreBLEU`

	WMT 16		WMT 14		WMT 15	
	en-ro	ro-en	en-de	de-en	en-de	de-en
autoregressive $b = 1$	31.93	31.55	22.71	26.39	23.40	26.49
autoregressive $b = 4$	32.40	32.06	23.45	27.02	24.12	27.05
Gu et al. (2017) greedy	27.29	29.06	17.69	21.47	—	—
Gu et al. (2017) NPD w/ 100 samples	29.79	31.44	19.17	23.20	—	—
Lee et al. (2018) 1 iteration	24.45	23.73	—	—	12.65	14.48
Lee et al. (2018) best result	29.49	30.41	—	—	19.13	21.69
our autoregressive $b = 1$	21.19	29.64	22.94	28.58	25.12	28.89
deep encoder	17.33	22.85	12.21	12.53	13.14	18.34
+ weight averaging	18.47	**24.68**	14.65	16.72	16.74	18.47
+ beam search	18.70	25.28	15.19	17.58	17.59	18.70
encoder-decoder	18.51	22.37	13.29	17.98	16.01	19.55
+ weight averaging	**19.54**	24.67	16.56	**18.64**	19.46	**21.74**
+ beam search	19.81	25.21	17.09	18.80	20.59	22.55
encoder-decoder w/ pos. encoding	18.13	22.75	12.51	11.35	15.35	19.30
+ weight averaging	19.31	24.21	**17.37**	18.07	**20.30**	19.64
+ beam search	19.93	24.71	17.68	19.80	20.67	20.43

Table 1: Quantitative results in terms of BLEU score of the proposed methods compared to other non-autoregressive models. Note that our method uses only a single pass through the network and should be compared with greedy decoding by Gu et al. (2017) and 1 model iteration by Lee et al. (2018).

	en → de		de → en	
	AR	NAR	AR	NAR
Correct	65	23	67	13
Comprehensible	93	71	92	51
Too short	1	16	0	36
Missing verb	4	35	0	8
Corrupt. named entity	1	27	8	21
Corrupt. other words	1	20	0	46

Table 2: Results of manual evaluation of the autoregressive (AR) and non-autoregressive (NAR) models (in percents).

2014) and measure the Pearson correlation with the length of the source sentence and the number of null symbols generated in the output. With a growing sentence length, the scores degrade more in the non-autoregressive model ($r = -0.42$) than in its autoregressive counterpart ($r = -0.39$). The relation between sentence-level BLEU and the source length is plotted in Figure 2. The sentence-level score is mildly correlated with the number of null symbols in the non-autoregressive output ($r = 0.15$). This suggests that increasing the splitting factor k in Equation 1 might improve the model performance. However, it also reduces the efficiency in terms of GPU memory usage.

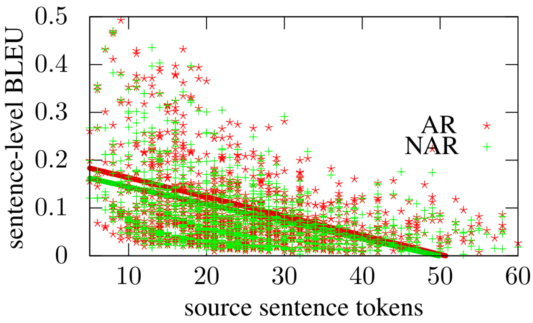

Figure 2: Comparison of the sentence-level BLEU of our English-to-German autoregresssive (AR) and non-autoregressive (NAR) models given the length of the source sentence.

Figure 3 shows the comparison of the decoding time by autoregressive and non-autoregressive models. The average times of decoding a single sentence are shown in Table 3. We suspect that the small difference between CPU and GPU times in the non-autoregressive setup is caused by the CPU-only implementation of the CTC decoder in TensorFlow (Abadi et al., 2015).

6 Conclusions

In this work, we presented a novel method for training a non-autoregressive model end-to-end using connectionist temporal classification. We evaluated the proposed method on neural machine

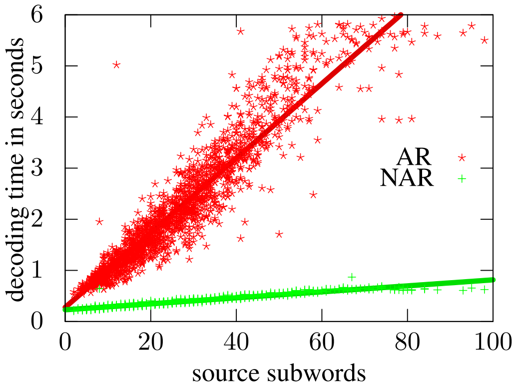

Figure 3: Comparison of CPU decoding time by our autoregressive (AR) and non-autoregressive (NAR) models based on the source sentence length.

	CPU	GPU
AR, $b = 1$	2247 ms	1200 ms
NAR	386 ms	350 ms

Table 3: Average per sentence decoding time for en-de translation.

translation in two language pairs and compared the results to the previous work.

In general, the results match the translation quality of equivalent variants of the models presented in the previous work. The BLEU score is usually around 80–90% of the score of the autoregressive baselines. We measured a 4-times speedup compared to our autoregressive baseline, which is a smaller gain than reported by the authors of the previous work. We suspect this might be due to a larger overhead with data loading and processing in Neural Monkey compared to Tensor2Tensor (Vaswani et al., 2018) used by others.

As a future work, we can try to improve the performance of the model by iterative denoising as done by Lee et al. (2018) while keeping the non-autoregressive nature of the decoder.

Another direction of improving the model might be efficient implementation of beam search which can contain rescoring using an external language model as often done in speech recognition (Graves et al., 2013). The non-autoregressive model would play a role a of the translation model in the traditional statistical MT problem decomposition.

Acknowledgments

This research has been funded by the Czech Science Foundation grant no. P103/12/G084, Charles University grant no. 976518 and SVV project no. 260 453.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation (WMT16). In *Proceedings of the First Conference on Machine Translation (WMT). Volume 2: Shared Task Papers*, volume 2, pages 131–198, Stroudsburg, PA, USA. Association for Computational Linguistics, Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia

Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1243–1252.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *CoRR*, abs/1711.02281.

Jindřich Helcl and Jindřich Libovický. 2017. Neural Monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, (107):5–17.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1712.05690.*

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viegas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *CoRR*, abs/1802.06901.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Lawrence R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Prediction Improves Simultaneous Neural Machine Translation

Ashkan Alinejad [1], Maryam Siahbani[2], Anoop Sarkar[1]
[1]Simon Fraser University, Burnaby, BC, Canada
{aalineja,anoop}@sfu.ca
[2]University of the Fraser Valley, Abbotsford, BC, Canada
maryam.siahbani@ufv.ca

Abstract

Simultaneous speech translation aims to maintain translation quality while minimizing the delay between reading input and incrementally producing the output. We propose a new general-purpose prediction action which predicts future words in the input to improve quality and minimize delay in simultaneous translation. We train this agent using reinforcement learning with a novel reward function. Our agent with prediction has better translation quality and less delay compared to an agent-based simultaneous translation system without prediction.

1 Introduction

One of the next significant challenges in machine translation research is to make translation ubiquitous using real-time translation. Simultaneous machine translation aims to address this issue by interleaving reading the input with writing the output translation. Current Simultaneous Neural Machine Translation (SNMT) systems (Satija and Pineau, 2016; Cho and Esipova, 2016; Gu et al., 2017) use an AGENT to control an incremental encoder-decoder (or sequence to sequence) NMT model. Each READ adds more information to the encoder RNN, and each WRITE produces more output using the decoder RNN. In this paper, we propose adding a new action to the AGENT: a PREDICT action that predicts what words might appear in the input stream. Prediction was previously proposed in simultaneous statistical machine translation (Grissom II et al., 2014) but has not been studied in the context of Neural Machine Translation (NMT). In SNMT systems, prediction of future words augments the encoder-decoder model with possible future contexts to produce output translations earlier (minimize delay) and/or produce better output translations (improve translation quality). Our experiments show that prediction improves SNMT in both these measures.

2 Simultaneous Translation Framework

An agent-based framework whose actions decide whether to translate or wait for more input is a natural way to extend neural MT to simultaneous neural MT and has been explored in (Satija and Pineau, 2016; Gu et al., 2017) which contains two main components: The ENVIRONMENT which receives the input words $X = \{x_1, \dots, x_N\}$ from the source language and incrementally generates translated words $W = \{w_1, \dots, w_M\}$ in the target language; And the AGENT which decides an action for each time step, a_t. The AGENT generates an action sequence $A = \{a_1, \dots, a_T\}$ to control the ENVIRONMENT.

Previous models only include two actions: READ and WRITE. We extend the model by adding the third action called PREDICT. Action READ is simply sending a new word to the ENVIRONMENT and generating a candidate word in the target language. In action WRITE, the AGENT takes current candidate word and sends it to the output. For PREDICT, the AGENT predicts the next word in the input and treats it like a READ action. The following section explains how the ENVIRONMENT deals with different actions.

2.1 ENVIRONMENT

The ENVIRONMENT is an attention-based Encoder-Decoder MT system (Bahdanau et al., 2014) which is adopted to simultaneous translation task. The Encoder receives the embedded representation of the input words (including predicted ones) and converts them into context vectors $H_n^\rho = \{h_1, \dots, h_{n+\rho}\}$ using a gated RNN (GRU) where n is the number of input words so far and ρ is the number of predicted words since the last READ. Whenever the AGENT decides to READ, ρ will be set to 0, and $h_n = f_{\text{ENC}}(h_{n-1}, x_n)$, where x_n is the next input words ($n \le N$). But if the action is PREDICT, $\rho > 0$, the AGENT predicts a new word x'_ρ and the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3022–3027
Brussels, Belgium, October 31 - November 4, 2018.

context vector $h_{n+\rho} = f_{\text{ENC}}(h_{n+\rho-1}, x'_\rho)$ will be added to $H_n^\rho = \{h_1, \ldots, h_{n+\rho}\}$.

At each time step t, the decoder uses the current context vectors (H_n^ρ) to generate the next candidate output (y_t):

$$\begin{aligned} c_t &= a_{\text{ATT}}(z_{m-1}, H_n^\rho) \\ s_t &= f_{\text{DEC}}(z_{m-1}, w_{m-1}, c_t) \\ p(y_t|y_{<t}, H_n^\rho) &= g(w_{m-1}, z_m, c_t) \\ y_t &= \arg\max_y p(y|y_{<t}, H_n^\rho) \end{aligned}$$

where w_{m-1} is the previous output word, and a_{ATT} is an attention model (Bahdanau et al., 2014), f and g are nonlinear functions, and c_t is the current context vector.

If the action a_t is either READ or PREDICT the current candidate y_t will be ignored (wait for better predictions). But in the case of WRITE, the candidate y_t is produced as the next output word w_m and then the decoder state will be updated ($w_m \leftarrow y_t$, $z_m \leftarrow s_t$).

Note that as soon as the AGENT decides to READ, all the hidden vectors generated by PREDICT actions will be discarded ($H_n^\rho = H_n^0 = \{h_1, \ldots, h_n\}$).

Figure 1 shows an example of how a sentence can be translated using our modified translation framework [1].

2.2 AGENT

The AGENT is a separate component which examines the ENVIRONMENT at each time step and decides on the actions that lead to better translation quality and lower delay. The agent in the *greedy decoding* framework (Gu et al., 2017) was trained using reinforcement learning with the policy gradient algorithm (Williams, 1992), which observes the current state of the ENVIRONMENT at time step t as o_t where $o_t = [c_t;\ s_t;\ w_m]$. A RNN with one hidden layer passed through a softmax function generates the probability distribution over the actions a_t at each step. Therefore, policy π_θ will be computed as:

$$\begin{aligned} u_t &= f_\theta(u_{t-1}, o_t) \\ \pi_\theta(a_t|a < t, o \leq t) &\propto g_\theta(u_t) \end{aligned}$$

Where u_t is the hidden state of the AGENT's RNN.

3 Training the AGENT with Prediction

In order to speed-up the training process, we have restricted AGENT's options by removing redundant operations. As illustrated in Figure 2, after a series of WRITE, the AGENT cannot choose to PREDICT, and after a sequence of PREDICTs, READ is not an option.

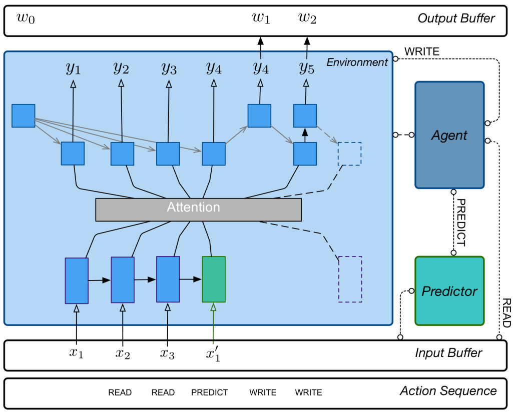

Figure 1: A schematic of our model. The ENVIRONMENT starts with reading the 'Start of Sentence' symbol as x_1 and generating y_1 from the decoder. Based on the output of the network at each time step, the AGENT decides whether to READ, WRITE or PREDICT for the following time steps.

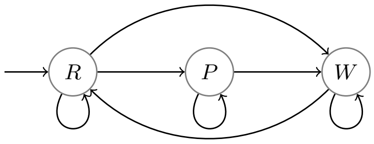

Figure 2: Action transition graph. R, P, and W stands for READ, PREDICT and WRITE actions respectively.

Reward Function: The total reward at any time step is calculated as the cumulative sum of rewards for actions at each preceding step. All the evaluation metrics have been modified to be computed for every time step.

Quality: We use a modified smoothed version of BLEU score (Chen and Cherry, 2014) multiplied by Brevity Penalty (Lin and Och, 2004) for evaluating the impact of each action on translation quality. At each point in time, the reward for translation quality is:

$$r_t^Q = \begin{cases} \Delta\text{BLEU}(t) & t < T \\ \text{BLEU}(W, W^*) & t = T \end{cases}$$

The $\Delta\text{BLEU}(t)$ is the difference between BLEU score of the translated sentence at the previous time step and the current time step; $\Delta\text{BLEU}(t) = \text{BLEU}(W^t, W^*) - \text{BLEU}(W^{t-1}, W^*)$; where W^t is the prefix of the translated sentence at time t.

Delay: The Delay reward is used to motivate the AGENT to minimize delay. We use Average Proportion (AP) (Cho and Esipova, 2016) for this purpose, which is the average number of source words

---

[1] The pseudo-code is available in supplementary material (Algorithm 1).

needed when translating each word. Given the source words X and translated words W, AP can be computed as:

$$d(X, W) = \frac{1}{|X||W|} \sum_t s(t)$$

$$r_t^D = \begin{cases} 0 & t < T \\ d(X, W) & t = T \end{cases}$$

Where $s(t)$ denotes the number of source words the WRITE action uses at time step t (for any other actions, $s(t)$ would be zero). The delay reward is smoothed using a *Target Delay* which is a scalar constant denoted by d^* (Gu et al., 2017):

$$r_t^D = \lfloor d_t - d^* \rfloor$$

Prediction Rewards for Quality and Delay alone do not motivate the AGENT to choose prediction and in preliminary experiments, after a number of steps, the number of prediction actions became zero. We address this problem by defining Prediction Quality (PQ) which rewards the AGENT for changes in BLEU score after each prediction action. By initializing $r_0^p = 0$, the prediction reward can be written as:

$$r_t^p = \begin{cases} \Delta\text{BLEU}(t) & a_t = \text{W}, a_{t-1} = \text{P} \\ r_{t-1}^p & a_t = \text{W}, a_{t-1} \neq \text{P} \\ 0 & \text{otherwise} \end{cases}$$

Final Reward The final reward function is calculated as the combination of quality, delay, and prediction rewards:

$$r_t = \alpha\, r_t^Q + \beta\, r_t^D + \lambda\, r_t^p \qquad (1)$$

The trade-off between better translation quality and minimal delay is achieved by modifying the parameters α, β, and λ.

Reinforcement Learning is used to train the AGENT using a policy gradient algorithm (Gu et al., 2017; Williams, 1992) which searches for the maximum in $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} r_t\right]$ using the gradient: $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)\right]$ where $Q = \sum_{t=1}^{T} r_t$. The gradient for a sentence is the cumulative sum of gradients at each time step. We pre-train the ENVIRONMENT on full sentences using log-loss $\log p(\mathbf{y}|\mathbf{x})$.

4 Experiments

We train and evaluate our model on English-German (EN-DE) in both directions. We use WMT 2015 for training and Newstest 2013 for validation and testing. All sentences have been tokenized and the words are segmented using byte pair encoding (BPE) (Sennrich et al., 2016).

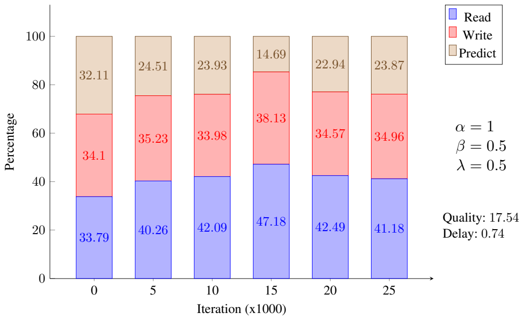

Figure 3: Action distribution for English to German translation in the first 25000 iterations. The numbers in each bar are the average action percentage over the previous 5000 iterations.

Model Configuration For a fair comparison, we follow the settings that worked the best for the greedy decoding model in (Gu et al., 2017) and set the target delay d^* for the AGENT to 0.7. The ENVIRONMENT consists of two unidirectional layers with 1028 GRU units for encoder and decoder. We train the network using AdaDelta optimizer, a batch of size 32 and a fixed learning rate of 0.0001 without decay. We use softmax policy via recurrent networks with 512 GRU units and a softmax function for the AGENT and train it using Adam optimizer (Kingma and Ba, 2014). The batch size for the AGENT is 10, and the learning rate is 2e-6. The word predictor is a two layer RNN Language model which consists of two layers of 1024 units, followed by a softmax layer. The batch size is 64 with a learning rate of 2e-5. The predictor has been trained on the WMT'16 dataset and tested on Newstest'16 corpora for both languages. The perplexity of our language model is reported in Table 1. We set $\alpha = 1$, $\beta = 0.5$ and $\lambda = 0.5$. We tried different settings for these hyperparameters during training and picked values that gave us the best Quality and Delay on the training data.

Results and Analysis Figure 4 shows that as the sentence length increases, prediction helps translation quality due to complex reordering and multiclausal sentences; However, for shorter samples where the structure of the sentences are simpler, the prediction action cannot improve translation quality. Table 2 compares our model with the Greedy Decoding (GD) model in terms of translation quality and latency. It shows that the prediction mechanism outperforms the GD model in terms of BLEU and average proportion (AP).

The delay evaluation measure (AP) counts about the same number of READs and WRITEs. It

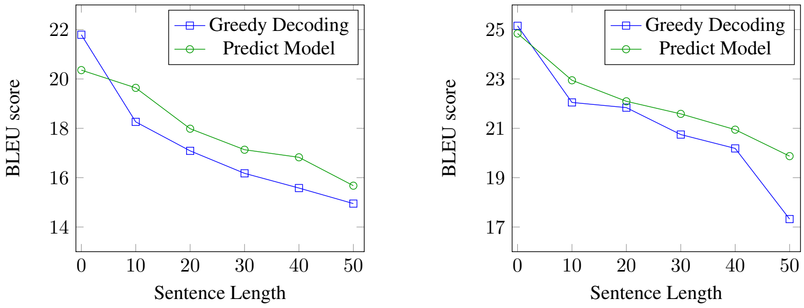

(a) EN→DE (b) DE→EN

Figure 4: Comparing translation quality between prediction model and greedy decoding method for various sentence lengths.

	Perplexity	
	Test	Train
English	17.5917	8.1350
German	19.8532	11.1808

Table 1: Performance of our predictor for both English and German languages.

	EN→DE			DE→EN		
	BLEU	AP	μ	BLEU	AP	μ
GD	16.75	0.79	5.6	21.43	0.77	6
PM	**17.54**	**0.74**	**4.7**	**21.83**	**0.70**	**5.2**

Table 2: Comparison of translation quality (BLEU), Average Proportion (AP), and average segment length (μ) for Greedy Decoding (GD) model with Prediction Mechanism (PM) model.

does not account for less delay as longer sentences are produced. A better measure than AP might be needed to emphasize delay differences. Therefore we also report the average segment length (μ), which is computed as the average number of consecutive READs in each sentence. In both EN→DE and DE→EN experiments, our model constantly decreases the segment length by around 1 word which results in less latency.

In order to evaluate the effectiveness of our proposed reward function for PREDICT action, we have explored various values for its hyper-parameters (α, β, and λ). Our empirical results show that the best trade-off between quality and delay is achieved when around 20 percent of the actions are PREDICT for both EN→DE and DE→EN translation tasks (Figure 3). When there is no reward for PREDICT action ($\lambda = 0$), the AGENT prefers other actions, and the number of PREDICT actions turns into zero immediately after training the AGENT. If the reward for prediction is valued too highly, the ENVIRONMENT depends more on predicted words and the translation quality decreases[2].

5 Related work

Early work in SNMT was done in speech, where the incoming signals were segmented based on acoustic or statistical cues (Bangalore et al., 2012; Fügen et al., 2007). (Sridhar et al., 2013; Matusov et al., 2007; Yarmohammadi et al., 2013; Siahbani et al., 2014) use a separate segmentation step and incrementally translate each segment using a standard phrase-based MT system. (Matsubara et al., 2000) applied pattern matching to predict target-side verbs in Japanese to English translation. (Grissom II et al., 2014) used reinforcement learning to predict the next word and the sentence-final verb in a statistical MT model. These models reduce the delay but are not trained end-to-end like our agent-based SNMT system. (Cho and Esipova, 2016) proposed a non-trainable heuristic agent which is not able to trade-off quality with delay. It always prefers to read more words from the input and this approach does not work well in practice. (Satija and Pineau, 2016) introduced a trainable agent which they trained using Deep Q networks (Mnih et al., 2015). We modified the SNMT trainable agent in (Gu et al., 2017) and added a new non-trivial PREDICT action to the agent. We compare to their model and show better results in delay and quality.

6 Conclusion

We introduce a new prediction action in a trainable agent for simultaneous neural machine translation. With prediction, the agent can be informed about future time steps in the input stream. Compared to a very strong baseline our results show that prediction can lower delay and improve the

---

[2] See Figure 6 in supplementary materials for more numerical results.

translation quality, especially for longer sentences and translating from an SOV (subject-object-verb) language (DE) to an SVO language (EN).

Acknowledgments

We would like to thank the anonymous reviewers for their helpful remarks. The research was also partially supported by the Natural Sciences and Engineering Research Council of Canada grants NSERC RGPIN-2018-06437 and RGPAS-2018-522574 and a Department of National Defence (DND) and NSERC grant DGDND-2018-00025 to the third author.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 437–445, Stroudsburg, PA, USA. Association for Computational Linguistics.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *WMT*, pages 362–367.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *CoRR*, abs/1606.02012.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine Translation*, 21(4):209–252.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Empirical Methods in Natural Language Processing*.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Valencia, Spain.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shigeki Matsubara, Keiichi Iwashima, Nobuo Kawaguchi, Katsuhiko Toyama, and Yasuyoshi Inagaki. 2000. Simultaneous japanese-english interpretation based on early prediction of english verb. *Proceedings of the 4th Symposium on Natural Languauge Processing(SNLP-2000)*, pages 268–273.

Evgeny Matusov, Dustin Hillard, Mathew Magimai-Doss, Dilek Hakkani-Tür, Mari Ostendorf, and Hermann Ney. 2007. Improving speech translation with automatic boundary prediction. In *Eighth Annual Conference of the International Speech Communication Association*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –.

Harsh Satija and Joelle Pineau. 2016. Simultaneous machine translation using deep reinforcement learning. *Abstraction in Reinforcement Learning Workshop, ICML 2016*, (33).

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Maryam Siahbani, Ramtin Mehdizadeh Seraj, Baskaran Sankaran, and Anoop Sarkar. 2014. Incremental translation using hierarchichal phrase-based translation system. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 71–76. IEEE.

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pages 229–256.

Mahsa Yarmohammadi, Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. 2013. Incremental segmentation and decoding strategies for simultaneous translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1032–1036.

Training Deeper Neural Machine Translation Models with Transparent Attention

Ankur Bapna * **Mia Xu Chen** * **Orhan Firat** * **Yuan Cao** * **Yonghui Wu**
ankurbpn,miachen,orhanf,yuancao@google.com
Google AI

Abstract

While current state-of-the-art NMT models, such as RNN seq2seq and Transformers, possess a large number of parameters, they are still shallow in comparison to convolutional models used for both text and vision applications. In this work we attempt to train significantly (2-3x) deeper Transformer and Bi-RNN encoders for machine translation. We propose a simple modification to the attention mechanism that eases the optimization of deeper models, and results in consistent gains of 0.7-1.1 BLEU on the benchmark WMT'14 English-German and WMT'15 Czech-English tasks for both architectures.

1 Introduction

The past few years have seen significant advances in the quality of machine translation systems, owing to the advent of neural sequence to sequence models. While current state of the art models come in different flavours, including Transformers (Vaswani et al., 2017), convolutional seq2seq models (Gehring et al., 2017) and LSTMs (Chen et al., 2018), all of these models follow the seq2seq with attention (Bahdanau et al., 2015) paradigm.

While revolutionary new architectures have contributed significantly to these quality improvements, the importance of larger model capacities cannot be downplayed. The first major improvement in NMT quality since the switch to neural models, amongst other factors, was brought about by a huge scale up in model capacity (Zhou et al., 2016; Wu et al., 2016). While there are multiple approaches to increase capacity, deeper models have been shown to extract more expressive features (Mhaskar et al., 2016; Telgarsky, 2016; Eldan and Shamir, 2015), and have resulted in significant gains for vision tasks over the past few years (He et al., 2015; Srivastava et al., 2015).

Despite this being an obvious avenue for improvement, research in deeper models is often restricted by computational constraints. Additionally, deep models are often plagued by trainability concerns like vanishing or exploding gradients (Bengio et al., 1994). These issues have been studied in the context of capturing long range dependencies in recurrent architectures (Pascanu et al., 2012; Hochreiter et al., 2001), but resolving these deficiencies in Transformers or LSTM seq2seq models deeper than 8 layers is unfortunately under-explored (Wang et al., 2017; Barone et al., 2017; Devlin, 2017).

In this study we take the first step towards training extremely deep models for translation, by training deep encoders for Transformer and LSTM based models. As we increase the encoder depth the vanilla Transformer models completely fail to train. We also observe sub-optimal performance for LSTM models, which we believe is associated with trainability issues. To ease optimization we propose an enhancement to the attention mechanism, which allows us to train deeper models and results in consistent gains on the WMT'14 En→De and WMT'15 Cs→En tasks.

2 Transparent Attention

While the effect of attention on the forward pass is exalted with visualizations and linguistic interpretations, its influence on the gradient flow is often forgotten. Consider the original seq2seq model *without* attention (Sutskever et al., 2014). To propagate the error signal from the last layer of the decoder to the first layer of the encoder, it has to pass through multiple time-steps in the decoder, survive the encoder-decoder bottleneck, and pass through multiple time-steps in the encoder, before reaching the parameter to be updated. There is some loss of information at every step, especially in

---

* Equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3028–3033
Brussels, Belgium, October 31 - November 4, 2018.

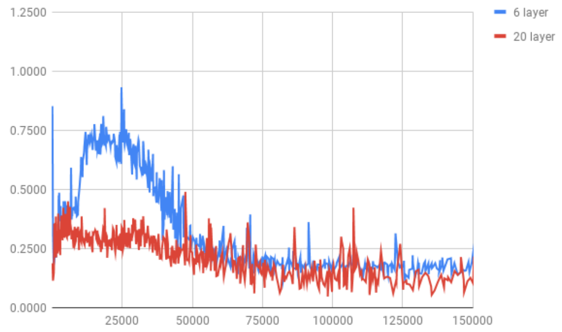

Figure 1: Grad-norm ratio (r_t) vs training step (t) comparison for a 6 layer (blue) and 20 layer (red) Transformer trained on WMT 14 En→De.

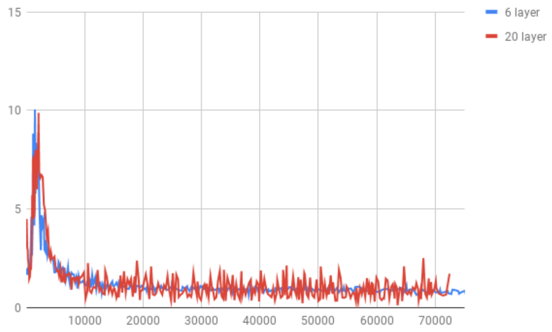

Figure 2: Grad-norm ratio (r_t) vs training step (t) comparison for a 6 layer (blue) and 20 layer (red) RNMT+ model trained on WMT 14 En→De.

the early stages of training. Attention (Bahdanau et al., 2015) creates a direct path from the decoder to the topmost layer of the encoder, ensuring its efficient dispersal over time. This increase in inter-connectivity significantly shortens the credit-assignment path (Britz et al., 2017), making the network less susceptible to optimization pathologies like vanishing gradients.

For deeper networks the error signal also needs to traverse along the depth of the encoder. We propose an extension to the attention mechanism that behaves akin to creating weighted residual connections along the encoder depth, allowing the dispersal of error signal simultaneously over encoder depth and time. Using trainable weights, this 'transparent' attention allows the model the flexibility to adjust the gradient flow to different layers in the encoder depending on its training phase.

2.1 Experimental Setup

We train our models on the standard WMT'14 En→De dataset. Each sentence is tokenized with the Moses tokenizer before breaking into subword units similar to (Sennrich et al., 2016). We use a shared vocabulary of 32k units for each language pair. We report all our results on newstest 2014, and use a combination of newstest 2012 and newstest 2013 for validation. To verify our results, we also evaluate our models on WMT'15 Cs→En. Here we use newstest 2013 for validation and newstest 2015 as the test set. To evaluate the models we compute BLEU on the tokenized, true-case output. We report the mean post-convergence score over a window of 21 checkpoints, obtained using dev performance, following (Chen et al., 2018).

2.2 Baseline Experiments

We base our study on two architectures: Transformer (Vaswani et al., 2017) and RNMT+ (Chen et al., 2018). We choose a smaller version of each model to fit deep encoders with up to 20 layers on a single GPU. All our models are trained on eight P100 GPUs with synchronous training, and optimized using Adam (Kingma and Ba, 2014). For both architectures we train four models, with 6, 12, 16 and 20 encoder layers. We use 6 and 8 decoder layers for all our transformers and RNMT+ experiments respectively. We also report performance for the standard Transformer Big and RNMT+ setups, as described in (Chen et al., 2018), for comparison against higher capacity models.

Transformer: We use the latest version of the Transformer base model, using the implementation from (Chen et al., 2018). We modify the learning rate schedule to use a learning rate of 3.0 and $40,000$ warmup steps.

RNMT+: We implemented a smaller version of the En→De RNMT+ model based on the description in (Chen et al., 2018), with 512 LSTM nodes in both encoder and decoder.

2.3 Analysis

From Tables 1 and 2, we notice that the deeper Transformer encoders completely fail to train. To understand what goes wrong we keep track of the grad norm ratio $r_t = \left(\|\nabla_{h_1} L^{(t)}\| \big/ \|\nabla_{h_N} L^{(t)}\|\right), t = 1 \ldots T$, where $L^{(t)}$ is the loss at time step t, N is the number of layers in the encoder, h_1 is the output of the first encoder layer, h_N is the output of the N-th encoder layer, and T is the total number of train-

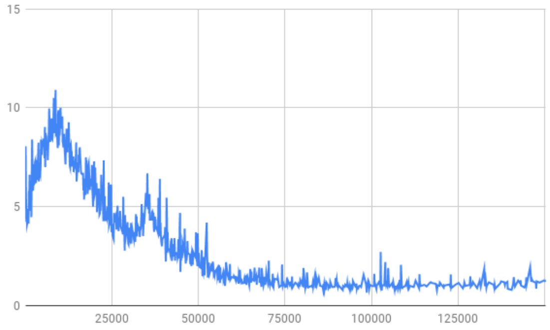

Figure 3: Grad-norm ratio (r_t) vs training step for 20 layer Transformer with transparent attention.

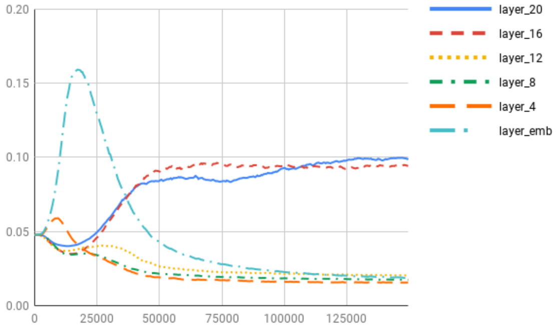

Figure 4: Plot illustrating the variations in the learned attention weights $s_{i,6}$ for the 20 layer Transformer encoder over the training process.

ing steps. We use r_t as a diagnostic measure for two reasons: First, it indicates if training is suffering from exploding or vanishing gradients. Second, when a network is properly trained the lowest layers usually converge quickly, whereas the topmost layers take longer (Raghu et al., 2017). We therefore expect that, for a healthy training process, r_t is relatively large during the early stages of training when updates to lower layers are larger than upper layers. We observe this in most successful Transformer and RNMT+ training runs.

Figure 1 illustrates the r_t curves for the 6-layer and 20-layer Transformers. As expected, the shallow model has a high r_t value during early stages of training. For the deep model, however, r_t remains flat at a much smaller value throughout training. We also observe that r_t remains below 1.0 for both models, although the problem seems much less severe for the shallow model.

From Tables 3 and 4, we also observe that the performance of deep RNMT+ encoders is not significantly impacted, reaching the level of the 6 layer model. This is supported by the RNMT+ r_t curves in Figure 2, which indicate few differences in the learning dynamics of the shallow and deep models. This contrasts with the Transformer experiments, where increasing the depth leads to an unstable training process.

To gain further insights into the stability of the two architectures we completely remove the residual connections from their encoders. Residual connections have been shown, in theory and practice, to improve training stability and performance of deeper networks (see (He et al., 2015; Philipp et al., 2017; Hardt and Ma, 2017; Orhan, 2017)). Removing residual connections leads to disastrous results for the Transformer, where the training process either does not converge or results in significantly worse results. On the other hand, the 6 layer RNMT+ converges with only a slight degradation in quality. Deeper versions of RNMT+ fail to train in the absence of residual connections.

2.4 Regulating Deep Encoder Gradients with Transparent Attention

Our baseline experiments reveal that mechanisms to regulate gradient flow can be critical to improving the optimization of deeper encoders. Since the only difference between our shallow and deep models is the number of layers in the encoder, the trainability issues are likely to be associated with gradient flow through the encoder.

To improve gradient flow we let the decoder attend weighted combinations of all encoder layer outputs, instead of just the top encoder layer. Similar approaches have been found to be useful in deep convolutional networks, for example (Shen and Zeng, 2016; Huang et al., 2016a; Srivastava et al., 2015; Huang et al., 2016b), but this remains un-investigated in sequence-to-sequence models. We formulate our proposal below.

Assume the model has N encoder layers and M encoder-decoder attention modules. For Transformer models each decoder layer attends the encoder, so M is equivalent to the number of decoder layers ($M = 6$). For RNMT+, attention is only applied in the first decoder layer, thus $M = 1$. Let the activations from the i-th encoder layer be $\{h_t^i | t = 1 \dots T\}$, and embeddings be layer 0. Then the traditional attention module attends to $\{h_t^N | \quad t = 1 \dots T\}$. In transparent attention we evaluate M weighted combinations of the encoder outputs, one corresponding to each attention mod-

En→De WMT 14	Transformer (Base)				(Big)
Encoder layers	6	12	16	20	6
Num. Parameters	94M	120M	137M	154M	375M
Baseline	27.26	*	*	*	27.94
Baseline - residuals	*	6.00	*	*	N/A
Transparent	27.52	27.79	**28.04**	27.96	N/A

Table 1: BLEU scores on En→De newstest 2014 with Transformers. * indicates that a model failed to train.

Cs→En WMT 15	Transformer (Base)				(Big)
Encoder layers	6	12	16	20	6
Num. Parameters	94M	120M	137M	154M	375M
Baseline	27.20	*	*	*	27.76
Baseline - residuals	25.83	*	*	*	N/A
Transparent	27.41	27.69	**27.93**	27.80	N/A

Table 2: BLEU scores Cs→En newstest 2015 with Transformers. * indicates that a model failed to train.

ule. We define a $(N+1) \times M$ weight vector W, which is learned during training.[1] We apply dropout to W since we empirically found it helpful to stabilize training. We then compute softmax s to normalize the weights.

$$s_{i,j} = \frac{e^{W_{i,j}}}{\Sigma_{k=0}^{N} e^{W_{k,j}}}, \quad j = 1 \dots M \tag{1}$$

We now define

$$z_t^j = \Sigma_{i=1}^{N+1} s_{i,j} h_t^i, \quad t = 1 \dots T, \quad j = 1 \dots M \tag{2}$$

Now attention module j attends to $\{z_t^j | \quad t = 1 \dots T\}$. Since in RNMT+ a projection is applied to the encoder final layer output, we apply a projection to the weighted combination of encoder outputs before the attention module.

3 Results and Analysis

Our results, from tables 1 and 2, indicate that adding transparent attention improves the performance of most of our transformer experiments, but the gains are most pronounced for deeper models. While the baseline transformer fails to train with 12 layers or deeper encoders, transparent attention allows us to train encoders with up to 20 layers, improving by more than 0.7 BLEU points on both datasets. Relative to Transformer Big, deeper models seem to result in better or comparable performance with less than half the model capacity.

We also observe gains of 0.7 and 1.0 BLEU for RNMT+ models, on En→De and Cs→En respectively, as indicated by Tables 3 and 4. However, experiments comparing wide models against deeper ones are inconclusive. While deeper models perform slightly better than a wide model with double their capacity on Cs-En, they are clearly out-performed by the larger model on En-De.

The r_t plot in Figure 3, also indicates that the learning dynamics now resemble what we expect to see with stable training. We also notice that the scale of r_t now resembles that of the RNMT+ model, although the lower layers converge more slowly for the Transformer, possibly because it uses a much smaller learning rate.

A plot of the weights $s_{i,j}$, in Figure 4, also seems to support our findings. The scalar weights for the lowest embeddings layer grow rapidly in the early stages of training, but once these layers converge the weights for layers 16 and 20 become much larger. The weights for the top few layers remain comparable at convergence, suggesting that the observed gains in performance might also be partially associated with an ensembling effect of the encoder features, similar to the effect observed in (Peters et al., 2018).

---

[1] Here +1 is for the embedding layer.

En→De WMT 14	RNMT+ (512)				(1024)
Encoder layers	6	12	16	20	6
Num. Parameters	128M	165M	191M	216M	379M
Baseline	26.63	26.32	26.49	26.33	**28.49**
Baseline - residuals	26.37	*	*	*	N/A
Transparent	26.61	26.87	27.07	**27.33**	N/A

Table 3: BLEU scores on En→De newstest 2014 with RNMT+. * indicates that a model failed to train.

Cs→En WMT 15	RNMT+ (512)				(1024)
Encoder layers	6	12	16	20	6
Num. Parameters	128M	165M	191M	216M	379M
Baseline	25.77	25.86	26.02	25.75	26.66
Baseline - residuals	25.43	*	*	*	N/A
Transparent	26.69	26.74	**26.79**	26.72	N/A

Table 4: BLEU scores Cs→En newstest 2015 with RNMT+. * indicates that a model failed to train.

4 Conclusions and Future Work

In this work we explore deeper encoders for Transformer and RNMT+ based machine translation models. We observe that Transformer models are extremely difficult to train when encoder depth is increased beyond 12 layers. While RNMT+ models train with deeper encoders, we did not observe any big performance improvements.

We associated the difficulty in training deeper encoders with hindered gradient flow, and resolved it by proposing the transparent attention mechanism. This enabled us to successfully train deeper Transformer and RNMT+ models, resulting in consistent gains in translation quality on both WMT'14 En→De and WMT'15 Cs→En.

Our results show that there is potential for improvement in translation quality by training deeper architectures, even though they pose optimization challenges. While this study explores training deeper encoders for narrow models, we plan to further study extremely deep and wide models to utilize the full strength of these architectures.

5 Acknowledgments

We would like to thank the Google Brain and Google Translate teams for their foundational contributions to this project.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Antonio Valerio Miceli Barone, Jindřich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017. Deep architectures for neural machine translation. *arXiv preprint arXiv:1707.07631*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, Copenhagen, Denmark. Association for Computational Linguistics.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*.

Jacob Devlin. 2017. Sharp models on dull hardware: Fast and accurate neural machine translation decoding on the cpu. *arXiv preprint arXiv:1705.01991*.

Ronen Eldan and Ohad Shamir. 2015. The power of depth for feedforward neural networks. *CoRR*, abs/1512.03965.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Moritz Hardt and Tengyu Ma. 2017. Identity matters in deep learning. *ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2016a. Densely connected convolutional networks. *CoRR*, abs/1608.06993.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. 2016b. Deep networks with stochastic depth. *CoRR*, abs/1603.09382.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. 2016. Learning functions: when is deep better than shallow. *arXiv preprint arXiv:1603.00988*.

A. Emin Orhan. 2017. Skip connections as effective symmetry-breaking. *CoRR*, abs/1701.09175.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

George Philipp, Dawn Song, and Jaime G. Carbonell. 2017. Gradients explode - deep networks are shallow - resnet explained. *CoRR*, abs/1712.05577.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep understanding and improvement. *arXiv preprint arXiv:1706.05806*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Falong Shen and Gang Zeng. 2016. Weighted residuals for very deep networks. *CoRR*, abs/1605.08831.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Matus Telgarsky. 2016. Benefits of depth in neural networks. *CoRR*, abs/1602.04485.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. *arXiv preprint arXiv:1705.00861*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199*.

Context and Copying in Neural Machine Translation

Rebecca Knowles
Dept. of Computer Science
Johns Hopkins University
`rknowles@jhu.edu`

Philipp Koehn
Dept. of Computer Science
Johns Hopkins University
`phi@jhu.edu`

Abstract

Neural machine translation systems with subword vocabularies are capable of translating or copying unknown words. In this work, we show that they learn to copy words based on both the context in which the words appear as well as features of the words themselves. In contexts that are particularly copy-prone, they even copy words that they have already learned they should translate. We examine the influence of context and subword features on this and other types of copying behavior.

1 Introduction

In translation, certain tokens – often names and numbers – should be copied from the source sentence to the target sentence. Word copying is fairly straightforward in phrase-based statistical machine translation, where unknown words can be left untranslated (copied to the target). It poses more of a challenge in neural machine translation systems, which often use limited or subword vocabularies and soft attention rather than strict alignment. This has resulted in a variety of approaches to copying, which make use of pre-/post-processing and/or network modifications (e.g. explicit switching between generation and copying).

Neural machine translation models that use subword vocabularies to perform open-vocabulary translation have been observed to correctly translate unknown words or copy words (one subword at a time, if need be) even when the full word to be translated or copied was not observed in training. Koehn and Knowles (2017) found that neural machine translation systems using subword vocabularies outperformed phrase-based statistical machine translation systems on the translation of unknown words. This raises the questions that we seek to answer: to what extent does byte-pair encoding[1] solve the copying problem (without requiring modifications to the network structure)? More generally, what are subword neural machine translation models learning about copying?

We find that neural machine translation systems (with attention, trained on subword vocabularies) learn to copy words (both novel and observed) based on their sentential contexts. Additionally, though the models have no knowledge about the components of each subword unit, they learn that certain categories of tokens (e.g. capitalized tokens) tend to be copied. We use quantitative and qualitative evaluations to shed light on what these models learn about copying tokens and about the contexts in which copying occurs.

2 Related Work

Prior work on copying in neural machine translation has typically focused on rare or unknown words. Luong et al. (2015) augment data with word alignments to train a neural machine translation system (without attention) that emits both a translation and source word positions for any out-of-vocabulary (OOV) tokens emitted. They post-process OOVs with a dictionary or by copying. Currey et al. (2017) augment training data with monolingual target language text as bitext and find that it improves copying in low-resource settings. Ott et al. (2018) and Khayrallah and Koehn (2018) examine negative effects of source copying.

Both Gu et al. (2016) and Gulcehre et al. (2016) modify neural sequence to sequence models to explicitly perform copying. Gu et al. (2016) focus on monolingual tasks (dialogue systems and summarization), proposing a model that can both generate and copy text. Gulcehre et al. (2016) perform experiments on neural machine translation (with attention), using whole-word vocabularies (and an UNK token to represent unknown words). Their model incorporates a switching variable that determines whether to copy or generate a translation.

In this work, we focus on subword vocabularies for neural machine translation, using byte-pair en-

---

[1] A type of subword vocabulary (Sennrich et al., 2016b).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3034–3041
Brussels, Belgium, October 31 - November 4, 2018.

Data	% Tokens Copied	
	DE	*EN*
Europarl	1.8%	2.0%
News Commentary	2.9%	3.3%
Full Training (EN–DE)	7.6%	8.1%
Full Training (DE–EN)	8.6%	9.2%

Table 1: Percentage of tokens which should be copied, across training data sources.

coding (BPE, Sennrich et al. (2016b)). The other approaches described are somewhat orthogonal to the use of subword vocabularies, but may require modifications to handle subwords.

3 Data and Models

We train German–English (DE–EN) and English–German (EN–DE) neural machine translation models with attention, similar to the University of Edinburgh's WMT 2016 submissions (Sennrich et al., 2016a). Models are trained using the Marian toolkit (Junczys-Dowmunt et al., 2018).[2] We use the WMT parallel text[3] (Europarl, News Commentary, and CommonCrawl) along with synthetic backtranslated data.[4]

4 Initial Analysis

We analyze the training data to learn about the prevalence and characteristics of words that should be copied in translation and the contexts in which they occur. We consider both the full training data (including backtranslations and CommonCrawl) and cleaner subsets. We restrict our search for copied words to tokens of length 3 or more characters.[5] Our heuristic for detecting copied tokens is this: a word is a "copied token" if it appears the same number of times in both the source and target sentence.[6] As we will show, copied words tend to belong to specific categories (proper nouns, numbers, etc.) which coincide with their repeated appearance in certain contexts (e.g. names following titles like "Ms" or "Prime Minister").

4.1 Where do copied words appear?

In Table 1, we see that between 1.8% and 9.2% of tokens are copied.[7] Though the majority (or near-majority) of sentences do not contain any copied words (of length 3 or more), copied words are still quite prevalent: approximately 18% of sentences in each full training dataset contain one, 4% to 5% contain four, and there is a long tail (one sentence contains 70). Sentences with many copied words often contain direct quotations, third language text (not source/target), or a sequence of copied words (e.g. comma-separated numbers or names).

The cleaner Europarl and News Commentary corpora have lower percentages of copied tokens than the overall training data. Of particular note, the backtranslated data contains some examples of copying that we'd prefer for the system *not* to learn, such as target language words appearing untranslated in the (backtranslated) source side data.

4.2 What words are copied?

We first examine the part-of-speech (POS) tags[8] of copied words. In the EN–DE training data, most copied words are tagged on the English side as NNP (proper noun, singular), including names of individuals, places, or organizations (eg. González, Wales, Union). The next most frequent categories are CD (cardinal number) – including numbers like *42* that should be copied and ones like *seven* which should be translated – and NN (noun, singular or mass). The results are similar for DE–EN training data (tagged on German with a different tag set): PROPN (proper noun) is the most frequent tag for copied words, followed by NUM (numbers) and NOUN. Punctuation would rank highly if we included short tokens.

5 Experiments and Analysis

We address two main questions: (1) Do certain *contexts* encourage copying? (2) Do certain *words* exhibit features that make them more likely to be copied (regardless of context)?

5.1 Contexts

Working from the intuition that certain contexts indicate that copying should occur – for example, a name following a title like "Ms" or "Frau"

---

[2] We use recommended settings and early stopping, with results comparable to WMT 2016 systems, with BLEU scores of 39.9 (DE–EN) and 33.2 (EN–DE) on the 2016 test set.

[3] http://www.statmt.org/wmt16/translation-task.html

[4] http://data.statmt.org/rsennrich/wmt16_backtranslations/

[5] This has the benefit of removing words like *in* which are the same in German and English, but may nonetheless be considered translations rather than copies.

[6] In DE–EN, we find one notable exception to this heuristic – *was* – which is a homograph, not a copy. It makes up $<1\%$ of copied tokens in Europarl/News Commentary.

[7] The two full training sets differ due to the synthetic backtranslated data; the rest of the corpora are identical.

[8] POS tags are generated by the Stanford POS tagger (Toutanova et al., 2003). For English: english-left3words-distsim.tagger. For German: german-ud.tagger.

S: Therefore, Mrs Ashton, your role in this is invaluable.
R: Darum, Frau Ashton, ist Ihre Aufgabe in diesem Zusammenhang von unschätzbarem Wert.
T: Therefore, Mrs [NNP], your role in this is invaluable.
E1: Therefore, Mrs BBC, your role in this is invaluable.
D1: Deshalb, Frau BBC, ist Ihre Rolle hierbei von [...]
E2: Therefore, Mrs June, your role in this is invaluable.
D2: Deshalb, Frau June, ist Ihre Rolle dabei von [...]
E3: Therefore, Mrs Lutreo, your role in this is invaluable.
D3: Daher, Frau Lutreo, ist Ihre Rolle hierbei von [...]

Table 2: Source, reference, template, and examples of template-token combinations. *E1* has a word usually (76.0% of the time) copied in training, *E2* has one rarely (0.8% of the time) copied, and *E3* has a novel one. In training, 84.8% of NNPs with this left bigram context (“, Mrs”) were copied.

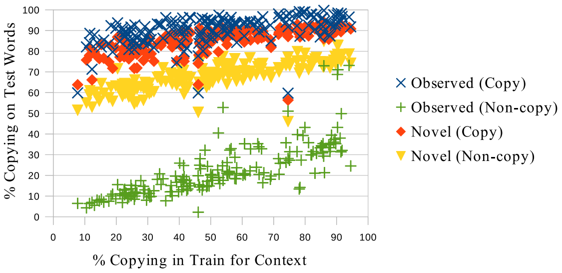

Figure 1: Percent of NNP (EN–DE) tokens copied by how copy-prone the context is, by category. Each point is the percentage of copying for all within-category words, across all example templates for one particular context (averaged over between 1,100 (novel-non-copy) and 13,150 (observed-non-copy) binary copy values).

should often be copied – we examine the relationship between context and copying, focusing on left bigram contexts. We show that the machine translation system learns that certain contexts are so indicative of copying that it will even copy (not translate) words that it has *learned to translate* if they are seen in a sufficiently copy-prone context.

For each POS, we collect a set of left bigram contexts that precede a word with that tag. We filter by frequency and diversity of tokens following the bigram.[9] For each context-POS pair, we select 50 random templates from the training data containing the bigram context followed by a word with that POS.[10] Each context-POS pair is associated with a percentage that represents how often it exhibited copying in the training data. For example, in the copy-prone context “thank Mrs [NNP]” the NNP was copied 91.1% of the time, compared to 15.3% of the time in “Republic of [NNP]”.[11]

We take all word types with a given POS tag from the WMT 2016 test set, dividing them into four categories based on two binary distinctions: *observed* (in training data) or *novel* (not observed in training), and *copy* (typically copied) or *non-copy* (not typically copied) and filter the observed ones based on training frequency. We count words as *non-copy* if they were copied $\leq$ 30% of the time, and as *copy* if they were copied $\geq$ 70% of the time.[12] We then combine each word with each POS-appropriate example template and perform preprocessing (including BPE) and translation.[13] Table 2 shows examples.

For each context, we calculate the percentage (across all example templates for that context and all words, separated by observed/novel and copy/non-copy categories) of the time that the words in that context were copied. We then compare it to the percentage of the time that copying occurred for that context-POS tag pair in training.

Figure 1 shows NNP (EN–DE) results. Both observed-copy and novel-copy words behave almost identically, with copying percentages generally above 80%, and a slight trend upward as contexts become more copy-prone (moving to the right along the horizontal axis). Novel-non-copy words shadow these, but with a drop in copying percentage (see Section 5.3). Most interesting is the observed-non-copy category. In contexts that are not copy-prone, minimal copying occurs.[14] However, as they are placed in increasingly copy-prone contexts, even these words that the system has learned it should *translate* are being copied. We observe the same trend for words tagged NN and CD, and for PROPN, NOUN, and NUM words in the DE–EN direction. This demonstrates that the machine translation system has learned that certain contexts are copy-prone.

We manually analyze outliers that appear much

---

[9] See Appendix A for details and examples of contexts.

[10] We select contexts and templates from the full training data, rather than only Europarl/News Commentary, because we are interested in what patterns the model is learning from *all* data to which it has been exposed.

[11] For DE–EN translation, we see similar patterns: two of the three most copy-prone PROPN left bigram contexts are “sagte Frau” and “sagte Herr” (“said Ms/Mr”), while many less copy-prone ones end with articles.

[12] See Appendix B for details.

[13] We use the Marian batch decoder, with recommended settings: beam size 6 and length normalization penalty of 0.6.

[14] Note that some of the non-copy words were sometimes copied in training data, even if only in backtranslations.

	Drop	**Change**	**Other**
Novel-Copy	24	102	60
Novel-Non-Copy	14	128	50
Observed-Copy	51	6	126
Observed-Non-Copy	12	1	186

Table 3: Counts of automatically detected output categories (*drop*, *change*, and *other*) for a sample of NNP tokens (EN–DE) that were not copied.

more or less copy-prone than expected. In both cases, the cause appears the same: the context occurred repeatedly in many very similar sentences in the training data. Highly copy-prone contexts that produced copying percentages greater than 70% even in observed-non-copy tokens often appeared in common boilerplate text (e.g. "stay at [NNP]" or "rates for [NNP]" followed by "Hotel").[15] Where we observe lower than expected rates (e.g. ") of [NNP]"), we find that the system may have memorized training sentences.

5.2 Analysis of Words That Are Not Copied

When words are not copied, what sort of output is the system producing? We find that it typically falls into one of four categories: *drop* (no target token aligns with the source token), *change* (the word is changed: partially translated, transliterated, or inflected even if it is not a target language word), *substitution* (the word is replaced with a fluent but not adequate substitute), or *translation* (translated into a target language word).

We begin with an automatic analysis. We randomly sample 200 examples each of sentences containing words that were not copied for novel-copy, novel-non-copy, observed-copy, and observed-non-copy NNPs (EN–DE). We retranslate each sentence and produce a soft alignment matrix from the attention mechanism, then convert the soft alignments between BPE segments into hard alignments between the source word and one or more target words.[16] A word has been *dropped* if it is unaligned. We count a word as being *changed* if any words it is aligned to have any subword (BPE segment) overlap with the original word's subwords. Both *substitution* and *translation* fall under *other*; we analyze those manually.

Results are shown in Table 3.[17] For all novel words, the most frequent output type is *change*. For example, the novel NNP *Bishnu* is changed into *Bischnu* in German.[18] Other changes include translations of parts of the word, and concatenation with other tokens. The output token often starts with the same character or sequence of characters as the source token.[19]

We manually inspect examples in the *other* category. For observed-non-copy words, almost all are translations (e.g. *Sea* translated correctly as *Meer*), as expected. For observed-copy words, we see a mix of translations and other changes to the words, which are almost evenly split between substitutions and small changes. These include inflections (e.g. *Bremen magazine* reasonably translated as *Bremer Magazin*[20]).

Within the *other* category, perhaps the most interesting cases are those where words appear to be substituted with a fluent but not adequate alternative. Many substitutions occur when the rare word is inserted next to a word that often forms a collocation (like "United States" – in sentences that include "in the [NNP] States" the translation sometimes defaults to a translation of "United States" regardless of the actual NNP inserted in place of "United"). Others have a less common NNP swapped for one that belongs to a similar semantic category (e.g. the place name *Dublin* being generated instead of the less common *Halle* – as Arthur et al. (2016) and others observed). For novel-copy words labeled as *other*, three quarters are substitutions and one quarter exhibit small changes. The reverse is true for novel-non-copy words: the majority exhibit small changes while almost thirty percent are substitutions.

5.3 Properties of Copied Words

Certain words exhibit properties that make them more likely to be copied, regardless of context. At first glance, it seems unintuitive that the rate of copying of novel-copy words and novel-non-copy words differs (Fig. 1) – the model has never observed any of these words, and they are being presented in identical contexts – why does it differentiate between them? Doing so indicates that the model has learned what makes a sequence of

---

[15] Since hidden representations contain whole sentence information, right side context may influence copying too.

[16] We use AmuNMT (Junczys-Dowmunt et al., 2016), producing slightly different output. See Appendix C for details.

[17] Rows do not sum to 200 because some words in our random sample were copied by the the AmuNMT decoder.

[18] A near-transliteration – the "sh"/"sch" transformation is seen in EN–DE cognates, e.g. "ship" and "Schiff".

[19] Appendix D contains examples of this and more.

[20] *Bremen* and *Bremer* are unique BPE segments, so the *change* heuristic could not be applied.

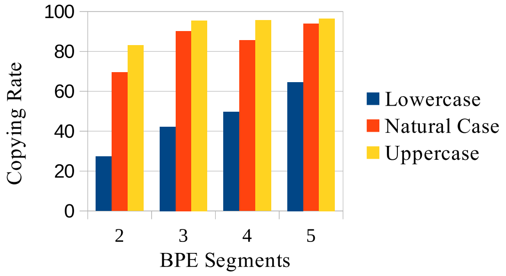

Figure 2: Copying rate based on casing and number of BPE segments for novel NNP words (EN–DE), averaged across all NNP contexts.

subwords likely to be copied.

Belinkov et al. (2017) observe that neural machine translation models may encode information about part-of-speech, which could be used when determining whether or not to copy (but does not explain within-POS differences). For numbers, it mainly learns to copy numerical portions while changing commas to periods and vice versa (as required by the target language's conventions). Nouns and proper nouns are more interesting: some should be translated (e.g. novel noun compounds like *hallmate*), or, in the case of misspellings (e.g. *manfacturer*), corrected, while others should be copied. For novel NN words, there is another striking difference between copy and non-copy: most of the former contain capital letters and most of the latter do not.

5.4 Capitalization and Copying

To experiment with the influence of capitalization on copying, we take each novel NNP word (96 copy and 22 non-copy) and convert it to lowercase, leave it in its natural case (all have at least one uppercase letter), or convert it to uppercase. We then translate all of them in all NNP contexts (from previous EN–DE experiments). Using only novel words sidesteps the issue of truecasing.

Lowercase words are the least frequently copied (average copy rate of 40.2%), uppercase words are the most copied (94.4%), and the natural case falls in the middle (81.7%). However, changing casing changes the BPE segmentation, and uppercase words tend to be split into more pieces: a mean of 4.4 segments, as compared to means of 3.1 (lowercase) and 2.9 (natural case). The number of subword segments correlates positively with copying rate (Fig. 2), but, controlling for that, we still find that NNP words that are completely capitalized tend to be copied more than those with the same number of subword segments but only lowercased letters, suggesting that the system is encoding information about the connection between capitalization and copying. We also perform this experiment with PROPN words in the DE–EN direction, and find that increased capitalization increases copying, though we do *not* find there that an increase in the number of BPE segments increases copying. The true casing of the word consistently falls between these two extremes. The high copying rate of fully-capitalized words is intuitive: acronyms are often both uppercased and copied from source to target. That is not to say that the model always learns to copy acronyms; it also learns to translate them when appropriate (such as *GDP* to *BIP*). There is always an interplay between learned translations and features that may encourage copying.

The connection between copying rate and capitalization provides one explanation for the gap in behavior of the two novel word types, and demonstrates that features of words influence copying. Note that it learns this behavior based on training data, without access to information at a finer granularity (character-level) than the subword units.

6 Conclusion

We show that subword vocabulary neural machine translation systems learn about copying from context and the subwords themselves. The effect of context is strong enough to cause words that would otherwise be translated to be copied. Characteristics of subword tokens play a role in copying behavior, with capitalized tokens more likely to be copied. We leave as future work a deeper analysis of the level of character-awareness encoded in representations of the BPE segments as a byproduct of training. We provide an analysis of what happens when words are not copied, showing expected differences between novel words and words that were observed during training. Additionally, we provide more examples and evidence of the problem of substituting fluent but non-adequate translations for rare or unknown words.

Acknowledgments We thank the reviewers and our colleagues for comments and suggestions. Rebecca Knowles was supported by a National Science Foundation Graduate Research Fellowship under Grant No. DGE-1232825. This work was also funded by the IARPA MATERIAL project.

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Austin, Texas. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156, Copenhagen, Denmark. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? a case study on 30 translation directions. In *Proceedings of the 9th International Workshop on Spoken Language Translation (IWSLT)*, Seattle, WA.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, Andr F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in c++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Huda Khayrallah and Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83, Melbourne, Australia. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation (WMT)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Appendices

A Collection of Left Bigram Contexts

We use left bigram contexts as a proxy to evaluate the contexts in which words are copied. Here, we provide details of the context selection process described in Section 5.1. In overview: for each POS, we collect the full set of left bigram contexts that ever precede a word with that tag, then filter by frequency and subsequent token diversity.

For each POS, first, we find all left bigram contexts $(tok0, tok1, copiedword)$ that occur in the training data (where the copied word was tagged with the given POS). We then filter this set so that it only contains contexts $(tok0, tok1)$ that appeared at least 1000 times (left of NNP/PROPN) or 500 times (left of CD/NN/NUM/NOUN) in the training data. To ensure that we're not simply capturing collocations ("European Union"), we filter out left bigram contexts that have been followed by fewer than 150 unique types with that particular POS. This results in between 53 and 276 contexts, as shown in Table 4.

POS	**Num. Contexts**
NNP	176
NN	82
CD	74
PROPN	276
NOUN	66
NUM	53

Table 4: Context counts by POS tag (NNP, NN, CD for EN–DE; PROPN, NOUN, NUM for DE–EN), selected as described in Appendix A.

Each context is then associated with a copying rate, calculated as the number of times the token (with the given POS tag) following $(tok0, tok1)$ is copied, divided by the total number of times $(tok0, tok1)$ was observed to be followed by a token with that POS tag. In Table 5, we show the most- and least-copy-prone contexts for EN–DE (those with the highest and lowest copying rates).

POS	**Context**	**Copy Rate**
NNP	Finance Minister	94.5%
	rates for	94.0%
	congratulate Mr	91.7%
	between the	10.5%
	President ,	7.7%
CD	updated on	94.0%
	the B	0.1%
NN	notified when	97.3%
	the first	0.6%

Table 5: Left bigram contexts with the highest/lowest copying rates (EN–DE), by POS tag.

	Novel		Observed	
POS	**Copy**	**Non-C.**	**Copy**	**Non-C.**
NNP	96	22	251	263
NN	14	16	13	1664
CD	3	29	60	44
PROPN	92	76	463	418
NOUN	12	222	29	2176
NUM	2	29	55	68

Table 6: Counts of each word type by novel/observed, copy/non-copy distinction and POS tag (NNP, NN, CD are EN–DE; PROPN, NOUN, NUM are DE–EN).

B Collection and Labeling of Copy/Non-Copy Words

In Section 5.1, we give a high-level description of how we collect and label words. All words that we examine are labeled as either *copy* or *non-copy*. For words that were observed in training, we filter out those that appeared fewer than 1000 times. We label them as *copy* if they were copied $\geq 70\%$ of the time in training data (according to the heuristic described in Section 4), and as *non-copy* if they were copied $\leq 30\%$ of the time in training data, discarding the remainder. For words that were unobserved in training, we used the same threshold but calculate it over all instances in the test data (with no requirement that they appear a certain number of times). Table 6 shows the number of words selected after filtering and thresholding.

C Attention and Alignments

We produce soft alignments (the attention matrix) using the AmuNMT decoder with the "return-nematus-alignment" flag set. It performs normalization differently than Marian's decoder (producing slightly different outputs for many sentences, including sometimes copying words that were not copied in our original translations).

For each target (subword) token, we align it to the source (subword) token with the highest soft alignment weight. Given our source word of interest s (composed of subword segments $s_1 \ldots s_n$), we define its translation to be the list of all target words t (composed of subword segments $t_1 \ldots t_m$) for which any subword t_i was aligned to a subword s_j of s.

D Additional Examples

D.1 Changes

Here we show additional examples of *changes*, like the transliteration-like change of *Bishnu* to *Bischnu* described in Section 5.2.

There are also partial translations when BPE segments are full source language words – like *Thneed* (segmented “Th@@ need”) becoming *ThNotwendigkeit* (segmented “Th@@ Notwendigkeit” – *Notwendigkeit* is a valid translation of *need*). Sometimes, a token *is* copied but then concatenated with another token.

Even without overlap of BPE segments between the source and the translation, changed words sometimes share a number of characters (especially at the beginning or end of a word). Half of the *other* category output of *Thneed* (“Th@@ need”) begin with the letter “T” (but not the BPE token “Th@@”). This may suggest some level of character-awareness in the representations of BPE segments, produced as a byproduct of training. We leave a deeper analysis of this to future work.

D.2 NNP Substitutions

Here we provide additional examples of substitutions, as seen in Section 5.2. These findings provide additional support and nuance to the study of this phenomenon of neural machine translation system errors.

Many substitutions occur when the rare word is inserted next to a word that often forms a collocation (like “United States” or “European Union” or “Madam President”). For example, in a template where “in the [NNP]” is followed by “States”, inserting the NNP *Accies* results in “in the *Accies* States” – which was then translated by the system as “in den Vereinigten Staaten” (gloss: “in the United States”).

We also observe examples that may have to do with a combination of (in)frequency of tokens and the context. For example, we have the novel NNP *Sloveina* (perhaps a misspelling of *Slovenia*), which is often replaced with *Slowaken* (*Slovakia*) when translated to German. In another sentence, we find that “this year, *Angela* expects” is translated to “in diesem Jahr erwartet *Merkel*” despite *Merkel* appearing nowhere in the source text. The first and last names of German chancellor Angela Merkel appear frequently together in training data, and thus likely have sufficiently similar representations. We see other similar substitutions: *Mitt* for *Romney*, *US* for *Obama*, and *Thomas* for *Sarah*. Sometimes a specific name is replaced with a title, such as “your prime minister, *York*” being translated as “ihr Premierminister, *Herr Präsident*” (glossed as “your prime minister, *Mr. President*”).

E Additional Plots

Here we include plots for DE–EN PROPN. Fig. 3 shows context experiments. It shows similar trends to Fig. 1, but with a greater gap between novel- copy/non-copy words. As noted, the DE–EN capitalization experiments (Fig. 4) show the same trends as EN–DE in terms of capitalization (despite the capitalization of all nouns in German), but not in terms of numbers of BPE segments.

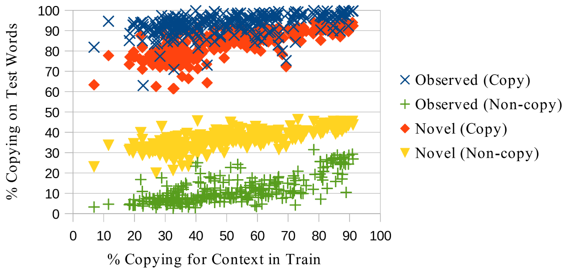

Figure 3: Percent of PROPN (DE–EN) tokens copied by how copy-prone the context is, by category. Each point is the percentage of copying for all within-category words, averaged across all example templates for one particular context.

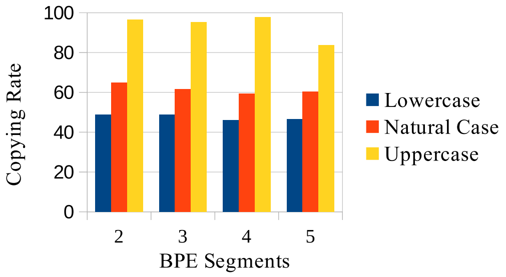

Figure 4: Copying rate based on casing and number of BPE segments for novel PROPN words (DE–EN), averaged across all PROPN contexts.

Encoding Gated Translation Memory into Neural Machine Translation

Qian Cao and **Deyi Xiong**[*]
School of Computer Science and Technology, Soochow University, Suzhou, China
`20174227009@stu.suda.edu.cn`; `dyxiong@suda.edu.cn`

Abstract

Translation memories (TM) facilitate human translators to reuse existing repetitive translation fragments. In this paper, we propose a novel method to combine the strengths of both TM and neural machine translation (NMT) for high-quality translation. We treat the target translation of a TM match as an additional reference input and encode it into NMT with an extra encoder. A gating mechanism is further used to balance the impact of the TM match on the NMT decoder. Experiment results on the UN corpus demonstrate that when fuzzy matches are higher than 50%, the quality of NMT translation can be significantly improved by over 10 BLEU points.

1 Introduction

Neural machine translation, an emerging machine translation (MT) technology, has made remarkable progress in the past few years (Cho et al., 2014; Sutskever et al., 2014), which strongly encourages many translation agencies to embrace it for product deployment. A natural question during this deployment is how the strengths of both the traditional TM and new NMT technologies can be combined together for professional high-quality translation.

Such attempts to the TM and MT combination have been already conducted in the context of statistical machine translation (SMT). A variety of efforts have been made to incorporate matched translation segments from TM into SMT (Koehn and Senellart, 2010). Partially inspired by these efforts, we aim at combining TM and NMT in this paper.

Different from TM and SMT, both of which use symbolic fragments to construct translations, NMT induces translations from a real-valued continuous space. Furthermore, NMT is trained in an end-to-end fashion, which makes it not easy to be amenable to external intervention. Therefore, incorporating TM as external knowledge into NMT is challenging.

In this paper, we propose a novel and effective method to address this issue in the combination of TM and NMT. The key idea behind this method is to mimic human translators in translating a source sentence given a similar source sentence with a translation. We treat the matched TM translation as an additional signal and try to encode it with a new encoder to guide the NMT decoder to translate the current sentence. Specifically, we first find the sentence that is most similar to the current source sentence from TM by calculating their semantic similarity based on sentence embeddings. In order to prevent the TM matched translation from dominating the decoding process, we introduce a gate mechanism to balance the TM translation signal and the current source sentence which are encoded separately by two different encoders.

A series of experiments on the Chinese-English UN corpus demonstrate that when fuzzy matches are over 50%, the proposed method can significantly improve NMT with the gated TM signal. We also conduct an in-depth analysis on the TM gate, which shows that the gate can indeed regulate the information flow from TM to the NMT decoder.

2 Encoding Gated TM into NMT

In this section, we elaborate our proposed method that encodes translation memories into neural machine translation with a gating mechanism. We refer to our method as NMT-GTM, which consists of three essential components: i) coupled encoders that encodes both the source sentence and matched TM translation separately, ii) a TM gating network that controls the encoded signal from the TM matched translation and iii) a TM-guided decoder

---

[*] Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3042–3047
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

that incorporates the gated TM signal into the decoding. The diagram of NMT-GTM is shown in Figure 1.

For each source sentence src, we retrieve TM to find the most similar sentence to it. Different from the combination of TM and SMT, we define the best TM match as the sentence with the highest cosine similarity which is calculated based on sentence embeddings (Le and Mikolov, 2014), instead of being selected based on fuzzy match score. This is consistent with NMT that performs in an embedding-defined semantic space. But we display our results in experiments according to fuzzy match scores for easy understanding. We use tm_s to denote the most semantically similar sentence to src from TM and tm_t its translation.

2.1 Coupled Encoders

We use a pair of encoders to separately encode the source sentence src and its matched TM translation tm_t. Both encoders are running independently of each other with bidirectional GRU recurrent neural networks[1] (Chung et al., 2014). Accordingly, two separate attention networks are employed to obtain context representations for both src and tm_t, which we denote as c^{src} and c^{tm_t} respectively. The attention network for the TM matched translation is able to help detect matched translation segments from tm_t for the decoder.

2.2 TM Gating Network

When we translate a source sentence, in addition to the input of the sentence itself, we also have a TM matched translation (tm_t) semantically similar to the sentence as an additional input. We want the additional input to act as a translation example for providing positive guide to target word prediction. In order to balance the information flow from the two inputs (src and tm_t) into the decoder, we further introduce a TM gating network to control the respective proportions of tm_t and src, partially inspired by Tu et al. (2017) who propose a gating mechanism to combine source and target contexts. We formulate the TM gating network as follows:

$$g^{tm} = f(s_{t-1}, y_{t-1}, c^{src}, c^{tm\_t})$$

where s_{t-1} is the previous hidden state, y_{t-1} is the previously predicted target word, and f is a logistic sigmoid function.

	train	dev	test
#Sentences	1, 117, 452	804	1, 614
Average FMS	0.1890	0.5493	0.5392

Table 1: Statistics of the training data, development and test set. FMS: fuzzy match score.

2.3 TM-Guided Decoder

In the TM-guided decoder, we integrate the gated TM information into the decoding process and use the context representations of src and tm_t to predict the hidden state of the decoder in each time step. The decoder hidden state s_t is computed as follows:

$$s_t = GRU(s_{t-1}, y_{t-1}, c^{src} * (1 - g^{tm}), c^{tm\_t} * g^{tm})$$

where * is an element-wise multiplication.

The conditional probability of the next word y_t is calculated as follows:

$$p(y_t | y_{<t}, src) = g(f(s_t, y_{t-1}, c^{src}))$$

Please notice that we only incorporate the gated TM into the hidden state of the decoder, rather than the prediction of the next word. Our goal is to correctly translate the source sentence with reference to the translation of the TM match tm_t. In other words, tm_t only plays a supporting role in translation. We don't want too much information from TM to affect the translation of the source sentence. Therefore, we incorporate the gated TM in a way that it can only indirectly influence the target generation via hidden states. In our experiments, we observe that this helps our proposed model to faithfully translate a source sentence, instead of copying all information from the TM matched translation, especially for source sentences with slight differences (e.g., dates or numbers) from TM matches.

3 Experiments

We conducted a series of experiments on Chinese-English corpus to evaluate the effectiveness of the proposed NMT-GTM and analyzed the TM gate.

3.1 Experimental Settings

Our data come from the Chinese-English United Nations Parallel Corpus (Rafalovitch et al., 2009), which consists of official records and other parliamentary documents. Since large-scale public

---

[1] In this paper, we use GRU encoders and decoders. However, our method can be applicable to other encoders and decoders.

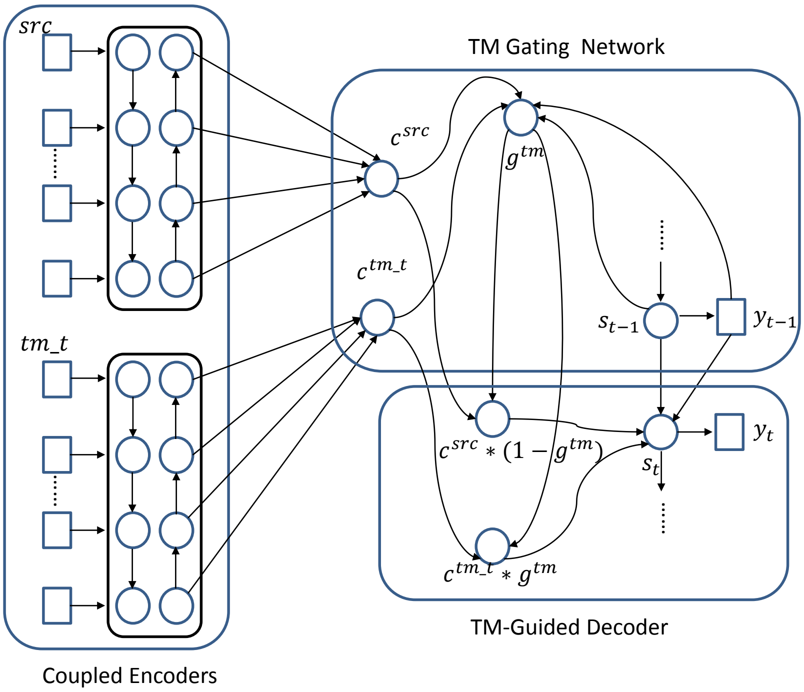

Figure 1: Model Architecture of NMT-GTM

FMS	**#Sentences**
$[0.9, 1.0)$	171
$[0.8, 0.9)$	182
$[0.7, 0.8)$	178
$[0.6, 0.7)$	179
$[0.5, 0.6)$	181
$[0.4, 0.5)$	177
$[0.3, 0.4)$	180
$[0.2, 0.3)$	185
$(0.0, 0.2)$	181

Table 2: The numbers of sentences of the test set in each fuzzy match score group.

translation memories are not easily available, we built a translation memory from the UN corpus. Specifically, we divided the Chinese-English UN corpus into two parts UN_a and UN_b with equal size. For each source sentence s_a from UN_a, we chose the source sentence s_b from UN_b that has the highest semantically similarity to s_a, computed in the way described in the last section. In doing so, we built a corpus with matched pairs $(s_a/t_a, s_b/t_b)$ where $t_{a/b}$ are translations corresponding to $s_{a/b}$. Then we computed the fuzzy match score for each pair of source sentences as follows:

$$FMS(s_a, s_b) = 1 - \frac{Levenshtein(s_a, s_b)}{max(|s_a|, |s_b|)}$$

where $Levenshtein(s_a, s_b)$ is the word-based Levenshtein Distance between s_a and s_b. The fuzzy match score can also be calculated with other methods, e.g., the method introduced in (Bloodgood and Strauss, 2015). We leave FMS estimated with different methods to our future work. We selected all pairs $(s_a/t_a, s_b/t_b)$ with a fuzzy match score $FMS >= 0.5$. From those pairs with $FMS < 0.5$, we randomly selected 20% of them. These selected pairs were then divided into 9 groups according to their fuzzy match scores (e.g., $FMS \in [0.5, 0.6)$). We randomly chose approximately the same number of sentences from each group to create a development set and test set. The remaining data were used to create the training data (i.e., $\{(s_a, t_b, t_a)_{selected}\}$) and translation memory (i.e., $\{(s_b, t_b)_{selected}\}$). Statistics of the training data, development and test set are shown in Table 1. The numbers of sentences of the test set in each fuzzy match score group are presented in Table 2.

We used RNNSearch as our NMT baseline. We set the maximum sentence length of training corpus to 50 words both for the Chinese and English sides. The sizes of vocabularies of both sides were

FMS	**RNNSearch**	**NMT-GTM**	**TM**
$[0.9, 1.0)$	43.97	77.67	94.23
$[0.8, 0.9)$	47.32	79.78	79.84
$[0.7, 0.8)$	50.95	**71.53**	67.11
$[0.6, 0.7)$	56.12	**65.39**	58.93
$[0.5, 0.6)$	65.01	**66.46**	46.99
$[0.4, 0.5)$	67.83	66.30	34.67
$[0.3, 0.4)$	58.51	56.83	22.93
$[0.2, 0.3)$	46.12	44.42	9.72
$(0.0, 0.2)$	31.41	29.83	1.18
$(0.0, 1.0)$	51.11	**61.43**	47.16

Table 3: BLEU scores for translations from RNNSearch, NMT-GTM and TM.

FMS	**ref_as_TM**	**TM_ave_gate**	**ref_ave_gate**
$[0.9, 1.0)$	81.51	0.6712	0.6735
$[0.8, 0.9)$	85.94	0.6543	0.6582
$[0.7, 0.8)$	85.87	0.6385	0.6477
$[0.6, 0.7)$	83.13	0.6075	0.6267
$[0.5, 0.6)$	84.55	0.5995	0.6218
$[0.4, 0.5)$	85.13	0.5755	0.6035
$[0.3, 0.4)$	78.63	0.5721	0.6083
$[0.2, 0.3)$	76.78	0.5652	0.6409
$(0.0, 0.2)$	70.89	0.5633	0.6699
$(0.0, 1.0)$	81.04	0.6047	0.6388

Table 4: Changes of the TM gate. The second column shows the BLEU scores with reference translations being used as additional TM inputs. The third column represents the average gate values of the standard setting, while the last column represents the average gate values when references are used as additional TM inputs.

set to 30k. For those words that are not in the vocabulary, we replaced them with a special token UNK. We set the dropout to 0.5. All the other settings were the same as those described by Bahdanau et al. (2014). We used the stochastic gradient descent algorithm with Adam (Kingma and Ba, 2014) to train NMT models. The learning rate was set to 0.0004. The size of mini-batch was set to 80 sentences. The beam size was set to 10 during decoding.

For the proposed NMT-GTM model, we used tuples (src, tm_t, tgt) as input. The rest of the parameter settings were consistent with the baseline model. To calculate the cosine similarity, we used the fasttext tool [2] with the dimension of 100 to obtain sentence embeddings.

3.2 Experimental Results

Table 3 shows the results of different NMT systems measured by BLEU (Papineni et al., 2002). From the table, we can find that when fuzzy match scores are over 50%, the extra introduction of TM information can significantly help NMT to better translate. Even when fuzzy match scores are lower than 50%, the translation quality does not drop too much. On the entire test set, the proposed gated combination model of TM and NMT improves the translation quality by 10.32 BLEU points over the baseline.

In addition, in order to investigate how similar the matched TM translations tm_t are to the reference translations ref, we also measured the BLEU scores of the matched TM translations against the reference translations. The results are also shown in Table 3, indicated as TM.

3.3 Analysis

We further took a deep look into how the TM gate is varying when we incorporate TM matches with different fuzzy match scores. As a comparison, we used the reference translations as the matched TM translations and incorporated them into NMT-GTM to check the changes of the gate. The BLEU scores measured when we used reference translations as matched TM translations as well as average gate values are shown in Table 4. The results demonstrate that when the matched TM is semantically closer to the current source sentence, the TM gate is larger, indicating that more information from the matched TM translation is used to guide the decoder.

Table 5 shows an example from our test set. The highlighted fragments of the source sentence and the matched TM source sentence are not actually the same in terms of their surface forms. However, they are semantically close and can be translated into the same target translation. Our proposed NMT-GMT is able to successfully incorporate the translation of such a fragment into the decoder.

4 Related Work

Various strategies have been proposed to combine TM and SMT (Koehn and Senellart, 2010; He et al., 2010). Their key ideas are to integrate the translations of the same fragments from TM into SMT, and let SMT only translate those different parts. In order to better model this process, Wang et al. (2013, 2014) use different features to allow relevant TM information to guide SMT decoding.

---

[2] Available at: https://fasttext.cc/

src	主席 说 ， 津巴布韦 代表 根据 议事规则 43 条 要求 参加 该项 目的 讨论
ref	the chairman said that the representative of zimbabwe asked to participate in the discussion of the item in accordance with rule 43 of the rules of procedure .
tm_s	主席 说 ， 塞尔维亚 代表 请求 依据 议事规则 第 43 条 参与 讨论 项目
tm_t	the chairman said that the representative of serbia had asked to participate in the discussion of the item in accordance with rule 43 of the rules of procedure .
RNNSearch	the chairman said that the representative of zimbabwe, in accordance with rule 43, requested a discussion of the item .
NMT-GTM	the chairman said that the representative of zimbabwe had asked to participate in the discussion of the item in accordance with rule 43 of the rules of procedure .

Table 5: A translation example from the test set. Semantically similar fragments are highlighted with red color.

The related work on combining TM and NMT is quite limited. Gu et al. (2017) propose a TM-NMT model that first finds the most similar segments through search engines according to fuzzy match scores and saves them as key-value pairs in memory. In the subsequent decoding, the saved information is used to help decoding. Our work is significantly different from theirs in two aspects. First, we use semantic similarity based on sentence embeddings to detect the best TM matches rather than the fuzzy match score. Second, we encode the entire TM matched translation rather than segments into NMT with coupled encoders and a gating network.

Our work is also related to multi-source NMT (Zoph and Knight, 2016). The difference is that in our case, the multiple source inputs are just semantically similar, rather than identical. This is the reason that we use a gate to combine these inputs.

5 Conclusion and Future work

In this paper, we have presented a novel gated method to encode translation memory into NMT so as to convey the information of the matched TM translation into the NMT decoder. Extensive experiments verify that our method can indeed effectively improve translation quality, especially when fuzzy match scores are higher than 50%. Further analysis reveals that the proposed TM gate is able to vary according to the similarity between the matched TM translation and the current sentence.

Acknowledgments

The present research was supported by the National Natural Science Foundation of China (Grants No. 61622209 and 61861130364). We would like to thank three anonymous reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Michael Bloodgood and Benjamin Strauss. 2015. Translation memory retrieval methods. *Computer Science*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2017. Search Engine Guided Non-Parametric Neural Machine Translation. *arXiv preprint arXiv:1705.07267*.

Yifan He, Yanjun Ma, Josef van Genabith, and Andy Way. 2010. Bridging SMT and TM with translation recommendation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 622–630. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn and Jean Senellart. 2010. Convergence of translation memory and statistical machine translation. In *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, pages 21–31.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of* *the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Alexandre Rafalovitch, Robert Dale, et al. 2009. United nations general assembly resolutions: A six-language parallel corpus. In *Proceedings of the MT Summit*, volume 12, pages 292–299.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *Transactions of the Association of Computational Linguistics*, 5(1):87–99.

Kun Wang, Chengqing Zong, and Keh-Yih Su. 2013. Integrating translation memory into phrase-based machine translation during decoding. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 11–21.

Kun Wang, Chengqing Zong, and Keh-Yih Su. 2014. Dynamically integrating cross-domain translation memory into phrase-based machine translation during decoding. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 398–408.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of NAACL-HLT*, pages 30–34.

Automatic Post-Editing of Machine Translation: A Neural Programmer-Interpreter Approach

Thuy-Trang Vu
Faculty of Information Technology
Monash University, Austrlia
`vuth0001@student.monash.edu`

Gholamreza Haffari
Faculty of Information Technology
Monash University, Austrlia
`first.last@monash.edu`

Abstract

Automated Post-Editing (PE) is the task of automatically correcting common and repetitive errors found in machine translation (MT) output. In this paper, we present a neural programmer-interpreter approach to this task, resembling the way that humans perform post-editing using discrete edit operations, which we refer to as programs. Our model outperforms previous neural models for inducing PE programs on the WMT17 APE task for German-English up to +1 BLEU score and -0.7 TER scores.

1 Introduction

Automatic post-editing (APE) is the automated task that aims to correct common systematic and repetitive errors found in machine translation (MT) output. APE systems can also be used to adapt general-purpose MT output to specific domains without re-training MT models, or to incorporate information which is not available or expensive to compute at MT decoding stage. Post-editing is considered as the modification process of a machine translated text with a minimum labor effort rather than re-translation from scratch.

Previous studies in neural APE have primarily concentrated on formalizing APE as a monolingual MT problem in the target language, with or without conditioning on the source sentence (Pal et al., 2016; Chatterjee et al., 2017). MT approach has suffered from *over-correction* where APE system performs unnecessary correction leading to paraphasing and the degradation of the output quality (Bojar et al., 2016, 2017).

Recent works (Libovický et al., 2016; Berard et al., 2017) have attempted to learn the predict of sequence of post-editing operations, e.g. insertion and deletion, to induce APE *programs* to turn the machine translated text into the desired output. Previous program induction approaches suffer from *over-cautiousness*, where the APE system tends to keep the machine translated text without any modification (Bojar et al., 2017).

In this paper, we propose a programmer-interpreter approach to the APE task to address the over-cautiousness problem. Our architecture includes an interpreter module, which executes the previous editing action before generating the next one. This is in contrast to the previous work, where the *full* program is induced before it is *executed*. The ability of execution *immediately* at every time step provides a proper conditioning context based on the actual partial edited sentence to assist better prediction of the next operation. Moreover, the execution module can be pre-trained on monolingual target text, enabling our architecture to benefit from monolingual data in addition to PE data, which is hard to obtain. Our model is jointly trained on translation task and APE program induction task. The multi-task architecture allows the model to reconstruct the source-target alignment of the black-box MT system and inject it into post-editing task.

We compare our programmer-interpreter architecture against previous works on the English-German APE based on the data for this task in WMT16 and WMT17. Compared to the previous work on APE program induction, our architecture achieves improvements up to +1 BLEU and -0.7 TER scores. Our analysis also shows that APE programs generated by our model are not only better at correcting errors but also attempt to perform more editing actions.

2 Related Work

Pal et al. (2016) has applied the SEQ2SEQ model to APE. Their monolingual MT learned to post-edit English-Italian Google Translation output and

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3048–3053
Brussels, Belgium, October 31 - November 4, 2018.

Source	Does not have a menu bar .
MT	Weist$_1$ keine$_2$ Menüleiste$_3$ angezeigt$_4$.$_5$
Reference	Weist $_1$ keine$_2$ Menüleiste$_3$ auf$_4$.$_5$
APE prog.	*KEEP$_1$ KEEP$_2$ KEEP$_3$ auf $_4$ DEL$_4$ KEEP$_5$ STOP*

Figure 1: An example of PE program when executing on MT would return the PE reference. The color denotes the alignment between MT, reference PE and the APE program. The number subscript shows the edit position in original MT sentence.

was able to reduce the preposition related errors. Blindly performing edition over MT output, the monolingual APE has difficulty to correct missing word or information in the source sentence. Neural multi-source MT architectures are applied to better capture the connection between the source sentence/machine translated text and the PE output (Libovický et al., 2016; Varis and Bojar, 2017; Junczys-Dowmunt and Grundkiewicz, 2017). Chatterjee et al. (2017) ensemble several different models including monolingual MT (TGT→PE), bilingual MT (SRC→PE), and multi-source (SRC,TGT→PE). Libovický et al. (2016); Berard et al. (2017) have proposed learning to predict the sequence of edit operations, aka the program, to produce the post-editing sentence (c.f. §3).

Our work is motivated by Ling et al. (2017) on learning to indirectly solve an algebraic word problem by inducing a program which generates the answer together with an explanation. It further builds up on recent work on neural programmer-interpreter (Reed and De Freitas, 2016), where a neural network programmer learns to program an interpreter. The architecture is then trained using expert action trajectories as programs.

3 The NPI-APE Approach

Given a source sentence $\boldsymbol{s}$ and a machine translated sentence $\boldsymbol{m}$, the goal is to find a post-edited sentence $\boldsymbol{t} = \arg\max_{\boldsymbol{t}'} P_{\text{ape}}(\boldsymbol{t}'|\boldsymbol{m}, \boldsymbol{s})$ where $P_{\text{ape}}(.)$ is our probabilistic APE model. In our proposed approach, we aim to find an editing action sequence $\boldsymbol{z}$ to *execute* in order to generate the desired post-edited sentence,

$$P_{\text{ape}}(\boldsymbol{t}|\boldsymbol{m}, \boldsymbol{s}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} P_{\text{ape}}(\boldsymbol{t}, \boldsymbol{z}|\boldsymbol{m}, \boldsymbol{s}).$$

We decompose the joint probability of a program and an output as:

$$P_{\text{ape}}(\boldsymbol{z}, \boldsymbol{t}|\boldsymbol{m}, \boldsymbol{s}) = \prod_{i=1}^{|\boldsymbol{z}|} P_{\text{prog}}(z_i|\boldsymbol{t}_{\leq j_{i-1}}, \boldsymbol{m}, \boldsymbol{s}) \quad (1)$$

$$\times P_{\text{intp}}(t_{j_i}|m_{k_i}, z_i) \quad (2)$$

where $P_{\text{prog}}(z_i|\boldsymbol{t}_{\leq j_{i-1}}, \boldsymbol{m}, \boldsymbol{s})$ is the programmer's probability in producing the next edit operation z_i given the post edited output $\boldsymbol{t}_{\leq j_{i-1}}$ generated from the operations so far $\boldsymbol{z}_{\leq i-1}$, and $P_{\text{intp}}(t_{j_i}|m_{k_i}, z_i)$ is the interpreter's probability of outputing t_{j_i} given the edit operation z_i and the MT word m_{k_i}.

Following Berard et al.(2017), our action sequence is performed on the MT sentence from left to right. At each position, we can take one of the following editing operations: (i) KEEP to keep the word and go to the next word, (ii) DELETE to delete the word and go to the next word, (iii) INSERT(WORD) to insert a new WORD and stay in that position, or (iv) STOP to terminate the process. In other words, the size of the operation set equals the size of the target vocabulary plus three, where we add the symbols KEEP, DELETE, and STOP as new tokens. Furthermore, j_i is the number of KEEP and INSERT(WORD) operations, and k_i is the number of KEEP and DELETE operations in the sequence of operations $\boldsymbol{z}_{\leq i}$. This *hard attention* mechanism is the outcome of the semantics of the operations, and injects task knowledge into the model. Moreover, $P_{\text{intp}}(t|m, z)$ is 1 if the output word t is consistent with performing the operation z on m, and zero otherwise.

Our decomposition of the joint probability of a program and post-edited output is distinguished from that proposed in (Berard et al., 2017), $P_{\text{ape}}(\boldsymbol{t}, \boldsymbol{z}|\boldsymbol{m}, \boldsymbol{s}) = P_{\text{intp}}(\boldsymbol{t}|\boldsymbol{z}, \boldsymbol{m}) P_{\text{prog}}(\boldsymbol{z}|\boldsymbol{m}, \boldsymbol{s})$. Crucially, in our decomposition (eqns 1 and 2), the programming and interpreting are interleaved at each position, whereas in (Berard et al., 2017) the programming is fully done before the interpretation phase and they are independent.

3.1 Neural Architecture and Joint Training

The architecture consists of three components (i) A SEQ2SEQ model to translate the source sentence to the target in the forced-decoding mode (MT), (ii) A SEQ2SEQ model to incrementally generate the sequence of edit operations (Action Generator), and (iii) An RNN to summarize the post edited sequence of words produced from the execution of actions generated so far (Interpreter).

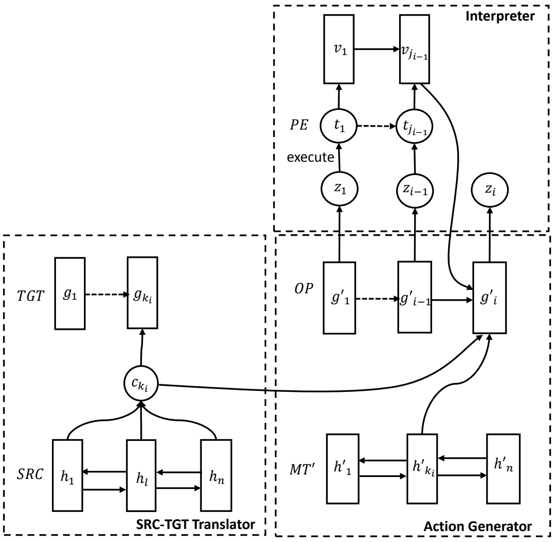

Figure 2: Our proposed NPI-APE model

The encoder and decoder of the MT component are a bidirectional and unidirectional LSTM, whose states are denoted by $\boldsymbol{h}_l$ and $\boldsymbol{g}_k$, respectively. Similarly, the encoder and decoder of the AG (Action Generator) component are a bidirectional and unidirectional LSTM, whose states are denoted by $\boldsymbol{h}'_k$ and $\boldsymbol{g}'_i$, respectively. The states of the unidirectional RNN in the interpreter are denoted by $\boldsymbol{v}_j$.

The next edit operation z_i is generated from the decoder state of the AG $\boldsymbol{g}'_i$ (see Figure 2), which is computed from the previous state $\boldsymbol{g}'_{i-1}$ and a context including the following: (i) $\boldsymbol{h}'_{k_i}$ where k_i is the index of the MT output word currently processed, (ii) $\boldsymbol{c}_{k_i}$ which is the context vector from the MT component when generating the current target word, and (iii) $\boldsymbol{v}_{j_{i-1}}$ which is the last hidden state of the interpreter RNN encoding the post edited sentence generated so far.

The model is trained *jointly* for the translation task (SRC→TGT), and for the post editing task (SRC,TGT→OP,PE). For the training data, we compute the lowest-cost sequence of editing operations (OP) using dynamic programming, where the cost of insertion and deletion are 1.

4 Experiments

Dataset. We evaluate the proposed approach on the English-to-German (En-De) post-editing task in the IT domain using the data from WMT16[1] and WMT17.[2] The official WMT'16 and WMT'17 dataset contains 12K and 11K post-editing triplets (English, translated German, post-edited German) respectively in IT domain. We concatenated them to an 23K triplets. A synthetic corpus of 500K triplets (Junczys-Dowmunt and Grundkiewicz, 2016) is also available as additional training data. We performed our experiment in two different settings with and without synthetic data for comparison with Berard et al. (2017).

The RNN in the interpreter component can be thought of as a language model. This paves the way to pre-train it using monolingual text. We collect in-domain IT text from OPUS[3] from the following sections: GNOME, KDE, KDEdoc, OpenOffice, OpenOffice3, PHP and Ubuntu. After tokenizing, filtering out sentences containing special characters, and removing duplications, we obtain around 170K sentences.

Setup. There are three components in our architecture: machine translation (MT), action generator (AG), and interpreter (LM). We compare our MT+AG+LM architecture against MT+AG[4] Berard et al. (2017) which does not have the LM component. The size of the hidden dimensions (LSTMs in the MT and AG, and simple RNN in the LM component) as well as word embedding in these models is set to 128.

Furthermore, we compare against monolingual SEQ2SEQ (TGT→PE) as well as the multi-source SEQ2SEQ (SRC+TGT→PE) (Varis and Bojar, 2017). Monolingual SEQ2SEQ (TGT→PE) model is an attentional SEQ2SEQ model (Bahdanau et al., 2015) that takes target sentence as input and outputs desired PE sentence. In multi-source SEQ2SEQ (SRC+TGT→PE), we use two encoders for source and target sentences and concatenate their context vectors. In both models, the encoder and decoder contain a single layer of bidirectional and unidirectional LSTM respectively. The size of the LSTM hidden dimensions and word embedding in these models is set to 256 and 128, respectively. This ensures almost the same number of parameters (~13M) in all architectures.

Training. We use a multi-task scenario to jointly train the parameters of the components in MT+AG as well as MT+AG+LM models. For the latter, we warm start the embedding of the target words with

---

[1] http://www.statmt.org/wmt16/ape-task.html

[2] http://www.statmt.org/wmt17/ape-task.html

[3] http://opus.nlpl.eu/

[4] The AG decoder in MT+AG conditions a state on the last generated action as well.

		dev		test2016		test2017	
	Model	TER	BLEU	TER	BLEU	TER	BLEU
	Original MT	24.81	62.92	24.76	62.11	24.48	62.49
12K	TGT → PE	63.76	21.32	60.96	22.11	65.13	18.13
	SRC+TGT → PE	51.41	34.04	48.27	35.24	50.98	31.52
	MT+AG	23.74	65.95	23.53	65.22	23.77	64.34
	MT+AG+LM	**23.36**†	**66.24**	**23.24**†	**65.53**†	**23.45**†	**64.65**†
500K+12K	TGT → PE	50.91	30.88	48.62	32.55	52.07	27.98
	SRC+TGT → PE	30.97	53.97	30.20	53.92	32.82	50.30
	MT+AG	22.82	66.51	22.87	65.67	23.58	64.35
	MT+AG+LM	**22.67**	**67.17**†	**22.53**†	**66.30**†	**23.03**†	**65.31**†
23K	TGT → PE	57.02	27.87	55.52	27.8	60.06	22.78
	SRC+TGT → PE	38.06	47.42	36.61	47.93	39.86	43.52
	MT+AG	23.10	66.60	22.82	66.15	23.14	65.19
	MT+AG+LM	**22.61**†	**67.19**†	**22.42**†	**66.53**†	**22.84**†	**65.52***†
500K+23K	TGT → PE	48.89	34.29	47.12	34.75	51.13	29.59
	SRC+TGT → PE	28.61	57.47	27.79	57.61	30.34	53.26
	MT+AG	22.38	67.34	22.14	66.53	22.71	65.34
	MT+AG+LM	**21.99***†	**67.50***	**22.07***	**66.67***	**22.58***	**65.50**

Table 1: TER and BLEU scores of our model (MT+AG+LM) v.s. the rest on various data conditions for the EN-DE post-editing task. **bold**: Best results within a data condition; *: Best results across data conditions
; †: Statistically significant compared to MT+AG with pvalue ≤ 0.05.

those obtained from training the LM component on the monolingual text[5].

All models are trained with SGD, where the learning rate is initialised to 1 and decays after each epoch. The learning rate is decayed 0.8 after every epoch for model trained with official post-editing data, and 0.5 every half epoch for model with synthetic data. All models use the same vocabulary on the same data condition. The Vocabulary size is 30K for large dataset experiments, and 27K/19K for the 23K/12K data conditions. In all experiments, the best model is selected based on TER on the validation set. For decoding, we use beam search with the beam size of 10.

4.1 Results

Table 1 shows the result on different training datasets to compare our model against the baselines. Original MT is the strong standard *do-nothing* baseline, i.e. copying the MT translation as the PE output. In all settings, our MT+AG+LM models outperforms the MT+AG and monolingual/multi-source SEQ2SEQ models. Specifically, our model outperform MT+AG in 500K+12K training condition by almost 1 BLEU score on test2017.

As expected, the models trained on 23K data perform better than those trained on 12K; further gains are obtained by adding 500K synthetic data.

		Sentences		Actions	
	Model	Mod	Prec.	Mod.	Prec.
dev	MT+AG_23K	546	61.17%	986	60.45%
	MT+AG+LM_23K	558	65.59%	1098	65.21%
	MT+AG_500+23K	673	61.81%	1477	62.02%
	MT+AG+LM_500+23K	682	65.84%	1560	68.21%
test2016	MT+AG_23K	1039	60.92%	1766	62.63%
	MT+AG+LM_23K	1041	66.09%	1919	65.61%
	MT+AG_500+23K	1269	63.04%	2728	63.71%
	MT+AG+LM_500+23K	1251	63.87%	2814	63.50%
test2017	MT+AG_23K	897	55.85%	1535	57.79%
	MT+AG+LM_23K	952	61.24%	1853	60.44%
	MT+AG_500+23K	1182	54.99%	2599	55.71%
	MT+AG+LM_500+23K	1180	56.78%	2653	59.67%

Table 2: Sentence precision and action precision of models trained on 23K and 500K+23K dataset.

Interestingly, training MT+AG and MT+AG+LM models on 23K data lead to better TER/BLEU than those trained on 500K+12K. This implies the importance of in-domain training data, as the synthetic corpus is created using general domain *Common-Crawl* corpus.

4.2 Analysis

We perform fine-grained analysis of the changes made by our model vs MT+AG in order to understand the sources of improvements. For different data conditions, Table 2 shows the number of modified sentences by each model as well as the sentence level precision defined as the fraction of sentences with improved TER. Moreover, it reports the total number of actions generated by the model on sentences with improved TER, as well as the precision of such actions, i.e. the fraction of those observed in the ground truth action trajecto-

---

[5] The MT, AG, and LM components share the target word embedding table in our model. Similarly, MT and AG components share the target embedding in the MT+AG model.

ries.

As reported in Berard et al. (2017), one major challenge of predicting action sequences is the class imbalance. The model is often too conservative about its edits, and tends to use the KEEP far more than the INSERT and DELETE actions. Our Programmer-Interpreter model tackles this problem, as evidenced by its comparable number of modified sentences, but with higher sentence and action level precision in almost all cases.

5 Conclusion

In this paper, we have presented a neural programmer-interpreter approach to automated post-editing of MT output. Our approach interleaves generating the sequence of edit actions by a programmer component, and executing those actions with an interpreter component. This leads to better capturing the history of the past generated actions when generating the next action. Our approach achieves up to +1 BLEU and -.7 TER improvement compared to a variant in which programming is not interleaved with execution. Future work includes inducing macro-actions composed of simpler building block actions.

Acknowledgments

The authors are grateful to the anonymous reviewers for their helpful comments and corrections. This work was supported by the Multi-modal Australian ScienceS Imaging and Visualisation Environment (MASSIVE) (www. massive.org.au), and partially supported by a Google Faculty Award to GH and the Australian Research Council through DP160102686.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations (ICLR) 2015*.

Alexandre Berard, Laurent Besacier, and Olivier Pietquin. 2017. LIG-CRIStAL submission for the WMT 2017 automatic post-editing task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 623–629, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Rajen Chatterjee, M. Amin Farajian, Matteo Negri, Marco Turchi, Ankit Srivastava, and Santanu Pal. 2017. Multi-source neural automatic post-editing: FBK's participation in the WMT 2017 APE shared task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 630–638, Copenhagen, Denmark. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation*, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2017. An exploration of neural sequence-to-sequence architectures for automatic post-editing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 120–129. Asian Federation of Natural Language Processing.

Jindřich Libovický, Jindřich Helcl, Marek Tlustý, Ondřej Bojar, and Pavel Pecina. 2016. CUNI system for WMT16 automatic post-editing and multimodal translation tasks. In *Proceedings of the First Conference on Machine Translation*, pages 646–654, Berlin, Germany. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Santanu Pal, Sudip Kumar Naskar, Mihaela Vela, and Josef van Genabith. 2016. A neural network based approach to automatic post-editing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 281–286.

Scott Reed and Nando De Freitas. 2016. Neural programmer-interpreters. In *Proceedings of International Conference on Learning Representations (ICLR) 2016*.

Dusan Varis and Ondřej Bojar. 2017. CUNI system for WMT17 automatic post-editing task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 661–666, Copenhagen, Denmark. Association for Computational Linguistics.

Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation

Yilin Yang[1] **Liang Huang**[1,2] **Mingbo Ma**[1,2]
[1]Oregon State University
Corvallis, OR, USA
[2]Baidu Research
Sunnyvale, CA, USA
{yilinyang721, liang.huang.sh, cosmmb}@gmail.com

Abstract

Beam search is widely used in neural machine translation, and usually improves translation quality compared to greedy search. It has been widely observed that, however, beam sizes larger than 5 hurt translation quality. We explain why this happens, and propose several methods to address this problem. Furthermore, we discuss the optimal stopping criteria for these methods. Results show that our hyperparameter-free methods outperform the widely-used hyperparameter-free heuristic of length normalization by +2.0 BLEU, and achieve the best results among all methods on Chinese-to-English translation.

1 Introduction

In recent years, neural machine translation (NMT) has surpassed traditional phrase-based or syntax-based machine translation, becoming the new state of the art in MT (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2014). While NMT training is typically done in a "local" fashion which does not employ any search (bar notable exceptions such as Ranzato et al. (2016), Shen et al. (2016), and Wiseman and Rush (2016)), the decoding phase of all NMT systems universally adopts beam search, a widely used heuristic, to improve translation quality.

Unlike phrase-based MT systems which enjoy the benefits of very large beam sizes (in the order of 100–500) (Koehn et al., 2007) , most NMT systems choose tiny beam sizes up to 5; for example, Google's GNMT (Wu et al., 2016) and Facebook's ConvS2S (Gehring et al., 2017) use beam sizes 3 and 5, respectively. Intuitively, the larger the beam size is, the more candidates it explores, and the better the translation quality should be. While this definitely holds for phrase-based MT systems, surprisingly, it is *not* the case for NMT: many researchers observe that translation quality degrades with beam sizes beyond 5 or 10 (Tu et al., 2017; Koehn and Knowles, 2017). We call this phenomenon the *"beam search curse"*, which is listed as one of the six biggest challenges for NMT (Koehn and Knowles, 2017).

However, there has not been enough attention on this problem. Huang et al. (2017) hint that length ratio is the problem, but do not explain why larger beam sizes cause shorter lengths and worse BLEU. Ott et al. (2018) attribute it to two kinds of "uncertainties" in the training data, namely the *copying* of source sentence and the *non-literal* translations. However, the first problem is only found in European language datasets and the second problem occurs in all datasets but does not seem to bother pre-neural MT systems. Therefore, their explanations are not satisfactory.

On the other hand, previous work adopts several heuristics to address this problem, but with various limitations. For example, RNNSearch (Bahdanau et al., 2014) and ConvS2S use *length normalization*, which (we will show in Sec. 6) seems to somewhat alleviate the problem, but far from being perfect. Meanwhile, He et al. (2016) and Huang et al. (2017) use word-reward, but their *reward* is a hyper-parameter to be tuned on dev set.

Our contributions are as follows:

- We explain why the beam search curse exists, supported by empirical evidence (Sec. 3).
- We review existing rescoring methods, and then propose ours to break the beam search curse (Sec. 4). We show that our hyperparameter-free methods outperfrom the previous hyperparameter-free method (length normalization) by +2.0 BLEU (Sec. 6).
- We also discuss the stopping criteria for our rescoring methods (Sec. 5). Experiments

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3054–3059
Brussels, Belgium, October 31 - November 4, 2018.

show that with optimal stopping alone, the translation quality of the length normalization method improves by +0.9 BLEU.

After we finish our paper, we became aware of a parallel work (Murray and Chiang, 2018) that also reveals the same root cause we found for the beam search curse: the length ratio problem.

2 Preliminaries: NMT and Beam Search

We briefly review the encoder-decoder architecture with attention mechanism (Bahdanau et al., 2014). An RNN encoder takes an input sequence $\mathbf{x} = (x_1, ..., x_m)$, and produces a sequence of hidden states. For each time step, the RNN decoder will predict the probability of next output word given the source sequence and the previously generated prefix. Therefore, when doing greedy search, at time step i, the decoder will choose the word with highest probability as y_i. The decoder will continue generating until it emits `</eos>`. In the end, the generated hypothesis is $\mathbf{y} = (y_1, ..., y_n)$ with $y_n =$ `</eos>`, with model score

$$S(\mathbf{x}, \mathbf{y}) = \textstyle\sum_{i=1}^{|\mathbf{y}|} \log p(y_i \mid \mathbf{x},\, y_{1..\{i-1\}}) \qquad (1)$$

As greedy search only explores a single path, we always use beam search to improve search quality. Let b denote the beam size, then at step i the beam B_i is an *ordered* list of size b:

$$B_0 = [\langle \texttt{<s>},\ p(\texttt{<s>} \mid \mathbf{x})\rangle]$$

$$B_i = \overset{b}{\mathbf{top}}\{\langle \mathbf{y}' \circ y_i,\ s \cdot p(y_i | \mathbf{x}, \mathbf{y})\rangle \mid \langle \mathbf{y}', s\rangle \in B_{i-1}\}$$

In the most naive case, after reaching the maximum length (a hard limit), we get N possible candidate sequences $\{\mathbf{y}_1, ..., \mathbf{y}_N\}$. The *default* strategy chooses the one with highest model score. We will discuss more sophistcated ways of stopping and choosing candidates in later sections.

3 Beam Search Curse

The most popular translation quality metric, BLEU (Papineni et al., 2002), is defined as:

$$\text{BLEU} = bp \cdot \exp(1/4 \textstyle\sum_{n=1}^{4} \log p_n) \qquad (2)$$

$$\text{where } \ bp = \min\{e^{1-1/lr}, 1\} \qquad (3)$$

$$\text{where } \ lr = |\mathbf{y}| / |\mathbf{y}^*| \qquad (4)$$

Here p_n are the n-gram precisions, and $|\mathbf{y}|$ and $|\mathbf{y}^*|$ denote the hypothesis and reference lengths, while bp is the *brevity penalty* (penalizing short translations) and lr is the *length ratio* (Shi et al., 2016; Koehn and Knowles, 2017), respectively.

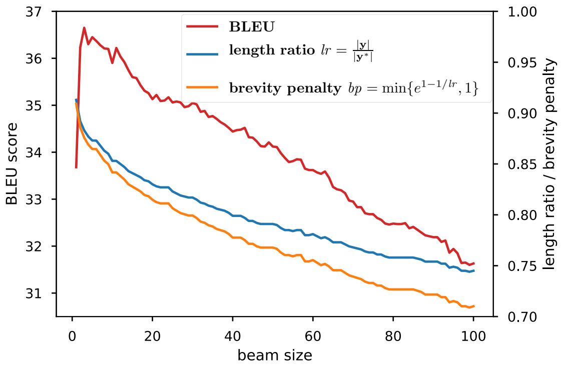

Figure 1: As beam size increases beyond 3, BLEU score on the dev set gradually drops. All terms are calculated by *multi-bleu.pl*.

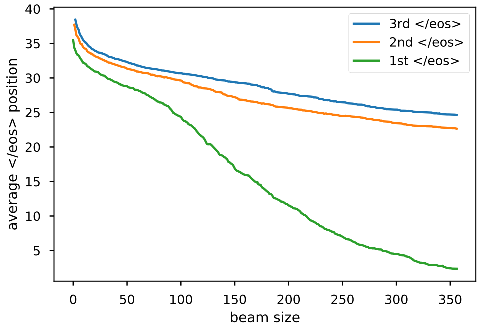

Figure 2: Searching algorithm with larger beams generates `</eos>` earlier. We use the average first, second and third `</eos>` positions on the dev set as an example.

With beam size increasing, $|\mathbf{y}|$ decrases, which causes the length ratio to drop, as shown in Fig. 1. Then the brevity penalty term, as a function of the length ratio, decreases even more severely. Since bp is a key factor in BLEU, this explains why the beam search curse happens.[1]

The reason why $|\mathbf{y}|$ decreases as beam size increases is actually twofold:

1. As beam size increases, the more candidates it could explore. Therefore, it becomes easier for the search algorithm to find the `</eos>` symbol. Fig. 2 shows that the `</eos>` indices decrease steadily with larger beams.[2]

---

[1] The length ratio is *not* just about BLEU: if the hypothesis length is only 75% of reference length, something that should have been translated must be missing; i.e., bad adequacy. Indeed, Murray and Chiang (2018) confirm the same phenomenon with METEOR.

[2] Pre-neural SMT models, being probabilistic, also favor short translations (and derivations), which is addressed by word (and phrase) reward. The crucial difference between SMT and NMT is that the former stops when covering the whole input, while the latter stops on emitting `</eos>`.

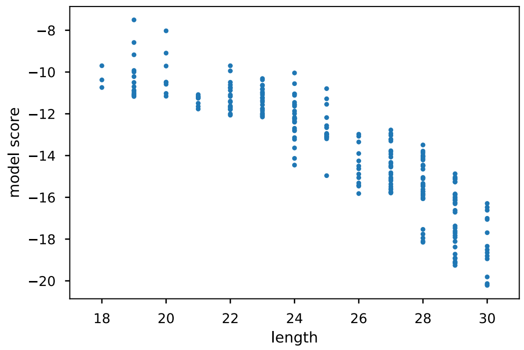

Figure 3: Candidate lengths vs. model score. This scatter plot is generated from 242 finished candidates when translated from one source sequence with beam size 80.

2. Then, as shown in Fig. 3, shorter candidates have clear advantages *w.r.t.* model score.

Hence, as beam size increases, the search algorithm will generate shorter candidates, and then prefer even shorter ones among them.[3]

4 Rescoring Methods

We first review existing methods to counter the length problem and then propose new ones to address their limitations. In particular, we propose to predict the target length from the source sentence, in order to choose a hypothesis with proper length.

4.1 Previous Rescoring Methods

RNNSearch (Bahdanau et al., 2014) first introduces the *length normalization* method, whose score is simply the average model score:

$$\hat{S}_{\text{length\_norm}}(\mathbf{x}, \mathbf{y}) = S(\mathbf{x}, \mathbf{y})/|\mathbf{y}| \qquad (5)$$

This is the most widely used rescoring method since it is hyperparameter-free.

GNMT (Wu et al., 2016) incorporates length and coverage penalty into the length normalization method, while also adding two hyperparameters to adjust their influences. (please check out their paper for exact formulas).

Baidu NMT (He et al., 2016) borrows the *Word Reward* method from pre-neural MT, which gives a reward r to every word generated, where r is a hyperparameter tuned on the dev set:

$$\hat{S}_{\text{WR}}(\mathbf{x}, \mathbf{y}) = S(\mathbf{x}, \mathbf{y}) + r \cdot |\mathbf{y}| \qquad (6)$$

Based on the above, Huang et al. (2017) propose a variant called *Bounded Word-Reward* which only rewards up to an “optimal” length. This length is calculated using a fixed “generation ratio” gr, which is the ratio between target and source sequence length, namely the average number of target words generated for each source word. It gives reward r to each word up to a bounded length $L(\mathbf{x}, \mathbf{y}) = \min\{|\mathbf{y}|, gr \cdot |\mathbf{x}|\}$:

$$\hat{S}_{\text{BWR}}(\mathbf{x}, \mathbf{y}) = S(\mathbf{x}, \mathbf{y}) + r \cdot L(\mathbf{x}, \mathbf{y}) \qquad (7)$$

4.2 Rescoring with Length Prediction

To remove the fixed generation ratio gr from Bounded Word-Reward, we use a 2-layer MLP, which takes the mean of source hidden states as input, to predict the generation ratio $gr^*(\mathbf{x})$. Then we replace the fixed ratio gr with it, and get our predicted length $L_{pred}(\mathbf{x}) = gr^*(\mathbf{x}) \cdot |\mathbf{x}|$.

4.2.1 Bounded Word-Reward

With predicted length, the new predicted bound and final score would be:

$$L^*(\mathbf{x}, \mathbf{y}) = \min\{|\mathbf{y}|, L_{pred}(\mathbf{x})\} \qquad (8)$$

$$\hat{S}_{\text{BWR}^*}(\mathbf{x}, \mathbf{y}) = S(\mathbf{x}, \mathbf{y}) + r \cdot L^*(\mathbf{x}, \mathbf{y}) \qquad (9)$$

While the predicted length is more accurate, there is still a hyperparameter r (word reward), so we design two methods below to remove it.

4.2.2 Bounded Adaptive-Reward

We propose *Bounded Adaptive-Reward* to automatically calculate proper reward based on the current beam. With beam size b, the reward for time step t is the average negative log-probability of the words in the current beam.

$$r_t = -(1/b)\sum_{i=1}^{b} \log p(\text{word}_i) \qquad (10)$$

Its score is very similar to (7):

$$\hat{S}_{\text{AdaR}}(\mathbf{x}, \mathbf{y}) = S(\mathbf{x}, \mathbf{y}) + \sum_{t=1}^{L^*} r_t \qquad (11)$$

4.2.3 BP-Norm

Inspired by the BLEU score definition, we propose *BP-Norm* method as follows:

$$\hat{S}_{bp}(\mathbf{x}, \mathbf{y}) = \log bp + S(\mathbf{x}, \mathbf{y})/|\mathbf{y}| \qquad (12)$$

bp is the same brevity penalty term as in (3). Here, we regard our predicted length as the reference

---

[3] Murray and Chiang (2018) attribute the fact that beam search prefers shorter candidates to the *label bias problem* (Lafferty et al., 2001) due to NMT’s local normalization.

length. The beauty of this method appears when we drop the logarithmic symbol in (12):

$$\begin{aligned}\exp(\hat{S}_{bp}(\mathbf{x},\mathbf{y})) &= bp \cdot \left(\prod_{i=1}^{|\mathbf{y}|} p(y_i|...)\right)^{1/|\mathbf{y}|} \\ &= bp \cdot \exp\left(\frac{1}{|\mathbf{y}|}\sum_{i=1}^{|\mathbf{y}|} \log p(y_i|...)\right)\end{aligned}$$

which is in the same form of BLEU score (3).

5 Stopping Criteria

Besides rescoring methods, the stopping criteria (when to stop beam search) is also important, for both efficiency and accuracy.

5.1 Conventional Stopping Criteria

By default, OpenNMT-py (Klein et al., 2017) stops when the topmost beam candidate stops, because there will not be any future candidates with higher model scores. However, this is not the case for other rescoring methods; e.g., the score of length normalization (5) could still increase.

Another popular stopping criteria, used by RNNSearch (Bahdanau et al., 2014), stops the beam search when exactly b finished candidates have been found. Neither method is optimal.

5.2 Optimal Stopping Criteria

For Bounded Word-Reward, Huang et al. (2017) introduces a provably-optimal stopping criterion that could stop both early and optimally. We also introduce an optimal stopping criterion for BP-Norm. Each time we generate a finished candidate, we update our best score $\hat{S}^\star$. Then, for the topmost beam candidate of time step t, we have:

$$\hat{S}_{bp} = \frac{S_{t,0}}{t} + \min\{1 - \frac{L_{pred}}{t}, 0\} \le \frac{S_{t,0}}{R} \quad (13)$$

where R is the maximum generation length. Since $S_{t,0}$ will drop after time step t, if $\frac{S_{t,0}}{R} \le \hat{S}^\star$, we reach optimality. This stopping criterion could also be applied to length normalization (5).

Meawhile, for Bounded Adaptive-Reward, we can have a similar optimal stopping criterion: If the score of topmost beam candidate at time step $t > L_{pred}$ is lower than $\hat{S}^\star$, we reach optimality.

Proof. The first part of $\hat{S}_{\text{AdaR}}$ in (11) will decrease after time step t, while the second part stays the same when $t > L_{pred}$. So the score in the future will monotonically decrease. ☐

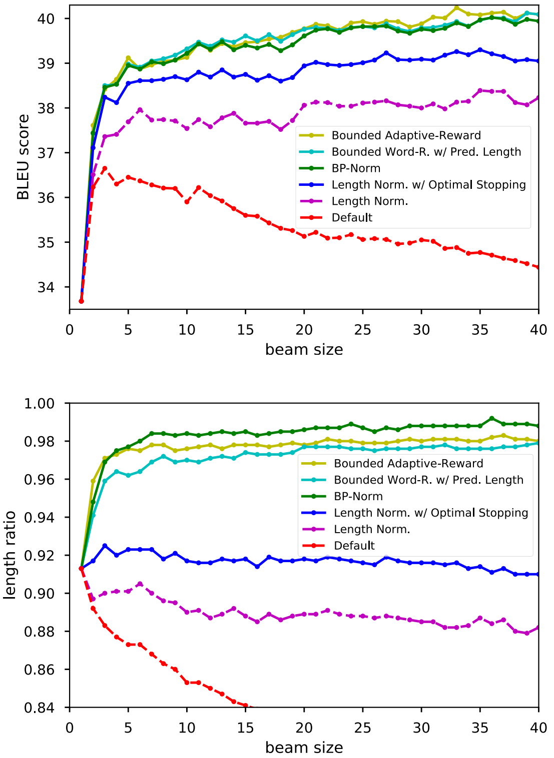

Figure 4: The BLEU scores and length ratios ($lr = |\mathbf{y}|/|\mathbf{y}^*|$) of various rescoring methods.

6 Experiments

Our experiments are on Chinese-to-English translation task, based on the OpenNMT-py codebase.[4] We train our model on 2M sentences, and apply BPE (Sennrich et al., 2015) on both sides, which reduces Chinese and English vocabulary sizes down to 18k and 10k respectively. We then exclude pairs with more than 50 source or target tokens. We validate on NIST 06 and test on NIST 08 (newswire portions only for both). We report case-insensitive, 4 reference BLEU scores.

We use 2-layers bidirectional LSTMs for the encoder. We train the model for 15 epochs, and choose the one with lowest perplexity on the dev set. Batch size is 64; both word embedding and hidden state sizes 500; and dropout 0.3. The total parameter size is 28.5M.

6.1 Parameter Tuning and Results

We compare all rescoring methods mentioned above. For the length normalization method, we also show its results with *optimal stopping*.

For Bounded Word-Reward method with and without our predicted length, we choose the best r on the dev set seperately. The length normal-

---

[4] `https://github.com/OpenNMT/OpenNMT-py`

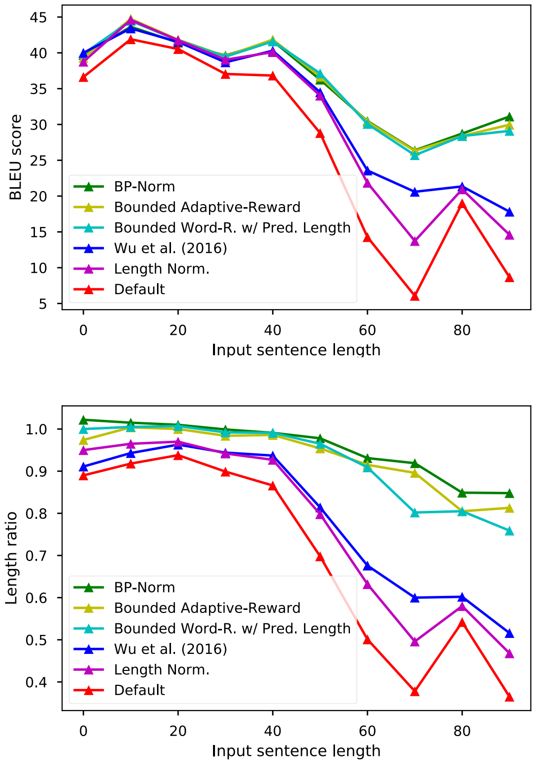

Figure 5: BLEU scores and length ratios on the dev set over various input sentence lengths.

Small beam ($b = 14, 15, 16$)	dev		test	
	BLEU	ratio	BLEU	ratio
Moses (b=70)	30.14	-	29.41	-
Default (b=5)	36.45	0.87	32.88	0.87
Length Norm.	37.73	0.89	34.07	0.89
+ optimal stopping*	38.69	0.92	35.00	0.92
Wu et al. (2016) α=β=0.3	38.12	0.89	34.26	0.89
Bounded word-r. r=1.3	39.22	0.98	35.76	0.98
with predicted length				
Bounded word-r. r=1.4*	**39.53**	0.97	35.81	0.97
Bounded adaptive-reward*	39.44	0.98	35.75	0.98
BP-Norm*	39.35	0.98	**35.84**	0.99

Large beam ($b = 39, 40, 41$)	dev		test	
	BLEU	ratio	BLEU	ratio
Moses (b=70)	30.14	-	29.41	-
Default (b=5)	36.45	0.87	32.88	0.87
Length Norm.	38.15	0.88	34.26	0.88
+ optimal stopping*	39.07	0.91	35.14	0.91
Wu et al. (2016) α=β=0.3	38.40	0.89	34.41	0.88
Bounded word-r. r=1.3	39.60	0.98	35.98	0.98
with predicted length				
Bounded word-r. r=1.4*	40.11	0.98	36.13	0.97
Bounded adaptive-reward*	**40.14**	0.98	**36.23**	0.98
BP-Norm*	39.97	0.99	**36.22**	0.99

Table 1: Average BLEU scores and length ratios over small and large beams. $\star$ indicates our methods.

ization used by Wu et al. (2016) has two hyper-parameters, namely α for length penalty and β for coverage penalty. We jointly tune them on the dev set, and choose the best config. (α=0.3, β=0.3).

Figure 4 show our results on the dev set. We see that our proposed methods get the best performance on the dev set, and continue growing as beam size increases. We also observe that optimal stopping boosts the performance of length normalization method by around +0.9 BLEU. In our experiments, we regard our predicted length as the *maximum generation length* in (13). We further observe from Fig. 5 that our methods keep the length ratio close to 1, and greatly improve the quality on longer input sentences, which are notoriously hard for NMT (Shen et al., 2016).

Table 1 collects our results on both dev and test sets. Without loss of generality, we show results with both small and large beam sizes, which average over b=14,15,16 and b=39,40,41, respectively.

6.2 Discussion

From Table 1, we could observe that with our length prediction model, Bounded word-reward method gains consistent improvement. On the other hand, results from length normalization method show that optimal stopping technique gains significant improvement by around +0.9 BLEU. While with both, our proposed methods beat all previous methods, and gain improvement over hyperparameter-free baseline (i.e. length normalization) by +2.0 BLEU.

Among our proposed methods, Bounded word-reward has the reward r as an hyper-parameter, while the other two methods get rid of that. Among them, we recommend the BP-Norm method, because it is the simplest method, and yet works equally well with others.

7 Conclusions

We first explain why the beam search curse exists and then formalize all previous rescoring methods. Beyond that, we also propose several new methods to address this problem. Results from the Chinese-English task show that our hyperparameter-free methods beat the hyperparameter-free baseline (length normalization) by +2.0 BLEU.

Acknowledgements

Kenton Lee suggested the length prediction idea. This work was partially supported by DARPA N66001-17-2-4030, and NSF IIS-1817231 and IIS-1656051. We thank the anonymous reviewers for suggestions and Juneki Hong for proofreading.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proc. of ICML*.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with smt features. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 151–157. AAAI Press.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *EMNLP*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*, volume 3, page 413.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *CoRR*, abs/1706.03872.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *Proceedings of WMT 2018*.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. *arXiv preprint arXiv:1803.00047*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadephia, USA.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL*.

Xing Shi, Kevin Knight, and Deniz Yuret. 2016. Why neural translations are the right length. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2278–2282.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *AAAI*, pages 3097–3103.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of EMNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Multi-Multi-View Learning:
Multilingual and Multi-Representation Entity Typing

Yadollah Yaghoobzadeh
Microsoft Research
Montreal, Canada
`yayaghoo@microsoft.com`

Hinrich Schütze
LMU Munich
Munich, Germany
`inquiries@cislmu.org`

Abstract

Knowledge bases (KBs) are paramount in NLP. We employ *multiview learning* for increasing accuracy and coverage of entity type information in KBs. We rely on two *metaviews*: language and representation. For language, we consider high-resource and low-resource languages from Wikipedia. For representation, we consider representations based on the context distribution of the entity (i.e., on its embedding), on the entity's name (i.e., on its surface form) and on its description in Wikipedia. The two metaviews language and representation can be freely combined: each pair of language and representation (e.g., German embedding, English description, Spanish name) is a distinct view. Our experiments on entity typing with fine-grained classes demonstrate the effectiveness of multiview learning. We release MVET, a large multiview – and, in particular, multilingual – entity typing dataset we created. Mono- and multilingual fine-grained entity typing systems can be evaluated on this dataset.

1 Introduction

Accurate and complete knowledge bases (KBs) are paramount in NLP. Entity typing, and in particular fine-grained entity typing, is an important component of KB completion with applications in NLP and knowledge engineering. Studies so far have been mostly for English (Yaghoobzadeh and Schütze, 2015), but also for Japanese (Suzuki et al., 2016).

We employ *multiview learning* for increasing accuracy and coverage of entity type information in KBs. We rely on two *metaviews*: language and representation. For language, we take high- and low-resource languages from Wikipedia. For representation, we consider representations based on the context distribution of the entity (i.e., on its embedding), on the entity's name (i.e., on its surface form) and on its description in Wikipedia. The two metaviews language and representation can be freely combined: each pair of language and representation (e.g., German embedding, English description, Spanish name) is a distinct view.

Views are defined as kinds of information about an instance that have three properties (Blum and Mitchell, 1998; Xu et al., 2013). (i) Sufficiency. Each view is sufficient for classification on its own. (ii) Compatibility. The target functions in all views predict the same labels for cooccurring features with high probability. (iii) Conditional independence. The views are conditionally independent given the class label.

As in most cases of multiview learning, these three properties are only approximately true for our problem. (i) Not every view is sufficient for every instance. While a name like "George Washington Bridge" is sufficient for typing the entity as "bridge", the name "Washington" is not sufficient for entity typing. (ii) Cases of incompatibility exist. For example, the "Bering Land Bridge" is not a bridge. (iii) Views have some degree of conditional dependence. For example, if a bridge is a viaduct, not a bridge proper, then the description of the bridge will contain more occurrences of the word "viaduct" than for proper bridges whose name does not contain the word "viaduct".

In summary, we make three main contributions. (i) We formalize entity typing as a multiview problem by introducing two metaviews, language and representation; each combination of instances of these two metaviews defines a distinct view. (ii) We show that this formalization is effective for entity typing as a key task in KB completion: multiview and crossview learning outperform single-view learning by a large margin, especially for rare entities and low-resource languages. (iii) We release MVET (Multiview Entity Typing), a large

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3060–3066
Brussels, Belgium, October 31 - November 4, 2018.

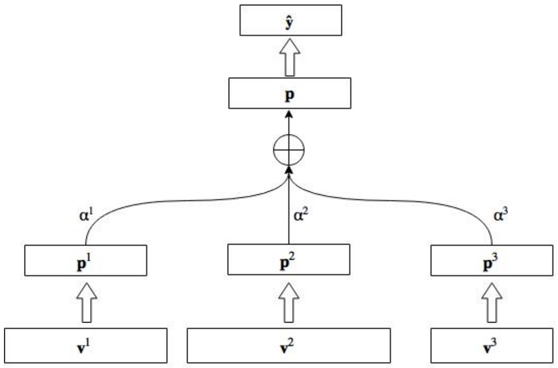

Figure 1: Attention-based multiview learning. View specific representations $\mathbf{v}^j$ of the entity are transformed to a shared space and summed by attention weights α^j into aggregated multiview representation $\mathbf{p}$. A one-hidden-layer perceptron computes output vector $\hat{\mathbf{y}}$.

multiview and, in particular, multilingual dataset, for entity typing[1]. This dataset can be used for mono- and multilingual fine-grained entity typing evaluations. In contrast to prior work on entity typing based on Clueweb (a commercial corpus), all our data can be released publicly because it is based on Wikipedia.

2 Multilingual Multi-Representation Entity Typing

We address the task of entity typing (Yaghoobzadeh and Schütze, 2015), i.e., assigning to a given entity one or more types from a set of types T. E.g., Churchill is a politician and writer.

Our key idea is that we can tap two different information sources for entity typing, which we will refer to as *metaviews*: language and representation. For the language metaview, we consider N languages (English, German, . . .). For the representation metaview, we consider three representations: based on the entity's context distribution, based on its canonical name and based on its description. Each combination of language and representation defines a separate view of the entity, i.e., we have up to $3N$ views. The views are not completely independent of each other: what is written about an entity in English and German is correlated and information derived from the entity name is correlated with its description; see discussion in §1. Still, each view contains information complementary to each other view.

For the *context view* of the representation metaview, we use entity embeddings (Yaghoobzadeh and Schütze, 2015): each mention of an entity in Wikipedia – identified using Wikipedia hyperlinks – is replaced by the entity's unique identifier. We can then run standard embedding learning. For the *name view*, we take the sum of the embeddings of the words of the entity name. The *description view* is based on the entity's Wikipedia page; see §3.

We represent view j of an entity e as the vector or embedding $\mathbf{v}^j \in \mathbb{R}^{d_j}$. We combine these embeddings into a multiview representation $\mathbf{p} \in \mathbb{R}^d$ of entity e. As discussed above, each $\mathbf{v}^j$ contributes potentially complementary information.

After learning $\mathbf{p}$, a one-hidden-layer perceptron computes the type predictions $\hat{\mathbf{y}} \in \mathbb{R}^{|T|}$:

$$\hat{\mathbf{y}} = \sigma\Big(\mathbf{W}_o f\big(\mathbf{W}_h \mathbf{p}\big)\Big) \qquad (1)$$

f is leaky rectifier, $\mathbf{W}_h \in \mathbb{R}^{h \times d}$, $\mathbf{W}_o \in \mathbb{R}^{|T| \times h}$.

The cost function is binary cross entropy summed over types and training examples:

$$\sum_{i,t} (\mathbf{y}_{i,t} \log(\hat{\mathbf{y}}_{i,t}) + (1{-}\mathbf{y}_{i,t})(\log(1{-}\hat{\mathbf{y}}_{i,t}))) \qquad (2)$$

where $\mathbf{y}_{i,t}$ and $\hat{\mathbf{y}}_{i,t}$ are the gold and prediction for type t of example i.

A simple and effective way of computing the representation $\mathbf{p}$ of an entity is what we refer to as **MULTIVIEW-CON**: a concatenation of the n view embeddings, followed by a non-linear transformation:

$$\mathbf{p} = \tanh(\mathbf{W}_1[\mathbf{v}^1; \mathbf{v}^2; ...; \mathbf{v}^n]^\intercal)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times (\Sigma_{j=1}^{n} d_j)}$ is the transformation matrix.

Concatenation may not be effective because some entities have pages in all Wikipedias and thus have $200 \times 3 = 600$ views whereas others occur only in one. Also, the views might have different qualities. Therefore, we consider attention-based weighted average or **MULTIVIEW-ATT** as an alternative to MULTIVIEW-CON. Embeddings $\mathbf{v}^j$ live in different spaces, so we first transform them using language specific matrices $\mathbf{W}^j \in \mathbb{R}^{d \times d_j}$:

$$\mathbf{p}^j = \tanh(\mathbf{W}^j \mathbf{v}^j) \qquad (3)$$

Then, we compute the attention weights:

$$\alpha^j = \mathrm{softmax}(\mathbf{a}^T \mathbf{p}^j)$$

---

[1] Our dataset and code are available at: `http://github.com/yyaghoobzadeh/MVET`

where $\mathbf{a} \in \mathbb{R}^d$ is a vector that is trained to weight the vectors $\mathbf{p}^j$. The **MULTIVIEW-ATT** representation is then defined as:

$$\mathbf{p} = \sum_j \alpha^j \mathbf{p}^j$$

A schematic architecture is shown in Figure 1.

We also experiment with two alternatives. **MULTIVIEW-AVG**: We set all $\alpha^j = 1/n$, i.e., the entity representation is a simple average. **MULTIVIEW-MAX**: We apply per-dimension maxpooling, $\mathbf{p}_i = \max_j \mathbf{p}_i^j$. The idea here is to capture the most significant features across views.

3 Dataset and Experiments

In this section, we first introduce our new dataset and then describe our results.

Multiview entity typing (MVET) dataset. Wikipedia and Freebase are our sources for creation of MVET. We try to map each English Wikipedia article of an entity to Freebase. Freebase types are mapped to 113 FIGER tags (Ling and Weld, 2012). We use Wikipedia interlingual links to build multilingual datasets by identifying corresponding Wikipedia articles in non-English languages. So for each entity, we have the English article name as well as the names in other languages (if they exist) and FIGER types of the entity. We use these multilingual names and Wikipedias to build our representation views as described in §3.

We experiment with ten languages: English (EN), German (DE), Farsi (FA), Spanish (ES); and Arabic (AR), French (FR), Italian (IT). Polish (PL), Portuguese (PT), Russian (RU). The procedure described above gives us around 2M entities. We divide them into train (50%), dev (20%) and test (30%) and, for efficient training, sample them stratified by type to ensure enough entities per type. The final dataset used in our experiments contains about 74k / 35k / 50k train / dev / test entities and 102 FIGER types. Dev is used to optimize model hyperparameters. Appendix A gives some more statistics for MVET.

Learning representation views. We refer to the three instances of the representation metaview (see §1) as **CTXT** (contexts), **NAME** (name) and **DESC** (description).

For learning CTXT embeddings, we train WANG2VEC (Ling et al., 2015) on Wikipedia after having replaced hyperlinked mentions of an entity with its ID. NAME is derived from publicly available 300-dimensional fastText (Bojanowski et al., 2017) embeddings. We use the average of the words in a name as its NAME embedding[2]. If a word does not have a fastText embedding, we apply the fastText model to compute it. So there are no unknown words in our dataset. For DESC, we extract the keywords (using tf-idf) of the first paragraph of the Wikipedia article of an entity. The DESC embedding is the average fastText embedding of the keywords.

To reiterate the complementarity of the three representation views: names are ambiguous, but if we use the names of an entity in different languages, we can mitigate this ambiguity. E.g., “Apple” can refer to an entity or a fruit in English, but only to an entity in French. Similarly, the description of an entity is a high quality textual source to extract information from. The simplest case of complementarity is that not all views are available. An entity can be completely missing from one of the languages; it may not have a description because only a stub is provided; etc.

3.1 Results

Evaluation metric. Following prior work in entity typing (Yaghoobzadeh and Schütze, 2017), we evaluate by micro F_1, a global summary score of all system predictions. Entity frequency is an important variable, so we report results for tail (frequency <10, n=35,533), head (frequency >100, n=2,638) and all entities.

Table 1 shows results for entity typing on our dataset, MVET. We start with FIGMENT (Yaghoobzadeh and Schütze, 2017) baseline results on MVET dataset (line 0), which is the state-of-the-art system in entity typing. FIGMENT is equivalent to our MULTIVIEW-CON model with only English-CTXT, -NAME and -DESC representations.

Lines 1–4, 9–12, 17–20 are singleview results, e.g., F_1 for tail is 62.0 for the English-CTXT view. Lines 5–8, 13–16, 21–24 combine the four languages; so these are multiview results for the language metaview. All four multiview models are better than the corresponding singleview models in the same block. Lines 25–28 show results for the combination of the two metaviews; a total of

---

[2] Some of the Wikipedia titles contain a category inside parentheses, e.g.., “Washington (state)”. We remove these parentheses and their content from the titles, if they exist, and then use the titles as our names.

twelve views are combined (four languages times three representations). The multi-multi-view models on lines 25–28 outperform all other results. Comparing line 25 and FIGMENT (line 0), adding representations from three more languages result in .5%, .4%, .9% improvements for all, tail and head entities. Line 26 by using ATT improves the results further especially for the tail entities. These results confirm the effectiveness of our contributions: adding language as a metaview, and using ATT instead of CON to combine multiple views.

Lines 29–32 show results for using NAME representations in six additional languages: AR, FR, IT, PL, PT, RU. F_1 is up to more than one percent better than on lines 13–16. This demonstrates the benefit of using more languages – although the effect is limited since only the long tail of entities can improve.

Lines 33–36 vs. lines 25–28 make the same comparison (NAME4 vs. NAME10) for multi-multi-view. By ATT, we get a small improvement for tail, but not for head (line 34 vs. line 26). Apparently, there is noise added by considering more languages and this hurts the results for head entities.

A general tendency is that ATT performs better compared to MAX, AVG and CON as the number of views increases (lines 30 and 34) and so the average number of views without information (i.e., missing views) for an entity increases. In contrast to MAX, ATT can combine different views. In contrast to CON and AVG, ATT can ignore some of them based on low attention weights.

3.2 Analysis: Crossview Learning

To analyze whether sharing parameters across views is important, Table 2 compares (i) SINGLE: twelve different singleview models with (ii) CROSS: a single crossview model that is trained on a training set that combines the twelve individual singleview training sets. For CROSS, we use Eq. 3 with view specific transformation matrices, mapping views in different spaces into a common space, and then Eq. 1 with shared parameters across views. The number of parameters in application is the same for SINGLE and CROSS.

Table 2 shows consistent and clear improvements of CROSS compared to SINGLE, except for English CTXT and NAME. The English Wikipedia is much larger than the others, so that embeddings based on it have high quality. But our results demonstrate that training one model with common parameters over all inputs is helping the classification for non high-resource views.

			all	tail	head
	0	FIGMENT	88.1	87.4	89.1
CTXT	1	EN	71.7	62.0	88.5
	2	DE	31.7	14.6	76.6
	3	ES	20.3	6.1	67.5
	4	FA	09.3	04.5	42.3
	5	MULTIVIEW-CON	**73.7**	**64.6**	89.6
	6	MULTIVIEW-ATT	73.6	64.5	89.2
	7	MULTIVIEW-MAX	**73.7**	64.2	**89.8**
	8	MULTIVIEW-AVG	73.0	63.5	89.3
NAME4	9	EN	73.4	73.1	76.1
	10	DE	34.1	23.8	68.5
	11	ES	29.8	20.4	65.3
	12	FA	17.2	13.0	47.3
	13	MULTIVIEW-CON	75.8	75.0	81.0
	14	MULTIVIEW-ATT	**76.1**	**75.3**	**81.2**
	15	MULTIVIEW-MAX	75.9	75.2	81.0
	16	MULTIVIEW-AVG	75.8	75.1	80.9
DESC	17	EN	77.6	79.5	67.1
	18	DE	28.9	20.6	51.4
	19	ES	23.1	16.2	47.0
	20	FA	12.2	10.7	29.3
	21	MULTIVIEW-CON	**81.6**	**82.3**	**78.2**
	22	MULTIVIEW-ATT	79.6	80.6	73.8
	23	MULTIVIEW-MAX	79.5	80.5	74.4
	24	MULTIVIEW-AVG	78.9	79.7	75.2
CTXT +NAME4 +DESC	25	MULTIVIEW-CON	88.6	87.8	**90.0**
	26	MULTIVIEW-ATT	**89.0**	**88.3**	89.8
	27	MULTIVIEW-MAX	87.9	86.9	89.6
	28	MULTIVIEW-AVG	87.1	86.2	88.4
NAME10	29	MULTIVIEW-CON	76.7	75.8	82.4
	30	MULTIVIEW-ATT	**77.3**	**76.4**	**82.5**
	31	MULTIVIEW-MAX	77.0	76.1	81.9
	32	MULTIVIEW-AVG	76.3	75.4	82.0
CTXT +NAME10 +DESC	33	MULTIVIEW-CON	88.5	87.7	**89.9**
	34	MULTIVIEW-ATT	**89.2**	**88.6**	89.8
	35	MULTIVIEW-MAX	87.6	86.5	89.4
	36	MULTIVIEW-AVG	86.7	85.7	88.1

Table 1: Micro F_1 for entity typing. NAME4/NAME10 = name embeddings from 4 or 10 languages.

Multiview learning exploits the complementarity of views: if an entity's type cannot be inferred from one view, then other views may have the required information. Table 2 shows that using multiple views has a second beneficial effect: even if applied to a single view, a model trained on multiple views performs better. Kan et al. (2016)'s image recognition and Pappas and Popescu-Belis (2017)'s document classification findings are similar. Thus, not only does the increased amount of available information boost performance in the multiview setup, but also we can enable crossview transfer and learn a model that makes better predictions even if information is only available from a single view.

	CTXT		NAME		DESC	
	SINGLE	CROSS	SINGLE	CROSS	SINGLE	CROSS
EN	**71.7**	71.6	**73.4**	72.8	77.6	**82.0**
DE	31.7	**32.3**	34.1	**34.7**	28.9	**35.7**
ES	20.3	**21.2**	29.8	**30.7**	23.1	**28.6**
FA	9.3	**9.5**	17.2	**18.3**	12.2	**15.3**

Table 2: Micro F_1 (all entities) for twelve singleview models (SINGLE) and one crossview model (CROSS)

4 Related Work

Entity and mention typing. In this work, we assume that a predefined set of fine-grained types is given. Entity typing, i.e., predicting types of a knowledge base entity (Neelakantan and Chang, 2015; Yaghoobzadeh and Schütze, 2015), is the focus of this paper. Mention typing, i.e., predicting types of a mention in a particular context (Ling and Weld, 2012; Rabinovich and Klein, 2017; Shimaoka et al., 2017; Murty et al., 2018), is a related task. Mention typing models can be evaluated for entity typing when aggregating their predictions (Yaghoobzadeh and Schütze, 2015; Yaghoobzadeh et al., 2017, 2018). Therefore, our public and large entity typing dataset, MVET, can be used as an alternative to the small manually annotated mention typing datasets like the commonly used FIGER (Ling and Weld, 2012). We leave this to the future work.

Multilingual entity typing. We build multilingual dataset and models for entity typing. Most work on entity typing has been monolingual; e.g., Yaghoobzadeh and Schütze (2015) (English); and Suzuki et al. (2016) (Japanese). There is work on mention typing (van Erp and Vossen, 2017). Lin et al. (2017) that uses mono- and crosslingual attention for relation extraction. Crosslingual entity linking is an important related task, where the task is to link mentions of entities in multilingual text to a knowledge base (Tsai and Roth, 2016). Many entities are not sufficiently annotated in Wikipedia, and therefore crosslingual entity linking is necessary to learn informative context representations from multiple languages.

Multi-representation of entities. Aggregating information from multiple sources to learn entity representations has been explored for entity typing (Yaghoobzadeh and Schütze, 2017; Yaghoobzadeh et al., 2018), entity linking (Gupta et al., 2017) and relation extraction (Wang and Li, 2016). Here, we add language as a new "dimension" to multi-representations: each language contributes a different CTXT, NAME and DESC representation.

Our multilingual and multi-representation models are examples of **multiview learning**. Xu et al. (2013) and Zhao et al. (2017) review the literature on multiview learning. Amini et al. (2009) cast multilingual text classification, a task related to entity typing, as multiview learning. Qu et al. (2017) address node classification and link prediction by attention-based multiview representations of graph nodes. We also adopt a similar approach in our multiview representations for entity typing.

5 Conclusion

We formalized entity typing as a multiview problem by introducing two metaviews, language and representation; each combination of their instances defines a distinct view. Our experiments showed the effectivess of this formalization by outperforming the state-of-the-art model. Our basic idea of metaview learning is general and is applicable to related tasks, e.g., to relation extraction. We release a large and public multiview and, in particular, multilingual, entity typing dataset.

A MVET Dataset Statistics

	NAME	CTXT	DESC
EN	160,083	121,004	159,458
DE	50,853	37,054	45,516
ES	42,279	24,528	37,420
FA	23,389	11,725	18,735
RU	37,233	0	0
FR	54,434	0	0
AR	18,379	0	0
PT	31,879	0	0
PL	35,675	0	0
IT	41,686	0	0

Table 3: The statistics of our MVET dataset for each language and representation view. The total number of entities is 160,083 with 102 types.

Acknowledgments

We thank Ehsaneddin Asgari, Katharina Kann, T.J. Hazen and the anonymous reviewers for their helpful feedback on earlier drafts. This work was partially supported by the European Research Council, Advanced Grant NonSequeToR # 740516.

References

Massih-Reza Amini, Nicolas Usunier, and Cyril Goutte. 2009. Learning from multiple partially observed views - an application to multilingual text categorization. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 28–36.

Avrim Blum and Tom M. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Marieke van Erp and Piek Vossen. 2017. Multilingual fine-grained entity typing. In *Language, Data, and Knowledge - First International Conference*, pages 262–275.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690.

Meina Kan, Shiguang Shan, and Xilin Chen. 2016. Multi-view deep network for cross-view classification. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4847–4855.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 34–43.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings fo the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1299–1304.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, Toronto, Ontario, Canada.

Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. 2018. Hierarchical losses and new resources for fine-grained entity typing and linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 97–109.

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–525.

Nikolaos Pappas and Andrei Popescu-Belis. 2017. Multilingual hierarchical attention networks for document classification. In *Proceedings of 8th International Joint Conference on Natural Language Processing*.

Meng Qu, Jian Tang, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. 2017. An attention-based collaboration framework for multi-view network representation learning. In *Proceedings of the 2017 ACM Int. Conf. on Information and Knowledge Management*.

Maxim Rabinovich and Dan Klein. 2017. Fine-grained entity typing with high-multiplicity assignments. *CoRR*, abs/1704.07751.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. pages 1271–1280.

Masatoshi Suzuki, Koji Matsuda, Satoshi Sekine, Naoaki Okazaki, and Kentaro Inui. 2016. Fine-grained named entity classification with wikipedia article vectors. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2016, Omaha, NE, USA, October 13-16, 2016*, pages 483–486.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual wikification using multilingual embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 589–598.

Zhigang Wang and Juan-Zi Li. 2016. Text-enhanced representation learning for knowledge graph. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1293–1299.

Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *CoRR*, abs/1304.5634.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schuetze. 2018. Corpus-level fine-grained entity typing. *Journal of Artificial Intelligence Research*, 61:835–862.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1183–1194.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2017. Multi-level representations for fine-grained typing of knowledge base entities. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 578–589.

Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. 2017. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54.

Word Embeddings for Code-Mixed Language Processing

Adithya Pratapa, Monojit Choudhury, Sunayana Sitaram
Microsoft Research, India
{t-pradi,monojitc,sunayana.sitaram}@microsoft.com

Abstract

We compare three existing bilingual word embedding approaches, and a novel approach of training skip-grams on synthetic code-mixed text generated through linguistic models of code-mixing, on two tasks - sentiment analysis and POS tagging for code-mixed text. Our results show that while CVM and CCA based embeddings perform as well as the proposed embedding technique on semantic and syntactic tasks respectively, the proposed approach provides the best performance for both tasks overall. Thus, this study demonstrates that existing bilingual embedding techniques are not ideal for code-mixed text processing and there is a need for learning multilingual word embedding from the code-mixed text.

1 Introduction

Word embeddings are useful for a variety of NLP tasks, as they allow to generalize the system on much larger corpora than the annotated dataset for the task. In recent times, there has been some interest in bilingual word embeddings, where words from two languages are embedded into the same space. The primary advantage of bilingual embeddings is in solving tasks involving reasoning across two languages, such as Machine Translation (Zou et al., 2013; Gouws et al., 2015; Vulić and Moens, 2016) and cross-lingual IR (Vulić and Moens, 2015), as well as allowing transfer of models learnt on a resource-rich language on to a resource poor language (Adams et al., 2017; Fang and Cohn, 2017). One of the potential, yet unexplored, applications of bilingual word embeddings is in the processing of code-mixed language.

Code-mixing (CM) refers to fluid alternation between two or more languages in a single conversation/sentence (Myers-Scotton, 1993). CM is a common phenomenon observed in almost all multilingual societies (Parshad et al., 2016; Rijhwani et al., 2017). Consequently, in recent times, processing of CM text and speech has been receiving a growing amount of interest and attention from the NLP community (Solorio and Liu, 2008; Li and Fung, 2014; Solorio et al., 2014; Sharma et al., 2016; Rudra et al., 2016). Since CM text draws words and linguistic structures from multiple languages, use of bilingual word embeddings for processing of such text could not only be useful, but also necessary. On the other hand, while there is some work that uses embeddings for CM text (Prabhu et al., 2016) (at sub-word level), we do not know of any study that systematically explores the usefulness of bilingual word embedding techniques in CM text processing.

Further, we argue that since all the standard bilingual word embedding techniques are designed to work on or across monolingual texts rather than on a mixture of the two languages, these techniques may not be ideal for learning embeddings for CM tasks. There are emergent syntactic structures and cross-lingual semantic associations in CM text, that do not exist in the individual monolingual corpora (Sec 3). Hence, ideally, word embeddings for CM tasks should be trained on real CM data.

In this paper, we compare three popular bilingual word embedding techniques (Sec 2): Bilingual correlation based embeddings (BiCCA) (Faruqui and Dyer, 2014), Bilingual compositional model (BiCVM) (Hermann and Blunsom, 2014) and Bilingual Skip-gram (BiSkip) (Luong et al., 2015) on two tasks for CM text - sentiment analysis, a semantic task, and POS tagging, a syntactic task. On the same tasks, we also compare word embeddings learnt from synthetic CM data (generated using linguistic models as proposed in a recent work (Pratapa et al., 2018)) (Sec 3). Note that Wick et al. (2016) use artificial code mixed data to learn

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3067–3072
Brussels, Belgium, October 31 - November 4, 2018.

multilingual embeddings for cross-lingual tasks, but their aim is to generate bilingual embeddings for monolingual or cross-lingual tasks.

Our study shows that even though in certain NLP tasks specific embeddings might perform well, in general bilingual embedding techniques like BiCCA, BiCVM and BiSkip are not ideal for processing CM language. Embeddings learnt from CM data, even if artificially generated, performs consistently better across tasks. Our initial results are promising and provide several interesting directions for further exploration.

2 Bilingual Embeddings

In the past few years, there has been a growing interest in learning bilingual embeddings (Upadhyay et al., 2016; Ruder et al., 2017) with a focus on cross-lingual transfer, which helps in building NLP models for low-resource languages. Upadhyay et al. (2016) provide an empirical comparison of four cross-lingual word embedding models varying in terms of the amount of supervision. Ruder et al. (2017) establishes the similarities among numerous cross-lingual word embedding models and shows that many models optimize for similar objectives. Along similar lines as Upadhyay et al. (2016), in this work, we chose the following three representative bilingual word embedding models for CM tasks. Training data is described later in Section 4.

2.1 Bilingual Correlation Based Embeddings (BiCCA)

Faruqui and Dyer (2014) proposed CCA based bilingual embeddings, where bilingual evidence is incorporated into word representations by performing canonical correlation analysis (CCA) on monolingual embeddings using a bilingual dictionary. The monolingual embedding matrices W_{L_1} and W_{L_2} can be of different dimensions and words with their translations from the dictionary are used to obtain matrices W'_{L_1} and W'_{L_2}. Using CCA, the individual projection matrices are computed, which are then utilized to project the embeddings into the same embedding space.

BiCCA embeddings were shown to perform well on syntactic tasks like cross lingual dependency parsing, but performed relatively poorly on cross lingual semantic tasks (Upadhyay et al., 2016). We learn the monolingual embeddings by training a skip-gram model (Mikolov et al., 2013) for 5 iterations with a window size of 5 and 10 negative samples. We built a bilingual dictionary of approximately 38k pairs by using word alignments. The dictionary contains word pairs (w_1, w_2), such that w_2 is aligned to w_1 the highest number of times and vice-versa. We use the `crosslingual-cca`[1] toolkit. To remain consistent with other embedding models, we choose the top (k=) 200 correlated dimensions.

2.2 Bilingual Compositional Model (BiCVM)

This approach, proposed by Hermann and Blunsom (2014), is based on the assumption that parallel sentences from different languages have equivalent meanings and thus should have similar sentence representations. Along with monolingual regularizers, the model optimizes for the aligned sentences to be closer to each other than randomly chosen negative samples, using a noise-contrastive update.

BiCVM is found to perform well on monolingual word similarity (SimLex-999, Upadhyay et al. (2016)) and is comparable to BiSkip on semantic tasks like cross-lingual document classification. We use the `bicvm`[2] toolkit to generate embeddings using the parallel English-Spanish data. We train an additive model for 100 iterations, with a hinge loss margin of 200, noise parameter of 10 and batch size of 50.

2.3 Bilingual Skip-gram Model (BiSkip)

The Skip-gram model proposed by Mikolov et al. (2013) has been adapted to the bilingual setting in Luong et al. (2015), where the model learns to predict word contexts cross-lingually. Along with the monolingual skip-gram with negative sampling objectives, BiSkip includes two more objectives L_{12} and L_{21} when predicting cross-lingually from L_1 to L_2 and vice-versa.

It has the best performance compared to the other embedding techniques on semantic tasks like cross-lingual dictionary induction and cross-lingual document classification. We use parallel sentences and word level alignments to train the `biskip`[3] model. We train the model using the toolkit for 5 iterations with 10 negative samples, high frequency threshold of 0.001, window size of 5 and cross-lingual weight as 1.

---

[1] `https://github.com/mfaruqui/crosslingual-cca`

[2] `https://github.com/karlmoritz/bicvm/`

[3] `https://github.com/lmthang/bivec`

3 Synthetic CM data (gCM) based Embeddings

The aforementioned embedding techniques, owing to their ability to project the words of the two languages into a single space, are expected to be helpful in processing of CM text. However, we believe that CM text is distinct in its syntactic, semantic and statistical properties from the corresponding monolingual texts. For example, there has been a lot of work on understanding and establishing the grammatical constraints of CM (Joshi, 1985; Poplack, 1980; DiSciullo et al., 1986) and these syntactic constraints might introduce interesting collocations in the word space, such as that between English verbs and the Hindi verb "kar" (to do) in English-Hindi CM (Kachru, 1978).

In a recent work (Pratapa et al., 2018), we presented a methodology to generate linguistic theory based synthetic CM data (gCM) and showed its effectiveness in CM language modeling. Synthetic CM data was generated by employing Equivalence Constraint (EC) theory (Poplack, 1980; Sankoff, 1998). The generation model builds a lattice of grammatically valid CM sentences by imposing the EC theory constraints while utilizing monolingual parallel data, word-level alignments and the parse of English sentences. The Spanish parse is formed by projecting the English parse onto the Spanish sentence. For each monolingual pair of input sentences, a large number (generally exponential in terms of average input sentence length) of CM sentences were generated. We observed a non-uniform scaling of low and high frequency words and proposed two sampling techniques to overcome this frequency bias.

In the current work, we use this generation model (as proposed in (Pratapa et al., 2018)) to create training data for learning CM word embeddings. In fact, we also noticed the frequency bias in the word embedding space (Figure 1a). We adapt the two sampling strategies from the original paper, 1. Random sampling (χ-***gCM***), for every monolingual pair, sample random k CM sentences and, 2. SPF-based sampling (ρ-***gCM***), sample k CM sentences for each monolingual pair such that they have SPF (switch point fraction, i.e fraction of word boundaries where the language changes) distribution similar to real CM data (Pratapa et al., 2018). We were able to alleviate the frequency bias using the SPF-based sampling (Figure 1b).

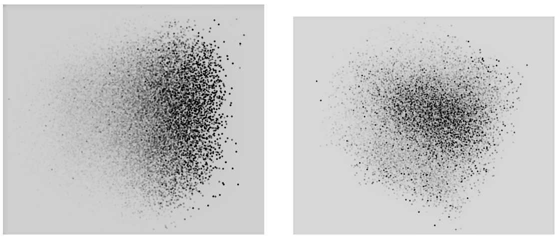
(a) (b)

Figure 1: (a) PCA projections of skip-gram (with negative sampling) embeddings trained on entire gCM corpus, (b) ρ-sampled gCM. The color gradient (light → dark) is based on the frequency of word (low → high) in gCM corpus.

4 Evaluation

Data: Though CM is a common phenomenon, there is a scarcity of real code-mixed data in textual form. Hence, we learn CM embeddings from the English-Spanish parallel corpora (for BiCCA, BiCVM and BiSkip models) and the synthetic CM data obtained from Pratapa et al. (2018). This constitutes to approximately 4.5M parallel sentences and also bilingual supervision in the form of word and sentence alignments. The synthetic data creation procedure is described in the original paper. Both the sampling techniques, with k=2 result in a gCM corpus of approx. 8M sentences. We also combine monolingual data with these gCM corpus resulting in approx. 17M sentences. We train a skip-gram model for 10 iterations, with a window size of 5 and 5 negative samples, resulting in χ-***gCM-Skip*** and ρ-***gCM-Skip*** embeddings.

To quantitatively compare the embedding models, we chose two CM tasks, one semantic (Sentiment Analysis) and one syntactic task (POS tagging). Our choice of tasks is primarily motivated by the availability of annotated CM data. There has been prior work on CM sentiment identification (Vilares and Alonso, 2016; Joshi et al., 2016; Rudra et al., 2016; Prabhu et al., 2016) and POS tagging (Solorio and Liu, 2008; Vyas et al., 2014; AlGhamdi et al., 2016; Ghosh et al., 2016). But we are not aware of any work that utilizes pre-trained bilingual embeddings for these tasks.

4.1 Sentiment Analysis

Vilares and Alonso (2016) provide 2103 sentiment annotated CM tweets. The data contains 650 positive, 529 negative and 924 neutral tweets and we split the data in 8:1:1 ratio (train:validation:test)

Embedding	**Sentiment**			**POS**	
	CM Overall	SemEval 2014	TASS 2016	CM Overall	at SP
None	54.4 (1.3)	64.5 (0.6)	61.4 (1.0)	84.5 (0.3)	74.0 (0.7)
BiCCA	57.6 (3.0)	64.6 (1.0)	59.5 (1.8)	**84.7** (0.8)	**75.0** (1.8)
BiCVM	**64.3** (1.3)	66.8 (1.0))	61.9 (1.0)	82.0 (0.5)	70.6 (1.7)
BiSkip	61.5 (1.7)	66.6 (0.9)	**63.9** (1.2)	84.4 (0.7)	73.8 (0.9)
χ-gCM-Skip	62.0 (1.9)	**67.4** (1.3)	63.2 (1.5)	**84.8** (0.6)	74.0 (0.6)
ρ-gCM-Skip	**64.6** (2.0)	**67.7** (1.4)	**63.8** (2.2)	**84.9** (0.7)	**75.3** (1.7)

Table 1: The performance of different pre-trained embeddings on Sentiment (F1 score) and POS tasks (Accuracy). The reported values are mean and deviation (in parentheses) values computed over multiple runs.

while ensuring the sentiment distribution remains the same[4].

Model: We train a LSTM based sequence classifier with single hidden layer (dim=50) and dropout of 0.5. We use ADAM optimizer with learning rate of 0.001 and momentum parameter of 0.9. We train the model for a maximum of 10 epochs with a mini-batch size of 100. Our model is built using the Microsoft CNTK framework. We varied the CM sample size (k) (described in 3) over $\{1,2,5,10,20\}$ and found the best performance with k=2. To check the robustness of CM embeddings, we also evaluate them on monolingual English[5] (SemEval 2014 Subtask B, (7177, 1199, 2865)) and Spanish[6] (TASS 2016, (6000, 1220, 1000)) tasks. The numbers in the parenthesis indicate the number of train, validation and test instances.

Results: Table 1 shows the results on Sentiment task. While all embedding models significantly improve over the baseline ('None'), ρ-gCM-Skip and BiCVM perform the best. Instead of linguistically motivated data, we tried with CM data created by random juxtaposition of monolingual fragments and it gave a F1 score of 56.0% and the minimal gain is possibly only because of the monolingual fragments in the embedding training data.

4.2 Part of Speech (POS) Tagging

Of the two corpora utilized in AlGhamdi et al. (2016), we chose Bangor Miami Corpus[7] over Spanglish Corpus (Solorio and Liu, 2008) owing to the larger size of the former corpus. Bangor Miami corpus consists of conversations of Spanish speakers in Florida but also fluent in English. In contrast to AlGhamdi et al. (2016), we only consider the code mixed utterances with significant ($\geq$30%) fraction of English and Spanish. This accounts to 982 sentences and 7705 tokens, split in 8:1:1 ratio.

Model: We use a bidirectional LSTM model with CRF as the output layer (Rei and Yannakoudakis, 2016)[8], with hidden layer dimension of 50 and dropout of 0.5. The model trains for a maximum of 20 epochs and terminates if there is no improvement in validation accuracy for 5 consecutive epochs. We use ADADELTA optimizer with a learning rate of 1.0.

Results: ρ-gCM-Skip, χ-gCM-Skip perform the best on the POS task (see Table. 1). Unlike the sentiment task, BiCCA performs close to the best while BiCVM does the worst. We also report accuracies at switch points (SP), which we believe is challenging for a CM POS tagger. As expected these accuracies are lower, but ρ-gCM-Skip and BiCCA still perform the best.

5 Conclusion

As we expected, pretrained bilingual word embeddings help improve syntactic and semantic CM processing tasks. Similar to (Upadhyay et al., 2016), we also note that BiCVM operating at sentence level performs better only on semantic tasks, while BiCCA does well only on syntactic tasks due to their usage of word alignments. Though ρ-gCM-Skip embeddings learnt from synthetic data

---

[4] These statistics are for the re-crawled tweets; some of the original tweets are no longer available.

[5] `alt.qcri.org/semeval2014/task9/`

[6] `www.sepln.org/workshops/tass/2016/`

[7] `bangortalk.org.uk/speakers.php?c=miami`

[8] `https://github.com/marekrei/sequence-labeler`

performed only marginally better than BiCVM and BiCCA on semantic and syntactic tasks respectively, it is the only embedding model that did consistently well across tasks. Thus, our study shows that standard bilingual embeddings are not well suited, in general, for CM tasks; embeddings learnt from CM data, either real or synthetic, seems much more useful. Further, along the lines of (Pratapa et al., 2018), our study also shows that synthetic CM data is a reasonably good proxy for real data.

While our experiments show a promising direction towards obtaining bilingual embeddings for CM tasks, there are several interesting ideas that are worth exploring. In particular, the linguistic model used for generating artificial CM data only addresses the syntactic constraints of CM, but not other kinds of constraints such as *lexical choice* which in a particular CM context might be overly skewed towards one language (like the English words 'school' and 'vote' are more common than their Hindi translations in English-Hindi CM (Bali et al., 2014)), and semantic/pragmatic constraints that make the choice of a particular language more common in some contexts (e.g., Hindi used more commonly for negative sentiment during English-Hindi CM (Rudra et al., 2016)). Similarly, the sense distribution of polysemous words can vary widely between a monolingual and CM corpus. For instance, the word 'school' in English has several meanings such as (a) an institute of education, (b) group of artists, writers etc., and (c) a large group of fish. However, in a Spanish dominant sentence or corpus, school is primarily, if not only, used in sense (a).

As future work, it will be interesting to explore techniques that can generate artificial CM data following the lexical, semantic and pragmatic constraints, or develop novel embedding techniques that can appropriately interpolate between real and artificial CM data to learn collocations that arise due to not only syntactic but also lexical, semantic and pragmatic aspects of code-mixing.

Acknowledgements

We would like to thank Kalika Bali, Gayatri Bhat and Sandipan Dandapat for their valuable suggestions and help in the creation of the synthetic CM corpus.

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 937–947.

Fahad AlGhamdi, Giovanni Molina, Mona Diab, Thamar Solorio, Abdelati Hawwari, Victor Soto, and Julia Hirschberg. 2016. Part of speech tagging for code switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 98–107.

Kalika Bali, Y. Vyas, J. Sharma, and M. Choudhury. 2014. "I am borrowing ya mixing?" An analysis of English-Hindi code mixing in Facebook. In *Proc. First Workshop on Computational Approaches to Code Switching, EMNLP*.

A.-M. DiSciullo, Pieter Muysken, and R. Singh. 1986. Government and code-mixing. *Journal of Linguistics*, 22:1–24.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 587–593.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*.

Souvick Ghosh, Satanu Ghosh, and Dipankar Das. 2016. Part-of-speech tagging of code-mixed social media text. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 90–97.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *International Conference on Machine Learning*, pages 748–756.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Models for Compositional Distributional Semantics. In *Proceedings of ACL*.

A. K. Joshi. 1985. Processing of Sentences with Intrasentential Code Switching. In D. R. Dowty, L. Karttunen, and A. M. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 190–205. Cambridge University Press, Cambridge.

Aditya Joshi, Ameya Prabhu, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of Hindi-English code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491.

Braj B Kachru. 1978. Toward structuring code-mixing: An Indian perspective. *International Journal of the Sociology of Language*, 1978(16):27–46.

Ying Li and P Fung. 2014. Language modeling with functional head constraint for code switching speech recognition. In *EMNLP*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *NAACL Workshop on Vector Space Modeling for NLP*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Carol Myers-Scotton. 1993. *Duelling Languages:Grammatical structure in Code-switching*. Clarendon Press, Oxford.

Rana D. Parshad, Suman Bhowmick, Vineeta Chand, Nitu Kumari, and Neha Sinha. 2016. What is India speaking? Exploring the “Hinglish” invasion. *Physica A*, 449:375–389.

Shana Poplack. 1980. Sometimes Ill start a sentence in Spanish y termino en espaol. *Linguistics*, 18:581–618.

Ameya Prabhu, Aditya Joshi, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of Hindi-English code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491.

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1543–1553.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1181–1191.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. 2017. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1971–1982.

Sebastian Ruder, Ivan Vuli, and Anders Sgaard. 2017. A survey of cross-lingual embedding models. *arXiv preprint arXiv:1706.04902*.

Koustav Rudra, S Rijhwani, R Begum, K Bali, M Choudhury, and N Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do Hindi-English speakers do on Twitter? In *EMNLP*, pages 1131–1141.

David Sankoff. 1998. A formal production-based explanation of the facts of code-switching. *Bilingualism: language and cognition*, 1(01):39–50.

A. Sharma, S. Gupta, R. Motlani, P. Bansal, M. Srivastava, R. Mamidi, and D.M Sharma. 2016. Shallow parsing pipeline for Hindi-English code-mixed social media text. In *Proceedings of NAACL-HLT*.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for English-Spanish code-switched text. In *Proc. of EMNLP*.

Thamar Solorio et al. 2014. Overview for the first shared task on language identification in code-switched data. *In 1st Workshop on Computational Approaches to Code Switching, EMNLP*, pages 62–72.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. *arXiv preprint arXiv:1604.00425*.

David Vilares and Miguel A Alonso. 2016. En-es-cs: An English-Spanish code-switching twitter corpus for multilingual sentiment analysis. In *LREC*.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 363–372. ACM.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. POS tagging of English-Hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979.

Michael Wick, Pallika Kanani, and Adam Craig Pocock. 2016. Minimally-constrained multilingual embeddings via artificial code-switching. In *AAAI*, pages 2849–2855.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.

On the Strength of Character Language Models for Multilingual Named Entity Recognition

Xiaodong Yu[†], Stephen Mayhew[*], Mark Sammons[†], Dan Roth[*]
[†]University of Illinois, Urbana-Champaign, [*]University of Pennsylvania
`{xyu71,mssammon}@illinois.edu, {mayhew,danroth}@seas.upenn.edu`

Abstract

Character-level patterns have been widely used as features in English Named Entity Recognition (NER) systems. However, to date there has been no direct investigation of the inherent differences between name and non-name tokens in text, nor whether this property holds across multiple languages. This paper analyzes the capabilities of corpus-agnostic Character-level Language Models (CLMs) in the binary task of distinguishing name tokens from non-name tokens. We demonstrate that CLMs provide a simple and powerful model for capturing these differences, identifying named entity tokens in a diverse set of languages at close to the performance of full NER systems. Moreover, by adding very simple CLM-based features we can significantly improve the performance of an off-the-shelf NER system for multiple languages.[1]

1 Introduction

In English, there is strong empirical evidence that the character sequences that make up proper nouns tend to be distinctive. Even divorced of context, a human reader can predict that "hoekstenberger" is an entity, but "abstractually"[2] is not. Some NER research explores the use of character-level features including capitalization, prefixes and suffixes (Cucerzan and Yarowsky, 1999; Ratinov and Roth, 2009), and character-level models (CLMs) (Klein et al., 2003) to improve the performance of NER, but to date there has been no systematic study isolating the utility of CLMs in capturing distinctions between name and non-name tokens in English or across other languages.

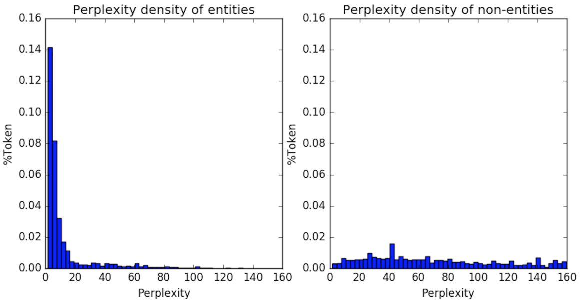

Figure 1: Perplexity histogram of entity (left) and non-entity tokens (right) in CoNLL Train calculated by entity CLM for both sides. The graphs show the percentage of tokens (y axis) with different levels of CLM perplexities (x axis). The entity CLM gives a low average perplexity and small variance to entity tokens (left), while giving non-entity tokens much higher perplexity and higher variance (right).

We conduct an experimental assessment of the discriminative power of CLMs for a range of languages: English, Amharic, Arabic, Bengali, Farsi, Hindi, Somali, and Tagalog. These languages use a variety of scripts and orthographic conventions (for example, only three use capitalization), come from different language families, and vary in their morphological complexity. We demonstrate the effectiveness of CLMs in distinguishing name tokens from non-name tokens, as illustrated by Figure 1, which shows perplexity histograms from a CLM trained on entity tokens. Our models use only individual tokens, but perform extremely well in spite of taking no account of word context.

We then assess the utility of directly adding simple features based on this CLM implementation to an existing NER system, and show that they have a significant positive impact on performance across many of the languages we tried. By adding very simple CLM-based features to the system, our scores approach those of a state-of-the-art NER system (Lample et al., 2016) across multiple languages, demonstrating both the unique importance and the broad utility of this approach.

---

[1] The code and resources for this publication can be found at: `https://cogcomp.org/page/publication_view/846`

[2] Not a real name or a real word.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3073–3077
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

	Train		Test	
Language	Entity	Non-entity	Entity	Non-entity
English	29,450	170,524	7,194	38,554
Amharic	5,886	46,641	2,077	16,235
Arabic	7,640	52,968	1,754	15,073
Bengali	15,288	108,592	4,573	32,929
Farsi	4,547	50,084	1,608	13,968
Hindi	5,565	69,267	1,947	23,853
Somali	6,467	51,034	1,967	14,545
Tagalog	11,525	102,894	3,186	29,228

Table 1: Data statistics for all languages, showing number of entity and non-entity tokens in Train and Test.

2 Methods

2.1 Character Language Models

We propose a very simple model in which we train an entity CLM on a list of entity tokens, and a non-entity CLM on a list of non-entity tokens. Both lists are unordered, with all entries treated independently. Each token is split into characters and treated as a "sentence" where the characters are the "words." For example, "Obama" is an entity token, and is split into "O b a m a". From these examples we learn a score measuring how likely it is that a sequence of characters forms an entity. At test time, we also split each word into characters and determine perplexity using the entity and non-entity CLMs. We assign the label corresponding to the lower perplexity CLM.

We experiment with four different kinds of language model: N-gram model, Skip-gram model, Continuous Bag-of-Words model (CBOW), and Log-Bilinear model (LB). We demonstrate that the N-gram model is best suited for this task.

Following Peng and Roth (2016), we implement N-gram using SRILM (Stolcke, 2002) with order 6 and Witten-Bell discounting.[3] For Skip-Gram and CBOW CLMs, we use the Gensim implementation (Rehurek and Sojka, 2010) for training and inference, and we build the LB CLM using the OxLM toolkit (Baltescu et al., 2014).

2.2 Data

To determine whether name identifiability applies to languages other than English, we conduct experiments on a range of languages for which we had previously gathered resources (such as Brown clusters): English, Amharic, Arabic, Bengali, Farsi, Hindi, Somali, and Tagalog.

For English, we use the original splits from the ubiquitous CoNLL 2003 English dataset (Sang and Meulder, 2003), which is a newswire dataset annotated with Person (PER), Organization (ORG), Location (LOC) and Miscellaneous (MISC). To collect the list of entities and non-entities as the training data for the Entity and Non-Entity CLMs, we sample a large number of PER/ORG/LOC and non-entities from Wikipedia, using types derived from their corresponding FreeBase entities (Ling and Weld, 2012).

For all other languages, we use a subset of the corpora from the LORELEI project annotated for the NER task (Strassel and Tracey, 2016). We build our entity list using the tokens labeled as entities in the training data, and our non-entity list from the remaining tokens. These two lists are then used to train two CLMs, as described above.

Our datasets vary in size of entity and non-entity tokens, as shown in Table 1. The smallest, Farsi, has 4.5K entity and 50K non-entity tokens; the largest, English, has 29K entity and 170K non-entity tokens.

3 CLM for Named Entity Identification

In this section, we first show the power of CLMs for distinguishing between entity and non-entity tokens in English, and then that this power is robust across a variety of languages.

We refer to this task as Named Entity Identification (NEI), because we are concerned only with finding an entity span, not its label. We differentiate it from Named Entity Recognition (NER), in which both span and label are required. To avoid complicating this straightforward approach by requiring a separate mention detection step, we evaluate at the token-level, as opposed to the more common phrase-level evaluation. We also apply one heuristic: if a word has length 1, we automatically predict 'O' (or non-entity). This captures most punctuation and words like 'I' and 'a'.

Figure 1 shows that for the majority of entity tokens, the entity CLM computes a relatively low perplexity compared to non-entity tokens. Though there also exist some non-entities with low entity CLM perplexity, we can still reliably identify a large proportion of non-entity words by setting a threshold value for entity CLM perplexity. If a token perplexity lies above this threshold, we label it as a non-entity token. The threshold is tuned on development data.

---

[3] We experimented with different orders on development data, but found little difference between them.

Model	**eng**	**amh**	**ara**	**ben**	**fas**	**hin**	**som**	**tgl**	avg
Exact Match	43.4	54.4	29.3	47.7	30.5	30.9	46.0	23.7	37.5
Capitalization	79.5	-	-	-	-	-	69.5	77.6	-
SRILM	**92.8**	**69.9**	**54.7**	**79.4**	**60.8**	**63.8**	**84.1**	**80.5**	**70.5**
Skip-gram	76.0	53.0	29.7	41.4	30.8	29.0	51.1	61.5	42.4
CBOW	73.7	50.0	28.1	40.6	32.6	26.5	56.4	62.5	42.4
Log-Bilinear	82.8	64.5	46.1	70.8	50.4	54.8	78.1	74.9	62.8
CogCompNER (ceiling)	96.5	73.8	64.9	80.6	64.1	75.9	89.4	88.6	76.8
Lample et al. (2016) (ceiling)	96.4	84.4	69.8	87.6	76.4	86.3	90.9	91.2	83.8

Table 2: Token level identification F1 scores. Averages are computed over all languages other than English. Two baselines are also compared here: *Capitalization* tags a token in test as entity if it is capitalized; and *Exact Match* keeps track of entities seen in training, tagging tokens in Test that exactly match some entity in Train. The bottom section shows state-of-the-art models which use complex features for names, including contextual information. Languages in order are: English, Amharic, Arabic, Bengali, Farsi, Hindi, Somali, and Tagalog. The rightmost column is the average of all columns excluding English.

Since we also build a CLM for non-entities, we can also compare the entity and non-entity perplexity scores for a token. For those tokens not excluded using the threshold as described above, we compare the perplexity scores of the two models and assign the label corresponding to the model yielding the lower score.

We compare SRILM against Skip-gram and CBOW, as implemented in Gensim, and the Log-Bilinear (LB) model. We trained both CBOW and Skip-gram with window size 3, and size 20. We tuned LB, and report results with embedding size 150, and learning rate 0.1. Despite tuning the neural models, the simple N-gram model outperforms them significantly, perhaps because of the relatively small amount of training data.[4]

We compare the CLM's Entity Identification against two state-of-the-art NER systems: CogCompNER (Khashabi et al., 2018) and LSTM-CRF (Lample et al., 2016). We train the NER systems as usual, but at test time we convert all predictions into binary token-level annotations to get the final score. As Table 2 shows, the result of N-gram CLM, which yields the highest performance, is remarkably close to the result of state-of-the-art NER systems (especially for English) given the simplicity of the model.

4 Improving NER with CLM features

In this section we show that we can augment a standard NER system with simple features based on our entity/non-entity CLMs to improve performance in many languages. Based on their superior performance as reported in Section 3, we use the N-gram CLMs.

4.1 Features

We define three simple features that capture information provided by CLMs and which we expect to be useful for NER.

Entity Feature We define one "isEntity" feature based on the perplexities of the entity and non-entity CLMs. We compare the perplexity calculated by entity CLM and non-entity CLM described in Section 3, and return a boolean value indicating whether the entity CLM score is lower.

Language Features We define two language-related features: "isArabic" and "isRussian". We observe that there are many names in English text that originate from other languages, resulting in very different orthography than native English names. We therefore build two language-based CLMs for Arabic and Russian. We collect a list of Arabic names and a list of Russian names by scraping name-related websites, and train an Arabic CLM and a Russian CLM. For each token, when the perplexity of either the Arabic or the Russian CLM is lower than the perplexity of the Non-Entity CLM, we return True, indicating that this entity is likely to be a name from Arabic/Russian. Otherwise, we return False.

---

[4] We also tried a simple RNN+GRU language model, but found that the results were underwhelming.

Model		eng	amh	ara	ben	fas	hin	som	tgl	avg
Lample et al. (2016)	Full	90.94	**73.2**	57.2	**77.7**	**61.2**	**77.7**	81.3	**83.2**	**73.1**
	Unseen	86.11	51.9	30.2	57.9	41.4	62.2	66.5	72.8	54.7
CogCompNER	Full	90.88	67.5	54.8	74.5	57.8	73.5	82.0	80.9	70.1
	Unseen	84.40	42.7	25.0	51.9	31.5	53.9	67.2	68.3	48.6
CogCompNER+LM	Full	**91.21**	71.3	**59.1**	75.5	59.0	74.2	**82.1**	78.5	71.4
	Unseen	85.20	48.4	32.0	54.0	31.2	55.4	68.0	65.2	50.6

Table 3: NER results on 8 languages show that even a simplistic addition of CLM features to a standard NER model boosts performance. CogCompNER is run with standard features, including Brown clusters; (Lample et al., 2016) is run with default parameters and pre-trained embeddings. *Unseen* refers to performance on named entities in Test that were not seen in the training data. *Full* is performance on all entities in Test. Averages are computed over all languages other than English.

4.2 Experiments

We use CogCompNER (Khashabi et al., 2018) as our baseline NER system because it allows easy integration of new features, and evaluate on the same datasets as before. For English, we add all features described above. For other languages, due to the limited training data, we only use the "isEntity" feature. We compare with the state-of-the-art character-level neural NER system of (Lample et al., 2016), which inherently encodes comparable information to CLMs, as a way to investigate how much of that system's performance can be attributed directly to name-internal structure.

The results in Table 3 show that for six of the eight languages we studied, the baseline NER can be significantly improved by adding simple CLM features; for English and Arabic, it performs better even than the neural NER model of (Lample et al., 2016). For Tagalog, however, adding CLM features actually impairs system performance.

In the same table, the rows marked "unseen" report systems' performance on named entities in Test that were not seen in the training data. This setting more directly assesses the robustness of a system to identify named entities in new data. By this measure, Farsi NER is not improved by name-only CLM features and Tagalog is impaired. Benefits for English, Hindi, and Somali are limited, but are quite significant for Amharic, Arabic, and Bengali.

5 Discussion

Our results demonstrate the power of CLMs for recognizing named entity tokens in a diverse range of languages, and that in many cases they can improve off-the-shelf NER system performance even when integrated in a simplistic way.

However, the results from Section 4.2 show that this is not true for all languages, especially when only considering unseen entities in Test: Tagalog and Farsi do not follow the trend for the other languages we assessed even though CLM performs well for Named Entity Identification.

While the end-to-end model developed by (Lample et al., 2016) clearly includes information comparable to that in the CLM, it requires a fully annotated NER corpus, takes significant time and computational resources to train, and is non-trivial to integrate into a new NER system. The CLM approach captures a very large fraction of the entity/non-entity distinction capacity of full NER systems, and can be rapidly trained using only entity and non-entity token lists – i.e., it is corpus-agnostic. For some languages it can be used directly to improve NER performance; for others (such as Tagalog), the strong NEI performance indicates that while it does not immediately boost performance, it can ultimately be used to improve NER there too.

6 Related Work

Cucerzan and Yarowsky (1999) is one of the earliest works to use character-based features (character tries) for NER. The approach of Klein et al. (2003) was one of the original papers in the CoNLL 2003 NER shared task. Their approach, which ranked in the top 3 for both English and German shared tasks, used character-based features for NER. They do two experiments: one with a character-based HMM, another with using character n-grams as features to a maximum entropy model. The focus on character-level patterns is similar to our work, but without the specific exploration of language models alone.

Using character-based models similar to ours, Smarr and Manning (2002) show that unseen noun phrases can be accurately classified into a small number of categories using only a character-based model independent of context. We tackle a somewhat more challenging task of distinguishing entities from non-entities. Lample et al. (2016) use character embeddings in an LSTM-CRF model. Their ablation studies show that character-level features improve performance significantly.

We are not aware of any work that directly evaluates CLMs for identifying name tokens, nor of work that demonstrates the utility of character-level information for identifying names in multiple languages.

7 Conclusions and Future Work

We have shown, in a series of simple experiments, that in many languages names are identifiable by character patterns alone, and that character level patterns have strong potential for building better NER systems.

In the future, we plan to make a more thorough analysis of reasons for the high variance in NER performance. In particular, we will study why it is possible, as with Tagalog, to have high Named Entity Identification results but lose points in NER.

Acknowledgements

This work was supported by a grant from Google, and by Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We appreciate the helpful discussions and suggestions from Haoruo Peng and Qiang Ning, and from the anonymous EMNLP reviewers.

References

Paul Baltescu, Phil Blunsom, and Hieu Hoang. 2014. Oxlm: A neural language modelling framework for machine translation. *The Prague Bulletin of Mathematical Linguistics* 102(1):81–92. https://ufal.mff.cuni.cz/pbml/102/art-baltescu-blunsom-hoang.pdf.

Silviu Cucerzan and David Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. In *EMNLP*.

Daniel Khashabi, Mark Sammons, Ben Zhou, Tom Redman, Christos Christodoulopoulos, Vivek Srikumar, Nicholas Rizzolo, Lev Ratinov, Guanheng Luo, Quang Do, Chen-Tse Tsai, Subhro Roy, Stephen Mayhew, Zhili Feng, John Wieting, Xiaodong Yu, Yangqiu Song, Shashank Gupta, Shyam Upadhyay, Naveen Arivazhagan, Qiang Ning, Shaoshi Ling, and Dan Roth. 2018. CogCompNLP: Your swiss army knife for nlp. In *11th Language Resources and Evaluation Conference*.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. 2003. Named entity recognition with character-level models. In *CoNLL*.

Guillaume Lample, Miguel Ballesteros, Sandeep K Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *HLT-NAACL*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. http://aiweb.cs.washington.edu/ai/pubs/ling-aaai12.pdf.

Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*. http://cogcomp.org/papers/PengRo16.pdf.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*. http://cogcomp.org/papers/RatinovRo09.pdf.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Joseph Smarr and Christopher D. Manning. 2002. Classifying unknown proper noun phrases without context.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*.

Stephanie Strassel and Jennifer Tracey. 2016. Lorelei language packs: Data, tools, and resources for technology development in low resource languages.

Code-switched Language Models
Using Dual RNNs and Same-Source Pretraining

Saurabh Garg* **Tanmay Parekh*** **Preethi Jyothi**

Indian Institute of Technology, Bombay
`{saurabhgarg,tanmayb,pjyothi}@cse.iitb.ac.in`

Abstract

This work focuses on building language models (LMs) for code-switched text. We propose two techniques that significantly improve these LMs: 1) A novel recurrent neural network unit with dual components that focus on each language in the code-switched text separately 2) Pretraining the LM using synthetic text from a generative model estimated using the training data. We demonstrate the effectiveness of our proposed techniques by reporting perplexities on a Mandarin-English task and derive significant reductions in perplexity.

1 Introduction

Code-switching is a widespread linguistic phenomenon among multilingual speakers that involves switching between two or more languages in the course of a single conversation or within a single sentence (Auer, 2013). Building speech and language technologies to handle code-switching has become a fairly active area of research and presents a number of interesting technical challenges (Çetinoglu et al., 2016). Language models for code-switched text is an important problem with implications to downstream applications such as speech recognition and machine translation of code-switched data. A natural choice for building such language models would be to use recurrent neural networks (RNNs) (Mikolov et al., 2010), which yield state-of-the-art language models in the case of monolingual text. In this work, we explore mechanisms that can significantly improve upon such a baseline when applied to code-switched text. Specifically, we develop two such mechanisms:

- We alter the structure of an RNN unit to include separate components that focus on each language in code-switched text separately while coordinating with each other to retain contextual information across code-switch boundaries. Our new model is called a *Dual RNN Language Model* (D-RNNLM), described in Section 2.
- We propose using *same-source pretraining* – i.e., pretraining the model using data sampled from a generative model which is itself trained on the given training data – before training the model on the same training data (see Section 3). We find this to be a surprisingly effective strategy.

We study the improvements due to these techniques under various settings (e.g., with and without access to monolingual text in the candidate languages for pretraining). We use perplexity as a proxy to measure the quality of the language model, evaluated on code-switched text in English and Mandarin from the SEAME corpus. Both the proposed techniques are shown to yield *significant perplexity improvements (up to 13% relative)* over different baseline RNNLM models (trained with a number of additional resources). We also explore how to combine the two techniques effectively.

Related Work: Adel et al. (2013) was one of the first works to explore the use of RNNLMs for code-switched text. Many subsequent works explored the use of external sources to enhance code-switched LMs, including the use of part-of-speech (POS) tags, syntactic and semantic features (Yeh et al., 2010; Adel et al., 2014, 2015) and the use of machine translation systems to generate synthetic text (Vu et al., 2012). Prior work has also explored the use of interpolated LMs trained separately on monolingual texts (Bhuvanagiri and Kopparapu, 2010; Imseng et al., 2011; Li et al., 2011; Baheti et al., 2017). Linguistic constraints governing code-switching have also

---

*Joint first authors

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3078–3083
Brussels, Belgium, October 31 - November 4, 2018.

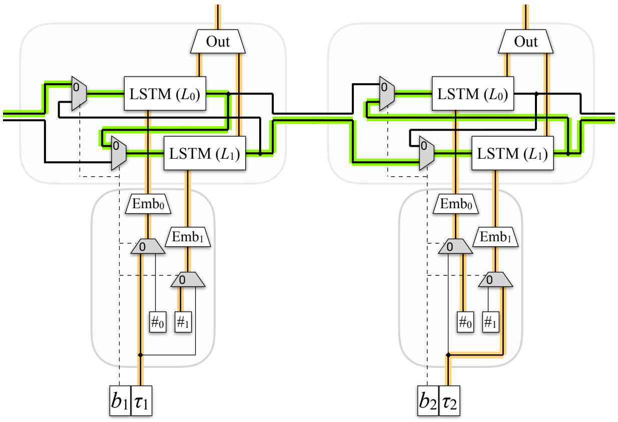

Figure 1: Illustration of the dual RNNLM (see the text for a detailed description). The highlighted left-to-right path (in green) indicates the flow of state information, when $b_1 = 0$ and $b_2 = 1$ (corresponding to token τ_1 belonging to language L_0 and τ_2 belonging to L_1). The highlighted bottom-to-top path (in orange) indicates the inputs and outputs.

been used as explicit priors to model when people switch from one language to another. Following this line of enquiry, (Chan et al., 2004) used grammar rules to model code-switching; (Li and Fung, 2013, 2014) incorporated syntactic constraints with the help of a code-switch boundary prediction model; (Pratapa et al., 2018) used a linguistically motivated theory to create grammatically consistent synthetic code-mixed text.

2 Dual RNN Language Models

Towards improving the modeling of code-switched text, we introduce Dual RNN Language Models (D-RNNLMs). The philosophy behind D-RNNLMs is that two different sets of neurons will be trained to (primarily) handle the two languages. (In prior work (Garg et al., 2018), we applied similar ideas to build dual N-gram based language models for code-switched text.)

As shown in Figure 1, the D-RNNLM consists of a “Dual LSTM cell” and an input encoding layer. The Dual LSTM cell, as the name indicates, has two long short-term memory (LSTM) cells within it. The two LSTM cells are designated to accept input tokens from the two languages L_0 and L_1 respectively, and produce an (unnormalized) output distribution over the tokens in the same language. When a Dual LSTM cell is invoked with an input token τ, the two cells will be invoked sequentially. The first (upstream) LSTM cell corresponds to the language that τ belongs to, and gets τ as its input. It passes on the resulting state to the downstream LSTM cell (which takes a dummy token as input). The unnormalized outputs from the two cells are combined and passed through a soft-max operation to obtain a distribution over the union of the tokens in the two languages. Figure 1 shows a circuit representation of this configuration, using multiplexers (shaded units) controlled by a selection bit b_i such that the i^{th} token τ_i belongs to L_{b_i}.

The input encoding layer also uses multiplexers to direct the input token to the upstream LSTM cell. Two dummy tokens $\#_0$ and $\#_1$ are added to L_0 and L_1 respectively, to use as inputs to the downstream LSTM cell. The input tokens are encoded using an embedding layer of the network (one for each language), which is trained along with the rest of the network to minimize a cross-entropy loss function.

The state-update and output functions of the Dual LSTM cell can be formally described as follows. It takes as input (b, τ) where b is a bit and τ is an input token, as well as a state vector of the form (h_0, h_1) corresponding to the state vectors produced by its two constituent LSTMs. Below we denote the state-update and output functions of these two LSTMs as $H_b(\tau, h)$ and $O_b(\tau, h)$ (for $b = 0, 1$):

$$H((b,\tau),(h_0,h_1)) = (h_0', h_1') \qquad \text{where}$$

$$(h_0', h_1') = \begin{cases} (H_0(\tau,h_0), H_1(\#_1,h_0')) & \text{if } b=0 \\ (H_0(\#_0,h_1'), H_1(\tau,h_1)) & \text{if } b=1 \end{cases}$$

$$O((b,\tau),(h_0,h_1)) = \text{softmax}(o_0,o_1) \qquad \text{where}$$

$$(o_0, o_1) = \begin{cases} (O_0(\tau,h_0), O_1(\#_1,h_0')) & \text{if } b=0 \\ (O_0(\#_0,h_1'), O_1(\tau,h_1)) & \text{if } b=1. \end{cases}$$

Note that above, the inputs to the downstream LSTM functions H_{1-b} and O_{1-b} are expressed in terms of h_b' which is produced by the upstream LSTM.

3 Same-Source Pretraining

Building robust LMs for code-switched text is challenging due to the lack of availability of large amounts of training data. One solution is to artificially generate code-switched to augment the training data. We propose a variant of this approach – called same-source pretraining – in which the actual training data itself is used to train a generative model, and the data sampled from this model is used to pretrain the language model.

Same-source pretraining can leverage powerful training techniques for generative models to train a

	Train	Dev	Test
# Utterances	$74,927$	$9,301$	$9,552$
# Tokens	$977,751$	$131,230$	$114,546$
# English Tokens	$316,726$	$30,154$	$50,537$
# Mandarin Tokens	$661,025$	$101,076$	$64,009$

Table 1: Statistics of data splits derived from SEAME.

language model. We note that the generative models by themselves are typically trained to minimize a different objective function (e.g., a discrimination loss) and need not perform well as language models.[*]

Our default choice of generative model will be an RNN (but see the end of this paragraph). To complete the specification of same-source pretraining, we need to specify how it is trained from the given data. Neural language models trained using the maximum likelihood training paradigm tend to suffer from the *exposure bias* problem during inference when the model generates a text sequence by conditioning on previous tokens that may never have appeared during training. Scheduled sampling (Bengio et al., 2015) can help bridge this gap between the training and inference stages by using model predictions to synthesize prefixes of text that are used during training, rather than using the actual text tokens. A more promising alternative to generate text sequences was recently proposed by Yu et al. (2017) where sequence generation is modeled in a generative adversarial network (GAN) based framework. This model – referred to as "SeqGAN" – consists of a generator RNN and a discriminator network trained as a binary classifier to distinguish between real and generated sequences. The main innovation of SeqGAN is to train the generative model using policy gradients (inspired by reinforcement learning) and use the discriminator to determine the reward function. We experimented with using both naïve and scheduled sampling based training; using SeqGAN was a consistently better choice (5 points or less in terms of test perplexities) compared to these two sampling methods. As such, we use SeqGAN as our training method for the generator. We also experiment with replacing the RNN with a Dual RNN as the generator in the SeqGAN training and observe small but consistent reductions in perplexity.

4 Experiments and Results

Dataset Preparation: For our experiments, we use code-switched text from the SEAME corpus (Lyu et al., 2010) which contains conversational speech in Mandarin and English. Since there is no standardized task based on this corpus, we construct our own training, development and test sets using a random 80-10-10 split. Table 1 shows more details about our data sets. (Speakers were kept disjoint across these datasets.)

Evaluation Metric: We use token-level perplexity as the evaluation metric where tokens are words in English and characters in Mandarin. The SEAME corpus provides word boundaries for Mandarin text. However, we used Mandarin characters as individual tokens since a large proportion of Mandarin words appeared very sparsely in the data. Using Mandarin characters as tokens helped alleviate this issue of data sparsity; also, applications using Mandarin text are typically evaluated at the character level and do not rely on having word boundary markers (Vu et al., 2012).

Outline of Experiments: Section 4.1 will explore the benefits of both our proposed techniques – (1) using D-RNNLMs and (2) using text generated from SeqGAN for pretraining – in isolation and in combination. Section 4.2 will introduce two additional resources (1) monolingual text for pretraining and (2) a set of syntactic features used as additional input to the RNNLMs that further improve baseline perplexities. We show that our proposed techniques continue to outperform the baselines albeit with a smaller margin. All these perplexity results have been summarized in Table 2.

4.1 Improvements Over the Baseline

This section focuses only on the numbers listed in the first two columns of Table 2. The Baseline model is a 1-layer LSTM LM with 512 hidden nodes, input and output embedding dimensionality of 512, trained using SGD with an initial learning rate of 1.0 (decayed exponentially after 80 epochs at a rate of 0.98 till 100 epochs) The development and test set perplexities using the baseline are 89.60 and 74.87, respectively.

The D-RNNLM is a 1-layer language model with each LSTM unit having 512 hidden nodes. The training paradigm is similar to the above-mentioned setting for the baseline model.[†] We see

---

[*] In our experiments, we found the preplexity measures for the generative models to be an *order of magnitude* larger than that of the LMs we construct.

[†] D-RNNLMs have a few additional parameters. How-

	w/o syntactic features				with syntactic features			
	w/o mono. data		with mono. data		w/o mono. data		with mono. data	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test
Baseline	89.60	74.87	74.06	61.66	81.87	68.23	71.04	59.00
D-RNNLM	88.68	72.29	72.41	60.73	81.01	66.26	70.83	59.04
With RNNLM SeqGAN	79.16	65.96	72.51	60.56	77.30	63.75	68.43	55.71
With D-RNNLM SeqGAN	**78.63**	**65.41**	**72.33**	**60.30**	**77.19**	**63.63**	**67.79**	**55.60**

Table 2: Development set and test set perplexities using RNNLMs and D-RNNLMs with various pretraining strategies.

consistent improvements in test perplexity when comparing a D-RNNLM with an RNNLM (i.e. 74.87 drops to 72.29).[‡]

Next, we use text generated from a SeqGAN model to pretrain the RNNLM.[§] We use our best trained RNNLM baseline as the generator within SeqGAN. We sample 157,440 sentences (with a fixed sentence length of 20) from the SeqGAN model; this is thrice the amount of code-switched training data. We first pretrain the baseline RNNLM with this sampled text, before training it again on the code-switched text. This gives significant reductions in test perplexity, bringing it down to 65.96 (from 74.87).

Finally, we combine both our proposed techniques by replacing the generator with our best-trained D-RNNLM within SeqGAN. Although there are other ways of combining both our proposed techniques, e.g. pretraining a D-RNNLM using data sampled from an RNNLM SeqGAN, we found this method of combination to be most effective. We see modest but consistent improvements with D-RNNLM SeqGAN over RNNLM SeqGAN in Table 2, further validating the utility of D-RNNLMs.

4.2 Using Additional Resources

We employed two additional resources to further improve our baseline models. First, we used monolingual text in the candidate languages to pretrain the RNNLM and D-RNNLM models. We used transcripts from the Switchboard corpus[¶] for English; AIShell[‖] and THCHS30[**] corpora for Mandarin monolingual text data. This resulted in a total of ≈3.1 million English tokens and ≈2.5 million Mandarin tokens. Second, we used an additional set of input features to the RNNLMs and D-RNNLMs that were found to be useful for code-switching in prior work (Adel et al., 2014). The feature set included part-of-speech (POS) tag features and Brown word clusters (Brown et al., 1992), along with a language ID feature. We extracted POS tags using the Stanford POS-tagger[††] and we clustered the words into 70 classes using the unsupervised clustering algorithm by Brown et al. (1992) to get Brown cluster features.

The last six columns in Table 2 show the utility of using either one of these resources or both of them together (shown in the last two columns). The perplexity reductions are largest (compared to the numbers in the first two columns) when combining both these resources together. Interestingly, all the trends we observed in Section 4.1 still hold. D-RNNLMs still consistently perform better than their RNNLM counterparts and we obtain the best overall results using D-RNNLM SeqGAN.

5 Discussion and Analysis

	Eng-Eng	Eng-Man	Man-Eng	Man-Man
RNNLM	133.18	157.18	2617.28	34.98
D-RNNLM	140.37	151.38	2452.16	32.89
Mono RNNLM	101.61	181.28	2510.48	30.00
Mono D-RNNLM	101.66	156.44	2442.81	29.64
RNNLM SeqGAN	120.28	154.44	2739.85	30.40
D-RNNLM SeqGAN	120.26	149.68	2450.85	30.60

Table 3: Decomposed perplexities on the development set on all four types of tokens from various models.

Table 3 shows how the perplexities on the development set from six of our prominent models decompose into the perplexities contributed by English tokens preceded by English tokens (Eng-Eng), Eng-Man, Man-Eng and Man-Man tokens. This analysis reveals a number of interesting observations. 1) The D-RNNLM mainly improves over the baseline on the "switching tokens", Eng-

---

ever, increasing the capacity of an RNNLM to exactly match this number makes its test perplexity worse; RNNLM with 720 hidden units gives a development set perplexity of 91.44 and 1024 hidden units makes it 91.46.

[‡] Since D-RNNLMs use language ID information, we also trained a baseline RNNLM with language ID features; this did not help reduce the baseline test perplexities. In future work, we will explore alternate LSTM-based models that incorporate language ID information (Chandu et al., 2018)

[§] To implement SeqGAN, we use code from `https://github.com/LantaoYu/SeqGAN`.

[¶] http://www.openslr.org/5/

[‖] http://www.openslr.org/33/

[**] http://www.openslr.org/18/

[††] `https://nlp.stanford.edu/software/tagger.shtml`

Man and Man-Eng. 2) The RNNLM with monolingual data improves most over the baseline on "the monolingual tokens", Eng-Eng and Man-Man, but suffers on the Eng-Man tokens. The D-RNNLM with monolingual data does as well as the baseline on the Eng-Man tokens and performs better than "Mono RNNLM" on all other tokens. 3) RNNLM SeqGAN suffers on the Man-Eng tokens, but helps on the rest; in contrast, D-RNNLM SeqGAN helps on *all tokens* when compared with the baseline.

	SeqGAN-RNNLM	SeqGAN-DLM
Bigram	25.57	31.33
Trigram	75.88	83.86
Quadgram	137.98	145.71

Table 4: Percentage of new n-grams generated.

As an additional measure of the quality of text generated by RNNLM SeqGAN and D-RNNLM SeqGAN, in Table 4, we measure the diversity in the generated text by looking at the increase in the number of unique n-grams with respect to the SEAME training text. D-RNNLM SeqGAN is clearly better at generating text with larger diversity, which could be positively correlated with the perplexity improvements shown in Table 2.

While we do not claim same-source pretraining may be an effective strategy in general, we show it is useful in low training-data scenarios. Even with only $\frac{1}{16}$th of the original SEAME training data used for same-source pretraining, development and test perplexities are reduced to 84.45 and 70.59, respectively (compared to 79.16 and 65.96 using the entire training data).

6 Conclusion

D-RNNLMs and same-source pretraining provide significant perplexity reductions for code-switched LMs. These techniques may be of more general interest. Leveraging generative models to train LMs is potentially applicable beyond code-switching; D-RNNLMs could be generalized beyond LMs, e.g. speaker diarization. We leave these for future work to explore.

7 Acknowledgments

The authors thank the anonymous reviewers for their helpful comments and suggestions. The last author gratefully acknowledges financial support from Microsoft Research India (MSRI) for this project.

References

Heike Adel, Dominic Telaar, Ngoc Thang Vu, Katrin Kirchhoff, and Tanja Schultz. 2014. Combining recurrent neural networks and factored language models during decoding of code-switching speech. In *Proceedings of Interspeech*.

Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz. 2015. Syntactic and semantic features for code-switching factored language models. *Proceedings of IEEE Transactions on Audio, Speech, and Language Processing*, 23(3):431–440.

Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz. 2013. Recurrent neural network language modeling for code switching conversational speech. In *Proceedings of ICASSP*, pages 8411–8415. IEEE.

Peter Auer. 2013. *Code-switching in conversation: Language, interaction and identity*. Routledge.

Ashutosh Baheti, Sunayana Sitaram, Monojit Choudhury, and Kalika Bali. 2017. Curriculum design for code-switching: Experiments with language identification and language modeling with deep neural networks. In *Proceedings of ICON*, pages 65–74.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of NIPS*, pages 1171–1179.

K Bhuvanagiri and Sunil Kopparapu. 2010. An approach to mixed language automatic speech recognition. *Proceedings of Oriental COCOSDA, Kathmandu, Nepal*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Özlem Çetinoglu, Sarah Schulz, and Ngoc Thang Vu. 2016. Challenges of computational processing of code-switching. *Proceedings of EMNLP*, page 1.

Joyce YC Chan, PC Ching, Tan Lee, and Helen M Meng. 2004. Detection of language boundary in code-switching utterances by bi-phone probabilities. In *Proceedings of International Symposium on Chinese Spoken Language Processing*, pages 293–296. IEEE.

Khyathi Chandu, Thomas Manzini, Sumeet Singh, and Alan W Black. 2018. Language informed modeling of code-switched text. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 92–97.

Saurabh Garg, Tanmay Parekh, and Preethi Jyothi. 2018. Dual language models for code mixed speech recognition. In *Proceedings of Interspeech*.

David Imseng, Hervé Bourlard, Mathew Magimai Doss, and John Dines. 2011. Language dependent universal phoneme posterior estimation for mixed language speech recognition. In *Proceedings of ICASSP*, pages 5012–5015.

Ying Li and Pascale Fung. 2013. Improved mixed language speech recognition using asymmetric acoustic model and language model with code-switch inversion constraints. In *Proceedings of ICASSP*, pages 7368–7372. IEEE.

Ying Li and Pascale Fung. 2014. Language modeling with functional head constraint for code switching speech recognition. In *Proceedings of EMNLP*.

Ying Li, Pascale Fung, Ping Xu, and Yi Liu. 2011. Asymmetric acoustic modeling of mixed language speech. In *Proceedings of ICASSP*, pages 5004–5007.

Dau-Cheng Lyu, Tien-Ping Tan, Eng-Siong Chng, and Haizhou Li. 2010. An analysis of a Mandarin-English code-switching speech corpus: SEAME. *Proceedings of Age*, 21:25–8.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech*.

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of ACL*, volume 1, pages 1543–1553.

Ngoc Thang Vu, Dau-Cheng Lyu, Jochen Weiner, Dominic Telaar, Tim Schlippe, Fabian Blaicher, Eng-Siong Chng, Tanja Schultz, and Haizhou Li. 2012. A first speech recognition system for Mandarin-English code-switch conversational speech. In *Proceedings of ICASSP*, pages 4889–4892. IEEE.

Ching Feng Yeh, Chao Yu Huang, Liang Che Sun, and Lin Shan Lee. 2010. An integrated framework for transcribing Mandarin-English code-mixed lectures with improved acoustic and language modeling. In *Proceedings of Chinese Spoken Language Processing (ISCSLP), 2010 7th International Symposium on*, pages 214–219. IEEE.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of AAAI*, pages 2852–2858.

Part-of-Speech Tagging for Code-Switched, Transliterated Texts without Explicit Language Identification

Kelsey Ball
kelseytaylorball@gmail.com

Dan Garrette
Google Research
dhgarrette@google.com

Abstract

Code-switching, the use of more than one language within a single utterance, is ubiquitous in much of the world, but remains a challenge for NLP largely due to the lack of representative data for training models. In this paper, we present a novel model architecture that is trained exclusively on monolingual resources, but can be applied to unseen code-switched text at inference time. The model accomplishes this by jointly maintaining separate word representations for each of the possible languages—or scripts in the case of transliteration—allowing each to contribute to inferences without forcing the model to commit to a language. Experiments on Hindi-English part-of-speech tagging demonstrate that our approach outperforms standard models when training on monolingual text without transliteration, and testing on code-switched text with alternate scripts.

1 Introduction

Code-switching,[1] the linguistic phenomenon of switching between more than one language within a single utterance, is ubiquitous in multilingual societies worldwide, but presents particular challenges for even the most standard NLP tasks. For example, state-of-the-art neural part-of-speech (POS) taggers are trained on POS-annotated data to learn the relationships between words and tags, and rely on word embeddings trained from large (usually unannotated) corpora to capture information about each word's typical contexts (Ling et al., 2015; Wang et al., 2015). But while monolingual corpora, both annotated and unannotated, are relatively easy to acquire, code-switched text is much more difficult to collect in large quantities. Annotated code-switched texts in particular are not only expensive to create, requiring particularly skilled bilingual linguists as annotators, but they will never exist for the complete set of pairs (or triples, etc.) of languages between which an individual may switch. Further complicating the problem is that texts featuring code-switching also tend to feature *transliteration*—writing in a non-standard script—to avoid having to switch keyboard settings mid-sentence. When Hindi is transliterated from Devanagari to Latin script, tokens are no longer easy to distinguish by language.[2] Previous work addresses this problem by first attempting to identify the correct language and form of a word before feeding it into an appropriate monolingual tagger (Vyas et al., 2014; Solorio and Liu, 2008; Sharma et al., 2016). As is typically the case in NLP, such pipelines suffer from the problem of cascading errors; e.g., failures of the language identification will cause problems in the tag prediction (Barman et al., 2016). Other approaches have trained supervised models on POS-annotated, code-switched data (Jamatia et al., 2015; Ghosh et al., 2016; Gupta et al., 2017; Barman et al., 2016; Sequiera et al., 2015, *inter alia*), resources which are expensive to create and unavailable for most language pairs.

In this work, we present a novel POS-tagging model architecture that can be trained exclusively on available monolingual standard-orthography resources, but that can be applied to texts that contain code-switching and transliteration. Because our model learns from monolingual training data, we are able to learn a code-switched POS-tagger for any combination of languages with POS-annotated data, regardless of whether any existing

---

[1] The terms *code-mixing* and *-switching* are used for similar phenomena; for simplicity, we use only *code-switching*.

[2] We will use the term *transliterated* to imply conversion from a standard script to a non-standard script, e.g., Devanagari to Latin for Hindi, and *detransliterated* to imply the reverse, and, for simplicity, when we say *written in Devanagari* or *Latin*, we mean Devanagari or Latin *script*.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3084–3089
Brussels, Belgium, October 31 - November 4, 2018.

(annotated or unannotated) code-switched corpus exists. By allowing multiple word embeddings (one for each of the possible languages) to jointly represent each token, we avoid explicit language identification and the resulting error propagation.

2 Task Setup

Training data We train our POS tagger using the Universal Dependencies (UD) treebanks for Hindi and English, which are annotated for part-of-speech (Nivre et al., 2016).[3] The English treebank contains roughly 22K sentences; the Hindi contains roughly 16K sentences. Sentences in both languages are taken from news text and similarly formal sources, and the Hindi is written entirely in Devanagari.

Test data We evaluated our POS tagging models on the code-switched Twitter data released by Bhat et al. (2018), which is annotated using the UD POS tag set.[4] All tweets are written entirely in Latin script and have at least one token of both Hindi and English, with 45% of the total tokens being Hindi, 38% English, 17% labeled as universal, mixed, named entity, or acronym.

Word embeddings For monolingual word embeddings, we use *indic-word2vec*,[5] for which the English embeddings are trained from 280M sentences and Hindi from 40M. The word vectors are learned using a skip-gram model with negative sampling, implemented in the word2vec toolkit (Mikolov et al., 2013). We use the 50,000 most frequent words in each vocabulary.

Transliteration To convert words between scripts, we make use of the *Indictrans* open-source transliterator,[6] which is able to convert, bidirectionally, between Latin and the scripts of Indic languages (Bhat et al., 2015). Because Hindi does not have a formal system for transliteration, those writing Hindi in Latin—or English in Devanagari—tend to write phonetically according to their own conventions. Thus, we make use of the ability of Indictrans to generate multiple transliteration alternatives, simulating the variation in Hindi transliterated writing.

3 Model

The central challenge of our task is to be able to train on data that differs drastically from the text seen at inference time. Because of the inherent limitations on what kinds of data are available—annotations and word embeddings are generally only available on monolingual data written in standard scripts—we are interested in designing a model that can be trained on monolingual sentences in which English is written in Latin script and Hindi in Devanagari, but is able to predict POS tags on code-mixed text in which both English and Hindi are written in Latin script.

We accomplish this by representing input words not as a single vector (or as a word embedding affixed to sub-word representations), but as a joint representation that maintains embeddings for each language such that they can be learned and updated during training from the monolingual data, and mixed appropriately at inference time without having to commit to one or the other language. As seen in Figure 1, our model represents an input as the concatenation of four parts: a Hindi word embedding, an English word embedding, the output of a bi-LSTM over Hindi character embeddings, and a bi-LSTM over English characters.

Since our model is trained on monolingual Hindi and English sentences, when a token is encountered during training, we know what language it is in. If we encounter an English sentence, then for each token, we look up its word embedding in the appropriate monolingual vector space to use in the English side of the full representation, and we use the word's characters as input to the English character LSTM. For Hindi sentences, the procedure is similar but for the other side of the full representation. In either case, when we are training on a word in one language, we use zero vectors as the inputs for the other language's side of the full input representation.

At inference time, we will encounter only Latin script words, but we will not know the language of any token. Therefore, we use our joint-representation model to represent the token as both languages simultaneously. The Latin word is looked up in the English word embeddings to use as input to the English side, and the Latin characters are used for the English character LSTM. For the Hindi side we detransliterate the input word to Devanagari and perform the same actions.

The design of our model has a few important

---

[3] `universaldependencies.org/`

[4] `github.com/CodeMixed UniversalDependencies/UD_Hindi_English`

[5] `bitbucket.org/irshadbhat/indic-word2vec-embeddings`

[6] `github.com/libindic/indic-trans`

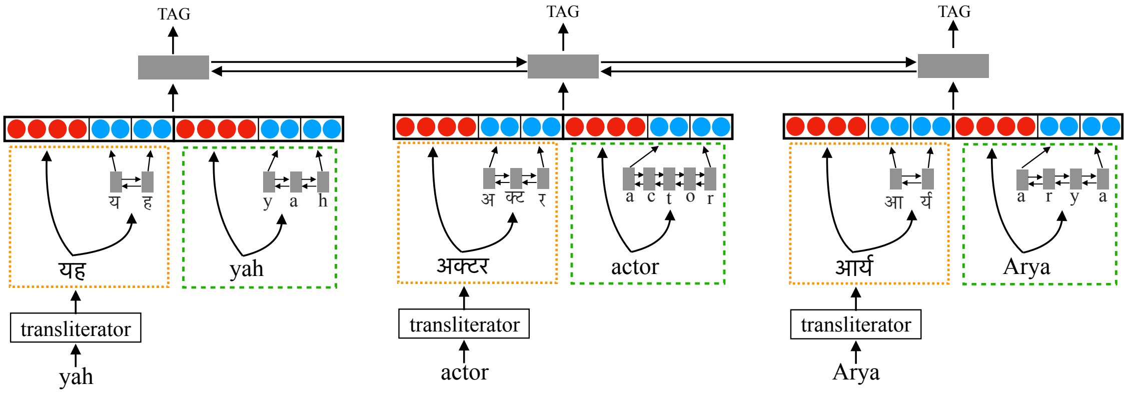

Figure 1: Our model. All inputs are treated simultaneously as both languages. At training time, for a (monolingual) Hindi input, only the contents of the orange/dotted boxes apply, and the English half of the vector is all zeros; for (monolingual) English, only the green/dashed boxes apply. At inference time, where the language of each token is unknown, the full figure applies.

advantages. First, because of Hindi's lack of a formalized transliteration system, Latin-script Hindi tends to have a very high amount of spelling variation. In our model (as opposed to the baseline model described below), the script conversion always goes from Latin to Devanagari, which naturally allows us to take unpredictable variant spellings and collapse them into standardized spellings, making it more likely for us to find the word in the embeddings listing. Second, there are some types of words for which it does not make sense to say that they are in one language or another. For example, a person's name might be equally valid as Devanagari Hindi or Latin English. In these cases, our model would allow both to influence a prediction as well as the rest of the bi-LSTM chain.

Finally, our model allows for the incorporation of external language prediction information, including soft predictions. To test this, we ran experiments in which we used a fully-unsupervised token-level language identification approach based on Rijhwani et al. (2017) to estimate a probability distribution over language labels for each token in the test data. Our approach was to run Forward-Backward on an HMM that was initialized to prefer same-language transitions and whose emissions were initialized using Laplace-smoothed maximum-likelihood-estimated character n-gram probabilities from each language's monolingual corpus (Dempster et al., 1977; Kupiec, 1992). We used the induced language probabilities to weight each side of the input representation at inference time: if the unsupervised model's output said that a particular test token's probability of being Hindi was 70%, then we would multiply the Hindi side of the input vector by 0.7 and the English side by 0.3, thus instructing the model to let the Hindi word embedding (and character-LSTM output) exert more influence during POS prediction.

4 Experiments

In our evaluation, we used the framework of a basic bi-LSTM tagger to compare our model for generating representations of tokens that maintain language ambiguity against a standard model that uses a single word embedding. In this section, we explain the experimental setup, baseline approach, and our experimental variants.[7]

4.1 POS tagger framework

For all of our experiments, we follow the basic structure of the bi-LSTM architecture of Bhat et al. (2018): word and sub-word embeddings are concatenated to form the input representations of each token, and connected together by a bi-LSTM that outputs POS predictions via multilayer perceptron and softmax layers (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005; Ling et al., 2015; Wang et al., 2015). Sub-word embeddings are the outputs of a bi-LSTM over characters (20-dim); word embeddings are initialized with *indic-word2vec* (64-dim) embeddings. Both character and word embeddings are updated during training. The bi-LSTM has 100

---

[7] Our code, implemented in DyNet (Neubig et al., 2017), can be found at https://github.com/kelseyball/cs-transliterated-pos-tagging.

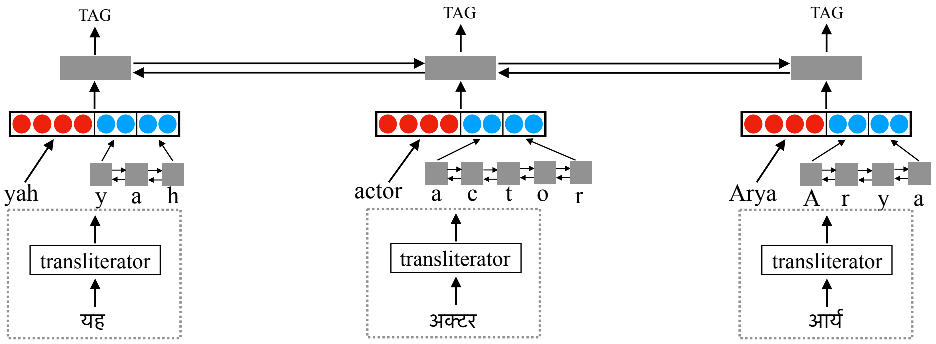

Figure 2: Baseline model. All inputs are treated as if they are one language. The contents of the dotted boxes only apply at training time and only to (monolingual) Hindi inputs since annotated Devanagari Hindi data must be converted to Latin script; at inference time, inputs are always in Latin script.

hidden states. We use an SGD trainer with learning rate=0.1 and dropout of 0.3 across 20 epochs. We train the model it by pooling POS-annotated English and Hindi UD sentences; the English is entirely Latin script, and Hindi is entirely Devanagari. Our baseline and experimental models differ in how a token is represented.

4.2 Baseline

Previous work on code-switched POS tagging has achieved accuracies as high as 90.20% (Bhat et al., 2018), but because those kinds of accuracies require supervised, in-domain training data, they are not directly comparable with the scenario that we are concerned with. Therefore, we compare against a natural baseline for our task: use a monolingual tagger and treat everything as if it were a single language. To deal with the problem of separately trained word embeddings, we use the approach and implementation of Artetxe et al. (2018)[8] to transform both sets of monolingual embeddings so that translationally equivalent words have similar embeddings. We also ran the baseline without this transformation, but found that using the transformed embeddings resulted in a 0.5% absolute improvement in accuracy over just using the separately trained embeddings directly.

Training For an English sentence, each token concatenates the English-lookup embedding to the character LSTM output, and similarly for Hindi, with the exception that the word is transliterated before inputting to the character LSTM. Because of the high spelling variation, we selected Latin spellings by sampling uniformly from the top-5 transliteration results. Always taking the 1-best resulted in a 4.3% absolute loss.

Model		**Acc.**
Baseline	(monolingual representation)	70.92
Ours	(multi-language representation)	75.29
Ours	w/ forced language choice	76.04
Ours	w/ languages weighted by HMM	**77.40**
Ours	w/ oracle language choice	80.53
Bhat et al. (2018)	w/ supervision	90.20

Table 1: Results showing POS prediction accuracy. The baseline model treats all inputs as effectively monolingual (§4.2); the middle three lines are our model: giving equal weight to each language's side of the token's representation, using the unsupervised HMM 1-best output to zero-out one language's side of the token's representation, and using the unsupervised HMM probability estimates to weight each side of the token's representation. The oracle line is our model using gold language labels to zero-out one language's side.

Inference Each word is looked up in the English embeddings *and* its detransliteration is looked up in the in the Hindi embeddings. If the word is found in only one, that vector is used, and if found in neither, we use an *unknown word* vector. For a fair comparison to our model, if both lookups return results, we use the 1-best prediction from the unsupervised language identification HMM as discussed in §3 to decide which to use.

4.3 Unsupervised soft language prediction

One feature of our model is its ability to incorporate externally-derived, per-token soft language

---

[8] `github.com/artetxem/vecmap`

predictions as weights at inference time without having to make hard language decisions. To evaluate whether this improves accuracy, we used the fully-unsupervised HMM language prediction (§3) and tested our model using: no weighting, leaving the embeddings untouched; using the HMM's 1-best prediction to zero out the other language's inputs; and multiplying each language's embeddings by its probability estimated by the HMM.

The results of our evaluation can be seen in Table 1. While the best scenario for the baseline tagger gives 70.92% accuracy, the most basic setup for our model yields 75.29%. Informing our model with some fully-unsupervised HMM-derived language information, but forcing a hard language decision, is somewhat beneficial, yielding 76.04%, though it trades off the benefits of flexibility and generality in representation for some extra information. And finally, using that unsupervised information for *soft* weighting performs even better, reaching 77.40%, a total absolute improvement over the baseline of ∼6.5%, clearly demonstrating the value of maintaining ambiguity about a word's language.

5 Conclusion

In this paper, we presented a novel POS-tagging model for code-mixed, transliterated text that avoids the need for expensive, supervised corpora or language-identification pipelines. We demonstrated that avoiding explicit language identification in code-switched contexts is beneficial. Though we emphasize the paucity of annotated code-switched corpora, previous work has demonstrated that a small amount of supervision can go a long way (Garrette and Baldridge, 2013); future work might explore how much supervision is needed to close the gap between our model and Bhat et al. (2018).

While we address some of the most prominent issues with code-switching, our model does not deal with style, formality, or domain mismatches between the formal training data and informal evaluation data. This problem is treated in recent work on POS-tagging for social media text (Owoputi et al., 2013; Gimpel et al., 2011), but these issues have yet to be fully explored in code-switched contexts. We leave these for future work.

Acknowledgements

This work was supported by a Fulbright-Nehru grant from the United States-India Educational Foundation (USIEF) for the first author. We would like to thank Irshad Bhat, Dipti Misra Sharma, Manish Shrivastava, Karl Pichotta, Emily Pitler, Luke Zettlemoyer, Phoebe Mulcaire, Gaurav Singh Tomar, Michael Collins, and the anonymous reviewers for their assistance and feedback.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of AAAI*.

Utsab Barman, Joachim Wagner, and Jennifer Foster. 2016. Part-of-speech tagging of code-mixed social media content: Pipeline, stacking and joint modelling. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*.

Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Manish Shrivastava, and Dipti Misra Sharma. 2018. Universal dependency parsing for Hindi-English code-switching. In *Proceedings of NAACL*.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2015. IIIT-H system submission for FIRE2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1).

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of NAACL*.

Souvick Ghosh, Satanu Ghosh, and Dipankar Das. 2016. Part-of-speech tagging of code-mixed social media text. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of ACL*.

Alex Graves and Jurgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5).

Deepak Kumar Gupta, Shubham Tripathi, Asif Ekbal, and Pushpak Bhattacharyya. 2017. SMPOST: parts of speech tagger for code-mixed Indic social media text. *http://arxiv.org/abs/1702.00167*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed English-Hindi Twitter and Facebook chat messages. In *Proceedings of RANLP*.

Julian Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. *Speech and Language*, 6.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, et al. 2017. Dynet: The dynamic neural network toolkit. *ArXiv e-prints*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of LREC*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Sekhar Maddila. 2017. Estimating code-switching on Twitter with a novel generalized word-level language detection technique. In *Proceedings of ACL*.

Royal Sequiera, Monojit Choudhury, and Kalika Bali. 2015. POS tagging of Hindi-English code mixed text from social media: Some machine learning experiments. In *Proceedings of International Conference on NLP*.

Arnav Sharma, Sakshi Gupta, Raveesh Motlani, Piyush Bansal, Manish Shrivastava, Radhika Mamidi, and Dipti Misra Sharma. 2016. Shallow parsing pipeline for Hindi-English code-mixed social media text. In *Proceedings of NAACL*.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for English-Spanish code-switched text. In *Proceedings of EMNLP*.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. POS tagging of English-Hindi code-mixed social media content. In *Proceedings of EMNLP*.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint abs/1510.06168*.

Zero-shot User Intent Detection via Capsule Neural Networks

Congying Xia[1]*, **Chenwei Zhang**[1]*, **Xiaohui Yan**[2], **Yi Chang**[3,4,5], **Philip S. Yu**[1]
[1]Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607 USA
[2]Huawei Technologies, San Jose, CA 95050 USA
[3]College of Artificial Intelligence, Jilin University, Changchun, China
[4]College of Computer Science and Technology, Jilin University, Changchun, China
[5]Key Laboratory of Symbolic Computation and
Knowledge Engineering of Ministry of Education, China
{cxia8,czhang99,psyu}@uic.edu,
yanxiaohui2@huawei.com, yichang@acm.org

Abstract

User intent detection plays a critical role in question-answering and dialog systems. Most previous works treat intent detection as a classification problem where utterances are labeled with predefined intents. However, it is labor-intensive and time-consuming to label users' utterances as intents are diversely expressed and novel intents will continually be involved. Instead, we study the zero-shot intent detection problem, which aims to detect emerging user intents where no labeled utterances are currently available. We propose two capsule-based architectures: INTENTCAPSNET that extracts semantic features from utterances and aggregates them to discriminate existing intents, and INTENTCAPSNET-ZSL which gives INTENTCAPSNET the zero-shot learning ability to discriminate emerging intents via knowledge transfer from existing intents. Experiments on two real-world datasets show that our model not only can better discriminate diversely expressed existing intents, but is also able to discriminate emerging intents when no labeled utterances are available.

1 Introduction

With the increasing complexity and accuracy of speech recognition technology, companies are striving to deliver intelligent conversation understanding systems as people interact with software agents that run on speaker devices or smart phones via natural language interface (Hoy, 2018). Products like Apple's Siri, Amazon's Alexa and Google Assistant are able to interpret human speech and respond them via synthesized voices.

With recent developments in deep neural networks, user intent detection models (Hu et al., 2009; Xu and Sarikaya, 2013; Zhang et al., 2016; Liu and Lane, 2016; Chen et al., 2016b) are proposed to classify user intents given their diversely expressed utterances in the natural language. The decent performances on intent detection usually come with deep neural network classifiers optimized on large-scale utterances which are human-labeled among existing predefined user intents.

As more features and skills are being added to devices which expand their capabilities to new programs, it is common for voice assistants to encounter the scenario where no labeled utterance of an emerging user intent is available in the training data, as illustrated in Figure 1. Current intent detection methods train classifiers in a supervised fashion and they are good at discriminating existing intents such as Get Weather and Play Music whose labeled utterances are already available. However, these models, by the nature of designs, are incapable to detect utterances of emerging intents like AddToPlaylist and RateABook, since no labeled utterances are available. Moreover, it's labor-intensive and time-consuming to annotate utterances of emerging intents and retrain the whole intent detection model.

Thus, it is imperative to develop intent detection models with the zero-shot learning (ZSL) ability (Lampert et al., 2014; Socher et al., 2013; Changpinyo et al., 2016): the ability to expand classifiers and the intent detection space beyond the existing intents, of which we have labeled utterances during training, to emerging intents, of which no labeled utterances are available.

The research on zero-shot intent detection is still in its infancy. Previous zero-shot learning methods for intent detection utilize external resources such as label ontologies (Ferreira et al., 2015a,b) or manually defined attributes that describe intents (Yazdani and Henderson, 2015) to associate existing and emerging intents, which require extra annotation. Compatibility-based methods for zero-shot intent detection (Chen et al., 2016a; Kumar et al., 2017) assume the capability

---

*Indicates Equal Contribution

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3090–3099
Brussels, Belgium, October 31 - November 4, 2018.

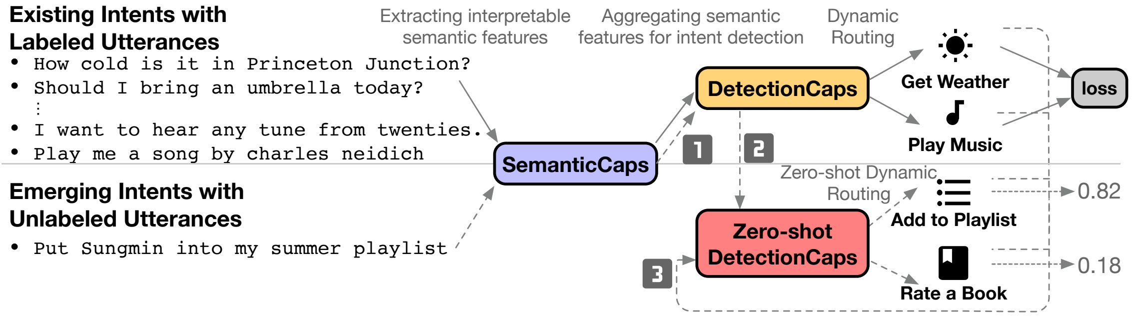

Figure 1: Illustration of the proposed IntentCapsNet-ZSL model for zero-shot intent detection: labeled utterances with existing intents like GetWeather and PlayMusic are used to train an intent detection classifier among existing intents, in which SemanticCaps extract intepretable semantic features and DetectionCaps dynamically aggregate semantic features for intent detection using a novel routing-by-agreement mechanism. For emerging intents, IntentCapsNet-ZSL builds zero-shot DetectionCaps that utilize the (1) outputs of SemanticCaps, (2) the routing information on existing intents from DetectionCaps, and (3) similarities of the emerging intent label to existing intent labels to discriminate emerging intents like AddToPlayist from RateABook. Solid lines indicate the training process and dash lines indicate the zero-shot inference process.

of learning a high-quality mapping from the utterance to its intent directly, so that such mapping can be further capitalized to measure the compatibility of an utterance with emerging intents. However, the diverse semantic expressions may impede the learning of such mapping.

In this work, we make the very first attempt to tackle the zero-shot intent detection problem with a capsule-based (Hinton et al., 2011; Sabour et al., 2017) model. A capsule houses a vector representation of a group of neurons, and the orientation of the vector encodes properties of an object (like the shape/color of a face), while the length of the vector reflects its probability of existence (how likely a face with certain properties exists). The capsule model learns a hierarchy of feature detectors via a routing-by-agreement mechanism: capsules for detecting low-level features (like nose/eyes) send their outputs to high-level capsules (such as faces) only when there is a strong agreement of their predictions to high-level capsules.

The aforementioned properties of capsule models could be quite appealing for text modeling, specifically in this case, modeling the user utterance for intent detection: low-level semantic features such as the get_action, time and city_name contribute to a more abstract intent (GetWeather) collectively. A semantic feature, which may be expressed quite differently among users, can contribute more to one intent than others. The dynamic routing-by-agreement mechanism can be used to dynamically assign a proper contribution of each semantic and aggregate them to get an intent representation.

More importantly, we discover the potential of zero-shot learning ability on the capsule model, which is not yet widely recognized. It makes the capsule model even more suitable for text modeling when no labeled utterances are available for emerging intents. The ability to neglect the disagreed output of low-level semantics for certain intents during routing-by-agreement encourages the learning of generalizable semantic features that can be adapted to emerging intents. For each emerging intent with no labeled utterances, a Zero-shot DetectionCaps is constructed explicitly by using not only semantic features SemanticCaps extracted, but also existing routing agreements from DetectionCaps and similarities of an emerging intent label to existing intent labels.

In summary, the contributions of this work are:

- Expanding capsule neural networks to text modeling, by extracting and aggregating semantics from utterances in a hierarchical manner;
- Proposing a novel and effective capsule-based model for zero-shot intent detection;
- Showing and interpreting the effectiveness of our model on two real-world datasets.

2 Problem Formulation

In this section, we first define related concepts, and formally state the problem.

Intent. An intent is a purpose, or a goal that underlies a user-generated utterance (Watson Assistant, 2017). An utterance can be associated with one or multiple intents. We only consider the basic case that an utterance is with a single intent. However, utterances with multiple intents can be handled by segmenting them into single-intent snippets using sequential tagging tools like CRF (Lafferty et al., 2001), which we leave for future works.

Intent Detection. Given a labeled training dataset where each sample has the following format: (x, y) where x is an utterance and y is its intent label, each training example is associated with one of K existing intents $y \in Y = \{y_1, y_2, ..., y_K\}$. The intent detection task tries to associate an utterance $x_{existing}$ with its correct intent category in the existing intent classes Y.

Zero-shot Intent Detection. Given the labeled training set $\{(x, y)\}$ where $y \in Y$, the zero-shot intent detection task aims to detect an utterance $x_{emerging}$ which belongs to one of L emerging intents $z \in Z = \{z_1, z_2, ..., z_L\}$ where $Y \cap Z = \varnothing$.

3 Approach

We propose two architectures based on capsule models: INTENTCAPSNET that is trained to discriminate among utterances with existing labels, e.g. existing intents for intent detection; INTENTCAPSNET-ZSL that gives zero-shot learning ability to INTENTCAPSNET for discriminating unseen labels, i.e. emerging intents in this case. As shown in Figure 2, the cores of the proposed architectures are three types of capsules: SemanticCaps that extract interpretable semantic features from the utterance, DetectionCaps that aggregate semantic features for intent detection, and Zero-shot DetectionCaps which discriminate emerging intents.

3.1 SemanticCaps

In the original capsule model (Sabour et al., 2017), convolution-based PrimaryCaps are introduced as the first layer to obtain different vectorized features from the raw input image. While in this work, an intrinsically similar motivation is adopted to extract different semantic features from the raw utterance by a new type of capsule named SemanticCaps. Unlike the PrimaryCaps which use convolution operators with a large reception field to extract spacial-proximate features, the SemanticCaps is based on a bi-direction recurrent neural network with multiple self-attention heads, where each self-attention head focuses on certain part of the utterance and extracts a semantic feature that may not be expressed by words in proximity.

Given an input utterance $x = (\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_T})$ of T words, each word is represented by a vector of dimension D_W that can be pre-trained using a skip-gram language model (Mikolov et al., 2013).

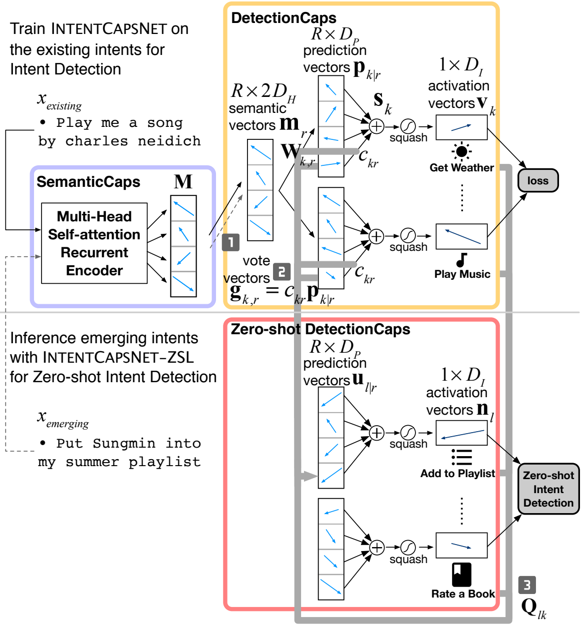

Figure 2: The architecture of INTENTCAPSNET and INTENTCAPSNET-ZSL. During training, utterances with existing intents are fed into the SemanticCaps which output vectorized semantic features, i.e. semantic vectors. Then DetectionCaps combine these features into higher-level prediction vectors and output an activation vector for intent detection on each existing intent. During inference, emerging utterances take advantages of the SemanticCaps trained in INTENTCAPSNET to extract semantic features from the utterance (shown in 1), then the vote vectors on the existing intents are transferred to emerging intents (shown in 2) using similarities between existing and emerging intents (shown in 3). The obtained activation vectors for emerging intents are used for zero-shot intent detection.

A recurrent neural network such as a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) is applied to sequentially encode the utterance into hidden states:

$$\begin{aligned}
\overrightarrow{\mathbf{h}}_t &= \text{LSTM}_{fw}(\mathbf{w}_t, \overleftarrow{\mathbf{h}}_{t-1}), \\
\overleftarrow{\mathbf{h}}_t &= \text{LSTM}_{bw}(\mathbf{w}_t, \overleftarrow{\mathbf{h}}_{t+1}).
\end{aligned} \tag{1}$$

For each word $\mathbf{w_t}$, we concatenate each forward hidden state $\overrightarrow{\mathbf{h}}_t$ obtained from the forward LSTM_{fw} with a backward hidden state $\overleftarrow{\mathbf{h}}_t$ from LSTM_{bw} to obtain a hidden state $\mathbf{h_t}$ for the word $\mathbf{w_t}$. The whole hidden state matrix can be defined as $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T) \in \mathbb{R}^{T \times 2D_H}$, where D_H is the number of hidden units in each LSTM.

Inspired by the success of self-attention mechanisms (Vaswani et al., 2017; Lin et al., 2017) for sentence embedding, we adopt a multi-head self-attention framework where each self-attention head is encouraged to be attentive to a specific semantic feature of the utterance, such as certain sets of keywords or phrases in the utterance: one self-attention may be attentive for the "get" action in GetWeather, while another one may be attentive to city_name in GetWeather: it decides for itself what semantics to be attentive to.

A self-attention weight matrix $\mathbf{A}$ is computed as:

$$\mathbf{A} = \text{softmax}\left(\mathbf{W}_{s2}\tanh\left(\mathbf{W}_{s1}\mathbf{H}^T\right)\right), \quad (2)$$

where $\mathbf{W}_{s1} \in \mathbb{R}^{D_A \times 2D_H}$ and $\mathbf{W}_{s2} \in \mathbb{R}^{R \times D_A}$ are weight matrices for the self-attention. D_A is the hidden unit number of self-attention and R is the number of self-attention heads. The softmax function makes sure for each self-attention head, the attentive scores on all the words sum to one.

A total number of R semantic features are extracted from the input utterance, each from a separate self-attention head: $\mathbf{M} = \mathbf{AH}$, where $\mathbf{M} = (\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_R) \in \mathbb{R}^{R \times 2D_H}$. Each $\mathbf{m}_r$ is a $2D_H-$dimensional semantic vector.

Each semantic vector will have a distinguishable orientation when the objective is properly regularized (details in Equation 6), as we want each attention to be attentive to a unique semantic feature of the utterance. The vector representation adopted in capsules is suitable to portray the low-level semantic properties as well as high-level intents of the utterance, where the orientation of a vector represents semantic/intent properties that may slightly vary depending on the expressions. The capsule encourages the learning of generalizable semantic vectors: less informative semantic properties for one intent may not be penalized by their orientations: they simply possess small norms as they are less likely to exist.

3.2 DetectionCaps

The output of SemanticCaps are low-level vector representations of R different semantic features extracted from the utterances. To combine these features into higher-level representations, we build DetectionCaps that choose different semantic features dynamically so as to form an intent representation for each intent via an unsupervised routing-by-agreement mechanism.

As a semantic feature may contribute differently in detecting different intents, the DetectionCaps first encode semantic features with respect to each intent:

$$\mathbf{p}_{k|r} = \mathbf{m}_r \mathbf{W}_{k,r}, \quad (3)$$

where $k \in \{1, 2, ..., K\}$, $r \in \{1, 2, ..., R\}$. $\mathbf{W}_{k,r} \in \mathbb{R}^{2D_H \times D_P}$ is the weight matrix of the DetectionCaps, $\mathbf{p}_{k|r}$ is the prediction vector of the r-th semantic feature of an existing intent k, and D_P is the dimension of the prediction vector.

Dynamic Routing-by-agreement. The prediction vectors obtained from SemanticCaps route dynamically to DetectionCaps. The DetectionCaps computes a weighted sum over all prediction vectors:

$$\mathbf{s}_k = \sum_r^R c_{kr} \mathbf{p}_{k|r}, \quad (4)$$

where c_{kr} is the coupling coefficient that determines how informative, or how much contribution the r-th semantic feature is to the intent y_k. c_{kr} is calculated by an unsupervised, iterative dynamic routing-by-agreement algorithm (Sabour et al., 2017), which is briefly recalled in Algorithm 1. As shown in this algorithm, b_{kr} is the initial logit representing the log prior probability that a SemanticCap r is coupled to an DetectionCap k.

Algorithm 1 Dynamic routing algorithm

```
procedure DYNAMIC ROUTING($\mathbf{p}_{k|r}$, $iter$)
    for all semantic capsule r and intent capsule k: $b_{kr} \leftarrow 0$.
    for $iter$ iterations do
        for all SemanticCaps r: $c_r \leftarrow \text{softmax}(\mathbf{b}_r)$
        for all DetectionCaps k: $s_k \leftarrow \Sigma_r c_{kr} \mathbf{p}_{k|r}$
        for all DetectionCaps k: $v_k = \text{squash}(\mathbf{s}_k)$
        for all SemanticCaps r and DetectionCaps k:
            $b_{kr} \leftarrow b_{kr} + \mathbf{p}_{k|r} \cdot \mathbf{v}_k$
    end for
    Return $\mathbf{v}_k$
end procedure
```

The squashing function $\text{squash}(\cdot)$ is applied on $\mathbf{s}_k$ to get an activation vector $\mathbf{v}_k$ for each existing intent class k:

$$\mathbf{v}_k = \frac{\|\mathbf{s}_k\|^2}{1 + \|\mathbf{s}_k\|^2} \frac{\mathbf{s}_k}{\|\mathbf{s}_k\|}, \quad (5)$$

where the orientation of the activation vector $\mathbf{v}_k$ represents intent properties while its norm indicates the activation probability. The dynamic routing-by-agreement mechanism assigns low c_{kr} when there is inconsistency between $p_{k|r}$ and v_k, which ensures the outputs of the SemanticCaps get sent to appropriate subsequent DetectionCaps.

Max-margin Loss for Existing Intents. The loss function considers both the max-margin loss on each labeled utterance, as well as a regularization term that encourages each self-attention head to be attentive to a different semantic feature of the utterance:

$$\mathcal{L} = \sum_{k=1}^{K} \{ [\![y = y_k]\!] \cdot \max(0, m^+ - \|\mathbf{v}_k\|)^2 + \lambda\, [\![y \neq y_k]\!] \cdot \max(0, \|\mathbf{v}_k\| - m^-)^2 \} + \alpha ||\mathbf{A}\mathbf{A}^T - I||_F^2, \tag{6}$$

where $[\![]\!]$ is an indicator function, y is the ground truth intent label for the utterance x, λ is a down-weighting coefficient, m^+ and m^- are margins. α is a non-negative trade-off coefficient that encourages the discrepancies among different attention heads.

3.3 Zero-shot DetectionCaps

To detect emerging intents effectively, Zero-shot DetectionCaps are designed to transfer knowledge from existing intents to emerging intents.

Knowledge Transfer Strategies. As SemanticCaps are trained to extract semantic features from utterances with various existing intents, a self-attention head which has similar extraction behavior among existing and emerging intents may help transfer knowledge. For example, a self-attention head that extracts the "play" action mentioned by `turn on/I want to hear` in the beginning of an utterance for PlayMusic is helpful if it is also attentive to expressions for the "add" action like `add/I want to have` in the beginning of an utterance with an emerging intent AddtoPlaylist.

The coupling coefficient c_{kr} learned by DetectionCaps in a totally unsupervised fashion embodies rich knowledge of how informative r-th semantic is to the existing intent k. We can capitalize on the existing routing information for emerging intents. For example, how the word `play` routes to GetWeather can be helpful in routing the word `add` to AddtoPlaylist.

The intent labels also contain knowledge of how two intents are similar with each other. For example, an emerging intent AddtoPlaylist can be closer to one existing intent PlayMusic than GetWeather due to the proximity of the embedding of Playlist to Play or Music, than Weather.

Build Vote Vectors. As the routing information and the semantic extraction behavior are strongly coupled (c_{kr} is calculated by $\mathbf{p}_{k|r}$ iteratively in Line 4-6 of Algorithm 1) and their products are summarized to get the activation vector v_k for intent k (Line 5-6 of Algorithm 1), we denote vectors before summation as vote vectors:

$$\mathbf{g}_{k,r} = c_{kr}\mathbf{p}_{k|r}, \tag{7}$$

where $\mathbf{g}_{k,r}$ is the r-th vote vector for an existing intent k.

Zero-shot Dynamic Routing. The zero-shot dynamic routing utilizes vote vectors from existing intents to build intent representations for emerging intents via a similarity metric between existing intents and emerging intents.

Since there are K existing intents and L emerging intents, the similarities between existing and emerging intents form a matrix $\mathbf{Q} \in \mathbb{R}^{L \times K}$. Specifically, the similarity between an emerging intent $z_l \in Z$ and an existing intent $y_k \in Y$ is computed as:

$$q_{lk} = \frac{exp\left\{-d\left(\mathbf{e}_{z_l}, \mathbf{e}_{y_k}\right)\right\}}{\sum_{k=1}^{K} exp\left\{-d\left(\mathbf{e}_{z_l}, \mathbf{e}_{y_k}\right)\right\}}, \tag{8}$$

where

$$d\left(\mathbf{e}_{z_l}, \mathbf{e}_{y_k}\right) = \left(\mathbf{e}_{z_l} - \mathbf{e}_{y_k}\right)^T \Sigma^{-1} \left(\mathbf{e}_{z_l} - \mathbf{e}_{y_k}\right). \tag{9}$$

$\mathbf{e}_{z_l}, \mathbf{e}_{y_k} \in \mathbb{R}^{D_I \times 1}$ are intent embeddings computed by the sum of word embeddings of the intent label. Σ models the correlations among intent embedding dimensions and we use $\Sigma = \sigma^2 I$. σ is a hyper-parameter for scaling. The prediction vectors for emerging intents are thus computed as:

$$\mathbf{u}_{l|r} = \sum_{k=1}^{K} q_{lk}\mathbf{g}_{k,r}. \tag{10}$$

We feed the prediction vector $\mathbf{n}_l$ to Algorithm 1 and derive activation vectors $\mathbf{n}_l$ on emerging intents as the output. The final intent representation $\mathbf{n}_l$ for each emerging intent is updated toward the direction where it coincides with representative votes vectors.

We can easily classify the utterance of emerging intents by choosing the activation vector with the largest norm $\hat{z} = \arg\max_{z_l \in Z} \|\mathbf{n}_l\|$.

4 Experiment Setup

To demonstrate the effectiveness of our proposed models, we apply INTENTCAPSNET to detect existing intents in an intent detection task, and use INTENTCAPSNET-ZSL to detect emerging intents in a zero-shot intent detection task.

Datasets. For each task, we evaluate our proposed models by applying it on two real-word

Model	**SNIPS-NLU** (on 5 existing intents)				**CVA** (on 80 existing intents)			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
TFIDF-LR	0.9546	0.9551	0.9546	0.9545	0.7979	0.8104	0.7979	0.7933
TFIDF-SVM	0.9584	0.9586	0.9584	0.9581	0.7989	0.8111	0.7989	0.7942
CNN	0.9595	0.9596	0.9595	0.9595	0.8223	0.8288	0.8223	0.8210
RNN	0.9516	0.9522	0.9516	0.9518	0.8286	0.8330	0.8286	0.8275
GRU	0.9535	0.9535	0.9535	0.9534	0.8239	0.8281	0.8239	0.8216
LSTM	0.9569	0.9573	0.9569	0.9569	0.8319	0.8387	0.8319	0.8306
Bi-LSTM	0.9501	0.9502	0.9501	0.9502	0.8428	0.8479	0.8428	0.8419
Self-attention Bi-LSTM	0.9524	0.9522	0.9524	0.9522	0.8521	0.8590	0.8521	0.8513
INTENTCAPSNET	**0.9621**	**0.9620**	**0.9621**	**0.9620**	**0.9088**	**0.9160**	**0.9088**	**0.9023**

Table 1: Intention detection results using INTENTCAPSNET on two datasets. All the metrics (Accuray, Precision, Recall and F1) are reported using the average value weighted by their support on per class.

datasets: SNIPS Natural Language Understanding benchmark (SNIPS-NLU) and a Commercial Voice Assistant (CVA) dataset. The statistical information on two datasets are shown in Table 2. SNIPS-NLU[1] is an English natural language corpus collected in a crowdsourced fashion to benchmark the performance of voice assistants. CVA is a Chinese natural language corpus collected anonymously from a commercial voice assistant on smart phones.

Dataset	**SNIPS-NLU**	**CVA**
Vocab Size	10,896	1,709
Number of Samples	13,802	9,992
Average Sentence Length	9.05	4
Number of Existing Intents	5	80
Number of Emerging Intents	2	20

Table 2: Dataset statistics.

Baselines. We first compare the proposed capsule-based model INTENTCAPSNET with other text classification alternatives on the detection of existing intents: 1) TFIDF-LR/TFIDF-SVM: we use TF-IDF to represent the utterance and use logistic regression/support vector machine as classifiers. 2) CNN: a convolutional neural network (Kim, 2014) that uses convolution and pooling operations, which is popular for text classification. 3) RNN/GRU/LSTM/BiLSTM: we adopt different types of recurrent neural networks: the vanilla recurrent neural network (RNN), gated recurrent unit (GRU) (Tang et al., 2015), long short-term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997), and bi-directional long short-term memory (Bi-LSTM) (Schuster and Paliwal, 1997). Their last hidden states are used for classification. 4) Self-Attention Bi-LSTM: we apply a Bi-LSTM model with self-attention mechanism (Lin et al., 2017) and the output sentence embedding is used for classification.

We also compare our proposed model INTENTCAPSNET-ZSL with different zero-shot learning strategies: 1) DeViSE (Frome et al., 2013) finds the most compatible emerging intent label for an utterance by learning a linear compatibility function between utterances and intents; 2) CMT (Socher et al., 2013) introduces non-linearity in the compatibility function; CMT and DeViSE are originally designed for zero-shot image classification based on pretrained CNN features. We use LSTM to encode the utterance and adopt their zero-shot learning strategies in our task; 3) CDSSM (Chen et al., 2016a) uses CNN to extract character-level sentence features, where the utterance encoder shares the weights with the label encoder; 4) Zero-shot DNN (Kumar et al., 2017) further improves the performance of CDSSM by using separate encoders for utterances and intent. The proposed model INTENTCAPSNET-ZSL can be seen as a hybrid model: it has the advantages of the compatibility models to model the correlations between utterances and intents directly; it also explicitly derives intent representations for emerging intents without labeled utterances.

Dataset	D_W	D_H	D_A	R	σ	α
SNIPS-NLU	300	32	20	3	4	0.0001
CVA	200	200	100	8	1	0.01

Table 3: Hyperparameter settings.

Implementation Details. The hyperparameters used for experiments are shown in Table 3. We use three fold cross-validation to choose hyperpa-

---

[1] https://github.com/snipsco/nlu-benchmark/

Model	**SNIPS-NLU** (on 2 emerging intents)				**CVA** (on 20 emerging intents)			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
DeViSE (Frome et al., 2013)	0.7447	0.7448	0.7447	0.7446	0.7809	0.8060	0.7809	0.7617
CMT (Socher et al., 2013)	0.7396	**0.8266**	0.7396	0.7206	0.7721	0.7728	0.7721	0.7445
CDSSM (Chen et al., 2016a)	0.7588	0.7625	0.7588	0.7580	0.2140	0.4072	0.2140	0.1667
Zero-shot DNN (Kumar et al., 2017)	0.7165	0.7330	0.7165	0.7116	0.7903	0.8240	0.7903	0.7774
INTENTCAPSNET-ZSL w/o Self-attention	0.7587	0.7764	0.7588	0.7547	0.8103	0.8512	0.8103	0.8115
INTENTCAPSNET-ZSL w/o Bi-LSTM	0.7619	0.7631	0.7619	0.7616	0.8366	**0.8770**	0.8366	0.8403
INTENTCAPSNET-ZSL w/o Regularizer	0.7675	0.7676	0.7675	0.7675	0.8544	0.8730	0.8544	0.8553
INTENTCAPSNET-ZSL	**0.7752**	0.7762	**0.7752**	**0.7750**	**0.8628**	0.8751	**0.8629**	**0.8635**

Table 4: Zero-shot intention detection results using INTENTCAPSNET-ZSL on two datasets. All the metrics (Accuray, Precision, Recall and F1) are reported using the average value weighted by their support on per class.

rameters. The dimension of the prediction vector D_P is 10 for both datasets. $D_I = D_W$ because we use the averaged word embeddings contained in the intent label as the intent embedding. An additional input dropout layer with a dropout keep rate 0.8 is applied to the SNIPS-NLU dataset. In the loss function, the down-weighting coefficient λ is 0.5, margins m_k^+ and m_k^- are set to 0.9 and 0.1 for all the existing intents. The iteration number $iter$ used in the dynamic routing algorithm is 3. Adam optimizer (Kingma and Ba, 2014) is used to minimize the loss.

5 Results

Quantitative Evaluation. The intention detection results on two datasets are reported in Table 1, where the proposed capsule-based model INTENTCAPSNET performs consistently better than bag-of-word classifiers using TF-IDF, as well as various neural network models designed for text classification. These results demonstrate the novelty and effectiveness of the proposed capsule-based model INTENTCAPSNET in modeling text for intent detection.

Also, we report results on zero-shot intention detection task in Table 4, where our model INTENTCAPSNET-ZSL outperforms other baselines that adopt different zero-shot learning strategies. CMT has higher precision but low accuracy and recall on the SNIPS-NLU dataset. CDSSM fails on CVA dataset, probabily because the character-level model is suitable for English corpus but not for CVA, which is in Chinese.

Ablation Study. To study the contribution of different modules of INTENTCAPSNET-ZSL for zero-shot intent detection, we also report ablation test results in Table 4. "w/o Self-attention" is the model without self-attention: the last forward/backward hidden states of the bi-LSTM recurrent encoder are used; "w/o Bi-LSTM" uses the LSTM with only a forward pass; "w/o Regularizer" does not encourage discrepancies among different self-attention heads: it adopts $\alpha = 0$ in the loss function. Generally, from the lower part of Table 4 we can see that all modules contribute to the effectiveness of the model. On the SNIPS-NLU dataset, each of the three modules has a comparable contribution to the whole model (around 2-3% improvement in F1 score). While on the CVA dataset, the self-attention plays the most important role, which gives the model a 5.2% improvement in F1 score.

Discriminative Emerging Intent Representations. Besides quantitative evidences supporting the effectiveness of the INTENTCAPSNET-ZSL, we visualize activation vectors of emerging intents in Figure 3. Since the activation vectors of utterances with emerging intents are of high dimension and we are interested in their orientations which indicate their intent properties, t-SNE is applied on the normal vector of the activation vectors to reduce the dimension to 2. We color the utterances according to their ground-truth emerging intent labels.

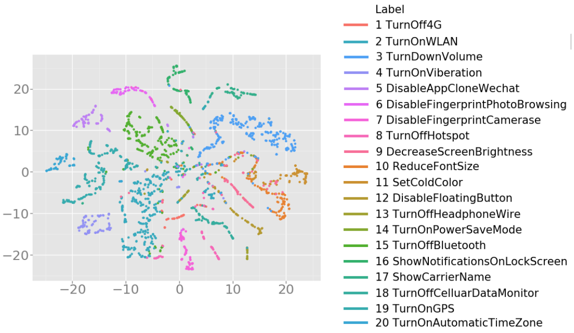

Figure 3: t-SNE visualization of normal activation vectors of utterances with 20 emerging intents in CVA.

As illustrated in Figure 3, INTENTCAPSNET-ZSL has the ability to learn discriminative intent representations for emerging intents in zero-shot

DetectionCaps, so that utterances with different intents naturally have different orientations. In the meanwhile, utterances of the same emerging intent but with nuances in expressions result in their proximity in the t-SNE space. However, we do observe less satisfied cases where the model mistake an emerging intent DecreaseScreenBrightness (No. 9) with ReduceFontSize (No. 10) and SetColdColor (No. 11). When we check activation vectors of intents in Figure 3 we also find that these three intents tend to have similar representations around the area (15, -5). We think it is due to their inherent similarity as these three intents all try to tune display configurations.

6 Interpretability

Capsule models try to bring more interpretability when compared with traditional deep neural networks. We provide case studies here toward the intepretability of the proposed model in 1) extracting meaningful semantic features and 2) transferring knowledge from existing intents to emerging intents.

Extracting Meaningful Semantic Features. To show that SemanticCaps have the ability to extract meaningful semantic features from the utterance, we study the self-attention matrix **A** within the SemanticCaps and visualize the attention scores of utterances on both existing and emerging intents.

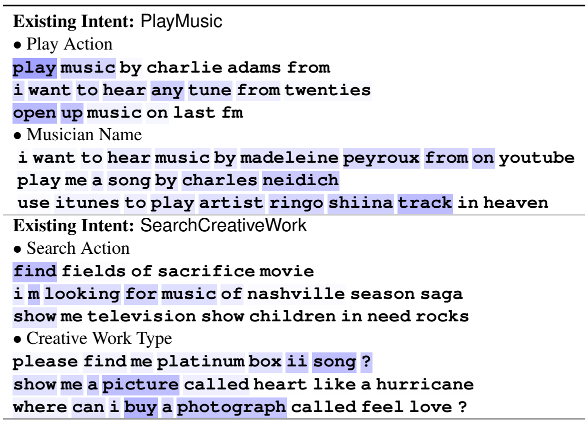

Existing Intent: PlayMusic
- Play Action

play music by charlie adams from
i want to hear any tune from twenties
open up music on last fm

- Musician Name

i want to hear music by madeleine peyroux from on youtube
play me a song by charles neidich
use itunes to play artist ringo shiina track in heaven

Existing Intent: SearchCreativeWork
- Search Action

find fields of sacrifice movie
i m looking for music of nashville season saga
show me television show children in need rocks

- Creative Work Type

please find me platinum box ii song ?
show me a picture called heart like a hurricane
where can i buy a photograph called feel love ?

Table 5: Attentions on utterances with existing intents on SNIPS-NLU.

From Table 5 we can see that each self-attention head almost always focuses on one unique semantic feature of the utterance. For example, in the intent of PlayMusic one self-attention head always focuses on the "play" action while another attention focuses on musician names. We also observe that the learned attention adopts well to diverse expressions. For example, the self-attention head in PlayMusic is attentive to various mentions of musician names when they are followed by words like `by`, `play` and `artist`, even when named entities are not tagged and given to the model. The self-attention head that extracts the "search" action in SearchCreativeWork is able to be attentive to various expressions such as `find`, `looking for` and `show`.

Extraction-behavior Transfer by SemanticCaps. More importantly, we observe appealing extraction behaviors of SemanticCaps on utterances of emerging intents as well, even if they are not trained to perform semantic extraction on utterances of emerging intents.

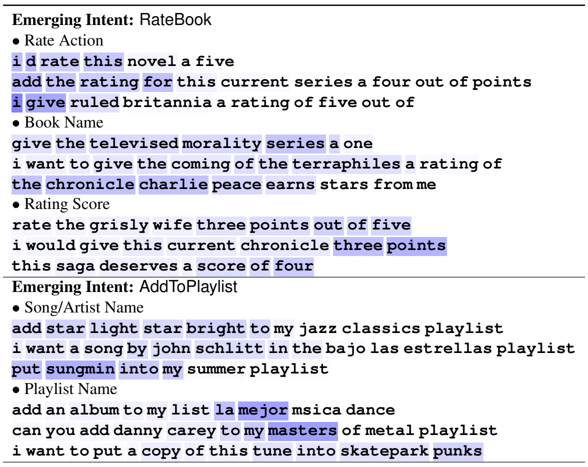

Emerging Intent: RateBook
- Rate Action

i d rate this novel a five
add the rating for this current series a four out of points
i give ruled britannia a rating of five out of

- Book Name

give the televised morality series a one
i want to give the coming of the terraphiles a rating of
the chronicle charlie peace earns stars from me

- Rating Score

rate the grisly wife three points out of five
i would give this current chronicle three points
this saga deserves a score of four

Emerging Intent: AddToPlaylist
- Song/Artist Name

add star light star bright to my jazz classics playlist
i want a song by john schlitt in the bajo las estrellas playlist
put sungmin into my summer playlist

- Playlist Name

add an album to my list la mejor msica dance
can you add danny carey to my masters of metal playlist
i want to put a copy of this tune into skatepark punks

Table 6: Attentions on utterances with emerging intents on SNIPS-NLU.

From Table 6 we observe that the same self-attention head that extracts "play" action in the existing intent PlayMusic is also attentive to words or phrases referring to the "rate" action in an emerging intent RateABook: like `rate`, `add the rating`, and `give`. Other self-attention heads are almost always focusing on other aspects of the utterances such as the book name or the actual rating score.

Such behavior not only shows that SemanticCaps have the capacity to learn an intent-independent semantic feature extractor, which extracts generalizable semantic features that either existing or emerging intent representations are built upon, but also indicates that SemanticCaps has the ability to transfer extraction behaviors among utterances of different intents.

Knowledge Transfer via Intent Similarity. Beside extracting semantic features and utilizing existing routing information, we use similarities between intent embeddings to help transfer vote vectors from INTENTCAPSNET to INTENTCAPSNET-ZSL. We study the similarity distribution of each emerging intents to all existing intents in Figure 4.

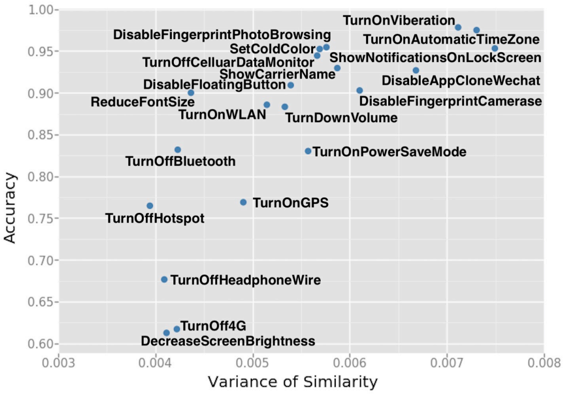

Figure 4: Accuracy vs. variance of the similarity distribution for 20 emerging intents in CVA dataset.

The y axis is the zero-shot detection accuracy on each emerging intent in the CVA dataset. The x axis measures var($\mathbf{q}_l$), the variance of the similarity distribution of each emerging intent l to all the existing intents. If an emerging intent has a high variance in the similarity distribution, it means that some existing intents have higher similarities with this emerging intent than others: the model is more certain about which existing intent to transfer the similarity knowledge from, based on intent label similarities. In this case, 13 out of 20 emerging intents with high variances where var($\mathbf{q}_l$) > 0.005 always have a decent performance (Accuracy$\geqslant$0.83). While a low variance does not necessarily always lead to less satisfied performances as some intents can rely on existing intents more evenly together, but with less confidence on each, for knowledge transfer.

7 Conclusions

In this paper, a capsule-based model, namely INTENTCAPSNET, is first introduced to harness the advantages of capsule models for text modeling in a hierarchical manner: semantic features are extracted from the utterances with self-attention, and aggregated via the dynamic routing-by-agreement mechanism to obtain utterance-level intent representations. We believe that the inductive biases subsumed in such capsule-based hierarchical learning schema have broader applicability on various text modeling tasks, besides its evidenced performance on the intent detection task we studied in this paper. The proposed INTENTCAPSNET-ZSL model further introduces zero-shot learning ability to the capsule model via various means of knowledge transfer from existing intents for discriminating emerging intents where no labeled utterances or excessive external resources are available. Experiments on two real-world datasets show the effectiveness and intepretability of the proposed models.

Acknowledgments

We thank the reviewers for their valuable comments. This work is supported in part by NSF through grants IIS-1526499, IIS-1763325, and CNS-1626432, and NSFC 61672313. Xiaohui Yan's work is funded by the National Natural Science Foundation of China (NSFC) under Grant No. 61502447.

References

Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. 2016. Synthesized classifiers for zero-shot learning. In *CVPR*, pages 5327–5336.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016a. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *ICASSP*, pages 6045–6049.

Yun-Nung Chen, Dilek Hakkani-Tür, Gökhan Tür, Jianfeng Gao, and Li Deng. 2016b. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *INTERSPEECH*, pages 3245–3249.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefevre. 2015a. Online adaptative zero-shot learning spoken language understanding using word-embedding. In *ICASSP*, pages 5321–5325.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015b. Zero-shot semantic parser for spoken language understanding. In *INTERSPEECH*, pages 1403–1407.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *NIPS*, pages 2121–2129.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming auto-encoders. In *ICANN*, pages 44–51.

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

Matthew B Hoy. 2018. Alexa, siri, cortana, and more: An introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88.

Jian Hu, Gang Wang, Fred Lochovsky, Jian-tao Sun, and Zheng Chen. 2009. Understanding user's query intent with wikipedia. In *WWW*, pages 471–480.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Anjishnu Kumar, Pavankumar Reddy Muddireddy, Markus Dreyer, and Björn Hoffmeister. 2017. Zero-shot learning across heterogeneous overlapping domains. In *INTERSPEECH*, volume 2017, pages 2914–2918.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2014. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *INTERSPEECH*, pages 685–689.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *NIPS*, pages 3859–3869.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *NIPS*, pages 935–943.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

IBM Watson Assistant. 2017. Defining intents. In `https://console.bluemix.net/docs/services/conversation/intents.html#defining-intents`.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *ASRU*, pages 78–83.

Majid Yazdani and James Henderson. 2015. A model of zero-shot learning of spoken language understanding. In *EMNLP*, pages 244–249.

Chenwei Zhang, Wei Fan, Nan Du, and Philip S Yu. 2016. Mining user intentions from medical queries: A neural network based heterogeneous jointly modeling approach. In *WWW*, pages 1373–1384.

Hierarchical Neural Networks for Sequential Sentence Classification in Medical Scientific Abstracts

Di Jin
MIT
`jindi15@mit.edu`

Peter Szolovits
MIT
`psz@mit.edu`

Abstract

Prevalent models based on artificial neural network (ANN) for sentence classification often classify sentences in isolation without considering the context in which sentences appear. This hampers the traditional sentence classification approaches to the problem of sequential sentence classification, where structured prediction is needed for better overall classification performance. In this work, we present a hierarchical sequential labeling network to make use of the contextual information within surrounding sentences to help classify the current sentence. Our model outperforms the state-of-the-art results by 2%-3% on two benchmarking datasets for sequential sentence classification in medical scientific abstracts.

1 Introduction

Since 1665, over 50 million scholarly research articles have been published (Jinha, 2010), with approximately 2.5 million new scientific papers coming out each year (Ware and Mabe, 2015). While this enormous corpus provides us with the ability to conclusively accept or reject hypotheses and yields insight into promising research directions, it is getting harder and harder to extract useful information from the literature in an efficient and timely manner due to its sheer amount. Therefore, an automatic and intelligent tool to help users locate the information of interest quickly and comprehensively is highly desired.

When searching for relevant literature for a certain field, investigators first check the abstracts of scientific papers to see whether they match the criterion of interest. This process can be expedited if the abstracts are structured; that is, if the rhetorical structural elements of scientific abstracts such as *purpose, methods, results, and conclusions* (American National Standards Institute, 1979) are explicitly stated. However, even today, a significant portion of scientific abstracts is still unstructured, which causes great difficulty in information retrieval. In this paper, we develop a machine-learning based approach to automatically categorize sentences in scientific abstracts into rhetorical sections so that the desired information can be efficiently retrieved.

In a scientific abstract, each sentence can be assigned to a rhetorical structural element sequentially. This rhetorical structure profiling process can be formulated as a sequential sentence classification task, as the element assignment of any single sentence is greatly associated with the assignments of the surrounding sentences. This is in contrast to the general sentence classification problem, where each sentence is classified individually and no contextual information can be used. Previous state-of-the-art methods relied on Conditional Random Fields (CRFs) to take into account the inter-dependence between subsequent labels, which improved joint sentence classification performance by considering the label sequence information. In this work, we add a bi-directional long short-term memory (bi-LSTM) layer over the representations of individual sentences so that it can encode the contextual content and semantics from preceding and succeeding sentences for better categorical inference of the current one.

In this work, we present a hierarchical neural network model for the sequential sentence classification task, which we call a hierarchical sequential labeling network (HSLN). Our model first uses a RNN or CNN layer to individually encode the sentence representation from the sequence of word embeddings, then uses another bi-LSTM layer to take as input the individual sentence representation and output the contextualized sentence representation, subsequently uses a single-hidden-layer feed-forward network to transform the sentence

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3100–3109
Brussels, Belgium, October 31 - November 4, 2018.

representation to the probability vector, and finally optimizes the predicted label sequence jointly via a CRF layer. We evaluate our model on two benchmarking datasets, PubMed RCT (Dernoncourt and Lee, 2017) and NICTA-PIBOSO (Kim et al., 2011), which were both generated from the PubMed database[1]. Our key contributions are summarized as follows:

1. Based on the previous best performing architecture for sequential sentence classification (Dernoncourt et al., 2016), we add one more layer to extract contextual information from surrounding sentences for more accurate prediction of the current one. Together with the CRF algorithm, this allows us to make use of not only the preceding labels' information but also the content and semantics of adjacent sentences to infer the label of the target sentence.

2. We remove the need for a character-based word embedding component without sacrificing performance. For individual sentence encoding, we propose the use of a CNN module as an alternative to RNN for small datasets, suffering less from over-fitting as evidenced by our experiments. Moreover, we incorporate attention-based pooling in both RNN and CNN models to further improve the performance.

3. We adopt dropout with expectation-linear regularization instead of the standard one to reduce the performance gap between training and test phases.

4. We obtain state-of-the-art results on two datasets for sequential sentence classification in medical abstracts, outperforming the previous best models by at least 2% in terms of F1 scores.

2 Related Work

Previous systems for sequential sentence classification concentrate on the rhetorical structure analysis of biomedical abstracts. They are mainly based on naive Bayes (Ruch et al., 2007), support vector machine (SVM) (McKnight and Srinivasan, 2003; Yamamoto and Takagi, 2005; Liu et al., 2013), Hidden Markov Model (HMM) (Lin et al., 2006), and CRF (Kim et al., 2011; Hassanzadeh et al., 2014; Hirohata et al., 2008; Chung, 2009). All these methods heavily rely on numerous carefully hand-engineered features such as lexical (bag-of-words (BOW)), semantic (hypernyms, synonyms), structural (part of speech (POS) tags, lemmas, orthographic shapes, headings), statistical (statistical distributions of token types) and sequential (sentence position, surrounding features, predicted labels) features.

In contrast, current emerging artificial neural network (ANN) based models have removed the need for manually selected features; instead, features are self-learned from the token and/or character embeddings. These deep learning models have revolutionized the natural language processing (NLP) field with state-of-the-art results achieved in various tasks, including the most relevant text classification task (Kim, 2014; Zhang et al., 2016; Conneau et al., 2017; Lai et al., 2015; Joulin et al., 2016; Ma et al., 2015). Most of these models are built upon deep CNNs or RNNs as well as combinations of them, where CNN is good at extracting local n-gram features while RNN is suitable for sequence modeling.

The above-mentioned works for short-text classification do not consider any context of sentence semantics in the models, making them underperform in the sequential sentence classification scenario, where surrounding sentences can play a big role in inferring the label of the current sentence. Recent works that apply deep neural networks to the sequential sentence classification problem include the system proposed by Lee et al. (Lee and Dernoncourt, 2016), where the preceding utterances were used to help classify the current utterance in a dialog into the corresponding dialogue act. Most recent work from Dernoncourt et al. (Dernoncourt et al., 2016) used a CRF layer to optimize the predicted label sequence, where the preceding labels have influence on determining the current label. This model outperformed the state-of-the-art results on two datasets PubMed RCT and NICTA-PIBOSO for sentence classification in medical abstracts.

3 Proposed Model

Notation We denote scalars in italic lowercase (e.g., k), vectors in bold italic lowercase (e.g., $\boldsymbol{s}$) and matrices in italic uppercase (e.g., W). Colon notations $x_{i:j}$ and $\boldsymbol{s}_{i:j}$ are used to denote the se-

---

[1] https://www.ncbi.nlm.nih.gov/pubmed/

quence of scalars $(x_i, x_{i+1}, ..., x_j)$ and vectors $(\boldsymbol{s}_i, \boldsymbol{s}_{i+1}, ..., \boldsymbol{s}_j)$.

Our model is composed of four components: the word embedding layer, the sentence encoding layer, the context enriching layer, and the label sequence optimization layer. In the following sections they will be discussed in detail.

3.1 Word Embedding Layer

Given a sentence $\boldsymbol{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_N \end{bmatrix}$ comprising N words, this layer maps each word to a real-valued vector as its lexical-semantic representation. Word representations are encoded by the column vector in the embedding matrix $W^{word} \in \mathbb{R}^{d^w \times |V|}$, where d^w is the dimension of the word vector and V is the vocabulary of the dataset. Each column $W_i^{word} \in \mathbb{R}^{d^w}$ is the word embedding vector for the i^{th} word in the vocabulary. The word embeddings W^{word} can be pre-trained on large unlabeled datasets using unsupervised algorithms such as word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and fastText (Bojanowski et al., 2016).

3.2 Sentence Encoding Layer

This layer takes as input the embedding vector of each token in a sentence from the word embedding layer and produces a vector **s** to encode this sentence. The sequence of embedding vectors is first processed by a bi-directional RNN (bi-RNN) or CNN layer, similar to the ones used in the text classification before (Kim, 2014; Lee and Dernoncourt, 2016; Liu et al., 2016). This layer outputs a sequence of hidden states $\boldsymbol{h}_{1:N}$ ($\boldsymbol{h} \in \mathbb{R}^{d^{hs}}$) for a sentence of N words with each hidden state corresponding to a word. To form the final representation vector $\boldsymbol{s}$ of this sentence, attention-based pooling is used, which can be described using the following equations:

$$A = \text{softmax}(U_s \tanh(W_s H + \boldsymbol{b}_s)), \quad (1)$$

$$S = AH^T, \quad (2)$$

where $H = \begin{bmatrix} h_1 & h_2 & \cdots & h_N \end{bmatrix} \in \mathbb{R}^{d^{hs} \times N}$, $W_s \in \mathbb{R}^{d^a \times d^{hs}}$ is the transformation matrix for soft alignment, $\boldsymbol{b}_s \in \mathbb{R}^{d^a}$ is the bias vector, $U_s \in \mathbb{R}^{r \times d^a}$ is the token level context matrix used to measure the relevance or importance of each token with respect to the whole sentence, softmax is performed along the second dimension of its input matrix, and $A \in \mathbb{R}^{r \times N}$ is the attention matrix.

Here each row of U_s is a context vector $\boldsymbol{u}_s \in \mathbb{R}^{d^a}$ and it is expected to reflect an aspect or component of the semantics of a sentence. To represent the overall semantics of the sentence, we use multiple context vectors to focus on different parts of this sentence.

Finally, the sentence encoding vector $\boldsymbol{s} \in \mathbb{R}^{rd^{hs}}$ is obtained by reshaping the matrix S into a vector.

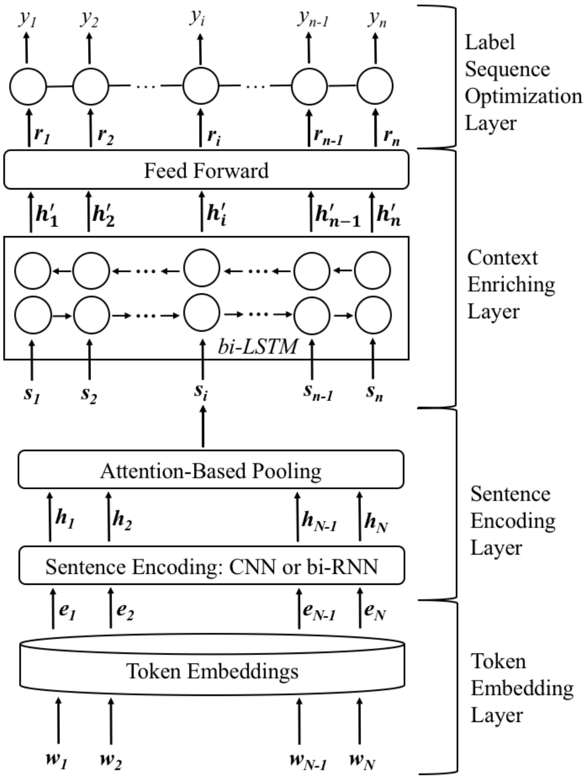

Figure 1: Model architecture. $\boldsymbol{w}$: original word; $\boldsymbol{e}$: word embedding vector; $\boldsymbol{h}$: sentence-level hidden state output by the bi-RNN or CNN layer; $\boldsymbol{s}$: sentence representation vector; $\boldsymbol{h'}$: abstract-level hidden state output by the bi-LSTM layer; $\boldsymbol{r}$: sentence label probability vector; y: predicted sentence label.

3.3 Context Enriching Layer

This layer takes as input the sequence of individual sentence encoding vectors in a given abstract of n sentences obtained from the last sentence encoding layer, with each vector corresponding to a sentence. It outputs a new sequence of contextualized sentence encoding vectors, which are enriched with the contextual information from surrounding sentences. Specifically, the sequence of individual sentence encoding vectors is input into a bi-LSTM layer, which produces a sequence of hidden state vectors $\boldsymbol{h'}_{1:n}$ ($\boldsymbol{h'} \in \mathbb{R}^{d^{hd}}$) with each corresponding to a sentence. Each of these vectors is subsequently input to a feed-forward neural network with only one hidden layer to get the corresponding probability vector $\boldsymbol{r} \in \mathbb{R}^{l}$, which represents the probability that this sentence belongs to each label, where l is the number of labels.

3.4 Label Sequence Optimization Layer

Within the abstract, the sequence of sentence categories implicitly follows some patterns. For example, the category *Results* is always followed by *Conclusion*, and the category *Methods* is certainly after the *Background*. Making use of such patterns can boost the classification performance via the CRF algorithm (Lample et al., 2016). Given the sequence of probability vectors $\boldsymbol{r}_{1:n}$ from the last context enriching layer for an abstract of n sentences, this layer outputs a sequence of labels $y_{1:n}$, where y_i represents the predicted label assigned to the i^{th} sentence.

In the CRF algorithm, in order to model dependencies between subsequent labels, we incorporate a matrix T that contains the transition probabilities between two subsequent labels; we define $T[i,j]$ as the probability that a token with label i is followed by a token with the label j. The score of a label sequence $y_{1:n}$ is defined as the sum of the probabilities of individual labels and the transition probabilities:

$$s(y_{1:n}) = \sum_{i=1}^{n} \boldsymbol{r}_i(y_i) + \sum_{i=2}^{n} T[y_{i-1}, y_i]. \qquad (3)$$

The score in the above equation can be transformed into the probability of a certain label sequence by taking a softmax operation over all possible label sequences:

$$p(y_{1:n}) = \frac{e^{s(y_{1:n})}}{\sum_{\hat{y}_{1:n} \in Y} e^{s(\hat{y}_{1:n})}}, \qquad (4)$$

where Y denotes the set of all possible label sequences. During the training phase, the objective is to maximize the probability of the gold label sequence. In the testing phase, given an input sequence, the corresponding sequence of predicted labels is chosen as the one that maximizes the score, computed via the Viterbi algorithm (Forney, 1973).

4 Experiments

4.1 Datasets

We evaluate our model on two sources of benchmarking datasets on medical scientific abstracts, where each sentence of the abstract is annotated with one label that is associated with the rhetorical structure. Table 1 summarizes the statistics of the two datasets.

NICTA-PIBOSO This dataset[2] was shared from the ALTA 2012 Shared Task (Amini et al., 2012), the goal of which is to build automatic sentence classifiers that can map the sentences from biomedical abstracts into a set of pre-defined categories for Evidence-Based Medicine (EBM).

PubMed RCT This new dataset was curated by (Dernoncourt and Lee, 2017)[3] and is currently the largest dataset for sequential sentence classification. It is based on the PubMed database of biomedical literature and each sentence of each abstract is labeled with its role in the abstract using one of the following classes: *background, objective, method, result, and conclusion*. Table 2 presents an example abstract comprising structured sentences with their annotated labels.

4.2 Training Settings

For both datasets, test performance is assessed on the training epoch with best validation performance and F1 scores (weighted average by support (the number of true instances for each label)) are reported as the results.

The token embeddings were pre-trained on a large corpus combining Wikipedia, PubMed, and PMC texts (Moen and Ananiadou, 2013) using the word2vec tool[4] (denoted as “Word2vec-wiki+P.M.”). They are fixed during the training phase to avoid over-fitting. We also tried other types of word embeddings, such as the word2vec embeddings pre-trained on the Google News dataset[5] (denoted as “Word2vec-News”), word2vec embeddings pre-trained on the Wikipedia corpus[6] (denoted as “Word2vec-wiki”), GloVe embeddings pre-trained on the cor-

---

[2] This dataset can be found online at https://www.kaggle.com/c/alta-nicta-challenge2

[3] This dataset can be downloaded from https://github.com/Franck-Dernoncourt/pubmed-rct

[4] The word vectors can be downloaded at http://bio.nlplab.org/

[5] https://code.google.com/archive/p/word2vec/

[6] https://github.com/jind11/word2vec-on-wikipedia

Dataset	$\|C\|$	$\|V\|$	Train	Validation	Test
NICTA-PIBOSO	6	17k	720 (7.7k)	80 (0.9k)	200 (2.2k)
PubMed 20k	5	68k	15k (180k)	2.5k (30k)	2.5k (30k)
PubMed 200k	5	331k	190k (2.2M)	2.5k (29k)	2.5k (29k)

Table 1: Datasets statistics. $|C|$ denotes the number of labels, $|V|$ represents the vocabulary size. For the train, validation, and test sets, we indicate the number of abstracts followed by the number of sentences in parentheses.

Category	**Sentences**
BACKGROUND	Emotional eating is associated with overeating and the development of obesity. [...]
OBJECTIVES	The aim of this study was to test if attention bias for food moderates the effect of self-reported emotional eating during sad mood (vs neutral mood) on actual food intake. [...]
METHODS	Participants (N = 85) were randomly assigned to one of the two experimental mood induction conditions (sad/neutral). [...]
RESULTS	[...] Yet, attention maintenance on food cues was significantly related to increased intake specifically in the neutral condition, but not in the sad mood condition.
CONCLUSIONS	The current findings show that self-reported emotional eating (based on the DEBQ) might not validly predict who overeats when sad, at least not in a laboratory setting with healthy women. [...]

Table 2: A typical abstract example with structured sentences and their corresponding annotated labels. The PMID of this abstract is 24854809.

pus of Wikipedia 2014 + Gigaword 5[7] (denoted as "Glove-wiki"), fastText embeddings pre-trained on Wikipedia[8] (denoted as "FastText-wiki"), and fastText embeddings initialized with the standard GloVe Common Crawl embeddings and then fine-tuned on PubMed abstracts plus MIMIC-III notes (denoted as "FastText-P.M.+MIMIC"). The comparison results are summarized in the next section.

The model is trained using the Adam optimization method (Kingma and Ba, 2014). The learning rate is initially set as 0.003 and decayed by 0.9 after each epoch. For regularization, dropout (Srivastava et al., 2014) is applied to each layer. For the version of dropout used in practice (e.g., the dropout function implemented in the TensorFlow and Pytorch libraries), the model ensemble generated by dropout in the training phase is approximated by a single model with scaled weights in the inference phase, resulting in a gap between training and inference. To reduce this gap, we adopted the dropout with expectation-linear regularization introduced by Ma et al. (2016) to explicitly control the inference gap and thus improve the generalization performance.

Hyperparameters were optimized via grid search based on the validation set and the best configuration is shown in Table 3. The window sizes of the CNN encoder in the sentence encoding layer are 2, 3, 4 and 5. The RNN encoder in the sentence encoding layer is set as LSTM for the PubMed datasets and gated recurrent unit (GRU) for the NICTA-PIBOSO dataset. Code for this work is available online[9].

5 Results and Discussion

Table 4 compares our model against the best performing models in the literature (Dernoncourt et al., 2016; Liu et al., 2013). There are two variants of our model in terms of different implementations of the sentence encoding layer: the model that uses bi-RNN to encode the sentence is called HSLN-RNN; while the model that uses the CNN module is named HSLN-CNN. We have evaluated both model variants on all datasets. And as evidenced by Table 4, our best model can improve the F1 scores by 2%-3% in absolute number compared with the previous best published results for all

---

[7] http://nlp.stanford.edu/data/glove.6B.zip

[8] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

[9] https://github.com/jind11/HSLN-Joint-Sentence-Classification

Parameter	PubMed		NICTA	
	RNN	CNN	RNN	CNN
d^{hs}	200	-	200	-
d^{hd}	200	200	200	300
d^{a}	200	100	250	75
d^{c}	-	200	-	150
r	15	1	5	4
β	0.01	0.001	0.01	0.01
dr	0.5	0.5	0.6	0.6

Table 3: Hyperparameter settings. d^{hs}: hidden size of the sentence-level RNN layer (single direction); d^{hd}: hidden size of the abstract-level bi-LSTM layer (single direction); d^{a}: dimension of the context vector $\boldsymbol{u}_s$; r: number of context vectors; β: coefficient of the dropout regularization added to the total loss; dr: dropout.

datasets. For the PubMed 20k and 200k datasets, our HSLN-RNN model achieves better results; however, for the NICTA dataset, the HSLN-CNN model performs better. This makes sense because the CNN sentence encoder has fewer parameters to be optimized, thus the HSLN-CNN model is less likely to over-fit in a smaller dataset such as NICTA. With sufficient data, however, the increased capacity of the HSLN-RNN model offers performance benefits. To be noted, this performance gap between RNN and CNN sentence encoder gets larger as the dataset size increases from 20k to 200k for the PubMed dataset.

Model	PubMed		NICTA
	20k	200k	
Best Published			
Marco Lui (Lui, 2012)	-	-	82.0
bi-ANN (Dernoncourt et al., 2016)	90.0	91.6	82.7
Our Models			
HSLN-CNN	92.2	92.8	**84.7**
HSLN-RNN	**92.6**	**93.9**	84.3

Table 4: Comparison of F1 scores (weighted average by support (the number of true instances for each label)) between our model and the best published methods. The presented results of our model are evaluated on the test set of the run with the highest F1 score on the validation set.

Table 5 presents the ablation analysis of our model (on the PubMed 20k dataset), where we remove one component at a time and quantify the performance drop (reported on F1 scores). As can be seen from Table 5, our HSLN-CNN model uniformly suffers a little more from the component removal than the HSLN-RNN model, indicating that the HSLN-RNN model is more robust. When the context enriching layer is removed, both models experience the most significant performance drop and can only be on par with the previous state-of-the-art results, strongly demonstrating that this proposed component is the key to the performance improvement of our model. Furthermore, even without the label sequence optimization layer, our model still significantly outperforms the best published methods that are empowered by this layer, indicating that the context enriching layer we propose can help optimize the label sequence by considering the context information from the surrounding sentences. Last but not the least, the dropout regularization and attention-based pooling components we add to our system can help further improve the model in a limited extent.

Model	HSLN-RNN	HSLN-CNN
Full Model	**92.6**	**92.2**
– context	90.0	89.0
– seq. opt.	92.3	91.8
– dropout reg.	92.4	91.9
– attention	92.4	91.7

Table 5: Ablation analysis. F1 scores are reported. "– context" is our model without the context enriching layer. "– seq. opt." is our model without the label sequence optimization layer. "– dropout reg.' is our model using the standard dropout strategy without the expectation-linearization regularization. "– attention" refers to the model without attention-based pooling, i.e., in the sentence encoding layer, the final hidden state is used for the HSLN-RNN model while max-pooling is used for the HSLN-CNN model.

Table 6 and 7 detail the results of classification for each label in terms of performance scores (precision, recall and F1) and confusion matrix, respectively (for our HSLN-RNN model trained on the PubMed 20k dataset). These show that the classifier is very good at predicting the labels *Methods*, *Results* and *Conclusions*, whereas the greatest difficulty the classifier has is in distinguishing *Background* sections from *Objectives* sections. One fifth of *Background* sentences are incorrectly classified as *Objectives*, while around one forth of *Objectives* sentences are wrongly assigned to the label of *Background*. We conjecture this difficulty mainly comes from the fact that the difference between *Background* and *Ob-*

jectives sentences in terms of writing style is less obvious compared with the other sections of the abstract. Moreover, our model has some difficulty in telling *Methods* sentences apart from *Results* sentences.

Label	P	R	F1	Support
Background	78.5	80.0	79.2	3077
Objectives	74.2	69.9	72.0	2333
Methods	95.0	97.7	96.3	9884
Results	96.8	95.3	96.0	9713
Conclusions	97.6	96.5	97.1	4571
Total	92.6	92.7	92.6	29578

Table 6: Results (presented in percentage) in terms of precision (P), recall (R) and F-measure (F1) on the test set for each label obtained by our HSLN-RNN model on the PubMed 20k dataset.

	B	**C**	**M**	**O**	**R**
B	2460	4	69	537	7
C	4	4413	11	1	142
M	37	11	9657	27	152
O	632	0	68	1630	3
R	2	95	362	1	9253

Table 7: Confusion matrix obtained by our model on the PubMed 20k dataset. Rows correspond to predicted labels, and columns correspond to true labels. B represents background, O represents objectives, M represents methods, R represents results, and C represents conclusions.

Table 8 presents a few examples of prediction errors that are produced by our HSLN-RNN model trained on the PubMed 20k dataset. This error analysis suggests that one of the biggest model error sources could be from the debatable gold standard labels of the dataset. For example, the sentence "Depressive disorders are one of the leading components of the global burden of disease with a prevalence of up to 14% in the general population." is indeed introducing the background of the problem (depressive disorders) on which this article is going to focus; however, the gold label classifies it into the *Objective* category. For another instance, the sentence "A post hoc analysis was conducted with the use of data from the evaluation study of congestive heart failure and pulmonary artery catheterization effectiveness (escape)." belongs to the *Result* label according to the gold standard, but it makes more sense that it should be classified as a *Method* label.

Figure 2 presents an example of the transition matrix after the HSLN-RNN model has been trained on the PubMed 20k dataset, which encodes the transition probability between two subsequent labels. It effectively reflects what label is the most likely one that follows the current one. For example, by comparing the transition scores in the *Result* row in Figure 2, we can conclude that a sentence pertaining to the *Result* is typically followed by a sentence pertaining to the *Conclusion* and is unlikely to be followed by a sentence in the *Background* category (transition scores of 2.48 vs -5.46), which makes sense. From this transition matrix, we can figure out the most probable label sequence: $Background \rightarrow Objective \rightarrow Method \rightarrow Result \rightarrow Conclusion$, which is also consistent with our expectations.

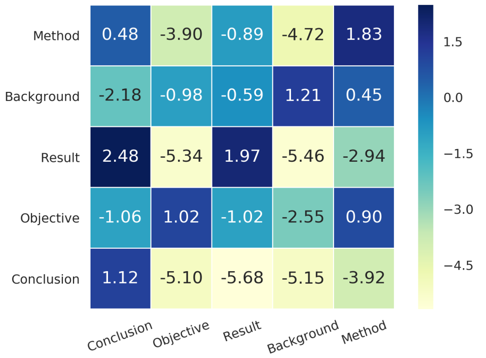

Figure 2: Transition matrix of label sequence after the HSLN-RNN model has been trained on the PubMed 20k dataset. The rows represent the label of the previous sentence, while the columns represent the label of the current sentence.

In order to test the importance of pretrained word embeddings, we performed experiments with different sets of publicly published word embeddings, as well as our locally curated word embeddings, to initialize our model. Table 9 gives the performance of six different word embeddings for our HSLN-RNN model trained on the PubMed 20k dataset. According to Table 9, the training methods that create the word embeddings do not have a strong influence on model performance, but the corpus they are trained on does. The combination of Wikipedia and PubMed abstracts as the corpus for unsupervised word embedding training yields the best result, and the individual use of either the Wikipedia corpus or the PubMed abstracts performs much worse. Although the dataset we

Sentence	**Predicted**	**Gold**
Depressive disorders are one of the leading components of the global burden of disease with a prevalence of up to 14% in the general population. [25829103]	Background	Objective
This study assessed whether diets with different fat quality and supplementation with coenzyme Q10 (CoQ) affect the metabolomic profile in urine. [24986061]	Objective	Background
A post hoc analysis was conducted with the use of data from the evaluation study of congestive heart failure. [24845963]	Method	Result
Hence, 47 secondary schools from all 12 districts of the city [...] are participating in the study. [25150368]	Result	Method
This study investigated whether oxytocin can affect attentional bias in social anxiety. [25552432]	Objective	Method
We hypothesize that BMC+Phone and BMC+Home will produce greater reductions in BMI percentiles than BMC alone. [24456698]	Conclusion	Method

Table 8: Examples of prediction errors of our HSLN-RNN model trained on the PubMed 20k dataset. Each sentence is followed by the PMID of the abstract that this sentence belongs to, which is enclosed in middle brackets. The "Predicted" column indicates the label predicted by our model for a given sentence. The "Gold" column indicates the gold label of the sentence.

are using for evaluation is also from PubMed abstracts, using only the PubMed abstracts together with MIMIC notes without the Wikipedia corpus does not guarantee better result (see the "FastText-P.M.+MIMIC" embeddings in Table 9), which may be because the corpus size of PubMed abstracts plus MIMIC notes (about 12.8 million abstracts and 1 million notes) is not large enough for good embedding training compared with the corpus consisting of at least billion tokens such as the Wikipedia.

Embedding	Dimension	P.M. 20k
Glove-wiki	200	92.0
FastText-wiki	300	92.2
FastText-P.M.+MIMIC	300	92.0
Word2vec-News	300	92.2
Word2vec-wiki	200	92.1
Word2vec-wiki+P.M.	200	**92.6**

Table 9: Comparison of performance with different choices of word embeddings for our HSLN-RNN model trained on the PubMed 20k dataset (reported on F1-scores on the test set). "P.M." means PubMed.

6 Conclusion

In this work, we have presented an ANN based hierarchical sequential labeling network to classify sentences that appear sequentially in text. We demonstrate that incorporating the contextual information from surrounding sentences to help classify the current one by using an LSTM layer to sequentially process the encoded sentence representations can improve the overall quality of predictions. Our model outperforms the state-of-the-art results by 2%-3% on two datasets for sequential sentence classification in medical abstracts. We expect that our proposed model can be generalized to any problem that is related to sequential sentence classification, such as the paragraph-level sequential sentence categorization in full-text articles for better text mining and document retrieval (Westergaard et al., 2018).

7 Future Work

Although the whole PubMed database contains over 2 million abstracts with part of them accompanied by full-text articles, only a small fraction of them are structured and contain the label information utilized in this work. We plan to make use of the rest unannotated abstracts or full texts to pre-train our model and then fine tune it to the target annotated datasets inspired by the work from (Howard and Ruder, 2018) so that the performance can be further boosted.

Acknowledgments

This work was supported by funding grant U54-HG007963 from National Human Genome Research Institute (NHGRI).

References

American National Standards Institute. 1979. American national standard for writing abstracts (ansi z39. 14-1979).

Iman Amini, David Martinez, Diego Molla, et al. 2012. Overview of the alta 2012 shared task.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Grace Y Chung. 2009. Sentence retrieval for abstracts of randomized controlled trials. *BMC medical informatics and decision making*, 9(1):10.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116.

Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *arXiv preprint arXiv:1710.06071*.

Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. 2016. Neural networks for joint sentence classification in medical paper abstracts. *arXiv preprint arXiv:1612.05251*.

G David Forney. 1973. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Hamed Hassanzadeh, Tudor Groza, and Jane Hunter. 2014. Identifying scientific artefacts in biomedical literature: The evidence based medicine use case. *Journal of biomedical informatics*, 49:159–170.

Kenji Hirohata, Naoaki Okazaki, Sophia Ananiadou, and Mitsuru Ishizuka. 2008. Identifying sections in scientific abstracts using conditional random fields. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Arif E Jinha. 2010. Article 50 million: an estimate of the number of scholarly articles in existence. *Learned Publishing*, 23(3):258–263.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Su Nam Kim, David Martinez, Lawrence Cavedon, and Lars Yencken. 2011. Automatic classification of sentences to support evidence based medicine. In *BMC bioinformatics*, volume 12, page S5. BioMed Central.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *HLT-NAACL*.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827*.

Jimmy Lin, Damianos Karakos, Dina Demner-Fushman, and Sanjeev Khudanpur. 2006. Generative content models for structural analysis of medical abstracts. In *Proceedings of the Workshop on Linking Natural Language Processing and Biology: Towards Deeper Biological Literature Analysis*, pages 65–72. Association for Computational Linguistics.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Yuanchao Liu, Feng Wu, Ming Liu, and Bingquan Liu. 2013. Abstract sentence classification for scientific papers based on transductive svm. *Computer and Information Science*, 6(4):125.

Marco Lui. 2012. Feature stacking for sentence classification in evidence-based medicine. In *Proceedings of the Australasian Language Technology Association Workshop 2012*, pages 134–138.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based convolutional neural networks for sentence embedding. *arXiv preprint arXiv:1507.01839*.

Xuezhe Ma, Yingkai Gao, Zhiting Hu, Yaoliang Yu, Yuntian Deng, and Eduard Hovy. 2016. Dropout with expectation-linear regularization. *arXiv preprint arXiv:1609.08017*.

Larry McKnight and Padmini Srinivasan. 2003. Categorization of sentence types in medical abstracts. In *AMIA Annual Symposium Proceedings*, volume 2003, page 440. American Medical Informatics Association.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

SPFGH Moen and Tapio Salakoski2 Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing. In *Proceedings of the 5th International Symposium on Languages in Biology and Medicine, Tokyo, Japan*, pages 39–43.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Patrick Ruch, Celia Boyer, Christine Chichester, Imad Tbahriti, Antoine Geissbühler, Paul Fabry, Julien Gobeill, Violaine Pillet, Dietrich Rebholz-Schuhmann, Christian Lovis, et al. 2007. Using argumentation to extract key sentences from biomedical abstracts. *International journal of medical informatics*, 76(2-3):195–200.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Mark Ware and Michael Mabe. 2015. The STM report: An overview of scientific and scholarly journal publishing.

David Westergaard, Hans-Henrik Stærfeldt, Christian Tønsberg, Lars Juhl Jensen, and Søren Brunak. 2018. A comprehensive and quantitative comparison of text-mining in 15 million full-text articles versus their corresponding abstracts. *PLoS computational biology*, 14(2):e1005962.

Yasunori Yamamoto and Toshihisa Takagi. 2005. A sentence classification system for multi biomedical literature summarization. In *Data Engineering Workshops, 2005. 21st International Conference on*, pages 1163–1163. IEEE.

Rui Zhang, Honglak Lee, and Dragomir Radev. 2016. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361*.

Investigating Capsule Networks with Dynamic Routing for Text Classification

Wei Zhao[1,2], **Jianbo Ye**[3], **Min Yang**[1]*, **Zeyang Lei**[4], **Soufei Zhang**[5], **Zhou Zhao**[6]
[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[2] Tencent [3] Pennsylvania State University
[4] Graduate School at Shenzhen, Tsinghua University
[5] Nanjing University of Posts and Telecommunications [6] Zhejiang University

Abstract

In this study, we explore capsule networks with dynamic routing for text classification. We propose three strategies to stabilize the dynamic routing process to alleviate the disturbance of some noise capsules which may contain "background" information or have not been successfully trained. A series of experiments are conducted with capsule networks on six text classification benchmarks. Capsule networks achieve competitive results over the compared baseline methods on 4 out of 6 datasets, which shows the effectiveness of capsule networks for text classification. We additionally show that capsule networks exhibit significant improvement when transfer single-label to multi-label text classification over the competitors. To the best of our knowledge, this is the first work that capsule networks have been empirically investigated for text modeling[1].

1 Introduction

Modeling articles or sentences computationally is a fundamental topic in natural language processing. It could be as simple as a keyword/phrase matching problem, but it could also be a nontrivial problem if compositions, hierarchies, and structures of texts are considered. For example, a news article which mentions a single phrase "US election" may be categorized into the political news with high probability. But it could be very difficult for a computer to predict which presidential candidate is favored by its author, or whether the author's view in the article is more liberal or more conservative.

Earlier efforts in modeling texts have achieved limited success on text categorization using a simple bag-of-words classifier (Joachims, 1998; McCallum et al., 1998), implying understanding the meaning of the individual word or n-gram is a necessary step towards more sophisticated models. It is therefore not a surprise that distributed representations of words, a.k.a. word embeddings, have received great attention from NLP community addressing the question "what" to be modeled at the basic level (Mikolov et al., 2013; Pennington et al., 2014). In order to model higher level concepts and facts in texts, an NLP researcher has to think cautiously the so-called "what" question: *what is actually modeled beyond word meanings*. A common approach to the question is to treat the texts as sequences and focus on their spatial patterns, whose representatives include convolutional neural networks (CNNs) (Kim, 2014; Zhang et al., 2015; Conneau et al., 2017) and long short-term memory networks (LSTMs) (Tai et al., 2015; Mousa and Schuller, 2017). Another common approach is to completely ignore the order of words but focus on their compositions as a collection, whose representatives include probabilistic topic modeling (Blei et al., 2003; Mcauliffe and Blei, 2008) and Earth Mover's Distance based modeling (Kusner et al., 2015; Ye et al., 2017).

Those two approaches, albeit quite different from the computational perspective, actually follow a common measure to be diagnosed regarding their answers to the "what" question. In neural network approaches, spatial patterns aggregated at lower levels contribute to representing higher level concepts. Here, they form a recursive process to articulate what to be modeled. For example, CNN builds convolutional feature detectors to extract local patterns from a window of vector sequences

---

* Corresponding author (min.yang@siat.ac.cn)

[1] Codes are publicly available at: `https://github.com/andyweizhao/capsule_text_classification`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3110–3119
Brussels, Belgium, October 31 - November 4, 2018.

and uses max-pooling to select the most prominent ones. It then hierarchically builds such pattern extraction pipelines at multiple levels. Being a spatially sensitive model, CNN pays a price for the inefficiency of replicating feature detectors on a grid. As argued in (Sabour et al., 2017), one has to choose between replicating detectors whose size grows exponentially with the number of dimensions, or increasing the volume of the labeled training set in a similar exponential way. On the other hand, methods that are spatially insensitive are *perfectly* efficient at the inference time regardless of any order of words or local patterns. However, they are unavoidably more restricted to encode rich structures presented in a sequence. Improving the efficiency to encode spatial patterns while keeping the flexibility of their representation capability is thus a central issue.

A recent method called capsule network introduced by Sabour et al. (2017) possesses this attractive potential to address the aforementioned issue. They introduce an iterative routing process to decide the credit attribution between nodes from lower and higher layers. A metaphor (also as an argument) they made is that human visual system intelligently assigns parts to wholes at the inference time without hard-coding patterns to be perspective relevant. As an outcome, their model could encode the intrinsic spatial relationship between a part and a whole constituting viewpoint invariant knowledge that automatically generalizes to novel viewpoints. In our work, we follow a similar spirit to use this technique in modeling texts. Three strategies are proposed to stabilize the dynamic routing process to alleviate the disturbance of some noise capsules which may contain "background" information such as stop words and the words that are unrelated to specific categories. We conduct a series of experiments with capsule networks on top of the pre-trained word vectors for six text classification benchmarks. More importantly, we show that capsule networks achieves significant improvement when transferring single-label to multi-label text classifications over the compared baseline methods.

2 Our Methodology

Our capsule network, depicted in Figure 1, is a variant of the capsule networks proposed in Sabour et al. (2017). It consists of four layers: n-gram convolutional layer, primary capsule layer, convolutional capsule layer, and fully connected capsule layer. In addition, we explore two capsule frameworks to integrate these four components in different ways. In the rest of this section, we elaborate the key components in detail.

2.1 N-gram Convolutional Layer

This layer is a standard convolutional layer which extracts n-gram features at different positions of a sentence through various convolutional filters.

Suppose $\mathbf{x} \in \mathbb{R}^{L \times V}$ denotes the input sentence representation where L is the length of the sentence and V is the embedding size of words. Let $\mathbf{x}_i \in \mathbb{R}^V$ be the V-dimensional word vector corresponding to the i-th word in the sentence. Let $W^a \in \mathbb{R}^{K_1 \times V}$ be the filter for the convolution operation, where K_1 is the N-gram size while sliding over a sentence for the purpose of detecting features at different positions. A filter W^a convolves with the word-window $\mathbf{x}_{i:i+K_1-1}$ at each possible position (with stride of 1) to produce a column feature map $\mathbf{m}^a \in \mathbb{R}^{L-K_1+1}$, each element $m_i^a \in \mathbb{R}$ of the feature map is produced by

$$m_i^a = f(\mathbf{x}_{i:i+K_1-1} \circ W^a + \mathbf{b}_0) \qquad (1)$$

where $\circ$ is element-wise multiplication, $\mathbf{b}_0$ is a bias term, and f is a nonlinear activate function (i.e., ReLU). We have described the process by which one feature is extracted from one filter. Hence, for $a = 1, \ldots, B$, totally B filters with the same N-gram size, one can generate B feature maps which can be rearranged as

$$\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_{\mathrm{B}}] \in \mathbb{R}^{(L-K_1+1) \times B} \qquad (2)$$

2.2 Primary Capsule Layer

This is the first capsule layer in which the capsules replace the scalar-output feature detectors of CNNs with vector-output capsules to preserve the instantiated parameters such as the local order of words and semantic representations of words.

Suppose $p_i \in \mathbb{R}^d$ denotes the instantiated parameters of a capsule, where d is the dimension of the capsule. Let $W^b \in \mathbb{R}^{B \times d}$ be the filter shared in different sliding windows. For each matrix multiplication, we have a window sliding over each N-gram vector denoted as $\mathbf{M}_i \in \mathbb{R}^B$, then the corresponding N-gram phrases in the form of capsule are produced with $p_i = (W^b)^T \mathbf{M}_i$.

The filter W^b multiplies each N-gram vector in $\{\mathbf{M_i}\}_{i=1}^{L-K_1+1}$ with stride of 1 to produce a

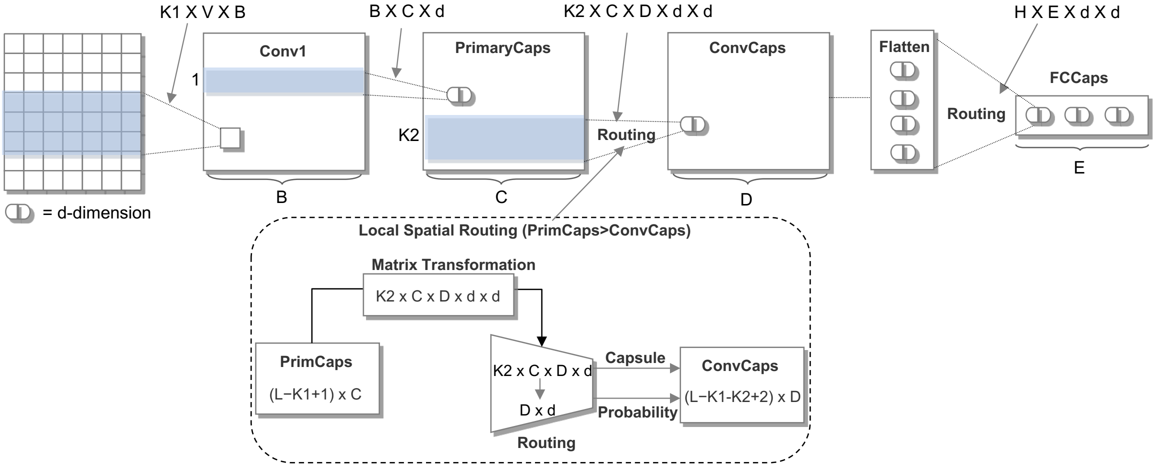

Figure 1: The Architecture of Capsule network for text classification. The processes of dynamic routing between consecutive layers are shown in the bottom.

column-list of capsules $\mathbf{p} \in \mathbb{R}^{(L-K_1+1)\times d}$, each capsule $p_i \in \mathbb{R}^d$ in the column-list is computed as

$$p_i = g(W^b \mathbf{M}_i + \mathbf{b}_1) \qquad (3)$$

where g is nonlinear squash function through the entire vector, $\mathbf{b}_1$ is the capsule bias term. For all C filters, the generated capsule feature maps can be rearranged as

$$\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_C] \in \mathbb{R}^{(L-K_1+1)\times C\times d}, \qquad (4)$$

where totally $(L - K_1 + 1) \times C$ d-dimensional vectors are collected as capsules in $\mathbf{P}$.

2.2.1 Child-Parent Relationships

As argued in (Sabour et al., 2017), capsule network tries to address the representational limitation and exponential inefficiencies of convolutions with transformation matrices. It allows the networks to automatically learn child-parent (or part-whole) relationships constituting viewpoint invariant knowledge that automatically generalizes to novel viewpoints.

In this paper, we explore two different types of transformation matrices to generate prediction vector (vote) $\hat{u}_{j|i} \in \mathbb{R}^d$ from its child capsule i to the parent capsule j. The first one shares weights $W^{t_1} \in \mathbb{R}^{N\times d\times d}$ across child capsules in the layer below, where N is the number of parent capsules in the layer above. Formally, each corresponding vote can be computed by:

$$\hat{u}_{j|i} = W_j^{t_1} u_i + \hat{b}_{j|i} \in \mathbb{R}^d \qquad (5)$$

where u_i is a child-capsule in the layer below and $\hat{b}_{j|i}$ is the capsule bias term.

In the second design, we replace the shared weight matrix $W_j^{t_1}$ with non-shared weight matrix $W_{i,j}^{t_2}$, where the weight matrices $W^{t_2} \in \mathbb{R}^{H\times N\times d\times d}$ and H is the number of child capsules in the layer below.

2.3 Dynamic Routing

The basic idea of dynamic routing is to construct a non-linear map in an iterative manner ensuring that the output of each capsule gets sent to an appropriate parent in the subsequent layer:

$$\left\{\hat{u}_{j|i} \in \mathbb{R}^d\right\}_{i=1,...,H,j=1...,N} \mapsto \left\{v_j \in \mathbb{R}^d\right\}_{j=1}^{N}.$$

For each potential parent, the capsule network can increase or decrease the connection strength by dynamic routing, which is more effective than the primitive routing strategies such as max-pooling in CNN that essentially detects whether a feature is present in any position of the text, but loses spatial information about the feature. We explore three strategies to boost the accuracy of routing process by alleviating the disturbance of some noisy capsules:

Orphan Category Inspired by Sabour et al. (2017), an additional "orphan" category is added to the network, which can capture the "background" information of the text such as stop words and the words that are unrelated to specific categories, helping the capsule network model the child-parent relationship more efficiently. Adding "orphan" category in the text is more effective than in image since there is no single consistent "background" object in images, while the stop words are consistent in texts such as predicate “s”, “am” and pronouns “his”, “she”.

Leaky-Softmax We explore Leaky-Softmax Sabour et al. (2017) in the place of standard softmax while updating connection strength between the children capsules and their parents. Despite the orphan category in the last capsule layer, we also need a light-weight method between two consecutive layers to route the noise child capsules to extra dimension without any additional parameters and computation consuming.

Coefficients Amendment We also attempt to use the probability of existence of child capsules in the layer below to iteratively amend the connection strength as Eq.6.

Algorithm 1: Dynamic Routing Algorithm

```
procedure ROUTING($\hat{u}_{j|i}$, $\hat{a}_{j|i}$, $r$, $l$)
Initialize the logits of coupling coefficients
  $b_{j|i} = 0$
for $r$ iterations do
    for all capsule $i$ in layer $l$ and capsule $j$ in
      layer $l+1$:
      $c_{j|i} = \hat{a}_{j|i} \cdot \text{leaky-softmax}(b_{j|i})$
    for all capsule $j$ in layer $l+1$:
      $v_j = g(\sum_i c_{j|i}\hat{u}_{j|i}), \quad a_j = |v_j|$
    for all capsule $i$ in layer $l$ and capsule $j$ in
      layer $l+1$: $b_{j|i} = b_{j|i} + \hat{u}_{j|i} \cdot v_j$
return $v_j$,$a_j$
```

Given each prediction vector $\hat{u}_{j|i}$ and its probability of existence $\hat{a}_{j|i}$, where $\hat{a}_{j|i} = \hat{a}_i$, each iterative coupling coefficient of connection strength $c_{j|i}$ is updated by

$$c_{j|i} = \hat{a}_{j|i} \cdot \text{leaky-softmax}(b_{j|i}) \tag{6}$$

where $b_{j|i}$ is the logits of coupling coefficients. Each parent capsule v_j in the layer above is a weighted sum over all prediction vectors $\hat{u}_{j|i}$:

$$v_j = g(\sum_i c_{j|i}\hat{u}_{j|i}), \quad a_j = |v_j| \tag{7}$$

where a_j is the probabilities of parent capsules, g is nonlinear squash function Sabour et al. (2017) through the entire vector. Once all of the parent capsules are produced, each coupling coefficient $b_{j|i}$ is updated by:

$$b_{j|i} = b_{j|i} + \hat{u}_{j|i} \cdot v_j \tag{8}$$

For simplicity of notation, the parent capsules and their probabilities in the layer above are denoted as

$$v, a = \text{Routing}(\hat{\text{u}}) \tag{9}$$

where $\hat{u}$ denotes all of the child capsules in the layer below, v denotes all of the parent-capsules and their probabilities a.

Our dynamic routing algorithm is summarized in Algorithm 1.

2.4 Convolutional Capsule Layer

In this layer, each capsule is connected only to a local region $K_2 \times C$ spatially in the layer below. Those capsules in the region multiply transformation matrices to learn child-parent relationships followed by routing by agreement to produce parent capsules in the layer above.

Suppose $W^{c_1} \in \mathbb{R}^{D \times d \times d}$ and $W^{c_2} \in \mathbb{R}^{K_2 \times C \times D \times d \times d}$ denote shared and non-shared weights, respectively, where $K_2 \cdot C$ is the number of child capsules in a local region in the layer below, D is the number of parent capsules which the child capsules are sent to. When the transformation matrices are shared across the child capsules, each potential parent-capsule $\hat{u}_{j|i}$ is produced by

$$\hat{u}_{j|i} = W_j^{c_1} u_i + \hat{b}_{j|i} \tag{10}$$

where $\hat{b}_{j|i}$ is the capsule bias term, u_i is a child capsule in a local region $K_2 \times C$ and $W_j^{c_1}$ is the j_{th} matrix in tensor W^{c_1}. Then, we use routing-by-agreement to produce parent capsules feature maps totally $(L - K_1 - K_2 + 2) \times D$ d-dimensional capsules in this layer. When using the non-shared weights across the child capsules, we replace the transformation matrix $W_j^{c_1}$ in Eq. (10) with $W_j^{c_2}$.

2.5 Fully Connected Capsule Layer

The capsules in the layer below are flattened into a list of capsules and fed into fully connected capsule layer in which capsules are multiplied by transformation matrix $W^{d_1} \in \mathbb{R}^{E \times d \times d}$ or $W^{d_2} \in \mathbb{R}^{H \times E \times d \times d}$ followed by routing-by-agreement to produce final capsule $v_j \in \mathbb{R}^d$ and its probability $a_j \in \mathbb{R}$ for each category. Here, H is the number of child capsules in the layer below, E is the number of categories plus an extra orphan category.

2.6 The Architectures of Capsule Network

We explore two capsule architectures (denoted as Capsule-A and Capsule-B) to integrate these four

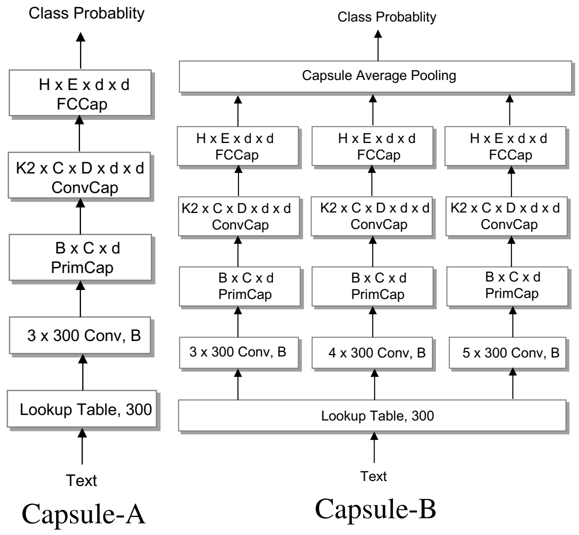

Figure 2: Two architectures of capsule networks.

components in different ways, as depicted in Figure 2.

Capsule-A starts with an embedding layer which transforms each word in the corpus to a 300-dimensional ($V = 300$) word vector, followed by a 3-gram ($K_1 = 3$) convolutional layer with 32 filters ($B = 32$) and a stride of 1 with ReLU non-linearity. All the other layers are capsule layers starting with a $B \times d$ primary capsule layer with 32 filters ($C = 32$), followed by a $3 \times C \times d \times d$ ($K_2 = 3$) convolutional capsule layer with 16 filters ($D = 16$) and a fully connected capsule layer in sequence.

Each capsule has 16-dimensional ($d = 16$) instantiated parameters and their length (norm) can describe the probability of the existence of capsules. The capsule layers are connected by the transformation matrices, and each connection is also multiplied by a routing coefficient that is dynamically computed by routing by agreement mechanism.

The basic structure of Capsule-B is similar to Capsule-A except that we adopt three parallel networks with filter windows (N) of 3, 4, 5 in the N-gram convolutional layer (see Figure 2). The final output of the fully connected capsule layer is fed into the average pooling to produce the final results. In this way, Capsule-B can learn more meaningful and comprehensive text representation.

3 Experimental Setup

3.1 Experimental Datasets

In order to evaluate the effectiveness of our model, we conduct a series of experiments on six benchmarks including: movie reviews (MR) (Pang and Lee, 2005), Stanford Sentiment Treebankan extension of MR (SST-2) (Socher et al., 2013), Subjectivity dataset (Subj) (Pang and Lee, 2004), TREC question dataset (TREC) (Li and Roth, 2002), customer review (CR) (Hu and Liu, 2004), and AG's news corpus (Conneau et al., 2017). These benchmarks cover several text classification tasks such as sentiment classification, question categorization, news categorization. The detailed statistics are presented in Table 1.

Dataset	Train	Dev	Test	Classes	Classification Task
MR	8.6k	0.9k	1.1k	2	review classification
SST-2	8.6k	0.9k	1.8k	2	sentiment analysis
Subj	8.1k	0.9k	1.0k	2	opinion classification
TREC	5.4k	0.5k	0.5k	6	question categorization
CR	3.1k	0.3k	0.4k	2	review classification
AG's news	108k	12.0k	7.6k	4	news categorization

Table 1: Characteristics of the datasets.

3.2 Implementation Details

In the experiments, we use 300-dimensional word2vec (Mikolov et al., 2013) vectors to initialize embedding vectors. We conduct mini-batch with size 50 for AG's news and size 25 for other datasets. We use Adam optimization algorithm with 1e-3 learning rate to train the model. We use 3 iteration of routing for all datasets since it optimizes the loss faster and converges to a lower loss at the end.

3.3 Baseline methods

In the experiments, we evaluate and compare our model with several widely used baseline methods including: LSTM/Bi-LSTM (Cho et al., 2014), tree-structured LSTM (Tree-LSTM) (Tai et al., 2015), LSTM regularized by linguistic knowledge (LR-LSTM) (Qian et al., 2016), CNN-rand/CNN-static/CNN-non-static (Kim, 2014), very deep convolutional network (VD-CNN) (Conneau et al., 2017), and character-level convolutional network (CL-CNN) (Zhang et al., 2015).

4 Experimental Results

4.1 Quantitative Evaluation

In our experiments, the evaluation metric is classification accuracy. We summarize the experimental results in Table 2. From the results, we observe that the capsule networks achieve best results on 4 out of 6 benchmarks, which verifies the effectiveness of the capsule networks. In particular, our model substantially and consistently outperforms

	MR	**SST2**	**Subj**	**TREC**	**CR**	**AG's**
LSTM	75.9	80.6	89.3	86.8	78.4	86.1
BiLSTM	79.3	83.2	90.5	89.6	82.1	88.2
Tree-LSTM	80.7	85.7	91.3	91.8	83.2	90.1
LR-LSTM	81.5	**87.5**	89.9	-	82.5	-
CNN-rand	76.1	82.7	89.6	91.2	79.8	92.2
CNN-static	81.0	86.8	93.0	92.8	84.7	91.4
CNN-non-static	81.5	87.2	93.4	**93.6**	84.3	92.3
CL-CNN	-	-	88.4	85.7	-	92.3
VD-CNN	-	-	88.2	85.4	-	91.3
Capsule-A	81.3	86.4	93.3	91.8	83.8	92.1
Capsule-B	**82.3**	86.8	**93.8**	92.8	**85.1**	**92.6**

Table 2: Comparisons of our capsule networks and baselines on six text classification benchmarks.

Dataset	Train	Dev	Test	Description
Reuters-Multi-label	5.8k	0.6k	0.3k	only multi-label data in test
Reuters-Full	5.8k	0.6k	3.4k	full data in test

Table 3: Characteristics of Reuters-21578 corpus.

the simple deep neural networks such as LSTM, Bi-LSTM and CNN-rand by a noticeable margin on all the experimental datasets. Capsule network also achieves competitive results against the more sophisticated deep learning models such as LR-LSTM, Tree-LSTM, VC-CNN and CL-CNN. Note that Capsule-B consistently performs better than Capsule-A since Capsule-B allows to learn more meaningful and comprehensive text representation. For example, a combination of N-gram convolutional layer with filter windows of $\{3,4,5\}$ can capture the 3/4/5-gram features of the text which play a crucial role in text modeling.

4.2 Ablation Study

To analyze the effect of varying different components of our capsule architecture for text classification, we also report the ablation test of the capsule-B model in terms of using different setups of the capsule network. The experimental results are summarized in Table 5. Generally, all three proposed dynamic routing strategies contribute to the effectiveness of Capsule-B by alleviating the disturbance of some noise capsules which may contain "background" information such as stop words and the words that are unrelated to specific categories.

5 Single-Label to Multi-Label Text Classification

Capsule network demonstrates promising performance in single-label text classification which assigns a label from a predefined set to a text (see Table 2). Multi-label text classification is, however, a more challenging practical problem. From single-label to multi-label (with n category labels) text classification, the label space is expanded from n to 2^n, thus more training is required to cover the whole label space. For single-label texts, it is practically easy to collect and annotate the samples. However, the burden of collection and annotation for a large scale multi-label text dataset is generally extremely high. How deep neural networks (e.g., CNN and LSTM) best cope with multi-label text classification still remains a problem since obtaining large scale of multi-label dataset is a time-consuming and expensive process. In this section, we investigate the capability of capsule network on multi-label text classification by using only the single-label samples as training data. With feature property as part of the information extracted by capsules, we may generalize the model better to multi-label text classification without an over extensive amount of labeled data.

The evaluation is carried on the Reuters-21578 dataset (Lewis, 1992). This dataset consists of 10,788 documents from the Reuters financial newswire service, where each document contains either multiple labels or a single label. We re-process the corpus to evaluate the capability of capsule networks of transferring from single-label to multi-label text classification. For dev and training, we only use the single-label documents in the Reuters dev and training sets. For testing, Reuters-Multi-label only uses the multi-label documents in testing dataset, while Reuters-Full includes all documents in test set. The characteristics of these two datasets are described in Table 3.

Following (Sorower, 2010), we adopt Micro Averaged Precision (Precision), Micro Averaged Recall (Recall) and Micro Averaged F1 scores (F1) as the evaluation metrics for multi-label text classification. Any of these scores are firstly computed on individual class labels and then averaged over all classes, called label-based measures. In addition, we also measure the Exact Match Ratio (ER) which considers partially correct prediction as incorrect and only counts fully correct samples.

The experimental results are summarized in Table 4. From the results, we can observe that the capsule networks have substantial and significant improvement in terms of all four evaluation metrics over the compared baseline methods on the test sets in both Reuters-Multi-label and Reuters-Full datasets. In particular, larger improvement

	Reuters-Multi-label				Reuters-Full			
	ER	Precision	Recall	F1	ER	Precision	Recall	F1
LSTM	23.3	86.7	54.7	63.5	62.5	78.6	72.6	74.0
BiLSTM	26.4	82.3	55.9	64.6	65.8	83.7	75.4	77.8
CNN-rand	22.5	88.6	56.4	67.1	63.4	78.7	71.5	73.6
CNN-static	27.1	91.1	59.1	69.7	63.3	78.5	71.2	73.3
CNN-non-static	27.4	92.0	59.7	70.4	64.1	80.6	72.7	75.0
Capsule-A	57.2	88.2	80.1	82.0	66.0	83.9	**80.5**	80.2
Capsule-B	**60.3**	**95.4**	**82.0**	**85.8**	**67.7**	**86.4**	80.1	**81.4**

Table 4: Comparisons of the capability for transferring from single-label to multi-label text classification on Reuters-Multi-label and Reuters-Full datasets. For fair comparison, we use margin-loss for our model and other baselines.

	Iteration	**Accuracy**
Capsule-B + Sabour's routing	3	81.4
Capsule-B + our routing	1	81.4
Capsule-B + our routing	3	82.3
Capsule-B + our routing	5	81.6
w/o Leaky-softmax	3	81.7
w/o Orphan Category	3	81.9
w/o Amendent Coeffient	3	82.1

Table 5: Ablation study of Capsule-B on MR dataset. The standard routing is routing-by-agreement algorithm without leaky-softmax and orphan category in the last capsule layer. More ablations are discussed in Appendix.

is achieved on Reuters-Multi-label dataset which only contains the multi-label documents in the test set. This is within our expectation since the capsule network is capable of preserving the instantiated parameters of the categories trained by single-label documents. The capsule network has much stronger transferring capability than the conventional deep neural networks. In addition, the good results on Reuters-Full also indicate that the capsule network has robust superiority over competitors on single-label documents.

5.1 Connection Strength Visualization

To visualize the connection strength between capsule layers clearly, we remove the convolutional capsule layer and make the primary capsule layer followed by the fully connected capsule layer directly, where the primary capsules denote N-gram phrases in the form of capsules. The connection strength shows the importance of each primary capsule for text categories, acting like a parallel attention mechanism. This should allow the capsule networks to recognize multiple categories in the text even though the model is trained on single-label documents.

Due to space reasons, we choose a multi-label document from Reuters-Multi-label test set whose category labels (i.e., Interest Rates and Money/Foreign Exchange) are correctly predicted (fully correct) by our model with high confidence ($p > 0.8$) to report in Table 6. The category-specific phrases such as "interest rates" and "foreign exchange" are highlighted with red color. We use the tag cloud to visualize the 3-gram phrases for Interest Rates and Money/Foreign Exchange categories. The stronger the connection strength, the bigger the font size. From the results, we observe that capsule networks can correctly recognize and cluster the important phrases with respect to the text categories. The histograms are used to show the intensity of connection strengths between primary capsules and the fully connected capsules, as shown in Table 6 (bottom line). Due to space reasons, five histograms are demonstrated. The routing procedure correctly routes the votes into the Interest Rates and Money/Foreign Exchange categories.

To experimentally verify the convergence of the routing algorithm, we also plot learning curve to show the training loss over time with different iterations of routing. From Figure 3, we observe that the Capsule-B with 3 or 5 iterations of routing optimizes the loss faster and converges to a lower loss at the end than the capsule network with 1 iteration.

6 Related Work

Early methods for text classification adopted the typical features such as bag-of-words, n-grams, and their TF-IDF features (Zhang et al., 2008) as

U.K. MONEY RATES FIRM ON LAWSON STERLING TARGETS	Interest Rates	Money/Foreign Exchange
Interest rates on the London money market were slightly firmer on news U.K. Chancellor of the Exchequer Nigel Lawson had stated target rates for sterling against the dollar and mark, dealers said. They said this had come as a surprise and expected the targets, 2.90 marks and 1.60 dlrs, to be promptly tested in the foreign exchange markets. Sterling opened 0.3 points lower in trade weighted terms at 71.3. Dealers noted the chancellor said he would achieve his goals on sterling by a combination of intervention in currency markets and interest rates. Operators feel the foreign exchanges are likely to test sterling on the downside and that this seems to make a fall in U.K. Base lending rates even less likely in the near term, dealers said. The feeling remains in the market, however, that fundamental factors have not really changed and that a rise in U.K. Interest rates is not very likely. The market is expected to continue at around these levels, reflecting the current 10 pct base rate level, for some time. The key three months interbank rate was 1/16 point firmer at 10 9-7/8 pct.		

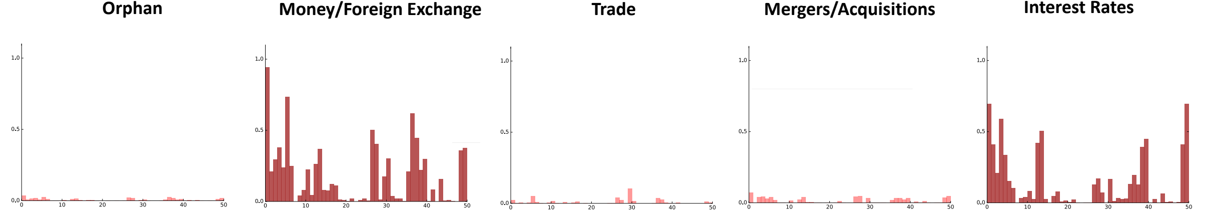

Table 6: Visualization of connection strength between primary capsules and the FC capsules by 3-gram phrases cloud and histogram of the their intensities. x axis denotes primary capsules (3-gram phrases) selected for demonstration, y axis denotes intensity of connection strength. The results are retrieved from Capsule-B trained with 3 routing iterations. The category-specific key-phrases in red color in raw text (first column) are annotated manually for reference.

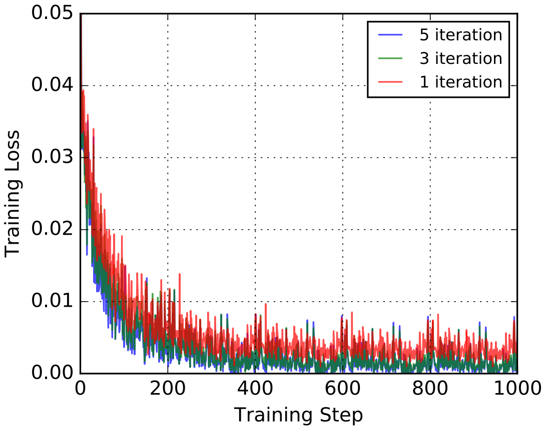

Figure 3: Training loss of Capsule-B on Reuters-Multi-label dataset.

input of machine learning algorithms such as support vector machine (SVM) (Joachims, 1998), logistic regression (Genkin et al., 2007), naive Bayes (NB) (McCallum et al., 1998) for classification. However, these models usually heavily relied on laborious feature engineering or massive extra linguistic resources.

Recent advances in deep neural networks and representation learning have substantially improved the performance of text classification tasks. The dominant approaches are recurrent neural networks, in particular LSTMs and CNNs. (Kim, 2014) reported on a series of experiments with CNNs trained on top of pre-trained word vectors for sentence-level classification tasks. The CNN models improved upon the state of the art on 4 out of 7 tasks. (Zhang et al., 2015) offered an empirical exploration on the use of character-level convolutional networks (Convnets) for text classification and the experiments showed that Convnets outperformed the traditional models. (Joulin et al., 2016) proposed a simple and efficient text classification method fastText, which could be trained on a billion words within ten minutes. (Conneau et al., 2017) proposed a very deep convolutional networks (with 29 convolutional layers) for text classification. (Tai et al., 2015) generalized the LSTM to the tree-structured network topologies (Tree-LSTM) that achieved best results on two text classification tasks.

Recently, a novel type of neural network is proposed using the concept of capsules to improve the representational limitations of CNN and RNN. Hinton et al. (2011) firstly introduced the concept of "capsules" to address the representational limitations of CNNs and RNNs. Capsules with transformation matrices allowed networks to automatically learn part-whole relationships. Consequently, Sabour et al. (2017) proposed capsule networks that replaced the scalar-output feature detectors of CNNs with vector-output capsules and max-pooling with routing-by-agreement. The capsule network has shown its potential by achieving a state-of-the-art result on MNIST data. Unlike max-pooling in CNN, however, Capsule network do not throw away information about the precise position of the entity within the region. For lowlevel capsules, location information is place-coded by which capsule is active. (Xi et al., 2017) further tested out the application of capsule networks on CIFAR data with higher dimensionality. (Hinton et al., 2018) proposed a new iterative routing procedure between capsule layers based on the EM algorithm, which achieves significantly better accuracy on the smallNORB data set. (Zhang et al., 2018) generalized existing routing methods within the framework of weighted kernel density estimation. To date, no work investigates the performance of capsule networks in NLP tasks. This study herein takes the lead in this topic.

7 Conclusion

In this paper, we investigated capsule networks with dynamic routing for text classification. Three strategies were proposed to boost the performance of the dynamic routing process to alleviate the disturbance of noisy capsules. Extensive experiments on six text classification benchmarks show the effectiveness of capsule networks in text classification. More importantly, capsule networks also show significant improvement when transferring single-label to multi-label text classifications over the co baseline methods.

References

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116.

Alexander Genkin, David D Lewis, and David Madigan. 2007. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304.

Geoffrey Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with em routing. *https://openreview.net/forum?id=HJWLfGWRb*.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. In *Annual meeting on Association for Computational Linguistics*, pages 427–431.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, pages 1746–1751. Association for Computational Linguistics.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

David D. Lewis. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–50.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Jon D Mcauliffe and David M Blei. 2008. Supervised topic models. In *Advances in neural information processing systems*, pages 121–128.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Amr Mousa and Björn Schuller. 2017. Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1023–1032.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. 2016. Linguistically regularized lstms for sentiment classification. *arXiv preprint arXiv:1611.03949*.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Mohammad S Sorower. 2010. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1556–1566.

Edgar Xi, Selina Bing, and Yang Jin. 2017. Capsule network performance on complex data. *arXiv preprint arXiv:1712.03480*.

Jianbo Ye, Yanran Li, Zhaohui Wu, James Z Wang, Wenjie Li, and Jia Li. 2017. Determining gains acquired from word embedding quantitatively using discrete distribution clustering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1847–1856.

Suofei Zhang, Wei Zhao, Xiaofu Wu, and Quan Zhou. 2018. Fast dynamic routing based on weighted kernel density estimation. *arXiv preprint arXiv:1805.10807*.

Wen Zhang, Taketoshi Yoshida, and Xijin Tang. 2008. Tfidf, lsi and multi-word in information retrieval and text categorization. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 108–113. IEEE.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Topic Memory Networks for Short Text Classification

Jichuan Zeng[1]*, **Jing Li**[2]†, **Yan Song**[2], **Cuiyun Gao**[1], **Michael R. Lyu**[1], **Irwin King**[1]
[1] Department of Computer Science and Engineering
The Chinese University of Hong Kong, HKSAR, China
[2] Tencent AI Lab, Shenzhen, China
[1] `{jczeng,cygao,lyu,king}@cse.cuhk.edu.hk`
[2] `{ameliajli,clksong}@tencent.com`

Abstract

Many classification models work poorly on short texts due to data sparsity. To address this issue, we propose *topic memory networks* for short text classification with a novel topic memory mechanism to encode latent topic representations indicative of class labels. Different from most prior work that focuses on extending features with external knowledge or pre-trained topics, our model jointly explores topic inference and text classification with memory networks in an end-to-end manner. Experimental results on four benchmark datasets show that our model outperforms state-of-the-art models on short text classification, meanwhile generates coherent topics.

1 Introduction

Short texts have become an important form for individuals to voice opinions and share information on online platforms. A large body of daily-generated contents, such as tweets, web search snippets, news feeds, and forum messages, have far outpaced the reading and understanding capacity of individuals. As a consequence, there is a pressing need for automatic language understanding techniques for processing and analyzing such texts (Zhang et al., 2018). Among those techniques, text classification is a critical and fundamental one proven to be useful in various downstream applications, such as text summarization (Hu et al., 2015), recommendation (Zhang et al., 2012), and sentiment analysis (Chen et al., 2017).

Although many classification models like support vector machines (SVMs) (Wang and Manning, 2012) and neural networks (Kim, 2014; Xiao and Cho, 2016; Joulin et al., 2017) have demonstrated their success in processing formal and well-edited texts, such as news articles (Zhang et al., 2015b), their performance is inevitably compromised when directly applied to short and informal online texts. This inferior performance is attributed to the severe data sparsity nature of short texts, which results in the limited features available for classifiers (Phan et al., 2008). To alleviate the data sparsity problem, some approaches exploit knowledge from external resources like Wikipedia (Jin et al., 2011) and knowledge bases (Lucia and Ferrari, 2014; Wang et al., 2017a). These approaches, however, rely on a large volume of high-quality external data, which may be unavailable to some specific domains or languages (Li et al., 2016a).

Training instances
R_1: [SuperBowl] I'll do anything to see the **Steelers** win.
R_2: [New.Music.Live] Please give **wristbands**, she have major **Bieber** Fever.

Test instance
S: [New.Music.Live] I will do anything for **wristbands**, gonna tweet till I win.

Table 1: Tweet examples for classification. R_i denotes the i-th training instance; S denotes a test instance. [class] is the ground-truth label. **Bold** words are indicative of an instance's class label.

To illustrate the difficulties in classifying short texts, we take the tweet classification in Table 1 as an example. In the test instance S, only given the 11 words it contains, it is difficult to understand why its label is New.Music.Live. Without richer context, classifiers are likely to classify S into the same category as the training instance R_1, which happens to share many words with S, in spite of the different categories they belong to,[1] rather than R_2, which only shares the word "*wristbands*" with S. Under this circumstance, how might we enrich the context of these short texts? If looking at R_2, we can observe that the semantic meaning of "*wristbands*" can be extended from its co-

---

* This work was mainly conducted when Jichuan Zeng was an intern in Tencent AI Lab.

† Jing Li is the corresponding author.

[1] R_1 is about SuperBowl, the annual championship game of the National Football League. R_2 and S are both about New.Music.Live, the flagship live music show.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3120–3131
Brussels, Belgium, October 31 - November 4, 2018.

occurrence with "*Bieber*", which is highly indicative of New.Music.Live.[2] Such relation can further help in recognizing the word "*wristbands*" to be important when classifying the test instance S.

Motivated by the above-mentioned observations, we present a novel neural framework, named as *topic memory networks* (TMN), for short text classification that does not rely on external knowledge. Our model can identify the indicative words for classification, e.g., "*wristbands*" in S, via jointly exploiting the document-level word co-occurrence patterns, e.g., "*wristbands*" and "*Bieber*" in R_2. To be more specific, built upon the success of neural topic models (Srivastava and Sutton, 2017; Miao et al., 2017), our model is capable of discovering *latent topics*[3], which can capture the co-occurrence of words in document level. To employ the *latent topics* for short text classification, we propose a novel *topic memory mechanism*, which is inspired by memory networks (Weston et al., 2014; Graves et al., 2014), that allows the model to put attention upon the indicative latent topics useful to classification. With such corpus-level latent topic representations, each short text instance is enriched, which thus helps alleviate the data sparsity issues.

In prior research, though the effects of topic models for short text classification have been explored (Phan et al., 2008; Ren et al., 2016), existing methods tend to use pre-trained topics as features. To the best of our knowledge, our model is the first to encode latent topic representations via memory networks for short text classification, which allows joint inference of latent topics.

To evaluate our model, we experiment and compare it with existing methods on four benchmark datasets. Experimental results indicate that our model outperforms state-of-the-art counterparts on short text classification. The quantitative and qualitative analysis illustrate the capability of our model in generating topic representations that are meaningful and indicative of different categories.

2 Topic Memory Networks

In this section, we describe our topic memory networks (TMN), whose overall architecture is shown in Figure 1. There are three major components: (1) a neural topic model (NTM) to induce latent topics (described in Section 2.1), (2) a topic memory mechanism that maps the inferred latent topics to classification features (described in Section 2.2), and (3) a text classifier, which produces the final classification labels for instances. These three components can be updated simultaneously via a joint learning process, which is introduced in Section 2.3. In particular, for the classifier, our TMN framework allows the combination of multiple options, e.g., CNN and RNN, which can be determined by the specific application scenario.

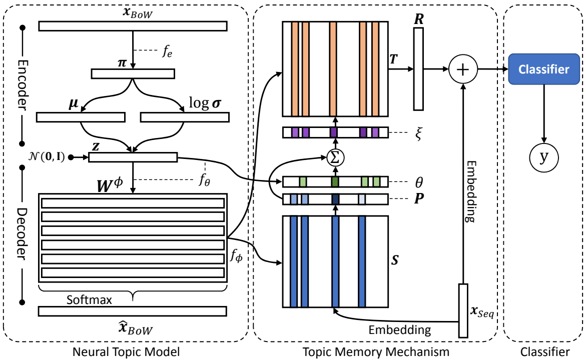

Figure 1: The overall framework of our topic memory networks. The dotted boxes from left to right show the neural topic model, the topic memory mechanism, and the classifier. Here the classifier allows multiple options and the details are left out.

Formally, given $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^M\}$ as the input with M short text instances, each instance $\mathbf{x}$ is processed into two representations: bag-of-words (BoW) term vector $\mathbf{x}_{BoW} \in \mathbb{R}^V$ and word index sequence vector $\mathbf{x}_{Seq} \in \mathbb{R}^L$, where V is the vocabulary size and L is the sequence length. $\mathbf{x}_{BoW}$ is fed into the neural topic model to induce latent topics. Such topics are further matched with the embedded $\mathbf{x}_{Seq}$ to learn classification features in the topic memory mechanism. Then, the classifier concatenates the representations produced by the topic memory mechanism and the embedded $\mathbf{x}_{Seq}$ to predict the classification label y for $\mathbf{x}$.

2.1 Neural Topic Model

Our topic model is inspired by neural topic model (NTM) (Miao et al., 2017; Srivastava and Sutton, 2017) that induces latent topics in neural networks. NTM is based on variational auto-encoder (VAE) (Kingma and Welling, 2013), involved with a continuous latent variable $\mathbf{z}$ as an intermediate representation. Here in NTM, the latent variable

---

[2] Justine Bieber was on New.Music.Live in 2011. There was a business activity for this event that gave free wristbands to fans if they supported Bieber on Twitter.

[3] Latent topics are the distributional clusters of words that frequently co-occur in some of the instances instead of widely appearing throughout the corpus (Blei et al., 2003).

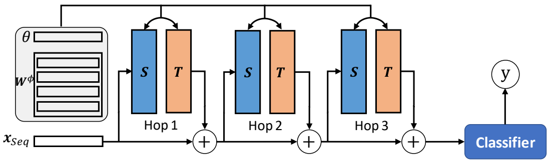

Figure 2: Topic memory network with three hops.

$\mathbf{z} \in \mathbb{R}^K$, where K denotes the number of topics. In the following, we describe the generation and the inference of the model in turn.

NTM Generation. Similar to LDA-style topic models, we assume $\mathbf{x}$ having a topic mixture θ represented as a K-dimensional distribution, which is generated via Gaussian softmax construction (Miao et al., 2017). Each topic k is represented by a word distribution ϕ_k over the vocabulary. Specifically, the generation story for $\mathbf{x}$ is:

- Draw latent variable $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$
- $\theta = \text{softmax}(f_\theta(\mathbf{z}))$
- For the n-th word in $\mathbf{x}$:
  - Draw word $w_n \sim \text{softmax}(f_\phi(\theta))$

where $f_*(\cdot)$ is a neural perceptron that linearly transforms inputs, activated by a non-linear transformation. Here we use rectified linear units (ReLUs) (Nair and Hinton, 2010) as activate functions. The prior parameters of $\mathbf{z}$, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, are estimated from the input data and defined as:

$$\boldsymbol{\mu} = f_\mu(f_e(\mathbf{x}_{BoW})), \log \boldsymbol{\sigma} = f_\sigma(f_e(\mathbf{x}_{BoW})) \tag{1}$$

Note that NTM is based on VAE, where an encoder estimates the prior parameters and a decoder describes the generation story. Compared with the basic VAE, NTM includes the additional distributional vectors θ and ϕ, which can yield latent topic representations and thus ensuring their better interpretability in learning process (Miao et al., 2017).

NTM Inference. In NTM, we use variational inference (Blei et al., 2016) to approximate a posterior distribution over $\mathbf{z}$ given all the instances. The loss function of NTM is defined as

$$\mathcal{L}_{NTM} = D_{KL}(q(\mathbf{z}) \,||\, p(\mathbf{z} \,|\, \mathbf{x})) - \mathbb{E}_{q(\mathbf{z})}[p(\mathbf{x} \,|\, \mathbf{z})] \tag{2}$$

the negative of variational lower bound, where $q(\mathbf{z})$ is a standard Normal prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$. $p(\mathbf{z} \,|\, \mathbf{x})$ and $p(\mathbf{x} \,|\, \mathbf{z})$ are probabilities to describe encoding and decoding processes, respectively.[4] Due to the space limitation, we leave out the derivation details and refer the readers to Miao et al. (2017).

2.2 Topic Memory Mechanism

We exploit a topic memory mechanism to map the latent topics produced by NTM (described in Section 2.1) to the features for classification. Inspired by memory networks (Weston et al., 2014; Sukhbaatar et al., 2015), we design two memory matrices, a source memory $\boldsymbol{S}$ and a target memory $\boldsymbol{T}$, both of which are in $K \times E$ size (K for the number of topics and E for the pre-defined size of word embeddings). $\boldsymbol{S}$ and $\boldsymbol{T}$ are produced by two ReLU-actived neural perceptrons, both taking the topic-word weight matrix $\boldsymbol{W}^\phi \in \mathbb{R}^{K \times V}$ as inputs. Recall that in NTM, we use $f_\phi(\cdot)$ to compute the word distributions given θ. $\boldsymbol{W}^\phi$ is the kernel weight matrix of $f_\phi(\cdot)$, where $\boldsymbol{W}^\phi_{k,v}$ represents the importance of the v-th word in reflecting the k-th topic. Assuming $\boldsymbol{U}$ as the embedded $\mathbf{x}_{Seq}$ (word sequence form of $\mathbf{x}$), in source memory, we compute the match between the k-th topic and the embedding of the l-th word in $\mathbf{x}_{Seq}$ by

$$\boldsymbol{P}_{k,l} = \text{sigmoid}(\mathbf{W}^s[\boldsymbol{S}_k; \boldsymbol{U}_l] + b^s) \tag{3}$$

where $[\mathbf{x}; \mathbf{y}]$ denotes the merge of $\mathbf{x}$ and $\mathbf{y}$, and we use concatenation operation here (Dou, 2017; Chen et al., 2017). $\mathbf{W}^s$ and b^s are parameters to be learned. To further combine the instance-topic mixture θ with $\boldsymbol{P}$, we define the integrated memory weights as

$$\xi_k = \theta_k + \gamma \sum_l \boldsymbol{P}_{k,l} \tag{4}$$

where γ is the pre-defined coefficient. Then, in target memory, via weighting target memory matrix $\boldsymbol{T}$ with ξ, we obtain the output representation $\boldsymbol{R}$ of the topic memory mechanism:

$$\boldsymbol{R}_k = \xi_k \boldsymbol{T}_k \tag{5}$$

The concatenation of $\boldsymbol{R}$ and $\boldsymbol{U}$ (embedded $\mathbf{x}_{Seq}$) further serves as the features for classification.

In particular, similar to the memory networks in prior research (Sukhbaatar et al., 2015; Chen et al., 2017), our model can be extended to handle multiple computation layers (hops). As shown in Figure 2, each hop contains a source matrix and a target matrix, and different hops are stacked following the way presented in Sukhbaatar et al. (2015).

---

[4] In implementation, to smooth the gradients, we apply reparameterization on $\mathbf{z}$ following previous work (Kingma and Welling, 2013; Rezende et al., 2014).

Dataset	# of labels	# of docs	Avg len per doc	Vocab size
Snippets	8	12,332	17	7,334
TagMyNews	7	32,567	8	9,433
Twitter	50	15,056	5	6,962
Weibo	50	21,944	6	10,121

Table 2: Statistics of the experimental datasets. Labels refers to class labels. Avg len per doc refers to the average count of words in each document instance.

2.3 Joint Learning

The entire TMN model integrates the three modules in Figure 1, i.e., the neural topic model, the topic memory mechanism, and the classifier, which can be updated simultaneously in one framework. In doing so, we jointly tackle topic modeling and classification, and define the loss function of the overall framework to combine the two effects as following:

$$\mathcal{L} = \mathcal{L}_{NTM} + \lambda \mathcal{L}_{CLS} \tag{6}$$

where $\mathcal{L}_{NTM}$ represents the loss of NTM and $\mathcal{L}_{CLS}$ is the cross entropy reflecting classification loss. λ is the trade-off parameter controlling the balance between topic model and classification.

3 Experiment Setup

3.1 Datasets

We conduct experiments on four short text datasets, namely, Snippets, TagMyNews, Twitter, and Weibo. Their details are described as follows.

Snippets. This dataset contains Google search snippets released by Phan et al. (2008). There are eight ground-truth labels, e.g., *health* and *sport*.

TagMyNews. We use the news titles as instances from the benchmark classification dataset released by Vitale et al. (2012).[5] This dataset contains English news from really simple syndication (RSS) feeds. Each news feed (with its title) is annotated with one from seven labels, e.g., *sci-tech*.

Twitter. This dataset is used to evaluate tweet topic classification, which is built on the dataset released by TREC2011 microblog track.[6] Following previous settings (Yan et al., 2013; Li et al., 2016a), hashtags, i.e., user-annotated topic labels in each tweet such as "*#Trump*" and "*#SuperBowl*", serve as our ground-truth class labels. Specifically, we construct the dataset with the following steps. First, we remove the tweets without hashtags. Second, we rank hashtags by their frequencies. Third, we manually remove the hashtags that cannot mark topics, such as "*#fb*" for indicating the source of tweets from Facebook, and combine the hashtags referring to the same topic, such as "*#DonaldTrump*" and "*#Trump*". Finally, we select the top 50 frequent hashtags, and all tweets containing these hashtags.

Weibo. To evaluate our model on a different language other than English, we employ a Chinese dataset with short segments of text for topic classification. This dataset is released by Li et al. (2016b) with a collection of messages posted in June 2014 on Weibo, a popular Twitter alike platform in China.[7] Similar to Twitter, Weibo allows up to 140 Chinese characters in its messages. In this Weibo dataset, each Weibo message is labeled with a hashtag as its category, and there are 50 distinct hashtag labels in total, following the same procedure performed for the Twitter dataset.

Table 2 shows the statistic information of the four datasets. Each dataset is randomly split into 80% for training and 20% for test. 20% of randomly selected training instances are used to form development set. We preprocess our English datasets, i.e., Snippets, TagMyNews, and Twitter, with *gensim tokenizer*[8] for tokenization. As to the Chinese Weibo dataset, we use FudanNLP toolkit (Qiu et al., 2013)[9] for word segmentation. In addition, for each dataset, we maintain a vocabulary built based on the training set with removal of stop words[10] and words occurring less than 3 times. The inputs of topic models $\mathbf{x}_{BoW}$ are constructed based on this vocabulary following common topic model settings (Blei et al., 2003; Miao et al., 2016). Differently, we use the raw word sequence (without words removal) for the inputs of classification $\mathbf{x}_{Seq}$ as is done in previous work of text classification (Kim, 2014; Liu et al., 2017).

3.2 Model Settings

We use pre-trained embeddings to initialize all word embeddings. For Snippets and TagMyNews

---

[5] `http://acube.di.unipi.it/tmn-dataset/`

[6] `http://trec.nist.gov/data/tweets`

[7] The original dataset contains conversations to enrich the context of Weibo posts, which are not considered here.

[8] `https://radimrehurek.com/gensim/utils.html`

[9] `https://github.com/FudanNLP/fnlp`

[10] `https://radimrehurek.com/gensim/parsing/preprocessing.html`

Models	Snippets		TagMyNews		Twitter		Weibo	
	Acc	Avg F1	Acc	Avg F1	Acc	Avg F1	Acc	Avg F1
Comparison models								
Majority Vote	0.202	0.068	0.247	0.098	0.073	0.010	0.102	0.019
SVM+BOW (Wang and Manning, 2012)	0.210	0.080	0.259	0.058	0.070	0.009	0.116	0.039
SVM+LDA (Blei et al., 2003)	0.689	0.694	0.616	0.593	0.159	0.111	0.192	0.147
SVM+BTM (Yan et al., 2013)	0.772	0.772	0.686	0.677	0.232	0.164	0.331	0.277
SVM+NTM (Miao et al., 2017)	0.779	0.776	0.664	0.654	0.261	0.177	0.379	0.348
AttBiLSTM (Zhang and Wang, 2015)	0.943	0.943	0.838	0.828	0.375	0.348	0.547	0.547
CNN (Kim, 2014)	0.944	0.944	0.843	0.843	0.381	0.362	0.553	0.550
CNN+TEWE (Ren et al., 2016)	0.944	0.944	0.846	0.846	0.385	0.368	0.537	0.532
CNN+NTM	0.945	0.945	0.844	0.844	0.382	0.365	0.556	0.556
Our models								
TMN (*Separate TM Inference*)	0.961	0.961	0.848	0.847	0.394	**0.386**	0.568	0.569
TMN (*Joint TM Inference*)	**0.964**	**0.964**	**0.851**	**0.851**	**0.397**	0.375	**0.591**	**0.589**

Table 3: Comparisons of accuracy (Acc) and average F1 (Avg F1) on four benchmark datasets. Our TMN, either with separate or joint TM inference, performs significantly better than all the comparisons ($p < 0.05$, paired t-test).

datasets, we use pre-trained `GloVe` embeddings (Pennington et al., 2014)[11]. For Twitter and Weibo datasets, we pre-train embeddings on large-scale external data with 99M tweets and 467M Weibo messages, respectively. For the number of topics, we follow previous settings (Yan et al., 2013; Das et al., 2015; Dieng et al., 2016) to set $K = 50$. For all the other hyperparameters, we tune them on the development set by grid search. For our classifier, we employ CNN in experiment because of its better performance in short text classification than its counterparts such as RNN (Wang et al., 2017a). The hidden size of CNN is set as 500. The dimension of word embedding $E = 200$. $\gamma = 0.8$ for trading off θ and $\boldsymbol{P}$, and $\lambda = 1.0$ for controlling the effects of topic model and classification. In the learning process, we run our model for 800 epochs with early-stop strategy applied (Caruana et al., 2000).

3.3 Comparison Models

For comparison, we consider a weak baseline of majority vote, which assigns the major class labels in training set to all test instances. We further compare with the widely-used baseline SVM+BOW, SVM with unigram features (Wang and Manning, 2012). We also consider other SVM-based baselines: SVM+LDA, SVM+BTM, SVM+NTM, whose features are topic distributions for instances learned by LDA (Blei et al., 2003), BTM (Yan et al., 2013), and NTM (Miao et al., 2017), respectively. In particular, BTM is one of the state-of-the-art topic models for short texts. To compare with neural classifiers, we test bidirectional long short-term memory with attention (AttBiLSTM) (Zhang et al., 2015a) and convolutional neural network (CNN) classifiers (Kim, 2014). No topic representation is encoded in these two classifiers. We also compare with the state-of-the-art short-text classifier CNN+TEWE (Ren et al., 2016), i.e., CNN classifier with topic-enriched word embeddings (TEWE), where the word embeddings are enriched by pre-trained NTM-inferred topic models. Moreover, to investigate the effectiveness of our proposed topic memory mechanism, we compare with CNN+NTM, which concatenates the representations learned by CNN and topics induced by NTM as classification features. In addition, we compare with our variant, TMN (*Separate TM Inference*), where topics are induced separately before classification, and only used for initializing the topic memory. To be consistent, our model with a joint learning process for topic modeling and classification, described in Section 2.3, is named as TMN (*Joint TM Inference*). Note that the comparison CNN-based models share the same settings as our model, and the hidden size for each direction of BiLSTM is set to 100.

4 Experimental Results

4.1 Classification Comparison

Table 3 shows the comparison on classification results, where the accuracy and average F1 scores on different classes labels are reported. We have the following observations.

• ***Topic representations are indicative features.*** On all four datasets, simply by combining topic representations into features, SVM models produce better results than the models without ex-

---

[11] `http://nlp.stanford.edu/data/glove.6B.zip` (200d)

Model	Snippets	TagMyNews	Twitter
LDA	0.436	0.449	0.436
BTM	0.435	0.463	0.435
NTM	0.463	0.468	0.463
TMN	**0.487**	**0.499**	**0.468**

Table 4: C_V coherence scores for topics generated by various models. Higher is better. The best result in each column is in **bold**.

LDA	mubarak bring run obama democracy speech believe regime power bowl
BTM	mubarak egypt push internet people government phone hosni need son
NTM	mubarak people egyptian egypt stay tomorrow protest news phone protester
TMN	mubarak protest protester tahrir square egyptian al jazeera repo cairo

Table 5: Top 10 representative terms of the sample latent topics discovered by various topic models from Twitter dataset. We interpret the topics as "*Egyptian revolution of 2011*" according to their word distributions. Non-topic words are wave-underlined and in blue, and off-topic words are underlined and in red.

ploiting topic features (*i.e.*, SVM+BOW). This observation indicates that latent topic representations captured at corpus level are helpful to alleviate the data sparsity problem in short text classification.

• ***Neural network models are effective.*** It is seen that neural models based on either CNN or AttBiLSTM yield better results than SVM. This observation shows the effectiveness of representation learning in neural networks for short texts.

• ***CNN serves as a better classifier for short texts than AttBiLSTM.*** In comparison of CNN and AttBiLSTM without taking topic features, we observe that CNN yields generally better results on all the four datasets. This is consistent with the discovery in Wang et al. (2017a), where CNN can better encode short texts than sequential models.

• ***Topic memory is useful to classification.*** By exploring topic representations in memory mechanisms, our TMN model, inferring topic models either separately or jointly with classification, significantly outperform the best comparison models on each of the four datasets. Particularly, when compared with CNN+TEWE and CNN+NTM, both concatenating topics as part of the features, the results yielded by TMN are better. This demonstrates the effectiveness of topic memory to learn indicative topic representations for short text classification.

• ***Jointly inferring latent topics is effective to text classification.*** In comparison between two TMN variants, TMN (*Joint TM Inference*) produces better classification results, though large margin improvements are not observed on the three English datasets, i.e., TagMyNews, Snippets, and Twitter. This may be because the classifiers do not rely too much on high-quality latent topics, since other features may be sufficient to indicate the labels, e.g., word positions in the instance. As a result, better topic models, learned via jointly induced with classification, may not provide richer information for classification. Nevertheless, we notice that on Chinese Weibo dataset, the jointly trained topic model improves the accuracy and average F1 by 2.3% and 2.0%, respectively. It may result from the prevalence of word order misuse in informal Weibo messages. This mis-order phenomenon is common in Chinese and generally does not affect understanding. The rich information conveyed by Chinese characters are capable of indicating semantic meanings of words even without correct orders (Qin et al., 2016; Wang et al., 2017b). As a result, the CNN classifier, which encodes orders of words, may also bring such mis-order noise to classification. For these instances with misordered words, a better topic model that learns text instances as unordered words, provides useful representations that compensate the loss of information in word orders and in turn improves the performance of text classification.

4.2 Topic Coherence Comparison

In Section 4.1, we find that TMN can significantly outperform comparison models on short text classification. In this section, we study whether jointly learning topic models and classification can be helpful in producing coherent and meaningful topics. We use the C_V metric (Röder et al., 2015) computed by Palmetto toolkit[12] to evaluate the topic coherence, which has been shown to give the closest scores to human evaluation compared to other widely-used topic coherence metrics like NPMI (Bouma, 2009). Table 4 shows the comparison results of LDA, BTM, NTM, and TMN on the three English datasets.[13] Note that we do not report C_V scores for Chinese Weibo dataset as the Palmetto toolkit cannot process Chinese topics.

---

[12] `https://github.com/dice-group/Palmetto`

[13] In the rest of this paper, without otherwise indicated, TMN is used as a short form for TMN (*Joint TM Inference*).

# of Hops	Snippets	TagMyNews	Twitter	Weibo
TMN-1H	0.958	0.841	0.382	0.568
TMN-2H	**0.964**	0.843	0.383	0.578
TMN-3H	0.962	0.845	0.384	0.581
TMN-4H	0.961	0.846	0.389	0.582
TMN-5H	0.960	**0.851**	**0.397**	**0.591**
TMN-6H	0.958	0.848	0.388	0.579

Table 6: The impact of the # of hops on accuracy.

As can be seen, TMN yields higher C_V scores by large margins than all others in comparison. This indicates that jointly exploring classification would be effective in producing coherent topics. The reason is that the supervision from classification labels can guide unsupervised topic models in discovering meaningful and interpretable topics. We also observe that NTM produces better results than LDA and BTM, which implies the effectiveness of inducing topic models by neural networks.

To further analyze the quality of yielded topics, Table 5 shows the top 10 words of the sample latent topics reflecting "*Egyptian revolution of 2011*" discovered by various models. We find that LDA yields off-topic word "*bowl*". For the results of BTM and NTM, though we do not find off-topic words, non-topic words like "*need*" and "*stay*" are included.[14] The topic generated by TMN appears to be the best, which presents indicative words like "*tahrir*" and "*cairo*", for the event.

4.3 Results with Varying Hyperparameters

We further study the impact of two important hyperparameters in TMN, i.e., the hop number and the topic number, which will be discussed in turn.

Impact of Hop Numbers. Recall that Figure 2 shows the capacity of TMN in combining multiple hops. Here we analyze the effects of hop numbers on the accuracy of TMN. Table 6 reports the results, where NH refers to using N hops ($N = 1, 2, ..., 6$). As can be seen, generally, TMN with 5 hops achieves the best accuracy on most datasets except for Snippets dataset. We also observe that, although within a particular range, more hops can produce better accuracy, the increasing trends are not always monotonic. For example, TMN-6H always exhibits lower accuracy than TMN-5H. This observation implies that the overall representation ability of TMN is enhanced as the increasing complexity of the model via combining more hops. However, this enhancement will reach saturation when the hop number exceeds a threshold, which is 5 hops for most datasets in our experiment.

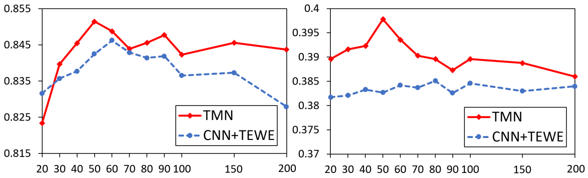

(a) TagMyNews (b) Twitter

Figure 3: The impact of topic numbers, where the horizontal axis shows the number of topics and the vertical axis shows the accuracy.

Impact of Topic Numbers. Figure 3 shows the accuracy of TMN and CNN+TEWE (the best comparison model in Table 3) given varying K, the number of topics on TagMyNews and Twitter datasets.[15] As we can see, the curves of all the models are not monotonic and the best accuracy is achieved given a particular number of topics, e.g., K=50 for TMN on TagMyNews dataset. When comparing different curves, we observe that TMN yields consistently better accuracy than CNN+TEWE, a comparison model shown in Table 3, which demonstrates the robust performance of TMN over varying number of topics.

4.4 A Case Study on Topic Memory

Section 4.1 demonstrates the effectiveness of using topic memory on short text classification. To further understand why, in this section, we use the test instance S in Table 1 to analyze what the information captured by topic memory is indicative of class labels. Recall that the label of S, which should be New.Music.Live, can be indicated by containing word "*wristbands*" and the collocation of "*wristbands*" and "*Bieber*" in training instance R_2 labeled New.Music.Live. Figure 4 shows the heatmaps of the weight matrix $\boldsymbol{P}$ in topic memory and the topic mixture θ captured by NTM for instance S. As can be seen, the top 3 words for the latent topic with the largest value in θ are "*bieber*", "*justine*", and "*tuesday*", which can effectively indicate the class label of S to be *New.Music.Live* because Justine Bieber was there on Tuesday. Interestingly, S contains none of the top three words. The latent semantic relations of S and these words are purely uncovered by the co-occurrence of words in S with other instances in

---

[14] Off-topic words are more likely to be interpreted to reflect other topics. Non-topic words cannot clearly indicate the corresponding topic.

[15] We observe similar distributions on Snippets and Weibo.

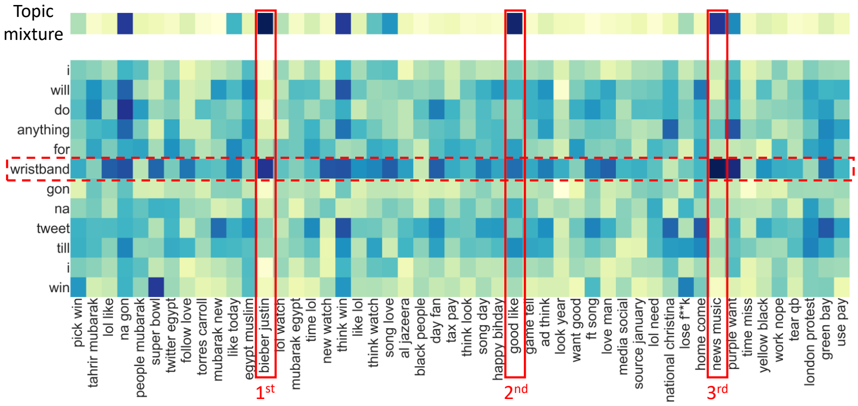

(a)

1st Topic	2nd Topic	3rd Topic
bieber justin tuesday newmusiclive think thought jb new music live	good like fan look tweet thing know rumble na right	news music forever change right attack time wing aistdirect films

(b)

Figure 4: Topic memory visualization for test instance S shown in Table 1. (a) Heatmaps of topic mixture θ (the upper one) and topic memory weight matrix $\boldsymbol{P}$ (the lower one) illustrating the relevance between the words of S (left) and the learned topics (bottom, with top-2 words displayed). The red dotted rectangle indicates the representation for "*wristband*", the topical word in S. The red rectangles with solid frames indicates the 3 most relevant topics ordered by θ. (b) Top-10 words of these topics indicated by ϕ.

the corpus, which further shows the benefit of using latent topics for alleviating the sparsity in short texts. We also observe that topic memory learns different representations for topical word "*wristband*", highly indicating instance label, and background words, such as "*i*" and "*for*". This explains why topic memory is effective to classification.

4.5 Error Analysis

In this section, we take our classification results on TagMyNews dataset as an example to analyze our errors. We observe that one major type of incorrect prediction should be ascribed to the polysemy phenomenon. For example, the instance "*NBC gives 'the voice' post super bowl slot*" should be categorized as *entertainment*. However, failing to understand the particular meaning of "*the voice*" here as the name of a television singing competition, our model mistakenly categorizes this instance as sport because of the occurrence "*super bowl*". In future work, we would exploit context-sensitive topical word embeddings (Witt et al., 2016), which is able to distinguish the meanings of the same word in different contexts. Another main error type comes from the failure to capture phrase-level semantics. Taking "*On the merits of face time and living small*" as an example, without understanding "*face time*" as a phrase, our model wrongly predicts its category as business instead of its correct label as sci_tech. Such errors can be reduced by enhancing our NTM to phrase discovery topic models (Lindsey et al., 2012; He, 2016), which is worthy exploring in future work.

5 Related Work

Our work mainly builds on two streams of prior work: short text classification and topic models.

Short Text Classification. In the line of short text classification, most work focuses on alleviating the severe sparsity issues in short texts (Yan et al., 2013). Some previous efforts encode knowledge from external resource (Jin et al., 2011; Lucia and Ferrari, 2014; Wang et al., 2017a; Ma et al., 2018). Instead, our work learns effective representations only from internal data. For some specific classification tasks, such as sentiment analysis, manually-crafted features are designed to fit the target task (Pak and Paroubek, 2010; Jiang et al., 2011). Distinguished from them, we employ deep learning framework for representation learning, which requires no feature engineering process and thus ensures its general applicability to diverse classification scenarios. In comparison with the established classifiers applying deep learning methods (dos Santos and Gatti, 2014; Lee and Dernoncourt, 2016), our work differs from them in the leverage of corpus-level latent topic representations for alleviating data sparsity issues. In existing classification models using topic features, pre-trained topic mixtures are leveraged as part of features (Phan et al., 2008; Ren et al., 2016; Chen et al., 2017). Differently, our model encodes topic representations in a memory mechanism where topics are induced jointly with text classification in an end-to-end manner.

Topic Models. Well-known topic models, e.g., probabilistic latent semantic analysis (pLSA) (Hofmann, 1999) and latent Dirichlet allocation (LDA) (Blei et al., 2003), have shown advantages in capturing effective semantic representations, and proven beneficial to varying downstream applications, such as summarization (Haghighi and Vanderwende, 2009) and recommendation (Zeng et al., 2018; Bai et al., 2018). For short text data, topic model variants have been proposed to reduce the effects of sparsity issues on topic modeling, such as biterm topic model (BTM) (Yan et al., 2013) and LeadLDA (Li et al., 2016b). Recently, owing to the popularity of variational auto-encoder (VAE) (Kingma and Welling, 2013), it is able to induce latent topics in neural networks, namely, neural topic models (NTM) (Miao et al., 2017; Srivastava and Sutton, 2017). Although the concept of NTM has been mentioned earlier in Cao et al. (2015), their model is based on matrix factorization. Differently, VAE-style NTM (Srivastava and Sutton, 2017; Miao et al., 2017) follows the LDA fashion as probabilistic generative models, which is easy to interpret and extend. The NTM in our framework is in VAE-style, whose effects on short text classification serve as the key focus of our work.

6 Conclusion

We have presented topic memory networks that exploit corpus-level topic representations with a topic memory mechanism for short text classification. The model alleviates data sparsity issues via jointly learning latent topics and text categories. Empirical comparisons with state-of-the-art models on four benchmark datasets have demonstrated the validity and effectiveness of our model, where better results have been achieved on both short text classification and topic coherence evaluation.

Acknowledgements

This work is partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 and No. CUHK 14234416 of the General Research Fund), and Microsoft Research Asia (2018 Microsoft Research Asia Collaborative Research Award). We thank Shuming Shi, Dong Yu, Tong Zhang, Baolin Peng, Haoli Bai, and the three anonymous reviewers for the insightful suggestions on various aspects of this work.

References

Haoli Bai, Zhuangbin Chen, Michael R. Lyu, Irwin King, and Zenglin Xu. 2018. Neural Relational Topic Model for Scientific Article Analysis. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018*. ACM.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2016. Variational Inference: A Review for Statisticians. *CoRR*, abs/1601.00670.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Gerlof Bouma. 2009. Normalized (Pointwise) Mutual Information in Collocation Extraction. *Proceedings of GSCL*, pages 31–40.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A Novel Neural Topic Model and Its Supervised Extension. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas, USA, pages 2210–2216.

Rich Caruana, Steve Lawrence, and C. Lee Giles. 2000. Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, an"d Early Stopping. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems, NIPS 2000*, Denver, CO, USA, pages 402–408.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent Attention Network on Memory for Aspect Sentiment Analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, Copenhagen, Denmark, pages 452–461.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for Topic Models with Word Embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, Volume 1: Long Papers*, Beijing, China, pages 795–804.

Adji B. Dieng, Chong Wang, Jianfeng Gao, and John William Paisley. 2016. TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency. *CoRR*, abs/1611.01702.

Zi-Yi Dou. 2017. Capturing User and Product Information for Document Level Sentiment Analysis with Deep Memory Network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, Copenhagen, Denmark, pages 521–526.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. *CoRR*, abs/1410.5401.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2009*, pages 362–370, Boulder, Colorado, USA.

Yulan He. 2016. Extracting Topical Phrases from Clinical Documents. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, USA, pages 2957–2963.

Thomas Hofmann. 1999. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99*, Berkeley, CA, USA, pages 50–57.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. LCSTS: A Large Scale Chinese Short Text Summarization Dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, Lisbon, Portugal, pages 1967–1972.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent Twitter Sentiment Classification. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, Portland, Oregon, USA, pages 151–160.

Ou Jin, Nathan Nan Liu, Kai Zhao, Yong Yu, and Qiang Yang. 2011. Transferring Topical Knowledge from Auxiliary Long Texts for Short Text Clustering. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011*, Glasgow, United Kingdom, pages 775–784.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL*, Doha, Qatar, pages 1746–1751.

Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *CoRR*, abs/1312.6114.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks. In *The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016*, San Diego California, USA, pages 515–520.

Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. 2016a. Topic Modeling for Short Texts with Auxiliary Word Embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016*, Pisa, Italy, pages 165–174.

Jing Li, Ming Liao, Wei Gao, Yulan He, and Kam-Fai Wong. 2016b. Topic Extraction from Microblog Posts Using Conversation Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Volume 1: Long Papers*, Berlin, Germany.

Robert V. Lindsey, William Headden, and Michael Stipicevic. 2012. A Phrase-Discovering Topic Model Using Hierarchical Pitman-Yor Processes. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012*, Jeju Island, Korea, pages 214–222.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 1: Long Papers*, Vancouver, Canada, pages 1–10.

William Lucia and Elena Ferrari. 2014. EgoCentric: Ego Networks for Knowledge-based Short Text Classification. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014*, Shanghai, China, pages 1079–1088.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted Aspect-Based Sentiment Analysis via Embedding Commonsense Knowledge into an Attentive LSTM. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering Discrete Latent Topics with Neural Variational Inference. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, Sydney, NSW, Australia, pages 2410–2419.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural Variational Inference for Text Processing. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, New York City, NY, USA, pages 1727–1736.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Haifa, Israel, pages 807–814.

Alexander Pak and Patrick Paroubek. 2010. Twitter Based System: Using Twitter for Disambiguating Sentiment Ambiguous Adjectives. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010*, Uppsala University, Uppsala, Sweden, pages 436–439.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL*, Doha, Qatar, pages 1532–1543.

Xuan Hieu Phan, Minh Le Nguyen, and Susumu Horiguchi. 2008. Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008*, Beijing, China, pages 91–100.

Zengchang Qin, Yonghui Cong, and Tao Wan. 2016. Topic Modeling of Chinese Language beyond a Bag-of-Words. *Computer Speech & Language*, 40:60–78.

Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. FudanNLP: A Toolkit for Chinese Natural Language Processing. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Improving Twitter Sentiment Classification Using Topic-Enriched Multi-Prototype Word Embeddings. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, USA, pages 3038–3044.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*, Beijing, China, pages 1278–1286.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the Space of Topic Coherence Measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015*, Shanghai, China, pages 399–408.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, COLING 2014*, Dublin, Ireland, pages 69–78.

Akash Srivastava and Charles Sutton. 2017. Autoencoding Variational Inference For Topic Models. *arXiv preprint arXiv:1703.01488*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems, NIPS 2015*, Montreal, Quebec, Canada, pages 2440–2448.

Daniele Vitale, Paolo Ferragina, and Ugo Scaiella. 2012. Classification of Short Texts by Deploying Topical Annotations. In *Advances in Information Retrieval - 34th European Conference on IR Research, ECIR 2012*, Barcelona, Spain, pages 376–387.

Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017a. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, Melbourne, Australia, pages 2915–2921.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2017b. Exploiting Word Internal Structures for Generic Chinese Sentence Representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, Copenhagen, Denmark, pages 298–303.

Sida I. Wang and Christopher D. Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, ACL 2012, Volume 2: Short Papers*, Jeju Island, Korea, pages 90–94.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory Networks. *CoRR*, abs/1410.3916.

Nils Witt, Christin Seifert, and Michael Granitzer. 2016. Explaining Topical Distances Using Word Embeddings. In *27th International Workshop on Database and Expert Systems Applications, DEXA 2016 Workshops*, Porto, Portugal, pages 212–217.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers. *CoRR*, abs/1602.00367.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A Biterm Topic Model for Short Sexts. In *22nd International World Wide Web Conference, WWW 2013*, Rio de Janeiro, Brazil, pages 1445–1456.

Xingshan Zeng, Jing Li, Lu Wang, Nicholas Beauchamp, Sarah Shugars, and Kam-Fai Wong. 2018. Microblog conversation recommendation via joint modeling of topics and discourse. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, New Orleans, Louisiana, USA, pages 375–385.

Dongxu Zhang and Dong Wang. 2015. Relation Classification via Recurrent Neural Network. *CoRR*, abs/1508.01006.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015a. Bidirectional Long Short-Term Memory Networks for Relation Classification. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation, PACLIC 29*, Shanghai, China.

Weinan Zhang, Dingquan Wang, Gui-Rong Xue, and Hongyuan Zha. 2012. Advertising Keywords Recommendation for Short-Text Web Pages Using Wikipedia. *ACM TIST*, 3(2):36:1–36:25.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015b. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems, NIPS 2015*, Montreal, Quebec, Canada, pages 649–657.

Yingyi Zhang, Jing Li, Yan Song, and Chengzhi Zhang. 2018. Encoding conversation context for neural keyphrase extraction from microblog posts. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, New Orleans, Louisiana, USA, pages 1676–1686.

Few-Shot and Zero-Shot Multi-Label Learning for Structured Label Spaces

Anthony Rios
Department of Computer Science
University of Kentucky
Lexington, KY
`anthony.rios1@uky.edu`

Ramakanth Kavuluru
Division of Biomedical Informatics
University of Kentucky
Lexington, KY
`ramakanth.kavuluru@uky.edu`

Abstract

Large multi-label datasets contain labels that occur thousands of times (frequent group), those that occur only a few times (few-shot group), and labels that never appear in the training dataset (zero-shot group). Multi-label few- and zero-shot label prediction is mostly unexplored on datasets with large label spaces, especially for text classification. In this paper, we perform a fine-grained evaluation to understand how state-of-the-art methods perform on infrequent labels. Furthermore, we develop few- and zero-shot methods for multi-label text classification when there is a known structure over the label space, and evaluate them on two publicly available medical text datasets: MIMIC II and MIMIC III. For few-shot labels we achieve improvements of 6.2% and 4.8% in R@10 for MIMIC II and MIMIC III, respectively, over prior efforts; the corresponding R@10 improvements for zero-shot labels are 17.3% and 19%.

1 Introduction

Unlike in binary or multi-class problems, for multi-label classification a model assigns a set of labels to each input instance (Tsoumakas et al., 2010). Large-scale multi-label text classification problems can be found in several domains. For example, Wikipedia articles are annotated with labels used to organize documents and facilitate search (Partalas et al., 2015). Biomedical articles indexed by the PubMed search engine are manually annotated with medical subject headings (Tsatsaronis et al., 2012). In healthcare facilities, medical records are assigned a set of standardized codes for billing purposes (NCHS, 1978). Automatically annotating tweets with hashtags, while the labels are not fixed, can also be represented as a large-scale multi-label classification problem (Weston et al., 2014).

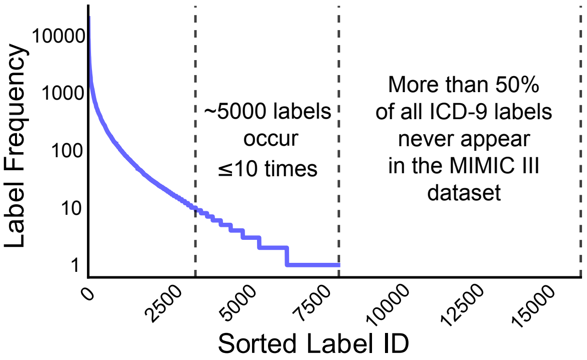

Figure 1: This plot shows the label frequency distribution of ICD-9 codes in MIMIC III.

There are two major difficulties when developing machine learning methods for large-scale multi-label text classification problems. First, the documents may be long, sometimes containing more than a thousand words (Mullenbach et al., 2018). Finding the relevant information in a large document for a specific label results in needle in a haystack situation. Second, data sparsity is a common problem; as the total number of labels grows, a few labels may occur frequently, but most labels will occur infrequently. Rubin et al. (2012) refer to datasets that have long-tail frequency distributions as "power-law datasets". Methods that predict infrequent labels fall under the paradigm of few-shot classification which refers to supervised methods in which only a few examples, typically between 1 and 5, are available in the training dataset for each label. With predefined label spaces, some labels may never appear in the training dataset. Zero-shot problems extend the idea of few-shot classification by assuming no training data is available for the labels we wish to predict at test time. In this paper, we explore both of these issues, long documents and power-law datasets, with an emphasis on analyzing the few- and zero-shot aspects of large-scale multi-label problems.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3132–3142
Brussels, Belgium, October 31 - November 4, 2018.

In Figure 1, we plot the label frequency distribution of diagnosis and procedure labels for the entire MIMIC III (Johnson et al., 2016) dataset. A few labels occur more than 10,000 times, around 5,000 labels occur between 1 and 10 times, and of the 17,000 diagnosis and procedure labels, more than 50% never occur. There are a few reasons a label may never occur in the training dataset. In healthcare, sevearl disorders are rare; therefore corresponding labels may not have been observed yet in a particular clinic. Sometimes new labels may be introduced as the field evolves leading to an *emerging label* problem. This is intuitive for applications such as hashtag prediction on Twitter. For example, last year it would not have made sense to annotate tweets with the hashtag #EMNLP2018. Yet, as this year's conference approaches, labeling tweets with the #EMNLP2018 will help users find relevant information.

Infrequent labels may not contribute heavily to the overall accuracy of a multi-label model, but in some cases, correct prediction of such labels is crucial but not straightforward. For example, in assigning diagnosis labels to EMRs, it is important that trained human coders are both accurate and thorough. Errors may cause unfair financial burden on the patient. Coders may have an easier time assigning frequent labels to EMRs because they are encountered more often. Also, frequent labels are generally easier to predict using machine-learning based methods. However, infrequent or obscure labels will be easily confused or missed causing billing mistakes and/or causing the coders to spend more time annotating each record. Thus, we believe methods that handle infrequent and unseen labels in the multi-label setting are important.

Current evaluation methods for large-scale multi-label classification mostly ignore infrequent and unseen labels. Popular evaluation measures focus on metrics such as micro-F1, recall at k (R@k), precision at k (P@k), and macro-F1. As it is well-known that micro-F1 gives more weight to frequent labels, papers on this topic also report macro-F1, the average of label-wise F1 scores, which equally weights all labels. Unfortunately, macro-F1 scores are generally low and the corresponding performance differences between methods are small. Moreover, it is possible to improve macro-F1 by only improving a model's performance on frequent labels, further confounding its interpretation. Hence we posit that macro-F1 is not enough to compare large-scale multi-label learning methods on infrequent labels and it does not directly evaluate zero-shot labels. Here, we take a step back and ask: can the model predict the correct few-shot (zero-shot) labels from the set of all few-shot (zero-shot) labels? To address this, we test our approach by adapting the *generalized zero-shot classification* evaluation methodology by Xian et al. (2017) to the multi-label setting.

In this paper, we propose and evaluate a neural architecture suitable for handling few- and zero-shot labels in the multi-label setting where the output label space satisfies two constraints: (1). the labels are connected forming a DAG and (2). each label has a brief natural language descriptor. These assumptions hold in several multi-label scenarios including assigning diagnoses/procedures to EMRs, indexing biomedical articles with medical subject headings, and patent classification. Taking advantage of this prior knowledge on labels is vital for zero-shot prediction. Specifically, using the EMR coding use-case, we make the following contributions:

1. We overcome issues arising from processing long documents by introducing a new neural architecture that expands on recent attention-based CNNs (ACNNs (Mullenbach et al., 2018)). Our model learns to predict few- and zero-shot labels by matching discharge summaries in EMRs to feature vectors for each label obtained by exploiting structured label spaces with graph CNNs (GCNNs (Kipf and Welling, 2017)).

2. We provide a fine-grained evaluation of state-of-the-art EMR coding methods for frequent, few-shot, and zero-shot labels. By evaluating power-law datasets using an extended generalized zero-shot methodology that also includes few-shot labels, we present a nuanced analysis of model performance on infrequent labels.

2 Related Work

Large-Scale Text Classification. Linear methods have been successfully applied to large-scale problems (Tang et al., 2009; Papanikolaou et al., 2015; Rios and Kavuluru, 2015). For traditional micro- and macro-F1 measures, Tang et al. (2009) show that linear methods suffer using naive thresholding strategies because infrequent labels generally need a smaller threshold. Generative models have also been promising for datasets with many labels (Rubin et al., 2012). Intuitively, by using a prior distribution over the label space, infrequent labels can be modeled better. Finally, large-scale classification is also pursued as "extreme classification" (Yu et al., 2014; Bhatia et al., 2015) where the focus is on ranking measures that ignore infrequent labels. Neural networks (NNs) perform well for many small-scale classification tasks (Kim, 2014; Kalchbrenner et al., 2014). Recently, researchers have been exploring NN methods for large-scale problems. Yang et al. (2016) develop a hierarchical attentive NN for datasets with over a million documents, but their datasets contain few labels. Nam et al. (2014) show that feed-forward NNs can be successfully applied to large-scale problems through the use of a multi-label binary cross-entropy loss function. Vani et al. (2017) introduce a grounded recurrent neural network (RNN) that iteratively updates its predictions as it processes a document word-by-word. Baumel et al. (2018) experiment with both CNNs and RNNs for medical coding. Finally, Mullenbach et al. (2018) expand on prior ACNNs (Yang et al., 2016; Allamanis et al., 2016) to develop a label-wise attention framework where the most informative ngrams are extracted for each label in the dataset. Our attention mechanism extends their work to the zero-shot setting.

Few-Shot and Zero-Shot Learning. While neural networks are generally considered to need large datasets, they have been shown to work well on few-shot classification tasks. To handle infrequent labels, most NN methods use a k-NN-like approach. Siamese NNs (Koch et al., 2015) learn a nonlinear distance metric using a pairwise loss function. Matching networks (Vinyals et al., 2016) introduce an instance-level attention method to find relevant neighbors. Prototypical Networks (Snell et al., 2017) average all instances in each class to form "prototype label vectors" and train using a traditional cross-entropy loss. In our prior work (Rios and Kavuluru, 2018), we combine matching networks with a sophisticated thresholding strategy. However, in Rios and Kavuluru (2018) we did not explore the few- and zero-shot settings.

Zero-shot learning has not been widely explored in the large-scale multi-label classification scenario. Like neural few-shot methods, neural zero-shot methods use a matching framework. Instead of matching input instances with other instances, they are matched to predefined label vectors. For example, the Attributes and Animals Dataset (Xian et al., 2017) contains images of animals and the label vectors consist of features describing the types of animals (e.g., stripes: yes). When feature vectors for labels are not available, the average of the pretrained word embeddings of the class names have been used. The attribute label embedding method (Akata et al., 2016) uses a pairwise ranking loss to match zero-shot label vectors to instances. Romera-Paredes and Torr (2015) introduced the "embarrassingly simple zero-shot learning" (ESZSL) method which is trained using a mean squared error loss. A few zero-shot methods do not translate well to multi-label problems. CONSE (Mikolov et al., 2013) averages the embeddings for the top predicted supervised label vectors to match to zero-shot label vectors. CONSE assumes that both supervised and zero-shot labels cannot be assigned to the same instance. In this paper, we expand on the generalized zero-shot evaluation methodology introduced by Xian et al. (2017) to large-scale multi-label classification. Finally, it is important to note that zero-shot classification has been previously studied in the multi-label setting (Mensink et al., 2014). However, they focus on image classification and use datasets with around 300 labels.

Graph Convolutional Neural Networks. GCNNs generalize CNNs beyond 2d and 1d spaces. Defferrard et al. (2016) developed spectral methods to perform efficient graph convolutions. Kipf and Welling (2017) assume a graph structure is known over input instances and apply GCNNs for semi-supervised learning. GCNNs are applied to relational data (e.g., link prediction) by Schlichtkrull et al. (2018). GCNNs have also had success in other NLP tasks such as semantic role labeling (Marcheggiani and Titov, 2017), dependency parsing (Strubell and McCallum, 2017), and machine translation (Bastings et al., 2017).

There are three GCNN papers that share similarities with our work. (i) Peng et al. (2018) use a GCNN on a word co-occurrence graph for large-scale text classification where the GCNN operates on documents/words, while our GCNN operates on the labels. (ii) Chen et al. (2017) use GCNNs on structured label spaces. However, their

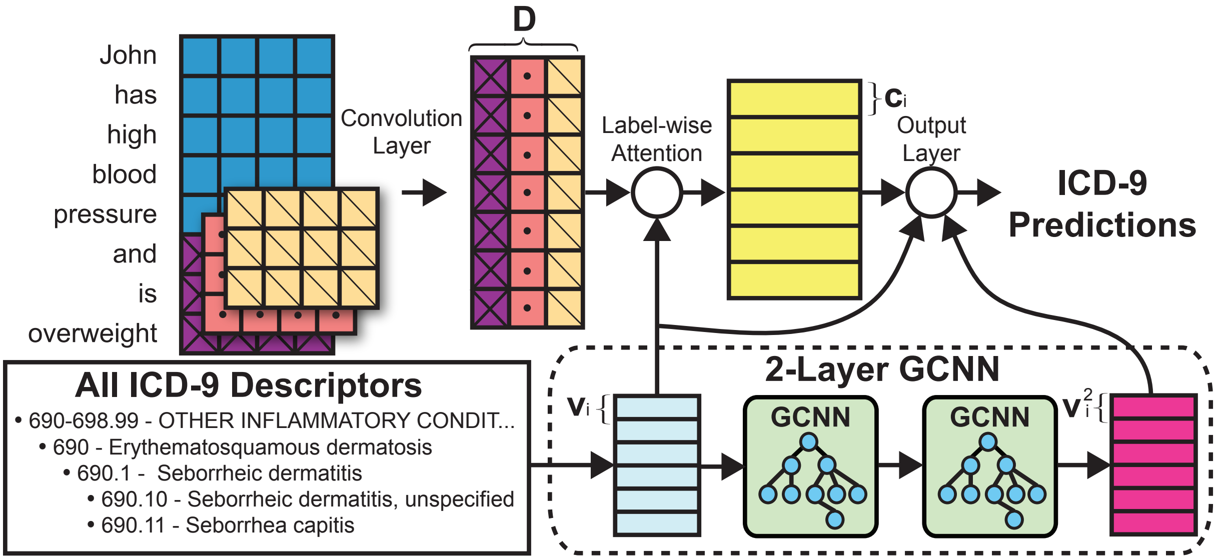

Figure 2: This figure provides a visual overview of our method. Intuitively, our method has two main components. The first component is a CNN that operates operates on the EMRs. The other component is a 2-layer GCNN which creates the label-specific attention vectors and label-vectors used for ranking using ICD-9 descriptions as input.

experiments focus on smaller label spaces and do not handle/assess zero-shot and few-shot labels. Also, their experiments for text classification do not incorporate attention and simply use an average of word vectors to represent each document. (iii) Wang et al. (2018) propose a zero-shot GCNN image classification method for structured multi--class problems. We believe their method may transfer to the multi-label text classification setting but exact modifications to affect that are not clear (i.e., their semi-supervised approach may not be directly applicable). Likewise, porting to text is nontrivial for long documents.

3 Method

Figure 2 shows the overall schematic of our architecture. Intuitively, we incorporate four main components. First, we assume we have the full English descriptor/gloss for each label we want to predict. We form a vector representation for each label by averaging the word embeddings for each word in its descriptor. Second, the label vectors formed from the descriptor are used as attention vectors (label-wise attention) to find the most informative ngrams in the document for each label. For each label, this will produce a separate vector representation of the input document. Third, the label vectors are passed through a two layer GCNN to incorporate hierarchical information about the label space. Finally, the vectors returned from the GCNN are matched to the document vectors to generate predictions.

Convolutional Neural Network. Contrary to prior CNN methods for text (Kim, 2014), instead of using a max-over-time pooling layer, we learn to find relevant ngrams in a document for each label via label-wise attention (Mullenbach et al., 2018). The CNN will return a document feature matrix $\mathbf{D} \in \mathbb{R}^{(n-s+1)\times u}$ where each column of $\mathbf{D}$ is a feature map, u is the total number of convolution filters, n is the number of words in the document, and s is the width of convolution filters.

Label Vectors. To be able to predict labels that were not in the training dataset, we avoid learning label specific parameters. We use the label descriptors to generate a feature vector for each label. First, to preprocess each descriptor, we lowercase all words and remove stop-words. Next, each label vector is formed by averaging the remaining words in the descriptor

$$\mathbf{v}_i = \frac{1}{|N|}\sum_{j\in N} \mathbf{w}_j, \qquad i = 1,\dots,L, \qquad (1)$$

where $\mathbf{v}_i \in \mathbb{R}^d$, L is the number of labels, and N is the index set of the words in the descriptor. Prior zero-shot work has focused on projecting input instances into the same semantic space as the label vectors (Sandouk and Chen, 2016). For zero-shot image classification, this is a non-trivial task. Because we work with textual data, we simply share the word embeddings between the convolutional layer and the label vector creation step to form $\mathbf{v}_i$.

Label-Wise Attention. Similar to the work by Mullenbach et al. (2018), we employ label-wise attention to avoid the needle in the haystack situation encountered with long documents. The issue with simply using a single attention vector or using max-pooling is that we assume a single vector can capture everything required to predict every label. For example, with a single attention, we would only look at one spot in the document and assume that spot contains the relevant information needed to predict all labels. In the multi-class setting, this assumption is plausible. However, for large multi-label problems, the relevant information for each label may be scattered throughout the document – the problem is worse when the documents are very long. Using label-wise attention, our model can focus on different sections. We also need to find relevant information for zero-shot classes. So we use the label vectors $\mathbf{v}_i$ rather than learning label specific attention parameters. First, we pass the document feature matrix $\mathbf{D}$ through a simple feed-forward neural network

$$\mathbf{D}^2 = \tanh(\mathbf{D}\,\mathbf{W}_b + \mathbf{b}_b)$$

where $\mathbf{W}_b \in \mathbb{R}^{u \times d}$ and $\mathbf{b}_b \in \mathbb{R}^d$. This mapping is important because the dimensionality of the ngram vectors (rows) in $\mathbf{D}$ depends on u, the number of scores we generate for each ngram. Given $\mathbf{D}^2$, we generate the label-wise attention vector

$$\mathbf{a}_i = softmax(\mathbf{D}^2\,\mathbf{v}_i), \quad i = 1, \ldots, L, \qquad (2)$$

where $\mathbf{a}_i \in \mathbb{R}^{n-s+1}$ measures how informative each ngram is for the i-th label. Finally, we use $\mathbf{D}$, and generate L label-specific document vector representations

$$\mathbf{c}_i = \mathbf{a}_i^T\,\mathbf{D}, \quad i = 1, \ldots, L,$$

such that $\mathbf{c}_i \in \mathbb{R}^u$. Intuitively, $\mathbf{c}_i$ is the weighted average of the rows in $\mathbf{D}$ forming a vector representation of the document for the i-th label.

GCNN Output Layer. Traditionally, the output layer of a CNN would learn label specific parameters optimized via a cross-entropy loss. Instead, our method attempts to match documents to their corresponding label vectors. In essence, this becomes a retrieval problem. Before using each document representation $\mathbf{c}_i$ to score its corresponding label, we take advantage of the structured knowledge we have over our label space using a 2-layer GCNN. For both the MIMIC II and MIMIC III datasets, this information is hierarchical. A snippet of the hierarchy can be found in Figure 2.

Starting with the label vectors $\mathbf{v}_i$, we combine the label vectors of the children and parents for the i-th label to form

$$\mathbf{v}_i^1 = f(\mathbf{W}^1\mathbf{v}_i + \sum_{j \in \mathcal{N}_p} \frac{\mathbf{W}_p^1\mathbf{v}_j}{|\mathcal{N}_p|} + \sum_{j \in \mathcal{N}_c} \frac{\mathbf{W}_c^1\mathbf{v}_j}{|\mathcal{N}_c|} + \mathbf{b}_g^1)$$

where $\mathbf{W}^1 \in \mathbb{R}^{q \times d}$, $\mathbf{W}_p^1 \in \mathbb{R}^{q \times d}$, $\mathbf{W}_c^1 \in \mathbb{R}^{q \times d}$, $\mathbf{b}_g^1 \in \mathbb{R}^q$, f is the rectified linear unit (Nair and Hinton, 2010) function, and $\mathcal{N}_c$ ($\mathcal{N}_p$) is the index set of the i-th label's children (parents). We use different parameters to distinguish each edge type. In this paper, given we only deal with hierarchies, the edge types include edges from parents, from children, and self edges. This can be adapted to arbitrary DAGs, where parent edges represent all incoming edges and the child edges represent all outgoing edges for each node.

The second layer follows the same formulation as the first layer with

$$\mathbf{v}_i^2 = f(\mathbf{W}^2\mathbf{v}_i^1 + \sum_{j \in \mathcal{N}_p} \frac{\mathbf{W}_p^2\mathbf{v}_j^1}{|\mathcal{N}_p|} + \sum_{j \in \mathcal{N}_c} \frac{\mathbf{W}_c^2\mathbf{v}_j^1}{|\mathcal{N}_c|} + \mathbf{b}_g^2)$$

where $\mathbf{W}^2 \in \mathbb{R}^{q \times q}$, $\mathbf{W}_p^2 \in \mathbb{R}^{q \times q}$, $\mathbf{W}_c^2 \in \mathbb{R}^{q \times q}$, and $\mathbf{b}_g^2 \in \mathbb{R}^q$. Next, we concatenate both the averaged description vector (from equation (1)) with the GCNN label vector to form

$$\mathbf{v}_i^3 = \mathbf{v}_i \mid\mid \mathbf{v}_i^2,$$

where $\mathbf{v}_i^3 \in \mathbb{R}^{d+q}$. Now, to compare the final label vector $\mathbf{v}_i^3$ with its document vector $\mathbf{c}_i$, we transform the document vector into

$$\mathbf{e}_i = ReLU(\mathbf{W}_o\mathbf{c}_i + \mathbf{b}_o), \quad i = 1, \ldots, L,$$

where $\mathbf{W}_o \in \mathbb{R}^{(q+d) \times u}$ and $\mathbf{b}_o \in \mathbb{R}^{q+d}$. This transformation is required to match the dimension to that of $\mathbf{v}_i^3$. Finally, the prediction for each label i is generated via

$$\hat{y}_i = sigmoid(\mathbf{e}_i^T\,\mathbf{v}_i^3), \quad i = 1, \ldots, L.$$

During experiments, we found that using either the output layer GCNN or a separate GCNN for the attention vectors (equation (2)) did not result in an improvement and severely slowed convergence.

Training. We train our model using a multi-label binary cross-entropy loss (Nam et al., 2014)

$$\mathcal{L} = \sum_{i=1}^{L} \big[- y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i) \big],$$

where $y_i \in \{0, 1\}$ is the ground truth for the i-th label and $\hat{y}_i$ is our sigmoid score for the i-th label.

4 Experiments

In this paper, we use two medical datasets for evaluation purposes: MIMIC II (Jouhet et al., 2012) and MIMIC III (Johnson et al., 2016). Both datasets contain discharge summaries annotated with a set of ICD-9 diagnosis and procedure labels. Discharge summaries are textual documents consisting of, but not limited to, physician descriptions of procedures performed, diagnoses made, the patient's medical history, and discharge instructions. Following a generalized zero-shot learning evaluation methodology (Xian et al., 2017), we split the ICD-9 labels into three groups based on frequencies in the training dataset: The frequent group S that contains all labels that occur > 5 times, the few-shot group F that contains labels that occur between 1 and 5 times, and the zero-shot group Z of labels that never occur in the training dataset, but occur in the test/dev sets. The groups are only used for evaluation. That is, during training, systems are *optimized over all labels simultaneously*. Instances that do not contain few- or zero-shot classes are removed from their respective groups during evaluation. This grouping is important to assess how each model performs across labels grouped by label frequency. Our evaluation methodology differs from that of Xian et al. (2017) in two ways. First, because each instance is labeled with multiple labels, the same instance can appear in all groups — S, F, and Z. Second, instead of top-1 accuracy or HIT@k evaluation measures, we focus on R@k to handle multiple labels. At a high level, we want to examine whether a model can distinguish the correct few-shot (zero-shot) labels from the set of all few-shot (zero-shot) labels. Therefore, the R@k measures in Tables 2 and 3, and Figure 3 are computed relative to each group.

Evaluation Measures. The overall statistics for these two datasets are reported in Table 1. For reproducibility purposes, we use the same training/test splits of the MIMIC II as Perotte et al. (2013). Following the procedures in Perotte et al. (2013) and Vani et al. (2017), for each diagnosis and procedure label assigned to each medical report, we add its parents using the ICD-9 hierarchy. Each report in MIMIC II is annotated with nearly 37 labels on average using hierarchical label expansion.

			# Labels		
Dataset	# Train	# Test	S	F	Z
MIMIC II	18822	1711	3228	3459	355
MIMIC III	37016	1356	4403	4349	178

Table 1: Dataset statistics for MIMIC II and MIMIC III.

MIMIC III does not contain a standardized training/test split. Therefore, we create our own split that ensures the same patient does not appear in both the training and test datasets. Unlike the MIMIC II dataset, we do not augment the labels using the ICD-9 hierarchy. The ICD-9 hierarchy has three main levels. For MIMIC III, level 0 labels make up about 5% of all occurrences, level 1 labels make up about 62%, and level 2 (leaf level) labels make up about 33%. Also, each MIMIC III instance contains16 ICD-9 labels on average.

ICD-9 Structure and Descriptors. The International Classification of Diseases (ICD) contains alphanumeric diagnosis and procedure codes that are used by hospitals to standardize their billing practices. In the following experiments, we use the 9th edition of the ICD[1]. Each ICD-9 identifier contains between 3 to 5 alphanumeric characters of the form $abc.xy$. The alphanumeric structure defines a simple hierarchy over all ICD-9 codes. For example, "systolic heart failure" (428.2) and "diastolic heart failure" (428.3) are both children of the "heart failure" code 428. Furthermore, sequential codes are grouped together. For instance, numeric codes in the range 390-459 contain "Diseases of the Circulatory System". Furthermore, each code, including groups of codes (390-459), contain short descriptors, where the average descriptor length contains seven words[2]. In this work, we use both the group descriptors and in-

---

[1] The US transitioned from ICD-9 to ICD-10 in 2015. Unfortunately, at the time of publication, large publicly available ICD-10 EMR datasets are unavailable.

[2] The descriptors and hierarchy used in this paper can be found at `https://bioportal.bioontology.org/ontologies/ICD9CM`

	S		F		Z		Harmonic Average	
	R@5	R@10	R@5	R@10	R@5	R@10	R@5	R@10
Random	0.000	0.000	0.000	0.000	0.011	0.032	0.000	0.000
Logistic (Vani et al., 2017) *	0.137	0.247	0.001	0.003	–	–	–	–
CNN (Baumel et al., 2018) *	0.138	0.250	0.050	0.082	–	–	–	–
ACNN (Mullenbach et al., 2018) *	**0.138**	**0.255**	0.046	0.081	–	–	–	–
Match-CNN (Rios and Kavuluru, 2018)	0.137	0.247	0.031	0.042	–	–	–	–
ESZSL + W2V	0.074	0.119	0.008	0.017	0.080	0.172	0.020	0.041
ESZSL + W2V 2	0.050	0.086	0.025	0.044	0.103	0.189	0.043	0.076
ESZSL + GRALS	0.135	0.238	0.081	0.123	0.085	0.136	0.095	0.152
ZACNN	0.135	0.245	0.103	0.149	0.147	0.221	0.128	0.205
ZAGCNN	0.135	0.247	**0.130**	**0.185**	**0.269**	**0.362**	**0.160**	**0.246**

Table 2: MIMIC II results across frequent (S), few-shot (F), and zero-shot (Z) groups. We mark prior methods for MIMIC datasets that we implemented with a *.

	S		F		Z		Harmonic Average	
	R@5	R@10	R@5	R@10	R@5	R@10	R@5	R@10
Random	0.000	0.000	0.000	0.000	0.038	0.052	0.000	0.000
Logistic (Vani et al., 2017) *	0.273	0.427	0.014	0.014	–	–	–	–
CNN (Baumel et al., 2018) *	0.269	0.413	0.058	0.085	–	–	–	–
ACNN (Mullenbach et al., 2018) *	**0.288**	**0.458**	0.130	0.168	–	–	–	–
Match-CNN (Rios and Kavuluru, 2018)	0.278	0.426	0.049	0.060	–	–	–	–
ESZSL + W2V	0.135	0.191	0.031	0.051	0.157	0.257	0.065	0.105
ESZSL + W2V 2	0.127	0.189	0.031	0.048	0.148	0.305	0.063	0.102
ESZSL + GRALS	0.256	0.393	0.033	0.060	0.076	0.138	0.064	0.114
ZACNN	0.278	0.435	0.152	0.195	0.364	0.442	0.232	0.310
ZAGCNN	0.283	0.445	**0.166**	**0.216**	**0.428**	**0.495**	**0.252**	**0.337**

Table 3: MIMIC III results across frequent (S), few-shot (F), and zero-shot (Z) groups. We mark prior methods for MIMIC datasets that we implemented with a *.

dividual descriptors as input to the GCNN. At test time, we ignore the group codes.

Implementation Details. For the CNN component of our model, we use 300 convolution filters with a filter size of 10. We use 300 dimensional word embeddings pretrained on PubMed biomedical article titles and abstracts. To avoid overfitting, we use dropout directly after the embedding layer with a rate of 0.2. For training we use the ADAM (Kingma and Ba, 2015) optimizer with a minibatch size of 8 and a learning rate of 0.001. q, the GCNN hidden layer size, is set to 300. The code for our method is available at `https://github.com/bionlproc/multi-label-zero-shot`.

Thresholding has a large influence on traditional multi-label evaluation measures such as micro-F1 and macro-F1 (Tang et al., 2009). Hence, we report both recall at k (R@k) and precision at k (P@k) which do not require a specific threshold. R@k is preferred for few- and zero-shot labels, because P@k quickly goes to zero as k increases and gets bigger than the number of group specific labels assigned to each instance. Furthermore, for medical coding, these models are typically used as a recommendation engine to help coders. Unless a label appears at the top of the ranking, the annotator will not see it. Thus, ranking metrics better measure the usefulness of our systems.

Baseline Methods. For the frequent and few-shot labels we compare to state-of-the-art methods on the MIMIC II and MIMIC III datasets including ACNN (Mullenbach et al., 2018) and a CNN method introduced in Baumel et al. (2018). We also compare with the L1 regularized logistic regression model used in Vani et al. (2017). Finally, we compare against our prior EMR coding method, Match-CNN (Rios and Kavuluru, 2018).

	P@10	R@10	Macro-F1
CNN	0.562	0.407	0.028
ACNN	**0.624**	**0.452**	**0.068**
Match-CNN	0.561	0.415	0.033
ZACNN	0.577	0.429	0.037
ZAGCNN	0.587	0.439	0.038

Table 4: P@k, R@k, and macro-F1 results over all labels (the union of S, F, and Z).

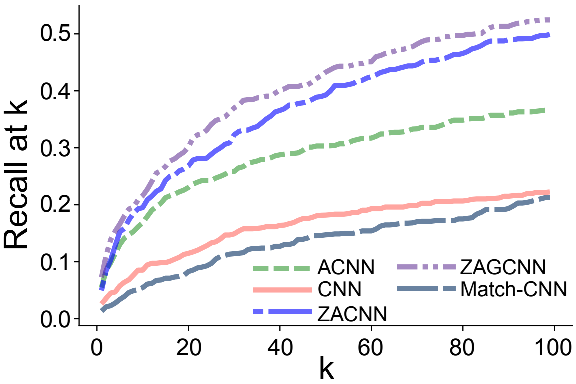

Figure 3: This graph plots the MIMIC III R@k for few-shot (F) labels at different k values.

For zero-shot learning, we compare our results with ESZSL (Romera-Paredes and Torr, 2015). To use ESZSL, we must specify feature vectors for each label. For zero-shot methods, the label vectors used are crucial regardless of the learning method used. Therefore, we evaluate ESZSL with three different sets of label vectors. We average 200 dimensional ICD-9 descriptor word embeddings generated by Pyysalo et al. (2013) which are pretrained on PubMed, Wikipedia, and PubMed Central (ESZSL + W2V). We lowercased descriptors and removed stop-words. We also compare with label vectors derived from our own 300 dimensional embeddings (ESZSL + W2V 2) pretrained on PubMed indexed titles and abstracts. Finally, we generate label vectors using the ICD-9 hierarchy. Specifically, let $\mathbf{Y} \in \mathbb{R}^{N \times L}$ be the document label matrix where N is the total number of documents. We factorize $\mathbf{Y}$ into two matrices $\mathbf{U} \in \mathbb{R}^{N \times 300}$ and $\mathbf{V} \in \mathbb{R}^{300 \times L}$ using graph regularized alternating least squares (GRALS) (Rao et al., 2015). Finally, we also report a baseline using a random ordering on labels, which is important for zero-shot labels — because the total number of such labels is small, the chance that the correct label is in the top k is higher compared to few-shot and frequent labels.

We compare two variants of our method: zero-shot attentive GCNN (ZAGCNN), which is the full method described in Section 3 and a simpler variant without the GCNN layers, zero-shot attentive CNN (ZACNN)[3].

Results. Table 2 shows the results for MIMIC II. Because the label set for each medical record is augmented using the ICD-9 hierarchy, we expect methods that use the hierarchy to have an advantage. Table 2 results do not rely on thresholding because we evaluate using the relative ranking of groups with similar frequencies. ACNN performs best on frequent labels. For few-shot labels, ZAGCNN outperforms ACNN by over 10% in R@10 and by 8% in R@5; compared to these R@k gains for few-shot labels, our loss on frequent labels is minimal (< 1%). We find that the word embedding derived label vectors work best for ESZSL on zero-shot labels. However, this setup is outperformed by GRALS derived label vectors on the frequent and few-shot labels. On zero-shot labels, ZAGCNN outperforms the best ESZSL variant by over 16% for both R@5 and R@10. Also, we find that the GCNN layers help both few- and zero-shot labels. Finally, similar to the setup in Xian et al. (2017), we also compute the harmonic average across all R@5 and all R@10 scores. The metric is only computed for methods that can predict zero-shot classes. We find that ZAGCNN outperforms ZACNN by 4% for R@10.

We report the MIMIC III results in Table 3. Unlike for MIMIC II, the label sets were not expanded using the ICD-9 hierarchy. Yet, we find substantial improvements on both few- and zero-shot labels using a GCNN. ZAGCNN outperforms ACNN by almost 5% and ZACNN by 1% in R@10 on few-shot classes. However, ACNN still outperforms all other methods on frequent labels, but by only 0.3% when compared with ZAGCNN. For zero-shot labels, ZAGCNN outperforms ZACNN by over 5% and outperforms the best ESZSL method by nearly 20% in R@10. We find that ZACNN slightly underperforms ZAGCNN on frequent labels with more prominent differences showing up for infrequent labels.

In Table 4 we compare the P@10, R@10, and

---

[3] We name our methods with the "zero-shot" prefix because they are primarily designed for such scenarios, although as we show later that these methods are effective for both few-shot and frequent labels

macro-F1 measures across all three groups (the union of S, F, and Z) on the MIMIC III dataset. We emphasize that the evaluation metrics are calculated over all labels and are not averages of the metrics computed independently for each group. We find that R@10 is nearly equivalent to the R@10 on the frequent group in Table 3. Furthermore, we find that ACNN outperforms ZAGCNN in P@10 by almost 4%. To compare all methods with respect to macro-F1, we simply threshold each label at 0.5. Both R@k and P@k give more weight to frequent labels, thus it is expected that ACNN outperforms ZAGCNN for frequent labels. However, we also find that ACNN outperforms our methods with respect to Macro-F1.

Given macro-F1 equally weights all labels, does the higher macro score mean ACNN performs better across infrequent labels? In Figure 3, we plot the MIMIC III R@k for the neural methods with k ranging from 1 to 100. We find as k increases, the differences between ZAGCNN and ACNN become more evident. Given Figure 3 and the scores in Table 3, it is clear that ACNN does not perform better than ZAGCNN with respect to few- and zero-shot labels. The improvement in macro-F1 for ACNN is because it performs better on frequent labels. In general, infrequent labels will have scores much less than 0.5. If we rank all labels ($S \cup F \cup Z$), we find that few-shot labels only occur among the top 16 ranked labels (average number of labels for MIMIC III) for 6% of the test documents that contain them. This suggests that many frequent irrelevant labels have higher scores than the correct few-shot label.

Why do the rankings among few- and zero-shot labels matter if they are rarely ranked above irrelevant frequent labels? If we can predict which instances contain infrequent labels (novelty detection), then we can help human coders by providing them with multiple recommendation lists — a list of frequent labels and a list of infrequent/zero-shot labels. Also, while we would ideally want a single method that performs best for both frequent and infrequent labels, currently we find that there is a trade-off between them. Hence it may be reasonable to use different methods in combination depending on label frequency.

5 Conclusion and Future Work

In this paper, we performed a fine-grained evaluation of few- and zero-shot label learning in the large-scale multi-label setting. We also introduced a neural architecture that incorporates label descriptors and the hierarchical structure of the label spaces for few- and zero-shot prediction. For these infrequent labels, previous evaluation methodologies do not provide a clear picture about what works. By evaluating power-law datasets using a generalized zero-shot learning methodology, we provide a staring point toward a better understanding. Our proposed architecture also provides large improvements on infrequent labels over state-of-the-art automatic medical coding methods.

We believe there are two important avenues for future work.

1. For medical coding, a wealth of unstructured domain expertise is available in biomedical research articles indexed by PubMed. These articles are annotated with medical subject headings (MeSH terms), which are organized in a hierarchy. Relationships between MeSH terms and ICD-9 codes are available in Unified Medical Language System (UMLS (Bodenreider, 2004)). If we can take advantage of all this structured and unstructured information via methods such as transfer learning or multi-task learning, then we may be able to predict infrequent labels better.

2. For our method to be useful for human coders, it is important to develop an accurate novelty detector. We plan to study methods for determining if an instance contains an infrequent label and if it does, how many infrequent labels it should be annotated with. In essence, this is an extension of the Meta-Labeler (Tang et al., 2009) methodology and open classification (Shu et al., 2017). If we can predict if an instance contains infrequent labels, then we can recommend few- and zero-shot labels only when necessary.

Acknowledgements

Thanks to the outstanding reviewers who provided invaluable insights to improve our manuscript. This research is supported by the U.S. National Library of Medicine through grant R21LM012274. We also gratefully acknowledge the support of the NVIDIA Corporation for its donation of the Titan X Pascal GPU used for this research.

References

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2016. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1425–1438.

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *International Conference on Machine Learning*, pages 2091–2100.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967.

Tal Baumel, Jumana Nassour-Kassis, Michael Elhadad, and Noemie Elhadad. 2018. Multi-label classification of patient notes a case study on icd code assignment. In *AAAI Joint Workshop on Health Intelligence*.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pages 730–738.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Meihao Chen, Zhuoru Lin, and Kyunghyun Cho. 2017. Graph convolutional networks for classification with a structured label space. *arXiv preprint arXiv:1710.04908*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Liwei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3.

V Jouhet, G Defossez, A Burgun, P le Beux, P Levillain, P Ingrand, and V Claveau. 2012. Automated classification of free-text pathology reports for registration of incident cases of cancer. *Methods of Information in Medicine*, 51(3):242–251.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modeling sentences. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1506–1515.

Thomas Mensink, Efstratios Gavves, and Cees GM Snoek. 2014. Costa: Co-occurrence statistics for zero-shot classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2441–2448.

Tomas Mikolov, Andrea Frome, Samy Bengio, Jonathon Shlens, Yoram Singer, Greg S Corrado, Jeffrey Dean, and Mohammad Norouzi. 2013. Zero-shot learning by convex combination of semantic embeddings. In *International Conference on Learning Representations (ICLR)*.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-scale multi-label text classification - revisiting neural networks. In *Machine Learning and Knowledge Discovery in Databases, ECML PKDD*, pages 437–452.

NCHS. 1978. International classification of diseases, ninth revision, clinical modification (icd-9-cm). *Hyattsville (MD): National Center for Health Statistics*.

Yannis Papanikolaou, Grigorios Tsoumakas, Manos Laliotis, Nikos Markantonatos, and Ioannis P Vlahavas. 2015. Auth-atypon at bioasq 3: Large-scale semantic indexing in biomedicine. In *CLEF (Working Notes)*.

Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artières, George Paliouras, Éric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Gallinari. 2015. LSHTC: A benchmark for large-scale text classification. *CoRR*, abs/1503.08581.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *World Wide Web Conference on World Wide Web*, pages 1063–1072.

Adler Perotte, Rimma Pivovarov, Karthik Natarajan, Nicole Weiskopf, Frank Wood, and Noémie Elhadad. 2013. Diagnosis code assignment: models and evaluation metrics. *Journal of the American Medical Informatics Association*, 21(2):231–237.

S. Pyysalo, F. Ginter, H. Moen, T. Salakoski, and S. Ananiadou. 2013. Distributional semantics resources for biomedical text processing. In *LBM*, pages 39–44.

Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in neural information processing systems*, pages 2107–2115.

A Rios and R Kavuluru. 2015. Analyzing the moving parts of a large-scale multi-label text classification pipeline: Experiences in indexing biomedical articles. In *IEEE International Conference on Healthcare Informatics*, volume 2015, pages 1–7.

Anthony Rios and Ramakanth Kavuluru. 2018. Emr coding with semi-parametric multi-head matching networks. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, volume 1, pages 2081–2091.

Bernardino Romera-Paredes and Philip Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161.

Timothy N Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. 2012. Statistical topic models for multi-label document classification. *Machine Learning*, 88(1-2):157–208.

Ubai Sandouk and Ke Chen. 2016. Multi-label zero-shot learning via concept embedding. *arXiv preprint arXiv:1606.00282*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of 15th European Semantic Web Conference (ESWC 2018)*.

Lei Shu, Hu Xu, and Bing Liu. 2017. Doc: Deep open classification of text documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2911–2916.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090.

Emma Strubell and Andrew McCallum. 2017. Dependency parsing with dilated iterated graph CNNs. In *Workshop on Structured Prediction for Natural Language Processing, SPNLP@EMNLP*, pages 1–6.

Lei Tang, Suju Rajan, and Vijay K Narayanan. 2009. Large scale multi-label classification via metalabeler. In *Proceedings of the 18th international conference on World wide web*, pages 211–220. ACM.

George Tsatsaronis, Michael Schroeder, Georgios Paliouras, Yannis Almirantis, Ion Androutsopoulos, Eric Gaussier, Patrick Gallinari, Thierry Artieres, Michael R Alvers, Matthias Zschunke, et al. 2012. Bioasq: A challenge on large-scale biomedical semantic indexing and question answering. In *AAAI fall symposium: Information retrieval and knowledge discovery in biomedical text*.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. 2010. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685.

Ankit Vani, Yacine Jernite, and David Sontag. 2017. Grounded recurrent neural networks. *arXiv preprint arXiv:1705.08557*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Xiaolong Wang, Yufei Ye, and Abhinav Gupta. 2018. Zero-shot recognition via semantic embeddings and knowledge graphs. *arXiv preprint arXiv:1803.08035*.

Jason Weston, Sumit Chopra, and Keith Adams. 2014. # tagspace: Semantic embeddings from hashtags. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1822–1827.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning-the good, the bad and the ugly. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3077–3086. IEEE Computer Society.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1480–1489.

Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit S. Dhillon. 2014. Large-scale multi-label learning with missing labels. In *Proceedings of the 31th International Conference on Machine Learning*, pages 593–601.

Automatic Poetry Generation with Mutual Reinforcement Learning

Xiaoyuan Yi[1,2,3], **Maosong Sun**[1,2,4]*, **Ruoyu Li**[5], **Wenhao Li**[1,2,3]
[1]Department of Computer Science and Technology, Tsinghua University
[2]Institute for Artificial Intelligence, Tsinghua University
[3]State Key Lab on Intelligent Technology and Systems, Tsinghua University
[4]Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University
[5]6ESTATES PTE LTD, Singapore
`{yi-xy16, liwh16}@mails.tsinghua.edu.cn`,
`sms@mail.tsinghua.edu.cn`, `liruoyu@6estates.com`

Abstract

Poetry is one of the most beautiful forms of human language art. As a crucial step towards computer creativity, automatic poetry generation has drawn researchers' attention for decades. In recent years, some neural models have made remarkable progress in this task. However, they are all based on maximum likelihood estimation, which only learns common patterns of the corpus and results in loss-evaluation mismatch. Human experts evaluate poetry in terms of some specific criteria, instead of word-level likelihood. To handle this problem, we directly model the criteria and use them as explicit rewards to guide gradient update by reinforcement learning, so as to motivate the model to pursue higher scores. Besides, inspired by writing theories, we propose a novel mutual reinforcement learning schema. We simultaneously train two learners (generators) which learn not only from the teacher (rewarder) but also from each other to further improve performance. We experiment on Chinese poetry. Based on a strong basic model, our method achieves better results and outperforms the current state-of-the-art method.

1 Introduction

Language is one of the most important forms of human intelligence and poetry is a concise and graceful art of human language. Across different countries, nationalities and cultures, poetry is always popular, having far-reaching influence on the development of human society.

In this work, we concentrate on automatic poetry generation. Besides the long-term goal of building artificial intelligence, research on this task could become the auxiliary tool to better analyse poetry and understand the internal mechanism of human writing. In addition, these generation systems are also helpful for electronic entertainments and literary education.

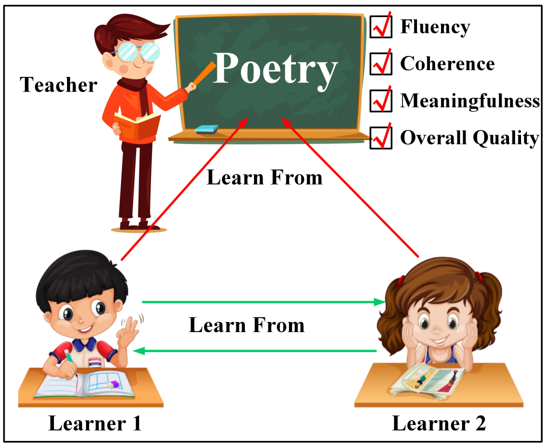

Figure 1: An artistic illustration of our mutual reinforcement learning method.

In recent years, neural networks have proven to be powerful on poetry generation. Some neural models are proposed and achieve significant improvement. However, existing models are all based on maximum likelihood estimation (MLE), which brings two substantial problems. First, MLE-based models tend to remember common patterns of the poetry corpus (Zhang et al., 2017), such as high-frequency bigrams and stop words, losing some diversity and innovation for generated poetry. Moreover, based on word-level likelihood, two kinds of loss-evaluation mismatch (Wiseman and Rush, 2016) arise. One is *evaluation granularity mismatch*. When evaluating, human experts usually focus on sequence level (a poem line) or discourse level (a whole poem), while MLE optimizes word-level loss, which fails to hold a wider view of generated poems. The other is *criteria mismatch*. Instead of the likelihood, humans usually evaluate poetry in terms of some criteria. In this work we focus on the main four criteria

---

[*] Corresponding author: sms@mail.tsinghua.edu.cn.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3143–3153
Brussels, Belgium, October 31 - November 4, 2018.

(Manurung, 2003; Zhang and Lapata, 2014; Yan, 2016; Yi et al., 2017): **fluency** (are the lines fluent and well-formed?), **coherence** (is the poem as a whole coherent in meaning and theme?), **meaningfulness** (does the poem convey some certain messages?), **overall quality** (the reader's general impression on the poem). This mismatch may make the model lean towards optimizing easier criteria, e.g., fluency, and ignore other ones.

To tackle these problems, we directly model the four aforementioned human evaluation criteria and use them as explicit rewards to guide gradient update by reinforcement learning. This is a criterion-driven training process, which motivates the model to generate poems with higher scores on these criteria. Besides, in writing theories, writing requires observing other learners (Bandura, 2001). It is also shown that writing is supported as an activity in which writers will learn from more experienced writers, such as other students, teachers, or authors (Prior, 2006). Therefore it is necessary to equip generators with the ability of mutual learning and communication. Inspired by this, we propose a novel mutual reinforcement learning schema (Figure 1), where we simultaneously train two *learners* (generators). During the training process, one learner will learn not only from the *teacher* (rewarder) but also from the other. We will show this mutual learning-teaching process leads to better results.

In summary, our contributions are as follows:

- To the best of our knowledge, for the sake of tackling the loss-evaluation mismatch problem in poetry generation, we first utilize reinforcement learning to model and optimize human evaluation criteria.
- We propose a novel mutual reinforcement learning schema to further improve performance, which is **transparent** to model architectures. One can apply it to any poetry generation model.
- We experiment on Chinese quatrains. Both automatic and human evaluation results show that our method outperforms a strong basic method and the state-of-the-art model.

2 Related Work

As a desirable entry point of automatic analysing, understanding and generating literary text, the research on poetry generation has lasted for decades. In recent twenty years, the models can be categorized into two main paradigms.

The first one is based on statistical machine learning methods. Genetic algorithms (Manurung, 2003; Levy, 2001), Statistical Machine Translation (SMT) approaches (He et al., 2012; Jiang and Zhou, 2008) and Automatic Summarization approaches (Yan et al., 2013) are all adopted to generate poetry.

More recently, the second paradigm, neural network, has shown great advantages in this task, compared to statistical models. Recurrent Neural Network (RNN) is first used to generate Chinese quatrains by (Zhang and Lapata, 2014). To improve fluency and coherence, Zhang's model needs to be interpolated with extra SMT features as shown in their paper. Focusing on coherence, some works (Yi et al., 2017; Wang et al., 2016a) use sequence-to-sequence model with attention mechanism (Bahdanau et al., 2015) to generate poetry. Wang et al. (2016b) design a special Planning schema, which plans some sub-keywords in advance by a language model and then generates each line with the planned sub-keyword to improve coherence. Pursuing better overall quality, Yan (2016) proposes an iterative polishing schema to generate Chinese poetry, which refines the poem generated in one pass for several times. Aiming at enhancing meaningfulness, Ghazvininejad et al. (2016) extend user keywords to incorporate richer semantic information. Zhang et al. (2017) combine a neural memory, which saves hundreds of human-authored poems, with a sequence-to-sequence model to improve innovation of generated poems and achieve style transfer.

These neural structures have made some progress and improved different aspects of generated poetry. Nevertheless, as discussed in Section 1, the two essential problems, lack of diversity and loss-evaluation mismatch, are still challenging resulting from MLE. Compared to further adjusting model structures, we believe a better solution is to design more reasonable optimization objectives.

Deep Reinforcement Learning (DRL) first shows its magic power in automatic game playing, such as Atari electronic games (Mnih et al., 2013) and the game of Go (Silver et al., 2016). Soon, DRL is used to playing text games (Narasimhan et al., 2015; He et al., 2016) and then applied to dialogue generation (Li et al., 2016b).

From the perspective of poetry education, the teacher will judge student-created poems in terms of some specific criteria and guide the student to cover the shortage, which naturally accords with DRL process. Therefore we take advantage of DRL. We design four automatic rewarders for the criteria, which act as the teacher. Furthermore, we train two generators and make them learn from each other, which imitates the mutual learning of students, as a step towards multi-agent DRL in literary text generation.

3 Methods

3.1 Basic Generation Model

We apply our method to a basic poetry generation model, which is pre-trained with MLE. Therefore, we first formalize our task and introduce this model.

The inputs are user topics specified by K keywords, $\mathcal{W} = \{w_k\}_{k=1}^{K}$. The output is a poem consisting of n lines, $\mathcal{P} = L_1, L_2, \cdots, L_n$. Since we take the line-by-line generation process, the task can be converted to the generation of an i-th line given previous i-1 lines $L_{1:i-1}$ and $\mathcal{W}$.

We use GRU-based (Cho et al., 2014) sequence-to-sequence model. $\overrightarrow{h}_t$, $\overleftarrow{h}_t$ and s_t represent the forward encoder, backward encoder and decoder hidden states respectively. For each topic word $w_k = c_1, c_2, \cdots, c_{T_k}$, we feed characters into the encoder and get the keyword representation $v_k = [\overrightarrow{h}_{T_k}; \overleftarrow{h}_{T_k}]$, where [;] means concatenation. Then we get the topic representation by[1]:

$$o = f(\frac{1}{K}\sum_{t=1}^{K} v_k), \tag{1}$$

where f defines a non-linear layer.

Denote the generated i-th line in decoder, $Y = (y_1 y_2 \ldots y_{T_i})$. $e(y_t)$ is the word embedding of y_t. The probability distribution of each y_t to be generated in L_i is calculated by:

$$s_t = GRU(s_{t-1}, [e(y_{t-1}); o; g_{i-1}]), \tag{2}$$

$$P(y_t|y_{1:t-1}, L_{1:i-1}, \mathcal{W}) = softmax(W s_t), \tag{3}$$

where W is the projection parameter. g_{i-1} is a global history vector, which records what has been generated so far and provides global-level information for the model. Once L_i is generated, it is updated by a convolutional layer:

$$a_t = f([s_t; \cdots; s_{t+d-1}]), \tag{4}$$

$$g_i = f(g_{i-1}, \sum_t a_t), g_0 = \mathbf{0}, \tag{5}$$

where $\mathbf{0}$ is a vector with all 0-s and d is convolution window size. Then the basic model is pre-trained by minimizing standard MLE loss:

$$\mathcal{L}_{MLE}(\theta) = -\sum_{m=1}^{M} logP(\mathcal{P}^m|\mathcal{W}^m; \theta), \tag{6}$$

where M is data size and θ is the parameter set to be trained.

This basic model is a modified version of (Yan, 2016). The main differences are that we replace vanilla RNN with GRU unit, use convolution to calculate the line representation rather than directly use the last decoder hidden state, and we remove the polishing schema to better obverse the influence of DRL itself. We select this model as our basic framework since it achieves satisfactory performance and the author has done thorough comparisons with other models, such as (Yan et al., 2013) and (Zhang and Lapata, 2014).

3.2 Single-Learner Reinforcement Learning

Before presenting the single-learner version of our method (abbreviated as **SRL**), we first design corresponding automatic rewarders for the four human evaluation criteria.

Fluency Rewarder. We use a neural language model to measure fluency. Given a poem line L_i, higher probability $P_{lm}(L_i)$ indicates the line is more likely to exist in the corpus and thus may be more fluent and well-formed. However, it's inadvisable to directly use $P_{lm}(L_i)$ as the reward, since over high probability may damage diversity and innovation. We expect moderate probabilities which fall into a reasonable range, neither too high nor too low. Therefore, we define the fluency reward of a poem $\mathcal{P}$ as:

$$r(L_i) = max(|P_{lm}(L_i) - \mu| - \delta_1 * \sigma, 0), \tag{7}$$

$$R_1(\mathcal{P}) = \frac{1}{n}\sum_{i=1}^{n} exp(-r(L_i)), \tag{8}$$

where μ and σ are the mean value and standard deviation of P_{lm} calculated over all training sets. δ_1 is a hyper-parameter to control the range.

Coherence Rewarder. For poetry, good coherence means each line L_i should be coherent with

---

[1] For brevity, we omit biases in all equations.

previous lines in a poem. We use Mutual Information (MI) to measure the coherence of L_i and $L_{1:i-1}$. As shown in (Li et al., 2016a), MI of two sentences, S_1 and S_2, can be calculated by:

$$MI(S_1, S_2) = logP(S_2|S_1) - \lambda logP(S_2), \quad (9)$$

where λ is used to regulate the weight of generic sentences. Based on this, we calculate the coherence reward as:

$$\begin{aligned} MI(L_{1:i-1}, L_i) = & logP_{seq2seq}(L_i|L_{1:i-1}) \\ & - \lambda logP_{lm}(L_i), \end{aligned} \quad (10)$$

$$R_2(\mathcal{P}) = \frac{1}{n-1}\sum_{i=2}^{n} MI(L_{1:i-1}, L_i), \quad (11)$$

where $P_{seq2seq}$ is a GRU-based sequence-to-sequence model, which takes the concatenation of previous i-1 lines as input, and predicts L_i. A better choice is to use a dynamic λ instead of a static one. Here we directly set $\lambda = exp(-r(L_i)) + 1$, which gives smaller weights to lines with extreme language model probabilities.

Meaningfulness Rewarder. In dialogue generation task, neural models are prone to generate generic sentences such as "I don't know" (Li et al., 2016a; Serban et al., 2016). We observed similar issues in poetry generation. The basic model tends to generate some common and meaningless words, such as *bu zhi* (don't know), *he chu* (where), and *wu ren* (no one). It's quite intractable to quantify the meaningfulness of a whole poem, but we find that TF-IDF values of human-authored poems are significantly higher than values of generated ones (Figure 2). Consequently, we utilize TF-IDF to motivate the model to generate more meaningful words. This is a simple and rough attempt, but it makes generated poems more "meaningful" from the readers perspective.

Direct use of TF-IDF leads to serious out-of-vocabulary (OOV) problem and high variance, because we need to sample poems during the training process of DRL, which causes many OOV words. Therefore we use another neural network to smooth TF-IDF values. In detail, we have:

$$R_3(\mathcal{P}) = \frac{1}{n}\sum_{i=1}^{n} F(L_i), \quad (12)$$

where $F(L_i)$ is a neural network which takes a line as input and predicts its estimated TF-IDF value. For each line in training sets, we calculate standard TF-IDF values of all words and use the average as the line TF-IDF value. Then we use them to train $F(L_i)$ with Huber loss.

Overall Quality Rewarder. The three kinds of rewards above are all based on line-level. In fact, human experts will also focus on discourse-level to judge the overall quality of a poem, ignoring some minor defects. We train a neural classifier to classify a given poem (in terms of the concatenation of all lines) into three classes: computer-generated poetry (class 1), ordinary human-authored poetry (class 2) and masterpiece (class 3). Then we get the reward by:

$$R_4(\mathcal{P}) = \sum_{k=1}^{3} P_{cl}(k|\mathcal{P}) * k. \quad (13)$$

This classifier should be as reliable as possible. Due to the limited amount of masterpieces, normal classifiers don't work well. Therefore we use an adversarial training based classifier (Miyato et al., 2017), which achieves F-1 0.96, 0.73, 0.76 for the three classes respectively on the validation set.

Based on these rewarders, the total reward is:

$$R(\mathcal{P}) = \sum_{j=1}^{4} \alpha_j * \tilde{R}_j(\mathcal{P}), \quad (14)$$

where α_j is the weight and the symbol ˜ means the four rewards are re-scaled to the same magnitude. As (Gulcehre et al., 2018), we reduce the variance by:

$$R^{'}(\mathcal{P}) = \frac{R(\mathcal{P}) - b_u}{\sqrt{\sigma_u^2 + \epsilon}} - B(\mathcal{P}), \quad (15)$$

where b_u and σ_u are running average and standard deviation of R respectively. $B(\mathcal{P})$ is a neural network trained with Huber loss, which takes a poem as input and predicts its estimated reward.

DRL Process. For brevity, we use $P_g(\cdot|\mathcal{W};\theta)$ to represent a basic generator and use REINFORCE algorithm (Williams, 1992) to optimize the model, which minimizes:

$$\mathcal{L}_{DRL}(\theta) = -\sum_{m=1}^{M} \mathbb{E}_{\mathcal{P}\sim P_g(\cdot|\mathcal{W}^m;\theta)}(R^{'}(\mathcal{P})). \quad (16)$$

Training with solely Eq.(16) is unstable. Lacking of original MLE supervisory signals, the

Algorithm 1 Global Mutual Learning

```
Set history reward lists $V_1$ and $V_2$ empty;
for number of iterations do
    Sample batch $(\mathcal{W}^m, \mathcal{P}_g^m)$ from training data set;
    for each $\mathcal{W}^m$ do
        Sample $\mathcal{P}_1^m \sim P_g(\cdot|\mathcal{W}^m; \theta_1)$;
        Sample $\mathcal{P}_2^m \sim P_g(\cdot|\mathcal{W}^m; \theta_2)$;
        Add $R(\mathcal{P}_1^m)$ to $V_1$, $R(\mathcal{P}_2^m)$ to $V_2$
    end for
    Set $\mathcal{L}_M(\theta_1)$=$\mathcal{L}(\theta_1)$, $\mathcal{L}_M(\theta_2)$=$\mathcal{L}(\theta_2)$;
    if mean value $\overline{V_2} > \overline{V_1} * (1 + \delta_3)$ then
        $\mathcal{L}_M(\theta_1)$=$\mathcal{L}(\theta_1) + KL(P_g(\theta_2)||P_g(\theta_1))$;
    else if $\overline{V_1} > \overline{V_2} * (1 + \delta_3)$ then
        $\mathcal{L}_M(\theta_2)$=$\mathcal{L}(\theta_2) + KL(P_g(\theta_1)||P_g(\theta_2))$;
    end if
    Update $\theta_1$ with $\mathcal{L}_M(\theta_1)$, $\theta_2$ with $\mathcal{L}_M(\theta_2)$;
end for
```

model is easy to get lost and totally ignore the corresponding topics specified by $\mathcal{W}$, leading to explosive increase of MLE loss. We use two steps to alleviate this issue. The first one is the Teacher Forcing (Li et al., 2017). For each $\mathcal{W}$, we estimate $\mathbb{E}(R^{'}(\mathcal{P}))$ by n_s sampled poems, as well as the ground-truth $\mathcal{P}_g$ whose reward is set to $max(R^{'}(\mathcal{P}_g), 0)$. The second step is to combine MLE loss and DRL loss as:

$$\mathcal{L}(\theta) = (1 - \beta) * \mathcal{L}_{MLE}(\theta) + \beta * \tilde{\mathcal{L}}_{DRL}(\theta), \tag{17}$$

where ˜ means the DRL loss is re-scaled to the same magnitude with MLE loss. Ultimately, we use Eq.(17) to fine-tune the basic model.

3.3 Mutual Reinforcement Learning

As discussed in Section 1 & 2, to further improve the performance, we mimic the mutual writing learning activity by simultaneously training two generators defined as $P_g(\theta_1)$ and $P_g(\theta_2)$. The two learners (generators) learns not only from the teacher (rewarders) but also from each other.

From the perspective of machine learning, one generator may not explore the policy space sufficiently and thus is easy to get stuck in the local minima. Two generators can explore along different directions. Once one generator finds a better path (higher reward), it can communicate with the other and lead it towards this path. This process could also be considered as the ensemble of different generators during the training phase.

Models	$\tilde{R}_1$	$\tilde{R}_2$	$\tilde{R}_3$	$\tilde{R}_4$	R
Base	0.156	0.214	0.509	0.351	0.282
Mem	0.192	0.257	0.467	0.383	0.308
MRL	**0.207**	**0.268**	**0.613**	**0.494**	**0.369**
GT	0.582	0.609	0.625	0.759	0.649
SRL	0.169	0.228	0.563	0.432	0.321
LMRL	0.187	0.246	0.602	0.467	0.348
GMRL	0.199	0.262	0.606	0.480	0.360
MRL	**0.207**	**0.268**	**0.613**	**0.494**	**0.369**

Table 1: Automatic rewards of different models and strategies. $\tilde{R}_1$: fluency, $\tilde{R}_2$: coherence, $\tilde{R}_3$: meaningfulness, $\tilde{R}_4$: overall quality, R: weighted-average reward. LMRL: local MRL, GMRL: global MRL.

We implement the Mutual Reinforcement Learning (abbreviated as **MRL**) by two methods.

Local MRL. The first one is a simple instance-based method. For the same input, suppose $\mathcal{P}_1$, $\mathcal{P}_2$ are generated by $P_g(\theta_1)$ and $P_g(\theta_2)$ respectively. If $R(\mathcal{P}_1) > R(\mathcal{P}_2) * (1 + \delta_2)$ and $\tilde{R}_j(\mathcal{P}_1) > \tilde{R}_j(\mathcal{P}_2)$ for all j, then $P_g(\theta_2)$ uses $\mathcal{P}_1$ instead of $\mathcal{P}_2$ to update itself in Eq.(16) and vice versa. That is, if a learner creates a significantly better poem, then the other learner will learn it. This process gives a generator more high-reward instances and allows it to explore larger space along a more proper direction so as to escape from the local minima.

Global MRL. During the training process, we need to sample poems from the generator, and hence local MRL may cause high variance. Instead of an instance, mutual learning can also be applied on the distribution level. We can pull the distribution of a generator towards that of the other by minimizing KL divergence of them. We detail this method in algorithm 1. The inner thought is that if learner 1 is generally better than learner 2, that is, during the creating history, learner 1 achieves higher average rewards, then learner 2 should directly learn from learner 1, rather than learn the poem itself. This process allows the generator to learn from long-period history and focus on a higher level.

In practice, we combine these two methods by simultaneously communicating high-reward samples and using KL loss, which leads to the best testing rewards (Table 1).

4 Experiments

4.1 Data and Setups

Our corpus consists of three sets: 117,392 Chinese quatrains (CQ), 10,000 Chinese regulated verses (CRV) and 10,000 Chinese iambics (CI). As men-

Models	Bigram Ratio	Jaccard
Base	0.126	0.214
Mem	**0.184**	0.183
MRL	0.181	**0.066**
GT	0.218	0.006
SRL	0.133	0.146
LMRL	0.178	0.085
GMRL	**0.186**	0.075
MRL	0.181	**0.066**

Table 2: Automatic evaluation results of diversity and innovation. The Jaccard values are multiplied by 10 for clearer observation. We expect higher bigram ratio and smaller Jaccard values.

tioned, we experiment on the generation of quatrain which is the most popular genre of Chinese poetry and accounts for the largest part of our corpus. From the three sets, we randomly select 10% for validation. From CQ, we select another 10% for testing. The rest are used for training.

For our model and baseline models, we run TextRank (Mihalcea and Tarau, 2004) on all training sets and then extract four keywords from each quatrain. Then we build four < keyword(s), poem > pairs for each quatrain using 1 to 4 keywords respectively, so as to enable the model to cope with different numbers of keywords.

For the models and rewarders, the sizes of word embedding and hidden state are 256 and 512 respectively. History vector size is 512 and convolution window size $d = 3$. The word embedding is initialized with pre-trained word2vec vectors. We use tanh as the activation function. For other more configurations of the basic model, we directly follow (Yan, 2016).

P_{lm} and $P_{seq2seq}$ are trained with the three sets. We train $F(L_i)$ and $B(\mathcal{P})$ with the CQ, CRV and 120,000 generated poems. There are 9,465 masterpieces in CQ. We use these poems, together with 10,000 generated poems and 10,000 ordinary human-authored poems to train the classifier P_{cl}. For training rewarders, half of the generated poems are sampled and the other half are generated with beam search (beam size 20). For testing, all models generate poems with beam search.

We use Adam (Kingma and Ba, 2015) with shuffled mini-batches. The batch size is 64 for MLE and 32 for DRL. For DRL, we random select batches to fine-tune the basic model. We set $\delta_1 = 0.5$, $\delta_2 = 0.1$, $\delta_3 = 0.001$, $\alpha_1 = 0.25$, $\alpha_2 = 0.31$, $\alpha_3 = 0.14$, $\alpha_4 = 0.30$, $n_s = 4$, and $\beta = 0.7$.

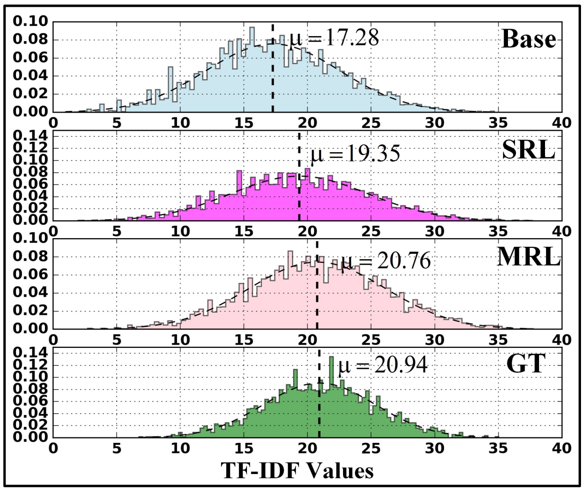

Figure 2: TF-IDF distributions of poems generated by different models. We show real TF-IDF, instead of the estimated $\tilde{R}_3$.

A key point for MRL is to give the two pretrained generators some diversity, which can be achieved by using different model structures or parameters. Here we simply initialize the generators differently and train one of them for more epoches.

4.2 Models for Comparisons

We compare **MRL**[2] (our model, with both local and global mutual learning), **GT** (ground-truth, namely human-authored poems), **Base** (the basic model described in Section 3.1) and **Mem** (Zhang et al., 2017). The Mem model is the current state-of-the-art model for Chinese quatrain generation, which also achieves the best innovation so far.

4.3 Automatic Evaluation

Some previous models (He et al., 2012; Zhang and Lapata, 2014; Yan, 2016) adopt BLEU and perplexity as automatic evaluation metrics. Nevertheless, as discussed in Section 1, word-level likelihood or n-gram matching will greatly diverge from human evaluation manner. Therefore we dispense with them and automatically evaluate generated poems as follows:

Rewarder Scores. The four rewarder scores are objective and model-irrelevant metrics which approximate corresponding human criteria. They

---

[2] Due to length limit, we only display the better of the two simultaneously trained generators. Our source code will be available at https://github.com/XiaoyuanYi/MRLPoetry.

Models	Fluency	Coherence	Meaning	Overall Quality
Base	3.28	2.77	2.63	2.58
Mem	3.23	2.88	2.68	2.68
MRL	**4.05****	**3.81****	**3.68****	**3.60****
GT	4.14	4.11^{++}	4.16^{++}	3.97^{++}

Table 3: Human evaluation results. Diacritic ** ($p < 0.01$) indicates MRL significantly outperforms baselines; ++ ($p < 0.01$) indicates GT is significantly better than all models.

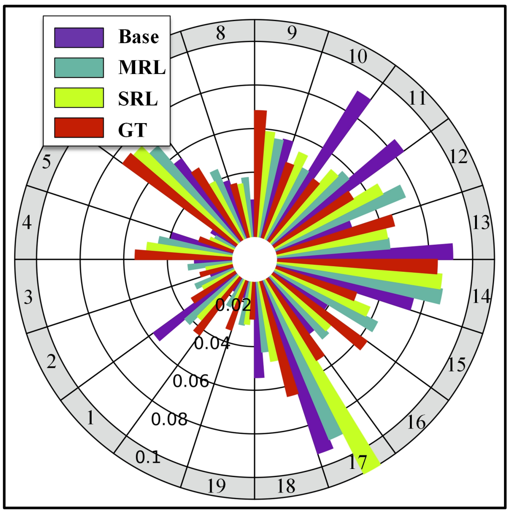

Figure 3: Topic Distributions of different models.

can reflect poetry quality to some extent. As shown in Table 1, on each criterion, GT gets much higher rewards than all these models. Compared to Base, MRL gets closer to GT and achieves 31% improvement on the weighted average reward. Mem outperforms Base on the criteria except for meaningfulness ($\tilde{R}_3$). This is mainly because Mem generates more distinct words (Table 2), but these words tend to concentrate on the high-frequency area, resulting in unsatisfactory TF-IDF reward. We also test different strategies of MRL. With naive single-learner RL, the improvement is limited, only 14%. With mutual RL, the improvement increases to 27%. Combining local MRL and global MRL leads to another 4% improvement. The results demonstrate our explicit optimization (RL) is more effective than the implicit ones and MRL gets higher scores than SRL.

Diversity and Innovation. Poetry is a kind of literature text with high requirements on diversity and innovation. Users don't expect the machine to always generate monotonous poems. We evaluate innovation of generated poems by distinct bigram ratio as (Li et al., 2016b). More novel generated bigrams can somewhat reflect higher innovation. The diversity is measured by bigram-based average Jaccard similarity of each two generated poems. Intuitively, a basic requirement for innovation is that, with different inputs, the generated poems should be different from each other.

As shown in Table 2, Mem gets the highest bigram ratio, close to GT, benefiting from its specially designed structure for innovation. Our MRL achieves 43% improvement over Base, comparable to Mem. We will show later this satisfactory performance may lie in the incorporation of TF-IDF (Figure 2). On Jaccard, MRL gets the best result due to the utilization of MI. MI brings richer context-related information which can enhance diversity as shown in (Li et al., 2016a). In fact, human-authored poems often contain strong diversity of personal emotion and experience. Therefore, despite prominent improvement, there is still a large gap between MRL and GT.

TF-IDF Distribution. As mentioned, the basic model tends to generate common and meaningless words. Consequently, we use TF-IDF as one of the rewards. Figure 2 shows the TF-IDF distributions. As we can see, Base generates poems with lower TF-IDF compared to GT, while MRL pulls the distribution towards that of GT, making the model generate more meaningful words and hence benefiting innovation and diversity.

Topic Distribution. We run LDA (Blei et al., 2003) with 20 topics on the whole corpus and then inference the topic of each generated poem. Figure 3 gives the topic distributions. Poems generated by Base center in a few topics, which again demonstrates the claim: MLE-based models tend to remember the common patterns. In contrast, human-authored poems spread on more topics. After fine-tuning by our MRL method, the topic distribution shows better diversity and balance.

4.4 Human Evaluation

From the testing set, we randomly select 80 sets of keywords to generate poems with these mod-

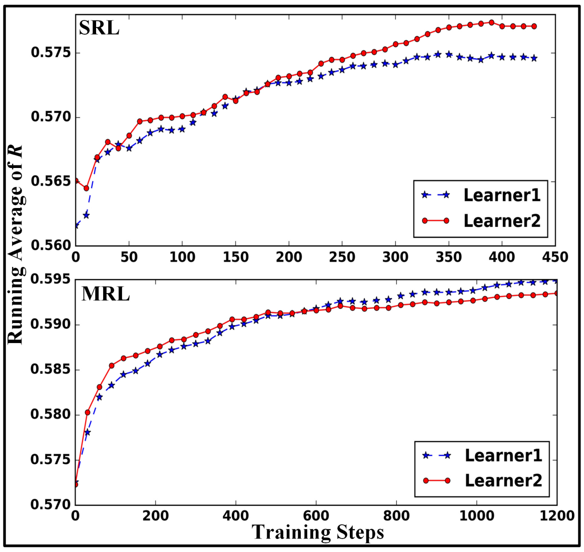

Figure 4: The learning curves of SRL and MRL. Learner (generator) 2 is pre-trained for more epoches to allow some diversity.

els. For GT, we select poems containing the given words. Therefore, we obtain 320 quatrains (80*4). We invite 12 experts on Chinese poetry to evaluate these poems in terms of the four criteria: fluency, coherence, meaningfulness and overall quality and each needs to be scored in a 5-point scale ranging from 1 to 5. Since it's tiring to evaluate all poems for one person, we randomly divide the 12 experts into three groups. Each group evaluates the randomly shuffled 320 poems (80 for each expert). Then for each model, each poem, we get 3 scores on each criterion and we use the average to alleviate individual preference.

Table 3 gives human evaluation results. MRL achieves better results than the other two models. Since fluency is quite easy to be optimized, our method gets close to human-authored poems on Fluency. The biggest gap between MRL and GT lies on Meaning. It's a complex criterion involving the use of words, topic, emotion expression and so on. The utilization of TF-IDF does ameliorate the use of words on diversity and innovation, hence improving Meaningfulness to some extent, but there are still lots to do.

4.5 Further Analyses and Discussions

In this section we give more discussions.

Learning Curve. We show the learning curves of SRL and MRL in Figure 4. As we can see, for SRL, the adequately pre-trained generator 2 always gets higher rewards than the other one during the DRL training process. With the increase of training steps, the gap between their rewards gets larger. After several hundred steps, rewards of the two generators converge.

For MRL, generator 2 gets higher rewards at the beginning, but it is exceeded by generator 1 since generator 1 learns from it and keeps chasing. Finally, the two generators converge to higher rewards compared to SRL.

Case Study. We show some generated poems in Figure 5. The Base model generates two words, ‘sunset’ and ‘moon’ in poem (1), which appear together and thus cause the conflict of time. The word ‘fishing jetty’ is confusing without any necessary explanation in the context. In contrast, poem (2) describes a clearer scene and expresses some emotion: a lonely man takes a boat from morning till night and then falls asleep solitarily.

In poem (3), Mem generates some meaningful words, such as ‘phoenix tree’, ‘wild goose’ and ‘friend’. However, there isn't any clue to link them together, resulting in poor coherence. On the contrary, things in poem (4) are tightly connected. For example, ‘moonlight’ is related to ‘night’; ‘rain’, ‘frost’ and ‘dew’ are connected with ‘cold’.

Poem (5) expresses almost nothing. The first two lines seem to talk about the change of time. But the last two lines are almost unrelated to ‘time change’. Poem (6) talks about an old poet, with the description of cheap wine, poem and dream, expressing something about life and time. However, the human-authored poem (7) does much better. It seems to describe a mosquito, but in fact, it's a metaphor of the author himself.

5 Conclusion and Future Work

In this work, we address two substantial problems in automatic poetry generation: lack of diversity, and loss-evaluation mismatch, which are caused by MLE-based neural models. To this end, we directly model the four widely used human evaluation criteria and design corresponding automatic rewarders. We use these explicit rewards to guide gradient update by reinforcement learning. Furthermore, inspired by writing theories, we propose a novel mutual learning schema to further improve the performance. Mimicking the poetry learning activity, we simultaneously train two generators, which will not only be taught by the rewarders but also learn from each other. Experi-

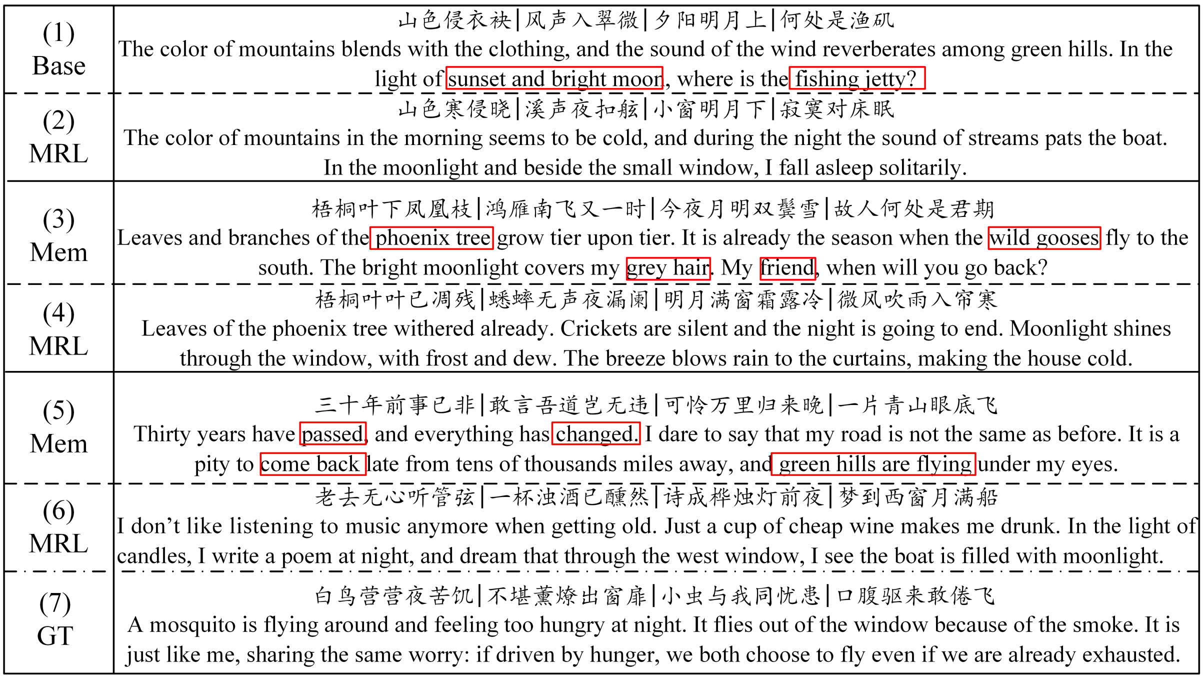

(1) Base	山色侵衣袂|风声入翠微|夕阳明月上|何处是渔矶 The color of mountains blends with the clothing, and the sound of the wind reverberates among green hills. In the light of sunset and bright moon, where is the fishing jetty?
(2) MRL	山色寒侵晓|溪声夜扣舷|小窗明月下|寂寞对床眠 The color of mountains in the morning seems to be cold, and during the night the sound of streams pats the boat. In the moonlight and beside the small window, I fall asleep solitarily.
(3) Mem	梧桐叶下凤凰枝|鸿雁南飞又一时|今夜月明双鬓雪|故人何处是君期 Leaves and branches of the phoenix tree grow tier upon tier. It is already the season when the wild gooses fly to the south. The bright moonlight covers my grey hair. My friend, when will you go back?
(4) MRL	梧桐叶叶已凋残|蟋蟀无声夜漏阑|明月满窗霜露冷|微风吹雨入帘寒 Leaves of the phoenix tree withered already. Crickets are silent and the night is going to end. Moonlight shines through the window, with frost and dew. The breeze blows rain to the curtains, making the house cold.
(5) Mem	三十年前事已非|敢言吾道岂无违|可怜万里归来晚|一片青山眼底飞 Thirty years have passed, and everything has changed. I dare to say that my road is not the same as before. It is a pity to come back late from tens of thousands miles away, and green hills are flying under my eyes.
(6) MRL	老去无心听管弦|一杯浊酒已醺然|诗成桦烛灯前夜|梦到西窗月满船 I don't like listening to music anymore when getting old. Just a cup of cheap wine makes me drunk. In the light of candles, I write a poem at night, and dream that through the west window, I see the boat is filled with moonlight.
(7) GT	白鸟营营夜苦饥|不堪薰燎出窗扉|小虫与我同忧患|口腹驱来敢倦飞 A mosquito is flying around and feeling too hungry at night. It flies out of the window because of the smoke. It is just like me, sharing the same worry: if driven by hunger, we both choose to fly even if we are already exhausted.

Figure 5: Sampled poems generated by different models. Poems between two solid lines are generated with the same input keywords. Some defects are shown in red boxes.

mental results show our method achieves significant improvement both on automatic rewards and human evaluation scores, outperforming the current state-of-the-art model[3].

There are still lots to do. Can we better model the meaningfulness of a whole poem? Can we quantify some other intractable criteria, e.g, poeticness? Besides, we only tried two learners in this work. Would the collaboration of more learners lead to better results? How to design the methods of communication among many generators? We will explore these questions in the future.

Acknowledgments

We would like to thank Cheng Yang, Jiannan Liang, Zhipeng Guo, Huimin Chen and anonymous reviewers for their insightful comments. This research is funded by the National 973 project (No. 2014CB340501). It is also partially supported by the NExT++ project, the National Research Foundation, Prime Ministers Office, Singapore under its IRC@Singapore Funding Initiative.

References

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*, San Diego, CA.

Albert Bandura. 2001. Social cognitive theory: An agentic perspective. *Annual Review of Psychology*, 52(1):1–26.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *Machine Learning Research*, (3):993–1022.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, Austin, Texas. Association for Computational Linguistics.

Caglar Gulcehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. 2018. Dynamic neural turing machine with continuous and discrete addressing schemes. *Neural Computation*, 30(4):857–884.

---

[3] Our method will be incorporated into **Jiuge**, the THUNLP online poetry generation system, `https://jiuge.thunlp.cn`.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630, Berlin, Germany. Association for Computational Linguistics.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating chinese classical poems with statistical machine translation models. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1650–1656, Toronto, Canada.

Long Jiang and Ming Zhou. 2008. Generating chinese couplets using a statistical mt approach. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 377–384, Manchester, UK.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 2015 International Conference on Learning Representations*, San Diego, CA.

Robert P. Levy. 2001. A computational model of poetic creativity with neural network as measure of adaptive fitness. In *Proceedings of the ICCBR-01 Workshop on Creative Systems*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169, Copenhagen, Denmark. Association for Computational Linguistics.

Hisar Maruli Manurung. 2003. *An Evolutionary Algorithm Approach to Poetry Generation*. Ph.D. thesis, University of Edinburgh.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *Proceedings of the 2017 International Conference on Learning Representations*, Toulon, France.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *Computer Science*.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *Computer Science*, 40(4):1–5.

Paul Prior. 2006. *Handbook of Writing Research*, chapter A Sociocultural Theory of Writing. Guilford Press.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3784, Phoenix, Arizona.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016a. Chinese song iambics generation with neural attention-based model. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2943–2949, New York, USA.

Zhe Wang, Wei He, Hua Wu nad Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016b. Chinese poetry generation with planning based neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics:Technical Papers*, pages 1051–1060, Osaka, Japan.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Rui Yan. 2016. I, poet:automatic poetry composition through recurrent neural networks with iterative polishing schema. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2238–2244, New York, USA.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 2013. I, poet:automatic chinese poetry composition through a generative summarization framework under constrained optimization. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 2197–2203, Beijing, China.

Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. 2017. Generating chinese classical poems with rnn encoder-decoder. In *Proceedings of the Sixteenth Chinese Computational Linguistics*, pages 211–223, Nanjing, China.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative chinese poetry generation using neural memory. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1364–1373. Association for Computational Linguistics.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Doha, Qatar. Association for Computational Linguistics.

Variational Autoregressive Decoder for Neural Response Generation

Jiachen Du[1], Wenjie Li[2], Yulan He[3], Lidong Bing[4], Ruifeng Xu[1]*, Xuan Wang[1]
[1]Department of Computer Science, Harbin Institute of Technology (Shenzhen), China
[2] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[3] Department of Computer Science, University of Warwick, United Kingdom
[4] Tencent AI Lab, China
dujiachen@stmail.hitsz.edu.cn, cswjli@comp.polyu.edu.hk
y.he@cantab.net, lyndonbing@tencent.com
xuruifeng@hit.edu.cn, wangxuan@cs.hitsz.edu.cn

Abstract

Combining the virtues of probability graphic models and neural networks, Conditional Variational Auto-encoder (CVAE) has shown promising performance in many applications such as response generation. However, existing CVAE-based models often generate responses from a single latent variable which may not be sufficient to model high variability in responses. To solve this problem, we propose a novel model that sequentially introduces a series of latent variables to condition the generation of each word in the response sequence. In addition, the approximate posteriors of these latent variables are augmented with a backward Recurrent Neural Network (RNN), which allows the latent variables to capture long-term dependencies of future tokens in generation. To facilitate training, we supplement our model with an auxiliary objective that predicts the subsequent bag of words. Empirical experiments conducted on the OpenSubtitle and Reddit datasets show that the proposed model leads to significant improvements on both relevance and diversity over state-of-the-art baselines.

1 Introduction

Recently, variational Bayesian models have shown attractive merits from both theoretical and practical perspectives (Kingma and Welling, 2013). As one of the most successful variational Bayesian models, Conditional Variational Auto-Encoder (CVAE) (Kingma et al., 2014) was proposed to improve upon the traditional Sequence-to-Sequence (Seq2Seq) dialogue models. The CVAE based models incorporate stochastic latent variables into decoders in order to generate more relevant and diverse responses (Serban et al., 2017; Zhao et al., 2017; Shen et al., 2017). However, existing CVAE based models normally rely on the unimodal distribution with a single latent variable to provide the global guidance to response generation, which is not sufficient to capture the complex semantics and high variability of responses. As a result, the autoregressive decoders used in response generation always tend to ignore these oversimple latent variables and degrade the CVAE based model to the simple Seq2Seq model (aka. the *model collapse* problem).

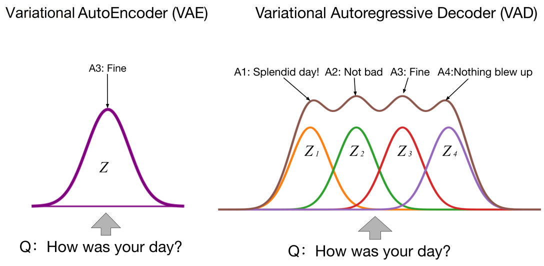

Figure 1: Distributions of latent variable

As illustrated in Figure 1, the unimodal latent variable z used in the conventional VAE usually captures simple unimodal pattern of responses. However, in open-domain conversations, an utterance may have various responses which form complex multimodal distributions. To overcome this problem and improve the quality of generated responses, we propose a novel model, named Variational Autoregressive Decoder (VAD) to iteratively incorporate a series of latent variables into the autoregressive decoder. In particular, a distinct latent variable sampled from CVAE is associated with each time step of the generation, and it is used to condition the next state of the autoregressive decoder (e.g., the hidden state of a RNN). These latent variables at different time steps are integrated by autoregressive decoder to model mutilmodal distribution of text sequences and capture variability of responses as depicted in Figure 1.

---

*Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3154–3163
Brussels, Belgium, October 31 - November 4, 2018.

Partially inspired by the sequential VAE-based models adopted in speech generation (Goyal et al., 2017; Bayer and Osendorfer, 2014), in our VAD the approximate posterior of the latent variable at each time step is augmented by the corresponding hidden state of a backward RNN running through the remaining response sequence. Since the hidden states of the backward RNN contain the information of the succeeding words in the response, they can be used as the guidance for the latent variables to capture the long-term dependency on the future content.

It has been found that auxiliary losses that predict another task-related objective could help latent variables capture more information from different perspectives when training the VAE based models (Zhao et al., 2017). To enhance VAD, we propose a purposely designed auxiliary loss to use the latent variable at each time step to predict the Bag-Of-Words (BOW) of the succeeding subsequence. The proposed auxiliary loss could essentially help VAD to generate more coherent responses.

Experimental results show that the proposed VAD model outperforms the conventional response generation models when evaluated automatically and manually on the OpenSubtitle and Reddit datasets. The contributions in this work are two-fold:

- We propose a novel VAD model for response generation that can better capture the high variability of responses by sequentially associating latent variables to different time steps of autoregressive decoder and approximating the posterior of latent variables by augmenting the hidden states of a backward RNN.
- A BOW based auxiliary objective is proposed to help preserving the diversity of generated responses.

2 Related Work

2.1 Conversational Systems

As neural network based models dominate the research in natural language processing, Seq2Seq models have been widely used for response generation (Sordoni et al., 2015). However, Seq2seq models suffer from the problem of generating generic responses, such as *I don't know* (Li et al., 2016a). Various approaches have been proposed to address this problem, including adding additional information (Li et al., 2016b; Xing et al., 2017; Zhou et al., 2017b) and modifying the architecture of existing models (Li et al., 2016a; Xu et al., 2017; Zhou et al., 2017a).

Another solution to address this problem is to add stochastic latent variables in order to change the deterministic structure of Seq2Seq models. VAE (Kingma and Welling, 2013) is one of the most successful models (Serban et al., 2017; Zhao et al., 2017; Shen et al., 2017; Cao and Clark, 2017). However, VAE-based models only use a single latent variable to encode the whole response sequence, thus suffering from the ***model collapse*** problem (Bowman et al., 2016). To overcome this problem, we propose a novel model that based on the variational autoregressive decoder to better represent highly structural latent variables.

2.2 Variational Autoregressive Models

Recently, some works attempted to combine VAE with autoregressive models to better process input sequences. Broadly speaking, they can be categorized into two groups. Methods in the first group leverage autoregressive models to improve the inference of traditional VAEs. The most well-known model is Inverse Autoregressive Flow (IAF), which used a series of invertible transformations based on the autoregressive model to construct the latent variables (Kingma et al., 2016; Chen et al., 2017). Methods in the second group focus on improving autoregressive models like RNNs by adding variational inference (Bayer and Osendorfer, 2014; Chung et al., 2015; Fraccaro et al., 2016; Goyal et al., 2017). These models usually modeled continuous data such as images and audio signals. For dealing with discrete data such as text, (Li et al., 2017) applied variational recurrent neural networks (VRNN) for text summarization.

Our proposed framework is based on the second line of research, but is different from the previous research as it develops a new strategy of combining VAE with RNN for response generation.

3 Proposed VAD Model

As shown in Figure 2, we use the Seq2Seq model as the basic architecture. The Seq2Seq model is an encoder-decoder neural framework for mapping a source sequence to a target sequence (Sutskever et al., 2014). The input of Seq2seq response generation model is variable-length query sequence

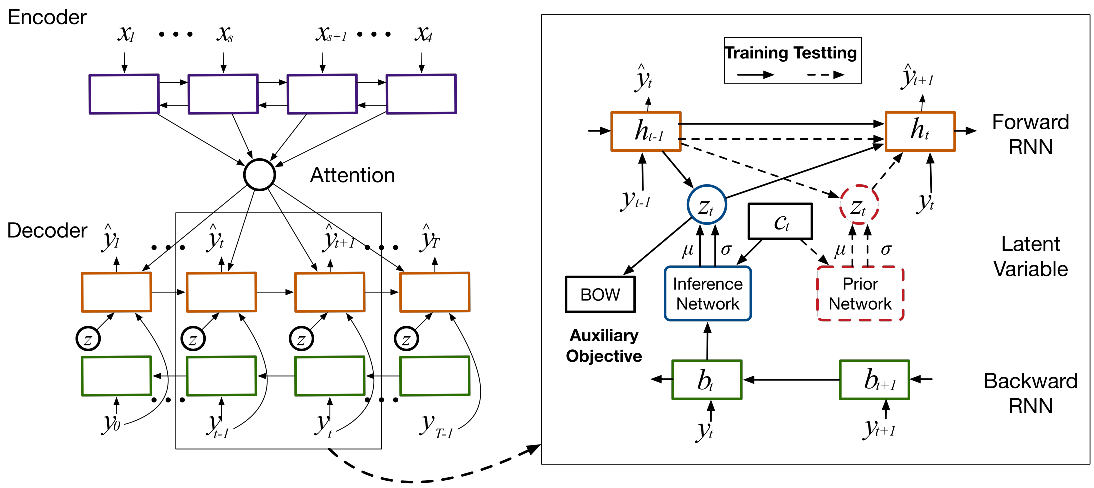

Figure 2: Sequence-to-sequence model using sequential variational decoder.

$\boldsymbol{x} = \{x_1, \ldots, x_m\}$, and the output is a response sequence $\boldsymbol{y} = \{y_1, \ldots, y_n\}$. Both the encoder and decoder are the Recurrent Neural Networks (RNN) with Gated Recurrent Units (GRU) (Chung et al., 2014).

The encoder is a bidirectional GRU that encodes the query sequence as the concatenation of the hidden states of a forward and a backward GRUs. The semantic of word t in the query sequence is represented by $h_t^e = [\overrightarrow{h_t^e}, \overleftarrow{h_t^e}]$, where

$$\begin{aligned} \overrightarrow{h_t^e} &= \overrightarrow{\text{GRU}}(x_t, \overrightarrow{h_{t-1}^e}) \\ \overleftarrow{h_t^e} &= \overleftarrow{\text{GRU}}(x_t, \overleftarrow{h_{t+1}^e}) \end{aligned} \qquad (1)$$

The decoder is a GRU with hidden state h_t^d at each step. The input at step t is the concatenation of previous word in response sequence y_{t-1} and the context vector c_t computed by a neural attention model. The context vector c_t is the weighted sum of the whole encoder's hidden states computed by:

$$\begin{aligned} \alpha_{s,t} &= f_{attention}([h_s^e, h_{t-1}^d]) \\ c_t &= \sum_{s=1}^{m} \alpha_{s,t} h_s^e \end{aligned} \qquad (2)$$

where $f_{attention}$ is a one-layer neural network that produces attention weights, $\alpha_{s,t}$ is the attention weight evaluating the correlation between encoder's hidden state h_s^e and hidden state of decoder h_{t-1}^d. The decoder predicts the next word $\hat{y}_t$ by jointly considering previous word y_{t-1}, attentional context c_t and previous hidden state h_{t-1}^d.

3.1 Conditional Variational Auto-Encoder

The decoder of VAD is based on the Conditional VAE (CVAE) framework (Kingma et al., 2014), which approximates the distribution of random variable $\boldsymbol{y}$ (response) conditioned on $\boldsymbol{x}$ (i.e., query) by incorporating an latent variable z. CVAE introduces a parameterized conditional posterior distribution $q_\theta(z|\boldsymbol{y}, \boldsymbol{x})$ to approximate true posterior distribution $p(z|\boldsymbol{y}, \boldsymbol{x})$. By injecting $q_\theta(z|\boldsymbol{y}, \boldsymbol{x})$, the conditional marginal distribution of $p(\boldsymbol{y}|\boldsymbol{x})$ can be maximized by approximating the Evidence Lower Bound (ELBO):

$$\log p_\phi(\boldsymbol{y}|\boldsymbol{x}) \geq \log p(\boldsymbol{y}|\boldsymbol{x}) - \text{KL}(q_\theta(z|\boldsymbol{y}, \boldsymbol{x}) || p(z|\boldsymbol{y}, \boldsymbol{x}))$$

where KL denotes the Kullback-Leibler divergence. ELBO can be rewritten as a regularized auto-encoder function:

$$\mathcal{L} = \mathbb{E}_{q_\theta(z|\boldsymbol{y},\boldsymbol{x})}\left[p_\phi(\boldsymbol{y}|z, \boldsymbol{x})\right] - \text{KL}(q_\theta(z|\boldsymbol{y}, \boldsymbol{x}) || p_\phi(z|\boldsymbol{x}))$$

where $p_\phi(\boldsymbol{y}|z, \boldsymbol{x})$ is the decoder that decodes $\boldsymbol{y}$ from the latent variable z and conditional variable $\boldsymbol{x}$, $q_\theta(z|\boldsymbol{y}, \boldsymbol{x})$ is the inference model that approximates the true posterior, $p_\phi(z|\boldsymbol{x})$ is the prior model that samples the latent variable from the prior distribution, θ, ϕ are the parameters of the inference and decoder models, respectively. All parameterized distributions are modeled by neural networks.

In the training phase, the latent variable z is sampled from both the inference model and the prior model. z from the inference model is then used to condition the generated distribution $p(\boldsymbol{y}|z, \boldsymbol{x})$. Meanwhile, CVAE minimizes the KL divergence between the latent variables from these two models. This process makes it possible for CVAE to samples z from the prior model only when decoding in the testing phase.

Different from the previous work on CVAE-based response generation that only relys on a single latent variable (Serban et al., 2017; Zhao et al., 2017; Shen et al., 2017), our proposed model incorporates a series of latent variables into the autoregressive decoder. Inspired by the work on variational recurrent neural networks (Goyal et al., 2017; Bayer and Osendorfer, 2014), our model sequentially decodes the response sequence conditioned on the latent variable z_t at each time step by $p_\phi(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x}) = \prod_t p(y_t|\boldsymbol{y}_{<t}, \boldsymbol{z}_t, \boldsymbol{x})$.

3.2 Variational Autoregressive Decoder

Traditional CVAE-based models only use a single standard normal distribution to model the latent variable z. They are usually difficult to model the multi-modal distribution of responses $p(\boldsymbol{y}|z, \boldsymbol{x})$. To overcome this limitation, we propose a Variational Autoregressive Decoder (VAD) that decomposes $\boldsymbol{z}$ into sequential variables z_t at each time step t during response generation. Owing to the autoregressive structure of VAD, the hidden state of backward RNN $\overleftarrow{h^d_t}$ is used to condition the latent variable z_t, which can be seen as a long-term guidance to the generation. Moreover, we propose a novel auxiliary objective, which is specially designed for VAD, to avoid *model collapse*.

At each time step, the decoder uses a forward GRU to process the sequence and predicts the next token by a feed-forward network f_{output} with the softmax activation function. The input to GRU is the combination of the previous word's embedding y_{t-1}, the context vector produced by an attention model c_t and the latent variable z_t. The process is described by,

$$\overrightarrow{h^d_t} = \overrightarrow{\mathrm{GRU}}([y_{t-1}, c_t, z_t], \overrightarrow{h^d_{t-1}}) \tag{3}$$

$$p_\phi(y_t|\boldsymbol{y}_{<t}, \boldsymbol{z}_t, \boldsymbol{x}) = f_{\text{output}}([\overrightarrow{h^d_t}, c_t]) \tag{4}$$

where, $\overrightarrow{h^d_t}$ is the hidden state produced by the forward GRU at time step t. c_t is the attentional weighted sum of the encoder's output.

Inference Model We use the hidden states of the backward RNN running through the response sequence as an additional input to the inference model. The backward RNN processes the sequence by,

$$\overleftarrow{h^d_t} = \overleftarrow{\mathrm{GRU}}(y_{t+1}, \overleftarrow{h^d_{t+1}}) \tag{5}$$

The backward hidden state $\overleftarrow{h^d_t}$ contains the information of succeeding tokens, and it serves as a future plan for generation. By combining the information produced by the backward RNN, the inference model has a better capability of approximating the real posterior distribution.

Considering context variable c_t at each time step as a substitute of the condition variable $\boldsymbol{x}$ in (3.1), c_t is also fed to the inference model. The inference model is a feed-forward neural network f_{infer}. The approximated distribution $q(z_t|\boldsymbol{y}, \boldsymbol{x})$ is a normal distribution $\mathcal{N}(\mu^i, \sigma^i)$, which is parameterized by the output of f_{infer}:

$$[\mu^i, \sigma^i] = f_{\text{infer}}([\overrightarrow{h^d_{t-1}}, c_t, \overleftarrow{h^d_t}]) \tag{6}$$

$$q_\theta(z_t|\boldsymbol{y}, \boldsymbol{x}) = \mathcal{N}(\mu^i, \sigma^i) \tag{7}$$

where the sampling process of z_t is done by reparameterization (Kingma and Welling, 2013).

Prior Model The prior network can only use the observable variables in the testing phase to sample z_t. The observable variables include the previous hidden state $\overrightarrow{h^d_{t-1}}$ and the context variable c_t. The prior model is also modeled by a feed-forward network f_{prior} as follows.

$$[\mu^p, \sigma^p] = f_{\text{prior}}([\overrightarrow{h^d_{t-1}}, c_t]) \tag{8}$$

$$p_\phi(z_t|\boldsymbol{y}_{<t}, \boldsymbol{x}) = \mathcal{N}(\mu^p, \sigma^p) \tag{9}$$

where μ^p, σ^p are the parameters of prior normal distribution.

Auxiliary Objective As discussed in Section 1, the decoder based on the autoregressive model often ignores the latent variables and causes the model to collapse. One way to alleviate this problem is to add an auxiliary loss to the training objective (Zhao et al., 2017; Goyal et al., 2017). To allow the latent variables to capture the information from a different perspective, we use ***Sequential Bag of Word (SBOW)*** as the auxiliary objective for the proposed VAD model. The idea of the ***SBOW*** auxiliary objective is to sequentially predict the bag of succeeding words $\boldsymbol{y}_{bow(t+1,T)}$ in the response using the latent variable z_t at each time step. This auxiliary objective can be seen as the prediction of candidate words for future generation.

Our ***SBOW*** is specially designed for VAD. It is different from the Bag-of-Words (BOW) auxiliary loss used in the CVAE-based models (Zhao et al., 2017), which only uses the latent variable to predict the Bag-Of-Words of the whole sequence. VAD with ***SBOW*** sequentially produces the auxiliary loss for each time step of generation. The auxiliary loss at each time step is computed by

$$p_\xi(\boldsymbol{y}_{bow(t+1,T)}|z_{t:T}) = f_{\text{auxiliary}}(z_t) \tag{10}$$

where $\boldsymbol{y}_{bow(t+1,T)}$ is the bag-of-word vector of the words from $t+1$ to T in the response, and $f_{\text{auxiliary}}$ is a feed-forward neural network with the softmax output.

3.3 Learning

The loss function of our model is the sum of the losses at each time step, including the weighed sum of the ELBO loss $\mathcal{L}_{ELBO}(t)$ and the auxiliary loss $\mathcal{L}_{AUX}(t)$ where $\mathcal{L}_{ELBO}(t)$ can be further decomposed into a log-likelihood loss and the KL divergence:

$$\begin{aligned}\mathcal{L} &= \sum_t [\mathcal{L}_{ELBO}(t) + \alpha\mathcal{L}_{AUX}(t)] \\ &= \sum_t [(\mathcal{L}_{LL}(t) - \mathcal{L}_{KL}(t)) + \alpha\mathcal{L}_{AUX}(t)]\end{aligned} \tag{11}$$

Here, $\mathcal{L}_{LL}(t)$ denotes the log-likelihood loss when predicting y_t. $\mathcal{L}_{KL}(t)$ is the KL-divergence of the approximate posteriori q_θ and priori p_ϕ at time step t. $\mathcal{L}_{AUX}(t)$ is the auxiliary loss when predicting ***SBOW*** as described in Section 3.2. α is the weight controlling the auxiliary loss. The losses are computed by

$$\begin{aligned}\mathcal{L}_{LL}(t) &= \mathbb{E}_{q_\theta(z_t|\boldsymbol{y},\boldsymbol{z})}[\log p_\theta(y_t|\boldsymbol{y}_{<t}, z_t, x_t)] \\ \mathcal{L}_{KL}(t) &= \mathrm{KL}(q_\theta(z_t|\boldsymbol{y},\boldsymbol{x})||p_\phi(z_t|\boldsymbol{y}_{<t},\boldsymbol{x})) \\ \mathcal{L}_{AUX}(t) &= \mathbb{E}_{q_\theta(z_t|\boldsymbol{y},\boldsymbol{z})}\left[\log p_\xi(\boldsymbol{y}_{bow(t+1,T)}|z_t)\right]\end{aligned}$$

All the parameters are learned by optimizing Equation (11) and updated with back-propagation.

4 Experimental Setup

4.1 Datasets

We evaluate the proposed model on two datasets: **OpenSubtitles** and **Reddit**. The OpenSubtitles dataset contains subtitles for movies in various languages. Here, we only choose the English version of OpenSubtiles. The Reddit dataset is crawled from comments of Reddit[1] which is an American social news discussion website. We collected more than 10 million single-turn dialogues from 100 topics posted in 2017. For each dataset, we randomly select 6 million conversations for training, 10k for validation and 5k for testing. For every conversation, we remove the sentences whose length is shorter than 6 words and only keep the first 40 words for sentences longer than 40. We keep top 15k frequent words as the vocabulary for OpenSubtitles and 20k frequent words for Reddit.

4.2 Hyper-parameters and Training Setup

We use the pre-trained GloVe 300-dimensional word embeddings for both the encoder and the decoder. The encoder is a bidirectional RNN with GRU with the size of the hidden state set to 512. The size of the hidden states of GRU in the decoder is also set to 512. We apply Layer Normalization when training the decoder. The size of the latent variables is set to 400. The inference network and the prior network are all one-layer feed-forward network. All weights are initialized by the xavier method (Glorot and Bengio, 2010). The model is trained end-to-end by Adam optimizer (Kingma and Ba, 2014) with the learning rate set to 10^{-4} and gradient clipped at 1. When generating text, we adopt the greedy strategy and the KL-annealing strategy, with the temperature varying from 0 to 1 and increased by 10^{-5} after each iteration of batch update.

4.3 Baselines

We compare our proposed model with the following three baselines:

- **Seq2Seq**: Sequence-to-Sequence model with attention (Sordoni et al., 2015).
- **CVAE**: Conditional Variational Auto-Encoder for generating responses (Serban et al., 2017). Different from our model, CVAE uses a unimodal Gaussian distribution to model the whole response and append the output of VAE as an additional input to decoder. We also use the KL annealing strategy when training CVAE with the same parameter setting as in our model.

---

[1] `http://www.reddit.com`

- **CVAE+*BOW* loss**: CVAE model with the auxiliary bag-of-words loss (Zhao et al., 2017).

4.4 Metrics

We employ three types of commonly used automatic evaluation metrics and human evaluation in our experiments:

Embedding Similarity: Embedding-based metrics compute the cosine similarity between the sentence embedding of a ground-truth response and that of the generated one. There are various ways to derive the sentence-level embedding from the constituent word embeddings. In our experiments, we apply three most commonly used strategies to obtain the sentence-level embeddings. $\mathbf{EMB_A}$ calculates the average of word embeddings in a sentence. $\mathbf{EMB_E}$ takes the most extreme value among all words for each dimension of word embeddings in a sentence. $\mathbf{EMB_G}$ greedily calculates the maximum of cosine similarity of each token in two sentences and take the average of them to get the final matching score (Liu et al., 2016).

RUBER Score: RUBER (Referenced metric and Unreferenced metric Blended Evaluation Routine) is a newly proposed metric for evaluating the quality of response in conversations that show high correlation with human annotation (Tao et al., 2017). RUBER evaluates the generated responses by taking into account both the ground-truth responses and the given queries. For the *referenced* metric, RUBER calculates the embedding-based cosine similarity between a generated response and its corresponding ground-truth. For the *unreferenced* metric, RUBER firstly trains a neural network by a response retrieval task and evaluates the relatedness between a generated response and its query. Evaluating RUBER score can be treated as a rough simulation to the well-known *Turing Test*. For blending the two metrics, there are two strategies: taking the geometric mean ($\mathbf{RUB_G}$) or the arithmetic mean ($\mathbf{RUB_A}$). The RUBER score ranges between 0 and 1 and higher scores imply better relatedness.

Diversity: Diversity metrics evaluate the informativeness and diversity of generated responses. In our experiments, we use $\mathbf{Dist_1}$ and $\mathbf{Dist_2}$ (Li et al., 2016a) to evaluate the diversity and **Entropy** to measure the informativeness. $\mathbf{Dist_1}$ (or $\mathbf{Dist_2}$) calculates the ratio of the number of unique unigrams (or bigrams) against the total number of unigrams (or bigrams). Higher $\mathbf{Dist_1}$ (or $\mathbf{Dist_2}$) implies more diverse vocabularies used in responses. **Entropy** as a metric proposed by (Serban et al., 2017) calculates the average entropy in a generated response. According to information theory, it is known that low-frequent words have higher entropy and carries more information. Therefore, we use this **Entropy** to measure the informativeness and diversity of the generated responses. The unit of **Entropy** is *bit* and Higher **Entropy** correlates to more informative response.

Human Evaluation: In human evaluation, 10 research students are arranged to rate the generated responses generated by CVAE with *BOW* auxiliary loss and our model. We randomly selected 100 queries from the Reddit dataset[2] and used each model to generate the best responses. Each query with its ground-truth response and the two generated responses are simultaneously shown to the human evaluators. The evaluators are asked to rate the responses based on grammatical correctness, coherence and relevance to queries (tie is permitted).

5 Results

5.1 Quantitative Analysis

The experimental results evaluated by automatic metrics on the OpenSubtitles and the Reddit datasets are shown in Table 1 and 2, respectively. It is observed that both CVAE-based models and our proposed models outperform Seq2Seq by a large margin, showing the effectiveness of adding variational latent variable for response generation. However, using different structure of variational models leads to differences in performance on both plausibility and diversity. Our model with or without the SBOW auxiliary loss outperforms CVAE as observed by the significant boost in semantic relevance-oriented metrics (embedding similarities and RUBER score) and diversity-oriented metrics. This is mainly due to the different strategy employed for representing latent variables. CVAE only uses a unimodal latent variable as the semantic signal of the whole response sequence which limits its capability of capturing

---

[2] The reason of not conducting the human evaluation on the OpenSubtitles dataset is that query-response pairs in the OpenSubtitles dataset are extracted from movie scripts and hence are more difficult to evaluate without the context information.

Method	Embedding Similarity			RUBER		Diversity		
	EMB_A	EMB_E	EMB_G	Rub_G	Rub_A	Dist_1	Dist_2	Entropy
Ground Truth	1.000	1.000	1.000	0.872	0.881	0.091	0.423	11.886
Seq2Seq	0.572	0.493	0.487	0.441	0.462	0.015	0.053	6.730
CVAE	0.639	0.531	0.578	0.562	0.580	0.026	0.102	8.215
CVAE+BOW loss	0.659	0.530	0.526	0.602	0.597	0.041	0.302	9.519
Ours (without *SBOW*)	0.678	0.520	0.563	0.591	0.604	0.031	0.259	8.815
Ours	**0.714**	**0.582**	**0.642**	**0.635**	**0.642**	**0.053**	**0.404**	**10.976**

Table 1: Experimental results on the OpenSubtitles dataset.

Method	Embedding Similarity			RUBER		Diversity		
	EMB_A	EMB_E	EMB_G	Rub_G	Rub_A	Dist_1	Dist_2	Entropy
Ground Truth	1.000	1.000	1.000	0.842	0.869	0.083	0.399	10.089
Seq2Seq	0.520	0.382	0.377	0.371	0.386	0.007	0.042	6.003
CVAE	0.602	0.496	0.531	0.541	0.555	0.019	0.097	7.010
CVAE+BOW loss	0.659	0.531	0.578	0.591	0.604	0.026	0.282	9.215
Ours (without *SBOW*)	0.628	0.540	0.563	0.607	0.610	0.021	0.216	8.222
Ours	**0.692**	**0.556**	**0.598**	**0.622**	**0.629**	**0.046**	**0.391**	**10.043**

Table 2: Experimental results on the Reddit dataset.

Models	OpenSubtitles	Reddit
Ground Truth	15.31	17.48
CVAE+*BOW*	9.66	10.83
Ours	11.81	14.09

Table 3: The average length of responses.

Models	Wins	Loses	Ties
CVAE+*BOW*	0.207	0.678	0.115
Ours	0.685	0.200	0.115

Table 4: Results of human judgment on the generated responses.

variability of response sequences. By incorporating a series of time-varying latent variables into each step of autoregressive decoder, our model is able to model more complicated multimodal distributions of response sequences and capture more detailed semantic information.

Since adding the auxiliary loss could alleviate the *model collapse* problem, we found that CVAE model with the *BOW* auxiliary loss outperforms our basic model without auxiliary loss, especially on the diversity metrics. When adding the proposed *SBOW* auxiliary loss into our model, we found that our generated responses have shown better diversity compared to those generated by CVAE+*BOW* loss. The encouraging improvement is attributed to the autoregressive structure of our variational inferences, which makes it possible to gradually introduce additional information of *SBOW*. To better demonstrate the impact of *SBOW*, we calculate the average length of the generated responses of our model and CVAE with *BOW* loss and show the results in Table 3. It is observed that our model with *SBOW* can generate longer responses than CVAE+*BOW*. The results validate the effectiveness of adding the *SBOW* auxiliary objective into our model.

The evaluation results of human judgment is shown in Table 4. It is observed that the responses generated by our proposed VAD is more plausible than CVAE+*BOW* from human perspectives. We also conduct t-test to compare our model with CVAE+*BOW*. The results show that the improvement of VAD over CVAE+*BOW* is statistically significant ($p < 0.01$).

5.2 Qualitative Analysis

Case Study To empirically analyze the quality of the generated responses, we show some example responses generated by our model and two baselines (Seq2Seq and CVAE+*BOW*) in Table 5. It is observed that Seq2seq often generates generic responses that starting with '*I don't know*' or '*I am not sure*', since the deterministic structure of Seq2seq limits the diversity of generation. Injecting variational latent variables avoids dull responses as can be seen from the responses gen-

Dataset	Query	Seq2seq	CVAE+*BOW*	Ours
OpenSub	why he is not here ?	i do n't know where	he 's not present .	maybe he 's been caught in the rain too .
	that wasn ' t easy for him .	i ' m not so sure i know.	did he do good job and good job ?	it 's a tough job but it 's a great deal of money .
	what is the alternative solution ?	i ' m not interested.	i am the solution.	i ' m afraid there is no alternative .
Reddit	he looks exactly like my australian uncle .	he is NUM NUM	lol , you know him , he is my uncle .	nope , he 's a young man and was born in LOC .
	why no windows ?	i ' m not so sure you 're right .	do you like windows NUM ?	cuz linux can be a great os .
	i hope the grand tour will make an episode .	i ca n't commit post though .	i 'm wondering getting it .	i would hope that it will be on netflix as well .

Table 5: Example responses generated by our model and two baselines (Seq2Seq and CVAE+*BOW*) from the OpenSubtitles and the Reddit Datasets.

erated by CVAE+*BOW* and our model. However, we found that CVAE+*BOW* tends to copy the given queries (the first and fourth example in Table 4) and repeatedly generate redundant tokens (the second example). The generated responses of our model are more fluent and relevant to queries. Also, our model generates longer responses compared to the baselines.

KL Divergence Visualization In order to demonstrate that our model is able to alleviate the ***model collapse*** problem of VAE, we visualize the KL divergence between the approximate posterior distribution $q_\theta(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{x})$ and priori $p_\phi(\boldsymbol{z}|\boldsymbol{x})$ during the training process of our models and CVAE with *BOW* loss in Figure 3. As we know, when variational models ignore the latent variable, the generated value $\boldsymbol{y}$ will be independent of the latent variable $\boldsymbol{z}$ which causes the KL divergence in Equation (3.1) to approach 0. The higher KL value during training means more dependence between $\boldsymbol{y}$ and $\boldsymbol{z}$. In this experiment, we use the same KL annealing strategies for our model and CVAE+*BOW* as described in Section 4.2. The KL divergence of the two models on the OpenSubtitles and the Reddit datasets during training is plotted in Figure 3. It is observed that the KL divergence of our model converges to a higher value compared to that of CVAE+*BOW*. It shows that our model could better alleviate the *model collapse* problem.

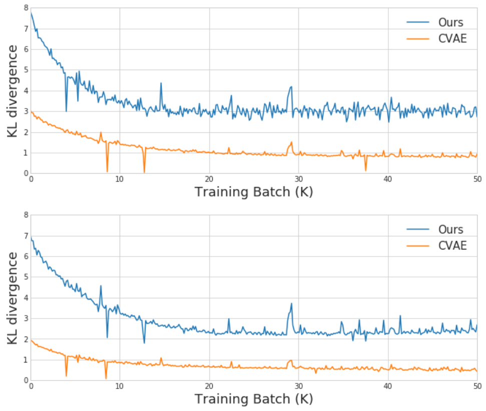

Figure 3: KL divergence during training.

6 Conclusion

In this paper, a novel variational autoregressive decoder is proposed to improve the performance of VAE-based models for open-domain response generation. By injecting the variational inference into the RNN-based decoder and applying carefully designed conditional variables and auxiliary objective for latent variables, the proposed model is expected to better modeling semantic information of text in conversations. Quantitative and qualitative experimental results show clear performance improvement of the proposed model over competitive baselines. In future works, we will explore the use of other attributes of responses such as Part-of-Speech (POS) tags and chunking sequences as additional conditions for better response generation.

Acknowledgements

This work was supported by National Natural Science Foundation of China U1636103, 61632011, Key Technologies Research and Development Program of Shenzhen JSGG20170817140856618, Shenzhen Foundational Research Funding 20170307150024907, Research Grants Council of Hong Kong (PolyU 152036/17E, 152040/18E), The Hong Kong Polytechnic University (G-YBJP, G-YBP6). and Innovate UK (grant no. 103652).

References

Justin Bayer and Christian Osendorfer. 2014. Learning stochastic recurrent networks. In *NIPS 2014 Workshop on Advances in Variational Inference*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, page 10.

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 182–187.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017. Variational lossy autoencoder. In *5th International Conference on Learning Representations (ICLR)*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. 2016. Sequential neural models with stochastic layers. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, pages 2199–2207.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. 2017. Z-forcing: Training stochastic recurrent networks. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, pages 6697–6707.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Proceedings of the 30th Advances in Neural Information Processing Systems*, pages 4743–4751.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL-HLT*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 994–1003.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 3295–3301.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (EMNLP)*, pages 504–509.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2017. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. *arXiv preprint arXiv:1701.03079*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*.

Zhen Xu, Bingquan Liu, Baoxun Wang, SUN Chengjie, Xiaolong Wang, Zhuoran Wang, and Chao Qi. 2017. Neural response generation via gan with an approximate embedding layer. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 617–626.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 654–664.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017a. Mechanism-aware neural machine for dialogue response generation. In *Proceddings of 31st AAAI Conference on Artificial Intelligence*.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2017b. Emotional chatting machine: emotional conversation generation with internal and external memory. *arXiv preprint arXiv:1704.01074*.

Integrating Transformer and Paraphrase Rules for Sentence Simplification

Sanqiang Zhao† , **Rui Meng**†, **Daqing He**†, **Saptono Andi***, **Parmanto Bambang***
† Department of Informatics and Networked Systems , School of Computing and Information
* Department of Health Information Management, School of Health and Rehabilitation Sciences
University of Pittsburgh
Pittsburgh, PA, 15213
`{sanqiang.zhao, rui.meng, daqing, ans38, parmanto}@pitt.edu`

Abstract

Sentence simplification aims to reduce the complexity of a sentence while retaining its original meaning. Current models for sentence simplification adopted ideas from machine translation studies and implicitly learned simplification mapping rules from normal-simple sentence pairs. In this paper, we explore a novel model based on a multi-layer and multi-head attention architecture and we propose two innovative approaches to integrate the Simple PPDB (A Paraphrase Database for Simplification), an external paraphrase knowledge base for simplification that covers a wide range of real-world simplification rules. The experiments show that the integration provides two major benefits: (1) the integrated model outperforms multiple state-of-the-art baseline models for sentence simplification in the literature (2) through analysis of the rule utilization, the model seeks to select more accurate simplification rules. The code and models used in the paper are available at https://github.com/Sanqiang/text_simplification.

1 Introduction

Sentence simplification aims to reduce the complexity of a sentence while retaining its original meaning. It can benefit individuals with low-literacy skills (Watanabe et al., 2009) including children, non-native speakers and individuals with language impairments such as dyslexia (Rello et al., 2013), aphasic (Carroll et al., 1999).

Most of the previous studies tackled this task in a way similar to machine translation (Xu et al., 2015a; Zhang and Lapata, 2017), in which models are trained on a large number of pairs of sentences, each consisting of a normal sentence and a simplified sentence. Statistical and neural network modeling are two major methods used for this task. The statistical models have the benefit of easily integrating with human-curated rules and features, thus they generally perform well even they are trained with a limited number of data. In contrast, neural network models could learn the simplifying rules automatically without the need for feature engineering, but at the cost of requiring a huge amount of training data. Even though models based on neural networks have outperformed the statistical methods in multiple Natural Language Processing (NLP) tasks, their performance in sentence simplification is still inferior to that of statistical models (Xu et al., 2015a; Zhang and Lapata, 2017). We speculate that current training datasets may not be large and broad enough to cover common simplification situations. However, human-created resources do exist which can provide abundant knowledge for simplification. This motivates us to investigate if it is possible to train neural network models with these types of resources.

Another limitation to using existing neural network models for sentence simplification is that they are only able to capture frequent transformations; they have difficulty in learning rules that are not frequently observed despite their significance. This may be due to nature of neural networks (Feng et al., 2017): during training, a neural network tunes its parameters to learn how to simplify different aspects of the sentence, which means that all the simplification rules are actually contained in the shared parameters. Therefore, if one simplification rule appears more frequently than others, the model will be trained to be more focused on it than the infrequent ones. Meanwhile, models tend to treat infrequent rules as noise if they are merely trained using sentence pairs. If we can leverage an additional memory component to maintain simplification rules individually, it would prevent the model from forgetting low-frequency rules as well as help it to distinguish real rules from noise. Therefore, we propose the Deep Memory Augmented Sentence Simplification (DMASS) model. For comparison pur-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3164–3173
Brussels, Belgium, October 31 - November 4, 2018.

pose, we also introduce another approach, Deep Critic Sentence Simplification (DCSS) model, to encourage applying the less frequently occurring rules by revising the loss function. It this way, simplification rules are encouraged to maintained internally in the shared parameters while avoiding the consumption of an unwieldy amount of additional memory.

In this study, we propose two improvements to the neural network models for sentence simplification. For the first improvement, we propose to use a multi-layer, multi-head attention architecture (Vaswani et al., 2017). Compared to RNN/LSTM (Recurrent Neural Network / Long Short-term Memory), the multi-layer, multi-head attention model would be able to selectively choose the correct words in the normal sentence and simplify them more accurately.

Secondly, we propose two new approaches to integrate neural networks with human-curated simplification rules. Note that previous studies rarely tried to incorporate explicit human language knowledge into the encoder-decoder model. Our first approach, DMASS, maintains additional memory to recognize the context and output of each simplification rules. Our second approach, DCSS, follows a more traditional approach to encode the context and output of each simplification rules into the shared parameters.

Our empirical study demonstrates that our model outperforms all the previous sentence simplification models. They achieve both a good coverage of rules to be applied (recall) and a high accuracy gained by applying the correct rules (precision).

2 Related Work

Sentence Simplification For statistical modeling, Zhu et al. (2010) proposed a tree-based sentence simplification model drawing inspiration from statistical machine translation. Woodsend and Lapata (2011) employed quasi-synchronous grammar and integer programming to score the simplification rules. Wubben et al. (2012) proposed a two-stage model PBMT-R, where a standard phrase-based machine translation (PBMT) model was trained on normal-simple aligned sentence pairs, and several best generations from PBMT were re-ranked based how dissimilar they were to a normal sentence. Hybrid, a model proposed by Narayan and Gardent (2014) was also a two-stage model combining a deep semantic analysis and machine translation framework. SBMT-SARI (Xu et al., 2016) achieved state-of-the-art performance by employing an external knowledge base to promote simplification. In terms of neural network models, Zhang and Lapata (2017) argued that the RNN/LSTM model generated sentences but it does not have the capability to simplify them. They proposed DRESS and DRESS-LS that employ reinforcement learning to reward simpler outputs. As they indicated, the performance is still inferior due to the lack of external knowledge. Our proposed model is designed to address the deficiency of current neural network models which are not able to integrate an external knowledge base.

Augmented Dynamic Memory Despite positive results obtained so far, a particular problem with the neural network approach is that it has a tendency towards favoring to frequent observations but overlooking special cases that are not frequently observed. This weakness with regard to infrequent cases has been noticed by a number of researchers who propose an augmented dynamic memory for multiple applications, such as language models (Daniluk et al., 2017; Grave et al., 2016), question answering (Miller et al., 2016), and machine translation (Feng et al., 2017; Tu et al., 2017). We find that current sentence simplification models suffer from a similar neglect of infrequent simplification rules, which inspires us to explore augmented dynamic memory.

3 Our Sentence Simplification Models

3.1 Multi-Layer, Multi-Head Attention

Our basic neural network-based sentence simplification model utilizes a multi-layer and multi-head attention architecture (Vaswani et al., 2017). As shown in Figure 1, our model based on the Transformer architecture works as follows: given a pair consisting a normal sentence I and a simple sentence O, the model learns the mapping from I to O.

The encoder part of the model (see the left part of Figure 1) encodes the normal sentence with a stack of L identical layers. Each layer has two sublayers: one layer is for multi-head self-attention and the other one is a fully connected feed-forward neural network for transformation. The multi-head self-attention layer encodes the output from the

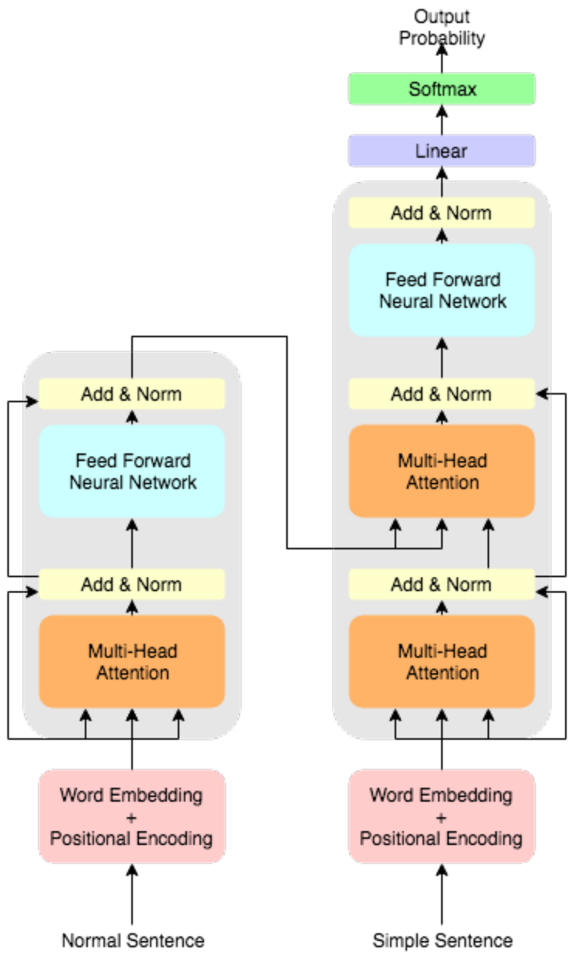

Figure 1: Diagram of the Transformer architecture

previous layer into hidden state $e_{(s,l)}$ (step s and layer l) as shown in Equation 1, where $\alpha^{enc}_{(s',l)}$ indicates the attention distribution over the step s' and layer l. Each hidden state summarizes the hidden states in the previous layer through the multi-head attention function $a()$ (Vaswani et al., 2017) where H refers to the number of heads.

The right part of Figure 1 denotes the decoder for generating the simplified sentence. The decoder also consists of a stack of L identical layers. In addition to the same two sub-layers as those in the encoder part, the decoder also inserts another multi-head attention layer aiming to attend on the encoder outputs. The bottom multi-head self-attention plays the same role as the one in the encoder, where the hidden state $d_{(s,l)}$ is computed in the Equation 2. The upper multi-head attention layer is used to seek relevant information from encoder outputs. Through the same mechanism, context vector $c_{(s,l)}$ (step s and layer l) is computed in the Equation 3.

$$e_{(s,l)} = \sum_{s'} \alpha^{enc}_{(s',l)} e_{(s',l-1)}, \quad \alpha^{enc}_{(s',l)} = a(e_{(s,l)}, e_{(s',l-1)}, H) \tag{1}$$
[1]

$$d_{(s,l)} = \sum_{s_\prime} \alpha^{dec}_{(s_\prime,l)} d_{(s_\prime,l-1)}, \quad \alpha^{dec}_{(s_\prime,l)} = a(d_{(s,l)}, c_{(s_\prime,l-1)}, H) \tag{2}$$
[2]

$$c_{(s,l)} = \sum_{s_\prime} \alpha^{dec2}_{(s_\prime,l)} e_{(s_\prime,L)}, \quad \alpha^{dec2}_{(s_\prime,l)} = a(d_{(s,l)}, e_{(s_\prime,L)}, H) \tag{3}$$

The model is trained to minimize the negative log-likelihood of the simple sentence, $L_{seq} = -logP(O|I,\theta)$ where θ represents all the parameters in the current model.

3.2 Integrating with Simple PPDB

A previous study (Xu et al., 2016) has demonstrated the benefits of using an external knowledge base in conjunction with a statistical simplification model. However, as far as we know, no efforts have been made to integrate neural network models with the knowledge base, and our study is the first to meet this goal.

Weight	Type	Rule
0.99623	[VP]	recipient → have receive
0.75530	[NN]	recipient → winner
0.58694	[NN]	recipient → receiver
0.46935	[NN]	recipient → host

Table 1: Examples from the Simple PPDB

Simple PPDB (Pavlick and Callison-Burch, 2016) refers to a paraphrase knowledge base for simplification. It is a refined version of another knowledge, PPDB (Ganitkevitch et al., 2013), which was originally designed to support paraphrase. Simple PPDB contains 4.5 million paraphrase rules, each of which provides the mapping from a normal phrase to a simplified phrase, the syntactic type of the normal phrase, and the simplification weight. Table 1 shows four examples, where "recipient" can be simplified to "winner" with a weight 0.75530 if "recipient" is a singular noun (NN).

3.2.1 Deep Critic Sentence Simplification Model (DCSS)

The Simple PPDB offers guidance about whether a word needs to be simplified and how it should

---

[1] The lowest hidden state $e_{(:,0)}$ is the word embedding.

[2] The lowest context vector $c_{(:,0)}$ is the word embedding.

be simplified. The Deep Critic Sentence Simplification (DCSS) model is designed to apply rules identified by the Simple PPDB by introducing a new loss function. Different from the standard loss function that minimizes the distance away from the ground truth, the new loss function aims to maximize the likelihood of applying simplification rules. It also reweights the probability of generating each word by its simplification weight in order to relieve the problem of overlooking infrequent simplification rules.

For example, given a normal sentence in the training set, "the recipient of the kate greenaway medal", the simplified sentence is "the winner of the kate greenaway medal.", where "recipient" is simplified to "winner", which is identified by Simple PPDB. The major goal of the loss functions is to support the model in generating the simplified word "winner" while deterring the model from generating the word "recipient". Specifically, for an applicable simplification rule, our new loss function maximizes the probability of generating the simplified form (word "winner") and meanwhile minimizes the probability of generating the original form (word "recipient"). As in Equation 4, where w_{rule} indicates the weight of the simplification rule provided by the Simple PPDB, once the model generates "recipient", the model is criticized to generate word "winner"; when model predicts correctly with "winner", the model is trained to minimize the probability of "recipient". In this way, the model avoids selecting normal words and instead becomes inclined to choose the simplified words.

$$L_{critic} = \begin{cases} -w_{rule} logP(winner|I, \theta) \\ \qquad\qquad \text{if model generates recipient} \\ w_{rule} logP(recipient|I, \theta) \\ \qquad\qquad \text{if model generates winner} \end{cases} \tag{4}$$

The L_{critic} merely focuses on the words identified by the Simple PPDB and L_{seq} focuses on the entire vocabulary. So, the model is trained in an end-to-end fashion by minimizing L_{seq} and L_{critic} alternately.

3.2.2 Deep Memory Augmented Sentence Simplification Model (DMASS)

DCSS, similar to the majority of neural network models, uses a piece of shared memory, i.e. the parameters, as the media to store the learned rules from the data. As a result, it still focuses much more on rules that are frequently observed and ignores the rules observed infrequently. However, infrequent rules are still important, particularly when the training data is limited.

In order to make full use of the rules in the knowledge base, we introduce the Deep Memory Augmented Sentence Simplification (DMASS) model. DMASS has an augmented dynamic memory to record multiple key-value pairs for each rule in the Simple PPDB. The key vector stores a context vector that is computed based on the weighted average of encoder hidden states and the current decoder hidden states. The value vector stores the output vector.

Our DMASS model is illustrated in Figure 2. Given the same example normal sentence " the recipient of kate greenaway medal", Simple PPDB determines that the word "recipient" should be simplified to "winner". The encoder represents the normal sentence as a list of hidden states, [$e_{(1,L)}$, $e_{(2,L)}$, ...] where L indicates the final layer of encoder hidden states. When predicting the next word in the simplified sentence, the decoder of layer j represents the previous words as hidden states [$d_{(1,j)}$, ...]. $c_{(1,j)}$ refers to the current context vector following attention layer, which is the weighted average of [$e_{(1,L)}$, $e_{(2,L)}$, ...] based on $d_{(1,j)}$. A feed-forward fully connected neural network (FFN) combines the output of the decoder and the output from memory read module into the final output r_{winner}. In addition to the word prediction, $c_{(1,j)}$ and r_{winner} will be sent to memory update module.

In the remainder of this section, we will introduce the two modules of DMASS mentioned above: Memory Read Module and Memory Update Module.

Memory Read Module The memory read module incorporates rules into prediction. As shown in Figure 2, current augmented memory contains three candidate rules for the word "recipient", which indicates that it can be simplified into "winner", "receiver" or "host", respectively. The current context vector $c_{(1,j)}$ is treated as a query to search for suitable rules by using Equation 5, where α_i^r denotes the weight for i^{th} rule, which is computed through the dot product between current context vector $c_{(1,j)}$ and c_i. Then using Equation 6, α_i^r weights each output vector to generate mem-

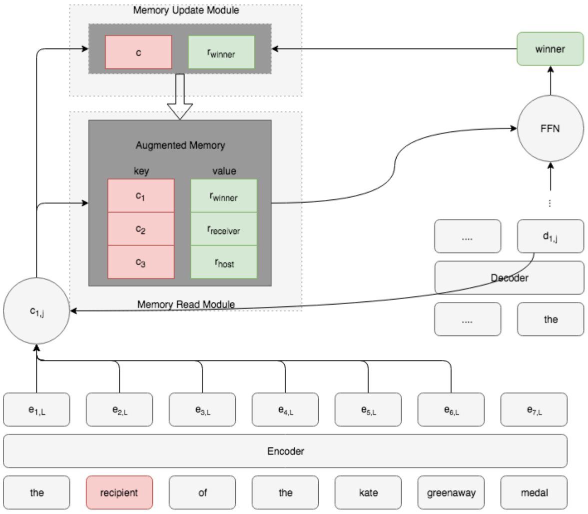

Figure 2: Diagram of DMASS Model

ory read output.

$$\alpha_i^r = \frac{e_i}{\sum_j e_j} \qquad e_i = exp(c_{(1,j)} \cdot c_i) \tag{5}$$

$$r_o = \sum \alpha_i^r r_r \qquad r_r \in [r_{winner}, r_{receiver}, r_{host}] \tag{6}$$

Memory Update Module The task of the memory update module is to update the key and value vectors in the augmented memory. Once the model predicts the output vector r_{winner}, both r_{winner} and the current context vector $c_{1,j}$ are sent to the memory update module. If the augmented memory does not contain the key-value pair for the rule, $c_{1,j}$ and r_{winner} are appended to the memory. If the augmented memory contains the key-value pair, the key vector is updated as the mean of current key vector and $c_{1,j}$. Similarly, the value vector is also updated as the mean of current value vector and r_{winner}.

4 Experiments

Dataset We utilize the dataset *WikiLarge* (Zhang and Lapata, 2017) for training. It is the largest Wikipedia corpus, constructed by merging previously created simplification corpora. Specifically, the training dataset contains 296,402 normal-simple sentence pairs gathered from (Zhu et al., 2010; Woodsend and Lapata, 2011; Kauchak, 2013). For validation and testing, we use the dataset *Turk* created by (Xu et al., 2016). In this dataset, eight simplified reference sentences for each normal sentence are used as the ground-truth, all of which are generated by Amazon Mechanical Turk workers. The *Turk* dataset contains 2,000 data samples for validation and 356 samples for testing. We consider the *Turk* to be the most reliable data set because (1) it is human-generated and (2) it contains multiple simplification references for each normal sentence due to the existence of multiple equally good simplifications of each sentence. We also include the second test set *Newsela*, a corpus introduced by (Xu et al., 2015b) who argue that only using normal-simple sentence pairs from Wikipedia is suboptimal due to the automatic sentence alignment which unavoidably introduces errors, and the uniform writing style which leads to systems that generalize poorly. The test set contains 1,419 normal-simple sentence pairs[3]. To demonstrate that our models are able to perform well on a different style of corpus, we report the results of *Newsela* test set by using the models trained/tuned on *Turk* dataset. Following Zhang and Lapata (2017)'s way, we tag and anonymize name entities with a special token in the format of **NE@N**, where NE includes $\{PER, LOC, ORG\}$ and N indicates the N^{th} distinct NE type of entity. We also replace those tokens occurring three times or less in the training set with a mark "UNK" as mentioned in (Zhang and Lapata, 2017).

Evaluation Metrics We report the results of the experiment with two metrics that are widely used in the literature: SARI (Xu et al., 2016) and FKGL (Kincaid et al., 1975). FKGL computes the sentence length and word length as a way to measure the simplicity of a sentence. The lower value of FKGL indicates simpler sentence. FKGL measures the simplicity of a sentence without considering the ground truth simplification references and it correlates little with human judgment (Xu et al., 2016), so we also use another metric, SARI. SARI, which stands for "System output Against References and against the normal sentence", computes the arithmetic mean of N-grams (N includes 1,2,3 and 4) F1-score of three rewrite operations: addition, deletion, and keeping. Specifically, it rewards addition operations where a word in the generated simplified sentence does not appear in the normal sentence but is mentioned in the reference sentences. It also rewards words kept or deleted in both the simplified sentence and the reference sentences. In our experiment, we also present the F1-score of three rewrite

---

[3] Because the earlier publications don't provide preprocess details, we use our own script to pre-process the articles into sentence pairs.

operations: addition, deletion, and keeping. Xu et al. (2016) demonstrated that SARI correlates most closely to human judgments in sentence simplification tasks. Thus, we treated SARI as the most important measurement in our study.

Because SARI rewards deleting and adding separately, we also include another metric to measure the correctness of lexical transformation, namely word simplification, verified by Simple PPDB. By comparing the normal sentence and ground truth simplified references, we collect rules that are correct to be used for simplifying each normal sentence. Then we calculate the precision, recall, and F1 score for using the correct rules. As a result, the recall expresses the coverage of rules to be applied, and the precision implies the accuracy gained by applying the correct rules.

Training Details We initialized the encoder and decoder word embedding lookup matrices with 300-dimensional Glove vectors(Pennington et al., 2014). The word embedding dimensionality and the number of hidden units are set to 300. During the training, we regularize all layers with a dropout rate of 0.2 (Srivastava et al., 2014). For multi-layer and multi-head architecture, 4 encoder and decoder layers (set L as 4) and 5 multi-attention heads (set H as 5) are used. We will discuss the trade-off between different layers and different heads in Sections 4.1. For DMASS, we use the context vector based on the first layer of the decoder (set j as 1). For optimization, we use Adagrad (Duchi et al., 2011) with the learning rate set to 0.1. The gradient is truncated by 4 (Pascanu et al., 2013).

4.1 Impacts of Multi-Layer, Multi-Head Attention Architecture

The reason to employ the Transformer architecture in the sentence simplification task is that we believe that its multi-layer, multi-head attention provides a better capability of modeling both the overall context and the important cues for sentence simplification. In this section, we examine the applicability of multi-layer, multi-head attention architecture to the sentence simplification task. We compare our results against the RNN/LSTM-based sentence simplification models. Note that the results of our models presented here have not been integrated with the Simple PPDB.

Table 2 shows the experiment results where LxHy indicates a run with Transformer using x layers and y heads. When compared with results of RNN/LSTM, our Transformer-based model performed better in terms of SARI and FKGL values. In addition, with the increased number of layers or heads, the values of SARI and FKGL improve accordingly. In the remainder of this section, we analyze the insights of these results in detail.

In our tasks, FKGL measures the sentence length and the word length as two factors for evaluating a simplified sentence. Therefore, we include *Wlen(Word Length)* and *Slen(Sentence Length)* into our analysis. As shown in Table 2, models with higher numbers of layers and/or heads do generally reduce the average word length and the average sentence length, which indicates that the higher number of layers and/or heads in the model leads to simpler outcomes.

It has been found that SARI correlates most closely to human judgment (Xu et al., 2016). To further analyze the effects of SARI, we study the impacts of three rewrite operations in SARI: add, delete, and keep. As shown in Table 2, we find that the improvement mostly results from correctly adding simplified words and deleting normal words, but not from keeping words. By analyzing the outputs, the increased number of layers or heads results in better capability to simplify the words. Specifically, models with the greater number of layers or heads tend to remove the normal words and add simplified words. However, they may introduce inaccurate simplified words, thereby driving down the F1 score for keeping words. We believe the Simple PPDB, which offers guidance about whether words need to be simplified and how they should be simplified, provides an ideal method to alleviate this issue.

4.2 Impacts of Integrating the Simple PPDB

In order to make comprehensive comparisons with the state-of-the-art models, we include multiple baselines from the literature, including PBMT-R (Wubben et al., 2012), Hybrid (Narayan and Gardent, 2014), and SBMT-SARI (Xu et al., 2016). We also include several strong baselines based on neural networks such as RNN/LSTM, DRESS, DRESS-LS (Zhang and Lapata, 2017) as shown in Tables 3 and 4 We developed three models for this experiment. They are DMASS, DCSS, and DMASS+DCSS, where DMASS+DCSS indicates the combination of DMASS and DCSS. The

Model	FKGL	Factors in FKGL		SARI	F1 for operations in FKGL		
		WLen	SLen		Add	Delete	Keep
RNN/LSTM	8.67	1.34	21.68	35.66	3.00	28.95	75.03
Transformer (L1H5)	8.59	1.34	21.39	35.88	2.69	30.46	74.50
Transformer (L2H5)	8.11	1.33	20.52	36.88	3.48	33.26	73.91
Transformer (L3H5)	7.71	1.32	19.77	38.02	4.14	37.41	72.51
Transformer (L4H1)	7.49	1.31	19.41	37.88	4.05	37.35	72.23
Transformer (L4H2)	7.40	1.31	19.19	38.35	4.58	39.77	70.70
Transformer (L4H5)	7.22	1.30	19.00	38.84	4.78	41.19	70.53

Table 2: Comparison of transformers with different layers and heads of attention on Turk dataset

Model	FKGL	Factors in FKGL		SARI	F1 for operations of SARI			Rule Utilization		
		WLen	SLen		Add	Delete	Keep	Prec	Recall	F1
PBMT-R	8.35	1.30	22.08	38.56	5.73	36.93	73.02	14.60	22.29	15.01
Hybrid	4.71	1.28	13.38	31.40	5.49	45.48	46.86	10.62	7.61	7.62
SBMT-SARI	7.49	1.18	23.50	39.96	5.97	41.43	72.51	13.30	28.96	15.77
RNN/LSTM	8.67	1.34	21.68	35.66	3.00	28.95	75.03	13.67	14.83	11.65
DRESS	6.80	1.34	16.55	37.08	2.94	43.14	65.16	13.06	12.50	10.77
DRESS-LS	6.92	1.35	16.76	37.27	2.82	42.21	66.78	12.40	11.36	9.83
DMASS	7.41	1.29	20.00	39.81	5.04	41.94	72.46	17.97	25.54	18.12
DCSS	7.34	1.31	19.30	39.26	5.29	41.24	71.26	13.14	21.30	13.87
DMASS+DCSS	7.18	1.27	20.10	40.42	5.48	45.55	70.22	16.25	30.42	18.98
DMASS$_{beam=4}$	8.20	1.30	21.66	39.16	4.90	38.41	74.18	18.53	25.46	18.40
DCSS$_{beam=4}$	7.97	1.32	20.56	39.11	5.10	38.87	73.36	14.36	20.96	14.48
DMASS+DCSS$_{beam=4}$	7.93	1.28	21.49	40.34	5.73	42.55	72.74	18.55	31.56	20.81
DMASS$_{beam=8}$	8.23	1.30	21.68	39.15	4.95	37.80	74.69	18.44	25.34	18.32
DCSS$_{beam=8}$	7.97	1.32	20.56	39.11	5.10	38.87	73.36	14.37	20.96	14.80
DMASS+DCSS$_{beam=8}$	8.04	1.29	21.64	40.45	5.72	42.23	73.41	19.46	31.99	21.51

Table 3: Performance of baselines and proposed models on the Turk dataset.

Model	FKGL	Factors in FKGL		SARI	F1 for operations of SARI			Rule Utilization		
		WLen	SLen		Add	Delete	Keep	Prec	Recall	F1
RNN/LSTM	6.09	1.22	18.67	21.09	11.10	38.78	13.39	12.62	22.63	14.68
DRESS	4.96	1.23	15.27	25.70	10.65	52.59	13.86	12.56	17.88	13.28
DRESS-LS	5.07	1.24	15.47	24.91	11.21	49.74	13.76	12.61	17.50	13.42
DMASS	5.38	1.20	17.47	25.41	11.88	50.39	13.97	16.32	34.79	20.00
DCSS	5.64	1.22	17.58	24.31	13.52	45.60	13.81	15.20	30.38	18.39
DMASS+DCSS	5.17	1.18	17.60	27.28	11.56	56.10	14.19	15.98	40.64	20.98
DMASS$_{beam=4}$	5.64	1.21	17.79	24.09	13.96	44.47	13.85	17.40	35.97	21.37
DCSS$_{beam=4}$	5.80	1.22	17.85	23.28	15.28	40.76	13.81	16.77	31.81	20.06
DMASS+DCSS$_{beam=4}$	5.42	1.19	17.81	26.39	13.92	51.13	14.13	18.71	43.36	24.23
DMASS$_{beam=8}$	5.68	1.21	17.83	23.95	14.25	43.74	13.86	17.69	36.37	21.74
DCSS$_{beam=8}$	5.77	1.22	17.76	23.18	15.65	40.08	13.82	17.18	32.18	20.50
DMASS+DCSS$_{beam=8}$	5.43	1.19	17.83	26.29	14.08	50.62	14.17	18.89	43.54	24.47

Table 4: Performance of baselines and proposed models on the Newsela dataset.

subscript *beam* indicates the size of beam search.

Results with FKGL Metric As shown in Tables 3 and 4, Hybrid achieves the lowest (thus the best) FKGL score, and DRESS and DRESS-LS have the second best FKGL scores. All the other models including ours do not perform as well as these two. But FKGL measures the simplicity of a sentence without considering the ground truth simplification references, so high FKGL may be at the cost of losing information and readability.

To further analyze the FKGL results, we examine the average sentence length and word length of the outcomes of the models and they are listed as WLen (Word Length) and SLen (Sentence Length) in Tables 3 and 4. Hybrid, DRESS, and DRESS-LS are good at generating shorter sentences, but they are not as good at choosing shorter words. In contrast, SBMT-SARI, DCSS, and DMASS all generate shorter words. Therefore, we believe that, by optimizing language model as a goal for the reinforcement learning, DRESS and DRESS-LS are tuned to simplify sentences by shortening the sentence lengths. In contrast, with the help of an integrated external knowledge base, SBMT-SARI and our models have more capability to generate shorter words in order to simplify sentences. Therefore, these two sets of models complete sentence simplification tasks via different routes, and perhaps there should be an exploration of combining these two routes for even more successful sentence simplification.

Another interesting finding is that the larger beam search size increases average word length slightly. This is because the larger beam search size mitigates the issue of the inaccurate simplification so that fewer words are simplified. To measure the correctness of simplification, we analyze the SARI metric and Rule Utilization.

Results with SARI Metric SARI is the most reliable metric for the sentence simplification task (Xu et al., 2016), therefore we would like to present more detailed discussion regarding the SARI results. As shown in Tables 3 and 4, DMASS+DCSS achieves the best SARI score, which demonstrates the effectiveness of integrating the knowledge base Simple PPDB for sentence simplification.

To further examine the impacts of the F1 scores for three operations in calculating the SARI scores, as shown in Tables 3 and 4, DMASS+DCSS, as well as other models with high SARI performance benefit greatly by correctly adding and deleting words. We believe these benefits mostly result from the integration with the knowledge base, which provides reliable guidance about which words to modify. SBMT-SARI, which represents a previous state-of-the-art model that also integrates with knowledge bases, performs best in correctly adding new words but performs inferiorly in deleting/keeping words. By analyzing the outputs, SBMT-SARI acts aggressively to simplify as many words as possible. But it also results in incorrect simplification. DRESS and DRESS-LS are inclined to generate the shorter sentence, which leads to high F1 scores for deleting words, but it lags behind other models in adding/keeping words.

DMASS leverages an additional memory component to maintain the simplification rules; DCSS uses internal memory to store those rules. A large number of simplification rules might confuse the model with limited internal memory. This might be the reason why DMASS works better than DCSS. By taking a two-way advantage of both models, DMASS+DCSS takes a two-fisted approach to store the simplification rules in both additional and internal memory. As a result, DMASS+DCSS achieves the best performance in SARI.

Results with Rule Utilization In this section, we evaluate the models' capabilities for word transformation. The majority of previous approaches, except for the SBMT-SARI, perform poorly in recall. We believe the knowledge base Simple PPDB will reduce uncertainty in the word selection.

As before, SBMT-SARI acts aggressively to simplify every word in the sentence. Such an aggressive action leads to relatively high performance in recall. However, it does not achieve a strong performance in precision. DMASS performs better in terms of rule utilization as compared to DCSS by leveraging an additional memory. DMASS+DCSS takes advantage of both approaches that store the simplification rules in additional and internal memory. This combined model is guaranteed to apply more accurate rules.

As compared to the loose relationship between SARI and beam search size, we find that that beam search size correlates strongly with the performance in rule utilization. Thus, we believe larger beam search size contributes to good coverage of rules to be applied as well as accuracy in applying rules.

5 Conclusion

In this paper, we propose two innovative approaches for sentence simplification based on neural networks. Both approaches are based on multi-layer and multi-head attention architecture and integrated with the Simple PPDB, an external sentence simplification knowledge base, in different ways. By conducting a set of experiments, we demonstrate that the proposed models perform better than existing methods and achieve new state-of-the-art in sentence simplification. Our experiments firstly prove that the multi-layer and multi-head attention architecture has an excellent capability to understand the text by accurately selecting specific words in a normal sentence and then choosing right simplified words. Secondly, by integrating with the knowledge base, our models outperform multiple state-of-the-art baselines for sentence simplification. Compared to previous models which integrated with the knowledge base, our models, especially, DMASS+DCSS, provide both good coverage of rules to be applied and accuracy in applying the correct rules. In future, we would like to investigate deeper into the different effects of additional memory and internal memory.

6 Acknowledge

This research was supported in part by the University of Pittsburgh Center for Research Computing through the resources provided. The research is funded in part by grants the National Institute on Disability, Independent Living, and Rehabilitation Research (NIDILRR) #90RE5018 and #90DP0064, and by Pittsburgh Health Data Alliance's "CARE" Project.

References

John A Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *EACL*, pages 269–270.

Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. *arXiv preprint arXiv:1702.04521*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Yang Feng, Shiyue Zhang, Andi Zhang, Dong Wang, and Andrew Abel. 2017. Memory-augmented neural machine translation. *arXiv preprint arXiv:1708.02005*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *ACL (1)*, pages 1537–1546.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 435–445.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Ellie Pavlick and Chris Callison-Burch. 2016. Simple ppdb: A paraphrase database for simplification. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 143.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Luz Rello, Clara Bayarri, Azuki Gòrriz, Ricardo Baeza-Yates, Saurabh Gupta, Gaurang Kanvinde, Horacio Saggion, Stefan Bott, Roberto Carlini, and Vasile Topac. 2013. Dyswebxia 2.0!: more accessible text for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, page 25. ACM.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2017. Learning to remember translation history with a continuous cache. *arXiv preprint arXiv:1711.09367*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: reading assistance for low-literacy readers. In *Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36. ACM.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the conference on empirical methods in natural language processing*, pages 409–420. Association for Computational Linguistics.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015a. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015b. Problems in current text simplification research: New data can help. *Transactions of the Association of Computational Linguistics*, 3(1):283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931*.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1353–1361. Association for Computational Linguistics.

Learning Neural Templates for Text Generation

Sam Wiseman **Stuart M. Shieber** **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA
{swiseman,shieber,srush}@seas.harvard.edu

Abstract

While neural, encoder-decoder models have had significant empirical success in text generation, there remain several unaddressed problems with this style of generation. Encoder-decoder models are largely (a) uninterpretable, and (b) difficult to control in terms of their phrasing or content. This work proposes a neural generation system using a hidden semi-markov model (HSMM) decoder, which learns latent, discrete templates jointly with learning to generate. We show that this model learns useful templates, and that these templates make generation both more interpretable and controllable. Furthermore, we show that this approach scales to real data sets and achieves strong performance nearing that of encoder-decoder text generation models.

1 Introduction

With the continued success of encoder-decoder models for machine translation and related tasks, there has been great interest in extending these methods to build general-purpose, data-driven natural language generation (NLG) systems (Mei et al., 2016; Dušek and Jurcıcek, 2016; Lebret et al., 2016; Chisholm et al., 2017; Wiseman et al., 2017). These encoder-decoder models (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015) use a neural encoder model to represent a source knowledge base, and a decoder model to emit a textual description word-by-word, conditioned on the source encoding. This style of generation contrasts with the more traditional division of labor in NLG, which famously emphasizes addressing the two questions of "what to say" and "how to say it" separately, and which leads to systems with explicit content selection, macro- and micro-planning, and surface realization components (Reiter and Dale, 1997; Jurafsky and Martin, 2014).

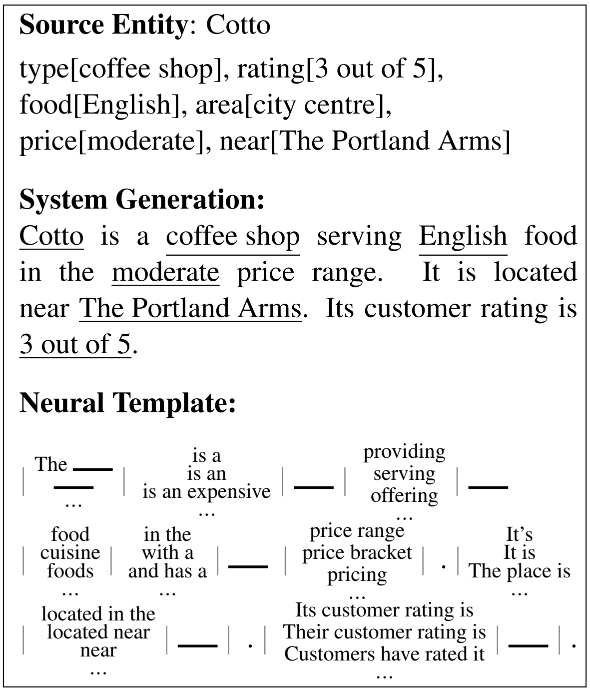

Source Entity: Cotto

type[coffee shop], rating[3 out of 5], food[English], area[city centre], price[moderate], near[The Portland Arms]

System Generation:

Cotto is a coffee shop serving English food in the moderate price range. It is located near The Portland Arms. Its customer rating is 3 out of 5.

Neural Template:

Figure 1: An example template-like generation from the E2E Generation dataset (Novikova et al., 2017). Knowledge base x (top) contains 6 records, and $\hat{y}$ (middle) is a system generation; records are shown as `type[value]`. An induced neural template (bottom) is learned by the system and employed in generating $\hat{y}$. Each cell represents a segment in the learned segmentation, and "blanks" show where slots are filled through copy attention during generation.

Encoder-decoder generation systems appear to have increased the fluency of NLG outputs, while reducing the manual effort required. However, due to the black-box nature of generic encoder-decoder models, these systems have also largely sacrificed two important desiderata that are often found in more traditional systems, namely (a) interpretable outputs that (b) can be easily controlled in terms of form and content.

This work considers building interpretable and controllable neural generation systems, and proposes a specific first step: a new data-driven generation model for learning discrete, *template-like*

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3174–3187
Brussels, Belgium, October 31 - November 4, 2018.

structures for conditional text generation. The core system uses a novel, neural hidden semi-markov model (HSMM) decoder, which provides a principled approach to template-like text generation. We further describe efficient methods for training this model in an entirely data-driven way by backpropagation through inference. Generating with the template-like structures induced by the neural HSMM allows for the explicit representation of what the system intends to say (in the form of a learned template) and how it is attempting to say it (in the form of an instantiated template).

We show that we can achieve performance competitive with other neural NLG approaches, while making progress satisfying the above two desiderata. Concretely, our experiments indicate that we can induce explicit templates (as shown in Figure 1) while achieving competitive automatic scores, and that we can control and interpret our generations by manipulating these templates. Finally, while our experiments focus on the data-to-text regime, we believe the proposed methodology represents a compelling approach to learning discrete, latent-variable representations of conditional text.

2 Related Work

A core task of NLG is to generate textual descriptions of knowledge base records. A common approach is to use hand-engineered templates (Kukich, 1983; McKeown, 1992; McRoy et al., 2000), but there has also been interest in creating templates in an automated manner. For instance, many authors induce templates by clustering sentences and then abstracting templated fields with hand-engineered rules (Angeli et al., 2010; Kondadadi et al., 2013; Howald et al., 2013), or with a pipeline of other automatic approaches (Wang and Cardie, 2013).

There has also been work in incorporating probabilistic notions of templates into generation models (Liang et al., 2009; Konstas and Lapata, 2013), which is similar to our approach. However, these approaches have always been conjoined with discriminative classifiers or rerankers in order to actually accomplish the generation (Angeli et al., 2010; Konstas and Lapata, 2013). In addition, these models explicitly model knowledge base field selection, whereas the model we present is fundamentally an end-to-end model over generation segments.

Recently, a new paradigm has emerged around neural text generation systems based on machine translation (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015). Most of this work has used unconstrained black-box encoder-decoder approaches. There has been some work on discrete variables in this context, including extracting representations (Shen et al., 2018), incorporating discrete latent variables in text modeling (Yang et al., 2018), and using non-HSMM segmental models for machine translation or summarization (Yu et al., 2016; Wang et al., 2017; Huang et al., 2018). Dai et al. (2017) develop an approximate inference scheme for a neural HSMM using RNNs for continuous emissions; in contrast we maximize the exact log-marginal, and use RNNs to parameterize a discrete emission distribution. Finally, there has also been much recent interest in segmental RNN models for non-generative tasks in NLP (Tang et al., 2016; Kong et al., 2016; Lu et al., 2016).

The neural text generation community has also recently been interested in "controllable" text generation (Hu et al., 2017), where various aspects of the text (often sentiment) are manipulated or transferred (Shen et al., 2017; Zhao et al., 2018; Li et al., 2018). In contrast, here we focus on controlling either the content of a generation or the way it is expressed by manipulating the (latent) template used in realizing the generation.

3 Overview: Data-Driven NLG

Our focus is on generating a textual description of a knowledge base or meaning representation. Following standard notation (Liang et al., 2009; Wiseman et al., 2017), let $x = \{r_1 \dots r_J\}$ be a collection of records. A record is made up of a *type* ($r.t$), an *entity* ($r.e$), and a *value* ($r.m$). For example, a knowledge base of restaurants might have a record with $r.t$ = `Cuisine`, $r.e$ = `Denny's`, and $r.m$ = `American`. The aim is to generate an adequate and fluent text description $\hat{y}_{1:T} = \hat{y}_1, \dots, \hat{y}_T$ of x. Concretely, we consider the E2E Dataset (Novikova et al., 2017) and the WikiBio Dataset (Lebret et al., 2016). We show an example E2E knowledge base x in the top of Figure 1. The top of Figure 2 shows an example knowledge base x from the WikiBio dataset, where it is paired with a *reference* text $y = y_{1:T}$ at the bottom.

The dominant approach in neural NLG has been

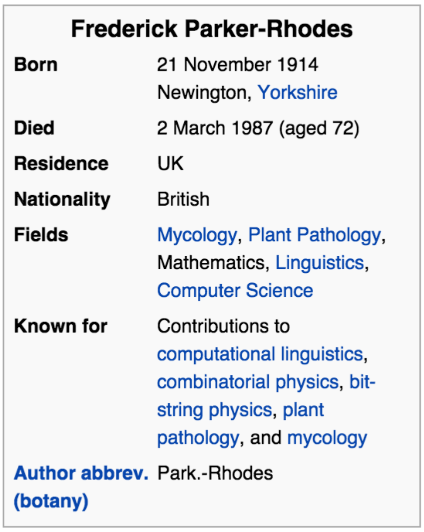

Frederick Parker-Rhodes

Born	21 November 1914 Newington, Yorkshire
Died	2 March 1987 (aged 72)
Residence	UK
Nationality	British
Fields	Mycology, Plant Pathology, Mathematics, Linguistics, Computer Science
Known for	Contributions to computational linguistics, combinatorial physics, bit-string physics, plant pathology, and mycology
Author abbrev. (botany)	Park.-Rhodes

Frederick Parker-Rhodes (21 March 1914 - 21 November 1987) was an English linguist, plant pathologist, computer scientist, mathematician, mystic, and mycologist.

Figure 2: An example from the WikiBio dataset (Lebret et al., 2016), with a database x (top) for Frederick Parker-Rhodes and corresponding reference generation y (bottom).

to use an encoder network over x and then a conditional decoder network to generate y, training the whole system in an end-to-end manner. To generate a description for a given example, a black-box network (such as an RNN) is used to produce a distribution over the next word, from which a choice is made and fed back into the system. The entire distribution is driven by the internal states of the neural network.

While effective, relying on a neural decoder makes it difficult to understand what aspects of x are correlated with a particular system output. This leads to problems both in controlling fine-grained aspects of the generation process and in interpreting model mistakes.

As an example of why controllability is important, consider the records in Figure 1. Given these inputs an end-user might want to generate an output meeting specific constraints, such as not mentioning any information relating to customer rating. Under a standard encoder-decoder style model, one could filter out this information either from the encoder or decoder, but in practice this would lead to unexpected changes in output that might propagate through the whole system.

As an example of the difficulty of interpreting mistakes, consider the following actual generation from an encoder-decoder style system for the records in Figure 2: ”frederick parker-rhodes (21 november 1914 - 2 march 1987) was an english mycology and plant pathology, mathematics at the university of uk.” In addition to not being fluent, it is unclear what the end of this sentence is even attempting to convey: it may be attempting to convey a fact not actually in the knowledge base (e.g., where Parker-Rhodes studied), or perhaps it is simply failing to fluently realize information that *is* in the knowledge base (e.g., Parker-Rhodes's country of residence).

Traditional NLG systems (Kukich, 1983; McKeown, 1992; Belz, 2008; Gatt and Reiter, 2009), in contrast, largely avoid these problems. Since they typically employ an explicit planning component, which decides which knowledge base records to focus on, and a surface realization component, which realizes the chosen records, the intent of the system is always explicit, and it may be modified to meet constraints.

The goal of this work is to propose an approach to neural NLG that addresses these issues in a principled way. We target this goal by proposing a new model that generates with template-like objects induced by a neural HSMM (see Figure 1). Templates are useful here because they represent a fixed plan for the generation's content, and because they make it clear what part of the generation is associated with which record in the knowledge base.

4 Background: Semi-Markov Models

What does it mean to learn a template? It is natural to think of a template as a sequence of typed text-segments, perhaps with some segments acting as the template's "backbone" (Wang and Cardie, 2013), and the remaining segments filled in from the knowledge base.

A natural probabilistic model conforming with this intuition is the hidden semi-markov model (HSMM) (Gales and Young, 1993; Ostendorf et al., 1996), which models latent segmentations in an output sequence. Informally, an HSMM is much like an HMM, except emissions may last multiple time-steps, and multi-step emissions need not be independent of each other conditioned on the state.

We briefly review HSMMs following Murphy (2002). Assume we have a sequence of observed tokens $y_1 \dots y_T$ and a discrete, latent state $z_t \in \{1, \dots, K\}$ for each timestep. We additionally use two per-timestep variables to model multi-step segments: a length variable $l_t \in \{1, \ldots, L\}$ specifying the length of the current segment, and a deterministic binary variable f_t indicating whether a segment finishes at time t. We will consider in particular *conditional* HSMMs, which condition on a source x, essentially giving us an HSMM decoder.

An HSMM specifies a joint distribution on the observations and latent segmentations. Letting θ denote all the parameters of the model, and using the variables introduced above, we can write the corresponding joint-likelihood as follows

$$p(y, z, l, f \mid x; \theta) = \prod_{t=0}^{T-1} p(z_{t+1}, l_{t+1} \mid z_t, l_t, x)^{f_t} \times \prod_{t=1}^{T} p(y_{t-l_t+1:t} \mid z_t, l_t, x)^{f_t},$$

where we take z_0 to be a distinguished start-state, and the deterministic f_t variables are used for excluding non-segment log probabilities. We further assume $p(z_{t+1}, l_{t+1} \mid z_t, l_t, x)$ factors as $p(z_{t+1} \mid z_t, x) \times p(l_{t+1} \mid z_{t+1})$. Thus, the likelihood is given by the product of the probabilities of each discrete state transition made, the probability of the length of each segment given its discrete state, and the probability of the observations in each segment, given its state and length.

5 A Neural HSMM Decoder

We use a novel, neural parameterization of an HSMM to specify the probabilities in the likelihood above. This full model, sketched out in Figure 3, allows us to incorporate the modeling components, such as LSTMs and attention, that make neural text generation effective, while maintaining the HSMM structure.

5.1 Parameterization

Since our model must condition on x, let $\boldsymbol{r}_j \in \mathbb{R}^d$ represent a real embedding of record $r_j \in x$, and let $\boldsymbol{x}_a \in \mathbb{R}^d$ represent a real embedding of the entire knowledge base x, obtained by max-pooling coordinate-wise over all the $\boldsymbol{r}_j$. It is also useful to have a representation of just the unique *types* of records that appear in x, and so we also define $\boldsymbol{x}_u \in \mathbb{R}^d$ to be the sum of the embeddings of the unique types appearing in x, plus a bias vector and followed by a ReLU nonlinearity.

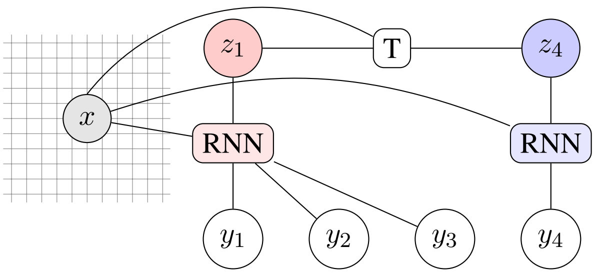

Figure 3: HSMM factor graph (under a known segmentation) to illustrate parameters. Here we assume z_1 is in the "red" state (out of K possibilities), and transitions to the "blue" state after emitting three words. The transition model, shown as T, is a function of the two states and the neural encoded source x. The emission model is a function of a "red" RNN model (with copy attention over x) that generates words 1, 2 and 3. After transitioning, the next word y_4 is generated by the "blue" RNN, but independently of the previous words.

Transition Distribution The transition distribution $p(z_{t+1} \mid z_t, x)$ may be viewed as a $K \times K$ matrix of probabilities, where each row sums to 1. We define this matrix to be

$$p(z_{t+1} \mid z_t, x) \propto \boldsymbol{A}\boldsymbol{B} + \boldsymbol{C}(\boldsymbol{x}_u)\boldsymbol{D}(\boldsymbol{x}_u),$$

where $\boldsymbol{A} \in \mathbb{R}^{K \times m_1}$, $\boldsymbol{B} \in \mathbb{R}^{m_1 \times K}$ are state embeddings, and where $\boldsymbol{C} : \mathbb{R}^d \rightarrow \mathbb{R}^{K \times m_2}$ and $\boldsymbol{D} : \mathbb{R}^d \rightarrow \mathbb{R}^{m_2 \times K}$ are parameterized non-linear functions of $\boldsymbol{x}_u$. We apply a row-wise $\mathrm{softmax}$ to the resulting matrix to obtain the desired probabilities.

Length Distribution We simply fix all length probabilities $p(l_{t+1} \mid z_{t+1})$ to be uniform up to a maximum length L.[1]

Emission Distribution The emission model models the generation of a text segment conditioned on a latent state and source information, and so requires a richer parameterization. Inspired by the models used for neural NLG, we base this model on an RNN decoder, and write a segment's probability as a product over token-level probabilities,

$$p(y_{t-l_t+1:t} \mid z_t = k, l_t = l, x) = \prod_{i=1}^{l_t} p(y_{t-l_t+i} \mid y_{t-l_t+1:t-l_t+i-1}, z_t = k, x) \times p(\text{</seg>} \mid y_{t-l_t+1:t}, z_t = k, x) \times \mathbf{1}_{\{l_t = l\}},$$

---

[1] We experimented with parameterizing the length distribution, but found that it led to inferior performance. Forcing the length probabilities to be uniform encourages the model to cluster together functionally similar emissions of different lengths, while parameterizing them can lead to states that specialize to specific emission lengths.

where </seg> is an end of segment token. The RNN decoder uses attention and copy-attention over the embedded records $\boldsymbol{r}_j$, and is conditioned on $z_t = k$ by concatenating an embedding corresponding to the k'th latent state to the RNN's input; the RNN is also conditioned on the entire x by initializing its hidden state with $\boldsymbol{x}_a$.

More concretely, let $\boldsymbol{h}^k_{i-1} \in \mathbb{R}^d$ be the state of an RNN conditioned on x and $z_t = k$ (as above) run over the sequence $y_{t-l_t+1:t-l_t+i-1}$. We let the model attend over records $\boldsymbol{r}_j$ using $\boldsymbol{h}^k_{i-1}$ (in the style of Luong et al. (2015)), producing a context vector $\boldsymbol{c}^k_{i-1}$. We may then obtain scores $\boldsymbol{v}_{i-1}$ for each word in the output vocabulary,

$$\boldsymbol{v}_{i-1} = \boldsymbol{W} \tanh(\boldsymbol{g}^k_1 \circ [\boldsymbol{h}^k_{i-1}, \boldsymbol{c}^k_{i-1}]),$$

with parameters $\boldsymbol{g}^k_1 \in \mathbb{R}^{2d}$ and $\boldsymbol{W} \in \mathbb{R}^{V \times 2d}$. Note that there is a $\boldsymbol{g}^k_1$ vector for each of K discrete states. To additionally implement a kind of slot filling, we allow emissions to be directly copied from the value portion of the records r_j using copy attention (Gülçehre et al., 2016; Gu et al., 2016; Yang et al., 2016). Define copy scores,

$$\rho_j = \boldsymbol{r}_j^{\mathsf{T}} \tanh(\boldsymbol{g}^k_2 \circ \boldsymbol{h}^k_{i-1}),$$

where $\boldsymbol{g}^k_2 \in \mathbb{R}^d$. We then normalize the output-vocabulary and copy scores together, to arrive at

$$\widetilde{\boldsymbol{v}}_{i-1} = \mathrm{softmax}([\boldsymbol{v}_{i-1}, \rho_1, \ldots, \rho_J]),$$

and thus

$$p(y_{t-l_t+i} = w \mid y_{t-l_t+1:t-l_t+i-1}, z_t = k, x) = \widetilde{\boldsymbol{v}}_{i-1,w} + \sum_{j: r_j.m = w} \widetilde{\boldsymbol{v}}_{i-1,V+j}.$$

An Autoregressive Variant The model as specified assumes segments are independent conditioned on the associated latent state and x. While this assumption still allows for reasonable performance, we can tractably allow interdependence between tokens (but not segments) by having each next-token distribution depend on all the previously generated tokens, giving us an autoregressive HSMM. For this model, we will in fact use $p(y_{t-l_t+i} = w \mid y_{1:t-l_t+i-1}, z_t = k, x)$ in defining our emission model, which is easily implemented by using an additional RNN run over all the preceding tokens. We will report scores for both non-autoregressive and autoregressive HSMM decoders below.

5.2 Learning

The model requires fitting a large set of neural network parameters. Since we assume z, l, and f are unobserved, we marginalize over these variables to maximize the log marginal-likelihood of the observed tokens y given x. The HSMM marginal-likelihood calculation can be carried out efficiently with a dynamic program analogous to either the forward- or backward-algorithm familiar from HMMs (Rabiner, 1989).

It is actually more convenient to use the backward-algorithm formulation when using RNNs to parameterize the emission distributions, and we briefly review the backward recurrences here, again following Murphy (2002). We have:

$$\begin{aligned}
\beta_t(j) &= p(y_{t+1:T} \mid z_t = j, f_t = 1, x) \\
&= \sum_{k=1}^{K} \beta^*_t(k)\, p(z_{t+1} = k \mid z_t = j) \\
\beta^*_t(k) &= p(y_{t+1:T} \mid z_{t+1} = k, f_t = 1, x) \\
&= \sum_{l=1}^{L} \Big[\beta_{t+l}(k)\, p(l_{t+1} = l \mid z_{t+1} = k) \\
&\qquad p(y_{t+1:t+l} \mid z_{t+1} = k, l_{t+1} = l) \Big],
\end{aligned}$$

with base case $\beta_T(j) = 1$. We can now obtain the marginal probability of y as $p(y \mid x) = \sum_{k=1}^{K} \beta^*_0(k)\, p(z_1 = k)$, where we have used the fact that f_0 must be 1, and we therefore train to maximize the log-marginal likelihood of the observed y:

$$\ln p(y \mid x; \theta) = \ln \sum_{k=1}^{K} \beta^*_0(k)\, p(z_1 = k). \tag{1}$$

Since the quantities in (1) are obtained from a dynamic program, which is itself differentiable, we may simply maximize with respect to the parameters θ by back-propagating through the dynamic program; this is easily accomplished with automatic differentiation packages, and we use `pytorch` (Paszke et al., 2017) in all experiments.

5.3 Extracting Templates and Generating

After training, we could simply condition on a new database and generate with beam search, as is standard with encoder-decoder models. However, the structured approach we have developed allows us to generate in a more template-like way, giving us more interpretable and controllable generations.

[The Golden Palace]$_{55}$ [is a]$_{59}$ [coffee shop]$_{12}$ [providing]$_{3}$ [Indian]$_{50}$ [food]$_{1}$ [in the]$_{17}$ [£20-25]$_{26}$ [price range]$_{16}$ [.]$_{2}$ [It is]$_{8}$ [located in the]$_{25}$ [riverside]$_{40}$ [.]$_{53}$ [Its customer rating is]$_{19}$ [high]$_{23}$ [.]$_{2}$

Figure 4: A sample Viterbi segmentation of a training text; subscripted numbers indicate the corresponding latent state. From this we can extract a template with $S = 17$ segments; compare with the template used at the bottom of Figure 1.

First, note that given a database x and reference generation y we can obtain the MAP assignment to the variables z, l, and f with a dynamic program similar to the Viterbi algorithm familiar from HMMs. These assignments will give us a typed segmentation of y, and we show an example Viterbi segmentation of some training text in Figure 4. Computing MAP segmentations allows us to associate text-segments (i.e., phrases) with the discrete labels z_t that frequently generate them. These MAP segmentations can be used in an exploratory way, as a sort of dimensionality reduction of the generations in the corpus. More importantly for us, however, they can also be used to guide generation.

In particular, since each MAP segmentation implies a sequence of hidden states z, we may run a *template extraction* step, where we collect the most common "templates" (i.e., sequences of hidden states) seen in the training data. Each "template" $z^{(i)}$ consists of a sequence of latent states, with $z^{(i)} = z_1^{(i)}, \ldots z_S^{(i)}$ representing the S distinct segments in the i'th extracted template (recall that we will technically have a z_t for each time-step, and so $z^{(i)}$ is obtained by collapsing adjacent z_t's with the same value); see Figure 4 for an example template (with $S = 17$) that can be extracted from the E2E corpus. The bottom of Figure 1 shows a visualization of this extracted template, where discrete states are replaced by the phrases they frequently generate in the training data.

With our templates $z^{(i)}$ in hand, we can then restrict the model to using (one of) them during generation. In particular, given a new input x, we may generate by computing

$$\hat{y}^{(i)} = \arg\max_{y'} p(y', z^{(i)} \mid x), \qquad (2)$$

which gives us a generation $\hat{y}^{(i)}$ for each extracted template $z^{(i)}$. For example, the generation in Figure 1 is obtained by maximizing (2) with x set to the database in Figure 1 and $z^{(i)}$ set to the template extracted in Figure 4. In practice, the $\arg\max$ in (2) will be intractable to calculate exactly due to the use of RNNs in defining the emission distribution, and so we approximate it with a constrained beam search. This beam search looks very similar to that typically used with RNN decoders, except the search occurs only over a segment, for a particular latent state k.

5.4 Discussion

Returning to the discussion of controllability and interpretability, we note that with the proposed model (a) it is possible to explicitly force the generation to use a chosen template $z^{(i)}$, which is itself automatically learned from training data, and (b) that every *segment* in the generated $\hat{y}^{(i)}$ is typed by its corresponding latent variable. We explore these issues empirically in Section 7.1.

We also note that these properties may be useful for other text applications, and that they offer an additional perspective on how to approach latent variable modeling for text. Whereas there has been much recent interest in learning continuous latent variable representations for text (see Section 2), it has been somewhat unclear what the latent variables to be learned are intended to capture. On the other hand, the latent, template-like structures we induce here represent a plausible, probabilistic latent variable story, and allow for a more controllable method of generation.

Finally, we highlight one significant possible issue with this model – the assumption that segments are independent of each other given the corresponding latent variable and x. Here we note that the fact that we are allowed to condition on x is quite powerful. Indeed, a clever encoder could capture much of the necessary interdependence between the segments to be generated (e.g., the correct determiner for an upcoming noun phrase) in its encoding, allowing the segments themselves to be decoded more or less independently, given x.

6 Data and Methods

Our experiments apply the approach outlined above to two recent, data-driven NLG tasks.

6.1 Datasets

Experiments use the E2E (Novikova et al., 2017) and WikiBio (Lebret et al., 2016) datasets, examples of which are shown in Figures 1 and 2, respectively. The former dataset, used for the

2018 E2E-Gen Shared Task, contains approximately 50K total examples, and uses 945 distinct word types, and the latter dataset contains approximately 500K examples and uses approximately 400K word types. Because our emission model uses a word-level copy mechanism, any record with a phrase consisting of n words as its value is replaced with n positional records having a single word value, following the preprocessing of Lebret et al. (2016). For example, "type[coffee shop]" in Figure 1 becomes "type-1[coffee]" and "type-2[shop]."

For both datasets we compare with published encoder-decoder models, as well as with direct template-style baselines. The E2E task is evaluated in terms of BLEU (Papineni et al., 2002), NIST (Belz and Reiter, 2006), ROUGE (Lin, 2004), CIDEr (Vedantam et al., 2015), and METEOR (Banerjee and Lavie, 2005).[2] The benchmark system for the task is an encoder-decoder style system followed by a reranker, proposed by Dušek and Jurcıcek (2016). We compare to this baseline, as well as to a simple but competitive non-parametric template-like baseline ("SUB" in tables), which selects a training sentence with records that maximally overlap (without including extraneous records) the unseen set of records we wish to generate from; ties are broken at random. Then, word-spans in the chosen training sentence are aligned with records by string-match, and replaced with the corresponding fields of the new set of records.[3]

The WikiBio dataset is evaluated in terms of BLEU, NIST, and ROUGE, and we compare with the systems and baselines implemented by Lebret et al. (2016), which include two neural, encoder-decoder style models, as well as a Kneser-Ney, templated baseline.

6.2 Model and Training Details

We first emphasize two additional methodological details important for obtaining good performance.

Constraining Learning We were able to learn more plausible segmentations of y by constraining the model to respect word spans $y_{t+1:t+l}$ that appear in some record $r_j \in x$. We accomplish this by giving zero probability (within the backward recurrences in Section 5) to any segmentation that splits up a sequence $y_{t+1:t+l}$ that appears in some r_j, or that includes $y_{t+1:t+l}$ as a subsequence of another sequence. Thus, we maximize (1) subject to these hard constraints.

Increasing the Number of Hidden States While a larger K allows for a more expressive latent model, computing K emission distributions over the vocabulary can be prohibitively expensive. We therefore tie the emission distribution between multiple states, while allowing them to have a different transition distributions.

We give additional architectural details of our model in the Supplemental Material; here we note that we use an MLP to embed $\boldsymbol{r}_j \in \mathbb{R}^d$, and a 1-layer LSTM (Hochreiter and Schmidhuber, 1997) in defining our emission distributions. In order to reduce the amount of memory used, we restrict our output vocabulary (and thus the height of the matrix $\boldsymbol{W}$ in Section 5) to only contain words in y that are *not* present in x; any word in y present in x is assumed to be copied. In the case where a word y_t appears in a record r_j (and could therefore have been copied), the input to the LSTM at time $t+1$ is computed using information from r_j; if there are multiple r_j from which y_t could have been copied, the computed representations are simply averaged.

For all experiments, we set $d=300$ and $L=4$. At generation time, we select the 100 most common templates $z^{(i)}$, perform beam search with a beam of size 5, and select the generation with the highest overall joint probability.

For our E2E experiments, our best non-autoregressive model has 55 "base" states, duplicated 5 times, for a total of $K=275$ states, and our best autoregressive model uses $K=60$ states, without any duplication. For our WikiBio experiments, both our best non-autoregressive and autoregressive models uses 45 base states duplicated 3 times, for a total of $K=135$ states. In all cases, K was chosen based on BLEU performance on held-out validation data. Code implementing our models is available at `https://github.com/harvardnlp/neural-template-gen`.

7 Results

Our results on automatic metrics are shown in Tables 1 and 2. In general, we find that the templated baselines underperform neural models, whereas our proposed model is fairly competitive with neural models, and sometimes even out-

---

[2] We use the official E2E NLG Challenge scoring scripts at `https://github.com/tuetschek/e2e-metrics`.

[3] For categorical records, like "familyFriendly", which cannot easily be aligned with a phrase, we simply select only candidate training sentences with the same categorical value.

	BLEU	NIST	ROUGE	CIDEr	METEOR
			Validation		
D&J	69.25	8.48	72.57	2.40	47.03
SUB	43.71	6.72	55.35	1.41	37.87
NTemp	64.53	7.66	68.60	1.82	42.46
NTemp+AR	67.07	7.98	69.50	2.29	43.07
			Test		
D&J	65.93	8.59	68.50	2.23	44.83
SUB	43.78	6.88	54.64	1.39	37.35
NTemp	55.17	7.14	65.70	1.70	41.91
NTemp+AR	59.80	7.56	65.01	1.95	38.75

Table 1: Comparison of the system of Dušek and Jurcıcek (2016), which forms the baseline for the E2E challenge, a non-parametric, substitution-based baseline (see text), and our HSMM models (denoted "NTemp" and "NTemp+AR" for the non-autoregressive and autoregressive versions, resp.) on the validation and test portions of the E2E dataset. "ROUGE" is ROUGE-L. Models are evaluated using the official E2E NLG Challenge scoring scripts.

	BLEU	NIST	ROUGE-4
Template KN †	19.8	5.19	10.7
NNLM (field) †	33.4	7.52	23.9
NNLM (field & word) †	34.7	7.98	25.8
NTemp	34.2	7.94	35.9
NTemp+AR	34.8	7.59	38.6
Seq2seq (Liu et al., 2018)	43.65	-	40.32

Table 2: Top: comparison of the two best neural systems of Lebret et al. (2016), their templated baseline, and our HSMM models (denoted "NTemp" and "NTemp+AR" for the non-autoregressive and autoregressive versions, resp.) on the test portion of the WikiBio dataset. Models marked with a † are from Lebret et al. (2016), and following their methodology we use ROUGE-4. Bottom: state-of-the-art seq2seq-style results from Liu et al. (2018).

Travellers Rest Beefeater
name[Travellers Rest Beefeater], customerRating[3 out of 5], area[riverside], near[Raja Indian Cuisine]
1. [Travellers Rest Beefeater]$_{55}$ [is a]$_{59}$ [3 star]$_{43}$ [restaurant]$_{11}$ [located near]$_{25}$ [Raja Indian Cuisine]$_{40}$ [.]$_{53}$ 2. [Near]$_{31}$ [riverside]$_{29}$ [,]$_{44}$ [Travellers Rest Beefeater]$_{55}$ [serves]$_{3}$ [3 star]$_{50}$ [food]$_{1}$ [.]$_{2}$ 3. [Travellers Rest Beefeater]$_{55}$ [is a]$_{59}$ [restaurant]$_{12}$ [providing]$_{3}$ [riverside]$_{50}$ [food]$_{1}$ [and has a]$_{17}$ [3 out of 5]$_{26}$ [customer rating]$_{16}$ [.]$_{2}$ [It is]$_{8}$ [near]$_{25}$ [Raja Indian Cuisine]$_{40}$ [.]$_{53}$ 4. [Travellers Rest Beefeater]$_{55}$ [is a]$_{59}$ [place to eat]$_{12}$ [located near]$_{25}$ [Raja Indian Cuisine]$_{40}$ [.]$_{53}$ 5. [Travellers Rest Beefeater]$_{55}$ [is a]$_{59}$ [3 out of 5]$_{5}$ [rated]$_{32}$ [riverside]$_{43}$ [restaurant]$_{11}$ [near]$_{25}$ [Raja Indian Cuisine]$_{40}$ [.]$_{53}$

Table 3: Impact of varying the template $z^{(i)}$ for a single x from the E2E validation data; generations are annotated with the segmentations of the chosen $z^{(i)}$. Results were obtained using the NTemp+AR model from Table 1.

performs them. On the E2E data, for example, we see in Table 1 that the SUB baseline, despite having fairly impressive performance for a non-parametric model, fares the worst. The neural HSMM models are largely competitive with the encoder-decoder system on the validation data, despite offering the benefits of interpretability and controllability; however, the gap increases on test.

Table 2 evaluates our system's performance on the test portion of the WikiBio dataset, comparing with the systems and baselines implemented by Lebret et al. (2016). Again for this dataset we see that their templated Kneser-Ney model underperforms on the automatic metrics, and that neural models improve on these results. Here the HSMMs are competitive with the best model of Lebret et al. (2016), and even outperform it on ROUGE. We emphasize, however, that recent, sophisticated approaches to encoder-decoder style database-to-text generation have since surpassed the results of Lebret et al. (2016) and our own, and we show the recent seq2seq style results of Liu et al. (2018), who use a somewhat larger model, at the bottom of Table 2.

7.1 Qualitative Evaluation

We now qualitatively demonstrate that our generations are controllable and interpretable.

Controllable Diversity One of the powerful aspects of the proposed approach to generation is that we can manipulate the template $z^{(i)}$ while leaving the database x constant, which allows for easily controlling aspects of the generation. In Table 3 we show the generations produced by our model for five different neural template sequences $z^{(i)}$, while fixing x. There, the segments in each generation are annotated with the latent states determined by the corresponding $z^{(i)}$. We see that these templates can be used to affect the word-ordering, as well as which fields are mentioned in the generated text. Moreover, because the discrete states align with particular fields (see below), it is generally simple to automatically infer to which fields particular latent states correspond, allowing users to choose which template best meets their requirements. We emphasize that this level of controllability is much harder to obtain for encoder-decoder models, since, at best, a large amount of sampling would be required to avoid generating around a particular mode in the conditional distribution, and even then it would be difficult to control the sort of generations obtained.

kenny warren

name: kenny warren, **birth date:** 1 april 1946, **birth name:** kenneth warren deutscher, **birth place:** brooklyn, new york, **occupation:** ventriloquist, comedian, author, **notable work:** book - the revival of ventriloquism in america

1. [kenneth warren deutscher]$_{132}$ [( ]$_{75}$ [born]$_{89}$ [april 1, 1946]$_{101}$ [ )]$_{67}$ [is an american]$_{82}$ [author]$_{20}$ [and]$_{1}$ [ventriloquist and comedian]$_{69}$ [.]$_{88}$
2. [kenneth warren deutscher]$_{132}$ [( ]$_{75}$ [born]$_{89}$ [april 1, 1946]$_{101}$ [ )]$_{67}$ [is an american]$_{82}$ [author]$_{20}$ [best known for his]$_{95}$ [the revival of ventriloquism]$_{96}$ [.]$_{88}$
3. [kenneth warren]$_{16}$ [“kenny” warren]$_{117}$ [( ]$_{75}$ [born]$_{89}$ [april 1, 1946]$_{101}$ [ )]$_{67}$ [is an american]$_{127}$ [ventriloquist, comedian]$_{28}$ [.]$_{133}$
4. [kenneth warren]$_{16}$ [“kenny” warren]$_{117}$ [( ]$_{75}$ [born]$_{89}$ [april 1, 1946]$_{101}$ [ )]$_{67}$ [is a]$_{104}$ [new york]$_{98}$ [author]$_{20}$ [.]$_{133}$
5. [kenneth warren deutscher]$_{42}$ [is an american]$_{82}$ [ventriloquist, comedian]$_{118}$ [based in]$_{15}$ [brooklyn, new york]$_{84}$ [.]$_{88}$

Table 4: Impact of varying the template $z^{(i)}$ for a single x from the WikiBio validation data; generations are annotated with the segmentations of the chosen $z^{(i)}$. Results were obtained using the NTemp model from Table 2.

Interpretable States Discrete states also provide a method for interpreting the generations produced by the system, since each segment is explicitly typed by the current hidden state of the model. Table 4 shows the impact of varying the template $z^{(i)}$ for a single x from the WikiBio dataset. While there is in general surprisingly little stylistic variation in the WikiBio data itself, there is variation in the information discussed, and the templates capture this. Moreover, we see that particular discrete states correspond in a consistent way to particular pieces of information, allowing us to align states with particular field types. For instance, birth names have the same hidden state (132), as do names (117), nationalities (82), birth dates (101), and occupations (20).

To demonstrate empirically that the learned states indeed align with field types, we calculate the average purity of the discrete states learned for both datasets in Table 5. In particular, for each discrete state for which the majority of its generated words appear in some r_j, the *purity* of a state's record type alignment is calculated as the percentage of the state's words that come from the most frequent record type the state represents. This calculation was carried out over training examples that belonged to one of the top 100 most frequent templates. Table 5 indicates that discrete states learned on the E2E data are quite pure. Discrete states learned on the WikiBio data are less pure, though still rather impressive given that there are approximately 1700 record types represented in the WikiBio data, and we limit the number of states to 135. Unsurprisingly, adding autoregressiveness to the model decreases purity on both datasets, since the model may rely on the autoregressive RNN for typing, in addition to the state's identity.

	NTemp	NTemp+AR
E2E	89.2 (17.4)	85.4 (18.6)
WikiBio	43.2 (19.7)	39.9 (17.9)

Table 5: Empirical analysis of the average purity of discrete states learned on the E2E and WikiBio datasets, for the NTemp and NTemp+AR models. Average purities are given as percents, and standard deviations follow in parentheses. See the text for full description of this calculation.

8 Conclusion and Future Work

We have developed a neural, template-like generation model based on an HSMM decoder, which can be learned tractably by backpropagating through a dynamic program. The method allows us to extract template-like latent objects in a principled way in the form of state sequences, and then generate with them. This approach scales to large-scale text datasets and is nearly competitive with encoder-decoder models. More importantly, this approach allows for controlling the diversity of generation and for producing interpretable states during generation. We view this work both as the first step towards learning discrete latent variable template models for more difficult generation tasks, as well as a different perspective on learning latent variable text models in general. Future work will examine encouraging the model to learn maximally different (or minimal) templates, which our objective does not explicitly encourage, templates of larger textual phenomena, such as paragraphs and documents, and hierarchical templates.

Acknowledgments

SW gratefully acknowledges the support of a Siebel Scholars award. AMR gratefully acknowledges the support of NSF CCF-1704834, Intel Research, and Amazon AWS Research grants.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(04):431–455.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of nlg systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. *CoRR*, abs/1702.06235.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Hanjun Dai, Bo Dai, Yan-Ming Zhang, Shuang Li, and Le Song. 2017. Recurrent hidden semi-markov model. In *International Conference on Learning Representations*.

Ondrej Dušek and Filip Jurcıcek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 45.

Mark JF Gales and Steve J Young. 1993. *The theory of segmental hidden Markov models*. University of Cambridge, Department of Engineering.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *ACL*.

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

Blake Howald, Ravikumar Kondadadi, and Frank Schilder. 2013. Domain adaptable semantic clustering in statistical nlg. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*, pages 143–154.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. In *International Conference on Learning Representations*.

Dan Jurafsky and James H Martin. 2014. *Speech and language processing*. Pearson London.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1406–1415.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *International Conference on Learning Representations*.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *J. Artif. Intell. Res.(JAIR)*, 48:305–346.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *ACL*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, pages 1203–1213.

J. Li, R. Jia, H. He, and P. Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *North American Association for Computational Linguistics (NAACL)*.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL*, pages 91–99. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Liang Lu, Lingpeng Kong, Chris Dyer, Noah A Smith, and Steve Renals. 2016. Segmental recurrent neural networks for end-to-end speech recognition. *Interspeech 2016*, pages 385–389.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 1412–1421.

Kathleen McKeown. 1992. *Text generation - using discourse strategies and focus constraints to generate natural language text*. Studies in natural language processing. Cambridge University Press.

Susan W McRoy, Songsak Channarukul, and Syed S Ali. 2000. Yag: A template-based generator for real-time systems. In *Proceedings of the first international conference on Natural language generation-Volume 14*, pages 264–267. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT*, pages 720–730.

Kevin P Murphy. 2002. Hidden semi-markov models (hsmms). *unpublished notes*, 2.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany.

Mari Ostendorf, Vassilios V Digalakis, and Owen A Kimball. 1996. From hmm's to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Transactions on speech and audio processing*, 4(5):360–378.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NIPS 2017 Autodiff Workshop*.

Lawrence R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Hao Tang, Weiran Wang, Kevin Gimpel, and Karen Livescu. 2016. End-to-end training approaches for discriminative segmental models. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 496–502. IEEE.

Ke M Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Chong Wang, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. 2017. Sequence modeling via segmentations. In *International Conference on Machine Learning*, pages 3674–3683.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1395–1405.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. *CoRR*, abs/1611.01628.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1307–1316.

Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pages 5897–5906.

A Supplemental Material

A.1 Additional Model and Training Details

Computing $\boldsymbol{r}_j$ A record r_j is represented by embedding a feature for its type, its position, and its word value in $\mathbb{R}^d$, and applying an MLP with ReLU nonlinearity (Nair and Hinton, 2010) to form $\boldsymbol{r}_j \in \mathbb{R}^d$, similar to Yang et al. (2016) and Wiseman et al. (2017).

LSTM Details The initial cell and hidden-state values for the decoder LSTM are given by $\boldsymbol{Q}_1 \boldsymbol{x}_a$ and $\tanh(\boldsymbol{Q}_2 \boldsymbol{x}_a)$, respectively, where $\boldsymbol{Q}_1, \boldsymbol{Q}_2 \in \mathbb{R}^{d \times d}$.

When a word y_t appears in a record r_j, the input to the LSTM at time $t+1$ is computed using an MLP with ReLU nonlinearity over the concatenation of the embeddings for r_j's record type, word value, position, and a feature for whether it is the final position for the type. If there are multiple r_j from which y_t could have been copied, the computed representations are averaged. At test time, we use the MAP r_j to compute the input, even if there are multiple matches. For y_t which could not have been copied, the input to the LSTM at time $t+1$ is computed using the same MLP over y_t and three dummy features.

For the autoregressive HSMM, an additional 1-layer LSTM with d hidden units is used. We experimented with having the autoregressive HSMM consume either tokens $y_{1:t}$ in predicting y_{t+1}, or the average embedding of the field *types* corresponding to copied tokens in $y_{1:t}$. The former worked slightly better for the WikiBio dataset (where field types are more ambiguous), while the latter worked slightly better for the E2E dataset.

Transition Distribution The function $\boldsymbol{C}(\boldsymbol{x}_u)$, which produces hidden state embeddings conditional on the source, is defined as $\boldsymbol{C}(\boldsymbol{x}_u) = \boldsymbol{U}_2(\mathrm{ReLU}(\boldsymbol{U}_1 \boldsymbol{x}_u))$, where $\boldsymbol{U}_1 \in \mathbb{R}^{m_3 \times d}$ and $\boldsymbol{U}_2 \in \mathbb{R}^{K \times m_2 \times m_3}$; $\boldsymbol{D}(x)$ is defined analogously. For all experiments, $m_1 = 64$, $m_2 = 32$, and $m_3 = 64$.

Optimization We train with SGD, using a learning rate of 0.5 and decaying by 0.5 each epoch after the first epoch in which validation log-likelihood fails to increase. When using an autoregressive HSMM, the additional LSTM is optimized only after the learning rate has been decayed. We regularize with Dropout (Srivastava et al., 2014).

A.2 Additional Learned Templates

In Tables 6 and 7 we show visualizations of additional templates learned on the E2E and WikiBio data, respectively, by both the non-autoregressive and autoregressive HSMM models presented in the paper. For each model, we select a set of five dissimilar templates in an iterative way by greedily selecting the next template (out of the 200 most frequent) that has the highest percentage of states that do not appear in the previously selected templates; ties are broken randomly. Individual states within a template are visualized using the three most common segments they generate.

1.	The Waterman The Golden Palace Browns Cambridge …	is a is an is a family friendly …	Italian French fast food …	restaurant pub place …	with a with with an …	average high low …	customer rating price range rating …	.
2.	There is a There is a cheap There is an …	restaurant coffee shop French restaurant …	The Mill Bibimbap House The Twenty Two …	located in the located on the located north of the …	centre of the city river city centre …	that serves serving that provides …	fast food sushi take-away deliveries …	.
3.	The Olive Grove The Punter The Cambridge Blue …	restaurant pub …	serves offers has …	fast food sushi take-away deliveries …	.			
4.	The Child friendly The average priced …	restaurant coffee shop French restaurant …	The Mill Bibimbap House The Twenty Two …	serves offers has …	English Indian Italian …	food cuisine dishes …	.	
5.	The Strada The Dumpling Tree Alimentum …	provides serves offers …	Indian Chinese English …	food in the food at a food and has a …	customer rating of price range of rating of …	1 out of 5 average 5 out of 5 …	.	

1.	The Eagle The Golden Curry Zizzi …	provides providing serves …	Indian Chinese English …	food cuisine Food …	in the with a and has a …	high moderate average …	price range customer rating rating …	.	It is They are It's …	near located in the located near …	riverside city centre Cafe Sicilia …	.	Its customer rating is It has a The price range is …	1 out of 5 average high …	.
2.	Located near Located in the Near …	The Portland Arms riverside city centre …	is an is a family friendly there is a …	Italian fast food French …	restaurant called place called restaurant named …	The Waterman Cocum Loch Fyne …	.								
3.	A An A family friendly …	Italian fast food French …	restaurant pub coffee shop …	is called named …	The Waterman Cocum Loch Fyne …	.									
4.	Located near Located in the Near …	The Portland Arms riverside city centre …	,	The Eagle The Golden Curry Zizzi …	is a is a family friendly is an …	cheap family-friendly family friendly …	Italian fast food French …	restaurant pub coffee shop …	.						
5.	A An A family friendly …	Italian fast food French …	restaurant pub coffee shop …	near located in the located near …	riverside city centre Cafe Sicilia …	is called named …	The Waterman Cocum Loch Fyne …	.							

Table 6: Five templates extracted from the E2E data with the NTemp model (top) and the Ntemp+AR model (bottom).

1.	william henry george augustus frederick marie anne de bourbon ...	(was (; ...	born born on born 1 ...	1968 1960 1970 ...	)])] ...	is an american is a russian was an american ...	politician actor football player ...	.		
2.	sir captain lieutenant ...	john herbert hartley donald charles cameron ...	was a was a british was an english ...	world war i world war first world war ...	national team organization super league ...	.				
3.	john herbert hartley donald charles cameron ...	is a was a is an ...	indie rock death metal ska ...	band midfielder defenceman ...	from for based in ...	australia los angeles, california chicago ...	.			
4.	john herbert hartley donald charles cameron ...	was a is a is a former ...	american major league baseball australian ...	football professional baseball professional ice hockey ...	midfielder defender goalkeeper ...	.				
5.	james william john william ...	“ billy ” wilson smith “ jack ” henry ...	(	1900 c. 1894 1913 ...	–	france budapest buenos aires ...	)	is an american is an english was an american ...	footballer professional footballer rules footballer ...	
	who plays for who currently plays for who played with ...	paganese south melbourne fc dynamo kyiv ...	in the of the and the ...	vicotiral football league national football league australian football league ...	(	vfl nfl afl ...	)	.		
1.	aftab ahmed anderson da silva david jones ...	(; ...	born born on born 1 ...	1951 1970 1974 ...	)] ...	is an american was an american is an english ...	actor actress cricketer ...	.		
2.	aftab ahmed anderson da silva david jones ...	was a is a former is a ...	world war i liberal baseball ...	member of the party member of the recipient of the ...	austrian pennsylvania montana ...	house of representatives legislature senate ...	.			
3.	adjutant lieutenant captain ...	aftab ahmed anderson da silva david jones ...	was a is a former is a ...	world war i liberal baseball ...	member of the party member of the recipient of the ...	knesset scottish parliament fc lokomotiv liski ...	.			
4.	william john william james “ ...	“ billy ” watson smith jim ” edward ...	(	1913 c. 1900 1913 ...	– in - ...	1917 surrey, england british columbia ...	)	was an american was an australian is an american ...	football player rules footballer defenceman ...	
	who plays for who currently plays for who played with ...	collingwood st kilda carlton ...	in the of the and the ...	victorial football league national football league australian football league ...	(	vfl afl nfl ...	)	.		
5.	aftab ahmed anderson da silva david jones ...	is a is a former is a female ...	member of the party member of the recipient of the ...	knesset scottish parliament fc lokomotiv liski ...	.					

Table 7: Five templates extracted from the WikiBio data with the NTemp model (top) and the Ntemp+AR model (bottom).

Multi-Reference Training with Pseudo-References for Neural Translation and Text Generation

Renjie Zheng [1] **Mingbo Ma** [1,2] **Liang Huang** [1,2]
[1]School of EECS, Oregon State University, Corvallis, OR
[2]Baidu Research, Sunnyvale, CA
zheng@renj.me {cosmmb, liang.huang.sh}@gmail.com

Abstract

Neural text generation, including neural machine translation, image captioning, and summarization, has been quite successful recently. However, during training time, typically only one reference is considered for each example, even though there are often multiple references available, e.g., 4 references in NIST MT evaluations, and 5 references in image captioning data. We first investigate several different ways of utilizing multiple human references during training. But more importantly, we then propose an algorithm to generate exponentially many pseudo-references by first compressing existing human references into lattices and then traversing them to generate new pseudo-references. These approaches lead to substantial improvements over strong baselines in both machine translation (+1.5 BLEU) and image captioning (+3.1 BLEU / +11.7 CIDEr).

1 Introduction

Neural text generation has attracted much attention in recent years thanks to its impressive generation accuracy and wide applicability. In addition to demonstrating compelling results for machine translation (MT) (Sutskever et al., 2014; Bahdanau et al., 2014), by simple adaptation, practically very same or similar models have also proven to be successful for summarization (Rush et al., 2015; Nallapati et al., 2016) and image or video captioning (Venugopalan et al., 2015; Xu et al., 2015a).

The most common neural text generation model is based on the encoder-decoder framework (Sutskever et al., 2014) which generates a variable-length output sequence using an RNN-based decoder with attention mechanisms (Bahdanau et al., 2014; Xu et al., 2015b). There are many recent efforts in improving the generation accuracy, e.g., ConvS2S (Gehring et al., 2017) and Transformer (Vaswani et al., 2017). However, all these efforts are limited to training with a single reference even when multiple references are available.

Multiple references are essential for evaluation due to the non-uniqueness of translation and generation unlike classification tasks. In MT, even though the training sets are usually with single reference (bitext), the evaluation sets often come with multiple references. For example, the NIST Chinese-to-English and Arabic-to-English MT evaluation datasets (2003–2008) have in total around 10,000 Chinese sentences and 10,000 Arabic sentences each with 4 different English translations. On the other hand, for image captioning datasets, multiple references are more common not only for evaluation, but also for training, e.g., the MSCOCO (Lin et al., 2014) dataset provides 5 references per image and PASCAL-50S and ABSTRACT-50S (Vedantam et al., 2015) even provide 50 references per image. Can we use the extra references during training? How much can we benefit from training with multiple references?

We therefore first investigate several different ways of utilizing existing human-annotated references, which include Sample One (Karpathy and Fei-Fei, 2015), Uniform, and Shuffle methods (explained in Sec. 2). Although Sample One has been explored in image captioning, to the best of our knowledge, this is the first time that an MT system is trained with multiple references.

Actually, four or five references still cover only a tiny fraction of the exponentially large space of potential references (Dreyer and Marcu, 2012). More importantly, encouraged by the success of training with multiple human references, we further propose a framework to generate many more pseudo-references automatically. In particular, we design a neural multiple-sequence alignment algo-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3188–3197
Brussels, Belgium, October 31 - November 4, 2018.

rithm to compress all existing human references into a lattice by merging similar words across different references (see examples in Fig. 1); this can be viewed as a modern, neural version of paraphrasing with multiple-sequence alignment (Barzilay and Lee, 2003, 2002). We can then generate theoretically exponentially more references from the lattice.

We make the following main contributions:

- Firstly, we investigate three different methods for multi-reference training on both MT and image captioning tasks (Section 2).
- Secondly, we propose a novel neural network-based multiple sequence alignment model to compress the existing references into lattices. By traversing these lattices, we generate exponentially many new pseudo-references (Section 3).
- We report substantial improvements over strong baselines in both MT (+1.5 BLEU) and image captioning (+3.1 BLEU / +11.7 CIDEr) by training on the newly generated pseudo-references (Section 4).

2 Using Multiple References

In order to make the multiple reference training easy to adapt to any frameworks, we do not change anything from the existing models itself. Our multiple reference training is achieved by converting a multiple reference dataset to a single reference dataset without losing any information.

Considering a multiple reference dataset D, where the i^{th} training example, $(\mathbf{x}_i, Y_i)$, includes one source input $\mathbf{x}_i$, which is a source sentence in MT or image vector in image captioning, and a reference set $Y_i = \{\mathbf{y}_i^1, \mathbf{y}_i^2, ...\mathbf{y}_i^K\}$ of K references. We have the following methods to convert the multiple reference dataset to a single reference dataset D' (note that the following $D'_{\text{sample one}}$, D'_{uniform} and D'_{shuffle} are ordered sets):

Sample One: The most straightforward way is to use a different reference in different epochs during training to explore the variances between references. For each example, we randomly pick one of the K references in each training epoch (note that the random function will be used in each epoch). This method is commonly used in existing image captioning literatures, such as (Karpathy and Fei-Fei, 2015), but never used in MT. This approach can be formalized as:

$$D'_{\text{sample one}} = \bigcup_{i=1}^{|D|} \{(\mathbf{x}_i, \mathbf{y}_i^{k_i})\}, k_i = \text{rand}(1, ..., K)$$

Uniform: Although all references are accessible by using Sample One, it is not guaranteed that all references are used during training. So we introduce **Uniform** which basically copies $\mathbf{x}_i$ training example K times and each time with a different reference. This approach can be formalized as:

$$D'_{\text{uniform}} = \bigcup_{i=1}^{|D|} \bigcup_{k=1}^{K} \{(\mathbf{x}_i, \mathbf{y}_i^k)\}$$

Shuffle is based on Uniform, but shuffles all the source and reference pairs in random order before each epoch. So, formally it is:

$$D'_{\text{shuffle}} = \text{Shuffle}(D'_{\text{uniform}})$$

Sample One is supervised by different training signals in different epochs while both Uniform and Shuffle include all the references at one time. Note that we use mini-batch during training. When we set the batch size equal to the entire training set size in both Uniform and Shuffle, they become equivalent.

3 Pseudo-References Generation

In text generation tasks, the given multiple references are only a small portion in the whole space of potential references. To cover a larger number of references during training, we want to generate more pseudo-references which is similar to existing ones.

Our basic idea is to compress different references $\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_K$ into a lattice. We achieve this by merging similar words in the references. Finally, we generate more pseudo-references by simply traversing the compressed lattice and select those with high quality according to its BLEU score.

Take the following three references from the NIST Chinese-to-English machine translation dataset as an example:

1. `Indonesia reiterated its opposition to foreign military presence`
2. `Indonesia repeats its opposition against station of foreign troops in Indonesia`
3. `Indonesia reiterates opposition to garrisoning foreign armies`

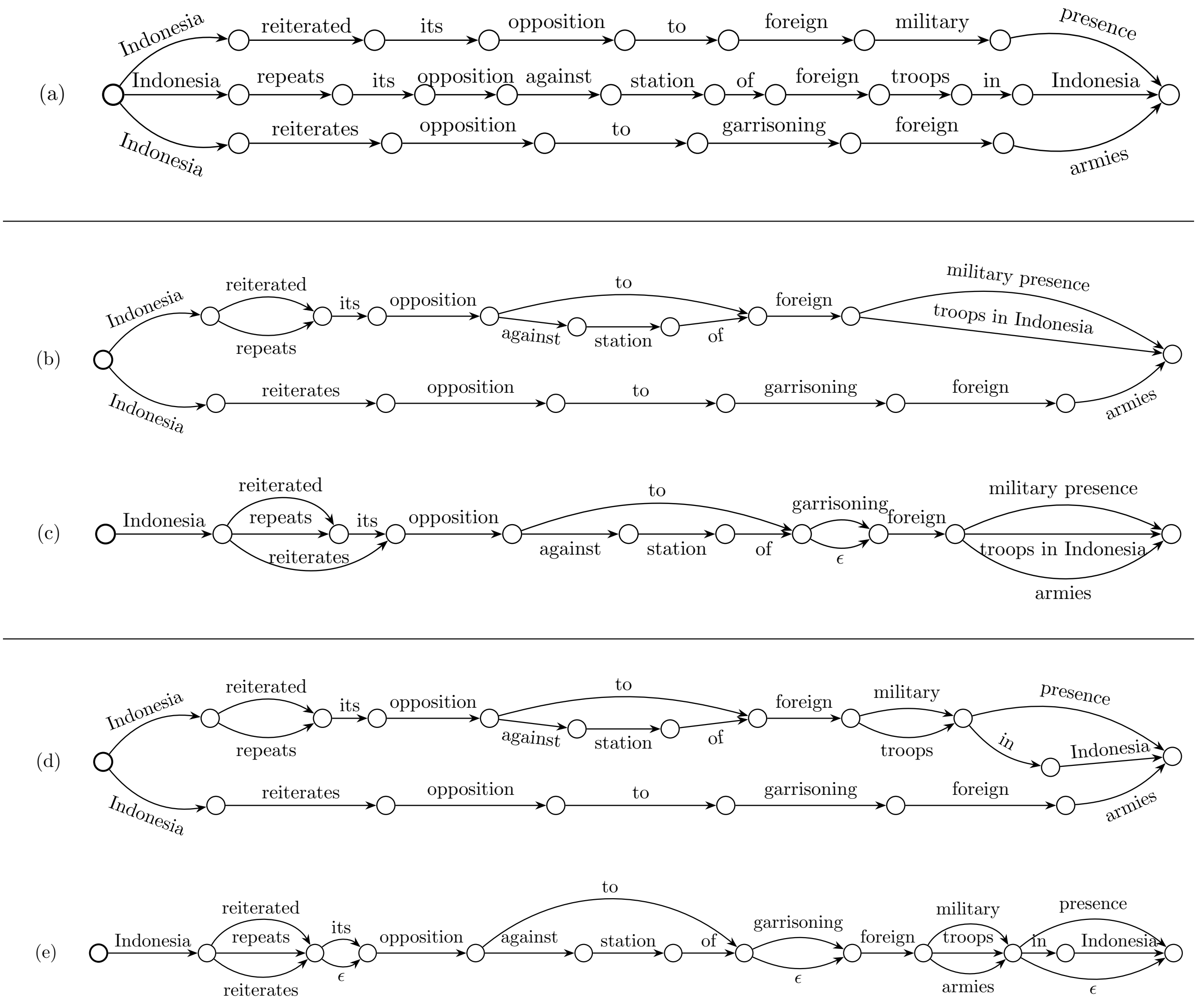

Figure 1: Lattice construction with word alignment. (b-c) is hard word alignment and 33 pseudo-references can be generated. (d-e) is soft word alignment, 213 pseudo-references can be generated.

3.1 Naive Idea: Hard Word Alignment

The simplest way to compress different references into a lattice is to do pairwise reference compression iteratively. At each time, we select two references and merge the same words in them.

Considering the previous example, we can derive an initial lattice from the three references as shown in Fig. 1(a). Assume that we first do a pairwise reference compression on first two references, we can merge at four sharing words: `Indonesia`, `its`, `opposition` and `foreign`, and the lattice will turn to Fig. 1(b). If we further compress the first and third references, we can merge at `Indonesia`, `opposition`, `to` and `foreign`, which gives the lattice Fig. 1(c). By simply traversing the final lattice, 33 new pseudo-references can be generated. For example:

1. `Indonesia reiterated its opposition to garrisoning foreign armies`
2. `Indonesia repeats its opposition to foreign military presence`
3. `Indonesia reiterates opposition to foreign troops in Indonesia`
4. ...

However, this simple hard alignment method (only identical words can be aligned) suffers from two problems:

1. Different words may have similar meanings and need to be merged together. For example, in the previous example, `reiterated`, `repeats` and `reiterates` should be merged together. Similarly, `military`, `troops` and `armies` also have similar meanings. If the

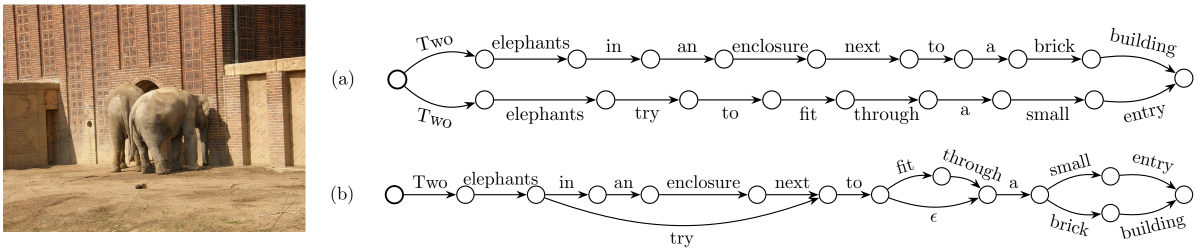

Figure 2: Mistakes from hard word alignment by merging at "`to`".

lattice can align these words, we can generate the lattice shown in Fig. 1(e) which can generate 213 pseudo-references.

2. Identical words may have different meaning in different contexts and should not be merged. Considering the following two references from the COCO image captioning dataset (corresponding picture is shown in Fig. 2):

1. `Two elephants in an enclosure next to a brick building`
2. `Two elephants try to fit through a small entry`

Following the previously described algorithm, we can merge the two references at "`two elephants`", at "`to`" and at "`a`". However, "`to`" in the two references are very different (it is a preposition in the first reference and an infinitive in the second) and should not be merged. Thus, the lattice in Fig. 2(b) will generate the following wrong pseudo-references:

1. `Two elephants try to a small entry`
2. `Two elephants in an enclosure next to fit through a brick building`

Therefore, we need to investigate a better method to compress the lattice.

3.2 Measuring Word Similarity in Context

To tackle the above listed two problems of hard alignment, we need to identify synonyms and words with similar meanings. Barzilay and Lee (2002) utilize an external synonyms dictionary to get the similarity score between words. However, this method ignores the given context of each word. For example, in Fig. 1(a), there are two `Indonesia`'s in the second path of reference. If we use a synonyms dictionary, both `Indonesia` tokens will be aligned to the `Indonesia` in the first or third sentence with the same score. This incorrect alignment would lead to meaningless lattice.

Thus, we introduce the semantic substitution matrix which measures the semantic similarity of each word pairs in context. Formally, given a sentence pair $\mathbf{y}_i$ and $\mathbf{y}_j$, we build a semantic substitution matrix $M = \mathbb{R}^{|\mathbf{y}_i| \times |\mathbf{y}_j|}$, whose cell $M_{u,v}$ represents the similarity score between word $\mathbf{y}_{i,u}$ and word $\mathbf{y}_{j,v}$.

We propose a new neural network-based multiple sequence alignment algorithm to take context into consideration. We first build a language model (LM) to obtain the semantic representation of each word, then these word representations are used to construct the semantic substitution matrix between sentences.

Fig. 3 shows the architecture of the bidirectional LM (Mousa and Schuller, 2017). The optimization goal of our LM is to minimize the i^{th} word's prediction error given the surrounding word's hidden state:

$$p(w_i \mid \overrightarrow{\mathbf{h}_{i-1}} \oplus \overleftarrow{\mathbf{h}_{i+1}}) \qquad (1)$$

For any new given sentences, we concatenate both forward and backward hidden states to represent each word $\mathbf{y}_{i,u}$ in a sentence $\mathbf{y}_i$. We then calculate the normalized *cosine* similarity score of word $\mathbf{y}_{i,u}$ and $\mathbf{y}_{j,v}$ as:

$$M_{u,v} = cosine(\overrightarrow{\mathbf{h}_u} \oplus \overleftarrow{\mathbf{h}_u}, \overrightarrow{\mathbf{h}_v} \oplus \overleftarrow{\mathbf{h}_v}) \qquad (2)$$

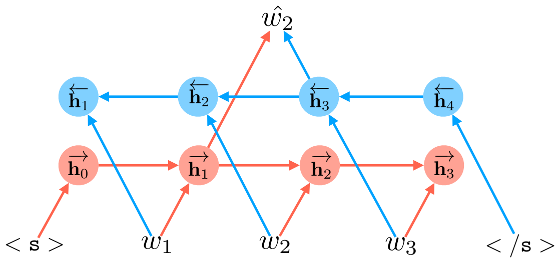

Figure 3: Bidirectional Language Model

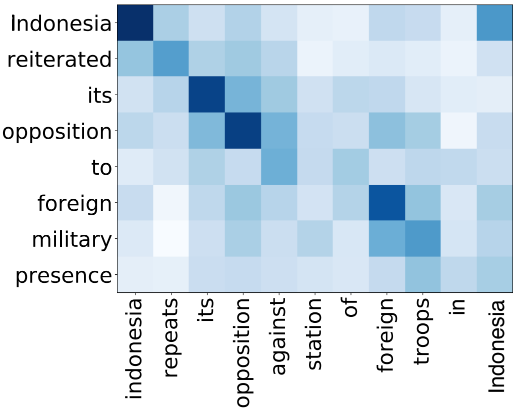

Figure 4: Semantic Substitution Matrix

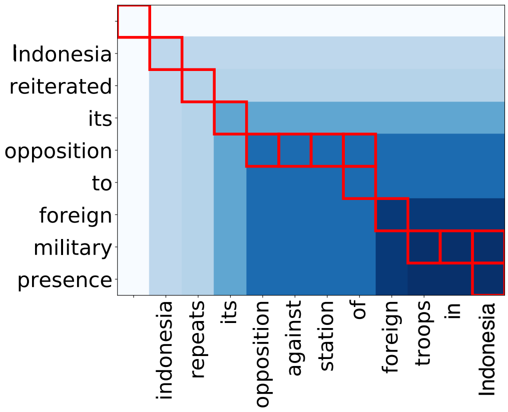

Figure 5: Dynamic Programming on Semantic Substitution Matrix

Fig. 4 shows an example of the semantic substitution matrix of first two sentences in example references of Fig. 1(a).

3.3 Iterative Pairwise Word Alignment using Dynamic Programming

With the help of semantic substitution matrix $M_{u,v}$ which measures pairwise word similarity, we need to find the optimal word alignment to compress references into a lattice.

Unfortunately, this computation is exponential in the number of sequences. Thus, we use iterative pairwise alignment which greedily merges sentence pairs (Durbin et al., 1998).

Based on pairwise substitution matrix we can define an optimal pairwise sequence alignment as an optimal path from $M_{0,0}$ to $M_{|\mathbf{y}_i|,|\mathbf{y}_j|}$. This is a dynamic programming problem with the state transition function described in Equation (3). Fig. 5 shows the optimal path according to the semantic substitution matrix in Fig. 4. There is a gap if the continuous step goes vertical or horizontal, and an alignment if it goes diagonal.

$$opt(u,v) = \begin{cases} opt(u-1,v-1)+M_{u,v} \\ opt(u-1,v) \\ opt(u,v-1) \end{cases} \quad (3)$$

What order should we follow to do the iterative pairwise word alignment? Intuitively, we need to compress the most similar reference pair first, since this compression will lead to more aligned words. Following this intuition, we order reference pairs by the maximum alignment score $opt(|\mathbf{y}_i|,|\mathbf{y}_j|)$ (i.e. the score of bottom-right cell in Fig. 5) which is the sum of all aligned words. Using this order, we can iteratively merge each sentence pair in descending order, unless both the sentences have already been merged (this will prevent generating a cyclic lattice).

Since the semantic substitution matrix $M_{u,v}$, defined as a normalized cosine similarity, scales in $(0, 1)$, it's very likely for the DP algorithm to align unrelated words. To tackle this problem, we deduct a global penalty p from each cell of $M_{u,v}$. With the global penalty p, the DP algorithm will not align a word pair $(\mathbf{y}_{i,u}, \mathbf{y}_{i,v})$ unless $M_{u,v} \geq p$.

After the pairwise references alignment, we merge those aligned words. For example, in Fig. 1, after we generate an initial lattice as shown in Fig. 1(a), we then calculate the maximum alignment score of all sentence pairs. After that, the lattice turns into Fig. 1(d) by merging the first two references (assuming they have the highest score) according to pairwise alignment shown in Fig. 5. Then we pick the sentence pair with next highest alignment score (assuming it's the last two sentences). Similar to the previous step, we find alignments according to the dynamic programming and merge to the final lattice (see Fig. 1(e)).

3.4 Traverse Lattice and Pseudo-References Selection by BLEU

We generate pseudo-references by simply traversing the generated lattice. For example, if we traverse the final lattice shown in Fig. 1(e), we can generate 213 pseudo-refrences in total.

Then, we can put those generated pseudo-references to expand the training dataset. To balance the number of generated pseudo-references for each example, we force the total number of pseudo-references from each example to be

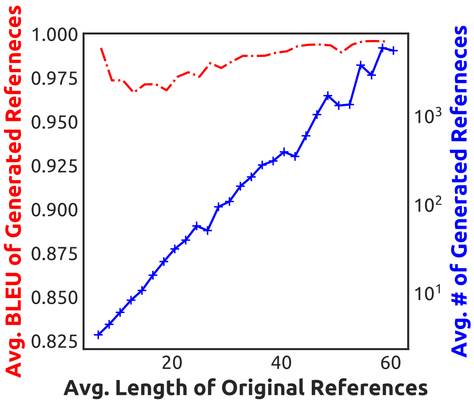

(a) Machine Translation Dataset

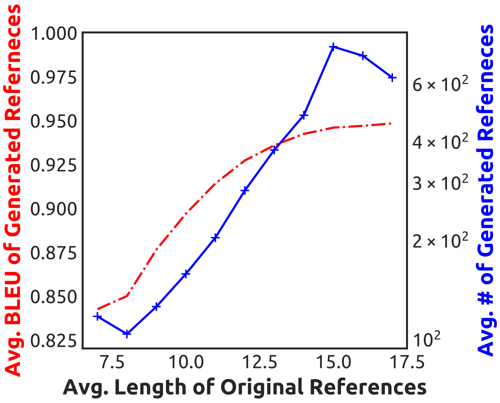

(b) Image Captioning Dataset

Figure 6: Analysis of generated references

K'. For those examples generating k pseudo-references and $k > K'$, we calculate all pseudo-references' BLEU scores based on gold references, and only keep top $K' - k$ pseudo-references with highest BLEU score.

4 Experiments

To investigate the empirical performances of our proposed algorithm, we conduct experiments on machine translation and image captioning.

4.1 Machine Translation

We evaluate our approach on NIST Chinese-to-English translation dataset which consists of 1M pairs of single reference data and 5974 pairs of 4 reference data (NIST 2002, 2003, 2004, 2005, 2006, 2008). Table 1 shows the statistics of this dataset. We first pre-train our model on a 1M pairs single reference dataset and then train on the NIST 2002, 2003, 2004, 2005. We use the NIST 2006 dataset as validation set and NIST 2008 as test sets.

Fig. 6(a) analyzes the number and quality of generated references using our proposed approach. We set the global penalty as 0.9 and only calculate the top 50 generated references for the average BLEU analysis. From the figure, we can see that when the sentence length grows, the number of generated references grows exponentially. To generate enough references for the following experiments, we set an initial global penalty as 0.9 and gradually decrease it by 0.05 until we collect no less than 100 references. We train a bidirectional language model on the pre-training dataset and training dataset with Glove (Pennington et al., 2014) word embedding size of 300 dimension, for 20 epochs to minimize the perplexity

We employ byte-pair encoding (BPE) (Sennrich et al., 2015) which reduces the source and target language vocabulary sizes to 18k and 10k. We adopt length reward (Huang et al., 2017) to find optimal sentence length. We use a two layer bidirectional LSTM as the encoder and a two layer LSTM as the decoder. We perform pre-training for 20 epochs to minimize perplexity on the 1M dataset, with a batch size of 64, word embedding size of 500, beam size of 15, learning rate of 0.1, learning rate decay of 0.5 and dropout rate of 0.3. We then train the model in 30 epochs and use the best batch size among 100, 200, 400 for each update method. These batch sizes are multiple of the number of references used in experiments, so it is guaranteed that all the references of one single example are in one batch for the Uniform method. The learning rate is set as 0.01 and learning rate decay as 0.75. We do each experiment three times and report the average result.

Table 2 shows the translation quality on the dev-set of machine translation task. Besides the original 4 references in the training set, we generate another four dataset with 10, 20, 50 and 100 references including pseudo-references using hard word alignment and soft word alignment. We compare the three update methods (Sample One, Uniform, Shuffle) with always using the first reference (First). All results of soft word alignment are better than corresponding hard word alignment results and the best result is achieved with 50 references using Uniform and soft word alignment. According to Table 3, Shuffle with original 4 references has +0.7 BLEU improvement and Uniform

Task		Pre-training	Training	Validation	Testing
Machine Translation	# of examples	1,000,000	4,667	616	691
	# of refs per example	1	4	4	4
Image Captioning	# of examples	-	113,287	5,000	5,000
	# of refs per example	-	5	5	5

Table 1: Statistics of datasets used in following experiments.

# of Refs	Method	BLEU	
0	Pre-train	37.44	
1	First*	38.64	
4	Sample One	38.81	
	Uniform	38.78	
	Shuffle	38.87	
Includes Pseudo-Refs		Hard Align	Soft Align
10	Sample One	37.48	39.41
	Uniform	39.20	39.35
	Shuffle	39.13	39.53
20	Sample One	37.27	38.70
	Uniform	39.14	39.46
	Shuffle	39.12	39.42
50	Sample One	37.42	37.62
	Uniform	39.30	**39.65**
	Shuffle	38.98	39.08
100	Sample One	37.54	37.63
	Uniform	39.23	39.46
	Shuffle	38.88	39.03

Table 2: BLEU on the MT validation set. * Baseline

# of Refs	Method	BLEU
0	Pre-train	33.58
1	First*	34.49
4	Shuffle	35.20 (+0.7)
†50	Uniform	**35.98** (+1.5)

Table 3: BLEU on the MT test set. †Includes pseudo-references generated by soft word alignment algorithm. *Baseline.

# of Refs	Method	BLEU		CIDEr	
1	First	26.27		79.05	
5	Sample One*	29.03		85.39	
	Uniform	30.05		89.76	
	Shuffle	30.41		91.21	
Includes Pseudo-Refs		Hard Align		Soft Align	
		BLEU	CIDEr	BLEU	CIDEr
10	Sample One	30.63	91.76	30.98	92.02
	Uniform	30.40	91.48	30.77	91.89
	Shuffle	30.68	92.01	30.91	92.22
20	Sample One	30.69	92.25	30.91	92.32
	Uniform	30.73	91.69	31.03	92.61
	Shuffle	31.56	94.99	**31.92**	**95.59**
50	Sample One	30.76	91.81	31.07	92.17
	Uniform	30.66	92.30	30.99	92.61
	Shuffle	30.83	93.26	31.06	94.19

Table 4: BLEU/CIDEr on the image captioning validation set. *Baseline.

# of Refs	Method	BLEU	CIDEr
1	First	26.70	80.70
5	Sample One*	28.67	85.41
5	Shuffle	30.94 (+2.3)	94.10 (+8.7)
†20	Shuffle	**31.79** (+3.1)	**97.10** (+11.7)

Table 5: BLEU/CIDEr on the image captioning test set with soft. † Includes pseudo-references generated by soft word alignment algorithm. * Baseline.

with 50 references has +1.5 BLEU improvement. From Fig. 7(b), we can see that using the Sample One method, the translation quality drops dramatically with more than 10 references. This may be due to the higher variance of used reference in each epoch.

4.2 Image Captioning

For the image captioning task, we use the widely-used MSCOCO image captioning dataset. Following prior work, we use the Kapathy split (Karpathy and Fei-Fei, 2015). Table 1 shows the statistics of this dataset. We use Resnet (He et al., 2016) to extract image feature of 2048 feature size and simple fully connected layer of size 512 to an LSTM decoder. We train every model for 100 epochs and calculate the BLEU score on validation set and select the best model. For every update method, we find the optimal batch size among 50, 250, 500, 1000 and we use a beam size of 5.

Fig. 6(b) analyzes the correlation between average references length with the number and quality of generated references. We set global penalty as 0.6 (which is also adopted for the generated references in the following experiments) and calculate the top 50 generated references for the average BLEU analysis. Since the length of original references is much shorter than the previous machine translation dataset, it has worse quality and fewer generated references.

Table 4 shows that the best result is achieved with 20 references using Shuffle. This result is

Image	Original References
	a gray tabby cat is curled in a red bowl that sits on a table near a window a brown and black cat is sleeping in a bowl on a table a grey tiger cat sleeping in a brown bowl on a table an image of a cat sitting inside of a bowl on the kitchen table a cat asleep in a fruit bowl on a dining room table

Generated Lattice using Soft Alignment

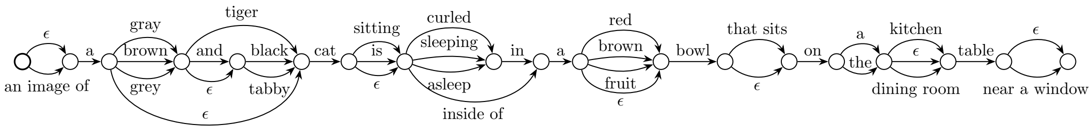

ID	Pseudo-references	BLEU
1	a grey tiger cat sleeping in a brown bowl on a table near a window	100.0
2	a grey tiger cat sleeping in a brown bowl on a dining room table	100.0
3	a brown and black cat is sleeping in a bowl on the kitchen table	100.0
...	...	...
48	a grey tiger cat sleeping in a fruit bowl on a table	97.1
49	a cat asleep in a red bowl that sits on a table	97.1
50	a gray tabby cat is sleeping in a bowl on a table	97.1
...	...	...
73723	a grey and tabby cat inside of a red bowl on the dining room table	0.0
73724	a grey and tabby cat inside of a red bowl on a kitchen table	0.0

Table 6: Training example that generates maximum number of pseudo-references (73724). The selected 8 pseudo-references are sorted according to their BLEU score.

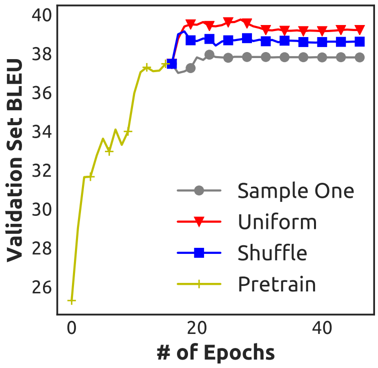

(a) Learning curve of different methods with 50 References

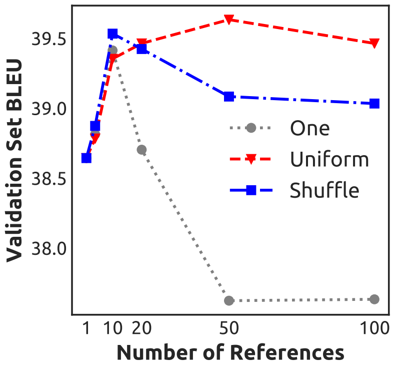

(b) MT with different number of references

Figure 7: Translation quality of machine translation task on dev-set with soft alignment

different from the result of machine translation task where Uniform method is the best. This may be because the references in image captioning dataset are much more diverse than those in machine translation dataset. Different captions of one image could even talk about different aspects. When using the Uniform method, the high variance of references in one batch may harm the model and lead to worse text generation quality. Table 5 shows that it outperforms Sample One with 4 original references, which is adopted in previous work (Karpathy and Fei-Fei, 2015), +3.1 BLEU score and +11.7 CIDEr.

4.3 Case Study

Fig. 6 shows a training example in the COCO dataset and its corresponding generated lattice and pseudo-references which is sorted according to its BLEU score. Our proposed algorithm generates 73724 pseudo-references in total. All the top 50 pseudo-references' BLEU scores are above 97.1 and the top three even achieve 100.0 BLEU score though they are not identical to any original references. Although the BLEU of last two sentences is 0.0, they are still valid to describe this picture.

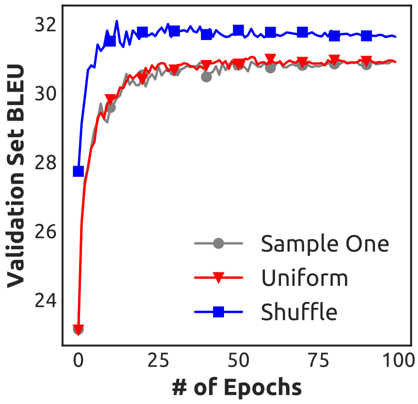

(a) Learning curve of different methods with 20 References

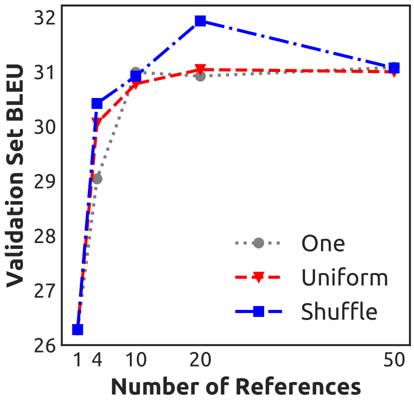

(b) Image captioning with different number of references

Figure 8: Text generation quality of image captioning task on validation set with soft alignment

5 Conclusions

We introduce several multiple-reference training methods and a neural-based lattice compression framework, which can generate more training references based on existing ones. Our proposed framework outperforms the baseline models on both MT and image captioning tasks.

Acknowledgments

This work was supported in part by DARPA grant N66001-17-2-4030, and NSF grants IIS-1817231 and IIS-1656051. We thank the anonymous reviewers for suggestions and Juneki Hong for proofreading.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*.

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*.

Markus Dreyer and Daniel Marcu. 2012. Hyter: Meaning-equivalent semantics for translation evaluation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 162–171. Association for Computational Linguistics.

Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. 1998. Biological sequence analysis: probabilistic models of proteins and nucleic acids.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition CVPR*.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *EMNLP 2017*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context.

Amr Mousa and Björn Schuller. 2017. Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*.

Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. 2016. Classify or select: Neural architectures for extractive document summarization. *CoRR*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Proceedings of the 27th International Conference on Neural Information Processing Systems*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence - video to text. In *ICCV*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015a. Show, attend and tell: Neural image caption generation with visual attention. *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015b. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of Machine Learning Research*.

Knowledge Graph Embedding with Hierarchical Relation Structure

Zhao Zhang[1,2], **Fuzhen Zhuang**[1,2*], **Meng Qu**[3], **Fen Lin**[4], **Qing He**[1,2]
[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing 100190, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]Rutgers Business School, Rutgers University, New Jersey, 07102, USA
[4]Search Product Center, WeChat Search Application Department, Tencent, China
`{zhangzhao2017, zhuangfuzhen, heqing}@ict.ac.cn`
∗corresponding author

Abstract

The rapid development of knowledge graphs (KGs), such as Freebase and WordNet, has changed the paradigm for AI-related applications. However, even though these KGs are impressively large, most of them are suffering from incompleteness, which leads to performance degradation of AI applications. Most existing researches are focusing on knowledge graph embedding (KGE) models. Nevertheless, those models simply embed entities and relations into latent vectors without leveraging the rich information from the relation structure. Indeed, relations in KGs conform to a three-layer hierarchical relation structure (HRS), i.e., semantically similar relations can make up relation clusters and some relations can be further split into several fine-grained sub-relations. Relation clusters, relations and sub-relations can fit in the top, the middle and the bottom layer of three-layer HRS respectively. To this end, in this paper, we extend existing KGE models TransE, TransH and DistMult, to learn knowledge representations by leveraging the information from the HRS. Particularly, our approach is capable to extend other KGE models. Finally, the experiment results clearly validate the effectiveness of the proposed approach against baselines.

1 Introduction

Knowledge Graphs (KGs) are extremely useful resources for many AI-related applications, such as question answering, information retrieval and query expansion. Indeed, KGs are multi-relational directed graphs composed of entities as nodes and relations as edges. They represent information about real-world entities and relations in the form of knowledge triples, which is denoted as (h, r, t), where h and t correspond to the head and tail entities and r denotes the relation between them, e.g., $(Donald\ Trump, presidentOf, USA)$. Large scale, collaboratively created KGs , such as Freebase (Bollacker et al., 2008), WordNet (Miller, 1994), Yago (Suchanek et al., 2007), Gene Ontology (Sherlock, 2009), NELL (Carlson et al., 2010) and Google's KG[1], have recently become available. However, despite the impressively large sizes, the coverage of most existing KGs are far from complete. This has motivated research in knowledge base completion task, which includes KGE methods aiming to embed entities and relations in KGs into low-dimensional embeddings.

In the literature, there are a number of studies about KGE models. These models embed entities and relations into latent vectors and complete KGs based on these vectors, such as TransE (Bordes et al., 2013), TransH (Wang et al., 2014) and TransR (Lin et al., 2015b). However, most of the existing works simply embed relations into vectors. Less efforts have been made for investigating the rich information from the relation structure. Indeed, in this research, we define a three-layer hierarchical relation structure (HRS), which can be conformed by relation clusters, relations and sub-relations in KGs.

- **Relation clusters**: Semantically similar relations are often observed in Large-scales KGs. For example, the relation *'producerOf'* and *'directorOf'* may be semantically related if both of them describe a relation between a person and a film. These semantically similar relations can make up relation clusters. We believe the information from semantically similar relations is of great value, and relations in the same group can be trained in a collective way to facilitate the knowledge sharing when learning the embeddings of related relations.

- **Relations**: A relation connects the head and

---

[1] https://www.google.com/intl/es419/insidesearch/features/search/knowledge.html

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3198–3207
Brussels, Belgium, October 31 - November 4, 2018.

tail entities in a knowledge triple, denoted as (h, r, t), where h and t correspond to the head and tail entities and r denotes the relation between them.

- **Sub-relations**: There are relations that have multiple semantic meanings and can be split into several sub-relations. For example, the relation $partOf$ has at least two semantics: location-related as $(New\ York$, $partOf$, $USA)$ and composition-related as $(monitor$, $partOf$, $television)$. We believe the sub-relations can give fine-grained descriptions for each relation.

The relation clusters, relations and sub-relations correspond to the top, middle and bottom layer of the three-layer HRS.

In this paper, we extend state-of-the-art models TransE (Bordes et al., 2013), TransH (Wang et al., 2014) and DistMult (Yang et al., 2015) to learn knowledge representations by leveraging the rich information from the HRS. Moreover, the same technique can easily be used to extend other state-of-the-art models and utilize the HRS information. In the proposed models, for each knowledge triple (h, r, t), the embedding of r is the sum of three embedding vectors, which correspond to the three layers of the HRS respectively and therefore, the information from the HRS is leveraged. Particularly, instead of using additional information like text or paths, our model simply use the knowledge triples in KGs and the rich information from the HRS. Extensive experiments on popular benchmark data sets demonstrate the effectiveness of our models.

In summary, we highlight our key contributions as follows,

1. We propose a technique by making use of the HRS information to conduct the KGE task, and extend three state-of-the-art models to utilize this technique. The technique can be easily applied to other KGE models.

2. Our proposed models don't use additional information like text or paths, instead, we only use the knowledge triples in KGs and take advantage of the rich information from the HRS.

3. We evaluate our models on popular benchmark data sets, and the results show that our extended models achieve substantial improvements against the original models as well as other state-of-the-art baselines.

2 Preliminaries and Related Work

We extend three popular KGE models by leveraging the HRS information in this study. Therefore, in this section, we first introduce the three existing models TransE (Bordes et al., 2013), TransH (Wang et al., 2014) and DistMult (Yang et al., 2015) in detail. Then, we further summarize other state-of-the-art models on the topic of KGE.

2.1 TransE, TransH and DistMult

Recently, a number of KGE models have been proposed. These methods learn low-dimensional vector representations for entities and relations (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015b).

TransE (Bordes et al., 2013) is one of the most widely used model, which views relations as translations from a head entity to a tail entity on the same low-dimensional hyperplane, i.e, $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when (h, r, t) holds. This indicates that $\mathbf{t}$ should be the nearest neighbor of $\mathbf{h} + \mathbf{r}$. In this case, the score function of TransE is defined as

$$f_r(h, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{L_n}, \quad (1)$$

which can be measured by L_1 or L_2 norm. Positive triples are supposed to have lower scores than negative ones.

TransH (Wang et al., 2014) introduces a mechanism of projecting entities into relation-specific hyperplanes that enables different roles of an entity in different relations. TransH models the relation as a vector $\mathbf{r}$ on a hyperplane $\mathbf{w}_r$ and assumes that $\mathbf{h}_\perp + \mathbf{r} \approx \mathbf{t}_\perp$ when (h, r, t) holds, where $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$ are the projection of $\mathbf{h}$ and $\mathbf{t}$ in the relation-specific hyperplane. The score function of TransH is defined as

$$f_r(h, t) = \|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|_2^2, \quad (2)$$

where $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r$, $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r$ and $\|\mathbf{w}_r\|_2 = 1$. Like triples in TransE, positive triples in TransH should have lower scores than negative ones.

DistMult (Yang et al., 2015) adopts a bilinear score function to compute the scores given (h, r, t) triples. The score function is defined as

$$f_r(h, t) = \mathbf{h}\mathbf{M}_r\mathbf{t}, \quad (3)$$

where $\mathbf{M}_r$ is a relation-specific diagonal matrix, which represents the characteristics of a relation. Different from TransE and TransH, positive triples should have larger scores than negative ones.

2.2 Other KGE Models

Besides TransE, TransH and DistMult, there are also many models on the topic of KGE. TransR (Lin et al., 2015b) embeds entities and relations into separate entity space and relation-specific spaces. ComplEx (Welbl et al., 2016) extends DistMult to embed entities and relations into complex vectors instead of real-valued ones. HolE (Nickel et al., 2016) employs circular correlations to create compositional representations. ProjE (Shi and Weninger, 2017) adopts a two-layer network to embed entities and relations. Other KGE models also try to embeds entities and relations in various ways, such as Unstructured Model (Bordes et al., 2012a, 2014), Structured Embedding (Bordes et al., 2012b), Single Layer Model (Socher et al., 2013), Semantic Matching Energy (Bordes et al., 2012a, 2014), NTN Model (Socher et al., 2013), etc.

Many efforts have been devoted to building models using additional information like paths or text. For instance, PTransE (Lin et al., 2015a) and R-GCN (Schlichtkrull et al., 2017) use paths as additional information, while DKRL (Xie et al., 2016) and SSP (Xiao et al., 2017) adopt text to assist the embedding task.

Some KGE works focus on making use of the information from relations. CTransR (Lin et al., 2015b), TransD (Ji et al., 2015) and TransG (Xiao et al., 2016) try to find fine-grained representations for each relation. However, these works didn't utilize the information from semantically similar relations and the HRS is also not exploited. Different from the above studies, we believe semantically similar relations can make up relation clusters, and some relations may have multiple semantic meanings and can be split into fine-grained sub-relations. In this paper, we take advantage of the three-layer HRS and conduct the KGE task by extending three widely used models.

3 Methodology

In this section, we provide the technical details of how to extend existing KGE models by leveraging the HRS information. We first formally define the HRS and its integration with existing models.Then we introduce the new loss functions of extended models TransE-HRS, TransH-HRS and DistMult-HRS. Finally, two variants of the HRS models and implementation details are provided.

3.1 Hierarchical Relation Structure

Given a KG $\mathcal{G} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ are the entity (node) set and relation (edge) set respectively. We believe the relations in KGs can make up relation clusters as well as be split into fine-grained sub-relations. On the one hand, large scale KGs always have semantically related relations. The information from semantically similar relations is of great value and these relations should be trained in a collective way. In this way, meaningful associations among related relations can be utilized and less frequent relations can be enriched with more training data. On the other hand, some relations may have multiple semantic meanings and can be split into several sub-relations, which can provide fine-grained descriptions for each relation. In general, relations in KGs conform to a three-layer HRS, as shown in Figure 1. The HRS include a relation cluster layer, a relation layer and a sub-relation layer, which are denoted in yellow, green and blue in Figure 1 respectively.

For a triple (h, r, t) in the HRS model, the embedding of r is comprised of three parts: the relation cluster embedding $\mathbf{r}_c$, relation-specific embedding $\mathbf{r}'$ and sub-relation embedding $\mathbf{r}_s$, which is denotes as

$$\mathbf{r} = \mathbf{r}_c + \mathbf{r}' + \mathbf{r}_s. \tag{4}$$

According to the above equation, the embedding of each relation can leverage the information from the three-layer HRS. The relation clusters and sub-relations are determined by k-means algorithm based on the results of TransE:

- **Relation clusters**. We first run TransE on a given data set and obtain the embeddings of relations $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$, ..., $\mathbf{r}_{|\mathcal{R}|}$, where $|\mathcal{R}|$ is the number of relations. Then, the k-means algorithm is applied on these embeddings. In this way, we get relation clusters $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, ..., $\mathcal{C}_{|\mathcal{C}|}$, where $\mathcal{C}$ is the set of relation clusters. Previous studies have shown that the embeddings of semantically similar relations locate near each other in the latent space (Yang et al., 2015). In this way, we are able to find relation clusters composed of semantically related relations.

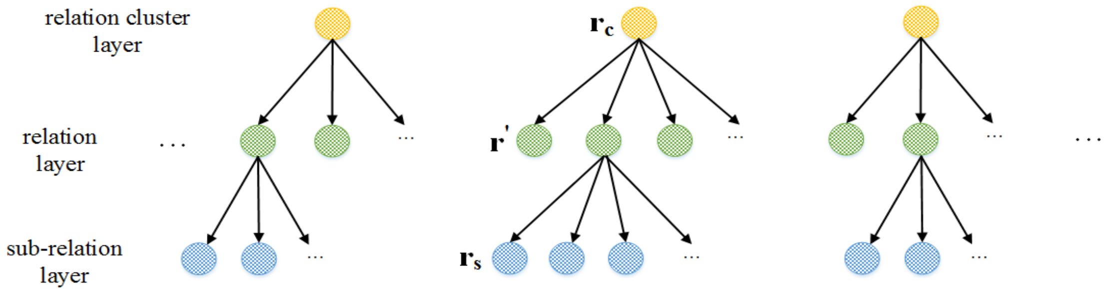

Figure 1: Hierarchical Relation Structure

- **Sub-relations**. TransE assumes that $\mathbf{t} - \mathbf{h} \approx \mathbf{r}$ when (h, r, t) holds. For each triple (h, r, t), we define that $\widehat{\mathbf{r}} = \mathbf{t} - \mathbf{h}$, where $\mathbf{h}$ and $\mathbf{t}$ are obtained from the results of TransE. For each relation, we collect all the $\widehat{\mathbf{r}}$ and adopt the k-means algorithm to cluster these vectors into several groups $\mathcal{S}_1^r$, $\mathcal{S}_2^r$, $\mathcal{S}_3^r$, ..., $\mathcal{S}_{n_r}^r$, where n_r is the number of sub-relations for relation r. Each group corresponds to a fine-grained sub-relation.

3.2 Loss Function

The loss of the extended HRS model is comprised of two parts, as is shown in Equation (5),

$$L_{Total} = L_{Orig} + L_{HRS}, \tag{5}$$

where L_{Orig} is the loss function of the original model, while L_{HRS} is the loss function for the HRS information.

We know that TransE, TransH and DistMult all adopt a margin-based ranking loss. Taking TransE as an example, the loss function of TransE for the first part L_{Orig} is shown as Equation (6),

$$L_{Orig} = \sum_{c=1}^{|\mathcal{C}|} \sum_{r \in \mathcal{C}_c} \sum_{(h,r,t) \in \triangle_r} \sum_{(h',r,t') \in \triangle'_r} [\gamma + f_r(h,t) - f_r(h',t')]_+, \tag{6}$$

where $[x]_+ = max(0, x)$, $\triangle_r$ denotes the set of positive triples for relation r and $\triangle'_r = \{(h', r, t) | h' \in \mathcal{E}\} \cup \{(h, r, t') | t' \in \mathcal{E}\}$ is the set of negative ones for relation r. γ is the margin separating the positive triples from the negative ones. $f_r(h, t)$ is the score function as shown in Equation (7),

$$f_r(h,t) = \left\| \mathbf{h} + \mathbf{r}_c + \mathbf{r}' + \mathbf{r}_s - \mathbf{t} \right\|_{L_n}, \tag{7}$$

which can be measured by L_1 or L_2 norm. Positive triples are supposed to have lower scores than negative ones.

The second part, L_{HRS}, is composed of three regularized terms, which is shown in Equation (8),

$$L_{HRS} = \lambda_1 \sum_{\mathbf{r}_c \in \mathcal{C}} \|\mathbf{r}_c\|_2^2 + \lambda_2 \sum_{\mathbf{r}' \in \mathcal{R}} \|\mathbf{r}'\|_2^2 + \lambda_3 \sum_{\mathbf{r}_s \in \mathcal{S}} \|\mathbf{r}_s\|_2^2, \tag{8}$$

where $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_{|\mathcal{C}|}\}$ is the set of relation clusters, $\mathcal{S} = \{\mathcal{S}_1^r, \mathcal{S}_2^r, \mathcal{S}_3^r, ..., \mathcal{S}_{n_r}^r | r \in \mathcal{R}\}$ is the set of fine-grained sub-relations, n_r is the number of sub-relations for relation r. λ_1, λ_2 and λ_3 are trade-off parameters. Large value of λ_1 will result in the separate training of each relation, while large value of λ_2 will lead to all relations in the same relation cluster sharing the same embedding vector. λ_3 should be larger than λ_1 and λ_2 to restrict $\mathbf{r}_s$ to be a small value, i.e., the sub-relations from the same relation should be close.

3.3 Variants of the HRS Model and Implementation details

Additionally, we introduce two variants of the HRS model: the top-middle model and the middle-bottom model. The top-middle model only uses the HRS by leveraging the information from the top to the middle layer. For this model, the relation embedding and the loss for HRS is defined as Equation (9) and (10).

$$\mathbf{r} = \mathbf{r}_c + \mathbf{r}', \tag{9}$$

$$L_{HRS} = \lambda_1 \sum_{\mathbf{r}_c \in \mathcal{C}} \|\mathbf{r}_c\|_2^2 + \lambda_2 \sum_{\mathbf{r}' \in \mathcal{R}} \|\mathbf{r}'\|_2^2. \tag{10}$$

While the middle-bottom model only utilizes the information from the middle to the bottom layer. The relation embedding and HRS loss are defined as Equation (11) and (12).

$$\mathbf{r} = \mathbf{r}' + \mathbf{r}_s, \tag{11}$$

$$L_{HRS} = \lambda_2 \sum_{\mathbf{r}' \in \mathcal{R}} \|\mathbf{r}'\|_2^2 + \lambda_3 \sum_{\mathbf{r}_s \in \mathcal{S}} \|\mathbf{r}_s\|_2^2 . \quad (12)$$

The learning process of the extended models is carried out by using the Adam (Kingma and Ba, 2014) optimizer. For the extended models of TransE, all the entity and relation embedding parameters are initialized with a uniform distribution $U\left[-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}\right]$ following TransE, where k is the dimension of the embedding space. For the extended models of TransH and DistMult, we initialize these parameters with the results of TransE. For the relation cluster embeddings and sub-relation embeddings, we initialize all the parameters with the value of zero.

4 Experiments

4.1 Data Sets

In this research, we evaluate the performances of our extended models on popular benchmarks FB15k (Bordes et al., 2013), FB15k-237 (Toutanova and Chen, 2015), FB13(Socher et al., 2013), WN18 (Bordes et al., 2013) and WN11 (Socher et al., 2013). FB15k, FB15k-237 and FB13 are extracted from Freebase (Bollacker et al., 2008), which provides general facts of the world. WN18 and WN11 are obtained from WordNet (Miller, 1994), which provides semantic knowledge of words. FB15k-237 and WN18 are used for the task of link prediction, FB13 and WN11 are used for the triple classification task, while FB15k is used for both tasks. The statistics of the five data sets are summarized in Table 1.

Table 1: Statistics of the Five Datasets.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	#triples in Train/Valid/Test
FB15k	14,951	1,345	483,142 / 50,000 / 59,071
FB15k-237	14,541	237	272,115 / 17,535 / 20,466
FB13	75,043	13	316,232 / 5,908 / 23,733
WN18	40,943	18	141,442 / 5,000 / 5,000
WN11	38,696	11	112,581 / 2,609 / 10,544

4.2 Baselines

To demonstrate the effectiveness of our models, we compare results with the following baselines.

- TransE (Bordes et al., 2013): one of the most widely used KGE models.
- TransH (Wang et al., 2014): a KGE model which adopts relation-specific hyperplanes to lay entities and relations.
- DistMult (Yang et al., 2015): a state of the art model which uses a bilinear score function to compute scores of knowledge triples.
- CTransR (Lin et al., 2015b): a pioneering KGE model which exploits fine-grained sub-relations for each relation.
- TransD (Ji et al., 2015): an improvement of CTransR, which embeds KGs using dynamic mapping matrices.
- TransG (Xiao et al., 2016): the first generative KGE model that uses a non-parametric bayesian model to embed KGs.

4.3 Link Prediction

Link prediction, a.k.a. knowledge graph completion, aims to fill the missing values into incomplete knowledge triples. More formally, the goal of link prediction is to predict either the head entity in a given query $(?, r, t)$ or the tail entity in a given query $(h, r, ?)$.

4.3.1 Experimental Settings

All the parameters are set by some preliminary test. For TransE-HRS, TransE-top-middle and TransE-middle-bottom, λ_1, λ_2, λ_3 and the margin γ are set as $\lambda_1 = 1e-5, \lambda_2 = 1e-4, \lambda_3 = 1e-3, \gamma = 2$. For the extended models of TransH, we set the parameters as $\lambda_1 = 1e-5, \lambda_2 = 1e-5, \lambda_3 = 1e-3, \gamma = 1$. For the extended models of DistMult, the parameters are set as $\lambda_1 = 1e-5, \lambda_2 = 1e-4, \lambda_3 = 1e-3, \gamma = 1$. For all the above models, the learning rate ς, batch size b and embedding size k are set as $\varsigma = 1e-3, b = 4096, k = 100$. The L_1 norm is adopted by the score function of TransE and its extended models. The number of relation clusters are set as 300, 120 and 10 for FB15k, FB15k-237 and WN18 respectively. For all the data sets, we generate 3 sub-relations for relations that have more than 500 occurrences in the training set. For all the extended models and baselines, we produce negative triples following the "bern" sampling strategy which was introduced in TransH (Wang et al., 2014). For baselines TransE, TransH and DistMult, the embedding parameters of entities and relations are initialized the same way as the extended models for a fair comparison.

In the test phase, we replace the head and tail entities with all the entities in KG in turn for each triple in the test set. Then we compute a score for each corrupted triple. Note that for each corrupted triple (h', r, t'), the sub-relation is determined by $\mathbf{t}' - \mathbf{h}'$, i.e., the k-means model is adopted to assign $\mathbf{t}' - \mathbf{h}'$ to a specific sub-relation of r. We rank all the candidate entities according to the scores. Specifically, positive candidates are supposed to precede negative ones. Finally, the rank of the correct entity is stored. We compare our models with baselines using the following metrics: (1) Mean Rank (MR, the mean of all the predicted ranks); (2) Mean Reciprocal Rank (MRR, the mean of all the reciprocals of predicted ranks); (3) Hits@n (Hn, the proportion of ranks not larger than n). Lower values of MR and larger values of MRR and Hn indicate better performance. All the results are reported in the "filtered" setting (Bordes et al., 2013).

4.3.2 Experimental Results

Evaluation results are shown in Table 2. We divide all the results into 4 groups. The second, third and forth group are results of TransE, TransH, DistMult and their extended models respectively, while the first group are results of other state-of-the-art competitors. Results in bold font are the best results in the group and the underlined results denote the best results in the column. From Table 1, we have the following findings: (1) Our extended models outperform the original models, which indicates that the information learned from the HRS is valuable; (2) For WN18, the results from 'top-middle' models of TransE, TransH and DistMult are worse than the original models, and HRS models can't outperform middle-bottom ones. We conjecture the reason lies as follows: WN18 has only 18 relations and the semantic correlation among relations is small. In this case, the information learned from the top to the middle layer of the HRS may lead to worse results since for each relation, even though the information learned from semantically similar relations are useful, the information learned from unrelated relations may damage the results. The results indicate that HRS models are especially useful for KGs with dense semantic distributions over relations; (3) For WN18, TransE-middle-bottom and DistMult-middle-bottom achieve the best results on MRR, Hits@10, Hits@3 and Hits@1 while failing to get the best results on MR in the same group. Further analysis shows that in the results of TransE-middle-bottom, 56 test triples get ranks more than 10000, leading to more than 110 MR loss. While in the results of DistMult-middle-bottom, there exist 37 test triples whose ranks are more than 7000, which would lead to about 50 MR loss. Indeed, MR is sensitive to these high ranks, which lead to worse results on the metric of MR; (4) From all the results, based on the good basic model DistMult, the extended models of DistMult can achieve the best performance compared with other state-of-the-art baselines CTransR, TransD and TransG.

We also provide some case studies on relation clusters and sub-relations. Table 3 shows some relation clusters of FB15k. Cluster 1 to 3 are Olympics-related, basketball-related and software-related relations respectively. From Table 3 we can see that semantically related relations can join the same cluster. Table 4 shows some (head, tail) pairs for the sub-relations of *'/educational_institution/education/degree'*. Sub-relation 1 to 3 are about the degree of Doctor, Master and Bachelor respectively. Table 5 gives some (head, tail) pairs for the sub-relations of *'/music/artist/genre'*. Sub-relation 1 and 2 are about rock music and pop music respectively while sub-cluster 3 is about other kinds of music. From Table 4 and 5, we can see that different sub-relations give fine-grained descriptions for each relation.

4.3.3 Parameter Study

In this section, we study the performance affected by the number of relation clusters $\mathcal{N}_1$ as well as the number of sub-relations for each relation $\mathcal{N}_2$. The results in Figure 2 and 3 clearly show that there exists an optimal value of $\mathcal{N}_1$ and $\mathcal{N}_2$ for each dataset. All three models keep achieving better results as we increase the number of clusters from 0 to the optimal value. Then, after $\mathcal{N}_1$ and $\mathcal{N}_2$ exceed the optimal point, the performance starts falling down. The reason lies as: (1) Smaller value of $\mathcal{N}_1$ leads to large-sized relation clusters. Some unrelated relations may join in the same large-sized cluster and degrade the performance of our models. Larger value of $\mathcal{N}_1$ leads to small-sized relation clusters, thus less information can be leveraged by each relation, leading to the unsatisfying performance; (2) Smaller value of $\mathcal{N}_2$ can't provide sufficient representations for each relation and degrade the performance of our models. Larger value of $\mathcal{N}_2$ may lead to lacking of training data for each sub-relation and also result in the unsatisfying performance.

Table 2: Link prediction results on FB15k, FB15k-237 and WN18. We implement TransE, TransH, DistMult and their extended models by ourselves. The code of CTransR, TransD and TransG are taken from https://github.com/thunlp/TensorFlow-TransX, https://github.com/thunlp/KB2E and https://github.com/BookmanHan/Embedding respectively.

	FB15k					FB15k-237					WN18				
	MR	MRR	H10	H3	H1	MR	MRR	H10	H3	H1	MR	MRR	H10	H3	H1
CTransR	81	0.408	0.740	0.573	0.314	279	0.298	0.469	0.301	0.198	228	0.816	0.923	0.842	0.316
TransD	90	0.658	0.781	0.586	0.324	256	0.286	0.453	0.291	0.179	215	0.823	0.928	0.851	0.336
TransG	101	0.672	0.802	0.591	0.322	309	0.304	0.471	0.298	0.182	466	0.830	0.936	0.876	0.764
TransE	91	0.404	0.688	0.493	0.251	375	0.207	0.377	0.227	0.125	**387**	0.408	0.925	0.725	0.067
TransE-top-middle	61	0.463	0.730	0.556	0.315	286	0.258	0.440	0.286	0.170	609	0.402	0.919	0.710	0.058
TransE-middle-bottom	51	0.493	0.738	0.582	0.355	232	0.310	0.486	0.332	0.202	474	**0.496**	**0.945**	**0.890**	**0.112**
TransE-HRS	**49**	**0.510**	**0.767**	**0.610**	**0.361**	230	**0.311**	**0.487**	0.353	**0.215**	477	0.490	0.943	0.883	0.106
TransH	63	0.394	0.713	0.519	0.210	311	0.211	0.386	0.224	0.132	388	0.437	0.919	0.832	0.039
TransH-top-middle	65	0.477	0.737	0.561	0.308	275	0.272	0.461	0.291	0.185	411	0.416	0.890	0.813	0.034
TransH-middle-bottom	50	0.469	0.742	0.583	0.343	271	0.269	0.466	0.286	0.191	**283**	**0.491**	**0.942**	**0.880**	**0.113**
TransH-HRS	47	**0.509**	**0.783**	**0.639**	**0.390**	**243**	**0.309**	**0.491**	**0.346**	**0.216**	296	0.482	0.940	0.861	0.097
DistMult	95	0.642	0.813	0.726	0.523	251	0.244	0.423	0.261	0.159	261	0.806	0.931	0.904	0.713
DistMult-top-middle	85	0.677	0.830	0.746	0.589	243	0.286	0.461	0.291	0.192	246	0.769	0.903	0.853	0.681
DistMult-middle-bottom	83	0.682	0.828	0.758	0.606	246	0.291	0.475	0.306	0.199	226	**0.912**	**0.947**	**0.913**	**0.879**
DistMult-HRS	**72**	**0.739**	**0.846**	**0.799**	**0.661**	**232**	**0.315**	**0.496**	**0.350**	**0.241**	**206**	0.891	0.932	0.901	0.736

Table 3: Examples of Relation Clusters in FB15k

	relations
1	/olympics/olympic_athlete/medals_won./olympics/olympic_medal_honor/country /olympics/olympic_athlete/country./olympics/olympic_athlete_affiliation/country
2	/sports/sports_team/roster./basketball/basketball_roster_position/player, /basketball/basketball_team/roster./sports/sports_team_roster/player /basketball/basketball_team/roster./basketball/basketball_roster_position/player
3	/computer/software/developer, /computer/operating_system/developer, /cvg/computer_videogame/developer

Table 4: Examples of Sub-relations for Relation '/educational_institution/education/degree' in FB15k

	(head, tail)
1	(Munich Institute of Technology, Doctors of Medicine), (California Institute of Technology, Higher Doctorate), ...
2	(Central Michigan College of Education, M.Sc.), (The University of Pittsburgh, M.Sc.), ...
3	(University of Massachusetts, Amherst, Bachelor's Degree), (New Mexico State College, Bachelor's Degree), ...

Table 5: Examples of Sub-relations for Relation '/music/artist/genre' in FB15k

	(head, tail)
1	(Steve Stills, Rock Music), (Velvet Underground, Rock Music), (Benjamin Chase Harper, Rock Music), ...
2	(Justin Beiber, Pop Music), (Natalie Maria Cole, Pop Music), (Peter Thorkelson, Pop Music), ...
3	(Billy Preston, R & B), (Earth Wind Fire, Funk Rap), (Alvin Joiner, Hip-hop), ...

4.4 Triple Classification

In order to testify the discriminative capability of our models, we conduct a triple classification task aiming to predict the label (True or False) of a given triple (h, r, t).

4.4.1 Experimental Settings

In this paper, we use three datasets WN11, FB13 and FB15k to evaluate our models. The data sets WN11 and FB13 released by NTN (Socher et al., 2013) already have negative triples. The test set of FB15k only contains correct triples, which requires us to construct negative triples. In this study, we construct negative triples following the same setting used for FB13 (Socher et al., 2013). For the extended models of TransE, λ_1, λ_2, λ_3 and γ are set as $\lambda_1 = 1e-5$, $\lambda_2 = 1e-5$, $\lambda_3 = 1e-3$ and $\gamma = 4$. For the extended models of TransH, we set $\lambda_1 = 1e-5$, $\lambda_2 = 1e-4$, $\lambda_3 = 1e-3$ and $\gamma = 5$. While for the extended models of DistMult, parameters are set as $\lambda_1 = 1e-5$, $\lambda_2 = 1e-4$, $\lambda_3 = 1e-2$ and $\gamma = 4$. For WN11 and FB13, we generate 2 sub-relations for each relation. For FB15k, we generate 3 sub-relations for

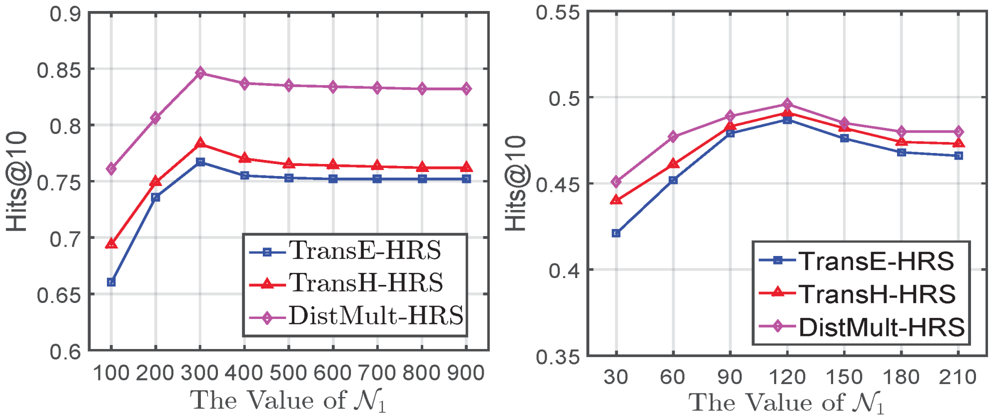

(a) FB15k (b) FB15k-237

Figure 2: The Change of Hits@10 with The Value of $\mathcal{N}_1$ Increasing.

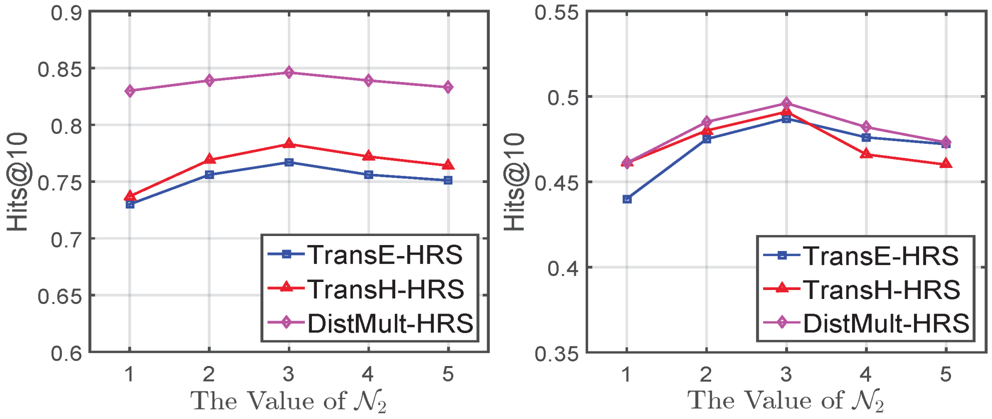

(a) FB15k (b) FB15k-237

Figure 3: The Change of Hits@10 with The Value of $\mathcal{N}_2$ Increasing.

relations that have more than 500 occurrences in the training set. Other parameters are set as introduced in Section 4.3.1. We follow the same decision process as NTN (Socher et al., 2013): for TransE and TransH, a triple is predicted to be positive if $f_r(h,t)$ is below a threshold, while for DistMult, a triple is regarded as a positive one if $f_r(h,t)$ is above a threshold; otherwise negative. The thresholds are determined on the validation set. We adopt accuracy as our evaluation metric.

4.4.2 Experimental Results

Finally, the evaluation results in Table 6 lead to the following findings: (1) Our models outperform other baselines on WN11 and FB15k, and obtain comparable results with baselines on FB13, which validate the effectiveness of our models; (2) The extended models TransE-HRS, TransH-HRS and DistMult-HRS achieve substantial improvements against the original models. On WN11, TransE-HRS outperforms TransE with a margin as large as 10.9%. These improvements indicates the technique of utilizing the HRS information is capable to be extended to different KGE models. Figure 4

Table 6: Triple Classification Results. The results of baselines on WN11 and FB13 are directly taken from the original paper except DistMult. We obtain other results by ourselves.

Model	WN11	FB13	FB15k	Avg
CTransR	85.7	-	84.4	-
TransD	86.4	89.1	88.2	87.9
TransG	87.4	87.3	88.5	87.7
TransE	75.9	81.5	78.7	78.7
TransH	78.8	83.3	81.1	81.1
DistMult	87.1	86.2	86.3	86.5
TransE-HRS	86.8	88.4	87.6	87.6
TransH-HRS	**87.6**	88.9	**88.7**	**88.4**
DistMult-HRS	**88.9**	89.0	**89.1**	**89.0**

shows the classification accuracy of different relations on WN11. We can see that extended models significantly improve the original models in each relation classification task, which again validate the effectiveness of our models.

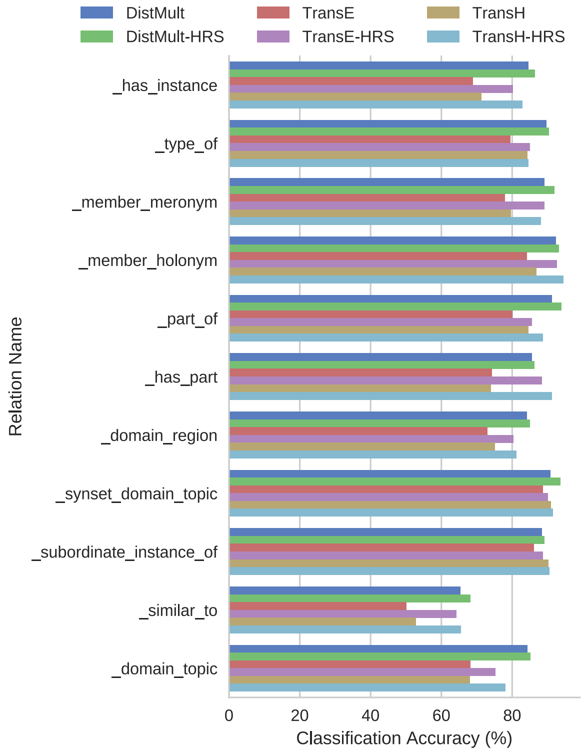

Figure 4: Classification Accuracies of Different Relations on WN11

5 Conclusion

In this paper, we found that relations in KGs conform to a three-layer HRS. This HRS model provides a critical capacity for embedding entities and relations, and along this line we extended three state-of-the-art models to leverage the HRS information. The technique we used can be easily applied to extend other KGE models. Moreover, our proposed models don't need additional information like text or paths, instead, we made full use of the knowledge triples in KGs and the rich information from the HRS. We evaluate our model on the link prediction task and triple classification task. The results show that our extended models achieve substantial improvements against the original models as well as other baseline competitors.

In the future, we will utilize more sophisticated models to leverage the HRS information, e.g, (1) utilize the embeddings of the three layers in a more sophisticated way instead of sum them together; (2) determine the number of relation clusters and sub-relations automatically instead of manually.

Acknowledgements

The research work is supported by the National Key Research and Development Program of China under Grant No. 2018YFB1004300, the National Natural Science Foundation of China under Grant No. 61773361, 61473273, 91546122, Guangdong provincial science and technology plan projects under Grant No. 2015 B010109005, the Project of Youth Innovation Promotion Association CAS under Grant No. 2017146. This work is also partly supported by the funding of WeChat cooperation project. We thank Bo Chen, Leyu Lin, Cheng Niu, Xiaohu Cheng for their constructive advices.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Antoine Bordes, Xavier Glorot, and Jason Weston. 2012a. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2012b. Learning structured embeddings of knowledge bases. In *AAAI*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

George A. Miller. 1994. Wordnet: a lexical database for english. In *The Workshop on Human Language Technology*.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *AAAI*.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. In *arXiv*.

Gavin Sherlock. 2009. Gene ontology: tool for the unification of biology. *Canadian Institute of Food Science and Technology Journal*, 22(4):415.

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *AAAI*.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago:a core of semantic knowledge. In *WWW*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *The Workshop on Continuous Vector Space MODELS and Their Compositionality*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. Transg : A generative model for knowledge graph embedding. In *ACL*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2017. Ssp: Semantic space projection for knowledge graph embedding with text descriptions. In *AAAI*.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*.

Bishan Yang, Wentau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Embedding Multimodal Relational Data for Knowledge Base Completion

Pouya Pezeshkpour
University of California
Irvine, CA
pezeshkp@uci.edu

Liyan Chen
University of California
Irvine, CA
liyanc@uci.edu

Sameer Singh
University of California
Irvine, CA
sameer@uci.edu

Abstract

Representing entities and relations in an embedding space is a well-studied approach for machine learning on relational data. Existing approaches, however, primarily focus on simple link structure between a finite set of entities, ignoring the variety of data types that are often used in knowledge bases, such as text, images, and numerical values. In this paper, we propose multimodal knowledge base embeddings (MKBE) that use different neural encoders for this variety of observed data, and combine them with existing relational models to learn embeddings of the entities and multimodal data. Further, using these learned embedings and different neural *decoders*, we introduce a novel multimodal imputation model to generate missing multimodal values, like text and images, from information in the knowledge base. We enrich existing relational datasets to create two novel benchmarks that contain additional information such as textual descriptions and images of the original entities. We demonstrate that our models utilize this additional information effectively to provide more accurate link prediction, achieving state-of-the-art results with a considerable gap of 5-7% over existing methods. Further, we evaluate the quality of our generated multimodal values via a user study. We have release the datasets and the open-source implementation of our models at https://github.com/pouyapez/mkbe.

1 Introduction

Knowledge bases (KB) are an essential part of many computational systems with applications in search, structured data management, recommendations, question answering, and information retrieval. However, KBs often suffer from incompleteness, noise in their entries, and inefficient inference under uncertainty. To address these issues, learning relational knowledge representations has been a focus of active research (Bordes et al., 2011, 2013; Yang et al., 2015; Nickel et al., 2016; Trouillon et al., 2016; Dettmers et al., 2018). These approaches represent relational triples, that consist of a subject entity, relation, and an object entity, by learning fixed, low-dimensional representations for each entity and relation from observations, encoding the uncertainty and inferring missing facts accurately and efficiently. The subject and the object entities come from a fixed, enumerable set of entities that appear in the knowledge base.

Knowledge bases in the real world, however, contain a wide variety of data types beyond these direct links. Apart from relations to a fixed set of entities, KBs often not only include numerical attributes (such as ages, dates, financial, and geoinformation), but also textual attributes (such as names, descriptions, and titles/designations) and images (profile photos, flags, posters, etc.). These different types of data can play a crucial role as extra pieces of evidence for knowledge base completion. For example the textual descriptions and images might provide evidence for a person's age, profession, and designation. In the multimodal KB shown in Figure 1 for example, the image can be helpful in predicting of Carles Puyol's occupation, while the description contains his nationality. Incorporating this information into existing approaches as entities, unfortunately, is challenging as they assign each entity a distinct vector and predict missing links (or attributes) by enumerating over the possible values, both of which are only possible if the entities come from a small, enumerable set. There is thus a crucial need for relational modeling that goes beyond just the link-based view of KB completion, by not only utilizing multimodal information for better link prediction between existing entities, but also being able to generate missing multimodal values.

In this paper, we introduce *multimodal knowl-*

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3208–3218
Brussels, Belgium, October 31 - November 4, 2018.

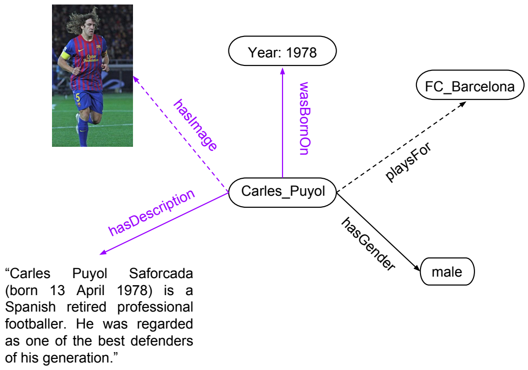

Figure 1: **Example of a Multimodal KB.** Graph representation of (a part of) a KB that consists of regular links (in black) and multimodal ones (in purple) that we support in this work.

edge base embeddings (MKBE) for modeling knowledge bases that contain a variety of data types, such as links, text, images, numerical, and categorical values. We propose neural encoders and decoders to replace initial layers of any embedding-based relational model; we apply them to DistMult (Yang et al., 2015) and ConvE (Dettmers et al., 2018) here. Specifically, instead of learning a distinct vector for each entity and using enumeration to predict links, MKBE includes the following extensions: (1) introduce additional neural encoders to embed multimodal evidence types that the relational model uses to predict links, and (2) introduce neural decoders that use an entity's embedding to generate its multimodal attributes (like image and text). For example, when the object of a triple is an image, we encode it into a fixed-length vector using a CNN, while textual objects are encoded using RNN-based sequence encoders. The scoring module remains identical to the underlying relational model; given the vector representations of the subject, relation, and object of a triple, we produce a score indicating the probability that the triple is correct using DistMult or ConvE. After learning the KB representation, neural decoders use entity embeddings to generate missing multimodal attributes, for example, generating the description of a person from their structured information in the KB. This unified framework allows for flow of the information across the different relation types (multimodal or otherwise), providing a more accurate modeling of relational data.

We provide an evaluation of our proposed approach on two relational KBs. Since we are introducing the multimodal KB completion setting, we provide two benchmarks, created by extending the existing YAGO-10 and MovieLens-100k datasets to include additional relations such as textual descriptions, numerical attributes, and images of the entities. We demonstrate that MKBE utilizes the additional information effectively to provide gains in link-prediction accuracy, achieving state-of-the-art results on these datasets for both the DistMult and the ConvE scoring functions. We evaluate the quality of multimodal attributes generated by the decoders via user studies that demonstrate their realism and information content, along with presenting examples of such generated text and images.

2 Multimodal KB Completion

As described earlier, KBs often contain different types of information about entities including links, textual descriptions, categorical attributes, numerical values, and images. In this section, we briefly introduce existing relational embedding approaches that focus on modeling the linked data using distinct, dense vectors. We then describe MKBE that extends these approaches to the multimodal setting, i.e., modeling the KB using all the different information to predict the missing links and impute the missing attributes.

2.1 Background on Link Prediction

Factual statements in a knowledge base are represented using a triple of subject, relation, and object, $\langle s, r, o\rangle$, where $s, o \in \xi$, a set of entities, and $r \in \mathcal{R}$, a set of relations. Respectively, we consider two goals for relational modeling, (1) to train a machine learning model that can score the *truth* value of any factual statement, and (2) to predict missing links between the entities. In existing approaches, a scoring function $\psi : \xi \times \mathcal{R} \times \xi \rightarrow \mathbb{R}$ (or sometimes, $[0, 1]$) is learned to evaluate whether any given fact is true, as per the model. For predicting links between the entities, since the set ξ is small enough to be enumerated, missing links of the form $\langle s, r, ?\rangle$ are identified by enumerating all the objects and scoring the triples using ψ (i.e. assume the resulting entity comes from a known set). For example, in Figure 1, the goal is to predict that Carles Puyol plays for Barcelona.

Many of the recent advances in link prediction use an embedding-based approach; each entity in ξ and relation in $\mathcal{R}$ are assigned distinct, dense vectors, which are then used by ψ to compute the score. In DistMult (Yang et al., 2015), for example, each entity i is mapped to a d-dimensional dense vector

($\mathbf{e}_i \in \mathbb{R}^d$) and each relation r to a diagonal matrix $\mathbf{R}_r \in \mathbb{R}^{d\times d}$, and consequently, the score for any triple $\langle s, r, o\rangle$ is computed as $\psi(s, r, o) = \mathbf{e}_s^T \mathbf{R}_r \mathbf{e}_o$. Along similar lines, ConvE (Dettmers et al., 2018) uses vectors to represent the entities and the relations, $\mathbf{e}_s, \mathbf{e}_o, \mathbf{r}_r \in \mathbb{R}^{d\times 1}$, then, after applying a CNN layer on $\mathbf{e}_s$ and $\mathbf{r}_r$, combines it with $\mathbf{e}_o$ to score a triplet, i.e. the scoring function $\psi(s, r, o)$ is $f(\text{vec}(f([\bar{\mathbf{e}_s}; \bar{\mathbf{r}_r} * w]))W)\mathbf{e}_o$. Other relational embedding approaches primarily vary in their design of the scoring function (Bordes et al., 2013; Yang et al., 2015; Nickel et al., 2016; Trouillon et al., 2016), but share the shortcoming of assigning distinct vectors to every entity, and assuming that the possible object entities can be enumerated. In this work we focus on DistMult because of its simplicity, popularity, and high accuracy, and ConvE because of its state-of-the-art results.

2.2 Problem Setup

When faced with additional triples in form of multimodal data, the setup of link prediction is slightly different. Consider a set of all potential multimodal objects, $\mathcal{M}$, i.e. possible images, text, numerical, and categorical values, and multimodal evidence triples, $\langle s, r, o\rangle$, where $s \in \xi$, $r \in \mathcal{R}$, and $o \in \mathcal{M}$. Our goals with incorporating multimodal information into KB remain the same: we want to be able to score the truth of any triple $\langle s, r, o\rangle$, where o is from ξ (link data) or from $\mathcal{M}$ (multimodal data), and to be able to predict missing value $\langle s, r, ?\rangle$ that may be from ξ or $\mathcal{M}$ (depending on r). For the example in Figure 1, in addition to predicting that Carles Puyol plays for Barcelona from multimodal evidence, we are also interested in generating an image for Carles Puyol, if it is missing.

Existing approaches to this problem assume that the subjects and the objects are from a fixed set of entities ξ, and thus are treated as indices into that set, which fails for the multimodal setting primarily for two reasons. First, learning distinct vectors for each object entity does not apply to multimodal values as they will ignore the actual content of the multimodal attribute. For example, there will be no way to generalize vectors learned during training to unseen values that might appear in the test; this is not a problem for the standard setup due to the assumption that all entities have been observed during training. Second, in order to predict a missing *multimodal* value, $\langle s, r, ?\rangle$, enumeration is not possible as the search space is potentially infinite (or at least intractable to search).

2.3 Multimodal KB Embeddings (MKBE)

To incorporate such multimodal *objects* into the existing relational models like DistMult and ConvE, we propose to learn embeddings for these types of data as well. We utilize recent advances in deep learning to construct *encoders* for these objects to represent them, essentially providing an embedding $\mathbf{e}_o$ for any object value.

The overall goal remains the same: the model needs to utilize all the observed subjects, objects, and relations, across different data types, in order to estimate whether any fact $\langle s, r, o\rangle$ holds. We present an example of an instantiation of MKBE for a knowledge base containing YAGO entities in Figure 2a. For any triple $\langle s, r, o\rangle$, we embed the subject (*Carles Puyol*) and the relation (such as playsFor, wasBornOn, or playsFor) using a direct lookup. For the object, depending on the domain (indexed, string, numerical, or image, respectively), we use approrpiate encoders to compute its embedding $\mathbf{e}_o$. As in DistMult and ConvE, these embeddings are used to compute the score of the triple.

Via these neural encoders, the model can use the information content of multimodal objects to predict missing links where the objects are from ξ, however, learning embeddings for objects in $\mathcal{M}$ is not sufficient to generate missing multimodal values, i.e. $\langle s, r, ?\rangle$ where the object is in $\mathcal{M}$. Consequently, we introduce a set of neural decoders $D : \xi \times \mathcal{R} \rightarrow \mathcal{M}$ that use entity embeddings to *generate* multimodal values. An outline of our model for imputing missing values is depicted in Figure 2b. We will describe these decoders in Section 2.5.

2.4 Encoding Multimodal Data

Here we describe the encoders we use for multimodal objects. A simple example of MKBE is provided in Figure 2a. As it shows, we use different encoder to embed each specific data type.

Structured Knowledge Consider a triplet of information in the form of $\langle s, r, o\rangle$. To represent the subject entity s and the relation r as independent embedding vectors (as in previous work), we pass their one-hot encoding through a dense layer. Furthermore, for the case that the object entity is categorical, we embed it through a dense layer with a recently introduced selu activation (Klambauer et al., 2017), with the same number of nodes as the

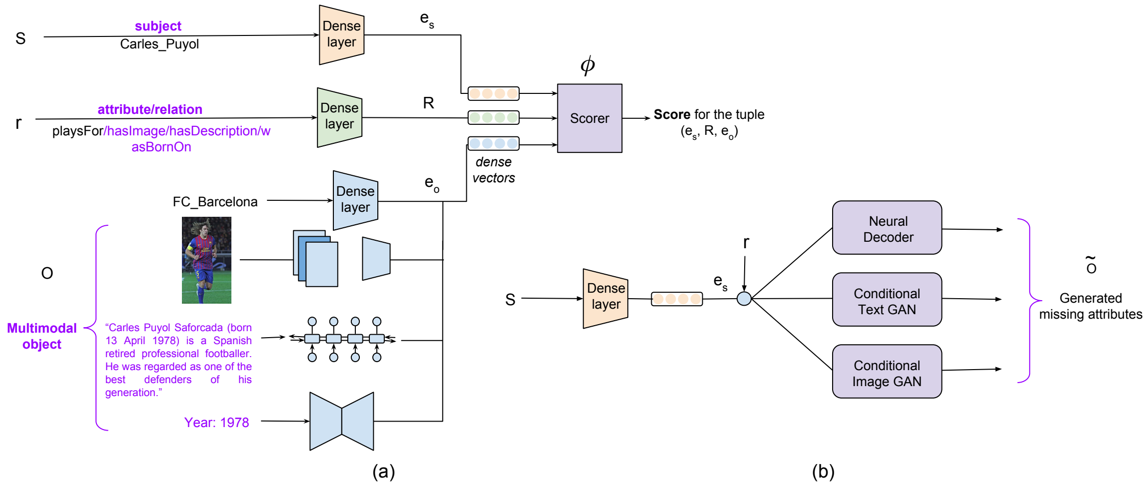

Figure 2: **Multimodal KB Embeddings (MKBE):** (a) Proposed architecture that, given any entity and its relations, uses domain-specific encoders to embed each object. The embeddings of entities, and the relation are then used to score the *truth* value of the triple by the *Scorer*. (b) Architecture of the proposed work for multimodal attributes recovery. Given an entity, we use its learned embeddings from (a) as the context for attribute-specific decoders to generate the missing values.

embedding space dimension.

Numerical Objects in the form of real numbers can provide a useful source of information and are often quite readily available. We use a feed forward layer, after standardizing the input, in order to *embed* the numbers (in fact, we are projecting them to a higher-dimensional space, from $\mathbb{R} \rightarrow \mathbb{R}^d$). It is worth noting that existing methods treat numbers as distinct *entities*, e.g., learn independent vectors for numbers 39 and 40, relying on data to learn that these values are similar to each other.

Text Since text can be used to store a wide variety of different types of information, for example names versus paragraph-long descriptions, we create different encoders depending on the lengths of the strings involved. For attributes that are fairly short, such as names and titles, we use character-based stacked, bidirectional GRUs to encode them, similar to Verga et al. (2016), using the final output of the top layer as the representation of the string. For strings that are much longer, such as detailed descriptions of entities consisting of multiple sentences, we treat them as a sequence of words, and use a CNN over the word embeddings, similar to Francis-Landau et al. (2016), in order to learn the embedding of such values. These two encoders provide a fixed length encoding that has been shown to be an accurate semantic representation of strings for multiple tasks (Dos Santos and Gatti, 2014).

Images Images can also provide useful evidence for modeling entities. For example, we can extract person's details such as gender, age, job, etc., from image of the person (Levi and Hassner, 2015), or location information such as its approximate coordinates, neighboring locations, and size from map images (Weyand et al., 2016). A variety of models have been used to compactly represent the semantic information in the images, and have been successfully applied to tasks such as image classification, captioning (Karpathy and Fei-Fei, 2015), and question-answering (Yang et al., 2016). To embed images such that the encoding represents such semantic information, we use the last hidden layer of VGG pretrained network on Imagenet (Simonyan and Zisserman, 2015), followed by compact bilinear pooling (Gao et al., 2016), to obtain the embedding of the images.

Training We follow the setup from Dettmers et al. (2018) that consists of binary cross-entropy loss without negative sampling for both ConvE and DisMult scoring. In particular, for a given subject-relation pair (s, r), we use a binary *label* vector $\mathbf{t}^{s,r}$ over all entities, indicating whether $\langle s, r, o\rangle$ is observed during training. Further, we denote the model's probability of truth for any triple $\langle s, r, o\rangle$ by $p_o^{s,r}$, computed using a sigmoid over $\psi(s, r, o)$. The binary cross-entropy loss is thus defined as:

$$\sum_{(s,r)} \sum_{o} t_o^{s,r} \log(p_o^{s,r}) + (1 - t_o^{s,r}) \log(1 - p_o^{s,r}).$$

We use the same loss for multimodal triples as well, except that the summation is restricted to the objects of the same modality, i.e. for an entity s

and its text description, $\mathbf{t}^{s,r}$ is a one-hot vector over all descriptions observed during training.

2.5 Decoding Multimodal Data

Here we describe the decoders we use to generate multimodal values for entities from their embeddings. The multimodal imputing model is shown in Figure 2b, which uses different neural decoders to generate missing attributes (more details are provided in supplementary materials).

Numerical and Categorical data To recover the missing numerical and categorical data such as dates, gender, and occupation, we use a simple feed-forward network on the entity embedding to predict the missing attributes. In other words, we are asking the model, if the actual birth date of an entity is not in the KB, what will be the most likely date, given the rest of the relational information. These decoders are trained with embeddings from Section 2.4, with appropriate losses (RMSE for numerical and cross-entropy for categories).

Text A number of methods have considered generative adversarial networks (GANs) to generate grammatical and linguistically coherent sentences (Yu et al., 2017; Rajeswar et al., 2017; Guo et al., 2017). In this work, we use the adversarially regularized autoencoder (ARAE) (Zhao et al., 2017) to train generators that decodes text from continuous codes, however, instead of using the random noise vector z, we condition the generator on the entity embeddings.

Images Similar to text recovery, to find the missing images we use conditional GAN structure. Specifically, we combine the BE-GAN (Berthelot et al., 2017) structure with pix2pix-GAN (Isola et al., 2017) model to generate high-quality images, conditioning the generator on the entity embeddings in the knowledge base representation.

3 Related Work

There is a rich literature on modeling knowledge bases using low-dimensional representations, differing in the operator used to score the triples. In particular, they use matrix and tensor multiplication (Nickel et al., 2011; Yang et al., 2015; Socher et al., 2013), Euclidean distance (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015), circular correlation (Nickel et al., 2016), or the Hermitian dot product (Trouillon et al., 2016) as scoring function. However, the *objects* for all of these approaches are a fixed set of entities, i.e., they only embed the structured links between the entities. Here, we use different types of information (text, numerical values, images, etc.) in the encoding component by treating them as relational triples.

Table 1: **Data Statistics** of the two benchmark datasets we are using. The numbers in bold are our contributions to the datasets.

	MovieLens	**YAGO-10**
#Link Types	13	45
#Entities	2,625	123,182
#Link Triples	100,000	1,079,040
#Numerical Attributes	2,625	**111,406**
#Image Attributes	**1,651**	**61,246**
#Text Attributes	1,682	**107,326**

A number of methods utilize an extra type of information as the observed features for entities, by either merging, concatenating, or averaging the entity and its features to compute its embeddings, such as numerical values (Garcia-Duran and Niepert, 2017) (we use KBLN from this work to compare it with our approach using only numerical as extra attributes), images (Xie et al., 2017; Oñoro-Rubio et al., 2017) (we use IKRL from the first work to compare it with our approach using only images as extra attributes), text (McAuley and Leskovec, 2013; Zhong et al., 2015; Toutanova et al., 2015, 2016; Xie et al., 2016; Tu et al., 2017), and a combination of text and image (Sergieh et al., 2018). Further, Verga et al. (2016) address the multilingual relation extraction task to attain a universal schema by considering raw text with no annotation as extra feature and using matrix factorization to jointly embed KB and textual relations (Riedel et al., 2013). In addition to treating the extra information as features, graph embedding approaches (Schlichtkrull et al., 2017; Kipf and Welling, 2016) consider observed attributes while encoding to achieve more accurate embeddings.

The difference between MKBE and these mentioned approaches is three-fold: (1) we are the first to use different types of information in a unified model, (2) we treat these different types of information (numerical, text, image) as relational triples of structured knowledge instead of predetermined features, i.e., first-class citizens of the KB, and not auxiliary features, and (3) our model represents uncertainty in them, supporting the missing values and facilitating recovery of missing values.

Table 2: **Rating Prediction in MovieLens.** Results for models that use: rating information (R), movie-attribute (M), user-attribute (U), movies' title text (T), and poster images (P).

	Models	MRR	Hits@1	Hits@2	RMSE
DistMult	Ratings Only, R	0.62	0.40	0.69	1.48
	R+M+U	0.646	0.423	0.708	1.37
	R+M+U+T	0.650	**0.424**	**0.73**	**1.23**
	R+M+U+P	**0.652**	0.413	0.712	1.27
	R+M+U+T+P	0.644	0.42	0.72	1.3
ConvE	Ratings Only, R	0.683	0.47	0.81	1.47
	R+M+U	0.702	0.49	0.83	1.39
	R+M+U+T	**0.728**	**0.513**	**0.85**	1.13
	R+M+U+P	0.726	0.512	0.83	1.13
	R+M+U+T+P	0.726	0.512	0.84	**1.09**

4 Evaluation Benchmarks

To evaluate the performance of our multimodal relational embeddings approach, we provide two new benchmarks by extending existing datasets. Table 1 provides the statistics of these datasets.

MovieLens-100k dataset (Harper and Konstan, 2016) is a popular benchmark in recommendation systems to predict user ratings with contextual features, containing around 1000 users on 1700 movies. MovieLens already contains rich relational data about occupation, gender, zip code, and age for users and genre, release date, and the titles for movies. We augment this data with movie posters collected from TMDB (https://www.themoviedb.org/). We treat the 5-point ratings as five different relations in KB triple format, i.e., $\langle \text{user}, r = 5, \text{movie} \rangle$, and evaluate the rating predictions as other relations are introduced.

YAGO-10 Even though MovieLens has a variety of data types, it is still quite small, and is over a specialized domain. We also consider a second dataset that is much more appropriate for knowledge graph completion and is popular for link prediction, the YAGO3-10 knowledge graph (Suchanek et al., 2007; Nickel et al., 2012). This graph consists of around 120,000 entities, such as people, locations, and organizations, and 37 relations, such as kinship, employment, and residency, and thus much closer to the traditional information extraction goals. We extend this dataset with the textual description (as an additional relation) and the images associated with each entity (for half of the entities), provided by *DBpedia* (Lehmann et al., 2015). We also include additional relations such as `wasBornOnDate` that have dates as values.

Table 3: **Link Prediction in YAGO-10.** Results shown for models using: structured information (S), textual description of the entities (D), dates as numerical information (N), and images (I). Published refers to Dettmers et al. (2018).

	Models	MRR	Hits@1	Hits@3	Hits@10
DistMult	Published	0.337	0.237	0.379	0.54
	Links only, S	0.326	0.221	0.375	0.538
	S+D	0.36	0.262	0.395	0.571
	S+N	0.325	0.213	0.382	0.517
	S+I	0.342	0.235	0.352	0.618
	S+D+N	0.359	0.243	0.401	0.679
	S+D+N+I	**0.372**	**0.268**	**0.418**	**0.792**
ConvE	Published	0.523	0.448	0.564	0.658
	Links only, S	0.482	0.372	0.519	0.634
	S+D	0.564	0.478	0.595	0.713
	S+N	0.549	0.462	0.587	0.701
	S+I	0.566	0.471	0.597	0.72
	S+D+N	**0.588**	0.517	0.603	**0.722**
	S+D+N+I	0.584	**0.52**	**0.604**	0.698
	KBLN	0.503	0.41	0.549	0.658
	IKRL	0.509	0.423	0.556	0.663

5 Experiment Results

In this section, we first evaluate the ability of MKBE to utilize the multimodal information by comparing to DistMult and ConvE through a variety of tasks. Then, by considering the recovery of missing multimodal values (text, images, and numerical) as the motivation, we examine the capability of our models in generation. Details of the hyperparameters and model configurations is provided in the supplementary material, and the source code and the datasets to reproduce the results is available at https://github.com/pouyapez/mkbe.

5.1 Link Prediction

In this section, we evaluate the capability of MKBE in the link prediction task. The goal is to calculate MRR and Hits@ metric (ranking evaluations) of recovering the missing entities from triples in the test dataset, performed by ranking all the entities and computing the rank of the correct entity. Similar to previous work, here we focus on providing the results in a filtered setting, that is we only rank triples in the test data against the ones that never appear in either train or test datasets.

MovieLens-100k We train the model using `Rating` as the relation between users and movies. We use a character-level GRU for the movie titles, a separate feed-forward network for age, zip code, and release date, and finally, we use a VGG network on the posters (for every other relation we use a dense layer). Table 2 shows the link (rating) prediction

Table 4: **Per-Relation Breakdown** showing performance of each model on different relations.

Relation	**Links Only**		**+Numbers**		**+Description**		**+Images**	
	MRR	Hits@1	MRR	Hits@1	MRR	Hits@1	MRR	Hits@1
isAffiliatedTo	0.524	0.401	0.551	0.467	**0.572**	**0.481**	0.569	0.478
playsFor	0.528	0.413	0.554	0.471	**0.574**	**0.486**	0.566	0.476
hasGender	0.798	0.596	0.799	0.599	0.813	0.627	**0.842**	**0.683**
isConnectedTo	0.482	0.367	**0.497**	0.379	0.492	**0.384**	0.484	0.372
isMarriedTo	0.365	0.207	0.387	0.221	0.404	0.296	**0.413**	**0.326**

evaluation on MovieLens when test data is consisting only of rating triples. We calculate our metrics by ranking the five relations that represent ratings instead of object entities. We label models that use ratings as R, movie-attributes as M, user-attributes as U, movie titles as T, and posters as P. As shown, the model R+M+U+T outperforms others with a considerable gap demonstrating the importance of incorporating extra information. Hits@1 for the baseline is *40%*, matching existing recommendation systems (Guimerà et al., 2012). From these results, we see that the models benefit more from titles as compared to the posters.

YAGO-10 The result of link prediction on our YAGO dataset is provided in Table 3. We label models using structured information as S, entity-description as D, numerical information as N, and entity-image as I. We see that the model that encodes all type of information consistently performs better than other models, indicating that the model is effective in utilizing the extra information. On the other hand, the model that uses only text performs the second best, suggesting the entity descriptions contain more information than others. It is notable that model S is outperformed by all other models, demonstrating the importance of using different data types for attaining higher accuracy. This observation is consistent across both DistMult and ConvE, and the results obtained on ConvE are the new state-of-art for this dataset (as compared to Dettmers et al. (2018)). Furthermore, we implement KBLN (Garcia-Duran and Niepert, 2017) and IKRL (Xie et al., 2017) to compare them with our S+N and S+I models. Our models outperform these approaches, in part because both of these methods require same multimodal attributes for both of the subject and object in each triple.

Relation Breakdown We perform additional analysis on the YAGO dataset to gain a deeper understanding of the performance of our model using ConvE method. Table 4 compares our models on some of the most frequent relations. As shown, the model that includes textual description significantly benefits isAffiliatedTo, and playsFor relations, as this information often appears in text. Moreover, images are useful for hasGender and isMarriedTo, while for the relation isConnectedTo, numerical (dates) are more effective than images.

Table 5: **Predicting Numbers and Categories** for YAGO (dates) and MovieLens (genres), using models with access with different information.

Models	**Search**	**Decoding**
S+N	62.49	58.7
S+N+D	59.42	56.2
S+N+I	59.86	55.8
All Info	**57.62**	**54.1**

(a) RMSE (years) in YAGO

Models	**Accuracy**
R+M	71.82
R+M+U	71.98
R+M+U+T	73.01
R+M+U+P	73.77
All Info	**75.89**

(b) Genres in MovieLens

5.2 Imputing Multimodal Attributes

Here we present an evaluation on imputing multimodal attributes (text, image and numerical).

Numerical and Categorical Table 5a shows performance of predicting missing numerical attributes in the data, evaluated via holding out 10% of the data. We only consider numerical values (dates) that are more recent than $1000AD$ to focus on more relevant entities. In addition to the neural decoder, we train a search-based decoder as well by considering all 1017 choices in the interval $[1000, 2017]$, and for each triple in the test data, finding the number that the model scores the highest; we use this value to compute the RMSE. As we can see, *all info* outperform other methods on both datasets, demonstrating MKBE is able to utilize different multimodal values for modeling numerical information. Further, the neural decoder performs better than the search-based one, showing the importance of proper decoder, even for finite, enumerable sets. Along the same line, Table 5b shows genre prediction accuracy on 10% of held-

Table 6: **Evaluating Generated Titles** for MovieLens using movies embeddings conditioned on just the ratings (R) and all the information. We present the accuracy of the users in guessing whether the generated title for a movie was real (yes/no), and genre of the movie (4 choices).

Models	Real vs Fake	Genre
R	63	27.2
R+M+U+T+P	73	41.6
Reference	90	68

Table 7: **Evaluating Generated Text and Images** for YAGO using entity embeddings conditioned on just the links (S) or all information. We present the accuracy of the users in guessing whether the generated text/image for a person was real (yes/no), gender of the person, age (<35, or ≥35), and occupation (3 choices).

	Models	Real	Gender	Age	Occup.
descrip.	S	57.1	72.1	59	71.4
	S+N+D+I	59.2	77.2	63.4	78.6
	Reference	67.8	83.2	69.5	90.4
images	S	60	67	53	43
	S+N+D+I	67	77	53	52
	Reference	96	1.0	83	82

out MovieLens dataset. Again, the model that uses all the information outperforms other methods.

MovieLens Titles For generating movie titles, we randomly consider 200 of them as test, 100 as validation, and the remaining ones as training data. The goal here is to generate titles for movies in the test data using the previously mentioned GAN structure. To evaluate our results we conduct a human experiment on Amazon Mechanical Turk (AMT) asking participant two questions: (1) whether they find the movie title real, and (2) which of the four genres is most appropriate for the given title. We consider 30 movies each as reference titles, fake titles generated from only ratings as conditional data, and fake titles conditioned on all the information. Further, each question was asked for 3 participants, and the results computed over the majority choice are shown in Table 6. Fake titles generated with all the information are more similar to reference movie titles, demonstrating that the embeddings that have access to more information effectively generate higher-quality titles.

Table 8: **Generated Descriptions** for "Carles Puyol" (and the corresponding reference from the DBpedia) by embeddings trained from just the links (S) and all of the information (S+N+D+I).

Model	Generated Descriptions
Reference	⟨subject⟩ (born 13 April 1978) is a Spanish retired professional footballer.
Only S	⟨subject⟩ (born 25 January 1949) is a Georgian football coach and former professional player.
S+N+D+I	⟨subject⟩ (born 22 April 1967) is an English former football player.

Table 9: **Generated Images** for YAGO. We consider athletes, and male and female celebrities, and compare their reference images with corresponding ones generated from all the information.

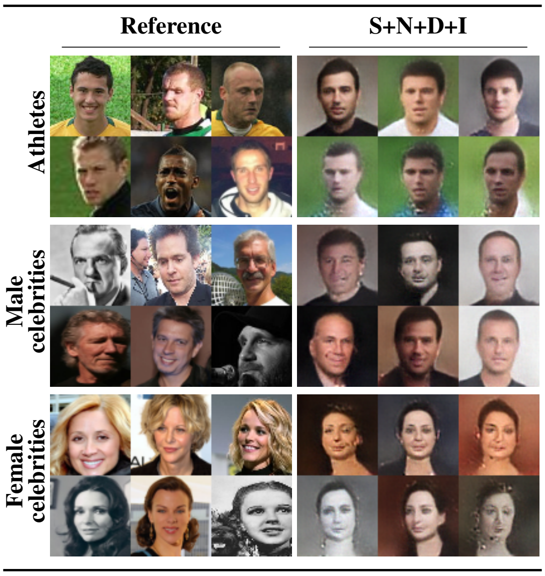

YAGO Descriptions The goal here is to generate descriptive text for entities from their embeddings. Since the original descriptions can be quite long, we consider first sentences that are less than 30 tokens, resulting in 96,405 sentences. We randomly consider 3000 of them as test, 3000 as validation, and the remaining as training data for the decoder. To evaluate the quality of the generated descriptions, and whether they are appropriate for the entity, we conduct a user study asking participants if they can guess the realness of sentences and the occupation (*entertainer*, *sportsman*, or *politician*), gender, and age (above or below 35) of the subject entity from the description. We provide 30 examples for each model asking each question from 3 participants and calculate the accuracy of the majority vote. The results presented in Table 7 show that the models are fairly competent in informing the users of the entity information, and further, descriptions generated from embeddings that had access to more information outperforms the model with only structured data. Examples of generated descriptions are provided in Table 8 (in addition to screenshots of user study, more examples of generated descriptions, and MovieLens titles are provided in supplementary materials).

YAGO Images Here, we evaluate the quality of images generated from entity embeddings by humans ($31,520$, split into train/text). Similar to descriptions, we conduct a study asking users to guess the realness of images and the occupation, gender, and age of the subject. We provide 30 examples for each model asking each question from 3 participants, and use the majority choice.

The results in Table 7 indicate that the images generated with embeddings based on all the information are more accurate for gender and occupation. Guessing age from the images is difficult since the image on DBpedia may not correspond to the age of the person, i.e. some of the older celebrities had photos from their youth. Examples of generated images are shown in Table 9.

6 Discussion and Limitations

An important concern regarding KB embedding approaches is their scalability. While large KBs are a problem for all embedding-based link prediction techniques, MKBE is not significantly worse than existing ones because we treat multimodal information as additional triples. Specifically, although multimodal encoders/decoders are more expensive to train than existing relational models, the cost is still additive as we are effectively increasing the size of the training dataset.

In addition to scalability, there are few other challenges when working with multimodal attributes. Although multimodal evidence provides more information, it is not at all obvious which parts of this additional data are informative for predicting the relational structure of the KB, and the models are prone to overfitting. MKBE builds upon the design of neural encoders and decoders that have been effective for specific modalities, and the results demonstrate that it is able to utilize the information effectively. However, there is still a need to further study models that capture multimodal attributes in a more efficient and accurate manner.

Since our imputing multimodal attributes model is based on GAN structure and the embeddings learned from KB representation, the generated attributes are directly limited by the power of GAN models and the amount of information in the embedding vectors. Although our generated attributes convey several aspects of corresponding entities, their quality is far from ideal due to the size of our datasets (both of our image and text datasets are order of magnitude smaller than common datasets in the existing text/image genration literature) and the amount of information captured by embedding vectors (the knowledge graphs are sparse). In future, we would like to (1) expand multimodal datasets to have more attributes (use many more entities from YAGO), and (2) instead of using learned embeddings to generate missing attributes, utilize the knowledge graph directly for generation.

7 Conclusion

Motivated by the need to utilize multiple sources of information, such as text and images, to achieve more accurate link prediction, we present a novel neural approach to multimodal relational learning. We introduce MKBE, a link prediction model that consists of (1) a compositional encoding component to jointly learn the entity and multimodal embeddings to encode the information available for each entity, and (2) adversarially trained decoding component that use these entity embeddings to impute missing multimodal values. We enrich two existing datasets, YAGO-10 and MovieLens-100k, with multimodal information to introduce benchmarks. We show that MKBE, in comparison to existing link predictors DistMult and ConvE, can achieve higher accuracy on link prediction by utilizing the multimodal evidence. Further, we show that MKBE effectively incorporates relational information to generate high-quality multimodal attributes like images and text. We have release the datasets and the open-source implementation of our models at https://github.com/pouyapez/mkbe.

Acknowledgements

We would like to thank Zhengli Zhao, Robert L. Logan IV, Dheeru Dua, Casey Graff, and the anonymous reviewers for their detailed feedback and suggestions. This work is supported in part by Allen Institute for Artificial Intelligence (AI2) and in part by NSF award #IIS-1817183. The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

References

David Berthelot, Tom Schumm, and Luke Metz. 2017. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence (AAAI)*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. *Conference on Artificial Intelligence (AAAI)*.

Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. 2016. Compact bilinear pooling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 317–326.

Alberto Garcia-Duran and Mathias Niepert. 2017. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Roger Guimerà, Alejandro Llorente, Esteban Moro, and Marta Sales-Pardo. 2012. Predicting human preferences using the block structure of complex social networks. *PloS one*, 7(9):e44620.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long text generation via adversarial training with leaked information. *Conference on Artificial Intelligence (AAAI)*.

F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning (NIPS-16 BDL)*.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. *Advances in Neural Information Processing Systems*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Gil Levi and Tal Hassner. 2015. Age and gender classification using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 34–42.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Conference on Artificial Intelligence (AAAI)*, pages 2181–2187.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM Conference on Recommender Systems*, pages 165–172. ACM.

Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016. Holographic embeddings of knowledge graphs. In *Conference on Artificial Intelligence (AAAI)*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning (ICML)*, pages 809–816.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: scalable machine learning for linked data. In *International Conference on World Wide Web*, pages 271–280. ACM.

Daniel Oñoro-Rubio, Mathias Niepert, Alberto García-Durán, Roberto González-Sánchez, and Roberto J López-Sastre. 2017. Representation learning for visual-relational knowledge graphs. *arXiv preprint arXiv:1709.02314*.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. 2017. Adversarial generation of natural language. *Workshop on Representation Learning for NLP*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, pages 74–84.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *European Semantic Web Conference*.

Hatem Mousselly Sergieh, Teresa Botschen, Iryna Gurevych, and Stefan Roth. 2018. A multimodal translation-based approach for knowledge graph representation learning. In *Joint Conference on Lexical and Computational Semantics*, pages 225–234.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *International Conference on World Wide Web*, pages 697–706. ACM.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*, volume 15, pages 1499–1509.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Association for Computational Linguistics (ACL)*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1722–1731.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Conference on Artificial Intelligence (AAAI)*, pages 1112–1119.

Tobias Weyand, Ilya Kostrikov, and James Philbin. 2016. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer.

Ruobing Xie, Zhiyuan Liu, Tat-seng Chua, Huanbo Luan, and Maosong Sun. 2017. Image-embodied knowledge representation learning. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Conference on Artificial Intelligence (AAAI)*, pages 2659–2665.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *International Conference on Learning Representations (ICLR)*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Conference on Artificial Intelligence (AAAI)*, pages 2852–2858.

Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. 2017. Adversarially regularized autoencoders. *arXiv preprint arXiv:1706.04223*.

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 267–272.

Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction

Yi Luan   Luheng He   Mari Ostendorf   Hannaneh Hajishirzi
University of Washington
{luanyi, luheng, ostendor, hannaneh}@uw.edu

Abstract

We introduce a multi-task setup of identifying and classifying entities, relations, and coreference clusters in scientific articles. We create SciERC, a dataset that includes annotations for all three tasks and develop a unified framework called Scientific Information Extractor (SciIE) for with shared span representations. The multi-task setup reduces cascading errors between tasks and leverages cross-sentence relations through coreference links. Experiments show that our multi-task model outperforms previous models in scientific information extraction without using any domain-specific features. We further show that the framework supports construction of a scientific knowledge graph, which we use to analyze information in scientific literature.[1]

1 Introduction

As scientific communities grow and evolve, new tasks, methods, and datasets are introduced and different methods are compared with each other. Despite advances in search engines, it is still hard to identify new technologies and their relationships with what existed before. To help researchers more quickly identify opportunities for new combinations of tasks, methods and data, it is important to design intelligent algorithms that can extract and organize scientific information from a large collection of documents.

Organizing scientific information into structured knowledge bases requires information extraction (IE) about scientific entities and their relationships. However, the challenges associated with scientific IE are greater than for a general domain. First, annotation of scientific text requires domain expertise which makes annotation costly and limits resources.

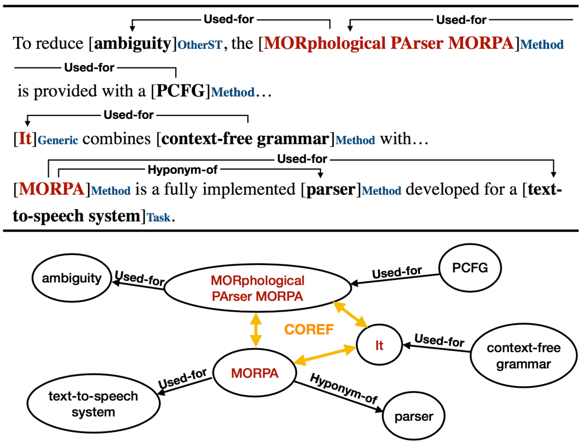

To reduce **[ambiguity]**OtherST, the **[MORphological PArser MORPA]**Method is provided with a **[PCFG]**Method…

[It]Generic combines **[context-free grammar]**Method with…

[MORPA]Method is a fully implemented **[parser]**Method developed for a **[text-to-speech system]**Task.

Figure 1: Example annotation: phrases that refer to the same scientific concept are annotated into the same coreference cluster, such as *MORphological PAser MORPA*, *it* and *MORPA* (marked as red).

In addition, most relation extraction systems are designed for within-sentence relations. However, extracting information from scientific articles requires extracting relations across sentences. Figure 1 illustrates this problem. The cross-sentence relations between some entities can only be connected by entities that refer to the same scientific concept, including generic terms (such as the pronoun *it*, or phrases like *our method*) that are not informative by themselves. With co-reference, *context-free grammar* can be connected to *MORPA* through the intermediate co-referred pronoun *it*. Applying existing IE systems to this data, without co-reference, will result in much lower relation coverage (and a sparse knowledge base).

In this paper, we develop a unified learning model for extracting scientific entities, relations, and coreference resolution. This is different from previous work (Luan et al., 2017b; Gupta and Manning, 2011; Tsai et al., 2013; Gábor et al., 2018) which often addresses these tasks as independent

---

[1] Data and code are publicly available at: `http://nlp.cs.washington.edu/sciIE/`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3219–3232
Brussels, Belgium, October 31 - November 4, 2018.

components of a pipeline. Our unified model is a multi-task setup that shares parameters across low-level tasks, making predictions by leveraging context across the document through coreference links. Specifically, we extend prior work for learning span representations and coreference resolution (Lee et al., 2017; He et al., 2018). Different from a standard tagging system, our system enumerates all possible spans during decoding and can effectively detect overlapped spans. It avoids cascading errors between tasks by jointly modeling all spans and span-span relations.

To explore this problem, we create a dataset SCIERC for scientific information extraction, which includes annotations of scientific terms, relation categories and co-reference links. Our experiments show that the unified model is better at predicting span boundaries, and it outperforms previous state-of-the-art scientific IE systems on entity and relation extraction (Luan et al., 2017b; Augenstein et al., 2017). In addition, we build a scientific knowledge graph integrating terms and relations extracted from each article. Human evaluation shows that propagating coreference can significantly improve the quality of the automatic constructed knowledge graph.

In summary we make the following contributions. We create a dataset for scientific information extraction by jointly annotating scientific entities, relations, and coreference links. Extending a previous end-to-end coreference resolution system, we develop a multi-task learning framework that can detect scientific entities, relations, and coreference clusters without hand-engineered features. We use our unified framework to build a scientific knowledge graph from a large collection of documents and analyze information in scientific literature.

2 Related Work

There has been growing interest in research on automatic methods for information extraction from scientific articles. Past research in scientific IE addressed analyzing citations (Athar and Teufel, 2012b,a; Kas, 2011; Gabor et al., 2016; Sim et al., 2012; Do et al., 2013; Jaidka et al., 2014; Abu-Jbara and Radev, 2011), analyzing research community (Vogel and Jurafsky, 2012; Anderson et al., 2012), and unsupervised methods for extracting scientific entities and relations (Gupta and Manning, 2011; Tsai et al., 2013; Gábor et al., 2016).

More recently, two datasets in SemEval 2017 and 2018 have been introduced, which facilitate research on supervised and semi-supervised learning for scientific information extraction. SemEval 17 (Augenstein et al., 2017) includes 500 paragraphs from articles in the domains of computer science, physics, and material science. It includes three types of entities (called keyphrases): Tasks, Methods, and Materials and two relation types: hyponym-of and synonym-of. SemEval 18 (Gábor et al., 2018) is focused on predicting relations between entities within a sentence. It consists of six relation types. Using these datasets, neural models (Ammar et al., 2017, 2018; Luan et al., 2017b; Augenstein and Søgaard, 2017) are introduced for extracting scientific information. We extend these datasets by increasing relation coverage, adding cross-sentence coreference linking, and removing some annotation constraints. Different from most previous IE systems for scientific literature and general domains (Miwa and Bansal, 2016; Xu et al., 2016; Peng et al., 2017; Quirk and Poon, 2017; Luan et al., 2018; Adel and Schütze, 2017), which use preprocessed syntactic, discourse or coreference features as input, our unified framework does not rely on any pipeline processing and is able to model overlapping spans.

While Singh et al. (2013) show improvements by jointly modeling entities, relations, and coreference links, most recent neural models for these tasks focus on single tasks (Clark and Manning, 2016; Wiseman et al., 2016; Lee et al., 2017; Lample et al., 2016; Peng et al., 2017) or joint entity and relation extraction (Katiyar and Cardie, 2017; Zhang et al., 2017; Adel and Schütze, 2017; Zheng et al., 2017). Among those studies, many papers assume the entity boundaries are given, such as (Clark and Manning, 2016), Adel and Schütze (2017) and Peng et al. (2017). Our work relaxes this constraint and predicts entity boundaries by optimizing over all possible spans. Our model draws from recent end-to-end span-based models for coreference resolution (Lee et al., 2017, 2018) and semantic role labeling (He et al., 2018) and extends them for the multi-task framework involving the three tasks of identification of entity, relation and coreference.

Neural multi-task learning has been applied to a range of NLP tasks. Most of these models share word-level representations (Collobert and Weston, 2008; Klerke et al., 2016; Luan et al., 2016, 2017a; Rei, 2017), while Peng et al. (2017) uses high-order cross-task factors. Our model instead propagates cross-task information via span representations, which is related to Swayamdipta et al. (2017).

3 Dataset

Our dataset (called SciERC) includes annotations for scientific entities, their relations, and coreference clusters for 500 scientific abstracts. These abstracts are taken from 12 AI conference/workshop proceedings in four AI communities from the Semantic Scholar Corpus[2]. SciERC extends previous datasets in scientific articles SemEval 2017 Task 10 (SemEval 17) (Augenstein et al., 2017) and SemEval 2018 Task 7 (SemEval 18) (Gábor et al., 2018) by extending entity types, relation types, relation coverage, and adding cross-sentence relations using coreference links. Our dataset is publicly available at: `http://nlp.cs.washington.edu/sciIE/`. Table 1 shows the statistics of SciERC.

Annotation Scheme We define six types for annotating scientific entities (Task, Method, Metric, Material, Other-ScientificTerm and Generic) and seven relation types (Compare, Part-of, Conjunction, Evaluate-for, Feature-of, Used-for, Hyponym-Of). Directionality is taken into account except for the two symmetric relation types (Conjunction and Compare). Coreference links are annotated between identical scientific entities. A Generic entity is annotated only when the entity is involved in a relation or is coreferred with another entity. Annotation guidelines can be found in Appendix A. Figure 1 shows an annotated example.

Following annotation guidelines from QasemiZadeh and Schumann (2016) and using the BRAT interface (Stenetorp et al., 2012), our annotators perform a greedy annotation for spans and always prefer the longer span whenever ambiguity occurs. Nested spans are allowed when a subspan has a relation/coreference link with another term outside the span.

Human Agreements One domain expert annotated all the documents in the dataset; 12% of the data is dually annotated by 4 other domain experts to evaluate the user agreements. The kappa score for annotating entities is 76.9%, relation extraction is 67.8% and coreference is 63.8%.

Statistics	SciERC	SemEval 17	SemEval 18
#Entities	8089	9946	7483
#Relations	4716	672	1595
#Relations/Doc	9.4	1.3	3.2
#Coref links	2752	-	-
#Coref clusters	1023	-	-

Table 1: Dataset statistics for our dataset SciERC and two previous datasets on scientific information extraction. All datasets annotate 500 documents.

Comparison with previous datasets SciERC is focused on annotating cross-sentence relations and has more relation coverage than SemEval 17 and SemEval 18, as shown in Table 1. SemEval 17 is mostly designed for entity recognition and only covers two relation types. The task in SemEval 18 is to classify a relation between a pair of entities given entity boundaries, but only intra-sentence relations are annotated and each entity only appears in one relation, resulting in sparser relation coverage than our dataset (3.2 vs. 9.4 relations per abstract). SciERC extends these datasets by adding more relation types and coreference clusters, which allows representing cross-sentence relations, and removing annotation constraints. Table 1 gives a comparison of statistics among the three datasets. In addition, SciERC aims at including broader coverage of general AI communities.

4 Model

We develop a unified framework (called SciIE) to identify and classify scientific entities, relations, and coreference resolution across sentences. SciIE is a multi-task learning setup that extends previous span-based models for coreference resolution (Lee et al., 2017) and semantic role labeling (He et al., 2018). All three tasks of entity recognition, relation extraction, and coreference resolution are treated as multinomial classification problems with shared span representations. SciIE benefits from expressive contextualized span representations as classifier features. By sharing span representations, sentence-level tasks can benefit from information propagated from coreference resolution across sentences, without increasing the complexity of inference. Figure 2 shows a high-level overview of the SciIE multi-task framework.

4.1 Problem Definition

The input is a document represented as a sequence of words $D = \{w_1, \dots, w_n\}$, from which we derive $S = \{s_1, \dots, s_N\}$, the set of all possible

---

[2] These conferences include general AI (AAAI, IJCAI), NLP (ACL, EMNLP, IJCNLP), speech (ICASSP, Interspeech), machine learning (NIPS, ICML), and computer vision (CVPR, ICCV, ECCV) at `http://labs.semanticscholar.org/corpus/`

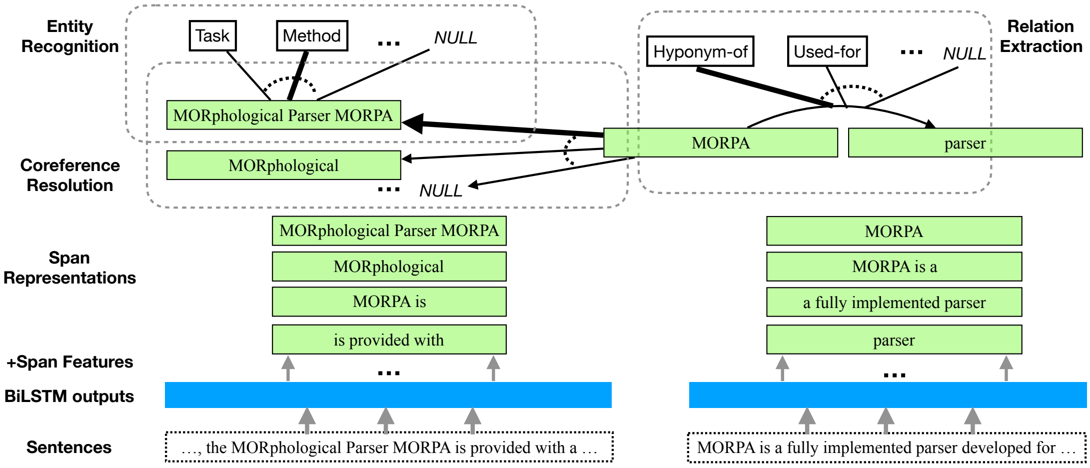

Figure 2: Overview of the multitask setup, where all three tasks are treated as classification problems on top of shared span representations. Dotted arcs indicate the normalization space for each task.

within-sentence word sequence spans (up to a reasonable length) in the document. The output contains three structures: the entity types E for all spans S, the relations R for all pair of spans $S \times S$, and the coreference links C for all spans in S. The output structures are represented with a set of discrete random variables indexed by spans or pairs of spans. Specifically, the output structures are defined as follows.

Entity recognition is to predict the best entity type for every candidate span. Let L_{E} represent the set of all possible entity types including the null-type ϵ. The output structure E is a set of random variables indexed by spans: $e_i \in L_{\mathrm{E}}$ for $i = 1, \ldots, N$.

Relation extraction is to predict the best relation type given an ordered pair of spans (s_i, s_j). Let L_{R} be the set of all possible relation types including the null-type ϵ. The output structure R is a set of random variables indexed over pairs of spans (i, j) that belong to the same sentence: $r_{ij} \in L_{\mathrm{R}}$ for $i, j = 1, \ldots, N$.

Coreference resolution is to predict the best antecedent (including a special null antecedent) given a span, which is the same mention-ranking model used in Lee et al. (2017). The output structure C is a set of random variables defined as: $c_i \in \{1, \ldots, i-1, \epsilon\}$ for $i = 1, \ldots, N$.

4.2 Model Definition

We formulate the multi-task learning setup as learning the conditional probability distribution $P(E, R, C|D)$. For efficient training and inference, we decompose $P(E, R, C|D)$ assuming spans are conditionally independent given D:

$$P(E, R, C \mid D) = P(E, R, C, S \mid D) \qquad (1)$$
$$= \prod_{i=1}^{N} P(e_i \mid D) P(c_i \mid D) \prod_{j=1}^{N} P(r_{ij} \mid D),$$

where the conditional probabilities of each random variable are independently normalized:

$$P(e_i = e \mid D) = \frac{\exp(\Phi_{\mathrm{E}}(e, s_i))}{\sum_{e' \in L_{\mathrm{E}}} \exp(\Phi_{\mathrm{E}}(e', s_i))} \qquad (2)$$
$$P(r_{ij} = r \mid D) = \frac{\exp(\Phi_{\mathrm{R}}(r, s_i, s_j))}{\sum_{r' \in L_{\mathrm{R}}} \exp(\Phi_{\mathrm{R}}(r', s_i, s_j))}$$
$$P(c_i = j \mid D) = \frac{\exp(\Phi_{\mathrm{C}}(s_i, s_j))}{\sum_{j' \in \{1, \ldots, i-1, \epsilon\}} \exp(\Phi_{\mathrm{C}}(s_i, s_{j'}))},$$

where Φ_{E} denotes the unnormalized model score for an entity type e and a span s_i, Φ_{R} denotes the score for a relation type r and span pairs s_i, s_j, and Φ_{C} denotes the score for a binary coreference link between s_i and s_j. These Φ scores are further decomposed into span and pairwise span scores computed from feed-forward networks, as will be explained in Section 4.3.

For simplicity, we omit D from the Φ functions and S from the observation.

Objective Given a set of all documents $\mathcal{D}$, the model loss function is defined as a weighted sum of the negative log-likelihood loss of all three tasks:

$$-\sum_{(D, R^*, E^*, C^*) \in \mathcal{D}} \Big\{ \lambda_{\mathrm{E}} \log P(E^* \mid D) \qquad (3)$$
$$+ \lambda_{\mathrm{R}} \log P(R^* \mid D) + \lambda_{\mathrm{C}} \log P(C^* \mid D) \Big\}$$

where E^*, R^*, and C^* are gold structures of the entity types, relations, and coreference, respectively. The task weights λ_{E}, λ_{R}, and λ_{C} are introduced as hyper-parameters to control the importance of each task.

For entity recognition and relation extraction, $P(E^* \mid D)$ and $P(R^* \mid D)$ are computed with the definition in Equation (2). For coreference resolution, we use the marginalized loss following Lee et al. (2017) since each mention can have multiple correct antecedents. Let C_i^* be the set of all correct antecedents for span i, we have: $\log P(C^* \mid D) = \sum_{i=1..N} \log \sum_{c \in C_i^*} P(c \mid D)$.

4.3 Scoring Architecture

We use feedforward neural networks (FFNNs) over *shared span representations* $\mathbf{g}$ to compute a set of span and pairwise span scores. For the span scores, $\phi_e(s_i)$ measures how likely a span s_i has an entity type e, and $\phi_{\text{mr}}(s_i)$ and $\phi_{\text{mc}}(s_i)$ measure how likely a span s_i is a mention in a relation or a coreference link, respectively. The pairwise scores $\phi_r(s_i, s_j)$ and $\phi_{\text{c}}(s_i, s_j)$ measure how likely two spans are associated in a relation r or a coreference link, respectively. Let $\mathbf{g}_i$ be the fixed-length vector representation for span s_i. For different tasks, the span scores $\phi_{\text{x}}(s_i)$ for x $\in \{e, \text{mc}, \text{mr}\}$ and pairwise span scores $\phi_{\text{y}}(s_i, s_j)$ for y $\in \{r, \text{c}\}$ are computed as follows:

$$\begin{aligned} \phi_{\text{x}}(s_i) &= \mathbf{w}_{\text{x}} \cdot \text{FFNN}_{\text{x}}(\mathbf{g}_i) \\ \phi_{\text{y}}(s_i, s_j) &= \mathbf{w}_{\text{y}} \cdot \text{FFNN}_{\text{y}}([\mathbf{g}_i, \mathbf{g}_j, \mathbf{g}_i \odot \mathbf{g}_j]), \end{aligned}$$

where $\odot$ is element-wise multiplication, and $\{\mathbf{w}_{\text{x}}, \mathbf{w}_{\text{y}}\}$ are neural network parameters to be learned.

We use these scores to compute the different Φ:

$$\begin{aligned} \Phi_{\text{E}}(e, s_i) &= \phi_e(s_i) \qquad (4) \\ \Phi_{\text{R}}(r, s_i, s_j) &= \phi_{\text{mr}}(s_i) + \phi_{\text{mr}}(s_j) + \phi_r(s_i, s_j) \\ \Phi_{\text{C}}(s_i, s_j) &= \phi_{\text{mc}}(s_i) + \phi_{\text{mc}}(s_j) + \phi_{\text{c}}(s_i, s_j) \end{aligned}$$

The scores in Equation (4) are defined for entity types, relations, and antecedents that are not the null-type ϵ. Scores involving the null label are set to a constant 0: $\Phi_{\text{E}}(\epsilon, s_i) = \Phi_{\text{R}}(\epsilon, s_i, s_j) = \Phi_{\text{C}}(s_i, \epsilon) = 0$.

We use the same span representations $\mathbf{g}$ from (Lee et al., 2017) and share them across the three tasks. We start by building bi-directional LSTMs (Hochreiter and Schmidhuber, 1997) from word, character and ELMo (Peters et al., 2018) embeddings.

For a span s_i, its vector representation $\mathbf{g}_i$ is constructed by concatenating s_i's left and right end points from the BiLSTM outputs, an attention-based soft "headword," and embedded span width features. Hyperparameters and other implementation details will be described in Section 6.

4.4 Inference and Pruning

Following previous work, we use beam pruning to reduce the number of pairwise span factors from $O(n^4)$ to $O(n^2)$ at both training and test time, where n is the number of words in the document. We define two separate beams: B_{C} to prune spans for the coreference resolution task, and B_{R} for relation extraction. The spans in the beams are sorted by their span scores ϕ_{mc} and ϕ_{mr} respectively, and the sizes of the beams are limited by $\lambda_{\text{C}} n$ and $\lambda_{\text{R}} n$. We also limit the maximum width of spans to a fixed number W, which further reduces the number of span factors to $O(n)$.

5 Knowledge Graph Construction

We construct a scientific knowledge graph from a large corpus of scientific articles. The corpus includes all abstracts (110k in total) from 12 AI conference proceedings from the Semantic Scholar Corpus. Nodes in the knowledge graph correspond to scientific entities. Edges correspond to scientific relations between pairs of entities. The edges are typed according to the relation types defined in Section 3. Figure 4 shows a part of a knowledge graph created by our method. For example, *Statistical Machine Translation (SMT)* and *grammatical error correction* are nodes in the graph, and they are connected through a *Used-for* relation type. In order to construct the knowledge graph for the whole corpus, we first apply the SciIE model over single documents and then integrate the entities and relations across multiple documents (Figure 3).

Extracting nodes (entities) The SciIE model extracts entities, their relations, and coreference

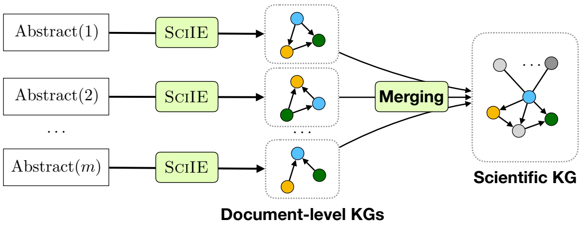

Figure 3: Knowledge graph construction process.

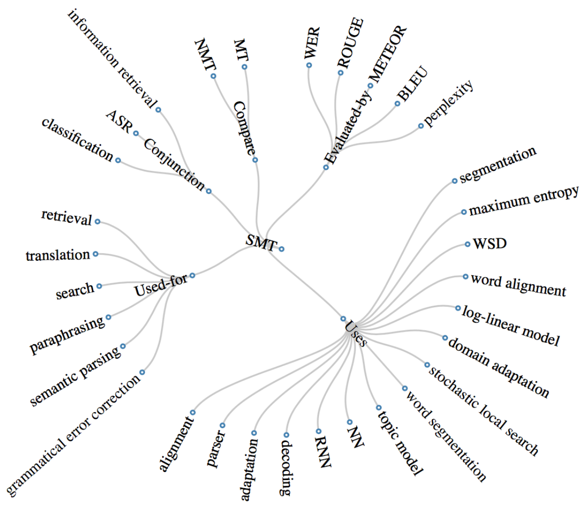

Figure 4: A part of an automatically constructed scientific knowledge graph with the most frequent neighbors of the scientific term *statistical machine translation (SMT)* on the graph. For simplicity we denote *Used-for (Reverse)* as *Uses*, *Evaluated-for (Reverse)* as *Evaluated-by*, and replace common terms with their acronyms. The original graph and more examples are given Figure 10 in Appendix B.

clusters within one document. Phrases are heuristically normalized (described in Section 6) using entities and coreference links. In particular, we link all entities that belong to the same coreference cluster to replace generic terms with any other non-generic term in the cluster. Moreover, we replace all the entities in the cluster with the entity that has the longest string. Our qualitative analysis shows that there are fewer ambiguous phrases using coreference links (Figure 5). We calculate the frequency counts of all entities that appear in the whole corpus. We assign nodes in the knowledge graph by selecting the most frequent entities (with counts $> k$) in the corpus, and merge in any remaining entities for which a frequent entity is a substring.

Assigning edges (relations) A pair of entities may appear in different contexts, resulting in different relation types between those entities (Figure 6). For every pair of entities in the graph, we calculate the frequency of different relation types across the whole corpus.We assign edges between entities by selecting the most frequent relation type.

6 Experimental Setup

We evaluate our unified framework SciIE on SciERC and SemEval 17. The knowledge graph for scientific community analysis is built using the Semantic Scholar Corpus (110k abstracts in total).

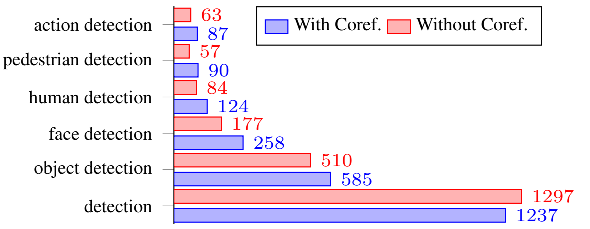

Figure 5: Frequency of detected entities with and without coreferece resolution: using coreference reduces the frequency of the generic phrase *detection* while significantly increasing the frequency of specific phrases. Linking entities through coreference helps disambiguate phrases when generating the knowledge graph.

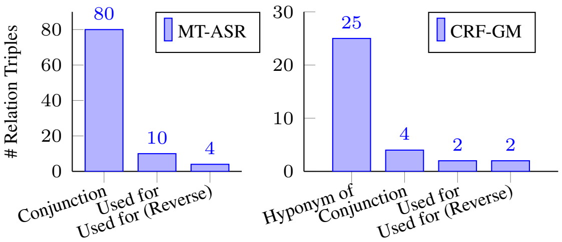

Figure 6: Frequency of relation types between pairs of entities: (*left*) automatic speech recognition (ASR) and machine translation (MT), (*right*) conditional random field (CRF) and graphical model (GM). We use the most frequent relation between pairs of entities in the knowledge graph.

6.1 Baselines

We compare our model with the following baselines on SciERCdataset:

- **LSTM+CRF** The state-of-the-art NER system (Lample et al., 2016), which applies CRF on top of LSTM for named entity tagging, the approach has also been used in scientific term extraction (Luan et al., 2017b).
- **LSTM+CRF+ELMo** LSTM+CRF with ELMo as an additional input feature.
- **E2E Rel** State-of-the-art joint entity and relation extraction system (Miwa and Bansal, 2016) that has also been used in scientific literature (Peters et al., 2017; Augenstein et al., 2017). This system uses syntactic features such as part-of-speech tagging and dependency parsing.

- **E2E Rel(Pipeline)** Pipeline setting of E2E Rel. Extract entities first and use entity results as input to relation extraction task.

- **E2E Rel+ELMo** E2E Rel with ELMo as an additional input feature.

- **E2E Coref** State-of-the-art coreference system Lee et al. (2017) combined with ELMo. Our system SciIE extends E2E Coref with multi-task learning.

In the SemEval task, we compare our model SciIE with the best reported system in the SemEval leaderboard (Peters et al., 2017), which extends E2E Rel with several in-domain features such as gazetteers extracted from existing knowledge bases and model ensembles. We also compare with the state of the art on keyphrase extraction (Luan et al., 2017b), which applies semi-supervised methods to a neural tagging model.[3]

6.2 Implementation details

Our system extends the implementation and hyper-parameters from Lee et al. (2017) with the following adjustments. We use a 1 layer BiLSTM with 200-dimensional hidden layers. All the FFNNs have 2 hidden layers of 150 dimensions each. We use 0.4 variational dropout (Gal and Ghahramani, 2016) for the LSTMs, 0.4 dropout for the FFNNs, and 0.5 dropout for the input embeddings. We model spans up to 8 words. For beam pruning, we use $\lambda_{\text{C}} = 0.3$ for coreference resolution and $\lambda_{\text{R}} = 0.4$ for relation extraction. For constructing the knowledge graph, we use the following heuristics to normalize the entity phrases. We replace all acronyms with their corresponding full name and normalize all the plural terms with their singular counterparts.

7 Experimental Results

We evaluate SciIE on SciERC and SemEval 17 datasets. We provide qualitative results and human evaluation of the constructed knowledge graph.

7.1 IE Results

	Dev			Test		
Model	P	R	F1	P	R	F1
LSTM+CRF	67.2	65.8	66.5	62.9	61.1	62.0
LSTM+CRF+ELMo	68.1	66.3	67.2	63.8	63.2	63.5
E2E Rel(Pipeline)	66.7	65.9	66.3	60.8	61.2	61.0
E2E Rel	64.3	68.6	66.4	60.6	61.9	61.2
E2E Rel+ELMo	67.5	66.3	66.9	63.5	63.9	63.7
SciIE	70.0	66.3	**68.1**	67.2	61.5	**64.2**

(a) Entity recognition.

	Dev			Test		
Model	P	R	F1	P	R	F1
E2E Rel(Pipeline)	34.2	33.7	33.9	37.8	34.2	35.9
E2E Rel	37.3	33.5	35.3	37.1	32.2	34.1
E2E Rel+ELMo	38.5	36.4	37.4	38.4	34.9	36.6
SciIE	45.4	34.9	**39.5**	47.6	33.5	**39.3**

(b) Relation extraction.

	Dev			Test		
Model	P	R	F1	P	R	F1
E2E Coref	59.4	52.0	55.4	60.9	37.3	46.2
SciIE	61.5	54.8	**58.0**	52.0	44.9	**48.2**

(c) Coreference resolution.

Table 2: Comparison with previous systems on the development and test set for our three tasks. For coreference resolution, we report the average P/R/F1 of MUC, B^3, and CEAF_{ϕ_4} scores.

Results on SciERC Table 2 compares the result of our model with baselines on the three tasks: entity recognition (Table 2a), relation extraction (Table 2b), and coreference resolution (Table 2c). As evidenced by the table, our unified multi-task setup SciIE outperforms all the baselines. For entity recognition, our model achieves 1.3% and 2.4% relative improvement over LSTM+CRF with and without ELMo, respectively. Moreover, it achieves 1.8% and 2.7% relative improvement over E2E Rel with and without ELMo, respectively. For relation extraction, we observe more significant improvement with 13.1% relative improvement over E2E Rel and 7.4% improvement over E2E Rel with ELMo. For coreference resolution, SciIE outperforms E2E Coref with 4.5% relative improvement. We still observe a large gap between human-level performance and a machine learning system. We invite the community to address this challenging task.

Ablations We evaluate the effect of multi-task learning in each of the three tasks defined in our dataset. Table 3 reports the results for individual tasks when additional tasks are included in the learning objective function. We observe that performance improves with each added task in the objective. For example, Entity recognition (65.7) benefits from both coreference resolution (67.5) and relation extraction (66.8). Relation extrac-

---

[3] We compare with the inductive setting results.

Task	Entity Rec.	Relation	Coref.
Multi Task (SciIE)	68.1	39.5	58.0
Single Task	65.7	37.9	55.3
+Entity Rec.	-	38.9	57.1
+Relation	66.8	-	57.6
+Coreference	67.5	39.5	-

Table 3: Ablation study for multitask learning on SciERC development set. Each column shows results for the target task.

tion (37.9) significantly benefits when multi-tasked with coreference resolution (7.1% relative improvement). Coreference resolution benefits when multi-tasked with relation extraction, with 4.9% relative improvement.

Results on SemEval 17 Table 4 compares the results of our model with the state of the art on the SemEval 17 dataset for tasks of span identification, keyphrase extraction and relation extraction as well as the overall score. Span identification aims at identifying spans of entities. Keyphrase classification and relation extraction has the same setting with the entity and relation extraction in SciERC. Our model outperforms all the previous models that use hand-designed features. We observe more significant improvement in span identification than keyphrase classification. This confirms the benefit of our model in enumerating spans (rather than BIO tagging in state-of-the-art systems). Moreover, we have competitive results compared to the previous state of the art in relation extraction. We observe less gain compared to the SciERC dataset mainly because there are no coference links, and the relation types are not comprehensive.

7.2 Knowledge Graph Analysis

We provide qualitative analysis and human evaluations on the constructed knowledge graph.

Scientific trend analysis Figure 7 shows the historical trend analysis (from 1996 to 2016) of the most popular applications of the phrase *neural network*, selected according to the statistics of the extracted relation triples with the 'Used-for' relation type from speech, computer vision, and NLP conference papers. We observe that, before 2000, *neural network* has been applied to a greater percentage of speech applications compared to the NLP and computer vision papers. In NLP, neural networks first gain popularity in language modeling and then extend to other tasks such as POS Tagging and Machine Translation. In computer vision, the application of neural networks gains popularity in *object recognition* earlier (around 2010) than the other two more complex tasks of *object detection* and *image segmentation* (hardest and also the latest).

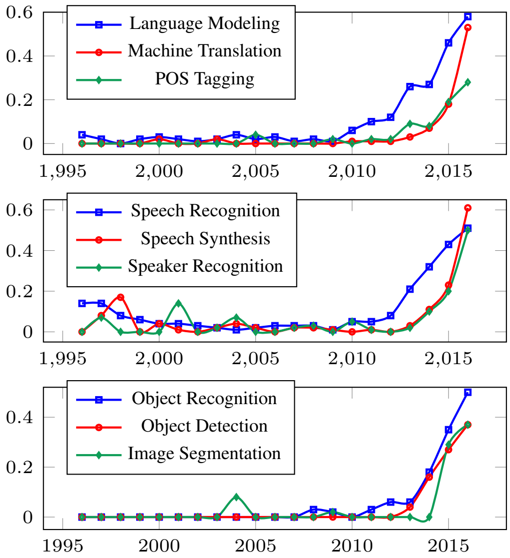

Figure 7: Historical trend for top applications of the keyphrase *neural network* in NLP, speech, and CV conference papers we collected. y-axis indicates the ratio of papers that use *neural network* in the task to the number of papers that is about the task.

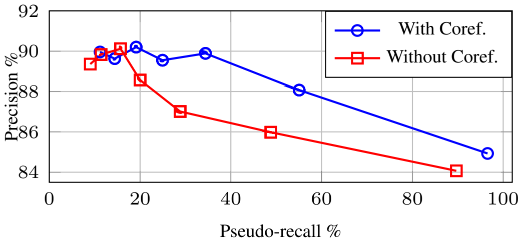

Figure 8: Precision/pseudo-recall curves for human evaluation by varying cut-off thresholds. The AUC is 0.751 with coreference, and 0.695 without.

Knowledge Graph Evaluation Figure 8 shows the human evaluation of the constructed knowledge graph, comparing the quality of automatically generated knowledge graphs with and without the coreference links. We randomly select 10 frequent scientific entities and extract all the relation triples that include one of the selected entities leading to 1.5k relation triples from both systems. We ask four domain experts to annotate each of these ex-

	Span Indentification			Keyphrase Extraction			Relation Extraction			Overall		
Model	P	R	F1	P	R	F1	P	R	F1	P	R	F1
(Luan 2017)	-	-	56.9	-	-	45.3	-	-	-	-	-	-
Best SemEval	55	54	55	44	43	44	36	23	**28**	44	41	43
SciIE	62.2	55.4	**58.6**	48.5	43.8	**46.0**	40.4	21.2	27.8	48.1	41.8	**44.7**

Table 4: Results for scientific keyphrase extraction and extraction on SemEval 2017 Task 10, comparing with previous best systems.

tracted relations to define ground truth labels. Each domain expert is assigned 2 or 3 entities and all of the corresponding relations. Figure 8 shows precision/recall curves for both systems. Since it is not feasible to compute the actual recall of the systems, we compute the pseudo-recall (Zhang et al., 2015) based on the output of both systems. We observe that the knowledge graph curve with coreference linking is mostly above the curve without coreference linking. The precision of both systems is high (above 84% for both systems), but the system with coreference links has significantly higher recall.

8 Conclusion

In this paper, we create a new dataset and develop a multi-task model for identifying entities, relations, and coreference clusters in scientific articles. By sharing span representations and leveraging cross-sentence information, our multi-task setup effectively improves performance across all tasks. Moreover, we show that our multi-task model is better at predicting span boundaries and outperforms previous state-of-the-art scientific IE systems on entity and relation extraction, without using any hand-engineered features or pipeline processing. Using our model, we are able to automatically organize the extracted information from a large collection of scientific articles into a knowledge graph. Our analysis shows the importance of coreference links in making a dense, useful graph.

We still observe a large gap between the performance of our model and human performance, confirming the challenges of scientific IE. Future work includes improving the performance using semi-supervised techniques and providing in-domain features. We also plan to extend our multi-task framework to information extraction tasks in other domains.

Acknowledgments

This research was supported by the Office of Naval Research under the MURI grant N00014-18-1-2670, NSF (IIS 1616112, III 1703166), Allen Distinguished Investigator Award, and gifts from Allen Institute for AI, Google, Amazon, and Bloomberg. We are grateful to Waleed Ammar and AI2 for sharing the Semantic Scholar Corpus. We also thank the anonymous reviewers, UW-NLP group and Shoou-I Yu for their helpful comments.

References

Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based summarization of scientific papers. In *Proc. Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. volume 1, pages 500–509.

Heike Adel and Hinrich Schütze. 2017. Global normalization of convolutional neural networks for joint entity and relation classification. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*. pages 1723–1729.

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, et al. 2018. Construction of the literature graph in semantic scholar. In *Proc. Conf. North American Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT), (Industry Papers)*. pages 84–91.

Waleed Ammar, Matthew Peters, Chandra Bhagavatula, and Russell Power. 2017. The ai2 system at semeval-2017 task 10 (scienceie): semi-supervised end-to-end entity and relation extraction. In *Proc. Int. Workshop on Semantic Evaluation (SemEval)*. pages 592–596.

Ashton Anderson, Dan McFarland, and Dan Jurafsky. 2012. Towards a computational history of the ACL: 1980-2008. In *Proc. ACL Special Workshop on Rediscovering 50 Years of Discoveries*. pages 13–21.

Awais Athar and Simone Teufel. 2012a. Context-enhanced citation sentiment detection. In *Proc. Conf. North American Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. pages 597–601.

Awais Athar and Simone Teufel. 2012b. Detection of implicit citations for sentiment detection. In *Proc.*

ACL Workshop on Detecting Structure in Scholarly Discourse. pages 18–26.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proc. Int. Workshop on Semantic Evaluation (SemEval)*.

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary classification. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*. pages 341–346.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. *CoRR* abs/1606.01323.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. Int. Conf. Machine Learning (ICML)*. pages 160–167.

Huy Hoang Nhat Do, Muthu Kumar Chandrasekaran, Philip S Cho, and Min Yen Kan. 2013. Extracting and matching authors and affiliations in scholarly documents. In *Proc. ACM/IEEE-CS Joint Conference on Digital libraries*. pages 219–228.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, and Thierry Charnois. 2018. Semeval-2018 Task 7: Semantic relation extraction and classification in scientific papers. In *Proc. Int. Workshop on Semantic Evaluation (SemEval)*.

Kata Gabor, Haifa Zargayouna, Davide Buscaldi, Isabelle Tellier, and Thierry Charnois. 2016. Semantic annotation of the ACL anthology corpus for the automatic analysis of scientific literature. In *Proc. Language Resources and Evaluation Conference (LREC)*.

Kata Gábor, Haïfa Zargayouna, Isabelle Tellier, Davide Buscaldi, and Thierry Charnois. 2016. Unsupervised relation extraction in specialized corpora using sequence mining. In *International Symposium on Intelligent Data Analysis*. Springer, pages 237–248.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*.

Sonal Gupta and Christopher D Manning. 2011. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *Proc. IJCNLP*. pages 1–9.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *ACL*.

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

Kokil Jaidka, Muthu Kumar Chandrasekaran, Beatriz Fisas Elizalde, Rahul Jha, Christopher Jones, Min-Yen Kan, Ankur Khanna, Diego Molla-Aliod, Dragomir R Radev, Francesco Ronzano, et al. 2014. The computational linguistics summarization pilot task. In *Proc. Text Analysis Conference*.

Miray Kas. 2011. Structures and statistics of citation networks. Technical report, DTIC Document.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*. volume 1, pages 917–928.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *HLT-NAACL*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. Conf. North American Assoc. for Computational Linguistics (NAACL)*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *NAACL*.

Yi Luan, Chris Brockett, Bill Dolan, Jianfeng Gao, and Michel Galley. 2017a. Multi-task learning for speaker-role adaptation in neural conversation models. In *Proc. IJCNLP*.

Yi Luan, Yangfeng Ji, Hannaneh Hajishirzi, and Boyang Li. 2016. Multiplicative representations for unsupervised semantic role induction. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*. page 118.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017b. Scientific information extraction with semi-supervised neural tagging. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. The uwnlp system at semeval-2018 task 7: Neural relation extraction model with selectively incorporated concept embeddings. In *Proc. Int. Workshop on Semantic Evaluation (SemEval)*. pages 788–792.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*. pages 1105–1116.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Trans. Assoc. for Computational Linguistics (TACL)* 5:101–115.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*. volume 1, pages 1756–1765.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Behrang QasemiZadeh and Anne-Kathrin Schumann. 2016. The ACL RD-TEC 2.0: A language resource for evaluating term extraction and entity recognition methods. In *LREC*.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proc. European Chapter Assoc. for Computational Linguistics (EACL)*. pages 1171–1182.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*.

Yanchuan Sim, Noah A Smith, and David A Smith. 2012. Discovering factions in the computational linguistics community. In *Proc. ACL Special Workshop on Rediscovering 50 Years of Discoveries*. pages 22–32.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proc. of the 2013 workshop on Automated knowledge base construction*. ACM, pages 1–6.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proc. European Chapter Assoc. for Computational Linguistics (EACL)*. pages 102–107.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. *CoRR* abs/1706.09528.

Chen-Tse Tsai, Gourab Kundu, and Dan Roth. 2013. Concept-based analysis of scientific literature. In *Proc. ACM Int. Conference on Information & Knowledge Management*. ACM, pages 1733–1738.

Adam Vogel and Dan Jurafsky. 2012. He said, she said: Gender in the ACL anthology. In *Proc. ACL Special Workshop on Rediscovering 50 Years of Discoveries*. pages 33–41.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *HLT-NAACL*.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. In *Proc. Int. Conf. Computational Linguistics (COLING)*. pages 1461–1470.

Congle Zhang, Stephen Soderland, and Daniel S. Weld. 2015. Exploiting parallel news streams for unsupervised event extraction. *TACL* 3:117–129.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*. pages 1730–1740.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*. volume 1, pages 1227–1236.

A Annotation Guideline

A.1 Entity Category

- **Task**: Applications, problems to solve, systems to construct.

  E.g. information extraction, machine reading system, image segmentation, etc.

- **Method**: Methods , models, systems to use, or tools, components of a system, frameworks.

  E.g. language model, CORENLP, POS parser, kernel method, etc.

- **Evaluation Metric**: Metrics, measures, or entities that can express quality of a system/method.

  E.g. F1, BLEU, Precision, Recall, ROC curve, mean reciprocal rank, mean-squared error, robustness, time complexity, etc.

- **Material**: Data, datasets, resources, Corpus, Knowledge base.

  E.g. image data, speech data, stereo images, bilingual dictionary, paraphrased questions, CoNLL, Panntreebank, WordNet, Wikipedia, etc.

- **Evaluation Metric**: Metric measure or term that can express quality of a system/method.

  E.g. F1, BLEU, Precision, Recall, ROC curve, mean reciprocal rank, mean-squared error,robustness, compile time, time complexity...

- **Generic**: General terms or pronouns that may refer to a entity but are not themselves informative, often used as connection words.

  E.g model, approach, prior knowledge, them, it...

A.2 Relation Category

Relation link can not go beyond sentence boundary. We define 4 asymmetric relation types (*Used-for*, *Feature-of*, *Hyponym-of*, *Part-of*), together with 2 symmetric relation types (*Compare*, *Conjunction*). B always points to A for asymmetric relations

- **Used-for**: B is used for A, B models A, A is trained on B, B exploits A, A is based on B. E.g.

  > The TISPER system has been designed to enable many text applications.
  >
  > Our method models user proficiency.
  >
  > Our algorithms exploits local soothness.

- **Feature-of**: B belongs to A, B is a feature of A, B is under A domain. E.g.

  > prior knowledge of the model
  >
  > genre-specific regularities of discourse structure
  >
  > English text in science domain

- **Hyponym-of**: B is a hyponym of A, B is a type of A. E.g.

  > TUIT is a software library
  >
  > NLP applications such as machine translation and language generation

- **Part-of**: B is a part of A... E.g.

  > The system includes two models: speech recognition and natural language understanding
  >
  > We incorporate NLU module to the system.

- **Compare**: Symmetric relation (use blue to denote entity). Opposite of conjunction, compare two models/methods, or listing two opposing entities. E.g.

  > Unlike the quantitative prior, the qualitative prior is often ignored...
  >
  > We compare our system with previous sequential tagging systems...

- **Conjunction**: Symmetric relation (use blue to denote entity). Function as similar role or use/incorporate with. E.g.

  > obtained from human expert or knowledge base
  >
  > NLP applications such as machine translation and language generation

A.3 Coreference

Two Entities that points to the same concept.

- **Anaphora and Cataphora**:

  > We introduce a machine reading system... The system...
  >
  > The prior knowledge include...Such knowledge can be applied to...

- **Coreferring noun phrase**:

  > We develop a part-of-speech tagging system...The POS tagger...

A.4 Notes

1. Entity boundary annotation follows the ACL RD-TEC Annotation Guideline (QasemiZadeh and Schumann, 2016), with the extention that spans can be embedded in longer spans, only if the shorter span is involved in a relation.

2. Do not include determinators (such as the, a), or adjective pronouns (such as this,its, these, such) to the span. If generic phrases are not involved in a relation, do not tag them.

3. Do not tag relation if one entity is:

    - Variable bound:
      We introduce a neural based approach.. *Its* benefit is...
    - The word *which*:
      We introduce a neural based approach, *which* is a...

4. Do not tag coreference if the entity is

    - Generically-used Other-ScientificTerm:
      ...advantage gained from *local smoothness* which... We present algorithms exploiting *local smoothness* in more aggressive ways...
    - Same scientific term but refer to different examples:
      We use a *data structure*, we also use another *data structure*...

5. Do not label negative relations:

    X is not used in Y or X is hard to be applied in Y

B Annotation and Knowledge Graph Examples

Here we take a screen shot of the BRAT interface for an ACL paper in Figure 9. We also attach the original figure of Figure 3 in Figure 10. More examples can be found in the project website[4].

---

[4] `http://nlp.cs.washington.edu/sciIE/`

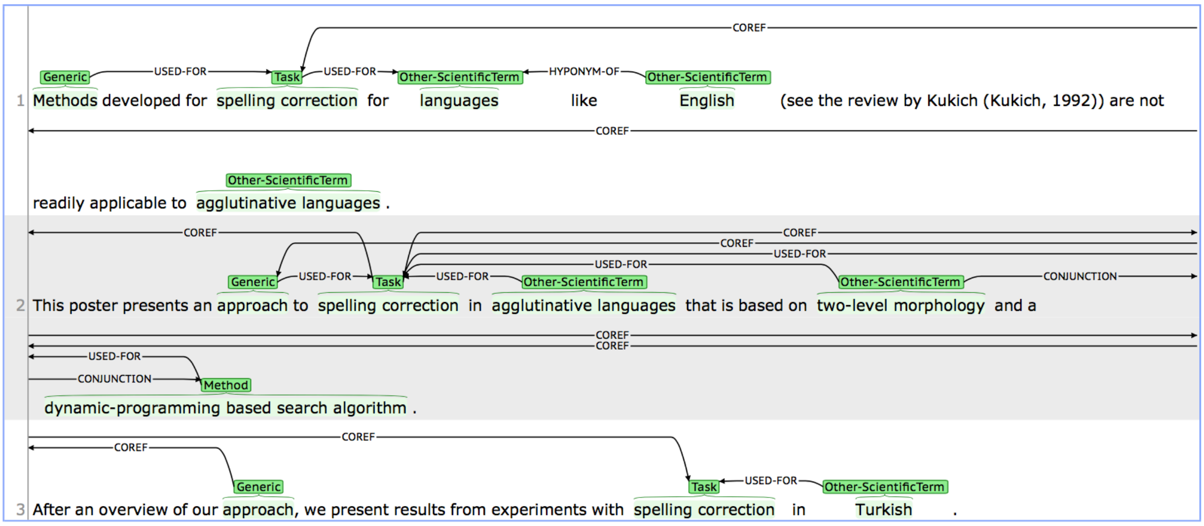

1 Methods developed for spelling correction for languages like English (see the review by Kukich (Kukich, 1992)) are not readily applicable to agglutinative languages .

2 This poster presents an approach to spelling correction in agglutinative languages that is based on two-level morphology and a dynamic-programming based search algorithm .

3 After an overview of our approach, we present results from experiments with spelling correction in Turkish .

Figure 9: Annotation example 1 from ACL

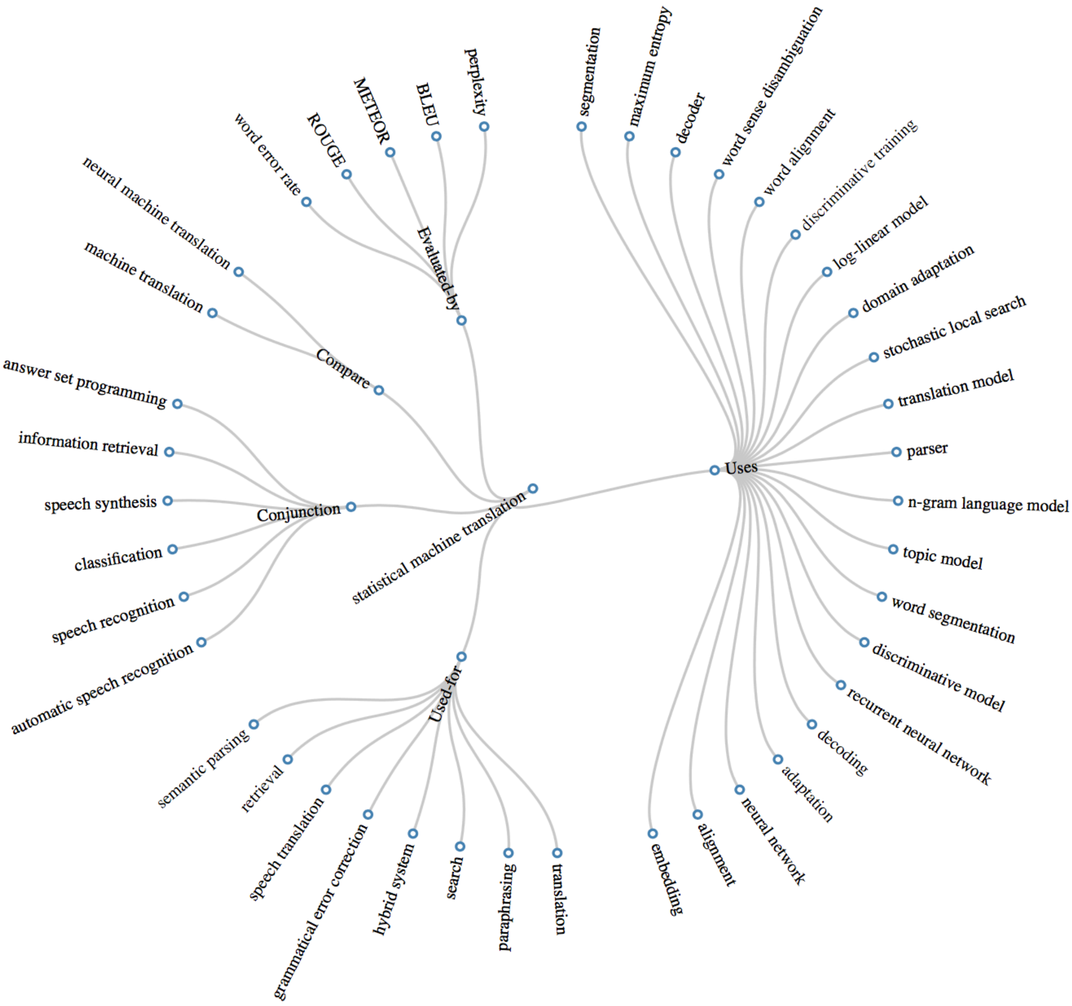

Figure 10: An example of our automatically generated knowledge graph centered on *statistical machine translation*. This is the original figure of Figure 4.

Playing 20 Question Game with Policy-Based Reinforcement Learning

Huang Hu[1]*, **Xianchao Wu**[3], **Bingfeng Luo**[1], **Chongyang Tao**[1],
Can Xu[2], **Wei Wu**[2] and **Zhan Chen**[3]
[1]Peking University, Beijing, China
[2]Microsoft Corporation, Beijing, China
[3]Microsoft Development Co., Ltd, Tokyo, Japan
[1]{tonyhu,bf_luo,chongyangtao}@pku.edu.cn
[2,3]{xiancwu,can.xu,wuwei,zhanc}@microsoft.com

Abstract

The 20 Questions (Q20) game is a well known game which encourages deductive reasoning and creativity. In the game, the answerer first thinks of an object such as a famous person or a kind of animal. Then the questioner tries to guess the object by asking 20 questions. In a Q20 game system, the user is considered as the answerer while the system itself acts as the questioner which requires a good strategy of question selection to figure out the correct object and win the game. However, the optimal policy of question selection is hard to be derived due to the complexity and volatility of the game environment. In this paper, we propose a novel policy-based Reinforcement Learning (RL) method, which enables the questioner agent to learn the optimal policy of question selection through continuous interactions with users. To facilitate training, we also propose to use a reward network to estimate the more informative reward. Compared to previous methods, our RL method is robust to noisy answers and does not rely on the Knowledge Base of objects. Experimental results show that our RL method clearly outperforms an entropy-based engineering system and has competitive performance in a noisy-free simulation environment.

1 Introduction

The 20 Question Game (Q20 Game) is a classic game that requires deductive reasoning and creativity. At the beginning of the game, the ***answerer*** thinks of a target object and keeps it concealed. Then the ***questioner*** tries to figure out the target object by asking questions about it, and the answerer answers each question with a simple “Yes”, “No” or “Unknown”, honestly. The questioner wins the game if the target object is found within 20 questions. In a Q20 game system, the user is considered as the answerer while the system itself acts as the questioner which requires a good question selection strategy to win the game.

As a game with the hype *read your mind*, Q20 has been played since the 19th century, and was brought to screen in the 1950s by the TV show *Twenty Questions*. Burgener’s program (Burgener, 2006) further popularized Q20 as an electronic game in 1988, and modern virtual assistants like Microsoft XiaoIce and Amazon Alexa also incorporate this game into their system to demonstrate their intelligence.

However, it is not easy to design the algorithm to construct a Q20 game system. Although the decision tree based method seems like a natural fit to the Q20 game, it typically require a well defined Knowledge Base (KB) that contains enough information about each object, which is usually not available in practice. Burgener (2006) instead uses a object-question relevance table as the pivot for question and object selection, which does not depend on an existing KB. Wu et al. (2018) further improve the relevance table with a lot of engineering tricks. Since these table-based methods greedily select questions and the model parameters are only updated by rules, their models are very sensitive to noisy answers from users, which is common in the real-world Q20 games. Zhao and Maxine (2016) utilizes a value-based Reinforcement Learning (RL) model to improve the generalization ability but still relies on the existing KB.

In this paper, we formulate the process of question selction in the game as a Markov Decision Process (MDP), and further propose a novel policy-based RL framework to learn the optimal policy of question selection in the Q20 game. Our questioner agent maintains a probability distribution over all objects to model the confidence of the target object, and updates the confidence based on answers from the user. At each time-step. the agent uses a policy network $\pi_\theta(a|\boldsymbol{s})$ to take in

---

[*] The work was done when the first author was an intern in Microsoft XiaoIce team.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3233–3242
Brussels, Belgium, October 31 - November 4, 2018.

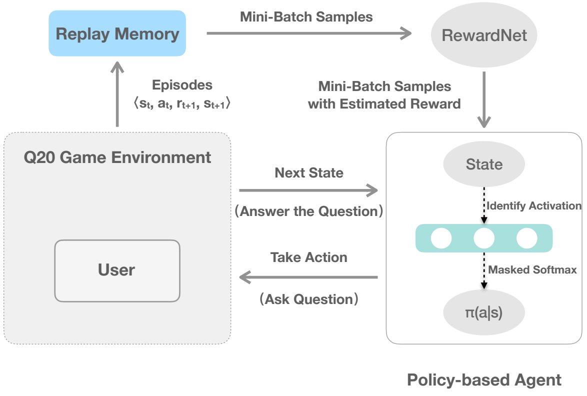

Figure 1: The overview of our RL framework.

the confidence vector and output a question distribution for selecting the next question. To solve the problem that there is no immediate reward for each selected question, we also propose to employ a RewardNet to estimate the appropriate immediate reward at each time-step, which is further used to calculate the long-term return to train our RL model. Our RL framework makes the agent robust to noisy answers since the model parameters are fully learnable and the question distribution from $\pi_\theta(a|\boldsymbol{s})$ provides us with a principled way to sample questions, which enables the agent to jump out of the local optimum caused by incorrect answers and also introduces more randomness during training to improve the model generalization ability. Furthermore, the ability to sample questions, compared to greedy selection, also improves the diversity of the questions asked by our agent, which is crucial for user experience.

Our contributions can be summarized as follows: (1) We propose a novel RL framework to learn the optimal policy of question selection in the Q20 game without any dependencies on the existing KBs of target objects. Our trained agent is robust to noisy answers and has a good diversity in its selected questions. (2) To make the reward more meaningful, we also propose a novel neural network on reward function approximation to deliver the appropriate immediate rewards at each time-step. (3) Extensive experiments show that our RL method clearly outperforms a highly engineered baseline in the real-world Q20 games where noisy answers are common. Besides, our RL method is also competitive to that baseline on a noise-free simulation environment.

2 Method

In this section, we first describe our RL framework for playing the Q20 game, which is shown in the Fig. 1. The user in our system is the answerer who thinks of a target object o_{tgt} in the object set $\mathcal{O}$ at the beginning of the game. Our policy-based agent acts as the questioner that can ask 20 questions to figure out what exactly o_{tgt} is. Specifically, an internal state vector $\boldsymbol{s}$ is maintained by our agent, which describes the confidence about o_{tgt}. At each time-step t, the agent picks up the promising action (select a question) according to the policy $\pi_\theta(a|\boldsymbol{s}_t)$, and transits from the state $\boldsymbol{s_t}$ to the next state $\boldsymbol{s_{t+1}}$ after receiving the answer ("Yes"/"No"/"Unknown") from the user. The historical trajectories $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$ are stored in a *replay memory* which enables the agent to be trained on previously observed data by sampling from it. Note that only when a guess is made about o_{tgt} at the end of game can the agent receive a reward signal, which makes it unable to distinguish the importance of each selected question. Therefore, we design a RewardNet to learn the more informative reward at each time-step and thus lead the agent to achieve the better performance.

In the rest of this section, we first describe how to formulate the Q20 game into a RL framework, and then introduce the RewardNet. Finally, we will demonstrate our training procedure in detail.

2.1 Modeling of the Q20 Game

In the Q20 game, the goal of our agent is to figure out the object o_{tgt} that the user thinks of at the beginning of game by asking 20 questions. We formulate the process of question selection as a finite Markov Decision Process (MDP) which can be solved with RL. A tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\rangle$ is defined to represent the MDP, where $\mathcal{S}$ is the continuous state space, $\mathcal{A} = \{a_1, a_2, \cdots, a_m\}$ is the set of all available actions, $\mathcal{P}(S_{t+1} = s' | S_t = s, A_t = a)$ is the transition probability matrix, $\mathcal{R}(s, a)$ is the reward function and $\gamma \in [0, 1]$ is the discount factor used to calculate the long-time return. In the RL framework, at each time-step t, the agent takes an action a_t under the state s_t according to the policy $\pi_\theta(a|s_t)$. After interacting with the environment, the agent receives a reward scalar r_{t+1} and transits to the next state s_{t+1}, then another time-step begins. All these trajectories $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$ in a game constitute an episode which is an instance of the finite MDP. The long-time return G_t of the time-step t is calculated as follows:

$$G_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{1}$$

In the following parts, we describe each component of RL corresponding to the Q20 game.

Environment. The major component of our environment is the user in the Q20 game who decides the target object o_{tgt} and answers questions from the agent. Besides, the environment also needs to deliver the reward based on the outcome of the game and store historical data into the replay memory (see Fig. 1).

Action. Since the agent interacts with the user by asking questions, the action $a_t \in \mathcal{A}$ taken by our agent refers to selecting the question q_{a_t} at time-step t, and $\mathcal{A}$ is the set of the indices to all available questions in the Q20 game.

State. In our method, we use the state $\boldsymbol{s}_t$ to keep track of the current confidence of target object o_{tgt}. Specifically $\boldsymbol{s}_t \in \mathbb{R}^{|\mathcal{O}|}$ and $\sum_{i=1}^{n} s_{t,i} = 1$, where $\mathcal{O} = \{o_1, o_2, \cdots, o_n\}$ represents the set of all the objects that can be chosen by the user. Therefore, the state $\boldsymbol{s}_t$ is a probability distribution over all the objects and $s_{t,i}$ is the confidence that the object o_i is the target object o_{tgt} at time-step t.

The initial state $\boldsymbol{s}_0$ can either be a uniform distribution or initialized by the prior knowledge. We observe that users typically prefer to choose popular objects which are more concerned by the public. For example, the founder of Tesla Inc. and the designer of SpaceX, "Elon Musk", is more likely to be chosen compared to a CEO of a new startup. Motivated by this, we could use the yearly retrieval frequency $C(o_i)$ of object o_i on a commercial search engine to calculate the initial state $\boldsymbol{s}_0$, where $s_{0,i} = C(o_i) \;/\; \sum_{j=1}^{n} C(o_j)$.

Transition Dynamics. In our method, the transition dynamics is deterministic. Given the object set $\mathcal{O}$ and the question set $\mathcal{A}$, we collect the normalized probabilities of the answer over "Yes", "No" and "Unknown" for each object-question pair. And the rule of state transition is define as:

$$\boldsymbol{s}_{t+1} = \boldsymbol{s}_t \odot \boldsymbol{\alpha} \tag{2}$$

where $\boldsymbol{\alpha}$ depends on the answer x_t to the question q_{a_t} which is selected by the agent at the step t:

$$\alpha = \begin{cases} [R(1, a_t), \dots, R(|\mathcal{O}|, a_t)], & x_t = Yes \\ [W(1, a_t), \dots, W(|\mathcal{O}|, a_t)], & x_t = No \\ [U(1, a_t), \dots, U(|\mathcal{O}|, a_t)], & x_t = Unk \end{cases} \tag{3}$$

where $\mathcal{O}$ is the object set and for each object-question pair (o_i, q_j), $R(i, j)$ and $W(i, j)$ are calculated as follows:

$$\begin{aligned} R(i,j) &= \frac{C_{yes}(i,j) + \delta}{C_{yes}(i,j) + C_{no}(i,j) + C_{unk}(i,j) + \lambda} \\ W(i,j) &= \frac{C_{no}(i,j) + \delta}{C_{yes}(i,j) + C_{no}(i,j) + C_{unk}(i,j) + \lambda} \end{aligned} \tag{4}$$

$R(i, j)$ and $W(i, j)$ are probabilities of answering "Yes" and "No" to question q_j with respect to the object o_i respectively. $C_{yes}(i, j)$, $C_{no}(i, j)$ and $C_{unk}(i, j)$ are frequencies of answering "Yes", "No" and "Unknown" to question q_j with respect to the object o_i. δ and λ are smoothing parameters. Then the probability of answering "Unknown" to question q_j with respect to the object o_i is:

$$U(i, j) = 1 - R(i, j) - W(i, j) \tag{5}$$

In this way, the confidence $s_{t,i}$ that the object o_i is the target object o_{tgt} is updated following the user's answer x_t to the selected question q_{a_t} at the time-step t.

Policy Network. We directly parameterize the policy $\pi_\theta(a|\boldsymbol{s_t})$ with a neural network which maps the state $\boldsymbol{s_t}$ to a probability distribution over all available actions: $\pi_\theta(a|\boldsymbol{s_t}) = \mathbb{P}[a|\boldsymbol{s_t}; \theta]$. The parameters θ are updated to maximize the expected return which is received from the environment. Instead of learning a greedy policy in value-based methods like DQN, the policy network is able to learn a stochastic policy which can increase the diversity of questions asked by our agent and potentially make the agent more robust to noisy answers in the real-world Q20 game. The policy $\pi_\theta(a|\boldsymbol{s})$ is modeled by a Multi-Layer Perceptron (MLP) and the output layer is normalized by using a masked softmax function to avoid selecting the question that has been asked before. Because asking the same question twice does not provide extra information about o_{tgt} in a game.

2.2 Problem of Direct Reward

For most reinforcement learning applications, it is always a critical part to design reward functions, especially when the agent needs to precisely take actions in a complex task. A good reward function can improve the learning efficiency and help the agent achieve better performances.

In the Q20 game, however, the immediate reward r_t of selecting question q_{a_t} is unknown at the time-step t ($t < T$) because each selected question is just answered with a simple "Yes", "No" or

"Unknown" and there is no extra information provided by user. Only when the game ends $(t = T)$ can the agent receive a reward signal of win or loss. So we intuitively consider the ***direct reward***: $r_T = 30$ and -30 for the win and loss respectively while $r_t = 0$ for all $t < T$. Unfortunately, the direct reward is not discriminative because the agent receives the same immediate reward $r_t = 0\,(t < T)$ for selecting both good and bad questions. For example, if the o_{tgt} is "Donald Trump", then selecting question (a) "Is your role the American president?" should receive more immediate reward r_t than selecting question (b) "Has your role been married?". The reason is that as for the o_{tgt}, question (a) is more relevant and can narrow down the searching space to a greater extent.

Therefore, it is necessary to design a better reward function to estimate a non-zero immediate reward r_t, and make the long-time return $G_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$ more informative.

2.3 Reward Function Approximation by Neural Network

To solve the problem of the direct reward, we propose a reward function which employs a neural network to estimate a non-zero immediate reward r_t at each time-step. So that G_t can be more informative, which thus leads to a better trained questioner agent.

The reward function takes the state-action pair (s_t, a_t) as input and outputs the corresponding immediate reward r_{t+1}. In our method, we use a MLP with sigmoid output to learn the appropriate immediate reward during training, and this network is referred as ***RewardNet***. In each episode, the long-term return G_t is used as a surrogate indicator of r_{t+1} to train our RewardNet with the following loss function:

$$L_1(\sigma) = (R(s_t, a_t; \sigma) - \text{sigmoid}(G_t))^2 \quad (6)$$

where σ is the network parameters. Here we apply the sigmoid function on G_t so as to prevent G_t from growing too large. Besides, we also use the replay memory to store both old and recent experiences, and then train the network by sampling mini-batches from it. The training process based on the experience replay technique can decorrelate the sample data and thus make the training of the RewardNet more efficient.

Furthermore, since the target object o_{tgt} can be obtained at the end of each episode, we can use the extra information provided by o_{tgt} to estimate a better immediate reward r_t. To capture the relevance between the selected questions and o_{tgt} in an episode, we further propose a ***object-aware RewardNet*** which takes the $\langle s_t, a_t, o_{tgt}\rangle$ tuple as input and produces corresponding r_{t+1} as output. The detailed training algorithm is shown in Algo. 1.

Algorithm 1: Training Object-Aware RewardNet

```
Initialize replay memory $\mathcal{D}_1$ to capacity $N_1$
Initialize RewardNet with random weights $\sigma$
for episode $i \leftarrow 1$ to $Z$ do
    User chooses object $o_i$ from $\mathcal{O}$
    Initialize temporary set $S_1$ and $S_2$
    Play with policy $\pi_\theta(a_t|s_t)$, and store $(s_t, a_t)$ in $S_1$, where $t \in [0, T]$
    $r_T \leftarrow 30$ or $-30$ for a win or loss
    for $(s_t, a_t)$ in $S_1$ do
        Get $r_{t+1}$ from RewardNet
        Store $(s_t, a_t, r_{t+1})$ tuple in $S_2$
    for $(s_t, a_t, r_{t+1})$ in $S_2$ do
        $G_t \leftarrow \sum_{k=0}^{T} \gamma^k r_{t+k+1}$
        $r'_{t+1} \leftarrow sigmoid(G_t)$
        Store $(s_t, a_t, o_i, r'_{t+1})$ in $\mathcal{D}_1$
        if $len(\mathcal{D}_1) > K_1$ then
            Sample mini-batch from $\mathcal{D}_1$
            Update $\sigma$ with loss $L_1(\sigma)$ in Eq. 6
```

2.4 Training the Policy-Based Agent

We train the policy network using REINFORCE (Williams, 1992) algorithm and the corresponding loss function is defined as follows:

$$L_2(\theta) = -\mathbb{E}_{\pi_\theta}[\log \pi_\theta(a_t|s_t)(G_t - b_t)] \quad (7)$$

where the baseline b_t is a estimated value of the expected future reward at the state s_t, which is produced by a value network $\mathcal{V}_\eta(s_t)$. Similarly, the value network $\mathcal{V}_\eta(s_t)$ is modeled as a MLP which takes the state s_t as input and outputs a real value as the expected return. By introducing the baseline b_t for the policy gradient, we can reduce the variance of gradients and thus make the training process of policy network more stable. The network parameters η are updated by minimizing the loss function below:

$$L_3(\eta) = (\mathcal{V}_\eta(s_t) - G_t)^2 \quad (8)$$

Note that, in our method, both the RewardNet and the value network $\mathcal{V}_\eta(s_t)$ approximate the reward during training. But the difference lies in that the RewardNet is designed to estimate a appropriate non-zero reward r_t and further derive the more informative return G_t while $\mathcal{V}_\eta(s_t)$ aims to learn a baseline b_t to reduce the variance of policy gradients. We combine both of two networks to improve the gradients for our policy network and thus lead to a better agent. The training procedure is described in Algo. 2.

Algorithm 2: Training the Agent

```
Initialize replay memory $\mathcal{D}_2$ to capacity $N_2$
Initialize policy net $\pi$ with random weights $\theta$
Initialize value net $\mathcal{V}$ with random weights $\eta$
Initialize RewardNet with random weights $\sigma$
for $episode\ i \leftarrow 1$ to $Z$ do
    Rollout, collect rewards, and save the
      history in $S_2$ (4-10 in Algo. 1)
    for $(s_t, a_t, r_{t+1})$ in $S_2$ do
        $G_t \leftarrow \sum_{k=0}^{T} \gamma^k r_{t+k+1}$
        Update RewardNet (13-17 in
          Algo. 1)
        Store $(s_t, a_t, G_t)$ in $\mathcal{D}_2$
        if $len(\mathcal{D}_2) > K_2$ then
            Sample mini-batch from $\mathcal{D}_2$
            Update $\eta$ with loss $L_3$ in Eq. 8
            Update $\theta$ with loss $L_2$ in Eq. 7
```

3 Experimental Setup

We use a user simulator to train our questioner agent and test the agent with the simulated answerer and real users. Specifically, our experiments answer three questions: (1) Is our method more robust in real-world Q20 games, compared to the methods based on relevance table? (Section. 4.2) And how does it perform in the simulation environment? (Section. 4.1) (2) Does our RewardNet help in the training process? (Section. 4.3) (3) How the winning rate grows with the number of questions, and whether it is possible to stop earlier? (Section. 4.4)

3.1 User Simulator

Training the RL agent is challenging because the agent needs to continuously interact with the environment. To speed up the training process of the proposed RL model, we construct a user simulator which has enough prior knowledge to choose objects and answer questions selected by the agent.

We collect 1,000 famous people and 500 questions for them. Besides, for every person-question pair in our dataset, a prior frequency distribution over "Yes", "No" and "Unknown" is also collected from thousands of real users. For example, as for "Donald Trump", question (a) "Is your role the American president?" is answered with "Yes" for 9,500 times, "No" for 50 times and "Unknown" for 450 times. We use Eq.4 and 5 to construct three matrices $R, W, U \in \mathbb{R}^{|\mathcal{O}|*|\mathcal{A}|}$ ($|\mathcal{O}| = 1000, |\mathcal{A}| = 500$) which are used for state transition in the Section. 2.1. Then given the object o_i and question q_j, the user simulator answers "Yes", "No" and "Unknown" when $R(i,j)$, $W(i,j)$, and $U(i,j)$ has the max value among them respectively.

Constructed by the prior knowledge, the simulator can give noise-free answer in most cases. Because the prior frequency distribution for each person-question pair is collected from thousands of users with the assumption that most of them do not lie when answering questions in the Q20 game.

In an episode, the simulator randomly samples a person following the object distribution s_0, which is generated from the object popularity (see the state part of Section. 2.1), as the target object. Then the agent gives a guess when the number of selected questions reaches 20. After that, the simulator check the agent's answer and return a reward signal of win or loss. There is only one chance for the agent to guess in an episode. The win and loss reward are 30 and -30 respectively.

3.2 Implementation Details

While the architectures of the policy network, RewardNet and value network can vary in different scenarios, in this paper, we simply use the MLP with one hidden layer of size 1,000 for all of them, but with different parameters. These networks take in the state vector directly, which is a probability distribution over all objects. The RewardNet further takes in the one-hot vector of action a_t. Based on the input of RewardNet, the object-aware RewardNet takes one more target object o_{tgt} as the feature which is also a one-hot vector.

We use the ADAM optimizer (Kingma and Ba, 2014) with the learning rate 1e-3 for policy network and 1e-2 for both RewardNet and value network. The discounted factor γ for calculating the long-term return is 0.99. The model was trained up

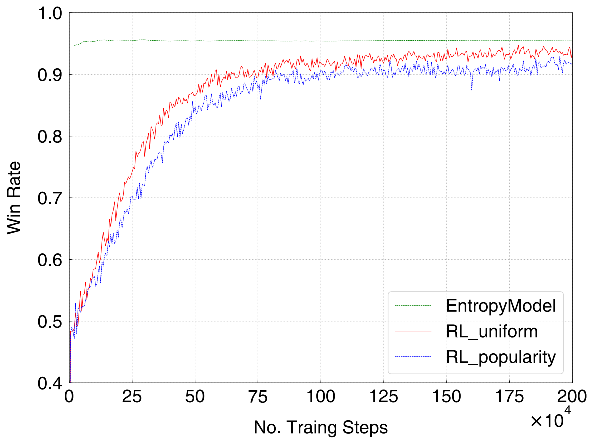

Figure 2: Win Rate Curves in Simulation Environment.

to 2,000,000 steps (2,00,000 games) and the policy network was evaluated every 5,000 steps. Each evaluation records the agent's performance with a greedy policy for 2,000 independent episodes. The 2,000 target objects for these 2,000 episodes are randomly selected following the distribution s_0, which is generated from the object popularity and kept the same for all the training settings.

3.3 Competitor

We compare our RL method with the entropy-based model proposed by Wu et al. (2018), which utilizes the real-world answers to each object-question pair to calculate an object-question relevance matrix with the entropy-based method. The relevance matrix is then used for question ranking and object ranking via carefully designed formulas and engineering tricks. Since this method is shown to be effective in their production environment, we consider it to be a strong baseline to our proposed RL model.

4 Experimental Results

4.1 Simulated Evaluation

We first evaluate our agent and the entropy-based baseline (referred to as `EntropyModel`, see Section. 3.3) by using the simulated user (Section. 3.1). To investigate which initialization strategy of the state s_0 is better (see the state part of Section. 2.1), we further evaluate two variants of our model: the agent with uniform distribution s_0 (`RL_uniform`) and the agent with the distribution s_0 initialized by the prior knowledge on the object popularity (`RL_popularity`).

Fig. 2 shows the curves on the win rate of these methods evaluated on 2,000 independent episodes with respect to the number of training steps. Note that, the `EntropyModel` only needs to update its statistics during training and has already accumulated a significant number of data since it has been run for over a year in their production environment. Therefore, only a small fraction of its statistics can be changed, which leads to a small rise at the beginning of training, and its win rate remains at around 95% afterwards.

On the other hand, both our RL models continuously improve the win rate with the growing number of interactions with the user simulator, and they achieve 50% win rate after around 20,000 steps. As we can see, although the s_0 initialized with the prior knowledge of object popularity keeps consistent with the object selection strategy of the simulator, the agent with uniform distribution s_0 (`RL_uniform`) still performs clearly better than the agent with s_0 based on the prior knowledge (`RL_popularity`). The reason is that the former can explore the Q20 game environment more fully. The prior knowledge based s_0 helps the agent narrow down the candidate space more quickly when the target object is a popular object. However, it also becomes misleading when the target object is not popular and makes the agent even harder to correct the confidence of the target object. On the contrary, the uniform distribution s_0 makes the agent keep track of the target object only based on the user's answers. And the superior performance of the `RL_uniform` indicates that our question selection policy is highly effective, which means it is not necessary to use the `RL_popularity` to increase the win rate of hot objects in the game.

As shown in Fig. 2, `RL_uniform` achieves win rate 94% which is very close to `EntropyModel`. Compared to our RL method, `EntropyModel` needs more user data to calculate their entropy-based relevance matrix and involves many engineering tricks. The fact that `RL_uniform` is competitive to `EntropyModel` in the noise-free simulation environment indicates that our RL method is very cost-effective: it makes use of user data more efficiently and is easier to implement.

4.2 Human Evaluation

To further investigate the performance of our RL method in the real-world Q20 game where noisy answers are common, we also conduct an human evaluation experiment. Specifically, we let real

	Win Rate
`EntropyModel`	71.3%
`RL_uniform`	75.9%

Table 1: Win Rate on Human Evaluation.

users to play the game with `EntropyModel` and `RL_uniform` for 1,000 times respectively. In the real-world Q20 game, users sometimes make mistakes when they answer the questions during the game. For example, as for the target object "Donald Trump", question (a) "Is your role the American president?" is sometimes answered with "No" or "Unknown" by real users. On the contrary, the simulator hardly makes such mistakes since we have provided it with enough prior knowledge. As shown in Table. 1, `RL_uniform` outperforms `EntropyModel` by about 4.5% on win rate in the real-world Q20 games. It shows that our RL method is more robust to noisy answers than `EntropyModel`. Specifically, the robustness of our RL method to the noise is shown in the following two aspects. First, compared to the rule-based statistics update in `EntropyModel`, our RL model can be trained by modern neural network optimizers in a principled way, which results in the better generalization ability of our model. Secondly, different from the `EntropyModel` selecting the top-ranked question at each time-step, `RL_uniform` samples a question following its question probability distribution $\pi_\theta(a|\boldsymbol{s})$, which enables our agent to jump out of the local optimum caused by incorrect answers from users. And since more randomness is introduced by sampling from the question probability distribution during training, it also improves the tolerance of our model towards the unexpected question sequences.

Besides, we also find some interesting cases during human evaluation. Sometimes, the RL agent selects a few strange questions which seems to be not that much relevant to the chosen object, but it can still find the correct answer at the end of game. This situation is caused by the fact that our method samples questions based on the output of policy net, rather than greedy selection during training. We find that this phenomenon increases the user experience since it makes the agent more unpredictable to the users.

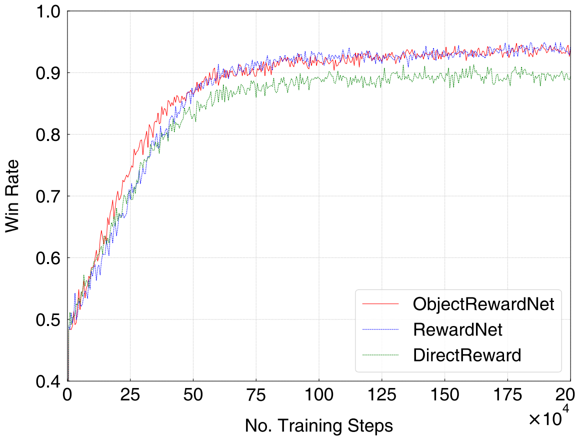

Figure 3: Effectiveness of RewardNet.

4.3 The Effectiveness of RewardNet

To investigate the effectiveness of our RewardNet (Section. 2.3), we further evaluate three variants of our model in the simulation environment: the model trained with with direct reward, RewardNet, and object-aware RewardNet, which are referred to as `DirectReward`, `RewardNet`, and `ObjectRewardNet` respectively. They are all trained with the uniform distribution $\boldsymbol{s}_0$.

As shown in Fig. 3, `DirectReward` converges in the early steps and has a relatively poor performance with the win rate 89%. Both `RewardNet` and `ObjectRewardNet` achieve the better performance with a win rate of 94% after convergence. This clear improvement shows that the more informative long-term return, calculated with the immediate reward delivered by our RewardNet method, significantly helps the training of the agent.

Furthermore, as shown in Fig. 3, we can also see that `ObjectRewardNet` learns faster than `RewardNet` in the early steps. This indicates that `ObjectRewardNet` can estimate the immediate reward more quickly with the extra information provided by the target object, which leads to the faster convergence of the agent.

4.4 Win Rate Regarding Question Numbers

In this section, we investigate how the win rate grows with the number of asked questions and whether a early-stop strategy can be adopted in the game. We use the user simulator to play the game with the `RL_uniform` agent and two settings are taken into account: the simulator samples the target object following the uniform object distribution (`UnifSimulator`), and samples following

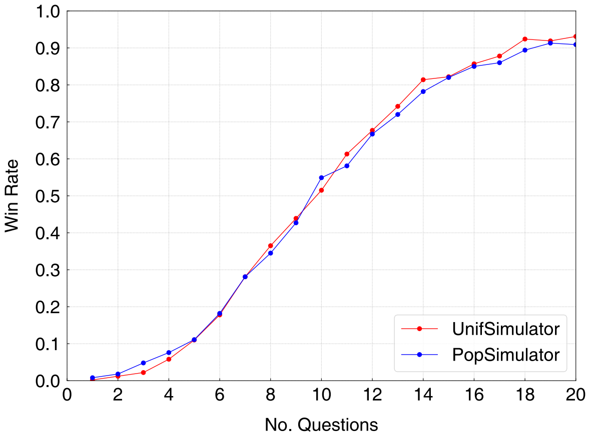

Figure 4: Win Rate Regarding Numbers of Questions.

the prior object distribution based on the object popularity (`PopSimulator`). We perform 1,000 simulations for each number of questions, and the win rate curve is shown in Fig. 4.

As we can see that `UnifSimulator` achieves the win rate of 80% with only 14 questions in both settings. And the flat curves in the region after 18 questions indicate that the game can be early stopped with the almost same win rate at step 18. Since a lower win rate is acceptable sometimes, other early-stop strategies can also be derived for the better user experience with the trade-off between the win rate and game steps.

Besides, the fact that `RL_uniform` performs similarly under both settings actually shows that our RL method is robust to different objects. It also performs well on infrequent objects where we may have the limited user data for constructing a well-tuned state transition dynamics.

4.5 Case Study

When our agent is playing the game with real users, we select two cases from records. In the first case, the person that the user chooses is Cristiano Ronaldo, the famous football player. As we can see in Tab. 2, our agent can still figure out the target person while No.17 and No.19 questions are answered wrong by the user, which indicates our agent is robust to noisy answers. In the second case, the chosen person is Napoleon Bonaparte who was the French Emperor. Although there are some other candidates satisfied the constraints, the target person can be figured out because of the people popularity, which is shown in Tab. 3.

5 Related Work

Q20. The Q20 game is popularized as an electronic game by the program of Robin Burgener in 1988 (Burgener, 2006), which uses a object-question relevance table to rank questions and target objects. Wu et al. (Wu et al., 2018) improves the relevance table with entropy-based metrics, and uses complicated engineering tricks to make it perform quite well in their production environment. These table-based methods use rules to update parameters, which makes them easily affected by noisy answers. Besides, Zhao and Maxine (2016) also explores Q20 in their dialogue state tracking research. However, they only use a small toy Q20 setting where the designed questions are about 6 person attributes in the Knowledge Base (KB). Since their method relies on the KB for narrowing down the scope of target object, it is not applicable to real-world Q20 games where a well-defined object KB is often unavailable. Compared to previous approaches, our RL method is robust to the answer noise and does not rely on the KB.

Deep Reinforcement Learning. DRL has witnessed great success in playing complex games like Atari games (Mnih et al., 2015) , Go (Silver et al., 2016), and etc. In the natural language processing (NLP), DRL is also used to play text-based games (Narasimhan et al., 2015), and used to handle fundamental NLP tasks like machine translation (He et al., 2016) and machine comprehension (Hu et al., 2017) as well. Our Q20 game lies in the intersection of the field of game and NLP. In this work, we propose a policy-based RL model that acts as the questioner in the Q20 game, and it exhibits the superior performance in our human evaluation.

Natural Language Games. In the literature, there are some works focusing on solving and generating English riddles (De Palma and Weiner, 1992; Binsted, 1996) and Chinese character riddles (Tan et al., 2016). Compared to riddles, the Q20 game is a sequential decision process which requires careful modeling of this property.

6 Conclusions

In this paper, we propose a policy-based RL method to solve the question selection problem in the Q20 Game. Instead of using the direct reward, we further propose an object-aware RewardNet to estimate the appropriate non-zero reward and thus make the long-time return more informative. Compared to previous approaches, our RL method is more robust to the answer noise which is common in the real-world Q20 game. Besides, our RL agent can also ask various questions and does not require the existing KB and complicated engineering tricks. The experiments on a noisy-free simulation environment show that our RL method is competitive to an entropy-based engineering system, and clearly outperforms it on the human evaluation where noisy answers are common.

As for the future work, we plan to explore methods to use machine reading to automatically construct the state transition dynamics from corpora like Wikipedia. In this way, we can further build an end-to-end framework for the large-scale Q20 games in the real world.

Acknowledgement

We gratefully thank the anonymous reviewers for their insightful comments and suggestions on the earlier version of this paper. The first author also thanks the Microsoft for providing resources for the research.

References

Kim Binsted. 1996. Machine humour: An implemented model of puns.

Robin Burgener. 2006. Artificial neural network guessing method and game.

Paul De Palma and E Judith Weiner. 1992. Riddles: accessibility and knowledge representation. In *Proceedings of the 14th conference on Computational linguistics-Volume 4*, pages 1121–1125. Association for Computational Linguistics.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *CoRR, abs/1705.02798*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

V Mnih, K Kavukcuoglu, D Silver, A. A. Rusu, J Veness, M. G. Bellemare, A Graves, M Riedmiller, A. K. Fidjeland, and G Ostrovski. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Chuanqi Tan, Furu Wei, Li Dong, Weifeng Lv, and Ming Zhou. 2016. Solving and generating chinese character riddles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 846–855.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Xianchao Wu, Huang Hu, Momo Klyen, Kyohei Tomita, and Zhan Chen. 2018. Q20: Rinna riddles your mind by asking 20 questions. In *Japan NLP*.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 1.

No.	Question	User's Answer
1	Was the person born in Asia?	No
2	Is the person very famous?	Yes
3	Is the person a actor or actress?	No
4	Is the person still alive?	Yes
5	Was the person born in the 1990s?	No
6	Is the person the founder of a famous company?	No
7	Did the person finish the college in USA?	No
8	Is the person a famous singer?	No
9	Is the person male?	Yes
10	Is the person related to sports?	Yes
11	Is the person a football player?	Yes
12	Is the person a midfielder?	No
13	Is the person played for a European football club?	Yes
14	Is the person playing in the Spanish Premier League?	Yes
15	Is the person famous for the handsome or beautiful looks?	Yes
16	Does the person have big muscles?	Yes
17	Does the person have brown hair?	No
18	Will you be happy when you see the person?	Yes
19	Does the person have brothers or sisters in the family?	No
20	Is the person engaged in many charity activities?	Unknown

Table 2: The person that the user chooses is Cristiano Ronaldo, the famous football player. As we can see in table, our agent can still figure out the target person while No.17 and No.19 are answered wrong by the user, which indicates our agent is robust to noisy answers.

No.	Question	User's Answer
1	Is the person female?	No
2	Is the person still alive?	No
3	Does the person have children?	Yes
4	Does the person have brothers or sisters in the family?	Yes
4	Is the person very smart?	Yes
5	Was the person born in America?	No
6	Is the person the white man?	Yes
7	Is the person's family very rich?	No
8	Is the person a controversial figure in history?	Yes
9	Is the person related to politics?	Yes
10	Does the person have good looks?	Unknown
11	Does the person have short hair?	Yes
12	Is the person very famous?	Yes
13	Has the person once been very powerful?	Yes
14	Is the character of the person very aggressive?	No
15	Has the person been the president of a country?	Yes
16	Is the person a military?	Yes
17	Has the person once killed men?	No
18	Was the person born in Britain?	No
19	Was the person one of famous leaders in the World War II?	No
20	Has the person once been the emperor?	Yes

Table 3: In this case, the person that the user chooses is Napoleon Bonaparte, the French Emperor. Although there are some other candidates satisfied the constraints, our agent can figure out the target person because of the people popularity.

Multi-Hop Knowledge Graph Reasoning with Reward Shaping

Xi Victoria Lin **Richard Socher** **Caiming Xiong**
Salesforce Research
{xilin,rsocher,cxiong}@salesforce.com

Abstract

Multi-hop reasoning is an effective approach for query answering (QA) over incomplete knowledge graphs (KGs). The problem can be formulated in a reinforcement learning (RL) setup, where a policy-based agent sequentially extends its inference path until it reaches a target. However, in an incomplete KG environment, the agent receives low-quality rewards corrupted by false negatives in the training data, which harms generalization at test time. Furthermore, since no golden action sequence is used for training, the agent can be misled by spurious search trajectories that incidentally lead to the correct answer. We propose two modeling advances to address both issues: (1) we reduce the impact of false negative supervision by adopting a pretrained one-hop embedding model to estimate the reward of unobserved facts; (2) we counter the sensitivity to spurious paths of on-policy RL by forcing the agent to explore a diverse set of paths using randomly generated edge masks. Our approach significantly improves over existing path-based KGQA models on several benchmark datasets and is comparable or better than embedding-based models.

1 Introduction

Large-scale knowledge graphs (KGs) support a variety of downstream NLP applications such as semantic search (Berant et al., 2013) and dialogue generation (He et al., 2017). Whether curated automatically or manually, practical KGs often fail to include many relevant facts. A popular approach for modeling incomplete KGs is knowledge graph embeddings, which map both entities and relations in the KG to a vector space and learn a truth value function for any potential KG triple parameterized by the entity and relation vectors (Yang et al., 2014; Dettmers et al., 2018).

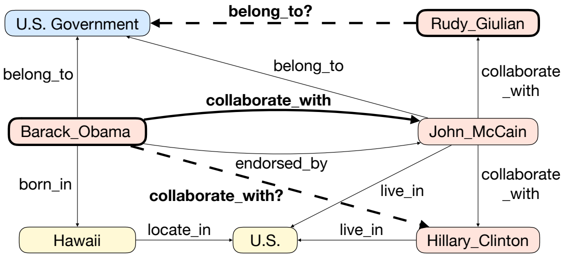

Figure 1: Example of an incomplete knowledge graph which contains missing links (dashed lines) that can possibly be inferred from existing facts (solid lines).

Embedding based approaches ignore the symbolic compositionality of KG relations, which limit their application in more complex reasoning tasks. An alternative solution for KG reasoning is to infer missing facts by synthesizing information from multi-hop paths, e.g. *bornIn(Obama, Hawaii)* $\wedge$ *locatedIn(Hawaii, US)* $\Rightarrow$ *bornIn(Obama, US)*, as shown in Figure 1. Path-based reasoning offers logical insights of the underlying KG and are more directly interpretable. Early work treats it as a link prediction problem and perform maximum-likelihood classification over either discrete path features (Lao et al., 2011, 2012; Gardner et al., 2013) or their hidden representations in a vector space (Guu et al., 2015; Toutanova et al., 2016; McCallum et al., 2017).

More recent work formulates multi-hop reasoning as a sequential decision problem, and leverages reinforcement learning (RL) to perform effective path search (Xiong et al., 2017; Das et al., 2018; Shen et al., 2018; Chen et al., 2018). In particular, MINERVA (Das et al., 2018) uses the REINFORCE algorithm (Williams, 1992) to train an end-to-end model for multi-hop KG query answering: given a query relation and a source entity, the trained agent searches over the KG starting from the source and arrives at the candidate answers without access to any pre-computed paths.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3243–3253
Brussels, Belgium, October 31 - November 4, 2018.

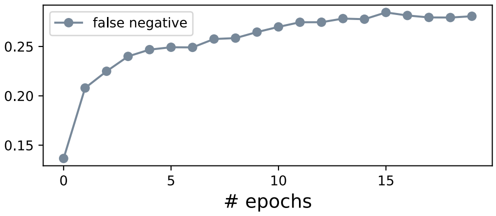

Figure 2: Percentage of false negatives hit (where the model predicted an answer that exists in the full KG but cannot be identified by the training subset) in the first 20 epochs of walk-based QA training on the UMLS knowledge graph (Kok and Domingos, 2007).

We refer to the RL formulation adopted by MINERVA as "learning to walk towards the answer" or "walk-based query-answering (QA)". Walk-based QA eliminates the need to precompute path features, yet this setup poses several challenges for training. First, because practical KGs are intrinsically incomplete, the agent may arrive at a correct answer whose link to the source entity is missing from the training graph without receiving any reward (false negative targets, Figure 2). Second, since no ground truth path is available for training, the agent may traverse spurious paths that lead to a correct answer only incidentally (false positive paths). Because REINFORCE (Williams, 1992) is an on-policy RL algorithm (Sutton and Barto, 1998) which encourages past actions with high reward, it can bias the policy toward spurious paths found early in training (Guu et al., 2017).

We propose two modeling advances for RL approaches in the walk-based QA framework to address the aforementioned problems. First, instead of using a binary reward based on whether the agent has reached a correct answer or not, we adopt pre-trained state-of-the-art embedding-based models (Dettmers et al., 2018; Trouillon et al., 2016) to estimate a ***soft reward*** for target entities whose correctness cannot be determined. As embedding-based models capture link semantics well, unobserved but correct answers would receive a higher reward score compared to a true negative entity using a well-trained model. Second, we perform ***action dropout*** which randomly blocks some outgoing edges of the agent at each training step so as to enforce effective exploration of a diverse set of paths and dilute the negative impact of the spurious ones. Empirically, our overall model significantly improves over state-of-the-art multi-hop reasoning approaches on four out of five benchmark KG datasets (UMLS, Kinship, FB15k-237, WN18RR). It is also the first path-based model that achieves consistently comparable or better performance than embedding-based models. We perform a thorough ablation study and result analysis, demonstrating the effect of each modeling innovation.

2 Approach

In this section, we first review the walk-based QA framework (§2.2) and the on-policy reinforcement learning approach proposed by Das et al. (2018) (§2.3,§2.4). Then we describe our proposed solutions to the false negative reward and spurious path problems: knowledge-based reward shaping (§2.5) and action dropout (§2.6).

2.1 Formal Problem Definition

We formally represent a knowledge graph as $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the set of relations. Each directed link in the knowledge graph $l = (e_s, r, e_o) \in \mathcal{G}$ represents a fact (also called a triple).

Given a query $(e_s, r_q, ?)$, where e_s is the source entity and r_q is the relation of interest, the goal is to perform an efficient search over $\mathcal{G}$ and collect the set of possible answers $E_o = \{e_o\}$ where $(e_s, r_q, e_o) \notin \mathcal{G}$ due to incompleteness.

2.2 Reinforcement Learning Formulation

The search can be formulated as a Markov Decision Process (MDP) (Sutton and Barto, 1998): starting from e_s, the agent sequentially selects an outgoing edge l and traverses to a new entity until it arrives at a target. Specifically, the MDP consists of the following components (Das et al., 2018).

States Each state $s_t = (e_t, (e_s, r_q)) \in \mathcal{S}$ is a tuple where e_t is the entity visited at step t and (e_s, r_q) are the source entity and query relation. e_t can be viewed as the state-dependent information while (e_s, r_q) are the global context shared by all states.

Actions The set of possible actions $A_t \in \mathcal{A}$ at step t consists of the outgoing edges of e_t in $\mathcal{G}$, i.e., $A_t = \{(r', e') | (e_t, r', e') \in \mathcal{G}\}$. To give the agent the option to terminat a search, a self-loop edge is added to every A_t. When search is unrolled for a fixed number of steps T, the self-loop acts similarly to a "stop" action.

Transition A transition function $\delta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is defined by $\delta(s_t, A_t) = \delta(e_t, (e_s, r_q), A_t)$. In walk-based QA, the transition is determined by $\mathcal{G}$.

Rewards In the default formulation, the agent receives a terminal reward of 1 if it arrives at a correct target entity when search ends and 0 otherwise.

$$R_b(s_T) = \mathbb{1}\{(e_s, r_q, e_T) \in \mathcal{G}\}. \tag{1}$$

2.3 Policy Network

The search policy is parameterized using state information and global context, plus the search history (Das et al., 2018).

Specifically, every entity and relation in $\mathcal{G}$ is assigned a dense vector embedding $\mathbf{e} \in \mathbb{R}^d$ and $\mathbf{r} \in \mathbb{R}^d$. A particular action $a_t = (r_{t+1}, e_{t+1}) \in A_t$ is represented as the concatenation of the relation embedding and the end node embedding $\mathbf{a}_t = [\mathbf{r}; \mathbf{e}'_t]$.

The search history $h_t = (e_s, r_1, e_1, \ldots, r_t, e_t) \in \mathcal{H}$ consists of the sequence of actions taken up to step t, and can be encoded using an LSTM:

$$\mathbf{h}_0 = \text{LSTM}(\mathbf{0}, [\mathbf{r}_0; \mathbf{e}_s]) \tag{2}$$

$$\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{a}_{t-1}),\ t > 0, \tag{3}$$

where r_0 is a special start relation introduced to form a start action with e_s.

The action space A_t is encoded by stacking the embeddings of all actions in it: $\mathbf{A}_t \in \mathbb{R}^{|A_t| \times 2d}$. And the policy network π is defined as:

$$\pi_\theta(a_t|s_t) = \sigma(\mathbf{A}_t \times W_2 \,\text{ReLU}(W_1[\mathbf{e}_t; \mathbf{h}_t; \mathbf{r}_q])), \tag{4}$$

where σ is the softmax operator.

2.4 Optimization

The policy network is trained by maximizing the expected reward over all queries in $\mathcal{G}$:

$$J(\theta) = \mathbb{E}_{(e_s, r, e_o) \in \mathcal{G}}[\mathbb{E}_{a_1, \ldots, a_T \sim \pi_\theta}[R(s_T|e_s, r)]]. \tag{5}$$

The optimization is done using the REINFORCE (Williams, 1992) algorithm, which iterates through all (e_s, r, e_o) triples in $\mathcal{G}$[1] and updates θ with the following stochastic gradient:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_{t=1}^{T} R(s_T|e_s, r) \log \pi_\theta(a_t|s_t). \tag{6}$$

2.5 Knowledge-Based Reward Shaping

According to Equation 1, the agent receives a binary reward based solely on the observed answers in $\mathcal{G}$. However, $\mathcal{G}$ is intrinsically incomplete and this approach penalizes the false negative search attempts identically to true negatives. To alleviate this problem, we adopt existing KG embedding models designed for the purpose of KG completion (Trouillon et al., 2016; Dettmers et al., 2018) to estimate a soft reward for target entities whose correctness is unknown.

Formally, the embedding models map $\mathcal{E}$ and $\mathcal{R}$ to a vector space, and estimate the likelihood of each fact $l = (e_s, r, e_t) \in \mathcal{G}$ using $f(e_s, r, e_t)$, a composition function of the entity and relation embeddings. f is trained by maximizing the likelihood of all facts in $\mathcal{G}$. We propose the following reward shaping strategy (Ng et al., 1999):

$$R(s_T) = R_b(s_T) + (1 - R_b(s_T))f(e_s, r_q, e_T). \tag{7}$$

Namely, if the destination e_T is a correct answer according to $\mathcal{G}$, the agent receives reward 1. Otherwise the agent receives a fact score estimated by $f(e_s, r_q, e_T)$, which is pre-trained. Here we keep f in its general form and it can be replaced by any state-of-the-art model (Trouillon et al., 2016; Dettmers et al., 2018) or ensemble thereof.

2.6 Action Dropout

The REINFORCE training algorithm performs on-policy sampling according to $\pi_\theta(a_t|s_t)$, and updates θ stochastically using Equation 6. Because the agent does not have access to any oracle path, it is possible for it to arrive at a correct answer e_o via a path irrelevant to the query relation. As shown in Figure 1, the path *Obama —endorsedBy→ McCain —liveIn→ U.S. ←locatedIn— Hawaii* does not infer the fact $bornIn(Obama, Hawaii)$.

Discriminating paths of different qualities is non-trivial, and existing RL approaches for walk-based KGQA largely rely on the terminal reward to bias the search. Since there are usually more spurious paths than correct ones, spurious paths are often found first, and following exploration can be increasingly biased towards them (Equation 6).

---

[1] This training strategy treats a query with $n > 1$ answers as n single-answer queries. In particular, given a query $(e_s, r_q, ?)$ with multiple answers $\{e_{t_1}, \ldots e_{t_n}\}$, when training w.r.t. the example (e_s, r_q, e_{t_i}), MINERVA removes all $\{e_{t_j} | j \neq i\}$ observed in the training data from the possible set of target entities in the last search step so as to force the agent to walk towards e_{t_i}. We adopt the same technique in our training.

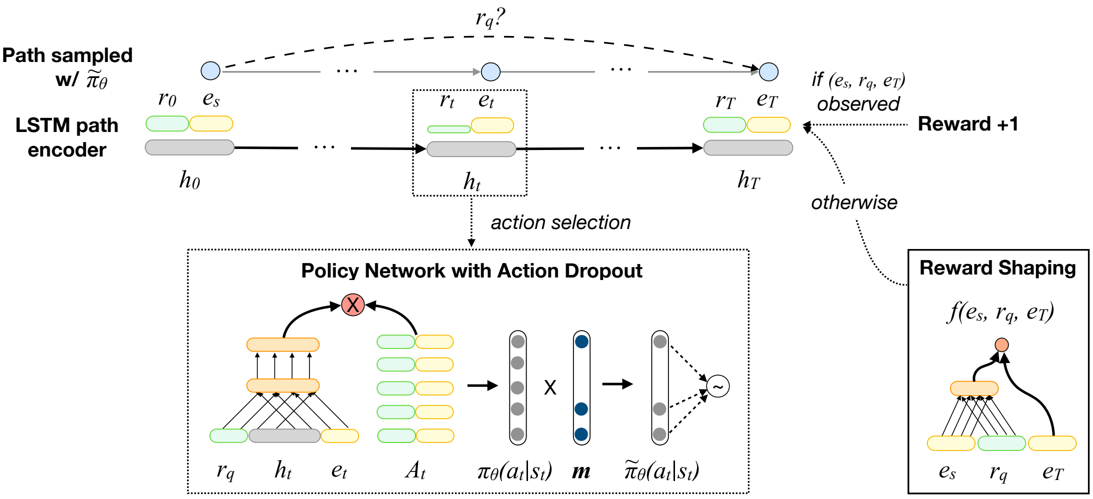

Figure 3: Overall training approach. At each time step t, the agent samples an outgoing link according to $\tilde{\pi}_\theta(a_t|s_t)$, which is the stochastic REINFORCE policy $\pi_\theta(a_t|s_t)$ perturbed by a random binary mask $\mathbf{m}$. The agent receives reward 1 if stopped at an observed answer of the query $(e_s, r_q, ?)$; otherwise, it receives reward $f(e_s, r_q, e_T)$ estimated by the reward shaping (RS) network. The RS network is pre-trained and doesn't receive gradient updates.

Entities with larger fan-in (in-degree) and fan-out (out-degree) often exacerbate this problem.

Guu et al. (2017) identified a similar issue in RL-based semantic parsing with weak supervision, where programs that do not semantically match the user utterance frequently pass the tests. To solve this problem, Guu et al. (2017) proposed randomized beam search combined with a meritocratic update rule to ensure all trajectories that obtain rewards are up-weighted roughly equally.

Here we propose the action dropout technique which achieves similar effect as randomized search and is simpler to implement over graphs. Action dropout randomly masks some outgoing edges for the agent in the sampling step of REINFORCE. The agent then performs sampling[2] according to the adjusted action distribution

$$\tilde{\pi}_\theta(a_t|s_t) \propto (\pi_\theta(a_t|s_t) \cdot \mathbf{m} + \epsilon) \tag{8}$$

$$m_i \sim Bernoulli(1-\alpha), i = 1, \dots |A_t|, \tag{9}$$

where each entry of $\mathbf{m} \in \{0,1\}^{|A_t|}$ is a binary variable sampled from the Bernoulli distribution with parameter $1-\alpha$. A small value ϵ is used to smooth the distribution in case $\mathbf{m} = \mathbf{0}$, where $\tilde{\pi}_\theta(a_t|s_t)$ becomes uniform.

Our overall approach is illustrated in Figure 3.

3 Related Work

In this section, we summarize the related work and discuss their connections to our approach.

3.1 Knowledge Graph Embeddings

KG embeddings (Bordes et al., 2013; Socher et al., 2013; Yang et al., 2014; Trouillon et al., 2016; Dettmers et al., 2018) are one-hop KG modeling approaches which learn a scoring function $f(e_s, r, e_o)$ to define a fuzzy truth value of a triple in the embedding space. These models can be adapted for query answering by simply return the e_o's with the highest $f(e_s, r, e_o)$ scores. Despite their simplicity, embedding-based models achieved state-of-the-art performance on KGQA (Das et al., 2018). However, such models ignore the symbolic compositionality of KG relations, which limits their usage in more complex reasoning tasks. The reward shaping (RS) strategy we proposed is a step to combine their capability in modeling triple semantics with the symbolic reasoning capability of the path-based approach.

3.2 Multi-Hop Reasoning

Multi-hop reasoning focus on learning symbolic inference rules from relational paths in the KG and has been formulated as sequential decision problems in recent works (Xiong et al., 2017; Das et al., 2018; Shen et al., 2018; Chen et al., 2018). In particular, DeepPath (Xiong et al., 2017) first adopted REINFORCE to search for generic representative paths between pairs of entities. DIVA (Chen et al., 2018) also performs generic path search between entities using RL and its variational objective can be interpreted as model-based reward assignment. MINERVA (Das et al., 2018) first introduced RL

---

[2] We only modify the sampling distribution and still use $\pi_\theta(a_t|s_t)$ to compute the gradient update in equation 6.

to search for answer entities of a particular KG query end-to-end. MINERVA uses entropy regularization to softly encourage the policy to sample diverse paths, and we show that hard action dropout is more effective in this setup. ReinforceWalk (Shen et al., 2018) further proposed to solve the reward sparsity problem in walk-based QA using off-policy learning. ReinforceWalk scores the search targets with a value function which is updated based on the search history cached through epochs. In comparison, we leveraged existing embedding-based models for reward shaping, which is much more efficient during training.

3.3 Reinforcement Learning

Recently, RL has seen a variety of applications in NLP including machine translation (Ranzato et al., 2015), summarization (Paulus et al., 2017), and semantic parsing (Guu et al., 2017). Compared to the domain of gaming (Mnih et al., 2013) where RL is mostly applied for, RL formulations in NLP often have a large discrete action space. For example, in machine translation, the space of possible actions is the entire vocabulary of a language. Walk-based QA also suffers from this problem, as some entities may have thousands of neighbors (e.g. $U.S.$). Since often there is no golden path available for a KG reasoning problem, we cannot leverage supervised pre-training to initialize the path search following the common practice in RL-based natural language generation (Ranzato et al., 2015). On the other hand, the inference paths being studied in a KG are often much shorter (usually containing 2-5 steps) compared to the target sentences in the NL generation problems (often containing 20-30 words), which simplifies the training to some extent.

4 Experiment Setup

We evaluate our modeling contributions on five KGs from different domains and exhibiting different graph properties (§ 4.1). We compare with two classes of state-of-the-art KG models: multi-hop neural symbolic approaches and KG embeddings (§4.2). In this section, we describe the datasets and our experiment setup in detail.

4.1 Dataset

We adopt five benchmark KG datasets for query answering: (1) Alyawarra Kinship, (2) Unified Medical Language Systems (Kok and Domingos, 2007), (3) FB15k-237 (Toutanova et al., 2015), (4) WN18RR (Dettmers et al., 2018), and (5) NELL-995 (Xiong et al., 2017). The statistics of the datasets are shown in Table 1.

Dataset	**#Ent**	**#Rel**	**#Fact**	**#degree**	
				mean	**median**
Kinship	104	25	8,544	85.15	82
UMLS	135	46	5,216	38.63	28
FB15k-237	14,505	237	272,115	19.74	14
WN18RR	40,945	11	86,835	2.19	2
NELL-995	75,492	200	154,213	4.07	1

Table 1: KGs used in the experiments sorted by increasing sparsity level.

4.2 Baselines and Model Variations

We compare with three embedding based models: DistMult (Yang et al., 2014), ComplEx (Trouillon et al., 2016) and ConvE (Dettmers et al., 2018). We also compare with three multi-hop neural symbolic models: (a) NTP-λ, an improved version of Neural Theorem Prover (Rocktäschel and Riedel, 2017), (b) Neural Logical Programming (NeuralLP) (Yang et al., 2017) and (c) MINERVA. For our own approach, we include two model variations that use ComplEx and ConvE as the reward shaping modules respectively, denoted as Ours(ComplEx) and Ours(ConvE). We quote the results of NeuralLP, NTP-λ and MINERVA reported in Das et al. (2018), and replicated the embedding based systems.[3]

4.3 Implementation Details

Beam Search Decoding We perform beam search decoding to obtain a list of *unique* entity predictions. Because multiple paths may lead to the same target entity, we compute the list of unique entities reached in the final search step and assign each of them the maximum score of all paths that led to it. We then output the top-ranked unique entities. We find this approach to improve over directly taking the entities ranked at the beam top, as many of them are repetitions.

KG Setup Following previous work, we treat every KG link as bidirectional and augment the graph with the reversed (e_o, r^{-1}, e_s) links. We use the same train, dev, and test set splits as Das et al. (2018). We exclude any link from the dev and

---

[3] Das et al. (2018) reported MINERVA results with the entity embedding usage as an extra hyperparameter – the quoted performance of MINERVA in Table 2 on UMLS and Kinship were obtained with entity embeddings setting to zero. In contrast, our system always uses trained entity embeddings.

Model	UMLS			Kinship			FB15k-237			WN18RR			NELL-995		
	@1	@10	MRR	@1	@10	MRR	@1	@10	MRR	@1	@10	MRR	@1	@10	MRR
DistMult (Yang et al., 2014)	82.1	96.7	86.8	48.7	90.4	61.4	32.4	60.0	41.7	**43.1**	52.4	**46.2**	55.2	78.3	64.1
ComplEx (Trouillon et al., 2016)	89.0	99.2	93.4	**81.8**	**98.1**	**88.4**	32.8	61.6	42.5	41.8	48.0	43.7	64.3	86.0	72.6
ConvE (Dettmers et al., 2018)	**93.2**	**99.4**	**95.7**	79.7	**98.1**	87.1	**34.1**	**62.2**	**43.5**	40.3	**54.0**	44.9	**67.8**	**88.6**	**76.1**
NeuralLP (Yang et al., 2017)	64.3	96.2	77.8	47.5	91.2	61.9	16.6	34.8	22.7	37.6	**65.7**	46.3	–	–	–
NTP-λ (Rocktäschel et. al. 2017)	84.3	**100**	91.2	75.9	87.8	79.3	–	–	–	–	–	–	–	–	–
MINERVA (Das et al., 2018)	72.8	96.8	82.5	60.5	92.4	72.0	21.7	45.6	29.3	41.3	51.3	44.8	**66.3**	83.1	72.5
Ours(ComplEx)	88.7	98.5	92.9	**81.1**	**98.2**	**87.8**	**32.9**	54.4	39.3	**43.7**	54.2	**47.2**	65.5	83.6	72.2
Ours(ConvE)	**90.2**	99.2	**94.0**	78.9	**98.2**	86.5	32.7	**56.4**	**40.7**	41.8	51.7	45.0	65.6	**84.4**	**72.7**

Table 2: Query answering performance compared to state-of-the-art embedding based approaches (top part) and multi-hop reasoning approaches (bottom part). The @1, @10 and MRR metrics were multiplied by 100. We highlight the best approach in each category.

test set (and its reversed link) from the train set. Following Das et al. (2018), we cut the maximum number of outgoing edges of an entity by threshold η to prevent GPU memory overflow: for each entity we keep its top-η neighbors with the highest PageRank scores (Page et al., 1999) in the graph.

Hyperparameters We set the entity and relation embedding size to 200 for all models. We use Xavier initialization (Glorot and Bengio, 2010) for the embeddings and the NN layers. For ConvE, we use the same convolution layer and label smoothing hyperparameters as Dettmers et al. (2018). For path-based models, we use a three-layer LSTM as the path encoder and set its hidden dimension to 200. We perform grid search on the reasoning path length $(2, 3)$, the node fan-out threshold η (256-512) and the action dropout rate α (0.1-0.9). Following Das et al. (2018), we add an entropy regularization term in the objective and tune the weight parameter β within 0-0.1. We use Adam optimization (Kingma and Ba, 2014) and search the learning rate (0.001-0.003) and mini-batch size (128-512).[4] For all models we apply dropout to the entity and relation embeddings and all feed-forward layers, and search the dropout rates within 0-0.5. We use a decoding beam size of 512 for NELL-995 and 128 for the other datasets.

Evaluation Protocol We convert each triple (e_s, r, e_o) in the test set into a query and compute ranking-based evaluation metrics. The models take e_s, r as the input and output a list of candidate answers $E_o = [e^1, \ldots, e^L]$ ranked in decreasing order of confidence score. We compute r_{e_o}, the rank of e_o among E_o, after removing the other correct answers from E_o and use it to compute two types of metrics: (1) Hits@k which is the percentage of examples where $r_{e_o} \leq k$ and (2) mean reciprocal rank (MRR) which is the mean of $1/r_{e_o}$ for all examples in the test set. We use the entire test set for evaluation, with the exception of NELL-995, where test triples with unseen entities are removed following Das et al. (2018).

Our Pytorch implementation of all experiments is released at `https://github.com/salesforce/MultiHopKG`.

5 Results

5.1 Model Comparison

Table 2 shows the evaluation results of our proposed approach and the baselines. The top part presents embedding based approaches and the bottom part presents multi-hop reasoning approaches.[5]

We find embedding based models perform strongly on several datasets, achieving overall best evaluation metrics on UMLS, Kinship, FB15K-237 and NELL-995 despite their simplicity. While previous path based approaches achieve comparable performance on some of the datasets (WN18RR, NELL-995, and UMLS), they perform significantly worse than the embedding based models on the other datasets (9.1 and 14.2 absolute points lower on Kinship and FB15k-237 respectively). A possible reason for this is that embedding based methods map every link in the KG into the same embedding space, which implicitly encodes the connectivity of the whole graph. In contrast, path based models use the discrete represen-

---

[4] On some datasets, we found larger batch size to continue improving the performance but had to stop at 512 due to memory constraints.

[5] We report the model robustness measurements in § A.1.

Model	UMLS	Kinship	FB15k237	WN18RR	NELL995
Ours(ConvE)	**73.0**	**75.0**	**38.2**	43.8	**78.8**
−RS	67.7	66.5	35.1	**45.7**	78.4
−AD	61.3	65.4	31.0	39.1	76.1

Table 3: Comparison of dev set MRR of Ours(ConvE) and models without reward shaping and action dropout.

tation of a KG as input, and therefore have to leave out a significant proportion of the combinatorial path space by selection. For some path based approaches, computation cost is a bottleneck. In particular, NeuralLP and NTP-λ failed to scale to the larger datasets and their results are omitted from the table, as Das et al. (2018) reported.

Ours is the first multi-hop reasoning approach which is consistently comparable or better than embedding based approaches on all five datasets. The best single model, Ours(ConvE), improves the SOTA performance of path-based models on three datasets (UMLS, Kinship, and FB15k-237) by 4%, 9%, and 39% respectively. On NELL-995, our approach did not significantly improve over existing SOTA. The NELL-995 dataset consists of only 12 relations in the test set and, as we further detail in the analysis (§ 5.3.3), our approach is less effective for those relation types.

The model variations using different reward shaping modules perform similarly. While a better reward shaping module typically results in a better overall model, an exception is WN18RR, where ComplEx performs slightly worse on its own but is more helpful for reward shaping. We left the study of the relationship between the reward shaping module accuracy and the overall model performance as future work.

5.2 Ablation Study

We perform an ablation study where we remove reward shaping (−RS) and action dropout (−AD) from Ours(ConvE) and compare their MRRs to the whole model on the dev sets.[6] As shown in Table 3, on most datasets, removing each component results in a significant performance drop. The exception is WN18RR, where removing the ConvE reward shaping module improves the performance.[7] Removing reward shaping on NELL-995 does not change the results significantly. In general, removing action dropout has a greater impact, suggesting that thorough exploration of the path space is important across datasets.

5.3 Analysis

5.3.1 Convergence Rate

We are interested in studying the impact of each proposed enhancement on the training convergence rate. In particular, we expect reward shaping to accelerate the convergence of RL (to a better performance level) as it propagates prior knowledge about the underlying KG to the agent. On the other hand, a fair concern for action dropout is that it can be slower to train, as the agent is forced to explore a more diverse set of paths. Figure 4 eliminates this concern.

The first row of Figure 4 shows the changes in dev set MRR of Ours(ConvE) (green $*$) and the two ablated models w.r.t. # epochs. In general, the proposed approach is able to converge to a higher accuracy level much faster than either of the ablated models and the performance gap often persists until the end of training (on UMLS, Kinship, and FB15k-237). Particularly, on FB15k-237, our approach still shows improvement even after the two ablated models start to overfit, with −AD beginning to overfit sooner. On WN18RR, introducing reward shaping hurt dev set performance from the beginning, as discussed in § 5.2. On NELL-995, Ours(ConvE) performs significantly better in the beginning, but −RS gradually reaches a comparable performance level.

It is especially interesting that introducing action dropout immediately improves the model performance on all datasets. A possible explanation for this is that by exploring a more diverse set of paths the agent learns search policies that generalize better.

5.3.2 Path Diversity

We also compute the total number of unique paths the agent explores during training and visualize its change w.r.t. # training epochs in the second row of Figure 4. When counting a unique path, we include both the edge label and intermediate entity.

---

[6] According to Table 3 and Table 2, the dev and test set evaluation metrics differ significantly on several datasets. We discuss the cause of this in § A.2.

[7] A possible explanation for this is that as path-based models tend to outperform the embedding based approaches on WN18RR, ConvE may be supplying more noise than useful information about the KG. Yet counter-intuitively, we found that adding the ComplEx reward shaping module helps, despite the fact that ComplEx performs slightly worse than ConvE on this dataset. This indicates that dev set accuracy is not the only factor which determines the effectiveness of reward shaping.

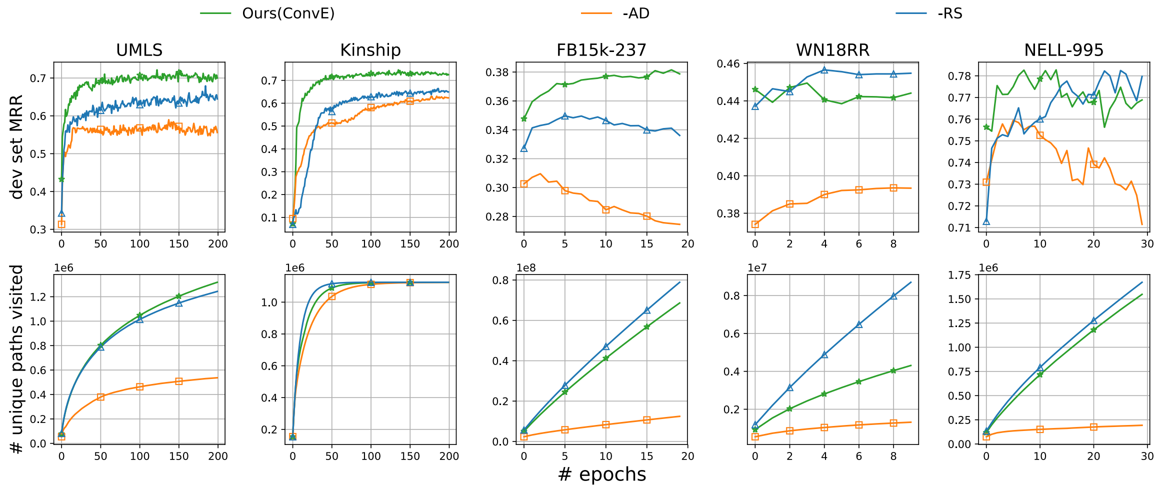

Figure 4: Illustration of convergence rate and path exploration efficiency. The three curves in each subplot represents Ours(ConvE) (green $*$) and the two ablated models: $-$RS (blue $\triangle$) and $-$AD (orange $\square$). The top row shows the change of dev set MRR and the bottom row shows the growth of # unique paths explored w.r.t. # epochs.

Dataset	**To-many**				**To-one**			
	%	Ours(ConvE)	$-$RS	$-$AD	%	Ours(ConvE)	$-$RS	$-$AD
UMLS	99.1	73.1	67.9 (**-7**%)	61.3 (**-16**%)	0.9	62.5	55.5 (**-11**%)	54.4 (**-13**%)
Kinship	100	75	66.5 (**-11**%)	65.4 (**-13**%)	0	–	–	–
FB15k-237	76.6	28.3	24.5 (**-13**%)	20.9 (**-26**%)	23.4	72	69.8 (**-3**%)	63.9 (**-11**%)
WN18RR	52.8	65	65.7 (+1%)	57.9 (**-11**%)	47.2	20.1	23.2 (+16%)	18.1 (**-10**%)
NELL-995	12.9	55.7	62.1 (+12%)	56.9 (+2%)	87.1	81.4	80.7 (**-1**%)	80.5 (**-1**%)

Table 4: MRR evaluation of different relation types (to-many vs. to-one) on five datasets. The % columns show the percentage of examples of each relation type found in the development split of the corresponding dataset. In general, our proposed techniques improve the prediction results for to-many relations more significantly.

First we observe that, on all datasets, the agent explores a large number of paths before reaching a good performance level. The speed of path discovery slowly decreases as training progresses. On smaller KGs (UMLS and Kinship), the rate of encountering new paths is significantly lower after a certain number of epochs, and the dev set accuracy plateaus correspondingly. On much larger KGs (FB15k-237, WN18RR, and NELL-995), we did not observe a significant slowdown before severe overfitting occurs and the dev set performance starts to drop. A possible reason for this is that the larger KGs are more sparsely connected compared to the smaller KGs (Table 1), therefore it is less efficient to gain generalizable knowledge from the KG by exploring a limited proportion of the path space through sampling.

Second, while removing action dropout significantly lowers the effectiveness of path exploration (orange $\square$ vs. green $*$), we observe that removing reward shaping (blue $\triangle$) slightly increases the # paths visited if the action dropout rate is kept the same. This indicates that the correlation between # paths explored and dev set performance is not strictly positive. The best performing model is not always the model that explored the largest # paths. It also demonstrates the role of reward shaping as a regularizer which guides the agent to avoid noisy paths with its prior knowledge.

5.3.3 Performance w.r.t. Relation Types

We investigate the behaviors of our proposed approach w.r.t different relation types. For each KG, we classify its set of relations into two categories based on the answer set cardinality. Specifically, we define the metric ξ_r as the average answer set cardinality of all queries with topic relation r. We count r as a "to-many" relation if $\xi_r > 1.5$, which indicates that most queries in relation r has more than 1 correct answer; we count r as a "to-one" relation otherwise, meaning most queries of this relation have only 1 correct answer.

Table 4 shows the percentage of examples of to-many and to-one relations on each dev dataset and the MRR evaluation metrics of previously studied models computed on the examples of each relation

Dataset	**Seen Queries**				**Unseen Queries**			
	%	Ours(ConvE)	−RS	−AD	%	Ours(ConvE)	−RS	−AD
UMLS	97.2	73.1	67.9 (**-7**%)	61.4 (**-16**%)	2.8	68.5	61.5 (**-10**%)	58.7 (**-14**%)
Kinship	96.8	75.1	66.5 (**-11**%)	65.8 (**-12**%)	3.2	73.6	64.3 (**-13**%)	53.3 (**-27**%)
FB15k-237	76.1	28.3	24.3 (**-14**%)	20.6 (**-27**%)	23.9	70.9	69.1 (**-2**%)	63.9 (**-10**%)
WN18RR	41.8	60.8	62.0 (+2%)	53.4 (**-12**%)	58.2	31.5	33.9 (+7%)	28.8 (**-9**%)
NELL-995	15.3	40.4	45.9 (+14%)	42.5 (+5%)	84.7	85.5	84.7 (**-1**%)	84.3 (**-1**%)

Table 5: MRR evaluation of seen queries vs. unseen queries on five datasets. The % columns show the percentage of examples of seen/unseen queries found in the development split of the corresponding dataset.

type. Since UMLS and Kinship are densely connected, they almost exclusively contain to-many relations. FB15k-237 mostly contains to-many relations. In Figure 4, we observe the biggest relative gains from the ablated models on these three datasets. WN18RR is more balanced and consists of slightly more to-many relations than to-one relations. The NELL-995 dev set is a unique one which almost exclusively consists of to-one relations. There is no common performance pattern over the two relation types across datasets: on some datasets all models perform better on to-many relations (UMLS, WN18RR) while others show the opposite trend (FB15k-237, NELL-995). We leave the study of these discrepancies to future work.

We show the relative performance change of the ablated models −RS and −AD w.r.t. Ours(ConvE) in parentheses. We observe that in general our proposed enhancements are effective in improving query-answering over both relation types (more effective for to-many relations). However, adding the ConvE reward shaping module on WN18RR hurts the performance over both to-many and to-one relations (more for to-one relations). On NELL-995, both techniques hurt the performance over to-many relations.

5.3.4 Performance w.r.t. Seen Queries vs. Unseen Queries

Since most benchmark datasets randomly split the KG triples into train, dev and test sets, the queries that have multiple answers may fall into multiple splits. As a result, some of the test queries $(e_s, r_q, ?)$ are seen in the training set (with a different set of answers) while the others are not. We investigate the behaviors of our proposed approach w.r.t. seen and unseen queries.

Table 5 shows the percentage of examples associated with seen and unseen queries on each dev dataset and the corresponding MRR evaluation metrics of previously studied models. On most datasets, the ratio of seen vs. unseen queries is similar to that of to-many vs. to-one relations (Table 4) as a result of random data split, with the exception of WN18RR. On some datasets, all models perform better on seen queries (UMLS, Kinship, WN18RR) while others reveal the opposite trend. We leave the study of these model behaviors to future work. On NELL-995 both of our proposed enhancements are not effective over the seen queries. In most cases, our proposed enhancements improve the performance over unseen queries, with AD being more effective.

6 Conclusions

We propose two modeling advances for end-to-end RL-based knowledge graph query answering: (1) reward shaping via graph completion and (2) action dropout. Our approach improves over state-of-the-art multi-hop reasoning models consistently on several benchmark KGs. A detailed analysis indicates that the access to a more accurate environment representation (reward shaping) and a more thorough exploration of the search space (action dropout) are important to the performance boost.

On the other hand, the performance gap between RL-based approaches and the embedding-based approaches for KGQA remains. In future work, we would like to investigate learnable reward shaping and action dropout schemes and apply model-based RL to this domain.

Acknowledgements

We thank Mark O. Riedl, Yingbo Zhou, James Bradbury and Vena Jia Li for their feedback on early draft of the paper, and Mark O. Riedl for helpful conversations on reward shaping. We thank the anonymous reviewers and the Salesforce research team members for their thoughtful comments and discussions. We thank Fréderic Godin for pointing out an error in Equation 8 in an early version of the paper.

References

Regina Barzilay and Min-Yen Kan, editors. 2017. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In (Yarowsky et al., 2013), pages 1533–1544.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2787–2795, USA. Curran Associates Inc.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. 2018. Variational knowledge graph reasoning. *CoRR*, abs/1803.06581.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom M. Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In (Yarowsky et al., 2013), pages 833–838.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 318–327. The Association for Computational Linguistics.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In (Barzilay and Kan, 2017), pages 1051–1062.

Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors. 2017. *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In (Barzilay and Kan, 2017), pages 1766–1776.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Stanley Kok and Pedro M. Domingos. 2007. Statistical predicate invention. In *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 433–440. ACM.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 529–539, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ni Lao, Amarnag Subramanya, Fernando C. N. Pereira, and William W. Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1017–1026. ACL.

Andrew McCallum, Arvind Neelakantan, Rajarshi Das, and David Belanger. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 132–141. Association for Computational Linguistics.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 278–287. Morgan Kaufmann.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In (Guyon et al., 2017), pages 3791–3803.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. Reinforcewalk: Learning to walk in graph with monte carlo tree search. *CoRR*, abs/1802.04394.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 926–934, USA. Curran Associates Inc.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, pages 1499–1509. The Association for Computational Linguistics.

Kristina Toutanova, Xi Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 564–573. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575.

Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In (Guyon et al., 2017), pages 2316–2325.

David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors. 2013. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL.

Neural Transductive Learning and Beyond: Morphological Generation in the Minimal-Resource Setting

Katharina Kann
Center for Data Science
New York University, USA
`kann@nyu.edu`

Hinrich Schütze
CIS
LMU Munich, Germany
`inquiries@cislmu.org`

Abstract

Neural state-of-the-art sequence-to-sequence (seq2seq) models often do not perform well for small training sets. We address paradigm completion, the morphological task of, given a partial paradigm, generating all missing forms. We propose two new methods for the minimal-resource setting: (i) *Paradigm transduction*: Since we assume only few paradigms available for training, neural seq2seq models are able to capture relationships between paradigm cells, but are tied to the idiosyncracies of the training set. Paradigm transduction mitigates this problem by exploiting the input subset of inflected forms at test time. (ii) *Source selection with high precision (SHIP)*: Multi-source models which learn to automatically select one or multiple sources to predict a target inflection do not perform well in the minimal-resource setting. SHIP is an alternative to identify a reliable source if training data is limited. On a 52-language benchmark dataset, we outperform the previous state of the art by up to 9.71% absolute accuracy.

1 Introduction

Morphological generation of previously unencountered word forms is a crucial problem in many areas of natural language processing (NLP). High performance can lead to better systems for downstream tasks, e.g., machine translation (Tamchyna et al., 2017). Since existing lexicons have limited coverage, learning morphological inflection patterns from labeled data is an important mission and has recently been the subject of multiple shared tasks (Cotterell et al., 2016, 2017a).

In morphologically rich languages, words inflect, i.e., they change their surface form in oder to express certain properties, e.g., number or tense. A word's canonical form, which can be found in a dictionary, is called the lemma, and the set of all inflected forms is referred to as the lemma's paradigm.

	Singular	Plural
NOM	*Schneemann*	*Schneemänner*
GEN	**Schneemannes**	*Schneemänner*
DAT	*Schneemann*	**Schneemännern**
ACC	*Schneemann*	*Schneemänner*
LEMMA	Schneemann	

Figure 1: The paradigm of the German noun "Schneemann" ("snowman"). In this running example, the input subset is bold, the output subset italic.

In this work, we address paradigm completion (PC), the morphological task of, given a partial paradigm of a lemma, generating all of its missing forms. For the partial paradigm represented by the input subset {("Schneemannes", GEN;SG), ("Schneemännern", DAT;PL)} of the German noun "Schneemann" shown in Figure 1, the goal of PC is to generate the output subset consisting of the six remaining forms.

Neural seq2seq models define the state of the art for morphological generation if training sets are large; however, they have been less successful in the low-resource setting (Cotterell et al., 2017a). In this paper, we address an even more extreme *minimal-resource* setting: for some of our experiments, our training sets only contain $k \approx 10$ paradigms. Each paradigm has multiple cells, so the number of *forms* (as opposed to the number of *paradigms*) is not necessarily minimal. However, we will see that generalizing from paradigm to paradigm is a key challenge, making the number of paradigms a good measure of the effective training set size.

We propose two PC methods for the minimal-resource setting: *paradigm transduction* and *source selection with high precision (SHIP)*. We define a learning algorithm as *transductive*[1] if its goal is to generalize from specific training examples to specific test examples (Vapnik, 1998). In contrast, in-

---

[1] In order to avoid ambiguity, "transduction" is never used in the sense of string-to-string transduction in this paper.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3254–3264
Brussels, Belgium, October 31 - November 4, 2018.

ductive inference learns a general model that is independent of any test set. Predictions of transductive inference for the same item are different for different test sets. There is no such dependence in inductive inference. Our motivation for transduction is that, in the minimal-resource setting, neural seq2seq models capture relationships between paradigm cells like affix substitution and umlauting, but are tied to the idiosyncracies of the k training paradigms. For example, if all source forms in the training set start with "b" or "d", a purely inductive model may then be unable to generate targets with different initials. By transductive inference on the information available in the input subset at test time, i.e., the given partial paradigm, our model can learn idiosyncracies. For example, if the input subset sources start with "p", we can learn to generate output subset targets that start with "p". Thus, we exploit the input subset for learning idiosyncracies at test time and then generate the output subset using a modified model. This setup employs standard inductive training (on the training set) for learning general rules of inflectional morphology and transductive inference (on the test set) for learning idiosyncracies. Our use of transduction is innovative in that most previous work has addressed unstructured problems whereas our problem is structured: we complete a paradigm, a complex structure of forms, each of them labeled with a morphological tag. Thus, the test set contains labels, whereas, in transduction for unstructured problems, the test set is a flat set of unlabeled instances. We view our work as an extension of transduction to the structured case, even though not all elements of the theory developed by Vapnik (1998) carry over.

The motivation for our second PC method for limited training data, SHIP, is as follows. Multi-source models can learn which combination of sources most reliably predicts the target in the high-resource, but less well in the minimal-resource setting. SHIP models the relationship between paradigm slots using edit trees (Chrupała et al., 2008), in order to measure how deterministic each transformation is. Then, it identifies the most deterministic source slot for the generation of each target inflection.

Paradigm transduction and SHIP can be employed separately or in combination. Our experiments show that, in an extreme minimal-resource setting, a combination of SHIP and a non-neural approach is most effective; for slightly more data, a combination of a neural model, paradigm transduction and SHIP obtains the best results.

Contributions. (i) We introduce neural paradigm transduction, which exploits the structure of the PC task to mitigate the negative effect of limited training data. (ii) We propose SHIP, a new algorithm for picking a single reliable source for PC in the minimal-resource setting. (iii) On average over all languages of a 52-language benchmark dataset, our approaches outperform state-of-the-art baselines by up to 9.71% absolute accuracy.

2 Paradigm Completion

In this section, we formally define our task, developing the notation for the rest of the paper.

Given the set of morphological tags $T(w)$ of a lemma w, we define the paradigm of w as the set of tuples of inflected form f_k and tag t_k:

$$\pi(w) = \big\{ \big(f_k[w], t_k\big) \big\}_{t_k \in T(w)} \tag{1}$$

The example in Figure 1 thus corresponds to: π(Schneemann) = $\big\{\big($"Schneemann", NOM;SG$\big)$ ... $\big($"Schneemänner", ACC;PL$\big)\big\}$.

A training set in our setup consists of complete paradigms, i.e., all inflected forms of each lemma are available. This simulates a setting in which a linguist annotates complete paradigms, as done, e.g., in Sylak-Glassman et al. (2016). In contrast, each element of the test set is a partial paradigm, which we refer to as the *input subset*. This simulates a setting in which we collect all forms of a lemma occurring in a (manually or automatically) annotated input corpus; this set will generally not be complete. The PC task consists of generating the *output subset* of the paradigm, i.e., the forms belonging to form-tag pairs which are missing from the collected subset.

3 Method

Our approach for PC is based on MED (*Morphological Encoder-Decoder*), a state-of-the-art model for morphological generation in the high-resource case, which was developed by Kann and Schütze (2016b). In this section, we first cover required background on MED and then introduce our new approaches.

3.1 MED

Input and output format. MED converts one inflected form of a paradigm into another, given the two respective tags. Thus, the input of MED is a sequence of subtags of the source and the target form (e.g., NOM and SG are subtags of NOM;SG), as well as the characters of the source form. All elements are represented by embeddings, which are trained together with the model. The output of MED is the character sequence of the target inflected form.

An example from the paradigm in Figure 1 is:

INPUT: DAT^S PL^S GEN^T SG^T Schneemännern
OUTPUT: Schneemannes

Encoder. The model's encoder consists of a bidirectional gated recurrent neural network (GRU) with a single hidden layer. It reads an input vector sequence $x = (x_1,...,x_{X_t})$ and encodes it from two opposite directions into two hidden representations $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ as

$$\overrightarrow{h_t} = \text{GRU}(x_t, \overrightarrow{h_{t-1}}) \tag{2}$$

$$\overleftarrow{h_t} = \text{GRU}(x_t, \overleftarrow{h_{t+1}}) \tag{3}$$

which are concatenated to

$$h_t = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right] \tag{4}$$

Decoder. The decoder, another GRU with a single hidden layer, defines a probability distribution over the output vocabulary, which, for paradigm completion, consists of the characters in the language, as

$$p(y) = \prod_{t=1}^{T_y} \text{GRU}(y_{t-1}, s_{t-1}, c_t) \tag{5}$$

s_t denotes the state of the decoder at step t, and c_t is the sum of the hidden representations of the encoder, weighted by an attention mechanism.

Additional background on the general model architecture is given in Bahdanau et al. (2015); details on MED can be found in Kann and Schütze (2016b).

3.2 Semi-supervised MED

In order to make use of unlabeled data with MED, Kann and Schütze (2017) defined an auxiliary autoencoding task and proposed a multi-task learning approach.

For this extension, an additional symbol is added to the input vocabulary. Each input is then of the form $(\mathbf{A} \mid M^+)\,\Sigma^+$, with $\mathbf{A}$ being a novel tag for autoencoding, Σ being the alphabet of the language, and M being the set of morphological subtags of the source and the target. As for the basic MED model, all parts of the input are represented by embeddings.

The training objective is to maximize the joint likelihood for the tasks of paradigm completion and autoencoding:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) = \; & \textstyle\sum_{(s,t^S,t^T,w)\in\mathcal{D}} && \log p_{\boldsymbol{\theta}}(w \mid e_\theta(t^S,t^T,s)) \\ & +\textstyle\sum_{a\in\mathcal{A}} && \log p_{\boldsymbol{\theta}}(a \mid e_\theta(a))\end{aligned}$$

where $\mathcal{A}$ is a set of autoencoding examples, e_θ is the encoder, and $\mathcal{D}$ is a labeled training set of tuples of source s, morphological source tag t^S, morphological target tag t^T, and target w.

3.3 MED for Paradigm Completion

MED was originally developed for morphological reinflection. Thus, it operates on pairs consisting of a single source and a single target form. In order to use it for paradigm completion, where multiple source forms are given, and multiple target forms are expected, we convert the given data into a suitable format in the way described in the following.

For a lemma w, let $J(w)$ be the set of tags in the input subset. Recall that $J(w)$ is a subset of $T(w)$, the set of all tags, at test time, but that training paradigms are complete, i.e., $J(w) = T(w)$ for the training set.

For both training of the inductive model and paradigm transduction, we generate $|J(w)|(|J(w)|-1)$ training examples

$$(t_i, t_j, f_i[w]) \mapsto f_j[w]$$

one for each pair of different tags in $J(w)$. We also generate autoencoding training examples for all tags in $J(w)$ (removing duplicates):

$$\big(\mathbf{A}, f_i[w]\big) \mapsto f_i[w]$$

For the German lemma "Schneemann", assume:

$J(\text{Schneemann}) = \{\text{GEN;SG,DAT;PL}\}$

at test time. We then produce the following training examples for paradigm transduction:

(DAT^S PL^S GEN^T SG^T Schneemännern) ↦ Schneemannes
(GEN^S SG^S DAT^T PL^T Schneemannes) ↦ Schneemännern
(**A** Schneemannes) ↦ Schneemannes
(**A** Schneemännern) ↦ Schneemännern

For completing a partial paradigm, we then select one source form per target slot (the lemma, unless stated otherwise) and create all forms corresponding to the tags in $J(w) \setminus T(w)$ one by one.

3.4 Paradigm Transduction

Motivation. In the minimal-resource setting, parameter estimates are tied to the idiosyncracies of the lemmas seen in training, due to overfitting. Our example in §1 is that the model has difficulties producing initial letters not seen during training. However, within each paradigm, forms are generally similar; thus, input subset sources contain valuable information about how to generate output

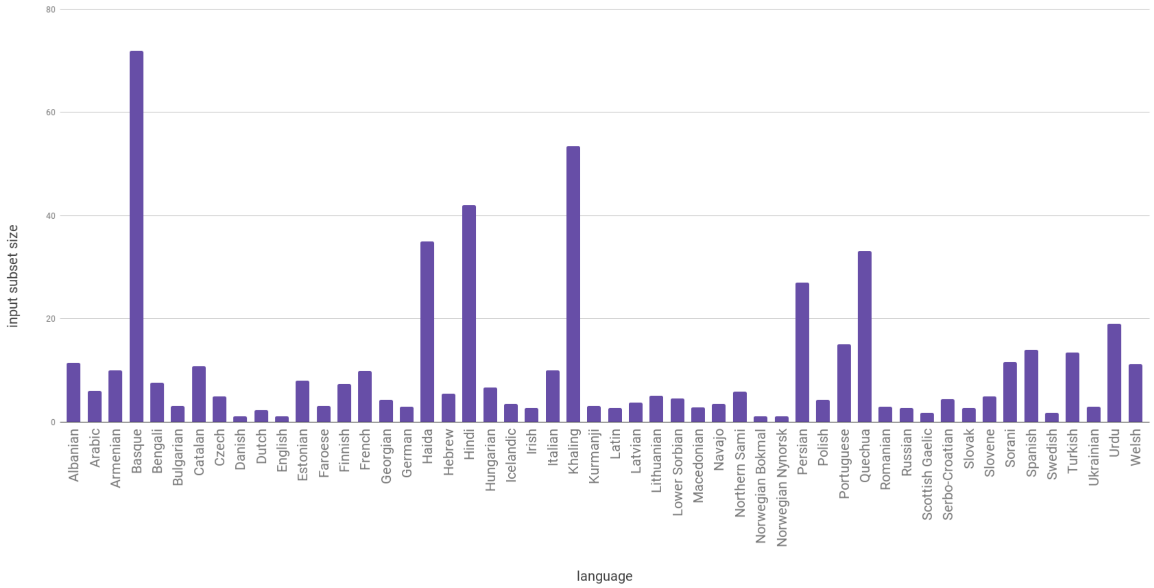

Figure 2: Average amount of sources in the input subset for paradigm transduction, per language.

subset targets. Based on this observation, we solve the problem of overfitting by transduction: we teach the model test idiosyncracies by training it on the input subset before generating the output subset.

Method description. We first train a general model on the training set in the standard supervised learning setup, i.e., the setup which is called inductive inference by Vapnik (1998). At test time, we take the general model as initialization and continue training on examples generated from the input subset as described in §3.3. We do this separately for each lemma, satisfying the defining criterion of transductive inference that predictions depend on the test data. Also, different input subsets (i.e., different subsets of the same paradigm) can in general make different predictions on an output subset target.

Paradigm transduction is expected to perform best in a setting in which many forms of each paradigm are given as input, i.e., when $|J(w)|$ is big. In Figure 2 we show the average sizes of the input subsets for all languages in our experiments.

3.5 Source Selection with High Precision

During PC, some sources contain more information relevant to generating certain targets than others. For instance, the nominative singular and accusative singular in German are generally identical (cf. Figure 1); thus, for generating the accusative singular, we should use the nominative singular as source if it is available—rather than, say, the dative plural.

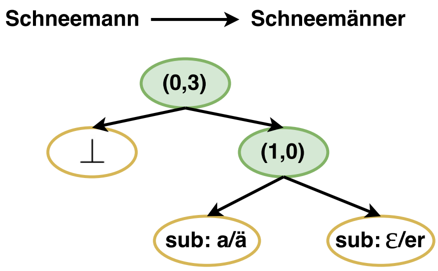

Figure 3: Edit tree example. Each node gives lengths of the parts before/after LCS, e.g., the root has LCS "Schneem", before part ϵ and after part "ann", thus the lengths are "(0,3)". "sub" = "substitution".

In fact, for many languages, the entire paradigm of most lemmas is deterministic if the right source forms are known and used for the right targets. A set of forms that determines all other inflected forms is called *principal parts* (Finkel and Stump, 2007). Based on this theory, Cotterell et al. (2017b) induce topologies and jointly decode entire paradigms, thus making use of all available forms. However, their method is only applicable if good estimates of the probabilities $p(f_j[w]|f_i[w])$ for source $f_i[w]$ and target $f_j[w]$ can be obtained, and they train on hundreds of paradigms per part of speech (POS) and language, which are not available in our setup.

We propose an alternative for the minimal-resource setting: SHIP, which selects a single best source for each target and is based on edit trees. An edit tree $e(f_i[w], f_j[w])$ is a transformation from a source $f_i[w]$ to a target $f_j[w]$ (Chrupała et al., 2008); see Figure 3. It is constructed by first determining

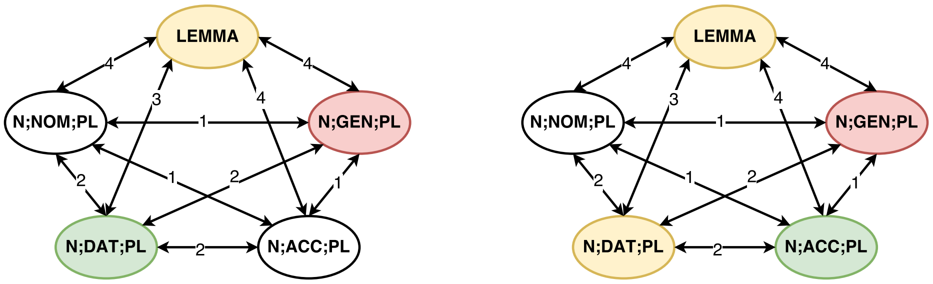

Figure 4: SHIP example for German plural forms (SET1). For the graph constructed in training (see §3.5), subgraphs are extracted in testing for input subset sizes two (left) and three (right). Input subset: yellow and green. Output subset: white and red. For generation of the target shown in red, SHIP selects the source shown in green.

the longest common substring (LCS) (Gusfield, 1997) of $f_i[w]$ and $f_j[w]$ and then modeling the prefix and suffix pairs of the LCS recursively. In the case of an empty LCS, $e(f_i[w], f_j[w])$ is the substitution operation that replaces $f_i[w]$ with $f_j[w]$.

We construct edit trees for each pair $(f_i[w], f_j[w])$ in the training set, count the number n_{ij} of different edit trees for $t_i \mapsto t_j$, and construct a fully connected graph. The tags are nodes of the graph, and the counts n_{ij} are weights. Edges are undirected, since edit trees are bijections (cf. Figure 4). We then interpret the weight of an edge as a measure of the (un)reliability of the corresponding two source-target relationships. Our intuition is that the fewer different edit trees relate source and target, the more reliable the source is for generating the target.

At test time, we find for each target t_j a source t_k such that $n_{kj} \leq n_{ij} \forall i \in J(w)$. We then use $f_k[w]$ to generate $f_j[w]$. Again, Figure 4 shows examples.

4 Experiments

4.1 Data

We run experiments on the datasets from task 2 of the CoNLL–SIGMORPHON 2017 shared task, which have been created using UniMorph (Kirov et al., 2018). We give a short overview here; see (Cotterell et al., 2017a) for details. The dataset contains, for each of 52 languages, a development set of 50 partial paradigms, a test set of 50 partial paradigms, and three training sets of complete paradigms. Training set sizes are 10 (SET1), 50 (SET2), and 200 (SET3). Recall that we view the number of paradigms (not the number of forms) as the best measure of the amount of training data available. Even for SET3, there are only 200 lemmas per language in the training set, which are additionally distributed over multiple POS tags, compared to >600 lemmas per POS used by Cotterell et al. (2017b). We, thus, want to emphasize that all settings—SET1, SET2, and SET3—can be considered low-resource.

We produce training sets for our encoder-decoder as described in §3.3, but limit the total number of training examples to 200,000.

4.2 Hyperparameters

With our hyperparameters, we follow Kann and Schütze (2016a). In particular, our encoder and decoder GRUs have 100-dimensional hidden states. Our embeddings are 300-dimensional. For training, we use stochastic gradient descent, ADADELTA (Zeiler, 2012), and minibatches of size 20. After experiments on the development set, we decide on training SET1, SET2, and SET3 models for 50, 30, and 20 epochs, respectively. For paradigm transduction, we train all models for 25 additional epochs.

4.3 Baselines

In the following, we describe our baselines. COPY, MED, and PT are used for ablation and SIG17 for comparison with the state of the art.

COPY. As targets in many paradigm cells in many languages are identical to the lemma, we consider a copy baseline that simply copies the lemma.

MED. This is the model by Kann and Schütze (2016b), which performed best at SIGMORPHON 2016. For decoding, the lemma is used. Since MED is designed for the high-resource setting, we do not expect good performance for our minimal-resource scenario, but the comparison shows how much our enhancements improve performance.

Pure paradigm transduction (PT). PT is a seq2seq model exclusively trained on the input subset. Its performance sheds light on the importance of the initial inductive training.

SIG17. SIG17 is the official baseline of the CoNLL–SIGMORPHON 2017 shared task, which was developed to perform well with very little

	SET1	SET2	SET3
BL: COPY	.0810	.0810	.0810
BL: MED	.0004	.0432	.4211
BL: PT	.0833	.0833	.0775
BL: SIG17	.5012	.6576	.7707
SIG17+SHIP	**.5971**	.7355	.8008
MED+PT	.5808	.7486	.8454
MED+PT+SHIP	.5793	**.7547**	**.8483**

Table 1: Accuracy on PC for SIG17+SHIP (the shared task baseline SIG17 with SHIP), MED+PT (MED with paradigm transduction), MED+PT+SHIP (MED with paradigm transduction and SHIP), as well as all baselines (*BL*). Results are averaged over all languages, and best results are in bold; detailed accuracies for all languages can be found in Appendix A.

training data. Its design follows Liu and Mao (2016): SIG17 first aligns each input lemma and output inflected form. Afterwards, it assumes that each aligned pair can be split into a prefix, a stem, and a suffix. Based on this alignment, the system extracts prefix (resp. suffix) rules from the prefix (resp. suffix) pairings. At test time, suitable rules are applied to the input string to generate the target; more details can be found in Cotterell et al. (2017a).

4.4 Results

Our results are shown in Table 1. For SET1, SIG17+SHIP obtains the highest accuracy, while, for SET2 and SET3, MED+PT+SHIP performs best. This difference can be easily explained by the fact that the performance of neural networks decreases rapidly for smaller training sets, and, while paradigm transduction strongly mitigates this problem, it cannot completely eliminate it. Overall, however, SIG17+SHIP, MED+PT, and MED+PT+SHIP all outperform the baselines by a wide margin for all settings.

Effect of paradigm transduction. On average, MED+PT clearly outperforms SIG17, the strongest baseline: by .0796 (.5808-.5012) on SET1, .0910 (.7486-.6576) on SET2, and .0747 (.8454-.7707) on SET3.

However, looking at each language individually (refer to Appendix A for those results), we find that MED+PT performs poorly for a few languages, namely Danish, English, and Norwegian (Bokmål & Nynorsk). We hypothesize that this can most likely be explained by the size of the input subset of those languages being small (cf. Figure 2 for average input subset sizes per language). Recall that the input subset is explored by the model during transduction. Most poorly performing languages have input subsets containing only the lemma; in this case paradigm transduction reduces to autoencoding the lemma. Thus, we conclude that paradigm transduction can only improve over MED if two or more sources are given.

Conversely, if we consider only the languages with an average input subset size of more than 15 (Basque, Haida, Hindi, Khaling, Persian, and Quechua), the average accuracy of MED+PT for SET1 is 0.9564, compared to an overall average of 0.5808. This observation shows clearly that paradigm transduction obtains strong results if many forms per paradigm are given.

Effect of SHIP. Further, Table 1 shows that SIG17+SHIP is better than SIG17 by .0959 (.5971-.5012) on SET1, .0779 (.7355-.6576) on SET2, and .0301 (.8008-.7707) on SET3. Stronger effects for smaller amounts of training data indicate that SHIP's strategy of selecting a single reliable source is more important for weaker final models; in these cases, selecting the most deterministic source reduces errors due to noise.

In contrast, the performance of MED, the neural model, is relatively independent of the choice of source; this is in line with earlier findings (Cotterell et al., 2016). However, even for MED+PT, adding SHIP (i.e., MED+PT+SHIP) slightly increases accuracy by .0061 (.7547-.7486) on SET2, and .0029 (.8483-.8454) on SET3 (L53).

Ablation. MED does not perform well for either SET1 or SET2. In contrast, on SET3 it even outperforms SIG17 for a few languages. However, MED loses against MED+PT in all cases, highlighting the positive effect of paradigm transduction.

Looking at PT next, even though PT does not have a zero accuracy for any setting or language, it performs consistently worse than MED+PT. For SET3, PT is even lower than MED on average, by .3436 (.4211-.0775). Note that, in contrast to the other methods, PT's performance is not dependent on the size of the training set. The main determinant for PT's performance is the size of the input subset during transductive inference. If the input subset is large, PT can perform better than MED, e.g., for Hindi and Urdu. For Khaling SET1, PT even outperforms both MED and SIG17. However, in most cases, PT does not perform well on its own.

MED+PT outperforms both MED and PT. This confirms our initial intuition: MED and PT learn complementary information for paradigm

input	output		
	MED	PT	MED+PT
Schneemann N;GEN;PL	GetGächen	Scnneeeeennnnnnnnnnnnnnnnnnnnn	Schneemänner
dish V;V.PTCP;PRS	dising	dish	dishing
creer V;SBJV;PRS;1;PL	crezcamos	creyemos	creamos

Table 2: Analysis of the outputs of MED, PT, and MED+PT for SET2. Top to bottom: German, English, Spanish. MED and PT produce incorrect, MED+PT correct inflections.

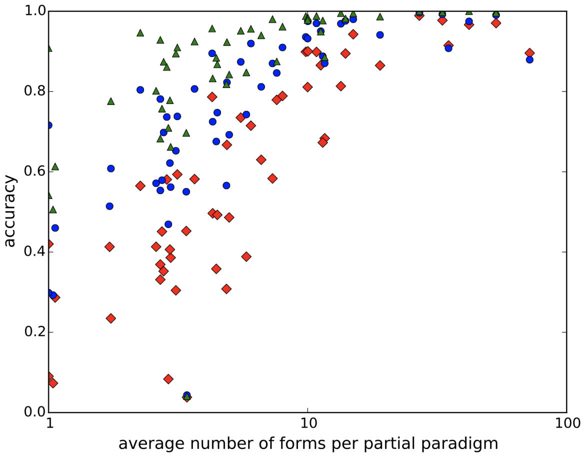

Figure 5: Accuracy of MED+PT as a function of the average input subset size. Red/diamonds: SET1; blue/circles: SET2; green/triangles: SET3.

completion. The base model learns the general structure of the language (i.e., correspondences between tags and inflections) while paradigm transduction teaches the model which character sequences are common in a specific test paradigm.

5 Analysis

5.1 On the Size of the Input Subset

We expect paradigm transduction to become more effective as the size of the input subset increases. Figure 5 shows the accuracy of MED+PT as a function of the average input subset size for SET1, SET2, and SET3. Accuracy for languages with input set sizes above 15 is higher than .8 in all settings. In general, languages with larger input set sizes perform better. The correlation is not perfect because languages have different degrees of morphological regularity. However, the overall trend is clearly recognizable.

The organizers of CoNLL–SIGMORPHON provided large input subsets in the development and test sets of languages with large paradigms. Thus, PT performs better for languages with many inflected forms per paradigm, i.e., large $|T(w)|$.

5.2 On the Effect of Paradigm Transduction

We further analyze why paradigm transduction improves the performance of the base model MED, using the German, English, and Spanish SET2 examples for MED, PT, and MED+PT given in Table 2.

German. MED generates an almost random sequence. However, it learns that the umlaut "ä" must appear in the target. PT only produces correct characters, but it produces far too many. The reason may be that the model is trained on both a double "e" and a double "n", learning that "e" and "n" are likely to appear repeatedly. MED+PT generates the correct target.

English. MED fails to generate "h" because the bigram "sh" did not occur in training, and so the probability of "h" following "s" is estimated to be low. PT fails to produce the suffix "ing", since it does not occur in the input subset, and, thus, PT has no way of learning it. Again, MED+PT generates the correct target.

Spanish. MED produces "crezcamos", a form that has the correct tag V;SBJV;PRS;1;PL, but is a form of "crecer" (which appears in the training set), not of "creer" (which does not). This demonstrates the problems resulting from a lack of lemma diversity during training. PT produces a combination of several of the forms in the input subset: subjunctive forms beginning with "crey" and "creemos" V;IND;PRS;1;PL. Again, MED+PT generates the correct target.

Overall, this analysis confirms that MED learns relationships between paradigm cells, while paradigm transduction adds knowledge about the idiosyncracies of a partial test paradigm.

5.3 Comparison to Multi-Source Models

In this section, we explicitly compare our approach to neural multi-source models for morphological generation.

Following Kann et al. (2017a), we employ attention-based RNN encoder-decoder networks with two or four input sources. The input to a

	SET1				SET2				SET3			
	1	2	4	+PT	1	2	4	+PT	1	2	4	+PT
dutch	.00	.00	.00	**.49**	.04	.01	.00	**.78**	.43	.65	.72	**.87**
german	.00	.00	.00	**.65**	.00	.00	.01	**.75**	.44	.42	.59	**.88**
icelandic	.00	.00	.00	**.41**	.03	.02	.02	**.50**	.24	.33	.35	**.77**
spanish	.00	.00	.00	**.92**	.03	.09	.09	**.98**	.59	.63	.83	**.99**
welsh	.00	.00	.00	**.91**	.05	.14	.15	**.97**	.35	.53	.70	**.99**

Table 3: MED accuracy on five randomly selected languages with 1, 2, and 4 sources and combined with paradigm transduction ("+PT"). Best results in bold.

multi-source model is the concatenation of all sources and corresponding tags. During training, we randomly sample (with repetition) one or three additional forms from the paradigm of each example. At test time, we sample the additional forms from the given partial paradigm; without repetition first, but repeating if not enough inflected forms are available. For autoencoding examples in the training data, we simply concatenate two or four copies of the source and the autoencoding tag. We randomly select five languages for this experiment.

Table 3 shows that, for SET3, four sources (column header "4") are generally better than two sources ("2"), which in turn are better than one source ("1"); thus, as expected, making additional sources available in training improves results. We attribute one exception (German accuracy is .4391 for "1" and .4179 for "2") to the noisiness of the problem—training sets in terms of number of paradigms are relatively small, even for SET3.

The improvements we see for SET3 are large. This suggests that using more than four sources would further improve results and perhaps reach the level of performance of MED+PT, at the cost of a long training time. However, for SET1 and SET2, there is no consistent improvement from 1 to 2 to 4 sources. While it is possible that further optimization could improve the best multi-source result given in Table 3, the gap to MED+PT is very large, and the improvement from 2 to 4 is small. This indicates that multi-source methods cannot compete with transductive learning for SET1 and SET2.

5.4 Qualitative Analysis of SHIP

For a qualitative analysis of SHIP, we look at the sources it selects for French verbs on the development set; the complete diagram is shown in Figure 6. For most verbs, future and conditional can be predicted from COND;1;PL (e.g., "finirions"), and indicative present, indicative imparfait and subjunctive present from IND;PRS;3;PL (e.g., "finissent").

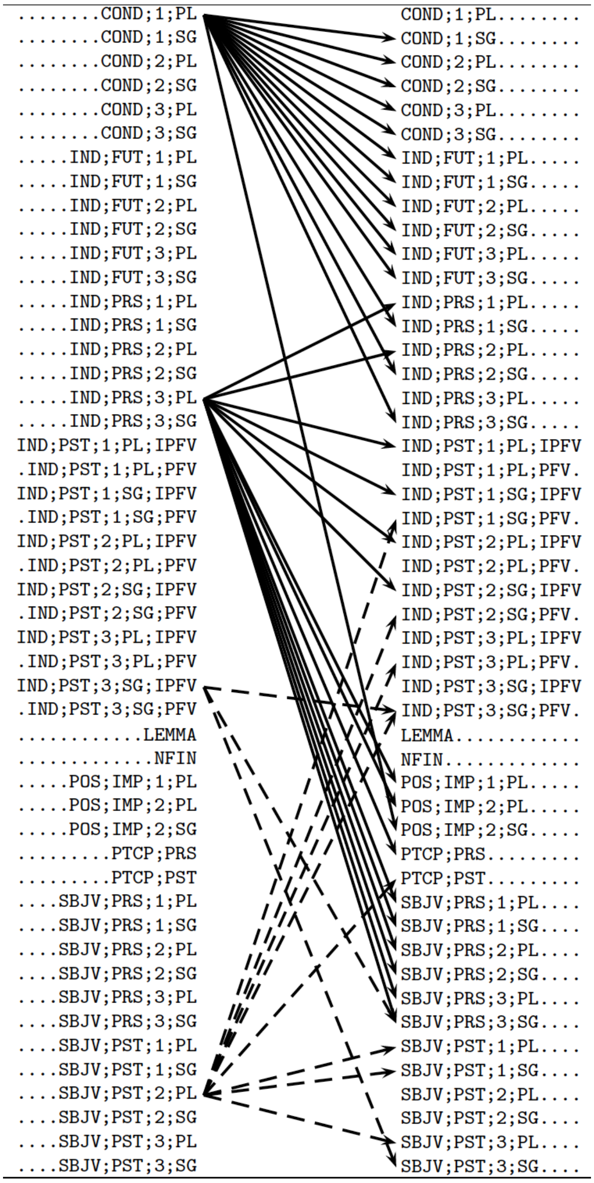

Figure 6: Right: output set target to be generated. Left: input set source selected by SHIP. Arrows for the two most frequently selected sources are solid, arrows for the two least frequently selected sources are dashed.

In case of ties, SHIP selects the alphabetically first tag; this explains why COND;1;PL gets preference over IND;PRS;3;PL for indicative present singular. These two forms represent two of the *principal* *parts* of French conjugation, the infinitive (almost always derivable from COND;1;PL) and the stem that is used for plural indicative, imparfait, and other paradigm cells—which is sometimes not derivable from the infinitive as is the case for "finir". In comparison, IND;PST;3;SG;IPFV and SBJV;PST;2;PL are less reliable sources. But they are still reasonably accurate if no better alternative is available; consider the following SBJV;PST;2;PL → IND;PST;1;SG;PFV generations: "parlassiez" ↦ "parlai", "finissiez" ↦ "finis", "missiez" ↦ "mis", "prissiez" ↦ "pris".

We thus conclude that SHIP indeed learns to select appropriate source forms.

6 Related Work

Morphological generation. In the last two years, most work on paradigm completion has been done in the context of the SIGMORPHON 2016 and the CoNLL–SIGMORPHON 2017 shared tasks (Cotterell et al., 2016, 2017a). Due to the success of neural seq2seq models in 2016 (Kann and Schütze, 2016b; Aharoni et al., 2016), systems developed for the 2017 edition were mostly neural (Makarov et al., 2017; Bergmanis et al., 2017; Zhou and Neubig, 2017). Besides the shared task systems, Kann and Schütze (2017) presented a paradigm completion model for a multi-source setting that made use of an attention mechanism to decide which input form to attend to at each time step. They used randomly chosen, independent pairs of source and target forms for training. This differs crucially from the setting we consider in that no complete paradigms were available in their training sets. Only Cotterell et al. (2017b) addressed essentially the same task we do, but they only considered the high-resource setting: their models were trained on hundreds of complete paradigms. The experiments reported in §5.3 empirically confirm that inductive-only models perform poorly in our setting.

Several ways to employ neural models for morphological generation with limited data have been proposed, e.g., semi-supervised training (Zhou and Neubig, 2017; Kann and Schütze, 2017) or simultaneous training on multiple languages (Kann et al., 2017b). The total number of sources in the training set in some of our settings may be comparable to this earlier work, but our training sets are less diverse since many forms come from the same paradigm. We argue in §1 that the number of paradigms (not the number of sources) measures the effective size of the training set.

Other important work on morphological generation—neural and non-neural—includes Dreyer et al. (2008); Durrett and DeNero (2013); Hulden et al. (2014); Nicolai et al. (2015); Faruqui et al. (2016); Yin et al. (2016).

Seq2seq models in NLP. Even though neural seq2seq models were originally designed for machine translation (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015), their application has not stayed limited to this area. Similar architectures have been successfully applied to many seq2seq tasks in NLP, e.g., syntactic parsing (Vinyals et al., 2015), language correction (Xie et al., 2016), normalization of historical texts (Bollmann et al., 2017), or text simplification (Nisioi et al., 2017). Transductive inference is similar to domain adaptation, e.g., in machine translation (Luong and Manning, 2015). One difference is that training set and test set can hardly be called different domains in paradigm completion. Another difference is that explicit structured labels (the morphological tags of the forms in the input subset) are available at test time in paradigm completion.

7 Conclusion

We presented two new methods for minimal-resource paradigm completion: paradigm transduction and SHIP. Paradigm transduction learns general inflection rules through standard inductive training and idiosyncracies of a test paradigm through transduction. We showed that paradigm transduction effectively mitigates the problem of overfitting due to a lack of diversity in the training data. SHIP is a robust non-neural method that identifies a single reliable source for generating a target. In the minimal-resource setting, this is an effective alternative to learning how to combine evidence from multiple sources. Considering the average over all languages of a 52-language benchmark dataset, we outperform the previous state of the art by at least 7.07%, and up to 9.71% absolute accuracy.

Acknowledgments

We would like to thank Samuel Bowman, Ryan Cotterell, Nikita Nangia, and Alex Warstadt for their feedback on this work.

References

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The biu-mit systems for the sigmorphon 2016 shared task for morphological reinflection. In *SIGMORPHON*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *CoNLL–SIGMORPHON*.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *ACL*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *SSST*.

Grzegorz Chrupała, Georgiana Dinu, and Josef Van Genabith. 2008. Learning morphology with morfette. In *LREC*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017a. The CoNLL–SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *CoNLL–SIGMORPHON*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *SIGMORPHON*.

Ryan Cotterell, John Sylak-Glassman, and Christo Kirov. 2017b. Neural graphical models over strings for principal parts morphological paradigm completion. In *EACL*.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *EMNLP*.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *NAACL-HLT*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *NAACL-HLT*.

Raphael Finkel and Gregory Stump. 2007. Principal parts and degrees of paradigmatic transparency. Technical report, Department of Computer Science, University of Kentucky, Lexington, KY.

Dan Gusfield. 1997. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge university press.

Mans Hulden, Markus Forsberg, and Malin Ahlberg. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *EACL*.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017a. Neural multi-source morphological reinflection. In *EACL*.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017b. One-shot neural cross-lingual transfer for paradigm completion. In *ACL*.

Katharina Kann and Hinrich Schütze. 2016a. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *SIGMORPHON*.

Katharina Kann and Hinrich Schütze. 2016b. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*.

Katharina Kann and Hinrich Schütze. 2017. The LMU system for the CoNLL–SIGMORPHON 2017 shared task on universal morphological reinflection. In *CoNLL–SIGMORPHON*.

Katharina Kann and Hinrich Schütze. 2017. Unlabeled data for morphological generation with character-based sequence-to-sequence models. In *SCLeM*.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian J Mielke, Arya McCarthy, Sandra Kübler, et al. 2018. Unimorph 2.0: Universal morphology. In *LREC*.

Ling Liu and Lingshuang Jack Mao. 2016. Morphological reinflection with conditional random fields and unsupervised features. In *SIGMORPHON*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *IWSLT*.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *CoNLL–SIGMORPHON*.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *NAACL-HLT*.

Sergiu Nisioi, Sanja Stajner, Simone Paolo Ponzetto, Paolo Rosso, and Heiner Stuckenschmidt. 2017. Exploring neural text simplification models. In *ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

John Sylak-Glassman, Christo Kirov, and David Yarowsky. 2016. Remote elicitation of inflectional paradigms to seed morphological analysis in low-resource languages. In *LREC*.

Aleš Tamchyna, Marion Weller-Di Marco, and Alexander Fraser. 2017. Modeling target-side inflection in neural machine translation. In *WMT*.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *TACL*, 4:259–272.

Matthew D Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv:1212.5701*.

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *ACL*.

Implicational Universals in Stochastic Constraint-Based Phonology

Giorgio Magri
CNRS / University of Paris 8
`magrigrg@gmail.com`

Abstract

This paper focuses on the most basic implicational universals in phonological theory, called *T-orders* after Anttila and Andrus (2006). It shows that the T-orders predicted by stochastic and categorical Optimality Theory coincide. Analogously, the T-orders predicted by stochastic and categorical Harmonic Grammar coincide. In other words, these stochastic constraint-based frameworks do not tamper with the typological structure induced by the corresponding categorical frameworks.

1 Introduction

Phonology has traditionally focused on *alternations* revealed by paradigms such as the German final devoicing examples [baːt]/[bɛːdə] (‘bath-SG/PL’) and [tsuːk]/[tsyːgə] (‘train-SG/PL’). These alternations are usually modeled through *phonological grammars* which map from *underlying representations* (URs) to *surface representations* (SRs) (Chomsky and Halle, 1968). Constraint-based implementations of this combinatorial phonological theory include *Optimality Theory* (OT; Prince and Smolensky, 1997, 2004) and *Harmonic Grammar* (HG; Legendre et al., 1990; Smolensky and Legendre, 2006), reviewed below in sections 4 and 5.

More recently, phonology has extended its empirical coverage from categorical alternations to patterns of phonologically conditioned variation and gradient phonological (or phonotactic) judgements (see for instance Anttila, 2012 and Coetzee and Pater, 2011). This extension of the empirical coverage has required a corresponding extension of the theoretical framework. A phonological grammar cannot be construed anymore as a categorical function from URs to SRs. Instead, it must be construed as a function from URs to probability distributions over the entire set of SRs. Constraint-based implementations of this stochastic theory include *partial order OT* (Anttila, 1997b), *stochastic OT* (SOT; Boersma, 1997, 1998), and *stochastic HG* (SHG; Boersma and Pater, 2016)[1], recalled below in sections 4 and 5. Another framework explored in the recent literature on probabilistic constraint-based phonology is *MaxEnt* (ME; Goldwater and Johnson, 2003; Hayes and Wilson, 2008). Its T-orders are discussed in a companion paper (Anttila and Magri, 2018).

How can we investigate and understand the typological structure encoded by a probabilistic phonological framework? In the case of a categorical framework such as OT or HG, the predicted typological structure can be investigated directly by exhaustively listing all the grammars predicted for certain constraint and candidate sets. That is possible because the predicted typology of grammars is usually *finite*. The situation is rather different for probabilistic frameworks: the predicted typology always consists of an *infinite* number of probability distributions which therefore cannot be exhaustively listed and directly inspected. A more indirect strategy is needed to chart the predicted typological structure.

A natural indirect strategy that gets around the problem raised by an infinite typology is to enumerate, not the individual languages in the typology, but the set of implicational universals predicted by the typology. An *implicational universal* is an implication $P \xrightarrow{\mathfrak{T}} \widehat{P}$ which holds of a given typology $\mathfrak{T}$ whenever *every* language in the typology that satisfies the antecedent property P also satisfies the consequent property $\widehat{P}$ (Greenberg, 1963). Since implicational universals take

---

[1] Boersma and Pater (2016) actually use the term “noisy HG” instead of “stochastic HG”. We prefer “stochastic HG” to stress the analogy with Boersma’s earlier framework of stochastic OT. Furthermore, we prefer to use “stochastic” to describe a property of the framework, reserving “noisy” to describe a property of the learning scenario (as opposed to noise-free). Hayes (2017) discusses further stochastic variants of categorical HG.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3265–3274
Brussels, Belgium, October 31 - November 4, 2018.

into account every language in the typology, they chart the boundaries and measure the richness of the typological structure predicted by $\mathfrak{T}$.

Which antecedent and consequent properties P and $\widehat{P}$ should we focus on? To start from the simplest case, let us consider a typology $\mathfrak{T}$ of *categorical* phonological grammars, construed traditionally as mappings from URs to SRs. Within this categorical framework, the simplest, most basic, most atomic antecedent property P is the property of mapping a certain specific UR x to a certain specific SR y. Analogously, the simplest consequent property $\widehat{P}$ is the property of mapping a certain specific UR $\widehat{\mathsf{x}}$ to a certain specific SR $\widehat{\mathsf{y}}$. We thus focus on the following class of implications:

Definition 1 *The implicational universal* $(\mathsf{x},\mathsf{y}) \overset{\mathfrak{T}}{\rightarrow} (\widehat{\mathsf{x}},\widehat{\mathsf{y}})$ *holds relative to a categorical typology* $\mathfrak{T}$ *provided each grammar in* $\mathfrak{T}$ *which succeeds at the antecedent mapping (i.e., it maps the UR* x *to the SR* y*), also succeeds at the consequent mapping (i.e., it maps the UR* $\widehat{\mathsf{x}}$ *to the SR* $\widehat{\mathsf{y}}$*).* □

The relation $\overset{\mathfrak{T}}{\rightarrow}$ thus defined over mappings is a partial order (under mild additional assumptions). It is called the *T-order* induced by the typology $\mathfrak{T}$ (Anttila and Andrus, 2006). For example, any dialect of English that deletes t/d at the end of a coda cluster before a vowel also deletes it before a consonant (Guy, 1991; Kiparsky, 1993; Coetzee, 2004). The implication (/cost.us/, [cos.us]) → (/cost.me/, [cos.me]) thus holds relative to the typology $\mathfrak{T}$ of English dialects.

Implicational universals can also be statistical. For instance, in dialects of English where t/d deletion applies variably, deletion has been found to be more frequent before consonants than before vowels. To model these frequency effects, we need to consider a typology $\mathfrak{T}$ of *probabilistic* phonological grammars, construed as functions from URs to probability distributions over SRs. We propose to extend the notion of T-orders from the categorical to the probabilistic setting as follows:

Definition 2 *The implicational universal* $(\mathsf{x},\mathsf{y}) \overset{\mathfrak{T}}{\rightarrow} (\widehat{\mathsf{x}},\widehat{\mathsf{y}})$ *holds relative to a probabilistic typology* $\mathfrak{T}$ *provided each grammar in* $\mathfrak{T}$ *assigns a probability to the consequent mapping* $(\widehat{\mathsf{x}},\widehat{\mathsf{y}})$ *which is at least as large as the probability it assigns to the antecedent mapping* (x,y)*.* □

To illustrate, the implication (/cost.us/, [cos.us]) → (/cost.me/, [cos.me]) also holds relative to the typology $\mathfrak{T}$ of English dialects with variable deletion because the probability of the consequent (/cost.me/, [cos.me]) (i.e., the frequency of deletion before a consonant) in any dialect is at least as large as the probability of the antecedent (/cost.us/, [cos.us]) (i.e., the frequency of deletion before a vowel).

The original categorical definition 1 of T-orders is a special case of the probabilistic definition 2. In fact, suppose that a categorical grammar succeeds on the antecedent mapping (x,y). That grammar construed probabilistically thus assigns probability 1 to the antecedent mapping. Definition 2 then requires that grammar to also assign probability 1 to the consequent mapping $(\widehat{\mathsf{x}},\widehat{\mathsf{y}})$. In other words, the grammar construed categorically succeeds on the consequent mapping, as required by the original definition 1 of categorical T-orders.

T-orders are defined at the level of mappings from URs to SRs. They thus allow for cross-framework comparisons, even bridging across categorical and probabilistic frameworks. This paper (together with the companion Anttila and Magri 2018) thus uses T-orders to compare the probabilistic implementations of constraint-based phonology with the original categorical implementations.

The main result reported in this paper is that the T-orders predicted by stochastic OT (and by partial order OT) coincide with those predicted by categorical OT, no matter what the candidate and constraint sets look like, as shown in section 4. Analogously, the T-orders predicted by stochastic HG coincide with those predicted by categorical HG, as shown in section 5. In other words, these stochastic frameworks do not tamper with the typological structure induced by the original categorical frameworks, at least when that structure is measured in terms of T-orders. These specific results about OT and HG are derived as a special case of a more general result on stochastic typologies, developed in sections 2 and 3.

As discussed in a companion paper (Anttila and Magri, 2018), the situation is very different for ME. Both ME and stochastic HG can be construed as probabilistic variants of categorical HG. Stochastic and categorical HG share the same T-orders. The ME T-orders instead obey a rather different underlying convex geometry and turn out to be much sparser. In other words, ME yields a much richer probabilistic extension of categorical HG than stochastic HG does. Section 6 concludes the paper by discussing these results in the context of the recent literature on probabilistic constraint-based phonology.

2 Categorical and stochastic phonology

We assume a relation Gen which pairs each UR x with a set $Gen(\mathsf{x})$ of *candidate* SRs. As recalled above, a *categorical* phonological grammar G takes a UR x and selects a corresponding SR $\mathsf{y} = G(\mathsf{x})$ from the candidate set $Gen(\mathsf{x})$. A stochastic phonological grammar G instead takes a UR x and returns a probability distribution $G(\cdot\,|\,\mathsf{x})$ over $Gen(\mathsf{x})$ which assigns a probability $G(\mathsf{y}|\,\mathsf{x})$ to each candidate SR y in $Gen(\mathsf{x})$. This section illustrates a general method to leverage a given typology $\mathfrak{T}$ of categorical grammars into a typology of stochastic grammars. Sections 4 and 5 will then show that various stochastic frameworks in the recent constraint-based literature (such as partial order OT, stochastic OT, and stochastic HG) all fit within this general scheme.

Following common practice in constraint-based phonology, we assume that the categorical typology $\mathfrak{T}$ only contains a finite number of grammars.[2] We consider a probability mass function p over $\mathfrak{T}$. Thus, p assigns to each categorical grammar G in $\mathfrak{T}$ a nonnegative probability mass $p(G) \geq 0$ and these masses sum up to 1, namely $\sum_{G\in\mathfrak{T}} p(G) = 1$. We can then define the stochastic grammar G_p corresponding to the probability mass function p as the function which takes a UR x and returns the probability distribution $G_p(\cdot\,|\,\mathsf{x})$ over the candidate set $Gen(\mathsf{x})$ defined as in (1). It says that the probability $G_p(\mathsf{y}\,|\,\mathsf{x})$ that the UR x is mapped to the SR y is he probability mass allocated by p to the region $\{G \in \mathfrak{T}\,|\,G(\mathsf{x}) = \mathsf{y}\}$ of the typology $\mathfrak{T}$ consisting of those categorical grammars which succeed on the mapping (x, y).

$$G_p(\mathsf{y}\,|\,\mathsf{x}) = \sum_{\{G\in\mathfrak{T}\,|\,G(\mathsf{x})=\mathsf{y}\}} p(G) \qquad (1)$$

We assume next that each categorical grammar in the typology $\mathfrak{T}$ returns a *unique* SR y for each UR x.[3] This assumption suffices to ensure that G_p is indeed a probability distribution, namely that the sum of the probabilities $G_p(\mathsf{y}\,|\,\mathsf{x})$ over the candidates y in $Gen(\mathsf{x})$ is equal to 1, as shown in (2).

$$\sum_{\mathsf{y}\in Gen(\mathsf{x})} G_p(\mathsf{y}\,|\,\mathsf{x}) \overset{(a)}{=} \sum_{\mathsf{y}\in Gen(\mathsf{x})} \sum_{\{G\in\mathfrak{T}\,|\,G(\mathsf{x})=\mathsf{y}\}} p(G) \overset{(b)}{=} \sum_{G\in\mathfrak{T}} p(G) \overset{(c)}{=} 1 \qquad (2)$$

In step (2a), we have used the definition (1) of $G_p(\mathsf{y}\,|\,\mathsf{x})$. In step (2b), we have used the fact that every grammar in $\mathfrak{T}$ maps x to a unique SR y, so that the sets $\{G \in \mathfrak{T}\,|\,G(\mathsf{x}) = \mathsf{y}\}$ partition the typology $\mathfrak{T}$ into disjoint sets as y spans the candidate set $Gen(\mathsf{x})$. In step (2c), we have used the fact that p is a probability mass function over $\mathfrak{T}$ and thus adds up to 1.

A family $\mathcal{P}$ of probability mass functions $p_1, p_2, \ldots$ over the finite categorical typology $\mathfrak{T}$ thus induces a typology $\{G_{p_1}, G_{p_2}, \ldots\}$ of stochastic grammars. It is called the *stochastic* typology corresponding to the categorical typology $\mathfrak{T}$ and the probability family $\mathcal{P}$, and it is denoted by $\mathfrak{T}_{\mathcal{P}}$. We denote by $\xrightarrow{\mathfrak{T}}$ the T-order relative to the categorical typology $\mathfrak{T}$ in the sense of definition 1 and by $\xrightarrow{\mathfrak{T}_{\mathcal{P}}}$ the T-order relative to the stochastic typology $\mathfrak{T}_{\mathcal{P}}$ in the sense of definition 2. We want to investigate the relationship between these categorical and stochastic T-orders.

3 Relationship between categorical and stochastic T-orders

Let us suppose that the implication $(\mathsf{x}, \mathsf{y}) \overset{\mathfrak{T}}{\rightarrow} (\widehat{\mathsf{x}}, \widehat{\mathsf{y}})$ holds between an antecedent mapping (x, y) and a consequent mapping $(\widehat{\mathsf{x}}, \widehat{\mathsf{y}})$ relative to a categorical typology $\mathfrak{T}$. By definition 1, this means that every categorical grammar G in the typology $\mathfrak{T}$ that maps the antecedent UR x to the antecedent SR y (namely, $G(\mathsf{x}) = \mathsf{y}$) also maps the consequent UR $\widehat{\mathsf{x}}$ to the consequent SR $\widehat{\mathsf{y}}$ (namely, $G(\widehat{\mathsf{x}}) = \widehat{\mathsf{y}}$), yielding the inclusion (3).

$$\underset{\substack{\text{grammars consistent}\\ \text{with the antecedent}\\ \text{mapping}}}{\{G \in \mathfrak{T}\,|\,G(\mathsf{x})=\mathsf{y}\}} \subseteq \underset{\substack{\text{grammars consistent}\\ \text{with the consequent}\\ \text{mapping}}}{\{G \in \mathfrak{T}\,|\,G(\widehat{\mathsf{x}})=\widehat{y}\}} \qquad (3)$$

By (1), this inclusion (3) entails that the probability assigned by G_p to the consequent mapping $(\widehat{\mathsf{x}}, \widehat{\mathsf{y}})$ is at least as large as the probability assigned to the antecedent mapping (x, y), as stated in (4). This entailment follows from the sheer fact that

---

[2] This assumption always holds in OT, as the number of constraint rankings is finite. It might fail in HG, but only in rather pathological situations which do not seem germane to natural language phonology.

[3] Suppose instead that a categorical grammar were to return two different SRs y_1 and y_2 for some UR x. How should we interpret such a scenario? Plausibly, we should interpret the two SRs y_1 and y_2 as free variants with equal probability of 0.5 (while all other candidates have probability 0). But this means that our grammar is stochastic, not categorical.

probabilities are monotonic relative to set inclusion. The entailment from the inclusion (3) to the inequality (4) thus holds under no assumptions whatsoever on the probability mass function p used to define the stochastic grammar G_p.

$$\underset{\substack{\text{probability of the}\\ \text{antecedent mapping}}}{G_p(\mathsf{y}\,|\,\mathsf{x})} \;\leq\; \underset{\substack{\text{probability of the}\\ \text{consequent mapping}}}{G_p(\widehat{\mathsf{y}}\,|\,\widehat{\mathsf{x}})} \tag{4}$$

The latter inequality (4) finally says that the implication $(\mathsf{x},\mathsf{y}) \xrightarrow{\mathfrak{T}_{\mathcal{P}}} (\widehat{\mathsf{x}},\widehat{\mathsf{y}})$ holds also relative to the stochastic typology $\mathfrak{T}_{\mathcal{P}}$ in the sense of definition 2. In conclusion, a categorical T-order always entails the corresponding stochastic T-order, no matter the shape of the family $\mathcal{P}$ of probability mass functions used to derive the stochastic typology $\mathfrak{T}_{\mathcal{P}}$ from the categorical typology $\mathfrak{T}$.

We now turn to the reverse entailment. Suppose that an implication $(\mathsf{x},\mathsf{y}) \xrightarrow{\mathfrak{T}_{\mathcal{P}}} (\widehat{\mathsf{x}},\widehat{\mathsf{y}})$ holds between an antecedent mapping (x,y) and a consequent mapping $(\widehat{\mathsf{x}},\widehat{\mathsf{y}})$ relative to the stochastic typology $\mathfrak{T}_{\mathcal{P}}$. By definition 2, this means in turn that the inequality (4) holds between the probabilities $G_p(\mathsf{y}\,|\,\mathsf{x})$ and $G_p(\widehat{\mathsf{y}}\,|\,\widehat{\mathsf{x}})$ of the antecedent and the consequent mappings relative to any probability mass function p in the family $\mathcal{P}$. Suppose by contradiction that the corresponding implication $(\mathsf{x},\mathsf{y}) \xrightarrow{\mathfrak{T}} (\widehat{\mathsf{x}},\widehat{\mathsf{y}})$ relative to the original categorical typology $\mathfrak{T}$ instead fails. By definition 1, this means that the set inclusion (3) fails because there exists some grammar G_0 with the properties in (5): G_0 succeeds on the antecedent mapping, namely it maps x to y; but G_0 fails on the consequent mapping, namely it maps $\widehat{\mathsf{x}}$ to some loser candidate $\widehat{\mathsf{z}}$ different from the intended winner candidate $\widehat{\mathsf{y}}$.

$$G_0(\mathsf{x}) = \mathsf{y}, \qquad G_0(\widehat{\mathsf{x}}) = \widehat{\mathsf{z}} \neq \widehat{\mathsf{y}} \tag{5}$$

We would like to derive a contradiction from the assumption (4) that the stochastic implication $(\mathsf{x},\mathsf{y}) \xrightarrow{\mathfrak{T}_{\mathcal{P}}} (\widehat{\mathsf{x}},\widehat{\mathsf{y}})$ holds and the assumption (5) that the categorical implication $(\mathsf{x},\mathsf{y}) \xrightarrow{\mathfrak{T}} (\widehat{\mathsf{x}},\widehat{\mathsf{y}})$ fails.

Yet, no contradiction arises in the general case. Indeed, suppose that the probability mass functions in the family $\mathcal{P}$ all happen to assign zero (or tiny) probability mass to this grammar G_0 which flouts the categorical implication because of (5). This problematic grammar G_0 thus bears no (or only a tiny) effect on the total probabilities $G_p(\mathsf{y}\,|\,\mathsf{x})$ and $G_p(\widehat{\mathsf{y}}\,|\,\widehat{\mathsf{x}})$ of the two mappings (x,y) and $(\widehat{\mathsf{x}},\widehat{\mathsf{y}})$. The probability inequality (4) is therefore not necessarily compromised by the offensive behavior (5) of G_0, as long as the other grammars in the typology comply.

In order to derive a contradiction from these two conditions (4) and (5), we need to make some assumptions on the family $\mathcal{P}$ of probability mass functions. Indeed, the problem just discussed arises when *every* probability mass function p in $\mathcal{P}$ assigns zero (or tiny) probability to the problematic grammar G_0. We need to rule out this scenario. We propose to achieve that through the assumption that the family $\mathcal{P}$ satisfies the following

Definition 3 *The family $\mathcal{P}$ of probability mass functions over the finite categorical typology $\mathfrak{T}$ is* sufficiently rich *in the sense that for every categorical grammar G in $\mathfrak{T}$ and for any two URs x and $\widehat{\mathsf{x}}$, the following inequalities*

$$G_p(G(\mathsf{x})|\,\mathsf{x}) > 1/2, \quad G_p(G(\widehat{\mathsf{x}})|\,\widehat{\mathsf{x}}) > 1/2 \tag{6}$$

hold for some probability mass function p in $\mathcal{P}$. □

Here is the intuition behind this definition. Suppose that for every categorical grammar G, the family $\mathcal{P}$ contains a probability mass function p which assigns all the probability mass to that grammar G. By (1), the corresponding stochastic grammar G_p assigns probability 1 to the mappings enforced by G, as stated in (7).

$$G_p(G(\mathsf{x})|\,\mathsf{x}) = 1 \text{ for every UR } \mathsf{x} \tag{7}$$

In other words, the stochastic grammar G_p "coincides" with the categorical grammar G and the stochastic typology $\mathfrak{T}_{\mathcal{P}}$ thus "contains" or "extends" the original categorical typology $\mathfrak{T}$. In this special case, we obviously expect the stochastic implication $(\mathsf{x},\mathsf{y}) \xrightarrow{\mathfrak{T}_{\mathcal{P}}} (\widehat{\mathsf{x}},\widehat{\mathsf{y}})$ to entail the categorical implication $(\mathsf{x},\mathsf{y}) \xrightarrow{\mathfrak{T}} (\widehat{\mathsf{x}},\widehat{\mathsf{y}})$, as desired.

Condition (6) required by definition 3 is a weaker version of the latter condition (7). First, it is weaker because the requirement $G_p(G(\mathsf{x})|\,\mathsf{x}) = 1$ is replaced with the weaker requirement $G_p(G(\mathsf{x})|\,\mathsf{x}) > 1/2$: the probability assigned to the mappings enforced by G needs not be 1, as long as it is large enough, namely larger than 1/2. Second, this requirement $G_p(G(\mathsf{x})|\mathsf{x}) > 1/2$ needs not be satisfied by a unique mass p for all URs: it suffices to look at just two URs at the time.

If the family $\mathcal{P}$ is sufficiently rich in the sense of definition 3, the two conditions (4) and (5) are indeed contradictory. In fact, condition (5) now ensures that $\mathcal{P}$ contains a probability mass function

p_0 such that the corresponding stochastic grammar G_{p_0} maps x to y with probability larger than 1/2 and it maps $\widehat{\mathsf{x}}$ to $\widehat{\mathsf{z}}$ with probability larger than 1/2. The latter fact means in turn that G_{p_0} maps $\widehat{\mathsf{x}}$ to $\widehat{\mathsf{y}}$ with probability smaller than 1/2, because the probabilities of the various candidates $\widehat{\mathsf{y}}, \widehat{\mathsf{z}}, \dots$ in $Gen(\widehat{\mathsf{x}})$ must add up to 1. In conclusion, we have obtained $G_{p_0}(\mathsf{y}\,|\,\mathsf{x}) > 1/2$ and $G_{p_0}(\widehat{\mathsf{y}}\,|\,\widehat{\mathsf{x}}) < 1/2$, in blatant contradiction of (4).

The preceding reasoning is summarized in the following proposition 1, which says that the T-order relative to a categorical typology $\mathfrak{T}$ and the T-order relative to the corresponding stochastic typology $\mathfrak{T}_{\mathcal{P}}$ coincide, no matter what the family $\mathcal{P}$ of probability mass functions looks like, as long as it is sufficiently rich, in the sense of definition 3. Identity of T-orders holds even when the family $\mathcal{P}$ is infinite, so that the stochastic typology $\mathfrak{T}_{\mathcal{P}}$ contains an infinite number of stochastic grammars, while the categorical typology $\mathfrak{T}$ contains only a finite number of grammars.

Proposition 1 *Consider a finite typology $\mathfrak{T}$ of categorical grammars and a family $\mathcal{P}$ of probability mass functions on $\mathfrak{T}$. Let $\mathfrak{T}_{\mathcal{P}}$ be the typology of the corresponding stochastic grammars, defined through (1). If $\mathcal{P}$ is sufficiently rich in the sense of definition 3, the T-order $\xrightarrow{\mathfrak{T}}$ relative to the categorical typology $\mathfrak{T}$ and the T-order $\xrightarrow{\mathfrak{T}_{\mathcal{P}}}$ relative to the stochastic typology $\mathfrak{T}_{\mathcal{P}}$ coincide.* □

In the rest of the paper, we apply this result to various categorical and stochastic frameworks for constraint-based phonology.

4 Categorial OT, partial order OT, and stochastic OT induce the same T-orders

In this section, we focus on categorical and stochastic OT. We assume a set of n constraints $C_1, \dots, C_k, \dots, C_n$ and some candidacy relation Gen. We recall that a constraint C_k *prefers* a mapping (x, y) to another mapping (x, z) provided C_k assigns less violations to the former than to the latter, namely $C_k(\mathsf{x}, \mathsf{y}) < C_k(\mathsf{x}, \mathsf{z})$. A *constraint ranking* is an arbitrary linear order $\gg$ over the constraint set. A constraint ranking $\gg$ *prefers* a mapping (x, y) to another mapping (x, z) provided the highest $\gg$-ranked constraint which distinguishes between the two mappings (x, y) and (x, z) prefers (x, y). The *categorical OT grammar* corresponding to a ranking $\gg$ maps a UR x to that SR y such that $\gg$ prefers the mapping (x, y) to the mapping (x, z) corresponding to any other candidate z in $Gen(\mathsf{x})$ (Prince and Smolensky, 2004). We denote by $\overset{\mathrm{OT}}{\to}$ the T-order corresponding to the typology $\mathfrak{T}$ of the categorical OT grammars corresponding to all constraint rankings, in the sense of definition 2.

To illustrate, consider the following three constraints (from Kiparsky, 1993) for the process of t/d deletion mentioned in section 1: C_1 = SYLLABLEWELLFORMEDNESS (SWF) penalizes codas and tautosyllabic consonant clusters; C_2 = ALIGN penalizes resyllabification across word boundaries; and C_3 = MAX penalizes segment deletion. Suppose that the UR /cost us/ comes with the three candidate SRs [cost.us] (faithful), [cos.us] (with deletion), and [cos.tus] (with resyllabification). Analogously, suppose that the UR /cost me/ comes with the three candidate SRs [cost.me], [cos.me], and [cos.tme]. It is easy to verify that the implication (/cost.us/, [cos.us]) $\overset{\mathrm{OT}}{\to}$ (/cost.me/, [cos.me]) holds relative to the OT typology generated by constraints C_1, C_2, C_3 in the sense of definition 1: every ranking of the three constraints which succeeds on the antecedent mapping (/cost.us/, [cos.us]) also succeeds on the consequent mapping (/cost.me/, [cos.me]). In other words, t/d deletion before a vowel entails deletion before a consonant.

We now turn to the stochastic counterpart of this categorical framework. A *ranking vector* $\boldsymbol{\theta} = (\theta_1, \dots, \theta_k, \dots, \theta_n) \in \mathbb{R}^n$ assigns a numerical *ranking value* θ_k to each constraint C_k. The *stochastic ranking vector* $\boldsymbol{\theta} + \boldsymbol{\epsilon} = (\theta_1 + \epsilon_1, \dots, \theta_n + \epsilon_n)$ is obtained by adding to the ranking values $\theta_1, \dots, \theta_n$ some numbers $\epsilon_1, \dots, \epsilon_n$ sampled independently from each other according to some distribution $\mathcal{D}$ on $\mathbb{R}$. If the distribution $\mathcal{D}$ is continuous, the probability that two stochastic ranking values $\theta_h + \epsilon_h$ and $\theta_k + \epsilon_k$ coincide is equal to zero. The stochastic ranking vector $\boldsymbol{\theta} + \boldsymbol{\epsilon}$ thus describes the unique ranking $\gg_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}$ which respects the relative size of the stochastic ranking values: a constraint C_h is ranked above a constraint C_k according to $\gg_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}$ (namely, $C_h \gg_{\boldsymbol{\theta}+\boldsymbol{\epsilon}} C_k$) if and only if the stochastic ranking value of the former is larger than that of the latter (namely, $\theta_h + \epsilon_h > \theta_k + \epsilon_k$). A ranking vector $\boldsymbol{\theta}$ thus induces the probability mass function $p^{\mathcal{D}}_{\boldsymbol{\theta}}$ defined in (8) over the categorical OT typology $\mathfrak{T}$. Obviously, this definition yields a probability mass, namely the sum of the masses $p^{\mathcal{D}}_{\boldsymbol{\theta}}(G)$ over all the

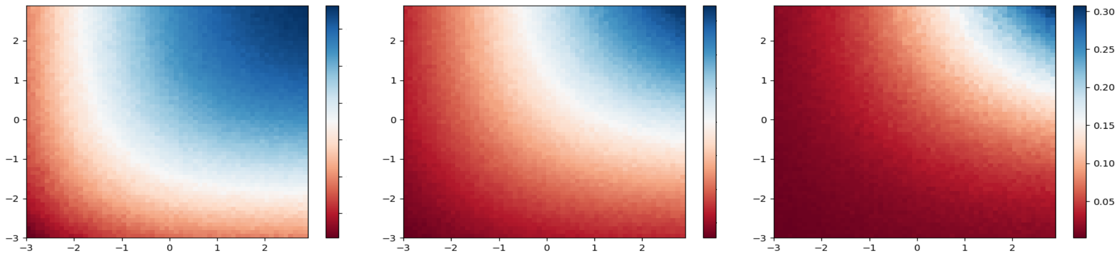

(a) (/cost.us/, [cos.us]), $\theta_3 = -3$ (b) (/cost.us/, [cos.us]), $\theta_3 = 0$ (c) (/cost.us/, [cos.us]), $\theta_3 = 3$

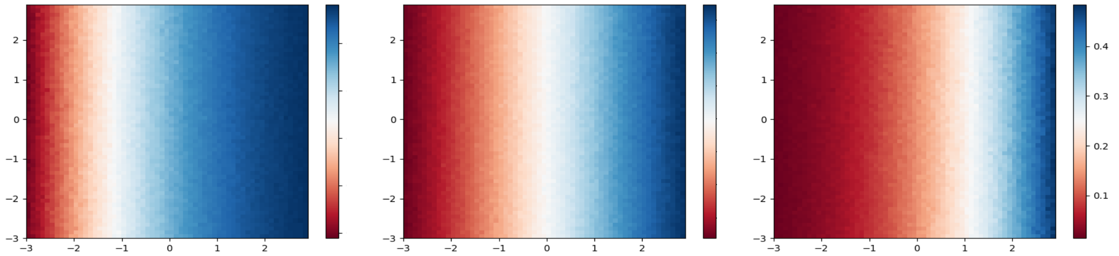

(d) (/cost.me/, [cos.me]), $\theta_3 = -3$ (e) (/cost.me/, [cos.me]), $\theta_3 = 0$ (f) (/cost.me/, [cos.me]), $\theta_3 = 3$

Figure 1: SOT probabilities of the antecedent mapping (/cost.us/, [cos.us]) and the consequent mapping (/cost.me/, [cos.me]) as a function of θ_1 (horizontal axis) and θ_2 (vertical axis) for three choices of θ_3

categorical OT grammars G in $\mathfrak{T}$ is indeed 1.

$$p^{\mathcal{D}}_{\boldsymbol{\theta}}(G) = \text{the probability of sampling } \epsilon_1, \ldots, \epsilon_n \overset{i.i.d.}{\sim} \mathcal{D} \text{ such that the OT grammar corresponding to the ranking } \gg_{\boldsymbol{\theta}+\boldsymbol{\epsilon}} \text{ is indeed } G \tag{8}$$

The typology of stochastic grammars $\mathfrak{T}_{\mathcal{P}}$ obtained as in section 2 from the categorical OT typology $\mathfrak{T}$ and the family $\mathcal{P} = \{p^{\mathcal{D}}_{\boldsymbol{\theta}} \mid \boldsymbol{\theta} \in \mathbb{R}^n\}$ of probability mass functions $p^{\mathcal{D}}_{\boldsymbol{\theta}}$ corresponding to all ranking vectors $\boldsymbol{\theta}$ is called *stochastic OT* (SOT; Boersma, 1997, 1998). We denote by $\overset{\text{SOT}}{\longrightarrow}$ the T-orders corresponding to SOT in the sense of definition 2.

What is the typological structure encoded by SOT's T-orders? Given that the original OT typology is finite (because there are only a finite number of constraint rankings) while the SOT typology is infinite (it contains an infinite number of grammars which assign different probabilities), how much of OT's typological structure is preserved in SOT? These questions are crucial for phonological theory but technically nontrivial. To illustrate, figure 1 plots the SOT probability of the mappings (/cost.us/, [cos.us]) and (/cost.me/, [cos.me]) relative to the three constraints C_1, C_2, C_3 listed above as a function of the ranking value θ_1 of constraint C_1 (horizontal axis) and the ranking value θ_2 of constraint C_2 (vertical axis) for three choices of the ranking value θ_3 of constraint C_3.[4] These plots suggest that the implication (/cost.us/, [cos.us]) $\overset{\text{SOT}}{\rightarrow}$ (/cost.me/, [cos.me]) holds in SOT: the probability of the consequent (/cost.me/, [cos.me]) (plotted in the bottom row) seems to be always larger than the probability of the antecedent (/cost.us/, [cos.us]) (plotted in the top row). But how can this conjecture be checked, given that SOT probabilities seem not to admit a closed-form expression?

The result obtained in section 3 provides a straightforward solution to this problem. Suppose that there exists a positive constant Δ large enough that the distribution $\mathcal{D}$ concentrates most of the probability mass on the interval $[-\Delta, +\Delta]$, as stated in (9). This assumption holds in particular when $\mathcal{D}$ has a bounded support or it is defined through a density (such as a gaussian, as assumed in Boersma, 1997, 1998).

$$(\mathcal{D}([-\Delta, +\Delta]))^n > 1/2 \tag{9}$$

For any constraint ranking $\gg$, consider a ranking vector $\boldsymbol{\theta}$ such that the top $\gg$-ranked constraint has the largest ranking value; the second top $\gg$-ranked constraint has the second largest ranking value; and so on. Furthermore, assume that these ranking values are spaced apart by more than 2Δ. Since the numbers $\epsilon_1, \ldots, \epsilon_n$ are all bounded between $-\Delta$ and $+\Delta$ with probability at least 1/2

---

[4] These plots (as well as those in figure 2) are obtained by sampling for 10,000 times from each stochastic grammar. The distribution $\mathcal{D}$ is a gaussian with mean 0 and variance 2.

and since the ranking values are spaced apart by more than 2Δ, the constraint ranking $\gg_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}$ corresponding to the stochastic ranking vector $\boldsymbol{\theta}+\boldsymbol{\epsilon}$ coincides with the original ranking $\gg$ with probability at least 1/2. In other words, the probability mass function $p^{\mathcal{D}}_{\boldsymbol{\theta}}$ corresponding to this ranking vector $\boldsymbol{\theta}$ according to (8) assigns more than half of the probability mass to the OT grammar corresponding to the ranking $\gg$. The family $\mathcal{P} = \{p^{\mathcal{D}}_{\boldsymbol{\theta}} \,|\, \boldsymbol{\theta} \in \mathbb{R}^n\}$ is therefore sufficiently rich in the sense of definition 3. Proposition 1 thus yields the following

Corollary 1 *Under the mild assumption (9) on the distribution $\mathcal{D}$, the T-order $\xrightarrow{SOT}$ relative to SOT is identical to the T-order $\xrightarrow{OT}$ relative to categorical OT for any constraint and candidate set.* □

In conclusion, despite the SOT typology being infinite, SOT induces the same typological structure as categorical OT, at least when typological structure is measured in terms of T-orders. Furthermore, the technical problem of computing T-orders relative to SOT is reduced to the much easier problem of computing T-orders relative to categorical OT, which indeed admits an efficient solution (Magri, 2018a). This result extends to *partial order OT* (Anttila, 1997a), as the latter is a special case of SOT.

5 Categorial HG and stochastic HG induce the same T-orders

This section shows that completely analogous considerations hold for HG. A weight vector $\mathbf{w} = (w_1, \ldots, w_k, \ldots, w_n) \in \mathbb{R}^n_+$ assigns a nonnegative weight $w_k \geq 0$ to each constraint C_k. The $\mathbf{w}$-*harmony* of a mapping (x, y) is the weighted sum of the constraint violations multiplied by -1, namely $-\sum_{k=1}^{n} w_k C_k(\mathsf{x}, \mathsf{y})$. Because of the minus sign, mappings with a large harmony have few constraint violations. The categorical HG grammar corresponding to a weight vector $\mathbf{w}$ maps a UR x to the surface form y such that the mapping (x, y) has a larger $\mathbf{w}$-harmony than the mapping (x, z) corresponding to any other candidate z in $Gen(\mathsf{x})$ (Legendre et al., 1990; Smolensky and Legendre, 2006). We denote by $\overset{\text{HG}}{\to}$ the T-order corresponding to the typology $\mathfrak{T}$ of the categorical HG grammars corresponding to all nonnegative weight vectors, in the sense of definition 2.

To illustrate, it is easy to verify that the implication (/cost.us/, [cos.us]) $\overset{\text{HG}}{\to}$ (/cost.me/, [cos.me]) considered above holds also relative to the HG typology in the sense of definition 1: every weighting of the three constraints which succeeds on the antecedent mapping (/cost.us/, [cos.us]) also succeeds on the consequent mapping (/cost.me/, [cos.me]). In general, the HG typology is a proper superset of the OT typology (when the set of URs is finite). The HG T-order is therefore a subset of the corresponding OT T-order.

We now turn to the stochastic counterpart of this categorical framework. The *stochastic weight vector* $\mathbf{w}+\boldsymbol{\epsilon} = (w_1+\epsilon_1, \ldots, w_n+\epsilon_n)$ is obtained by adding to the weights $w_1, \ldots, w_n$ some numbers $\epsilon_1, \ldots, \epsilon_n$ sampled independently from each other according to some distribution $\mathcal{D}$ on $\mathbb{R}$.[5] A weight vector $\mathbf{w}$ induces the corresponding probability mass function $p^{\mathcal{D}}_{\mathbf{w}}$ on the categorical HG typology $\mathfrak{T}$ defined in (10). Obviously, this definition yields a probability mass, namely the sum of the masses $p^{\mathcal{D}}_{\mathbf{w}}(G)$ over all the categorical grammars G in the HG typology $\mathfrak{T}$ is equal to 1.

$$p^{\mathcal{D}}_{\mathbf{w}}(G) = \begin{array}{l}\text{the probability of sampling}\\ \epsilon_1, \ldots, \epsilon_n \overset{i.i.d.}{\sim} \mathcal{D} \text{ such that the}\\ \text{HG grammar corresponding to}\\ \text{the weight vector } \mathbf{w}+\boldsymbol{\epsilon} \text{ is } G\end{array} \tag{10}$$

The typology of stochastic grammars $\mathfrak{T}_{\mathcal{P}}$ obtained as in section 2 from the categorical HG typology $\mathfrak{T}$ and the family $\mathcal{P} = \{p^{\mathcal{D}}_{\mathbf{w}} \,|\, \mathbf{w} \in \mathbb{R}^n_+\}$ of probability mass functions $p^{\mathcal{D}}_{\mathbf{w}}$ corresponding to all nonnegative weight vectors $\mathbf{w}$ is called *stochastic HG* (SHG; Boersma and Pater, 2016). We denote by $\xrightarrow{\text{SHG}}$ the T-orders corresponding to SHG in the sense of definition 2.

To illustrate, figure 2 plots the SHG probability of the mappings (/cost.us/, [cos.us]) and (/cost.me/, [cos.me]) as a function of the ranking values of the three constraints C_1, C_2, C_3 listed above. These plots suggest that the implication (/cost.us/, [cos.us]) $\overset{\text{SHG}}{\to}$ (/cost.me/, [cos.me]) holds in SHG as well: the probability of the consequent (/cost.me/, [cos.me]) (plotted in the bottom row) seems to be always larger than the probability of the antecedent (/cost.us/, [cos.us]) (plotted in the top row). The result obtained in section 3 makes sense of this observation, as follows.

---

[5] Some component $w_k+\epsilon_k$ of the corrupted weight vector $\mathbf{w}+\boldsymbol{\epsilon}$ could be negative. In this case, $\mathbf{w}+\boldsymbol{\epsilon}$ could correspond to no HG grammar in $\mathfrak{T}$ and the probability mass defined in (10) could therefore add up to less than 1. This problem can be avoided simply by truncating the corrupted weights at zero, namely by replacing $w_k+\epsilon_k$ with $\max\{w_k+\epsilon_k, 0\}$ in the definition of the corrupted weight vector $\mathbf{w}+\boldsymbol{\epsilon}$ (Magri, 2015).

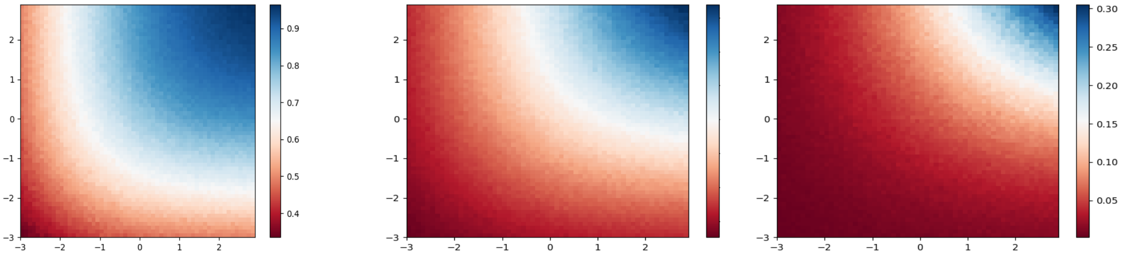

(a) (/cost.us/, [cos.us]), $\theta_3 = -3$ (b) (/cost.us/, [cos.us]), $\theta_3 = 0$ (c) (/cost.us/, [cos.us]), $\theta_3 = 3$

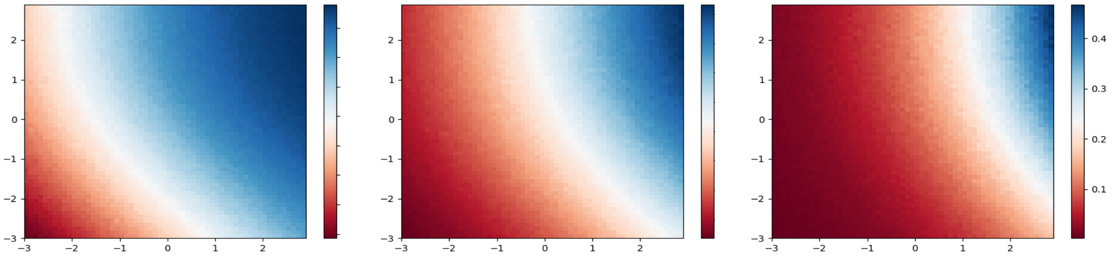

(d) (/cost.me/, [cos.me]), $\theta_3 = -3$ (e) (/cost.me/, [cos.me]), $\theta_3 = 0$ (f) (/cost.me/, [cos.me]), $\theta_3 = 3$

Figure 2: SHG probabilities of the antecedent mapping (/cost.us/, [cos.us]) and the consequent mapping (/cost.me/, [cos.me]) as a function of θ_1 (horizontal axis) and θ_2 (vertical axis) for three choices of θ_3

We consider two URs x and $\widehat{\mathsf{x}}$. We assume that x comes with a finite number $m+1$ of candidates $\mathsf{y}, \mathsf{z}_1, \dots, \mathsf{z}_m$ and that $\widehat{\mathsf{x}}$ comes with a finite number $\widehat{m}+1$ of candidates $\widehat{\mathsf{y}}, \widehat{\mathsf{z}}_1, \dots, \widehat{\mathsf{z}}_{\widehat{m}}$. This assumption is nonrestrictive. In fact, each UR admits only a finite number of optima in HG (Magri, 2018b). Candidate sets can thus be assumed to be finite without loss of generality. We consider a categorical HG grammar in the typology $\mathfrak{T}$ and assume that it maps x and $\widehat{\mathsf{x}}$ to y and $\widehat{\mathsf{y}}$, respectively. This means that any weight vector $\mathbf{w} = (w_1, \dots, w_n)$ corresponding to this HG grammar assigns a larger harmony to the winner mappings (x, y) and $\widehat{\mathsf{x}}, \widehat{\mathsf{y}})$ than to any of the loser mappings $(\mathsf{x}, \mathsf{z}_i)$ and $(\widehat{\mathsf{x}}, \widehat{\mathsf{z}}_j)$ respectively, as stated in (11).

$$\begin{aligned}
&\underbrace{\min_{i=1,\dots,m} \sum_k w_k(C_k(\mathsf{x},\mathsf{z}_i) - C_k(\mathsf{x},\mathsf{y}))}_{\xi} > 0 \\
&\underbrace{\min_{j=1,\dots,\widehat{m}} \sum_k w_k(C_k(\widehat{\mathsf{x}},\widehat{\mathsf{z}}_j) - C_k(\widehat{\mathsf{x}},\widehat{\mathsf{y}}))}_{\widehat{\xi}} > 0
\end{aligned} \tag{11}$$

Let B be an upper bound on the constraint violation differences, so that $|C(\mathsf{x}, \mathsf{z}_i) - C(\mathsf{x}, \mathsf{y})| \leq B$ and $|C(\widehat{\mathsf{x}}, \widehat{\mathsf{z}}_j) - C(\widehat{\mathsf{x}}, \widehat{\mathsf{y}})| \leq B$ for every $i = 1, \dots, m$ and $j = 1, \dots, \widehat{m}$. Suppose again that there exists a positive constant Δ large enough that the distribution $\mathcal{D}$ concentrates most of the probability mass on $[-\Delta, +\Delta]$, in the sense that it satisfies the inequality (9). We consider the weight vector $\lambda\mathbf{w} = (\lambda w_1, \dots, \lambda w_n)$ obtained by rescaling the weight vector $\mathbf{w}$ by a positive scalar $\lambda > 0$ sufficiently large, in the sense of (12).

$$\lambda > \max\left\{\frac{n\Delta B}{\xi}, \frac{n\Delta B}{\widehat{\xi}}\right\} \tag{12}$$

Whenever $\boldsymbol{\epsilon} \in [-\Delta, +\Delta]^n$, the HG grammar corresponding to the stochastic rescaled weight vector $\lambda\mathbf{w} + \boldsymbol{\epsilon}$ maps the UR x to the SR y, as shown in (13). An analogous reasoning shows that it also maps $\widehat{\mathsf{x}}$ to $\widehat{\mathsf{y}}$. In step (13a), we have used the definition (11) of ξ. In step (13b), we have lower bounded $C(\mathsf{x}, \mathsf{z}_i) - C(\mathsf{x}, \mathsf{y})$ with $-B$. In step (13c), we have used the definition (12) of λ.

$$\begin{aligned}
&\sum_k (\lambda w_k + \epsilon_k)(C_k(\mathsf{x},\mathsf{z}_i) - C_k(\mathsf{x},\mathsf{y})) = \\
&= \lambda \sum_k w_k(C_k(\mathsf{x},\mathsf{z}_i) - C_k(\mathsf{x},\mathsf{y})) + \\
&\qquad + \sum_k \epsilon_k(C_k(\mathsf{x},\mathsf{z}_i) - C_k(\mathsf{x},\mathsf{y})) \\
&\overset{(a)}{\geq} \lambda\xi + \sum_k \epsilon_k(C_k(\mathsf{x},\mathsf{z}_i) - C_k(\mathsf{x},\mathsf{y})) \\
&\overset{(b)}{\geq} \lambda\xi - n\Delta B \overset{(c)}{>} 0
\end{aligned} \tag{13}$$

The intuition behind this reasoning (13) is as follows. The rescaled weight vector $\lambda\mathbf{w}$ generates

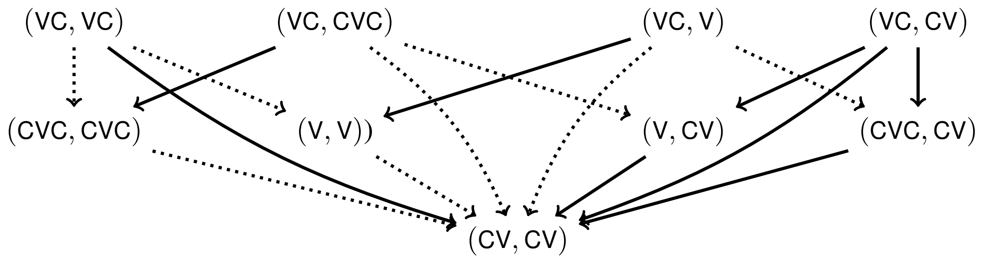

Figure 3: Solid arrows are entailments that hold in OT, HG, SOT, SHG, and ME; dotted arrows are entailments that fail in ME.

the same HG grammar as the weight vector $\mathbf{w}$. If λ is large, the nonzero weights of the rescaled vector $\lambda\mathbf{w}$ are very large (in absolute value). On the other hand, the stochastic values $\epsilon_1, \ldots, \epsilon_n$ are instead small (because bounded between $-\Delta$ and $+\Delta$) and therefore negligible relative to the rescaled weights. The original weight vector $\mathbf{w}$ and the stochastic rescaled vector $\boldsymbol{\epsilon} + \lambda\mathbf{w}$ thus generate the same HG grammar.

In conclusion, the SOT grammar $G_{p^{\mathcal{D}}_{\lambda\mathbf{w}}}$ corresponding to the probability mass function $p^{\mathcal{D}}_{\lambda\mathbf{w}}$ (corresponding to the rescaled weight vector $\lambda\mathbf{w}$) satisfies the identities $G_{p^{\mathcal{D}}_{\lambda\mathbf{w}}}(\mathsf{y}\,|\,\mathsf{x}) > 1/2$ and $G_{p^{\mathcal{D}}_{\lambda\mathbf{w}}}(\widehat{\mathsf{y}}\,|\,\widehat{\mathsf{x}}) > 1/2$. This shows that the family $\mathcal{P} = \{p^{\mathcal{D}}_{\mathbf{w}} \,|\, \mathbf{w} \in \mathbb{R}^n_+\}$ of probability masses is sufficiently rich in the sense of definition 3. Proposition 1 thus yields the following:

Corollary 2 *Under the mild assumption (9) on the distribution $\mathcal{D}$, the T-order $\xrightarrow{SHG}$ relative to SHG is identical to the T-order $\xrightarrow{HG}$ relative to categorical HG for any constraint set and any candidate set which assigns a finite number of candidates to each UR (while the number of URs can be infinite).* □

6 Conclusions

Phonology has traditionally focused on patterns of categorical alternations modeled within categorical frameworks such as OT and HG. More recently, phonology has extended its empirical coverage to quantitative data such as gradient judgments and patterns of variation. This move has required a parallel extension from categorical to stochastic frameworks, such as partial order OT, stochastic OT, and stochastic HG. These stochastic frameworks are "extensions" of the original categorical frameworks in the sense discussed in section 3. One might thus expect the stochastic frameworks to be typologically less restrictive than the original categorical frameworks. This paper has shown that is not the case, at least when typological restrictiveness is measured in terms of the most basic implicational universals, namely T-orders. Indeed, the T-orders induced by partial order and stochastic OT coincide with those induced by categorical OT. Analogously, the T-orders induced by stochastic HG coincide with those induced by categorical HG.

As discussed in a companion paper (Anttila and Magri, 2018), the situation is very different in ME. To illustrate, consider the *basic syllable system* of Prince and Smolensky (2004). The set of forms consists of the four syllable types CV, CVC, V, and VC. Each of them is a candidate of each other. The constraint set consists of the four constraints ONSET, NOCODA, MAX, and DEP. The HG and OT T-orders coincide and consist of 16 entailments with a feasible antecedent, plotted in figure 3. These entailments extend to SOT and SHG, by virtue of the corollaries 1 and 2 obtained above.

ME instead misses the eight dotted entailments. Of the eight entailments which do survive in ME, seven are such that the antecedent and the consequent surface form coincide, plus the entailment $(\mathsf{VC}, \mathsf{VC}) \rightarrow (\mathsf{CV}, \mathsf{CV})$, which is a quirk due to the fact that VC is the most marked syllable type. This restriction to entailments whose antecedent and consequent surface forms coincide is not phonologically plausible. Anttila and Magri (2018) conclude that the ME formalism imposes typological restrictions at odds with phonological intuition.

Acknowledgments

The research reported in this paper has been funded by the *Agence National de la Recherche* (project title: 'The mathematics of segmental phonotactics'). This paper is part of a larger project on T-orders, developed in collaboration with Arto Anttila. His comments on this paper are gratefully acknowledged.

References

Arto Anttila. 1997a. Deriving variation from grammar: A study of Finnish genitives. In Frans Hinskens, Roeland van Hout, and Leo Wetzels, editors, *Variation, change and phonological theory*, pages 35–68. John Benjamins, Amsterdam. Rutgers Optimality Archive ROA-63, http://ruccs.rutgers.edu/roa.html.

Arto Anttila. 1997b. *Variation in Finnish phonology and morphology*. Ph.D. thesis, Stanford University.

Arto Anttila. 2012. Modeling phonological variation. In Abigail C. Cohn, Cécile Fougeron, and Marie Huffman, editors, *The Oxford Handbook of Laboratory Phonology*, pages 76–91. Oxford University Press, Oxford.

Arto Anttila and Curtis Andrus. 2006. T-orders. Manuscript and software (Stanford).

Arto Anttila and Giorgio Magri. 2018. T-orders across categorical and probabilistic constraint-based phonology. Manuscript (Stanford, CNRS).

Paul Boersma. 1997. How we learn variation, optionality and probability. In *Proceedings of the Institute of Phonetic Sciences (IFA) 21*, pages 43–58, University of Amsterdam. Institute of Phonetic Sciences.

Paul Boersma. 1998. *Functional Phonology*. Ph.D. thesis, University of Amsterdam, The Netherlands. The Hague: Holland Academic Graphics.

Paul Boersma and Joe Pater. 2016. Convergence properties of a gradual learning algorithm for Harmonic Grammar. In John McCarthy and Joe Pater, editors, *Harmonic Grammar and Harmonic Serialism*. Equinox Press, London.

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. Harper and Row, New York.

Andries W. Coetzee. 2004. *What it Means to be a Loser: Non-Optimal Candidates in Optimality Theory*. Ph.D. thesis, University of Massachusetts, Amherst.

Andries W. Coetzee and Joe Pater. 2011. The place of variation in phonological theory. In John Goldsmith, Jason Riggle, and Alan Yu, editors, *Handbook of phonological theory*, pages 401–434. Blackwell, Cambridge.

Sharon Goldwater and Mark Johnson. 2003. Learning OT constraint rankings using a Maximum Entropy model. In *Proceedings of the Stockholm Workshop on Variation Within Optimality Theory*, pages 111–120, Stockholm University.

Joseph H. Greenberg. 1963. *Universals of Language*. MIT Press, Cambridge, MA.

G. Guy. 1991. Explanation in variable phonology. *Language Variation and Change*, 3:1–22.

Bruce Hayes. 2017. Varieties of Noisy Harmonic Grammar. In *Proceedings of the 2016 Annual Meeting in Phonology*, pages –.

Bruce Hayes and Colin Wilson. 2008. A Maximum Entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39:379–440.

Paul Kiparsky. 1993. An OT perspective on phonological variation. Handout (Stanford).

Géraldine Legendre, Yoshiro Miyata, and Paul Smolensky. 1990. Harmonic Grammar: A formal multi-level connectionist theory of linguistic well-formedness: Theoretical foundations. In *Annual conference of the Cognitive Science Society 12*, pages 388–395, Mahwah, NJ. Lawrence Erlbaum.

Giorgio Magri. 2015. How to keep the HG weights non-negative: the truncated Perceptron reweighing rule. *Journal of Language Modeling*, 3.2:345–375.

Giorgio Magri. 2018a. Efficient computation of implicational universals in constraint-based phonology through the Hyperplane Separation Theorem. Manuscript (CNRS).

Giorgio Magri. 2018b. Finiteness of optima in constraint-based phonology. Manuscript (Stanford, CNRS).

Alan Prince and Paul Smolensky. 1997. Optimality: From neural networks to universal grammar. *Science*, 275:1604–1610.

Alan Prince and Paul Smolensky. 2004. *Optimality Theory: Constraint Interaction in generative grammar*. Blackwell, Oxford. Original version, Technical Report CU-CS-696-93, Department of Computer Science, University of Colorado at Boulder, and Technical Report TR-2, Rutgers Center for Cognitive Science, Rutgers University, April 1993. Available from the Rutgers Optimality Archive as ROA 537.

Paul Smolensky and Géraldine Legendre. 2006. *The Harmonic Mind*. MIT Press, Cambridge, MA.

Explaining Character-Aware Neural Networks for Word-Level Prediction: Do They Discover Linguistic Rules?

Fréderic Godin, Kris Demuynck, Joni Dambre, Wesley De Neve and Thomas Demeester

IDLab, Ghent University - imec, Ghent, Belgium
`firstname.lastname@ugent.be`

Abstract

Character-level features are currently used in different neural network-based natural language processing algorithms. However, little is known about the character-level patterns those models learn. Moreover, models are often compared only quantitatively while a qualitative analysis is missing. In this paper, we investigate which character-level patterns neural networks learn and if those patterns coincide with manually-defined word segmentations and annotations. To that end, we extend the contextual decomposition (Murdoch et al., 2018) technique to convolutional neural networks which allows us to compare convolutional neural networks and bidirectional long short-term memory networks. We evaluate and compare these models for the task of morphological tagging on three morphologically different languages and show that these models implicitly discover understandable linguistic rules.

1 Introduction

Character-level features are an essential part of many Natural Language Processing (NLP) tasks. These features are for instance used for language modeling (Kim et al., 2016), part-of-speech tagging (Plank et al., 2016) and machine translation (Luong and Manning, 2016). They are especially useful in the context of part-of-speech and morphological tagging, where for example the suffix *-s* can easily differentiate plural words from singular words in English or Spanish.

The use of character-level features is not new. Rule-based taggers were amongst the earliest systems that used character-level features/rules for grammatical tagging (Klein and Simmons, 1963). Other approaches rely on fixed lists of affixes (Ratnaparkhi, 1996; Toutanova et al., 2003). Next, these features are used by a tagging model, such as a rule-based model or statistical model. Rule-based taggers are transparent models that allow us to easily trace back why the tagger made a certain decision (e.g., Brill (1994)). Similarly, statistical models are merely a weighted sum of features.

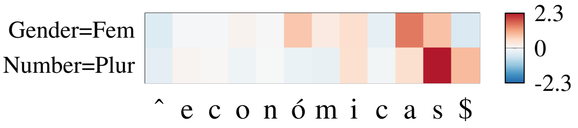

Figure 1: Individual character contributions of the Spanish adjective *económicas*. The character *a* has the highest positive (red) contribution for predicting the label *Gender=Fem*, and the character *s* for predicting the label *Number=Plur*. This coincides with our linguistic knowledge of Spanish.

For example, Brill (1994)'s transformation-based error-driven tagger uses a set of templates to derive rules by fixing errors. The following rule template:

> *"Change the most-likely tag* ***X*** *to* ***Y*** *if the last (1,2,3,4) characters of the word are* ***x****"*,

resulted in the rule:

> *"Change the tag* ***common noun*** *to* ***plural common noun*** *if the word has suffix* ***-s****"*.

Subsequently, whenever the tagger makes a tagging mistake, it is easy to trace back why this happened. Following the above rule, the word *mistress* will mistakingly be tagged as a plural common noun while it actually is a common noun[1].

This is in stark contrast with the most recent generation of part-of-speech and morphological taggers which mainly rely on neural networks.

---

[1] In Brill (1994), an additional rule encodes an exception to this rule to correctly tag the word *mistress*.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3275–3284
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

Words are split into individual characters and are in general either aggregated using a Bidirectional Long Short-Term Memory network (BiLSTM) (Plank et al., 2016) or Convolutional Neural Network (CNN) (dos Santos and Zadrozny, 2014). However, it is currently unknown which character-level patterns these neural network models learn and whether these patterns coincide with our linguistic knowledge. Moreover, different neural network architectures are currently only compared quantitatively and lack a qualitative analysis.

In this paper, we investigate which character patterns neural networks learn and to what extent those patterns comprise any known linguistic rules. We do this for three morphologically different languages: Finnish, Spanish and Swedish. A Spanish example is shown in Figure 1. By visualizing the contributions of each character, we observe that the model indeed uses the suffix *-s* to correctly predict that the word is plural.

Our main contributions are as follows:

- We show how word-level tagging decisions can be traced back to specific sets of characters and interactions between them.
- We extend the contextual decomposition method (Murdoch et al., 2018) to CNNs.
- We quantitatively compare CNN and BiLSTM models in the context of morphological tagging by performing an evaluation on three manually segmented and morphologically annotated corpora.
- We found out that the studied neural models are able to implicitly discover character patterns that coincide with the same rules linguists use to indicate the morphological function of subword segments.

Our implementation is available online[2].

2 Related Work

Neural network-based taggers currently outperform statistical taggers in morphological tagging (Heigold et al., 2017) and part-of-speech tagging (Plank et al., 2016) for a wide variety of languages. Character-level features form a crucial part of many of these systems. Generally, two neural network architectures are considered for aggregating the individual characters: a BiLSTM (Ling et al., 2015; Plank et al., 2016) or a CNN (dos Santos and Zadrozny, 2014; Bjerva et al., 2016; Heigold et al., 2017). These architectures outperform similar models that use manually defined features (Ling et al., 2015; dos Santos and Zadrozny, 2014). However, it is still unclear which useful character-level features they have learned. Architectures are compared quantitatively but lack insight into learned patterns. Moreover, Vania and Lopez (2017) showed in the context of language modeling that training a BiLSTM on ground truth morphological features still yields better results than eight other character-based neural network architectures. Hence, this raises the question which patterns neural networks learn and whether these patterns coincide with manually-defined linguistic rules.

While a number of interpretation techniques have been proposed for images (Springenberg et al., 2014; Selvaraju et al., 2017; Shrikumar et al., 2017), these are generally not applicable in the context of NLP where LSTMs are mainly used. Moreover, gradient-based techniques are not trustworthy when strongly saturating activation functions such as *tanh* and *sigmoid* are used (e.g., Li et al. (2016a)). Hence, current interpretations in NLP are limited to visualizing the magnitude of the LSTM hidden states of each word (Linzen et al., 2016; Radford et al., 2017; Strobelt et al., 2018), removing words (Li et al., 2016b; Kádár et al., 2017) or changing words (Linzen et al., 2016) and measuring the impact, or training surrogate tasks (Adi et al., 2017; Chrupała et al., 2017; Belinkov et al., 2017). These techniques only provide limited local interpretations and do not model fine-grained interactions of groups of inputs or intermediate representations. In contrast, Murdoch et al. (2018) recently introduced an LSTM interpretation technique called Contextual Decomposition (CD), providing a solution to the aforementioned issues. We will build upon this interpretation technique and introduce an extension for CNNs, making it possible to compare different neural network architectures within a single interpretation framework.

3 Method

For visualizing the contributions of character sets, we use the recently introduced Contextual Decomposition (CD) framework, as originally developed for LSTMs (Murdoch et al., 2018), and extend it to

---

[2] `https://github.com/FredericGodin/ContextualDecomposition-NLP`

CNNs. First, we introduce the concept of CD, followed by the extension for CNNs. For details on CD for LSTMs, we refer the reader to the aforementioned paper. Finally, we explain how the CD of the final classification layer is done.

3.1 Contextual decomposition

The idea behind CD is that, in the context of character-level decomposition, we can decompose the output value of the network for a certain class into two distinct groups of contributions: (1) contributions originating from a specific character or set of characters within a word and (2) contributions originating from all the other characters within the same word.

More generally, we can decompose every output value z of every neural network component into a relevant contribution β and an irrelevant contribution γ:

$$z = \beta + \gamma \tag{1}$$

3.2 Decomposing CNN layers

A CNN typically consist of three components: the convolution itself, an activation function and an optional max-pooling operation. We will discuss each component in the next paragraphs.

Decomposing the convolution Given a sequence of character embeddings $\boldsymbol{x}_1, ..., \boldsymbol{x}_T \in \mathbb{R}^{d_1}$ of length T, we can calculate the convolution of size n of a single filter over the sequence $\boldsymbol{x}_{1:T}$ by applying the following equation to each n-length subsequence $\{\boldsymbol{x}_{t+i}, i = 0, .., n-1\}$, denoted as $\boldsymbol{x}_{t:t+n-1}$:

$$z_t = \sum_{i=0}^{n-1} W_i \cdot \boldsymbol{x}_{t+i} + b, \tag{2}$$

with $z_t \in \mathbb{R}$ and where $W \in \mathbb{R}^{d_1 \times n}$ and $b \in \mathbb{R}$ are the weight matrix and bias of the convolutional filter. W_i denotes the i-th column of the weight matrix W.

When we want to calculate the contribution of a subset of characters, where $\boldsymbol{S}$ is the set of corresponding character position indexes and $\boldsymbol{S} \subseteq \{1, ..., T\}$, we should decompose the output of the filter z_t into three parts:

$$z_t = \beta_t + \gamma_t + b. \tag{3}$$

That is, the relevant contribution β_t originating from the selected subset of characters with indexes $\boldsymbol{S}$, the irrelevant contribution γ_t originating from the remaining characters in the sequence, and a bias which is deemed neutral (Murdoch et al., 2018).

This can be achieved by decomposing the convolution itself as follows:

$$\beta_t = \sum_{i=0}^{n-1} W_i \cdot \boldsymbol{x}_{t+i} \qquad (t+i) \in \boldsymbol{S}, \tag{4}$$

$$\gamma_t = \sum_{i=0}^{n-1} W_i \cdot \boldsymbol{x}_{t+i} \qquad (t+i) \notin \boldsymbol{S}, \tag{5}$$

Linearizing the activation function After applying a linear transformation to the input, a nonlinearity is typically applied. In CNNs, the ReLU activation function is often used.

In Murdoch et al. (2018), a linearization method for the non-linear activation function f is proposed, based on the differences of partial sums of all N components y_i involved in the preactivation sum z_t. In other words, we want to split $f_{ReLU}(z_t) = f_{ReLU}(\sum_{i=1}^{N} y_i)$ into a sum of individual linearized contributions $L_{f_{ReLU}}(y_i)$, namely $f_{ReLU}(\sum_{i=1}^{N} y_i) = \sum_{i=1}^{N} L_{f_{ReLU}}(y_i)$. To that end, we compute $L_{f_{ReLU}}(y_k)$, the linearized contribution of y_k as the average difference of partial sums over all possible permutations $\pi_1, ..., \pi_{M_N}$ of all N components y_i involved:

$$L_f(y_k) = \frac{1}{M_N} \sum_{i=1}^{M_N} [f(\sum_{l=1}^{\pi_i^{-1}(k)} y_{\pi_i(l)}) - f(\sum_{l=1}^{\pi_i^{-1}(k)-1} y_{\pi_i(l)})] \tag{6}$$

Consequently, we can decompose the output c_t after the activation function as follows:

$$\begin{aligned} c_t &= f_{ReLU}(z_t) && (7) \\ &= f_{ReLU}(\beta_{z,t} + \gamma_{z,t} + b) && (8) \\ &= L_{ReLU}(\beta_{z,t}) \\ &\quad + [L_{ReLU}(\gamma_{z,t}) + L_{ReLU}(b)] && (9) \\ &= \beta_{c,t} + \gamma_{c,t} && (10) \end{aligned}$$

Following Murdoch et al. (2018), $\beta_{c,t}$ contains the contributions that can be directly attributed to the specific set of input indexes $\boldsymbol{S}$. Hence, the bias b is part of $\gamma_{c,t}$. Note that, while the decomposition in Eq. (10) is exact in terms of the total sum, the individual attribution to relevant ($\beta_{c,t}$) and irrelevant ($\gamma_{c,t}$) is an approximation, due to the linearization.

Max-pooling over time When applying a fixed-size convolution over a variable-length sequence, the output is again of variable size. Hence, a max-pooling operation is executed over the time dimension, resulting in a fixed-size representation that is independent of the sequence length:

$$c = \max_t(c_t). \tag{11}$$

Instead of applying a max operation over the $\beta_{c,t}$ and $\gamma_{c,t}$ contributions separately, we first determine the position t of the highest c_t value and propagate the corresponding $\beta_{c,t}$ and $\gamma_{c,t}$ values.

3.3 Calculating the final contribution scores

The final layer is a classification layer, which is the same for a CNN- or LSTM-based architecture. The probability p_j of predicting class j is defined as follows:

$$p_j = \frac{e^{W_j \cdot \boldsymbol{x} + b_j}}{\sum_{i=1}^{C} e^{W_i \cdot \boldsymbol{x} + b_i}}, \tag{12}$$

in which $W \in \mathbb{R}^{d_2 \times C}$ is a weight matrix and W_i the i-th column, $\boldsymbol{x} \in \mathbb{R}^{d_2}$ the input, $\boldsymbol{b} \in \mathbb{R}^{d_2}$ the bias vector and b_i the i-th element, d_2 the input vector size and C the total number of classes.

The input $\boldsymbol{x}$ is either the output $\boldsymbol{c}$ of a CNN or $\boldsymbol{h}$ of a LSTM. Consequently, we can decompose $\boldsymbol{x}$ into $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ contributions. In practice, we only consider the preactivation and decompose it as follows:

$$W_j \cdot \boldsymbol{x} + b_j = W_j \cdot \boldsymbol{\beta} + W_j \cdot \boldsymbol{\gamma} + b_j. \tag{13}$$

Finally, the contribution of a set of characters with indexes $\boldsymbol{S}$ to the final score of class j is equal to $W_j \cdot \boldsymbol{\beta}$. The latter score is used throughout the paper for visualizing contributions of sets of characters.

4 Experimental Setup

We execute experiments on morphological tagging in three different languages: Finnish, Spanish and Swedish. We describe the dataset in Section 4.1, whereas model and training details can be found in Section 4.2.

4.1 Dataset

For our experiments, we use the Universal Dependencies 1.4 (UD) dataset (Nivre et al., 2016), which contains morphological features for a large number of sentences. Additionally, we acquired manually-annotated character-level morphological segmentations and labels for a subset of the test set for three morphological different languages: Finnish, Spanish and Swedish. [3]

Table 1: Overview of the training, validation and test set used.

	Finnish	**Spanish**	**Swedish**
Train words	53547	62556	16295
Valid words	2317	4984	1731
Test words	2246	956	3538
Annotated Test pairs	278	340	137

For each language, Silfverberg and Hulden (2017) selected the first non-unique 300 words from the UD test set and manually segmented each word according to the associated lemma and morphological features in the dataset. Whenever possible, they assigned each feature to a specific subset of characters. For example, the Spanish word "económicas" is segmented as follows:

- económic : lemma=económico
- a : gender=feminine
- s : number=plural

For our experiments, we are only interested in word/feature pairs for which a feature can be assigned to a specific subset of characters. Hence, we filter the test set on those specific word/feature pairs. In the above example, we have two word/feature pairs. This resulted in 278, 340 and 137 word/feature pairs for Finnish, Spanish and Swedish, respectively. Using the same procedure, we selected relevant feature classes, resulting in 12, 6 and 9 feature classes for Finnish, Spanish and Swedish, respectively.[4] For each class, when a feature was not available, we introduced an additional *Not Applicable* (NA) label.

We always train and validate on the full UD dataset for which we have filtered out all duplicate words. After that, we perform our analysis on either the UD test set or the annotated subset of manually segmented and annotated words. An overview can be found in Table 1.

---

[3] Available online: http://github.com/mpsilfve/ud-segmenter/commit/5959214d494cbc13e53e1b26650813ff950d2ee3

[4] Full list available as supplementary material

4.2 Model

We experiment with both a CNN and BiLSTM architecture for character-level modeling of words.

At the input, we split every word into characters and add a start-of-word (^) and an end-of-word ($) character. With every character, we associate a character embedding of size 50.

Our CNN architecture is inspired by Kim et al. (2016) and consists of a set of filters of varying width, followed by a ReLU activation function and a max-over-time pooling operation. We adopt their small-CNN parameter choices and have 25, 50, 75, 100, 125 and 150 convolutional filters of size 1, 2, 3, 4, 5 and 6, respectively. We do not add an additional highway layer.

For the character-level BiLSTM architecture, we follow the variant used in Plank et al. (2016). That is, we simply run a BiLSTM over all the characters and concatenate the final forward and backward hidden state. To obtain a similar number of parameters as the CNN model, we set the hidden state size to 100 units for each LSTM.

Finally, the word-level representation generated by either the CNN or BiLSTM architecture is classified by a multinomial logistic regression layer. Each morphological class type has a different layer. We do not take into account context to rule out any influence originating from somewhere other than the characters of the word itself.

Training details For morphological tagging, we train a single model for all classes at once. We minimize the joint loss by summing the cross-entropy losses of each class. We orthogonally initialize all weight matrices, except for the embeddings, which are uniformly initialized ([-0.01;0.01]). All models are trained using Adam (Kingma and Ba, 2015) with minibatches of size 20 and learning rate 0.001. No specific regularization is used. We select our final model based on early stopping on the validation set.

5 Experiments

First, we verify that the CD algorithm works correctly by executing a controlled experiment with a synthetic token. Next, we quantitatively and qualitatively evaluate on the full test set.

5.1 Validation of contextual decomposition for convolutional neural networks

To verify that the contextual decomposition of CNNs works correctly, we devise an experiment in which we add a synthetic token to a word of a certain class, testing whether this token gets a high attribution score with respect to that specific class.

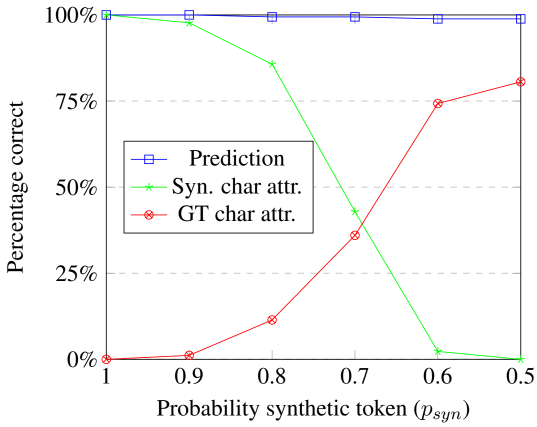

Figure 2: Comparison of the contribution of the synthetic character versus the Ground Truth (GT) character for the class $t = 1$. The prediction curve denotes the classification accuracy for class $t = 1$, and consequently, the prediction curve denotes the upper bound for the attributions.

Given a word w and a corresponding binary label t, we add a synthetic character c to the beginning of word w with probability p_{syn} if that word belongs to the class $t = 1$ and with probability $1 - p_{syn}$ if that word belongs to the class $t = 0$. Consequently, if $p_{syn} = 1$, the model should predict the label with a 100% accuracy, thus attributing this to the synthetic character c. When $p_{syn} = 0.5$, the synthetic character does not provide any additional information about the label t, and c should thus have a small contribution.

Experimental setup We train a CNN model on the Spanish dataset and only use words having the morphological label *number*. This label has two classes *plur* and *sing*, and assign those classes to the binary labels zero and one, respectively. Furthermore, we add a synthetic character to each word with probability p_{syn}, varying p_{syn} from 1 to 0.5 with steps of 0.1. We selected 112 unique word/feature pairs from our test set with label *sing* or *plur*. While plurality is marked by the suffix *s*, a variety of suffixes are used for the singular form. Therefore, we focus on the latter class ($t = 1$). The corresponding suffix is called the Ground Truth (GT) character.

To measure the impact of p_{syn}, we add a synthetic character to each word of the class $t = 1$ and

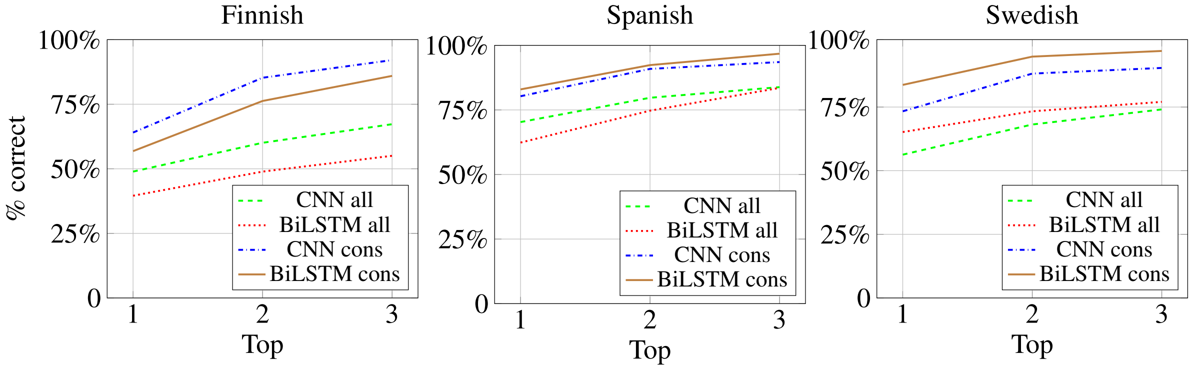

Figure 3: Evaluation of the attributions of CNN and BiLSTM models on the three different languages.

calculate the contribution of each character by using the CD algorithm. We run the experiment five times with a different random seed and report the average correct attribution. The attribution is correct if the contribution of the synthetic/GT character is the highest contribution of all character contributions.

Results The results of our evaluation are depicted in Figure 2. When $p_{syn} = 1$, all words of the class $t = 1$ contain the synthetic character, and consequently, the accuracy for predicting $t = 1$ is indeed 100%. Moreover, the correct prediction is effectively attributed to the synthetic character ('syn. char attr.' in Figure 2 at 100%), with the GT character being deemed irrelevant. When the synthetic character probability p_{syn} is lowered, the synthetic character is less trustworthy and the GT character becomes more important (increasing 'GT char attr.' in Figure 2). Finally, when $p_{syn} = 0.5$, the synthetic character is equally plausible in both classes. Hence, the contribution of the synthetic character becomes irrelevant and the model attributes the prediction to other characters.

Consequently, we can conclude that whenever there is a clear character-level pattern, the model learns the pattern and the CD algorithm is able to accurately attribute it to the correct character.

5.2 Evaluation of character-level attribution

In this section, we measure and analyze (1) which characters contribute most to the final prediction of a certain label and (2) whether those contributions coincide with our linguistic knowledge about a language. To that end, we train a model to predict morphological features, given a particular word. The model does not have prior word segmentation information and thus needs to discover useful character patterns by itself. After training, we calculate the attribution scores of each character pattern within a word with respect to the correct feature class using CD, and evaluate whether this coincides with the ground truth attribution.

Table 2: Average accuracy of all models trained on Finnish, Spanish and Swedish for the task of morphological feature prediction for all unique words in the full UD test set.

	Finnish	**Spanish**	**Swedish**
Maj. Vote	82.20%	72.39%	69.79%
CNN	94.81%	88.93%	**90.09%**
BiLSTM	**95.13%**	**89.33%**	89.45%

Model We train CNN and BiLSTM models on Finnish, Spanish and Swedish. The average accuracies on the full test set are reported in Table 2.[5] As a reference for the trained models' ability to predict morphological feature classes, we provide a naive baseline, constructed from the majority vote for each feature type.

Overall, our neural models yield substantially higher average accuracies than the baseline and perform very similar. Consequently, both the CNN and LSTM models learned useful character patterns for predicting the correct morphological feature classes. Hence, this raises the question whether these patterns coincide with our linguistic knowledge.

Evaluation For each annotated word/feature pair, we measure if the ground truth character se-

---

[5] The results of the individual classes are provided as supplementary material.

(a) Example of Finnish. Word (verb): *ol**i**vat (were)*, target class: *Tense=Past*

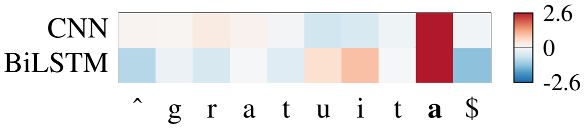

(b) Example of Spanish. Word (adjective): *gratuit**a** (free)*, target: *Gender=Fem.*

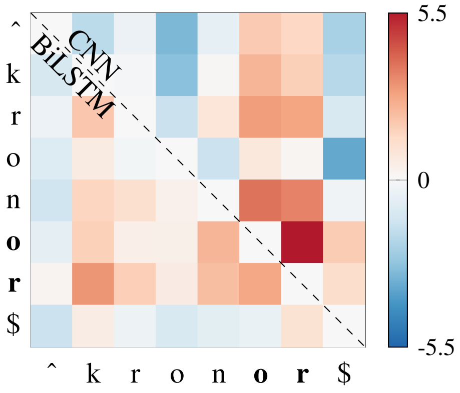

(c) Example of Swedish. Word (noun): *kron**or** (Swedish valuta as in dollars)*, target: *Number=Plur.*

Figure 4: Character-level contributions for predicting a particular class. Positive contributions are highlighted in red and negative contributions in blue. The ground truth character sequence is highlighted in bold.

quence corresponds to the set or sequence of characters with the same length within the considered word that has the highest contribution for predicting the correct label for that word.

In the first setup, we only compare with character sequences having a consecutive set of characters (denoted *cons*). In the second setup, we compare with any set of characters (denoted *all*). We rank the contributions of each character set and report top one, two, and three scores. Because start-of-word and end-of-word characters are not annotated in the dataset, we do not consider them part of the candidate character sets.

Results The aggregated results for all classes and character sequence lengths are shown in Figure 3. In general, we observe that for almost all models and setups, the contextual decomposition attribution coincides with the manually-defined segmentations for at least half of the word/feature pairs. When we only consider the top two consecutive sequences (marked as *cons*), accuracies range from 76% up to 93% for all three languages. For Spanish and Swedish, the top two accuracies for character sets (marked as *all*) are still above 67%, despite the large space of possible character sets, whereas all ground truth patterns are consecutive sequences. While the accuracy for Finnish is lower, the top two accuracy is still above 50%.

Examples for Finnish, Spanish and Swedish are shown in Figure 4. For Finnish, the character with the highest contribution *i* coincides with the ground truth character for the CNN model. This is not the case for the BiLSTM model which focuses on the character *v*, even though the correct label is predicted. For Spanish, both models strongly focus on the ground truth character *a* for predicting the feminine gender. For Swedish, the ground truth character sequence is the suffix *or* which denotes plurality. Given that *or* consists of two characters, all contributions of character sets of two characters are visualized. As can be seen, the most important set of two characters is {o,r} for the CNN and {k,r} for the BiLSTM model. However, {o,r} is the second most important character set for the BiLSTM model. Consequently, the BiLSTM model deemed the interaction between a root and suffix character more important than between two suffix characters.

5.3 Analysis of learned patterns

In the previous section, we showed that there is a strong relationship between the manually-defined morphological segmentation and the patterns a neural network learns. However, there is still an accuracy gap between the results obtained using consecutive sequences only and results obtained using all possible character sets. Hence, this leads to the question which patterns the neural network focuses on, other than the manually defined patterns we evaluated before. To that end, for each of the three languages, we selected a morphological class of interest and evaluated for all words in the full UD test set that were assigned to that class what the most important character set of length one, two and three was. In other words, we evaluated for each word for which the class was cor-

Table 3: The most frequent character sets used by a model for predicting a specific class. The frequency of occurrence is shown between brackets. An underscore denotes an unknown character.

		One character	**Two characters**	**Three characters**	**Examples**
Finnish Tense=Past	BiL.	i (69%), t (22%), v (4%), a (2%)	ti (13%), t_i (12%), v_t (9%), ui (6%)	tti (8%), iv_t (5%), t__ti (3%), sti (3%)	ol**iv**a**t**, näy**tti**kään
	CNN	i (71%), t (8%), s (6%), o (5%)	ui (12%), si (11%), ti (11%), oi (9%)	a__ui (3%), tii (3%), iv__$ (2%), ui__t (2%)	tie**si**, me**i**dät
Spanish Gend=Fem	BiL.	a (69%), i (16%), d (6%), e (4%)	as (23%), a$ (13%), ad (7%), ia (5%)	ia$ (4%), ad$ (3%), da$ (3%), ca$ (2%)	toleranc**ia**, ciud**ad**
	CNN	a (77%), ó (14%), n (4%), d (3%)	a$ (34%), as (20%), da (8%), ió (7%)	dad (5%), da$ (4%), a_ió (4%), sió (2%)	firm**as**, preci**sió**n
Swedish Numb=Plur	BiL.	n (25%), r (19%), a (14%), g (7%)	na (13%), a__r (4%), or (3%), n__r (3%)	iga (5%), rna (3%), ner (1%), der (1%)	kron**or**, perio**der**
	CNN	n (21%), a (18%), r (15%), d (5%)	rn (8%), na (5%), or (4%), er (3%)	rna (7%), arn (3%), iga (2%), n_ar (2%)	krafte**rna**, sak**er**

rectly predicted, which character set had the highest positive contribution towards predicting that class. The results can be found in Table 3.

Finnish In Finnish, adding the suffix *i* to a verb, transforms it in the past tense. Sometimes the character *s* is added, resulting in the suffix *si*. The latter is a frequently used bigram pattern by the CNN but less by the BiLSTM. The BiLSTM combines the suffix *i* with another suffix *vat* which denotes third person plural in the character pattern *iv_t*.

Spanish While there is no single clear-cut rule for the Spanish gender, in general the suffix *a* denotes the feminine gender in adjectives. However, there exist many nouns that are feminine but do not have the suffix *a*. Teschner and Russell (1984) identify *d*, and *ión* as typical endings of feminine nouns, which our models identified too as for example *ad$* or *ió/sió*.

Swedish In Swedish, there exist four suffixes for creating a plural form: *or*, *ar*, *(e)r* and *n*. Both models identified the suffix *or*. However, similar to Finnish, multiple suffixes are merged. In Swedish, the suffix *na* only occurs together with one of the first three plural suffixes. Hence, both models correctly identified this pattern as an important pattern for predicting the class number=plural, rather than the linguistically-defined pattern.

5.4 Interactions of learned patterns

In the previous section, the pattern *a$* showed to be the most important pattern in 34% of the correctly-predicted feminine Spanish words in our dataset. However, there exist many words that end with the character *a* that are not feminine. For example the third person singular form of the verb *gustar* is *gusta*. Hence, this raises the question if the model will classify *gusta* wrongly as feminine or correctly as NA. As an illustration of the applicability of CD for morphological analysis, we will study this case in more detail.

From the full UD test set, we selected all words that end with the character *a* and that do not belong to the class gender=feminine. Using the Spanish CNN model, we predicted the gender class for each word and divided the words into two groups: predicted as feminine and predicted as not-feminine (_NA_ or masculine). The resulted in 44 and 199 words. Next, for each word in both groups we calculated the most positively and negatively contributing character set out of all possible character sets of any length within the considered word, using the CD algorithm. We compared the contribution scores in both groups using a Kruskal-Wallis significance test.[6] While no significant ($p < 0.05$) difference could be found between the positive contributions of both groups (p=1.000), a borderline significant difference could be found between the negative

---

[6] The full statistical analysis is provided as supplementary material.

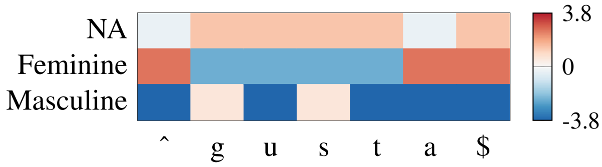

Figure 5: Visualization of the most positively and negatively contributing character set for each class of the morphological feature class gender for the Spanish verb *gusta* (likes).

contributions of words predicted as feminine and words predicted as not-feminine (p=0.070).

Consequently, the CNN model's classification decision is based on finding enough negative evidence to counteract the positive evidence found in the pattern *a$*, which CD was able to uncover.

A visualization of this interaction is shown in Figure 5 for the word *gusta*. While the positive evidence is the strongest for the class feminine, the model identifies the verb stem *gust* as negative evidence which ultimately leads to the correct final prediction NA.

6 Conclusion

While neural network-based models are part of many NLP systems, little is understood on how they handle the input data. We investigated how specific character sequences at the input of a neural network model contribute to word-level tagging decisions at the output, and if those contributions follow linguistically interpretable rules.

First, we presented an analysis and visualization technique to decompose the output of CNN models into separate input contributions, based on the principles outlined by Murdoch et al. (2018) for LSTMs. This allowed us then to quantitatively and qualitatively compare the character-level patterns the CNNs and BiLSTMs learned for the task of morphological tagging. We showed that these patterns generally coincide with the morphological segments as defined by linguists for three morphologically different languages, but that sometimes other linguistically plausible patterns are learned. Finally, we showed that our CD algorithm for CNNs is able to explain why the model made a wrong or correct prediction.

By visualizing the contributions of each input unit or combinations thereof, we believe that much can be learned on how a neural network handles the input data, why it makes certain decisions, or even for debugging neural network models.

Acknowledgments

The authors would like to thank the anonymous reviewers and members of IDLab for their valuable feedback. FG would like to thank Kim Bettens for helping out with the statistical analysis.

The research activities as described in this paper were funded by Ghent University, imec, Flanders Innovation & Entrepreneurship (VLAIO), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *International Conference on Representation Learning (ICLR)*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the Association for Computational Linguistics (ACL)*.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Eric Brill. 1994. A report of recent progress in transformation-based error-driven learning. In *Proceedings of the Workshop on Human Language Technology (HLT)*.

Grzegorz Chrupała, Lieke Gelderloos, and Afra Alishahi. 2017. Representations of language in a model of visually grounded speech signal. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Representation Learning (ICLR)*.

Sheldon Klein and Robert F. Simmons. 1963. A computational approach to grammatical coding of english words. *Journal of the Association for Computing Machinery*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016a. Visualizing and understanding neural models in nlp. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of Association for Computational Linguistics (ACL)*.

W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from LSTMs. *International Conference on Learning Representations (ICLR)*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *International Confernce of Machine Learning (ICML)*.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision (ICCV)*.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Miikka Silfverberg and Mans Hulden. 2017. Automatic morpheme segmentation and labeling in universal dependencies resources. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2014. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806.

Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M. Rush. 2018. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*.

Richard Teschner and William Russell. 1984. The gender patterns of spanish nouns: an inverse dictionary-based analysis. *Hispanic Linguistics 1*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the Association for Computational Linguistics (ACL)*.

Adapting Word Embeddings to New Languages with Morphological and Phonological Subword Representations

Aditi Chaudhary, Chunting Zhou, Lori Levin,
Graham Neubig, David R. Mortensen, Jaime G. Carbonell
Language Technologies Institute, Carnegie Mellon University
`{aschaudh,ctzhou,lsl,gneubig,dmortens,jgc}@cs.cmu.edu`

Abstract

Much work in Natural Language Processing (NLP) has been for resource-rich languages, making generalization to new, less-resourced languages challenging. We present two approaches for improving generalization to low-resourced languages by adapting continuous word representations using linguistically motivated subword units: phonemes, morphemes and graphemes. Our method requires neither parallel corpora nor bilingual dictionaries and provides a significant gain in performance over previous methods relying on these resources. We demonstrate the effectiveness of our approaches on Named Entity Recognition for four languages, namely Uyghur, Turkish, Bengali and Hindi, of which Uyghur and Bengali are low resource languages, and also perform experiments on Machine Translation. Exploiting subwords with transfer learning gives us a boost of +15.2 NER F1 for Uyghur and +9.7 F1 for Bengali. We also show improvements in the monolingual setting where we achieve (avg.) +3 F1 and (avg.) +1.35 BLEU.

1 Introduction

Continuous word representations have demonstrated utility in state-of-the-art neural models for several NLP tasks, such as named entity recognition (NER; Ma and Hovy (2016)), machine reading (Tan et al., 2017), sentiment analysis (Tang et al., 2016; Yu et al., 2018), and machine translation (MT; Qi et al. (2018)). While the training of these word vectors does not rely on explicit human supervision, their quality is highly contingent on the size and quality of the unlabeled corpora available. There are over 7000 languages in the world (Hammarström et al., 2018), and corpora with sufficient size and coverage are available for just a handful, making it unclear how these methods will perform in the more common low-resource setting.

Disheartening though this high dependence on resources sounds, several efforts (Adams et al., 2017; Haghighi et al., 2008; Bharadwaj et al., 2016; Mayhew et al., 2017) have shown considerable performance gains across different tasks in the low resource setting by transferring knowledge from related high-resource languages. Most existing approaches for learning cross-lingual word embeddings (Ruder, 2017) either extend the monolingual objective function by adding a cross-lingual regularization objective which is then jointly optimized or use mapping-based approaches to align similar words across languages. These post-hoc coordination methods rely on bilingual lexicons or parallel corpora, which are typically of limited quantity and uncertain quality.

In this paper, we take a different task: focusing instead on the *similarity of the surface forms, phonology, or morphology* of the two transfer languages. Specifically, inspired by Ling et al. (2015), who demonstrate the effectiveness of character-level modeling for knowledge sharing in multilingual scenarios, we propose two approaches to transfer word embeddings using different types of linguistically-inspired subword-level information. Both approaches focus on mapping the low resource language embeddings closer to those of the high resource language and are executed using two different training regimes. We explore the effect of different subword units—characters, lemmas, inflectional properties, and phonemes— as each one offers a unique linguistic insight, discussed more in Section 3. Our proposed approaches do require language specific resources, but importantly do not depend on cross-lingual resources and achieve considerable performance gains over existing methods which do.

We evaluate our proposed approach on two downstream tasks: NER, which deals with detecting and classifying Named Entities (NEs) into pre-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3285–3295
Brussels, Belgium, October 31 - November 4, 2018.

defined categories (Nadeau and Sekine, 2007), and MT to English. For the purposes of error analysis and discussion, we focus on the NER task in particular. NEs are typically noun phrases and occur rarely in the corpus, making the generalization across types and domains difficult. We chose NER as our test bed because word vectors have a direct impact on NER model performance— as suggested by (Ruder, 2017) and observed by us in Table 3, where the model without any pre-trained embeddings scores an average of 18 F1 points less. It thus provides a transparent way to measure the effectiveness of different subword units.

This paper makes the following contributions:

1. We show that embeddings trained on subword representations yield better task performance than those trained only on whole words. This is especially true in a transfer setting, where subword representations also outperform a word alignment based method. We further show that embeddings trained on morphological representations often outperform those trained only on whole words.
2. We demonstrate that training embeddings on character-based phonemic representations presents substantial performance advantages over training on orthographic characters in some transfer settings, e.g. when there are script differences across languages. These advantages are in addition to those from morphological representations (lemmas and morphological properties).
3. We produce continuous representations for each subword unit, giving researchers the ability to use them in their own tasks as they see fit. The code [1] for training word embeddings and the embeddings [2] which produced the best results are publicly available. We also release morphological analyzers for Hindi and Bengali[3].

2 Skipgram Objective

The two most popular training objectives for monolingual word embeddings are the skipgram and continuous-bag-of-words (CBOW), introduced by Mikolov et al. (2013a). The skipgram model attempts to predict the context surrounding a word, given the word itself whereas CBOW predicts the word given its context. Formally, given a corpus having a sequence of words $w_1, w_2, \cdots, w_T$, the skip-gram model maximizes the following log-likelihood:

$$\sum_{i=1}^{T} \sum_{v \in C_i} \log p(v|w_i) \tag{1}$$

where C_i are the context tokens, within a specified window of the focus word w_i and $p(v|w_i)$ is the probability of observing context word v given focus word w_i. The skipgram was originally defined using the softmax function:

$$p(v|w_i) = \frac{e^{s(v,w_i)}}{\sum_{j=1}^{W} e^{s(w_i,j)}} \tag{2}$$

where s is a scoring function mapping v and w_i to $\mathbb{R}$. The summation in the denominator is over the entire vocabulary W which makes this formulation computationally inefficient as cost of gradient computation is proportional to W which is quite large ($\sim 10^6$). Mikolov et al. (2013b) hence employ negative sampling to make this computation efficient and robust (Levy et al., 2015) and give better representations for infrequent words[4], which is crucial for the low resource settings. Negative sampling represents the above objective function (Equation 1) using a binary logistic loss as shown below:

$$\sum_{i=1}^{T} \left(\sum_{w_c \in C_i} l(s(w_i, w_c)) + \sum_{w_n \in N_i} l(-s(w_i, w_n)) \right) \tag{3}$$

where N_i are the negative words sampled randomly from vocabulary and l is the log-sigmoid function. The scoring function s is a dot product similarity function given by $s(w_i, w_c) = \mathbf{u}_{w_i}^{\top} \mathbf{v}_{w_c}$ where $\mathbf{u}_{w_i}$ and $\mathbf{v}_{w_c}$ are the embeddings of the focus word and its context word respectively.

3 Subword Representation

Mikolov et al. (2013b)'s model fails to capture internal structure of words and does not generalize for out of vocabulary words that may share morphemes with in-vocabulary words. The problems of this method are particularly salient for

---

[1] `https://github.com/Aditi138/Embeddings`

[2] `https://github.com/Aditi138/Embeddings/tree/master/embeddings_released`

[3] `https://github.com/dmort27/mstem`

[4] `https://code.google.com/archive/p/word2vec/`

graphemes	⟨قارىيالمايدۇ⟩
phonemes	/qarijalmajdu/
morphemes	/qari-jal-ma-jdu/
lemma+tag	qari+Verb+Pot+Neg+Pres+A3sg
gloss	‘s/he can’t care for’

Figure 1: Representations of a word in Uyghur

morphologically rich languages such as Turkish, Uyghur, Hindi, and Bengali. Although, given a large enough training corpus, most or all morphological forms of a lexeme (of which there may be many) could theoretically learn to have similar vector representations, it will be vastly more data efficient if we can take into account regularities of their form to model morphology explicitly. We explore the following methods for doing so:

Orthographic units: Wieting et al. (2016) and Bojanowski et al. (2016) show the utility of character-level modeling by representing the focus word w_i as a set of its character ngrams, denoted by $\mathbf{u}_{w_i} = \frac{1}{|G|} \sum_{g \in G} \mathbf{x}_g$, where G is the set of character ngrams and $\mathbf{x}_g$ is the vector representation of ngram g. Such representations capture morphological information in a brute-force but principled fashion—words that share the same morpheme are more likely to share the same character ngrams than words that do not.

Morphological units: Previous work has found that morphological relationships between words can be captured more directly if embeddings are trained on morphological representations (Luong et al., 2013; Botha and Blunsom, 2014; Cotterell and Schütze, 2015). Avraham and Goldberg (2017) explicitly model lemmas (stems or citation forms) and morphological properties (the sets of which are sometimes called “tags”) for training the word embeddings. Lemmas capture information about the lexical identity of a word and are closely correlated with the semantics of a word; tags capture information about the syntactic context of a word. See Figure 1 for an example. We take inspiration from the above work in adapting these subword units for cross-lingual transfer.

Phonological units: Subword units other than tags might seem to be of no use in closely-related languages with different scripts (such as Serbian and Croatian). Following Bharadwaj et al. (2016), we convert text from its orthographic form into a phonemic representation, stated in terms of the International Phonetic Alphabet (IPA). We then train embeddings on this representation. This means that, roughly speaking, morphemes that sound the same will be represented in the same way across languages.

phoneme ngrams	$x_{<qa} + x_{qar} + \ldots + x_{majdu>}$
lemma	x_{qari}
morphemes	$x_{Verb} + x_{Pot} + \ldots + x_{A3sg}$

Figure 2: Vector representations of a word in Uyghur

4 Cross-lingual Transfer

In this section we discuss in detail both our approaches for cross-lingual transfer along with the relevant baselines.

4.1 Proposed Approach

We propose to use phoneme ngrams, represented using IPA, in addition to the lemma and morphological tags, to enable effective transfer across languages. Tsvetkov and Dyer (2016) demonstrate the effectiveness of projecting words from orthographic space to phonemic space as related languages often share similar phonological patterns. More formally, let P_w be the set of linguistic properties of a word consisting of the phoneme ngrams $(\mathbf{I}_g)$, lemma $(\mathbf{L})$ and individual morphological tags $(\mathbf{M}_m)$. The focus word is then represented as the average sum of its linguistically motivated subword units:

$$\mathbf{v}_{w_c} = \frac{1}{|P_{w_c}|} \sum_{p \in P_{w_c}} \mathbf{x}_p$$

where $\mathbf{x}_p$ is the vector representation of subword unit p of word w_c. The average operation is important to remove any bias towards words having too many or too few subword units. For instance, the Uyghur word in Figure 1 is represented using its phoneme-ngrams ranging from 3-grams to 6-grams, lemma and morphological tags as shown in Figure 2. Avraham and Goldberg (2017) instead encode the different morphological inflections as one tag, so that *Verb+Pot+Neg+Pres+A3sg* would be encoded as $x_{Verb+Pot+Neg+Pres+A3sg}$. We encode each property in a tag separately to avoid data sparsity issues and empirically find this approach to perform better.

We present two training regimes for transferring knowledge from a related language, namely **CT-**

Joint and **CT-FineTune** by explicitly incorporating the subword units. We hypothesize that having word representations of both languages lying in a similar space will aid the low resource language in leveraging resources from the high resource language, including annotations for the downstream task. These two regimes are described below:

CT-Joint: This model explicitly maps the word representations of the two languages into the same space by training simultaneously on both. This is achieved simply by combining the corpora of both the high-resource and the low-resource language and training jointly using the skip-gram objective, discussed above. The central intuition is as follows: once two related languages are placed in the same phonological and morphological space, they will share many subword units in common and this will make joint training profitable. Duong et al. (2016) and Gouws et al. (2015) have previously shown the advantages of joint training and we observe this to be true in our case as well.

CT-FineTune: This model implicitly maps the word representations of the two languages into the same space. The model attempts this by taking the learned continuous representations of the high resource subword units, referred to by $\mathbf{x}^{Hi}_{SWU}$, and uses them to initialize the model for the low resource language. The model is first trained using all subword units on the high resource language and the learned representations are then used for initializing the subword units for the low resource language. To elucidate which pretrained subword helped the most on the low resource language, we use the same model for different experiments, which is trained using all subword units—phoneme-ngrams, lemma and morphological properties. The linguistic intuition behind CT-FineTune is similar to that behind CT-Joint. This idea of transferring parameters from high resource language has been previously explored by Zoph et al. (2016) for low resource neural machine translation which showed considerable improvement.

5 Evaluation

In this section, we first describe the model setup for training word embeddings followed by details on NER and MT experiments.

5.1 Implementation details

We base our model on the C++ implementation of *fasttext*[5](Bojanowski et al., 2016) with modifications as described above.

Data: We represent a word in the training corpus using the format presented by Avraham and Goldberg (2017). For instance, the Uyghur word in Figure 1 is represented as follows: phoneme **ipa:** *qarijalmajdu*, lemma **l:***qari*, and morphological inflections **m:***Verb+Pot+Neg+Pres+A3sg*. We consider phoneme-ngrams ranging from 3-grams to 6-grams and append a special start symbol < and end symbol > to the word. We discard unigrams and bigram ngrams on the assumption that they don't contribute much to the word.

Linguistic properties: We experiment with different subword units for both the transfer setting and the monolingual setting. We use the orthography-to-IPA tool *Epitran* (Mortensen et al., 2018) to obtain the phonemic representations. The lemmas and morphological properties for a word in context are obtained using a rule-based morphological analyzer in such a fashion as to produce tags similar to the high resource language. For Turkish we use the morphological disambiguator developed by (Shen et al., 2016), which in turn is based on an FST-based morphological analyzer developed by Oflazer (1994). For Uyghur, we took a (parser combinator based) morphological analyzer that had been developed for a DARPA LORELEI evaluation and modified it to output part-of-speech tags and to use a property set that was as close as possible to that of the Oflazer Turkish analyzer. The analyzer for Turkish produces 116 inflectional properties and for Uyghur we get 54 properties, of which 64% are shared with Turkish. Unfortunately, we did not have access to existing morphological analyzers for Hindi or Bengali. Many Hindi morphological analyzers exist, but they are not typically released publicly (Malladi and Mannem, 2013; Goyal and Lehal, 2008). We developed our own analyzers using a stemmer-like framework[6] over a span of few weeks (2-3), which gave 8 unique morphological tags for Hindi and 10 for Bengali (for both languages, noun inflection only) of which just 2 were shared with Hindi.

Morphologically speaking, we only use inflec-

---

[5] `https://github.com/facebookresearch/fastText/`

[6] `https://github.com/dmort27/mstem`

tional properties. For most languages, we considered derivational affixes to be part of the stem, since they change the meaning and grammatical category of the word rather than simply expressing syntactic information. An exception to this was Turkish, where the available morphological analyzer segments all affixes off from the root. However, even there we confined our use of morphological properties to inflectional properties. Derivational affixes display scopal behavior; since we wanted to treat the morphological properties of a word as a set, rather than a sequence, we were required to choose this option.

Hyperparameters: During training, we consider context tokens within a window size 3 of the focus word and we sample 5 negative examples from the vocabulary. We chose a window size of only 3 based on the fact that we are working with morphologically rich languages with a relatively high information to token ratio (otherwise a window size of 5 may be more appropriate). Subword units are initialized with uniform samples from $\left[\frac{-1}{dim}, \frac{1}{dim}\right]$ where $dim = 100$. We use the same training regime as Bojanowski et al. (2016). For CT-FineTune, instead of uniform samples we initialize the subword units of the low resource language from the learnt $\mathbf{x}_{SWU}^{Hi}$.

Lang.	Train	Dev	Test
Turkish	3376	1126	1126
Uyghur	1822	240	2448*
Hindi	3974	497	497
Bengali	1908	53	7012

Table 1: Sentences in train/dev/test set for NER. (*Unsequestered set. The full test set has 12,546 sentences.)

5.2 Baselines

For comparison, we train multilingual embeddings using MultiCCA (Ammar et al., 2016) as our baseline. It employs canonical correlation analysis by projecting multiple languages in the same shared space of one language, also referred to as multilanguage space. This method learns linear projections for each language into this common language space using bilingual lexicons. English is used as a common vector space due to availability of corresponding bilingual lexicons between English and each of our languages. For a fair comparison, we run MultiCCA on monolingual embeddings trained with different subword units. We use 100 dimension (Bojanowski et al., 2016) embeddings for English.

For NER, we also compare with Bharadwaj et al. (2016) who use a neural attention model over phonological features and report the best performance for Turkish using transfer from Uzbek and Uyghur, and Mayhew et al. (2017) who use a cheap translation method to translate training data from high-resource language into the low-resource language and report best NER results for Uyghur, as part of the LORELEI program. Our work differs from these primarily on two fronts: a) it is independent of the downstream task and can easily be adapted across various tasks, and b) it doesn't require parallel corpora or bilingual dictionaries. For our monolingual experiments, we compare our proposed approach with models using subword representations—Bojanowski et al. (2016) and Avraham and Goldberg (2017).

5.3 Named Entity Recognition Task

We use state-of-the-art NER architecture (Ma and Hovy, 2016) as our model for evaluation. The task is to identify NEs and categorize them into four types. Since this is a supervised model, the performance is highly contingent on the quality of labeled data. F1 scores are used as the evaluation metric.

5.3.1 Experiments

We conduct the two main sets of NER experiments,

1. Transfer experiments on the low resource languages—Uyghur and Bengali—using Turkish and Hindi as the high resource languages respectively. We show results using both our proposed models, CT-Joint and CT-FineTune.

2. Monolingual experiments on all four languages: Uyghur, Turkish, Bengali and Hindi. We do an ablation study using different combinations of subword units.

These language pairs were chosen partly out of convenience—the data were available to us as part of the DARPA LORELEI program—and partly because they satisfied certain deeper desiderata. Turkish and Uyghur are fairly closely related to one another, as are Hindi and Bengali. Despite this relationship, the members of both pairs are written

Model	subword units	Uyghur	Bengali
CT-Joint	phoneme-ngrams + lemma + morph	55.00	**60.33**
	phoneme-ngrams + lemma	**56.20**	59.63
	phoneme-ngrams	54.90	58.50
	phoneme	51.30	53.75
	char-ngrams + lemma + morph	50.20	55.10
	char-ngrams + lemma	48.20	53.83
	char-ngrams	49.60	52.77
	word	51.80	53.69
CT-FineTune	phoneme-ngrams + lemma + morph	48.60	56.19
	lemma + morph	52.80	57.72
	phoneme-ngrams + lemma	51.00	56.83
	phoneme-ngrams	50.50	57.69
	phoneme	49.20	59.86
MultiCCA (Baseline)	char-ngrams + lemma + morph	41.00	50.63
	char-ngrams + lemma	43.10	50.63
	char-ngrams	45.80	38.06
	word	42.70	45.86

Table 2: Transfer experiments on NER. Metric F1 (out of 100%). Uyghur transfer is from Turkish; Bengali transfer is from Hindi

in different scripts (Roman and Perso-Arabic; Devanagari and Bengali). Finally, all four languages are morphologically rich, especially Turkish and Uyghur. These qualities allow us to showcase the value of embeddings with subword units.

Data Preprocessing: We use data, comprised of unlabeled corpora, English bilingual dictionaries, annotations, from the Linguistic Data Consortium (LDC) language packs—Turkish and Hindi [7], Bengali[8], from which we generate train-dev-test splits. Uyghur data was released as part of LoReHLT16 task, organized by NIST [9] under the aegis of DARPA, and training annotations were acquired using native speakers as part of the task. For Uyghur we evaluate on an unsequestered set consisting of 199 annotated evaluation documents, released by NIST. For Turkish, Hindi and Bengali, we create our own train-dev-test splits (Table 1). The exact documents from which Mayhew et al. (2017) and Bharadwaj et al. (2016) created their test set is not apparent. The Uyghur corpus has 27 million tokens and the Turkish corpus has about 40 million tokens. Although Bengali is widely-spoken and the unlabeled corpus contains more than 140 million tokens, there are very few named entity annotations available, making it a low-resource language for the purposes of this exercise. To have a fair experimental setup across language pairs, we sub-sample the Bengali and Hindi corpora to have comparable corpus sizes with Uyghur and Turkish respectively. We also up-sample the low resource data for both unlabeled corpora and NER annotations, so the model doesn't become biased towards the high resource language.

NER model setup: We train the model using 100-dimensional word embeddings, pre-trained using the above discussed strategies, and use hidden dimension of size 100 for each direction of the LSTM. Stochastic gradient descent was used as the optimizer with a learning rate of 0.015. Dropout of 0.5 was used in the LSTM layer to prevent over-fitting. Uyghur and Turkish were trained for 100 epochs, Bengali and Hindi converged after 70 epochs.

5.3.2 Results and Discussion

Transfer Experiments: From Table 2 we note that our CT-Joint model trained with phoneme-ngrams, lemma, and morphological tags outperforms the MultiCCA baseline by a significant margin. MultiCCA strongly depends on

---

[7] LDC2014E115,LDC2017E62,`http://www.cfilt.iitb.ac.in/iitb_parallel/`

[8] LDC2017E60, LDC2015E13

[9] `https://www.nist.gov/`

Model	subword units	Turkish	Uyghur	Hindi	Bengali
Ours	Char-ngrams + Lemma + Morph	68.06	**52.50**	73.15	**52.77**
	Char-ngrams + Lemma	**68.61**	52.40	73.37	52.09
	Char-ngrams + Morph	67.97	47.80	**73.46**	52.06
prop2vec	Word + Lemma	66.52	46.00	71.82	50.03
	Word + Morph	64.45	46.00	71.52	49.27
	Word + Lemma + Morph	68.46	47.70	70.51	48.16
fastText	Char-ngrams	66.81	50.80	72.67	52.10
word2vec	Word	62.85	46.80	72.04	49.83
Random	No embedding	58.94	31.30	59.89	21.25

Table 3: NER results for monolingual experiments. Metric F1 (out of 100%)

Model	Uyghur* (unseq.)	Uyghur*	Turkish	Bengali
Ours	**56.20**	**56.00**	**68.61**	**60.33**
Bharadwaj et al. (2016)	–	51.2	66.47	–
Mayhew et al. (2017)	51.32	55.6	53.44	45.70

Table 4: Comparison with previous work using data released by DARPA LORELEI. Metric F1 (out of 100%)
*Official NIST scores.

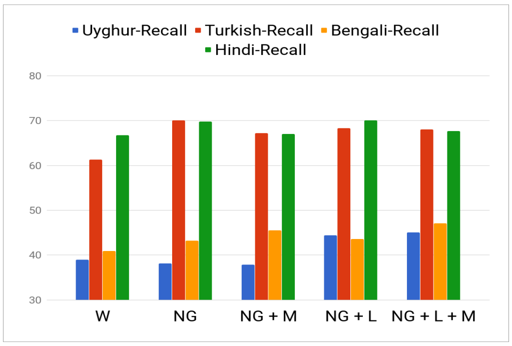

Figure 3: Recall for all languages (monolingual) **W**:Word, **NG**:Char-ngrams, **L**:Lemma, **M**:Morph

bilingual dictionaries which is possibly why it performs poorly in our low resource setting, where these dictionaries are not of high quality. The advantage of phoneme-ngrams over char-ngrams is quite apparent here. *phoneme-ngrams+lemma+morph* performs +5.2 F1 points better than *char-ngram+lemma+morph* for both Uyghur and Bengali, and similar increase is observed across other combinations, the only exception being the *phoneme* case for Uyghur which performed -0.5 F1 with respect to its counterpart *word*.

We find CT-Joint to be consistently better performing than CT-FineTune. Interestingly, the performance of CT-FineTune model converges to the monolingual performance. We hypothesize that the model forgets the pre-trained subword units as training progresses.

For CT-FineTune, the column *subword units* in Table 2 refers to the subword units which were pre-trained on a high resource related language. For example *lemma + morph* means lemma and morph embeddings are first pre-trained on the resouce-rich language and then used to initialize the respective lemma and morph representations for the low resource language.

Monolingual Experiments: Table 3 shows our results on all languages. We get +5.8 F1 points for Turkish, +4.8 F1 for Uyghur, +0.8 F1 for Hindi and +0.7 F1 for Bengali over the existing methods. We observe that a combination of character-ngrams, lemma and morphological properties gives the best performance for Uyghur and Bengali. Adding *morph* hurts in Turkish, in contrast to Hindi, where it helps. Section 5.3.3 discuses plausible reasons for this.

We report official NIST scores on the full evaluation set for Uyghur, as part of LORELEI Offical Retest. Additionally, we compare our results with

Model	subword units	Uyghur	Bengali
CT-Joint	phoneme-ngrams + lemma + morph	23.04	7.88
	phoneme-ngrams + lemma	23.24	7.62
	phoneme-ngrams	23.25	7.45
CT-FineTune	lemma + morph	23.71	7.58

Table 5: Transfer experiments for MT. Metric: BLEU. Uyghur transfer is from Turkish; Bengali transfer is from Hindi

Model	subword units	Uyghur	Bengali
Ours	Char-ngrams + Lemma + Morph	23.59	**7.96**
	Char-ngrams + Lemma	**23.91**	7.77
	Char-ngrams + Morph	23.27	7.88
fastText	Char-ngrams	23.24	7.91
word2vec	Word	23.31	6.64
Random	No embedding	23.51	6.23

Table 6: MT results for monolingual experiments. Metric: BLEU

the best results reported on the same LORELEI dataset. Results are seen in Table 4.

5.3.3 Error analysis

We plot recall curves for all languages. As seen in Figure 3, adding subword units boosts the recall consistently across all languages, more so for Uyghur. For Turkish, *lemma* performs better than *lemma+morph*, perhaps because the morphological analyzer outputs so many redundant properties which reduce the distance between words that are not particularly similar. In contrast, *morph* helps and *lemma* hurts in Hindi, perhaps because the morph analyzer outputs only a small number of highly informative properties, but is a poor general-purpose lemmatizer.

We analyze our results for Uyghur language, as it was part of the LORELEI challenge and presents a situation close to a real-life application. We base our analysis on the unsequestered set since annotations for full test data are not released. There are 1,341 NE's in this set, 396 of which are covered by the word embeddings when trained with just monolingual corpus. One obvious advantage of jointly training with a resource-rich corpus is that coverage of NEs increases, as validated in our case where jointly training with Turkish corpus adds 114 more NEs.

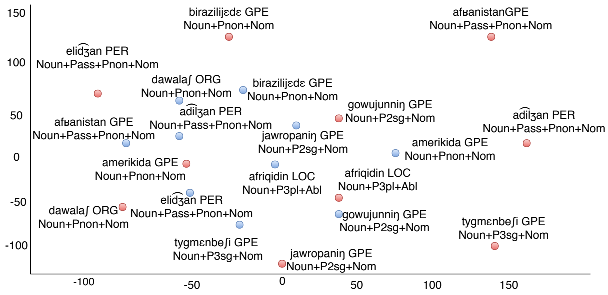

Figure 4: Two-dimensional PCA projection of select NEs from word embeddings for Uyghur—**CT-Joint** model trained with *phoneme-ngrams+lemma+morph* (blue) and monolingual model trained with *char-ngrams+lemma+morph*

Figure 4 shows ten named entities in two different embeddings (CT-Joint: *phoneme-ngrams+lemma+morph* and monolingual: *char-ngram+lemma+morph*). The difference is striking—in the monolingual condition, the NEs are widely dispersed, but in the bilingual condition, the NEs cluster together. This suggests that phonologically-mediated transfer through Turkish is resulting in embeddings in which NEs are close to one another, relative to monolingual Uyghur embeddings.

5.4 Machine Translation Task

In addition to NER, we test the performance of our proposed approaches on the MT task to test generality of our conclusions. We use XNMT toolkit

(Neubig et al., 2018) to translate sentences from the low-resource language to English. We run similar transfer and monolingual experiments as done for NER. Due to space limitations, we use select subword combinations for the experiments, details of which can be found in Appendix. BLEU is used as the evaluation metric.

From Table 6, we observe that the combination of *character-ngrams* and *lemma* performs the best for Uyghur (+0.1) and the combination of *character-ngrams*, *lemma* and *morph* gives the best performance for Bengali (+1.7), over the *word* baseline, which demonstrates the importance of subword units for low-resource MT as well. One likely reason that the combination of character-ngrams and lemmas consistently show the best performance is that, together, they capture lexical similarity, which is more important to translation than the syntactic information captured by morphological inflection ("morph"). However, experiments using CT-Joint and CT-FineTune (Table 5) do not follow the same trend as that of NER. We hypothesize that this is because the MT models were trained on a training set that did not have translation pairs from the high resource language. As Qi et al. (2018) note, when training MT systems on a single language pair, it is less necessary for the embeddings to be coordinated across the languages.

6 Related Work

Word Embedding Models: Most algorithms for learning embeddings take inspiration from language modeling (Bengio et al., 2003), motivated by distributional hypothesis (Harris, 1954), and employ a shallow neural network to map the words into a low dimensional space. Pennington et al. (2014) built over the above local context window model by combining it with global matrix factorization (Levy and Goldberg, 2014). Recently, Peters et al. (2018) show significant gains across various tasks by learning word vectors as hidden states of a deep bi-directional language model. This was originally conceived for resource-abundant languages, hence it is as-of-yet unclear how generalizable they are to low-resource settings.

Modeling subword information: Various methods have validated the importance of modeling subword units in downstream tasks. Xu et al. (2016); Chen et al. (2015) experiment at the character level whereas Luong et al. (2013) use morphemes as a basic unit in recursive neural network (RNN) to get morphologically-aware word representations. Xu and Liu (2017) incorporate the morphemes' meanings as part of the word representation to implicitly model the morphological knowledge.

Transfer learning: Most recent works using transfer in low resource setting are coupled tightly with the downstream task. Jin and Kann (2017) use morpheme units for cross-lingual transfer in a paradigm completion task using sequence-to-sequence models. Tsai et al. (2016) employ a language-independent method for NER by grounding non-English phrases to English Wikipedia. Interestingly, Kim et al. (2017) use separate encoders for modeling language-specific and language-agnostic features for part-of-speech (POS) tagging, and make use of no cross-lingual resources.

7 Conclusion

In this paper, we explored two simple methods for cross-lingual transfer, both of which are task-independent and use transfer learning for leveraging subword information from resource-rich languages, especially through phonological and morphological representations. CT-Joint and CT-FineTune do not require morphological analyzers, but we have found that even a morphological analyzer built in 2-3 weeks can boost performance and is a worthwhile investment of resources. Preliminary evaluation on a separate task of MT reconfirms the utility of subword units and further research will reveal what these learned subword representations can contribute to other tasks.

Acknowledgement

This work is sponsored by Defense Advanced Research Projects Agency Information Innovation Office (I2O). Program: Low Resource Languages for Emergent Incidents (LORELEI). Issued by DARPA/I2O under Contract No. HR0011-15-C0114. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 937–947.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Oded Avraham and Yoav Goldberg. 2017. The interplay of semantics and morphology in word embeddings. *CoRR*, abs/1704.01938.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1462–1472.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning*, pages 1899–1907.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *IJCAI*, pages 1236–1242. AAAI Press.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 1287–1292, Denver, Colorado. Association for Computational Linguistics, Association for Computational Linguistics.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. *arXiv preprint arXiv:1606.09403*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *International Conference on Machine Learning*, pages 748–756.

Vishal Goyal and Gupreet Singh Lehal. 2008. Hindi morphological analyzer and generator. In *2008 First International Conference on Emerging Trends in Engineering and Technology*, pages 1156–1159.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. *Proceedings of ACL-08: HLT*, pages 771–779.

Harald Hammarström, Sebastian Bank, Robert Forkel, and Martin Haspelmath. 2018. *Glottolog 2.0*. Max Planck Institute for the Science of Human History, Jena. Available online at http://glottolog.org, Accessed on 2018-05-21.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Huiming Jin and Katharina Kann. 2017. Exploring cross-lingual transfer of morphological knowledge in sequence-to-sequence models. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 70–75.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530. Association for Computational Linguistics.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

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

Deepak Kumar Malladi and Prashanth Mannem. 2013. Statistical morphological analyzer for hindi. In *International Joint Conference on Natural Language Processing*, pages 1007—1011, Nagoya, Japan.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2536–2545.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

David R. Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision G2P for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*. Publisher: John Benjamins Publishing Company.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, John Hewitt, Rachid Riad, and Liming Wang. 2018. XNMT: The extensible neural machine translation toolkit. In *Conference of the Association for Machine Translation in the Americas (AMTA) Open Source Software Showcase*, Boston.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. *Literary and Linguistic Computing*, 9(2):137–148.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535. Association for Computational Linguistics.

Sebastian Ruder. 2017. A survey of cross-lingual embedding models. *arXiv preprint arXiv:1706.04902*.

Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, and Chris Dyer. 2016. The role of context in neural morphological disambiguation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2017. S-net: From answer extraction to answer generation for machine reading comprehension. *arXiv preprint arXiv:1706.04815*.

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Trans. on Knowl. and Data Eng.*, 28(2):496–509.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 219–228.

Yulia Tsvetkov and Chris Dyer. 2016. Cross-lingual bridges with models of lexical borrowing. *J. Artif. Int. Res.*, 55(1):63–93.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*.

Jian Xu, Jiawei Liu, Liangang Zhang, Zhengyu Li, and Huanhuan Chen. 2016. Improve Chinese word embeddings by exploiting internal structure. In *HLT-NAACL*.

Yang Xu and Jiawei Liu. 2017. Implicitly incorporating morphological information into word embedding. *arXiv preprint arXiv:1701.02481*.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. 2018. Refining word embeddings using intensity scores for sentiment analysis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(3):671–681.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*.

A Computational Exploration of Exaggeration

Enrica Troiano and **Carlo Strapparava** and **Gözde Özbal** and **Serra Sinem Tekiroğlu**
FBK-Irst, Trento, Italy
`enrica.troiano@gmail.com, strappa@fbk.eu, gozbalde@gmail.com, tekiroglu@fbk.eu`

Abstract

Several NLP studies address the problem of figurative language, but among non-literal phenomena, they have neglected exaggeration. This paper presents a first computational approach to this figure of speech. We explore the possibility to automatically detect exaggerated sentences. First, we introduce HYPO, a corpus containing overstatements (or hyperboles) collected on the web and validated via crowdsourcing. Then, we evaluate a number of models trained on HYPO, and bring evidence that the task of hyperbole identification can be successfully performed based on a small set of semantic features.

1 Introduction

Very often, when humans recount their experiences, describe objects or verbalize ideas, they exaggerate in some respects. "*All you need is love*", sang The Beatles in one of their most iconic recordings; many centuries before, Shakespeare had Romeo say that there was "*more danger in Juliet's eyes than in twenty swords*"; and Maximus Meridius, a Roman commander in the movie The Gladiator, incited his legion to battle by the words "*At my signal, unleash hell*". Exaggerating, like in these quotes, is a linguistic tendency that unfolds in a variety of situations. From TV advertisements to debates in politics, our verbal productions are infused with statements, or more precisely overstatements, that puff up facts.

The study of exaggeration dates back to ancient Greece, and its centrality in the spectrum of figures of speech has long been established. Empirical evidence has shown that it is the most used rhetorical device, second only to metaphor (Kreuz et al., 1996), but in comparison to its kin tropes, this phenomenon represents an under-researched field in NLP. In effect, the problem of automatic detection of exaggerations (or hyperboles) has been quite dismissed. On the one hand, determining that a sentence speaks an excess is often a complex and context dependent act, on the other, no resource has ever been made available to specifically address the figure. To overcome this issue, we build HYPO, the first dataset focused on exaggeration, and introduce the task of automatic hyperbole detection. Specifically, the goal of this work is to find an automatic solution to establish if the topic of a sentence is aggrandized or, on the contrary, is presented as it is in reality.

Effective identification of overstatements would benefit theoretical and applied approaches to natural language. It can represent both a deterministic strategy to test hypotheses about exaggerations, as well as a boost for dialogue systems, information extraction, and all such AI endeavors to understand how humans talk. In fact, automating the intuition that a text says more than is true can meet a number of useful applications. Endowed with this ability, automatic tools might support our recognition of some kind of fake news, which blow information out of proportion; they might ascertain whether the promises of testimonials and politicians constitute a form of puffery; they might even be exploited for health-related objectives, as auxiliary tools for the diagnosis of psychological conditions (e.g., depression, narcissistic disorder) which use exaggerations to depict extremely unnuanced views of the world.

To tackle the problem, the paper starts with a synthesis of past theoretical work, which provides a guideline for building HYPO. Data was collected by a manual crawl on the web and its quality was tested via crowdsourcing. We then use the overstatements of HYPO for learning to classify a sentence as hyperbolic or literal. Experimental results show that a number of models can solve the problem with above chance accuracy, thus confirming that the task is feasible and that HYPO can be ex-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3296–3304
Brussels, Belgium, October 31 - November 4, 2018.

ploited for similar purposes in future work.

2 Related Work

A long lasting challenge in NLP is to reason about figurative language, and in fact, scientists have achieved groundbreaking results in the study of a range of figures. Nevertheless, they have given scarce attention to hyperbole – if any at all. Only relatively recently has this trope become an object of interest of corpus linguistics, and some insights on its nature have been delivered by statistics extracted from written and spoken sources.

The bulk of research points out that a hyperbole is not perceived nor produced as a lie. It makes things sound bigger than they actually are, but with the goal of emphasizing them, of clarifying ideas, of adding interest and humor to the conversation (Roberts and Kreuz, 1994). In other words, speakers exaggerate to accentuate the one element, account or aspect of reality that is important to them, in order to make others see truth from their perspective (Ritter, 2012). Hearers, on their part, suspend disbelief towards the literal content of the hyperbolic expression, which is mendacious, and recover its intended meaning, which is instead in accord with reality. Basically, to find this non-literal content, they reformulate a hyperbole as a paraphrase devoid of figures of speech, which is softer than the hyperbole itself and still conveys a strong concept (Fogelin, 1988).

Another point on which the literature agrees is that hyperboles are pragmatic acts. They may entirely rely on the concrete context of their production, like in the statement "It took ages to build the castle", which carries both an exaggerated and a literal sense, depending on whether it refers to a playful sand castle or to a strong walled construction. McCarthy and Carter (2004) endorse this view based on the need of contextual, extra-linguistic information, and claim that understanding an overstatement is to perceive a 'contrast' between the expression itself and its referent, i.e. a discrepancy between reality as it is and as is described.

Focusing on this contrast, Cano Mora (2010) provides a handcrafted framework of the semantic fields in which it tends to arise. Her taxonomy develops along a quantitative and a qualitative dimensions, suggesting that any exaggeration inflates either a measurable or a subjective property of the topic of discourse. However, given that any type of speaking non-literal is a departure from actual facts, the notion of 'contrast' can be referred to figures of speech in general. With this regard, an important point is made about the peculiarities of hyperbole (Colston and O'Brien, 2000; Carston and Wearing, 2011). Exaggerating presents facts with a greater degree, e.g. being bigger, more desirable, so it prompts a contrast of magnitude. Other tropes engender a contrast of kind for they portray an object via some types of qualities that it does not actually have. Metaphors, for instance, borrow those qualities from the vehicle term.

Therefore, the major contribution yielded by past corpus research is to have made explicit a few typically hyperbolic characteristics, which will come handy for the construction of our corpus.

3 HYPO

Hyperboles over-blow the truth, by augmenting or down-toning the qualities of the referent of discourse, e.g., an event, an object, a person, etc. If a referent has a feature X, a hyperbole presents it as having more of that X than warranted by reality (Claridge, 2011): what is big becomes bigger (e.g. [1]) and what is small becomes smaller (e.g. [2]). This causes a discrepancy between the linguistic expression and the actual state of affairs, and provides the former with a non-literal meaning.

[1] Her morning jog turned into a marathon.

[2] I'm ready in no time.

Drawing on example from (McCarthy and Carter, 2004), we use this 'contrast' or 'counter-factuality' as the central condition that makes a statement hyperbolic. It can be noticed that the extent of the contrast corresponds to different types of overstatements, which can go from slight distortions of real situations (e.g. [1]), to representations of absurd worlds (e.g. [2]). In the second instance, the contrast is grasped with no need for pragmatic information, but other types of overstatements can be perceived as such only if one knows the contextual setting in which they were produced. In those cases, readers evaluate the credibility of the sentence, that is, they confront how the sentence presents property X of the topic to how they expect it to be, given their previous experience and their common knowledge about the world (Ferré, 2014). For instance, although one cannot verify if proposition [1] is in contrast with the truth, hardly would one believe that a routine jog has equaled a marathon.

[3] I won't wait for you: it took you centuries to get dressed.

[4] That bag is to die for!

The above examples picture some peculiar traits of hyperboles. First, they require the possibility to present concepts with different intensities, by shifting ideally towards the extremes of a continuous semantic scale. Second, there are two scales on which this shift occurs (Cano Mora, 2010). One is quantitative, and it allows to inflate objective, measurable features (e.g. time [3]), while the other is qualitative and serves to augment subjective characteristics (e.g. appreciation [4]).

Another defining element of hyperboles is their emotional bent. When we exaggerate, we accentuate some pieces of evidence that supports our thoughts and perspectives, while downplaying competing alternatives. We present an extravagant view of reality with the goal of manifesting our positive or negative involvement towards it. This might appear obvious for qualitative hyperboles such as [4], for the property they enlarge is subjective, but quantitative overstatements have an evaluative trait as well. As an example, the overstatement [3] frames an antipathetic position by increasing an impartial, unemotional measure.

To sum up, there are three criteria that were deemed to characterize a hyperbole. They are the non-literal meaning, the upsurge on a semantic scale and a connotative trait. Hyperbolic candidates entered the corpus only if they were considered to have a figurative content rather than a deceptive one, if the figurative component emerged as a contrast of magnitude with reality, and if the contrast seemed to color reality with an evaluative tone.

3.1 Dataset Description

HYPO is a collection of 709 objects that comply with the three definitional requirements, and that can be deemed exaggerated without knowing the context in which they were originally produced[1].

Sentences can be semantically grouped into basic and composite items, which respectively preserve and modify the semantic domain of their referent (Claridge, 2011). For instance, [5] is basic, because the intended and literal meanings have the same domain (i.e. size) though they differ in magnitude. Instead, the exclamation in [6] is composite, as the intended meaning concerns human emotions but is expressed with a quality inherent to stones. This results both in a metaphor, since it describes a psychological state by means of a domain transfer, and in a hyperbole, because among all the things that have limited movement capabilities, it pinpoints the stones, which are completely motionless.

[5] The house is the size of a postage stamp.

[6] First I was afraid, I was petrified!

[7] I avoid crowded places like the plague.

[8] She agreed with every word of my argument.

Other hyperboles arise from the combination with figures different than metaphors, such as comparatives [7] and synecdoches [8].

From a syntactic point of view, exaggerations in the data either spread over phrases, or pop up as single words that belong to any sorts of grammatical classes. One peculiar instance is that of Extreme Case Formulations (ECFs), which are adjectives ("absolute, whole"), quantifiers and nouns ("all, no, everybody"), adverbs ("always, never"), phrases ("as good as it gets") and superlative constructions ("ever, the most") that engender extreme utterances by evoking the highest degree on a semantic scale (Pomerantz, 1986). Although ECFs might not be "heard as absurd or counterfactual and often display a degree of conventionality" (McCarthy and Carter, 2004), they share the three fundamental characteristics of the trope: they build expressions around semantic acmes, which lessens their credibility and prompts hearers to grasp their non-literal content, namely, one that expresses an evaluation. Some ECFs are at work in the following propositions.

[9] This is the best pizza in history.

[10] Everybody loves chocolate.

Within the above groups lie qualitative [9] and quantitative [5] hyperboles, creative [7] and conventional [10] ways of exaggerating. Notably,

---

[1] As an example, rather than "It took ages to build the castle", which can be read as hyperbolic or literal depending on contextual information, our dataset would be appended with "It took ages to build the castle. After a few minutes, my little brother had already destroyed it!", which is non-literal regardless of the context of its production.

such categories can criss-cross (Peña and Ruiz de Mendoza, 2017) like in "I'd bake cakes again and again and again", an instance of a quantitative hyperbole that encompasses multiple tokens, as it is merged with a polysyndeton.

4 Resource Construction

The collection of data for HYPO proceeded along two lines. One involved our own effort to invent overstatements. The other consisted of a manual crawl of the Internet and targeted the scripts of animated cartoons, advertisements, love letters, clickbait headlines, as well as other material that allegedly fulfilled the communicative objectives of exaggerations (see Roberts and Kreuz, 1994), and therefore was likely to include instances of this figure of speech. A total of 804 overstatements was obtained which comply with the definition presented in Section 3.

These candidate items underwent a validation stage on the Crowdflower/Figure Eight platform[2], where their hyperbolicity was judged by external annotators, and the final corpus was determined. Ideally, our microtask was feasible by any English speaker. It asked workers to read 10 sentences, one of which was a test item, and to answer 6 questions aimed at determining if and how the texts overstate their topic. Gold units consisted of 42 hyperboles, together with 50 non-hyperbolic statements to ensure that raters would not learn to mark all items as exaggerated. Each of the 854 sentences was expected to be judged by 5 annotators provided with minimal instructions on the subject at hand.

In particular, the annotators decided if a sentence contained a hyperbole (*Question 1*), and if so, they highlighted the words that caused its hyperbolic bent (*Question 2*), paraphrased it without exaggerating (*Question 3*), classified the hyperbole as quantitative or qualitative (*Question 4*), rated the degree of the exaggeration (*Question 5*) and established whether it was creative or conventional (*Question 6*).

With the first question, candidate items for HYPO were validated. In the second, annotators selected the smallest number of words (or minimal units) that convey the exaggeration, like "million" in "I've told you a *million* times". This assignment enabled us to collect the units perceived as hyperbolic, and to use them for creating another corpus, that is, a dataset of sentences which include the hyperbolic tokens of HYPO, but without their hyperbolic twist (see 4.2). Similarly, the rationale beyond the third question was the construction of a corpus devoid of exaggerations. People had to paraphrase the sentences without exaggerating. They were encouraged to reword or delete the portion of text highlighted in *Question 2*, such that the resulting sentences would have differed from the original ones only for the absence of hyperbolic tokens. Basically, we explored the idea that understanding a hyperbole requires a sort of readjustment, where concepts are toned down, but still to a high degree. For instance, an overstatement such as "I've told you a *million* times" could have been modified as in "I've told you a lot of times".

In *Question 4*, the annotators decided if a hyperbole belonged to a quantitative or a qualitative class, according to whether it exaggerated something which is objective and quantifiable with a number (e.g. "He *never* says no") or which is subjective and unmeasurable (e.g. "He seems to *come from another world*!"). This was meant to label sentences with one of the two semantic dimensions along which a concept is aggrandized, as proposed in the literature (Cano Mora, 2010). As for *Question 5*, an exaggeration was rated as either 'Possible', if it denoted an extreme but conceivable situation (e.g. "I avoid crowded places like the plague"), or 'Impossible', when it described an absurd or paradoxical situation (e.g. "My father always works"). We collected these scores to test the hypothesis that sentences with a higher hyperbolic degree are classified more accurately in the experiment because their excess is easier to detect. In the sixth question, exaggerations were judged either as conventional ("He died of envy") or creative ways to express an idea ("It got so cold that all spoken words froze solid"). The answers were expected to show if there is a correlation between conventional and 'Possible' hyperboles of *Question 5* (i.e. if conventional items are perceived as softer exaggerations)[3].

4.1 Evaluation Measures

Only 750 sentence received 5 reliable judgments. Therefore, to evaluate the quality of the results, the

---

[2] https://www.figure-eight.com

[3] For the quality check, we only required the raters to correctly judge questions 1 and 2. Question 3 was not included as no unique answer exists. Questions 4, 5 and 6 were also not included because, due to their difficulty, we expected some amount of disagreement among the raters.

inter-annotator agreement was observed on those. We calculated two measures of agreement for the task of recognizing hyperboles (Question 1). First, we measured the raw agreement (RA) that is the proportion of items with unanimous judgments (i.e. the sentences marked as either hyperbolic or non-hyperbolic by all the workers) out of the total number of items with 5 judgments. The final score showed that annotators agreed in 58.5% of the cases, which we considered an acceptable metric for the reliability of the annotation process.

That people had the same understanding of hyperboles in more than half the items seemed reasonable, because it can be challenging to establish if a statement is extreme, especially for non-expert workers. In fact, contextual knowledge, which in everyday situations enables interlocutors to grasp the inflation of a sentence, is neglected in an experiment that revolves around isolated textual passages. Moreover, the RA measure is extremely sensitive to the inconsistency of results, as the presence of one incongruous annotation out of 5 is enough to lower the outcome.

Therefore, we used a second measure to take into consideration the difficulty of having 5 people agree. Since items were labeled by more than two annotators, we computed a pairwise agreement, as suggested by (Artstein and Poesio, 2008). We exploited the Observed Agreement (A_O) as defined in Fleiss's κ, which consists of the proportion of items on which pairs of annotators agree out of the total number of judgment pairs. Also in this case we observed a substantial agreement (A_O=80.2%) corroborating the quality of our results.

4.2 Final Datasets

The logic behind the validation of our resource was to obtain adequate information to train a classifier of hyperboles. Given the consistency of annotations, we selected the sentences to be inserted into the corpus of hyperboles and built two control datasets. The outcome is a set of hyperboles (HYPO) with two types of non-figurative counterparts, that is, their paraphrases, and literal sentences that include the words that are hyperbolic in HYPO, but with a literal connotation. By means of illustration, while HYPO would incorporate a sentence like "Her morning jog turned into a marathon", the Paraphrases corpus and the Minimal Units Corpus would respectively include "Her morning jog turned into a very long run" and "There is a marathon in the city today".

HYPO The sentences in HYPO are the items that at least 3 annotators out of 5 perceived as exaggerations. From the total of 854 judged sentences, we discarded the non-hyperbolic units, those that received less than 5 judgments and those which were not classified as hyperbolic by the majority of annotators. The final result is a corpus of 709 hyperboles[4].

Paraphrases Corpus The corpus of paraphrases was created by choosing one paraphrase for each hyperbolic sentence, i.e. the one that introduced the smallest change in the syntax and the semantics of the corresponding hyperbole. The result is a collection of 709 sentences which say the same things as their hyperbolic counterparts but are devoid of exaggerations.

Minimal Units Corpus To build this dataset, we used the minimal units selected by the annotators in Question 2. We wanted to end up with sentences which are not hyperbolic and which contain the same hyperbolic words of HYPO.

For each exaggeration we considered the tokens that were selected by the majority of annotators. They either consisted of a single term, or a phrase, or long-distance words. Then, we extracted sentences containing these tokens from sources such as the WaCKy corpus, a dump of the English Wikipedia, whose editorial criteria force the entries to be neutral and verifiable, and thus, unlikely to incorporate excessive statements. Whenever this approach did not produce results, we ran a Google search. The final corpus contains 698 non-hyperbolic sentences.

5 Experiment

The task of hyperbole detection is formulated as a supervised learning problem, and specifically, as a sentence-based classification with two classes, that is, hyperbolic and literal.

5.1 Features Set

To describe sentences, we define two sets of features that incorporate the qualitative and quantitative criteria proposed in the literature[5].

---

[4] The dataset will be made publicly available under a Creative Commons License for free cultural works.

[5] These terms do not refer to the semantic scale on which the shift of meaning occurs, which vary depending on the topic of hyperboles, but to the counterfactuality and the connotative traits that characterize them all.

All of the features take on values in the interval [-1,1], and some are encoded in multiple ways to ensure richer information. As for the quantity group, the notion that hyperboles say *more of* X when X is the case is decomposed into two features, i.e. imageability and unexpectedness, while the qualitative marker of hyperboles, or the view that they shape a speaker's perspective about X, is rendered by the polarity, the subjectivity and the emotional strength of sentences.

Imageability is the degree to which a word can evoke a mental image. Speakers hyperbolize to convey meanings with strength, and we assumed that such a goal might be backed by a highly picturable vocabulary. This feature is extracted from the resource of Tsvetkov et al. (2014), who propagated the imageability ratings of the MRC psycholinguistic database to 150.114 terms. For each sentence, we averaged the imageability values of all its words.

Unexpectedness refers to the fact that hyperboles are less predictable expressions than literals. Basically, minimal hyperbolic units modify the real characteristics of X, and in this sense, they are incoherent with the rest of discourse about X: they are out of context, and come unexpected to the hearers of overstatements.

We conjecture that word vectors may capture if an expression is being used "unexpectedly" because they encode the contexts in which terms frequently occur, as well as contrasts and similarities among their meanings. In fact, according to the Distributional Hypothesis (Harris, 1954), lexical items appearing in similar contexts tend to have similar meanings. Our expectation is that the words of a figurative sentence carry less similar meanings compared to those of a literal instance, and hence, that their vectorial representations turn out to be more distant.

For every sentence in the dataset, we map its words onto both the pre-trained vectors of Mikolov et al. (2013), obtained from the Skip-gram model, and the GloVe vectors by Pennington et al. (2014). Then, we consider the cosine distance between all possible word pairs to score their semantic similarity. The unexpectedness feature of a sentence is found in two ways: as the average similarity among all of its word pairs, and as the lowest of those pair similarities. Both measures are separately computed with Skip-Gram and GloVe vectors, resulting in 4 scores.

Polarity corresponds to the sentiment of a statement. It is extracted through both TextBlob (Loria, 2014), a system that directly scores sentences, and SentiWords (Gatti et al., 2016), which lists polarity values for 155k POS-tagged lemmas. With this lexicon, we find the sentiment of a sentence by averaging the sentiment of all of its word lemmas.

Subjectivity, found with TextBlob, which specifies if a statement conveys an objective information or a personal opinion. Ideally, the higher the absolute value of polarity in the range [0,1], the more subjective a sentence.

Emotional intensity stands for the strength of sentiment. It captures if the utterer of a sentence is sympathetic towards the thing being said, while quantifying the emphasis with which such position is communicated. The scores are obtained from VADER (Hutto and Gilbert, 2014).

5.2 Experimental Setup and Results

A simple pre-processing is applied to HYPO, the Paraphrases and the Minimal Units corpora prior to the experiment. We remove the stopwords, for which the quality scores are not available. The resulting sentences are represented by 9-dimensional vectors, in which 5 entries stand for the quantity features and 4 belong to the quality group.

For classification, we experiment with various algorithms, such as Logistic Regression (LR), Naive Baies (NB), k-Nearest Neighbors (KNN), Decision Trees (DT), Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA). The models are evaluated by comparing their average accuracy on a 10-fold cross-validation against three baselines: a classifier that randomly associates inputs to the hyperbolic and literal labels, one that classifies sentences using the 300 features of the pre-trained Skip-gram representations, and a third that relies on the 300-dimensional GloVe vectors trained by their authors. Sentences fed to the baseline classifiers were represented as the sum of their word vectors[6].

---

[6] Other types of compositional methods yield results analogous to those in Table 1 and Table 2.

	Baseline1	QQ	Skip-gram	GloVe	Skip-gram+QQ	GloVe+QQ
LR	.50	.64	.68	.66	**.72**	.69
KNN	.50	.63	.47	.43	.52	.48
NB	.50	.66	.69	.66	.69	.68
DT	.50	.60	.54	.53	.55	.54
SVM	.50	.64	.15	.62	.63	.64
LDA	.50	.61	.67	.65	.68	.67

Table 1: Mean accuracy of 10-fold cross validation in the Hype-Par setting. Column QQ shows results for our handcrafted quality and quantity features; the last two columns are concatenations of QQ with the Skip-gram and GloVe baseline features.

	Baseline1	QQ	Skip-gram	GloVe	Skip-gram+QQ	GloVe+QQ
LR	.50	.44	.60	.58	.61	.59
KNN	.50	.47	.49	.48	.50	.51
NB	.50	.50	.64	.64	.62	**.68**
DT	.50	.51	.51	.52	.56	.54
SVM	.50	.02	.09	.59	.13	.60
LDA	.50	.52	.54	.34	.57	.56

Table 2: Mean accuracy of 10-fold cross validation in the Hype-Min setting.

Cross-validation is conducted in two settings. In one (Hype-Par), the non-hyperbolic sentences are paraphrases, in the other (Hype-Min), literal data come from the Minimal Units Corpus. The results are shown in Table 1 and Table 2.

While in the Hype-Min setting the performance is not satisfying, estimators achieve above chance accuracy using paraphrases as literal inputs. In fact, in Table 1, the accuracy scores based on quantity-quality vectors (QQ column) suggest that our handcrafted features are actually useful for detecting hyperboles. Therefore, to gain further insight on their informativeness, we conduct a recurrent feature ablation and observed how different subsets affect predictions. Figure 1 illustrates that 5 features can maximize the accuracy of LR. SVM and LDA behave the same with a set of the same size, and they all assign high weights to imageability, unexpectedness and subjectivity. The three models become comparable to, and yet do not outperform, the second and third baselines.

In the attempt to improve the models' description of the data, we repeat the experiment with yet another set of features. We merge the QQ with the Skip-Gram and GloVe features, by separately concatenating the two types of vectors to our data representations (Skip-Gram+QQ and GloVe+QQ columns).

An interesting trend appears both for Hype-Par and Hype-Min: with Skip-Gram+QQ, algorithms perform better than relying on Skip-Gram or QQ alone, and the same happens for Glove+QQ. The new sets of features produce a consistent improvement over the baselines and over our own features. LR outstands other classifiers in the Skip-Gram+QQ combination, reaching .72 mean accuracy and .76 average F1-score (see Table 3).

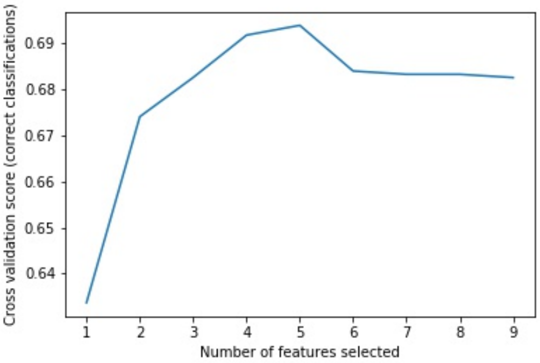

Figure 1: Recurrent Feature Elimination with LR in the Hype-Par setting.

5.3 Analysis

The concatenation of vectors enhance performances and it provides evidence that our quantity and quality vectors enrich both Skip-Gram and GloVe with some useful information about hyperbole. This observation, together with the outcome of LR, suggests that the task introduced in the

	Hype-Par (LR, Skip-Gram+QQ)	Hype-Min (NB, GloVe+QQ)
Precision	.76	.54
Recall	.76	.53
F1-score	**.76**	.52

Table 3: Average precision, recall and F-1 values obtained in cross validation for the best performing models in Hype-Par and Hype-Min.

	Conventional	Measurable	Possible
YES	50	18	20
NO	43	75	73

Table 4: Annotators' judgments about 93 misclassified hyperboles.

present work delineates a promising field of inquiry, and that it can benefit from our QQ features.

To deepen our understanding of the results, we analyze the errors made in the Hype-Par classification. This conclusive stage of the study serves to probe if the overstatements that tend to be misclassified share some characteristics that are missing from our group of features.

From a test set of 468 data points, we collect those that are incorrectly labeled by all of the models, which comprise 185 sentences, 93 of which are hyperbolic. Examples are: "He's more aged than the hills" (Hype), "They will die of envy" (Hype), "You get into that university, you won't get out alive" (Hype), "He hiccuped for a long time" (Par).

We investigate the characteristics for which we collected judgments in the Crowdflower experiment (Questions 4, 5 and 6 in the Section above), that is, the measurable trait of exaggerations, and their degree of hyperbolicity and conventionality. Table 4 details the judgments about misclassified hyperboles, as rated by the the majority of their annotators. More than half hyperboles were declared conventional, impossible and non-measurable.

It appears that conventional hyperboles are more difficult to recognize (the test set comprises an equal number of conventional and creative items). This is not surprising if we consider that the classifiers have especially relied on the unexpectedness feature. Vectors encode information relative to the context where words occur, so they might capture that the words of a conventional hyperbole are likely to be used together, since they are highly common in language (e.g. [2]).

As for the second characteristic, the majority of hyperboles amplifies, according to the annotators, an unmeasurable trait of the topic of discourse. This sheds light on the topic of the errors: understanding that a sentence overshoots reality with respect to a subjective impression (e.g. heaviness of commitment [3]) is harder than with an objective quality (e.g. age [1]). In future work, we may investigate how to better formalize the counterfactuality condition, by specifying different strategies to use in the two cases.

Lastly, errors regarding the degree of hyperbolicity run up against our expectations. We hypothesized that a more exaggerated hyperbole (i.e. impossible) is easier to identify, but statistics suggest the opposite. Impossible hyperboles may be obvious for humans, who use pragmatic knowledge, but not for an agent which entirely relies on linguistic information. Prospective research may test if this problem can be overcome with the help of multimodal strategies.

6 Conclusions

Hyperbole, the figure of exaggeration and one of the hallmarks of human communication, is tackled in this paper from a computational perspective. Our research aimed at answering the question whether it is possible to endow a system with the ability to identify exaggerated sentences. Experimental results showed promising directions for their automatic detection and suggest that the execution of this task can be based on semantic features. As a novel approach to hyperboles, the project started with no related studies to compare to, nor useful resources to investigate. Hence, its main contribution to the field of NLP is the proposal of a new task together with the construction of a corpus of hyperboles, and its main achievement is the devising of a procedure to learn such figurative mechanism. Specifically, the described experiment tested the hypothesis that quantity and quality, which emerge from the body of literature as two core features of exaggerations, are also useful for the automatic processing of the figure.

As a future work, we plan to investigate if our QQ-based models are enhanced by extra-linguistic knowledge, and to incorporate contextual, multimodal features along semantic ones.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Laura Cano Mora. 2010. All or nothing: A semantic analysis of hyperbole. *Revista de Lingüística y Lenguas Aplicadas*, 4(1):25–35.

Robyn Carston and Catherine Wearing. 2011. Metaphor, hyperbole and simile: A pragmatic approach. *Language and Cognition*, 3(2):283–312.

Claudia Claridge. 2011. *Hyperbole in English: A Corpus-Based Study of Exaggeration*. Cambridge University Press.

Herbert L. Colston and Jennifer O'Brien. 2000. Contrast of kind versus contrast of magnitude: The pragmatic accomplishments of irony and hyperbole. *Discourse processes*, 30(2):179–199.

Gaëlle Ferré. 2014. Multimodal hyperbole. *Multimodal Communication*, 3(1):25–50.

Robert J. Fogelin. 1988. *Figuratively Speaking: Revised Edition*. New Haven: Yale University Press.

Lorenzo Gatti, Marco Guerini, and Marco Turchi. 2016. Sentiwords: Deriving a high precision and high coverage lexicon for sentiment analysis. *IEEE Transactions on Affective Computing*, 7(4):409–421.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Clayton J. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*.

Roger J. Kreuz, Richard M. Roberts, Brenda K. Johnson, and Eugenie L. Bertus. 1996. Figurative language occurrence and co-occurrence in contemporary literature. In Roger J. Kreuz & Mary Sue MacNealy, editor, *Empirical Approaches to Literature and Aesthetics*, pages 83–97. Norwood, NJ: Ablex Publishing Corporation.

Steven Loria. 2014. Textblob: simplified text processing. Retrieved 04 March 2017 from `http://textblob.readthedocs.io/en/dev/`.

Michael McCarthy and Ronald Carter. 2004. "There's millions of them": hyperbole in everyday conversation. *Journal of pragmatics*, 36(2):149–184.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ICLR Workshop*.

M^{a} Sandra Peña and Francisco J. Ruiz de Mendoza. 2017. Construing and constructing hyperbole. *Studies in Figurative Thought and Language*, 56:41.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Anita Pomerantz. 1986. Extreme case formulations: A way of legitimizing claims. *Human studies*, 9(2):219–229.

Joshua R. Ritter. 2012. Recovering hyperbole: Rethinking the limits of rhetoric for an age of excess. *Philosophy & Rhetoric*, 45(4):406–428.

Richard M. Roberts and Roger J. Kreuz. 1994. Why do people use figurative language? *Psychological science*, 5(3):159–163.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 248–258.

Building Context-aware Clause Representations for Situation Entity Type Classification

Zeyu Dai, Ruihong Huang
Department of Computer Science and Engineering
Texas A&M University
`{jzdaizeyu, huangrh}@tamu.edu`

Abstract

Capabilities to categorize a clause based on the type of situation entity (e.g., events, states and generic statements) the clause introduces to the discourse can benefit many NLP applications. Observing that the situation entity type of a clause depends on discourse functions the clause plays in a paragraph and the interpretation of discourse functions depends heavily on paragraph-wide contexts, we propose to build context-aware clause representations for predicting situation entity types of clauses. Specifically, we propose a hierarchical recurrent neural network model to read a whole paragraph at a time and jointly learn representations for all the clauses in the paragraph by extensively modeling context influences and inter-dependencies of clauses. Experimental results show that our model achieves the state-of-the-art performance for clause-level situation entity classification on the genre-rich MASC+Wiki corpus, which approaches human-level performance.

1 Introduction

Clauses in a paragraph play different discourse and pragmatic roles and have different aspectual properties (Smith, 1997; Verkuyl, 2013) accordingly. We aim to categorize a clause based on its aspectual property and more specifically, based on the type of Situation Entity (SE)[1] (e.g., events, states, generalizing statements and generic statements) the clause introduces to the discourse, following the recent work by (Friedrich et al., 2016). Understanding SE types of clauses is beneficial for many NLP tasks, including discourse mode identification[2] (Smith, 2003, 2005), text summarization, information extraction and question answering.

The situation entity type of a clause reflects discourse roles the clause plays in a paragraph and discourse role interpretation depends heavily on paragraph-wide contexts. Recently, Friedrich et al. (2016) used insightful syntactic-semantic features extracted from the target clause itself for SE type classification, which has achieved good performance across several genres when evaluated on the newly created large dataset MASC+Wiki. In addition, Friedrich et al. (2016) implemented a sequence labeling model with conditional random fields (CRF) (Lafferty et al., 2001) for fine-tuning a sequence of predicted SE types. However, other than leveraging common SE label patterns (e.g., GENERIC clauses tend to cluster together.), this approach largely ignored the wider contexts a clause appears in when predicting its SE type.

To further improve the performance and robustness of situation entity type classification, we argue that we should consider influences of wider contexts more extensively, not only by fine-tuning a sequence of SE type predictions, but also in deriving clause representations and obtaining precise individual SE type predictions. For example, we distinguish GENERIC statements from GENERALIZING statements depending on if a clause expresses general information over classes or kinds instead of specific individuals. We recognize the latter two clauses in the following paragraph as GENERALIZING because both clauses describe situations related to the Amazon river:

(1): *[Today, the Amazon river is experiencing a crisis of overfishing.]*STATE *[Both subsistence fishers and their commercial rivals compete in netting large quantities of pacu,]*GENERALIZING

---

[1] The Situation Entity (SE) type of a clause is defined with respect to three situation-related features: the main NP referent type (specific or generic), fundamental aspectual class (stative or dynamic), and whether the situation evoked is episodic or habitual (Friedrich and Palmer, 2014b).

[2] E.g., EVENTs and STATEs are dominant in *narratives* while GENERALIZINGs and GENERICs are dominant in *informative* discourses.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3305–3315
Brussels, Belgium, October 31 - November 4, 2018.

*[which bring good prices at markets in Brazil and abroad.]*GENERALIZING

If we ignore the wider context, the second clause can be wrongly recognized as GENERIC easily since "fishers" usually refer to one general class rather than specific individuals. However, considering the background introduced in first clause, "fishers" here actually refer to the fishers who fish on Amazon river which become specific individuals immediately.

Therefore, we aim to build context-aware clause representations dynamically which are informed by their paragraph-wide contexts. Specifically, we propose a hierarchical recurrent neural network model to read a whole paragraph at a time and jointly learn representations for all the clauses in the paragraph. Our paragraph-level model derive clause representations by modeling inter-dependencies between clauses within a paragraph. In order to further improve SE type classification performance, we also add an extra CRF layer at the top of our paragraph-level model to fine-tune a sequence of SE type predictions over clauses (Friedrich et al., 2016), which however is not our contribution.

Experimental results show that our paragraph-level neural network model greatly improves the performance of SE type classification on the same MASC+Wiki (Friedrich et al., 2016) corpus and achieves robust performance close to human level. In addition, the CRF layer further improves the SE type classification results, but by a small margin. We hypothesize that situation entity type patterns across clauses may have been largely captured by allowing the preceding and following clauses to influence semantic representation building for a clause in the paragraph-level neural net model.

2 Related Work

2.1 Linguistic Categories of SE Types

The situation entity types annotated in the MASC+Wiki corpus (Friedrich et al., 2016) were initially introduced by Smith (2003), which were then extended by (Palmer et al., 2007; Friedrich and Palmer, 2014b). The situation entity types can be divided into the following broad categories:

- **Eventualities** (EVENT, STATE and REPORT): for clauses representing actual happenings and world states. STATE and EVENT are two fundamental aspectual classes of a clause (Siegel and McKeown, 2000) which can be distinguished by the semantic property of dynamism. REPORT is a subtype of EVENT for quoted speech.
- **General Statives** (GENERIC and GENERALIZING): for clauses that express general information over classes or kinds, or regularities related to specific main referents. The type GENERIC is for utterances describing a general class or kind rather than any specific individuals (e.g., *People love dogs.*). The type GENERALIZING is for habitual utterances that refer to ongoing actions or properties of specific individuals (e.g., *Audubon educates the public.*).
- **Speech Acts** (QUESTION and IMPERATIVE): for clauses expressing two types of speech acts (Searle, 1969).

2.2 Situation Entity (SE) Type Classification

Although situation entities have been well-studied in linguistics, there were only several previous works focusing on data-driven SE type classification using computational methods. Palmer et al. (2007) first implemented a maximum entropy model for SE type classification relying on words, POS tags and some linguistic cues as main features. This work used a relatively small dataset (around 4300 clauses) and did not achieve satisfied performance (around 50% of accuracy).

To bridge the gap, Friedrich et al. (2016) created a much larger dataset MASC+Wiki (more than 40,000 clauses) and achieved better SE type classification performance (around 75% accuracy) by using rich features extracted from the target clause. The feature sets include POS tags, Brown cluster features, syntactic and semantic features of the main verb and main referent as well as features indicating the aspectual nature of a clause. Friedrich et al. (2016) further improved the performance by implementing a sequence labeling (CRF) model to fine-tune a sequence of SE type predictions and noted that much of the performance gain came from modeling the label pattern that GENERIC clauses often occur together. In contrast, we focus on deriving dynamic clause representations informed by paragraph-level contexts and model context influences more extensively.

Becker et al. (2017) proposed a GRU based neural network model that predicts the SE type for one clause each time, by encoding the content of the target clause using a GRU and incorporating several sources of context information, including contents and labels of preceding clauses as well as genre information, using additional separate GRUs (Chung et al., 2014). This model is different from our approach that processes one paragraph (with a sequence of clauses) at a time and extensively models inter-dependencies of clauses.

Other related tasks include predicting aspectual classes of verbs (Friedrich and Palmer, 2014a), classifying genericity of noun phrases (Reiter and Frank, 2010) and predicting clause habituality (Friedrich and Pinkal, 2015).

2.3 Paragraph-level Sequence Labeling

Learning latent representations and predicting a sequence of labels from a long sequence of sentences (clauses), such as a paragraph, is a challenging task. Recently, various neural network models, including Convolution Neural Network (CNN) (Wang and Lu, 2017), Recurrent Neural Network (RNN) based models (Wang et al., 2015; Chiu and Nichols, 2016; Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016) and Sequence to Sequence models (Vaswani et al., 2016; Zheng et al., 2017), have been applied to the general task of sequence labeling. Among them, the bidirectional LSTM (Bi-LSTM) model (Schuster and Paliwal, 1997) has been widely used to process a paragraph for applications such as language generation (Li et al., 2015), dialogue systems (Serban et al., 2016) and text summarization (Nallapati et al., 2016), because of its capabilities in modeling long-distance dependencies between words. In this work, we use two levels of Bi-LSTMs connected by a max-pooling layer to abstract clause representations by extensively modeling paragraph-wide contexts and inter-dependencies between clauses.

3 The Hierarchical Recurrent Neural Network for SE Type Classification

We design an unified neural network to extensively model word-level dependencies as well as clause-level dependencies in deriving clause representations for SE type prediction. Figure 1 shows the architecture of the proposed paragraph-level neural network model which includes two Bi-LSTM layers, one max-pooling layer in between and one final softmax prediction layer.

Given the word sequence of one paragraph as input, the word-level Bi-LSTM will firstly generate a sequence of hidden states as word representations, then a max-pooling layer will be applied to abstract clause embeddings from word representations within a clause. Next, another clause-level Bi-LSTM will run over the sequence of clause embeddings and derive final clause representations by further modeling semantic dependencies between clauses within a paragraph. The softmax prediction layer will then predict a sequence of situation entity (SE) types with one label for each clause, based on the final clause representations.

Word Vectors: To transform the one-hot representation of each word into its distributed word vector (Mikolov et al., 2013), we used the pre-trained 300-dimension Google English word2vec embeddings[3]. For the words which are not included in the vocabulary of Google word2vec, we randomly initialize their word vectors with each dimension sampled from the range $[-0.25, 0.25]$.

For situation entity type classification, it is important to recognize certain types of words such as punctuation marks (e.g., "?" for QUESTION and "!" for IMPERATIVE) as well as entities such as locations and time values. We therefore created feature-rich word vectors by concatenating word embeddings with parts-of-speech (POS) tag and named-entity (NE) tag one-hot embeddings[4].

Deriving Clause Representations: In designing the model, we focus on building clause representations that sufficiently leverage cues from paragraph-wide contexts for SE type prediction, including both preceding and following clauses in a paragraph. To process long paragraphs which may contain a number of clauses, we utilize a two-level bottom-up abstraction approach and progressively obtain the compositional representation of each word (low-level) and then compute a compositional representation of each clause (high-level), with a max-pooling layer in between.

At both word-level and clause-level, we choose the Bi-LSTM as our basic neural net component for representation learning, mainly considering its ability to capture long-distance dependencies between words (clauses) and to integrate influences of context words (clauses) from both directions.

Given a word sequence $X = (x_1, x_2, ..., x_L)$

---

[3] Downloaded from `https://docs.google.com/uc?id=0B7XkCwpI5KDYNlNUTTlSS21pQmM`

[4] Our feature-rich word vectors are of dimension 343, including 300 dimensions for Google word2vec + 36 dimensions for POS tags + 7 dimensions for NE tags. We used the Stanford CoreNLP to generate POS tags and NE tags.

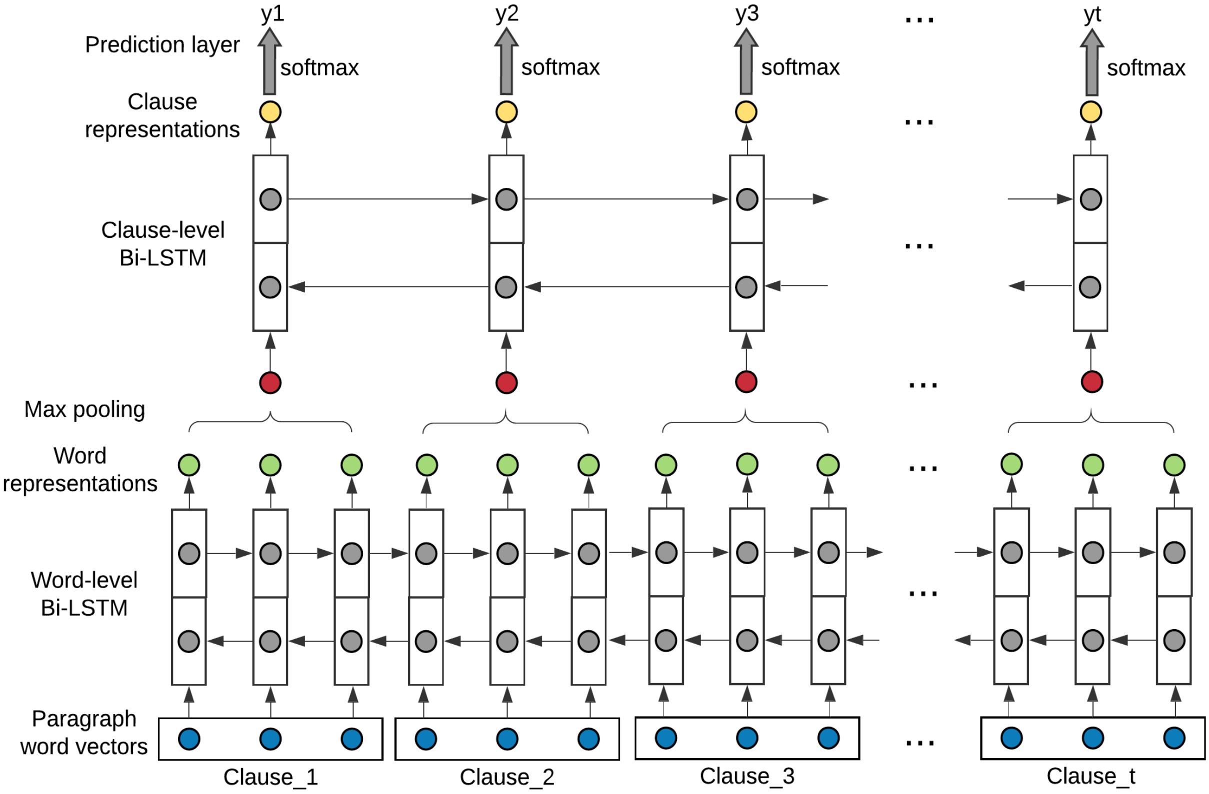

Figure 1: The Paragraph-level Model Architecture for Situation Entity Type Classification.

in a paragraph as the input, the word-level Bi-LSTM will process the input paragraph by using two separate LSTMs, one processes the word sequence from the left to right while the other processes the sequence from the right to left. Therefore, at each word position t, we obtain two hidden states $\overrightarrow{h_t}, \overleftarrow{h_t}$ and concatenate them to get the word representation $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. Then we apply the max-pooling operation over the sequence of word representations for words within a clause in order to get the initial clause embedding:

$$h_{Clause}[j] = \max_{t=Clause\_start}^{Clause\_end} h_t[j] \quad (1)$$

$$where, 1 \leq j \leq hidden\_unit\_size \quad (2)$$

Next, the clause-level Bi-LSTM will process the sequence of initial clause embeddings in a paragraph and generate refined hidden states $\overrightarrow{h_{Clause\_t}}$ and $\overleftarrow{h_{Clause\_t}}$ at each clause position t. Then, we concatenate the two hidden states for a clause to get the final clause representation $h_{Clause\_t} = [\overrightarrow{h_{Clause\_t}}, \overleftarrow{h_{Clause\_t}}]$.

Situation Entity Type Classification: Finally, the prediction layer will predict the situation entity type for each clause by applying the softmax function to its clause representation:

$$y_t = softmax(W_y * h_{Clause\_t} + b_y) \quad (3)$$

3.1 Fine-tune Situation Entity Predictions with a CRF Layer

Previous studies (Friedrich et al., 2016; Becker et al., 2017) show that there exist common SE label patterns between adjacent clauses. For example, Friedrich et al. (2016) reported the fact that GENERIC sentences usually occur together in a paragraph. Following (Friedrich et al., 2016), in order to capture SE label patterns in our hierarchical recurrent neural network model, we add a CRF layer at the top of the softmax prediction layer (shown in figure 2) to fine-tune predicted situation entity types.

The CRF layer will update a state-transition matrix, which can effectively adjust the current label depending on its preceding and following labels. Both the training and decoding procedures of the CRF layer can be conducted efficiently using the Viterbi algorithm. With the CRF layer, the model jointly assigns a sequence of SE labels, one label per clause, by considering individual clause representations as well as common SE label patterns.

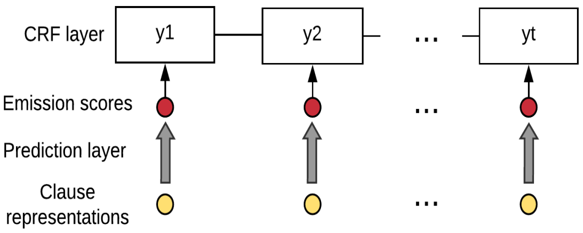

Figure 2: Fine-tune a Situation Entity Label Sequence with a CRF layer.

3.2 Parameter Settings and Model Training

We finalized hyperparameters based on the best performance with 10-fold cross-validation on the training set. The word vectors were fixed during model training. Both word representations and clause representations in the model are of 300 dimensions, and all the Bi-LSTM layers contain 300 hidden units as well. To avoid overfitting, we applied dropout mechanism (Hinton et al., 2012) with dropout rate of 0.5 to both input and output vectors of Bi-LSTM layers. To deal with the exploding gradient problem in LSTMs training, we utilized gradient clipping (Pascanu et al., 2013) with gradient L2-norm threshold of 5.0 and used L2 regularization with $\lambda = 10^{-4}$ simultaneously. These parameters remained the same for all our proposed models including our own baseline models.

We chose the standard cross-entropy loss function for training our neural network models and adopted Adam (Kingma and Ba, 2014) optimizer with the initial learning rate of 0.001 and the batch size[5] of 128. All our proposed models were implemented with Pytorch[6] and converged to the best result within 40 epochs. Note that to diminish the effects of randomness in training neural network models and report stable experimental results, we ran each of the proposed models as well as our own baseline models ten times and reported the averaged performance across the ten runs.

4 Evaluation

4.1 Dataset and Preprocessing

The MASC+Wiki Corpus: We evaluated our neural network model on the MASC+Wiki corpus[7] (Friedrich et al., 2016), which contains more than 40,000 clauses and is the largest annotated dataset for situation entity type classification. The MASC+Wiki dataset is composed of documents from Wikipedia and MASC (Ide et al., 2008) covering as many as 13 written genres (e.g., news, essays, fiction, etc). Table 1 shows statistics of the dataset, from which you can see that the SE type distribution is highly imbalanced. The majority SE type of MASC documents is STATE while the majority SE type of Wikipedia documents is GENERIC. To make our results comparable with previous works (Friedrich et al., 2016; Becker et al., 2017), we used the same 80:20 train-test split with balanced genre distributions.

SE type	MASC	Wiki	Count
STATE	49.8%	24.3%	18337
EVENT	24.3%	18.9%	9688
REPORT	4.8%	0.9%	1617
GENERIC	7.3%	49.7%	7582
GENERALIZING	3.8%	2.5%	1466
QUESTION	3.3%	0.1%	1056
IMPERATIVE	3.2%	0.2%	1046

Table 1: MASC+Wiki Dataset Statistics.

Preprocessing: As described in (Friedrich et al., 2016), texts were split into clauses using SPADE (Soricut and Marcu, 2003). There are 4,784 paragraphs in total in the corpus; and on average, each paragraph contains 9.6 clauses. In figure 4, the horizontal axis shows the distribution of paragraphs based on the number of clauses in a paragraph. The annotations of clauses are stored in separate files from the text files. To recover the paragraph contexts for each clause, we matched its content with the corresponding raw document.

4.2 Systems for Comparisons

We compare the performance of our neural network model with two recent SE type classification models on the MASC+Wiki corpus as well as humans' performance (upper bound).

- CRF (Friedrich et al., 2016): a CRF model that relies heavily on features extracted from the target clause itself.
- GRU (Becker et al., 2017): a GRU based neural network model that incorporates context information by using separate GRU units and predicts the SE type for one clause each time.
- Humans (Friedrich et al., 2016): one annotator's performance when using two other an-

---

[5] Counted as the number of SEs rather than paragraph instances.

[6] `http://pytorch.org/`

[7] `www.coli.uni-saarland.de/projects/sitent/page.php?id=resources`

Model	Macro	Acc	STA	EVE	REP	GENI	GENA	QUE	IMP
Humans	78.6	79.6	82.8	80.5	81.5	75.1	45.8	90.7	93.6
CRF (Friedrich et al., 2016)	71.2	76.4	80.6	78.6	78.9	68.3	29.4	84.4	75.3
Clause-level Bi-LSTM	74.4	78.3	82.6	81.3	84.9	66.2	36.1	88.5	80.9
Paragraph-level Model	77.6	81.2	84.3	**82.1**	85.3	76.4	43.2	**90.8**	81.2
Paragraph-level Model+CRF	**77.8**	**81.3**	84.3	82.0	**85.7**	**77.0**	**43.5**	90.4	**81.5**

Table 2: Situation Entity Type Classification Results on the Training Set of MASC+Wiki with 10-Fold Cross-Validation. We report accuracy (Acc), macro-average F1-score (Macro) and class-wise F1 scores for STATE (STA), EVENT (EVE), REPORT (REP), GENERIC (GENI), GENERALIZING (GENA), QUESTION (QUE) and IMPERATIVE (IMP).

Model	Macro	Acc
CRF (Friedrich et al., 2016)	69.3	74.7
GRU (Becker et al., 2017)	68.0	71.1
Clause-level Bi-LSTM	73.5	76.7
Paragraph-level Model	77.0	80.0
Paragraph-level Model + CRF	**77.4**	**80.7**

Table 3: Situation Entity Type Classification Results on the Test Set of MASC+Wiki. We report accuracy (Acc) and macro-average F1 (Macro).

notators' annotation as "gold labels". It has been reported that labeling SE types is a non-trivial task even for humans.

In addition, we implemented a clause-level Bi-LSTM model as our own baseline, which takes a single clause as its input. Since there is only one clause, the upper Bi-LSTM layer shown in Figure 1 is meaningless and removed in the clause-level Bi-LSTM model.

4.3 Experimental Results

Following the previous work (Friedrich et al., 2016) on the same task and dataset, we report accuracy and macro-average F1-score across SE types on the test set of MASC+Wiki.

The first section of Table 3 shows the results of the previous works. The second section shows the result of our implemented clause-level Bi-LSTM baseline, which already outperforms the previous best model. This result proves the effectiveness of the Bi-LSTM + max pooling approach in clause representation learning (Conneau et al., 2017). The third section reports the performance of the paragraph-level models that uses paragraph-wide contexts as input. Compared with the baseline clause-level Bi-LSTM model, the basic paragraph-level model achieves 3.5% and 3.3% of performance gains in macro-average F1-score and accuracy respectively. Building on top of the basic paragraph-level model, the CRF layer further improves the SE type prediction performance slightly by 0.4% and 0.7% in macro-average F1-score and accuracy respectively. Therefore, our full model with the CRF layer achieves the state-of-the-art performance on the MASC+Wiki corpus.

5 Analysis

5.1 10-Fold Cross-Validation

We noticed that the previous work (Friedrich et al., 2016) did not publish the class-wise performance of their model on the test set, instead, they reported the detailed performance on the training set using 10-fold cross-validation. For direct comparisons, we also report our 10-fold cross-validation results[8] on the training set of MASC+Wiki.

Table 2 reports the cross-validation classification results. Consistently, our clause-level baseline model already outperforms the previous best model. By exploiting paragraph-wide contexts, the basic paragraph-level model obtains consistent performance improvements across all the classes compared with the baseline clause-level prediction model, especially for the classes GENERIC and GENERALIZING, where the improvements are significant. After using the CRF layer to fine-tune the predicted SE label sequence, slight performance improvements were observed on the four small classes. Overall, the full paragraph-level neural network model achieves the best macro-average F1-score of 77.8% in predicting SE types, which not only outperforms all previous approaches but also reaches human-like performance on some classes.

---

[8] The original folds split used by Friedrich et al. (2016) is not available. So we manually split folds by ourselves with even genre distribution across folds.

Model	Macro	Acc	STA	EVE	REP	GENI	GENA	QUE	IMP
CRF (Friedrich et al., 2016)	66.6	71.8	78.2	77.0	76.8	44.8	27.4	81.8	70.8
Clause-level Bi-LSTM	69.3	73.3	79.5	78.7	82.8	47.6	31.9	86.9	77.7
Paragraph-level Model	73.2	77.2	81.5	80.1	83.2	64.7	37.2	88.1	**77.8**
Paragraph-level Model+CRF	**73.5**	**77.4**	81.5	**80.3**	**83.7**	**66.5**	**37.4**	**88.5**	76.7

Table 4: Cross-genre Classification Results on the Training Set of MASC+Wiki. We report accuracy (Acc), macro-average F1-score (Macro) and class-wise F1 scores.

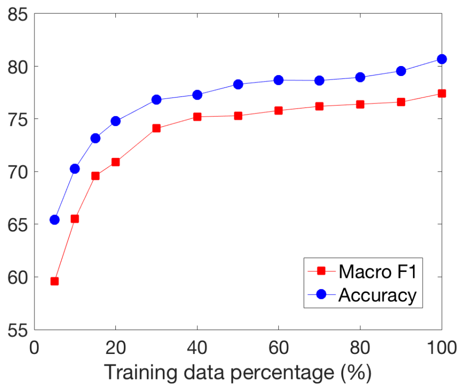

Figure 3: Learning Curve of the Paragraph-level Model + CRF on MASC+Wiki.

5.2 Impact of Genre

Considering that MASC+Wiki is rich in written genres, we additionally conduct cross-genre classification experiments, where we use one genre of documents for testing and the other genres of documents for training. The purpose of cross-genre experiment is to see whether the model can work robustly across genres.

Table 4 shows cross-genre experimental results of our neural network models on the training set of MASC+Wiki by treating each genre as one cross-validation fold. As we expected, both the macro-average F1-score and class-wise F1 scores are lower compared with the results in Table 2 where in-genre data were used for model training as well. But the performance drop on the paragraph-level models is little, which clearly outperform the previous system (Friedrich et al., 2016) and the baseline model by a large margin. As shown in Table 5, benefited from modeling wider contexts and common SE label patterns, our full paragraph-level model improves performance across almost all the genres. The high performance in the cross-genre setting demonstrates the robustness of our paragraph-level model across genres.

Genre	Baseline	Full Model	Humans
blog	66.7	**70.3**	72.9
email	71.1	**71.5**	67.0
essays	61.2	**64.1**	64.6
ficlets	67.9	**68.8**	81.7
fiction	70.2	**72.1**	76.7
gov-docs	68.6	**68.9**	72.6
jokes	70.0	**75.0**	82.0
journal	**66.7**	66.4	63.7
letters	68.6	**71.2**	68.0
news	70.4	**72.7**	78.6
technical	55.7	**60.5**	54.7
travel	51.3	**53.6**	48.9
wiki	55.2	**60.6**	69.2

Table 5: Cross-genre Classification Results by Genre on the Training Set of MASC+Wiki. Baseline: Clause-level Bi-LSTM; Full Model: Paragraph-level Model + CRF. We report macro-average F1-score for each genre.

5.3 Impact of Training Data Size

In order to understand how much training data is required to train the paragraph-level model and obtain a good performance for SE type classification, we plot the learning curve shown in Figure 3 by training the full model several times using an increasing amount of training data. The classification performance increased quickly before the amount of training data was increased to 30% of the full training set; then the learning curve starts to become saturated afterwards. We conclude that the paragraph-level model can achieve a high performance quickly without requiring a large amount of training data.

5.4 Impact of Paragraph Length

To study the influence of paragraph lengths to the performance of the paragraph-level models, we report the performance of our proposed models on subsets of the test set, with paragraphs divided based on the number of clauses in a para-

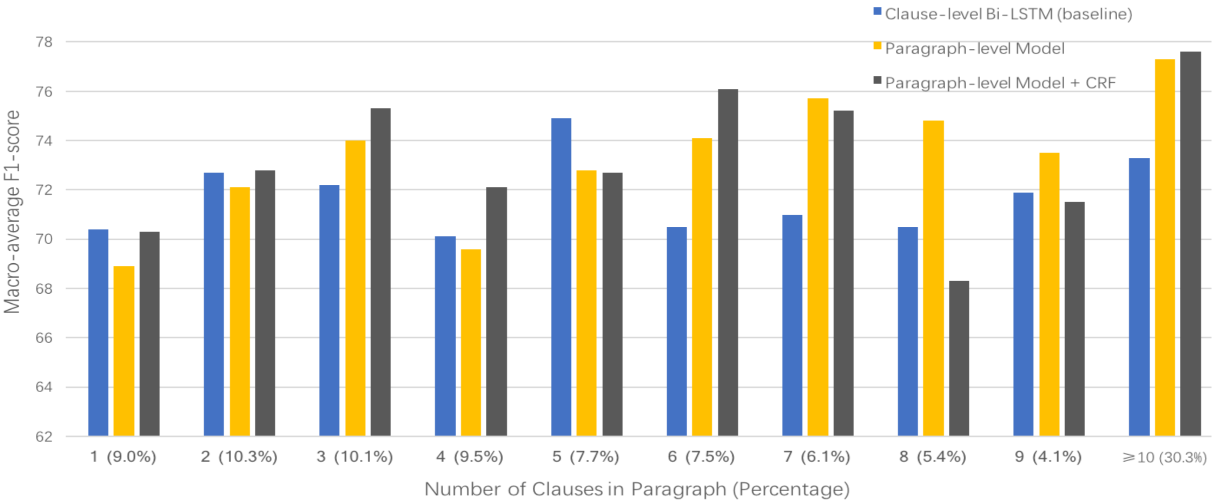

Figure 4: Impact of Paragraph Lengths. We plot the macro-average F1-score for each paragraph length.

graph. The histogram in Figure 4 compares performance of the two paragraph-level models and the baseline model. Note that the last bucket (paragraphs containing ten or more clauses) of the histogram is especially large and contains over 30% of all the paragraphs in the test set. Clearly, the paragraph-level model greatly outperforms the baseline clause-level model on paragraphs containing more than 6 clauses, which covers over 50% of the test set. Adding the CRF layer further improves the performance of the paragraph-level model on long paragraphs (with 10 or more clauses), while the influences to the performance are mixed on short paragraphs. Therefore, it is beneficial to model wider paragraph-level contexts and inter-dependencies between clauses for situation entity type classification, especially when processing long paragraphs.

5.5 Impact of Discourse Connective Phrases

As one aspect of modeling context influences and clause inter-dependencies in SE type identification, we investigated the role of discourse connective phrases in determining the SE type of clauses they connect. Our assumption is that discourse connectives are important to glue clauses together and removing them affects text coherence and information flow between clauses. Intuitively, the connective “and” may occur between two clauses with the same SE type; “for example” may indicate that the following clause is not GENERIC. Therefore, we designed a pilot experiment to see whether discourse connective phrases are indispensable in building clause representations.

In this pilot experiment, we extracted a list of 100 explicit discourse connectives. PDTB corpus (Prasad et al., 2008) and identified clauses that start with a discourse connecte[9]. Then we ran the full paragraph-level model with one modification, i.e., disregarding words in connective phrases when conducting the max-pooling operation in equation (1), thus we did not consider discourse connective phrases directly when building a clause representation.

As shown in Table 6, for clauses containing a discourse connective phrase, both macro-average F1-score and accuracy dropped due to the exclusion of discourse connective phrases. The performance was negatively influenced across all the SE types except the type of QUESTION and IMPERATIVE[10]. The performance decreases on three SE types, REPORT, GENERIC and GENERALIZING, are noticeable. To some extent, this pilot study shows that modeling text coherence and the overall discourse structure of a paragraph is important in situation entity type classification.

5.6 Confusion Matrix

Table 7 reports the confusion matrix of the full model on the training set of MASC+Wiki with cross-validation. We can see that the four situation entity types, including two eventualities (STATE and EVENT) and two general sta-

---

[9] We found that 20.6% of clauses in the MASC+Wiki corpus contain a discourse connective phrase.

[10] A possible explanation is that recognizing QUESTION (IMPERATIVE) clauses mainly relies on seeing certain punctuation marks and key words, such as “?” (“!”) and “why” (“please”), which are independent from discourse connectives.

Macro	Acc	STA	EVE	REP	GENI	GENA	QUE	IMP
-1.2	-0.9	-1.0	-0.8	-2.3	-3.4	-2.2	0.5	0.3

Table 6: Impact of Discourse Connective Phrases. We report performance losses (percentages) on clauses containing a connective phrase, when discourse connective phrases were excluded from clause representation building.

SE Type		Predicted						
		STA	EVE	REP	GENI	GENA	QUES	IMP
Gold	STA	12558	980	32	931	155	51	85
	EVE	819	6626	116	242	124	11	16
	REP	42	143	1097	3	4	1	2
	GENI	1157	175	3	4523	117	14	14
	GENA	281	254	5	161	431	5	12
	QUES	51	7	2	8	1	773	4
	IMP	106	21	7	18	3	3	650

Table 7: Confusion Matrix of the Paragraph-level Model + CRF on the Training Set of MASC+Wiki with 10-Fold Cross-Validation.

tives (GENERIC and GENERALIZING), are often mutually confused with each other. To further improve the performance of situation entity type classification, it is important to accurately detect events within a clause (for fixing STATE/EVENT errors) and identify the genericity of main referents (for fixing STATE/GENERIC and GENERIC/GENERALIZING errors), which can be potentially achieved by incorporating linguistic features into neural net models.

6 Conclusion

We presented a paragraph-level neural network model for situation entity (SE) type classification which builds context-aware clause representations by modeling inter-dependencies of clauses in a paragraph. Evaluation shows that the paragraph-level model outperforms previous systems for SE type classification and approaches human-level performance. In the future, we plan to incorporate SE type information in various downstream applications, e.g., many information extraction applications that require distinguishing specific fact descriptions from generic statements.

Acknowledgments

This work was partially supported by the National Science Foundation via NSF Award IIS-1755943. Disclaimer: the views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or the U.S. Government. In addition, we gratefully acknowledge the support of NVIDIA Corporation for their donation of one GeForce GTX TITAN X GPU used for this research.

References

Maria Becker, Michael Staniek, Vivi Nastase, Alexis Palmer, and Anette Frank. 2017. Classifying semantic clause types: Modeling context and genre characteristics with recurrent neural networks and attention. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 230–240.

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

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 681–691.

Annemarie Friedrich and Alexis Palmer. 2014a. Automatic prediction of aspectual class of verbs in context. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 517–523.

Annemarie Friedrich and Alexis Palmer. 2014b. Situation entity annotation. In *Proceedings of LAW VIII-The 8th Linguistic Annotation Workshop*, pages 149–158.

Annemarie Friedrich, Alexis Palmer, and Manfred Pinkal. 2016. Situation entity types: automatic classification of clause-level aspect. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1757–1768.

Annemarie Friedrich and Manfred Pinkal. 2015. Automatic recognition of habituals: a three-way classification of clausal aspect. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2471–2481.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

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

Nancy Ide, Collin Baker, Christiane Fellbaum, and Charles Fillmore. 2008. Masc: The manually annotated sub-corpus of american english. In *In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC*. Citeseer.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, volume 951, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1106–1115.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1064–1074.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016*, page 280.

Alexis Palmer, Elias Ponvert, Jason Baldridge, and Carlota Smith. 2007. A sequencing model for situation entity classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 896–903.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *In Proceedings of LREC*.

Nils Reiter and Anette Frank. 2010. Identifying generic noun phrases. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 40–49. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

John R Searle. 1969. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Eric V Siegel and Kathleen R McKeown. 2000. Learning methods to combine linguistic indicators: Improving aspectual classification and revealing linguistic insights. *Computational Linguistics*, 26(4):595–628.

Carlota S. Smith. 1997. *The Parameter of Aspect.* Springer Science & Business Media.

Carlota S. Smith. 2003. *Modes of Discourse: The Local Structure of Texts.* Cambridge University Press.

Carlota S Smith. 2005. Aspectual entities and tense in discourse. In *Aspectual inquiries*, pages 223–237. Springer.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with lstms. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 232–237.

Hendrik Jacob Verkuyl. 2013. *On the compositional nature of the aspects*, volume 15. Springer Science & Business Media.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168*.

Qingqing Wang and Yue Lu. 2017. A sequence labeling convolutional network and its application to handwritten string recognition. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2950–2956. AAAI Press.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1227–1236.

Hierarchical Dirichlet Gaussian Marked Hawkes Process for Narrative Reconstruction in Continuous Time Domain

Yeon Seonwoo, **Sungjoon Park** and **Alice Oh**
Department of Computing, KAIST, Republic of Korea
`{yeon.seonwoo, sungjoon.park}@kaist.ac.kr, alice.oh@kaist.edu`

Abstract

In news and discussions, many articles and posts are provided without their related previous articles or posts. Hence, it is difficult to understand the context from which the articles and posts have occurred. In this paper, we propose the Hierarchical Dirichlet Gaussian Marked Hawkes process (HD-GMHP) for reconstructing the narratives and thread structures of news articles and discussion posts. HD-GMHP unifies three modeling strategies in previous research: temporal characteristics, triggering event relations, and meta information of text in news articles and discussion threads. To show the effectiveness of the model, we perform experiments in narrative reconstruction and thread reconstruction with real world datasets: articles from the New York Times and a corpus of Wikipedia conversations. The experimental results show that HD-GMHP outperforms the baselines of LDA, HDP, and HDHP for both tasks.

1 Introduction

Online news sites and discussion forums generate large volumes of articles and discussions, which we can call "events". To fully understand the discussions and the news stories, one often needs a larger context for that text, such as what related posts and relevant articles have been posted before. For instance, to understand a news article about the *presidential elections*, we would need to know the history of the candidates' political actions through relevant previous articles. While there are some news articles with a curated set of related articles and discussion threads with a well-organized structure, there are many more articles and discussion threads for which the structure is absent or incomplete. In this context, automatically reconstructing the narrative of articles and thread structure is an important problem.

Generally, textual information and various meta information such as location and keywords are used as features to solve this problem of narrative reconstruction. With these features, previous research mainly focus on three modeling strategies. First, they model the triggering relationship of events to identify which preceding events led to the occurrence of the current event. Second, they use meta information such as location and keywords. Third, they consider the temporal characteristics in the event stream, such that events in close temporal proximity are more likely to be related. However, there is no method that effectively considers all three of these. In narrative reconstruction, there are several approaches that focus on using meta information and temporal characteristics with clustering methods (Zhou et al., 2016; Tang et al., 2015; Ahmed et al., 2011), and there are several approaches using the Hawkes process to model the temporal characteristics (Du et al., 2015; Mavroforakis et al., 2017; Jankowiak and Gomez-Rodriguez, 2017). In thread reconstruction, there are approaches that focus on modeling triggering relationships of events and using meta information (Kim et al., 2010; Louis and Cohen, 2015; Wang et al., 2011b).

In this paper, we propose a novel Gaussian Marked Hawkes Process (GMHP) that effectively reconstructs the narrative structure of articles and the thread structure of discussions considering all three modeling strategies. GMHP uses the Hawkes process to model events in continuous time, a Gaussian distribution for modeling the meta information of text, and the mixture of Gaussian for modeling the triggering relationships of events. The detailed modeling strategies are described as follows. We use the Hawkes process to model time in the continuous domain, as the Hawkes process is a stochastic process used to understand a sequence of events in continuous time

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3316–3325
Brussels, Belgium, October 31 - November 4, 2018.

(Iwata et al., 2013; Rong et al., 2015). To use meta information, we represent text and meta information in a general vector form and use the Hawkes process to handle the vector of event information with a Gaussian distribution. To model the triggering relationships, we assume a model structure parameterized by each preceding event so that an event can be directly generated from a probability distribution parameterized by preceding events.

The GMHP models a single narrative or thread in event streams. To find the narratives or threads from a mixture of event streams, we combine our GMHP model with the Hierarchical Dirichlet Process to build HD-GMHP.

We evaluate the effectiveness of our model with two real world datasets: articles from the New York Times, and discussion threads from Wikipedia. In the New York Times dataset, we perform a narrative reconstruction experiment and compare the results with the human annotated narrative labels. In the Wikipedia discussion corpus, we perform two kinds of thread reconstruction experiment. One is grouping posts in the same thread. The other is reconstructing the post-reply structure of the posts. From these experiments, we see that our model outperforms the state-of-the-art model, the hierarchical Dirichlet Hawkes process (HDHP) (Mavroforakis et al., 2017).

The contributions of our research are threefold. First, we propose the Gaussian Marked Hawkes Process that effectively models a single narrative (event stream) with all three modeling strategies used in previous research. Second, we propose HD-GMHP, a combination of the GMHP model with the HDP to reconstruct the narratives of articles and the thread structure of discussions from a mixture of event streams. Finally, we propose a novel inference algorithm of the HD-GMHP with the Sequential Monte Carlo method (Doucet et al., 2001).

2 Related Work

Narrative Reconstruction: One major approach to reconstructing narratives from news articles is clustering articles by using a variant of the Chinese Restaurant Process (CRP). Related work such as (Zhou et al., 2016; Tang et al., 2015; Ahmed et al., 2011) models chronologically ordered news articles with text and various meta information including author, organization, keywords, and location. They use the CRP, distant-dependent CRP (Blei and Frazier, 2011). There is research that uses recurrent CRP (Ahmed and Xing, 2008) and exponential time decaying kernel to model probability of time difference between two relevant events. But they use discrete time information instead of continuous form and handcrafted parameters of the kernel (Ahmed et al., 2011).

There is another approach that reconstructs narratives by directly extracting important sentences from articles. (Xu et al., 2013) proposes a model that considers the sentence and image level narrative reconstruction as an optimization problem and solves it by maximizing the divergence of narratives with some constraints. (Wang et al., 2016) solves the narrative reconstruction problem as a sentence recommendation problem and uses matrix factorization. But these existing models focus on how to handle text and meta information of articles, while our model uses the Hawkes process to effectively model continuous time information of events.

Discussion Thread Reconstruction: There are several approaches to reconstruct threads from a corpus of unstructured discussions. (Wang et al., 2011a) uses Conditional Random Field to reconstruct reply structure in discussion corpus. (Balali et al., 2014) uses content, time and author information as features of a single post with rank SVM to reconstruct thread structure. (Dehghani et al., 2013; Aumayr et al., 2011) uses SVM and a decision tree with meta information of posts.

However, a major limitation in these previous research is that they are assuming that for each post, the main thread where it belongs is given. That is, the problem they solve is finding the post for which a post is immediately replying, rather than treating the corpus as a single set of posts with no known information about the threads, the initial post of each thread, and the posts that belong to each thread. This limitation of the previous research means those approaches are not applicable in more general online conversation data, such as IRC or a Facebook group chat which is a massive unstructured online discussion for which the initial post of a thread is not labeled. Unlike this strong assumption in previous research, we use a more general assumption that the initial posts are unknown, so our approach would be applicable to a wider, more general discussion data. Also, as in the narrative reconstruction research area, previous research focuses on how to handle text and meta information in posts. Again, unlike previous research, our research uses the Hawkes process to model continuous time information.

Continuous Time Modeling: The Hawkes process, a stochastic process that models continuous time information of events with event occurrence history, is an effective solution to model events in continuous time. One of the main research themes in the Hawkes process literature is finding which events trigger which other events. (He et al., 2015) models the topic diffusion patterns in a social network by inferring the triggering node with the Hawkes process. The Hawkes process is also used to model social event streams (Rong et al., 2015) and to classify rumors (Lukasik et al., 2016), and a combination of the Hawkes process and the Dirichlet mixture model is used to cluster event streams (Xu and Zha, 2017).

Recent research clusters text streams with the Hawkes process and the Chinese Restaurant Process or the Chinese Restaurant Franchise (Mavroforakis et al., 2017; Du et al., 2015). They use the bag-of-words representation of text in their model, while (Jankowiak and Gomez-Rodriguez, 2017) proposes a Hawkes process model that can handle a more general vector representation of events. The main difference of our model compared to this research is that we add the triggering relationship of two events. With this addition, our model can reconstruct narratives with an explicit relation of two documents.

3 Hawkes Processes

Before we describe our proposed model, we briefly explain the Hawkes process, one of two main stochastic processes used in our model. We leave out the explanation of the HDP due to space.

The Hawkes process (Hawkes, 1971) is a subclass of temporal point processes, whose functional form for intensity with exponential decaying kernel is represented as

$$\lambda^*(t) = \lambda_0(t) + \int_0^t \alpha\beta e^{-\beta(t-s)} dN(s),$$

where the intensity, $\lambda^*(t)$ represents the conditional probability of an event occurrence within time window $[t, t+dt)$. The Hawkes process is used to model the number of occurrences of events where one event can trigger other events. In the equation above, the base intensity $\lambda_0(t)$ models the intensity of events that occur on their own initiative whereas $\alpha\beta e^{-\beta(t-s)}$ models the intensity of events that are triggered by the previous event that occurred at time s. Here, multiplication of α and β represents influence of the previous event and β represents decaying rate of the influence. Thus, the effect of the previous event exponentially decays with respect to the time difference. From the definition of intensity $\lambda^*(t)$, the derived likelihood form of the Hawkes process is as follows,

$$f(D|\Theta) = e^{-\Lambda(T)} \prod_{i=1}^{n} \lambda^*(t_i), \tag{1}$$

where $\Lambda(T) = \int_0^T \lambda^*(t)dt$.

4 Problem Setting

In this section, we define the event stream and the narrative and the thread reconstruction problems.

Definition of Event Stream: If a text appears at time t_i, we define the event s_i as $(t_i, \vec{e_i}, z_i, x_i)$. Here, $\vec{e_i}$ is the feature vector of the text, x_i is the latent global cluster indicator of event s_i which represents the cluster for events with similar text information, and z_i is the latent local cluster indicator for events that are temporally related in the same cluster. We define event stream S as $[s_1, .., s_n]$.

Assumptions: 1) We assume that two events in same local cluster occur in near time and have similar feature vectors $\vec{e}$. These properties are called temporal and spatial locality. 2) We assume hierarchy structure of a global cluster and a local cluster. That is, one global cluster can consist of multiple local clusters.

Problem Formulation: We formulate the spatial locality of two events in the same local cluster with a Gaussian distribution. If two events s_i and s_j are in the same local cluster and $t_i > t_j$, then we assume the later event $\vec{e_i}$ is generated from one of two relations,

$$\vec{e_i} \sim \mathcal{N}(\vec{e_j}, \Sigma_v), \quad \vec{e_i}, \sim \mathcal{N}(\vec{e}_0, \Sigma_0).$$

Here, $\vec{e}_0$ is the base event vector and Σ_0 is the covariance matrix of the cluster. Σ_v is the covariance matrix of the Gaussian distribution generated by a past event in the cluster.

We use the Hawkes process to formulate the temporal locality of two events in the same local cluster. If event s_i and s_j are in the same local cluster and $t_i > t_j$, then t_i is generated from in either following relations,

$$t_i \sim \text{Poisson Process}(\mu),$$
$$t_i - t_j \sim \text{Hawkes}(\alpha, \beta).$$

Here, if t_i is generated from Hawkes process of parameter α and β with time t_j and $\vec{e}_i$ is generated from $\vec{e}_j$, then we say that event s_j is the *parent event* of event s_i.

We formulate the hierarchy structure of the global cluster and the local cluster with Hierarchical Dirichlet Process (Teh et al., 2006). If the parameters θ_{z_i} of local clusters $z_1, z_2, .., z_n$ are equal to the parameters of the global cluster Θ_x, then we say that there is a hierarchy between all the local clusters and the global cluster. And this hierarchy structure can be written as follows,

$$\Theta_x = \theta_{z_1} = \theta_{z_2} = ... = \theta_{z_n}.$$

Now, we define the narrative reconstruction and the thread reconstruction problem as a problem of inferring the latent variables in S.

5 Model

We now describe clustering a mixture of event stream S with the Gaussian Marked Hawkes Process and the hierarchical Dirichlet process. We first propose Gaussian Marked Hawkes Process (GMHP) that models temporal and spatial locality assumptions that described in section 4. And after defining the GMHP, we propose Hierarchical Dirichlet Gaussian Marked Hawkes Process (HD-GMHP), a combination of the GMHP with the Hierarchical Dirichlet Process. The GMHP models event streams with the same local cluster z and HDP groups the local clusters to one global cluster x.

5.1 Gaussian Marked Hawkes Processes

5.1.1 Model Description

In GMHP, we assume events are generated by a past event or by their own initiative. If event s_i is generated by event s_j, then we say that event s_j is the parent event of event s_i. If event s_i occurs on their own initiative, the index of the parent event is 0. We define the intensity function with the given parent event c_i as follows,

$$\lambda(t_i|c_i) = \begin{cases} \mu & \text{if } c_i = 0 \\ \alpha\beta e^{-\beta(t_i - t_{c_i})} & \text{otherwise} \end{cases} \quad (2)$$

To model the spatial locality of two D-dimensional event vectors $\vec{e}_i$, $\vec{e}_{c_i}$, we define probability distribution for $\vec{e}_i$ as follows,

$$p_{c_i}(\vec{e}_i) = \begin{cases} \mathcal{N}(\vec{e}_i|\vec{e}_0, \Sigma_0) & \text{if } c_i = 0 \\ \mathcal{N}(\vec{e}_i|\vec{e_{c_i}}, \Sigma_v) & \text{otherwise} \end{cases} \quad (3)$$

Here, $\vec{e}_0$ is the base event vector for when $c_i = 0$. Σ_0 and Σ_v are covariance matrix for when an event occurs by their own initiative or occurs by past event. From the above definitions, we can calculate the intensity of the event vector $\vec{e}$ at time t as follows,

$$\lambda_{\vec{e}}(t) = \mu\mathcal{N}(\vec{e}|\vec{e}_0, \Sigma_0) + \sum_{t_j<t} \lambda(t|j)\mathcal{N}(\vec{e}|\vec{e}_j, \Sigma_v). \quad (4)$$

The total intensity of GMHP can be obtained by integrating the above intensity with the event vector $\vec{e}$.

$$\lambda(t) = \int_{\mathbb{R}^D} \lambda_{\vec{e}}(t)\, d\vec{e} = \mu + \sum_{t_j<t} \lambda(t|j). \quad (5)$$

5.1.2 Parameter estimation

From equation 1, the likelihood of the observed event stream can be computed as follows,

$$f(D|\theta) = e^{-\Lambda(T)} \prod_{i=1}^{n} \sum_{0\le j<i} p_j(\vec{e_i})\lambda(t_i|j), \quad (6)$$

where $\Lambda(T) = \mu T + \sum_{i=1}^{n} \alpha(1 - e^{-\beta(T-t_i)})$.

Since the likelihood of GMHP is hard to maximize, instead of using the likelihood, we define a likelihood with the given parent events as follows,

$$\begin{aligned} f(D|C,\theta) =& e^{-\Lambda(T)} \times \prod_{i=1}^{n}\{(\mu\mathcal{N}(\vec{e}_i|\vec{e}_0, \Sigma_0))^{C_{i0}} \times \\ & \prod_{j=1}^{i-1}(\alpha\beta e^{\beta(t_i-t_j)}\mathcal{N}(\vec{e}_i|\vec{e}_j, \Sigma_v))^{C_{ij}}\}, \end{aligned} \quad (7)$$

where C_{ij} becomes 1 when $c_i = j$ and 0 otherwise. By maximizing equation 7, we can estimate the parameter $\theta = \{\mu, \alpha, \vec{e}_0, \Sigma_0, \Sigma_v\}$. The inference step of the parent events is described in section 6.

5.2 Modeling a Mixture of GMHP with the HDP

When clustering a mixture of streams using the Hawkes process, the exponential triggering function prevents two events with a large time difference from being assigned to the same global cluster. To solve this problem, (Mavroforakis et al., 2017) uses the HDP instead of using the Dirichlet process used in (Du et al., 2015). The hierarchy structure of the HDP assigns a cluster label with a probability proportional to the size of the cluster. This allows assignment of two events with a large time difference to the same cluster. For the same reason, we use the HDP to model mixture of the GMHP. We consider each GMHP in mixture as a table in the Chinese Restaurant Franchise metaphor. Since the intensity of k'th GMHP, $\lambda_k(t)$ represents how likely an event occurs in table k at time t, we use the intensity as the number of customers in the CRF metaphor. The whole generative process of HD-GMHP is as follows.

1. Initialize the number of local clusters $K = 0$, the number of global clusters $M = 0$.
2. For $n \in 1, 2, ..., N$
   (a) Draw t_n from Hawkes($\lambda_0 + \sum_{k=0}^{K} \lambda_k$)
   (b) Draw z_n as follows.
   $$z_n \sim \lambda_0 \delta(K+1) + \sum_{k=1}^{K} \lambda_k \delta(k) \quad (8)$$
   (c) If $z_n = K + 1$, assign global cluster x_n, which is interpreted as parameter(θ_{x_n}) for local cluster z_n, and Increment K. Here, N_m is number of local cluster in global cluster m.
   $$x_n \sim \gamma\delta(M+1) + \sum_{m=1}^{M} N_m \delta(m) \quad (9)$$
   (d) If $x_n = M + 1$, increment M and draw new parameter as follows.
   $\alpha_{x_n} \sim \Gamma(\alpha_a, \beta_a)$, $\mu_{x_n} \sim \Gamma(\alpha_\mu, \beta_\mu)$
   $\frac{1}{\Sigma_0^{x_n}} \sim \Gamma(\alpha_0, \beta_0)$, $\frac{1}{\Sigma_v^{x_n}} \sim \Gamma(\alpha_v, \beta_v)$
   $\vec{e}_{0,x_n} \sim \mathcal{N}(\vec{e}_0, \Sigma_0^{x_n}/\vec{\lambda}_{e0})$
   Note that we assume that the covariance matrix $\Sigma_v^{x_n}$, and $\Sigma_0^{x_n}$ are diagonal.
   (e) Draw c_n and $\vec{e}_n$. Here, $g_{x_n}(t) = \alpha_{x_n}\beta e^{-\beta(t_n - t)}$.
   $$c_n \sim \mu_{x_n}\delta(N_{z_n} + 1) + \sum_{j=1}^{N_{z_n}} g_{x_n}(t_j)\delta(j) \quad (10)$$
   if $c_n = N_{z_n} + 1$, then replace c_n with 0 and sample event vector.
   $$\vec{e}_n \sim \begin{cases} \mathcal{N}(\vec{e}_{0,x_n}, \Sigma_0^{x_n}) & \text{if } c_n = 0 \\ \mathcal{N}(\vec{e}_{c_n}, \Sigma_v^{x_n}) & \text{otherwise} \end{cases} \quad (11)$$

λ_0, γ, $\vec{e}_0, \vec{\lambda}_{e0}$, (α_a, β_a), (α_μ, β_μ), (α_0, β_0), and (α_v, β_v) are the hyperparameters used in HD-GMHP.

6 Inference

To infer the latent variables z and x for each event from an observed event stream $s_o^{1:n}$ where $s_o^i = (t_i, \vec{e}_i)$ with observation time T, we propose an online inference algorithm with Sequential Monte Carlo (SMC) (Doucet et al., 2001). To calculate the posterior of the latent variables z and x for each timestamp t_i in the inference, we need the estimated parameter to calculate the intensity at each time t_i, $\lambda(t_i)$. As described in section 5.1.2, the parameter estimation step needs the parent event information. In our proposed inference, the parent events are inferred from SMC. The inference algorithm is summarized in algorithm 1.

Algorithm 1 Inference

```
Input: Stream data $S_o$
Initialize $w_1^i = \frac{1}{P}, i \in \{1, 2, ..., P\}$.
for $n = 1$ to $N$ do
    for $i = 1$ to $P$ do
        Update $\Theta$ as described in section 6.2.
        Sample $(x, z, c)_n^i$ with equation 16, 10
        Update $w_n^i$ with equation 17
    end for
    Normalize $w_n^{1:P}$
    if $||w_n||_2^{-2} < thresh$ then
        Resample particles
    end if
end for
```

6.1 Sequential Monte Carlo with parent event inference

To approximate the posterior of the latent variables, SMC samples the latent variables from the proposal distribution and calculates the weight of each sampled variables which is called the particle weight. To infer the parent event in SMC, we define the particle weight of our modified SMC as follows:

$$w_n^i = \frac{p(\psi_{1:n}^i | s_o^{1:n})}{q(\psi_{1:n}^i | s_o^{1:n})} \frac{p(c_{1:n}^i | \psi_{1:n}^i, s_o^{1:n})}{q(c_{1:n}^i | \psi_{1:n}^i, s_o^{1:n})} \quad (12)$$

Here, ψ_n^i is (x_n^i, z_n^i). Let the left part on the right hand term and right part on the right hand term of the equation 12 are $w_{\psi_n}^i$ and $w_{c_n}^i$. Then the terms can be calculated by $w_{\psi_n}^i = \eta_\psi w_{\psi_{n-1}}^i$ and $w_{c_n}^i = \eta_c w_{c_{n-1}}^i$, where the η_ψ is

$$\frac{p(\vec{e}_n, t_n, \psi_n | s_o^{1:n-1}, \psi_{1:n-1}^i)}{q(\psi_n^i | \psi_{1:n-1}^i, s_o^{1:n})}. \quad (13)$$

and η_c is

$$p(\vec{e}_n | t_n, s_o^{1:n-1}, \delta_{1:n}^i) \frac{p(c_n^i | t_n, \delta_{1:n-1}^i, s_o^{1:n-1}, \psi_n^i)}{q(c_n^i | \delta_{1:n-1}^i, s_o^{1:n}, \psi_n^i)}. \quad (14)$$

Here, δ_n^i is (x_n^i, z_n^i, c_n^i).

We use $p(\psi_n|\psi_{1:n-1}, s_o^{1:n})$ as the proposal distribution of ψ_n^i in the equation 13 to minimize the variance of w_n^i (Doucet et al., 2000) and $p(c_n|\delta_{1:n-1}, \psi_n, t_n, s_o^{1:n-1})$ as the proposal distribution of c_n^i in the equation 14. From the above proposal distribution, $\eta_{c_n}^{\ i}$ can be calculated as $\eta_{c_n}^{\ i} = p(\vec{e}_n|t_n, s_o^{1:n-1}, c_{1:n}^i, \psi_{1:n})$ and $\eta_{\psi_n}^{\ i}$ can be calculated by the following form .

$$\begin{aligned}\eta_{\psi_n}^{\ i} =& p(t_n|\psi_{1:n-1}, z_n, s_o^{1:n-1}) \\ & \times \sum_{z_n} (p(\vec{e}_n|\psi_{1:n-1}, z_n, t_n, s_o^{1:n-1}) \\ & \times p(z_n|t_n, \psi_{1:n-1}, s_o^{1:n-1}))\end{aligned} \tag{15}$$

From the proposal distribution of ψ_n^i, we can sample ψ_n^i as follows:

$$\begin{aligned}p(\psi_n|rest) \propto & p(\vec{e}_n|\psi_{1:n}, t_n, s_o^{1:n-1}) \\ & \times p(\psi_n|\psi_{1:n-1}, t_n, s_o^{1:n-1}) \\ & \times p(t_n|\psi_{1:n-1}, s_o^{1:n-1})\end{aligned} \tag{16}$$

Here, the term $p(\vec{e}_n|\psi_{1:n}, t_n, s_o^{1:n-1}) \times p(\psi_n|\psi_{1:n-1}, t_n, s_o^{1:n})$ can be simply calculated by the student's t-distribution derived from the conjugate relation between the parameter $\{\vec{e}_{k0}, \Sigma_{0,k}, \Sigma_{v,k}\}$ and the normal-inverse-gamma and inverse-gamma prior in the generative process of HD-GMHP.

From $\eta_{c_n}^{\ i} = p(\vec{e}_n|t_n, s_o^{1:n-1}, c_{1:n}^i, \psi_{1:n})$ and 15, the particle weight can be updated by the following,

$$\begin{aligned}w_n^i \propto & w_{n-1}^i \\ & \times p(t_n|s_o^{1:n-1}, \psi_{1:n}^i) p(\vec{e}_n|c_n, t_n, s_o^{1:n-1}, \psi_{1:n}) \\ & \times \sum_{z_n} (p(\vec{e}_n|z_n, \psi_{1:n-1}, t_n, s_o^{1:n-1}) \\ & \times p(z_n|t_n, \psi_{1:n-1}, s_o^{1:n-1})).\end{aligned} \tag{17}$$

When calculating the probability of t_n in 17, we assume the parameters $\mu_{1:K}, \alpha_{1:K}$ are given (Carvalho et al., 2010). From the likelihood of GMHP, the probability term $p(t_n|\psi_{1:n}, s_o^{1:n-1})$ in equation 17 can be calculated by $\lambda_{z_n}(t_n)e^{-\Lambda(t_n, t_{n-1})}$. Where $\Lambda(t_n, t_n - 1)$ is

$$\begin{aligned}& \lambda_0(t_n - t_{n-1}) + (t_n - t_{n-1}) \sum_{k=1}^{K} \mu_k \\ & + \frac{1}{\beta}(1 - e^{-\beta(t_n - t_{n-1})}) \sum_{k=1}^{K} \lambda_k(t_{n-1}).\end{aligned} \tag{18}$$

In the case of the probability term $p(\vec{e}_n|c_n, rest)$ and $p(\vec{e}_n|z_n, rest)$ in 17, as explained in the sampling process of ψ_n, we can calculate the terms by student's t-distribution. With the particle weight update rule 17 and the parameter update rule described in section 6.2, we infer latent variables with algorithm 1.

6.2 Updating Parameter

From the equation 7 and the prior of the parameters used in GMHP, we can estimate the parameters by following form.

$$\alpha_m = \frac{\alpha_a - 1 + \sum_{x_i = m} \sum_{0<j<i} C_{ij}}{\beta_a + \sum_{x_i=m} (1 - e^{-\beta(T - t_i)})} \tag{19}$$

$$\mu_m = \frac{\alpha_\mu - 1 + \sum_{x_i=m} C_{i0}}{\beta_\mu + \sum_{\theta_k = \Theta_m} (T - t_{0,k})} \tag{20}$$

$$\vec{e}_{0,m} = \frac{\vec{e}_0 \circ \vec{\lambda}_{e0} + \sum_{x_i=m} C_{i0}\vec{e}_i}{\vec{\lambda}_{e0} + \sum_{x_i=m} C_{i0}} \tag{21}$$

$$\begin{aligned}diag(\Sigma_0^m) =& \{\vec{\lambda}_{e0} \circ (\vec{e}_{0,m} - \vec{e}_0)^2 + 2\vec{\beta}_0 \\ & + \sum_{x_i=m} C_{i0}(\vec{e}_i - \vec{e}_{0,m})^2\} \\ & \times \{2\vec{\alpha}_0 + 3 + \sum_{x_i=m} C_{i0}\}^{-1}\end{aligned} \tag{22}$$

$$diag(\Sigma_v^m) = \frac{2\vec{\beta}_v + \sum_{x_i=m} \sum_{0<j<i} C_{ij}(\vec{e}_i - \vec{e}_j)^2}{2 + 2\vec{\alpha}_v + \sum_{x_i=m} \sum_{0<j<i} C_{ij}} \tag{23}$$

6.3 Approximation

To reduce the computation time in the inference algorithm, we use several approximation strategies.

6.3.1 Marginal distribution Approximation

To calculate $p(\vec{e}_n|z_n, \psi_{1:n-1}, t_n, {s_o}^{1:n-1})$ in the equation 17, we need marginalization of $p(\vec{e}_n, c_n|z_n, \psi_{1:n-1}, t_n, s_o^{1:n-1})$ which takes time complexity of O(n of events in z_n) and cause the time complexity of the equation 17 to be $O(n)$. To reduce the time complexity, we note that event vector $\vec{e}_n$ is sampled from a Gaussian mixture that the influence of each Gaussian distribution is exponentially decreases. We assume the marginal distribution $p(\vec{e}_n|z_n, \psi_{1:n-1}, t_n, s_o^{1:n-1})$ can be approximated to $p(\vec{e}_n|c_{1:n} = 0, z_n, \psi_{1:n-1}, t_n, s_o^{1:n-1})$. From the approximation, we can calculate the posterior predictive with student's t-distribution. The result of approximation is as follows,

$$p(\vec{e}_n|z_n, \psi_{1:n-1}, t_n, s_o^{1:n-1}) = t_{\nu_n}(\vec{e}_n|\vec{m}_n, \frac{\vec{\kappa}_n + 1}{\vec{\kappa}_n \nu_n}\vec{S}_n), \quad (24)$$

where

$$\nu_n = 2\alpha_0 + N_{z_n}, \quad \kappa_n = \vec{\lambda}_{e_0} + N_{z_n},$$

$$\vec{m}_n = \frac{\vec{\lambda}_{e_0} \circ \vec{e}_0 + \sum_{z_i = z_n} \vec{e}_i}{\vec{\kappa}_n},$$

$$\vec{S}_n = 2\beta_0 + \sum_{z_i = z_n} \vec{e}_i^{\,2} + \vec{\lambda}_{e_0} \circ \vec{e}_0^{\,2} - \kappa_n \vec{m}_n^{\,2}.$$

To calculate $p(\vec{e}_n|c_n, t_n, s_o^{1:n-1}, \psi_{1:n})$ in the equation 17, we need to calculate posterior predictive for each past event. To reduce the computation time in the process of calculation, we approximate the probability distribution $p(\vec{e}_n|c_n, t_n, s_o^{1:n-1}, \psi_{1:n})$ as follows.

$$p(\vec{e}_n|c_n, t_n, s_o^{1:n-1}, \psi_{1:n}) \approx \begin{cases} \mathcal{N}(\vec{e}_{0,x_n}, \Sigma_0^{x_n}) & \text{if } c_n = 0 \\ \mathcal{N}(\vec{e}_{c_n}, \Sigma_v^{x_n}) & \text{otherwise} \end{cases} \quad (25)$$

6.3.2 Sampling c_n from recent W events

Sampling c_n has time complexity of $O(N_{z_n})$. To reduce the time complexity to $O(1)$, we sample c_n from recent W events in the local cluster z_n.

7 Experiment

In this section, we demonstrate the narrative reconstruction and thread reconstruction performance of our model on a corpus of the New York Times articles and the Wikipedia conversation dataset.

7.1 Dataset

New York Times Dataset: We collected 112,538 New York Times news articles from January 2016 to July 2017. The dataset contains the text, timestamp, the news section, and the keywords. These keywords are semantic tags specified by the newsroom to indicate the main topics of the articles. We select news articles in sections "U.S.", "World", "Opinion", and "Sports" that contain at least one of the top ten most frequently used keywords. The statistics of these keywords are described in table 1. Further, we select articles with more than ten words in its body. The final number of articles used in our experiment is 16,858. The dataset is publicly available [1].

Table 1: Statistics of keywords. "N" column lists the number of articles with the corresponding keyword.

Keyword	N
Trump, Donald J	7940
Presidential Election of 2016	5737
United States Politics and Government	4986
Republican Party	2371
Clinton, Hillary Rodham	2330
Baseball	2058
United States International Relations	1817
Terrorism	1618
Obama, Barack	1551
Russia	1400

Wikipedia Conversation Dataset is released by (Danescu-Niculescu-Mizil et al., 2012). The dataset contains the timestamp, the initial post of the conversation, "reply to" link information, and the text information of each post in conversation threads in Wikipedia talk pages. We select threads that have ten or more posts from September 2010 to December 2010. The final number of posts used in our experiment is 2,004 and the final number of threads is 154.

7.2 Preprocessing

To apply our model to the real world datasets, we represent each event with time information and an event vector. For the time information, we take the first article or post and set the time as zero, the last article or post as one, and scale the timestamps of all other articles and posts accordingly. To extract the event vectors, we use different vectorization methods for the two datasets. For the NYT dataset, we use the document topic vector from LDA (Blei et al., 2003). For the Wikipedia dataset, because there are only a few words in each post, we cannot use the LDA topic vector, so we use the averaged word embedding vector (Mikolov et al., 2013) of the words used in each post.

---

[1] `https://github.com/yeonsw/NYT-dataset`

Table 2: Narrative reconstruction results in NYT dataset and post grouping results in Wikipedia conversation dataset.

		AMI	ARI
NYT	LDA + DBSCAN	0.0627	0.0117
	HDP + DBSCAN	0.0260	0.0203
	HDHP	0.1768	0.0746
	HD-GMHP (100D)	**0.2479**	**0.1416**
Wiki	W2V + DBSCAN	0.0055	0.0001
	HDHP	0.4240	0.3512
	HD-GMHP (100D)	**0.5848**	**0.3834**

7.3 Task

Narrative reconstruction: To demonstrate the narrative reconstruction performance of our model, we apply the inference method to our corpus of NYT articles. We use a set of multiple keywords of each article as the ground truth label. Then we run our model and consider the set of articles with the same global cluster information as one narrative. We compare the results with the ground truth labels using the common clustering metrics AMI and ARI (Hubert and Arabie, 1985; Vinh et al., 2010) to evaluate the narrative reconstruction performance of our model. We compare HD-GMHP with the following baselines: LDA and HDP with DBSCAN, and the Hierarchical Dirichlet Hawkes Process (HDHP) (Mavroforakis et al., 2017) which is a state-of-the-art model for text and continuous timestamps of an event. Also, to measure the similarity of each recovered narrative and the ground truth narrative, we use the F1 score of the top ten narratives.

Thread reconstruction: In this experiment, we use two evaluation criteria. One is post grouping and the other is reply structure recovery, which is simply the recovery of the child nodes. Here, we use a different child node recovery task compared to the child node recovery used in previous research. In our task, we do not give the initial post of each thread, while previous research does. This makes thread reconstruction problem more general and more difficult.

In post grouping, we use the initial post of each of the posts as the ground truth label and measure the clustering metrics used in the NYT dataset. In the child node recovery experiment, we use the parent event information inferred from our method as the recovered tree structure of the threads. We measure the performance with node precision and node recall metrics (Wang et al., 2011a; Dehghani et al., 2013). We compare our model with the following baselines: HDHP, and a naive baseline that reconstructs threads in the form of a single linked list of posts in chronological order.

Table 3: F1-score of each label

Label	N	HD-GMHP	HDHP
Baseball	2011	0.8114	0.8899
Trump, Donald &Politics and Government	1664	0.1833	0.2157
Terrorism	1260	0.6052	0.4059
Trump, Donald &Election	1110	0.2537	0.1939
Trump, Donald	994	0.0975	0.1227
Politics and Government	822	0.1677	0.1215
Clinton, Hillary &Election &Trump, Donald	755	0.1754	0.1402
Election	714	0.1378	0.1280
Clinton, Hillary &Election	665	0.1669	0.1157
Russia	637	0.3177	0.2223
Micro F-score	N/A	**0.2874**	0.2189
Macro F-score	N/A	**0.3637**	0.3165

7.4 Metrics

AMI, ARI are commonly used to measure clustering performance (Hubert and Arabie, 1985; Vinh et al., 2010). $\mathbf{P_{node}}$, $\mathbf{R_{node}}$ measure local similarity between two thread structures (Wang et al., 2011a).

$$\mathrm{P_{node}} = \frac{1}{N} \sum_{i=1:N} \frac{|\mathrm{child_{GT}}(i) \cap \mathrm{child_{E}}(i)|}{|\mathrm{child_{E}}(i)|}$$

$$\mathrm{R_{node}} = \frac{1}{N} \sum_{i=1:N} \frac{|\mathrm{child_{GT}}(i) \cap \mathrm{child_{E}}(i)|}{|\mathrm{child_{GT}}(i)|}$$

where, $\mathrm{child_{GT}}(i)$ and $\mathrm{child_{E}}(i)$ are the sets of children of node i in the ground truth thread structure and the recovered thread structure, respectively. The author (Wang et al., 2011a) also proposed $\mathrm{P_{path}}$, $\mathrm{R_{path}}$ to measure the similarity of the global structure of two threads. The path metrics are sensitive to the recovered initial post of each thread, but since we do not give the initial post of each thread in our experiment, the path metrics are

Table 4: Reply structure recovery results in Wikipedia conversation dataset.

	P_{node}	R_{node}	$F1_{node}$
Naive Baseline	0.3223	**0.6501**	0.4310
HDHP	0.5598	0.5834	0.5714
HD-GMHP	**0.6433**	0.5468	**0.5911**

no longer proper in our experiment. So we measure the node metrics only.

7.5 Results

Table 2 shows the clustering accuracy of our method and the baseline methods in real world datasets. We average the results with five runs for each model. The highest value for each metric is indicated with boldface. From the results, we establish that our model outperforms the baseline methods in both the NYT narrative reconstruction task and the Wikipedia thread reconstruction task.

For the NYT, to see the accuracy of our model in more detail, we compute and show the F-scores for the top ten most frequent labels and the micro and macro averages in table 3. To compute the F-score between the true labels and the recovered cluster labels, we select the cluster with the highest F-score as the corresponding cluster. From the results, we establish that our model performs better than the baseline model, HDHP.

Table 4 shows the thread reconstruction results of our model and the baseline models in the Wikipedia conversation dataset. Since the HDHP model does not infer the parent event, we reconstruct threads in the form of chronologically ordered linked list of posts in each local cluster that inferred from HDHP. From the $F1_{node}$ score of the results, we establish our model performs better than other baseline models.

To demonstrate the robustness of HD-GMHP on dimensional change of the input vector, we measure the performance of each task in using 50, 100, and 150 dimensional vectors. The results are described in table 5 and 6. From the results, we verify there are no drastic changes in performance in both the NYT dataset and the Wikipedia dataset.

8 Conclusion

In this paper, we defined the narrative and thread reconstruction problems as clustering problems. To cluster the event streams with continuous time information and triggering event information, we proposed the Gaussian Marked Hawkes process that models event streams with additional event information represented in a vector form. Furthermore, we combined our model GMHP with the HDP to cluster event streams (HD-GMHP). We showed that our model performs better than several baseline methods in both narrative reconstruction in a dataset of NYT articles and thread reconstruction in a dataset of Wikipedia conversations.

Table 5: Model Robustness on dimensional change of input vectors in NYT dataset.

	AMI	ARI
HD-GMHP (50D)	0.2310	0.1518
HD-GMHP (100D)	0.2479	0.1416
HD-GMHP (150D)	0.2421	0.1191

Table 6: HD-GMHP model robustness on dimensional change of input vector in Wikipedia conversation dataset.

	AMI	ARI	P_{node}	R_{node}
50D	0.5836	0.3782	0.6466	0.5554
100D	0.5848	0.3834	0.6433	0.5468
150D	0.5948	0.3670	0.6450	0.5473

Acknowledgments

Yeon Seonwoo is supported by NCSoft Corporation. Sungjoon Park and Alice Oh are supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921).

References

Amr Ahmed, Qirong Ho, Jacob Eisenstein, Eric Xing, Alexander J Smola, and Choon Hui Teo. 2011. Unified analysis of streaming news. In *WWW*.

Amr Ahmed and Eric Xing. 2008. Dynamic non-parametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering. In *SIAM International Conference on Data Mining*.

Erik Aumayr, Jeffrey Chan, and Conor Hayes. 2011. Reconstruction of threaded conversations in online discussion forums. In *ICWSM*.

A Balali, H Faili, and M Asadpour. 2014. A supervised approach to predict the hierarchical structure of conversation threads for comments. *The Scientific World Journal*, 2014.

David M Blei and Peter I Frazier. 2011. Distance dependent chinese restaurant processes. *Journal of Machine Learning Research*, 12(Aug):2461–2488.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Carlos M Carvalho, Michael S Johannes, Hedibert F Lopes, Nicholas G Polson, et al. 2010. Particle learning and smoothing. *Statistical Science*, 25(1):88–106.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *WWW*.

Mostafa Dehghani, Azadeh Shakery, Masoud Asadpour, and Arash Koushkestani. 2013. A learning approach for email conversation thread reconstruction. *Journal of Information Science*, 39(6):846–863.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. 2001. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer.

Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. 2000. Rao-blackwellised particle filtering for dynamic bayesian networks. In *UAI*.

Nan Du, Mehrdad Farajtabar, Amr Ahmed, Alexander J Smola, and Le Song. 2015. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *SIGKDD*.

Alan Hawkes. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, pages 83–90.

Xinran He, Theodoros Rekatsinas, James Foulds, Lise Getoor, and Yan Liu. 2015. Hawkestopic: A joint model for network inference and topic modeling from text-based cascades. In *ICML*.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2(1):193–218.

Tomoharu Iwata, Amar Shah, and Zoubin Ghahramani. 2013. Discovering latent influence in online social activities via shared cascade poisson processes. In *SIGKDD*.

Martin Jankowiak and Manuel Gomez-Rodriguez. 2017. Uncovering the spatiotemporal patterns of collective social activity. In *SIAM International Conference on Data Mining*.

Su Nam Kim, Li Wang, and Timothy Baldwin. 2010. Tagging and linking web forum posts. In *CoNLL*.

Annie P Louis and Shay B Cohen. 2015. Conversation trees: A grammar model for topic structure in forums. In *EMNLP*.

Michal Lukasik, PK Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *ACL*.

Charalampos Mavroforakis, Isabel Valera, and Manuel Gomez-Rodriguez. 2017. Modeling the dynamics of learning activity on the web. In *WWW*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Yu Rong, Hong Cheng, and Zhiyu Mo. 2015. Why it happened: Identifying and modeling the reasons of the happening of social events. In *SIGKDD*.

Siliang Tang, Fei Wu, Si Li, Weiming Lu, Zhongfei Zhang, and Yueting Zhuang. 2015. Sketch the storyline with charcoal: A non-parametric approach. In *IJCAI*.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854.

Hongning Wang, Chi Wang, ChengXiang Zhai, and Jiawei Han. 2011a. Learning online discussion structures by conditional random fields. In *SIGIR*.

Li Wang, Marco Lui, Su Nam Kim, Joakim Nivre, and Timothy Baldwin. 2011b. Predicting thread discourse structure over technical web forums. In *EMNLP*.

William Yang Wang, Yashar Mehdad, Dragomir R Radev, and Amanda Stent. 2016. A low-rank approximation approach to learning joint embeddings of news stories and images for timeline summarization. In *NAACL*.

Hongteng Xu and Hongyuan Zha. 2017. A dirichlet mixture model of hawkes processes for event sequence clustering. In *NIPS*.

Shize Xu, Shanshan Wang, and Yan Zhang. 2013. Summarizing complex events: a cross-modal solution of storylines extraction and reconstruction. In *EMNLP*.

Deyu Zhou, Haiyang Xu, Xin-Yu Dai, and Yulan He. 2016. Unsupervised storyline extraction from news articles. In *IJCAI*.

Investigating the Role of Argumentation in the Rhetorical Analysis of Scientific Publications with Neural Multi-Task Learning Models

Anne Lauscher,[1,2] **Goran Glavaš,**[1] **Simone Paolo Ponzetto,**[1] and **Kai Eckert**[2]
[1]Data and Web Science Research Group
University of Mannheim, Germany
[2]Web-based Information Systems and Services
Stuttgart Media University, Germany
{anne, goran, simone}@informatik.uni-mannheim.de
{lauscher, eckert}@hdm-stuttgart.de

Abstract

Exponential growth in the number of scientific publications yields the need for effective automatic analysis of rhetorical aspects of scientific writing. Acknowledging the argumentative nature of scientific text, in this work we investigate the link between the argumentative structure of scientific publications and rhetorical aspects such as discourse categories or citation contexts. To this end, we (1) augment a corpus of scientific publications annotated with four layers of rhetoric annotations with argumentation annotations and (2) investigate neural multi-task learning architectures combining argument extraction with a set of rhetorical classification tasks. By coupling rhetorical classifiers with the extraction of argumentative components in a joint multi-task learning setting, we obtain significant performance gains for different rhetorical analysis tasks.

1 Introduction

Scientific publications, as "tools of persuasion" in research (Gilbert, 1977), are carefully composed documents written to convince the reader of the validity and merit of the researchers' work. As such, they are inherently argumentative and often adhere to well-trodden rhetorical patterns and argumentation schemes of the respective research field. The accelerated growth of scientific literature (Bornmann and Mutz, 2015) makes exploration and analysis of relevant publications increasingly difficult. This yields the need for automatic analyses of these documents, including their argumentative and rhetorical structure.

Accordingly, computational models already support publication analysis tasks, e.g., classification of citation purpose and polarity (Jha et al., 2017; Lauscher et al., 2017b, *inter alia*) and classification of (sentential) discourse roles (Teufel et al., 1999; Liakata et al., 2010, *inter alia*). Further, rhetorical predictions at the (sub-)sentence level obtained using these models have been shown useful in higher-level downstream tasks such as publication classification (Teufel et al., 1999), (extractive) publication summarization (Cohan and Goharian, 2015), and research trend prediction (McKeown et al., 2016).

To allow for the holistic analysis of scientific publications with respect to the interactions between different rhetorical aspects of scientific text Fisas et al. (2016) created a corpus of scientific publications with manual annotations of several high-level rhetorical aspects of scientific writing (e.g., sentence-level discourse roles), but without annotations of the argumentative structure of publications. Despite (1) scientific texts being inherently argumentative (Gilbert, 1976), (2) the existence of theoretical argumentative frameworks (Toulmin, 2003; Kirschner et al., 2015), and (3) a wide range of argument extraction models in other domains (e.g., debates or essays, see Palau and Moens (2009); Habernal and Gurevych (2017), *inter alia*), there is still very little work on automatic argumentation mining from scientific literature. Consequently, there has been no work analyzing associations between argumentation and other rhetorical constructs in scientific writing, although such dependencies exist. Consider the following example:

> *"In general, our OMR preserves the high frequency content of the motion quite well [claim], since inverse rate control is directed by Jacobian values [data]."*

Here, the authors make a *claim* (underlined text) about their approach and support it with a technical fact (*data*) about the method (wave-underlined text). At the same time, regarding other rhetorical constructs, this sentence is stating the subjective aspect of *advantage* (of the proposed method), belongs to the discourse category of *outcome* (of the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3326–3338
Brussels, Belgium, October 31 - November 4, 2018.

authors' work), and may be considered *relevant* for the (extractive) summary of the publication. We argue that these rhetorical dimensions are interconnected and that fine-grained argumentation underpins other rhetorical layers in scientific text. For example, sentences stating an *advantage* of a method are likely to be argumentative and may contain *claims* that should be included in the summary.

Assuming that argumentation guides rhetorics in scientific text, we investigate neural multi-task learning (MTL) models which couple argument extraction with several other rhetorical analysis tasks. To this end, we augment the existing corpus of scientific publications (Fisas et al., 2016), containing several layers of rhetorical annotations, with an additional layer of argumentative components and relations. We then explore two neural MTL architectures based on shared recurrent encoders, intra-sentence attention, and private task-specific classifiers and couple the neural architectures with a joint MTL objective with uncertainty-based weighting of task-specific losses (Kendall et al., 2018). We validate our approach by testing that it outperforms traditional machine learning models in single-task settings. We finally show that coupling rhetorical analysis tasks with argument extraction using MTL models significantly improves the results for the rhetorical analysis tasks.

Contributions. We create the first corpus of scientific publications in English annotated with fine-grained argumentative structures and carry out the first study on dependencies between different rhetorical dimensions in scientific writing. Using MTL models, we show that argumentation informs other rhetorical analysis tasks. Finally, in the context of MTL research, our results indicate that the dynamic uncertainty-based loss weighting (Kendall et al., 2018) is beneficial for high-level natural language processing tasks.

2 Related Work

We provide an overview of (1) studies analyzing rhetorical aspects in scientific publications and (2) a large body of work on argumentation mining.

2.1 Rhetorical Analysis of Scientific Texts

Previous work has analyzed a number of rhetorical aspects of scientific publications. Teufel et al. (1999, 2009) analyzed the discourse structure of scientific publications. They annotated sentences with discourse categories named *argumentative zones*. Liakata et al. (2010) proposed a more general discourse scheme dubbed *core scientific concepts* and in subsequent work (Liakata et al., 2012) trained a conditional random fields (CRF) model to assign discourse labels to text spans. Several authors focused on tasks relating to citations: extraction of citation context (e.g., Abu-Jbara et al., 2013; Jha et al., 2017), classification of citation polarity (e.g., Athar, 2011) and purpose (e.g., Teufel et al., 2006; Jochim and Schütze, 2012), and the automatic detection of referenced parts of the cited publication (Jaidka et al., 2017). Both discourse and citation information have been exploited for summarizing scientific publications (Cohan and Goharian, 2015; Teufel and Moens, 2002; Abu-Jbara and Radev, 2011; Chen and Zhuge, 2014; Lauscher et al., 2017a). Intuitively, citation contexts may contain information relevant to the summary. Similarly, summaries commonly contain sentences with diversified discourse properties.

Fisas et al. (2016) provided different layers of rhetorical annotations on the same corpus of scientific text.Their Dr. Inventor Corpus is annotated with a combination of existing discourse annotation schemes (Teufel et al., 2009; Liakata et al., 2010) and citation-based annotations. Despite the argumentative nature of scientific texts, the Dr. Inventor Corpus contains no annotations of argumentative components such as claims. Several computational studies followed, addressing the rhetorical tasks corresponding to the layers of the Dr. Inventor Corpus (Ronzano and Saggion, 2015, 2016; Accuosto et al., 2017), but none of them investigated dependencies between different tasks.

The work of Kirschner et al. (2015) is the closest to ours, since they also annotated scientific publications with fine-grained argumentation. However, their corpus is in German and contains no annotations of other rhetorical dimensions. Moreover, their corpus is significantly smaller than the Dr. Inventor Corpus (Fisas et al., 2016). In contrast, we augment the Dr. Inventor Corpus with an argumentation layer, effectively allowing for combinations of argumentation extraction and other rhetorical analysis tasks in MTL settings.

2.2 Argumentation Mining

Argumentation mining (AM) refers to extracting (and ideally understanding) arguments from natural language text (Lippi and Torroni, 2015, 2016) and includes tasks like argument detection (Palau and Moens, 2009), argument component identification (Daxenberger et al., 2017), and argument relation classification (Boltužić and Šnajder, 2014). In their pioneering work on automatic AM, Palau and Moens (2009) discriminated argumentative from non-argumentative sentences and proposed a rule-based approach for extracting argumentative structures in documents. Habernal and Gurevych (2016, 2017) extracted argumentative components from online discussions. They framed the argumentative component extraction as a sequence labeling task and applied structured SVMs as a learning model.

Recent work started exploiting dependencies between AM tasks using global optimization (Peldszus and Stede, 2015; Persing and Ng, 2016; Stab et al., 2014) and MTL models (Eger et al., 2017; Niculae et al., 2017). Peldszus and Stede (2015) used decoding based on minimum spanning trees to jointly predict argumentative segments and their types as well as argumentative relations, to generate an argumentation graph from text. Persing and Ng (2016) and Stab and Gurevych (2017) similarly produced argumentative structures by globally optimizing local predictions of argumentative components and relations. Potash et al. (2017) proposed a neural architecture based on a pointer network for jointly predicting types of argumentative components and identifying argumentative relations. In a similar effort, Eger et al. (2017) combined the AM tasks using the MTL framework of Søgaard and Goldberg (2016). Remedying for data sparsity, Schulz et al. (2018) treated different argumentation formalisms as different tasks and combined respective extraction tasks and datasets in a MTL setting. In contrast to these efforts that combine several AM subtasks or formalisms with joint optimization and MTL models, in this work we examine the dependencies between argumentative components and other rhetorical aspects of scientific writing.

3 Data Annotation

We first briefly describe the Dr. Inventor Corpus (Fisas et al., 2016), which we augment with argumentative annotations. We then explain in more detail our argumentation annotation scheme and the annotation process.

3.1 Dr. Inventor Corpus

We chose the Dr. Inventor Corpus (Fisas et al., 2015, 2016) as a starting point for two reasons. First, containing 40 publications with a total of 10, 789 sentences, it is one of the largest corpora of scientific text manually labeled with rhetorical information. Secondly, it contains *four* different layers of rhetorical annotations: (1) a *discourse* layer, specifying discourse roles of sentences, (2) a *citation context* layer, specifying the textual context of citations, (3) a layer with *subjective aspect* categories assigned to sentences, and (4) a *summarization relevance* layer, indicating how relevant sentences are for the summary. The overview of labels for all annotation layers with the distribution of instances across labels is shown in Table 1. For more details on the original Dr. Inventor Corpus we refer the reader to (Fisas et al., 2015, 2016).

Annotation Layer	Labels	%
Discourse Role	*Background*	20
	Challenge	5
	Approach	57
	Outcome	16
	Future Work	2
Citation Purpose	*Criticism*	23
	Comparison	9
	Use	11
	Substantiation	1
	Basis	5
	Neutral	53
Subjective Aspect	*Advantage*	33
	Disadvantage	16
	Adv.-Disadv.	3
	Disadv.-Adv.	1
	Novelty	13
	Common Practice	32
	Limitation	2
Summarization Relevance	*Totally irrelevant*	66
	Should not appear	6
	May appear	14
	Relevant	6
	Very relevant	8

Table 1: Annotation layers of the Dr. Inventor Corpus (Fisas et al., 2016) with label distributions .

3.2 Argumentation Annotation Scheme

We considered several existing argumentation frameworks (e.g., Anscombre and Ducrot, 1983; Walton et al., 2008; Dung, 1995, *inter alia*) and selected the Toulmin's model (Toulmin, 2003) as a starting point for our study. We chose the Toulmin's model because: (1) it is a well-established in philosophy as well as in computer science (e.g, Freeman, 1991; Bench-Capon, 1998; Verheij, 2009, *inter alia*) and (2) it contains different types of argumentative components and relations between them into account, which is useful for fine-grained argumentative analyses.

To test the applicability of the framework for our purposes,we first carried out a small preliminary annotation round with two expert annotators and adjusted the annotation scheme according to their observations.

Argumentative components. We devised an adapted version of the Toulmin model,[1] containing the following argumentative components:

- *Background claim*: An argumentative statement related to the work of other authors, state-of-the-art methods, or common practices;

  "The range of breathtaking realistic 3D models is only limited by the creativity of artists and resolution of devices."

- *Own claim*: An argumentative statement about own work, covered by the publication itself;

  "Using our method, character authors may use any tool they like to author characters."

- *Data*: A fact that the authors state as evidence that either supports or contradicts a claim.

  "SSD is widely adopted in games, virtual reality, and other realtime applications due to its ease of implementation and low cost of computing."

Argumentative components are annotated as arbitrary spans of text (in terms of length, annotated components ranged from a single token to multiple sentences). Annotators were instructed to annotate the shortest possible span of text that completely captures the argumentative component. Thus, we do not bind arguments to sentences, i.e., we allow for fine-grained argumentative components.

Argumentative relations. Authors connect argumentative components in order to form convincing reasoning chains. To allow for the detection of long argumentation chains, we also annotated relations between argumentative components. Following proposals from previous work (Dung, 1995; Bench-Capon, 1998), we distinguish between three relation types:

- *Supports*: indicates that a *claim* component is supported by a data component or another claim. The (assumed) validity of the *supporting* component (data or claim) contributes to the validity of the *supported* claim.

- *Contradicts*: indicates that the validity of a claim decreases with the validity of another argumentative component. If an argumentative component is assumed to be true, the claim it contradicts is assumed to be false, and vice versa.

- *Same claim*: connects different mentions of what is essentially the same claim. It is common to repeat important claims (e.g., the central claim) of the work several times in the publication (*claim coreference*).

Further details about the annotation scheme can be found in the annotation guidelines we provided to our annotators.[2]

3.3 Annotation Procedure and Results

Annotation process. We hired four annotators for the task, one of whom we considered to be an *expert* annotator[3] and executed the process in two phases. In the first phase, we calibrated the annotators for the task in five iterations, on five publications from the Dr. Inventor Corpus. After all annotators labeled one of the five documents, we met with them, discussed the disagreements, identified erroneous annotations, and, when required, revised the annotation guidelines. At the end of the calibration phase, the annotators re-annotated the five calibration publications and resolved the remaining disagreements by consensus.

In Figure 1 we show the IAA for both component identification and relation classification, in terms of averaged pairwise F_1 score,[4] after each of the five calibration iterations. It can be seen that the discussions in the calibration phase helped to get a common understanding of the task among the annotators. However, we note that when considering argumentative relations in addition to the components only, the agreement decreases. Apart from the increased complexity compared to the component identification only this is due to the high ambiguity of argumentative structures, which is one of the main challenges in argument mining,

---

[1] We omitted some of Toulmin's component types (e.g., *Backing*) due to very rare occurrence in the corpus.

[2] `http://data.dws.informatik.uni-mannheim.de/sci-arg/annotation_guidelines.pdf`

[3] A researcher in computer science, albeit not in computer graphics, which is the domain of the corpus.

[4] We measured the agreement in terms of the F_1 measure because (1) it is straight-forward to compute, (2) it is directly interpretable, and (3) it can account for spans of varying length, allowing for computing relaxed agreements in terms of partial overlaps, and (4) the chance-corrected measures, e.g., Cohen's Kappa, approach F_1-measure when the number of negative instances grows (Hripcsak and Rothschild, 2005).

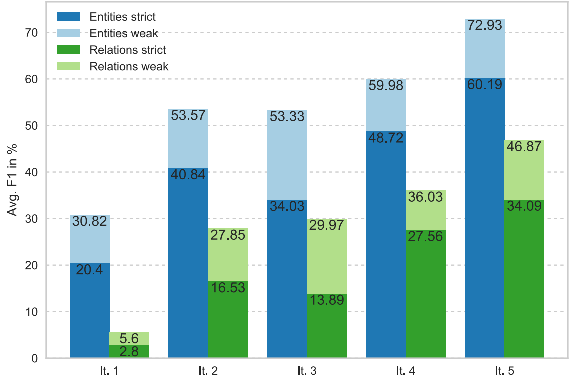

Figure 1: IAA evolution over calibration phases (*blue* for argumentative components; *green* for relations). We report both *strict* (annotated components match in span and type; relations match in type and components at both ends match strictly) and *relaxed* agreement scores (components match in type and overlap in span; relations match in type and their components at both ends match according to the relaxed criterion).

as suggested by Stab et al. (2014). Moreover, disagreements in the argumentative component identification are propagated and cause disagreements in relation annotations, since relation annotations match only when the agreement criterion for the components at both ends is met. Interestingly, the average agreement of our *expert* annotator with *non-expert* annotators was similar to the average agreement between non-expert annotators. This is encouraging, because it suggests that annotating argumentative structures in scientific text does not require expert knowledge of the domain. In the second phase, we evenly split the remaining 35 documents of the Dr. Inventor Corpus among the four annotators, without any overlaps.

The augmented corpus. We make the Dr. Inventor Corpus augmented with argumentation annotations (together with the annotation guidelines) publicly available.[5] The final corpus contains $12,289$ annotations of argumentative components and $6,530$ relation annotations. We show the distributions of labels in Table 2.

The number of *own claims* doubles the number of *background claims*. This is not surprising considering that the Dr. Inventor Corpus contains only original research articles (i.e., no survey nor position articles), in which authors primarily emphasize the contributions of their own work. There are two main reasons for having a smaller number of *data* components compared to *claims*. On one hand, there are longer argumentative chains in which claims are supported by other claims (i.e., only the first claim is supported by the data component). On the other hand, there is also a non-negligible amount of standalone (i.e., unsupported and unchallenged) claims, implied also by having less annotated relations than claims.

Category	Label	Occurrences	%
Component	*Background claim*	2,751	22.4
	Own claim	5,445	44.3
	Data	4,093	33.3
Relation	*Supports*	5,790	88.7
	Contradicts	696	10.7
	Semantically same	44	0.7

Table 2: Distributions of labels of argumentative components and relations in the corpus.

	AC	DR	SA	SR
AC	–	–	–	–
DR	**0.22**	–	–	–
SA	0.08	0.11	–	–
SR	0.04	0.10	0.13	–
CC	**0.18**	0.10	0.04	0.01

Table 3: Normalized mutual information between the label sets of the annotation layers indicating argument components (AC), discourse roles (DR), subjective aspects (SA), and citation contexts (CC) in the extended Dr. Inventor Corpus.

To obtain an initial insight on the interrelations between the different rhetorical aspects in scientific writing, we conduct an information-theoretic analysis and assess the amount of information shared among the annotation layers by computing the normalized mutual information (Strehl and Ghosh, 2003). Normalized mutual information is a variant of mutual information, which has been shown to correlate with the gains that can be obtained in multi-task learning settings (Bjerva, 2017). The results can be seen in Table 3. The strongest link is observed between argument components and discourse roles, followed by argument components and citation contexts.

---

[5] `http://data.dws.informatik.uni-mannheim.de/sci-arg/compiled_corpus.zip`

4 Multi-task Learning for Rhetorical Analysis of Scientific Writing

We next exploit the augmented corpus to exploit the dependencies between argumentation and other rhetorical dimensions. To this end, we adopt neural MTL as a methodological framework.

4.1 Tasks

The following are the rhetorical analysis and argument extraction tasks we investigate.

Argumentative Component Identification (ACI). The task is to extract and classify argumentative components. We frame ACI as a token-level sequence labeling task: given a sequence of tokens $\mathbf{x} = (x_1, .., x_n)$ of length n, the task is to assign a sequence of tags $\mathbf{y}_{aci} = (y_1, .., y_n), y_i \in Y_{aci}$. The tagset Y_{aci} contains seven token-level tags, obtained by combining the standard B-I-O annotation scheme with three types of argumentative components: *Own claim*, *Background claim*, and *Data*.

Discourse Role Classification (DRC). The multi-class classification task in which each sentence needs to be assigned one out of the set of discourse roles $Y_{drc} = \{$*Background*, *Unspecified*, *Challenge*, *FutureWork*, *Approach*, *Outcome*$\}$.

Citation Context Identification (CCI). The task is to identify the span of the publication text that introduces or explains a reference. It is also a token-level sequence-labeling task – a sequence of tags $\mathbf{y}_{cci} = (y_1, .., y_n)$ with $y_i \in Y_{cci} = \{B_{CC}, I_{CC}, O\}$ is assigned to a sequence of tokens $\mathbf{x} = (x_1, .., x_n)$.

Subjective Aspect Classification (SAC). Another sentence-level classification task in which each sentence is assigned one of the subjective aspect labels, $Y_{sac} = \{$*None*, *Limitation*, *Advantage*, *Disadvantage-Advantage*, *Disadvantage*, *Common Practice*, *Novelty*, *Advantage-Disadvantage*$\}$.

Summary Relevance Classification (SRC). The task is to predict the relevance of a sentence for the (extractive) summary of the publication. Each sentence needs to be assigned one of the labels $Y_{src} = \{$*Very relevant*, *Relevant*, *May appear*, *Should not appear*, *Totally irrelevant*$\}$.

ACI and CCI are token-level sequence labeling tasks. The remaining three tasks can be cast as either (1) plain sentence classification tasks or (2) sentence-level sequence labeling tasks (assuming that there are regularities in sequences of sentence-level labels that can be captured). We propose one MTL architecture for each of the two possibilities.

4.2 Multi-Task Learning Models

We propose two different MTL architectures for the rhetorical and argumentative analysis of scientific publications. The *Simple model* treats sentence-level tasks (DRC, SAC, and SRC) as plain classification tasks (i.e., the prediction for each sentence ignores the content and labels of other, neighboring sentences). The *Hierarchical model* addresses sentence-level tasks as sequence labeling tasks. This model can be seen as a hierarchical sequence labeling model, in which the sentence-level recurrent network is stacked on top of the token-level sequence labeling network. Both architectures are illustrated in Figure 2.

Token-level Predictions. Given a sentence $s_i = (x_{i1}, .., x_{in})$ out of a sequence of sentences $d = (s_1, .., s_m)$ we first retrieve the pre-trained embedding vector for each token x_{ij}.We then obtain context-aware token representations h_{ij} by applying a bidirectional recurrent network with long short-term memory cells (Hochreiter and Schmidhuber, 1997) on the sequence of pre-trained word embeddings:

$$h_{ij} = [\overrightarrow{LSTM}(x_{i1}, \ldots, x_{ij}); \overleftarrow{LSTM}(x_{in}, \ldots, x_{ij})] \,. \quad (1)$$

This token-level Bi-LSTM encoder is shared between the tasks combined by the MTL models. Next, we define a separate classifier for each of the token-level (TL) tasks (i.e., ACI and CCI) and feed the contextualized token representations h_{ij} to these classifiers. Each of the classifiers is defined as a feed-forward network with a single hidden layer. The label probability distribution is obtained by applying the *softmax* function on its output.

$$y_{ijt} = \text{softmax}(W_t h_{ij} + b_t) \,, \quad (2)$$

where $W_t \in \mathbb{R}^{2K \times |Y_t|}$ and $b_t \in \mathbb{R}^{|Y_t|}$ are the task-specific classification parameters for the task t, with K being the size of the LSTM state and $|Y_t|$ the number of discrete labels of task t.

Sentence-level Predictions. We learn to aggregate a sentence representation s_i from contextualized vectors of its tokens, h_{ij} (produced by the

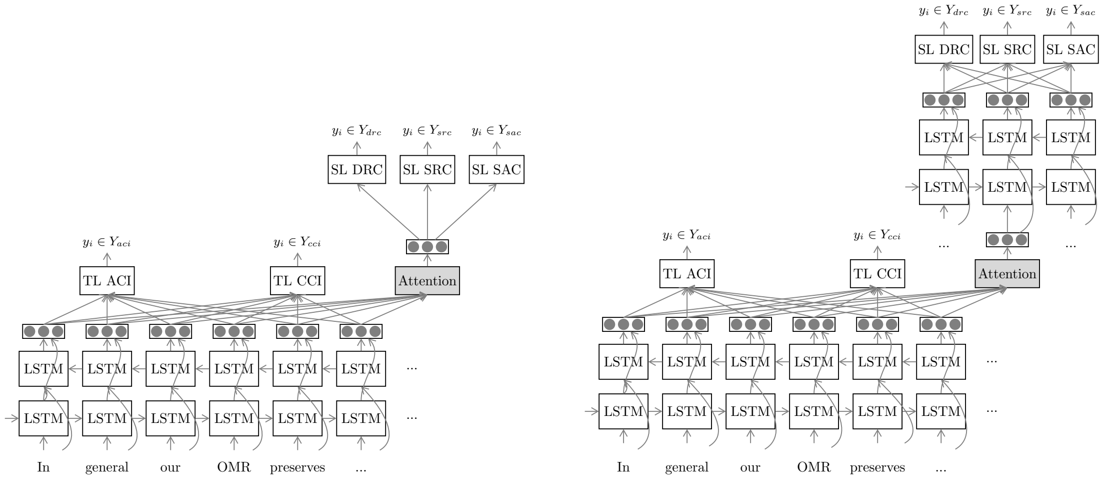

(a) Simple model.

(b) Hierarchical model.

Figure 2: Neural MTL architectures for rhetorical and argumentative analysis of scientific publications: (a) the *Simple model* addresses sentence-level tasks (DRC, SAC, SRC) as plain classification tasks, whereas (b) the *Hierarchical model* treats sentence-level tasks as sequence labeling tasks. Both models address ACI and CCI as token-level sequence labeling tasks.

token-level Bi-LSTM), using the intra-sentence attention mechanism (Yang et al., 2016):

$$s_i = \sum_j \alpha_{ij} h_{ij} \,, \tag{3}$$

with the weights α_i computed dynamically as:

$$\alpha_i = \text{softmax}(U_i \, u_{att}) \,, \tag{4}$$

where u_{att} is the trainable attention head vector and U_i is a matrix with non-linearly transformed token representations (h_{ij}) as rows:

$$U_{ij} = \tanh(W_{att} h_{ij} + b_{att}) \,. \tag{5}$$

In the *Simple* architecture, sentence representations s_i are fed directly to the sentence-level task-specific classifiers, which are also feed-forward networks with a single hidden layer:

$$y_{it} = \text{softmax}(W_t s_i + b_t) \,. \tag{6}$$

Within the *Hierarchical* architecture, sentence representations are first contextualized with representations of other sentences via the sentence-level Bi-LSTM layer (denoted with the function Bi-LSTM$_S$) and then forwarded to the classifier:

$$y_{it} = \text{softmax}(W_t \text{Bi-LSTM}_S(s_i) + b_t) \,. \tag{7}$$

Joint optimization and loss functions. All of the tasks we consider are framed as multi-class classification tasks. Thus, we simply specify all task-specific losses to be L2-regularized cross-entropy errors. Let y_{to} be the one-hot ground truth label vector for the prediction instance o[6] of the task t, and let y'_{to} be the predicted probability distribution over the task labels for the same instance. With Y_t as the set of labels for task t, the task-specific loss L_t is computed as follows:

$$L_t = \lambda \|\Omega_t\|_2 - \sum_o \sum_{k=1}^{|Y_t|} y_{to}^{(k)} \cdot \ln\left(y_{to}^{\prime(k)}\right) , \tag{8}$$

where Ω_t is the set of model's parameters relevant for the task t[7] and λ is the regularization factor. We train the MTL model jointly on different tasks by defining and minimizing the joint loss function L that combines task-specific losses L_t. Instead of using constant weights, we opt for dynamic weighting of task-specific losses during the training process, based on the homoscedastic uncertainty of tasks, as proposed by Kendall et al. (2018):

$$L = \sum_t \frac{1}{2\sigma_t^2} L_t + \ln \sigma_t^2 \,, \tag{9}$$

where σ_t is the variance of the task-specific loss over training instances, used to quantify the uncertainty of task t. Kendall et al. (2018) show that better MTL results can be obtained by dynamically

---

[6] The prediction instance is a token for ACI and CCI, and a sentence for DRC, SAC, and SRC.

[7] The set of relevant parameters differs across tasks: for token-level tasks (e.g., ARI) Ω_t denotes token-level Bi-LSTM parameters and the parameters W_t and b_t of task t's classifier; for a sentence-level task (e.g., DRC) within the *Hierarchical* architecture, Ω_t includes all parameters of both token- and sentence-level Bi-LSTMs, intra-sentence attention parameters, and parameters of the task-specific classifier.

assigning less weight to the more uncertain tasks, as opposed to constant task weights throughout the whole training process.

5 Evaluation

We run two sets of experiments. First, we evaluate the performance of the *Simple* and the *Hierarchical* neural models on individual tasks (i.e., in single-task learning (STL) scenarios). We then evaluate the impact of the argumentative signal on other dimensions of rhetorical analysis by combining them in joint MTL settings.

5.1 Experimental Setup.

We randomly split the corpus on the document-level into train (roughly 70%, 28 documents containing 6,697 sentences) and test portions (roughly 30%; 12 documents with 2,874 sentences). We used roughly 20% of the train portion as the validation set for model selection.

Model configuration and training. We ran an initial grid search on the validation set with possible values for the hyperparameters learning rate $\nu \in \{10^{-4}, 10^{-5}\}$, L2 regularization factor $\lambda \in \{0.001, 0, 0001\}$, and LSTM states $K \in \{64, 128, 256\}$ and found the hyperparameter configuration $\nu = 10^{-4}$, $\lambda = 0.001$, and $K = 128$ to be optimal for the vast majority of the STL and MTL models. In all experiments, we represent tokens with pre-trained 300-dimensional GloVe embeddings (Pennington et al., 2014)[8] and optimize the model parameters using the Adam algorithm (Kingma and Ba, 2015). We initialize all model parameters using Xavier initialization (Glorot and Bengio, 2010), train the models in batches of $N = 16$ sentences and apply early stopping based on the validation set performance.

Baselines. As a type of "sanity check", we first compare the performance of the two neural architectures against traditional supervised machine learning algorithms on each of the tasks separately. For the token-level sequence labeling tasks (ACI and CCI) we use Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (Lafferty et al., 2001) as baselines. The HMM works directly on the tokens, while we feed either the lexical representation or the embedding representation of the tokens as features for the CRF. For the sentence classification tasks (DRC, SAC, and SRC), we evaluate as baselines (1) the linear Support Vector Machines (SVM) with TF-IDF feature vectors and (2) SVM with RBF kernel and embedding features. In the latter case we obtain a sentence representation by averaging the pre-trained embeddings of sentence words. We tune the hyperparameter values of the SVM by conducting a grid search with possible penality parameter values $c \in \{0.1, 1.0, 10.0\}$ (linear SVM and SVM with RBF kernel) and the parameter of the radial basis function $\gamma \in \{0.01, 0.1, 1.0\}$ (SVM with RBF kernel). The possible hyperparameter values for the L1 regularization coefficient $c1$ and for L2 regularization coefficient $c2$ of the CRF are $c1, c2 \in \{0.1, 0.2, 0.001, 0.0001\}$.

In MTL experiments, we consider the respective task performances from single-task experiments and MTL with a joint loss function with equal weighting of the task losses as baselines.

Model	ACI			CCI		
	P	R	F	P	R	F
HMM	30.8	17.2	20.8	18.3	13.1	15.0
CRF$_{lexical}$	38.8	29.1	31.7	15.3	17.8	16.4
CRF$_{embeddings}$	37.9	23.3	26.1	12.8	1.4	2.5
Neural: *Simple*	**47.0**	**44.5**	**44.7**	**48.7**	**43.8**	**46.1**

Table 4: Single-task results for token-level tasks (macro-averaged F_1 performances).

Model	DRC	SAC	SRC
SVM$_{tfidf}$	34.0	10.3	22.2
SVM$_{embeddings}$	25.7	08.5	19.3
Neural: *Simple*	**44.1**	**20.5**	31.5
Neural: *Hierarchical*	42.6	19.1	**33.2**

Table 5: Single-task results for sentence-level tasks (macro-averaged F_1 scores).

Single-Task Experiments. We first report the model performances for individual tasks in STL settings. Results for token-level tasks are shown in Table 4, whereas Table 5 displays results for sentence-level tasks. The scores (precision, recall, and F_1 score) are reported as macro-averages over all task labels. Expectedly, our neural architectures substantially outperform the traditional machine learning baselines on all tasks. For the three sentence-level tasks, the *Hierarchical* architecture outperforms the *Simple* model only when classifying sentences by summary relevance (SRC). This result seems intuitive – a *Very relevant* sentence is likely to be surrounded with *Relevant* and *May*

---

[8] `http://nlp.stanford.edu/data/glove.840B.300d.zip`.

	CCI	DRC	SAC	SRC
Single Task				
Simp	46.1	44.1	20.5	31.5
Hier	–	42.6	19.1	33.2
Multi Task (w. ACI)				
$Simp_{0.5}$	43.8 (44.2)	43.5 (41.6)	18.0 (42.0)	32.2 (41.9)
$Simp_{\text{uncert}}$	**49.9** (40.5)	**45.2** (38.6)	**22.1** (39.4)	**34.8** (41.0)
$Hier_{0.5}$	–	41.6 (42.1)	17.8 (42.9)	30.3 (43.4)
$Hier_{\text{uncert}}$	–	**43.9** (40.8)	18.9 (41.6)	**34.8** (40.8)

Table 6: MTL results: rhetorical analysis tasks coupled with argumentative component identification. We report the F1 score macro-averaged over the classes. The scores achieved for ACI are shown in parentheses.[9]

appear sentences (and an *Irrelevant* sentence with other *Irrelevant* and *Should not appear* sentences). The fact that we observe no gains from the additional sentence-level Bi-LSTM encoder for the DRC and SAC tasks suggests that the content of the sentence informs its discourse role and subjective aspect much more strongly than neighboring sentences. In other words, the DRC and SAC seem to be more localized classification tasks than SRC.

Multi-Task Learning Results. Our core research question relates to the effect that recognizing fine-grained argumentative components has on other rhetorical analysis tasks. This is why, in our central set of experiments, we evaluate MTL models with homoscedastic uncertainty weighting which combine the ACI (as an auxiliary task) with each of the four other tasks. In each multi-task learning model, the token-level Bi-LSTM encoder is shared between the two tasks. For sentence-level tasks (DRC, SAC, SRC), we evaluate both the *Simple* and *Hierarchical* architecture. In Table 6 we show the performance of the MTL models on rhetorical analysis tasks (these can be compared to the respective single-task model performances from Tables 4 and 5.

When coupled in MTL settings with argumentation component identification (ACI) using the joint loss formulation of Kendall et al. (2018), the results significantly[10] improve for all rhetorical analysis tasks and models (except for SAC with the *Hierarchical* model), in comparison with the respective single-task models. However, the performance for the argumentation component identification does not improve in MTL. In other words, the extraction of fine-grained argumentative components seems to inform higher-level rhetorical analysis tasks, but not vice-versa. This indeed supports the hypothesis that argumentation guides scientific writing and influences rhetorical structure of publications. Furthermore, our results support the findings of Schulz et al. (2018) who show that, opposed to initial results of Alonso and Plank (2017), MTL can yield performance gains for higher-level semantic tasks.

6 Conclusion

Acknowledging the argumentative nature of scientific text, we investigated the role of argumentation in the rhetorical analysis of scientific publications. We first extended an existing corpus annotated with four different layers of rhetorical information with annotations of argumentative components and relations, creating the largest argumentation-labeled corpus of scientific text in English. We explored intuitive neural architectures with recurrent encoders for argument extraction and rhetorical analysis tasks and showed significant improvements over traditional machine learning models. We then coupled argument extraction with different rhetorical analysis tasks in MTL models with dynamic loss weighting and demonstrated that the argumentative signal has a positive impact on high-level rhetorical analysis tasks.

Admittedly, the corpus we used in this work is limited to the domain of computer graphics. Nonetheless, we believe that our findings relating to the argumentative nature of scientific text and links between argumentation and other rhetorical aspects generalize to other domains too. This is also supported by the comparable agreement observed between expert and non-expert annotators.

In the future work, we would like to extend the collection of scientific text to other fields. Next, we intend to explore a wider range of MTL models, especially those involving more than two tasks. Having annotated argumentative relations, we will work on models for their automated identification in scientific publications.

Acknowledgments

This research was partly funded by the German Research Foundation (DFG), grant number EC 477/5-1 (LOC-DB). We thank our four annotators for their dedicated annotation effort and the anonymous reviewers for constructive and insightful comments.

---

[9] In the multi-task settings, the early stopping criterion was based on the auxiliary task score.

[10] Differences significant at $p < 0.05$, tested using the non-parametric stratified shuffling test (Yeh, 2000).

References

Amjad Abu-Jbara, Jefferson Ezra, and Dragomir Radev. 2013. Purpose and polarity of citation: Towards nlp-based bibliometrics. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 596–606, Atlanta, Georgia. Association for Computational Linguistics.

Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based summarization of scientific papers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Volume 1*, pages 500–509, Portland, OR, USA. Association for Computational Linguistics.

Pablo Accuosto, Francesco Ronzano, Daniel Ferrés, and Horacio Saggion. 2017. Multi-level mining and visualization of scientific text collections: Exploring a bi-lingual scientific repository. In *Proceedings of the 6th International Workshop on Mining Scientific Publications*, pages 9–16, Toronto, ON, Canada. Association for Computing Machinery.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? Semantic sequence prediction under varying data conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 44–53, Valencia, Spain. Association for Computational Linguistics.

Jean-Claude Anscombre and Oswald Ducrot. 1983. *L'argumentation Dans La Langue*. Editions Mardaga.

Awais Athar. 2011. Sentiment analysis of citations using sentence structure-based features. In *Proceedings of the ACL 2011 Student Session*, pages 81–87, Portland, OR, USA. Association for Computational Linguistics.

Trevor JM Bench-Capon. 1998. Specification and implementation of toulmin dialogue game. In *Proceedings of the 11th Conference on Legal Knowledge Based Systems*, pages 5–20, Groningen, Netherlands. Foundation for Legal Knowledge Based Systems.

Johannes Bjerva. 2017. Will my auxiliary tagging task help? estimating auxiliary tasks effectivity in multi-task learning. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 216–220, Gothenburg, Sweden. Association for Computational Linguistics.

Filip Boltužić and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the 1st Workshop on Argumentation Mining*, pages 49–58, Baltimore, MD, USA. Association for Computational Linguistics.

Lutz Bornmann and Rüdiger Mutz. 2015. Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references. *Journal of the Association for Information Science and Technology*, 66(11):2215–2222.

Jingqiang Chen and Hai Zhuge. 2014. Summarization of scientific documents by detecting common facts in citations. *Future Generation Computer Systems*, 32:246–252.

Arman Cohan and Nazli Goharian. 2015. Scientific article summarization using citation-context and article's discourse structure. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 390–400, Lisbon, Portugal. Association for Computational Linguistics.

Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab, and Iryna Gurevych. 2017. What is the essence of a claim? Cross-domain claim identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2066, Copenhagen, Denmark. Association for Computational Linguistics.

Phan Minh Dung. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 11–22, Vancouver, Canada. Association for Computational Linguistics.

Beatriz Fisas, Francesco Ronzano, and Horacio Saggion. 2016. A multi-layered annotated corpus of scientific papers. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 3081–3088, Portorož, Slovenia. European Language Resources Association.

Beatriz Fisas, Horacio Saggion, and Francesco Ronzano. 2015. On the discoursive structure of computer graphics research papers. In *Proceedings of The 9th Linguistic Annotation Workshop*, pages 42–51, Denver, CO, USA. Association for Computational Linguistics.

James B. Freeman. 1991. Dialectics and the macrostructure of arguments: A theory of argument structure.

G Nigel Gilbert. 1976. The transformation of research findings into scientific knowledge. *Social Studies of Science*, 6(3-4):281–306.

G Nigel Gilbert. 1977. Referencing as persuasion. *Social Studies of Science*, 7(1):113–122.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256, Sardinia, Italy. Proceedings of Machine Learning Research.

Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? Analyzing and predicting convincingness of web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 1122, Berlin, Germany. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

George Hripcsak and Adam S. Rothschild. 2005. Agreement, the F-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298.

Kokil Jaidka, Muthu Kuma Chandrasekaran, Devanshu Jain, and Min-Yen Kan. 2017. Overview of the CL-SciSumm 2017 Shared Task. In *Proceedings of the Computational Linguistics Scientific Summarization Shared Task*, pages 1–15, Tokyo, Japan. CEUR-WS.

Rahul Jha, Amjad-Abu Jbara, Vahed Qazvinian, and Dragomir R. Radev. 2017. NLP-driven citation analysis for scientometrics. *Natural Language Engineering*, 23(1):93–130.

Charles Jochim and Hinrich Schütze. 2012. Towards a generic and flexible citation classifier based on a faceted classification scheme. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 1343–1358, Mumbai, India. The COLING 2012 Organizing Committee.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA. Institute of Electrical and Electronics Engineers.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, CA, USA.

Christian Kirschner, Judith Eckle-Kohler, and Iryna Gurevych. 2015. Linking the thoughts: Analysis of argumentation structures in scientific publications. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 1–11, Denver, CO, USA. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, Williamstown, MA, USA. Morgan Kaufmann Publishers Inc.

Anne Lauscher, Goran Glavaš, and Kai Eckert. 2017a. Citation-based summarization of scientific articles using semantic textual similarity. In *Proceedings of the Computational Linguistics Scientific Summarization Shared Task*, pages 33–42, Tokyo, Japan. CEUR-WS.

Anne Lauscher, Goran Glavaš, Simone Paolo Ponzetto, and Kai Eckert. 2017b. Investigating convolutional networks and domain-specific embeddings for semantic classification of citations. In *Proceedings of the 6th International Workshop on Mining Scientific Publications*, pages 24–28, Toronto, ON, Canada. Association for Computing Machinery.

Maria Liakata, Shyamasree Saha, Simon Dobnik, Colin Batchelor, and Dietrich Rebholz-Schuhmann. 2012. Automatic recognition of conceptualization zones in scientific articles and two life science applications. *Bioinformatics*, 28(7):991–1000.

Maria Liakata, Simone Teufel, Advaith Siddharthan, and Colin R Batchelor. 2010. Corpora for the conceptualisation and zoning of scientific papers. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, pages 2054–2061, Valetta, Malta. European Language Resources Association.

Marco Lippi and Paolo Torroni. 2015. Argument mining: A machine learning perspective. In *Theory and Applications of Formal Argumentation: Third International Workshop, TAFA 2015, Buenos Aires, Argentina, July 25-26, 2015, Revised Selected Papers*, Lecture Notes in Artifical Intelligence, pages 163–176. Springer International Publishing.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology*, 16(2):10:1–10:25.

Kathy McKeown, Hal Daume, Snigdha Chaturvedi, John Paparrizos, Kapil Thadani, Pablo Barrio, Or Biran, Suvarna Bothe, Michael Collins, Kenneth R. Fleischmann, Luis Gravano, Rahul Jha, Ben King, Kevin McInerney, Taesun Moon, Arvind Neelakantan, Diarmuid O'Seaghdha, Dragomir Radev, Clay Templeton, and Simone Teufel. 2016. Predicting the impact of scientific concepts using full-text features. *Journal of the Association for Information Science and Technology*, 67(11):2684–2696.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument mining with structured SVMs and RNNs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 985–995, Vancouver, Canada. Association for Computational Linguistics.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107, Barcelona, Spain. Association for Computing Machinery.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948, Lisbon, Portugal. Association for Computational Linguistics.

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

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1384–1394, San Diego, CA, USA. Association for Computational Linguistics.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Here's my point: Joint pointer architecture for argument mining. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1364–1373, Copenhagen, Denmark. Association for Computational Linguistics.

Francesco Ronzano and Horacio Saggion. 2015. Dr. Inventor Framework: Extracting structured information from scientific publications. In *Discovery Science, 18th International Conference, DS 2015, Banff, AB, Canada, October 4-6, 2015. Proceedings*, Lecture Notes in Computer Science, pages 209–220, Banff, Canada. Springer, Cham.

Francesco Ronzano and Horacio Saggion. 2016. Knowledge extraction and modeling from scientific publications. In *Semantics, Analytics, Visualization. Enhancing Scholarly Data, Second International Workshop, SAVE-SD 2016, Montreal, QC, Canada, April 11, 2016, Revised Selected Papers*, Lecture Notes in Computer Science, pages 11–25, Montreal, Canada. Springer, Cham.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multi-task learning for argumentation mining in low-resource settings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Volume 2, Short Papers*, pages 35–41, New Orleans, Louisiana. Association for Computational Linguistics.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 231–235, Berlin, Germany. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Christian Stab, Christian Kirschner, Judith Eckle-Kohler, and Iryna Gurevych. 2014. Argumentation mining in persuasive essays and scientific articles from the discourse structure perspective. In *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing*, pages 21–25. CEUR-WS.

Alexander Strehl and Joydeep Ghosh. 2003. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617.

Simone Teufel, Jean Carletta, and Marc Moens. 1999. An annotation scheme for discourse-level argumentation in research articles. In *Proceedings of the 9th Conference on European Chapter of the Association for Computational Linguistics*, pages 110–117, Bergen, Norway. Association for Computational Linguistics.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445.

Simone Teufel, Advaith Siddharthan, and Colin Batchelor. 2009. Towards discipline-independent argumentative zoning: Evidence from chemistry and computational linguistics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3*, pages 1493–1502, Singapore. Association for Computational Linguistics.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 103–110, Sydney, Australia. Association for Computational Linguistics.

Stephen E. Toulmin. 2003. *The Uses of Argument*, updated edition. Cambridge University Press.

Bart Verheij. 2009. The toulmin argument model in artificial intelligence. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 219–238. Springer US, Boston, MA.

Douglas Walton, Chris Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, CA, USA. Association for Computational Linguistics.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 947–953, Saarbrücken, Germany. Association for Computational Linguistics.

Neural Ranking Models for Temporal Dependency Structure Parsing

Yuchen Zhang
Brandeis University
`yuchenz@brandeis.edu`

Nianwen Xue
Brandeis University
`xuen@brandeis.edu`

Abstract

We design and build the first neural temporal dependency parser. It utilizes a neural ranking model with minimal feature engineering, and parses time expressions and events in a text into a temporal dependency tree structure. We evaluate our parser on two domains: news reports and narrative stories. In a parsing-only evaluation setup where gold time expressions and events are provided, our parser reaches 0.81 and 0.70 f-score on unlabeled and labeled parsing respectively, a result that is very competitive against alternative approaches. In an end-to-end evaluation setup where time expressions and events are automatically recognized, our parser beats two strong baselines on both data domains. Our experimental results and discussions shed light on the nature of temporal dependency structures in different domains and provide insights that we believe will be valuable to future research in this area.

1 Introduction

Temporal relation classification is important for a range of NLP applications that include but are not limited to story timeline construction, question answering, summarization, etc. Most work on temporal information extraction models the task as a pair-wise classification problem (Bethard et al., 2007; Chambers et al., 2007; Chambers and Jurafsky, 2008; Ning et al., 2018a): given an individual pair of time expressions and/or events, the system predicts whether they are temporally related and which specific relation holds between them. An alternative approach is to model the temporal relations in a text as a temporal dependency structure (TDS) for the entire text (Kolomiyets et al., 2012). Such a temporal dependency structure has the advantage that (1) it can be easily used to infer additional temporal relations between time expressions and/or events that are not directly connected via the transitivity properties of temporal relations, (2) it is computationally more efficient because a model does not need to consider all pairs of time expressions and events in a text, and (3) it is easier to use for downstream applications such as timeline construction.

However, most existing automatic systems are pair-wise models trained with traditional statistical classifiers using a large number of manually crafted features (Bethard et al., 2017). The few exceptions include the work of Kolomiyets et al. (2012), which describes a temporal dependency parser based on traditional feature-based classifiers, and Dligach et al. (2017), which describes a system using neural network based models to classify individual temporal relations. More recently, a semi-structured approach has also been proposed (Ning et al., 2018b).

In this work, taking advantage of a newly available data set annotated with temporal dependency structures – the Temporal Dependency Tree (TDT) Corpus[1] (Zhang and Xue, 2018), we develop a neural temporal dependency structure parser using minimal hand-crafted linguistic features. One of the advantages of neural network based models is that they are easily adaptable to new domains, and we demonstrate this advantage by evaluating our temporal dependency parser on data from two domains: news reports and narrative stories. Our results show that our model beats a strong logistic regression baseline. Direct comparison with existing models is impossible because the only similar dataset used in previous work that we are aware of is not available to us (Kolomiyets et al., 2012), but we show that our models are competitive against similar systems reported in the literature.

The main contributions of this work are:

- We design and build the first end-to-end neu-

---

[1] `https://github.com/yuchenz/structured_temporal_relations_corpus`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3339–3349
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

ral temporal dependency parser. The parser is based on a novel neural ranking model that takes a raw text as input, extracts events and time expressions, and arranges them in a temporal dependency structure.

- We evaluate the parser by performing experiments on data from two domains: news reports and narrative stories, and show that our parser is competitive against similar parsers. We also show the two domains have very different temporal structural patterns, an observation that we believe will be very valuable to future temporal parser development.

The rest of the paper is organized as follows. Since temporal structure parsing is a relatively new task, we give a brief problem description in §2. We describe our end-to-end pipeline system in §3. Details of the neural ranking model are discussed in §4. In §5 we present and discuss our experimental results, and error analysis are presented in §6. In §7 we discuss related work and situate our work in the broader context, and we conclude our paper in §8.

2 Problem Description

In this section we give a brief description of the temporal dependency parsing task (more details in Zhang and Xue (2018)). In a temporal structure parsing task, a text is parsed into a dependency tree structure that represents the inherent temporal relations among time expressions and events in the text. The nodes in this tree are mostly time expressions and events which are represented as contiguous spans of words in the text. They can also be pre-defined meta nodes, which serve as reference times for other time expressions and events, and they constitute the top-most part of the tree. For example, *Past_Ref*, *Present_Ref*, *Future_Ref*, and *Document Creation Time (DCT)* are all pre-defined meta nodes. The edges in the tree represent temporal relations between each parent-child pair. The temporal relations can be one of *Includes*, *Before*, *Overlap*, and *After*, or *Depend-on* which holds between two time expressions. Unlike syntactic dependency parsing where each word in a sentence is a node in the dependency structure, in a temporal dependency structure only some of the words in a text are nodes in the structure. Therefore, this process naturally falls into two stages: first time expression and event recognition, and then temporal relation parsing. Figure 1 is an example temporal dependency tree for a news report paragraph.

Due to the fact that different types of time expressions and events behave differently in terms of what can be their antecedents, and recognition of these types can be helpful for determining temporal relations, finer classifications of time expressions and events are also defined. Time expressions are further classified into *Vague Time*, *Absolute Concrete Time*, and *Relative Concrete Time*, according to whether or not the time expression can be located on the timeline, and whether or not the interpretation of its temporal location depends on another time expression. Events are further classified into *Eventive Event*, *State*, *Habitual Event*, *Completed Event*, *Ongoing Event*, *Modalized Event*, *Generic Habitual*, and *Generic State*, according to the eventuality type of the event. Our experiments will show that these fine-grained classifications are very helpful for the overall temporal structure parsing accuracy.

3 A Pipeline System

We build a two-stage pipeline system to tackle this temporal structure parsing problem. The first stage performs event and time expression identification. In this stage, given a text as input, spans of words that indicate events or time expressions are identified and categorized. We model this stage as a sequence labeling process. A standard Bi-LSTM sequence model coupled with BIO labels is applied here. Word representations are the concatenation of word and POS tag embeddings.

The second stage performs the actual temporal structure parsing by identifying the antecedent for each time expression and event, and identifying the temporal relation between them. In this stage, given events and time expressions identified in the first stage as input, the model outputs a temporal dependency tree in which each child node is an event or time expression that is temporally related to another event or time expression or pre-defined meta node as its parent node. This stage is modeled as a ranking process: for each node, a finite set of neighboring nodes are first selected as its candidate parents. These candidates are then ranked with a neural network model and the highest ranking candidate is selected as its parent. We use a ranking model because it is simple, more intuitive and easier to train than a traditional transition-based or graph-based model, and

Example News Paragraph[a]**:**
Jorn Utzon, the Danish architect who designed the Sydney Opera House, has died$_{e1}$ in Copenhagen. Born$_{e2}$ in 1918$_{t1}$, Mr Utzon was inspired$_{e3}$ by Scandinavian functionalism in architecture, but made a number of inspirational trips$_{e4}$, including to Mexico and Morocco. In 1957$_{t2}$, Mr Utzon's now-iconic shell-like design for the Opera House unexpectedly won$_{e5}$ a state government competition for the site on Bennelong Point on Sydney Harbour. However, he left$_{e6}$ the project in 1966$_{t3}$. His plans for the interior of the building were not completed$_{s1}$. The Sydney Opera House is$_{s2}$ one of the world's most classic modern buildings and a landmark Australian structure. It was declared$_{e7}$ a UNESCO World Heritage site last year$_{t4}$.

[a] From a news report in *The Telegraph*.

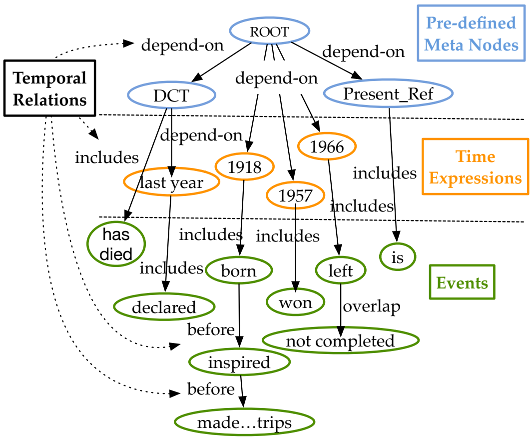

Figure 1: Example text and its temporal dependency tree. DCT is Document Creation Time.

the learned model rarely makes mistakes that violate the structural constraint of a tree.

Since the model we use for Stage 1 is a very standard model with little modifications, we don't describe it in detail in this paper due to the limitation of space. Our neural ranking model for Stage 2 is described in detail in the next section.

4 Neural Ranking Model

4.1 Model Description

We use a neural ranking model for the parsing stage. For each time expression or event node i in a text, a group of candidate parent nodes (time expressions, events, or pre-defined meta nodes) are selected. In practice, we select a window from the beginning of the text to two sentences after node i, and select all nodes in this window and all pre-defined meta nodes as the candidate parents if node i is an event. Since the parent of a time expression can only be a pre-defined meta node or another time expression as described in Zhang and Xue (2018), we select all time expressions in the same window and all pre-defined meta nodes as the candidate parents if node i is a time expression. Let y'_i be a candidate parent of node i, a score is then computed for each pair of (i, y'_i).Through ranking, the candidate with the highest score is then selected as the final parent for node i.

Model architecture is shown in Figure 2. Word embeddings are used as word representations (e.g. $\boldsymbol{w}_k$). A Bi-LSTM sequence layer is built on each word over the entire text, computing Bi-LSTM output vectors for each word (e.g. $\boldsymbol{w}_k^*$). The node representation for each time expression or event is the summation of the Bi-LSTM output vectors of all words in the text span (e.g. $\boldsymbol{x}_i$). The pair representation for node i and one of its candidates y'_i is the concatenation of the Bi-LSTM output vectors of these two nodes $\boldsymbol{g}_{i,y'_i} = [\boldsymbol{x}_i, \boldsymbol{x}_{y'_i}]$, which is then sent through a Multi-Layer Perceptron to compute a score for this pair s_{i,y'_i}. Finally all pair scores of the current node i are concatenated into vector $\boldsymbol{c}_i$, and taking $softmax$ on it generates the final distribution $\boldsymbol{o}_i$, which is the probability distribution of each candidate being the parent of node i.

Formally, the **Forward Computation** is:

$$\begin{aligned}
\boldsymbol{w}_k^* &= BiLSTM(\boldsymbol{w}_k) \\
\boldsymbol{x}_i &= sum(\boldsymbol{w}_{k-1}^*, \boldsymbol{w}_k^*, \boldsymbol{w}_{k+1}^*) \\
\boldsymbol{g}_{i,y'_i} &= [\boldsymbol{x}_i, \boldsymbol{x}_{y'_i}] \\
\boldsymbol{h}_{i,y'_i} &= tanh(\mathbf{W_1} \cdot \boldsymbol{g}_{i,y'_i} + \mathbf{b_1}) \\
s_{i,y'_i} &= \mathbf{W_2} \cdot \boldsymbol{h}_{i,y'_i} + \mathbf{b_2} \\
\boldsymbol{c}_i &= [s_{i,1}, ..., s_{i,i-1}, s_{i,i+1}, ..., s_{i,i+t}] \\
\boldsymbol{o}_i &= softmax(\boldsymbol{c}_i)
\end{aligned}$$

4.2 Learning

Let D be the training data set of K texts, N_k the number of nodes in text D_k, and y_i the gold parent for node i. Our neural model is trained to maximize $P(y_1, ..., y_{N_k}|D_k)$ over the whole training set. More specifically, the cost function is defined as follows:

$$\begin{aligned}
C &= -log \prod_k^K P(y_1, ..., y_{N_k}|D_k) \\
&= -log \prod_k^K \prod_i^{N_k} P(y_i|D_k) \\
&= \sum_k^K \sum_i^{N_k} -logP(y_i|D_k)
\end{aligned}$$

For each training example, cross-entropy loss is minimized:

$$L = -logP(y_i|D_k) \\ = -log\frac{exp[s_{i,y_i}]}{\sum_{y_i'} exp[s_{i,y_i'}]}$$

where $s_{i,y_i'}$ is the score for child-candidate pair (i, y_i') as described in §4.1.

4.3 Decoding

During decoding, the parser constructs the temporal dependency tree incrementally by identifying the parent node for each event or time expression in textual order. To ensure the output parse is a valid dependency tree, two constraints are applied in the decoding process: (i) there can only be one parent for each node, and (ii) descendants of a node cannot be its parent to avoid cycles. Candidates violating these constraints are omitted from the ranking process.[2]

4.4 Temporal Relation Labeling

The neural model described above generates an unlabeled temporal dependency tree, with each parent being the most salient reference time for the child. However it doesn't model the specific temporal relation (e.g. "before", "overlap") between a parent and a child. We extend this basic architecture to both identify parent-child pairs and predict their temporal relations. In this new model, instead of ranking child-candidate pairs (i, y_i'), we rank child-candidate-relation tuples (i, y_i', l_k), where l_k is the kth relation in the pre-defined set of possible temporal relation labels L. We compute this ranking by re-defining the pair score $s_{i,y_i'}$. Here, pair score $s_{i,y_i'}$ is no longer a scalar score but a vector $\boldsymbol{s}_{i,y_i'}$ of size $|L|$, where $\boldsymbol{s}_{i,y_i'}[k]$ is the scalar score for y_i' being the parent of i with temporal relation l_k. Accordingly, the lengths of $\boldsymbol{c}_i$ and $\boldsymbol{o}_i$ are $number\ of\ candidates * |L|$. Finally, the tuple (i, y_i', l_k) associated with the highest score in $\boldsymbol{o}_i$ predicts that y_i' is the parent for i with temporal relation label l_k.

4.5 Variations of the Basic Neural Model

4.5.1 Linguistically Enriched Models

A variation of the basic neural model is a model that takes a few linguistic features as input explicitly. In this model, we extend the pair representation $\boldsymbol{g}_{i,y_i'}$ with local features: $\boldsymbol{g}_{i,y_i'} = [\boldsymbol{x}_i, \boldsymbol{x}_{y_i'}, \phi_{i,y_i'}]$.

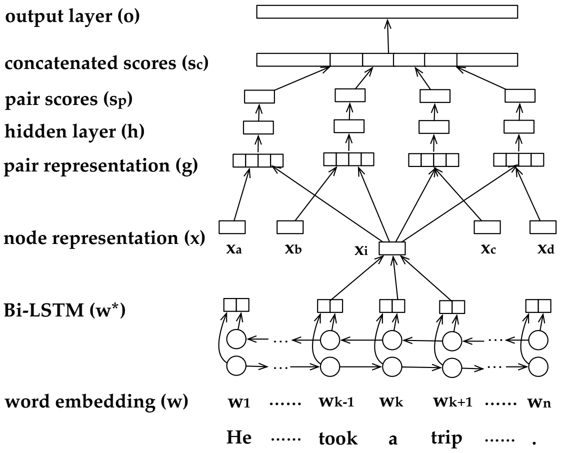

Figure 2: Neural Ranking Model Architecture. $\boldsymbol{x_i}$ is the current child node, and $\boldsymbol{x_a}, \boldsymbol{x_b}, \boldsymbol{x_c}, \boldsymbol{x_d}$ are the candidate parent nodes for $\boldsymbol{x_i}$. Arrows from Bi-LSTM layer to $\boldsymbol{x_a}, \boldsymbol{x_b}, \boldsymbol{x_c}, \boldsymbol{x_d}$ are not shown.

Time and event type feature: Stage 1 of the pipeline not only extracts text spans that are time expressions or events, but also labels them with pre-defined categories of different types of time expressions and events. Readers are referred to Zhang and Xue (2018) for the full category list. Through a careful examination of the data, we notice that time expressions or events are selective as to what types of time expression or events can be their parent. In other words, the category of the child time expression or event has a strong indication on which candidate can be its parent. For example, a time expression's parent can only be another time expression or a pre-defined meta node, and can never be an event; and an eventive event's parent is almost certainly another eventive event, and is highly unlikely to be a stative event. Therefore, we include the time expression and event type information predicted by stage 1 in this model as a feature. More formally, we represent a time/event type as a fixed-length embedding $\boldsymbol{t}$, and concatenate it to the pair representation $\boldsymbol{g}_{i,y_i'} = [\boldsymbol{x}_i, \boldsymbol{x}_{y_i'}, \boldsymbol{t}_i, \boldsymbol{t}_{y_i'}]$.

Distance features: Distance information can be useful for predicting the parent of a child. Intuitively, candidates that are closer to the child are more likely to be the actual parent. Through data examination, we also find that a high percentage of nodes have parents in close proximity. Therefore,

---

[2] An alternative decoding approach would be to perform a global search for a Maximum Spanning Tree. However, due to the nature of temporal structures, our greedy decoding process rarely hits the constraints.

we include two distance features in this model: the node distance between a candidate and the child $\boldsymbol{nd}_{i,y'_i}$, and whether they are in the same sentence $\boldsymbol{ss}_{i,y'_i}$. One-hot representations are used for both features to represent according conditions listed in Table 1.

conditions for feature $\boldsymbol{nd}_{i,y'_i}$:
$i.node\_id - y'_i.node\_id = 1$
$i.node\_id - y'_i.node\_id > 1$ and $i.sent\_id = y'_i.sent\_id$
$i.node\_id - y'_i.node\_id > 1$ and $i.sent\_id \neq y'_i.sent\_id$
$i.node\_id - y'_i.node\_id < 1$
conditions for feature $\boldsymbol{ss}_{i,y'_i}$:
$i.sent\_id = y'_i.sent\_id$
$i.sent\_id \neq y'_i.sent\_id$

Table 1: Conditions for node distance and same sentence features.

The final pair representation for our linguistically enriched model is as follows:

$$\boldsymbol{g}_{i,y'_i} = [\boldsymbol{x}_i, \boldsymbol{x}_{y'_i}, \boldsymbol{t}_i, \boldsymbol{t}_{y'_i}, \boldsymbol{nd}_{i,y'_i}, \boldsymbol{ss}_{i,y'_i}]$$

4.5.2 Attention Model on Time and Event Representation

In the basic neural model, a straight-forward sum-pooling is used as the multi-word time expression and event representation. However, multi-word event expressions usually have meaning-bearing head words. For example, in the event "took a trip", "trip" is more representative than "took" and "a". Therefore, we add an attention mechanism (Bahdanau et al., 2014) over the Bi-LSTM output vectors in each multi-word expression to learn a task-specific notion of headedness (Lee et al., 2017):

$$\alpha_t = tanh(\boldsymbol{W} \cdot \boldsymbol{w}^*_t)$$

$$w_{i,t} = \frac{exp[\alpha_t]}{\sum_{k=START(i)}^{END(i)} exp[\alpha_k]}$$

$$\hat{\boldsymbol{x}}_i = \sum_{t=START(i)}^{END(i)} w_{i,t} \cdot \boldsymbol{w}^*_t$$

where $\hat{\boldsymbol{x}}_i$ is a weighted sum of Bi-LSTM output vectors in span i. The weights $w_{i,t}$ are automatically learned. The final pair representation for our attention model is as follows:

$$\boldsymbol{g}_{i,y'_i} = [\boldsymbol{x}_i, \boldsymbol{x}_{y'_i}, \boldsymbol{t}_i, \boldsymbol{t}_{y'_i}, \boldsymbol{nd}_{i,y'_i}, \boldsymbol{ss}_{i,y'_i}, \hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_{y'_i}]$$

This model variation is also beneficial in an end-to-end system, where time expression and event spans are automatically extracted in Stage 1. When extracted spans are not guaranteed correct time expressions and events, an attention layer on a slightly larger context of an extracted span has a better chance of finding representative head words than a sum-pooling layer strictly on words within a event or time expression span.

5 Experiments

5.1 Data

All of our experiments are conducted on the datasets described in Zhang and Xue (2018). This is a temporal dependency structure corpus in Chinese. It covers two domains: news reports and narrative fairy tales. It consists of 115 news articles sampled from Chinese TempEval2 datasets (Verhagen et al., 2010) and Chinese Wikipedia News[3], and 120 fairy tale stories sampled from Grimm Fairy Tales[4]. 20% of this corpus, distributed evenly on both domains, are double annotated with high inter-annotator agreements. We use this part of the data as our development and test datasets (10% documents for development and 10% for testing), and the remaining 80% as our training dataset.

5.2 Baseline Systems

We build two baseline systems to compare with our neural model. The first is a simple baseline which links every time expression or event to its immediate previous time expression or event. According to our data, if only position information is considered, the most likely parent for a child is its immediate previous time expression or event. This baseline uses the most common temporal relation edge label in the training datasets, i.e. "overlap" for news data, and "before" for grimm data.

The second baseline is a more competitive baseline for stage 2 in the pipeline. It takes the output of the first stage as input, and uses a similar ranking architecture but with logistic regression classifiers instead of neural classifiers. The purpose of this baseline is to compare our neural models against a traditional statistical model under otherwise similar settings. We conduct robust feature engineering on this logistic regression model to make sure it is a strong benchmark to compete against. Table 2 lists the features and feature combinations used in this model.

---

[3] `https://zh.wikinews.org`

[4] `https://www.grimmstories.com/zh/grimm_tonghua/index`

time type and event type features:
$i.type$ and $y_i'.type$
if $i.type = absolute\ time$ and $y_i'.type = root$
if $i.type = time$ and $y_i'.type = root$
are $i.type$ and $y_i'.type$ time, eventive, or stative
are $i.type$ and $y_i'.type$ root, time, or event
are $i.type$ and $y_i'.type$ root, time, eventive, or stative
if $i.type = y_i'.type = event$ and $\hat{y}.type = state$, for all $\hat{y}$ between i and y_i'
distance features:
if $i.sent\_id = y_i'.sent\_id$
$i.node\_id - y_i'.node\_id$
if $i.node\_id - y_i'.node\_id = 1$
combination features:
if $i.type = state$ and $i.sent\_id \neq y_i'.sent\_id$
if $i.type = state$ and $i.node\_id - y_i'.node\_id = 1$
if $i.type = y_i'.type = event$ and $i.node\_id - y_i'.node\_id = 1$
if $i.type = state$ and $y_i'.type = event$ and $i.node\_id - y_i'.node\_id = 1$ and $i.node\_id\_in\_sent = 1$ and $i.sent\_id \neq 1$
other features:
if i and y_i' are in quotation marks

Table 2: Features in the logistic regression system.

evaluated	**news**			**grimm**		
label	**p**	**r**	**f**	**p**	**r**	**f**
all span	.81	.74	.78	.83	.74	.78
time	.83	.81	.82	.97	.62	.76
event	.81	.73	.77	.83	.74	.78

Table 3: Stage 1 cross-validation on span detection and binary time/event recognition.

time/event type	**news**	**grimm**
vague time	.77	.82
concrete absolute	.67	-
concrete relative	.75	-
event	.61	.77
state	.65	.61
completed	.62	.26
modalized	.46	.31

Table 4: Stage 1 (time/event type recognition) cross-validation f1-scores on the full set.

5.3 Evaluation

We perform two types of evaluations for our systems. First, we evaluate the stages of the pipeline and the entire pipeline, i.e. end-to-end systems where both time expression and event recognition, as well as temporal dependency structures are automatically predicted. Our models are compared against the two strong baselines described in §5.2. These evaluations are described in §5.3.1.

The second evaluation focuses only on the temporal relation structure parsing part of our pipeline (i.e. Stage 2), using gold standard time expression and event spans and labels. Since most previous work on temporal relation identification use gold standard time expression and event spans, this evaluation gives us some sense of how our models perform against models reported in previous work even though a strict comparison is impossible because different data sets are used. These evaluations are described in §5.3.2.

All neural networks in this work are implemented in Python with the DyNet library (Neubig et al., 2017). The code is publicly available[5].

For Stage 1, all models are trained with Adam optimizer with early stopping and learning rate 0.001. The dimensions of word embeddings, POS tag embeddings, Bi-LSTM output vectors, and MLP hidden layers are tuned on the dev set to 256, 32, 256, and 256 respectively. POS tags in Stage 1 are acquired using the joint POS tagger from Wang and Xue (2014). The tagger is trained on Chinese Treebank 7.0 (Xue et al., 2010). For Stage 2, the dimensions of word embeddings, time/event type embeddings, Bi-LSTM output vectors, and MLP hidden layers are tuned on the dev set to 32, 16, 32, and 32 respectively. The optimizer is Adam with early stopping and learning rate 0.001.

5.3.1 End-to-End System Evaluation

Stage 1: Time and Event Recognition For Stage 1 in the pipeline, we perform BIO tagging with the full set of time expression and event types (i.e. a 11-way classification on all extracted spans). Extracted spans will be nodes in the final dependency tree, and time/event types will support features in the next stage. We evaluate Stage 1 performance using 10-fold cross-validation of the entire data set. We use the "exact match" evaluation metrics for BIO sequence labeling tasks, and compute precision, recall, and f-score for each label type.

We first ignore fine-grained time/event types and only evaluate unlabeled span detection and time/event binary classification to show how well our system identify events and time expressions, and how well our system distinguishes time expressions from events. Table 3 shows the cross-validation results on these two evaluations. Span detection and event recognition show similar per-

---

[5] `https://github.com/yuchenz/tdp_ranking`

		news				grimm			
	model	unlabeled f		labeled f		unlabeled f		labeled f	
		dev	test	dev	test	dev	test	dev	test
temporal relation parsing with gold spans	Baseline-simple	.64	.68	.47	.43	.78	**.79**	.39	.39
	Baseline-logistic	.81	.79	.63	.54	.74	.74	.60	.63
	Neural-basic	.78	.75	.67	.57	.72	.74	.60	.63
	Neural-enriched	.80	.78	.67	.59	.76	.77	.63	.65
	Neural-attention	**.83**	**.81**	**.76**	**.70**	**.79**	**.79**	**.66**	**.68**
end-to-end systems with automatic spans	Baseline-simple	.39	.40	.26	.25	**.44**	.47	.27	.25
	Baseline-logistic	.36	.34	.24	.22	.43	**.49**	.33	.37
	Neural-basic	.37	.36	.21	.23	.42	.45	.33	.35
	Neural-enriched	.51	.52	.32	.35	**.44**	**.49**	.33	.37
	Neural-attention	**.54**	**.54**	**.36**	**.39**	**.44**	**.49**	**.35**	**.39**

Table 5: Stage 2 results (f-scores) with gold spans and timex/event labels (top), and automatic spans and timex/event labels generated by stage 1 (bottom). Best performances are in bold.

formance on both news and narrative domains. Time expressions have a higher recognition rate than events in news data, which is consistent with the observation that time expressions usually have a more limited vocabulary and more strict lexical patterns. On the other hand, due to the scarcity of time expressions in the Grimm data, time expression recognition in this domain has a very high precision but low recall, which results in a much lower f-score than news.

Labeled full set evaluation results on time/event type classification are reported in Table 4. Time expressions have higher recognition rates than events on both domains, and dominant event types ("event", "state", etc.) have higher and more stable recognition rates than other types. Event types with very few training instances, such as "modalized event" (<7%), achieve lower and more unstable recognition rates. Other types with less than 2% instances achieve close to 0 recognition f-scores, and are not reported in this table.

Stage 2: Temporal Dependency Parsing For Stage 2 in the pipeline, we conduct experiments on the five systems described above: a simple baseline, a logistic regression baseline, a basic neural model, a linguistically enriched neural model, and an attention neural model. All models are trained on automatically predicted spans of time expressions and events, and time/event types generated by Stage 1 using 10-fold cross-validation, with gold standard edges (and edge labels) mapped onto the automatic spans. Evaluations in Stage 2 are against gold standard spans and edges, and evaluation metrics are precision, recall, and f-score on $\langle child, parent\rangle$ tuples for unlabeled trees, and $\langle child, relation, parent\rangle$ triples for labeled trees.

Bottom rows in Table 5 report the end-to-end performance of our five systems on both domains. On both labeled and unlabeled parsing, our basic neural model with only lexical input performs comparable to the logistic regression model. And our enriched neural model with only three simple linguistic features outperforms both the logistic regression model and the basic neural model on news, improving the performance by more than 10%. However, our models only slightly improve the unlabeled parsing over the simple baseline on narrative Grimm data. This is probably due to (1) it is a very strong baseline to link every node to its immediate previous node, since in an narrative discourse linear temporal sequences are very common; and (2) most events breaking the temporal linearity in a narrative discourse are implicit stative descriptions which are harder to model with only lexical and distance features. Finally, attention mechanism improves temporal relation labeling on both domains.

5.3.2 Temporal Relation Evaluation

To facilitate comparison with previous work where gold events are used as parser input, we report our results on temporal dependency parsing with gold time expression and event spans in Table 5 (top rows). These results are in the same ballpark as what is reported in previous work on temporal relation extraction. The best performance in Kolomiyets et al. (2012) are 0.84 and 0.65 f-scores for unlabeled and labeled parses, achieved by temporal structure parsers trained and evaluated on narrative children's stories. Our best performing model (Neural-attention) reports 0.81 and 0.70 f-scores on unlabeled and labeled parses respectively, showing similar performance. It is important to note, however, that these two works use different data sets, and are not directly comparable. Finally, parsing accuracy with gold time/event spans as input is substantially higher than that with predicted spans, showing the effects of error propagation.

6 Error Analysis

We perform error analysis on the output of our best model (Neural-attention) on the development data sets. We focus on analyzing our neural ranking model (i.e. Stage 2), with gold time expression and event spans and labels as input.

First, we look at errors by the types of antecedents. Most events in both news and grimm data depend on their immediate previous event or time expression as their reference time parent. 71% of the events in the news data set and 78% of the events in the Grimm data have the immediate previous node as their antecedent. The confusion matrix in Table 6 illustrates how strongly this bias affects our models. Our model learns the bias and incorrectly links around half of the events (47% in news and 46% in grimm) to their immediate previous node when the *correct* temporal dependency is further back in the text.

	news			**grimm**		
	pre	**far**	**total**	**pre**	**far**	**total**
pre	317	11	328	750	60	810
far	65	72	137	104	122	226
total	382	83	465	854	182	1036

Table 6: Parent node confusion matrix. Rows are gold parents and columns are automatically parsed parents. "pre" means the parent is the immediate previous node of the child event, "far" means the parent is further back from the child event.

Second, we look at errors in temporal relation labels. Considering only correctly recognized parent-child pairs, we draw a confusion matrix as in Table 7. Our data has very few *after* relations in both domains, which explains why the model has difficulty identifying this relation. There are also very few *include* and *depend-on* relations in the Grimm data, however they are identified with a relatively high accuracy. This is probably because, according to the temporal dependency structure design (Zhang and Xue, 2018), these relations hold only between restricted pairs of parent and child: *include* requires a time expression parent and an event child, and *depend-on* requires that the parent be the rootf. The main confusion among temporal relations is between *before* and *overlap*. In news data, with a high occurrence of *overlap* relations (60% *overlap* and 5% *before*), most *before* parents are wrongly recognized as *overlap*. Grimm data has a more balanced distribution of these two temporal relations (46% *overlap* and 50% *before*), however, 13% *before* and 17% *overlap* are wrongly labeled as the other.

news	**be**	**af**	**ov**	**in**	**de**	**total**
before	1	0	21	2	0	24
after	1	0	1	0	0	2
overlap	1	0	295	0	0	296
include	0	0	4	52	0	56
depend-on	0	0	0	0	117	117
total	3	0	321	54	117	495

grimm	**be**	**af**	**ov**	**in**	**de**	**total**
before	367	0	55	0	0	422
after	1	0	2	0	0	3
overlap	74	1	314	0	0	389
include	3	0	0	10	0	13
depend-on	0	0	0	0	12	12
total	445	1	371	10	12	839

Table 7: Temporal relation confusion matrix. Rows are gold relation labels and columns are automatic relation labels. "be, af, ov, in, de" stand for "before, after, overlap, include, and depend-on".

7 Related Work

7.1 Related Work on Temporal Relation Modeling

There is a significant amount of research on temporal relation extraction (Bethard et al., 2007; Bethard, 2013; Chambers and Jurafsky, 2008; Chambers et al., 2014; Ning et al., 2018a). Most of the previous work models temporal relation extraction as pair-wise classification between individual pairs of events and/or time expressions. Some of the models also add a global reasoning step to local pair-wise classification, typically using Integer Linear Programming, to exploit the transitivity property of temporal relations (Chambers and Jurafsky, 2008). Such a pair-wise classification approach is often dictated by the way the data is annotated. In most of the widely used temporal data sets, temporal relations between individual pairs of events and/or time expressions are annotated independently of one another (Pustejovsky et al., 2003; Chambers et al., 2014; Styler IV et al., 2014; O'Gorman et al., 2016; Mostafazadeh et al., 2016).

Our work is most closely related to that of Kolomiyets et al. (2012), which also treats temporal relation modeling as temporal dependency structure parsing. However, their dependency structure, as described in Bethard et al. (2012), is only over events, excluding time expressions which are an important source of temporal information, and it also excludes *states* (stative events), which makes the temporal dependency structure incomplete. Moreover, their corpus only consists of data in the narrative stories domain. We instead choose to develop our model based on the data set described in Zhang and Xue (2018), which introduces a more comprehensive and linguistically grounded annotation scheme for temporal dependency structures. This structure includes both events and time expressions, and uses the linguistic notion of *temporal anaphora* to guide the annotation of the temporal dependency structure. Since in this temporal dependency structure each parent-child pair is considered to be an instance of temporal anaphora, the parent is also called the *antecedent* and the child is also referred to as the *anaphor*. Their corpus consists of data from two domains: news reports and narrative stories.

More recently, Ning et al. (2018b) proposed a semi-structured approach to model temporal relations in a text. Based on the observation that not all pairs of events have well-defined temporal relations, they propose a multi-axis representation in which well-defined temporal relations only hold between events on the same axis. The temporal relations between events in a text form multiple disconnected subgraphs. Like other work before them, their annotation scheme only covers events, to the exclusion of time expressions.

7.2 Related Work on Neural Dependency Parsing

Most prior work on neural dependency parsing is aimed at syntactic dependency parsing, i.e. parsing a sentence into a dependency tree that represents the syntactic relations among the words. Recent work on dependency parsing typically uses transition-based or graph-based architectures combined with contextual vector representations learned with recurrent neural networks (e.g. Bi-LSTMs) (Kiperwasser and Goldberg, 2016).

Temporal dependency parsing is, however, different from syntactic dependency parsing. In temporal dependency parsing, for each event or time expression, there is more than one other event or time expression that can serve as its reference time, while the most closely related one is selected as the gold standard reference time parent. This naturally falls into a ranking process where all possible reference times are ranked and the best is selected. In this sense our neural ranking model for temporal dependency parsing is closely related to the neural ranking model for coreference resolution described in Lee et al. (2017), both of which extract related spans of words (entity mentions for coreference resolution, and events or time expressions for temporal dependency parsing). However, our temporal dependency parsing model differs from Lee et al's coreference model in that the ranking model for coreference only needs to output the best candidate for each individual pairing and cluster all pairs that are coreferent to each other. In contrast, our ranking model for temporal dependency parsing needs to rank not only the candidate antecedents but also the temporal relations between the antecedent and the anaphor. In addition, the model also adds connectivity and acyclic constraints in the decoding process to guarantee a tree-structured output.

8 Conclusion and Future Work

In this paper, we present the first end-to-end neural temporal dependency parser. We evaluate the parser with both gold standard and automatically recognized time expressions and events. In both experimental settings, the parser outperforms two strong baselines and shows competitive results against prior temporal systems.

Our experimental results show that the model performance drops significantly when automatically predicted event and time expressions are used as input instead of gold standard ones, indicating an error propagation problem. Therefore, in future work we plan to develop joint models that simultaneously extract events and time expressions, and parse their temporal dependency structure.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 2, pages 10–14.

Steven Bethard, Oleksandr Kolomiyets, and Marie-Francine Moens. 2012. Annotating story timelines as temporal dependency structures. In *Proceedings of the eighth international conference on language resources and evaluation (LREC)*, pages 2721–2726. ELRA.

Steven Bethard, James H Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 11–18. IEEE.

Steven Bethard, Guergana Savova, Martha Palmer, and James Pustejovsky. 2017. Semeval-2017 task 12: Clinical tempeval. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 565–572, Vancouver, Canada. Association for Computational Linguistics.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Nathaniel Chambers and Daniel Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *EMNLP-2008*.

Nathaniel Chambers, Shan Wang, and Daniel Jurafsky. 2007. Classifying temporal relations between events. In *ACL-2007*.

Dmitriy Dligach, Timothy Miller, Chen Lin, Steven Bethard, and Guergana Savova. 2017. Neural temporal relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 746–751.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv:1603.04351*.

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2012. Extracting narrative timelines as temporal dependency structures. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 88–97. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. Caters: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the The 4th Workshop on EVENTS: Definition, Detection, Coreference, and Representation, San Diego, California, June. Association for Computational Linguistics*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. 2018a. Improving temporal relation extraction with a globally acquired statistical resource. *arXiv preprint arXiv:1804.06020*.

Qiang Ning, Hao Wu, and Dan Roth. 2018b. A multi-axis annotation scheme for event temporal relations. *arXiv preprint arXiv:1804.07828*.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer event description: Integrating event coreference with temporal, causal and bridging annotation. *Computing News Storylines*, page 47.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40. Lancaster, UK.

William F Styler IV, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova, et al. 2014. Temporal annotation in the clinical domain. *Transactions of the Association for Computational Linguistics*, 2:143–154.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 57–62. Association for Computational Linguistics.

Zhiguo Wang and Nianwen Xue. 2014. Joint pos tagging and transition-based constituent parsing in chinese with non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 733–742.

Nianwen Xue, Zixin Jiang, Xiuhong Zhong, Martha Palmer, Fei Xia, Fu-Dong Chiou, and Meiyu Chang. 2010. Chinese treebank 7.0. *Linguistic Data Consortium, Philadelphia*.

Yuchen Zhang and Nianwen Xue. 2018. Structured interpretation of temporal relations. In *Proceedings of 11th edition of the Language Resources and Evaluation Conference (LREC-2018)*, Miyazaki, Japan.

Causal Explanation Analysis on Social Media

Youngseo Son, Nipun Bayas, and H. Andrew Schwartz
Stony Brook University
Stony Brook, NY
`{yson,nbayas,has}@cs.stonybrook.edu`

Abstract

Understanding causal explanations — reasons given for happenings in one's life — has been found to be an important psychological factor linked to physical and mental health. Causal explanations are often studied through manual identification of phrases over limited samples of personal writing. Automatic identification of causal explanations in social media, while challenging in relying on contextual and sequential cues, offers a larger-scale alternative to expensive manual ratings and opens the door for new applications (e.g. studying prevailing beliefs about causes, such as climate change). Here, we explore automating causal explanation analysis, building on discourse parsing, and presenting two novel subtasks: *causality detection* (determining whether a causal explanation exists at all) and *causal explanation identification* (identifying the specific phrase that is the explanation). We achieve strong accuracies for both tasks but find different approaches best: an SVM for causality prediction ($F1 = 0.791$) and a hierarchy of Bidirectional LSTMs for *causal explanation identification* ($F1 = 0.853$). Finally, we explore applications of our complete pipeline ($F1 = 0.868$), showing demographic differences in mentions of causal explanation and that the association between a word and sentiment can change when it is used within a causal explanation.

1 Introduction

Explanations of happenings in one's life, *causal explanations*, are an important topic of study in social, psychological, economic, and behavioral sciences. For example, psychologists have analyzed people's causal explanatory style (Peterson et al., 1988) and found strong negative relationships with depression, passivity, and hostility, as well as positive relationships with life satisfaction, quality of life, and length of life (Scheier et al., 1989; Carver and Gaines, 1987; Peterson et al., 1988).

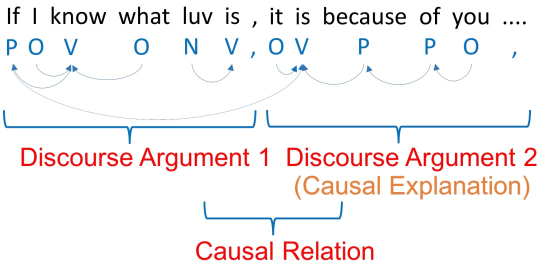

Figure 1: A casual relation characterizes the connection between two discourse arguments, one of which is the *causal explanation*.

To help understand the significance of causal explanations, consider how they are applied to measuring optimism (and its converse, pessimism) (Peterson et al., 1988). For example, in "My parser failed *because I always have bugs.*", the emphasized text span is considered a causal explanation which indicates pessimistic personality – a negative event where the author believes the cause is pervasive. However, in "My parser failed *because I barely worked on the code.*", the explanation would be considered a signal of optimistic personality – a negative event for which the cause is believed to be short-lived.

Language-based models which can detect causal explanations from everyday social media language can be used for more than automating optimism detection. Language-based assessments would enable other large-scale downstream tasks: tracking prevailing causal beliefs (e.g., about climate change or autism), better extracting process knowledge from non-fiction (e.g., gravity causes objects to move toward one another), or detecting attribution of blame or praise in product or service reviews ("I loved this restaurant because the fish was cooked to perfection").

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3350–3359
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

In this paper, we introduce causal explanation analysis and its subtasks of detecting the presence of causality (*causality prediction*) and identifying explanatory phrases (*causal explanation identification*). There are many challenges to achieving these task. First, the ungrammatical texts in social media incur poor syntactic parsing results which drastically affect the performance of discourse relation parsing pipelines [1]. Many causal relations are *implicit* and do not contain any discourse markers (e.g., 'because'). Further, *Explicit* causal relations are also more difficult in social media due to the abundance of abbreviations and variations of discourse connectives (e.g., 'cuz' and 'bcuz').

Prevailing approaches for social media analyses, utilizing traditional linear models or bag of words models (e.g., SVM trained with n-gram, part-of-speech (POS) tags, or lexicon-based features) alone do not seem appropriate for this task since they simply cannot segment the text into meaningful discourse units or discourse arguments [2] such as clauses or sentences rather than random consecutive token sequences or specific word tokens. Even when the discourse units are clear, parsers may still fail to accurately identify discourse relations since the content of social media is quite different than that of newswire which is typically used for discourse parsing.

In order to overcome these difficulties of discourse relation parsing in social media, we simplify and minimize the use of syntactic parsing results and capture relations between discourse arguments, and investigate the use of a recursive neural network model (RNN). Recent work has shown that RNNs are effective for utilizing discourse structures for their downstream tasks (Ji and Smith, 2017; Bhatia et al., 2015; Wieting et al., 2015; Paulus et al., 2014), but they have yet to be directly used for discourse relation prediction in social media. We evaluated our model by comparing it to off-the-shelf end-to-end discourse relation parsers and traditional models. We found that the SVM and random forest classifiers work better than the LSTM classifier for the causality detection, while the LSTM classifier outperforms other models for identifying causal explanation.

The contributions of this work include: (1) the proposal of models for both (a) causality prediction and (b) causal explanation identification, (2) the extensive evaluation of a variety of models from social media classification models and discourse relation parsers to RNN-based application models, demonstrating that feature-based models work best for causality prediction while RNNs are superior for the more difficult task of causal explanation identification, (3) performance analysis on architectural differences of the pipeline and the classifier structures, (4) exploration of the applications of causal explanation to downstream tasks, and (5) release of a novel, anonymized causality Facebook dataset along with our causality prediction and causal explanation identification models.

2 Related Work

Identifying causal explanations in documents can be viewed as discourse relation parsing. The Penn Discourse Treebank (PDTB) (Prasad et al., 2007) has a 'Cause' and 'Pragmatic Cause' discourse type under a general 'Contingency' class and Rhetorical Structure Theory (RST) (Mann and Thompson, 1987) has a 'Relations of Cause'. In most cases, the development of discourse parsers has taken place *in-domain*, where researchers have used the existing annotations of discourse arguments in newswire text (e.g. Wall Street Journal) from the discourse treebank and focused on exploring different features and optimizing various types of models for predicting relations (Pitler et al., 2009; Park and Cardie, 2012; Zhou et al., 2010). In order to further develop automated systems, researchers have proposed end-to-end discourse relation parsers, building models which are trained and evaluated on the annotated PDTB and RST Discourse Treebank (RST DT). These corpora consist of documents from Wall Street Journal (WSJ) which are much more well-organized and grammatical than social media texts (Biran and McKeown, 2015; Lin et al., 2014; Ji and Eisenstein, 2014; Feng and Hirst, 2014).

Only a few works have attempted to parse discourse relations for out-of-domain problems such as text categorizations on social media texts; Ji and Bhatia used models which are pretrained with RST DT for building discourse structures from movie reviews, and Son adapted the PDTB discourse re-

---

[1] Off-the-shelf Penn Discourse Treebank (PDTB) end-to-end parsers perform poorly on our Facebook causal prediction dataset (see Table 3)

[2] Each discourse relation theory uses a different term for minimal discourse text spans: 'Elementary Discourse Unit (EDU)' in RST and 'Discourse Argument' in PDTB. We will call it 'Discourse Argument' in this paper, since we adapted the PDTB text segmentation method.

lation parsing approach for capturing counterfactual conditionals from tweets (Bhatia et al., 2015; Ji and Smith, 2017; Son et al., 2017). These works had substantial differences to what propose in this paper. First, Ji and Bhatia used a pretrained model (not fully optimal for some parts of the given task) in their pipeline; Ji's model performed worse than the baseline on the categorization of legislative bills, which is thought to be due to legislative discourse structures differing from those of the training set (WSJ corpus). Bhatia also used a pretrained model finding that utilizing discourse relation features did not boost accuracy (Bhatia et al., 2015; Ji and Smith, 2017). Both Bhatia and Son used manual schemes which may limit the coverage of certain types of positive samples–Bhatia used a hand-crafted schema for weighting discourse structures for the neural network model and Son manually developed seven surface forms of counterfactual thinking for the rule-based system (Bhatia et al., 2015; Son et al., 2017). We use social-media-specific features from pretrained models which are directly trained on tweets and we avoid any hand-crafted rules except for those included in the existing discourse argument extraction techniques.

The automated systems for discourse relation parsing involve multiple subtasks from segmenting the whole text into discourse arguments to classifying discourse relations between the arguments. Past research has found that different types of models and features yield varying performance for each subtask. Some have optimized models for discourse relation classification (i.e. given a document indicating if the relation existing) without discourse argument parsing using models such as Naive-Bayes or SVMs, achieve relatively stronger accuracies but a simpler task than that associated with discourse arguments (Park and Cardie, 2012; Zhou et al., 2010; Pitler et al., 2009). Researchers who, instead, tried to build the end-to-end parsing pipelines considered a wider range of approaches including sequence models and RNNs (Biran and McKeown, 2015; Feng and Hirst, 2014; Ji and Eisenstein, 2014; Li et al., 2014). Particularly, when they tried to utilize the discourse structures for out-domain applications, they used RNN-based models and found that those models are advantageous for their downstream tasks (Bhatia et al., 2015; Ji and Smith, 2017).

In our case, for identifying causal explanations from social media using discourse structure, we build an RNN-based model for its structural effectiveness in this task (see details in section 3.2). However, we also note that simpler models such as SVMs and logistic regression obtained the state-of-the-art performances for text categorization tasks in social media (Lynn et al., 2017; Mohammad et al., 2013), so we build relatively simple models with different properties for each stage of the full pipeline of our parser.

3 Methods

We build our model based on PDTB-style discourse relation parsing since PDTB has a relatively simpler text segmentation method;[3] for explicit discourse relations, it finds the presence of discourse connectives within a document and extracts discourse arguments which parametrize the connective while for implicit relations, it considers all adjacent sentences as candidate discourse arguments.

3.1 Dataset

We created our own causal explanation dataset by collecting 3,268 random Facebook status update messages. Three well-trained annotators manually labeled whether or not each message contains the causal explanation and obtained 1,600 causality messages with substantial agreement ($\kappa = 0.61$). We used the majority vote for our gold standard. Then, on each causality message, annotators identified which text spans are causal explanations.

For each task, we used 80% of the dataset for training our model and 10% for tuning the hyperparameters of our models. Finally, we evaluated all of our models on the remaining 10% (Table 1 and Table 2). For causal explanation detection task, we extracted discourse arguments using our parser and selected discourse arguments which most cover the annotated causal explanation text span as our gold standard.

3.2 Model

We build two types of models. First, we develop feature-based models which utilize features of the successful models in social media analysis and causal relation discourse parsing. Then, we

---

[3] RST parsing builds fully hierarchical discourse tree structures out of the whole span of target text which highly depends on syntactic parsing and exact matching of elementary discourse units which are extremely hard to obtain from social media texts

Dataset	Causality	Non-Causal	Total
Training	1,284	1,330	2,614
Validation	150	177	327
Test	164	163	327
Total	1,598	1,670	3,268

Table 1: Number of messages containing causality or not in our dataset.

Causality messages	CE DA	Total DA
Training	1,278	5,606
Validation	160	652
Test	160	757
Total	1,598	7,015

Table 2: The number of discourse arguments in causality messages. Across 1,598 total causality messages, we found 7,015 discourse arguments (Total DA) and the one which covers annotated causal explanation are used as causal explanation discourse arguments (CE DA)

build a recursive neural network model which uses distributed representation of discourse arguments as this approach can even capture latent properties of causal relations which may exist between distant discourse arguments. We specifically selected bidirectional LSTM since the model with the discourse distributional structure built in this form outperformed the traditional models in similar NLP downstream tasks (Ji and Smith, 2017).

Discourse Argument Extraction As the first step of our pipeline, we use Tweebo parser (Kong et al., 2014) to extract syntactic features from messages. Then, we demarcate sentences using punctuation (',') tag and periods. Among those sentences, we find discourse connectives defined in PDTB annotation along with a Tweet POS tag for conjunction words which can also be a discourse marker. In order to decide whether these connectives are really discourse connectives (e.g., I went home, **but** he stayed) as opposed to simple connections of two words (I like apple **and** banana) we see if verb phrases [4] exist before and after the connective by using dependency parsing results. Although discourse connective disambiguation is a complicated task which can be much improved by syntactic features (Pitler and Nenkova, 2009), we try to minimize effects of syntactic parsing and simplify it since it is highly error-prone in social media. Finally, according to visual inspection, emojis ('E' tag) are crucial for discourse relation in social media so we take them as separate discourse arguments (e.g.,in "My test result... :(" the sad feeling is caused by the test result, but it cannot be captured by plain word tokens).

Feature Based Models We trained a linear SVM, an rbf SVM, and a random forest with N-gram, charater N-gram, and tweet POS tags, sentiment tags, average word lengths and word counts from each message as they have a pivotal role in the models for many NLP downstream tasks in social media (Mohammad et al., 2013; Lynn et al., 2017). In addition to these features, we also extracted *First-Last, First3* features and *Word Pairs* from every adjacent pair of discourse arguments since these features were most helpful for causal relation prediction (Pitler et al., 2009). *First-Last, First3* features are first and last word and first three words of two discourse arguments of the relation, and *Word Pairs* are the cross product of words of those discourse arguments. These two features enable our model to capture interaction between two discourse arguments. (Pitler et al., 2009) reported that these two features along with verbs, modality, context, and polarity (which can be captured by N-grams, sentiment tags and POS tags in our previous features) obtained the best performance for predicting Contingency class to which causality belongs.

Recursive Neural Network Model We load the GLOVE word embedding (Pennington et al., 2014) trained in Twitter [5] for each token of extracted discourse arguments from messages. For the distributional representation of discourse arguments, we run a Word-level LSTM on the words' embeddings within each discourse argument and concatenate last hidden state vectors of forward LSTM ($\overrightarrow{h}$) and backward LSTM ($\overleftarrow{h}$) which is suggested by (Ji and Smith, 2017) ($DA = [\overrightarrow{h}; \overleftarrow{h}]$). Then, we feed the sequence of the vector representation of discourse arguments to the Discourse-argument-level LSTM (DA-level LSTM) to make a final prediction with log softmax function. With this structure, the model can learn the representation of interaction of tokens inside each discourse argument, then capture discourse relations across all of the discourse argu-

---

[4] minimal discourse unit is verb phrases with very few exceptions (Prasad et al., 2007)

[5] `http://nlp.stanford.edu/data/glove.twitter.27B.zip`

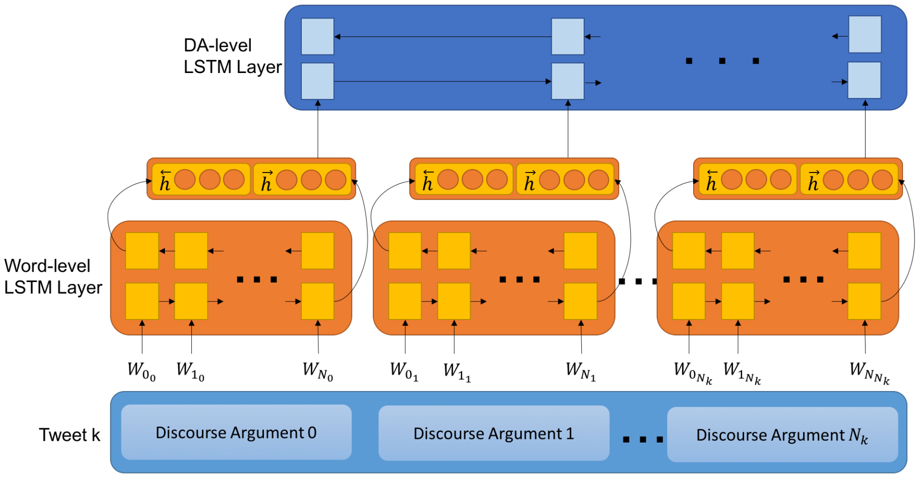

Figure 2: LSTM classifier for causality detection and explanation identification

ments in each message (Figure 2). In order to prevent the overfitting, we added a dropout layer between the Word-level LSTM and the DA-level LSTM layer.

Architectural Variants We also explore subsets of the full RNN architecture, specifically with one of the two LSTM layers removed. In the first model variant, we directly input all word embeddings of a whole message to a BiLSTM layer and make prediction (**Word LSTM**) without the help of the distributional vector representations of discourse arguments. In the second model variant, we take the average of all word embeddings of each discourse argument ($DA_k = \frac{1}{N_k}\sum_{i=1}^{N_k} W_i$), and use them as inputs to a BiLSTM layer (**DA AVG LSTM**) as the average vector of embeddings were quite effective for representing the whole sequence (Ji and Smith, 2017; Wieting et al., 2015). As with the full architectures, for CP both of these variants ends with a many-to-one classification per message, while the CEI model ends with a sequence of classifications.

3.3 Experiment

Feature Based Model We explored three types of models (RBF SVM, Linear SVM, and Random Forest Classifier) which have previously been shown empirically useful for the language analysis in social media. We filtered out low frequency **Word Pairs** features as they tend to be noisy and sparse (Pitler et al., 2009). Then, we conducted univariate feature selection to restrict all remaining features to those showing at least a small relationship with the outcome. Specifically, we keep all features passing a family-wise error rate of $\alpha = 60$ with the given outcome. After comparing the performance of the optimized version of each model, we also conducted a feature ablation test on the best model in order to see how much each feature contributes to the causality prediction.

Neural Network Model We used bidirectional LSTMs for causality classification and causal explanation identification since the discourse arguments for causal explanation can show up either before and after the effected events or results and we want our model to be optimized for both cases. However, there is a risk of overfitting due to the dataset which is relatively small for the high complexity of the model, so we added a dropout layer (p=0.3) between the Word-level LSTM and the DA-level LSTM.

For tuning our model, we explore the dimensionality of word vector and LSTM hidden state vectors of discourse arguments of 25, 50, 100, and 200 as pretrained GLOVE vectors were trained in this setting. For optimization, we used Stochastic Gradient Descent (SGD) and Adam (Kingma and Ba, 2014) with learning rates 0.01 and 0.001.

We ignore missing word embeddings because our dataset is quite small for retraining new word embeddings. However, if embeddings are extracted as separate discourse arguments, we used the average of all vectors of all discourse arguments in that message. Average embeddings have performed well for representing text sequences in other tasks (Wieting et al., 2015).

Model	F1
(Biran and McKeown, 2015)	0.434
(Lin et al., 2014)	0.638
Linear SVM	**0.791**
RBF SVM	**0.777**
Random Forest	0.771
LSTM	0.758

Table 3: Causality prediction performance across different predictive models. Bold indicates significant improvement over the LSTM

Model	F1
All	0.791
- First-Last, First3	0.788
- Word Pairs	0.787
- POS tags	0.734
- (Char + Word) N-grams	0.769
- Sentiment tags	0.791

Table 4: Feature ablation test of Linear SVM for causality prediction

Model Evaluation We first use state-of-the-art PDTB taggers for our baseline (Lin et al., 2014; Biran and McKeown, 2015) for the evaluation of the causality prediction of our models ((Biran and McKeown, 2015) requires sentences extracted from the text as its input, so we used our parser to extract sentences from the message). Then, we compare how models work for each task and disassembled them to inspect how each part of the models can affect their final prediction performances. We conducted McNemar's test to determine whether the performance differences are statistically significant at $p < .05$.

4 Results

We investigated various models for both causality detection and explanation identification. Based on their performances on the task, we analyzed the relationships between the types of models and the tasks, and scrutinized further for the best performing models. For performance analysis, we reported weighted F1 of classes.

4.1 Causality Prediction

In order to classify whether a message contains causal relation, we compared off-the-shelf PDTB parsers, linear SVM, RBF SVM, Random forest and LSTM classifiers. The off-the-shelf parsers achieved the lowest accuracies ((Biran and McKeown, 2015) and (Lin et al., 2014) in Table 3). This result can be expected since 1) these models were trained with news articles and 2) they are trained for all possible discourse relations in addition to causal relations (e.g., contrast, condition, etc). Among our suggested models, SVM and random forest classifier performed better than LSTM and, in the general trend, the more complex the models were, the worse they performed. This suggests that the models with more direct and simpler learning methods with features might classify the causality messages better than the ones more optimized for capturing distributional information or non-linear relationships of features.

Model	Prec	Rec	F1
Linear SVM	0.773	0.727	0.743
RBF SVM	0.773	0.727	0.743
Random Forest	0.747	0.790	0.746
LSTM	**0.851**	**0.858**	**0.853**

Table 5: Causal explanation identification performance. Bold indicates significant imrpovement over next best model ($p < .05$)

Causality Classifier Analysis Table 4 shows the results of a feature ablation test to see how each feature contributes to causality classification performance of the linear SVM classifier. POS tags caused the largest drop in F1. We suspect POS tags played a unique role because discourse connectives can have various surface forms (e.g., because, cuz, bcuz, etc) but still the same POS tag 'P'. Also POS tags can capture the occurrences of modal verbs, a feature previously found to be very useful for detecting similar discourse relations (Pitler et al., 2009). N-gram features caused 0.022 F1 drop while sentiment tags did not affect the model when removed. Unlike the previous work where *First-Last, First3* and *Word pairs* tended to gain a large F1 increase for multiclass discourse relation prediction, in our case, they did not affect the prediction performance compared to other feature types such as POS tags or N-grams.

4.2 Causal Explanation Identification

In this task, the model identifies causal explanations given the discourse arguments of the causality message. We explored over the same models as those we used for causality (sans the output layer), and found the almost opposite trend of performances (see Table 5). The Linear SVM ob-

Model	CP (F1)	CEI (F1)
Full LSTM	0.758	0.853
DA AVG LSTM	0.685	0.818
Word LSTM	0.694	0.792

Table 6: The effect of Word-level LSTM (Word LSTM) and discourse argument LSTM (DA AVG LSTM) for causality prediction (CP) and causal explanation identification (CEI). Note that, as described in methods, there are architectural differences for CP and CEI models with the same names, most notably that the output layer is always a single classification for CP and a sequence of classifications for CEI.

tained lowest F1 while the LSTM model made the best identification performance. As opposed to the simple binary classification of the causality messages, in order to detect causal explanation, it is more beneficial to consider the relation across discourse arguments of the whole message and implicit distributional representation due to the implicit causal relations between two distant arguments.

4.3 Architectural Variants

For causality prediction, we experimented with only word tokens in the whole message without help of Word-level LSTM layer (**Word LSTM**), and F1 dropped by 0.064 (CP in Table 6). Also, when we used the average of the sequence of word embeddings of each discourse argument as an input to the DA-level LSTM and it caused F1 drop of 0.073. This suggests that the information gained from both the interaction of words in and in between discourse arguments help when the model utilizes the distributional representation of the texts.

For causal explanation identification, in order to test how the LSTM classifier works without its capability of capturing the relations between discourse arguments, we removed DA-level LSTM layer and ran the LSTM directly on the word embedding sequence for each discourse argument for classifying whether the argument is causal explanation, and the model had 0.061 F1 drop (**Word LSTM** in CEI in Table 6). Also, when we ran DA-level LSTM on the average vectors of the word sequences of each discourse argument of messages, F1 decreased to 0.818. This follows the similar pattern observed from other types of models performances (i.e., SVMs and Random Forest classifiers) that the models with higher complexity for capturing the interaction of discourse arguments tend to identify causal explanation with the higher accuracies.

For CEI task, we found that when the model ran on the sequence representation of discourse argument (**DA AVG LSTM**), its performance was higher than the plain sequence of word embeddings (**Word LSTM**). Finally, in both subtasks, when the models ran on both Word-level and DA-Level (**Full LSTM**), they obtained the highest performance.

Model	Prec	Rec	F1
CP + CEI$_{causal}$	**0.864**	**0.877**	**0.868**
CP + CEI$_{all}$	0.842	0.864	0.848
CEI$_{causal}$ Only	0.847	0.788	0.810
CEI$_{all}$ Only	0.836	0.848	0.842

Table 7: The effect of Linear SVM Cauality model (CP) within our pipeline. CEI$_{all}$: LSTM CEI models trained on all messages; CEI$_{causal}$: LSTM CEI models trained only on causality messages (CEI$_{causal}$); CP + CEI$_{all|causal}$: the combination of Linear SVM and each LSTM model. Bold: significant ($p < .05$) increase in F1 over the next best model, suggesting the two-step approach worked best.

4.4 Complete Pipeline

Evaluations thus far zeroed-in on each subtask of causal explanation analysis (i.e. CEI only focused on data already identified to contain causal explanations). Here, we seek to evaluate the complete pipeline of CP and CEI, starting from all of test data (those or without causality) and evaluating the final accuracy of CEI predictions. This is intended to evaluate CEI performance under an applied setting where one does not already know whether a document has a causal explanation.

There are several approaches we could take to perform CEI starting from unannotated data. We could simply run CEI prediction by itself (**CEI Only**) or the pipeline of CP first and then only run CEI on documents predicted as causal (**CP + CEI**). Further, the CEI model could be trained only on those documents annotated causal (as was done in the previous experiments) or on all training documents including many that are not causal.

Table 7 show results varying the pipeline and how CEI was trained. Though all setups performed decent ($F1 > 0.81$) we see that the pipelined approach, first predicting causality (with the linear SVM) and then predicting causal explanations only for those with marked causal (CP + CEI$_{causal}$) yielded the strongest results. This also utilized the CEI model only trained on those annotated causal. Besides performance, an added benefit from this two step approach is that the CP step is less computational intensive of the CEI step and approximately 2/3 of documents will never need the CEI step applied.

Limitations. We had an inevitable limitation on the size of our dataset, since there is no other causality dataset over social media and the annotation required an intensive iterative process. This might have limited performances of more complex models, but considering the processing time and the computation load, the combination of the linear model and the RNN-based model of our pipeline obtained both the high performance and efficiency for the practical applications to downstream tasks. In other words, it's possible the linear model will not perform as well if the training size is increased substantially. However, a linear model could still be used to do a first-pass, computationally efficient labeling, in order to shortlist social media posts for further labeling from an LSTM or more complex model.

5 Exploration

Here, we explore the use of causal explanation analysis for downstream tasks. First we look at the relationship between use of causal explanation and one's demographics: age and gender. Then, we consider their use in sentiment analysis for extracting the causes of polarity ratings. Research involving human subjects was approved by the University of Pennsylvania Institutional Review Board.

Demographic differences. We first explored variance in number of causality posts by demographics. To do this, we used self-authored posts from a random 300 consenting-users of the MyPersonality dataset (Kosinski et al., 2013). For each user we calculate a cp_ratio, defined as the number of causality predicted posts divided by their total number of posts, indicating the percentage of their posts which include a causal explanation. We then correlated this ratio with real-valued age using Pearson correlation and looked the differences by dichotomous gender using Cohen's d (the difference in standardized means; only binary gender was available). We found significant ($p < .05$) moderate-sized associations for both, indicating both older individuals and females were likely to use more causal explanations.

	CE Top Ngrams	**Non-CE** Top Ngrams
1	worst	not
2	was	no
3	not	”
4	the worst	asked
5	horrible	she
6	rude	told
7	bad	said
8	overpriced	minutes
9	over	?
10	slow	me

Table 8: Top words most associated with negative reviews from within causal explanations (CE) and outside of causal explanation (Non-CE).

Causality in Sentiment Analysis We explored the application of causality explanation identification for sentiment analysis using the Yelp polarity dataset (Zhang et al., 2015). We randomly selected 10,000 of both positive and negative reviews and ran our complete pipeline on them to extract the causal explanations from the reviews. We then analyzed the ngrams from (a) causal explanation and (b) all other discourse arguments testing for associations with polarity. We used the a Bayesian interpretation of the log odds ratio using an informative dirichlet prior defined by Monroe et al. (2008). We found difference in the top ngrams depending on whether the argument the ngram originated from was a causal explanation or not (see Table 8). Top ngrams for causal explanations included more content words (e.g. 'rude', 'overpriced', 'slow') suggesting analyzing causal explanations within reviews can better target the *reasons* for the negative review.

6 Conclusion

We developed a pipeline for causal explanation analysis over social media text, including both causality prediction and causal explanation identification. We examined a variety of model types and RNN architectures for each part of the pipeline, finding an SVM best for causality prediction and a hierarchy of BiLSTMs for causal explanation identification, suggesting the later task relies more heavily on sequential information. In fact, we found replacing either layer of the hierarchical LSTM architecture (the word-level or the DA-level) with a an equivalent "bag of features" approach resulted in reduced accuracy. Results of our whole pipeline of causal explanation analysis were found quite strong, achieving an $F1 = 0.868$ at identifying discourse arguments that are causal explanations.

Finally, we demonstrated use of our models in applications, finding associations between demographics and rate of mentioning causal explanations, as well as showing differences in the top words predictive of negative ratings in Yelp reviews. Utilization of discourse structure in social media analysis has been a largely untapped area of exploration, perhaps due to its perceived difficulty. We hope the strong results of causal explanation identification here leads to the integration of more syntax and deeper semantics into social media analyses and ultimately enables new applications beyond the current state of the art.

Acknowledgments

This work was supported, in part, by a grant from the Templeton Religion Trust (ID #TRT0048). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. We also thank Laura Smith, Yiyi Chen, Greta Jawel and Vanessa Hernandez for their work in identifying causal explanations.

References

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. *arXiv preprint arXiv:1509.01599*.

Or Biran and Kathleen McKeown. 2015. Pdtb discourse parsing as a tagging task: The two taggers approach. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 96–104.

Charles S Carver and Joan Gollin Gaines. 1987. Optimism, pessimism, and postpartum depression. *Cognitive therapy and Research*, 11(4):449–462.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 511–521.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *ACL (1)*, pages 13–24.

Yangfeng Ji and Noah Smith. 2017. Neural discourse structure for text categorization. *arXiv preprint arXiv:1702.01829*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2061–2069.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(02):151–184.

Veronica Lynn, Youngseo Son, Vivek Kulkarni, Niranjan Balasubramanian, and H Andrew Schwartz. 2017. Human centered nlp with user-factor adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1146–1155.

William C Mann and Sandra A Thompson. 1987. *Rhetorical structure theory: A theory of text organization*. University of Southern California, Information Sciences Institute.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 108–112. Association for Computational Linguistics.

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In *Advances in Neural Information Processing Systems*, pages 2888–2896.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Christopher Peterson, Martin E Seligman, and George E Vaillant. 1988. Pessimistic explanatory style is a risk factor for physical illness: a thirty-five-year longitudinal study. *Journal of personality and social psychology*, 55(1):23.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 683–691. Association for Computational Linguistics.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 13–16. Association for Computational Linguistics.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie L Webber. 2007. The penn discourse treebank 2.0 annotation manual.

Michael F Scheier, Karen A Matthews, Jane F Owens, George J Magovern, R Craig Lefebvre, R Anne Abbott, and Charles S Carver. 1989. Dispositional optimism and recovery from coronary artery bypass surgery: the beneficial effects on physical and psychological well-being. *Journal of personality and social psychology*, 57(6):1024.

Youngseo Son, Anneke Buffone, Joe Raso, Allegra Larche, Anthony Janocko, Kevin Zembroski, H Andrew Schwartz, and Lyle Ungar. 2017. Recognizing counterfactual thinking in social media texts. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 654–658.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1507–1514. Association for Computational Linguistics.

LRMM: Learning to Recommend with Missing Modalities

Cheng Wang
NEC Laboratories Europe

Mathias Niepert
NEC Laboratories Europe

{cheng.wang, mathias.niepert}@neclab.eu

Hui Li[*]
The University of Hong Kong
hli2@cs.hku.hk

Abstract

Multimodal learning has shown promising performance in content-based recommendation due to the auxiliary user and item information of multiple modalities such as text and images. However, the problem of incomplete and missing modality is rarely explored and most existing methods fail in learning a recommendation model with missing or corrupted modalities. In this paper, we propose LRMM, a novel framework that mitigates not only the problem of missing modalities but also more generally the cold-start problem of recommender systems. We propose modality dropout (*m*-drop) and a multimodal sequential autoencoder (*m*-auto) to learn multimodal representations for complementing and imputing missing modalities. Extensive experiments on real-world Amazon data show that LRMM achieves state-of-the-art performance on rating prediction tasks. More importantly, LRMM is more robust to previous methods in alleviating data-sparsity and the cold-start problem.

1 Introduction

Recommender systems (RS) are useful filtering tools which aid customers in a personalized way to make better purchasing decisions and whose recommendations are based on the customer's preferences and purchasing histories. Recommender systems can be roughly divided into collaborative filtering (CF) (Koren et al., 2009) or content-based filtering (CBF) (Pazzani and Billsus, 2007) methods. CF-based methods predict the product preference of users based on their previous purchasing and reviewing behavior by computing latent representations of users and products. Standard matrix factorization (MF) and its variants are widely used in CF approaches (Koren et al., 2009). While CF-based approaches were demonstrated to perform well in many application domains (Ricci et al., 2015), these methods are based solely on the sparse user-item rating matrix and, therefore, suffer from the so-called *cold-start* problem (Schein et al., 2002; Huang and Lin, 2016; Wang et al., 2017) as shown in Figure 1(b)+(c). For new users without a rating history and newly added products with few or no ratings, the systems fail to generate high-quality personalized recommendations.

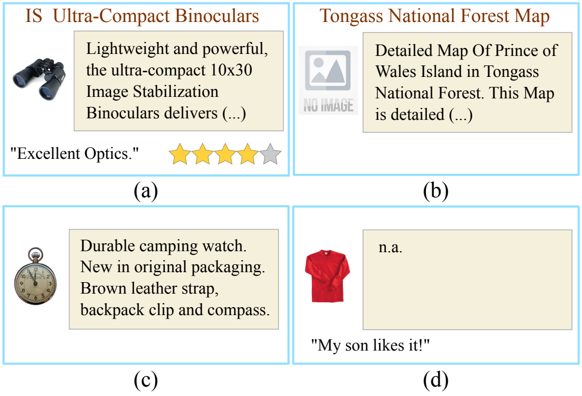

(a) (b) (c) (d)

Figure 1: Examples of typical multimodal product data from online retailers: image, title, description, reviews, star ratings. The cold-start problem is present in cases (b) and (c) where neither review text nor ratings are available.

Alternatively, CBF approaches incorporate auxiliary modalities/information such as product descriptions, images, and user reviews to alleviate the cold-start problem by leveraging the correlations between multiple data modalities. Unfortunately, a pure CBF method often suffers difficulties in generating a recommendation on incomplete and missing data (Sedhain et al., 2015; Wang et al., 2016b; Volkovs et al., 2017; García-Durán et al., 2018).

In this work, a multimodal imputation framework (LRMM) is proposed to make RS robust to incomplete and missing modalities. First, LRMM learns multimodal correlations (Ngiam et al., 2011; Srivastava and Salakhutdinov, 2012;

---

[*] Work done in NEC Laboratories Europe.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3360–3370
Brussels, Belgium, October 31 - November 4, 2018.

Wang et al., 2016a, 2018) from product images, product metadata (title+description), and product reviews. We propose modality dropout (*m*-drop) which randomly drops representations of some data modalities. In combination with the modality dropout approach, a sequential autoencoder (*m*-auto) for multi-modal data is trained to reconstruct missing modalities and, at test time, is used to impute missing modalities through its learned reconstruction function.

Multimodal imputation for recommender systems is a non-trivial issue. (1) Existing RS methods usually assume that all data modalities are available during training and inference. In practice, however, incomplete and missing data modalities are very common. (2) At its core it addresses the cold-start problem. In the context of missing modalities, cold-start can be viewed as missing user or item preference information.

With this paper we make the following contributions:

- For the first time, we introduce multimodal imputation in the context of recommender systems.
- We reformulate the data-sparsity and cold-start problem when data modalities are missing.
- We show that the proposed method achieves state-of-the-art results and is competitive with or outperforms existing methods on multiple data sets.
- We conduct additional extensive experiments to empirically verify that our approach alleviates the missing data modalities problem.

The rest of paper is structured as follows: Section 2 introduces our proposed methods. Section 3 describes the experiments and reports on the empirical results. In section 4 we discuss the method and its advantages and disadvantages, and in section 5 we discuss related work. Section 6 concludes this work.

2 Proposed Methods

The general framework of LRMM is depicted in Figure 2. There are two objectives for LRMM: (1) learning multimodal embeddings that capture inter-modal correlations, complementing missing modalities (Sec. 2.1); (2) learning intra-modal distributions where missing modalities are reconstructed via a missing modality imputation mechanism (sec. 2.2 and 2.3).

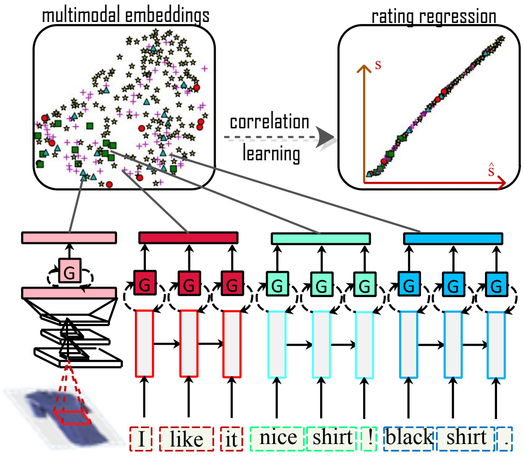

Figure 2: Overview of LRMM. It adopts CNN for visual embeddings (pink part) and three LSTMs for textual embeddings of user review text (red part), item review text (green part) and item metadata (blue part), respectively. The generative (autoencoder) model is used to reconstruct modality-specific embeddings and impute missing modality. Missing user and item review text lead to user- and item-based cold-start respectively.

2.1 Learning Multimodal Embeddings

We denote a user $\mathbf{u}$ having k review texts as $\mathbf{r}^u{=}(r^u_{o_1}, r^u_{o_2}, ..., r^u_{o_k})$ where $r^u_{o_i}$ represents review text written by $\mathbf{u}$ for item o_i. An item $\mathbf{o}$ is denoted as $\mathbf{r}^o{=}(r^o_{u_1}, r^o_{u_2}, ..., r^o_{u_p})$ where $r^o_{u_j}$ represents the review text written by user $\mathbf{u}_j$ for item $\mathbf{o}$. Following Zheng et al. (2017), to represent each user and item, the reviews of $\mathbf{u}$ and $\mathbf{o}$ are concatenated into one review history document:

$$\mathbf{D}^u = r^u_{o_1} \oplus r^u_{o_2} \oplus, ..., \oplus r^u_{o_k} \tag{1}$$

$$\mathbf{D}^o = r^o_{u_1} \oplus r^o_{u_2} \oplus, ..., \oplus r^o_{u_p} \tag{2}$$

where $\oplus$ is the concatenation operator. Similarly, the metadata of each item $\mathbf{o}$ can be represented as $\mathbf{D}^m$. For readability, we use u, o, m, v to denote user, item, metadata, and the image modality, respectively.

For text-based representation learning for user and item, unlike Zheng et al. (2017) in which CNNs (Convolutional Neural Networks) with Word2Vec (Mikolov et al., 2013) are employed, our method treats text as sequential data and learns embeddings over word sequences by maximizing the following probabilities:

$$p(\mathbf{x}_1^g, ..., \mathbf{x}_T^g) = \prod\nolimits_{t=1}^{T^g} p(\mathbf{x}_t^g | \mathbf{x}_1^g, ..., \mathbf{x}_{t-1}^g) \quad (3)$$

$$p(\mathbf{x}_t^g | \mathbf{x}_1^g, ..., \mathbf{x}_{t-1}^g) = p(\mathbf{x}_t^g | \mathbf{e}_t^g) \quad (4)$$

$$\mathbf{e}_t^g = \mathcal{M}^g(\mathbf{e}_{t-1}^g, \mathbf{x}_t^g; \mathbf{\Theta}^g) \quad (5)$$

where $\mathcal{M}^g$, $g \in \{u, o, m\}$ is a recurrent model and $(\mathbf{x}_1^g, ..., \mathbf{x}_T^g)$ is the word sequence of either review or metadata text, each $\mathbf{x}_t^g \in \mathcal{V}$ and $\mathcal{V}$ is a vocabulary set. T^g is the length of input and output sequence and $\mathbf{e}_t^g$ is the hidden state computed from the corresponding LSTM (Long Short Term Memory) (Hochreiter and Schmidhuber, 1997) by:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{pmatrix} \quad (6)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \quad (7)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (8)$$

where $\mathbf{i}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$ are input, forget and output gate respectively, $\mathbf{c}_t$ is memory cell, $\mathbf{h}_t$ is the hidden output that we used for computing user or item embedding $\mathbf{e}^g$, $g \in \{u, o, m\}$.

As we treat each text document $\mathbf{D}^g$ as a word sequence of length T^g, we adopt average pooling on word embeddings for each modality to obtain document-level representations:

$$\mathbf{e}^g = \frac{\sum_{t \in T^g,\, g \in \{u,o,m\}} \mathbf{e}_t^g}{T^g} \quad (9)$$

Visual embeddings $\mathbf{e}^v$ are extracted with a pretrained CNN and transformed by a function f

$$\mathbf{e}^v = f(\text{CNN}(I, \mathbf{\Theta}_c); \mathbf{\Theta}_f), \quad (10)$$

where $\mathbf{\Theta}_f \in \mathbb{R}^{4096 \times d}$ to ensure $\mathbf{e}^v$ has same dimension as the user $\mathbf{e}^u$, item $\mathbf{e}^o$, and metadata embedding $\mathbf{e}^m$. The multimodal joint embedding then can be learned by a shared layer and used for making a prediction:

$$\hat{\mathbf{s}} = f_s(\mathbf{W}_s(\mathbf{e}^u \oplus \mathbf{e}^o \oplus \mathbf{e}^m \oplus \mathbf{e}^v) + \mathbf{b}_s) \quad (11)$$

where $f_s : \mathbb{R}^{4 \times d} \to \mathbb{R}^1$, parameterized with $\mathbf{W}_s$ and $\mathbf{b}_s$, is a scoring function to map the multimodal joint embedding to a rating score.

2.2 Modality Dropout

Modality dropout (*m*-drop) is designed to remove a data modality during training according to some parametric distribution. This is motivated by dropout (Srivastava et al., 2014) which randomly masks hidden layer activations to zero to increase the generalization capability of the underlying model. More formally, *m*-drop changes the original feed-forward equation:

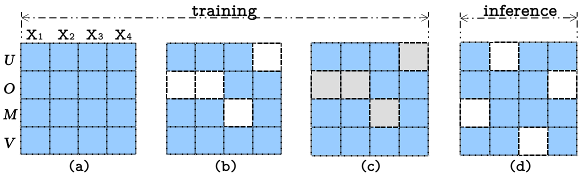

Figure 3: Missing modality imputation. (a) Full training data, (b) *m*-drop randomly drops modalities, (c) *m*-auto learns to reconstruct missing data based on existing data. (d) Inference with missing modalities. Dropping user and item view is equivalent to learning models being able to address cold-start problem.

$$\mathbf{Z}^{(L+1)} = \varphi(\mathbf{W}^{(L+1))}\mathbf{X}^{(L)} + \mathbf{b}^{(L+1)}) \quad (12)$$

being able to randomly drop modality by:

$$r^{(L)} \sim \text{Bernoulli}\ (p_m) \quad (13)$$

$$k^{(L)} \sim \text{Bernoulli}\ (1 - 1/n_m) \quad (14)$$

$$\widetilde{\mathbf{X}}^{(L)} = (\mathbf{X}^{(L)} \odot \mathbf{r}^{(L)}) \odot \mathbf{k}^{(L)} \quad (15)$$

$$\mathbf{Z}^{(L+1)} = \varphi(\mathbf{W}^{(L+1))}\widetilde{\mathbf{X}}^{(L)} + \mathbf{b}^{(L+1)}) \quad (16)$$

where each sample $\mathbf{X}_1 = x_1, ..., x_{n_m}$ and n_m is the number of modalities. $\mathbf{r}^{(L)}$ is a vector of independent Bernoulli random variables each of which has probability p_m of being 1. $\mathbf{k}^{(L)}$ is a vector of independent variables which indicate the dropout on modality with a given probability. $\varphi(\cdot)$ is an activation function.

Figure 3 (a-b) shows how *m*-drop works. Note the differences between modality dropout (*m*-drop) and original dropout: (1) *m*-drop targets specifically the multimodal scenario where some modalities are completely missing; and (2) *m*-drop is performed on the input layer ($L \equiv 0$).

2.3 Mutlimodal Sequential Autoencoder

The autoencoder has been used in prior work (Sedhain et al., 2015; Strub et al., 2016) to reconstruct missing elements (mostly ratings) in recommender systems. This is equivalent to the case of missing at random (MAR). For MAR, it is rare to have a continuous large block of missing entries (Tran et al., 2017). Differently, in recommending with missing modality, the missing entries typically occur in a *large continuous block*. For instance, an extreme case is the absence of all item reviews and ratings (data sparsity is 100%, leading to the so-called item cold-start problem). Existing methods (Lee and Seung, 2000; Koren, 2008; Marlin, 2003; Wang and Blei, 2011; McAuley and Leskovec, 2013; Li et al., 2017; Zheng et al., 2017) have difficulties when entire data modalities are missing during the *training* and/or *inference* stages.

To address this limitation, we propose a multimodal sequence autoencoder (*m*-auto) to impute textual sequential embeddings and visual embeddings for the missing modalities. Modality-specific autoencoders are placed between the modality-specific encoders (i.e., CNN and LSTMs) and the shared layer (equation 11). The reconstruction layers, therefore, can capture the inter-modal and intra-modal correlations. More formally, for each data modality $g \in \{u, o, m, v\}$, the modality-specific encoder is given as

$$\mathbf{e}^g_{hid} = \text{sigm}(\mathbf{W}^g_{vh}\mathbf{e}^g_{in} + \mathbf{b}^g_{vh}) \tag{17}$$

and the modality-decoder is given as

$$\mathbf{e}^g_{recon} = \frac{1}{T^g}\sum_{t \in T^g} \text{sigm}(\mathbf{W}^g_{hv}\mathbf{e}^g_{hid} + \mathbf{b}^g_{hv}) \tag{18}$$

where $\mathbf{W}_{vh} \in \mathbb{R}^{d \times d_h}$ and $\mathbf{W}_{hv} \in \mathbb{R}^{d_h \times d}$ are weights, $\mathbf{b}_{vh}$, $\mathbf{b}_{hv}$ are biases receptively for visible-to-hidden, and hidden-to-visible layers. $\mathbf{e}^g_{in}$, $\mathbf{e}^g_{hid}$ present the original, hidden word-level embeddings, and $\mathbf{e}^g_{recon}$ is the reconstructed document-level embeddings. The $\mathbf{e}^g$ is a modality-specific embedding.

m-auto is different from previous reconstruction models(Sedhain et al., 2015; Strub et al., 2016) in that its reconstructions are based on inter-modal and intra-modal correlations in the context of multimodal learning.

2.4 Model Optimization

The optimization of the network is formulated as a regression problem by minimizing the mean squared error (MSE) loss $\mathcal{L}_{reg}$:

$$\mathcal{L}_{reg} = \frac{1}{|\mathcal{D}|}\sum_{(u,o,m,v) \in \mathcal{D}} (\hat{\mathbf{s}} - \mathbf{s})^2 + \lambda \parallel \mathbf{\Theta}_r \parallel_2 \tag{19}$$

where $\hat{\mathbf{s}}$ and $\mathbf{s}$ are the predicted and truth rating scores. $|\mathcal{D}|$ is dataset size , λ is weight decay parameter and $\mathbf{\Theta}_r$ is regression model parameters. To constrain the representations to be compact in reconstruction, a penalty term is utilized

$$\mathcal{H} = \sum_{i=1}^{h_n} \log \frac{\rho}{\hat{\rho}_i} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_i} \tag{20}$$

where ρ and $\hat{\rho}$ are sparsity parameters and average activation of hidden unit i, h_n is the number of hidden units. The reconstruction loss for each modality is now

$$\begin{aligned}\mathcal{L}^g_{recon} =& \frac{1}{|\mathcal{D}|}\sum_{g \in \{u,o,m,v\}} \|\mathbf{e}^g_{recon} - \mathbf{e}^g_{in}\|^2 \\ &+ \lambda_\rho \sum_{g \in \{u,o,m,v\}} \mathcal{H}^g\end{aligned} \tag{21}$$

where λ_ρ is a sparsity regularization term. The objective of the entire model is then

$$\mathcal{L} = \alpha\mathcal{L}_{reg} + \beta \sum_{g \in \{u,o,m,v\}} \mathcal{L}^g_{recon} \tag{22}$$

where α and β are learnable parameters. The model is learned in an end-to-end fashion through back-prorogation (LeCun et al., 1989).

3 Experiments

This section evaluates LRMM on rating prediction tasks with real-world datasets. We firstly compare LRMM with recent methods (sec. 3.4), then we empirically show the effectiveness of LRMM in alleviating the cold-start, the incomplete/missing data, and the data sparsity problem (sec. 3.5-3.8).

3.1 Datasets and Evaluation Metrics

We conducted experiments on the Amazon dataset (McAuley et al., 2015; He and McAuley, 2016)[1], which is widely used for the study of recommender systems. It consists of different modalities such as text, image, and numerical data. We used 4 out of 21 categories: Sports and Outdoors (S&O), Health and Personal Care (H&P), Movies and TV, Electronics. Some statistics of the datasets are listed in Table 1. We randomly split each dataset into 80% training, 10% validation, and 10% test data. Each input instance consists of four parts $\mathbf{x}^{(i)} = (\mathbf{x}^{(i)}_u, \mathbf{x}^{(i)}_o, \mathbf{x}^{(i)}_m, \mathbf{x}^{(i)}_v)$, where $\mathbf{x}^{(i)}_u$ and $\mathbf{x}^{(i)}_i$ are the concatenated reviews of users and items in the training data. $\mathcal{V}$ is the vocabulary that was built based on reviews and metadata on the training data. Words with an absolute frequency of at least 20 are included in the vocabulary.

---

[1] `http://jmcauley.ucsd.edu/data/amazon/`

Table 1: Datasets

Dataset	S&O	H&P	Movie	Electronics
Users	35494	38599	111149	192220
Items	16415	17909	27019	59782
Samples	272453	336769	974582	1614105
$\lvert\mathcal{V}\rvert$	42095	47476	160117	198598

To evaluate the proposed models on the task of rating prediction, we employed two metrics, namely, Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{|\mathcal{D}|} \sum_{(u,o,m,v)\in\mathcal{D}} (\hat{\mathbf{s}}_{i,j} - \mathbf{s}_{i,j})^2} \quad (23)$$

and Mean Absolute Error (MAE)

$$MAE = \frac{1}{|\mathcal{D}|} \sum_{(u,o,m,v)\in\mathcal{D}} |\hat{\mathbf{s}}_{i,j} - \mathbf{s}_{i,j}| \quad (24)$$

where $\hat{\mathbf{s}}_{i,j}$ and $\mathbf{s}_{i,j}$ represent the predicted rating score and ground truth rating score that user i gave to item j.

3.2 Baselines and Competing Methods

We compare our models with several baselines[2]. The baselines can be categorized into three groups.

1. Matrix factorization: *NMF* (Lee and Seung, 2000) and *SVD++* (Koren, 2008).
2. Topic model methods: *URP* (Marlin, 2003), *CTR* (Wang and Blei, 2011), *HFT* (McAuley and Leskovec, 2013) and *RMR* (Ling et al., 2014).
3. Deep learning models: *NRT* (Li et al., 2017) and *DeepCoNN* (Zheng et al., 2017), which are current state-of-the-art approaches.

We also include a naive method—*Offset* (McAuley and Leskovec, 2013) which simply takes the average across all training ratings.

3.3 Implementation

We implemented LRMM with Theano[3]. The weights for the non-recurrent layer were initialized by drawing from the interval $\left[-\sqrt{\frac{6}{N_{in}+N_{out}}}, \sqrt{\frac{6}{N_{in}+N_{out}}}\right]$ (N is the number of units) uniformly at random. We used 1024 hidden units for the autoencoder. The LSTMs have 256 hidden units and the internal weights $\mathbf{W}$ are orthogonally initialized (Saxe et al., 2014). We used a batch size of 256, $\lambda = 0.0001$, sparsity parameter $\rho = 0.05$, $\lambda_\rho = 0.01$, an initial learning rate of 0.0001 and a dropout rate of 0.5 after the recurrent layer. The models were optimized with ADADELTA (Zeiler, 2012). The length of the user, item and meta-data document $\mathbf{D}^u$, $\mathbf{D}^o$, and $\mathbf{D}^m_o$ were fixed to $L = 100$. We truncated documents with more than 100 words. The image features are extracted from the first fully-connected layer of CNN on ImageNet (Russakovsky et al., 2015).

We implemented NMF and SVD++ with the SurPrise package[4]. Offset and HFT were implemented by modifying authors' implementation[5]. For DeepCoNN, we adapted the implementation from (Chen et al., 2018)[6]. The numbers of other methods are taken from Li et al. (2017).

3.4 Compare with State-of-the-art

First, we compare LRMM with state-of-the-art methods listed in Sec. 3.2. In this setting, LRMM is trained with all data modalities and tested with different missing modality regimes. Table 2 lists the results on the four datasets. By leveraging multimodal correlations, LRMM significantly outperforms MF-based models (i.e. NMF, SVD++) and topic-based methods (i.e., URP, CTR, RMR, and HFT). LRMM also outperforms recent deep learning models (i.e., NRT, DeepCoNN) with respect to almost all metrics.

LRMM is the only method with a robust performance for the cold-start recommendation problem where user review or item review texts are removed. While the cold-start recommendation is more challenging, LRMM(-U) and LRMM(-O) are still able to achieve a similar performance to the baselines in the standard recommendation setting. For example, RMSE 1.101 (LRMM(-O)) to 1.107 (NRT) on Electronics, MAE 0.680 (LRMM(-O)) to 0.667 (DeepCoNN)on S&O. We conjecture that the cross-modality dependen-

---

[2] To make a fair comparison, implemented baselines are trained with grid search (for NMF and SVD++, regularization [0.0001, 0.0005, 0.001], learning rate [0.0005, 0.001, 0.005, 0.01]. For HFT, regularization [0.0001, 0.001, 0.01, 0.1, 1], lambda [0.1, 0.25, 0.5, 1]). For DeepCoNN, we use the suggested default parameters. The best scores are reported.

[3] `http://www.deeplearning.net/software/theano/`

[4] `http://surpriselib.com/`

[5] `http://cseweb.ucsd.edu/~jmcauley/code/code_RecSys13.tar.gz`

[6] `https://github.com/chenchongthu/DeepCoNN`

Table 2: Comparison on datasets with the baselines. '+F': tested with all modalities(U,O,M,V), '-X': dropping one modality, '-U' and '-O': user and item cold-start scenario.

Dataset	S&O		H&P		Movie		Electronics	
Models	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Offset	0.979	0.769	1.247	0.882	1.389	0.933	1.401	0.928
NMF	0.948	0.671	1.059	0.761	1.135	0.794	1.297	0.904
SVD++	0.922	0.669	1.026	0.760	1.049	0.745	1.194	0.847
URP	-	-	-	-	1.006	0.764	1.126	0.860
RMR	-	-	-	-	1.005	0.741	1.123	0.822
HFT	0.924	0.659	1.040	0.757	0.997	0.735	1.110	0.807
DeepCoNN	0.943	0.667	1.045	0.746	1.014	0.743	1.109	0.797
NRT	-	-	-	-	0.985	**0.702**	1.107	0.806
LRMM(+F)	**0.886**	**0.624**	**0.988**	**0.708**	**0.983**	0.716	**1.052**	**0.766**
LRMM(-U)	0.936	0.719	1.058	0.782	1.086	0.821	1.138	0.900
LRMM(-O)	0.931	0.680	1.039	0.805	1.074	0.855	1.101	0.864
LRMM(-M)	0.887	0.625	0.989	0.710	0.991	0.725	1.053	0.766
LRMM(-V)	0.886	0.624	0.989	0.708	0.991	0.725	1.052	0.766

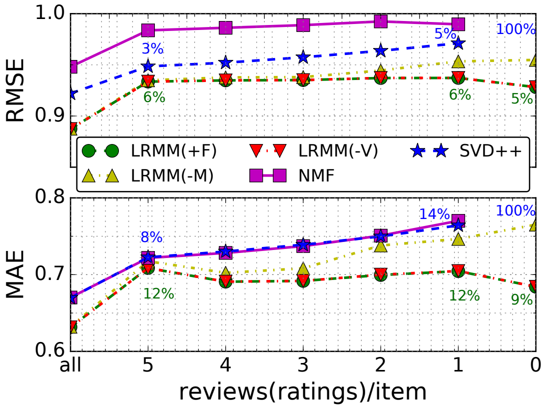

Figure 4: Performance with reduced reviews (ratings) on S&O dataset. (% : performance drops for SVD++ (in blue) and LRMM(+F) (in green))

cies (Srivastava and Salakhutdinov, 2012) make LRMM more robust when modalities are missing. Table 5 lists some randomly selected rating predictions. Similar to Table 2, missing user (-U) and item (-O) preference significantly deteriorates the performance.

3.5 Cold-Start Recommendation

Prior work (McAuley and Leskovec, 2013; Zhang et al., 2017) has considered users (items) with sparse preference information as the cold-start problem (e.g., Figure 1(d)), that is, where there is still some information available. In practice, preference information could be missing in larger quantities or even be entirely absent (e.g., Figure 1(b-c)). In this situation, the aforementioned methods are not applicable as they require some data to work with. In this experiment, we examine how LRMM leverages modality correlations to alleviate the data sparsity problem when training data becomes even sparser. To this end, we train models for the item cold-start problem by reducing the number of reviews (for LRMM) and ratings (for NMF and SVD++) of each item in the training set.

Figure 4 demonstrates the robustness of LRMM when the training data becomes more sparse. Note that NMF and SVD++ fail to train models when there is no ratings data available. In contrast, LRMM is trained by leveraging item images and metadata even if item reviews are completely missing for a product. The average number of reviews per item on this dataset is 16.7. Reducing the number of ratings to 5 severely degrades the performance of NMF, SVD++, and LRMM. However, LRMM remains rather stable in maintaining good performance when considering the performance degradation at 5, 1, and 0 reviews (ratings), respectively. One interesting observation is that, with a reduced number of reviews, the product metadata plays a more and more important role in maintaining the performance: LRMM(-V) is close to LRMM(+F) in Figure 4 while the gap between LRMM(-M) and LRMM(+F) is large.

3.6 Missing Modality Imputation

The proposed *m*-drop and *m*-auto methods allow LRMM to be more robust to missing data modalities. Table 3 lists the results of train-

Table 3: The performance of training with missing modality imputation.

Dataset	S&O		H&P	
Models	RMSE	MAE	RMSE	MAE
LRMM(+F)	0.997	0.790	1.131	0.912
LRMM(-U)	0.998	0.795	1.132	0.914
LRMM(-O)	0.999	0.796	1.133	0.917
LRMM(-M)	0.998	0.797	1.133	0.913
LRMM(-V)	0.997	0.791	1.132	0.913

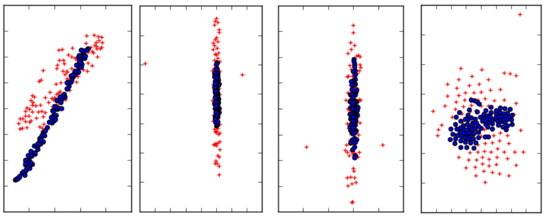

(a) $\mathbf{e}^v$ (b) $\mathbf{e}^u$ (c) $\mathbf{e}^o$ (d) $\mathbf{e}^m$

Figure 5: Visualization of embeddings (blue) and reconstructed embeddings (red)

ing LRMM with missing data modalities for the modality dropout ratio $p_m = 0.5$ on the S&O and H&P datasets, respectively. Both RMSE and MAE of LRMM deteriorate but are still comparable to the MF-based approaches NMF and SVD++. However, the proposed method LRMM is robust to missing data in both training and inference stages, a problem rarely addressed by existing approaches. In Figure 5, we visualized the modality-specific embeddings and their reconstructed embeddings of 100 randomly selected samples with t-SNE (van der Maaten and Hinton, 2008). The plots suggest that it is more challenging to reconstruct item metadata and image embeddings as compared to the user or item embeddings. One possible explanation is that some selected metadata contains noisy data (e.g., "*ISBN - 9780963235985*", "*size: 24* ×*46*" and "*Dimensions: 15W* × *22H*") for which visual data is more diverse. This would increase the difficulty of incorporating visual data into the embeddings.

3.7 The Effect of Text Length

To alleviate the data sparsity problem, existing work (McAuley and Leskovec, 2013; Zhang et al., 2017) concatenates review texts and utilizes topic modeling (e.g. HFT) or CNNs combined with Word2Vec (e.g. DeepCoNN) to learn user or item embeddings. Differently, LRMM treats the concatenated reviews as sequential data and learns sequence embeddings with RNNs. In this experiment, we show that learning sequential embedding is beneficial on sparse data because it is unnecessary to exploit all reviews so as to reach good performance. Figure 6 shows the performance of LRMM with varied word sequence lengths. In general, sequence embeddings learned with larger length achieve better performance. Note that, by considering a certain amount of words (e.g. L=50), LRMM is able to achieve a result as good as accounting more words (e.g. L=100 or 200). Although this is dataset-dependent to some degree, e.g., LRMM (L=200) improves RMSE and MAE in a certain margin as compared to L=100 on the H&P data, it demonstrates the superiority of sequential user or item embeddings as compared to topic and CNN+Word2Vec embeddings on more sparse data as shown in Table 2.

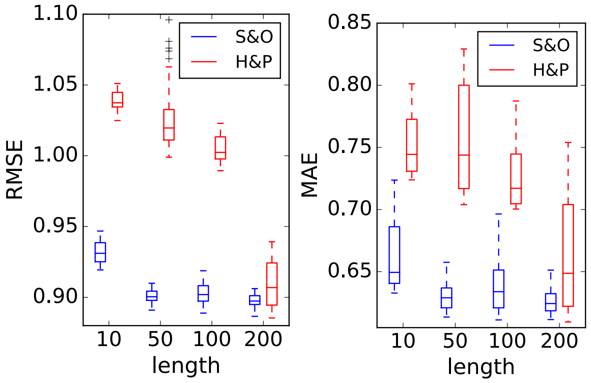

Figure 6: RMSE and MAE with varied text lengths on S&O and H&P datasets.

3.8 Cross-Domain Adaptation

To consider an even more challenging situation we explore cases where the full training set is missing. Inspired by the recent success of domain adaptation (DA) (Csurka, 2017), a special form of transfer learning (Pan and Yang, 2010; Weiss et al., 2016), we perform the recommendation task on the target domain test set $\mathcal{D}^t_{test}$ (e.g., "Sport") but with the model $\mathcal{C}$ trained on a different domain training set $\mathcal{D}^s_{train}$ (e.g. "Movie"). This is achieved by extracting the multimodal embeddings on the source domain and by performing prediction on the target domain. Table 4 shows the performance of LRMM when performing adaptation from larger datasets to smaller datasets. Although the performance is not as good as on $\mathcal{D}^s_{test}$, LRMM is still able to obtain decent results even

Table 4: Cross-Domain Adaptation with LRMM

$\mathcal{D}^s \rightarrow \mathcal{D}^t$	**+F**	**-U**	**-O**	**-M**	**-V**
Movie→S&O	1.061	1.013	1.071	1.061	1.062
Movie→H&P	1.190	1.140	1.170	1.190	1.190
Elect.→S&O	1.072	1.012	1.088	1.073	1.073
Elect.→H&P	1.191	1.137	1.180	1.191	1.192

Table 5: Exemplary rating prediction on S&O datatset. 'T' means true ratings, the best prediction is in blue, the worst prediction is in red.

Item image	T	+F	-U	-O	-M	-V
	3	3.18	**3.97**	3.48	**3.03**	3.28
	3	3.36	**4.07**	3.5	3.33	**3.27**
	5	**4.63**	4.50	**4.36**	4.60	**4.63**
	3	**3.11**	**3.77**	3.49	3.44	3.57
	4	**4.00**	**4.31**	3.87	3.92	4.02

Table 6: Examples on H&P datatset with domain adaptation. The model is trained on Movie dataset.

Item image	T	+F	-U	-O	-M	-V
	4	**4.01**	4.18	**3.70**	4.04	4.05
	2	**2.58**	**3.85**	2.82	2.82	2.76
	3	2.97	**4.33**	2.77	**2.99**	2.96
	4	4.09	**3.68**	4.14	4.09	**4.04**
	5	**4.99**	**4.39**	4.47	4.94	**5.01**

without using training data $\mathcal{D}^t_{train}$. Table 6 shows some example rating predictions on DA for different categories of products. It demonstrates the strong generalization capability of DA from one product category to another.

4 Discussion

Empirically, we have shown that multimodal learning (**+F**) plays an important role in mitigating the problems associated with missing data/modality and, in particular, those associated with the cold-start problem (**-U** and **-O**) of recommender systems. The proposed method LRMM is in line and grounded in recent developments (e.g. DeepCoNN, NRT) to incorporate multimodal data. LRMM distinguishes itself from previous methods: (1) the cold-start problem is reformulated in the context of missing modality; (2) A novel multimodal imputation method which consists of *m*-drop and *m*-auto is proposed to learn models more robust to missing data modalities in both the training and inference stages.

5 Related Work

Collaborative filtering (CF) is the most commonly used approach for recommender systems. CF methods generally utilize the item-user feedback matrix. Matrix factorization (MF) is the most popular CF method (Koren et al., 2009) due to its simplicity, performance, and high accuracy as demonstrated in previous work (Chen et al., 2015). Another strength of MF, making it widely used in recommender systems, is that side information other than existing ratings can easily be integrated into the model to further increase its accuracy. Such information includes social network data (Li et al., 2015; Lagun and Agichtein, 2015; Zhao et al., 2016; Xiao et al., 2017), locations of users and items (Lu et al., 2017) and visual appearance (He and McAuley, 2016; Salakhutdinov and Mnih, 2007) proposed Probabilistic Matrix Factorization (PMF) which extends MF to a probabilistic linear model with Gaussian noise. Following PMF, there are many extensions (Salakhutdinov et al., 2007; Chen et al., 2013; Zheng et al., 2016; Zhang et al., 2016; He et al., 2016b, 2017) aiming to improve its accuracy.

Unfortunately, CF methods suffer from the cold-start problem when dealing with new items or users without rich information. Content based filtering (CBF) (Pazzani and Billsus, 2007), on the other hand, is able to alleviate the cold-start problem by taking auxiliary product and user information (texts, images, videos, etc.) into consideration. Recently, several approaches (Almahairi et al., 2015; Xu et al., 2014; He et al., 2014; Tan et al., 2016) were proposed to consider the information of review text to address the data sparsity problem which leads to the cold-start problem. The topic model (e.g. LDA (Blei et al., 2003)) based approaches including CTR (Wang and Blei, 2011), HFT (McAuley and Leskovec, 2013), RMR (Ling et al., 2014), TriRank (He et al., 2015), and sCVR (Ren et al., 2017) achieve significant improvements compared to previous work on recommender systems.

Inspired by the recent success of deep learning techniques (Krizhevsky et al., 2012; He et al., 2016a), some deep network based recommendation approaches have been introduced (Wang et al., 2015; Sedhain et al., 2015; Wang et al., 2016b; Seo et al., 2017; Xue et al., 2017; Zhang et al., 2017). Deep cooperative neural network (Deep-CoNN) (Zheng et al., 2017) was introduced to learn a joint representation from items and users using two coupled network for rating prediction. It is the first approach to represent users and items in a joint manner with review text. TransNets (Catherine and Cohen, 2017) extends Deep-CoNN by introducing an additional latent layer representing the user-item pair. NRT (Li et al., 2017) is a method for rating prediction and abstractive tips generation (Zhou et al., 2017). A four-layer neural network was used for rating regression model. NRT outperforms the state-of-the-art methods on rating prediction. There is a large body of work for recommender systems and we refer the reader to for surveys of state-of-the-art CF based approaches, CBF methods, and deep learning based methods, respectively (Shi et al., 2014; Lops et al., 2011; Zhang et al., 2017).

Our work differs from previous work in that we simultaneously address various types of missing data together with the data-sparsity and cold-start problems.

6 Conclusion

We presented LRMM, a framework that improves the performance and robustness of recommender systems under missing data. LRMM makes novel contributions in two ways: multimodal imputation and jointly alleviating the missing modality, data sparsity, and cold-start problem for recommender systems. It learns to recommend when entire modalities are missing by leveraging inter- and intra-modal correlations from data through the proposed *m*-drop and *m*-auto methods. LRMM achieves state-of-the-art performance on multiple data sets. Empirically, we analyzed LRMM in different data sparsity regimes and demonstrated the effectiveness of LRMM. We aim to explore a generalized domain adaptation approach for recommender systems with missing data modalities.

References

Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron C. Courville. 2015. Learning distributed representations from reviews for collaborative filtering. In *RecSys*, pages 147–154.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Rose Catherine and William W. Cohen. 2017. Transnets: Learning to transform for recommendation. In *RecSys*, pages 288–296.

Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *WWW*.

Tianqi Chen, Hang Li, Qiang Yang, and Yong Yu. 2013. General functional matrix factorization using gradient boosting. In *ICML (1)*, volume 28, pages 436–444.

Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander Rudnicky. 2015. Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding. In *ACL-IJCNLP*, volume 1, pages 483–494.

Gabriela Csurka. 2017. Domain adaptation for visual applications: A comprehensive survey. *CoRR*, abs/1702.05374.

Alberto García-Durán, Roberto Gonzalez, Daniel Oñoro-Rubio, Mathias Niepert, and Hui Li. 2018. Transrev: Modeling reviews as translations from users to items. *CoRR*, abs/1801.10095.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517.

Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*, pages 1661–1670.

Xiangnan He, Ming Gao, Min-Yen Kan, Yiqun Liu, and Kazunari Sugiyama. 2014. Predicting the popularity of web 2.0 items based on user comments. In *SIGIR*, pages 233–242.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*, pages 173–182.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016b. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558.

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

Yu-Yang Huang and Shou-De Lin. 2016. Transferring user interests across websites with unstructured text for cold-start recommendation. In *EMNLP*, pages 805–814.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434.

Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114.

Dmitry Lagun and Eugene Agichtein. 2015. Inferring searcher attention by jointly modeling user interactions and content salience. In *SIGIR*, pages 483–492.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562.

Hui Li, Dingming Wu, Wenbin Tang, and Nikos Mamoulis. 2015. Overlapping community regularization for rating prediction in social recommender systems. In *RecSys*, pages 27–34.

Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural rating regression with abstractive tips generation for recommendation. In *SIGIR*, pages 345–354.

Guang Ling, Michael R. Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *RecSys*, pages 105–112.

Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer.

Ziyu Lu, Hui Li, Nikos Mamoulis, and David W. Cheung. 2017. HBGG: a hierarchical bayesian geographical model for group recommendation. In *SDM*, pages 372–380.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Benjamin M. Marlin. 2003. Modeling user rating profiles for collaborative filtering. In *NIPS*, pages 627–634.

Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, pages 165–172.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. 2011. Multimodal deep learning. In *ICML*, pages 689–696.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359.

Michael J. Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341.

Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. 2017. Social collaborative viewpoint regression with explainable recommendations. In *WSDM*, pages 485–494.

Francesco Ricci, Lior Rokach, and Bracha Shapira, editors. 2015. *Recommender Systems Handbook*. Springer.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. 2007. Restricted boltzmann machines for collaborative filtering. In *ICML*, volume 227, pages 791–798.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. 2002. Methods and metrics for cold-start recommendations. In *SIGIR*, pages 253–260.

Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW (Companion Volume)*, pages 111–112.

Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *RecSys*, pages 297–305.

Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 47(1):3:1–3:45.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Nitish Srivastava and Ruslan Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *NIPS*, pages 2231–2239.

Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 11–16. ACM.

Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-boosted latent topics: Understanding users and items with ratings and reviews. In *IJCAI*, pages 2640–2646.

Luan Tran, Xiaoming Liu, Jiayu Zhou, and Rong Jin. 2017. Missing modalities imputation via cascaded residual autoencoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1405–1414.

Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. In *NIPS*, pages 4964–4973.

Cheng Wang, Haojin Yang, and Christoph Meinel. 2016a. A deep semantic framework for multimodal representation learning. *Multimedia Tools Appl.*, 75(15):9255–9276.

Cheng Wang, Haojin Yang, and Christoph Meinel. 2018. Image captioning with deep bidirectional lstms and multi-task learning. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(2s):40.

Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456.

Hao Wang, Xingjian Shi, and Dit-Yan Yeung. 2016b. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In *NIPS*, pages 415–423.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*, pages 1235–1244.

Xuepeng Wang, Kang Liu, and Jun Zhao. 2017. Handling cold-start problem in review spam detection by jointly embedding texts and behaviors. In *ACL*, volume 1, pages 366–376.

Karl R. Weiss, Taghi M. Khoshgoftaar, and Dingding Wang. 2016. A survey of transfer learning. *J. Big Data*, 3:9.

Lin Xiao, Zhang Min, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2017. Learning and transferring social and item visibilities for personalized recommendation. In *CIKM*, pages 337–346.

Yinqing Xu, Wai Lam, and Tianyi Lin. 2014. Collaborative filtering incorporating review text and co-clusters of hidden user communities and item groups. In *CIKM*, pages 251–260.

Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. In *IJCAI*, pages 3203–3209.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete collaborative filtering. In *SIGIR*, pages 325–334.

Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep learning based recommender system: A survey and new perspectives. *CoRR*, abs/1707.07435.

Wayne Xin Zhao, Sui Li, Yulan He, Edward Y. Chang, Ji-Rong Wen, and Xiaoming Li. 2016. Connecting social media to e-commerce: Cold-start product recommendation using microblogging information. *IEEE Trans. Knowl. Data Eng.*, 28(5):1147–1159.

Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*, pages 425–434.

Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A neural autoregressive approach to collaborative filtering. In *ICML*, volume 48, pages 764–773.

Ming Zhou, Mirella Lapata, Furu Wei, Li Dong, Shaohan Huang, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *EACL (1)*, pages 623–632.

Content Explorer: Recommending Novel Entities for a Document Writer

Michal Lukasik
Google AI
mlukasik@google.com

Richard Zens
Google AI
zens@google.com

Abstract

Background research is an essential part of document writing. Search engines are great for retrieving information once we know what to look for. However, the bigger challenge is often identifying topics for further research. Automated tools could help significantly in this discovery process and increase the productivity of the writer. In this paper, we formulate the problem of recommending topics to a writer. We consider this as a supervised learning problem and run a user study to validate this approach. We propose an evaluation metric and perform an empirical comparison of state-of-the-art models for extreme multi-label classification on a large data set. We demonstrate how a simple modification of the cross-entropy loss function leads to improved results of the deep learning models.

1 Introduction

An important part of writing about some topic is researching relevant background material, which can be a challenging and tedious task. If the author has a clear idea what to write about, web search engines provide an excellent tool for retrieving information. However, the author often first has to spend time discovering which topics are related, i.e., she needs to conduct *exploratory search* (White and Roth, 2009). This is challenging with traditional keyword-based search engines like *Google*, which are tailored to providing the information that a user is explicitly searching for (Marie, 2014).

Exploratory search poses a challenge in the writing process. It has been shown that the *exploration* phase elicits strong negative feelings in students writing essays (Smith and Campbell, 1999; Kuhlthau, 1990) and that students consider the research activity to be at the cost of other pending commitments (Smith and Campbell, 1999). Moreover, exploratory search is more cognitively demanding than lookup search tasks (Marie, 2014). Automatic tools tailored to exploratory search for new content could significantly alleviate the difficulty the writer faces in the research phase.

Intuitively, the tool should suggest topics for further research which are relevant and interesting. Hence the suggestions need to be related to the document and should not be too obvious. For instance, a student writing about the *monarch butterfly* might benefit from a suggestion *milkweed* as it is both relevant and interesting. On the other hand, *insect* would be a rather poor suggestion as it is too obvious. To summarize, the key challenge we set out to address is: *For a given piece of text, what are the related topics and how important are they?* In Section 3.2 we describe a user study where annotators evaluated usefulness of entities, which we then use to identify characteristics of good suggestions.

It is worth mentioning that systems for providing automatic recommendations of entities to a document writer have recently been introduced by the industry, including Google Explore in Docs[1] and Microsoft Researcher[2]. The systems are proprietary and their design is not published, making the comparison not possible. To the best of our knowledge this is the first documented attempt to address the problem of recommending novel entities to a document writer. We hope this paper will attract the interest of the research community and inspire future work.

The contributions of this paper are:

1. Formulating the problem of recommending future entities to a document writer.

---

[1] http://bit.ly/2f6CVeR

[2] http://bit.ly/29XaPAJ

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3371–3380
Brussels, Belgium, October 31 - November 4, 2018.

2. Conducting a user study to identify what topic suggestions users find useful.

3. Formulating the problem and defining an automatic evaluation metric, both motivated by the user study.

4. Evaluating state of the art approaches to extreme multi-label classification.

5. Demonstrating how a modified loss function helps improve over state of the art neural network models.

2 Related Work

Recommending topics to a document writer can be viewed in the context of different fields which we discuss below.

Exploratory Search White and Roth (2009) define exploratory search as "information seeking problem context that is open-ended, persistent, and multi-faceted" and "information-seeking processes that are opportunistic, iterative, and multi-tactical". Research on exploratory search focuses on supporting a user in the interactive and iterative process of seeking for information and includes: designing better interfaces, visualization of search results, clustering of results, supporting serendipitous discoveries, supporting different user profiles (Marie, 2014). Instead, in this work we aim to fully automate discovery of *relevant* and *interesting* topics to the document writer. Moreover, the input in our case is an initial portion of a document written by a user rather than a query or a single entity, as is usually the case in the information retrieval setups.

Extreme Multi-label Classification Extreme multi-label classification (XML) is an instance of a multi-label classification problem (i.e., where multiple labels can be assigned to a single example at the same time) under a large label space. There are multiple works investigating XML, including random forest (RF) (Prabhu and Varma, 2014; Jain et al., 2016) and embedding (Bhatia et al., 2015) approaches. These methods rely on bag of words feature representation, ignoring the sequential nature of text. Neural networks (NNs) have been successful in modeling NLP tasks through their ability to learn structure, however have not been widely studied for XML problems. Covington et al. (2016) applied NNs to YouTube video recommendation, and Liu et al. (2017) showed a convolutional neural network architecture to outperform strong baselines on a range of NLP tasks. The importance of label weighting for recommending rare items has not been considered in the NNs for extreme multi-label classification, which is a central problem in our task, as in other tasks with extreme label spaces (Jain et al., 2016). In Section 5 we describe the state of the art RF and NN approaches to XML.

Entity Retrieval and Tagging Assigning entities to an input has been considered, however in different contexts. In tag recommendation, a text is summarized with a set of entities (Song et al., 2011) and in entity search one needs to answer queries about entities against a set of documents and entities (Cheng et al., 2007). These tasks are different as no new content is predicted.

The Related Entity Finding (REF) challenge of the Text Retrieval Conference (TREC) considered a task where given a source entity, a relation and a target type, a target entity needs to be identified satisfying the required relation (Balog et al., 2010). This is a different problem since no specification for relations between the entities is provided in the input document from a user.

The work on semantic relatedness of entities (Milne and Witten, 2008) is different since, as explained in the introduction, good entity recommendations are not necessarily those which are semantically related to entities from the user text. Bordino et al. (2013) considered retrieving entities related to a query in the question answering scenario. The authors did not set the problem in a supervised learning setup and instead find entities closest in terms of similarity of documents containing them. In contrast, in Section 3.2 we justify a supervised learning setup of the task.

The TREC Complex Answer Retrieval Track (CAR)[3] is a challenge where based on a document outline, related text passages and entities are retrieved (Dietz et al., 2017). In this formulation it is assumed that a general outline of what a document author intends to cover is given, thus providing a clear guidance for what is relevant. Instead, our input is an initial part of a document and we seek to find novel entities based on the input.

---

[3] `http://trec-car.cs.unh.edu`

Recommending Rare Items Information retrieval applications emphasized the importance of retrieving rare labels rather than common ones which are likely to be already known to a user. Baeza-Yates and Ribeiro-Neto (1999) defined novelty of a set of recommendations as the proportion of unknown items to the user, a challenging definition to work with when user's knowledge is unknown (Hurley and Zhang, 2011). Bordino et al. (2013) explored the problem of retrieving *serendipitous* results when retrieving answers to queries. The authors built an information retrieval system based on finding entities most often co-occurring with a query entity and employed IDF (inverse document frequency) for filtering out overly generic answers. Also many other works employ IDF for rewarding rare items (Zhou et al., 2010; Vargas and Castells, 2011; Wu et al., 2014; Jain et al., 2016). Here we take a supervised learning approach to the entity recommendation problem, demonstrate the usefulness of IDF scoring in the context of our problem with a user study, and utilize IDF in the evaluation metric.

Hurley and Zhang (2011) define novelty of an item in the context of the set of relevant recommendations using an average dissimilarity from other items in the set. Similarly, the Maximal Marginal Relevance metric evaluates a set of retrieved items in an information retrieval problem by rewarding the diversity of the set (Carbonell and Goldstein, 1998). Note this is a different notion of novelty from that considered in this work.

3 Investigating the Problem

In this section we investigate the problem formulation and what it means for an entity to be a useful suggestion for a user.

3.1 Problem Definition

Let a document d be represented as a sequence of entities $(e^1, \dots, e^{|d|}) = E$. We partition these entities into two sets C_d and F_d: those which occur in the first h sentences (in our case $h = 10$) and those which don't. Hence, $C_d \cap F_d = \emptyset$ and $C_d \cup F_d = E$.

We are not interested in retrieving all entities that could occur in the future, because it is not feasible to provide all such recommendations to a user. Instead, we focus on ranking the target entities, and selecting the most relevant k to suggest, which is a standard practice in retrieval tasks (Jain et al., 2016).

3.2 What Entities are Good Suggestions?

In this section we describe a user study we ran for testing the hypotheses about which entities constitute useful recommendations. In particular, we test whether what a user writes next in a document is actually considered to be a good recommendation by the raters and whether IDF score correlates with how useful an entity is considered.

Human Evaluation Dataset We consider 1000 Wikipedia documents for a human evaluation study. For each document, let E denote the set of all entities that occur in the document and E_p denote entities occurring in the initial passage p. We found 5 most co-occurring entities from $E \setminus E_p$[4] with the entities from E_p across sentences from a large scale web documents corpus (the dataset described later in Section 4). Annotators were then asked to rate entity suggestions against an input passage p in terms of how *useful* they are in the process of continuing to write the document. Each entity was rated by 3 annotators with a score ranging from 1 to 5 and the total number of annotators was 845. To evaluate agreement among raters, we employ the Intraclass Coefficient for consistency[5] (ICC; (McGraw and Wong, 1996)) for two-way random effects model with the effects corresponding to a rater and a rated entity. We found the average score ICC to be equal $\mathrm{ICC}(\mathrm{C}, 3) = 0.69$. We refer the reader to the supplemental material for details about how ICC was applied.

Evaluating the Supervised Setup We found that the mean rating for entities which actually occur in the future is 3.24 ± 0.83, whereas the mean rating for the other entities is 2.59 ± 0.93. Running an independent T-test for comparing means between the two groups yields a p-value $p < 0.001$. This supports the supervised learning setup of entity suggestion, where we split documents from a corpus into two parts, and unseen entities from later parts form labels for the initial parts.

Does IDF Correlate with Usefulness? One of the requirements we set for the entity recommendation is the interestingness of the entities. As

---

[4] I.e., excluding entities occurring in the passage.

[5] Intraclass Coefficient is an approach to evaluate inter-rater reliability when ratings are organized into groups, as in our case, where entities are grouped into passages against which they are scored.

reported in Section 2, one popular approach to recommending rare items in information retrieval tasks is weighting their utility by their IDF score (Bordino et al., 2013; Jain et al., 2016). Let us analyze in the document writing setup how useful users perceive entities which score high in IDF. To this end, we measure how the *ground truth reviewer* (the average rating given by the human annotators to an item) correlates against the baseline rating *B* and how it correlates against the IDF weighted rating *W*. Here, the baseline rating *B* simply rates an entity with 0 if it does not occur in the future of the document and with 1 if it does occur in the future of the document. The IDF weighted rating *W* rates an entity with its IDF score multiplied by the score returned by the reviewer *B*. We employ ICC(A, 1) to find the agreements between pairs of ratings. We find ICC(A, 1) between the *ground truth reviewer* and *B* to be 0.30, and between the *ground truth reviewer* and *W* to be 0.42. This shows that entities with high IDF scores are perceived as more useful. Note the potential space for improvement by finding a metric which would yield even higher correlation against the human rating than IDF.

3.3 Automatic Evaluation

Recall that we aim at rewarding entities which are both *relevant* (i.e., within the target set of entities) and *interesting* (i.e., are not obvious, as the entity *insect* was for the *monarch butterfly* example from Section 1). Since we care only about the k highest ranked recommendations from the system, a potentially useful metric for evaluation is prec@k$=\frac{1}{k}\sum_{j=1}^{k} I_{\{p^j \in F_d\}}$, where p^j are the predictions. Even though prec@k captures relevancy, it fails to distinguish between generic and specific entities. Documents tend to contain many entities in the target set, some of which are generic (e.g. *insect*). Prec@k scores all entities the same and rewards predictions of generic entities occurring across almost all web documents (e.g. *Internet*), which are arguably not interesting.

We propose a metric based on cumulative gain (Järvelin and Kekäläinen, 2002), where we use IDF for scoring the relevance of labels:

$$\text{CG-IDF@k} = \sum_{j=1}^{k} I_{\{p^j \in F_d\}} \text{IDF}(p^j).$$

To facilitate interpretability, we use a normalized version of CG-IDF@k, the normalized cumulative gain (NCG-IDF@k). NCG-IDF@k is obtained by dividing CG-IDF@k by the maximum sum of IDF scores of k entities from the target set.

Why would IDF help make entities more interesting? As shown by the user study in Section 3.2 IDF correlates with how relevant an entity is perceived by a user. Moreover, as reported in Section 2, IDF is widely used for boosting rare, novel items (Bordino et al., 2013; Jain et al., 2016).

4 Dataset

We consider the problem of recommending topics that an author writing a document might be interested in writing about next. In Section 3.2 we used a human evaluation study to support a supervised learning setup for this problem, where what is written later in a document is deemed to be a good recommendation for an initial part. The Web is a rich source of documents which facilitates construction of large datasets. Below we describe details of how we construct the dataset composed of web documents and report basic statistics thereof.

Construction We collected a dataset of 10M web documents with high pagerank (Page et al., 1998) scores across the Web as of November 2017. We ran entity recognition and linking to the Freebase knowledge graph using the Google NLP cloud.[6] Moreover, only the 100K most frequent entities are kept.[7] We randomly select 10K documents for the test set and 10K for the validation set which we use for hyperparameter tuning.

Statistics As shown in Figure 1 the document frequencies for the 100K most frequent entities from the dataset follow a power law distribution, with a small number of very frequent entities and many infrequent ones. The average number of documents per target entity is 4694.95, which is 3 orders of magnitude smaller than the maximum frequency. We found that the average number of future entities per document is 96, and the average number of input entities is 10. In Figure 2 we show the percentage of times a context entity is followed by a particular target entity (row and column, respectively). Notice the matrix is asymmetric. For

---

[6] `http://tinyurl.com/h246dnz`

[7] 100K entities are kept due to the challenges in handling larger output spaces by the models. Scaling to larger output spaces is possible but requires modifications which we leave for future work.

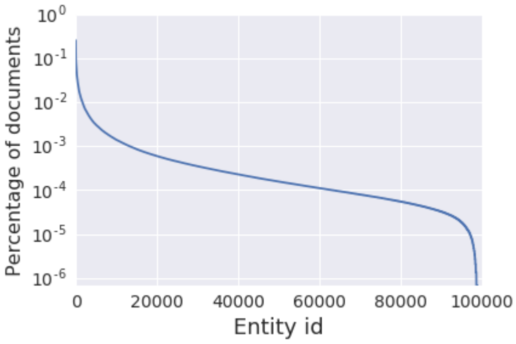

Figure 1: Percentage of documents for which each target entity occurs.

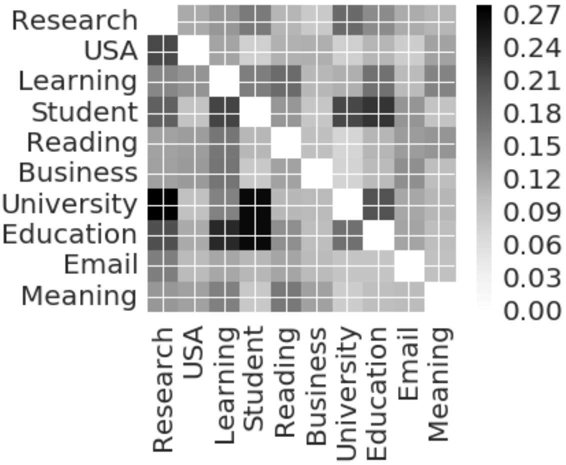

Figure 2: A heatmap showing what percentage of times a context entity (row) is followed by a particular target entity (column).

instance, USA is more likely to be followed by Research than the other way around.

5 Baselines and Models

In this section we describe the models that we applied to the entity recommendation problem.

5.1 Linear Models

Let us represent input entities via n-hot vector representation, i.e., vector entries corresponding to entities present in the input set are set to one, whereas other entries are set to zero. We consider *linear models*, where such an input vector is multiplied by a square weight matrix of size #entities × #entities, and a resulting vector contains scores for predicted entities. There are different possibilities for filling the entries of the weight matrix. We considered multiple options:

1. N(C, F) obtained by putting the raw co-occurrences of the context entity C (corresponding to a row) and the future entity F (corresponding to a column).

2. P$(F|C)$ obtained by normalizing the N(C, F) matrix row-wise.

3. PPMI(C, F) (positive pointwise mutual information) aims to show how much more likely an entity F is to occur for a context C compared to observing them independently (Jurafsky and Martin, 2000). PPMI is a popular preprocessing step on the co-occurrence matrix before applying dimensionality reduction, such as SVD (see Section 5.2) (Herbelot and Vecchi, 2015).

5.2 Matrix Factorization

Matrix factorization methods (MFM) are among the most popular approaches for recommendation systems (Koren et al., 2009). This approach works similar to the *linear model*, except that the resulting matrix is decomposed into a sequence of smaller matrices, the product of which approximates the *linear model*. This approach reduces the number of parameters, which speeds predictions, saves memory, and may improve robustness to overfitting. We employ SVD and reduce the rank of the resulting matrix of the *linear model* to 100. MFM is applied analogously to the *linear model*, namely a vector encoding input entities is mapped into a vector with scores of target entities.

5.3 Random Forests

FastXML FastXML (Prabhu and Varma, 2014) is one of the most popular approaches to extreme multi-label classification. Each tree in this random forest model is grown recursively by splitting each node by a separating hyperplane. The hyperplane weight vector is chosen by optimizing for nDCG score (a non-differentiable loss which poses issues in the NN framework) and is additionally regularized with ℓ_1 norm to induce sparsity. Each leaf node contains a probability distribution over labels. At prediction time, the distributions from reached leaf nodes across the trees are aggregated and a final ranking of entities is created.

PFastreXML PFastreXML (Jain et al., 2016) builds on the FastXML model by introducing two modifications. First, the nDCG loss function is replaced by a propensity weighted nDCG (in our case, the IDF weighted nDCG), resulting in rare items ranked higher. Second, the final ranking is re-ranked using tail label classifiers, which aims at further improvements in how highly rare items are scored.

Experimental Setup The scale of our dataset is bigger than that reported by Jain et al. (2016) and consequently the size of the generated random forests tends to be very large. Therefore, apart from keeping most of the hyperparameters as reported by Jain et al. (2016), in order to limit the size of generated trees, we modify two hyperparameters: the maximum number of labels per leaf node is set to 50 (instead of 10), and the maximum number of training examples per leaf node is set to 50 (instead of 10). Even under such a setup, the size of the generated models is around 150GB.

5.4 Neural models

Recently NN models have been applied to extreme classification. In this section we discuss the YouTube neural model used for recommending YouTube videos (Covington et al., 2016) and next the XML-CNN model which has been demonstrated to achieve competitive results on a few NLP tasks (Liu et al., 2017).

The YouTube Model A NN model (depicted in Figure 3(a)) has been successfully applied for the YouTube recommendation problem (Covington et al., 2016). In the model, the input entities are embedded into a latent dimensionality V. The embeddings are then summed, and the resulting vector is passed through a number of layers, where in each layer an input vector is passed through matrix multiplication and a rectified linear unit (RELU) (Nair and Hinton, 2010). Afterwards, the resulting vector is converted into the space of dimensionality equal to the number of entities via an output layer. This way, scores are obtained over the entity space, which are used to choose the highest scored entities for predictions. The loss function is the cross-entropy (CE) between the soft-maxed activations and a uniform distribution over the target entities.

XML-CNN Convolutional neural networks (CNNs) have been successfully applied to a range of NLP problems (Kim, 2014; Bitvai and Cohn, 2015). Recently Liu et al. (2017) demonstrated how CNNs can be effective for XML problems. We depict their architecture in Figure 3(b). The input sequence is embedded in V dimensions, and passed through a convolutional layer, a fully connected layer and an output layer. After the convolutional layer the authors employ dynamic pooling which helps retain information about where in the input sequence the convolutions got triggered. The loss function is binary cross entropy (BCE), which considers labels individually rather than jointly. BCE is given by the formula $\mathrm{BCE}(\mathbf{p},\mathbf{y}) = \sum_{j=1}^{j=M} y_j \log(\sigma(p_j)) + (1-y_j)\log(1-\sigma(p_j))$, where σ is the sigmoid function. Lastly, a hidden bottleneck layer is used, which is motivated by introducing better generalization. In the experiments we did not find the dynamic max pooling to be beneficial, and instead we found max pooling with a higher number of filters (512 compared to 32 in (Liu et al., 2017)) to be better, keeping the number of parameters.

Modified Loss Function A potential problem with CE and BCE is that they reward all entities equally. However, as demonstrated in Section 3.2, some entities are more valuable than others. Since we would like to promote interesting entities over generic ones, we consider an alternative training loss function to BCE and CE which incorporates the IDF scores of target entities. We use CE, but instead of comparing the soft-maxed activations to a uniform distribution over target entities, we compare against normalized IDF scores from the training set. We call this loss function CE-IDF. When using CE-IDF with XML-CNN, we found that adding ℓ_2 norm of the weights and the cross-entropy regularization (Pereyra et al., 2017) helps prevent the model from overfitting. We select the hyperparameters controlling these two regularization terms using a held out validation set. This contribution is analogous to that of PFastreXML over FastXML due to Jain et al. (2016), where weighting the loss function in random forests by label propensity scores helps achieve better propensity weighted results.

Experimental Setup To regularize the networks we use a 50% dropout rate (Srivastava et al., 2014). We set the dimensionality of the embeddings to 1000, and the hidden layer size to 512. The hyperparameters for XML-CNN are set as reported by Liu et al. (2017), except for the ℓ_2 and cross-entropy regularization hyperparameters

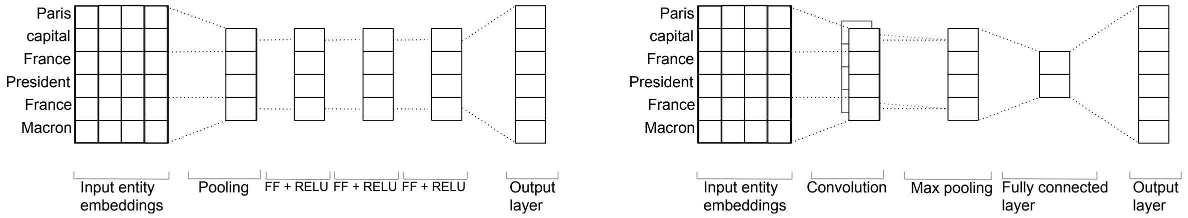

(a) YouTube model (Covington et al., 2016)

(b) XML-CNN (Liu et al., 2017)

Figure 3: Neural network architectures applied to the entity recommendation problem.

	NCG-IDF				prec				model
	@1	@3	@5	@7	@1	@3	@5	@7	size
$p(F)$	14.86	14.50	14.61	14.56	36.76	32.69	31.12	30.23	28B
N(C,F) base	20.12	19.95	19.43	18.91	46.92	42.89	40.29	38.15	4.7GB
SVD	19.99	19.79	19.20	18.72	46.68	42.60	39.88	37.81	0.1GB
P$(F\|C)$ base	22.92	22.77	22.17	21.86	51.50	47.81	44.97	43.19	4.7GB
SVD	21.56	21.00	20.65	20.31	48.79	44.44	42.22	40.54	0.1GB
PPMI(C,F) base	22.92	20.86	19.45	18.42	23.95	21.17	19.33	18.01	4.7GB
SVD	18.22	18.22	18.03	17.89	26.49	25.36	24.31	23.49	0.1GB
Youtube base	31.30	31.12	31.07	31.02	54.85	52.62	51.09	49.79	1.8GB
IDF	32.95	32.24	32.00	31.62	50.95	49.07	47.63	46.57	1.8GB
XML-CNN base	33.13	32.20	31.93	31.75	58.72	53.60	52.05	50.35	1.8GB
IDF	33.89	33.42	33.02	32.76	51.99	49.54	48.40	47.24	1.8GB
FastXML	35.31	34.39	33.74	33.15	69.98	63.78	60.16	57.37	150GB
PFastreXML	36.19	34.94	34.12	33.31	55.09	51.51	49.50	47.85	150GB

Table 1: Experimental results for NCG-IDF@k and prec@k scores for different methods.

which were selected on the validation set. Note we optimize all parameters, including the entity embeddings, on the training data.

6 Experiments

In Table 1 we report results from the experiments on the 10M web documents dataset for prec and NDCG-IDF metrics for $k = 1, 3, 5, 7$, limiting k to small values as is common in the recommendation problems from large sets of items (Jain et al., 2016). The $p(F)$ baseline always predicts entities according to their frequency over the training set. It can be viewed as maximum likelihood estimate (MLE) for the model which is only composed of a bias vector (i.e., input entities are ignored). Notice the relatively high performance when the most popular entities are taken. For example, in 36.76% of cases entity *Research* (the most popular future entity from the corpus) is in the future of the document (as can be also seen from Figure 1, where the entity *Research* corresponds to the leftmost point of the graph). This constitutes a high value, as the vocabulary consists of $100K$ entities.

Among the *linear models*, P$(F|C)$ yields the highest scores, significantly outperforming the baselines. PPMI$(F|C)$ model yields relatively high NCG-IDF scores (although in most cases lower than P$(F|C)$), and very low precision scores. Notice that the SVD methods are consistently worse than the *linear models*. This shows that no additional generalization is gained when lowering the number of parameters of the *linear models*. When experimenting with higher ranks for SVD decomposition we found the performance increases, but does not improve over the *linear models*.

NNs improve over *linear models* according to both NCG-IDF and prec scores. This is especially apparent for NCG-IDF, where the relative improvement is very significant. XML-CNN is in all cases better than the Youtube model, which shows how utilizing more linguistic structure than simply bag of entities is helpful in the NN framework. Both Youtube and XML-CNN models with a modified loss function improve over the basic NN models in terms of the NCG-IDF metrics, showing that a simple adjustment of a loss function in the NN framework can lead to more rare

input	predictions			
	XML-CNN base	XML-CNN IDF	FastXML	PFastreXML
NASA recently released a study suggesting that the Antarctic Ice Sheet is gaining more ice than it is losing – a finding that at first blush seems to contradict the idea of global warming.	*glacier*, *Greenland*, *Earth*, *research*	*sea level rise*, *glacier*, *temperature*, *meltwater*	*ocean*, *research*, *temperature*, *understanding*	*glacier mass balance*, *ice shelf*, *glaciology*, *Greenland ice sheet*
Making your mobile web app talk: software Architecture conference. Microservices training O'reilly.	*learning*, *project*, *experience*, *information*	*application software*, *Javascript*, *presentation*, *organization*	*technology*, *service*, *learning*, *industry*	*technology*, *application software*, *project*, *Open Source*
Face recognition algorithms use a large dataset of photos labeled as having a face or not to estimate a function that predicts the presence y of a face from pixels x. As empirical economists, how can we use them?	*number*, *research*, *information*, *result*	*analysis*, *result*, *sample*, *statistics*	*information*, *data*, *number*, *result*	*result*, *analysis*, *set*, *R programming language*
Here we report the isolation of an arsenate reductase gene (PvACR2) from gametophytes that can suppress the arsenate sensitivity and arsenic hyperaccumulation phenotypes of yeast (Saccharomyces cerevisiae).	*information*, *e-mail*, *industry*, *2017*	*e-mail*, *information*, *learning*, *reading*	*cell*, *addition*, *data*, *analysis*	*Vector (biology)*, *DNA/RNA primer*, *Plant Physiology Journal*, *Primary Structure*

Table 2: Example inputs and corresponding top 4 entity predictions from the models.

entities being recommended. This comes at the cost of lowering prec@k scores, which however correlates with user judgments to a lesser extent, as we showed in Section 3.2. The Random Forest models turn out to be the most competitive. Notice that no linguistic structure is captured in FastXML models, only the bags of entities. This is in contrast with XML-CNN approach which looks at local contexts of feature entities. FastXML performs particularly well on the precision scores, which however is not necessarily useful, as demonstrated in the examples discussed later.

Last, we inspect the sizes of the different models reported in the rightmost column of Table 1. Model size is an important factor to consider in practical applications, e.g. when deploying a system on the device. PFastreXML model takes 150GB, by far the most of all methods, resulting in its capability in recommending tail entities. The *linear models* take 4.7GB related to the fact that in the full $100K \times 100K$ co-occurrence matrix approximately 11% of entries are non-zero. Applying SVD matrix decomposition helps reduce this size significantly. The NN models take around 2GB, significantly less than the random forests.

Analysis To demonstrate the usefulness of the models, in Table 2 we report top 4 entity predictions from XML-CNN and FastXML models for a few example inputs from the test set. Notice how predictions from XML-CNN base are more generic than from XML-CNN IDF. In particular, for the first input related to Antarctic Ice Sheet gaining ice entities *Earth* and *research* are recommended. The relevance of them is clear, however their usefulness is doubtful due to how obvious to the writer they may be. Due to often making such safe predictions XML-CNN base scores higher in precision than XML-CNN IDF. XML-CNN IDF instead makes more specific recommendations, such as *sea level rise* or *temperature* for the first input. This shows how much more beneficial scoring high in NDCG-IDF compared to precision is. An analogous phenomenon takes place between FastXML and PFastreXML – despite FastXML achieving very high precision scores, the predictions tend to be less interesting.

When comparing PFastreXML results against XML-CNN IDF, the results become even more specific. For the first input the entities such as *glaciology* or *Greenland ice sheet* are recommended. For the third input about econometrics and machine learning, both XML-CNN base and FastXML predict a generic entity *information*, XML-CNN IDF predicts more specific *statistics*, and PFastreXML recommends *R programming language*, a popular statistics toolkit for statisticians and mathematicians with support for machine learning. The usefulness of PFastreXML predictions is particularly profound for the last example about gene biology, where all other approaches back off to very generic entities.

7 Conclusions

In this paper we introduced the problem of entity recommendation for a document writer. Good entity recommendations need to be both relevant and interesting, which we motivated with a user study. In particular, we showed how entities which users write in the document are considered as good recommendations and how IDF score correlates with how useful an entity is considered to be. We corroborated this with example predictions, showing how models scoring higher in metrics weighted by IDF provide more interesting suggestions.

The problem of recommending content to document writers has recently been addressed by industry with tools like Google Explore in Docs and Microsoft Researcher, however the systems are proprietary and the methods have not been published. In particular, this work is the first to formalize the problem and provide insights about it to the research community. We hope that this work will inspire further research on recommending novel content to document writers.

Many avenues for further work can be identified, including: finding a better metric capturing novelty of entities, analyzing the influence of the size of the input passage to quality of predictions, and experimenting with new models. Moreover, recent work has considered incorporating knowledge graph information for better use of entity features (Dalton et al., 2014; Liu et al., 2018). This could also be explored for better feature representation in our problem.

Acknowledgments

We would like to thank Kishore Papineni and Shankar Kumar for inspiring and useful discussions which greatly impacted the shape of this work. We also thank the anonymous reviewers for their insightful suggestions.

References

Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Krisztian Balog, Pavel Serdyukov, and Arjen P. de Vries. 2010. Overview of the TREC 2010 entity track. In *Proceedings of The Nineteenth Text REtrieval Conference, TREC 2010*.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 730–738. Curran Associates, Inc.

Zsolt Bitvai and Trevor Cohn. 2015. Non-linear text regression with a deep convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 180–185.

Ilaria Bordino, Yelena Mejova, and Mounia Lalmas. 2013. Penguins in sweaters, or serendipitous entity search on user-generated content. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 109–118, New York, NY, USA. ACM.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336, New York, NY, USA. ACM.

Tao Cheng, Xifeng Yan, and Kevin Chen-Chuan Chang. 2007. Supporting entity search: A large-scale prototype search engine. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 1144–1146, New York, NY, USA. ACM.

Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 191–198, New York, NY, USA. ACM.

Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 365–374, New York, NY, USA. ACM.

Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. 2017. TREC complex answer retrieval overview. In *Proceedings of The Twenty-Sixth Text REtrieval Conference, TREC 2017*.

Aurélie Herbelot and Eva Maria Vecchi. 2015. Building a shared world: mapping distributional to model-theoretic semantic spaces. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 22–32, Lisbon, Portugal. Association for Computational Linguistics.

Neil Hurley and Mi Zhang. 2011. Novelty and diversity in top-n recommendation – analysis and evaluation. *ACM Trans. Internet Technol.*, 10(4):14:1–14:30.

Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 935–944, New York, NY, USA. ACM.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st edition. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Carol C. Kuhlthau. 1990. Inside the search process: Information seeking from the users perspective. *Journal of the American Society for Information Science*, 5(42):361–371.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 115–124, New York, NY, USA. ACM.

Zheng-Hao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. *CoRR*, abs/1805.07591.

Nicolas Marie. 2014. *Linked data based exploratory search*. Theses, Université Nice Sophia Antipolis.

K. O. McGraw and S. P. Wong. 1996. Forming inferences about some intraclass correlation coefficients. *Psychological Methods*, 1(1):30–46.

David Milne and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of AAAI 2008*.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814. Omnipress.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The PageRank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548.

Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 263–272, New York, NY, USA. ACM.

David Smith and Jennifer Campbell. 1999. The impact of students' approaches to essay writing on the quality of their essays. *Assessment & Evaluation in Higher Education*, 24(3):327.

Yang Song, Lu Zhang, and C. Lee Giles. 2011. Automatic tag recommendation algorithms for social recommender systems. *ACM Trans. Web*, 5(1):4:1–4:31.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Saúl Vargas and Pablo Castells. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 109–116, New York, NY, USA. ACM.

Ryen W. White and Resa A. Roth. 2009. *Exploratory Search: Beyond the Query-Response Paradigm*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers.

Hao Wu, Xiaohui Cui, Jun He, Bo Li, and Yijian Pei. 2014. On improving aggregate recommendation diversity and novelty in folksonomy-based social systems. *Personal Ubiquitous Comput.*, 18(8):1855–1869.

Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. 2010. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515.

A Genre-Aware Attention Model to Improve the Likability Prediction of Books

Suraj Maharjan⋆ **Manuel Montes-y-Gómez**† **Fabio A. González**‡ **Thamar Solorio**⋆
⋆Department of Computer Science, University of Houston
†Instituto Nacional de Astrofisica Optica y Electronica, Puebla, Mexico
‡Systems and Computer Engineering Department, Universidad Nacional de Colombia
`{smaharjan2,tsolorio}@uh.edu`
`mmontesg@ccc.inoep.mx, fagonzalezo@unal.edu.co`

Abstract

Likability prediction of books has many uses. Readers, writers, as well as the publishing industry, can all benefit from automatic book likability prediction systems. In order to make reliable decisions, these systems need to assimilate information from different aspects of a book in a sensible way. We propose a novel multimodal neural architecture that incorporates genre supervision to assign weights to individual feature types. Our proposed method is capable of dynamically tailoring weights given to feature types based on the characteristics of each book. Our architecture achieves competitive results and even outperforms state-of-the-art for this task.

1 Introduction

Book likability prediction is an important but challenging task. It can be a valuable resource for supporting buying decisions. The experience of choosing a book can be daunting for readers, considering the overwhelming number of books being published. On the other hand, being able to predict how a book will fare in the market has relevant economic value for the publishing industry in order to increase their revenue. The current process is guided by humans, but this is error-prone, very subjective, and a non-scalable process.

An alternative to the human-guided process is to design a reliable automatic system that predicts the likability of books. Such a system, we argue, must be able to take into account all of the many aspects involved in the eventual success of a book. These include not only the topic of the book and the writing style of the author, but in the case of creative writing, also include elements such as creativity, plot structure, and the flow of sentiments (Hall, 2012; Archer and Jockers, 2016; Maharjan et al., 2018; Kar et al., 2018). Other relevant aspects influencing readers' interest for a book could be the cover and the title of the book.

We believe that in addition to the ability to incorporate the different aspects, it is equally important to have a robust mechanism that gives higher weight to the most relevant aspects, while at the same time disregards the noisy or redundant aspects. Traditionally, this is achieved by searching through multiple feature combination experiments for an optimal combination of different feature types (Yang and Pedersen, 1997; Forman, 2003). The main problem with these methods is that they are time-consuming and too rigid. The resulting feature types are fixed for every document. In some books, the style of the author may contribute more than the specific topic, whereas the reverse may be true for other books. These methods lack the ability to dynamically assign weights to different features based on the characteristics of a particular test instance. Most likely, a more flexible scheme that adjusts feature weights based on the current book, can lead to better results.

This paper attempts to solve this problem by introducing a novel method that is capable of automatically combining information from different aspects and learning to weight them dynamically for each book in order to improve likability prediction. Our method also extends the attention model to incorporate domain specific information like the genre of books. As far as we know, we are the first to use genre supervision while computing attention weights and to use them in the field of feature importance. There are many potentially relevant aspects of books that make them likable by readers. Here we focus on different textual modalities, like the lexical, stylistic, syntactic, and neural representations, along with the visual modality from book covers. Our main contributions in this paper are as follows:

- We propose a novel neural architecture,

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3381–3391
Brussels, Belgium, October 31 - November 4, 2018.

which incorporates genre supervision for computing attention weights to learn the importance of hand-engineered and deep learning features coming from different modalities for predicting the likability of books.

- We show through our results that an adaptive combination of features with the genre-aware attention model performs better than strong baselines and also outperforms state-of-the-art.
- We present visualizations that increase interpretability of our results and also demonstrate the advantages of our model.

Along with these contributions, we also show that book cover images contain sufficient information by themselves to perform likability classification, although their contribution becomes negligible in the presence of strong textual features.

2 Methodology

We propose a model that we call Genre-Aware Attention model (GA), which dynamically weights features coming from different aspects of a book by using genre supervision. We first feed our textual and visual features through a non-linear layer to train higher feature representations. We then use our genre-aware attention model to compute appropriate weights for these feature representations. The motivation to add genre information comes from our previous work showing that adding genre classification as an auxiliary task to success prediction improved results (Maharjan et al., 2017). Moreover, it is also reasonable to expect that different genres should have different sets of features that are more relevant when trying to predict whether readers will like the book. For instance, in *Science Fiction*, the theme may be more relevant than say, in *Drama*, where the characters and their interactions or their struggles might be more relevant for likeability prediction.

2.1 Features

For our features, we build on the work by Maharjan et al. (2017) that provides a comprehensive exploration of different hand-crafted features and neural representations. They showed that a combination of writing density (WR) (distribution of word, character, sentences, and paragraphs), Book2Vec, and recurrent neural network representations (RNN) works well for books. Similar to their work, our textual features consist of word, character, and typed character n-grams (Sapkota et al., 2015), syntactic features, sentiment and sentic concepts and scores (SCS) (Cambria et al., 2014), style-related WR and readability (R), and neural representations learned using Word2Vec (Mikolov et al., 2013), Doc2Vec and RNN. We consider these categories of the textual features as different modalities or sources since they capture different aspects of a book and are generated by different processes. In addition to these features, we also add visual information extracted from the book covers. To extract the visual features, we rely on state-of-the-art visual feature extractor methods like VGG (Simonyan and Zisserman, 2014) and Resnet (He et al., 2016), initialized with the weights trained on the Imagenet dataset.

2.2 Genre-Aware Attention Model

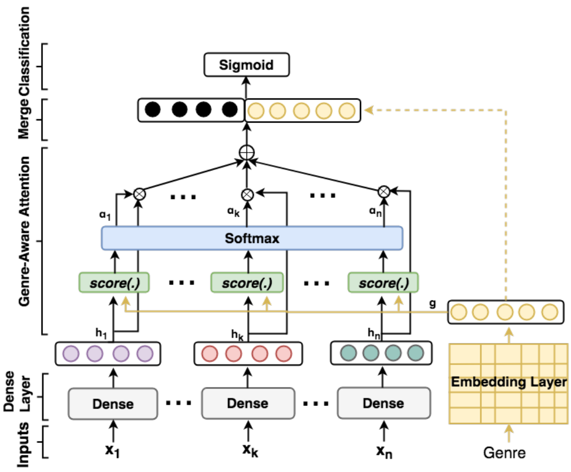

Figure 1: Genre-Aware Attention Model.

Figure 1 shows the overall architecture of our Genre-Aware Attention model. Let X be a collection of books. For a book $x \epsilon X$, let $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ be the feature representations from the different textual modalities and the visual modality. Since these features have different dimensions, we first pass them through a non-linear layer to project them into a space with the same dimension using Equation 1. This will allow us to perform a weighted average of features from different modalities according to their importance:

$$\mathbf{h_i} = selu(\mathbf{W_h x_i} + \mathbf{b_h}) \tag{1}$$

The source code and data for this paper can be downloaded from `https://github.com/sjmaharjan/genre_aware_attention` where i is the index of the modality whose feature representation is fed into the network, $\mathbf{W_h}$ is the weight matrix, $\mathbf{x_i}$ is the input feature vector for the ith modality, $\mathbf{b_h}$ is the bias, and $selu$ (Klambauer et al., 2017) is the activation function. All of these feature vectors from different modalities may not be equally important to the final representation and in turn to the likability prediction task. We use the genre-aware attention mechanism to learn the importance of each of these features towards our task and aggregate them to get the final representation. The final book representation $\mathbf{r}$ is the weighted sum of $\mathbf{h_i}$ vectors:

$$\mathbf{r} = \sum_i \alpha_i \mathbf{h_i} \tag{2}$$

where α_i are the weights measuring the importance of the different modalities. The GA model combines the genre vector $\mathbf{g} \epsilon \mathbb{R}^{d_g}$ (d_g being the dimension of the genre vector) while computing the α weights. The α_i weights are computed as follows:

$$\alpha_i = \frac{\exp(score(\mathbf{h_i}, \mathbf{g}))}{\sum_{i'} \exp(score(\mathbf{h_{i'}}, \mathbf{g}))} \tag{3}$$

and the $score(.)$ function is defined as:

$$score(\mathbf{h_i}, \mathbf{g}) = \mathbf{v}^T selu(\mathbf{W_a h_i} + \mathbf{W_g g} + \mathbf{b_a}) \tag{4}$$

where, $\mathbf{W_a}$ and $\mathbf{W_g}$ are the weight matrices and $\mathbf{v}$ is the weight vector. The addition of $\mathbf{W_g g}$ incorporates genre supervision. These parameters are shared across all modalities. This will prevent parameter explosion that is likely to occur when the number of modalities is high, which is the case for us. To further investigate the effect of the genre, we also experiment by concatenating the genre vector $\mathbf{g}$ to the final weighted averaged vectors from different modalities $\mathbf{r}$ to obtain $\mathbf{r};\mathbf{g}$. The dotted line from genre vector $\mathbf{g}$ represents this in Figure 1. We then use a non-linear layer with sigmoid activation to project the book representation (either $\mathbf{r}$ or the concatenation $\mathbf{r};\mathbf{g}$) to class probabilities.

$$\hat{p} = \sigma(\mathbf{W_c r} + \mathbf{b_c}) \tag{5}$$

where, $\mathbf{W_c}$ is the weight matrix and $\mathbf{b_c}$ is the bias vector. Finally, we train the network by minimizing the binary cross entropy loss using Adam (Kingma and Ba, 2015).

$$L = -\sum_i p_i \log \hat{p}_i \tag{6}$$

where, p_i and $\hat{p}_i$ are true labels and predictions, respectively.

3 Dataset

We experiment with the dataset collected by Maharjan et al. (2017). The dataset consists of books from eight different genres: *Detective Mystery*, *Drama*, *Fiction*, *Historical Fiction*, *Love Stories*, *Poetry*, *Science Fiction*, and *Short Stories*. These books have been reviewed by at least ten reviewers. Based on the average rating received by the books on Goodreads[1], they labeled the books into two categories: *Successful* and *Unsuccessful*. The collection has a total of 1,003 books. However, the dataset did not include book covers. We augmented this dataset by downloading the covers from Goodreads. Since this dataset only contains publicly available books, all of them were published over 100 years ago. Some of the books only had the title of the book on a plain background as their cover images on Goodreads. We manually searched for these books with Google Image Search and found the actual covers for most of them. However, even after an exhaustive search, we were unable to obtain proper covers for 21 books. We did not remove these books from the dataset for the sake of comparison with Maharjan et al. (2017).

4 Experiments and Results

We used the same train and test folds as used by Maharjan et al. (2017) for all of our experiments. The dataset consists of 349 books belonging to the *Unsuccessful* class and 654 books belonging to the *Successful* class. Since the dataset is imbalanced, they as well as we use weighted F1-score to evaluate the performance.

4.1 Baselines

The most naive baseline will be to predict the majority class for all test instances. This majority class baseline yields a weighted F1-score of 50.6% for the likability classification task. This baseline will help to understand whether our proposed model is actually learning from the data at all. Apart from this, we compare with the results from Maharjan et al. (2017) and we also define several other baselines to validate the superiority of our proposed model. All of the

---

[1] https://www.goodreads.com/

baseline methods are listed below:

Mah'17: The current state-of-the-art for this dataset by Maharjan et al. (2017). They have several results on various combinations of textual features.

Mah'17+Vis: This method is the extension of the Mah'17 method with the addition of visual features. Similar to them, we use the SVM classifier under two settings: Single-task (ST) and Multi-task (MT). In ST, we simply predict the likability of books. In MT, along with predicting likability, we also predict genre simultaneously. This experiment will allow us to make a direct comparison with Mah'17 regarding the effect of adding visual modalities.

Concatenation: Similar to GA, we first feed the features from different modalities through a non-linear layer each having the same number of neurons. We then concatenate them to obtain the final representation for a book. We send this representation to a sigmoid layer for success prediction.

Average Pooling: Instead of concatenation, we take an average of the features after passing them through the non-linear layer. This is also comparable to an attention model assigning equal weights to all modalities.

Attention: We use a multilayer perceptron to learn the appropriate weights for each of the features from different modalities. This method is similar to our proposed method, except that we do not use genre information for computing the attention weights. We compute the $score(.)$ as $\mathbf{v}^T selu(\mathbf{W_a h_i} + \mathbf{b_a})$, without the genre information. This experiment will help us understand the importance of genre in computing weights for the feature types.

Bilinear Model: We combine the non-linear transformed modalities $\mathbf{h_1}, \ldots, \mathbf{h_n}$ using a bilinear form $(\mathbf{h_i}^T \mathbf{W_b h_j} + \mathbf{b_b})$, where $\mathbf{W_b} \epsilon \mathbb{R}^{k \times d_{h_i} \times d_{h_j}}$ is the weight tensor and $\mathbf{b_b}$ is the bias vector (Socher et al., 2013; Laha and Raykar, 2016; Fukui et al., 2016; Gao et al., 2016). This operation gives us a k-dimensional vector. In the case of more than two modalities, we first create $\binom{n}{2}$ pairs of these modalities and combine each of them using a bilinear form. The final book vector is the concatenation of the resulting vectors from each of these pairs. Bilinear models are used in the visual question answering community to fuse visual and textual information (Fukui et al., 2016). This experiment will help us understand how our proposed model compares with other state-of-the-art multimodal approaches.

For all these models as well, we also performed additional experiments by concatenating the genre vector $\mathbf{g}$ with the final representations $\mathbf{r}$ obtained from each of these models to study the significance of including genre explicitly for likability prediction.

4.2 Experimental Settings

For the experiments involving the SVM classifier, we tuned the C hyper-parameter with values {1e-4, ..., 1e4} by performing three-fold grid search over the training data and then used the best hyper-parameters to train the final model. For the neural network experiments, we first separated 20% of the training data as a validation set and tuned dropout rates {0.2, 0.4, 0.5}, different weights initialization schemes {Glorot Uniform (Glorot and Bengio, 2010), LeCun Uniform (LeCun et al., 1998)}, learning rate with Adam {1e-4, ..., 1e-1}, number of hidden neurons in different layers {100, 200}, and batch size {1, 4, 8} with early stopping criteria. We initialized the genre embeddings with orthogonal vectors.

4.3 Results

Table 1 shows and compares our results with different baselines. We experimented with both low performing as well as high performing features and their combinations as found by Maharjan et al. (2017). We obtained the best weighted F1-score of 75.4% with our proposed GA+Genre concatenation model. This is 4.2% and 8.7% above the corresponding results reported by Mah'17 with their MT and ST settings, respectively. We also see a significant* improvement of 6.5% (over MT) and 22.2% (over ST) when using *RNN* features with our proposed method as compared to Mah'17. These results support the superiority of our method in learning high-quality book representations than Mah'17's state-of-the-art methods.

The results also show that it is beneficial to use at least some form of attention over just *Average Pooling*. This suggests that using all available features without regards to their individual contribution towards the task at hand can actually worsen the performance. Our proposed model is capable

---

*We used the McNemar significance test.

	Mah'17		Mah'17+Vis		Concatenation		Average Pooling		Bilinear		Attention		Genre Attention	
Features	**ST (SVM)**	**MT (SVM)**	**ST (SVM)**	**MT (SVM)**	**-**	**+ Genre**	**-**	**+ Genre**	**-**	**+ Genre**	**-**	**+ Genre**	**-**	**+ Genre**
Bigram	65.9	68.5	68.8	65.6	67.6	60.3	61.9	59.1	66.9	67.9	62.1	69.2	65.9	**70.2**
Clausal	50.6	55.8	59.0	53.9	57.0	58.2	60.7	60.1	55.9	**67.4**	50.6	66.5	50.7	**67.4**
Readability	50.6	63.4	60.8	58.8	54.3	59.3	59.3	58.3	60.8	60.8	53.6	65.7	52.1	**67.6**
RNN	52.9	68.6	68.9	68.9	71.9	69.3	67.6	68.4	69.5	71.9	71.4	70.7	**75.1**	73.7
Book2Vec (DBoW+DMM)	69.5	72.9	69.5	72.9	66.1	65.4	64.4	63.9	61.9	66.5	68.7	68.2	70.5	**73.4**
SCS,WR,Typed n-gram	72.0	71.0	69.2	68.8	67.9	60.8	63.1	66.7	67.6	69.2	69.2	69.1	72.5	**73.0**
WR,Book2Vec,RNN	70.1	73.5	66.1	71.6	70.6	69.0	66.7	70.1	70.1	66.4	70.0	70.5	**73.7**	73.1
All best handcrafted + RNN	66.7	71.2	69.3	72.2	68.1	67.4	66.6	65.7	50.6	50.6	69.8	70.6	70.2	**75.4**

Table 1: Weighted F1-scores(%) for different multimodal methods for books' likability classification task (ST=Single Task, MT=Multitask, SCS=Sentic Concepts and Scores, WR=Writing Density, RNN=Recurrent Neural Network Representations, + Genre= genre embedding $\mathbf{g}$ concatenated with the final book vector $\mathbf{r}$). Our baselines and proposed method include visual features as well.

of assigning importance to these features and the results clearly show that this works to our benefit. The results also demonstrate the added advantage of using genre supervision while computing feature weights. There is a considerable improvement in the performance over the *Attention* method after taking the genre information into account using our GA method. We suspect that the genre meta-information is helping to learn more specialized weights based on the genre of the books.

With the neural baseline methods like *Concatenation* and *Average pooling*, we do not always see improvement in performance after combining the genre information with the final book representation. Apart from these two, the combination of genre information does improve the results for other methods. The *Bilinear* and *Attention* methods seem to be able to utilize this information well. However, none of these methods are capable of doing better than our method. GA and GA+Genre concatenation models always achieve the best performance for all experiments. This also illustrates the latent power of our method to better exploit domain information like genre for performance improvement.

Another interesting finding is that with the addition of multiple modalities, the performance of Bilinear methods degrades to the majority class baseline (Table 1, last row). This may be due to parameter explosion with the increase in the number of modalities. However, our method is able to selectively weight the feature sources and discount the effect of redundant and irrelevant features to obtain the best performance, even with a larger number of modalities. In short, we see that our proposed method is able to cope with feature pollution and parameter explosion.

Next, we investigate the addition of visual information with the textual information for the likability prediction of books. Under the ST setting with SVMs, we see that the low performing textual features are benefited significantly by the addition of visual features, sometimes even outperforming the MT setting (Table 1, rows 1-4). However, the visual features are not able to contribute much when combined with strong textual features that were already performing well. On the other hand, for the MT setting, the performance decreases for most of the feature combinations with the addition of the visual modality. We suspect that book covers are not very helpful at predicting genre and thus the MT setting does not do well with additional visual features.

Visual Results: Our next set of experiments considers only the visual information for books' likability prediction. Even though we do believe that this current corpus might not be ideal for using cover features, we believe it is still interesting to explore whether the current book covers have sufficient information to perform likability classification with reasonable accuracy. We used VGG and Resnet to extract features from book cover images. We replaced the top layers by a dense layer of 256 neurons, and a classification layer (eight neurons with softmax for genre classification and one neuron with sigmoid activation for success classification). We also added a dropout layer in between the dense and the classification layer. The layers were initialized with weights trained on the Imagenet dataset.

Tasks	**Likability**		**Genre**	
Features	**ST (F1)**	**MT (F1)**	**ST (F1)**	**MT (F1)**
VGG	59.9	**61.8**	**24.7**	24.1
Resnet	58.7	60.0	24.6	24.0
VGG + SVM	58.8	57.7	25.4	19.6
Resnet + SVM	**59.5**	54.5	**25.9**	19.7

Table 2: Weighted F1-scores(%) for visual features for likability and genre classification of books with Single Task (ST) and Multitask (MT) settings.

Table 2 shows the results with only the visual features for likability and genre classification of books under the ST and MT settings. We obtain the highest weighted F1-score of 61.8% and 25.9% for likability and genre classification tasks, respectively. With the neural experimental setup, we get similar performance under the ST and the MT settings for both tasks. We also experimented with transferring the visual feature vectors to the SVM classifier under the ST and the MT settings. We saw a decrease in performance under the MT settings with both the VGG and Resnet features (Table 2, last two rows). This is the opposite of the Mah'17 results for the textual features as seen in Table 1. The reason behind this may be due to the fact that the textual features are better at both the likability and the genre classification tasks individually, whereas the visual features are not as good as the textual features for the genre classification task. Iwana et al. (2016) also concluded that genre classification with book covers is a difficult task as book covers have images with few visual features or ambiguous features.

These results also empirically verify the decrease in performance for the MT settings with the addition of visual features for likability prediction. Although these results are significantly lower (p<0.001*) than our best results, they are still better than the majority baseline (50.6% and 10.7% for success and genre classification tasks, respectively). These results support our hypothesis that the books' cover images correlate with the likability of books. Also, they dictate for the need of extracting other features that consider different aspects of books.

5 Attention Weights Visualization

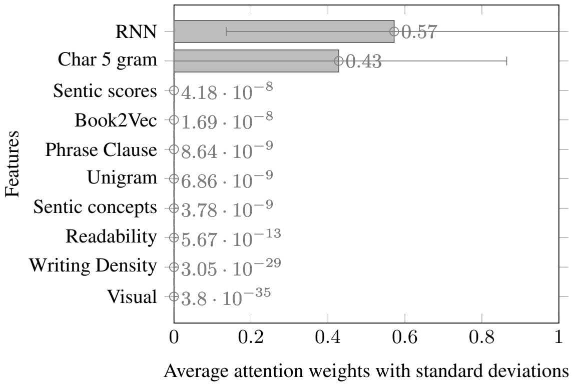

Figure 2: Feature importance for the feature combination: All best handcrafted, RNN, and visual (RNN = Recurrent Neural Networks).

Figure 2 shows the average attention weights given by the best model to the different feature types for the books in the test set. The purpose of this visualization is to understand which aspects of a book are deemed to be more important by the model. The figure shows that most of the weights are assigned to the *Char* 5-*gram* and the *RNN* representations. The results in Table 1 also support that *RNN* features are indeed one of the most important features. The contribution of the visual representations is negligible in the presence of strong textual features. The results in Table 1 also validate this finding. These two textual features also dominate over the other weaker textual features. In the same way, as for the visual features, we see negligible weights assigned to the other textual features as well. Our model seems to have learned that the *Char* 5-*gram* and the *RNN* features can cover the information given by the rest of the features. The *Char* 5-*gram* feature is capable of capturing the content, topic, and style of a text and as such might be able to cover the *Unigram* and *Sentic Concepts* features. Likewise, the *Book2Vec* features may be non-essential in the presence of the *RNN* representations. The model is reducing redundant information that does not aid the classification task and instead might just add noise.

In order to validate that features given the top weights by our model are indeed the best features for the task, we ran an experiment with only the *Char* 5-*gram* and the *RNN* features. We were able to obtain a weighted F1-score of 73.6% with just these two features. This score is close to the best score of 75.4%, showing that these features are indeed good features for the task. Also, note that our model was able to figure out this feature set automatically, while using traditional methods would have entailed performing multiple experiments ($2^n - 1$ experiments, where n is the number of feature types) which is often times not possible to do exhaustively. There is still an extra boost when using the whole feature set rather than using just the *Char* 5-*gram* and the *RNN* features. Since our method tailors the feature weights to each book and its genre as well, the boost likely comes from the presence of other visual and textual features, which at least for some books must be informative.

We just saw that only two out of all feature types are given most of the weights. However, the results in Table 1 show that even without these fea-

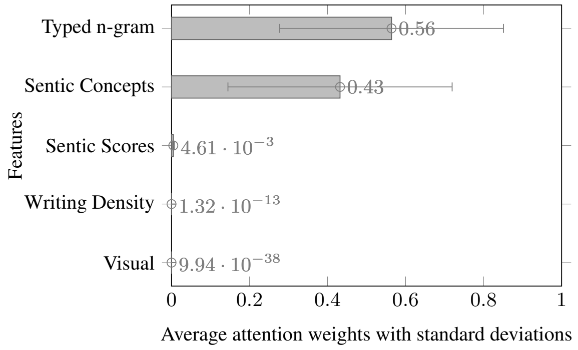

Figure 3: Feature importance for feature combination Sentic Concepts and Scores, Writing Density, Typed n-gram, and Visual.

tures, we are able to get good performances. To understand this, we analyze a model that does well without these two features. Figure 3 plots the average attention weights for a model with *Sentic Concepts and Scores*, *Writing Density*, *Typed* n-*grams*, and *Visual* features' combination. We see that the weights now shift to *Typed* n-*grams*, and *Sentic Concepts and Scores*. The topic and content captured through *Sentic concepts* and the style with *Typed* n-*grams* prove important. These features capture different aspects of books and are not strongly correlated with one another. Our model is capable of figuring out that in the absence of *Char* 5-*gram*, which encompasses all this information, these other features need to be made more prominent. We can also see that the model knows three different feature types to capture the same amount of information as captured by the two best ones from before.

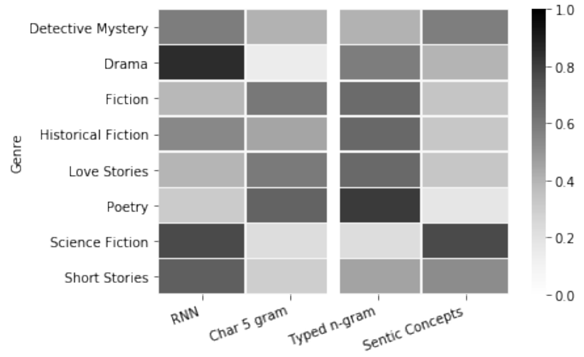

Figure 4: Average attention weights with respect to genre for the best features from two models.

Figure 4 further breaks down the attention weights by genre for *RNN* and *Char* 5-*gram*, and *Typed* n-*grams* and *Sentic Scores and Concepts*. From the figure, it is evident that different genres respond differently to each feature type. Comparing the two models, we see that *Char* 5-*gram* activates similarly to *Typed* n-*gram*, and *RNN* similarly to *Sentic concepts* for different genres.

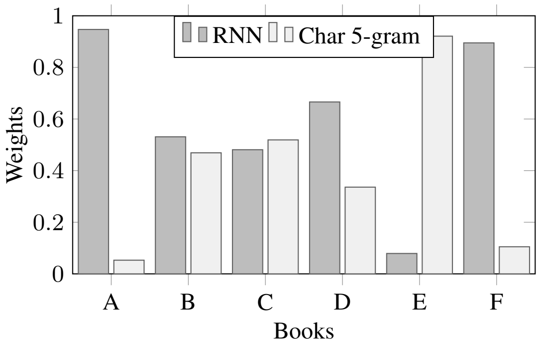

Figure 5: Feature importance as assigned by attention weights for two most important features for six different books: *A=The Count of Monte Cristo, B=The Scouts of the Valley, C=The Daughter of the Commandant, D=The Northern Light , E=The Great Secret, F=House of the Seven Gables.*

Figure 5 shows the feature importance for the *Char 5-gram* and *RNN* feature types for six different books having different attention weights for the two features. This validates our assumption that the model is able to dynamically learn and assign weights to different modalities, not only according to the genre but also according to the characteristics of each book. The high variance of attention weights for the top features in Figures 2 and 3 also support this claim. This gives an edge to our model and helps it excel over all other methods.

6 Error Analysis

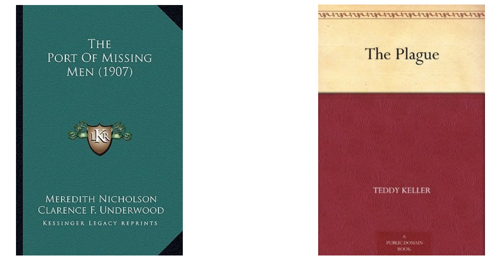

(a) The Port of Missing Men (b) The Plague

Figure 6: Books misclassified by visual features but correctly classified when textual features are added.

We took the books that were misclassified when we used the visual features only but were correctly predicted after the combination with the textual dimensions. As expected, we found that the books without proper covers were misclassified by visual features. But upon addition of other textual

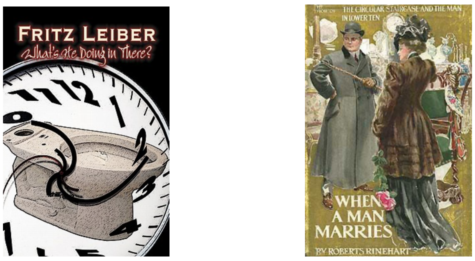

(a) What's He Doing in There? (b) When a Man Marries

Figure 7: Books misclassified by visual and text features' combination but correctly classified when only visual features are used.

features, they were correctly classified. Figure 6 shows the cover image of two of such books. The fact that the cover has no images with just plain background, and title, leaves little information for the visual modality. Similarly, we also analyzed the books that were correctly classified by visual features only and misclassified when textual features were added. Figure 7 shows two such books. Both the cover image and the title (present in the cover) of these two books seem to be interesting and are very likely to attract a reader's attention.

7 Related Work

Prior works have shown that stylistic traits to be useful features to predict success of books (Ashok et al., 2013; Underwood and Sellers, 2016; Maharjan et al., 2017). Ashok et al. (2013) used stylistic features extracted using the first 1K sentences from books to classify highly successful literature from less successful literature. van Cranenburgh and Bod (2017) used lexical and rich syntactic tree features to distinguish the degrees of high and less literary novels. Louis and Nenkova (2013) defined genre-specific and general features to predict the article quality in science journalism domain. Maharjan et al. (2017) compared their work with Ashok et al. (2013) and presented a new dataset for the book success prediction task. Their multitask approach with the combination of deep representations and hand-crafted features improved the classification results. Maharjan et al. (2018) also showed that modeling sequential flow of emotions across entire books improves likability prediction of books. Iwana et al. (2016) used neural networks to learn relationships between book covers and genre. They showed that book covers tend to have carefully designed color and tone, objects, and text. Our work relies on prior works' hand-engineered and deep learning features but differs in a way how these features are combined to produce a meaningful book representations.

The attention mechanism (Bahdanau et al., 2014) has been successfully applied in enhancing the document representation for several text classification (Zhang et al., 2016; Wang et al., 2016b), sentiment classification (Kar et al., 2017; Nguyen and Shirai, 2015; Wang et al., 2016a), question answering (Tan et al., 2015; Chen et al., 2016a; Hermann et al., 2015), named entity recognition (Bharadwaj et al., 2016; Aguilar et al., 2017), summarization (Rush et al., 2015), image-captioning (Xu et al., 2015) tasks. Zhang et al. (2017) used summary vectors and position vectors while computing the attention weights for the slot filling problem. Chen et al. (2016b) applied user preferences and product characteristics as attentions to words and sentences in reviews to learn the final representation for the sentences and reviews. They used these representation to do the sentiment classification task and showed that adding user information was much more effective in enhancing the document representations than the product information. Similar to their idea, we fuse the genre information while computing attention weights.

8 Conclusions and Future Work

We present a novel method to fuse the information coming from different modalities using a genre-aware attention mechanism to predict the likability of books. We showed that our proposed method outperforms strong baselines and state-of-the-art by learning to distinguish the important features from irrelevant or redundant ones. Other methods either suffered from feature pollution or parameter explosion and yielded low performance. Along with this, our results also showed that the book cover images by themselves also have sufficient information to perform success prediction. However, the difficulty in predicting genre from book covers decreased the performance in multi-task settings with additional visual features. We also used different visualizations to support our findings and improve interpretability of our model. As future work, we will extend the proposed method to include components that learn weights for individual feature elements and not only the entire feature type. This could likely result in higher quality multimodal representations.

Acknowledgments

This work was partially funded by the National Science Foundation under award 1462141. We would like to thank the anonymous reviewers, John Arevalo, Deepthi Mave, and Prasha Shrestha for reviewing the paper and providing helpful comments and suggestions.

References

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 148–153, Copenhagen, Denmark. Association for Computational Linguistics.

Jodie Archer and Matthew L Jockers. 2016. *The Bestseller Code: Anatomy of the Blockbuster Novel*. St. Martin's Press.

Vikas Ashok, Song Feng, and Yejin Choi. 2013. Success with style: Using writing style to predict the success of novels. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1764, Seattle, Washington, USA. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1462–1472, Austin, Texas. Association for Computational Linguistics.

Erik Cambria, Daniel Olsher, and Dheeraj Rajagopal. 2014. Senticnet 3: A common and common-sense knowledge base for cognition-driven sentiment analysis. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1515–1521. AAAI Press.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016a. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany. Association for Computational Linguistics.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016b. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659, Austin, Texas. Association for Computational Linguistics.

Andreas van Cranenburgh and Rens Bod. 2017. A data-oriented model of literary language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1228–1238, Valencia, Spain. Association for Computational Linguistics.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 457–468, Austin, Texas. Association for Computational Linguistics.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. 2016. Compact bilinear pooling. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

James W Hall. 2012. *Hit Lit: Cracking the Code of the Twentieth Century's Biggest Bestsellers*. Random House.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Brian Kenji Iwana, Syed Tahseen Raza Rizvi, Sheraz Ahmed, Andreas Dengel, and Seiichi Uchida. 2016. Judging a book by its cover. *arXiv preprint arXiv:1610.09204*.

Sudipta Kar, Suraj Maharjan, and Thamar Solorio. 2017. Ritual-uh at semeval-2017 task 5: Sentiment analysis on financial data using neural networks. In *Proceedings of the 11th International Workshop on*

Semantic Evaluation (SemEval-2017), pages 877–882, Vancouver, Canada. Association for Computational Linguistics.

Sudipta Kar, Suraj Maharjan, and Thamar Solorio. 2018. Folksonomication: Predicting tags for movies from plot synopses using emotion flow encoded neural network. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2879–2891, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. *CoRR*, abs/1706.02515.

Anirban Laha and Vikas Raykar. 2016. An empirical evaluation of various deep learning architectures for bi-sequence classification tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2762–2773, Osaka, Japan. The COLING 2016 Organizing Committee.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. 1998. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50, London, UK, UK. Springer-Verlag.

Annie Louis and Ani Nenkova. 2013. What makes writing great? first experiments on article quality prediction in the science journalism domain. *Transactions of the Association for Computational Linguistics*, 1.

Suraj Maharjan, John Arevalo, Manuel Montes, Fabio A. González, and Thamar Solorio. 2017. A multi-task approach to predict likability of books. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1217–1227, Valencia, Spain. Association for Computational Linguistics.

Suraj Maharjan, Sudipta Kar, Manuel Montes, Fabio A. Gonzalez, and Thamar Solorio. 2018. Letting emotions flow: Success prediction by modeling the flow of emotions in books. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 259–265, New Orleans, Louisiana. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *International Conference on Learning Representations (ICLR), Workshop*.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2509–2514, Lisbon, Portugal. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–102, Denver, Colorado. Association for Computational Linguistics.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Ted Underwood and Jordan Sellers. 2016. The longue durée of literary prestige. *Modern Language Quarterly*, 77(3):321–344.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016a. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

Zhongqing Wang, Yue Zhang, Sophia Lee, Shoushan Li, and Guodong Zhou. 2016b. A bilingual attention network for code-switched emotion prediction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1624–1634, Osaka, Japan. The COLING 2016 Organizing Committee.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81.

Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2449–2460, Osaka, Japan. The COLING 2016 Organizing Committee.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark. Association for Computational Linguistics.

Thread Popularity Prediction and Tracking with a Permutation-invariant Model

Hou Pong Chan^{1,2} **and Irwin King**^{1,2}
^{1}Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
^{2}Shenzhen Key Laboratory of Rich Media Big Data Analytics and Application,
Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China
{hpchan, king}@cse.cuhk.edu.hk

Abstract

The task of thread popularity prediction and tracking aims to recommend a few popular comments to subscribed users when a batch of new comments arrive in a discussion thread. This task has been formulated as a reinforcement learning problem, in which the reward of the agent is the sum of positive responses received by the recommended comments. In this work, we propose a novel approach to tackle this problem. First, we propose a deep neural network architecture to model the expected cumulative reward (Q-value) of a recommendation (action). Unlike the state-of-the-art approach, which treats an action as a sequence, our model uses an attention mechanism to integrate information from a set of comments. Thus, the prediction of Q-value is invariant to the permutation of the comments, which leads to a more consistent agent behavior. Second, we employ a greedy procedure to approximate the action that maximizes the predicted Q-value from a combinatorial action space. Different from the state-of-the-art approach, this procedure does not require an additional pre-trained model to generate candidate actions. Experiments on five real-world datasets show that our approach outperforms the state-of-the-art.

1 Introduction

Online discussion forums allow people to join in-depth conversations about different topics in form of threads. Each thread corresponds to one conversation, which is initiated by a post and users respond to it with comments. In addition, a comment can be further replied by another comment, forming a discussion tree. Users who are interested in a particular thread will subscribe to it. After the subscription, users will receive a notification when a new comment arrives in that thread. However, the speed of content generation in a well-known discussion forum is breakneck. For instance, in Reddit[1], there were more than 900 million comments posted in 2017 (Reddit, 2017). Hence, merely pushing every new comment to the subscribers leads to a poor user experience. Motivated by this issue, He et al. (2016c) proposed the task of thread *popularity prediction* and *tracking*. When N new comments arrive in a thread, the system performs one step of recommendation by pushing K comments to the subscribers. We want to maximize the sum of popularities of the recommended comments over all recommendation steps. The popularity of a comment is measured by the number of positive reactions it received, e.g., the rating. With the assumption that a user needs to know the prior context in order to understand a comment, the system can only recommend new comments that are in the subtrees of previously recommended comments. Thus, the selection of comments at the current recommendation step will affect the comments that we can choose in the future recommendation steps.

To incorporate the long-term consequences of recommendations, the task of thread popularity prediction and tracking has been formulated as a reinforcement learning problem, in which an agent selects an action (a set of K comments) according to its current state (previous recommended comments), with the goal of maximizing the cumulative reward (total popularities of the recommended comments over all recommendation steps). The optimal action of the agent at each step is the action that maximizes the Q-function, $Q(s, a)$, which denotes the long-term reward of choosing action a in state s. In practice, we learn this Q-function using a parametric function, $Q(s, a; \theta)$, where θ is the model parameter vector. Thus, the predicted optimal action of the agent is the action

---

[1] https://www.reddit.com/

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3392–3401
Brussels, Belgium, October 31 - November 4, 2018.

that maximizes $Q(s, a; \theta)$.

This reinforcement learning problem has two main challenges. First, we need to develop a parametric model, $Q(s, a; \theta)$, to approximate the Q-function. Second, finding the action that maximizes $Q(s, a; \theta)$ requires the prediction of all $\binom{N}{K}$ possible actions, which is intractable. Thus, we need a procedure to approximate the predicted optimal action from a combinatorial action space.

To address the first challenge, He et al. (2016c) proposed a neural network model, DRRN-BiLSTM, to approximate the Q-function. In this model, a bi-directional long short-term memory (LSTM) (Graves and Schmidhuber, 2005) is used to encode the set of K comments in an action. To address the second challenge, they proposed the two-stage Q-learning procedure to approximate the predicted optimal action (He et al., 2017). In this procedure, the agent uses a pre-trained and less-sophisticated model to rank all possible actions, then it uses the DRRN-BiLSTM to re-rank the top-M actions and selects the best one. However, this approach has two limitations. First of all, bi-directional LSTM is a sequence model, which treats the set of K comments in an action as a sequence. Although they tried to fix this problem by feeding randomly-permuted comments to the model, a different permutation of the same set of comments leads to a different Q-value prediction. Thus, the agent may not consistently select the predicted optimal action. Secondly, the two-stage Q-learning procedure requires an additional pre-trained model to generate candidate actions.

Our work addresses these two limitations as follows. We propose a novel neural network model, DRRN-Attention, to approximate the Q-function. In our model, we use an attention mechanism (Bahdanau et al., 2014) to integrate the information from a set of comment into an action embedding vector. In a nutshell, the attention mechanism outputs a weighted sum of the comment representations, where the weights are learned by a subnetwork to indicate the importance of each comment. Thus, the action embedding is invariant to the permutation of the comments, which leads to a permutation invariant Q-value prediction. Next, we employ a greedy procedure to approximate the action that maximizes $Q(s, a; \theta)$. This procedure only requires the prediction of $O(NK)$ actions, which is significantly lower than $\binom{N}{K}$. Moreover, it does not require an additional pre-trained model to generate candidate actions.

In our experiments, we evaluate the performance of our DRRN-Attention model and the greedy approximation procedure against the baselines on five real-world datasets. Experimental results demonstrate that our approach beats the baselines on four of the datasets and achieves a competitive performance on one of the datasets. Furthermore, we analyze the performance of our approach across four action sizes ($K = 2, 3, 4, 5$). Our approach consistently achieves a higher cumulative reward than the baselines across all these action sizes.

We summarize our contributions as follow: (1) a new neural network architecture to model the Q-value of the agent which is invariant to the permutation of sub-actions; (2) a greedy procedure for the agent to select an action from the combinatorial action space without an additional pre-trained model; and (3) the new state-of-the-art performances on five real-world datasets.

2 Related Work

2.1 Reinforcement Learning in Text-based Tasks

Reinforcement learning has been widely applied in various text-based tasks. There are several articles in literature studying the tasks of mapping instruction manuals to a sequence of commands, such as game commands (Branavan et al., 2011), software commands (Branavan et al., 2010), and navigation directions (Vogel and Jurafsky, 2010). In the task of text-based game, an agent selects a textual command from a set of feasible commands at every time step. Narasimhan et al. (2016) considered a special case that all the textual commands have a fixed structure, while He et al. (2016b) and Chen et al. (2017) considered another case that all commands are free text.

In the task of thread popularity prediction and tracking, the agent selects a set of K comments from N available comments at every time step, where each comment is a free text. He et al. (2016c) proposed two different approaches to tackle this task. In their first approach, the agent uses the Deep Reinforcement Relevance Network (DRRN) (He et al., 2016b) to model the Q-function of selecting a comment. In their second approach, the agent uses the DRRN-BiLSTM (He et al., 2016c) to model the Q-function of an action. To due with the combinatorial action space, the agent uses uniform sampling to generate a set of M candidate actions. To improve this random sampling scheme, they proposed the two-stage Q-learning procedure in their later work (He et al., 2017), which used a pre-trained model to generate M candidate actions. Their experimental results showed that using DRRN-BiLSTM with two-stage Q-learning procedure outperforms all other existing methods. The difference between our model and DRRN-BiLSTM is that we use attention to encode a set of comments rather than using a bi-directional LSTM. Besides, the greedy procedure in our approach does not require any extra pre-trained model. He et al. (2017) also considered a special case that the agent can access an external knowledge source to augment the state representation. This setting is orthogonal to this work since we focus on the action encoding and the approximation of predicted optimal action.

One line of research focused on the integration of sequence-to-sequence (SEQ2SEQ) model (Sutskever et al., 2014) and reinforcement learning framework, examples including dialogue generation (Dhingra et al., 2017; Li et al., 2016; Su et al., 2016), question answering system (Buck et al., 2017), and machine translation (He et al., 2016a). In these tasks, the agent selects an action by generating a free text using a SEQ2SEQ model.

2.2 Deep Learning on Sets

Most of the deep learning models on sets employed attention to integrate information from a set of input. This idea was first introduced in the read-process-and-write network (Vinyals et al., 2016), which uses a process module to perform multiple steps of attention over a set of vectors to obtain a permutation-invariant embedding. Our work adapts this idea to aggregate a set of comment embedding vectors. In the domain of graph learning, several models (Sukhbaatar et al., 2015; Zhang et al., 2017) learn an embedding of a node by attending over its neighboring nodes. All of the above models can be interpreted as a special case of memory network (Weston et al., 2015; Sukhbaatar et al., 2015; Zhang et al., 2017), if we view the set of feature vectors as external memory. Max-pooling is another promising technique for the problem of learning on sets. Qi et al. (2017) used max-pooling to aggregate the feature vectors of a set of 3D geometry points. Recently, Zaheer et al. (2017) derived the necessary and sufficient conditions for a neural network layer to be permutation invariant.

2.3 Popularity Prediction

Another related line of research is popularity prediction problem in a supervised learning setting. Yano and Smith (2010) used the LDA topic model (Blei et al., 2003) to predict the number of comments of a blog post in a political blog. There are also several studies focused on the task of predicting the number of reshares on Facebook (Cheng et al., 2014) and the number of retweets in tweeter based on the text content (Tan et al., 2014; Hong et al., 2011). Recently, Cheng et al. (2017) proposed a neural network model to learn comment embeddings for the task of community endorsement prediction in a supervised learning setting.

3 Preliminary

3.1 Discussion Tree

A discussion thread in an online forum can be represented as a tree. Each node in the tree stores a free text. The root node represents the post of the thread and each non-root node represents a comment of the thread. There is a directed edge from node u to node v if and only if a comment (or post) u is replied by comment v. This tree keeps growing as new comments are submitted to the thread.

3.2 Problem Definition

The task of thread popularity prediction and tracking is formally defined as a reinforcement learning problem. We use M_t to denote the set of comments that are being tracked at time t. Given a discussion thread, we start an episode as follows. First, we initialize M_1 to be the post of a thread. Then, at each time step t the agent performs the following operations:

- Read the current state s_t, which is all the previously tracked comments $\{M_1, \ldots, M_t\}$.
- Read N new comments, $c_t = \{c_{t,1}, \ldots, c_{t,N}\}$, in the subtree of M_t.
- Select a set of K comments from c_t to recommend, $a_t = \{c_t^1, \ldots, c_t^K\}$, where $c_t^i \in c_t$ for $i = 1, \ldots, K$ and $c_t^i \neq c_t^j$ for $i \neq j$.

- Receive a reward, $r_{t+1} = \sum_{i=1}^{K} \eta_{c_t^i}$, where $\eta_{c_t^i}$ is the number of positive reactions received by comment c_t^i.
- Track the set of recommended comments in the next time step, $M_{t+1} = a_t$.

The episode terminates when no more new comments appear in the subtree of M_t. The goal of the agent is to maximize the cumulative reward.

3.3 Q-function

The state-action value function (or Q-function), $Q(s, a)$, is defined as the expected cumulative reward starting from state s and taking action a. More formally, $Q(s, a) = \mathbb{E}[\sum_{l=0}^{+\infty} \gamma^l r_{t+1+l} | s_t = s, a_t = a]$, where $\gamma \in (0, 1]$ is a discount factor for future rewards. Since the goal of the agent is to maximize the cumulative reward, the optimal action for each state is the action that achieves the highest Q-value. Thus, the Q-function is associated with an optimal policy: in every state, the agent selects the action that maximizes the Q-function, i.e., $a_t = \operatorname{argmax}_a Q(s_t, a)$, $\forall t$. Since this Q-function is unknown to the agent, we approximate the Q-function using a parametric model, $Q(s, a; \theta)$, and update the parameters θ using received rewards.

3.4 Exploration-Exploitation Trade-off

The agent needs to balance the exploration-exploitation trade-off when selecting an action. On one hand, the agent can choose the action with the highest estimated Q-value to exploit its current knowledge of the Q-function. On the other hand, the agent can choose a non-greedy action to get more information about the Q-value of other actions. The balance between exploration and exploitation can be achieved by using the ϵ-greedy policy, in which the agent selects a random action with probability ϵ, and selects a greedy action with probability $1 - \epsilon$. Note that the term "greedy" in the ϵ-greedy policy means that the agent selects the action that is predicted to be optimal, i.e., select $a_t = \operatorname{argmax}_a Q(s_t, a; \theta)$. It does not refer to the greedy procedure, which is used to approximate the predicted optimal action in a combinatorial action space.

4 DRRN-Attention Model

In this work, we propose a new deep neural network model, named DRRN-Attention, to approximate the Q-function for the task of thread popularity prediction and tracking. The input to our model is a state, s_t, and an action, $a_t = \{c_t^1, \ldots, c_t^K\}$, as defined in Section 3.2. The output is the prediction of Q-value, i.e., $Q(s_t, a_t; \theta) \in \mathbb{R}$. Figure 1 illustrates the overall architecture of DRRN-Attention. We divide our model into three modules as follows.

4.1 Text Representation Module

The text representation module reads s_t and $a_t = \{c_t^1, \ldots, c_t^K\}$. We first convert $s_t, c_t^1, \ldots, c_t^K$ into bag-of-words (BOW) representations, $b_{s_t}, b_{c_t^1}, \ldots, b_{c_t^K}$. Then, we use a 2-layer feedforward neural network to embed b_{s_t} into a d-dimensional state embedding vector, $m_{s_t} \in \mathbb{R}^d$. After that, we use another 2-layer feedforward neural network to embed $b_{c_t^i}$ into a d-dimensional comment embedding vector, $m_{c_t^i} \in \mathbb{R}^d$, for $i = 1, \ldots, K$. This module outputs m_{s_t} and $\{m_{c_t^1}, \ldots, m_{c_t^K}\}$.

4.2 Set Embedding Module

The input to this module is a set of d-dimensional comment embeddings, $\{m_{c_t^1}, \ldots, m_{c_t^K}\}$. The output is an action embedding vector, $m_{a_t} \in \mathbb{R}^{h+d}$, which is invariant to the ordering of comment embeddings. The module consists of a single-layer LSTM with a hidden size of h, and a shared attention mechanism, $f : \mathbb{R}^h \times \mathbb{R}^d \to \mathbb{R}$. The initial hidden state, $q_0 \in \mathbb{R}^h$, of the LSTM is a trainable vector. Inspired by (Vinyals et al., 2016), we perform L steps of computations over the comment embedding vectors. More specifically, at each step of computation $l = 0, 1, \ldots, L - 1$:

- The query vector, $q_l \in \mathbb{R}^h$, is the current hidden state of the LSTM.
- Apply the attention mechanism to compute an attention coefficient, $e_{i,l}$, between the query, q_l, and a comment embedding, $m_{c_t^i}$, for $i = 1, \ldots, K$. In general, this framework is agnostic to the underlying attention mechanism. In this work, we closely follow the attentional setup in (Bahdanau et al., 2014), as shown in the following equation.

$$e_{i,l} = v^T \tanh(\mathbf{W_e} m_{c_t^i} + \mathbf{U_e} q_l), \qquad (1)$$

where $v \in \mathbb{R}^{h'}$, $\mathbf{W}_e \in \mathbb{R}^{h' \times d}$, and $\mathbf{U}_e \in \mathbb{R}^{h' \times h}$.

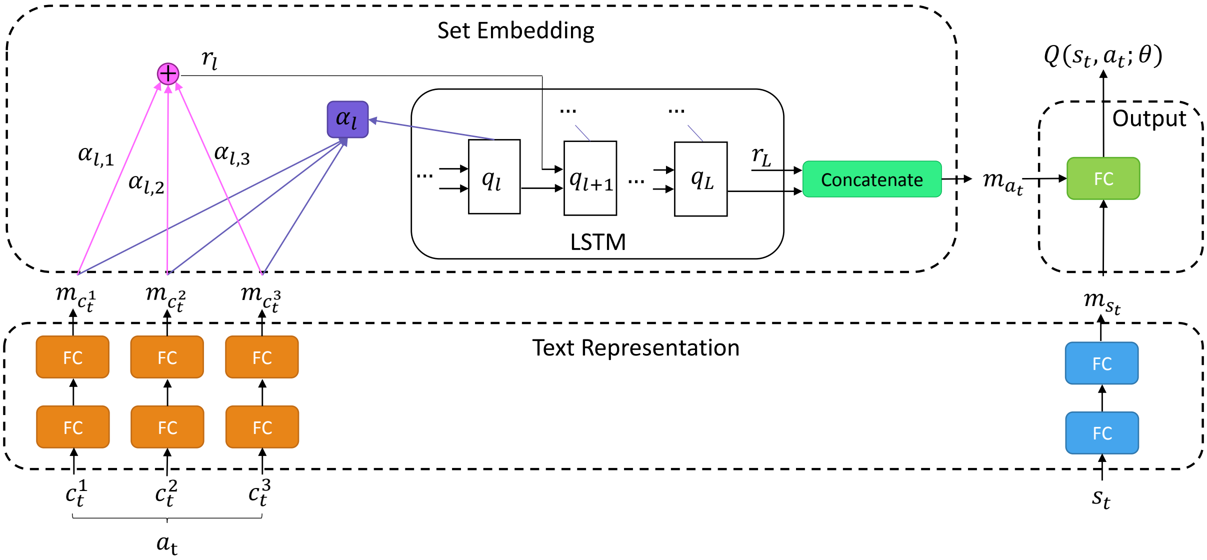

Figure 1: Architecture of DRRN-Attention model. The text representation module first converts state s_t and each comment c_t^i in action a_t into embedding vectors. Then, the comment embedding vectors are passed to the set embedding module to learn an action embedding vector. Finally, the state embedding and the action embedding are passed to the output module to output a prediction of Q-value.

- Apply softmax function to normalize the attention coefficients,

$$\alpha_{i,l} = \frac{\exp(e_{i,l})}{\sum_{j=1}^{K} \exp(e_{j,l})}. \tag{2}$$

- Use the normalized attention coefficients to compute a weighted sum of the comment embedding vectors, as the readout in this computation step,

$$r_l = \sum_{i=1}^{K} \alpha_{i,l} m_{c_t^i}. \tag{3}$$

- The LSTM takes q_l and r_l as input and computes the next hidden state, q_{l+1},

$$q_{l+1} = \text{LSTM}([q_l, r_l]). \tag{4}$$

Note that swapping any two comment embedding vectors, $m_{c_t^i}$ and $m_{c_t^j}$, will not affect the query vector q_l as well as the attention readout r_l. After L steps of computation, this module concatenates q_L and r_L to yield the final output action embedding, $m_{a_t} = [q_L, r_L] \in \mathbb{R}^{h+d}$.

4.3 Output Module

The input to this module is a state embedding vector, $m_{s_t} \in \mathbb{R}^d$, and an action embedding vector, $m_{a_t} \in \mathbb{R}^{h+d}$. We simply concatenate m_{s_t} and m_{a_t} and pass them through a fully-connected layer. The output is $Q(s_t, a_t; \theta) \in \mathbb{R}$, which is the prediction of $Q(s_t, a_t)$.

Algorithm 1 Greedy$(s_t, c_t, Q(\cdot, \cdot; \theta), K)$

```
$a = \emptyset$
for $i = 1 \to K$ do
    $c^* = \text{argmax}_{c \in c_t \setminus a} Q(s_t, a \cup c; \theta)$
    $a = a \cup c^*$
return $a$
```

5 Greedy Procedure

The next challenge that we need to address is to approximate the predicted optimal action in a combinatorial action space. Finding the predicted optimal action, $\text{argmax}_a Q(s_t, a; \theta)$, is intractable since it requires the prediction of all $\binom{N}{K}$ actions. In this work, we use a greedy procedure to compute an approximation. The complete procedure is shown in Algorithm 1. We start from an empty action, $a_t = \emptyset$, and then iteratively adds into a_t the comments that leads to the largest increase in $Q(s_t, a_t; \theta)$, until $|a_t| = K$. The procedure consists of K iterations. In each iteration i, we need to predict the Q-value for $n - i$ actions. In total, it only requires the prediction of $O(NK)$ actions, which is tractable. The advantage of this procedure over existing methods is that it does not require another pre-trained model to generate candidate actions.

6 Parameter Learning

We use the deep Q-learning algorithm (Mnih et al., 2015), which is a variant of the traditional Q-learning algorithm (Watkins and Dayan, 1992), to learn the model parameters of $Q(s, a; \theta)$ from the received rewards. The complete training procedure is shown in Algorithm 2 in the Appendix. The network parameters θ are first initialized arbitrarily. At each time step t, the agent selects an action a_t according to the ϵ-greedy policy, receives a reward r_{t+1}, and transits to the next state s_{t+1}. Thus, it yields a transition tuple, $\zeta_t = (s_t, a_t, r_{t+1}, s_{t+1})$. Instead of using the current transition tuple, ζ_t, to update the parameters, we first store ζ_t into an experience memory, D. This experience memory has a limited capacity, $|D|$, and the stored transition tuples are rewritten in a first-in-first-out manner. Then, we sample mini-batches of transition tuples (s, a, r, s') from D uniformly at random. Using the sampled transition tuples, we perform a step of stochastic gradient descent to minimize the following loss function,

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim U(D)}[(y - Q(s, a; \theta))^2], \quad (5)$$

where $y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$ is the Q-learning target, θ^- are the network parameters of the Q-learning target. We update θ^- to match the network parameters, θ, after every F time steps, where F is a hyperparameter.

7 Experimental Setup

In the experiments, we analyze the performances of different neural network models and different approximation procedures. First, their performances are evaluated on five real-world datasets, with a fix action size K. Then, we evaluate their performances with different action sizes, on one dataset. For each experiment setting, we do the following comparative analysis:

- Compare the performance of our DRRN-Attention model with the baseline models using different approximation procedures.
- Compare the performance of the greedy procedure with the baseline approximation procedures using different neural network models.
- Find the approach (combination of neural network model and approximation procedure) that achieves the best performance.

Finally, we conduct a case study to better illustrate the difference between our DRRN-Attention model and the DRRN-BiLSTM baseline.

Subreddit	**# Posts**	**# Comments**
askscience	0.94k	0.15M
askmen	4.45k	0.94M
todayilearned	9.44k	4.65M
worldnews	8.00k	4.28M
nfl	11.73k	5.72M

Table 1: Basic statics of discussion threads data from five subreddits.

7.1 Datasets

All the experiments are conducted on discussion thread data from the Reddit discussion forum. In Reddit, threads are grouped into different categories, called subreddits, according to different discussion themes. Registered users are allowed to give up-votes or down-votes to a comment, these votes are then aggregated to compute a *karma score* for the comment. We use it as the reward for recommending that comment. Using the post IDs provided by He et al. (2016c), we crawl five datasets from five different subreddits respectively, including askscience, askmen, todayilearned, worldnews, and nfl. These subreddits cover a wide range of discussion topics and language styles. The basic statistics of the datasets are presented in Table 1. Since some of the posts and comments were deleted by Reddit, we remove all the deleted posts and comments from the datasets. Thus, the statistics of our datasets are different from that in He et al. (2016c). For each dataset, we use the simulator provided by He et al. (2016c) to partition 90% of the data as a training set, and 10% of the data as a testing set.

7.2 Evaluation

The evaluation metric is the cumulative reward per episode averaged over 1,000 episodes (He et al., 2016c). For each setting, we evaluate an agent as follows. First, we train the agent on the training set using Algorithm 2 in the Appendix for 3,500 episodes. Then, we test the agent using the testing set for 1,000 episodes and choose every action according to the ϵ-greedy policy, but the agent cannot use the received rewards to update the model parameters. We repeat the testing for five repetitions and report the mean and the standard deviation of the evaluation metric. Throughout the training and testing, we fix $\epsilon = 0.1$.

Models	Procedure	Askscience	Askmen	Todayilearned	Worldnews	NFL
DRRN-BiLSTM	Random	546.9±21.8	183.1±7.2	654.9±27.3	427.4±21.9	438.7±16.1
	Two-stage	643.2±28.2	184.3±9.2	659.0±25.6	424.0±14.0	455.9±18.7
	Greedy	672.2±22.2	190.2±11.5	665.7±11.4	428.8±21.0	459.7±9.3
DRRN-Mean	Random	594.5±21.9	184.7±6.9	649.6±21.7	413.9±11.7	434.1±22.9
	Two-stage	581.1±13.0	183.3±5.6	670.4±29.7	426.7±24.2	441.9±20.3
	Greedy	732.3±25.7	194.2±7.8	680.0±31.0	435.4±18.5	452.3±22.6
Our model	Random	648.6±19.6	186.1±8.2	670.5±26.5	429.4±20.8	452.5±17.9
	Two-stage	685.6±24.6	184.7±7.5	672.2±37.8	426.1±18.2	**460.3**±**15.7**
	Greedy	**757.4**±**10.9**	**210.6**±**11.3**	**689.5**±**13.3**	**436.0**±**21.9**	454.2±12.0

Table 2: Comparison of average episodic reward on different datasets.

7.3 Baselines

Our DRRN-Attention model is compared with two baselines. The first one is DRRN-BiLSTM, which is the current state-of-the-art model to approximate the Q-value for this task (He et al., 2016c). We modify the DRRN-BiLSTM model by replacing the Bi-directional LSTM with a mean operator and call this new model DRRN-Mean. This DRRN-mean is used as the second baseline model. In addition, we compare the greedy procedure with two baseline approximation procedures. The first is random sampling procedure in (He et al., 2016c). The second is the two-stage Q-learning procedure in (He et al., 2017), which is the state-of-the-art approximation procedure for this task.

7.4 Implementation Details

In preprocessing, we remove all punctuations and lowercase all alphabetic characters. To construct the bag-of-words representations, we use the dictionary provided by He et al. (2016c). This dictionary contains the most frequent 5,000 words in the data. All model parameters are initialized by a uniform distribution within the interval $[-0.1, 0.1]$. In our DRRN-Attention model, we set the comment embedding size d to 16, the hidden size of the LSTM h to 16, the hidden size of the attention mechanism h' to 16, and the steps of attention L to 2. In the text embedding module of DRRN-Attention, each fully-connected layer has a hidden size of 16. In the baseline models, we set the hidden size of the bidirectional LSTM to 20, the comment embedding size to 20. The text embedding module of the baselines has two layers and each layer has a hidden size of 20. For the deep Q-learning algorithm, we set $F = 1000$. We update the model parameters using stochastic gradient descent with RMSprop (Tieleman and Hinton, 2012). We set different initial learning rates for different datasets (askscience: 0.00001; askmen: 0.00008; todayilearned, worldnews, and nfl: 0.00002). The mini batch size is 100. All the above hyperparameters are tuned by five-fold cross-validation. We also found that the model performances tuned by five-fold cross-validation are similar to that tuned by the testing set. We set the remaining hyperparameters according to (He et al., 2017). The memory size $|D|$ is set to 10000. The discount factor γ is set to 0.9. The candidate size m of the baseline approximation procedures is set to 10.

8 Experimental Results

8.1 Agent Performances on Various Datasets

In this section, the performances of different neural network models and approximation procedures are evaluated on five datasets with $N = 10$, $K = 3$. The results are shown in Table 2. We analyze the performances of our DRRN-Attention model in each approximation procedure. With the random sampling procedure, our model achieves a higher cumulative reward than the baseline models across all datasets. When using the two-stage Q-learning procedure, or the greedy procedure, our model outperforms the baselines on four of the datasets; its performance is competitive to the baselines in the remaining dataset. Next, we analyze the performances of the greedy procedure in each neural network model. When using the DRRN-BiLSTM model or the DRRN-Mean model to parameterize the Q-function, the greedy procedure achieves a higher cumulative reward than other two baseline procedures across all datasets. When using the DRRN-Attention model, the greedy procedure outperforms the baseline procedures in four of the datasets. In sum up, us-

Models	Procedure	K=2	K=3	K=4	K=5
DRRN-BiLSTM	Random	431.6±18.8	546.9±21.8	743.7±15.9	899.5±53.9
	Two-stage	484.3±10.6	643.2±28.2	765.7±32.8	928.6±20.5
	Greedy	467.8±24.5	672.2±22.2	772.9±28.2	929.2±18.6
Our model	Random	482.7±18.1	648.6±19.6	806.0±13.5	941.2±7.3
	Two-stage	537.5±26.1	685.6±24.6	772.8±22.8	903.5±22.4
	Greedy	**545.2±27.3**	**757.4±10.9**	**820.1±24.2**	**944.1±29.6**

Table 3: Comparison of average episodic reward with different action sizes on askscience dataset.

ing DRRN-Attention model with the greedy procedure outperforms all the baselines in four of the datasets.

8.2 Agent Performances on Various Action Sizes

We evaluate all the neural network models and approximation procedures across various action sizes with $K = 2, 3, 4, 5$ and fix $N = 10$ on the askscience dataset. The results are presented in Table 3. Our DRRN-Attention model outperforms the baselines across all the action sizes form $K = 2$ to $K = 5$ with the random sampling procedure or the greedy procedure. With the two-stage Q-learning procedure, our model achieves a higher cumulative reward than the baseline models when $K = 2, 3, 4$. Then, we analyze the performance of the greedy procedure in each neural network model. When using the DRRN-BiLSTM to parameterize the Q-function, the greedy procedure outperforms the baseline procedures when $K = 3, 4, 5$. When using our DRRN-Attention, the greedy procedure achieves a higher cumulative reward than the baselines when $K = 2, 3, 5$. Overall, using the DRRN-Attention model with the greedy procedure achieves the best performances across all the action sizes form $K = 2$ to $K = 5$.

8.3 Case Study

Table 4 presents an example of Q-value prediction of a state and three sub-actions on the askscience dataset. In this study, we enumerate every permutation of these three sub-actions, e.g., (1, 3, 2) denotes a permutation of comments that we place comment (1) in the first position, comment (3) in the second position, and comment (2) in the third position. Then, we use a trained DRRN-BiLSTM model and a trained DRRN-Attention model to predict the Q-value of each permutation of comments. When we use the DRRN-BiLSTM model, a different permutation of comments yields a different Q-value prediction. The Q-value prediction of the permutation (1, 3, 2) almost triples that of the permutation (2, 3, 1). On the other hand, any permutation of the comments does not change the Q-value prediction when we use our DRRN-Attention model. This example demonstrates that the ordering of the comments can significantly affect the predicted Q-value when we use the DRRN-BiLSTM model.

State
Is the heat I feel when I face a bonfire transmitted to me mostly by infrared radiation or by heated air?
Sub-actions (comments)
(1) Should it also be taken into consideration, that electromagnetic radiation is received differently depending on it's wavelength? (2) Are the light from the fire and it's heat one in the same? Because when I'm sitting at a campfire and it starts making my face feel hot... (3) The infrared radiation of a hot object is proportional to the fourth power of T, where T is the centigrade temperature + 273, ...

Permutation	$Q(s, a; \theta)$ **by DRRN-BiLSTM**	$Q(s, a; \theta)$ **by DRRN-Attention**
(1, 2, 3)	131.3	117.0
(1, 3, 2)	132.1	117.0
(2, 1, 3)	67.1	117.0
(2, 3, 1)	45.6	117.0
(3, 1, 2)	84.5	117.0
(3, 2, 1)	60.4	117.0

Table 4: A Q-value prediction example using DRRN-BiLSTM and DRRN-Attention.

8.4 Discussions

As mentioned in Section 7.1, the datasets that we use have a fewer number of comments than the datasets used by previous work. Table 5 com-

Subreddit	**# Cmts (their)**	**# Cmts (our)**
askscience	0.32M	0.15M
askmen	1.06M	0.94M
todayilearned	5.11M	4.65M
worldnews	5.99M	4.28M
nfl	6.12M	5.72M

Table 5: The number of comments in the datasets used by He et al. (2016c) and us.

Subreddit	**Reward (their)**	**Reward (our)**
askscience	833.9	643.2
askmen	148.0	184.3
todayilearned	697.9	659.0

Table 6: The cumulative reward achieved by the DRRN-BiLSTM + two-stage Q-learning baseline reported by He et al. (2017) and us.

pares the number of comments in the datasets used by (He et al., 2016c) and us. The experiments in (He et al., 2017) used three of the datasets (askscience, askmen, and todayilearned) from (He et al., 2016c). We compare our results and the results reported in (He et al., 2017) of the DRRN-BiLSTM + two-stage Q-learning baseline on these three datasets in Table 6. On the askscience and todayilearned datasets, the results of our implementation are worse than the results reported by them. Since the number of comments in our askscience dataset is only half of that in (He et al., 2017), the results of our implementation on the askscience dataset are significantly worse than the results reported by them. On the askmen dataset, the number of comments that we use is slightly less than the askmen dataset used by them. However, the results of our implementation on askmen are better than the results reported by them. We suspect that the deleted comments may have low karma scores, which cause the agent to achieve a higher cumulative reward.

9 Conclusion

In this work, we propose a new approach to the task of thread popularity prediction and tracking. In our approach, we propose a new neural network architecture, DRRN-Attention, to approximate the Q-function, which well respect the permutation invariance of the comments in an action. Moreover, our approach employs the greedy procedure to approximate the predicted optimal action, which does not require an additional pre-trained model to generate candidate actions. Empirical studies on real data demonstrate that our approach beats the current state-of-the-art in most of the experimental settings.

Acknowledgments

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 of the General Research Fund) and the Ministry of Education of China (D.01.16.00101). We would like to thank Jiani Zhang, Hongyi Zhang, Wang Chen, Yifan Gao, and Xiaotian Yu for their comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2014.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

S. R. K. Branavan, David Silver, and Regina Barzilay. 2011. Learning to win by reading manuals in a monte-carlo framework. In *ACL*, 2011.

S. R. K. Branavan, Luke S. Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *ACL*, 2010.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2017. Ask the right questions: Active question reformulation with reinforcement learning. In *ICLR*, 2018.

Jianshu Chen, Chong Wang, Lin Xiao, Ji He, Lihong Li, and Li Deng. 2017. Q-LDA: uncovering latent patterns in text-based sequential decision processes. In *NIPS*, pages 4984–4993, 2017.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2017. A factored neural network model for characterizing online discussions in vector space. In *EMNLP*, pages 2296–2306, 2017.

Justin Cheng, Lada A. Adamic, P. Alex Dow, Jon M. Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted? In *WWW*, pages 925–936, 2014.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *ACL*, pages 484–495, 2017.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016a. Dual learning for machine translation. In *NIPS*, pages 820–828, 2016.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016b. Deep reinforcement learning with a natural language action space. In *ACL*, Volume 1, 2016.

Ji He, Mari Ostendorf, and Xiaodong He. 2017. Reinforcement learning with external knowledge and two-stage q-functions for predicting popular reddit threads. *CoRR*, abs/1704.06217.

Ji He, Mari Ostendorf, Xiaodong He, Jianshu Chen, Jianfeng Gao, Lihong Li, and Li Deng. 2016c. Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads. In *EMNLP*, pages 1838–1848, 2016.

Liangjie Hong, Ovidiu Dan, and Brian D. Davison. 2011. Predicting popular messages in twitter. In *WWW*, pages 57–58, 2011

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *EMNLP*, pages 1192–1202, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *EMNLP*, pages 2355–2365, 2016.

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 77–85, 2017.

Reddit. 2017. The best of reddit in 2017. `https://redditblog.com/2017/12/19/the-best-of-reddit-in-2017/`. Accessed: May 22, 2018.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Continuously learning neural dialogue management. *CoRR*, abs/1606.02689.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*, pages 2440–2448, 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on twitter. In *ACL*, pages 175–185, 2014.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *CoRR*, abs/1710.10903.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *ICLR*, 2015.

Adam Vogel and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *ACL*, pages 806–814, 2010.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *ICLR*, 2015.

Tae Yano and Noah A Smith. 2010. What's worthy of comment? content and comment volume in political blogs. In *ICWSM*, 2010.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan R. Salakhutdinov, and Alexander J. Smola. 2017. Deep sets. In *NIPS*, pages 3394–3404, 2017.

Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. 2017. Dynamic key-value memory networks for knowledge tracing. In *WWW*, pages 765–774, 2017.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *CoRR*, abs/1803.07294.

IARM: Inter-Aspect Relation Modeling with Memory Networks in Aspect-Based Sentiment Analysis

Navonil Majumder[†], **Soujanya Poria**[‡], **Alexander Gelbukh**[†],
Md. Shad Akhtar[Φ], **Erik Cambria**[‡], **Asif Ekbal**[Φ]
[†]Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico
[‡]School of Computer Science and Engineering, Nanyang Technological Univerisity, Singapore
[Φ]Computer Science and Engineering, Indian Institute of Technology, Patna, India

`navo@nlp.cic.ipn.mx`, `sporia@ntu.edu.sg`, `gelbukh@gelbukh.com`,
`shad.pcs15@iitp.ac.in`, `cambria@ntu.edu.sg`, `asif@iitp.ac.in`

Abstract

Sentiment analysis has immense implications in modern businesses through user-feedback mining. Large product-based enterprises like Samsung and Apple make crucial business decisions based on the large quantity of user reviews and suggestions available in different e-commerce websites and social media platforms like Amazon and Facebook. Sentiment analysis caters to these needs by summarizing user sentiment behind a particular object. In this paper, we present a novel approach of incorporating the neighboring aspects related information into the sentiment classification of the target aspect using memory networks. Our method outperforms the state of the art by 1.6% on average in two distinct domains.

1 Introduction

Sentiment analysis plays a huge role in user-feedback extraction from different popular e-commerce websites like Amazon, eBay, etc. Large enterprises are not only interested in the overall user sentiment about a given product, but the sentiment behind the finer aspects of a product is also very important to them. Companies allocate their resources to research, development, and marketing based on these factors. Aspect-based sentiment analysis (ABSA) caters to these needs.

Users tend to express their opinion on different aspects of a given product. For example, the sentence "*Everything is so easy to use, Mac software is just so much simpler than Microsoft software.*" expresses sentiment behind three aspects: "*use*", "*Mac software*", and "*Microsoft software*" to be positive, positive, and negative respectively. This leads to two tasks to be solved: aspect extraction (Shu et al., 2017) and aspect sentiment polarity detection (Wang et al., 2016). In this paper, we tackle the latter problem by modeling the relation among different aspects in a sentence.

Recent works on ABSA does not consider the neighboring aspects in a sentence during classification. For instance, in the sentence "*The menu is very limited - I think we counted 4 or 5 entries.*", the sub-sentence "*I think ... entries*" does not reflect the true sentiment behind containing aspect "*entries*", unless the other aspect "*menu*" is considered. Here, the negative sentiment of "*menu*" induces "*entries*" to have the same sentiment. We hypothesize that our architecture iteratively models the influence from the other aspects to generate accurate target aspect representation.

In sentences containing multiple aspects, the main challenge an Aspect-Based-Sentiment-Analysis (ABSA) classifier faces is to correctly connect an aspect to the corresponding sentiment-bearing phrase (typically adjective). Let us consider this sentence "*Coffee is a better deal than overpriced cosi sandwiches*". Here, we find two aspects: *"coffee"* and *"cosi sandwiches"*. It is clear in this sentence that the sentiment of *"coffee"* is expressed by the sentimentally charged word *"better"*; on the other hand, *"overpriced"* carries the sentiment of *"cosi sandwiches"*. The aim of the ABSA classifier is to learn these connections between the aspects and their sentiment bearing phrases.

In this work, we argue that during sentiment prediction of an aspect (say *"coffee"* in this case), the knowledge of the existence and representation of the other aspects (*"cosi sandwiches"*) in the sentence is beneficial. The sentiment of an aspect in a sentence can influence the succeeding aspects due to the presence of conjunctions. In particular, for sentences containing conjunctions like *and, not only, also, but, however, though,* etc., aspects tend

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3402–3411
Brussels, Belgium, October 31 - November 4, 2018.

to share their sentiments. In the sentence *“Food is usually very good, though I wonder about freshness of raw vegetables”*, the aspect *“raw vegetables”* does not have any trivial sentiment marker linked to it. However, the positive sentiment of *“food”*, due to the word “”*good*”, and presence of conjunction “*though*” determines the sentiment of *“raw vegetables”* to be negative. Thus, aspects when arranged as a sequence, reveal high correlation and interplay of sentiments.

To model these scenarios, firstly, following Wang et al. (2016), we independently generate aspect-aware sentence representations for all the aspects using gated recurrent unit (GRU) (Chung et al., 2014) and attention mechanism (Luong et al., 2015). Then, we employ memory networks (Sukhbaatar et al., 2015) to repeatedly match the target aspect representation with the other aspects to generate more accurate representation of the target aspect. This refined representation is fed to a softmax classifier for final classification. We empirically show below that our method outperforms the current state of the art (Ma et al., 2017) by 1.6% on average in two distinct domains: restaurant and laptop.

The rest of the paper structured as follows: Section 2 discusses previous works; Section 3 delves into the method we present; Section 4 mentions the dataset, baselines, and experimental settings; Section 5 presents and analyzes the results; finally, Section 6 concludes this paper.

2 Related Works

Sentiment analysis is becoming increasingly important due to the rise of the need to process textual data in wikis, micro-blogs, and other social media platforms. Sentiment analysis requires solving several related NLP problems, like aspect extraction (Poria et al., 2016). Aspect based sentiment analysis (ABSA) is a key task of sentiment analysis which focuses on classifying sentiment of each aspect in the sentences.

In this paper, we focus on ABSA, which is a key task of sentiment analysis that aims to classify sentiment of each aspect individually in a sentence. In recent days, thanks to the increasing progress of deep neural network research (Young et al., 2018), novel frameworks have been proposed, achieving notable performance improvement in aspect-based sentiment analysis.

The common way of doing ABSA is feeding the aspect-aware sentence representation to the neural network for classification. This was first proposed by Wang et al. (2016) where they appended aspect embeddings with the each word embeddings of the sentence to generate aspect-aware sentence representation. This representation was further fed to an attention layer followed by softmax for final classification.

More recently, Ma et al. (2017) proposed a model where both context and aspect representations interact with each other’s attention mechanism to generate the overall representation. Tay et al. (2017) proposed word-aspect associations using circular correlation as an improvement over Wang et al. (2016)’s work. Also, Li et al. (2018) used transformer networks for target-oriented sentiment classification.

ABSA has also been researched from a question-answering perspective where deep memory networks have played a major role (Tang et al., 2016b; Li et al., 2017). However, unlike our proposed method, none of these methods have tried to model the inter-aspect relations.

3 Method

In this section, we formalize the task and present our method.

3.1 Problem Definition

Input We are given a sentence $S = [w_1, w_2, \dots, w_L]$, where w_i are the words and L is the maximum number of words in a sentence. Also, the given aspect-terms for sentence S are $A_1, A_2, \dots, A_M$, where $A_i = [w_k, \dots, w_{k+m-1}]$, $1 \leq k \leq L$, $0 < m \leq L - k + 1$, and M is the maximum number of aspects in a sentence.

Output Sentiment polarity (1 for positive, 0 for negative, and 2 for neutral) for each aspect-term A_i.

3.2 Model

The primary distinction between our model and the literature is the consideration of the neighboring aspects in a sentence with the target aspect. We assume that our inter-aspect relation modeling (IARM) architecture[1] models the relation between the target aspect and surrounding aspects, while filtering out irrelevant information. Fig. 1 depicts our model.

---

[1] Implementation available on `http://github.com/senticnet/IARM`

3.2.1 Overview

Our IARM model can be summarized with the following steps:

Input Representation We replace the words in the input sentences and aspect-terms with pre-trained Glove word embeddings (Pennington et al., 2014). For multi-worded aspect-terms, we take the mean of constituent word embeddings as aspect representation.

Aspect-Aware Sentence Representation Following Wang et al. (2016), all the words in a sentence are concatenated with the given aspect representation. These modified sequence of words are fed to a gated recurrent unit (GRU)[2] for context propagation, followed by an attention layer to obtain the aspect-aware sentence representation; we obtain for all the aspects in a sentence.

Inter-Aspect Dependency Modeling We employ memory network (Sukhbaatar et al., 2015) to model the dependency of the target aspect with the other aspects in the sentence. This is achieved through matching target-aspect-aware sentence representation with aspect-aware sentence representation of the other aspects. After a certain number of iterations of the memory network, we obtain a refined representation of the sentence that is relevant to the sentiment classification of the target aspect. Further, this representation is passed to a softmax layer for final classification. The following subsections discuss these steps in details.

3.2.2 Input Representation

The words (w_i) in the sentences are represented with 300 (D) dimensional Glove word embeddings (Pennington et al., 2014), resulting sentence $S \in \mathbb{R}^{L \times D}$.

Similarly, aspect terms are represented with word embeddings. Multi-worded aspect terms are averaged over the constituent words. This results aspect representation $a_i \in \mathbb{R}^D$ for i^{th} aspect term.

3.2.3 Aspect-Aware Sentence Representation

It would be fair to assume that not all the words in a sentence carry sentimental information of a particular aspect (e.g., stop words have no impact). This warrants a sentence representation that reflects the sentiment of the given aspect. To achieve this, we first concatenate aspect a_i to all the words in the sentence S:

$$S_{a_i} = [w_1 \oplus a_i, w_2 \oplus a_i, \dots, w_L \oplus a_i] \in \mathbb{R}^{L \times 2D}. \quad (1)$$

In order to propagate the context information within the sentence, we feed S_{a_i} to a Gated Recurrent Unit (GRU) with output size D_s (kindly refer to Table 1 for the value). We denote this GRU as GRU_s. $GRU_{\cdot}$ is described as follows:

$$z = \sigma(x_t U_{\cdot}^{z} + s_{t-1} W_{\cdot}^{z}), \quad (2)$$

$$r = \sigma(x_t U_{\cdot}^{r} + s_{t-1} W_{\cdot}^{r}), \quad (3)$$

$$h_t = \tanh(x_t U_{\cdot}^{h} + (s_{t-1} * r) W_{\cdot}^{h}), \quad (4)$$

$$s_t = (1 - z) * h_t + z * s_{t-1}, \quad (5)$$

where h_t and s_t are the hidden outputs and the cell states respectively at time t. We obtain $R_{a_i} = GRU_s(S_{a_i})$, where $R_{a_i} \in \mathbb{R}^{L \times D_s}$ and the GRU_s has the following parameters: $U_s^z \in \mathbb{R}^{2D \times D_s}$, $W_s^z \in \mathbb{R}^{D_s \times D_s}$, $U_s^r \in \mathbb{R}^{2D \times D_s}$, $W_s^r \in \mathbb{R}^{D_s \times D_s}$, $U_s^h \in \mathbb{R}^{2D \times D_s}$, $W_s^h \in \mathbb{R}^{D_s \times D_s}$.

To amplify the sentimentally relevant words to aspect a_i, we employ an attention layer to obtain the aspect-aware sentence representation (it is effectively a refined aspect representation) r_{a_i}:

$$z = R_{a_i} W_s + b_s, \quad (6)$$

$$\alpha = \text{softmax}(z), \quad (7)$$

$$r_{a_i} = \alpha^T R_{a_i}, \quad (8)$$

where $z = [z_1, z_2, \dots, z_L] \in \mathbb{R}^{L \times 1}$, $\text{softmax}(x) = [e^{x_1}/\sum_j e^{x_j}, e^{x_2}/\sum_j e^{x_j}, \dots]$, $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_L] \in \mathbb{R}^{L \times 1}$, $r_{a_i} \in \mathbb{R}^{D_s}$, $W_s \in \mathbb{R}^{D_s \times 1}$, and b_s is a scalar.

3.2.4 Inter-Aspect Dependency Modeling

We feed $R = [r_{a_1}, r_{a_2}, \dots, r_{a_M}] \in \mathbb{R}^{M \times D_s}$ to a GRU (GRU_a) of size D_o (kindly refer to Table 1 for the value) to propagate aspect information among the aspect-aware sentence representations and obtain $Q = GRU_a(R)$, where $Q \in \mathbb{R}^{M \times D_o}$ and GRU_a has the following parameters: $U_a^z \in \mathbb{R}^{D_s \times D_o}$, $W_a^z \in \mathbb{R}^{D_o \times D_o}$, $U_a^r \in \mathbb{R}^{D_s \times D_o}$, $W_a^r \in \mathbb{R}^{D_o \times D_o}$, $U_a^h \in \mathbb{R}^{D_s \times D_o}$, $W_a^h \in \mathbb{R}^{D_o \times D_o}$. This partially helps to model the dependency among aspects in a sentence.

After this, in order to further inter-aspect dependency modeling, we employ memory networks (Sukhbaatar et al., 2015), where the target-aspect representation (target-aspect-aware sentiment representation) r_{a_t} is supplied as the query.

---

[2] LSTM (Hochreiter and Schmidhuber, 1997) yields similar performance, but requires training more parameters

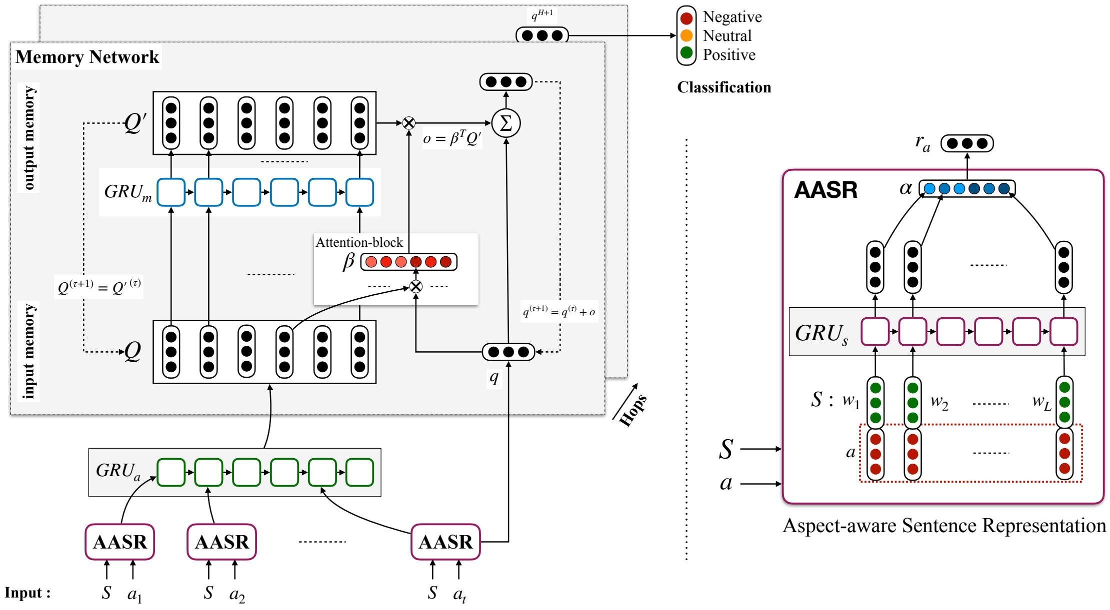

Figure 1: The proposed IARM architecture; AASR stands for *Aspect-Aware Sentence Representation*.

r_{a_t} is transformed into internal query state (q) with a fully connected layer as

$$q = \tanh(r_{a_t} W_T + b_T), \quad (9)$$

where $q \in \mathbb{R}^{D_o}$, $W_T \in \mathbb{R}^{D_s \times D_o}$, and $b_T \in \mathbb{R}^{D_o}$.

Input Memory Representation All the aspects in the sentence are stored in memory. Each aspect is represented by their corresponding aspect-aware sentence representation in Q. An attention mechanism is used to read these memories from Q (Weston et al., 2014). We compute the match between the query q and the memory slots in Q with inner product:

$$z = qQ^T, \quad (10)$$

$$\beta = softmax(z), \quad (11)$$

where $z = [z_1, z_2, \dots, z_M] \in \mathbb{R}^{M \times 1}$, $\beta = [\beta_1, \beta_2, \dots, \beta_M] \in \mathbb{R}^{M \times 1}$. Here, β_i is the measure of relatedness between target aspect and aspect i i.e., the attention score.

Output Memory Representation We choose the output memory vectors (Q') to be a refined version of the input memory vectors (Q), obtained by applying a GRU of size D_o (named GRU_m) on Q. Hence,

$$Q' = GRU_m(Q), \quad (12)$$

where GRU_m has the following parameters: $U_m^z \in \mathbb{R}^{D_o \times D_o}$, $W_m^z \in \mathbb{R}^{D_o \times D_o}$, $U_m^r \in \mathbb{R}^{D_o \times D_o}$, $W_m^r \in \mathbb{R}^{D_o \times D_o}$, $U_m^h \in \mathbb{R}^{D_o \times D_o}$, $W_m^h \in \mathbb{R}^{D_o \times D_o}$.

The response vector o is obtained by summing output vectors in Q', weighted by the relatedness measures in β:

$$o = \beta^T Q', \quad (13)$$

where $o \in \mathbb{R}^{D_o}$.

Final Classification (Single Hop) In the case of single hop, target aspect representation q is added with memory output o to generate refined target aspect representative. This sum is passed to a softmax classifier of size C ($C = 3$ due to the classes of sentiment polarity):

$$\mathcal{P} = \text{softmax}((q + o)W_{smax} + b_{smax}), \quad (14)$$

$$\hat{y} = \underset{i}{\text{argmax}}(\mathcal{P}[i]), \quad (15)$$

where $W_{smax} \in \mathbb{R}^{D_o \times C}$, $b_{smax} \in \mathbb{R}^C$, and $\hat{y}$ is the estimated sentiment polarity (0 for negative, 1 for positive, and 2 for neutral).

Multiple Hops We use total H (kindly refer to Table 1 for the value) number of hops in our model. Each hop generates a finer aspect representation q. Hence, we formulate the hops in the following way:

- Query (q) at the end of hop τ is updated as

$$q^{(\tau+1)} = q^{(\tau)} + o. \tag{16}$$

- Output memory vectors of hop τ, $Q'^{(\tau)}$, is updated as the input memory vectors of hop $\tau + 1$:

$$Q^{(\tau+1)} = Q'^{(\tau)}. \tag{17}$$

After H hops, $q^{(H)}$ becomes the target-aspect-aware sentence representation vector for the final classification:

$$\mathcal{P} = \text{softmax}(q^{(H+1)} W_{smax} + b_{smax}), \tag{18}$$

$$\hat{y} = \underset{i}{\text{argmax}}(\mathcal{P}[i]), \tag{19}$$

where $W_{smax} \in \mathbb{R}^{D_o \times C}$, $b_{smax} \in \mathbb{R}^C$, and $\hat{y}$ is the estimated sentiment polarity (0 for negative, 1 for positive, and 2 for neutral). The whole algorithm is summarized in Algorithm 1.

3.3 Training

We train the network for 30 epochs using categorical cross entropy with L2-regularizer as loss function (L):

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=0}^{C-1} y_{ik} \log \mathcal{P}[k] + \lambda \left\|\theta\right\|_2, \tag{20}$$

where N is the number of samples, i is the sample index, k is the class value, λ is the regularization weight (we set it to 10^{-4}),

$$y_{ik} = \begin{cases} 1, & \text{if expected class value of sample } i \text{ is } k, \\ 0, & \text{otherwise,} \end{cases} \tag{21}$$

and θ is the set of parameters to be trained, where

$$\theta = \{U^z_{\{s,a,m\}}, W^z_{\{s,a,m\}}, U^r_{\{s,a,m\}}, W^r_{\{s,a,m\}}, U^h_{\{s,a,m\}}, W^h_{\{s,a,m\}}, W_s, b_s, W_T, b_T, W_{smax}, b_{smax}\}.$$

As optimization algorithm, Stochastic Gradient Descent (SGD)-based ADAM algorithm (Kingma and Ba, 2014) is used with learning-rate 0.001 due to its parameter-wise adaptive learning scheme.

Hyper-Parameters We employed grid-search to obtain the best hyper-parameter values. Table 1 shows the best choice of these values.

Algorithm 1 IARM algorithm

```
procedure TRAINANDTESTMODEL($U$, $V$)    ▷ $U$ = train set, $V$ = test set
    Aspect-aware sentence representation extraction:
    for i:[1,$M$] do    ▷ generate for all the aspects in the sentence
        $r_{a_i} \leftarrow AspectAwareSentRep(S, a_i)$

    Query generation:
    $q \leftarrow FCLayer(r_{a_t})$    ▷ Transform the target-aspect-aware sentence representation to the query of memory network

    Memory networks:
    $Q \leftarrow GRU_a([r_{a_1}, r_{a_2}, \ldots, r_{a_M}])$    ▷ initial input memory
    $Q' \leftarrow GRU_m(Q)$    ▷ initial output memory

    for i:[1,$H$] do    ▷ memory network hops
        $z \leftarrow qQ^T$    ▷ match with target aspect
        $\beta \leftarrow softmax(z)$
        $o \leftarrow \beta^T Q'$    ▷ response vector
        $Q \leftarrow Q'$    ▷ input memory for the next hop
        $q \leftarrow q + o$    ▷ update target-aspect-aware sentence representation (query)

    Classification:
    $\hat{y} = \underset{j}{\text{argmax}}(softmax(q)[j])$    ▷ softmax classification
    $TestModel(V)$

procedure ASPECTAWARESENTREP(S,a)    ▷ generation of aspect-aware sentence representation
    $R_a \leftarrow GRU_s([w_1 \oplus a, w_2 \oplus a, \ldots, w_L \oplus a])$    ▷ $S = [w_1, w_2, \ldots, w_L]$
    $z \leftarrow FCLayer(R_a)$
    $\alpha \leftarrow softmax(z)$
    $r_a \leftarrow \alpha^T R_a$
    return $r_a$

procedure TESTMODEL($V$)
    Similar to the training phase, $V$ is passed through the learnt models to get the classification outputs. Section 3.3 mentions the trainable parameters ($\theta$).
```

4 Experiments

In this section, we discuss the dataset used and different experimental settings devised for the evaluation of our model.

4.1 Dataset Details

We evaluate our model with SemEval-2014 ABSA dataset[3]. It contains samples from two different domains: Restaurant and Laptop. Table 2 shows the distribution of these samples by class labels. Also, Table 3 shows the count of the samples with single aspect sentence and multi-aspect sentence.

---

[3] http://alt.qcri.org/semeval2014/task4

4.2 Baseline Methods

We compare our method against the following baseline methods:

LSTM Following Wang et al. (2016), the sentence is fed to a long short-term memory (LSTM) network to propagate context among the constituent words. The mean of all the hidden outputs from the LSTM is taken as the sentence representation, which is fed to a softmax classifier. Aspect-terms have no participation in the classification process.

TD-LSTM Following Tang et al. (2016a), sequence of words preceding (left context) and succeeding (right context) target aspect term are fed to two different LSTMs. Mean of the hidden outputs of the LSTMs are concatenated and fed to softmax classifier.

AE-LSTM Following Wang et al. (2016), the sentence is fed to an LSTM for context propagation. Then, the hidden outputs are concatenated with target-aspect representation, from which attention scores are calculated. Hidden outputs are pooled based on the attention scores to generate intermediate aspect representation. Final representation is generated as the sum of the affine transformations of intermediate representation and final LSTM hidden output. This representation is fed to softmax classifier.

ATAE-LSTM Following Wang et al. (2016), ATAE-LSTM is identical to AE-LSTM, except the LSTM is fed with the concatenation of aspect-term representation and word representation.

IAN Following Ma et al. (2017), target-aspect and its context are sent to two distinct LSTMs and the means of the hidden outputs are taken as intermediate aspect representation and context representation respectively. Attention scores are generated from the hidden outputs of both LSTMs which is used to generate final aspect and context representation. The concatenation of these two vectors are sent to a softmax classifier for final classification.

Hyper-Parameter	Restaurant	Laptop
D_s	300	400
D_o	350	400
Hop Count	3	10

Table 1: Hyper-parameter choices.

Domain	Positive		Negative		Neutral	
	Train	Test	Train	Test	Train	Test
Restaurant	2164	728	805	196	633	196
Laptop	987	341	866	128	460	169

Table 2: Distribution of the samples by class labels in SemEval 2014 dataset.

4.3 Experimental Settings

In order to draw a comprehensive comparison between our IARM model and the baseline methods, we performed the following experiments:

Overall Comparison IARM is compared with the baseline methods for both of the domains.

Single Aspect and Multi Aspect Scenarios In this setup, samples with single aspect and multi aspect sentences are tested independently on the trained model. For IAN, we ran our own experiments for this scenario.

Cross-Domain Evaluation Here, the model trained for restaurant domain is tested with the test set for laptop domain and vice versa. For IAN, we ran our own experiments for this scenario.

5 Results and Discussion

We discuss the results of different experiments below:

Overall Comparison We present the overall performance of our model against the baseline methods in Table 4.

It is evident from the results that our IARM model outperforms all the baseline models, including the state of the art, in both of the domains. We obtained bigger improvement in laptop domain, of 1.7%, compared to restaurant domain, of 1.4%. This shows that the inclusion of the neighboring aspect information and memory network has an overall positive impact on the classification process.

Single Aspect and Multi-Aspect Scenarios Following Table 5, our IARM model beats the

Domain	Train		Test	
	SA	MA	SA	MA
Restaurant	1007	2595	285	835
Laptop	917	1396	259	379

Table 3: Distribution of the samples by single aspect/multi aspect sentence criteria in SemEval 2014 (SA: Single Aspect, MA: Multi Aspect).

Model	Domain	
	Restaurant	Laptop
Majority Voting	53.4	65.0
LSTM	74.3	66.5
TD-LSTM	75.6	68.1
AE-LSTM	76.2	68.9
ATAE-LSTM	77.2	68.7
IAN (SoA)	78.6	72.1
IARM	**80.0**	**73.8**

Table 4: Domain-wise accuracy (%) of the discussed models. Best accuracy for each domain is marked with bold font.

state of the art in both single aspect and multi-aspect scenarios in both of the domains. It is interesting that both model perform better in multi-aspect scenario for restaurant domain. However, for laptop domain IAN performs better in single aspect scenario, even though there are more multi-aspect samples than single aspect samples (shown in Table 3). This indicates the failure of IAN model to learn multi-aspect scenario, where IARM model performs significantly better.

Model	Restaurant		Laptop	
	SA	MA	SA	MA
IAN (SoA)	75.4	77.7	72.5	71.6
IARM	**78.6**	**80.48**	**73.4**	**74.1**

Table 5: Accuracy of the models for single aspect and multi aspect scenario; SA: Single Aspect, MA: Multi Aspect.

Cross-Domain Evaluation Following Table 6, IARM outperforms the state of the art IAN by 2% in both cross-domain scenarios. This indicates the ability of IARM in learning general domain-independent semantic structures from the training data.

Model	Rest → Lap	Lap → Rest
IAN (SoA)	64.6	72.0
IARM	**66.7**	**74.0**

Table 6: Accuracy for cross-domain evaluation; Rest: Restaurant domain, Lap: Laptop domain; A → B signifies train-set is the train-set of domain A and test-set is the test-set of domain B.

5.1 Case Study

We analyze and compare IARM and IAN with single aspect and multi-aspect samples from the SemEval 2014 dataset.

Single Aspect Case It is evident from Table 5, that IARM outperforms IAN in single-aspect scenario. For example, the sentence "*I recommend any of their salmon dishes......*" having aspect "*salmon dishes*", with positive sentiment, fails to be correctly classified by IAN as the attention network focuses on the incorrect word "*salmon*", as shown in Fig. 2a. Since, "*salmon*" does not carry any sentimental charge, the network generates a ineffective aspect-aware sentiment representation, which leads to misclassification.

On the other hand, IARM succeeds in this case, because the word-level attention network generates correct attention value as α in Eq. (7). α for this case is depicted in Fig. 2b, where it is clear that the network emphasizes the correct sentiment-bearing word "*recommended*". This leads to effective aspect-aware sentence representation by the network, making correct final classification.

Multi-Aspect Case IARM also outperforms IAN in multi-aspect scenario, which can be observed in Table 5. We suspect that the presence of multiple aspects in sentence makes IAN network perplexed as to the connection between aspect and the corresponding sentiment-bearing word in the sentence. For example, the sentence "*Coffee is a better deal than overpriced cosi sandwiches*" contains two aspects: "*coffee*" and "*better*". Clearly, the sentiment behind aspect "*coffee*" comes from the word "*better*" and the same for aspect "*cosi sandwiches*" comes from "*overpriced*". However, IAN fails to make this association for the aspect "*cosi sandwiches*", evident from the attention weights of IAN shown in Fig. 3a where the emphasis is on "*better*". This leads to imperfect aspect-aware sentence representation generation, resulting misclassification of the target aspect to be positive.

However, IARM resolves this issue with the combination of word-level aspect aware attention (α) and the memory network. Since, the memory network compares the target-aspect-aware sentence representation with the sentence representations for the other aspects repeatedly, eventually the correct representation for the target aspect emerges from the memory network.

Also, the consideration of surrounding aspects forces the network to better distinguish the sentiment-bearing words for a particular aspect. These points are reflected in the α attention weights of the aspects "*coffee*" and "*cosi sand-*

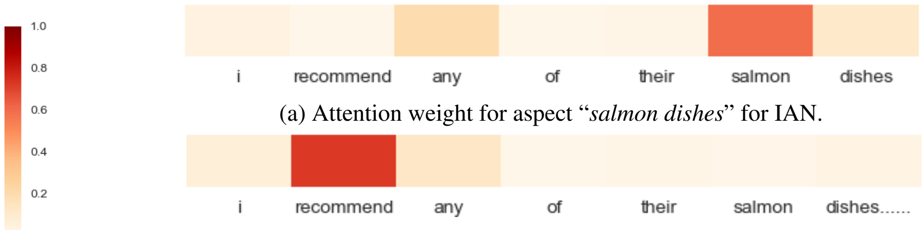

(a) Attention weight for aspect "*salmon dishes*" for IAN.

(b) Attention weight for aspect "*salmon dishes*" for IARM.

Figure 2: Attention weights for IAN and IARM for "*I recommend any of their salmon dishes*".

(a) Attention weights for aspect "*cosi sandwiches*" for IAN.

(b) Attention weights for aspect "*cosi sandwiches*" for IARM.

(c) Attention weights for aspect "*coffee*" for IARM.

Figure 3: Attention weights for IAN and IARM for the sentence "*Coffee is a better deal than overpriced cosi sandwiches*".

wiches", shown in Fig. 3b and Fig. 3c respectively, where the network emphasizes the correct sentiment-bearing words for each aspect, "*better*" and "*overpriced*", respectively. Again, the memory network compares the target aspect-aware sentence representation for "*cosi sandwiches*" with the same for "*coffee*" and incorporates relevant information into the target-aspect representation q in Eq. (16) along several hops.

This phenomenon is indicated in Fig. 4a, where the degree of incorporation of the aspect terms is measured by the attention weights β in Eq. (11). Here, the network is incorporating information from aspect "*coffee*" into aspect "*cosi sandwiches*" over three hops. We surmise that this information is related to the sentiment-bearing word "*better*" of the aspect "*coffee*", because a comparison using the word "*better*" implies the presence of a good ("*coffee*") and a bad ("*cosi sandwiches*") object. However, this semantics is misconstrued by IAN, which leads to aspect misclassification.

IARM performs considerably well when conjunction plays a vital role in understanding the sentence structure and meaning for sentiment analysis. For example, "*my favs here are the tacos pastor and the tostada de tinga*" where the aspects *"tacos pastor"* and *"tostada de tinga"* are connected using conjunction *"and"* and both rely on the sentiment bearing word *favs*. Such complex relation between the aspects and the corresponding sentiment-bearing word is grasped by IARM as shown in Fig. 4b. Another example where the inter-aspect relation is necessary for the correct classification is shown in Fig. 5, where the aspects *"atmosphere"* and *"service"* both rely on the sentiment bearing word *"good"*, due to the conjunction *"and"*.

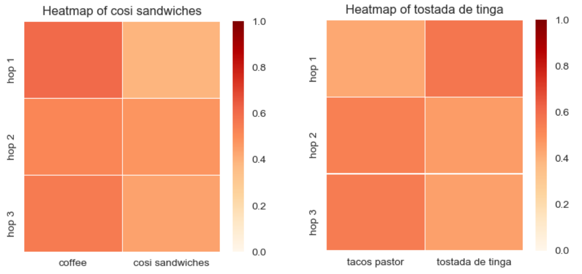

(a) Memory network attention weights for the sentence "*Coffee is a better deal than overpriced cosi sandwiches.*".

(b) Memory network attention weights for the sentence "*my favs here are the tacos pastor and the tostada de tinga.*".

Figure 4: Memory network attention weights for IARM.

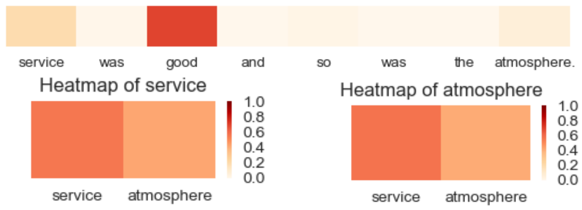

Figure 5: Memory network attention weights for IARM for the sentence "*service was good and so was the atmosphere.*" The word importance heatmap is for the aspect *"atmosphere"*.

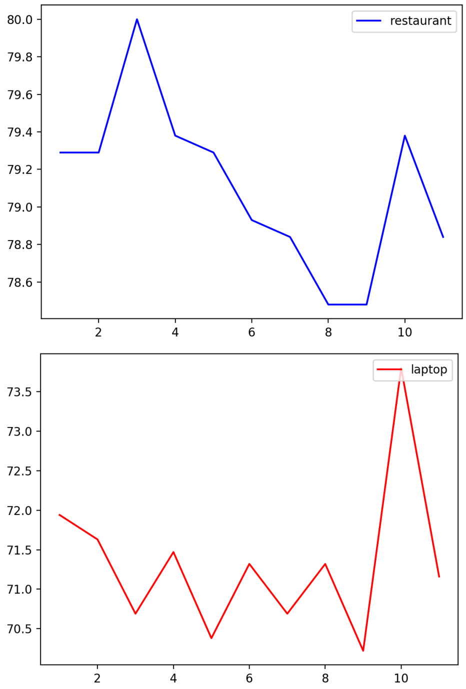

Figure 6: Hop-Accuracy plot for both domains.

5.2 Error Analysis

IARM also fails to correctly classify in some cases, e.g., in the sentence "*They bring a sauce cart up to your table and offer you 7 or 8 choices of sauces for your steak (I tried them ALL).*", the aspect "*choices of sauces*" is misclassified by our network as neutral. This happened due to the IARM's inability to correctly interpret the positive sentiment behind "*7 or 8 choices of sauces*" .

Again, the IARM could not correctly classify aspect the "*breads*" to be positive in the sentence "*Try the homemade breads.*". This happened, because the word "*try*" itself is not sentimentally charged, but can carry sentimental meaning given the right context. This context was not recognized by IARM, which led to misclassification.

5.3 Hop-Performance Relation

In our experiments, we tried different hop counts of the memory network. We observed that the network performs best with three hops for restaurant domain and ten hops for laptop domain, which is shown in the hop count - performance plot in Figure Fig. 6. It can be observed that the plot for restaurant domain is smoother than the plot for laptop domain. We assume that this is due to the restaurant domain having higher number of samples than laptop domain, as shown in Table 2.

Also, the plot for restaurant domain shows a downward trend over the increasing number of hops, with spikes in hop 3, hop 10. This suggests a irregular cyclic nature of the memory network where those certain hop counts yields higher quality representations than their neighbor. The same cannot be said for laptop domain as the plot presents a zig-zag pattern.

6 Conclusion

In this paper, we presented a new framework, termed IARM, for aspect-based sentiment analysis. IARM leverages recurrent memory networks with multihop attention mechanism. We empirically illustrate that an aspect in a sentence is influenced by its neighboring aspects. We exploit this property to obtain state-of-the-art performance in aspect-based sentiment analysis in two distinct domains: restaurant and laptop. However, there remains plenty of room for improvement in the memory network, e.g., for generation of better aspect-aware representations.

References

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. 2017. Deep memory networks for attitude identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 671–680. ACM.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 946–956, Melbourne, Australia. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive Attention Networks for Aspect-Level Sentiment Classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4068–4074.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49.

Lei Shu, Hu Xu, and Bing Liu. 2017. Lifelong learning crf for supervised aspect extraction. *arXiv preprint arXiv:1705.00251*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end Memory Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2440–2448, Cambridge, MA, USA. MIT Press.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective LSTMs for Target-Dependent Sentiment Classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307, Osaka, Japan. The COLING 2016 Organizing Committee.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. *arXiv preprint arXiv:1712.05403*.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75.

Limbic: Author-Based Sentiment Aspect Modeling Regularized with Word Embeddings and Discourse Relations

Zhe Zhang
Watson Group
IBM Corporation
Research Triangle Park, NC 27703-9141
zhangzhe@us.ibm.com

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
singh@ncsu.edu

Abstract

We propose Limbic, an unsupervised probabilistic model that addresses the problem of discovering aspects and sentiments and associating them with authors of opinionated texts. Limbic combines three ideas, incorporating authors, discourse relations, and word embeddings. For discourse relations, Limbic adopts a generative process regularized by a Markov Random Field. To promote words with high semantic similarity into the same topic, Limbic captures semantic regularities from word embeddings via a generalized Pólya Urn process. We demonstrate that Limbic (1) discovers aspects associated with sentiments with high lexical diversity; (2) outperforms state-of-the-art models by a substantial margin in topic cohesion and sentiment classification.

1 Introduction

How can we understand opinionated texts, e.g., social media postings, expressing sentiments about various entities? Three phenomena are key. First, even for similar entities, authors may differ both on aspects and sentiments about those aspects. For example, when reviewing a hotel, Alice may consider aspects such as *Concierge* and *Room*, whereas Bob may consider aspects such as *Nearby* and *Room*. Capturing similarities and differences among authors can help produce recommendations for services that are better aligned with a user's expectations (Wang et al., 2013). Second, reviews exhibit discourse structure, i.e., relations between propositions, which carries valuable information about sentiment. Third, crucial relationships between rare words are lost because each review may be short and use distinct rare words.

Probabilistic topic models (Hofmann, 1999; Blei et al., 2003) provide an unsupervised means to learn latent constructs from texts. Author-specific topic discovery associates texts with their authors (Rosen-Zvi et al., 2004; Kim et al., 2012; Diao and Jiang, 2013) but ignores sentiments. Sentiment analysis methods jointly model aspects and sentiments but exclude either authors (Lazaridou et al., 2013), discourse relations (Mukherjee et al., 2014; Poddar et al., 2017), or both (Jo and Oh, 2011; Lin et al., 2012; Kim et al., 2013).

Word co-occurrence sparsity plagues existing approaches, which model documents as distributions over latent topics and estimate them from word co-occurrence. Since word frequency follows a power law, most words are rare and representative words of a topic rarely co-occur, especially in short opinionated texts, despite semantic proximity. For example, a reviewer would not use both *spotless* and *immaculate* to express a positive sentiment toward the cleanliness of a hotel room. Losing information about word relatedness impedes learning effectiveness, producing topics that are not semantically cohesive.

We contribute Limbic, an unsupervised probabilistic model for discovering author-based aspects and sentiments from opinionated texts that incorporates discourse-level topic modeling and semantic cohesion. (1) It associates authors and sentiment-aspect pairs by generating a mixture over sentiments and aspects for each author. (2) It captures discourse relations by applying a Markov Random Field over Sentiment Expression Units (SEUs), i.e., text elements describing sentiment-aspect pairs. (3) It promotes words with high semantic similarity into the same topic by incorporating semantic regularities from word embeddings using a generalized Pólya Urn process.

We empirically compare Limbic with state-of-the-art models using datasets from two domains. Qualitatively, Limbic discovers aspect-sentiment pairs with higher lexical diversity. Quantitatively, Limbic obtains substantial improvements in topic cohesion and sentiment classification.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3412–3422
Brussels, Belgium, October 31 - November 4, 2018.

2 Model and Inference in Limbic

We now introduce our proposed model.

2.1 Sentiment Expression Unit (SEU)

Existing topic models represent documents as bags of words or as sentences. Bag-of-words models, e.g., LDA (Blei et al., 2003), AT (Rosen-Zvi et al., 2004), JST (Lin et al., 2012), JAST (Mukherjee et al., 2014), and AATS (Poddar et al., 2017), rely on word co-occurrence at the document level, which is problematic when applied to opinionated texts. Sentence-based models, e.g., ASUM (Jo and Oh, 2011), assume that words appearing in a sentence belong to the same aspect and sentiment, which often fails to hold in real text. For instance, the TripAdvisor review sentence *Service was good and friendly, location is good and my room was spacious but oldish*, exhibits three aspects, *Service*, *Location*, and *Room*, and two sentiments. Zhang and Singh's (2014) segmentation algorithm leverages transition cues to convert sentences into segments. Although transition cues are good indicators for capturing sentiment change, their algorithm disregards syntactic information in sentences, which also helps reveal changes of aspects and sentiments.

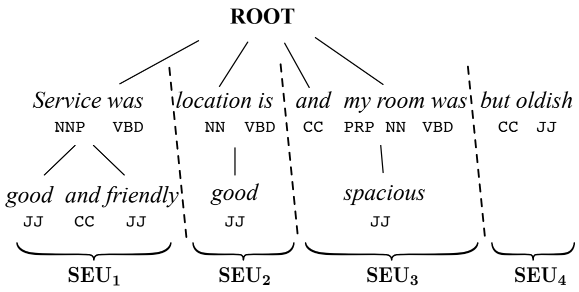

Figure 1: Generate SEUs from a sentence.

We propose a concept of sentiment expression unit (SEU). Each SEU contains either a sentiment, or an aspect, or both. We extract SEUs by incorporating both discourse and syntactic information. We first split sentences in reviews into snippets based on contradiction transition cues, such as *but*. Then we apply a grammar parser on each snippet. We extract phrases from snippets by using two syntactic patterns commonly observed in opinionated texts including (1) existential (EX) with verb (VB) and adjective (JJ) and (2) noun (NN) with verb (VB) and adjective (JJ). If a phrase matches a pattern, we identify it as an SEU. Otherwise, the phrase joins its following phrases iteratively until the combination matches a pattern. Figure 1 demonstrates the process of generating SEUs from the above hotel review sentence.

2.2 Discourse Relation

Markov Random Field (MRF) is a probabilistic framework to model statistical dependencies between variables. Limbic applies an MRF to capture the discourse relations between SEUs. Given a document containing N SEUs, let a_i and s_i be the aspect and sentiment assignments of SEU$_\mathrm{i}$, respectively. Limbic creates an undirected edge $\langle s_i, s_j \rangle$ between the sentiment assignments of this SEU and its preceding SEU. Let r be the discourse relation between SEUs, Limbic imposes a binary potential on the edge.

Limbic focuses on two discourse relations frequently observed in opinionated texts: *Comparison* and *Expansion*. Comparison highlights prominent differences between two SEUs and often signals a change of sentiment regardless of the change of aspect. For example, in SEU$_1$: {The location was great} and SEU$_2$: {but it was just too noisy}, we see that *but* indicates a sentiment difference. Other transition cues for Comparison include *however*, *in contrast*, and such.

Expansion extends the discourse and indicates a continuation of sentiment across SEUs. For example, in SEU$_3$: {There are no safes here which is unfortunate} and SEU$_4$: {And speaking of unfortunate, the breakfast is hardly impressive}, we see that *and* and *unfortunate* indicate the negative sentiment in SEU$_3$ continues toward aspect *Breakfast* in SEU$_4$. Other transition cues for Expansion include *also*, *moreover*, and such.

Formally, $\mathcal{R}_{r,i,j}$ asserts discourse relation r between SEU$_i$ and SEU$_j$. For Comparison, $\mathcal{R}_{c,i,j}$ holds if $s_i \neq s_j$, SEU$_j$ contains Comparison cues, and (1) SEU$_j$ contains syntactic patterns described in Section 2.1 and $a_i \neq a_j$ or (2) SEU$_j$ contains incomplete syntactic patterns and $a_i = a_j$.

For Expansion, $\mathcal{R}_{e,i,j}$ holds if $s_i = s_j$, SEU$_j$ contains Expansion cues, and (1) SEU$_j$ contains syntactic patterns and $a_i = a_j$ or (2) SEU$_j$ contains incomplete syntactic patterns and $a_i \neq a_j$.

Given document d, the joint probability of its sentiment assignments is:

$$p(\boldsymbol{s}|\theta_d) = \prod_i p(s_i|\theta_d) \exp\{\lambda \sum_{r=1}^{R}(I(\mathcal{R}_{r,i-1,i}))\}, \tag{1}$$

where R is the number of discourse relation types; θ_d is the sentiment distribution of d; I is an identity function that returns 1 if its argument is true; λ controls reinforcing the effects of discourse relations.

Take the expansion relation, for example. During the sampling process, Equation 1 generates a large value if two SEUs share an expansion relation and have the same sentiments and yields a small value if the two SEUs have different sentiments. Therefore, SEUs in an expansion relation have a high probability to be associated with the same sentiment.

2.3 Generative Process

Figure 2 shows Limbic's model. With $Dir(\cdot)$ and $Mul(\cdot)$ as Dirichlet and multinomial distributions, hyperparameter α is the Dirichlet prior of the word distribution ϕ, β is the Dirichlet prior of the sentiment distribution θ, and γ is the Dirichlet prior of the aspect distribution ψ. Given a set of reviews D written by a set of authors U with regards to a set of aspects T and a set of sentiments S, the generative process in Limbic is as follows.

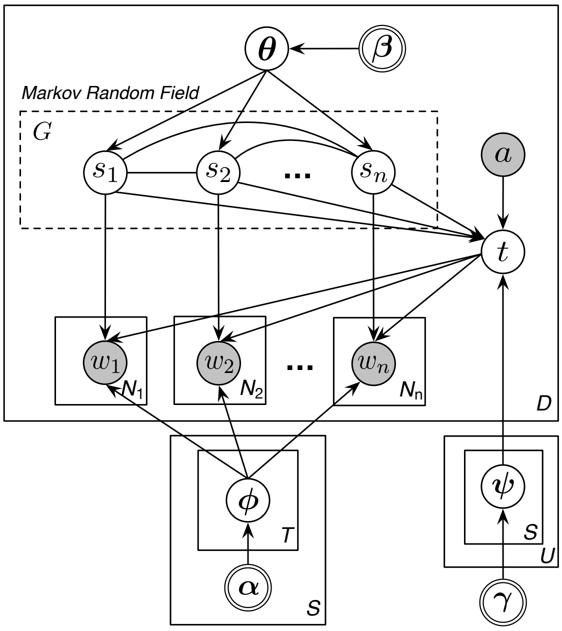

Figure 2: Generative process of Limbic.

First, for each pair of aspect t and sentiment s, draw a word distribution $\phi_{t,s} \sim Dir(\alpha)$. Second, for each author a and each sentiment s, draw an aspect distribution $\psi_{s,a} \sim Dir(\gamma)$. Third, given a review d written by a, draw a sentiment distribution $\theta_d \sim Dir(\beta)$, and for each SEU in d, (a) choose a sentiment s using Equation 1; (b) given s, choose an aspect $t \sim Mul(\psi_{s,a})$; (c) given t and s, sample word $w \sim Mul(\phi_{t,s})$.

2.4 Model Inference

Limbic estimates $p(\boldsymbol{s},\boldsymbol{t}|\boldsymbol{w},a)$, the posterior distribution of latent variables, sentiments $\boldsymbol{s}$ and aspects $\boldsymbol{t}$, given all words used in reviews written by author a. We factor the joint probability of the assignments of sentiments, aspects, and words for a:

$$p(\boldsymbol{s},\boldsymbol{t},\boldsymbol{w}|a,\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = p(\boldsymbol{w}|\boldsymbol{s},\boldsymbol{t},\boldsymbol{\alpha})p(\boldsymbol{t}|\boldsymbol{s},a,\boldsymbol{\gamma})p(\boldsymbol{s}|\boldsymbol{\beta}). \tag{2}$$

By integrating over $\Phi = \{\boldsymbol{\phi_i}\}_{i=1}^{S\times T}$, we calculate the first term of Equation 2 as follows.

$$p(\boldsymbol{w}|\boldsymbol{s},\boldsymbol{t},\boldsymbol{\alpha}) = \int p(\boldsymbol{w}|\boldsymbol{s},\boldsymbol{t},\Phi)p(\Phi|\boldsymbol{\alpha})\mathrm{d}\Phi = \left(\frac{\Gamma(\sum_{w=1}^{W}\alpha_w)}{\prod_{w=1}^{W}\Gamma(\alpha_w)}\right)^{S\times T} \times \prod_{s=1}^{S}\prod_{t=1}^{T}\frac{\prod_{w=1}^{W}\Gamma(n_{s,t}^{w}+\alpha_w)}{\Gamma[\sum_{w=1}^{W}(n_{s,t}^{w}+\alpha_w)]}, \tag{3}$$

where W is the size of the vocabulary; $n_{s,t}^{w}$ equals the number of occurrences of the word w that are assigned to sentiment s and aspect t; and $\Gamma(\cdot)$ is the Gamma function.

Next, by integrating over $\Psi_a = \{\boldsymbol{\psi_i}\}_{i=1}^{S}$, we calculate the second term in Equation 2 as follows.

$$p(\boldsymbol{t}|\boldsymbol{s},\boldsymbol{\gamma},a) = \int p(\boldsymbol{t}|\boldsymbol{s},\Psi_a,a)p(\Psi_a|\boldsymbol{\gamma})\mathrm{d}\Psi_a = \left(\frac{\Gamma(\sum_{t=1}^{T}\gamma_t)}{\prod_{t=1}^{T}\Gamma(\gamma_t)}\right)^{S} \times \prod_{s=1}^{S}\frac{\prod_{t=1}^{T}\Gamma(n_{s,a}^{t}+\gamma_t)}{\Gamma[\sum_{t=1}^{T}(n_{s,a}^{t}+\gamma_t)]}, \tag{4}$$

where $n_{s,a}^{t}$ equals the number of SEUs in author a's reviews associated with sentiment s and aspect t.

Similarly, for the third term in Equation 2, by integrating over $\Theta = \{\boldsymbol{\theta_i}\}_{i=1}^{D}$, we obtain

$$p(\boldsymbol{s}|\boldsymbol{\beta}) = \int p(\boldsymbol{s}|\Theta)p(\Theta|\boldsymbol{\beta})\mathrm{d}\Theta = \left(\frac{\Gamma(\sum_{s=1}^{S}\beta_s)}{\prod_{s=1}^{S}\Gamma(\beta_s)}\right)^{D} \times \prod_{d=1}^{D}\frac{\prod_{s=1}^{S}\Gamma(n_{d}^{s}+\beta_s)}{\Gamma[\sum_{s=1}^{S}(n_{d}^{s}+\beta_s)]} \tag{5}$$
$$\times \prod_{l=1}^{L}\exp\{\lambda\sum_{r=1}^{R}(I(\mathcal{R}_{r,i-1,i}))\},$$

where D is the number of reviews; n_d^s is the number of times that an SEU from review d is associated with sentiment s; and n_d is the number of SEUs in review d; L is the number of SEUs.

We obtain the conditional probability for a via Gibbs sampling (Liu, 1994)

$$\begin{aligned}
&p(s_i = s, t_i = t | \boldsymbol{s_{-i}}, \boldsymbol{t_{-i}}, \boldsymbol{w}, a) \\
&\quad \propto \frac{n_{s,a,-i}^t + \gamma_t}{\sum_{t=1}^T (n_{s,a,-i}^t + \gamma_t)} \\
&\quad \times \frac{n_{d,-i}^s + \beta_s}{\sum_{s=1}^S (n_{d,-i}^s + \beta_s)} \\
&\quad \times \frac{\prod_v \prod_{c=0}^{C_v^i - 1} (n_{t,s,-i}^v + \alpha_v + c)}{\prod_{c=0}^{C_i - 1} (n_{t,s,-i} + \sum_{w=1}^W \alpha_w + c)} \\
&\quad \times \exp\{\lambda \sum_{r=1}^{R} (I(\mathcal{R}_{r,i-1,i}))\}
\end{aligned} \tag{6}$$

where $n_{s,a}^t$, as in Equation 4, is the number of SEUs associated with sentiment s and aspect t from reviews written by author a; n_d^s is the number of SEUs from review d associated with sentiment s; C_i is the number of words in SEU$_i$; C_v^i is the number of words v in SEU$_i$; $n_{t,s}^v$ is the number of words v assigned sentiment s and aspect t; $n_{t,s}$ is the number of words assigned sentiment s and aspect t in all reviews; an index of $-i$ means we exclude SEU$_i$ from the count; R, I, and $\mathcal{R}$ are as in Equation 1.

Equations 7, 8, and 9, respectively, approximate the probabilities of word w occurring given sentiment s and aspect t; of aspect t of an SEU occurring given sentiment s and author a; of sentiment s occurring given document d.

$$\phi_{s,t,w} = \frac{n_{s,t}^w + \alpha_w}{\sum_{w=1}^W (n_{s,t}^w + \alpha_w)}. \tag{7}$$

$$\psi_{s,t,a} = \frac{n_{s,a}^t + \gamma_t}{\sum_{t=1}^T (n_{s,a}^t + \gamma_t)}. \tag{8}$$

$$\theta_{d,s} = \frac{n_d^s + \beta_s}{\sum_{s=1}^S (n_d^s + \beta_s)}. \tag{9}$$

Incorporating Word Embeddings. Word embedding approaches (Mikolov et al., 2013; Pennington et al., 2014) leverage local contextual information surrounding words to map the words into continuous vector representations. Word embeddings are known to effectively capture semantic and syntactic regularities among words. Based on word embeddings trained using Word2Vec on a hotel review dataset, we observe that the generated word embeddings correctly link opinionated words that are semantically correlated even though they do not co-occur frequently. For example, the three closest words of *spotless* are *immaculate*, *clean*, and *well appointed*.

A Generalized Pólya Urn Process. To promote words with high semantic similarity into the same topic, Limbic incorporates semantic regularities from word embeddings using a generalized Pólya Urn process (Mimno et al., 2011). Start with an urn containing colored balls. At each time step, randomly choose a ball from the urn, observe its color, and return it to the urn with one replicated ball of the same color. A Pólya Urn model describes a random sampling process with reinforcement. In a generalized Pólya Urn process, given a sampled ball with a color, we put back that ball along with a certain number of balls of similar colors. When applied to document generation, balls of different colors represent distinct words. The similarity of colors represents semantic similarity of the words.

Given words v and w in vocabulary W, we compute their semantic similarity $sim(v, w)$ based on the cosine similarity between their word embeddings. For word v, we create its similarity word set $\mathbb{S}_v$ by adding all words $w \in W$ for which $sim(v, w)$ is higher than a predefined threshold ϵ. During sampling, if word v is drawn, we reinforce $w \in \mathbb{S}_v$ via a predefined weight ρ which controls the reinforcement of semantically similar words.

Sentiment Alignment. Widely used Word embedding approaches, such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and fastText (Bojanowski et al., 2017), are semantically oriented and do not explicitly encode sentiment information in the generated word-vector representations. Hence, semantically related words with opposite polarity may have close vectors. For example, *smell* and *aroma* are synonyms but *smell* often expresses a negative sentiment toward aspect *Cleanliness* whereas *aroma* is often positive. Simply promoting all words may adversely affect the generated topics. Therefore, we calculate the sentiment alignment of each word in a vocabulary based on its average cosine similarity to the words in a general sentiment word list. In the sampling process, we promote words only if their sentiments align with sampled sentiments.

3 Evaluation

To assess Limbic's effectiveness, we prepare online review datasets from two domains. TripUser is a collection of hotel reviews from TripAdvisor. It contains 28,165 reviews posted by 202 randomly selected reviewers, each of whom contributes at least 100 hotel reviews. YelpUser is a set of restaurant reviews from Yelp Dataset Challenge (2017). It contains 23,873 restaurant reviews posted by 144 users, each of whom contributes at least 100 reviews. Table 1 reports statistics on the datasets.

Table 1: Summary of the evaluation datasets.

Statistic	TripUser	YelpUser
Number of reviews	28,165	23,873
Number of SEUs	484,805	359,191
Average SEUs / review	17	15
Average words / SEU	7	6

We remove stop words and HTML tags, expand typical abbreviations, and mark special named entities using a rule-based algorithm (e.g., replace a monetary amount by #MONEY#) and the Stanford Named Entity Recognizer (Finkel et al., 2005). To handle negation, we employ the Stanford Dependency Parser to detect negations. For any word in a negation relation, we add the negated term as a prefix of the word, e.g., *not_work*. Finally, we split each review into SEUs. Datasets and source code are publicly available for research purposes (Limbic, 2018).

3.1 Parameter Settings

Limbic includes three hand-tuned hyperparameters that influence its sampling via a smoothing effect on the associated multinomial distribution. It uses a short list of sentiment words shown in Table 2 as prior knowledge to set asymmetric priors.

Table 2: Sentiment words used as prior knowledge.

Positive
good, nice, excellent, positive, fortunate, correct, free, love attractive, awesome, perfect, comfortable, enjoy, amazing fun, glad, great, happy, impressive, superior, thank, best satisfied, worth, not_bad, recommend, fantastic, favorite
Negative
bad, nasty, poor, negative, unfortunate, wrong, inferior slow, junk, mess, not_good, not_like, not_recommend unacceptable, upset, waste, small, worthless, problem complain, terrible, trouble, regret, annoying, not_worth sorry, disappointed, worst, hate

Consider hyperparameter α, the Dirichlet prior of the word distribution. For any word in the positive list, $\alpha = 0$ if the word appears in an SEU assigned a negative sentiment, and $\alpha = 5$ if the word appears in an SEU assigned a positive sentiment, and conversely for words in the negative list. For all remaining words, we set $\alpha = 0.05$. And, hyperparameter $\beta = 5$ for both sentiments is the Dirichlet prior of the sentiment distribution. Using T as the number of aspects, hyperparameter $\gamma = \frac{50}{T}$ is the Dirichlet prior of the aspect distribution. We set the number of sentiments, S, to two (positive and negative), although our approach generalizes to additional sentiment categories.

For each fold in cross validation, we pretrain two sets of Word2Vec (Mikolov et al., 2013) word embeddings with 300 dimensions and a window size of five using the training split in TripUser (hotels) and YelpUser (restaurants). We exclude words with frequency lower than three. We set the reinforcement weight ρ to 0.3 and 0.1 for hotel and restaurant reviews, respectively, and set the similarity threshold ϵ to 0.6. For all models, we perform 1,000 Gibbs iterations with a burn-in phase of 200 and a sampling gap of 50 iterations.

3.2 Sentiment Aspect Discovery

Our first experiment shows how Limbic discovers sentiment-aspect pairs. We apply Limbic and all baseline models (AT, JST, ASUM, and AATS) to TripUser and YelpUser with the number of aspects set to 30. We manually assign an aspect for each cluster of words. ASUM generates the best results among baseline models. For brevity, we show only some aspects identified by Limbic and ASUM.

Table 3 (top) shows the results on hotel reviews. We see that Limbic discovers word clusters with higher lexical diversity than ASUM. For example, for aspect *Decoration*, in addition to words, *decor*, *modern* and *design*, Limbic discovers words *contemporary*, *minimalist*, *chic*, and so on. For aspect *Service*, comparing with ASUM, Limbic extracts an expanded list of sentiment words including *competent*, *knowledgeable*, and so on.

Limbic discovers finer and more distinctive word clusters than ASUM. For example, for aspect *Cleanliness*, ASUM generates a word cluster that includes negative sentiment words toward multiple entities, such as *carpet* and *hallway*. Limbic generates two distinctive word clusters for aspect

Table 3: Top words discovered from hotel (top section) and restaurant (bottom section) reviews.

Decoration		*Service*		*Cleanliness*			*Environment*
ASUM	Limbic	ASUM	Limbic	ASUM	Limbic	Limbic	Limbic
room	**contemporary**	staff	staff	carpet	smell	carpet	beautiful
modern	decor	friendly	friendly	smell	room	wallpaper	setting
furniture	colour	helpful	helpful	stain	cigarette	old	peaceful
decor	**tasteful**	desk	attentive	bathroom	smoke	furniture	relaxing
wood	room	front	courteous	room	floor	paint	golf
lobby	**marble**	english	hostess	dirty	odor	need	lush
design	**chic**	professional	professional	furniture	elevator	stain	gorgeous
wall	lobby	efficient	**gracious**	wall	reek	worn	environment
color	modern	service	**accommodating**	hallway	odour	carpeting	countryside
look	**elegant**	reception	**welcoming**	paint	smelt	bedspread	atmosphere
style	**artwork**	attentive	desk	smoke	smoking	remodel	ground
bathroom	**stone**	polite	**knowledgeable**	worn	hallway	update	course
dark	stylish	speak	**competent**	clean	stair	room	place
ceiling	**minimalist**	courteous	front	old	non-smoking	look	quiet
decorate	design	pleasant	service	tile	lift	bathroom	surroundings
Service		*Decoration*		*Portion*		*Mexican*	*Seafood*
ASUM	Limbic	ASUM	Limbic	ASUM	Limbic	Limbic	Limbic
friendly	friendly	decor	decor	portion	portion	asada	crab
server	service	atmosphere	atmosphere	size	size	carne	lobster
staff	attentive	place	**sleek**	small	half	taco	scallop
service	staff	modern	interior	large	large	tostada	shrimp
attentive	helpful	feel	vibe	plate	huge	burrito	roll
helpful	**efficient**	restaurant	**ambiance**	huge	serving	bean	risotto
nice	server	cool	clean	price	big	refried	salmon
waitress	**prompt**	inside	modern	big	could	corn	mussel
owner	**consistently**	clean	**ambience**	share	salad	guacamole	calamari
bartender	nice	vibe	**cozy**	#MONEY#	eat	cabbage	bisque
waiter	**professional**	nice	**contemporary**	enough	plate	torta	tuna
greet	host	like	place	half	share	pinto	jumbo
manager	great	interior	**comfortable**	generous	bowl	guac	clam
hostess	**quick**	space	**stylish**	bowl	sandwich	jalapeno	ahi
table	**knowledgeable**	look	inside	inch	generous	flour	tempura

Cleanliness. One cluster contains words, such as *smoke* and *reek*, which describe bad odor in room and hallway. The other cluster contains words such as, *worn* and *stain*, describes negative sentiments toward carpets. By capturing word semantic relatedness, Limbic discovers highly diverse aspects, including those that arise rarely in reviews, such as *peaceful*, *relaxing*, and *lush*, as positive words describing aspect *Environment*.

Limbic yields promising results for restaurant reviews. In Table 3 (bottom), we see that Limbic yields more specific sentiment words than ASUM. Aspect *Service* in Limbic contains additional positive words, *efficient*, *prompt*, *knowledgeable*, and so on. For aspect *Decoration*, Limbic produces *sleek*, *ambiance*, and so on. By incorporating constraints from discourse relations, Limbic yields aspects that are more sentiment coherent. For example, we see that positive aspect *Portion* in ASUM contains the negative word *small* whereas words in aspect *Portion* in Limbic are all positive.

We observe that restaurants associate more complex aspects than hotels—presumably, because of the large variety of cuisines and thus, on average, smaller data relevant to a cuisine. Titov and McDonald's (2008b) Multi-Grain LDA (MG-LDA) model performs well for hotel reviews but discovers only few ratable aspects from restaurant reviews, which they ascribe to the relatively small occurrences of words describing aspects for specific cuisines (e.g., *Italian*) and general categories (e.g., *Meat*), compared with the words describing major aspects, such as *Service*. In contrast, Limbic discovers words describing specific cuisines, such as *Mexican* and *Seafood*.

3.3 Quantitative Evaluation

Whether topics (word clusters) are semantically cohesive is an important factor in assessing topic modeling approaches. Normalized Pointwise Mutual Information (NPMI) (Lau et al., 2014) has strong correlation with human-judged topic coherence ratings and is widely used for accessing topic modeling approaches (Nguyen et al., 2015a,b;

Table 4: Topic coherence: Hotel reviews.

NPMI	T=10	T=20	T=30	T=40	T=50	T=60
AT	3.64	4.04	4.37	4.49	4.86	5.14
AATS	5.63	9.08	10.41	10.78	11.05	11.00
JST	8.99	10.78	11.45	11.54	11.56	11.46
ASUM	9.48	10.64	11.02	11.33	11.39	11.56
Limbic	**16.04**†	**17.16**†	**17.75**†	**17.65**†	**17.16**†	**16.60**†
W2V	T=10	T=20	T=30	T=40	T=50	T=60
AT	0.10	0.10	0.10	0.09	0.09	0.09
AATS	0.13	0.16	0.18	0.18	0.18	0.18
JST	0.15	0.18	0.18	0.19	0.19	0.19
ASUM	0.17	0.18	0.18	0.18	0.18	0.18
Limbic	**0.35**†	**0.37**†	**0.38**†	**0.37**†	**0.35**†	**0.34**†

Table 5: Topic coherence: Restaurant reviews.

NPMI	T=10	T=20	T=30	T=40	T=50	T=60
AT	5.64	5.21	5.30	5.65	6.54	7.94
AATS	6.05	8.02	9.03	9.35	9.90	9.95
JST	9.46	11.13	11.73	11.92	**12.14**	**12.31**
ASUM	8.81	9.7	9.92	10.09	10.07	10.04
Limbic	**11.72**†	**13.41**†	**13.77**†	**13.06**†	12.08	11.30
W2V	T=10	T=20	T=30	T=40	T=50	T=60
AT	0.16	0.15	0.14	0.13	0.13	0.14
AATS	0.11	0.14	0.16	0.17	0.18	0.18
JST	0.21	0.21	0.20	0.20	0.20	0.19
ASUM	0.20	0.19	0.18	0.18	0.17	0.17
Limbic	**0.28**†	**0.29**†	**0.27**†	**0.25**†	**0.25**†	**0.25**†

Yang et al., 2017). More recently, O'Callaghan et al. (2015) propose a topic coherence measure, W2V, based on word embeddings. For completeness, we adopt both metrics. Topics with higher scores of NPMI and W2V are semantically more coherent. We compare Limbic with four baselines: AT, JST, ASUM, and AATS, using both TripUser and YelpUser based on the top 15 words in each sentiment-aspect pair. For each number of aspects, we perform five-fold cross-validation. We perform the two-tailed exact permutation test (Good, 2005) on the improvement of Limbic over the best performing baseline. (Throughout, $*$ and $\dagger$ indicate significance at 0.05 and 0.001, respectively.)

Table 4 shows average NPMI and W2V scores of each model on hotel reviews for different numbers of aspects. We observe that Limbic statistically outperforms the other models for both metrics in all settings. Limbic yields substantial improvements, with average gains over the second best models of 6.00 and 0.18 in NPMI and W2V, respectively, which validates that the incorporation of semantic regularities helps Limbic promote semantically equivalent and related words into the same aspect-sentiment pair. Of the baseline models, AT yields the lowest topic coherence. AATS outperforms AT but does not perform well when the number of aspects is small, possibly due to the undesirable mixture of words with different aspects, topics, and sentiments in individual sentences. ASUM, and JST yield comparable results that are consistently better than AATS. Table 5 shows similar conclusions for restaurant reviews.

3.4 Sentiment Classification

We now evaluate Limbic for document-level sentiment classification vis à vis JST, ASUM, and AATS. For comparison purposes, we add a supervised baseline, BiLSTM, using the bidirectional LSTM model (Schuster and Paliwal, 1997). BiLSTM uses 100 as hidden state size and 0.2 as both the recurrent dropout rate and the dropout rate in the last layer. For training, we run 20 epochs with a minibatch size of 1,000. We use two datasets, TripUser and YelpUser. To collect ground-truth labels, we use integer ratings (three and above as positive and rest as negative). Note that our review datasets are imbalanced. Our results are based on five-fold cross-validation (80% of each author's reviews for training and 20% for testing) with the two-tailed exact permutation test. As our principal evaluation metrics, we adopt accuracy; the receiver operating characteristic (ROC) curve; and area under the curve (AUC). ROC and AUC are standard metrics used for evaluating classifiers on data with class imbalance (Bradley, 1997; Hoens and Chawla, 2013).

Tables 6 and 7 report accuracy and AUC on hotel and restaurant reviews. AATS yields high accuracy but low AUC due to a strong bias toward the majority class. Compared with AATS, JST yields higher AUC for both datasets but lower accuracy for TripUser. ASUM outperforms JST, indicating that sentences are more effective as units of sentiment analysis than bags of words. Limbic significantly outperforms ASUM in all settings. For hotel reviews, Limbic attains average gains of 4.0% and 2.3% in accuracy and AUC, respectively. For restaurant reviews, Limbic yields average gains of 5.1% and 3.0% in accuracy and AUC, respectively. In Figure 3 and 4, we compare the ROC curves of Limbic with baselines. The ROC curves show how the true positive rate (TPR) (vertical axis) varies with the false positive rate (FPR) (horizontal axis) by moving the decision boundary. We see that for all FPRs, Limbic yields the highest TPRs. Its ROC curves dominate other models' curves. The results demonstrate that, among all models, Lim-

Table 6: Accuracy and AUC of sentiment classification on hotel reviews.

	T=10		T=20		T=30		T=40		T=50		T=60	
	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
BiLSTM	0.907	0.820	0.907	0.820	0.907	0.820	0.907	0.820	0.907	0.820	0.907	0.820
AATS	0.729	0.485	0.794	0.454	0.809	0.443	0.824	0.475	0.835	0.468	0.839	0.482
JST	0.601	0.813	0.609	0.818	0.634	0.826	0.641	0.830	0.654	0.835	0.665	0.836
ASUM	0.793	0.828	0.804	0.832	0.819	0.829	0.835	0.838	0.850	0.835	0.872	0.829
Limbic	**0.838**†	**0.849***	**0.859**†	**0.853***	**0.868**†	**0.857***	**0.870**†	**0.859***	**0.885**†	**0.858***	**0.890**	**0.858***

Table 7: Accuracy and AUC of sentiment classification on restaurant reviews.

	T=10		T=20		T=30		T=40		T=50		T=60	
	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
BiLSTM	0.876	0.841	0.876	0.841	0.876	0.841	0.876	0.841	0.876	0.841	0.876	0.841
AATS	0.763	0.500	0.789	0.472	0.785	0.474	0.804	0.477	0.813	0.499	0.816	0.492
JST	0.579	0.708	0.589	0.713	0.595	0.724	0.606	0.731	0.629	0.727	0.642	0.733
ASUM	0.773	0.775	0.799	0.778	0.811	0.778	0.836	0.775	0.859	0.752	0.867	0.744
Limbic	**0.872**†	**0.797***	**0.873**†	**0.803**†	**0.874**†	**0.795**†	**0.876**†	**0.795**†	**0.877**†	**0.798**‡	**0.876**†	**0.794**†

bic achieves the best tradeoff between positive and negative sentiment classes.

3.5 Model Analysis

To understand the contributions of incorporating authors, discourse relations, and word embeddings, we evaluate variants of Limbic for SEU-level sentiment classification on two datasets: tSEU and tSEU(D). We create tSEU by randomly selecting 200 hotel reviews by seven authors. We manually annotate the sentiments of each SEU, obtaining 2,692 SEUs. We create tSEU(D) by selecting reviews in tSEU containing at least one Comparison or Expansion. We define three variants of Limbic (L): L_A with just authors, no discourse relations or word embeddings; L_{AD} with authors and discourse relations but no word embeddings; L_{AW} with authors and word embeddings but no discourse relations. Table 8 compares Limbic with L_A L_{AD}, and L_{AW}. We observe that for both datasets, incorporating discourse relations improves accuracy. By incorporating word embeddings, L_{AW} yields better accuracy than L_{AD}, showing that word embeddings add more value to Limbic than discourse relations do.

Table 8 (lower part) reports p-values (two-tailed test using a χ^2 distribution) from McNemar's Test on pairwise comparisons (Alpaydin, 2010, p. 501). We see that Limbic is significantly different from each variant, including L_A (omitted for space).

3.6 Related Work

Sentiment and aspect discovery are often based on Latent Dirichlet Allocation (LDA) (Blei et al., 2003). LDA represents a document (for us, a review) as a mixture of topics, each topic being a multinomial distribution over words. The learning process approximates the topic and word distributions based on their co-occurrence in documents. Titov and McDonald's (2008b) model handles global and local topics involved in documents, and their (2008a) framework discovers topics using aspect ratings provided by reviewers. JST (Lin et al., 2012) and ASUM (Jo and Oh, 2011) model a review via multinomial distributions of topics and sentiments and use them to condition the probability of generating words. Kim et al. (2013) extend ASUM by allowing its probabilistic model to discover a hierarchical structure of aspect-based sentiments. Lazaridou et al. (2013) introduce discourse transitions into the document generating process as aspect and sentiment shifters. Although the above models produce good results, they omit author information, which is an intrinsic attribute of opinionated texts.

Table 8: SEU sentiment classification accuracy.

Accuracy	L_A	L_{AD}	L_{AW}	L	
tSEU	0.702	0.723	0.733	**0.750**	
tSEU(D)	0.692	0.715	0.716	**0.735**	
p value	L_{AD}:L_A	L_{AW}:L_A	L_{AD}:L_{AW}	L:L_{AD}	L:L_{AW}
tSEU	0.015	0.003	0.331	0.005	0.002
tSEU(D)	0.017	0.042	0.934	0.055	0.003

Rosen-Zvi et al.'s Author Topic model (AT) (2004) captures authorship by building a topic distribution for each author. When generating a word

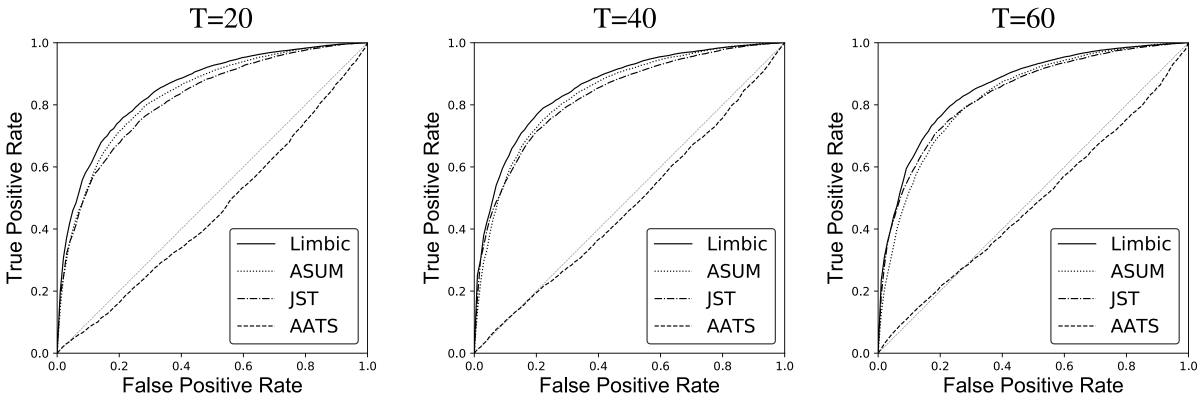

Figure 3: ROC curves comparing the performance of three models on hotel reviews.

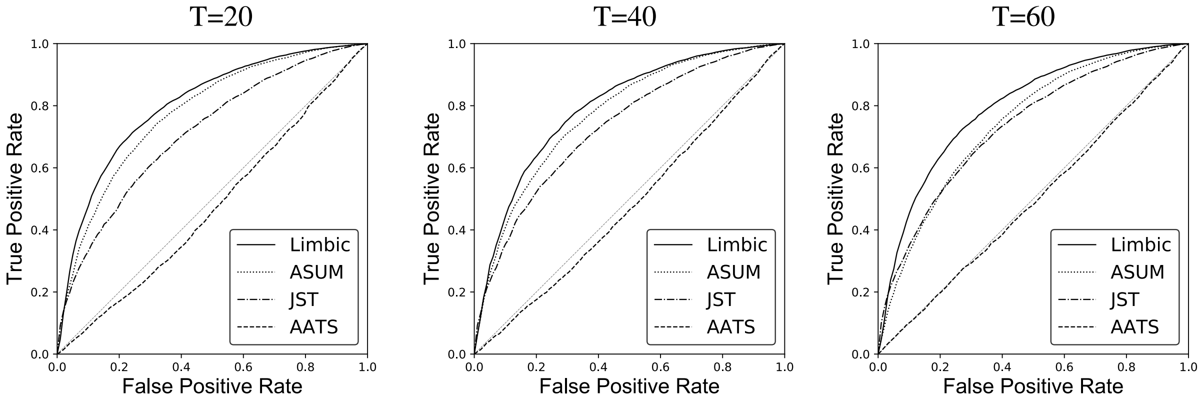

Figure 4: ROC curves comparing the performance of three models on restaurant reviews.

in a document, AT conditions the probability of the topic assignment on the author of the document. Kim et al.'s (2012) topic model captures entities mentioned in documents and models the probability of generating a word as conditioned on both entity and topic. Diao and Jiang's (2013) jointly model topics, events, and users on Twitter. Although these models capture the author associated with a text, they do not handle sentiments.

Mukherjee et al.'s (2014) JAST model jointly considers authors, sentiments, topics, and ratings. JAST does not consider discourse relations and word semantic similarity in its generative process. Poddar et al. (2017) propose a model that jointly considers author, aspect, sentiment, and the non-repetitive generation of aspect sequences. The model uses a Bernoulli process to capture the non-repetitive nature of aspect sequences. This mechanism does not consider discourse relations or syntactic information.

4 Conclusion and Discussion

Limbic provides an unsupervised method to discover aspects and sentiments from opinionated texts. By incorporating authors as a factor, Limbic allows for reviews written by the same or similar authors to exhibit an idiosyncratic preference toward certain aspects and sentiments. It assigns aspects of SEUs by sampling author-specific aspect distributions. This makes the model more suitable for opinionated texts in which aspects and sentiments are tightly bound to authors who follow their specific criteria and preferences when writing reviews. By incorporating a Markov Random Field and word embeddings into its sampling process, Limbic imposes constraints associated with discourse relations, effectively captures word semantic relatedness, and generates word clusters with high topic cohesion and lexical diversity. In future work, we plan to extend Limbic to capture long-distance discourse relations and the influence decay of discourse relations between SEUs as their distance increases.

5 Acknowledgments

Thanks to Chung-Wei Hang and the anonymous reviewers for helpful comments, corrections and inspiration.

References

Ethem Alpaydin. 2010. *Introduction to Machine Learning*, 2nd edition. MIT Press, Cambridge, Massachusetts.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Andrew P. Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159.

Qiming Diao and Jing Jiang. 2013. A unified model for topics, events and users on Twitter. In *Proceedings of the 18^{th} Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1869–1879, Seattle.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43^{rd} Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370, Ann Arbor.

Phillip I. Good, editor. 2005. *Permutation, Parametric, and Bootstrap Tests of Hypotheses*, 3rd edition. Springer Series in Statistics. Springer-Verlag, New York.

T. Ryan Hoens and Nitesh V. Chawla. 2013. Imbalanced datasets: From sampling to classifiers. In Haibo He and Yunqian Ma, editors, *Imbalanced Learning: Foundations, Algorithms, and Applications*, pages 43–59. John Wiley & Sons, Hoboken, New Jersey.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of 15^{th} Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 289–296, Stockholm.

Yohan Jo and Alice Haeyun Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the 4^{th} ACM International Conference on Web Search and Data Mining (WSDM)*, pages 815–824, Hong Kong.

Hyungsul Kim, Yizhou Sun, Julia Hockenmaier, and Jiawei Han. 2012. ETM: Entity topic models for mining documents associated with entities. In *Proceedings of the 12^{th} IEEE International Conference on Data Mining (ICDM)*, pages 349–358, Brussels.

Suin Kim, Jianwen Zhang, Zheng Chen, Alice H. Oh, and Shixia Liu. 2013. A hierarchical aspect-sentiment model for online reviews. In *Proceedings of the 27^{th} AAAI Conference on Artificial Intelligence (AAAI)*, pages 804–812, Bellevue.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14^{th} Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 530–539, Gothenburg.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A Bayesian model for joint unsupervised induction of sentiment, aspect and discourse representations. In *Proceedings of the 51^{st} Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1630–1639, Sofia, Bulgaria.

Limbic. 2018. https://research.csc.ncsu.edu/mas/code/limbic/. Accessed: 08/23/2018.

Chenghua Lin, Yulan He, Richard Everson, and Stefan M. Rüger. 2012. Weakly supervised joint sentiment-topic detection from text. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):1134–1145.

Jun S. Liu. 1994. The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27^{th} Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119, Lake Tahoe.

David M. Mimno, Hanna M. Wallach, Edmund M. Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the 16^{th} Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 262–272, Edinburgh.

Subhabrata Mukherjee, Gaurab Basu, and Sachindra Joshi. 2014. Joint author sentiment topic model. In *Proceedings of the 14^{th} International Conference on Data Mining (SDM)*, pages 370–378, Philadelphia.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015a. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics (TACL)*, 3:299–313.

Thang Nguyen, Jordan L. Boyd-Graber, Jeffrey Lund, Kevin D. Seppi, and Eric K. Ringger. 2015b. Is your anchor going up or down? Fast and accurate supervised topic models. In *Proceedings of the 16^{th} Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 746–755, Denver.

Derek O'Callaghan, Derek Greene, Joe Carthy, and Pádraig Cunningham. 2015. An analysis of the coherence of descriptors in topic modeling. *Expert Systems with Applications*, 42(13):5645–5657.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the* 19^{th} *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha.

Lahari Poddar, Wynne Hsu, and Mong-Li Lee. 2017. Author-aware aspect topic sentiment model to retrieve supporting opinions from reviews. In *Proceedings of the* 22^{nd} *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 472–481, Copenhagen.

Michal Rosen-Zvi, Thomas L. Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the* 20^{th} *Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 487–494, Banff, Canada.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Ivan Titov and Ryan T. McDonald. 2008a. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the* 46^{th} *Annual Meeting on Association for Computational Linguistics (ACL)*, pages 308–316, Columbus, Ohio.

Ivan Titov and Ryan T. McDonald. 2008b. Modeling online reviews with multi-grain topic models. In *Proceedings of the* 17^{th} *International Conference on World Wide Web (WWW)*, pages 308–316, Beijing.

Feng Wang, Weike Pan, and Li Chen. 2013. Recommendation for new users with partial preferences by integrating product reviews with static specifications. In *Proceedings of the* 21^{st} *International Conference on User Modeling, Adaptation, and Personalization (UMAP)*, pages 281–288, Rome.

Weiwei Yang, Jordan L. Boyd-Graber, and Philip Resnik. 2017. Adapting topic models using lexical associations with tree priors. In *Proceedings of the* 22^{nd} *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1901–1906, Copenhagen.

Yelp. 2017. Yelp dataset challenge. https://www.yelp.com/dataset_challenge/. Accessed: 05/20/2018.

Zhe Zhang and Munindar P. Singh. 2014. ReNew: A semi-supervised framework for generating domain-specific lexicons and sentiment analysis. In *Proceedings of the* 52^{nd} *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 542–551, Baltimore.

An Interpretable Neural Network with Topical Information for Relevant Emotion Ranking

Yang Yang† **Deyu Zhou***† **Yulan He**§
†School of Computer Science and Engineering, Key Laboratory of Computer Network
and Information Integration, Ministry of Education, Southeast University, China
§Department of Computer Science, University of Warwick, UK
{yyang, d.zhou}@seu.edu.cn, y.he@cantab.net

Abstract

Text might express or evoke multiple emotions with varying intensities. As such, it is crucial to predict and rank multiple relevant emotions by their intensities. Moreover, as emotions might be evoked by hidden topics, it is important to unveil and incorporate such topical information to understand how the emotions are evoked. We proposed a novel interpretable neural network approach for relevant emotion ranking. Specifically, motivated by transfer learning, the neural network is initialized to make the hidden layer approximate the behavior of topic models. Moreover, a novel error function is defined to optimize the whole neural network for relevant emotion ranking. Experimental results on three real-world corpora show that the proposed approach performs remarkably better than the state-of-the-art emotion detection approaches and multi-label learning methods. Moreover, the extracted emotion-associated topic words indeed represent emotion-evoking events and are in line with our common-sense knowledge.

1 Introduction

With the growth of social web, people tend to share their opinions, feelings and attitudes on the social platforms such as online news sites and blogs. Emotion detection can enhance the understanding of users' emotional states, which is useful in many downstream applications such as human-computer interaction and personalized recommendation. Therefore, it is crucial to predict emotions from texts accurately (Picard and Picard, 1997).

Research on emotion detection can be roughly categorized into two types: generative model based and discriminative model based. Generative model based approaches (Bao et al., 2012; Rao et al., 2014a) usually build on topic models and assume texts are generated from emotions and hidden topics. While these models can extract emotion-associated topics, they perform less satisfactorily in emotion classification since they are not optimized directly to minimize the misclassification rate. Discriminative model based approaches consider each emotion category as a class label and typically cast emotion detection as a classification problem. Approaches to the prediction of both multiple emotions and their intensities include (Zhou et al., 2018, 2016; Wang and Pal, 2015). Those approaches usually assumed word-level representations and ignored the latent topical information behind words, therefore failed to effectively distinguish different emotions carried by the same word in different topical contexts.

In this paper, we focus on relevant emotion ranking (RER) by differentiating relevant emotions from irrelevant ones and only learning the rankings of relevant emotions while ignoring the irrelevant ones. A neural network with a novel loss function is proposed to tackle the RER problem. A topic representing a real-world event, an abstract entity, or an object could indicate the subject or context of the emotion. Different topics might contain or invoke different emotions (Stoyanov and Cardie, 2008). Incorporating such latent topics is essential for discovering topic-associated emotions. Motivated by transfer learning, we incorporate hidden topics and the topic distributions generated from a topic model into a neural network for RER. The main contributions of the paper are summarized below:

- A novel Interpretable Neural Network for Relevant Emotion Ranking (INN-RER) is proposed. A novel error function is employed to optimize the whole network for parameter estimation. To the best of our knowledge, it is the first neural network based approach for RER.

---

[*] Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3423–3432
Brussels, Belgium, October 31 - November 4, 2018.

- To understand how the emotions are evoked, the neural network is initialized to make its hidden layer approximate the behavior of topic models so that the topical information is unveiled and incorporated.

- Experimental results on three different real-world corpora show that the proposed method can effectively deal with the emotion detection problem and perform better than the state-of-the-art emotion detection methods and multi-label learning methods. Moreover, emotion-association topic words extracted by INN-RER indeed represent emotion-evoking events.

2 Related Work

In general, approaches for emotion detection can be divided into two categories: generative model based and discriminative model based. Generative model based approaches typically built on topic models. For example, the emotion-topic model (Bao et al., 2012) was proposed by adding an extra emotion layer into traditional topic models to capture the generation of both emotions and topics from text at the same time. Other topic model based approaches such as affective topic model (Rao et al., 2014a), multi-label supervised topic model and sentiment latent topic model (Rao et al., 2014b) also modeled the emotions and topics simultaneously. Contextual sentiment topic model (Rao, 2016) assumed each word is either drawn from a background theme, a contextual theme or a topic and explicitly distinguished between context-dependent and context-independent topics.

For discriminative model based methods, emotion detection is often casted as a classification problem by considering each emotion category as a class label. If only choosing the strongest emotion as the label for a given text, emotion detection is essentially a single-label classification problem. Lin et al. (2008) studied the classification of news articles into different categories based on readers' emotions with various combinations of feature sets. Strapparava and Mihalcea (2008) proposed several knowledge-based and corpus-based methods for emotion classification. Quan et al. (2015) proposed a logistic regression model with emotion dependency for emotion detection. Latent variables were introduced to model the latent structure of input text. Li et al. (2016) combined bi-term topic model and conventional neural network to detect single social emotion from short texts. To predict multiple emotions simultaneously, emotion detection can be solved using multi-label classification. Bhowmick (2009) presented a method for classifying news sentences into multiple emotion categories using an ensemble based multi-label classification technique. Wang and Pal (2015) output multiple emotions with intensities using non-negative matrix factorization with several novel constraints such as topic correlation and emotion bindings. To predict multiple emotions with different intensities in a single sentence, Zhou et al. (2016) proposed a novel approach based on emotion distribution learning. Following this way, a relevant label ranking framework for emotion detection was proposed for predict multiple relevant emotions as well as the ranking of emotions based on their intensities (Zhou et al., 2018).

Our work is partly inspired by (Zhou et al., 2018) for relevant emotion ranking, but with the following differences: (1) our model takes into account latent topics in texts for emotion detection, which was ignored in the model proposed in (Zhou et al., 2018); (2) our model is built upon topic models and neural networks with a novel objective function defined to consider the interplay between topics and emotions, while the model in (Zhou et al., 2018) was developed based on a ranking framework with a linear objective function which was not able to describe complex relations between the input texts and their emotions.

3 The Proposed Approach

Assuming a set of T emotions $L = \{e_1, e_2, ...e_T\}$ and a set of n text instances $X = \{x_1, x_2, x_3, ..., x_n\}$, each instance $x_i \in \mathbb{R}^d$ is associated with a ranked list of its relevant emotions $R_i \subseteq L$ and also a list of irrelevant emotions $\overline{R_i} = L - R_i$. Relevant emotion ranking aims to learn a score function $\mathbf{g}(x_i) = [g_1(x_i), ..., g_T(x_i)]$ which assigns a score $g_j(x_i)$ to each emotion $e_j, (j \in \{1, ..., T\})$. In order to differentiate relevant emotions from irrelevant ones, we need to define a threshold Θ which could be simply set to a fixed value or learned from data (Fürnkranz et al., 2008). Those emotions with scores lower than the threshold will be considered as irrelevant and hence discarded. The identification of relevant emotions and their ranking can be obtained simultaneously according to their scores assigned by the learned ranking function **g**. As mentioned before, it is unnecessary to consider the rankings of irrelevant emotions since they might introduce errors into the model during the learning process.

We propose an Interpretable Neural Network for Relevant Emotion Ranking (INN-RER) built upon a multi-layer feed-forward neural network. Instead of using the simple sum-of-squares error function, a novel loss function is designed and employed. Accordingly, a new learning algorithm is proposed to minimize the new loss function. Furthermore, motivated by transfer learning, topical information generated from a topic model is transferred into the neural network by making its hidden layer approximate the behavior of topic models.

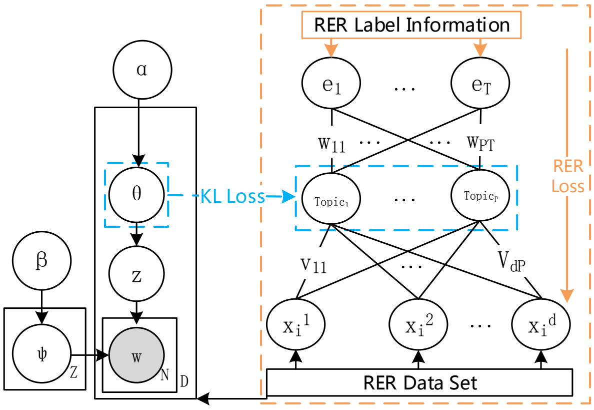

Figure 1: The overall framework of Interpretable Neural Network for Relevant Emotion Ranking (INN-RER).

The overall framework of INN-RER is shown in Figure 1. The left part is a typical topic model (Blei et al., 2003). It is designed for discovering the main topics that pervade a large unstructured collection of documents. A document is allowed to contain a mixture of topics with different weights. As such, a document d can be represented by its topic distribution θ_d. The right part is a three-layer neural network. It has d input units corresponding to the d-dimensional feature vector of each training sample x_i, T output units corresponding to all possible emotion labels, and one hidden layer with P hidden units corresponding to the hidden topics. The input layer is fully connected to the hidden layer with weights $V = [v_{qh}](1 \leq q \leq d, 1 \leq h \leq P)$ and the hidden layer is fully connected to the output layer with weights $W = [w_{hj}](1 \leq h \leq P, 1 \leq j \leq T)$. The bias parameters $\alpha_h(1 \leq h \leq P)$ of the hidden units are considered as weights from an extra input unit with a fixed value of 1. Similarly, the bias parameters $\beta_j(1 \leq j \leq T)$ of the output units are considered as weights from an extra hidden unit, with a fixed value of 1.

The learning process of INN-RER consists of two main steps. Firstly, the first two layers of the network are initialized based on the output of the topic model. The feature transformation in neural network is conducted by minimizing the Kullback-Leibler (KL) divergence between the topic distribution θ produced by the topic model and the approximated distribution $[Topic_1, Topic_2, ..., Topic_P]$ learned by the first two layers of the neural network, which is denoted by the blue rectangular dash line boxes in Figure 1. Then, the whole network is learnt and fine-tuned based on the novel loss function, which is denoted as the orange rectangular dash line boxes. Each step will be described in details in the following subsections.

3.1 INN-RER Initialization

As the number of hidden neurons and its semantic meaning is usually treated as a black box in conventional neural networks, the generated topics from the topic model are employed for guiding the construction of the hidden layer in the proposed neural network. By doing that, semantic topic information is incorporated to enhance the interpretability and accuracy of the proposed neural network. For a particular text sample x_i in training set G, the input layer takes its term-frequency representation x_i^q as the input and feeds it to the hidden layer. Assuming the total number of topics is fixed as P, then the hidden layer would contain P neurons. The topic mixture θ_{x_i} generated from the topic model is approximated by the weights connecting the input and the hidden layers. Mathematically, the initialization procedure learns a function $f(x_i^q|v_{qh}, \alpha_h)$ so that the output of $f(x_i^q|v_{qh}, \alpha_h)$ is as close to θ_{x_i} as possible, where x_i^q, v_{qh}, α_h and θ_{x_i} denote the input, weight vector, bias of the first two layers of the network and the topic distribution of text x_i generated from the topic model, respectively. A softmax function is applied to the output of the hidden layer, i.e., $f(x_i^q|v_{qh}, \alpha_h)$, and the Kullback-Leibler divergence (Leahy, 2006) is employed as follows:

$$L(\theta, f) = \theta \log \frac{\theta}{f} + (1-\theta) \log \frac{1-\theta}{1-f} \qquad (1)$$

where θ denotes a topic distribution derived from the topic model, and f denotes the output of the hidden layer. The KL divergence is a measure of the difference between two distributions. It is always non-negative and equals to zero when the two distributions are the same. As shown in Equation 1, the KL divergence can describe the difference between the topic distribution generated from topic models and the approximate distributions learned in the initialization procedure. Note that the topic distribution for a document generated by the topic model is used as the supervision information for initializing INN-RER. Thus, maximizing the log-likelihood is equivalent to minimizing the KL divergence according to Equation 1, and its gradients are as follows:

$$\frac{\partial L(\theta, f)}{\partial v_{qh}} = -(\theta_{x_i,h} - f_h(x_i^q | v_{qh}, \alpha_h)) \cdot x_i^q \qquad (2)$$

$$\frac{\partial L(\theta, f)}{\partial \alpha_h} = -(\theta_{x_i,h} - f_h(x_i^q | v_{qh}, \alpha_h)) \qquad (3)$$

According to the gradient descent method, the first two layers can be initialized iteratively by Equation 2 and 3. The initialization procedure for INN-RER is shown in Algorithm 1. η_{init} with the subscript $init$ represents the learning rate during the initialization procedure and λ is the penalty term. Note that the first two layer should be learnt from topic model as much as possible in order to incorporate topic information, thus the learning rate term η_{init} should be larger than the learning rate during training procedure.

3.2 INN-RER Learning

This step aims to optimize the three-layer neural network to tackle the relevant emotion ranking problem. It can adjust the neural network initialized at previous step at the same time. An intuitive way is to define the global error function for the network on the training set. However, some important characteristics of relevant emotion ranking, such as ranking, not considering irrelevant emotions, are not considered in the classical back propagation algorithm (Rumelhart et al., 1988).

Algorithm 1 Algorithm of INN-RER Initialization.

```
Input: $x_i^q$: Term frequency of text $x_i$; $\theta_{x_i}$: Topic distribution of text $x_i$
Output: $\Delta v, \Delta\alpha$: gradient approximation of initialization procedure
Initialize $\Delta v, \Delta\alpha$ as random values
for each iteration do
    for each text $x_i \in G$ do
        for $q = 1, ..., d, h = 1, ..., P$ do
            $\Delta v_{qh} \leftarrow \Delta v_{qh} + \eta_{init} \cdot (\theta_{x_i,h} - f_h(x_i^q | v_{qh}, \alpha_h)) \cdot x_i^q + \lambda \cdot \Delta v_{qh}$
        end for
        for $h = 1, ..., P$ do
            $\Delta\alpha_h \leftarrow \Delta\alpha_h + \eta_{init} \cdot (\theta_{x_i,h} - f_h(x_i^q | v_{qh}, \alpha_h)) + \lambda \cdot \Delta\alpha_h$
        end for
    end for
end for
```

The error function defined in traditional neural network such as mean-square error only focuses on individual label discrimination, i.e. whether a predicted label is correct or not. It does not consider the correlations between different labels of a training instance, e.g., relevant emotions should be ranked higher than irrelevant ones and there is a ranking for relevant emotions according to their intensities. Therefore, to fulfil the requirements of relevant emotion ranking, a novel global error function is defined as follows:

$$\begin{aligned} E = \sum_{i=1}^{n} \sum_{e_t \in R_i} \sum_{e_s \in \prec(e_t)} & \frac{1}{norm_{t,s}} \\ & [\, \exp(-(g_t(x_i) - g_s(x_i))) + \\ & \quad \omega_{ts}(g_t(x_i) - g_s(x_i))^2 \,] \end{aligned} \qquad (4)$$

Here, emotion e_t and emotion e_s are two emotion labels and e_s is less relevant than emotion e_t, represented by $e_s \in \prec (e_t)$. The normalization term $norm_{t,s}$ is used to balance emotion pairs (e_t, e_s) to avoid dominated terms by their set sizes. The term $g_t(x_i) - g_s(x_i)$ measures the difference between two emotion outputs, e_t and e_s, of a given input text x_i. We want the difference as larger as possible. Furthermore, the negation of this difference is fed to the exponential function in order to severely penalize the i-th error term if emotion e_t is much smaller than e_s. As the relationships among different emotions can provide important clues for emotion detection, we further incorporate the information into the loss function as constraints. Here, ω_{ts} is the relationship between emotion e_t and e_s which is calculated by Pearson correlation coefficient (Nicewander, 1988).

The minimization of the global relevant emotion ranking loss function defined in Equation 4 is carried out by gradient descent combined with the back propagation (Rumelhart et al., 1988). For training instance x_i and its label set L_i, the actual output of the j-th output neuron is(omitting the superscript i without loss of generality):

$$g_j = f(netg_j + \beta_j) \tag{5}$$

where β_j is the bias of the j-th output neuron which is a "tanh" function:

$netg_j$ is the input to the j-th output neuron:

$$netg_j = \sum_{h=1}^{P} b_h w_{hj} \tag{6}$$

where w_{hj} is the weight which connects the h-th hidden neuron and the j-th output neuron, and P is the number of hidden neurons, i.e., the topics. b_h is the output of the h-th hidden neuron:

$$b_h = f(netb_h + \alpha_h) \tag{7}$$

where α_h is the bias of the h-th hidden neuron, $f()$ is also a "tanh" function. $netb_h$ is the input to the h-th hidden neuron:

$$netb_h = \sum_{q=1}^{d} x^q v_{qh} \tag{8}$$

where x^q is the q-th dimension of instance x. v_{qh} is the weight which connects the q-th input neuron and the h-th hidden neuron.

"tanh" function is differentiable, the error of the j-th output neuron can be defined as:

$$d_j = \begin{cases} [\frac{1}{norm}exp(-(g_j - g_s)) + 2\omega_{js}(g_j - g_s)] \\ (1+g_j)(1-g_j), if e_j \in R_i \& e_s \in \prec (e_j) \\ [-\frac{1}{norm}exp(-(g_t - g_j)) - 2\omega_{tj}(g_t - g_j)] \\ (1+g_j)(1-g_j), \\ if(e_j \in R_i \& e_j \in \prec (e_t)) or \\ (e_j \in \overline{R_i} \& e_j \in \prec (e_t)) \end{cases} \tag{9}$$

Similarly, the error of the h-th hidden neuron can be defined as:

$$e_h = \left(\sum_{j=1}^{T} g_j w_{hj} \right) (1+b_h)(1-b_h) \tag{10}$$

In order to reduce the error of the neural network INN-RER, we can use gradient descent strategy:

$$\begin{aligned} \Delta w_{hj} &= -\eta \frac{\partial E_i}{\partial w_{hj}} = -\eta \frac{\partial E_i}{\partial netg_j} \frac{\partial netg_j}{\partial w_{hj}} \\ &= \eta d_j \left[\frac{\partial [\sum_{h=1}^{P} b_h w_{hj}]}{\partial w_{hj}} \right] = \eta d_j b_h \end{aligned} \tag{11}$$

$$\begin{aligned} \Delta v_{qh} &= -\eta \frac{\partial E_i}{\partial v_{qh}} = -\eta \frac{\partial E_i}{\partial netb_h} \frac{\partial netb_h}{\partial v_{qh}} \\ &= \eta e_h \left[\frac{\partial [\sum_{q=1}^{d} x^q v_{qh}]}{\partial v_{qh}} \right] = \eta e_h x^q \end{aligned} \tag{12}$$

the biases are updated as follows:

$$\Delta \beta_j = \eta d_j ; \Delta \alpha_h = \eta e_h \tag{13}$$

where η is the learning rate.

The training process of the neural network is presented in Algorithm 2.

Algorithm 2 Algorithm of INN-RER Learning.

Input: x_i^q: Term frequency of text x_i; $\Delta v, \Delta \alpha$: Parameters after initialization; L: emotion labels

Output: A predictable neural network INN-RER.

```
Initialize INN-RER network parameters from Algorithm 1
for each iteration do
    for each text $x_i \in G$ do
        Forward compute output of INN-RER's score function g given $x_i$.
        Backward compute the gradient according to g and $L$ based on the relevant emotion ranking loss function with learning rate of $\eta_{learn}$ and penalty term $\lambda$.
    end for
end for
```

4 Experiments

We evaluate the proposed approach on the following three corpora:

Sina Social News (News) was collected from the Sina news *Society* channel where readers can choose one of the six emotions such as *Amusement*, *Touching*, *Anger*, *Sadness*, *Curiosity*, and *Shock* after reading a news article. As Sina is one of the largest online news sites in China, it is sensible to carry out experiments to explore the readers' emotion (social emotion). News articles with less than 20 votes were discarded since few votes can not be considered as proper representation of social emotion. In total, 5,586 news articles published from January 2014 to July 2016 were kept, together with the readers' emotion votes.

Ren-CECps corpus (Blogs) (Quan and Ren, 2010) contains 1,487 blogs in Chinese. Each document is annotated with eight basic emotions from writer's perspective, including *anger*, *anxiety*, *expect*, *hate*, *joy*, *love*, *sorrow* and *surprise*, together with their emotion scores indicating the level of emotion intensity in the range of $[0, 1]$. Higher scores represent higher emotion intensity.

SemEval (Strapparava and Mihalcea, 2007) is an English data set containing 1,250 news headlines extracted from Google news, CNN, and many other portals. The news headlines are typically short. Each headline was manually scored in a fine-grained valence scale of 0 to 100 across 6 emotions (i.e., *anger*, *disgust*, *fear*, *joy*, *sad* and *surprise*). After pruning 4 items with the total scores equal to 0, 1246 headlines are got for the experiments.

News		Blogs		SemEval	
Category	#Votes	Category	#Scores	Category	#Scores
Touching	694,006	Joy	349.2	anger	12042
Shock	572,651	Hate	174.2	disgust	7634
Amusement	869,464	Love	610.6	fear	20306
Sadness	837,431	Sorrow	408.4	joy	23613
Curiosity	212,559	Anxiety	422.6	sad	24039
Anger	1,109,315	Surprise	59.2	surprise	21495
		Anger	116.4		
		Expect	385.5		
All	4,295,426	All	2526.1	All	109,129

Table 1: Statistics for the three corpora used in our experiments.

The statistics of the three corpora are shown in Table 1. The first two corpora were preprocessed by using the python jieba segmenter[1] for word segmentation and filtering. The third corpus SemEval is in English and can be tokenized by white spaces. Stop words and words appeared only once or in less than two documents were removed to alleviate data sparsity. Next, TF-IDF (term frequency-inverse document frequency) was used to extract the features from text. TF-IDF is a numerical statistic method that is designed to reflect how important a word is to a document in a corpus. In our experiments, we set the dimension of each text representation to 2,000 according to the ranking of the TF-IDF weights with each dimension of term-frequency(TF) features. After that, the text representations are fed into the proposed INN-RER method.

η_{init}, η_{learn}, λ, the number of iterations and the number of topics are set to 0.9, 0.1, 0.001, 100 and 60 respectively. The parameters were chosen by 10-fold cross-validation. The topic distribution used in INN-RER are derived in different ways. For long text such as News and Blogs, Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is employed for generating topic distributions. For short texts in Semeval, bi-term topic model (BTM) (Cheng et al., 2014) was used, since short text typically contains a few words which results in sparse word co-occurrence patterns. BTM is a variant of LDA which effectively infers the latent topic distribution of short text by modeling the generation of bi-terms in the whole corpus and it alleviates the problem of sparsity at the document level. For each method, 10-fold cross validation is conducted using the same feature construction method to get the final performance.

Evaluation metrics typically used in multi-label learning and label ranking are employed which are different from those of classical single-label learning systems (Sebastiani, 2001). The detailed explanation of evaluation metrics are presented in Table 2. Note that metrics from PRO Loss to $F1_{exam}$ work by evaluating performance on each test example separately and returning the mean value across test set. MicroF1 and MacroF1 work by evaluating performance on each emotion category separately and returning the macro/micro-averaged value across all emotion categories.

4.1 Experimental Results

There are several approaches addressing multiple emotions detection from texts. Three generative model based baselines and three discriminative model based baselines are chosen.

- **Emotion Distribution Learning (EDL)** (Zhou et al., 2016) learns a mapping

---

[1] https://github.com/fxsjy/jieba

Corpus	Category	Method	Criteria								
			PL(↓)	HL(↓)	RL(↓)	OE(↓)	AP(↑)	Cov(↓)	F1(↑)	MiF1(↑)	MaF1(↑)
News	Generative	MSTM	0.3343	0.4065	0.3097	0.2123	0.6677	3.3202	0.5666	0.5853	0.5044
		SLTM	0.3205	0.3639	0.2753	0.2008	0.7326	2.9863	0.6095	0.6429	0.4899
		ATM	0.3192	0.3743	0.2507	0.1947	0.7490	2.9369	0.6127	0.6412	0.4885
	Discriminative	EDL	0.2348	0.2510	0.1616	0.2243	0.8372	2.1940	0.6260	0.6454	0.5703
		EmoDetect	0.2157	0.2575	0.1538	0.1627	0.8605	2.1761	0.6697	0.6739	0.5359
		RER	0.2142	0.2498	0.1491	0.1513	0.8633	2.1989	0.6820	0.6919	0.6198
	Our model	**INN-RER(-t)**	0.1998	0.2420	0.1393	0.1456	0.8715	2.1377	**0.7116**	0.7137	0.6242
		INN-RER	**0.1973**	**0.2312**	**0.1353**	**0.1331**	**0.8764**	**2.1339**	0.7108	**0.7161**	**0.6282**
Blogs	Generative	MSTM	0.3567	0.4171	0.3030	0.4761	0.6046	3.7005	0.5236	0.4978	0.4758
		SLTM	0.3148	0.3769	0.2397	0.4598	0.6547	**3.2513**	0.5757	0.5865	0.5283
		ATM	0.3493	0.3890	0.2885	0.4385	0.6278	3.4278	0.5105	0.5260	0.5026
	Discriminative	EDL	0.3385	0.3916	0.2550	0.4206	0.6962	4.2491	0.5060	0.5396	0.4131
		EmoDetect	0.3115	0.3848	0.2123	0.2880	0.7617	4.1650	0.5340	0.5492	0.4387
		RER	0.3007	0.3657	0.2043	0.2728	0.7746	4.1638	0.5957	0.6084	**0.5342**
	Our model	**INN-RER(-t)**	0.2868	0.3268	0.1993	0.2695	0.7751	3.9653	0.6132	0.6165	0.5069
		INN-RER	**0.2829**	**0.3209**	**0.1924**	**0.2626**	**0.7784**	3.6418	**0.6187**	**0.6225**	0.5133
SemEval	Generative	MSTM	0.3524	0.3835	0.2796	0.3698	0.7653	3.1986	0.6902	0.7133	0.5854
		SLTM	0.3155	0.3253	0.2370	0.3150	0.8052	2.9589	0.7016	0.7278	0.5889
		ATM	0.3138	0.3276	0.2389	0.3767	0.8302	2.8976	0.7039	**0.7292**	0.5244
	Discriminative	EDL	0.4130	0.4291	0.3401	0.3875	0.7345	3.3433	0.4002	0.4136	0.3813
		EmoDetect	0.3176	0.3167	0.2411	0.2308	0.8241	3.0439	0.6275	0.6245	0.5385
		RER	**0.2907**	0.3128	0.2389	**0.2220**	0.8302	2.9963	0.6839	0.6898	**0.6283**
	Our model	**INN-RER(-t)**	0.3213	0.3026	0.2331	0.2388	0.8364	2.8773	0.7019	0.7118	0.5973
		INN-RER	0.3194	**0.3005**	**0.2302**	0.2261	**0.8379**	**2.8632**	**0.7081**	0.7156	0.6093

Table 3: Experimental results of the proposed approach and the baselines. 'PL' represent Pro Loss, 'HL' represents Hamming Loss, 'RL' represents ranking loss, 'OE' represents one error, 'AP' represent average precision, 'Cov' represent coverage, 'F1' represents $F1_{exam}$, MiF1' represents MicroF1, 'MaF1' represents MacroF1. "↓" indicates "the smaller the better", while "↑" indicates "the larger the better". The best performance on each evaluation measure is highlighted by boldface.

Name	**Definition**
PRO Loss	$\frac{1}{n}\sum_{i=1}^{n}\sum_{e_t\in R_i\cup\{\Theta\}}\sum_{e_s\in\prec(e_t)}\frac{1}{norm_{t,s}}l_{t,s}$ $l_{t,s}$ is a modified 0-1 error;$norm_{t,s}$is the set size of label pair(e_t,e_s)
Hamming Loss	$\frac{1}{nT}\sum_{i=1}^{n}\lvert\hat{R}_i\triangle R_i\rvert$ The predicted relevant emotions: $\hat{R}_i$.
Ranking Loss	$\frac{1}{n}\sum_{i=1}^{n}(\sum_{(e_t,e_s)\in R_i\times\overline{R_i}}\delta[g_t(x_i)<g_s(x_i)])/(\lvert R_i\rvert\times\lvert\overline{R_i}\rvert)$ where δ is the indicator function.
One Error	$\frac{1}{n}\sum_{i=1}^{n}\delta[\underset{\mathbf{e_t}}{\text{argmax}}\, g_t(x_i)\notin R_i]$
Average Precision	$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lvert R_i\rvert}\times$ $(\sum_{t:e_t\in R_i}\lvert\{e_s\in R_i\vert g_s(x_i)>g_t(x_i)\}\rvert)/(\lvert\{e_s\vert g_s(x_i)>g_t(x_i)\}\rvert)$
Coverage	$\frac{1}{n}\sum_{i=1}^{n}\max_{t:e_t\in R_i}\lvert\{e_s\vert g_s(x_i)>g_t(x_i)\}\rvert$
$F1_{exam}$	$\frac{1}{n}\sum_{i=1}^{n}2\lvert R_i\cap\hat{R}_i\rvert/(\lvert R_i\rvert+\lvert\hat{R}_i\rvert)$
MicroF1	$F1(\sum_{t=1}^{T}TP_t,\sum_{t=1}^{T}FP_t,\sum_{t=1}^{T}TN_t,\sum_{t=1}^{T}FN_t)$
MacroF1	$\frac{1}{T}\sum_{t=1}^{T}F1(TP_t,FP_t,TN_t,FN_t)$

Table 2: Evaluation criteria for the Multi-Label Learning (MLL) methods. TP_t, FP_t, TN_t, FN_t represent the number of true positive, false positive, true negative, and false negative test examples with respect to emotion t respectively. $F1(TP_t, FP_t, TN_t, FN_t)$ represent specific binary classification metric F1 (Manning et al., 2008).

function from texts to their emotion distributions based on label distribution learning.

- **EmoDetect** (Wang and Pal, 2015) outputs the emotion distribution based on a dimensionality reduction method using non-negative matrix factorization which combines several constraints such as emotions bindings and topic correlations.

- **RER** (Zhou et al., 2018) predicts multiple emotions and their rankings from text based on relevant emotion ranking using support vector machines.

- **Multi-label supervised topic model (MSTM) and Sentiment latent topic model (SLTM)** (Rao et al., 2014b): As the variants of supervised topic models, MSTM and SLTM connect latent topics with evoked emotions of

Touching		Anger		Amusement	
Topic 1	Topic 2	Topic 1	Topic 2	Topic 1	Topic 2
救人(save)	教师(teacher)	歹徒(ruffian)	犯罪(sin)	男女(men and women)	网上(network)
照顾(take care of)	辛苦(hard)	强行(force)	嫌疑人(suspect)	宾馆(hotel)	醉酒(drunkenness)
身亡(sacrifice)	落水(fall into water)	猥亵(obscenity)	徒刑(imprisonment)	服务(service)	检察院(procuratorate)
治疗(cure)	年轻(youth)	女童(girl)	打人(beat)	照片(photo)	违法(illegal)
生命(life)	病情(state of an illness)	杀害(murder)	殴打(hit)	报警(call the police)	罚款(penalty)
老人(older)	坚持(persist)	造成(cause)	工地(construction site)	鉴定(authenticate)	调查(investigate)
感谢(grateful)	群众(public)	派出所(police station)	交警(traffic police)	诈骗(defraud)	违规(get out of line)
医院(hospital)	车祸(traffic accident)	作案(commit a crime)	采访(interview)	网络(internet)	现金(cash)
感动(moved)	感动(touching)	死亡(death)	曝光(exposure)	离婚(divorce)	警官(police officer)
Sadness		Curiosity		Shock	
Topic 1	Topic 2	Topic 1	Topic 2	Topic 1	Topic 2
失踪(disappear)	车祸(car accident)	家长(parents)	监控(monitoring)	抢劫(rob)	菜刀(kitchen knife)
不幸(misfortune)	小偷(thief)	中国(China)	妇女(women)	尸体(corpse)	脖子(neck)
去世(pass away)	公安(public security)	婚姻(marriage)	春节(spring festival)	紧急(emergency)	重症(sever illness)
杀害(murder)	围观(watch)	健康(health)	医院(hospital)	现场(scene)	地铁(subway）
犯罪(crime)	鉴定(identify)	女子(women)	怀孕(pregnancy)	安全(security)	新闻(news)
遭到(suffer)	道歉(apologize)	年轻(young)	早上(morning)	治疗(cure)	竟然(unexpectedly)
公安局(Public Security Bureau)	激动(excite)	结婚(marry up)	抢救(rescue)	生命(life)	银行(bank)
出事(have an accident)	执法(enforce the law)	男性(men)	无效(in vain)	检查(examine)	赔偿(compensate)
媒体(media)	派出所(police station)	现金(money)	喜欢(like)	家属(family member)	消费(consume)

Figure 2: The top topic words under each emotion category from the News corpus.

texts. MSTM first generates a set of topics from words, and then samples emotions from each topic. SLTM generates topics directly from emotions.

- **Affective topic model (ATM)** (Rao et al., 2014a) employs the exponential distribution to generate ratings for each emotion.

We also evaluated INN-RER with random initialization instead of the proposed initialization procedure, which is denoted as INN-RER(-t).

Experimental results on the three corpora are summarized in Table 3. It can be observed from the table that: (1) INN-RER outperforms the baselines on almost all evaluation metrics across all the data sets; (2) INN-RER achieves better performance on almost all the evaluation metrics than INN-RER(-t), which further verifies the effectiveness of incorporating the topic information; (3) Both INN-RER and INN-RER(-t) perform remarkably better than RER which is based on linear models. It verifies the effectiveness of using the neural networks for RER task, which are able to learn dynamic and complex functions.

4.2 INN-RER Interpretation

In addition to comparing the performance of the proposed model with several baselines, we also present the experimental results from the perspective of result interpretation to fully understand INN-RER. The topic words of each emotion in three corpora are extracted according to the ranking of weights learned by INN-RER, i.e., the probabilities of topics conditioned on emotions (weights between the hidden layer and the output layer) and words conditioned on topics (weights between the input layer and the hidden layer). Results are shown in Figure 2, 3, 4 respectively. It can be observed that the extracted topics words under each emotion category correspond to a certain event, which evokes the emotion. It is in ac-

Joy	Anger	Sad	Disgust	Fear	Surprise
home	kill	flu	sex	kill	sue
heart	attack	cancer	immigr	danger	korea
game	violenc	terror	scandal	iran	blast
youtub	terror	danger	porn	dead	north
movie	stop	health	charg	state	fight
friend	fire	kill	insist	fear	war
sleep	blast	flood	women	terror	nuclear
miss	death	crash	held	global	shoot
award	condemn	end	girl	attack	protest

Figure 3: The top topic words under each emotion category from the Semeval corpus.

Joy	Hate	Love	Sorrow
花儿(flower)	孤独(lonely)	学习(study)	角落(corner)
新年(ney year)	面对(face with)	比赛(competition)	希望(hope)
宝贝(baby)	无情(heartless)	开心(happy)	天堂(heaven)
享受(enjoy)	重新(again)	感觉(feeling)	寂寞(lonely)
快乐(happy)	情绪(emotion)	心境(mood)	地震(earthquake)
祝福(wish)	失去(lose)	充满(full of)	使命(mission)
宝宝(baby)	脾气(temper)	文化(culture)	男朋友(boyfriend)
开心(joyful)	痛苦(pain)	作品(production)	离开(leave)
微笑(smile)	完全(entirely)	丰富(rich)	无奈(helpless)
Anxiety	Surprise	Anger	Expect
房子(house)	彩虹(rainbow)	离开(leave)	希望(hope)
婚姻(marriage)	北海道(Hokkaido)	离婚(divorce)	责任(responsible)
老公(husband)	忽然(sudden)	无奈(helpless)	女性(women)
错误(error)	记忆(memory)	法律(law)	奥运会(Olympic)
心情(mood)	礼物(gift)	银行(bank)	幸福(happiness)
陌生(strange)	奇迹(miracle)	道德(morality)	行为(action)
家里(family)	据说(reputedly)	情感(emotion)	努力(strive)
上班(on duty)	好奇(curious)	悲伤(sorrow)	以后(later)
城市(city)	季节(season)	自己(self)	精彩(splendid)

Figure 4: The top topic words for each emotion category from the Blogs corpus.

Corpus	Method	Criteria								
		PL(↓)	HL(↓)	RL(↓)	OE(↓)	AP(↑)	Cov(↓)	F1(↑)	MiF1(↑)	MaF1(↑)
News	RANK-SVM	0.2842	0.2872	0.2114	0.2034	0.7967	2.5358	0.5066	0.5656	0.5298
	BP-MLL	0.2118	0.2399	0.1443	0.1544	0.8677	2.1738	0.6881	0.6915	0.6013
	LIFT	0.2224	0.3363	0.1382	0.1411	0.8234	2.1394	0.6646	0.6801	0.6151
	INN-RER	**0.1973**	**0.2312**	**0.1353**	**0.1331**	**0.8764**	**2.1339**	**0.7108**	**0.7161**	**0.6282**
Blogs	RANK-SVM	0.3888	0.3786	0.3356	0.3219	0.7030	4.0801	0.3489	0.3686	0.3210
	BP-MLL	0.2987	0.3281	0.2141	0.2727	0.7267	3.9802	0.5844	0.6065	0.4833
	LIFT	0.3452	0.3817	0.3089	0.3306	0.7557	**3.1290**	0.6053	0.6113	**0.5155**
	INN-RER	**0.2829**	**0.3209**	**0.1924**	**0.2626**	**0.7784**	3.6418	**0.6187**	**0.6225**	0.5133
SemEval	RANK-SVM	0.3452	0.3617	0.3083	0.3006	0.7557	3.1290	0.6253	0.6472	0.5955
	BP-MLL	0.3790	0.3656	0.3605	0.3790	0.7495	3.2097	0.5868	0.6101	0.5402
	LIFT	0.4279	0.4651	0.3627	0.4113	0.7344	3.2823	0.6299	0.6469	**0.6112**
	INN-RER	**0.3194**	**0.3005**	**0.2302**	**0.2261**	**0.8379**	**2.8632**	**0.7081**	**0.7156**	0.6093

Table 4: Comparison with Multi-Label Learning (MLL) Methods. The evaluation criteria are same as in Table 3.

cord with what has been observed in social psychology (Stoyanov and Cardie, 2008). For example, in the Sina corpus, Topic 1 under the emotion *touching* is about **“heroic rescue”**; Topic 1 under the emotion *anger* is about **“sexual molestation of a child”** and Topic 2 under the emotion *sadness* is about an **“car accident”**. In the SemEval and the Blog corpora, we can also find that topic words listed under each emotion category are related to some social events. For example, in the SemEval corpus, the *Joy* topic is about **“home entertainment”** and the *Anger* topic is about **”terrorist attack”**. In the Blog corpus, the *sorrow* topic is about **”earthquake and the lost of their loved ones”**. The extracted emotion-associated topic words unveil how the corresponding emotion is evoked. By incorporating topical information into neural network learning, we are able to obtain more interpretable results from INN-RER.

4.3 Comparison with Multi-Label Methods

Since relevant emotion ranking can be seen as an extension of multi-label learning, the proposed INN-RER is also compared with three widely used well-established multi-label learning methods such as LIFT (Zhang, 2011), Rank-SVM (Zhang and Zhou, 2014) and BP-MLL (Zhang and Zhou, 2006). In our experiments, LIFT used linear kernel and Rank-SVM uses the RBF kernel with the width σ set to 1 using the threshold Θ which is initialized as 0.15 after normalization.

The results of INN-RER in comparison with MLL baselines are presented in Table 4. It can be observed that INN-RER outperforms all the baselines across all the datasets on all evaluation measures most of the time. This further verifies the effectiveness of our proposed INN-RER for multi-label emotion detection due to its consideration of rankings of the relevant emotions and the incorporation of topic models.

5 Conclusion

In this paper, we have proposed a novel interpretable neural network for relevant emotion ranking. Specifically, motivated by transfer learning, the neural network is initialized to make its hidden layer approximate the behavior of a topic model. Moreover, a novel error function is defined to optimize the whole neural network for relevant emotion ranking. Experimental results on three real-world corpora show that the proposed approach performs remarkably better than the state-of-the-art emotion detection approaches and multi-label learning methods. Moreover, the extracted emotion-associated topic words indeed represent emotion-evoking events which are in line with our common-sense knowledge. In the future, we will explore the possibility of learning a topic model and an emotion ranking function simultaneously in a unified framework.

Acknowledgments

The work was supported by the National Key R&D Program of China (No. 2017YFB1002801), the National Natural Science Foundation of China (61772132,61528302) and the Natural Science Foundation of Jiangsu Province of China (BK20161430).

References

Shenghua Bao, Shengliang Xu, Li Zhang, Rong Yan, Zhong Su, Dingyi Han, and Yong Yu. 2012. Mining social emotions from affective text. *IEEE transactions on knowledge and data engineering*, 24(9):1658–1670.

Plaban Kumar Bhowmick. 2009. Reader perspective emotion analysis in text through ensemble based multi-label classification framework. *Computer and Information Science*, 2(4):64.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Xueqi Cheng, Xiaohui Yan, Yanyan Lan, and Jiafeng Guo. 2014. Btm: Topic modeling over short texts. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):2928–2941.

Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. 2008. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153.

D. E. Leahy, D. P. Searson. 2006. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(2):69.

Xiangsheng Li, Jianhui Pang, Biyun Mo, and Yanghui Rao. 2016. Hybrid neural networks for social emotion detection over short text. In *International Joint Conference on Neural Networks*, pages 537–544.

Kevin Hsin-Yih Lin, Changhua Yang, and Hsin-Hsi Chen. 2008. Emotion classification of online news articles from the reader's perspective. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 220–226. IEEE Computer Society.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. An introduction to information retrieval. *Journal of the American Society for Information Science and Technology*, 43(3):824–825.

W. Alan Nicewander. 1988. Thirteen ways to look at the correlation coefficient. *American Statistician*, 42(1):59–66.

Rosalind W Picard and Roalind Picard. 1997. *Affective computing*, volume 252. MIT press Cambridge.

Changqin Quan and Fuji Ren. 2010. Sentence emotion analysis and recognition based on emotion words using ren-cecps. *International Journal of Advanced Intelligence Paradigms*, 2(1):105–117.

Xiaojun Quan, Qifan Wang, Ying Zhang, Luo Si, and Liu Wenyin. 2015. Latent discriminative models for social emotion detection with emotional dependency. *ACM Transactions on Information Systems (TOIS)*, 34(1):2.

Yanghui Rao. 2016. Contextual sentiment topic model for adaptive social emotion classification. *IEEE Intelligent Systems*, 31(1):41–47.

Yanghui Rao, Qing Li, Wenyin Liu, Qingyuan Wu, and Quan Xiaojun. 2014a. Affective topic model for social emotion detection. *Neural Netw*, 58(5):29–37.

Yanghui Rao, Qing Li, Xudong Mao, and Wenyin Liu. 2014b. Sentiment topic models for social emotion mining. *Information Sciences*, 266(5):90–100.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1988. *Learning internal representations by error propagation*. MIT Press.

Fabrizio Sebastiani. 2001. Machine learning in automated text categorization. *Acm Computing Surveys*, 34(1):1–47.

Veselin Stoyanov and Claire Cardie. 2008. Annotating topics of opinions. In *International Conference on Language Resources and Evaluation, Lrec 2008, 26 May - 1 June 2008, Marrakech, Morocco*, pages 3213–3217.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 70–74. Association for Computational Linguistics.

Carlo Strapparava and Rada Mihalcea. 2008. Learning to identify emotions in text. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1556–1560. ACM.

Yichen Wang and Aditya Pal. 2015. Detecting emotions in social media: A constrained optimization approach. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 996–1002.

Min Ling Zhang. 2011. Lift: multi-label learning with label-specific features. In *International Joint Conference on Artificial Intelligence*, pages 1609–1614.

Min Ling Zhang and Zhi Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge Data Engineering*, 18(10):1338–1351.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

Deyu Zhou, Yang Yang, and Yulan He. 2018. Relevant emotion ranking from text constrained with emotion relationships. In *Meeting of the North American Chapter of the Association for Computation Linguistics*.

Deyu Zhou, Xuan Zhang, Yin Zhou, Quan Zhao, and Xin Geng. 2016. Emotion distribution learning from texts. In *Conference on Empirical Methods in Natural Language Processing*, pages 638–647.

Multi-grained Attention Network for Aspect-Level Sentiment Classification

Feifan Fan[1] **Yansong Feng**[12*] **Dongyan Zhao**[13]
[1]Institute of Computer Science and Technology, Peking University, China
[2]MOE Key Laboratory of Computational Linguistics, Peking University, China
[3]Beijing Institute of Big Data Research, China
{fanff, fengyansong, zhaody}@pku.edu.cn

Abstract

We propose a novel multi-grained attention network (MGAN) model for aspect level sentiment classification. Existing approaches mostly adopt coarse-grained attention mechanism, which may bring information loss if the aspect has multiple words or larger context. We propose a fine-grained attention mechanism, which can capture the word-level interaction between aspect and context. And then we leverage the fine-grained and coarse-grained attention mechanisms to compose the MGAN framework. Moreover, unlike previous works which train each aspect with its context separately, we design an aspect alignment loss to depict the aspect-level interactions among the aspects that have the same context. We evaluate the proposed approach on three datasets: laptop and restaurant are from SemEval 2014, and the last one is a twitter dataset. Experimental results show that the multi-grained attention network consistently outperforms the state-of-the-art methods on all three datasets. We also conduct experiments to evaluate the effectiveness of aspect alignment loss, which indicates the aspect-level interactions can bring extra useful information and further improve the performance.

1 Introduction

Aspect level sentiment classification is a fundamental task in sentiment analysis (Pang et al., 2008; Liu, 2012), which aims to infer the sentiment polarity (e.g. positive, neutral, negative) of sentence with respect to the aspects. For example, in sentence "*I like coming back to Mac OS but this laptop is lacking in speaker quality compared to my $400 old HP laptop*", the polarity of the sentence towards the aspect "*Mac OS*" is positive while the polarity is negative in terms of aspect "*speaker quality*".

Many statistical methods, such as support vector machine (SVM) (Wagner et al., 2014; Kiritchenko et al., 2014), are employed with well-designed handcrafted features. In recent years, neural network models (Socher et al., 2011; Dong et al., 2014; Nguyen and Shirai, 2015) are studied to automatically learn low-dimensional representations for aspects and their context. Attention mechanism (Wang et al., 2016; Li et al., 2017; Ma et al., 2017) is also be studied to characterize the effect of aspect on enforcing the model to pay more attention on the important words of the context. Previous works (Tang et al., 2016b; Chen et al., 2017) mainly employed the simple averaged aspect vector to learn the attention weights on the context words. Ma et al. [2017] further proposed the bidirectional attention mechanism, which interactively learns the attention weights on context/aspect words, with respect to the averaged vector of aspect/context, respectively.

These above attention methods are all at the coarse-grained level, which simply averages the aspect/context vector to guide learning the attention weights on the context/aspect words. The simple average pooling mechanism might cause information loss, especially for the aspect with multiple words or larger context. For example, in sentence "*I like coming back to Mac OS but this laptop is lacking in speaker quality compared to my $400 old HP laptop*", the simple averaged vector of long context might lose information when steering the attention weights on aspect words. Similarly, the simple averaged vector of aspect (i.e. *"speaker quality"*) may deviate from the intuitive core meaning (*i.e. "quality"*) when enforcing the model to pay varying attentions on the context words. From another perspective, previous works all regard the aspect and its context words as one instance, and train each instance separately. However, they do not consider the relationship among

---

[*] corresponding author.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3433–3442
Brussels, Belgium, October 31 - November 4, 2018.

the instances that have the same context words. The aspect-level interactions among the instances with same context might bring extra useful information. Considering the above example, intuitively, the aspect *"speaker quality"* should pay more attention on *"lacking"* and less attention on *"like"*, compared with aspect *"Mac OS"*, since they have different sentiment polarities.

In this paper, we propose a multi-grained attention network to address the above two issues in aspect level sentiment classification. Specifically, we propose a fine-grained attention mechanism (*i.e. F-Aspect2Context* and *F-Context2Aspect*), which is employed to characterize the word-level interactions between aspect and context words, and relieve the information loss occurred in coarse-grained attention mechanism. In addition, we utilize the bidirectional coarse-grained attention (*i.e. C-Aspect2Context* and *C-Context2Aspect*) and combine them with fine-grained attention vectors to compose the multi-grained attention network for the final sentiment polarity prediction, which can leverage the advantages of them. More importantly, in order to make use of the valuable aspect-level interaction information, we design an aspect alignment loss in the objective function to enhance the difference of the attention weights towards the aspects which have the same context and different sentiment polarities. As far as we know, we are the first to explore the interactions among the aspects with the same context.

To evaluate the proposed approach, we conduct experiments on three datasets: laptop and restaurant are from the SemEval 2014 Task 4 and the third one is a tweet collection. Experimental results show that our method achieves the best performance on all three datasets.

2 Related Work

Aspect-level sentiment analysis is a branch of sentiment classification, which requires considering both the sentence and aspect information.

Traditional approaches (Jiang et al., 2011; Kiritchenko et al., 2014; Vo and Zhang, 2015) regard this task as the text classification problem and design effective features, which are utilized in statistical learning algorithms for training a classifier. Kiritchenko et al. [2014] proposed to use SVM based on n-gram features, parse features and lexicon features, which achieved the best performance in SemEval 2014. Vo and Zhang [2015] designed sentiment-specific word embedding and sentiment lexicons as rich features for prediction. These methods highly depend on the effectiveness of the laborious feature engineering work and easily reach the performance bottleneck.

In recent works, there are growing studies on neural network based methods due to their capability of encoding original features as continuous and low-dimensional vectors without feature engineering. Recursive Neural Network (Socher et al., 2011; Dong et al., 2014; Nguyen and Shirai, 2015) are studied to conduct semantic compositions on tree structures, and generate representations for prediction. Methods on LSTM (Hochreiter and Schmidhuber, 1997) were proposed to model the context information and use an aggregated vector for prediction. TD-LSTM (Tang et al., 2016a) adopted LSTM to model the left context and right context of the aspect, and concatenate them as the representation for prediction. However, these works only focused on modeling the contexts without considering the aspects, which performed an important role in estimate the sentiment polarity.

Attention mechanisms (Wang et al., 2016; Lei et al., 2016; Li et al., 2017) are studied to enhance the influence of aspects on the final representation for prediction. Many approaches (Tang et al., 2016b; Chen et al., 2017) adopted the averaged aspect vector to learn the attention weights on the hidden vectors of context words. Ma et al. [2017] further proposed bidirectional attention mechanism, which also learns the attention weights on aspect words towards the averaged vector of context words. These attention methods only consider the coarse-grained level attention, through using the simple averaged aspect/context vector to steer the attention weights learning on the context/aspect words, which might cause some information loss on the long aspect or context case.

In contrast, motivated by the bidirectional attention flow approaches (Seo et al., 2017; Pan et al., 2017) in machine comprehension, we propose a fine-grained attention mechanism which is responsible for linking and fusing information from the aspect and the context words. Furthermore, we leverage both the coarse-grained and fine-grained attentions to compose the multi-grained attention network (MGAN). In addition, existing works train each instance separately. However, we observe that the interactions among the aspects, which have the same context words, could bring extra useful information. Thus we design the aspect alignment loss in the objective function to depict such kind of relationship, which is the first work to explore the aspect-level interactions.

3 Our Approach

3.1 Task Definition

Given a sentence $s = \{w_1, w_2, \cdots, w_N\}$ consisting of N words, and an aspect list $A = \{a_1, \cdots, a_k\}$, where the aspect list size is k and each aspect $a_i = \{w_{i_1}, \cdots, w_{i_M}\}$ is a subsequence of sentence s, which contains $M \in [1, N)$ words. Aspect-level sentiment classification evaluates sentiment polarity of the sentence s with respect to each aspect a_i.

We present the overall architecture of the proposed Multi-grained Attention Network (MGAN) model in Figure 1. It consists of the Input Embedding layer, the Contextual Layer, the Multi-grained Attention Layer and the Output Layer.

3.2 Input Embedding Layer

Input Embedding Layer maps each word to a high dimensional vector space. We employ the pre-trained word vector, GloVe (Pennington et al., 2014), to obtain the fixed word embedding of each word. Specifically, we denote the embedding lookup matrix as $\mathbb{L} \in \mathbb{R}^{d_v \times |V|}$, where d_v is the word vector dimension and $|V|$ is the vocabulary size.

3.3 Contextual Layer

We employ a bidirectional Long Short-Term Memory Network (BiLSTM) on top of the embedding layer to capture the temporal interactions among words. Specifically, at time step t, given the input word embedding x, the update process of forward LSTM network can be formalized as follows:

$$i_t = \sigma(\overrightarrow{W}_i \cdot [\overrightarrow{h}_{t-1}, \overrightarrow{x}_t] + \overrightarrow{b}_i) \tag{1}$$

$$f_t = \sigma(\overrightarrow{W}_f \cdot [\overrightarrow{h}_{t-1}, \overrightarrow{x}_t] + \overrightarrow{b}_f) \tag{2}$$

$$o_t = \sigma(\overrightarrow{W}_o \cdot [\overrightarrow{h}_{t-1}, \overrightarrow{x}_t] + \overrightarrow{b}_o) \tag{3}$$

$$g_t = tanh(\overrightarrow{W}_g \cdot [\overrightarrow{h}_{t-1}, \overrightarrow{x}_t] + \overrightarrow{b}_g) \tag{4}$$

$$\overrightarrow{c}_t = f_t * \overrightarrow{c}_{t-1} + i_t * g_t \tag{5}$$

$$\overrightarrow{h}_t = o_t * tanh(\overrightarrow{c}_t) \tag{6}$$

Where σ is the sigmoid activation function, i_t, f_t, o_t are the input gate, forget gate and output gate, respectively. $\overrightarrow{W}_i, \overrightarrow{W}_f, \overrightarrow{W}_o, \overrightarrow{W}_g \in \mathbb{R}^{d*(d+d_v)}$, $\overrightarrow{b}_i$, $\overrightarrow{b}_f$, $\overrightarrow{b}_o$, $\overrightarrow{b}_g \in \mathbb{R}^d$, and d is the hidden dimension size. The backward LSTM does the similar process and we can get the concatenated output $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t] \in \mathbb{R}^{2d}$. Given the word embeddings of a context sentence s and a corresponding aspect a_j, we will employ the BiLSTM separately and get the sentence contextual output $H \in \mathbb{R}^{2d*N}$ and aspect contextual output $Q \in \mathbb{R}^{2d*M}$.

In addition, considering that the context words with closer distance to an aspect may have higher influence to the aspect, we utilize the position encoding mechanism to simulate the observation. Formally, the weight for a context word w_j, which has l word-level distance from the aspect (here we treat the aspect phrase as a single unit), is defined as follows:

$$w_t = 1 - \frac{l}{N - M + 1} \tag{7}$$

Specifically, we treat the weights of words within the aspect as 0 in order to focus on the context words in the sentence. Then we can obtain the final contextual outputs of context words $H = [H_1 * w_1, \cdots, H_N * w_N]$.

3.4 Multi-grained Attention Layer

Attention mechanism is a common way to capture the interactions between the aspect and context words. Previous methods (Tang et al., 2016b; Ma et al., 2017; Chen et al., 2017) only adopt coarse-grained attentions, which simply use the averaged aspect/context vector as the guide to learn the attention weights on context/aspect. However, the simple average pooling in generating the guide vector might bring some information loss, especially for the aspect with multiple words or larger context. We propose the fine-grained attention mechanism, which is responsible for linking and fusing information from the aspect and context words. This mechanism is designed to capture the word-level interactions which estimate how each aspect/context word affect each context/aspect word. In addition, we concatenate both the fine-grained and coarse-grained attention vectors to obtain the final representation. From other perspective, we observe the relationship among aspects can introduce extra valuable information. Hence, we propose an aspect alignment loss, which is designed to strengthen the at-

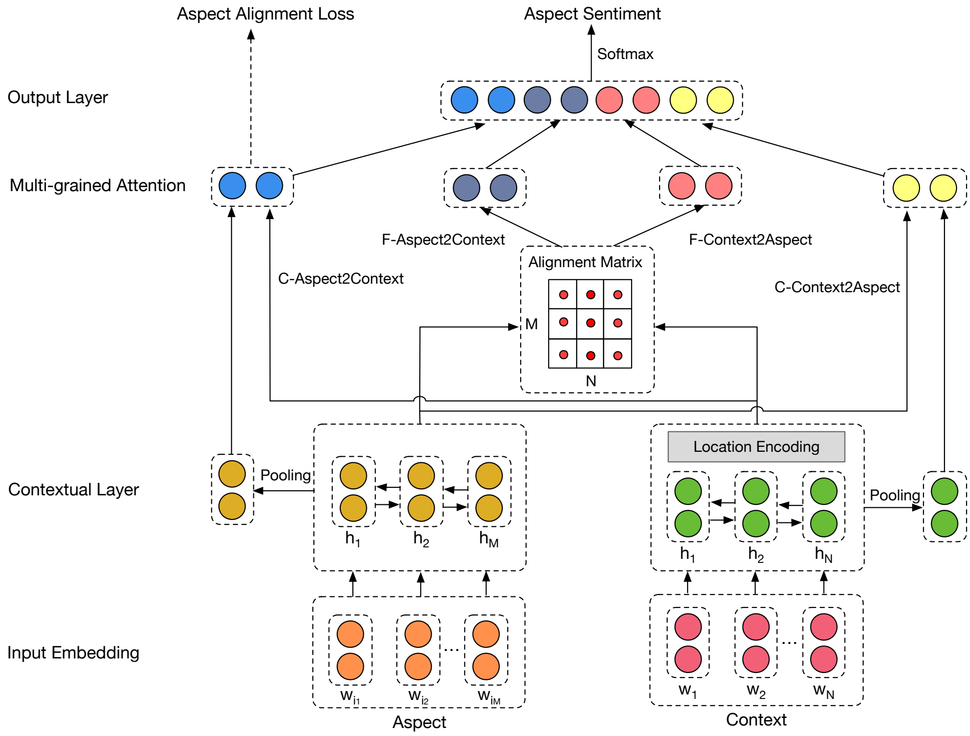

Figure 1: The architecture of the proposed multi-grained attention network.

tention difference among aspects with same context and different sentiment polarities.

Coarse-grained Attention

Coarse-grained attention is a widely used mechanism to capture the interactions between aspect and context, which utilizes an averaged aspect vector to steer the attention weights on the context words. Follow the work in (Ma et al., 2017), we employ the bidirectional attention mechanism, namely *C-Aspect2Context* and *C-Context2Aspect*.

(1) C-Aspect2Context learns to assign attention scores to the context words with respect to the averaged aspect vector. Here we employ an average pooling layer above aspect contextual output Q to generate the averaged aspect vector $Q_{avg} \in \mathbb{R}^{2d}$. For each word vector H_i in context, we can compute the attention score a_i^{ca} as follows:

$$s_{ca}(Q_{avg}, H_i) = Q_{avg} * W_{ca} * H_i \tag{8}$$

$$a_i^{ca} = \frac{exp(s_{ca}(Q_{avg}, H_i))}{\sum_{k=1}^{N} exp(s_{ca}(Q_{avg}, H_k))} \tag{9}$$

Where the score function s_{ca} computes the weight which indicates the importance of a context word towards aspect sentiment. $W_{ca} \in \mathbb{R}^{2d*2d}$ is the attention weight matrix. Then the weighted combination of the context output $m^{ca} \in \mathbb{R}^{2d}$ is calculated as follows:

$$m^{ca} = \sum_{i=1}^{N} a_i^{ca} \cdot H_i \tag{10}$$

(2) C-Context2Aspect learns to assign attention weights on aspects words, which follows the similar learning process with *C-Aspect2Context*. We utilize the average pooling mechanism to obtain the averaged context vector H_{avg}, and compute the weights for each word w_i in the aspect phrase. We compute the final weighted combination of aspect vector $m^{cc} \in \mathbb{R}^{2d}$ as follows:

$$s_{cc}(H_{avg}, Q_i) = H_{avg} * W_{cc} * Q_i \tag{11}$$

$$a_i^{cc} = \frac{exp(s_{cc}(H_{avg}, Q_i))}{\sum_{k=1}^{M} exp(s_{cc}(H_{avg}, Q_k))} \tag{12}$$

$$m^{cc} = \sum_{i=1}^{M} a_i^{cc} \cdot Q_i \tag{13}$$

where $W^{cc} \in \mathbb{R}^{2d*2d}$ is the attention weight matrix.

Fine-grained Attention

As introduced above, we propose a fine-grained attention mechanism to characterize the word-level interactions and evaluate how each aspect/context word affect each context/aspect word. Considering the previous example "*I like coming back to Mac OS but this laptop is lacking in speaker quality compared to my \$400 old HP laptop*", the word *"quality"* in aspect *"speaker quality"* should have more effect on the context words compared with word *"speaker"*. Accordingly, the context words should pay more attention on *"quality"* instead of *"speaker"*.

Formally, we define an alignment matrix $U \in \mathbb{R}^{N*M}$, between the contextual output of and the context H and the aspect Q, where U_{ij} indicates the similarity between i-th context word and j-th aspect word. The similarity matrix U is computed by

$$U_{ij} = W_u([H_i; Q_j; H_i * Q_j]) \tag{14}$$

Where $W_u \in \mathbb{R}^{1*6d}$ is the weight matrix, $[;]$ is the vector concatenation across row, $*$ is the elementwise multiplication. Then we use U to calculate the attention vectors in both directions.

(1) F-Aspect2Context estimates which context word has the closest similarity to one of the aspect word and are hence critical for determining the sentiment. We can compute the attention weights a^{fa} on context words by

$$s_i^{fa} = max(U_{i,:}) \tag{15}$$

$$a_i^{fa} = \frac{exp(s_i^{fa})}{\sum_{k=1}^{N} exp(s_k^{fa})} \tag{16}$$

where s_i^{fa} obtains the maximum similarity across column. And then we can get the attended vector $m^{fa} \in \mathbb{R}^{2d}$ as follows:

$$m^{fa} = \sum_{i=1}^{N} a_i^{fa} \cdot H_i \tag{17}$$

(2) F-Context2Aspect measures which aspect words are most relevant to each context word. Let $a_i^{fc} \in \mathbb{R}^M$ be the attention weights on aspect contextual output Q with respect to the i-th context word vector H_i. The attended aspect vector $q^{fc} \in \mathbb{R}^{2d*N}$ is defined as follows:

$$a_{ij}^{fc} = \frac{exp(U_{ij})}{\sum_{k=1}^{M} exp(U_{ik})} \tag{18}$$

$$q_i^{fc} = \sum_{j=1}^{M} a_{ij}^{fc} \cdot Q_j \tag{19}$$

Then we use an average pooling layer on q^{fc} to get the attended vector $m^{fc} \in \mathbb{R}^{2d}$:

$$m^{fc} = Pooling([q_1^{fc}, \cdots, q_N^{fc}]) \tag{20}$$

3.5 Output Layer

At last, we concatenate both the coarse-grained and fine-grained attention vectors as the final representation $m \in \mathbb{R}^{8d}$, which will be fed to a softmax layer for determining the aspect sentiment polarity.

$$m = [m^{ca}; m^{cc}; m^{fa}; m^{fc}] \tag{21}$$

$$p = softmax(W_p * m + b_p) \tag{22}$$

where $p \in \mathbb{R}^C$ is the probability distribution for the polarity of aspect sentiment, $W_p \in \mathbb{R}^{C*8d}$ and $b_p \in \mathbb{R}^C$ are the weight matrix and bias, respectively. Here we set $C = 3$, which is the number of aspect sentiment classes.

3.6 Model Training

Aspect Alignment Loss

Existing approaches train each aspect with its context separately, without considering the relationship among the aspects. However, we observe the aspect-level interactions can bring extra valuable information. In order to enhance the attention differences of aspects, which have the same context and different sentiment polarities, we design the aspect alignment loss on the C-Aspect2Context attention weights. C-Aspect2Context is employed to find the important context words in terms of a specific aspect. With the constraint of aspect alignment loss, each aspect will pay more attention on the important words through the comparisons with other related aspects. In terms of the previous example, the aspect *"speaker quality"* should pay more attention on *"lacking"* and less attention on *"like"*, compared with aspect *"Mac OS"* due to their different sentiment polarities.

Specifically, for each aspect pair a_i and a_j in aspect list A, we compute the square loss on the coarse-grained attention vector a_i^{ca} and a_j^{ca}, and also estimate the distance $d_{ij} \in [0, 1]$ between a_i and a_j as the loss weight.

$$d_{ij} = \sigma(W_d([Q_i; Q_j; Q_i * Q_j]) \tag{23}$$

$$\mathcal{L}_{align} = -\sum_{i=1}^{M-1} \sum_{j=i+1, y_i \neq y_j}^{M} \sum_{k=1}^{N} d_{ij} \cdot (a_{ik}^{ca} - a_{jk}^{ca})^2 \tag{24}$$

Where σ is the sigmoid function, $W_d \in \mathbb{R}^{1*6d}$ is weight matrix for computing the distance, y_i and y_j are the true labels of the aspect a_i and a_j, a_{ik}^{ca} and a_{jk}^{ca} are the attention weights on k-th context word towards aspect a_i and a_j, respectively.

For training the multi-grained attention network (MGAN), we should optimize all the parameters Θ from the LSTM networks: $[W_i, W_o, W_f, W_g, b_i, b_o, b_f, b_g]$, the attention and alignment loss parameters: $[W_{ca}, W_{cc}, W_u, W_d]$ and softmax parameters: $[W_p, b_p]$. The final loss function is consisting of the cross-entropy loss, aspect alignment loss and regularization item as follows:

$$\mathcal{L} = -\sum_{i=1}^{C} y_i log(p_i) + \beta \mathcal{L}_{align} + \lambda \|\Theta\|^2 \tag{25}$$

Where $\beta \geq 0$ and $\lambda \geq 0$ controls the influence of the aspect alignment loss and the L_2 regularization item, respectively. We employ the stochastic gradient descent (SGD) optimizer to compute and update the training parameters. In addition, we utilize dropout strategy to avoid overfitting.

4 Experiments

In this section, we conduct experiments to evaluate our two hypotheses: (1) whether the word-level interaction between aspect and context can help relieve the information loss and improve the performance. (2) whether the relationship among the aspects, which have the same context and different sentiment polarities, can bring extra useful information.

4.1 Experiment Setting

We conduct experiments on three datasets, as shown in Table 1. The first two are from the SemEval 2014 Task 4[1] (Pontiki et al., 2014), which contains the reviews in laptop and restaurants, respectively. The third one is a tweet collection, which are gathered by (Dong et al., 2014). Each aspect with the context is labeled by three sentiment polarities, namely *Positive*, *Neutral* and *Negative*. In addition, we adopt *Accuracy* and *Macro-F1* as the metrics to evaluate the performance of aspect-level sentiment classification, which is widely used in previous works (Tang et al., 2016b; Ma et al., 2017; Chen et al., 2017; Wang et al., 2016).

In our experiments, word embeddings for both context and aspect words are initialized by Glove (Pennington et al., 2014). The dimension of word embedding d_v and hidden state d are set to 300. The weight matrix and bias are initialized by sampling from a uniform distribution $U(0.01, 0.01)$. The coefficient λ of L_2 regularization item is 10^{-5}, the parameter β of aspect alignment loss and drop out rate are set to 0.5.

Dataset	Positive		Neutral		Negative	
	Train	Test	Train	Test	Train	Test
Laptop	994	341	870	128	464	169
Restaurant	2164	728	807	196	637	196
Twitter	1561	173	3127	346	1560	173

Table 1: The statistics of the datasets.

4.2 Compared Methods

To evaluate the performance of proposed approach, we compared with the following methods:

Majority is the basic baseline, which chooses the largest sentiment polarity in the training set to each instance in the test set.

Feature+SVM (Kiritchenko et al., 2014) uses n-gram features, parse features and lexicon features based on SVM, which achieves the state-of-the-art performance in SemEval 2014.

LSTM (Wang et al., 2016) utilizes one LSTM network to learn the hidden states and obtain the averaged vector to predict the sentiment polarity.

ATAE-LSTM (Wang et al., 2016) learns attention embeddings and combine them with the LSTM hidden states to predict the polarity.

TD-LSTM (Tang et al., 2016a) employs two directional LSTM networks, which estimate the left context and right context of the target aspect, respectively. Finally it takes the last hidden states of LSTM networks for prediction.

MemNet (Tang et al., 2016b) applys multi-hop attentions on the word embeddings, learns the attention weights on context word vectors with respect to the averaged query vector.

IAN (Ma et al., 2017) interactively learns the coarse-grained attentions between the context and aspect, and concatenate the vectors for prediction.

BILSTM-ATT-G (Liu and Zhang, 2017) models left and right context with two attention-based LSTMs and utilizes gates to control the importance of left context, right context and the entire sentence for prediction.

RAM(Chen et al., 2017) learns multi-hop attentions on the hidden states of bidirectional LSTM networks for context words, and proposes to use GRU network to get the aggregated vector from the attentions. Similar with **MemNet**, the atten-

---

[1] The detailed task introduction can be found in http://alt.qcri.org/semeval2014/task4/.

tion weights on context words are steered by the simple averaged aspect vector.

We also list the variants of MGAN model, which are used to analyze the effects of coarse-grained attention, fine-grained attention and aspect alignment loss, respectively.

MGAN-C only employs the coarse-grained attentions for prediction, which is similar with **IAN**.

MGAN-F only utilizes the proposed fine-grained attentions for prediction.

MGAN-CF adopts both the coarse-grained and fine-grained attentions, while without applying the aspect alignment loss.

MGAN is the complete multi-grained attention network model.

4.3 Overall Performance Comparison

Table 2 shows the performance comparison results of MGAN with other baseline methods. We can have the following observations.

(1) **Majority** performs worst since it only utilizes the data distribution information. **Feature+SVM** can achieve much better performance on all the datasets, with the well-designed feature engineering. Our method **MGAN** outperforms **Majority** and **Feature+SVM** since **MGAN** could learn the high quality representation for prediction.

(2) **ATAE-LSTM** is better than **LSTM** since it employs attention mechanism on the hidden states and combines with attention embedding to generate the final representation. **TD-LSTM** performs slightly better than **ATAE-LSTM**, and it employs two LSTM networks to capture the left and right context of the aspect. **TD-LSTM** performs worse than our method **MGAN** since it could not properly pay more attentions on the important parts of the context.

(3) **IAN** achieves slightly better results with the previous LSTM-based methods, which interactively learns the attended aspect and context vector as final representation. Our method consistently performs better than **IAN** since we utilize the fine-grained attention vectors to relieve the information loss in **IAN**. **MemNet** continuously learns the attended vector on the context word embedding memory, and updates the query vector at each hop. **BILSTM-ATT-G** models left context and right context using attention-based LSTMs, which achieves better performance than **MemNet**. **RAM** performs better than other baselines. It employs bidirectional LSTM network to generate contextual memory, and learns the multiple attended vector on the memory. Similar with **MemNet**, it utilizes the averaged aspect vector to learn the attention weights on context words.

Our proposed **MGAN** consistently performs better than **MemNet**, **BILSTM-ATT-G** and **RAM** on all three datasets. On one hand, they only consider to learn the attention weights on context towards the aspect, and do not consider to learn the weights on aspect words towards the context. On the other hand, they just use the averaged aspect vector to guide the attention, which will lose some information, especially on the aspects with multiple words. From another perspective, our method employs the aspect alignment loss, which can bring extra useful information from the aspect-level interactions.

4.4 Analysis of MGAN model

Table 3 shows the performance comparison among the variants of MGAN model. We can have the following observations.

(1) the proposed fine-grained attention mechanism **MGAN-F**, which is responsible for linking and fusing the information between the context and aspect word, achieves competitive performance compared with **MGAN-C**, especially on laptop dataset. To investigate this case, we collect the percentage of aspects with different word lengths in Table 4. We can find that laptop dataset has the highest percentage on the aspects with more than two words, and the second-highest percentage on two words. It demonstrates **MGAN-F** has better performance on aspects with more words, and make use of the word-level interactions to relieve the information loss occurred in coarse-grained attention mechanism.

(2) **MGAN-CF** is better than both **MGAN-C** and **MGAN-F**, which demonstrates the coarse-grained attentions and fine-grained attentions could improve the performance from different perspectives. Compared with **MGAN-CF**, the complete **MGAN** model gains further improvement by bringing the aspect alignment loss, which is designed to capture the aspect level interactions. Specifically, we collect the statistics of sentence-level with different aspect amounts, which is shown in Table 5. We can observe that both laptop and restaurant datasets have relatively high percentage on the sentences with multiple aspects.

Method	Laptop		Restaurant		Twitter	
	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
Majority	0.5350	0.3333	0.6500	0.3333	0.5000	0.3333
Feature-SVM	0.7049	-	0.8016	-	0.6340	0.6330
ATAE-LSTM	0.6870	-	0.7720	-	-	-
TD-LSTM	0.7183	0.6843	0.7800	0.6673	0.6662	0.6401
IAN	0.7210	-	0.7860	-	-	-
MemNet	0.7237	-	0.8032	-	0.6850	0.6691
BILSTM-ATT-G	0.7312	0.6980	0.7973	0.6925	0.7038	0.6837
RAM	0.7449	0.7135	0.8023	0.7080	0.6936	0.6730
MGAN	**0.7539**	**0.7247**	**0.8125**	**0.7194**	**0.7254**	**0.7081**

Table 2: The performance comparisons of different methods on the three datasets, where the results of baseline methods are retrieved from published papers. The best performances are marked in bold.

Method	Laptop		Restaurant		Twitter	
	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
MGAN-C	0.7273	0.6933	0.8054	0.7099	0.7153	0.6952
MGAN-F	0.7398	0.7082	0.8000	0.7092	0.7110	0.6918
MGAN-CF	0.7445	0.7121	0.8089	0.7135	0.7254*	0.7081*
MGAN	**0.7539**	**0.7247**	**0.8125**	**0.7194**	**0.7254**	**0.7081**

Table 3: The performance comparisons of MGAN variants. $*$ means MGAN-CF and MGAN can be regarded as the same method on twitter dataset.

Dataset	#words=1	#words=2	#words>2
Laptop	61.60%	29.16%	9.24%
Restaurant	74.47%	17.32%	8.21%
Twitter	29.99%	69.91%	0.10%

Table 4: The percentage of aspects with different word length on three datasets. Here we give the overall statistic of each dataset.

The improved performance on the two datasets shows the importance of capturing the aspect-level interactions. In terms of twitter dataset, almost all of the sentences only has one aspect. In this case, the method **MGAN** can be regarded as **MGAN-CF**.

Dataset	#aspects=1	#aspects=2	#aspects>2
Laptop	63.94%	23.32%	12.74%
Restaurant	50.89%	28.60%	20.51%
Twitter	99.91%	0.09%	0.00%

Table 5: The percentage of sentences with different aspect numbers on three datasets. Aspects with the same context are regarded as the same sentence.

4.5 Case Study

In order to demonstrate the effect of aspect alignment loss, we visualize the attention weights of the C-Aspect2Context mechanism. Figure 2 shows the attention weights of two aspects “resolution” and “fonts”, whose sentiment polarities are positive and negative, respectively. From the above two bars, we can observe that the C-Aspect2Context can enforce the model to pay more attentions on the important words with respect to the aspect. For example, in terms of the aspect“resolution”, the words “has”, “higher” and “but” have higher attention weights compared with other words. In contrast, aspect “fonts” pays more attentions on words “but”, “fonts” and “small”. In addition, we evaluate the effect of aspect alignment loss, which enhances the attention difference between the aspect “resolution” and “fonts”. For the two bars at bottom, we can find that aspect “fonts” has more attention on “small” and less attention on “higher”, compared with the aspect “resolution”. This phenomenon shows that with the constraint of aspect alignment loss, C-Aspect2Context can not only learn the important context words for each aspect, but also can make the attention gaps on the important words be as large as possible for aspects with different polarities.

5 Conclusion

In this paper, we propose a multi-grained attention network (MGAN) for aspect-level sentiment classification. Specifically, we propose a fine-grained attention mechanism, which is responsible for linking and fusing the words from the aspect and context, to capture the word-level interaction. And we combine it with the coarse-grained atten-

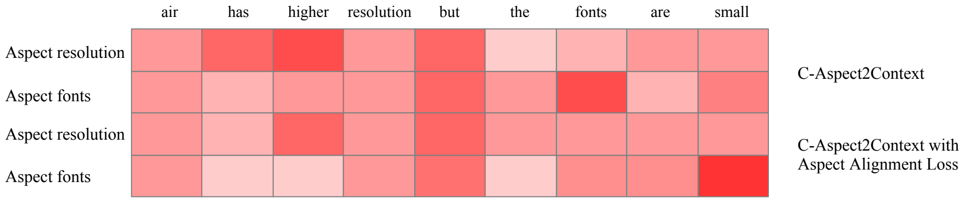

Figure 2: The attention visualizations on aspect "resolution" and "fonts". The above two bars are from the C-Aspect2Context attention mechanism, and the two bars at bottom are from the C-Aspect2Context attention mechanism with the constraint of aspect alignment loss.

tion mechanism to compose the MGAN model. In addition, we design an aspect alignment loss to characterize the aspect-level interactions among aspects, which have the same context and different sentiment polarities, to explore extra valuable information. Experimental results demonstrate the effectiveness of our approach on three public datasets.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61672057, 61672058); KLSTSPI Key Lab. of Intelligent Press Media Technology. For any correspondence, please contact Yansong Feng.

References

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 49–54.

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

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117.

Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. 2017. Deep memory networks for attitude identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 671–680. ACM.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 572–577.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *IJCAI*.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2509–2514.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098*.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. *Proceedings of the*

8th International Workshop on Semantic Evaluation (SemEval 2014), pages 27–35.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. pages 214–224.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI*, pages 1347–1353.

Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. Dcu: Aspect-based polarity classification for semeval task 4. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 223–229.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615.

Attentive Gated Lexicon Reader with Contrastive Contextual Co-Attention for Sentiment Classification

Yi Tay$^{\dagger *}$, **Luu Anh Tuan**$^{\psi *}$, **Siu Cheung Hui**$^{\phi}$, **Jian Su**$^{\delta}$
†`ytay017@e.ntu.edu.sg`
$^{\psi}$`at.luu@i2r.a-star.edu.sg`
$^{\phi}$`asschui@ntu.edu.sg`
$^{\delta}$`sujian@i2r.a-star.edu.sg`
$^{\dagger,\phi}$School of Computer Science and Engineering, Nanyang Technological University
$^{\psi,\delta}$A*Star, Institute for Infocomm Research, Singapore

Abstract

This paper proposes a new neural architecture that exploits readily available sentiment lexicon resources. The key idea is that that incorporating a word-level prior can aid in the representation learning process, eventually improving model performance. To this end, our model employs two distinctly unique components, i.e., (1) we introduce a lexicon-driven contextual attention mechanism to imbue lexicon words with long-range contextual information and (2), we introduce a contrastive co-attention mechanism that models contrasting polarities between all positive and negative words in a sentence. Via extensive experiments, we show that our approach outperforms many other neural baselines on sentiment classification tasks on multiple benchmark datasets.

1 Introduction

Across the rich history of sentiment analysis research (Kim and Hovy, 2004; Liu, 2012; Pang et al., 2008), sentiment lexicons have been extensively used as features for sentiment classification tasks. Lexicons, either handcrafted or algorithmically generated, consist of words and their associated polarity score. For instance, lexicons assign a high positive score for the word '*excellent*' but a negative score for the word '*terrible*'. Traditionally, the summation of lexicon scores has been treated as a reasonable heuristic estimate (or feature) that is capable of supporting opinion mining applications. Throughout the years, plenty of lexicon lists have been built for various specific domains or general purposes (Hu and Liu, 2004; Mohammad et al., 2013; Wilson et al., 2005). They are indeed valuable resources that should be exploited.

However, sentiment lexicons are in reality hardly useful without context. After all, the complexity and ambiguity of natural language pose great challenges for the crude bag-of-words generalization of lexicons. Firstly, the concept of semantic compositionality is non-existent in simple lexicon approaches which raises problems when handling flipping negation (*not happy*), content word negation (*ameliorates pain*) or unbounded dependencies (*no body passed the exam*). Secondly, lexicons also do not handle word sense, e.g., not being able to differentiate the meaning of *hot* in the phrases *'a hot, attractive person'* and a *'a scorching hot day'*. Thirdly, simple summation over lexicon scores cannot deal with sentences with double contrasting polarities, e.g., the lexicon polarity score of *'Thanks for making this uncomfortable situation more comfortable'* becomes negative because *uncomfortable* has a higher negative lexicon score over the positive score of the word *comfortable*. Lastly, strongly positive or negative words may occur in neutral context which forces an inclination of predictions towards a non-neutral polarity. As such, the exploitation of readily available lexicon lists is an inherently challenging task.

Deep learning has demonstrated incredibly competitive performance in many NLP tasks (Liu et al., 2015; Bradbury et al., 2016; Tai et al., 2015). With no exception, the task of sentiment analysis is recently also dominated by neural architectures. It has been proven from the fact that the top systems from SemEval Sentiment analysis challenges (e.g., notably 2016 and 2017) have mainly leveraged the effectiveness of deep learning models. The main advantage of deep learning approach is that it is effective in exploring both linguistic and semantic relations between words, thus can overcomes the problems of lexicon-based approach. However, current deep learning approach

---

[*] Denotes equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3443–3453
Brussels, Belgium, October 31 - November 4, 2018.

for sentiment analysis usually faces with the major shortcoming, i.e., being limited by the quantity of high quality labeled data. Manual labeling of data, however, is costly and require domain expert knowledge which is not always available in practice.

Given the pros and cons of previous two previous approaches, we aim to combine the best of both worlds - the traditional sentiment lexicon and modern deep learning architectures. To the best of our knowledge, the only work that combines the two paradigms within *end-to-end* neural networks is the Lexicon RNN model (Teng et al., 2016). In their approach, sentiment lexicons are extracted from the hidden states of a recurrent neural network and passed through a simple feed-forward neural network to produce a new polarity weight. This approach, however, has some limitations which will be illustrated using the following example:

"Thanks for making this horrible situation at work more bearable."

Firstly, the Lexicon RNN does not consider the interactions between positive or negative lexicon words, which makes it susceptible to misleading strong lexicon priors. In the above example, the word '*horrible*' is a strongly negative word in most lexicons. As a result, the Lexicon RNN (and many other lexicon based approaches in general) will assign a negative polarity to the sentence. Clearly, modeling similarity between two contrasting polarity words ('*horrible*' and '*bearable*') can help the model resolve this confusion. Secondly, the RNN encoder in the Lexicon RNN is restricted by the sequential nature of the recurrent model, resulting in a limited global view of the entire sentence. For example, the word pairs ('*horrible*', '*bearable*') and ('*thanks*', '*bearable*') are useful for detecting the polarity of the sentence but do not have any explicit interaction even with a sequential RNN encoder. Moreover, the word pair ('*thanks*', '*bearable*') is very far apart in the above example sentence, making it challenging for RNN encoders to capture interactions between them. Finally, the Lexicon RNN faces difficulty dealing with more than two classes due to its design, i.e., linear combination of two scalar scores. In order to cope with this weakness, the authors define hardcoded *dataset specific* thresholds for 5-way classification. Adapting this to 3-way (positive, negative and neutral) is cumbersome as thresholds have to be found by either maximizing over the development set or defined heuristically.

In this paper, we introduce a new end to end paradigm that integrates lexicon information into neural network for the task of sentiment analysis. More specifically, instead of learning a lexicon-based score, we propose to learn an auxiliary embedding by exploiting lexicon information. The key motivation behind the auxiliary representation is that compositional learning with prior/global knowledge of positive and negative inclined words can lead to improved representations. Next, a gating mechanism controls the additive blend between this lexicon-based representation and a standard attention-based recurrent model. In essence, this supporting network aims to learn a *'lexicon-based'* view of the sentence and can be interpreted as *'learning to compose'* by exploiting lexicon information. Finally, instead of the combination of two scalar values (the base lexicon score and sentence bias score) as in the Lexicon RNN model, we propose to use the k-class softmax function at the final layer. Intuitively, it is a more natural solution for fine-grained sentiment classification over the cumbersome tuning of ad-hoc threshold values. Our contributions can be summarized as follows:

- We propose to learn an auxiliary embedding by exploiting lexicon information rather than learning a lexicon-based score. Its design is a more natural and flexible solution for k-class sentiment classification.
- We propose a *contextual attention* (CA) mechanism that learns to attend to lexicon words based on the context. Unlike Lexicon RNN which extracts the hidden representations from the recurrent model, contextual attention allows a wider, global and more complete view of the context (sentence) by matching against every single word in the sentence. In addition to semantic *compositionality*, our model also benefits from semantic *similarity*.
- We propose to model the interaction between the positive and negative lexicon words inside the neural network. Positive and negative lexicon words are modeled seperately and subsequently compared using *contrasive co-attention* (CC) which learns the relative importance of positive lexicons with respect

to negative lexicons (and vice versa). Modeling such intricacies between positive and negative words allows our model to deal with scenarios such as contrasting polarities, neutrality and also sarcasm. We also discover that our CC mechanism produces a neutralizing effect which negates misleading attention on words with intense polarity even though the context is neutral.

Overall, we propose AGLR (*Attentive Gated Lexicon Reader*), a new attention-based neural architecture that exploits sentiment lexicons for learning to compose an auxiliary sentence embedding. Our model achieves state-of-the-art performance on several benchmark datasets. Finally, our AGLR, a *single neural model*, also achieves competitive performance with respect to top teams in SemEval runs which are mostly comprised of extensively engineered ensembles.

2 Related Work

Sentiment lexicons have a rich traditional in sentiment analysis research and have been exploited in many statistical methods across the years (Hu and Liu, 2004; Kim and Hovy, 2004; Agarwal et al., 2011; Mohammad et al., 2013; Tang et al., 2014b,a; Teng et al., 2016). It is easy to see how sentiment lexicons are able to benefit opinion mining applications. More specifically, sentiment lexicons form an integral role in the winning solutions of SemEval 2013 (Mohammad et al., 2013) and 2014 (Miura et al., 2014). In many of these these approaches, standard machine learning classifiers (such as Support Vector Machines) are trained on discrete features partly derived from resources such as sentiment lexicon.

In recent years, we see a shift of the state-of-the-art from discrete models to neural models (Socher et al., 2013; Kim, 2014; Dong et al., 2014; Tang et al., 2016; Tai et al., 2015; Ren et al., 2016; Zhang et al., 2016; Teng and Zhang, 2016). This ranges from learning sentiment-specific word embeddings (Tang et al., 2014b; Faruqui et al., 2015) to end-to-end neural architectures (Teng et al., 2016; Angelidis and Lapata, 2017). The winning solution of SemEval 2016 (Deriu et al., 2016) utilized ensembles of convolutional neural networks (CNN). Recurrent-based models such as the bidirectional long short-term memory (BiLSTM) (Hochreiter and Schmidhuber, 1997; Graves et al., 2013) are popular and standard strong baselines for many opinion mining tasks including sentiment analysis (Tay et al., 2017) and sarcasm detection (Tay et al., 2018c). These neural models such as the BiLSTM are capable of modeling semantic compositionality and produce a feature vector which can be used for classification.

To integrate the information of lexicon inside Lexicon RNN model, Teng et al. (2016) proposed to use the hidden representations from a BiLSTM to influence the lexicon score, i.e., learning context-sensitive lexicon features. However, our method can be considered as a vastly different paradigm and instead learns a d-dimensional embedding using neural attention (Bahdanau et al., 2014; Luong et al., 2015) instead of a lexicon score. The key idea of neural attention is that it allows neural networks to look (or attend) to certain words in a sequence. This concept has indeed profoundly impacted the fields of NLP, giving rise to many variant architectures including end-to-end memory networks (Sukhbaatar et al., 2015; Li et al., 2017).

Our approach draws inspiration from memory networks and co-attentive models for machine comprehension (Xiong et al., 2016; Seo et al., 2016). In fact, the auxiliary network can be interpreted as a form of *multi-layered attention* which draws connection to vanilla memory networks. Attending over two sequences (or bidirectional attention) are intuitive approaches for NLP tasks such as information retrieval (Tay et al., 2018b) and generic text matching (Tay et al., 2018a). In our work, we adapt this to model the similarities between (1) lexicon-context and (2) contrasting polarities which borrows inspiration from (Riloff et al., 2013). Since our matching problem is derived from the same sequence (identified by a lexicon prior), this work can be interpreted as a form of self-attention (Vaswani et al., 2017) which draws relations to the intra-attentive model for sarcasm detection (Tay et al., 2018c).

3 Attentive Gated Lexicon Reader

In this section, we describe our proposed deep learning model for sentiment classification. The key idea of our model is to generate two representations, i.e., a lexicon-based auxiliary embedding of the sentence and also a generic compositional representation of the sentence. The former is generated via a supporting network that consists of contextual attention and contrastive co-attention layers. The latter is generated by a vanilla attention-based BiLSTM model. A gating mechanism then combines them for prediction.

3.1 Embedding Layer

Firstly, we extract all lexicon words from the input sequence and then separately[1] denote them as positive or negative words. Overall, our model accepts three sequences as an input. (1) the original sentence, (2) a list of positive lexicon words found in the sentence and (3) a list of negative lexicon words found in the sentence. The three sequences are indexed into a word embedding layer $W \in \mathbb{R}^{|V| \times d}$ which outputs three matrices $S \in \mathbb{R}^{d \times L_s}$ (sentence embeddings), $P \in \mathbb{R}^{d \times G_p}$ (positive lexicon embeddings) and $N \in \mathbb{R}^{d \times G_n}$ (negative lexicon embeddings). d is the dimensionality of the word embeddings and L_s, G_p and G_n are the maximum sequence lengths of sentence, positive lexicon and negative lexicon respectively.

3.2 Learning Sentence Representation

To learn sentence representations of the input sequence, we pass $S = (w_1, w_2 \cdots w_{L_s})$ into a Bidirectional Long Short-Term Memory (LSTM) layer. As such, the output of the BiLSTM is described as follows:

$$h_t = BiLSTM(h_{t-1}, w_t) \tag{1}$$

where h_t is the hidden representation at step t. Given a sequence of inputs $w_1, w_2 \cdots w_L$, the output of the BiLSTM layer is a sequence of hidden states $h_1, h_2 \cdots h_L$. Note that since we use a bidirectional LSTM, then $h_t \in \mathbb{R}^{2r}$ where r is the dimensionality of the BiLSTM layer. In our case r is set to $\frac{d}{2}$ such that the output vector has dimensionality d.

Sentence Attention To learn a final sentence representation of the sentence, we adopt an attention mechanism. The attention mechanism is defined by the following equations:

$$\mathbf{Y} = \tanh(\mathbf{W}_y \, H) \tag{2}$$

$$a_c = softmax(w_y^T \, \mathbf{Y}) \text{ and } s = H \, a_c^T \tag{3}$$

where $s \in \mathbb{R}^d$ is the output sentence representation, $\mathbf{W}_y \in \mathbb{R}^{d \times d}$ and $w_y \in \mathbb{R}^d$ are parameters of the attention layer. Intuitively, the attention layer learns to pay attention to important segments of the sentence, producing a weighted representation of the hidden states of the BiLSTM layer.

3.3 Learning Auxiliary Lexicon Embedding

This layer aims to learn a single d-dimensional lexicon-based representation of the sentence. In order to learn the lexicon embedding, our model adopts a two layer attention mechanism, namely the contextual attention (CA) and contrastive co-attention (CC).

Contextual Attention (CA) We utilize an attention mechanism to learn the relative importance of each lexicon word based on the sentence representation. This layer is applied to and is functionally identical for both P and N. As such, for notational convenience, we use Q to represent either positive (P) or negative (N), and G to represent the maximum number of lexicon words. Let $Q \in \mathbb{R}^{G \times d}$ be a sequence of lexicon words and $H \in \mathbb{R}^{L_s \times d}$ be the intermediate hidden representations obtained from the contextual BiLSTM layer:

$$M = tanh(Q \, \mathbf{U} \, H^T) \tag{4}$$

where $\mathbf{U} \in \mathbb{R}^{d \times d}$ are the parameters of this layer. Next, we apply a column-wise max pooling of M. The key idea is to generate an attention vector:

$$a = sm(\max_{col}(M)) \; ; \; c_i = a_i * q_i \tag{5}$$

where $a \in \mathbb{R}^G$. The softmax function normalizes the values of the vector $\max_{col}(M)$ into a probability distribution. To learn the context-sensitive weight importance of each lexicon word, we then apply the attention vector on Q. $C = \{c_1, c_2 \cdots c_G\}$ is the context-sensitive lexicon representation of Q. Intuitively, the CA mechanism attends to each lexicon word based on its *maximum* influence on each word of the main sentence. There are several advantages to our context attention mechanism. Unlike Lexicon RNN which simply extracts the hidden representation (generated from BiLSTM) of the lexicon word, our approach has a global view of the entire sentence which allows each lexicon word to benefit from wider contextual knowledge as opposed to being limited to the temporal compositionality provided by the BiLSTM layer. Overall, the outputs of this layer are two matrices (positive and negative lexicon embeddings) which are context-sensitive.

---

[1] For our experiments, we mainly use ST140 lexicon and therefore use $score > 0$ to separate positive and negative words. Notably, about $\approx 85\%$ of all words in the sentence has a lexicon assignment.

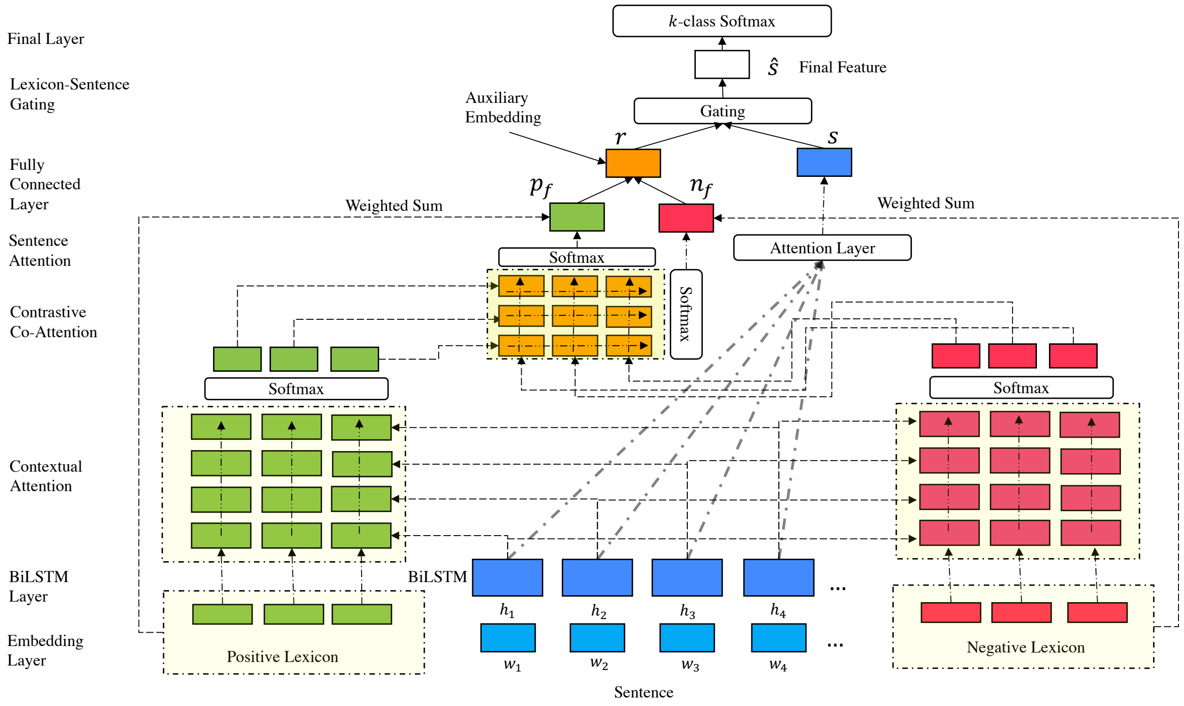

Figure 1: Illustration of our proposed Attentive Gated Lexicon Reader model (*best viewed in color*).

Note that these lexicon embeddings retain their dimensionality passing through this layer.

Contrastive Co-Attention (CC) This layer aims to model the contrast between polarities. Intuitively, this layer helps to model sentences with double or conflicting polarities. It also aims to negate strongly positive or negative words in the case of a neutral context. In order to do so, we employ a contrastive co-attention model that learns to weight the relative importance of each positive lexicon word based on the negative lexicon (and vice versa). We accept the contextualized positive and negative lexicon embeddings from the previous layer as an input. Let $\hat{P} \in \mathbb{R}^{G \times d}$ be the contextualized positive lexicons and $\hat{N} \in \mathbb{R}^{G \times d}$ be the contextualized negative lexicons, our co-attention layer learns a soft attention alignment between positive and negative lexicon embeddings. Similar to our contextual attention layer, we first learn an affinity matrix Z that models the relationship between positive and negative lexicon embeddings:

$$Z = tanh(P\,\mathbf{A}\,N^T) \tag{6}$$

Next, we apply both column-wise and row-wise max-pooling on the affinity matrix Z to obtain two attention vectors. The two attention vectors are then normalized with the softmax function (denoted as sm).

$$a_p = sm(\max_{col}(Z))\ ;\ a_n = sm(\max_{row}(Z)) \tag{7}$$

a_p is the attention vector for the positive lexicon embeddings and a_n is the attention vector for the negative lexicon embeddings. The final vector representations are therefore:

$$p_f = P\,a_p^\top\ ;\ n_f = N\,a_n^\top \tag{8}$$

where $p_f \in \mathbb{R}^d$ and $n_f \in \mathbb{R}^d$ are the vector representations for positive lexicon and negative lexicon respectively. Note that this layer, unlike the contextual attention layer, is named 'co-attention' because both P and N are both 'attended over' concurrently. It is also good to note that attentions are applied over the original embeddings P, N and not the *contextualized* embeddings $\hat{P}, \hat{N}$.

Fully-Connected Layer Next, we pass the concatenation of p and n through a fully-connected layer to learn the final representation for the auxiliary lexicon embedding, i.e., $r = tanh(\mathbf{W}_h\ ([p; n]) + b_h)$ where $\mathbf{W}_h \in \mathbb{R}^{2d \times d}$ are the parameters of the hidden layer and b_h is the bias value. The output $r \in \mathbb{R}^d$ is the final auxiliary lexicon-based embedding.

Learning Final Representations To combine the lexicon-based representation with the sentence representation, we adopt a gating mechanism.

$$\hat{s} = \sigma(w_g \odot r) \odot r + (1 - \sigma(w_g \odot r)) \odot s \quad (9)$$

where $w_g \in \mathbb{R}^d$ are the parameters of this layer, σ is the sigmoid function. $\hat{s}$ is the overall final representation.

3.4 Final Layer and Optimization

Finally, we pass $\hat{s}$ the overall final representation into a softmax layer.

$$y = softmax(\mathbf{W}_f \, \hat{s} + b_f) \quad (10)$$

where $y \in \mathbb{R}^k$, where k is the number of classes (2 for positive and negative and 3 including neutral). $\mathbf{W}_f$ and b_f are standard parameters of a linear regression layer. For optimization, we adopt the standard cross entropy loss function with L2 regularization.

$$L = -\sum_{i=1}^{N} [y_i \log o_i + (1 - y_i) \log(1 - o_i)] + R \quad (11)$$

where o is the output of the softmax layer and $R = \lambda \, \|\psi\|_2^2$ is the L2 regularization.

4 Empirical Evaluation

This section describes our empirical experiments.

4.1 Evaluation Procedure

In this section, we describe the datasets used, evaluation metric and implementation details.

Datasets We conduct our experiments[2] on subsets of sentiment analysis benchmarks from SemEval 2013 (Nakov et al., 2013), SemEval 2014 (Rosenthal et al., 2014) and SemEval 2016 (Nakov et al., 2016). More specifically, we focus on the sentence level of sentiment analysis and evaluate on the datasets of SemEval 2013 task 2, SemEval 2014 task 9 and SemEval 2016 task 4, which we will name as *SemEval13*, *SemEval14* and *SemEval16* respectively in this section. For fair comparison, we use the same setting of training, development and testing as in SemEval competitions. To further evaluate the performance of methods when data is limited, for *SemEval16*, we experiment on two different training settings. The first, TRAIN, uses only the 2016 training set while the other, TRAIN-ALL, appends the 2013 training set to the 2016 training set, following the official setting of SemEval 2016 while TRAIN explores the setting where training data is limited.

Evaluation Metrics We evaluate on two settings, i.e., 3-way (positive, negative and neutral) and also binary (positive and negative) classification. We report the accuracy and macro-averaged F1 score for all settings.

Compared Baselines In this section, we list the neural baselines we use for comparisons.

- **NBOW-MLP** (Neural Bag-of-Words + Multi-layered Perceptron) is a simple sum of all word embeddings which is connected to a 2-layer MLP of 100 dimensions.
- **CNN** (Convolutional Neural Network) is another popular neural encoder for learning sentence representations. We use a filter size of 3 and 150 filters.
- **BiLSTM** (Bidirectional Long Short-Term Memory) is a standard strong neural baseline for many NLP tasks. The size of the LSTM is set to 150.
- **AT-BiLSTM** (Attention-based BiLSTM) is an extension of the BiLSTM model with neural attention.
- **Lexicon RNN** (Lexicon Recurrent Neural Network) is the model of (Teng et al., 2016). The first neural model that incorporates sentiment lexicon. The size of the BiLSTM in this model is also set to 150.

All models except Lexicon RNN optimize the softmax cross entropy loss. The authors use Lexicon RNN for binary and 5-way classification. In order to adapt Lexicon RNN to 3-way classification (positive, negative, neutral), we adapt the 5-way formulation that minimizes the MSE (mean square error) loss to 3-way. The output is scaled[3] to $s \in [-1, 1]$ where $s > 0.25$ is treated as positive, $s < -0.25$ is treated as negative and everything in between is neutral.

---

[2] SemEval 2015 was omitted due to space in favor of SemEval 2016 since testing sets are significantly larger in the latter.

[3] We experimented with other thresholds but found 0.25 to work the best.

	SemEval13				*SemEval14*					
	3-way		Binary		3-way		Binary		AVG	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
NBOW-MLP	65.18	60.94	85.44	82.30	65.68	60.35	89.44	81.60	76.44	71.30
CNN	71.41	68.23	85.74	82.60	70.05	66.22	89.86	82.09	79.27	74.79
BiLSTM	72.06	70.00	85.89	82.79	71.62	68.34	90.20	83.09	79.94	76.06
AT-BiLSTM	72.21	69.89	86.13	83.22	71.83	68.01	90.20	83.46	80.09	76.15
Lexicon RNN	69.97	68.69	86.43	83.54	70.75	67.06	**91.13**	**84.60**	79.57	75.97
AGLR	**73.27**	**71.79**	**86.72**	**84.18**	**73.29**	**70.48**	90.37	84.15	**80.91**	**77.65**

Table 1: Experimental results on test datasets *SemEval2013* and *SemEval2014*.

	Sem2016 (TRAIN)				*Sem2016* (TRAIN-ALL)					
	3-way		Binary		3-way		Binary		AVG	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
NBOW-MLP	54.31	52.90	79.69	77.33	61.09	55.91	84.90	82.01	70.00	67.04
CNN	54.67	52.17	79.79	75.28	62.68	57.71	84.10	81.31	70.31	66.62
BiLSTM	55.57	52.33	81.90	77.12	63.26	60.31	85.89	84.14	71.66	68.50
AT-BiLSTM	56.95	54.53	80.09	73.93	64.20	61.64	86.77	83.67	72.00	68.44
Lexicon RNN	51.02	50.45	81.72	79.00	61.41	60.50	86.68	83.82	70.21	68.44
AGLR	**59.01**	**56.67**	**82.79**	**80.10**	**66.62**	**64.36**	**87.16**	**84.98**	**73.90**	**71.53**

Table 2: Experimental results on *Sem2016* with two training settings TRAIN and TRAIN-ALL.

Implementation Details Text are lowercased, tokenized using NLTK's tweet tokenizer and padded to the maximum sequence length of the dataset. For Lexicon RNN and AGLR, we use[4] the ST140 (Sentiment140) lexicon which was created by distant supervision (Mohammad et al., 2013). The maximum numbers of positive lexicons and negative lexicons extracted per sample are tuned amongst $\{5, 8, 10\}$. Models are trained for a maximum of 30 epochs with early stopping if the performance on the development set does not improve after 5 epochs. Results reported are the test scores from the model that performed best on the development set. The batch size is tuned amongst $\{50, 100, 300\}$. L2 Regularization tuned amongst $\{10^{-6}, 10^{-7}, 10^{-8}\}$. Dropout is set to 0.5. We optimized all networks with the RMSprop optimizer and with initial learning rate tuned amongst $\{0.01, 0.005, 0.001\}$. Word embeddings are initialized with Glove 27B (Pennington et al., 2014) ($d = 200$) trained on tweets and are trainable parameters. The size of the BiLSTM is $d = 100$.

4.2 Experimental Results

Table 1 and Table 2 report the results of our experiments. The results on TRAIN-ALL are higher than TRAIN for *SemEval16* in lieu of the larger dataset. Firstly, we observe that our proposed AGLR outperforms all neural baselines on 3-way classification. The overall performance of AGLR achieves state-of-the-art performance. On average, AGLR outperforms Lexicon RNN and AT-BiLSTM by $1\% - 3\%$ in terms of F1 score. We also observe that AGLR always improves AT-BiLSTM which ascertains the effectiveness of learning auxiliary lexicon embeddings. The key idea here is that the auxiliary lexicon embeddings provide a *different view* of the sentence which supports the network in making predictions.

We also observe that Lexicon RNN does not handle 3-way classification well. Even though it has achieved good performance on binary classification, the performance on 3-way classification is lackluster (the performance of AGLR outperforms Lexicon RNN by up to 8% on *SemEval16* TRAIN). This could also be attributed to the MSE based loss function. Conversely, by learning an auxiliary embedding (instead of a scalar score), our model becomes more flexible at the final layer and can be adapted to using a k softmax function. Finally, we observe that BiLSTM and AT-BiLSTM outperform Lexicon RNN on average with Lexicon RNN being slightly better on binary classification.

---

[4] We also used Bing Liu's opinion lexicon but found it to perform slightly worse.

Comparisons against Top SemEval Systems Table 3 reports the results of our proposed approach against the top team of each SemEval run, i.e., *NRC-Canada* (Mohammad et al., 2013) for 2013 Task 2, *Team-X* (Miura et al., 2014) for 2014 Task 9, *SwissCheese* (Deriu et al., 2016) for 2016 Task 4. We follow the exact training datasets allowed for each SemEval run. Following the competition setting, with the exception of accuracy for SemEval 2016, all metrics reported are the macro averaged F1 score of positive and negative classes.

		Top System	Ours
SemEval13	Tweets	69.02	**70.10**
SemEval14	Tweets	70.96	**71.11**
	Sarcasm	56.50	**58.87**
	LiveJournal	69.44	**72.52**
SemEval16	Tweets	**63.30**	61.90
	Tweets (Acc)	64.60	**66.60**

Table 3: Comparisons against top SemEval systems. Results reported are the F^{PN} metric scores used in the SemEval tasks.

We observe that AGLR achieves competitive performance relative to the top runs in SemEval 2013, 2014 and 2016. It is good to note that SemEval approaches are often heavily engineered containing ensembles and many handcrafted features which include extensive use of sentiment lexicons, POS tags and negation detectors. Recent SemEval runs gravitate towards neural ensembles. For instance, the winning approach for *SwissCheese* (SemEval 2016) uses an ensemble of 6 CNN models along with a meta-classifier (random forest classifier). On the other hand, our proposed model is *a single neural model*. In addition, SwissCheese also uses emoticon-based distant supervision which exploits a huge corpus of sentences (millions) for training. Conversely, our approach only uses the 2013 and 2016 training sets which are significantly smaller. Given these conditions, we find it remarkable that our single model is able to achieve competitive performance relative to the extensively engineered approach of SwissCheese. Moreover, we actually outperform significantly in terms of pure accuracy. AGLR performs competitively on SemEval 2013 and 2014 as well. The good performance on the *sarcasm* dataset could be attributed to our contrastive attention mechanism.

Ablation Study In this section, we study the impacts and contribution of the different components of our model. Specifically, we tested 3 settings. The first, we removed CC only. In this case, positive and negative lexicons are *summed* instead of a *weighted summed* using attention. In the next setting, we removed CA only. Similarly, embeddings are summed instead of attentively summed. Finally, we removed both CA and CC. In this case, all lexicons are considered neural bag-of-words (NBOW) and passed to the MLP layer. Table 4 shows the results of this ablation study on *SemEval16* using the TRAIN-ALL setting.

Model	Acc	F1
AT-BiLSTM only	64.20 (-2.42)	61.64 (-2.96)
AGLR (*-CA and -CC*)	62.42 (-4.20)	59.48 (-4.88)
AGLR (*-CA*)	65.81 (-0.81)	60.47 (-3.89)
AGLR (*-CC*)	64.38 (-2.24)	61.26 (-3.10)
AGLR	66.62	64.36

Table 4: Ablation study on *SemEval16* (TRAIN-ALL)

It is clear that both CC and CA are critical to the performance of AGLR. Removing either or both can cause performance to degrade. In particular, we also observe that CA seems to be less important than CC, i.e., performance drops more as compared to removing CA. We also note that removing both and a simple NBOW for lexicons can degrade performance since the base AT-BiLSTM is better than using NBOW lexicons as an auxiliary support. As such, the design of the auxiliary embeddings must be treated with care.

Qualitative Analysis In order to study what are the specific roles of the contextual and contrastive attention mechanism, we inspect the attention maps over the positive and negative lexicons. We use the following example in which the ground truth label is *positive*: *"Very excited about Tuesday night @user free iced coffee and smoothies courtesy of Dunkin Donuts will be set up."*. Figure 2a shows the attention maps for contextual attention. We observe that contextual attention focuses more on the context, i.e., focusing on words such as 'night', 'iced coffee' and 'smoothies'. On the other hand, Figure 2b shows the attention maps after contrastive attention. We observe that contrastive attention learns more polarity specific attentions, i.e., shifting some focus to *'very excited'*. We also observe that the contrastive attention tends to shift its attention weights to less meaningful words for the negative lexicon if the ground truth label is positive (and vice versa). We believe that this indicates that there is an absence of negative sentiment.

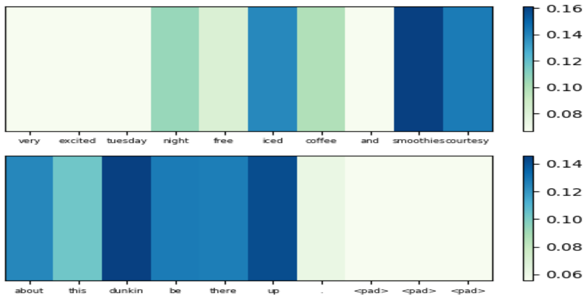

(a) Attention over positive and negative lexicons for contextual attention.

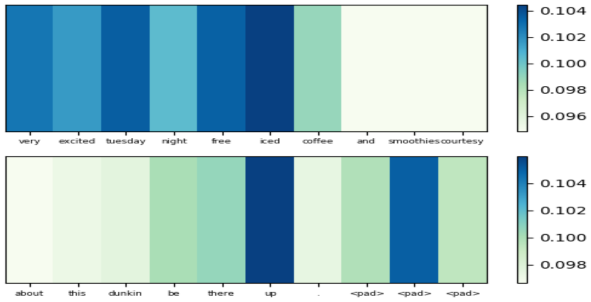

(b) Attention over positive and negative lexicons for contrastive attention.

Figure 2: Visualization of Contextual Attention and Contrastive Co-Attention.

5 Conclusion

We proposed a novel method of incorporating lexicons into neural models for the task of sentiment analysis. More specifically, we learn an auxiliary lexicon embedding using neural attention. Our proposed model AGLR achieves an overall state-of-the-art performance on multiple benchmark datasets outperforming strong neural baselines such as AT-BiLSTM and Lexicon RNN. The performance of AGLR is also competitive relative to top SemEval systems which utilized neural ensembles or very extensive feature engineering.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the workshop on languages in social media*. Association for Computational Linguistics, pages 30–38.

Stefanos Angelidis and Mirella Lapata. 2017. Multiple instance learning networks for fine-grained sentiment analysis. *arXiv preprint arXiv:1711.09645* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *CoRR* abs/1611.01576.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurelien Lucchi, Valeria De Luca, and Martin Jaggi. 2016. Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision.

Li Dong, Furu Wei, Chuanqi Tan, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, pages 6645–6649.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*. pages 168–177. https://doi.org/10.1145/1014052.1014073.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, page 1367.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. 2017. Deep memory networks for attitude identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. pages 671–680.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5(1):1–167.

Pengfei Liu, Shafiq R. Joty, and Helen M. Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of* *the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 1433–1443. http://aclweb.org/anthology/D/D15/D15-1168.pdf.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Yasuhide Miura, Shigeyuki Sakaki, Keigo Hattori, and Tomoko Ohkuma. 2014. Teamx: A sentiment analyzer with enhanced lexicon mapping and weighting scheme for unbalanced data.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242* .

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *SemEval@ NAACL-HLT*. pages 1–18.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. volume 2, pages 312–320.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Improving twitter sentiment classification using topic-enriched multi-prototype word embeddings.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 73–80. http://www.aclweb.org/anthology/S14-2009.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. Citeseer.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 2440–2448.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective lstms for target-dependent sentiment classification. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 3298–3307.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014a. Building large-scale twitter-specific sentiment lexicon: A representation learning approach.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014b. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1555–1565.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. *arXiv preprint arXiv:1712.05403* .

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018a. Hermitian co-attention networks for text matching in asymmetrical domains. In *IJCAI*. pages 4425–4431.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018b. Multi-cast attention networks for retrieval-based question answering and response prediction. *arXiv preprint arXiv:1806.00778* .

Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. 2018c. Reasoning with sarcasm by reading in-between. *arXiv preprint arXiv:1805.02856* .

Zhiyang Teng, Duy-Tin Vo, and Yue Zhang. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 1629–1638. http://aclweb.org/anthology/D/D16/D16-1169.pdf.

Zhiyang Teng and Yue Zhang. 2016. Bidirectional tree-structured lstm with head lexicalization. *arXiv preprint arXiv:1611.06788* .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 5998–6008.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*. pages 347–354. http://aclweb.org/anthology/H/H05/H05-1044.pdf.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR* abs/1611.01604.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis.

Contextual Inter-modal Attention for Multi-modal Sentiment Analysis

Deepanway Ghosal[†], **Md Shad Akhtar**[†], **Dushyant Chauhan**[†], **Soujanya Poria**[⋆],
Asif Ekbal[†] **and Pushpak Bhattacharyya**[†]
[†] *Department of Computer Science & Engineering, Indian Institute of Technology Patna, India*
`{deepanway.me14,shad.pcs15,1821CS18,asif,pb}@iitp.ac.in`
[⋆] *School of Computer Science and Engineering, Nanyang Technological University, Singapore*
`sporia@ntu.edu.sg`

Abstract

Multi-modal sentiment analysis offers various challenges, one being the effective combination of different input modalities, namely *text*, *visual* and *acoustic*. In this paper, we propose a recurrent neural network based multi-modal attention framework that leverages the contextual information for utterance-level sentiment prediction. The proposed approach applies attention on multi-modal multi-utterance representations and tries to learn the contributing features amongst them. We evaluate our proposed approach on two multi-modal sentiment analysis benchmark datasets, *viz.* CMU Multi-modal Opinion-level Sentiment Intensity (CMU-MOSI) corpus and the recently released CMU Multi-modal Opinion Sentiment and Emotion Intensity (CMU-MOSEI) corpus. Evaluation results show the effectiveness of our proposed approach with the accuracies of 82.31% and 79.80% for the MOSI and MOSEI datasets, respectively. These are approximately 2 and 1 points performance improvement over the state-of-the-art models for the datasets.

1 Introduction

Traditionally, sentiment analysis (Pang and Lee, 2005, 2008) has been applied to a wide variety of texts (Hu and Liu, 2004; Liu, 2012; Turney, 2002; Akhtar et al., 2016, 2017; Mohammad et al., 2013). In contrast, multi-modal sentiment analysis has recently gained attention due to the tremendous growth of many social media platforms such as YouTube, Instagram, Twitter, Facebook (Chen et al., 2017; Poria et al., 2016, 2017d,b; Zadeh et al., 2017, 2016) etc. It depends on the information that can be obtained from more than one modality (e.g. *text*, *visual* and *acoustic*) for the analysis. The motivation is to leverage the varieties of (often distinct) information from multiple sources for building an efficient system. For example, it is a non-trivial task to detect the sentiment of a sarcastic sentence "*My neighbours are home!! it is good to wake up at 3am in the morning.*" as *negative* considering only the textual information. However, if the system has access to some other sources of information, e.g. *visual*, it can easily detect the unpleasant gestures of the speaker and would classify it with the *negative* sentiment polarity. Similarly, for some instances acoustic features such as *intensity*, *pitch*, *pause* etc. have important roles to play in the correctness of the system. However, combining these information in an effective manner is a non-trivial task that researchers often have to face (Zadeh et al., 2017; Chen et al., 2017).

A video provides a good source for extracting multi-modal information. In addition to the *visual frames*, it also provides information such as *acoustic* and *textual* representation of *spoken language*. Additionally, a speaker can utter multiple utterances in a single video and these utterances can have different sentiments. The sentiment information of an utterance often has inter-dependence on other contextual utterances. Classifying such an utterance in an independent manner poses many challenges to the underlying algorithm.

In this paper, we propose a novel method that employs a recurrent neural network based multi-modal multi-utterance attention framework for sentiment prediction.We hypothesize that applying attention to contributing neighboring utterances and/or multi-modal representations may assist the network to learn in a better way. The main challenge in multi-modal sentiment analysis lies in the proper utilization of the information extracted from multiple modalities. Although it is often argued that incorporation of all the available modalities is always beneficial for enhanced performance, it must be noted that not all the modalities play equal role. Another concern in multi-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3454–3466
Brussels, Belgium, October 31 - November 4, 2018.

modal framework is that the presence of noise in one modality can affect the overall performance. To better address these concerns we propose a novel fusion method by focusing on inter-modality relations computed between the target utterance and its context. We argue that in multi-modal sentiment classification, not only the relation among two modalities of the same utterance is important, but also relatedness with the modalities across its context are important.

Think of an utterance U_t that constitutes of three modalities, say A_t (i.e. audio), V_t (i.e. visual) and T_t (i.e. text). Let us also assume U_k being a member of the contextual utterances consisting of the modalities - A_k, V_k and T_k. In this case, our model computes the relatedness among the modalities (for e.g., V_t and T_k) of U_t and U_k in order to produce a richer multi-modal representation for final classification. The attention mechanism is then used to attend to the important contextual utterances having higher relatedness or similarity (computed using inter-modality correlations) with the target utterance.

Unlike previous approaches that simply apply attentions over the contextual utterance for classification, we attend over the contextual utterances by computing correlations among the modalities of the target utterance and the context utterances. This explicitly helps us to distinguish which modalities of the relevant contextual utterances are more important for sentiment prediction of the target utterance. The model facilitates this modality selection by attending over the contextual utterances and thus generates better multi-modal feature representation when these modalities from the context are combined with the modalities of the target utterance. We evaluate our proposed approach on two recent benchmark datasets, i.e. CMU-MOSI (Zadeh et al., 2016) and CMU-MOSEI (Zadeh et al., 2018c), with one being the largest (CMU-MOSEI) available dataset for multi-modal sentiment analysis (c.f. Section 4.1). Evaluation shows that the proposed attention framework attains better performance than the state-of-the-art systems for various combinations of input modalities (i.e. *text*, *visual* & *acoustic*).

The main contributions of our proposed work are three-fold: **a)** we propose a novel technique for multi-modal sentiment analysis; **b)** we propose an effective attention framework that leverages contributing features across multiple modalities and neighboring utterances for sentiment analysis; and **c)** we present the state-of-the-art systems for sentiment analysis in two different benchmark datasets.

2 Related Work

A survey of the literature suggests that multi-modal sentiment prediction is relatively a new area as compared to textual based sentiment prediction (Morency et al., 2011; Mihalcea, 2012; Poria et al., 2016, 2017b; Zadeh et al., 2018a). A good review covering the literature from uni-modal analysis to multi-modal analysis is presented in (Poria et al., 2017a). An application of multi-kernel learning based fusion technique was proposed in (Poria et al., 2016), where they employed deep convolutional neural networks for extracting the textual features and fused it with other (*visual* & *acoustic*) modalities for prediction.

Zadeh et al. (2016) introduced the multi-modal dictionary to better understand the interaction between facial gestures and spoken words when expressing the sentiment. Authors introduced the MOSI dataset, the first of its kind to enable the studies of multi-modal sentiment intensity analysis. Zadeh et al. (2017) proposed a Tensor Fusion Network (TFN) model to learn the intra-modality and inter-modality dynamics of the three modalities (i.e. text, visual and acoustic). They reported the improved accuracy using multi-modality on the CMU-MOSI dataset. An application to leverage on the gated multi-modal embedded Long Short Term Memory (LSTM) with temporal attention (GME-LSTM(A)) for the word-level fusion of multi-modality inputs is proposed in (Chen et al., 2017). The Gated Multi-modal Embedding (GME) alleviates the difficulties of fusion while the LSTM with Temporal Attention (LSTM(A)) performs word-level fusion.

The works mentioned above did not take contextual information into account. Poria et al. (2017b) proposed a LSTM based framework that leverages the contextual information to capture the inter-dependencies between the utterances. In another work, Poria et al. (2017d) proposed an user opinion based framework to combine the three modality inputs (i.e. *text, visual & acoustic*) by applying a multi-kernel learning based method. Zadeh et al. (2018a) proposed multi-attention blocks (MAB) to capture information across three modalities (*text, visual & acoustic*). They reported improved accuracies in the range of

2-3% over the state-of-the-art models for the different datasets.

The fundamental difference between our proposed method and the existing works is that our framework applies focus on the neighboring utterances to leverage contextual information for utterance-level sentiment prediction. To the best of our knowledge, our current work is the very first of its kind that attempts to employ multi-modal attention block (exploiting neighboring utterances) for sentiment prediction. We use *multi-modal attention* framework that leverages contributing features across *multiple modalities* and the *neighboring utterances* for sentiment analysis.

3 Proposed Methodology

In our proposed framework, we aim to leverage the multi-modal and contextual information for predicting the sentiment of an utterance. Utterances of a particular speaker in a video represent the time series information and it is logical that the sentiment of a particular utterance would affect the sentiments of the other neighboring utterances. To model the relationship with the neighboring utterances and multi-modality, we propose a recurrent neural network based multi-modal attention framework. The proposed framework takes multi-modal information (i.e. *text*, *visual* & *acoustic*) for a sequence of utterances and feeds it into three separate bi-directional Gated Recurrent Unit (GRU) (Cho et al., 2014). This is followed by a dense (fully-connected) operation which is shared across the time-steps or utterances (one each for text, visual & acoustic). We then apply multi-modal attention on the outputs of the dense layers. The objective is to learn the joint-association between the multiple modalities & utterances, and to emphasize on the contributing features by putting more attention to these. In particular, we employ bi-modal attention framework, where an attention function is applied to the representations of *pairwise* modalities i.e. *visual-text*, *text-acoustic* and *acoustic-visual*. Finally, the outputs of pairwise attentions along with the representations are concatenated and passed to the *softmax* layer for classification. We call our proposed architecture ***Multi-Modal Multi-Utterance - Bi-Modal Attention (MMMU-BA)*** framework. An overall architecture of the proposed *MMMU-BA* framework is illustrated in Figure 1. Please refer to Figure 3 in appendix for illustration of attention computation.

For comparison, we also experiment with two other variants of the proposed *MMMU-BA* framework i.e. a). *Multi-Modal Uni-Utterance-Self Attention (MMUU-SA)* framework and b). *Multi-Utterance-Self Attention (MU-SA)* framework. The architecture of these variants differ with respect to the attention computation module and the naming conventions "*MMMU*", "*MMUU*" or "*MU*" signify the information that participates in the attention computation. For example, in *MMMU-BA*, we compute attention over the multi-modal and multi-utterance inputs, whereas in *MMUU-SA*, the attention is computed over the mutli-modal but uni-utterance inputs. In contrast, we compute attention over only multi-utterance inputs in *MU-SA*. Rest of the components for all the three variants remain same.

3.1 Multi-modal Multi-utterance - Bi-modal Attention (*MMMU-BA*) Framework

Assuming a particular video has '*u*' utterances, the raw *utterance level* multi-modal features are represented as $T_R \in \mathbb{R}^{u\times 300}$ (*raw text*), $V_R \in \mathbb{R}^{u\times 35}$ (*raw visual*) and $A_R \in \mathbb{R}^{u\times 74}$ (*raw acoustic*). Three separate Bi-GRU layers with forward & backward state concatenation are first applied on the raw data followed by the fully-connected dense layers, resulting in $T \in \mathbb{R}^{u\times d}$ (*text*), $V \in \mathbb{R}^{u\times d}$ (*visual*) and $A \in \mathbb{R}^{u\times d}$ (*acoustic*), where '*d*' is the number of neurons in the dense layer. Finally, *pairwise-attentions* are computed on various combinations of three modalities- (*V*, *T*), (*T*, *A*) & (*A*, *V*). In particular the attention between *V* and *T* is computed as follows:

- ***Bi-modal Attention:*** Modality representations of *V* & *T* are obtained from the Bi-GRU network, and hence contain the contextual information of the utterances for each modality. At first, we compute a pair of matching matrices $M_1, M_2 \in \mathbb{R}^{u\times u}$ over two representations that account for the cross-modality information.

$$M_1 = V.T^T \quad \& \quad M_2 = T.V^T$$

- ***Multi-Utterance Attention:*** As mentioned earlier, in the proposed model we aim to leverage the contextual information of each utterance for the prediction. We compute the probability distribution scores ($N_1 \in \mathbb{R}^{u\times u}$ & $N_2 \in \mathbb{R}^{u\times u}$) over each utterance of bi-modal attention matrices M_1 & M_2 using a softmax function. This essentially computes the attention weights for the contextual

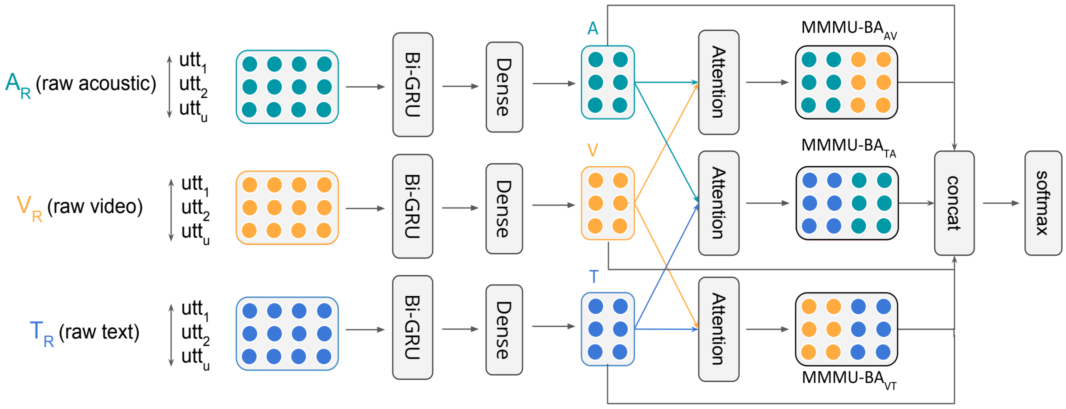

Figure 1: Overall architecture of the proposed MMMU-BA framework.

utterances. Finally, soft attention is applied over the multi-modal multi-utterance attention matrices to compute the modality-wise attentive representations (i.e. O_1 & O_2).

$$N_1(i,j) = \frac{e^{M_1(i,j)}}{\sum_{k=1}^{u} e^{M_1(i,k)}} \quad \text{for} \quad i,j = 1,..,u$$
$$N_2(i,j) = \frac{e^{M_2(i,j)}}{\sum_{k=1}^{u} e^{M_2(i,k)}} \quad \text{for} \quad i,j = 1,..,u.$$
$$O_1 = N_1.T \quad \& \quad O_2 = N_2.V$$

- ***Multiplicative Gating & Concatenation:*** Finally, a multiplicative gating function following (Dhingra et al., 2016) is computed between the multi-modal utterance specific representations of each individual modality and the other modalities. This element-wise matrix multiplication assists in attending to the important components of multiple modalities and utterances.

$$A_1 = O_1 \odot V \quad \& \quad A_2 = O_2 \odot T$$

Attention matrices A_1 & A_2 are then concatenated to obtain the *MMMU-BA$_{VT}$* $\in \mathbb{R}^{u\times 2d}$ between V and T.

$$\textit{MMMU-BA}_{VT} = concat[A_1, A_2]$$

***MMMU-BA$_{AV}$* & *MMMU-BA$_{TA}$* computations:** Similar to *MMMU-BA$_{VT}$*, we follow the same procedure to compute *MMMU-BA$_{AV}$* & *MMMU-BA$_{TA}$*. For a data source comprising of raw visual (V_R), acoustic (A_R) & text (T_R) modalities, at first, we compute the bi-modal attention pairs for each combination i.e. *MMMU-BA$_{VT}$*, *MMMU-BA$_{AV}$* & *MMMU-BA$_{TA}$*. Finally, motivated by the residual skip connection network (He et al., 2016), we concatenate the bi-modal attention pairs with individual modalities (i.e. *V*, *A* & *T*) to boost the gradient flow to the lower layers. This concatenated feature is then used for final classification.

3.2 Multi-Modal Uni-Utterance - Self Attention (*MMUU-SA*) Framework

MMUU-SA framework does not account for information from the other utterances at the attention level, rather it utilizes multi-modal information of single utterance for predicting the sentiment. For a video having *'q'* utterances, *'q'* separate attention blocks are needed, where each block computes the self-attention over multi-modal information of a single utterance. Let $X_{u_p} \in \mathbb{R}^{3\times d}$ is the information matrix of the p^{th} utterance where the three *'d'* dimensional rows are the outputs of the dense layers for the three modalities.

The attention matrix $A_{u_p} \in \mathbb{R}^{3\times d}$ is computed separately for, p = 1^{st}, 2^{nd}, ... q^{th} utterances. Finally, for each utterance p, A_{u_p} and X_{u_p} are concatenated and passed to the output layer for classification. Please refer to the appendix for more details.

3.3 Multi-Utterance - Self Attention (*MU-SA*) Framework

In *MU-SA* framework, we apply self attention on the utterances of each modality separately, and use these for classification. In contrast to *MMUU-SA* framework, *MU-SA* utilizes the contextual information of the utterances at the attention level. Let, $T \in \mathbb{R}^{u\times d}$ (*text*), $V \in \mathbb{R}^{u\times d}$ (*visual*) and $A \in \mathbb{R}^{u\times d}$ (*acoustic*) are the outputs of the dense layers. For the three modalities, three separate attention blocks are required, where each block takes multi-utterance information of a single modality and computes the self attention matrix. Attention matrices A_t, A_v and A_a are computed for text, visual and acoustic, respectively. Finally A_v, A_t, A_a, V, T & A are concatenated and passed to the output layer for classification.

4 Datasets, Experiments and Analysis

In this section we describe the datasets used for our experiments and report the results along with the necessary analysis.

4.1 Datasets

We evaluate our proposed approach on two benchmark datasets, namely CMU Multi-modal Opinion-level Sentiment Intensity (CMU-MOSI) corpus (Zadeh et al., 2016) and the recently published CMU Multi-modal Opinion Sentiment and Emotion Intensity (CMU-MOSEI) dataset (Zadeh et al., 2018c). CMU-MOSI dataset consists of 93 videos spanning over 2199 utterances. Each utterance has a sentiment label associated with it. It has 52, 10 & 31 videos in training, validation & test set accounting for 1151, 296 & 752 utterances.

CMU-MOSEI has 3229 videos with 22676 utterances from more than 1000 online YouTube speakers. The training, validation & test set comprise of 16216, 1835 & 4625 utterances, respectively. More details about these datasets are presented in the appendix.

Each utterance in CMU-MOSI dataset has been annotated as either positive or negative, whereas in CMU-MOSEI dataset labels are in the continuous range of -3 to +3. However, in this work we project the instances of CMU-MOSEI in a *two-class classification* setup with $values \geq 0$ signify positive sentiments and $values < 0$ signify negative sentiments. We adopt such a strategy to be consistent with the previous published works on CMU-MOSI datasets (Poria et al., 2017b; Chen et al., 2017).

4.2 Feature extraction

We use the CMU-Multi-modal Data SDK[1] (Zadeh et al., 2018a) for feature extraction. For MOSEI dataset, *word-level* features were provided where *text* features were extracted by *GloVe embeddings*, *visual* features by *Facets*[2] & *acoustic* features by *CovaRep* (Degottex et al., 2014). Thereafter, we compute the average of *word-level* features in an utterance to obtain the *utterance-level* features. For each word, the dimension of the feature vector is set to 300 (*text*), 35 (*visual*) & 74 (*acoustic*).

In contrast, for MOSI dataset we use *utterance-level* features[3] provided in (Poria et al., 2017b). These *utterance-level* features represent the outputs of a convolutional neural network (Karpathy et al., 2014), 3D convolutional neural network (Ji et al., 2013) & openSMILE (Eyben et al., 2010) for *text*, *visual* & *acoustic* modalities, respectively. Dimensions of *utterance-level* features are 100, 100 & 73 for *text*, *visual* & *acoustic*, respectively.

4.3 Experiments

We evaluate our proposed approach for CMU-MOSI (test data) & CMU-MOSEI (dev data) [4]. Accuracy score is used as the evaluation metric.

We use Bi-directional GRUs having 300 neurons, each followed by a dense layer consisting of 100 neurons. Utilizing the dense layer, we project the input features of all the three modalities to the same dimensions. We set *dropout=0.5 (MOSI) & 0.3 (MOSEI)* as a measure of regularization. In addition, we also use *dropout=0.4 (MOSI) & 0.3 (MOSEI)* for the Bi-GRU layers. We employ *ReLu* activation function in the dense layers, and *softmax* activation in the final classification layer. For training the network we set the batch size=32, use *Adam* optimizer with *cross-entropy* loss function and train for 50 epochs. We report the average result of 5 runs for all our experiments.

We experiment with all the valid combinations of *uni-modal* (where only one modality is taken at a time), *bi-modal* (any two modalities are taken at a time) and *tri-modal* (all three modalities are taken at a time) inputs for *text*, *visual* and *acoustic*. In multi-modal attention frameworks i.e. *MMMU-BA* & *MMUU-SA*, the attention is computed over at least two modalities, hence, these two frameworks are not-applicable (*NA*) for *uni-modal* experiments in Table 1).

For MOSEI dataset, we obtain better performance with *text*. Subsequently, we take two modalities at a time for constructing bi-modal inputs and feed it to the network. For *text-acoustic* input pairs, we obtain the highest accuracies with 79.74%, 79.60% and 79.32% for *MMMU-BA*,

---

[1] `https://github.com/A2Zadeh/CMU-MultimodalDataSDK`

[2] `https://pair-code.github.io/facets/`

[3] `https://github.com/SenticNet/contextual-sentiment-analysis`

[4] Gold annotation of CMU-MOSEI test data wasn't released at the time of paper submission.

Modality	T	V	A	CMU-MOSEI *MMMU-BA*	 *MMUU-SA*	 *MU-SA*	CMU-MOSI *MMMU-BA*	 *MMUU-SA*	 *MU-SA*
Uni-modal	✓	-	-	*NA*	*NA*	78.23	*NA*	*NA*	80.18
	-	✓	-	*NA*	*NA*	74.84	*NA*	*NA*	63.70
	-	-	✓	*NA*	*NA*	75.88	*NA*	*NA*	62.10
Bi-modal	✓	✓	-	**79.40**	79.02	79.26	**81.51**	80.85	80.45
	✓	-	✓	**79.74**	79.60	79.32	**80.58**	80.31	79.78
	-	✓	✓	**76.66**	76.46	76.43	**65.16**	64.22	63.22
Tri-modal	✓	✓	✓	**79.80**	79.76	79.63	**82.31**	79.52	80.58

Table 1: Experimental results on CMU-MOSEI and CMU-MOSI datasets. *MMMU-BA* & *MMUU-SA* frameworks require atleast two modalities to compute the attentions, hence, these two frameworks are *not-applicable (NA)* for *uni-modal* inputs. All results are average of 5 runs with different random seeds. **T:** *Text*, **V:** *Visual*, **A:** *Acoustic*. Results are reported in accuracy.

MMUU-SA and *MU-SA* frameworks, respectively. The results that we obtain from the bi-modal combinations suggest that the *text-acoustic* combination is a better choice than the others as it improves the overall performance. Finally, we experiment with tri-modal inputs and observe an improved performance of 79.80%, 79.76% and 79.63% for *MMMU-BA*, *MMUU-SA* and *MU-SA* frameworks, respectively. This improvement entails that combination of all the three modalities is a better choice. The performance improvement was also found to be statistically significant (*T-test*) than the bi-modality and uni-modality inputs. Further, we observe that the *MMMU-BA* framework reports the best accuracy of 79.80% for the MOSEI dataset, thus supporting our claim that multi-modal attention framework (i.e. *MMMU-BA*) captures more information than the self-attention frameworks (i.e. *MMUU-SA* & *MU-SA*).

4.4 Analysis of Attention Mechanism

We analyze the attention values to understand the learning behavior of the proposed architecture. To illustrate, we take an example video from the CMU-MOSI test dataset. The transcript of the utterances for this particular video are presented in Table 2. The gold sentiments are positive for all the utterances except u_3 & u_4. We found that the proposed tri-modal *MMMU-BA* model predicts the labels of all the nine instances correctly, whereas other models make at least one misclassification. For the proposed tri-modal *MMMU-BA* model, the heatmaps of the pair-wise *MMMU-BA* softmax attention weights N_1 & N_2 of *visual-text*, *acoustic-visual* & *text-acoustic* are illustrated in Figure 2a, Figure 2b & Figure 2c, respectively. N_1 & N_2 are the softmax attention weights obtained from the pairwise matching matrices M_1 & M_2. Elements of the rows of N_1 & N_2 matrices signify different weights across multiple utterances. From the attention heatmaps, it is evident that by applying different weights across contextual utterances and modalities the model is able to predict labels of all the utterances correctly. All the heatmaps justify that the model learns to incorporate multi-modal & multi-utterance information and thus is able to correctly predict the labels of all the utterances. For example, heatmap of *MMMU-BA*$_{VT}$ (Figure 2a) signifies that elements of N_1 are weighted higher than N_2, and thus the model puts more attention on the *textual* part and relatively lesser on the *visual* part (as N_1 is multiplied with T & N_2 is multiplied with V). Also it can be concluded that textual features of the first few utterances are the most helpful compared to the rest of the textual features and visual features.

The softmax attention weights of *text* (N_t), *visual* (N_v) & *acoustic* (N_a) in tri-modal *MU-SA* model are illustrated in Figure 2d, Figure 2e & Figure 2f, respectively. The attention matrices are 9*9 dimensional. This model wrongly predicts the label of the utterance u_5. On the other hand, softmax attention weights in tri-modal *MMUU-SA* model are illustrated in Figure 2g. Nine separate attention weights (N_{u_1}, N_{u_2}, .., N_{u_9}) are computed for the nine utterances. This model wrongly predicts the labels of the utterances u_4 & u_5.

We further analyze our proposed architecture (i.e. *MMMU-BA*) *with* and *without* attention. In MOSI for tri-modal inputs, the *MMMU-BA* architecture reports a reduced accuracy of 80.89% without attention framework as compared to 82.31% with attention. We observe similar performance in the MOSEI dataset, where we obtain 79.02%

	Transcript	**Gold Label**	**Predicted**		
			MMMU-BA	***MMUU-SA***	***MU-SA***
u_1	*well he plays that well so he's good villain*	Positive	Positive	Positive	Positive
u_2	*he also has some really cool guns so that its like a desert eagle but it has to barrels to it*	Positive	Positive	Positive	Positive
u_3	*it's pretty pretty scary looking*	Negative	Negative	Negative	Negative
u_4	*i wouldn't want that pointed at me*	**Negative**	**Negative**	***Positive***	**Negative**
u_5	*and i who would i mean*	**Positive**	**Positive**	***Negative***	***Negative***
u_6	*um like i said i thought the movie was great*	Positive	Positive	Positive	Positive
u_7	*the action they do have is really well done*	Positive	Positive	Positive	Positive
u_8	*um they did a good job with car*	Positive	Positive	Positive	Positive
u_9	*they did a good job with fight scenes*	Positive	Positive	Positive	Positive

Table 2: Transcript, gold labels and predicted labels of a video in CMU-MOSI dataset having nine utterances. 7 utterances are labeled positive whereas 2 utterances are labeled negative. Predicted labels are for our different tri-modal models. Bolded labels are misclassified by at least one model.

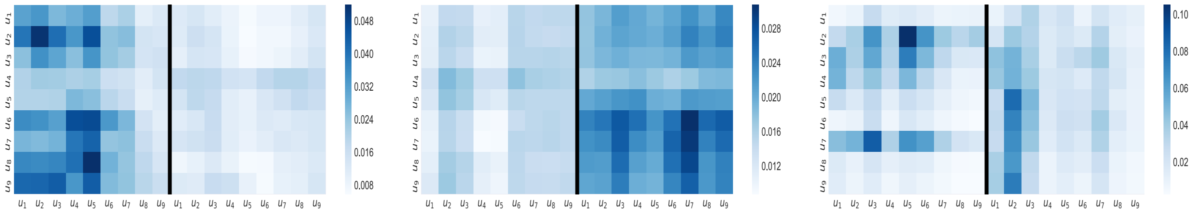

(a) *Softmax attention weights* N_1 *&* N_2 *for MMMU- BA*$_{VT}$.
(b) *Softmax attention weights* N_1 *&* N_2 *for MMMU- BA*$_{AV}$.
(c) *Softmax attention weights* N_1 *&* N_2 *for MMMU- BA*$_{TA}$.

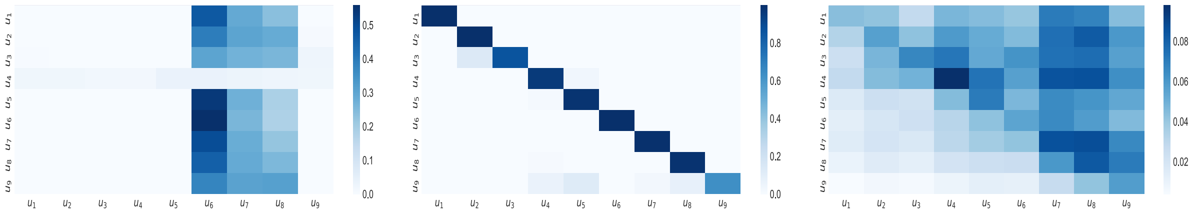

(d) *MU-SA*$_{text}$ matrix.
(e) *MU-SA*$_{visual}$ matrix.
(f) *MU-SA*$_{acoustic}$ matrix.

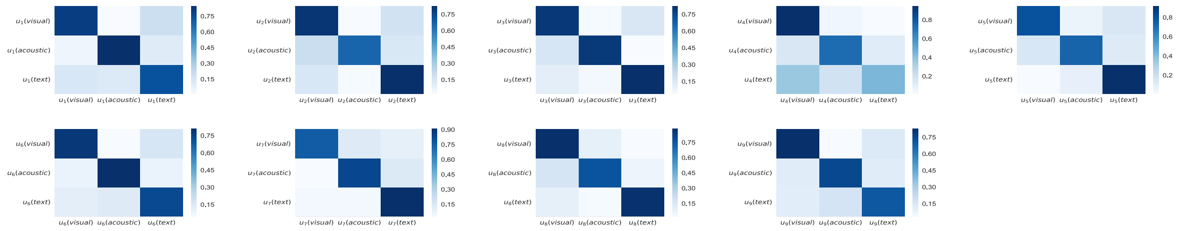

(g) Softmax attention weights (N_{u_1}, N_{u_2}, .., N_{u_9}) for *MMUU-SA* model.

Figure 2: (a), (b) & (c): Pair-wise softmax attention weights N_1 & N_2 of *visual-text*, *acoustic-visual* & *text-acoustic* in Tri-modal *MMMU-BA* model. Solid line at the center represents boundary of N_1 & N_2. The heatmaps represent attention weights of a particular utterance with respect to other utterances in N_1 & N_2. (d), (e) & (f) Softmax attention weights of *text* (N_t), *visual* (N_v) and *acoustic* (N_a) in Tri-modal MU-SA model. This model wrongly predicts the label of utterance u_5. (g) Softmax attention weights of the 9 utterances (N_{U_1}, N_{U_2}, .., N_{U_9}) in Tri-modal MMUU-SA model. This model wrongly predicts the label of utterance u_4 & u_5. The Tri-modal MMMU-BA model predicts all 9 instances correctly, whereas, the other two models makes at least one misclassification. Heatmap signifies that the model is able to predict labels of all the utterances correctly by incorporating multi-modal & multi-utterance information.

T	V	A	CMU-MOSEI		CMU-MOSI	
			w/ attention	*w/o attention*	*w/ attention*	*w/o attention*
✓	✓	-	**79.40**	78.27	**81.51**	80.71
✓	-	✓	**79.74**	78.12	**80.58**	80.18
-	✓	✓	**76.66**	76.32	**65.16**	63.69
✓	✓	✓	**79.80**	79.02	**82.31**	80.89

Table 3: Analysis of attention mechanism in *MMMU-BA* architecture. *w/ attention* → with multi-modal multi utterance attention mechanism and *w/o attention* → without attention mechanism.

accuracy without attention framework against 79.80% accuracy with attention framework. Statistical *T-test* shows these improvements to be significant. We also observed the similar trends for bi-modal inputs in both the datasets. All these experiments (c.f. Table 3) suggest that the attention framework is an important component in our proposed architecture, and in absence of this the network finds it more difficult for learning in all the cases (i.e. bi-modal & tri-modal input setups).

We successfully show that attention computation on pairwise combination of modalities (i.e. bi-modal attention framework) is more effective than the combination of self-attention on single modality. Further for the completeness of the proposed approach, we also experiment with tri-modal attention framework (attention is computed on three modalities at a time). Though the results that we obtain are convincing, it does not improve the performance over the bi-modal attention framework. We obtain the accuracies of 79.58% & 81.25% on MOSEI and MOSI, respectively, for the tri-modal attention framework.

4.5 Comparative Analysis

For MOSI datasets we compare the performance of our proposed approach with the the following state-of-the-art systems: i). Poria et al. (2017b)-LSTM-based sequence model to capture the contextual information of the utterances; ii). Poria et al. (2017c)- Tensor level fusion technique for combining all the three modalities; iii). Chen et al. (2017)-A gated multi-modal embedded LSTM with temporal attention (GME-LSTM(A)) for word-level fusion of multi-modality inputs. and iv). Zadeh et al. (2018a)- Multiple attention blocks for capturing the information across the three modalities.

In Table 4 we present the comparative performance between our proposed model and other state-of-the-art systems. In MOSI dataset, Poria et al. (2017b; 2017c) reported the accuracies of 80.3% & 81.3 %, respectively, utilizing tri-modal inputs. Zadeh et al. (2018a) obtained an accuracy of & 77.4%. Chen et al. (2017) reported accuracies of 75.7% (LSTM(A)) & 76.5% (GME-LSTM(A)) for two variants of their model. In contrast to the state-of-the-art systems, our proposed model attains an improved accuracy of 82.31% when we utilize all the three modalities, i.e. *text*, *visual* & *acoustic*. Our proposed system also obtains better performance as compared to the state-of-the-arts for bi-modal inputs.

For MOSEI dataset, we evaluate against the following systems: i) Poria et al. (2017b), ii) Zadeh et al. (2018a), and iii) Zadeh et al. (2018b), where authors proposed a memory fusion network for multi-view sequential learning. We evaluate the system of Poria et al. (2017b) on MOSEI dataset and obtain 77.64% accuracy with the tri-modal inputs. Authors in (Zadeh et al., 2018a) & (Zadeh et al., 2018b) reported the accuracy 76.0% and 76.4%, respectively, with the tri-modal inputs. In comparison, our proposed approach yields an accuracy of 79.80%. As reported in Table 4 the proposed approach also attains better performance for all the bi-modal and uni-modal input combinations when compared to Poria et al. (2017b).

As reported in Table 4, we observe that the performance achieved in our proposed approach is significantly better in comparison to the state-of-the-art systems with p-value< 0.05 (obtained using T-test). For further analysis, we also report results for *three-class classification* (positive, neutral & negative classes) problem setup for MOSEI dataset in Table 7. Note that this setup is not feasible in MOSI as labels are only positive or negative.

4.6 Error Analysis

We perform error analysis on the predictions of our proposed *MMMU-BA* model with all the three input sources. Confusion matrices for both the datasets are demonstrated in Table 5. For MOSEI dataset we observe that the precision and recall for positive class (84% *precision* & 88% *recall*; are quite encouraging. However, the same are comparatively on the lower side for the negative class (68% *precision* & 58% *recall*. In contrast, for the MOSI dataset - which is relatively balanced - we obtain quite similar performance for both the classes i.e. positive (86% *precision* & 85% *recall*) and negative (77% *precision* & 75% *recall*). Please

Modality	T	V	A	CMU-MOSEI				CMU-MOSI					
				Poria et al. (2017b)	Zadeh et al. (2018a)	Zadeh et al. (2018b)	Proposed	Poria et al. (2017b)	Poria et al. (2017c)	Chen et al. (2017) LSTM(A)	Chen et al. (2017) GME-LSTM(A)	Zadeh et al. (2018a)	Proposed
Uni-modal	✓	-	-	76.75	-	-	**78.23**	78.1	-	71.3	-	-	**80.18**
	-	✓	-	71.84	-	-	**74.84**	60.3	-	52.3	-	-	63.70
	-	-	✓	70.94	-	-	**75.88**	55.8	-	55.4	-	-	62.10
Bi-modal	✓	✓	-	77.03	-	-	**79.40**	79.3	79.9	74.3	-	-	**81.51**
	✓	-	✓	76.89	-	-	**79.74**	80.2	80.1	73.5	-	-	**80.58**
	-	✓	✓	72.74	-	-	**76.66**	62.1	62.9	-	-	-	**65.16**
Tri-modal	✓	✓	✓	77.64	76.0	76.4	**79.80**	80.3	81.3	75.7	76.5	77.4	**82.31**
T-test (p-values)				-	-	-	*0.0025*	-	-	-	-	-	*0.0006*

Table 4: Comparative analysis of the proposed approach with recent state-of-the-art systems. Significance T-test p-values < 0.05

MOSEI

102	234
1230	269
Positive	Negative

63	215
404	70
Positive	Negative

MOSI

Table 5: Confusion matrix for tri-modal *MMMU-BA*.

	Text	Actual	Predicted	Possible Reason
MOSI	*At first I thought the movie would appeal more to younger audience.*	negative	positive	Implicit sentiment.
	Its really non-stop from beginning to end.	negative	positive	
	But its action isn't particularly memorable.	negative	positive	Negation & strong word.
	I mean I don't regret seeing it.	positive	negative	
	Um I was really looking forward to it.	negative	positive	Sarcastic sentence.
MOSEI	*And when I was going to school it was really difficult for me to find avenues and resources to be able to reach higher education.*	negative	positive	Implicit sentiment.
	We could have a decision from the court on the stay any day now.	positive	negative	
	Holidays never really happen in online courses I guess.	negative	positive	Negation & strong word.
	Young people dropping out of the labour market are actually not counted anymore as unemployed as they are inactive.	positive	negative	
	Thank you for your efforts and consideration.	negative	positive	Sarcastic sentence.

Table 6: **Error Analysis:** Frequent error cases and their possible reasons of failure for the tri-modal *MMMU-BA* framework.

Metric	CMU-MOSEI	
	Poria et al. (2017b)	MMMU-BA (Tri-modal)
Accuracy	61.89	**63.30**
F1 Score	61.60	**63.07**

Table 7: Three class (positive, negative, neutral) classification results in MOSEI dataset.

refer to the appendix for PR curves of different input combinations.

We further analyze our outputs qualitatively and list a few frequently occurring error categories with examples in Table 6.

5 Conclusion

In this paper, we have proposed a recurrent neural network based multi-modal attention framework that leverages the contextual information for utterance-level sentiment prediction. The network learns on top of three modalities, *viz.* text, visual and acoustic, considering sequence of utterances in a video. Through evaluation results on two benchmark datasets (one being the popular & commonly used (MOSI) and other being the most recent & largest (MOSEI) dataset for multi-modal sentiment analysis), we successfully showed that the proposed attention based framework performs better than various state-of-the-art systems.

In future, we would like to investigate new techniques, and explore the ways to handle implicit sentiment and sarcasm. Future direction of work also include adding more dimensions, e.g. emotion analysis & intensity prediction.

6 Acknowledgment

The research reported here is partially supported by SkyMap Global India Private Limited. Asif Ekbal acknowledges the Young Faculty Research Fellowship (YFRF), supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

References

Md Shad Akhtar, Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2017. Feature selection and ensemble construction: A two-step method for aspect based sentiment analysis. *Knowledge-Based Systems*, 125:116 – 135.

Md Shad Akhtar, Ayush Kumar, Asif Ekbal, and Pushpak Bhattacharyya. 2016. A Hybrid Deep Learning Architecture for Sentiment Analysis. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016): Technical Papers, December 11-16, 2016*, pages 482–493, Osaka, Japan.

Minghai Chen, Sen Wang, Paul Pu Liang, Tadas Baltrušaitis, Amir Zadeh, and Louis-Philippe Morency. 2017. Multimodal sentiment analysis with word-level fusion and reinforcement learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 163–171. ACM.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

G. Degottex, J. Kane, T. Drugman, T. Raitio, and S. Scherer. 2014. Covarep - a collaborative voice analysis repository for speech technologies. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 960–964.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1459–1462. ACM.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th KDD*, pages 168–177, Seattle, WAs.

Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 2013. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Rada Mihalcea. 2012. Multimodal sentiment analysis. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis, WASSA@ACL 2012, July 12, 2012, Jeju Island, Republic of Korea*, page 1.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 321–327, Atlanta, Georgia, USA. Association for Computational Linguistics.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: harvesting opinions from the web. In *Proceedings of the 13th International Conference on Multimodal Interaction, ICMI 2011, Alicante, Spain, November 14-18, 2011*, pages 169–176.

Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135.

Soujanya Poria, Erik Cambria, Rajiv Bajpai, and Amir Hussain. 2017a. A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion*, 37:98–125.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017b. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 873–883.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Mazumder, Amir Zadeh, and Louis-Philippe Morency. 2017c. Multi-level multiple attentions for contextual multimodal sentiment analysis. In *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 1033–1038. IEEE.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 439–448. IEEE.

Soujanya Poria, Haiyun Peng, Amir Hussain, Newton Howard, and Erik Cambria. 2017d. Ensemble application of convolutional neural networks and multiple kernel learning for multimodal sentiment analysis. *Neurocomputing*, 261:217–230.

P. D. Turney. 2002. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Association for Computational Linguistics (ACL)*, pages 417–424, Philadelphia, USA.

A Zadeh, PP Liang, S Poria, P Vij, E Cambria, and LP Morency. 2018a. Multi-attention recurrent network for human communication comprehension. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-2018)*, pages 5642 – 5649, New Orleans, USA.

A. Zadeh, R. Zellers, E. Pincus, and L. P. Morency. 2016. Multimodal Sentiment Intensity Analysis in Videos: Facial Gestures and Verbal Messages. *IEEE Intelligent Systems*, 31(6):82–88.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1114, Copenhagen, Denmark. Association for Computational Linguistics.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018b. Memory Fusion Network for Multi-view Sequential Learning. *arXiv preprint arXiv:1802.00927*.

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018c. Multimodal Language Analysis in the Wild: CMU-MOSEI Dataset and Interpretable Dynamic Fusion Graph. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2236–2246. Association for Computational Linguistics.

7 Appendix

7.1 Multi-Modal Uni-Utterance - Self Attention (*MMUU-SA*) Framework

Let $X_p \in \mathbb{R}^{3\times d}$ is the information matrix of the p^{th} utterance where the three *'d'* dimensional rows are the outputs of the time-distributed dense layer for the three modalities. Computation in the p^{th} attention block proceeds as follows:

$$\begin{aligned} M_{u_p} &= X_{u_p}.X_{u_p}^T \\ N_{u_p}(i,j) &= \frac{e^{M_{u_p}(i,j)}}{\sum_{k=1}^{3} e^{M_{u_p}(i,k)}} \quad \text{for} \quad i,j = 1,2,3; \\ O_{u_p} &= N_{u_p}.X_{u_p} \\ A_{u_p} &= O_{u_p} \odot X_{u_p} \end{aligned}$$

The attention matrix $A_{u_p} \in \mathbb{R}^{3\times d}$ is computed separately for, p = 1^{st}, 2^{nd}, ... q^{th} utterances. Finally, for each utterance p, A_{u_p} and X_{u_p} are concatenated and passed to the output layer for classification.

7.2 Multi-Utterance - Self Attention (*MU-SA*) Framework

In *MU-SA* framework, we apply self attention on the utterances of each modality separately, and use these for classification. In contrast to *MMUU-SA* framework, *MU-SA* utilizes the contextual information of the utterances at the attention level. Let, $T \in \mathbb{R}^{u\times d}$ (*text*), $V \in \mathbb{R}^{u\times d}$ (*visual*) and $A \in \mathbb{R}^{u\times d}$ (*acoustic*) are the outputs of the dense layers. For the three modalities, three separate attention blocks are required, where each block takes multi-utterance information of a single modality and computes the self attention matrix. Specifically, the *MU-SA* attention (A_v) on V (visual) will be computed as follows,

$$\begin{aligned} M_v &= V.V^T \\ N_v(i,j) &= \frac{e^{M_v(i,j)}}{\sum_{k=1}^{u} e^{M_v(i,k)}} \quad \text{for} \quad i,j = 1,..,u \\ O_v &= N_v.V \\ A_v &= O_v \odot V \end{aligned}$$

The attention matrix $A_p \in \mathbb{R}^{3\times d}$ is computed for $p = 1^{st}, 2^{nd}, ...u^{th}$ utterances. Finally, for each utterance u, A_p and X_p are concatenated and passed to the output layer with softmax activation for classification.

7.3 Dataset Statistics

Dataset statistics are presented in Table 8.

7.4 Attention Computation

MMMU-BA$_{VT}$ attention computation is illustrated in Figure 3.

Statistics	**CMU-MOSI**			**CMU-MOSEI**		
	Tr	Dv	Ts	Tr	Dv	Ts
#Videos	52	10	31	2250	300	679
#Utterance	1151	296	752	16216	1835	4625
#Utterance/Video - Min	9	9	10	1	1	1
#Utterance/Video - Max	63	34	43	98	37	52
#Utterance/Video - Avg	24.692	22.9	22.129	7.207	6.116	6.821
#Positive	556	153	467	11498	1332	3281
#Negative	595	143	285	4718	503	1344
#Words/Utter. - Min	1	1	1	1	1	1
#Words/Utter. - Max	99	44	108	515	224	549
#Words/Utter. - Avg	11.533	10.786	13.176	18.227	18.498	18.658
#Utter-Len/Video - Min	0.219s	0.648s	0.229s	0.089s	0.22s	0.15s
#Utter-Len/Video - Max	38.233s	13.599s	31.957s	208.27s	90.42s	188.22s
#Utter-Len/Video - Avg	3.635s	3.538s	4.536s	6.896s	6.960s	7.158s
#Speakers	89			1000		

(a) Data Statistics. ***Tr***→Train set; ***Dv***→Development set; ***Ts***→Test set;

Table 8: Dataset statistics for MOSI (Zadeh et al., 2016) and MOSEI (Zadeh et al., 2018c).

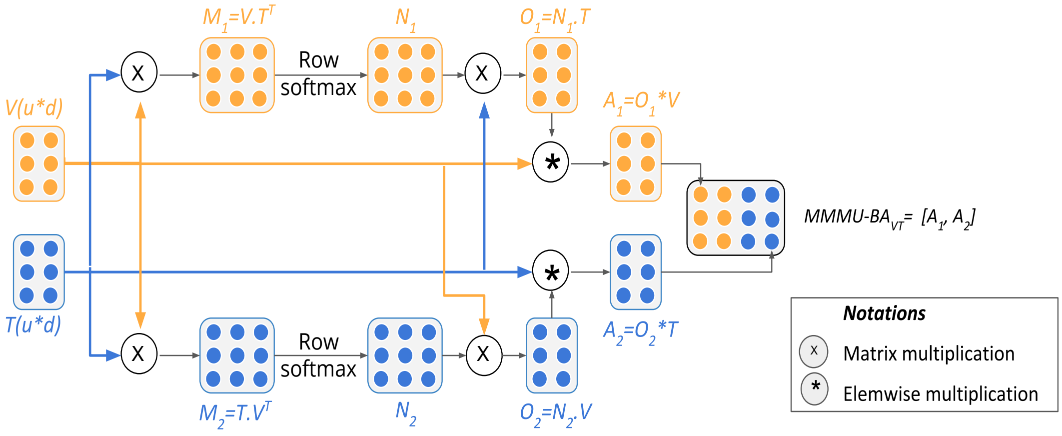

Figure 3: *MMMU-BA*$_{VT}$ attention computation.

7.5 Precision-Recall (PR) curve

We illustrate the precision, recall & f-measure for different input combinations in Figure 4 & Figure 5.

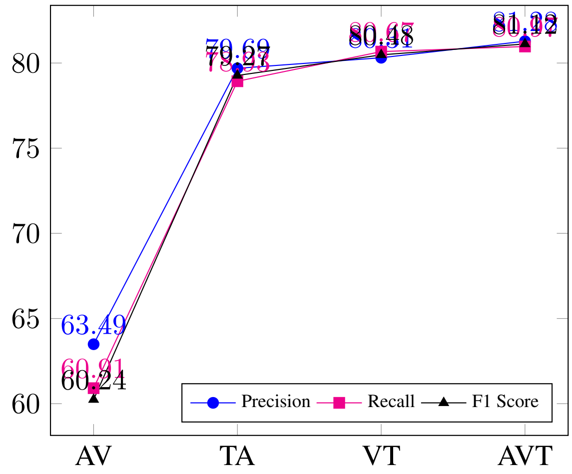

Figure 4: *Precision*, *Recall* & *F-measure* for different input combinations in *MMMU-BA* architecture of MOSI dataset.

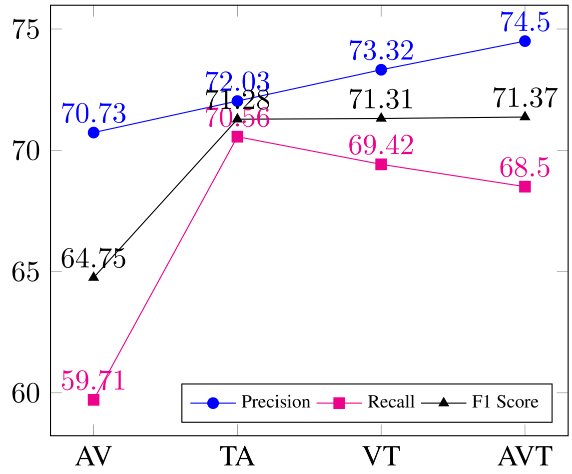

Figure 5: *Precision*, *Recall* & *F-measure* for different input combinations in *MMMU-BA* architecture of MOSEI dataset.

Adaptive Semi-supervised Learning for Cross-domain Sentiment Classification

Ruidan He[†‡], **Wee Sun Lee**[†], **Hwee Tou Ng**[†], and **Daniel Dahlmeier**[‡]
[†]Department of Computer Science, National University of Singapore
[‡]SAP Innovation Center Singapore
[†]`{ruidanhe,leews,nght}@comp.nus.edu.sg`
[‡]`d.dahlmeier@sap.com`

Abstract

We consider the cross-domain sentiment classification problem, where a sentiment classifier is to be learned from a source domain and to be generalized to a target domain. Our approach explicitly minimizes the distance between the source and the target instances in an embedded feature space. With the difference between source and target minimized, we then exploit additional information from the target domain by consolidating the idea of semi-supervised learning, for which, we jointly employ two regularizations – entropy minimization and self-ensemble bootstrapping – to incorporate the unlabeled target data for classifier refinement. Our experimental results demonstrate that the proposed approach can better leverage unlabeled data from the target domain and achieve substantial improvements over baseline methods in various experimental settings.

1 Introduction

In practice, it is often difficult and costly to annotate sufficient training data for diverse application domains on-the-fly. We may have sufficient labeled data in an existing domain (called the source domain), but very few or no labeled data in a new domain (called the target domain). This issue has motivated research on cross-domain sentiment classification, where knowledge in the source domain is transferred to the target domain in order to alleviate the required labeling effort.

One key challenge of domain adaptation is that data in the source and target domains are drawn from different distributions. Thus, adaptation performance will decline with an increase in distribution difference. Specifically, in sentiment analysis, reviews of different products have different vocabulary. For instance, restaurants reviews would contain opinion words such as “tender”, “tasty”, or “undercooked” and movie reviews would contain “thrilling”, “horrific”, or “hilarious”. The intersection between these two sets of opinion words could be small which makes domain adaptation difficult.

Several techniques have been proposed for addressing the problem of domain shifting. The aim is to bridge the source and target domains by learning domain-invariant feature representations so that a classifier trained on a source domain can be adapted to another target domain. In cross-domain sentiment classification, many works (Blitzer et al., 2007; Pan et al., 2010; Zhou et al., 2015; Wu and Huang, 2016; Yu and Jiang, 2016) utilize a key intuition that domain-specific features could be aligned with the help of domain-invariant features (pivot features). For instance, “hilarious” and “tasty” could be aligned as both of them are relevant to “good”.

Despite their promising results, these works share two major limitations. First, they highly depend on the heuristic selection of pivot features, which may be sensitive to different applications. Thus the learned new representations may not effectively reduce the domain difference. Furthermore, these works only utilize the unlabeled target data for representation learning while the sentiment classifier was solely trained on the source domain. There have not been many studies on exploiting unlabeled target data for refining the classifier, even though it may contain beneficial information. How to effectively leverage unlabeled target data still remains an important challenge for domain adaptation.

In this work, we argue that the information from unlabeled target data is beneficial for domain adaptation and we propose a novel **D**omain **A**daptive **S**emi-supervised learning framework (DAS) to better exploit it. Our main intuition is to treat the problem as a semi-supervised learning task by considering target instances as unla-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3467–3476
Brussels, Belgium, October 31 - November 4, 2018.

beled data, assuming the domain distance can be effectively reduced through domain-invariant representation learning. Specifically, the proposed approach jointly performs feature adaptation and semi-supervised learning in a multi-task learning setting. For feature adaptation, it explicitly minimizes the distance between the encoded representations of the two domains. On this basis, two semi-supervised regularizations – entropy minimization and self-ensemble bootstrapping – are jointly employed to exploit unlabeled target data for classifier refinement.

We evaluate our method rigorously under multiple experimental settings by taking label distribution and corpus size into consideration. The results show that our model is able to obtain significant improvements over strong baselines. We also demonstrate through a series of analysis that the proposed method benefits greatly from incorporating unlabeled target data via semi-supervised learning, which is consistent with our motivation. Our datasets and source code can be obtained from `https://github.com/ruidan/DAS`.

2 Related Work

Domain Adaptation: The majority of feature adaptation methods for sentiment analysis rely on a key intuition that even though certain opinion words are completely distinct for each domain, they can be aligned if they have high correlation with some domain-invariant opinion words (pivot words) such as "excellent" or "terrible". Blitzer et al. (2007) proposed a method based on structural correspondence learning (SCL), which uses pivot feature prediction to induce a projected feature space that works well for both the source and the target domains. The pivot words are selected in a way to cover common domain-invariant opinion words. Subsequent research aims to better align the domain-specific words (Pan et al., 2010; He et al., 2011; Wu and Huang, 2016) such that the domain discrepancy could be reduced. More recently, Yu and Jiang (2016) borrow the idea of pivot feature prediction from SCL and extend it to a neural network-based solution with auxiliary tasks. In their experiment, substantial improvement over SCL has been observed due to the use of real-valued word embeddings. Unsupervised representation learning with deep neural networks (DNN) such as denoising autoencoders has also been explored for feature adaptation (Glorot et al., 2011; Chen et al., 2012; Yang and Eisenstein, 2014). It has been shown that DNNs could learn transferable representations that disentangle the underlying factors of variation behind data samples.

Although the aforementioned methods aim to reduce the domain discrepancy, they do not explicitly minimize the distance between distributions, and some of them highly rely on the selection of pivot features. In our method, we formally construct an objective for this purpose. Similar ideas have been explored in many computer vision problems, where the representations of the underlying domains are encouraged to be similar through explicit objectives (Tzeng et al., 2014; Ganin and Lempitsky, 2015; Long et al., 2015; Zhuang et al., 2015; Long et al., 2017) such as maximum mean discrepancy (MMD) (Gretton et al., 2012). In NLP tasks, Li et al. (2017) and Chen et al. (2017) both proposed using adversarial training framework for reducing domain difference. In their model, a sub-network is added as a domain discriminator while deep features are learned to confuse the discriminator. The feature adaptation component in our model shares similar intuition with MMD and adversary training. We will show a detailed comparison with them in our experiments.

Semi-supervised Learning: We attempt to treat domain adaptation as a semi-supervised learning task by considering the target instances as unlabeled data. Some efforts have been initiated on transfer learning from unlabeled data (Dai et al., 2007; Jiang and Zhai, 2007; Wu et al., 2009). In our model, we reduce the domain discrepancy by feature adaptation, and thereafter adopt semi-supervised learning techniques to learn from unlabeled data. Primarily motivated by (Grandvalet and Bengio, 2004) and (Laine and Aila, 2017), we employed entropy minimization and self-ensemble bootstrapping as regularizations to incorporate unlabeled data. Our experimental results show that both methods are effective when jointly trained with the feature adaptation objective, which confirms to our motivation.

3 Model Description

3.1 Notations and Model Overview

We conduct most of our experiments under an unsupervised domain adaptation setting, where we have no labeled data from the target domain. Consider two sets D_s and D_t. $D_s = \{\mathbf{x}_i^{(s)}, \mathbf{y}_i^{(s)}\}|_{i=1}^{n_s}$ is from the source domain with n_s labeled examples, where $\mathbf{y}_i \in \mathbb{R}^C$ is a one-hot vector representation of sentiment label and C denotes the number of classes. $D_t = \{\mathbf{x}_i^{(t)}\}|_{i=1}^{n_t}$ is from the target domain with n_t unlabeled examples. $N = n_s + n_t$ denotes the total number of training documents including both labeled and unlabeled[1]. We aim to learn a sentiment classifier from D_s and D_t such that the classifier would work well on the target domain. We also present some results under a setting where we assume that a small number of labeled target examples are available (see Figure 3).

For the proposed model, we denote G parameterized by θ_g as a neural-based feature encoder that maps documents from both domains to a shared feature space, and $\mathcal{F}$ parameterized by θ_f as a fully connected layer with softmax activation serving as the sentiment classifier. We aim to learn feature representations that are domain-invariant and at the same time discriminative on both domains, thus we simultaneously consider three factors in our objective: (1) minimize the classification error on the labeled source examples; (2) minimize the domain discrepancy; and (3) leverage unlabeled data via semi-supervised learning.

Suppose we already have the encoded features of documents $\{\boldsymbol{\xi}_i^{(s,t)} = G(\mathbf{x}_i^{(s,t)}; \theta_g)\}|_{i=1}^{N}$ (see Section 4.1), the objective function for purpose (1) is thus the cross entropy loss on the labeled source examples

$$L = -\frac{1}{n_s} \sum_{i=1}^{n_s} \sum_{j=1}^{C} \mathbf{y}_i^{(s)}(j) \log \tilde{\mathbf{y}}_i^{(s)}(j) \qquad (1)$$

where $\tilde{\mathbf{y}}_i^{(s)} = \mathcal{F}(\boldsymbol{\xi}_i^{(s)}; \theta_f)$ denotes the predicted label distribution. In the following subsections, we will explain how to perform feature adaptation and domain adaptive semi-supervised learning in details for purpose (2) and (3) respectively.

3.2 Feature Adaptation

Unlike prior works (Blitzer et al., 2007; Yu and Jiang, 2016), our method does not attempt to align domain-specific words through pivot words. In our preliminary experiments, we found that word embeddings pre-trained on a large corpus are able to adequately capture this information. As we will later show in our experiments, even without adaptation, a naive neural network classifier with pre-trained word embeddings can already achieve reasonably good results.

We attempt to explicitly minimize the distance between the source and target feature representations ($\{\boldsymbol{\xi}_i^{(s)}\}|_{i=1}^{n_s}$ and $\{\boldsymbol{\xi}_i^{(t)}\}_{i=1}^{n_t}$). A few methods from literature can be applied such as Maximum Mean Discrepancy (MMD) (Gretton et al., 2012) or adversary training (Li et al., 2017; Chen et al., 2017). The main idea of MMD is to estimate the distance between two distributions as the distance between sample means of the projected embeddings in Hilbert space. MMD is implicitly computed through a characteristic kernel, which is used to ensure that the sample mean is injective, leading to the MMD being zero if and only if the distributions are identical. In our implementation, we skip the mapping procedure induced by a characteristic kernel for simplifying the computation and learning. We simply estimate the distribution distance as the distance between the sample means in the current embedding space. Although this approximation cannot preserve all statistical features of the underlying distributions, we find it performs comparably to MMD on our problem. The following equations formally describe the feature adaptation loss $\mathcal{J}$:

$$\mathcal{J} = KL(\mathbf{g}_s||\mathbf{g}_t) + KL(\mathbf{g}_t||\mathbf{g}_s) \qquad (2)$$

$$\mathbf{g}'_s = \frac{1}{n_s}\sum_{i=1}^{n_s} \boldsymbol{\xi}_i^{(s)}, \qquad \mathbf{g}_s = \frac{\mathbf{g}'_s}{\|\mathbf{g}'_s\|_1} \qquad (3)$$

$$\mathbf{g}'_t = \frac{1}{n_t}\sum_{i=1}^{n_t} \boldsymbol{\xi}_i^{(t)}, \qquad \mathbf{g}_t = \frac{\mathbf{g}'_t}{\|\mathbf{g}'_t\|_1} \qquad (4)$$

L_1 normalization is applied on the mean representations $\mathbf{g}'_s$ and $\mathbf{g}'_t$, rescaling the vectors such that all entries sum to 1. We adopt a symmetric version of KL divergence (Zhuang et al., 2015) as the distance function. Given two distribution vectors $P, Q \in \mathbb{R}^k$, $KL(P||Q) = \sum_{i=1}^{k} P(i) \log(\frac{P(i)}{Q(i)})$.

3.3 Domain Adaptive Semi-supervised Learning (DAS)

We attempt to exploit the information in target data through semi-supervised learning objectives, which are jointly trained with L and $\mathcal{J}$. Normally, to incorporate target data, we can minimize the cross entropy loss between the true label distributions $\mathbf{y}_i^{(t)}$ and the predicted label distributions

---

[1] Note that unlabeled source examples can also be included for training. In that case, $N = n_s + n_t + n_{s'}$ where $n_{s'}$ denotes the number of unlabeled source examples. This corresponds to our experimental setting 2. For simplicity, we only consider n_s and n_t in our description.

$\tilde{\mathbf{y}}_i^{(t)} = \mathcal{F}(\boldsymbol{\xi}_i^{(t)}; \theta_f)$ over target samples. The challenge here is that $\mathbf{y}_i^{(t)}$ is unknown, and thus we attempt to estimate it via semi-supervised learning. We use entropy minimization and bootstrapping for this purpose. We will later show in our experiments that both methods are effective, and jointly employing them overall yields the best results.

Entropy Minimization: In this method, $\mathbf{y}_i^{(t)}$ is estimated as the predicted label distribution $\tilde{\mathbf{y}}_i^{(t)}$, which is a function of θ_g and θ_f. The loss can thus be written as

$$\Gamma = -\frac{1}{n_t} \sum_{i=1}^{n_t} \sum_{j=1}^{C} \tilde{\mathbf{y}}_i^{(t)}(j) \log \tilde{\mathbf{y}}_i^{(t)}(j) \qquad (5)$$

Assume the domain discrepancy can be effectively reduced through feature adaptation, by minimizing the entropy penalty, training of the classifier is influenced by the unlabeled target data and will generally maximize the margins between the target examples and the decision boundaries, increasing the prediction confidence on the target domain.

Self-ensemble Bootstrapping: Another way to estimate $\mathbf{y}_i^{(t)}$ corresponds to bootstrapping. The idea is to estimate the unknown labels as the predictions of the model learned from the previous round of training. Bootstrapping has been explored for domain adaptation in previous works (Jiang and Zhai, 2007; Wu et al., 2009). However, in their methods, domain discrepancy was not explicitly minimized via feature adaptation. Applying bootstrapping or other semi-supervised learning techniques in this case may worsen the results as the classifier can perform quite bad on the target data.

Inspired by the ensembling method proposed in (Laine and Aila, 2017), we estimate $\mathbf{y}_i^{(t)}$ by forming ensemble predictions of labels during training, using the outputs on different training epochs. The loss is formulated as follows:

$$\Omega = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} \tilde{\mathbf{z}}_i^{(s,t)}(j) \log \tilde{\mathbf{y}}_i^{(s,t)}(j) \qquad (6)$$

where $\tilde{\mathbf{z}}$ denotes the estimated labels computed on the ensemble predictions from different epochs. The loss is applied on all documents. It serves for bootstrapping on the unlabeled target data, and it also serves as a regularization that encourages the network predictions to be consistent in different training epochs. Ω is jointly trained with L, $\mathcal{J}$, and Γ. Algorithm 1 illustrates the overall training process of the proposed domain adaptive semi-supervised learning (DAS) framework.

Algorithm 1 Pseudocode for training DAS

```
Require: $D_s$, $D_t$, $G$, $\mathcal{F}$
Require: $\alpha$ = ensembling momentum, $0 \leq \alpha < 1$
Require: $w(t)$ = weight ramp-up function
  $\mathbf{Z} \leftarrow 0_{[N \times C]}$
  $\tilde{\mathbf{z}} \leftarrow 0_{[N \times C]}$
  for $t \in [1, \textit{max-epochs}]$ do
    for each minibatch $B^{(s)}$, $B^{(t)}$, $B^{(u)}$ in
            $D_s$, $D_t$, $\{x_i^{(s,t)}\}|_{i=1}^{N}$ do
      compute loss $L$ on $[\mathbf{x}_{i \in B^{(s)}}, \mathbf{y}_{i \in B^{(s)}}]$
      compute loss $\mathcal{J}$ on $[\mathbf{x}_{i \in B^{(s)}}, \mathbf{x}_{j \in B^{(t)}}]$
      compute loss $\Gamma$ on $\mathbf{x}_{i \in B^{(t)}}$
      compute loss $\Omega$ on $[\mathbf{x}_{i \in B^{(u)}}, \tilde{\mathbf{z}}_{i \in B^{(u)}}]$
      $\textit{overall-loss} \leftarrow L + \lambda_1 \mathcal{J} + \lambda_2 \Gamma + w(t)\Omega$
      update network parameters
    end for
    $\mathbf{Z}'_i \leftarrow \mathcal{F}(G(\mathbf{x}_i))$, for $i \in N$
    $\mathbf{Z} \leftarrow \alpha \mathbf{Z} + (1 - \alpha)\mathbf{Z}'$
    $\tilde{\mathbf{z}} \leftarrow \textit{one-hot-vectors}(\mathbf{Z})$
  end for
```

In Algorithm 1, λ_1, λ_2, and $w(t)$ are weights to balance the effects of $\mathcal{J}$, Γ, and Ω respectively. λ_1 and λ_2 are constant hyper-parameters. We set $w(t) = \exp[-5(1 - \frac{t}{\textit{max-epochs}})^2]\lambda_3$ as a Gaussian curve to ramp up the weight from 0 to λ_3. This is to ensure the ramp-up of the bootstrapping loss component is slow enough in the beginning of the training. After each training epoch, we compute $\mathbf{Z}'_i$ which denotes the predictions made by the network in current epoch, and then the ensemble prediction $\mathbf{Z}_i$ is updated as a weighted average of the outputs from previous epochs and the current epoch, with recent epochs having larger weight. For generating estimated labels $\tilde{\mathbf{z}}_i$, $\mathbf{Z}_i$ is converted to a one-hot vector where the entry with the maximum value is set to one and other entries are set to zeros. The self-ensemble bootstrapping is a generalized version of bootstrappings that only use the outputs from the previous round of training (Jiang and Zhai, 2007; Wu et al., 2009). The ensemble prediction is likely to be closer to the correct, unknown labels of the target data.

Domain		#Pos	#Neg	#Neu	Total
Book	Set 1	2000	2000	2000	6000
	Set 2	4824	513	663	6000
Electronics	Set 1	2000	2000	2000	6000
	Set 2	4817	694	489	6000
Beauty	Set 1	2000	2000	2000	6000
	Set 2	4709	616	675	6000
Music	Set 1	2000	2000	2000	6000
	Set 2	4441	785	774	6000

(a) Small-scale datasets

Domain	#Pos	#Neg	#Neu	Total
IMDB	55,242	11,735	17,942	84,919
Yelp	155,625	29,597	45,941	231,163
Cell Phone	148,657	24,343	21,439	194,439
Baby	126,525	17,012	17,255	160,792

(b) Large-scale datasets

Table 1: Summary of datasets.

4 Experiments

4.1 CNN Encoder Implementation

We have left the feature encoder G unspecified, for which, a few options can be considered. In our implementation, we adopt a one-layer CNN structure from previous works (Kim, 2014; Yu and Jiang, 2016), as it has been demonstrated to work well for sentiment classification tasks. Given a review document $\mathbf{x} = (x_1, x_2, ..., x_n)$ consisting of n words, we begin by associating each word with a continuous word embedding (Mikolov et al., 2013) $\mathbf{e}_x$ from an embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times d}$, where V is the vocabulary size and d is the embedding dimension. E is jointly updated with other network parameters during training. Given a window of dense word embeddings $\mathbf{e}_{x_1}, \mathbf{e}_{x_2}, ..., \mathbf{e}_{x_l}$, the convolution layer first concatenates these vectors to form a vector $\hat{\mathbf{x}}$ of length ld and then the output vector is computed by Equation (7):

$$Conv(\hat{\mathbf{x}}) = f(\mathbf{W} \cdot \hat{\mathbf{x}} + \mathbf{b}) \tag{7}$$

$\theta_g = \{\mathbf{W}, \mathbf{b}\}$ is the parameter set of the encoder G and is shared across all windows of the sequence. f is an element-wise non-linear activation function. The convolution operation can capture local contextual dependencies of the input sequence and the extracted feature vectors are similar to n-grams. After the convolution operation is applied to the whole sequence, we obtain a list of hidden vectors $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n)$. A max-over-time pooling layer is applied to obtain the final vector representation $\boldsymbol{\xi}$ of the input document.

4.2 Datasets and Experimental Settings

Existing benchmark datasets such as the Amazon benchmark (Blitzer et al., 2007) typically remove reviews with neutral labels in both domains. This is problematic as the label information of the target domain is not accessible in an unsupervised domain adaptation setting. Furthermore, removing neutral instances may bias the dataset favorably for max-margin-based algorithms like ours, since the resulting dataset has all uncertain labels removed, leaving only high confidence examples. Therefore, we construct new datasets by ourselves. The results on the original Amazon benchmark is qualitatively similar, and we present them in Appendix A for completeness since most of previous works reported results on it.

Small-scale datasets: Our new dataset was derived from the large-scale Amazon datasets[2] released by McAuley et al. (2015). It contains four domains[3]: Book (BK), Electronics (E), Beauty (BT), and Music (M). Each domain contains two datasets. Set 1 contains 6000 instances with exactly balanced class labels, and set 2 contains 6000 instances that are randomly sampled from the large dataset, preserving the original label distribution, which we believe better reflects the label distribution in real life. The examples in these two sets do not overlap. Detailed statistics of the generated datasets are given in Table 1a.

In all our experiments on the small-scale datasets, we use set 1 of the source domain as the only source with sentiment label information during training, and we evaluate the trained model on set 1 of the target domain. Since we cannot control the label distribution of unlabeled data during training, we consider two different settings:

Setting (1): Only set 1 of the target domain is used as the unlabeled set. This tells us how the method performs in a condition when the target domain has a close-to-balanced label distribution. As we also evaluate on set 1 of the target domain, this is also considered as a transductive setting.

Setting (2): Set 2 from both the source and target domains are used as unlabeled sets. Since set 2 is directly sampled from millions of reviews, it better reflects real-life sentiment distribution.

Large-scale datasets: We further conduct experiments on four much larger datasets: IMDB[4]

---

[2] http://jmcauley.ucsd.edu/data/amazon/

[3] The original reviews were rated on a 5-point scale. We label them with rating < 3, > 3, and $= 3$ as negative, positive, and neutral respectively.

[4] IMDB is rated on a 10-point scale, and we label reviews with rating < 5, > 6, and $= 5/6$ as negative, positive, and neutral respectively.

(I), Yelp2014 (Y), Cell Phone (C), and Baby (B). IMDB and Yelp2014 were previously used in (Tang et al., 2015; Yang et al., 2017). Cell phone and Baby are from the large-scale Amazon dataset (McAuley et al., 2015; He and McAuley, 2016). Detailed statistics are summarized in Table 1b. We keep all reviews in the original datasets and consider a transductive setting where all target examples are used for both training (without label information) and evaluation. We perform sampling to balance the classes of labeled source data in each minibatch $B^{(s)}$ during training.

4.3 Selection of Development Set

Ideally, the development set should be drawn from the same distribution as the test set. However, under the unsupervised domain adaptation setting, we do not have any labeled target data at training phase which could be used as development set. In all of our experiments, for each pair of domains, we instead sample 1000 examples from the training set of the source domain as development set. We train the network for a fixed number of epochs, and the model with the minimum classification error on this development set is saved for evaluation. This approach works well on most of the problems since the target domain is supposed to behave like the source domain if the domain difference is effectively reduced.

Another problem is how to select the values for hyper-parameters. If we tune λ_1 and λ_2 directly on the development set from the source domain, most likely both of them will be set to 0, as unlabeled target data is not helpful for improving in-domain accuracy of the source domain. Other neural network models also have the same problem for hyper-parameter tuning. Therefore, our strategy is to use the development set from the target domain to optimize λ_1 and λ_2 for one problem (e.g., we only do this on E→BK), and fix their values on the other problems. This setting assumes that we have at least two labeled domains such that we can optimize the hyper-parameters, and then we fix them for other new unlabeled domains to transfer to.

4.4 Training Details and Hyper-parameters

We initialize word embeddings using the 300-dimension GloVe vectors supplied by Pennington et al., (2014), which were trained on 840 billion tokens from the Common Crawl. For each pair of domains, the vocabulary consists of the top 10000 most frequent words. For words in the vocabulary but not present in the pre-trained embeddings, we randomly initialize them.

We set hyper-parameters of the CNN encoder following previous works (Kim, 2014; Yu and Jiang, 2016) without specific tuning on our datasets. The window size is set to 3 and the size of the hidden layer is set to 300. The nonlinear activation function is Relu. For regularization, we also follow their settings and employ dropout with probability set to 0.5 on $\boldsymbol{\xi}_i$ before feeding it to the output layer $\mathcal{F}$, and constrain the l_2-norm of the weight vector θ_f, setting its max norm to 3.

On the small-scale datasets and the Aamzon benchmark, λ_1 and λ_2 are set to 200 and 1, respectively, tuned on the development set of task E→BK under setting 1. On the large-scale datasets, λ_1 and λ_2 are set to 500 and 0.2, respectively, tuned on I→Y. We use a Gaussian curve $w(t) = \exp[-5(1 - \frac{t}{t_{max}})^2]\lambda_3$ to ramp up the weight of the bootstrapping loss Ω from 0 to λ_3, where t_{max} denotes the maximum number of training epochs. We train 30 epochs for all experiments. We set λ_3 to 3 and α to 0.5 for all experiments.

The batch size is set to 50 on the small-scale datasets and the Amazon benchmark. We increase the batch size to 250 on the large-scale datasets to reduce the number of iterations. RMSProp optimizer with learning rate set to 0.0005 is used for all experiments.

4.5 Models for Comparison

We compare with the following baselines:

(1) **Naive**: A non-domain-adaptive baseline with bag-of-words representations and SVM classifier trained on the source domain.

(2) **mSDA** (Chen et al., 2012): This is the state-of-the-art method based on discrete input features. Top 1000 bag-of-words features are kept as pivot features. We set the number of stacked layers to 3 and the corruption probability to 0.5.

(3) **NaiveNN**: This is a non-domain-adaptive CNN trained on source domain, which is a variant of our model by setting λ_1, λ_2, and λ_3 to zeros.

(4) **AuxNN** (Yu and Jiang, 2016): This is a neural model that exploits auxiliary tasks, which has achieved state-of-the-art results on cross-domain sentiment classification. The sentence encoder used in this model is the same as ours.

(5) **ADAN** (Chen et al., 2017): This method exploits adversarial training to reduce representa-

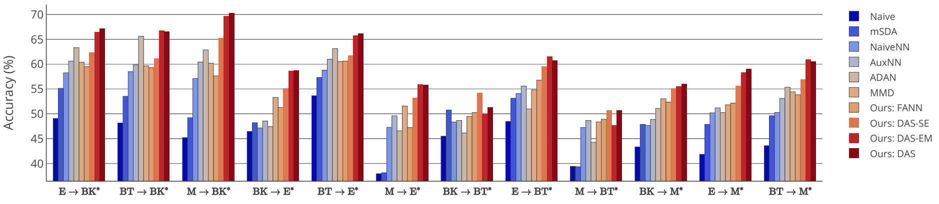

(a) Accuracy on the small-scale dataset under setting 1.

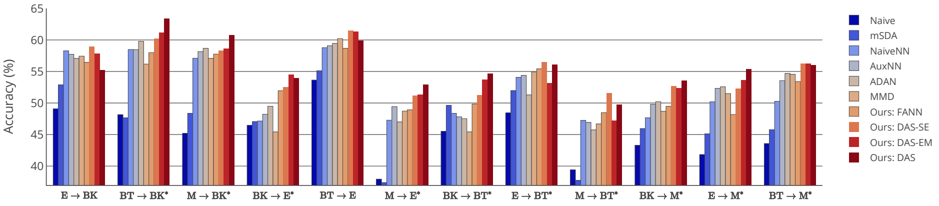

(b) Accuracy on the small-scale dataset under setting 2.

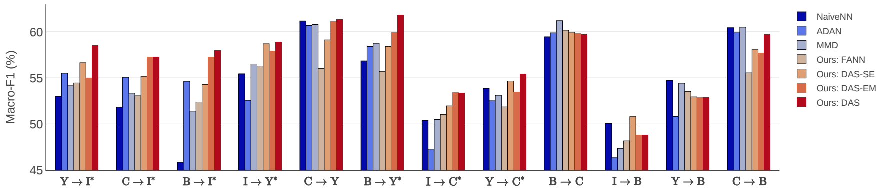

(c) Macro-F1 on the large-scale dataset.

Figure 1: Performance comparison. Average results over 5 runs with random initializations are reported for each neural method. * indicates that the proposed method (either of DAS, DAS-EM, DAS-SE) is significantly better than other baselines (baseline 1-6) with $p < 0.05$ based on one-tailed unpaired t-test.

tion difference between domains. The original paper uses a simple feedforward network as encoder. For fair comparison, we replace it with our CNN-based encoder. We train 5 iterations on the discriminator per iteration on the encoder and sentiment classifier as suggested in their paper.

(6) **MMD**: MMD has been widely used for minimizing domain discrepancy on images. In those works (Tzeng et al., 2014; Long et al., 2017), variants of deep CNNs are used for encoding images and the MMDs of multiple layers are jointly minimized. In NLP, adding more layers of CNNs may not be very helpful and thus those models from image-related tasks can not be directly applied to our problem. To compare with MMD-based method, we train a model that jointly minimize the classification loss L on the source domain and MMD between $\{\boldsymbol{\xi}_i^{(s)}|_{i=1}^{n_s}\}$ and $\{\boldsymbol{\xi}_i^{(t)}|_{i=1}^{n_t}\}$. For computing MMD, we use a Gaussian RBF which is a common choice for characteristic kernel.

In addition to the above baselines, we also show results of different variants of our model. **DAS** as shown in Algorithm 1 denotes our full model. **DAS-EM** denotes the model with only entropy minimization for semi-supervised learning (set $\lambda_3 = 0$). **DAS-SE** denotes the model with only self-ensemble bootstrapping for semi-supervised learning (set $\lambda_2 = 0$). **FANN** (feature-adaptation neural network) denotes the model without semi-supervised learning performed (set both λ_2 and λ_3 to zeros).

4.6 Main Results

Figure 1[5] shows the comparison of adaptation results (see Appendix B for the exact numerical numbers). We report classification accuracy on the small-scale dataset. For the large-scale dataset, macro-F1 is instead used since the label distribution in the test set is extremely unbalanced. Key observations are summarized as follows. (1) Both DAS-EM and DAS-SE perform better in most cases compared with ADAN, MDD, and FANN, in which only feature adaptation is performed. This demonstrates the effectiveness of the pro-

---

[5] We exclude results of Naive, mSDA and AuxNN on the large-scale dataset. Both Naive and mSDA have difficulties to scale up to the large dataset. AuxNN relies on manually selecting positive and negative pivots before training.

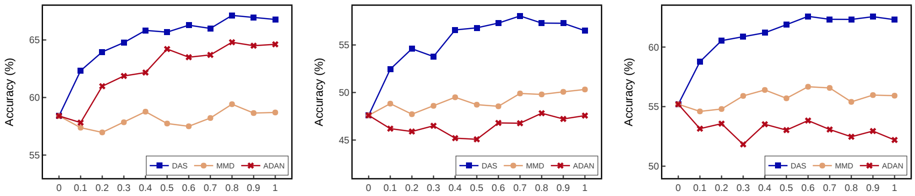

(a) Percentage of unlabeled target examples (BT->BK) (b) Percentage of unlabeled target examples (M->E) (c) Percentage of unlabeled target examples (E->BT)

Figure 2: Accuracy vs. percentage of unlabeled target training examples.

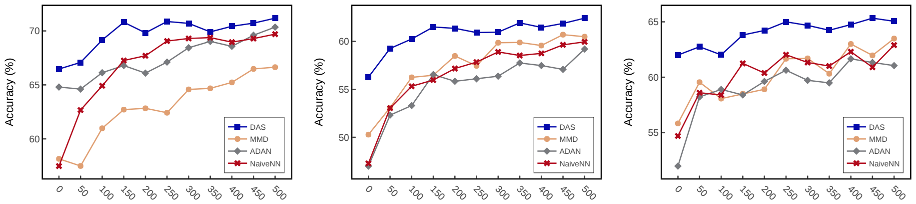

(a) # of labeled target examples (BT->BK) (b) # of labeled target examples (M->E) (c) # of labeled target examples (E->BT)

Figure 3: Accuracy vs. number of labeled target training examples.

posed domain adaptive semi-supervised learning framework. DAS-EM is more effective than DAS-SE in most cases, and the full model DAS with both techniques jointly employed overall has the best performance. (2) When comparing the two settings on the small-scale dataset, all domain-adaptive methods[6] generally perform better under setting 1. In setting 1, the target examples are balanced in classes, which can provide more diverse opinion-related features. However, when considering unsupervised domain adaptation, we should not presume the label distribution of the unlabeled data. Thus, it is necessary to conduct experiments using datasets that reflect real-life sentiment distribution as what we did on setting2 and the large-scale dataset. Unfortunately, this is ignored by most of previous works. (3) Word-embeddings are very helpful, as we can see even NaiveNN can substantially outperform mSDA on most tasks.

To see the effect of semi-supervised learning alone, we also conduct experiments by setting $\lambda_1 = 0$ to eliminate the effect of feature adaptation. Both entropy minimization and bootstrapping perform very badly in this setting. Entropy minimization gives almost random predictions with accuracy below 0.4, and the results of bootstrapping are also much lower compared to NaiveNN. This suggests that the feature adaptation component is essential. Without it, the learned target representations are less meaningful and discriminative. Applying semi-supervised learning in this case is likely to worsen the results.

4.7 Further Analysis

In Figure 2, we show the change of accuracy with respect to the percentage of unlabeled data used for training on three particular problems under setting 1. The value at $x = 0$ denotes the accuracies of NaiveNN which does not utilize any target data. For DAS, we observe a nonlinear increasing trend where the accuracy quickly improves at the beginning, and then gradually stabilizes. For other methods, this trend is less obvious, and adding more unlabeled data sometimes even worsen the results. This finding again suggests that the proposed approach can better exploit the information from unlabeled data.

We also conduct experiments under a setting with a small number of labeled target examples available. Figure 3 shows the change of accuracy with respect to the number of labeled target examples added for training. We can observe that DAS is still more effective under this setting, while the performance differences to other methods gradually decrease with the increasing number of labeled target examples.

4.8 CNN Filter Analysis

In this subsection, we aim to better understand DAS by analyzing sentiment-related CNN filters. To do that, 1) we first select a list of the most related CNN filters for predicting each sentiment label (positive, negative neutral). Those filters can be identified according to the learned weights θ_f

---

[6] Results of Naive and NaiveNN do not change under both settings as they are only trained on the source domain.

best-value-at	highly-recommend-!	nars-are-amazing	beauty-store-suggested	since-i-love
good-value-at	highly-advise-!	ulta-are-fantastic	durable-machine-and	years-i-love
perfect-product-for	gogeous-absolutely-perfect	length-are-so	perfect-length-and	bonus-i-love
great-product-at	love-love-love	expected-in-perfect	great-store-on	appearance-i-love
amazing-product-∗	highly-recommend-for	setting-works-perfect	beauty-store-for	relaxing-i-love

(a) NaiveNN

prices-my-favorite	so-nicely-!	purchase-thanks-!	feel-wonderfully-clean	are-really-cleaning
brands-my-favorite	more-affordable-price	buy-again-!	on-nicely-builds	washing-and-cleaning
very-great-stores	shampoo-a-perfect	without-hesitation-!	polish-easy-and	really-good-shampoo
great-bottle-also	an-excellent-value	buy-this-!	felt-cleanser-than	deeply-cleans-my
scent-pleasantly-floral	really-enjoy-it	discount-too-!	honestly-perfect-it	totally-moisturize-our

(b) FANN

bath-'s-wonderful	love-fruity-sweet	feeling-smooth-radiant	cleans-thoroughly-*	excellent-everyday-lotion
all-pretty-affordable	absorb-really-nicely	love-lavender-scented	loving-this-soap	affordable-cleans-nicely
it-delivers-fabulous	shower-lather-wonderfully	am-very-grateful	bed-of-love	fantastic-base-coat
and-blends-nicely	*-smells-fantastic	love-fruity-fragrances	shower-!-*	nice-gentle-scrub
heats-quickly-love	and-clean-excellent	perfect-beautiful-shimmer	radiant-daily-moisturizer	surprisingly-safe-on

(c) DAS

Table 2: Comparison of the top trigrams (each column) from the target domain (beauty) captured by the 5 most positive-sentiment-related CNN filters learned on E→BT. $*$ denotes a padding.

of the output layer $\mathcal{F}$. Higher weight indicates stronger relatedness. 2) Recall that in our implementation, each CNN filter has a window size of 3 with Relu activation. We can thus represent each selected filter as a ranked list of trigrams with highest activation values.

We analyze the CNN filters learned by NaiveNN, FANN and DAS respectively on task E→BT under setting 1. We focus on E→BT for study because electronics and beauty are very different domains and each of them has a diverse set of domain-specific sentiment expressions. For each method, we identify the top 10 most related filters for each sentiment label, and extract the top trigrams of each selected filter on both source and target domains. Since labeled source examples are used for training, we find the filters learned by the three methods capture similar expressions on the source domain, containing both domain-invariant and domain-specific trigrams. On the target domain, DAS captures more target-specific expressions compared to the other two methods. Due to space limitation, we only present a small subset of positive-sentiment-related filters in Table 2. The complete results are provided in Appendix C. From Table 2, we can observe that the filters learned by NaiveNN are almost unable to capture target-specific sentiment expressions, while FANN is able to capture limited target-specific words such as "clean" and "scent". The filters learned by DAS are more domain-adaptive, capturing diverse sentiment expressions in the target domain.

5 Conclusion

In this work, we propose DAS, a novel framework that jointly performs feature adaptation and semi-supervised learning. We have demonstrated through multiple experiments that DAS can better leverage unlabeled data, and achieve substantial improvements over baseline methods. We have also shown that feature adaptation is an essential component, without which, semi-supervised learning is not able to function properly. The proposed framework could be potentially adapted to other domain adaptation tasks, which is the focus of our future studies.

References

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In *Annual Meeting of the Association for Computational Linguistics*.

Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *The 29th International Conference on Machine Learning*.

Xilun Chen, Yu Sun, Ben Athiwarakun, Claire Cardie, and Kilian Weinberger. 2017. Adversarial deep averaging networks for cross-lingual sentiment classifier. In *Arxiv e-prints arXiv:1606.01614*.

Wenyuan Dai, Gui rong Xue, Qiang Yang, and Yong Yu. 2007. Transferring naive Bayes classifiers for text classification. In *AAAI Conference on Artificial Intelligence*.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: a deep learning approach. In *The 28th International Conference on Machine Learning*.

Yves Grandvalet and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. In *Neural Information Processing Systems*.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773.

Ruining He and Julian McAuley. 2016. Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *ACL*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Conference on Empirical Methods in Natural Language Processing*.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representation*.

Zheng Li, Yun Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *The 26th International Joint Conference on Artificial Intelligence*.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. 2017. Deep transfer learning with joint adaptation networks. In *International Conference on Machine Learning*.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *The 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *The 19th International World Wide Web Conference*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representation of users and products for document level sentiment classification. In *Annual Meeting of the Association for Computational Linguistics*.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: maximizing for domain invariance. In *Arxiv e-prints arXiv:1412.3474*.

Dan Wu, Wee Sun Lee, Nan Ye, and Hai Leong Chieu. 2009. Domain adaptive bootstrapping for named entity recognition. In *Conference on Empirical Methods in Natural Language Processing*.

Fangzhao Wu and Yongfeng Huang. 2016. Sentiment domain adaptation with multiple sources. In *Annual Meeting of the Association for Computational Linguistics*.

Wei Yang, Wei Lu, and Vincent W. Zheng. 2017. A simple regularization-based algorithm for learning cross-domain word embeddings. In *Conference on Empirical Methods in Natural Language Processing*.

Yi Yang and Jacob Eisenstein. 2014. Fast easy unsupervised domain adaptation with marginalized structured dropout. In *Annual Meeting of the Association for Computational Linguistics*.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Conference on Empirical Methods in Natural Language Processing*.

Guangyou Zhou, Tingting He, Wensheng Wu, and Xiaohua Tony Hu. 2015. Linking heterogeneous input features with pivots for domain adaptation. In *The 24th International Joint Conference on Artificial Intelligence*.

Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. 2016. Bi-transferring deep neural networks for domain adaptation. In *Annual Meeting of the Association for Computational Linguistics*.

Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. 2015. Supervised representation learning: transfer learning with deep autoencoders. In *The 24th International Joint Conference on Artificial Intelligence*.

ExtRA: Extracting Prominent Review Aspects from Customer Feedback

Zhiyi Luo, Shanshan Huang, Frank F. Xu
Bill Yuchen Lin, Hanyuan Shi, Kenny Q. Zhu
Shanghai Jiao Tong University, Shanghai, China
{jessherlock, huangss_33}@sjtu.edu.cn, frankxu2004@gmail.com
{yuchenlin, shihanyuan}@sjtu.edu.cn, kzhu@cs.sjtu.edu.cn

Abstract

Many existing systems for analyzing and summarizing customer reviews about products or service are based on a number of prominent review aspects. Conventionally, the *prominent review aspects* of a product type are determined manually. This costly approach cannot scale to large and cross-domain services such as Amazon.com, Taobao.com or Yelp.com where there are a large number of product types and new products emerge almost everyday. In this paper, we propose a novel framework, for extracting the most prominent aspects of a given product type from textual reviews. The proposed framework, ExtRA, extracts K most prominent aspect terms or phrases which do not overlap semantically automatically without supervision. Extensive experiments show that ExtRA is effective and achieves the state-of-the-art performance on a dataset consisting of different product types.

1 Introduction

Online user review is an essential part of e-commerce. Popular e-commerce websites feature an enormous amount of text reviews, especially for popular products and services. To improve the user experience and expedite the shopping process, many websites provide qualitative and quantitative analysis and summary of user reviews, which is typically organized by different *prominent review aspects*. For instance, Figure 1 shows a short review passage from a customer on TripAdvisor.com, and the customer is also asked to give scores on several specific aspects of the hotel, such as *location* and *cleanness*. With aspect-based reviews summary, potential customers can assess a product from various essential aspects very efficiently and directly. Also, aspect-based review summary offers an effective way to group products by their prominent aspects and hence enables quick comparison.

"Miami Vacation"
●●●○○ Reviewed 5 days ago

Pool is small and only 4 ft but refreshing. Hot tub also there. Staff were super friendly each day. Room was nothing special but clean and comfy. Lots of restaurants and bars nearby. Breakfast was great and despite being a busy weekend there was always a big selection available.

Stayed June 2016, traveled with family

●●○○○ Value	●●●○○ Rooms
●●●●● Location	●●●●○ Cleanliness
●●●○○ Sleep Quality	●●●●○ Service

Figure 1: An example user review about a hotel on TripAdvisor. The grades are organized by different prominent review aspects: *value*, *rooms*, etc.

Existing approaches for producing such prominent aspect terms have been largely manual work (Poria et al., 2014; Qiu et al., 2011). This is feasible for web services that only sell (or review) a small number of product types of the same domain. For example, TripAdvisor.com only features travel-related products, and Cars.com only reviews automobiles, so that human annotators can provide appropriate aspect terms for customers based on their domain knowledge. While it is true that the human knowledge is useful in characterizing a product type, such manual approach does not scale well for general-purpose e-commerce platforms, such as Amazon, eBay, or Yelp, which feature too many product types, not to mention that new product and service types are emerging everyday. In these cases, manually selecting and pre-defining aspect terms for each type is too costly and even impractical.

Moreover, the key aspects of a product type may also change over time. For example, in the past, people care more about the screen size and signal intensity when reviewing cell phones. These aspects are not so much of an issue in present days. People instead focus on battery life and processing speed, etc. Therefore, there is a growing need to automatically extract prominent aspects from user

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3477–3486
Brussels, Belgium, October 31 - November 4, 2018.

reviews.

A related but different task is *aspect-based opinion mining* (Su et al., 2008; Zeng and Li, 2013). Here techniques have been developed to automatically mine product-specific "opinion phrases" such as those shown in Figure 2. In this example, the most frequently mentioned opinion phrases about a phone model along with the mention frequency are displayed. Their goal is to get the *fine-grained* opinion summary on possibly overlapping aspects of a particular product. For example, "good looks" and "beautiful screen" both comments on the "appearance" aspect of the phone. However, these aspects are implicit and can't be used in aspect-based review summarization directly. The main disadvantage of these opinion phrases is that their aspects differ from product to product, making it difficult to compare the product side by side.

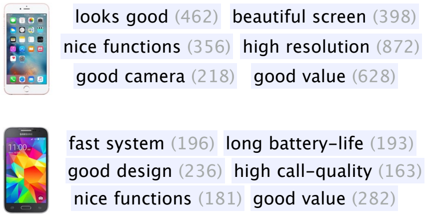

Figure 2: Automatic review summarization for two mobile phones on an e-commerce website

The goal of this paper is to develop an unsupervised framework for automatically extracting K most prominent, non-overlapping review aspects for a given type of product from user review texts. Developing such an unsupervised framework is challenging for the following reasons:

- The extracted prominent aspects not only need to cover as many customer concerns as possible but also have little semantic overlap.
- The expression of user opinions is highly versatile: aspect terms can be expressed either explicitly or implicitly. For example, the mention of "pocket" implies the aspect "size".
- Product reviews are information rich. A short piece of comments may target multiple aspects, so topics transit quickly from sentence to sentence.

Most previous unsupervised approaches for the prominent aspect extraction task are variants of topic modeling techniques (Lakkaraju et al., 2011; Lin and He, 2009; Wang et al., 2011a). The main problem of such approaches is that they typically use only word frequency and co-occurrence information, and thus degrade when extracting aspects from sentences that appear different on the surface but actually discuss similar aspects.

Given all review text about a certain product type, our framework, ExtRA, extracts most prominent aspect terms in four main steps: first it extracts potential aspect terms from text corpus by lexico-syntactic analysis; then it associates the terms to synsets in WordNet and induce a sub-graph that connect these terms together; after that it ranks the aspect terms by a personalized page rank algorithm on the sub-graph; and finally picks the top K non-overlapping terms using the subsumption relation in the subgraph.

The main contributions in this paper are as follows:

1. We propose a novel framework for extracting prominent aspects from customer review corpora (Section 2), and provide an evaluation dataset for future work in this research area.
2. Extensive experiments show that our unsupervised framework is effective and outperforms the state-of-the-art methods by a substantial margin (Section 3).

2 Framework

In this section, we first state the *review aspect extraction problem*, then present the workflow of our method, shown in Figure 3.

2.1 Problem Statement

The review aspect extraction problem is given all the text reviews about one type of product or service, extract K words (or phrases), each of which represents a prominent and distinct review aspect.

For instance, if the given product type is *hotel*, we expect a successful extraction framework to extract $K = 5$ aspect terms as follows: *room, location, staff, breakfast, pool*.

2.2 Aspect Candidates Extraction

Following the observation of Liu (2004; 2015), we assume that aspect terms are nouns and noun phrases. First, we design a set of effective syntactic rules, which can be applied across domains, to collect the aspect candidates from review texts.

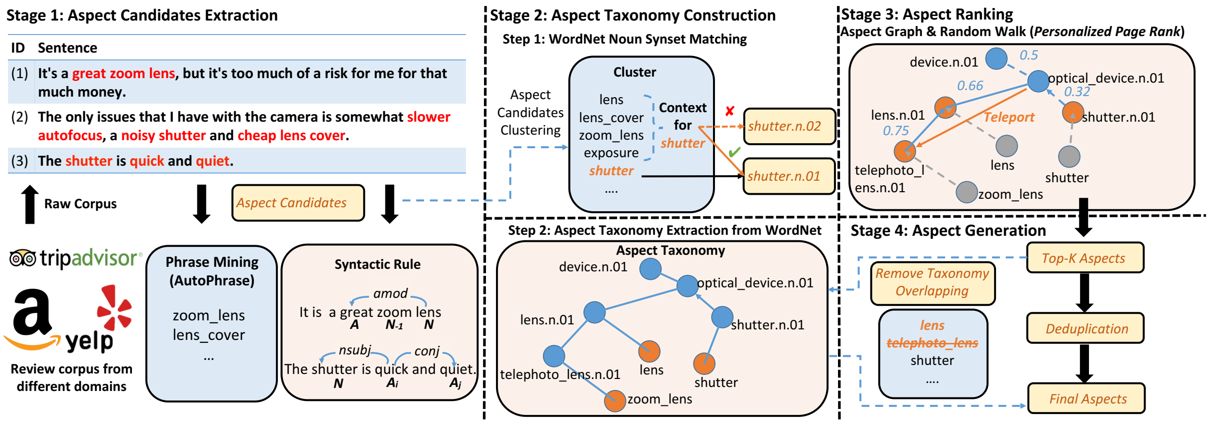

Figure 3: Overall framework.

We mainly use the adjectival modifier dependency relation (*amod*) and the nominal subject relation (*nsubj*) to extract the aspect-opinion pairs $\langle N, A\rangle$. In addition, we leverage the conjunction relation (*conj*) between adjectives to complement the extracted pairs. Formally, the extraction rules can be specified as follows:

Rule 1. If $amod(N, A)$, then extract $\langle N, A\rangle$.

Rule 2. If $nsubj(A, N)$, then extract $\langle N, A\rangle$.

Rule 3. If $\langle N, A_i\rangle$ and $conj(A_i, A_j)$, then extract $\langle N, A_j\rangle$.

In this case, N indicates a noun, and A (e.g. A_i, A_j) is an adjective. The dependencies (e.g. $amod(N, A)$) are expressed as *rel(head, dependent)*, where *rel* is the dependency relation which holds between *head* and *dependent*. Note that many aspects are expressed by phrases, thus, we extend the phrases as aspect candidates by introducing the extension rules as follows:

Rule E1. If $\langle N, A\rangle$ and $N_{-1}\_N \in P$, then use $\langle N_{-1}\_N, A\rangle$ to replace $\langle N, A\rangle$.

Rule E2. If $\langle N, A\rangle$ and $N\_N_{+1} \in P$, then use $\langle N\_N_{+1}, A\rangle$ to replace $\langle N, A\rangle$.

where N_{-1} and N_{+1} denotes the noun word, and the subscript represents displacement to N in the sentence. We use AutoPhrase (Liu et al., 2017) to extract a set of phrases P with high coherence. Then we use P to filter out the incoherent phrases so as to obtain the high-quality phrases as aspect candidates. The example in Figure 3 (Stage 1) demonstrates the extraction process. For example, we extract the pair ⟨ *great*, *zoom_lens* ⟩ from sentence (1) by applying *Rule 1* and *Rule E1*. Similarly, the extraction rules match ⟨ *slower*, *autofocus* ⟩, ⟨ *noisy*, *shutter* ⟩, ⟨ *cheap*, *lens_cover* ⟩ in sentence (2) and ⟨ *quick*, *shutter* ⟩, ⟨ *quiet*, *shutter* ⟩ in sentence (3) as potential aspect-opinion pairs. After extracting such pairs from the text reviews, we sort them by the number of occurrences, and extract the nouns and noun phrases in the top pairs as aspect candidates, assuming that the most prominent aspects are subsumed by those candidates terms.

2.3 Aspect Taxonomy Construction

The aspect candidates extracted in the last stage come with the counts modified by adjectives. We can directly use such raw counts to rank the aspect candidates. This is one of the baseline models in our experiments. However, such ranking usually suffers from the aspect overlapping problem which obviously violates the principle of pursuing both coverage and distinctiveness of prominent aspects. For example, given the number of prominent aspects K as 5, we can extract both of ‘location‘ and ‘place‘ aspects from the hotel reviews. In order to solve this problem, we construct an aspect taxonomy to obtain such overlapping information between aspect candidates by leveraging the WordNet ontology.

2.3.1 WordNet Synset Matching

First, we need to match our aspect candidates onto WordNet synsets. The accuracy of synset matching is very important for our aspect taxonomy construction. This is actually a classical word sense disambiguation (WSD) problem. Our initial attempt is to use a Python WSD tool (Tan, 2014). For each aspect candidate, we take it as the target and randomly sample a bunch of sentences that contain this target. We use the extended word sense disambiguation algorithm (Banerjee and Pedersen, 2003) in this tool. We count the total occurrences for each noun sense (synset) of the candidate and match the candidate to the most frequent synset. However, such a method is not good enough for our problem, as shown in the results later. It only considers the local context information within the review sentence. Whats more, the review sentences are usually very short and colloquial, which makes it more difficult to match properly by a common WSD algorithm. Therefore, it is critical to construct more reliable contexts for aspect candidate matching.

To achieve this goal, we cluster the aspect candidates with similar semantics together. Then, for each aspect candidate, we take the other candidates within the same cluster as its context for later disambiguation. As shown in the first step of stage 2 in Figure 3, the semantic similar aspect candidates such as *lens, lens_cover, zoom_lens, exposure and shutter* are clustered together. For example, we can disambiguate the sense of *shutter* by leveraging *lens, lens_cover, zoom_lens, and exposure*. We observed that our aspect candidates can be fine-grain clustered with a two-stage k-means clustering method,[1] which generates the better context for the aspect candidates. More specifically, for a particular aspect candidate a_t from the cluster $C = \{a_1, a_2, ..., a_t, ..., a_n\}$, we calculate the context vector of a_t as:

$$c(a_t) = \sum_{i=1, i \neq t}^{n} E(a_i), \quad (1)$$

where $c(a_t)$ denotes the context vector of a_t, and $E(a_i)$ represents the embedding of a_i. The set of candidate synsets $S(a_t) = \{s_1^t, s_2^t, ..., s_m^t\}$ consists of the noun senses (e.g. s_i^t) of a_t from WordNet. Each sense s_i^t is associated with a *gloss* g_i^t (i.e. a brief definition of s_i^t) which covers the semantics of the sense. Therefore, we encode s_i^t as the summation of the word vectors in g_i^t:

$$v(s_i^t) = \sum_{j=1}^{q} E(w_j^{t,i}), \quad (2)$$

$W(g_i^t)$ is the sequence of words in g_i^t, i.e., $W(g_i^t) = [w_1^{t,i}, w_2^{t,i}, ..., w_q^{t,i}]$. For each candidate sense s_i^t of the aspect candidate a_t, we calculate the cosine semantic similarity between $v(s_i^t)$ and $c(a_t)$, and match a_t to the most similar s_i^t.

2.3.2 Aspect Taxonomy Extraction from WordNet

In order to construct the aspect taxonomy from WordNet, we first extract the hypernym paths for every matched synsets in the previous step. By definition, a hypernym path p of synset s is the is-a relation path from s to the root synset (i.e. entity.n.01 for nouns). We extract the hypernym paths for each matched synset s_i in the WordNet ontology. Next, we scan over all the collected paths once to construct the aspect taxonomy which is a directed acyclic graph (DAG). In p, s_1 is the synset matched from our potential aspects, and s_{i+1} is the hypernym of s_i. As shown in step 2 of Stage 2 in Figure 3, we match the aspect candidate *shutter* to *shutter.n.01*. The only one hypernym path of *shutter.n.01* is *[shutter.n.01, optical_device.n.01, device.n.01, ..., entity.n.01]*.

However, the matched synset usually has multiple hypernym paths in WordNet. We use the following strategy to compact and minimize the aspect taxonomy:

- Among all the paths from an aspect candidate s_1, we will keep those paths that contain more than 1 aspect candidates, unless there's only one path from s_1. If all paths contain only 1 aspect candidate s_1 each, we will keep all of them.
- To further optimize the taxonomy structure, we induce a minimum subgraph from the original taxonomy using a heuristic algorithm (Kou et al., 1981). Such a subgraph satisfies the following conditions: 1) it contains all the nodes matched from aspect candidates; 2) the total number of nodes in the graph is minimal. Consequently, the induced graph is a weakly connected DAG.

After acquiring the aspect taxonomy for the given product or service, we can now tell if two aspects are semantically overlapped or not.

2.4 Aspect Ranking

In this section, we propose a novel method based on personalized page rank to compute the overall rank values for the potential aspects by leveraging the aspect taxonomy.

---

[1] The implementation details are in Section 3.2.

Let the aspect taxonomy be a graph $G = (V, E)$. Each node $v \in V$ is a synset in the aspect taxonomy and encoded as a vector by instantiating E as Glove embeddings in (2) . Each edge $e = \langle u, v \rangle$ carries a weight which is the semantic similarity between the nodes u and v, computed using cosine similarity.

Next, we perform the random walks on our constructed aspect taxonomy. The propagation starts from candidate aspect nodes in the aspect taxonomy, which are called *seeds* here. The rank values (aspect importance) of all nodes are:

$$x_t = (1 - \alpha) * Ax_{t-1} + \alpha * E, \qquad (3)$$

where t is the time step in random walk process. In the initial state E(i.e. x_0), the aspect importance only distributes on the seeds ($v \in V_b$). E_i is the i-th dimension of E, indicating the portion of aspect importance on node s_i at time step 0. E is calculated as follows:

$$E_i = \begin{cases} \frac{f(le(s_i))}{\sum_{j=1}^{n} f(le(s_j))} & \text{, if } s_i \text{ is a seed} \\ 0 & \text{, otherwise,} \end{cases} \qquad (4)$$

where n is the number of nodes in the graph, s_i is the synset node, $le(s_i)$ denotes the lemma form of s_i, and $f(le(s_i))$ represents the frequency that $le(s_i)$ is modified by adjectives.

The aspect importance is updated using the transition probabilities matrix A which are the normalized weights on the edges of the taxonomy. α is the teleport probability, which is the probability of returning to the initial distribution at each time step. α determines the distance of propagation of the taxonomy.

2.5 Aspect Generation

Finally, we generate the prominent aspects using the rank values of the aspects as well as the is-a relations in the aspect taxonomy.

We sort $le(s_i)$ in decreasing order by their rank values. We essentially take the top aspects from the sorted list. However there might be two types of overlapping that we need to avoid: i) duplicate: different synset nodes may map to the same aspects, i.e., $le(s_i) = le(s_j), s_i \neq s_j$ (aspects); ii) taxonomy overlap: the later aspect in the list is the hypernym or hyponym of the one of previous aspects. To this end, we just skip overlapped aspect, and move along the list until we generate K non-overlapping prominent aspects from the list.

3 Experiments

We compare the ExtRA framework with multiple strong baselines on extracting aspect terms from user reviews. We first introduce the dataset and the competing models, then show the quantitative evaluation as well as qualitative analysis for different models.

3.1 Dataset

We use the customer review corpora of 6 kinds of product and service [2] collected from popular websites, including Amazon, TripAdvisor and Yelp. The number of hotel reviews (Wang et al., 2011b) in the original corpus is huge. Therefore, we randomly sample 20% of the reviews to perform our experiments. The statistics of the corpora are shown in Table 1.

Table 1: Dataset statistics.

Product type	Source	#Reviews
hotel	TripAdvisor	3,155,765
mobile phone	Amazon	185,980
mp3 player	Amazon	30,996
laptop	Amazon	40,744
cameras	Amazon	471,113
restaurant	Yelp	269,000

Existing published aspect extraction datasets (Hu and Liu, 2004; Popescu and Etzioni, 2007; Pavlopoulos and Androutsopoulos, 2014; Ding et al., 2008) include only fine-grained aspects from reviews, which are not suitable for evaluating the performance of prominent aspects extraction. Therefore, we build a new evaluation dataset particularly for this task. Following the previous work (Ganu et al., 2009; Brody and Elhadad, 2010; Zhao et al., 2010; Wang et al., 2015) as well as the popular commercial websites (e.g. TripAdvisor), which most manually labeled 3-6 prominent aspects for rating, we set K as five. Therefore, we ask each annotator who are familiar with the domain to give 5 aspect terms which they think are most important for each category. We have five annotators in total. [3] One prominent aspect can be expressed by different terms. Thus, it is difficult to achieve a satisfied inner-agreement. We propose two evaluation methods, especially the soft accuracy in Section 3.3.1 to compensate

---

[2] The data is available from http://times.cs.uiuc.edu/~wang296/Data/ and https://www.yelp.com/dataset

[3] The complete labeled set of ExtRA is released at http://adapt.seiee.sjtu.edu.cn/extra/.

this problem. To acquire a relatively higher inner-agreement, we educate the annotators with top 100 frequent aspect candidates as hints. Though, they are not required to pick up labels from the candidates. The inter-annotator agreement of each product type shown in Table 3 is computed as the average jaccard similarity between every two annotators.

3.2 Baselines and ExtRA

We introduce three topic modeling based baselines for the task. These are **LDA** (Blei et al., 2003), **BTM** (Cheng et al., 2014) and **MG-LDA** (Titov and McDonald, 2008). MG-LDA is a strong baseline which attempts to capture multi-grain topics (i.e. global & local), where the local topics correspond to the rateable prominent aspects. We treat each review as a document and perform those models to extract K topics. Then, we select most probable words in each topic as our extracted aspect terms. To prevent extracting the same aspects (w) from different topics, we only keep w for the topic t with the highest probability $p(w|t)$ value, then re-select aspects for the other topics until we get K different aspects. For fair comparison among different models, the number of target aspects K is set as 5. The hyper-parameter of MG-LDA (global topics) is set to 30 with fine-tuning.

Another syntactic rule-based baseline model **AmodExt** is from the first stage of our framework. After extracting the aspect candidates using *amod-rule* in Section 2.2, we sort the aspect candidates by their counts of extracted occurrences. Then select the top K candidates as the prominent aspects.

ABAE (He et al., 2017) is a neural based model that can to infer K aspect types. Each *aspect type* is a ranked list of representative words. To generate K prominent aspects, we first infer K aspect types using *ABAE*, then select the most representative word from each aspect type.

For **ExtRA**, in the taxonomy construction stage, we use a two-stage K-means clustering method for synset matching task, and the cluster number is auto-tuned using silhouette score (Rousseeuw, 1987). We use SkipGram (Mikolov et al., 2013) model to train the embeddings on review texts for k-means clustering. We set the dimension of the embeddings as 100 and run 64 epochs for each product corpora. In the aspect ranking stage, we empirically set the teleport probability α as 0.5 which indicates that the expected walk-length from the seeds is $\frac{1}{\alpha} = 2$.

3.3 Evaluation

In this section, we compare ExtRA with five baseline models both quantitatively and qualitatively.

3.3.1 Quantitative Evaluation

First, we perform two experiments to justify our aspect taxonomy construction stage:

- To justify the synset matching step, we compare our proposed cluster method with classical WSD algorithm (Lesk) on matching accuracy. We manually label 100 sampled synset nodes for each category. The synset matching accuracies are shown in Table 2. We can see that our clustering method is effective for the synset matching task.

- We induce the aspect taxonomy using a heuristic algorithm to obtain more compact and aspect-oriented subgraph. We show the size of aspect taxonomy induced before and after taxonomy minimization in Figure 4.

Next, we evaluate our model as well as above baselines on the evaluation dataset described above. We did not remove the duplicate aspect labels for the qualitative evaluation, since the repeated aspects are assume to be better. For a given category, we first calculate the percentage of the 25 labels that exactly match one of the 5 aspect terms generated by the model as the *hard accuracy* of the model. Formally, $Aspects(m) = [a_1, a_2, a_3, a_4, a_5]$ denotes the five prominent aspects generated from model m for the given category. $L = [l_1, l_2, ..., l_{25}]$ are the 25 golden aspect terms, where $L^{(h)} = [l_{5h-4}, ..., l_{5h}]$ are from the h-th human annotator. The hard accuracy is defined as:

$$hacc(m) = \frac{\sum_{i=1}^{25} hit(Aspects(m), l_i)}{25} \quad (5)$$

$$hit(Aspects(m), l_i) = \begin{cases} 1, & l_i \in Aspects(m) \\ 0, & \text{otherwise,} \end{cases} \quad (6)$$

However, counting the number of exact matches makes the accuracy score discrete and coarse. Besides, it penalizes aspect terms that don't match the label but actually have similar meanings. To

Table 2: WordNet Synset matching accuracies

	hotel	mp3	cameras	mobile phone	laptop	restaurant
LESK	0.71	0.59	0.62	0.64	0.53	0.65
Cluster	**0.86**	**0.83**	**0.74**	**0.80**	**0.78**	**0.69**

Table 3: Inner-annotator agreements

	hotel	mp3	cameras	mobile phone	laptop	restaurant
Jaccard	0.470	0.554	0.304	0.440	0.271	0.671

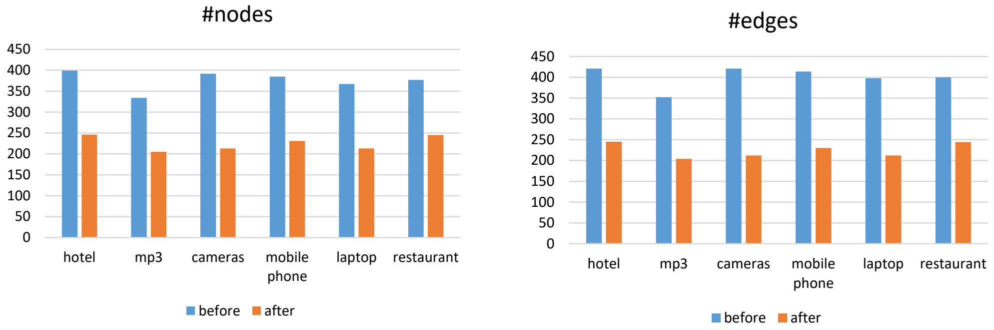

Figure 4: Statistics of induced aspect taxonomy before and after taxonomy minimization

Table 4: Comparison of *hard* (upper row) & *soft* (lower row) accuracies using different models for aspect extraction.

	LDA	BTM	MG-LDA	ABAE	AmodExt	ExtRA
hotel	0.16	0.16	0.16	0.16	0.44	**0.56**
	0.50	0.49	0.67	0.35	0.65	**0.70**
mp3	0.0	0.08	0.08	0.0	0.35	**0.44**
	0.47	0.49	0.47	0.32	0.58	**0.60**
camera	0.24	**0.40**	0.28	0.04	0.04	0.32
	0.56	**0.69**	0.54	0.29	0.41	0.55
mobile phone	0.16	0.0	0.28	0.0	0.52	**0.60**
	0.58	0.33	0.58	0.31	**0.73**	0.71
laptop	0.08	0.24	0.24	0.0	0.24	**0.28**
	0.40	0.50	0.50	0.22	0.51	**0.53**
restaurant	0.20	0.0	0.0	0.0	**0.56**	**0.56**
	0.49	0.38	0.42	0.29	**0.77**	0.72

remedy this, we propose the *soft accuracy* evaluation measure. For each set of five golden labels from h-th annotator, we first align each generated aspect $a_k \in Aspects(m)$ with one golden aspect $l_j \in L^{(h)}$ (i.e. $align^{(h)}(a_k) = l_j$). We align the exact match terms together, and then choose the optimal alignment for the others by permuting all possible alignments. The optimal alignment $align^{(h)}(a_k)$ acheives maximum soft accuracy. Then we calculate the soft matching score between $Aspects(m)$ and $L^{(h)}$ as $\sum_{k=1}^{K} sim(a_k, align^{(h)}(a_k))$, where sim is the cosine similarity computed by Glove (2014) [4]. We then compute the soft accuracy measure as follows:

$$sacc(m) = \frac{1}{5} * \sum_{h=1}^{5} \sum_{k=1}^{K} sim(a_k, align^{(h)}(a_k)), \tag{7}$$

where $K = 5$ in this case. The comparison results are shown in Table 4.

Our model (ExtRA) outperforms all the other baselines in all categories except cameras using the hard accuracy measure. Besides, ExtRA is the best model on four out of six products under the soft accuracy measure. As shown in Table 2, the accuracy for synset matching is relatively low for cameras and restaurant, resulting in the lower accuracy in overall aspect extraction.

3.3.2 Qualitative Analysis

To qualitatively evaluate different models, we present the extracted 5 aspect terms by each model from each domain in Table 5. Our model (ExtRA) has significant advantage over other baselines for that we can do better aspect extraction with reasonable results, and extract not only words but also phrases as prominent aspects, *e.g. sound quality, image quality*. The proposed model avoid the overlapping aspects appeared in our strong baseline (AmodExt) by deduplication using generated aspect taxonomy information. The overlapping aspects are marked in italics. For example, both *lo-*

---

[4] We use the GloVe embeddings with 300 dimensions, trained from 840B tokens using common crawl data.

Table 5: The five prominent aspect terms

hotel	LDA	room, pool, stay, good, nice
	BTM	walk, good, room, stay, check
	MGLDA	room, stay, good, location, staff
	ABAE	shouted, room, terrific, accommodation, alexanderplatz
	AmodExt	room, *location, place*, view, staff
	ExtRA	**room, location, view, staff, service**
mp3	LDA	work, great, good, music, ipod
	BTM	battery, ipod, work, song, good
	MGLDA	battery, ipod, music, song, good
	ABAE	documentation, content, portability, bought, table
	AmodExt	drive, quality, sound, feature, device
	ExtRA	**drive, sound_quality, feature, screen, software**
cameras	LDA	lens, picture, buy, video, mode
	BTM	battery, picture, function, lens, good
	MGLDA	battery, picture, good, mpcture, mode
	ABAE	toy, picture, mailed, ultrazoom, sharpness
	AmodExt	*picture, photo*, quality, feature, shot
	ExtRA	**image_quality, photograph, feature, shot, lens**
mobile phone	LDA	battery, buy, good, apps, work
	BTM	core, good, work, para, apps
	MGLDA	work, battery, screen, good, card
	ABAE	cracked, amazing, continuously, archive, bought
	AmodExt	feature, screen, price, camera, quality
	ExtRA	**feature, price, screen, quality, service**
laptop	LDA	screen, good, buy, drive, chromebook
	BTM	windows, screen, work, drive, good
	MGLDA	windows, battery, screen, good, year
	ABAE	salign, returned, affordable, downloads, position
	AmodExt	drive, machine, price, screen, life
	ExtRA	**drive, price, screen, deal, performance**
restaurant	LDA	food, good, room, time, great
	BTM	good, room, pour, time, order
	MGLDA	great, good, place, time, make
	ABAE	jones, polite, told, chickpea, place
	AmodExt	food, service, place, experience, price
	ExtRA	**service, food, experience, company, price**

cation and *place* are extracted as top aspects, but they mean nearly the same concept. The results from other baseline methods, inevitably contain some sentiment words and opinions, like *good, nice, great, etc.* Our model resolves such drawback by extracting aspect candidates from only nouns and using syntactic rules to find words that are frequently modified by adjectives.

4 Related Work

Existing research on *aspect-based review analysis* has focused on mining opinion based on given aspects (Su et al., 2008; Zeng and Li, 2013) or jointly extracting the aspects and sentiment (Lin and He, 2009; Zhao et al., 2010; Qiu et al., 2011; Wang et al., 2015; Liu et al., 2016). They are mostly interested in detecting aspect words in a given sentence, whereas our goal is to extract the most prominent aspects of a type of product from a large number of reviews about that product type. We divide the existing work on review aspect extraction into three types:

- *rule-based* methods, most of which utilize handcrafted rules to extract candidate aspects and then perform clustering algorithm on them.
- *topic modeling based* methods, which directly model topics from texts and then extract aspects from the topics.
- *neural network based* methods, which takes advantage of the recent deep neural network models.

4.1 Rule-based Methods

These methods leverage word statistical and syntactic features to manually design rules, recognizing aspect candidates from texts. Poria et al. (2014) use manually crafted mining rules. Qiu et al. (2011) also used rules, plus the Double Propagation method to better relate sentiment to aspects. Gindl et al. (2013) cooperate the Double Propagation with anaphora resolution for identifying co-references to improve the accuracy. Su et al. (2008) used a clustering method to map the implicit aspect candidates (which were assumed to be the noun form of adjectives in the paper) to explicit aspects. Zeng et al. (2013) mapped implicit features to explicit features using a set of sentiment words and by clustering explicit feature-sentiment pairs. Rana et al. (2017) propose a two-fold rules-based model, using rules defined by sequential patterns. Their first fold extracts aspects associated with domain independent opinions and the second fold extracts aspects associated with domain dependent opinions.

However, such rule-based models are designed for extracting product features which can not easily adapt to our K most prominent aspect extraction problem. Besides, most of them require human efforts to collect lexicons and to carefully design complex rules and thus do not scale very well.

4.2 Topic Modeling Based Methods

Most work in this domain are based on two basic models, pLSA(Hofmann, 1999) and LDA(Blei et al., 2003). The variants of these models consider two special features of review texts: 1) topics shift quickly between sentences, 2) sentiment plays an important role and there is a strong correlation between sentiments and aspects. The approach of Lin et al. (2011) models are parallel aspects and sentiments per review. Lin et al. (2009) models the dependency between the latent aspects and ratings. Wang et al. (2011a) proposed a generative model which incorporates topic modeling technique into the latent rating regression model (Wang et al., 2010). Moghaddam et al. (2012) made a nice summarization of some basic variations of LDA for opinion mining. In stead of using topics, our method relies on word embeddings to capture the latent semantics of words and phrases and achieves better results. *MG-LDA* (Titov and McDonald, 2008) is a variant of LDA that can also model topics at different granularities, which are based on extensions to standard topic modeling methods such as LDA and PLSA to induce multigrain topics. *D-PLDA* (Moghaddam and Ester, 2012), is a variant of LDA models, which is designed specifically for modeling topics from user reviews. D-PLDA only considers opinion-related terms and phrases, and nouns and phrases are controlled by two separate hidden parameters. Thus, the model needs aspects, ratings, and phrases as input, which are all very expensive.

4.3 Neural Network Based Methods

He et al. (2017) propose a neural attention model for identifying aspect terms. Their goal is similar to ours but instead of directly comparing their extracted terms with the gold standard, they ask human judges to map the extracted terms to one of the prominent gold aspects manually before computing the precision/recall. This evaluation methodology mixed machine results with human judgment and is problematic in our opinion. Our experiments showed that their output aspects are too fine-grained and can not be used as prominent aspects.

5 Conclusion

In this paper, we propose an unsupervised framework ExtRA for extracting the most prominent aspect terms about a type of product or service from user reviews, which benefits both qualitative and quantitative aspect-based review summarization. Using WordNet as a backbone, and by running personalized page rank on the network, we can produce aspect terms that are both important and non-overlapping. Results show that this approach is more effective than a number of other strong baselines.

Acknowledgment

Kenny Q. Zhu is the contact author and was supported by NSFC grants 91646205 and 61373031. Thanks to the anonymous reviewers for their valuable feedback.

References

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Ijcai*, volume 3, pages 805–810.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812. Association for Computational Linguistics.

Xueqi Cheng, Xiaohui Yan, Yanyan Lan, and Jiafeng Guo. 2014. Btm: Topic modeling over short texts. *TKDE*.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proc. WSDM*.

Gayatree Ganu, Noemie Elhadad, and Amélie Marian. 2009. Beyond the stars: improving rating predictions using review text content. In *WebDB*, volume 9, pages 1–6. Citeseer.

Stefan Gindl, Albert Weichselbraun, and Arno Scharl. 2013. Rule-based opinion target and aspect extraction to acquire affective knowledge. In *Proc. of WWW*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *ACL*.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proc. of SIGIR*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proc. of KDD*.

L Kou, George Markowsky, and Leonard Berman. 1981. A fast algorithm for steiner trees. *Acta informatica*.

Himabindu Lakkaraju, Chiranjib Bhattacharyya, Indrajit Bhattacharya, and Srujana Merugu. 2011. Exploiting coherence for the simultaneous discovery of latent facets and associated sentiments. In *SDM*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proc. of CIKM*.

Jialu Liu, Jingbo Shang, and Jiawei Han. 2017. Phrase mining from massive text and its applications. *Synthesis Lectures on Data Mining and Knowledge Discovery*.

Qian Liu, Zhiqiang Gao, Bing Liu, and Yuanlin Zhang. 2015. Automated rule selection for aspect extraction in opinion mining. In *IJCAI*, volume 15, pages 1291–1297.

Qian Liu, Bing Liu, Yuanlin Zhang, Doo Soon Kim, and Zhiqiang Gao. 2016. Improving opinion aspect extraction using semantic similarity and aspect associations.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*.

Samaneh Moghaddam and Martin Ester. 2012. On the design of lda models for aspect-based opinion mining. In *Proc. of CIKM*.

John Pavlopoulos and Ion Androutsopoulos. 2014. Aspect term extraction for sentiment analysis: New datasets, new evaluation measures and an improved unsupervised method. *Proceedings of LASMEACL*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Ana-Maria Popescu and Orena Etzioni. 2007. Extracting product features and opinions from reviews. In *Natural language processing and text mining*.

Soujanya Poria, Erik Cambria, Lun-Wei Ku, Chen Gui, and Alexander Gelbukh. 2014. A rule-based approach to aspect extraction from product reviews. In *Proc. of the Workshop on Natural Language Processing for Social Media*.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *CL*.

Toqir A Rana and Yu-N Cheah. 2017. A two-fold rule-based model for aspect extraction. *Expert Systems with Applications*.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Qi Su, Xinying Xu, Honglei Guo, Zhili Guo, Xian Wu, Xiaoxun Zhang, Bin Swen, and Zhong Su. 2008. Hidden sentiment association in chinese web opinion mining. In *Proc. of WWW*.

Liling Tan. 2014. Pywsd: Python implementations of word sense disambiguation (wsd) technologies [software].

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *WWW*.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proc. of KDD*.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011a. Latent aspect rating analysis without aspect keyword supervision. In *Proc. of KDD*.

Hongning Wang, Chi Wang, ChengXiang Zhai, and Jiawei Han. 2011b. Learning online discussion structures by conditional random fields. In *SIGIR*.

Linlin Wang, Kang Liu, Zhu Cao, Jun Zhao, and Gerard de Melo. 2015. Sentiment-aspect extraction based on restricted boltzmann machines. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 616–625.

Lingwei Zeng and Fang Li. 2013. A classification-based approach for implicit feature identification. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 56–65. Association for Computational Linguistics.

Cross-Lingual Cross-Platform Rumor Verification Pivoting on Multimedia Content

Weiming Wen[1], **Songwen Su**[1], **Zhou Yu**[1]

[1]University of California, Davis
`wmwen@ucdavis.edu`

Abstract

With the increasing popularity of smart devices, rumors with multimedia content become more and more common on social networks. The multimedia information usually makes rumors look more convincing. Therefore, finding an automatic approach to verify rumors with multimedia content is a pressing task. Previous rumor verification research only utilizes multimedia as input features. We propose not to use the multimedia content but to find external information in other news platforms pivoting on it. We introduce a new features set, cross-lingual cross-platform features that leverage the semantic similarity between the rumors and the external information. When implemented, machine learning methods utilizing such features achieved the state-of-the-art rumor verification results.

1 Introduction

Social network's unmoderated nature leads to the spread and emergence of information with questionable sources. With the increasing popularity of the social media, we are exposed to a plethora of rumors. Here we borrow the rumor definition from DiFonzo and Bordia (2007) as unverified information. Unmoderated rumors have not only caused financial losses to trading companies but also panic for the public (Matthews, 2013). Especially if rumors contain multimedia content, the public generally accepts the multimedia information as a "proof of occurrence" of the event (Sencar and Memon, 2009). Readers usually don't have time to look through similar events across different platforms to make an informed judgment. Therefore, even if a credible platform, such as CNN, has debunked a rumor, it can still go viral on other social media platforms.

Intuitively, people believe fake rumors would contain fabricated multimedia content. Boididou et al. (2015b) used forensics features for detecting multimedia fabrication to verify rumors. However, these features did not lead to noticeable improvement. We suspect that this is because un-tampered multimedia content can still convey false information when paired with fake news from a separate event. For example, Figure 1 shows one fake post on MH 370 that used a real video about US Airways Flight 1549. Inspired by the fact that readers tend to search related information covered by different media outlets to garner an objective view, we propose to verify rumors pivoting on multimedia content to tackle such problems. Compared to keywords, searching information pivoting on the visual content is more effective and accurate.


Malaysian Airlines Finally Found! Watch the video.
news.yahoo.com | 594,298 people shared this video.
VIRALGAG.NET

Figure 1: A video of US Airways Flight 1549 was borrowed by news on Malaysia Airlines Flight 370.

In order to access information from different platforms easily, we created a new rumor verification dataset by expanding a Twitter rumor dataset to include webpages from different social media platforms using search engines. Previous rumor verification datasets are mainly monolingual, such as English (Derczynski et al., 2017) or Chinese (Wu et al., 2015). However, textual information in the native language where the rumor happened can be more helpful when it comes to ver-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3487–3496
Brussels, Belgium, October 31 - November 4, 2018.

ifying worldwide rumors. Therefore, we not only indexed English webpages by searching Google with images but also included Chinese webpages via Baidu.

We next introduced our cross-lingual cross-platform features which capture the similarity and agreement among rumors with posts from different social media. We built an automatic verification model using the proposed features and achieved the state-of-the-art performance on the MediaEval 2015's Verifying Multimedia Use (VMU 2015) dataset (Boididou et al., 2015a) utilizing information from Google.

Collecting and annotating rumors in foreign languages is difficult and time-consuming, especially for languages with low rumor verification labeling. Finding out an automatic way to verify those rumors in an unsupervised way is also meaningful. Since our cross-lingual cross-platform features are adaptable to rumors in different languages, we demonstrated that these features could transfer learned knowledge by training on one language and testing on another. Such cross-lingual adaptation ability is especially useful for predicting rumors that have low annotation resource with available annotated rumors in languages such as English.

We published our code and dataset on GitHub[1].

2 Related work

Previous research has utilized multimedia information for rumor verification in various ways. Zampoglou et al. (2015); Jin et al. (2015); Boididou et al. (2015b) verified rumors by leveraging forensic features which are extracted to ensure the digital images are not tempered (Sencar and Memon, 2009). However, none of these studies found such information useful for rumor verification on a Twitter-based multimedia dataset. Jin et al. (2017) incorporated image features using a pre-trained deep convolutional neural network (Krizhevsky et al., 2012) on the extended Twitter-based multimedia dataset. Although the image features improve their results, their framework cannot outperform methods other than multimodal fusing networks. One possible reason is that the multimedia content in fake rumors is borrowed from another real event and usually their content corresponds to the text of the rumors. In this case, the image itself is real but not real in the context of the fake news. We thus propose to leverage the multimedia information by finding the agreement and disagreement among posts that are from different social media platforms but share similar visual contents.

The agreement between rumors and their comments is used heavily in automatic verification. Mendoza et al. (2010) declared that fake rumors tended to have more people question their validity. Later, Qazvinian et al. (2011) first annotated comments on tweets as supporting, denying or querying, and then used such stance information in the classification to leverage the "wisdom of crowds". Recently, the best performing system (Enayet and El-Beltagy, 2017) in RumourEval shared task at SemEval 2017 (Derczynski et al., 2017) also used such information. However, the crowd is not always wise. For example, Starbird et al. (2014) suspected the correctness of public opinions in rumors, pointing out some certain fake news received more support than questions. In our work, instead of using the "wisdom of crowds", we used the knowledge from different news platforms to assist rumor verification.

Computational journalism (Cohen et al., 2011) exploits external knowledge widely. Diakopoulos et al. (2012) first leveraged information from reliable sources in the context of journalism. They developed a tool for journalists to search for and assess sources in social media around breaking news events. Ciampaglia et al. (2015) utilized factual knowledge bases, such as Wikipedia, to assess the truth of simple statements. Shao et al. (2016) designed a system for tracking rumors on different platforms, which is probably the closest work to ours. However, they did not utilize cross-platform information for rumor verification. Our proposed method is able to leverage information on any platform to verify rumors as long as it has both textual and multimedia information.

3 CCMR dataset

We created a cross-lingual cross-platform multimedia rumor verification dataset (CCMR) to study how to leverage information from different media platforms and different languages to verify rumor automatically. CCMR consists of three sub-datasets: CCMR Twitter, CCMR Google, and CCMR Baidu.

CCMR Twitter is borrowed from VMU 2015 dataset (Boididou et al., 2015a). There are 17

---

[1] https://github.com/WeimingWen/CCRV

ID	Event	Twitter		Google			Baidu		
		Real	**Fake**	**Real**	**Fake**	**Others**	**Real**	**Fake**	**Others**
01	Hurricane Sandy	4,664	5,558	1,836	165	203	693	134	291
02	Boston Marathon bombing	344	189	619	54	49	317	55	16
03	Sochi Olympics	0	274	139	132	76	64	124	53
04	MA flight 370	0	310	143	65	115	80	59	31
05	Bring Back Our Girls	0	131	29	42	37	2	6	4
06	Columbian Chemicals	0	185	35	2	26	19	1	0
07	Passport hoax	0	44	24	0	2	16	0	4
08	Rock Elephant	0	13	3	17	0	4	2	14
09	Underwater bedroom	0	113	1	58	0	0	37	13
10	Livr mobile app	0	9	0	4	11	0	0	0
11	Pig fish	0	14	3	13	4	1	12	7
12	Solar Eclipse	140	137	40	64	39	0	10	91
13	Girl with Samurai boots	0	218	2	52	6	2	48	0
14	Nepal Earthquake	1,004	356	257	60	107	159	19	81
15	Garissa Attack	73	6	60	0	3	36	1	0
16	Syrian boy	0	1,786	4	1	3	0	0	0
17	Varoufakis and zdf	0	61	2	0	18	0	0	0
	Total	6,225	9,404	3,197	729	699	1,393	508	605

Table 1: CCMR dataset statistics.

events containing fake and real posts with images or videos shared on Twitter. We created CCMR Google and CCMR Baidu by searching Google and Baidu indexed webpages that share similar multimodal content with CCMR Twitter. The upper part of Figure 2 shows the collection process. We searched Google with every image (URL for video) in CCMR Twitter to get English webpages. Then we indexed those webpages to form CCMR Google. Similarly, we searched Baidu to get Chinese webpages and created CCMR Baidu. Two human annotators manually annotated both datasets. The annotation is for better analysis and dataset quality control. None of it is utilized during our feature extraction process. Annotators were asked to label collected webpages based on their title and multimedia content. If they are not enough to tell fake news from real news, the webpage is labeled as "others". The Cohen's kappa coefficient for two annotators is 0.8891 in CCMR Google and 0.7907 in CCMR Baidu. CCMR has 15,629 tweets indexed by CCMR Twitter (Twitter), 4,625 webpages indexed by CCMR Google (Google) and 2,506 webages indexed by CCMR Baidu (Baidu) related to 17 events. The webpages from Google and Baidu are in English and Chinese respectively. The statistics of the CCMR dataset with respect to each event is listed in Table 1.

3.1 Observation in Annotation

We observe that 15.7% of webpages are fake in CCMR Google while 20.3% in CCMR Baidu. We speculate that this is because all events in CCMR dataset took place outside China. Chinese webpages searched via Baidu are thus more likely to mistake the information. In the manual annotation process, we found that many images are actually borrowed from other events, which confirms our assumption. Another interesting observation is that webpages indexed by Baidu tend to have more exaggerating or misleading titles to attract click rates. We labeled such webpages as fake if they also convey false information through their multimedia content.

We also found that news in different languages has different distributions concerning the subtopics of the event. For example, in the Boston marathon bombing, Baidu indexed Chinese reports generally put more emphasis on a Chinese student who is one of the victims, while Google indexed English reports cover a wider range of subtopics of the event, such as the possible bomber. This phenomenon is understandable as social media from a specific country usually focus more on information related to their readers.

4 Framework Overview

Figure 2 describes the overview of our framework. After collecting CCMR dataset in Section 3, we first performed Twitter rumor verification leveraging Google in Section 6 as shown in the bottom left of the figure. We extracted cross-lingual cross-platform features for tweets in CCMR Twitter leveraging webpages from CCMR Google (TFG). Section 5 discusses the automatic construction of this feature set. We then use the features to verify rumors automatically.

We then performed Twitter rumor verification

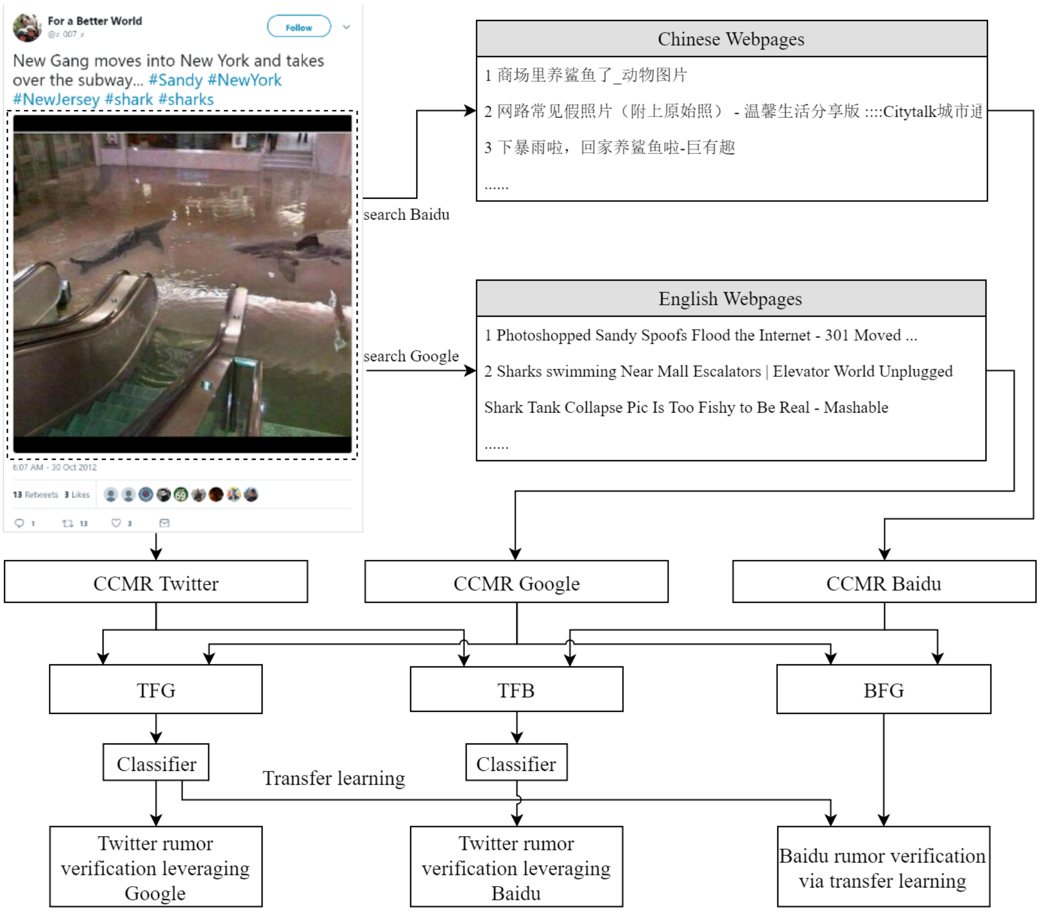

New Gang moves into New York and takes over the subway... #Sandy #NewYork #NewJersey #shark #sharks

1 商场里养鲨鱼了_动物图片
2 网路常见假照片（附上原始照）- 温馨生活分享版 ::::Citytalk城市通
3 下暴雨啦，回家养鲨鱼啦-巨有趣
......

1 Photoshopped Sandy Spoofs Flood the Internet - 301 Moved ...
2 Sharks swimming Near Mall Escalators | Elevator World Unplugged
Shark Tank Collapse Pic Is Too Fishy to Be Real - Mashable
......

Figure 2: The information flow of our proposed pipeline. TFG represents the cross-lingual cross-platform features for tweets leveraging Google information, while TFB is similar but leverages Baidu information instead. BFG means cross-lingual cross-platform features for Baidu leveraging Google information.

leveraging Baidu in Section 7 to test if our method can verify rumors by borrowing information from different languages and platforms. This experiment is meant to demonstrate that our method is language and platform agnostic. Such an advantage also enables our method to use one language information to predict in another language (see the experiment in Section 8). We extracted cross-lingual cross-platform features for tweets leveraging webpages from CCMR Baidu instead (TFB) and used it to verify tweets in Section 6.

In Section 8, we performed Baidu rumor verification via transfer learning to test the cross-lingual adaptation ability of the cross-lingual cross-platform features. We treated Chinese webpages in CCMR Baidu as rumors and empirically verified them via transfer learning. We extracted cross-lingual cross-platform features for Baidu webpages leveraging Google (BFG) in Section 6. Since BFG and TFG are both cross-lingual cross-platform features leveraging Google, we adopted the classifier pre-trained with TFG on CCMR Twitter to verify webpages in CCMR Baidu using BFG, under the assumption that tweets and webpages follow a similar distribution.

Although we labeled webpages in CCMR Google and CCMR Baidu, we did not leverage the annotation here because annotation is time-consuming and using annotation information is not generalizable to other datasets.

5 Cross-lingual Cross-platform Features

We propose a set of cross-lingual cross-platform features to leverage information across different social media platforms. We first embed both the rumor and the titles of the retrieved webpages into 300-dimension vectors with a pre-trained multilingual sentence embedding. It is trained using English-Chinese parallel news and micro-blogs in UM-Corpus (Tian et al., 2014). We encode English-Chinese parallel sentences with the same word dictionary, as they share some tokens such as

URLs and punctuation. We then use a two-layer bidirectional gated recurrent unit (GRU) (Cho et al., 2014) to generate hidden states. We obtain the embedding by averaging the hidden states of the GRU. A pairwise ranking loss is used to force the cosine distance between embeddings of paired sentences to be small and unpaired sentences to be large. We train our multilingual sentence embedding on 453,000 pairs of English-Chinese parallel sentences and evaluated it on another 2000 sentence pairs. Our published code includes the implementation details of the multilingual sentence embedding.

After obtaining the embeddings of the rumor and the titles of the retrieved webpages, we further calculate the distance and agreement features between these embeddings to create a set of cross-lingual cross-platform features. In total, there are 10 features, two for distance features and eight for agreement features.

5.1 Distance Features

We compute the cosine distances between the embeddings of the target rumors and the titles of the retrieved webpages. The distance indicates if the rumor has similar meaning with the retrieved webpages that have similar multimedia content. We calculate the mean and variance of the distance.

The mean of the distance indicates the average similarity between a rumor and its corresponding webpages from other platforms. A high value in mean suggests that the rumor is very different from the retrieved information from other platforms. We suspect that rumors with this property have a higher probability of being fake. Because the rumor might have borrowed the image from another event that was covered in the retrieved information. Meanwhile, the variance indicates how much these retrieved webpages are different from each other. A high variance indicates that the multimedia information is used by different events or is described in different statements. So the event or the statement the rumor covers could be fake.

5.2 Agreement Features

We first pre-train an agreement classifier on a stance detection dataset provided by the Fake News Challenge[2]. This dataset provides pairs of English sentences with their agreement annotations in “agree”, “disagree”, “discuss” and “unrelated”. Figure 3 shows four example body texts of a headline corresponding to each type of annotation in the dataset. During the training process, we embed the sentences in the Fake News Challenge dataset using our pre-trained multilingual sentence embedding. We then concatenate the embeddings of the sentence pair as the input. We use a multi-layer perceptron to pre-train our agreement classifier. We randomly select a balanced development set containing 250 pairs for each label (1000 in total) and train our agreement classifier on the rest of 74,385 pairs. The agreement classifier achieves 0.652 in the macro-averaged F1-score on the development set. Our published code also includes the details.

EXAMPLE HEADLINE

“Robert Plant Ripped up $800M Led Zeppelin Reunion Contract”

EXAMPLE BODY TEXTS

“Led Zeppelin’s Robert Plant turned down £500 MILLION to reform supergroup.”

CORRECT ANNOTATION: **AGREE**

“No, Robert Plant did not rip up an $800 million deal to get Led Zeppelin back together.”

CORRECT ANNOTATION: **DISAGREE**

“Robert Plant reportedly tore up an $800 million Led Zeppelin reunion deal.”

CORRECT ANNOTATION: **DISCUSSES**

“Richard Branson’s Virgin Galactic is set to launch SpaceShipTwo today.”

CORRECT ANNOTATION: **UNRELATED**

Figure 3: An example headline and its body texts of the Fake News Challenge dataset.

We calculate the agreement features using the mean and variance of the prediction probability between the rumor and all the retrieved webpages. There are in total four agreement labels. Therefore, we have eight agreement features in total. Agreement features capture information about if the rumor’s statement agrees with the corresponding information in other platforms. Besides being able to gain similar benefits as distance features, our agreement features also capture the case that the information stance is portrayed differently by different resource rumors. Conflicting information will also be an indicator of fake news.

---

[2] http://www.fakenewschallenge.org/

6 Rumor Verification Leveraging Cross-platform Information

We extracted cross-lingual cross-platform features for tweets in CCMR Twitter leveraging Google (TFG) and evaluated the effectiveness of TFG on rumor verification tasks.

We proposed a simple multi-layer perceptron classifier (MLP) to leverage the extracted features. MLP has two fully-connected hidden layers of 20 neurons with ReLU as the activation function. Each layer is followed by a dropout of 0.5. We evaluated TFG in two settings: *task* and *event*. 1) The *task* setting used event 1-11 for training and event 12-17 for testing according to (Boididou et al., 2015a). 2) The *event* setting evaluated the model performance on a leave-one-event-out cross-validation fashion. F1-score is used for evaluation metric. Since both collecting source rumors from Google and doing feature extraction for a given tweet can be done automatically, it is fair to compare the performance of our model with baselines described below.

6.1 Baselines

We adopted three best performing models in the VMU 2015 task as our baselines:

UoS-ITI (UoS) (Middleton, 2015) uses a natural language processing pipeline to verify tweets. It is a rule-based regular expression pattern matching method. It ranks evidence from Twitter according to the most trusted and credible sources.

MCG-ICT (MCG) (Jin et al., 2015) is an approach including two levels of classification. It treats each image or video in the dataset as a topic and uses the credibility of these topics as a new feature for the tweets. They used the tweet-based and user-based features (Base), such as the number of hashtags in the tweet or the number of friends of the user who posted the tweet.

CERTH-UNITN (CER) (Boididou et al., 2015b) uses an agreement-based retraining scheme. It takes advantage of its own predictions to combine two classifiers built from tweet-based and user-based features (Base). Besides features provided by the task, it included some additional features such as the number of nouns in tweets and trust scores of URLs in tweets obtained from third-party APIs.

Method	F1-Task	F1-Event
UoS-ITI	0.830	0.224
MCG-ICT	**0.942**	0.756
CERTH-UNITN	0.911	0.693
TFG	0.908	**0.822**
BFG	0.810	0.739
Combo	0.899	0.816

Table 2: The *task* and *event* settings performance.

ID	UoS	MCG	CER	TFG	TFB
01	0.658	0.594	**0.718**	0.715	0.704
02	0.007	0.494	**0.745**	0.557	0.448
03	0.057	0.882	0.595	**0.956**	0.822
04	0.538	0.826	0.717	**0.856**	0.678
05	0.000	**0.988**	0.947	0.969	0.956
06	0.555	0.949	0.916	0.912	**1.000**
07	0.000	**1.000**	0.475	0.989	0.989
08	0.000	0.870	**1.000**	0.960	**1.000**
09	0.000	0.772	0.996	**1.000**	0.996
10	0.000	0.615	0.821	**0.875**	-
11	0.000	**0.963**	0.000	**0.963**	0.667
12	0.000	0.655	**0.754**	0.677	0.656
13	0.000	0.954	0.795	**0.998**	0.850
14	0.000	0.330	0.419	0.409	**0.430**
15	0.000	0.130	**0.156**	0.145	0.154
16	**1.000**	0.990	0.999	0.996	-
17	**1.000**	0.827	**1.000**	0.992	-
Avg	0.224	0.756	0.693	**0.822**	0.739

Table 3: F1-scores for each event.

6.2 Results

We describe the *task* setting results in Table 2, and detailed per-event results in Table 3. Although TFG does not achieve the highest F1-score in the *task* setting, it is mainly due to the split of the dataset. More than half of the tweets in the test set do not have images. Thus we can only leverage cross-platform information by searching videos' URLs, which results in less accurate cross-lingual cross-platform features. In the *event* setting, which has a more fair comparison, TFG outperformed other methods with a big margin ($p<0.001$). It is surprising to see that only 10 features extracted from external resources indexed by search engines leveraged by a simple classifier can bring such a big performance boost.

To further explore the quality of the cross-lingual cross-platform features, we calculated the Pearson correlation coefficient (PCC) between each feature with respect to the tweet's label (fake or real). We evaluated both TFG and features used by baseline models. Table 4 lists the top six features with the highest absolute PCC values. A positive value indicates this feature positively correlates with fake news. We can see four out of the top six features are cross-lingual cross-platform features. The variance of the unrelated probability (unrelated variance) has the highest score, which further validates our design intuition that tweets might convey false information when they have different agreement with all other webpages that shared similar multimedia content. The second feature, "distance var" is also highly correlated with fake news. This result supports our hypothesis that if there is a large information dissimilarity across different platforms, there is a high probability of fake information involved. The only feature from baselines (56 features in total) in the top six features is whether a tweet contains an exclamation mark or not (containsExclaimationMark).

Feature	PCC
unrelated variance	0.306
distance variance	0.286
agree variance	0.280
discuss mean	-0.231
unrelated mean	0.210
containsExclamationMark	0.192

Table 4: Top six features correlated with fake news.

6.3 Analysis

We found that Google webpages usually cover a complete set of different information. There are usually both fake news and real news that debunk these fake ones. As a result, there is a big information variance among all those webpages' titles, which is captured by the cross-lingual cross-platform features. Therefore, TFG performs much better than baselines in a number of events, such as Event 03 (Sochi Olympics). The statistics of the CCMR dataset, described in Table 1, also supports our observation that the labels of posts in CCMR Google are distributed more evenly compared to other media sources.

However, the F1-score of TFG is very low in Event 15 (Garissa Attack). Gunmen stormed the Garissa University College in this event. We analyzed the Google webpages' titles that share the same image in this event. Although some titles are related to the event, more of them are talking about completely unrelated information such as "Daily Graphic News Sun 20th Oct, 2013 — GhHeadlines Total News ...". This webpage's title only shows its published date and the name of the website. Such noise hurt the performance of the cross-lingual cross-platform features. Since we did not perform any manual labeling or filtering, sometimes the crawled webpages can be misleading. To analyze the prevalence of such noise, we randomly picked 100 Google webpages and 100 Baidu webpages from CCMR and counted the number of noisy posts. The ratio of noise is 22% in Google and 18% in Baidu. However, even with such noise, our proposed methods can still outperform current state-of-the-art methods.

7 Rumor Verification Leveraging Cross-lingual Information

We tested if our cross-lingual cross-platform features are able to leverage external information from another language for rumor verification. We simply replaced the Google webpages with Baidu webpages to extract features for tweets (TFB), because we have a pre-trained multilingual sentence embedding that can project Chinese and English to a shared embedding space. We used the same classifier, MLP to evaluate the performance of TFB with both baselines and TFG.

7.1 Results

Experiment results using the *task* setting are shown in Table 2 and the detailed per-event results are listed in Table 3. Similar to the problem in TFG, we can not obtain any Chinese webpages related to events such as Syrian boy, Varoufakis and zdf, which cover most tweets in the test set. Those missing features make TFB perform poorly in the *task* setting. However, TFB performs better than two of the baselines in the *event* setting. If we exclude events without Baidu webpages (event 10, 16 and 17), the average F1-score of UoS, MCG and CER are 0.130, 0.732 and 0.660, which are all lower than TFB's. The performance of TFB proves that our method can be generalized across languages or platforms.

To further test the robustness of our cross-lingual cross-platform features, we also examined if it would still work when leveraging external information that contains different languages. We extracted the cross-lingual cross-platform features for tweets leveraging Google and Baidu webpages together (Combo) and accessed the performance of Combo using MLP similarly. The performance of Combo is also listed in Table 2. Since Combo would contain noise introduced from combining webpages indexed by different search engines, it is not surprising that Combo performs slightly worse than TFG extracted from Google webpages which already cover a wide range of information solely. However, Combo performs much better than TFB which only leverages Baidu webpages. It proves that our cross-lingual cross-platform features are robust enough to utilize combined external information from different languages and platforms.

7.2 Analysis

We checked the actual titles of webpages from CCMR in certain events to analyze the reason for TFB's worse performance exhaustively. We found that those Baidu webpages' titles often talk about subtopics different from the target rumor's on Twitter, while Google webpages are more related. For example, the F1-score of TFB is much lower than TFG's in event 02 (Boston Marathon bombing). This performance corresponds to our observation in Section 3 that Baidu webpages mainly focus on a subtopic related to the Chinese student instead of other things discussed on Twitter.

8 Low-resource Rumor Verification via Transfer Learning

We extracted cross-lingual cross-platform features of webpages in CCMR Baidu leveraging information from Google (BFG). Then we applied Transfer, MLP in Section 6 trained on the whole CCMR Twitter using TFG, to verify those webpages. Because this pre-trained model is for binary classification, only webpages labeled as real or fake in CCMR Baidu are involved.

Since webpages in CCMR Baidu do not share the same features with tweets, such as the number of likes and retweets, we adopted a random selection model as our baseline. It would predict a rumor as real or fake with the same probability. We compared the performance of the Transfer model with this baseline on each event. F1-score is also used for evaluation metric.

8.1 Results

Table 5 lists the detailed results of our transfer learning experiment. We achieved much better performance compared to the baseline with statistical significance ($p<0.001$), which indicates that our cross-lingual cross-platform feature set can be generalized to rumors in different languages. It enables the trained classifier to leverage the information learned from one language to another.

ID	Event	Random	Transfer
01	Hurricane Sandy	0.247	**0.287**
02	Boston Marathon bombing	0.230	**0.284**
03	Sochi Olympics	0.555	**0.752**
04	MH flight 370	0.407	**0.536**
05	Bring Back Our Girls	0.500	**0.923**
06	Columbian Chemicals	0.000	**0.100**
07	Passport hoax	0.000	0.000
08	Rock Elephant	0.000	**0.500**
09	Underwater bedroom	0.577	**0.972**
10	Livr mobile app	-	-
11	Pig fish	0.375	**1.00**
12	Solar Eclipse	0.571	**0.889**
13	Girl with Samurai boots	0.559	**0.925**
14	Nepal Earthquake	**0.227**	0.211
15	Garissa Attack	**0.125**	0.059
16	Syrian boy	-	-
17	Varoufakis and zdf	-	-
	Avg	0.312	**0.531**

Table 5: Rumor verification performance on the CCMR Baidu, where - indicates there is no webpage in that event.

8.2 Analysis

In event 11 (Pig fish), Transfer achieves much higher performance than the random baseline. Generally, Baidu webpages' titles are semantically different from tweets. However, in this particular event, the textual information of those titles and tweets are semantically close. As a result, models learned from English rumors can easily work on Chinese rumors, which is helpful for our transfer learning. Figure 4 shows three Twitter-Baidu rumor pairs with similar meaning in this event.

Transfer obtains pretty low F1-scores in event 07 (Passport hoax). The annotation conflict caused its weak performance. This event is about a Child drew all over his dads passport and made his dad stuck in South Korea. During the manual annotation process, we found out that it is a real event confirmed by official accounts according to one news article from Chinese social media[3], while CCMR Twitter labeled such tweets as fake. Since Transfer is pre-trained using Twitter dataset, it is not surprising that Transfer achieves 0 in F1-score on this event. The annotation conflict also brings out that rumor verification will benefit from utilizing cross-lingual and cross-platform information.

---

[3] http://new.qq.com/cmsn/20140605/20140605002796

Twitter-Baidu Parallel Rumors

1 New species of fish found at Arkansas

潮汕某海边抓到一只人脸怪鱼,眼睛鼻子嘴巴清晰可见…

2 Southern Alabama pigfish, tastes just like bacon!

你们吃过猪鱼吗?吃过吗吃过吗~_哈尔滨吧_百度贴吧

3 Pigfish

猪嘴鱼_湖北工程职业学院吧_百度贴吧

Figure 4: Example parallel rumors in the *Pig fish* event.

9 Conclusion

We created a new multimedia rumor verification dataset by extending a multimedia Twitter dataset with external webpages from Google and Baidu that share similar image content. We designed a set of cross-lingual cross-platform features that leverage the similarity and agreement between information across different platforms and languages to verify rumors. The proposed features are compact and generalizable across languages. We also designed a neural network based model that utilizes the cross-lingual cross-platform features and achieved state-of-the-art results in automatic rumor verification.

References

Christina Boididou, Katerina Andreadou, Symeon Papadopoulos, Duc-Tien Dang-Nguyen, Giulia Boato, Michael Riegler, and Yiannis Kompatsiaris. 2015a. Verifying multimedia use at mediaeval 2015. In *MediaEval*.

Christina Boididou, Symeon Papadopoulos, Duc-Tien Dang-Nguyen, Giulia Boato, and Yiannis Kompatsiaris. 2015b. The certh-unitn participation@ verifying multimedia use 2015. In *MediaEval*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193.

Sarah Cohen, James T Hamilton, and Fred Turner. 2011. Computational journalism. *Communications of the ACM*, 54(10):66–71.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. *arXiv preprint arXiv:1704.05972*.

Nicholas Diakopoulos, Munmun De Choudhury, and Mor Naaman. 2012. Finding and assessing social media information sources in the context of journalism. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2451–2460. ACM.

Nicholas DiFonzo and Prashant Bordia. 2007. Rumor, gossip and urban legends. *Diogenes*, 54(1):19–35.

Omar Enayet and Samhaa R El-Beltagy. 2017. Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474.

Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, and Jiebo Luo. 2017. Multimodal fusion with recurrent neural networks for rumor detection on microblogs. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 795–816. ACM.

Zhiwei Jin, Juan Cao, Yazi Zhang, and Yongdong Zhang. 2015. Mcg-ict at mediaeval 2015: Verifying multimedia use with a two-level classification model. In *MediaEval*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Christopher Matthews. 2013. How does one fake tweet cause a stock market crash. *Wall Street & Markets: Time*.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. 2010. Twitter under crisis: Can we trust what we rt? In *Proceedings of the first workshop on social media analytics*, pages 71–79. ACM.

Stuart Middleton. 2015. Extracting attributed verification and debunking reports from social media: Mediaeval-2015 trust and credibility analysis of image and video.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics.

Husrev T Sencar and Nasir Memon. 2009. Overview of state-of-the-art in digital image forensics. In *Algorithms, Architectures and Information Systems Security*, pages 325–347. World Scientific.

Chengcheng Shao, Giovanni Luca Ciampaglia, Alessandro Flammini, and Filippo Menczer. 2016. Hoaxy: A platform for tracking online misinformation. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 745–750. International World Wide Web Conferences Steering Committee.

Kate Starbird, Jim Maddock, Mania Orand, Peg Achterman, and Robert M Mason. 2014. Rumors, false flags, and digital vigilantes: Misinformation on twitter after the 2013 boston marathon bombing. *iConference 2014 Proceedings*.

Liang Tian, Derek F Wong, Lidia S Chao, Paulo Quaresma, Francisco Oliveira, and Lu Yi. 2014. Um-corpus: A large english-chinese parallel corpus for statistical machine translation. In *LREC*, pages 1837–1842.

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 651–662. IEEE.

Markos Zampoglou, Symeon Papadopoulos, and Yiannis Kompatsiaris. 2015. Detecting image splicing in the wild (web). In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–6. IEEE.

Extractive Adversarial Networks: High-Recall Explanations for Identifying Personal Attacks in Social Media Posts

Samuel Carton
University of Michigan
scarton@umich.edu

Qiaozhu Mei
University of Michigan
qmei@umich.edu

Paul Resnick
University of Michigan
presnick@umich.edu

Abstract

We introduce an adversarial method for producing high-recall explanations of neural text classifier decisions. Building on an existing architecture for extractive explanations via hard attention, we add an adversarial layer which scans the residual of the attention for remaining predictive signal. Motivated by the important domain of detecting personal attacks in social media comments, we additionally demonstrate the importance of manually setting a semantically appropriate "default" behavior for the model by explicitly manipulating its bias term. We develop a validation set of human-annotated personal attacks to evaluate the impact of these changes.

1 Introduction

The task of explaining classifier decisions has recently attracted increased attention from the research community. It is important for several reasons, including: 1) The increasing performance gap between simple-and-interpretable models and complex-but-opaque models (which demand more sophisticated explanation techniques); 2) The increasing ubiquity of machine learning in business and government and the concomitant need to understand the decisions of models in high-stakes situations; and 3) A rising awareness of the limitations of machine learning and the need for ways to better utilize intrinsically unreliable models (whose weaknesses can potentially be ameliorated by good explanations).

A common way to explain why a model classified an example a certain way is to extract a sparse subset of features that were responsible for the model's decision, sometimes described as a saliency mask or "rationale" in the case of text (Guidotti et al., 2018). This type of local explanation may not completely elucidate why a given example is assigned a given outcome, but it does simplify the relationship by identifying what attributes were considered in the decision.

Existing work on this topic has not explicitly addressed the problem of local feature redundancy. That is, when two features are equally predictive of an outcome, which of them should be included in the saliency mask for that decision? Typical sparsity constraints encourage minimal sufficient masks–unveiling just enough of the example to justify the outcome.

There are domains, however, where it may be important to produce complete explanations rather than minimal explanations. One example is the task of detecting content in online social media that violates a platform's policies. Explanatory models can potentially help human moderators make quicker and more consistent decisions about whether to remove comments (Lakkaraju et al., 2016). However, we propose that truly minimal explanations are liable to give only a partial portrait of why a comment is objectionable, making it harder to render a fair holistic decision. If used to explain to a poster why their post was removed, a minimal explanation can actually be misleading, by implying that some of what was objectionable about their post was benign just because it didn't add marginal signal to the overall classification.

We use an extractive explanatory neural network to identify which social media comments contain personal attacks and which words in those comments are the basis for classifying them as containing personal attacks. We train this model on a large dataset (Wulczyn et al., 2017) of comments labeled for the presence of such attacks, and use the explanatory capacity of the model to identify spans that constitute personal attacks within those comments. We extend the work of (Lei et al., 2016) in using one recurrent neural net (RNN) to produce an explanatory hard-attention rationale

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3497–3507
Brussels, Belgium, October 31 - November 4, 2018.

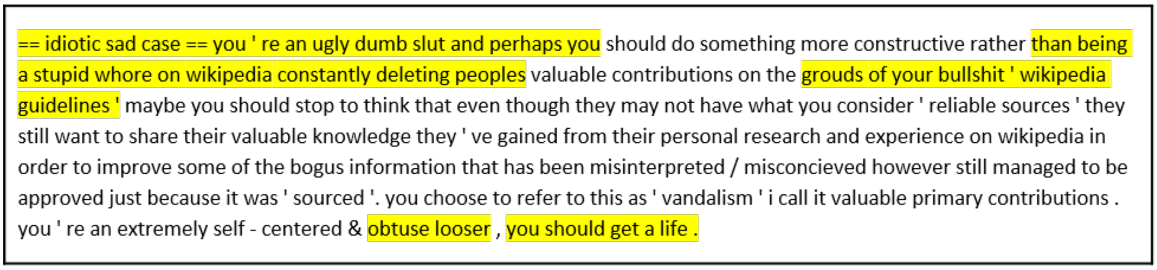

== idiotic sad case == you ' re an ugly dumb slut and perhaps you should do something more constructive rather than being a stupid whore on wikipedia constantly deleting peoples valuable contributions on the grouds of your bullshit ' wikipedia guidelines ' maybe you should stop to think that even though they may not have what you consider ' reliable sources ' they still want to share their valuable knowledge they ' ve gained from their personal research and experience on wikipedia in order to improve some of the bogus information that has been misinterpreted / misconcieved however still managed to be approved just because it was ' sourced '. you choose to refer to this as ' vandalism ' i call it valuable primary contributions . you ' re an extremely self - centered & obtuse looser , you should get a life .

Figure 1: An example of a highly-attacking comment from the test set, rationalized by the model

and a second RNN to make a prediction, the two models trained in an end-to-end fashion.

To produce complete (i.e high-recall) explanations, we add to this existing architecture a second, adversarial predictive layer whose purpose is to try to make predictions based on what is left out of the rationale. We then add a term to the attention layer objective function which encourages it to fool this secondary predictive layer into making poor predictions by including all predictive signal (i.e personal attacks) in the mask that it generates.

We also show that manipulating the model bias term to set a semantically appropriate "default behavior" or "null hypothesis" for the model significantly improves performance. That is, by explicitly choosing what output a zero-information, empty explanation should correspond to, the model is able to learn explanations that correspond more closely with human-generated data.

To summarize, the contributions of this paper are as follows:

- We articulate explanation as an adversarial problem and introduce an adversarial scheme for extraction of complete (high-recall) explanations for text classifier decisions.

- We demonstrate the value of explicitly setting a default output value in such an explanatory model via bias term manipulation.

- We apply explanatory machine learning for the first time to the task of detecting personal attacks in social media comments, and develop a validation dataset for this purpose.

2 Related work

2.1 Online abuse

Online abuse (of which personal attacks are a major dimension) has recently attracted increased attention as a computational problem. Scholarly work has assessed the prevalence and impact of such abuse (Lenhart et al., 2016; Anderson et al., 2014; Pew, 2016; Anderson et al., 2016), while several initiatives have sought to construct datasets for its study (Wulczyn et al., 2017; Abbott et al., 2016; Kennedy et al., 2017; Napoles et al., 2017; Golbeck et al., 2017).

Naturally, much recent work has gone into the use of machine learning to detect online abuse and its perpetrators (Nobata et al., 2016; Pavlopoulos et al., 2017; Cheng et al., 2015), including a workshop at the most recent ACL conference (Association for Computational Linguistics, 2017). However, the idea of fully-automated moderation by machine learning has attracted criticism as being subject to bias, inaccuracy, manipulation and frustration on the user end (Pavlopoulos et al., 2017; Binns et al., 2017; Blackwell et al., 2018; Hosseini et al., 2017; Adams and Dixon, 2017). We propose interpretable models as one potential solution to some of these problems.

2.2 Interpretable machine learning

Major points of division in the interpretability literature include: 1) local vs. global interpretability; 2) post-hoc vs. built-in interpretability; 3) explanation type; 4) input data type; and 5) evaluation metric. Our model is a built-in, feature-based locally interpretable model for text that we evaluate

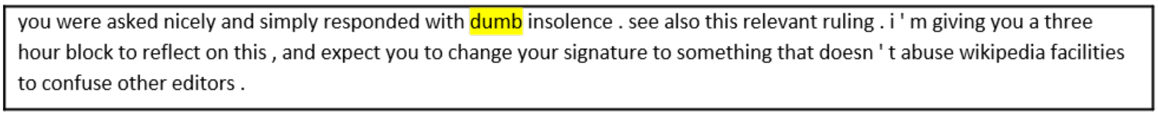

you were asked nicely and simply responded with dumb insolence . see also this relevant ruling . i ' m giving you a three hour block to reflect on this , and expect you to change your signature to something that doesn ' t abuse wikipedia facilities to confuse other editors .

Figure 2: An example of a not-very-attacking example from the test set, rationalized by the model

relative to a human gold-standard. Guidotti et al. (2018) provides a recent survey of the field.

Recent work on interpretability has focused on local (i.e. instancewise) feature-based explanations. Attention models implicitly produce this type of explanation in the form of attention weights over input features. Lei et al. (2016) utilizes a regularized hard attention mechanism to identify the locally minimum sufficient subset of tokens to make accurate predictions.

Post-hoc methods seek to retroactively probe the behavior of an existing non-explanatory model. These include model-specific gradient-based attribution methods, pioneered by Simonyan et al. (2013) and reviewed recently by Ancona et al. (2017), which have tended to originate in the image classification domain and transfer to other domains such as text (e.g. Arras et al. (2017)). Conversely, LIME (Ribeiro et al., 2016) is a prominent recent model-agnostic work in this space, building local linear approximations of a model and using the coefficients thereof to explain its behavior. Li et al. (2016) trains a hard attention layer to flip the decisions of an existing model.

While feature-based explanations are the most common approach, other forms have been proposed, including: similarity to learned "prototypes" which represent clusters of items from the training data (Li et al., 2017); high-precision feature interaction rules (Ribeiro et al., 2018); reference to predefined human-friendly concepts (Kim et al., 2017); and generated natural language (Ehsan et al., 2017). Likewise, many evaluation criteria have been proposed. These include fully automated evaluation (Arras et al., 2017); comparison to human gold standards (Lei et al., 2016); and human task performance (Nguyen, 2018). Doshi-Velez and Kim (2017) and Gilpin et al. (2018) both present reviews and taxonomies of evaluation types.

2.3 Adversarial learning

Generative Adversarial Networks (Goodfellow et al., 2014) involve the use of a discriminative model to help a generative model match its output to an existing data distribution via an adversarial minimax game. Such models have achieved good results on various generative tasks such as image synthesis (Zhang et al., 2018) and text generation (Yu et al., 2016). Recently, adversarial schemes have begun to be adapted for non-strictly-generative tasks such as fake review detection (Aghakhani et al., 2018), improving the robustness of predictive models to adversarial attacks (Zhao et al., 2017) and image retrieval (Song, 2017).

Ideas similar to the adversarial scheme used in this paper have come not from interpretability but rather from weakly-supervised object localization. Wei et al. (2017) uses a similar scheme to accomplish more complete detection of object shapes in images by iteratively erasing the regions that a predictive model lends the most attention, and forcing it to adjust to the occluded image. Edwards and Storkey (2016) is also similar, using an autoencoder rather than a masking layer to remove identifying information from images.

3 Model

The goal of our architecture is to highlight personal attacks in text when such are present, and to highlight little or nothing when there are none, while also performing accurate overall prediction.

These requirements prompt two important edge cases: first, there may be no particular predictive signal in the comment text (i.e. no personal attacks); in a more typical explanatory setting there is always assumed to be some explanation for a decision. Second, there may be redundant signal (i.e. multiple personal attacks), more than is strictly required for accurate prediction, and we assume that it is desirable to identify all of it. We address both of these cases with modifications to the original model architecture.

The model (Figure 3A) is a hard attention architecture which uses one RNN to extract an attention mask of either 0 or 1 for each token, and a different RNN to make a prediction from the attention-masked text (detailed in Figure 3B). Following (Lei et al., 2016), we refer to the mask-producing layer g as the *generator*, but for clarity we call the predictive layer f the *predictor* rather than the encoder. Again following previous work, we refer to the output z of the generator as the *rationale*, in that it rationalizes the prediction of the predictor. We also refer to the inverse rationale, defined as 1-z , as the *antirationale*.

To this basic two-layer scheme, we add a secondary, adversarial predictor f_2, which views the text masked by the antirationale rather than the rationale. The secondary predictor's role is to act as an adversarial discriminator–it tries to make accurate predictions on the antirationale, while the

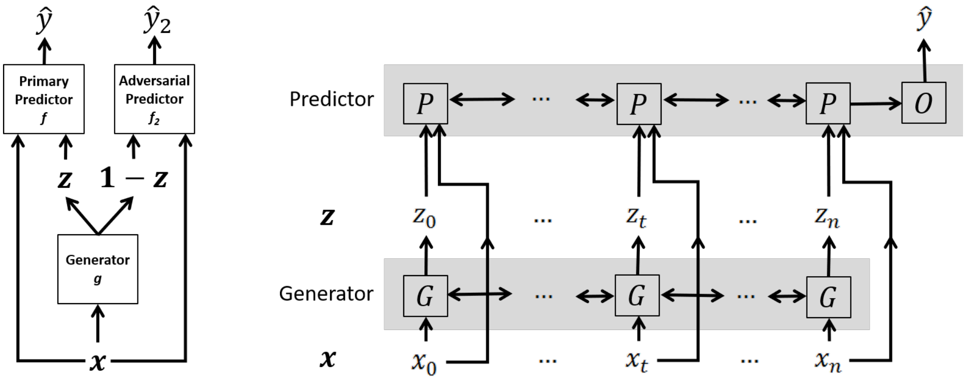

(A) **(B)**

Figure 3: (A) Overall architecture. Generator and predictors are RNNs; (B) Detail of interaction between generator and one predictor layer. G and P are recurrent units of any kind. O is a sigmoid output layer.

generator tries to prevent it from doing so, which ensures that all predictive signal ends up in the rationale.

3.1 Primary predictor

The primary predictor f is an RNN which views the input text masked by the rationale produced by the generator. Its objective is simply to reduce its own squared loss:

$$cost_f(z, x, y) = \left[f(x, z) - y\right]^2 \tag{1}$$

3.1.1 Default behavior via predictor bias term manipulation

The default behavior of the model is the prediction the predictor makes if the input is entirely masked by the rationale: $f(x, 0)$. When working with a recurrent unit that has no internal bias term, this behavior is entirely determined by the bias term of the final sigmoid output layer, $\sigma(wx + b)$, which with typical random initialization of b results in a default predicted value of roughly 0.5.

However, this 0.5 default value is not always optimal or semantically appropriate to the predictive task. In the personal attack detection task, if no attacks can be detected, the “natural” default target value for a text should be close to 0. We show in the experiments that manually setting the output layer bias term b to $logit(0.05) = -2.94$, so that the default predicted value is 0.05, improves model performance.

3.2 Secondary adversarial predictor

The secondary adversarial predictor is an RNN which views the input text masked by the antirationale, defined as 1 minus the rationale z. Its purpose is to encourage high-recall explanations by trying to make accurate predictions from the antirationale, while the generator tries to prevent it from doing so.

However, if the adversarial predictor’s objective function were simply $\left[f_2(x, 1 - z) - y\right]^2$, it would be able to gain an unfair advantage from the presence of masking in the antirationale. Seeing evidence of ”blanked-out” tokens would tell it that personal attacks were present in that comment, giving it strong hint that the target value is close to 1.0 and vice-versa (see figure 4A).

To take away this advantage, the input to the adversarial predictor has to be permuted such that the mask itself is no longer correlated with the target value, while still allowing it to scan the antirationale for residual predictive signal.

Our solution is to replace the masks of half the items in a training batch with the masks of other items in the batch. We order the batch by target value. If item x_i is selected for replacement, it gets the mask of item x_{N-i} where N is the size of the batch. We call this permutation function c:

$$c(z_i) = c(g(x_i)) = \begin{cases} g(x_i) & \text{if } k_i = 1 \\ g(x_{N-i}) & \text{if } k_i = 0 \end{cases}$$

$$x_i \in \{x_0, ..., x_N\} \qquad k_i \sim Bernoulli(0.5)$$

This ensures that low-target-value items get masks associated with high target values and vice-versa, to maximize the dissociation between masks and target values. Figure 4B demonstrates an example of such permutation. This may slow down the learning, since the adversarial predictor

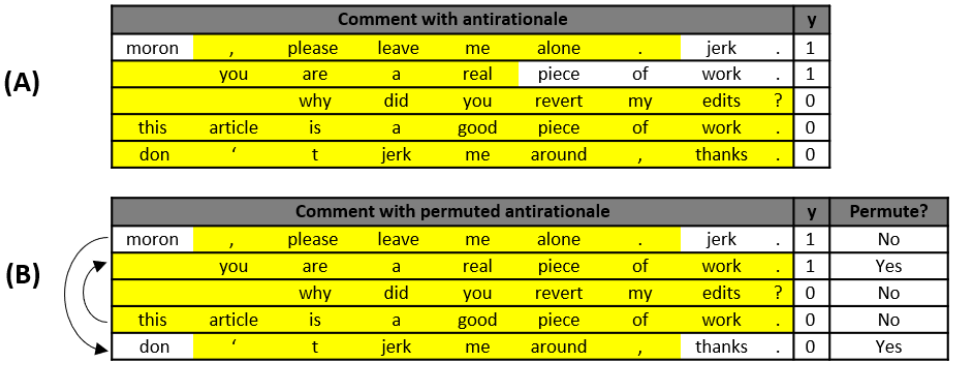

(A)

Comment with antirationale									y
moron	,	please	leave	me	alone	.	jerk	.	1
	you	are	a	real	piece	of	work	.	1
		why	did	you	revert	my	edits	?	0
this	article	is	a	good	piece	of	work	.	0
don	'	t	jerk	me	around	,	thanks	.	0

(B)

Comment with permuted antirationale									y	Permute?
moron	,	please	leave	me	alone	.	jerk	.	1	No
	you	are	a	real	piece	of	work	.	1	Yes
		why	did	you	revert	my	edits	?	0	No
this	article	is	a	good	piece	of	work	.	0	No
don	'	t	jerk	me	around	,	thanks	.	0	Yes

Figure 4: (A) Fabricated sample batch masked by antirationales. Note the correlation between mask and target; (B) The batch with some antirationales switched with those of other items. The correlation no longer holds.

will sometimes have access to somewhat different features of the input than it will have on the test data, but it should not lead to incorrect learning, since the training data always has the correct label, regardless of the mask.

With $c(1-z)$ as the permuted antirationale resulting from applying this randomization process. The objective for the secondary, adversarial predictor is its predictive accuracy on this permuted antirationale:

$$cost_{f_2}(z,x,y) = \left[f_2(x, c(1-z)) - y\right]^2 \qquad (2)$$

3.3 Generator

Given that the two predictors are trying to minimize error on the rationale and (permuted) antirationale respectively, the objective function for the generator is as follows:

$$\begin{aligned} &cost_g(z,x,y) = && (3)\\ &\quad \left[f(x,z) - y\right]^2 && (3.1)\\ &\quad +\lambda_1 \|z\| && (3.2)\\ &\quad +\lambda_1\lambda_2 \sum_t |z_t - z_{t-1}| && (3.3)\\ &\quad +\lambda_3 \left[f_2(x, 1-z) - f_2(x, 0)\right]^2 && (3.4)\end{aligned}$$

Terms 3.1-3.3 are present in the model of Lei et al. Term 3.1 encourages the generator to allow the primary predictor to make accurate predictions, prevents it from obscuring any tokens that would prevent the predictor from doing so. Term 3.2 encourages the generator to produce minimal rationales; obscuring as many tokens as possible. Term 3.3 encourages rationale coherence by punishing the number of transitions in the rationale; it encourages few contiguous phrases rather than many fragments in the rationale.

In theory, these three terms ensure high precision, selecting the minimal (term 3.2) rationale with sufficient signal for accurate prediction (term 3.1), subject to a coherence constraint (term 3.3).

Term 3.4, which is new, ensures recall by encouraging the adversarial predictor's prediction on the antirationale to be similar to the prediction it would make with no information at all (aka the default value). That is, **the antirationale should contain no predictive signal**. Any personal attacks left out of the rationale would appear in the antirationale, letting the adversarial predictor make a more accurate prediction, which would be penalized by term 3.4.

3.4 Extractive Adversarial Network

In the GAN framework (Goodfellow et al., 2014), a discriminator attempts to accurately classify synthetic examples which a generator is striving to match to the distribution of the true data. In our framework, the adversarial predictor attempts to accurately classify censored examples which the generator is striving to strip of all predictive signal. The discriminator in the GAN framework is trained half on real data, and half on fakes; our adversarial predictor is trained half on correctly-masked items and half on items with permuted masks. Where our framework differs from GAN is instead of generating adversarial examples which are compared to true examples, our architecture extracts a modified example out of an existing example, and so can therefore be described as an Extractive Adversarial Network (EAN).

3.5 Implementation details

For comparability with the original algorithm, we use the same recurrent unit (RCNN) and

REINFORCE-style policy gradient optimization process (Williams, 1992) as Lei et al. (2016) to force the generator outputs to be a discrete 0 or 1. In this framework, the continuous output of the generator on each token is treated as a probability from which the mask is then sampled to produce a discrete value for each token. The gradient across this discontinuity is approximated as:

$$\frac{\partial \mathrm{E}_{z\sim g(x)}[cost_g(z,x,y)]}{\partial\theta_g}$$
$$= \mathrm{E}_{z\sim g(x)}\left[cost_g(z,x,y)\frac{\partial \log p(z|x)}{\partial\theta_g}\right]$$

In theory, one would sample z several times from the generator g to produce a good estimate of the gradient. In practice, we find that a single sample per epoch is sufficient. The predictors f and f_2 are trained as normal, as the error gradient with respect to their parameters is smooth.

We employ a particular hard attention model, but the idea of an adversarial critic is not limited to either hard attention or any particular recurrent unit. In a soft attention setting, our adversarial scheme will actually encourage "harder" attention by encouraging any non-zero attention weight to go to 1.0 (or else the inverse of that weight will leave predictive signal in the anti-explanation).

The attention weights produced by the generator are applied to the predictor at the output rather than the input level. When the recurrent unit P of the predictor operates on a token x_t modified by attention weight z_t, it ingests x_t normally, but depending on z_t it either produces its own output or forwards that of the previous token:

$$P(x_t, z_t) = z_t P_{base}(x_t) \cdot (1 - z_t) P_{base}(x_{t-1})$$

We investigate a similar range of sparsity hyperparameter values as the original model [1]. The weight on the inverse term only matters relative to the model sparsity, as that term cooperates rather than competing with the predictive accuracy term (because it almost never hurts accuracy to add more to the rationale). Therefore we set λ_3 to 1.0 when we want to include the inverse term.

We use Word2Vec (Mikolov et al., 2013) to create input token word vectors and Adam (Kingma and Ba, 2014) for optimization.

4 Data

To train our model of personal attacks, we use the dataset introduced by (Wulczyn et al., 2017), which consists of roughly 100,000 Wikipedia revision comments labeled via crowsourcing for aggression, toxicity and the presence of personal attacks. This dataset includes its own training, development and test set split, which we also use.

To this dataset we add a small validation set of personal attack rationales. 40 undergraduate students used Brat (Stenetorp et al., 2012) to highlight sections of comments that they considered to constitute personal attacks. Comments were sampled in a stratified manner from the development and test sets of the Wulczyn et al. dataset, and each student annotated roughly 150 comments, with each comment viewed by roughly 4 annotators. To calculate gold-standard rationales, we take the majority vote among annotators for each token in each comment. 1089 distinct comments were annotated, split between a development and test set of 549 and 540 examples respectively.

The Krippendorff's alpha on our validation set is 0.53 at the whole-comment level. This value is comparable with that of Wulczyn et al. (2017) (0.45). Agreement at the token level is a lower 0.41, because this includes tokens which are a matter of preference among annotators, such as articles and adverbs, as well as content tokens.

5 Experiments

We show that both modifications to the original algorithm, bias term manipulation and inverse term, increase the tokenwise F1 of the predicted rationales relative to our human-annotated test set. All hyperparameters were tuned to maximize tokenwise F1 on the development set. [2]

5.1 Baselines

We generate six baselines for comparison with our variant of the (Lei et al., 2016) architecture. These include the following:

Sigmoid predictor (logistic regression): Bag-of-words representation with a sigmoid output layer.

RNN predictor: The same sequence model used for the predictor, but with no generator layer.

Mean human performance: The mean tokenwise performance of human annotators measured

---

[1] λ_1=[0.0003, 0.0006, 0.0009, 0.0012, 0.0015, 0.0018, 0.0021], λ_2=[0, 1, 2]

[2] λ_1=0.0006 for variants without inverse term, λ_1=0.0015 for variant with inverse term, λ_2=2 (Tuned for maximum F1 on original model, then held constant for comparability)

Model	Rationale						Prediction		
	Tokenwise			Phrasewise					
	F1	Pr.	Rec.	F1	Pr.	Rec.	MSE	Acc.	F1
Sigmoid predictor	-	-	-	-	-	-	0.029	0.94	0.74
RNN predictor	-	-	-	-	-	-	0.018	0.95	0.78
Mean human performance	0.55	0.62	0.57	0.72	0.78	0.69	-	-	-
Sigmoid predictor + feature importance	0.20	**0.62**	0.12	**0.64**	**0.59**	0.70	0.029	0.94	0.74
RNN predictor + sigmoid generator	0.29	0.22	0.45	0.31	0.19	0.92	0.038	0.91	0.70
RNN predictor + LIME	0.33	0.29	0.39	0.4	0.25	**0.96**	0.018	0.95	0.78
Lei2016	0.44	0.38	0.52	0.51	0.38	0.83	0.021	0.95	0.77
Lei2016 + bias	0.49	0.48	0.49	0.60	0.46	0.86	0.02	0.95	0.77
Lei2016 + bias + inverse (EAN)	**0.53**	0.48	**0.58**	0.61	0.47	0.87	0.021	0.95	0.77

Table 1: Rationale performance relative to human annotations. Prediction accuracy is based on a binary threshold of 0.5. Performance of both Lei2016 model variants is significantly different from the baseline model (McNemar's test, $p < 0.05$)

against the majority vote for the comments they annotated (with their vote left out).

Sigmoid predictor + feature importance: Bag-of-words representation with sigmoid output layer, with post-hoc feature importance based on model coefficients. Cutoff threshold for features tuned to maximize rationale F1 on development set.

RNN predictor + sigmoid generator: Rationale mask generated by sigmoid layer applied independently to each input token. Prediction layer is same as predictor.

RNN predictor + LIME: Rationale mask generated by applying LIME (Ribeiro et al., 2016) post-hoc to RNN layer predictions. Masking threshold tuned to maximize rationale F1.

5.2 Rationale performance

In the main experiment, we evaluate model rationales relative to rationales created by human annotators. In our validation dataset, human annotators typically chose to annotate personal attacks at the phrase level; hence in the sentence "Get a job, you hippie s***bag", the majority-vote rationale consists of the entire sentence, where it could arguably consist of the last two or even the last word. Therefore, in addition to tokenwise precision, recall and positive F1, we also report a relaxed "phrasewise" version of these metrics where any time we capture part of a contiguous rationale chunk, that is considered a true positive.

We report results for the original model (i.e. terms 3.1-3.3 in the objective function), the original model with its bias term set for a default value of 0.05, and the bias-modified model with the additional inverse term (term 3.4). For every model variant, we optimized hyperparameters for tokenwise F1 on the development set. We also report results for the baselines described above.

Table 1 displays the results. The difference in performance between the three baselines that don't use a RNN generator and the three model variants that do demonstrates the importance of context in recognizing personal attacks within text. The relative performance of the three variants of the Lei et al. model show that both modifications, setting the bias term and the addition of the adversarial predictor, lead to marginal improvements in tokenwise F1. The best-performing model approaches average human performance on this metric.

The phrasewise metric is relaxed. It allows a contiguous personal attack sequence to be considered captured if even a single token from the sequence is captured. The results on this metric show that in an absolute sense, 87% of personal attacks are at least partially captured by the algorithm. The simplest baseline, which produces rationales by thresholding the coefficients of a logistic regression model, does deceptively well on this metric by only identifying attacking words like "jerk" and "a**hole", but its poor tokenwise performance shows that it doesn't mimic human highlighting very well.

5.3 Original model tokenwise recall

A perplexing result of the rationale performance comparison is how good the tokenwise recall of the model is *without* the inverse term. Without it, the model is encouraged to find the minimal rationale which offers good predictive performance. Comments with more than one personal attack

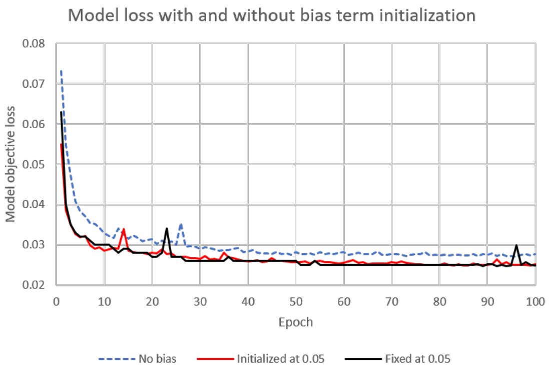

Figure 5: Evolution of model loss over time with and without bias term manipulation

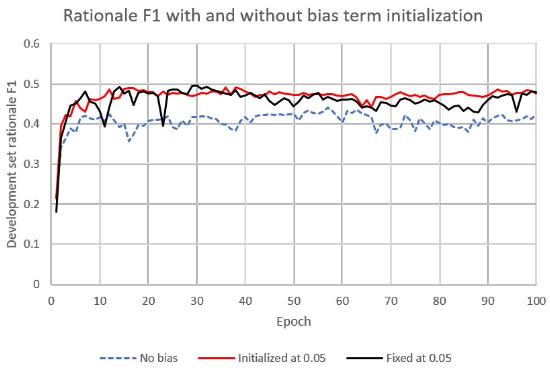

Figure 6: Evolution of development set rationale F1 score over time with and without bias term manipulation

(e.g. Figure 1) constitute 29% of those with at least one attack and 13% of all comments in our validation set. For comments like these, the model should in theory only identify one such attack. However, it tends to find more information than needed, leading to a higher-than-expected recall of .52 in the best overall version of this variant.

To explain this behavior, we run a leave-one-out experiment on the original+bias and original+bias+inverse model variants. For each distinct contiguous rationale chunk predicted by each model (when it generates multi-piece rationales), we try removing this chunk from the predicted rationale, running the prediction layer on the reduced rationale, and seeing whether the result lowers the value of the overall objective function.

For the original+bias model variant, we find that performing this reduction improves the value of the objective function 65% of the time. However, the combined average impact of these reductions on the objective function is to worsen it. What this means is while 65% of distinct phrases discovered by the generator are unnecessary for accurate prediction, the 35% of them that are necessary lead to a major decrease in predictive accuracy.

That is, the generator "hedges its bets" with respect to predictive accuracy by including more information in the rationales than it has to, and experiences a better global optimum as a result. This behavior is less prominent with the inclusion of the inverse term, where the percentage of unnecessary rationale phrases falls to 47%.

5.4 Impact of bias term manipulation

In theory, the model should learn a good bias term for the predictor layer, and therefore the idea of explicitly initializing or fixing the bias term to match the semantics of the task should not impact model performance or represent much of a contribution.

In practice however, as figures 5 and 6 demonstrate, the initialization of the bias term has a big impact on even the long-term learning behavior of the model. Using the best hyperparameters for the original no-bias, no-inverse-term model, figure 5 shows that either initializing or permanently fixing the predictor bias for a default output value of 0.05 leads to improved model loss with respect to its own objective function. Figure 6 shows a similar pattern for tokenwise F1 score.

6 Discussion and future work

One interpretation of the impact of the bias term on model behavior is that an explanation of "why" is really an explanation of "why not"–that is, an explanation is information that distinguishes an item from some alternative hypothesis, and explicitly choosing what this alternative is can improve explanation performance (particularly precision).

Manually setting the model to produce some reasonable default value for an empty rationale makes sense in our setting, but not in domains where there is no default value, such as the beer review dataset of (Lei et al., 2016). A more general approach would be to base explanations on confidence rather than accuracy, where the default value would simply be the mean and variance of the training data, and explanations would consist of tokens that tighten the bounds on the output.

A surprising finding is that the original algorithm often ends up defying its own objective and finds more complete rationales than needed. The leave-one-out experiment described above suggests that the reason for this behavior is that it

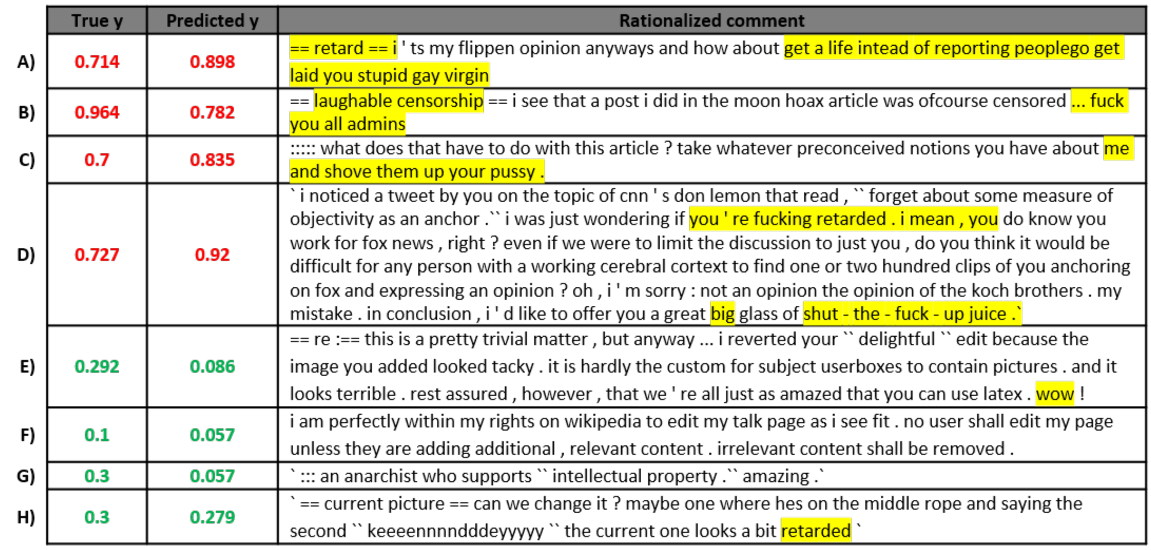

	True y	Predicted y	Rationalized comment
A)	0.714	0.898	== retard == i ' ts my flippen opinion anyways and how about get a life intead of reporting peoplego get laid you stupid gay virgin
B)	0.964	0.782	== laughable censorship == i see that a post i did in the moon hoax article was ofcourse censored ... fuck you all admins
C)	0.7	0.835	::::: what does that have to do with this article ? take whatever preconceived notions you have about me and shove them up your pussy .
D)	0.727	0.92	` i noticed a tweet by you on the topic of cnn ' s don lemon that read , `` forget about some measure of objectivity as an anchor .`` i was just wondering if you ' re fucking retarded . i mean , you do know you work for fox news , right ? even if we were to limit the discussion to just you , do you think it would be difficult for any person with a working cerebral cortext to find one or two hundred clips of you anchoring on fox and expressing an opinion ? oh , i ' m sorry : not an opinion the opinion of the koch brothers . my mistake . in conclusion , i ' d like to offer you a great big glass of shut - the - fuck - up juice .`
E)	0.292	0.086	== re :== this is a pretty trivial matter , but anyway ... i reverted your `` delightful `` edit because the image you added looked tacky . it is hardly the custom for subject userboxes to contain pictures . and it looks terrible . rest assured , however , that we ' re all just as amazed that you can use latex . wow !
F)	0.1	0.057	i am perfectly within my rights on wikipedia to edit my talk page as i see fit . no user shall edit my page unless they are adding additional , relevant content . irrelevant content shall be removed .
G)	0.3	0.057	` ::: an anarchist who supports `` intellectual property .`` amazing .`
H)	0.3	0.279	` == current picture == can we change it ? maybe one where hes on the middle rope and saying the second `` keeeennnndddeyyyyy `` the current one looks a bit retarded `

Figure 7: Further examples of labeled and rationalized comments. Items E) and G) show that the algorithm struggles with sarcasm.

is how the generator deals with predictive uncertainty, and that it achieves a better global optimum by producing locally suboptimal rationales.

While this "bug" proves useful in our case, it may not generalize. In our setting the adversarial predictor gives a modest improvement in recall; it will produce a larger improvement in settings where the unaltered algorithm is more successful at producing the minimal explanations described by its objective function. Li et al. (2016) finds that a memory network predictor requires less occlusion than an LSTM to flip its predictions, indicating that choice of model can effect completeness of explanations.

In theory, interpretable models can aid human moderaters by pointing them directly at the potentially objectionable content in a comment and giving them a starting point for making their own holistic decision about the comment. However, there are potential pitfalls. Adding explanations as a model output gives the model another way to be wrong–one which humans may be even less able to troubleshoot than simple misclassification. Relatedly, explanations may inspire overconfidence in model predictions. Extensive user testing would clearly be needed before any deployment.

One final concern is the question of whether human-like explanations are really optimal explanations. Are high-recall explanations that mimic human highlighting tendencies really optimal for the types of moderating/self-moderating tasks involved in the domain of personal attacks in online social media? Again, this question can only be answered with human subject experimentation, which we plan to approach in future work.

7 Conclusion

The main contribution of this paper is to frame explanation as an adversarial problem, thereby addressing explanation recall for the first time that we are aware of. We do so by introducing an adversarial framework (an "extractive adversarial network") for ensuring that redundant predictive signal is not omitted from a model's explanations. We also show that choosing a null hypothesis for the model by setting the model bias term improves explanation precision.

Secondarily, we make a domain-specific contribution by applying interpretable machine learning for the first time to the problem of identifying personal attacks in social media comments, with the hope of developing more transparent semi-automated moderation systems. We show that we approach human performance on a dataset we develop for this purpose.

Acknowledgement

This material is based upon work supported by the National Science Foundation under grant numbers 1717688, 1633370 and 1620319. We also thank David Jurgens and Yue Wang for their helpful comments and suggestions.

References

Robert Abbott, Brian Ecker, Pranav Anand, and Marilyn Walker. 2016. Internet Argument Corpus 2.0: An SQL Schema for Dialogic Social Media and the Corpora to Go With It. In *Language Resources and Evaluation Conference (LREC 2016)*.

CJ Adams and Lucas Dixon. 2017. Better discussions with imperfect models. *The False Positive - Medium*. [Blog post].

Hojjat Aghakhani, Aravind Machiry, Shirin Nilizadeh, Christopher Kruegel, and Giovanni Vigna. 2018. Detecting Deceptive Reviews using Generative Adversarial Networks. *arXiv:1805.10364 [cs]*. ArXiv: 1805.10364.

Marco Ancona, Enea Ceolini, Cengiz ztireli, and Markus Gross. 2017. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. *arXiv:1711.06104 [cs, stat]*. ArXiv: 1711.06104.

Ashley A. Anderson, Dominique Brossard, Dietram A. Scheufele, Michael A. Xenos, and Peter Ladwig. 2014. The Nasty Effect: Online Incivility and Risk Perceptions of Emerging Technologies: Crude comments and concern. *Journal of Computer-Mediated Communication*, 19(3):373–387.

Ashley A. Anderson, Sara K. Yeo, Dominique Brossard, Dietram A. Scheufele, and Michael A. Xenos. 2016. Toxic Talk: How Online Incivility Can Undermine Perceptions of Media. *International Journal of Public Opinion Research*.

Leila Arras, Grgoire Montavon, Klaus-Robert Mller, and Wojciech Samek. 2017. Explaining Recurrent Neural Network Predictions in Sentiment Analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 159–168, Copenhagen, Denmark. Association for Computational Linguistics.

Association for Computational Linguistics. 2017. *Proceedings of the 1st Workshop on Abusive Language Online*. ACL, Vancouver.

Reuben Binns, Michael Veale, Max Van Kleek, and Nigel Shadbolt. 2017. Like Trainer, Like Bot? Inheritance of Bias in Algorithmic Content Moderation. In *Social Informatics*, volume 10540, pages 405–415, Cham. Springer.

Lindsay Blackwell, Jill Dimond, Sarita Schoenebeck, and Cliff Lampe. 2018. Classification and Its Consequences for Online Harassment: Design Insights from HeartMob. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW 2018)*, pages 1–19.

Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Antisocial Behavior in Online Discussion Communities. In *Proceedings of the International Conference on Web and Social Media*, page 10.

Finale Doshi-Velez and Been Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. *arXiv:1702.08608 [cs, stat]*. ArXiv: 1702.08608.

Harrison Edwards and Amos Storkey. 2016. Censoring Representations with an Adversary. In *Proceedings of the International Conference on Learning Representations*.

Upol Ehsan, Brent Harrison, Larry Chan, and Mark O. Riedl. 2017. Rationalization: A Neural Machine Translation Approach to Generating Natural Language Explanations. *arXiv:1702.07826 [cs]*. ArXiv: 1702.07826.

Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining Explanations: An Approach to Evaluating Interpretability of Machine Learning. *arXiv:1806.00069 [cs, stat]*. ArXiv: 1806.00069.

Jennifer Golbeck, Rajesh Kumar Gnanasekaran, Raja Rajan Gunasekaran, Kelly M. Hoffman, Jenny Hottle, Vichita Jienjitlert, Shivika Khare, Ryan Lau, Marianna J. Martindale, Shalmali Naik, Heather L. Nixon, Zahra Ashktorab, Piyush Ramachandran, Kristine M. Rogers, Lisa Rogers, Meghna Sardana Sarin, Gaurav Shahane, Jayanee Thanki, Priyanka Vengataraman, Zijian Wan, Derek Michael Wu, Rashad O. Banjo, Alexandra Berlinger, Siddharth Bhagwan, Cody Buntain, Paul Cheakalos, Alicia A. Geller, and Quint Gergory. 2017. A Large Labeled Corpus for Online Harassment Research. In *Proceedings of the 2017 ACM on Web Science Conference*, pages 229–233. ACM Press.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Riccardo Guidotti, Anna Monreale, Franco Turini, and Dino Pedreschi. 2018. A Survey Of Methods For Explaining Black Box Models. *arXiv preprint arXiv:1802.01933*.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving Googles Perspective API Built for Detecting Toxic Comments. *arXiv preprint arXiv:1702.08138*.

George Kennedy, Andrew McCollough, Edward Dixon, Alexei Bastidas, John Ryan, Chris Loo, and Saurav Sahay. 2017. Technology Solutions to Combat Online Harassment. In *Proceedings of the First Workshop on Abusive Language Online*, pages 73–77. Association for Computational Linguistics.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. 2017. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). *arXiv:1711.11279 [stat]*. ArXiv: 1711.11279.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations*.

Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. 2016. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1675–1684. ACM Press.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing Neural Predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117.

Amanda Lenhart, Michele Ybarra, Kathryn Zickuhr, and Myeshia Price-Feeney. 2016. Online Harassment, Digital Abuse, and Cyberstalking in America. Technical report, Data & Society Research Institute.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding Neural Networks through Representation Erasure. *arXiv preprint arXiv:1612.08220*.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. 2017. Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network that Explains Its Predictions. *arXiv:1710.04806 [cs, stat]*. ArXiv: 1710.04806.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Courtney Napoles, Joel Tetreault, Aasish Pappu, Enrica Rosato, and Brian Provenzale. 2017. Finding Good Conversations Online: The Yahoo News Annotated Comments Corpus. In *Proceedings of the 11th Linguistic Annotation Workshop*.

Dong Nguyen. 2018. Comparing Automatic and Human Evaluation of Local Explanations for Text Classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1069–1078.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deep Learning for User Comment Moderation. In *Proceedings of the First Workshop on Abusive Language Online*, pages 25–35.

Pew. 2016. The Political Environment on Social Media. Technical report, Pew Research Center.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High Precision Model-Agnostic Explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, page 9.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv:1312.6034 [cs]*. ArXiv: 1312.6034.

Jingkuan Song. 2017. Binary Generative Adversarial Networks for Image Retrieval. *arXiv:1708.04150 [cs]*. ArXiv: 1708.04150.

Pontus Stenetorp, Sampo Pyysalo, Goran Topi, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107.

Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. 2017. Object Region Mining with Adversarial Erasing: A Simple Classification to Semantic Segmentation Approach. *arXiv:1703.08448 [cs]*. ArXiv: 1703.08448.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *AAAI Conference on Artificial Intelligence*.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2018. Self-Attention Generative Adversarial Networks. *arXiv:1805.08318 [cs, stat]*. ArXiv: 1805.08318.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2017. Generating Natural Adversarial Examples. *arXiv:1710.11342 [cs]*. ArXiv: 1710.11342.

Automatic Detection of Vague Words and Sentences in Privacy Policies

Logan Lebanoff and **Fei Liu**
Department of Computer Science
University of Central Florida, Orlando, FL 32816, USA
`loganlebanoff@knights.ucf.edu` `feiliu@cs.ucf.edu`

Abstract

Website privacy policies represent the single most important source of information for users to gauge how their personal data are collected, used and shared by companies. However, privacy policies are often vague and people struggle to understand the content. Their opaqueness poses a significant challenge to both users and policy regulators. In this paper, we seek to identify vague content in privacy policies. We construct the first corpus of human-annotated vague words and sentences and present empirical studies on automatic vagueness detection. In particular, we investigate context-aware and context-agnostic models for predicting vague words, and explore auxiliary-classifier generative adversarial networks for characterizing sentence vagueness. Our experimental results demonstrate the effectiveness of proposed approaches. Finally, we provide suggestions for resolving vagueness and improving the usability of privacy policies.

S1	We *may* use the *information* automatically collected from your computer or *other devices* for the following uses... (Vagueness: 3.8)
S2	In addition, in *some* cases the Sites can deliver content based on your current location if you choose to enable that feature. (Vagueness: 2.25)
S3	Our Sites and Services *may*, *from time to time*, provide links to sites operated by *third parties*. (Vagueness: 3.2)
S4	To customize and serve advertising and *other* marketing communications that *may* be visible to you on our Sites and Services or *elsewhere* on the internet. (Vagueness: 4)
S5	This includes your credit card number, income level, or *any other* information that would *normally* be considered confidential. (Vagueness: 3)

Table 1: Example human-annotated vague words and sentences. Vague words are *italicized*. Averaged sentence vagueness is given in the parentheses. Higher score is more vague.

1 Introduction

Website privacy policies are difficult to read and people struggle to understand the content. Recent studies (Sadeh et al., 2013) have raised concerns over their opaqueness, which poses a considerable challenge to both Internet users and policy regulators. Nowadays, consumers supply their personal information to online websites in exchange for personalized services; they are surrounded by smart gadgets such as voice assistants and surveillance cameras, which constantly monitor their activities in the home and work environments. Without clearly specifying how users' information will be collected, used and shared, there is a substantial risk of information misuse, including undesired advertisements and privacy breaches. Especially with recent high-profile cases involving Facebook and Cambridge Analytica, the public is becoming more aware and concerned with how their information is handled.

Privacy policies are binding agreements between companies and users that stipulate how companies collect, use, and share users' personal information. They are lengthy and difficult to read. Bhatia et al. (2016) suggested two possible causes for this. First, privacy policies must be *comprehensive* in order to cover a variety of uses (e.g., in-store and online purchases). Second, the policies have to be *accurate* to all data practices and systems. Clearly, it would be difficult for a company's legal counsel to anticipate all future needs. They need to resort to vague language to describe the content, causing it to be difficult to read and compromising the effectiveness of privacy policies.

In this paper, we present the first study on automatic detection of vague content in website privacy policies. We construct a sizable corpus containing word- and sentence-level human annotations of vagueness for privacy policy documents. The corpus contains a total of 133K words and

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3508–3517
Brussels, Belgium, October 31 - November 4, 2018.

4.5K sentences. Our methods for automatically detecting vague words and sentences are based on deep neural networks, which have demonstrated impressive recent success. Specifically, we investigate context-aware and context-agnostic models for predicting word vagueness, where feature representations of words are built with and without considering their surrounding words. By this, we seek to verify the hypothesis that vagueness is an intrinsic property of words and has little to do with context. To understand sentence vagueness, we explore auxiliary-classifier generative adversarial networks (AC-GAN, Odena et al., 2018). The model has performed strongly on vision tasks (e.g., image synthesis), however, whether it can be adapted to handle text data has not been thoroughly investigated. We train the AC-GAN model to discriminate between real/fake privacy policy sentences while simultaneously classifying sentences exhibiting different levels of vagueness, including "clear," "somewhat clear," "vague," and "extremely vague," thus improving the model's generalization capabilities. The detected vague words and sentences can assist users in browsing privacy policy documents, and privacy regulators in assessing the clarity of privacy policy practices. Our research contributions include the following:

- we present the first study on automatic detection of vague content in privacy policies. Vague content compromises the usability of privacy policies and there is an urgent need to identify and resolve vagueness;
- we construct a sizable text corpus including human annotations for 133K words and 4.5K sentences of privacy policy texts. The data[1] is available publicly to advance research on language vagueness; and
- we investigate both context-aware and context-agnostic methods for predicting vague words. We also explore the auxiliary-classifier generative adversarial networks for characterizing sentence vagueness. This is the first study leveraging deep neural networks for detecting vague content in privacy policies.

2 Related Work

Privacy policies are often verbose, difficult to read, and perceived as ineffective (McDonald and Cranor, 2008). In particular, vague language in these documents hurts understanding. *"A term is regarded as vague if it admits borderline cases, where speakers are reluctant to say either the term definitely applies or definitely does not apply,"* a definition of vagueness quoted from (van Deemter, 2010). Legal scholars and language philosophers strive to understand vagueness from a theoretical perspective (Keefe, 2000; Shapiro, 2006). The "sorites paradox" describes the phenomenon of vagueness (Keefe, 2000). It states that small changes in the object do not affect the applicability of a vague term. For example, a room can remain "bright" even if the light is dimmed little by little until it is entirely extinguished, thus creating a paradox. Hyde (2014) further suggests that vagueness is a feature pertaining to multiple syntactic categories. Nouns, adjectives and adverbs (e.g., "child", "tall", "many") are all susceptible to reasoning. These studies often focus on linguistic case studies but not on developing resources for automatic detection of vagueness.

Recent years have seen a growing interest in using natural language processing techniques to improve the effectiveness of website privacy policies. Sadeh et al. (2013) describe a Usable Privacy Policy Project that seeks to semi-automate the extraction of salient details from privacy policies. Other studies include crowdsourcing privacy policy annotations and categorizing data practices (Ammar et al., 2012; Massey et al., 2013; Wilson et al., 2016b,a), grouping text segments related to certain policy issues (Liu et al., 2014; Ramanath et al., 2014), summarizing terms of services (Braun et al., 2017), identifying user opt-out choices (Sathyendra et al., 2017), and many others. These studies emphasize the "too long to read" issue of privacy policies but leave behind the "difficult to understand" aspect, such as identifying and eliminating vague content.

The work of (Liu et al., 2016) is close to ours. The authors attempt to learn vector representations of words in privacy policies using deep neural networks, where the vectors encode not only semantic/syntactic aspects but also vagueness of words. The model is later fed to an interactive visualization tool (Strobelt et al., 2016) to test its ability to discover related vague terms. While promising, their approach is not fully automatic, and the feasibility of detecting vague words and sentences in an automatic manner is still left untested.

In this work we conduct the first study to auto-

---

[1] `https://loganlebanoff.github.io/data/vagueness_data.tar.gz`

matically detect vague content from privacy policies. We ask human annotators to label vague words and sentences and train supervised classifiers to do the same. Classifying vague words is a challenging task, because vagueness is an understudied property and it spans multiple syntactic categories (e.g., "usually," "personal data," "necessary"). Neural network classifiers such as CNN and LSTM have demonstrated prior success on text classification tasks (Zhang and Wallace, 2015), but whether they can be utilized to identify vague terms is not well understood.

For sentence classification, we investigate auxiliary classifier generative adversarial networks (AC-GAN, Odena et al., 2018). GANs have seen growing popularity in recent years (Mirza and Osindero, 2014; Yu et al., 2016; Li et al., 2017; Gu et al., 2018; Cai and Wang, 2018). AC-GAN is a variant of GAN that generates word sequences using class-conditional probabilities. E.g., it generates "fake" privacy policy sentences exhibiting different degrees of vagueness (e.g., "clear," "vague," "extremely vague"). AC-GAN nicely combines real (human-annotated) and fake (synthetic) privacy policy sentences in a discriminative framework to improve the model's generalization capabilities. This can be equated to a semi-supervised learning paradigm through augmentation of the dataset with generated sentences. Data augmentation is particularly valuable for vagueness detection, which generally has small expensive datasets. We perform a full analysis on AC-GAN and compare it to state-of-the-art systems.

3 The Corpus

Annotating vague words and sentences is a non-trivial task. We describe our effort to select privacy policy sentences for annotation, recruit qualified workers, and design annotation guidelines.

We select 100 website privacy policies from the collection gathered by Liu et al. (2014). The documents are quite lengthy, containing on average 2.3K words. More importantly, most content is not vague. To obtain a more balanced corpus, a filtering step is used to select only sentences that have a moderate-to-high chance of containing vague content. Fortunately, Bhatia et al. (2016) provide a list of 40 cue words for vagueness, manually compiled by policy experts. We therefore retain only sentences containing one of the cue words for further annotation. A brief examination shows that most of the sentences removed from the corpus are indeed clear. Even with this bias, the resulting corpus still contains a small portion of clear sentences (See Figure 1). The reason is that a cue word can be used in a way that is not vague. For example, in the sentence "Users *may* post to our website," the word *may* indicates permission but not possibility, and therefore the sentence is not vague.

Vague Term	**Freq.**	**Vague Term**	**Freq.**
may	1,575	other information	30
personal info.	465	non-personal info.	30
information	302	sometimes	27
other	261	reasonably	26
some	214	appropriate	25
certain	205	necessary	24
third parties	183	certain info.	23
third party	134	typically	22
pers. iden. info.	88	affiliates	21
time to time	75	reasonable	20
most	54	non-personal	19
generally	52	personally iden.	18
personal data	52	such as	18
third-party	49	usually	17
others	41	personal	16
general	39	may be	15
many	37	content	14
various	36	otherwise	14
might	35	periodically	14
services	33	similar	14

Table 2: The most frequent vague terms identified by human annotators and their frequencies in our corpus. "pers.," "iden." and "info." are shorthand for "personally," "identifiable" and "information."

Reidenberg et al. (2015) discuss attempts to use crowdsourced workers as a cost-effective alternative to policy experts for annotating privacy policies. In this study, we hire crowd workers from the Amazon Mechanical Turk platform. To recruit quality workers, we require them to reside in the U.S. and be proficient in English; they are skilled workers maintaining a task success rate of 90% or above. We provide example labelled vague terms obtained from the case studies described in Bhatia et al. (2016) to reduce discrepancies among workers. The annotators are then asked to use their best judgment to perform the task.

Given a privacy policy sentence, the annotators are instructed to identify all vague terms[2] and assign a score of vagueness to the sentence. A vague term is limited to be 5 words or less (e.g., "including but not limited to"). We use this rule to prevent annotators from tagging an entire sentence/clause as vague. A slider is provided in the interface to al-

---

[2] We use "term" to denote either a single word or a phrase.

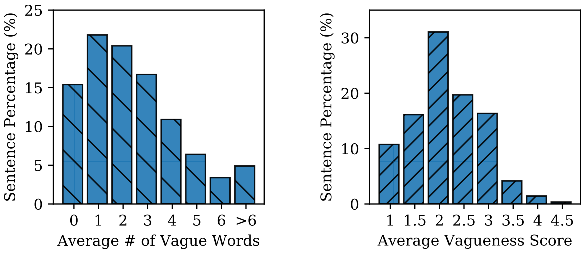

Figure 1: (Left) Percentage of sentences containing different numbers of vague words. (Right) Perc. of sentences with different levels of vagueness. 1 is clear, 5 is extremely vague.

Sent: This includes your credit card number , income level , or any other information that would normally be considered confidential .

Annotator 1: any, other, normally
Annotator 2: any other information
Annotator 3: normally, confidential, any other

Ground Truth Labels: [This]$_0$ [includes]$_0$ [your]$_0$ [credit]$_0$ [card]$_0$ [number]$_0$, [income]$_0$ [level]$_0$, [or]$_0$ ***[any]***$_1$ ***[other]***$_1$ [information]$_0$ [that]$_0$ [would]$_0$ ***[normally]***$_1$ [be]$_0$ [considered]$_0$ [confidential]$_0$.

Table 3: Ground truth labels are obtained by consolidating human-annotated vague terms; "any," "other," "normally" are labelled 1 because they are selected by 2 or more annotators.

low annotators to select a vagueness score for the sentence: 1 is extremely clear and 5 is extremely vague. We design a human intelligence task (HIT) to include 5 privacy policy sentences and a worker is rewarded $0.05 for completing the task. Five human workers are recruited to perform each task.

We obtain annotations for 133K words and 4.5K sentences. The average sentence vagueness score is 2.4±0.9. As of inter-annotator agreement[3], we find that 47.2% of the sentences have their vagueness scores agreed by 3 or more annotators; 12.5% of the sentence vagueness scores are agreed by 4 or more annotators. Furthermore, the annotators are not required to select vague words if they believe the sentences are clear. We remove vague words selected by a single annotator. Among the rest, 46.1% of the words are selected by 3 or more annotators; 18.5% of the words are selected by 4 or more annotators. These results suggest that, although annotating vague terms and sentences is considered challenging, our annotators can reach a reasonable degree of agreement. We present example vague terms in Table 2. Note that we obtain a total of 1,124 unique vague terms, which go well beyond the 40 cue words used for sentence preselection. Figure 1 shows more statistics on sentence vagueness, including (i) the percentages of sentences containing different numbers of vague words, and (ii) the percentages of sentences whose vagueness scores fall in different ranges.

4 Word Vagueness

We seek to test an important hypothesis related to word vagueness. We conjecture that vagueness is an intrinsic property of words; whether a word is vague or not has little to do with its context words. To verify this hypothesis, we build context-aware and context-agnostic models to classify each word in a privacy policy sentence as either vague or non-vague. The ground-truth labels are obtained by consolidating human annotations (see Table 3 for an example). A word is labelled 1 if it is selected by two or more annotators, otherwise 0. We describe details of the two classifiers below.

Context-aware classifier. It builds feature representations of words based on the surrounding context words. Given its strong performance, we construct a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) for this purpose. A word is replaced by its word2vec embedding (Mikolov et al., 2013) before it is fed to the model. For each time step, we concatenate the hidden states obtained from the forward and backward passes and use it as input to a feedforward layer with sigmoid activation to predict if a word is vague or non-vague. Because single words consist of the majority of the human-annotated vague terms, we choose to use binary word labels instead of a BIO scheme (Chiu and Nichols, 2016) for sequence tagging. Figure 2 shows the architecture.

Context-agnostic classifier. It uses intrinsic feature representations of words without considering the context. Specifically, we represent a word using its word2vec embedding, then feed it to a feedforward layer with sigmoid activation to obtain the prediction (Figure 2). We train the classifier using a list of unique words obtained from the training data; a word is considered positive if it has a ground truth label of 1 in any sentence, otherwise negative. Note that the ratio of positive/negative unique words in our corpus is 1068/3176=0.34. At test time, we apply the binary classifier to each word of the test set. A word is assigned the same label regardless of which sentence it appears in. We adopt this setting to ensure the context-aware and context-agnostic results are comparable.

5 Sentence Vagueness

We next investigate how vagueness is manifested in privacy policy sentences. Our goal is to assign a

---

[3] We choose not to calculate a kappa statistic, as labelling vague words/sentences is not a clear-cut classification task.

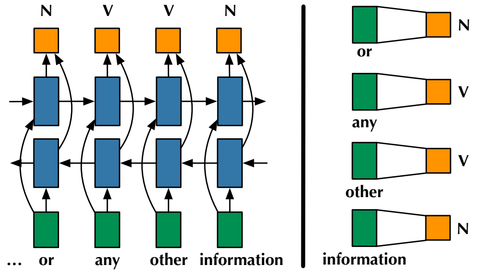

Figure 2: (Left) Context-aware word classifier implemented as a bidirectional LSTM. (Right) Context-agnostic classifier. “V” and “N” are shorthands for “vague” and “non-vague.”

label to each sentence indicating its level of vagueness. We derive ground truth sentence labels by averaging over vagueness scores assigned by human annotators, and further discretizing the scores into four buckets: [1,2), [2,3), [3,4), [4,5], respectively corresponding to “clear,” “somewhat clear,” “vague,” and “extremely vague” categories. The sentences in the four buckets respectively consist of 26.9%, 50.8%, 20.5%, and 1.8% of the total annotated sentences. We choose to predict discrete labels instead of continuous scores because labels are more informative to human readers. E.g., a label of “extremely vague” is more likely to trigger user alerts than a score of 4.2.

5.1 Auxiliary-Classifer GAN

Predicting vague sentences is a nontrivial task due to the complexity and richness of natural language. We propose to tackle this problem by exploring the auxiliary classifier generative adversarial networks (AC-GAN, Odena et al., 2018). We choose GAN because of its ability to combine text generation and classification in a unified framework (Yu et al., 2016; Li et al., 2017; Gu et al., 2018). Privacy policy sentences are particularly suited for text generation because the policy language is restricted and a text generator can effectively learn the patterns. AC-GAN has a great potential to make use of both human-annotated data and “fake” augmented data for classification. The system architecture is presented in Figure 3. The generator learns to generate “fake” privacy policy sentences and sentences exhibiting different levels of vagueness using class conditional probabilities (hence the name auxiliary-classifer GAN). The discriminator learns to discriminate among real/fake sentences as well as sentences of different levels of vagueness. They are jointly trained using a heuristic, non-saturating game loss. In the following we present the model details.

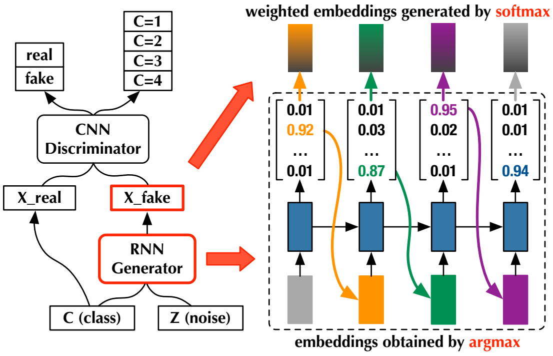

Figure 3: System architecture for AC-GAN. (Left) The Generator generates plausible privacy policy sentences (X_fake). The Discriminator must learn to differentiate between real and fake sentences as well as predicting the vagueness category (C) of the sentences. (Right) RNN-generator. A vocabulary distribution is produced for each step. Gumbel-softmax is applied to the distributions to calculate weighted embeddings to be used by the Discriminator (arrows pointing up). Argmax is applied to the distributions to retrieve embeddings to be passed to the next step (arrows pointing down).

5.2 Sentence Generator

The generator focuses on generating “fake” samples that resemble privacy policy sentences of a given vagueness category. This is denoted by $P(X|C)$, where $X = \{x_t\}_{t=1}^{T}$ is a sequence of words and $C \in \{1,2,3,4\}$ is a vagueness category. A vagueness category is randomly sampled in the generation process, and the generator attempts to generate a sentence of that vagueness level. A typical RNN text generator unrolls the sequence X one word at a time until an end-of-sentence symbol (EOS) is reached. At time step t, it samples a word x_t from a vocabulary-sized vector of probability estimates $P(x_t)$:

$$x_t \sim P(x_t) = \text{softmax}(\mathbf{a}_t), \tag{1}$$

$$\mathbf{a}_t = \mathbf{W}\mathbf{h}_t + \mathbf{b}, \tag{2}$$

$$\mathbf{h}_t = f_{RNN}(\mathbf{h}_{t-1}, x_{t-1}), \tag{3}$$

where $\mathbf{a}_t$ is a vector of activation values and $\mathbf{h}_t$ is the t-th RNN hidden state. We train a neural text generator, implemented as Long Short-Term Memory networks (Hochreiter and Schmidhuber, 1997), on a large collection of privacy policy sentences using cross-entropy loss. While generating natural language sentences is successfully tackled by recurrent neural networks, the generated sentences are not necessarily vague. Training the generator only on vague sentences is impractical because there is a limited number of annotated sentences. In this paper we introduce a new way of defining class conditional probabilities:

$$x_t \sim P(x_t|C) = \text{softmax}(\mathbf{a}_t + \lambda_C \mathbf{v}), \quad (4)$$

where $\mathbf{v}$ is a vocabulary-sized, trainable vector indicating how likely a vocabulary word is vague. λ_C is a coefficient for vagueness category C. The underlying assumption is that a "clear" sentence is less likely to contain vague words (λ_C is negative), whereas an "extremely vague" sentence tends to contain many vague words (λ_C is positive).

Finally, the generated "fake" sentences, together with "real" sentences labelled by human annotators, are fed to the discriminator for training a classifier discriminating between real/fake sentences and sentences of different vagueness levels. Nevertheless, there remains a critical issue with the current system: we cannot backpropagate through discrete samples X. As a result, the generator parameters cannot be properly updated using backpropagation. To circumvent this issue, we attempt the reparameterization trick with Gumbel-Softmax relaxation (Gu et al., 2018).

Straight-Through Gumbel-Softmax. Two competing issues exist in the RNN generator. First, the discriminator requires a continuous form for each generated word to keep the entire model differentiable. Second, the generator requires a discrete choice for each word to generate a sentence, rather than propagating "partial words" through the sequence. To solve this problem, the *softmax* distribution over vocabulary words is sent to the discriminator, while the *argmax* over the distribution is sent to the next step of the generator. This system is referred to as Straight-Through Gumbel.

We explain the process of calculating the softmax distribution to send to the discriminator. To simulate the random-sampling process, the approach applies reparameterization to shift randomness from sampling a discrete variable x_t (Eq. (4)) to sampling a continuous noise vector $\mathbf{z}_t$ following the Gumbel distribution (Eq. (5)). The noise vector is added to the activation $\mathbf{a}_t + \lambda_C \mathbf{v}$ to compute the *argmax* (Eq. (6)). To simulate the argmax operation, a temperature parameter τ is applied to *softmax* (Eq. (7)), where small values of τ greatly skew the distribution, causing it to peak at the largest value, while still remaining differentiable. Similar reparameterization is also used for variational auto-encoders (Kingma and Welling, 2014).

$$\mathbf{z}_t \sim \text{Gumbel}(z) \quad (5)$$

$$x_t = \text{argmax}(\mathbf{a}_t + \mathbf{z}_t + \lambda_C \mathbf{v}) \quad (6)$$

$$P(x_t|C) = \text{softmax}(\frac{\mathbf{a}_t + \mathbf{z}_t + \lambda_C \mathbf{v}}{\tau}) \quad (7)$$

The generator requires a discrete word to propagate to the next time step of the RNN. The word with the maximum activation value is chosen as shown in (Eq. (6)). An illustration of ST Gumbel is presented in Figure 3.

5.3 Sentence Discriminator

A sentence discriminator learns to perform two tasks simultaneously. Given a privacy policy sentence X, it predicts a probability distribution over its sources, denoted by $P(S|X)$, where S = {real, fake}; and a probability distribution over its level of vagueness, denoted by $P(C|X)$, C = {clear, somewhat clear, vague, extremely vague}. The learning objective for the discriminator is to maximize the log-likelihood of making correct predictions on both tasks, denoted by $L_C + L_S$, where L_C and L_S are defined in Eq. (8) and (9).

$$L_C = \mathbb{E}[\log P(C = c|X_{real+fake})] \quad (8)$$

$$\begin{aligned} L_S &= \mathbb{E}[\log P(S = real|X_{real})] \\ &\quad + \mathbb{E}[\log P(S = fake|X_{fake})] \end{aligned} \quad (9)$$

The ground truth vagueness labels C for real sentences are annotated by human annotators. For fake sentences the labels are randomly sampled in the generation process; and conditioned on the sampled vagueness labels, fake sentences are generated using $P(x_t|C)$ (Eq. (7)).

$$L'_C = \mathbb{E}[\log P(C = c|X_{fake})] \quad (10)$$

$$L'_S = \mathbb{E}[\log P(S = real|X_{fake})] \quad (11)$$

The *generator* is trained to maximize $L'_C + L'_S$ as illustrated in Eq. (10-11). Intuitively, the generator is rewarded (or punished) only based on the "fake" samples it produces. It is rewarded by generating sentences correctly exhibiting different levels of vagueness, denoted by (L'_C). It is also rewarded by generating sentences that look "real" and cannot be easily distinguished by the discriminator (L'_S). Eq. (11) corresponds to a heuristic, non-saturating game loss that mitigates gradient saturation (Goodfellow, 2016).

We experiment with two variants of the discriminator, implemented respectively using the convolutional neural networks (CNN) (Zhang and Wallace, 2015) and LSTM (Hochreiter and Schmid-

System	Word-Level		
	P (%)	R (%)	F (%)
Context-Agnostic	11.30	**78.15**	19.71
Context-Aware	**68.39**	53.57	**60.08**

Table 4: Results of detecting vague words in privacy policies using context-aware and context-agnostic classifiers.

huber, 1997). In both cases, the discriminator assigns a source and a vagueness label to each sentence. The CNN discriminator scans through each sentence using using a sliding window and apply a number of filters to each window. A max pooling over the sequence is performed to create a feature map for the sentence. This feature map is treated as the sentence representation. It is fed to two separate dense layers with softmax activation to predict $P(C|X)$ and $P(S|X)$ respectively. In contrast, the LSTM discriminator runs a forward pass through the sentence and uses the last hidden state as the sentence representation. Similarly, this representation is fed to two dense layers used to predict $P(C|X)$ and $P(S|X)$. Both methods produce probability estimations using a shared sentence representation. Given the scarcity of labelled sentences, this multitask setting is expected to improve the model's generalization capabilities.

6 Experiments

We conduct experiments on the annotated corpus using a 5-fold cross validation; 10% of the training data in each fold are reserved for validation. In the following sections we present details of experimental settings and report results on detecting vague words and sentences in privacy policy texts.

6.1 Parameter Settings

The Xavier scheme (Glorot and Bengio, 2010) is used for parameter initialization. For the context-aware word classifier, the bidirectional LSTM has 512 hidden units. For AC-GAN, the CNN discriminator uses convolutional filters of size $\{3, 4, 5\}$ and 128 filters for each size. The LSTM generator and discriminator both have 512 hidden units. The generator is further pretrained on 82K privacy policy sentences using a 10K vocabulary. The coefficient λ_C is set to $\{-1, 0, 1, 2\}$ respectively for 'clear,' 'somewhat clear,' 'vague,' and 'extremely vague' categories. $\mathbf{v}$ is initialized as a binary vector, where an entry is set to 1 if it is one of the 40 cue words for vagueness (Bhatia et al., 2016). Word embeddings are initialized to their word2vec embeddings and are made trainable during the entire training process.

S1	... while we use **[reasonable]**$_{tp}$ **[efforts]**$_{fn}$ to protect your PII, we can not guarantee its absolute security.
S2	We use **[third-party]**$_{fn}$ advertising companies to serve **[some]**$_{tp}$ of the ads when you visit our web site.
S3	The **[information]**$_{fn}$ we obtain from **[those services]**$_{fn}$ **[often depends]**$_{fp}$ on your settings or their privacy policies, so be sure to check what those are.
S4	In the event of an insolvency, bankruptcy or receivership, **[personal data may]**$_{tp}$ also be transferred as a business asset.

Table 5: Examples of detected vague words in privacy policies. $[\cdot]_{tp}$ denotes true positive, $[\cdot]_{fp}$ is false positive, $[\cdot]_{fn}$ is false negative. All unmarked words are true negatives.

False Alarms		Misses	
POS Tag	Perc. (%)	POS Tag	Perc. (%)
Adjective	**37.19**	Noun	**47.64**
Noun	35.24	Adjective	25.07
Verb	20.53	Verb	13.31
Adverb	4.63	Adverb	5.62
Determiner	1.72	Determiner	2.79

Table 6: The most frequent part-of-speech (POS) tags appeared in false alarms and misses of detected vague words.

6.2 Predicting Vague Words

We compare context-aware with context-agnostic classifiers on detecting vague words in privacy policy text. The goal is to test an important hypothesis: that vagueness is an intrinsic property of words, thus a word being vague has little to do with its context. Results are presented in Table 4.

Interestingly, context-agnostic classifier yields a high recall score (78.15%) despite it ignoring context. This result indicates word vagueness can be encoded in distributed word embeddings. However, the low precision (11.30%) suggests that context is important for fine-grained analysis. While it is possible for experts to create a comprehensive list of vague terms for assessing privacy policies, extra effort is required to verify the tagged vague terms. Using a context-aware classifier produces more balanced results, improving the F-score from 19.71% to 60.08%. Our findings suggest that the context information is necessary for detecting vague words.

In Table 5, we present examples of detected vague words. The nouns have caught our attention. The classifier misses several of these, including "efforts," "information," "services," per-

System	Sentence-Level P (%)	R (%)	F (%)
Baseline (Majority)	25.77	50.77	34.19
LSTM	47.79	50.06	47.88
CNN	49.66	52.51	50.18
AC-GAN (Full Model)	51.00	53.50	50.42
AC-GAN (Vagueness Only)	**52.90**	**54.64**	**52.34**

Table 7: Results on classifying vague sentences.

haps because there is no clear definition for these terminologies. In Table 6, we found nouns consist of 47.64% of all the miss-detected vague words, while adjectives consist of 37.19% of the false alarms. There is also an interesting phenomenon. In S3, "Information" and "those services" are considered more vague by humans than "often depends." However, if those terms are removed from the sentence, yielding "The [..] we obtain from [..] often depends on your settings or their privacy policies." In this case, the vagueness of "often depends" become more prominent and is captured by our system. It suggests that the degree of vagueness may be relative, depending on if other terms in the sentence are more vague.

6.3 Predicting Vague Sentences

In Table 7 we present results on classifying privacy policy sentences into four categories: clear, somewhat clear, vague, and extremely vague. We compare AC-GAN with three baselines: CNN and LSTM trained on human-annotated sentences, and a majority baseline that assigns the most frequent label to all test sentences. We observe that the AC-GAN models (using CNN discriminator) perform strongly, surpassing all baseline approaches. CNN shows strong performance, yielding an F-score of 50.92%. A similar effect has been demonstrated on other sentence classification tasks, where CNN outperforms LSTM and logistic regression classifiers (Kim, 2014; Zhang and Wallace, 2015). We report results of AC-GAN using the CNN discriminator. Comparing "Full Model" with "Vagueness Only," we found that allowing the AC-GAN to only discriminate sentences of different levels of vagueness, but not real/fake sentences, yields better results. We conjecture this is because training GAN models, especially with a multitask learning objective, can be unstable and more effort is required to balance the two objectives (L_S and L_C). Example sentences generated by AC-GAN are presented in Table 9.

% (Freq)	Clear	SomeC	Vague	ExtrV
Clear	**39.4** (477)	59.8 (723)	0.7 (8)	0.2 (2)
SomeC	12.4 (284)	**85.2** (1945)	2.4 (54)	0.0 (1)
Vague	3.4 (31)	89.6 (828)	**7.0** (65)	0.0 (0)
ExtrV	1.2 (1)	88.9 (72)	9.9 (8)	**0.0** (0)

Table 8: Confusion matrix for sentence classification. The decimal values are the percentage of system-identified sentences that were placed in the specified vagueness class. For example: the item in (row 1, col 2) conveys that 59.8% of sentences (absolute count is 723) identified by the system as "clear" were actually "somewhat clear" according to humans.

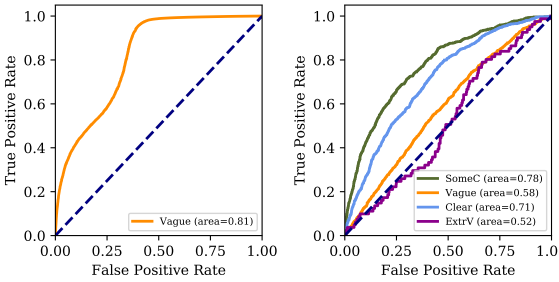

Figure 4: ROC curves for classifying vague words (left) and sentences (right).

Figure 4 shows the ROC curves of the four vagueness classes. Because the dataset is imbalanced, the ROC curves are more informative than F-scores. The "clear" and "somewhat clear" classes yield promising AUC scores of 0.71 and 0.78 respectively. The "vague" and "extremely vague" classes are more challenging. They are also the minority classes, consisting of 20.5% and 1.8% of the annotated data. Confusion matrix in Table 8 reveals that the majority of the sentences are tagged as "somewhat clear," while 7.0% of the vague sentences are tagged as vague. It suggests more annotated data may be helpful to enable the classifier to distinguish "vague" and "extremely vague" sentences. Interestingly, we found there is little correlation between the sentence vagueness score and sentence length (Pearson correlation r=0.18, p <0.001) while there is a relatively strong correlation (r=0.57, p <0.001) between sentence vagueness and the number of vague words in it. This finding verifies our hypothesis that vague words seem to increase the perceived sentence vagueness.

Lessons learned. We summarize some lessons learned from annotating and detecting vague content in privacy policies, useful for policy regulators, users and website operators. In general, privacy policies are suggested to:

- provide clear definitions for key concepts. Lacking definition is a major source of confu-

Clear	Our commitment to travian games uses paid services or send an order online.
	To learn how important anonymization it, we provide a separate medicare.
SomeC	Slate use certain cookies and offers.
	Visitors who apply us an credit card may sign up.
Vague	There may take certain incidents various offerings found on various topics; some or all individual has used.
	You may modify certain edit or otherwise delete certain features or a similar id will no longer.
ExtrV	Also, some apps may offer contests, sweepstakes, games or some community where necessary.
	If necessary, buying or clarify certain links, certain features of our site may place or some or some features may offer stack or unauthorized access some some functionality.

Table 9: Plausible sentences generated by AC-GAN. They exhibit different levels of vagueness. "SomeC" and "ExtrV" are shorthands for "somewhat clear" and "extremely vague."

sion for the unfamiliar reader. Example concepts include personally identifiable information, personal (non-personal) information, third parties, service providers, subsidiaries, etc.

- suppress the use of vague words. There are on average 2.5 vague words per sentence in our corpus. The more vague words, the more likely the sentence is perceived as vague ($r = 0.57$);
- use sentences with simple syntactic structure to ease understanding. A sophisticated sentence with vague terms in it, e.g., "You may request deletion of your personal data by us, but please note that we may be required (by law *or otherwise*) to keep this information and not delete it..." appears especially confusing to readers.

7 Conclusion

In this paper we present the first empirical study on automatic detection of vague content in privacy policies. We create a sizable text corpus including human annotations of vague words and sentences. We further investigate the feasibility of predicting vague words and sentences using deep neural networks. Specifically we investigate context-agnostic and context-aware models for detecting vague words, and AC-GAN for detecting vague sentences. Our results suggest that a supervised paradigm for vagueness detection provides a promising avenue for identifying vague content and improving the usability of privacy policies.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and suggestions, which help to improve this paper. This work is in part supported by an ORC In-House Research Grant.

References

Waleed Ammar, Shomir Wilson, Norman Sadeh, and Noah A. Smith. 2012. Automatic categorization of privacy policies: A pilot study. Technical Report CMU-LTI-12-019, Carnegie Mellon University.

Jaspreet Bhatia, Travis D. Breaux, Joel R. Reidenberg, and Thomas B. Norton. 2016. A theory of vagueness and privacy risk perception. In *Proceedings of the IEEE International Conference on Requirements Engineering (RE)*.

Daniel Braun, Elena Scepankova, Patrick Holl, and Florian Matthes. 2017. SaToS: Assessing and summarising terms of services from german webshops. In *Proceedings of the 10th International Conference on Natural Language Generation (INLG)*.

Liwei Cai and William Yang Wang. 2018. KBGAN: Adversarial learning for knowledge graph embeddings. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

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

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Ian Goodfellow. 2016. NIPS 2016 tutorial: Generative adversarial networks. *arXiv:1701.00160* .

Jiatao Gu, Daniel Jiwoong Im, and Victor O.K. Li. 2018. Neural machine translation with gumbel-greedy decoding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*.

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

Dominic Hyde. 2014. Sorites paradox. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*.

Rosanna Keefe. 2000. *Theories of Vagueness*. Cambridge University Press.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jiwei Li, Will Monroe, Tianlin Shi, Sebastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Fei Liu, Nicole Fella, and Kexin Liao. 2016. Modeling language vagueness in privacy policies using deep neural networks. In *Proceedings of the AAAI Fall Symposium on Privacy and Language Technologies*.

Fei Liu, Rohan Ramanath, Norman Sadeh, and Noah A. Smith. 2014. A step towards usable privacy policy: Automatic alignment of privacy statements. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*.

Aaron K. Massey, Jacob Eisenstein, Annie I. Anton, and Peter P. Swire. 2013. Automated text mining for requirements analysis of policy documents. *Proceedings of the 21st IEEE International Requirements Engineering Conference* .

Aleecia M. McDonald and Lorrie Faith Cranor. 2008. The cost of reading privacy policies. *I/S: A Journal of Law and Policy for the Information Society* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *CoRR* abs/1411.1784.

Augustus Odena, Christopher Olah, and Jon Shlens. 2018. Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the Thirty-fifth International Conference on Machine Learning (ICML)*.

Rohan Ramanath, Fei Liu, Norman Sadeh, and Noah A. Smith. 2014. Unsupervised alignment of privacy policies using hidden markov models. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics (ACL 2014)*.

Joel R. Reidenberg, Travis Breaux, Lorrie Faith Cranor, Brian French, Amanda Grannis, James T. Graves, Fei Liu, Aleecia M. McDonald, Thomas B. Norton, Rohan Ramanath, N. Cameron Russell, Norman Sadeh, and Florian Schaub. 2015. Disagreeable privacy policies: Mismatches between meaning and users' understanding. *Berkeley Law Technology Journal* 30(1).

Norman Sadeh, Alessandro Acquisti, Travis Breaux, Lorrie Cranor, Aleecia McDonald, Joel Reidenberg, Noah Smith, Fei Liu, Cameron Russel, Florian Schaub, and Shomir Wilson. 2013. The usable privacy policy project: Combining crowdsourcing, machine learning and natural language processing to semi-automatically answer those privacy questions users care about. Technical Report CMU-ISR-13-119, Carnegie Mellon University.

Kanthashree Mysore Sathyendra, Shomir Wilson, Florian Schaub, Sebastian Zimmeck, and Norman Sadeh. 2017. Identifying the provision of choices in privacy policy text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Stewart Shapiro. 2006. *Vagueness in Context*. Oxford: Oxford University Press.

Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M. Rush. 2016. Visual analysis of hidden state dynamics in recurrent neural networks. In *arXiv:1606.07461*.

Kees van Deemter. 2010. *Not Exactly: In Praise of Vagueness*. New York: Oxford University Press.

Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N. Cameron Russell, Thomas B. Norton, Eduard Hovy, Joel R. Reidenberg, and Norman Sadeh. 2016a. The creation and analysis of a website privacy policy corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shomir Wilson, Florian Schaub, Rohan Ramanath, Norman Sadeh, Fei Liu, Noah A. Smith, and Frederick Liu. 2016b. Crowdsourcing annotations for websites' privacy policies: Can it really work? In *Proceedings of the 25th International World Wide Web Conference (WWW)*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *http://arxiv.org/abs/1510.03820v4* .

Multi-view Models for Political Ideology Detection of News Articles

Vivek Kulkarni
Department of Computer Science
University of California, Santa Barbara
`vvkulkarni@cs.ucsb.edu`

Junting Ye
Department of Computer Science
Stony Brook University
`juyye@cs.stonybrook.edu`

Steven Skiena
Department of Computer Science
Stony Brook University
`skiena@cs.stonybrook.edu`

William Yang Wang
Department of Computer Science
University of California, Santa Barbara
`william@cs.ucsb.edu`

Abstract

A news article's title, content and link structure often reveal its political ideology. However, most existing works on automatic political ideology detection only leverage textual cues. Drawing inspiration from recent advances in neural inference, we propose a novel attention based multi-view model to leverage cues from all of the above views to identify the ideology evinced by a news article. Our model draws on advances in representation learning in natural language processing and network science to capture cues from both textual content and the network structure of news articles. We empirically evaluate our model against a battery of baselines and show that our model outperforms state of the art by 10 percentage points F1 score.

1 Introduction

Many issues covered or discussed by the media and politicians today are so subtle that even word-choice may require one to adopt a particular ideological position (Iyyer et al., 2014). For example, conservatives tend to use the term `tax reform`, while liberals use `tax simplification`. Though objectivity and unbiased reporting remains a cornerstone of professional journalism, several scholars argue that the media displays ideological bias (Gentzkow and Shapiro, 2010; Groseclose and Milyo, 2005; Iyyer et al., 2014). Even if one were to argue that such bias may not be reflective of a lack of objectivity, prior research Dardis et al. (2008); Card et al. (2015) note that framing of topics can significantly influence policy.

Since manual detection of political ideology is challenging at a large scale, there has been extensive work on developing computational models for automatically inferring the political ideology of articles, blogs, statements, and congressional speeches (Gentzkow and Shapiro, 2010; Iyyer et al., 2014; Preoţiuc-Pietro et al., 2017; Sim et al., 2013). In this paper, we consider the detection of ideological bias at the news article level, in contrast to recent work by Iyyer et al. (2014) who focus on the sentence level or the work of (Preoţiuc-Pietro et al., 2017) who focus on inferring ideological bias of social media users. Prior research exists on detecting ideological biases of news articles or documents (Gentzkow and Shapiro, 2010; Gerrish and Blei, 2011; Iyyer et al., 2014). However, all of these works generally only model the text of the news article. However, in the online world, news articles do not just contain text but have a rich structure to them. Such an online setting influences the article in subtle ways: (a) choice of the title since this is what is seen in snippet views online (b) links to other news media and sources in the article and (c) the actual textual content itself. Except for the textual content, prior models ignore the rest of these cues. Figure 1 shows an example from `The New York Times`. Note the presence of hyperlinks in the text, which link to other sources like `The Intercept`(Figure 1a). We hypothesize that such a link structure is reflective of homophily between news sources sharing similar political ideology – homophily which can be exploited to build improved predictive models (see Figure 1b). Building on this insight, we propose a new model **MVDAM**: *Multi-view document attention model* to detect the ideological bias of news articles by leveraging cues from multiple views: the title, the link structure, and the article content. Specifically, our contributions are:

1. We propose a generic framework **MVDAM** to incorporate multiple views of the news article and show that our model outperforms state of the art by 10 percentage points on the F1 score.
2. We propose a method to estimate the ideological proportions of sources and rank them

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3518–3527
Brussels, Belgium, October 31 - November 4, 2018.

by the degree to which they lean towards a particular ideology.

3. Finally, differing from most works, which typically focus on congressional speeches, we conduct ideology detection of news articles by assembling a *large-scale diverse dataset* spanning more than 50 sources.

2 Related Work

Several works study the detection of political ideology through the lens of computational linguistics and natural language processing (Laver et al., 2003; Monroe and Maeda, 2004; Thomas et al., 2006; Lin et al., 2008; Carroll et al., 2009; Ahmed and Xing, 2010; Gentzkow and Shapiro, 2010; Gerrish and Blei, 2011; Sim et al., 2013). Gentzkow and Shapiro (2010) first attempt to rate the ideological leaning of news sources by proposing a measure called "slant index" which captures the degree to which a particular newspaper uses partisan terms or co-allocations. Gerrish and Blei (2011) predict the voting patterns of Congress members based on supervised topic models. Other works use topic models to analyze bias in news articles, blogs, and political speeches (Ahmed and Xing, 2010; Lin et al., 2008). Sim et al. (2013) propose a novel HMM-based model to infer the ideological proportions of the rhetoric used by political candidates in their campaign speeches which relies on a fixed lexicon of bigrams associated with ideologies.

The work that is most closely related to our work is that of Iyyer et al. (2014); Preoţiuc-Pietro et al. (2017). Iyyer et al. (2014) use recurrent neural networks to predict political ideology of congressional debates and articles in the ideological book corpus (IBC) and demonstrate the importance of compositionality in predicting ideology where modifier phrases and punctuality affect the political ideological position. Preoţiuc-Pietro et al. (2017) propose models to infer political ideology of Twitter users based on their everyday language. Most crucially, they also show how to effectively use the relationship between user groups to improve prediction accuracy. Our work draws inspiration from both of these works but differentiates itself from these in the following aspects: We leverage the structure of a news article by noting that an article is just not free-form text, but has a rich structure to it. In particular, we model cues from the title, the inferred network, and the content in a joint generic neural variational inference framework to yield improved models for this task. Furthermore, differing from Iyyer et al. (2014), we also incorporate attention mechanisms in our model which enables us to inspect which sentences (or words) have the most predictive power as captured by our model. Finally, since we work with news articles (which also contain hyperlinks), naturally our setting is different from all other previous works in general (which mostly focus on congressional debates) and in particular from Iyyer et al. (2014) where only textual content is modeled or Preoţiuc-Pietro et al. (2017) which focuses on social media users.

3 Dataset Construction

News Sources We rely on the data released by ALLSIDES.COM[1] to obtain a list of 59 US-based news sources along with their political ideology ratings: LEFT, CENTER or RIGHT which specify our target label space. While we acknowledge that there is no "perfect" measure of political ideology, ALLSIDES.COM is an apt choice for two main reasons. First, and most importantly the ratings are based on a blind survey, where readers are asked to rate news content without knowing the identity of the news source or the author being rated. This is also precisely the setting in which our proposed computational models operate (where the models have access to the content but are agnostic of the source itself) thus seeking to mirror human judgment closely. Second, these are normalized by ALLSIDES to ensure they closely reflect popular opinion and political diversity present in the United States. These ratings also correlate with independent measurements made by the PEW RESEARCH CENTRE. All these observations suggest that these ratings are fairly robust and generally "reflective of the average judgment of the American People"[2].

Content Extraction Given the set of news sources selected above, we extract the article content for these news sources. We control for time by obtaining article content over a fixed time-period for all sources. Specifically, we spider several news sources and perform data cleaning. In particular, the spidering component collates the raw HTML of news sources into a storage engine (MongoDB). We track thousands of US based news outlets including country wide popular news sources as well as many local/state news based outlets like the

---

[1] https://www.allsides.com/media-bias/media-bias-ratings

[2] https://www.allsides.com/media-bias/about-bias

A Run of Good Fortune

Around the same time Mr. Pruitt invested in the house in Oklahoma City, he had finished a big business deal that involved Mr. Kelly, Mr. Wagner and a campaign donor who ran a large staffing company.

A baseball player in college, Mr. Pruitt bought an approximately 25 percent stake in the Oklahoma City RedHawks and became the team's managing partner, making him a highly visible spokesman for the local team. Mr. Wagner also purchased a small stake, and Mr. Kelly's bank provided financing for the deal, as first reported by The Intercept, which also disclosed the bank's loans for one of Mr. Pruitt's suburban Tulsa homes.

(a) A sample news article. Note the presence of hyperlinks to other sources like The Intercept.

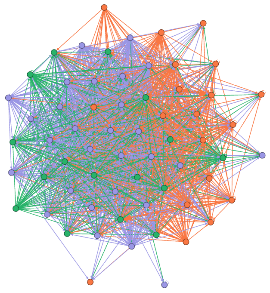

(b) Homophily in link structure (viewed in color) of various news sources which can be observed by noting the presence of clusters corresponding to political ideologies. The blue, orange and green clusters correspond to left, right and center leaning sources respectively.

Figure 1: Our proposed framework MVDAM models multiple views of the news article including the content and the link structure. Figure 1a shows a sample article from the New York Times. The presence of such links can provide informative signals for predictive tasks like ideology detection primarily due to homophily (Figure 1b).

Boston Herald[3]. However, in this paper, we consider only the 59 US news sources for which we can derive ground truth labels for political ideology. For each of the news sources considered, we extract the title, the cleaned pre-processed content, and the hyperlinks within the article that reveal the network structure. The label for each article is the label assigned to its source as obtained from ALLSIDES. We choose a random sample of $120,000$ articles and create 3 independent splits for training $(100,000)$, validation $(10,000)$ and test $(10,000)$ with a roughly balanced label distribution. [4]

Data Pre-processing and Cleaning Since the labels were derived from the source, we are careful to remove any systematic features in each article which are trivially reflective of the source, since that would result in over-fitting. In particular we perform the following operations: (a) **Remove source link mentions** When modeling the link structure of an article, we explicitly remove any link to the source itself. Second, we also explicitly remove any systematic link structures in articles that are source specific. In particular, some sources may always have links to other domains (like their own franchisees or social media sites). These links are removed explicitly by noting their high frequency. (b) **Remove headers, footers, advertisements** News sources systematically introduce footers, and advertisements which we remove explicitly. For example, every article of the `The Daily Beast` has the following footer `You can subscribe to the Daily Beast here` which we filter out.

4 Models and Methods

Problem Formulation Given $\boldsymbol{X} = \{\boldsymbol{X_{title}}, \boldsymbol{X}_{net}, \boldsymbol{X_{content}}\}$ which represents a set of multi-modal features of news articles and a label set $\boldsymbol{Y} = \{\text{LEFT}, \text{CENTER}, \text{RIGHT}\}$, we would like to model $\Pr(\boldsymbol{Y}|\boldsymbol{X})$.

Overview of MVDAM We consider a Bayesian approach with stochastic attention units to effectively model textual cues. Bayesian approaches with stochastic attention have been noted to be quite effective at modeling ambiguity as well as avoiding over-fitting scenarios especially in the case of small training data sets (Miao et al., 2016). In particular, we assume a latent representation $\boldsymbol{h}$ learned from the multiple modalities in $\boldsymbol{X}$ which is then mapped to the label space $\boldsymbol{Y}$. In the most general setting, instead of learning a deterministic encoding $\boldsymbol{h}$ given $\boldsymbol{X}$, we posit a latent distribution over the hidden representation $\boldsymbol{h}$, $\Pr(\boldsymbol{h}|\boldsymbol{X})$ to model the overall document where $\Pr(\boldsymbol{h}|\boldsymbol{X})$ is parameterized by a diagonal Gaussian distribution $\mathcal{N}(\boldsymbol{h}|\mu(\boldsymbol{X}), \sigma^2(\boldsymbol{X}))$.

Specifically, consider the distribution $\Pr(\boldsymbol{Y}|\boldsymbol{X})$ which can be written as follows:

$$\Pr(\boldsymbol{Y}|\boldsymbol{X}) = \sum_{\boldsymbol{h}} \Pr(\boldsymbol{Y}|\boldsymbol{h}) \Pr(\boldsymbol{h}|\boldsymbol{X}) \qquad (1)$$

As noted by Miao et al. (2016), computing the exact posterior is in general intractable. Therefore, we posit a variational distribution $q_\phi(\boldsymbol{h})$ and maximize

---

[3] This is a part of an ongoing project called MediaRank. More details can be found at `http://media-rank.com`

[4] Note that we do not restrict the articles to be strictly political since even articles on other topics like health and sports can be reflective of political ideology (Hoberman, 1977).

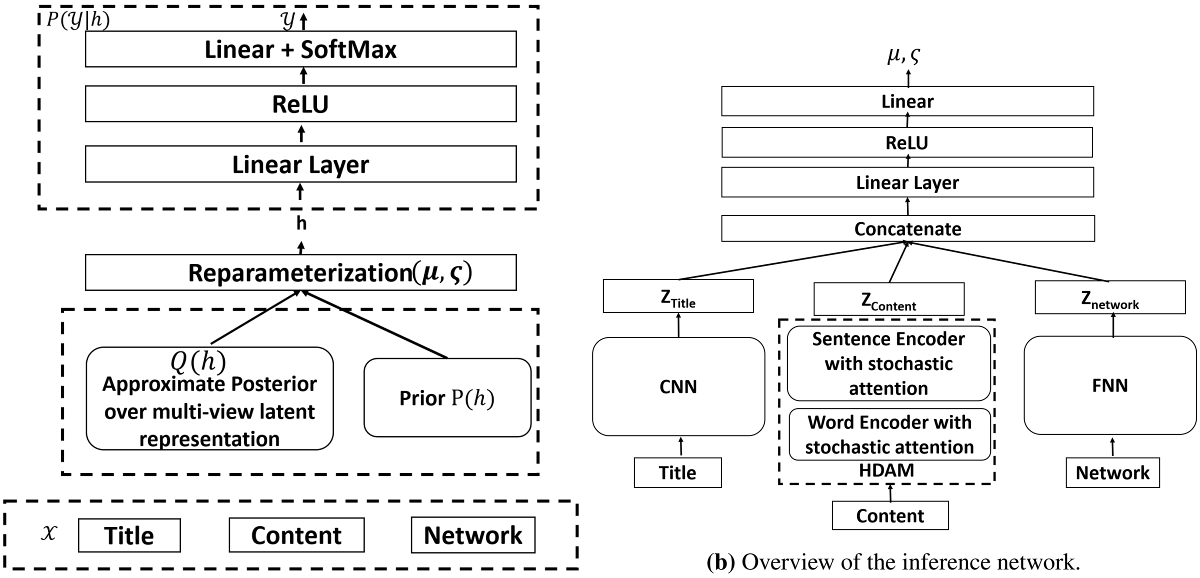

(a) Overview of our full model.

(b) Overview of the inference network.

Figure 2: A broad overview of our MVDAM model depicting the three major components:a discriminator, an inference network and a prior and captures cues from multiple views of the news article. As noted by Miao et al. (2016) we use stochastic attention units which are shown to model ambiguity better. We thus train the model end-to-end using neural variational inference.

the evidence lower bound $\mathcal{L} \leq \Pr(\boldsymbol{y}|\boldsymbol{X})$ namely,

$$\mathcal{L} = \mathbb{E}_{q_\phi(\boldsymbol{h})}[p(\boldsymbol{Y}|\boldsymbol{h})] - D_{KL}(q_\phi(\boldsymbol{h})||p(\boldsymbol{h}|\boldsymbol{X})), \tag{2}$$

where $p(\boldsymbol{Y}|\boldsymbol{h})$ denotes a probability distribution over $\boldsymbol{Y}$ given the latent representation $\boldsymbol{h}$, and $p(\boldsymbol{h}|\boldsymbol{X})$ denotes the probability distribution over $\boldsymbol{h}$ conditioned on $\boldsymbol{X}$.

Equation 2 can be interpreted as consisting of three components, each of which can modeled separately: (a) **Discriminator** $p(\boldsymbol{Y}|\boldsymbol{h})$ can be viewed as a discriminator given the hidden representation $\boldsymbol{h}$. Maximizing the first term is thus equivalent to minimizing the cross-entropy loss between the model's prediction and true labels. (b) The second term, the KL Divergence term consists of two components: (1) **Approximate Posterior** The term $q_\phi(\boldsymbol{h})$ also known as the approximate posterior parameterizes the latent distribution which encodes the multi-modal features $\boldsymbol{X}$ of a document. (2) **Prior** The term $p(\boldsymbol{h}|\boldsymbol{X})$ can be viewed as a prior which can be uninformative (a standard Gaussian prior in the most general case, or any other prior model based on other features). We now discuss how we model each of these components in detail.

4.1 Discriminator

We use a simple feed-forward network with a linear layer that accepts as input the latent hidden representation of $\boldsymbol{X}$, followed by a ReLU for non-linearity followed by a linear layer and a final softmax layer to model this component.

4.2 Approximate Posterior

Here we model the approximate posterior $q_\phi(\boldsymbol{h})$ by an inference network shown succinctly in Figure 2b. The inference network takes as input the features $\boldsymbol{X}$ and learns a corresponding hidden representation $\boldsymbol{h}$. More specifically, it outputs two components: $(\mu,\ \varsigma)$ corresponding to the mean and log-variance of the gaussian parametrizing the hidden representation $\boldsymbol{h}$. We model this using a "multi-view" network which incorporates hidden representations learned from multiple modalities into a joint representation. Specifically, given d-dimensional hidden representations corresponding to multiple modalities z_{title}, $z_{network}$, and $z_{content}$ the model first concatenates these representations into a single $3d$-dimensional representation z_{concat} which is then input through a 2-layer feed-forward network to output a d-dimensional mean vector μ and a d-dimensional log-variance vector ς that parameterizes the latent distribution governing $\boldsymbol{h}$. We now discuss the models used for capturing each view.

4.2.1 Modeling the Title

We learn a latent representation of the title of a article by using a convolutional network. Convolutional networks have been shown to be very effective for modeling short sentences like titles of news articles. In particular, we use the same architecture proposed by (Kim, 2014). The input words of the title are mapped to word embeddings and concatenated and passed through convolutional filters of varying window sizes. This is then followed by a max-over-time pooling (Collobert et al., 2011). The outputs of this layer are input to a fully connected layer of dimension d with drop-out which outputs $\boldsymbol{z}_{title}$, the latent representation of the title.

4.2.2 Modeling the Network Structure of articles

Capturing the network structure of article consists of two steps: (a) Learning a network representation of each source based on its social graph G. (b) Using the learned representation of each source to capture the link structure of a particular article.

We use a state-of-the-art network representation learning algorithm to learn representations of nodes in a social network. In particular, we use Node2Vec (Grover and Leskovec, 2016), which learns a d-dimensional representation of each source given the hyperlink structure graph G. Node2Vec seeks to maximize the log likelihood of observing the neighborhood of a node $\mathcal{N}(u)$, given the node u. Let $\boldsymbol{F}$ be a matrix of size (V, d) where $\boldsymbol{F}(u)$ represents the embedding of node u. We then maximize the following likelihood function $\max_{\boldsymbol{F}} \sum_u \log \Pr(\mathcal{N}(u)|u)$. We model the above likelihood similar to the Skip-gram architecture (Mikolov et al., 2013) by assuming that the likelihood of observing a node $v \in \mathcal{N}(u)$ is conditionally independent of any other node in the neighborhood given u. That is $\log \Pr(\mathcal{N}(u)|u) = \sum_{v \in \mathcal{N}(u)} \log \Pr(v|u)$. We then model $\Pr(v|u) = \frac{e^{\boldsymbol{F}(u).\boldsymbol{F}(v)}}{\sum_v e^{\boldsymbol{F}(u).\boldsymbol{F}(v)}}$. Having fully specified the log likelihood function, we can now optimize it using stochastic gradient ascent.

Having learned the embedding matrix $\boldsymbol{F}$ for each source node, we now model the link structure of any given article $\mathcal{A}$ simply by the average of the network embedding representations for each link l referenced in $\mathcal{A}$. In particular, we compute $\boldsymbol{z}_{network}$ as: $\boldsymbol{z}_{network} = \frac{1}{|\mathcal{A}|} \sum_{l \in \mathcal{A}} \boldsymbol{F}(l)$.

4.2.3 Modeling the Content of articles

To model the content of an article, we use a hierarchical approach with attention. In particular, we compute attention at both levels: (a) words and (b) sentences. We closely follow the approach by (Yang et al., 2016) which learns a latent representation of a document d using both word and sentence attention models.

We model the article $\mathcal{A}$ hierarchically, by first representing each sentence i with a hidden representation s_i. We model the fact that not all words contribute equally in the sentence through a word level attention mechanism. We then learn the representation of the article $\mathcal{A}$ by composing these individual sentence level representations with a sentence level attention mechanism.

Learning sentence representations We first map each word to its embedding matrix through a lookup embedding matrix $\boldsymbol{W}$. We then learn a hidden representation of the given sentence h_{it} centered around word w_i by embedding the sentence through a bi-directional GRU as described by (Bahdanau et al., 2014). Since not all words contribute equally to the representation of the sentence, we introduce a word level attention mechanism which attempts to extract relevant words that contribute to the meaning of the sentence. Specifically we learn a word level attention matrix $\boldsymbol{W_w}$ as follows $\alpha_i \propto \exp(\boldsymbol{W_w} h_{it} + b_w), s_i = \sum_t \alpha_i h_{it}$ where s_i is the latent representation of the sentence i.

Composing sentence representations We follow a similar method to learn a latent representation of an article. Given the embedding s_i of each sentence in the article, we learn a hidden representation of the given sentence h_i centered around s_i by embedding the list of sentences through a bi-directional GRU as described by (Bahdanau et al., 2014). Once again, since not all sentences contribute equally to the representation of the article, we introduce a sentence level attention mechanism which attempts to extract relevant sentences that contribute to the meaning of the article. Specifically we learn the weights of a sentence level attention matrix $\boldsymbol{W}_s$ as $\alpha_s \propto \exp(\boldsymbol{W}_s h_s + b_s), \boldsymbol{z}_{content} = \sum_s \alpha_s h_s$, where $\boldsymbol{z}_{content}$ is the latent representation of the article. In this case we let the hidden representation of the sentence be a stochastic representation similar to the work by (Miao et al., 2016) and use the Gaussian re-parameterization trick to enable training via end-to-end gradient based methods [5]. Such techniques have been shown to be useful in

---

[5] Using deterministic sentence representations is a special case.

modeling ambiguity and also generalize well to small training datasets (Miao et al., 2016).

4.3 Prior

The prior models $p(\boldsymbol{h}|\boldsymbol{X})$ in Equation 2. Note that our proposed framework is general and can be used to incorporate a variety of priors. Here, we assume the prior is drawn from a Gaussian distribution with diagonal co-variances. The KL Divergence term in Equation 2 can thus be analytically computed. In particular, the KL Divergence between two K dimensional Gaussian distributions $\boldsymbol{A}, \boldsymbol{B}$ with means μ_A, μ_B and diagonal co-variances κ_A, κ_B is:

$$D_{KL}(\boldsymbol{A}, \boldsymbol{B}) = -\frac{1}{2}\sum_{j=1}^{j=K}(1 + \log\frac{\kappa_{Aj}}{\kappa_{Bj}} - \frac{\kappa_{Aj}}{\kappa_{Bj}} - (\mu_{Aj} - \mu_{Bj})^2/\kappa_{Bj}) \quad (3)$$

Parameter Estimation Having described precisely, the models for each of the components in Equation 2, we can reformulate the maximization of the variational lower bound to the following loss function on the set of all learn-able model parameters θ: $\mathcal{J}(\theta)$ as follows:

$$\mathcal{J}(\theta) = \mathrm{NLL}(\boldsymbol{y}|\boldsymbol{X}) + \lambda \mathrm{D}_{KL}(\boldsymbol{q}(\boldsymbol{h})||\boldsymbol{p}(\boldsymbol{h}|\boldsymbol{X})), \quad (4)$$

where NLL is the negative log likelihood loss computed between the predicted label and the true label, and λ is a hyper-parameter that controls the amount of regularization offered by the KL Divergence term. We use ADADELTA to minimize this loss function.

5 Experiments

We evaluate our model against several competitive baselines which model only a single view to place our model in context:

1. **Chance Baseline** We consider a simple baseline that returns a draw from the label distribution as the prediction.
2. **Logistic Regression LR (Title)** We consider a bag of words classifier using Logistic Regression that can capture linear relationships in the feature space and use the words of the title as the feature set.
3. **CNN (Title)** We consider a convolutional net classifier based on exactly the same architecture as (Kim, 2014) which uses the title of the news article. Convolutional Nets have been shown to be extremely effective at classifying short pieces of text and can capture non-linearities in the feature space(Kim, 2014).
4. **FNN (Network)** We also consider a simple fully-connected feed forward neural network using only the network features to characterize the predictive power of the network alone.
5. **HDAM Model (Content)** We use the state of the art hierarchical document attention model proposed by (Yang et al., 2016) that models the content of the article using both word and sentence level attention mechanisms.

We consider three different flavors of our proposed model which differ in the subset of modalities used (a) **Title and Network** (b) **Title and Content**, and (c) **Full model**: Title, Network, and Content. We train all of our models and the baselines on the training data set choosing all hyper-parameter using the validation set. We report the performance of all models on the held-out test set.

Experimental Settings We set the embedding latent dimension captured by each view to be 128 including the final latent representation obtained by fusing multiple modalities. In case of the CNN's, we consider three convolutional filters of window sizes $3, 4, 5$ each yielding a 100 dimensional feature map followed by max-over time pooling which is then passed through a fully connected layer to yield the output. In all the neural models, we used AdaDelta with an initial learning rate of 1.0 to learn the parameters of the model via back-propagation.

Model	Views	P	R	F1
CHANCE	–	34.53	34.59	34.53
LR	Title	59.53	59.42	59.12
CNN	Title	59.26	59.40	59.24
FNN	Network	68.28	56.54	55.10
HDAM	Content	69.85	68.72	68.92
MVDAM	Title, Network	**69.87**	**69.71**	**69.66**
MVDAM	Title, Content	**70.84**	**70.19**	**69.54**
MVDAM	Title, Network, Content	**80.10**	**79.56**	**79.67**

Table 1: Precision, Recall, and F1 scores of our model MVDAM on the test set compared with several baselines. All flavors of our model significantly outperform baselines and yield state of the art performance.

5.1 Results and Analysis

Quantitative Results Table 1 shows the results of the evaluation. First note that the logistic regression classifier and the CNN model using the Title outperforms the CHANCE classifier significantly

(F1: $\mathbf{59.12}, \mathbf{59.24}$ vs $\mathbf{34.53}$). Second, only modeling the network structure yields a F1 of $\mathbf{55.10}$ but still significantly better than the chance baseline. This confirms our intuition that modeling the network structure can be useful in prediction of ideology. Third, note that modeling the content (HDAM) significantly outperforms all previous baselines (F1:$\mathbf{68.92}$). This suggests that content cues can be very strong indicators of ideology. Finally, all flavors of our model outperform the baselines. Specifically, observe that incorporating the network cues outperforms all uni-modal models that only model either the title, the network, or the content. It is also worth noting that without the network, only the title and the content show only a small improvement over the best performing baseline ($\mathbf{69.54}$ vs $\mathbf{68.92}$) suggesting that the network yields distinctive cues from both the title, and the content. Finally, the best performing model effectively uses all three modalities to yield a F1 score of $\mathbf{79.67}$ outperforming the state of the art baseline by 10 percentage points. Altogether our results suggest the superiority of our model over competitive baselines. In order to obtain deeper insights into our model, we also perform a qualitative analysis of our model's predictions.

Visualizing Attention Scores Figure 3 shows a visualization of sentences based on their attention scores. Note that for a left leaning article (see Figure 3a), the model focuses on sentences involving `gun-control, feminists, and transgender`. In contrast, a visualization of sentence attention scores for an article which the model predicted as "right-leaning" ((see Figure 3b)) reveals a focus on words like `god, religion` etc. These observations qualitatively suggest that the model is able to effectively pick up on content cues present in the article. By examining the distribution over the sentence indices corresponding to the maximum attention scores, we noted that only in about half the instances, the model focuses its greatest attention on the beginning of the article suggesting that the ability to selectively focus on sentences in the news article contributes to the superior performance.

Challenging Cases In Table 2, we highlight some of the challenges of our model. In particular, our model finds it quite challenging to identify the political ideology of the source for articles that are non-political and related to global events, or entertainment. Examples include instances like `Tourist dies hiking in Australia Outback heat` or `Juan Williams makes the 'case for Oprah'`. We also note that articles with "click-baity" titles like `We are all Just Overclocked Chimpanzees` are not necessarily discriminative of the underlying ideology. In summary, while our proposed model significantly advances the state of art, it also suggests scope for further improvement especially in identifying political ideologies of articles in topics like Entertainment or Sports. For example, prior research suggests that engagement in particular sports is correlated with the political leanings (Hoberman, 1977) which suggest that improved models might need to capture deeper linguistic and contextual cues.

Ideological Proportions of News Sources Finally, we compute the expected proportion of an ideology in a given source based on the probability estimates output by our model for the various articles. While one might expect that the expected degree of "left-ness" (or "right-ness") for a given source can easily be computed by taking a simple mean of the prediction probability for the given ideology over all articles belonging to the source, such an approach can be in-accurate because the probability estimates output by the model are not necessarily calibrated and therefore cannot be interpreted as a confidence value. We therefore use isotonic regression to calibrate the probability scores output by the model. Having calibrated the probability scores, we now compute the degree to which a particular news source leans toward an ideology by simply computing the mean output score over all articles corresponding to the source. Table 3 shows the top 10 sources ranked according to their proportions for each ideology. We note that sources like CNN, Buzz Feed, SF Chronicle are considered more left-leaning than the Washington Post. Similarly, we note that NPR and Reuters are considered to be the most center-aligned while Breitbart, Infowars and Blaze are considered to be most right-aligned by our model. These observations are moderately aligned with survey results that place news sources on the ideology spectrum based on the political beliefs of their consumers [6].

---

[6] http://www.journalism.org/2014/10/21/political-polarization-media-habits/pj_14-10-21_mediapolarization-08/

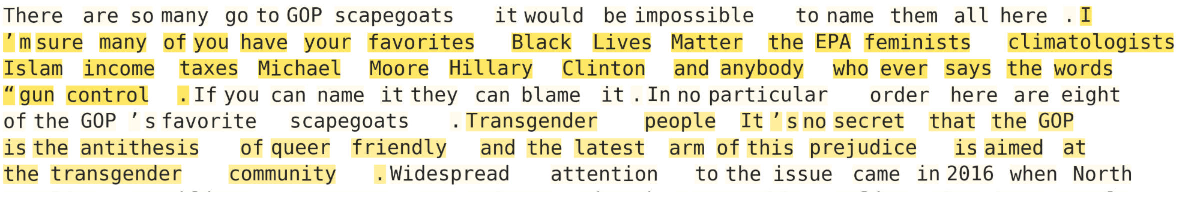
There are so many go to GOP scapegoats it would be impossible to name them all here . I
’ m sure many of you have your favorites Black Lives Matter the EPA feminists climatologists
Islam income taxes Michael Moore Hillary Clinton and anybody who ever says the words
“ gun control . If you can name it they can blame it . In no particular order here are eight
of the GOP ’ s favorite scapegoats . Transgender people It ’ s no secret that the GOP
is the antithesis of queer friendly and the latest arm of this prejudice is aimed at
the transgender community . Widespread attention to the issue came in 2016 when North

(a) Sample attention on sentences for a Left aligned article.

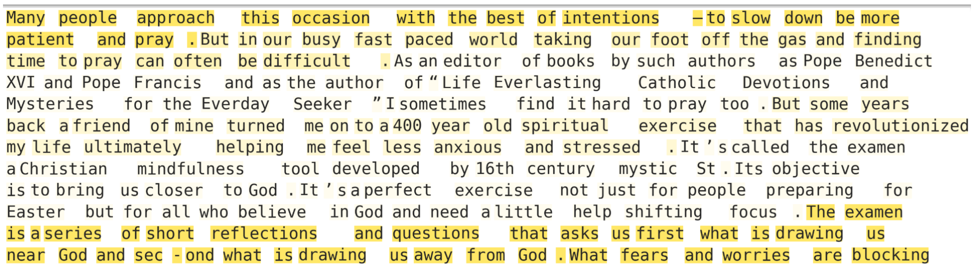
Many people approach this occasion with the best of intentions – to slow down be more
patient and pray . But in our busy fast paced world taking our foot off the gas and finding
time to pray can often be difficult . As an editor of books by such authors as Pope Benedict
XVI and Pope Francis and as the author of “ Life Everlasting Catholic Devotions and
Mysteries for the Everday Seeker ” I sometimes find it hard to pray too . But some years
back a friend of mine turned me on to a 400 year old spiritual exercise that has revolutionized
my life ultimately helping me feel less anxious and stressed . It ’ s called the examen
a Christian mindfulness tool developed by 16th century mystic St . Its objective
is to bring us closer to God . It ’ s a perfect exercise not just for people preparing for
Easter but for all who believe in God and need a little help shifting focus . The examen
is a series of short reflections and questions that asks us first what is drawing us
near God and sec - ond what is drawing us away from God . What fears and worries are blocking

(b) Sample attention on sentences for a Right aligned article.

Figure 3: Visualization of attention on different sentences on two sample articles from the Left and Right aligned sources respectively. Note the different focus based in the ideology reflected by the highlighted words.

Article Title	Source Label	Predicted Label
Juan Williams Makes the ’Case for Oprah’	Right	Left
Tourist dies hiking in Australia Outback heat	Right	Left
Back From China, UCLA Basketball Players Plagued by Father	Right	Left
Democrat Ralph Northam Elected Governor of Virginia	Right	Left
South Africa blighted by racially charged farm murders	Right	Left
Lawsuit: Stripper punched man, knocked out his front tooth	Left	Right
Heres How to Keep Fake News Off Twitter	Left	Right
We Are All Just Overclocked Chimpanzee	Left	Right
Curious Arctic Fox Pups Destroy Hidden Camera In The Most Adorable Way	Left	Right
I am American, Jewish, and banned from Israel for my activism	Left	Right

Table 2: Few failure cases of our model illustrating what our model finds challenging. Articles with “click-baity” titles are not necessarily very discriminative of the ideology. Similarly, articles that are *non-political* and related to global events or entertainment are quite challenging.

Rank	Source
1	CNN
2	BuzzFeed
3	SF Chronicle
4	CBS News
5	BoingBoing
6	Mother Jones
7	Think Progress
8	The Atlantic
9	The Washington Post
10	Rolling Stone

(a) Left aligned

Rank	Source
1	NPR
2	Reuters
3	USA Today
4	BBC
5	CNBC
6	Chicago Tribune
7	Business Insider
8	Forbes
9	APR
10	The Wall Street Journal

(b) Centre aligned

Rank	Source
1	Breitbart
2	Infowars
3	Blaze
4	Fox News
5	KSL
6	Townhall
7	CBN
8	ConservativeHQ
9	NewsMax
10	DailyWire

(c) Right aligned

Table 3: A Top 10 ranking of Ideological sources as obtained by our model which correlate moderately with external surveys.

6 Conclusion

We proposed a model to leverage cues from multiple views in the predictive task of detecting political ideology of news articles. We show that incorporating cues from the title, the link structure and the content significantly beats state of the art. Finally, using the predicted probabilities of our model, we draw on methods for probability calibration to rank news sources by their ideological proportions which moderately correlates with existing surveys on the ideological placement of news sources. To conclude, our proposed framework effectively leverages cues from multiple views to yield state of the art interpret-able performance and sets the stage for future work which can easily incorporate other modalities like audio, video and images.

Acknowledgments

We thank the anonymous reviewers for their comments. This research was supported in part by DARPA Grant D18AP00044 funded under the DARPA YFA program. This work was also partially supported by NSF grants DBI-1355990 and IIS-1546113. The authors are solely responsible for the contents of the paper, and the opinions expressed in this publication do not reflect those of the funding agencies.

References

Amr Ahmed and Eric P Xing. 2010. Staying informed: supervised and semi-supervised multi-view topical analysis of ideological perspective. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1140–1150. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Dallas Card, Amber E Boydstun, Justin H Gross, Philip Resnik, and Noah A Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 438–444.

Royce Carroll, Jeffrey B Lewis, James Lo, Keith T Poole, and Howard Rosenthal. 2009. Measuring bias and uncertainty in dw-nominate ideal point estimates via the parametric bootstrap. *Political Analysis*, 17(3):261–275.

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

Frank E Dardis, Frank R Baumgartner, Amber E Boydstun, Suzanna De Boef, and Fuyuan Shen. 2008. Media framing of capital punishment and its impact on individuals' cognitive responses. *Mass Communication & Society*, 11(2):115–140.

Matthew Gentzkow and Jesse M Shapiro. 2010. What drives media slant? evidence from us daily newspapers. *Econometrica*, 78(1):35–71.

Sean Gerrish and David M Blei. 2011. Predicting legislative roll calls from text. In *Proceedings of the 28th international conference on machine learning (icml-11)*, pages 489–496.

Tim Groseclose and Jeffrey Milyo. 2005. A measure of media bias. *The Quarterly Journal of Economics*, 120(4):1191–1237.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

John M Hoberman. 1977. Sport and political ideology. *Journal of sport and social issues*, 1(2):80–114.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1113–1122.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Michael Laver, Kenneth Benoit, and John Garry. 2003. Extracting policy positions from political texts using words as data. *American Political Science Review*, 97(2):311–331.

Wei-Hao Lin, Eric Xing, and Alexander Hauptmann. 2008. A joint topic and perspective model for ideological discourse. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 17–32. Springer.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Burt L Monroe and Ko Maeda. 2004. Talks cheap: Text-based estimation of rhetorical ideal-points.

Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. 2017. Beyond binary labels: political ideology prediction of twitter users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 729–740.

Yanchuan Sim, Brice DL Acree, Justin H Gross, and Noah A Smith. 2013. Measuring ideological proportions in political speeches. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 91–101.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 327–335. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Predicting Factuality of Reporting and Bias of News Media Sources

Ramy Baly[1], **Georgi Karadzhov**[3], **Dimitar Alexandrov**[3], **James Glass**[1], **Preslav Nakov**[2]
[1]MIT Computer Science and Artificial Intelligence Laboratory, MA, USA
[2]Qatar Computing Research Institute, HBKU, Qatar;
[3]Sofia University, Bulgaria
`{baly, glass}@mit.edu, pnakov@qf.org.qa`
`{georgi.m.karadjov, Dimityr.Alexandrov}@gmail.com`

Abstract

We present a study on predicting the factuality of reporting and bias of news media. While previous work has focused on studying the veracity of claims or documents, here we are interested in characterizing entire news media. These are under-studied but arguably important research problems, both in their own right and as a prior for fact-checking systems. We experiment with a large list of news websites and with a rich set of features derived from (*i*) a sample of articles from the target news medium, (*ii*) its Wikipedia page, (*iii*) its Twitter account, (*iv*) the structure of its URL, and (*v*) information about the Web traffic it attracts. The experimental results show sizable performance gains over the baselines, and confirm the importance of each feature type.

1 Introduction

The rise of social media has democratized content creation and has made it easy for everybody to share and spread information online. On the positive side, this has given rise to citizen journalism, thus enabling much faster dissemination of information compared to what was possible with newspapers, radio, and TV. On the negative side, stripping traditional media from their gate-keeping role has left the public unprotected against the spread of misinformation, which could now travel at breaking-news speed over the same democratic channel. This has given rise to the proliferation of false information that is typically created either (*a*) to attract network traffic and gain financially from showing online advertisements, e.g., as is the case of *clickbait*, or (*b*) to affect individual people's beliefs, and ultimately to influence major events such as political elections (Vosoughi et al., 2018). There are strong indications that false information was weaponized at an unprecedented scale during the 2016 U.S. presidential campaign.

"Fake news", which can be defined as "fabricated information that mimics news media content in form but not in organizational process or intent" (Lazer et al., 2018), became the word of the year in 2017, according to Collins Dictionary. "Fake news" thrive on social media thanks to the mechanism of sharing, which amplifies effect. Moreover, it has been shown that "fake news" spread faster than real news (Vosoughi et al., 2018). As they reach the same user several times, the effect is that they are perceived as more credible, unlike old-fashioned spam that typically dies the moment it reaches its recipients. Naturally, limiting the sharing of "fake news" is a major focus for social media such as Facebook and Twitter.

Additional efforts to combat "fake news" have been led by fact-checking organizations such as Snopes, FactCheck and Politifact, which manually verify claims. Unfortunately, this is inefficient for several reasons. First, manual fact-checking is slow and debunking false information comes too late to have any significant impact. At the same time, automatic fact-checking lags behind in terms of accuracy, and it is generally not trusted by human users. In fact, even when done by reputable fact-checking organizations, debunking does little to convince those who already believe in false information.

A third, and arguably more promising, way to fight "fake news" is to focus on their source. While "fake news" are spreading primarily on social media, they still need a "home", i.e., a website where they would be posted. Thus, if a website is known to have published non-factual information in the past, it is likely to do so in the future. Verifying the reliability of the source of information is one of the basic tools that journalists in traditional media use to verify information. It is also arguably an important prior for fact-checking systems (Popat et al., 2017; Nguyen et al., 2018).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3528–3539
Brussels, Belgium, October 31 - November 4, 2018.

Fact-checking organizations have been producing lists of unreliable online news sources, but these are incomplete and get outdated quickly. Therefore, there is a need to predict the factuality of reporting for a given online medium automatically, which is the focus of the present work. We further study the bias of the source (left vs. right), as the two problems are inter-connected, e.g., extreme-left and extreme-right websites tend to score low in terms of factual reporting. Our contributions can be summarized as follows:

- We focus on an under-explored but arguably very important problem: predicting the factuality of reporting of a news medium. We further study bias, which is also under-explored.
- We create a new dataset of news media sources, which has annotations for both tasks, and is 1-2 orders of magnitude larger than what was used in previous work. We release the dataset and our code, which should facilitate future research.[1]
- We use a variety of sources such as (*i*) a sample of articles from the target website, (*ii*) its Wikipedia page, (*iii*) its Twitter account, (*iv*) the structure of its URL, and (*v*) information about the Web traffic it has attracted. This combination, as well as some of the sources, are novel for these problems.
- We further perform an ablation study of the impact of the individual (groups of) features.

The remainder of this paper is organized as follows: Section 2 provides an overview of related work. Section 3 describes our method and features. Section 4 presents the data, the experiments, and the evaluation results. Finally, Section 5 concludes with some directions for future work.

2 Related Work

Journalists, online users, and researchers are well-aware of the proliferation of false information, and thus topics such as credibility and fact-checking are becoming increasingly important. For example, the ACM Transactions on Information Systems journal dedicated, in 2016, a special issue on Trust and Veracity of Information in Social Media (Papadopoulos et al., 2016).

There have also been some related shared tasks such as the SemEval-2017 task 8 on Rumor Detection (Derczynski et al., 2017), the CLEF-2018 lab on Automatic Identification and Verification of Claims in Political Debates (Atanasova et al., 2018; Barrón-Cedeño et al., 2018; Nakov et al., 2018), and the FEVER-2018 task on Fact Extraction and VERification (Thorne et al., 2018).

The interested reader can learn more about "fake news" from the overview by Shu et al. (2017), which adopted a data mining perspective and focused on social media. Another recent survey was run by Thorne and Vlachos (2018), which took a fact-checking perspective on "fake news" and related problems. Yet another survey was performed by Li et al. (2016), covering truth discovery in general. Moreover, there were two recent articles in *Science*: Lazer et al. (2018) offered a general overview and discussion on the science of "fake news", while Vosoughi et al. (2018) focused on the process of proliferation of true and false news online. In particular, they analyzed 126K stories tweeted by 3M people more than 4.5M times, and confirmed that "fake news" spread much wider than true news.

Veracity of information has been studied at different levels: (*i*) claim-level (e.g., *fact-checking*), (*ii*) article-level (e.g., *"fake news" detection*), (*iii*) user-level (e.g., *hunting for trolls*), and (*iv*) medium-level (e.g., *source reliability estimation*). Our primary interest here is in the latter.

2.1 Fact-Checking

At the claim-level, fact-checking and rumor detection have been primarily addressed using information extracted from social media, i.e., based on how users comment on the target claim (Canini et al., 2011; Castillo et al., 2011; Ma et al., 2015, 2016; Zubiaga et al., 2016; Ma et al., 2017; Dungs et al., 2018; Kochkina et al., 2018). The Web has also been used as a source of information (Mukherjee and Weikum, 2015; Popat et al., 2016, 2017; Karadzhov et al., 2017b; Mihaylova et al., 2018; Baly et al., 2018).

In both cases, the most important information sources are *stance* (does a tweet or a news article agree or disagree with the claim?), and *source reliability* (do we trust the user who posted the tweet or the medium that published the news article?). Other important sources are linguistic expression, meta information, and temporal dynamics.

---

[1] The data and the code are at `http://github.mit.edu/CSAIL-SLS/News-Media-Reliability/`

2.2 Stance Detection

Stance detection has been addressed as a task in its own right, where models have been developed based on data from the Fake News Challenge (Riedel et al., 2017; Thorne et al., 2017; Mohtarami et al., 2018; Hanselowski et al., 2018), or from SemEval-2017 Task 8 (Derczynski et al., 2017; Dungs et al., 2018; Zubiaga et al., 2018). It has also been studied for other languages such as Arabic (Darwish et al., 2017b; Baly et al., 2018).

2.3 Source Reliability Estimation

Unlike stance detection, the problem of source reliability remains largely under-explored. In the case of social media, it concerns modeling the user[2] who posted a particular message/tweet, while in the case of the Web, it is about the trustworthiness of the source (the URL domain, the medium). The latter is our focus in this paper.

In previous work, the source reliability of news media has often been estimated automatically based on the general stance of the target medium with respect to known manually fact-checked claims, without access to gold labels about the overall medium-level factuality of reporting (Mukherjee and Weikum, 2015; Popat et al., 2016, 2017, 2018). The assumption is that reliable media agree with true claims and disagree with false ones, while for unreliable media it is mostly the other way around. The trustworthiness of Web sources has also been studied from a Data Analytics perspective. For instance, Dong et al. (2015) proposed that a trustworthy source is one that contains very few false facts. In this paper, we follow a different approach by studying the source reliability as a task in its own right, using manual gold annotations specific for the task.

Note that estimating the reliability of a source is important not only when fact-checking a claim (Popat et al., 2017; Nguyen et al., 2018), but it also gives an important prior when solving article-level tasks such as "fake news" and click-bait detection (Brill, 2001; Finberg et al., 2002; Hardalov et al., 2016; Karadzhov et al., 2017a; De Sarkar et al., 2018; Pan et al., 2018; Pérez-Rosas et al., 2018).

2.4 "Fake News" Detection

Most work on "fake news" detection has relied on medium-level labels, which were then assumed to hold for all articles from that source.

Horne and Adali (2017) analyzed three small datasets ranging from a couple of hundred to a few thousand articles from a couple of dozen sources, comparing (*i*) real news vs. (*ii*) "fake news" vs. (*iii*) satire, and found that the latter two have a lot in common across a number of dimensions. They designed a rich set of features that analyze the text of a news article, modeling its complexity, style, and psychological characteristics. They found that "fake news" pack a lot of information in the title (as the focus is on users who do not read beyond the title), and use shorter, simpler, and repetitive content in the body (as writing fake information takes a lot of effort). Thus, they argued that the title and the body should be analyzed separately.

In follow-up work, Horne et al. (2018b) created a large-scale dataset covering 136K articles from 92 sources from `opensources.co`, which they characterize based on 130 features from seven categories: structural, sentiment, engagement, topic-dependent, complexity, bias, and morality. We use this set of features when analyzing news articles.

In yet another follow-up work, Horne et al. (2018a) trained a classifier to predict whether a given news article is coming from a reliable or from an unreliable ("*fake news*" or *conspiracy*)[3] source. Note that they assumed that all news from a given website would share the same reliability class. Such an assumption is fine for training (distant supervision), but we find it problematic for testing, where we believe manual documents-level labels are needed.

Potthast et al. (2018) used 1,627 articles from nine sources, whose factuality has been manually verified by professional journalists from BuzzFeed. They applied stylometric analysis, which was originally designed for authorship verification, to predict factuality (fake vs. real).

Rashkin et al. (2017) focused on the language used by "fake news" and compared the prevalence of several features in articles coming from trusted sources vs. hoaxes vs. satire vs. propaganda. However, their linguistic analysis and their automatic classification were at the article level and they only covered eight news media sources.

---

[2] User modeling in social media and news community forums has focused on finding malicious users such as opinion manipulation *trolls*, paid (Mihaylov et al., 2015b) or just perceived (Mihaylov et al., 2015a; Mihaylov and Nakov, 2016; Mihaylov et al., 2018; Mihaylova et al., 2018), *sockpuppets* (Maity et al., 2017), *Internet water army* (Chen et al., 2013), and *seminar users* (Darwish et al., 2017a).

[3] We show in parentheses, the labels from `opensources.co` that are used to define a category.

Unlike the above work, (*i*) we perform classification at the news medium level rather than focusing on an individual article. Thus, (*ii*) we use reliable manually-annotated labels as opposed to noisy labels resulting from projecting the category of a news medium to all news articles published by this medium (as most of the work above did).[4] Moreover, (*iii*) we use a much larger set of news sources, namely 1,066, which is 1-2 orders of magnitude larger than what was used in previous work. Furthermore, (*iv*) we use a larger number of features and a wider variety of feature types compared to the above work, including features extracted from knowledge sources that have been largely neglected in the literature so far such as information from Wikipedia and the structure of the medium's URL.

2.5 Media Bias Detection

As we mentioned above, bias was used as a feature for "fake news" detection (Horne et al., 2018b). It has also been the target of classification, e.g., Horne et al. (2018a) predicted whether an article is biased (*political* or *bias*) vs. unbiased. Similarly, Potthast et al. (2018) classified the bias in a target article as (*i*) left vs. right vs. mainstream, or as (*ii*) hyper-partisan vs. mainstream. Finally, Rashkin et al. (2017) studied propaganda, which can be seen as extreme bias. See also a recent position paper (Pitoura et al., 2018) and an overview on bias the Web (Baeza-Yates, 2018).

Unlike the above work, we focus on bias at the medium level rather than at the article level. Moreover, we work with fine-grained labels on an ordinal scale rather then having a binary setup (some work above had three degrees of bias, while we have seven).

3 Method

In order to predict the factuality of reporting and the bias for a given news medium, we collect information from multiple relevant sources, which we use to train a classifier. In particular, we collect a rich set of features derived from (*i*) a sample of articles from the target news medium, (*ii*) its Wikipedia page if it exists, (*iii*) its Twitter account if it exists, (*iv*) the structure of its URL, and (*v*) information about the Web traffic it has attracted. We describe each of these sources below.

Articles We argue that analysis (textual, syntactic and semantic) of the content of the news articles published by a given target medium should be critical for assessing the factuality of its reporting, as well as of its potential bias. Towards this goal, we borrow a set of 141 features that were previously proposed for detecting "fake news" articles (Horne et al., 2018b), as we have described above. These features are used to analyze the following article characteristics:

- **Structure**: POS tags, linguistic features based on the use of specific words (function words, pronouns, etc.), and features for clickbait title classification from (Chakraborty et al., 2016);
- **Sentiment**: sentiment scores using lexicons (Recasens et al., 2013; Mitchell et al., 2013) and full systems (Hutto and Gilbert, 2014);
- **Engagement**: number of shares, reactions, and comments on Facebook;
- **Topic**: lexicon features to differentiate between science topics and personal concerns;
- **Complexity**: type-token ratio, readability, number of cognitive process words (identifying discrepancy, insight, certainty, etc.);
- **Bias**: features modeling bias using lexicons (Recasens et al., 2013; Mukherjee and Weikum, 2015) and subjectivity as calculated using pre-trained classifiers (Horne et al., 2017);
- **Morality**: features based on the Moral Foundation Theory (Graham et al., 2009) and lexicons (Lin et al., 2017)

Further details are available in (Horne et al., 2018b). For each target medium, we retrieve some articles, then we calculate these features separately for the title and for the body of each article, and finally we average the values of the 141 features over the set of retrieved articles.

Wikipedia We further leverage Wikipedia as an additional source of information that can help predict the factuality of reporting and the bias of a target medium. For example, the absence of a Wikipedia page may indicate that a website is not credible. Also, the content of the page might explicitly mention that a certain website is satirical, left-wing, or has some property related to our task.

---

[4] Two notable exceptions are (Potthast et al., 2018) and (Pérez-Rosas et al., 2018), who use news articles whose factuality has been manually checked and annotated.

Accordingly, we extract the following features:

- *Has Page:* indicates whether the target medium has a Wikipedia page;
- Vector representation for each of the following segments of the Wikipedia page, whenever applicable: *Content*, *Infobox*, *Summary*, *Categories*, and *Table of Contents*. We generate these representations by averaging the word embeddings (pretrained word2vec embeddings) of the corresponding words.

Twitter Given the proliferation of social media, most news media have Twitter accounts, which they use to reach out to more users online. The information that can be extracted from a news medium's Twitter profile can be valuable for our tasks. In particular, we use the following features:

- *Has Account:* Whether the medium has a Twitter account. We check this based on the top results for a search against Google, restricting the domain to `twitter.com`. The idea is that media that publish unreliable information might have no Twitter accounts.
- *Verified:* Whether the account is verified by Twitter. The assumption is that "fake news" media would be less likely to have their Twitter account verified. They might be interested in pushing their content to users via Twitter, but they would also be cautious about revealing who they are (which is required by Twitter to get them verified).
- *Created:* The year the account was created. The idea is that accounts that have been active over a longer period of time are more likely to belong to established media.
- *Has Location:* Whether the account provides information about its location. The idea is that established media are likely to have this public, while "fake news" media may want to hide it.
- *URL Match:* Whether the account includes a URL to the medium, and whether it matches the URL we started the search with. Established media are interested in attracting traffic to their website, while fake media might not. Moreover, some fake accounts mimic genuine media, but have a slightly different domain, e.g., `.com.co` instead of `.com`.
- *Counts*: Statistics about the number of friends, statuses, and favorites. Established media might have higher values for these.
- *Description:* A vector representation generated by averaging the *Google News* embeddings (Mikolov et al., 2013) of all words of the profile description paragraph. These short descriptions might contain an open declaration of partisanship, i.e., left or right political ideology (bias). This could also help predict factuality as extreme partisanship often implies low factuality. In contrast, "fake news" media might just leave this description empty, while high-quality media would want to give some information about who they are.

URL We also collect additional information from the website's URL using character-based modeling and hand-crafted features. URL features are commonly used in phishing website detection systems to identify malicious URLs that aim to mislead users (Ma et al., 2009). As we want to predict a website's factuality, using URL features is justified by the fact that low-quality websites sometimes try to mimic popular news media by using a URL that looks similar to the credible source. We use the following URL-related features:

- *Character-based:* Used to model the URL by representing it in the form of a one-hot vector of character n-grams, where $n \in [2, 5]$. Note that these features are not used in the final system as they could not outperform the baseline (when used in isolation).
- *Orthographic:* These features are very effective for detecting phishing websites, as malicious URLs tend to make excessive use of special characters and sections, and ultimately end up being longer. For this work, we use the length of the URL, the number of sections and the excessive use of special characters such as digits, hyphens and dashes. In particular, we identify whether the URL contains digits, dashes or underscores as individual symbols, which were found to be useful as features for detecting phishing URLs (Basnet et al., 2014). We also check whether the URL contains short (less than three symbols) or long sections (more than ten symbols), as a high number of such sections could indicate an irregular URL.

Name	URL	Factuality	Twitter Handle	Wikipedia page
Associated Press	`http://apnews.com`	*Very High	@apnews	`~/wiki/Associated_Press`
NBC News	`http://www.nbcnews.com/`	High	@nbcnews	`~/wiki/NBC_News`
Russia Insider	`http://russia-insider.com`	Mixed	@russiainsider	`~/wiki/Russia_Insider`
Patriots Voice	`http://patriotsvoice.info/`	Low	@pegidaukgroup	N/A

Table 1: Examples of media with various factuality scores. (*In our experiments, we treat *Very High* as *High*.)

Name	URL	Bias	Twitter Handle	Wikipedia page
Loser.com	`http://loser.com`	Extreme Left	@Loser_dot_com	`~/Loser.com`
Die Hard Democrat	`http://dieharddemocrat.com/`	Left	@democratdiehard	N/A
Democracy 21	`http://www.democracy21.org/`	Center-Left	@fredwertheimer	`~/Democracy_21`
Federal Times	`http://www.federaltimes.com/`	Center	@federaltimes	`~/Federal_Times`
Gulf News	`http://gulfnews.com/`	Center-Right	@gulf_news	`~/Gulf_News`
Fox News	`http://www.foxnews.com/`	Right	@foxnews	`~/Fox_News`
Freedom Outpost	`http://freedomoutpost.com/`	Extreme Right	@FreedomOutpost	N/A

Table 2: Examples of media with various bias scores.

- *Credibility:* Model the website's URL credibility by analyzing whether it (*i*) uses `https://`, (*ii*) resides on a blog-hosting platform such as `blogger.com`, and (*iii*) uses a special top-level domain, e.g., `.gov` is for governmental websites, which are generally credible and unbiased, whereas `.co` is often used to mimic `.com`.

Web Traffic Analyzing the web traffic to the website of the medium might be useful for detecting phishy websites that come and disappear in certain patterns. Here, we only use the reciprocal value of the website's *Alexa Rank*,[5] which is a global ranking for over 30 million websites in terms of the traffic they receive.

We evaluate the above features in Section 4, both individually and as groups, in order to determine which ones are important to predict factuality and bias, and also to identify the ones that are worth further investigation in future work.

4 Experiments and Evaluation

4.1 Data

We use information about news media listed on the Media Bias/Fact Check (MBFC) website,[6] which contains manual annotations and analysis of the factuality of reporting and/or bias for over 2,000 news websites. Our dataset includes 1,066 websites for which *both* bias and factuality labels were explicitly provided, or could be easily inferred (e.g., *satire* is of low factuality).

We model factuality on a 3-point scale (*Low*, *Mixed*, and *High*),[7] and bias on a 7-point scale (*Extreme-Left*, *Left*, *Center-Left*, *Center*, *Center-Right*, *Right*, and *Extreme-Right*).

Some examples from our dataset are presented in Table 1 for factuality of reporting, and in Table 2 for bias. In both tables, we show the names of the media, as well as their corresponding Twitter handles and Wikipedia pages, which we found automatically. Overall, 64% of the websites in our dataset have Wikipedia pages, and 94% have Twitter accounts. In cases of "fake news" sites that try to mimic real ones, e.g., `ABCnews.com.co` is a fake version of `ABCnews.com`, it is possible that our Twitter extractor returns the handle for the real medium. This is where the *URL Match* feature comes handy (see above).

Table 3 provides detailed statistics about the dataset. Note that we have 1-2 orders of magnitude more media sources than what has been used in previous studies, as we already mentioned in Section 2 above.

Factuality		Bias	
Low	256	Extreme-Left	21
Mixed	268	Left	168
High	542	Center-Left	209
		Center	263
		Center-Right	92
		Right	157
		Extreme-Right	156

Table 3: Label distribution (counts) in our dataset.

---

[5] `http://www.alexa.com/`

[6] `https://mediabiasfactcheck.com`

[7] MBFC also uses *Very High* as a label, but due to its very small size, we merged it with *High*.

Source	**Feature**	**Dim.**	**Factuality**				**Bias**			
			Macro-F$_1$	**Acc.**	**MAE**	**MAE^M**	**Macro-F$_1$**	**Acc.**	MAE	MAE^M
Majority Baseline			22.47	50.84	0.73	1.00	5.65	24.67	1.39	1.71
Traffic	*Alexa rank*	1	22.46	50.75	0.73	1.00	7.76	25.70	1.38	1.71
URL	*URL structure*	12	39.30	53.28	0.68	0.81	13.50	23.64	1.65	2.06
Twitter	*created at.*	1	30.72	52.91	0.69	0.92	5.65	24.67	1.39	1.71
	has account	1	30.72	52.91	0.69	0.92	5.65	24.67	1.39	1.71
	verified	1	30.72	52.91	0.69	0.92	5.65	24.67	1.39	1.71
	has location	1	36.73	52.72	0.69	0.82	9.44	24.86	1.54	1.85
	URL match	2	39.98	54.60	0.66	0.72	10.16	25.61	**1.51**	1.97
	description	300	44.79	51.41	0.65	0.70	19.08	25.33	1.73	2.04
	counts	5	46.88	**57.22**	**0.57**	0.66	18.34	24.86	1.62	2.01
	Twitter – All	308	**48.23**	54.78	0.59	**0.64**	**21.38**	**27.77**	1.58	**1.83**
Wikipedia	*has page*	1	43.53	59.10	0.57	0.63	14.33	26.83	1.63	2.14
	table of content	300	43.95	51.04	0.60	0.65	15.10	22.96	1.86	2.25
	categories	300	46.36	53.70	0.65	0.61	25.64	32.16	1.70	2.10
	information box	300	46.39	51.14	0.71	0.65	19.79	26.85	1.68	1.99
	summary	300	51.88	58.91	0.54	0.52	30.02	37.43	1.47	**1.98**
	content	300	55.29	62.10	0.51	0.50	**30.92**	**38.61**	**1.51**	2.01
	Wikipedia – All	301	**55.52**	**62.29**	**0.50**	**0.49**	28.66	35.93	1.51	2.00
Articles	*title*	141	53.20	59.57	0.51	0.58	30.91	37.52	1.29	1.53
	body	141	**58.02**	**64.35**	**0.43**	**0.51**	**36.63**	**41.74**	**1.15**	**1.43**

Table 4: Results for factuality and bias prediction. **Bold** values indicate the best-performing feature type in its family of features, while **underlined** values indicate the best-performing feature type overall.

In order to compute the article-related features, we did the following: (*i*) we crawled 10–100 articles per website (a total of 94,814), (*ii*) we computed a feature vector for each article, and (*iii*) we averaged the feature vectors for the articles from the same website to obtain the final vector of article-related features.

4.2 Experimental Setup

We used the above features in a Support Vector Machine (SVM) classifier, training a separate model for factuality and for bias. We report results for 5-fold cross-validation. We tuned the SVM hyper-parameters, i.e., the cost C, the kernel type, and the kernel width γ, using an internal cross-validation on the training set and optimizing macro-averaged F_1. Generally, the RBF kernel performed better than the linear kernel.

We report accuracy and macro-averaged F_1 score. We also report Mean Average Error (MAE), which is relevant given the ordinal nature of both the factuality and the bias classes, and also MAE^M, which is a variant of MAE that is more robust to class imbalance. See (Baccianella et al., 2009; Rosenthal et al., 2017) for more details about MAE^M vs. MAE.

4.3 Results and Discussion

We present in Table 4 the results of using features from the different sources proposed in Section 3. We start by describing the contribution of each feature type towards factuality and bias.

We can see that the textual features extracted from the ARTICLES yielded the best performance on factuality. They also perform well on bias, being the only type that beats the baseline on MAE. These results indicate the importance of analyzing the contents of the target website. They also show that using the *titles* only is not enough, and that the article *bodies* contain important information that should not be ignored.

Overall, the WIKIPEDIA features are less useful for factuality, and perform reasonably well for bias. The best features from this family are those about the page *content*, which includes a general description of the medium, its history, ideology and other information that can be potentially helpful. Interestingly, the *has page* feature alone yields sizable improvement over the baseline, especially for factuality. This makes sense given that trustworthy websites are more likely to have Wikipedia pages; yet, this feature does not help much for predicting political bias.

Features	Macro-F$_1$	Acc.	MAE	MAE^M
MAJORITY BASELINE	22.47	50.84	0.73	1.00
FULL	59.91	65.48	0.41	0.44
FULL W/O TRAFFIC	59.90	65.39	0.41	0.43
FULL W/O TWITTER	59.52	65.10	0.41	0.47
FULL W/O URL	57.23	63.32	0.44	0.49
FULL W/O ARTICLES	56.15	63.13	0.46	0.51
FULL W/O WIKIPEDIA	55.93	63.23	0.44	0.52

Table 5: Ablation study for the contribution of each feature type for predicting the factuality of reporting.

Features	7-Way Bias				3-Way Bias			
	Macro-F$_1$	Acc.	MAE	MAE^M	Macro-F$_1$	Acc.	MAE	MAE^M
MAJORITY BASELINE	5.65	24.67	1.39	1.71	22.61	51.33	0.49	0.67
FULL	37.50	39.87	1.25	1.55	61.31	68.86	0.39	0.53
FULL W/O TRAFFIC	37.49	39.84	1.25	1.55	61.30	68.86	0.38	0.53
FULL W/O TWITTER	36.88	39.49	1.20	1.38	63.27	69.89	0.38	0.50
FULL W/O URL	36.60	39.68	1.24	1.48	60.93	68.11	0.40	0.53
FULL W/O WIKIPEDIA	34.75	37.62	1.33	1.58	59.92	66.89	0.41	0.54
FULL W/O ARTICLES	29.95	36.96	1.40	1.85	53.67	62.48	0.47	0.62

Table 6: Ablation study for the contribution of each feature type for predicting media bias.

The TWITTER features perform moderately for factuality and poorly for bias. This is not surprising, as we normally may not be able to tell much about the political ideology of a website just by looking at its Twitter profile (not its tweets!) unless something is mentioned in its *description*, which turns out to perform better than the rest of the features from this family. We can see that the *has twitter* feature is less effective than *has wiki* for factuality, which makes sense given that Twitter is less regulated than Wikipedia. Note that the *counts* features yield reasonable performance, indicating that information about activity (e.g., number of statuses) and social connectivity (e.g., number of followers) is useful. Overall, the TWITTER features seem to complement each other, as their union yields the best performance on factuality.

The URL features are better used for factuality rather than bias prediction. This is mainly due to the nature of these features, which are aimed at detecting phishing websites, as we mentioned in Section 3. Overall, this feature family yields slight improvements, suggesting that it can be useful when used together with other features.

Finally, the *Alexa rank* does not improve over the baseline, which suggests that more sophisticated TRAFFIC-related features might be needed.

4.4 Ablation Study

Finally, we performed an ablation study in order to evaluate the impact of removing one family of features at a time, as compared to the FULL system, which uses all the features. We can see in Tables 5 and 6 that the FULL system achieved the best results for factuality, and the best macro-F$_1$ for bias, suggesting that the different types of features are largely complementary and capture different aspects that are all important for making a good classification decision.

For factuality, excluding the WIKIPEDIA features yielded the biggest drop in performance. This suggests that they provide information that may not be available in other sources, including the ARTICLES, which achieved better results alone. On the other hand, excluding the TRAFFIC feature had no effect on the model's performance.

For bias, we experimented with classification on both a 7-point and a 3-point scale.[8] Similarly to factuality, the results in Table 6 indicate that WIKIPEDIA offers complementary information that is critical for bias prediction, while TRAFFIC makes virtually no difference.

---

[8] We performed the following mapping: {*Extreme-Right, Right*}→Right, {*Extreme-Left, Left*}→Left, and {*Center, Right-Center, Left-Center*}→Center

5 Conclusion and Future Work

We have presented a study on predicting factuality of reporting and bias of news media, focusing on characterizing them as a whole. These are under-studied, but arguably important research problems, both in their own right and as a prior for fact-checking systems.

We have created a new dataset of news media sources that has annotations for both tasks and is 1-2 orders of magnitude larger than what was used in previous work. We are releasing the dataset and our code, which should facilitate future research.

We have experimented with a rich set of features derived from the contents of (*i*) a sample of articles from the target news medium, (*ii*) its Wikipedia page, (*iii*) its Twitter account, (*iv*) the structure of its URL, and (*v*) information about the Web traffic it has attracted. This combination, as well as some of the types of features, are novel for this problem.

Our evaluation results have shown that most of these features have a notable impact on performance, with the articles from the target website, its Wikipedia page, and its Twitter account being the most important (in this order). We further performed an ablation study of the impact of the individual types of features for both tasks, which could give general directions for future research.

In future work, we plan to address the task as ordinal regression, and further to model the inter-dependencies between factuality and bias in a joint model. We are also interested in characterizing the factuality of reporting for media in other languages. Finally, we want to go beyond *left vs. right* bias that is typical of the Western world and to model other kinds of biases that are more relevant for other regions, e.g., *islamist vs. secular* is one such example for the Muslim World.

Acknowledgments

This research was carried out in collaboration between the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) and the Qatar Computing Research Institute (QCRI), HBKU.

We would like to thank Israa Jaradat, Kritika Mishra, Ishita Chopra, Laila El-Beheiry, Tanya Shastri, and Hamdy Mubarak for helping us with the data extraction, cleansing, and preparation.

Finally, we thank the anonymous reviewers for their constructive comments, which have helped us improve this paper.

References

Pepa Atanasova, Lluís Màrquez, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims, task 1: Check-worthiness. In *CLEF 2018 Working Notes. Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, CEUR Workshop Proceedings, Avignon, France. CEUR-WS.org.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2009. Evaluation measures for ordinal regression. In *Proceedings of the 9th IEEE International Conference on Intelligent Systems Design and Applications*, ISDA '09, pages 283–287, Pisa, Italy.

Ricardo Baeza-Yates. 2018. Bias on the web. *Commun. ACM*, 61(6):54–61.

Ramy Baly, Mitra Mohtarami, James Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018. Integrating stance detection and fact checking in a unified corpus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 21–27, New Orleans, LA, USA.

Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Pepa Atanasova, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims, task 2: Factuality. In *CLEF 2018 Working Notes. Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, CEUR Workshop Proceedings, Avignon, France. CEUR-WS.org.

Ram B Basnet, Andrew H Sung, and Quingzhong Liu. 2014. Learning to detect phishing URLs. *International Journal of Research in Engineering and Technology*, 3(6):11–24.

Ann M Brill. 2001. Online journalists embrace new marketing function. *Newspaper Research Journal*, 22(2):28.

Kevin R. Canini, Bongwon Suh, and Peter L. Pirolli. 2011. Finding credible information sources in social networks based on content and social structure. In *Proceedings of the IEEE International Conference on Privacy, Security, Risk, and Trust, and the IEEE International Conference on Social Computing*, SocialCom/PASSAT '11, pages 1–8, Boston, MA, USA.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 675–684, Hyderabad, India.

Abhijnan Chakraborty, Bhargavi Paranjape, Kakarla Kakarla, and Niloy Ganguly. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '16, pages 9–16, San Francisco, CA, USA.

Cheng Chen, Kui Wu, Venkatesh Srinivasan, and Xudong Zhang. 2013. Battling the Internet Water Army: detection of hidden paid posters. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 116–120, Niagara, Canada.

Kareem Darwish, Dimitar Alexandrov, Preslav Nakov, and Yelena Mejova. 2017a. Seminar users in the Arabic Twitter sphere. In *Proceedings of the 9th International Conference on Social Informatics*, SocInfo '17, pages 91–108, Oxford, UK.

Kareem Darwish, Walid Magdy, and Tahar Zanouda. 2017b. Improved stance prediction in a user similarity feature space. In *Proceedings of the Conference on Advances in Social Networks Analysis and Mining*, ASONAM '17, pages 145–148, Sydney, Australia.

Sohan De Sarkar, Fan Yang, and Arjun Mukherjee. 2018. Attending sentences to detect satirical fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3371–3380, Santa Fe, NM, USA.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, pages 60–67, Vancouver, Canada.

Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-based trust: Estimating the trustworthiness of web sources. *Proc. VLDB Endow.*, 8(9):938–949.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can rumour stance alone predict veracity? In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3360–3370, Santa Fe, NM, USA.

Howard Finberg, Martha L Stone, and Diane Lynch. 2002. Digital journalism credibility study. *Online News Association. Retrieved November*, 3:2003.

Jesse Graham, Jonathan Haidt, and Brian A Nosek. 2009. Liberals and conservatives rely on different sets of moral foundations. *Journal of personality and social psychology*, 96(5):1029.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M. Meyer, and Iryna Gurevych. 2018. A retrospective analysis of the fake news challenge stance-detection task. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 1859–1874, Santa Fe, NM, USA.

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2016. In search of credible news. In *Proceedings of the 17th International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, AIMSA '16, pages 172–180, Varna, Bulgaria.

Benjamin Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *CoRR*, abs/1703.09398.

Benjamin Horne, Sibel Adali, and Sujoy Sikdar. 2017. Identifying the social signals that drive online discussions: A case study of Reddit communities. In *Proceedings of the 26th IEEE International Conference on Computer Communication and Networks*, ICCCN '17, pages 1–9, Vancouver, Canada.

Benjamin D. Horne, William Dron, Sara Khedr, and Sibel Adali. 2018a. Assessing the news landscape: A multi-module toolkit for evaluating the credibility of news. In *Proceedings of the The Web Conference*, WWW '18, pages 235–238, Lyon, France.

Benjamin D. Horne, Sara Khedr, and Sibel Adali. 2018b. Sampling the news producers: A large news and feature data set for the study of the complex media landscape. In *Proceedings of the Twelfth International Conference on Web and Social Media*, ICWSM '18, pages 518–527, Stanford, CA, USA.

Clayton Hutto and Eric Gilbert. 2014. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the 8th International Conference on Weblogs and Social Media*, ICWSM '14, Ann Arbor, MI, USA.

Georgi Karadzhov, Pepa Gencheva, Preslav Nakov, and Ivan Koychev. 2017a. We built a fake news & clickbait filter: What happened next will blow your mind! In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 334–343, Varna, Bulgaria.

Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017b. Fully automated fact checking using external sources. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 344–353, Varna, Bulgaria.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3402–3413, Santa Fe, NM, USA.

David M.J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. 2018. The science of fake news. *Science*, 359(6380):1094–1096.

Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. *SIGKDD Explor. Newsl.*, 17(2):1–16.

Ying Lin, Joe Hoover, Morteza Dehghani, Marlon Mooijman, and Heng Ji. 2017. Acquiring background knowledge to improve moral value prediction. *arXiv preprint arXiv:1709.05467*.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI '16, pages 3818–3824, New York, NY, USA.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1751–1754, Melbourne, Australia.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL '17, pages 708–717, Vancouver, Canada.

Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. 2009. Identifying suspicious URLs: An application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 681–688, Montreal, Canada.

Suman Kalyan Maity, Aishik Chakraborty, Pawan Goyal, and Animesh Mukherjee. 2017. Detection of sockpuppets in social media. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 243–246, Portland, OR, USA.

Todor Mihaylov, Georgi Georgiev, and Preslav Nakov. 2015a. Finding opinion manipulation trolls in news community forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, CoNLL '15, pages 310–314, Beijing, China.

Todor Mihaylov, Ivan Koychev, Georgi Georgiev, and Preslav Nakov. 2015b. Exposing paid opinion manipulation trolls. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, RANLP '15, pages 443–450, Hissar, Bulgaria.

Todor Mihaylov, Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Georgi Georgiev, and Ivan Koychev. 2018. The dark side of news community forums: Opinion manipulation trolls. *Internet Research*.

Todor Mihaylov and Preslav Nakov. 2016. Hunting for troll comments in news community forums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 399–405, Berlin, Germany.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadjov, and James Glass. 2018. Fact checking in community forums. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI '18, pages 879–886, New Orleans, LA, USA.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '13, pages 746–751, Atlanta, GA, USA.

Lewis Mitchell, Morgan R Frank, Kameron Decker Harris, Peter Sheridan Dodds, and Christopher M Danforth. 2013. The geography of happiness: Connecting Twitter sentiment and expression, demographics, and objective characteristics of place. *PloS one*, 8(5):e64417.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 767–776, New Orleans, LA, USA.

Subhabrata Mukherjee and Gerhard Weikum. 2015. Leveraging joint interactions for credibility analysis in news communities. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 353–362, Melbourne, Australia.

Preslav Nakov, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Wajdi Zaghouani, Pepa Atanasova, Spas Kyuchukov, and Giovanni Da San Martino. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims. In *Proceedings of the Ninth International Conference of the CLEF Association: Experimental IR Meets Multilinguality, Multimodality, and Interaction*, Lecture Notes in Computer Science, pages 372–387, Avignon, France. Springer.

An T. Nguyen, Aditya Kharosekar, Matthew Lease, and Byron C. Wallace. 2018. An interpretable joint graphical model for fact-checking from crowds. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI '18, New Orleans, LA, USA.

Jeff Z. Pan, Siyana Pavlova, Chenxi Li, Ningxi Li, Yangmei Li, and Jinshuo Liu. 2018. Content based fake news detection using knowledge graphs. In *Proceedings of the International Semantic Web Conference*, ISWC '18, Monterey, CA, USA.

Symeon Papadopoulos, Kalina Bontcheva, Eva Jaho, Mihai Lupu, and Carlos Castillo. 2016. Overview of the special issue on trust and veracity of information in social media. *ACM Trans. Inf. Syst.*, 34(3):14:1–14:5.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3391–3401, Santa Fe, NM, USA.

Evaggelia Pitoura, Panayiotis Tsaparas, Giorgos Flouris, Irini Fundulaki, Panagiotis Papadakos, Serge Abiteboul, and Gerhard Weikum. 2018. On measuring bias in online information. *SIGMOD Rec.*, 46(4):16–21.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility assessment of textual claims on the web. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2173–2178, Indianapolis, IN, USA.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the Web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17, pages 1003–1012, Perth, Australia.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2018. CredEye: A credibility lens for analyzing and explaining misinformation. In *Proceedings of The Web Conference 2018*, WWW '18, pages 155–158, Lyon, France.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 231–240, Melbourne, Australia.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP '17, pages 2931–2937, Copenhagen, Denmark.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 1650–1659, Sofia, Bulgaria.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *ArXiv:1707.03264*.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, pages 502–518, Vancouver, Canada.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36.

James Thorne, Mingjie Chen, Giorgos Myrianthous, Jiashu Pu, Xiaoxuan Wang, and Andreas Vlachos. 2017. Fake news stance detection using stacked ensemble of classifiers. In *Proceedings of the EMNLP Workshop on Natural Language Processing meets Journalism*, pages 80–83, Copenhagen, Denmark.

James Thorne and Andreas Vlachos. 2018. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3346–3359, Santa Fe, NM, USA.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 809–819, New Orleans, LA, USA.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, Michal Lukasik, Kalina Bontcheva, Trevor Cohn, and Isabelle Augenstein. 2018. Discourse-aware rumour stance classification in social media using sequential classifiers. *Inf. Process. Manage.*, 54(2):273–290.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLoS ONE*, 11(3):1–29.

Legal Judgment Prediction via Topological Learning

Haoxi Zhong[*], **Zhipeng Guo**[*], **Cunchao Tu, Chaojun Xiao, Zhiyuan Liu**[†], **Maosong Sun**
Department of Computer Science and Technology
State Key Lab on Intelligent Technology and Systems
Institute for Artificial Intelligence, Tsinghua University, Beijing, China
`{zhonghx18,gzp17,xiaocj16}@mails.tsinghua.edu.cn, tucunchao@gmail.com`
`{lzy,sms}@tsinghua.edu.cn`

Abstract

Legal **J**udgment **P**rediction (**LJP**) aims to predict the judgment result based on the facts of a case and becomes a promising application of artificial intelligence techniques in the legal field. In real-world scenarios, legal judgment usually consists of multiple subtasks, such as the decisions of applicable law articles, charges, fines, and the term of penalty. Moreover, there exist topological dependencies among these subtasks. While most existing works only focus on a specific subtask of judgment prediction and ignore the dependencies among subtasks, we formalize the dependencies among subtasks as a Directed Acyclic Graph (DAG) and propose a topological multi-task learning framework, TOPJUDGE, which incorporates multiple subtasks and DAG dependencies into judgment prediction. We conduct experiments on several real-world large-scale datasets of criminal cases in the civil law system. Experimental results show that our model achieves consistent and significant improvements over baselines on all judgment prediction tasks. The source code can be obtained from `https://github.com/thunlp/TopJudge`.

1 Introduction

Legal **J**udgment **P**rediction (**LJP**) aims to predict the judgment results of legal cases according to the fact descriptions. It is a critical technique for the legal assistant system. On the one hand, LJP can provide low-cost but high-quality legal consulting services to the masses who are unfamiliar with legal terminology and the complex judgment procedures. On the other hand, it can serve as the handy reference for professionals (e.g., lawyers and judges) and improve their work efficiency.

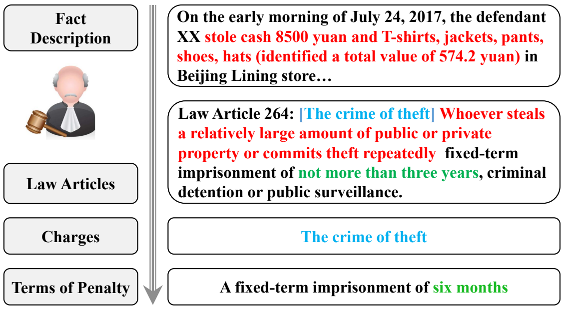

Figure 1: An illustration of the judicial logic of human judges in civil law system.

LJP has been studied for decades (Kort, 1957; Ulmer, 1963; Nagel, 1963; Keown, 1980; Segal, 1984; Lauderdale and Clark, 2012; Ye et al., 2018; Hu et al., 2018), and most existing works formalize LJP as a text classification task. For example, some works (Liu et al., 2004; Liu and Hsieh, 2006) propose to extract shallow textual features (e.g. characters, words, and phrases) for charge prediction. Katz et al. (2017) predict the US Supreme Court's decisions based on efficient features from case profiles. Luo et al. (2017) propose an attention-based neural model for charge prediction by incorporating the relevant law articles.

Despite these efforts in designing efficient features and employing advanced NLP techniques, LJP is still confronted with two major challenges:

Multiple Subtasks in Legal Judgment: Practically, legal judgment usually consists of detailed and complicated subclauses, such as charges, the term of penalty, and fines. Specifically, for those countries with the *civil law system* (e.g., China, France, and Germany), the prediction of relevant articles is also considered to be one of the fundamental subtasks, which will guide the prediction for other subtasks. In other words, all these subtasks compose the complete form of judgment pre-

---

[*] Indicates equal contribution. The order is determined by dice rolling.
[†] Corresponding author.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3540–3549
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

diction. Nevertheless, existing works on LJP usually focus on one specific subtask of judgments, which does not conform to the real scenarios. Although some methods (Luo et al., 2017) are developed to predict law articles and charges at the same time, their models are designed for a specific set of subtasks which are hard to scale to other subtasks.

Topological Dependencies between Subtasks: For human judges, there exists a strict order among the subtasks of legal judgment. As illustrated in Fig. 1, given the fact description of a specific case, a judge in the civil law system first decides which law articles are relevant to the scenario, and then determines the charges according to the instructions of relevant law articles. Based on these results, the judge further confirms the term of penalty and fines. How to simulate the judicial logic of human judges and model the topological dependencies among legal subtasks will deeply influence the creditability and interpretability of judgment prediction.

As stated above, conventional works cannot handle these two challenges due to both the limitation of specific tasks and neglecting topological dependencies. To address these issues, we propose to model the multiple subtasks in judgment prediction jointly under a novel multi-task learning framework.

We model the topological dependencies among these subtasks with a Directed Acyclic Graph (DAG), which means all subtasks are arranged in topological order. If the judgment of the j-th subtask t_j depends on the output of the i-th subtask t_i, then t_i appears earlier than t_j in such order. It is notable that such formulation provides an explicit explanation of dependency relations among subtasks.

Accordingly, we introduce topological learning for LJP and propose a unified framework, named as TOPJUDGE. Specifically, given the encoded representation of the fact description, TOPJUDGE predicts the outputs of all the subtasks following the topological order, and the output of a specific subtask will be affected by all the subtasks it depends on. In contrast with conventional multi-task learning, our model takes the explicit topological dependencies of LJP subtasks into consideration and is flexible to handle other LJP subtasks. Moreover, the topological order of legal dependencies renders our model interpretable and reliable.

To verify the effectiveness and flexibility of TOPJUDGE, we conduct a series of experiments on several real-world large-scale datasets. Experimental results show that our model achieves significant and consistent improvements over state-of-the-art models on all tasks and datasets. To summarize, we make several noteworthy contributions as follows:

(1) We are the first to explore and formalize the multiple subtasks of legal judgment under a joint learning framework. Moreover, we formulate the dependencies among the subtasks of LJP as a form of DAG and introduce this prior knowledge to enhance judgment prediction.

(2) We propose a novel judgment prediction framework, TOPJUDGE, to unify multiple subtasks and make judgment predictions through topological learning. This model can handle any form of DAG dependent subtasks, which has been verified in the experiments.

(3) We carry out experiments on several large-scale real-world datasets, and our model significantly and consistently outperforms all the baselines on all subtasks.

2 Related Work

2.1 Judgment Prediction

Employing automatic analysis techniques for legal judgment has drawn attention from researchers in the legal field for decades. Early works usually focus on analyzing existing legal cases in specific scenarios with mathematical and statistical algorithms (Kort, 1957; Ulmer, 1963; Nagel, 1963; Keown, 1980; Segal, 1984; Lauderdale and Clark, 2012).

With the development of machine learning and text mining techniques, more researchersformalize this task under text classification frameworks. Most of these studies attempt to extract efficient features from text content (Liu and Hsieh, 2006; Lin et al., 2012; Aletras et al., 2016; Sulea et al., 2017) or case annotations (e.g., dates, terms, locations, and types) (Katz et al., 2017). However, these conventional methods can only utilize shallow textual features and manually designed factors, both require massive human efforts and usually suffer from the generalization issue when applied to other scenarios.

Inspired by the success of neural networks on NLP tasks (Kim, 2014; Baharudin et al., 2010; Tang et al., 2015), researchers began to handle LJP by incorporating neural models with legal knowledge. For example, Luo et al. (2017) present an attention-based neural network that jointly models charge prediction and relevant article extraction. Hu et al. (2018) incorporate 10 discriminative legal attributes to predict few-shot and confusing charges. Nevertheless, these models are designed for specific subtasks and thus non-trivial to be extended to more subtasks of LJP with complex dependencies. Besides, Ye et al. (2018) utilize a Seq2Seq model to generate court views with fact descriptions and predicted charges in Chinese civil law.

2.2 Multi-task Learning

Multi-task learning (MTL) aims to exploit the commonalities and differences across relevant tasks by solving them at the same time. It can transfer useful information among various tasks and has been applied to a wide range of areas, including NLP (Collobert and Weston, 2008), speech recognition (Deng et al., 2013), and computer vision (Girshick, 2015; Mao et al., 2017).

There have been numerous successful usages of MTL in NLP tasks. Most works follow the *hard parameter sharing* setting by sharing representations or some encoding layers among relevant tasks. For example, Collobert and Weston (2008) use shared word embeddings in solving part-of-speech tagging and semantic role labeling tasks. Liu et al. (2015) share the encoding layers of input queries to address query classification and information retrieval. Dong et al. (2015) and Luong et al. (2016) propose to share encoders or decoders to improve one (many) to many neural machine translation. Firat et al. (2016) propose to share attention mechanism in multi-way, multilingual machine translation. Besides *hard parameter sharing*, *soft parameter sharing* is another common approach in MTL. It assumes that each task owns its specific parameters and the distance between parameters in different tasks should be close to each other. For example, Duong et al. (2015) employ L2 distance for regularization, while Yang and Hospedales (2017) use the trace norm. Liu et al. (2016) introduce gates among task-specific RNN layers to control the information flow. Ruder et al. (2017) introduce a model which can decide the amount of sharing between different NLP tasks. There are also some works focusing on increasing tasks (Hashimoto et al., 2017) or handing unlabeled data (Augenstein et al., 2018).

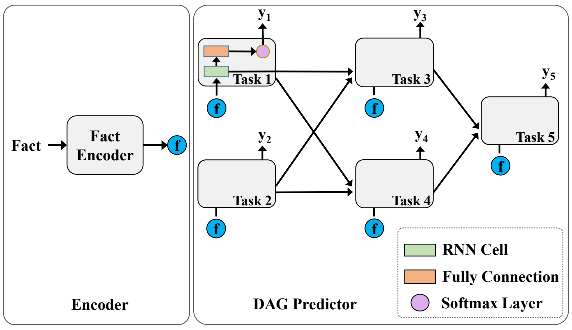

Figure 2: The framework of TopJudge.

In this work, we introduce a topological learning framework TOPJUDGE to handle multiple subtasks in LJP. Different to conventional MTL models which focus on how to share parameters among relevant tasks, TOPJUDGE models the explicit dependencies among these subtasks with scalable DAG forms.

3 Method

In the following parts, we will first give the essential definitions of LJP task. We then introduce the DAG dependencies of the subtasks in LJP. And finally, we describe the neural encoder for fact representation and the judgment predictor for the subtasks with DAG dependencies. The overall framework of TopJudge has been shown in Fig 2.

3.1 Problem Formulation

We will focus on the LJP tasks in civil law. Suppose the fact description of a case is a word sequence $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, where n is the length of $\mathbf{x}$ and each word x_i comes from a fixed vocabulary W. Based on the fact description $\mathbf{x}$, the task of LJP T aims to predict judgment results of applicable law articles, charges, term of penalty, fines and so on. Formally, we assume T contains $|T|$ subtasks, i.e., $T = \{t_1, t_2, \dots, t_{|T|}\}$, each of which is a classification task. For the i-th subtask $t_i \in T$, we aim to predict the corresponding result $\mathbf{y}_i \subseteq Y_i$, where Y_i is a subtask-specific label set. Take the subtask of charge prediction for example, the corresponding label set should contain *Theft*, *Traffic Violation*, *Intentional Homicide* and so on.

3.2 DAG Dependencies of Subtasks

We assume that the dependencies among multiple subtasks of LJP form a DAG. As a result, the task list T should satisfy topological constraints. Formally, we use the notation $t_i \lhd t_j$ to denote that

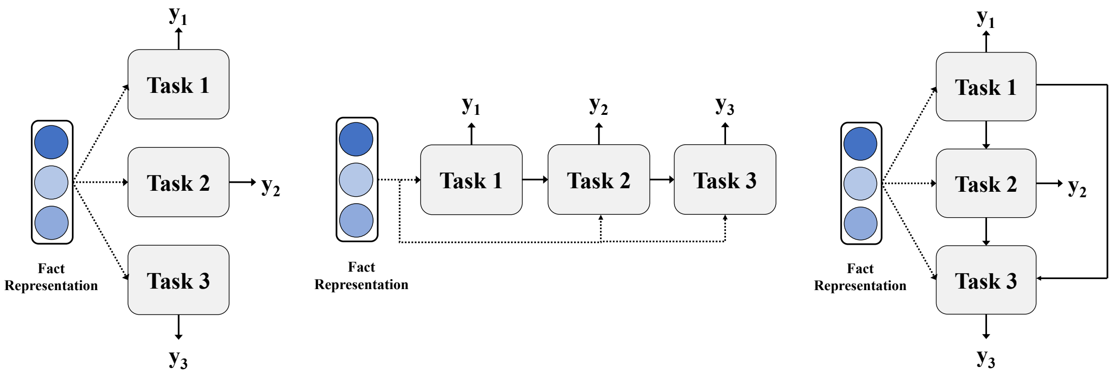

(a) Typical MTL form of DAG (b) Sequential form of DAG (c) General form of DAG

Figure 3: Three typical forms of DAG dependencies.

the j-th subtask depends on the i-th subtask, and $D_j = \{t_i \mid t_i \lhd t_j\}$ to denote the dependency set. The task list T can be ordered to satisfy the following constraint

$$i < j, \quad \forall (i, j) \in \{(i, j) \mid t_i \in D_j\}. \tag{1}$$

We demonstrate the flexibility of our formulation by describing two special cases: (1) As shown in Fig. 3 (a), if no dependencies exist, i.e., $D_j = \emptyset$, it corresponds to the typical MTL setting where we simultaneously make predictions for all subtasks. (2) As shown in Fig. 3 (b), if each task only depends on its previous task, i.e., $D_j = \{t_{j-1}\}$, it forms a sequential learning process.

3.3 Neural Encoder for Fact Descriptions

We employ a fact encoder to generate the fact description's vector representation as the input of TOPJUDGE. In the following part, we briefly introduce an encoder based on Convolutional Neural Networks (CNN) (Kim, 2014).

Taking a word sequence $\mathbf{x}$ as input, the CNN encoder computes the text representation through three layers, i.e., lookup layer, convolution layer and pooling layer.

Lookup We first convert each word x_i in $\mathbf{x}$ into its word embedding $\mathbf{x}_i \in \mathbb{R}^k$, where k is the dimension of word embeddings. The word embedding sequence is then represented as

$$\hat{\mathbf{x}} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}. \tag{2}$$

Convolution A convolution operation involves a convolution matrix $\mathbf{W} \in \mathbb{R}^{m \times (h \times k)}$, which is applied to a sliding window of length h with number of filters m to produce a feature map by

$$\mathbf{c}_i = \mathbf{W} \cdot \mathbf{x}_{i:i+h-1} + \mathbf{b}, \tag{3}$$

where $\mathbf{x}_{i:i+h-1}$ is the concatenation of word embeddings within the i-th window and $\mathbf{b} \in \mathbb{R}^m$ is the bias vector. By applying convolution over each window, we obtain $\mathbf{c} = \{\mathbf{c}_1, \ldots, \mathbf{c}_{n-h+1}\}$.

Pooling We apply per-dimension max-pooling over $\mathbf{c}$ and obtain the final fact representation $\mathbf{d} = [d_1, \ldots, d_m]$ by

$$d_t = \max(\mathbf{c}_{1,t}, \ldots, \mathbf{c}_{n-h+1,t}), \forall t \in [1, m]. \tag{4}$$

3.4 Judgment Predictor over DAG

Based on the DAG assumption, we obtain an ordered task list $T^* = [t_1, t_2, \ldots, t_{|T|}]$. For each task $t_j \in T$, we aim to predict its judgment result $\mathbf{y}_j$ based on the fact representation vector $\mathbf{d}$ and the judgment results of its dependent tasks.

For prediction, we employ a specific LSTM cell for each task and get the output of each task in the topological order. More specifically, for each task $t_j \in T$, we obtain its final judgment result through three steps, i.e., cell initialization, task-specific representation, and prediction.

Cell Initialization As stated above, the prediction result of t_j will be conditioned on the fact representation $\mathbf{d}$ and the outputs of all dependent tasks $\mathbf{y}_k, \forall t_k \in D_j$. Hence, we have

$$\begin{bmatrix} \bar{\mathbf{h}}_j \\ \bar{\mathbf{c}}_j \end{bmatrix} = \sum_{t_i \in D_j} \left(\mathbf{W}_{i,j} \begin{bmatrix} \mathbf{h}_i \\ \mathbf{c}_i \end{bmatrix} \right) + \mathbf{b}_j \tag{5}$$

Here, $\mathbf{h}_i$ and $\mathbf{c}_i$ are the hidden state and memory cell of t_i. $\bar{\mathbf{h}}_j$ and $\bar{\mathbf{c}}_j$ are the initial hidden state and memory cell of t_j. $\mathbf{W}_{i,j}$ and $\mathbf{b}_j$ are transformation matrices and bias vectors specific to t_i and t_j.

Task-Specific Representation Taking the fact representation $\mathbf{d}$, the initial hidden state $\bar{\mathbf{h}}_j$, and the initial memory cell $\bar{\mathbf{c}}_j$ as inputs, we process them with an LSTM cell (Hochreiter and Schmidhuber, 1997).

We regard the final hidden state $\mathbf{h}_j$ as the task-specific representation of task t_j. The last cell state $\mathbf{c}_j$ is used to compose the initial hidden state for the downstream tasks by Eq. 5

Prediction With the representation $\mathbf{h}_j$, we apply an affine transformation followed by softmax and obtain the final prediction as

$$\hat{\mathbf{y}}_j = \text{softmax}\left(\mathbf{W}_j^p \mathbf{h}_j + \mathbf{b}_j^p\right). \tag{6}$$

Here, $\mathbf{W}_j^p$ and $\mathbf{b}_j^p$ are parameters specific to task t_j.

With the prediction result $\hat{\mathbf{y}}_j$, we minimize the cross-entropy between $\hat{\mathbf{y}}_j$ and $\mathbf{y}_j$ as follows:

$$\mathcal{L}_j(\hat{\mathbf{y}}_j, \mathbf{y}_j) = -\sum_{k=1}^{|Y_j|} \mathbf{y}_{j,k} \log(\hat{\mathbf{y}}_{j,k}). \tag{7}$$

3.5 Training

We use cross-entropy loss for each subtask and sum up losses to train TOPJUDGE:

$$\mathcal{L} = \sum_{j=1}^{|T|} \lambda_j \mathcal{L}_j(\hat{\mathbf{y}}_j, \mathbf{y}_j), \tag{8}$$

where λ_j is the weight factor for each subtask t_j. The DAG dependencies of subtasks ensure that our model is differentiable and can be trained in an end-to-end fashion. In practice, we set all weights λ_j to 1, and employ Adam (Kingma and Ba, 2015) for optimization. We also apply dropout (Srivastava et al., 2014) on the fact representation to prevent overfitting.

4 Experiments

To evaluate the proposed TOPJUDGE framework, we conduct a series of experiments on LJP over three large-scale datasets of criminal cases in China. We select three representative judgment prediction subtasks for comparison, including law articles, charges, and the terms of penalty.

4.1 Dataset Construction

As there are no publicly available LJP datasets in previous works, we collect and construct three different LJP datasets, including **CJO**, **PKU**, and **CAIL**. CJO consists of criminal cases published by the Chinese government from China Judgement Online[1]. PKU contains criminal cases published by Peking University Law Online[2]. CAIL (Chinese AI and Law Challenge) is another criminal case dataset for competition released by the Supreme People's Court of China[3]. The details of CAIL can be found in Xiao et al. (2018).

Datasets	CJO	PKU	CAIL
Cases	1,007,744	175,744	113,536
Law Articles	98	68	105
Charges	99	64	122
Term of Penalty	11	11	11

Table 1: The statistics of different datasets.

For all datasets we mentioned above, as the documents are well-structured and human-annotated, we can easily extract *fact descriptions*, *applicable law articles*, *charges* and *the terms of penalty* from each document using regular expressions. We have manually checked a randomly sampled set of cases, and extraction errors are negligible.

In real-world scenarios, there are some cases with multiple defendants and multiple charges, which will increase the complexity of judgment prediction. As our model aims to explore the effectiveness of considering topological dependencies between various subtasks, we filter out these cases and leave them as our future work.

Meanwhile, there are also some infrequent charges and law articles, such as *money laundering*, *smuggling of nuclear materials* and *tax dodge*. We filter out these infrequent charges and law articles and only keep those with frequencies greater than 100. For the term of penalty, we divide the terms into non-overlapping intervals. We list detailed statistics of these datasets in Table 1.

4.2 Baselines

For comparison, we employ the following text classification models and judgment prediction methods as baselines:

TFIDF+SVM: We employ term-frequency inverse document frequency (TFIDF) (Salton and Buckley, 1988) to extract word features and utilize SVM (Suykens and Vandewalle, 1999) for text classification.

CNN: We employ CNN with multiple filter widths (Kim, 2014) for fact encoding and classification.

Hierarchical LSTM (HLSTM): Tang et al. (2015) employs hierarchical neural networks to learn document representations in sentiment clas-

---

[1] `http://wenshu.court.gov.cn/`

[2] `http://www.pkulaw.com/`

[3] `http://cail.cipsc.org.cn/index.html`

	Tasks	Law Articles				Charges				The Term of Penalty			
	Metrics	Acc.	MP	MR	F_1	Acc.	MP	MR	F_1	Acc.	MP	MR	F_1
Single	TFIDF+SVM	82.4	45.5	26.7	30.2	82.2	47.4	27.9	31.3	48.5	36.0	16.7	16.5
	CNN	92.5	46.9	38.4	40.0	92.3	41.2	32.3	33.7	57.4	35.6	22.2	22.7
	HLSTM	91.4	38.6	37.3	36.9	91.8	37.8	36.0	35.2	56.1	22.5	25.0	23.3
Multi	Fact-Law Att.	93.5	50.9	45.6	45.9	93.4	47.2	41.4	41.5	56.3	31.3	26.4	26.7
	PM	93.7	51.9	44.1	44.9	93.6	45.5	39.1	39.3	58.2	38.2	24.9	26.8
	CNN-MTL	94.3	53.0	46.0	46.9	94.1	48.5	41.7	42.5	58.7	39.9	28.8	29.4
	HLSTM-MTL	92.4	45.5	41.4	41.0	92.3	41.9	36.6	35.9	54.9	30.6	26.6	26.4
Ours	**TOPJUDGE**	**94.4**	**53.9**	**47.3**	**48.2**	**94.9**	**53.9**	**48.2**	**49.1**	**58.8**	**40.2**	**32.9**	**32.8**

Table 2: Judgment prediction results on CJO.

	Tasks	Law Articles				Charges				The Term of Penalty			
	Metrics	Acc.	MP	MR	F_1	Acc.	MP	MR	F_1	Acc.	MP	MR	F_1
Single	TFIDF+SVM	80.9	51.3	32.6	36.4	81.0	53.4	35.4	38.7	45.3	30.4	17.4	17.2
	CNN	93.1	64.3	52.6	54.3	93.3	61.9	49.3	51.1	57.6	24.1	23.1	23.3
	HLSTM	91.7	54.4	53.4	50.9	91.9	52.5	48.9	47.3	54.3	20.6	21.7	19.0
Multi	Fact-Law Att.	93.9	68.1	63.4	63.5	94.2	65.8	58.5	58.7	55.7	27.7	27.4	26.5
	PM	94.4	69.6	61.0	62.2	94.3	65.1	56.2	57.2	58.2	36.2	26.4	27.1
	CNN-MTL	95.0	73.8	64.9	66.0	95.0	70.7	60.6	61.7	**58.4**	36.0	28.7	28.9
	HLSTM-MTL	93.9	71.2	64.6	65.1	93.8	67.8	60.0	60.7	55.4	31.3	26.2	25.7
Ours	**TOPJUDGE**	**95.4**	**76.4**	**67.6**	**68.4**	**95.6**	**75.9**	**69.6**	**70.9**	57.8	**38.9**	**32.1**	**31.8**

Table 3: Judgment prediction results on PKU.

sification. Based on this work, we employ an LSTM for sentence representations and another one to obtain the representation of complete fact descriptions.

Fact-Law Attention Model: Luo et al. (2017) proposes a neural charge prediction model by capturing the interaction between fact descriptions and applicable laws with attention mechanism.

Pipeline Model (PM): To demonstrate the advantage of TOPJUDGE on modeling subtasks jointly, we also implement a pipelined method for comparison. Here, we train 3 separate CNN classifiers for law articles, charges, and term of penalty. For each subtask, the input is the concatenation of the fact representation and the embeddings for predicted labels of previous subtasks.

Besides, we compare our model with conventional MTL methods that do not consider the dependencies among subtasks as in Fig. 3 (a). These methods are denoted as **CNN-MTL** and **HLSTM-MTL**, where we implement the fact encoder as in Fig. 2 using CNN or HLSTM respectively.

4.3 Experimental Settings

As the case documents are written in Chinese with no space between words, we employ THULAC (Sun et al., 2016) for word segmentation. Afterward, we adopt the Skip-Gram model (Mikolov et al., 2013) to pre-train word embeddings on these case documents, with embedding size set to 200 and frequency threshold set to 25.

For all models, we set the fact representation and task-specific representation size to 256. Meanwhile, we set the maximum sentence length to 128 words and maximum document length to 32 sentences.

For training, the learning rate of Adam optimizer is 10^{-3}, and the dropout probability is 0.5. We also set the batch size to 128 for all models. We train every model for 16 epochs, and evaluate the final model on the testing set.

We employ *accuracy* (Acc.), *macro-precision* (MP), *macro-recall* (MR) and *macro-F_1* (F_1) as evaluation metrics. Here, the macro-precision/recall/F_1 are calculated by averaging the precision/recall/F_1 of each category.

4.4 Results and Analysis

We evaluate the performance on three LJP subtasks, including law articles (denoted as t_1), charges (denoted as t_2), and the terms of penalty (denoted as t_3). Experimental results are shown in Tables 2, 3, and 4. Note that, we implement TOPJUDGE with the dependency relationship in Fig. 3 (c), i.e.,

$$D_1 = \phi, D_2 = \{t_1\}, D_3 = \{t_1, t_2\}. \quad (9)$$

	Tasks	Law Articles				Charges				The Term of Penalty			
	Metrics	Acc.	MP	MR	F_1	Acc.	MP	MR	F_1	Acc.	MP	MR	F_1
Single	TFIDF+SVM	60.1	54.9	45.3	46.3	59.2	53.9	45.0	45.7	28.4	22.9	20.0	18.1
	CNN	81.4	74.4	64.1	65.7	80.7	77.3	65.5	67.2	28.8	34.7	27.8	28.6
	HLSTM	-	-	-	-	-	-	-	-	-	-	-	-
Multi	Fact-Law Att.	70.9	64.8	63.6	59.1	68.7	66.1	65.3	60.1	36.5	29.9	27.6	27.1
	PM	84.7	80.7	68.6	70.8	83.6	81.6	70.0	72.1	**40.0**	**37.4**	32.0	31.6
	CNN-MTL	84.5	80.0	68.1	70.3	83.4	81.6	69.1	71.6	39.5	37.2	32.3	31.3
	HLSTM-MTL	-	-	-	-	-	-	-	-	-	-	-	-
Ours	**TOPJUDGE**	**86.3**	**81.9**	**71.1**	**73.4**	**85.7**	**83.4**	**76.0**	**78.3**	38.3	36.1	**33.1**	**32.1**

Table 4: Judgment prediction results on CAIL. Note that, "-" means the model does not converge within 128 epochs.

This means that the prediction of charges depends on law articles, and the prediction of term of penalty depends on both law articles and charges. Such explicit dependencies conform to the judicial logic of human judges, which will be verified in later sections. These results show that:

(1) The proposed TOPJUDGE model outperforms other baselines significantly on most subtasks and datasets. It demonstrates the effectiveness and robustness of our proposed framework.

(2) Compared with conventional single-task models, e.g., CNN and HLSTM, MTL methods take advantage of the correlation among relevant subtasks and thus achieve promising improvements. It indicates the importance of modeling LJP subtasks jointly.

(3) Moreover, TOPJUDGE significantly outperforms typical MTL models, especially on the prediction of charges and the terms of penalty. It verifies the rationality and importance of modeling dependencies over LJP subtasks with DAG.

4.5 Ablation Analysis

Tasks	t_1		t_2		t_3	
Metrics	Acc.	F_1	Acc.	F_1	Acc.	F_1
TOPJUDGE	**95.4**	**68.4**	**95.6**	**70.9**	**57.8**	**31.8**
$- t_3 \lhd t_1$	95.2	67.7	95.4	70.3	57.4	31.2
$- t_2 \lhd t_1$	94.8	64.7	94.9	60.2	57.0	31.6
ϕ	94.7	64.4	94.9	60.1	57.8	27.6

Table 5: Ablation analysis on PKU.

To further illustrate the significance of legal dependencies and explore how the DAG dependencies influence the performance, we evaluate the performance of TOPJUDGE under various DAG architectures. Using Eq. 9 as the full dependencies, we remove the dependency of $t_3 \lhd t_1$ (law articles and term of penalty, corresponding to the sequential form in Fig. 3), $t_2 \lhd t_1$ (law articles and charges), and all dependencies respectively. Results are summarized in Table 5.

We observe that the performance of TOPJUDGE decreases on all tasks after removing either dependency. More specifically, when we dropped dependencies $t_3 \lhd t_1$ and $t_2 \lhd t_1$ respectively, significant decreases are observed for t_3 and t_2 correspondingly. This demonstrates that incorporating dependencies is beneficial for relevant subtasks, verifying its guiding role in the civil law system.

Meanwhile, we note that there are two main differences between TOPJUDGE and traditional multi-task models, namely the **Cell Initialization** and the **Task-Specific Representation**. We can see that if we eliminate **Cell Initialization** from TOPJUDGE, the dependencies will not be represented in the model and it will become similar to **CNN-MTL**. If we eliminate the **Task-Specific Representation** from TOPJUDGE, TOPJUDGE will become the same as the **Pipeline Model**. In a word, the main improvement of our models comes from the combination.

4.6 Case Study

We give some intuitive examples to demonstrate the significance of TOPJUDGE on LJP subtasks.

As shown in Table 6, case 1 is about negligently causing a fire. The fact description of this case states "The defendant pulled up weeds in the fields and piled them up in haphazard stacks. Afterward, he lighted them up and triggered the forest fires..." TOPJUDGE predicts all judgments correctly, while CNN-MTL fails to predict the charge and term of penalty. Moreover, CNN-MTL obtains conflicting judgments, i.e., "*crime of arson*" and "*1-2 years*", due to its neglecting of dependencies of these subtasks. According to the legal provisions of law article 115, the crime of arson should be sentenced to more than 10 years.

Methods	Law	Charge	Term (months)
CNN-MTL	115	Arson(×)	12-24(×)
TOPJUDGE	115	Negligently Causing a Fire	0-6
CNN-MTL	293	Intentional Damage to Property(×)	12-24(×)
TOPJUDGE	293	Affray	0-6

Table 6: Example cases and their prediction results.

Case 2 in Table 6 is another evidence of the insufficiency of conventional MTL on LJP. This case is about picking quarrels and provoking troubles. Both CNN-MTL and TOPJUDGE succeed to predict the relevant law articles (i.e., law article 293 of the crime of affray). However, CNN-MTL is confused between "*crime of affray*" and "*crime of intentional destruction or damage of properties*", two charges similar to each other. Conversely, TOPJUDGE can utilize the prediction result of law articles and consequently prevent this confusion.

To summarize, modeling the explicit dependencies among various subtasks can remarkably help the LJP model address the issue of predicting conflicting results.

4.7 Error Analysis

Prediction errors induced by our proposed model can be traced down into the following causes.

Data Imbalance. For the subtasks of law articles and charges, our model achieves more than 90% on *accuracy*, while only about 60% for *macro-F1*. This issue is much more severe on the subtask of the term of penalty, which our model yields a poor performance of only 30% *macro-F1*. The bad performance is mainly due to the imbalance of category labels, e.g., there are only a few training instances where the term is "*life imprisonment or death penalty*". Most judgment prediction approaches perform poorly (especially for *Recall*) on these labels as listed in Fig. 4. Instance weighting schemes can be introduced to address this issue in future works.

Incomplete Information. Following existing LJP works, we predict the final judgment according to the fact descriptions, which is incomplete as compared to the whole materials relevant to this case. In Chinese Law, there are certain circumstances under which the sentence can be shortened. For example, minors usually receive a lightened penalty, and those guilty of misdemeanors are allowed for a secured pending trial while paying a security deposit. However, such information is not included in the fact descriptions. The lack of such information also raises difficulties for judgment prediction, especially for the prediction of the term of penalty. In Fig. 4, we can see that the highest error rate comes from the cases with a short term of penalty. Our model fails to distinguish the cases with no penalty and those with 0-6 months term of imprisonment.

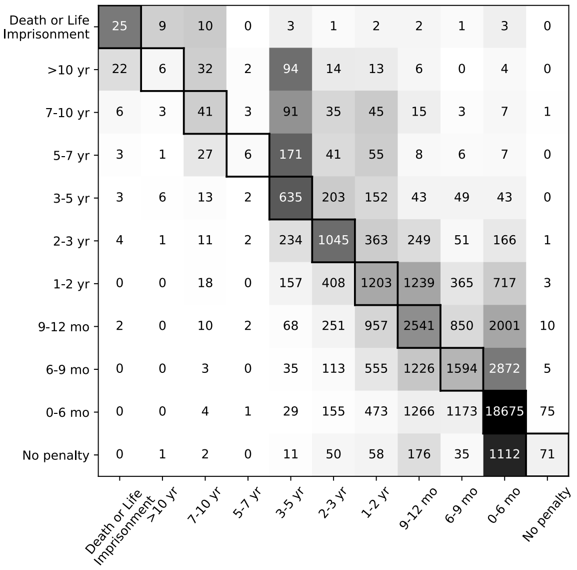

	Death or Life Imprisonment	>10 yr	7-10 yr	5-7 yr	3-5 yr	2-3 yr	1-2 yr	9-12 mo	6-9 mo	0-6 mo	No penalty
Death or Life Imprisonment	25	9	10	0	3	1	2	2	1	3	0
>10 yr	22	6	32	2	94	14	13	6	0	4	0
7-10 yr	6	3	41	3	91	35	45	15	3	7	1
5-7 yr	3	1	27	6	171	41	55	8	6	7	0
3-5 yr	3	6	13	2	635	203	152	43	49	43	0
2-3 yr	4	1	11	2	234	1045	363	249	51	166	1
1-2 yr	0	0	18	0	157	408	1203	1239	365	717	3
9-12 mo	2	0	10	2	68	251	957	2541	850	2001	10
6-9 mo	0	0	3	0	35	113	555	1226	1594	2872	5
0-6 mo	0	0	4	1	29	155	473	1266	1173	18675	75
No penalty	0	1	2	0	11	50	58	176	35	1112	71

Figure 4: The confusion matrix in the subtask of predicting the term of penalty on the PKU dataset. The rows denote the ground truth while the columns denote the prediction results.

5 Conclusion

In this paper, we focus on the task of legal judgment prediction (LJP) and address multiple subtasks of judgment predication with a topological learning framework. To be specific, we formalize the explicit dependencies over these subtasks in a DAG form, and propose a novel MTL framework, TOPJUDGE, by integrating the DAG dependencies. Experimental results on three LJP subtasks and three different datasets show that our TOPJUDGE outperforms all single-task baselines and conventional MTL models consistently and significantly.

In the future, we will seek to explore the following directions: (1) We will explore more LJP subtasks and more scenarios of cases such as multiple defendants and charges to investigate the effectiveness of TOPJUDGE. (2) We will explore how to incorporate into LJP the temporal factors, which are not considered in this work.

6 Acknowledgements

This work is supported by the National Natural Science Foundation of China (NSFC No. 61572273, 61772302) and the research fund of Powerlaw Inc. for AI+Law Technology. This work is also funded by China Association for Science and Technology (2016QNRC001). Tu is also supported by China Postdoctoral Innovative Talent Support Programme.

References

Nikolaos Aletras, Dimitrios Tsarapatsanis, Daniel Preotiuc-Pietro, and Vasileios Lampos. 2016. Predicting judicial decisions of the european court of human rights: A natural language processing perspective. *PeerJ Computer Science*, 2.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. In *Proceedings of NAACL*.

Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. 2010. A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology*, 1(1):4–20.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

Li Deng, Geoffrey Hinton, and Brian Kingsbury. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *Proceedings of ICASSP*, pages 8599–8603.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of ACL*, volume 1, pages 1723–1732.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of ACL*, volume 2, pages 845–850.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of NAACL*, pages 866–875.

Ross Girshick. 2015. Fast r-cnn. In *Proceedings of ICCV*, pages 1440–1448.

Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of EMNLP*, pages 1923–1933.

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

Zikun Hu, Xiang Li, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. 2018. Few-shot charge prediction with discriminative legal attributes. In *Proceedings of COLING*.

Daniel Martin Katz, Michael J Bommarito II, and Josh Blackman. 2017. A general approach for predicting the behavior of the supreme court of the united states. *Plos one*, 12(4).

R Keown. 1980. Mathematical models for legal prediction. *Computer/LJ*, 2:829.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Fred Kort. 1957. Predicting supreme court decisions mathematically: A quantitative analysis of the "right to counsel" cases. *American Political Science Review*, 51(1):1–12.

Benjamin E Lauderdale and Tom S Clark. 2012. The supreme court's many median justices. *American Political Science Review*, 106(4):847–866.

Wanchen Lin, Tsung Ting Kuo, and Tung Jia Chang. 2012. Exploiting machine learning models for chinese legal documents labeling, case classification, and sentencing prediction. In *Processdings of ROCLING*, page 140.

Chaolin Liu, Cheng Tsung Chang, and Jim How Ho. 2004. Case instance generation and refinement for case-based criminal summary judgments in chinese. *Journal of Informationence & Engineering*, 20(4):783–800.

Chaolin Liu and Chwen Dar Hsieh. 2006. Exploring phrase-based classification of judicial documents for criminal charges in chinese. In *Proceedings of ISMIS*, pages 681–690.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of IJCAI*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of NAACL*, pages 912–921.

Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang, and Dongyan Zhao. 2017. Learning to predict charges for criminal cases with legal basis. In *Proceedings of EMNLP*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of ICLR*.

Jiayuan Mao, Tete Xiao, Yuning Jiang, and Zhimin Cao. 2017. What can help pedestrian detection? In *Proceedings of CVPR*, pages 3127–3136.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Stuart S Nagel. 1963. Applying correlation analysis to case prediction. *Texas Law Review*, 42:1006.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Jeffrey A Segal. 1984. Predicting supreme court cases probabilistically: The search and seizure cases, 1962-1981. *American Political Science Review*, 78(4):891–900.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Octavia Maria Sulea, Marcos Zampieri, Mihaela Vela, and Josef Van Genabith. 2017. Exploring the use of text classification in the legal domain. In *Proceedings of ASAIL workshop*.

Maosong Sun, Xinxiong Chen, Kaixu Zhang, Zhipeng Guo, and Zhiyuan Liu. 2016. Thulac: An efficient lexical analyzer for chinese.

Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP*, pages 1422–1432.

S Sidney Ulmer. 1963. Quantitative analysis of judicial processes: Some practical and theoretical applications. *Law and Contemporary Problems*, 28:164.

Chaojun Xiao, Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Yansong Feng, Xianpei Han, Zhen Hu, Heng Wang, et al. 2018. Cail2018: A large-scale legal dataset for judgment prediction. *arXiv preprint arXiv:1807.02478*.

Yongxin Yang and Timothy Hospedales. 2017. Deep multi-task representation learning: A tensor factorisation approach. In *Proceedings of ICLR*.

Hai Ye, Xin Jiang, Zhunchen Luo, and Wenhan Chao. 2018. Interpretable charge predictions for criminal cases: Learning to generate court views from fact descriptions. In *Proceedings of NAACL*.

Hierarchical CVAE for Fine-Grained Hate Speech Classification

Jing Qian, Mai ElSherief, Elizabeth Belding, William Yang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
{jing_qian,mayelsherif,ebelding,william}@cs.ucsb.edu

Abstract

Existing work on automated hate speech detection typically focuses on binary classification or on differentiating among a small set of categories. In this paper, we propose a novel method on a fine-grained hate speech classification task, which focuses on differentiating among 40 hate groups of 13 different hate group categories. We first explore the Conditional Variational Autoencoder (CVAE) (Larsen et al., 2016; Sohn et al., 2015) as a discriminative model and then extend it to a hierarchical architecture to utilize the additional hate category information for more accurate prediction. Experimentally, we show that incorporating the hate category information for training can significantly improve the classification performance and our proposed model outperforms commonly-used discriminative models.

1 Introduction

The impact of the vast quantities of user-generated web content can be both positive and negative. While it improves information accessibility, it also can facilitate the propagation of online harassment, such as hate speech. Recently, the Pew Research Center[1] reported that "roughly four-in-ten Americans have personally experienced online harassment, and 63% consider it a major problem. Beyond the personal experience, two-thirds of Americans reported having witnessed abusive or harassing behavior towards others online."

In response to the growth in online hate, there has been a trend of developing automatic hate speech detection models to alleviate online harassment (Warner and Hirschberg, 2012; Waseem and Hovy, 2016). However, a common problem with these methods is that they focus on coarse-grained classifications with only a small set of categories. To the best of our knowledge, the existing work on hate speech detection formulates the task as a classification problem with no more than seven classes. Building a model for more fine-grained multiclass classification is more challenging since it requires the model to capture finer distinctions between each class.

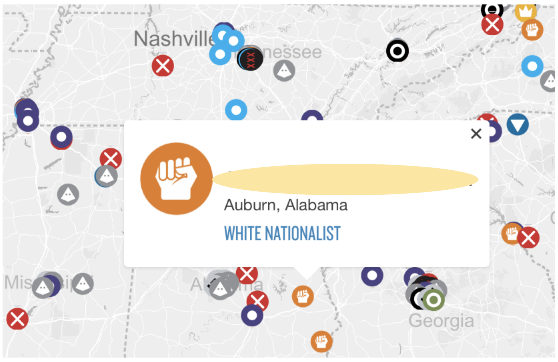

Figure 1: A portion of the hate group map published by the Southern Poverty Law Center (SPLC). Each marker represents a hate group. The markers with the same pattern indicate the corresponding hate groups share the same ideology. The white box shows an example of a hate group in Auburn, Alabama under the category of "White Nationalist". Due to the sensitivity of the data, we mask the name of the group.

Moreover, fine-grained classification is necessary for fine-grained hate speech analysis. Figure 1 is a portion of the hate group map published by the Southern Poverty Law Center (SPLC)[2], where a hate group is defined as "an organization that based on its official statements or principles, the statements of its leaders, or its activities has beliefs or practices that attack or malign an entire class of people, typically for their immutable characteristics." The SPLC divides the 954 hate groups in the United States into 17 categories ac-

---

[1] http://www.pewinternet.org/2017/07/11/online-harassment-2017/

[2] https://www.splcenter.org

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3550–3559
Brussels, Belgium, October 31 - November 4, 2018.

cording to their ideologies (e.g. Racist Skinhead, Anti-Immigrant, and others). The SPLC monitors hate groups throughout the United States by a variety of methodologies to determine the activities of groups and individuals: reviewing hate group publications and reports by citizens, law enforcement, field sources and the news media, and conducting their own investigations. Therefore, building automatic classification models to differentiate between the social media posts from different hate groups is both challenging and of practical significance.

In this paper, we propose a fine-grained hate speech classification task that separates tweets posted by 40 hate groups of 13 different hate group categories. Although CVAE is commonly used as a generative model, we find it can achieve competitive results and tends to be more robust when the size of the training dataset decreases, compared to the commonly used discriminative neural network models. Based on this insight, we design a Hierarchical CVAE model (HCVAE) for this task, which leverages the additional hate group category (ideology) information for training. Our contributions are three-fold:

- This is the first paper on fine-grained hate speech classification that attributes hate groups to individual tweets.
- We propose a novel Hierarchical CVAE model for fine-grained tweet hate speech classification.
- Our proposed model improves the Micro-F1 score of up to 10% over the baselines.

In the next section, we outline the related work on hate speech detection, fine-grained text classification, and Variational Autoencoder. In Section 3, we explore the CVAE as a discriminative model, and our proposed method is described in Section 4. Experimental results are presented and discussed in Section 5. Finally, we conclude in Section 6.

2 Related Work

2.1 Hate Speech Detection

An extensive body of work has been dedicated to automatic hate speech detection. Most of the work focuses on binary classification. Warner and Hirschberg (2012) differentiate between anti-semitic or not. Gao et al. (2017), Zhong et al. (2016), and Nobata et al. (2016) differentiate between abusive or not. Waseem and Hovy (2016), Burnap and Williams (2016), and Davidson et al. (2017) focus on three-way classification. Waseem and Hovy (2016) classify each input tweet as racist hate speech, sexist hate speech, or neither. Burnap and Williams (2016) build classifiers for hate speech based on race, sexual orientation or disability, while Davidson et al. (2017) train a model to differentiate among three classes: containing hate speech, only offensive language, or neither. Badjatiya et al. (2017) use the dataset provided by Waseem and Hovy (2016) to do three-way classification. Our work is most closely related to (Van Hee et al., 2015), which focuses on fine-grained cyberbullying classification. However, this study only focuses on seven categories of cyberbullying while our dataset consists of 40 classes. Therefore, our classification task is much more fine-grained and challenging.

2.2 Fine-grained Text Classification

Our work is also related to text classification. Convolutional Neural Networks (CNN) have been successfully applied to the text classification task. Kim (2014) applies CNN at the word level while Zhang et al. (2015) apply CNN at the character level. Johnson and Zhang (2015) exploit the word order of text data with CNN for accurate text categorization. Socher et al. (2013) introduces the Recursive Neural Tensor Network for text classification. Recurrent Neural Networks (RNN) are also commonly used for text classification (Tai et al., 2015; Yogatama et al., 2017). Lai et al. (2015) and Zhou et al. (2015) further combine RNN with CNN. Tang et al. (2015) and Yang et al. (2016) exploit the hierarchical structure for document classification. Tang et al. (2015) generate from the sentence representation to the document representation while Yang et al. (2016) generate from the word-level attention to the sentence-level attention. However, the division of the hierarchies in our HCVAE is according to semantic levels, rather than according to document compositions. We generate from the category-level representations to the group-level representations. Moreover, the most commonly used datasets by these works (Yelp reviews, Yahoo answers, AGNews, IMDB reviews (Diao et al., 2014)) have no more than 10 classes.

2.3 Variational Autoencoder

Variational Autoencoder (VAE) (Kingma and Welling, 2013) has achieved competitive results in

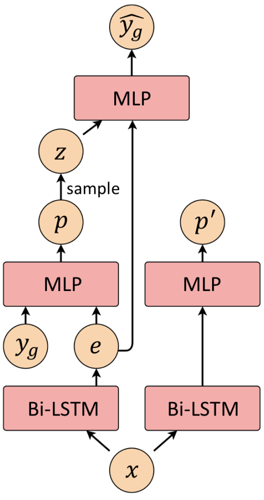

Figure 2: The structure of the baseline CVAE model during training. Bi-LSTM is a bidirectional LSTM layer. MLP is a Multilayer Perceptron. x is the embedded input text. y_g is the ground truth hate group label. $\hat{y_g}$ is the output prediction of the hate group. p is the posterior distribution of the latent variable z while p' is the prior distribution of z. Note that this structure is used for training. During testing, the posterior network is replaced by the prior network to compute $\hat{y_g}$ and thus y_g is not available during testing. Refer to Section 3 for detailed explanation.

many complicated generation tasks, such as handwritten digits (Kingma and Welling, 2013; Salimans et al., 2015), faces (Kingma and Welling, 2013; Rezende et al., 2014), and machine translation (Zhang et al., 2016). CVAE (Larsen et al., 2016; Sohn et al., 2015) is an extension of the original VAE framework that incorporates conditions during generation. In addition to image generation, CVAE has also been successfully applied to several NLP tasks, such as dialog generation (Zhao et al., 2017). Although so far CVAE has always been used as a generative model, we explore the performance of the CVAE as a discriminative model and further propose a hierarchical CVAE model, which exploits the hate group category (ideology) information for training.

3 CVAE Baseline

We formulate our classification task as the following equation:

$$Obj = \sum_{(x,y_g)\in X} \log p(y_g|x) \qquad (1)$$

where x, y_g are the tweet text and hate group label respectively, X is the dataset. Instead of directly parameterizing $p(y_g|x)$, it can be further written as the following equation:

$$p(y_g|x) = \int_z p(y_g|z,x)p(z|x)dz \qquad (2)$$

where z is the latent variable. Since the integration over z is intractable, we instead try to maximize the corresponding evidence lower bound (ELBO):

$$\begin{aligned} ELBO =& E[\log p(y_g|z,x)] - \\ & D_{KL}[q(z|x,y_g)||p(z|x)] \end{aligned} \qquad (3)$$

where D_{KL} is the KullbackLeibler (KL) divergence. $p(y_g|z,x)$ is the likelihood distribution, $q(z|x,y_g)$ is the posterior distribution, and $p(z|x)$ is the prior distribution. These three distributions are parameterized by $p_\varphi(y_g|z,x)$, $q_\alpha(z|x,y_g)$, and $p_\beta(z|x)$. Therefore, the training loss function is:

$$\begin{aligned} \mathcal{L} =& L_{REC} + L_{KL} \\ =& E_{z\sim p_\alpha(z|x,y_g)}[-\log p_\varphi(y_g|z,x)] + \\ & D_{KL}[q_\alpha(z|x,y_g)||p_\beta(z|x)] \end{aligned} \qquad (4)$$

The above loss function consists of two parts. The first part L_{REC} is the reconstruction loss. Optimizing L_{REC} can push the predictions made by the posterior network and the likelihood network closer to the ground truth labels. The second part L_{KL} is the KL divergence loss. Optimizing it can push the output distribution of the prior network and that of the posterior network closer to each other, such that during testing, when the ground truth label y_g is no longer available, the prior network can still output a reasonable probability distribution over z.

Figure 2 illustrates the structure of the CVAE model during training (the structure used for testing is different). The likelihood network $p_\varphi(y_g|z,x)$ is a Multilayer Perceptron (MLP). The structure of both the posterior network $q_\alpha(z|x,y_g)$ and the prior network $p_\beta(z|x)$ is a bidirectional Long Short-Term Memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) layer followed by an MLP. The Bi-LSTM is used to encode the tweet text. The only difference between the posterior and the prior network is that for the posterior network the input for the MLP is the encoded text concatenated with the group label while for the prior network only the encoded text is fed forward. During testing, the posterior network will be replaced by the prior network to generate the distribution over the latent variable (i.e. p' will replace

p). Thus during testing, the ground truth labels will not be used to make predictions.

We assume the latent variable z has a multivariate Gaussian distribution: $p = \mathcal{N}(\mu, \Sigma)$ for the posterior network, and $p' = \mathcal{N}(\mu', \Sigma')$ for the prior network. The detailed computation process is as follows:

$$e = f(x) \tag{5}$$

$$\mu, \Sigma = s(y_g \oplus e) \tag{6}$$

$$\mu', \Sigma' = s'(f'(x)) \tag{7}$$

where f is the Bi-LSTM function and e is the output of the Bi-LSTM layer at the last time step. s is the function of the MLP in the posterior network and s' is that of the MLP in the prior network. The notation $\oplus$ means concatenation. The latent variables z and z' are randomly sampled from $\mathcal{N}(\mu, \Sigma)$ and $\mathcal{N}(\mu', \Sigma')$, respectively. During training, the input for the likelihood network is z:

$$\hat{y_g} = w(z) \tag{8}$$

where w is the function of the MLP in the likelihood network. During testing, the prior network will substitute for the posterior network. Thus for testing, the input for the likelihood is z' instead of z:

$$\hat{y_g} = w(z') \tag{9}$$

4 Our Approach

One problem with the above CVAE method is that it utilizes the group label for training, but ignores the available hate group category (ideology) information of the hate speech. As mentioned in Section 1, hate groups can be divided into different categories in terms of ideologies. Each hate group belongs to a specific hate group category. Considering this hierarchical structure, the hate category information can potentially help the model to better capture the subtle differences between the hate speech from different hate groups. Therefore, we extend the baseline CVAE model to incorporate the category information. In this case, the objective function is as follows:

$$\begin{aligned} Obj = & \sum_{(x,y_c,y_g)\in X} \log p(y_c, y_g|x) \\ = & \sum_{(x,y_c,y_g)\in X} \log p(y_c|x) + \log p(y_g|x, y_c) \end{aligned} \tag{10}$$

where y_c is the hate group category label and

$$p(y_c|x) = \int_{z_c} p(y_c|z_c, x)p(z_c|x)dz_c \tag{11}$$

$$p(y_g|x, y_c) = \int_{z_g} p(y_g|z_g, x, y_c)p(z_g|x, y_c)dz_g \tag{12}$$

where z_c and z_g are latent variables. Therefore, the ELBO of our method is the sum of the ELBOs of $\log p(y_c|x)$ and $\log p(y_g|x, y_c)$:

$$\begin{aligned} ELBO = & E[\log p(y_c|z_c, x)] - \\ & D_{KL}[q(z_c|x, y_c)||p(z_c|x)] + \\ & E[\log p(y_g|z_g, x, y_c)] - \\ & D_{KL}[q(z_g|x, y_c, y_g)||p(z_g|x, y_c)] \end{aligned} \tag{13}$$

During testing, the prior networks will substitute the posterior networks and the ground truth labels y_c and y_g are not utilized. Hence the prior $p(z_g|x, y_c)$ in the above equation cannot be parametrized as a network that directly takes the ground truth label y_c and x as inputs. Instead, we parameterize it as shown in the right part of Figure 3. We assume that u' is trained to be a latent representation of y_c, so we use u' and x as inputs for this prior network.

According to the ELBO above, the corresponding loss function $\mathcal{L}$ is the combination of the loss function for the category classification ($\mathcal{L}_c$) and that for the group classification ($\mathcal{L}_g$).

$$\begin{aligned} \mathcal{L} = & \mathcal{L}_c + \mathcal{L}_g, \text{ where} \\ \mathcal{L}_c = & E_{z_c \sim q_\alpha(z_c|x,y_c)}[-\log p_\varphi(y_c|z_c, x)] + \\ & D_{KL}[q_\alpha(z_c|x, y_c)||p_\beta(z_c|x)], \text{ and} \\ \mathcal{L}_g = & E_{z_g \sim q_\eta(z_g|x,y_c,y_g)}[-\log p_\theta(y_g|z_g, x, y_c)] + \\ & D_{KL}[q_\eta(z_g|x, y_c, y_g)||p_\gamma(z_g|x, u)] \end{aligned} \tag{14}$$

Figure 3 shows the structure of our model for training. By assuming the latent variables z_c and z_g have multivariate Gaussian distributions, the actual outputs of the posterior and prior networks are the mean and variance: $p_c = \mathcal{N}(\mu_c, \Sigma_c)$, $p_g = \mathcal{N}(\mu_g, \Sigma_g)$ for the two posterior networks, and $p'_c = \mathcal{N}(\mu'_c, \Sigma'_c)$, $p'_g = \mathcal{N}(\mu'_g, \Sigma'_g)$ for the two prior networks. Note that in addition to these four distributions, there is another distribution $p_x = \mathcal{N}(\mu_x, \sigma_x)$ as shown in Figure 3. This distribution is generated only with the input text x, so it provides the basic distribution (in order to avoid confusion, we call it the basic distribution instead

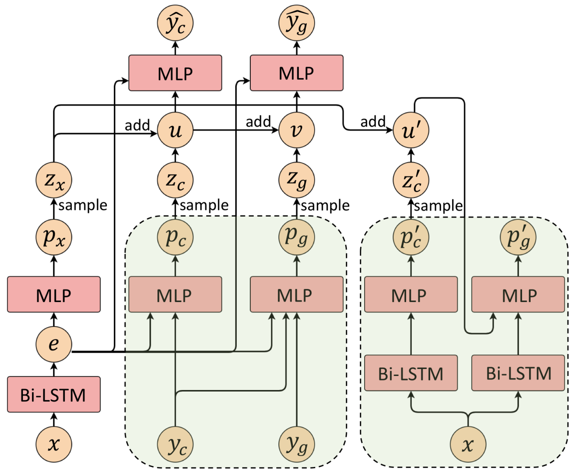

Figure 3: The structure of the HCVAE during training. Bi-LSTM is a bidirectional LSTM layer. MLP is a Multilayer Perceptron. x is the embedded input text. y_c and y_g are the ground truth hate category label and hate group label. $\hat{y_c}$ and $\hat{y_g}$ are the output predictions of the hate category and hate group. z_c, z_g, and z'_c are latent variables. In the left dotted box are two posterior networks. In the right dotted box are two prior networks. Note that this structure is used for training. During testing, the posterior networks will be substituted by the posterior networks (i.e. the left dotted box will be substituted with the right one) to compute $\hat{y_c}$ and $\hat{y_g}$. Thus y_c and y_g are not available during testing. Refer to Section 4 for detailed explanation.

of the prior distribution) for both two posterior networks and two prior networks. With this basic distribution, the two posterior networks only need to capture the additional signals learned from x and labels. Similarly, the prior networks only need to learn the additional signals learned by the posterior networks. The detailed computation process during training is shown as the following equations:

$$e = f(x) \tag{15}$$

$$\mu_x, \Sigma_x = s_x(e) \tag{16}$$

$$\mu_c, \Sigma_c = s_c(y_c \oplus e) \tag{17}$$

$$\mu_g, \Sigma_g = s_g(y_c \oplus y_g \oplus e) \tag{18}$$

$$\mu'_c, \Sigma'_c = s'_c(f'_c(x)) \tag{19}$$

where f is the Bi-LSTM function and e is the output of the Bi-LSTM layer at the last time step. s_x, s_c, s_g, and s'_c are the functions of four different MLPs. f'_c is the Bi-LSTM function in the prior network $p_\beta(z_c|x)$. The latent variables z_x, z_c, z_g, and z'_c are randomly sampled from the Gaussian distributions $\mathcal{N}(\mu_x, \Sigma_x)$, $\mathcal{N}(\mu_c, \Sigma_c)$, $\mathcal{N}(\mu_g, \Sigma_g)$, and $\mathcal{N}(\mu'_c, \Sigma'_c)$ respectively. As mentioned above, p_x is the basic distribution while p_c, p_g, p'_c, and p'_g are trained to capture the additional signals. Therefore, z_x is added to z_c and z'_c, then the results u and u' are further added to z_g and z'_g, respectively, as shown in the following equations:

Algorithm 1 Train & Test Algorithm

```
function TRAIN($X$)
    randomly initialize network parameters;
    for $epoch = 1, E$ do
        for $(text, category, group)$ in $X$ do
            get embeddings $x$ and one-hot vectors $y_c$, $y_g$;
            compute $e$ with the Bi-LSTM;
            compute $\mu_x$, $\Sigma_x$, $\mu_c$, $\Sigma_c$, $\mu_g$, $\Sigma_g$;
            sample $z_x = reparameterize(\mu_x, \Sigma_x)$;
            sample $z_c = reparameterize(\mu_c, \Sigma_c)$;
            sample $z_g = reparameterize(\mu_g, \Sigma_g)$;
            $u = z_x + z_c$;
            $v = u + z_g$;
            compute $\hat{y_c}$ and $\hat{y_g}$ according to Eq. 24- 25;
            $\mathcal{L}_{REC} = BCE(\hat{y_c}, y_c) + BCE(\hat{y_g}, y_g)$;
            compute $\mu'_c$, $\Sigma'_c$;
            sample $z'_c = reparameterize(\mu'_c, \Sigma'_c)$;
            $u' = z_x + z'_c$;
            compute $\mu'_g$, $\Sigma'_g$;
            $\mathcal{L}_{KL} = D_{KL}(p_c||p'_c) + D_{KL}(p_g||p'_g)$;
            $\mathcal{L} = \mathcal{L}_{KL} + \mathcal{L}_{REC}$;
            update network parameters on $\mathcal{L}$;
        end for
    end for
end function

function TEST($X$)
    for $text$ in $X$ do
        get embeddings $x$;
        compute $e$ with the Bi-LSTM;
        compute $\mu_x$, $\Sigma_x$, $\mu'_c$, $\Sigma'_c$;
        sample $z_x = reparameterize(\mu_x, \Sigma_x)$;
        sample $z'_c = reparameterize(\mu'_c, \Sigma'_c)$;
        $u' = z_x + z'_c$;
        compute $\mu'_g$, $\Sigma'_g$;
        sample $z'_g = reparameterize(\mu'_g, \Sigma'_g)$;
        $v' = u' + z'_g$;
        compute $\hat{y_c}$ and $\hat{y_g}$ according to Eq. 26- 28;
    end for
end function
```

$$u' = z_x + z'_c \tag{20}$$

$$\mu'_g, \Sigma'_g = s'_g(u' \oplus f'_g(x)) \tag{21}$$

$$u = z_x + z_c \tag{22}$$

$$v = u + z_g \tag{23}$$

where f'_g is the Bi-LSTM function and s'_g is the function of the MLP in the prior network $p_\gamma(z_g|x, u)$. $+$ is element-wise addition. During training, u and v are fed into the likelihood networks:

$$\hat{y_c} = w_c(e \oplus u) \tag{24}$$

$$\hat{y_g} = w_g(e \oplus v) \tag{25}$$

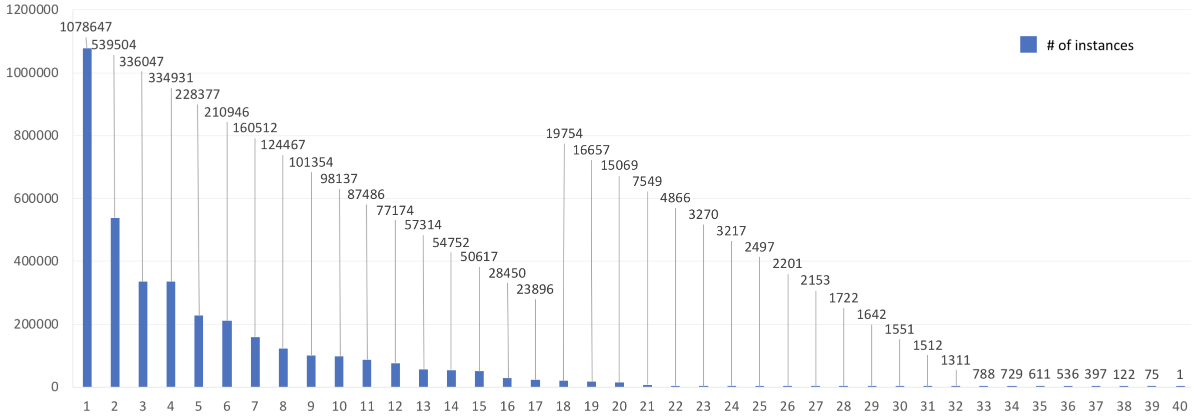

Figure 4: The distribution of the data for 40 hate groups. The X-Axis is each hate group. The Y-Axis is the number of tweets collected for each hate group.

where w_c and w_g are the functions of the MLPs in two likelihood networks. During testing, the prior networks will substitute the posterior networks, so the latent variable z'_g is randomly sampled from the Gaussian distributions $\mathcal{N}(\mu'_g, \Sigma'_g)$, and the last four equations above (Equation 22 - 25) will be replaced by the following ones:

$$\hat{y_c} = w_c(e \oplus u') \tag{26}$$

$$v' = u' + z'_g \tag{27}$$

$$\hat{y_g} = w_g(e \oplus v') \tag{28}$$

Algorithm 1 illustrates the complete training and testing process. BCE refers to the Binary Cross Entropy loss. Note that during training, the ground truth category labels and group labels are fed to the posterior network to generate latent variables. But during testing, the latent variables are generated by the prior network, which only utilizes the texts as inputs.

5 Experiments

5.1 Dataset

We collect the data from 40 hate group Twitter accounts of 13 different hate ideologies, e.g., white nationalist, anti-immigrant, racist skinhead, among others. The detailed themes and core values behind each hate ideology are discussed in the SPLC ideology section.[3] For each hate ideology, we collect a set of Twitter handles based on hate groups identified by the SPLC center.[4] For each hate ideology, we select the top three handles in terms of the number of followers. Due to ties, there are four different groups in several categories of our dataset. The dataset consists of all the content (tweets, retweets, and replies) posted with each account from the group's inception date, as early as 07-2009, until 10-2017. Each instance in the dataset is a tuple of (tweet text, hate category label, hate group label). The complete dataset consists of approximately 3.5 million tweets. Note that due to the sensitive nature of the data, we anonymously reference the Twitter handles for each hate group by using IDs throughout this paper. The distribution of the data is illustrated in Figure 4.

5.2 Experimental Settings

In addition to the discriminative CVAE model described in Section 3, we implement for other baseline methods and an upper bound model as follows.

Support Vector Machine (SVM): We implement an SVM model with linear kernels. We use L2 regularization and the coefficient is 1. The input features are the Term Frequency Inverse Document Frequency (TF-IDF) values of up to 2-grams.

Logistic Regression (LR): We implement the Logistic Regression model with L2 regularization. The penalty parameter C is set to 1. Similar to the SVM model, we use TF-IDF values of up to 2-grams as features.

Character-level Convolutional Neural Network (Char-CNN): We implement the character-level CNN model for text classification as described in (Zhang et al., 2015). It is 9 layers deep with 6 convolutional layers and 3 fully-connected layers.

---

[3] https://www.splcenter.org/fighting-hate/extremist-files/ideology

[4] https://www.splcenter.org/fighting-hate/extremist-files/groups

Dataset	Complete			Subset		
Metric	Macro-F1	Micro-F1	Weighted-F1	Macro-F1	Micro-F1	Weighted-F1
upper bound	.697	.904	.881	–	–	–
SVM + tf-idf	.653	.834	.835	**.521**	.771	.772
LR + tf-idf	.586	.787	.792	.494	.727	.736
Char-CNN	.604	.840	.853	.457	.730	.744
Bi-LSTM	.627	.767	.745	.353	.599	.570
CVAE	.520	.799	.830	.453	.774	.784
HCVAE (our)	**.664**	**.844**	**.858**	.469	**.787**	**.799**

Table 1: Experimental results. Complete: The performance achieved when 90% of the entire dataset is used for training. Subset: The performance achieved when only 10% of the dataset is used for training. The best results are in bold.

The configurations of the convolutional layers are kept the same as those in (Zhang et al., 2015).
Bi-LSTM: The model consists of a bidirectional LSTM layer followed by a linear layer. The embedded tweet text x is fed into the LSTM layer and the output at the last time step is then fed into the linear layer to predict $\hat{y_g}$.
Upper Bound: The upper bound model also consists of a bidirectional LSTM layer followed by a linear layer. The difference between this model and Bi-LSTM is that it takes the tweet text x along with the ground truth category label y_c as input during both training and testing. The LSTM layer is used to encode x. The encoding result is concatenated with the ground truth category label and then fed into the linear layer to give the prediction of the hate group $\hat{y_g}$. Since it utilizes y_c for prediction during testing, this model sets an upper bound performance for our method.

For the baseline CVAE, Bi-LSTM, the upper bound model, and the HCVAE, we use randomly initialized word embeddings with size 200. All the neural network models are optimized by the Adam optimizer with learning rate 1e-4. The batch size is 20. The hidden size of all the LSTM layers in these models is 64 and all the MLPs are three-layered with non-linear activation function Tanh. For the CVAE and the HCVAE, the size of the latent variables is set to 256.

All the baseline models and our model are evaluated on two datasets. We first use the complete dataset for training and testing. 90% of the instances are used for training and the rest for testing. In order to evaluate the robustness of the baseline models and our model, we also randomly selected a subset (10%) of the original dataset for training while the testing dataset is fixed. Since the upper bound model is used to set an upper bound on performance, we do not evaluate it on the smaller training dataset. We use Macro-F1, Micro-F1, and Weighted-F1 to evaluate the classification results. As shown in Figure 4, the dataset is highly imbalanced, which causes problems for training the above models. In order to alleviate this problem, we use a weighted random sampler to sample instances from the training data. However, the testing dataset is not sampled, so the distribution of the testing dataset remains the same as that of the original dataset. This allows us to evaluate the models' performance on the data with a realistic distribution.

5.3 Experimental Results

The experimental results are shown in Table 1. The testing dataset keeps the imbalanced distribution of the original data, so the Macro-F1 score of each method is significantly lower than the Micro-F1 score and the Weighted-F1 score. Comparing the performance of the Bi-LSTM model and that of the baseline CVAE model shows that although CVAE is traditionally used as a generative model, it can also achieve competitive performance as a discriminative model. All the methods achieve lower F1 scores when using the smaller training dataset. Due to the imbalanced distribution of our dataset, half of the 40 groups has less than 1k tweets in the smaller training dataset, which leads to the sharp decline in the Macro-F1 scores of all the methods. However, the performance of both CVAE-based models degrades less than that of the other two neural network models (the Bi-LSTM model and the Char-CNN model) according to the Micro-F1 Weighted-F1 scores. These two CVAE-based models tend to be more robust when the size of the training dataset is smaller. The difference between the CVAE-based models and the other two models is that both the Bi-LSTM model and the Char-CNN model directly compress the input text into a fixed-length latent

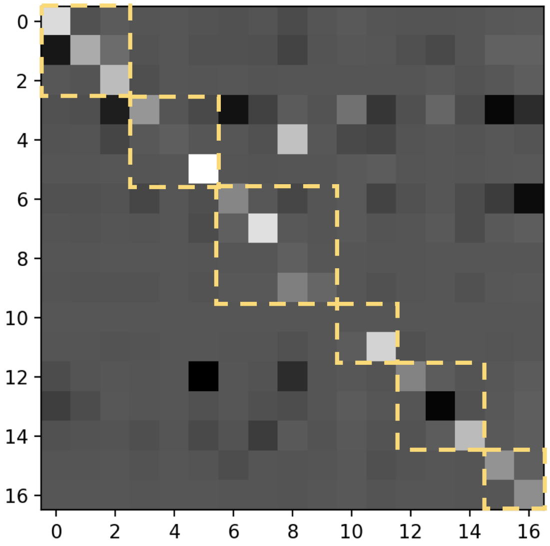

Figure 5: This figure compares a subset (17 hate groups of 6 categories) of the predictions made by the baseline CVAE and the HCVAE. The X and Y axes are the index of each hate group. The hate groups of the same category are grouped together as shown in the dashed squares. The color of the grid (i,j) is mapped from $r_{hcvae}(i,j) - r_{cvae}(i,j)$, where $r(i,j)$ is the fraction of the group i's instances among the instances predicted as the group j. A higher difference value corresponds to a lighter color. A grid darker than the background is mapped from a negative value while a grid lighter than the background is mapped from a positive one.

variable while the CVAE model explicitly models the posterior and likelihood distributions over the latent variable. The result is that, during testing, the inference strategy of the Bi-LSTM model and the Char-CNN model is actually comparing the compressed text to the compressed instances in the training dataset while the strategy of the CVAE-based models is to compare the prior distributions over the latent variable. Compared with the compressed text, prior distributions may capture higher level semantic information and thus enable better generalization.

By further utilizing the hate category label for training, the HCVAE outperforms the baseline CVAE on all three metrics. Figure 5 illustrates the difference between the predictions made by the HCVAE and the CVAE. As shown in the figure, most of the lighter girds are in the dashed squares while most of the darker girds are outside. This indicates that the HCVAE model tends to correct the prediction of the CVAE's misclassified instances to be closer to the ground truth. In most cases, the misclassified instances can be directly corrected to the ground truth (the diagonal). In other cases, they are not corrected to the ground truth but are corrected to the hate groups of the same categories as the ground truth (the dashed squares). This shows that additional ideology information is useful for the model to better capture the subtle differences among tweets posted by different hate groups.

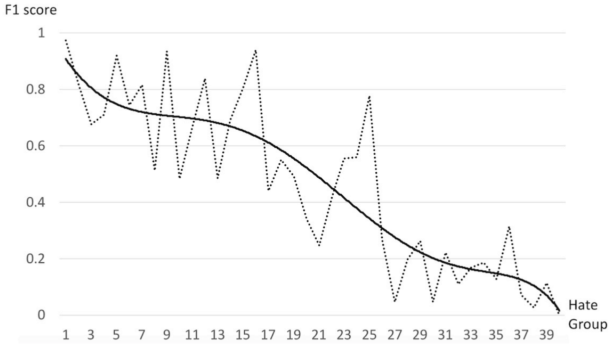

Figure 6: The F1 score achieved by our method on each hate group. The Y-axis is the F1 score. The X-axis is the hate group sorted by the number of instances in the dataset from larger to smaller. The dashed line shows the F1 scores and the solid line is the corresponding trendline.

Although our method outperforms the baseline methods, there is still a gap between its performance and the upper bound performance. We analyze this in the following section.

5.4 Error Analysis

As mentioned above, in some cases our model misclassified the tweet as a group under the same category as the ground truth. Take, for instance, the tweet from *Group1*: *The only good #muslim is a #dead #muslim*. Our HCVAE model predicts it as from *Group2*. But both *Group1* and *Group2* are under the category "neo nazi". One possible reason for this kind of error is that the imbalance of the dataset adversely affects the performance of our method. Figure 6 shows the F1 scores achieved by the HCVAE on each group. The performance of the model tends to be much lower when the number of the group's training instances decreases. Although we use the weighted random sampler to alleviate the problem of the imbalanced dataset, repeating the very limited data instances (less than 3k) of a group during training cannot really help the posterior and prior networks to give a reasonable distribution that can generalize well on unseen inputs. Therefore, when the model comes into the instances in the testing dataset, the performance can be less satisfactory. This is a common problem for all the methods, which is the cause of the significantly lower Macro-F1 scores.

Another type of error is caused by the noisy dataset. Take, for instance, the tweet from *Group3* under the category "ku klux klan": *we must secure the existence of our people and future for the White Children !!* Our model classifies it as from *Group4* under the category "neo nazi", which makes sense but is an error.

6 Conclusion

In this paper, we explore the CVAE as a discriminative model and find that it can achieve competitive results. In addition, the performance of the CVAE-based models tend to be more stable than that of the others when the dataset gets smaller. We further propose an extension of the basic discriminative CVAE model to incorporate the hate group category (ideology) information for training. Our proposed model has a hierarchical structure that mirrors the hierarchical structure of the hate groups and the ideologies. We apply the HCVAE to the task of fine-grained hate speech classification, but this Hierarchical CVAE framework can be easily applied to other tasks where the hierarchical structure of the data can be utilized.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Pete Burnap and Matthew L Williams. 2016. Us and Them: Identifying Cyber Hate on Twitter across Multiple Protected Characteristics. *EPJ Data Science*, 5(1):11.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 193–202. ACM.

Lei Gao, Alexis Kuppersmith, and Ruihong Huang. 2017. Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 774–782.

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

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. 2016. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pages 1558–1566. JMLR. org.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286.

Tim Salimans, Diederik Kingma, and Max Welling. 2015. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1556–1566.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2015. Detection and Fine-grained Classification of Cyberbullying Events. In *RANLP'15: International Conference Recent Advances in Natural Language Processing*, pages 672–680.

William Warner and Julia Hirschberg. 2012. Detecting Hate Speech on the World Wide Web. In *ACL'12: Proceedings of the 2nd Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Biao Zhang, Deyi Xiong, Hong Duan, Min Zhang, et al. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–664.

Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J Miller, and Cornelia Caragea. 2016. Content-Driven Detection of Cyberbullying on the Instagram Social Network. In *IJCAI'16: Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 3952–3958.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Residualized Factor Adaptation for Community Social Media Prediction Tasks

Mohammadzaman Zamani,[1] **H. Andrew Schwartz,**[1] **Veronica E. Lynn,**[1]
Salvatore Giorgi,[2] and **Niranjan Balasubramanian**[1]
[1] Computer Science Department, Stony Brook University
[2]Department of Psychology, University of Pennsylvania
`mzamani@cs.stonybrook.edu`

Abstract

Predictive models over social media language have shown promise in capturing community outcomes, but approaches thus far largely neglect the socio-demographic context (e.g. age, education rates, race) of the community from which the language originates. For example, it may be inaccurate to assume people in Mobile, Alabama, where the population is relatively older, will use words the same way as those from San Francisco, where the median age is younger with a higher rate of college education. In this paper, we present *residualized factor adaptation*, a novel approach to community prediction tasks which both (a) effectively integrates community attributes, as well as (b) adapts linguistic features to community attributes (factors). We use eleven demographic and socioeconomic attributes, and evaluate our approach over five different community-level predictive tasks, spanning health (heart disease mortality, percent fair/poor health), psychology (life satisfaction), and economics (percent housing price increase, foreclosure rate). Our evaluation shows that *residualized factor adaptation* significantly improves 4 out of 5 community-level outcome predictions over prior state-of-the-art for incorporating socio-demographic contexts.

1 Introduction

Adapting to human factors has been shown to benefit NLP tasks, especially in tasks that involve predictions over individual social media posts (e.g., sentiment (Hovy, 2015), sarcasm, and stance detection (Lynn et al., 2017)). The main idea behind these approaches is that knowing who wrote a piece of text can help models better understand how to process it. This paper develops methods that apply this idea to community-level prediction tasks, which require making decisions over posts from a community of users. Many community-level outcomes and community-wide language are linked to socio-demographic factors (age, gender, race, education, income levels) with many social scientific studies supporting their predictive value (Cohen et al., 2003), and should therefore affect how a model treats social media-based language features. For example, a high prevalence of the word "bike" in San Francisco, CA might be a signal that exercise is common in the area, while its high prevalence in Mobile, Alabama might indicate greater interest in motor bikes. We present a method for building language-based predictive models which integrate in and adapt to attributes of the communities generating the language.

This work aims to unify two different approaches developed for adapting to human factors and use them for incorporating community attributes in community-level prediction tasks: (1) *residualized controls:* whereby a model is trained in two steps: first over the factors/controls and then fitting the language to the residuals of the control model (Zamani and Schwartz, 2017), and (2) *user-factor adaptation:* whereby linguistic features are adapted, or treated differently, based on the continuous-valued factors of the authors of the features (Lynn et al., 2017).

Combining factor adaptation (FA) and residualized control (RC) into RFA is a non-trivial task. The intent behind both methods are quite different: whereas RC attempts to address the inherent heterogeneity between robust control variables and noisy linguistic variables, FA enables a model to treat linguistic features differently depending on the factors. From a statistical learning perspective, RC separates inference over controls from inference over language (model level integration), while FA brings controls and language together and makes the inference as one single step (data level integration). Additionally, FA has stricter bounds in the number of factors it can accommodate because each new factor has a multiplicative

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3560–3569
Brussels, Belgium, October 31 - November 4, 2018.

effect on the number of learned parameters. On the other hand, each new factor for RC typically only adds one new parameter. Here, we endeavour to develop RFA such that it achieves the benefits of both approaches with little lost to the limitations. RFA inherits the challenges of the FA method with feature explosion. We address this through a systematic exploration of both feature and factor selection.

The main contributions of this work include: (1) the introduction of residualized factor adaptation which effectively combines extra-linguistic and language features, (2) the first empirical evaluation of applying factor adaptation for community-level prediction tasks, (3) analysis of the impact of the size of factors and factor selection in adaptation, and (4) state-of-the-art accuracies for each of the five tasks for which we evaluate RFA.

2 Background

Social media provides easy access to a vast amount of language written by a diverse group of users, making it an increasingly popular resource for measuring community health, psychology, and economics (Coppersmith et al., 2015; Eichstaedt et al., 2015; Weeg et al., 2015; Mowery et al., 2016; Haimson and Hayes, 2017). Coppersmith et al. (2015), for instance, examine trends in language use among Twitter users who self-reported one of ten mental health diagnoses. Eichstaedt et al. (2015) and Weeg et al. (2015) use Twitter to predict the prevalence rates of various health outcomes, such as heart disease mortality and depression, at the county level. Haimson and Hayes (2017) tracked changes in the emotional well-being of transgender communities on Tumblr between 2015 and 2016.

Socio-demographics are often correlated with health outcomes (such as age and heart disease), which is why such variables are often used as controls during analysis (Coppersmith et al., 2015; Dos Reis and Culotta, 2015; Eichstaedt et al., 2015; Weeg et al., 2015). Because of their predictive power, socio-demographics and other extra-linguistic information can additionally be leveraged when building the model itself.

However, a central challenge in integrating community attributes is that they have very different properties than linguistic features and can be lost, in essence, like a needle in a haystack. For example, linguistic features like n-grams are high dimensional, with each dimension having high covariance with other dimensions and likely very little relationship with the outcome. On the other hand community features may be measured more robustly and are relatively low dimensional, often obtained through well-defined measurements. Not surprisingly, Zamani and Schwartz (2017) showed a naive combination that simply concatenates these two sets of features risks losing the effective extra-linguistic features in a sea of weak linguistic features. They go on to show a *residualized control* approach achieves significantly greater accuracy at economic prediction by first learning a model using extra-linguistic features (i.e. controls or community factors) and then train a language model on top of the residual error of the previous model.

It is possible that even when extra-linguistic features are not directly beneficial for prediction, they can still affect people's language. Other related works consider how the meaning of language changes depending on who states it. For instance, when an NLP PhD student says the word 'paper' he/she usually means something different than when a 5^{th} grade student uses the same word (i.e. 'research paper' versus 'piece of paper'). Hu et al. (2017) noted the same words can have different meanings if different people say them. This idea of contextualizing language with extra-linguistic information has been the basis for multiple models: Hovy (2015) learn age- and gender-specific word embeddings, leading to significant improvements for three text classification tasks. Volkova et al. (2013) found that using gender-specific features lead to improvements in sentiment analysis over a gender-agnostic model. Most recently, Lynn et al. (2017) proposed a domain adaptation-inspired method for composing user-level, extra-linguistic information with message-level features, leading to improvements for multiple text classification tasks; we build off of this approach and that of Zamani and Schwartz (2017) in this paper.

While Lynn et al. (2017) injected user-level info into message-level tasks, we are investigating whether same-level adaptation techniques are similarly useful.

We also try to find the circumstances under which each of the adaptation and residualized control approaches are more powerful, and we take on the non-trivial task of exploiting concepts from both the adaptation and the residualized control techniques at the same time, finding that they add even more power when combined with one another.

3 Method

We describe *residualized factor adaptation* (RFA), an approach to text-based prediction utilizing extra-linguistic factors (also called controls – often demographic or socioeconomic information). The key challenge for RFA lies in effectively combining two different types of features. The language-based features, extracted from the tweets, are numerous but are only weak indicators of the outcomes. The socioeconomic and demographic features, on the other hand, are strong indicators but fewer in number. Naively combining both sets of features ignores this crucial difference in their predictive abilities, potentially resulting in important features getting drowned out.

We first describe two methods that effectively combine extra-linguistic factors at two different levels: 1) *Residualized control* is a model-level combination method which builds different models for each type of feature, then combines the results of these models to make the final outcome prediction. 2) *Factor adaptation* is a feature-level combination method that composes the two feature sets with one another to produce a transformed feature space over which a single model may be built. Finally, we present our combined method of *Residualized Factor Adaptation* which takes advantage of both concepts without exploding model parameters.

3.1 Residualized Control Prediction

Language-based features and community-level attributes are qualitatively different modalities. The extra-linguistic variables, while few in number, are mostly unbiased and follow a normal distribution, which can be used to build a strong outcome predictor. However, without special treatment, the signal in extra-linguistic variables can be overwhelmed when combined with a large number of language-based features.

The residualized control approach (Zamani and Schwartz, 2017) avoids this issue by building two models. The first is a prediction model built over the extra-linguistic variables (or *controls*) alone. The error, or *residuals*, produced by this first model represents the information that was unable to be predicted using the extra-linguistic variables alone. The language-based features are therefore brought in to improve upon the initial predictions by using the residuals as training labels for a model based on the linguistic features. In this way, the language-based features are able to account for additional information not captured by the initial extra-linguistic feature-only model. At test time, each instance is fed to both prediction models, and the final outcome is given as a sum of the predictions from both models — the outcome predicted by the extra-linguistic model adjusted for error by the language-based model.

Formally, given extra-linguistic features X_{EL} and language features X_L, the residualized control models are built as follows:

$$\hat{Y} = \alpha \times X_{EL} + \beta \tag{1}$$

$$\epsilon = Y - \hat{Y} \tag{2}$$

$$\epsilon \simeq \gamma \times X_L + \lambda \tag{3}$$

The extra-linguistic control model is parameterized by α weights and the β bias term. ϵ denotes the residual, i.e., the error of the extra-linguistic model. The language-based model aims to predict the ϵ residuals, with γ weights and the λ bias term as parameters.

The motivation for this approach is that extra-linguistic features are more informative and less noisy than the language ones. By exploiting this two-stage learning procedure, the model is biased toward favoring the role of extra-linguistics over language features, which prevents the powerful but rare extra-linguistic features from being lost among thousands of noisy language features.

3.2 Factor Adaptation

Lynn et al. (2017) introduced *user-factor adaptation*, a technique for combining message-level features with user-level information (or *factors*) at the feature level. User-factor adaptation, which is based on the feature augmentation approach for domain adaptation (Daumé III, 2007), uses the extra-linguistic features to transform the language-based features. Each of the language-based features has additional, corresponding features that are a composite of itself and an extra-linguistic factor. In this way, the model is able to capture both factor-specific and factor-general properties of each of the language-based features.

Following the work of Lynn et al. (2017), we use a multiplicative composition function for combining the linguistic and extra-linguistic features.

Instead of using user-level factors, we use extra-linguistic variables obtained at the community level, as described below. More formally, let V_j be a matrix such that:

$$\forall j \in \{0, d\} : V_j = v_j \odot \mathbf{1}^l \tag{4}$$

where d is the number of extra-linguistic factors. With n as the number of data instances, let v_j be a column vector of height n where element $v_{j,i}$ is the score of extra-linguistic feature j for instance i. Having l as the number of language variables, in Eq. 4 for each factor j we make a matrix of size $n \times l$, named V_j, in which every column is equal to v_j, and V_j has the same dimensions as language feature matrix X_L. Now for each factor j we use the Hadamard product to multiply V_j with X_L. In this way each row of X_L will be multiplied by the corresponding row in V_j, which is also equal to the corresponding value in v_j. We therefore can write the factor adaptation as follows:

$$X_A = [V_1 \odot X_L, V_2 \odot X_L, \cdots, V_d \odot X_L] \tag{5}$$

The adapted features together with the original language-based features are used for building a single prediction model:

$$\hat{Y} = \alpha \times [X_L, X_A] + \beta \tag{6}$$

3.3 Residualized Factor Adaptation

Even though both the residualized control and factor adaptation approaches exploit extra-linguistics, they combine these in very different ways. The former does it at the model level by learning different models for different types of features and combining those models together. The latter does it at the data level by first combining both sets of features into a transformed feature set and then learning a single model on the obtained features. In addition, these approaches have different motivations and aim to accomplish different objectives.

These modeling differences suggest that the two approaches could have complementary benefits. Residualized factor adaptation (RFA), our proposed method, inherits the advantages of both the residualized control and adaptation techniques, and is depicted in Fig. 1. There are four main steps:

Step 1: Extra-linguistic control model. We build a regression model solely based on the extra-linguistics, as shown in Equation 1, and then compute the residual error of that model as in Equation 2. This error is ultimately used as the outcome label in the final step of RFA.

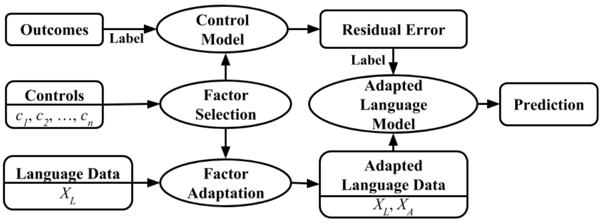

Figure 1: Components of residualized factor adaptation. X_L is language data (topic and n-gram features) and X_A is adapted language data.

Step 2: Factor selection. Adaptation to many factors can increase the model parameters drastically. We explore multiple options for selecting a subset of factors from the available extra-linguistic variables. First, we consider manually selected extra-linguistic factors that are known to influence language use more than others. Second, we use the correlation of the factors with the outcome. Last, we use PCA, an unsupervised method to generate new, lower-dimensional factors from the originals. The purpose of factor selection is to reduce the variance and chance of overfitting.

Step 3: Factor adaptation. We modify the original factor adaptation approach to account for the larger number of factors and features in this task. First, we normalize selected factors by min-max scaling and then multiply the language features by these selected factors as shown in Eq. 5. As we describe later on in Section 4.2, we use n-grams and topics as separate feature sets of language data, so adaptation gives us two corresponding sets of features: adapted n-grams and adapted topics. We then standardize the adapted features using Z-scores and perform feature reduction on each of the four sets of features separately. These reduced features sets are concatenated into a single large set and fed as input to a learning algorithm in the next step.

Step 4: Residual Prediction. The final step, as shown in Eq. 7, is to learn the residual errors of the extra-linguistic control model, using language and adapted language features. Here, we first apply feature selection and reduction on each language feature set: topics, n-grams, adapted-topics and adapted-n-grams. Then we put all of them into a single feature space on which we learn a model to predict the residual error from Step 1.

To produce the final outcome predictions, the predicted error from this model is combined with the predicted outcomes of the extra-linguistic control model from Step 1.

The choice of feature selection is vital for RFA, both due to the fact that it multiplies the number of language features by the number of extra-linguistic features, and because it uses extra-linguistic features at two levels, one separately and one in integration with language features, potentially leading to overfitting. In Section 5.2, we investigate different methods for feature selection to improve RFA's performance.

As Fig. 1 shows, RFA is structured similarly to residualized control. However, residualized control uses language data at its final step, whereas RFA uses both language and adapted language, which is obtained using the factor adaptation technique. This helps RFA to benefit from the advantages of both residualized control and factor adaptation. In other words, RFA combines linguistic and extra-linguistic features on both the feature/data level and the model level. Eq. 7 formulates the RFA method, in which ϵ comes from Eq. 2 and X_A is defined in Eq. 5.

$$\epsilon \simeq \gamma \times [X_L, X_A] + \lambda \tag{7}$$

4 Evaluation Setup

Our task is to predict various community-level outcomes based on publicly available data, including social media and other extra-linguistic data such as socioeconomic and demographic information. We focus on two health-related outcomes: heart disease mortality rate (Eichstaedt et al., 2015) and percent fair/poor health life (Culotta, 2014); one psychology-related outcome: life satisfaction (Schwartz et al., 2013a); and two economy outcomes: Increased real estate price rate and foreclosure rate (Zamani and Schwartz, 2017). Our high-level approach is to train separate regression models for each outcome. For each county, the input is a set of tweets posted by users from that county as well as aggregate values of socioeconomic and demographic variables for the county, including median income, percentage with bachelors degrees and median age. The full list of socioeconomic/demographic variables are given in Section 4.1. The open-source Differential Language Analysis ToolKit was used for the entire analysis pipeline (feature extraction through modeling) (Schwartz et al., 2017)[1].

4.1 Data Set

Our evaluation dataset includes information from three sources: (1) *language data* from Twitter messages, (2) *extra-linguistic data* consisting of 11 socioeconomic and demographic variables, and (3) *outcome data* consisting of 5 county-wise outcomes from 3 categories: Health, Psychology, and Economy.

Our language data can be divided into two groups, (1) for Health and Psychological outcomes and (2) for Economical outcomes. The language data we use for Health- and Psychology-related outcomes was derived from Twitter's 10% random stream collected from July 2009 to February 2015 and includes 1.64 billion tweets (Giorgi et al., 2018)[2]. For Economy outcomes, we used the language data from Zamani and Schwartz (2017). This data was derived from Twitter's 1% random stream collected from 2011 to 2013 and includes 131 million tweets. In both cases, the tweets were mapped to counties based on users' self-reported location strings using the procedure proposed by Schwartz et al. (2013a).

The extra-linguistic data consists of 11 variables used in previous work: 4 *socioeconomic* variables including median income, unemployment rate, percentage of bachelors degrees, and percentage of high school degree, as well as 7 *demographic* variables including median age; percentage: female, black, Hispanic, foreign-born, married; and population density (Census Bureau, 2010). All variables were obtained from the US Census (Census Bureau, 2010), and we henceforth refer to them collectively as *extra-linguistic features*. This dataset is only collected every 10 years, so the 2010 US Census is the most recent dataset for all of the *socioeconomic* and *demographic* variables at the county level.

We consider 5 county-wise measurements as outcomes, 2 health-related (heart disease mortality rate, fair/poor health life), 1 psychological (life satisfaction), and 2 economic (yearly increased real estate price rate, yearly foreclosure rate). Health and psychological data was gathered from the Centers for Disease Control and Prevention (2010b) and contains between 1,630 to 1,749 counties, depending on the outcome. The

---

[1] Available at `https://github.com/dlatk`

[2] Available at `https://github.com/wwbp`

Domain		Lang.	3 Socio-Demographic Factors					All Factors				
			Controls Only	*Added-Controls*	*RC*	*FA*	*RFA*	*Controls Only*	*Added-Controls*	*RC*	*FA*	*RFA*
Health	HD	0.585	0.423	0.590	0.620	0.628	0.638	0.515	0.597	0.630	0.636	**0.657** *
	FP	0.602	0.434	0.606	0.619	0.647	0.647	0.609	0.632	0.657	0.685	0.680
Psych.	LS	0.214	0.148	0.219	0.292	0.308	0.338	0.326	0.352	0.376	0.353	0.396 *
Econ.	IP	0.245	0.072	0.243	0.266	0.274	0.307	0.240	0.226	0.330	0.344	**0.402** *
	FC	0.153	0.128	0.156	0.197	0.218	0.238	0.160	0.161	0.209	0.240	**0.276** *
	Avg.	0.360	0.241	0.362	0.398	0.415	0.434	0.370	0.394	0.440	0.452	**0.482** *

Table 1: R^2 (variance explained) of residualized factor adaptation (RFA) versus baseline models. Results are shown for 3 hand-picked factors (age, race, education) as well as all factors. RC is residualized control and FA is factor adaptation. Each row is color-coded separately, from red (lowest value) to green (highest values). Bold and * indicate a significant ($p < .05$) reduction in error over the next best model (bold) and FA (*), respectively, according to paired t-tests.

		Lang.	*Controls Only*	*Added-Controls*	*RC*	*FA*	*RFA*
Health	HD	0.765	0.718	0.773	0.794	0.798	**0.811** *
	FP	0.776	0.781	0.795	0.811	0.828	0.825
Psych.	LS	0.463	0.571	0.594	0.614	0.595	0.630 *
Econ.	IP	0.496	0.490	0.476	0.575	0.587	**0.634** *
	FC	0.391	0.401	0.401	0.457	0.490	**0.526** *
	Avg.	0.578	0.592	0.608	0.650	0.659	**0.685** *

Table 2: Pearson-r of residualized factor adaptation (RFA) versus baseline models (for comparison to other work which uses Pearson-r as the accuracy metric). Results are only shown for all factors. RC is residualized control and FA is factor adaptation. Each row is color-coded separately, from red (lowest value) to green (highest values). Bold and * indicate a significant ($p < .05$) reduction in error over the next best model (bold) and over FA (*), respectively, according to paired t-tests.

economic outcomes, which have been used previously in (Zamani and Schwartz, 2017), were gathered for the year 2013 from Zillow[3]. They contain 427 counties' foreclosure rate and 717 counties' increased real estate price rate.

4.2 Baselines

Our baselines consist of a controls-only prediction model and a language-only prediction model.

Controls-only. The controls-only model is a simple regression model trained over all the 11 extra-linguistic features.

Language-only. Building this baseline consists of three main steps: extracting linguistic features, performing feature reduction, and running ridge-regression (Goeman et al., 2016). Our linguistic features are n-gram features (1-3 grams) and topic features which include mentions of 2,000 LDA (Blei et al., 2003) derived topics previously estimated from social media (Schwartz et al., 2013b).

For language data, we first pruned the sparse n-gram features to only include those that were mentioned in at lease a percentage of the counties, then due to the importance of word count in performance of language predictive models(Zamani et al., 2018) we exploit a word count threshold and drop counties with fewer words. Then we run a correlation threshold to only keep the highest correlated features and finally we perform a randomized principal components analysis (RPCA), an approximate PCA based on stochastic re-sampling (Rokhlin et al., 2009). We apply the correlation threshold and RPCA steps for n-grams and topics independently.

For language data associated with health and psychology outcomes, we pruned the sparse n-gram features to only include those that were mentioned in at least 95% of the counties, and used 20,000 as the word count threshold, resulting in 27,250 n-grams total.

With only 1,749 training instances (one per county), feature selection and dimensionality reduction become necessary for avoiding overfitting. We first limit the features to the top 10,000 n-grams with the highest linear relationships to each outcome. As the topic features are more informa-

---

[3] http://www.zillow.com/research/data/

tive than a single n-gram, we choose to retain all 2,000 topics at this step. Then after performing RPCA we only keep 100 features for each group of ngrams and topcs.

For the language data associated with economy outcomes, we pruned the n-gram features to only include those that were mentioned in at least 10% of the counties, and used 10,000 as the word-count threshold, resulting in 8,897 n-grams across 717 training instances. We use the top 8,000 n-grams and the top 1,500 topic features with the highest linear relationships to each outcome. at the end by applying RPCA we limit the dimension of each feature set to 100.

We compare performances of residualized control, factor adaptation and residualized factor adaptation (RFA). For all these models, we use the same settings as above to generate language features.

5 Results

5.1 Comparison of RC, FA, and RFA

We first compare factor adaptation (FA), residualized control (RC), and residualized factor adaptation (RFA) using three manually selected factors: age, race (percentage of black population), and education (percentage with bachelor's degree) rates. These three factors are often used as "controls" in prior work (Schwartz et al., 2013a; Culotta, 2014; Eichstaedt et al., 2015; Curtis et al., 2018)[4] and also represent examples of demographic and socioeconomic measurements.

In order to ensure a fair comparison, we use the same extra-linguistic features for all models. As mentioned earlier, a naive method is to directly combine the extra-linguistic features with language ones in a single feature set. Here, we also compare this simple model, which we call *added-controls*, with the other three models. In addition, we consider a linear model solely using extra-linguistics, which we call *controls only*. Evaluation is done using 10-fold cross-validation. R^2, or variance explained, is used to measure accuracy.

Table 1 compares results in terms of variance explained, when using the three hand-picked factors vs. using all 11 extra-linguistic factors (Since past work has also used the Pearson-r metric, Table 2 shows the same results for all factors in terms of Pearson-r). As the table shows, FA outperforms controls only, added-controls, and residualized control. RFA does even better and outperforms FA on both the hand-picked factors and when using the entire set of factors. These results demonstrate the complementary nature of the residualized control and factor adaptation approaches and the benefits of combining them.

	No FS	Separated FS	Combined FS	Early FS
HD	0.656	**0.657**	0.65	0.639
FP	0.678	**0.68**	0.676	0.661
LS	0.364	0.396	0.391	**0.401**
IP	**0.425**	0.402	0.392	0.336
FC	0.187	**0.276**	0.268	0.241
AVG	0.462	**0.482**	0.475	0.456

Table 3: Comparing R^2 using different methods of feature selection. Outcomes are heart disease (HD), fair/poor health (FP), life satisfaction (LS), increased price (IP), and foreclosure rate (FC). FS stands for feature selection. Bold cells have the highest R^2 for each outcome.

Even though adding controls directly, as in the "added-controls" column, works better than language-only and controls-only models, it is worse than any other model that exploits both language and extra-linguistic data. This motivates the need for combining different types of features in both an additive (residualized control) and multiplicative (factor adaptation) style.

Overall, these results show the power of RFA over the other models. RFA's improvement over FA was statistically significant for 4 out of 5 outcomes, and 3 out of 5 for residualized control. Recall that added-controls, residualized control, FA, and RFA all have access to the same set of information. The gains of RFA over FA show that RFA's structure utilizing residualized control is better suited for combining extra-linguistic and language-only features.

5.2 Feature Selection

Here we investigate the impact of feature selection on the overall performance of RFA. We consider three different combination of adaptation and feature selection, as well as adaptation without any feature selection: (1) *SeparatedFS:* apply feature selection separately on language features and adapted language features; (2) *CombinedFS:* combine language features and adapted language features into one feature set and then apply feature selection; (3) *EarlyFS:* apply feature selection on

---

[4] Income has also been used frequently but it has been shown to correlate strongly with education rates.

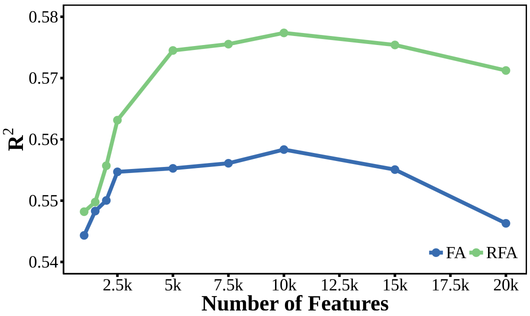

Figure 2: Effect of increasing the number of selected features in univariate feature selection on both factor adaptation (FA) and residualized factor adaptation (RFA) by looking at the average R^2 among health and psychological outcomes. All 11 factors are used in all cases.

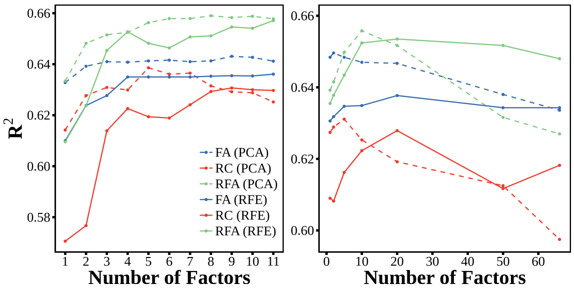

Figure 3: Effect of increasing number of factors on R^2 of residualized factor adaptation (RFA), factor adaptation (FA) and residualized controls (RC) for heart disease outcome. Factors are obtained through Recursive Feature Elimination (RFE) or PCA. Left plot is with original factors, and right plot is with interaction factors (the product from pairing factors).

language features, then apply adaptation on the selected features; and (4) *NoFS:* perform adaptation without any feature selection. Table 3 shows the performance of each method on all 5 outcomes, as well as the the average. SeparatedFS performs better than the others in 3 out of 5 cases, as well as leading the average R^2 across all 5 outcomes. In addition, it produces the most stable results in comparison to the other methods. We therefore use this method for RFA.

We perform another experiment to find the best parameter for the univariate feature selection, that is, the value of k when selecting the k-best n-gram features. Figure 2 shows the results of varying the number of features used for FA and RFA. We report the average R^2 across the 3 health- and psychology-related outcomes. In general, selecting more features leads to better results, though eventually performance does begin to suffer. Recall that our feature selection approach is to first select the k-best n-gram features based on their linear relationship with the outcome, then do a PCA on these k features to obtain a reduced-dimension vector. Even though the feature selection doesn't directly increase the size of our models, it effectively increases the amount of information available to the models, leading to the positive trends we see in Figure 2.

5.3 Increasing Factors and Factor Selection

This experiment has two objectives: first to find out how the number of factors affects performance, and second to find an automated way to select a good subset from the extra-linguistic factors. Here we vary the number of factors from 1 to 11 (i.e. all factors) and compare the effects on RFA, FA, and RC. Factor selection in this experiment is done in two ways, supervised and unsupervised. For the supervised selection we use Recursive Feature Elimination, in which for each k, the least significant factors are recursively dropped until only k factors remain. For unsupervised selection, we use PCA to build k new factors with the highest variance.

The left of Figure 3 shows how the performance of RFA, FA and RC change for heart disease outcome as the number of factors increase, using both PCA and RFE as factor selection methods.

RFA outperforms FA at every factor number, and begins to outperform RC as the number of factors increases. RFA's performance, in general, tends to increase as we add more factors. Using PCA, RFA reaches close to its best performance very quickly, requiring only 5 or 6 factors; adding more factors results in longer runtimes for minimal gain. However, in the case of RFE, using more factors appears to be worthwhile. FA and RC both quickly plateau, or even decline, as more features are added.

Since performance generally improved as more factors were added, we explored adding more factors beyond the 11 that are available to us. To this end we create new factors by multiplying the existing factors with one another. To account for variance in the factor ranges, we first min-max normalize each factor. Then we consider every pair of factors and multiply their normalized values together and re-normalize these new values to create a new factor. This gives a total of 55 new factors in addition to the original 11. We rerun our experiments with this new pool of 66 factors.

The right of Figure 3 shows the results of using this expanded pool of factors. Here, the performance begins to taper off beyond 15 factors for both FA and RFA. Overall, PCA obtains its best performance with only a few factors, but then begins to suffer as more factors are added. RFE, on the other hand, tends to perform worse than PCA initially but remains relatively stable as more factors are added. These newly-created features turned out to be less effective than the original eleven, suggesting that the trade-off in increasing factors via combination is not worthwhile.

Overall, even though reducing the number of factors through PCA-based factor selection could not beat the best accuracy, it is still very competitive. Given the potentially huge number of features obtained through factor adaptation, this slight decrease in performance may be worth potential increases in runtime. RFE-based factor selection, however, helps with neither the runtime nor the performance.

6 Conclusions

Language-based prediction tasks involving communities can benefit from both socio-demographic factors and linguistic features. Because this information comes from different sources and has different distributions, effective mechanisms are needed for combining them. In this paper, we present *residualized factor adaptation*, a method that unites two ways of approaching this problem, one where strong community attributes are augmented (i.e. additive use of factors) with weak but noisy language features, and the other where the contextual differences in language use are mediated via community attributes (i.e. adaptation to community factors). The proposed method effectively combines the complementary benefits of both residualized control and factor adaptation approaches to yield substantial gains over differing community-level prediction tasks across three domains. We see this work as part of a growing need for application-oriented approaches that not only leverage large data effectively by themselves, but do so in the context of other social scientific information that is already available and valuable.

Acknowledgments

This work was supported, in part, by a grant from the Templeton Religion Trust (ID TRT0048). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

Centers for disease control and prevention. (2010). underlying cause of death 1999-2010. cdc wonder online database [data set]. retrieved from http://wonder.cdc.gov/ucd-icd10.html.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

United States Census Bureau. 2010. Profile of general population and housing characteristics: 2010 demographic profile data. https://factfinder.census.gov/faces/tableservices/jsf/pages/productview.xhtml?pid=DEC_10_DP_DPDP1&prodType=table.

Deborah A. Cohen, Thomas A. Farley, and Karen Mason. 2003. Why is poverty unhealthy? social and physical mediators. *Social Science & Medicine*, 57(9):1631–1641.

Glen Coppersmith, Mark Dredze, Craig Harman, and Kristy Hollingshead. 2015. From ADHD to SAD: Analyzing the language of mental health on Twitter through self-reported diagnoses. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology*, pages 1–10.

Aron Culotta. 2014. Estimating county health statistics with twitter. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 1335–1344. ACM.

Brenda Curtis, Salvatore Giorgi, Anneke E. K. Buffone, Lyle H. Ungar, Robert D. Ashford, Jessie Hemmons, Dan Summers, Casey Hamilton, and H. Andrew Schwartz. 2018. Can twitter be used to predict county excessive alcohol consumption rates? *PLOS ONE*, 13(4):1–16.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*.

Virgile Landeiro Dos Reis and Aron Culotta. 2015. Using matched samples to estimate the effects of exercise on mental health from Twitter. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 182–188.

Johannes C. Eichstaedt, H. Andrew Schwartz, Margaret L. Kern, Gregory Park, Darwin R. Labarthe, Raina M. Merchant, Sneha Jha, Megha Agrawal, Lukasz A. Dziurzynski, Maarten Sap, et al. 2015. Psychological language on twitter predicts county-level heart disease mortality. *Psychological Science*, 26(2):159–169.

Salvatore Giorgi, Daniel Preotiuc-Pietro, Anneke Buffone, Daniel Rieman, Lyle Ungar, and H. Andrew

Schwartz. 2018. The remarkable benefit of user-level aggregation for lexical-based population-level predictions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Jelle Goeman, Rosa Meijer, and Nimisha Chaturvedi. 2016. L1 and l2 penalized regression models.

Oliver L Haimson and Gillian R Hayes. 2017. Changes in social media affect, disclosure, and sociality for a sample of transgender americans in 2016's political climate. In *ICWSM*, pages 72–81.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of ACL*.

Tianran Hu, Ruihua Song, Maya Abtahian, Philip Ding, Xing Xie, and Jiebo Luo. 2017. A world of difference: Divergent word interpretations among people. In *ICWSM*.

Veronica E. Lynn, Youngseo Son, Vivek Kulkarni, Niranjan Balasubramanian, and H. Andrew Schwartz. 2017. Human centered NLP with user-factor adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1157–1166.

Danielle Mowery, Albert Park, Mike Conway, and Craig Bryan. 2016. Towards automatically classifying depressive symptoms from Twitter data for population health. *Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*, page 182.

Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. 2009. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Richard E Lucas, Megha Agrawal, Gregory J Park, Shrinidhi K Lakshmikanth, Sneha Jha, Martin EP Seligman, et al. 2013a. Characterizing geographic variation in well-being using tweets. In *ICWSM*.

H. Andrew Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Lukasz Dziurzynski, Stephanie M. Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin E. P. Seligman, et al. 2013b. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9):e73791.

H. Andrew Schwartz, Salvatore Giorgi, Maarten Sap, Patrick Crutchley, Johannes Eichstaedt, and Lyle Ungar. 2017. Dlatk: Differential language analysis toolkit. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of EMNLP*.

Christopher Weeg, H Andrew Schwartz, Shawndra Hill, Raina M Merchant, Catalina Arango, and Lyle Ungar. 2015. Using Twitter to measure public discussion of diseases: A case study. *JMIR Public Health and Surveillance*, 1(1).

Mohammadzaman Zamani, Anneke Buffone, and H Andrew Schwartz. 2018. Predicting human trustfulness from facebook language. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 174–181.

Mohammadzaman Zamani and H Andrew Schwartz. 2017. Using twitter language to predict the real estate market. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 28–33.

Framing and Agenda-setting in Russian News: a Computational Analysis of Intricate Political Strategies

Anjalie Field♠ **Doron Kliger**♦ **Shuly Wintner**♦
Jennifer Pan♥ **Dan Jurafsky**♥ **Yulia Tsvetkov**♠
♠Carnegie Mellon University ♦University of Haifa ♥Stanford University
`{anjalief, ytsvetko}@cs.cmu.edu,`
`{kliger@econ, shuly@cs}.haifa.ac.il, {jp1, jurafsky}@stanford.edu`

Abstract

Amidst growing concern over media manipulation, NLP attention has focused on overt strategies like censorship and "fake news". Here, we draw on two concepts from the political science literature to explore subtler strategies for government media manipulation: agenda-setting (selecting what topics to cover) and framing (deciding how topics are covered). We analyze 13 years (100K articles) of the Russian newspaper *Izvestia* and identify a strategy of distraction: articles mention the U.S. more frequently in the month directly following an economic downturn in Russia. We introduce embedding-based methods for cross-lingually projecting English frames to Russian, and discover that these articles emphasize U.S. moral failings and threats to the U.S. Our work offers new ways to identify subtle media manipulation strategies at the intersection of agenda-setting and framing.

1 Introduction

Authoritarian countries such as Russia and China have received a great deal of attention for trying to control and distort the spread of information through "fake news" and censorship. However, authoritarian governments might also use subtle tactics of media manipulation that are much harder to detect, like flooding communication channels with irrelevant information or highlighting particular viewpoints of an event to distract public attention (Rozenas and Stukal, Forthcoming; Munger et al., 2018; King et al., 2017). "Fake news" can be identified by fact checkers. Censorship can be detected by checking what content is no longer available. However, we have no systematic way of identifying more subtle forms of media manipulation. To date, research has been limited to occasional leaks to reveal ground truth data (King et al., 2017). This paper proposes techniques—grounded in economics and political science—to automatically identify subtle manipulation at scale, and applies these techniques to study Russian media.

These subtle manipulation strategies can be understood through *agenda-setting*—selecting *what* topics to cover—and *framing*—*how* aspects of those topics are highlighted to promote particular interpretations (Entman, 2007; Ghanem and McCombs, 2001). For example, abortion can be framed in terms of the life of a child or a woman's freedom of choice (Tankard Jr, 2001). Agenda-setting and framing can have a significant influence on public opinion by attending to particular issues at the exclusion of others (McCombs, 2002; Boydstun et al., 2013). Both concepts have been well-studied in English-speaking democratic countries, but understudied in other settings. Here, we apply these concepts to the study of media manipulation in Russia, particularly as strategies of an autocratic regime.

We focus on Russia, because of intense interest in the way Russia is shaping the global information environment (Van Herpen, 2015). Many Russian media outlets are state-owned or heavily influenced by the government. We focus on news coverage from 2003–2016 in one of the most widely-read newspapers in Russia: *Izvestia*. Despite a brief period of autonomy, *Izvestia* has become strongly influenced by the government (Jones, 2002).

Prior work has identified a relationship between negative economic performance in Russia, such as stock market declines, and "selection attribution" in state-controlled media outlets, where negative events are blamed on foreign officials while positive events are credited to domestic officials (Rozenas and Stukal, Forthcoming). We build on these findings and investigate the relationship between economic performance, including that of the Russia Trading System Index (RTSI) and gross

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3570–3580
Brussels, Belgium, October 31 - November 4, 2018.

domestic product (GDP), and news coverage of foreign events. We primarily investigate coverage of the United States because Russia has seen the U.S. as its main rival since the Cold War, and we expect news coverage of foreign events to focus disproportionally on the U.S.

We first establish a strong negative correlation between Russia's economic situation and the proportion of news focused on the U.S. (§2). We then show that the correlation is directed: economic indicators precede (and thereby Granger-cause) the increase in foreign news coverage (§2.2). We consider this a clear case of agenda-setting. We then investigate *how* these news articles *frame* the U.S. We develop a distant supervision method to project English framing annotations onto our Russian corpus (§3), and draw on the projected frames to analyze manipulation strategies in news about the U.S. (§5).

The contributions of this work are manifold: we show how framing and agenda-setting (concepts traditionally applied to policy debates) can be used to understand media manipulation strategies; we use economic metrics to automate the identification of agenda-setting; we devise a novel method for cross-lingual projection of framing annotations; and we use these annotations to show how agenda-setting is realized.

2 Agenda-Setting

All media outlets inevitably use a form of *agenda-setting*: deciding what is "newsworthy" by covering some topics at the exclusion of others. Agenda-setting can powerfully sway the focus of public opinion (McCombs, 2002). We hypothesize that in countries with weak democratic institutions and in particular, with state-controlled media, the government may actively use agenda-setting to shape public opinion. We observe this phenomenon by comparing how much Russian newspapers describe the U.S. and the state of the Russian economy. We then use Granger-causality to show that a decline the Russian stock market is followed by an increase in U.S. news coverage.

Our results are based on a corpus of over 100,000 articles from the newspaper *Izvestia* published in 2003–2016 (see Appendix A for details).

2.1 Correlations

We compared the salience of news focused on the U.S. with indicators that reflect the economic state of Russia to test our hypothesis: that news coverage of the U.S. is used to distract the public from negative economic events. We first performed an initial, simplistic study of this agenda-setting strategy. We define *U.S. coverage* as the ratio of *Izvestia* articles that mention the U.S. at least twice to the total number of articles in any given time slice (in our initial study, a year). We show in Figure 1 the U.S. coverage plotted against Russian GDP, in an annual resolution. We find a strong negative Pearson's correlation (r=-0.83): mentions of the U.S. in *Izvestia* increase as economic indicators deteriorate. The one exception to this trend is 2008, during which there was a high amount of U.S. news coverage and the Russian GDP peaked. This year coincides with both the U.S. financial crisis and the Obama-McCain Presidential election, which would explain a focus on U.S. events regardless of the Russian economic situation.

U.S. coverage, measured by counting mentions of the U.S. in *Izvestia*, is inversely related to the level of the Russian GDP. This negative correlation indicates the possibility of intentional agenda-setting by the Russian government.

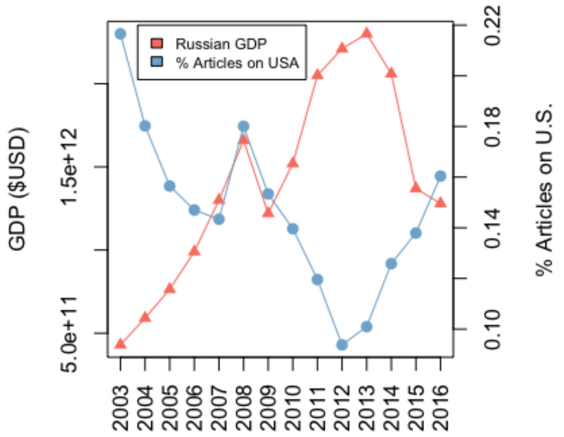

Figure 1: Proportion of articles that mention the U.S. at least twice (blue) and Russian GDP (red), 2003–2016.

We now extend these preliminary results in several ways. First, we refine the definition of *U.S. coverage* by using two metrics: **article level,** the number of articles that mention the U.S. at least twice normalized by the total number of articles in the time slice; and **word level,** the frequency of the occurrences of the U.S., normalized by the total number of words in the time slice. Second, we compare these metrics to *two* economic indicators: GDP (in USD) and the index of the Russian stock market (RTSI), in rubles.[1] Third, we refine the time-resolution and use yearly, quarterly, and monthly time slices.

Table 1 reports the correlations between the two metrics of U.S. coverage and (monthly, quarterly, and yearly) RTSI and GDP values. At all levels, there are strong negative correlations between the proportion of news focused on the U.S. and economic state.

Level:	**Article**	**Word**
RTSI (Monthly, rubles)	-0.54	-0.52
GDP (Quarterly, USD)	-0.69	-0.65
GDP (Yearly, USD)	-0.83	-0.79

Table 1: Pearson's correlation between news coverage of the U.S. and economic indicators.

2.2 Granger Causality

Next, we hypothesize that these correlations are in fact directed: a change in the economy is followed by a change in U.S. news coverage. To investigate this hypothesis we employ Granger causality (Granger, 1988). The key concept behind Granger causality is that cause precedes effect. Thus, a time series X is said to Granger-cause a times series Y if past values x_{t-i} are significant indicators in predicting y_t. First, we computed the article-level (a_t) and word-level (w_t) metrics at a monthly granularity from 2003 to 2016; we also extracted the RTSI monthly close price (in rubles) for the same time period (r_t). We then calculated the percentage change of these series as: $C(w_t) = \frac{w_t}{w_{t-1}} - 1$, and equivalently calculated $C(a_t)$ and $C(r_t)$. By taking the percent change of both series, we control for long term trends (e.g., stock markets tend to trend upwards over time), and instead focus on short-term relations: does a change in the economy directly precede a change in news coverage?

We computed Granger causality between $C(w_t)$ and $C(r_t)$ by fitting a linear regression model:

$$C(w_t) = \sum_{i=1}^{m} \alpha_i(C(w_{t-i})) + \sum_{j=1}^{n} \beta_j(C(r_{t-j}))$$

where m and n denote how far back in time we look (denoted as m-lag or n-lag). We can say that r_t Granger-causes w_t if we find that β is significantly different from zero.

Tables 2 and 3 report the results. A p-value ≤ 0.05 indicates significance; thus we find 1-lag RTSI values Granger-cause coverage of U.S. news by both the word-level and article-level metrics. Importantly, the r_{t-1} coefficient is negative, which indicates that a decline in the stock market is followed by an increase in U.S. news coverage. In the 2-lag analysis, the r_{t-2} values are not significant, which suggests that the changes in news coverage follow changes in the stock market within one month.[2] For completeness, we also computed Granger causality in the reverse direction: i.e., does a change in U.S. news coverage Granger-cause a change in the stock market? As expected, we found no significant results.

	1-Lag		**2-Lag**	
	$\alpha; \beta$	**p-Value**	$\alpha; \beta$	**p-Value**
w_{t-1}	-0.233	0.003	-0.320	0.00005
w_{t-2}	-	-	-0.301	0.0001
r_{t-1}	-0.352	0.0334	-0.369	0.024
r_{t-2}	-	-	-0.122	0.458

Table 2: Granger causality between % change in RTSI and frequency of USA (word level).

	1-Lag		**2-Lag**	
	$\alpha; \beta$	**p-Value**	$\alpha; \beta$	**p-Value**
a_{t-1}	-0.222	0.005	-0.290	0.000289
a_{t-2}	-	-	-0.270	0.000634
r_{t-1}	-0.311	0.035	-0.329	0.0267
r_{t-2}	-	-	-0.091	0.543

Table 3: Granger causality between % change in RTSI and frequency of USA (article level).

3 Framing Analysis

We hypothesize that *framing* can further our understanding of why Russian media focuses on the U.S. during economic downturns. By identifying common frames in news coverage of the U.S., we see how the concepts of agenda-setting and framing work together to manipulate public attention. In this section, we first define the concept of framing and demonstrate why existing methods are insufficient for analysis of the *Izvestia* corpus. We then present a new method for analyzing frames and evaluate it quantitatively through hand-annotations and qualitatively through a series

---

[1] Stock market values were obtained from the Moscow exchange website. GDP values were obtained from OECD.

[2] We computed Granger causality at a quarterly and yearly level and found no significant causal relationship. This result is unsurprising; the monthly analysis suggests trends in news coverage are largely driven by the previous month, so we would not expect causality at a quarterly or yearly level.

of examples. Finally, we use this method to contextualize strategies of media manipulation in the *Izvestia* corpus.

3.1 Background on Framing Analyses

While agenda-setting broadly refers to what topics a text covers, framing refers to which *attributes* of those topics are highlighted. Several aspects of framing make the concept difficult to analyze. First, just defining framing has been "notoriously slippery" (Boydstun et al., 2013). Frames can occur as stock phrases, i.e. "death tax" vs. "estate tax", but they can also occur as broader associations or sub-topics (Tsur et al., 2015; McCombs, 2002). Frames also need to be distinguished from similar concepts, like sentiment and stance. For example, the same frame can be used to take different stances on an issue: one politician might argue that immigrants boost the economy by starting new companies that create jobs, while another might argue that immigrants hurt the economy by taking jobs away from U.S. citizens (Baumer et al., 2015; Gamson and Modigliani, 1989). Finally, unlike classification tasks where each article is assigned to a single category, most articles employ a variety of frames (Ghanem and McCombs, 2001).

Recent work has attempted to address these conceptual challenges by defining broad framing categories. The Policy Frames Codebook defines a set of 15 frames (one of which is "Other") commonly used in media for a broad range of issues (Boydstun et al., 2013). In a follow-up work, the authors use these frames to build The Media Frames Corpus (MFC), which consists of articles related to 3 issues: immigration, tobacco, and same-sex marriage (Card et al., 2015). About 11,900 articles are hand-annotated with frames: annotators highlight spans of text related to each frame in the codebook and assign a single "primary frame" to each document. However, the MFC, like other prior framing analyses, relies heavily on labor-intensive manual annotations.

The primary automated methods have relied on probabilistic topic models (Tsur et al., 2015; Boydstun et al., 2013; Nguyen et al., 2013; Roberts et al., 2013). Although topic models can show researchers what themes are salient in a corpus, they have two main drawbacks: they tend to be corpus-specific and hard to interpret. Topics discovered in one corpus are likely not relevant to a different corpus, and it is difficult to compare the outputs of topic models run on different corpora. Other automated framing analyses have used the annotations of the Media Frame Corpus to predict the primary frame of articles (Card et al., 2016; Ji and Smith, 2017), or used classifiers to identify language specifically related to framing (Baumer et al., 2015). Importantly, all of these methods focus exclusively on English data sets. While unsupervised methods like topic models can be applied to other languages, any supervised method requires annotated data, which does not exist in other languages.

3.2 Framing Analysis Methodology

Our goal is to develop a method that is easy to interpret and applicable across-languages. In order to ensure our analysis is interpretable, we ground our method using the annotations of the Media Frames Corpus. However, because the MFC is entirely in English and our test corpora is in Russian, we cannot use a fully supervised method. Instead, we use the MFC annotations to derive lexicons for each frame, which we then translate into Russian. We use query-expansion to reduce the noisiness of machine translation and make the lexicons specific to the *Izvestia* corpus, rather than specific to the MFC. We evaluate the derived lexicons in English and in Russian. Finally, we use these lexicons to analyze frames in *Izvestia* and identify strategies of media manipulation. Our method allows for in-depth analysis by identifying primary and secondary frames in a document and specific words that signify frames.

Generating framing lexicons Although our primary test corpus is in Russian, we also use English test corpora for evaluation; thus, we describe our method as applicable to either language. First, we use the MFC annotations to derive a lexicon of English words related to each frame in the Policy Frames Codebook. For a given frame F we measure pointwise mutual information (Church and Hanks, 1990) for each word in the corpus as:

$$I(F, w) = log\frac{P(F, w)}{P(F)P(w)} = log\frac{P(w \mid F)}{P(w)}$$

We estimate $P(w|F)$ by taking all text segments annotated with frame F, and computing $\frac{Count(w)}{Count(allwords)}$. We similarly compute $P(w)$ from the entire corpus. We then use the 250 words with the highest $I(w, F)$ as the base framing lexicon for frame F, denoted F_{base}. We discard all words that occur in fewer than 0.5% of documents or in more than 98% of documents.

Translation and extension of framing lexicons Next, we use query-expansion to alter F_{base}, with the goal of generalizing the lexicon. Without this step, our lexicons are biased towards words common in English news articles, particularly words specific to the 3 policy issues in the MFC.

When our test corpus is in a different language (i.e. Russian), we use Google Translate to translate F_{base} into the new language. We restrict our vocabulary to the 50,000 most frequent words in the test corpus.

Then, to perform the query-expansion, we train 200-dimensional word embeddings on a large background corpus in the test language, using CBOW with a 5-word context window (Mikolov et al., 2013). We compute the center of each lexicon, c, by summing the embeddings for all words in the lexicon. We then identify up to the K nearest neighbors to this center, determined by the cosine distance from c, as long as the cosine distance is not greater than a manually-chosen threshold (t).[3] We again filtered the final set by removing all words that occur in fewer than 0.5% of documents or in more than 98% of documents.

The final lexicons contain between 100 and 300 words per frame. Table 4 depicts a few examples of lexicon words extracted from the MFC, and words in our final lexicons. We can observe that words in F_{rus} are closely related to words in F_{base}, but also specific to Russian culture and politics.

We consider a document to employ a frame F if the document contains at least 3 instances of a word from F's lexicon. We assign the primary frame of a document to be its most common frame, determined by the number of words from each framing lexicon in the document .[4]

F_{base}	F_{rus}
Political	
republican-controlled	bills
filibuster	conservative
gubernatorial	parlimentary
Economic	
cents	deductions
holdings	tax
profitable	fines
Public Sentiment	
gallup	activism
demonstrators	facsim
rallied	vote

Table 4: Example lexicons extracted from the MFC and transfered to the *Izvestia* corpus.

4 Evaluation of Framing Lexicons

We can evaluate the English lexicons using annotated data from the MFC. For the Russian lexicons, since we do not have annotated Russian data, we instead conduct an annotation task. These evaluation metrics determine how well our method captures which frames are present in a text. Finally, we also qualitatively compare our method to existing methods for framing analysis, specifically topic models.

English Evaluations We first evaluate our lexicons on two tasks using the MFC annotations: primary frame identification and identification of all frames in a document.

Primary frame identification is a 15-class classification problem. Two prior studies evaluate models on this task: Card et al. (2016) and Ji and Smith (2017). Following these studies, we evaluate our model using 10-fold cross-validation on only the "Immigration" subset of the MFC. We use 9 folds to generate framing lexicons and the 10th fold to evaluate. To train word embeddings, we use the entire MFC corpus combined with over 1 million New York Times articles from 1986 - 2016 (Fast and Horvitz, 2017). Table 5 shows the accuracy of our model. Our results outperform Card et al. (2016) and are comparable to Ji and Smith (2017). Furthermore, unlike prior methods, our method is able to transfer to different domains and languages without needing further annotated data.

Ji and Smith (2017)	58.4
Card et al. (2016)	56.8
Our model	57.3

Table 5: Accuracy of primary frame classification.

However, our main interest is in measuring the salience of frames in general, not merely focusing on the primary frame. Thus, we also use our

---

[3] When the test corpus is in English, we set t to 0.4 and K to 500 and we add the identified neighbors to F_{base}. When our test corpus is in Russian, we choose to discard our base lexicon, to prevent the final lexicons from being too U.S.-specific. Instead, we set t to 0.3 and K to 1000, which increases the number of neighbors identified, and we keep only these neighbors in the final lexicon.

[4] We do not generate a lexicon for the "Other" frame, and instead assign a document's primary frame as "Other" only if it does not contain at least 3 words from any framing lexicon. Throughout this process, we use small subsets of the "tobacco" articles for parameter tuning.

lexicons to identify the presence of any frames in a document. As the MFC has multiple annotators, we define a frame to be present in a document if any annotator identified the frame, and use this as gold standard test data. In evaluating our lexicons, we consider a frame to be present in a document if the document contains at least 3 tokens from the frame's lexicon.

To the best of our knowledge, identifying all frames in a document is a new task that was not attempted in prior work. Thus, we use a logistic regression model with bag-of-word features as a standard baseline. As above, we evaluate using 10-fold cross validation on the "Immigration" subset of the MFC. Table 6 shows that our method outperforms the baseline, with the exception of 2 frames, even though the baseline is fully supervised, whereas our method is distantly supervised. We note that the poorest performing frames, "External Regulation and Reputation" and "Morality" are the frames which are least common in this subset of the data – each frame occurs in fewer than 500 articles. When we run the same 10-fold cross validation evaluation on the "Samesex" subsection of the MFC, where the "Morality" frame occurs in over 1000 articles, we achieve a higher F1 score (0.65).

	Ours	**Baseline**
Capacity & Resources	**0.53**	0.48
Crime & Punishment	**0.78**	0.76
Cultural Identity	0.57	**0.62**
Economic	**0.69**	0.67
External Regulation	0.25	**0.47**
Fairness & Equality	**0.50**	0.44
Health & Safety	**0.58**	0.53
Legality & Constitutionality	**0.80**	0.76
Morality	**0.31**	0.25
Policy Prescription	**0.72**	0.69
Political	**0.80**	0.77
Public Sentiment	**0.54**	0.47
Quality of Life	**0.65**	0.63
Security & Defense	0.63	0.63

Table 6: F1 Scores for identification of all frames in a document.

Russian Evaluations Next, we evaluate the quality of our method on the Russian data set. Unlike in English, we do not have frame-annotated data in Russian. We instead performed the intruder detection task, an established method for evaluating topic models (Chang et al., 2009). For each frame F we randomly sampled 5 words from the framing lexicon F_{rus} and 1 word from the lexicon of a different frame, which has no overlap with F_{rus}. We then presented two (native Russian speaking) annotators with the frame heading and the set of 6 words, and asked them to choose which word did not belong in the set. We evaluated 15 sets or 75 words per frame.

Framing can be subjective, and we do not necessarily expect annotators to interpret frames in the same way. We calculate two forms of accuracy: "soft", whether *any* annotator correctly identified the intruder; and "hard", whether both did. We also report average precision as defined in (Chang et al., 2009), i.e. the average number of annotators that correctly identified the intruder, averaged across all sets.

We briefly summarize results here and report them fully in Appendix B. All accuracies are significantly better than random guessing, and no soft accuracy falls below 60%. Only two frames have an average accuracy $\leq 60\%$, "Fairness and Equality" and "Morality", both very abstract concepts. In these frames, we also see a larger difference between hard and soft accuracies, which reflects the subjectivity of framing. The MFC annotators sometimes disagreed on the correct annotations, even after discussing their disagreements (Boydstun et al., 2013). Thus, we can attribute some of the differences between hard and soft accuracies to this subjectivity.

Qualitative Comparison to Structured Topic Models We also qualitatively compared the information our framing lexicons provide with information provided by a Structured Topic Model (STM) (Roberts et al., 2013). We find that our approach is better able to capture frames the way a reader might conceptualize them, whereas topic models are useful for finding fine-grained corpus-specific topics.

Topic models are a common way to analyze frames in a text (Nguyen et al., 2013); the STM specifically allows correlation between topics and covariates. We trained an STM with 10, 15, 20, 25, and 50 topics on U.S.-focused articles in the *Izvestia* corpus, including publication date (month and year) as a covariate. We selected the 20 topic model as having the most coherent topics. Throughout this section, we refer to topics using their most representative words as determined by the "Lift" metric (Roberts et al., 2013).

We randomly selected a sample document for each primary frame to investigate. The framing lexicons are able to connect corpus-specific vocabulary to higher-level concepts. For example, an article describing movies about the U.S. prison facility at Guantanamo Bay has two main STM topics: [laden, sentence, prison] and [author, viewer, filming]. Similarly, the framing lexicons identify ‘Cultural Identity” as the primary frame. However, a secondary frame in the document is “Morality”, captured by words: writer, form, Christ, art. While both the STM and the framing lexicons capture major details of the article, the framing lexicons additionally tie the article to morality, because words like “art” in this corpus are often signs of a moral framework.

Nevertheless, when the STM identifies a topic similar to a frame, we find correlations with the related lexicon, i.e. there is a 0.75 correlation between the frequency of words in the Legality, Constitutionality, Jurisdiction lexicon and the monthly average proportion of each document assigned to the topic [yukos, bill, legislation].

Additionally, the framing lexicons tend to have higher precision in identifying relevant articles than the STM. Topics are commonly identified by their most probably words, which may not occur at all in documents associated with the topic. For example, the STM assigns an article about smoking policies in the U.S. to 3 main topics: [laden, sentence, prison], [kosovo, falcons, because], [author, viewer, filming], none of which are closely related to the article. In contrast, because assignments to the framing lexicons are made directly from words in the lexicon, we can be confident that articles assigned to each frame have words from the actual lexicon, and are very likely related to the frame. The framing lexicons assign the primary frame as “Policy” for this article, which is a good fit. Neither method captures that the article is also related to health.

Finally, the STM is useful for finding fine-grained topics, beyond the Policy Frames Codebook. For example, we find a “sports” topic: [match, nhl, team]. These topics tend to be corpus-specific and more concrete than the framing lexicons: no STM topic captures “Quality of Life”.

5 Identifying Media Manipulation

We first use the generated framing lexicons to determine which frames are frequently associated with the U.S. We then break the frames into finer-categories and manually look at sample articles to determine why associating these frames with the U.S. constitutes a media manipulation strategy. We find that as the stock market declines, not only is news focused more on the U.S., but also emphasizes threats to the U.S.

5.1 Salient frames

To estimate which frames are associated with the U.S. we compute *normalized pointwise-mutual information* (nPMI) between the U.S. and each frame F[5] by mapping the mutual information score onto a [-1,1] scale. A value of 1 represents complete co-occurrence; a value of 0 represents complete independence. By using nPMI, we measure which frames are *overrepresented* in U.S.-focused news, as compared to other news.

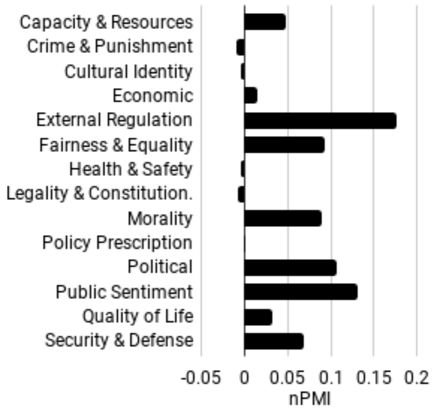

Figure 2: nPMI between U.S. and each frame.

Figure 2 shows the nPMI score between the U.S. and each frame for all articles in our corpus. As any news article about the U.S. is by definition externally focused, the frame with the strongest association is unsurprisingly “External Regulation and Reputation”. Other frames with strong associations include “Morality”, “Political”, “Public Sentiment”, and “Security and Defense”. These frames demonstrate what type of news events in the U.S are reported in Russia. As an example, we look at an article that uses a combination of these frames. The article describes cooling relations between Russia and the U.S. It explains that anti-Russian sentiment will be prevalent in the U.S. during upcoming elections, when politicians on both sides will play the “Russia card”. It ultimately attributes the cooling relations

---

[5] As above, we consider an article to be U.S.-focused if it mentions the U.S. at least 2 times, and we consider an article to employ frame F if it uses at least 3 words from F’s lexicon.

to a mismatch of values and ideology between the two nations. The framing lexicons well-capture the numerous themes in this article. Specifically, the frames identified and the related framing lexicon words are:

Political: electorate, election, former, pre-election, political scientists, congress, president, post, bush

Public Sentiment: elections, campaign, pre-election, democrats, republicans

External Regulation and Reputation: west, war, former, washington, politics, summit, exacerbation, west, decision, bush, president

Morality: peace, sins, ideals, love, values

Fairness and Equality: politics, love, values

This article uses several strategies to promote unity in Russia and actively separate Russia from Western culture, including criticizing American politics and emphasizing a difference of values. Russian articles use a combination of frames to describe the U.S., which demonstrates the importance of looking at all frames in a document, rather than just the primary frame. In the following sections, we provide additional examples demonstrating how combinations of frames can generate anti-U.S. sentiment.

5.2 Salient words within frames

We expect different aspects of frames to be foregrounded during economic upturns than in downturns. To investigate these differences, we define a set of months M_t^+, as the 10% of months where RTSI showed the greatest growth, and a corresponding set M_t^- where RTSI showed the greatest decline. We then take M_{t+1}^+ as the month directly following every month in M_t^+, and we similarly define M_{t+1}^-. From the analysis in §2.2, we expect media manipulation strategies to decline from M_t^+ to M_{t+1}^+, and increase from M_t^- to M_{t+1}^-. For each frame, we take the subset of U.S.-focused articles that use the frame. Then, we use log odds with a Dirichlet prior (Jurafsky et al., 2014; Monroe et al., 2008) to identify words that are overrepresented or underrepresented from M_t^+ to M_{t+1}^+ and from M_t^- to M_{t+1}^-.[6] Thus, for each frame, we identify words which become more common after a stock market downturn and become less common after a stock market upturn. We refer to these words as AgendaLex.

We found the Security and Defense AgendaLex and the Crime and Punishment AgendaLex to be surprisingly coherent, both containing words related to terrorism and countries enemy to the U.S., including bombs, missiles, Guantanamo, North Korea, Iraq, etc. We found a correlation of -0.49 between the frequency of words from the Security and Defense AgendaLexin U.S.-focused articles and the RSTI (-0.49). A 1-lag Granger causality test (to what extent does a change in RTSI Granger-cause a change in the prevalence of the Security and Defense AgendaLex?) has a p-value of 0.0051. As the stock market declines, not only does the news focus more on the U.S., the news focuses specifically on terrorists and other enemies to the U.S. In the next section, we refine this conclusion by looking at sample articles.

5.3 Examples of framing during downturns

By reading sample articles from months just after stock market downturns that used words from the Security and Defense lexicon and AgendaLex, we identified three common strategies for distracting Russian citizens from negative economic events: villainizing the U.S., describing threats to the U.S., and promoting the Russian military over the U.S. military.

First, some articles focus on immoral actions of the U.S. military, describing U.S. troops as "Nazi", or U.S. campaigns in Iraq as "barbaric" or causing "horror and outrage throughout the world". Others discuss Guantanamo Bay, employing the "Morality" or "Legality, Constitutionality, Jurisdiction" frames. By portraying the U.S. government negatively, actions of the Russian government appear positively by comparison. Promoting unity by presenting an external enemy is a well-studied political strategy.

Second, many articles, often in passing, described threats to the U.S. An article about terror attacks in Paris mentions U.S. involvement with words from the Crime and Punishment and Security and Defense lexicons: terrorist, terrorists, special services. An article about the conflict between Israel and Palestine describes increased security in the U.S. An article about a U.S. military operation refers more directly to threats to the U.S. by claiming the killed terrorist will simply be replaced "and everything will start afresh - explosions, chases, roundups...unlucky businessmen, successful terrorists". The articles

---

[6] We take the 500 words with the largest increase in salience from M_t^- to M_{t+1}^- intersected with the 500 words with the largest decrease in salience from M_t^+ to M_{t+1}^+.

portray the U.S. as an unsafe place to live, making Russia seem like a preferable home.

A third type of article also presents Russia as safe by downplaying U.S. military threat: “the missile defense system of the USA does not pose a real threat to Russia’s strategic nuclear forces.” or describing the growth of Russian technology compared to ‘impotent’ American counterparts.

6 Related Work

Most studies on Russian media manipulation focus on state-owned television networks, such as *Channel 1* and *RT*. Strategies identified in these outlets include spreading confusion (Paul and Matthews, 2016) and “selection attribution”, in which negative economic events are attributed to foreign entities and positive events are attributed to Russian officials (Rozenas and Stukal, Forthcoming). Similar strategies have been identified in social media in China and in Venezuela: these regimes flood communication channels with irrelevant information or general “cheerleading”, presumably to distract the public from current events (King et al., 2017; Munger et al., 2018). We expand on these analyses as we study manipulation strategies in a more automated way, through Granger-causality and framing lexicons. We further draw parallels between these strategies and theories of agenda-setting and framing.

Furthermore, our method for analyzing frames contributes to the growing body of work on automated-framing analysis (Nguyen et al., 2013; Boydstun et al., 2013; Card et al., 2016; Baumer et al., 2015). While past work uses fully-supervised methods, which are not applicable to languages lacking training data, or unsupervised topic models, which can be difficult to interpret, we take a semi-supervised approach: using statistical metrics and word embeddings to generate corpus-specific lexicons based on common frameworks.

We additionally integrate our framing and agenda-setting analysis with economic indicators through the concept of Granger causality. While the concept of Granger causality is not new in economics, it is less common in NLP and social sciences. Moreover, modeling relationships between news and economic indicators is a relatively recent area. Most research has focused on using text to predict economic indicators using a variety of features, from frequencies of keywords to sentiment of social media posts (Nardo et al., 2016). Kang et al. (2017) combine text and Granger causality for a different task: automatically explaining causes of time series events. Our study differs from past work in that we reverse the direction: rather than using news articles to model changes in economic data, we use economic data to show changes in news articles.

7 Conclusions

We show that natural language technology, in addition to its ability to address overt manipulation strategies like “fake news” and censorship, has the potential to shed light on more subtle political manipulation strategies, specifically distraction. We offer a way to define these strategies by drawing on social science theories of agenda-setting and framing, combining them with a novel methodology for cross-lingual projection of framing annotations. We investigate how the resulting frames are used in the Russian newspaper *Izvestia*, and show that it reports on negative events in the U.S. as a way of distracting from economic downturns in the Russian economy.

Our approach and our findings serve as a starting point for further research on automating the identification and analysis of media manipulation strategies. These include identification of more nuanced framing strategies, such as mitigation, projections of power among entities mentioned in news, identification of over- and under-represented events, and in general, detection of biases in news articles as a means for understanding trustworthiness in media reports.

Acknowledgments

We gratefully acknowledge our helpful reviewers and annotators and Ethan Fast for providing the NYT corpus. This research was supported by Grant No. 2017699 from the United States-Israel Binational Science Foundation (BSF), by Grant No. 1813153 from the United States National Science Foundation (NSF), and by the Stanford Cyber Initiative. Further, this material is based upon work supported by the NSF Graduate Research Fellowship Program under Grant No. DGE1745016. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

Eric Baumer, Elisha Elovic, Ying Qin, Francesca Polletta, and Geri Gay. 2015. Testing and comparing computational approaches for identifying the language of framing in political news. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1472–1482.

Amber E Boydstun, Justin H Gross, Philip Resnik, and Noah A Smith. 2013. Identifying media frames and frame dynamics within and across policy issues. In *New Directions in Analyzing Text as Data Workshop, London*.

Dallas Card, Amber E Boydstun, Justin H Gross, Philip Resnik, and Noah A Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 438–444.

Dallas Card, Justin Gross, Amber Boydstun, and Noah A Smith. 2016. Analyzing framing through the casts of characters in the news. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1410–1420.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Robert M Entman. 2007. Framing bias: Media in the distribution of power. *Journal of communication*, 57(1):163–173.

Ethan Fast and Eric Horvitz. 2017. Long-term trends in the public perception of artificial intelligence. In *Proceedings of the 2017 Conference of the Association for the Advancement of Artificial Intelligence*, pages 963–969.

William A Gamson and Andre Modigliani. 1989. Media discourse and public opinion on nuclear power: A constructionist approach. *American journal of sociology*, 95(1):1–37.

Salma I Ghanem and Maxwell McCombs. 2001. The convergence of agenda setting and framing. In *Framing public life*, pages 83–98. Routledge.

Clive WJ Granger. 1988. Some recent development in a concept of causality. *Journal of econometrics*, 39(1-2):199–211.

Yangfeng Ji and Noah A Smith. 2017. Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 996–1005.

Adam Jones. 2002. The russian press in the post-soviet era: a case study of izvestia. *Journalism Studies*, 3(3):359–375.

Dan Jurafsky, Victor Chahuneau, Bryan Routledge, and Noah Smith. 2014. Narrative framing of consumer sentiment in online restaurant reviews. *First Monday*, 19(4).

Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen, and Eduard Hovy. 2017. Detecting and explaining causes from text for a time series event. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2758–2767.

Gary King, Jennifer Pan, and Margaret E Roberts. 2017. How the chinese government fabricates social media posts for strategic distraction, not engaged argument. *American Political Science Review*, 111(3):484–501.

Maxwell McCombs. 2002. The agenda-setting role of the mass media in the shaping of public opinion. In *Proceedings of the 2002 Conference of Mass Media Economics, London School of Economics*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *In Proceedings of the 2013 International Conference on Learning Representations*.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403.

Kevin Munger, Richard Bonneau, Jonathan Nagler, and Joshua A Tucker. 2018. Elites tweet to get feet off the streets: Measuring regime social media strategies during protest. *Political Science Research and Methods*, pages 1–20.

Michela Nardo, Marco Petracco-Giudici, and Minás Naltsidis. 2016. Walking down wall street with a tablet: A survey of stock market predictions using the web. *Journal of Economic Surveys*, 30(2):356–369.

Viet-An Nguyen, Jordan L Boyd-Graber, and Philip Resnik. 2013. Lexical and hierarchical topic regression. In *Advances in neural information processing systems*, pages 1106–1114.

Christopher Paul and Miriam Matthews. 2016. The Russian "Firehose of Falsehood" propaganda model. Perspective, RAND Corporation, Santa Monica, CA.

Margaret E Roberts, Brandon M Stewart, Dustin Tingley, and Edoardo M Airoldi. 2013. The structural topic model and applied social science. In *Presented at the NIPS Workshop on Topic Models: Computation, Application, and Evaluation*, pages 1–20.

Arturas Rozenas and Denis Stukal. Forthcoming. How autocrats manipulate economic news: Evidence from russia's state-controlled television. *Journal of Politics*.

James W Tankard Jr. 2001. The empirical approach to the study of media framing. In *Framing public life*, pages 111–121. Routledge.

Oren Tsur, Dan Calacci, and David Lazer. 2015. A frame of mind: Using statistical models for detection of framing and agenda setting campaigns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1629–1638.

Marcel H Van Herpen. 2015. *Putin's propaganda machine: Soft power and russian foreign policy*. Rowman & Littlefield.

Identifying the sentiment styles of YouTube's vloggers

Bennett Kleinberg
Department of Psychology
University of Amsterdam

Department of Security
and Crime Science
University College London
b.a.r.kleinberg@uva.nl

Maximilian Mozes
Department of
Informatics
Technical University
of Munich
mozes@cs.tum.edu

Isabelle van der Vegt
Department of Security and
Crime Science
University College London
isabelle.vegt.17@ucl.ac.uk

Abstract

Vlogs provide a rich public source of data in a novel setting. This paper examined the continuous sentiment styles employed in 27,333 vlogs using a dynamic intra-textual approach to sentiment analysis. Using unsupervised clustering, we identified seven distinct continuous sentiment trajectories characterized by fluctuations of sentiment throughout a vlog's narrative time. We provide a taxonomy of these seven continuous sentiment styles and found that vlogs whose sentiment builds up towards a positive ending are the most prevalent in our sample. Gender was associated with preferences for different continuous sentiment trajectories. This paper discusses the findings with respect to previous work and concludes with an outlook towards possible uses of the corpus, method and findings of this paper for related areas of research.

1 Introduction

Vlogging or video-blogging has become one of the most popular video formats on social media platforms like YouTube. Vlogs have been referred to as 'conversational video-blogs' (Biel and Gatica-Perez, 2010) or 'monologue-like' videos (Aran et al., 2014), and are officially defined in the Cambridge dictionary of English as "a record of your thoughts, opinions or experiences that you film and publish on the internet"[1]. The vast amount of vlogs on YouTube comprises a rich body of visual and textual data, usually covering a vlogger's daily life. The transcripts of vlogs provide researchers with ample opportunity to examine natural language in this young field of communication. Nevertheless, little attention has thus far been paid to investigating the language used in YouTube vlogs.

Much of the literature concerning Youtube vlogs focuses on the visual modality (Aran et al., 2014) or meta-indicators such as views and subscriber counts (Borghol et al., 2012). With the current study, we aim to address this gap in the literature by automatically analyzing the linguistic styles used by YouTube's vloggers. Building on a novel approach to examining the continuous sentiment structure, we seek to shed light on the different temporal trajectories used by vloggers, and, by doing so, we expect to gain a deeper understanding of language use in vlogs.

1.1 Previous research on vlogs

Research on YouTube vlogs has thus far mainly focused on metadata indicators and how they impact video popularity. For example, a pronounced "rich-get-richer" effect regarding video popularity is often found. When controlling for the content, previous views are the best predictor of later video popularity (Borghol et al., 2012). In addition, video age also predicts popularity: when the content is similar, early uploaders have an advantage over later uploaders in terms of popularity (Borghol et al., 2012). Furthermore, related video recommendations shown during or after a playing video also serve as an important indicator of the view count for YouTube videos (Zhou et al., 2010).

Another strand of research has examined the visual content of vlogs. Across 2,268 single-person YouTube videos, four clusters of user activity level

---

[1] `https://dictionary.cambridge.org/dictionary/english/vlog`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3581–3590
Brussels, Belgium, October 31 - November 4, 2018.

(e.g., stationary or in motion), editing choices, and video quality were identified (Aran et al., 2014). Videos from a cluster with highly edited, active vlogs received more views than simple conversational videos in which the vlogger is stationary in front of the camera. Concerning audio data, videos in which the vlogger is "talking more, faster, and using few pauses" receive more views on average (Biel and Gatica-Perez, 2010). Similarly, the distance of the vlogger to the camera, looking time, and 'looking-while-speaking' were also found to significantly correlate with view count. It has been suggested that these nonverbal characteristics might be related to specific personality traits that promote effective communication and video popularity (Biel and Gatica-Perez, 2010). Lastly, a smaller line of research examines the linguistic content of YouTube vlogs. For example, a manual inspection of a random sample of 100 vlogs suggests that female vloggers are more likely than male vloggers to vlog about personal matters (Molyneaux et al., 2008). In contrast, male YouTubers focused more on entertainment and technology. Research on the linguistic content of vlogs is scarce, which is why the present paper presents a novel methodology and corpus for examining the text modality. In the next section, the proposed method is explained in conjunction with previous research utilizing intra-textual sentiment analysis.

1.2 Continuous sentiment trajectories

With increasing amounts of textual data available online, diverse methods for examining natural language and its linguistic style and content are applied on a large scale. Applications range from the automated detection of fake reviews (Ott et al., 2013), fake news (Thorne et al., 2018; Pérez-Rosas et al., 2018) and lies and truths (Mihalcea and Strapparava, 2009; Kleinberg et al., 2018) to detecting political sentiments (Cambria, 2016).

One important strand of computational linguistics examines the sentiment of texts (i.e., how positive or negative the emotional valence of the content is). Typically, a sentiment score is reported for a piece or section of text (e.g., a sentence or a whole text). However, a recently emerged literature examines shifts in emotional valence throughout a text (hence intra-textual) to assess sentiment over time. This focus on the temporal dimension resembles work on storyline extraction, where the timeline of events is considered, including events that give rise to a 'climax' of a story (Caselli and Vossen, 2016). With intra-textual sentiment analysis, it is also possible to visualize language use leading to a climax in (positive or negative) sentiment. This method has thus far been applied to extract the intra-textual sentiment dynamics in novels (Gao et al., 2016) and was used to identify key narrative moments. Within this same context, Jockers (2015a) analyzed the intra-textual sentiments of over 40.000 novels using hierarchical clustering and observed six shapes, each representing a common sentiment structure in novels (e.g., a 'man-in-hole' structure showcasing a positive-negative-positive sentiment).

A few studies have further examined emotional plot shapes in different bodies of text. Six common emotional arcs were found to underlie 1,327 fictional stories, termed the 'rags to riches' (i.e., a rise in sentiment), 'tragedy' or 'riches to rags' (i.e., a fall in sentiment), 'man in hole' (i.e., fall-rise), 'Icarus' (i.e., rise-fall), 'Cinderella' (i.e., rise-fall-rise), and 'Oedipus' styles (i.e., fall-rise-fall; see Reagan et al., 2016). Comparing these emotional arcs to the number of downloads for each of the fiction titles, the authors suggest that 'Icarus', 'Oedipus', and 'Man in a hole' are the most successful plot shapes.

In a different study IBM's Watson Tone Analyzer was used to examine the continuous trajectories (i.e., in terms of emotion: e.g., joy; language: e.g., analytical language, and personality: e.g., extraversion) in public speaking (Tanveer et al., 2018). The authors investigated the audience's perception of linguistic structures of 2,007 publicly available TED talks. After identifying clusters and comparing these to ratings on the TED website, they found that 'flat' (i.e., less diverse) plot shapes are more likely to be rated as 'long-winded', emphasizing the importance of narrative variety for speech success. Furthermore, the majority of speeches showed a positive ending. While specific linguistic (or visual) characteristics of a video may not be accessible prior to watching a video, its style and structure may play a role in capturing and holding on to the attention of a viewer. Since video streaming services tend to only consider "quality views" (i.e., longer than X amount of time, (YouTube.com, 2018), examining the continuous sentiment styles over narrative time may be a worthwhile endeavor.

1.3 The current study

This investigation aims to use intra-textual sentiment analysis to examine the continuous sentiment styles used in YouTube vlogs. Specifically, we primarily aim to identify distinct trajectories of continuous sentiment. In addition, we explore the sentiment styles' relationship to gender and their overall prevalence on YouTube. Besides these aims, we introduce a novel method of extracting sentiment that is sensitive to valence shifters but can be used on non-punctuated data (e.g., YouTube vlogs). We also present a publicly available corpus of YouTube vlog transcripts.

2 Data

The data and code used to produce the analysis and data in this paper are publicly available.[2] [3]

2.1 Vlog selection

Vlogs were selected by choosing vlog channels (i.e., where YouTubers post their videos) from a list of the top 250 YouTube channels (ranked by subscriber count) in the 'People & Blogs' category (retrieved from www.socialblade.com on 29 March 2018). Channels were selected for inclusion if the videos on the channel were in English and the channel solely focused on vlogs – excluding channels with gaming, music and prank videos. The aim was to select a sample of 20 female and 20 male vloggers from the list of 250 YouTube channels. However, the majority of channels in this list were male or family vloggers, resulting in a somewhat unbalanced sample. After inspecting all 250 channels in the list, 37 channels that satisfied the aforementioned criteria remained.

2.2 Scraping vlog transcripts

To obtain vlog transcripts, we accessed all videos published on the selected YouTube channels and used www.downsub.com to access video transcripts. That website downloads and returns YouTube transcripts for a specified video URL directly in the browser. We developed a python script that takes video URLs as input and communicates indirectly with YouTube via downsub.com to request and retrieve video transcripts. We did not differentiate between manually-added and automatically-generated transcripts but encountered several cases in which neither was available for a certain video. In this case, we ignored the affected video and proceeded without considering it further in the analysis.

2.3 Preprocessing

The retrieved transcripts were XML-encoded, and the preprocessing included the removal of all XML tags to provide human-readable sequences. Furthermore, each row of the XML transcripts contained the word sequence of the vlog with its corresponding start and end time. Since the YouTube transcripts do not include any punctuation information, we decoded each row by removing the XML tags and merged the resulting human-readable sequences to one continuous string for each vlog.

2.4 Feature extraction

The primary aim of this investigation was to examine the continuous sentiment structures of popular vlogs. Taking inspiration from previous work on the narrative arcs of novels (Gao et al., 2016), we wanted to analyze how the sentiment of the spoken content of a vlog moves dynamically throughout the vlog's narrative time. When conducting sentiment extraction, an important consideration is the treatment of valence shifters that influence the meaning of a sentiment. For example, an utterance such as "this was not a bad day" should be rated as a positive sentiment since the valence of "bad" is shifted through the negator "not". Existing methods of extracting sentiment, however, either ignore valence shifters (Jockers, 2015b) or are sentence-based (i.e., they provide weighted sentiments only per sentence, e.g., the *R*-package *sentimentr*, Rinker, 2018b). Since the scraped transcripts are non-punctuated, but we also wanted to account for valence shifters, we built a "naïve context" sentiment extractor that is sensitive to negators (e.g., not, doesn't), [de-]amplifiers (e.g., really, hardly), and adversative conjunctions (e.g., but, however). The rationale is inspired by the algorithm behind *sentimentr* (Rinker, 2018b) but extends that approach to data that are non-punctuated or very brief.

Specifically, our algorithm identifies each sentiment as matched with the 'Jockers & Rinker Polarity Lookup Table' from the *lexicon* R package

---

[2] Data and code for vlog scraping: `https://github.com/ben-aaron188/narrative_structures`.

[3] Code for feature extraction algorithm: `https://github.com/ben-aaron188/naive_context_sentiment`

(Rinker, 2018a) and then constructs a "naïve context" cluster (i.e., without relying on punctuation or other structure) around that sentiment (here: two words before and after the sentiment). For each context cluster, the raw sentiment is then weighted by the presence of valence shifters. For example, "this was not a bad day in the sun" would result in a cluster around the identified sentiment word "bad" of (not a **bad** day in). The weights assigned to the valence shifters in this study (Negator: -1.00, Amplifier: 1.50, De-amplifier: 0.50, Adversative conjunction: 0.25) are motivated by those in the similar software package *sentimentr*.[4] Lastly, the sentiment word in the cluster is replaced by its sentiment value: (-1, 1, -0.75, 1, 1). We can then calculate the product of all vector elements and retrieve the weighted sentiment of that cluster (e.g., 0.75 for "not a bad day in").

We performed the naïve sentiment extraction on each vlog transcript resulting in a vector consisting of zeros (for words that did not match the sentiment lookup table) and weighted sentiment values. That vector was transformed to a standardized narrative time from 0 to 100 using the discrete cosine transformation from the *syuzhet* R package (Jockers, 2015b). The sentiment values were scaled from -1 (lowest sentiment per vlog transcript) to +1 (highest sentiment per vlog transcript). The transformation yielded a vector of 100 sentiment coordinates for each transcript.

3 Results

3.1 Corpus statistics

The final corpus consisted of the transcripts of 27,333 vlogs from 24 vloggers with a total corpus size of 40,318,924 words and vlogs accounting for more than 24 billion views. These are all vlogs of which we were able to retrieve a transcript of at least ten words. Table 1 shows that female vloggers were underrepresented in the sample. More than half of the vlogs stem from male vloggers and a third from families who vlog. Since view count is highly dependent on the number of days the vlog is online, we corrected the view count for each vlog by dividing it by the number of days it was online. We then excluded 462 vlogs (1.69%) that were considered view count outliers (i.e., more than three standard deviations above the mean). All subsequent analyses were conducted on the final sample with these outliers excluded.

	Overall	**Female**	**Male**
# of vlog channels	24	12.50% (n = 3)	54.17% (n = 13)
# of videos	27,333	7.64% (n = 2,087)	49.02% (n = 13,399)
Videos/channel	1138.88	695.67	949.92
Avg. length (# of words)	1475.10 (746.53)	1480.06 (610.72)	1366.31 (804.68)
Avg. view count	893,932 (2,566,021)	456,603 (718,895)	898,089 (1,900,855)
Avg. view count (after outlier removal)	762,143 (1,479,588)	412,592 (490,792)	817,029 (1,476,804)
Stand. view count	1553.69 (2733.35)	1457.40 (2508.95)	1620.12 (2875.96)

Table 1. Descriptive statistics (mean, SD).

3.2 Identifying continuous sentiment styles

The primary aim of this paper was to examine whether the narrative structure of vlogs can be captured by a few overarching sentiment styles. We used the binned sentiments extracted for each vlog in an unsupervised non-hierarchical *k*-means cluster analysis. Using the within-cluster-sum-of-squares in a scree plot for 1 to 30 clusters, we observed an inflection after seven clusters and therefore decided to build a *k*-means model with *k*=7 (Figure 1 and 2).

The *k*-means model assigned one cluster to each vlog transcript, and we subsequently averaged the continuous sentiment structure of all vlogs

---

[4] Consensus on values for valence shifter weights has yet to emerge in literature. The algorithm is available as an *R*-implementation to modify these values.

Cluster	Description	Label	% of vlogs	Avg. length	Stand. view count
1	Highly positive start, followed by semi-negative section, mildly positive section, negative ending	“Downhill from here”	16.06	1477.41 (741.71)	1524.52 (2738.15)
2	Negative start, followed by positive section, mildly negative section, semi-positive ending	“Mood swings”	11.84	1503.70 (759.19)	1504.72 (2702.29)
3	Negative first half, positive second half	“Rags to riches”	15.92	1454.41 (727.83)	1557.04 (2697.74)
4	Positive first half, negative second half	“Riches to rags”	12.36	1485.32 (769.01)	1477.71 (2615.45)
5	Semi-positive start, followed by a highly negative section, semi-positive end	“Bump in the road”	13.46	1480.21 (729.47)	1507.75 (2650.23)
6	Majority of the narrative is semi-negative, with highly positive end	"End on a high note"	18.39	1507.69 (701.18)	1818.73 (3030.10)
7	Majority of the narrative is positive with two peaks	“Twin peaks”	11.97	1402.86 (788.97)	1368.05 (2511.51)

Table 2. Narrative styles taxonomy and descriptive statistics.

belonging to each cluster. Figure 1 and 2 show the distinct average shapes of the continuous sentiment structures of each cluster. The dotted red lines indicate the upper and lower boundaries of the sentiment shape +/- one standard deviation; the blue lines show the 99% confidence intervals. The seven plot shapes represent the average sentiment of all vlogs in that shape cluster in each bin of standardized narrative time (1-100). For example, for the first shape (Figure 1, second plot), the sentiment starts highly positive in the first quantile of the narrative time and then quickly dips below zero to moderately negative sentiment. Halfway through the narrative time, the sentiment neutralizes and becomes negative again until the last quantile from which it gradually becomes less negative and ends in a nearly neutral sentiment.

Table 2 proposes a taxonomy of these seven continuous sentiment styles of YouTube vlogs and provides descriptive statistics for each. We used a linear mixed effects model to test for an effect of the clusters on the view count and transcript length. To account for dependence in the data (i.e., that we have multiple transcripts per vlogger), we included the vlogger as a random effect in the model. The analysis indicated that there was no significant difference in the corrected view count between the clusters; non-standardized β-coefficient of cluster = 6.78, *se* = 7.40, $t(451.25) = 0.92$, $p = .360$. Nor was there a difference in transcript length; $\beta = -1.23$, *se* = 2.07, $t(114.10) = -0.60$, $p = .552$. Thus, it is not statistically justified to argue that one particular narrative style attracted more views than others. Nor do these results provide support for a relationship between a vlog’s length (as measured by the number of words in its transcript) and a particular sentiment style (e.g., that lengthier vlogs contain more sentiment shifts).

A one-sided Chi-square test revealed that the number of vlogs was not uniform per cluster, $X^2(6) = 720.61$, $p < .001$. The observed frequencies of vlogs in each cluster deviated significantly from the frequency that is expected if the vlogs were distributed uniformly ($n = 3832$, 14.29%). Standardized residuals (z-scores) of the observed relative to the expected frequencies can be used to examine which observations deviated in which direction. There were significantly more vlogs than expected in the “downhill-from-here” cluster ($z = 8.30$), the “rags-to-riches” cluster ($z = 7.64$), and the “end-on-a-high-note” cluster (z = 19.21). The clusters “mood swings” (z = -11.45), “riches-to-rags” (z = -9.02), “bump-in-the-road” (z = -3.86), and “twin peaks” (z = -10.82) were significantly underrepresented. Aside from the considerable overrepresentation of the “end-on-a-high-note” cluster and underrepresentation of the “mood swings” and “twin peaks” clusters, the distribution is rather harmonious.[5]

3.3 Additional analysis: Sentiment styles and gender

We also assess whether there is a relationship between gender and the continuous sentiment style. There was significant association as indicated with a 3 (male, female, family) by 7 (clusters) Chi-square test, $X^2(12) = 134.82$, $p < .001$. Table 3 shows that family vloggers used the “twin peaks” style significantly more often than expected and used the “end-on-a-high-note” style significantly less often than expected (* = significant at $p < .01$). For female vloggers, the most used continuous sentiment style was “riches to rags”, while they used “end-on-a-high-note” less often than expected. Male vloggers preferred the "end-on-a-high-note" style while they used the styles "downhill from here" and "twin peaks" less often. Since we did not ascertain a balanced gender distribution, these findings should be treated cautiously and subjected to replications on datasets more suitable for gender analysis.

Cluster	Family	Female	Male
Downhill from here	2.23	1.26	-2.88*
Mood swings	-2.31	1.96	1.25
Rags to riches	2.13	-1.95	-1.08
Riches to rags	-2.05	4.88*	-0.56
Bump in the road	1.69	-1.12	-1.08
End on a high note	-5.16*	-6.03*	8.32*
Twin peaks	3.83*	2.25	-4.99*

Table 3. Standardized residuals for the cluster-by-gender association.

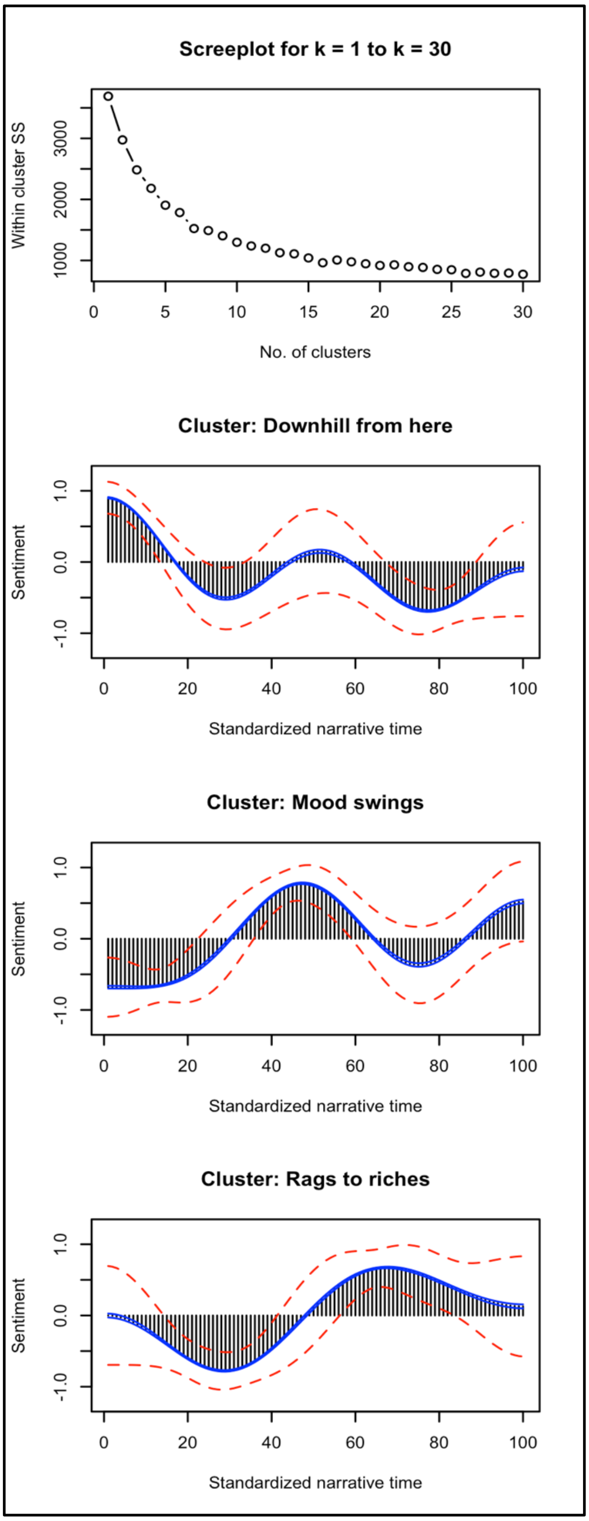

Figure 1. Scree plot and average sentiment style shapes for clusters 1-3. *Note:* Dotted red lines = +/- 1 SD; blue lines = 99% CI.

---

[5] The appendix provides a list of URLs to vlogs and vlog channels typical of a particular style.

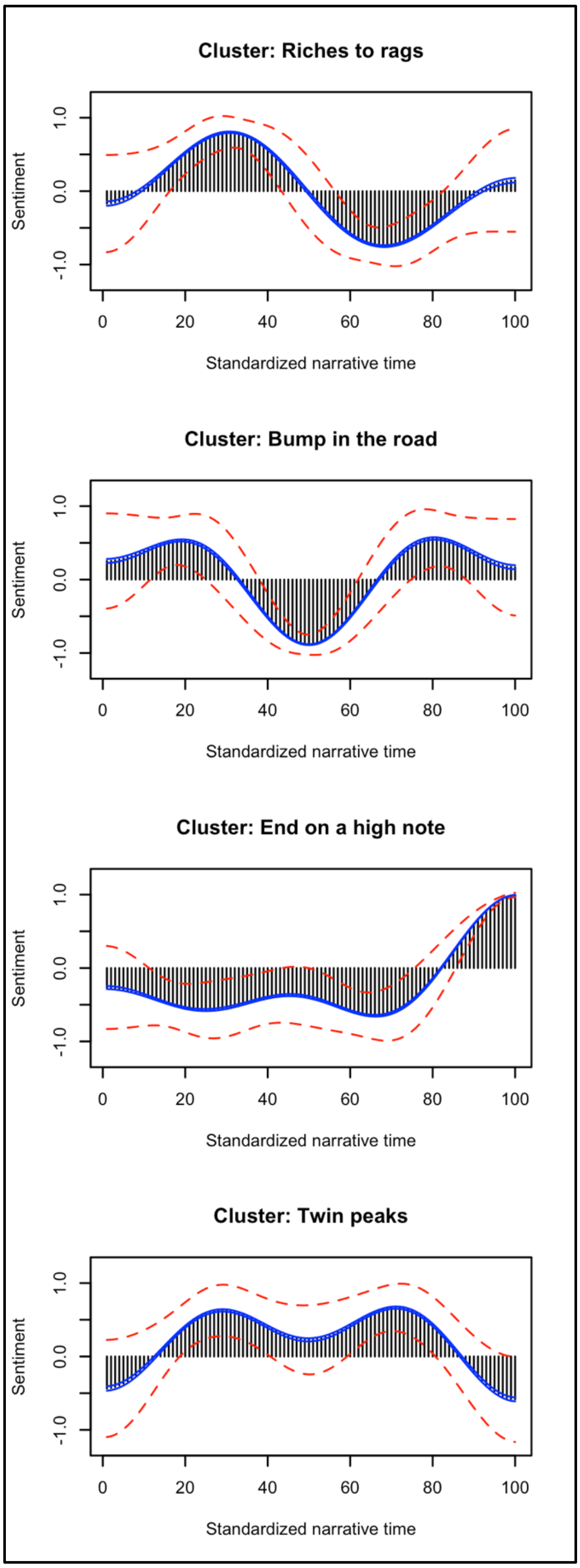

Cluster: Riches to rags

Cluster: Bump in the road

Cluster: End on a high note

Cluster: Twin peaks

Figure 2. Average sentiment style shapes for clusters 4-7. *Note:* Dotted red lines = +/- 1 SD; blue lines = 99% CI.

4 Discussion

In this paper, we examined the continuous sentiment structures of a new corpus of YouTube vlog transcripts using intra-textual sentiment analysis. We were able to identify seven distinct continuous sentiment styles. The “rags to riches”, “riches to rags” and the “end-on-a-high-note” styles displayed an alternating pattern with one shift between predominantly positive and negative content. By contrast, the “bump in the road” and “twin peaks” styles show a marked negative middle part of the vlog and an overall positive sentiment with a slightly negative beginning and end, respectively. The most volatile styles were the “downhill from here” and the “mood swings” styles with both showing three shifts in sentiment.

While no continuous sentiment style was related to higher view counts, we observed that the “end-on-a-high-note” style was the most common in our corpus. The pattern of this particular style suggests that vloggers build up their content towards a positive ending following the idea to ‘end strong’, possibly to engage their viewers to keep watching subsequent videos. A similar trajectory was found for TED speeches (Tanveer et al., 2018), suggesting that YouTube vloggers may employ strategies similar to those of prolific public speakers. However, since the “downhill from here” style is second most frequent in our dataset, these findings do not indicate any tendency to a general sentiment development over time. In fact, the “downhill from here” and “end on a high note” styles exhibit a somewhat opposing behavior since the first gradually decreases the sentiment over time whereas the second aims at developing a positive sentiment towards the end of the video. When projecting this observation to the gender-specific findings, one can see that the differences between family, female and male vloggers might account for these contradictory findings. Both family and female vloggers (representing 45.83% of all vloggers) used the “downhill from here” more often than male vloggers, whereas male vloggers used the “end on a high note” style most often.

Gender differences might help further understand vlog style choice: family vloggers used the "twin peaks" style more often than female or male vloggers, with the latter significantly under-using that style. Female vloggers resorted to the "riches to rags" style most often, while male vloggers preferred the "end-on-a-high-note" style.

Vlogs made by females started with a definite positive buildup followed by a marked dip towards clear negative sentiment and a gradual increase towards the positive near the end of the vlog. Family vloggers were the only ones who preferred a style that ended with a sentiment leading toward the negative.

Contrary to the findings of Tanveer et al. (2018), we did not find support for the notion that diverse styles characterized by higher sentiment volatility are more popular: both YouTuber's themselves, as well as the audience, did not show a preference for the most volatile narrative styles. The intra-textual analysis approach helped us to uncover dynamics in the sentiment of vlogs that would otherwise have been blurred by an overall sentiment score. In many cases (e.g., for the "rags to riches" and "riches to rags" styles) the sentiment dynamics would have canceled each other out to a neutral sentiment.

4.1 Limitations

Despite the interesting initial findings using the new corpus of vlog transcripts and the linguistic-temporal analysis, the current study is not without its limitations. First, we partially relied on automated transcription of the vlogs, so the retrieved transcripts consisted of texts generated with YouTube's speech recognition software. Although recent advancements in machine learning research provide promising improvements to speech recognition (LeCun et al., 2015; Zhang et al., 2017b; Zhang et al., 2017a) and contributed to an accuracy increase of fifty percent for YouTube's automatic captions in English (Official YouTube Blog, 2017), the automatic generation of YouTube video captions remains a challenging task (Liao et al., 2013). It could be that potential inaccuracies might have affected our findings and future studies could set out to replicate our results using user-provided high-quality vlog transcripts.

Second, the sample consisted of YouTubers that already are successful concerning subscriber count, and we might, therefore, have painted a skewed picture of the vlogging landscape. For future studies, it would be interesting to look at potential moderating variables such as the vloggers 'vlogging age' or the vlog's topic. Possibly, the use of sentiment (and other linguistic dimensions) might differ between starting YouTubers and those who are more prolific and regularly attract millions of views.

Third, by using vlog transcripts, we resorted solely to the linguistic modality of vlogs and did not look at the use of visual aspects in the vlogs. Using a similar dynamic method, future studies could look at how the use of visual content behaves and interacts with language use over the narrative time.

4.2 Future work

Using the corpus, method, and findings presented here as a starting point, we hope that future research can extend this exploratory work.

- For example, it would be interesting to perform unsupervised clustering analyses on the vlog level to find "vlog-twins" or even "vlogger-twins". Using observations that are similar in a multidimensional feature space could be exploited for semi-experimental studies where the effects of individual variables (e.g., a change in vlogging style) can be isolated towards drawing conclusions of causal nature.

- Moreover, the sentiment is only one dimension of the linguistic aspects of vlogs, and the current analysis might be extended to different constructs. For example, a significant challenge lies in detecting extremist content on social media platforms like YouTube (Burgess, 2017) and the dynamic approach might help uncover extremist parts in videos and enable a more fine-grained inspection of vlogs.

- The temporal aspect of narrative time could be broadened so that the evolution of the narrative style across videos of individual vloggers can be captured. This would be a useful method to identify changes in vlogging strategy.

- Finally, the intra-textual method might help in detecting online misinformation or deception within texts. A major challenge, for example, for linguistic deception detection lies in identifying lies embedded within a mostly truthful statement (Bachenko et al., 2008). Similarly, it would be interesting to examine whether a dynamic linguistic-temporal approach as used here can aid in the detection of misinformation online (e.g., Pérez-Rosas et al., 2018). The method used here and in a few related studies (Jockers, 2015a; Gao et al., 2016) could, therefore, also be of interest to researchers across disciplines.

5 Conclusion

Vlogging is a unique and novel means of communication. We explored the transcripts of vlogs as a source for linguistic analysis, and, by looking at intra-textual sentiment dynamics of each vlog, we identified seven distinct continuous sentiment styles. Vlogs ending on a positive note were the most prevalent, and we observed that gender was associated with different vlogging style preferences. The current paper presented an initial glimpse at the rich data source of vlogs, and a dynamic sentiment analysis approach helped uncover continuous sentiment structures. As such we hope the corpus, method, and findings presented here function as an impetus towards more analyses on this emerging means of online communication.

References

Oya Aran, Joan-Isaac Biel, and Daniel Gatica-Perez. 2014. Broadcasting oneself: Visual discovery of vlogging styles. *IEEE Transactions on multimedia*, 16(1):201–215.

Joan Bachenko, Eileen Fitzpatrick, and Michael Schonwetter. 2008. Verification and implementation of language-based deception indicators in civil and criminal narratives. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 41–48. Association for Computational Linguistics.

Joan-Isaac Biel and Daniel Gatica-Perez. 2010. Vlogcast yourself: Nonverbal behavior and attention in social media. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, page 50. ACM.

Youmna Borghol, Sebastien Ardon, Niklas Carlsson, Derek Eager, and Anirban Mahanti. 2012. The untold story of the clones: content-agnostic factors that impact YouTube video popularity. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1186–1194. ACM.

Matt Burgess. 2017. Google's using a combination of AI and humans to remove extremist videos from YouTube. http://www.wired.co.uk/article/google-youtube-ai-extremist-content

Erik Cambria. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107.

Tommaso Caselli and Piek Vossen. 2016. The Storyline Annotation and Representation Scheme (StaR): A Proposal. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 67–72, Austin, Texas. Association for Computational Linguistics.

Jianbo Gao, Matthew L. Jockers, John Laudun, and Timothy Tangherlini. 2016. A multiscale theory for the dynamical evolution of sentiment in novels. In *Behavioral, Economic and Socio-cultural Computing (BESC), 2016 International Conference on*, pages 1–4. IEEE.

Matthew Jockers. 2015a. Revealing Sentiment and Plot Arcs with the Syuzhet Package. http://www.matthewjockers.net/2015/02/02/syuzhet/

Matthew Jockers. 2015b. *Syuzhet: Extract Sentiment and Plot Arcs from Text*. https://github.com/mjockers/syuzhet

Bennett Kleinberg, Yaloe van der Toolen, Aldert Vrij, Arnoud Arntz, and Bruno Verschuere. 2018. Automated verbal credibility assessment of intentions: The model statement technique and predictive modeling. *Applied Cognitive Psychology*, 32(3):354–366, May.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Hank Liao, Erik McDermott, and Andrew Senior. 2013. Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 368–373. IEEE.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312. Association for Computational Linguistics.

Heather Molyneaux, Susan O'Donnell, Kerri Gibson, and Janice Singer. 2008. Exploring the gender divide on YouTube: An analysis of the creation and reception of vlogs. *American Communication Journal*, 10(2):1–14.

Official YouTube Blog. 2017. One billion captioned videos. https://youtube.googleblog.com/2017/02/one-billion-captioned-videos.html

Myle Ott, Claire Cardie, and Jeffrey T. Hancock. 2013. Negative deceptive opinion spam. In *HLT-NAACL*, pages 497–501.

Veronica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018*, Santa Fe, New Mexico, USA, August.

Andrew J. Reagan, Lewis Mitchell, Dilan Kiley, Christopher M. Danforth, and Peter Sheridan Dodds. 2016. The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science*, 5(1):31.

Tyler Rinker. 2018a. *lexicon: Lexicon Data*. http://github.com/trinker/lexicon

Tyler Rinker. 2018b. *sentimentr: Calculate Text Polarity Sentiment*. http://github.com/trinker/sentimentr

M. Iftekhar Tanveer, Samiha Samrose, Raiyan Abdul Baten, and M. Ehsan Hoque. 2018. Awe the Audience: How the Narrative Trajectories Affect Audience Perception in Public Speaking. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 24. ACM.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for Fact Extraction and VERification. *arXiv:1803.05355 [cs]*, March. arXiv: 1803.05355.

YouTube.com. 2018. How video views are counted - YouTube Help. https://support.google.com/youtube/answer/2991785?hl=en

Ying Zhang, Mohammad Pezeshki, Philémon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courville. 2017a. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*.

Yu Zhang, William Chan, and Navdeep Jaitly. 2017b. Very deep convolutional networks for end-to-end speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4845–4849. IEEE.

Renjie Zhou, Samamon Khemmarat, and Lixin Gao. 2010. The impact of YouTube recommendation system on video views. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 404–410. ACM.

Native Language Identification with User Generated Content

Gili Goldin
Dept. of Computer Science
University of Haifa
Israel
gili.sommer@gmail.com

Ella Rabinovich*
Dept. of Computer Science
University of Toronto
Canada
ellarabi@gmail.com

Shuly Wintner
Dept. of Computer Science
University of Haifa
Israel
shuly@cs.haifa.ac.il

Abstract

We address the task of native language identification in the context of social media content, where authors are highly-fluent, advanced nonnative speakers (of English). Using both linguistically-motivated features and the characteristics of the social media outlet, we obtain high accuracy on this challenging task. We provide a detailed analysis of the features that sheds light on differences between native and nonnative speakers, and among nonnative speakers with different backgrounds.

1 Introduction

The task of *native language identification* (NLI) aims at determining the native language (L1) of an author given only text in a foreign language (L2). NLI has gained much popularity recently, usually with an eye to educational applications (Tetreault et al., 2013): the errors that learners make when they write English depend on their native language (Swan and Smith, 2001), and understanding the different types of errors is a prerequisite for correcting them (Leacock et al., 2010). Consequently, tutoring applications can use NLI to offer better targeted advice to language learners.

However, the NLI task is not limited to the language of learners; it is relevant also, perhaps even more so, in the (much more challenging) context of highly-fluent, advanced nonnative speakers. While the English language dominates the internet, native English speakers are far outnumbered by speakers of English as a foreign language. Consequently, a vast amount of static and dynamic web content is continuously generated by nonnative writers. Developing methods for identifying the native language of nonnative English authors on social media outlets is therefore an important and pertinent goal.

We address the task of native language identification in the context of user generated content (UGC) in online communities. Specifically, we use a large corpus of English *Reddit* posts in which the L1 of authors had been accurately annotated (Rabinovich et al., 2018). On this dataset, we define three closely-related tasks: (i) distinguishing between native and nonnative authors; (ii) determining to which language family the native language of nonnative authors belongs; (iii) identifying the native language of nonnative authors. Importantly, we employ features that take advantage of both linguistic traits present in the texts and the characteristics of the social media outlet. We obtain excellent results: up to 92% accuracy for distinguishing between natives and nonnatives, and up to 69% for the 23-way NLI classification task.[1]

The contribution of this paper is manifold. First, this is one of the first works to address NLI with highly-advanced nonnatives; it is also among the first to address the task in the context of UGC. Furthermore, we define a plethora of features, some which have been used in earlier works but others that are novel. In particular, we define a set of features that rely on the characteristics of the social media outlet, thereby extending the task somewhat, from linguistic analysis to user profiling. Finally, we provide a detailed analysis of the results, including the specific contribution of various features and feature sets. This analysis will be instrumental for future extensions of our work.

2 Related work

The NLI task was introduced by Koppel et al. (2005), who worked on the International Corpus of Learner English (Granger, 2003), which includes texts written by students from Russia, the

---

*Work done while the second author was at the University of Haifa.

[1] It is important to note that some of our features are specific to the Reddit corpus and will not easily generalize to other datasets.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3591–3601
Brussels, Belgium, October 31 - November 4, 2018.

Czech Republic, Bulgaria, France, and Spain. The same experimental setup was adopted by several other authors (Tsur and Rappoport, 2007; Wong and Dras, 2009, 2011). The task gained popularity with the release of nonnative *TOEFL* essays by the Educational Testing Service (Blanchard et al., 2013); this dataset has been used for the first NLI Shared Task (Tetreault et al., 2013) and also for the 2017 NLI Shared Task (Malmasi et al., 2017).

Our task is closely related to the task of *dialect identification*, in which the goal is to discriminate among similar languages, language varieties and dialects. Classic machine learning classification methods are usually applied for this task, often with SVM models. The best reported features include word and character n-grams, part of speech n-grams and function words (Zampieri et al., 2017; Malmasi and Zampieri, 2017).

The current state of the art in NLI, according to Malmasi and Dras (2017), utilizes some variant of contemporary machine learning classifier with the following types of features: (i) word, lemma and character n-grams, (ii) function words (FW), (iii) part-of-speech (POS) n-grams, (iv) adaptor grammar collocations, (v) Stanford dependencies, (vi) CFG rules, and (vii) Tree Substitution Grammar fragments. The best result under cross-validation on the TOEFL dataset, which includes 11 native languages (with a rather diverse distribution of language families), was 85.2% accuracy. Applying these methods to different datasets (the ASK corpus of learners of Norwegian (Tenfjord et al., 2006) and the Jinan Chinese Learner Corpus (Wang et al., 2015), 10-11 native languages in each) resulted in 76.5% accuracy for the Chinese data and 81.8% for the Norwegian data, with LDA-based classification yielding top results.

Notably, all these works identify the native language of *learners*. Identifying the native language of advanced, fluent speakers is a much harder task. Furthermore, our dataset includes texts by native speakers of 23 languages, more than double the number of languages used in previous works; and our L1s are all European, and often typologically close, which makes the task much harder.

Two recent works address the task of NLI on UGC in social media. Anand et al. (2017) summarized the shared task on Indian NLI: given a corpus of Facebook English comments, the task was to identify which of six Indian languages is the L1 of the author. The best reported result was 48.8%, obtained by an SVM with character and word n-grams as features. These are content based features that are highly domain-dependent and are not likely to generalize across domains. Volkova et al. (2018) explored the contribution of various (lexical, syntactic, and stylistic) signals for predicting the foreign language of non-English speakers based on their English posts on Twitter. This effectively results in a 12-way classification task, with 12 different L1s (data sizes are distributed very unevenly), and the best results are unsurprisingly obtained with word unigrams and bigrams.

In contrast to these two studies, we work with many more L1s (23); we explore various types of features, including features based on social network structures and content-independent features; and we evaluate our classifiers both in and outside of the domain of training.

Several works address social aspects of social networks, and in particular identify "influential" users (Ghosh and Lerman, 2010; Trusov et al., 2010; Afrasiabi Rad and Benyoucef, 2011). Network structure has been shown to be useful in other tasks of user profiling, such as geolocation (Jurgens et al., 2015). Our design of the social network features (Section 3.5.4) are motivated by these works.

Works that aim to distinguish between native and nonnative authors (Bergsma et al., 2012; Rabinovich et al., 2016; Tomokiyo and Jones, 2001) typically rely on lexical and grammatical characteristics that reflect influences of L1 on L2. We used such features, but augmented them by features that can be induced from the network structure of social media outlets (Jurgens et al., 2015). To the best of our knowledge, ours is the first work that extensively exploits social network properties for the task of NLI. Our work is also inspired by research on the (related but different) task of identifying translations (Baroni and Bernardini, 2006; Rabinovich and Wintner, 2015; Volansky et al., 2015) and their source language (Koppel and Ordan, 2011; Rabinovich et al., 2017).

3 Experimental setup

3.1 Dataset

Reddit is an online community consisting of thousands of forums for news aggregation, content rating, and discussions. Content entries are organized by areas of interest called *subreddits*, ranging from main forums that receive much attention to smaller ones that foster discussion on niche areas. Subreddit topics include news, science, arts, and many others. An increasing body of work has used Reddit data for social media analysis (Jurgens et al., 2015; Newell et al., 2016, and many more).

We used the Reddit dataset released by Rabinovich et al. (2018). It includes Reddit posts (both initial submissions and subsequent comments), focusing on subreddits (Europe, AskEurope, EuropeanCulture) whose content is generated by users specifying their country as a *flair* (metadata attribute). We refer to these subreddits as *European*. Following Rabinovich et al. (2018), we view the country information as an accurate, albeit not perfect, proxy for the native language of the author.

Rabinovich et al. (2018) justified their trust in the accuracy of the L1 annotation; we conducted an additional validation of the data.[2] We used a specific Reddit thread in which users were asked to comment in their native language. We collected the comments in this thread of all the users in our dataset. Then, we used the Polyglot language identification tool to determine the language of the comments. We filtered out short comments, comments for which the tool's confidence was low, and comments in English of users from non-English speaking countries. Of the remaining 572 users, 479 (84%) contributed comments in the language that we considered their native. We inspected the remaining users, and for many (albeit not all) we attribute the mismatch to errors in the tool (i.e., comments in Serbian written in the Latin alphabet are wrongly predicted to be in closely-related Slavic languages). We conclude that the accuracy of the L1 annotation is high; finally, we note additionally that any noise in this labeling can only work against us in this work.

We filtered out data from multilingual countries (Belgium, Canada, and Switzerland). Rabinovich et al. (2018) showed that the English of reddit non-native authors is highly advanced, almost at the level of native speakers, making the NLI task particularly demanding.

All the posts in the dataset are associated with a unique user ID. The dataset also contains submissions and comments in other subreddits that were written by the same authors, based on their user ID. This provided us with an out-of-domain test set for evaluating the robustness of our methods. The collected data reflect about 50 (mostly European) countries, and consist of over 230M sentences, or 3.5B tokens, annotated with authors' L1.

3.2 Preprocessing

Each sentence in the dataset is tagged with the author's user ID, the subreddit it appeared in and the author's country. We segmented the dataset into *chunks* of 100 sentences, each chunk containing sentences authored by the same user.[3] The sentences were kept in their original order in the posts; users with fewer than 100 sentences were filtered out. We also left out native languages with fewer than 100 users (after the initial filtering). The resulting dataset includes 23 native languages spanning 29 countries, and consists of 34,511 unique users, almost 200M sentences and over 3B tokens. The countries and languages reflected in the dataset are listed in Table 10 (see Supplementary Materials).

All chunks were annotated for part-of-speech using Spacy. We used Aspell to spell-check the texts; every misspelled word in the original chunk was annotated with the first correction suggested by the spell checker. We also extracted from reddit additional social network properties such as users' *karma* scores, number of comments and submissions, number of comments per submission, as well as the number of months each user was active on Reddit and all the subreddits that each user in our dataset posted in (see Section 3.5.4). The processed dataset will be made publicly available.

3.3 Task

We define three classification tasks: **binary classification,** distinguishing between native and non-native authors; **language family classification** determining the language family (Germanic, BaltoSlavic, Romance, or native English) of the user; and **language identification** whose goal is to identify the native language of the user.

Different countries which have the same official language (e.g., Germany and Austria) were tagged with the same language label. For example, USA, UK, Ireland, New Zealand and Australia were all tagged with the label 'English' for the NLI task.

We then randomly downsampled the data to ensure that each class had the same number of users.

---

[2] We thank an anonymous reviewer for suggesting this.

[3] Similar classification tasks, e.g., the TOEFL task (Tetreault et al., 2013; Malmasi et al., 2017), used single essays as the unit for classification. Tasks aiming to identify translation and its source language typically use chunks of 2000 tokens (Volansky et al., 2015). We plan to experiment also with smaller chunks.

See the precise details in Section A.1 of the Supplementary Materials.

3.4 Methodology

We cast NLI as a supervised classification task and used *logistic regression* (as implemented in Scikit-learn) as a classification model. We defined several features that had been proven useful for similar tasks; some of them are general stylistic features that are presumably content-independent: these include function words, POS n-grams, simplification measures such as sentence length, etc. (Rabinovich and Wintner, 2015; Volansky et al., 2015). Other features are content based; most obviously, token n-grams, but also character n-grams (Avner et al., 2016). We expect content-based features to be highly accurate but also highly domain-dependent, and in the case of our dataset, topic-dependent. Content-independent features are expected to be weaker yet more robust.

In addition, we used features that reflect spelling and grammar errors. We assume that native and nonnative speakers make different kinds of errors in English, and that the errors of nonnatives may reveal traces of their L1 (Kochmar, 2011; Berzak et al., 2015).

Aiming to enhance the quality of classification we exploited properties that can be induced from conversational networks. We hypothesize that native speakers of the same language tend to interact more with each other (than with speakers of other languages). We hypothesize further that native speakers post more than nonnatives, and hence we defined user *centrality* measures that reflect that. We also hypothesize that native speakers' posts tend to be more spontaneous, coherent and clear, thereby drawing more attention. To reflect that, we counted the number of comments, up-votes and down-votes that were submitted to each post. While these and similar properties have been studied in the domain of social networking, to the best of our knowledge this is the first attempt to use an extensive set of features inferred from social networks for the NLI task.

3.5 Features

We designed several features to be used in all three tasks. In this section we describe these features.

3.5.1 Content features

Authors are more likely to write about topics that are related to their country and their culture, hence features that reflect content may help distinguish among authors from different countries (and, therefore, languages). For example, the word *'Paris'* is more likely to occur in texts written by French authors, while the word *'canal'* is more likely to occur in texts of Dutch authors. We defined features that take text content into account. We expect these features to yield high accuracy when testing on the training domain, but much lower accuracy when testing on different domains.

Character tri-grams The top 1000 most frequent character 3-grams in the dataset were used as features. For each chunk the value of a certain character 3-gram feature was the number of its occurrences in the chunk normalized by the total number of character 3-grams in the chunk.

Token uni-grams The top 1000 most frequent tokens in the dataset were used as features. For each chunk the value of a certain token feature was the number of its occurrences in the chunk normalized by the total number of tokens in the chunk.

3.5.2 Spelling and grammar

We used a spell checker (Section 3.1) to discover the (first) closest correction for each word marked as incorrect. Based on this correction, we defined several edit-distance-based features using Python's Python-Levenshtein extension.

Edit distance Assuming that nonnative speakers will make more spelling errors than natives, we used the average Levenshtein distance between the original word and the correction offered by the spell checker, for all words in a chunk, as a feature.

Spelling errors Again, we assume that the spelling errors that nonnatives make may reflect properties of their L1; this has already been shown for learners (Tsvetkov et al., 2013). Using the edit distance between a mis-spelled word w in a text chunk, marked by the spell checker, and its suggested correction c, we extract insertions, deletions and substitutions that yield c from w and use them as features. For each chunk, the value of this feature is the number of occurrences of each substitution (a character pair), insertions, and deletions in the chunk. We only used the top-400 most frequent substitutions.

We initially classified spelling errors as content-independent features, assuming that they would reflect transfer of linguistic phenomena from L1. However, having analyzed this feature type, we observed that many of the mis-spelled words turned out to be non-English words, which apparently are abundant in our dataset even after removing non-English sentences. We therefore view this feature as content dependent.

Grammar errors We hypothesize that grammar errors made by nonnatives may reflect grammatical structures revealing their L1. We therefore used LanguageTool, a rule-based grammar checker, to identify grammatical errors in the text.[4] We defined an indicator binary feature for each of the (over 2000) grammar rules detected by the grammar checker.[5]

3.5.3 Content-independent features

Content-based features may overly depend on the domain of the training data, and consequently be less effective when testing on different domains. Content-independent features are expected to be more robust when they are used out-of-domain.

Function words Function words are highly frequent and as such they are assumed to be selected unconsciously; they are therefore considered to reflect style, rather than content. Function words have been used successfully in a variety of style-based classification tasks (Mosteller and Wallace, 1963; Koppel and Ordan, 2011; Volansky et al., 2015; Rabinovich et al., 2016). We used as features (the frequencies of) about 400 function words, taken from Volansky et al. (2015).

POS tri-grams POS n-grams are assumed to reflect (shallow) grammar. The native language of the author is likely to influence the structure of his or her productions in English, and we assume that this will be reflected in this feature set. We used as features the normalized frequency of the top 300 most frequent POS tri-grams in the data set.[6]

Sentence length Texts of nonnative speakers are assumed to be simpler than those of natives; in particular, we expect them to have shorter sentences. The value of this feature is the average length of the sentences in the chunk.

3.5.4 Social network features

We defined several features that are extracted from the social network data, particularly its structure. First, we defined feature sets that express the *centrality* of users, under the assumption that native speakers would be more central on social networks. Consequently, this set of features is expected to be beneficial mainly for the binary native/nonnative classification.

User centrality in the social network of Reddit can be reflected in various ways:

Karma Reddit assigns a *karma* score to each user. This score "reflects how much good the user has done for the reddit community. The best way to gain karma is to submit links that other people like and vote for".[7] The Karma score is an undisclosed function of two separate scores: *link karma*, which is calculated from the user's posts that contain links, and *comment karma*, which is computed from the user's comments. We extracted both types of karma scores for all users in the dataset and used each of them (specifically, the user's monthly average scores) as a feature.

Average score Reddit calculates a *score* for each submission as the number of up-votes minus the number of down-votes the submission received. We used the user's average score per month as a feature.

Average number of submissions We counted for each user the total number of submissions he or she authored. For each chunk the value of this feature is the user's average number of submissions per month.

Average number of comments Same as the above, but counting user's comments (responses to submissions) instead of submissions.

Most popular subreddits Finally, we assume that native speakers of the same language tend to interact more with each other than with others, and we also assume that they are more likely to be interested in similar topics, influenced by their country and culture; specifically, we hypothesize that the forums in which users post most will be common for users from the same country. Therefore, we extracted for each country in the dataset the most popular subreddits among users from this country. For each country, we sorted subreddits according to the number of users from this country who posted at least once in this subreddit. The 30 most popular subreddits of each country

---

[4] We used the Python wrapper for LanguageTool.

[5] The list of English grammar rules is available online.

[6] We also experimented with POS 5-grams but they did not yield better results.

[7] The Reddit Wiki.

were taken as features. The unique list of popular subreddits contains 141 subreddits. For each chunk the value of a certain subreddit feature was a binary value indicating whether or not the author of this chunk has posted in this subreddit.

3.6 Evaluation

It is well known that similar classification tasks are highly domain-dependent; simply put, the properties of the domain overshadow the much more subtle signal of the author's L1. To test the robustness of various feature sets in the face of domain noise, we defined two evaluation scenarios: *in-domain*, where training and testing is done only on chunks from the European subreddits; and *out-of-domain*, where we train on chunks from the European subreddits and test on chunks from other subreddits, making sure they were authored by different users. Note that the out-of-domain corpus spans tens of thousands of subreddits with a huge number of topics. The precise evaluation scenario is somewhat involved and is detailed in Section A.2 of the Supplementary Materials. We report *accuracy*, defined as the percentage of chunks that were classified correctly out of the total number of chunks.

4 Results

We implemented the features discussed in Section 3.5 and evaluated the accuracy of the three classification tasks mentioned in Section 3.4 under the configurations described in Section 3.6. The trivial baseline for the binary classification task is 50%, for language family classification 25%, and for the language identification task 4.35%.

4.1 Individual feature sets

The accuracy results for each feature set described in Section 3.5 for the in-domain evaluation scenario are presented in Table 1.

Feature Set	Binary	Families	NLI
Char. 3-grams	85.58	78.20	62.06
Token unigrams	86.26	69.36	31.26
Spelling	71.04	52.18	27.74
Grammar errors	66.79	37.96	8.36
FW	80.34	57.80	20.15
POS 3-grams	69.14	50.29	13.30
Sentence length	50.37	26.14	4.79
Social network	57.92	32.39	5.75
Subreddits	87.08	82.56	74.46

Table 1: In-domain accuracy, individual feature sets

Evidently, all feature sets outperform the baseline, although some are far better than others. The feature that yields the best accuracy is *Subreddits*, with 87% accuracy on the binary task, 82% on the language family task and 74% on the NLI task. We elaborate on this feature in Section 4.2 below. As expected, the content based features yield relatively high results when the evaluation is in-domain. POS 3-grams and function words yield reasonable results, but not as good as in other classification setups (e.g., Rabinovich et al. (2016)), where the evaluation was done by shuffling texts of various users. As we evaluate on chunks of single users, the personal style of the user may dominate the subtler signal of his or her native language. Sentence length performs poorly, even on the binary task. Our assumption was that the *social network* feature set will work well only for the binary classification; this seems to be borne out by the results.

4.2 Feature combination

We now set out to investigate different feature combinations in both evaluation scenarios, aiming to define feature types that yield the best in-domain accuracy, as well as those that are most robust and generalize well out-of-domain.

Table 2 depicts the results obtained by combining character trigrams, tokens, and spelling features (Sections 3.5.1, 3.5.2). As expected, these content features yield excellent results in-domain, but the accuracy deteriorates out-of-domain, especially in the most challenging task of NLI.

	Binary	Families	NLI
In-domain	91.07	83.51	70.26
Out-of-domain	81.49	65.37	35.99

Table 2: Results: content features

The content-independent features (Section 3.5.3), whose contribution is depicted in Table 3, indeed fare worse, but are seemingly more robust outside the domain of training.

	Binary	Families	NLI
In-domain	81.89	62.40	22.38
Out-of-domain	74.56	52.35	14.86

Table 3: Results: content-independent features

Table 4 shows the results obtained by combining the spelling features with the grammar features

(Section 3.5.2). Clearly, these two feature types reflect somewhat different phenomena, as the results are better than using any of the two alone.

	Binary	Families	NLI
In-domain	72.93	55.59	26.74
Out-of-domain	70.24	47.23	14.15

Table 4: Results: grammar and spelling features

Table 5 shows the accuracy obtained by all the centrality features (Section 3.5.4), excluding the most popular subreddits. As expected, the contribution of these features is small, and is most evident on the binary task. The signal of the native language reflected by these features is very subtle, but is nonetheless present, as the results are consistently higher than the baseline.

	Binary	Families	NLI
In-domain	57.92	32.39	5.75
Out-of-domain	56.29	30.70	5.60

Table 5: Results: centrality features

Finally, the contribution of the most popular subreddits feature is shown in Table 6. The results for this single feature type are superb, both in- and out-of-domain. However, as this feature is unique to the dataset used for the present work, it is hard to see it generalized to similar tasks that use other datasets, even in the context of UGC.

	Binary	Families	NLI
In-domain	87.08	82.56	74.46
Out-of-domain	85.49	82.17	73.63

Table 6: Results: most popular subreddits

Therefore, we report the results obtained with *all* features, with (Table 7) and without (Table 8) the reddit-specific most popular subreddit feature.

	Binary	Families	NLI
In-domain	93.40	90.41	86.05
Out-of-domain	87.19	83.43	78.99

Table 7: Results: all features

	Binary	Families	NLI
In-domain	92.21	82.51	68.97
Out-of-domain	79.34	66.21	36.16

Table 8: Results: all features except subreddits

Summing up, we have shown that the challenging task of native language identification in the context of user generated content, where English texts are authored by highly competent nonnative speakers with as many as 23 native languages, can be accomplished with very high accuracy, as high as 86% when evaluated in-domain, and almost 79% out-of-domain (Table 7). While these results deteriorate when the specific characteristics of our dataset are not taken advantage of, we still obtain very high accuracy on the binary task of distinguishing native from nonnative speakers, and on the four-way task of identifying the language family of the authors' L1 (Table 8).

4.3 Dialect robustness

To assess the robustness of our results, especially in the context of dialect identification, we repeated the experiments in a special scenario: we trained classifiers on all the data, but removed from the English training set users from Ireland. Then, we tested the classifiers only on users from Ireland. We used all the features listed above, except the subreddit feature.

The results are 59.09% accuracy in-domain, compared with 69.21% in the standard scenario, where users from Ireland are also used for training; and 37.51% out-of-domain, compared with 47.85% in the standard scenario. In both cases, accuracy drops by 10 percent points. We conclude that our method is reasonably robust to dialectal variation, at least in the case of English varieties.

5 Analysis

We now set out to analyze some of the more interesting features, both in terms of their contribution to the accuracy of the classification and in terms of what they reveal about the English of advanced nonnative speakers.

5.1 Social network features

Subreddits This feature set works so well because many of the most popular subreddits in which users post are culturally revealing. Specifically, there is a significant presence in this list to (subreddits focusing on) specific countries. Very likely, most of the active users in those subreddits reside in these countries, thereby revealing their native language. This corroborates our hypothesis that native users of the same language tend to be active in mutual subreddits.

Network structure Table 9 lists the average values of the centrality features, comparing native vs. nonnative authors. The average values are higher for native users than for the nonnative ones in all of the centrality features, as we hypothesized. Evidently, native speakers are more central in social networks than nonnative ones.

	Native		nonnative	
	Avg	std	Avg	std
Score	1349	2383	906	1886
# comments	147	173	92	112
# submissions	5	21	4	13
Comment karma	787	1260	529	837
Link karma	202	1012	141	580

Table 9: Centrality features: average values and standard deviation

5.2 Spelling

Edit Distance As expected, the average word edit distance of native users (0.048) was significantly lower compared to nonnative ones (0.071).

Substitutions Most revealing was the analysis of substitutions suggested by the spell checker, as they shed light on phonetic and orthographic influences of the authors' L1 on their English. We list below some of the most common spelling errors.

Vowels Replacing *'e'* with *'a'* was twice as common among nonnative users than native ones. Examples include *'existance'*, *'independance'*, *'privillages'*, and *'apparantly'*. Similarly, replacing *'y'* with *'i'* was three times more common for nonnatives: *'synonims'*, *'analized'*, etc. Replacing *'o'* with *'a'* was common among nonnatives, especially in the context of diphthongs: *'enaugh'* instead of *'enough'*, or *'cauntry'* for *'country'*.

Voicing Replacing *'f'* with *'v'* was common mostly among German speakers: *'devense'*, *'bevore'*, *'sacrivice'*, etc. Another error that was relatively common in texts written by German speakers is the replacement of *'d'* with *'t'*: *'unterstand'*, *'canditate'*, *'upgradet'*, *'hundret'*, etc. Confusing *'z'* with *'s'* was very common across all L1s, even for natives. Among native users this reflects spelling variations between US and UK English. Thus, the spell-checker marks the following forms, i.a., in New Zealand English: *'Organisation'*, *'Recognise'*, *'Realise'*, *'Criticise'*, etc. Replacing *'s'* with *'z'* was not as common in the dataset, and was present mostly in texts of French users: *'advertize'*, *'tablez'*, and, most frequently, *'surprize'*.

Other substitutions Replacing *'c'* with *'k'* was almost four times more common with nonnatives; it was significantly more common among Germanic and Balto-Slavic speakers, and much less common among Romance speakers. Examples include *'inspektor'*, *'klassik'*, etc. Replacing *'t'* with *'c'* was common in words in which the *'t'* is pronounced [ʃ]: *'negociate'*, *'nacional'*. This error was prevalent in texts of Spanish authors.

Insertions and deletions Insertion of *'o'* was common for all nonnative speakers, often when the word contains one *'o'* but the pronunciation is [u], e.g., *'proove'* instead of 'prove'. Spurious occurrences of *'e'* were also very common among all nonnative users, especially authors whose L1 was French: *'governement'*, *'unemployement'*, *'explicitely'*. Deletions of *'e'* were also very common, especially in the context of words that end with *'ely'*: *'definitly'*, *'completly'*, *'extremly'*, *'absolutly'*, etc. Spurious instances of *'u'* were mostly present in texts of authors with Germanic and Romance L1s, e.g.: *'languague'*, *'percentuage'*.

Wrong insertions of *'l'* were very common, especially at the end of words that end with *'l'*: *'untill'*, *'controll'*, *'usefull'*. Deletion of *'l'* was common for all nonnative users, especially with Balto-Slavic L1s. The most common context for this error is words ending with *'ally'*: *'literaly'*, *'actualy'*, *'basicaly'*, *'illegaly'*, *'totaly'*, *'personaly'*, etc.

The most common deletion among nonnatives was omission of the first *'r'* in *'surprise'*, followed by omitting the first *'n'* in *'government'*.

5.3 Grammar

We list below some of the grammar rules whose violations distinguish well between native and nonnative speakers, using the original grammar checker rule names. Unsurprisingly, several grammar rules were violated much more (twice as frequently) by nonnative users:

adverb_word_order wrong position of adverb, e.g., *'people sometimes will respond'* instead of *'people will sometimes respond'*.

cd_nn agreement error of a numeral followed by a singular count noun, e.g., *'I have 5 book'*.

this_nns using *'this'* instead of *'these'* or vice versa, e.g., *'you don't know what these symbol* *represent'*.

did_baseform using a tensed verb after *'did'* or *'didn't'*: *'the court didn't gave him a fair trial'*.

a_uncountable an indefinite article before noncount nouns: *'smaller places have an access to...'*.

fewer_less confusing *'fewer'* with *'less'*: *'with less possibilities'*.

much_countable using *'much'* instead of *'many'*: *'no matter how much people'*. This error was much more common among nonnative users, although, among native speakers, it was significantly more common in texts written by users from New-Zealand and Ireland than in texts of other English speaking users.

en_a_vs_an confusing *'a'* with *'an'*: *'it provides a organized way to discuss'*. This error was very common among speakers of Germanic and Romance languages, but less common among speakers of Balto-slavic languages (presumably due to the lack of articles in their L1s).

In contrast, some grammar rules were violated more by native speakers:

possessive_apostrophe omitting the apostrophe in possessive *''s'*: *'they had 20% of the worlds remittance'*. This error was more than twice as common in texts of natives.

try_and the verb *'try'* followed by *'and'*; this is common in colloquial speech, but is prescriptively wrong: *'a candidate should try and represent'*. This rule was violated over three times more frequently by native speakers (but rarely in texts of New-Zealand users).

their_is *'there'* and *'their'* are commonly confused; this rule spots such cases by the presence of *'be'*: *'their are a lot of'*.

about_its_nn confusing *'its'* and *'it's'* is common; this rule identifies wrong usage after a preposition: *'lash out regularly towards it's neighbors'*. This error was most common in texts of English speakers from Australia, Ireland and the UK, but not the US.

Summing up, it seems that nonnative speakers make more grammatical errors, while the mistakes of native speakers either stem from sloppy writing style and lack of attention, or reflect style variations and casual style rather than actual errors.

6 Conclusion

We described a system that can accurately identify the native language of highly-advanced, fluent nonnative authors as reflected in the social media Reddit corpus. This is among the first studies to perform NLI in the highly challenging scenario of user generated content, particularly at such a large scale. We showed that while content-dependent features yield more accurate results, features that abstract away from content tend to be more robust when tested out of the domain of training. The in-depth analysis of spelling and grammar errors demonstrates that mistakes made by nonnative speakers reflect traces of their native language. We also illuminated some of the social characteristics of native and nonnative authors on social media outlets.

Our future plans include adaptation of the trained models to additional corpora, e.g., user generated content collected from *Facebook* and *Twitter*. Furthermore, we plan to devise *unsupervised* approaches to the identification of native language with the same dataset. We would also like to test the classifiers defined here in the more challenging scenario of smaller text chunks (e.g., 10–20 sentences rather than the 100-sentence text chunks we used here). Finally, we are currently experimenting with adversarial learning models for this task.

Acknowledgements

This research was supported by Grant No. 2017699 from the United States-Israel Binational Science Foundation (BSF) and by Grant No. 1813153 from the United States National Science Foundation (NSF).

References

Amir Afrasiabi Rad and Morad Benyoucef. 2011. Towards detecting influential users in social networks. In *E-Technologies: Transformation in a Connected World*, pages 227–240, Berlin, Heidelberg. Springer.

Kumar M Anand, Ganesh HB Barathi, Shivkaran Singh, KP Soman, and Paolo Rosso. 2017. Overview of the INLI PAN at FIRE-2017 track on Indian native language identification. Unpublished manuscript.

Ehud Alexander Avner, Noam Ordan, and Shuly Wintner. 2016. Identifying translationese at the word and sub-word level. *Digital Scholarship in the Humanities*, 31(1):30–54.

Marco Baroni and Silvia Bernardini. 2006. A new approach to the study of Translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing*, 21(3):259–274.

Shane Bergsma, Matt Post, and David Yarowsky. 2012. Stylometric analysis of scientific articles. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 327–337. Association for Computational Linguistics.

Yevgeni Berzak, Roi Reichart, and Boris Katz. 2015. Contrastive analysis with predictive power: Typology driven estimation of grammatical error distributions in ESL. In *Proceedings of the 19th Conference on Computational Natural Language Learning*, pages 94–102.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A corpus of non-native english. *ETS Research Report Series*, 2013(2):i–15.

Rumi Ghosh and Kristina Lerman. 2010. Predicting influential users in online social networks. In *Proceedings of KDD workshop on Social Network Analysis (SNA-KDD)*.

Sylviane Granger. 2003. The International Corpus of Learner English: a new resource for foreign language learning and teaching and second language acquisition research. *Tesol Quarterly*, pages 538–546.

David Jurgens, Tyler Finethy, James McCorriston, Yi Tian Xu, and Derek Ruths. 2015. Geolocation prediction in Twitter using social networks: A critical analysis and review of current practice. In *Proceedings of the 10th International AAAI Conference on Web and Social Media*, pages 188–197.

Ekaterina Kochmar. 2011. Identification of a writer's native language by error analysis. Master's thesis, University of Cambridge.

Moshe Koppel and Noam Ordan. 2011. Translationese and its dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1318–1326, Portland, Oregon, USA. Association for Computational Linguistics.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically determining an anonymous author's native language. *Intelligence and Security Informatics*, pages 41–76.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. *Automated Grammatical Error Detection for Language Learners*. Morgan and Claypool.

Shervin Malmasi and Mark Dras. 2017. Native language identification using stacked generalization. ArXiv:1703.06541 [cs.CL].

Shervin Malmasi, Keelan Evanini, Aoife Cahill, Joel Tetreault, Robert Pugh, Christopher Hamill, Diane Napolitano, and Yao Qian. 2017. A report on the 2017 native language identification shared task. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 62–75. Association for Computational Linguistics.

Shervin Malmasi and Marcos Zampieri. 2017. Arabic dialect identification using iVectors and ASR transcripts. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 178–183. Association for Computational Linguistics.

Frederick Mosteller and David L Wallace. 1963. Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed Federalist Papers. *Journal of the American Statistical Association*, 58(302):275–309.

Edward Newell, David Jurgens, Haji Mohammad Saleem, Hardik Vala, Jad Sassine, Caitrin Armstrong, and Derek Ruths. 2016. User migration in online social networks: A case study on reddit during a period of community unrest. In *Proceedings of the Tenth International AAAI Conference on Web and Social Media*, pages 279–288.

Ella Rabinovich, Sergiu Nisioi, Noam Ordan, and Shuly Wintner. 2016. On the similarities between native, non-native and translated texts. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, pages 1870–1881.

Ella Rabinovich, Noam Ordan, and Shuly Wintner. 2017. Found in translation: Reconstructing phylogenetic language trees from translations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 530–540. Association for Computational Linguistics.

Ella Rabinovich, Yulia Tsvetkov, and Shuly Wintner. 2018. Native language cognate effects on second language lexical choice. *Translactions of the Association for Computational Linguistics*, 6.

Ella Rabinovich and Shuly Wintner. 2015. Unsupervised identification of translationese. *Transactions of the Association for Computational Linguistics*, 3:419–432.

Michael Swan and Bernard Smith. 2001. *Learner English*, second edition. Cambridge University Press, Cambridge.

Kari Tenfjord, Paul Meurer, and Knut Hofland. 2006. The ASK corpus - a language learner corpus of Norwegian as a second language. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA).

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth*

Workshop on Building Educational Applications Using NLP. Association for Computational Linguistics.

Laura Mayfield Tomokiyo and Rosie Jones. 2001. You're not from 'round here, are you?: naive Bayes detection of non-native utterance text. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics*, pages 1–8. Association for Computational Linguistics.

Michael Trusov, Anand V. Bodapati, and Randolph E. Bucklin. 2010. Determining influential users in internet social networks. *Journal of Marketing Research*, 47(4):643–658.

Oren Tsur and Ari Rappoport. 2007. Using classifier features for studying the effect of native language on the choice of written second language words. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 9–16. Association for Computational Linguistics.

Yulia Tsvetkov, Naama Twitto, Nathan Schneider, Noam Ordan, Manaal Faruqui, Victor Chahuneau, Shuly Wintner, and Chris Dyer. 2013. Identifying the L1 of non-native writers: the CMU-Haifa system. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 279–287. Association for Computational Linguistics.

Vered Volansky, Noam Ordan, and Shuly Wintner. 2015. On the features of translationese. *Digital Scholarship in the Humanities*, 30(1):98–118.

Svitlana Volkova, Stephen Ranshous, and Lawrence Phillips. 2018. Predicting foreign language usage from English-only social media posts. In *Proceedings of NAACL-2018*.

Maolin Wang, Shervin Malmasi, and Mingxuan Huang. 2015. The Jinan Chinese learner corpus. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 118–123.

Sze-Meng Jojo Wong and Mark Dras. 2009. Contrastive analysis and native language identification. In *Proceedings of the Australasian Language Technology Association Workshop 2009*, pages 53–61, Sydney, Australia.

Sze-Meng Jojo Wong and Mark Dras. 2011. Exploiting parse structures for native language identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1600–1610, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, Jörg Tiedemann, Yves Scherrer, and Noëmi Aepli. 2017. Findings of the VarDial evaluation campaign 2017. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 1–15. Association for Computational Linguistics.

Beyond Error Propagation in Neural Machine Translation: Characteristics of Language Also Matter

Lijun Wu[1]*, **Xu Tan**[2]*, **Di He**[3], **Fei Tian**[2], **Tao Qin**[2], **Jianhuang Lai**[1] and **Tie-Yan Liu**[2]
[1]School of Data and Computer Science, Sun Yat-sen University
[2]Microsoft Research
[3]Key Laboratory of Machine Perception, MOE, School of EECS, Peking University
wulijun3@mail2.sysu.edu.cn; stsljh@mail.sysu.edu.cn;
{xuta,fetia,taoqin,tyliu}@microsoft.com; di_he@pku.edu.cn

Abstract

Neural machine translation usually adopts autoregressive models and suffers from exposure bias as well as the consequent error propagation problem. Many previous works have discussed the relationship between error propagation and the *accuracy drop* (i.e., the left part of the translated sentence is often better than its right part in left-to-right decoding models) problem. In this paper, we conduct a series of analyses to deeply understand this problem and get several interesting findings. (1) The role of error propagation on accuracy drop is overstated in the literature, although it indeed contributes to the accuracy drop problem. (2) Characteristics of a language play a more important role in causing the accuracy drop: the left part of the translation result in a right-branching language (e.g., English) is more likely to be more accurate than its right part, while the right part is more accurate for a left-branching language (e.g., Japanese). Our discoveries are confirmed on different model structures including Transformer and RNN, and in other sequence generation tasks such as text summarization.

1 Introduction

Neural machine translation (NMT) has attracted much research attention in recent years (Bahdanau et al., 2014; Shen et al., 2018; Song et al., 2018; Xia et al., 2018; He et al., 2016; Wu et al., 2017, 2018). The major approach to the task typically leverages an encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014) and the decoder usually generates the target tokens one by one from left to right autoregressively, in which the generation of a target token is conditioned on previously generated target tokens.

It has been observed that for an NMT model with left-to-right decoding, the right part words in its translation results are usually worse than the left part words in terms of accuracy (Zhang et al., 2018; Bengio et al., 2015; Ranzato et al., 2015; Hassan et al., 2018; Liu et al., 2016b,a). This phenomenon is referred to as *accuracy drop* in this paper. A straightforward explanation to accuracy drop is *error propagation*: If a word is mistakenly predicted during inference, the error will be propagated and the future words conditioned on this one will be impacted. Different methods have been proposed to address the problem of accuracy drop (Liu et al., 2016a,b; Hassan et al., 2018).

Instead of solving the problem, in this paper, we aim to deeply understand the causes of the problem. In particular, we want to answer the following two questions:

- Is error propagation the main cause of accuracy drop?
- Are there any other causes leading to accuracy drop?

To answer these two questions, we conduct a series of experiments to analyze the problem.

First, we train NMT models separately using left-to-right and right-to-left decoding (Sennrich et al., 2016; Liu et al., 2016b; He et al., 2017; Gao et al., 2018) on several language pairs (i.e., German to English, English to German, and English to Chinese). If error propagation is the main cause of accuracy drop, then the right part words in the translation results generated by right-to-left NMT models should be more accurate than the left part words. However, we observe the opposite phenomenon that the accuracy of the right part words of the translated sentences in both left-to-right and right-to-left models is lower than that of the left part, which contradicts with error propagation. This shows that error propagation alone cannot well explain the accuracy drop and even

---

[*] Authors contribute equally to this work.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3602–3611
Brussels, Belgium, October 31 - November 4, 2018.

suggests that error propagation may not exist or matter.

Second, to further investigate the influence of error propagation on accuracy drop, we conduct a set of experiments with teacher forcing (Williams and Zipser, 1989) during inference, in which we feed the ground-truth preceding words to predict the next target word. Teacher forcing eliminates exposure bias as well as error propagation in inference. The results verify the existence of error propagation, since the later part (the right part in left-to-right decoding and the left part in right-to-left decoding) of the translation results get more accuracy improvement with teacher forcing, regardless of the decoding direction. Meanwhile, the accuracy of the right part is still lower than that of the left part with teacher forcing, which demonstrates that there must be some other causes apart from error propagation leading to accuracy drop.

Third, inspired by linguistics, we find that the concept of branching (Berg et al., 2011; Payne, 2006) can help to explain the problem. We conduct the third set of experiments to study the correlation between language branching and accuracy drop. We find that if a target language is right branching such as English, the accuracy of the left part words is usually higher than that of the right part words, no matter for left-to-right or right-to-left NMT models, while for a left-branching target language such as Japanese, the accuracy of the left part words is usually lower than that of the right part, no matter for which models. The intuitive explanation is that a right-branching language has a clearer structure pattern (easier to predict) in the left part of sentence than that in the right part, since the main subject of the sentence is usually put in the left part. We calculate two statistics to verify this assumption: n-gram statistics (including n-gram frequency and conditional probabilities) and dependency parsing statistics. For right-branching languages, we found higher n-gram frequency/conditional probabilities as well as more dependencies in the left part compared with that in the right part. The opposite results are also found in left-branching languages.

We summarize our findings as follows.

- Through empirical analyses, we find that the influence of error propagation is overstated in the literature, which may misguide the future research. Error propagation alone cannot fully explain the accuracy drop in the left or right part of sentence.
- We find the branching in linguistics well correlates with accuracy drop in the left or right part of sentence and the corresponding analysis on n-gram and dependency parsing statistics well explain this phenomenon.

Our studies show that linguistics can be very helpful to understand existing machine learning models and build better models for language related tasks. We hope that our work can bring some insights to the research on neural machine translation. We believe that our findings can help us to design better translation models. For example, the finding on language branching suggests us to use left-to-right NMT models for right-branching languages such as English and right-to-right NMT models for left-branching languages such as Japanese.

2 Related Work

2.1 Exposure Bias and Error Propagation

Exposure bias and error propagation are two different concepts but often mentioned together in literature (Bengio et al., 2015; Shen et al., 2016; Ranzato et al., 2015; Liu et al., 2016b,a; Zhang et al., 2018; Hassan et al., 2018). Exposure bias refers to the fact that the sequence generation model is usually trained with teacher-forcing while generates the sequence autoaggressviely during inference. This discrepancy between training and inference can yield errors that accumulate quickly along the generated sequence, which is known as error propagation (Bengio et al., 2015; Shen et al., 2016; Ranzato et al., 2015).

Bengio et al. (2015) propose the scheduled sampling method to eliminate the exposure bias and the resulting error propagation, which achieves promising performance on sequence generation tasks such as image captioning. Shen et al. (2016); Ranzato et al. (2015) improve the basic maximum likelihood estimation (MLE) with reinforcement learning or minimum risk training and aim to address the limitation of MLE training and exposure bias problem.

2.2 Tackling Accuracy Drop

(Liu et al., 2016b,a; Zhang et al., 2018; Hassan et al., 2018) mainly ascribe accuracy drop (the accuracy of right part words is worse than that in the left part in most cases) to error propagation and propose different methods to solve this problem. Liu et al. (2016b,a); Hassan et al. (2018) use agreement regularization between the left-to-right and right-to-left models to achieve better performance. Zhang et al. (2018) and (Hassan et al., 2018) propose to use two-pass decoding to refine the generated sequence to yield better quality.

All these works focus on error propagation and accuracy drop. To our knowledge, there is no deep study about other causes of accuracy drop. In this paper, we aim to conduct such a study. Our study shows that accuracy drop is not only caused by error propagation, but also the characteristics of language itself.

3 Error Propagation and Accuracy Drop

3.1 Error Propagation is Not the Only Cause

A left-to-right NMT model feeds target tokens one by one from left to right in training and generate target tokens one by one from left to right during inference, while a right-to-left NMT model trains and generates token in the reverse direction. Intuitively, if error propagation is the root cause of accuracy drop, then a right-to-left NMT model will generate translations with better right half accuracy than the left half. In this section, we study the results of both left-to-right and right-to-left NMT models to analyze the relationship between error propagation and accuracy drop.

We conduct experiments on three translation tasks with different language pairs, which include: IWSLT 2014 German-English (De-En), WMT 2014 English-German (En-De) and WMT 2017 English-Chinese (En-Zh). We choose the state-of-the-art NMT model Transformer (Vaswani et al., 2017) as the basic model structure and train two separate models with left-to-right and right-to-left decoding on each language pair. More details about the datasets and model descriptions can be found in supplementary materials (section A.1 and A.2). We evenly split each generated sentence into the left half and the right half with same number of words[1]. Then for both the left and right half, we compute their accuracy with respect to the reference target sentence, in terms of BLEU score (Papineni et al., 2002) [2].

	De-En	En-De	En-Zh
left-to-right	31.42	26.93	20.79
right-to-left	30.00	25.35	20.23

Table 1: BLEU scores on the test set of the three translation tasks with both left-to-right and right-to-left decoding.

left-to-right	De-En	En-De	En-Zh
Left	**10.17**	**7.90**	**7.41**
Right	8.39	6.60	5.91
right-to-left	De-En	En-De	En-Zh
Right	7.83	6.45	5.77
Left	**9.41**	**7.11**	**7.01**

Table 2: BLEU scores of the left and right half of left-to-right and right-to-left NMT models. In (Liu et al., 2016a), the authors report the *partial BLEU* score without length penalty, our result is consistent with *partial BLEU* if simply removing length penalty when calculating BLEU.

We first report the BLEU scores of the full translation results (without split) in Table 1. As can be seen, the accuracy of the model is comparable to state-of-the-art results (Vaswani et al., 2017; Wang et al., 2017, 2018). Afterwards we report the BLEU scores of the left half and the right half in Table 2. We have several observations.

- When translating from left-to-right, the BLEU score of the left half is higher than the right half on all the three tasks, which is consistent with previous observation and is able to be explained via error propagation.
- When translating from right-to-left, the accuracy of the left half (in this way it's the later part of the generated sentence) is still higher than the right half. Such an observation is contradictory to the previous analyses between error propagation and accuracy drop, which regard that accumulated error brought by exposure bias will deteriorate the quality in later part of translation (i.e., the left half).

The inconsistent observation above suggests that error propagation is not the only cause of accuracy drop that there are other factors beyond er-

---

[1] 1) For most of the sentences, the last word of the sentence is period which is easy to decode. To make a fair comparison, we simply remove the last period before dividing the translation sentence. 2) For sentence with an odd number of words, we simply remove the word in the middle position to make the left half and right half have the same number of words.

[2] We use the *multi-bleu.perl* script `https://github.com/moses-smt/mosesdecoder/scripts/generic/multi-bleu.perl`. When computing BLEU score of the left or right half, the reference is the full reference sentence.

De-En	**left-to-right**			**right-to-left**		
	0	1	Δ	0	1	Δ
Left	**10.17**	**10.71**	0.54	**9.41**	**10.41**	1.00
Right	8.39	9.25	0.86	7.83	8.45	0.62

En-De	**left-to-right**			**right-to-left**		
	0	1	Δ	0	1	Δ
Left	**7.90**	**9.43**	1.53	**7.11**	**10.71**	3.60
Right	6.60	8.36	1.76	6.45	8.37	1.92

En-Zh	**left-to-right**			**right-to-left**		
	0	1	Δ	0	1	Δ
Left	**7.41**	**9.11**	1.70	**7.01**	**9.83**	2.82
Right	5.91	8.55	2.64	5.77	7.54	1.77

Table 3: BLEU scores. ”0” represents the translation results without teacher forcing during inference, and ”1” represents the translation results with teacher forcing during inference. Δ represents the BLEU score improvement of teacher forcing over normal translation.

ror propagation for accuracy drop. It even challenges the existence of error propagation: does error propagation really exist? In the next section we try to answer this question through teacher forcing experiments.

3.2 The Influence of Error Propagation

Teacher forcing (Williams and Zipser, 1989) in sequence generation means that when training a sequence generation model, we feed the previous ground-truth tokens as inputs to predict the next target word. Here we apply teacher forcing in the inference phase of NMT: to generate the next word $\hat{y}_i$, we input the preceding ground-truth words $y_{<i}$ rather than the previously generated words $\hat{y}_{<i}$, which largely alleviates the effect of error propagation, since there will be no error propagated from the previously generated words.

Same as last section, we evaluate the quality of the left and right half of the translation results generated by both the left-to-right and right-to-left models. The results are summarized in Table 3. For comparison, we also include the BLEU scores of normal translation (without teacher forcing). We have several findings from Table 3 as follows:

- Exposure bias exists. The accuracy of both left and right half tokens in the normal translation is lower than that in teacher forcing, which feeds the ground-truth tokens as inputs. This demonstrates that feeding the previously generated tokens (which might be incorrect) in inference indeed hurts translation accuracy.
- Error propagation does exist. We find the error is accumulated along the sequential generation of the sentence. Taking En-Zh and the left-to-right NMT model as an example, the BLEU score improvement of the right half (the second half of the generation) of teacher forcing over normal translation is 2.64, which is much larger than the accuracy improvement of the left half (the first half of the generation): 1.70. Similarly, for En-Zh with the right-to-left NMT model, the BLEU score improvement of the left half (the second half of the generation) of teacher forcing over normal translation is 2.82, which is much larger than the accuracy improvement of the right half (the first half of the generation): 1.77.
- Other causes exist. Taking En-De translation with the left-to-right model as an example, the accuracy of the left half (9.43) is higher than that of the right half (8.36) when there is no error propagation with teacher forcing. Similar results can be found in other language pairs and models. This suggests that there must be some other causes leading to accuracy drop, which will be studied in the next section.

4 Language Branching Matters

Section 3.1 and 3.2 together show that error propagation has influence on but is not the only cause of accuracy drop. We hypothesize that the language itself, i.e., its characteristics, may explain the phenomenon of accuracy drop.

Watanabe and Sumita (2002) finds that left-to-right decoding performs better for Japanese-English translation while right-to-left decoding performs better for English-Japanese translation. We conduct the same analysis settings as in Section 3.1 and 3.2 on English-Japanese (En-Jp) translation dataset. More details about this dataset and model descriptions can be found in supplementary materials (section A.1 and A.2).

Table 4 shows the BLEU score on the En-Jp test set. It can be observed that regardless of decoding direction (i.e., from left-to-right or from right-to-left) and with or without teacher forcing, the accuracy of the right half is always higher than that in the left half. This observation on Japanese is

	left-to-right		**right-to-left**	
	0	1	0	1
left	7.90	9.91	7.45	8.95
right	**8.70**	**11.52**	**9.24**	**10.59**

Table 4: BLEU scores on En-Jp test set. ”0” represents the normal translation results, and ”1” represents the teacher-forcing translation results.

opposite to English, German and Chinese in Section 3.1 and 3.2, and motivates us to investigate the differences between these languages.

We find that a linguistics concept, the branching, can differentiate Japanese from other languages such as English/German. Branching refers to the shape of the parse trees that represent the structure of sentences (Berg et al., 2011; Payne, 2006). Usually, right-branching sentences are head-initial, which means the main subject of the sentence is described first, and is followed by a sequence of modifiers that provide additional information about the subject. On the contrary, left-branching sentences are head-final that putting such modifiers to the left of the sentence (Payne, 2006).

English is a typical right-branching language, while Japanese is almost fully left-branching (Wikipedia, 2018). The two languages demonstrate the opposite phenomenon of accuracy drop as shown in previous studies. When we say a language is typical left/right-branching, we mean most of the sentences in this language follows the left/right-branching structure. While being predominantly right-branching, German is less conclusively so than English. Chinese features a mixture of head-final and head-initial structures, with the noun phrases are head-final while the strict head/complement ordering sentences are head-initial as right-branching (Wikipedia, 2018), but less conclusively than German.

We believe the language branching is a main cause of accuracy drop. Intuitively, the main subject of a right-branching sentence is described first (in the left part) and is followed by additional modifiers (in the right part) (Berg et al., 2011). Therefore, the left half of a right-branching sentence is more likely to possess a clearer structure pattern and thus lead to higher generation accuracy than in the right part, since the main subject is usually simpler and clearer than the modifiers that providing additional information about the subject. In next section, we will verify this intuition this assumption from a statistical perspective.

5 Correlation between Language Branching and Accuracy Drop

As previous work (Arpit et al., 2017) shows, neural networks are easy to learn and memorize simple patterns but difficult to make a correct prediction on noise examples. In this section, we study different branching languages from two aspects, including the n-gram statistics of a target language, which has been used as a kind of characterization of hardness of learning (Bengio et al., 2009), and the dependency statistics in parse trees. We show that these statistics well correlate with the accuracy drop between the left half and the right half of translation results.

5.1 N-gram Statistics

Intuitively speaking, if a pattern occurs frequently and deterministically, it is easy to be learned by neural networks. By comparing the general statistics on the n-gram frequency and n-gram conditional probability of the left and right half tokens, we link the language branching to accuracy drop.

Denote a bilingual dataset $D = \{(x_i, y_i\}, i = 1, \cdots, M$, where each y_i is a sequence of words $y_i = \{y_i^1, \cdots, y_i^{T_i}\}$, T_i is the length of y_i. $F_{i,n}^l$ and $P_{i,n}^l$ denote the average n-gram frequency and n-gram conditional probability of the left half of y_i [3], i.e.,

$$F_{i,n}^l = \frac{1}{T_i/2 - n + 1} \sum_{j=1}^{T_i/2-n+1} F(y_i^j, ..., y_i^{j+n-1}),$$

$$P_{i,n}^l = \frac{1}{T_i/2 - n + 1} \sum_{j=1}^{T_i/2-n+1} P(y_i^{j+n-1} | y_i^j, ..., y_i^{j+n-2}), \tag{1}$$

where $F(.)$ and $P(.)$ are the n-gram frequency and n-gram conditional probability calculated from the training dataset. Similarly, $F_{i,n}^r$ and $P_{i,n}^r$ denote the n-gram frequency and n-gram conditional probability of the right half.

We calculate the average n-gram frequencies F_n^l and F_n^r of the left half and right half over all the target sentences in the training set. We also calculate the average n-gram conditional probabilities P_n^l and P_n^r over all the training sentences to compare the uncertainty of phrases in the left half and

---

[3] Again, we assume T_i is an even number. If not, we simply remove the middle word of y_i, as done in Section 3.1.

right half.

$$
\begin{aligned}
F_n^l &= \frac{1}{M}\sum_{i=1}^{M} F_{i,n}^l,\ F_n^r = \frac{1}{M}\sum_{i=1}^{M} F_{i,n}^r, \\
P_n^l &= \frac{1}{M}\sum_{i=1}^{M} P_{i,n}^l,\ P_n^r = \frac{1}{M}\sum_{i=1}^{M} P_{i,n}^r.
\end{aligned}
\tag{2}
$$

We also calculate the ratio of the sentences that the frequency/conditional probability of left half is bigger/smaller than that in the right half, denoted as $RF_n^{l>r}$/$RF_n^{l<r}$ and $RP_n^{l>r}$/$RP_n^{l<r}$:

$$
\begin{aligned}
RF_n^{l>r} &= \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}\{F_{i,n}^l > F_{i,n}^r\}, \\
RF_n^{l<r} &= \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}\{F_{i,n}^l < F_{i,n}^r\}, \\
RP_n^{l>r} &= \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}\{P_{i,n}^l > P_{i,n}^r\}, \\
RP_n^{l<r} &= \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}\{P_{i,n}^l < P_{i,n}^r\}.
\end{aligned}
\tag{3}
$$

We choose $n = 2$ and 3 to calculate the metrics in Equation 2 and 3 on different translation datasets. The numbers are listed in Table 5 and 6.

We can see the 2/3-gram frequency as well as the conditional probability of the left half is higher than that of the right half for right-branching languages including English, German and Chinese in De-En, En-De and En-Zh translation datasets. For left-branching language Japanese, the result is opposite. The n-gram frequency and conditional probability statistics are consistent with our observations on accuracy drop in left/right-branching languages and verify our hypothesis: right-branching languages have clearer patterns in left part (with larger n-gram frequency as well as the conditional probability) and consequently leads to higher translation accuracy in the left part than the right part; left-branching languages are opposite.

We further visualize how the accuracy drop (between the left half and right half of the translations) correlates with the gap of n-gram statistics in the left and right part. The accuracy drop (e.g., BLEU score) of left/right half is taken from the teacher-forcing with left-to-right decoding in Table 3, and the n-gram gap is taken from the Δ in the last row of Table 5 and 6. Figure 1 shows strong correlation between accuracy drop and the gap of n-gram statistics: As the gap of n-gram statistics increases from negative values to positive values, the accuracy drop also increases from negative to positive.

	De-En		**En-De**	
	2-gram	**3-gram**	**2-gram**	**3-gram**
F_n^l	**5713.8**	**3122.7**	**13811.8**	**687.1**
F_n^r	3026.5	1377.6	11692.2	419.9
$RF_n^{l>r}$	**59.6%**	**55.8%**	**53.8%**	**53.6%**
$RF_n^{l<r}$	38.8%	37.6%	46.0%	45.0%
Δ	20.8%	18.2%	7.8%	8.6%
	En-Zh		**En-Jp**	
	2-gram	**3-gram**	**2-gram**	**3-gram**
F_n^l	**17707.0**	**1954.1**	18910.0	1350.0
F_n^r	16256.4	1250.5	**21076.7**	**1754.0**
$RF_n^{l>r}$	**51.9%**	**50.2%**	41.2%	38.0%
$RF_n^{l<r}$	46.7%	43.9%	**51.7%**	**52.3%**
Δ	5.2%	6.3%	-10.5%	-14.3%

Table 5: The n-gram frequency statistics on different translation datasets. F_n^l and F_n^r represent the average of n-gram frequency of left and right half of target sentences. $RF_n^{l>r}$ and $RF_n^{l<r}$ represent the ratio that the n-gram frequency of left half of sentences are bigger/smaller than that of the right half. $\Delta = RF_n^{l>r} - RF_n^{l<r}$. Note that the sum of $RF_n^{l>r}$ and $RF_n^{l<r}$ is less than 1 since sentence with less than 4 words does not contribute to the n-gram statistics.

	De-En		**En-De**	
	2-gram	**3-gram**	**2-gram**	**3-gram**
P_n^l	**0.137**	**0.181**	**0.082**	**0.155**
P_n^r	0.092	0.116	0.080	0.148
$RP_n^{l>r}$	**59.8%**	**56.6%**	**50.6%**	**51.7%**
$RP_n^{l<r}$	38.7%	36.4%	49.2%	47.0%
Δ	21.2%	20.2%	1.4%	4.7%
	En-Zh		**En-Jp**	
	2-gram	**3-gram**	**2-gram**	**3-gram**
P_n^l	**0.064**	**0.113**	0.082	0.171
P_n^r	0.055	0.108	**0.086**	**0.191**
$RP_n^{l>r}$	**52.1%**	**47.8%**	43.9%	39.4%
$RP_n^{l<r}$	46.6%	47.0%	**49.2%**	**50.9%**
Δ	5.5%	0.8%	-5.3%	-11.5%

Table 6: The n-gram conditional probability statistics on different translation datasets. P_n^l and P_n^r represent the average n-gram conditional probability of left and right half of target sentences. $RP_n^{l>r}$ and $RP_n^{l<r}$ represent the ratio that the n-gram frequency of left half are bigger/smaller than that of the right half. $\Delta = RP_n^{l>r} - RP_n^{l<r}$. Note that the sum of $RP_n^{l>r}$ and $RP_n^{l<r}$ is less than 1 due to two reasons: (1) sentence with less than 4 words does not contribute to the statistics, and (2) we remove the n-gram condition probability with the denominator less than 100 to make probability calculation robust.

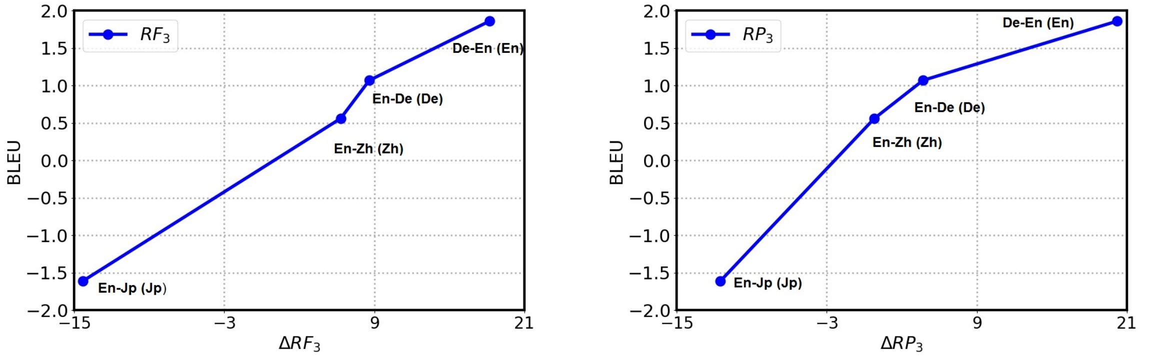

(a) Accuracy drop v.s 3-gram frequency gap (%).

(b) Accuracy drop v.s 3-gram conditional probability gap (%).

Figure 1: Accuracy drop (the gap between the left/right BLEU score) with respect to the ΔRF_3 and ΔRP_3 from Table 5 and 6 in the four translation tasks. The x-axis ΔRF_3 and ΔRP_3 represent the gap of between the left and right ratio of the 3-gram frequency/conditional probability defined in Table 5 and 6. The y-axis represents the accuracy drop in terms of BLEU score calculated by the teacher forcing decoding.

5.2 Dependency Statistics

In this subsection, we study language branching from the perspective of dependency structure. We hypothesize that if the left/right half of sentence contains more dependencies between its intra words, this half should be easier to predict, leading to higher accuracy. Here we analyze the English sentence in De-En translation and Japanese sentence in En-Jp translation, since English is fully right-branching and Japanese is fully left-branching as introduced before.

For English parsing, we utilize the well-acknowledged Standford Parser[4] to parse the sentences. After obtaining the parsing results, we split the sentence into left and right half, and separately count the numbers of dependencies in each half[5]. For Japanese, we leverage the open-source toolkit J.DepP[6] to parse the sentence, and then count the number of dependencies of each half.

We provide the results in Table 7. As can be observed, for English sentences, the left-half words depend more on each other than the right-half words, while for the Japanese sentences, the right-half words have more dependencies. This observation is consistent with our observations on accuracy drop, and can well explain the high accuracy of left part in English translation and right part in Japanese translation.

	English	Japanese
Left	**40242**	921735
Right	31509	**1570630**

Table 7: Number of dependencies in left and right half of English (De-En) and Japanese (En-Jp) training corpus. The number varies a lot since the two training corpus have different training sentences.

6 Extended Analyses and Discussions

We have analyzed the accuracy drop problem from the view of error propagation and language itself in previous sections. In this section, we further provide extended analyses and several discussions to give a more clear understanding of the accuracy drop problem.

6.1 More Languages on Left-Branching

The previous analyses are based on four languages, three right-branching (En, De, Zh) and one left-branching language (Jp). To avoid the experimental bias and randomness, we provide one more translation task, English-Turkish (En-Tr) translation[7], as Turkish is a left-branching language. We simply calculate the BLEU score of the left/right half in left-to-right and right-to-left

---

[4] https://nlp.stanford.edu/software/lex-parser.shtml

[5] For simplicity, we just count the number of dependency, without considering dependency types. The detailed parsing formats can be found in the supplementary material (Section A.3).

[6] http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/jdepp/

[7] The detailed dataset and model description can be found in supplementary material (section A.1 and section A.2).

	0	1
Left	**5.83**	7.44
Right	5.27	**7.96**

Table 8: BLEU scores on En-Tr test set with left-to-right generation. Normal translation is denoted as “0”, and teacher-forcing translation is denoted as “1”.

	left-to-right	**right-to-left**
Full	27.63	25.44
Left	**9.17**	**8.37**
Right	7.51	7.25

Table 9: BLEU scores on the left-to-right and right-to-left translation sentences on the De-En test set, with RNN-based model. “Full” means the BLEU score of the whole translation sentence.

decodings, as in Section 3.1 and 3.2.

The result is provided in Table 8. For the left-to-right decoding, the accuracy of the left half is higher than that of the right half in the normal translation. However, the accuracy of the right half becomes higher with teacher forcing translation. This demonstrates that English-Turkish translation performs similar to English-Japanese translation as the accuracy of right half is higher than that of the left half. But different from what we observed in Japanese, Turkish shows the opposite phenomenon: the influence of language branching is weaker than error propagation.

6.2 Other Model Structures

One may wonder whether the results in the paper are biased towards a certain model structure as we use Transformer on all the above analyses. To address such concerns, we conduct an additional experiment on De-En translation task with RNN (GRU)-based model[8]. The results are shown in Table 9 and the observations are consistent with what we observed on Transformer. The accuracy of the left half of the De-En translation sentence is always higher than the right half, in both the left-to-right and right-to-left decodings.

6.3 Other Sequence Generation Tasks

We conduct experimental analysis on abstractive summarization, which is also a sequence generation task. The goal of the task is to recap a long news sentence into a short summary. We use Gigaword dataset which contains $3.8M$ training pairs, $190k$ validation and $2k$ test pairs of English sentence, and train an RNN-based model for sentence summarization. The accuracy is measured by the commonly used metric ROUGE F1 score and are reported in Table 10.

	left-to-right		
	ROUGE-1	**ROUGE-2**	**ROUGE-L**
Full	35.55	16.66	33.01
Left	**24.44**	**9.87**	**23.34**
Right	21.31	8.32	20.38
	right-to-left		
	ROUGE-1	**ROUGE-2**	**ROUGE-L**
Full	35.22	16.55	32.59
Right	21.62	8.41	20.48
Left	**23.60**	**9.54**	**22.52**

Table 10: ROUGE F1 scores for left-to-right and right-to-left generated translation sentences in abstractive summarization task. ROUGE-N stands for N-gram based ROUGE F1 score, ROUGE-L stands for longest common subsequence based ROUGE F1 score. “Full” means the entire translation sentence.

We observe the same phenomenon as in translation tasks. The accuracy of the left half is always better than the right half, no matter in left-to-right or right-to-left decoding, since the target language English is a right-branching language.

7 Conclusion

In this work, we studied the problem of accuracy drop between the left half and the right half of the results generated by neural machine translation models. We found the influence of error propagation is overstated in literature and error propagation alone cannot explain accuracy drop. We showed that language branching well correlates to the accuracy drop problem and the evidences on n-gram statistics as well as the dependency statistics well support this correlation. Our discoveries suggest that left-to-right NMT models fit better for right-branching languages (e.g., English) and right-to-left NMT models fit better for left-branching languages (e.g., Japanese).

For future works, we will study more left/right-branching languages as well as other languages that have no obvious branching characteristics. We will also investigate how language branching influences other natural language tasks, especially for neural networks based models.

---

[8] The detailed setting for GRU based RNN model can be found in supplementary material (section A.2).

References

Devansh Arpit, Stanislaw K. Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 233–242. PMLR.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1171–1179.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Thomas Berg et al. 2011. *Structure in language: A dynamic perspective*. Routledge.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Fei Gao, Lijun Wu, Li Zhao, Tao Qin, Xueqi Cheng, and Tie-Yan Liu. 2018. Efficient sequence learning with group recurrent networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 799–808.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567.

Di He, Hanqing Lu, Yingce Xia, Tao Qin, Liwei Wang, and Tieyan Liu. 2017. Decoding with value networks for neural machine translation. In *Advances in Neural Information Processing Systems*, pages 178–187.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Lemao Liu, Andrew M. Finch, Masao Utiyama, and Eiichiro Sumita. 2016a. Agreement on target-bidirectional lstms for sequence-to-sequence learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2630–2637.

Lemao Liu, Masao Utiyama, Andrew M. Finch, and Eiichiro Sumita. 2016b. Agreement on target-bidirectional neural machine translation. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 411–416.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Thomas Payne. 2006. *Exploring language structure: a student's guide*. Cambridge University Press.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*, pages 371–376.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Yanyao Shen, Xu Tan, Di He, Tao Qin, and Tie-Yan Liu. 2018. Dense information flow for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1294–1303.

Kaitao Song, Xu Tan, Di He, Jianfeng Lu, Tao Qin, and Tie-Yan Liu. 2018. Double path networks for sequence to sequence learning. In *Proceedings of* *the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3064–3074.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Yijun Wang, Yingce Xia, Li Zhao, Jiang Bian, Tao Qin, Guiquan Liu, and Liu. 2018. Dual transfer learning for neural machine translation with marginal distribution regularization. In *AAAI*.

Yuguang Wang, Shanbo Cheng, Liyang Jiang, Jiajun Yang, Wei Chen, Muze Li, Lin Shi, Yanfeng Wang, and Hongtao Yang. 2017. Sogou neural machine translation systems for WMT17. In *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 410–415.

Taro Watanabe and Eiichiro Sumita. 2002. Bidirectional decoding for statistical machine translation. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*.

Wikipedia. 2018. Head-directionality parameter — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Head-directionality%20parameter&oldid=835385416`. [Online; accessed 08-May-2018].

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.

Yingce Xia, Xu Tan, Fei Tian, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2018. Model-level dual learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 5379–5388.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. *CoRR*, abs/1801.05122.

A Study of Reinforcement Learning for Neural Machine Translation

Lijun Wu[1]*, **Fei Tian**[2], **Tao Qin**[2], **Jianhuang Lai**[1] and **Tie-Yan Liu**[2]
[1]School of Data and Computer Science, Sun Yat-sen University
[2]Microsoft Research
wulijun3@mail2.sysu.edu.cn; stsljh@mail.sysu.edu.cn;
{fetia,taoqin,tyliu}@microsoft.com

Abstract

Recent studies have shown that reinforcement learning (RL) is an effective approach for improving the performance of neural machine translation (NMT) system. However, due to its instability, successfully RL training is challenging, especially in real-world systems where deep models and large datasets are leveraged. In this paper, taking several large-scale translation tasks as testbeds, we conduct a systematic study on how to train better NMT models using reinforcement learning. We provide a comprehensive comparison of several important factors (e.g., baseline reward, reward shaping) in RL training. Furthermore, to fill in the gap that it remains unclear whether RL is still beneficial when monolingual data is used, we propose a new method to leverage RL to further boost the performance of NMT systems trained with source/target monolingual data. By integrating all our findings, we obtain competitive results on WMT14 English- German, WMT17 English-Chinese, and WMT17 Chinese-English translation tasks, especially setting a state-of-the-art performance on WMT17 Chinese-English translation task.

1 Introduction

Recently, neural machine translation (NMT) (Bahdanau et al., 2015; Hassan et al., 2018; Wu et al., 2016; He et al., 2017; Xia et al., 2016, 2017; Wu et al., 2018b,a) has become more and more popular given its superior performance without the demand of heavily hand-crafted engineering efforts. It is usually trained to maximize the likelihood of each token in the target sentence, by taking the source sentence and the preceding (ground-truth) target tokens as inputs. Such training approach is referred as maximum likelihood estimation (MLE) (Scholz, 1985). Although easy to implement, the token-level objective function during training is inconsistent with sequence-level evaluation metrics such as BLEU (Papineni et al., 2002).

To address the inconsistency issue, reinforcement learning (RL) methods have been adopted to optimize sequence-level objectives. For example, policy optimization methods such as REINFORCE (Ranzato et al., 2016; Wu et al., 2017b) and actor-critic (Bahdanau et al., 2017) are leveraged for sequence generation tasks including NMT. In machine translation community, a similar method is proposed with the name 'minimum risk training' (Shen et al., 2016). All these works demonstrate the effectiveness of RL techniques for NMT models (Wu et al., 2016).

However, effectively applying RL to real-world NMT systems has not been fulfilled by previous works. First, most of, if not all, previous works verified their methods based on shallow recurrent neural network (RNN) models. However, to obtain state-of-the-art (SOTA) performance, it is essential to leverage recently derived deep models (Gehring et al., 2017; Vaswani et al., 2017), which are much more powerful.

Second, it is not easy to make RL practically effective given quite a few widely acknowledged limitations of RL method (Henderson et al., 2018) such as high variance of gradient estimation (Weaver and Tao, 2001), and objective instability (Mnih et al., 2013). Therefore, several tricks are proposed in previous works. However, it remains unclear, and no agreement is achieved on how to use these tricks in machine translation. For example, baseline reward method (Weaver and Tao, 2001) is suggested in (Ranzato et al., 2016; Nguyen et al., 2017; Wu et al., 2016) but not leveraged in (He and Deng, 2012; Shen et al., 2016).

Third, large-scale datasets, especially monolingual datasets are shown to significantly improve translation quality (Sennrich et al., 2015a; Xia et al.,

---

*This work was conducted at Microsoft Research Asia.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3612–3621
Brussels, Belgium, October 31 - November 4, 2018.

2016) with MLE training, while it remains nearly empty on how to combine RL with monolingual data in NMT.

In this paper, we try to fulfill these gaps and study how to practically apply RL to obtain strong NMT systems with quite competitive, even state-of-the-art performance. Several comprehensive studies are conducted on different aspects of RL training to figure out how to: 1) set efficient rewards; 2) combine MLE and RL objectives with different weights, which aims to stabilize the training procedure; 3) reduce the variance of gradient estimation.

In addition, given the effectiveness of leveraging monolingual data in improving translation quality, we further propose a new method to combine the strength of both RL training and source/target monolingual data. To the best of our knowledge, this is the first work that tries to explore the power of monolingual data when training NMT model with RL method.

We obtain some useful findings through the experiments on WMT17 Chinese-English (Zh-En), WMT17 English-Chinese (En-Zh) and WMT14 English-German (En-De) translation tasks. For instance, multinomial sampling is better than beam search in reward computation, and the combination of RL and monolingual data significantly enhances the NMT model performance. Our main contributions are summarized as follows.

- We provide the first comprehensive study on different aspects of RL training, such as how to setup reward and baseline reward, on top of quite competitive NMT models.
- We propose a new method that effectively leverages large-scale monolingual data, from both the source and target side, when training NMT models with RL.
- Combined with several of our findings and method, we obtain the SOTA translation quality on WMT17 Zh-En translation task, surpassing strong baseline (Transformer big model + back translation) by nearly 1.5 BLEU points. Furthermore, on WMT14 En-De and WMT17 En-Zh translation tasks, we can also obtain strong competitive results.

We hope that our studies and findings will benefit the community to better understand and leverage reinforcement learning for developing strong NMT models, especially in real-world scenarios faced with deep models and large amount of training data (including both parallel and monolingual data). Towards this end, we open source all our codes/dataset at `https://github.com/apeterswu/RL4NMT` to provide a clear recipe for performance reproduction.

2 Background

In this section, we first introduce the attention-based sequence-to-sequence learning framework for neural machine translation (NMT), and then introduce the basis of applying reinforcement learning to training NMT models.

2.1 Neural Machine Translation

Typical NMT models are based on the encoder-decoder framework with attention mechanism. The encoder first maps a source sentence $x = (x_1, x_2, ..., x_n)$ to a set of continuous representations $z = (z_1, z_2, ..., z_n)$. Given z, the decoder then generates a target sentence $y = (y_1, y_2, ..., y_m)$ of word tokens one by one. At each decoding step t of model training, the probability of generating a token y_t is maximized conditioned on x and $y_{<t} = (y_1, ..., y_{t-1})$. Given N training sentence pairs $\{x^i, y^i\}_{i=1}^N$, maximum likelihood estimation (MLE) is usually adopted to optimize the model, and the training objective is defined as:

$$\begin{aligned} L_{mle} &= \sum_{i=1}^{N} \log p(y^i|x^i) \\ &= \sum_{i=1}^{N} \sum_{t=1}^{m} \log p(y_t^i|y_1^i, ..., y_{t-1}^i, x^i), \end{aligned} \tag{1}$$

where m is the length of sentence y^i.

Among all the encoder-decoder models, the recently proposed Transformer (Vaswani et al., 2017) architecture achieves the best translation quality so far. The main difference between Transformer and previous RNNSearch (Bahdanau et al., 2015) or ConvS2S (Gehring et al., 2017) is that Transformer relies entirely on self-attention (Lin et al., 2017) to compute representations of source and target side sentences, without using recurrent or convolutional operations.

2.2 Training NMT with Reinforcement Learning

As aforementioned, reinforcement learning (RL) is leveraged to bridge the gap between training and inference of NMT, by directly optimizing the evaluation measure (e.g., BLEU) at training time. Specifically, NMT model can be viewed as an *agent*, which interacts with the *environment* (the previous words $y_{<t}$ and the context vector z available at each step t). The parameters of the agent define a *policy*, i.e., a conditional probability $p(y_t|x, y_{<t})$. The agent will pick an *action* , i.e., a candidate word out from the vocabulary, according to the policy. A terminal *reward* is observed once the agent generates a complete sequence $\hat{y}$. The reward for machine translation is the BLEU (Papineni et al., 2002) score, denoted as $R(\hat{y}, y)$, which is defined by comparing the generated $\hat{y}$ with the ground-truth sentence y. Note that here the reward $R(\hat{y}, y)$ is the sentence-level reward, i.e., a scalar for each complete sentence $\hat{y}$. The goal of the RL training is to maximize the expected reward:

$$\begin{aligned} L_{rl} &= \sum_{i=1}^{N} E_{\hat{y} \sim p(\hat{y}|x^i)} R(\hat{y}, y^i) \\ &= \sum_{i=1}^{N} \sum_{\hat{y} \in Y} p(\hat{y}|x^i) R(\hat{y}, y^i), \end{aligned} \tag{2}$$

where Y is the space of all candidate translation sentences, which is exponentially large due to the large vocabulary size, making it impossible to exactly maximize L_{rl}. In practice, REINFORCE (Williams, 1992) is usually leveraged to approximate the above expectation via sampling $\hat{y}$ from the policy $p(y|x)$, leading to the objective as maximizing:

$$\hat{L}_{rl} = \sum_{i=1}^{N} R(\hat{y}^i, y^i), \hat{y}^i \sim p(y|x^i), \forall i \in [N]. \tag{3}$$

Throughout the paper we will use REINFORCE as our policy optimization method for RL training.

3 Strategies for RL Training

Although training NMT with RL can fill in the gap between training objectives and evaluation metrics, it is not easy to successfully put RL training into practice. A key challenge is that RL methods are highly unstable and inefficient, due to the noise in gradient estimation and reward computation. To our best knowledge, currently there is no consensus, or even a systematic study on how to configure different setups for RL training to avoid such problems, especially for training deep NMT models on large scale datasets. We therefore aim to shed light on practical applications of RL for NMT training. For this purpose, we provide a comprehensive review of several important methods to stabilize RL training process in this section.

3.1 Reward Computation

It is critical to set up appropriate rewards for RL training, i.e., the $R(\hat{y}, y)$ in Eqn. (3). There are two important aspects to consider in configuring the reward $R(\hat{y}, y)$: how to sample training instance $\hat{y}$ and whether to use reward shaping.

Generate $\hat{y}$ There are two strategies to sample $\hat{y}$ for computing the BLEU reward $R(\hat{y}, y)$. The first one is *beam search* (Sutskever et al., 2014), it is a breadth-first search method that maintains a "beam" of the top-K scoring candidates (prefix hypothesis sentences) at each generation step. Then, for each candidate sentence in the beam, K most likely words are appended, resulting in a pool of $K \times K$ new candidates. Out from this pool, the top-K translations with largest probabilities are selected, and the beam search process continues. The second strategy is *multinomial sampling* (Chatterjee and Cancedda, 2010), which produces each word one by one through multinomial sampling over the model's output distribution. Both sampling strategies terminate the expansion of a candidate sentence when an 'end of sentence' (<EOS>) token is met.

The choice of different sampling strategies reflects the *exploration-exploitation* dilemma. Beam search strategy generates more accurate $\hat{y}$ by exploiting the probabilistic space output via current NMT model, while multinomial sampling pays more attention to explore more diverse candidates.

Whether to Use Reward Shaping From Eqn. (3) we can see that for the entire sequence $\hat{y}$, there is only one terminal reward $R(\hat{y}, y)$ available for model training. Note that the agent needs to take tens of actions (with the number depending on the length of $\hat{y}$) to generate a complete sentence $\hat{y}$, but only one reward is available for all those actions. Consequently, RL training is inefficient due to the sparsity of rewards, and the model updates each token in the training sentence with the same reward value without distinction. Reward shaping (Ng et al., 1999) is a strategy to overcome this shortcoming. In reward shaping, intermediate reward at each decoding step t is imposed and denoted as $r_t(\hat{y}_t, y)$. Bahdanau et al. (2017) sets up the intermediate reward as $r_t(\hat{y}_t, y) = R(\hat{y}_{1...t}, y) - R(\hat{y}_{1...t-1}, y)$, where $R(\hat{y}_{1...t}, y)$ is defined as the BLEU score of $\hat{y}_{1\ldots t}$ with respect to y. Note that we have $R(\hat{y}, y) = \sum_{t=1}^{m} r_t(\hat{y}_t, y)$, where m is the length of $\hat{y}$. During RL training, the cumulative reward $\sum_{\tau=t}^{m} r_\tau(\hat{y}_\tau, y)$ is used to update the policy at time step t. It is verified that using the shaped reward r_t instead of awarding the whole score $R(\hat{y}, y)$ does not change the optimal policy (Ng et al., 1999).

3.2 Variance Reduction of Gradient Estimation

As mentioned before, the REINFORCE algorithm suffers from high variance in gradient estimation, mainly caused by using single sample $\hat{y}$ to estimate the expectation. To reduce the variance, Ranzato et al. (2016) subtracts an average reward from the returned reward at each time step t, and the actual reward used to update the policy is

$$R(\hat{y}, y) - \hat{r}_t, \tag{4}$$

where $\hat{r}_t$ is the estimated average reward at step t, named as *baseline reward* (Weaver and Tao, 2001). Together with reward shaping, the updated reward becomes $\sum_{\tau=t}^{m} r_\tau(\hat{y}_\tau, y) - \hat{r}_t$ at step t.

Intuitively speaking, a baseline reward $\hat{r}_t$ is established, which either encourages a word choice $\hat{y}_t$ if the induced reward R satisfies $R > \hat{r}_t$, or discourages it if $R < \hat{r}_t$. Here R is either the terminal reward $R(\hat{y}, y)$ or the cumulative reward $\sum_{\tau=t}^{m} r_\tau(\hat{y}_\tau, y)$. Such estimated baseline reward $\hat{r}_t$ is designed to decrease the high variance of the gradient estimator.

In practice, the baseline reward $\hat{r}_t$ can be obtained through different approaches. For example, one may sample multiple sentences and use the mean terminal reward for these sentences as baseline reward. In our work, we adopt the function learning approach, using simple network (e.g., multi-layer perceptron) to build the learning function, which is the same as used in (Ranzato et al., 2016; Bahdanau et al., 2017).

3.3 Combine MLE and RL Objectives

The last important strategy we would like to mention is the combination of MLE training objective with RL objective, which is assumed to further stabilize RL training process (Wu et al., 2016; Li et al., 2017; Wu et al., 2017a).

A simple way is to linearly combine the MLE (Eqn. (1)) and RL (Eqn. (3)) objectives as follows:

$$L_{com} = \alpha * L_{mle} + (1 - \alpha) * \hat{L}_{rl}, \tag{5}$$

where α is the hyperparamter controlling the trade-off between MLE and RL objectives. We will empirically evaluate how different values of α impact the final translation accuracy.

4 RL Training with Monolingual Data

Previous works typically conduct RL training with only bilingual data for NMT. Monolingual data has been proved to be able to significantly improve the performance of NMT systems (Sennrich et al., 2015a; Xia et al., 2016; Cheng et al., 2016). It remains an open problem whether it is possible to combine the benefits of RL training and monolingual data such that even more competitive results can be obtained. In this section we provide several solutions for combination and will study them in next section. Note that all the settings discussed in this section are semi-supervised learning, i.e., both bilingual and monolingual data are available.

4.1 With Source-Side Monolingual Data

We first provide a solution to RL training with source-side monolingual data. As shown in Eqn. (3), in RL training we need to calculate the reward signal $R(\hat{y}, y)$ for each generated sentence $\hat{y}$, and therefore the reference sentence y seems to be a must-have, which unfortunately is missing for source-side monolingual data.

We tackle this challenge via generating pseudo target reference y by bootstrapping with the model itself. Apparently, for the source-side monolingual data, the pseudo target reference y should have good translation quality. Therefore, for each source-side monolingual sentence, we use the NMT model trained from the bilingual data to beam search a target sentence and treat it as the pseudo target reference y. Afterwards $\hat{y}$ is obtained via multinomial sampling to calculate the reward. Although multinomial sampling is usually not as good as sampling via beam search, the combination of beam search (to get the pseudo target reference sentence) and the multinomial sampling (to generate the action sequence of the agent) achieves good exploration-exploitation trade-off, since the pseudo target reference exploits the accuracy of current NMT model while $\hat{y}$ achieves better exploration.

4.2 With Target-Side Monolingual Data

For a target-side monolingual sentence, its source sentence x is missing, and consequently $\hat{y}$ is unavailable since it is sampled based on x. We tackle this challenge via back translation (Sennrich et al., 2015a). We first train a reverse NMT model from the target language to the source language with bilingual data. For each target-side monolingual sentence, using the reverse NMT model, we back translate it to get its pseudo source sentence x. We then pair the target monolingual data and its back-translated sentence as a pseudo bilingual sentence pair, which can be used for RL training in the same way as the genuine bilingual sentence pairs.

4.3 With both Source-Side and Target-Side Monolingual Data

A natural extension of previous discussions is to combine both the source-side and target-side monolingual data for RL training. We consider two combinations, the *sequential* method and the *unified* method. The former one sequentially leverages the source-side and target-side monolingual data for RL training. Specifically, we first train an MLE model using the bilingual data and source-side (or target-side) monolingual data; based on this MLE model, we then use REINFORCE for training with target-side (or source-side) monolingual data. For *unified* approach, we pack the paired data out from three domains together: the genuine bilingual data, the source monolingual data with its pseudo target references (introduced in subsection 4.1), and the target monolingual data with its back-translated samples (introduced in subsection 4.2). Then we treat the combined data as normal bilingual data on which the NMT model is trained via MLE or RL principles. Our goal is to investigate the model performance with different training data and find the best recipe of how to use these data in RL training. More details are introduced in next section.

5 Experiments

In this section, we provide a systematic study on aforementioned RL training strategies and the solutions of leveraging monolingual data. The RL training strategies are evaluated on bilingual datasets from three translation tasks, WMT14 English-German (En-De), WMT17 English-Chinese (En-Zh) and WMT17 Chinese-English (Zh-En), and we further conduct the experiments to leverage monolingual data in WMT17 Zh-En translation.

5.1 Experimental Settings

For the bilingual datasets, WMT17 (Bojar et al., 2017) En-Zh [1] and WMT17 Zh-En use the same dataset, which contains about $24M$ sentences pairs, including CWMT Corpus 2017 and UN Parallel Corpus V1.0. The Jieba[2] segmenter is used to perform Chinese word segmentation. We use byte pair encoding (BPE) (Sennrich et al., 2015b) to preprocess the source and target sentences, forming source-side and target-side dictionary with $40,000$ and $37,000$ types, respectively. We use the *newsdev2017* as the dev set and *newstest2017* as the test set. For the WMT14 En-De dataset, it contains about $4.5M$ training pairs, *newstest2012* and *newstest2013* are concatenated as the dev set and *newstest2014* acts as test set. Same as (Vaswani et al., 2017), we also perform BPE to process the En-De dataset, the shared source-target vocabulary contains about $37,000$ tokens.

For the monolingual dataset on Zh-En translation task, similar to (Sennrich et al., 2017), the Chinese monolingual data comes from LDC Chinese Gigaword (4th edition) and the English monolingual data comes from News Crawl 2016 articles. After preprocessing (e.g., language detection and filtering sentences with more than 80 words), we keep $4M$ Chinese sentences and $7M$ English sentences.

We adopt the Transformer model with *transformer_big* setting as defined in (Vaswani et al., 2017) for Zh-En and En-Zh translations, which achieves SOTA translation quality in several other datasets. For En-De translation, we utilize the *transformer_base_v1* setting. These settings are exactly same as used in the original paper, except we set the *layer_prepostprocess_dropout* for Zh-En and En-Zh translation to be 0.05. The optimizer used for MLE training is Adam (Kingma and Ba, 2015) with initial learning rate is 0.1, and we follow the same learning rate schedule in (Vaswani et al., 2017). During training, roughly $4,096$ source tokens and $4,096$ target tokens are paired in one mini batch. Each model is trained using 8 NVIDIA Tesla M40 GPUs. For RL training, the model is initialized with parameters of the MLE model (trained with only bilingual data), and we continue training it with learning rate 0.0001. Same as (Bahdanau et al., 2017), to calculate the BLEU reward, we start all n-gram counts from 1 instead of 0 and

---

[1] `http://www.statmt.org/wmt17/translation-task.html`

[2] `https://github.com/fxsjy/jieba`

Training Strategy	En-De	En-Zh	Zh-En
MLE	27.02	34.12	24.29
RL (beam + terminal)	27.06	34.25	24.42
RL (multinomial + terminal)	**27.22**	**34.46**	**24.70**
RL (beam + shaping)	27.04	34.28	24.47
RL (multinomial + shaping)	**27.23**	**34.47**	**24.72**

Table 1: Results of different strategies for reward computation. ‘beam’ refers to ‘beam search and ‘multinomial’ to ‘multinomial sampling’. While generating $\hat{y}$ through beam search, we use width 4. ‘shaping’ refers to using reward shaping and ‘terminal’ refers not.

multiply the resulting score by the length of the target reference sentence. For inference, we use beam search with width 6. We run each setting for at least 5 times and report the averaged case sensitive BLEU scores[3] (Papineni et al., 2002) on test set. The test set BLEU is chosen via the best configuration based on the validation set.

5.2 Results of of RL Training Strategies

We first evaluate different strategies for RL training, based only on bilingual datasets from previously introduced three translation tasks.

Reward Computation As reviewed in subsection 3.1, for reward computation, we need to consider how to sample $\hat{y}$ and whether to use reward shaping.

The results are shown in Table 1, where “RL” stands for RL training with the REINFORCE algorithm. We also report the performance of the pre-trained NMT model with the MLE loss. From the table, an interesting finding is that $\hat{y}$ sampled via beam search strategy is worse than that by multinomial sampling, with a gap of roughly 0.2-0.3 BLEU points on the test set (with significant test score $\rho < 0.05$). We therefore conjecture that exploration is more important than exploitation in reward computing: multinomial sampling brings more data diversity to the training of NMT model, while sentences generated by beam search are usually very similar to each other. Furthermore, we find that there is no big difference between the leverage of reward shaping or terminal reward, with only slightly better performance of reward shaping. We therefore use multinomial sampling and reward shaping in later experiments.

Training Strategy	En-De	En-Zh	Zh-En
RL	27.23	34.47	24.72
RL (baseline function)	27.25	34.43	24.73

Table 2: Results of variance reduction of gradient estimation.

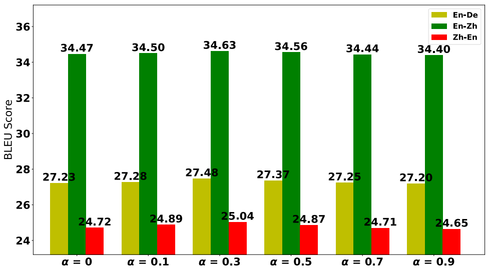

Figure 1: Results of different weights α to combine MLE and RL objectives.

Variance Reduction of Gradient Estimation Next we evaluate the strategies for reducing variance of gradient estimation (see section3.2). We want to know whether the *baseline reward* is necessary. To compute the baseline reward, similar to (Ranzato et al., 2016; Bahdanau et al., 2017), we build a two-layer MLP regressor with Relu (Nair and Hinton, 2010) activation units. The function takes the hidden states from decoder as input, and the parameters of the regressor are trained to minimize the mean squared loss of Eqn. (4). We first pre-train the baseline function for $20k$ steps/mini-batches, and then jointly train NMT model (with RL) and the baseline reward function.

Table 2 shows that the learning of baseline reward does not help RL training. This contradicts with previous observations (Ranzato et al., 2016), and seems to suggest that the variance of gradient estimation in NMT is not as large as we expected. The reason might be that the probability mass on the target-side language space induced by the NMT model is highly concentrated, making the sampled $\hat{y}$ representative enough in terms of estimating the expectation. Therefore, for the economic perspective, it is not necessary to add the additional steps of using baseline reward on RL training for NMT.

Combine MLE and RL Objectives As shown in Eqn. (5), the hyperparameter α controls the trade-off between MLE and RL objectives. For comparison, we set α to be [0, 0.1, 0.3, 0.5, 0.7, 0.9] in our experiments. The results are presented in Figure 1.

---

[3] Calculated by SacréBLEU toolkit, which produces exactly the same evaluation result as that in WMT17 Zh-En campaign. `https://github.com/awslabs/sockeye/tree/master/contrib/sacrebleu`

[Data] (Objective)	Valid	Test
[B] (MLE)	22.32	24.29
[B] (MLE) + [B] (RL)	22.87	25.04
[B] (MLE) + [Ms] (RL)	**23.03**	**25.22**
[B & Ms] (MLE)	24.31	25.31
[B & Ms] (MLE) + [B & Ms] (RL)	24.58	25.60

Table 3: Results with source monolingual data. “B” denotes bilingual data, “Ms” denotes source-side monolingual data, “&” denotes data combination.

The results show that combining the MLE objective with the RL objective achieves better performance (27.48 for En-De, 34.63 for En-Zh and 25.04 for Zh-En with $\alpha = 0.3$). This indicates that MLE objective is helpful to stabilize the training and improve the model performance, as we expected. However, further increasing α does not bring more gain. The best trade-off between MLE and RL objectives in our experiment is $\alpha = 0.3$. Therefore, we set $\alpha = 0.3$ in the following experiments.

5.3 Results of RL Training with Monolingual Data

In this subsection, we report the results on both valid and test set of RL training using bilingual and monolingual data in Zh-En translation. From Table 3 to Table 6, “RL” denotes the model trained with RL using multinomial sampling, reward shaping, no baseline reward, and combined objective, based on the observations in the last subsection. “B” denotes bilingual data, “Ms” denotes source-side monolingual data and “Mt” denotes target-side monolingual data, “&” denotes data combination.

With Source-Side Monolingual Data As discussed before, we use beam search with beam width 4 to sample the pseudo target sentence y for each monolingual sentence x. We consider several settings for RL training: 1) only source-side monolingual data; 2) the combination of bilingual and source-side monolingual data. We first train an MLE model using the augmented dataset combining the genuine bilingual data with the pseudo bilingual data generated from the monolingual data, and then perform RL training on this combined dataset. The results are shown in Table 3.

[Data] (Objective)	Valid	Test
[B] (MLE)	22.32	24.29
[B] (MLE) + [B] (RL)	22.87	25.04
[B] (MLE) + [Mt] (RL)	**22.96**	**25.15**
[B & Mt] (MLE)	24.14	25.24
[B & Mt] (MLE) + [B & Mt] (RL)	24.41	25.58

Table 4: Results with target monolingual data. “B” denotes bilingual data, “Mt” denotes target-side monolingual data, “&” denotes data combination.

[Data] (Objective)	Valid	Test
[B & Ms] (MLE)	24.31	25.31
[B & Ms] (MLE) + [B & Ms] (RL)	24.58	25.60
[B & Ms] (MLE) + [Mt] (RL)	**24.61**	**25.72**
[B & Mt] (MLE)	24.14	25.24
[B & Mt] (MLE) + [B & Mt] (RL)	24.41	25.58
[B & Mt] (MLE) + [Ms] (RL)	**24.75**	**25.92**

Table 5: Results of *sequential* approach for monolingual data. “B” denotes bilingual data, “Ms” denotes source-side monolingual data and “Mt” denotes target-side monolingual data, “&” denotes data combination.

[Data] (Objective)	Valid	Test
[B & Ms & Mt] (MLE)	25.58	26.13
+ [B & Ms & Mt] (RL)	**25.90**	**26.73**

Table 6: Results of *unified* approach for monolingual data. “+” means to initialize the RL model using above MLE model, which is trained on the combination of bilingual data, source-side monolingual data and target-side monolingual data.

With Target-Side Monolingual Data For target-side monolingual data, we first pre-train a translation model from English to Chinese [4], and use it to back translate target-side monolingual sentence y to get pseudo source sentence x. Similarly, we consider several settings for RL training: 1) only target-side monolingual data; 2) the combination of bilingual data and target-side monolingual data. We train an MLE model using both the genuine and the generated pseudo bilingual data, and then perform RL training on this data. The results are presented in Table 4.

From Table 3 and 4, we have several observations. First, monolingual data helps RL training, improving BLEU score from 25.04 to 25.22 ($\rho <$ 0.05) in Table 3. Second, when we only add monolingual data for RL training, the model achieves similar performance compared to MLE training with bilingual and monolingual data (e.g., 25.15 vs. 25.24 ($\rho <$ 0.05) in Table 4).

With both Source-Side and Target-Side Monolingual Data We have two approaches to use both source-side and target-side monolingual data, as described in subsection 4.3. The results are reported in Table 5 and Table 6.

From Table 5, we can observe that the *sequen-*

---

[4] The BLEU score of the En-Zh model is 34.12.

System	Architecture	BLEU
	Existing end-to-end NMT systems	
Vaswani et al. (2017)	Transformer	24.29
Sennrich et al. (2015a)	Transformer + Target Monolingual Data (i.e., back translation)	25.24
SougouKnowing	Stacked LSTM model + Reranking	24.00
SougouKnowing-ensemble	Stacked LSTM model + Reranking + Ensemble	26.40
	Our end-to-end NMT	
this work	Transformer + *RL*	25.04
	Transformer + Source Monolingual Data	25.31
	Transformer + Source Monolingual Data + *RL*	25.60
	Transformer + Target Monolingual Data	25.24
	Transformer + Target Monolingual Data + *RL*	25.58
	Transformer + Source & Target Monolingual Data	26.13
	Transformer + Source & Target Monolingual Data + *RL*	**26.73**

Table 7: Comparisons of different competitive end-to-end NMT systems. SougouKnowing results come from `http://matrix.statmt.org/matrix/systems_list/1878`.

tial training of monolingual data can benefit the model performance. Taking the last three rows as an example, the BLEU score of the MLE model trained on the combination of bilingual data and target-side monolingual data is 25.24; based on this model, RL training using the source-side monolingual data further improves the model performance by 0.7 ($\rho < 0.01$) BLEU points. From Table 6, we can observe on top of a quite strong MLE baseline (26.13), through the *unified* RL training, we can still improve the test set by 0.6 points to 26.73 ($\rho < 0.01$), which shows the effectiveness of combining source/target monolingual data and reinforcement learning.

5.4 Comparison with Other Models

At last, as a summary of our empirical results, we compare several representataive end-to-end NMT systems to our work in Table 7, which includes the Transformer (Vaswani et al., 2017) model, with/without back-translation (Sennrich et al., 2015a) and the best NMT system in WMT17 Chinese-English translation challenge[5] (SougouKnowing-ensemble). The results clearly show that after combing both source-side and target-side monolingual data with RL training, we obtain the state-of-the-art BLEU score 26.73, even surpassing the best ensemble model in WMT17 Zh-En translation challenge.

6 Related Work

Our work is mainly related with the literature of using reinforcement learning to directly optimize the evaluation measure for neural machine translation. Several representative works are (Ranzato et al., 2016; Shen et al., 2016; Bahdanau et al., 2017). In (Ranzato et al., 2016), the authors propose to train a neural translation model with the objective gradually shifting from maximizing token-level likelihood to optimizing the sentence-level BLEU score. Shen et al. (2016) proposes to adopt minimum risk training (Goel and Byrne, 2000) to minimize the task specific expected loss (i.e., induced by BLEU score) on NMT training data. Instead of the REINFORCE (Williams, 1992) algorithm used in the above two works, Bahdanau et al. (2017) further optimizes the policy by actor-critic algorithm. Wu et al. (2016) introduces a simple RL based method to optimize the stacked LSTM model for NMT, achieving better BLEU scores on English-French translation but not on English-German. Edunov et al. (2017) presents a comparative study of several classical structural prediction losses for NMT model, which also includes sequence-level loss but not exactly the same as RL.

Our work is also related with the research works that leverage monolingual data for improving NMT models (Zhang and Zong, 2016; Sennrich et al., 2015a; Wang et al., 2018; Xia et al., 2016; Cheng et al., 2016). Zhang and Zong (2016) exploits the source-side monolingual data in NMT. Sennrich et al. (2015a) proposes back-translation method to leverage target-side monolingual data for NMT. Xia et al. (2016) formulates the machine translation as a communication game, which leverages the power of two directional translation models and source/target monolingual data. Cheng et al. (2016) proposes a similar semi-supervised approach. However, none of these works have explored the power of monolingual data in the context of training NMT model with reinforcement learning.

---

[5] `http://matrix.statmt.org/matrix/systems_list/1878`

7 Conclusion

In this work, we presented a study of how to effectively train NMT models using reinforcement learning. Different RL strategies were evaluated in German-English, English-Chinese and Chinese-English translation tasks on large-scale bilingual datasets. We found that (1) multinomial sampling is better than beam search, (2) several previous tricks such as reward shaping and baseline reward does not make significant difference, and (3) the combination of the MLE and RL objectives is important. In addition, we explored the source/target monolingual data for RL training. By combing the power of RL and monolingual data, we achieve the state-of-the-art BLEU score on WMT17 Chinese-English translation task. We hope that our study and results can benefit the community and bring some insights on how to train deep NMT models with reinforcement learning and big data.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. *Fifth International Conference on Learning Representations*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Third International Conference on Learning Representations*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Samidh Chatterjee and Nicola Cancedda. 2010. Minimum error rate training by sampling the translation lattice. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 606–615. Association for Computational Linguistics.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. *meeting of the association for computational linguistics*, pages 1965–1974.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2017. Classical structured prediction losses for sequence to sequence learning. *Proceedings of NAACL-HLT 2018*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *Proceedings of the 34th international conference on machine learning*.

Vaibhava Goel and William J Byrne. 2000. Minimum bayes-risk automatic speech recognition. *Computer Speech & Language*, 14(2):115–135.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Di He, Hanqing Lu, Yingce Xia, Tao Qin, Liwei Wang, and Tieyan Liu. 2017. Decoding with value networks for neural machine translation. In *Advances in Neural Information Processing Systems*, pages 178–187.

Xiaodong He and Li Deng. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 292–301. Association for Computational Linguistics.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. *Thirthy-Second AAAI Conference On Artificial Intelligence*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Third International Conference on Learning Representations*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *Fifth International Conference on Learning Representations*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *Advances in neural information processing systems, workshop*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning*, pages 807–814.

Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. 2017. Reinforcement learning for bandit neural machine translation with simulated human feedback. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *Fourth International Conference on Learning Representations*.

FW Scholz. 1985. Maximum likelihood estimation. *Encyclopedia of statistical sciences*.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of edinburgh's neural mt systems for wmt17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. *meeting of the association for computational linguistics*, pages 1683–1692.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Yijun Wang, Yingce Xia, Li Zhao, Jiang Bian, Tao Qin, Guiquan Liu, and T Liu. 2018. Dual transfer learning for neural machine translation with marginal distribution regularization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 538–545. Morgan Kaufmann Publishers Inc.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Lijun Wu, Xu Tan, Di He, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018a. Beyond error propagation in neural machine translation: Characteristics of language also matter. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Lijun Wu, Fei Tian, Li Zhao, Jianhuang Lai, and Tie-Yan Liu. 2018b. Word attention for sequence to sequence text understanding. In *AAAI*.

Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017a. Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.

Lijun Wu, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017b. Sequence prediction with unlabeled data by reward function learning. *IJCAI-17*, pages 3098–3104.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Weiying Ma. 2016. Dual learning for machine translation. *neural information processing systems*, pages 820–828.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Advances in Neural Information Processing Systems*, pages 1784–1794.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

Meta-Learning for Low-Resource Neural Machine Translation

Jiatao Gu*[†], Yong Wang*[†], Yun Chen[†], Kyunghyun Cho[‡] and Victor O.K. Li[†]

[†]The University of Hong Kong
[‡]New York University, CIFAR Azrieli Global Scholar
[†]{jiataogu, wangyong, vli}@eee.hku.hk
[†]yun.chencreek@gmail.com
[‡]kyunghyun.cho@nyu.edu

Abstract

In this paper, we propose to extend the recently introduced model-agnostic meta-learning algorithm (MAML, Finn et al., 2017) for low-resource neural machine translation (NMT). We frame low-resource translation as a meta-learning problem, and we learn to adapt to low-resource languages based on multilingual high-resource language tasks. We use the universal lexical representation (Gu et al., 2018b) to overcome the input-output mismatch across different languages. We evaluate the proposed meta-learning strategy using eighteen European languages (Bg, Cs, Da, De, El, Es, Et, Fr, Hu, It, Lt, Nl, Pl, Pt, Sk, Sl, Sv and Ru) as source tasks and five diverse languages (Ro, Lv, Fi, Tr and Ko) as target tasks. We show that the proposed approach significantly outperforms the multilingual, transfer learning based approach (Zoph et al., 2016) and enables us to train a competitive NMT system with only a fraction of training examples. For instance, the proposed approach can achieve as high as 22.04 BLEU on Romanian-English WMT'16 by seeing only 16,000 translated words (~ 600 parallel sentences).

1 Introduction

Despite the massive success brought by neural machine translation (NMT, Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017), it has been noticed that the vanilla NMT often lags behind conventional machine translation systems, such as statistical phrase-based translation systems (PBMT, Koehn et al., 2003), for low-resource language pairs (see, e.g., Koehn and Knowles, 2017). In the past few years, various approaches have been proposed to address this issue. The first attempts at tackling this problem exploited the availability of monolingual corpora (Gulcehre et al., 2015; Sennrich et al., 2015; Zhang and Zong, 2016). It was later followed by approaches based on multilingual translation, in which the goal was to exploit knowledge from high-resource language pairs by training a single NMT system on a mix of high-resource and low-resource language pairs (Firat et al., 2016a,b; Lee et al., 2016; Johnson et al., 2016; Ha et al., 2016b). Its variant, transfer learning, was also proposed by Zoph et al. (2016), in which an NMT system is pretrained on a high-resource language pair before being finetuned on a target low-resource language pair.

In this paper, we follow up on these latest approaches based on multilingual NMT and propose a meta-learning algorithm for low-resource neural machine translation. We start by arguing that the recently proposed model-agnostic meta-learning algorithm (MAML, Finn et al., 2017) could be applied to low-resource machine translation by viewing language pairs as separate tasks. This view enables us to use MAML to find the initialization of model parameters that facilitate fast adaptation for a new language pair with a minimal amount of training examples (§3). Furthermore, the vanilla MAML however cannot handle tasks with mismatched input and output. We overcome this limitation by incorporating the universal lexical representation (Gu et al., 2018b) and adapting it for the meta-learning scenario (§3.3).

We extensively evaluate the effectiveness and generalizing ability of the proposed meta-learning algorithm on low-resource neural machine translation. We utilize 17 languages from Europarl and Russian from WMT as the source tasks and test the meta-learned parameter initialization against five target languages (Ro, Lv, Fi, Tr and Ko), in all cases translating to English. Our experiments using only up to 160k tokens in each of the target task reveal that the proposed meta-learning approach outperforms the multilingual translation

---

* Equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3622–3631
Brussels, Belgium, October 31 - November 4, 2018.

approach across all the target language pairs, and the gap grows as the number of training examples decreases.

2 Background

Neural Machine Translation (NMT) Given a source sentence $X = \{x_1, ..., x_{T'}\}$, a neural machine translation model factors the distribution over possible output sentences $Y = \{y_1, ..., y_T\}$ into a chain of conditional probabilities with a left-to-right causal structure:

$$p(Y|X;\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T'};\theta), \quad (1)$$

where special tokens y_0 ($\langle \mathrm{bos} \rangle$) and y_{T+1} ($\langle \mathrm{eos} \rangle$) are used to represent the beginning and the end of a target sentence. These conditional probabilities are parameterized using a neural network. Typically, an encoder-decoder architecture (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015) with a RNN-based decoder is used. More recently, architectures without any recurrent structures (Gehring et al., 2017; Vaswani et al., 2017) have been proposed and shown to speed up training while achieving state-of-the-art performance.

Low Resource Translation NMT is known to easily over-fit and result in an inferior performance when the training data is limited (Koehn and Knowles, 2017). In general, there are two ways for handling the problem of low resource translation: (1) utilizing the resource of unlabeled monolingual data, and (2) sharing the knowledge between low- and high-resource language pairs. Many research efforts have been spent on incorporating the monolingual corpora into machine translation, such as multi-task learning (Gulcehre et al., 2015; Zhang and Zong, 2016), back-translation (Sennrich et al., 2015), dual learning (He et al., 2016) and unsupervised machine translation with monolingual corpora only for both sides (Artetxe et al., 2017b; Lample et al., 2017; Yang et al., 2018).

For the second approach, prior researches have worked on methods to exploit the knowledge of auxiliary translations, or even auxiliary tasks. For instance, Cheng et al. (2016); Chen et al. (2017); Lee et al. (2017); Chen et al. (2018) investigate the use of a pivot to build a translation path between two languages even without any directed resource. The pivot can be a third language or even an image in multimodal domains. When pivots are not easy to obtain, Firat et al. (2016a); Lee et al. (2016); Johnson et al. (2016) have shown that the structure of NMT is suitable for multilingual machine translation. Gu et al. (2018b) also showed that such a multilingual NMT system could improve the performance of low resource translation by using a universal lexical representation to share embedding information across languages.

All the previous work for multilingual NMT assume the joint training of multiple high-resource languages naturally results in a universal space (for both the input representation and the model) which, however, is not necessarily true, especially for very low resource cases.

Meta Learning In the machine learning community, meta-learning, or learning-to-learn, has recently received interests. Meta-learning tries to solve the problem of "fast adaptation on new training data." One of the most successful applications of meta-learning has been on few-shot (or one-shot) learning (Lake et al., 2015), where a neural network is trained to readily learn to classify inputs based on only one or a few training examples. There are two categories of meta-learning:

1. learning a meta-policy for updating model parameters (see, e.g., Andrychowicz et al., 2016; Ha et al., 2016a; Mishra et al., 2017)

2. learning a good parameter initialization for fast adaptation (see, e.g., Finn et al., 2017; Vinyals et al., 2016; Snell et al., 2017).

In this paper, we propose to use a meta-learning algorithm for low-resource neural machine translation based on the second category. More specifically, we extend the idea of model-agnostic meta-learning (MAML, Finn et al., 2017) in the multilingual scenario.

3 Meta Learning for Low-Resource Neural Machine Translation

The underlying idea of MAML is to use a set of source tasks $\{\mathcal{T}^1, \ldots, \mathcal{T}^K\}$ to find the initialization of parameters θ^0 from which learning a target task $\mathcal{T}^0$ would require only a small number of training examples. In the context of machine translation, this amounts to using many high-resource language pairs to find good initial parameters and training a new translation model on a low-resource language starting from the found initial parame-

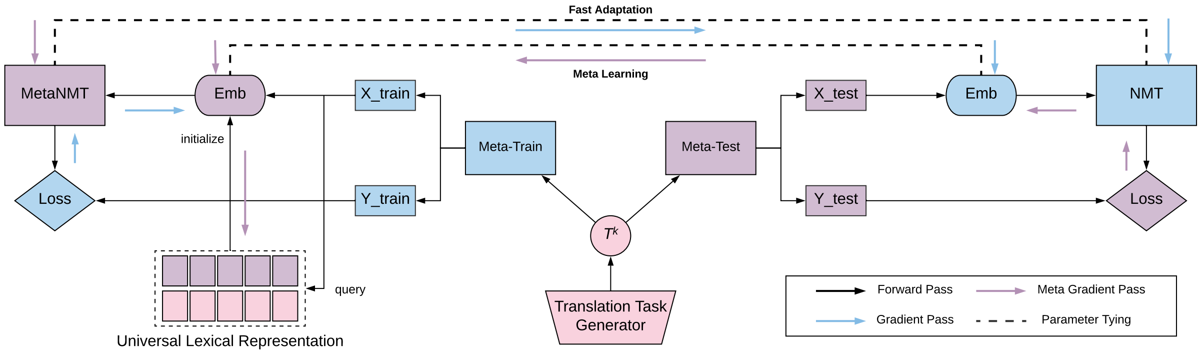

Figure 1: The graphical illustration of the training process of the proposed MetaNMT. For each episode, one task (language pair) is sampled for meta-learning. The boxes and arrows in blue are mainly involved in language-specific learning (§3.1), and those in purple in meta-learning (§3.2).

ters. This process can be understood as

$$\theta^* = \text{Learn}(\mathcal{T}^0; \text{MetaLearn}(\mathcal{T}^1, \ldots, \mathcal{T}^K)).$$

That is, we *meta-learn* the initialization from auxiliary tasks and continue to *learn* the target task. We refer the proposed meta-learning method for NMT to MetaNMT. See Fig. 1 for the overall illustration.

3.1 Learn: language-specific learning

Given any initial parameters θ^0 (which can be either random or meta-learned),

the prior distribution of the parameters of a desired NMT model can be defined as an isotropic Guassian:

$$\theta_i \sim \mathcal{N}(\theta_i^0, 1/\beta),$$

where $1/\beta$ is a variance. With this prior distribution, we formulate the language-specific learning process $\text{Learn}(D_{\mathcal{T}}; \theta^0)$ as maximizing the log-posterior of the model parameters given data $D_{\mathcal{T}}$:

$$\begin{aligned}\text{Learn}(D_{\mathcal{T}}; \theta^0) &= \arg\max_{\theta} \mathcal{L}^{D_{\mathcal{T}}}(\theta) \\ &= \arg\max_{\theta} \sum_{(X,Y)\in D_{\mathcal{T}}} \log p(Y|X,\theta) - \beta\|\theta - \theta^0\|^2,\end{aligned}$$

where we assume $p(X|\theta)$ to be uniform. The first term above corresponds to the maximum likelihood criterion often used for training a usual NMT system. The second term discourages the newly learned model from deviating too much from the initial parameters, alleviating the issue of overfitting when there is not enough training data. In practice, we solve the problem above by maximizing the first term with gradient-based optimization and early-stopping after only a few update steps. Thus, in the low-resource scenario, finding a good initialization θ^0 strongly correlates the final performance of the resulting model.

3.2 MetaLearn

We find the initialization θ^0 by repeatedly simulating low-resource translation scenarios using auxiliary, high-resource language pairs. Following Finn et al. (2017), we achieve this goal by defining the meta-objective function as

$$\begin{aligned}\mathcal{L}(\theta) =& \mathbb{E}_k \mathbb{E}_{D_{\mathcal{T}^k}, D'_{\mathcal{T}^k}} \\ &\left[\sum_{(X,Y)\in D'_{\mathcal{T}^k}} \log p(Y|X; \text{Learn}(D_{\mathcal{T}^k}; \theta)) \right],\end{aligned} \tag{2}$$

where $k \sim \mathcal{U}(\{1, \ldots, K\})$ refers to one meta-learning episode, and $D_{\mathcal{T}}$, $D'_{\mathcal{T}}$ follow the uniform distribution over $\mathcal{T}$'s data.

We maximize the meta-objective function using stochastic approximation (Robbins and Monro, 1951) with gradient descent. For each episode, we uniformly sample one source task at random, $\mathcal{T}^k$. We then sample two subsets of training examples independently from the chosen task, $D_{\mathcal{T}^k}$ and $D'_{\mathcal{T}^k}$. We use the former to *simulate* language-specific learning and the latter to *evaluate* its outcome. Assuming a single gradient step is taken only the with learning rate η, the simulation is:

$$\theta'_k = \text{Learn}(D_{\mathcal{T}^k}; \theta) = \theta - \eta \nabla_\theta \mathcal{L}^{D_{\mathcal{T}^k}}(\theta).$$

Once the simulation of learning is done, we evaluate the updated parameters θ'_k on $D'_{\mathcal{T}^k}$, The gradient computed from this evaluation, which we refer to as *meta-gradient*, is used to update the

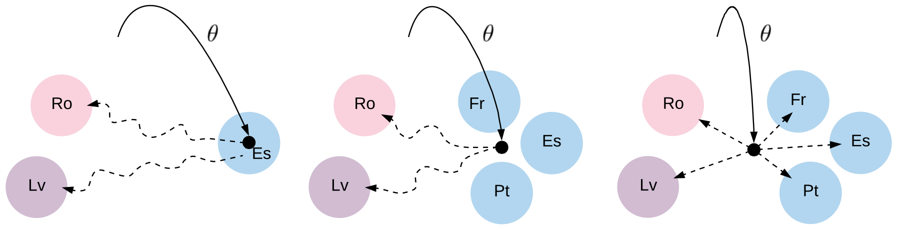

(a) Transfer Learning (b) Multilingual Transfer Learning (c) Meta Learning

Figure 2: An intuitive illustration in which we use solid lines to represent the learning of initialization, and dashed lines to show the path of fine-tuning.

meta model θ. It is possible to aggregate multiple episodes of source tasks before updating θ:

$$\theta \leftarrow \theta - \eta' \sum_k \nabla_\theta \mathcal{L}^{D'_{\mathcal{T}^k}}(\theta'_k),$$

where η' is the meta learning rate.

Unlike a usual learning scenario, the resulting model θ^0 from this meta-learning procedure is not necessarily a good model on its own. It is however a good starting point for training a good model using only a few steps of learning. In the context of machine translation, this procedure can be understood as finding the initialization of a neural machine translation system that could quickly adapt to a new language pair by simulating such a fast adaptation scenario using many high-resource language pairs.

Meta-Gradient We use the following approximation property

$$H(x)v \approx \frac{\nabla(x + \nu v) - \nabla(x)}{\nu}$$

to approximate the meta-gradient:[1]

$$\begin{aligned}
&\nabla_\theta \mathcal{L}^{D'}(\theta') = \nabla_{\theta'} \mathcal{L}^{D'}(\theta') \nabla_\theta (\theta - \eta \nabla_\theta \mathcal{L}^{D}(\theta)) \\
&= \nabla_{\theta'} \mathcal{L}^{D'}(\theta') - \eta \nabla_{\theta'} \mathcal{L}^{D'}(\theta') H_\theta(\mathcal{L}^{D}(\theta)) \\
&\approx \nabla_{\theta'} \mathcal{L}^{D'}(\theta') - \frac{\eta}{\nu} \left[\nabla_\theta \mathcal{L}^{D}(\theta) \Big|_{\hat{\theta}} - \nabla_\theta \mathcal{L}^{D}(\theta) \Big|_{\theta} \right],
\end{aligned}$$

where ν is a small constant and

$$\hat{\theta} = \theta + \nu \nabla_{\theta'} \mathcal{L}^{D'}(\theta').$$

In practice, we find that it is also possible to ignore the second-order term, ending up with the following simplified update rule:

$$\nabla_\theta \mathcal{L}^{D'}(\theta') \approx \nabla_{\theta'} \mathcal{L}^{D'}(\theta'). \tag{3}$$

Related Work: Multilingual Transfer Learning The proposed MetaNMT differs from the existing framework of multilingual translation (Lee et al., 2016; Johnson et al., 2016; Gu et al., 2018b) or transfer learning (Zoph et al., 2016). The latter can be thought of as solving the following problem:

$$\max_\theta \mathcal{L}^{\text{multi}}(\theta) = \mathbb{E}_k \left[\sum_{(X,Y) \in D_k} \log p(Y|X;\theta) \right],$$

where D_k is the training set of the k-th task, or language pair. The target low-resource language pair could either be a part of joint training or be trained separately starting from the solution θ^0 found from solving the above problem.

The major difference between the proposed MetaNMT and these multilingual transfer approaches is that the latter do not consider how learning happens with the target, low-resource language pair. The former explicitly incorporates the learning process within the framework by simulating it repeatedly in Eq. (2). As we will see later in the experiments, this results in a substantial gap in the final performance on the low-resource task.

Illustration In Fig. 2, we contrast transfer learning, multilingual learning and meta-learning using three source language pairs (Fr-En, Es-En and Pt-En) and two target pairs (Ro-En and Lv-En). Transfer learning trains an NMT system specifically for a source language pair (Es-En) and finetunes the system for each target language pair (Ro-En, Lv-En). Multilingual learning often trains a single NMT system that can handle many different language pairs (Fr-En, Pt-En, Es-En), which may or may not include the target pairs (Ro-En, Lv-En). If not, it finetunes the system for each target pair, similarly to transfer learning. Both of these however aim at directly solving the source tasks. On the other hand, meta-learning trains the NMT system to be *useful for fine-tuning* on various tasks including the source and target tasks. This is done by repeatedly simulating the learning process on

---

[1] We omit the subscript k for simplicity.

low-resource languages using many high-resource language pairs (Fr-En, Pt-En, Es-En).

3.3 Unified Lexical Representation

I/O mismatch across language pairs One major challenge that limits applying meta-learning for low resource machine translation is that the approach outlined above assumes the input and output spaces are shared across all the source and target tasks. This, however, does not apply to machine translation in general due to the vocabulary mismatch across different languages. In multilingual translation, this issue has been tackled by using a vocabulary of sub-words (Sennrich et al., 2015) or characters (Lee et al., 2016) shared across multiple languages. This surface-level sharing is however limited, as it cannot be applied to languages exhibiting distinct orthography (e.g., Indo-Euroepan languages vs. Korean.)

Universal Lexical Representation (ULR) We tackle this issue by dynamically building a vocabulary specific to each language using a key-value memory network (Miller et al., 2016; Gulcehre et al., 2018), as was done successfully for low-resource machine translation recently by Gu et al. (2018b). We start with multilingual word embedding matrices $\epsilon^k_{\text{query}} \in \mathbb{R}^{|V_k| \times d}$ pretrained on large monolingual corpora, where V_k is the vocabulary of the k-th language. These embedding vectors can be obtained with small dictionaries of seed word pairs (Artetxe et al., 2017a; Smith et al., 2017) or in a fully unsupervised manner (Zhang et al., 2017; Conneau et al., 2018). We take one of these languages k' to build universal lexical representation consisting of a universal embedding matrix $\epsilon_u \in \mathbb{R}^{M \times d}$ and a corresponding key matrix $\epsilon_{\text{key}} \in \mathbb{R}^{M \times d}$, where $M < |V'_k|$. Both $\epsilon^k_{\text{query}}$ and ϵ_{key} are fixed during meta-learning. We then compute the language-specific embedding of token x from the language k as the convex sum of the universal embedding vectors by

$$\epsilon^0[x] = \sum_{i=1}^{M} \alpha_i \epsilon_u[i],$$

where $\alpha_i \propto \exp\left\{-\frac{1}{\tau}\epsilon_{\text{key}}[i]^\top A \epsilon^k_{\text{query}}[x]\right\}$ and τ is set to 0.05. This approach allows us to handle languages with different vocabularies using a fixed number of shared parameters (ϵ_u, ϵ_{key} and A.)

Learning of ULR It is not desirable to update the universal embedding matrix ϵ_u when fine-tuning on a small corpus which contains a limited set of unique tokens in the target language, as it could adversely influence the other tokens' embedding vectors. We thus estimate the change to each embedding vector induced by language-specific learning by a separate parameter $\Delta\epsilon^k[x]$:

$$\epsilon^k[x] = \epsilon^0[x] + \Delta\epsilon^k[x].$$

During language-specific learning, the ULR $\epsilon^0[x]$ is held constant, while only $\Delta\epsilon^k[x]$ is updated, starting from an all-zero vector. On the other hand, we hold $\Delta\epsilon^k[x]$'s constant while updating ϵ_u and A during the meta-learning stage.

	# of sents.	# of En tokens	Dev	Test
Ro-En	0.61 M	16.66 M	–	31.76
Lv-En	4.46 M	67.24 M	20.24	15.15
Fi-En	2.63 M	64.50 M	17.38	20.20
Tr-En	0.21 M	5.58 M	15.45	13.74
Ko-En	0.09 M	2.33 M	6.88	5.97

Table 1: Statistics of full datasets of the target language pairs. BLEU scores on the dev and test sets are reported from a supervised Transformer model with the same architecture.

4 Experimental Settings

4.1 Dataset

Target Tasks We show the effectiveness of the proposed meta-learning method for low resource NMT with extremely limited training examples on five diverse target languages: Romanian (Ro) from WMT'16,[2] Latvian (Lv), Finnish (Fi), Turkish (Tr) from WMT'17,[3] and Korean (Ko) from Korean Parallel Dataset.[4] We use the officially provided train, dev and test splits for all these languages. The statistics of these languages are presented in Table 1. We simulate the low-resource translation scenarios by randomly sub-sampling the training set with different sizes.

Source Tasks We use the following languages from Europarl[5]: Bulgarian (Bg), Czech (Cs), Danish (Da), German (De), Greek (El), Spanish (Es), Estonian (Et), French (Fr), Hungarian (Hu), Italian (It), Lithuanian (Lt), Dutch (Nl), Polish (Pl), Portuguese (Pt), Slovak (Sk), Slovene (Sl) and

---

[2] http://www.statmt.org/wmt16/translation-task.html

[3] http://www.statmt.org/wmt17/translation-task.html

[4] https://sites.google.com/site/koreanparalleldata/

[5] http://www.statmt.org/europarl/

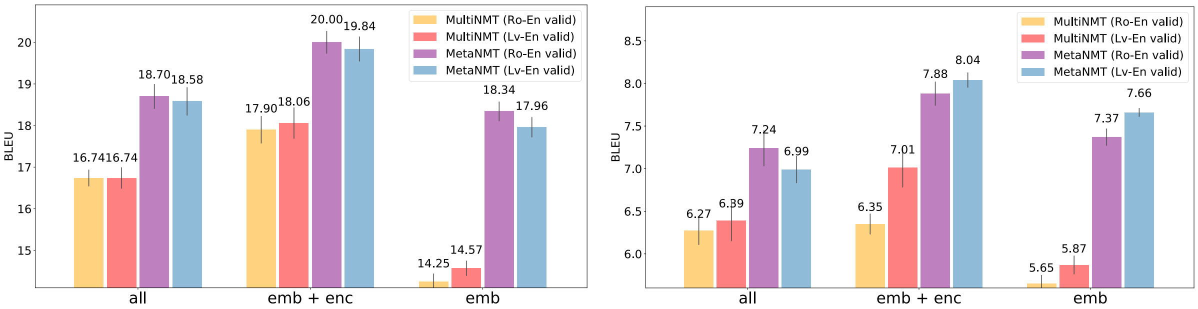

(a) Ro-En (b) Lv-En

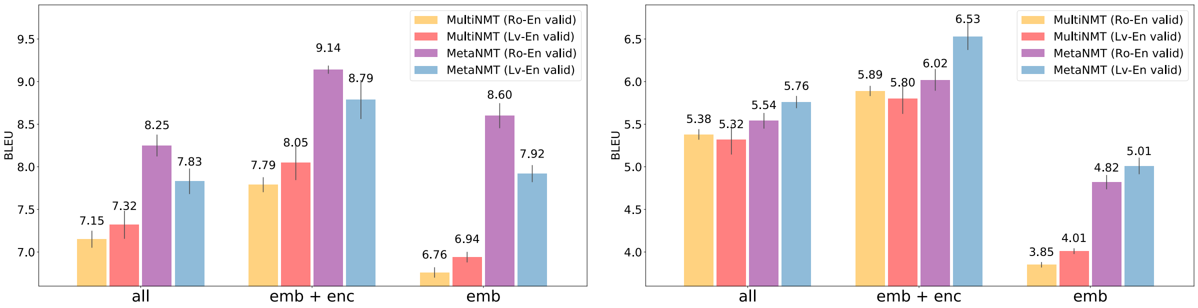

(c) Fi-En (d) Tr-En

Figure 3: BLEU scores reported on test sets for {Ro, Lv, Fi, Tr} to En, where each model is first learned from 6 source tasks (Es, Fr, It, Pt, De, Ru) and then fine-tuned on randomly sampled training sets with around 16,000 English tokens per run. The error bars show the standard deviation calculated from 5 runs.

Swedish (Sv), in addition to Russian (Ru)[6] to learn the intilization for fine-tuning. In our experiments, different combinations of source tasks are explored to see the effects from the source tasks.

Validation We pick either Ro-En or Lv-En as a validation set for meta-learning and test the generalization capability on the remaining target tasks. This allows us to study the strict form of meta-learning, in which target tasks are unknown during both training and model selection.

Preprocessing and ULR Initialization As described in §3.3, we initialize the query embedding vectors $\epsilon^k_{\text{query}}$ of all the languages. For each language, we use the monolingual corpora built from Wikipedia[7] and the parallel corpus. The concatenated corpus is first tokenized and segmented using byte-pair encoding (BPE, Sennrich et al., 2016), resulting in $40,000$ subwords for each language. We then estimate word vectors using fastText (Bojanowski et al., 2016) and align them across all the languages in an unsupervised way using MUSE (Conneau et al., 2018) to get multilingual word vectors. We use the multilingual word vectors of the 20,000 most frequent words in English to form the universal embedding matrix ϵ_u.

4.2 Model and Learning

Model We utilize the recently proposed Transformer (Vaswani et al., 2017) as an underlying NMT system. We implement Transformer in this paper based on (Gu et al., 2018a)[8] and modify it to use the universal lexical representation from §3.3. We use the default set of hyperparameters ($d_{\text{model}} = d_{\text{hidden}} = 512$, $n_{\text{layer}} = 6$, $n_{\text{head}} = 8$, $n_{\text{batch}} = 4000$, $t_{\text{warmup}} = 16000$) for all the language pairs and across all the experimental settings. We refer the readers to (Vaswani et al., 2017; Gu et al., 2018a) for the details of the model. However, since the proposed meta-learning method is model-agnostic, it can be easily extended to any other NMT architectures, e.g. RNN-based sequence-to-sequence models with attention (Bahdanau et al., 2015).

---

[6] A subsample of approximately 2M pairs from WMT'17.

[7] We use the most recent Wikipedia dump (2018.5) from https://dumps.wikimedia.org/backup-index.html.

[8] https://github.com/salesforce/nonauto-nmt

Meta-Train	Ro-En		Lv-En		Fi-En		Tr-En		Ko-En	
	zero	finetune	zero	finetune	zero	finetune	zero	finetune	zero	finetune
–		00.00 ± .00		0.00 ± .00		0.00 ± .00		0.00 ± .00		0.00 ± .00
Es	9.20	15.71 ± .22	2.23	4.65 ± .12	2.73	5.55 ± .08	1.56	4.14 ± .03	0.63	1.40 ± .09
Es Fr	12.35	17.46 ± .41	2.86	5.05 ± .04	3.71	6.08 ± .01	2.17	4.56 ± .20	0.61	1.70 ± .14
Es Fr It Pt	13.88	18.54 ± .19	3.88	5.63 ± .11	4.93	6.80 ± .04	2.49	4.82 ± .10	0.82	1.90 ± .07
De Ru	10.60	16.05 ± .31	5.15	7.19 ± .17	6.62	7.98 ± .22	3.20	6.02 ± .11	1.19	2.16 ± .09
Es Fr It Pt De Ru	15.93	20.00 ± .27	6.33	7.88 ± .14	7.89	9.14 ± .05	3.72	6.02 ± .13	1.28	2.44 ± .11
All	18.12	**22.04 ± .23**	9.58	**10.44 ± .17**	11.39	**12.63 ± .22**	5.34	**8.97 ± .08**	1.96	**3.97 ± .10**
Full Supervised	31.76		15.15		20.20		13.74		5.97	

Table 2: BLEU Scores w.r.t. the source task set for all five target tasks.

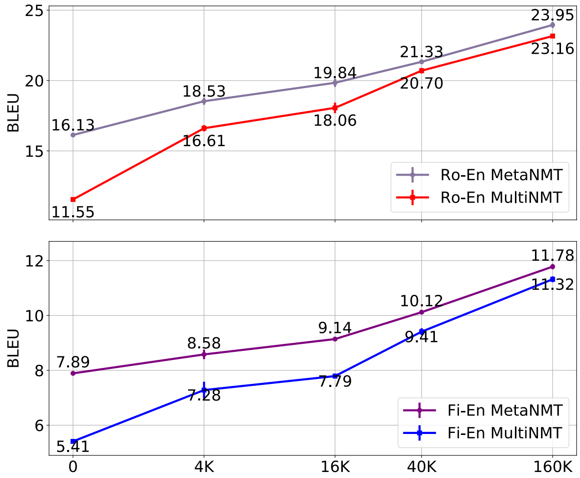

Figure 4: BLEU Scores w.r.t. the size of the target task's training set.

Learning We meta-learn using various sets of source languages to investigate the effect of source task choice. For each episode, by default, we use a single gradient step of language-specific learning with Adam (Kingma and Ba, 2014) per computing the meta-gradient, which is computed by the first-order approximation in Eq. (3).

For each target task, we sample training examples to form a low-resource task. We build tasks of 4k, 16k, 40k and 160k English tokens for each language. We randomly sample the training set five times for each experiment and report the average score and its standard deviation. Each fine-tuning is done on a training set, early-stopped on a validation set and evaluated on a test set. In default without notation, datasets of 16k tokens are used.

Fine-tuning Strategies The transformer consists of three modules; embedding, encoder and decoder. We update all three modules during meta-learning, but during fine-tuning, we can selectively tune only a subset of these modules. Following (Zoph et al., 2016), we consider three fine-tuning strategies; (1) fine-tuning all the modules (all), (2) fine-tuning the embedding and encoder, but freezing the parameters of the decoder (emb+enc) and (3) fine-tuning the embedding only (emb).

5 Results

vs. Multilingual Transfer Learning We meta-learn the initial models on all the source tasks using either Ro-En or Lv-En as a validation task. We also train the initial models to be multilingual translation systems. We fine-tune them using the four target tasks (Ro-En, Lv-En, Fi-En and Tr-En; 16k tokens each) and compare the proposed meta-learning strategy and the multilingual, transfer learning strategy. As presented in Fig. 3, the proposed learning approach significantly outperforms the multilingual, transfer learning strategy across all the target tasks regardless of which target task was used for early stopping. We also notice that the emb+enc strategy is most effective for both meta-learning and transfer learning approaches. With the proposed meta-learning and emb+enc fine-tuning, the final NMT systems trained using only a fraction of all available training examples achieve 2/3 (Ro-En) and 1/2 (Lv-En, Fi-En and Tr-En) of the BLEU score achieved by the models trained with full training sets.

vs. Statistical Machine Translation We also test the same Ro-En datasets with $16,000$ target tokens using the default setting of Phrase-based MT (Moses) with the dev set for adjusting the parameters and the test set for calculating the final performance. We obtain $4.79(\pm 0.234)$ BLEU point, which is higher than the standard NMT performance (0 BLEU). It is however still lower than both the multi-NMT and meta-NMT.

Impact of Validation Tasks Similarly to training any other neural network, meta-learning still requires early-stopping to avoid overfitting to a specific set of source tasks. In doing so, we observe that the choice of a validation task has non-negligible impact on the final performance. For instance, as shown in Fig. 3, Fi-En benefits more when Ro-En is used for validation, while the opposite happens with Tr-En. The relationship between the task similarity and the impact of a validation task must be investigated further in the future.

Training Set Size We vary the size of the target task's training set and compare the proposed meta-learning strategy and multilingual, transfer learning strategy. We use the emb+enc fine-tuning on Ro-En and Fi-En. Fig. 4 demonstrates that the meta-learning approach is more robust to the drop in the size of the target task's training set. The gap between the meta-learning and transfer learning grows as the size shrinks, confirming the effectiveness of the proposed approach on extremely low-resource language pairs.

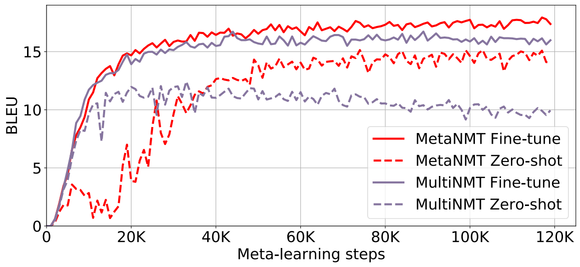

Figure 5: The learning curves of BLEU scores on the validation task (Ro-En).

Impact of Source Tasks In Table 2, we present the results on all five target tasks obtained while varying the source task set. We first see that it is always beneficial to use more source tasks. Although the impact of adding more source tasks varies from one language to another, there is up to 2× improvement going from one source task to 18 source tasks (Lv-En, Fi-En, Tr-En and Ko-En). The same trend can be observed even without any fine-tuning (i.e., unsupervised translation, (Lample et al., 2017; Artetxe et al., 2017b)). In addition, the choice of source languages has different implications for different target languages. For instance, Ro-En benefits more from $\{$Es, Fr, It, Pt$\}$ than from $\{$De, Ru$\}$, while the opposite effect is observed with all the other target tasks.

Training Curves The benefit of meta-learning over multilingual translation is clearly demonstrated when we look at the training curves in Fig. 5. With the multilingual, transfer learning approach, we observe that training rapidly saturates and eventually degrades, as the model overfits to the source tasks. MetaNMT on the other hand continues to improve and never degrades, as the meta-objective ensures that the model is adequate for fine-tuning on target tasks rather than for solving the source tasks.

Sample Translations We present some sample translations from the tested models in Table 3. Inspecting these examples provides the insight into the proposed meta-learning algorithm. For instance, we observe that the meta-learned model without any fine-tuning produces a word-by-word translation in the first example (Tr-En), which is due to the successful use of the universal lexcial representation and the meta-learned initialization. The system however cannot reorder tokens from Turkish to English, as it has not seen any training example of Tr-En. After seeing around 600 sentence pairs (16K English tokens), the model rapidly learns to correctly reorder tokens to form a better translation. A similar phenomenon is observed in the Ko-En example. These cases could be found across different language pairs.

6 Conclusion

In this paper, we proposed a meta-learning algorithm for low-resource neural machine translation that exploits the availability of high-resource languages pairs. We based the proposed algorithm on the recently proposed model-agnostic meta-learning and adapted it to work with multiple languages that do not share a common vocabulary using the technique of universal lexcal representation, resulting in MetaNMT. Our extensive evaluation, using 18 high-resource source tasks and 5 low-resource target tasks, has shown that the proposed MetaNMT significantly outperforms the existing approach of multilingual, transfer learning in low-resource neural machine translation across all the language pairs considered.

The proposed approach opens new opportunities for neural machine translation. First, it is a principled framework for incorporating various extra sources of data, such as source- and target-side monolingual corpora. Second, it is a generic framework that can easily accommodate existing and future neural machine translation systems.

Source (Tr)	google mülteciler için 11 milyon dolar toplamak üzere bağış eşleştirme kampanyasını başlattı .
Target	google launches donation-matching campaign to raise $ 11 million for refugees .
Meta-0	google refugee fund for usd 11 million has launched a campaign for donation .
Meta-16k	google has launched a campaign to collect $ 11 million for refugees .
Source (Ko)	이번에 체포되어 기소된 사람들 중에는 퇴역한 군 고위관리 , 언론인 , 정치인 , 경제인 등이 포함됐다
Target	among the suspects are retired military officials , journalists , politicians , businessmen and others .
Meta-0	last year , convicted people , among other people , of a high-ranking army of journalists in economic and economic policies , were included .
Meta-16k	the arrested persons were included in the charge , including the military officials , journalists , politicians and economists .

Table 3: Sample translations for Tr-En and Ko-En highlight the impact of fine-tuning which results in syntactically better formed translations. We highlight tokens of interest in terms of reordering.

Acknowledgement

This research was supported in part by the Facebook Low Resource Neural Machine Translation Award. This work was also partly supported by Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI) and Samsung Electronics (Improving Deep Learning using Latent Structure). KC thanks support by eBay, TenCent, NVIDIA and CIFAR.

References

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017a. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 451–462.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017b. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. 2017. A teacher-student framework for zero-resource neural machine translation. *arXiv preprint arXiv:1705.00753*.

Yun Chen, Yang Liu, and Victor OK Li. 2018. Zero-resource neural machine translation with multi-agent communication game. *arXiv preprint arXiv:1802.03116*.

Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. 2016. Neural machine translation with pivot languages. *arXiv preprint arXiv:1611.04928*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–Decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. *International Conference on Learning Representations*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL*.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *EMNLP*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018a. Non-autoregressive neural machine translation. *ICLR*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. 2018b. Universal neural machine translation for extremely low resource languages. *arXiv preprint arXiv:1802.05368*.

Caglar Gulcehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. 2018. Dynamic neural turing machine with continuous and discrete addressing schemes. *Neural computation*, 30(4):857–884.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

David Ha, Andrew Dai, and Quoc V Le. 2016a. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016b. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*.

Jason Lee, Kyunghyun Cho, Jason Weston, and Douwe Kiela. 2017. Emergent translation in multi-agent communication. *arXiv preprint arXiv:1710.06922*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2017. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090.

Ilya Sutskever, Oriol Vinyals, and Quôc Lê. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. *arXiv preprint arXiv:1804.09057*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*.

Unsupervised Statistical Machine Translation

Mikel Artetxe, Gorka Labaka, Eneko Agirre
IXA NLP Group
University of the Basque Country (UPV/EHU)
{mikel.artetxe,gorka.labaka,e.agirre}@ehu.eus

Abstract

While modern machine translation has relied on large parallel corpora, a recent line of work has managed to train Neural Machine Translation (NMT) systems from monolingual corpora only (Artetxe et al., 2018c; Lample et al., 2018). Despite the potential of this approach for low-resource settings, existing systems are far behind their supervised counterparts, limiting their practical interest. In this paper, we propose an alternative approach based on phrase-based Statistical Machine Translation (SMT) that significantly closes the gap with supervised systems. Our method profits from the modular architecture of SMT: we first induce a phrase table from monolingual corpora through cross-lingual embedding mappings, combine it with an n-gram language model, and fine-tune hyperparameters through an unsupervised MERT variant. In addition, iterative backtranslation improves results further, yielding, for instance, 14.08 and 26.22 BLEU points in WMT 2014 English-German and English-French, respectively, an improvement of more than 7-10 BLEU points over previous unsupervised systems, and closing the gap with supervised SMT (Moses trained on Europarl) down to 2-5 BLEU points. Our implementation is available at `https://github.com/artetxem/monoses`.

1 Introduction

Neural Machine Translation (NMT) has recently become the dominant paradigm in machine translation (Vaswani et al., 2017). In contrast to more rigid Statistical Machine Translation (SMT) architectures (Koehn et al., 2003), NMT models are trained end-to-end, exploit continuous representations that mitigate the sparsity problem, and overcome the locality problem by making use of unconstrained contexts. Thanks to this additional flexibility, NMT can more effectively exploit large parallel corpora, although SMT is still superior when the training corpus is not big enough (Koehn and Knowles, 2017).

Somewhat paradoxically, while most machine translation research has focused on resource-rich settings where NMT has indeed superseded SMT, a recent line of work has managed to train an NMT system without any supervision, relying on monolingual corpora alone (Artetxe et al., 2018c; Lample et al., 2018). Given the scarcity of parallel corpora for most language pairs, including less-resourced languages but also many combinations of major languages, this research line opens exciting opportunities to bring effective machine translation to many more scenarios. Nevertheless, existing solutions are still far behind their supervised counterparts, greatly limiting their practical usability. For instance, existing unsupervised NMT systems obtain between 15-16 BLEU points in WMT 2014 English-French translation, whereas a state-of-the-art NMT system obtains around 41 (Artetxe et al., 2018c; Lample et al., 2018; Yang et al., 2018).

In this paper, we explore whether the rigid and modular nature of SMT is more suitable for these unsupervised settings, and propose a novel unsupervised SMT system that can be trained on monolingual corpora alone. For that purpose, we present a natural extension of the skip-gram model (Mikolov et al., 2013b) that simultaneously learns word and phrase embeddings, which are then mapped to a cross-lingual space through self-learning (Artetxe et al., 2018b). We use the resulting cross-lingual phrase embeddings to induce a phrase table, and combine it with a language model and a distance-based distortion model to build a standard phrase-based SMT system. The weights of this model are tuned in an unsupervised manner through an iterative Minimum Error Rate Training (MERT) variant, and the entire system

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3632–3642
Brussels, Belgium, October 31 - November 4, 2018.

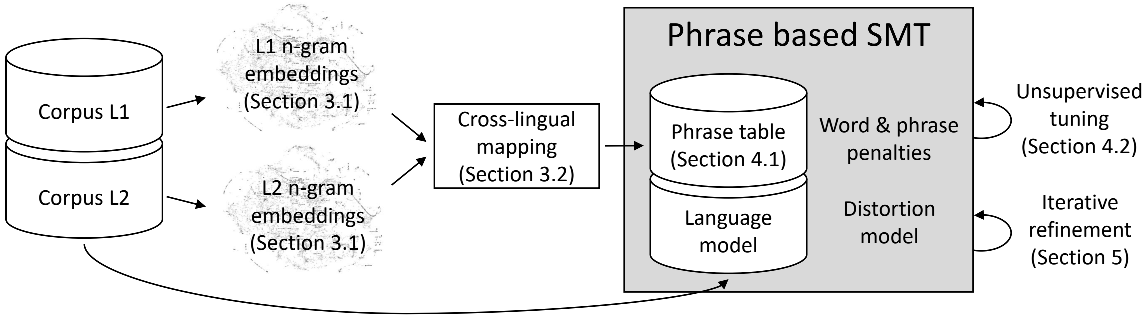

Figure 1: Architecture of our system, with references to sections.

is further improved through iterative backtranslation. The architecture of the system is sketched in Figure 1. Our experiments on WMT German-English and French-English datasets show the effectiveness of our proposal, where we obtain improvements above 7-10 BLEU points over previous unsupervised NMT-based approaches, closing the gap with supervised SMT (Moses trained on Europarl) down to 2-5 points.

The remaining of this paper is structured as follows. Section 2 introduces phrase-based SMT. Section 3 presents our unsupervised approach to learn cross-lingual n-gram embeddings, which are the basis of our proposal. Section 4 describes the proposed unsupervised SMT system itself, while Section 5 discusses its iterative refinement through backtranslation. Section 6 describes the experiments run and the results obtained. Section 7 discusses the related work on the topic, and Section 8 concludes the paper.

2 Background: phrase-based SMT

While originally motivated as a noisy channel model (Brown et al., 1990), phrase-based SMT is now formulated as a log-linear combination of several statistical models that score translation candidates (Koehn et al., 2003). The parameters of these scoring functions are estimated independently based on frequency counts, and their weights are then tuned in a separate validation set. At inference time, a decoder tries to find the translation candidate with the highest score according to the resulting combined model. The specific scoring models found in a standard SMT system are as follows:

- **Phrase table**. The phrase table is a collection of source language n-grams and a list of their possible translations in the target language along with different scores for each of them. So as to translate longer sequences, the decoder combines these partial n-gram translations, and ranks the resulting candidates according to their corresponding scores and the rest of scoring functions. In order to build the phrase table, SMT computes word alignments in both directions from a parallel corpus, symmetrizes these alignments using different heuristics (Och and Ney, 2003), extracts the set of consistent phrase pairs, and scores them based on frequency counts. For that purpose, standard SMT uses 4 scores for each phrase table entry: the direct and inverse lexical weightings, which are derived from word level alignments, and the direct and inverse phrase translation probabilities, which are computed at the phrase level.

- **Language model**. The language model assigns a probability to a word sequence in the target language. Traditional SMT uses n-gram language models for that, which use simple frequency counts over a large monolingual corpus with back-off and smoothing.

- **Reordering model**. The reordering model accounts for different word orders across languages, scoring translation candidates according to the position of each translated phrase in the target language. Standard SMT combines two such models: a distance based distortion model that penalizes deviation from a monotonic order, and a lexical reordering model that incorporates phrase orientation frequencies from a parallel corpus.

- **Word and phrase penalties**. The word and phrase penalties assign a fixed score to every generated word and phrase, and are useful to control the length of the output text and the preference for shorter or longer phrases.

Having trained all these different models, a tuning process is applied to optimize their weights in the resulting log-linear model, which typically maximizes some evaluation metric in a separate validation parallel corpus. A common choice is to optimize the BLEU score through Minimum Error Rate Training (MERT) (Och, 2003).

3 Cross-lingual n-gram embeddings

Section 3.1 presents our proposed extension of skip-gram to learn n-gram embeddings, while Section 3.2 describes how we map them to a shared space to obtain cross-lingual n-gram embeddings.

3.1 Learning n-gram embeddings

Negative sampling skip-gram takes word-context pairs (w, c), and uses logistic regression to predict whether the pair comes from the true distribution as sampled from the training corpus, or it is one of the k draws from a noise distribution (Mikolov et al., 2013b):

$$\log \sigma (w \cdot c) + \sum_{i=1}^{k} \mathbb{E}_{c_N \sim P_D} \left[\log \sigma (-w \cdot c_N) \right]$$

In its basic formulation, both w and c correspond to words that co-occur within a certain window in the training corpus. So as to learn embeddings for non-compositional phrases like *New York Times* or *Toronto Maple Leafs*, Mikolov et al. (2013b) propose to merge them into a single token in a pre-processing step. For that purpose, they use a scoring function based on their co-occurence frequency in the training corpus, with a discounting coefficient δ that penalizes rare words, and iteratively merge those above a threshold:

$$\text{score}(w_i, w_j) = \frac{\text{count}\,(w_i, w_j) - \delta}{\text{count}\,(w_i) \times \text{count}\,(w_j)}$$

However, we also need to learn representations for compositional n-grams in our scenario, as there is not always a 1:1 correspondence for n-grams across languages even for compositional phrases. For instance, the phrase *he will come* would typically be translated as *vendrá* into Spanish, so one would need to represent the entire phrase as a single unit to properly model this relation.

One option would be to merge all n-grams regardless of their score, but this is not straightforward given their overlapping nature, which is further accentuated when considering n-grams of different lengths. While we tried to randomly generate multiple consistent segmentations for each sentence and train the embeddings over the resulting corpus, this worked poorly in our preliminary experiments. We attribute this to the complex interactions arising from the stochastic segmentation (e.g. the co-occurrence distribution changes radically, even for unigrams), severely accentuating the sparsity problem, among other issues.

As an alternative approach, we propose a generalization of skip-gram that learns n-gram embeddings on-the-fly, and has the desirable property of unigram invariance: our proposed model learns the exact same embeddings as the original skip-gram for unigrams, while simultaneously learning additional embeddings for longer n-grams. This way, for each word-context pair (w, c) at distance d within the given window, we update their corresponding embeddings w and c with the usual negative sampling loss. In addition to that, we look at all n-grams p of different lengths that are at the same distance d, and for each pair (p, c), we update the embedding p through negative sampling. In order to enforce unigram invariance, the context c and negative samples c_N, which always correspond to unigrams, are not updated for (p, c). This allows to naturally learn n-gram embeddings according to their co-occurrence patterns as modeled by skip-gram, without introducing subtle interactions that affect its fundamental behavior.

We implemented the above procedure as an extension of *word2vec*, and use it to train monolingual n-gram embeddings with a window size of 5, 300 dimensions, 10 negative samples, 5 iterations and subsampling disabled. So as to keep the model size within a reasonable limit, we restrict the vocabulary to the most frequent 200,000 unigrams, 400,000 bigrams and 400,000 trigrams.

3.2 Cross-lingual mapping

Cross-lingual mapping methods take independently trained word embeddings in two languages, and learn a linear transformation to map them to a shared cross-lingual space (Mikolov et al., 2013a; Artetxe et al., 2018a). Most mapping methods are supervised, and rely on a bilingual dictionary, typically in the range of a few thousand entries, although a recent line of work has managed to achieve comparable results in a fully unsupervised manner based on either self-learning (Artetxe et al., 2017, 2018b) or adversarial training (Zhang et al., 2017a,b; Conneau et al., 2018).

In our case, we use the method of Artetxe et al.

(2018b) to map the n-gram embeddings to a shared cross-lingual space using their open source implementation VecMap[1]. Originally designed for word embeddings, this method builds an initial mapping by connecting the intra-lingual similarity distribution of embeddings in different languages, and iteratively improves this solution through self-learning. The method applies a frequency-based vocabulary cut-off, learning the mapping over the 20,000 most frequent words in each language. We kept this cut-off to learn the mapping over the most frequent 20,000 unigrams, and then apply the resulting mapping to the entire embedding space, including longer n-grams.

4 Unsupervised SMT

As discussed in Section 2, phrase-based SMT follows a modular architecture that combines several scoring functions through a log-linear model. Among the scoring functions found in standard SMT systems, the distortion model and word/phrase penalties are parameterless, while the language model is trained on monolingual corpora, so they can all be directly integrated into our unsupervised system. From the remaining models, typically trained on parallel corpora, we decide to leave the lexical reordering out, as the distortion model already accounts for word reordering. As for the phrase table, we learn cross-lingual n-gram embeddings as discussed in Section 3, and use them to induce and score phrase translation pairs as described next (Section 4.1). Finally, we tune the weights of the resulting log-linear model using an unsupervised procedure based on back-translation (Section 4.2).

Unless otherwise specified, we use Moses[2] with default hyperparameters to implement these different components of our system. We use KenML (Heafield et al., 2013), bundled in Moses by default, to estimate our 5-gram language model with modified Kneser-Ney smoothing, pruning n-grams longer than 3 with a single occurrence.

4.1 Phrase table induction

Given the lack of a parallel corpus from which to **extract phrase translation pairs**, every n-gram in the target language could be taken as a potential translation candidate for each n-gram in the source language. So as to keep the size of the phrase table within a reasonable limit, we train cross-lingual phrase embeddings as described in Section 3, and limit the translation candidates for each source phrase to its 100 nearest neighbors in the target language.

In order to estimate their corresponding **phrase translation probabilities**, we apply the softmax function over the cosine similarities of their respective embeddings. More concretely, given the source language phrase $\bar{e}$ and the translation candidate $\bar{f}$, their direct phrase translation probability is computed as follows[3]:

$$\phi(\bar{f}|\bar{e}) = \frac{\cos(\bar{e}, \bar{f})/\tau}{\sum_{\bar{f}'} \cos(\bar{e}, \bar{f}')/\tau}$$

Note that, in the above formula, $\bar{f}'$ iterates across all target language embeddings, and τ is a constant temperature parameter that controls the confidence of the predictions. In order to tune it, we induce a dictionary over the cross-lingual embeddings themselves with nearest neighbor retrieval, and use maximum likelihood estimation over it. However, inducing the dictionary in the same direction as the probability predictions leads to a degenerated solution (softmax approximates the hard maximum underlying nearest neighbor as τ approaches 0), so we induce the dictionary in the opposite direction and apply maximum likelihood estimation over it:

$$\min_{\tau} \sum_{\bar{f}} \log \phi(\bar{f}|\,\mathrm{NN}_{\bar{e}}(\bar{f})) + \sum_{\bar{e}} \log \phi(\bar{e}|\,\mathrm{NN}_{\bar{f}}(\bar{e}))$$

So as to optimize τ, we use Adam with a learning rate of 0.0003 and a batch size of 200, implemented in PyTorch.

In order to compute the **lexical weightings**, we align each word in the target phrase with the one in the source phrase most likely generating it, and take the product of their respective translation probabilities:

$$\mathrm{lex}(\bar{f}|\bar{e}) = \prod_i \max\left(\epsilon, \max_j w(\bar{f}_i|\bar{e}_j)\right)$$

The constant ϵ guarantees that each target language word will get a minimum probability mass, which is useful to model NULL alignments. In our experiments, we set $\epsilon = 0.001$, which we find to

---

[1] `https://github.com/artetxem/vecmap`

[2] `http://www.statmt.org/moses/`

[3] The inverse phrase translation probability $\phi(\bar{e}|\bar{f})$ is defined analogously.

Algorithm 1 Unsupervised tuning

```
Input: $m_{s \to t}$ (source-to-target models)
Input: $m_{t \to s}$ (target-to-source models)
Input: $c_s$ (source validation corpus)
Input: $c_t$ (target validation corpus)
Output: $w_{s \to t}$ (source-to-target weights)
Output: $w_{t \to s}$ (target-to-source weights)
$w_{t \to s} \leftarrow$ DEFAULT_WEIGHTS
repeat
    $bt_s \leftarrow$ TRANSLATE($m_{t \to s}$, $w_{t \to s}$, $c_t$)
    $w_{s \to t} \leftarrow$ MERT($m_{s \to t}$, $bt_s$, $c_t$)
    $bt_t \leftarrow$ TRANSLATE($m_{s \to t}$, $w_{s \to t}$, $c_s$)
    $w_{t \to s} \leftarrow$ MERT($m_{t \to s}$, $bt_t$, $c_s$)
until convergence
```

work well in practice. Finally, the word translation probabilities $w(\bar{f}_i|\bar{e}_j)$ are computed using the same formula defined for phrase translation probabilities (see above), with the difference that the partition function goes over unigrams only.

4.2 Unsupervised tuning

As discussed in Section 2, standard SMT uses MERT over a small parallel corpus to tune the weights of the different scoring functions combined through its log-linear model. Given that we only have access to monolingual corpora in our scenario, we propose to generate a synthetic parallel corpus through backtranslation (Sennrich et al., 2016) and apply MERT tuning over it, iteratively repeating the process in both directions (see Algorithm 1). For that purpose, we reserve a random subset of 10,000 sentences from each monolingual corpora, and run the proposed algorithm over them for 10 iterations, which we find to be enough for convergence.

5 Iterative refinement

The procedure described in Section 4 suffices to train an SMT system from monolingual corpora which, as shown by our experiments in Section 6, already outperforms previous unsupervised systems. Nevertheless, our proposed method still makes important simplifications that could compromise its potential performance: it does not use any lexical reordering model, its phrase table is limited by the underlying embedding vocabulary (e.g. it does not include phrases longer than trigrams, see Section 3.1), and the phrase translation probabilities and lexical weightings are estimated based on cross-lingual embeddings.

Algorithm 2 Iterative refinement

```
Input: $c_s$ (source language corpus)
Input: $c_t$ (target language corpus)
Input/Output: $m_{t \to s}$ (target-to-source models)
Input/Output: $w_{t \to s}$ (target-to-source weights)
Output: $m_{s \to t}$ (source-to-target models)
Output: $w_{s \to t}$ (source-to-target weights)
$train_s, val_s \leftarrow$ SPLIT($c_s$)
$train_t, val_t \leftarrow$ SPLIT($c_t$)
repeat
    $btt_s \leftarrow$ TRANSLATE($m_{t \to s}$, $w_{t \to s}$, $train_t$)
    $btv_s \leftarrow$ TRANSLATE($m_{t \to s}$, $w_{t \to s}$, $val_t$)
    $m_{s \to t} \leftarrow$ TRAIN($btt_s$, $train_t$)
    $w_{s \to t} \leftarrow$ MERT($m_{s \to t}$, $btv_s$, $val_t$)
    $btt_t \leftarrow$ TRANSLATE($m_{s \to t}$, $w_{s \to t}$, $train_s$)
    $btv_t \leftarrow$ TRANSLATE($m_{s \to t}$, $w_{s \to t}$, $val_s$)
    $m_{t \to s} \leftarrow$ TRAIN($btt_t$, $train_s$)
    $w_{t \to s} \leftarrow$ MERT($m_{t \to s}$, $btv_t$, $val_s$)
until convergence
```

In order to overcome these limitations, we propose an iterative refinement procedure based on backtranslation (Sennrich et al., 2016). More concretely, we generate a synthetic parallel corpus by translating the monolingual corpus in one of the languages with the initial system, and train and tune a standard SMT system over it in the opposite direction. Note that this new system does not have any of the initial restrictions: the phrase table is built and scored using standard word alignment with an unconstrained vocabulary, and a lexical reordering model is also learned. Having done that, we use the resulting system to translate the monolingual corpus in the other language, and train another SMT system over it in the other direction. As detailed in Algorithm 2, this process is repeated iteratively until some convergence criterion is met.

While this procedure would be expected to produce a more accurate model at each iteration, it also happens to be very expensive computationally. In order to accelerate our experiments, we use a random subset of 2 million sentences from each monolingual corpus for training[4], in addition to the 10,000 separate sentences that are held out as a validation set for MERT tuning, and perform a fixed number of 3 iterations of the above algorithm. Moreover, we use FastAlign (Dyer et al., 2013) instead of GIZA++ to make word alignment faster. Other than that, training over the synthetic

---

[4] Note that we reuse the original language model, which is trained in the full corpus.

	WMT-14				WMT-16	
	FR-EN	EN-FR	DE-EN	EN-DE	DE-EN	EN-DE
Artetxe et al. (2018c)	15.56	15.13	10.21	6.55	-	-
Lample et al. (2018)	14.31	15.05	-	-	13.33	9.64
Yang et al. (2018)	15.58	16.97	-	-	14.62	10.86
Proposed system	**25.87**	**26.22**	**17.43**	**14.08**	**23.05**	**18.23**

Table 1: Results of the proposed method in comparison to existing unsupervised NMT systems (BLEU).

parallel corpus is done through standard Moses tools with default settings.

6 Experiments and results

In order to make our experiments comparable to previous work, we use the French-English and German-English datasets from the WMT 2014 shared task. As discussed throughout the paper, our system is trained on monolingual corpora alone, so we take the concatenation of all News Crawl monolingual corpora from 2007 to 2013 as our training data, which we tokenize and truecase using standard Moses tools. The resulting corpus has 749 million tokens in French, 1,606 million tokens in German, and 2,109 million tokens in English. Following common practice, the systems are evaluated in newstest2014 using tokenized BLEU scores as computed by the `multi-bleu.perl` script included in Moses. In addition to that, we also report results in German-English newstest2016 (from WMT 2016), as this was used by some previous work in unsupervised NMT (Lample et al., 2018; Yang et al., 2018)[5]. So as to be faithful to our target scenario, we did not use any parallel data in these language pairs, not even for development purposes. Instead, we ran all our preliminary experiments on WMT Spanish-English data, where we made all development decisions.

We present the results of our final system in comparison to other previous work in Section 6.1. Section 6.2 then presents an ablation study of our proposed method, where we analyze the contribution of its different components. Section 6.3 compares the obtained results to those of different supervised systems, analyzing the effect of some of the inherent limitations of our method in a standard phrase-based SMT system. Finally, Section 6.4 presents some translation examples from our system.

6.1 Main results

We report the results obtained by our proposed system in Table 1. As it can be seen, our system obtains the best published results by a large margin, surpassing previous unsupervised NMT systems by around 10 BLEU points in French-English (both directions), and more than 7 BLEU points in German-English (both directions and datasets).

This way, while previous progress in the task has been rather incremental (Yang et al., 2018), our work represents an important step towards high-quality unsupervised machine translation, with improvements over 50% in all cases. This suggests that, in contrast to previous NMT-based approaches, phrase-based SMT may provide a more suitable framework for unsupervised machine translation, which is in line with previous results in low-resource settings (Koehn and Knowles, 2017).

6.2 Ablation analysis

We present ablation results of our proposed system in Table 2. The first row corresponds to the initial system with our induced phrase table (Section 4.1) and default weights as used by Moses, whereas the second row uses our unsupervised MERT procedure to tune these weights (Section 4.2). The remaining rows represent different iterations of our refinement procedure (Section 5), which uses backtranslation to iteratively train a standard SMT system from a synthetic parallel corpus.

The results show that the initial system with default weights (first row) is already better than previous unsupervised NMT systems (Table 1) by a substantial margin (2-6 BLEU points). Our unsupervised tuning procedure further improves results, bringing an improvement of over 1 BLEU

---

[5] Note that we use the same model trained in WMT 2014 for these experiments, so it is likely that our results could be further improved by using the more extensive data from WMT 2016.

	WMT-14				WMT-16	
	FR-EN	**EN-FR**	**DE-EN**	**EN-DE**	**DE-EN**	**EN-DE**
Unsupervised SMT	21.16	20.13	13.86	10.59	18.01	13.22
+ unsupervised tuning	22.17	22.22	14.73	10.64	18.21	13.12
+ iterative refinement (it1)	24.81	26.53	16.01	13.45	20.76	16.94
+ iterative refinement (it2)	**26.13**	**26.57**	17.30	13.95	22.80	18.18
+ iterative refinement (it3)	25.87	26.22	**17.43**	**14.08**	**23.05**	**18.23**

Table 2: Ablation results (BLEU). The last row corresponds to our full system. Refer to the text for more details.

		WMT-14				WMT-16	
		FR-EN	**EN-FR**	**DE-EN**	**EN-DE**	**DE-EN**	**EN-DE**
Supervised	NMT (transformer)	-	41.8	-	28.4	-	-
	WMT best	35.0	35.8	29.0	20.6	40.2	34.2
	SMT (europarl)	30.61	30.82	20.83	16.60	26.38	22.12
	+ w/o lexical reord.	30.54	30.33	20.37	16.34	25.99	22.20
	+ constrained vocab.	30.04	30.10	19.91	16.32	25.66	21.53
	+ unsup. tuning	29.32	29.46	17.75	15.45	23.35	19.86
Unsup.	Proposed system	25.87	26.22	17.43	14.08	23.05	18.23

Table 3: Results of the proposed method in comparison to supervised systems (BLEU). Transformer results reported by Vaswani et al. (2017). SMT variants are incremental (e.g. 2nd includes 1st). Refer to the text for more details.

point in both French-English directions, although its contribution is somewhat weaker for German-to-English (almost 1 BLEU point in WMT 2014 but only 0.2 in WMT 2016), and does not make any difference for English-to-German.

The proposed iterative refinement method has a much stronger positive effect, with improvements over 2.5 BLEU points in all cases, and up to 5 BLEU points in some. Most gains come in the first iteration, while the second iteration brings weaker improvements and the algorithm seems to converge in the third iteration, with marginal improvements for German-English and a small drop in performance for French-English.

6.3 Comparison with supervised systems

So as to put our results into perspective, Table 3 comprises the results of different supervised methods in the same test sets. More concretely, we report the results of the Transformer (Vaswani et al., 2017), an NMT system based on self-attention that is the current state-of-the-art in machine translation, along with the scores obtained by the best performing system in each WMT shared task at the time, and those of a standard phrase-based SMT system trained on Europarl and tuned on newstest2013 using Moses. We also report the effect of removing lexical reordering from the latter as we do in our initial system (Section 4), restricting the vocabulary to the most frequent unigram, bigram and trigrams as we do when training our embeddings (Section 3), and using our unsupervised tuning procedure over a subset of the monolingual corpus (Section 4.2) instead of using standard MERT tuning over newstest2013.

Quite surprisingly, our proposed system, trained exclusively on monolingual corpora, is relatively close to a comparable phrase-based SMT system trained on Europarl, with differences below 5 BLEU points in all cases and as little as 2.5 in some. Note that both systems use the exact same language model trained on News Crawl, making them fully comparable in terms of the monolingual corpora they have access to. While more of a baseline than the state-of-the-art, note that Moses+Europarl is widely used as a reference system in machine translation. As such, we think that our results are very encouraging, as they show

Source	**Reference**	**Proposed system**
D'autres révélations ont fait état de documents divulgués par Snowden selon lesquels la NSA avait intercepté des données et des communications émanant du téléphone portable de la chancelière allemande Angela Merkel et de ceux de 34 autres chefs d'État.	Other revelations cited documents leaked by Snowden that the NSA monitored German Chancellor Angela Merkel's cellphone and those of up to 34 other world leaders.	Other disclosures have reported documents disclosed by Snowden suggested the NSA had intercepted communications and data from the mobile phone of German Chancellor Angela Merkel and those of 32 other heads of state.
La NHTSA n'a pas pu examiner la lettre d'information aux propriétaires en raison de l'arrêt de 16 jours des activités gouvernementales, ce qui a ralenti la croissance des ventes de véhicules en octobre.	NHTSA could not review the owner notification letter due to the 16-day government shutdown, which tempered auto sales growth in October.	The NHTSA could not consider the letter of information to owners because of halting 16-day government activities, which slowed the growth in vehicle sales in October.
Le M23 est né d'une mutinerie, en avril 2012, d'anciens rebelles, essentiellement tutsi, intégrés dans l'armée en 2009 après un accord de paix.	The M23 was born of an April 2012 mutiny by former rebels, principally Tutsis who were integrated into the army in 2009 following a peace agreement.	M23 began as a mutiny in April 2012, former rebels, mainly Tutsi integrated into the national army in 2009 after a peace deal.
Tunks a déclaré au Sunday Telegraph de Sydney que toute la famille était «extrêmement préoccupée» du bien-être de sa fille et voulait qu'elle rentre en Australie.	Tunks told Sydney's Sunday Telegraph the whole family was "extremely concerned" about his daughter's welfare and wanted her back in Australia.	Tunks told The Times of London from Sydney that the whole family was "extremely concerned" of the welfare of her daughter and wanted it to go in Australia.

Table 4: Randomly chosen translation examples from French→English newstest2014.

that our fully unsupervised system is already quite close to this competitive baseline.

In addition to that, the results for the constrained variants of this SMT system justify some of the simplifications required by our approach. In particular, removing lexical reordering and constraining the phrase table to the most frequent n-grams, as we do for our initial system, has a relatively small effect, with a drop of less than 1 BLEU point in all cases, and as little as 0.28 in some. Replacing standard MERT tuning with our unsupervised variant does cause a considerable drop in performance, although it is below 2.5 BLEU points even in the worst case, and our unsupervised tuning method is still better than using default weights as reported in Table 2. This shows the importance of tuning in SMT, suggesting that these results could be further improved if one had access to a small parallel corpus for tuning.

6.4 Qualitative results

Table 4 shows some of the translations produced by the proposed system for French→English. Note that these examples where randomly taken from the test set, so they should be representative of the general behavior of our approach.

While the examples reveal certain adequacy issues (e.g. *The Times of London from Sidney* instead of *Sydney's Sunday Telegraph*), and the produced output is not perfectly grammatical (e.g. *go in Australia*), our translations are overall quite accurate and fluent, and one could get a reasonable understanding of the original text from them. This suggests that unsupervised machine translation can indeed be a usable alternative in low resource settings.

7 Related work

Similar to our approach, statistical decipherment also attempts to build machine translation systems from monolingual corpora. For that purpose, existing methods treat the source language as ciphertext, and model its generation through a noisy channel model involving two steps: the generation of the original English sentence and the probabilistic replacement of the words in it (Ravi and Knight, 2011; Dou and Knight, 2012). The English generative process is modeled using an n-gram language model, and the channel model parameters are estimated using either expectation maximization or Bayesian inference. Subsequent work has attempted to enrich these models with additional information like syntactic knowledge (Dou and Knight, 2013) and word embeddings (Dou et al., 2015). Nevertheless, these systems work in a word-by-word basis and have only been shown to work in limited settings, being often evaluated in word-level translation. In contrast, our method builds a fully featured phrase-based SMT system, and achieves competitive performance in a standard machine translation task.

More recently, Artetxe et al. (2018c) and Lample et al. (2018) have managed to train a standard attentional encoder-decoder NMT system from monolingual corpora alone. For that purpose, they use a shared encoder for both languages with pre-trained cross-lingual embeddings, and train the entire system using a combination of denoising, backtranslation and, in the case of Lample et al. (2018), adversarial training. This method was further improved by Yang et al. (2018), who use a separate encoder for each language, sharing only a subset of their parameters, and incorporate two generative adversarial networks. However, our results in Section 6.1 show that our SMT-based approach obtains substantially better results.

Our method is also connected to some previous approaches to improve machine translation using monolingual corpora. In particular, the generation of a synthetic parallel corpus through backtranslation (Sennrich et al., 2016), which is a key component of our unsupervised tuning and iterative refinement procedures, has been previously used to improve NMT. In addition, there have been several proposals to extend the phrase table of SMT systems by inducing translation candidates and/or scores from monolingual corpora, using either statistical decipherment methods (Dou and Knight, 2012, 2013) or cross-lingual embeddings (Zhao et al., 2015; Wang et al., 2016). While all these methods exploit monolingual corpora to enhance an existing machine translation system previously trained on parallel corpora, our approach learns a fully featured phrase-based SMT system from monolingual corpora alone.

8 Conclusions and future work

In this paper, we propose a novel unsupervised SMT system that can be trained on monolingual corpora alone. For that purpose, we extend the skip-gram model (Mikolov et al., 2013b) to simultaneously learn word and phrase embeddings, and map them to a cross-lingual space adapting previous unsupervised techniques (Artetxe et al., 2018b). The resulting cross-lingual phrase embeddings are used to induce a phrase table, which coupled with an n-gram language model and distance-based distortion yields an unsupervised phrase-based SMT system. We further improve results tuning the weights with our unsupervised MERT variant, and obtain additional improvements re-training the entire system through iterative back-translation. Our implementation is available as an open source project at `https://github.com/artetxem/monoses`.

Our experiments on standard WMT French-English and German-English datasets confirm the effectiveness of our proposal, where we obtain improvements above 10 and 7 BLEU points over previous NMT-based approaches, respectively, closing the gap with supervised SMT (Moses trained on Europarl) down to 2-5 points.

In the future, we would like to extend our approach to semi-supervised scenarios with small parallel corpora, which we expect to be particularly helpful for tuning purposes. Moreover, we would like to try a hybrid approach with NMT, using our unsupervised SMT system to generate a synthetic parallel corpus and training an NMT system over it through iterative backtranslation.

Acknowledgments

This research was partially supported by the Spanish MINECO (TUNER TIN2015-65308-C5-1-R, MUSTER PCIN-2015-226 and TADEEP TIN2015-70214-P, cofunded by EU FEDER), the UPV/EHU (excellence research group), and the NVIDIA GPU grant program. Mikel Artetxe enjoys a doctoral grant from the Spanish MECD.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 5012–5019.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of* the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018c. Unsupervised neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 266–275, Jeju Island, Korea. Association for Computational Linguistics.

Qing Dou and Kevin Knight. 2013. Dependency-based decipherment for resource-limited machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1668–1676, Seattle, Washington, USA. Association for Computational Linguistics.

Qing Dou, Ashish Vaswani, Kevin Knight, and Chris Dyer. 2015. Unifying bayesian inference and vector space models for improved decipherment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 836–845, Beijing, China. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified kneser-ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 12–21, Portland, Oregon, USA. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Rui Wang, Hai Zhao, Sabine Ploux, Bao-Liang Lu, and Masao Utiyama. 2016. A bilingual graph-based semantic model for statistical machine translation. In *IJCAI*, pages 2950–2956.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 46–55. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017a. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970, Vancouver, Canada. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945, Copenhagen, Denmark. Association for Computational Linguistics.

Kai Zhao, Hany Hassan, and Michael Auli. 2015. Learning translation models from monolingual continuous representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1527–1536, Denver, Colorado. Association for Computational Linguistics.

A Visual Attention Grounding Neural Model for Multimodal Machine Translation

Mingyang Zhou **Runxiang Cheng** **Yong Jae Lee** **Zhou Yu**
Department of Computer Science
University of California, Davis
`{minzhou, samcheng, yongjaelee, joyu}@ucdavis.edu`

Abstract

We introduce a novel multimodal machine translation model that utilizes parallel visual and textual information. Our model jointly optimizes the learning of a shared visual-language embedding and a translator. The model leverages a visual attention grounding mechanism that links the visual semantics with the corresponding textual semantics. Our approach achieves competitive state-of-the-art results on the Multi30K and the Ambiguous COCO datasets. We also collected a new multilingual multimodal product description dataset to simulate a real-world international online shopping scenario. On this dataset, our visual attention grounding model outperforms other methods by a large margin.

1 Introduction

Multimodal machine translation is the problem of translating sentences paired with images into a different target language (Elliott et al., 2016). In this setting, translation is expected to be more accurate compared to purely text-based translation, as the visual context could help resolve ambiguous multi-sense words. Examples of real-world applications of multimodal (vision plus text) translation include translating multimedia news, web product information, and movie subtitles.

Several previous endeavours (Huang et al., 2016; Calixto et al., 2017a; Elliott and Kádár, 2017) have demonstrated improved translation quality when utilizing images. However, how to effectively integrate the visual information still remains a challenging problem. For instance, in the WMT 2017 multimodal machine translation challenge (Elliott et al., 2017), methods that incorporated visual information did not outperform pure text-based approaches with a big margin.

In this paper, we propose a new model called Visual Attention Grounding Neural Machine Translation (VAG-NMT) to leverage visual information more effectively. We train VAG-NMT with a multitask learning mechanism that simultaneously optimizes two objectives: (1) learning a translation model, and (2) constructing a vision-language joint semantic embedding. In this model, we develop a visual attention mechanism to learn an attention vector that values the words that have closer semantic relatedness with the visual context. The attention vector is then projected to the shared embedding space to initialize the translation decoder such that the source sentence words that are more related to the visual semantics have more influence during the decoding stage. When evaluated on the benchmark Multi30K and the Ambiguous COCO datasets, our VAG-NMT model demonstrates competitive performance compared to existing state-of-the-art multimodal machine translation systems.

Another important challenge for multimodal machine translation is the lack of a large-scale, realistic dataset. To our knowledge, the only existing benchmark is Multi30K (Elliott et al., 2016), which is based on an image captioning dataset, Flickr30K (Young et al., 2014) with manual German and French translations. There are roughly 30K parallel sentences, which is relatively small compared to text-only machine translation datasets that have millions of sentences such as WMT14 EN→DE. Therefore, we propose a new multimodal machine translation dataset called IKEA to simulate the real-world problem of translating product descriptions from one language to another. Our IKEA dataset is a collection of parallel English, French, and German product descriptions and images from IKEA's and UNIQLO's websites. We have included a total of 3,600 products so far and will include more in the future.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3643–3653
Brussels, Belgium, October 31 - November 4, 2018.

2 Related Work

In the machine translation literature, there are two major streams for integrating visual information: approaches that (1) employ separate attention for different (text and vision) modalities, and (2) fuse visual information into the NMT model as part of the input. The first line of work learns independent context vectors from a sequence of text encoder hidden states and a set of location-preserving visual features extracted from a pre-trained convnet, and both sets of attentions affect the decoder's translation (Calixto et al., 2017a; Helcl and Libovický, 2017). The second line of work instead extracts a global semantic feature and initializes either the NMT encoder or decoder to fuse the visual context (Calixto et al., 2017b; Ma et al., 2017). While both approaches demonstrate significant improvement over their Text-Only NMT baselines, they still perform worse than the best monomodal machine translation system from the WMT 2017 shared task (Zhang et al., 2017).

The model that performs best in the multimodal machine translation task employed image context in a different way. (Huang et al., 2016) combine region features extracted from a region-proposal network (Ren et al., 2015) with the word sequence feature as the input to the encoder, which leads to significant improvement over their NMT baseline. The best multimodal machine translation system in WMT 2017 (Caglayan et al., 2017) performs element-wise multiplication of the target language embedding with an affine transformation of the convnet image feature vector as the mixed input to the decoder. While this method outperforms all other methods in WMT 2017 shared task workshop, the advantage over the monomodal translation system is still minor.

The proposed visual context grounding process in our model is closely related to the literature on multimodal shared space learning. (Kiros et al., 2014) propose a neural language model to learn a visual-semantic embedding space by optimizing a ranking objective, where the distributed representation helps generate image captions. (Karpathy and Li, 2014) densely align different objects in the image with their corresponding text captions in the shared space, which further improves the quality of the generated caption. In later work, multimodal shared space learning was extended to multimodal multilingual shared space learning. (Calixto et al., 2017c) learn a multi-modal multilingual shared space through optimization of a modified pairwise contrastive function, where the extra multilingual signal in the shared space leads to improvements in image-sentence ranking and semantic textual similarity task. (Gella et al., 2017) extend the work from (Calixto et al., 2017c) by using the image as the pivot point to learn the multilingual multimodal shared space, which does not require large parallel corpora during training. Finally, (Elliott and Kádár, 2017) is the first to integrate the idea of multimodal shared space learning to help multimodal machine translation. Their multi-task architecture called "imagination" shares an encoder between a primary task of the classical encoder-decoder NMT and an auxiliary task of visual feature reconstruction.

Our VAG-NMT mechanism is inspired by (Elliott and Kádár, 2017), but has important differences. First, we modify the auxiliary task as a visual-text shared space learning objective instead of the simple image reconstruction objective. Second, we create a visual-text attention mechanism that captures the words that share a strong semantic meaning with the image, where the grounded visual-context has more impact on the translation. We show that these enhancements lead to improvement in multimodal translation quality over (Elliott and Kádár, 2017).

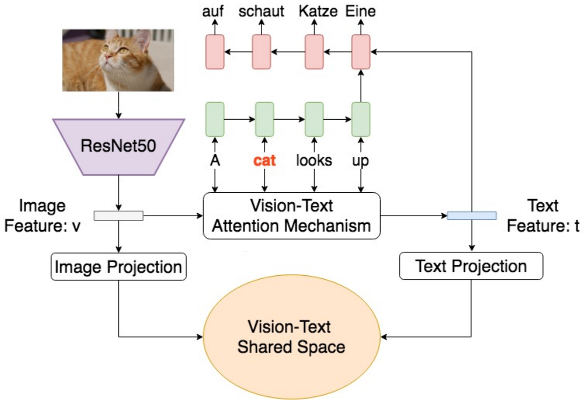

Figure 1: An overview of the VAG-NMT structure

3 Visual Attention Grounding NMT

Given a set of parallel sentences in language X and Y, and a set of corresponding images V paired with each sentence pair, the model aims to translate sentences $\{x_i\}_{i=1}^{N} \in X$ in language X to sentences $\{y_i\}_{i=1}^{N} \in Y$ in language Y with the assistance of images $\{v_i\}_{i=1}^{N} \in V$.

We treat the problem of multimodal machine translation as a joint optimization of two tasks: (1) learning a robust translation model and (2) constructing a visual-language shared embedding that grounds the visual semantics with text. Figure 1 shows an overview of our VAG-NMT model. We adopt a state-of-the-art attention-based sequence-to-sequence structure (Bahdanau et al., 2014) for translation. For the joint embedding, we obtain the text representation using a weighted sum of hidden states from the encoder of the sequence-to-sequence model and we obtain the image representation from a pre-trained convnet. We learn the weights using a visual attention mechanism, which represents the semantic relatedness between the image and each word in the encoded text. We learn the shared space with a ranking loss and the translation model with a cross entropy loss.

The joint objective function is defined as:

$$J(\theta_T, \phi_V) = \alpha J_T(\theta_T) + (1 - \alpha) J_V(\phi_V), \tag{1}$$

where J_T is the objective function for the sequence-to-sequence model, J_V is the objective function for joint embedding learning, θ_T are the parameters in the translation model, and ϕ_V are the parameters for the shared vision-language embedding learning, and α determines the contribution of the machine translation loss versus the visual grounding loss. Both J_T and J_V share the parameters of the encoder from the neural machine translation model. We describe details of the two objective functions in Section 3.2.

3.1 Encoder

We first encode an n-length source sentence $\{x\}$, as a sequence of tokens $x = \{x_1, x_2, \ldots, x_n\}$, with a bidirectional GRU (Schuster and Paliwal, 1997; Cho et al., 2014). Each token is represented by a one-hot vector, which is then mapped into an embedding e_i through a pre-trained embedding matrix. The bidirectional GRU processes the embedding tokens in two directions: left-to-right (forward) and right-to-left (backward). At every time step, the encoder's GRU cell generates two corresponding hidden state vectors: $\overrightarrow{h_i} = \overrightarrow{GRU(h_{i-1}, e_i)}$ and $\overleftarrow{h_i} = \overleftarrow{GRU(h_{i-1}, e_i)}$. The two hidden state vectors are then concatenated together to serve as the encoder hidden state vector of the source token at step i: $h_i = [\overleftarrow{h_i}, \overrightarrow{h_i}]$.

3.2 Shared embedding objective

After encoding the source sentence, we project both the image and text into the shared space to find a good distributed representation that can capture the semantic meaning across the two modalities. Previous work has shown that learning a multimodal representation is effective for grounding knowledge between two modalities (Kiros et al., 2014; Chrupala et al., 2015). Therefore, we expect the shared encoder between the two objectives to facilitate the integration of the two modalities and positively influence translation during decoding.

To project the image and the source sentence to a shared space, we obtain the visual embedding (v) from the pool5 layer of ResNet50 (He et al., 2015a) pre-trained on ImageNet classification (Russakovsky et al., 2015), and the source sentence embedding using the weighted sum of encoder hidden state vectors ($\{h_i\}$) to represent the entire source sentence (t). We project each $\{h_i\}$ to the shared space through an embedding layer. As different words in the source sentence will have different importance, we employ a visual-language attention mechanism—inspired by the attention mechanism applied in sequence-to-sequence models (Bahdanau et al., 2014)—to emphasize words that have the stronger semantic connection with the image. For example, the highlighted word "cat" in the source sentence in Fig. 1 has the more semantic connection with the image.

Specifically, we produce a set of weights $\beta = \{\beta_1, \beta_2, \ldots, \beta_n\}$ with our visual-attention mechanism, where the attention weight β_i for the i'th word is computed as:

$$\beta_i = \frac{\exp(z_i)}{\sum_{l=1}^{N} \exp(z_l)}, \tag{2}$$

and $z_i = \tanh(W_v v) \cdot \tanh(W_h h_i)$ is computed by taking the dot product between the transformed encoder hidden state vector h_i and the transformed image feature vector v, and W_v and W_h are the association transformation parameters.

Then, we can get a weighted sum of the encoder hidden state vectors $t = \sum_{i=1}^{n} \beta_i h_i$ to represent the semantic meaning of the entire source sentence. The next step is to project the source sentence feature vector t and the image feature vector v into the same shared space. The projected vector for text is: $t_{emb} = \tanh(W_{t_{emb}} t + b_{t_{emb}})$ and the projected vector for image is: $v_{emb} = \tanh(W_{v_{emb}} v + b_{v_{emb}})$.

We follow previous work on visual semantic embedding (Kiros et al., 2014) to minimize a pairwise ranking loss to learn the shared space:

$$J_V(\phi_V) = \sum_p \sum_k \max\{0, \gamma - s(v_p, t_p) + s(v_p, t_{k \neq p})\} + \sum_k \sum_p \max\{0, \gamma - s(t_k, v_k) + s(t_k, v_{p \neq k})\}, \tag{3}$$

where γ is a margin, and s is the cosine distance between two vectors in the shared space. k and p are the indexes of the images and text sentences, respectively. $t_{k \neq p}$ and $v_{p \neq k}$ are the contrastive examples with respect to the selected image and the selected source text, respectively. When the loss decreases, the distance between a paired image and sentence will drop while the distance between an unpaired image and sentence will increase.

In addition to grounding the visual context into the shared encoder through the multimodal shared space learning, we also initialize the decoder with the learned attention vector t such that the words that have more relatedness with the visual semantics will have more impact during the decoding (translation) stage. However, we may not want to solely rely on only a few most important words. Thus, to produce the initial hidden state of the decoder, we take a weighted average of the attention vector t and the mean of encoder hidden states:

$$s_0 = \tanh(W_{init}(\lambda t + (1 - \lambda)\frac{1}{N}\sum_i^N h_i)), \tag{4}$$

where λ determines the contribution from each vector. Through our experiments, we find the best value for λ is 0.5.

3.3 Translation objective

During the decoding stage, at each time step j, the decoder generates a decoder hidden state s_j from a conditional GRU cell (Sennrich et al., 2017) whose input is the previously generated translation token y_{j-1}, the previous decoder hidden state s_{j-1}, and the context vector c_j at the current time step:

$$s_j = \text{cGRU}(s_{j-1}, y_{j-1}, c_j) \tag{5}$$

The context vector c_j is a weighted sum of the encoder hidden state vectors, and captures the relevant source words that the decoder should focus on when generating the current translated token y_j. The weight associated with each encoder hidden state is determined by a feed-forward network. From the hidden state s_j we can predict the conditional distribution of the next token y_j with a fully-connected layer W_o given the previous token's language embedding e_{j-1}, the current hidden state d_j and the context vector for current step c_j:

$$p(y_j | y_{j-1}, x) = \text{softmax}(W_o o_t), \tag{6}$$

where $o_t = \tanh(W_e e_{j-1} + W_d d_j + W_c c_j)$. The three inputs are transformed with W_e, W_d, and W_c, respectively and then summed before being fed into the output layer.

We train the translation objective by optimizing a cross entropy loss function:

$$J_T(\theta_T) = -\sum_j \log p(y_j | y_{j-1}, x) \tag{7}$$

By optimizing the objective of the translation and the multimodal shared space learning tasks jointly along with the visual-language attention mechanism, we can simultaneously learn a general mapping between the linguistic signals in two languages and grounding of relevant visual content in the text to improve the translation.

4 IKEA Dataset

Previous available multimodal machine translation models are only tested on image caption datasets, we, therefore, propose a new dataset, IKEA, that has the real-world application of international online shopping. We create the dataset by crawling commercial products' descriptions and images from IKEA and UNIQLO websites. There are 3,600 products and we plan to expand the data in the future. Each sample is composed of the web-crawled English description of a product, an image of the product, and the web-crawled German or French description of the product.

Different than the image caption datasets, the German or French sentences in the IKEA dataset is not an exact parallel translation of its English sentence. Commercial descriptions in different languages can be vastly different in surface form but still keep the core semantic meaning. We think IKEA is a good data set to simulate real-world multimodal translation problems. The sentence in the IKEA dataset contains 60-70 words on average, which is five times longer than the average text length in Multi30K (Elliott et al., 2016). These sentences not only describe the visual appearance of the product, but also the product usage. Therefore, capturing the connection between

	English → German		English → French	
Method	BLEU	METEOR	BLEU	METEOR
Imagination (Elliott and Kádár, 2017)	30.2	51.2	N/A	N/A
LIUMCVC (Caglayan et al., 2017)	31.1 ± 0.7	52.2 ± 0.4	52.7 ± 0.9	69.5 ± 0.7
Text-Only NMT	**31.6 ± 0.5**	52.2 ± 0.3	53.5 ± 0.7	70.0 ± 0.7
VAG-NMT	**31.6 ± 0.3**	52.2 ± 0.3	**53.8 ± 0.3**	**70.3 ± 0.5**

Table 1: Translation results on the Multi30K dataset

	English → German		English → French	
Method	BLEU	METEOR	BLEU	METEOR
Imagination (Elliott and Kádár, 2017)	28.0	**48.1**	N/A	N/A
LIUMCVC (Caglayan et al., 2017)	27.1 ± 0.9	47.2 ± 0.6	43.5 ± 1.2	63.2 ± 0.9
Text-Only NMT	27.9 ± 0.6	47.8 ± 0.6	44.6 ± 0.6	64.2 ± 0.5
VAG-NMT	**28.3 ± 0.6**	48.0 ± 0.5	**45.0 ± 0.4**	**64.7 ± 0.4**

Table 2: Translation results on the Ambiguous COCO dataset

visual semantics and the text is more challenging on this dataset. The dataset statistics and an example of the IKEA dataset is in Appendix A.

5 Experiments and Results

5.1 Datasets

We evaluate our proposed model on three datasets: Multi30K (Elliott et al., 2016), Ambiguous COCO (Elliott et al., 2017), and our newly-collected IKEA dataset. The Multi30K dataset is the largest existing human-labeled dataset for multimodal machine translation. It consists of 31,014 images, where each image is annotated with an English caption and manual translations of image captions in German and French. There are 29,000 instances for training, 1,014 instances for validation, and 1,000 for testing. Additionally, we also evaluate our model on the Ambiguous COCO Dataset collected in the WMT2017 multimodal machine translation challenge (Elliott et al., 2017). It contains 461 images from the MSCOCO dataset (Lin et al., 2014), whose captions contain verbs with ambiguous meanings.

5.2 Setting

We pre-process all English, French, and German sentences by normalizing the punctuation, lower casing, and tokenizing with the Moses toolkit. A Byte-Pair-Encoding (BPE) (Sennrich et al., 2015) operation with 10K merge operations is learned from the pre-processed data and then applied to segment words. We restore the original words by concatenating the subwords segmented by BPE in post-processing. During training, we apply early stopping if there is no improvement in BLEU score on validation data for 10 validation steps. We apply beam search decoding to generate translation with beam size equal to 12. We evaluate the performance of all models using BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014). The setting used in IKEA dataset is the same as Multi30K, except that we lower the default batch size from 32 to 12; since IKEA dataset has long sentences and large variance in sentence length, we use smaller batches to make the training more stable. Full details of our hyperparameter choices can be found in Appendix B. We run all models five times with different random seeds and report the mean and standard deviation.

5.3 Results

We compare the performance of our model against the state-of-the-art multimodal machine translation approaches and the text-only baseline. The idea of our model is inspired by the "Imagination" model (Elliott and Kádár, 2017), which unlike our models, simply averages the encoder hidden states for visual grounding learning. As "Imagination" does not report its performance on Multi30K 2017 and Ambiguous COCO in its original paper, we directly use their result reported in the WMT 2017 shared task as a comparison. LIUMCVC is the best multimodal machine translation model in WMT 2017 multimodal machine translation challenge and exploits visual information with several different methods. We always compare our VAG-NMT with the method that has been reported to

	English → German		English → French	
Method	BLEU	METEOR	BLEU	METEOR
LIUMCVC-Multi	59.9 ± 1.9	63.8 ± 0.4	58.4 ± 1.6	64.6 ± 1.8
Text-Only NMT	61.9 ± 0.9	65.6 ± 0.9	65.2 ± 0.7	$\mathbf{69.0 \pm 0.2}$
VAG-NMT	$\mathbf{63.5 \pm 1.2}$	$\mathbf{65.7 \pm 0.1}$	$\mathbf{65.8 \pm 1.2}$	68.9 ± 1.4

Table 3: Translation results on the IKEA dataset

have the best performance on each dataset.

Our VAG-NMT surpasses the results of the “Imagination" model and the LIUMCVC’s model by a noticeable margin in terms of BLEU score on both the Multi30K dataset (Table 1) and the Ambiguous COCO dataset (Table 2). The METEOR score of our VAG-NMT is slightly worse than that of "Imagination" for English -> German on Ambiguous COCO Dataset. This is likely because the “Imagination" result was produced by ensembling the result of multiple runs, which typically leads to 1-2 higher BLEU and METEOR points compared to a single run. Thus, we expect our VAG-NMT to outperform the “Imagination" baseline if we also use an ensemble.

We observe that our multimodal VAG-NMT model has equal or slightly better result compared to the text-only neural machine translation model on the Multi30K dataset. On the Ambiguous COCO dataset, our VAG-NMT demonstrates clearer improvement over this text-only baseline. We suspect this is because Multi30K does not have many cases where images can help improve translation quality, as most of the image captions are short and simple. In contrast, Ambiguous COCO was purposely curated such that the verbs in the captions can have ambiguous meaning. Thus, visual context will play a more important role in Ambiguous COCO; namely, to help clarify the sense of the source text and guide the translation to select the correct word in the target language.

We then evaluate all models on the IKEA dataset. Table 3 shows the results. Our VAG-NMT has a higher value in BLEU and a comparable value in METEOR compared to the Text-only NMT baseline. Our VAG-NMT outperforms LIUMCVC’s multimodal system by a large margin, which shows that the LIUMCVC’s multimodal’s good performance on Multi30K does not generalize to this real-world product dataset. The good performance may come from the visual attention mechanism that learns to focus on text segments that are related to the images. Such attention therefore teaches the decoder to apply the visual context to translate those words. This learned attention is especially useful for datasets with long sentences that have irrelevant text information with respect to the image.

(a) *a cyclist is wearing a helmet*

(b) *a black dog and his favorite toys.*

Figure 2: Top five images retrieved using the given caption. The original corresponding image of the caption is highlighted with a red bounding box.

5.4 Multimodal embedding results

To assess the learned joint embedding, we perform an image retrieval task evaluated using the Recall@K metric (Kiros et al., 2014) on the Multi30K dataset.

We project the image feature vectors $V = \{v_1, v_2, \dots, v_n\}$ and their corresponding captions $S = \{s_1, s_2, \dots, s_n\}$ into a shared space. We follow the experiments conducted by the previous visual semantic embedding work (Kiros et al., 2014), where for each embedded text vector, we find the closest image vectors around it based on the cosine similarity. Then we can compute the recall rate of the paired image in the top K nearest neighbors, which is also known as $R@K$ score. The shared space learned with VAG-NMT achieves 64% R@1, 88.6% R@5, and 93.8% R@10 on Multi30K, which demonstrates the good quality of the learned shared semantics. We also achieved 58.13% R@1, 87.38% R@5 and 93.74% R@10 on IKEA dataset; 41.35% R@1, 85.48% R@5 and 92.56% R@10 on COCO dataset. Besides the quantitative results, we also show several qualitative results in Figure 2. We show the top five images retrieved by the example captions.

a person is skiing or snowboarding down a mountainside .

a mountain climber trekking through the snow with a pick and a blue hat .

the snowboarder is jumping in the snow .

two women are water rafting .

three people in a blue raft on a river of brown water .

people in rafts watch as two men fall out of their own rafts .

Figure 3: Visual-text attention score on sample data from Multi30K. The first and second rows show the three closest images to the caption *a person is skiing or snowboarding down a mountainside* and *two woman are water rafting*, respectively. The original caption is listed under each image. We highlight the three words with highest attention in red.

The images share "cyclist", "helmet", and "dog" mentioned in the caption.

5.5 Human evaluation

We use Facebook to hire raters that speak both German and English to evaluate German translation quality. As Text-Only NMT has the highest BLEU results among all baseline models, we compare the translation quality between the Text-Only and the VAG-NMT on all three datasets. We randomly selected 100 examples for evaluation for each dataset. The raters are informed to focus more on semantic meaning than grammatical correctness when indicating the preference of the two translations. They are also given the option to choose a tie if they cannot decide. We hired two raters and the inter-annotator agreement is 0.82 in Cohen's Kappa.

We summarize the results in Table 4, where we list the number of times that raters prefer one translation over another or think they are the same quality. Our VAG-NMT performs better than Text-Only NMT on MSCOCO and IKEA dataset, which correlates with the automatic evaluation metrics. However, the result of VAG-NMT is slightly worse than the Text-Only NMT on the Multi30K test dataset. This also correlates with the result of automatic evaluation metrics.

Finally, we also compare the translation quality between LIUMCVC multimodal and VAG-NMT on 100 randomly selected examples from the IKEA dataset. VAG-NMT performs better than LIUMCVC multimodal. The raters prefer our VAG-NMT in 91 cases, LIUMCVC multimodal in 68 cases, and cannot tell in 47 cases.

	MSCOCO	Multi30K	IKEA
Text-Only NMT	76	**72**	75
VAG-NMT	**94**	71	**82**
Tie	30	57	43

Table 4: Human evaluation results

6 Discussion

To demonstrate the effectiveness of our visual attention mechanism, we show some qualitative examples in Figure 3. Each row contains three images that share similar semantic meaning, which are retrieved by querying the image caption using our learned shared space. The original caption of each image is shown below each image. We highlight the three words in each caption that have the highest weights assigned by the visual attention mechanism.

In the first row of Figure 3, the attention mechanism assigns high weights to the words “skiing", “snowboarding", and “snow". In the second row, it assigns high attention to “rafting" or “raft" for every caption of the three images. These examples demonstrate evidence that our attention mechanism learns to assign high weights to words that have corresponding visual semantics in the image.

We also find that our visual grounding attention captures the dependency between the words that

	Source Caption:	a tennis player is moving to the side and is gripping his racquet with both hands .
	Text-Only NMT:	ein tennisspieler bewegt sich um die seite und greift mit beiden händen an den boden .
	VAG-NMT:	ein tennisspieler bewegt sich zur seite und greift mit beiden händen den schläger .
	Source Caption:	three skiers skiing on a hill with two going down the hill and one moving up the hill .
	Text-Only NMT:	drei skifahrer fahren auf skiern einen hügel hinunter und eine person fährt den hügel hinunter .
	VAG-NMT:	drei skifahrer auf einem hügel fahren einen hügel hinunter und ein bewegt sich den hügel hinauf .
	Source Caption:	a blue , yellow and green surfboard sticking out of a sandy beach .
	Text-Only NMT:	ein blau , gelb und grünes surfbrett streckt aus einem sandstrand .
	VAG-NMT:	ein blau , gelb und grüner surfbrett springt aus einem sandstrand .

Figure 4: Translations generated by VAG-NMT and Text-Only NMT. VAG-NMT performs better in the first two examples, while Text-Only NMT performs better in the third example. We highlight the words that distinguish the two systems' results in red and blue. Red words are marked for better translation from VAG-NMT and blue words are marked for better translation from Text-Only NMT.

have strong visual semantic relatedness. For example, in Figure 3, words, such as “raft",“river", and “water", with high attention appear in the image together. This shows that the visual dependence information is encoded into the weighted sum of attention vectors which is applied to initialize the translation decoder. When we apply the sequence-to-sequence model to translate a long sentence, the encoded visual dependence information strengthens the connection between the words with visual semantic relatedness. Such connections mitigate the problem of standard sequence-to-sequence models tending to forget distant history. This hypothesis is supported by the fact that our VAG-NMT outperforms all the other methods on the IKEA dataset which has long sentences.

Lastly, in Figure 4 we provide some qualitative comparisons between the translations from VAG-NMT and Text-Only NMT. In the first example, our VAG-NMT properly translates the word "racquet" to “den schläger", while the Text-Only NMT mistranslated it to “den boden" which means “ground" in English. We suspect the attention mechanism and visual shared space capture the visual dependence between the word “tennis" and “racquet". In the second example, our VAG-NMT model correctly translates the preposition “up" to “hinauf" but Text-Only NMT mistranslates it to “hinunter" which means “down" in English. We consistently observe that VAG-NMT translates prepositions better than Text-Only NMT. We think it is because the pre-trained convnet features captured the relative object position that leads to a better preposition choice. Finally, in the third example, we show a failure case where Text-Only NMT generates a better translation. Our VAG-NMT mistranslates the verb phrase “sticking out" to “springt aus" which means “jump out" in German, while Text-Only NMT translates to “streckt aus", which is correct. We find that VAG-NMT often makes mistakes when translating verbs. We think it is because the image vectors are pre-trained on an object classification task, which does not have any human action information.

7 Conclusion and Future Work

We proposed a visual grounding attention structure to take advantage of the visual information to perform machine translation. The visual attention mechanism and visual context grounding module help to integrate the visual content into the sequence-to-sequence model, which leads to better translation quality compared to the model with only text information. We achieved state-of-the-art results on the Multi30K and Ambiguous COCO dataset. We also proposed a new product dataset, IKEA, to simulate a real-world online product description translation challenge.

In the future, we will continue exploring different methods to ground the visual context into the translation model, such as learning a multimodal shared space across image, source language text, as well as target language text. We also want to change the visual pre-training model from an image classification dataset to other datasets that have both objects and actions, to further improve translation performance.

Acknowledge

We would like to thank Daniel Boyla for providing insightful discussions to help with this research. We also want to thank Chunting Zhou and Ozan Caglayan for suggestions on machine translation model implementation. This work was supported in part by NSF CAREER IIS-1751206.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Ozan Caglayan, Walid Aransa, Adrien Bardet, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, Marc Masana, Luis Herranz, and Joost van de Weijer. 2017. LIUM-CVC submissions for WMT17 multimodal translation task. *CoRR*, abs/1707.04481.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017a. Doubly-attentive decoder for multi-modal neural machine translation. *CoRR*, abs/1702.01287.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017b. Incorporating global visual features into attention-based neural machine translation. *CoRR*, abs/1701.06521.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017c. Multilingual multi-modal embeddings for natural language processing. *CoRR*, abs/1702.01101.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Grzegorz Chrupala, Ákos Kádár, and Afra Alishahi. 2015. Learning language through pictures. *CoRR*, abs/1506.03694.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *In Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the second shared task on multimodal machine translation and multilingual image description. *CoRR*, abs/1710.07177.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. *CoRR*, abs/1605.00459.

Desmond Elliott and Ákos Kádár. 2017. Imagination improves multimodal translation. *CoRR*, abs/1705.04350.

Spandana Gella, Rico Sennrich, Frank Keller, and Mirella Lapata. 2017. Image pivoting for learning multilingual multimodal representations. *CoRR*, abs/1707.07601.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015a. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015b. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852.

Jindrich Helcl and Jindrich Libovický. 2017. CUNI system for the WMT17 multimodal translation task. *CoRR*, abs/1707.04550.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based multimodal neural machine translation. In *WMT*.

Andrej Karpathy and Fei-Fei Li. 2014. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Mingbo Ma, Dapeng Li, Kai Zhao, and Liang Huang. 2017. OSU multimodal machine translation system report. *CoRR*, abs/1710.02718.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 91–99, Cambridge, MA, USA. MIT Press.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. *CoRR*, abs/1703.04357.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. 2017. Nictnaist system for wmt17 multimodal translation task. In *WMT*.

A IKEA Dataset Stats and Examples

We summarize the statistics of our IKEA dataset in Figure 5, where we demonstrate the information about the number of tokens, the length of the product description and the vocabulary size. We provide one example of the IKEA dataset in Figure 6.

Pair	EN-DE		EN-FR	
Language	EN	DE	EN	FR
Tokens	256355	216892	239966	275251
Min length	6	6	6	6
Max length	343	324	334	469
Avg length	71.4	60.4	72.2	82.9
Std dev	46.3	39.1	47.2	54.7
Vocabulary	6601	10468	6442	7575

Figure 5: Statistic of the IKEA dataset

(a) *Product image*

the 4 large drawers on casters give you an extra storage space under the bed . adjustable bed sides allow you to use mattresses of different thicknesses . 17 slats of layer glued birch adjust to your body weight and increase the suppleness of the mattress . wipe clean using a damp cloth and a mild cleaner . wipe dry with a clean cloth.

(b) *Source description*

die 4 geräumigen schubladen auf rollen sorgen für zusätzlichen stauraum unter dem bett . durch verstellbare bettseiten können matratzen in verschiedenen stärken verwendet werden . mit feuchtem tuch (evtl . mit mildem reinigungsmittel) abwischen . mit trockenem tuch nachwischen .

(c) *Target description in German*

Figure 6: An example of product description and the corresponding translation in German from the **IKEA dataset**. Both descriptions provide an accurate caption for the commercial characteristics of the product in the image, but the details in the descriptions are different.

B Hyperparameter Settings

In this Appendix, we share details on the hyperparameter settings for our model and the training process. The word embedding size for both encoder and decoder are 256. The Encoder is a one-layer bidirectional recurrent neural network with Gated Recurrent Unit (GRU), which has a hidden size of 512. The decoder is a recurrent neural network with conditional GRU of the same hidden size. The visual representation is a 2048-dim vector extracted from the `pool5` layer of a pre-trained ResNet-50 network. The dimension of the shared visual-text semantic embedding space is 512. We set the decoder initialization weight value λ to 0.5.

During training, we use Adam (Kingma and Ba, 2014) to optimize our model with a learning rate of $4e-4$ for German Dataset and $1e-3$ for French dataset. The batch size is 32. The total gradient norm is clipped to 1 (Pascanu et al., 2012). Dropout is applied at the embedding layer in the encoder, context vectors extracted from the encoder and the output layer of the decoder. For Multi30K German dataset the dropout probabilities are $(0.3, 0.5, 0.5)$ and for Multi30K French dataset the dropout probabilities are $(0.2, 0.4, 0.4)$. For the Multimodal shared space learning objective function, the margin size γ is set to 0.1. The objective split weight α is set to 0.99. We initialize the weights of all the parameters with the method introduced in (He et al., 2015b).

C Ablation Analysis on Visual-Text Attention

We conducted an ablation test to further evaluate the effectiveness of our visual-text attention mechanism. We created two comparison experiments where we reduced the impact of visual-text attention with different design options. In the first experiment, we remove the visual-attention mechanism in our pipeline and simply use the mean of the encoder hidden states to learn the shared embedding space. In the second experiment, we initialize the decoder with just the mean of encoder hidden states without the weighted sum of encoder states using the learned visual-text attention scores.

We run both experiments on Multi30K German Dataset five times and demonstrate the results in table 5. As can be seen, the performance of the changed translation model is obviously worse than the full VAG-NMT in both experiments. This observation suggests that the visual-attention mechanism is critical in improving the translation performance. The model improvement comes from the attention mechanism influencing the model's objective function and decoder's initialization.

	English → German	
Method	BLEU	METEOR
-attention in shared embedding	30.5 ± 0.6	51.7 ± 0.4
-attention in initialization	30.8 ± 0.8	51.9 ± 0.5
VAG-NMT	$\mathbf{31.6 \pm 0.6}$	$\mathbf{52.2 \pm 0.3}$

Table 5: Ablation analysis on visual-text attention mechanism in the Multi30K German dataset.

Sentiment Classification towards Question-Answering with Hierarchical Matching Network

Chenlin Shen[1], **Changlong Sun**[2], **Jingjing Wang**[1], **Yangyang Kang**[2],
Shoushan Li[1], **Xiaozhong Liu**[2], **Luo Si**[2], **Min Zhang**[1], **Guodong Zhou**[1,*]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]Alibaba Group, Hangzhou, China
[1]`{chenlin.scl,djingwang}@gmail.com`
[1]`{lishoushan,minzhang,gdzhou}@suda.edu.cn`
[2]`changlong.scl@taobao.com`
[2]`{yangyang.kangyy,xiaozhong.lxz,luo.si}@alibaba-inc.com`

Abstract

In an e-commerce environment, user-oriented question-answering (QA) text pair could carry rich sentiment information. In this study, we propose a novel task/method to address QA sentiment analysis. In particular, we create a high-quality annotated corpus with specially-designed annotation guidelines for QA-style sentiment classification. On the basis, we propose a three-stage hierarchical matching network to explore deep sentiment information in a QA text pair. First, we segment both the question and answer text into sentences and construct a number of [Q-sentence, A-sentence] units in each QA text pair. Then, by leveraging a QA bidirectional matching layer, the proposed approach can learn the matching vectors of each [Q-sentence, A-sentence] unit. Finally, we characterize the importance of the generated matching vectors via a self-matching attention layer. Experimental results, comparing with a number of state-of-the-art baselines, demonstrate the impressive effectiveness of the proposed approach for QA-style sentiment classification.

1 Introduction

Sentiment analysis, a.k.a. opinion mining, is a task which aims to identify the user sentiment orientation of a product/brand/service by monitoring the online textual data, e.g., reviews and social media messages. It has attracted huge attention in both academic and industrial communities due to its widespread applications, such like recommendation (Zhang et al., 2014) and social media mining (Chambers et al., 2015). As the fundamental component in sentiment analysis, sentiment classification mainly classifies the sentiment polarity as *positive* or *negative*, and has been well-studied from both sentence-level (Kim and Hovy, 2004) and document-level (Xu et al., 2016).

Question 1: *Is the screen clear? How is the battery?*
Answer 1: *It's a nice phone with high-quality screen. But the battery is not durable.*

Question 2: *Summer is coming, I'm afraid of getting darker. Is the sun cream really effective?*
Answer 2: *No, just depending on my own experience.*

Figure 1: Two examples of QA text pairs from "*customer questions & answers*" section in *Amazon*.

Recently, a new QA-style reviewing form, namely "*customer questions & answers*", has become increasingly popular on the giant e-commerce platforms, e.g., *Amazon* and *Taobao*. In this new form, a potential customer asks question(s) about the target product/service while other experienced user(s) can provide answer(s). With the widespread of such QA-style reviews, users find a different channel to efficiently explore rich and useful information, and service providers and scholars are paying more attention to its specific characteristics comparing with traditional reviews (Wachsmuth et al., 2014; Zhou et al., 2015a). Comparing to the traditional reviews, the QA style reviews can be more informative and convincing. More importantly, because answer providers are randomly picked from the users who already purchased the target item, this new form of review can be more reliable and trustful.

Regarding QA-style sentiment analysis, one straightforward method is to directly employ an existing sentiment classification approach that works well on traditional reviews, such as RNN (Nguyen and Shirai, 2015) and LSTM (Chen et al., 2016). However, because of the significant differences between QA-style and classical reviews, existing review mining algorithms, e.g., text-based sentiment analysis/classification, should not be di-

---

[*] Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3654–3663
Brussels, Belgium, October 31 - November 4, 2018.

rectly applied to this new kind of QA-style data. More detailed reasons can be found as the followings.

First, in QA-style text, the question and answer text are more likely to be two parallel units rather than a sequence form. On the one hand, for instance, in Figure 1, sentence "*It's a nice phone with high-quality screen.*" in **Answer 1** actually does not follow sentence "*How is the battery?*" in **Question 1** , but corresponds to sentence "*Is the screen clear?*" in **Question 1**. Therefore, when the question text and answer text are presented as two units in a sequence, it is rather difficult to capture the relationship between the question and its corresponding answer due to the possible long distance between them. On the other hand, there often exists both *positive* and *negative* sentiments in answer text according to different parts of question, and this specific case should be categorized as another category named *conflict*. For instance, in Figure 1, **Answer 1** "*It's a nice phone with high-quality screen. But the battery is not durable.*" is a *conflict* answer to **Question 1**. However, when this answer text is considered as a sequence, it is highly possible to be predicted as the category of *positive* or *negative* rather than *conflict*. In order to address these problems, a more appropriate approach is to segment both the question and answer text into some parallel sentences, and then construct the [Q-sentence, A-sentence] units in each QA text pair to detect in-depth sentiment information.

Second, although the main sentiment polarity is usually expressed from the answer text, the question text could also carry important sentiment tips to predict the sentiment polarity of a QA text pair. For instance, in Figure 1, we could hardly estimate the sentiment polarity solely based on **Answer 2**. However, when we take **Question 2**, "*Is the sun cream really effective?*", into consideration, it can be easier to label this QA text pair with a *negative* tag. In this study, we propose an approach to match the sentences inside the question and answer text bidirectionally.

Third, in each QA text pair, the importance degrees of different [Q-sentence, A-sentence] units can be different. For instance, in Figure 1, the [Q-sentence, A-sentence] unit, i.e., sentence "*Summer is coming, I'm afraid of getting darker.*" in **Answer 2** and sentence "*No, just depending on my own experience.*" in **Question 2**, makes tiny contribution to imply the sentiment polarity for the QA text pair. Therefore, a well-behaved network approach should consider the importance degrees of different [Q-sentence, A-sentence] units for predicting the sentiment polarity of a QA text pair.

The contribution of this paper is twofold. First, we propose a novel problem, QA-style sentiment analysis, and build a large-scale annotated corpus tailed for this task. The dataset is released to motivate future investigations for this track of research. Second, we propose a hierarchical matching network model to address the challenges of QA-style sentiment classification. Specifically, we first segment both the question and answer text into sentences and construct the [Q-sentence, A-sentence] units for each QA text pair. Then, by using a QA bidirectional matching layer, we encode each [Q-sentence, A-sentence] unit for exploring sentiment information. Finally, the self-matching attention layer in the model can capture the importance of these [Q-sentence, A-sentence] matching vectors obtained from QA bidirectional matching layer, which could effectively refine the evidence for inferring the sentiment polarity of a QA text pair. Experimental results show that the proposed approach significantly outperforms several strong baselines for QA-style sentiment classification.

2 Related Work

Sentiment classification has become a hot research field in NLP since the pioneering work by Pang et al. (2002). In general, the research on traditional sentiment classification has been carried out in different text levels, such like word-level, document-level and aspect-level.

Word-level sentiment classification has been studied in a long period in the research community of sentiment analysis. Some early studies have devoted their efforts to predicting the sentiment polarity of a word with different learning models and resources. Turney (2002) proposed an approach to predicting the sentiment polarity of words by calculating Pointwise Mutual Information (PMI) values between the seed words and the search hits. Hassan and Radev (2010) and Hassan et al. (2011) applied a Markov random walk model to determine the word polarities with a large word relatedness graph, and the synonyms and hypernyms in WordNet (Miller, 1995). More recently, some studies aim to learn better word embedding of a word rather than its polarity. Tang et al. (2014) developed three neural networks to learn word em-

	Positive	*Negative*	*Conflict*	*Neutral*	*Total*
Beauty	3,676	981	318	5,025	10,000
Shoe	4,025	819	412	4,744	10,000
Electronic	3,807	1,017	528	4,648	10,000

Table 1: Category distribution of the annotated data in three domains.

bedding by incorporating sentiment polarities of text in loss functions. Zhou et al. (2015b) employed both unsupervised and supervised neural networks to learn bilingual sentiment word embedding.

Document-level sentiment classification has also been studied in a long period in the research community of sentiment analysis. On one hand, many early studies have been devoted their efforts to various of aspects on learning approaches, such as supervised learning (Pang et al., 2002; Riloff et al., 2006), semi-supervised learning (Li et al., 2010; Xia et al., 2015; Li et al., 2015), and domain adaptation (Blitzer et al., 2007; He et al., 2011). On the other hand, many recent studies employ deep learning approaches to enhance the performances in sentiment classification. Tang et al. (2015) proposed a user-product neural network to incorporate both user and product information for sentiment classification. Xu et al. (2016) proposed a Cached Long Short-Term Memory neural networks (CLSTM) to capture the overall semantic information in long texts. More recently, Long et al. (2017) proposed a novel attention model, namely cognition-based attention, for sentiment classification.

Aspect-level sentiment classification is a relatively new research area in the research community of sentiment analysis and it is a fine-grained classification task. Recently, Wang et al. (2016) proposed an attention-based LSTM neural network to aspect-level sentiment classification by exploring the connection between an aspect and the content of a sentence. Tang et al. (2016) proposed a deep memory network with multiple attention-based computational layers to improve the performance. Wang et al. (2018) proposed a hierarchical attention network to explore both word-level and clause-level sentiment information towards a target aspect.

Unlike all the prior studies, this paper focuses on a very different kind of text representation, i.e., QA-style text level, for sentiment classification. To the best of our knowledge, this is the first attempt to perform sentiment classification on this text level.

3 Data Collection and Annotation

We collect QA text pairs from "*Asking All*" in *Taobao* (*Alibaba*)[1], which is the world's biggest e-commerce company. The QA text pairs are mainly from *Beauty*, *Shoe* and *Electronic* domains and each domain contains 10,000 QA text pairs.

We define four sentiment-related categories, i.e., *positive*, *negative*, *conflict* (both *positive* and *negative* sentiment) and *neutral* (neither *positive* nor *negative* sentiment). To guarantee a high annotation agreement, we propose some annotation guidelines after several times of annotation processes on a small size of data. Then, we ask more coders to annotate the whole data set according to these annotation guidelines.

The annotation guidelines contain two main groups. One contains the guidelines which aim to distinguish the categories of *neutral* and *non-neutral*, i.e.,

(a) A QA text pair in which the question and the answer do not match is annotated as a *neutral* sample. In this type of samples, the answer does not reply to the question correctly. **E1** is an example of this type where the question talks about the screen while the answer talks about the battery.

E1: Q: *Is the screen clear?*
A: *The battery life is decent.*

(b) A QA text pair with an unknown or uncertain answer is annotated as a *neutral* sample. **E2** is an example of this type.

E2: Q: *What about these sneakers?*
A: *I don't know, I bought it for my dad.*

(c) A QA text pair with only objective description is annotated as a *neutral* sample. **E3** is an example of this type.

E3: Q: *What's the operation system of the phone?*
A: *Android.*

(d) A QA text pair which compares two different products is annotated as a *neutral* sample. In this type of samples, two products are involved and it

---

[1] https://www.taobao.com/

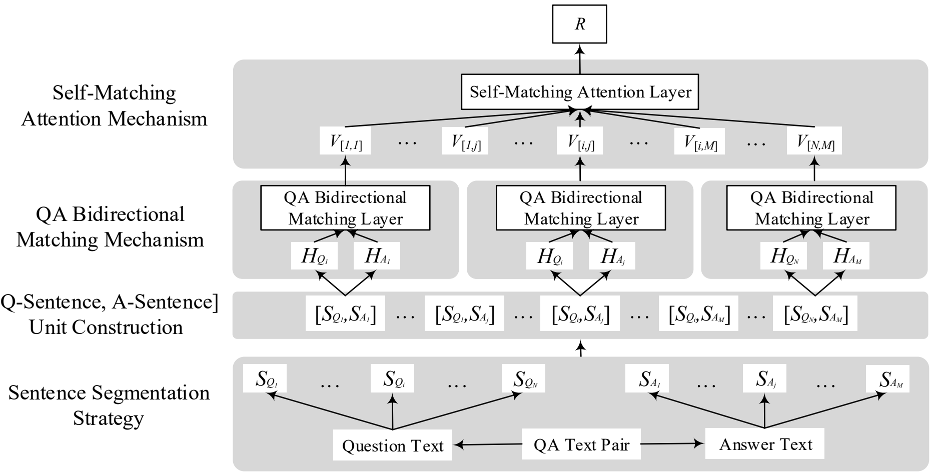

Figure 2: The overview of our approach to QA-style sentiment classification where S_{Q_i} denotes the i-th sentence in question text, S_{A_j} denotes the j-th sentence in answer text, H_{Q_i} and H_{A_j} denote the contextual representations for S_{Q_i} and S_{A_j} respectively, $V_{[i,j]}$ denotes the bidirectional matching vector for $[S_{Q_i}, S_{A_j}]$ unit through QA bidirectional matching layer, and R is the QA text pair representation refined by self-matching attention layer.

is sometimes difficult to tell the sentiment orientation of one product. **E4** is an example of this type.

E4: Q: *How about this phone when compared to iPhone 6s?*
A: *It's up to you, and they're not comparable.*

The other group contains the guidelines which aim to distinguish the categories of *positive* and *negative*, i.e.,
(e) If the answer text contains sentimental expressions to question like "*disappointed*", "*terrible*", and so on, we annotate it as *negative*. **E5** is an example of this type.

E5: Q: *How is the rock climbing shoe?*
A: *I am so disappointed, my feet felt hurt when I wore them.*

(f) If the answer text contains sentimental expressions to question like "*perfect*", "*satisfied*", and so on, we annotate it as *positive*. **E6** is an example of this type.

E6: Q: *How about the fragrance?*
A: *I am so satisfied, it smells distinctive.*

(g) If we cannot confirm the polarity of a QA text pair only depending on answer text, we annotate the polarity according to both the question and answer text. For instance, **E7** is an example with *positive* polarity, while **E8** is an example with *negative* polarity.

E7: Q: *Will the phone get hot when gaming?*
A: *No.*

E8: Q: *Is the sun cream really economic?*
A: *No.*

We assign two annotators to annotate each QA text pair, and the *Kappa* consistency check value of the annotation is 0.84. When annotators cannot reach an agreement, an expert will make the final decision, ensuring the quality of data annotation. Table 1 shows the category distribution of the corpus. To motivate other scholars to investigate this novel but important task, we share the data via Github[2].

4 Methodology

In this section, we introduce the proposed hierarchical matching network approach for QA-style sentiment classification. Figure 2 depicts the overview of the proposed approach.

4.1 QA Bidirectional Matching Mechanism

Word Encoding Layer: After sentence segmentation, the question text in a QA text pair contains N sentences, S_{Q_i} represents the i-th sentence in the question text. Similarly, the answer text in this QA text pair contains M sentences, S_{A_j} represents the j-th sentence in the answer text. We then construct [Q-sentence, A-sentence] units by pairing one sentence in the question text and one sentence in the answer text, and we obtain $N{*}M$ [Q-sentence, A-sentence] units at last.

Given a $[S_{Q_i}, S_{A_j}]$ unit in this QA text pair, i.e., Q-sentence S_{Q_i} with words $w_{i,n}, i \in [1, N], n \in$

---

[2] https://github.com/clshenNLP/QASC/

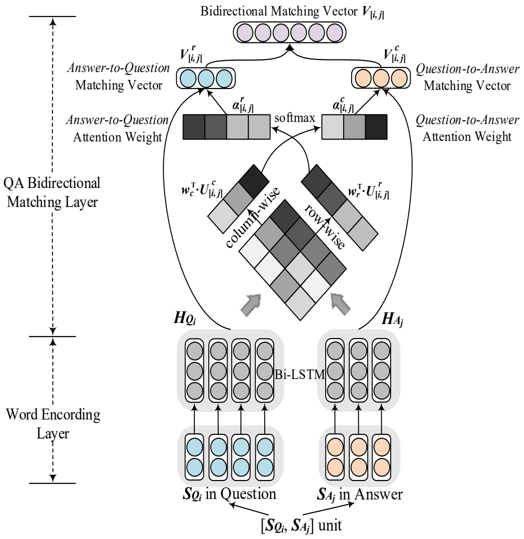

Figure 3: The detail architecture of QA bidirectional matching mechanism.

$[1, N_i]$ and A-sentence S_{A_j} with words $w_{j,m}, j \in [1, M], m \in [1, M_j]$, we first convert the words to their respective word embeddings ($x_{i,n} \in \mathbb{R}^d, i \in [1, N], n \in [1, N_i]$ and $x_{j,m}, j \in [1, M], m \in [1, M_j]$). We then use Bi-directional LSTM (namely Bi-LSTM), which can efficiently make use of past features (via forward states) and future features (via backward states) for a specific time step, to get contextual representations of S_{Q_i} and S_{A_j} individually. The representation of each word is formed by concatenating the forward and backward hidden states. For simplicity, we note contextual representation of S_{Q_i} as H_{Q_i} , and contextual representation of S_{A_j} as H_{A_j} respectively:

$$H_{Q_i} = [h_{i,1}, h_{i,2}, ..., h_{i,n}, ..., h_{i,N_i}] \tag{1}$$

$$H_{A_j} = [h_{j,1}, h_{j,2}, ..., h_{j,m}, ..., h_{j,M_j}] \tag{2}$$

where $h_{i,n} \in \mathbb{R}^{d'}$ denotes the word representation in S_{Q_i} at time step n, $h_{j,m} \in \mathbb{R}^{d'}$ denotes the word representation in S_{A_j} at time step m, and d' is the dimensionality of word representation.

QA Bidirectional Matching Layer: General neural network could not capture sentiment matching information in a $[S_{Q_i}, S_{A_j}]$ unit well. For the sake of solving this problem, we introduce the QA bidirectional matching layer to encapsulate the clues and interactions between S_{Q_i} and S_{A_j} synchronously (Tay et al., 2017; McCann et al., 2017). Figure 3 depicts the detail architecture of QA bidirectional matching mechanism. Specifically, we first calculate the bidirectional pair-wise matching matrix by using the following formula:

$$D_{[i,j]} = (H_{Q_i})^{\top} \cdot (H_{A_j}) \tag{3}$$

where $D_{[i,j]} \in \mathbb{R}^{N_i \times M_j}$ denotes the bidirectional matching matrix for the $[S_{Q_i}, S_{A_j}]$ unit. Each element in $D_{[i,j]}$ is the score that measures how well the word in S_{Q_i} semantically matches the word in S_{A_j} and vice versa.

Given the bidirectional matching matrix $D_{[i,j]}$, we use attention mechanism (Yang et al., 2016; Cui et al., 2017) to mine the sentiment matching information between question and answer from two directions, which could be seen as an *Answer-to-Question* attention and a *Question-to-Answer* attention as follows.

- ***Answer-to-Question*** **Attention:** We employ row-wise operations to compute the attention weight vector $\alpha^r_{[i,j]}$ as follows:

$$U^r_{[i,j]} = \tanh(W_r \cdot D^{\top}_{[i,j]}) \tag{4}$$

$$\alpha^r_{[i,j]} = \text{softmax}(w_r^{\top} \cdot U^r_{[i,j]}) \tag{5}$$

where $\alpha^r_{[i,j]} \in \mathbb{R}^{N_i}$ is the *Answer-to-Question* attention weight vector regarding the importance degrees of all words in Q-sentence S_{Q_i}, $W_r \in \mathbb{R}^{d' \times M_j}$ and $w_r \in \mathbb{R}^{d'}$ are weight matrices. After computing the *Answer-to-Question* attention weight vector, we can get the *Answer-to-Question* matching vector $V^r_{[i,j]} \in \mathbb{R}^{d'}$ as follows:

$$V^r_{[i,j]} = (H_{Q_i}) \cdot \alpha^r_{[i,j]} \tag{6}$$

- ***Question-to-Answer*** **Attention:** Simultaneously, we employ column-wise operations to calculate the attention weight vector $\alpha^c_{[i,j]}$ as follows:

$$U^c_{[i,j]} = \tanh(W_c \cdot D_{[i,j]}) \tag{7}$$

$$\alpha^c_{[i,j]} = \text{softmax}(w_c^{\top} \cdot U^c_{[i,j]}) \tag{8}$$

where $\alpha^c_{[i,j]} \in \mathbb{R}^{M_j}$ is the *Question-to-Answer* attention weight vector regarding the importance degrees of all words in A-sentence S_{A_j}, $W_c \in \mathbb{R}^{d' \times N_i}$ and $w_c \in \mathbb{R}^{d'}$ are weight matrices. After calculating the *Question-to-Answer* attention weight vector, we can get the *Question-to-Answer* matching vector $V^c_{[i,j]} \in \mathbb{R}^{d'}$ as follows:

$$V^c_{[i,j]} = (H_{A_j}) \cdot \alpha^c_{[i,j]} \tag{9}$$

Then, we combine *Answer-to-Question* and *Question-to-Answer* matching vectors to represent the final bidirectional matching vector of the $[S_{Q_i}, S_{A_j}]$ unit:

$$V_{[i,j]} = V^r_{[i,j]} \oplus V^c_{[i,j]} \tag{10}$$

where $\oplus$ denotes the concatenate operator, and $V_{[i,j]}$ denotes the bidirectional matching vector which integrates S_{Q_i} and S_{A_j} with each other.

4.2 Self-Matching Attention Mechanism

Through the QA bidirectional matching layer, informative bidirectional matching vectors are generated to pinpoint the sentiment matching information in each [Q-sentence, A-sentence] unit. Intuitively, each matching vector for [Q-sentence, A-sentence] unit holds different importance to a QA text pair. To better aggregate the evidence from these vectors for inferring the sentiment polarity of the QA text pair, we propose a self-matching attention layer, matching these informative vectors against themselves.

Self-Matching Attention Layer: As aforementioned, we have obtained $N*M$ bidirectional matching vectors through QA bidirectional matching layer, then we calculate the attention weight vector α with these matching vectors by following formulas:

$$V = [V_{[1,1]}, V_{[1,2]}, ..., V_{[i,j]}, ..., V_{[N,M]}] \tag{11}$$

$$U = \tanh(W_h \cdot V) \tag{12}$$

$$\alpha = \mathrm{softmax}(w_h^\top \cdot U) \tag{13}$$

where α is the attention weight vector which measures the importance of these matching vectors, W_h and w_h are the weight matrices.

Finally, we can get the QA text pair representation R as follows:

$$R = V \cdot \alpha \tag{14}$$

4.3 Classification Model

QA text pair representation R is a high level representation which can be used for classification. In our approach, we feed R to a *softmax* classifier:

$$p = \mathrm{softmax}(W_l \cdot R + b_l) \tag{15}$$

where p is a set of predicted distribution of the sentiment categories, i.e., *positive*, *negative*, *neutral*, and *conflict*. W_l is the weight matrix and b_l is the bias.

To learn the whole model, we train an end-to-end model given the training data, and the goal of training is to minimize the cross-entropy loss, i.e.,

$$L(\theta) = -\sum_{s=1}^{S}\sum_{k=1}^{K} y_s^k \cdot \log\hat{y}_s^k + \lambda\|\theta\|_2^2 \tag{16}$$

where S is the number of training data. y_s is the true sentiment label of the s-th sample. $\hat{y}_s$ is the predicted sentiment label of the s-th sample. K is number of all sentiment categories. λ is a L_2-regularization term, θ is the parameter set. In the above equation, the model parameters are optimized by using Adam (Kingma and Ba, 2014).

5 Experimentation

In this section, we evaluate the performances of the proposed approach for QA-style sentiment classification.

5.1 Experimental Settings

- **Data Sets:** As introduced in Section 3, the annotated QA text pairs cover three different domains. In each domain, we randomly split the data into a training set (80% in each category) and a test set (20% in each category). In addition, we set aside 10% from the training set as the development data for parameters tuning.
- **Word Segmentation and Embeddings:** FudanNLP[3] (Qiu et al., 2013) is employed to segment text into Chinese words and word2vec[4] (Mikolov et al., 2013) is employed to pre-train word embeddings. The vector dimensionality is set to be 100.
- **Sentence Segmentation:** CoreNLP[5] (Manning et al., 2014) is employed to segment both the question and answer text into sentences.
- **Hyper-parameters:** In the experiment, all out-of-vocabulary words are initialized by sampling from the uniform distribution $U(-0.01, 0.01)$. All weight matrices are given their initial values by sampling from uniform distribution $U(-0.01, 0.01)$. The LSTM hidden states are set to be 128 and all models are trained by mini-batch of 32 instances. The dropout rate is set to 0.2. The other hyper-parameters are tuned according to the development data.
- **Evaluation Metric:** The performance is evaluated using standard *Accuracy* and *Macro-F1*.

---

[3] https://github.com/FudanNLP/fnlp/

[4] https://code.google.com/archive/p/word2vec/

[5] http://stanfordnlp.github.io/CoreNLP/

	Beauty		*Shoe*		*Electronic*	
	Macro-F1	*Accuracy*	*Macro-F1*	*Accuracy*	*Macro-F1*	*Accuracy*
SVM	0.362	0.684	0.381	0.718	0.435	0.691
LSTM	0.499	0.712	0.520	0.754	0.562	0.715
Bi-LSTM	0.527	0.719	0.531	0.759	0.574	0.723
Bidirectional-Match	0.526	0.747	0.557	0.796	0.582	0.741
AtoQ-Match	0.543	0.745	0.602	0.792	0.567	0.754
QtoA-Match	0.573	0.751	0.647	0.807	0.608	0.752
Bidirectional-Match QA	0.583	0.760	0.666	0.815	0.617	0.764
HMN	**0.598**	**0.776**	**0.683**	**0.827**	**0.640**	**0.779**

Table 2: Performance of our approaches to QA-style sentiment classification in all domains.

5.2 Experimental Results

The following baseline approaches are employed for comparison. Note that all the approaches share the same word embeddings for fair comparison.

- **SVM:** This baseline employs support vector machine along with word embedding features. The question and answer text in a QA text pair are chained as a sequence.
- **LSTM:** A standard LSTM model utilizes word embeddings and concatenates the question and answer text as a sequence.
- **Bi-LSTM:** A bidirectional LSTM model which concatenates the question and answer text as a sequence.
- **Bidirectional-Match:** This approach employs QA bidirectional matching mechanism, without taking the sentence segmentation strategy and self-matching attention mechanism.
- **AtoQ-Match:** This approach takes the sentence segmentation strategy, and employs QA unidirectional matching mechanism (i.e., only using *Answer-to-Question* attention), but does not employ self-matching attention mechanism. We average the *Answer-to-Question* matching vectors to represent the QA text pair.
- **QtoA-Match:** This approach takes the sentence segmentation strategy, and employs QA unidirectional matching mechanism (i.e., only using *Question-to-Answer* attention), but does not employ self-matching attention mechanism.
- **Bidirectional-Match QA:** This approach takes the sentence segmentation strategy, and employs QA bidirectional matching mechanism, but does not employ self-matching attention mechanism.
- **HMN:** This is our hierarchical matching network model which takes the sentence segmentation strategy and employs both QA bidirectional matching mechanism and self-matching attention mechanism.

Table 2 summarizes the experimental results of all the approaches above, and we can find that:

(1) All LSTM-based approaches are superior to **SVM**, indicating the effectiveness of neural network for this task.

(2) The proposed approaches, with novel QA contextual representation, outperform the other baseline approaches.

(3) When only employing QA bidirectional matching mechanism, **Bidirectional-Match QA**, which takes the sentence segmentation strategy, consistently outperforms **Bidirectional-Match** (without sentence segmentation) in all domains. It confirms our hypothesis that sentence segmentation helps to extract the sentiment matching information between the question and answer.

(4) When comparing to QA unidirectional matching mechanism, **Bidirectional-Match QA**, which employs QA bidirectional matching mechanism, performs better than **AtoQ-Match** and **QtoA-Match**. It confirms our hypothesis that both the question and answer information contribute to sentiment polarity of the QA text pair.

(5) Impressively, the proposed approach **HMN** significantly outperforms all the other approaches in all domains (p-value$<$0.05 via t-test). It verifies the advantages of both QA bidirectional matching mechanism and self-matching attention mechanism for this task.

Besides, we also implement some more recent state-of-the-art approaches for sentiment classification, which are illustrated in Table 3. This result also supports the earlier findings.

- **CNN-Tensor** (Lei et al., 2015)**:** This is a state-of-the-art approach to sentence-level sentiment classification, which models n-gram interactions based on tensor product and evaluates all non-

	Beauty		*Shoe*		*Electronic*	
	Macro-F1	*Accuracy*	*Macro-F1*	*Accuracy*	*Macro-F1*	*Accuracy*
CNN-Tensor	0.500	0.731	0.535	0.765	0.576	0.734
Attention-LSTM	0.509	0.725	0.571	0.755	0.576	0.721
BiMPM	0.553	0.745	0.587	0.766	0.584	0.746
HMN	**0.598**	**0.776**	**0.683**	**0.827**	**0.640**	**0.779**

Table 3: The proposed approach vs. several strong baseline approaches in all domains.

E9: Domain: *Beauty* True Label: *neutral*			
Q: 请问各位朋友们，淡斑效果怎么样？谢谢！ (*Hey, friends, how about the spot-fading of this product? Thanks a lot!*) **A**: 讲真哦，保湿还不错！ (*To tell you the truth, it can moisturize effectively!*)			
CNN-Tensor	Attention-LSTM	BiMPM	HMN
✗ (*positive*)	✗ (*positive*)	✓ (*neutral*)	✓ (*neutral*)

E10: Domain: *Electronic* True Label: *conflict*			
Q: 这款笔记本怎么样呢？电池耐用吗？玩大型游戏卡不卡？ (*How about this notebook? Is the battery durable? Does the OS run fast when playing games?*) **A**: 电池不是很耐用。玩大型游戏有点卡。其他的都还好。 (*Battery isn't much durable. The OS doesn't run fast when playing games. The other aspects are good.*)			
CNN-Tensor	Attention-LSTM	BiMPM	HMN
✗ (*positive*)	✗ (*negative*)	✗ (*negative*)	✓ (*conflict*)

Table 4: Some examples in the test data with their predicted categories by some approaches where ✗ (or ✓) means that the predicted category is wrong (or correct).

consecutive *n*-gram vectors as a feature mapping operator for CNNs.

- **Attention-LSTM** (Wang et al., 2016)**:** This is a state-of-the-art approach to aspect-level sentiment classification. In our implementation, we ignore the aspect embedding and directly use the outputs of LSTM to yield the attention.
- **BiMPM** (Wang et al., 2017)**:** This is a state-of-the-art approach to QA matching, which matches the question and answer from multiple perspectives. In our implementation, we use the matching representation to perform QA-style sentiment classification with a *softmax* classifier.
- **HMN:** The proposed hierarchical matching network which employs both QA bidirectional matching mechanism and self-matching attention mechanism, and takes the sentence segmentation strategy.

Table 3 shows the comparison results of these strong baseline approaches and the proposed approach (**HMN**) in all domains. From this table, we can find that: (1) the approaches that take matching strategy, i.e., **BiMPM** and our approach **(HMN)**, outperform other approaches. (2) The proposed approach **(HMN)** significantly outperforms all the other baseline approaches in terms of both *Macro-F1* and *Accuracy* (p-value$<$0.05 via t-test), which confirms the initial hypotheses of this study.

E11: True Label: *negative*
Q: 这个手机的系统顺畅不？电池耐用吗？
(*Does the OS run fast? Is the battery durable?*)
A: 电池一点也不耐用。相机也一般。
(*The battery is not durable at all. The camera is also not good.*)

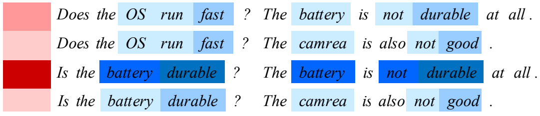

Figure 4: The attention visualizations for a QA text pair.

5.3 Case Study

Table 4 shows some examples, along with the predicted categories via different approaches. We can find that: (1) the approaches based on matching strategy (**BiMPM** and **HMN**) are well-performed, as shown in **E9**, when question and answer carrying different kinds of information. This is a unique challenge for QA-style sentiment mining, and traditional sentiment classification approaches can hardly address this problem. (2) The proposed approach **(HMN)** performs better than other approaches when dealing with *conflict* instances, as shown in **E10**.

5.4 Visualization of Attention

To get a better understanding of our proposed hierarchical matching network for QA-style sentiment classification, we picture the attention weights obtained from Equations (5), (8) and (13). For simplicity, we directly use the English translation of **E11** for illustration and adopt the visualization approach presented by Yang et al. (2016), as shown in Figure 2. Specifically, each line is a [Q-sentence, A-sentence] unit, where the red denotes the [Q-sentence, A-sentence] unit weight, the blue denotes the word weight in each [Q-sentence, A-sentence], and the color depth indicates the importance of attention weights (the darker the more important).

From Figure 4, we can see that the QA bidirectional matching layer always assigns reasonable attention weights to words in each [Q-sentence, A-sentence] unit which makes sentence from question and sentence from answer match correctly. In addition, the self-matching attention layer is able to select informative [Q-sentence, A-sentence] unit for predicting true sentiment polarity of this example.

6 Conclusion

In this paper, we propose a novel but important sentiment analysis task, i.e., QA-style sentiment mining, and we build a large-scale high-quality human annotated corpus for experiment. The dataset is shared to encourage other scholars to investigate this interesting problem. Moreover, we propose a hierarchical matching neural network model to enable QA bidirectional matching mechanism and self-matching attention mechanism for this task. Empirical studies show that the proposed approach significantly outperforms other strong baseline approaches in all the test domains for QA-style sentiment classification.

In the future, we would like to investigate some other network structures to explore deeper information in each QA text pair. Besides, we would like to test the effectiveness of the proposed approach to QA-style sentiment classification in some other languages.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work is partially supported by the National Key R&D Program of China under Grant No.2017YFB1002101 and two NSFC grants No.61331011, No.61672366. This work is also supported by the joint research project of Alibaba Group and Soochow University.

References

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL-2007*, pages 440–447.

Nathanael Chambers, Victor Bowen, Ethan Genco, Xisen Tian, Eric Young, Ganesh Harihara, and Eugene Yang. 2015. Identifying political sentiment between nation states with social media. In *Proceedings of EMNLP-2015*, pages 65–75.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Enhancing and combining sequential and tree LSTM for natural language inference. *arXiv preprint arXiv:1609.06038*.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of ACL-2017*, pages 593–602.

Ahmed Hassan, Amjad Abu-Jbara, Rahul Jha, and Dragomir Radev. 2011. Identifying the semantic orientation of foreign words. In *Proceedings of ACL-2011*, pages 592–597.

Ahmed Hassan and Dragomir Radev. 2010. Identifying text polarity using random walks. In *Proceedings of ACL-2010*, pages 395–403.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *Proceedings of ACL-2011*, pages 123–131.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING-2004*, pages 1367–1374.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding CNNs for text: non-linear, non-consecutive convolutions. In *Proceedings of EMNLP-2015*, pages 1565–1575.

Shoushan Li, Chu-Ren Huang, Guodong Zhou, and Sophia Yat Mei Lee. 2010. Employing personal/impersonal views in supervised and semi-supervised sentiment classification. In *Proceedings of ACL-2010*, pages 414–423.

Shoushan Li, Lei Huang, Jingjing Wang, and Guodong Zhou. 2015. Semi-stacking for semi-supervised sentiment classification. In *Proceedings of ACL-IJCNLP-2015*, pages 27–31.

Yunfei Long, Lu Qin, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017. A cognition based attention model for sentiment analysis. In *Proceedings of EMNLP-2017*, pages 462–471.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL-2014*, pages 55–60.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of NIPS-2017*, pages 6294–6305.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS-2013*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of EMNLP-2015*, pages 2509–2514.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of ACL-2002*, pages 79–86.

Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. FudanNLP: A toolkit for Chinese natural language processing. In *Proceedings of ACL-2013*, pages 49–54.

Ellen Riloff, Siddharth Patwardhan, and Janyce Wiebe. 2006. Feature subsumption for opinion analysis. In *Proceedings of EMNLP-2006*, pages 440–448.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of ACL-IJCNLP-2015*, pages 1014–1023.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of EMNLP-2016*, pages 214–224.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of ACL-2014*, pages 1555–1565.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL-2002*, pages 417–424.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, and Gregor Engels. 2014. Modeling review argumentation for robust sentiment analysis. In *Proceedings of COLING-2014*, pages 553–564.

Jingjing Wang, Jie Li, Shoushan Li, Yangyang Kang, Min Zhang, Luo Si, and Guodong Zhou. 2018. Aspect sentiment classification with both word-level and clause-level attention networks. In *Proceedings of IJCAI-2018*, pages 4439–4445.

Yequan Wang, Minlie Huang, Li Zhao, and Zhu Xiaoyan. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of EMNLP-2016*, pages 606–615.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of IJCAI-2017*, pages 4144–4150.

Rui Xia, Cheng Wang, Xin-Yu Dai, and Tao Li. 2015. Co-training for semi-supervised sentiment classification based on dual-view bags-of-words representation. In *Proceedings of ACL-IJCNLP-2015*, pages 1054–1063.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. In *Proceedings of EMNLP-2016*, pages 1660–1669.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT-2016*, pages 1480–1489.

Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of SIGIR-2014*, pages 83–92.

Guangyou Zhou, Tingting He, Jun Zhao, and Wensheng Wu. 2015a. A subspace learning framework for cross-lingual sentiment classification with partial parallel data. In *Proceedings of IJCAI-2015*, pages 1426–1433.

Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015b. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of ACL-IJCNLP-2015*, pages 430–440.

Cross-topic Argument Mining from Heterogeneous Sources

Christian Stab[*] and **Tristan Miller**[*†] and **Benjamin Schiller**[*] and
Pranav Rai[*] and **Iryna Gurevych**[*†]

[*]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
https://www.ukp.tu-darmstadt.de/

[†]Research Training Group AIPHES
Department of Computer Science, Technische Universität Darmstadt
https://www.aiphes.tu-darmstadt.de/

Abstract

Argument mining is a core technology for automating argument search in large document collections. Despite its usefulness for this task, most current approaches are designed for use only with specific text types and fall short when applied to heterogeneous texts. In this paper, we propose a new sentential annotation scheme that is reliably applicable by crowd workers to arbitrary Web texts. We source annotations for over 25,000 instances covering eight controversial topics. We show that integrating topic information into bidirectional long short-term memory networks outperforms vanilla BiLSTMs by more than 3 percentage points in F_1 in two- and three-label cross-topic settings. We also show that these results can be further improved by leveraging additional data for topic relevance using multi-task learning.

1 Introduction

Information retrieval and question answering are by now mature technologies that excel at answering factual queries on noncontroversial topics. However, they provide no specialized support for queries where there is no single canonical answer, as with topics that are controversial or opinion-based. For such queries, the user may need to carefully assess the stance, source, and supportability for each of the answers. These processes can be supported by argument mining (AM), a nascent area of natural language processing concerned with the automatic recognition and interpretation of arguments.

In this paper, we apply AM to the task of *argument search*—that is, searching a large document collection for arguments relevant to a given topic. Searching for and classifying relevant arguments plays an important role in decision making (Svenson, 1979), legal reasoning (Wyner et al., 2010), and the critical reading, writing, and summarization of persuasive texts (Kobayashi, 2009; Wingate, 2012). Automating the argument search process could ease much of the manual effort involved in these tasks, particularly if it can be made to robustly handle arguments from different text types and topics.

But despite its obvious usefulness, this sort of argument search has attracted little attention in the research community. This may be due in part to the limitations of the underlying models and training resources, particularly as they relate to heterogeneous sources. That is, most current approaches to AM are designed for use with particular text types, faring poorly when applied to new data (Daxenberger et al., 2017). Indeed, as Habernal et al. (2014) observe, while there is a great diversity of perspectives on how arguments can be best characterized and modelled, there is no “one-size-fits-all” argumentation theory that applies to the variety of text sources found on the Web.

To approach these challenges, we propose the novel task of topic-based sentential argument mining. Our contributions are as follows: (1) We propose a new argument annotation scheme applicable to the information-seeking perspective of argument search. We show it to be general enough for use on heterogeneous data sources, and simple enough to be applied manually by untrained annotators at a reasonable cost. (2) We introduce a novel corpus of heterogeneous text types annotated with topic-based arguments.[1] The corpus includes over 25,000 instances covering eight controversial topics. This is the first known resource that can be used to evaluate the performance of argument mining methods across topics in heterogeneous sources. (3) We investigate different approaches for incorporating topic information into neural networks and

---

[1] https://www.ukp.tu-darmstadt.de/sent_am

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3664–3674
Brussels, Belgium, October 31 - November 4, 2018.

show that including the topic vector into the ***i***- and ***c***-gates of the LSTM cell outperforms common attention-based approaches in two- and three-label cross-topic experiments. (4) We further improve the performance of the modified LSTM cell by leveraging additional data for topic relevance in a multi-task learning setup. (5) In the more challenging setup of cross-topic experiments, we show that our models yield considerably better performance than common BiLSTM models when little data of the target topic is available.

2 Related work

Most existing approaches treat argument mining at the discourse level, focusing on tasks such as segmenting argumentative discourse units (Ajjour et al., 2017; Goudas et al., 2014), classifying the function of argumentative discourse units (for example, as *claims* or *premises*) (Mochales-Palau and Moens, 2009; Stab and Gurevych, 2014), and recognizing argumentative discourse relations (Eger et al., 2017; Stab and Gurevych, 2017; Nguyen and Litman, 2016). These discourse-level approaches address the identification of argumentative structures within a single document but do not consider relevance to externally defined topics.

To date, there has been little research on the identification of topic-relevant arguments for argument search. Wachsmuth et al. (2017) present a generic argument search framework. However, it relies on already-structured arguments from debate portals and is not yet able to retrieve arguments from arbitrary texts. Levy et al. (2014) investigate the identification of topic-relevant claims, an approach that was later extended with evidence extraction to mine supporting statements for claims (Rinott et al., 2015). However, both approaches are designed to mine arguments from Wikipedia articles; it is unclear whether their annotation scheme is applicable to other text types. It is also uncertain that it can be easily and accurately applied by untrained annotators, since it requires unitizing (i.e., finding the boundaries of argument components at the token level). Hua and Wang (2017) identify sentences in cited documents that have been used by an editor to formulate an argument. By contrast, we do not limit our approach to the identification of sentences related to a given *argument*, but rather focus on the retrieval of any argument relevant to a given *topic*. The fact that we are concerned with retrieval of arguments also sets our work apart from the discourse-agnostic stance detection task of Mohammad et al. (2016), which is concerned with the identification of sentences expressing support or opposition to a given topic, irrespective of whether those sentences contain supporting evidence (as opposed to mere statements of opinion).

Cross-domain AM experiments have so far been conducted only for discourse-level tasks such as claim identification (Daxenberger et al., 2017), argumentative segment identification (Al-Khatib et al., 2016), and argumentative unit segmentation (Ajjour et al., 2017). However, the discourse-level argumentation models these studies employ seem to be highly dependent on the text types for which they were designed; they do not work well when applied to other text types (Daxenberger et al., 2017). The crucial difference between our own work and prior cross-domain experiments is that we investigate AM from heterogeneous texts across different *topics* instead of studying specific discourse-level AM tasks across restricted text types of existing corpora.

3 Corpus creation

There exists a great diversity in models of argumentation, which differ in their perspective, complexity, terminology, and intended applications (Bentahar et al., 2010). For the present study, we propose a model which, though simplistic, is nonetheless well-suited to the argument search scenario. We define an *argument* as a span of text expressing evidence or reasoning that can be used to either support or oppose a given topic. An argument need not be "direct" or self-contained—it may presuppose some common or domain knowledge, or the application of commonsense reasoning—but it must be unambiguous in its orientation to the topic. A *topic*, in turn, is some matter of controversy for which there is an obvious polarity to the possible outcomes—that is, a question of being either *for* or *against* the use or adoption of something, the commitment to some course of action, etc. In some graph-based models of argumentation (Stab, 2017, Ch. 2), what we refer to as a *topic* would be part of a *(major) claim* expressing a positive or negative stance, and our *arguments* would be *premises* with supporting/attacking *consequence relations* to the claim. However, unlike these models, which are typically used to represent (potentially deep or complex) argument structures at the discourse level, ours is a flat model that considers arguments in isolation from their surrounding context. A great

topic	sentence	label
nuclear energy	Nuclear fission is the process that is used in nuclear reactors to produce high amount of energy using element called uranium.	non-argument
nuclear energy	It has been determined that the amount of greenhouse gases have decreased by almost half because of the prevalence in the utilization of nuclear power.	supporting argument
minimum wage	A 2014 study [. . .] found that minimum wage workers are more likely to report poor health, suffer from chronic diseases, and be unable to afford balanced meals.	opposing argument
minimum wage	We should abolish all Federal wage standards and allow states and localities to set their own minimums.	non-argument

Table 1: Example annotations illustrating our annotation scheme.

advantage of this approach is that it allows annotators to classify text spans without having to read large amounts of context and without having to consider relations to other topics or arguments.

In this work, we consider only those topics that can be concisely and implicitly expressed through keywords, and those arguments that consist of individual sentences. Some examples, drawn from our dataset, are shown in Table 1. Note that while the fourth example expresses opposition to the topic, under our definition it is properly classified as a non-argument because it is a mere statement of stance that provides no evidence or reasoning.

Data. For our experiments we gathered a large collection of manually annotated arguments that cover a variety of topics and that come from a variety of text types. We started by randomly selecting eight topics (see Table 2) from online lists of controversial topics.[2] For each topic, we made a Google query for the topic name, removed results not archived by the Wayback Machine,[3] and truncated the list to the top 50 results. This resulted in a set of persistent, topic-relevant, largely polemical Web documents representing a range of genres and text types, including news reports, editorials, blogs, debate forums, and encyclopedia articles. We preprocessed each document with Apache Tika (Mattmann and Zitting, 2011) to remove boilerplate text. We then used the Stanford CoreNLP tools (Manning et al., 2014) to perform tokenization, sentence segmentation, and part-of-speech tagging on the remaining text, and removed all sentences without verbs or with less than three tokens. This left us with a raw dataset of 27,520 sentences (about 2,700 to 4,400 per topic).

Annotators classified the sentences using a browser-based interface that presents a set of instructions, a topic, a list of sentences, and a multiple-choice form for specifying whether each sentence is a supporting argument, an opposing argument, or not an argument with respect to the topic. (In preliminary experiments, we presented annotators with a fourth option for sentences that are ambiguous or incomprehensible. However, we found that these constituted less than 1% of the distribution and so mapped all such answers to the "no argument" class.)

Annotation experiments. We tested the applicability of our annotation scheme by untrained annotators by performing an experiment where we had a group of "expert" annotators and a group of untrained annotators classify the same set of sentences, and then compared the two groups' classifications. The data for this experiment consisted of 200 sentences randomly selected from each of our eight topics. Our expert annotators were two graduate-level language technology researchers who were fully briefed on the nature and purpose of the argument model. Our untrained annotators were anonymous American workers from the Amazon Mechanical Turk (AMT) crowdsourcing platform. Each sentence was independently annotated by the two expert annotators and ten crowd workers.

Inter-annotator agreement for our two experts, as measured by Cohen's κ, was 0.721; this exceeds the commonly used threshold of 0.7 for assuming the results are reliable (Carletta, 1996). We proceeded by having the two experts resolve their disagreements, resulting in a set of "expert" gold-standard annotations. Similar gold standards were produced for the crowd annotations by applying the MACE denoising tool (Hovy et al., 2013); we tested various thresholds (1.0, 0.9, and 0.8) to discard instances that could be confidently assigned a gold label. We then calculated κ between the remaining instances in the expert and crowd gold standards. In order to

---

[2] `https://www.questia.com/library/controversial-topics`, `https://www.procon.org/`

[3] `https://web.archive.org/`

topic	docs	sentences	no argument	support argument	oppose argument
abortion	50	3,929	2,427	680	822
cloning	50	3,039	1,494	706	839
death penalty	50	3,651	2,083	457	1,111
gun control	50	3,341	1,889	787	665
marijuana legalization	50	2,475	1,262	587	626
minimum wage	50	2,473	1,346	576	551
nuclear energy	50	3,576	2,118	606	852
school uniforms	50	3,008	1,734	545	729
total	400	25,492	14,353	4,944	6,195

Table 2: Corpus size and class distribution.

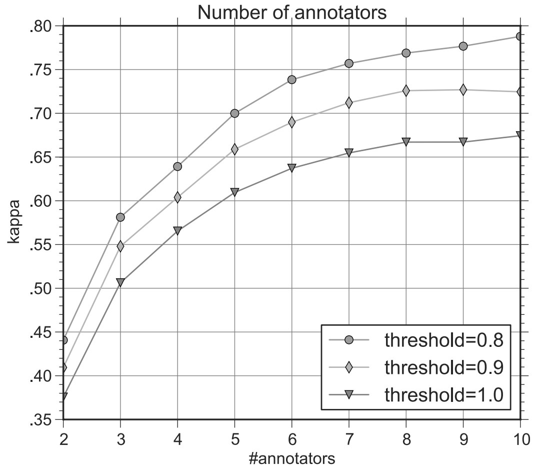

Figure 1: Influence of the number of crowd annotators and different MACE thresholds on κ.

determine the relationship between inter-annotator agreement and the number of crowd workers, we performed this procedure with successively lower numbers of crowd workers, going from the original ten annotators per instance down to two. The results are visualized in Fig. 1. We found that using seven annotators and a MACE threshold of 0.9 results in $\kappa = 0.723$; this gives us similar reliability as with the expert annotators without sacrificing much coverage. Table 3 shows the κ and percentage agreement for this setup, as well as the agreement between our expert annotators, broken down by topic.

We proceeded with annotating the remaining instances in our dataset using seven crowd workers each, paying a rate corresponding to the US federal minimum wage of $7.25/hour. Our total expenditure, including AMT processing fees, was $2,774.02. After MACE denoising, we were left with 25,492 gold-standard annotations. Table 2 provides statistics on the size and class distribution of the final corpus. We are releasing the gold-standard annotations for this dataset, and code for retrieving the original sentences from the Wayback Machine, under a Creative Commons licence.

	expert–expert		crowd–expert	
	%	κ	%	κ
abortion	.884	.651	.834	.660
cloning	.845	.712	.821	.704
death penalty	.851	.657	.770	.576
gun control	.907	.783	.796	.638
marijuana legalizat.	.850	.729	.854	.749
minimum wage	.885	.779	.858	.745
nuclear energy	.809	.686	.889	.825
school uniforms	.864	.767	.931	.889
average	.862	.721	.844	.723

Table 3: Agreement between experts, and between the expert and crowd gold standards.

4 Approaches for identifying arguments

We model the identification of arguments as a sentence-level classification task. In particular, given a sentence $\boldsymbol{\varsigma}$ with words $u_1, \ldots, u_{n_\varsigma}$ and a topic $\boldsymbol{\tau}$ of words $v_1, \ldots, v_{n_\tau}$ (e.g., "gun control" or "school uniforms"), we aim to classify $\boldsymbol{\varsigma}$ as a "supporting argument" or "opposing argument" if it includes a relevant reason for supporting or opposing the $\boldsymbol{\tau}$, or as a "non-argument" if it does not include a reason or is not relevant to $\boldsymbol{\tau}$. We also investigate a two-label classification where we combine supporting and opposing arguments into a single category; this allows us to evaluate argument classification independent of stance. We focus on the challenging task of cross-topic experiments, where one topic is withheld from the training data and used for testing. Here, we denote scalars by italic lowercase letters (e.g., t), vector representations by italic bold lowercase letters (e.g., $\boldsymbol{c}$), and matrices as italic bold uppercase letters (e.g., $\boldsymbol{W}$).

4.1 Integrating topic information

Since arguments need to be relevant to the given topic, we posit that providing topic information to the learner results in a more robust prediction capability in cross-topic setups. Below, we present two models that integrate the topic, one that uses an attention mechanism and another that includes the topic vector directly in the LSTM cell.

Outer-attention BiLSTM (outer-att). To let the model learn which parts of the sentence are relevant (or irrelevant) to the given topic, we use an attention-based neural network (Bahdanau et al., 2014) that learns an importance weighting of the input words depending on the given topic. In particular, we adopt an outer-attention mechanism similar to the one proposed by Hermann et al. (2015), which has achieved state-of-the-art results in related tasks such as natural language inference and recognizing textual entailment (Rocktäschel et al., 2015; Wang and Jiang, 2016). We combine the attention mechanism with a common BiLSTM model and, at time step t, determine the importance weighting for each hidden state $\boldsymbol{h}(t)$ as

$$\boldsymbol{m}(t) = \tanh(\boldsymbol{W}_h \boldsymbol{h}(t) + \boldsymbol{W}_p \boldsymbol{p}) \qquad (1)$$

$$f_{attention}(\boldsymbol{h}(t), \boldsymbol{p}) = \frac{\exp(\boldsymbol{w}_m^T \boldsymbol{m}(t))}{\sum_t \exp(\boldsymbol{w}_m^T \boldsymbol{m}(t))} \qquad (2)$$

where $\boldsymbol{W}_h$, $\boldsymbol{W}_p$, and $\boldsymbol{w}_m$ are trainable parameters of the attention mechanism and $\boldsymbol{p}$ is the average of all word embeddings of topic words $v_1, \ldots, v_{n_\tau}$. Using the importance weighting, we determine the final, weighted hidden output state $\boldsymbol{s}$ as

$$\alpha_t \propto f_{attention}(\boldsymbol{h}(t), \boldsymbol{p}) \qquad (3)$$

$$\boldsymbol{s} = \sum_{t=1}^{n} \boldsymbol{h}(t) \alpha_t . \qquad (4)$$

Finally, we feed $\boldsymbol{s}$ into a dense layer with a softmax activation function to get predictions for our two- or three-label setups.

Contextual BiLSTM (biclstm). A more direct approach to integrating an argument's topic is the *contextual LSTM* (CLSTM) architecture (Ghosh et al., 2016), where topic information is added as another term to all four gates of an LSTM cell. We, however, hypothesize that topic information is more relevant at the $\boldsymbol{i}$- and $\boldsymbol{c}$-gates, the former because it has the biggest impact on how a new token is processed and the latter because it is closely linked to how the sequence seen so far is to be interpreted and stored. To this end, we experimented with several modifications to the original CLSTM such as removing peepholes—i.e., removing gates' access to the cell state $\boldsymbol{c}$ (Gers and Schmidhuber, 2000)—and removing topic information from one or more gates. Empirical results on the validation set show that topic integration at the $\boldsymbol{i}$- and $\boldsymbol{c}$-gates only, and removal of all peephole connections, does indeed outperform the original CLSTM on our task by 1 percentage point. Our modified CLSTM (Fig. 2) is defined as

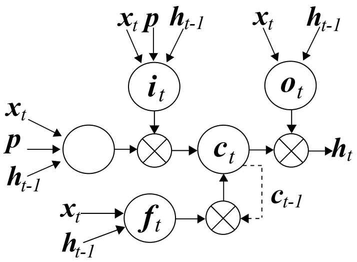

Figure 2: Architecture of a CLSTM cell.

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{x_i} \boldsymbol{x}_t + \boldsymbol{W}_{h_i} \boldsymbol{h}_{t-1} + \boldsymbol{b}_i + \boxed{\boldsymbol{W}_{\boldsymbol{p}_i} \boldsymbol{p}}) \qquad (5)$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{x_f} \boldsymbol{x}_t + \boldsymbol{W}_{h_f} \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \qquad (6)$$

$$\begin{aligned} \boldsymbol{c}_t = \boldsymbol{f}_t \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \sigma_c(&\boldsymbol{W}_{x_c} \boldsymbol{x}_t + \boldsymbol{W}_{h_c} \boldsymbol{h}_{t-1} \\ &+ \boldsymbol{b}_c + \boxed{\boldsymbol{W}_{\boldsymbol{p}_c} \boldsymbol{p}}) \end{aligned} \qquad (7)$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{x_o} \boldsymbol{x}_t + \boldsymbol{W}_{h_o} \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \qquad (8)$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \sigma_c(\boldsymbol{c}_t). \qquad (9)$$

Here $\boldsymbol{i}$, $\boldsymbol{f}$, and $\boldsymbol{o}$ represent the input, forget, and output gates; $\boldsymbol{c}$ the cell memory; $\boldsymbol{x}_t$ the embedded token of a sentence at timestep t; $\boldsymbol{h}_{t-1}$ the previous hidden state; and $\boldsymbol{b}$ the bias. σ and σ_c are the activation and recurrent activation functions, respectively. The novel terms for topic integration are outlined. We use this model bidirectionally, as we did with our BiLSTM network, and hence refer to it as biclstm.

4.2 Leveraging additional data

As we want to classify arguments related to specific topics, leveraging information that supports the classifier in the decision of topic-relation is crucial. The multi-task learning (mtl) and transfer learning (trl) models are able to make use of auxiliary data that can potentially improve the results on the main task. Thus, we extend our previously described models by integrating them into mtl and trl setups. We also choose to integrate two corpora

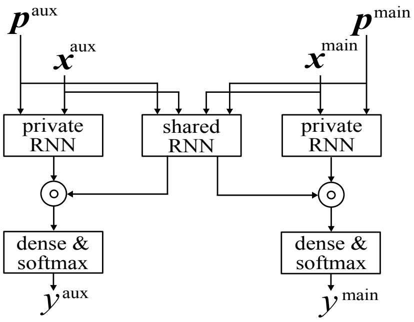

Figure 3: Multi-task learning architecture. The ⊚ symbol denotes the concatenation operator.

from which we expect to learn (a) topic-relevance and (b) the capability to distinguish between supporting and attacking arguments. The first corpus, DIP2016 (Habernal et al., 2016), consists of 49 queries from the educational domain and 100 documents for each query. Each document has its sentences annotated for relevance (true/false) to the query.[4] The second corpus, from SemEval-2016 Task 6 (Mohammad et al., 2016), consists of around 5000 multi-sentence tweets, a corresponding topic (e.g., “atheism”), and the author’s stance on the topic (for/against/neither).

For our mtl and trl approaches, we consider every possible pairing of a model (biclstm, outer-att, and the bilstm baseline we introduce in §5) with an auxiliary corpus (DIP2016, SemEval). We formalize our datasets as $S_k = \{(\boldsymbol{x}_i^k, \boldsymbol{p}_i^k, y_i^k) | i = 0, \ldots, |S_k|\}$, where k can be either our main dataset or an auxiliary dataset, $\boldsymbol{x}_i^k$ denotes a single sentence as a sequence of word embeddings and y_i^k its corresponding label in k, and $\boldsymbol{p}_i^k$ represents the corresponding averaged topic vector.

Transfer learning (trl). For trl, we use the approach of *parameter transfer* (Pan and Yang, 2010)—i.e., we do not modify the model used. Instead, we train the model twice: the first time, we train the model on the chosen auxiliary corpus, and the second time, we keep the trained model’s weights and train it with our own corpus. For the three-label setting, we have to modify the transfer model slightly for the DIP2016 corpus, since it provides only two labels for each training sample. In this case, we simply add a layer with two neurons on top of the layer with three neurons for training with the DIP2016 corpus and remove it afterwards for training with our corpus.

Multi-task learning (mtl). For mtl, we use a shared–private model (Liu et al., 2017), which showed promising results for text classification and word segmentation (Chen et al., 2017). (We also experimented with their adversarial approach to learn topic-invariant features, but abandoned this due to low scores.) The mtl base model consists of a private recurrent neural network (RNN) for both the auxiliary dataset and our dataset, plus a shared RNN that both datasets use (Fig. 3). The last hidden states of the RNNs are concatenated and fed through a dense layer and a softmax activation function. The model is trained in an alternating fashion—i.e., after each epoch the loss for the other dataset is minimized until each dataset has run for the set number of epochs, where the last epoch is always executed on our dataset. At prediction time, only the private RNN trained on our dataset and the shared RNN are used. The core idea is that the shared RNN learns what is relevant for both tasks, while the private ones learn only the task-specific knowledge.

For the cases of mtl+bilstm+*corpus*, mtl+biclstm+*corpus*, and mtl+outer-att+*corpus*, we simply switch the RNN with our bilstm, biclstm, and outer-att, respectively. For mtl+outer-att+*corpus*, we add the outer attention mechanism (see §4.1), modified for use with the mtl model, after each of the private RNNs, while additionally feeding it a second topic vector—the last hidden state of the shared RNN:

$$\boldsymbol{m}(t) = \tanh(\boldsymbol{W}_r \boldsymbol{h}_r(t) + \boldsymbol{W}_s \boldsymbol{h}_s + \boldsymbol{W}_p \boldsymbol{p}) \qquad (10)$$

$$f_{attention}(\boldsymbol{h}_r(t), \boldsymbol{h}_s, \boldsymbol{p}) = \frac{\exp(\boldsymbol{w}_m^T \boldsymbol{m}(t))}{\sum_t \exp(\boldsymbol{w}_m^T \boldsymbol{m}(t))} \qquad (11)$$

$$\alpha_t \propto f_{attention}(\boldsymbol{h}_r(t), \boldsymbol{h}_s, \boldsymbol{p}) \qquad (12)$$

$$\boldsymbol{s} = \sum_{t=1}^{n} \boldsymbol{h}_r(t) \alpha_t \qquad (13)$$

where $\boldsymbol{W}_r$, $\boldsymbol{W}_s$, and $\boldsymbol{W}_p$ are trainable weight matrices, $\boldsymbol{h}_r(t)$ is the hidden state of the private bilstm at timestep t, $\boldsymbol{h}_s$ is the last hidden state of the shared model, and $\boldsymbol{p}$ is the average of all word embeddings of topic words $v_1, \ldots, v_{n_\tau}$.

5 Evaluation

To evaluate the robustness of the models, we conduct cross-topic experiments to evaluate how well the models generalize to an unknown topic. To this end, we combine training (70%) and validation

---

[4] We only use 300K of the corpus’s 600K samples to ease hyperparameter tuning for our computation-heavy models.

	two labels			three labels				
model	F_1	P_{arg}	R_{arg}	F_1	P_{arg+}	P_{arg-}	R_{arg+}	R_{arg-}
bilstm (baseline)	.6069 ± .0074	.7339 ± .0110	.3844 ± .0122	.3796 ± .0079	.3484 ± .0479	.4710 ± .0210	.0963 ± .0148	.2181 ± .0181
lr-uni (baseline)	.5854 ± .0131	.6519 ± .0093	.3587 ± .0264	.3821 ± .0056	.2782 ± .0293	.4217 ± .0171	.1176 ± .0165	.2119 ± .0203
outer-att	.6213 ± .0106	.7309 ± .0108	.4138 ± .0237	.3873 ± .0081	**.3651** ± .0244	.4696 ± .0169	.1042 ± .0173	.2381 ± .0117
biclstm	.6414 ± .0129	.6244 ± .0132	**.7035** ± .0261	.4242 ± .0122	.2675 ± .0148	.3887 ± .0141	**.2817** ± .0369	.4028 ± .0496
tr+bilstm+semeval	.6297 ± .0073	**.7500** ± .0047	.4233 ± .0125	.3698 ± .0142	.3128 ± .0422	.4075 ± .0640	.0897 ± .0256	.2089 ± .0133
tr+outer-att+semeval	.6293 ± .0057	.7297 ± .0122	.4336 ± .0156	.3871 ± .0089	.3160 ± .0397	.4469 ± .0369	.1245 ± .0160	.2264 ± .0147
tr+biclstm+semeval	.6433 ± .0182	.6625 ± .0128	.6181 ± .0259	.3953 ± .0122	.2606 ± .0356	.4226 ± .0203	.1743 ± .0385	.3643 ± .0574
tr+bilstm+dip2016	.6254 ± .0133	.7073 ± .0114	.4200 ± .0253	.3628 ± .0136	.2396 ± .0605	.4470 ± .0319	.0517 ± .0284	.2298 ± .0245
tr+outer-att+dip2016	.6074 ± .0115	.7112 ± .0245	.4031 ± .0238	.3438 ± .0233	.2060 ± .1012	.4171 ± .0521	.1105 ± .0821	.2096 ± .0793
tr+biclstm+dip2016	.6110 ± .0206	.6954 ± .0491	.4904 ± .0502	.3595 ± .0226	.2272 ± .0516	.3474 ± .0539	.1191 ± .0856	.2886 ± .0714
mtl+bilstm+semeval	.6126 ± .0093	.7270 ± .0087	.3906 ± .0177	.3765 ± .0081	.3248 ± .0304	**.4812** ± .0340	.0888 ± .0137	.2153 ± .0162
mtl+outer-att+semeval	.6221 ± .0100	.7186 ± .0123	.4219 ± .0187	.3764 ± .0071	.3185 ± .0393	.4763 ± .0213	.0878 ± .0173	.2149 ± .0295
mtl+biclstm+semeval	.6519 ± .0079	.6495 ± .0143	.6690 ± .0333	.4147 ± .0105	.2769 ± .0332	.3819 ± .0141	.2465 ± .0497	**.4069** ± .0501
mtl+bilstm+dip2016	.6145 ± .0097	.7312 ± .0100	.3979 ± .0208	.3757 ± .0057	.3255 ± .0382	.4647 ± .0255	.0841 ± .0144	.2261 ± .0192
mtl+outer-att+dip2016	.6263 ± .0079	.7176 ± .0100	.4327 ± .0178	.3842 ± .0070	.3427 ± .0365	.4502 ± .0240	.1007 ± .0147	.2327 ± .0146
mtl+biclstm+dip2016	**.6662** ± .0148	.6463 ± .0105	.6719 ± .0489	**.4285** ± .0139	.2947 ± .0383	.3815 ± .0221	.2722 ± .0582	.3483 ± .0528

Table 4: Results for each model on the test sets. Bold numbers indicate the highest score in the column.

data (10%) of seven topics for training and parameter tuning, and use the test data (20%) of the eighth topic for testing. For encoding the words of sentence $\boldsymbol{\varsigma}$ and topic $\boldsymbol{\tau}$, we use 300-dimensional word embeddings trained on the Google News dataset by Mikolov et al. (2013). To handle out-of-vocabulary words, we create separate random word vectors for each.[5]

Since reporting single performance scores is insufficient to compare non-deterministic learning approaches like neural networks (Reimers and Gurevych, 2017), we report all results as averages over ten runs with different random seeds. As evaluation measures, we report the average macro F_1, as well as the precision and the recall for the argument class (P_{arg}, R_{arg}). For the three-label approach, we split the precision and recall for predicting supporting (P_{arg+}, R_{arg+}) and attacking arguments (P_{arg-}, R_{arg-}). As baselines, we use a simple bidirectional LSTM (Hochreiter and Schmidhuber, 1997), as well as a logistic regression model with lowercased unigram features, which has been shown to be a strong baseline for various other AM tasks (Daxenberger et al., 2017; Stab and Gurevych, 2017). We refer to these models as bilstm and lr-uni, respectively. All neural networks are trained using the Adam optimizer (Kingma and Ba, 2015) and cross-entropy loss function. For finding the best model, we run each for ten epochs and take the best model based on the lowest validation loss. In addition to that, we tune the hyperparameters of all neural networks (see Appendix A). To accelerate training, we truncate sentences at 60 words.[6]

5.1 Results

Two-label setup. The results in Table 4 show that all our models outperform the baselines for two-label prediction.[7] F_1 for biclstm improves by 3.5 percentage points over the bilstm baseline and by 5.6 over lr-uni. A main reason for this proves to be the substantial increase in recall for our topic-integrating models—outer-att and especially biclstm—in comparison to our baselines. These results show that knowledge of the argument's topic has a strong impact on argument prediction capability. Further, we observe that integrating biclstm in a multi-task learning setup in order to draw knowledge about topic relevance from the DIP2016 corpus (mtl+biclstm+dip2016) improves F_1 by an additional 2.5 percentage points. It achieves an F_1 of 0.6662, which is 19.48 percentage points less than the human upper bound of 0.861. When using the SemEval corpus, which holds less task-relevant knowledge for our two-label approach, we are able to gain only 1 percentage point when integrating it into mtl+biclstm+*corpus*.

For the transfer learning models that integrate the topic (tr+biclstm+*corpus* and tr+outer-att+*corpus*), the parameter transfer is mostly ineffective. If no topic is provided (tr+bilstm+*corpus*), the transfer learning models are able to improve over the baseline bilstm. This shows that the parameter transfer

---

[5] Each dimension is set to a random number between −0.01 and 0.01. Digits are mapped to the same random word vector.

[6] Only 244 of our sentences (<1%) exceed this length.

[7] Detailed results per topic are given in Appendix B.

itself can be of use, but confuses the model when combined with topic integration.

In general, we observe an overall lower score for trl models that use the DIP2016 corpus compared to those using the SemEval corpus. In contrast to the mtl model, for trl models *all* parameters are transferred to the main task, not just parameters that represent shared knowledge. Thus, we suspect the lower scores of the trl models with DIP2016 are due to overfitting on the vast number of samples which shape the parameters much more than the comparatively small SemEval corpus could.

Three-label setup. For the three-label approach, we observe overall lower scores due to the additional difficulty in distinguishing supporting from opposing arguments. As already observed in the two-label setup, biclstm outperforms both the bilstm and lr-uni baselines; here, the former by 4.5 and the latter by 4.2 percentage points in F_1. Again, this is caused by a substantial increase in recall and shows the impact that the available topic information has on the classifier's predictive power.

For transfer learning, we see similar results as for the two-label approach; both the DIP2016 and SemEval corpora have a generally negative impact when compared to the respective base models. The SemEval corpus does not provide the knowledge required to distinguish supporting from attacking arguments. We conclude that the original purpose of the SemEval task, stance recognition, is too different from our own. But in multi-task learning, where only the shared parameters are taken, we observe slight improvements when using biclstm with DIP2016; this correlates with the same model in the two-label setup.

5.2 Error analysis

To understand the errors of our best model, mtl-biclstm-dip, and the nature of this task, we manually analyzed 100 sentences randomly sampled from the false positive and false negative arguments of the three-label experiments (combining supporting and attacking arguments). Among the false positives, we found 48 off-topic sentences that were wrongly classified as arguments. The 52 on-topic false positives consist of non-argumentative background information or mere opinions without evidence (as with the first and fourth examples of Table 1) and questions about the topic. Among the false negatives, we found 65 arguments that did not explicitly refer to the topic but to related aspects that depend on background knowledge. For instance, the model fails to establish an argumentative link between the topic "gun control" and the Second Amendment to the US Constitution. Lastly, we inspected arguments that are incorrectly classified as supporting and/or opposing a topic. We found several samples in which the term "against" is not correctly interpreted and the argument is classified as supporting a topic. Similarly, for arguments incorrectly classified as attacking, we find various samples where the word "oppose" is used not to oppose the topic but to strengthen a supporting argument, as in "There is reason even for people who oppose the use of marijuana to support its legalization. . . "

5.3 Adapting to new topics

To evaluate the performance of the models in data-scarce scenarios, we gradually add target topic data to the training data and analyze the model performance on the target test set. Figure 4 shows model performance (F_1, P_{arg}, and R_{arg}) on the "marijuana legalization" topic when adding different amounts of randomly sampled topic-specific data to the training data (x-axes).[8] As the results show, the models that integrate the topic achieve higher recall when adding target topic data to the training data. For bilstm, we observe a drastic difference when compared to the other models; the recall for arguments stays at around 30% and rises only when integrating more than 60% target topic data. In strong contrast, topic-integrating models retrieve a much higher number of actual arguments at target topic augmentation levels as low as 20%. Further, and equally important, this does not come at the cost of precision; on the contrary, the precision is mostly steady and slowly rising after around 20% of target topic integration, leading to an overall higher F_1 for these models. Finally, in comparing F_1 between topic-integrating models and bilstm, we conclude that the former need much less target topic data to substantially improve their score, making them more robust in situations of data scarcity.

6 Conclusion

We have presented a new approach for searching a document collection for arguments relevant to a given topic. First, we introduced an annotation scheme suited to the information-seeking perspec-

---

[8] Each data point in the plot is the average score of ten runs with different random samples of target topic data.

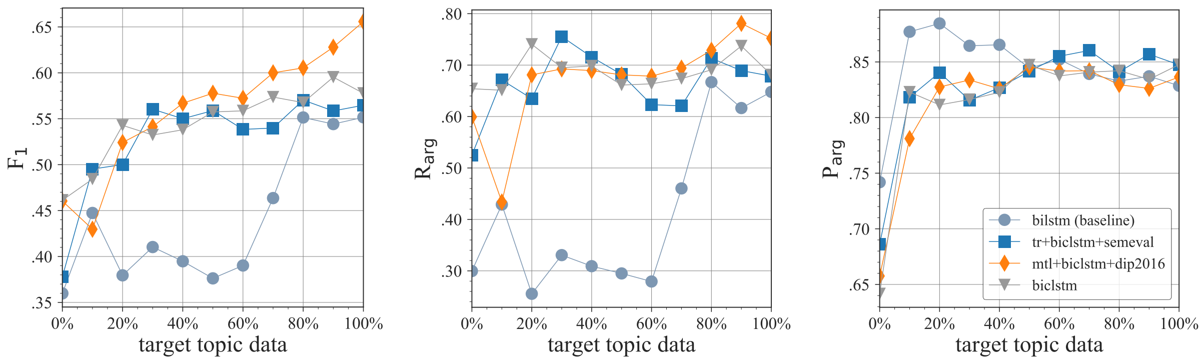

Figure 4: Model performance (y-axes) according to the amount of target topic data in the train sets (x-axes) for the "marijuana legalization" topic in the three-label setup.

tive of argument search and showed that it is cheaply but reliably applicable by untrained annotators to arbitrary Web texts. Second, we presented a new corpus, including over 25,000 instances over eight topics, that allows for cross-topic experiments using heterogeneous text types. Third, we conducted cross-topic experiments and showed that integrating topic information of arguments with our contextual BiLSTM leads to better generalization to unknown topics. Fourth, by leveraging knowledge from similar datasets and integrating our contextual BiLSTM into a multi-task learning setup, we were able to gain an improvement over our strongest baseline of 5.9 percentage points in F_1 in the two-label setup and 4.6 in the three-label setup. Finally, by gradually adding target topic data to our training set, we showed that, when available, even small amounts of target topic data (20%) have a strong positive influence on the recall of arguments.

In a separate, simultaneously written paper (Stab et al., 2018) we evaluate our models in real-world application scenarios by applying them to a large document collection and comparing the results to a manually produced gold standard. An online argument search engine implementing our approach is now available for noncommercial use at `https://www.argumentsearch.com/`. Furthermore, we are experimenting with language adaptation and plan to extend the tool to the German language. Preliminary results are presented in Stahlhut (2018). We also intend to investigate methods for grouping similar arguments.

Acknowledgements

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 03VP02540 (ArgumenText) and the DFG-funded research training group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1).

References

Yamen Ajjour, Wei-Fan Chen, Johannes Kiesel, Henning Wachsmuth, and Benno Stein. 2017. Unit segmentation of argumentative texts. In *Proceedings of the 4th Workshop on Argument Mining*, pages 118–128. Association for Computational Linguistics.

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Köhler, and Benno Stein. 2016. Cross-domain mining of argumentative text through distant supervision. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1395–1404. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Jamal Bentahar, Bernard Moulin, and Micheline Bélanger. 2010. A taxonomy of argumentation models used for knowledge representation. *Artificial Intelligence Review*, 33(3):211–259.

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1193–1203. Association for Computational Linguistics.

Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab, and Iryna Gurevych. 2017. What is the essence of a claim? Cross-domain claim identification. In *Proceedings of the 2017 Conference* *on Empirical Methods in Natural Language Processing*, pages 2045–2056. Association for Computational Linguistics.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22. Association for Computational Linguistics.

Felix A. Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *Proceedings of the IEEE–INNS–ENNS International Joint Conference on Neural Networks*, volume 3, pages 189–194. IEEE.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry P. Heck. 2016. Contextual LSTM (CLSTM) models for large scale NLP tasks. *CoRR*, abs/1602.06291.

Theodosis Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. 2014. Argument extraction from news, blogs, and social media. In *Artificial Intelligence: Methods and Applications*, volume 8445 of *Lecture Notes in Computer Science*, pages 287–299. Springer International Publishing.

Ivan Habernal, Judith Eckle-Kohler, and Iryna Gurevych. 2014. Argumentation mining on the Web from information seeking perspective. In *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing*, volume 1341 of *CEUR Workshop Proceedings*, pages 26–39. CEUR-WS.org.

Ivan Habernal, Maria Sukhareva, Fiana Raiber, Anna Shtok, Oren Kurland, Hadar Ronen, Judit Bar-Ilan, and Iryna Gurevych. 2016. New Collection Announcement: Focused Retrieval Over the Web. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 701–704. ACM.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

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

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130. Association for Computational Linguistics.

Xinyu Hua and Lu Wang. 2017. Understanding and detecting diverse supporting arguments of diverse types. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 203–208. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Keiichi Kobayashi. 2009. Comprehension of relations among controversial texts: Effects of external strategy use. *Instructional Science*, 37(4):311–324.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500. Dublin City University and Association for Computational Linguistics.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1–10.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Chris A. Mattmann and Jukka Zitting. 2011. *Tika in Action*. Manning Publications, Greenwich, CT, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Raquel Mochales-Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107. ACM.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 Task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 31–41. Association for Computational Linguistics.

Huy Nguyen and Diane Litman. 2016. Context-aware argumentative relation mining. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1127–1137. Association for Computational Linguistics.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348. Association for Computational Linguistics.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence – An automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450. Association for Computational Linguistics.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Christian Stab. 2017. *Argumentative Writing Support by Means of Natural Language Processing*. Dr.-Ing. thesis, Technische Universität Darmstadt.

Christian Stab, Johannes Daxenberger, Chris Stahlhut, Tristan Miller, Benjamin Schiller, Christopher Tauchmann, Steffen Eger, and Iryna Gurevych. 2018. ArgumenText: Searching for arguments in heterogeneous sources. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations (NAACL-HLT 2018)*, pages 21–25.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 46–56. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Chris Stahlhut. 2018. Searching arguments in German with ArgumenText. In *Proceedings of the First Biennial Conference on Design of Experimental Search & Information Retrieval Systems*, volume 2167 of *CEUR Workshop Proceedings*, page 104.

Ola Svenson. 1979. Process descriptions of decision making. *Organizational Behavior and Human Performance*, 23(1):86–112.

Henning Wachsmuth, Martin Potthast, Khalid Al Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59. Association for Computational Linguistics.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451. Association for Computational Linguistics.

Ursula Wingate. 2012. 'Argument!' Helping students understand what essay writing is about. *Journal of English for Academic Purposes*, 11(2):145–154.

Adam Wyner, Raquel Mochales-Palau, Marie-Francine Moens, and David Milward. 2010. Approaches to text mining arguments from legal cases. In Enrico Francesconi, Simonetta Montemagni, Wim Peters, and Daniela Tiscornia, editors, *Semantic Processing of Legal Texts: Where the Language of Law Meets the Law of Language*, volume 6036 of *Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence*, pages 60–79. Springer.

Summarizing Opinions: Aspect Extraction Meets Sentiment Prediction and They Are Both Weakly Supervised

Stefanos Angelidis and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
`s.angelidis@ed.ac.uk, mlap@inf.ed.ac.uk`

Abstract

We present a neural framework for opinion summarization from online product reviews which is knowledge-lean and only requires light supervision (e.g., in the form of product domain labels and user-provided ratings). Our method combines two weakly supervised components to identify salient opinions and form extractive summaries from multiple reviews: an aspect extractor trained under a multi-task objective, and a sentiment predictor based on multiple instance learning. We introduce an opinion summarization dataset that includes a training set of product reviews from six diverse domains and human-annotated development and test sets with gold standard aspect annotations, salience labels, and opinion summaries. Automatic evaluation shows significant improvements over baselines, and a large-scale study indicates that our opinion summaries are preferred by human judges according to multiple criteria.[1]

1 Introduction

Opinion summarization, i.e., the aggregation of user opinions as expressed in online reviews, blogs, internet forums, or social media, has drawn much attention in recent years due to its potential for various information access applications. For example, consumers have to wade through many product reviews in order to make an informed decision. The ability to summarize these reviews succinctly would allow customers to efficiently absorb large amounts of opinionated text and manufacturers to keep track of what customers think about their products (Liu, 2012).

The majority of work on opinion summarization is *entity-centric*, aiming to create summaries from text collections that are relevant to a particular entity of interest, e.g., product, person, company, and so on. A popular decomposition of the problem involves three subtasks (Hu and Liu, 2004, 2006): (1) *aspect extraction* which aims to find specific features pertaining to the entity of interest (e.g., battery life, sound quality, ease of use) and identify expressions that discuss them; (2) *sentiment prediction* which determines the sentiment orientation (positive or negative) on the aspects found in the first step, and (3) *summary generation* which presents the identified opinions to the user (see Figure 1 for an illustration of the task).

A number of techniques have been proposed for aspect discovery using part of speech tagging (Hu and Liu, 2004), syntactic parsing (Lu et al., 2009), clustering (Mei et al., 2007; Titov and McDonald, 2008b), data mining (Ku et al., 2006), and information extraction (Popescu and Etzioni, 2005). Various lexicon and rule-based methods (Hu and Liu, 2004; Ku et al., 2006; Blair-Goldensohn et al., 2008) have been adopted for sentiment prediction together with a few learning approaches (Lu et al., 2009; Pappas and Popescu-Belis, 2017; Angelidis and Lapata, 2018). As for the summaries, a common format involves a list of aspects and the number of positive and negative opinions for each (Hu and Liu, 2004). While this format gives an overall idea of people's opinion, reading the actual text might be necessary to gain a better understanding of specific details. Textual summaries are created following mostly extractive methods (but see Ganesan et al. 2010 for an abstractive approach), and various formats ranging from lists of words (Popescu and Etzioni, 2005), to phrases (Lu et al., 2009), and sentences (Mei et al., 2007; Blair-Goldensohn et al., 2008; Lerman et al., 2009; Wang and Ling, 2016).

In this paper, we present a neural framework for opinion extraction from product reviews. We follow the standard architecture for aspect-based summarization, while taking advantage of the success of neural network models in learning continuous features without recourse to preprocessing tools or linguistic annotations. Central to our system is the ability to accurately identify aspect-

---

[1] Our code and dataset are publicly available at `https://github.com/stangelid/oposum`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3675–3686
Brussels, Belgium, October 31 - November 4, 2018.

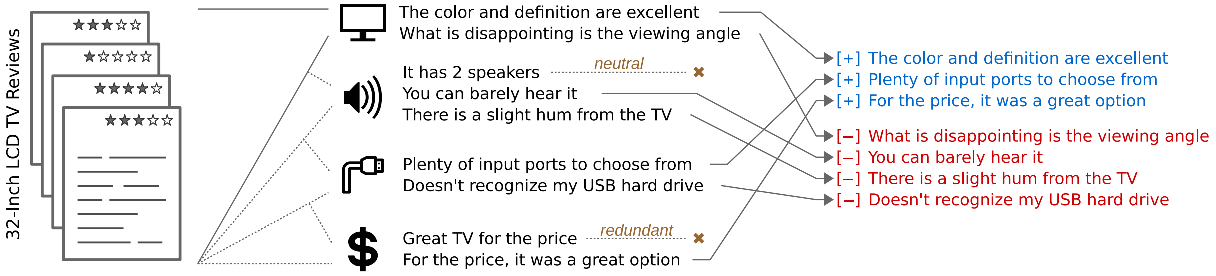

Figure 1: Aspect-based opinion summarization. Opinions on image quality, sound quality, connectivity, and price of an LCD television are extracted from a set of reviews. Their polarities are then used to sort them into positive and negative, while neutral or redundant comments are discarded.

specific opinions by using different sources of information freely available with product reviews (product domain labels, user ratings) and minimal domain knowledge (essentially a few aspect-denoting keywords). We incorporate these ideas into a recently proposed aspect discovery model (He et al., 2017) which we combine with a weakly supervised sentiment predictor (Angelidis and Lapata, 2018) to identify highly salient opinions. Our system outputs extractive summaries using a greedy algorithm to minimize redundancy. Our approach takes advantage of weak supervision signals only, requires minimal human intervention and no gold-standard salience labels or summaries for training.

Our contributions in this work are three-fold: a novel neural framework for the identification and extraction of salient customer opinions that combines aspect and sentiment information and does not require unrealistic amounts of supervision; the introduction of an opinion summarization dataset which consists of Amazon reviews from six product domains, and includes development and test sets with gold standard aspect annotations, salience labels, and multi-document extractive summaries; a large-scale user study on the quality of the final summaries paired with automatic evaluations for each stage in the summarization pipeline (aspects, extraction accuracy, final summaries). Experimental results demonstrate that our approach outperforms strong baselines in terms of opinion extraction accuracy and similarity to gold standard summaries. Human evaluation further shows that our summaries are preferred over comparison systems across multiple criteria.

2 Related Work

It is outside the scope of this paper to provide a detailed treatment of the vast literature on opinion summarization and related tasks. For a comprehensive overview of non-neural methods we refer the interested reader to Kim et al. (2011) and Liu and Zhang (2012). We are not aware of previous studies which propose a neural-based system for end-to-end opinion summarization without direct supervision, although as we discuss below, recent efforts tackle various subtasks independently.

Aspect Extraction Several neural network models have been developed for the identification of aspects (e.g., words or phrases) expressed in opinions. This is commonly viewed as a supervised sequence labeling task; Liu et al. (2015) employ recurrent neural networks, whereas Yin et al. (2016) use dependency-based embeddings as features in a Conditional Random Field (CRF). Wang et al. (2016) combine a recursive neural network with CRFs to jointly model aspect and sentiment terms. He et al. (2017) propose an aspect-based autoencoder to discover fine-grained aspects without supervision, in a process similar to topic modeling. Their model outperforms LDA-style approaches and forms the basis of our aspect extractor.

Sentiment Prediction Fully-supervised approaches based on neural networks have achieved impressive results on fine-grained sentiment classification (Kim, 2014; Socher et al., 2013). More recently, *Multiple Instance Learning* (MIL) models have been proposed that use freely available review ratings to train segment-level predictors. Kotzias et al. (2015) and Pappas and Popescu-Belis (2017) train sentence-level predictors under a MIL objective, while our previous work (Angelidis and Lapata, 2018) introduced MILNET, a hierarchical model that is trained end-to-end on document labels and produces polarity-based opinion summaries of single reviews. Here, we use MILNET to predict the sentiment polarity of individual opinions.

Multi-document Summarization A few extractive neural models have been recently applied to generic multi-document summarization. Cao et al. (2015) train a recursive neural network using a ranking objective to identify salient sentences, while follow-up work (Cao et al., 2017) employs a multi-task objective to improve sentence extraction, an idea we adapted to our task. Yasunaga et al. (2017) propose a graph convolution network to represent sentence relations and estimate sentence salience. Our summarization method is tailored to the opinion extraction task, it identifies aspect-specific and salient units, while minimizing the redundancy of the final summary with a greedy selection algorithm (Cao et al., 2015; Yasunaga et al., 2017). Redundancy is also addressed in Ganesan et al. (2010) who propose a graph-based framework for abstractive summarization. Wang and Ling (2016) introduce an encoder-decoder neural method for extractive opinion summarization. Their approach requires direct supervision via gold-standard extractive summaries for training, in contrast to our weakly supervised formulation.

3 Problem Formulation

Let C denote a corpus of reviews on a set of products $E_{\mathrm{C}} = \{e_i\}_{i=1}^{|E_{\mathrm{C}}|}$ from a domain d_{C}, e.g., televisions or keyboards. For every product e, the corpus contains a set of reviews $R_e = \{r_i\}_{i=1}^{|R_e|}$ expressing customers' opinions. Each review r_i is accompanied by the author's overall rating y_i and is split into segments $(s_1, \ldots, s_m)$, where each segment s_j is in turn viewed as a sequence of words $(w_{j1}, \ldots, w_{jn})$. A segment can be a sentence, a phrase, or in our case an *Elementary Discourse Unit* (EDU; Mann and Thompson 1988) obtained from a *Rhetorical Structure Theory* (RST) parser (Feng and Hirst, 2012). EDUs roughly correspond to clauses and have been shown to facilitate performance in summarization (Li et al., 2016), document-level sentiment analysis (Bhatia et al., 2015), and single-document opinion extraction (Angelidis and Lapata, 2018).

A segment may discuss zero or more *aspects*, i.e., different product attributes. We use $A_{\mathrm{C}} = \{a_i\}_{i=1}^{K}$ to refer to the aspects pertaining to domain d_{C}. For example, *picture quality*, *sound quality*, and *connectivity* are all aspects of televisions. By convention, a *general* aspect is assigned to segments that do not discuss any specific aspects. Let $A_s \subseteq A_{\mathrm{C}}$ denote the set of aspects mentioned in segment s; $pol_s \in [-1, +1]$ marks the *polarity* a segment conveys, where -1 indicates maximally negative and $+1$ maximally positive sentiment. An opinion is represented by tuple $o_s = (s, A_s, pol_s)$, and $O_e = \{o_s\}_{s \in R_e}$ represents the set of all opinions expressed in R_e.

For each product e, our goal is to produce a summary of the most salient opinions expressed in reviews R_e, by selecting a small subset $S_e \subset O_e$. We expect segments that discuss specific product aspects to be better candidates for useful summaries. We hypothesize that *general* comments mostly describe customers' overall experience, which can also be inferred by their rating, whereas aspect-related comments provide specific reasons for their overall opinion. We also assume that segments conveying highly positive or negative sentiment are more likely to present informative opinions compared to neutral ones, a claim supported by previous work (Angelidis and Lapata, 2018).

We describe our novel approach to aspect extraction in Section 4 and detail how we combine aspect, sentiment, and redundancy information to produce opinion summaries in Section 5.

4 Aspect Extraction

Our work builds on the aspect discovery model developed by He et al. (2017), which we extend to facilitate the accurate extraction of aspect-specific review segments in a more realistic setting. In this section, we first describe their approach, point out its shortcomings, and then present the extensions and modifications introduced in our *Multi-Seed Aspect Extractor* (MATE) model.

4.1 Aspect-Based Autoencoder

The *Aspect-Based Autoencoder* (ABAE; He et al. 2017) is an adaptation of the *Relationship Modeling Network* (Iyyer et al., 2016), originally designed to identify attributes of fictional book characters and their relationships. The model learns a segment-level aspect predictor without supervision by attempting to reconstruct the input segment's encoding as a linear combination of aspect embeddings. ABAE starts by pairing each word w with a pre-trained word embedding $\mathbf{v}_w \in \mathbb{R}^d$, thus constructing a word embedding dictionary $\mathbf{L} \in \mathbb{R}^{V \times d}$, where V is the size of the vocabulary. The model also keeps an aspect embedding dictionary $\mathbf{A} \in \mathbb{R}^{K \times d}$, where K is the number of aspects to be identified and i-th row $\mathbf{a}_i \in \mathbb{R}^d$ is a point in the word embedding space. Matrix $\mathbf{A}$ is initialized using the centroids from a

k-means clustering on the vocabulary's word embeddings.

The autoencoder, first produces a vector $\mathbf{v}_s$ for review segment $s = (w_1, \ldots, w_n)$ using an *attention encoder* that learns to attend on aspect words. A segment encoding is computed as the weighted average of word vectors:

$$\mathbf{v}_s = \sum_{i=1}^{n} c_i \mathbf{v}_{w_i} \qquad (1)$$

$$c_i = \frac{exp(u_i)}{\sum_{j=1}^{n} exp(u_j)} \qquad (2)$$

$$u_i = \mathbf{v}_{w_i}^{\mathsf{T}} \cdot \mathbf{M} \cdot \mathbf{v}'_s \,, \qquad (3)$$

where c_i is the i-th word's attention weight, $\mathbf{v}'_s$ is a simple average of the segment's word embeddings and attention matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ is learned during training.

Vector $\mathbf{v}_s$ is fed into a softmax classifier to predict a probability distribution over K aspects:

$$\mathbf{p}_s^{asp} = \mathrm{softmax}(\mathbf{W}\mathbf{v}_s + \mathbf{b}) \,, \qquad (4)$$

where $\mathbf{W} \in \mathbb{R}^{K \times d}$ and $\mathbf{b} \in \mathbb{R}^{K}$ are the classifier's weight and bias parameters. The segment's vector is then reconstructed as the weighted sum of aspect embeddings:

$$\mathbf{r}_s = \mathbf{A}^{\mathsf{T}} \mathbf{p}_s^{asp} \,. \qquad (5)$$

The model is trained by minimizing a reconstruction loss $J_r(\theta)$ that uses randomly sampled segments $n_1, n_2, \ldots, n_{k_n}$ as negative examples:[2]

$$J_r(\theta) = \sum_{s \in \mathrm{C}} \sum_{i=1}^{k_n} \max(0, 1 - \mathbf{r}_s \mathbf{v}_s + \mathbf{r}_s \mathbf{v}_{n_i}) \quad (6)$$

ABAE is essentially a neural topic model; it discovers topics which will hopefully map to aspects, without any preconceptions about the aspects themselves, a feature shared with most previous LDA-style aspect extraction approaches (Titov and McDonald, 2008a; He et al., 2017; Mukherjee and Liu, 2012). These models will set the number of topics to be discovered to a much larger number (~ 15) than the actual aspects found in the data (~ 5). This requires a many-to-one mapping between discovered topics and genuine aspects which is performed manually.

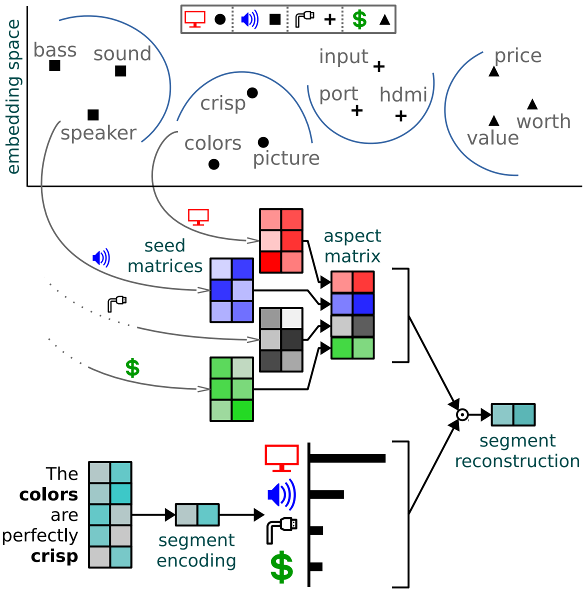

Figure 2: Multi-Seed Aspect Extractor (MATE).

4.2 Multi-Seed Aspect Extractor

Dynamic aspect extraction is advantageous since it assumes nothing more than a set of relevant reviews for a product and may discover unusual and interesting aspects (e.g., whether a plasma television has protective packaging). However, it suffers from the fact that the identified aspects are fine-grained, they have to be interpreted post-hoc, and manually mapped to coarse-grained ones.

We propose a new weakly-supervised set-up for aspect extraction which requires little human involvement. For every aspect $a_i \in A_{\mathrm{C}}$, we assume there exists a small set of seed words $\{sw_j\}_{j=1}^{l}$ which are good descriptors of a_i. We can think of these *seeds* as query terms that someone would use to search for segments discussing a_i. They can be set manually by a domain expert or selected using a small number of aspect-annotated reviews. Figure 2 (top) depicts four television aspects (*image*, *sound*, *connectivity* and *price*) and three of their seeds in word embedding space. MATE replaces ABAE's aspect dictionary with multiple seed matrices $\{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_K\}$. Every matrix $\mathbf{A}_i \in \mathbb{R}^{l \times d}$, contains one row per seed word and holds the seeds' word embeddings, as illustrated by the set of $[3 \times 2]$ matrices in Figure 2.

MATE still needs to produce an aspect matrix $\mathbf{A} \in \mathbb{R}^{K \times d}$, in order to reconstruct the input segment's embedding. We accomplish this by reducing each seed matrix to a single aspect embedding with the help of seed weight vectors $\mathbf{z}_i \in \mathbb{R}^{l}$ ($\sum_j z_{ij} = 1$), and concatenating the results, illus-

---

[2] ABAE also uses a uniqueness regularization term that is not shown here and is not used in our Multi-Seed Aspect Extractor model.

trated by the $[4 \times 2]$ aspect matrix in Figure 2:

$$\mathbf{a}_i = \mathbf{A}_i^{\mathsf{T}} \mathbf{z}_i \tag{7}$$

$$\mathbf{A} = [\mathbf{a}_1^{\mathsf{T}}; \ldots; \mathbf{a}_K^{\mathsf{T}}] \,. \tag{8}$$

The segment is reconstructed as in Equation (5). Weight vectors $\mathbf{z}_i$ can be uniform (for manually selected seeds), fixed, learned during training, or set dynamically for each input segment, based on the cosine distance of its encoding to each seed embedding. Our experiments showed that fixed weights, selected through a technique described below, result in most stable performance across domains. We only focus on this variant due to space restrictions (but provide more details in the supplementary material).

When a small number of aspect-annotated reviews are available, seeds and their fixed seed weights can be selected automatically. To obtain a ranked list of terms that are most characteristic for each aspect, we use a variant of the *clarity* scoring function which was first introduced in information retrieval (Cronen-Townsend et al., 2002). Clarity measures how much more likely it is to observe word w in the subset of segments that discuss aspect a, compared to the corpus as a whole:

$$\mathrm{score}_a(w) = t_a(w) \log_2 \frac{t_a(w)}{t(w)} \,, \tag{9}$$

where $t_a(w)$ and $t(w)$ are the l_1-normalized *tf-idf* scores of w in the segments annotated with aspect a and in all annotated segments, respectively. Higher scores indicate higher term importance and truncating the ranked list of terms gives a fixed set of seed words, as well as their seed weights by normalizing the scores to add up to one. Table 1 shows the highest ranked terms obtained for every aspect in the *televisions* domain of our corpus (see Section 6 for a detailed description of our data).

4.3 Multi-Task Objective

MATE (and ABAE) relies on the attention encoder to identify and attend to each segment's aspect-signalling words. The reconstruction objective only provides a weak training signal, so we devise a multi-task extension to enhance the encoder's effectiveness without additional annotations.

We assume that aspect-relevant words not only provide a better basis for the model's aspect-based reconstruction, but are also good indicators of the product's domain. For example, the words *colors* and *crisp*, in the segment "*The colors are perfectly crisp*" should be sufficient to infer that the segment comes from a television review, whereas the words *keys* and *type* in the segment "*The keys feel great to type on*" are more representative of the keyboard domain. Additionally, all four words are characteristic of specific aspects.

Aspect	**Top Terms**
Image	picture color quality black bright
Sound	sound speaker quality bass loud
Connectivity	hdmi port computer input component
Price	price value money worth paid
Apps & Interface	netflix user file hulu apps
Ease of Use	easy remote setup user menu
Customer Service	paid support service week replace
Size & Look	size big bigger difference screen
General	tv bought hdtv happy problem

Table 1: Highest ranked words for the television corpus according to Equation (9).

Let $\mathrm{C}_{all} = \mathrm{C}_1 \cup \mathrm{C}_2 \cup \ldots$ denote the union of multiple review corpora, where C_1 is considered *in-domain* and the rest are considered *out-of-domain*. We use $d_s \in \{d_{\mathrm{C}_1}, d_{\mathrm{C}_2}, \ldots\}$ to denote the true domain of segment s and define a classifier that uses the vectors from our segment encoder as inputs:

$$\mathbf{p}_s^{dom} = \mathrm{softmax}(\mathbf{W}_{\mathrm{C}} \mathbf{v}_s + \mathbf{b}_{\mathrm{C}}) \,, \tag{10}$$

where $\mathbf{p}_s^{dom} = \langle p^{(d_{\mathrm{C}_1})}, p^{(d_{\mathrm{C}_2})}, \ldots \rangle$ is a probability distribution over product domains for segment s and $\mathbf{W}_{\mathrm{C}}$ and $\mathbf{b}_{\mathrm{C}}$ are the classifier's weight and bias parameters. We use the negative log likelihood of the domain prediction as the objective function, combined with the reconstruction loss of Equation (5) to obtain a multi-task objective:

$$J_{\mathrm{MT}}(\theta) = J_r(\theta) - \lambda \sum_{s \in \mathrm{C}_{all}} \log p^{(d_s)} \,, \tag{11}$$

where λ controls the influence of the classification loss. Note that the negative log-likelihood is summed over all segments in C_{all}, whereas $J_r(\theta)$ is only summed over the in-domain segments $s \in \mathrm{C}_1$. It is important not to use the out-of-domain segments for segment reconstruction, as they will confuse the aspect extractor due to the aspect mismatch between different domains.

5 Opinion Summarization

We now move on to describe our opinion summarization framework which is based on the aspect extraction component discussed so far, a polarity prediction model, and a segment selection policy which identifies and discards redundant opinions.

Segment	Salience
1. The color and definition are perfect.	[+] 0.89
2. Set up was extremely easy,	[+] 0.79
3. Not worth $ 300.	[−] 0.75
4. The sound on this is horrendous.	[−] 0.52
5. The sound is TERRIBLE.	[−] 0.45
6. Nice and bright with good colors.	[+] 0.44

Table 2: Most salient opinions according to scores from Equation (12) for an LCD TV.

Domain	Products	Reviews	EDUs	Vocab
Laptop Cases	2,040 (10)	42,727 (100)	602K (1,262)	30,443
B/T Headsets	1,471 (10)	80,239 (100)	1.46M (1,344)	51,263
Boots	4,723 (10)	77,593 (100)	987K (1,198)	30,364
Keyboards	983 (10)	33,713 (100)	625K (1,396)	34,095
Televisions	1,894 (10)	56,510 (100)	1.47M (1,483)	59,051
Vacuums	1,184 (10)	68,266 (100)	1.50M (1,492)	46,259

Table 3: The OPOSUM corpus. Numbers in parentheses correspond to the human-annotated subset.

Opinion Polarity Aside from describing a product's aspects, segments also express polarity (i.e., positive or negative sentiment). We identify segment polarity with the recently proposed *Multiple Instance Learning Network* model (MILNET; Angelidis and Lapata 2018). Whilst trained on freely available document-level sentiment labels, i.e., customer ratings on a scale from 1 (negative) to 5 (positive), MILNET learns a segment-level sentiment predictor using a hierarchical, attention-based neural architecture.

Given review r consisting of segments $(s_1, \ldots, s_m)$, MILNET uses a CNN segment encoder to obtain segment vectors $(\mathbf{u}_1, \ldots, \mathbf{u}_m)$, each used as input to a segment-level sentiment classifier. For every vector $\mathbf{u}_i$, the classifier produces a sentiment prediction $\mathbf{p}_i^{stm} = \langle p_i^{(1)}, \ldots, p_i^{(M)} \rangle$, where $p_i^{(1)}$ and $p_i^{(M)}$ are probabilities assigned to the most negative and most positive sentiment class respectively. Resulting segment predictions $(\mathbf{p}_1^{stm}, \ldots, \mathbf{p}_m^{stm})$ are combined via a GRU-based attention mechanism to produce a document-level prediction $\mathbf{p}_r^{stm}$ and the model is trained end-to-end on the reviews' user ratings using negative log-likelihood.

The essential by-product of MILNET are segment-level sentiment predictions $\mathbf{p}_i^{stm}$, which are transformed into polarities pol_{s_i}, by projecting them onto the $[-1, +1]$ range using a uniformly spaced sentiment class weight vector.

Opinion Ranking Aspect predictions $\mathbf{p}_s^{asp} = \langle p_s^{(a_1)}, \ldots, p_s^{(a_K)} \rangle$ and polarities pol_s, form the opinion set $O_e = \{(s, A_s, pol_s)\}_{s \in R_e}$ for every product $e \in E_C$. For simplicity, we set the predicted aspect-set A_s to only include the aspect with the highest probability, although it is straightforward to allow for multiple aspects. We rank every opinion $o_s \in O_e$ according to its salience:

$$sal(o_s) = |pol_s| \cdot (\max_i p_s^{(a_i)} - p_s^{(\mathrm{GEN})}) \,, \quad (12)$$

where the quantity in parentheses is the probability difference between the most probable aspect and the *general* aspect. The salience score will be high for opinions that are very positive or very negative and are also likely to discuss a non-general aspect.

Opinion Selection The final step towards producing summaries is to discard potentially redundant opinions, something that is not taken into account by our salience scoring method. Table 2 shows a partial ranking of the most salient opinions found in the reviews for an LCD television. All segments provide useful information, but it is evident that segments 1 and 6 as well as 4 and 5 are paraphrases of the same opinions.

We follow previous work on multi-document summarization (Cao et al., 2015; Yasunaga et al., 2017) and use a greedy algorithm to eliminate redundancy. We start with the highest ranked opinion, and keep adding opinions to the final summary one by one, unless the cosine similarity between the candidate segment and any segment already included in the summary is lower than 0.5.

6 The OPOSUM Dataset

We created OPOSUM, a new dataset for the training and evaluation of **Opin**ion **Sum**marization models which contains Amazon reviews from six product domains: *Laptop Bags*, *Bluetooth Headsets*, *Boots*, *Keyboards*, *Televisions*, and *Vacuums*. The six training collections were created by downsampling from the *Amazon Product Dataset*[3] introduced in McAuley et al. (2015) and contain reviews and their respective ratings. The reviews were segmented into EDUs using a publicly available RST parser (Feng and Hirst, 2012).

To evaluate our methods and facilitate research, we produced a human-annotated subset of the dataset. For each domain, we uniformly sampled (across ratings) 10 different products with 10 reviews each, amounting to a total of 600 reviews, to be used only for development (300) and testing (300). We obtained EDU-level aspect annotations, salience labels and gold standard opinion

---

[3] `http://jmcauley.ucsd.edu/data/amazon/`

Aspect Extraction (F1)	L. Bags	B/T H/S	Boots	Keyb/s	TVs	Vac/s	**AVG**
Majority	37.9	39.8	37.1	43.2	41.7	41.6	40.2
ABAE	38.1	37.6	35.2	38.6	39.5	38.1	37.9
ABAE$_{init}$	41.6	48.5	41.2	41.3	45.7	40.6	43.2
MATE	46.2	52.2	45.6	43.5	48.8	42.3	46.4
MATE+MT	**48.6**	**54.5**	**46.4**	**45.3**	**51.8**	**47.7**	**49.1**

Salience (MAP/P@5)	L. Bags	B/T H/S	Boots	Keyb/s	TVs	Vac/s	**AVG**
MilNet	21.8 / 40.0	19.8 / 36.7	17.0 / 39.3	14.1 / 28.0	14.3 / 36.0	14.6 / 31.3	16.9 / 35.2
ABAE$_{init}$	19.9 / 48.5	27.5 / 49.7	13.8 / 28.1	19.0 / 44.9	16.8 / 42.4	16.1 / 34.0	18.8 / 41.3
MATE	23.0 / 57.1	30.9 / 50.7	15.4 / 31.9	21.0 / 43.1	18.7 / 44.7	19.9 / 44.0	21.5 / 45.2
MATE+MT	26.3 / 60.8	37.5 / 66.7	17.3 / 33.6	20.9 / 44.9	23.6 / 48.0	22.4 / 43.9	24.7 / 49.6
MilNet+ABAE$_{init}$	27.1 / 56.0	33.5 / 66.5	19.3 / 34.8	22.4 / 51.7	19.0 / 43.7	20.8 / 43.5	23.7 / 49.4
MilNet+MATE	28.2 / 54.7	36.0 / 66.5	21.7 / 39.3	24.0 / 52.0	20.8 / 46.1	23.5 / 49.3	25.7 / 51.3
MilNet+MATE+MT	**32.1 / 69.2**	**40.0 / 74.7**	**23.3 / 40.4**	**24.8 / 56.4**	**23.8 / 52.8**	**26.0 / 53.1**	**28.3 / 57.8**

Table 4: Experimental results for the identification of aspect segments (top) and the retrieval of salient segments (bottom) on OpoSum's six product domains and overall (AVG).

summaries, as described below. Statistics are provided in Table 3 and in supplementary material.

Aspects For every domain, we pre-selected nine representative aspects, including the *general* aspect. We presented the EDU-segmented reviews to three annotators and asked them to select the aspects discussed in each segment (multiple aspects were allowed). Final labels were obtained using a majority vote among annotators. Inter-annotator agreement across domains and annotated segments using Cohen's Kappa coefficient was $K = 0.61$ ($N =$ 8,175, $k = 3$).

Opinion Summaries We produced opinion summaries for the 60 products in our benchmark using a two-stage procedure. First, all reviews for a product were shown to three annotators. Each annotator read the reviews one-by-one and selected the subset of segments they thought best captured the most important and useful comments, without taking redundancy into account. This phase produced binary *salience* labels against which we can judge the ability of a system to identify important opinions. Again, using the Kappa coefficient, agreement among annotators was $K = 0.51$ ($N =$ 8,175, $k = 3$).[4] In the second stage, annotators were shown the salient segments they identified (for every product) and asked to create a final extractive summary by choosing opinions based on their popularity, fluency and clarity, while avoiding redundancy and staying under a budget of 100 words. We used ROUGE (Lin and Hovy, 2003) as a proxy to inter-annotator agreement. For every product, we treated one reference summary as system output and computed how it agrees with the rest. ROUGE scores are reported in Table 5 (last row).

7 Experiments

In this section, we discuss implementation details and present our experimental setup and results. We evaluate model performance on three subtasks: aspect identification, salient opinion extraction, and summary generation.

Implementation Details Reviews were lemmatized and stop words were removed. We initialized MATE using 200-dimensional word embeddings trained on each product domain using skip-gram (Mikolov et al., 2013) with default parameters. We used 30 seed words per aspect, obtained via Equation (9). Word embeddings $\mathbf{L}$, seed matrices $\{\mathbf{A}_i\}_{i=1}^{K}$ and seed weight vectors $\{\mathbf{z}_i\}_{i=1}^{K}$ were fixed throughout training. We used the Adam optimizer (Kingma and Ba, 2014) with learning rate 10^{-4} and mini-batch size 50, and trained for 10 epochs. We used 20 negative examples per input for the reconstruction loss and, when used, the multi-tasking coefficient λ was set to 10. Seed words and hyperparameters were selected on the development set and we report results on the test set, averaged over 5 runs.

Aspect Extraction We trained aspect models on the collections of Table 3 and evaluated their predictions against the human-annotated portion of each corpus. Our MATE model and its multi-task counterpart (MATE+MT) were compared against a majority baseline and two ABAE variants: vanilla ABAE, where aspect matrix $\mathbf{A}$ is initialized using k-means centroids and fine-tuned during training; and ABAE$_{init}$, where rows of $\mathbf{A}$

---

[4] While this may seem moderate, Radev et al. (2003) show that inter-annotator agreement for extractive summarization is usually lower ($K < 0.30$).

are fixed to the centroids of respective seed embeddings. This allows us to examine the benefits of our multi-seed aspect representation. Table 4 (top) reports the results using micro-averaged F1. Our models outperform both variants of ABAE across domains. ABAE_{init} improves upon the vanilla model, affirming that informed aspect initialization can facilitate the task. The richer multi-seed representation of MATE, however, helps our model achieve a 3.2% increase in F1. Further improvements are gained by the multi-task model, which boosts performance by 2.7%.

Opinion Salience We are also interested in our system's ability to identify salient opinions in reviews. The first phase of our opinion extraction annotation provides us with binary salience labels, which we use as gold standard to evaluate system opinion rankings. For every product e, we score each segment $s \in R_e$ using Equation (12) and evaluate the obtained rankings via Mean Average Precision (MAP) and Precision at the 5th retrieved segment (P@5).[5] Polarity scores were produced via MilNet; we obtained aspect probabilities from ABAE_{init}, MATE, and MATE+MT. We also experimented with a variant that only uses MilNet's polarities and, additionally, with variants that ignore polarities and only use aspect probabilities.

Results are shown in Table 4 (bottom). The combined use of polarity and aspect information improves the retrieval of salient opinions across domains, as all model variants that use our salience formula of Equation (12) outperform the MilNet- and aspect-only baselines. When comparing between aspect-based alternatives, we observe that the extraction accuracy correlates with the quality of aspect prediction. In particular, ranking using MilNet+MATE+MT gives best results, with a 2.6% increase in MAP against MilNet+MATE and 4.6% against MilNet+ABAE_{init}. The trend persists even when MilNet polarities are ignored, although the quality of rankings is worse in this case.

Opinion Summaries We now turn to the summarization task itself, where we compare our best performing model (MilNet+MATE+MT), with and without a redundancy filter (RD), against the following methods: a baseline that selects segments *randomly*; a *Lead* baseline that only selects the leading segments from each review; *SumBasic*, a generic frequency-based extractive summarizer (Nenkova and Vanderwende, 2005); *LexRank*, a generic graph-based extractive summarizer (Erkan and Radev, 2004); *Opinosis*, a graph-based abstractive summarizer that is designed for opinion summarization (Ganesan et al., 2010). All extractive methods operate on the EDU level with a 100-word budget. For Opinosis, we tested an aspect-agnostic variant that takes every review segment for a product as input, and a variant that uses MATE's groupings of segments to produce and concatenate aspect-specific summaries.

Summarization	ROUGE-1	ROUGE-2	ROUGE-L
Random	35.1	11.3	34.3
Lead	35.5	15.2	34.8
SumBasic	34.0	11.2	32.6
LexRank	37.7	14.1	36.6
Opinosis	36.8	14.3	35.7
Opinosis+MATE+MT	38.7	15.8	37.4
MilNet+MATE+MT	43.5	21.7	42.8
MilNet+MATE+MT+RD	**44.1**	**21.8**	**43.3**
Inter-annotator Agreement	54.7	36.6	53.9

Table 5: Summarization results on OpoSum.

	Inform.	Polarity	Coherence	Redund.
Gold	2.04	**8.70**	**10.93**	**6.11**
This work	**9.26**	3.15	1.11	2.96
Opinosis	-12.78	-10.00	-9.08	-9.45
Lead	1.48	-1.85	-2.96	0.37

Table 6: *Best-Worst Scaling* human evaluation.

Table 5 presents ROUGE-1, ROUGE-2 and ROUGE-L F1 scores, averaged across domains. Our model (MilNet+MATE+MT) significantly outperforms all comparison systems ($p < 0.05$; paired bootstrap resampling; Koehn 2004), whilst using a redundancy filter slightly improves performance. Assisting Opinosis with aspect predictions is beneficial, however, it remains significantly inferior to our model (see the supplementary material for additional results).

We also performed a large-scale user study. For every product in the OpoSum test set, participants were asked to compare summaries produced by: a (randomly selected) human annotator, our best performing model (MilNet+MATE+MT+RD), Opinosis, and the Lead baseline. The study was conducted on the Crowdflower platform using *Best-Worst Scaling* (BWS; Louviere and Woodworth 1991; Louviere et al. 2015), a less labour-intensive alternative to paired comparisons that has been shown to produce more reliable results than rating scales (Kiritchenko and Mohammad, 2017). We arranged every 4-tuple of competing summaries into four triplets. Every triplet was

---

[5] A system's salience ranking is individually compared against labels from each annotator and we report the average.

Product domain: Televisions
Product name: Sony BRAVIA 46-Inch HDTV

Human	Plenty of ports and settings. Easy hookups to audio and satellite sources. The sound is good and strong. This TV looks very good. and the price is even better. The on-screen menu/options is quite nice. and the internet apps work as expected. The picture is clear and sharp. which is TOO SLOW to stream HD video... The software and apps built into this TV. are difficult to use and setup. Their service is handled off shore making. communication a bit difficult. :(
LexRank	Get a Roku or Netflix box. I watch cable, Netflix, Hulu Plus, YouTube videos and computer movie files on it. Sound is good much better. DO NOT BUY! this SONY Bravia ‘ Smart ’ TV... and avoid the Sony apps at all costs. Because of these two issues, I returned the Sony TV. Also you can change the display and sound settings on each port. However, the streaming speed for netflix is just down right terrible. Most of the time I just quit. Since I do not own the cable box, So, I have the cable.
Opinosis	The picture and not bright at all even compared to my 6-year old sony lcd tv. It will not work with an hdmi. Connection because of a conflict with comcast’s dhcp. Being generous because I usuallly like the design and attention to detail of sony products). I am very disappointed with this tv for two reasons: picture brightness and channel menu. Numbers of options available in the on-line area of the tv are numerous and extremely useful. Wow look at the color, look at the sharpness of the picture, amazing and the amazing.
This work	Plenty of ports and settings and have been extremely happy with it. The sound is good and strong. The picture is beautiful. And the internet apps work as expected. And the price is even better. Unbelieveable picture and the setup is so easy. Wow look at the color, look at the sharpness of the picture. The Yahoo! widgets do not work. And avoid the Sony apps at all costs. Communication a bit difficult. :(

Figure 3: Human and system summaries for a product in the *Televisions* domain.

shown to three crowdworkers, who were asked to decide which summary was *best* and which one was *worst* according to four criteria: *Informativeness* (How much useful information about the product does the summary provide?), *Polarity* (How well does the summary highlight positive and negative opinions?), *Coherence* (How coherent and easy to read is the summary?) *Redundancy* (How successfully does the summary avoid redundant opinions?).

For every criterion, a system’s score is computed as the percentage of times it was selected as best minus the percentage of times it was selected as worst (Orme, 2009). The scores range from -100 (unanimously worst) to +100 (unanimously best) and are shown in Table 6. Participants favored our model over comparison systems across all criteria (all differences are statistically significant at $p < 0.05$ using post-hoc HD Tukey tests). Human summaries are generally preferred over our model, however the difference is significant only in terms of coherence ($p < 0.05$).

Finally, Figure 3 shows example summaries for a product from our televisions domain, produced by one of our annotators and by 3 comparison systems (LexRank, Opinosis and our MILNET+MATE+MT+RD). The human summary is primarily focused on aspect-relevant opinions, a characteristic that is also captured to a large extent by our method. There is substantial overlap between extracted segments, although our redundancy filter fails to identify a few highly similar opinions (e.g., those relating to the picture quality). The LexRank summary is inferior as it only identifies a few useful opinions, and instead selects many general or non-opinionated comments. Lastly, the abstractive summary of Opinosis does a good job of capturing opinions about specific aspects but lacks in fluency, as it produces grammatical errors. For additional system outputs, see supplementary material.

8 Conclusions

We presented a weakly supervised neural framework for aspect-based opinion summarization. Our method combined a seeded aspect extractor that is trained under a multi-task objective without direct supervision, and a multiple instance learning sentiment predictor, to identify and extract useful comments in product reviews. We evaluated our weakly supervised models on a new opinion summarization corpus across three subtasks, namely aspect identification, salient opinion extraction, and summary generation. Our approach delivered significant improvements over strong baselines in each of the subtasks, while a large-scale judgment elicitation study showed that crowdworkers favor our summarizer over competitive extractive and abstractive systems.

In the future, we plan to develop a more integrated approach where aspects and sentiment orientation are jointly identified, and work with additional languages and domains. We would also like to develop methods for abstractive opinion summarization using weak supervision signals.

Acknowledgments We gratefully acknowledge the financial support of the European Research Council (award number 681760).

References

Stefanos Angelidis and Mirella Lapata. 2018. Multiple Instance Learning Networks for Fine-Grained Sentiment Analysis. *Transactions of the Association for Computational Linguistics*, 6:17–31.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from RST discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218, Lisbon, Portugal.

Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George Reis, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *Proceedings of the WWW Workshop on NLP Challenges in the Information Explosion Era (NLPIX)*, Beijing, China.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2017. Improving multi-document summarization via text classification. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3053–3059, San Francisco, California, USA.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summarization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2153–2159, Austin, Texas, USA.

Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2002. Predicting query performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 299–306, New York, NY, USA.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Wei Vanessa Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 60–68, Jeju Island, Korea.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 340–348, Beijing, China.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 388–397, Vancouver, Canada.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceeding of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177, Seattle, Washington, USA.

Minqing Hu and Bing Liu. 2006. Opinion extraction and summarization on the web. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1621–1624, Boston, Massachusettes, USA.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544, San Diego, California.

Hyun Duk Kim, Kavita Ganesan, Parikshit Sondhi, and ChengXiang Zhai. 2011. Comprehensive review of opinion summarization. Technical report.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, Doha, Qatar.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Svetlana Kiritchenko and Saif Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 465–470, Vancouver, Canada.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain.

Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth. 2015. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606, Sydney, Australia.

Lun-Wei Ku, Yun-Ting Liang, and Hsin-Hsi Chen. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *AAAI Syposium on Computational Approaches to Analysing Weblogs*, pages 100–107, Palo Alto, California, USA.

Kevin Lerman, Sasha Blair-Goldensohn, and Ryan McDonald. 2009. Sentiment summarization: Evaluating and learning user preferences. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 514–522. Association for Computational Linguistics.

Junyi Jessy Li, Kapil Thadani, and Amanda Stent. 2016. The role of discourse units in near-extractive summarization. In *Proceedings of the SIGDIAL 2016 Conference, The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 137–147, Los Angeles, California, USA.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 71–78. Association for Computational Linguistics.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. *Mining Text Data, Springer*, pages 415–463.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443, Lisbon, Portugal.

Jordan J Louviere, Terry N Flynn, and Anthony Alfred John Marley. 2015. *Best-worst scaling: Theory, methods and applications*. Cambridge University Press.

Jordan J Louviere and George G Woodworth. 1991. Best-worst scaling: A model for the largest difference judgments. *University of Alberta: Working Paper*.

Yue Lu, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *Proceedings of the 18th International Conference on World Wide Web*, pages 131–140, Madrid, Spain.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52, Santiago, Chile.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *Proceedings of the 16th International Conference on World Wide Web*, pages 171–180, Banff, Alberta, Canada.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, California, USA.

Arjun Mukherjee and Bing Liu. 2012. Modeling review comments. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 320–329, Jeju Island, Korea.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. Technical report.

Bryan Orme. 2009. Maxdiff analysis: Simple counting, individual-level logit, and HB. Technical report.

Nikolaos Pappas and Andrei Popescu-Belis. 2017. Explicit document modeling through weighted multiple-instance learning. *Journal of Artificial Intelligence Research*, 58:591–626.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 339–346, Vancouver, British Columbia, Canada.

Dragomir R. Radev, Simone Teufel, Horacio Saggion, Wai Lam, John Blitzer, Hong Qi, Arda Çelebi, Danyu Liu, and Elliott Drabek. 2003. Evaluation challenges in large-scale document summarization. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 375–382. Association for Computational Linguistics.

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

Ivan Titov and Ryan McDonald. 2008a. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 308–316, Columbus, Ohio, USA.

Ivan Titov and Ryan McDonald. 2008b. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International Conference on World Wide Web*, pages 111–120, Beijing, China.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 47–57. Association for Computational Linguistics.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 616–626, Austin, Texas, USA.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, Vancouver, Canada.

Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2979–2985, New York, NY, USA.

Ying Zhao, George Karypis, and Usama Fayyad. 2005. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168.

CARER: Contextualized Affect Representations for Emotion Recognition

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, Yi-Shin Chen*

National Tsing Hua University
Hsinchu, Taiwan
{ellfae, tobbymailbox, yenhao0218, phejimlin, yishin} @gmail.com

Abstract

Emotions are expressed in nuanced ways, which varies by collective or individual experiences, knowledge, and beliefs. Therefore, to understand emotion, as conveyed through text, a robust mechanism capable of capturing and modeling different linguistic nuances and phenomena is needed. We propose a semi-supervised, graph-based algorithm to produce rich structural descriptors which serve as the building blocks for constructing contextualized affect representations from text. The pattern-based representations are further enriched with word embeddings and evaluated through several emotion recognition tasks. Our experimental results demonstrate that the proposed method outperforms state-of-the-art techniques on emotion recognition tasks.

1 Introduction

Emotions reflect different users' perspectives towards actions and events, therefore they are innately expressed in dynamic linguistic forms. Capturing these linguistic variations is challenging because it involves knowledge of linguistic phenomena such as slang and coded words. Previous methods model these linguistic behaviours through rule-based (Volkova and Bachrach, 2016) and statistics-based approaches (Becker et al., 2017). These methods construct features that rely on hand-crafted resources; thus, they cannot properly capture the *evolving* linguistic variability found in large-scale opinionated content.

Consider the social posts *"Thanks God for everything"* and *"Tnx mom for waaaaking me two hours early. Cant get asleep now"*, a lexicon-based model may not properly represent the emotion-relevant phrases: *"waaaaking me"*, *"Thanks God"*, and *"Tnx mom"*. First, the word *"waaaaking"* doesn't exist in the English vocabulary, hence its referent may vary from its standard form, *"waking"*. Secondly, knowledge of the semantic similarity between the words *"Thanks"* and *"Tnx"* is needed to establish any relationship between the last two phrases. Even if such relationship can be established through knowledge-based techniques, it's difficult to reliably determine the association of these phrases to a group of emotions. This is because traditional methods analyze data at the sentence level, which may be less effective as compared to methods that model the corpus as a complex network (Santos et al., 2017).

We represent an emotion corpus as a graph, which may suffer less from the problems mentioned above. This method efficiently captures the global mutual use of linguistic variations found in textual information. This is particularly important for linguistic behaviour that is socially and culturally influenced, as is common in opinionated content. Other advantages of the graph approach are that minimum domain knowledge and manual effort are required to capture important *contextual* and *latent* information, which are useful to disambiguate meaning in emotional expressions.

As an overview, we first collect an emotion dataset through noisy labels, annotated via *distant supervision* as in (Go et al., 2009). The graph-based mechanism helps to construct contextualized, pattern-based emotion features, which are further enriched with word embeddings in order to preserve semantic relationship between patterns. To evaluate the quality of patterns, emotion detection models are trained using various online classifiers and deep learning models. Our main contributions are as follows: 1) A graph-based algorithm for automatic emotion-relevant feature extraction, 2) a set of emotion-rich feature representations enhanced through word embeddings, 3) and a comprehensive performance analysis of various con-

---

* Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3687–3697
Brussels, Belgium, October 31 - November 4, 2018.

ventional learning models and deep learning models used for text-based emotion recognition.

The rest of the paper is organized as follows: Section 2 discusses the relevant literature and different aspects of emotion recognition research addressed in this work; then, Section 3 provides details of the proposed methodology for extracting contextualized emotion-relevant representations; next, Section 4 lists the constructed emotion recognition models and comparison models; later, Section 5 discusses the data collection and experimental results; and finally, Section 6 further explains key insights observed from the results.

2 Related Work

Emotion Lexica: Several works use hand-crafted features and statistics-based approaches to train emotion recognition models (Blitzer et al., 2007; Wang et al., 2012; Roberts et al., 2012; Qadir and Riloff, 2013; Volkova et al., 2013; Becker et al., 2017; Saravia et al., 2016a). Some of these studies rely on affect lexicons, such as *LIWC* (Pennebaker et al., 2007) and *WordNet Affect* (Strapparava et al., 2004), to extract emotion features from a text-based corpus. A recent study trained emotion detection systems built on emoticons and hashtag features (Volkova and Bachrach, 2016). Hand-crafted features are useful for emotion recognition but are usually constrained by manually created resources. Our graph-based features are obtained in an semi-supervised manner, requiring minimum domain expertise and no dependency of linguistic resources that quickly become outdated.

Emotion Corpora: There are several affective datasets such as *SemEval-2017 Task 4* (Rosenthal et al., 2017) and *Olympic games* dataset (Sintsova et al., 2013). However, these datasets are limited by quantity. We bootstrap a set of noisy labels to obtain large-scale emotion tweets, and then perform annotation via distant supervision as in (Go et al., 2009; González-Ibánez et al., 2011; Wang et al., 2012; Mohammad and Kiritchenko, 2015; Abdul-Mageed and Ungar, 2017). In emotion recognition studies, Plutchik's wheel of emotions (Plutchik, 2001) or Ekman's six basic emotions (Ekman, 1992), are commonly adopted to define emotion categories (Mohammad, 2012; Suttles and Ide, 2013). Similar to previous works, we rely on hashtags to define our emotion categories.

Feature Representations: Recent emotion recognition systems employ *representation learning* for automatic feature extraction (Poria et al., 2016; Savigny and Purwarianti, 2017; Abdul-Mageed and Ungar, 2017). In general, a combination of word embeddings (Mikolov et al., 2013) and a convolutional neural network (CNN) performs well for sentence classification tasks (Kim, 2014; Zhang et al., 2015). These models learn features which tend to have high coverage, high adaptability, require minimum supervision, and capture contextual information to some extent. We aim to leverage them and combine them with the proposed affect representations. Our graph-based feature extraction algorithm focuses on the underlying interactions between important linguistic components. Graph analysis measurements then help to output the building blocks for constructing pattern-based features. Hence, the patterns can be constructed to capture important contextual and latent emotion-relevant information.

3 Contextualized Affect Representations

In this section, we introduce a graph-based algorithm which helps to output the building blocks used to bootstrap a set of emotion-rich representations. The structural descriptions offered by the graph are particularly efficient at automatically surfacing important information (i.e., contextual and latent information) from a large-scale emotion corpus. Two different measurements are used to surface two families of words, which are in turn used to construct contextualized, pattern-based affect representations. The patterns are further enriched using word embeddings so as to preserve semantic relationship between patterns. After the patterns are constructed, the goal is to assign a weight to each pattern, referred to as a *pattern score*, which denotes how important a pattern p is to an emotion e. In the context of emotion classification, patterns and their weights play the role of features. The graph-based feature extraction algorithm is summarized in the following steps:

Step 1 (Normalization): First, we collected two separate datasets using the Twitter API: subjective tweets S (obtained through hashtags as weak labels) and objective tweets O (obtained from Twitter feeds of news accounts).[1] Both datasets are tokenized by white-spaces and then preprocessed by applying lower case and replacing user mentions and URLs with a <*usermention*> and <*url*>

---

[1] Each dataset contains 2+ million tweets. S was collected using 339 hashtags, similar to the process in Section 5.1.

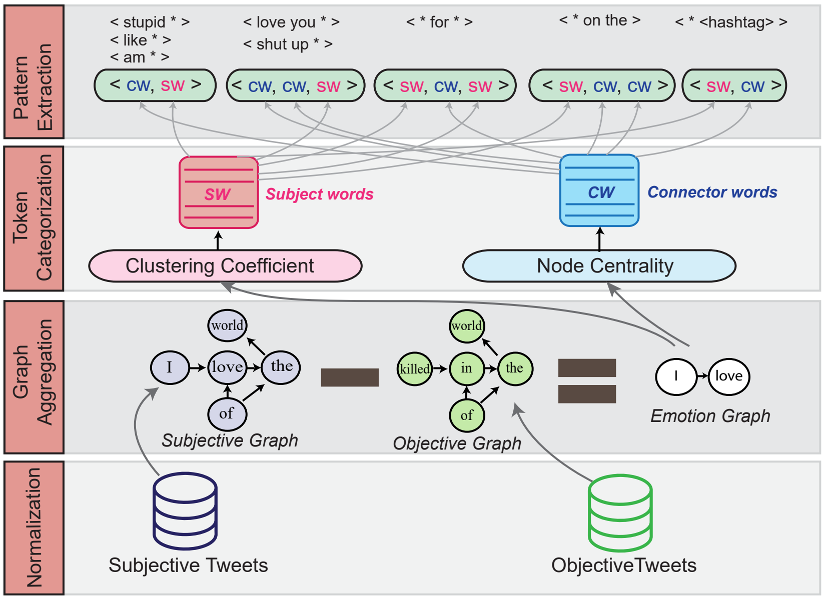

Figure 1: An overview of the important steps used to generate graph-based pattern representations.

placeholder, respectively. Hashtags are used as ground-truth in this work, so to avoid any bias we replace them with a *<hashtag>* placeholder.

Step 2 (Graph Construction): Given the normalized objective tweets O and subjective tweets S, two graphs are constructed: objective graph $G_o(V_o; A_o)$ and subjective graph $G_s(V_s; A_s)$, respectively. Vertices V is a set of nodes which represent the tokens extracted from the corpus. Edges, denoted as A, represent the relationship of words extracted using a window approach. These steps help to preserve the syntactic structure of the data. Given a post *"<usermention> last night's concert was just awesome !!!!! <hashtag>"*, the resulting arcs are: *"<usermention> → last"*, *"last → night"*, ... , *"!!!!! → <hashtag>"*.

Step 3 (Graph Aggregation): In this step we obtain a set of arcs that represent syntactic structures more common in subjective content. By adjusting graph G_s with G_o, we obtain a graph G_e, referred to as the *emotion graph*, which preserves emotion-relevant tokens and is obtained in two steps:

(1). For an arc $a_i \in A$, its normalized weight can be computed as shown in Equation 1.

$$w(a_i) = \frac{freq(a_i)}{\max_{j \in A} freq(a_j)} \tag{1}$$

where $freq(a_i)$ is the frequency of arc a_i.

(2). Subsequently, new weights for arcs $a_i \in G_e$ are assigned based on a pairwise adjustment as shown in Equation 2.

$$w(a_i) = \begin{cases} w(a_{s_i}) - w(a_{o_j}), & \text{if } a_{o_j} = a_{s_i} \in G_o \\ w(a_{s_i}), & \text{otherwise} \end{cases} \tag{2}$$

The resulting weights belonging to graph G_e were adjusted so that the most frequently occurring arcs in objective set G_o are weakened in G_e. As a result, arcs in G_e that have higher weights represent tokens that are more common in subjective content. Furthermore, arcs $a_i \in A_e$ are pruned based on a threshold ϕ_w[2].

Step 4 (Token Categorization): Two different graph measurements are used to extract two family of words from G_e. These will function as the building blocks to build contextualized patterns. We formalize this step as follows: Given an adjacency matrix $\mathbf{M}$, an entry $M_{i,j}$ is computed as:

$$M_{i,j} = \begin{cases} 1 & \text{if node } i \text{ and } j \text{ are linked in } G_e \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Then, the eigenvector centrality and clustering coefficient of all vertices in V_e are computed and used to categorize tokens into two types:

(1) **Connector Words:** To measure the influence of all nodes in graph G_e, we utilize eigenvec-

---

[2] ϕ_w is an experimentally determined threshold.

tor centrality, which is calculated as:

$$c_i = \frac{1}{\lambda} \sum_{j \in V_e} M_{i,j} c_j \tag{4}$$

where λ denotes a proportionality factor and c_i is the centrality score of node i.

Given λ as the corresponding eigenvalue, Equation 4 can be reformulated in vector notation form as $\mathbf{Mc} = \lambda \mathbf{c}$, where $\mathbf{c}$ is an eigenvector of $\mathbf{M}$. Given a selected eigenvector $\mathbf{c}$ and the eigenvector centrality score of node i, denoted as c_i, the final list of connector words, hereinafter referred to as CW, is obtained by retaining all tokens with $c_i > \phi_{eig}$ [3]. CW correspond to the set of nodes that are very frequent and have high connectivity to high-rank nodes (e.g., *"or"*, *"and"*, and *"my"*).

(2) **Subject Words:** In contrast, subject words or topical words are usually clustered together, i.e., many subject words are interconnected by the same connector words. Therefore, a coefficient is assigned to all nodes in G_e and is computed as:

$$cl_i = \frac{\sum_{j \neq i; k \neq j; k \neq i} M_{i,j} \times M_{i,k} \times M_{j,k}}{\sum_{j \neq i; k \neq j; k \neq i} M_{i,j} \times M_{i,k}} \times \frac{1}{|V_e|} \tag{5}$$

where cl_i denotes the average clustering coefficient of node i which captures the amount of interconnectivity among neighbours of node i. Similar to the connector words, the subject words, hereinafter referred to as SW, are obtained by retaining all the tokens with $cl_i > \phi_{cl}$ [4]. Examples of subjects words obtained are: *"never"* and *"life"*.

The subject words represent psychological oriented words similar to the LIWC affect lexicon (Pennebaker et al., 2007), while connector words reflect the set of most common words in the subjective tweets (e.g., pronouns, auxiliary verbs, and conjunctions). As presented by Chung and Pennebaker (2007), both connector words and subject words are important for conveying emotion. Influenced by their work, we aim to capture intricate relationships – through the graph – between these two families of words. The graph structure helps to preserve syntax and can automatically be used to surface emotion-relevant information.

One of the advantages of using graphs to represent syntactic relationships is that rare and important words are also surfaced. As shown in Table 1, informal words and misspellings, such as *"definetley"*, *"happnd"*, were surfaced. Words containing character repetitions help to express emotion intensity (e.g., *"plzzzzzzz"*, *"aaaaaaah"*, and *"yayyyyy"*). Interestingly, emotion-related coded words are also captured (e.g., *"juju"*, *"sh*t"*, *"4ever"*, and *"baobei"*[5]). All these examples show the benefit of using graph methods to capture emotion-relevant linguistic information.

Subject Words (SW)	Connector Words (CW)
baobei, juju	myy, ??!, urs, congrats
plzzzzzzz, aaaaaaah	bcoz, jus, tsk
happnd, yayyyyy	sh*t, smh, smhh, pfft
definetley, everytin	4ever, stfu, eff

Table 1: Examples of subject words and connector words automatically extracted from the emotion graph G_e.

Step 5 (Pattern Candidates): Given SW and CW, we bootstrap candidate patterns, which are more prevalent in opinionated content, while preserving syntactic structure. We provide the templates used to define the candidate patterns in Table 2. (sw and cw represent arbitrary tokens obtained from the sets SW and CW, respectively). It is important to clarify that sequences of size two and three were used in this work since this setting experimentally produced the best results.

Step 6 (Basic Pattern Extraction): A naive pattern extraction process consists of applying the syntactic templates to a dataset S_p[6] in an exhaustive manner. In addition, the sw component in each pattern is replaced with a "*" placeholder. This operation allows for unknown subject words, not present in our training corpus, to be considered when constructing features. This can enable many useful applications, such as applying the patterns to other domains. We are interested in patterns that are highly associated with subjectivity, so patterns frequently occurring above a threshold are kept and the rest are filtered out. In Table 2, we provide examples of the type of basic patterns extracted along with their corresponding templates.

3.1 Enriched Patterns

As they stand, the patterns constructed in the previous step contain limited information relevant to emotion classification. Therefore, the patterns are enriched using continuous word representations so as to preserve semantic relationship between pat-

---

[3] ϕ_{eig} is an experimentally determined threshold.

[4] ϕ_{cl} is an experimentally defined threshold.

[5] *baobei* is a Chinese word used to show strong affection.

[6] Dataset S_p (size=2+ mil.) is separately collected using similar steps as the subjective dataset S, mainly to avoid bias.

Templates	**Pattern Examples**
$< cw, sw >$	stupid * , like *, am *
$< cw, cw, sw >$	love you *, shut up *
$< sw, cw, sw >$	* for *
$< sw, cw, cw >$	* on the
$< sw, cw >$	* <hashtag>

Table 2: Examples of patterns and templates extracted through the basic pattern extraction mechanism.

terns. The motivation behind this step is to focus on patterns that may be more useful for an emotion classification task. Alternatively, the whole universe of patterns can also be used, but we show in the experiments that the former method significantly improves emotion recognition results.

Pre-trained Word Embeddings: First, we obtain Twitter-based, pre-trained word embeddings from (Deriu et al., 2017) and reweight them via a sentiment corpus through distant supervision (Read, 2005; Go et al., 2009).[7] We trained a fully connected neural network (1 hidden layer) with 10 epochs via backpropagation as in (Deriu et al., 2017). The embeddings size is $d = 52$. Note that term frequency-inverse document frequency (*tf-idf*) was used to reduce the vocabulary of words (from 140K to 20K words).

Word Clusters: We then apply agglomerative clustering to generate clusters of semantically related words through their word embedding information. To determine the quality of the clusters, they are compared with *WordNet-Affect* synsets (Strapparava et al., 2004) and tested for both *homogeneity* and *completeness*. We use *Ward's method* (Ward Jr, 1963) as the linkage criterion and *cosine distance* as the distance metric. The scikit-learn package (Pedregosa et al., 2011) was used to compute a total of $k = 1500$ clusters.

Enriched-Pattern Construction: The purpose of the word clusters is to enrich the patterns by preserving the semantic relationship between them, which is useful for classification purposes. We achieve this by revising the universe of patterns obtained from the *basic pattern extraction* step, and check to see if the words represented by the sw component exist in any of the word embedding clusters. This is done in an exhaustive manner, ensuring that all possible patterns in the dataset S_p are processed to meet the criteria. Furthermore, patterns that appear $<$ 10 times in dataset S_p are filtered out, producing a total of 476,174 patterns.

The resulting *enriched patterns*[8] now contain both the semantic information provided by the word embeddings and the contextual information provided through the graph components, hence the term *contextualized affect representations*.

3.2 Emotion Pattern Weighting

Before using the patterns for classification, they need to be weighted using a weighting mechanism such as *tf-idf* (Leskovec et al., 2014). The weights determine the importance of patterns to each emotion category. The proposed pattern weighting scheme used in this work is a customized version of *tf-idf*, coined as *pattern frequency-inverse emotion frequency* (*pf-ief*), and is defined in two steps. Firstly, we compute for pf as:

$$pf_{p,e} = \log \frac{\left(\sum_{p_i \in P_e} freq(p_i, e) \right) + 1}{freq(p, e) + 1} \tag{6}$$

where $freq(p, e)$ represents the frequency of p in e, and $pf_{p,e}$ denotes the logarithmically scaled frequency of a pattern p in a collection of texts related to emotion e.

Then we compute for ief as:

$$ief_p = \log \frac{freq(p, e) + 1}{\left(\sum_{e_j \in E} freq(p, e_j) \right) + 1} \tag{7}$$

where the inverse emotion frequency ief_p is a measure of the relevance of pattern p across all emotion categories.

Finally, we obtain a pattern score calculated as:

$$ps_{p,e} = pf_{p,e} \times ief_p \tag{8}$$

where $ps_{p,e}$ is the final score that reflects how important a pattern p is to an emotion class e.

4 Models

In this section, we present the emotion recognition models and comparison models used to evaluate the contextualized affect representations. More details are provided in Appendix A.

CARER: The proposed framework combines a multi-layer CNN architecture with a matrix form of the *enriched patterns*. The input $\mathbf{X} \in \mathbb{R}^{n \times m}$ denotes an embedding matrix where entry $X_{i,j}$

---

[7] We collected approximately 10 million tweets via sentiment emoticons (5+ mil. negative and 5+ mil. positive).

[8] Refer to Table 6 for enriched patterns examples.

represents the pattern score of enriched pattern i in emotion j.[9] $\mathbf{X}$ is fed into two 1-D convolutional layers with filters of sizes 3 and 16. The output of this process is passed through a ReLU activation function (Nair and Hinton, 2010) that produces a feature map matrix. A 1-max pooling layer (Boureau et al., 2010) of size 3 is then applied to each feature map. The results of the pooling are fed into two hidden layers of dimensions 512 and 128 in that order, each applied a dropout (Hinton et al., 2012) of 0.5 for regularization. We chose a batch size of 128 and trained for 4 epochs using Adam optimizer (Kingma and Ba, 2014). A softmax function is used to generate the final classifications. We use Keras (Chollet et al., 2015) to implement the CNN architecture.

Baseline Model: As baseline, we present a first-generation model (**CARER**$_\beta$) that employs primitive *enriched patterns*‡[10]. We adopt the CNN architecture used for **CARER**, however, this model differs in that the set of patterns used is significantly smaller as compared to the original size of the enriched patterns. The reason is because a different set of *primitive pattern templates* was used, which captured fewer patterns (187,648). This shows that the proposed method offers flexibility in terms of what templates to use and what size of patterns to generate. This could be useful in cases where there are limited computing and data resources, and for incorporating domain expertise.

Traditional Models: We also compare with various traditional methods (bag of words (**BoW**), character-level (**char**), **n-grams**, and **TF-IDF**) which are commonly used in sentence classification. To train the models we use the default stochastic gradient descent (SGD) classifier provided by scikit-learn (Pedregosa et al., 2011).

Deep Learning Models: Among the works that employ deep learning models for emotion recognition, they vary by the choice of input: *pre-trained word/character embeddings* and *end-to-end learned word/character representations*. Our work differs in that we utilize enriched graph-based representations as input. We compare with convolutional neural networks (**CNNs**), recurrent neural networks (**RNNs**), bidirectional gated recurrent neural networks (**BiGRNNs**), and word embeddings (**word2vec**) (Mikolov et al., 2013).

Emotions	Amount	Hashtags
sadness	214,454	#depressed, #grief
joy	167,027	#fun, #joy
fear	102,460	#fear, #worried
anger	102,289	#mad, #pissed
surprise	46,101	#strange, #surprise
trust	19,222	#hope, #secure
disgust	8,934	#awful, #eww
anticipation	3,975	#pumped, #ready

Table 3: Data statistics.

5 Experiments

5.1 Data

We construct a set of hashtags to collect a separate dataset of English tweets from the Twitter API. Specifically, we use the eight basic emotions: *anger*, *anticipation*, *disgust*, *fear*, *joy*, *sadness*, *surprise*, and *trust*. The hashtags (339 total) serve as noisy labels, which allow annotation via distant supervision as in (Go et al., 2009). To ensure data quality, we follow the pre-processing steps proposed by (Abdul-Mageed and Ungar, 2017), and considered the hashtag appearing in the last position of a tweet as the ground truth. We split the data into training (90%) and testing (10%) datasets. The final distribution of the data and a list of hashtag examples for each emotion are provided in Table 3. In the following section we evaluate the effectiveness of the enriched patterns on several emotion recognition tasks. We use *F1-score* as the evaluation metric, which is commonly used in emotion recognition studies due to the imbalanced nature of the emotion datasets.

5.2 Experimental Results

Traditional Features: As shown in Table 4, **TF-IDF** models produce better results than basic count-based features for both character-level and word-level feature extractors. These findings are consistent with the work of Zhang et al., (2015), where traditional methods, such as **n-gram**, were found to perform comparable to deep neural networks on various sentence classification tasks.

Pattern-based Approaches: The results of **CNN$_{\text{BASIC}}$**[11], which employs the *basic graph-based patterns* proposed in ***Step 6***, perform worse than most of the conventional approaches. Both **CARER**$_\beta$ and **CARER**, which use the *enriched patterns*, acquire better results than **CNN$_{\text{BASIC}}$**

---

[9] We use a zero-padding strategy as in (Kim, 2014).

[10] ‡ hereinafter refers to the primitive enriched patterns.

[11] CNN$_{\text{BASIC}}$ adopts CNN architecture of CARER.

and all the other conventional approaches. In fact, our method obtains the best F1-score on all eight emotions. We observed that there are significant gains in performance (↑27% and ↑12%) when using the enriched patterns as compared to the basic patterns and primitive patterns‡, respectively. This highlights the importance of the pattern enrichment procedure and the benefit of refining the pattern templates. Note that the baseline model, **CARER**$_\beta$, also performs better than all other the comparison models including the state-of-the-art methods (**DeepMoji** and **EmoNet**).

Comparison to state-of-the-art: Felbo et al., (2017) proposed a state-of-the-art emotion prediction model, **DeepMoji**, trained on billions of emoji-labeled tweets. We obtained their pre-trained model[12] and applied it to our dataset. As shown in Table 4, their model performs as well as other traditional methods. However, our model (**CARER**) significantly outperforms theirs (↑20%). Moreover, we re-implemented the GRNN model proposed in (Abdul-Mageed and Ungar, 2017). We also outperform their model (**EmoNet**) which manually trains word embeddings, similar to **DeepMoji**. The **CNN**$_{\mathbf{w2v}}$ model uses word embeddings trained on billions of tweets (Deriu et al., 2017), thus it performs better than all the other approaches, and closer to ours.

Results with Deep Learning: We offer more comparison with other various deep learning models as evaluated on Ekman's six basic emotions (i.e., *sadness*, *disgust*, *anger*, *joy*, *surprise*, and *fear*). For the **RNN**$_{\mathbf{w2v}}$ and **CNN**$_{\mathbf{char}}$ models, different inputs are used, as shown in Table 5. We feed the enriched patterns as embeddings to a **bidirectional GRNN**, which along with **CARER**$_{\mathbf{EK}}$ and **CARER**$_\beta$ outperform all the other methods.

Contextualized Approaches: **DeepMoji** is built on a stack of Bi-LSTM layers and performs much better with *six* emotion classes. However, using the enriched patterns as input, **CARER**$_{\mathbf{EK}}$[13] performs the best (81%). Note that the number of epochs used to train our models is much lower as compared to the other methods, which provides a strong case of the benefit of contextualizing features prior to training the models. Moreover, the important distinction between *connector words* and *subject words* helps to refine and surface relevant contextual information. We also show that the enriched patterns can be applied to other deep learning models besides CNN, such as **BiGRNN**, which leaves an opportunity to explore more complex architectures and fusion models in the future. More importantly, for problems that require deeper understanding of contextualized information, there is a need to go beyond traditional features and distributed representations.

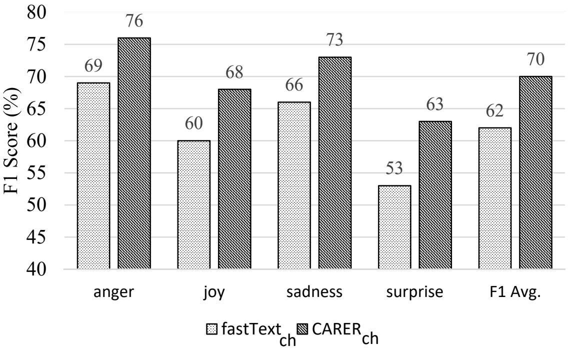

Figure 2: Comparison against Chinese word vectors.

Multilingual Capabilities: We also tested the effectiveness of the proposed feature extraction algorithm for the Chinese language. We collected Traditional Chinese datasets[14] from several of Facebook's fan pages and applied the same procedures as were done for the English datasets. User comments are considered as documents and the associated user reaction to the root post represents the emotion labels. For comparison, we obtained Chinese pre-trained word vectors computed through (Bojanowski et al., 2017), and trained a model (**fastText**$_{\mathbf{ch}}$) using the proposed CNN architecture. For our approach (**CARER**$_{\mathbf{ch}}$), we applied the same CNN architecture on the Chinese-based enriched patterns. As shown in Figure 2, our model performs significantly better on all four emotions (average F1 score of 70%). Overall, we show that the approach is not restricted to any specific language and that the enriched features are applicable to other languages and data sources. In the future, we seek to expand our methods to support other complex languages, such as Japanese, French, and Spanish, where there tends to exist fewer linguistic resources.

6 Analysis of Enriched Patterns

6.1 Pattern Coverage and Consistency

One of the advantages of the contextualized enriched patterns is that they possess high coverage

---

[12] Model obtained from github.com/bfelbo/deepmoji

[13] The proposed model trained on six emotions dataset.

[14] Details of the dataset are provided in Appendix B.

Models	Features	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	F1 Avg.
BoW	**word frequency**	0.53	0.08	0.17	0.53	0.71	0.60	0.36	0.33	0.57
BoW$_{\text{TF-IDF}}$	**TF-IDF**	0.55	0.09	0.18	0.57	0.73	0.62	0.39	0.35	0.60
n-gram	**word frequency**	0.56	0.09	0.17	0.57	0.73	0.64	0.42	0.39	0.61
n-gram$_{\text{TF-IDF}}$	**TF-IDF**	0.58	0.12	0.17	0.60	**0.75**	0.67	0.47	0.45	0.63
char_ngram	**character frequency**	0.49	0.06	0.12	0.46	0.67	0.55	0.30	0.28	0.52
char_ngram$_{\text{TF-IDF}}$	**TF-IDF**	0.53	0.07	0.15	0.53	0.71	0.59	0.35	0.31	0.57
LIWC	**affective words**	0.35	0.03	0.11	0.30	0.49	0.35	0.18	0.19	0.35
CNN$_{\text{w2v}}$	**word embeddings**	0.57	0.10	0.15	0.63	**0.75**	0.64	0.61	**0.70**	0.65
EmoNet	**word embeddings**	0.36	0.00	0.00	0.46	0.69	0.61	0.13	0.25	0.52
DeepMoji	**word embeddings**	0.60	0.00	0.03	0.49	**0.75**	0.67	0.20	0.27	0.59
CNN$_{\text{BASIC}}$	**basic patterns**	0.65	0.10	0.22	0.64	0.73	0.56	0.15	0.08	0.52
CARER$_{\beta}$	**enriched patterns**‡	0.61	0.31	0.34	0.67	**0.75**	0.68	0.60	0.55	**0.67**
CARER	**enriched patterns**	**0.74**	**0.41**	**0.43**	**0.79**	**0.83**	**0.82**	**0.76**	**0.75**	**0.79**

Table 4: Comparison of our model against various emotion recognition systems: **LIWC** uses a bag of words approach; **CNN$_{\text{w2v}}$** is the proposed CNN model and word vectors obtained from (Deriu et al., 2017); **char** refers to character-level features; **n-gram** employ unigrams, bigrams, and trigrams as features; **CNN$_{\text{BASIC}}$** uses the proposed CNN architecture with basic patterns; **EmoNet** (Abdul-Mageed and Ungar, 2017) and **DeepMoji** (Felbo et al., 2017) are state-of-the-art emotion recognition models.

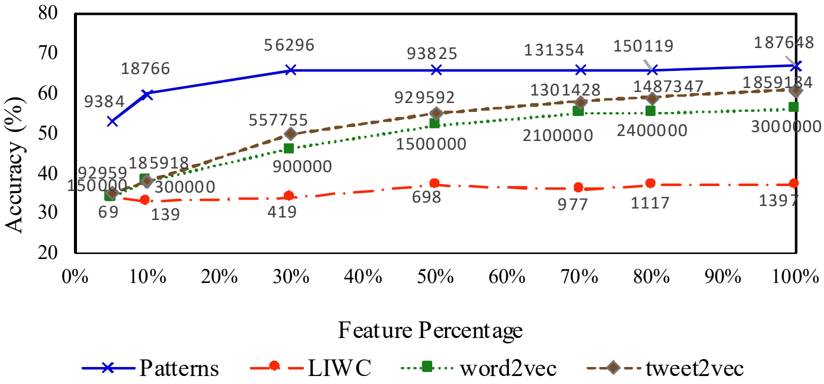

Figure 3: Consistency check for various affect resources.

Model	Input	Epochs	Accuracy
RNN$_{\text{w2v}}$	word2vec (Mikolov et al., 2013)	24	0.53
CNN$_{\text{char}}$	character embeddings (end-to-end)	50	0.63
CNN$_{\text{w2v}}$	word vectors (Deriu et al., 2017)	33	**0.69**
EmoNet	word embeddings (end-to-end)	23	0.58
DeepMoji	word embeddings (end-to-end)	100	0.63
BiGRNN	our enriched patterns‡	12	**0.68**
CARER$_{\beta}$	our enriched patterns‡	12	**0.72**
CARER$_{\text{EK}}$	our enriched patterns	12	**0.81**

Table 5: Comparison of our method against deep learning models, using Ekman's 6 emotions and the accuracy metric.

due to the way they were constructed. High coverage also means that the enriched patterns should demonstrate stability, in terms of how useful they are in an emotion classification task, even when reduced to smaller sizes. There are two cases where this could be useful: *limited data* and *limited computing resources*. Therefore, to test for pattern consistency, we randomly selected several pattern sizes[15] and trained a random forest classifier using the eight emotions dataset. This model performs comparable to **CARER$_{\beta}$** (average F1-score of 65%), and it has the benefit of faster training time, making it suitable for the aforementioned experiment. We compared with the results obtained from the **LIWC** lexicon (affect dimension) (Pennebaker et al., 2007), **word2vec** (Mikolov et al., 2013), and **tweet2vec** (Deriu et al., 2017).[16]

As shown in Figure 3, due to the limited coverage of the **LIWC** lexicon, such resources may not be feasible on evolving, large-scale datasets. In contrast, **word2vec** contains over 3 million unique word embeddings and has been proven effective for text classification. However, if we keep reducing the available word vectors of **word2vec**, which is common when there are limited computing resources, the accuracy keeps dropping at significant rates. **tweet2vec** has a similar effect. In the case of our patterns, the classification results remain relatively stable, even when reducing the patterns to 30% and 10% of the original size. These results show that the proposed features are feasible to address the text-based emotion recognition problem. Moreover, the patterns are highly beneficial where

---

[15] We employed the primitive patterns‡ used in CARER$_{\beta}$.

[16] Models were trained using the random forest implementation (depth=15 and estimators=50) provided by scikit-learn.

	Document	**GT**	**DeepMoji**	**EmoNet**	**CARER$_{\text{EK}}$**	**Enriched Patterns**
Short text	damn what a **night**	joy	surprise	sadness	joy	**what a** { *day*, *rush*, *pass* }
	want it to snow	joy	sadness	fear	joy	{ *need*, *hoping* } **it**
Rare words	**whaaaaaat.** i did not expect that at all	surprise	sadness	fear	surprise	{ *wondering*, *what* } **i**
Mixed emotions	got **thee** worst sleep ever	anger	sadness	sadness	anger	**got** { *madd*, *thatt*, *bacc* }
	what the fuck is **going** on !?	fear	anger	sadness	fear	**is** { *ends*, *finishes* }

Table 6: Classified documents in the testing data. Words in **bold blue** correspond to the sw component of one of the enriched patterns extracted from the document. Words in *italic green* represent other relevant word examples found in the cluster sw belongs to. Words in **bold pink** denote the connector word/s cw in the pattern; E.g., *"it"* is cw of patterns *"need it"* and *"hoping it"*. **GT** stands for ground truth. **DeepMoji**, **EmoNet**, and **CARER$_{\text{EK}}$** correspond to the models reported in Table 5.

there is shortage of computing and linguistic resources.

6.2 What's captured by CARER?

In Table 6, we provide samples extracted from the testing data. The examples show different cases where the comparison models struggled to capture important contextual information that helps to determine the emotion conveyed in the text. For instance, in the short text, *"damn what a night"*, only our model was able to interpret the statement as *joy* because it uses the *"what a"* pattern and its corresponding subject words to determine that this statement has a stronger association with *joy*. Our model also works well for capturing *rare words* and for *disambiguating* emotional meaning using the enriched and refined contextual information of the patterns. Rare words like *"whaaaaaat"* and *"thee"* help to implicitly convey intense emotional expressions, which are also captured and considered important by our enriched patterns. Emotion-relevant verbs, such as *"want"* and *"going"* are also considered important context that help to convey and interpret emotion. Overall, the enriched patterns efficiently capture important emotional information that other models seem to ignore.

7 Conclusion

We proposed a graph-based feature extraction mechanism to extract emotion-relevant representations in an unsupervised manner. The contextualized affect representations are further enriched with word embeddings and are used to train several deep learning-based emotion recognition models. The patterns capture implicit and explicit linguistic emotional information which significantly improves emotion recognition results.

We offered a detailed analysis demonstrating special cases where the patterns are helpful to further extract and understand emotional information from textual information. For instance, short text is a challenging problem in emotion recognition and various natural language tasks; the proposed contextualized patterns show promising results in addressing this issue by helping the models to capture nuanced information which is useful to determine the overall emotion expressed in a piece of text. The proposed method paves the way for building more interpretable emotion recognition systems which have various implications when investigating human behavioural data (Saravia et al., 2015, 2016b; Chang et al., 2016) and building empathy-aware conversational agents.

In the future work, we aim to investigate the graph-based patterns more in-depth and provide a more comprehensive and advanced theoretical discussion of how they are constructed. We also hope to keep improving the pattern weighting mechanism so as to improve the overall performance on emotion recognition tasks and minimize trade-off between pattern coverage and performance. We plan to employ transfer learning methods with the proposed enriched patterns and test on other emotion-related problems such as sentiment classification and sarcasm detection. The proposed methodology is also being expanded to support Spanish and Japanese emotion recognition tasks.

Acknowledgements

This research was supported by the Ministry of Science and Technology (#106-2221-E-007-115-MY2 and #106-3114-E-007-013).

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proc. of ACL*.

Karin Becker, Viviane Moreira, and Aline dos Santos. 2017. Multilingual emotion classification using supervised learning: Comparative experiments. *Information Processing & Management*, 53(3):684–704.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual recognition. In *Proc. of ICML*.

Chun-Hao Chang, Elvis Saravia, and Yi-Shin Chen. 2016. Subconscious crowdsourcing: A feasible data collection mechanism for mental disorder detection on social media. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 374–379. IEEE.

François Chollet et al. 2015. Keras. `https://github.com/keras-team/keras`.

Cindy Chung and James Pennebaker. 2007. The psychological functions of function words. *Social communication*, 1:343–359.

Jan Deriu, Aurelien Lucchi, Valeria De Luca, Aliaksei Severyn, Simon Müller, Mark Cieliebak, Thomas Hofmann, and Martin Jaggi. 2017. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proc. of ICWSM*.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proc of EMNLP*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(2009):12.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proc. of ACL*.

Geoffrey Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of EMNLP*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of massive datasets*. Cambridge university press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Saif M Mohammad. 2012. Emotional tweets. In *Proc. of ACL*.

Saif M Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326.

Vinod Nair and Geoffrey Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proc. of ICML*.

Febian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

James Pennebaker, Roger Booth, and Martha Francis. 2007. Linguistic inquiry and word count: Liwc [computer software]. *Austin, TX: liwc. net*.

Robert Plutchik. 2001. The nature of emotions human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American Scientist*, 89(4):344–350.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Proc. of ICDM*.

Ashequl Qadir and Ellen Riloff. 2013. Bootstrapped learning of emotion hashtags# hashtags4you. In *Proc. of Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

Jonathon Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proc. of ACL*.

Kirk Roberts, Michael Roach, Joseph Johnson, Josh Guthrie, and Sanda Harabagiu. 2012. Empatweet: Annotating and detecting emotions on twitter. In *Proc. of LREC*.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, Vancouver, Canada. Association for Computational Linguistics.

Leandro dos Santos, Edilson Corrêa Jr, Osvaldo Oliveira Jr, Diego Amancio, Letícia Mansur, and Sandra Aluísio. 2017. Enriching complex networks with word embeddings for detecting mild cognitive impairment from speech transcripts. *arXiv preprint arXiv:1704.08088*.

Elvis Saravia, Carlos Argueta, and Yi-Shin Chen. 2015. Emoviz: Mining the world's interest through emotion analysis. In *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*, pages 753–756. IEEE.

Elvis Saravia, Carlos Argueta, and Yi-Shin Chen. 2016a. Unsupervised graph-based pattern extraction for multilingual emotion classification. *Social Network Analysis and Mining*, 6(1):92.

Elvis Saravia, Chun-Hao Chang, Renaud Jollet De Lorenzo, and Yi-Shin Chen. 2016b. Midas: Mental illness detection and analysis via social media. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 1418–1421. IEEE.

Julio Savigny and Ayu Purwarianti. 2017. Emotion classification on youtube comments using word embedding. In *Proc. of ICAICTA*.

Valentina Sintsova, Claudiu-Cristian Musat, and Pearl Pu. 2013. Fine-grained emotion recognition in olympic tweets based on human computation. In *4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

Carlo Strapparava, Alessandro Valitutti, et al. 2004. Wordnet affect: an affective extension of wordnet. In *Proc. of LREC*.

Jared Suttles and Nancy Ide. 2013. Distant supervision for emotion classification with discrete binary values. In *International Conference on Intelligent Text Processing and Computational Linguistics*.

Svitlana Volkova and Yoram Bachrach. 2016. Inferring perceived demographics from user emotional tone and user-environment emotional contrast. In *Proc. of ACL*.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring sentiment in social media: Bootstrapping subjectivity clues from multilingual twitter streams. In *Proc. of ACL*.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2012. Harnessing twitter" big data" for automatic emotion identification. In *Proc. of SocialCom*.

Joe Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. of NIPS*.

Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency

Zhuang Ma
University of Pennsylvania*
zhuangma@wharton.upenn.edu

Michael Collins
Google AI Language and Columbia University†
mjcollins@google.com

Abstract

Noise Contrastive Estimation (NCE) is a powerful parameter estimation method for log-linear models, which avoids calculation of the partition function or its derivatives at each training step, a computationally demanding step in many cases. It is closely related to *negative sampling methods*, now widely used in NLP. This paper considers NCE-based estimation of *conditional* models. Conditional models are frequently encountered in practice; however there has not been a rigorous theoretical analysis of NCE in this setting, and we will argue there are subtle but important questions when generalizing NCE to the conditional case. In particular, we analyze two variants of NCE for conditional models: one based on a classification objective, the other based on a ranking objective. We show that the ranking-based variant of NCE gives consistent parameter estimates under weaker assumptions than the classification-based method; we analyze the statistical efficiency of the ranking-based and classification-based variants of NCE; finally we describe experiments on synthetic data and language modeling showing the effectiveness and trade-offs of both methods.

1 Introduction

This paper considers parameter estimation in conditional models of the form

$$p(y|x;\theta) = \frac{\exp\left(s(x,y;\theta)\right)}{Z(x;\theta)} \qquad (1)$$

where $s(x,y;\theta)$ is the unnormalized score of label y in conjunction with input x under parameters θ, $\mathcal{Y}$ is a finite set of possible labels, and $Z(x;\theta) = \sum_{y\in\mathcal{Y}} \exp\left(s(x,y;\theta)\right)$ is the partition function for input x under parameters θ.

It is hard to overstate the importance of models of this form in NLP. In log-linear models, including both the original work on maximum-entropy models (Berger et al., 1996), and later work on conditional random fields (Lafferty et al., 2001), the scoring function $s(x,y;\theta) = \theta \cdot f(x,y)$ where $f(x,y) \in \mathbb{R}^d$ is a feature vector, and $\theta \in \mathbb{R}^d$ are the parameters of the model. In more recent work on neural networks the function $s(x,y;\theta)$ is a non-linear function. In Word2Vec the scoring function is $s(x,y;\theta) = \theta_x \cdot \theta'_y$ where y is a word in the context of word x, and $\theta_x \in \mathbb{R}^d$ and $\theta'_y \in \mathbb{R}^d$ are "inside" and "outside" word embeddings x and y.

In many NLP applications the set $\mathcal{Y}$ is large. Maximum likelihood estimation (MLE) of the parameters θ requires calculation of $Z(x;\theta)$ or its derivatives at each training step, thereby requiring a summation over all members of $\mathcal{Y}$, which can be computationally expensive. This has led to many authors considering alternative methods, often referred to as "negative sampling methods", where a modified training objective is used that does not require summation over $\mathcal{Y}$ on each example. Instead negative examples are drawn from some distribution, and a objective function is derived based on binary classification or ranking. Prominent examples are the binary objective used in word2vec ((Mikolov et al., 2013), see also (Levy and Goldberg, 2014)), and the Noise Contrastive Estimation methods of (Mnih and Teh, 2012; Jozefowicz et al., 2016) for estimation of language models.

In spite of the centrality of negative sampling methods, they are arguably not well understood from a theoretical standpoint. There are clear connections to noise contrastive estimation (NCE) (Gutmann and Hyvärinen, 2012), a negative sampling method for parameter estimation in *joint* models of the form

$$p(y) = \frac{\exp\left(s(y;\theta)\right)}{Z(\theta)}; \quad Z(\theta) = \sum_{y\in\mathcal{Y}} \exp\left(s(y;\theta)\right) \qquad (2)$$

However there has not been a rigorous theoretical analysis of NCE in the estimation of *conditional* models of the form in Eq. 1, and we will argue there are subtle but important questions when generalizing NCE to the conditional case. In particular, the joint model in Eq 2 has a single partition function $Z(\theta)$ which is estimated as a param-

---

[*] Part of this work done at Google.

[†] Work done at Google.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3698–3707
Brussels, Belgium, October 31 - November 4, 2018.

eter of the model (Gutmann and Hyvärinen, 2012) whereas the conditional model in Eq 1 has a separate partition function $Z(x;\theta)$ for each value of x. This difference is critical.

We show the following (throughout we define $K \geq 1$ to be the number of negative examples sampled per training example):

- For any $K \geq 1$, a binary classification variant of NCE, as used by (Mnih and Teh, 2012; Mikolov et al., 2013), gives consistent parameter estimates under the assumption that $Z(x;\theta)$ is constant with respect to x (i.e., $Z(x;\theta) = H(\theta)$ for some function H). Equivalently, the method is consistent under the assumption that the function $s(x,y;\theta)$ is powerful enough to incorporate $\log Z(x;\theta)$.
- For any $K \geq 1$, a ranking-based variant of NCE, as used by (Jozefowicz et al., 2016), gives consistent parameter estimates under the much weaker assumption that $Z(x;\theta)$ can vary with x. Equivalently, there is no need for $s(x,y;\theta)$ to be powerful enough to incorporate $\log Z(x;\theta)$.
- We analyze the statistical efficiency of the ranking-based and classification-based NCE variants. Under respective assumptions, both variants achieve Fisher efficiency (the same asymptotic mean square error as the MLE) as $K \to \infty$.
- We discuss application of our results to approaches of (Mnih and Teh, 2012; Mikolov et al., 2013; Levy and Goldberg, 2014; Jozefowicz et al., 2016) giving a unified account of these methods.
- We describe experiments on synthetic data and language modeling evaluating the effectiveness of the two NCE variants.

2 Basic Assumptions

We assume the following setup throughout:

- We have sets $\mathcal{X}$ and $\mathcal{Y}$, where $\mathcal{X}, \mathcal{Y}$ are finite.
- There is some unknown joint distribution $p_{X,Y}(x,y)$ where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. We assume that the marginal distributions satisfy $p_X(x) > 0$ for all $x \in \mathcal{X}$ and $p_Y(y) > 0$ for all $y \in \mathcal{Y}$.
- We have training examples $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$ drawn I.I.D. from $p_{X,Y}(x,y)$.
- We have a scoring function $s(x,y;\theta)$ where θ are the parameters of the model. For example, $s(x,y;\theta)$ may be defined by a neural network.
- We use Θ to refer to the parameter space. We assume that $\Theta \subseteq \mathbb{R}^d$ for some integer d.
- We use $p_N(y)$ to refer to a distribution from which negative examples are drawn in the NCE approach. We assume that p_N satisfies $p_N(y) > 0$ for all $y \in \mathcal{Y}$.

We will consider estimation under the following two assumptions:

Assumption 2.1 *There exists some parameter value $\theta^* \in \Theta$ such that for all $(x,y) \in \mathcal{X} \times \mathcal{Y}$,*

$$p_{Y|X}(y|x) = \frac{\exp(s(x,y;\theta^*))}{Z(x;\theta^*)} \quad (3)$$

where $Z(x;\theta^*) = \sum_{y \in \mathcal{Y}} \exp(s(x,y;\theta^*))$.

Assumption 2.2 *There exists some parameter value $\theta^* \in \Theta$, and a constant $\gamma^* \in \mathbb{R}$, such that for all $(x,y) \in \mathcal{X} \times \mathcal{Y}$,*

$$p_{Y|X}(y|x) = \exp\left(s(x,y;\theta^*) - \gamma^*\right). \quad (4)$$

Assumption 2.2 is stronger than Assumption 2.1. It requires $\log Z(x;\theta^*) \equiv \gamma^*$ for all $x \in \mathcal{X}$, that is, the conditional distribution is perfectly self-normalized. Under Assumption 2.2, it must be the case that $\forall x \in \mathcal{X}$

$$\sum_y p_{Y|X}(y|x) = \sum_y \exp\{s(x,y;\theta^*) - \gamma^*\} = 1$$

There are $|\mathcal{X}|$ constraints but only $d+1$ free parameters. Therefore self-normalization is a non-trivial assumption when $|\mathcal{X}| \gg d$. In the case of language modeling, $|\mathcal{X}| = |V|^k \gg d+1$, where $|V|$ is the vocabulary size and k is the length of the context. The number of constraints grows exponentially fast.

Given a scoring function $s(x,y;\theta)$ that satisfies assumption 2.1, we can derive a scoring function s' that satisfies assumption 2.2 by defining

$$s'(x,y;\theta,\{c_x : x \in \mathcal{X}\}) = s(x,y;\theta) - c_x$$

where $c_x \in \mathbb{R}$ is a parameter for history x. Thus we introduce a new parameter c_x for each possible history x. This is the most straightforward extension of NCE to the conditional case; it is used by (Mnih and Teh, 2012). It has the clear drawback however of introducing a large number of additional parameters to the model.

3 Two Estimation Algorithms

Figure 1 shows two NCE-based parameter estimation algorithms, based respectively on *binary objective* and *ranking objective*. The input to either algorithm is a set of training examples $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$, a parameter K specifying the number of negative examples per training example, and a distribution $p_N(\cdot)$ from which negative examples are sampled. The algorithms differ only in the choice of objective function being optimized: L_B^n for binary objective, and L_R^n for ranking objective. Binary objective essentially corresponds to a problem where the scoring function $s(x, y; \theta)$ is used to construct a binary classifier that discriminates between positive and negative examples. Ranking objective corresponds to a problem where the scoring function $s(x, y; \theta)$ is used to rank the true label $y^{(i)}$ above negative examples $y^{(i,1)} \ldots y^{(i,K)}$ for the input $x^{(i)}$.

Our main result is as follows:

Theorem 3.1 *(Informal: see section 4 for a formal statement.) For any $K \geq 1$, the binary classification-based algorithm in figure 1 is consistent under Assumption 2.2, but is not always consistent under the weaker Assumption 2.1. For any $K \geq 1$, the ranking-based algorithm in figure 1 is consistent under either Assumption 2.1 or Assumption 2.2. Both algorithms achieve the same statistical efficiency as the maximum-likelihood estimate as $K \to \infty$.*

The remainder of this section gives a sketch of the argument underlying consistency, and discusses use of the two algorithms in previous work.

3.1 A Sketch of the Consistency Argument for the Ranking-Based Algorithm

In this section, in order to develop intuition underlying the ranking algorithm, we give a proof sketch of the following theorem:

Theorem 3.2 *(First part of theorem 4.1 below.) Define $\mathcal{L}_R^{\infty}(\theta) = \mathbb{E}[\mathcal{L}_R^n(\theta)]$. Under Assumption 2.1, $\bar{\theta} \in \arg\max_{\theta} \mathcal{L}_R^{\infty}(\theta)$ if and only if, for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$,*

$$p_{Y|X}(y|x) = \exp(s(x, y; \bar{\theta}))/Z(x, \bar{\theta}).$$

This theorem is key to the consistency argument. Intuitively as n increases $\mathcal{L}_R^n(\theta)$ converges to $\mathcal{L}_R^{\infty}(\theta)$, and the output to the algorithm converges to θ' such that $p(y|x; \theta') = p_{Y|X}(y|x)$ for all x, y. Section 4 gives a formal argument.

Inputs: Training examples $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$, sampling distribution $p_N(\cdot)$ for generating negative examples, an integer K specifying the number of negative examples per training example, a scoring function $s(x, y; \theta)$. Flags {BINARY = true, RANKING = false} if binary classification objective is used, {BINARY = false, RANKING = true} if ranking objective is used.

Definitions: Define $\bar{s}(x, y; \theta) = s(x, y; \theta) - \log p_N(y)$
Algorithm:

- For $i = 1 \ldots n$, $k = 1 \ldots K$, draw $y^{(i,k)}$ I.I.D. from the distribution $p_N(y)$. For convenience define $y^{(i,0)} = y^{(i)}$.

- If RANKING, define the ranking objective function

$$\mathcal{L}_R^n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log \frac{\exp(\bar{s}(x^{(i)}, y^{(i,0)}; \theta))}{\sum_{k=0}^{K} \exp(\bar{s}(x^{(i)}, y^{(i,k)}; \theta))},$$

  and the estimator $\widehat{\theta}_R = \operatorname*{argmax}_{\theta \in \Theta} \mathcal{L}_R^n(\theta)$.

- If BINARY, define the binary objective function

$$\begin{aligned} \mathcal{L}_B^n(\theta, \gamma) = &\frac{1}{n} \sum_{i=1}^{n} \Big\{ \log g(x^{(i)}, y^{(i,0)}; \theta, \gamma) \\ &+ \sum_{k=1}^{K} \log \Big(1 - g(x^{(i)}, y^{(i,k)}; \theta, \gamma)\Big) \Big\}, \end{aligned}$$

  and estimator $(\widehat{\theta}_B, \widehat{\gamma}_B) = \operatorname*{argmax}_{\theta \in \Theta, \gamma \in \Gamma} \mathcal{L}_B^n(\theta, \gamma)$, where

$$g(x, y; \theta, \gamma) = \frac{\exp\left(\bar{s}(x, y; \theta) - \gamma\right)}{\exp\left(\bar{s}(x, y; \theta) - \gamma\right) + K}.$$

- Define $\widehat{\theta} = \widehat{\theta}_R$ if RANKING and $\widehat{\theta} = \widehat{\theta}_B$ otherwise. Return $\widehat{\theta}$ and

$$\widehat{p}_{Y|X}(y|x) = \frac{\exp(s(x, y; \widehat{\theta}))}{\sum_{y \in \mathcal{Y}} \exp(s(x, y; \widehat{\theta}))}$$

Figure 1: Two NCE-based estimation algorithms, using ranking objective and binary objective respectively.

We now give a proof sketch for theorem 3.2. Consider the algorithm in figure 1. For convenience define $\bar{y}^{(i)}$ to be the vector $(y^{(i,0)}, y^{(i,1)}, \ldots, y^{(i,K)})$. Define $\alpha(x, \bar{y}) = \sum_{k=0}^{K} p_{X,Y}(x, \bar{y}_k) \prod_{j \neq k} p_N(\bar{y}_j)$, and

$$\begin{aligned} q(k|x, \bar{y}; \theta) &= \frac{\exp(\bar{s}(x, \bar{y}_k; \theta))}{\sum_{k=0}^{K} \exp(\bar{s}(x, \bar{y}_k; \theta))}, \\ \beta(k|x, \bar{y}) &= \frac{p_{X,Y}(x, \bar{y}_k) \prod_{j \neq k} p_N(\bar{y}_j)}{\alpha(x, \bar{y})} \\ &= \frac{p_{Y|X}(\bar{y}_k|x)/p_N(\bar{y}_k)}{\sum_{k=0}^{N} p_{Y|X}(\bar{y}_k|x)/p_N(\bar{y}_k)} \\ \mathrm{C}(x, \bar{y}; \theta) &= -\sum_{k=0}^{K} \beta(k|x, \bar{y}) \log q(k|x, \bar{y}; \theta) \end{aligned}$$

Intuitively, $q(\cdot|x, \bar{y}; \theta)$ and $\beta(\cdot|x, \bar{y})$ are posterior distributions over the true label $k \in \{0 \dots K\}$ given an input $x, \bar{y}$, under the parameters θ and the true distributions $p_{X,\bar{Y}}(x, \bar{y})$ respectively; $\mathrm{C}(x, \bar{y}; \theta)$ is the negative cross-entropy between these two distributions.

The proof of theorem 3.2 rests on two identities. The first identity states that the objective function is the expectation of the negative cross-entropy w.r.t. the density function $\frac{1}{K+1}\alpha(x, \bar{y})$ (see Section B.1.1 of the supplementary material for derivation):

$$\mathcal{L}_R^{\infty}(\theta) = \sum_x \sum_{\bar{y}} \frac{1}{K+1} \alpha(x, \bar{y}) \mathrm{C}(x, \bar{y}; \theta). \quad (5)$$

The second identity concerns the relationship between $q(\cdot|x, \bar{y}; \theta)$ and $\beta(\cdot|x, \bar{y})$. Under assumption 2.1, for all $x, \bar{y}$, $k \in \{0 \dots K\}$,

$$\begin{aligned} & q(k|x, \bar{y}; \theta^*) \\ = \; & \frac{p_{Y|X}(\bar{y}_k|x) Z(x; \theta^*)/p_N(y_k)}{\sum_{k=0}^{K} p_{Y|X}(\bar{y}_k|x) Z(x; \theta^*)/p_N(y_k)} \\ = \; & \beta(k|x, \bar{y}) \end{aligned} \quad (6)$$

It follows immediately through the properties of negative cross entropy that

$$\forall x, \bar{y}, \quad \theta^* \in \operatorname*{argmax}_{\theta} \mathrm{C}(x, \bar{y}; \theta) \quad (7)$$

The remainder of the argument is as follows:

- Eqs. 7 and 5 imply that $\theta^* \in \mathrm{argmax}_{\theta} \mathcal{L}_R^{\infty}(\theta)$.
- Assumption 2.1 implies that $\alpha(x, \bar{y}) > 0$ for all $x, \bar{y}$. It follows that any $\theta' \in \arg\max_{\theta} \mathcal{L}_R^{\infty}(\theta)$ satisfies

$$\begin{aligned} & \text{for all } x, \bar{y}, k, \\ & q(k|x, \bar{y}; \theta') = q(k|x, \bar{y}; \theta^*) = \beta(k|x, \bar{y}) \end{aligned} \quad (8)$$

Otherwise there would be some $x, \bar{y}$ such that $\mathrm{C}(x, \bar{y}; \theta') < \mathrm{C}(x, \bar{y}; \theta^*)$.

- Eq. 8 implies that $\forall x, y, \; p(y|x; \theta') = p(y|x; \theta^*)$. See the proof of lemma B.3 in the supplementary material.

In summary, the identity in Eq. 5 is key: the objective function in the limit, $\mathcal{L}_R^{\infty}(\theta)$, is related to a negative cross-entropy between the underlying distribution $\beta(\cdot|x, \bar{y})$ and a distribution under the parameters, $q(\cdot|x, \bar{y}; \theta)$. The parameters θ^* maximize this negative cross entropy over the space of all distributions $\{q(\cdot|x, \bar{y}; \theta), \theta \in \Theta\}$.

3.2 The Algorithms in Previous Work

To motivate the importance of the two algorithms, we now discuss their application in previous work.

Mnih and Teh (2012) consider language modeling, where $x = w_1 w_2 \dots w_{n-1}$ is a history consisting of the previous $n-1$ words, and y is a word. The scoring function is defined as

$$s(x, y; \theta) = (\sum_{i=1}^{n-1} C_i r_{w_i}) \cdot q_y + b_y - c_x$$

where r_{w_i} is an embedding (vector of parameters) for history word w_i, q_y is an embedding (vector of parameters) for word y, each C_i for $i = 1 \dots n-1$ is a matrix of parameters specifying the contribution of r_{w_i} to the history representation, b_y is a bias term for word y, and c_x is a parameter corresponding to the log normalization term for history x. Thus each history x has its own parameter c_x. The binary objective function is used in the NCE algorithm. The noise distribution $p_N(y)$ is set to be the unigram distribution over words in the vocabulary.

This method is a direct application of the original NCE method to conditional estimation, through introduction of the parameters c_x corresponding to normalization terms for each history. Interestingly, Mnih and Teh (2012) acknowledge the difficulties in maintaining a separate parameter c_x for each history, and set $c_x = 0$ for all x, noting that empirically this works well, but without giving justification.

Mikolov et al. (2013) consider an NCE-based method using the binary objective function for estimation of word embeddings. The skip-gram method described in the paper corresponds to a model where x is a word, and y is a word in the context. The vector v_x is the embedding for word x, and the vector v'_y is an embedding for word y (separate embeddings are used for x and y). The method they describe uses

$$\bar{s}(x, y; \theta) = v'_y \cdot v_x$$

or equivalently

$$s(x, y; \theta) = v'_y \cdot v_x + \log p_N(y)$$

The negative-sampling distribution $p_N(y)$ was chosen as the unigram distribution $p_Y(y)$ raised to the power $3/4$. The end goal of the method was to learn useful embeddings v_w and v'_w for each word in the vocabulary; however the method gives a consistent estimate for a model of the form

$$\begin{aligned} p(y|x) &= \frac{\exp\left(v'_y \cdot v_x + \log p_N(y)\right)}{\sum_y \exp\left(v'_y \cdot v_x + \log p_N(y)\right)} \\ &= \frac{p_N(y)\exp\left(v'_y \cdot v_x\right)}{Z(x;\theta)} \end{aligned}$$

assuming that Assumption 2.2 holds, i.e. $Z(x;\theta) = \sum_y p_N(y)\exp\left(v'_y \cdot v_x\right) \equiv H(\theta)$ which does not vary with x.

Levy and Goldberg (2014) make a connection between the NCE-based method of (Mikolov et al., 2013), and factorization of a matrix of pointwise mutual information (PMI) values of (x,y) pairs. Consistency of the NCE-based method under assumption 2.2 implies a similar result, specifically: if we define $p_N(y) = p_Y(y)$, and define $s(x,y;\theta) = v'_y \cdot v_x + \log p_N(y)$ implying $\bar{s}(x,y;\theta) = v'_y \cdot v_x$, then parameters v'_y and v_x converge to values such that

$$p(y|x) = \frac{p_Y(y)\exp\left(v'_y \cdot v_x\right)}{H(\theta)}$$

or equivalently

$$\text{PMI}(x,y) = \log\frac{p(y|x)}{p(y)} = v'_y \cdot v_x - \log H(\theta)$$

That is, following (Levy and Goldberg, 2014), the inner product $v'_y \cdot v_x$ is an estimate of the PMI up to a constant offset $H(\theta)$.

Finally, Jozefowicz et al. (2016) introduce the ranking-based variant of NCE for the language modeling problem. This is the same as the ranking-based algorithm in figure 1. They do not, however, make the connection to assumptions 2.2 and 2.1, or derive the consistency or efficiency results in the current paper. Jozefowicz et al. (2016) partially motivate the ranking-based variant throught the *importance sampling* viewpoint of Bengio and Senécal (2008). However there are two critical differences: 1) the algorithm of Bengio and Senécal (2008) does not lead to the same objective L_R^n in the ranking-based variant of NCE; instead it uses importance sampling to derive an objective that is similar but not identical; 2) the importance sampling method leads to a biased estimate of the gradients of the log-likelihood function, with the bias going to zero only as $K \to \infty$. In contrast the theorems in the current paper show that the NCE-based methods are *consistent for any value of K*. In summary, while it is tempting to view the ranking variant of NCE as an importance sampling method, the NCE-based view gives stronger guarantees for finite values of K.

4 Theory

This section states the main theorems. The supplementary material contains proofs. Throughout the paper, we use $\mathbb{E}_X[\cdot], \mathbb{E}_Y[\cdot], \mathbb{E}_{X,Y}[\cdot], \mathbb{E}_{Y|X=x}[\cdot]$ to represent the expectation w.r.t. $p_X(\cdot), p_Y(\cdot), p_{X,Y}(\cdot,\cdot), p_{Y|X}(\cdot|x)$. We use $\|\cdot\|$ to denote either the l_2 norm when the operand is a vector or the spectral norm when the operand is a matrix. Finally, we use $\Rightarrow$ to represent converge in distribution. Recall that we have defined

$$\bar{s}(x,y;\theta) = s(x,y;\theta) - \log p_N(y).$$

4.1 Ranking

In this section, we study noise contrastive estimation with ranking objective under Assumption 2.1. First consider the following function:

$$L_R^\infty(\theta) = \sum_{x,y_0,\cdots,y_K} p_{X,Y}(x,y_0)\prod_{i=1}^{K} p_N(y_i) \times \log\left(\frac{\exp(\bar{s}(x,y_0;\theta))}{\sum_{k=0}^{K}\exp(\bar{s}(x,y_k;\theta))}\right).$$

By straightforward calculation, one can find that

$$L_R^\infty(\theta) = \mathbb{E}\left[L_R^n(\theta)\right].$$

Under mild conditions, $L_R^n(\theta)$ converges to $L_R^\infty(\theta)$ as $n \to \infty$. Denote the set of maximizers of $L_R^\infty(\theta)$ by Θ_R^*, that is

$$\Theta_R^* = \arg\max_{\theta\in\Theta} L_R^\infty(\theta).$$

The following theorem shows that any parameter vector $\bar{\theta} \in \Theta_R^*$ if and only if it gives the correct conditional distribution $p_{Y|X}(y|x)$.

Assumption 4.1 *(Identifiability). For any $\theta \in \Theta$, if there exists a function $c(x)$ such that $s(x,y;\theta) - s(x,y;\theta^*) \equiv c(x)$ for all $(x,y) \in \mathcal{X} \times \mathcal{Y}$, then $\theta = \theta^*$ and thus $c(x) = 0$ for all x.*

Theorem 4.1 *Under Assumption 2.1, $\bar{\theta} \in \Theta_R^*$ if and only if, for all $(x,y) \in \mathcal{X} \times \mathcal{Y}$,*

$$p_{Y|X}(y|x) = \exp(s(x,y;\bar{\theta}))/Z(x,\bar{\theta}).$$

In addition, Θ_R^* is a singleton if and only if Assumption 4.1 holds.

Next we consider consistency of the estimation algorithm based on the ranking objective under the following regularity assumptions:

Assumption 4.2 *(Continuity). $s(x, y; \theta)$ is continuous w.r.t. θ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$.*

Assumption 4.3 *Θ_R^* is contained in the interior of a compact set $\Theta \subset \mathbb{R}^d$.*

For a given estimate $\widehat{p}_{Y|X}$ of the conditional distribution $p_{Y|X}$, define the error metric $d(\cdot, \cdot)$ by

$$d\left(\widehat{p}_{Y|X}, p_{Y|X}\right) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p_{X,Y}(x, y) \times \left(\widehat{p}_{Y|X}(y|x) - p_{Y|X}(y|x)\right)^2 .$$

For a sequence of IID observations $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)}), \ldots,$ define the sequences of estimates $(\widehat{\theta}_R^1,\ \widehat{p}_{Y|X}^1)$, $(\widehat{\theta}_R^2,\ \widehat{p}_{Y|X}^2), \ldots$ where the n^{th} estimate $(\widehat{\theta}_R^n,\ \widehat{p}_{Y|X}^n)$ is obtained by optimizing the ranking objective of figure 1 on $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})$.

Theorem 4.2 *(Consistency) Under Assumptions 2.1, 4.2, 4.3, the estimates based on the ranking objective are strongly consistent in the sense that for any fixed $K \geq 1$,*

$$\mathbb{P}\left\{ \lim_{n \to \infty} \min_{\theta^* \in \Theta_R^*} \|\widehat{\theta}_R^n - \theta^*\| = 0 \right\} = \mathbb{P}\left\{ \lim_{n \to \infty} d\left(\widehat{p}_{Y|X}^n,\ p_{Y|X}\right) = 0 \right\} = 1$$

Further, if Assumption 4.1 holds,

$$\mathbb{P}\left\{ \lim_{n \to \infty} \widehat{\theta}_R^n = \theta^* \right\} = 1.$$

Remark 4.1 *Thoughout the paper, all NCE estimators are defined for some fixed K. We suppress the dependence on K to simplify notation (e.g. $\widehat{\theta}_R^n$ should be interpreted as $\widehat{\theta}_R^{n,K}$).*

4.2 Classification

Now we turn to the analysis of NCE with binary objective under Assumption 2.2. First consider the following function,

$$L_B^\infty(\theta, \gamma) = \sum_{x,y} \left\{ p_{X,Y}(x, y) \log\left(g(x, y; \theta, \gamma)\right) + K p_X(x) p_N(y) \log\left(1 - g(x, y; \theta, \gamma)\right) \right\}$$

One can find that

$$L_B^\infty(\theta, \gamma) = \mathbb{E}\left[L_B^n(\theta, \gamma)\right].$$

Denote the set of maximizers of $L_B^\infty(\theta, \gamma)$ by Ω_B^* :

$$\Omega_B^* = \arg \max_{\theta \in \Theta, \gamma \in \Gamma} L_B^\infty(\theta, \gamma).$$

Parallel results of Theorem 4.1, 4.2 are established as follows.

Assumption 4.4 *(Identifiability). For any $\theta \in \Theta$, if there exists some constant c such that $s(x, y; \theta) - s(x, y; \theta^*) \equiv c$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$, then $\theta = \theta^*$ and thus $c = 0$.*

Assumption 4.5 *Ω_B^* is in the interior of $\Theta \times \Gamma$ where $\Theta \subset \mathbb{R}^d, \Gamma \subset \mathbb{R}$ are compact sets.*

Theorem 4.3 *Under Assumption 2.2, $(\bar{\theta}, \bar{\gamma}) \in \Omega_B^*$ if and only if, for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$,*

$$p_{Y|X}(y|x) = \exp(s(x, y; \bar{\theta}) - \bar{\gamma})$$

for all (x, y). Ω_B^* is a singleton if and only if Assumption 4.4 holds.

Similarly we can define the sequence of estimates $(\widehat{\theta}_B^1,\ \widehat{\gamma}_B^1,\ \widehat{p}_{Y|X}^1)$, $(\widehat{\theta}_B^2,\ \widehat{\gamma}_B^2,\ \widehat{p}_{Y|X}^2), \ldots$ based on the binary objective.

Theorem 4.4 *(Consistency) Under Assumption 2.2, 4.2, 4.5, the estimates defined by the binary objective are strongly consistent in the sense that for any $K \geq 1$,*

$$\mathbb{P}\left\{ \lim_{n \to \infty} \min_{(\theta^*, \gamma^*) \in \Omega_B^*} \|(\widehat{\theta}_B^n, \widehat{\gamma}_B^n) - (\theta^*, \gamma^*)\| = 0 \right\} = \mathbb{P}\left\{ \lim_{n \to \infty} d\left(\widehat{p}_{Y|X}^n,\ p_{Y|X}\right) = 0 \right\} = 1$$

If further Assumption 4.4 holds,

$$\mathbb{P}\left\{ \lim_{n \to \infty} (\widehat{\theta}_B^n, \widehat{\gamma}_B^n) = (\theta^*, \gamma^*) \right\} = 1.$$

4.3 Counterexample

In this section, we give a simple example to demonstrate that the binary classification approach fails to be consistent when assumption 2.1 holds but assumption 2.2 fails (i.e. the partition function depends on the input).

Consider $X \in \mathcal{X} = \{x_1, x_2\}$ with marginal distribution

$$p_X(x_1) = p_X(x_2) = 1/2,$$

and $Y \in \mathcal{Y} = \{y_1, y_2\}$ generated by the conditional model specified in assumption 2.1 with the score function parametrized by $\theta = (\theta_1, \theta_2)$ and

$$s(x_1, y_1; \theta) = \log \theta_1,$$

$s(x_1, y_2; \theta) = s(x_2, y_1; \theta) = s(x_2, y_2; \theta) = \log \theta_2$.

Assume the true parameter is $\theta^* = (\theta_1^*, \theta_2^*) = (1, 3)$. By simple calculation,

$$Z(\theta^*; x_1) = 4,\ Z(\theta^*; x_2) = 6,$$
$$p_{X,Y}(x_1, y_1) = 1/8, p_{X,Y}(x_1, y_2) = 3/8,$$
$$p_{X,Y}(x_2, y_1) = p_{X,Y}(x_2, y_2) = 1/4.$$

Suppose we choose the negative sampling distribution $p_N(y_1) = p_N(y_2) = 1/2$. For any $K \geq 1$, by the Law of Large Numbers, as n goes to infinity, $L_B^n(\theta, \gamma)$ will converge to $L_B^\infty(\theta, \gamma)$. Substitute in the parameters above. One can show that

$$\begin{aligned} L_B^\infty(\theta, \gamma) = & \frac{1}{8} \log \frac{2\theta_1}{2\theta_1 + K\exp(\gamma)} \\ & + \frac{K}{4} \log \frac{K\exp(\gamma)}{2\theta_1 + K\exp(\gamma)} \\ & + \frac{7}{8} \log \frac{2\theta_2}{2\theta_2 + K\exp(\gamma)} \\ & + \frac{3K}{4} \log \frac{K\exp(\gamma)}{2\theta_2 + K\exp(\gamma)}. \end{aligned}$$

Setting the derivatives w.r.t. θ_1, θ_2 to zero, one will obtain

$$\theta_1 = \frac{1}{4}\exp(\gamma),\ \ \theta_2 = \frac{7}{12}\exp(\gamma).$$

So for any $(\widetilde{\theta}_1, \widetilde{\theta}_2, \widetilde{\gamma}) \in \operatorname{argmax}_{\theta,\gamma} L_B^\infty(\theta, \gamma)$, $(\widetilde{\theta}_1, \widetilde{\theta}_2, \widetilde{\gamma})$ will satisfy the equalities above. Then the estimated distribution $\widetilde{p}_{Y|X}$ will satisfy

$$\frac{\widetilde{p}_{Y|X}(y_1|x_1)}{\widetilde{p}_{Y|X}(y_2|x_1)} = \frac{\widetilde{\theta}_1}{\widetilde{\theta}_2} = \frac{1/4}{7/12} = \frac{3}{7},$$

which contradicts the fact that

$$\frac{p_{Y|X}(y_1|x_1)}{p_{Y|X}(y_2|x_1)} = \frac{p_{X,Y}(x_1, y_1)}{p_{X,Y}(x_1, y_2)} = \frac{1}{3}.$$

So the binary objective does not give consistent estimation of the conditional distribution.

4.4 Asymptotic Normality and Statistical Efficiency

Noise Contrastive Estimation significantly reduces the computational complexity, especially when the label space $|\mathcal{Y}|$ is large. It is natural to ask: does such scalability come at a cost? Classical likelihood theory tells us, under mild conditions, the maximum likelihood estimator (MLE) has nice properties like asymptotic normality and Fisher efficiency. More specifically, as the sample size goes to infinity, the distribution of the MLE will converge to a multivariate normal distribution, and the mean square error of the MLE will achieve the Cramer-Rao lower bound (Ferguson, 1996).

We have shown the consistency of the NCE estimators in Theorem 4.2 and Theorem 4.4. In this part of the paper, we derive their asymptotic distribution and quantify their statistical efficiency. To this end, we restrict ourselves to the case where θ^* is identifiable (i.e. Assumptions 4.1 or 4.4 hold) and the scoring function $s(x, y; \theta)$ satisfies the following smoothness condition:

Assumption 4.6 *(Smoothness). The scoring function $s(x, y; \theta)$ is twice continuous differentiable w.r.t. θ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$.*

We first introduce the following maximum-likelihood estimator.

$$\begin{aligned} \widehat{\theta}^{\,\mathrm{MLE}} &= \arg\min_\theta\ L^n_{\mathrm{MLE}}(\theta) \\ &:= \arg\min_\theta \sum_{i=1}^n \log\left(\frac{\exp(s(x^{(i)}, y^{(i)}; \theta))}{\sum_{y\in\mathcal{Y}} \exp(s(x^{(i)}, y; \theta))}\right). \end{aligned}$$

Define the matrix

$$\mathcal{I}_{\theta^*} = \mathbb{E}_X\left[\mathrm{Var}_{Y|X=x}\left[\nabla_\theta s(x, y; \theta^*)\right]\right].$$

As shown below, $\mathcal{I}_{\theta^*}$ is essentially the Fisher information matrix under the conditional model.

Theorem 4.5 *Under Assumption 2.1, 4.1, 4.3, and 4.6, if $\mathcal{I}_{\theta^*}$ is non-singular, as $n \to \infty$*

$$\sqrt{n}(\widehat{\theta}^{\,MLE} - \theta^*)\ \Rightarrow\ \mathcal{N}(0, \mathcal{I}_{\theta^*}^{-1}).$$

For any given estimator $\widehat{\theta}$, define the scaled asymptotic mean square error by

$$\mathrm{MSE}_\infty(\widehat{\theta}) = \lim_{n\to\infty} \mathbb{E}\left[\left\|\sqrt{\frac{n}{d}}\left(\widehat{\theta} - \theta^*\right)\right\|^2\right],$$

where d is the dimension of the parameter θ^*. Theorem 4.5 implies that,

$$\mathrm{MSE}_\infty(\widehat{\theta}^{\,\mathrm{MLE}}) = \mathrm{Tr}(\mathcal{I}_{\theta^*}^{-1})/d.$$

where $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix. According to classical MLE theory (Ferguson, 1996), under certain regularity conditions, this is the best achievable mean square error. So the next question to answer is: can these NCE estimators approach this limit?

Assumption 4.7 *There exist positive constants* c, C *such that* $\sigma_{\min}(\mathcal{I}_{\theta^*}) \geq c$ *and*

$$\max_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \Big\{ |s(x,y;\theta^*)|, \|\nabla_\theta s(x,y;\theta^*)\|, \\ \|\nabla_\theta^2 s(x,y;\theta^*)\| \Big\} \leq C.$$

where $\sigma_{\min}(\cdot)$ *denotes the smallest singular value.*

Theorem 4.6 (Ranking) *Under Assumption 2.1, 4.1, 4.3, 4.6, 4.7, there exists an integer* K_0 *such that for all* $K \geq K_0$*, as* $n \to \infty$

$$\sqrt{n}\left(\widehat{\theta}_R - \theta^*\right) \Rightarrow \mathcal{N}(0, \mathcal{I}_{R,K}^{-1}), \qquad (9)$$

for some matrix $\mathcal{I}_{R,K}$*. There exists a constant* C *such that for all* $K \geq K_0$*,*

$$|\,\mathrm{MSE}_\infty(\widehat{\theta}_R) - \mathrm{MSE}_\infty(\widehat{\theta}^{MLE})| \leq C/\sqrt{K} \\ \|\mathcal{I}_{R,K}^{-1} - \mathcal{I}_{\theta^*}^{-1}\| \leq C/\sqrt{K}$$

Theorem 4.7 (Binary) *Under Assumption 2.2, 4.4, 4.5, 4.6, 4.7, there exists an integer* K_0 *such that, for any* $K \geq K_0$*, as* $n \to \infty$

$$\sqrt{n}\left(\widehat{\theta}_B - \theta^*\right) \Rightarrow \mathcal{N}(0, \mathcal{I}_{B,K}^{-1}), \qquad (10)$$

for some matrix $\mathcal{I}_{B,K}$*. There exists a constant* C *such that for all* $K \geq K_0$*,*

$$|\,\mathrm{MSE}_\infty(\widehat{\theta}_B) - \mathrm{MSE}_\infty(\widehat{\theta}^{MLE})| \leq C/K \\ \|\mathcal{I}_{B,K}^{-1} - \mathcal{I}_{\theta^*}^{-1}\| \leq C/K.$$

Remark 4.2 *Theorem 4.6 and 4.7 reveal that under respective model assumptions, for any given* $K \geq K_0$ *both NCE estimators are asymptotically normal and* $\sqrt{n}$*-consistent. Moreover, both NCE estimators approach Fisher efficiency (statistical optimality) as* K *grows.*

5 Experiments

5.1 Simulations

Suppose we have a feature space $\mathcal{X} \subset \mathbb{R}^d$ with $|\mathcal{X}| = m_x$, label space $\mathcal{Y} = \{1, \cdots, m_y\}$, and parameter $\theta = (\theta_1, \cdots, \theta_{m_y}) \in \mathbb{R}^{m_y \times d}$. Then for any given sample size n, we can generate observations $(x^{(i)}, y^{(i)})$ by first sampling $x^{(i)}$ uniformly from $\mathcal{X}$ and then sampling $y^{(i)} \in \mathcal{Y}$ by the condional model

$$p(y|x;\theta) = \exp(x'\theta_y) / \sum_{y=1}^{m_y} \exp(x'\theta_y).$$

We first consider the estimation of θ by MLE and NCE-ranking. We fix $d = 4, m_x = 200, m_y = 100$ and generate $\mathcal{X}$ and the parameter θ from separate mixtures of Gaussians. We try different configurations of (n, K) and report the KL divergence between the estimated distribution and true distribution, as summarized in the left panel of figure 2. The observations are:

- The NCE estimators are consistent for any fixed K. For a fixed sample size, the NCE estimators become comparable to MLE as K increases.
- The larger the sample size, the less sensitive are the NCE estimators to K. A very small value of K seems to suffice for large sample size.

Apparently, under the parametrization above, the model is not self-normalized. To use NCE-binary, we add an extra x-dependent bias parameter b_x to the score function (i.e. $s(x,y;\theta) = x'\theta_y + b_x$) to make the model self-normalized or else the algorithm will not be consistent. Similar patterns to figure 2 are observed when varying sample size and K (see Section A.1 of the supplementary material). However this makes NCE-binary not directly comparable to NCE-ranking/MLE since its performance will be compromised by estimating extra parameters and the number of extra parameters depends on the richness of the feature space $\mathcal{X}$. To make this clear, we fix $n = 16000, d = 4, m_y = 100, K = 32$ and experiment with $m_x = 100, 200, 300, 400$. The results are summarized on the right panel of figure 2. As $|\mathcal{X}|$ increases, the KL divergence will grow while the performance of NCE-ranking/MLE is independent of $|\mathcal{X}|$. Without the x-dependent bias term for NCE-binary, the KL divergence will be much higher due to lack of consistency (0.19, 0.21, 0.24, 0.26 respectively).

5.2 Language Modeling

We evaluate the performance of the two NCE algorithms on a language modeling problem, using the Penn Treebank (PTB) dataset (Marcus et al., 1993). We choose (Zaremba et al., 2014) as the benchmark where the conditional distribution is modeled by two-layer LSTMs and the parameters are estimated by MLE (note that the current state-of-the-art is (Yang et al., 2018)). Zaremba et al. (2014) implemented 3 model configurations: "Small" , "Medium" and "Large", which have 200, 650 and 1500 units per layer respectively. We follow their setup (model size, unrolled steps, dropout ratio, etc) but train the model by maximiz-

	Small		Medium		Large	
MLE	111.5		82.7		78.4	
NCE	Ranking	Binary	Ranking	Binary	Ranking	Binary
$K = 200$	113.8	106.8	83.2	82.1	79.3	76.0
$K = 400$	112.9	105.6	82.3	81.5	77.9	75.6
$K = 800$	111.9	105.3	81.4	81.6	77.8	75.7
$K = 1600$	110.6	**104.8**	81.7	81.5	77.5	75.9
reg-MLE	105.4		79.9		77.0	
reg-Ranking ($K = 1600$)	105.4		**79.8**		**75.0**	
reg-Binary ($K = 1600$)	**104.8**		82.5		75.7	

Table 1: Perplexity on the test set of Penn Treebank. We show performance for the ranking v.s. binary loss algorithms, with different values for K, and with/without regularization.

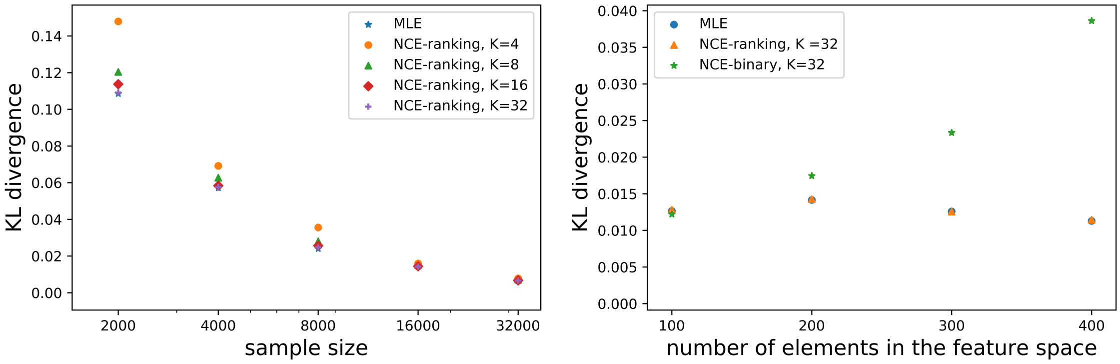

Figure 2: KL divergence between the true distribution and the estimated distribution.

ing the two NCE objectives. We use the unigram distribution as the negative sampling distribution and consider $K = 200, 400, 800, 1600$.

The results on the test set are summarized in table 1. Similar patterns are observed on the validation set (see Section A.2 of the supplementary material). As shown in the table, the performance of NCE-ranking and NCE-binary improves as the number of negative examples increases, and finally outperforms the MLE.

An interesting observation is, without regularization, the binary classification approach outperforms both ranking and MLE. This suggests the model space (two-layer LSTMs) is rich enough as to approximately incorporate the x-dependent partition function $Z(\theta; x)$, thus making the model approximately self-normalized. This motivates us to modify the ranking and MLE objectives by adding the following regularization term:

$$\frac{\alpha}{n}\sum_{i=1}^{n}\left(\log\left(\frac{1}{m}\sum_{j=1}^{m}\exp\left(\bar{s}(x^{(i)}, \widetilde{y}^{(i,j)}; \theta)\right)\right)\right)^2$$
$$\approx \alpha\,\mathbb{E}_X\left[(\log Z(x;\theta))^2\right],$$

where $\widetilde{y}^{(i,j)}, 1 \leq j \leq m$ are sampled from the noise distribution $p_N(\cdot)$. This regularization term promotes a constant partition function, that is $Z(x;\theta) \approx 1$ for all $x \in \mathcal{X}$. In our experiments, we fix m to be 1/10 of the vocabulary size, $K = 1600$ and tune the regularization parameter α. As shown in the last three rows of the table, regularization significantly improves the performance of both the ranking approach and the MLE.

6 Conclusions

In this paper we have analyzed binary and ranking variants of NCE for estimation of conditional models $p(y|x;\theta)$. The ranking-based variant is consistent for a broader class of models than the binary-based algorithm. Both algorithms achieve Fisher efficiency as the number of negative examples increases. Experiments show that both algorithms outperform MLE on a language modeling task. The ranking-based variant of NCE outperforms the binary-based variant once a regularizer is introduced that encourages self-normalization.

Acknowledgments

The authors thank Emily Pitler and Ali Elkahky for many useful conversations about the work, and David Weiss for comments on an earlier draft of the paper.

References

Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71.

Thomas Shelburne Ferguson. 1996. *A course in large sample theory*, volume 49. Chapman & Hall London.

Michael U Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2177–2185, Cambridge, MA, USA. MIT Press.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Andriy Mnih and Yee W Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1751–1758.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

CALCS: Continuously Approximating Longest Common Subsequence for Sequence Level Optimization

Semih Yavuz[*]
University of California, Santa Barbara
syavuz@cs.ucsb.edu

Chung-Cheng Chiu
Google Brain
chungchengc@google.com

Patrick Nguyen
Google Brain
drpng@google.com

Yonghui Wu
Google Brain
yonghui@google.com

Abstract

Maximum-likelihood estimation (MLE) is one of the most widely used approaches for training structured prediction models for text-generation based natural language processing applications. However, besides *exposure bias*, models trained with MLE suffer from *wrong objective* problem where they are trained to maximize the word-level correct next step prediction, but are evaluated with respect to sequence-level discrete metrics such as ROUGE and BLEU. Several variants of policy-gradient methods address some of these problems by optimizing for final discrete evaluation metrics and showing improvements over MLE training for downstream tasks like text summarization and machine translation. However, policy-gradient methods suffers from high sample variance, making the training process very difficult and unstable. In this paper, we present an alternative direction towards mitigating this problem by introducing a new objective (CALCS) based on a differentiable surrogate of longest common subsequence (LCS) measure that captures sequence-level structure similarity. Experimental results on abstractive summarization and machine translation validate the effectiveness of the proposed approach.

1 Introduction

Recently, deep neural networks have achieved state-of-the-art results in various tasks including computer vision, natural language processing, and speech processing. Specifically, neural text generation models, central focus of this work, have led to great progress in central downstream NLP tasks like text summarization, machine translation, and image captioning. For example, the abstractive summarization task, which has previously not been the popular choice for text summarization due to lack of appropriate text generation methods, has gained revived attention with the success of neural sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2015). There has been several recent work with an impressive progress on this task including (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Miao and Blunsom, 2016; See et al., 2017; Tan et al., 2017; Zhou et al., 2017). Machine translation is another central field in NLP where the emergence of neural sequence-to-sequence models has enabled viable alternative approaches (Luong et al., 2015; Bahdanau et al., 2015; Cho et al., 2014; Sutskever et al., 2014) to challenge traditional phrase-based methods (Koehn et al., 2003).

Most of the recent existing works on neural text generation are based on variants of sequence-to-sequence models with attention (Bahdanau et al., 2015) trained with Maximum-likelihood estimation (MLE) with teacher forcing. As Ranzato et al. (2016) points out in a previous work, these models have two major drawbacks. First, they are trained to maximize the probability of correct next word given the entire sequence of previous ground truth words. While, at test time, the models need to generate the entire sequence by feeding its own predictions at previous time steps. This discrepancy is called *exposure bias* and hurts the performance as the model is never exposed to its own predictions during training. The second drawback, called *wrong objective*, is due yet another discrepancy between training and testing. It refers to the critique (Ranzato et al., 2016) that MLE-trained models tend to have suboptimal performance as they are trained to maximize a convenient objective (i.e., maximum likelihood of word-level correct next step prediction) rather than a desirable sequence-level objective that correlates better with the common discrete evaluation metrics such as ROUGE (Lin and Och, 2004) for summarization,

---

[*] Work done while interning at Google Brain.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3708–3718
Brussels, Belgium, October 31 - November 4, 2018.

BLEU (Papineni et al., 2002) for translation, and word error rate for speech recognition, not log-likelihood. On the other hand, training models that directly optimize for such discrete metrics as objective is hard due to non-differentiable nature of the corresponding loss functions (Rosti et al., 2011). To address these issues, Ranzato et al. (2016) introduces an incremental learning recipe that uses a hybrid loss function combining REINFORCE (Williams, 1992) and cross-entropy. Recently, Paulus et al. (2018) also explored combining maximum-likelihood and policy gradient training for text summarization.

Towards sequence level optimization, previous works (Ranzato et al., 2016; Wu et al., 2016; Paulus et al., 2018) employ reinforcement learning (RL) with a policy-gradient approach which works around the difficulty of differentiating the reward function by using it as a weight. However, REINFORCE is known to suffer from high sample variance and credit assignment problems which makes the training process difficult and unstable besides resulting in models that are hard to reproduce (Henderson et al., 2018).

In this paper, we propose an alternative approach for sequence-level training with longest common subsequence (LCS) metric that measures the sequence-level structure similarity between two sequences. We essentially introduce a continuous approximation to the discrete LCS metric which can be directly optimized against using standard gradient-based methods. Our proposed approach has the advantage of being able to directly optimize for a surrogate reward as opposed to using the exact reward only as a weight as in RL-inspired works. Hence, it provides a viable alternative perspective to policy-gradient methods for side stepping the non-differentiability with respect to the exact reward. In addition, it simultenuously combats the *exposure bias* problem through exposing the model to its own predictions while computing our approximation to LCS metric.

To this end, we introduce a new learning recipe that incorporates the aformentioned continuous approximation to LCS metric (CALCS) as an additional objective on top of maximum-likelihood loss in existing neural text generation models. We evaluate the proposed approach on abstractive text summarization and machine translation tasks. To this end, we use recently introduced *pointer-generator* network (See et al., 2017) and *transformer* (Vaswani et al., 2017) as underlying baselines for summarization and machine translation, respectively. More precisely, we start from a pre-trained baseline model with cross-entropy loss, and continue training the model to optimize for the proposed differentiable objective based on CALCS. Using this recipe, we conduct various experiments on *CNN/Daily Mail* (Hermann et al., 2015; Nallapati et al., 2016) summarization and *WMT 2014 English-to-German* machine translation tasks. Experimental results validate the effectiveness of the proposed approach on both tasks.

2 Continuously Approximating Longest Common Subsequence Metric

In this work, we explore the potential use of longest common subsequence (LCS) metric from an algorithmic point of view to address the aforementioned *wrong objective* and *exposure bias* problems. LCS metric measures a sequence-level structure similarity between discrete sequences by identifying longest co-occurring in sequence n-grams and it has been shown to correlate well with human judgments for downstream text generation tasks (Lin and Och, 2004). To this end, we propose a way to continuously approximate LCS metric and use this differentiable approximation as the objective to train text generation models rather than the exact LCS measure, which is hard to optimize for due to non-differentiability of the corresponding loss function. Although such differentiable approximation provides a unique advantage from modeling and optimization perspective, the difficulty of controlling its tightness might be a potential drawback in terms of its applicability. In this section, we will first introduce our proposed approximation to LCS metric, and then provide a natural way to control its tightness.

Consider a sequence generation problem conditioned on an input sequence $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and let $\mathbf{y} = (y_1, y_2, \ldots, y_m)$ denote its corresponding ground-truth output sequence. Let

$$f(\mathbf{x}, \mathbf{\Theta}) = \mathbf{z} = (z_1, z_2, \ldots, z_k)$$

denote hypothesis sequence obtained by greedy decoding from a generic encoder-decoder architecture for input sequence $\mathbf{x}$, where $\mathbf{\Theta}$ represents model parameters. Also, let $p_1, p_2, \ldots, p_k$ be the probability distributions over vocabulary V at decoding time steps from which $z_1, z_2, \ldots, z_k$ are generated via *argmax* operator, respectively.

2.1 CaLcs

In this section, we define our approach to continuously approximate the longest common subsequence measure (LCS), which is an unnormalized version of ROUGE-L metric (Lin and Och, 2004) (See Appendix B) that is commonly used for performance evaluation of text summarization models. The main intuition behind our approach is to relax the common necessity for hard inferences while computing discrete metrics by instead comparing discrete tokens in a soft way. Towards this end, we start by defining LCS metric.

Definition 1. *Given two sequences* $\mathbf{y}$ *and* $\mathbf{z}$ *of tokens, longest common subsequence* $LCS(\mathbf{y}, \mathbf{z})$ *is defined as the longest sequence of tokens that appear left-to-right (but not necessarily in a contiguous block) in both sequences.*

The most common and intuitive solution for computing longest common subsequence is via dynamic programming. We will briefly revisit this here as it will be useful in terms of both recall and notational convenience while describing our surrogate LCS measure. Let $r_{i,j}$ denote the longest common subsequence between prefix sequences $\mathbf{y}_{[:i]} = (y_1, y_2, \ldots, y_i)$ and $\mathbf{z}_{[:j]} = (z_1, z_2, \ldots, z_j)$ of $\mathbf{y}$ and $\mathbf{z}$, respectively. A dynamic programming solution is given by

$$r_{i,j} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ r_{i-1,j-1} + 1 & \text{if } y_i = z_j \\ max(r_{i-1,j}, r_{i,j-1}) & \text{o/w.} \end{cases} \tag{1}$$

$r_{i,j}$ for all $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, k$. It can be computed in mk iterations using the formula in Eqn 1. After computing 2D dynamic programming matrix r, we obtain $\text{LCS}(\mathbf{y}, \mathbf{z}) = r_{m,k}$.

Towards removing the dependence on hard inference for computing LCS, we now define our approximation to longest common subsequence, which we call CaLcs. At high-level, the idea is to continuously relax the original LCS measure. To this end, we leverage output probability distributions $p_1, p_2, \ldots, p_k$ as soft predictions to refine the dynamic programming formulation for original LCS. More precisely, we recursively define soft longest common subsequence $s_{i,j}$ between prefixes $\mathbf{y}_{[:i]}$ and $\mathbf{z}_{[:j]}$ in analogous to $r_{i,j}$ as follows:

$$s_{i,j} = p_j^{(y_i)}(s_{i-1,j-1} + 1) + \tag{2}$$

$$(1 - p_j^{(y_i)}) \max(s_{i-1,j}, s_{i,j-1}) \tag{3}$$

for $i, j > 0$ and $s_{i,0} = s_{0,j} = 0$, where $p_j^{(y_i)}$ denote the probability of generating y_i at j-th decoding step. Intuitively, CaLcs replaces the hard token comparison $\mathbb{1}[y_i = z_j]$ in Eq. 1 with the probability $p_j^{(y_i)}$ as in Eq. 2. Interpreting the probability $p_j^{(y_i)}$ as a continuous relaxation of discrete comparison operator $\mathbb{1}[y_i = z_j]$, $s_{i,j}$ establishes a natural continuous approximation to $r_{i,j}$. Similar to LCS, after iteratively filling up $s_{i,j}$ matrix, we define

$$\text{CaLcs}(\mathbf{y}, \mathbf{z}) = s_{m,k} \tag{4}$$

Although the proposed approximation is a natural way of relaxing/extending the hard binary comparison of discrete tokens, it is not clear how tight the approximation is, which is established in the next section.

2.2 On the Tightness of Approximation

In this section, we first discuss the tightness of the proposed approximation, and then provide a natural way of controlling it.

2.2.1 Bounding the Approximation Error

We now present a bound on the approximation error of the proposed CaLcs compared to the original LCS measure. Characterization of this bound will enable us to theoretically argue about the feasibilty of using the proposed surrogate reward function for our objective as well as controlling its tightness.

LCS measure is intrinsically monotonic by definition. We start by a lemma that establishes a similar monotonicity property for CaLcs.

Lemma 1. *[Monotonicity] The following two inequalities*

$$s_{i,j} \leq s_{i,j+1} \leq s_{i,j} + 1$$

$$s_{i,j} \leq s_{i+1,j} \leq s_{i,j} + 1$$

hold for all $0 \leq i < m$ *and* $0 \leq j < k$.

Proof. See Appendix A for the proof. □

Having established a certain monotonicity property for CaLcs, we will discuss its approximation error to the original LCS measure. Let

$$\delta_{i,j} = s_{i,j} - r_{i,j} \tag{5}$$

denote the *approximation error* of CALCS to LCS measure between generated prefix sequence $\mathbf{y}_{[:i]}$ and the ground-truth prefix $\mathbf{z}_{[:j]}$.

Lemma 2. *Let* $P_{i,j} = \{(0,0),(i_1,j_1),\dots,(i_{q-1},j_{q-1}),(i_q,j_q)\}$ *denote the path of dynamic programming algorithm for LCS ending at* $(i,j) = (i_q,j_q)$ *cell of* $m \times k$ *grid. Then,*

$$|\delta_{i,j}| < \left|\delta_{i_{q-1},j_{q-1}}\right| + (1 - max(p_j)) \tag{6}$$

where $max(p_j) = \max\{p_j^{(1)}, p_j^{(2)}, \dots, p_j^{(|V|)}\}$.

Proof. We will establish the proof by investigating two cases and combining them.

CASE 1: $z_j = y_i$.
In this case, we have

$$r_{i,j} = 1 + r_{i-1,j-1} \tag{7}$$

and

$$(i_{q-1}, j_{q-1}) = (i-1, j-1) \tag{8}$$

by 1. Using Eq. 7, we get

$$\begin{aligned}
\delta_{i,j} &= s_{i,j} - r_{i,j} \\
&= \left(1 - p_j^{(y_i)}\right) \max\left(s_{i-1,j}, s_{i,j-1}\right) \\
&\quad + \ p_j^{(y_i)} \left(s_{i-1,j-1} + 1\right) - \left(1 + r_{i-1,j-1}\right) \\
&= \left(s_{i-1,j-1} - r_{i-1,j-1}\right) \\
&\quad + \ \left(1 - p_j^{(y_i)}\right) \Big[\max(s_{i-1,j}, s_{i,j-1}) \\
&\quad - \ (1 + s_{i-1,j-1})\Big]
\end{aligned}$$

Using the definition of δ and triangle inequality, we get

$$\begin{aligned}
|\delta_{i,j}| &\leq |\delta_{i-1,j-1}| + \left(1 - p_j^{(y_i)}\right) \Big| (1 + s_{i-1,j-1}) \\
&\quad - \ \max\left(s_{i-1,j}, s_{i,j-1}\right) \Big| \\
&\leq |\delta_{i-1,j-1}| + \left(1 - p_j^{(y_i)}\right)
\end{aligned} \tag{9}$$

where inequality 9 follows from the monotonicity established by Lemma 1.

Moreover, $z_j = y_i$ implies $p_j^{(y_i)} = \max(p_j)$ because $\mathbf{z}$ is generated by greedy decoding. Plugging this in Eq. 9 and using Eq. 8, we can immediately conclude that

$$|\delta_{i,j}| < \left|\delta_{i_{q-1},j_{q-1}}\right| + (1 - \max(p_j)) \tag{10}$$

CASE 2: $z_j \neq y_i$.
By definition 1, we have

$$r_{i,j} = \max\left(r_{i-1,j}, r_{i,j-1}\right).$$

Using this identity, we obtain

$$\begin{aligned}
\delta_{i,j} &= s_{i,j} - r_{i,j} \\
&= \left(1 - p_j^{(y_i)}\right) \max\left(s_{i-1,j}, s_{i,j-1}\right) \\
&+ p_j^{(y_i)} \left(s_{i-1,j-1} + 1\right) - \max\left(r_{i-1,j}, r_{i,j-1}\right) \\
&= p_j^{(y_i)} \left[\left(1 + s_{i-1,j-1}\right) - \max\left(s_{i-1,j}, s_{i,j-1}\right)\right] \\
&+ \left[\max\left(s_{i-1,j}, s_{i,j-1}\right) - \max\left(r_{i-1,j}, r_{i,j-1}\right)\right]
\end{aligned}$$

Applying triangle inequality on the last equation above, we get

$$\begin{aligned}
|\delta_{i,j}| \leq \\
&p_j^{(y_i)} \left|\left(1 + s_{i-1,j-1}\right) - \max\left(s_{i-1,j}, s_{i,j-1}\right)\right| \\
&+ \left|\max\left(s_{i-1,j}, s_{i,j-1}\right) - \max\left(r_{i-1,j}, r_{i,j-1}\right)\right| \\
&\leq \\
&p_j^{(y_i)} \left|\left(1 + s_{i-1,j-1}\right) - \max\left(s_{i-1,j}, s_{i,j-1}\right)\right| \\
&+ \max\left(\left|s_{i-1,j} - r_{i-1,j}\right|, \left|s_{i,j-1} - r_{i,j-1}\right|\right)
\end{aligned} \tag{11}$$

$$\begin{aligned}
&= \\
&p_j^{(y_i)} \left|\left(1 + s_{i-1,j-1}\right) - \max\left(s_{i-1,j}, s_{i,j-1}\right)\right| \\
&+ \max\left(\left|\delta_{i-1,j}\right|, \left|\delta_{i,j-1}\right|\right) \\
&\leq p_j^{(y_i)} + \max\left(\left|\delta_{i-1,j}\right|, \left|\delta_{i,j-1}\right|\right)
\end{aligned} \tag{12}$$

where inequality 12 follows from again the monotonicity of $s[\cdot,\cdot]$, and inequality 11 follows from the following identity that holds true for all real numbers $a, b, c, d \geq 0$

$$|\max(a,b) - \max(c,d)| \leq \max(|a-c|, |b-d|)$$

Moreover, since $z_j \neq y_i$, we know that $p_j^{(y_i)} \neq \max(p_j)$, which implies

$$p_j^{(y_i)} \leq 1 - \max(p_i). \tag{13}$$

Combining 11 and 13 completes the proof for this case. Finally, two cases investigated above together establish the proof of Lemma 2. ☐

Lemma 2 leads to the following important corollary.

Corollary 1. *Let* $P_{i,j} = \{(0,0),(i_1,j_1),\dots,(i_q,j_q)\}$ *be the path of dynamic programming algorithm for LCS ending at* $(i,j) = (i_q,j_q)$ *cell of* $m \times k$ *grid. Then,*

$$|\delta_{i,j}| \leq \sum_{w=1}^{q} (1 - \max(p_{j_w})) \tag{14}$$

Proof. Applying Lemma 2 iteratively and using $\delta_{0,0} = 0$, we get

$$\begin{aligned} |\delta_{i,j}| &\leq \left|\delta_{i_{q-1},j_{q-1}}\right| + (1 - \max(p_{j_q})) \\ \left|\delta_{i_{q-1},j_{q-1}}\right| &\leq \left|\delta_{i_{q-2},j_{q-2}}\right| + (1 - \max(p_{j_{q-1}})) \\ \left|\delta_{i_{q-2},j_{q-2}}\right| &\leq \left|\delta_{i_{q-3},j_{q-3}}\right| + (1 - \max(p_{j_{q-2}})) \\ &\vdots \\ |\delta_{i_1,j_1}| &\leq |\delta_{0,0}| + (1 - \max(p_{j_1})) \\ |\delta_{0,0}| &\leq 0 \end{aligned}$$

Summing $(q+1)$-many inequalities above side by side and cancelling out the same terms appearing on both sides of the resulting inequality establishes the proof of corollary. □

2.2.2 Controlling the Tightness of Approximation

Corollary 1 hints for a natural way of controlling the tightness of approximation CALCS by exploiting the peakedness of model's softmax output probability distributions. More precisely, upper bound on the approximation error is represented as a sum of $1 - \max(p_j)$'s, hence the more peaked the model's output probability distributions on average, the smaller the approximation error we are guaranteed by the established bounds.

We exploit this property to control the tightness of approximation by making a modification to computation of the proposed CALCS measure. Formally, let $l_1, l_2, \ldots, l_k$ denote the unnormalized logits of the model output before applying softmax to obtain probabilities $p_1, p_2, \ldots, p_k$ at decoding time steps, respectively. Hence,

$$p_j^{(i)} = \frac{\exp(l_j^{(i)})}{\sum_i \exp(l_j^{(i)})} \tag{15}$$

Recall that CALCS is computed using p_j's. Using *peaked softmax*, we can obtain more peaked probability distributions without causing any change in the actual generated sequence $\mathbf{z}$ via greedy decoding. This is simply because the order of probabilities for corresponding vocabulary words will not change, only the probability disribution p_j will get more peaked. So, we define *peaked softmax* operator with hyperparameter α as

$$p_j^{(i)}(\alpha) = \frac{\exp(l_j^{(i)}/\alpha)}{\sum_i \exp(l_j^{(i)}/\alpha)} \tag{16}$$

By Corollary 1, $|\delta_{i,j}| \to 0$ as $\alpha \to 0$ for CALCS measure computed with $p_j(\alpha)$. One can further attempt to use Corollary 1 as a guide to pinpoint a range of α values to force the approximation error within certain desired limits. We will use α as a hyperparameter in this work.

Corollary 1 is also useful for alternative ways of controlling the tightness of approximation such as incurring penalty for high-entropy output probability distributions or simply penalizing the maximum output probability values less than a desired threshold (that explicitly controls the tightness of the approximation). We leave such options of controlling the approximation error for future work.

With the guidance of Corollary 1 and *peaked softmax* in Eq. 16, we conclude that CALCS establishes a promising approximation for LCS measure. In the next section, we introduce a new objective function using CALCS as a continuously differentiable reward to be directly maximized.

2.3 Sequence Level Optimization via CALCS

In this section, we describe how to leverage CALCS to define a loss function for sequence level optimization. For notational consistency, we will use $f(\mathbf{x}, \mathbf{\Theta})$ to denote an encoder-decoder architecture that takes an input sequence $\mathbf{x}$ and outputs a sequence of tokens $\mathbf{z} = (z_1, z_2, \ldots, z_m)$ via greedy decoding from corresponding probability distributions $p_1, p_2, \ldots, p_m$ at each step.

For a pair of input sequence $\mathbf{x}$ and its corresponding ground-truth output sequence $\mathbf{y}$, we define

$$J_{\text{CALCS}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}) = -\log\left(\frac{\text{CALCS}(\mathbf{y}, f(\mathbf{x}, \mathbf{\Theta}))}{|\mathbf{y}|}\right) \tag{17}$$

as the loss function for a sample $(\mathbf{x}, \mathbf{y})$ based on the CALCS, where $|\mathbf{y}|$ denote the length of sequence $\mathbf{y}$. It is important to note here that while computing probability distribution p_t at decoding step t, we feed model's own prediction z_{t-1} at the previous time step to fight *exposure bias*.

It is important to observe here that $J_{\text{CALCS}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta})$ is differentiable in $p_1, p_2, \ldots, p_k$ by definition and each p_i is differentiable in model parameters $\mathbf{\Theta}$. Hence, $J_{\text{CALCS}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta})$ is differentiable in model parameters $\mathbf{\Theta}$, which allows us to directly optimize the network parameters with respect to LCS metric. The bound we established on the approximation error and our proposed strategy to control it theoretically ensures the feasibility of using the introduced loss function J_{CALCS} to optimize for LCS metric.

3 Model

In this section, we first briefly revisit the pointer-generator (See et al., 2017) and transformer (Vaswani et al., 2017) networks that are used as the underlying baselines in our experiments. Subsequently, we describe how the proposed objective function and its variants are used to train new summarization and machine translation models.

3.1 Baseline Models

Pointer-Generator Network. We use pointer-generator network (See et al., 2017) as our baseline sequence-to-sequence model for text summarization. It is essentially a hybrid between sequence-to-sequence model with attention (Bahdanau et al., 2015) and a pointer network (Vinyals et al., 2015) that supports two decoding modes, copying and generating, via a soft switch mechanism. This enables the model to copy a word from the input sequence based on the attention distribution. On each decoding time step t, the decoder LSTM is fed the word embedding of the previous word, and computes a decoder state s_t, an attention distribution a^t over the words of input article, and a probability $P_{\text{vocab}}(w)$ of generating word w for summary from output vocabulary V, which is then softly combined with the *copy* mode's probability distribution $P_{\text{copy}}(w)$ via soft switch probability $p_{\text{gen}} \in [0, 1]$ by

$$p_t^{(w)} = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) P_{\text{copy}}(w)$$

and

$$P_{\text{copy}}(w) = \sum_{\{i : w_i = w\}} a_i^t$$

where a_i^t indicates the attention probability on i-th word of the input article. The whole network is then trained end-to-end with the negative log-likelihood loss function of

$$J_{\text{PG}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}) = -\frac{1}{|y|} \sum_{t=1}^{|y|} \log(p_t^{(y_t)})$$

for a sample article-summary pair $(\mathbf{x}, \mathbf{y})$ where $\mathbf{\Theta}$ denote the learnable model parameters. It is important to note here that we do not use the coverage mechanism introduced by the original work (See et al., 2017) to prevent the potential repetition problem in the summaries generated by the model.

Transformer Network. For machine translation, we use the transformer network (Vaswani et al., 2017), which is a recently published model that achieved state-of-the-art results on *WMT 2014 English-to-German* MT task with less computational time owing to its highly parallelizable architecture. The core idea behind this model is to use stacked self-attention mechanisms along with point-wise, fully connected layers for both encoder and decoder to represent its input and output. For the sake of brevity, we refer the reader to (Vaswani et al., 2017) for further details regarding the architecture. Similar to previously defined loss functions, let $J_{\text{TF}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta})$ denote the per-example loss function of transformer networks for an input-output translation pair $(\mathbf{x}, \mathbf{y})$ where $\mathbf{\Theta}$ is again indicating the learnable model parameters.

3.2 Model Variants and Training

Let $\{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\}_{l=1}^{N}$ denote the set of training examples, where $\mathbf{x}^{(l)}$'s are input sequences, and $\mathbf{y}^{(l)}$'s are their corresponding ground-truth output sequences. Before optimizing for the introduced objective J_{CALCS}, we first train the corresponding baseline network by minimizing

$$J_{\{\text{PG,TF}\}}(\mathbf{\Theta}) = \frac{1}{N} \sum_{l=1}^{N} J_{\{\text{PG,TF}\}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}).$$

Unlike J_{CALCS}, loss functions $J_{\{\text{PG,TF}\}}$ for baseline models are computed by teacher forcing, feeding the previous ground-truth word at each decoding step. We will denote the baseline models by POINTGEN for pointer-generator network and TRANSFORMER for transformer network.

To optimize for the proposed objective J_{CALCS}, we initialize the model parameters $\mathbf{\Theta}$ from the pre-trained baseline network and continue training the model by minimizing the joint loss

$$J(\mathbf{\Theta}) = \lambda J_{\text{CALCS}}(\mathbf{\Theta}) + (1 - \lambda) J_{\{\text{PG,TF}\}}(\mathbf{\Theta}) \tag{18}$$

$$J_{\text{CALCS}}(\mathbf{\Theta}) = \frac{1}{N} \sum_{l=1}^{N} J_{\text{CALCS}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}) \tag{19}$$

where λ is a hyperparameter controlling the balance between the two losses. During the training with the joint loss, we compute $J_{\text{CALCS}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta})$, defined in Eq. 17, by performing $|y|$-many decoding steps as a simple strategy to prevent the model from gaming the training objec-

Model	ROUGE-1	ROUGE-L
(Nallapati et al., 2016)	35.46	32.65
w/o coverage (See et al., 2017)	36.44	33.42
w/ coverage (See et al., 2017)	39.53	36.38
LEAD-3 baseline (See et al., 2017)	**40.34**	36.57
RL (Paulus et al., 2018)	**41.16**	39.08
ML + RL (Paulus et al., 2018)	39.87	36.90
Our Models		
POINTGEN*	39.11	26.97**
POINTGEN*+SS	39.33	26.94**
POINTGEN*+SS+CALCS	**40.37**	**29.18****

Table 1: ROUGE F_1 results on *CNN/Daily Mail* summarization dataset. Our reimplementation of POINTGEN* corresponds to w/o coverage (See et al., 2017). ** sign near ROUGE-L results reported for our models indicates a difference in our ROUGE-L evaluation as explained below.

tive by generating longer and longer hypotheses instead of incurring an additional length penalty. We will refer to the resulting model trained with the loss function in Eq. 18 as $\{$POINTGEN, TRANSFORMER$\}$+CALCS depending on the baseline model.

4 Experiments

We numerically evaluate the proposed method on two sequence generation benchmarks: abstractive document-summarization and machine translation. We compare the results of the proposed method against the recently proposed strong baseline models (See et al., 2017) for summarization and and (Vaswani et al., 2017) for machine translation tasks.

4.1 Abstractive Summarization

We use a modified version of the CNN/Daily Mail dataset (Hermann et al., 2015) that is first used for summarization by (Nallapati et al., 2016). However, we follow the processing script provided by (See et al., 2017) to obtain *non-anonymized* version of the data that contains 287,226 training pairs, 13,368 validation pairs, and 11,490 test pairs of news articles (781 tokens on average) and their corresponding ground-truth summaries (56 tokens on average). We refer the reader to (See et al., 2017) for further details of the difference of their version from (Nallapati et al., 2016).

For training our baseline model, we use single layer LSTM encoder (bi-directional) and decoder with hidden dimensions of 512 and 1024, respectively. We use a vocabulary of 50k words for both source and target. Following the original paper, we also do not pre-train word embeddings, which are learned with the rest of model parameters during training. We use the Adam (Kingma and Ba, 2015) optimizer with a learning rate of 0.00001 for training. We pre-train the baseline model for 20k steps by applying greedy scheduled sampling (Bengio et al., 2015) with fixed ground-truth feeding probability of 75%. Once the baseline model training is complete, we start optimizing for CALCS objective as described in the previous section. Also, we set $\lambda = 1.0$ and $\alpha = 1.0$, which are tuned on the development set.

In Table 1, we report our main results on the summarization task. POINTGEN+SS refers to the baseline model trained with scheduled sampling. POINTGEN+SS+CALCS corresponds our model trained with CALCS starting from POINTGEN+SS model. Experimental results demonstrate that training with our proposed objective provides an improvement of 2.2 points in ROUGE-L score. This also provides empirical evidence to justify that our approximate CALCS effectively captures what the original LCS metric is supposed to measure, recalling ROUGE-L is a normalized LCS. The reason why ROUGE-L scores of our models are lower than previously reported is that we evaluate ROUGE-L score by taking the entire summary as a single sequence instead of splitting it into sentences, which is also the way we compute CALCS objective during the model training process. The main motivation behind this approach is to encourage the model to preserve the sentence order within a summary, and evaluate its performance in the same way. We consider the capability of preserving the order across produced sentences as an important attribute a multi-sentence summarization model should have in terms of readability and fluency of its generated summaries as a whole. When POINTGEN*+SS and POINTGEN*+SS+CALCS are evaluated by splitting the generated summaries into sentences, their corresponding ROUGE-L scores become 35.38 and 35.12, respectively. We also observe a nice side-improvement of 1.0 point in ROUGE-1 score over the baseline, which achieves a comparable performance with the long-overdue LEAD-3 baseline score. It might also be comparable to the recently reported state-of-the-art ROUGE-1 result on CNN/DailyMail dataset by Paulus et al. (2018) as they used a different dataset processing pipeline, which makes it difficult to directly compare with ours.

Model	BLEU
GNMT (Wu et al., 2016)	24.61
GNMT+RL (Wu et al., 2016)	24.60
TRANSFORMER (Vaswani et al., 2017)	27.3
WEIGHTED TRANSFORMER (Ahmed et al., 2018)	28.4
TRANSFORMER*	27.6
TRANSFORMER*+CALCS	**27.8**

Table 2: Machine translation results on *WMT 2014 English-to-German* task. TRANSFORMER* corresponds to our training of the original model in (Vaswani et al., 2017).

4.2 Machine Translation

We also evaluate our sequence-level training approach on the *WMT 2014 English-to-German* machine translation task, which contains 4.5M pairs of sentences.

To train our baseline transformer model, we closely follow the small model in the original transformer paper (Vaswani et al., 2017). We use a vocabulary of size 32k. Our encoder and decoder consist of $N = 6$ identical layers each. Following the notation in the original paper, we set the other parameters as $d_{\text{model}} = 512$, $d_{\text{ff}} = 2048$, $h = 8$, $P_{\text{drop}} = 0.1$. We set $\lambda = 0.3$ and $\alpha = 1.0$, which are tuned on the development set.

In Table 2, we show our empirical results on machine translation task. Our first observation is that our trained baseline transformer network achieves a better performance than the one reported in the original paper (Vaswani et al., 2017) by 0.3 BLEU score, which might be solely due to hyperparameter tuning. More importantly, we observe that training with our proposed CALCS objective leads to noticeable 0.2 BLEU point improvements over the baseline, which further reinforces our confidence in effectiveness of our proposed sequence-level training approach and its applicability to other sequence prediction tasks. It is also interesting to note that optimizing for LCS metric via its continuous approximation leads to improvements in evaluation with another discrete metric BLEU. On the other had, optimizing for the exact discrete metric BLEU via reinforcement learning strategy may not improve the evaluation performance in BLEU as reported by (Wu et al., 2016). As a final remark, we would like to note that our proposed approach is orthogonal to advancements in more expressive and powerful architecture designs. Hence it has the potential to provide further improvements over the recently proposed models such as WEIGHTED TRANSFORMER (Ahmed et al., 2018).

5 Related Work

Text Summarization. Before the successful application of neural generative models, most of the existing works on text summarization (Dorr et al., 2003; Durrett et al., 2016) have focused on extractive methods. While some of the early approaches have used a rich set of heuristic rules or sparse features to select textual units to include in the summary, more recent works (Cheng and Lapata, 2016; Nallapati et al., 2017) leverage neural models to select words and sentences from the original text. With the emergence of sequence-to-sequence models (Sutskever et al., 2014) and large-scale datasets like *CNN/Daily Mail* (Hermann et al., 2015; Nallapati et al., 2016) and *NYT* (Paulus et al., 2018), abstractive summarization of longer text have become a more feasible and popular task. Several recent approaches have been proposed to tackle abstractive summarization problem, where Nallapati et al. (2016) exploits hierarchical encoders, See et al. (2017) proposes *pointer-generator* network and coverage mechanism to overcome OOV and repetition problems, Tan et al. (2017) introduces a graph-based attention mechanism and hierarchical beam search strategy, and (Paulus et al., 2018) proposes to optimize for ROUGE metric via reinforcement learning. Although impressive progress has been achieved for sentence-level summarization, attempts on abstractive document summarization task are still in early stages where the simple LEAD-3 baseline performance is only very recently matched (Paulus et al., 2018).

Neural Machine Translation. With the recent success of encoder-decoder architectures (Sutskever et al., 2014; Bahdanau et al., 2015), neural machine translation systems has gained a a lot of attention both from academia (Cho et al., 2014; Luong et al., 2015; Luong and Manning, 2016) and industry (Wu et al., 2016; Vaswani et al., 2017; Ahmed et al., 2018) over statistical machine translation, which has been the dominating translation paradigm for years. Most of these works has focused more on enhancing the architecture design aspect to tackle with various challenges such as different attention mechanisms (Bahdanau et al., 2015; Luong et al., 2015), a character-level decoder (Chung et al., 2016), a translation coverage mechanism (Tu et al., 2016), and so on. However, only very recently, a few works (Wu et al., 2016; Ranzato et al., 2016;

Norouzi et al., 2016; Shen et al., 2016; Bahdanau et al., 2017; Zhukov and Kretov, 2017; Casas et al., 2018) have investigated sequence-level optimization by training to maximize BLEU score.

Neural Sequence Generation with RL. Most neural sequence generation models are trained with the objective of maximizing the probability of the next correct word. However, this results in a major discrepancy between training and test settings of these models because they are trained with cross-entropy loss at word-level, but evaluated based on sequence-level discrete metrics such as ROUGE (Lin and Och, 2004) or BLEU (Papineni et al., 2002). On the other hand, directly optimizing for such evaluation metrics is hard due to non-differentiable nature of the exact objective (Rosti et al., 2011). Recent works (Ranzato et al., 2016; Wu et al., 2016; Bahdanau et al., 2017; Paulus et al., 2018) address the difficulty of differentiating with respect to rewards based on such discrete metrics using variants of reinforcement learning. These methods essentially propose to mitigate the problem by optimizing the reward weighted log-likelihood of the hypothesis sequences generated by the model distribution. In this paper, we propose an alternative solution to tackle this problem by introducing a differentiable approximation to exact LCS metric that can be directly optimized by standard gradient-based methods without RL, while still addressing the *exposure bias* problem.

6 Conclusion and Future Work

In this work we explored an alternative approach for training text generation models with sequence-level optimization to combat *wrong objective* and *exposure bias* problems. We introduced a new objective function based on a continuous approximation of LCS metric that measures sequence-level structure similarity between sentences. We applied our proposed approach to *CNN/Daily Mail* dataset for long document summarization and *WMT 2014 English-to-German* machine translation task. By extending the objectives of strong neural baseline models with our proposed objective, we empirically demonstrated its effectiveness on these two tasks. Our proposed approach suggests a promising alternative to policy-gradient methods to side step the difficulty of differentiating w.r.t reward function while directly optimizing for sequence-level discrete metrics.

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2018. Weighted transformer network for machine translation. In *International Conference on Learning Representations (ICLR)*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengi. 2017. An Actor-Critic algorithm for sequence prediction. In *International Conference on Learning Representations (ICLR)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

N. Casas, M R. Costa-juss, and J.A. R. Fonollosa. 2018. A differentiable bleu loss. analysis and first results. *Workshop of the 6th International Conference on Learning Representations*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengi. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Sumit Chopra, Michael Auli, and M. Alexander Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *The North American Chapter of the Association for Computational Linguistics (NAACL)*.

J. Chung, K. Cho, , and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. 2018. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

C.Y. Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Minh-Thang Luong and Christopher D. Manning. 2016. A hybrid Word-Character approach to open vocabulary neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Yishu Miao and Phil Blunsom. 2016. Discrete generative models for sentence compression. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Computational Natural Language Learning (CoNLL)*.

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances in Neural Information Processing Systems (NIPS)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Antti-Veikko I. Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2011. Expected bleu training for graphs: Bbn system description for wmt11 system combination task. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

M. Alexander Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Z. Tu, Z. Lu, Y. Liu, X. Liu, and H Li. 2016. Coverage-based neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ashish Vaswani, Shazeer Noam, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems (NIPS)*.

R. J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Norouzim Mohammad, Wolfgang Macherey, Maxim Krikun, Cao Yuan, Qin Gao, and et al. Macherey, Klaus. 2016. Googles neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Qingyu Zhou, Na Yang, Fur Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Vlad Zhukov and Maksim Kretov. 2017. Differentiable lower bound for expected BLEU score. In *NIPS Workshop on Conversational AI*.

A Proof of Lemma 1

[Monotonicity] The following two inequalities

$$s_{i,j} \leq s_{i,j+1} \leq s_{i,j} + 1$$
$$s_{i,j} \leq s_{i+1,j} \leq s_{i,j} + 1$$

hold for all $0 \leq i < m$ and $0 \leq j < k$.

Proof. We will prove this lemma by induction on $i + j$.

Base Case: $i + j = 0$. In this case, we have $i = j = 0$. Since $s_{0,0} = s_{1,0} = s_{0,1} = 0$ by definition, both $s_{0,0} \leq s_{0,1} \leq s_{0,0} + 1$ and $s_{0,0} \leq s_{1,0} \leq s_{0,0} + 1$ hold.

Inductive Step: Assume for $i + j = l$ that

$$s_{i,j} \leq s_{i,j+1} \leq s_{i,j} + 1 \tag{20}$$
$$s_{i,j} \leq s_{i+1,j} \leq s_{i,j} + 1 \tag{21}$$

hold.

We will now prove that the inequalities of inductive hypothesis hold for $i + j = l + 1$.

We will start by showing $s_{i,j+1} \geq s_{i,j}$. By definition, we have

$$s_{i,j+1} = p^{(y_i)}_{j+1}(s_{i-1,j} + 1) + \tag{22}$$
$$(1 - p^{(y_i)}_{j+1}) \max(s_{i-1,j+1}, s_{i,j}) \tag{23}$$
$$\geq p^{(y_i)}_{j+1}(s_{i-1,j} + 1) + (1 - p^{(y_i)}_{j+1}) s_{i,j} \tag{24}$$
$$\geq p^{(y_i)}_{j+1} s_{i,j} + (1 - p_{j+1,y_i}) s_{i,j} \tag{25}$$
$$\geq s_{i,j} \tag{26}$$

where inequality 24 follows from the definition of max operator, and inequality 25 follows from induction assumption 20 because $(i - 1) + j = l$. Hence, final inequality 26 establishes the proof of $s_{i,j+1} \geq s_{i,j}$.

Now, we will show that $s_{i,j+1} \leq s_{i,j} + 1$ holds. Again by definition, we have

$$s_{i,j+1} = p^{(y_i)}_{j+1}(s_{i-1,j} + 1) + \tag{27}$$
$$(1 - p^{(y_i)}_{j+1}) \max(s_{i-1,j+1}, s_{i,j}) \tag{28}$$
$$\leq p^{(y_i)}_{j+1}(s_{i-1,j} + 1) + \tag{29}$$
$$(1 - p^{(y_i)}_{j+1})(s_{i-1,j} + 1) \tag{30}$$
$$\leq s_{i-1,j} + 1 \tag{31}$$
$$\leq s_{i,j} + 1 \tag{32}$$

where inequalities 30 and 32 follow from inequalities 20 and 21 of inductive step as $(i - 1) + j = l$.

Note that 26 and 32 completes the proof of $s_{i,j} \leq s_{i,j+1} \leq s_{i,j} + 1$ for $i + j = l + 1$. Following similar arguments, one can easily establish the correctness of $s_{i,j} \leq s_{i+1,j} \leq s_{i,j} + 1$ for $i + j = l + 1$, which completes the proof of Lemma by induction. ☐

B Definition of Rouge-L

Definition 2. *ROUGE-L is a discrete similarity metric that takes into account sentence level structure similarity by identifying longest co-occurring in-sequence n-grams automatically via longest common subsequence measure. Formally, given two sequences* $\mathbf{y}$ *and* $\mathbf{z}$ *of tokens, we define ROUGE-L*$(\mathbf{y}, \mathbf{z})$ *as the harmonic mean of precision* $\frac{LCS(\mathbf{y},\mathbf{z})}{k}$ *and recall* $\frac{LCS(\mathbf{y},\mathbf{z})}{m}$ *based on LCS measure, where* $k = |z|$ *and* $m = |y|$.

Pathologies of Neural Models Make Interpretations Difficult

Shi Feng[1] **Eric Wallace**[1] **Alvin Grissom II**[2] **Mohit Iyyer**[3,4]
Pedro Rodriguez[1] **Jordan Boyd-Graber**[1]
[1]University of Maryland [2]Ursinus College
[3]UMass Amherst [4]Allen Institute for Artificial Intelligence
`{shifeng,ewallac2,entilzha,jbg}@umiacs.umd.edu,`
`agrissom@ursinus.edu,miyyer@cs.umass.edu`

Abstract

One way to interpret neural model predictions is to highlight the most important input features—for example, a heatmap visualization over the words in an input sentence. In existing interpretation methods for NLP, a word's importance is determined by either input perturbation—measuring the decrease in model confidence when that word is removed—or by the gradient with respect to that word. To understand the limitations of these methods, we use input reduction, which iteratively removes the least important word from the input. This exposes pathological behaviors of neural models: the remaining words appear nonsensical to humans and are not the ones determined as important by interpretation methods. As we confirm with human experiments, the reduced examples lack information to support the prediction of any label, but models still make the same predictions with high confidence. To explain these counterintuitive results, we draw connections to adversarial examples and confidence calibration: pathological behaviors reveal difficulties in interpreting neural models trained with maximum likelihood. To mitigate their deficiencies, we fine-tune the models by encouraging high entropy outputs on reduced examples. Fine-tuned models become more interpretable under input reduction without accuracy loss on regular examples.

1 Introduction

Many interpretation methods for neural networks explain the model's prediction as a counterfactual: how does the prediction change when the input is modified? Adversarial examples (Szegedy et al., 2014; Goodfellow et al., 2015) highlight the instability of neural network predictions by showing how small perturbations to the input dramatically change the output.

SQuAD	
Context	In 1899, John Jacob Astor IV invested \$100,000 for Tesla to further develop and produce a new lighting system. Instead, Tesla used the money to fund his Colorado Springs experiments.
Original	What did Tesla spend Astor's money on ?
Reduced	did
Confidence	0.78 → 0.91

Figure 1: SQuAD example from the validation set. Given the original *Context*, the model makes the same correct prediction ("Colorado Springs experiments") on the *Reduced* question as the *Original*, with even higher confidence. For humans, the reduced question, "did", is nonsensical.

A common, non-adversarial form of model interpretation is feature attribution: features that are crucial for predictions are highlighted in a heatmap. One can measure a feature's importance by input perturbation. Given an input for text classification, a word's importance can be measured by the difference in model confidence before and after that word is removed from the input—the word is important if confidence decreases significantly. This is the leave-one-out method (Li et al., 2016b). Gradients can also measure feature importance; for example, a feature is influential to the prediction if its gradient is a large positive value. Both perturbation and gradient-based methods can generate heatmaps, implying that the model's prediction is highly influenced by the highlighted, important words.

Instead, we study how the model's prediction is influenced by the *unimportant* words. We use **input reduction**, a process that iteratively removes the unimportant words from the input while maintaining the model's prediction. Intuitively, the words remaining after input reduction should be important for prediction. Moreover, the words

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3719–3728
Brussels, Belgium, October 31 - November 4, 2018.

should match the leave-one-out method's selections, which closely align with human perception (Li et al., 2016b; Murdoch et al., 2018). However, rather than providing explanations of the original prediction, our reduced examples more closely resemble adversarial examples. In Figure 1, the reduced input is meaningless to a human but retains the same model prediction with high confidence. Gradient-based input reduction exposes pathological model behaviors that contradict what one expects based on existing interpretation methods.

In Section 2, we construct more of these counterintuitive examples by augmenting input reduction with beam search and experiment with three tasks: SQUAD (Rajpurkar et al., 2016) for reading comprehension, SNLI (Bowman et al., 2015) for textual entailment, and VQA (Antol et al., 2015) for visual question answering. Input reduction with beam search consistently reduces the input sentence to very short lengths—often only one or two words—without lowering model confidence on its original prediction. The reduced examples appear nonsensical to humans, which we verify with crowdsourced experiments. In Section 3, we draw connections to adversarial examples and confidence calibration; we explain why the observed pathologies are a consequence of the overconfidence of neural models. This elucidates limitations of interpretation methods that rely on model confidence. In Section 4, we encourage high model uncertainty on reduced examples with entropy regularization. The pathological model behavior under input reduction is mitigated, leading to more reasonable reduced examples.

2 Input Reduction

To explain model predictions using a set of important words, we must first define importance. After defining input perturbation and gradient-based approximation, we describe input reduction with these importance metrics. Input reduction drastically shortens inputs without causing the model to change its prediction or significantly decrease its confidence. Crowdsourced experiments confirm that reduced examples appear nonsensical to humans: input reduction uncovers pathological model behaviors.

2.1 Importance from Input Gradient

Ribeiro et al. (2016) and Li et al. (2016b) define importance by seeing how confidence changes when a feature is removed; a natural approximation is to use the gradient (Baehrens et al., 2010; Simonyan et al., 2014). We formally define these importance metrics in natural language contexts and introduce the efficient gradient-based approximation. For each word in an input sentence, we measure its importance by the change in the confidence of the original prediction when we remove that word from the sentence. We switch the sign so that when the confidence decreases, the importance value is positive.

Formally, let $\mathbf{x} = \langle x_1, x_2, \dots x_n \rangle$ denote the input sentence, $f(y \mid \mathbf{x})$ the predicted probability of label y, and $y = \operatorname{argmax}_{y'} f(y' \mid \mathbf{x})$ the original predicted label. The importance is then

$$g(x_i \mid \mathbf{x}) = f(y \mid \mathbf{x}) - f(y \mid \mathbf{x}_{-i}). \qquad (1)$$

To calculate the importance of each word in a sentence with n words, we need n forward passes of the model, each time with one of the words left out. This is highly inefficient, especially for longer sentences. Instead, we approximate the importance value with the input gradient. For each word in the sentence, we calculate the dot product of its word embedding and the gradient of the output with respect to the embedding. The importance of n words can thus be computed with a single forward-backward pass. This gradient approximation has been used for various interpretation methods for natural language classification models (Li et al., 2016a; Arras et al., 2016); see Ebrahimi et al. (2017) for further details on the derivation. We use this approximation in all our experiments as it selects the same words for removal as an exhaustive search (no approximation).

2.2 Removing Unimportant Words

Instead of looking at the words with high importance values—what interpretation methods commonly do—we take a complementary approach and study how the model behaves when the supposedly unimportant words are removed. Intuitively, the important words should remain after the unimportant ones are removed.

Our input reduction process iteratively removes the unimportant words. At each step, we remove the word with the lowest importance value until the model changes its prediction. We experiment with three popular datasets: SQuAD (Rajpurkar et al., 2016) for reading comprehension, SNLI (Bowman et al., 2015) for textual entailment, and VQA (Antol et al., 2015) for visual question answering. We describe each of these tasks and the model we use below, providing full details in the Supplement.

In SQuAD, each example is a context paragraph and a question. The task is to predict a span in the paragraph as the answer. We reduce only the question while keeping the context paragraph unchanged. The model we use is the DrQA Document Reader (Chen et al., 2017).

In SNLI, each example consists of two sentences: a premise and a hypothesis. The task is to predict one of three relationships: entailment, neutral, or contradiction. We reduce only the hypothesis while keeping the premise unchanged. The model we use is Bilateral Multi-Perspective Matching (BiMPM) (Wang et al., 2017).

In VQA, each example consists of an image and a natural language question. We reduce only the question while keeping the image unchanged. The model we use is Show, Ask, Attend, and Answer (Kazemi and Elqursh, 2017).

During the iterative reduction process, we ensure that the prediction does not change (exact same span for SQuAD); consequently, the model accuracy on the reduced examples is identical to the original. The predicted label is used for input reduction and the ground-truth is never revealed. We use the validation set for all three tasks.

Most reduced inputs are nonsensical to humans (Figure 2) as they lack information for *any* reasonable human prediction. However, models make confident predictions, at times even more confident than the original.

To find the shortest possible reduced inputs (potentially the most meaningless), we relax the requirement of removing only the least important word and augment input reduction with beam search. We limit the removal to the k least important words, where k is the beam size, and decrease the beam size as the remaining input is shortened.[1] We empirically select beam size five as it produces comparable results to larger beam sizes with reasonable computation cost. The requirement of maintaining model prediction is unchanged.

SNLI

Premise	Well dressed man and woman dancing in the street
Original	Two man is dancing on the street
Reduced	dancing
Answer	Contradiction
Confidence	0.977 → 0.706

VQA

Original	What color is the flower ?
Reduced	flower ?
Answer	yellow
Confidence	0.827 → 0.819

Figure 2: Examples of original and reduced inputs where the models predict the same *Answer*. *Reduced* shows the input after reduction. We remove words from the hypothesis for SNLI, questions for SQuAD and VQA. Given the nonsensical reduced inputs, humans would not be able to provide the answer with high confidence, yet, the neural models do.

With beam search, input reduction finds extremely short reduced examples with little to no decrease in the model's confidence on its original predictions. Figure 3 compares the length of input sentences before and after the reduction. For all three tasks, we can often reduce the sentence to only one word. Figure 4 compares the model's confidence on original and reduced inputs. On SQuAD and SNLI the confidence decreases slightly, and on VQA the confidence even increases.

2.3 Humans Confused by Reduced Inputs

On the reduced examples, the models retain their original predictions despite short input lengths. The following experiments examine whether these predictions are justified or pathological, based on how humans react to the reduced inputs.

For each task, we sample 200 examples that are correctly classified by the model and generate their reduced examples. In the first setting, we compare the human accuracy on original and reduced examples. We recruit two groups of crowd workers and task them with textual entailment, reading comprehension, or visual question answering. We show one group the original inputs and the other the reduced. Humans are no longer able to give

---

[1] We set beam size to $\max(1, \min(k, L-3))$ where k is maximum beam size and L is the current length of the input sentence.

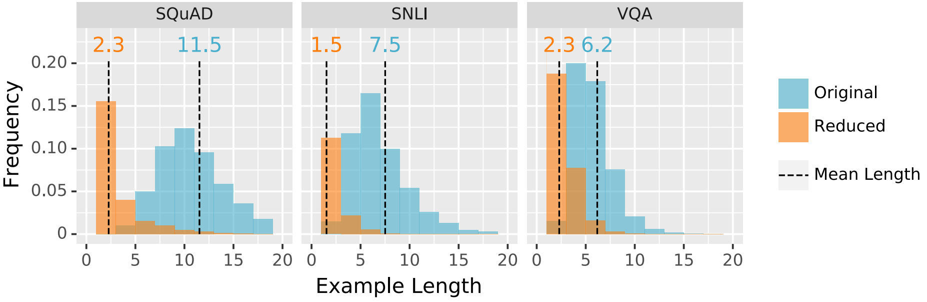

Figure 3: Distribution of input sentence length before and after reduction. For all three tasks, the input is often reduced to one or two words without changing the model's prediction.

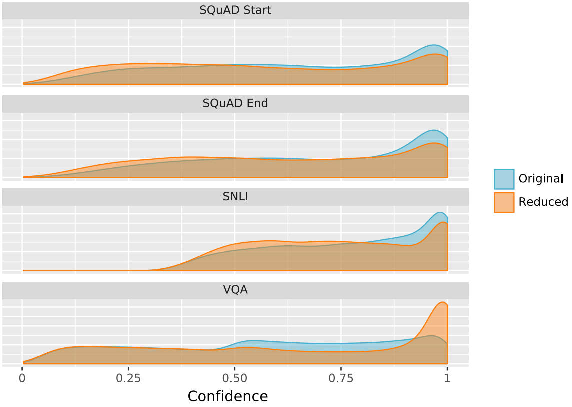

Figure 4: Density distribution of model confidence on reduced inputs is similar to the original confidence. In SQUAD, we predict the beginning and the end of the answer span, so we show the confidence for both.

the correct answer, showing a significant accuracy loss on all three tasks (compare *Original* and *Reduced* in Table 1).

The second setting examines how random the reduced examples appear to humans. For each of the original examples, we generate a version where words are randomly removed until the length matches the one generated by input reduction. We present the original example along with the two reduced examples and ask crowd workers their preference between the two reduced ones. The workers' choice is almost fifty-fifty (the *vs. Random* in Table 1): the reduced examples appear almost random to humans.

These results leave us with two puzzles: why are the models highly confident on the nonsensical reduced examples? And why, when the leave-one-out method selects important words that appear reasonable to humans, the input reduction process selects ones that are nonsensical?

Dataset	Original	Reduced	vs. Random
SQUAD	80.58	31.72	53.70
SNLI-E	76.40	27.66	42.31
SNLI-N	55.40	52.66	50.64
SNLI-C	76.20	60.60	49.87
VQA	76.11	40.60	61.60

Table 1: Human accuracy on *Reduced* examples drops significantly compared to the *Original* examples, however, model predictions are identical. The reduced examples also appear random to humans—they do not prefer them over random inputs (*vs. Random*). For SQUAD, accuracy is reported using F1 scores, other numbers are percentages. For SNLI, we report results on the three classes separately: entailment (-*E*), neutral (-*N*), and contradiction (-*C*).

3 Making Sense of Reduced Inputs

Having established the incongruity of our definition of importance *vis-à-vis* human judgements, we now investigate possible explanations for these results. We explain why model confidence can empower methods such as leave-one-out to generate reasonable interpretations but also lead to pathologies under input reduction. We attribute these results to two issues of neural models.

3.1 Model Overconfidence

Neural models are overconfident in their predictions (Guo et al., 2017). One explanation for overconfidence is overfitting: the model overfits the negative log-likelihood loss during training by learning to output low-entropy distributions over classes. Neural models are also overconfident on examples outside the training data distribution. As Goodfellow et al. (2015) observe for image classification, samples from pure noise can sometimes trigger highly confident predictions. These so-called *rubbish examples* are degenerate inputs that a human would trivially classify as not belonging to any class but for which the model predicts with high confidence. Goodfellow et al. (2015) argue that the rubbish examples exist for the same reason that adversarial examples do: the surprising linear nature of neural models. In short, the confidence of a neural model is not a robust estimate of its prediction uncertainty.

Our reduced inputs satisfy the definition of rubbish examples: humans have a hard time making predictions based on the reduced inputs (Table 1), but models make predictions with high confidence (Figure 4). Starting from a valid example, input reduction transforms it into a rubbish example.

The nonsensical, almost random results are best explained by looking at a complete reduction path (Figure 5). In this example, the transition from valid to rubbish happens immediately after the first step: following the removal of "Broncos", humans can no longer determine which team the question is asking about, but model confidence remains high. Not being able to lower its confidence on rubbish examples—as it is not trained to do so—the model neglects "Broncos" and eventually the process generates nonsensical results.

In this example, the leave-one-out method will not highlight "Broncos". However, this is not a failure of the interpretation method but of the model itself. The model assigns a low importance to "Broncos" in the first step, causing it to be removed—leave-one-out would be able to expose this particular issue by not highlighting "Broncos". However, in cases where a similar issue only appear after a few unimportant words are removed, the leave-one-out method would fail to expose the unreasonable model behavior.

Input reduction can expose deeper issues of model overconfidence and stress test a model's uncertainty estimation and interpretability.

3.2 Second-order Sensitivity

So far, we have seen that the output of a neural model is sensitive to small changes in its input. We call this *first-order* sensitivity, because interpretation based on input gradient is a first-order Taylor expansion of the model near the input (Simonyan et al., 2014). However, the *interpretation* also shifts drastically with small input changes (Figure 6). We call this *second-order* sensitivity.

The shifting heatmap suggests a mismatch between the model's first- and second-order sensitivities. The heatmap shifts when, with respect to the removed word, the model has low first-order sensitivity but high second-order sensitivity.

SQuAD
Context: The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott. The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott.

Question:
(0.90, 0.89) Where did the Broncos practice for the Super Bowl ?
(0.92, 0.88) Where did the practice for the Super Bowl ?
(0.91, 0.88) Where did practice for the Super Bowl ?
(0.92, 0.89) Where did practice the Super Bowl ?
(0.94, 0.90) Where did practice the Super ?
(0.93, 0.90) Where did practice Super ?
(0.40, 0.50) did practice Super ?

Figure 5: A reduction path for a SQuAD validation example. The model prediction is always correct and its confidence stays high (shown on the left in parentheses) throughout the reduction. Each line shows the input at that step with an underline indicating the word to remove next. The question becomes unanswerable immediately after "Broncos" is removed in the first step. However, in the context of the original question, "Broncos" is the least important word according to the input gradient.

Similar issues complicate comparable interpretation methods for image classification models. For example, Ghorbani et al. (2017) modify image inputs so the highlighted features in the interpretation change while maintaining the same prediction. To achieve this, they iteratively modify the input to maximize changes in the distribution of feature importance. In contrast, the shifting heatmap we observe occurs by only removing the least impactful features without a targeted optimization. They also speculate that the steepest gradient direction for the first- and second-order sensitivity values are generally orthogonal. Loosely speaking, the shifting heatmap suggests that the direction of the smallest gradient value can sometimes align with very steep changes in second-order sensitivity.

When explaining individual model predictions, the heatmap suggests that the prediction is made based on a weighted combination of words, as in a linear model, which is not true unless the model is indeed taking a weighted sum such as in a DAN (Iyyer et al., 2015). When the model composes representations by a non-linear combination of words, a linear interpretation oblivious to second-order sensitivity can be misleading.

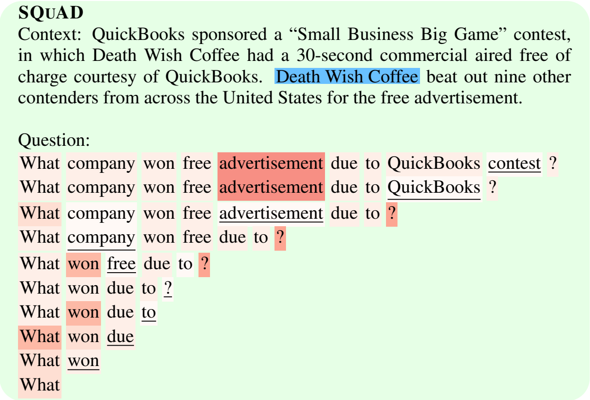

SQuAD

Context: QuickBooks sponsored a "Small Business Big Game" contest, in which Death Wish Coffee had a 30-second commercial aired free of charge courtesy of QuickBooks. Death Wish Coffee beat out nine other contenders from across the United States for the free advertisement.

Question:
What company won free advertisement due to QuickBooks contest ?
What company won free advertisement due to QuickBooks ?
What company won free advertisement due to ?
What company won free due to ?
What won free due to ?
What won due to ?
What won due to
What won due
What won
What

Figure 6: Heatmap generated with leave-one-out shifts drastically despite only removing the least important word (underlined) at each step. For instance, "advertisement", is the most important word in step two but becomes the least important in step three.

4 Mitigating Model Pathologies

The previous section explains the observed pathologies from the perspective of overconfidence: models are too certain on rubbish examples when they should not make *any* prediction. Human experiments in Section 2.3 confirm that the reduced examples fit the definition of rubbish examples. Hence, a natural way to mitigate the pathologies is to maximize model uncertainty on the reduced examples.

4.1 Regularization on Reduced Inputs

To maximize model uncertainty on reduced examples, we use the entropy of the output distribution as an objective. Given a model f trained on a dataset $(\mathcal{X}, \mathcal{Y})$, we generate reduced examples using input reduction for all training examples $\mathcal{X}$. Beam search often yields multiple reduced versions with the same minimum length for each input $\mathbf{x}$, and we collect all of these versions together to form $\tilde{\mathcal{X}}$ as the "negative" example set.

Let $\mathbb{H}(\cdot)$ denote the entropy and $f(y \mid \mathbf{x})$ denote the probability of the model predicting y given $\mathbf{x}$. We fine-tune the existing model to simultaneously maximize the log-likelihood on regular examples and the entropy on reduced examples:

$$\sum_{(\mathbf{x},y)\in(\mathcal{X},\mathcal{Y})} \log(f(y \mid \mathbf{x})) + \lambda \sum_{\tilde{\mathbf{x}}\in\tilde{\mathcal{X}}} \mathbb{H}\left(f(y \mid \tilde{\mathbf{x}})\right), \tag{2}$$

where hyperparameter λ controls the trade-off between the two terms. Similar entropy regularization is used by Pereyra et al. (2017), but not in combination with input reduction; their entropy term is calculated on regular examples rather than reduced examples.

	Accuracy		Reduced length	
	Before	After	Before	After
SQuAD	77.41	78.03	2.27	4.97
SNLI	85.71	85.72	1.50	2.20
VQA	61.61	61.54	2.30	2.87

Table 2: Model *Accuracy* on regular validation examples remains largely unchanged after fine-tuning. However, the length of the reduced examples (*Reduced length*) increases on all three tasks, making them less likely to appear nonsensical to humans.

4.2 Regularization Mitigates Pathologies

On regular examples, entropy regularization does no harm to model accuracy, with a slight increase for SQuAD (*Accuracy* in Table 2).

After entropy regularization, input reduction produces more reasonable reduced inputs (Figure 7). In the SQuAD example from Figure 1, the reduced question changed from "did" to "spend Astor money on ?" after fine-tuning. The average length of reduced examples also increases across all tasks (*Reduced length* in Table 2). To verify that model overconfidence is indeed mitigated—that the reduced examples are less "rubbish" compared to before fine-tuning—we repeat the human experiments from Section 2.3.

Human accuracy increases across all three tasks (Table 3). We also repeat the *vs. Random* experiment: we re-generate the random examples to match the lengths of the new reduced examples from input reduction, and find humans now prefer the reduced examples to random ones. The increase in both human performance and preference suggests that the reduced examples are more reasonable; model pathologies have been mitigated.

While these results are promising, it is not clear whether our input reduction method is necessary to achieve them. To provide a baseline, we fine-tune models using inputs randomly reduced to the same lengths as the ones generated by input reduction. This baseline improves neither the model accuracy on regular examples nor interpretability under input reduction (judged by lengths of reduced examples). Input reduction is effective in generating negative examples to counter model overconfidence.

SQUAD	
Context	In 1899, John Jacob Astor IV invested \$100,000 for Tesla to further develop and produce a new lighting system. Instead, Tesla used the money to fund his Colorado Springs experiments.
Original	What did Tesla spend Astor's money on ?
Answer	Colorado Springs experiments
Before	did
After	spend Astor money on ?
Confidence	0.78 → 0.91 → 0.52
SNLI	
Premise	Well dressed man and woman dancing in the street
Original	Two man is dancing on the street
Answer	Contradiction
Before	dancing
After	two man dancing
Confidence	0.977 → 0.706 → 0.717
VQA	
Original	What color is the flower ?
Answer	yellow
Before	flower ?
After	What color is flower ?
Confidence	0.847 → 0.918 → 0.745

Figure 7: SQUAD example from Figure 1, SNLI and VQA (image omitted) examples from Figure 2. We apply input reduction to models both *Before* and *After* entropy regularization. The models still predict the same *Answer*, but the reduced examples after fine-tuning appear more reasonable to humans.

	Accuracy		vs. Random	
	Before	After	Before	After
SQUAD	31.72	51.61	53.70	62.75
SNLI-E	27.66	32.37	42.31	50.62
SNLI-N	52.66	50.50	50.64	58.94
SNLI-C	60.60	63.90	49.87	56.92
VQA	40.60	51.85	61.60	61.88

Table 3: Human *Accuracy* increases after fine-tuning the models. Humans also prefer gradient-based reduced examples over randomly reduced ones, indicating that the reduced examples are more meaningful to humans after regularization.

5 Discussion

Rubbish examples have been studied in the image domain (Goodfellow et al., 2015; Nguyen et al., 2015), but to our knowledge not for NLP. Our input reduction process gradually transforms a valid input into a rubbish example. We can often determine which word's removal causes the transition to occur—for example, removing "Broncos" in Figure 5. These rubbish examples are particularly interesting, as they are also adversarial: the difference from a valid example is small, unlike image rubbish examples generated from pure noise which are far outside the training data distribution.

The robustness of NLP models has been studied extensively (Papernot et al., 2016; Jia and Liang, 2017; Iyyer et al., 2018; Ribeiro et al., 2018), and most studies define adversarial examples similar to the image domain: small perturbations to the input lead to large changes in the output. HotFlip (Ebrahimi et al., 2017) uses a gradient-based approach, similar to image adversarial examples, to flip the model prediction by perturbing a few characters or words. Our work and Belinkov and Bisk (2018) both identify cases where noisy user inputs become adversarial by accident: common misspellings break neural machine translation models; we show that incomplete user input can lead to unreasonably high model confidence.

Other failures of interpretation methods have been explored in the image domain. The sensitivity issue of gradient-based interpretation methods, similar to our shifting heatmaps, are observed by Ghorbani et al. (2017) and Kindermans et al. (2017). They show that various forms of input perturbation—from adversarial changes to simple constant shifts in the image input—cause significant changes in the interpretation. Ghorbani et al. (2017) make a similar observation about second-order sensitivity, that "the fragility of interpretation is orthogonal to fragility of the prediction".

Previous work studies biases in the annotation process that lead to datasets easier than desired or expected which eventually induce pathological models. We attribute our observed pathologies primarily to the lack of accurate uncertainty estimates in neural models trained with maximum likelihood. SNLI hypotheses contain artifacts that allow *training* a model without the premises (Gururangan et al., 2018); we apply input reduction at *test* time to the hypothesis. Similarly, VQA images are surprisingly unimportant for training a model; we reduce the question. The recent SQUAD 2.0 (Rajpurkar et al., 2018) augments the original reading comprehension task with an uncertainty modeling requirement, the goal being to make the task more realistic and challenging.

Section 3.1 explains the pathologies from the overconfidence perspective. One explanation for overconfidence is overfitting: Guo et al. (2017) show that, late in maximum likelihood training, the model learns to minimize loss by outputting low-entropy distributions without improving validation accuracy. To examine if overfitting can explain the input reduction results, we run input reduction using DRQA model checkpoints from every training epoch. Input reduction still achieves similar results on earlier checkpoints, suggesting that better convergence in maximum likelihood training cannot fix the issues by itself—we need new training objectives with uncertainty estimation in mind.

5.1 Methods for Mitigating Pathologies

We use the reduced examples generated by input reduction to regularize the model and improve its interpretability. This resembles adversarial training (Goodfellow et al., 2015), where adversarial examples are added to the training set to improve model robustness. The objectives are different: entropy regularization encourages high uncertainty on rubbish examples, while adversarial training makes the model less sensitive to adversarial perturbations.

Pereyra et al. (2017) apply entropy regularization on regular examples from the start of training to improve model generalization. A similar method is label smoothing (Szegedy et al., 2016). In comparison, we fine-tune a model with entropy regularization on the reduced examples for better uncertainty estimates and interpretations.

To mitigate overconfidence, Guo et al. (2017) propose *post-hoc* fine-tuning a model's confidence with Platt scaling. This method adjusts the softmax function's temperature parameter using a small held-out dataset to align confidence with accuracy. However, because the output is calibrated using the entire confidence distribution, not individual values, this does not reduce overconfidence on specific inputs, such as the reduced examples.

5.2 Generalizability of Findings

To highlight the erratic model predictions on short examples and provide a more intuitive demonstration, we present paired-input tasks. On these tasks, the short lengths of reduced questions and hypotheses obviously contradict the necessary number of words for a human prediction (further supported by our human studies). We also apply input reduction to single-input tasks including sentiment analysis (Maas et al., 2011) and Quizbowl (Boyd-Graber et al., 2012), achieving similar results.

Interestingly, the reduced examples transfer to other architectures. In particular, when we feed fifty reduced SNLI inputs from each class—generated with the BIMPM model (Wang et al., 2017)—through the Decomposable Attention Model (Parikh et al., 2016),[2] the same prediction is triggered 81.3% of the time.

6 Conclusion

We introduce input reduction, a process that iteratively removes unimportant words from an input while maintaining a model's prediction. Combined with gradient-based importance estimates often used for interpretations, we expose pathological behaviors of neural models. Without lowering model confidence on its original prediction, an input sentence can be reduced to the point where it appears nonsensical, often consisting of one or two words. Human accuracy degrades when shown the reduced examples instead of the original, in contrast to neural models which maintain their original predictions.

We explain these pathologies with known issues of neural models: overconfidence and sensitivity to small input changes. The nonsensical reduced examples are caused by inaccurate uncertainty estimates—the model is not able to lower its confidence on inputs that do not belong to any label. The second-order sensitivity is another issue why gradient-based interpretation methods may fail to align with human perception: a small change in the input can cause, at the same time, a minor change in the prediction but a large change in the interpretation. Input reduction perturbs the input multiple times and can expose deeper issues of model overconfidence and oversensitivity that other methods cannot. Therefore, it can be used to stress test the interpretability of a model.

Finally, we fine-tune the models by maximizing entropy on reduced examples to mitigate the deficiencies. This improves interpretability without sacrificing model accuracy on regular examples.

To properly interpret neural models, it is important to understand their fundamental characteristics: the nature of their decision surfaces, robustness against adversaries, and limitations of their training objectives. We explain fundamental difficulties of interpretation due to pathologies in neural models trained with maximum likelihood. Our

---

[2] `http://demo.allennlp.org/textual-entailment`

work suggests several future directions to improve interpretability: more thorough evaluation of interpretation methods, better uncertainty and confidence estimates, and interpretation beyond bag-of-word heatmap.

Acknowledgments

Feng was supported under subcontract to Raytheon BBN Technologies by DARPA award HR0011-15-C-0113. JBG is supported by NSF Grant IIS1652666. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor. The authors would like to thank Hal Daumé III, Alexander M. Rush, Nicolas Papernot, members of the CLIP lab at the University of Maryland, and the anonymous reviewers for their feedback.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *International Conference on Computer Vision*.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in NLP. In *Workshop on Representation Learning for NLP*.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research*.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proceedings of the International Conference on Learning Representations*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jordan L. Boyd-Graber, Brianna Satinoff, He He, and Hal Daumé III. 2012. Besting the quiz master: Crowdsourcing incremental classification games. In *Proceedings of Empirical Methods in Natural Language Processing*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the Association for Computational Linguistics*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the Association for Computational Linguistics*.

Amirata Ghorbani, Abubakar Abid, and James Y. Zou. 2017. Interpretation of neural networks is fragile. *arXiv preprint arXiv: 1710.10547*.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the International Conference of Machine Learning*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke S. Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of Empirical Methods in Natural Language Processing*.

Vahid Kazemi and Ali Elqursh. 2017. Show, ask, attend, and answer: A strong baseline for visual question answering. *arXiv preprint arXiv: 1704.03162*.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. 2017. The (un)reliability of saliency methods. *arXiv preprint arXiv: 1711.00867*.

Jiwei Li, Xinlei Chen, Eduard H. Hovy, and Daniel Jurafsky. 2016a. Visualizing and understanding neural models in NLP. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2016b. Understanding neural networks through representation erasure. *arXiv preprint arXiv: 1612.08220*.

Andrew Maas, Raymond Daly, Peter Pham, Dan Huang, Andrew Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In

Proceedings of the Association for Computational Linguistics.

W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from lstms. In *Proceedings of the International Conference on Learning Representations*.

Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition*.

Nicolas Papernot, Patrick D. McDaniel, Ananthram Swami, and Richard E. Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. *IEEE Military Communications Conference*.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of Empirical Methods in Natural Language Processing*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *Proceedings of the International Conference on Learning Representations*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the Association for Computational Linguistics*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ questions for machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?": Explaining the predictions of any classifier. In *Knowledge Discovery and Data Mining*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the Association for Computational Linguistics*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of the International Conference on Learning Representations*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Computer Vision and Pattern Recognition*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *International Joint Conference on Artificial Intelligence*.

Phrase-level Self-Attention Networks for Universal Sentence Encoding

Wei Wu[†], **Houfeng Wang**[†‡], **Tianyu Liu**[†] and **Shuming Ma**[†]

[†]MOE Key Lab of Computational Linguistics, Peking University, Beijing, 100871, China
[‡]Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, 221009, China
{wu.wei,wanghf,tianyu0421,shumingma}@pku.edu.cn

Abstract

Universal sentence encoding is a hot topic in recent NLP research. Attention mechanism has been an integral part in many sentence encoding models, allowing the models to capture context dependencies regardless of the distance between elements in the sequence. Fully attention-based models have recently attracted enormous interest due to their highly parallelizable computation and significantly less training time. However, the memory consumption of their models grows quadratically with sentence length, and the syntactic information is neglected. To this end, we propose Phrase-level Self-Attention Networks (PSAN) that perform self-attention across words inside a phrase to capture context dependencies at the phrase level, and use the gated memory updating mechanism to refine each word's representation hierarchically with longer-term context dependencies captured in a larger phrase. As a result, the memory consumption can be reduced because the self-attention is performed at the phrase level instead of the sentence level. At the same time, syntactic information can be easily integrated in the model. Experiment results show that PSAN can achieve the state-of-the-art transfer performance across a plethora of NLP tasks including sentence classification, natural language inference and sentence textual similarity.

1 Introduction

Following the success of word embeddings (Bengio et al., 2003; Mikolov et al., 2013), one of NLP's next challenges has become the hunt for universal sentence encoders. The goal is to learn a general-purpose sentence encoding model on a large corpus, which can be readily transferred to other tasks. The learned sentence representations are able to generalize to unseen combination of words, which makes them highly desirable for downstream NLP tasks, especially for those with relatively small datasets.

Previous models for sentence encoding typically rely on Recurrent Neural Networks (RNNs) (Hochreiter and Schmidhuber, 1997; Chung et al., 2014) or Convolutional Neural Networks (CNNs) (Kalchbrenner et al., 2014; dos Santos and Gatti, 2014; Kim, 2014; Mou et al., 2016) to produce context-aware representation. RNNs encode a sentence by reading words in sequential order, they are capable of learning long-term dependencies but are hard to parallelize and not time-efficient. CNNs focus on local or position-invariant dependencies but do not perform well on many tasks (Shen et al., 2017).

Fully attention-based neural networks have attracted wide interest recently, because they can model both dependencies while being more parallelizable and requiring significantly less time to train. Vaswani et al. (2017) proposed the multi-head attention to project a sentence to multiple semantic subspaces, then apply self-attention in each subspace and concatenate the attention results. Shen et al. (2017) proposed the directional self-attention, they apply forward and backward masks to the alignment score matrix to encode temporal order information, and computed attention at feature level to select the features that can best describe the word's meaning in given context. Effective as their models are, the memory required to store the alignment scores of all the token pairs grows quadratically with the sentence length. Furthermore, the syntactic property that is intrinsic to natural language is not considered at all.

Language is inherently tree structured, and the meaning of a sentence comes largely from composing the meanings of subtrees (Chomsky, 1957). Previous syntactic tree-based sentence encoders (Socher et al., 2013; Tai et al., 2015) mainly rely on recursive networks. Although the composition-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3729–3738
Brussels, Belgium, October 31 - November 4, 2018.

ality can be explicitly modeled, their models need expensive recursion computation and are hard to be trained by batched gradient descent methods.

In this paper, we propose the Phrase-level Self-Attention Networks (PSAN), for RNN/CNN-free sentence encoding, it inherits all the advantages of fully attention-based models while requires much less memory consumption. In addition, syntactic information can be incorporated into the model more easily. In our model, every sentence is split into multiple phrases based on parse tree, self-attention is performed at the phrase level instead of the sentence level, thus the memory consumption reduces rapidly as the number of phrases increases. Furthermore, a gated memory component is employed to refine word representations hierarchically by incorporating longer-term context dependencies. As a result, syntactic information can be integrated into the model without expensive recursion computation. At last, multi-dimensional attention is applied on the refined word representations to obtain the final sentence representation.

Following Conneau et al. (2017), we trained our sentence encoder on the SNLI (Bowman et al., 2015) dataset, and evaluate the quality of the obtained universal sentence representations on a wide range of transfer tasks. The SNLI dataset is extremely suitable for training sentence encoders because it is the largest high-quality human-annotated dataset that involves reasoning about the semantic relationships within sentences.

The main contributions of our work can be summarized as follows:

- We propose the Phrase-level Self-Attention mechanism (PSA) for contextualization. The memory consumption can be reduced because self-attention is performed at the phrase level instead of the sentence level.
- A gated memory updating mechanism is proposed to refine each word representation hierarchically by incorporating different levels of contextual information along the parse tree.
- Our proposed PSAN model outperforms the state-of-the-art supervised sentence encoders on a wide range of transfer tasks with significantly less memory consumption.

2 Proposed Model

In this section, we introduce the Phrase-level Self-Attention Networks (PSAN) for sentence encoding. A phrase is a group of words that carry a specific idiomatic meaning and function as a constituent in the syntax of a sentence. Words in a phrase are syntactically and semantically related to each other. Therefore, it can be advantageous to learn a context-aware representation inside a phrase while filtering out information from outside the phrase using self-attention mechanism. In an attempt to better utilize the tree structure which is intrinsic to language, we propose the gated memory updating mechanism to combine different levels of context information. At last, an attention mechanism is utilized to summarize all the token representations into a fixed-length sentence vector.

2.1 Phrase Division

The phrase structure organizes words into nested constituents which can be successively divided into their parts as we move down the constituency-based parse trees. One phrase division shows only one aspect of context dependency. In order to capture different levels of context dependencies, we can split a sentence at different granularities. The number of levels T is a hyper-parameter to be tuned.

We can break down the nodes at T different layers in the parse tree to capture T levels of context dependencies[1], as illustrated in Figure 1.

2.2 Phrase-level Self-Attention

This is the core component of our model. It aims to learn a context-aware representation for each token inside a phrase. In order to filter out information that is semantically or syntactically distant, self-attention is performed at the phrase level instead of the sentence level.

Similar to directional self-attention network (DiSAN) (Shen et al., 2017), Phrase-level Self-Attention uses multi-dimensional attention to compute the alignment score for each dimension of token embedding. Therefore, it can select the features that can best describe a word's specific meaning in any given context.

Given a phrase $P \in \mathbb{R}^{l \times d}$ represented as a sequence of word embeddings $[\boldsymbol{p}_1, \ldots, \boldsymbol{p}_l]$, where l is the length of the phrase and d is the dimension of word embedding representation, we first compute the alignment score for each token pair in the

---

[1] To avoid the situation that the produced phrases are too small, a phrase will not be further divided if its length is smaller than 4.

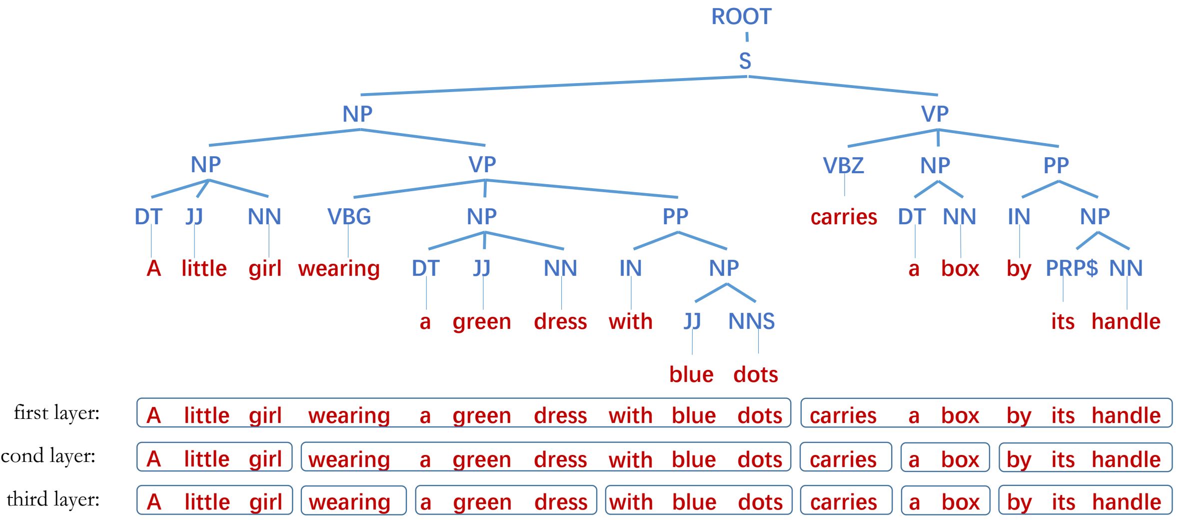

Figure 1: An example of phrase division, the sentence and its parse tree are from the SNLI training data. The division is started from the root of a parse tree. In this example, a phrase will not be further divided if it contains 3 or less words.

phrase:

$$\boldsymbol{a}_{ij} = \sigma\left(W^{a1}\boldsymbol{p}_i + W^{a2}\boldsymbol{p}_j + \boldsymbol{b}^a\right) + M_{ij}$$
$$M_{ij} = \begin{cases} 0, & i \neq j \\ -\infty, & i = j \end{cases} \tag{1}$$

where $\sigma(\cdot)$ is an activation function, $W^{a1}, W^{a2} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}^a \in \mathbb{R}^d$ are parameters to be learned, and M is a diagonal-diabled mask (Hu et al., 2017) that aims to prevent a word from being aligned with itself.

The output of the attention mechanism is a weighted sum of embeddings from all tokens for each token in the phrase:

$$\tilde{\boldsymbol{p}}_i = \sum_{j=1}^{l}\left[\frac{\exp(\boldsymbol{a}_{ij})}{\sum_{k=1}^{l}\exp(\boldsymbol{a}_{ik})} \odot \boldsymbol{p}_j\right] \tag{2}$$

where $\odot$ means point-wise product. Note that the alignment score for each token pair is a vector rather than a scalar in the multi-dimensional attention.

The final output of Phrase-level Self-Attention is obtained by comparing each input representation with its attention-weighted counterpart. We use a comparison function based on absolute difference and element-wise multiplication which was similar to Wang and Jiang (2016). This comparison function has the advantage of measuring the semantic similarity or relatedness of two sequences.

$$\boldsymbol{c}_i = \sigma\left(W^c\left[\left|\boldsymbol{p}_i - \tilde{\boldsymbol{p}}_i\right|; \boldsymbol{p}_i \odot \tilde{\boldsymbol{p}}_i\right] + \boldsymbol{b}^c\right) \tag{3}$$

where $W^c \in \mathbb{R}^{d\times 2d}$ and $\boldsymbol{b}^a \in \mathbb{R}^d$ are parameters to be learned. $\boldsymbol{c}_i$ is the representation for the i-th word in the phrase that captures local dependencies within the phrase.

At last, we put together the Phrase-level Self-Attention results for non-overlapping phrases from the same phrase division of a sentence. For the t-th phrase division we can get $C^{(t)} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{l_s}]$, the phrase-level self-attention results for the sentence from the t-th layer split, where l_s is the sentence length.

2.3 Gated Memory Updating

Above describes the Phrase-level Self-Attention (PSA) for one split of the parse tree. The parse tree can be split at different granularities. We propose a novel gated memory updating mechanism to refine each word representation hierarchically with longer-term dependencies captured in a larger granularity. Inspired by the idea of adaptive gate in highway networks (Srivastava et al., 2015), our memory mechanism add a gate to original memory networks (Weston et al., 2014; Sukhbaatar et al., 2015). This gate has the ability to determine the importance of the new input and the original memory in the memory updating.

$$\begin{aligned} C^{(t)} &= PSA\left(M^{(t-1)}\right) \\ G^{(t)} &= sigmoid\left(W^g\left[M^{(t-1)}; C^{(t)}\right] + \boldsymbol{b}^g\right) \\ M^{(t)} &= G^{(t)} \odot \sigma\left(W^m\left[M^{(t-1)}; C^{(t)}\right] + \boldsymbol{b}^m\right) \end{aligned} \tag{4}$$

where $W^g, W^m \in \mathbb{R}^{d\times 2d}$ and $\boldsymbol{b}^g, \boldsymbol{b}^m \in \mathbb{R}^d$ are parameters to be learned. Note that in order to share representation power and to reduce the number of parameters, the parameters of gated memory updating are shared among different layers.

2.4 Sentence Summarization

In this layer, self-attention mechanism is employed to summarize the refined representation of a sentence into a fixed-length vector. The self-attention mechanism can explore the dependencies among tokens within the whole sentence. As a result, global dependencies can also be incorporated in the model.

$$\begin{aligned} \boldsymbol{e}_i &= W^{e2}\sigma\left(W^{e1}\boldsymbol{m}_i^{(T)} + \boldsymbol{b}^{e1}\right) + \boldsymbol{b}^{e2} \\ \boldsymbol{v} &= \sum_{i=1}^{l}\left[\frac{\exp\left(\boldsymbol{e}_i\right)}{\sum_{j=1}^{l}\exp\left(\boldsymbol{e}_j\right)} \odot \boldsymbol{m}_i^{(T)}\right] \end{aligned} \tag{5}$$

where $W^g, W^m \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}^g, \boldsymbol{b}^m \in \mathbb{R}^d$ are parameters to be learned. After this step, the refined context-aware sentence representation is compressed into a fixed-length vector.

3 Experiments

In this section, we conduct a plethora of experiments to study the effectiveness of the PSAN model. Following Conneau et al. (2017), we train our sentence encoder using the SNLI dataset, and evaluate it across a variety of NLP tasks including sentence classification, natural language inference and sentence textual similarity.

3.1 Model Configuration

300-dimensional GloVe (Pennington et al., 2014) word embeddings (Common Crawl, uncased) are used to represent words. Following Parikh et al. (2016), out-of-vocabulary words are hashed to one of 128 random embeddings initialized by uniform distribution between (-0.05, 0.05). All the word embeddings remain fixed during training. Hidden dimension d is set to 300. All other parameters are initialized with Glorot normal initialization (Glorot and Bengio, 2010). Activation function $\sigma(\cdot)$ is ELU (Clevert et al., 2015) if not specified. Minibatch size is set to 16. The number of levels T is fixed to 3 in all of our experiments. The syntactic parse trees of SNLI are provided within the corpus. parse trees for all test corpus are produced by the Stanford PCFG Parser 3.5.2 (Klein and Manning, 2003), the same parser that produced parse trees for the SNLI dataset.

To train the model, Adadelta optimizer (Zeiler, 2012) with a learning rate of 0.75 is used on the SNLI dataset. The dropout (Srivastava et al., 2014) rate and L2 regularization weight decay factor γ are set to 0.5 and 5e-5. To test the model, the SentEval toolkit (Conneau and Kiela, 2018) is used as the evaluation pipeline for fairer comparison.

3.2 Training Setting

Natural language inference (NLI) is a fundamental task in the field of natural language processing that involves reasoning about the semantic relationship between two sentences, which makes it a suitable task to train sentence encoding models.

We conduct experiments on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). The dataset has 570k human-annotated sentence pairs, each labeled with one of the following pre-defined relationships: $Entailment$ (the premise entails the hypothesis), $Contradiction$ (they contradict each other) and $Neutral$ (they are irrelevant). Following previous work (Bowman et al., 2015; Mou et al., 2016), we remove the instances which annotators can not reach consensus on. In this way we get 549367/9842/9824 sentence pairs for train/validation/test set.

Following the siamese architecture (Bromley et al., 1993), we apply PSAN to both the premise and the hypothesis with their parameters tied. $\boldsymbol{v}^p$ and $\boldsymbol{v}^h$ are fixed-length vector representations for the premise and the hypothesis respectively. The final sentence-pair representation is formed by concatenating the original vectors with the absolute difference and element-wise multiplication between them:

$$\boldsymbol{v}^{inp} = \left[\boldsymbol{v}^p; \boldsymbol{v}^h; \left|\boldsymbol{v}^p - \boldsymbol{v}^h\right|; \boldsymbol{v}^p \odot \boldsymbol{v}^h\right] \tag{6}$$

At last, we feed the sentence-pair representation $\boldsymbol{v}^{inp}$ into a two layer feed-forward network and use a $softmax$ layer to make the classification. This is the de facto scheme for sentence encoders trained on SNLI. (Mou et al., 2016; Liu et al., 2016; Shen et al., 2017)

3.3 Evaluation Setting

To show the modeling capacity and robustness of our proposed model, we evaluate our model across a wide range of tasks that can be solved purely based on the encoded semantics. The set of tasks

dataset	size	task	output	# phrases / sent.			# words / phrase		
				$1st$	$2nd$	$3rd$	$1st$	$2nd$	$3rd$
MR	10662	sentiment	2	2.00	2.89	6.03	10.79	7.47	3.58
CR	3775	product reviews	2	1.99	3.22	6.02	10.11	6.25	3.34
MPQA	10606	opinion polarity	2	1.13	1.52	1.63	2.73	2.03	1.89
SUBJ	10000	subjectivity	2	1.98	3.29	4.51	5.61	3.40	2.48
SST2	70042	sentiment	2	1.95	3.35	5.03	5.53	3.22	2.15
SST5	11855	sentiment	5	2.00	3.53	6.10	10.08	5.71	3.31
TREC	5952	question type	6	2.00	3.73	5.59	5.03	2.98	1.99
SICK-E	9930	inference	3	1.93	3.40	4.92	5.01	2.85	1.97
SICK-R	9930	inference	[0, 5]	1.93	3.40	4.92	5.01	2.85	1.97
STS14	4500	semantic similarity	[0, 5]	1.96	3.58	5.12	5.34	2.92	2.04
MRPC	5803	paraphrase	2	1.99	3.31	4.55	5.59	3.37	2.65

Table 1: Statistics of the evaluation datasets. If the output is an integer, it represents the number of classes of the classification task. If the output is an interval, it represents the output range of the regression task. **# phrases / sent.** represents the average number of phrases per sentence for each layer of phrase division. **# words / phrase** represents the average number of words per phrase for each layer of phrase division.

was selected based on what appears to be the community consensus regarding the appropriate evaluations for universal sentence representations. To facilitate comparison, we use the same sentence evaluation tool as Conneau et al. (2017) to automate evaluation on all the tasks mentioned in this paper.

The transfer tasks used in evaluation can be concluded in the following classes: sentence classification (MR, CR, MPQA, SUBJ, SST2, SST5, TREC), natural language inference (SICK-E, SICK-R), semantic relatedness (STS14) and paraphrase detection (MRPC). Table 1 presents some statistics about the datasets [2].

3.4 Baselines

We compare our model with the following supervised sentence encoders:

- **BiLSTM-Max** (Conneau et al., 2017) is a simple but effective baseline that performs max-pooling over a bi-directional LSTM.
- **AdaSent** (Zhao et al., 2015) forms a hierarchy of representations from words to phrases and then to sentences through recursive gated local composition of adjacent segments.
- **TBCNN** (Mou et al., 2015) is a tree-based CNN model where convolution is applied over the parse tree.
- **DiSAN** (Shen et al., 2017) is composed of a directional self-attention block with temporal order encoded, and a multi-dimensional attention that compresses the sequence into a vector representation.

Model	dim	$\|\boldsymbol{\theta}\|$	SNLI	Micro	Macro
BiLSTM-Max	4096	40M	84.5	85.2	83.7
AdaSent	4096	36M	83 .4	82.0	80.9
TBCNN	300	3.5M	82.1	81.1	79.3
DiSAN	600	2.4M	85.6	84.7	83.4
PSAN	300	2.0M	**86.1**	**85.7**	**84.5**

Table 2: Performance on SNLI and transfer tasks of various sentence encoders. **dim**: the size of sentence representation. $|\boldsymbol{\theta}|$: the number of parameters. Test accuracies on SNLI, micro and macro averages of accuracies of dev set on transfer tasks are chosen as evaluation metrics.

4 Results and Analysis

4.1 Overall Performance

Experiment results of our model and four baselines are shown in Table 2. Micro and macro accuracies are two composite indicators for evaluating transfer performance of tasks whose metric is classification accuracy. Macro accuracy is the proportion of true results in the population of instances from all tasks. Micro accuracy is the arithmetic mean of dev accuracies for each task.

PSAN achieves the state-of-the-art performance

---

[2] For further information on the datasets, please refer to Conneau et al. (2017).

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	SICK-R	SICK-E	STS14
BiLSTM-max	79.9	**84.6**	92.1	89.8	83.3	88.7	**75.1/82.3**	0.885	86.3	.68/.65
AdaSent	77.0	82.0	89.9	87.2	82.3	85.6	72.6/80.0	0.855	83.1	.66/.62
TBCNN	75.4	81.6	89.1	85.9	79.4	83.7	72.0/78.6	0.839	82.1	.64/.61
DiSAN	79.7	84.1	**92.2**	89.5	82.9	88.3	75.1/81.8	0.860	85.1	.66/.64
PSAN	**80.0**	84.2	91.9	**89.9**	**83.8**	**89.1**	74.9/82.1	**0.891**	**86.9**	**.69/.67**

Table 3: Transfer test results for our model and various baselines. Classification accuracy is chosen as evaluation metric for datasets including MR, CR, SUBJ, MPQA, SST, TREC and SICK-E; Classification accuracy and F1-score are chosen for MRPC; Pearson correlation is chosen for SICK-R; Pearson and Spearman correlations are chosen for STS-14.

Model	**Acc(%)**
(1) PSA on the first layer only	84.9
(2) PSA on the second layer only	85.3
(3) PSA on the third layer only	84.6
(4) w/o PSA	85.3
(5) w/o syntactic division	85.5
(6) w/o gated memory updating	85.2
(7) w/o both	84.7
(8) Full Model	**86.1**

Table 4: Ablation studies on the SNLI dataset.

with considerably fewer parameters, outperforming a RNN-based model, a CNN-based model, a fully attention-based model and a model that utilize syntactic information. Especially when compared with previous best model *BiLSTM-Max*, PSAN can outperform their model with only 5% of their parameter numbers, demonstrating the effectiveness of our model at extracting semantically important information from a sentence.

In Table 3, we compare our model with baseline sentence encoders in each transfer task. PSAN can consistently outperform the baselines in almost every task considered. On the SICK dataset, which can be seen as an out-domain version of SNLI, our model can outperform the baselines by a large margin, demonstrating the semantic relationship learned on the SNLI can be well transfered to other domains. On the STS14 dataset, where sentence vectors can be more directly measured by the cosine distance, our model can also achieve the state-of-the-art performance, indicating that our learned sentence representations are of high quality.

4.2 Ablation Study

For thorough comparison, we implement seven extra baselines to analyze the improvements contributed by each part of our PSAN model:

- **PSA on the first/second/third layer only** only uses the Phrase-level Self-Attention at the first/second/third layer of phrase division.
- **w/o PSA** applies self-attention at the sentence level and uses the gated memory updating mechanism to refine each token representation hierarchically.
- **w/o syntactic division** divides each sentence equally into small blocks, and applies PSA within each block. The number of blocks equals the number of phrases in that layer.
- **w/o gated memory updating** concatenates the outputs of Phrase-level Self-Attention from three layers of phrase division and feeds the result to a feed-forward layer.
- **w/o both** applies self-attention at the sentence level, and uses sentence summarization to summarize the attention results into a fixed length vector.

The results are listed in Table 4. We can see that (2) performs best among (1), (2) and (3), demonstrating that the second layer split is more expressive, because the number of words per phrase in the second layer is the most suitable. It is neither too small to capture context dependencies, nor too large to filter out irrelevant noise. (8) outperforms (1), (2) and (3), showing that combining phrase-level information from different granularities can further improve performance.

We also experiment on models where the alignment matrix is calculated at the sentence level or at the syntactic-irrelevant block level. (5) performs quite well, showing that hierarchical refinement on smaller units can bring about reasonable

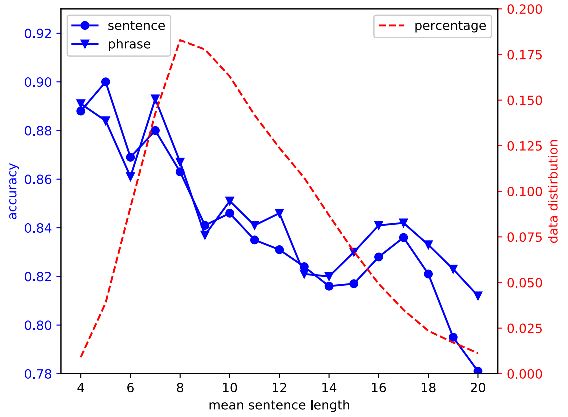

Figure 2: Fine-grained classification accuracies for PSAN and Sentence-level Self-Attention on the SNLI dataset are compared on the left, how data are distributed along sentence length is shown on the right.

performance gain. (8) outperforms (4) and (5), demonstrating syntactic information helps in sentence representation.

When comparing (6) with (8), we can tell that gated memory updating is a better method when used to refine token representation along the parse tree. We assume that memory updating resembles the tree structure of language in that larger phrase is composed in the knowledge of how smaller phrases are composed inside it.

Comparing (7) with (1), (2) and (3), we can find that performing self-attention at the phrase level is generally better than at the sentence level, indicating that reducing attention context into phrase level can effectively filter out words that are syntactically and semantically distant, thus focusing on the interaction with important words. Comparing (7) with (4), we can draw the conclusion that memory updating is effective even when the inputs to each layer are the same.

4.3 Analysis of Sentence Length

Long-term dependencies are typically hard to capture for sequential models like RNNs (Bengio et al., 1994; Hochreiter and Schmidhuber, 1997). We conduct experiments to see how performance changes as the sentence length increases. In Figure 2, we show the relationship between classification accuracy and the average length of sentence pair on the SNLI dataset. Sentence-level Self-Attention (w/o PSA model described in subsection 4.2) is used as a baseline for our model. PSAN

Model	**Memory(MB)**	**Acc(%)**
(1) Multi-head	1508	87.1
(2) DiSAN	2943	87.7
(3) PSAN	1192	**89.1**

Table 5: Memory consumption and test accuracy of three fully attention-based models on the TREC dataset.

outperforms Sentence-level Self-Attention model consistently for longer sentences of length 14 to 20. This demonstrates that incorporating syntactic information by performing self-attention at the phrase level and refining each word's representation hierarchically can help to capture long-term dependencies across words in a sentence.

4.4 Analysis of Memory Consumption

We conduct experiments to analyze the memory consumption reduction resulted from Phrase-level Self-Attention. To this end, we re-implement two fully attention-based models (Vaswani et al., 2017; Shen et al., 2017) on the TREC dataset. To make fair comparison, the dimensions of sentence vectors are set to 300, the same number as our model. Table 5 lists the results. Our PSAN model can outperform the other two fully attention-based models, while being more memory efficient. reducing more than 20% of memory consumption.

4.5 Visualization and Case Study

In order to analyze the attention changing process and the importance of each word in the sentence vector, we visualize the attention scores in the alignment matrix of each layer in Phrase-level Self-Attention and sentence summarization layer. To facilitate the visualization of the multi-dimension attention vector, we use the $l2$ norm of the attention vector for representation.

In Figure 3, we can see that, the difference in attention weights between semantically important and unimportant words gets larger as the context becomes larger. This implies that token representation can be gradually refined by the gated memory updating mechanism. Furthermore, the alignment matrix of a phrase can be refined even if the phrase division does not change between layers. For instance, the word "girl" gets larger attention weight in the second layer division than in the first layer. This demonstrates that the memory

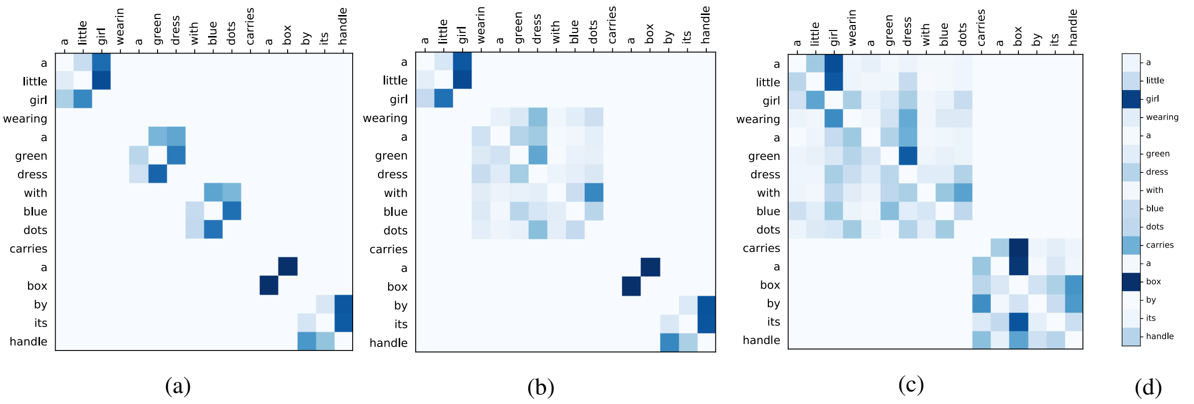

(a) (b) (c) (d)

Figure 3: (a) / (b) / (c): attention weights of Phrase-level Self-Attention mechanism in the third / second / first layer phrase division; (d): attention weights of the sentence summarization layer.

updating mechanism can gradually pick out important words for sentence representation. Finally, nouns and verbs dominate the attention weights, while stop words like “a” and “its”, contribute little to the final sentence representation, this indicates that PSAN can effectively pick out semantically important words that are most representative for the meaning of the whole sentence.

5 Related Work

Recently, self-attention mechanism has been successfully applied to the field of sentence encoding, it utilizes the attention mechanism to relate elements at different positions from a single sentence. Due to its direct access to each token representation, both long-term and local dependencies can be modeled flexibly. Liu et al. (2016) leveraged the average-pooled word representation to attend words appear in the sentence itself. Cheng et al. (2016) proposed the LSTMN model for machine reading, an attention vector is produced for each of its hidden states during the recurrent iteration, thus empowering the recurrent network with stronger memorization capability and the ability to discover relations among tokens. Lin et al. (2017) obtained a fixed-size sentence embedding matrix by introducing self-attention. Different from the feature-level attention used in our model, their attention mechanism extracted different aspects of the sentence into multiple vector representations, and utilized a penalization term to encourage the diversity of different attention results.

Syntactic information can be useful for understanding a natural language sentence. Many previous researches utilized syntactic information to build sentence encoder from composing the meanings of subtrees. Tree-LSTM (Tai et al., 2015; Zhu et al., 2015) composed its hidden state from an input vector and the hidden states of arbitrarily many child units. In Tree-based CNN (Mou et al., 2015, 2016), a set of subtree feature detectors slide over the parse tree of a sentence, and a max-pooling layer is utilized to aggregate information along different parts of the tree.

Apart from the models that use parse information, there have been several researches that aimed to learn the hierarchical latent structure of text by recursively composing words into sentence representation. Among them, neural tree indexer (Munkhdalai and Yu, 2017b) utilized LSTM or attentive node composition function to construct full n-ary tree for input text. Gumbel Tree-LSTM (Choi et al., 2018) used Straight-Through Gumbel-Softmax estimator to decide the parent node among candidates dynamically. A major drawback of these models is that the recursion computation can be expensive and hard to be processed in batches.

6 Conclusion

We propose the Phrase-level Self-Attention Networks (PSAN), a fully attention-based model that can utilize syntactic information for universal sentence encoding. By applying self-attention at the phrase level, we can filter out distant and unrelated words and focus on modeling interaction between semantically and syntactically important words, a gated memory updating mechanism is utilized to incorporate different levels of contextual information along the parse tree. Empirical results on a wide range of transfer tasks demonstrate the effectiveness of our model.

Acknowledgments

Our work is supported by National Natural Science Foundation of China under Grant No.61433015 and the National Key Research and Development Program of China under Grant No.2017YFB1002101. The corresponding authors of this paper are Houfeng Wang.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. In *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, pages 737–744.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 551–561.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structure. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, February 2-7, 2018, New Orleans, Louisiana, USA.*

Noem Chomsky. 1957. Syntactic structures. *International Journal of American Linguistics*, 149(3):174196.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 249–256.

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

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *CoRR, abs/1705.02798*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3294–3302.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Tsendsuren Munkhdalai and Hong Yu. 2017a. Neural semantic encoders. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*.

Tsendsuren Munkhdalai and Hong Yu. 2017b. Neural tree indexers for text understanding. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 41–45.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *CoRR*, abs/1709.04696.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *CoRR*, abs/1611.01747.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 4069–4076.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1604–1612.

BANDITSUM: Extractive Summarization as a Contextual Bandit

Yue Dong[*]
Mila/McGill University
yue.dong2
@mail.mcgill.ca

Yikang Shen[*]
Mila/University of Montréal
yi-kang.shen
@umontreal.ca

Eric Crawford
Mila/McGill University
eric.crawford
@mail.mcgill.ca

Herke van Hoof
University of Amsterdam
h.c.vanhoof
@uva.nl

Jackie C.K. Cheung
Mila/McGill University
jcheung
@cs.mcgill.ca

Abstract

In this work, we propose a novel method for training neural networks to perform single-document extractive summarization without heuristically-generated extractive labels. We call our approach BANDITSUM as it treats extractive summarization as a contextual bandit (CB) problem, where the model receives a document to summarize (the context), and chooses a sequence of sentences to include in the summary (the action). A policy gradient reinforcement learning algorithm is used to train the model to select sequences of sentences that maximize ROUGE score. We perform a series of experiments demonstrating that BANDITSUM is able to achieve ROUGE scores that are better than or comparable to the state-of-the-art for extractive summarization, and converges using significantly fewer update steps than competing approaches. In addition, we show empirically that BANDITSUM performs significantly better than competing approaches when good summary sentences appear late in the source document.

1 Introduction

Single-document summarization methods can be divided into two categories: extractive and abstractive. Extractive summarization systems form summaries by selecting and copying text snippets from the document, while abstractive methods aim to generate concise summaries with paraphrasing. This work is primarily concerned with extractive summarization. Though abstractive summarization methods have made strides in recent years, extractive techniques are still very attractive as they are simpler, faster, and more reliably yield semantically and grammatically correct sentences.

Many extractive summarizers work by selecting sentences from the input document (Luhn, 1958; Mihalcea and Tarau, 2004; Wong et al., 2008; Kågebäck et al., 2014; Yin and Pei, 2015; Cao et al., 2015; Yasunaga et al., 2017). Furthermore, a growing trend is to frame this sentence selection process as a sequential binary labeling problem, where binary inclusion/exclusion labels are chosen for sentences one at a time, starting from the beginning of the document, and decisions about later sentences may be conditioned on decisions about earlier sentences. Recurrent neural networks may be trained with stochastic gradient ascent to maximize the likelihood of a set of ground-truth binary label sequences (Cheng and Lapata, 2016; Nallapati et al., 2017). However, this approach has two well-recognized disadvantages. First, it suffers from exposure bias, a form of mismatch between training and testing data distributions which can hurt performance (Ranzato et al., 2015; Bahdanau et al., 2017; Paulus et al., 2018). Second, extractive labels must be generated by a heuristic, as summarization datasets do not generally include ground-truth extractive labels; the ultimate performance of models trained on such labels is thus fundamentally limited by the quality of the heuristic.

An alternative to maximum likelihood training

---

[*] Equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3739–3748
Brussels, Belgium, October 31 - November 4, 2018.

is to use reinforcement learning to train the model to directly maximize a measure of summary quality, such as the ROUGE score between the generated summary and a ground-truth abstractive summary (Wu and Hu, 2018). This approach has become popular because it avoids exposure bias, and directly optimizes a measure of summary quality. However, it also has a number of downsides. For one, the search space is quite large: for a document of length T, there are 2^T possible extractive summaries. This makes the exploration problem faced by the reinforcement learning algorithm during training very difficult. Another issue is that due to the sequential nature of selection, the model is inherently biased in favor of selecting earlier sentences over later ones, a phenomenon which we demonstrate empirically in Section 7. The first issue can be resolved to a degree using either a cumbersome maximum likelihood-based pre-training step (using heuristically-generated labels) (Wu and Hu, 2018), or placing a hard upper limit on the number of sentences selected. The second issue is more problematic, as it is inherent to the sequential binary labeling setting.

In the current work, we introduce BANDITSUM, a novel method for training neural network-based extractive summarizers with reinforcement learning. This method does away with the sequential binary labeling setting, instead formulating extractive summarization as a contextual bandit. This move greatly reduces the size of the space that must be explored, removes the need to perform supervised pre-training, and prevents systematically privileging earlier sentences over later ones. Although the strong performance of Lead-3 indicates that good sentences often occur early in the source article, we show in Sections 6 and 7 that the contextual bandit setting greatly improves model performance when good sentences occur late without sacrificing performance when good sentences occur early.

Under this reformulation, BANDITSUM takes the document as input and outputs an *affinity* for each of the sentences therein. An affinity is a real number in $[0, 1]$ which quantifies the model's propensity for including a sentence in the summary. These affinities are then used in a process of repeated sampling-without-replacement which does not privilege earlier sentences over later ones. BANDITSUM is free to process the document as a whole before yielding affinities, which permits affinities for different sentences in the document to depend on one another in arbitrary ways. In our technical section, we show how to apply policy gradient reinforcement learning methods to this setting.

The contributions of our work are as follows:

- We propose a theoretically grounded method, based on the contextual bandit formalism, for training neural network-based extractive summarizers with reinforcement learning. Based on this training method, we propose the BANDITSUM system for extractive summarization.
- We perform experiments demonstrating that BANDITSUM obtains state-of-the-art performance on a number of datasets and requires significantly fewer update steps than competing approaches.
- We perform human evaluations showing that in the eyes of human judges, summaries created by BANDITSUM are less redundant and of higher overall quality than summaries created by competing approaches.
- We provide evidence, in the form of experiments in which models are trained on subsets of the data, that the improved performance of BANDITSUM over competitors stems in part from better handling of summary-worthy sentences that come near the end of the document (see Section 7).

2 Related Work

Extractive summarization has been widely studied in the past. Recently, neural network-based methods have been gaining popularity over classical methods (Luhn, 1958; Gong and Liu, 2001; Conroy and O'leary, 2001; Mihalcea and Tarau, 2004; Wong et al., 2008), as they have demonstrated stronger performance on large corpora. Central to the neural network-based models is the encoder-decoder structure. These models typically use either a convolution neural network (Kalchbrenner et al., 2014; Kim, 2014; Yin and Pei, 2015; Cao et al., 2015), a recurrent neural network (Chung et al., 2014; Cheng and Lapata, 2016; Nallapati et al., 2017), or a combination of the two (Narayan et al., 2018; Wu and Hu, 2018) to create sentence and document representations, using word embeddings (Mikolov et al., 2013; Pennington et al.,

2014) to represent words at the input level. These vectors are then fed into a decoder network to generate the output summary.

The use of reinforcement learning (RL) in extractive summarization was first explored by Ryang and Abekawa (2012), who proposed to use the TD(λ) algorithm to learn a value function for sentence selection. Rioux et al. (2014) improved this framework by replacing the learning agent with another TD(λ) algorithm. However, the performance of their methods was limited by the use of shallow function approximators, which required performing a fresh round of reinforcement learning for every new document to be summarized. The more recent work of Paulus et al. (2018) and Wu and Hu (2018) use reinforcement learning in a sequential labeling setting to train abstractive and extractive summarizers, respectively, while Chen and Bansal (2018) combines both approaches, applying abstractive summarization to a set of sentences extracted by a pointer network (Vinyals et al., 2015) trained via REINFORCE. However, pre-training with a maximum likelihood objective is required in all of these models.

The two works most similar to ours are Yao et al. (2018) and Narayan et al. (2018). Yao et al. (2018) recently proposed an extractive summarization approach based on deep Q learning, a type of reinforcement learning. However, their approach is extremely computationally intensive (a minimum of 10 days before convergence), and was unable to achieve ROUGE scores better than the best maximum likelihood-based approach. Narayan et al. (2018) uses a cascade of filters in order to arrive at a set of candidate extractive summaries, which we can regard as an approximation of the true action space. They then use an approximation of a policy gradient method to train their neural network to select summaries from this approximated action space. In contrast, BANDITSUM samples directly from the true action space, and uses exact policy gradient parameter updates.

3 Extractive Summarization as a Contextual Bandit

Our approach formulates extractive summarization as a contextual bandit which we then train an agent to solve using policy gradient reinforcement learning. A bandit is a decision-making formalization in which an agent repeatedly chooses one of several actions, and receives a reward based on this choice. The agent's goal is to quickly learn which action yields the most favorable distribution over rewards, and choose that action as often as possible. In a *contextual* bandit, at each trial, a context is sampled and shown to the agent, after which the agent selects an action and receives a reward; importantly, the rewards yielded by the actions may depend on the sampled context. The agent must quickly learn which actions are favorable in which contexts. Contextual bandits are a subset of Markov Decision Processes in which every episode has length one.

Extractive summarization may be regarded as a contextual bandit as follows. Each document is a context, and each ordered subset of a document's sentences is a different action. Formally, assume that each context is a document d consisting of sentences $s = (s_1, \dots, s_{N_d})$, and that each action is a length-M sequence of unique sentence indices $i = (i_1, \dots, i_M)$ where $i_t \in \{1, \dots, N_d\}$, $i_t \neq i_{t'}$ for $t \neq t'$, and M is an integer hyper-parameter. For each i, the extractive summary induced by i is given by $(s_{i_1}, \dots, s_{i_M})$. An action i taken in context d is given a reward $R(i, a)$, where a is the gold-standard abstractive summary that is paired with document d, and R is a scalar reward function quantifying the degree of match between a and the summary induced by i.

A policy for extractive summarization is a neural network $p_\theta(\cdot|d)$, parameterized by a vector θ, which, for each input document d, yields a probability distribution over index sequences. Our goal is to find parameters θ which cause $p_\theta(\cdot|d)$ to assign high probability to index sequences that induce extractive summaries that a human reader would judge to be of high-quality. We achieve this by maximizing the following objective function with respect to parameters θ:

$$J(\theta) = E\left[R(i, a)\right] \tag{1}$$

where the expectation is taken over documents d paired with gold-standard abstractive summaries a, as well as over index sequences i generated according to $p_\theta(\cdot|d)$.

3.1 Policy Gradient Reinforcement Learning

Ideally, we would like to maximize (1) using gradient ascent. However, the required gradient cannot be obtained using usual techniques (e.g. simple backpropagation) because i must be discretely sampled in order to compute $R(i, a)$.

Fortunately, we can use the likelihood ratio gradient estimator from reinforcement learning and stochastic optimization (Williams, 1992; Sutton et al., 2000), which tells us that the gradient of this function can be computed as:

$$\nabla_\theta J(\theta) = E\left[\nabla_\theta \log p_\theta(i|d) R(i,a)\right] \tag{2}$$

where the expectation is taken over the same variables as (1).

Since we typically do not know the exact document distribution and thus cannot evaluate the expected value in (2), we instead estimate it by sampling. We found that we obtained the best performance when, for each update, we first sample one document/summary pair (d, a), then sample B index sequences $i^1, \ldots, i^B$ from $p_\theta(\cdot|d)$, and finally take the empirical average:

$$\nabla_\theta J(\theta) \approx \frac{1}{B}\sum_{b=1}^{B} \nabla_\theta \log p_\theta(i^b|d) R(i^b, a) \tag{3}$$

This overall learning algorithm can be regarded as an instance of the REINFORCE policy gradient algorithm (Williams, 1992).

3.2 Structure of $p_\theta(\cdot|d)$

There are many possible choices for the structure of $p_\theta(\cdot|d)$; we opt for one that avoids privileging early sentences over later ones. We first decompose $p_\theta(\cdot|d)$ into two parts: π_θ, a deterministic function which contains all the network's parameters, and μ, a probability distribution parameterized by the output of π_θ. Concretely:

$$p_\theta(\cdot|d) = \mu(\cdot|\pi_\theta(d)) \tag{4}$$

Given an input document d, π_θ outputs a real-valued vector of *sentence affinities* whose length is equal to the number of sentences in the document (i.e. $\pi_\theta(d) \in \mathbb{R}^{N_d}$) and whose elements fall in the range $[0, 1]$. The t-th entry $\pi(d)_t$ may be roughly interpreted as the network's propensity to include sentence s_t in the summary of d.

Given sentence affinities $\pi_\theta(d)$, μ implements a process of repeated sampling-without-replacement. This proceeds by repeatedly normalizing the set of affinities corresponding to sentences that have not yet been selected, thereby obtaining a probability distribution over unselected sentences, and sampling from that distribution to obtain a new sentence to include. This normalize-and-sample step is repeated M times, yielding M unique sentences to include in the summary.

At each step of sampling-without-replacement, we also include a small probability ϵ of sampling uniformly from all remaining sentences. This is used to achieve adequate exploration during training, and is similar to the ϵ-greedy technique from reinforcement learning.

Under this sampling scheme, we have the following expression for $p_\theta(i|d)$:

$$\prod_{j=1}^{M}\left(\frac{\epsilon}{N_d - j + 1} + \frac{(1-\epsilon)\pi(d)_{i_j}}{z(d) - \sum_{k=1}^{j-1}\pi(d)_{i_k}}\right) \tag{5}$$

where $z(d) = \sum_t \pi(d)_t$. For index sequences that have length different from M, or that contain duplicate indices, we have $p_\theta(i|d) = 0$. Using this expression, it is straightforward to use automatic differentiation software to compute $\nabla_\theta \log p_\theta(i|d)$, which is required for the gradient estimate in (3).

3.3 Baseline for Variance Reduction

Our sample-based gradient estimate can have high variance, which can slow the learning. One potential cause of this high variance can be seen by inspecting (3), and noting that it basically acts to change the probability of a sampled index sequence to an extent determined by the reward $R(i, a)$. However, since ROUGE scores are always positive, the probability of every sampled index sequence is increased, whereas intuitively, we would prefer to decrease the probability of sequences that receive a comparatively low reward, even if it is positive. This can be remedied by the introduction of a so-called baseline which is subtracted from all rewards.

Using a baseline $\overline{r}$, our sample-based estimate of $\nabla_\theta J(\theta)$ becomes:

$$\frac{1}{B}\sum_{i=1}^{B} \nabla_\theta \log p_\theta(i^b|d)(R(i^b, a) - \overline{r}) \tag{6}$$

It can be shown that the introduction of $\overline{r}$ does not bias the gradient estimator and can significantly reduce its variance if chosen appropriately (Sutton et al., 2000).

There are several possibilities for the baseline, including the long-term average reward and the average reward across different samples for one document-summary pair. We choose an approach known as self-critical reinforcement learning, in which the test-time performance of the current model is used as the baseline (Ranzato et al., 2015;

Rennie et al., 2017; Paulus et al., 2018). More concretely, after sampling the document-summary pair (d, a), we greedily generate an index sequence using the current parameters θ:

$$i_{greedy} = \arg\max_{i} p_\theta(i|d) \tag{7}$$

and calculate the baseline for the current update as $\bar{r} = R(i_{greedy}, a)$. This baseline has the intuitively satisfying property of only increasing the probability of a sampled label sequence when the summary it induces is better than what would be obtained by greedy decoding.

3.4 Reward Function

A final consideration is a concrete choice for the reward function $R(i, a)$. Throughout this work we use:

$$R(i, a) = \frac{1}{3}(\text{ROUGE-1}_f(i, a) + \text{ROUGE-2}_f(i, a) + \text{ROUGE-L}_f(i, a)). \tag{8}$$

The above reward function optimizes the average of all the ROUGE variants (Lin, 2004) while balancing precision and recall.

4 Model

In this section, we discuss the concrete instantiations of the neural network π_θ that we use in our experiments. We break π_θ up into two components: a document encoder $f_{\theta 1}$, which outputs a sequence of sentence feature vectors $(h_1, \ldots, h_{N_d})$ and a decoder $g_{\theta 2}$ which yields sentence affinities:

$$h_1, \ldots, h_{N_d} = f_{\theta 1}(d) \tag{9}$$

$$\pi_\theta(d) = g_{\theta 2}(h_1, \ldots, h_{N_d}) \tag{10}$$

Encoder. Features for each sentence in isolation are first obtained by applying a word-level Bidirectional Recurrent Neural Network (BiRNN) to the embeddings for the words in the sentence, and averaging the hidden states over words. A separate sentence-level BiRNN is then used to obtain a representations h_i for each sentence in the context of the document.

Decoder. A multi-layer perceptron is used to map from the representation h_t of each sentence through a final sigmoid unit to yield sentence affinities $\pi_\theta(d)$.

The use of a bidirectional recurrent network in the encoder is crucial, as it allows the network to process the document as a whole, yielding representations for each sentence that take all other sentences into account. This procedure is necessary to deal with some aspects of summary quality such as redundancy (avoiding the inclusion of multiple sentences with similar meaning), which requires the affinities for different sentences to depend on one another. For example, to avoid redundancy, if the affinity for some sentence is high, then sentences which express similar meaning should have low affinities.

5 Experiments

In this section, we discuss the setup of our experiments. We first discuss the corpora that we used and our evaluation methodology. We then discuss the baseline methods against which we compared, and conclude with a detailed overview of the settings of the model parameters.

5.1 Corpora

Three datasets are used for our experiments: the CNN, the Daily Mail, and combined CNN/Daily Mail (Hermann et al., 2015; Nallapati et al., 2016). We use the standard split of Hermann et al. (2015) for training, validating, and testing and the same setting without *anonymization* on the three corpus as See et al. (2017). The Daily Mail corpus has 196,557 training documents, 12,147 validation documents and 10,397 test documents; while the CNN corpus has 90,266/1,220/1,093 documents, respectively.

5.2 Evaluation

The models are evaluated based on ROUGE (Lin, 2004). We obtain our ROUGE scores using the standard pyrouge package[1] for the test set evaluation and a faster python implementation of the ROUGE metric[2] for training and evaluating on the validation set. We report the F1 scores of ROUGE-1, ROUGE-2, and ROUGE-L, which compute the uniform, bigram, and longest common subsequence overlapping with the reference summaries.

5.3 Baselines

We compare BANDITSUM with other extractive methods including: the Lead-3 model, SummaRuNNer (Nallapati et al., 2017), Refresh

---

[1] https://pypi.python.org/pypi/pyrouge/0.1.3

[2] We use the modified version based on https://github.com/pltrdy/rouge

(Narayan et al., 2018), RNES (Wu and Hu, 2018), DQN (Yao et al., 2018), and NN-SE (Cheng and Lapata, 2016). The Lead-3 model simply produces the leading three sentences of the document as the summary.

5.4 Model Settings

We use 100-dimensional Glove embeddings (Pennington et al., 2014) as our embedding initialization. We do not limit the sentence length, nor the maximum number of sentences per document. We use one-layer BiLSTM for word-level RNN, and two-layers BiLSTM for sentence-level RNN. The hidden state dimension is 200 for each direction on all LSTMs. For the decoder, we use a feed-forward network with one hidden layer of dimension 100.

During training, we use Adam (Kingma and Ba, 2015) as the optimizer with the learning rate of $5e^{-5}$, beta parameters $(0, 0.999)$, and a weight decay of $1e^{-6}$, to maximize the objective function defined in equation (1). We employ gradient clipping of 1 to regularize our model. At each iteration, we sample $B = 20$ times to estimate the gradient defined in equation 3. For our system, the reported performance is obtained within two epochs of training[3].

At the test time, we pick sentences sorted by the predicted probabilities until the length limit is reached. The full-length ROUGE F1 score is used as the evaluation metric. For M, the number of sentences selected per summary, we use a value of 3, based on our validation results as well as on the settings described in Nallapati et al. (2017).

6 Experiment Results

In this section, we present quantitative results from the ROUGE evaluation and qualitative results based on human evaluation. In addition, we demonstrate the stability of our RL model by comparing the validation curve of BanditSum with SummaRuNNer (Nallapati et al., 2017) trained with a maximum likelihood objective.

6.1 Rouge Evaluation

Model	ROUGE		
	1	2	L
Lead(Narayan et al., 2018)	39.6	17.7	36.2
Lead-3(ours)	40.0	17.5	36.2
SummaRuNNer	39.6	16.2	35.3
DQN	39.4	16.1	35.6
Refresh	40.0	18.2	36.6
RNES w/o coherence	41.3	**18.9**	**37.6**
BanditSum	**41.5**	18.7	**37.6**

Table 1: Performance comparison of different extractive summarization models on the combined CNN/Daily Mail test set using full-length F1.

Model	CNN			Daily Mail		
	1	2	L	1	2	L
Lead-3	28.8	11.0	25.5	41.2	18.2	37.3
NN-SE	28.4	10.0	25.0	36.2	15.2	32.9
Refresh	30.4	**11.7**	26.9	41.0	18.8	37.7
BanditSum	**30.7**	11.6	**27.4**	**42.1**	**18.9**	**38.3**

Table 2: The full-length ROUGE F1 scores of various extractive models on the CNN and the Daily Mail test set separately.

We present the results of comparing BanditSum to several baseline algorithms[4] on the CNN/Daily Mail corpus in Tables 1 and 2. Compared to other extractive summarization systems, BanditSum achieves performance that is significantly better than two RL-based approaches, Refresh (Narayan et al., 2018) and DQN (Yao et al., 2018), as well as SummaRuNNer, the state-of-the-art maximum liklihood-based extractive summarizer (Nallapati et al., 2017). BanditSum performs a little better than RNES (Wu and Hu, 2018) in terms of ROUGE-1 and slightly worse in terms of ROUGE-2. However, RNES requires pre-training with the maximum likelihood objective on heuristically-generated extractive labels; in contrast, BanditSum is very light-weight and converges significantly faster. We discuss the advantage of framing the extractive summarization based on the contextual bandit (BanditSum) over the sequential binary labeling setting (RNES) in the discussion Section 7.

We also noticed that different choices for the policy gradient baseline (see Section 3.3) in BanditSum affect learning speed, but do not significantly affect asymptotic performance. Models trained with an average reward baseline learned most quickly, while models trained with three different baselines (greedy, average reward in a

---

[3] Our code can be found at `https://github.com/yuedongP/summarization_RL`

[4] Due to different pre-processing methods and different numbers of selected sentences, several papers report different Lead scores (Narayan et al., 2018; See et al., 2017). We use the test set provided by Narayan et al. (2018). Since their Lead score is a combination of Lead-3 for CNN and Lead-4 for Daily Mail, we recompute the Lead-3 scores for both CNN and Daily Mail with the preprocessing steps used in See et al. (2017). Additionally, our results are not directly comparable to results based on the anonymized dataset used by Nallapati et al. (2017).

batch, average global reward) all perform roughly the same after training for one epoch. Models trained without a baseline were found to underperform other baseline choices by about 2 points of ROUGE score on average.

6.2 Human Evaluation

We also conduct a qualitative evaluation to understand the effects of the improvements introduced in BANDITSUM on human judgments of the generated summaries. To assess the effect of training with RL rather than maximum likelihood, in the first set of human evaluations we compare BANDITSUM with the state-of-the-art maximum likelihood-based model SummaRuNNer. To evaluate the importance of using an exact, rather than approximate, policy gradient to optimize ROUGE scores, we perform another human evaluation comparing BANDITSUM and Refresh, an RL-based method that uses the an approximation of the policy gradient.

We follow a human evaluation protocol similar to the one used in Wu and Hu (2018). Given a set of N documents, we ask K volunteers to evaluate the summaries extracted by both systems. For each document, a reference summary, and a pair of randomly ordered extractive summaries (one generated by each of the two models) is presented to the volunteers. They are asked to compare and rank the extracted summaries along three dimensions: overall, coverage, and non-redundancy.

Model	Overall	Coverage	Non-Redundancy
SummaRuNNer	1.67	**1.46**	1.70
BANDITSUM	**1.33**	1.54	**1.30**

Table 3: Average rank of human evaluation based on 5 participants who expressed 57 pairwise preferences between the summaries generated by SummaRuNNer and BANDITSUM. The model with the lower score is better.

Model	Overall	Coverage	Non-Redundancy
Refresh	1.53	**1.34**	1.55
BANDITSUM	**1.50**	1.58	**1.30**

Table 4: Average rank of manual evaluation with 4 participants who expressed 20 pairwise preferences between the summaries generated by Refresh and our system. The model with the lower score is better.

To compare with SummaRuNNer, we randomly sample 57 documents from the test set of DailyMail and ask 5 volunteers to evaluate the extracted summaries. While comparing with Refresh, we use the 20 documents (10 CNN and 10 DailyMail) provided by Narayan et al. (2018) to 4 volunteers. Tables 3 and 4 show the results of human evaluation in these two settings. BANDITSUM is shown to be better than Refresh and SummaRuNNer in terms of overall quality and non-redundancy. These results indicate that the use of the true policy gradient, rather than the approximation used by Refresh, improves overall quality. It is interesting to observe that, even though BANDITSUM does not have an explicit redundancy avoidance mechanism, it actually outperforms the other systems on non-redundancy.

6.3 Learning Curve

Reinforcement learning methods are known for sometimes being unstable during training. However, this seems to be less of a problem for BANDITSUM, perhaps because it is formulated as a contextual bandit rather than a sequential labeling problem. We show this by comparing the validation curves generated by BANDITSUM and the state-of-the-art maximum likelihood-based model – SummaRuNNer (Nallapati et al., 2017) (Figure 1).

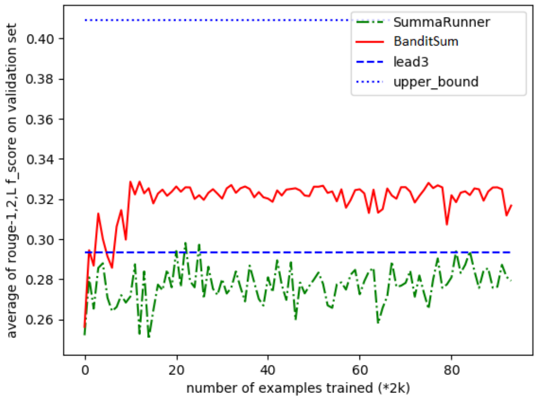

Figure 1: Average of ROUGE-1,2,L F1 scores on the Daily Mail validation set within one epoch of training on the Daily Mail training set. The x-axis (multiply by 2,000) indicates the number of data example the algorithms have seen. The supervised labels in SummaRuNNer are used to estimate the `upper_bound`.

From Figure 1, we observe that BANDITSUM converges significantly more quickly to good results than SummaRuNNer. Moreover, there is less variance in the performance of BANDITSUM.

One possible reason is that extractive summarization does not have well-defined supervised labels. There exists a mismatch between the provided labels and human-generated abstractive summaries. Hence, the gradient, computed from the maximum likelihood loss function, is not optimizing the evaluation metric of interest. Another important message is that both models are still far from the estimated upper bound[5], which shows that there is still significant room for improvement.

6.4 Run Time

On CNN/Daily mail dataset, our model's time-per-epoch is about 25.5 hours on a TITAN Xp. We trained the model for 3 epochs, which took about 76 hours in total. For comparison, DQN took about 10 days to train on a GTX 1080 (Yao et al., 2018). Refresh took about 12 hours on a single GPU to train (Narayan et al., 2018). Note that this figure does not take into account the significant time required by Refresh for pre-computing ROUGE scores.

7 Discussion: Contextual Bandit Setting Vs. Sequential Full RL Labeling

We conjecture that the contextual bandit (CB) setting is a more suitable framework for modeling extractive summarization than the sequential binary labeling setting, especially in the cases when good summary sentences appear later in the document. The intuition behind this is that models based on the sequential labeling setting are affected by the order of the decisions, which biases towards selecting sentences that appear earlier in the document. By contrast, our CB-based RL model has more flexibility and freedom to explore the search space, as it samples the sentences without replacement based on the affinity scores. Note that although we do not explicitly make the selection decisions in a sequential fashion, the sequential information about dependencies between sentences is implicitly embedded in the affinity scores, which are produced by bi-directional RNNs.

We provide empirical evidence for this conjecture by comparing BANDITSUM to the sequential RL model proposed by Wu and Hu (2018) (Figure 2) on two subsets of the data: one with good summary sentences appearing early in the article, while the other contains articles where good summary sentences appear late. Specifically, we construct two evaluation datasets by selecting the first 50 documents (D_{early}, i.e., best summary occurs early) and the last 50 documents (D_{late}, i.e., best summary occurs late) from a sample of 1000 documents that is ordered by the average extractive label index $\overline{idx}$. Given an article with n sentences indexed from $1, \dots, n$ and a greedy extractive labels set with three sentences (i, j, k)[6], the average index for the extractive label is computed by $\overline{idx} = (i + j + k)/3n$.

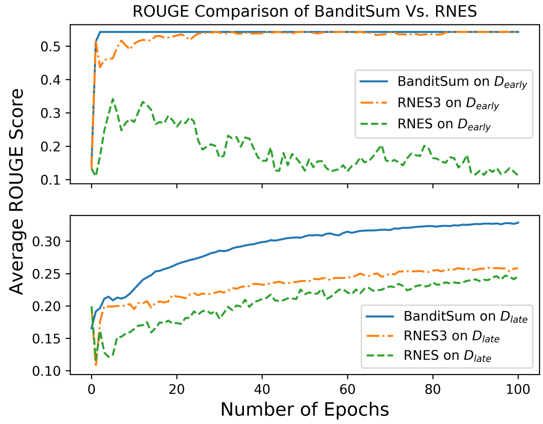

Figure 2: Model comparisons of the average value for ROUGE-1,2,L F1 scores ($\overline{f}$) on D_{early} and D_{late}. For each model, the results were obtained by averaging $\overline{f}$ across ten trials with 100 epochs in each trail. D_{early} and D_{late} consist of 50 articles each, such that the good summary sentences appear early and late in the article, respectively. We observe a significant advantage of BANDITSUM compared to RNES and RNES3 (based on the sequential binary labeling setting) on D_{late}.

Given these two subsets of the data, three different models (BANDITSUM, RNES and RNES3) are trained and evaluated on each of the two datasets without extractive labels. Since the original sequential RL model (RNES) is unstable without supervised pre-training, we propose the RNES3 model that is limited to select no more then three sentences. Starting with random initializations without supervised pre-training, we train each model ten times for 100 epochs and plot the learning curve of the average ROUGE-F1 score computed based on the trained model in Figure 2. We can clearly see that BANDITSUM finds a better so-

---

[5] The supervised labels for the `upper_bound` estimation are obtained using the heuristic described in Nallapati et al. (2017).

[6] For each document, a length-3 extractive summary with near-optimal ROUGE score is selected following the heuristic proposed by Nallapati et al. (2017).

lution more quickly than RNES and RNES3 on both datasets. Moreover, it displays a significantly speed-up in the exploration and finds the best solution when good summary sentences appeared later in the document (D_{late}).

8 Conclusion

In this work, we presented a contextual bandit learning framework, BANDITSUM , for extractive summarization, based on neural networks and reinforcement learning algorithms. BANDITSUM does not require sentence-level extractive labels and optimizes ROUGE scores between summaries generated by the model and abstractive reference summaries. Empirical results show that our method performs better than or comparable to state-of-the-art extractive summarization models which must be pre-trained on extractive labels, and converges using significantly fewer update steps than competing approaches. In future work, we will explore the direction of adding an extra coherence reward (Wu and Hu, 2018) to improve the quality of extracted summaries in terms of sentence discourse relation.

Acknowledgements

The research was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC). The authors would like to thank Compute Canada for providing the computational resources.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations (ICLR)*.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. 2015. Learning summary prior representation for extractive summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*.

John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 406–407. ACM.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 19–25. ACM.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive summarization using continuous vector space models. In *Workshop on Continuous Vector Space Models and their Compositionality (CVSC) at EACL*, pages 31–39.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference for Learning Representations (ICLR)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR) Workshop*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gůlçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*.

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

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7008–7024.

Cody Rioux, Sadid A Hasan, and Yllias Chali. 2014. Fear the reaper: A system for automatic multi-document summarization with reinforcement learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 681–690.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*, pages 256–265.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1073–1083.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1057–1063.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2692–2700.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 985–992.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

Kaichun Yao, Libo Zhang, Tiejian Luo, and Yanjun Wu. 2018. Deep reinforcement learning for extractive document summarization. *Neurocomputing*, 284:52–62.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL)*, pages 452–462.

Wenpeng Yin and Yulong Pei. 2015. Optimizing sentence modeling and selection for document summarization. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1383–1389.

A Word-Complexity Lexicon and A Neural Readability Ranking Model for Lexical Simplification

Mounica Maddela and **Wei Xu**
Department of Computer Science and Engineering
The Ohio State University
{maddela.4, xu.1265}@osu.edu

Abstract

Current lexical simplification approaches rely heavily on heuristics and corpus level features that do not always align with human judgment. We create a human-rated word-complexity lexicon of 15,000 English words and propose a novel neural readability ranking model with a Gaussian-based feature vectorization layer that utilizes these human ratings to measure the complexity of any given word or phrase. Our model performs better than the state-of-the-art systems for different lexical simplification tasks and evaluation datasets. Additionally, we also produce SimplePPDB++, a lexical resource of over 10 million simplifying paraphrase rules, by applying our model to the Paraphrase Database (PPDB).[1]

1 Introduction

Lexical simplification is an important subfield that is concerned with the complexity of words or phrases, and particularly how to measure readability and reduce the complexity using alternative paraphrases. There are three major lexical simplification tasks which effectively resemble a pipeline: (i) Complex Word Identification (Paetzold and Specia, 2016a; Yimam et al., 2017; Shardlow, 2013b) which involves identifying complex words in the sentence; (ii) Substitution Generation (Glavaš and Štajner, 2015; Coster and Kauchak, 2011) which involves finding alternatives to complex words or phrases; and (iii) Substitution Ranking (Specia et al., 2012) which involves ranking the paraphrases by simplicity. Lexical simplification also has practical real-world uses, such as displaying alternative expressions of complex words as reading assistance for children (Kajiwara et al., 2013), non-native speakers (Petersen and Ostendorf, 2007; Pellow and Eskenazi, 2014), lay readers (Elhadad and Sutaria, 2007; Siddharthan and Katsos, 2010), or people with reading disabilities (Rello et al., 2013).

Most current approaches to lexical simplification heavily rely on corpus statistics and surface level features, such as word length and corpus-based word frequencies (read more in §5). Two of the most commonly used assumptions are that simple words are associated with shorter lengths and higher frequencies in a corpus. However, these assumptions are not always accurate and are often the major source of errors in the simplification pipeline (Shardlow, 2014). For instance, the word *foolishness* is simpler than its meaning-preserving substitution *folly* even though *foolishness* is longer and less frequent in the Google 1T Ngram corpus (Brants and Franz, 2006). In fact, we found that 21% of the 2272 meaning-equivalent word pairs randomly sampled from PPDB[2] (Ganitkevitch et al., 2013) had the simpler word longer than the complex word, while 14% had the simpler word less frequent.

To alleviate these inevitable shortcomings of corpus and surface-based methods, we explore a simple but surprisingly unexplored idea – creating an English lexicon of 15,000 words with word-complexity ratings by humans. We also propose a new neural readability ranking model with a Gaussian-based feature vectorization layer, which can effectively exploit these human ratings as well as other numerical features to measure the complexity of any given word or phrase (including those outside the lexicon and/or with sentential context). Our model significantly outperforms the state-of-the-art on the benchmark SemEval-2012 evaluation for Substitution Ranking (Specia et al.,

---

[1] The code and data are publicly available on the authors' homepages and GitHub: `https://github.com/mounicam/lexical_simplification`.

[2] PPDB is a large paraphrase database derived from static bilingual translation data available at: `http://paraphrase.org`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3749–3760
Brussels, Belgium, October 31 - November 4, 2018.

2012; Paetzold and Specia, 2017), with or without using the manually created word-complexity lexicon, achieving a Pearson correlation of 0.714 and 0.702 respectively. We also apply the new ranking model to identify lexical simplifications (e.g., *commemorate* → *celebrate*) among the large number of paraphrase rules in PPDB with improved accuracy compared to previous work for Substitution Generation. At last, by utilizing the word-complexity lexicon, we establish a new state-of-the-art on two common test sets for Complex Word Identification (Paetzold and Specia, 2016a; Yimam et al., 2017). We make our code, the word-complexity lexicon, and a lexical resource of over 10 million paraphrase rules with improved readability scores (namely SimplePPDB++) all publicly available.

2 Constructing A Word-Complexity Lexicon with Human Judgments

We first constructed a lexicon of 15,000 English words with word-complexity scores assessed by human annotators.[3] Despite the actual larger English vocabulary size, we found that rating the most frequent 15,000 English words in Google 1T Ngram Corpus[4] was effective for simplification purposes (see experiments in §4) as our neural ranking model (§3) can estimate the complexity of any word or phrase even out-of-vocabulary.

We asked 11 non-native but fluent English speakers to rate words on a 6-point Likert scale. We found that an even number 6-point scale worked better than a 5-point scale in a pilot experiment with two annotators, as the 6-point scheme allowed annotators to take a natural two-step approach: first determine whether a word is simple or complex; then decide whether it is 'very simple' (or 'very complex'), 'simple' (or 'complex'), or 'moderately simple' (or 'moderately complex'). For words with multiple capitalized versions (e.g., *nature*, *Nature*, *NATURE*), we displayed the most frequent form to the annotators. We also asked the annotators to indicate the words for which they had trouble assessing their complexity due to ambiguity, lack of context or any other reason. All the annotators reported little difficulty, and explained possible reasons such as that word *bug* is simple regardless of its meaning as an *insect* in biology or an *error* in computer software.[5]

Word	**Avg**	**A1**	**A2**	**A3**	**A4**	**A5**
watch	1.0	1	1	1	1	1
muscles	1.6	2	1	2	2	1
sweatshirts	1.8	2	1	2	3	1
giant	2.0	2	3	1	1	3
pattern	2.4	2	3	2	3	2
Christianity	2.8	3	2	2	3	4
educational	3.2	3	3	3	3	4
revenue	3.6	4	4	3	3	4
cortex	4.2	4	4	4	4	5
crescent	4.6	5	5	5	5	3
Memorabilia	5.4	5	6	6	5	5
assay	5.8	6	6	6	5	6

Table 1: Word-Complexity lexicon consists of English words and their complexity scores obtained by averaging over human ratings. A1, A2, A3, A4 and A5 are ratings by five different annotators on a 6-point Likert scale (1 is the simplest and 6 is the most complex).

With our hired annotators, we were able to have most annotators complete half or the full list of 15,000 words for better consistency, and collected between 5 and 7 ratings for each word. It took most annotators about 2 to 2.5 hours to rate 1,000 words. Table 1 shows few examples from the lexicon along with their human ratings.

In order to assess the annotation quality, we computed the Pearson correlation between each annotator's annotations and the average of the rest of the annotations (Agirre et al., 2014). For our final word-complexity lexicon, we took an average of the human ratings for each word, discarding those (about 3%) that had a difference ≥ 2 from the mean of the rest of the ratings. The overall inter-annotator agreement improved from 0.55 to 0.64 after discarding the outlying ratings. For the majority of the disagreements, the ratings of one annotator and the mean of the rest were fairly close: the difference is ≤ 0.5 for 47% of the annotations; ≤ 1.0 for 78% of the annotations; and ≤ 1.5 for 93% of the annotations on the 6-point scale. We hired annotators of different native languages intentionally, which may have contributed to the variance in the judgments.[6] We leave further investigation and possible crowdsourcing annotation to future work.

---

[3] Download at `https://github.com/mounicam/lexical_simplification`

[4] `https://catalog.ldc.upenn.edu/ldc2006t13`

[5] The word-happiness lexicon (Dodds et al., 2011) of 10,222 words was also similarly created by human rating on the most frequent words without context or word-sense disambiguation.

[6] One recent work similarly observed lower inter-annotator agreement among non-native speakers than native speakers when asked to identify complex words in given text paragraphs (Yimam et al., 2017).

3 Neural Readability Ranking Model for Words and Phrases

In order to predict the complexity of any given word or phrase, within or outside the lexicon, we propose a Neural Readability Ranking model that can leverage the created word-complexity lexicon and take context (if available) into account to further improve performance. Our model uses a Gaussian-based vectorization layer to exploit numerical features more effectively and can outperform the state-of-the-art approaches on multiple lexical simplification tasks with or without the word-complexity lexicon. We describe the general model framework in this section, and task-specific configurations in the experiment section (§4).

3.1 Neural Readability Ranker (NRR)

Given a pair of words/phrases $\langle w_a, w_b \rangle$ as input, our model aims to output a real number that indicates the relative complexity $P(y|\langle w_a, w_b \rangle)$ of w_a and w_b. If the output value is negative, then w_a is simpler than w_b and vice versa. Figure 1 shows the general architecture of our ranking model highlighting the three main components:

1. An input **feature extraction layer** (§3.2) that creates lexical and corpus-based features for each input $f(w_a)$ and $f(w_b)$, and pairwise features $f(\langle w_a, w_b \rangle)$. We also inject the word-complexity lexicon into the model as a numerical feature plus a binary indicator.
2. A **Gaussian-based feature vectorization layer** (§3.3) that converts each numerical feature, such as the lexicon scores and n-gram probabilities, into a vector representation by a series of Gaussian radial basis functions.
3. A feedforward neural network performing regression with one **task-specific output node** that adapts the model to different lexical simplification tasks (§4).

Our model first processes each input word or phrase in parallel, producing vectorized features. All the features are then fed into a joint feedforward neural network.

3.2 Features

We use a combination of rating scores from the word-complexity lexicon, lexical and corpus features (Pavlick and Callison-Burch, 2016) and collocational features (Paetzold and Specia, 2017).

We inject the word-complexity lexicon into the NRR model by adding two features for each input word or phrase: a 0-1 binary feature representing the presence of a word (the longest word in a multi-word phrase) in the lexicon, and the corresponding word complexity score. For out-of-vocabulary words, both features have the value 0. We back-off to the complexity score of the lemmatized word if applicable. We also extract the following features: phrase length in terms of words and characters, number of syllables, frequency with respect to Google Ngram corpus (Brants and Franz, 2006), the relative frequency in Simple Wikipedia with respect to normal Wikipedia (Pavlick and Nenkova, 2015) and ngram probabilities from a 5-gram language model trained on the SubIMDB corpus (Paetzold and Specia, 2016c), which has been shown to work well for lexical simplification. For a word w, we take language model probabilities of all the possible n-grams within the context window of 2 to the left and right of w. When w is a multi-word phrase, we break w into possible n-grams and average the probabilities for a specific context window.

For an input pair of words/phrases $\langle w_a, w_b \rangle$, we include individual features $f(w_1)$, $f(w_2)$ and the differences $f(w_a) - f(w_b)$. We also use pairwise features $f(\langle w_a, w_b \rangle)$ including cosine similarity $cos(\overrightarrow{w}_a, \overrightarrow{w}_b)$ and the difference $\overrightarrow{w}_a - \overrightarrow{w}_b$ between the word2vec (Mikolov et al., 2013) embedding of the input words. The embeddings for a mutli-word phrase are obtained by averaging the embeddings of all the words in the phrase. We use the 300-dimensional embeddings pretrained on the Google News corpus, which is released as part of the word2vec package.[7]

3.3 Vectorizing Numerical Features via Gaussian Binning

Our model relies primarily on numerical features as many previous approaches for lexical simplification. Although these continuous features can be directly fed into the network, it is helpful to exploit fully the nuanced relatedness between different intervals of feature values.

We adopt a smooth binning approach and project each numerical feature into a vector representation by applying multiple Gaussian radial basis functions. For each feature f, we divide its

---

[7] `https://code.google.com/archive/p/word2vec/`

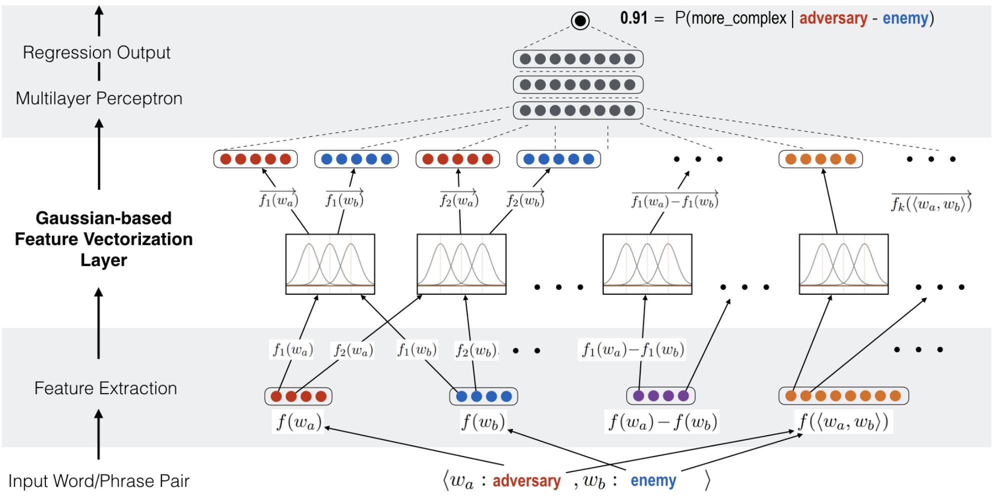

Figure 1: The neural readability ranking (NRR) model.

value range $[f_{min}, f_{max}]$ evenly into k bins and place a Gaussian function for each bin with the mean μ_j ($j \in \{1, 2, \dots, k\}$) at the center of the bin and standard deviation σ. We specify σ as a fraction γ of bin width:

$$\sigma = \frac{1}{k}(f_{max} - f_{min}) \cdot \gamma \tag{1}$$

where γ is a tunable hyperparameter in the model. For a given feature value $f(\cdot)$, we then compute the distance to each bin as follows:

$$d_j(f(\cdot)) = e^{-\frac{(f(\cdot)-\mu_j)^2}{2\sigma^2}} \tag{2}$$

and normalize to project into a k-dimensional vector $\overrightarrow{f(\cdot)} = (d_1, d_2, \dots, d_k)$.

We vectorize all the features except word2vec vectors, $\overrightarrow{f(w_a)}$, $\overrightarrow{f(w_b)}$, $\overrightarrow{f(w_a)-f(w_b)}$, and $\overrightarrow{f(\langle w_a, w_b\rangle)}$, then concatenate them as inputs. Figure 2 presents a motivating t-SNE visualization of the word-complexity scores from the lexicon after the vectorization in our NRR model, where different feature value ranges are gathered together with some distances in between.

3.4 Training and Implementation Details

We use PyTorch framework to implement the NRR model, which consists of an input layer, three hidden layers with eight nodes in each layer and the *tanh* activation function, and a single node linear output layer. The training objective is to minimize the Mean Squared Error (MSE):

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{3}$$

where y_i and $\hat{y}_i$ are the true and predicted relative complexity scores of $\langle w_a, w_b\rangle$ which can be configured accordingly for different lexical simplification tasks and datasets, m is the number of training examples, and θ is the set of parameters of the NRR model. We use Adam algorithm (Kingma and Ba, 2014) for optimization and also apply a dropout of 0.2 to prevent overfitting. We set the rate to 0.0005 and 0.001 for experiments in (§4.1) and (§4.2) respectively. For Gaussian binning layer, we set the number of bins k to 10 and γ to 0.2 without extensive parameter tuning. For each experiment,we report results with 100 epochs.

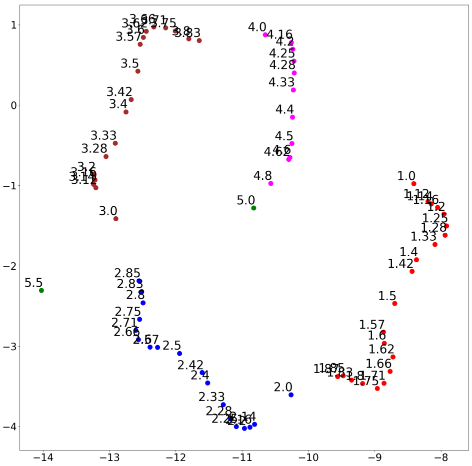

Figure 2: t-SNE visualization of the complexity scores, ranging between 1.0 and 5.0, of 300 random words from the word-complexity lexicon vectorized into 10-dimensional representations by applying Gaussian radial basis functions.

4 Lexical Simplification Applications

As the lexical simplification research field traditionally studies multiple sub-tasks and datasets, we present a series of experiments to demonstrate the effectiveness of our newly created lexicon and neural readability ranking (NRR) model.

4.1 Substitution Ranking

Given an instance consisting of a target complex word in a sentence and a set of candidate substitutions, the goal of the Substitution Ranking task is to rank the candidates in the order of their simplicity. In this section, we show that our proposed NRR model outperforms the state-of-the-art neural model on this task, with or without using the word-complexity lexicon.

Data. We use the dataset from the English Lexical Simplification shared-task at SemEval 2012 (Specia et al., 2012) for evaluation. The training and test sets consist of 300 and 1,710 instances, respectively, with a total of 201 target words (all single word, mostly polysemous) and each in 10 different sentences. One example of such instance contains a target complex word in context:

> *When you think about it, that's pretty terrible.*

and a set of candidate substitutions {*bad*, *awful*, *deplorable*}. Each instance contains at least 2 and an average of 5 candidates to be ranked. There are a total of 10034 candidates in the dataset, 88.5% of which are covered by our word-complexity lexicon and 9.9% are multi-word phrases (3438 unique candidates with 81.8% in-vocabulary and 20.2% multi-word).

Task-specific setup of the NRR model. We train the NRR model with every pair of candidates $\langle c_a, c_b \rangle$ in a candidate set as the input, and the difference of their ranks $r_a - r_b$ as the ground-truth label. For each such pair, we also include another training instance with $\langle c_b, c_a \rangle$ as the input and $r_b - r_a$ as the label. Given a test instance with candidate set C, we rank the candidates as follows: for every pair of candidates $\langle c_a, c_b \rangle$, the model predicts the relative complexity score $S(c_a, c_b)$; we then compute a single score $R(c_a) = \sum_{c_a \neq c_b \in C} S(c_a, c_b)$ for each candidate by aggregating pairwise scores and rank the candidates in the increasing order of these scores.

Comparison to existing methods. We compare with the state-of-the-art neural model (Paetzold and Specia, 2017) for substitution ranking with the best reported results on the SemEval 2012 dataset. Our baselines also include several other existing methods: Biran et al. (2011), Kajiwara et al. (2013), and Glavaš & Štajner (2015), which use carefully designed heuristic scoring functions to combine various information such as corpus statistics and semantic similarity measures from WordNet; Horn et al. (2014) and the Boundary Ranker (Paetzold and Specia, 2015), which respectively use a supervised SVM ranking model and pairwise linear classification model with various features. All of these methods have been implemented as part of the LEXenstein toolkit (Paetzold and Specia, 2015), which we use for the experimental comparisons here. In addition, we also compare to the best system (Jauhar and Specia, 2012) among participants at SemEval 2012, which used SVM-based ranking.

	P@1	Pearson
Biran et al. (2011)	51.3	0.505
Jauhar & Specia (2012)	60.2	0.575
Kajiwara et al. (2013)	60.4	0.649
Horn et al. (2014)	63.9	0.673
Glavaš & Štajner (2015)	63.2	0.644
Boundary Ranker	65.3	0.677
Paetzold & Specia (2017)	65.6	0.679
NRR$_{all}$	65.4	0.682
NRR$_{all+binning}$	66.6	0.702*
NRR$_{all+binning+WC}$	**67.3***	**0.714***

Table 2: Substitution Ranking evaluation on English Lexical Simplification shared-task of SemEval 2012. P@1 and Pearson correlation of our neural readability ranking (NRR) model compared to the state-of-the-art neural model (Paetzold and Specia, 2017) and other methods. $*$ indicates statistical significance ($p < 0.05$) compared to the best performing baseline (Paetzold and Specia, 2017).

Results. Table 2 compares the performances of our NRR model to the state-of-the-art results reported by Paetzold and Specia (2017). We use precision of the simplest candidate (P@1) and Pearson correlation to measure performance. P@1 is equivalent to TRank (Specia et al., 2012), the official metric for the SemEval 2012 English Lexical Simplification task. While P@1 captures the practical utility of an approach, Pearson correlation indicates how well the system's rankings correlate with human judgment. We train our NRR model with all the features (NRR$_{all}$) mentioned in §3.2 except the word2vec embedding features to avoid overfitting on the small training set. Our full model (NRR$_{all+binning+WC}$) exhibits a statistically significant improvement over the state-of-

paraphrases of ***'modification'*** ranked by simplicity	
SimplePPDB	*tweak, modify, process, variable, layout*
SimplePPDB++	*change, adjustment, amendment, shift, difference*
paraphrases of ***'aggregation'***	
SimplePPDB	*pod, swarm, node, clump, pool*
SimplePPDB++	*cluster, pool, collection, addition, grouping*
paraphrases of ***'of transnational corporation'***	
SimplePPDB	*of corporation, by corporation, of enterprise, of tncs, of business*
SimplePPDB++	*of business, of firm, of corporation, of company, of enterprise*
paraphrases of ***'should reject'***	
SimplePPDB	*refuse, discard, repudiate, shun, dismiss*
SimplePPDB++	*vote against, set aside, throw out, say no to, turn away*

Table 3: SimplePPDB++ includes lexical and phrasal paraphrases with improved readability ranking scores by our NRR$_{all+binning+WC}$ model. Shown are the top 5 ranked simplifications according to SimplePPDB++ for several input words/phrases, in comparison to the previous work of SimplePPDB (Pavlick and Callison-Burch, 2016).

the-art for both measures. We use paired bootstrap test (Berg-Kirkpatrick et al., 2012; Efron and Tibshirani, 1993) as it can be applied to any performance metric. We also conducted ablation experiments to show the effectiveness of the Gaussian-based feature vectorization layer ($_{+binning}$) and the word-complexity lexicon ($_{+WC}$).

4.2 SimplePPDB++

We also can apply our NRR model to rank the lexical and phrasal paraphrase rules in the Paraphrase Database (PPDB) (Pavlick et al., 2015), and identify good simplifications (see examples in Table 3). The resulting lexical resource, SimplePPDB++, contains all 13.1 million lexical and phrasal paraphrase rules in the XL version of PPDB 2.0 with readability scores in 'simplifying', 'complicating', or 'nonsense/no-difference' categories, allowing flexible trade-off between high-quality and high-coverage paraphrases. In this section, we show the effectiveness of the NRR model we used to create SimplePPDB++ by comparing with the previous version of SimplePPDB (Pavlick and Callison-Burch, 2016) which used a three-way logistic regression classifier. In next section, we demonstrate the utility of SimplePPDB++ for the Substitution Generation task.

Task-specific setup of NRR model. We use the same manually labeled data of 11,829 paraphrase rules as SimplePPDB for training and testing, of which 26.5% labeled as 'simplifying', 26.5% as 'complicating', and 47% as 'nonsense/no-difference'. We adapt our NRR model to perform the three-way classification by treating it as a regression problem. During training, we specify the ground truth label as follows: y = -1 if the paraphrase rule belongs to the 'complicating' class, y = +1 if the rule belongs to the 'simplifying'class, and y = 0 otherwise. For predicting, the network produces a single real-value output $\hat{y} \in [-1, 1]$ which is then mapped to three-class labels based on the value ranges for evaluation. The thresholds for the value ranges are -0.4 and 0.4 chosen by cross-validation.

	Acc.	**P$_{+1}$**	**P$_{-1}$**
Google Ngram Frequency	49.4	53.7	54.0
Number of Syllables	50.1	53.8	53.3
Character & Word Length	56.2	55.7	56.1
W2V	60.4	54.9	53.1
SimplePPDB	62.1	57.6	57.8
NRR$_{all}$	59.4	61.8	57.7
NRR$_{all+binning}$	64.1	62.1	59.8
NRR$_{all+binning+WC}$	**65.3***	**65.0***	**61.8***

Table 4: Cross-validation accuracy and precision of our neural readability ranking (NRR) model used to create SimplePPDB++, in comparison to the SimplePPDB and other baselines. P$_{+1}$ stands for the precision of 'simplifying' paraphrase rules and P$_{-1}$ for the precision of 'complicating' rules. * indicates statistical significance ($p < 0.05$) compared to the best performing baseline (Pavlick and Callison-Burch, 2016).

Comparison to existing methods. We compare our neural readability ranking (NRR) model used to create the SimplePPDB++ against SimplePPDB, which uses a multi-class logistic regression model. We also use several other baselines, including W2V which uses logistic regression with only word2vec embedding features.

Results. Following the evaluation setup in previous work (Pavlick and Callison-Burch, 2016), we compare accuracy and precision by 10-fold cross-validation. Folds are constructed in such a way that the training and test vocabularies are disjoint. Table 4 shows the performance of our model compared to SimplePPDB and other baselines. We use all the features (NRR$_{all}$) in §3.2 except for the context features as we are classifying paraphrase rules in PPDB that come with no context. SimplePPDB used the same features plus additional discrete features, such as POS tags, character unigrams and bigrams. Our neural readability ranking model alone with Gaussian binning (NRR$_{all+binning}$) achieves better accuracy and precision while using less features. Leveraging the lexicon (NRR$_{all+binning+WC}$) shows statistically significant improvements over SimplePPDB rankings based on the paired bootstrap test. The accuracy increases by 3.2 points, the precision for 'simplifying' class improves by 7.4 points and the precision for 'complicating' class improves by 4.0 points.

4.3 Substitution Generation

Substitution Generation is arguably the most challenging research problem in lexical simplification, which involves producing candidate substitutions for each target complex word/phrase, followed by the substitution ranking. The key focus is to not only have better rankings, but more importantly, to have a larger number of simplifying substitutions generated. This is a more realistic evaluation to demonstrate the utility of SimplePPDB++ and the effectiveness of the NRR ranking model we used to create it, and how likely such lexical resources can benefit developing end-to-end sentence simplification system (Narayan and Gardent, 2016; Zhang and Lapata, 2017) in future work.

Data. We use the dataset from (Pavlick and Callison-Burch, 2016), which contains 100 unique target words/phrases sampled from the Newsela Simplification Corpus (Xu et al., 2015) of news articles, and follow the same evaluation procedure. We ask two annotators to evaluate whether the generated substitutions are good simplifications.

Comparison to existing methods. We evaluate the correctness of the substitutions generated by SimplePPDB++ in comparison to several existing methods: Glavaš (Glavaš and Štajner, 2015), Kauchak (Coster and Kauchak, 2011), WordNet Generator (Devlin and Tait, 1998; Carroll et al., 1999), and SimplePPDB (Pavlick and Callison-Burch, 2016). Glavaš obtains candidates with the highest similarity scores in the GloVe (Pennington et al., 2014) word vector space. Kauchak's generator is based on Simple Wikipedia and normal Wikipedia parallel corpus and automatic word alignment. WordNet-based generator simply uses the synonyms of word in WordNet (Miller, 1995). For all the existing methods, we report the results based on the implementations in (Pavlick and Callison-Burch, 2016), which used SVM-based ranking. For both SimplePPDB and SimplePPDB++, extracted candidates are high quality paraphrase rules (quality score $\geq$3.5 for words and $\geq$4.0 for phrases) belonging to the same syntactic category as target word according to PPDB 2.0 (Pavlick et al., 2015).

	#PPs	**MAP**	**P@1**
Glavaš$_{(n=95)}$	—	22.8	13.5
WordNet$_{(n=82)}$	6.63	62.2	50.6
Kauchak$_{(n=48)}$	4.39	**76.4**†	68.9
SimplePPDB$_{(n=100)}$	8.77	67.8	78.0
SimplePPDB++$_{(n=100)}$	**9.52**	69.1	**80.2**

Table 5: Substitution Generation evaluation with Mean Average Precision, Precision@1 and the average number of paraphrases generated per target for each method. n is the number of target complex words/phrases for which the model generated > 0 candidates. Kauchak† has an advantage on MAP because it generates the least number of candidates. Glavaš is marked as '-' because it can technically generate as many words/phrases as are in the vocabulary.

Results. Table 5 shows the comparison of SimplePPDB and SimplePPDB++ on the number of substitutions generated for each target, the mean average precision and precision@1 for the final ranked list of candidate substitutions. This is a fair and direct comparison between SimplePPDB++ and SimplePPDB, as both methods have access to the same paraphrase rules in PPDB as potential candidates. The better NRR model we used in creating SimplePPDB++ allows improved selections and rankings of simplifying paraphrase rules than the previous version of SimplePPDB. As an additional reference, we also include the measurements for the other existing methods based on (Pavlick and Callison-Burch, 2016), which, by evaluation design, are focused on the comparison of precision while PPDB has full coverage.

4.4 Complex Word Identification

Complex Word Identification (CWI) identifies the difficult words in a sentence that need to be simplified. According to Shardlow (2014), this step can improve the simplification system by avoiding mistakes such as overlooking challenging words or oversimplifying simple words. In this section, we demonstrate how our word-complexity lexicon helps with the CWI task by injecting human ratings into the state-of-the-art systems.

Data. The task is to predict whether a target word/phrase in a sentence is 'simple' or 'complex', and an example instance is as follows:

Nine people were killed in the bombardment.

We conduct experiments on two datasets: (i) Semeval 2016 CWI shared-task dataset (Paetzold

	CWI SemEval 2016			CWIG3G2 2018		
	G-score	F-score	Accuracy	G-score	F-score	Accuracy
Length	47.8	10.7	33.2	70.8	65.9	67.7
Senses	57.9	12.5	43.6	67.7	62.3	54.1
SimpleWiki	69.7	16.2	58.3	73.1	66.3	61.6
NearestCentroid	66.1	14.8	53.6	75.1	66.6	76.7
SV000gg	77.3	24.3	77.6	74.9	73.8	78.7
WC-only	68.5	**30.5**	**87.7**	71.1	67.5	69.8
NearestCentroid$_{+WC}$	70.2	16.6	61.8	**77.3**	68.8	78.1
SV000gg$_{+WC}$	**78.1**	26.3	80.0	75.4	**74.8**	**80.2**

Table 6: Evaluation on two datasets for English complex word identification. Our approaches that utilize the word-complexity lexicon (WC) improve upon the nearest centroid (Yimam et al., 2017) and SV000gg (Paetzold and Specia, 2016b) systems. The best performance figure of each column is denoted in **bold** typeface and the second best is denoted by an underline.

CWI SemEval 2016	total (IV%)	unique (IV%)
simple	85621 (94.7%)	14129 (77.6%)
complex	4837 (57.4%)	3836 (54.8%)
CWIG3G2	total (IV%)	unique (IV%)
simple	20451 (89.8%)	5576 (82.1%)
complex	14428 (81.1%)	8376 (76.0%)

Table 7: Statistics of CWI datasets – total number of target words/phrases, number of unique targets, and in-vocabulary (IV) ratio with respect to our word-complexity lexicon.

and Specia, 2016a), which has been widely used for evaluating CWI systems and contains 2,237 training and 88,221 test instances from Wikipedia; and (ii) CWIG3G2 dataset (Yimam et al., 2017), which is also known as English monolingual CWI 2018 shared-task dataset (Yimam et al., 2018) and comprises of 27,299 training, 3,328 development and 4,252 test instances from Wikipedia and news articles. Table 7 shows the coverage of our word-complexity lexicon over the two CWI datasets.

Comparison to existing methods. We consider two state-of-the-art CWI systems: (i) the nearest centroid classifier proposed in (Yimam et al., 2017), which uses phrase length, number of senses, POS tags, word2vec cosine similarities, n-gram frequency in Simple Wikipedia corpus and Google 1T corpus as features; and (ii) SV000gg (Paetzold and Specia, 2016b) which is an ensemble of binary classifiers trained with a combination of lexical, morphological, collocational, and semantic features. The latter is the best performing system on the Semeval 2016 CWI dataset. We also compare to threshold-based baselines that use word length, number of word senses and frequency in the Simple Wikipedia.

Utilizing the word-complexity lexicon. We enhance the SV000gg and the nearest centroid classifier by incorporating the word-complexity lexicon as additional features as described in §3.2. We added our modifications to the implementation of SV000gg in the LEXenstein toolkit, and used our own implementation for the nearest centroid classifier. Additionally, to evaluate the word-complexity lexicon in isolation, we train a decision tree classifier with only human ratings as input (WC-only), which is equivalent to learning a threshold over the human ratings.

Results. We compare our enhanced approaches (SV000gg$_{+WC}$ and NC$_{+WC}$) and lexicon only approach (WC-only), with the state-of-the-art and baseline threshold-based methods. For measuring performance, we use F-score and accuracy as well as G-score, the harmonic mean of accuracy and recall. G-score is the official metric of the CWI task of Semeval 2016. Table 6 shows that the word-complexity lexicon improves the performance of SV000gg and the nearest centroid classifier in all the three metrics. The improvements are statistically significant according to the paired bootstrap test with $p < 0.01$. The word-complexity lexicon alone (WC-only) performs satisfactorily on the CWIG3G2 dataset, which effectively is a simple table look-up approach with extreme time and space efficiency. For CWI SemEval 2016 dataset, WC-only approach gives the best accuracy and F-score, though this can be attributed to the skewed distribution of dataset (only 5% of the test instances are 'complex').

5 Related Work

Lexical simplification: Prior work on lexical simplification depends on lexical and corpus-based features to assess word complexity. For complex word identification, there are broadly two lines of research: learning a frequency-based threshold over a large corpus (Shardlow, 2013b) or training an ensemble of classifiers over a combination of lexical and language model features

(Shardlow, 2013a; Paetzold and Specia, 2016a; Yimam et al., 2017; Kriz et al., 2018). Substitution ranking also follows similar trend. Biran et al. (2011) and Bott et al. (2012) employed simplicity measures based on word length and word frequencies from Wikipedia and Simple Wikipedia. Kajiwara et al. (2013) combined WordNet similarity measures with Simple Wikipedia frequencies. Glavaš and Štajner (2015) averaged the rankings produced by a collection of frequency, language model and semantic similarity features. Horn et al. (2014) trained an SVM classifier over corpus-based features.

Only recently, researchers started to apply neural networks to simplification tasks. To the best of our knowledge, the work by Paetzold and Specia (2017) is the first neural model for lexical simplification which uses a feedforward network with language model probability features. Our NRR model is the first pairwise neural ranking model to vectorize numeric features and to embed human judgments using a word-complexity lexicon of 15,000 English words.

Besides lexical simplification, another line of relevant research is sentence simplification that uses statistical or neural machine translation (MT) approaches (Xu et al., 2016; Nisioi et al., 2017; Zhang and Lapata, 2017; Vu et al., 2018; Guo et al., 2018). It has shown possible to integrate paraphrase rules in PPDB into statistical MT for sentence simplification (Xu et al., 2016) and bilingual translation (Mehdizadeh Seraj et al., 2015), while how to inject SimplePPDB++ into neural MT remains an open research question.

Lexica for simplification: There have been previous attempts to use manually created lexica for simplification. For example, Elhadad and Sutaria (2007) used UMLS lexicon (Bodenreider, 2007), a repository of technical medical terms; Ehara et al. (2010) asked non-native speakers to answer multiple-choice questions corresponding to 12,000 English words to study each user's familiarity of vocabulary; Kaji et al. (2012) and Kajiwara et al. (2013) used a dictionary of 5,404 Japanese words based on the elementary school textbooks; Xu et al. (2016) used a list of 3,000 most common English words; Lee and Yeung (2018) used an ensemble of vocabulary lists of different complexity levels. However, to the best of our knowledge, there is no previous study on manually building a large word-complexity lexicon with human judgments that has shown substantial improvements on automatic simplification systems. We were encouraged by the success of the word-emotion lexicon (Mohammad and Turney, 2013) and the word-happiness lexicon (Dodds et al., 2011, 2015).

Vectorizing features: Feature binning is a standard feature engineering and data processing method to discretize continuous values, more commonly used in non-neural machine learning models. Our work is largely inspired by recent works on entity linking that discussed feature quantization for neural models (Sil et al., 2017; Liu et al., 2016) and neural dependency parsing with embeddings of POS tags as features (Chen and Manning, 2014).

6 Conclusion

We proposed a new neural readability ranking model and showed significant performance improvement over the state-of-the-art on various lexical simplification tasks. We release a manually constructed word-complexity lexicon of 15,000 English words and an automatically constructed lexical resource, SimplePPDB++, of over 10 million paraphrase rules with quality and simplicity ratings. For future work, we would like to extend our lexicon to cover specific domains, different target users and languages.

Acknowledgments

We thank anonymous reviewers for their thoughtful comments. We thank Avirup Sil and Anastasios Sidiropoulos for valuable discussions, Sanja Štajner and Seid Muhie Yimam for sharing their code and data. We also thank the annotators: Jeniya Tabassum, Ashutosh Baheti, Wuwei Lan, Fan Bai, Alexander Konovalov, Chaitanya Kulkarni, Shuaichen Chang, Jayavardhan Reddy, Abhishek Kumar and Shreejit Gangadharan.

This material is based on research sponsored by the NSF under grants IIS-1822754 and IIS-1755898. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of the NSF or the U.S. Government.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Wiebe Janyce. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval@COLING)*.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An Empirical Investigation of Statistical Significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Or Biran, Samuel Brody, and Noemie Elhadad. 2011. Putting it Simply: A Context-Aware Approach to Lexical Simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*.

Olivier Bodenreider. 2007. The Unified Medical Language System (UMLS): integrating biomedical terminology. In *Proceedings of the Workshop on Biological, Translational, and Clinical Language Processing (BioNLP)*.

Stefan Bott, Luz Rello, Drndarvic Biljang, and Saiggon Horacio. 2012. Can Spanish Be Simpler? LexSiS: Lexical Simplification for Spanish. *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. *Linguistic Data Consortium (LDC)*.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying Text for Language-Impaired Readers. In *Proceedings of the 9th Conference of European chapter of the Association for Computational Linguistics (EACL)*.

Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

William Coster and David Kauchak. 2011. Learning to Simplify Sentences using Wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation (MTTG@ACL)*.

Siobhan Devlin and John Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic Databases*.

P. S. Dodds, E. M. Clark, S. Desu, M. R. Frank, A. J. Reagan, J. R. Williams, L. Mitchell, K. D. Harris, I. M. Kloumann, J. P. Bagrow, K. Megerdoomian, M. T. McMahon, B. F. Tivnan, and C. M. Danforth. 2015. Human language reveals a universal positivity bias. *Proceedings of the National Academy of Sciences (PNAS)*, 112(8):2389–2394.

Peter Sheridan Dodds, Kameron Decker Harris, Isabel M. Kloumann, Catherine A. Bliss, and Christopher M. Danforth. 2011. Temporal patterns of happiness and information in a global social network: Hedonometrics and Twitter. *Public Library of Science (PLOS ONE)*, 6(12):1–1.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall/CRC.

Yo Ehara, Nobuyuki Shimizu, Takashi Ninomiya, and Hiroshi Nakagawa. 2010. Personalized reading support for second-language web documents by collective intelligence. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI)*.

Noemie Elhadad and Komal Sutaria. 2007. Mining a Lexicon of Technical Terms and Lay Equivalents. In *Proceedings of the Workshop on Biological, Translational, and Clinical Language Processing (BioNLP)*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Goran Glavaš and Sanja Štajner. 2015. Simplifying Lexical Simplification: Do We Need Simplified Corpora? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Dynamic multi-level multi-task learning for sentence simplification. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a Lexical Simplifier Using Wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sujay Kumar Jauhar and Lucia Specia. 2012. UOW-SHEF: Simplex - Lexical Simplicity Ranking based on Contextual and Psycholinguistic features. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval@NAACL)*.

Nobuhiro Kaji, Daisuke Kawahara, Sadao Kurohash, and Satoshi Sato. 2012. Verb Paraphrase based on Case Frame Alignment. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tomoyuki Kajiwara, Hiroshi Matsumoto, and Kazuhide Yamamoto. 2013. Selecting Proper Lexical Paraphrase for Children. In *Proceedings of the 25th Conference on Computational Linguistics and Speech Processing (ROCLING)*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*.

Reno Kriz, Eleni Miltsakaki, Marianna Apidianaki, and Chris Callison-Burch. 2018. Simplification using paraphrases and context-based lexical substitution. In *The 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

John Lee and Chak Yan Yeung. 2018. Personalizing lexical simplification. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Dan Liu, Wei Lin, Shiliang Zhang, Si Wei, and Hui Jiang. 2016. Neural Networks Models for Entity Discovery and Linking. *Computing Research Repository (CoRR)*, 1611.03558.

Ramtin Mehdizadeh Seraj, Maryam Siahbani, and Anoop Sarkar. 2015. Improving Statistical Machine Translation with a Multilingual Paraphrase Database. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at International Conference on Learning Representations (ICLR)*.

George A. Miller. 1995. Wordnet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29(3):436–465.

Shashi Narayan and Claire Gardent. 2016. Unsupervised Sentence Simplification Using Deep Semantics. In *Proceedings of the 9th International Conference on Natural Language Generation (INLG)*.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring Neural Text Simplification Models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Gustavo Paetzold and Lucia Specia. 2015. LEXenstein: A Framework for Lexical Simplification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing: System Demonstrations (ACL-IJCNLP)*.

Gustavo Paetzold and Lucia Specia. 2016a. SemEval 2016 Task 11: Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval@NAACL)*.

Gustavo Paetzold and Lucia Specia. 2016b. SV000gg at Semeval-2016 Task 11: Heavy Gauge Complex Word Identification with System Voting. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval@NAACL)*.

Gustavo Paetzold and Lucia Specia. 2017. Lexical Simplification with Neural Ranking. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Gustavo Henrique Paetzold and Lucia Specia. 2016c. Unsupervised Lexical Simplification for Non-Native Speakers. In *Proceedings of the 30th Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*.

Ellie Pavlick and Chris Callison-Burch. 2016. Simple PPDB: A paraphrase database for simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ellie Pavlick and Ani Nenkova. 2015. Inducing Lexical Style Properties for Paraphrase and Genre Differentiation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevich, and Chris Callison-Burch Ben Van Durme. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

David Pellow and Maxine Eskenazi. 2014. An Open Corpus of Everyday Documents for Simplification Tasks. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR@EACL)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sarah E Petersen and Mari Ostendorf. 2007. Text Simplification for Language Learners: A Corpus Analysis. In *In Proceedings of Workshop on Speech and Language Technology in Education (SLaTE)*.

Luz Rello, Ricardo Baeza-Yates, and Horacio Saggion. 2013. The Impact of Lexical Simplification by Verbal Paraphrases for People with and without Dyslexia. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*.

Matthew Shardlow. 2013a. A Comparison of Techniques to Automatically Identify Complex Words. In *Proceedings of the ACL Student Research Workshop*.

Matthew Shardlow. 2013b. The CW Corpus: A New Resource for Evaluating the Identification of Complex Words. In *Proceedings of the 2nd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR@ACL)*.

Matthew Shardlow. 2014. Out in the Open: Finding and Categorising Errors in the Lexical Simplification Pipeline. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC)*.

Advaith Siddharthan and Napoleon Katsos. 2010. Reformulating Discourse Connectives for Non-Expert Readers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2017. Neural Cross-Lingual Entity Linking. In *Proceedings of the 30th Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*.

Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 2012. SemEval-2012 Task 1: English Lexical Simplification. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval@NAACL)*.

Tu Vu, Baotian Huu, Tsendsuren Munkhdalai, and Hong Yu. 2018. Sentence Simplification with Memory-Augmented Neural Networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Association for Computational Linguistics (TACL)*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics (TACL)*, 4:401–415.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H. Paetzold, Specia Lucia, Sanja tajner, Anas Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications (BEA@NAACL)*.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. CWIG3G2 - Complex Word Identification Task across Three Text Genres and Two User Groups. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP)*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Learning Latent Semantic Annotations for Grounding Natural Language to Structured Data

Guanghui Qin[1][*] **Jin-Ge Yao**[2] **Xuening Wang**[3] **Jinpeng Wang**[2] **Chin-Yew Lin**[2]
[1]Peking University, [2]Microsoft Research Asia, [3]University of California Los Angeles
`ghq@pku.edu.cn, sherry9788@g.ucla.edu`
`{jinge.yao, jinpwa, cyl}@microsoft.com`

Abstract

Previous work on grounded language learning did not fully capture the semantics underlying the correspondences between structured world state representations and texts, especially those between numerical values and lexical terms. In this paper, we attempt at learning explicit latent semantic annotations from paired structured tables and texts, establishing correspondences between various types of values and texts. We model the joint probability of data fields, texts, phrasal spans, and latent annotations with an adapted semi-hidden Markov model, and impose a soft statistical constraint to further improve the performance. As a by-product, we leverage the induced annotations to extract templates for language generation. Experimental results suggest the feasibility of the setting in this study, as well as the effectiveness of our proposed framework. [1]

1 Introduction

The meaning of natural language should always be accompanied by a context. *Grounded language acquisition* aims at learning the meaning of language in the context of an observed world state. A solution framework typically addresses the following subproblems: segmenting the text into meaningful phrasal units, determining which world state information is being referred to, and finding proper alignments from these units to the events of values in the world state.

The task has attracted much attention from the NLP community with a special focus on aligning text descriptions onto processed, structured event records (Snyder and Barzilay, 2007; Liang et al., 2009; Hajishirzi et al., 2011). Various statistical models have been proposed, attempting at characterizing the interaction between text spans and categorical values (e.g., *direction*=*'East'*) or strings (e.g., *person names*). The previously addressed term *semantic correspondences* narrowly describes the process of aligning natural language spans to different data fields.

TEAM	WIN	LOSS	PTS	...	PT-QT4
Raptors	33	15	120	...	31
Wizards	31	17	116	...	15

> The Toronto Raptors (33 - 15) barely edged out the Washington Wizards (31 - 17) 120 - 116 in overtime Saturday at the Verizon Center. The Raptors took a great lead before the fourth quarter, but the Wizards fought back by outscoring Toronto 31 - 15 ....

Figure 1: An example of the summary and the corresponding table.

However, there still exists a gap between alignment results and the underlying semantics. People tend to use various phrases to describe information that are inferred from different amounts of numerical values in a data field, or values derived from additional operations over fields. Consider the example description for a basketball game shown in Figure 1. The phrase *edged out* in the first sentence implies the fact that *the Toronto Raptors* had beaten their opponent by a relatively narrow margin. This could only be derived from an operation of subtraction between two scores that correspond to the field `PTS` for both teams in the event table, which leads to a relatively small difference of only four points. Previous efforts on learning semantic correspondences relying on *categorical* distributions (Liang et al., 2009) or *string pattern features* (Hajishirzi et al., 2012; Koncel-Kedziorski et al., 2014) do not have the capability to accurately capture *numerical information*, especially for the part that *does not appear explicitly* in the table and needs to be inferred.

---

[*] Contribution during internship at Microsoft Research Asia.

[1] Our implementation is available at `https://github.com/hiaoxui/D2T-Grounding`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3761–3771
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

Such kind of language grounding is important both for natural language understanding and for natural language generation. For language understanding, establishing explicit connections between symbols and values beyond ungrounded symbolic meaning representations will be useful for acquisition and inference of numerical commonsense (Narisawa et al., 2013). For language generation, properly aligned information is the key to acquiring patterns of various lexical choices under different world states (Roy and Reiter, 2005).

In this work, we make a step towards more explicit semantic correspondences between structured data and texts. Rather than only producing coarse alignments between data fields and text spans, we try to detect the latent semantics underlying these alignments by prompting explicit semantic annotations. We make the first attempt at utilizing publicly available datasets originally prepared for data-to-text language generation to produce such annotations for words and phrases in natural language without additional supervision.

Specifically, we conduct our study on a recently released dataset of descriptions for NBA basketball games with structured tables of game records. Different from a few popular datasets that have been well conjectured to be produced from rules (Reiter, 2017), the summaries are all written by humans. The text contains some numbers and proper nouns, which are easier to establish correspondence with data. However, the majority of texts contain many informative words, some of which need to be inferred indirectly from various types of values in the data cells. We want to establish explicit correspondences for them.

We derive a set of semantic labels from original data fields (Sec 4.1). These labels could be executed to establish direct correspondences to one or more values in the structured table. No annotation on the original dataset means unsupervised learning from weak distant supervision should be conducted. We design a semi-hidden Markov model to address this problem (Sec 4.2), which could align a semantic label to a word span. In the model, we leverage continuous probability distributions to model the correspondences between numerical values and lexical terms, which has not been well captured in previous work. To address the emerged issue of "garbage collection" that commonly appears in statistical alignment models (Sec 4.5), we add a soft statistical constraint via posterior regularization (Ganchev et al., 2010). As a by-product, we also show how the derived semantic annotations could be used to induce descriptive templates for data-to-text generation (Sec 5). Experimental results (Sec 6) suggest the feasibility of the setting in this study, and show the effectiveness of our proposed framework.

2 Related work

Grounded language acquisition has aroused wide interest in various disciplines (Siskind, 1996; Yu and Ballard, 2004; Gorniak and Roy, 2007; Yu and Siskind, 2013; Chrupała et al., 2015). Later work in the community of natural language processing also moved in this direction by relaxing the amount of supervision to enable a model to learn from ambiguous alignments (Kate and Mooney, 2007; Chen and Mooney, 2008). Some research aimed at establishing coarse alignments between simulated robot soccer game records and commentary sentences (Chen and Mooney, 2008; Chen et al., 2010; Bordes et al., 2010; Hajishirzi et al., 2011). For weather forecast domain, Liang et al. (2009) used a hierarchical hidden Markov model in order to map utterances to world states, which coped with segmentation and alignment together. More recently, Koncel-Kedziorski et al. (2014) tried to obtain the correspondences between real commentaries and structured football (soccer) events in multiple resolutions. We are distinct from this line of work in the fact that we aim at producing explicit semantic annotations that could capture information from structured tables. To achieve this goal, we need additional scaling or operations to enable data fields and values to be faithfully mapped onto texts. This will address the issue of the lack of consideration for the relationship between lexical terms and numerical values. Our approach makes a significant difference in that our framework could generalize to numerical values or value combinations that are unseen in training, and will not be simply reciting cooccurrence patterns of exact values in the training data.

Our work relates to learning executable semantic parsers under weak supervision. Early semantic parsing started from fully supervised training with annotated meaning representations available (Zettlemoyer and Collins, 2005; Ge and Mooney, 2006; Snyder and Barzilay, 2007), but more recent work focused on reducing the amount of supervision required (Artzi and Zettlemoyer, 2013). The intuition behind weakly supervised executable semantic parsing is that once the latent semantic representation has been executed, one could test whether the execution results could match the information with available weak supervision signals such as answers to natural language queries (Clarke et al., 2010; Liang et al., 2011), or task completion from instructional navigations (Misra et al., 2017). Such formulations have been adapted for question answering over structured tables (Pasupat and Liang, 2015; Krishnamurthy et al., 2017). However, the current research focus is to convert a natural language question into executable table queries and to directly retrieve results. They do not have the need of inference involving numerical commonsense implied by various lexical patterns. A few unsupervised approaches exist (Poon and Domingos, 2009; Poon, 2013) but only specific to translating language into queries in the highly structured database and cannot be applied to our domain.

Our approach is implemented as assigning tag annotations over text spans, which is conceptually related to fine-grained named entity tagging (Ling and Weld, 2012). Our setting only requires a rather weak and distant form of supervision from paired tables and texts without annotations for fine-grained alignments between phrases and data cells. Similar modeling and learning strategies could potentially be useful for considerably large tag space derived from structured knowledge bases in the future (Choi et al., 2018).

The feasibility of this work is partly due to the availability of data, mostly comes from the field of data-to-text language generation. Related work in data-to-text generation mainly focused on directly generating summary descriptions for structured data (Mei et al., 2016; Kiddon et al., 2016; Murakami et al., 2017; Wiseman et al., 2017), without establishing underlying semantic correspondences. Texts generated thereby can be fluent but not conforming to the input data, unlike template-based approaches where lexical choices could be directly controlled. In our work, we find the derived semantic correspondences between data and texts to be useful for template induction, either with simple heuristics to automatically extract description patterns (*how to say*) and corresponding triggers (*what & when to say*), or with more crafted discriminative learning approaches (cf. Angeli et al. (2010)).

3 Technical overview

Task Let $\mathcal{S}$ be the set of all world states, $\mathcal{W}$ be the set of all texts, $\mathcal{O}$ be the set of all executable operators, and $\mathcal{V}$ be the output space of $\mathcal{O}$. A world state $s \in \mathcal{S}$ is a table storing some information, or more specifically in this work, a tabular recording for a sports game. An operator $o \in \mathcal{O}$ can be executed on a world state to retrieve values, i.e., each o could be treated as a mapping of $\mathcal{S} \rightarrow \mathcal{V}$. The result of an operation can be a string, a continuous values or a discrete value. Meanwhile, each world state s is accompanied with a piece of description $\mathbf{w} \in \mathcal{W}$. Here $\mathbf{w}$ consists of a sequence of word tokens $\{w_i \in \mathbf{w}\}$. We further define a segmentation variable π, which could convert $\mathbf{w}$ into a sequence of word spans $\mathbf{c}$, containing each span of tokens $c^t \in \mathbf{c}$ that could be interpreted as a phrase. Note that we use superscript t to denote indices of phrases, and subscript i to denote indices of individual words. We further define $\mathbf{l}$ as a sequence of latent labels, and value of each l^t is an operator $o_i \in \mathcal{O}$. [2] For each world state s and corresponding description $\mathbf{w}$, we want to jointly find a proper segmentation π to obtain $\mathbf{c}$, and assign labels to every word span c^t.

We conduct this study on the ROTOWIRE subset of the openly available dataset released by Wiseman et al. (2017), containing text descriptions for NBA basketball games with structured tables of game statistics. Take Figure 1 as an example. The proper nouns (e.g. *Toronto Raptors*) appeared in the sentence can be assigned with a tag `Team_Name` in our tag set, which could then be aligned to the *Team Name* field in the table. Some numbers appearing in the text, such as *15* in the example, can be assigned with the label `Team_Losses`, with the executable annotation to extract the number of previously lost matches of the mentioned team. What we are more interested in is where the phrase *edge out* comes from. We are aiming at a model which is capable to capture the semantics behind the phrase *edge out*, which is used to describe an event that one team has beaten the other with close scores. In our annotation scheme, this phrase should be assigned with the tag `Team_Points_Delta`, and executing that will return the score difference between two teams.

The task is challenging in that there does not

---

[2] We will interchangeably use *labels*, *operators*, *tags* to refer to the latent executable semantic annotations in this paper.

Output Type	Example
String	`Team_City`
Categorical	`Player_Start_Position`
Numerical	`Team_Points_Delta`

Table 1: Examples from the derived tag set, where each tag could correspond to one of three types of values.

exist any other additional supervision signal. The learning process will mostly rely on statistical cooccurrences of information between the structured data and its text descriptions. Note that we assume a consistent structure (schema) throughout the whole dataset upon which the learning process will be performed.

Model To jointly learn word segmentations $\mathbf{c}$ and latent semantic annotations $\mathbf{l}$ between world state s and text $\mathbf{w}$ in a unified framework, we propose a generative model to characterize the joint distribution $P_s(\mathbf{l}, \pi, \mathbf{w}; \theta)$, parameterized by θ.

Learning The data contain paired texts and tables only, thus our model must learn segmentations and latent semantic annotations in an unsupervised fashion. The target is to maximize the complete data likelihood

$$\mathcal{L}(\theta) = \prod_{(s,\mathbf{w})\in\mathcal{D}} \sum_{\mathbf{l},\pi} P_s(\mathbf{l}, \pi, \mathbf{w}; \theta),$$

where $\mathcal{D}$ represents the whole training data. To reduce the search space in inference and to capture some patterns of content planning in the text descriptions, we adopt a Markov assumption over phrase segments, which leads to a hidden semi-Markov model (semi-HMM) (Murphy, 2002; Sarawagi and Cohen, 2005). The key part is to characterize different types of correspondences (Sec 4.2). We derive an expectation-maximization (EM) algorithm to perform maximum likelihood estimation, and introduce a soft statistical regularization to guide the model towards a better solution (Sec 4.5).

Inference Once the model has been trained, we use a Viterbi-like dynamic programming process to perform MAP inference to segment the texts and to assign the most likely tags for each span.

4 Framework

4.1 The set of annotations

We describe the process of how we derived our set of semantic annotations here. There are two kinds of specific tables for each NBA game in the dataset: *Box-Scores* and *Line-Scores*, respectively showing the performance statistics for individual players and the whole teams. The types of possible values for data fields are strings, categorical values and numerical values. *Box-Score* consists of 24 fields, of which four are string-values, one is categorical, the other 19 being numerical. *Line-Score* consists of 16 fields, containing two string-valued, one categorical, and 13 numerical fields. A semantic tag works on a specified kind of field type, from either team statistics or player statistics. For a tag that could align to one single field in the table, we let it return the exact value in the cell that could maximize the likelihood. For example, the tag `Team_City` is used to extract the name of the team in the table that corresponds to the current word span. For fields taking numerical values, we additionally allow tags to be able to perform mathematical operations, such as subtraction, between two values in the same field. [3] For example, the tag `Team_Points_Delta` can be executed to return the score difference between two teams. We list the different types of tags with examples shown in Table 1 and leave the entire tag set to Appendix A. Along with all these tags derived from the original data fields, we also include a special `NULL` tag which are supposed to be assigned to non-informative words or words containing information not contained in the given table.

Note that we impose little prior knowledge in this step. We simply over-generate all possible labels, and let the model figure out which part of them should be eventually used. Although the only compositional operation we used in this work is numeric subtraction, common operations that could produce string, categorical values or numbers could be easily introduced for other domains.

4.2 Semi-HMMs with continuous values

As previously mentioned, we will be modeling the joint distribution of word segmentation $\mathbf{c}$ and the latent semantic annotations $\mathbf{l}$ between paired world state s and text $\mathbf{w}$, which could be factorized as:

$$P_s(\mathbf{l}, \pi, \mathbf{w}) = P_s(\mathbf{w}, \pi|\mathbf{l}) \cdot P_s(\mathbf{l}),$$

and we write $P_s(\mathbf{w}, \pi|\mathbf{l})$ as $P_s(\mathbf{c}|\mathbf{l})$. In this section, we focus on the probability of the alignments between word spans and labels, namely, $P_s(\mathbf{c}|\mathbf{l})$.

Following Liang et al. (2009), we consider two aspects. One is *salience* that captures the intuition

---

[3] We limit the number of arguments within two in this work and leave more complex operators for future study.

that some fields should be more frequently mentioned than others (henceforth some latent tags should be more frequently triggered). The other is (local) *coherence*, which refers to the order in which the writer mentioning certain information tends to follow some patterns. To capture these two phenomena, we define a Markov model:

$$P_s(\mathbf{c}, \mathbf{l}) = \prod_t P(l^t | l^{t-1}) \cdot P_s(c^t | l^t),$$

where l^t is the annotated label at time stamp t, and we assume that the transition probabilities are independent of world state s. It resembles a standard form of HMMs, despite the subscript s in $P_s(\mathbf{c}|\mathbf{l})$. For different types of correspondences between l^t and c^t, we define different probability distributions to model $P_s(c^t|l^t)$:

(1) **Numerics-to-numerics**: The numbers in texts could sometimes be inaccurate due to some rounding customs, thus we use a Gaussian model for this type: $SoftIndicator(x, y|\sigma) = \mathcal{N}(x - y|0, \sigma)$, where $\mathcal{N}$ is the Gaussian density. When the output type of tag l^t is numeric and the word span c^t is a number, we set $P_s(c|l) = SoftIndicator(c, v_l|\sigma_l)$, where σ_l is different for different tags, and v_l is the corresponded value in the table for tag l. Note that when $\sigma \to 0$, $SoftIndicator$ reduces to an indicator function that only allows exact matching.

(2) **String-to-string**: Similarly for strings, since simple matching could fail if the text contains *Bob* to refer to *Bob Smith*. We simply use string matching to model the probability: $P_s(c|l) \propto Match(v_l, c)$, where the $Match$ function returns the number of shared words between cell value v_l and word c.

(3) **Category-to-string**: For labels correspond to discrete categorical values, such as `Sunday`, `PG` (*point guard*, a basketball position), we adopt the same method used by Liang et al. (2009): using a multinomial distribution over all word spans for each possible category:

$$P_s(c|l) = \nu_{c,v_l}, \quad \sum_c \nu_{c,v} = 1, \qquad (1)$$

where v_l is again the output value of tag l.

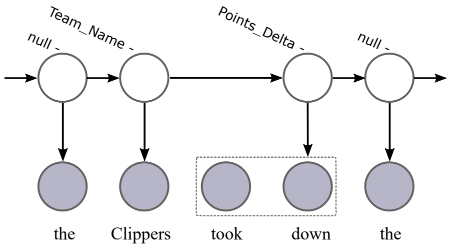

Figure 2: A Semi-HMM that can yield an entire phrase from one tag.

(4) **Numerics-to-string**: When the tag correspond to a numeric value v_l while the word span c is not a number, the problem resembles speech modeling (Huang et al., 1990). Applying the Bayes rule, we get: [4]

$$P_s(c|l) = P(c|l, v_l) \propto P(v_l|c, l) \cdot P(c|l),$$

where we collapse the relevant part from world state s into v_l. The intuition behind $P(v_l|c, l)$ is that when an informative word (e.g. *routed*) appears in the text, the corresponding values should have different chances to happen in the world, e.g. $P(30|routed, \texttt{Points\_Delta}) > P(3|routed, \texttt{Points\_Delta})$.

Due to the lack of prior knowledge on the distribution, we also model this term as Gaussian. [5] The result resembles a Gaussian mixture:

$$P_s(c|l) \propto \mathcal{N}(v_l; \mu_{c,l}, \sigma_{c,l}) \cdot \eta_{c,l},$$
$$\sum_c \eta_{c,l} = 1,$$

where $P(c|l) = \eta_{c,l}$ is also multinomial.

4.3 Modeling phrasal spans

Our model can enable phrase segmentation. Previously, Liang et al. (2009) treated the words inside a phrase individually and independently. This could be problematic in our scenario. For example, *take down* is used to describe a team defeating another, while separately both *take* and *down* are frequent words in general, making them difficult to be jointly assigned with the correct label as a whole. Instead, in our model we treat the phrasal

---

[4] We noticed that in parallel with our work, another study (Zhang et al., 2018) on verb selection for data-to-text generation also use the same strategy of Bayes rule to estimate parameters, and provide a noisy-channel interpretation.

[5] The double tails in Gaussian pdf have different interpretations: unlikely to occur, or unlikely to occur conditionally.

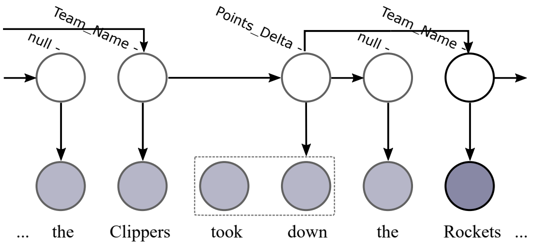

Figure 3: A slightly different Semi-HMM whose transition score is calculated by skipping `NULL` fields.

word span as a unit. The probability is assigned to the whole span of phrase instead of individual words, which will break the token-wise Markov property (Fig. 2, henceforth Semi-HMM). For efficient parameter estimation, We use a variant of the standard forward-backward algorithm by adding a parameter k, which is the maximum length of word spans, onto the Markov chain. We leave detailed descriptions to Appendix B.

4.4 Skipping null labels

Preliminary experiments suggest that the initial model have too many words assigned to the `NULL` tag. Informative alignments may not be adjacent, which breaks the simplest Markov assumptions. In our model, the transition score of two non-`NULL` labels can be calculated by skipping all the `NULL`s in between, as shown in Figure 3. This is implemented without breaking the overall Markov property with the following trick used in earlier work on statistical alignments (Brown et al., 1993): Suppose we have m labels (i.e., m latent states), we can design m different `NULL` labels that share the same emission score, while preserving their original outward-transition probabilities. The types of `NULL` labels are inherited from the previous label. This might seem to be wasteful at first sight as we use two-fold latent states, but the Markov property is successfully preserved, therefore simplifying our implementation.

4.5 Posterior regularization

The structured tables and text descriptions of the dataset were originally crawled from different sources. As a consequence, a non-negligible proportion of texts is in fact describing information outside the given table, such as historical records (e.g., *"win streak"*). Ideally, words in these parts of the text should remain unaligned, or in our setting, be annotated with the `NULL` tag. However, due to the notorious effect of *garbage collection* from statistical alignment (Brown et al., 1993), these words tend to be aligned to some irrelevant fields in the table which are rarely mentioned.

We address this issue by adding a soft statistical constraint in the form of posterior regularization (Ganchev et al., 2010; Graça et al., 2010). With posterior regularization, we could add certain types of statistical constraints to the E-step in the EM procedure, while keeping the inference tractable. The constraints should be in the form of:

$$\mathbb{E}[\mathbf{f}(\mathbf{w},\mathbf{l})] = \sum_i \mathbb{E}[\mathbf{f}(\mathbf{w},l_i)] \le \mathbf{b}_{\mathbf{w}}, \qquad (2)$$

where the features $\mathbf{f}$ should be defined on local cliques for tractable inference.

We use projected gradient descent to solve the E-step sub-problem in this work. The statistical constraint we add to the posterior is rather simple: For each sentence, we "encourage" at least a proportion of words to be aligned to `NULL` labels:

$$\mathbb{E}[-f(\mathbf{w},\mathbf{l})] \le -r_0 \cdot n, \qquad (3)$$

$$f(\mathbf{w},\mathbf{l}) = \sum_{i=1}^{n} \mathbb{1}(l_i = \texttt{NULL}), \qquad (4)$$

where r_0 is a adjustable ratio, n is the length of $\mathbf{w}$. We also tried other constraints but found this simple soft regularization performing well.

5 By-product: template induction

Intuitively, the assigned semantic correspondences could be useful to derive templates and trigger rules for language generation. In this work, we use the most straightforward heuristics to perform template induction, utilizing the established correspondences and inferred parameters. Specifically, we first blank out the correspondences of numerics-to-numerics and string-to-string to be empty slots and replace with the tag names. In the example of Figure 1, we could replace *Raptors* with `Team_Name`, and 120 with `Team_Points`, if they have been correctly aligned.

We also need to know when to use each template. We define a *template trigger* to be a quadruple $(c, l, \mu_{c,l}, \sigma_{c,l})$, where c is a phrase, l is a tag, $\mu_{c,l}$ and $\sigma_{c,l}$ are estimated Gaussian parameters. [6] We assign each template with a score to be the minimum probability for all triggers inside:

$$\text{score}(s,\mathbf{t}) = \min_i \mathcal{N}(t_i.l(s); t_i.\mu, t_i.\sigma), \qquad (5)$$

---

[6] To use a unified notation, for categorical-value triggers we set $\mu_{c,l} = \arg\max_{v_l} \nu_{c,v_l}$ (defined in (1)) and $\sigma_{c,l} = \epsilon$.

where $\mathbf{t} = \{t_i\}$ denotes all possible triggers in the template, and the tag l can be executed over the world state s to retrieve a value $l(s)$. We only consider sentences satisfying both of the following conditions in order to extract templates with high quality: (1) sentences aligned to $\leq$ two teams or one player, and (2) sentences with triggers derived from continuous distributions.

Now that the templates and triggers are ready for use, we will experiment with the following straightforward rules to perform data-to-text generation: For every game, we first generate a sentence describing the scoreline result, followed by three sentences describing other information about team performance. While keeping that no template is repeatedly used, we will then choose the template with the highest score for top ten players sorted by their game points.

6 Experiments

6.1 Experimental setup

We conducted experiments on the ROTOWIRE subset of the Wiseman et al. (2017) dataset. In our experiment, we restricted the maximum length of word span to two as a trade-off of speed and performance. Empirically, most of the phrases in the dataset are at a length of at most two. We empirically set the expected `NULL` ratio to be $r_0 = 0.5$.

We did the following pre-processing steps for all systems in comparison: we lemmatized all tokens in the sentences, and filtered out sentences containing less than five words since they are meaningless short sentences. To utilize the game dates, we converted them from calendar date to the day of week, e.g. 11/28/2016 is converted to *Monday* as a categorical value. Due to the huge noise in the ROTOWIRE dataset, containing many sentences irrelevant with their corresponding tables, we filtered out the sentences that contain no team or player names, or those that mention more than 2 players, as most of them are irrelevant texts.

Following Liang et al. (2009), we also used the parameters of a simpler model without Markov dependency (which was uniformly initialized) to initialize our complete model with obtained parameters, and then trained it until convergence. We adapted Liang et al. (2009)'s framework to the table schema in the ROTOWIRE dataset, and ran experiments accordingly as our baseline model.

6.2 Evaluation

6.2.1 Intrinsic evaluation

It is difficult to evaluate the accuracy of tag assignments for the entire dataset, since the tags are not annotated in the original data. We recruited three human annotators with familiarity in the domain of basketball games to label 300 sentences (around 8,000 tokens in total, and 30% of them are annotated with tags) from the test set. There exists a fraction (18%) where agreements were not made, we included all the proposed tags from the annotators to be correct. Also, because of the ambiguity of annotations, we use the base names of derived tags (e.g. `Rebounds_Delta`) for numerics-to-string relationship evaluation. (e.g. `Rebounds_Delta` is reduced to `Rebounds`) Finally, we calculated the precision and recall for non-`NULL` tag assignments at word-level.

	Precision	Recall	F1
Liang09	0.319	0.643	0.426
Liang09+PR	0.397	0.640	0.490
Semi-HMMs	0.254	0.765	0.381
Semi-HMMs+PR	**0.504**	**0.786**	**0.614**

Table 2: Word-level tag assignment results.

Correspondence type	Proportion	Accuracy
numerics-to-numerics	0.369	0.950
numerics-to-string	0.283	0.402
string-to-string	0.252	0.892
category-to-string	0.095	0.936

Table 3: The performance of Semi-HMMs+PR for different types of correspondences.

The results are shown in Table 2. The Liang et al. (2009) framework could still achieve around 65% recall, because there exist a large proportion of correspondences that could be easily captured by exact matches and simple categorical distributions. Our model without PR achieves lower precision than the baseline, because the baseline did not model numerics-to-string relationships and encountered less severe issues of garbage collection. We can observe that our initial model indeed outperforms the baseline system in recall, while PR helps a lot to avoid distraction from irrelevant information that should be tagged as `NULL`.

We also include more fine-grained results for different types of correspondences, shown in Table 3. As expected, numerics-to-string correspon-

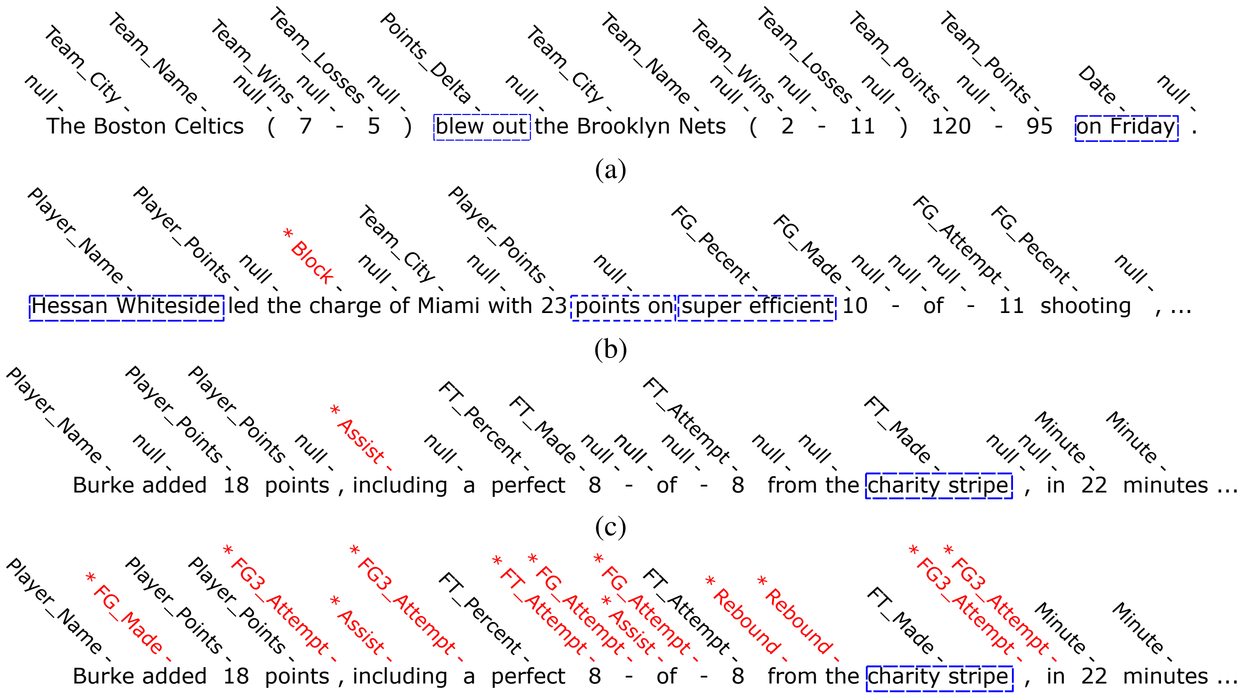

(a)

(b)

(c)

(d)

Figure 4: Example of latent tag assignment. Words in blue dashed boxes are phrases recognized by our model. The labels red with asterisks (*) are false assignments. (a), (b), (c) are example of Semi-HMMs-PR, while (d) is produced from the initial Semi-HMMs.

dences are the most difficult part in this study. Another notable thing is that although we found that around 40% of numerics-to-numerics correspondences were ambiguous due to the appearance of identical values from different table cells, our model could still achieve a high accuracy of 95.0%.

6.2.2 Extrinsic evaluation

We also tested how the derived templates could perform in language generation, when compared with the baseline using the same heuristics described in Sec 5. We report automatic metrics including BLEU scores and those based on relation extraction as proposed by Wiseman et al. (2017): precision & number of unique relations in generation (RG), precision & recall for content selection (CS), and content ordering (CO) score. These automatic metrics were designed for various aspects in NLG and may not all suit our main focus well, so we also conducted human evaluation on information correctness (1-5 scale ratings, the higher the better). We asked four human raters who are fluent in English and with familiarity in basketball terms to rate over outputs for 30 random games. Results are shown in Table 4. We can observe that templates derived from our model indeed outperform those from the baseline.

We put some inducted templates and generated text examples in the Appendix.

6.3 Analysis

Figure 4 shows some examples produced from our methods. Some of the alignments are meaningful, for example, the model assigned the word *perfect* with the annotation `FT_Percent`, which represents the percentage of free throws. Without PR, our model performed poorly by aligning many common words to those rarely mentioned cells. In this example, the `FT_Made` and `FT_Attempt` fields in the input table both have the same value 8, making it difficult for a model without proper local coherence modeling to distinguish between them. Because our initial model without PR cannot annotate `NULL` correctly, the Markov transition between these two numbers was intercepted by three meaningless tokens. However, after injecting the PR constraint, most of the unmentioned words were successfully identified. The model captured the pattern that `FT_Attempt` almost always follows `FT_Made`, making it correctly assigned these two labels.

We conducted ablation experiments for some

Model	RG(P%)	RG(#)	CS(P%)	CS(R%)	CO	BLEU	Correctness
Liang09+PR	85.83	33.29	14.33	31.09	6.25	8.34	2.60
Semi-HMMs+PR	90.47	41.79	21.63	50.17	9.63	9.45	3.58
Gold-standard	91.77	12.84	100	100	100	100	4.88

Table 4: Results for data-to-text generation (Kendall's W=0.83 from correctness raters)

of the components (Table 5). When setting the maximum phrase length to be $k = 1$, the model degenerates to a normal HMM. The performance measured by F1-score drops for only a little. One possible reason is that Semi-HMMs tend to output some meaningless combinations of words as phrases, such as the phrase *points on* in Figure 4 (b), which could lead to many redundant annotations that hampers precision albeit its help to recall. We also tried to disable the transition probabilities during both training and inference, which led to lower precision and lower recall naturally as there was no modeling for local coherence. Finally, by canceling the NULL-skipping mechanism, we found that the numerics-to-numerics annotation accuracy dropped from 95.0% to 88.8%. Many of the spurious numerics-to-numerics annotations, such as the *8* - *of* - *8* in Figure 4 (d), could be corrected using transition probabilities under the skipping-NULL mechanism (Figure 4 (c)).

	Precision	Recall	F1
Semi-HMMs+PR	0.504	0.786	0.614
HMM+PR	0.545	0.694	0.610
No transition	0.468	0.633	0.538
No skip	0.454	0.737	0.562

Table 5: Ablation results.

One additional advantage of our model is that we can easily verify what the model has captured. For the latent annotation `Team_Points_Delta`, we sort its corresponding phrases by weights $\frac{P(c|l)}{P(c)}$ and we list the top 12 weighted words in Figure 5. We can observe that most of the displayed phrases have strong semantic relationship with score differences. More interestingly, we found the mean and variance values estimated by the Gaussian distributions rather informative. When l = `Teams_Points_Delta`, we observed that $\mu_{l,\text{narrowly}} \approx 2$ while $\mu_{l,\text{blow out}} \approx 26$. We could infer the conditions under which some phrases should be used, providing useful insights for lexical choices in language generation.

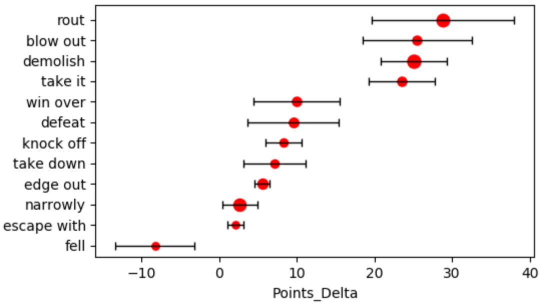

Figure 5: Mean±std for the top 12 weighted phrases assigned with `Team_Points_Delta`. (circle sizes are proportional to weights)

7 Conclusion

In this paper, we attempt to learn executable latent semantic annotations from paired structured tables and texts. We model the joint probability of data fields, texts, phrasal spans, and latent annotations with an adapted semi-hidden Markov model and impose a soft statistical constraint via posterior regularization. Experimental results suggest the feasibility of the setting in study and the effectiveness of our framework.

This is a preliminary study for using weak supervision from structured data and texts to address the challenging problem of language grounding. For future study, one could collect large-scale data and texts in other domains where more complex grounding on phrases such as "increasing trends" should be done. To enhance modeling power, unsupervised discriminative models that utilize rich features (Berg-kirkpatrick et al., 2010) could also be explored. We are also interested in collecting more high-quality parallel data to induce grounded compositional logic representations.

Acknowledgements

We would like to thank Hongyuan Mei and all the anonymous reviewers for giving helpful comments on an earlier draft of this paper.

References

G. Angeli, P. Liang, and D. Klein. 2010. A Simple Domain-Independent Probabilistic Approach to Generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Y. Artzi and L. S. Zettlemoyer. 2013. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics (TACL)*.

T. Berg-kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *North American Association for Computational Linguistics (NAACL)*.

A. Bordes, N. Usunier, and J. Weston. 2010. Label Ranking under Ambiguous Supervision for Learning Semantic Correspondences. In *International Conference on Machine Learning (ICML)*.

P. F. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*.

D. L. Chen, J. Kim, and R. J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research (JAIR)*.

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *International Conference on Machine Learning (ICML)*.

E. Choi, O. Levy, Y. Choi, and L. Zettlemoyer. 2018. Ultra-fine entity typing. In *Association for Computational Linguistics (ACL)*.

G. Chrupała, Á. Kádár, and A. Alishahi. 2015. Learning language through pictures. In *Association for Computational Linguistics (ACL)*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving Semantic Parsing from the World's Response. In *Computational Natural Language Learning (CoNLL)*.

K. Ganchev, J. V. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior Regularization for Structured Latent Variable Models. *Journal of Machine Learning Research (JMLR)*.

R. Ge and R. J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Association for Computational Linguistics (ACL)*.

P. Gorniak and D. Roy. 2007. Situated language understanding as filtering perceived affordances. *Cognitive Science*.

J. V. Graça, K. Ganchev, and B. Taskar. 2010. Learning Tractable Word Alignment Models with Complex Constraints. *Computational Linguistics*.

H. Hajishirzi, J. Hockenmaier, Erik T. Mueller, and E. Amir. 2011. Reasoning about RoboCup Soccer Narratives. In *Uncertainty in Artificial Intelligence (UAI)*.

H. Hajishirzi, M. Rastegari, A. Farhadi, and J. Hodgins. 2012. Semantic Understanding of Proefessional Soccer Commentaries. In *Uncertainty in Artificial Intelligence (UAI)*.

X. D. Huang, Y. Ariki, and M. A. Jack. 1990. *Hidden Markov Models for Speech Recognition*. Edinburgh University Press.

R. J. Kate and R. J. Mooney. 2007. Learning language semantics from ambiguous supervision. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

C. Kiddon, L. Zettlemoyer, and Y. Choi. 2016. Globally Coherent Text Generation with Neural Checklist Models. In *Empirical Methods in Natural Language Processing (EMNLP)*.

R. Koncel-Kedziorski, H. Hajishirzi, and A. Farhadi. 2014. Multi-Resolution Language Grounding with Weak Supervision. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Krishnamurthy, P. Dasigi, and M. Gardner. 2017. Neural Semantic Parsing with Type Constraints for Semi-Structured Tables. In *Empirical Methods in Natural Language Processing (EMNLP)*.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning Semantic Correspondences with Less Supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning Dependency-Based Compositional Semantics. In *Association for Computational Linguistics (ACL)*.

X. Ling and D. S. Weld. 2012. Fine-Grained Entity Recognition. In *AAAI Conference on Artificial Intelligence (AAAI)*.

H. Mei, M. Bansal, and M. R. Walter. 2016. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *North American Association for Computational Linguistics and Human Language Technology (NAACL-HLT)*.

D. Misra, J. Langford, and Y. Artzi. 2017. Mapping Instructions and Visual Observations to Actions with Reinforcement Learning. In *Empirical Methods in Natural Language Processing (EMNLP)*.

S. Murakami, A. Watanabe, A. Miyazawa, K. Goshima, T. Yanase, H. Takamura, and Y. Miyao. 2017. Learning to Generate Market Comments from Stock Prices. In *Association for Computational Linguistics (ACL)*.

K. P. Murphy. 2002. Hidden Semi-Markov Models (HSMMs).

K. Narisawa, Y. Watanabe, J. Mizuno, N. Okazaki, and K. Inui. 2013. Is a 204 cm Man Tall or Small? Acquisition of Numerical Common Sense from the Web. In *Association for Computational Linguistics (ACL)*.

P. Pasupat and P. Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Association for Computational Linguistics (ACL)*.

H. Poon. 2013. Grounded Unsupervised Semantic Parsing. In *Association for Computational Linguistics (ACL)*.

H. Poon and P. Domingos. 2009. Unsupervised semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

E. Reiter. 2017. You Need to Understand your Corpora! The Weathergov Example.

D. Roy and E. Reiter. 2005. Connecting language to the world. *Artificial Intelligence*.

S. Sarawagi and W. W. Cohen. 2005. Semi-Markov Conditional Random Fields for Information Extraction. In *Advances in Neural Information Processing Systems (NIPS)*.

J. M. Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*.

B. Snyder and R. Barzilay. 2007. Database-text alignment via structured multilabel classification. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

S. Wiseman, S. M. Shieber, and A. M. Rush. 2017. Challenges in Data-to-Document Generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

C. Yu and D. H. Ballard. 2004. On the Integration of Grounding Language and Learning Objects. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

H. Yu and J. M. Siskind. 2013. Grounded Language Learning from Video Described with Sentences. In *Association for Computational Linguistics (ACL)*.

L. S. Zettlemoyer and M. Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Uncertainty in Artificial Intelligence (UAI)*.

D. Zhang, J. Yuan, X. Wang, and A. Foster. 2018. Probabilistic Verb Selection for Data-to-Text Generation. *Transactions of the Association for Computational Linguistics (TACL)*.

Syntactic Scaffolds for Semantic Structures

Swabha Swayamdipta[♠] **Sam Thomson**[♠]
Kenton Lee[◊] **Luke Zettlemoyer**[♥] **Chris Dyer**[♡] **Noah A. Smith**[♥♣]

[♠]Language Technologies Institute, Carnegie Mellon University
[◊]Google AI Language
[♥]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♡]Google DeepMind
[♣]Allen Institute for Artificial Intelligence
`{swabha,sammthomson}@cs.cmu.edu` `{kentonl,cdyer}@google.com`
`{lsz,nasmith}@cs.washington.edu`

Abstract

We introduce the **syntactic scaffold**, an approach to incorporating syntactic information into semantic tasks. Syntactic scaffolds avoid expensive syntactic processing at runtime, only making use of a treebank during training, through a multitask objective. We improve over strong baselines on PropBank semantics, frame semantics, and coreference resolution, achieving competitive performance on all three tasks.

1 Introduction

As algorithms for the semantic analysis of natural language sentences have developed, the role of *syntax* has been repeatedly revisited. Linguistic theories have argued for a very tight integration of syntactic and semantic processing (Steedman, 2000; Copestake and Flickinger, 2000), and many systems have used syntactic dependency or phrase-based parsers as preprocessing for semantic analysis (Gildea and Palmer, 2002; Punyakanok et al., 2008; Das et al., 2014). Meanwhile, some recent methods forgo explicit syntactic processing altogether (Zhou and Xu, 2015; He et al., 2017; Lee et al., 2017; Peng et al., 2017).

Because annotated training datasets for semantics will always be limited, we expect that syntax—which offers an incomplete but potentially useful view of semantic structure—will continue to offer useful inductive bias, encouraging semantic models toward better generalization. We address the central question: is there a way for semantic analyzers to benefit from syntax without the computational cost of syntactic parsing?

We propose a multitask learning approach to incorporating syntactic information into learned representations of neural semantics models (§2). Our approach, the **syntactic scaffold**, minimizes an auxiliary supervised loss function, derived from a syntactic treebank. The goal is to steer the distributed, contextualized representations of words and spans toward accurate semantic *and* syntactic labeling. We avoid the cost of training or executing a full syntactic parser, and at test time (i.e., runtime in applications) the semantic analyzer has no additional cost over a syntax-free baseline. Further, the method does not assume that the syntactic treebank overlaps the dataset for the primary task.

Many semantic tasks involve labeling spans, including semantic role labeling (SRL; Gildea and Jurafsky, 2002) and coreference resolution (Ng, 2010) (tasks we consider in this paper), as well as named entity recognition and some reading comprehension and question answering tasks (Rajpurkar et al., 2016). These spans are usually syntactic constituents (cf. PropBank; Palmer et al., 2005), making phrase-based syntax a natural choice for a scaffold. See Figure 1 for an example sentence with syntactic and semantic annotations. Since the scaffold task is not an end in itself, we relax the syntactic parsing problem to a collection of independent span-level predictions, with no constraint that they form a valid parse tree. This means we never need to run a syntactic parsing algorithm.

Our experiments demonstrate that the syntactic scaffold offers a substantial boost to state-of-the-art baselines for two SRL tasks (§5) and coreference resolution (§6). Our models use the strongest available neural network architectures for these tasks, integrating deep representation learning (He et al., 2017) and structured prediction at the level of spans (Kong et al., 2016). For SRL, the base-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3772–3782
Brussels, Belgium, October 31 - November 4, 2018.

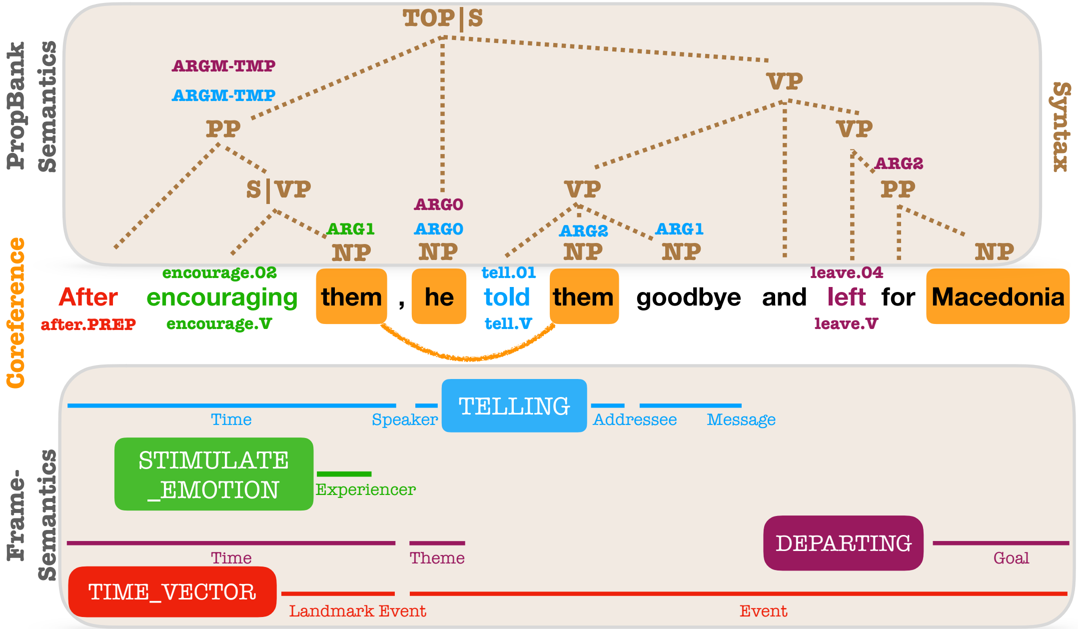

Figure 1: An example sentence with syntactic, PropBank and coreference annotations from OntoNotes, and author-annotated frame-semantic structures. PropBank SRL arguments and coreference mentions are annotated on top of syntactic constituents. All but one frame-semantic argument (Event) is a syntactic constituent. Targets evoke frames shown in the color-coded layers.

line itself is a novel globally normalized structured conditional random field, which outperforms the previous state of the art.[1] Syntactic scaffolds result in further improvements over prior work—3.6 absolute F_1 in FrameNet SRL, 1.1 absolute F_1 in PropBank SRL, and 0.6 F_1 in coreference resolution (averaged across three standard scores). Our code is open source and available at `https://github.com/swabhs/scaffolding`.

2 Syntactic Scaffolds

Multitask learning (Caruana, 1997) is a collection of techniques in which two or more tasks are learned from data with at least some parameters shared. We assume there is only one task about whose performance we are concerned, denoted T_1 (in this paper, T_1 is either SRL or coreference resolution). We use the term "scaffold" to refer to a second task, T_2, that can be combined with T_1 during multitask learning. A scaffold task is *only* used during training; it holds no intrinsic interest beyond biasing the learning of T_1, and after learning is completed, the scaffold is discarded.

A **syntactic scaffold** is a task designed to steer the (shared) model toward awareness of syntactic structure. It could be defined through a syntactic parser that shares some parameters with T_1's model. Since syntactic parsing is costly, we use simpler syntactic prediction problems (discussed below) that do not produce whole trees.

As with multitask learning in general, we do not assume that the same data are annotated with outputs for T_1 and T_2. In this work, T_2 is defined using phrase-structure syntactic annotations from OntoNotes 5.0 (Weischedel et al., 2013; Pradhan et al., 2013). We experiment with three settings: one where the corpus for T_2 does not overlap with the training datasets for T_1 (frame-SRL) and two where there is a complete overlap (PropBank SRL and coreference). Compared to approaches which require multiple output labels over the same data, we offer the major advantage of not requiring any assumptions about, or specification of, the relationship between T_1 and T_2 output.

3 Related Work

We briefly contrast the syntactic scaffold with existing alternatives.

Pipelines. In a typical pipeline, T_1 and T_2 are separately trained, with the output of T_2 used to define the inputs to T_1 (Wolpert, 1992). Using syntax as T_2 in a pipeline is perhaps the most

---

[1] This excludes models initialized with deep, contextualized embeddings (Peters et al., 2018), an approach orthogonal to ours.

common approach for semantic structure prediction (Toutanova et al., 2008; Yang and Mitchell, 2017; Wiseman et al., 2016).[2] However, pipelines introduce the problem of cascading errors (T_2's mistakes affect the performance, and perhaps the training, of T_1; He et al., 2013). To date, remedies to cascading errors are so computationally expensive as to be impractical (e.g., Finkel et al., 2006). A syntactic scaffold is quite different from a pipeline since the output of T_2 is never explicitly used.

Latent variables. Another solution is to treat the output of T_2 as a (perhaps structured) latent variable. This approach obviates the need of supervision for T_2 and requires marginalization (or some approximation to it) in order to reason about the outputs of T_1. Syntax as a latent variable for semantics was explored by Zettlemoyer and Collins (2005) and Naradowsky et al. (2012). Apart from avoiding marginalization, the syntactic scaffold offers a way to use auxiliary syntactically-annotated data as direct supervision for T_2, and it need not overlap the T_1 training data.

Joint learning of syntax and semantics. The motivation behind joint learning of syntactic and semantic representations is that any one task is helpful in predicting the other (Lluís and Màrquez, 2008; Lluís et al., 2013; Henderson et al., 2013; Swayamdipta et al., 2016). This typically requires joint prediction of the outputs of T_1 and T_2, which tends to be computationally expensive at both training and test time.

Part of speech scaffolds. Similar to our work, there have been multitask models that use part-of-speech tagging as T_2, with transition-based dependency parsing (Zhang and Weiss, 2016) and CCG supertagging (Søgaard and Goldberg, 2016) as T_1. Both of the above approaches assumed parallel input data and used both tasks as supervision. Notably, we simplify our T_2, throwing away the structured aspects of syntactic parsing, whereas part-of-speech tagging has very little structure to begin with. While their approach results in improved token-level representations learned via supervision from POS tags, these must still be composed to obtain span representations. Instead, our approach learns span-level representations from phrase-type supervision directly, for semantic tasks. Additionally, these methods explore architectural variations in RNN layers for including supervision, whereas we focus on incorporating supervision with minimal changes to the baseline architecture. To the best of our knowledge, such simplified syntactic scaffolds have not been tried before.

Word embeddings. Our definition of a scaffold task *almost* includes stand-alone methods for estimating word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Peters et al., 2018). After training word embeddings, the tasks implied by models like the skip-gram or ELMo's language model become irrelevant to the downstream use of the embeddings. A noteworthy difference is that, rather than pre-training, a scaffold is integrated directly into the training of T_1 through a multitask objective.

Multitask learning. Neural architectures have often yielded performance gains when trained for multiple tasks together (Collobert et al., 2011; Luong et al., 2015; Chen et al., 2017; Hashimoto et al., 2017). In particular, performance of semantic role labeling tasks improves when done jointly with other semantic tasks (FitzGerald et al., 2015; Peng et al., 2017, 2018). Contemporaneously with this work, Hershcovich et al. (2018) proposed a multitask learning setting for universal syntactic dependencies and UCCA semantics (Abend and Rappoport, 2013). Syntactic scaffolds focus on a primary semantic task, treating syntax as an auxiliary, eventually forgettable prediction task.

4 Syntactic Scaffold Model

We assume two sources of supervision: a corpus $\mathcal{D}_1$ with instances $\mathbf{x}$ annotated for the primary task's outputs $\mathbf{y}$ (semantic role labeling or coreference resolution), and a treebank $\mathcal{D}_2$ with sentences $\mathbf{x}$, each with a phrase-structure tree $\mathbf{z}$.

4.1 Loss

Each task has an associated loss, and we seek to minimize the combination of task losses,

$$\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_1} \mathcal{L}_1(\mathbf{x},\mathbf{y}) + \delta \sum_{(\mathbf{x},\mathbf{z})\in\mathcal{D}_2} \mathcal{L}_2(\mathbf{x},\mathbf{z}) \tag{1}$$

with respect to parameters, which are partially shared, where δ is a tunable hyperparameter. In

---

[2] There has been some recent work on SRL which completely forgoes syntactic processing (Zhou and Xu, 2015), however it has been shown that incorporating syntactic information still remains useful (He et al., 2017).

the rest of this section, we describe the scaffold task. We define the primary tasks in Sections 5–6.

Each input is a sequence of tokens, $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$, for some n. We refer to a **span** of contiguous tokens in the sentence as $\mathbf{x}_{i:j} = \langle x_i, x_{i+1}, \ldots, x_j \rangle$, for any $1 \leqslant i \leqslant j \leqslant n$. In our experiments we consider only spans up to a maximum length D, resulting in $O(nD)$ spans.

Supervision comes from a phrase-syntactic tree $\mathbf{z}$ for the sentence, comprising a syntactic category $z_{i:j} \in C$ for every span $\mathbf{x}_{i:j}$ in $\mathbf{x}$ (many spans are given a NULL label). We experiment with different sets of labels C (§4.2).

In our model, every span $\mathbf{x}_{i:j}$ is represented by an embedding vector $\mathbf{v}_{i:j}$ (see details in §5.3). A distribution over the category assigned to $z_{i:j}$ is derived from $\mathbf{v}_{i:j}$:

$$p(z_{i:j} = c \mid \mathbf{x}_{i:j}) = \operatorname*{softmax}_{c} \mathbf{w}_c \cdot \mathbf{v}_{i:j} \qquad (2)$$

where $\mathbf{w}_c$ is a parameter vector associated with category c. We sum the log loss terms for all the spans in a sentence to give its loss:

$$\mathcal{L}_2(\mathbf{x}, \mathbf{z}) = - \sum_{\substack{1 \leqslant i \leqslant j \leqslant n \\ j-i \leqslant D}} \log p(z_{i:j} \mid \mathbf{x}_{i:j}). \qquad (3)$$

4.2 Labels for the Syntactic Scaffold Task

Different kinds of syntactic labels can be used for learning syntactically-aware span representations:

- **Constituent identity**: $C = \{0, 1\}$; is a span a constituent, or not?
- **Non-terminal**: c is the category of a span, including a NULL for non-constituents.
- **Non-terminal and parent**: c is the category of a span, concatenated with the category of its immediate ancestor. NULL is used for non-constituents, and for empty ancestors.
- **Common non-terminals**: Since a majority of semantic arguments and entity mentions are labeled with a small number of syntactic categories,[3] we experiment with a three-way classification among (i) noun phrase (or prepositional phrase, for frame SRL); (ii) any other category; and (iii) NULL.

In Figure 1, for the span "*encouraging them*", the constituent identity scaffold label is 1, the non-terminal label is `S|VP`, the non-terminal and parent label is `S|VP+par=PP`, and the common non-terminals label is set to `OTHER`.

5 Semantic Role Labeling

We contribute a new SRL model which contributes a strong baseline for experiments with syntactic scaffolds. The performance of this baseline itself is competitive with state-of-the-art methods (§7).

FrameNet. In the FrameNet lexicon (Baker et al., 1998), a *frame* represents a type of event, situation, or relationship, and is associated with a set of semantic roles, called *frame elements*. A frame can be evoked by a word or phrase in a sentence, called a *target*. Each frame element of an evoked frame can then be realized in the sentence as a sentential span, called an *argument* (or it can be unrealized). Arguments for a given frame do not overlap.

PropBank. PropBank similarly disambiguates predicates and identifies argument spans. Targets are disambiguated to lexically specific *senses* rather than shared frames, and a set of generic roles is used for all targets, reducing the argument label space by a factor of 17. Most importantly, the arguments were annotated on top of syntactic constituents, directly coupling syntax and semantics. A detailed example for both formalisms is provided in Figure 1.

Semantic structure prediction is the task of identifying targets, labeling their frames or senses, and labeling all their argument spans in a sentence. Here we assume gold targets and frames, and consider only the SRL task.

Formally, a single input instance for argument identification consists of: an n-word sentence $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$, a single target span $t = \langle t_{\text{start}}, t_{\text{end}} \rangle$, and its evoked frame, or sense, f. The argument labeling task is to produce a **segmentation** of the sentence: $\mathbf{s} = \langle s_1, s_2, \ldots, s_m \rangle$ for each input $\mathbf{x}$. A **segment** $s = \langle i, j, y_{i:j} \rangle$ corresponds to a labeled span of the sentence, where the label $y_{i:j} \in \mathcal{Y}_f \cup \{\text{NULL}\}$ is either a role that the span fills, or NULL if the span does not fill any role. In the case of PropBank, $\mathcal{Y}_f$ consists of all possible roles. The segmentation is constrained so that argument spans cover the sentence and do not overlap ($i_{k+1} = 1 + j_k$ for s_k; $i_1 = 1$; $j_m = n$). Segments of length 1 such that $i = j$ are allowed. A separate segmentation is predicted for each target annotation in a sentence.

---

[3] In the OntoNotes corpus, which includes both syntactic and semantic annotations, 44% of semantic arguments are noun phrases and 13% are prepositional phrases.

5.1 Semi-Markov CRF

In order to model the non-overlapping arguments of a given target, we use a semi-Markov conditional random field (**semi-CRF**; Sarawagi et al., 2004). Semi-CRFs define a conditional distribution over labeled segmentations of an input sequence, and are globally normalized. A single target's arguments can be neatly encoded as a labeled segmentation by giving the spans in between arguments a reserved NULL label. Semi-Markov models are more powerful than BIO tagging schemes, which have been used successfully for PropBank SRL (Collobert et al., 2011; Zhou and Xu, 2015, *inter alia*), because the semi-Markov assumption allows scoring variable-length segments, rather than fixed-length label n-grams as under an $(n-1)$-order Markov assumption. Computing the marginal likelihood with a semi-CRF can be done using dynamic programming in $O(n^2)$ time (§5.2). By filtering out segments longer than D tokens, this is reduced to $O(nD)$.

Given an input $\mathbf{x}$, a semi-CRF defines a conditional distribution $p(\mathbf{s} \mid \mathbf{x})$. Every segment $s = \langle i, j, y_{i:j}\rangle$ is given a real-valued score, $\psi(\langle i, j, y_{i:j} = r\rangle, \mathbf{x}_{i:j}) = \mathbf{w}_r \cdot \mathbf{v}_{i:j}$, where $\mathbf{v}_{i:j}$ is an embedding of the span (§5.3) and $\mathbf{w}_r$ is a parameter vector corresponding to its label. The score of the entire segmentation $\mathbf{s}$ is the sum of the scores of its segments: $\Psi(\mathbf{x}, \mathbf{s}) = \sum_{k=1}^{m} \psi(s_k, \mathbf{x}_{i_k:j_k})$. These scores are exponentiated and normalized to define the probability distribution. The sum-product variant of the semi-Markov dynamic programming algorithm is used to calculate the normalization term (required during learning). At test time, the max-product variant returns the most probable segmentation, $\hat{\mathbf{s}} = \arg\max_{\mathbf{s}} \Psi(\mathbf{s}, \mathbf{x})$.

The parameters of the semi-CRF are learned to maximize a criterion related to the conditional log-likelihood of the gold-standard segments in the training corpus (§5.2). The learner evaluates and adjusts segment scores $\psi(s_k, \mathbf{x})$ for every span in the sentence, which in turn involves learning embedded representations for all spans (§5.3).

5.2 Softmax-Margin Objective

Typically CRF and semi-CRF models are trained to maximize a conditional log-likelihood objective. In early experiments, we found that incorporating a structured cost was beneficial; we do so by using a softmax-margin training objective (Gimpel and Smith, 2010), a "cost-aware" variant of log-likelihood:

$$\mathcal{L}_1 = -\sum_{(\mathbf{x},\mathbf{s}^*)\in\mathcal{D}_1} \log \frac{\exp \Psi(\mathbf{s}^*, \mathbf{x})}{Z(\mathbf{x}, \mathbf{s}^*)}, \tag{4}$$

$$Z(\mathbf{x}, \mathbf{s}^*) = \sum_{\mathbf{s}} \exp\{\Psi(\mathbf{s}, \mathbf{x}) + \mathrm{cost}(\mathbf{s}, \mathbf{s}^*)\}. \tag{5}$$

We design the cost function so that it factors by predicted span, in the same way Ψ does:

$$\mathrm{cost}(\mathbf{s}, \mathbf{s}^*) = \sum_{s\in\mathbf{s}} \mathrm{cost}(s, \mathbf{s}^*) = \sum_{s\in\mathbf{s}} \mathbb{I}(s \notin \mathbf{s}^*). \tag{6}$$

The softmax-margin criterion, like log-likelihood, is globally normalized over all of the exponentially many possible labeled segmentations. The following zeroth-order semi-Markov dynamic program (Sarawagi et al., 2004) efficiently computes the new partition function:

$$\alpha_j = \sum_{\substack{s=\langle i,j,y_{i:j}\rangle\\ j-i\leqslant D}} \alpha_{i-1} \exp\{\Psi(s, \mathbf{x}) + \mathrm{cost}(s, \mathbf{s}^*)\}, \tag{7}$$

where $Z = \alpha_n$, under the base case $\alpha_0 = 1$.

The prediction under the model can be calculated using a similar dynamic program with the following recurrence where $\gamma_0 = 1$:

$$\gamma_j = \max_{\substack{s=\langle i,j,y_{i:j}\rangle\\ j-i\leqslant D}} \gamma_{i-1} \exp \Psi(s, \mathbf{x}). \tag{8}$$

Our model formulation enforces that arguments do not overlap. We do not enforce any other SRL constraints, such as non-repetition of core frame elements (Das et al., 2012).

5.3 Input Span Representation

This section describes the neural architecture used to obtain the span embedding, $\mathbf{v}_{i:j}$, corresponding to a span $\mathbf{x}_{i:j}$ and the target in consideration, $t = \langle t_{\text{start}}, t_{\text{end}}\rangle$. For the scaffold task, since the syntactic treebank does not contain annotations for semantic targets, we use the last verb in the sentence as a placeholder target, wherever target features are used. If there are no verbs, we use the first token in the sentence as a placeholder target. The parameters used to learn $\mathbf{v}$ are shared between the tasks.

We construct an embedding for the span using

- $\mathbf{h}_i$ and $\mathbf{h}_j$: contextualized embeddings for the words at the span boundary (§5.3.1),
- $\mathbf{u}_{i:j}$: a span summary that pools over the contents of the span (§5.3.2), and

- $\mathbf{a}_{i:j}$: and a hand-engineered feature vector for the span (§5.3.3).

This embedding is then passed to a feedforward layer to compute the span representation, $\mathbf{v}_{i:j}$.

5.3.1 Contextualized Token Embeddings

To obtain contextualized embeddings of each token in the input sequence, we run a bidirectional LSTM (Graves, 2012) with ℓ layers over the full input sequence. To indicate which token is a predicate, a linearly transformed one-hot embedding $\mathbf{v}$ is used, following Zhou and Xu (2015) and He et al. (2017). The input vector representing the token at position q in the sentence is the concatenation of a fixed pretrained embedding $\mathbf{x}_q$ and $\mathbf{v}_q$. When given as input to the bidirectional LSTM, this yields a hidden state vector $\mathbf{h}_q$ representing the qth token in the context of the sentence.

5.3.2 Span Summary

Tokens within a span might convey different amounts of information necessary to label the span as a semantic argument. Following Lee et al. (2017), we use an attention mechanism (Bahdanau et al., 2014) to summarize each span. Each contextualized token in the span is passed through a feed-forward network to obtain a weight, normalized to give $\sigma_k = \operatorname{softmax}_{i \leqslant k \leqslant j} \mathbf{w}^{\text{head}} \cdot \mathbf{h}_k$, where $\mathbf{w}^{\text{head}}$ is a learned parameter. The weights σ are then used to obtain a vector that summarizes the span, $\mathbf{u}_{i:j} = \sum_{i \leqslant k \leqslant j; j-i<D} \sigma_k \cdot \mathbf{h}_k$.

5.3.3 Span Features

We use the following three features for each span:

- width of the span in tokens (Das et al., 2014)
- distance (in tokens) of the span from the target (Täckström et al., 2015)
- position of the span with respect to the target (*before*, *after*, *overlap*) (Täckström et al., 2015)

Each of these features is encoded as a one-hot-embedding and then linearly transformed to yield a feature vector, $\mathbf{a}_{i:j}$.

6 Coreference Resolution

Coreference resolution is the task of determining clusters of mentions that refer to the same entity. Formally, the input is a document $\mathbf{x} = x_1, x_2, \ldots, x_n$ consisting of n words. The goal is to predict a set of clusters $\mathbf{c} = \{c_1, c_2, \ldots\}$, where each cluster $c = \{s_1, s_2, \ldots\}$ is a set of spans and each span $s = \langle i, j \rangle$ is a pair of indices such that $1 \leqslant i \leqslant j \leqslant n$.

As a baseline, we use the model of Lee et al. (2017), which we describe briefly in this section. This model decomposes the prediction of coreference clusters into a series of span classification decisions. Every span s predicts an antecedent $w_s \in \mathcal{Y}(s) = \{\textsc{null}, s_1, s_2, \ldots, s_m\}$. Labels s_1 to s_m indicate a coreference link between s and one of the m spans that precede it, and NULL indicates that s does not link to anything, either because it is not a mention or it is in a singleton cluster. The predicted clustering of the spans can be recovered by aggregating the predicted links.

Analogous to the SRL model (§5), every span s is represented by an embedding $\mathbf{v}_s$, which is central to the model. For each span s and a potential antecedent $a \in \mathcal{Y}(s)$, pairwise coreference scores $\Psi(\mathbf{v}_s, \mathbf{v}_a, \phi(s, a))$ are computed via feedforward networks with the span embeddings as input. $\phi(s, a)$ are pairwise discrete features encoding the distance between span s and span a and metadata, such as the genre and speaker information. We refer the reader to Lee et al. (2017) for the details of the scoring function.

The scores from Ψ are normalized over the possible antecedents $\mathcal{Y}(s)$ of each span to induce a probability distribution for every span:

$$p(w_s = a) = \operatorname*{softmax}_{a \in \mathcal{Y}(s)} \Psi(\mathbf{v}_s, \mathbf{v}_a, \phi(s, a)) \tag{9}$$

In learning, we minimize the negative log-likelihood marginalized over the possibly correct antecedents:

$$\mathcal{L}_1 = -\sum_{s \in \mathcal{D}} \log \sum_{a^* \in G(s) \cap \mathcal{Y}(s)} p(w_s = a^*) \tag{10}$$

where $\mathcal{D}$ is the set of spans in the training dataset, and $G(s)$ indicates the gold cluster of s if it belongs to one and $\{\textsc{null}\}$ otherwise.

To operate under reasonable computational requirements, inference under this model requires a two-stage beam search, which reduces the number of span pairs considered. We refer the reader to Lee et al. (2017) for details.

Input span representation. The input span embedding, $\mathbf{v}_s$ for coreference resolution and its syntactic scaffold follow the definition used in §5.3, with the key difference of using no target features. Since there is a complete overlap of input sentences between $\mathcal{D}_{\text{sc}}$ and $\mathcal{D}_{\text{pr}}$ as the coreference annotations are also from OntoNotes (Pradhan et al., 2012), we reuse the **v** for the scaffold task. Additionally, instead of the entire document, each sentence in it is independently given as input to the bidirectional LSTMs.

7 Results

We evaluate our models on the test set of FrameNet 1.5 for frame SRL and on the test set of OntoNotes for both PropBank SRL and coreference. For the syntactic scaffold in each case, we use syntactic annotations from OntoNotes 5.0 (Weischedel et al., 2013; Pradhan et al., 2013).[4] Further details on experimental settings and datasets have been elaborated in the supplemental material.

Frame SRL. Table 1 shows the performance of all the scaffold models on frame SRL with respect to prior work and a semi-CRF baseline (§5.1) without a syntactic scaffold. We follow the official evaluation from the SemEval shared task for frame-semantic parsing (Baker et al., 2007).

Prior work for frame SRL has relied on predicted syntactic trees, in two different ways: by using syntax-based rules to prune out spans of text that are unlikely to contain any frame's argument; and by using syntactic features in their statistical model (Das et al., 2014; Täckström et al., 2015; FitzGerald et al., 2015; Kshirsagar et al., 2015).

The best published results on FrameNet 1.5 are due to Yang and Mitchell (2017). In their sequential model (SEQ), they treat argument identification as a sequence-labeling problem using a deep bidirectional LSTM with a CRF layer. In their relational model (REL), they treat the same problem as a span classification problem. Finally, they introduce an ensemble to integerate both models, and use an integer linear program for inference satisfying SRL constraints. Though their model does not do any syntactic pruning, it does use syntactic features for argument identification and labeling.[5]

Notably, all prior systems for frame SRL listed in Table 1 use a pipeline of syntax and semantics. Our semi-CRF baseline outperforms all prior work, without any syntax. This highlights the benefits of modeling spans and of global normalization.

Turning to scaffolds, even the most coarse-grained constituent identity scaffold improves the performance of our syntax-agnostic baseline. The nonterminal and nonterminal and parent scaffolds, which use more detailed syntactic representations, improve over this. The greatest improvements come from the scaffold model predicting common nonterminal labels (`NP` and `PP`, which are the most common syntactic categories of semantic arguments, vs. others): 3.6% absolute improvement in F_1 measure over prior work.

Contemporaneously with this work, Peng et al. (2018) proposed a system for joint frame-semantic and semantic dependency parsing. They report results for joint frame and argument identification, and hence cannot be directly compared in Table 1. We evaluated their output for argument identification only; our semi-CRF baseline model exceeds their performance by 1 F_1, and our common nonterminal scaffold by 3.1 F_1.[6]

Model	**Prec.**	**Rec.**	F_1
Kshirsagar et al. (2015)	66.0	60.4	63.1
Yang and Mitchell (2017) (REL)	71.8	57.7	64.0
Yang and Mitchell (2017) (SEQ)	63.4	66.4	64.9
†Yang and Mitchell (2017) (ALL)	70.2	60.2	65.5
Semi-CRF baseline	67.8	66.2	67.0
+ constituent identity	68.1	67.4	67.7
+ nonterminal and parent	68.8	68.2	68.5
+ nonterminal	69.4	68.0	68.7
+ common nonterminals	69.2	69.0	**69.1**

Table 1: Frame SRL results on the test set of FrameNet 1.5., using gold frames. Ensembles are denoted by †.

Model	**Prec.**	**Rec.**	F_1
Zhou and Xu (2015)	-	-	81.3
He et al. (2017)	81.7	81.6	81.7
He et al. (2018a)	83.9	73.7	82.1
Tan et al. (2018)	81.9	83.6	82.7
Semi-CRF baseline	84.8	81.2	83.0
+ common nonterminals	85.1	82.6	**83.8**

Table 2: PropBank sSRL results, using gold predicates, on CoNLL 2012 test. For fair comparison, we show only non-ensembled models.

---

[4] `http://cemantix.org/data/ontonotes.html`

[5] Yang and Mitchell (2017) also evaluated on the full frame-semantic parsing task, which includes frame-SRL as well as identifying frames. Since our frame SRL performance improves over theirs, we expect that incorporation into a full system (e.g., using their frame identification module) would lead to overall benefits as well; this experiment is left to future work.

[6] This result is not reported in Table 1 since Peng et al. (2018) used a preprocessing which renders the test set slightly larger — the difference we report is calculated using their test set.

Model	**MUC**			$\mathbf{B^3}$			$\mathbf{CEAF_{\phi_4}}$			**Avg.** $\boldsymbol{F_1}$
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	
Wiseman et al. (2016)	77.5	69.8	73.4	66.8	57.0	61.5	62.1	53.9	57.7	64.2
Clark and Manning (2016b)	79.9	69.3	74.2	71.0	56.5	63.0	63.8	54.3	58.7	65.3
Clark and Manning (2016a)	79.2	70.4	74.6	69.9	58.0	63.4	63.5	55.5	59.2	65.7
Lee et al. (2017)	78.4	73.4	75.8	68.6	61.8	65.0	62.7	59.0	60.8	67.2
+ common nonterminals	78.4	74.3	76.3	68.7	62.9	65.7	62.9	60.2	61.5	**67.8**

Table 3: Coreference resolution results on the test set on the English CoNLL-2012 shared task. The average F_1 of MUC, B^3, and $CEAF_{\phi_4}$ is the main evaluation metric. For fair comparison, we show only non-ensembled models.

PropBank SRL. We use the OntoNotes data from the CoNLL shared task in 2012 (Pradhan et al., 2013) for Propbank SRL. Table 2 reports results using gold predicates.

Recent competitive systems for PropBank SRL follow the approach of Zhou and Xu (2015), employing deep architectures, and forgoing the use of any syntax. He et al. (2017) improve on those results, and in analysis experiments, show that constraints derived using syntax may further improve performance. Tan et al. (2018) employ a similar approach but use feed-forward networks with self-attention. He et al. (2018a) use a span-based classification to jointly identify and label argument spans.

Our syntax-agnostic semi-CRF baseline model improves on prior work (excluding `ELMo`), showing again the value of global normalization in semantic structure prediction. We obtain further improvement of 0.8 absolute F_1 with the best syntactic scaffold from the frame SRL task. This indicates that a syntactic inductive bias is beneficial even when using sophisticated neural architectures.

He et al. (2018a) also provide a setup where initialization was done with deep contextualized embeddings, `ELMo` (Peters et al., 2018), resulting in 85.5 F_1 on the OntoNotes test set. The improvements from `ELMo` are methodologically orthogonal to syntactic scaffolds.

Since the datasets for learning PropBank semantics and syntactic scaffolds *completely* overlap, the performance improvement cannot be attributed to a larger training corpus (or, by extension, a larger vocabulary), though that might be a factor for frame SRL.

A syntactic scaffold can match the performance of a pipeline containing carefully extracted syntactic features for semantic prediction (Swayamdipta et al., 2017). This, along with other recent approaches (He et al., 2017, 2018b) show that syntax remains useful, even with strong neural models for SRL.

Coreference. We report the results on four standard scores from the CoNLL evaluation: MUC, B^3 and $CEAF_{\phi_4}$, and their average F_1 in Table 3. Prior competitive coreference resolution systems (Wiseman et al., 2016; Clark and Manning, 2016b,a) all incorporate synctactic information in a pipeline, using features and rules for mention proposals from predicted syntax.

Our baseline is the model from Lee et al. (2017), described in §6. Similar to the baseline model for frame SRL, and in contrast with prior work, this model does not use any syntax.

We experiment with the best syntactic scaffold from the frame SRL task. We used `NP`, `OTHER`, and NULL as the labels for the common nonterminals scaffold here, since coreferring mentions are rarely prepositional phrases. The syntactic scaffold outperforms the baseline by 0.6 absolute F_1. Contemporaneously, Lee et al. (2018) proposed a model which takes in account higher order inference and more aggressive pruning, as well as initialization with `ELMo` embeddings, resulting in 73.0 average F_1. All the above are orthogonal to our approach, and could be incorporated to yield higher gains.

8 Discussion

To investigate the performance of the syntactic scaffold, we focus on the frame SRL results, where we observed the greatest improvement with respect to a non-syntactic baseline.

We consider a breakdown of the performance by the syntactic phrase types of the arguments, provided in FrameNet[7] in Figure 2. Not surpris-

---

[7] We used FrameNet syntactic phrase annotations for analysis only, and not in our models, since they are annotated only for the gold arguments.

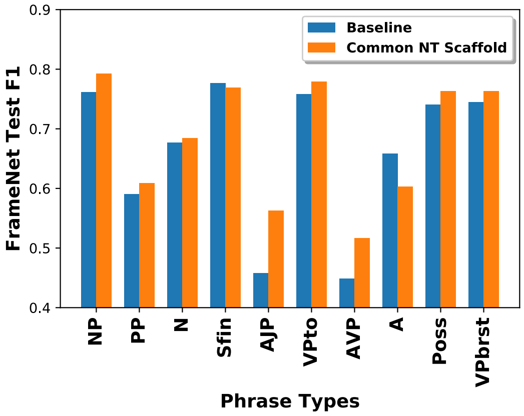

Figure 2: Performance breakdown by argument's phrase category, sorted left to right by frequency, for top ten phrase categories.

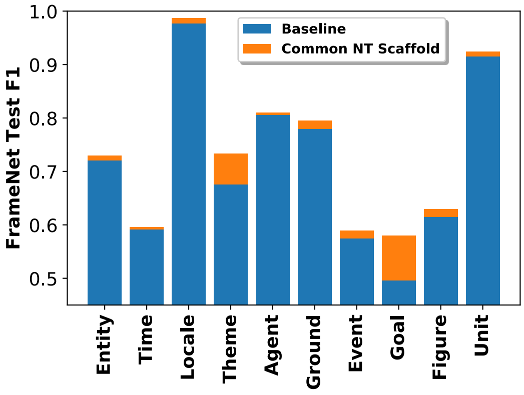

Figure 3: Performance breakdown by top ten frame element types, sorted left to right by frequency.

ingly, we observe large improvements in the common nonterminals used (`NP` and `PP`). However, the phrase type annotations in FrameNet do not correspond exactly to the OntoNotes phrase categories. For instance, FrameNet annotates non-maximal (`A`) and standard adjective phrases (`AJP`), while OntoNotes annotations for noun-phrases are flat, ignore the underlying adjective phrases. This explains why the syntax-agnostic baseline is able to recover the former while the scaffold is not.

Similarly, for frequent frame elements, scaffolding improves performance across the board, as shown in Fig. 3. The largest improvements come for `Theme` and `Goal`, which are predominantly realized as noun phrases and prepositional phrases.

9 Conclusion

We introduced syntactic scaffolds, a multitask learning approach to incorporate syntactic bias into semantic processing tasks. Unlike pipelines and approaches which jointly model syntax and semantics, no explicit syntactic processing is required at runtime. Our method improves the performance of competitive baselines for semantic role labeling on both FrameNet and PropBank, and for coreference resolution. While our focus was on span-based tasks, syntactic scaffolds could be applied in other settings (e.g., dependency and graph representations). Moreover, scaffolds need not be syntactic; we can imagine, for example, *semantic* scaffolds being used to improve NLP applications with limited annotated data. It remains an open empirical question to determine the relative merits of different kinds of scaffolds and multitask learners, and how they can be most productively combined. Our code is publicly available at `https://github.com/swabhs/scaffolding`.

Acknowledgments

We thank several members of UW-NLP, particularly Luheng He, as well as David Weiss and Emily Pitler for thoughtful discussions on prior versions of this paper. We also thank the three anonymous reviewers for their valuable feedback. This work was supported in part by NSF grant IIS-1562364 and by the NVIDIA Corporation through the donation of a Tesla GPU.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. ArXiv:1409.0473.

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval'07 Task 19: Frame semantic structure extraction. In *Proc. of SemEval*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of ACL*.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1).

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. ArXiv:1704.07556.

Kevin Clark and Christopher D Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Proc. of EMNLP*.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Proc. of ACL*.

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

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proc. of LREC*.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proc. of *SEM*.

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proc. of EMNLP*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proc. of EMNLP*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proc. of ACL*.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Proc. of NAACL*.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer.

Alex Graves. 2013. Generating sequences with recurrent neural networks. ArXiv:1308.0850.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proc. of EMNLP*.

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic feature selection for dependency parsing. In *Proc. of EMNLP*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proc. of ACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proc. of ACL*.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proc. of ACL*.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multi-lingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proc. of ACL*.

Diederik P. Kingma and Jimmy Ba. 2014. ADAM: A method for stochastic optimization. ArXiV:1412.6980.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Segmental Recurrent Neural Networks. In *Proc. of ICLR*.

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *Proc. of NAACL*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proc. of EMNLP*.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proc. of NAACL*.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *Transactions of the ACL*, 1:219–230.

Xavier Lluís and Lluís Màrquez. 2008. A joint model for parsing syntactic and semantic dependencies. In *Proc. of CoNLL*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. ArXiv:1511.06114.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. ArXiv:1301.3781.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proc. of ICML*.

Jason Naradowsky, Sebastian Riedel, and David A. Smith. 2012. Improving NLP through marginalization of hidden syntactic structure. In *Proc. of EMNLP*.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proc. of ACL*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. ArXiv:1802.05365.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proc. of CoNLL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proc. of EMNLP*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. ArXiv:1606.05250.

Sunita Sarawagi, William W Cohen, et al. 2004. Semi-markov conditional random fields for information extraction. In *Proc. of NIPS*, volume 17.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proc. of NIPS*.

Mark Steedman. 2000. Information structure and the syntax-phonology interface. *Linguistic Inquiry*, 31(4):649–689.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with Stack LSTMs. In *Proc. of CoNLL*.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. Arxiv:1706.09528.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the ACL*, 3:29–41.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proc. of AAAI*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. OntoNotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning global features for coreference resolution. In *Proc. of NAACL*.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Bishan Yang and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proc. of EMNLP*.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proc. of ACL*.

Hierarchical Quantized Representations for Script Generation

Noah Weber [1], **Leena Shekhar** [1], **Niranjan Balasubramanian**[1], **Nathanael Chambers** [2]
[1]Stony Brook University, NY, USA
{nwweber, lshekhar, niranjan}@cs.stonybrook.edu
[2]United States Naval Academy, MD, USA
nchamber@usna.edu

Abstract

Scripts define knowledge about how everyday scenarios (such as going to a restaurant) are expected to unfold. One of the challenges to learning scripts is the hierarchical nature of the knowledge. For example, a suspect arrested might plead innocent or guilty, and a very different track of events is then expected to happen. To capture this type of information, we propose an autoencoder model with a latent space defined by a hierarchy of categorical variables. We utilize a recently proposed vector quantization based approach, which allows continuous embeddings to be associated with each latent variable value. This permits the decoder to softly decide what portions of the latent hierarchy to condition on by attending over the value embeddings for a given setting. Our model effectively encodes and generates scripts, outperforming a recent language modeling-based method on several standard tasks, and allowing the autoencoder model to achieve substantially lower perplexity scores compared to the previous language modeling-based method.

1 Introduction

Scripts were originally proposed by Schank and Abelson (1977) as "structures that describe the appropriate sequence of events in a particular context". These event sequences define expectations for how common scenarios (such as going to a restaurant) should unfold, thus enabling better language understanding. Although scripts represented many other factors (roles, entry conditions, outcomes) recent work in script induction (Rudinger et al., 2015; Pichotta and Mooney, 2016; Peng and Roth, 2016) has focused on language modeling (LM) approaches where the "appropriate sequence of events" is the textual order of events (tuples of event predicates and their arguments). Modeling a distribution of text sequences gives the intuitive interpretation of *appropriate* event sequences being roughly equivalent to *probable* textual sequences. We continue with an LM approach, but we tackle two very important LM problems that have not yet been addressed with regards to event sequence modeling.

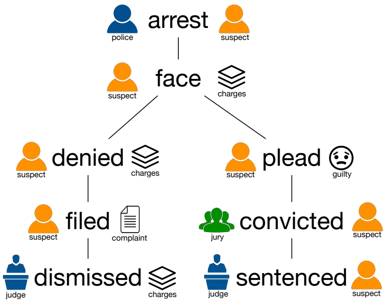

Figure 1: An automatically learned multi-track script. The left track is a dismissed case, and the right is a convicted suspect. Our model generated both tracks through a latent hierarchy.

The first problem to address is that *language models tend towards local coherency*. Count based models are restricted by window size and sparse counts, while neural language models are known to rely on the local context for predictions. Since scripts are meant to describe longer coherent scenarios, this is a major issue. For example, contradictory events like *(he denied charges)* and *(he pleads guilty)* are given high probability in a typical language model. Our model instead captures these variations with learned latent variables.

The second problem with recent work is that *the hierarchical nature of scripts is not explicitly captured*. A high level script (like a suspect getting arrested) can branch off into many possible variations. These variations are called the "tracks" of a script. Figure 1 shows a script with two tracks learned by our model. LM-based approaches often fail to explicitly capture this structure, instead

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3783–3792
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

throwing it all into one big distribution. This muddies the water for language understanding, making it difficult to tease apart differences like going to a *fancy* restaurant or a *casual* restaurant.

To remedy these problems, we propose a model that captures hierarchical structure via global latent variables. The latent variables are categorical (representing the various types of scripts and thier possible tracks and variations) and form a tree (or more generally, a DAG)[1], thus capturing hierarchical structure with the top (or bottom) levels of the tree representing high (or low) level features of the script. The top might control for large differences like restaurant vs crime, while the bottom selects between fancy and casual dining.

The overall model, which we describe below, takes the form of an autoencoder, with an encoder network inferring values of the latents and a decoder conditioned on the latents generating scripts. We show the usefulness of these latent representations against a prior RNN language model based system (Pichotta and Mooney, 2016) on several tasks. We additionally evaluate the perplexity of the system against the RNN language model, a task that autoencoder models have typically struggled with (Bowman et al., 2016). We find that the latent tree reduces model perplexity by a significant amount, possibly indicating the usefulness of the model in a more general sense.

2 Background

2.1 Variational Autoencoders

Variational Autoencoders (VAEs, Kingma and Welling (2014)) are generative models which learn latent codes z for the data x by maximizing a lower bound on the data likelihood:

$$\log(p(x)) \geq \mathrm{E}_{q(z|x)}[p(x|z)] - KL[q(z|x)||p(z)]$$

VAEs consist of two components: an encoder which parameterizes the latent posterior $q(z|x)$ and a decoder which parameterizes $p(x|z)$. The objective function can be made completely differentiable via the reparameterization trick, with the full model resembling an autoencoder and the KL term acting as a regularizer.

While VAEs have been useful in continuous domains, they have been less successful in generating discrete domains whose outputs have local syntactic regularities. Part of this is due to the “posterior collapse” problem (Bowman et al., 2016); when VAEs are equipped with powerful autoregressive decoders, they tend to ignore the latent, collapsing the posterior $q(z|x)$ to the (usually zero-mean Gaussian) prior $p(z)$. By doing this, the model takes no penalty from the KL term, but effectively ignores its encoder.

2.2 Vector Quantized Variational Autoencoders

Vector Quantized VAEs (VQ-VAEs, van den Oord et al. (2017)) are a recently proposed class of models which both alleviates the posterior collapse problem and allows the model to use a latent space of discrete values. In VQ-VAEs the latent z is represented as a categorical variable that can take on K values. Each of these values $k \in \{1, ..., K\}$ has associated with it a vector embedding e_k. The posterior of VQ-VAEs are discrete, deterministic, and parameterized as follows:

$$q(z=k|x) = \begin{cases} 1 & \text{k=argmin}_i ||f(x) - e_i||_2 \\ 0 & \text{elsewise} \end{cases}$$

where $f(x)$ is a function defined by an encoder network. The decoding portion of the network is similar to VAEs, where a decoder parameterizes a distribution $p(x|z = k) = g(e_k)$, where g is the decoder network, and e_k is the corresponding embedding, which is fed as input to the decoder. This process can be seen as a ”quantization” operation mapping the continuous encoder output to the latent embedding it falls closest to, and then feeding this latent embedding (in lieu of the encoder output) to the decoder.

The quantization operator is not differentiable, thus during training, the gradient of the loss with respect to the decoder input is used as an estimation to the gradient of the loss with respect to the encoder output. If one assumes a uniform prior over the latents (we do so here), then the KL term in the VAE objective becomes constant and may be ignored. In practice, multiple latent variables z may by used, each with their own (or shared) embeddings space.

3 Hierarchical Quantized Autoencoder

Our goal is to build a model that can generate globally coherent multi-track scripts which allow us to account for the different ways in which a script can unfold. The main idea behind our approach

---

[1] In this work we only look at linear chains of categorical variables, which is enough to encode trees (such as the one in Figure 1)

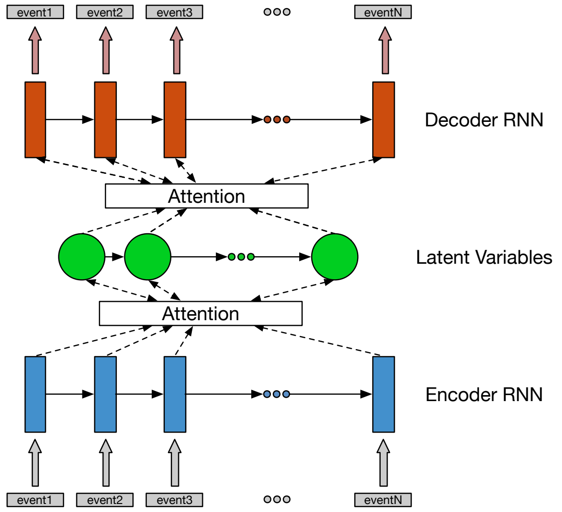

Figure 2: Hierarchical Quantized Autoencoder Architecture

is to use a hierarchical latent space which conditions the generation of the scripts. The VQ-VAE models (described earlier) provide a way to model discrete variables in an autoencoding framework while avoiding the posterior collapse problem. We build on this framework to propose a new HierarchicAl Quantized Autoencoder (HAQAE) model.

HAQAEs are autoencoders with M latent variables, $z_0, ..., z_M$. Each latent variable is categorical taking on K different values. Like in VQ-VAEs, every categorical value k for variable z has an associated embeddings e_{zk}. The latent variables are given a tree structure and the full posterior over all M latents **z** factorizes as:

$$q(\mathbf{z}|x) = q_0(z_0|x)\prod_{i=1}^{M-1} q_i(z_i|pr(z_i), x)$$

where $pr(z_i)$ denotes the parent of z_i in the tree. Since the latent variables are meant to capture the hierarchical categorization of the script, we make the assumption that when a higher level script category (for example, z_0) is observed with the actual sequence of events (x), determining the immediate lower level category (z_1) is a deterministic operation. Thus, similar to VQ-VAEs, we parameterize the individual factors of the posterior, $q_i(z_i = k|pr(z_i), x)$, as:

$$\begin{cases} 1 & \text{k=argmin}_j ||f_i(x, pr(z_i)) - e_{ij}||_2 \\ 0 & \text{elsewise} \end{cases}$$

where $f_i(x, pr(z_i))$ is an encoding function specific to latent z_i and e_{ij} is the jth value embeddings for z_i. The distribution $p(x|\mathbf{z})$ is similarly parameterized by an decoder function $g(\mathbf{z}_e)$, where $\mathbf{z}_e$ is the set of corresponding value embedding for each latent z_i. We describe the forms of the encoder and decoder in the next section.

3.1 HAQAE Encoder and Decoder

During the encoding process, certain parts of the input may provide more evidence towards different parts of the hierarchy. For example, the event *(he ate food)* gives evidence to the high level category of a *restaurant* script, while the more specific event *(he drank wine)* gives more evidence to the lower level category *fancy restaurant*. Thus during encoding, it makes sense to allow each latent to decide which parts of the input to take into consideration, based on its parent latents. This is accomplished by parameterizing the encoding function for latent z_i as an attention over the input x, with the parent of z_i (more specifically, the embedding for the parent's current value) acting as the 'query' vector. As is standard when using attention, the input sequence of events, $x = (x_1, ...x_n)$, is first encoded into a sequence of hidden states $\mathbf{h}_x = (h_1, ..., h_n)$ via a RNN encoder. The full encoding function for latent z_i can thus be written as:

$$f_i(x, pr(z_i)) = attn(\mathbf{h}_x, pr(z_i))$$

Though any attention formulation is possible, we use the bi-linear attention proposed in Luong et al. (2015) in our implementations. For the root of the latent tree (z_0), which has no parents, we use the averaged value of the encoder vectors $\mathbf{h}_x$ as the query vector for its attention.

We can define the decoder in a similar fashion. As is usually done, the distribution $p(x|\mathbf{z}_e)$ can be defined in an autoregressive manner using a RNN decoder network. Like the encoding process, different parts of the hierarchy may affect the generation of different parts of the input. We thus also allow the decoder network $g(\mathbf{z}_e)$ to be a RNN with a standard attention mechanism over the latent value embeddings, $\mathbf{z}_e$. Since the latent root z_0 is supposed to capture the highest level information about the script, we use its embedding value, (passed through an affine transformation and tanh activation) to initialize the hidden state of the decoder. Both encoder and decoder can be trained end to end using the same gradient estimation used for VQ-VAE.

3.2 Training Objective

The training objective for HAQAE is nearly the same as the VQ-VAE objective proposed in van den Oord et al. (2017). For a single training example, x_i the objective can be written as:

$$L = -\log p(x_i|\mathbf{z}) + \frac{1}{M}\sum_{j}^{M} L_j^R + \frac{1}{M}\sum_{j}^{M} L_j^C$$

where L_j^R and L_j^C are the *reconstruct* and *commit* loss for the jth latent variable. As in van den Oord et al. (2017), we let $sg(\cdot)$ stand for a *stop gradient* operator, such that any term passed to it is treated as a constant. The reconstruction loss is defined as:

$$L_j^R = ||sg(f_j(x, pr(z_j))) - e_j^*||_2^2$$

where e_j^* is the argmin value embedding for z_j (for the given input). The reconstruct loss is how the value embeddings for **z** are learned, and pushes the value embeddings to be closer to the output of the f_i. The commit loss is defined as:

$$L_j^C = \beta||f_j(x, pr(z_j)) - sg(e_j^*)||_2^2$$

which forces the encoder to push its output closer to *some* embedding, preventing a situation in which the encoder maps inputs far away from all embeddings. β is a hyperparameter that weighs the commit loss[2]. Note that the commitment loss may be propogated all the way up through the hierarchy of latent nodes. We allow the latent embeddings to receive updates only from the reconstruct and commit loss (*not* from the NLL loss).

4 Training Details

4.1 Dataset and Preprocessing

Dataset We use the New York Times Gigaword Corpus as our dataset. The corpus contains a total of around 1.8 million articles. We hold out 4000 articles from the corpus to construct our development (dev) set for hyperparameter tuning and 6000 articles for the test set. The input and output of the model is in the form of an event sequence. Each event is defined as a 4-tuple, (v, s, o, p), containing the verb, subject, object and preposition. Events without prepositions are given a null token in their preposition slot. The components of the events (the verb, subject, etc.) are all taken to be individual tokens, and can thus be treated more or less like normal text. For example, the events *(he played harp), (he touched moon)*, would be tokenized and given to the model as: *played he harp null tup touched he moon null*, where *null* is the null preposition token and *tup* is a special separation token between events.

We extract event tuples using Open Information Extraction system Ollie (Mausam et al., 2012). We then group together event tuples for 4 subsequent sentences to create a single event sequence. We also ignore tuples with common (*is, are, be, ...*) and repeating predicates. Finally we have 7123097, 19425, and 28667 event sequences for training, dev, and test dataset respectively. For all the experiments we fix the minimum and maximum sequence lengths to be 8 and 50 respectively.

4.2 HAQAE Model Details

The HAQAE model we use across all evaluations uses 5 discrete latent variables, structured in the form of a linear chain (thus no variable has more than one child or parent). Each variable can initially take on $K = 512$ values, with all latents having an embeddings dimension of 256. The encoder RNN that performs the initial encoding of the event sequence is a bidirectional, single layer RNN with GRU cell (Cho et al., 2014) with a hidden dimension of 512. The inputs to this encoder are word embeddings derived from the one-hot encodings of the tokens in the event sequence. The embeddings size is 300. We find initializing the embeddings with pretrained GloVe (Pennington et al., 2014) vectors to be useful. The decoder RNN is also a single layer RNN with GRU cells with a hidden dimension of 512 and 300 dimensional (initialized) word embeddings as inputs. For all experiments we use a vocabulary size of 50k. We train the model using Adam (Kingma and Ba, 2014) with a learning rate of 0.0005, and gradient clipping at 5.0. We find that the training converges around 1.5 epochs on our dataset. Further details can be found in our implementation[3]

4.3 Baselines

We compare the performance of our proposed model against three previous baselines and a modification of our HAQAE model that removes explicit dependencies between latents.

RNN Language Model For our first baseline system we train a RNN sequence model. This

---

[2] In our implementations we set $\beta = 0.25$

[3] `github.com/StonyBrookNLP/HAQAE`

model is 2 layered GRU cells with hidden size 512 and embedding size 300. We use Adam with a learning rate 0.001. To prevent the problem of exploding gradients, we clip the gradients at 10. We use uniform distribution [-0.1, 0.1] for random initialization and biases initialized to zero. We also use a dropout of 0.15 on the input and output embedding layers but none on the recurrent layers. We initialize the word embedding layer with pretrained Glove vectors as it improved the performance and makes the system directly comparable to HAQAE. We refer to this model as **RNNLM** in the following sections.

RNNLM + Role Embeddings We also reproduced the model from Pichotta and Mooney (2016) for comparison. This model is similar to the one above except that at each time step the model has an additional role marker input going into it. The marker guides the language model further by indicating what type of input is being currently fed to it: a subject, object, or predicate. These role embeddings are learned during training itself. Hyperparameters are exactly the same as the RNNLN except that the role embeddings have a dimension of 300. We will refer to this model as **RNNLM+Role**. We perform hyperparameter tuning of both the models using the development set. We use a vocabulary size of 50k. We trained both the baseline models for 2 epochs on the training set.

VAE We report results using a vanilla VAE model similar to the one used in Bowman et al. (2016). The encoder/decoder for the VAE baseline has the same specs as the encoder/decoder for the HAQAE model, with a latent dimension of 300. We use linear KL annealing for the first 15000 steps and 0.5 as the word dropout rate.

Hierarchyless HAQAE (NOHIER) In order to test the effect of explicitly having a hierarchy in the latent variables, we additionally train another HAQAE model with no explicit hierarchical latent space. The model still has 5 discrete latent variables like our original model, however each of the variables are independent of each other (given the input). All five variables are have an attention over the input and take the average of encoder vectors $\mathbf{h}_x$ as the query vector (as done with the latent root z_0 in the original model). We additionally designate one of the variables to be used to initialize the hidden state of the decoder. We found the same training hyperparmaters used in the training of the original model to work well here.

System	validation		test	
	NLL	PPL	NLL	PPL
RNNLM	4.52	91.84	4.51	90.92
RNNLM+Role	4.53	92.76	4.53	92.76
VAE	4.56	95.58	4.55	94.63
NOHIER	3.77	43.38	3.78	43.82
HAQAE	**3.73**	**41.68**	**3.74**	**42.10**

Table 1: Negative log-likelihood (NLL) and perplexity (PPL) measures on the validation and test set. Lower is better for both metrics.

5 Evaluation

5.1 Language Modeling: Perplexity

As our proposed models are essentially language models, it is natural to evaluate their perplexity scores, which can be viewed as an indirect measure of how well the models can identify scripts. We compute per-word perplexity and per-word negative log likelihood on the validation and test sets. We compute these values without considering the end-of-sentence (EOS) token. Table 1 gives these results. A good language model should assign low perplexity (high probability) to the validation and test sets. We observe that HAQAE achieves the minimum negative log likelihood and perplexity scores on both the validation and test sets as compared to the previous RNN-based models. The result is particularly interesting as autoencoders usually perform worse or comparable to other RNN language models in terms of perplexity (negative log likelihood) as is in the case of the vanilla VAE here; similar observations have also been made in Bowman et al. (2016).

5.2 Inverse Multiple Choice Narrative Cloze

Narrative cloze evaluations of event based language models (LMs) start with a sequence of events as input and test whether the LMs correctly predict a single held-out event. The standard narrative cloze task has various issues (Chambers, 2017). In our evaluations we opt instead for the multiple choice variant proposed in Granroth-Wilding and Clark (2016).

One of our goals is to test if our generative model can produce globally coherent scripts by evaluating their ability to generate coherent event sequences. To evaluate this we create a new *in-*

System	validation	test
RNNLM	25.30	26.30
RNNLM+Role	24.60	26.35
VAE	26.54	28.01
NOHIER	31.68	**34.00**
HAQAE	**31.80**	33.85

Table 2: Inverse narrative cloze accuracy(%) on randomly selected 2k validation and test set. Models scored on whether they assign a higher probability to legitimate event sequence over detractor event sequences. Higher is better.

verse narrative cloze task. Instead of being given an event sequence and predicting a single event to go with it, we instead are given only *one event* and the model must identify the rest of the event sequence. The model is identifying sequences of events, not single events. We chose this setup because sequences is what we ultimately want, but also because identifying single events resulted in very high scores (around 98% accuracy). This task proved to be more challenging as an evaluation.

We thus score a model based on the probability it assigns to event sequences that begin with a single input event. A legitimate event sequence should have high probability compared to an event sequence that is stitched together using two random event sequences. We create legitimate event sequences of a fixed length (six) by extracting actual event sequences observed in documents. For every legitimate event sequence, we use the first event in the sequence as a seed event. Then, we construct detractor event sequences for this seed by appending a different sequence of events (number of events being five) from a randomly chosen document. We create five such detractor sequences for every legitimate sequence. We rank the six sequences based on the probabilities assigned by the model and then evaluate the accuracy of the top ranked sequence. A random model will uniformly choose one among the six sequences and thus will score $16 = 16.60\%$ on the task. We report results averaged over 2000 sets of legitimate and detractor sequences.

Results in Table 2 show that the HAQAE is substantially better than both RNN LMs and vanilla VAE and similar to the NOHIER model, which shows the usefulness of the quantized embeddings overall as a global representation. The comparable results of the NOHIER model on this task might also indicate that explicitly modeling the hierarchical structure may not be completely necessary if ones only aim is to capture global coherence. The results on the perplexity task do indicate that overall, modeling the hierarchical structure is useful for better prediction.

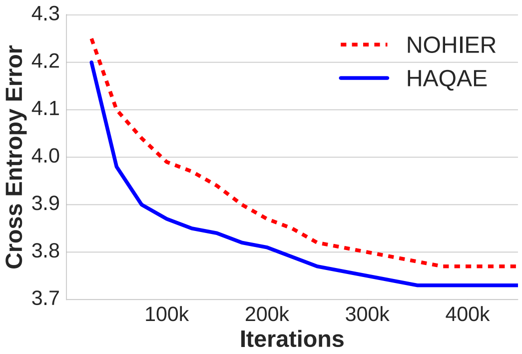

Figure 3: Cross entropy error on dev set of NOHIER (red) and HAQAE (blue) models as training progresses.

5.3 Comparing HAQAE and NOHIER

Both HAQAE and NOHIER models achieve the best results across all tasks. The HAQAE model does better on the perplexity task, while results of the two models on the cloze task are nearly the same. One clear benefit of explicitly connecting the latent variables together appears to be in the efficiency of the learning. The HAQAE model performs comparable or better than the NOHIER model despite (in this case at least) having fewer parameters[4].

The HAQAE model also appears to learn much faster than the NOHIER model. We show this in Figure 3, which shows the per-word cross entropy error on the validation set as training progresses. We observe that the cross entropy error drops much faster in the latter model than the former one. Also, the error is always lower for the HAQAE model.

One possibility is that the NOHIER model learns similar information as the HAQAE model, but due to its lack of explicit inductive bias, takes a longer time to learn this. We leave it as future work to confirm whether this is the case through an in depth study into the properties of the learned discrete latents.

---

[4] NOHIER has more parameters in our case due to each latent taking a bidirectional encoder state as a query vector, as opposed to taking the parent latent vector as the query

5.4 Evaluating Event Schemas

So far, we've evaluated how well the models recognize real textual events (perplexity) and how well the models predict events in scripts (narrative cloze). This section evaluates the script generation ability of the model, and specifically its ability to capture hierarchical information with different tracks in scripts (e.g., *pleading guilty* causes different events to occur than does *pleading innocent*). In many respects, this section illustrates best the power of HAQAE even though the results are partly subjective.

While we presented two automatic evaluations above, we shift to human judgment to evaluate the scripts themselves. We believe this complements the empirical gains already presented. The scripts generated by the models were shown to human judges and scored on several metrics.

Most previous work on script induction starts with a seed event and then grows the script based on measures of event proximity or from sampling the distribution with the seed as context. While effective in generating a bag of events, a major problem in all previous work is that conflicting events are included (sentenced *and* acquitted). While the events are related and part of the same high-level script, they should never appear together in an actual instance of a script.

In order to evaluate our model for this type of knowledge, we instead defined a seed as 2 events: the first event sets the general topic, and the second event starts a specific *track* in that topic. For instance, below are two seeds that are intended to generate two tracks for the same script:

"people reported fire"
"fire spread in neighborhood"

"people reported fire"
"fire spread to forest"

For each seed (2 events in one seed) we select the first 3 events generated by a model conditioned on the seed as context. The 2 events in a seed thus initialize the latent variable values, which then inform the decoder to generate more events (we choose the first 3). The strength of our model is that the second event helps select the more specific script track, and to ignore conflicting events in other tracks.

While generating for both RNNLM+Role and HAQAE models, we additionally enforce a constraint that restricts models from generating events that have a predicate that has already been generated, as well as events whose subject and object are the same.

We evaluated the RNNLM+Role model from Pichotta and Mooney (2016) against our proposed HAQAE. Each model was given 40 seeds (20 first event each with 2 contrasting second seed) and thus generated 40 scripts. The annotators were also shown the seeds (2 events), and then asked to rate each three-event sequence for various metrics described below.

Non-Sensical (Sense): Binary, is each event itself non-sensical or understandable?

Event Relevance (Rel): Binary, each event was scored for being relevant or not to the script topic. This ignored whether it was consistent with the seed's branch.

Coherency with Branch (Coh): 0-2, each event was scored for being coherent with the seed's specific branch (the second event). 0 means not at all, 1 means somewhat, and 2 means yes.

Branching Uniqueness (BranchU): 0-2, each pair of scripts (both branches of the same topic) were scored for overlap of events. 0 means similar events generated for both, 1 means some similar events, and 2 means distinct. This score is important because some RNN decoders might ignore the second event and focus on the general topic only.

Branching Quality (BranchQ): 0-2, each generated branch was scored for branch quality. 0 means the generated events are not specific to the branch, 1 means some are specific, and 2 means most/all events are specific. This is the most important score in measuring how well a model captures hierarchical structure and script tracks.

Two expert annotators evaluated the generated event sequences. In case of disagreements in scores, we also involved a third annotator to resolve these conflicts. Results for this task are shown in Table 3.

Both RNNLM and HAQAE produce sensical events, but the HAQAE model outperforms on all other metrics. It produces more relevant and coherent events for the topic at hand (relevance and coherency). But most important to the goals of this paper, it doubles the RNNLM scores on branching uniqueness and quality. This is because an RNNLM mostly generates from a bag of events after encoding the seed, but the HAQAE utilizes its latent space to produce branch-specific tracks of event sequences. Tables 4 show a few such ex-

System	Sense(%)	Rel(%)	Coh(0-2)	BranchU(0-2)	BranchQ(0-2)
RNNLM+Role	93.13	84.73	1.35	0.65	0.51
HAQAE	93.13	95.37	1.59	1.35	1.00

Table 3: Human evaluation of schemas generated for the seed events. HAQAE consistently performs better than the baseline model. Higher is better for all metrics.

RNN+Role	bomb found in backpack, bomb failed to detonate	bomb killed people, bomb detonated in blast
	bomb found in backpack, bomb detonated	explosion killed people, people injured in blast
	people reported fire, fire spread to forest	fire destroyed building
	people reported fire, fire spread to neighborhood	fire damaged building
HAQAE	bomb found in backpack, bomb failed to detonate	they found evidence, explosive hidden in luggage
	bomb found in backpack, bomb detonated	blast left crater, blast killed people
	people reported fire, fire spread to forest	fire burned acres
	people reported fire, fire spread to neighborhood	fire destroyed building

Table 4: Sample outputs from the baseline and our proposed system. The seeds (what is given to the system) are shown in the left column while the outputs are on the right. HAQAE is able to distinguish between the contrasting seeds. Red highlights the lack of branching quality in the baseline model and Blue highlights the correct behavior as exhibited by HAQAE.

person denied charges, lawsuit filed by person, judge dismissed lawsuit person denied charges, they accused person, person resigned in january
clinton carried promises, clinton began in 1988, clinton made changes campaign carried promises, campaign began on september, campaign made effort
campaign carried promises, campaign began on september, campaign made effort team carried for championship, team played in philadelphia, they won champoinship

Table 5: Results of changing a single latent variable while keeping others fixed. Lower level latents typically change ending/beginnings or entity names (Rows 1 and 2). The top level latent changes the topic and may occasionally preserve the form (Row 3)

amples.

5.5 Observations about the Latent Variables

We also look at how changing the values of various latent variables change the resulting output, in order to get a small idea as to what properties the variables capture. We find that the root level variable z_0 has the largest effect on the output, and typically corresponds to the domain that the sequence of events belong to. The non root variables generally change the output on a smaller scale, however we find no correspondence between the level of the variable and the amount of output that is affected upon changing its value.

One reason for the difficulty of interpreting the variables is that the model conditions on them through attention, thus changing the value of one does not necessarily need to have any effect.

We do find that changing the lower level latents generally leads to the ending/beginning of the sequence changing or the entities of the sequence changing (but still remaining in the same topical domain). We additionally find that changing the top level latent may often preserve the overall form of the event sequence, and only transform the topic. We provide examples of these output by our system in Table 5.

6 Related Work

Scripts were originally proposed by Schank and Abelson (1975) and further expanded upon in Schank and Abelson (1977). The notion of hierarchies in scripts has been studied in the works of Abbott et al. (1985) and Bower et al. (1979). Mooney and DeJong (1985) present an early non probabilistic system for extracting scripts from text. A highly related work by Miikkulainen (1990) provides an early example of a system explicitly designed to take advantage of the hierarchical nature of scripts, creating a model of scripts based on self organizing maps (Kohonen, 1982). Interestingly, self organizing maps also utilize vector quantization during learning (albeit in a different way than done here).

Recent work starting from Chambers and Jurafsky (2008) has focused on learning scripts as prototypical sequences of events using event co-occurrence. Further work has framed this task as a language modeling problem (Pichotta and Mooney, 2016; Rudinger et al., 2015; Peng and Roth, 2016). Other work has looked at learning more structured forms of script knowledge called *schemas* (Chambers, 2013; Balasubramanian et al., 2013; Nguyen et al., 2015) which focuses on additionally inducing script specific roles to be filled by entities. In this work we treat event components as separate tokens, though work has also looked into methods for composing this components into a single distributed event representation (Modi and Titov, 2014; Modi, 2016; Weber et al., 2018). We leave this as possible future work.

The hierarchical structure of our proposed model is similar to structure of the latent space in other VAE variants (Sonderby et al., 2016; Zhao et al., 2017), with the discrete variables and attentions in our model being the major differences. Hu et al. (2017) present a VAE based model for controllable text generation, with different latents controlling different aspects of the generated text, but requiring labels for semi-supervision. Other methods using discrete variables for VAEs have also been proposed (Rolfe, 2017), as have variations in the VQ-VAE learning process (Sonderby et al., 2017)

7 Conclusion

We proposed a new model, HAQAE, for script learning and generation that is one of the first to model the hierarchy that is inherent in this type of real-world knowledge. Previous work has focused on modeling event sequences with language models, while ignoring the problem of contradictory events and different tracks being jumbled together. The hierarchical latent space of HAQAE instead attends to the choice points in event sequences, and is able to provide some discrimination between tracks of events.

While HAQAE is motivated by the specific need for hierarchies in scripts, it can also be seen as a general event language model. As a language model HAQAE has a substantially lower perplexity on our test set than previous RNN models despite HAQAE's decoder having fewer parameters.

We also presented a new inverse narrative cloze task that is a multiple-choice selection of event sequences. It proved to be a very difficult task with systems producing accuracies in the mid 20% range. HAQAE and NOHEIR were the only systems to break 30 with a top accuracy of 34.0%. This further illustrates that using a latent space to capture script differences helps identify relevant sequences.

To our knowledge, all previous work on script induction has focused on learning single event sequences or bags of events. We view our proposed model as a new step toward learning different details about scripts, such as tracks and hierarchies. Though the proposed model works well empirically, understanding exactly what is learned in the latent variables is non trivial, and is a possible direction for future work.

7.1 Acknowledgements

This work is supported in part by the National Science Foundation under Grant IIS-1617969.

References

Valerie Abbott, John B Black, and Edward E Smith. 1985. The representation of scripts in memory. *Journal of Memory and Language*, 24(2):179 – 199.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *EMNLP*, pages 1721–1731.

Gordon H Bower, John B Black, and Terrence J Turner. 1979. Scripts in memory for text. *Cognitive psychology*, 11(2):177–220.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *CoNLL*.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

Nathanael Chambers. 2017. Behind the scenes of an evolving event cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 41–45.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. *Proceedings of ACL-08: HLT*, pages 789–797.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of* *the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *AAAI*, pages 2727–2733.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. *ICML*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *ICLR*.

Teuvo Kohonen. 1982. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421. The Association for Computational Linguistics.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Risto Miikkulainen. 1990. Script recognition with hierarchical feature maps. *Connection Science*, 2:83–101.

Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, Berlin, Germany.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 49–57, Ann Arbor, Michigan. Association for Computational Linguistics.

Raymond Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 681–687.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. In *ACL*, pages 188–197.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. *NIPS*.

Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Karl Pichotta and Raymond J Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*, pages 2800–2806.

Jason Tyler Rolfe. 2017. Discrete variational autoencoders. *ICLR*.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686.

Roger C Schank and Robert P Abelson. 1975. Scripts, plans, and knowledge. In *IJCAI*.

Roger C Schank and Robert P Abelson. 1977. Scripts, plans, goals and understanding: An inquiry into human knowledge structures. 1977. *Mhwah, NJ (US): Lawrence Erlbaum Associates*.

Casper Kaae Sonderby, Ben Poole, and Andriy Mnih. 2017. Continuous relaxation training of discrete latent variable image models. *NIPS Bayesian Deep Learning Workshop*.

Casper Kaae Sonderby, Tapani Raiko, Lars Maale, Sren Kaae Snderby, and Ole Winther. 2016. Ladder variational autoencoders. *NIPS*.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Event representations with tensor-based compositions. *AAAI*, abs/1711.07611.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2017. Learning hierarchical features from deep generative models. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 4091–4099. PMLR.

Semantic Role Labeling for Learner Chinese: the Importance of Syntactic Parsing and L2-L1 Parallel Data

Zi Lin[123], Yuguang Duan[3], Yuanyuan Zhao[125], Weiwei Sun[124] and Xiaojun Wan[12]
[1]Institute of Computer Science and Technology, Peking University
[2]The MOE Key Laboratory of Computational Linguistics, Peking University
[3]Department of Chinese Language and Literature, Peking University
[4]Center for Chinese Linguistics, Peking University
[5]Academy for Advanced Interdisciplinary Studies, Peking University
`{zi.lin,ariaduan,ws,wanxiaojun}@pku.edu.cn, zhaoyy1461@gmail.com`

Abstract

This paper studies semantic parsing for interlanguage (L2[1]), taking semantic role labeling (SRL) as a case task and learner Chinese as a case language. We first manually annotate the semantic roles for a set of learner texts to derive a gold standard for automatic SRL. Based on the new data, we then evaluate three off-the-shelf SRL systems, i.e., the PCFGLA-parser-based, neural-parser-based and neural-syntax-agnostic systems, to gauge how successful SRL for learner Chinese can be. We find two non-obvious facts: 1) the L1-sentence-trained systems performs rather badly on the L2 data; 2) the performance drop from the L1 data to the L2 data of the two parser-based systems is much smaller, indicating the importance of syntactic parsing in SRL for interlanguages. Finally, the paper introduces a new agreement-based model to explore the semantic coherency information in the large-scale L2-L1 parallel data. We then show such information is very effective to enhance SRL for learner texts. Our model achieves an F-score of 72.06, which is a 2.02 point improvement over the best baseline.

1 Introduction

A learner language (interlanguage) is an idiolect developed by a learner of a second or foreign language which may preserve some features of his/her first language. Previously, encouraging results of automatically building the syntactic analysis of learner languages were reported (Nagata and Sakaguchi, 2016), but it is still unknown how semantic processing performs, while parsing a learner language (L2) into semantic representations is the foundation of a variety of deeper analysis of learner languages, e.g., automatic essay scoring. In this paper, we study semantic parsing for interlanguage, taking semantic role labeling (SRL) as a case task and learner Chinese as a case language.

Before discussing a computation system, we first consider the linguistic competence and performance. Can human robustly understand learner texts? Or to be more precise, to what extent, a native speaker can understand the meaning of a sentence written by a language learner? Intuitively, the answer is towards the positive side. To validate this, we ask two senior students majoring in Applied Linguistics to carefully annotate some L2-L1 parallel sentences with predicate–argument structures according to the specification of Chinese PropBank (CPB; Xue and Palmer, 2009), which is developed for L1. A high inter-annotator agreement is achieved, suggesting the robustness of language comprehension for L2. During the course of semantic annotation, we find a non-obvious fact that we can re-use the semantic annotation specification, Chinese PropBank in our case, which is developed for L1. Only modest rules are needed to handle some tricky phenomena. This is quite different from syntactic treebanking for learner sentences, where defining a rich set of new annotation heuristics seems necessary (Ragheb and Dickinson, 2012; Nagata and Sakaguchi, 2016; Berzak et al., 2016).

Our second concern is to mimic the human's robust semantic processing ability by computer programs. The feasibility of reusing the annotation specification for L1 implies that we can reuse *standard* CPB data to train an SRL system to process learner texts. To test the *robustness* of the state-of-the-art SRL algorithms, we evaluate two types of SRL frameworks. The first one is a traditional SRL system that leverages a syntactic parser and heavy feature engineering to obtain *explicit* information of semantic roles (Feng et al., 2012).

---

[1] In this paper, we call sentences written by non-native speakers (henceforth, "L2 sentences"), aligned to their corrections by native speakers (henceforth, "L1 sentences") L2-L1 parallel sentences.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3793–3802
Brussels, Belgium, October 31 - November 4, 2018.

Furthermore, we employ two different parsers for comparison: 1) the PCFGLA-based parser, viz. Berkeley parser (Petrov et al., 2006), and 2) a minimal span-based neural parser (Stern et al., 2017). The other SRL system uses a stacked BiLSTM to implicitly capture local and non-local information (He et al., 2017). and we call it the neural syntax-agnostic system. All systems can achieve state-of-the-art performance on L1 texts but show a significant degradation on L2 texts. This highlights the weakness of applying an L1-sentence-trained system to process learner texts.

While the neural syntax-agnostic system obtains superior performance on the L1 data, the two syntax-based systems both produce better analyses on the L2 data. Furthermore, as illustrated in the comparison between different parsers, the better the parsing results we get, the better the performance on L2 we achieve. This shows that syntactic parsing is important in semantic construction for learner Chinese. The main reason, according to our analysis, is that the syntax-based system may generate correct syntactic analyses for partial grammatical fragments in L2 texts, which provides crucial information for SRL. Therefore, syntactic parsing helps build more generalizable SRL models that transfer better to new *languages*, and enhancing syntactic parsing can improve SRL to some extent.

Our last concern is to explore the potential of a large-scale set of L2-L1 parallel sentences to enhance SRL systems. We find that semantic structures of the L2-L1 parallel sentences are highly consistent. This inspires us to design a novel agreement-based model to explore such semantic coherency information. In particular, we define a metric for comparing predicate–argument structures and searching for relatively good automatic syntactic and semantic annotations to extend the training data for SRL systems. Experiments demonstrate the value of the L2-L1 parallel sentences as well as the effectiveness of our method. We achieve an F-score of 72.06, which is a 2.02 percentage point improvement over the best neural-parser-based baseline.

To the best of our knowledge, this is the first time that the L2-L1 parallel data is utilized to enhance NLP systems for learner texts.

For research purpose, we have released our SRL annotations on 600 sentence pairs and the L2-L1 parallel dataset [2].

2 Semantic Analysis of An L2-L1 Parallel Corpus

2.1 An L2-L1 Parallel Corpus

An L2-L1 parallel corpus can greatly facilitate the analysis of a learner language (Lee et al., 2017). Following Mizumoto et al. (2011), we collected a large dataset of L2-L1 parallel texts of Mandarin Chinese by exploring "language exchange" social networking services (SNS), i.e., Lang-8, a language-learning website where native speakers can freely correct the sentences written by foreign learners. The proficiency levels of the learners are diverse, but most of the learners, according to our judgment, is of intermediate or lower level.

Our initial collection consists of 1,108,907 sentence pairs from 135,754 essays. As there is lots of noise in raw sentences, we clean up the data by (1) ruling out redundant content, (2) excluding sentences containing foreign words or Chinese phonetic alphabet by checking the Unicode values, (3) dropping overly simple sentences which may not be informative, and (4) utilizing a rule-based classifier to determine whether to include the sentence into the corpus.

The final corpus consists of 717,241 learner sentences from writers of 61 different native languages, in which English and Japanese constitute the majority. As for completeness, 82.78% of the Chinese Second Language sentences on Lang-8 are corrected by native human annotators. One sentence gets corrected approximately 1.53 times on average.

In this paper, we manually annotate the predicate–argument structures for the 600 L2-L1 pairs as the basis for the semantic analysis of learner Chinese. It is from the above corpus that we carefully select 600 pairs of L2-L1 parallel sentences. We would choose the most appropriate one among multiple versions of corrections and recorrect the L1s if necessary. Because word structure is very fundamental for various NLP tasks, our annotation also contains *gold* word segmentation for both L2 and L1 sentences. Note that there are no natural word boundaries in Chinese

---

[2] The data is collected from Lang-8 (`www.lang-8.com`) and used as the training data in NLPCC 2018 Shared Task: Grammatical Error Correction (Zhao et al., 2018), which can be downloaded at `https://github.com/pkucoli/srl4il`

		P	R	F
ENG	L1	95.87	96.17	96.02
	L2	94.78	93.06	93.91
JPN	L1	97.95	98.69	98.32
	L2	96.07	97.48	96.77
RUS	L1	96.95	95.41	96.17
	L2	97.04	94.08	95.53
ARA	L1	96.95	97.76	97.35
	L2	97.12	97.56	97.34

Table 1: Inter-annotator agreement.

text. We first employ a state-of-the-art word segmentation system to produce initial segmentation results and then manually fix segmentation errors.

The dataset includes four typologically different mother tongues, i.e., English (ENG), Japanese (JPN), Russian (RUS) and Arabic (ARA). Subcorpus of each language consists of 150 sentence pairs. We take the mother languages of the learners into consideration, which have a great impact on grammatical errors and hence automatic semantic analysis. We hope that four selected mother tongues guarantee a good coverage of typologies. The annotated corpus can be used both for linguistic investigation and as test data for NLP systems.

2.2 The Annotation Process

Semantic role labeling (SRL) is the process of assigning semantic roles to constituents or their head words in a sentence according to their relationship to the predicates expressed in the sentence. Typical semantic roles can be divided into core arguments and adjuncts. The core arguments include *Agent*, *Patient*, *Source*, *Goal*, etc, while the adjuncts include *Location*, *Time*, *Manner*, *Cause*, etc.

To create a *standard* semantic-role-labeled corpus for learner Chinese, we first annotate a 50-sentence trial set for each native language. Two senior students majoring in Applied Linguistics conducted the annotation. Based on a total of 400 sentences, we adjudicate an initial gold standard, adapting and refining CPB specification as our annotation heuristics. Then the two annotators proceed to annotate a 100-sentence set for each language independently. It is on these larger sets that we report the inter-annotator agreement.

In the final stage, we also produce an adjudicated gold standard for all 600 annotated sentences. This was achieved by comparing the annotations selected by each annotator, discussing the differences, and either selecting one as fully correct or creating a hybrid representing the consensus decision for each choice point. When we felt that the decisions were not already fully guided by the existing annotation guidelines, we worked to articulate an extension to the guidelines that would support the decision.

During the annotation, the annotators apply both position labels and semantic role labels. Position labels include *S*, *B*, *I* and *E*, which are used to mark whether the word is an argument by itself, or at the beginning or in the middle or at the end of a argument. As for role labels, we mainly apply representations defined by CPB (Xue and Palmer, 2009). The predicate in a sentence was labeled as *rel*, the core semantic roles were labeled as *AN* and the adjuncts were labeled as *AM*.

2.3 Inter-annotator Agreement

For inter-annotator agreement, we evaluate the precision (P), recall (R), and F1-score (F) of the semantic labels given by the two annotators. Table 1 shows that our inter-annotator agreement is promising. All L1 texts have F-score above 95, and we take this as a reflection that our annotators are qualified. F-scores on L2 sentences are all above 90, just a little bit lower than those of L1, indicating that L2 sentences can be greatly understood by native speakers. Only modest rules are needed to handle some tricky phenomena:

1. The labeled argument should be strictly limited to the core roles defined in the frameset of CPB, though the number of arguments in L2 sentences may be more or less than the number defined.

2. For the roles in L2 that cannot be labeled as arguments under the specification of CPB, if they provide semantic information such as time, location and reason, we would labeled them as adjuncts though they may not be well-formed adjuncts due to the absence of function words.

3. For unnecessary roles in L2 caused by mistakes of verb subcategorization (see examples in Figure 3b), we would leave those roles unlabeled.

Table 2 further reports agreements on each argument (*AN*) and adjunct (*AM*) in detail, according to which the high scores are attributed to the high agreement on arguments (*AN*). The labels of *A3* and *A4* have no disagreement since they are sparse in CPB and are usually used to label specific semantic roles that have little ambiguity.

We also conducted in-depth analysis on inter-annotator disagreement. For further details, please refer to Duan et al. (2018).

		ENG	JPN	RUS	ARA
L1	*A0*	97.23	99.10	97.66	98.22
	A1	96.70	96.99	98.05	98.34
	A2	88.89	100.00	100.00	92.59
	A3	100.00	100.00	100.00	100.00
	A4	100.00	-	-	100.00
	AM	94.94	98.35	93.07	96.02
L2	*A0*	94.09	95.77	97.92	97.88
	A1	90.68	97.93	97.40	98.68
	A2	88.46	100.00	95.24	93.33
	A3	100.00	100.00	100.00	-
	A4	100.00	-	-	-
	AM	96.97	96.51	91.78	96.02

Table 2: Inter-annotator agreement (F-scores) relative to languages and role types.

3 Evaluating Robustness of SRL

3.1 Three SRL Systems

The work on SRL has included a broad spectrum of machine learning and deep learning approaches to the task. Early work showed that syntactic information is crucial for learning long-range dependencies, syntactic constituency structure and global constraints (Punyakanok et al., 2008; Täckström et al., 2015), while initial studies on neural methods achieved state-of-the-art results with little to no syntactic input (Zhou and Xu, 2015; Wang et al., 2015; Marcheggiani et al., 2017; He et al., 2017). However, the question whether fully labeled syntactic structures provide an improvement for neural SRL is still unsettled pending further investigation.

To evaluate the *robustness* of state-of-the-art SRL algorithms, we evaluate two representative SRL frameworks. One is a traditional syntax-based SRL system that leverages a syntactic parser and manually crafted features to obtain *explicit* information to find semantic roles (Gildea and Jurafsky, 2000; Xue, 2008) In particular, we employ the system introduced in Feng et al. (2012). This system first collects all c-commanders of a predicate in question from the output of a parser and puts them in order. It then employs a first order linear-chain global linear model to perform semantic tagging. For constituent parsing, we use two parsers for comparison, one is Berkeley parser[3] (Petrov et al., 2006), a well-known implementation of the unlexicalized latent variable PCFG model, the other is a minimal span-based neural parser based on independent scoring of labels and spans (Stern et al., 2017). As proposed in Stern et al. (2017), the second parser is capable of achieving state-of-the-art single-model performance on the Penn Treebank. On the Chinese TreeBank (CTB; Xue et al., 2005), it also outperforms the Berkeley parser for the in-domain test. We call the corresponding SRL systems as the **PCFGLA-parser-based** and **neural-parser-based** systems.

The second SRL framework leverages an end-to-end neural model to implicitly capture local and non-local information (Zhou and Xu, 2015; He et al., 2017). In particular, this framework treats SRL as a *BIO* tagging problem and uses a stacked BiLSTM to find informative embeddings. We apply the system introduced in He et al. (2017) for experiments. Because all syntactic information (including POS tags) is excluded, we call this system the **neural syntax-agnostic** system.

To train the three SRL systems as well as the supporting parsers, we use the CTB and CPB data [4]. In particular, the sentences selected for the CoNLL 2009 shared task are used here for parameter estimation. Note that, since the Berkeley parser is based on PCFGLA grammar, it may fail to get the syntactic outputs for some sentences, while the other parser does not have that problem. In this case, we have made sure that both parsers can parse all 1,200 sentences successfully.

3.2 Main Results

The overall performances of the three SRL systems on both L1 and L2 data (150 parallel sentences for each mother tongue) are shown in Table 3. For all systems, significant decreases on different mother languages can be consistently observed, highlighting the weakness of applying L1-sentence-trained systems to process learner texts. Comparing the two syntax-based systems with the neural syntax-agnostic system, we find that the overall ΔF, which denotes the F-score drop from L1 to L2, is smaller in the syntax-based framework

---

[3] `code.google.com/p/berkeleyparser/`

[4] Here we only use the trees that has semantic role annotations for parser training. This setup keeps us from overestimating the contribution of a parser.

		PCFGLA-parser-based SRL				Neural-parser-based SRL				Neural syntax-agnostic SRL			
		Arg.-F	Adj.-F	F	ΔF	Arg.-F	Adj.-F	F	ΔF	Arg.-F	Adj.-F	F	ΔF
ENG	L1	73.42	74.60	73.81	-4.61	72.54	76.97	74.00	-4.90	74.62	73.44	74.22	-6.65
	L2	68.05	71.63	69.20		67.12	73.24	69.10		66.50	69.75	67.57	
JPN	L1	73.88	76.80	75.10	-3.37	75.18	77.25	76.05	-3.79	76.50	78.22	77.22	-7.60
	L2	67.92	77.29	71.73		69.59	76.14	72.26		66.75	73.78	69.62	
RUS	L1	71.56	73.41	72.20	-4.28	73.20	73.72	73.39	-3.01	72.01	73.72	72.61	-5.60
	L2	68.41	66.93	67.92		68.93	73.11	70.38		65.50	69.91	67.01	
ARA	L1	73.72	64.36	70.82	-2.18	75.38	67.10	72.74	-2.80	74.10	70.44	72.92	-2.40
	L2	69.43	67.02	68.64		71.16	67.25	69.94		72.13	67.19	70.52	
ALL	L1	73.18	72.28	72.87	-3.59	74.05	73.73	73.94	-3.64	74.22	73.92	74.12	-5.41
	L2	68.52	70.77	69.28		69.20	72.39	70.30		67.99	70.08	68.71	

Table 3: Performances of the syntax-based and neural syntax-agnostic SRL systems on the L1 and L2 data. "ALL" denotes the overall performance.

than in the syntax-agnostic system. On English, Japanese and Russian L2 sentences, the syntax-based system has better performances though it sometimes works worse on the corresponding L1 sentences, indicating the syntax-based systems are more robust when handling learner texts.

Furthermore, the neural-parser-based system achieves the best overall performance on the L2 data. Though performing slightly worse than the neural syntax-agnostic one on the L1 data, it has much smaller ΔF, showing that as the syntactic analysis improves, the performances on both the L1 and L2 data grow, while the gap can be maintained. This demonstrates again the importance of syntax in semantic constructions, especially for learner texts.

3.3 Analysis

To better understand the overall results, we further look deep into the output by addressing the questions:

1. What types of error negatively impact both systems over learner texts?
2. What types of error are more problematic for the neural syntax-agnostic one over the L2 data but can be solved by the syntax-based one to some extent?

We first carry out a suite of empirical investigations by breaking down error types for more detailed evaluation. To compare two systems, we analyze results on ENG-L2 and JPN-L2 given that they reflect significant advantages of the syntax-based systems over the neural syntax-agnostic system. Note that the syntax-based system here refers to the neural-parser-based one. Finally, a concrete study on the instances in the output is conducted, as to validate conclusions in the previous step.

Operation	Description
Fix Labels (Fix)	Correct the span label if its boundary matches gold.
Move Arg. (Move)	Move a unique core argument to its correct position.
Merge Spans (Merge)	Combine two predicated spans into a gold span if they are separated by at most one word.
Split Spans (Split)	Split a predicated span into two gold spans that are separated by at most one word.
Fix Boundary (Boundary)	Correct the boundary of a span id its label matches an overlapping gold span.
Drop Arg. (Drop)	Drop a predicated argument that does not overlap with any gold span.
Add Arg. (Add)	Add a gold argument that does not overlap with any predicated span.

Table 4: Oracle transformations paired with the relative error reduction after each operation. The operations are permitted only if they do not cause any overlapping arguments

3.3.1 Breaking down Error Types

We employ 6 oracle transformations designed by He et al. (2017) to fix various prediction errors sequentially (see details in Table 4), and observe the relative improvements after each operation, as to obtain fine-grained error types. Figure 1 compares two systems in terms of different mistakes on ENG-L2 and JPN-L2 respectively. After fixing the boundaries of spans, the neural syntax-agnostic system catches up with the other, illustrating that though both systems handle boundary detection poorly on the L2 sentences, the neural syntax-agnostic one suffers more from this type of errors.

Excluding boundary errors (after moving, merg-

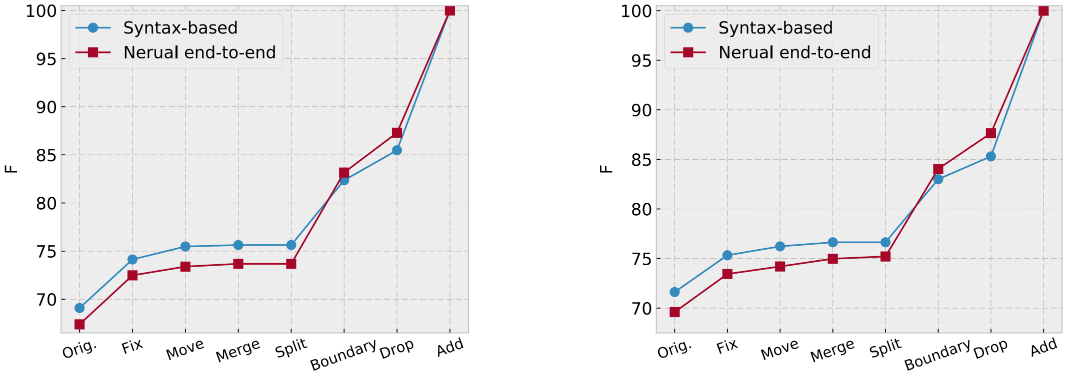

(a) ENG-L2

(b) JPN-L2

Figure 1: Relative improvements of performance after doing each type of oracle transformation in sequence over ENG-L2 and JPN-L2

ing, splitting spans and fixing boundaries), we also compare two systems on L2 in terms of detailed label identification, so as to observe which semantic role is more likely to be incorrectly labeled. Figure 2 shows the confusion matrices. Comparing (a) with (c) and (b) with (d), we can see that the syntax-based and the neural system often overly label *A1* when processing learner texts. Besides, the neural syntax-agnostic system predicts the adjunct *AM* more than necessary on L2 sentences by 54.24% compared with the syntax-based one.

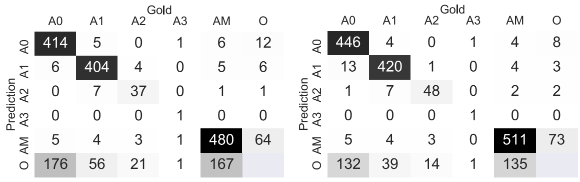

	A0	A1	A2	A3	AM	O
A0	414	5	0	1	6	12
A1	6	404	4	0	5	6
A2	0	7	37	0	1	1
A3	0	0	0	1	0	0
AM	5	4	3	1	480	64
O	176	56	21	1	167	

(a) Syntax-based system, L1

	A0	A1	A2	A3	AM	O
A0	446	4	0	1	4	8
A1	13	420	1	0	4	3
A2	1	7	48	0	2	2
A3	0	0	0	1	0	0
AM	5	4	3	0	511	73
O	132	39	14	1	135	

(b) Neural system, L1

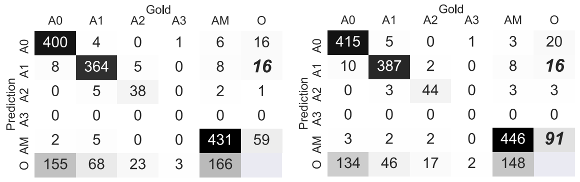

	A0	A1	A2	A3	AM	O
A0	400	4	0	1	6	16
A1	8	364	5	0	8	***16***
A2	0	5	38	0	2	1
A3	0	0	0	0	0	0
AM	2	5	0	0	431	59
O	155	68	23	3	166	

(c) Syntax-based system, L2

	A0	A1	A2	A3	AM	O
A0	415	5	0	1	3	20
A1	10	387	2	0	8	***16***
A2	0	3	44	0	3	3
A3	0	0	0	0	0	0
AM	3	2	2	0	446	***91***
O	134	46	17	2	148	

(d) Neural system, L2

Figure 2: Confusion matrix for each semantic role (here we add up matrices of ENG-L2 and JPN-L2). The predicted labels are only counted in three cases: (1) The predicated boundaries match the gold span boundaries. (2) The predicated argument does not overlap with any the gold span (Gold labeled as "O"). (3) The gold argument does not overlap with any predicted span (Prediction labeled as "O").

3.3.2 Examples for Validation

On the basis of typical error types found in the previous stage, specifically, boundary detection and incorrect labels, we further conduct an on-the-spot investigation on the output sentences.

Boundary Detection Previous work has proposed that the drop in performance of SRL systems mainly occurs in identifying argument boundaries (Màrquez et al., 2008). According to our results, this problem will be exacerbated when it comes to L2 sentences, while syntactic structure sometimes helps to address this problem.

Figure 3a is an example of an output sentence. The Chinese word "也" (also) usually serves as an adjunct but is now used for linking the parallel structure "用 汉语 也 说话 快" (using Chinese also speaking quickly) in this sentence, which is ill-formed to native speakers and negatively affects the boundary detection of *A0* for both systems.

On the other hand, the neural system incorrectly takes the whole part before "很 难" (very hard) as *A0*, regardless of the adjunct "对 我 来说" (for me), while this can be figured out by exploiting syntactic analysis, as illustrated in Figure 3c. The constituent "对 我 来说" (for me) has been recognized as a prepositional phrase (PP) attached to the VP, thus labeled as *AM*. This shows that by providing information of some well-formed sub-trees associated with correct semantic roles, the syntactic system can perform better than the neural one on SRL for learner texts.

Mistaken Labels A second common source of errors is wrong labels, especially for *A1*. Based on our quantitative analysis, as reported in Table 5,

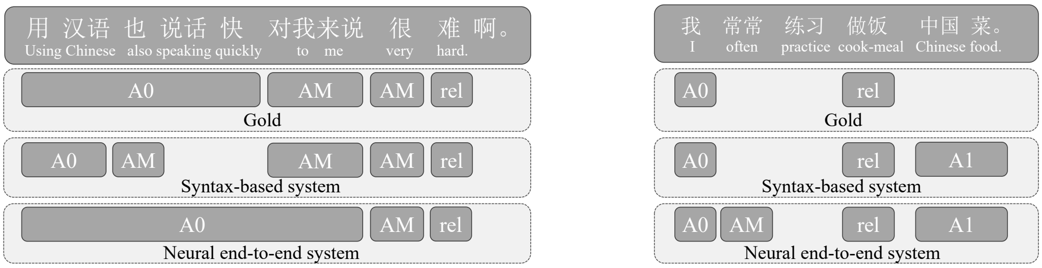

(a) SRL output of both systems for a L2 sentence, “用汉语也说话快对我来说很难” (using Chinese and also speaking quickly is very hard for me).

(b) SRL of both systems for a L2 sentence, “我常常做饭中国菜” (I often cook Chinese food).

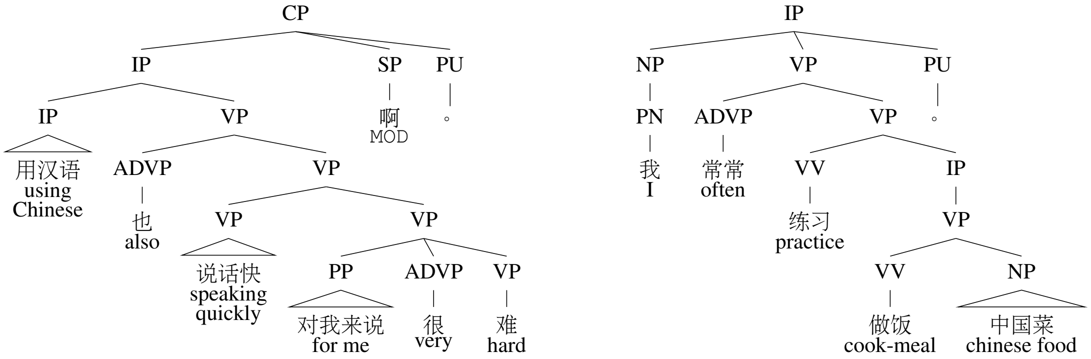

(c) Syntactic analysis for the sentence in Figure 3a

(d) Syntactic analysis for the sentence in Figure 3b

Figure 3: Two examples for SRL outputs of both systems and the corresponding syntactic analysis for the L2 sentences

these phenomena are mainly caused by mistakes of verb subcategorization, where the systems label more arguments than allowed by the predicates. Besides, the deep end-to-end system is also likely to incorrectly attach adjuncts *AM* to the predicates.

	Syntax	
Cause of error	YES	NO
Verb subcategorization	62.50%	62.50%
Labeling *A1* to punctuation	12.50%	6.25%
Word order error	6.25%	0.00%
Other types of error	18.75%	31.25%

Table 5: Causes of labeling unnecessary *A1*

Figure 3b is another example. The Chinese verb “做饭” (cook-meal) is intransitive while this sentence takes it as a transitive verb, which is very common in L2. Lacking in proper verb subcategorization, both two systems fail to recognize those verbs allowing only one argument and label the *A1* incorrectly.

As for *AM*, the neural system mistakenly adds the adjunct to the predicate, which can be avoided by syntactic information of the sentence shown in Figure 3d. The constituent “常常” (often) are adjuncts attached to VP structure governed by the verb “练习”(practice), which will not be labeled as *AM* in terms of the verb “做饭”(cook-meal). In other words, the hierarchical structure can help in argument identification and assignment by exploiting local information.

4 Enhancing SRL with L2-L1 Parallel Data

We explore the valuable information about the semantic coherency encoded in the L2-L1 parallel data to improve SRL for learner Chinese. In particular, we introduce an agreement-based model to search for high-quality automatic syntactic and semantic role annotations, and then use these annotations to retrain the two parser-based SRL systems.

4.1 The Method

For the purpose of harvesting the good automatic syntactic and semantic analysis, we consider the consistency between the automatically produced analysis of a learner sentence and its corresponding *well-formed* sentence. Determining the measurement metric for comparing predicate–argument structures, however, presents another challenge, because the words of the L2 sentence and its L1 counterpart do not necessarily match. To solve the problem, we use an automatic word aligner. BerkeleyAligner[5] (Liang et al., 2006), a state-of-the-art tool for obtaining a word alignment, is utilized.

The metric for comparing SRL results of two sentences is based on recall of $\langle w_p, w_a, r\rangle$ tuples, where w_p is a predicate, w_a is a word that is in the argument or adjunct of w_p and r is the corresponding role. Based on a word alignment, we define the shared tuple as a mutual tuple between two SRL results of an L2-L1 sentence pair, meaning that both the predicate and argument words are aligned respectively, and their role relations are the same. We then have two recall values:

- L2-recall is (# of shared tuples) / (# of tuples of the result in L2)
- L1-recall is (# of shared tuples) / (# of tuples of the result in L1)

In accordance with the above evaluation method, we select the automatic analysis of highest scoring sentences and use them to expand the training data. Sentences whose L1 and L2 recall are both greater than a threshold p are taken as good ones. A parser-based SRL system consists of two essential modules: a syntactic parser and a semantic classifier. To enhance the syntactic parser, the automatically generated syntactic trees of the sentence pairs that exhibit high semantic consistency are directly used to extend training data. To improve a semantic classifier, besides the consistent semantic analysis, we also use the outputs of the L1 but not L2 data which are generated by the neural syntax-agnostic SRL system.

4.2 Experimental Setup

Our SRL corpus contains 1200 sentences in total that can be used as an evaluation for SRL systems. We separate them into three data sets. The first data set is used as development data, which contains 50 L2-L1 sentence pairs for each language and 200 pairs in total. Hyperparameters are tuned using the development set. The second data set contains all other 400 L2 sentences, which is used as test data for L2. Similarly, all other 400 L1 sentences are used as test data for L1.

The sentence pool for extracting retraining annotations includes all English- and Japanese-native speakers' data along with its corrections. Table 6 presents the basic statistics. Around 8.5 – 11.9% of the sentence can be taken as high L1/L2 recall sentences, which serves as a reflection that argument structure is vital for language acquisition and difficult for learners to master, as proposed in Vázquez (2004) and Shin (2010). The threshold ($p = 0.9$) for selecting sentences is set upon the development data. For example, we use additional 156,520 sentences to enhance the Berkeley parser.

	ENG	JPN
#All sentence pairs	310,075	484,140
#Selected ($p = 0.9$)	36,979	41,281

Table 6: Statistics of unlabeled data.

4.3 Main Results

Table 7 summarizes the SRL results of the baseline PCFGLA-parser-based model as well as its corresponding retrained models. Since both the syntactic parser and the SRL classifier can be retrained and thus enhanced, we report the individual impact as well as the combined one. We can clearly see that when the PCFGLA parser is retrained with the SRL-consistent sentence pairs, it is able to provide better SRL-oriented syntactic analysis for the L2 sentences as well as their corrections, which are essentially L1 sentences. The outputs of the L1 sentences that are generated by the deep SRL system are also useful for improving the linear SRL classifier. A non-obvious fact is that such a retrained model yields better analysis for not only L1 but also L2 sentences. Fortunately, combining both results in further improvement.

		P	R	F
L2	Baseline	76.50	62.86	69.01
	Parser-retrained	78.74	65.87	71.73
	Classifier-retrained	77.11	64.17	70.05
	Both retrained	78.72	66.43	72.06
L1	Baseline	80.70	66.36	72.83
	Parser-retrained	82.20	68.26	74.59
	Classifier-retrained	80.98	67.57	73.67
	Both retrained	82.04	68.79	74.83

Table 7: Accuracies different PCFGLA-parser-based models on the two test data sets.

Table 8 shows the results of the parallel experiments based on the neural parser. Different from the PCFGLA model, the SRL-consistent trees only yield a slight improvement on the L2

---

[5] `code.google.com/archive/p/berkeleyaligner/`

data. On the contrary, retraining the SRL classifier is much more effective. This experiment highlights the different strengths of different frameworks for parsing. Though for *standard in-domain* test, the neural parser performs better and thus is more and more popular, for some other scenarios, the PCFGLA model is stronger.

		P	R	F
L2	Baseline	78.26	63.38	70.04
	Parser-retrained	78.19	63.78	70.25
	Classifier-retrained	78.88	64.65	71.06
	Both retrained	78.61	65.34	71.36
L1	Baseline	82.17	66.80	73.69
	Parser-retrained	81.95	66.92	73.68
	Classifier-retrained	82.08	67.69	74.20
	Both retrained	82.20	67.85	74.34

Table 8: Accuracies of different neural-parser-based models on the two test data sets.

Table 9 further shows F-scores for the baseline and the both-retrained model relative to each role type in detail. Given that the F-scores for both models are equal to 0 on *A3* and *A4*, we just omit this part. From the figure we can observe that, all the semantic roles achieve significant improvements in performances.

		A0	*A1*	*A2*	*AM*
L2	Baseline	67.95	71.21	51.43	70.20
	Both retrained	70.62	74.75	64.29	72.22
L1	Baseline	69.49	79.78	61.84	71.74
	Both retrained	73.15	80.90	63.35	73.02

Table 9: F-scores of the baseline and the both-retrained models relative to role types on the two data sets. We only list results of the PCFGLA-parser-based system.

5 Conclusion

Statistical models of annotating learner texts are making rapid progress. Although there have been some initial studies on defining annotation specification as well as corpora for syntactic analysis, there is almost no work on semantic parsing for interlanguages. This paper discusses this topic, taking Semantic Role Labeling as a case task and learner Chinese as a case language. We reveal three unknown facts that are important towards a deeper analysis of learner languages: (1) the robustness of language comprehension for interlanguage, (2) the weakness of applying L1-sentence-trained systems to process learner texts, and (3) the significance of syntactic parsing and L2-L1 parallel data in building more generalizable SRL models that transfer better to L2. We have successfully provided a better SRL-oriented syntactic parser as well as a semantic classifier for processing the L2 data by exploring L2-L1 parallel data, supported by a significant numeric improvement over a number of state-of-the-art systems. To the best of our knowledge, this is the first work that demonstrates the effectiveness of large-scale L2-L1 parallel data to enhance the NLP system for learner texts.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (61772036, 61331011) and the Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers and for their helpful comments. We also thank Nianwen Xue for useful comments on the final version. Weiwei Sun is the corresponding author.

References

Yevgeni Berzak, Jessica Kenney, Carolyn Spadine, Jing Xian Wang, Lucia Lam, Keiko Sophie Mori, Sebastian Garza, and Boris Katz. 2016. Universal dependencies for learner english. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 737–746, Berlin, Germany. Association for Computational Linguistics.

Yuguang Duan, Zi Lin, and Weiwei Sun. 2018. Is argument structure of learner chinese understandable: A corpus-based analysis. In *Proceedings of the 2018 International Conference on Bilingual Learning and Teaching (ICBLT-2018)*.

Minwei Feng, Weiwei Sun, and Hermann Ney. 2012. Semantic cohesion model for phrase-based SMT. In *Proceedings of COLING 2012*, pages 867–878, Mumbai, India. The COLING 2012 Organizing Committee.

Daniel Gildea and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 473–483.

John Lee, Keying Li, and Herman Leung. 2017. L1-l2 parallel dependency treebank as learner corpus. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 44–49, Pisa, Italy. Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL*, pages 104–111, New York City, USA. Association for Computational Linguistics.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. *arXiv preprint arXiv:1701.02593*.

Lluís Màrquez, Xavier Carreras, Kenneth C Litkowski, and Suzanne Stevenson. 2008. Semantic role labeling: an introduction to the special issue.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Ryo Nagata and Keisuke Sakaguchi. 2016. Phrase structure annotation and parsing for learner english. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1837–1847, Berlin, Germany. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Marwa Ragheb and Markus Dickinson. 2012. Defining syntax for learner language annotation. In *Proceedings of COLING 2012: Posters*, pages 965–974, Mumbai, India.

Gyu-Ho Shin. 2010. On the contribution of argument structure constructions to sentence meaning for korean learners of english. *ENGLISH TEACHING*, 65(4):209–227.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. *arXiv preprint arXiv:1705.03919*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Montserrat Martínez Vázquez. 2004. Learning argument structure generalizations in a foreign language. *VIAL, Vigo international journal of applied linguistics*, (1):151–165.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1626–1631.

Nianwen Xue. 2008. Labeling Chinese predicates with semantic roles. *Computational Linguistics*, 34:225–255.

Nianwen Xue and Martha Palmer. 2009. Adding semantic roles to the Chinese treebank. *Nat. Lang. Eng.*, 15:143–172.

Nianwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11:207–238.

Yuanyuan Zhao, Nan Jiang, Weiwei Sun, and Xiaojun Wan. 2018. Overview of the nlpcc 2018 shared task: Grammatical error correction. In *Natural Language Processing and Chinese Computing*, pages 439–445, Cham. Springer International Publishing.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1127–1137.

A Teacher-Student Framework for Maintainable Dialog Manager

Weikang Wang[1,2], **Jiajun Zhang**[1,2], **Han Zhang**[3],
Mei-Yuh Hwang[3], **Chengqing Zong**[1,2,4] and **Zhifei Li**[3]
[1] National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Mobvoi AI Lab, Seattle, US
[4] CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
`{weikang.wang, jjzhang, cqzong}@nlpr.ia.ac.cn`
`{hanzhang, mhwang, zfli}@mobvoi.com`

Abstract

Reinforcement learning (RL) is an attractive solution for task-oriented dialog systems. However, extending RL-based systems to handle new intents and slots requires a system redesign. The high maintenance cost makes it difficult to apply RL methods to practical systems on a large scale. To address this issue, we propose a practical teacher-student framework to extend RL-based dialog systems without retraining from scratch. Specifically, the "student" is an extended dialog manager based on a new ontology, and the "teacher" is existing resources used for guiding the learning process of the "student". By specifying constraints held in the new dialog manager, we transfer knowledge of the "teacher" to the "student" without additional resources. Experiments show that the performance of the extended system is comparable to the system trained from scratch. More importantly, the proposed framework makes no assumption about the unsupported intents and slots, which makes it possible to improve RL-based systems incrementally.

1 Introduction

With the flourish development of virtual personal assistants (e.g., Amazon Alexa and Google Assistant), task-oriented dialog systems, which can help users accomplish tasks naturally, have been a focal point in both academic and industry research. In the early work, the task-oriented dialog system is merely a set of hand-crafted mapping rules defined by experts. This is referred to as a rule-based system. Although rule-based systems often have acceptable performance, they are inconvenient and difficult to be optimized. Recently, reinforcement learning approaches have been applied to optimize dialog systems through interaction with a user simulator or employed real users online (Gašić et al., 2011; Su et al., 2016a; Li et al., 2016, 2017b). It has been proven that RL-based dialog systems can abandon hand-crafted dialog manager and achieve more robust performance than rule-based systems (Young et al., 2013).

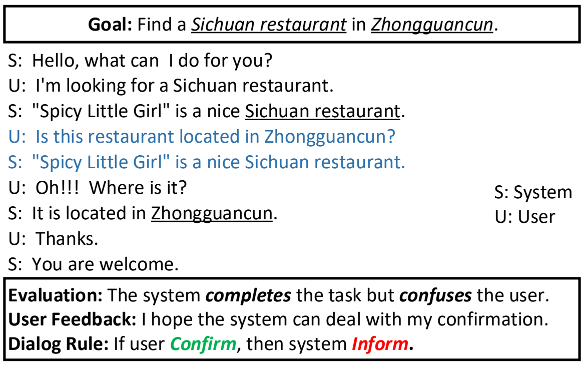
Goal: Find a *Sichuan restaurant* in *Zhongguancun*.

S: Hello, what can I do for you?
U: I'm looking for a Sichuan restaurant.
S: "Spicy Little Girl" is a nice Sichuan restaurant.
U: Is this restaurant located in Zhongguancun?
S: "Spicy Little Girl" is a nice Sichuan restaurant.
U: Oh!!! Where is it?
S: It is located in Zhongguancun.
U: Thanks.
S: You are welcome.

S: System
U: User

Evaluation: The system ***completes*** the task but ***confuses*** the user.
User Feedback: I hope the system can deal with my confirmation.
Dialog Rule: If user ***Confirm***, then system ***Inform***.

Figure 1: An example of a task-oriented dialog after the system comes online. The user is confused because the "*confirm*" intent has not been considered in the deployed system. Dialog rules should be embedded in a new system to handle such situations.

Typically, the first step of building RL-based dialog systems is defining a user model[1] and necessary system actions to complete a specific task (e.g., seek restaurants information or book hotels). Based on such ontology, developers can extract dialog features and train the dialog manager model in an interaction environment. Such systems work well if real users are consistent with the predefined user model. However, as shown in Fig. 1, the unanticipated actions[2] of real users will lead to a poor user experience.

In this situation, the original system should be extended to support new user actions based on user feedback. However, adding new intents or slots will change the predefined ontology. As a consequence, developers need to extract additional

---

[1] The user model defines what users can do in a dialog system, including domain specific intents and slots.

[2] User actions consist of intents and slots.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3803–3812
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

dialog features based on new ontology. Besides, new system actions may be added to deal with new user actions. The network architecture of the new system and the original one will be different. The new system can not inherit the parameters from the old one directly. It will make the original dialog manager model invalid. Therefore, developers have to retrain the new system by interacting with users from scratch. Though there are many methods to train a RL-based dialog manager efficiently (Su et al., 2016a, 2017; Lipton et al., 2017; Chang et al., 2017; Chen et al., 2017), the unmaintainable RL-based dialog systems will still be put on the shelf in real-world applications (Paek and Pieraccini, 2008; Paek, 2006).

To alleviate this problem, we propose a teacher-student framework to maintain the RL-based dialog manager without training from scratch. The idea is to transfer the knowledge of existing resources to a new dialog manager.

Specifically, after the system is deployed, if developers find some intents and slots missing before, they can define a few simple dialog rules to handle such situations. For example, under the condition shown in Fig. 1, a reasonable strategy is to inform the user of the location of this restaurant. Then we encode information of such hand-crafted logic rules into the new dialog manager model. Meanwhile, user logs and dialog policy of the original system can guide the new system to complete tasks like the original one. Under the guidance of the "teacher" (logic rules, user logs, and original policy), we can reforge an extended dialog manager (the "student") without a new interaction environment.

We conduct a series of experiments with simulated and real users on restaurant domain. The extensive experiments demonstrate that our method can overcome the problem brought by the unpredictable user behavior after deployment. Owing to reuse of existing resources, our framework saves time in designing new interaction environments and retraining RL-based systems from scratch. More importantly, our method does not make any assumptions about the unsupported intents and slots. So the system can be incrementally extended once developers find new intents and slots that are not taken into account before. As far as we know, we are the first to discuss the maintainability of deep reinforcement learning based dialog systems systematically.

2 Related Work

Dialog Manager The dialog manager of task-oriented dialog systems, which consists of a state tracker and a dialog policy module, controls the dialog flow. Recently, deep reinforcement learning (Mnih et al., 2013, 2015) has been applied to optimize the dialog manager in an "end-to-end" way, including *deep Q-Network* (Lipton et al., 2017; Li et al., 2017b; Peng et al., 2017; Zhao and Eskenazi, 2016) and *policy gradient* methods (Williams et al., 2017; Su et al., 2016b; Dhingra et al., 2017). RL methods have shown great potential in building a robust dialog system automatically. However, RL-based approaches are rarely used in real-world applications because of the maintainability problem (Paek and Pieraccini, 2008; Paek, 2006). To extend the domain of dialog systems, Gašic et al. (2014) explicitly defined kernel functions between the belief states that come from different domains. However, defining an appropriate kernel function is non-trivial when the ontology has changed drastically. Shah et al. (2016) proposed to integrate turn-level feedback with a task-level reward signal to learn how to handle new user intents. This approach alleviates the problem that arises from the difference between training and deployment phases. But it still fails when the developers have not considered all user actions in advance. Lipton et al. (2017) proposed to use BBQ-Networks to extend the domain. However, similar to Shah et al. (2016), the BBQ-Networks have reserved a few bits in the feature vector for new intents and slots. And system actions for handling new user actions have been considered in the original system design. This assumption is not practical enough. Compared to the existing domain extension methods, our work addresses a more practical problem: new intents and slots are unknown to the original system. If we need to extend the dialog system, we should design a new network architecture to represent new user actions and take new system actions into account.

Knowledge Distillation Our proposed framework is inspired by recent work in knowledge distillation (Bucilu et al., 2006; Ba and Caruana, 2014; Li et al., 2014). Knowledge distillation means training a compact model to mimic a larger teacher model by approximating the function learned by the teacher. Hinton et al. (2015) introduced knowledge distillation to transfer knowledge from

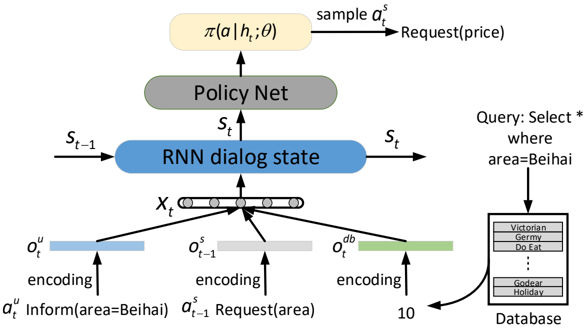

Figure 2: An overview of the RL-based dialog manager used in our work[3]. In the last turn, the system inquires "Where do you want to go?". In current turn, the user input is "Find a restaurant in Beihai.".

a large highly regularized model into a smaller model. The knowledge which can be transferred has not been restricted to models. Stewart and Ermon (2017) proposed to distill the physics and domain knowledge to train neural networks without labeled data. Hu et al. (2016) enabled a neural network to learn simultaneously from labeled instances as well as logic rules. Zhang et al. (2017) integrated multiple prior knowledge sources into neural machine translation using posterior regularization. Our experiments are based on such insights. Through defining appropriate regularization terms, we can distill different knowledge (e.g., trained model or prior knowledge) to a new designed model, alleviating the need for new labeled data or expensive interaction environments.

3 RL-based Dialog Manager

Before going to the details of our method, we provide some background on the RL-based dialog manager in this section. Fig. 2 shows an overview of such dialog manager. We describe each of the parts briefly below.

Feature Extraction At the *t-th* turn of a dialog, the user input u_t is parsed into domain specific intents and slots to form a semantic frame a_t^u by a language understanding (LU) module. o_t^u and o_{t-1}^s are the one-hot representations of such semantic frames for the current user input and the last system output respectively. Alternatively, o_t^u can be a simple n-grams representation of u_t. But the vocabulary size is relatively large in real-world applications. It will yield slow convergence in the absence of a LU module. Based on the slot-value pair output with the highest probability, a query is sent to a database to retrieve user requested information. o_t^{db} is the one-hot representation of the database result. As a result, the observable information x_t is the concatenation of o_t^u, o_{t-1}^s and o_t^{db}.

State Representation Based on the extracted feature vector x_t and previous internal state s_{t-1}, recurrent neural networks (RNNs) are used to infer the latent representation of dialog state s_t at step t. Current state s_t can be interpreted as the summary of dialog history h_t up to current step.

Dialog Policy Next, the dialog state representation s_t is fed into a policy network. The output $\pi(a|h_t;\theta)$ of the policy network is a probability distribution over a predefined system action set A_s. Lastly, the system samples an action $a_t^s \in A_s$ based on $\pi(a|h_t;\theta)$ and receives a new observation x_{t+1} with an assigned reward r_t. The policy parameters θ can be learned by maximizing the expected discounted cumulative rewards:

$$\mathcal{J}(\theta) = E\left(\sum_{k=0}^{T-t} \gamma^k r_{t+k}\right) \tag{1}$$

where T is the maximal step, and γ is the discount factor. Usually the parameters θ can be iteratively updated by *policy gradient* (Williams, 1992) approach. The policy gradient can be empirically estimated as:

$$\nabla_\theta \mathcal{J}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \, log \, \pi(a_{i,t}^s|h_{i,t};\theta)(G_{i,t} - b) \tag{2}$$

where N is the number of sampled episodes in a batch, $G_{i,t} = \sum_{k=0}^{T-t} \gamma^k r_{i,t+k}$ is the sum of discounted reward at step t in the episode i, and b is a baseline to estimate the average reward of current policy.

4 Notations and Problem Definition

Let A_u and A_s denote the supported user and system action sets in the original system design respectively. u_t denotes the user input in the *t-th* turn. The LU module converts u_t into a domain specific intent and associated slots to form a user action $a_t^u \in A_u$. The system will return an action $a_t^s \in A_s$ according to the dialog manager $\pi(\theta)$. Note that not all user actions are taken into account at the beginning of system design.

---

[3] Similar dialog model architectures can be found in recent work (Liu and Lane, 2017; Williams et al., 2017; Su et al., 2016b). But designing a dialog model architecture is not our main purpose. We focus on endowing RL-based dialog systems with maintainability and scalability. The dialog model used in our work can be replaced with the similar architectures in the related work.

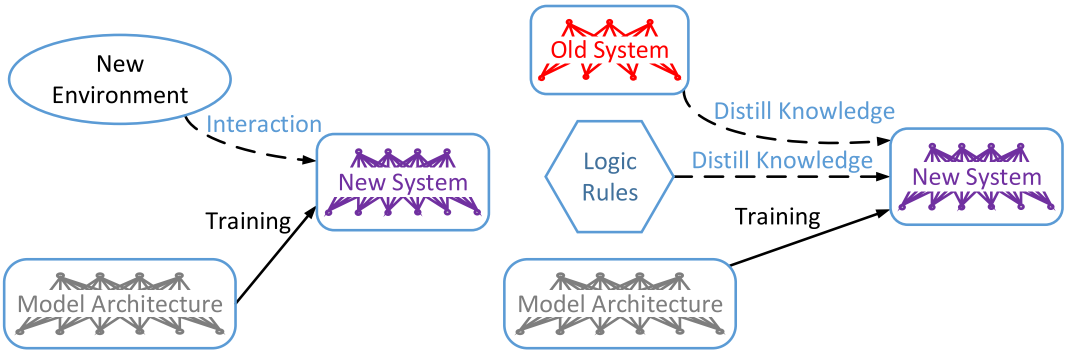

(a) Retraining a new system from scratch. (b) An overview of our proposed teacher-student framework.

Figure 3: Two kinds of strategies to extend the original system. (a) means redesigning and retraining a new system in an expensive interaction environment from scratch and (b) means transferring knowledge from existing resources to a new system. The network in red indicates the old system based on the original ontology. The networks in gray and purple represent the initialized and trained dialog manager models based on a new ontology respectively. On account of the change in ontology, the extended system has a different network architecture. The dash lines in (a) and (b) show the ability of a new model derives from various sources.

After deployment, the developers can find that some user actions A_{u_new} cannot be handled by the original system based on the human-machine interaction logs D. Generally speaking, A_{u_new} consists of new intents and slots. Our goal is to extend the original system to support the new user action set $A'_u = A_u \cup A_{u\_new}$. The extended dialog manager and new system action set are denoted as $\pi(\theta')$ and A'_s respectively. To handle new user actions, more system actions may be added to the new system. It means that A_s is a subset of A'_s.

5 Approach

Fig. 3 shows two kinds of strategies to extend the original system. The first strategy requires a new interaction environment. However, building a user simulator or hiring real users once the system needs to be extended is costly and impractical in real-world applications. By contrast, our method enhances the reuse of existing resources. The basic idea is to use the existing user logs, original dialog policy model and logic rules (“teacher”) to guide the learning process of a new dialog manager (“student”). Without an expensive interaction environment, the developers can maintain RL-based dialog systems as efficiently and straightforwardly as in rule-based systems.

5.1 Distill Knowledge from the Original System

Although the ontology of the new system is different from the original one, the extended dialog manager can still reuse dialog policy of the ill-considered system circuitously. Given user logs D and the original dialog manager $\pi(\theta)$, we define a loss $\mathcal{L}(\theta'; D, \theta)$ to minimize the difference between new dialog manager $\pi(\theta')$ and the old one:

$$\mathcal{L}(\theta'; D, \theta) = \sum_{d \in D} \sum_{t=1}^{|d|} KL(\, \pi(a|h_t; \theta) \,||\, \pi(a|h_t; \theta')\,) \quad (3)$$

where $\pi(a|h_t;\theta)$ and $\pi(a|h_t;\theta')$ are the policy distributions over A_s and A'_s given the same dialog history h_t. $|d|$ means turns of a specific dialog $d \in D$. To deal with unsupported user actions, A_s will be a subset of A'_s. As a result, the KL term in equation (3) can be defined as follows:

$$\begin{aligned} & KL(\, \pi(a|h_t; \theta) \,||\, \pi(a|h_t; \theta')\,) \\ = & \sum_{k=1}^{|A_s|} \pi(a_k|h_t;\theta) \left(log\pi(a_k|h_t;\theta) - log\pi(a_k|h_t;\theta') \right) \end{aligned} \quad (4)$$

As the original policy parameters θ are fixed, the loss function in equation (3) can be rewritten as:

$$\mathcal{L}(\theta'; D, \theta) = -\sum_{d \in D} \sum_{t=1}^{|d|} \sum_{k=1}^{|A_s|} \pi(a_k|h_t;\theta) log\pi(a_k|h_t;\theta') \quad (5)$$

This objective will transfer knowledge of the original system to the “student” at the turn level. Under the guidance of the original system, the extended system will be equipped with the primary strategy to complete a task.

5.2 Distill Knowledge from Logic Rules

It’s easy for the developers to give logic rules on the system responses to handle new user actions.

For example, if users ask to confirm a slot, the system should inform the value of that slot immediately. Note that these system actions which handle new user actions may not exist in the old model. It means the architecture of the new system is different from the old one.

We define a set of logic constraints $\mathcal{R} = \{(h_l, a_l)\}_{l=1}^{L}$, where $h_l \in H_\mathcal{R}$ indicates the dialog context condition in the l-th rule, and $a_l \in A'_s$ is the corresponding system action. The number of logic rules L is equal to the number of new user actions. These rules can be seen as triggers: if dialog context h_t in current turn t meets the context condition h_l defined in logic rules, then the system should execute a_l. In our work, we use the output of the LU module to judge whether the current dialog context meets the condition defined by logic rules. An alternative method is simple rules matching. To distill the knowledge of rules to a new system, we define a loss function $\mathcal{L}(\theta'; D, \mathcal{R})$ to embed such constraints in the new system:

$$\mathcal{L}(\theta'; D, \mathcal{R}) = -\sum_{d\in D}\sum_{t=1}^{|d|}\sum_{h_l\in H_\mathcal{R}} \mathbb{1}\{h_t = h_l\} \times \sum_{k=1}^{|A'_s|} \mathbb{1}\{a_k = a_l\}\, log\, \pi(a_k|h_t; \theta') \qquad (6)$$

Where $\mathbb{1}\{\cdot\}$ is the indicate function. Equation (6) suggests the new dialog manager $\pi(\theta')$ will be penalized if it violates the instructions defined by the dialog rules. Note that, for simplicity, we assume these rules are absolutely correct and mutually exclusive. Although this hypothesis may lead to a non-optimal dialog system, these rules define reasonable system actions to corresponding dialog contexts. It implies that the new system can be further refined by reinforcement learning once a new interaction environment is available.

5.3 Extension of Dialog Manager

In the absence of a new training environment, learning is made possible by exploiting structure that holds in the new dialog manager. On one hand, we expect the new system can complete tasks like the original one. On the other hand, it should satisfy the constraints defined by dialog rules. So, the learning objective of new dialog manager $\pi(\theta')$ can be defined as follows:

$$\mathcal{L}(\theta'; D, \theta, \mathcal{R}) = \begin{cases} \mathcal{L}(\theta'; D, \mathcal{R}) & \text{if } h_t \in H_\mathcal{R}\,; \\ \mathcal{L}(\theta'; D, \theta) & \text{else} \end{cases} \qquad (7)$$

When the dialog context h_t in the t-th turn satisfies a condition defined in $H_\mathcal{R}$, we distill knowledge of rules into the new system. Otherwise, we distill knowledge of the original system into the new one. Instead of retraining from scratch, developers can extend RL-based systems by reusing existing resources.

6 Experiments

To evaluate our method, we conduct experiments on a dialog system extension task of restaurant domain.

6.1 Domain

The dialog system provides restaurant information in Beijing. The database we use includes 2988 restaurants. This domain consists of 8 slots (*name*, *area*, *price range*, *cuisine*, *rating*, *number of comments*, *address* and *phone number*) in which the first four slots (*inform slots*) can be used for searching the desirable restaurant and all of these slots (*request slots*) can be asked by users. In each dialog, the user has a goal containing a set of slots, indicating the constraints and requests from users. For example, an *inform slot*, such as "*inform(cuisine=Sichuan cuisine)*", indicates the user finding a Sichuan restaurant, and a *request slot*, such as "*request(area)*", indicates the user asking for information from the system (Li et al., 2016, 2017b; Peng et al., 2017).

6.2 Measurements

A main advantage of our approach is that the unconsidered user actions can be handled in the extended system. In addition to traditional measurements (e.g., success rate, average turns and average reward), we define an objective measurement called "Satis." (user satisfaction) to verify this feature in the simulated evaluation. "Satis." indicates the rate at which the system takes reasonable actions in unsupported dialog situations. It can be calculated as follows:

$$Satis. = \frac{\sum_{d\in D}\sum_{t=1}^{|d|}\sum_{l=1}^{L}\mathbb{1}\{h_t = h_l\}\mathbb{1}\{a_t^s = a_l\}}{\sum_{d\in D}\sum_{t=1}^{|d|}\sum_{l=1}^{L}\mathbb{1}\{h_t = h_l\}} \qquad (8)$$

where h_t and a_t^s are the dialog history and system action in the t-th turn, h_l and a_l are dialog context condition and corresponding system action defined in the l-th rules. Intuitively, an unreasonable system reply will frustrate users and low "Satis." indicates a poor user experience.

	*Sim*1				*Sim*2			
LU Error Rate	Succ.	Turn	Reward	Satis.	Succ.	Turn	Reward	Satis.
0.00	0.962	13.6	3.94	-	0.901	13.2	2.95	0.57
0.05	0.937	13.7	3.41	-	0.877	14.4	2.41	0.48
0.10	0.910	14.3	2.65	-	0.841	13.9	1.41	0.47
0.20	0.845	15.2	0.58	-	0.784	14.7	0.01	0.47

Table 1: Performance of the original system when interacting with different user simulators. LU error means simulating slot errors and intent errors in different rates. Succ.: success rate, Turn: average turns, Reward: average reward.

LU Error Rate	0.00	0.05	0.10	0.20
Total	25857	26166	27077	28385
New User Actions	1853	1859	1998	1912

Table 2: Statistics of turns when S_1 interacts with *Sim*2.

Although “Satis.” is obtained based on our handcrafted dialog rules, it approximately measures the subjective experience of real users after system deployment.

6.3 User Simulator

Training RL-based dialog systems requires a large number of interactions with users. It’s common to use a user simulator to train RL-based dialog systems in an online fashion (Pietquin and Dutoit, 2006; Scheffler and Young, 2002; Li et al., 2016). As a consequence, we construct an agenda-based user simulator, which we refer to as *Sim*1, to train the original RL-based system. The user action set of *Sim*1 is denoted as A_u, which includes such intents[4]: “*hello*”, “*bye*”, “*inform*”, “*deny*”, “*negate*”, “*affirm*”, “*request*”, “*reqalts*” and “*null*”. The slots of *Sim*1 are shown in section 6.1. In each turn, the user action consists of a intent and slots and we append the value of slots according to the user goal.

6.4 Implementation of the Original System

For the original RL-based dialog system, a feature vector x_t of size 191 is extracted. This vector is the concatenation of encodings of LU results, the previous system reply, database results and the current turn number. The LU module is implemented with an SVM[5] for intent detection and a CRF[6] for slot filling. The language generation module is implemented by a rule-based method. The hidden dialog state representation is inferred by a GRU (Chung et al., 2014). We set the hidden states of the GRU to be 120. The policy network is implemented as a Multilayer Perceptron (MLP) with one hidden layer. The size of the hidden layer is 80. The output dimension of policy network is 15, which corresponds to the number of system actions. To encourage shorter interaction, we set a small per-turn negative reward $R_{turn} = -1$. The maximal turn is set to be 40. If the user goal is satisfied, the policy will be encouraged by a large positive reward $R_{succ} = 10$; otherwise the policy will be penalized by a negative reward $R_{fail} = -10$. Discounted factor $\gamma = 0.9$. The baseline b of current policy is estimated on sampled episodes in a batch. The batch size N is set to be 32. Adadelta (Zeiler, 2012) method is used to update model parameters. The **original system** S_1 is trained by interacting with *Sim*1. After about 2400 interactions, the performance of S_1 starts to converge.

6.5 Simulated Evaluation

To evaluate our approach, we design another user simulator, which we denote as *Sim*2, to simulate the unpredictable real customers. The user action set of *Sim*2 is denoted as A'_u. The difference between A_u and A'_u is reflected on the domain specific intents[7]. Specifically, in addition to the intents of *Sim*1, A'_u includes the “*confirm*” intent. The difference in user action sets will result in different interaction strategies between *Sim*1 and *Sim*2. To verify whether a recommended restaurant meets his (her) constraints, *Sim*1 can **only** request what the value of a specific slot is, but *Sim*2 can request or confirm.

After obtaining the original system S_1, we deploy it to interact with *Sim*1 and *Sim*2 respectively, under different LU error rates (Li et al., 2017a). In each condition, we simulate 3200 episodes to obtain the performance. Table 1 shows the

---

[4] A detail explanation of these intents is in DSTC2 (Henderson et al., 2013).

[5] We use the publicly available SVM tool at http://scikit-learn.org.

[6] We use the publicly available CRF tool at https://pypi.python.org/pypi/sklearn-crfsuite.

[7] A more complex situation is shown in the human evaluation.

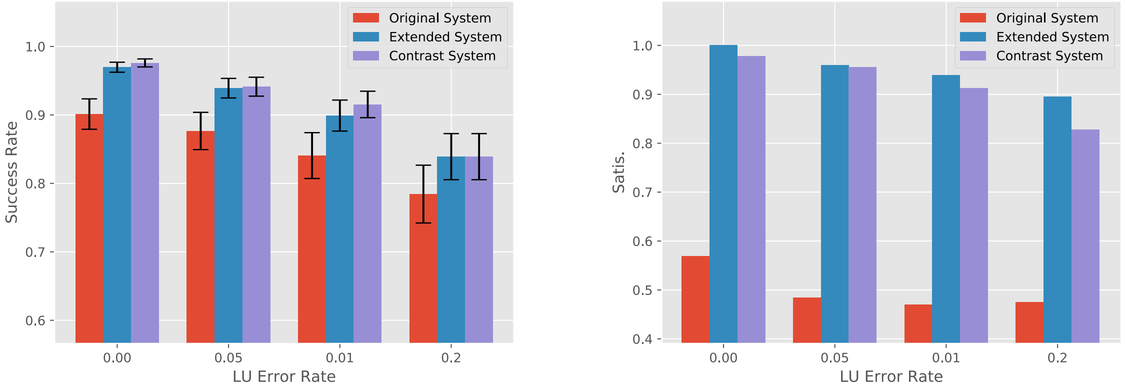

(a) Dialog success rate of different systems.

(b) Satis. of different systems.

Figure 4: Performance of different systems under simulation. The original, extended and contrast systems are shown in red, blue and purple bars. We test these systems by interacting with *Sim*2.

details of the test performance. Table 2 shows the statistics of turns when S_1 interacts with *Sim*2.

As shown in Table 1, S_1 achieves higher dialog success rate and rewards when testing with *Sim*1. When interacting with *Sim*2, nearly half of the responses to unsupported user actions are not reasonable. Notice even though *Sim*2 contains new user actions, some of the new actions might be appropriately handled by S_1. It may be due to the robustness of our RL-based system. But it's far from being desired. The unpredictable real user behavior in the deployment stage will lead to a poor user experience in real-world applications. It proves the importance of a maintainable system.

To maintain the original system, we define a few simple logic rules to handle unsupported user actions: if users confirm the value of a slot in current turn, the system should inform users of that value. These rules[8] are intuitive and reasonable to handle queries such as "Is this restaurant located in Zhongguancun?". There are four slots[9] that can be used for confirmation, so we define four logic rules in all. Due to the change in ontology, we add a new status in dialog features to represent the "confirm" intent of users. It leads to a change in the model architecture of extended dialog manager. Then we distill knowledge of the S_1 and logic rules into the **extended system**. No additional data is used to obtain the extended system.

For comparison, we retrain another new system (**contrast system**) from scratch by interacting with *Sim*2. After about 2600 interactions with *Sim*2, the performance of contrast system starts to converge. Note that in order to build the contrast system, the developers need to redesign a new user simulator or hire real users. It's expensive and impractical in industrial applications. Then we simulate 3200 interactions with *Sim*2 to obtain its performance. Fig. 4 illustrates the performance of different systems.

As can be seen, the extended system performs better than the original system in terms of dialog success rate and "Satis.". This is to a large degree attributed to the consideration of new user actions. Fig. 4(a) shows that the contrast system achieves higher dialog success rate than the extended system. But the gap is negligible. However, the contrast system is trained from scratch under a new interaction environment and the extended system is trained by transferring knowledge of the original system and logic rules. To train the contrast system, about 2600 episodes are sampled by interacting with a new interaction environment. But no additional data is used to train the extended system.

In Fig. 4(b), the "Satis." of the extended system is slightly higher than the contrast system. This is due to the fact that the extended system learns how to deal with new user actions from logic rules but the contrast system obtains dialog policy by exploring the environment. As a result, the contrast system learns a more flexible dialog policy than the extended system[10]. However, the "Satis." has a bias to the suboptimal rules,

---

[8] In the practical dialog system, we can inject more complex logic rules and take dialog history into account. These rules are not limited to question/answer mapping.

[9] They are "*name*", "*area*", "*price range*" and "*cuisine*".

[10] Table 6 in the Appendix shows sample dialogs from the extended system and contrast system.

Average Rating	2.35
Average Turns	12.1
Unseen *Intents*	takeTaxi, bookTable
Unseen *Inform Slots*	withoutWaiting, carryoutService, goodforDating, privateRoom, hasInternet
Unseen *Request Slots*	waitingLine, discount, businessHours
Unseen *Confirm Slots*	carryoutService, goodforDating, privateRoom, hasInternet

Table 3: Details of the real user logs. Users are encouraged to interact with the original system by unsupported intents and slots. We find there are 14 user actions unseen before.

Dialog Condition	System Action
takeTaxi	API call
bookTable	API call
inform unseen slots	recommend a restaurant
request unseen slots	offer information of slots
confirm unseen slots	offer information of slots

Table 4: Different types of rules for new user actions. Left column shows the dialog context condition; Right column shows the corresponding system action. We define 14 rules in all to handle newfound intents and slots shown in Table 3.

rather than the optimal policy gained from the environment. It suggests the extended system can be further refined by reinforcement learning once a new interaction environment is available.

6.6 Human Evaluation

In any case, the developers can't guarantee all user actions are considered. Fortunately, our method makes no assumptions about the new user actions and new dialog model architecture. As a result, the system can be extended over multiple iterations.

To evaluate this characteristic, we deploy the extended system[11] in section 6.5 to interact with real human users. Users are given a goal sampled from our corpus for reference. To elicit more complex situations, they are encouraged to interact with our system by new intents and slots related to the restaurant domain. At the end of each dialog, they are asked to give a subjective rating on the scale from 1 to 5 based on the naturalness of the system (1 is the worst, 5 is the best.). After filtering dialog sessions unrelated to our task, we collect 315 episodes in total. Table 3 shows the details of the user logs. As shown in Table 3, after deployment, there are a few slots and intents unseen before. For example, users may ask for the discount information or take a taxi to the restaurant. To represent the new intents and slots, the dimension of extracted dialog features is extended to 236. Meanwhile, the number of system actions is extended to 29 to handle new user actions. To deal with the newfound user actions, we define 14 rules in total. Table 4 shows the details of new defined logic rules. Then we distill the knowledge of the original system and logic rules into a new system. Fig. 5 shows the comparison in user ratings. The extended system significantly gets a higher subjective rating than the original one. It proves that the extended system can give reasonable responses in unseen dialog situations.

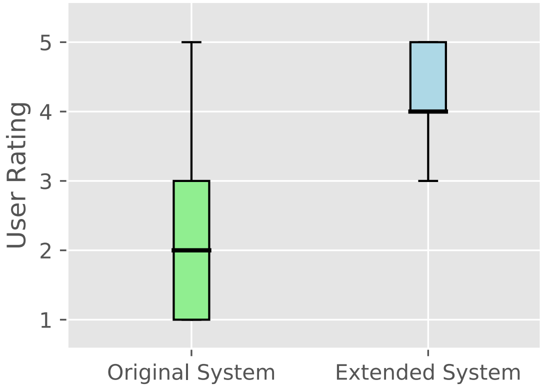

Figure 5: Distribution of user ratings.

Table 5 shows sample dialogs from the original system and extended system with real users. We can see that the extended system is much more coherent since it takes new user actions into account. It inspires us that a complicated RL-based dialog system can start from a simple one and be improved incrementally based on our framework. This design pattern will be much more efficient if the interaction environment is hard to access.

7 Conclusion

In this paper, we present a practical solution to maintain RL-based dialog systems without retraining from scratch. By reusing existing resources, developers can extend the RL-based dialog manager to deal with unpredictable user actions after deployment. Furthermore, our method holds no constraints on the architecture of the new system and retains the key benefit of data-driven methods. It means the new system can be maintained iteratively and further refined by RL once a new interaction environment is available. The sim-

---

[11] The extended system in the simulated evaluation will be the original system in our human evaluation.

Original System	Extended System
User: I want to find a Sichuan restaurant. *System*: Where do you want to go? *User*: Zhongguancun area. *System*: “Hot Charge” is a nice Sichuan restaurant in Zhongguancun area. *User*: **Can you tell me the discount?** *System*: **The average cost is 42 CNY.** *User*: **Tell me the discount!** *System*: **“Hot Charge” is a nice Sichuan restaurant in Zhongguancun area.** *User*: Bye. *System*: Bye.	*User*: I want to find a Sichuan restaurant in Zhongguancun area. *System*: “Hot Charge” is a nice Sichuan restaurant in in Zhongguancun area. *User*: **What’s the discount of this restaurant?** *System*: **This restaurant is offering 10% off.** *User*: **Is there a waiting line?** *System*: **Currently there are 5 parties waiting in line.** *User*: **Can you book a table for me?** *System*: **Ok.** *User*: Thanks. *System*: You are welcome.
Subjective Rating = 2	Subjective Rating = 5

Table 5: Sample dialogs of the original and extended systems in the human evaluation. The original system will collapse when encountering unseen slots and intents. After extension, the new system can handle more user actions.

ulation and human experiments show our proposed framework guarantees the maintainability and scalability in RL-based systems, which are necessary for any industrial application.

8 Acknowledgments

The research work described in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002103 and also supported by the Natural Science Foundation of China under Grant No. 61333018.

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM.

Cheng Chang, Runzhe Yang, Lu Chen, Xiang Zhou, and Kai Yu. 2017. Affordable on-line dialogue policy learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2190–2199.

Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu. 2017. Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2444–2454.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 484–495.

Milica Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 312–317. IEEE.

Milica Gašic, Dongho Kim, Pirros Tsiakoulis, Catherine Breslin, Matthew Henderson, Martin Szummer, Blaise Thomson, and Steve Young. 2014. Incremental on-line adaptation of pomdp-based dialogue managers to extended domains. *In Proceedings on InterSpeech*.

Matthew Henderson, Blaise Thomson, and Jason Williams. 2013. Dialog state tracking challenge 2 & 3.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2410–2420.

Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong. 2014. Learning small-size dnn with output-distribution-based criteria. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017a. Investigation of language understanding impact for reinforcement learning based dialogue systems. *arXiv preprint arXiv:1703.07055*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Xuijun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. 2017b. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2017. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. *arXiv preprint arXiv:1711.05715*.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. *arXiv preprint arXiv:1709.06136*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Tim Paek. 2006. Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment. In *Proc. Dialog-on-Dialog Workshop, Interspeech*.

Tim Paek and Roberto Pieraccini. 2008. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech communication*, 50(8):716–729.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2231–2240.

Olivier Pietquin and Thierry Dutoit. 2006. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):589–599.

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the second international conference on Human Language Technology Research*, pages 12–19. Morgan Kaufmann Publishers Inc.

Pararth Shah, Dilek Hakkani-Tür, and Larry Heck. 2016. Interactive reinforcement learning for task-oriented dialogue management. In *NIPS 2016 Deep Learning for Action and Interaction Workshop*.

Russell Stewart and Stefano Ermon. 2017. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*, pages 2576–2582.

Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. *arXiv preprint arXiv:1707.00130*.

Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016a. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2431–2441.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016b. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 665–677.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2017. Prior knowledge integration for neural machine translation using posterior regularization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1514–1523.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 1.

Discriminative Deep Dyna-Q: Robust Planning for Dialogue Policy Learning

Shang-Yu Su[⋆] **Xiujun Li**[†] **Jianfeng Gao**[†] **Jingjing Liu**[†] **Yun-Nung Chen**[⋆]
[†]Microsoft Research, Redmond, WA, USA
[⋆]National Taiwan University, Taipei, Taiwan
[⋆]`f05921117@ntu.edu.tw` [⋆]`y.v.chen@ieee.org`
[†]`{xiul,jfgao,jingjl}@microsoft.com`

Abstract

This paper presents a Discriminative Deep Dyna-Q (D3Q) approach to improving the effectiveness and robustness of Deep Dyna-Q (DDQ), a recently proposed framework that extends the Dyna-Q algorithm to integrate planning for task-completion dialogue policy learning. To obviate DDQ's high dependency on the quality of simulated experiences, we incorporate an RNN-based discriminator in D3Q to differentiate simulated experience from real user experience in order to control the quality of training data. Experiments show that D3Q significantly outperforms DDQ by controlling the quality of simulated experience used for planning. The effectiveness and robustness of D3Q is further demonstrated in a domain extension setting, where the agent's capability of adapting to a changing environment is tested.[1]

1 Introduction

There are many virtual assistants commercially available today, such as Apple's Siri, Google's Home, Microsoft's Cortana, and Amazon's Echo. With a well-designed dialogue system as an intelligent assistant, people can accomplish tasks via natural language interactions. Recent advance in deep learning has also inspired many studies in neural dialogue systems (Wen et al., 2017; Bordes et al., 2017; Dhingra et al., 2017; Li et al., 2017).

A key component in such task-completion dialogue systems is *dialogue policy*, which is often formulated as a reinforcement learning (RL) problem (Levin et al., 1997; Young et al., 2013). However, learning dialogue policy via RL from the scratch in real-world systems is very challenging, due to the inevitable dependency on the environment from which a learner acquires knowledge and receives rewards. In a dialogue scenario, real users act as the environment in the RL framework, and the system communicates with real users constantly to learn dialogue policy. Such process is very time-consuming and expensive for online learning.

One plausible strategy is to leverage user simulators trained on human conversational data (Schatzmann et al., 2007; Li et al., 2016), which allows the agent to learn dialogue policy by interacting with the simulator instead of real users. The user simulator can provide infinite simulated experiences without additional cost, and the trained system can be deployed and then fine-tuned through interactions with real users (Su et al., 2016; Lipton et al., 2016; Zhao and Eskenazi, 2016; Williams et al., 2017; Dhingra et al., 2017; Li et al., 2017; Liu and Lane, 2017; Peng et al., 2017b; Budzianowski et al., 2017; Peng et al., 2017a; Tang et al., 2018).

However, due to the complexity of real conversations and biases in the design of user simulators, there always exists the discrepancy between real users and simulated users. Furthermore, to the best of our knowledge, there is no universally accepted metric for evaluating user simulators for dialogue purpose (Pietquin and Hastie, 2013). Therefore, it remains controversial whether training task-completion dialogue agent via simulated users is a valid and effective approach.

A previous study, called Deep Dyna-Q (DDQ) (Peng et al., 2018), proposed a new strategy to learn dialogue policies with real users by combining the Dyna-Q framework (Sutton, 1990) with deep learning models. This framework incorporates a learnable environment model (*world model*) into the dialogue policy learning pipeline, which simulates dynamics of the environment and generates simulated user behaviors to supplement the limited amount of real user experience. In DDQ, real user experiences play two pivotal roles:

---

[1] The source code is available at `https://github.com/MiuLab/D3Q`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3813–3823
Brussels, Belgium, October 31 - November 4, 2018.

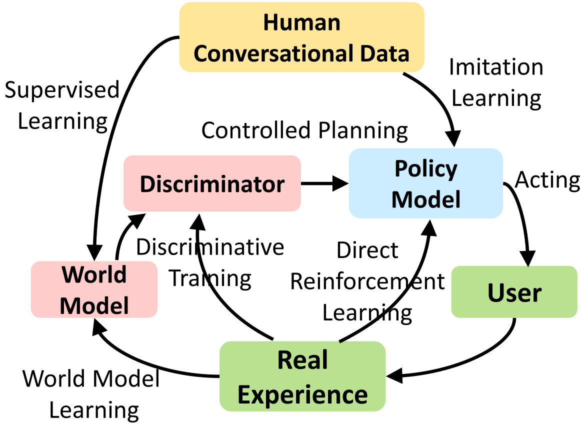

Figure 1: Proposed D3Q for dialogue policy learning.

1) directly improve the dialogue policy via RL; 2) improve the world model via supervised learning to make it behave more human-like. The former is referred to as *direct reinforcement learning*, and the latter *world model learning*. Respectively, the policy model is trained via real experiences collected by interacting with real users (*direct reinforcement learning*), and simulated experiences collected by interacting with the learned world model (*planning* or *indirect reinforcement learning*).

However, the effectiveness of DDQ depends upon the quality of simulated experiences used in planning. As pointed out in (Peng et al., 2018), although at the early stages of dialogue training it is helpful to perform planning aggressively with large amounts of simulated experiences regardless their quality, in the late stages when the dialogue agent has been significantly improved, low-quality simulated experiences often hurt the performance badly. Since there is no established method of evaluating the world model which generates simulated experiences, Peng et al. (2018) resorts to heuristics to mitigate the negative impact of low-quality simulated experiments, e.g., reducing the planning steps in the late stage of training. These heuristics need to be tweaked empirically, thus limit DDQ's applicability in real-world tasks.

To improve the effectiveness of planning without relying on heuristics, this paper proposes Discriminative Deep Dyna-Q (D3Q), a new framework inspired by generative adversarial network (GAN) that incorporates a discriminator into the planning process. The discriminator is trained to differentiate simulated experiences from real user experiences. As illustrated in Figure 1, all simulated experiences generated by the world model need to be *judged* by the discriminator, only the high-quality ones, which cannot be easily detected by the discriminator as being simulated, are used for planning. During the course of dialogue training, both the world model and discriminator are refined using the real experiences. So, the quality threshold held by the discriminator goes up with the world model and dialogue agent, especially in the late stage of training.

By employing the world model for planning and a discriminator for controlling the quality of simulated experiences, the proposed D3Q framework can be viewed as a model-based RL approach, which is generic and can be easily extended to other RL problems. In contrast, most model-based RL methods (Tamar et al., 2016; Silver et al., 2016; Gu et al., 2016; Racanière et al., 2017) are developed for simulation-based, synthetic problems (e.g., games), not for real-world problems. In summary, our main contributions in this work are two-fold:

- The proposed Discriminative Deep Dyna-Q approach is capable of controlling the quality of simulated experiences generated by the world model in the planning phase, which enables effective and robust dialogue policy learning.
- The proposed model is verified by experiments including simulation, human evaluation, and domain-extension settings, where all results show better sample efficiency over the DDQ baselines.

2 Discriminative Deep Dyna-Q (D3Q)

As illustrated in Figure 2, the D3Q framework consists of six modules: (1) an LSTM-based natural language understanding (NLU) module (Hakkani-Tür et al., 2016) for identifying user intents and extracting associated slots; (2) a state tracker (Mrkšić et al., 2017) for tracking dialogue states; (3) a dialogue policy that selects next action based on the current state; (4) a model-based natural language generation (NLG) module for generating natural language response (Wen et al., 2015); (5) a world model for generating simulated user actions and simulated rewards; and (6) an RNN-based discriminator for controlling the quality of simulated experience. Note that the controlled planning phase is realized through the world model and the discriminator, which are not

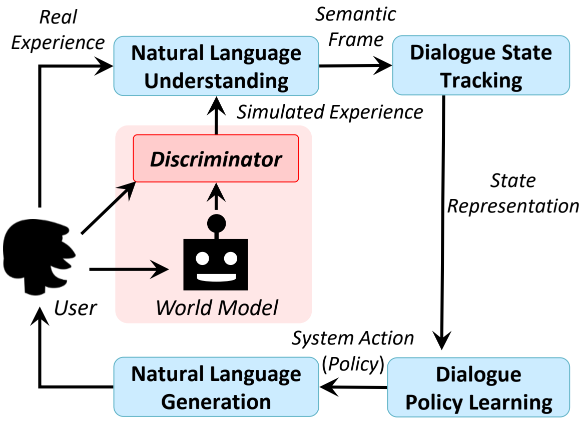

Figure 2: Illustration of the proposed D3Q dialogue system framework.

included in traditional framework of dialogue systems.

Figure 1 illustrates the whole process: starting with an initial dialogue policy and an initial world model (both are trained with pre-collected human conversational data), D3Q training consists of four stages: (1) *direct reinforcement learning*: the agent interacts with real users, collects real experiences and improves dialogue policy; (2) *world model learning*: the world model is learned and refined using real experience; (3) *discriminator learning*: the discriminator is learned and refined to differentiate simulated experience from real experience; and (4) *controlled planning*: the agent improves the dialogue policy using the high-quality simulated experience generated by the world model and the discriminator.

2.1 Direct Reinforcement Learning

In this stage, we use the vanilla deep Q-network (DQN) method (Mnih et al., 2015) to learn the dialogue policy based on real experience. We consider task-completion dialogue as a Markov Decision Process (MDP), where the agent interacts with a user through a sequence of actions to accomplish a specific user goal.

At each step, the agent observes the dialogue state s, and chooses an action a to execute, using an ϵ-greedy policy that selects a random action with probability ϵ or otherwise follows the greedy policy $a = \text{argmax}_{a'} Q(s, a'; \theta_Q)$. $Q(s, a; \theta_Q)$ which is the approximated value function, implemented as a Multi-Layer Perceptron (MLP) parameterized by θ_Q. The agent then receives reward r, observes next user response, and updates the state to s'. Finally, we store the experience tuple (s, a, r, s') in the replay buffer B^u. This cycle continues until the dialogue terminates.

We improve the value function $Q(s, a; \theta_Q)$ by adjusting θ_Q to minimize the mean-squared loss function as follows:

$$\begin{aligned} \mathcal{L}(\theta_Q) &= \mathbb{E}_{(s,a,r,s')\sim B^u}[(y_i - Q(s,a;\theta_Q))^2], \\ y_i &= r + \gamma \max_{a'} Q'(s', a'; \theta_{Q'}), \end{aligned} \tag{1}$$

where $\gamma \in [0, 1]$ is a discount factor, and $Q'(.)$ is the target value function that is only periodically updated (i.e., fixed-target). The dialogue policy can be optimized through $\nabla_{\theta_Q} \mathcal{L}(\theta_Q)$ by mini-batch deep Q-learning.

2.2 World Model Learning

To enable planning, we use a world model to generate simulated experiences that can be used to improve dialogue policy. In each turn of a dialogue, the world model takes the current dialogue state s and the last system action a (represented as an one-hot vector) as the input, and generates the corresponding user response o, reward r, and a binary variable t (indicating if the dialogue terminates). The world model $G(s, a; \theta_G)$ is trained using a multi-task deep neural network (Liu et al., 2015) to generate the simulated experiences. The model contains two classification tasks for simulating user responses o and generating terminal signals t, and one regression task for generating the reward r. The lower encoding layers are shared across all three tasks, while the upper layers are task-specific. $G(s, a; \theta_G)$ is optimized to mimic human behaviors by leveraging real experiences in the replay buffer B^u. The model architecture is illustrated in the left part of Figure 3.

$$\begin{aligned} h &= \tanh(W_h(s,a) + b_h), \\ r &= W_r h + b_r, \\ o &= \texttt{softmax}(W_a h + b_a), \\ t &= \texttt{sigmoid}(W_t h + b_t), \end{aligned}$$

where (s, a) is the concatenation of s and a, and all W and b are weight matrices and bias vectors, respectively.

2.3 Discriminator Learning

The discriminator, denoted by D, is used to differentiate simulated experience from real experience. D is a neural network model with its architecture illustrated in the right part of Figure 3. D employs

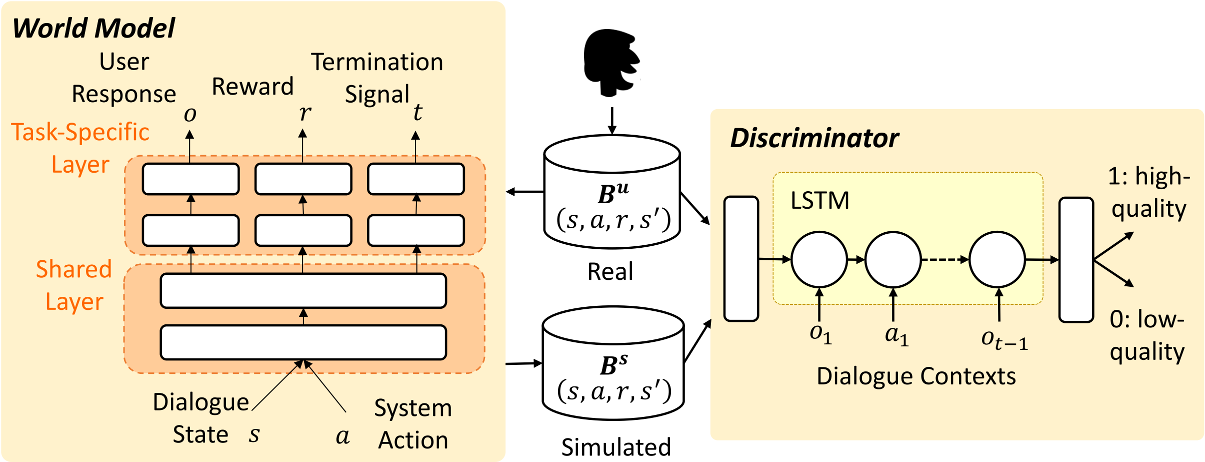

Figure 3: The model architectures of the world model and the discriminator for controlled planning.

an LSTM to encode a dialogue as a feature vector, and a Multi-Layer Perceptron (MLP) to map the vector to a probability indicating whether the dialogue looks like being generated by real users.

D is trained using the simulated experience generated by the world model G and the collected real experience x. We use the objective function as

$$\mathbb{E}_{real}[\log D(x)] + \mathbb{E}_{simu}[\log(1 - D(G(.)))]. \quad (2)$$

Practically, we use the mini-batch training and the objective function can be rewritten as

$$\frac{1}{m}\sum_{i=1}^{m}[\log D(x^{(i)}) + \log(1 - D(G(.)^{(i)}))], \quad (3)$$

where m represents the batch size.

2.4 Controlled Planning

In this stage, we apply the world model G and the discriminator D to generate high-quality simulated experience to improve dialogue policy. The D3Q method uses three replay buffers, B^u for storing real experience, B^s for simulated experience generated by G, and B^h for high-quality simulated experience generated by G and D. Learning and planning are implemented by the same DQN algorithm, operating on real experience in B^u for learning and on simulated experience in B^h for planning. Here we only describe how the high-quality simulated experience is generated.

At the beginning of each dialogue session, we uniformly draw a user goal (C, R) (Schatzmann et al., 2007), where C is a set of constraints and R is a set of requests. For example, in movie-ticket booking dialogue, *constraints* are the slots with specified values, such as the name, the date of the movie and the number of tickets to buy. And *requests* can contain slots which the user plans to acquire the values for, such as the start time of the movie. The first user action o_1 can be either a `request` or an `inform` dialogue act. A request dialogue act consists of a request slot, multiple constraint slots and the corresponding values, uniformly sampled from R and C. For example, `request(theater; moviename=avergers3)`. An inform dialogue act contains constraint-slots only. Semantic frames can also be transformed into natural language via NLG component, e.g., "*which theater will play the movie avergers3?*"

For each dialogue episode with a sampled user goal, the agent interacts with world model $G(s, a; \theta_G)$ to generate a simulated dialogue session, which is a sequence of simulated experience tuples (s, a, r, s'). We always store the G-generated session in B^s, but only store it in B^h if it is selected by discriminator D. We repeat the process until K simulated dialogue sessions are added in B^h, where K is a pre-defined planning step size. This can be viewed as a sampling process. In theory if the world model G is not well-trained this process could take forever to generate K high-quality samples accepted by D. Fortunately, this never happened in our experiments because D is trained using the simulated experience generated by G and D is updated whenever G is refined.

Now, we compare controlled planning in D3Q with the planning process in the original DDQ (Peng et al., 2018). In DDQ, after each step of di-

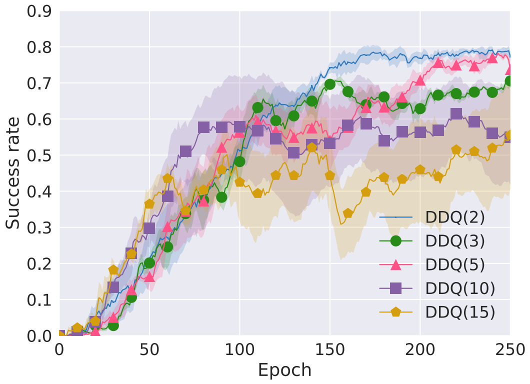

Figure 4: The learning curves of DDQ(K) agents where $(K-1)$ is the number of planning steps.

rect reinforcement learning, the agent improves its policy via K steps of planning. A larger planning step means that more simulated experiences generated by G are used for planning. Theoretically, larger amounts of *high-quality* simulated experiences can boost the performance of the dialogue policy more quickly. However, the world model by no means perfectly reflects real human behavior, and the generated experiences, if of low quality, can have negative impact on dialogue policy learning. Prior work resorts to heuristics to mitigate the impact. For example, Peng et al. (2018) proposed to reduce planning steps at the late stage of policy learning, thus forcing all DDQ agents to converge to the same one trained with a small number of planning steps.

Figure 4 shows the performance of DDQ agents with different planning steps without heuristics. It is observable that the performance is unstable, especially for larger planning steps, which indicates that the quality of simulated experience is becoming more pivotal as the number of planning steps increases.

D3Q resolves this issue by introducing a discriminator and allows only high-quality simulated experience, judged by the discriminator, to be used for planning. In the next section, we will show that D3Q does not suffer from the problem of DDQ and the D3Q training is quite stable even with large sizes of planning steps.

3 Experiments

We evaluate D3Q on the movie-ticket booking task with both simulated users and real users in two settings: full domain and domain extension.

Full Domain & Domain Extension	
Intent	request, inform, deny, confirm_question, confirm_answer, greeting, closing, not_sure, multiple_choice, thanks, welcome
Full Domain	
Slot	city, closing, date, distanceconstraints, greeting, moviename, numberofpeople, price, starttime, state, taskcomplete, theater, theater_chain, ticket, video_format, zip
Domain Extension	
Slot	city, closing, date, distanceconstraints, greeting, moviename, numberofpeople, price, starttime, state, taskcomplete, theater, theater_chain, ticket, video_format, zip, genre, other

Table 1: The data schema for full domain and domain extension settings.

3.1 Dataset

Raw conversational data in a movie-ticket booking scenario was collected via Amazon Mechanical Turk. The dataset has been manually labeled based on a schema defined by domain experts, as shown in Table 1, consisting of 11 intents and 16 slots in the full domain setting, while there are 18 slots in the domain extension setting. Most of these slots can be both "inform slots" and "request slots", except for a few. For example, the slot `number_of_people` is categorized as an inform slot but not a request slot, because arguably the user always knows how many tickets she/he wants. In total, the dataset contains 280 annotated dialogues, the average length of which is approximately 11 turns.

3.2 Baselines

To verify the effectiveness of D3Q, we developed different versions of task-completion dialogue agents as baselines to compare with.

- A **DQN** agent is implemented with only direct reinforcement learning in each episode.

- The **DQN(K)** has K times more real experiences than the DQN agent. The performance of DQN(K) can be viewed as the upper bound of DDQ(K) and D3Q(K) with the same number of planning steps $(K-1)$, as these models have the same training settings and the same amount of training samples during the entire learning process.

- The **DDQ(K)** agents are learned with an initial world model pre-trained on human conversational data, with $(K-1)$ as the number of planning steps. These agents store the simulated experience without being judged by the discriminator.

Proposed D3Q

- The **D3Q(K)** agents are learned through the process described in Section 2.4.
- The **D3Q(K, fixed θ_D)** agents are learned as described in Section 2.4 without training discriminator. The D3Q(K, fixed θ_D) agents are only evaluated in the simulation setting.

3.3 Implementation

Settings and Hyper-parameters ϵ-greedy is always applied for exploration. We set the discount factor γ = 0.9. The buffer size of B^u and B^h is set to 2000 and 2000 $\times K$ *planning steps*, respectively. The batch size is 16, and the learning rate is 0.001. To prevent gradient explosion, we applied gradient clipping on all the model parameters to maximum norm = 1. All the NN models are randomly initialized. The high-quality simulated experience buffer B^h and the simulated experience buffer B^s are initialized as empty. The target network is updated at the beginning of each training episode. The optimizer for all the neural networks is RMSProp (Hinton et al., 2012). The maximum length of a simulated dialogue is 40. If exceeding the maximum length, the dialogue fails. To make dialogue training efficient, we also applied a variant of imitation learning, called Reply Buffer Spiking (RBS) (Lipton et al., 2016), by building a simple and straightforward rule-based agent based on human conversational dataset. We then prefilled the real experience replay buffer B^u with experiences of 50 dialogues, before training for all the variants of models. The batch size for collecting experiences is 10, which means if the running agent is DDQ/D3Q(K), 10 real experience tuples and 10 $\times$ ($K - 1$) simulated experience tuples are stored into the buffers at every episode.

Agents For all the models (DQN, DDQ, and D3Q) and their variants, the value networks $Q(.)$ are MLPs with one hidden layer of size 80 and ReLU activation.

World Model For all the models (DDQ and D3Q) and their variants, the world models $M(.)$ are MLPs with one shared hidden layer of size 160, hyperbolic-tangent activation, and one encoding layer of hidden size 80 for each state and action input.

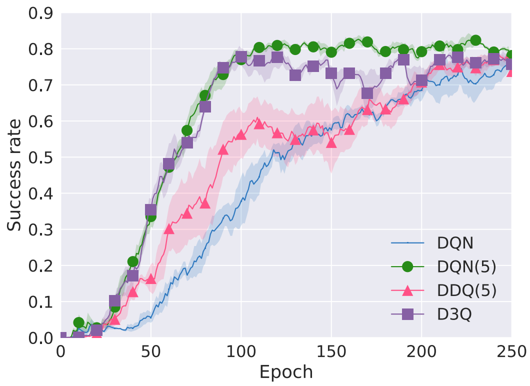

Figure 5: The learning curves of agents (DQN, DDQ, and D3Q) under the full domain setting.

Discriminator In the proposed D3Q framework, the LSTM cell is utilized, the hidden size is 128. The encoding layer for the current state and output layer are MLPs with single hidden layer of size 80. The threshold interval is set to range between 0.45 and 0.55, i.e., only when $0.45 \leq D(x) \leq 0.55$ that x would be stored into the buffer B^h.

3.4 Simulation Evaluation

In this setting, the dialogue agents are optimized by interacting with the user simulators instead of with real users. In another word, the world model is trained to mimic user simulators. In spite of the discrepancy between simulators and real users, this setting endows us with the flexibility to perform a detailed analysis of models without much cost, and to reproduce experimental results easily.

User Simulator We used an open-sourced task-oriented user simulator (Li et al., 2016) in our simulated evaluation experiments (Appendix A for more details). The simulator provides the agent with a simulated user response in each dialogue turn along with a reward signal at the end of the dialogue. A dialogue is considered successful if and only if a movie ticket is booked successfully, and the information provided by the agent satisfies all the constraints of the sampled user goal. At the end of each dialogue, the agent receives a positive reward $2 * L$ for success, or a negative reward $-L$ for failure, where L is the maximum number of turns in each dialogue, and is set to 40 in our experiments. Furthermore, in each turn, a reward -1 is provided to encourage shorter dialogues.

Agent	Epoch = 100			Epoch = 200			Epoch = 300		
	Success	**Reward**	**Turns**	**Success**	**Reward**	**Turns**	**Success**	**Reward**	**Turns**
DQN	.4467	2.993	23.21	.7000	36.08	17.84	.7867	48.45	13.91
DDQ(5)	.5467	16.57	20.07	**.7133**	**39.23**	**14.73**	**.8067**	**50.73**	14.13
DDQ(5, rand-init θ_G)	.6067	23.55	20.49	.6267	26.30	19.80	.6667	32.92	16.16
DDQ(5, fixed θ_G)	.5867	20.62	21.56	.1667	-33.71	29.41	.2267	-22.68	21.76
D3Q(5)	**.7467**	**43.59**	**14.03**	.6800	34.64	15.92	.7200	40.85	**13.11**
D3Q(5, fixed θ_D)	.6800	33.86	17.48	.7000	36.57	16.85	.6933	35.67	17.06
DQN(5)	*.7400*	*42.19*	*15.23*	*.8533*	*57.76*	*11.28*	*.7667*	*46.56*	*12.88*
DDQ(10)	.5733	24.00	**11.60**	.5533	19.89	15.01	.4800	10.04	17.12
DDQ(10, rand-init θ_G)	.5000	12.79	16.41	.5333	17.71	**14.57**	.6000	24.98	16.04
DDQ(10, fixed θ_G)	.3467	-10.25	25.69	.2400	-23.38	26.36	.0000	-55.53	33.07
D3Q(10)	.6333	28.99	16.01	.7000	37.24	15.52	.6667	33.09	15.83
D3Q(10, fixed θ_D)	**.7133**	**36.36**	20.48	**.8400**	**54.87**	20.48	**.7400**	**42.89**	**13.81**
DQN(10)	*.8333*	*55.5*	*11.00*	*.7733*	*47.99*	*11.61*	*.7733*	*47.68*	*12.24*

Table 2: Results of different agents at training epoch = {100, 200, 300}. Each number is averaged over 3 runs, each run tested on 50 dialogues. (Success: success rate, Reward: Average Reward, Turns: Average Turns)

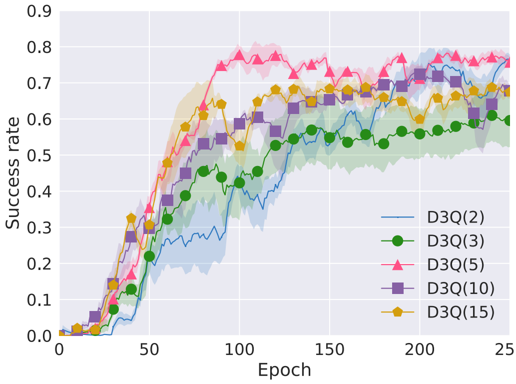

Figure 6: The learning curves of D3Q(K) agents which (K-1) is the number of planning steps (K = 2, 3, 5, 10, 15).

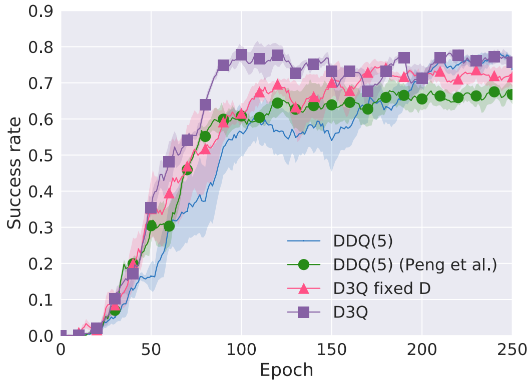

Figure 7: The learning curves of D3Q, DDQ(5), DDQ(5) (Peng et al., 2018), and D3Q fixed θ_D agents.

Full Domain The learning curves of the models in the full domain setting are depicted in the figure 5. The results show that the proposed D3Q agent (the pink curve) significantly outperforms the baselines DQN and DDQ(5), and has similar training efficiency to DQN(5). Note that here the planning steps of D3Q is 4, which means D3Q (pink) and DDQ(5) (purple) use the same amount of training samples (both real and simulated experiences) to update the agent throughout the whole training process. The difference between these two agents is that D3Q employs a discriminator as a quality judge. The experimental result shows that our proposed framework could boost the learning efficiency even without any pre-training on the discriminator. Furthermore, D3Q (pink) uses the same amount of training samples as DQN(5) (green), while the proposed model uses only 20% of real experience from human. The efficacy and feasibility of D3Q is hereby justly verified.

As mentioned in the previous section, a large number of planning steps means leveraging a large amount of simulated experience to train the agents. The experimental result (Figure 4) shows that the DDQ agents are highly sensitive to the quality of simulated experience. In contrast, the proposed D3Q framework demonstrates robustness to the number of planning steps (Figure 6). Figure 7 shows that D3Q also outperforms DDQ original setting (Peng et al., 2018) and D3Q without training discriminator. The performance detail including success rate, reward, an number of turns is shown in Table 2. From the table, with fewer simulated experiences, the difference between DDQ and D3Q may not be significant, where DDQ agents achieve about 50%-60% success rate and D3Q agents achieve higher than 68% success rate after 100 epochs. However, when the number of

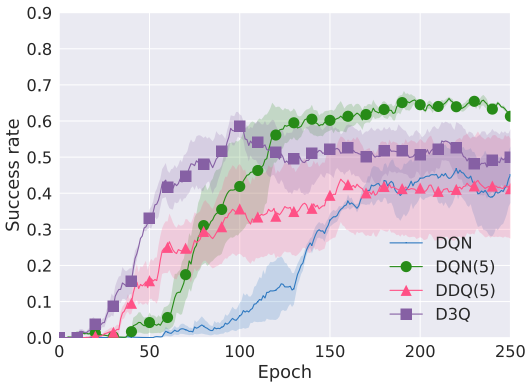

Figure 8: The learning curves of agents (DQN, DDQ, and D3Q) under the domain extension setting.

planning steps increases, more fake experiences significantly degrade the performance for DDQ agents, where DDQ(10, fixed θ_G) suffers from bad simulated experiences after 300 epochs and achieves 0% success rate.

Domain Extension In the domain extension experiments, more complicated user goals are adopted. Moreover, we narrow down the action space into a small subspace instead of that used in full-domain setting, and gradually introduce more complex user goals and expand the action space as the training proceeds. Specifically, we start from a set of necessary slots and actions to accomplish most of the user goals, and then extend the action space and complexity of user goals once every 20 epoch (after epoch 50). Note that the domain will keep extending and become full-domain after epoch 130. Such experimental setting makes the training environment more complicated and unstable than the previous full-domain one.

The results summarized in Figure 8 show that D3Q significantly outperforms the baseline methods, demonstrating its robustness. Furthermore, D3Q shows remarkable learning efficiency while extending the domain, which even outperforms DQN(5). A potential reason might be that the world model could improve exploration in such unstable and noisy environment.

3.5 Human Evaluation

In the human evaluation experiments, real users interact with different models without knowing which agent is behind the system. At the beginning of each dialogue session, one of the agents was randomly picked to converse with the user.

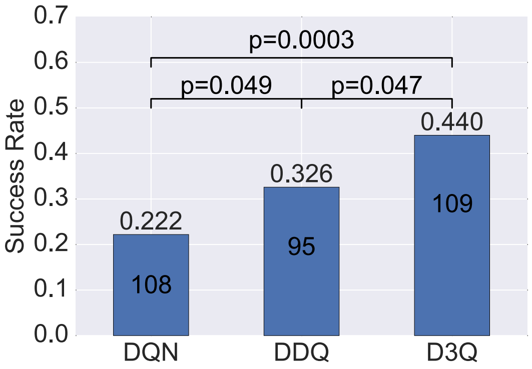

Figure 9: The human evaluation results of D3Q, DDQ(5), and D3Q in the full domain setting, the number of test dialogues indicated on each bar, and the p-values from a two-sided permutation test (difference in mean is significant with $p < 0.05$).

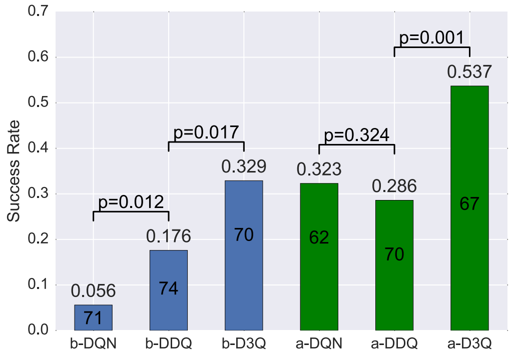

Figure 10: The human evaluation results of DQN, DDQ(5), and D3Q in the domain extension setting, the number of test dialogues indicated on each bar. The prefix 'b-' implies that the trained models are picked before the environment extends to full domain, while the prefix 'a-' indicates that the trained models are picked after the environment becomes full domain (difference in mean is significant with $p < 0.05$).

The user was instructed to converse with the agent to complete a task given a user goal sampled from the corpus. The user can abandon the task and terminate the dialogue at any time, if she or he believes that the dialogue was unlikely to succeed, or simply because the dialogue drags on for too many turns. In such cases, the dialogue session is considered as failure.

Full Domain Three agents (DQN, DDQ(5), and D3Q) trained in the full domain setting (Figure 5) at epoch 100 are selected for testing. As illustrated in Figure 9, the results of human evaluation are consistent with those in the simulation evaluation (Section 3.4), and the proposed D3Q significantly outperforms other agents.

Domain Extension To test the adaptation capability of the agents to the complicated, dynamically changing environment, we selected three trained agents (DQN, DDQ(5), and D3Q) at epoch 100 before the environment extends to full domain, and another three agents trained at epoch 200 after the environment becomes full domain. Figure 10 shows that the results are consistent with those in the simulation evaluation (Figure 8), and the proposed D3Q significantly outperforms other agents in both stages.

4 Conclusions

This paper proposes a new framework, Discriminative Deep Dyna-Q (D3Q), for task-completion dialogue policy learning. With a discriminator as judge, the proposed approach is capable of controlling the quality of simulated experience generated in the planning phase, which enables efficient and robust dialogue policy learning. Furthermore, D3Q can be viewed as a generic model-based RL approach easily-extensible to other RL problems.

We validate the D3Q-trained dialogue agent on a movie-ticket-booking task in the simulation, human evaluation, and domain-extension settings. Our results show that the D3Q agent significantly outperforms the agents trained using other state-of-the-art methods including DQN and DDQ.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback on the work. We would like to acknowledge the volunteers from Microsoft for participating the human evaluation experiments. Shang-Yu Su and Yun-Nung Chen are supported by the Ministry of Science and Technology of Taiwan and MediaTek Inc.

References

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proceedings of ICLR*.

Pawel Budzianowski, Stefan Ultes, Pei-Hao Su, Nikola Mrksic, Tsung-Hsien Wen, Inigo Casanueva, Lina Rojas-Barahona, and Milica Gasic. 2017. Subdomain modelling for dialogue management with hierarchical reinforcement learning. *arXiv preprint arXiv:1706.06210*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of ACL*, pages 484–495.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. 2016. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of Interspeech*.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1997. Learning dialogue strategies within the markov decision process framework. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 72–79. IEEE.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of The 8th International Joint Conference on Natural Language Processing*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. 2016. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *Proceedings of 2017 IEEE Workshop on Automatic Speech Recognition and Understanding*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1777–1788.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Yun-Nung Chen, and Kam-Fai Wong. 2017a. Adversarial advantage actor-critic model for task-completion dialogue policy learning. *arXiv preprint arXiv:1710.11277.*

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Kam-Fai Wong, and Shang-Yu Su. 2018. Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176.*

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017b. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2221–2230.

Olivier Pietquin and Helen Hastie. 2013. A survey on metrics for the evaluation of user simulations. *The knowledge engineering review.*

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. 2017. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 5694–5705.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL 2007; Companion Volume, Short Papers*, pages 149–152. Association for Computational Linguistics.

David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. 2016. The predictron: End-to-end learning and planning. *arXiv preprint arXiv:1612.08810.*

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689.*

Richard S Sutton. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the seventh international conference on machine learning*, pages 216–224.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. 2016. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162.

Da Tang, Xiujun Li, Jianfeng Gao, Chong Wang, Lihong Li, and Tony Jebara. 2018. Subgoal discovery for hierarchical dialogue policy learning. *arXiv preprint arXiv:1804.07855.*

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*, pages 438–449.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics.*

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560.*

A User Simulator

In the task-completion dialogue setting, the entire conversation is around a user goal implicitly, but the agent knows nothing about the user goal explicitly and its objective is to help the user to accomplish this goal. Generally, the definition of user goal contains two parts:

- *inform_slots* contain a number of slot-value pairs which serve as constraints from the user.
- *request_slots* contain a set of slots that user has no information about the values, but wants to get the values from the agent during the conversation. ticket is a default slot which always appears in the *request_slots* part of user goal.

To make the user goal more realistic, we add some constraints in the user goal: slots are split into two groups. Some of slots must appear in the user goal, we called these elements as *Required slots*. In the movie-booking scenario, it includes moviename, theater, starttime, date, numberofpeople; the rest slots are *Optional slots*, for example, theater_chain, video_format etc.

We generated the user goals from the labeled dataset using two mechanisms. One mechanism is to extract all the slots (known and unknown) from the first user turns (excluding the greeting user turn) in the data, since usually the first turn contains some or all the required information from user. The other mechanism is to extract all the slots (known and unknown) that first appear in all the user turns, and then aggregate them into one user goal. We dump these user goals into a file as the user-goal database. Every time when running a dialogue, we randomly sample one user goal from this user goal database.

A Self-Attentive Model with Gate Mechanism for Spoken Language Understanding

Changliang Li[1], **Liang Li**[2], **Ji Qi**[1]
[1]Kingsoft AI Lab
[2]Tsinghua University
[1]{lichangliang, qiji1}@kingsoft.com
[2]liliang17@mails.tsinghua.edu.cn

Abstract

Spoken Language Understanding (SLU), which typically involves intent determination and slot filling, is a core component of spoken dialogue systems. Joint learning has shown to be effective in SLU given that slot tags and intents are supposed to share knowledge with each other. However, most existing joint learning methods only consider joint learning by sharing parameters on surface level rather than semantic level. In this work, we propose a novel self-attentive model with gate mechanism to fully utilize the semantic correlation between slot and intent. Our model first obtains intent-augmented embeddings based on neural network with self-attention mechanism. And then the intent semantic representation is utilized as the gate for labelling slot tags. The objectives of both tasks are optimized simultaneously via joint learning in an end-to-end way. We conduct experiment on popular benchmark ATIS. The results show that our model achieves state-of-the-art and outperforms other popular methods by a large margin in terms of both intent detection error rate and slot filling F1-score. This paper gives a new perspective for research on SLU.

1 Introduction

One long-term goal in artificial intelligence field is to build an intelligent human-machine dialogue system, which is capable of understanding human's language and giving smooth and correct responses. A typical dialogue system is designed to execute the following components: (i) automatic speech recognition converts a spoken query into transcription, (ii) spoken language understanding component analyzes the transcription to extract semantic representations, (iii) dialogue manager interprets the semantic information and decides the best system action, according to which the system response is further generated either as a natural language output(Jurafsky, 2000).

In this paper, we focus on spoken language understanding which is a core component of a spoken dialogue system. It typically involves two major tasks, intent determination and slot filling. Intent determination aims to automatically identify the intent of the user as expressed in natural language. Slot filling aims to extract relevant semantic constituents from the natural language sentence towards achieving a goal.

Usually, intent detection and slot filling are carried out separately. However, separate modeling of these two tasks is constrained to take full advantage of all supervised signals. Joint learning of intent detection and slot filling is worthwhile for three reasons. Firstly, the two tasks usually appear simultaneously in SLU systems. Secondly, the information of one task can be utilized in the other task to promote each other and a joint prediction can be made (Zhang and Wang, 2016). For example, if the intent of a utterance is to find a flight, it is likely to contain the departure and arrival cities, and vice versa. Lastly, slot tags and intents, as semantics representations of user behaviours, are supposed to share knowledge with each other.

Recently, joint model for intent detection and slot filling has achieved much progress. (Xu and Sarikaya, 2013) proposed using CNN based triangular CRF for joint intent detection and slot filling. (Guo et al., 2014) proposed using a recursive neural network that learns hierarchical representations of the input text for the joint task. (Liu and Lane, 2016b) describes a recurrent neural network (RNN) model that jointly performs intent detection, slot filling and language modeling. The neural network models keep updating the intent prediction as word in the transcribed utterance arrives and uses it as contextual features in the joint model.

In this work, we propose a novel model for

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3824–3833
Brussels, Belgium, October 31 - November 4, 2018.

joint intent determination and slot filling by introducing self-attention and gating mechanism. Our model can fully utilize the semantic correlation between slot and intent. To the best of our knowledge, this is the first attempt to utilize intent-augmented embedding as a gate to guide the learning of slot filling task. To fully evaluate the efficiency of our model, we conduct experiment on Airline Travel Information Systems (ATIS) dataset (Hemphill et al., 1990), which is popularly used as benchmark in related work. And empirical results show that our independent model outperforms the previous best result by 0.54% in terms of F1-score on slot filling task, and gives excellent performance on intent detection task. Our joint model further promotes the performance and achieves state-of-the-art results on both tasks.

The rest of our paper is structured as follows: Section 2 discusses related work, Section 3 gives a detailed description of our model, Section 4 presents experiments results and analysis, and Section 5 summarizes this work and the future direction.

2 Related Work

There is a long research history for spoken dialogue understanding, which emerged in the 1990s from some call classification systems (Gorin et al., 1997) and the ATIS project. In this section, we describe some typical works on intent classification and slot-filling, which are both core tasks of SLU (De Mori, 2007).

For intent detection task, the early traditional method is to employ n-grams as features with generic entities, such as locations and dates (Zhang and Wang, 2016). This type of method is restricted to the dimensionality of the input space. Another line of popular approaches is to train machine learning models on labeled training data (Young, 2002; Hahn et al., 2011). For example, SVM (Haffner et al., 2003) and Adaboost (Schapire and Singer, 2000) have been explored to improve intent detection. Approaches based on neural network architecture have shown good performance on intent detection task. Deep belief networks (DBNs) have been first used in call routing classification (Deoras and Sarikaya, 2013). More recently, RNNs have shown excellent performance on the intent classification task (Ravuri and Stolcke, 2015).

For slot-filling task, traditional approaches are based on conditional random fields (CRF) architecture, which has strong ability on sequence labelling (Raymond and Riccardi, 2007). Recently, models based on neural network and its extensions have shown excellent performance on the slot filling task and outperform traditional CRF models. For example, (Yao et al., 2013) proposed to take words as input in a standard recurrent neural network language model, and then to predict slot labels rather than words on the output side. (Yao et al., 2014b) improved RNNs by using transition features and the sequence-level optimization criterion of CRF to explicitly model dependencies of output labels. (Mesnil et al., 2013) tried bi-directional and hybrid RNN to investigate using RNN for slot filling. (Yao et al., 2014a) introduced LSTM architecture for this task and obtained a marginal improvement over RNN. Besides, following the success of attention based models in the NLP field, (Simonnet et al., 2015) applied the attention-based encoder-decoder to the slot filling task, but without LSTM cells.

Recently, there has been some work on learning intent detection and slot filling jointly exploited by neural networks. Slot labels and intents, as semantics of user behaviors, are supposed to share knowledge with each other. (Guo et al., 2014) adapted recursive neural networks (RNNs) for joint training of intent detection and slot filling. (Xu and Sarikaya, 2013) described a joint model for intent detection and slot filling based on convolutional neural networks (CNN). The proposed architecture can be perceived as a neural network version of the triangular CRF model (TriCRF). (Hakkani-Tür et al., 2016) proposed a single recurrent neural network architecture that integrates the three tasks (domain detection, intent detection and slot filling for multiple domains) in a model. (Liu and Lane, 2016a) proposed an attention-based neural network model for joint intent detection and slot filling. Their joint model got the best performance of 95.98% slot filling F1-score and 1.57% intent error rate in the ATIS dataset.

Despite the great progress those methods have achieved, it is still a challenging and open task for intent detection and slot filling. Therefore, we are motivated to design a powerful model, which can improve the performance of SLU systems.

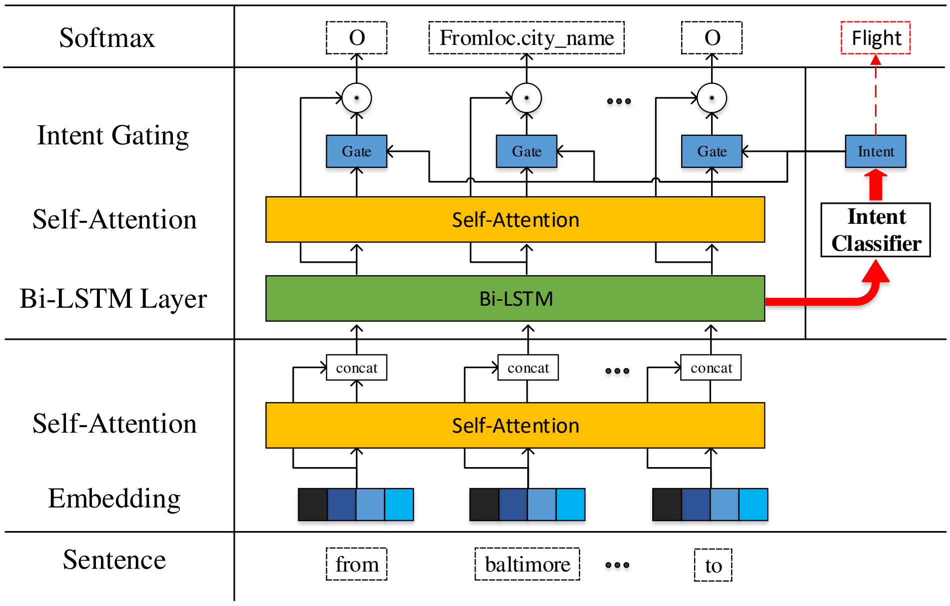

Figure 1: Illustration of our proposed model for joint intent detection and slot filling. Red arrows represent the intent classification task based on the weighted average of BiLSTM outputs. The embeddings coloured with different intensity values denote word-level and char-level embeddings (three kinds of convolution kernels).

3 Model

In this section, we present our model for the joint learning of intent detection and slot filling. Figure 1 gives an overview of our model.

The first layer maps input sequence into vectors by concatenating its word-level embeddings and character-level embeddings (obtained by convolution). And we use these vectors as merged embeddings in downstream layers. In many situations, contextual information is useful in sequence labelling. In this paper, we introduce an approach that leverages context-aware features at each time step. In particular, we make use of self-attention to produce context-aware representations of the embeddings. Then a bidirectional recurrent layer takes as input the embeddings and context-aware vectors to produce hidden states. In the last step, we propose to exploit the intent-augmented gating mechanism to match the slot label. The gate for a specific word is obtained by taking a linear transformation of the intent embedding and another contextual representation of this word computed by self-attention. We apply element-wise dot-product between the gate and each BiLSTM output.

Finally, a softmax layer is added to classify the slot labels on top of the gate layer. For simplicity, we only take the weighted average of BiLSTM outputs to predict the intent label.

The design of this structure is motivated by the effectiveness of multiplicative interaction among vectors and by self-attention mechanism which has been used successfully in a variety of tasks (Cheng et al., 2016; Vaswani et al., 2017; Lin et al., 2017). It also typically corresponds to our finding that the intent is highly correlated with slot label in some cases, so the semantics of intent should be useful for detecting the slot labels.

3.1 Embedding Layer

We first convert the indexed words $\mathbf{w} = (w_1, w_2, ..., w_T)$ to word-level embeddings $E^w = [e_1^w, e_2^w, ..., e_T^w]$, and character-level embeddings $E^c = [e_1^c, e_2^c, ..., e_T^c]$. Although word embeddings are sufficient for many NLP task, provided by a well-pretrained glove[1] or word2vec[2], character-level information provides some more prior knowledge (e.g. morphemes) to the embedding learning procedure. Some morphemic correlated words are more close in vector space, which is useful for identifying the slot labels. Character embeddings also alleviate the out-of-vocabulary (OOV) problem in the testing phase. In this paper we focus on a character-aware convolution layer used in (Kim et al., 2016) for words. The

---

[1] http://nlp.stanford.edu/projects/glove/

[2] https://code.google.com/p/word2vec/

character-level embeddings are generated by convolution over characters in the word with multiple window size to extract n-gram features.

Let $\mathcal{C}$ be the vocabulary of characters, $\mathcal{V}$ be the vocabulary of words. The dimensions of character-level embedding and word-level embedding are denoted as d_c and d_w, respectively. For each word $w_t \in \mathcal{V}$, characters in w_t constitute the matrix $C^t \in \mathbb{R}^{d_c \times l}$, where the columns corresponds to l character embeddings.

A narrow convolution is applied between C^t and a filter (or kernel) $H \in \mathbb{R}^{d_c \times w}$. Here we suppose the filter width is w. After that, we obtain a feature map $\mathbf{f}^t \in \mathbb{R}^{l-w+1}$ by adding a nonlinearity activation. The final n-gram features is generated by taking the max-over-time:

$$\mathbf{f}^t[i] = relu(H \cdot C^t[:, i : i+w-1] + b) \tag{1}$$

$$c^t = \max_i \mathbf{f}^t[i] \tag{2}$$

where $C^t[:, i : i+w-1]$ is the i-to-(i+w-1)-th column of C^t, and the character-level embedding e_t^c is made up of multiple c^t generated by different convolution kernels.

3.2 Self-Attention

Attention mechanism is usually used to guide the forming of sentence embedding, extra knowledge is also used to weigh the CNN or LSTM hidden states (i.e. document words sometimes attend to question information). However in slot filling task, the input to our model is just one sequence. So the attention mechanism used here is called self-attention, that is to say, the word at each time step attends to the whole words in this sentence. And it helps to determine which region is likely to be a slot. Since the embedding at each time step consists of multiple parts (i.e. word embedding and character embeddings of different kernel width), each part has its own semantic meaning. As shown in Figure 2, we divide the embedding into multiple parts and the attention of each part is processed within its corresponding dimension. In this approach, we restrict the interaction among different aspects of the embedding. We hypothesize that different semantic parts are relatively independent and play different roles in our network.

Suppose $M \in \mathbb{R}^{d_m \times T}$ to be the matrix containing sentence hidden vectors $[m_1, ..., m_T]$, where d_m is the dimension of these T vectors. Considering the characteristics of slot filing task, our aim is to encode each hidden vector into a context-aware representation. We achieve that by using attention over all the sentence hidden vectors M. Firstly, We linearly map all the vectors in M to three feature spaces by different projection parameters W_a, W_b and W_c, so the resulting vectors are expressed as M_a, M_b and M_c with the same shape as M. These matrices are shared across all time steps. Considering the structure of embedding which consists of K different parts (we use 4 kinds of embeddings with the same dimension), these transformed matrices are equally split into K parts. Furthermore, the attention weight is computed by dot product between M_a and M_b. Lastly, the attention output is a weighted sum of M_c.

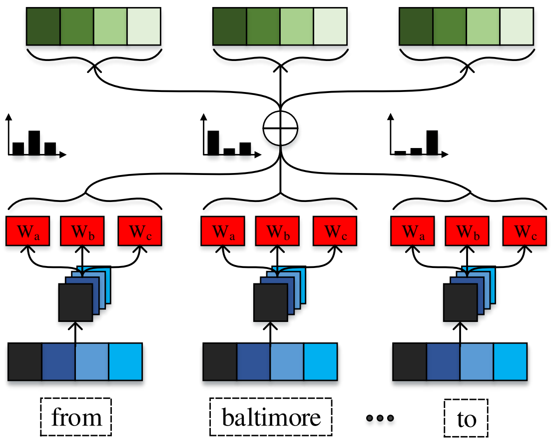

Figure 2: The structure of self-attention layer. Red coloured rectangles stand for matrices which map the input to different subspaces. These transformed vectors are divided into multiple parts for computing self-attention.

Specifically, we consider different K parts in detail for $k = 1, .., K$:

$$\begin{bmatrix} M_a \\ M_b \\ M_c \end{bmatrix} = \begin{bmatrix} W_a M \\ W_b M \\ W_c M \end{bmatrix} \tag{3}$$

$$\alpha_{k,t} = \text{softmax}(m_{k,a,t}^{\mathrm{T}} M_{k,b}) \tag{4}$$

$$\text{S-Att}(m_t, M) = [M_{1,c}\alpha_{1,t}^{\mathrm{T}}, ..., M_{K,c}\alpha_{K,t}^{\mathrm{T}}] \tag{5}$$

where $M_{k,a} \in \mathbb{R}^{(d_m/K) \times T}$ is the k-th part of M_a which is transformed from M by W_a. Index t is word position ranging over T time steps and $m_{k,a,t} \in \mathbb{R}^{d_m/K}$ is the t-th column of $M_{k,a}$. $\alpha_{k,t}$

is the attention weights over $M_{k,c}$. The output of self-attention module generated at time step t is the concatenation of K parts by using Equation 5.

3.3 BiLSTM

Character embeddings and word embeddings are both important features in our task. To further utilize these features, we associate each embedding with a context-aware representation which is typically implemented by self-attention mechanism. For current word w_t, the input of the recurrent layer at time step t is represented as x_t:

$$e_t^a = \text{S-Att}([e_t^c, e_t^w], E) \tag{6}$$

$$x_t = [e_t^c, e_t^w, e_t^a] \tag{7}$$

e_t^a is the context-aware vector of w_t which is obtained by applying self-attention mechanism on the concatenated embeddings $E = [e_1^c \parallel e_1^w, ..., e_T^c \parallel e_T^w]$.

It was difficult to train RNNs to capture long-term dependencies because the gradients tend to either vanish or explode. Therefore, some more sophisticated activation functions with gating units were designed. We use LSTM (Hochreiter and Schmidhuber, 1997) in this work:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{8}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{9}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{10}$$

$$\widetilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{11}$$

$$c_t = i_t \odot \widetilde{c}_t + f_t \odot c_{t-1} \tag{12}$$

$$h_t = o_t \odot tanh(c_t) \tag{13}$$

Where $\odot$ denotes element-wise product of two vectors. To consider both the previous history and the future history, we use BiLSTM as encoder in advance. The bi-directional LSTM (BiLSTM), a modification of the LSTM, consists of a forward and a backward LSTM. The encoder reads the input vectors $\mathbf{x} = (x_1, x_2, ..., x_T)$ and generates T hidden states by concatenating the forward and backward hidden states of BiLSTM:

$$\overrightarrow{h}_t = \overrightarrow{LSTM}(x_t, \overrightarrow{h}_{t-1}) \tag{14}$$

$$\overleftarrow{h}_t = \overleftarrow{LSTM}(x_t, \overleftarrow{h}_{t+1}) \tag{15}$$

$$h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t] \tag{16}$$

where $\overleftarrow{h}_t$ is the hidden state of backward pass in BLSTM and $\overrightarrow{h}_t$ is the hidden state of forward pass in BLSTM at time t.

3.4 Intent-Augmented Gating Mechanism

As described above, intent information is useful for slot filling task. To measure the probability of words in target slots and attend to the ones relevant to the intent, we add a gate to the output of BiLSTM layer. Let $H \in \mathbb{R}^{2d \times T}$ be a matrix consisting of hidden vectors $[h_1, ..., h_T]$ produced by BiLSTM. For each word, we use self-attention mechanism to form another context-aware representation, the gate vector h_t^* is calculated by linearly transforming the concatenation of the context-aware representation and the intent embedding vector v^{int} with a multi-layer perceptron (MLP) network. The intent label is provided by correct label during training phase, and by the output from intent classification layer in the test phase. Specifically, for $t = 1, ...T$:

$$s_t = \text{Self-Attention}(h_t, H) \tag{17}$$

$$h_t^* = \text{MLP}([s_t, v^{int}]) \tag{18}$$

$$o_t = h_t \odot h_t^* \tag{19}$$

We use element-wise multiplication to model the interaction between BiLSTM outputs and the gate vector.

3.5 Task Learning

The bidirectional recurrent layer converts a sequence of words $\mathbf{w} = (w_1, w_2, ..., w_T)$ into hidden states $H = [h_1, ..., h_T]$ which are shared by two tasks. We use simple attention pooling function denoted as f_{att} over H to get an attention-sum vector for intent label classification. The classified label y^{int} is transformed to an embedding v^{int} by matrix E^{int} for gate computing.

$$h^{int} = f_{att}(H) \tag{20}$$

$$y^{int} = \text{softmax}(W^{int} h^{int} + b^{int}) \tag{21}$$

During the training phase, model parameters are updated w.r.t. a cross-entropy loss between the predicted probabilities and the true label. The label with maximum probability will be selected as the predicted intent during the testing phase.

For another task, the hidden states processed by our gating layer are used for predicting slot labels.

$$y_t^{slot} = \text{softmax}(W^{slot} o_t + b^{slot}) \tag{22}$$

Slot filling can be defined as a sequence labelling problem which is to map a utterance sequence $\mathbf{w} = (w_1, ..., w_T)$ to its corresponding slot label sequence $\mathbf{y} = (y_1, ..., y_T)$. The objective is to maximize the likelihood of a sequence:

$$P(\mathbf{y}^{slot}|\mathbf{w}) = \prod_{t=1}^{T} P(y_t^{slot}|\mathbf{w}) \quad (23)$$

It is equal to minimize Negative Log-likelihood (NLL) of the correct labels for the predicted sequence $\mathbf{y}^{slot}$.

4 Experiments

4.1 Dataset

In order to evaluate the efficiency of our proposed model, we conduct experiments on ATIS (Airline Travel Information Systems) dataset, which is widely used as benchmark in SLU research (Price, 1990).

Figure 3 gives one example of sentence in ATIS dataset. The words are labelled with their value according to certain semantic frames. The slot labels of the words are represented in an In-Out-Begin (IOB) format and the intent is highlighted with a box surrounding it.

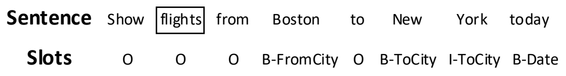

Sentence	Show	flights	from	Boston	to	New	York	today
Slots	O	O	O	B-FromCity	O	B-ToCity	I-ToCity	B-Date

Figure 3: Example of sentence annotated by slots sampled from ATIS corpus, the black boxed word indicates the intent.

In this paper, we use the ATIS corpus setting following previous related works (Liu and Lane, 2016a; Mesnil et al., 2015; Liu and Lane, 2015; Xu and Sarikaya, 2013; Tur et al., 2010). The training set contains 4978 utterances from ATIS-2 and ATIS-3 datasets, and test set contains 893 utterances from ATIS-3 NOV93 and DEC94 datasets. The number of slot labels is 127 and the intent has 18 different types.

4.2 Metrics

The performance of slot filling task is measured by the F1-score, while intent detection task is evaluated with prediction error rate that is the ratio of the incorrect intent of the test data.

4.3 Training Details

We preprocess the ATIS following (Yao et al., 2013; Liu and Lane, 2016a). To deal with unseen words in the test set, we mark those words that appear only once in the training set as ⟨UNK⟩, and use this label to represent those unseen words in the test set. Besides, each number is converted to the string DIGIT.

The model is implemented in the Tensorflow framework (Abadi et al., 2016). At training stage, we use LSTM cell as suggested in (Sutskever et al., 2014) and the cell dimension d is set to be 128 for both the forward and backward LSTM.

We set the dimension of word embedding d_w to be 64 and the dimension of character embedding d_c to be 128. We generate three character-level embeddings using multiple widths and filters (the convolution kernel width $w \in \{2, 3, 4\}$ with 64 filters each) followed by a max pooling layer over time. Then, the dimension of concatenated embeddings is 256. We make the dimensions of each parts equal for the convenience of dimension splitting during the self-attention in later stage. All the parameters in the network are randomly initialized with uniform distribution (Sussillo and Abbott, 2014) which are fine-tuned during training. We use the stochastic gradient descent algorithm (SGD) for updating parameters. And the learning rate is controlled by Adam algorithm (Kingma and Ba, 2014). The model is trained on all the training data with mini-batch size of 16. In order to enhance our model to generalize well, the maximum norm for gradient clipping is set to 5. We also apply layer normalization (Ba et al., 2016) on the self-attention layer after we add a residul connection between the output and input. Meanwhile, dropout rate 0.5 is applied on recurrent cell projection layer (Zaremba et al., 2014) and on each attention activation.

4.4 Independent Learning

The results of separate training for slot filling and intent detection are reported in Table 1 and Table 2 respectively. On the independent slot filling task, we fixed the intent information as the ground truth labels in the dataset. But on the independent intent detection task, there is no interaction with slot labels.

Table 1 compares F1-score of slot filling between our proposed architecture and some previous works. Our model achieves state-of-the-art results and outperforms previous best model by 0.56% in terms of F1-score. We attribute the improvement of our model to the following reasons: 1) The attention used in (Liu and Lane, 2016a) is vanilla attention, which is used to compute the de-

Methods	F1-score
CRF (Mesnil et al., 2013)	92.94
simple RNN (Yao et al., 2013)	94.11
CNN-CRF (Xu and Sarikaya, 2013)	94.35
LSTM (Yao et al., 2013)	94.85
RNN-SOP (Liu and Lane, 2015)	94.89
Deep LSTM (Yao et al., 2013)	95.08
RNN-EM (Peng et al., 2015)	95.25
Bi-RNN with Ranking Loss (Vu et al., 2016)	95.47
Encoder-labeler Deep LSTM (Kurata et al., 2016)	95.66
Attention BiRNN (Liu and Lane, 2016a)	95.75
BLSTM-LSTM (focus) (Zhu and Yu, 2017)	95.79
Our Model	**96.35**

Table 1: Results of independent training for slot filling in terms of F1-score.

Methods	Error(%)
Recursive NN (Guo et al., 2014)	4.60
Boosting (Tur et al., 2010)	4.38
Boosting + Simplified sentences (Tur et al., 2011)	3.02
Attention Enc-Dec (Liu and Lane, 2016a)	**2.02**
Our Model	2.69

Table 2: Results of independent training for intent detection in terms of error rate.

coding states. It is not suitable for our model since the embeddings are composed of several parts. Self-attention allows the model to attend to information jointly from different representation parts, so as to better understand the utterance. 2) intent-augmented gating layer connects the semantics of sequence slot labels, which captures complex interactions between the two tasks.

Table 2 compares the performance of our proposed model to previously reported results on intent detection task. Our model gives good performance in terms of classification error rate, but not as good as Attention Encoder-Decoder (with aligned inputs) method (Liu and Lane, 2016a). As their published state-of-the-art result described in (Liu and Lane, 2016a), their attention-based model is based on word-level embeddings. While in our model, we introduce character-level embeddings to improve the performance of joint learning. But independent learning for intent classification aims at capturing the global information of an utterance, not caring much about the details of specific word. The character-level embeddings introduced in our model bring very little hurt to independent learning of intent detection, as a trade-off in performance between both criterion.

4.5 Joint Learning

We compare our model against the following baseline models based on joint learning:

- **Recursive NN:** (Guo et al., 2014) employed recursive neural networks for joint training of two tasks.
- **Recursive NN + Viterbi:** (Guo et al., 2014) applied the Viterbi algorithm on Recursive NN to improve the result on slot filling.
- **Attention Enc-Dec:** (Liu and Lane, 2016a) proposed Attention Encoder-Decoder (with aligned inputs) which introduced context vector as the explicit aligned inputs at each decoding step.
- **Attention BiRNN:** (Liu and Lane, 2016a) introduced attention to the alignment-based RNN sequence labeling model. Such attention provides additional information to the intent classification and slot label prediction.

Methods	F1	Error(%)
Recursive NN (Guo et al., 2014)	93.22	4.60
Recursive NN+Viterbi (Guo et al., 2014)	93.96	4.60
Attention Enc-Dec (Liu and Lane, 2016a)	95.87	1.57
Attention BiRNN (Liu and Lane, 2016a)	95.98	1.79
Our Model	**96.52**	**1.23**

Table 3: Results of joint training for slot filling and intent detection.

Table 3 compares our joint model with reported results from previous works. We can see that our model achieves state-of-the-art results and outperforms previous best result by 0.54% in terms of F1-score on slot filling, and by 0.34% in terms of error rate on intent detection. This improvement is statistically significant. Besides, the joint learning

Methods	F1-Score	Error(%)
W/O char-embedding	96.30	1.23
W/O self-attention	96.26	1.34
W/O attention-gating	96.25	1.46
Full Model	**96.52**	**1.23**

Table 4: Feature ablation comparison of our proposed model on ATIS. slot filling and intent detection result are shown each row after after we exclude each feature from the full architecture

achieves better results than separate learning. It can be interpreted that the two tasks are highly correlated and boost the performance each other. The slot filling task enables the model to learn more meaningful representations which give more supervisory signals for the learning of shared parameters. Similarly, intent is also useful to determine the slot label.

4.6 Ablation Study

The ablation study is performed to evaluate whether and how each part of our model contributes to our full model. To further evaluate the advances of our gating architecture for joint learning, we ablate some techniques used in our model. We ablate three important components and conduct different approaches in this experiment. Note that all the variants are based on joint learning with intent-augmented gate:

- W/O char-embedding, where no character embeddings are added to the embedding layer. The embedding layer is composed of word embeddings only.
- W/O self-attention, where no self-attention is modelled after the embedding layer and in the intent-augmented gating layer. The intent gate is computed by the output of BiLSTM and intent embedding.
- W/O attention-gating, where no self-attention mechanism is performed in the intent-augmented gating layer. The gate is computed by the output of BiLSTM and intent embedding. But we still use the self-attention on top of embedding layer to augment the context information.

Table 4 shows the joint learning performance of our model on ATIS data set by removing one module at a time. We find that all variants of our model perform well based on our gate mechanism. As listed in the table, all features contribute to both slot filling and intent classification task.

If we remove the self-attention from the holistic model or just in the intent-augmented gating layer, the performance drops dramatically. The result can be interpreted that self-attention mechanism computes context representation separately and enhances the interaction of features in the same aspect. We can see that self-attention does improve performance a lot in a large scale, which is consistent with findings of previous work (Vaswani et al., 2017; Lin et al., 2017).

If we remove character-level embeddings and only use word-level embeddings, we see 0.22% drop in terms of F1-score. Though word-level embeddings represent the semantics of each word, character-level embeddings can better handle the out-of-vocabulary (OOV) problem which is essential to determine the slot labels.

5 Conclusion

In this paper, we propose a novel self-attentive model gated with intent for spoken language understanding. We apply joint learning on both intent detection and slot filling tasks. In our model, self-attention mechanism is introduced to better represent the semantic of utterance, and gate mechanism is introduced to make full use of the semantic correlation between slot and intent. Experiment results on ATIS dataset have shown efficiency of our model and outperforms the state-of-the-art approach on both tasks. Besides, our model also shows consistent performance gain over the independent training models. In future works, we plan to improve our model by introducing extra knowledge.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Renato De Mori. 2007. Spoken language understanding: A survey. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 365–376. IEEE.

Anoop Deoras and Ruhi Sarikaya. 2013. Deep belief network based semantic taggers for spoken language understanding. In *Interspeech*, pages 2713–2717.

Allen L Gorin, Giuseppe Riccardi, and Jeremy H Wright. 1997. How may i help you? *Speech communication*, 23(1):113–127.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 554–559. IEEE.

Patrick Haffner, Gokhan Tur, and Jerry H Wright. 2003. Optimizing svms for complex call classification. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–I. IEEE.

Stefan Hahn, Marco Dinarelli, Christian Raymond, Fabrice Lefevre, Patrick Lehnen, Renato De Mori, Alessandro Moschitti, Hermann Ney, and Giuseppe Riccardi. 2011. Comparing stochastic approaches to spoken language understanding in multiple languages. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1569–1583.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *INTERSPEECH*, pages 715–719.

Charles T Hemphill, John J Godfrey, George R Doddington, et al. 1990. The atis spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*, pages 96–101.

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

Dan Jurafsky. 2000. *Speech & language processing*. Pearson Education India.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentencelevel information with encoder lstm for natural language understanding. *arXiv preprint*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Bing Liu and Ian Lane. 2015. Recurrent neural network structured output prediction for spoken language understanding. In *Proc. NIPS Workshop on Machine Learning for Spoken Language Understanding and Interactions*.

Bing Liu and Ian Lane. 2016a. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Bing Liu and Ian Lane. 2016b. Joint online spoken language understanding and language modeling with recurrent neural networks. *arXiv preprint arXiv:1609.01462*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(3):530–539.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, pages 3771–3775.

Baolin Peng, Kaisheng Yao, Li Jing, and Kam-Fai Wong. 2015. Recurrent neural networks with external memory for spoken language understanding. In *Natural Language Processing and Chinese Computing*, pages 25–35. Springer.

Patti J Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Suman V Ravuri and Andreas Stolcke. 2015. Recurrent neural network and lstm models for lexical utterance classification. In *INTERSPEECH*, pages 135–139.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Eighth Annual Conference of the International Speech Communication Association*.

Robert E Schapire and Yoram Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168.

Edwin Simonnet, Nathalie Camelin, Paul Deléglise, and Yannick Estève. 2015. Exploring the use of attention-based recurrent neural networks for spoken language understanding. In *Machine Learning for Spoken Language Understanding and Interaction NIPS 2015 workshop (SLUNIPS 2015)*.

David Sussillo and LF Abbott. 2014. Random walk initialization for training very deep feedforward networks. *arXiv preprint arXiv:1412.6558*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? In *Spoken Language Technology Workshop (SLT), 2010 IEEE*, pages 19–24. IEEE.

Gokhan Tur, Dilek Hakkani-Tür, Larry Heck, and Sarangarajan Parthasarathy. 2011. Sentence simplification for spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5628–5631. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016. Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6060–6064. IEEE.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 78–83. IEEE.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014a. Spoken language understanding using long short-term memory neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 189–194. IEEE.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. 2014b. Recurrent conditional random field for language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4077–4081. IEEE.

Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. 2013. Recurrent neural networks for language understanding. In *Interspeech*, pages 2524–2528.

Steve J Young. 2002. Talking to machines (statistically speaking). In *INTERSPEECH*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*, pages 2993–2999.

Su Zhu and Kai Yu. 2017. Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5675–5679. IEEE.

Learning End-to-End Goal-Oriented Dialog with Multiple Answers

Janarthanan Rajendran*
University of Michigan
`rjana@umich.edu`

Jatin Ganhotra*
IBM Research
`jatinganhotra@us.ibm.com`

Satinder Singh
University of Michigan
`baveja@umich.edu`

Lazaros Polymenakos
IBM Research
`lcpolyme@us.ibm.com`

Abstract

In a dialog, there can be multiple valid next utterances at any point. The present end-to-end neural methods for dialog do not take this into account. They learn with the assumption that at any time there is only one correct next utterance. In this work, we focus on this problem in the goal-oriented dialog setting where there are different paths to reach a goal. We propose a new method, that uses a combination of supervised learning and reinforcement learning approaches to address this issue. We also propose a new and more effective testbed, permuted-bAbI dialog tasks [1] by introducing multiple valid next utterances to the original-bAbI dialog tasks, which allows evaluation of goal-oriented dialog systems in a more realistic setting. We show that there is a significant drop in performance of existing end-to-end neural methods from 81.5% per-dialog accuracy on original-bAbI dialog tasks to 30.3% on permuted-bAbI dialog tasks. We also show that our proposed method improves the performance and achieves 47.3% per-dialog accuracy on permuted-bAbI dialog tasks.

1 Introduction

End-to-end, neural conversation models that learn from chatlogs of human-to-human interaction hold the promise of quickly bootstrapping dialog systems and keep evolving them based on new data. Recent work ((Vinyals and Le, 2015; Bordes et al., 2016; Serban et al., 2016)) has shown that dialog models can be trained in an end-to-end manner with satisfactory results.

However, human dialog has some unique properties that many other learning tasks do not. For any given dialog state or input, multiple correct next utterances or answers may be possible; i.e given the dialog so far, there are several different utterances which one can say next that would be valid. However, this property of dialog is not taken into account in the present way of training end-to-end neural dialog systems.

There are two broad ways in which present day neural dialog systems can be trained: Supervised Learning (SL) and Reinforcement Learning (RL). In the RL setting, the dialog system learns through trial and error with reinforcement (rewards at the end or at key dialog points) from a human or a simulator. RL training for dialog is a hard problem to solve. It is difficult to define and award appropriate rewards, and to learn language from scratch through these rewards. RL training also demands a large amount of training interaction. In order to handle these challenges, RL methods are almost always complimented with a SL phase.

In SL setting, a fixed set of dialog data is collected from humans and the dialog system is trained to imitate that data. When a new dialog dataset is curated, the data is extracted from real-world chat logs from human-human conversation, where one human acts as the agent. It is not possible to know all of the valid next utterances for a given dialog state at any single time. A particular dialog in the dataset has access to only one of the valid next utterances given the dialog history and the current utterance. Another valid next utterance could be present in some other dialog in the dataset.

Since for a given dialog only one correct answer is available at any single time, the gradients are calculated based on the assumption that there is only one correct next utterance for the given dialog

---

* Equal Contribution

[1] permuted-bAbI-dialog-tasks - `https://github.com/IBM/permuted-bAbI-dialog-tasks`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3834–3843
Brussels, Belgium, October 31 - November 4, 2018.

state. This results in reducing the probability of other valid next utterances for that dialog. While all this is true for dialog in general, in this work, we focus on the goal-oriented dialog setting.

We propose a novel method which handles the issue of learning multiple possibilities for completing a goal-oriented dialog task. Our presentation is organized as follows: In Sections 2 and 3, we define the multiple-utterance problem and point out the limitations of current learning methodologies.

Section 4 describes our proposed method, which combines Supervised Learning and Reinforcement Learning approaches for handling multiple correct next utterances. In Section 5, we introduce permuted-bAbI dialog tasks, which is our proposed testbed for goal-oriented dialog. Section 6 details our experimental results across all datasets and all models.

2 Multiple-utterance problem in goal-oriented dialog

Goal-oriented dialog tasks are those in which there is an explicit goal that the system tries to achieve through the dialog. These tasks typically involve getting some information from the user, interacting with an external Knowledge Base (KB) and giving back information to the user. Simple examples (form filling) include restaurant reservation, hotel booking etc., whereas complex tasks could involve a combination of informative and form filling tasks (e.g. IT support, customer care etc.).

There could be multiple ways/strategies to achieve a given task. When a dataset is collected from different people performing the task, these different ways of solving the task get reflected in it. These variations could be as simple as difference in the order in which the system asks the information from the user, or as complex as following a completely different line of questions/answers to achieve the task. For example, in an IT support scenario, one may ask a sequence of standard questions or start from common problems and once eliminated, follow the standard set of questions. In the dataset, they turn into multiple valid next utterances for a given dialog so far.

Our objective in this paper is not to mimic all humans from whom the data was collected from or all the possible strategies, but rather to use that knowledge and learn to perform the task better and faster.

3 Issues with the present methods

Consider a goal-oriented dialog dataset for restaurant reservation where the dialog system has to acquire cuisine, location, number of people and price range information from the user before retrieving restaurant options. Consider two dialogs (A and B) in the dataset which have the same first system utterance (`S1a` is same as `S1b`). Let their dialog state vector after encoding the dialog until `S1` be s. This state vector s is what will be used for next utterance generation or retrieval. Their next utterance is different because of the variation in the order in which the information is asked from the user as shown below. i.e `S2a` is different from `S2b`.

Dialog A
U: Hello !
`S1a: Hello, what can I help you with today?`
U: Can you book a table for six people with French food?
`S2a: Where should it be?`
U: In Bombay
`S3a: Which price range are you looking for?`
U: I am looking for a cheap restaurant.
`S4a: Ok, let me look into some options for you.`

Dialog B
U: Hello !
`S1b: Hello, what can I help you with today?`
U: Can you book a table for six people with French food?
`S2b: Which price range are you looking for?`
U: I am looking for a cheap restaurant.
`S3b: Where should it be?`
U: In Bombay
`S4b: Ok, let me look into some options for you.`

These two dialogs might be present in different places in the dataset. When dialog A is part of the batch for which loss is calculated and parameters are updated, the dialog system is asked to produce `S2a` from s. Here, the loss could be negative log-likelihood, squared error or anything that tries to push s towards producing `S2a`. In this process, the probability of the dialog system producing `S2b`, an equally valid answer, is reduced. The

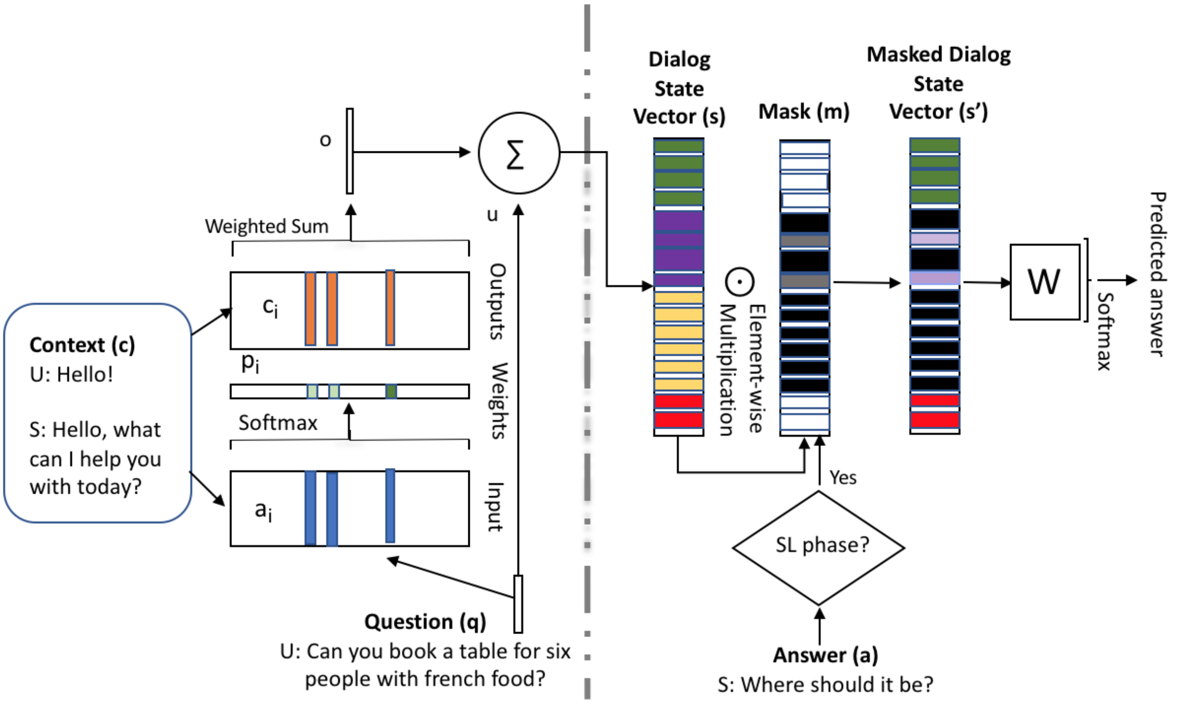

Figure 1: **Mask-memN2N** - *Left*: A single layer version of memN2N. *Right*: Masking

reverse happens when the dialog system encounters a batch with dialog B. This is true whether a softmax or a sigmoid non-linearity for each unit is used in the output layer. In essence, the system is expected to learn a one to many function, but is forced to produce only one of the valid outputs at one time (that is all we have at any one time), while telling that all other outputs are wrong.

Note that this could be a problem even when the dialogs are similar (semantically) in the beginning, but not the same exact dialog. For simplicity, we show an example where two dialogs have the same beginning and only 2 valid next utterances occur.

4 Proposed Method

The proposed method has two phases. In one phase, the dialog system tries to learn how to perform dialog from the dataset by trying to mimic it and in the other it learns through trial and error. The former uses supervised learning and the latter uses reinforcement learning. Consider a dialog state vector s. This has all the information from the dialog so far and is used for next utterance generation or retrieval. Any neural method such as memory network (Weston et al., 2014), HRED (Sordoni et al., 2015) etc. can be used for encoding and producing the dialog state vector s. As discussed earlier for the state vector s, there could be multiple valid next utterances.

During the SL phase, at each data point, the dialog system is trained to produce the one next utterance provided in that data point and is penalized even if it produces one of the other valid next utterances. We avoid this by providing the dialog system, the ability to use only parts of the state vector to produce that particular next utterance. This allows only parts of the network to be affected that were responsible for the prediction of that particular answer. The dialog system can retain other parts of the state vector and values in the network that stored information about other valid next utterances. This is achieved by generating a mask vector m which decides which parts of the state vector s should be used for producing that particular answer. This is achievable, as m is learned as function of s and the actual answer a present in the given dialog data point.

In the RL phase, however, the dialog system is rewarded if it produces an answer that is among any of the set of valid correct answers. While in the SL phase the dialog system had access to the actual answer a at given time to produce the mask, in the RL phase the dialog system produces the mask by only using the dialog state vector s.

$$\text{Supervised Learning phase} \tag{1}$$

$$m = \sigma(W_s s + W_a a + b_{sl})$$

$$s' = m * s$$

$$\text{Reinforcement Learning phase} \tag{2}$$

$$m = \sigma(W_s s + W_r s + b_{rl})$$

$$s' = m * s$$

where W's and b's are the parameters learned, σ is an element wise sigmoid non-linearity and s' is the masked dialog state vector that is used by the dialog system to perform the downstream task such as next utterance generation or retrieval. The parameters of the network that produce s and that follow s' are shared between the two phases. While there are different ways of combining the two phases during training, the RL phase which does not use the answer for its mask is what is used during testing. The masking approach described above is illustrated in Fig 1.

In this work, SL phase is performed first, followed by RL phase. In the SL phase, the dialog system learns different dialog responses and behaviours from the dataset. It has the ability to learn multiple possible next utterances without one contradicting/hindering the learning of the other much. In the RL phase, the dialog system might settle on a unique behaviour that it finds best for it to perform the task and uses that during test time as well.

5 Permuted bAbI dialog tasks

Bordes et al. (2016) proposed bAbI dialog tasks as a testbed to break down the strengths and shortcomings of end-to-end dialog systems in goal-oriented applications. There are five tasks, generated by a simulation set in the context of restaurant reservation, with the final goal of booking a table. The simulation is based on an underlying KB, whose facts contain restaurants and their properties. Tasks 1 (Issuing API calls) and 2 (Updating API calls) test the dialog system to implicitly track dialog state, whereas Task 3 (Displaying options) and 4 (Providing extra information) check if the system can learn to use KB facts in a dialog setting. Task 5 (Conducting full dialogs) combines all tasks.

(Bordes et al., 2016) used natural language patterns to create user and system utterances. There are 43 patterns for the user and 20 for the system, which were combined with the KB entities to form thousands of different utterances. However, on a closer analysis of the testbed, we observe that even though there are thousands of different utterances, these utterances always follow a fixed deterministic order (predefined by the simulation), which makes the tasks easier and unsuitable to mimic conversations in the real-world. For example, for Task 1, the system follows a predefined order to ask for missing fields required to complete the dialog state. In Task 3, all restaurants retrieved have a unique and different rating. While this makes evaluation deterministic and easier, these hidden settings in the simulation create conversations, which are simpler compared to real-world conversations for restaurant reservation.

We propose permuted-bAbI dialog tasks, an extension of original-bAbI dialog tasks, which make our proposed testbed more appropriate for evaluating dialog systems in goal-oriented setting. In *original-bAbI dialog tasks* at a given time in the conversation, there is **only one** correct system utterance. *Permuted-bAbI dialog tasks* allow **multiple** correct system utterances at a given point in the conversation.

We propose the following changes to original-bAbI dialog tasks. In Task 1, a user request defines a query that can contain from 0 to 4 of the required fields to make a reservation. The system asks questions to fill the missing fields and eventually generate the correct corresponding API call. However, the system asks for information in a *deterministic* order -
Cuisine → *Location* → *People* → *Price*
to complete the missing fields. In permuted-bAbI dialog tasks, we don't follow a deterministic order and allow the system to ask for the missing fields in any order.

In Task 3, for the API call matching the user request, the facts are retrieved from the KB and provided as part of dialog history. The system must propose options to users by listing the restaurant names sorted by their corresponding rating (from higher to lower) until users accept. However, each restaurant has a different rating. In permuted-bAbI dialog tasks, multiple restaurants can have the same rating but the system must still propose the restaurant names following the decreasing order of rating, which allows multiple valid next utterances.

In Task 5, Tasks 1-4 are combined to generate

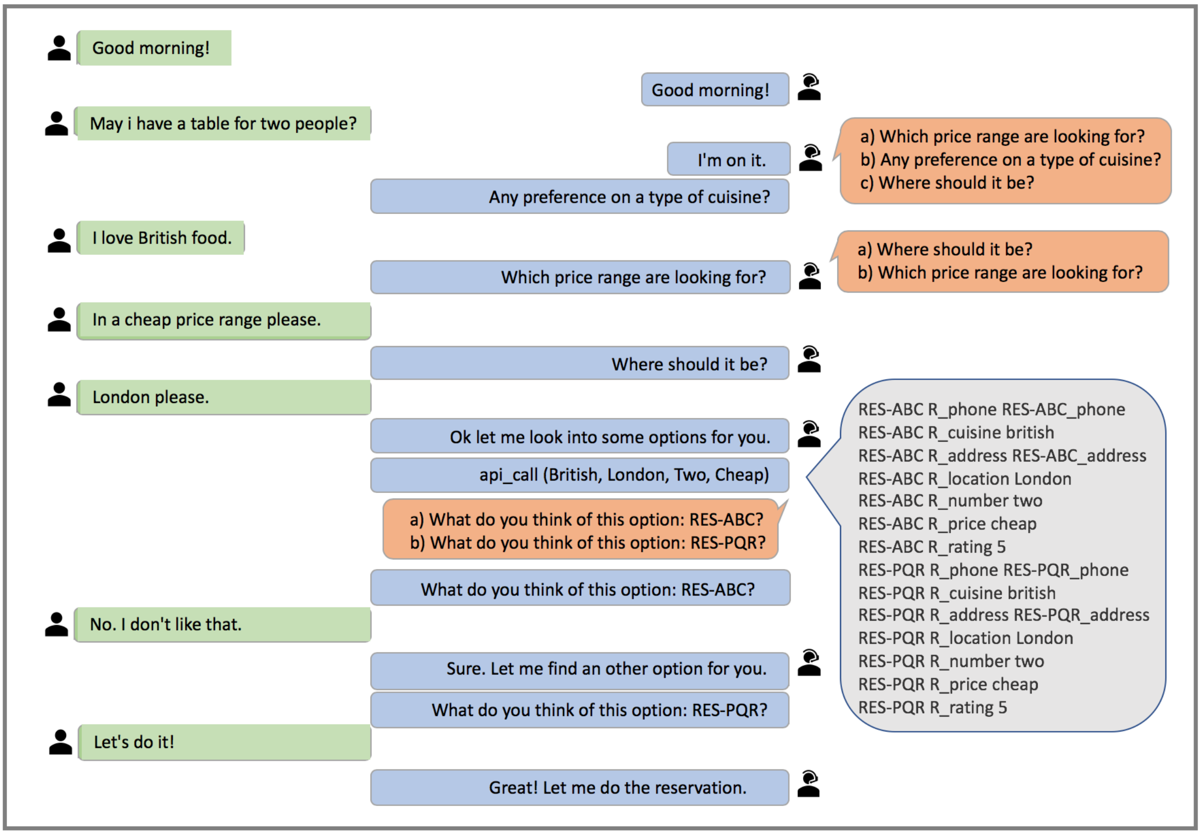

Good morning!

Good morning!

May i have a table for two people?

I'm on it.

Any preference on a type of cuisine?

a) Which price range are looking for?
b) Any preference on a type of cuisine?
c) Where should it be?

I love British food.

Which price range are looking for?

a) Where should it be?
b) Which price range are looking for?

In a cheap price range please.

Where should it be?

London please.

Ok let me look into some options for you.

api_call (British, London, Two, Cheap)

RES-ABC R_phone RES-ABC_phone
RES-ABC R_cuisine british
RES-ABC R_address RES-ABC_address
RES-ABC R_location London
RES-ABC R_number two
RES-ABC R_price cheap
RES-ABC R_rating 5
RES-PQR R_phone RES-PQR_phone
RES-PQR R_cuisine british
RES-PQR R_address RES-PQR_address
RES-PQR R_location London
RES-PQR R_number two
RES-PQR R_price cheap
RES-PQR R_rating 5

a) What do you think of this option: RES-ABC?
b) What do you think of this option: RES-PQR?

What do you think of this option: RES-ABC?

No. I don't like that.

Sure. Let me find an other option for you.

What do you think of this option: RES-PQR?

Let's do it!

Great! Let me do the reservation.

Figure 2: **Permuted-bAbI dialog tasks.** A user (in green) chats with a dialog system (in blue) to book a table at a restaurant. At a given point in the dialog, the dialog system has multiple correct next utterances (in orange). The dialog system can choose either of the multiple correct utterances as the next utterance. The list of restaurants are returned from the API_call (in grey) also contain multiple restaurants with the same rating, giving the dialog system more options to propose to the user.

full dialogs. In permuted-bAbI dialog tasks, we incorporate the changes for both Task 1 and Task 3 mentioned above to the final Task 5 (Conducting full dialogs).

Fig 2 shows a dialog sample from permuted-bAbI dialog tasks. Our proposed testbed (We release and show experiments on permuted version of Task 5, i.e. Conducting full dialogs set, as tasks 1-4 are subsets of a full conversation and don't represent a complete meaningful conversation standalone) is more closer to a real-world restaurant reservation conversation, in comparison to the original-bAbI dialog tasks. We release two versions of permuted-bAbI dialog tasks - permuted-bAbI dialog task* (the full dataset with all permutations. There are around 11,000 dialogs in each set and the exact number varies for train, val, test and test-OOV sets), which contains all permutations (the full dataset) and permuted-bAbI dialog task, which contains 1000 dialogs randomly sampled (we used random seed = 599 for sampling) from permuted-bAbI dialog task*. We choose a random 1000 subset from each of train, val, test and test-OOV sets to match the number of dialogs in original-bAbI dialog task. Another key point to choose a small subset and to not include all permutations in the training set is that it allows to mimic real-world data collection. For a real-world use-case, as the number of required fields and user options increase, the cost for gathering data covering all permutations will increase exponentially, and one can't guarantee that enough training examples for all permutations will be present in the collected dataset. Note that, since there are multiple correct next utterances, we also modify the evaluation criteria so that the system is rewarded if it predicts any of the multiple correct next utterances.

6 Experiments and Results

End-to-end memory networks (Sukhbaatar et al., 2015) are an extension of Memory Networks pro-

Dataset	no match-type		+ match-type	
	Per-turn	Per-dialog	Per-turn	Per-dialog
Original-bAbI dialog task	98.5	77.1	98.8	81.5
Permuted-bAbI dialog task*	96.4	58.2	96.9	63.9
Permuted-bAbI dialog task	91.8	22	93.3	30.3
OOV: Original-bAbI dialog task	65.6	0	78.3	0
OOV: Permuted-bAbI dialog task*	63.6	0	78.4	0
OOV: Permuted-bAbI dialog task	63.4	0.5	78.1	0.6

Table 1: **Test results for our baseline end-to-end memory network model across the three datasets.** Results (accuracy %) are given in the standard setup and out-of-vocabulary (OOV) setup. Results are given for both with and without match-type features.

posed by (Weston et al., 2014) which have been successful on various natural language processing tasks. End-to-end memory networks are trained end-to-end and use a memory component to store dialog history and short-term context to predict the required response. They perform well on original-bAbI dialog tasks and have been shown to outperform some other end-to-end architectures based on Recurrent Neural Networks. Hence, we chose them as end-to-end model baseline. We perform experiments on the three datasets mentioned above across all our models. We also perform experiments with match-type features proposed by (Bordes and Weston, 2016b), which allow the model to use type-information for entities like location, cuisine, phone number etc. The results for our baseline model, our proposed model and results on our ablation study are described below. The test results reported are calculated by choosing the model with highest validation per-turn accuracy across multiple runs.

6.1 Baseline model: memN2N

A single layer version of the memN2N model is shown in Fig.1. A given sentence (i) from the context (dialog history) is stored in the memory by it's input representation (a_i). Each sentence (i) also has a corresponding output representation (c_i). To identify the relevance of a memory for the next-utterance prediction, attention of query over memory is computed via dot product, where (p_i) represents the probability for each memory in equation 3. An output vector (o) is computed by a weighted sum of the memory embeddings (c_i) with their corresponding probabilities in equation 4. The output vector (o) represents the overall embedding for the context. The output vector (o) and query (u) added together represent the dialog state vector (s) in equation 5.

$$p_i = \text{Softmax}(u^T(a_i)) \tag{3}$$

$$o = \sum_i p_i c_i \tag{4}$$

$$s = (o + u) \tag{5}$$

Our results for our baseline model across the three datasets are given in Table 1. The hyperparameters used for training the baseline models are provided in Appendix A.1.

The first 3 rows show the results for the three datasets in the standard setup, and rows 4-6 show results in the Out-Of-Vocabulary (OOV) setting. Per-response accuracy counts the percentage of responses that are correct (i.e., the correct candidate is chosen out of all possible candidates). Note that, as mentioned above in Section 5, since there are multiple correct next utterances, a response is considered correct if it predicts any of the multiple correct next utterances. Per-dialog accuracy counts the percentage of dialogs where every response is correct. Therefore, even if only one response is incorrect, this might result in a failed dialog, i.e. failure to achieve the goal of restaurant reservation. We report results both with and without match type features, shown in the last two columns.

From Table 1, we observe that the baseline model performs poorly on permuted-bAbI dialog tasks (both full dataset and 1000 random dialogs). For permuted-bAbI dialog task*, the baseline model achieves 58.2% on per-dialog accuracy, but the number decreases to only 22% for permuted-bAbI dialog task (1000 random dialogs). This implies that only 1 out of every 4 dialogs might be successful in completing the goal. The results improve slightly by using match-type features, but 30% per-dialog accuracy is still

Model	no match-type		+ match-type	
	Per-turn	Per-dialog	Per-turn	Per-dialog
memN2N	91.8	22	93.3	30.3
memN2N + all-answers†	88.5	14.9	92.5	26.4
Mask-memN2N	93.4	**32**	95.2	**47.3**
OOV: memN2N	63.4	0.5	78.1	0.6
OOV: memN2N + all-answers†	60.8	0.5	74.9	0.6
OOV: Mask-memN2N	63.0	0.5	80.1	1

Table 2: **Test results for various models on permuted-bAbI dialog task.** Results (accuracy %) are given in the standard setup and OOV setup; and both with and without match-type features.

very low for real-world use. These results clearly demonstrate that the end-to-end memory network model does not perform well on our proposed testbed, which is more realistic and mimics real-world conversations more closely.

6.2 Mask End-to-End Memory Network: Mask-memN2N

Our model, Mask-memN2N, shown in Fig 1, is built on the baseline memN2N model described above, except for an additional masking performed to the dialog state vector. The SL phase is performed for the first 150 epochs. The best performing model chosen based on validation accuracy is used as a starting point for the RL phase. All parameters except for the network that produces the masks are shared between the two phases. During the SL phase, the mask parameters of the RL phase are pre-trained to match the mask produced in SL phase using an L2 loss. Through this approach, when the model transitions in the RL phase, it does not need to explore the valid masks and hence, the answers from scratch. Instead, its exploration will now be more biased towards relevant answers. For the RL phase, we use REINFORCE (Williams, 1992) for training the system. An additional loss term is added to increase entropy. The hyperparameters used, including the exact reward function are provided in Appendix A.2.

6.3 Model comparison

Our results for our proposed model and comparison with other models for permuted-bAbI dialog task are given in Table 2. Table 2 follows the same format as Table 1, except we show results for different models on permuted-bAbI dialog task. We show results for three models - memN2N, memN2N + all-answers and our proposed model, Mask-memN2N.

In the memN2N + all-answers model, we extend the baseline memN2N model and though not realistic, we provide information on all correct next utterances during training, instead of providing only one correct next utterance. The memN2N + all-answers model has an element-wise sigmoid at the output layer instead of a softmax, allowing it to predict multiple correct answers. This model serves as an important additional baseline, and clearly demonstrates the benefit of our proposed approach.

From Table 2, we observe that the memN2N + all-answers model performs poorly, in comparison to the memN2N baseline model both in standard setup and OOV setting, as well as with and without match-type features. This shows that the existing methods do not improve the accuracy of a dialog system even if all correct next utterances are known and used during training the model. Our proposed model performs better than both the baseline models. In the standard setup, the per-dialog accuracy increases from 22% to 32%. Using match-type features, the per-dialog accuracy increases considerably from 30.3% to 47.3%. In the OOV setting, all models perform poorly and achieve per-dialog accuracy of 0-1% both with and without match-type features. These results are similar to results for original-bAbI dialog Task 5 from Bordes and Weston (2016b) and our results with the baseline model.

Overall, Mask-memN2N is able to handle multiple correct next utterances present in permuted-bAbI dialog task better than the baseline models. This indicates that permuted-bAbI dialog task is a better and effective evaluation proxy compared to original-bAbI dialog task for real-world data. This also shows that we need better neural approaches, similar to our proposed model, Mask-

Model	Per-turn	Per-dialog
Mask-memN2N	93.4	32
Mask-memN2N (w/o entropy)	92.1	24.6
Mask-memN2N (w/o L2 mask pre-training)	85.8	2.2
Mask-memN2N (Reinforcement learning phase only)	16.0	0

Table 3: **Ablation study of our proposed model on permuted-bAbI dialog task.** Results (accuracy %) are given in the standard setup, without match-type features.

memN2N, for goal-oriented dialog in addition to better testbeds for benchmarking goal-oriented dialogs systems.

6.4 Ablation study

Here, we study the different parts of our model for better understanding of how the different parts influence the overall model performance. Our results for ablation study are given in Table 3. We show results for Mask-memN2N in various settings - a) without entropy, b) without pre-training mask c) reinforcement learning phase only.

Adding entropy for the RL phase seems to have improved performance a bit by assisting better exploration in the RL phase. When we remove L2 mask pre-training, there is a huge drop in performance. In the RL phase, the action space is large. In the bAbI dialog task, which is a retrieval task, it is all the candidate answers that can be retrieved which forms the action set. L2 mask pre-training would help the RL phase to try more relevant actions from the very start.

From Table 3 it is clear that the RL phase individually does not perform well; it is the combination of both the phases that gives the best performance. When we do only the RL phase, it might be very tough for the system to learn everything by trial and error, especially because the action space is so large. Preceding it with the SL phase and L2 mask pre-training would have put the system and its parameters at a good spot from which the RL phase can improve performance. Note that it would not be valid to check performance of the SL phase in the test set as the SL phase requires the actual answers for it to create the mask.

7 Related Work

End-to-end dialog systems have been trained to show satisfactory performance in goal-oriented setting, as shown by (Bordes et al., 2016) and (Seo et al., 2016)). The idea of allowing the system to learn to attend to different parts of the state vector at different times depending on the input that the proposed model uses has been used in different settings before. To name a few, Bahdanau et al. (2014) use it for Machine Translation (MT), where the MT system can attend to different words in the input language sentence while producing different words in the output language sentence. Xu et al. (2015) use it for image caption generation where the system attends to different parts of the image while generating different words in the caption. Madotto and Attardi (2017) use it for Question Answering (QA), where the QA system attends and updates different parts of the Recurrent Neural Network story state vector based on the sentence the system is reading in the input story.

In the past, goal-oriented dialog systems would model the conversation as partially observable Markov decision processes (POMDP) (Young et al. (2013)). However, such systems require many hand-crafted features for the state and action space representations, which limited their use only to narrow domains. In recent years, several corpora have been made available for building data-driven dialog systems (Serban et al., 2015). However, there are no good resources to train and test end-to-end models in goal-oriented scenarios.

Some datasets are proprietary (e.g., Chen et al. (2016)) or require participation to a specific challenge and signing a license agreement (e.g., DSTC4 (Kim et al., 2017)). Several datasets have been designed to train or test dialog state tracker components (Henderson et al. (2014), Asri et al. (2017), which are unsuitable for training end-to-end dialog systems, either due to limited number of conversations or due to noise. Recently, some datasets have been designed using crowdsourcing (Hixon et al. (2015), Wen et al. (2015), Su et al. (2015)) e.g., Amazon Mechanical Turk, CrowdFlower etc., but dialog systems built for them are harder to test automatically and involve another set of crowdsource workers for comparing them.

(Bordes et al., 2016) proposed goal-oriented bAbI dialog tasks, a testbed created from a simulation of restaurant reservation setting, which allows easy evaluation and interpretation of end-to-end dialog systems. However, the testbed doesn't mimic real world conversations and conversations are generated around a deterministic set of patterns. Recently, datasets designed using Wizard-of-Oz seting (Eric and Manning (2017), Wen et al. (2016)) show promise, but they are limited in scale, and evaluation metrics are either based around BLEU score and entity F1 scores or require crowdsource workers for evaluation.

8 Conclusion

We propose a method that uses masking to handle the issue of making wrong updates at different times because of the presence of multiple valid next utterances in a dataset, but having access to only one of them at any time. The method has a SL phase where the mask uses the answer as well, and an RL phase, where the system learns to generate the mask solely from its dialog state vector.

We modify the synthetic original-bAbI dialog task to create a more realistic and effective testbed, permuted-bAbI dialog task (which we have made publicly available), that has the issue of multiple next utterances as would be the case with any dataset created from human-human dialogs. Our experiments show that there is a significant drop in performance of the present neural methods in the permuted-bAbI dialog task compared to the original-bAbI dialog task. The experiments also confirm that the proposed method is a step in the right direction for bridging this gap.

A Appendix: Training Details

A.1 Baseline model: memN2N

The hyperparameters used for the baseline model are as follows: hops = 3, embedding_size = 20, batch_size = 32. The entire model is trained using stochastic gradient descent (SGD) (learning rate = 0.01) with annealing (anneal_ratio = 0.5, anneal_period = 25), minimizing a standard cross-entropy loss between $\hat{a}$ and the true label a. We add temporal features to encode information about the speaker for the given utterance (Bordes and Weston, 2016a) and use position encoding for encoding the position of words in the sentence (Sukhbaatar et al., 2015). We learn two embedding matrices A and C for encoding story, separate embedding matrix B for encoding query and weight matrices TA and TC for encoding temporal features. The same weight matrices are used for 3 hops[2].

A.2 Mask End-to-End Memory Network: Mask-memN2N

We use the same hyperparameters as the baseline model mentioned above. The additional hyperparameters are as follows: L2 loss coefficient = 0.1 for pre-training the RL phase mask, entropy with linear decay from 0.00001 to 0, positive reward = 5 for every correct action and negative reward = 0.5 for an incorrect action.

---

[2] We used 599 as the random seed for both tf.set_random_seed and tf.random_normal_initializer for our embedding matrices.

References

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv preprint arXiv:1704.00057*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv e-prints*, abs/1409.0473.

A. Bordes and J. Weston. 2016a. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Antoine Bordes and Jason Weston. 2016b. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Yun-Nung Chen, Dilek Hakkani-Tür, Gökhan Tür, Jianfeng Gao, and Li Deng. 2016. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *INTERSPEECH*, pages 3245–3249.

Mihail Eric and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 851–861.

Seokhwan Kim, Luis Fernando DHaro, Rafael E Banchs, Jason D Williams, and Matthew Henderson. 2017. The fourth dialog state tracking challenge. In *Dialogues with Social Robots*, pages 435–449. Springer.

Andrea Madotto and Giuseppe Attardi. 2017. Question dependent recurrent entity network for question answering. *CoRR*, abs/1707.07922.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Query-reduction networks for question answering. *arXiv preprint arXiv:1606.04582*.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2015. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Pei-Hao Su, David Vandyke, Milica Gasic, Dongho Kim, Nikola Mrksic, Tsung-Hsien Wen, and Steve Young. 2015. Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. *arXiv preprint arXiv:1508.03386*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

J. Weston, S. Chopra, and A. Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

AirDialogue: An Environment for Goal-Oriented Dialogue Research

Wei Wei[1], Quoc V. Le[2], Andrew M. Dai[2] and Li-Jia Li[1]

[1]Google Cloud AI
[2]Google Brain
wewei, qvl, adai, lijiali@google.com

Abstract

Recent progress in dialogue generation has inspired a number of studies on dialogue systems that are capable of accomplishing tasks through natural language interactions. A promising direction among these studies is the use of reinforcement learning techniques, such as self-play, for training dialogue agents. However, current datasets are limited in size, and the environment for training agents and evaluating process is relatively unsophisticated. We present AirDialogue, a large dataset that contains 402,038 goal-oriented conversations. To collect this dataset, we create a context-generator which provides travel and flight restrictions. We then ask human annotators to play the role of a customer or an agent and interact with the goal of successfully booking a trip given the restrictions. Key to our environment is the ease of evaluating the success of the dialogue, which is achieved by using ground-truth states (e.g., the flight being booked) generated by the restrictions. Any dialogue agent that does not generate the correct states is considered to fail. Our experimental results indicate that state-of-the-art dialogue models on the test dataset can only achieve a scaled score of 0.22 and an exact match score of 0.1 while humans can reach a score of 0.94 and 0.93 respectively, which suggests significant opportunities for future improvement.

1 Introduction

Designing machines to talk like a human is one of the most important goals of research in machine learning and natural language generation. (Turing, 1950; Levin et al., 1997, 2000; Banchs and Li, 2012). Rooted in seq2seq models (Sutskever et al., 2014; Cho et al., 2014), recent neural based dialogue models (Shang et al., 2015; Sordoni et al., 2015; Vinyals and Le, 2015; Li et al., 2016a; Wen et al., 2016; Bordes et al., 2017; Lewis et al., 2017; Pieraccini et al., 2009; Serban et al., 2017) have generated promising results. However, building a robust and reliable agent that can hold a conversation with humans while achieving a specific goal remains an open challenge. While a majority of previous work studied *chitchat models* (Ghazvininejad et al., 2018; Sordoni et al., 2015), in this paper we focus on *goal-oriented models* (Li et al., 2017; Liu and Lane, 2017; Liu et al., 2017) for conversations.

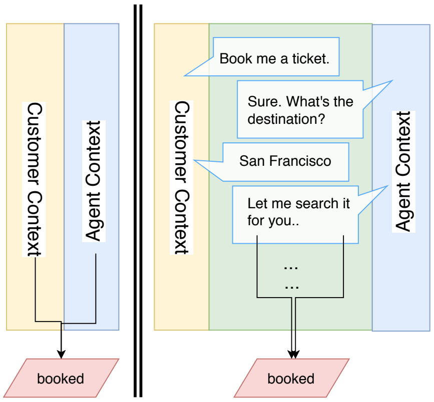

Figure 1: Goal-Oriented Dialogue Environment. (left) Context pairs are mapped to unique states in the environment. (right) Conversation models can only access its own private context and utterance in the public domain. At the end of the conversation, dialogue states are generated from one of the agents using information from the utterance.

We define a goal-driven dialogue to be a conversation that is conditioned on a pair of contexts $c = (u, v)$, with the goal of reaching the target state $s \in \mathcal{S}$. For a dialogue environment E, there exists a mapping f_E that maps from the context pair to the target state (i.e., $s = f_E(c)$). While an environment can access to the full context pair, a dialogue agent, say A_u, can only access its own private context u and the dialogue history $h_t = \{x_1, x_2, ...x_t\}$ with x_t being an utterance generated by one of the agents at time t (i.e., $x_{t+1} \sim A_v(x|v, h_t)$

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3844–3854
Brussels, Belgium, October 31 - November 4, 2018.

or $x_{t+1} \sim A_u(x|u, h_t)$). By forbidding accessing to the context of the other party, goal-driven dialogues will have to be developed so that the dialogue history h_t contains all the information that is necessary for a particular agent, say A_v to reach the target state s of the conversation defined by the environment through a mapping g_v (e.g., $s = g_v(h_t, c_v) = f_E(c)$). When one of u or v is a human, A_u and A_v will have to belong to a class of generators that respond in natural language.

We present AirDialogue, a large-scale corpus with 402,038 dialogues and an environment that makes it easy to simulate and evaluate goal-oriented dialogues. Our setting is centered around the theme of a flight booking session between a customer and a support agent. Since it's easy to find a rule based strategy to book a ticket given all the constraints, a mapping can be easily found in order to generate the ground-truth state (e.g., the ticket that needs to be booked) for each dialogue context so that we can evaluate the generated dialogue. In our environment, a context pair c always comes with a unique s. If the dialogue agent generates a state s' that is different from s, the agent has failed to achieve the goal. We use this as a mechanism to measure the performance of dialogue agents. We consider an additional metric to measure the "natural languageness" of the conversations so that the agents do not just exchange bits.

We have implemented some strong dialogue generation models and experimented with them on our dataset. Experimental results demonstrate that even the most advanced model can only achieve a benchmark score of 0.33. Comparing that to the human score of 0.94, that leaves for significant future improvement.

2 Existing Datasets

A comparison between the AirDialogue and several publicly available ones is shown in Table 1. Existing datasets are usually too small to support deep learning approaches to model dialogues generation. As a comparison, the WMT'15 English-Czech dataset (Luong and Manning, 2016), a benchmark dataset for machine translation, contains 15.8 million translation pairs whereas the current largest goal-oriented dataset has only 20,300 conversations. Synthesized data can also be an option to obtain a large dataset. However, these are often built from templated responses which make it meaningless for dialogue models to learn. Another issue with conversation datasets is the lack of a sophisticated environment that can be used to evaluate a generated dialogue. Some of the recent datasets provide an environment but are generally not representative enough to model real-world settings as illustrated by a narrow context space. As a result, the limited availability of datasets and complex environments have become a bottleneck for research in goal-oriented dialogue.

Our dataset has more than 20 times as many samples as found in the biggest of the existing public datasets. In addition to the number of samples, we have also compared the context complexity and the state complexity. Context complexity measures the unique number of context that a conversaion can be grounded into and state complexity measures the number of states that a conversation can reach. As we can see from the table, AirDialogue has the largest complexity in both context and state, giving it the flexibility to form a diverse selection of goal-oriented conversations. Our dataset also supports a wide range of tasks that can be found in the dialogue community. These include dialogue generation, state tracking and dialogue self-play.

3 Task Environment

We formulate the flight booking problem as a collaborative goal-driven dialogue problem that was defined in the introduction. Two types of agents are present: customers and agents. Dialogues are conditioned on a context pair $c = (c_c, c_a)$, with c_c being the context for the customer and c_a for the context of the agent. Here, the customer context $c_c = (tr, o)$ consists of the goal of the dialogue o (i.e., book, change or cancel) as well as the travel constants tr. Agent context $c_a = (db, r)$ consists of available flights in the database db and a field r indicating whether the customer has an existing reservation in the system. A final dialogue state s is derived at the end of the conversation once the agent has acquired all the information and the customer has confirmed all the changes in their reservation.

Task Logic. One of the main purposes of the flight booking problem is to mix decision making in the context of a dialogue. Figure 2 illustrates the task logic in order to successfully solve our problem. The goal of the conversation is provided as part of the customer's context, which has to be one of the following:

- book: make a new reservation

Dataset	Context complexity	State complexity	Supported tasks	Num. samples
Real Datasets				
AirDialogue	$\geq 4.43 \times 10^{178}$ [a]	750,000 [b]	Dialogue Generation Dialogue Self-play State Tracking	402,038
DSTC1-4 (Henderson et al., 2014)	Unknown	Unknown	State Tracking	20,300
Stanford CoCoA (He et al., 2017)	16	N/A	Dialogue Generation	11,157
Talk and Walk (de Vries et al., 2018)	80	2	Dialogue Generation	10,000
Negotiation Chatbot (Lewis et al., 2017)	3	7×3	Dialogue Generation Dialogue Self-play	5,808
Frames (El Asri et al., 2017)	Unknown	20	Dialogue Generation State Tracking	1,369
Key-Value Retrieval Networks (Eric et al., 2017)	284	284	Dialogue Generation State Tracking	3,031
Cambridge Restaurant System (Wen et al., 2016)	Unknown	Unknown	Dialogue Generation State Tracking	680
Synthesized Datasets				
AirDialogue Synthesized	$\geq 4.43 \times 10^{178}$	750,000	Dialogue Generation State Tracking Dialogue Self-play	-
Facebook bAbI dialog tasks (Bordes et al., 2017)	Unknown	Unknown	Dialogue Generation State Tracking	-
Task-Completion Dialogue Systems (Li et al., 2016b)	Unknown	3	Dialogue Generation State Tracking	-

[a] Calculated based on all possible combinations of customer and agent context features in Table 2 and Table 3. Assume 365 days a year, 24 airport codes, 8 airlines and 30 flights in the database with each flight having the same departure and arrival date as the intent and is always under the customer's budget. This is a conservative estimate since the actual dataset have flights with different dates and prices.

[b] Calculated based on 30 flights in the database, 5,000 names and 5 dialogue action states.

Table 1: Comparing AirDialogue to Existing Datasets

- change: change an existing reservation
- cancel: cancel an existing reservation.

The agent is then expected to follow the task logic and guide the conversation all the way to one of the five dialogue state actions. For example, when the goal o is "book", the agent will iterate through each of the customer's set of travel restrictions tr and search for available flights in db. If there are available flights, the conversation will be concluded with the status action "booked". Otherwise a status action of "no flight found" will be returned. On the other hand, the task logic for customers with a goal of "change" would be slightly different. Agents are supposed to check for r to determine whether a reservation exists. If it does, the agent will interact with the customer to update the travel constants tr. Otherwise, a status action will be selected with "no reservation". Similarly, the conversation will conclude "no flight found" if none of the flights in db satisfies the customers' need and "changed" if the the new flight is found. Finally, for customers who wish to cancel their ticket, the agents will perform a simple check and cancel if the reservation is found and "no reservation" otherwise.

Agent Context. There are two components in the agent context $c_a = (db, r)$. $db = (f_1, f_2, \ldots, f_m)$ is a list of flights each with 12 features listed in Table 3. Each feature has a prior distribution that we use to generate those settings. For example, 90% of the flights in the database would be economy class

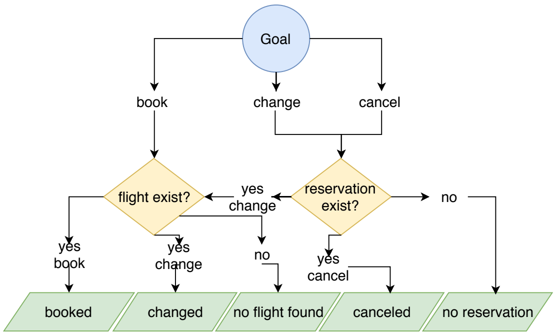

Figure 2: Task Logic of the flight booking problem

while 10% of the flights would be business class. The flight database is unique to each conversation. The price of the flights are drawn from a Gaussian distribution with mean μ and standard deviation $\sigma = \mu * \beta$. μ is 210 for economy class and 650 for business class. β is 0.2 for direct flights, 0.4 for flights with one connection and 0.6 for those with 2 connections. To simplify our setting, we only consider round trip flight tickets with both trips under the same airlines. r is simply a binary variable indicating whether the customer has previously made a reservation.

Customer Context. Customer context $c_c = (tr, o)$ also consists of two pieces. $tr = (tr_1, tr_2, \dots, tr_n)$ is a list of travel restrictions indicated in Table 2. Here we constrain the form of travel restrictions into the ones that are most useful for the flight booking situation, which is illustrated in Table 3. For example, customers may request a flight with either economy class, business class or accepts anything that is available. Some of the restrictions requires certain level of common sense knowledge to "translate" into an actual search query. Take travel time for example, a morning flight would corresponds the flight between 3am to 11am and a standard fare airline would be one of the big brand airline companies. The rest of the airlines are considered low-cost airlines. The probability of each occurrence that will be appeared in the customer context is also listed in the table.

Dialogue States. At the end of the conversation, agent will submit the dialogue states $s = (s_a, s_n, s_f)$, a state action s_a which will be one of the following 5 : "booked", "changed", "no flight found", "no reservation" and "cancel", the name of the customer s_n and the flight being selected for this dialogue s_f. Flights will be identified by a flight number that indicates one of the m flights in the database.

Environment. As we discussed earlier in the introduction, there exists a mapping $f : c \to s$ so that we can acquire the final dialogue state directly from the context pair. This mapping corresponds to our environment and the expected state $s' = f(c)$ generated from the context pair can be used to evaluate the state s generated from our algorithm.

Sentence Level Annotation. In addition to dialogue context and states, some of the sentences in the dialogues are also labeled during the data collection process. The sentence level annotation records the items agent clicked on the web UI when we were collecting the dialogue data. Agents are given the instructions to input all the travel constraints immediately after they receive them from the customers via the chat window.

4 Datasets

In this paper we present the AirDialogue dataset that contains a large collection of human generated dialogues. In addition, we also present the syntherized dataset generated using a templated simulator, along with an out-of-domain dataset that contains context that drawn from a different prior distribution than the previous two. AirDialogue and the synthesized datasets are divided into train, dev and eval sets randomly by applying a ratio of 80%, 10% and 10%. Details of the statistics are shown in Table 5.

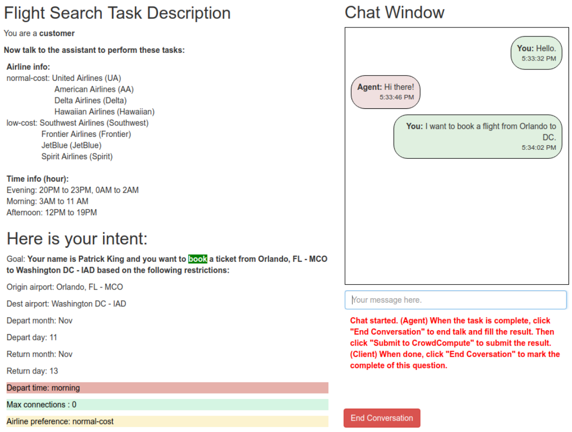
Flight Search Task Description

You are a **customer**

Now talk to the assistant to perform these tasks:

Airline info:
normal-cost: United Airlines (UA)
American Airlines (AA)
Delta Airlines (Delta)
Hawaiian Airlines (Hawaiian)
low-cost: Southwest Airlines (Southwest)
Frontier Airlines (Frontier)
JetBlue (JetBlue)
Spirit Airlines (Spirit)

Time info (hour):
Evening: 20PM to 23PM, 0AM to 2AM
Morning: 3AM to 11 AM
Afternoon: 12PM to 19PM

Here is your intent:

Goal: **Your name is Patrick King and you want to book a ticket from Orlando, FL - MCO to Washington DC - IAD based on the following restrictions:**

Origin airport: Orlando, FL - MCO

Dest airport: Washington DC - IAD

Depart month: Nov

Depart day: 11

Return month: Nov

Return day: 13

Depart time: morning

Max connections : 0

Airline preference: normal-cost

Chat Window

You: Hello.
5:33:32 PM

Agent: Hi there!
5:33:46 PM

You: I want to book a flight from Orlando to DC.
5:34:02 PM

Chat started. (Agent) When the task is complete, click "End Conversation" to end talk and fill the result. Then click "Submit to CrowdCompute" to submit the result. (Client) When done, click "End Coversation" to mark the complete of this question.

Figure 3: Customer's Interface

AirDialogue Dataset. To collect human annotated dialogue data, we first generate context pairs based on the prior distributions defined in Table 3

feature	goal	class	max. price	airline	dep./ret. time
condition (prob.)	book (80%) change (10%) cancel (10%)	economy (7%) business (3%) any (90%)	≤200 (25%) ≤500 (25%) ≤1000 (25%) any (25%)	standard fare (5%) UA, Delta AA, Hawaiian any (95%)	morning (3%) 3am-11am afternoon (4%) 12pm-19pm evening (3%) 20pm-2am any (90%) 0am-23pm
feature	dep./ret. month	dep./ret. day	max. conn.	dep./ret. airport	
condition (prob.)	uniform	uniform	0 (7%) 1 (90%) any (3%)	uniform	

Table 2: List of Customer Travel Restrictions

feature	dep./ret.city	dep./ret. month	dep./ret. day	dep./ret. time
range	categorical	1-12	1-31	00-23
prob.	uniform	uniform	uniform	uniform
feature	class	price	connections	airline
range	business,economy	0-5000	0,1,2	categorical
prob.	10%, 90%	See Section 3	7%, 90%, 3%	uniform

Table 3: Flight Features

avg. duration	4.7mins	vocab size	25,621
avg. dialogue len	115	avg. turns	14.10
avg. turn len	8.17	num. diag.	402,038
sent. annotation %	36.1	correct %	88.5

Table 4: Statistics of the AirDialogue Dataset

	Train	Dev	Test	Total
AirDialogue	321,460	40,363	40,215	402,038
Syntherized	320,000	40,000	40,000	400,000
OOD Context	-	40,000	40,000	80,000

Table 5: Statistics of All the Datasets Used in the Paper

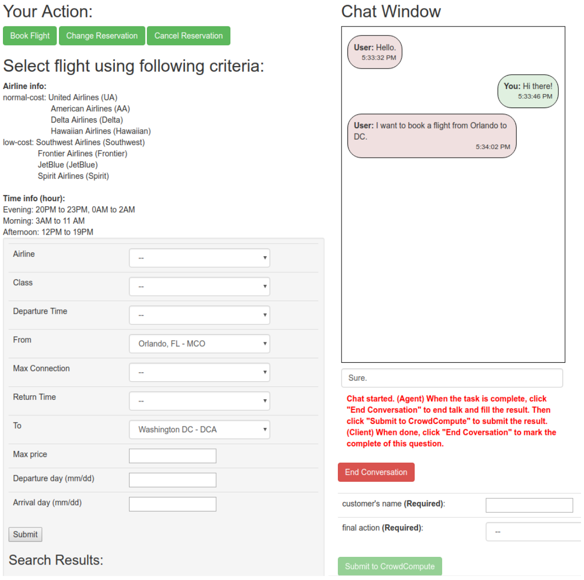

Your Action:

Select flight using following criteria:

Airline info:
normal-cost: United Airlines (UA)
American Airlines (AA)
Delta Airlines (Delta)
Hawaiian Airlines (Hawaiian)
low-cost: Southwest Airlines (Southwest)
Frontier Airlines (Frontier)
JetBlue (JetBlue)
Spirit Airlines (Spirit)

Time info (hour):
Evening: 20PM to 23PM, 0AM to 2AM
Morning: 3AM to 11 AM
Afternoon: 12PM to 19PM

Airline: --
Class: --
Departure Time: --
From: Orlando, FL - MCO
Max Connection: --
Return Time: --
To: Washington DC - DCA
Max price:
Departure day (mm/dd):
Arrival day (mm/dd):

Search Results:

Chat Window

User: Hello. 5:33:32 PM
You: Hi there! 5:33:46 PM
User: I want to book a flight from Orlando to DC. 5:34:02 PM

Sure.

Chat started. (Agent) When the task is complete, click "End Conversation" to end talk and fill the result. Then click "Submit to CrowdCompute" to submit the result. (Client) When done, click "End Coversation" to mark the complete of this question.

customer's name **(Required)**:
final action **(Required)**: --

Figure 4: Agent's Interface

and Table 2. 30 flights are generated for each dialogue. Annotators are then asked to role play the dialogue using the web interface illustrated in Figure 3 and Figure 4. The customer is shown with the goal and any requirements, as well as the chat history. The agent has a similar interface with the addition of a search feature that will search and return the cheapest flights that satisfy the given search constraints. The layouts and colors of the UI were optimized to reduce human errors. Human annotators are highly familiar with the settings of the task as most of them stayed in the project full time for more than 6 month. A human project manager manually examines roughly 5%-6% of the data each day and provide feedbacks to the human annotators to ensure the quality of the data collection. Table 4 shows some of the statistics of the AirDialogue dataset. On average, 88.5% of the dialogues generated by human reaches a perfect state. In the next Section we will analyze the types of human mistakes. In addition to dialogue history, we have also recorded agent search events (e.g. adding a new search constant through the web UI) on each turn, which are sentence level dialogue state annotations. Annotators are given the instructions to put search constraints immediately after they have received them from the natural conversation. 36.1% of the dataset dialogues have access to such information. Tracking search events provides a structured representation of progress of the dialogue.

Synthesized Dataset. In addition to the AirDialogue dataset collected using human annotators, we have also built a dialogue simulator to generate synthesized dialogues. The dialogue simulator relies on the context generator with the same set of priors. Synthesized dialogues are generated by following a set of templates and alternate between them randomly.

Out-of-domain Context Set. We have also generated an out-of-domain context set that does not contain any dialogues. This context set is generated by setting the goal probability from the one showing in Table 2 to a uniform distribution. The reservation probability is also changed from 10% to 70%. The sets of customer name and airport codes have also gone up significantly in those two datasets. This makes it difficult for models with fixed vocabulary size to perform well on those OOD datasets.

5 Data Analysis

5.1 Required Skills

This dataset presents many challenges for existing methods. Table 6 lists some of the skills that are required to accomplish the flight booking task.

Lexical and Syntactic Variations. Human language is diverse and there are many forms of lexical and syntactic variations. Taking the examples in Table 6, the amount of variation that appears in human dialogue poses great challenges for conversational models.

Applying External Knowledge. Another challenge in our data set is the use of external (commonsense) knowledge. Vaguely defined concepts such as morning and afternoon are used comfortably by humans. However, a learning algorithm needs to successfully adapt those concepts when searching for flights. An alternative way to solve this problem is to inject external knowledge into the algorithm, which is ananother important issue in dialogue systems.

Active Information Seeking Conversation. We have observed that human annotators who have high correct rates often have the habit of actively requesting information. They take extra steps to ensure all the flight search conditions are correctly communicated. This is especially important since customers are the only party in the dialogue who have access to the travel restrictions.

Goal-driven Dialogue Development. Another necessary skill to solve the flight booking problem is to develop dialogue that can be used to drive the conversation towards its end goal. Having such a goal in mind distinguishes goal-oriented models from chitchat models and makes the conversation more effective and efficient.

Reasoning over Large Structured Data. Selecting flights relies on effective methods to reason over a large scale structured database. This is a challenge that has practical impact but has rarely been addressed in previous research.

Learning from Multiple Solutions. A final challenge in the problem is the fact that there exists multiple equally optimal flights to the same set of customer restrictions.

5.2 Analysis on Human Mistakes

As we have reported in Table 7, the human error rate on this task is close to 10%. We have analyzed the human errors and grouped them into 6 categories. Here an invalid status indicates that agents have chosen a status that they are not supposed to reach according to Figure 2. For example, a "book" goal should not reach "no reservation" as an action status. "Wrong status", on the other hand, is a possible action status to reach but are not expected given the context of the conversation. Minor mistakes comprise of situations that include when agents misspell the name of the customer but get everything else correct. Those mistakes can be fixed in the dialogue from the ground truth. The majority (85%) of the errors happened when communicating flight search constraints, and entering wrong conditions that lead the search tool to return no results (6.8%).

6 Methods

6.1 Supervised Learning

Model Architecture. Our supervised dialogue model is built based on the seq2seq model(Sutskever et al., 2014). We treat both context from customer and agent as sequences and encode them using RNN. For customer context c_c we encode it using a single RNN. To encode agent context c_a we apply a hierarchical RNN structure by first encoding each flight using an RNN and then encode the outputs of each encoded flights along with the reservation information using another RNN. Utterance of time t is generated using

Skills	Examples
Lexical and Syntactic Variations	Customer: My travelling dates are **Aug 12-14**. Customer: I want to take off on **Sept 18** and please confirm my return ticket on **Sept 20**. Customer: Travelling dates are **12/13** and **12/15**.
Applying External Knowledge	Customer: I am traveling on Oct, 10 and I am returning on Oct, 12 in the **evening**. Customer: I prefer **normal-cost** airlines.
Active Information Seeking	Customer: And **please make sure** that the departure time is at afternoon. Agent: Do you have any **other specifications**? Agent: Can you **mention** Washington airport code for me? Agent: Do you like to travel in a **economy class or business class**?
Goal-driven Dialogue Development	Agent: Sure, please **provide me with your planned travel dates**. Agent: Hello, **how can I help you**? Agent: Thank you for reaching out to us. **Have a great time**.

Table 6: Conversational Skills required to accomplish the AirDialogue task.

invalid status	4.4%	minor mistakes	2.0%
reservation	1.0%	flight constant	85%
wrong condition	6.8%	wrong status	0.8%

Table 7: Human error statistics.

a sequence2sequence model by concatenating the context embedding along with the embeddings of conversation history h_{t-1}. Agent and customer will have their own model $P(x_t|h_{t-1}, c_a; \theta_a)$ and $P(x_t|h_{t-1}, c_c; \theta_c)$. At the end of the conversation the dialogue state will be generated in a sequence using another sequence2sequence model by taking the entire conversation history and the agent context, $P(s_i|s_{i-1}, h_T, c_a; \theta_s)$.

Optimization. During supervised learning, we optimize the model by considering the loss from both the dialogues x and their states s. A token x_t can belong to a either customer utterance ($x_t \in \pi_c$) or an agent utterance ($t \in \pi_a$). The parameters for supervised learning contains all the parameters of the models: $\Theta = \{\theta_a, \theta_c, \theta_s\}$. In supervised learning we optimize the following loss function.

$$\begin{aligned}\ell(\Theta) = -\sum_{(x,c,s)\in\mathcal{D}} \sum_{t=1}^{T} & \mathbb{1}_{\pi_c}(x_t)\log P(x_t|h_{t-1}, c_c; \theta_c) + \\ & \mathbb{1}_{\pi_a}(x_t)\log P(x_t|h_{t-1}, c_a; \theta_a) + \\ \sum_i & \log P(s_i|s_{1:i-1}, h_T, c_a; \theta_s)\end{aligned}$$

6.2 Reinforcement Learning Self-Play

Supervised learning for dialogue generation is known for many issues such as generating templated responses regardless of the inputs (Li et al., 2015). Here we design a reinforcement learning self-play algorithm to enable the model to learn from the environment by chatting with each other. Our self-play model is initialized using a model trained from the supervised learning. Since no conversation data is involved in the self-play, we generate context pairs directly from the context generator during training. Here we consider terminal rewards, which is generated by simulating the dialogue all the way to the end and compare the generated state s with the ground truth state s'. We use the scaled score as rewards introduced in the paragraph of Evaluation Metrics in Section 7.

Value Network. To reduce variance, we build a value network to provide a baseline estimate for returns. Both agent and customer gets their own value network $v_a(h_t|c_a; \theta_{v,a})$ and $v_c(h_t|c_c; \theta_{v,c})$. The value functions are parameterized by a seq2seq model and a linear transform applied on its output. During the training of the value functions, the main model parameters Θ are fixed and the only trainable variables are $\theta_v = \{\theta_{v,a}, \theta_{v,c}\}$.

Policy Network. We use the same structure as in supervised learning to be our policy network. We adopt REINFORCE algorithm (Williams, 1992) to optimize our algorithm using the following gradient.

$$\begin{aligned}&\nabla\ell_{RL}(\Theta) = \\ &\mathop{\mathbb{E}}_{x_t\in\pi_c}(R_t - v_c(h_{t-1}))\nabla\log P(x_t|h_{t-1}, c_c; \theta_c) + \\ &\mathop{\mathbb{E}}_{x_t\in\pi_a}(R_t - v_a(h_{t-1}))\nabla\log P(x_t|h_{t-1}, c_a; \theta_a) + \\ &\mathop{\mathbb{E}}_{s_i}(R_t - v_a(h_T, s_{i-1}))\nabla\log P(s_i|s_{i-1}, h_T, c_a; \theta_s)\end{aligned}$$

Experiments	pplx	BLEU	Name Acc.	Flight Acc.	Action Acc.
Synthesized dev	1.08	68.72	100%	100%	100%
Synthesized test	1.08	68.73	100%	100%	100%
AirDialogue dev	2.21	23.26	100%	100%	100%
AirDialogue test	2.20	23.75	100%	100%	100%

Table 8: Dialogue Generation and State Prediction

Experiments	Name	Flight	State	Total	BLEU
Supervised (Synthesized dev)	0.39(0%)	0.11(8%)	0.32(32%)	0.23(0.14)	**68.72**
Self-play (Synthesized dev)	**0.47(0%)**	**0.36(35%)**	**0.39(39%)**	**0.39(0.29)**	62.71
Supervised (Synthesized test)	0.39(0%)	0.08(4%)	0.33(33%)	0.22(0.12)	**68.73**
Self-play (Synthesized test)	**0.47(1%)**	**0.35(16%)**	**0.47(47%)**	**0.41(0.22)**	62.66
Supervised (AirDialogue dev)	0.4(0.9%)	0.07(1.2%)	0.12(12%)	0.15(0.04)	**23.26**
Self-play (AirDialogue dev)	**0.41(1%)**	**0.13(4%)**	**0.29(29%)**	**0.23(0.11)**	19.65
Supervised (AirDialogue test)	0.39(1%)	0.08(1.6%)	0.08(8%)	0.14(0.03)	**23.15**
Self-play (AirDialogue test)	**0.43(1%)**	**0.11(3%)**	**0.28(28%)**	**0.22(0.10)**	18.84
Human (AirDialogue test)	1 (98%)	0.92 (91.4%)	0.92 (91.8%)	0.94 (0.93)	-

Table 9: Dialogue Self-play Displayed with Scaled Scores and Their Exact Match Scores in the Parentheses.

Exp	Name	Flight	State	Dialogue
OOD1	0.4(0%)	0.1(1%)	0.18(18%)	0.18(0.06)
OOD2	0.4(0%)	0.09(2%)	0.21(21%)	0.19(0.07)

Table 10: Performance of Trained Self-play Models on Out-of-domain Context Pairs using AirDialogue Data

7 Experiments

Experiment Setup. We implemented our model using Tensorflow using SGD as the optimizer with a learning rate of 0.1 and a batch size of 64. The seq2seq model was implemented using 4 layers of GRU with a hidden unit 384. Greedy Decoder is used for seq2seq decoding. Inputs are tokenized using NLTK [1]. For AirDialogue dataset, tokens occurred less than 10 times are eliminated but no tokens are removed for the synthesized dataset. As a result, there are 5,547 tokens left the experiments. There are 700 tokens for the synthesized dataset and no tokens are eliminated during the pre-processing. In training we only applied the dialogues that have correct states.

Accelerate Training In the usual seq2seq diagram for dialogue generation, one would treat a single conversation with k turns as k different training samples by feeding conversation before the k^{th} turn into the encoder and use a decoder to generate the k^{th} turn. Such a training strategy would encode the dialogue history repeatedly. We apply a technique to speed up training that is illustrated in Figure 5. Here the encoder is never needed to encode a single dialogue multiple times since its outputs are reused for multiple turn predictions. The decoder generates the output sequence by alternating its states between previous decoder state and the encoder states. If the sentence is within the boundary of the current turn, its hidden state got passed from its previous state. Otherwise, its hidden state will be "reset" into the corresponding state in the encoder. One can easily implement this training strategy and use a pre-processed Boolean array to represent whether a token is within a turn for a specific agent.

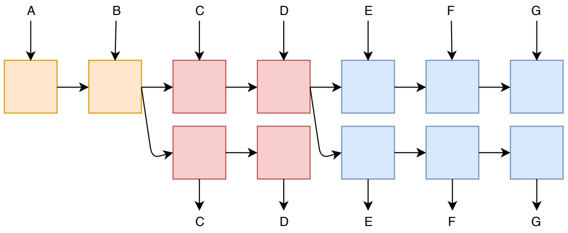

Figure 5: Techniques to speed up training. Here a conversation with 3 turns are annotated using colors. The encoder only needs to pass through the dialogue once for the entire dialouge sample to be trained.

Evaluation Metrics. We use perplexity and BLEU score to evaluate the quality of the language generated by the model. We also compare the dialogue state generated by the model s and the ground truth state s'. Two categories of the metrics are used: exact match scores and scaled scores. In an

---

[1] https://www.nltk.org

exact match metric, dialogue state is given a score of 1 if it matches exactly to the ground truth and 0 otherwise. In a scaled metric, scores are scaled between 0 and 1 to provide information that are of finer granularity. There are three dialogue states: name, flight and action. For name, scaled metric is chosen to be the character-wise F1 score. For flight, scaled metric is chosen to be 1 minus the scaled distance between the selected flight f and the ground truth F_g. Note there might be multiple optimal ground truth flights that have the same price and satisfy the customers' requirements. Therefore F_g should be a set of flights. The distance function $d(f1, f2)$ is a measure of distance on each of the flight features. The scaled score on flight is calculated as the following. Here F is a set of all flights in the datasbase.

$$score(f) = 1 - \frac{\min_{f' \in F_g} d(f, f')}{\max_{f1 \in F, f2 \in F_g} d(f1, f2)}$$

Dialogue action states can only have exact match metrics. Finally, the total score of a dialogue is taken to be a weighed sum scores of name, flight and dialogue status by a factor of 0.2, 0.5 and 0.3 for both scaled and discrete metrics.

Dialogue Generation and State Prediction. We train the models on the train sets and show their performs on the dev and test sets in Table 8. The BLEU score measured by comparing the generated response and the ground truth is around 68.7 for synthesized data and around 23 for AirDialogue. Given the fact that templated dialogues are easier to learn, it is expected that the synthesized dataset gets a high BLEU score. In the state prediction task, the model paper achieved a perfect accuracy across all the dialogue states given the ground truth dialogue and previous states. However, as we will see shortly, the triumph on ground truth hisotry might not be able to be transferred to self-play experiments, which generates dialogues that have different distributions from the ground truth data.

Dialogue Self-play. During the self-play experiments we perform similar predictions on the dialogue states. However, instead of asking the models to predict those states given ground truth history, we now ask the models to predict given the generated dialogues. Table 9 shows the results using both the supervised model and the self-play model. Here we see significantly improvements across all measures for self-play models compare to their supervised learning models. However, the fact that the exact match scores are so low indicates that our models are far from mastering the goal-oriented dialogue problem in the self-play setting as the rewards and accuracies are consistently low. As a comparison, human agents achieved nearly 90% on rewards across all categories, which sets a good target for future work in the field. One possible reason for the low exact match score but relatively high scaled score is because we use the scaled score as rewards in out self-play training. As a result, the metrics are highly tuned toward scaled scores instead of exact match scores. One can apply techniques such as pointer networks (Vinyals et al., 2015) which is possible to optimize exact match scores in a better way. To prevent language from degenerating into binary bits, we mix three supervised training steps on the train data with one reinforcement learning update during self-play training. By doing this, we are able to maintain a BLEU score at similar level compares to the supervised learning.

Out-Of-Domain Self-play. We have also conducted experiments on the out-of-domain context pairs. The results are shown in Table 10. The out-of-domain context pairs contain dialogue contexts with distribution far deviated from the training data. It is not surprised to see here that our model does not perform as good as in the testing data using the data it is familiar with.

8 Conclusions

In this paper, we propose an environment for goal-oriented dialogue research based on the problem of flight bookings. We have collected a dataset that is more than 400,000 conversations. Our environment allows easy generation of new dialogue contexts and allows verification of the generated dialogues, which can be used to support a wide range of research such as dialogue self-play. Although supervised learning seems to perform well in our setting, self-play poses a challenge for goal-oriented dialogue research. The gap between our self-play approach and the human baseline suggests possibilities for significant future improvements.

References

Rafael E Banchs and Haizhou Li. 2012. Iris: a chat-oriented dialogue system based on the vector space model. In *Proceedings of the ACL 2012 System Demonstrations*, pages 37–42. Association for Computational Linguistics.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 207–219. Association for Computational Linguistics.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49. Association for Computational Linguistics.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *AAAI Conference on Artificial Intelligence*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1766–1776, Vancouver, Canada. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1997. Learning dialogue strategies within the markov decision process framework. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 72–79. IEEE.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23.

Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016a. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202. Association for Computational Linguistics.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016b. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *arXiv preprint arXiv:1708.05956*.

Bing Liu, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck. 2017. End-to-end optimization of task-oriented dialogue model with deep reinforcement learning. *arXiv preprint arXiv:1711.10712*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063. Association for Computational Linguistics.

Roberto Pieraccini, David Suendermann, Krishna Dayanidhi, and Jackson Liscombe. 2009. Are we there yet? research in commercial spoken dialog systems. In *International Conference on Text, Speech and Dialogue*, pages 3–13. Springer.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C Courville. 2017. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pages 3288–3294.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Alan M. Turing. 1950. I.computing machinery and intelligence. *Mind*, LIX(236):433–460.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the 31st International Conference on Machine Learning*.

Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. 2018. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

QuaSE: Sequence Editing under Quantifiable Guidance*

Yi Liao[13], **Lidong Bing**[2], **Piji Li**[12], **Shuming Shi**[2], **Wai Lam**[1], **Tong Zhang**[2]
[1]The Chinese University of Hong Kong
[2]Tencent AI Lab
[3]Noah's Ark Lab, Huawei Technologies
{yliao, pjli, wlam}@se.cuhk.edu.hk
{lyndonbing, shumingshi, bradymzhang}@tencent.com

Abstract

We propose the task of Quantifiable Sequence Editing (QuaSE): editing an input sequence to generate an output sequence that satisfies a given numerical outcome value measuring a certain property of the sequence, with the requirement of keeping the main content of the input sequence. For example, an input sequence could be a word sequence, such as review sentence and advertisement text. For a review sentence, the outcome could be the review rating; for an advertisement, the outcome could be the click-through rate. The major challenge in performing QuaSE is how to perceive the outcome-related wordings, and only edit them to change the outcome. In this paper, the proposed framework contains two latent factors, namely, outcome factor and content factor, disentangled from the input sentence to allow convenient editing to change the outcome and keep the content. Our framework explores the pseudo-parallel sentences by modeling their content similarity and outcome differences to enable a better disentanglement of the latent factors, which allows generating an output to better satisfy the desired outcome and keep the content. The dual reconstruction structure further enhances the capability of generating expected output by exploiting the couplings of latent factors of pseudo-parallel sentences. For evaluation, we prepared a dataset of Yelp review sentences with the ratings as outcome. Extensive experimental results are reported and discussed to elaborate the peculiarities of our framework. [1]

1 Introduction

Typical neural text generation is observed suffering from the problems of repetitions in word n-grams, producing monotonous language, and generating short common sentences (Li et al., 2017). To solve these problems, some researchers branch out into the way of post-editing (could be under some guidance, say sentiment polarity) a given message to generate text of better quality. For example, skeleton-based text generation first outlines a skeleton in the form of phrases/words, and then starts from the skeleton to generate text (Wang et al., 2017; Xiao et al., 2016). Another line of works conduct editing on an existing sentence and expect that the output will serve particular purposes better (Guu et al., 2018). Similarly in conversation, some systems post-edit the retrieval results to generate new sentences as the response (Song et al., 2016). The third type is to perform editing on the input under the guidance of specific style. For example, Shen et al. (2017) take a sentence with negative sentiment as input, and edit it to transfer its sentiment polarity into positive.

In this paper, we generalize the third type of post-editing into a more general scenario, named **Qua**ntifiable **S**equence **E**diting (**QuaSE**). Specifically, in the training stage, each input sentence is associated with a numeric outcome. For example, the outcome of a review sentence is its rating, ranging from 1 to 5; the outcome of each advertisement is its click-through rate. In the test stage, given an input sentence and a specified outcome target, a model needs to edit the input to generate a new sentence that will satisfy the outcome target with high probability. Meanwhile, the output sentence should keep the content described by the input. For example, given the input sentence "The food is terrible", a desired output sentence could be "The food is OK" under the expected outcome

---

* The work described in this paper was done when Yi Liao was an intern at Tencent AI Lab. The work is partially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: CUHK413510)

[1] Our code and data are available at `https://bitbucket.org/leoeaton/quase/src/master/`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3855–3864
Brussels, Belgium, October 31 - November 4, 2018.

"3.1" (a neutral sentiment), and "The food is delicious" under the expected outcome "4.0". If no outcome target is given, the model could generate "The food is extremely delicious", by defaulting the best outcome, or "The food is extremely terrible", by defaulting the worst outcome.

Our problem setting is more general than previous works in two major aspects: (1) The outcome here is numerical, and it can be regarded as a generalization of the categorical outcome in (Shen et al., 2017; Hu et al., 2017; Gao et al., 2018). With such numerical outcome, it is impossible to construct two corpora as counterpart of each other as done in (Shen et al., 2017; Gao et al., 2018). (2) The editing operation is under a quantifiable guidance, i.e. the specified outcome or the defaulted extrema. For example, we can specify a particular target rating, such as 3.1 or 4.0, as the expected outcome. Although Mueller et al. (2017) also take outcome-associated sentences for training, their model does not perform such outcome-guided editing for sentence generation.

Considering that the goal of the task is to generate an output that satisfies a specified outcome and keeps the content unchanged, QuaSE is challenging in a few aspects. Firstly, a model should be able to perceive the association between an outcome and its relevant wordings. For the previous example "The food is terrible", the model needs to figure out that the low rating is indicated by the word "terrible", instead of "food". Secondly, when performing editing, the model should keep the content, and only edits the outcome-related wordings. Moreover, the model needs to take a specified outcome into account and generate an output that satisfies the specified outcome value with high probability. Continuing the running example, given the expected outcome 3.1, "The food is OK" is an appropriate output, but "The food is extremely delicious" and "The service is OK" are not. Thirdly, we do not have readily available data, such as data points like [input sentence: "The food is terrible", expected outcome: 4.0, output sentence: "The food is delicious"] to show the model what the revised output should look like, that meet our need to train models.

We propose a framework to address this task. The fundamental module of our framework is a Variational Autoencoder (VAE) (Kingma and Welling, 2013) to encode each input sentence into a latent content factor and a latent outcome factor, capturing the content and the outcome related wordings respectively. We propose to leverage pseudo-parallel sentence pairs (i.e, the two sentences in a pair have the same or very similar content, but different outcome values) to enhance our model's capability of disentangling the two factors, which allows attributing the wording difference of the sentences in a pair to the outcome factor, and the wording similarity to the content factor. For sentence reconstruction, we employ a Recurrent Neural Network (RNN) based decoder (Sutskever et al., 2014) that takes as input the combination of a content factor and an outcome factor. To further enhance the capability of generating expected output, we introduce a dual reconstruction structure which exploits the couplings of latent factors of pseudo-parallel sentences. Specifically, it attempts to reconstruct one sentence in a pair from the combination of its outcome factor and the other sentence's content factor, based on the intuition that the wording difference in a pair is outcome-related. In the test stage, taking a sentence and a specified outcome target as input, our model generates a revised sentence which likely satisfies the specified target, and meanwhile the content is preserved as much as possible.

To evaluate the efficacy of our framework, we prepared a dataset of Yelp review sentences with the ratings as outcome. Compared with state-of-the-art methods handling similar tasks, experimental results show that our framework can generate more accurate revisions to satisfy the target outcome and transfer the sentiment polarity, meanwhile it keeps the original content better. Ablation studies illustrate the effectiveness of the designed components for enhancing the performance. We have released the prepared dataset and the code of our model to facilitate other researchers to do further research along this line, refer to Footnote 1.

2 Model Description

2.1 Problem Setting and Model Overview

In the task of Quantifiable Sequence Editing (QuaSE), the aim is to edit an input sentence X_0 under the guidance of an expected outcome value R^* to generate a new sentence X^* that will satisfy R^* with high probability. For training a model, we are given a set of sentence-outcome tuples (X, R).

Our proposed model for training is depicted in Figure 1. The left hand side models individual sentences. Specifically, it employs two encoders,

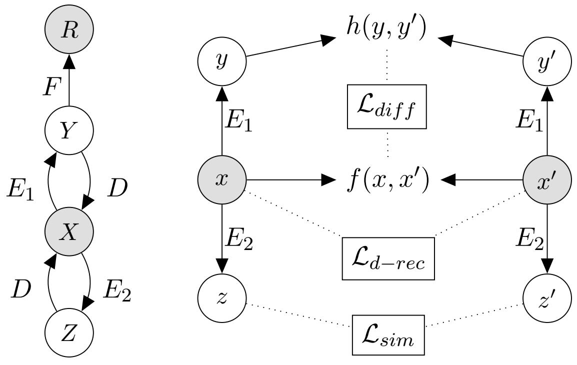

Figure 1: Model Overview.

i.e. E_1 and E_2, to encode a single sentence X into two latent factors Y and Z which capture the outcome and content properties respectively. In contrast, Mueller et al. (2017) employ a single factor for capturing these two properties, which limits the capability of distinguishing one property from the other. As a consequence, when editing a sentence towards a given outcome, the sentence content is likely to be changed, which should be suppressed as much as possible. An RNN-based decoder D takes the concatenation of Y and Z to reconstruct the input X. Moreover, a transformation function F predicts R with Y.

The right hand side models pseudo-parallel sentence pairs (automatically generated from the above tuples), so we first introduce the concept of pseudo-parallel sentences as follows. Let (x, x') denote a pair of pseudo-parallel sentences, x and x' should describe the same or similar content, but their outcomes are different. Note that we use lowercase letters to denote variables related to sentence pairs for better clarity. For two sentences in a pair, the difference of their outcome factors $h(y, y')$ is attributed to their wording difference $f(x, x')$, resulting in the loss $\mathcal{L}_{diff}$; the similar contents of two sentences should result in similar content factors, i.e. minimizing the loss $\mathcal{L}_{sim}$; moreover, a dual reconstruction loss $\mathcal{L}_{d-rec}$ is minimized to enhance the capability of generating expected output.

Overall, the model minimizes the losses from modeling single sentences and sentence pairs. After the model is trained, a separated component is applied for editing an input sentence to output a revision that satisfies a specified outcome target.

2.2 Modeling Single Sentences

In probabilistic theory, we need to maximize the log-likelihood of observing the training sentence-outcome tuples (X,R), denoted as follows:

$$\log \int p(X, R) = \log \int p(X|Y, Z)p(Y, Z)dYdZ + \log \int p(R|Y)p(Y)dY \tag{1}$$

However, the integration in the first term on the right hand side is intractable. Inspired by the idea of VAE (Kingma and Welling, 2013), we alternatively maximize the Evidence Lower Bound (ELBO) (Blei et al., 2016) incorporating variational distributions, i.e. $q(Y|X)$ and $q(Z|X)$. Thus, this term is approximated as follows:

$$\begin{aligned} &\log \int p(X|Y, Z)p(Y, Z)dYdZ \geq -[\mathcal{L}_{rec} + \mathcal{L}_{kl}] \\ &\mathcal{L}_{rec} = -\mathbb{E}_{Y,Z\sim q(Y|X),q(Z|X)}[\log p(X|Y, Z)] \\ &\mathcal{L}_{kl} = KL[q(Y|X)|p(Y)] + KL[q(Z|X)|p(Z)] \end{aligned} \tag{2}$$

where, the term $\mathcal{L}_{rec}$ denotes the error of reconstructing X. As advocated by (Kingma and Welling, 2013) and (Bowman et al., 2016), the variational distributions $q(Y|X)$ and $q(Z|X)$ are modelled as Gaussian distributions, i.e. $q(Y|X) = \mathcal{G}(\mu_{Y|X}, \sigma_{Y|X})$, and $q(Z|X) = \mathcal{G}(\mu_{Z|X}, \sigma_{Z|X})$. The expectation $\mathbb{E}(\cdot)$ can be efficiently approximated using one Monte-Carlo sample, for example, $Y \sim q(Y|X)$ and $Z \sim q(Z|X)$. In practise, we can alternatively employ $Y = \mu_{Y|X}$ and $Z = \mu_{Z|X}$ instead of sampling since they are the means of the Gaussian distributions. We employ two encoder networks E_1 and E_2 to generate $\mu_{Y|X}$ and $\mu_{Z|X}$ respectively from the sentence X, i.e. $\mu_{Y|X} = E_1(X)$ and $\mu_{Z|X} = E_2(X)$. $p(X|Y, Z)$ is the probability of observing the sentence X given Y and Z, which is modelled by a decoder network D. Thus, the reconstruction loss can be rewritten as:

$$\mathcal{L}_{rec} = H(X, D(E_1(X), E_2(X))) \tag{3}$$

where H is the cross entropy loss for the decoder.

The term $\mathcal{L}_{kl}$ in Equation 2 denotes the KL-divergence between the variational posterior distribution and the prior distribution. Following previous works (Mueller et al., 2017), the priors $p(Y)$ and $p(Z)$ are defined as a zero-mean Gaussian distribution, i.e. $p(Y) = p(Z) = \mathcal{G}(\mathbf{0}, \mathbf{I})$. The loss $\mathcal{L}_{kl}$ serves as a regularization term enforcing that the variational posterior distribution resembles the prior distribution, which also avoids overfitting.

The second term in Equation 1 models the log-likelihood of the outcomes. We adopt the usually used Taylor approximation for the calculation, where this term is approximated by an affine transformation from the outcome factor Y to the outcome R, denoted as $F(Y)$. Then, we define the loss as the square error between R and $F(Y)$:

$$\mathcal{L}_{mse} = (R - F(Y))^2 \tag{4}$$

Although Mueller et al. (2017) also model individual sentences and their outcomes, in their model, each sentence is only encoded into one latent factor to capture both outcome and content properties. In contrast, we disentangle two latent factors from a single sentence to model the outcome and the content separately to provide more flexibility. Moreover, such design allows the incorporation of the pseudo-parallel sentences, which will be described in the next subsection.

2.3 Exploiting Pseudo-Parallel Sentences

As mentioned above, pseudo-parallel sentences are similar in terms of the content but different in terms of the outcome. E.g., Table 1 shows a pair of pseudo-parallel sentences, where both talk about "the restaurant", but with different sentiments (i.e. ratings). For the pair (x, x'), let y and y' denote their outcome factors, z and z' denote their content factors. We design three components to leverage pseudo-parallel sentences to enhance our model's capabilities of disentangling the two types of factors and generating the desired output sentences.

x	I will never come back to the restaurant.
x'	I will definitely come back to the restaurant, recommend!

Table 1: A pair of pseudo-parallel sentences.

2.3.1 Modeling Outcome Difference

We exploit the wording difference $f(x, x')$ between x and x'. Note that the preparation (discussed in Section 4.1) determines that a pair of pseudo-parallel sentences are very likely to differ in the outcome factors, denoted as $h(y, y')$. Thus, by aligning the surface wording difference of two sentences in a pair and the difference in their outcome factors, we intend to improve the performance of the encoder E_1 for generating the outcome factor. $f(x, x')$ and $h(y, y')$ are defined as follows:

$$\begin{aligned} f(x,x') &= inc(x,x') \oplus dec(x,x') \\ h(y,y') &= y - y' = E_1(x) - E_1(x') \end{aligned} \tag{5}$$

where $inc(x, x')$ and $dec(x, x')$ are embeddings capturing the wording difference between x and x'. $inc(x, x')$ denotes the "increment" from x to x', i.e. the terms that appear in x' but not in x. $dec(x, x')$ denotes the "decrement". If there are multiple terms in the difference, we sample one term for inc or dec. For the example in Table 1, $dec(x, x')$ is the embedding of "never", and $inc(x, x')$ could be the embedding of "definitely" or "recommend". The effect of outliers during sampling anneals since the training data contain sufficient pairs of sentences. The symbol $\oplus$ denotes concatenation. $h(y, y')$ is defined as the subtraction between the outcome factors.

We employ a regression network U to align $f(x, x')$ and $h(y, y')$, and the loss $\mathcal{L}_{diff}$ is:

$$\mathcal{L}_{diff} = ||h(y, y') - U[f(x, x')]||^2 \tag{6}$$

2.3.2 Modeling Content Similarity

Another property of two pseudo-parallel sentences is that they share similar content. To capture it, we design a loss function minimizing the square error between the content factors.

$$\mathcal{L}_{sim} = ||z - z'||^2 = ||E_2(x) - E_2(x')||^2 \tag{7}$$

Minimizing $\mathcal{L}_{sim}$ helps the encoder E_2 generate the content factor more accurately.

2.3.3 Dual Reconstruction

The decoder D is not only used in Section 2.2 to reconstruct a single training sentence, but also employed for generating output sentences in the test stage (Section 3). To improve the robustness of D, we propose a dual reconstruction component based on the pseudo-parallel sentences. Different from reconstructing an original sentence in Section 2.2, in the dual reconstruction, given a sentence x, we reconstruct its dual sentence x'.

Specifically, we first encode x and x' into their outcome factors y/y' and content factors z/z'. Since x shares similar content with x', its content factor z, when combined with the outcome factor y' of x', should nearly reconstruct x'. For such dual reconstruction, the loss is written as:

$$\begin{aligned} \mathcal{L}^{d-rec}_{x';x} &= H(x', D(E_1(x'), E_2(x))) \\ &= H(x', D(y', z)) \end{aligned} \tag{8}$$

The same dual reconstruction process applies to the counterpart of x', i.e. x. Thus, the whole dual reconstruction loss is as follows:

$$\mathcal{L}_{d-rec} = \mathcal{L}^{d-rec}_{x';x} + \mathcal{L}^{d-rec}_{x;x'} \tag{9}$$

Note that the encoders E_1/E_2 and the decoder D here refer to exactly the same networks (i.e., the parameters are shared) as used in Section 2.2.

The specific design of the networks are as follows. E_1/E_2: RNNs of GRUs with a fully connected neural network appended to the last state to add some noise, which is a reparameterization alternative for sampling. Their outputs are the outcome and content factors, respectively. D: An RNN of GRU cells. The RNN takes the concatenation of an outcome factor and a content factor as the initial state for decoding. F: A fully connected network. It takes an outcome factor as input and outputs an outcome value. U: A fully connected network. It takes $f(x, x')$ as input to predict $h(y, y')$.

2.4 Joint Training

Considering all the aforementioned components, we define a joint loss function as:

$$\begin{aligned}\mathcal{L}_{joint} =& \lambda_{rec}\mathcal{L}_{rec} + \lambda_{kl}\mathcal{L}_{kl} + \lambda_{mse}\mathcal{L}_{mse} + \\ & \lambda_{diff}\mathcal{L}_{diff} + \lambda_{sim}\mathcal{L}_{sim} + \lambda_{d-rec}\mathcal{L}_{d-rec}\end{aligned} \tag{10}$$

in which each component is associated with a weight. Following the sigmoid annealing schedule (Bowman et al., 2016), we design the following strategy to tune the weights: (1) Tune the weights λ_{rec} and λ_{mse} on the validation dataset under the metric MAE (refer to Section 4.3), while fixing the other weights to zeros. We set $\lambda_{rec} + \lambda_{mse} = 1$; (2) Fixed the weights tuned in the first step. For each remaining loss, gradually increase the weight from 0 to 1 during the training, until the reconstruction loss $\mathcal{L}_{rec}$ or the outcome prediction loss $\mathcal{L}_{mse}$ becomes worse. The strategy prioritizes $\mathcal{L}_{rec}$ and $\mathcal{L}_{mse}$, since they are the core components for generating the revised sentences.

3 Editing under Quantifiable Guidance

In the test, the trained model edits an input sentence X_0 and outputs a revision X^* that is likely to satisfy the specified outcome target R^*, and meanwhile preserves the content as much as possible.

We first encode X_0 with E_1 and E_2 to get Y_0 and Z_0 respectively. The next step is to modify Y_0 to get a new outcome factor Y^* that is likely to generate the target outcome R^*. The process to determine a suitable Y^* is as follows. We first assume Y follows the Gaussian distribution $Y \sim \mathcal{G}(Y_0 = E_1(X_0), \sigma)$, the mean of which is Y_0. Then we choose $\mathcal{C} = \{Y : \mathcal{G}(Y|E_1(X_0), \sigma) > \tau\}$ as the feasible range for Y^*, where τ is a threshold. $\mathcal{C}$ will expand if τ is set smaller, and thus allowing more revisions. Finally, Y^* is determined as follows:

$$Y^* = \arg\min_{Y \in \mathcal{C}} (F(Y) - R^*)^2 \tag{11}$$

Note that in (Mueller et al., 2017), Y^* is determined as $\arg\max_{Y \in \mathcal{C}} F(Y)$, which does not consider an outcome target. The revised sentence X^* is generated from X_0 and Y^* via the decoder D:

$$X^* = D(Y^*, Z_0) \tag{12}$$

Thus, the content of X_0 is preserved with Z_0, and the expected outcome is achieved with Y^*.

4 Experiments

4.1 Dataset Preparation

Our dataset contains sentences extracted from Yelp reviews [2], where each review is associated with a rating in {1, 2, 3, 4, 5}. Specifically, we employ the sentences with sentiment polarity (i.e. positive or negative) used in (Shen et al., 2017) as the primary portion of our data. After some cleaning, we obtain about 520K sentences. To add neutral sentences, we randomly select 80K sentences from the original reviews with neutral sentiment (i.e. rating 3). To make sure that the neural sentences added by us are describing the same domain, we only pick neural sentences whose tokens are all in the vocabulary of the primary data. The vocabulary size of the dataset is 9,625. In total, our dataset contains 599K sentences, and we randomly hold 50K for test, 10K for validation, and the remaining for training.

For training, we need each input sentence being associated with a rating value, and for test, we need to measure the rating of a generated sentence to check if the generated sentence satisfies the specified outcome target. Therefore, an automatic method is needed for measuring the rating values of training sentences and generated sentences. We employ the sentiment analyzer in Stanford CoreNLP (Manning et al., 2014) to do so. Specifically, we first invoke CoreNLP to output the probability of each rating in {1, 2, 3, 4, 5} for a sentence, then we take the sum of the probability-multiplied ratings as the sentence rating. Some statistics of the data is given in Table 2. Hereafter, we use "rating" and "outcome" interchangeably.

---

[2] `https://www.yelp.com/dataset/challenge`

Rating interval	$[1,2)$	$[2,3)$	$[3,4)$	$[4,5]$
Sentence#	34273	231740	165159	167803

Table 2: Numbers of sentences in each rating interval.

One may think that would it be possible to use the original rating given by Yelp users as outcome for training? We did not use it for two reasons: (1) We want the ratings of training sentences and generated sentences are measured with a consistent method; (2) In fact, we find that the predicted rating with CoreNLP has a Pearson correlation of 0.85 with the rating given by users. Note that the original Yelp data only has ratings for entire reviews. We derived the sentence ratings by users like this: a sentence takes as its rating the average of the ratings of those reviews where it appears in. Human evaluation in (Shen et al., 2017) shows that a similar method for deriving polarity is basically reasonable as well.

For preparing the pseudo-parallel sentences, we first follow the ideas in (Guu et al., 2018) to generate some initial pairs. Specifically, we first calculate the Jaccard Index (JI) for each pair of sentences, and keep those with JI values no less then 0.5 as the initial pairs. Note that such initial pairs could contain many false positives (roughly 10% as manually evaluated on the Yelp corpus in (Guu et al., 2018)), because the JI calculation does not distinguish content words and outcome words. To solve this problem, we add another constraint: for an initial pair to be regarded as a pseudo-parallel sentence pair, the difference of the two sentences' ratings should be no less than 2. Here, the idea is that given the two sentences are similar enough in wordings (JI $\geq$ 0.5), if their rating scores are dissimilar enough, it looks plausible to conjecture that their wording difference is more likely outcome-related and causes the rating difference. In fact, such wording difference is exactly what we want to capture with pseudo-parallel sentence pairs. In total, we obtain about 604K sentence pairs from the single training sentences. For conducting the joint training with both single sentences and pseudo-parallel pairs, we make each data point composed of a single sentence and a sentence pair. To do so, we couple each sentence pair with a single sentence, thus we can use all pairs for training. Note that because we have more sentence pairs, some single sentences are used twice randomly in composing the data points.

4.2 Comparative Methods

Our model is compared with two state-of-the-art models handling similar tasks.

Sequence to Better Sequence (S2BS) (Mueller et al., 2017): For training, S2BS also requires each sentence is associated with an outcome. For test, S2BS only revises a sentence such that the output is associated with a higher outcome, which is not a specified value. For comparison, we adapt our revision method for S2BS, by which their trained model is able to conduct quantifiable sentence revision. We tune the parameters for S2BS by following the suggestions in their source code.

Text Style Transfer (TST) (Shen et al., 2017): In TST, the sentiment of each sentence is labelled as negative or positive. The model is able to revise a negative sentence into positive, or vice versa. Their task can be treated as a special case of our QuaSE task: we set the outcome target to 1 for the input sentences that are associated with outcomes larger than the neutral rating 3, thus, our task is equal to revising a positive sentence into negative. We follow the suggested parameters reported in (Shen et al., 2017).

4.3 Evaluation Metric and Parameter Setting

Considering that our model's task is to revise a sentence such that its outcome (predicted by Stanford CoreNLP) satisfies a specified target, we define the metric as the mean absolute error (MAE) between the specified target outcome and the outcomes of revised sentences.

$$MAE = \frac{1}{|S|} \sum_{X_i \in S} |R_i - R^*| \qquad (13)$$

where S is the set of revised sentences X_i, R^* is the target outcome, and R_i is the outcome of X_i.

After tuning on the validation set, the determined parameters are: $\lambda_{rec} = 0.75$, $\lambda_{kl} = 0.6$, $\lambda_{mse} = 0.25$, $\lambda_{diff} = 0.2$, $\lambda_{sim} = 0.2$, $\lambda_{d-rec} = 0.1$, and the dimensions of the two factors are both 50. The parameter τ for revision takes $\exp(-100000)$ for both our model and S2BS.

4.4 Automatic Evaluation

We compare our model with S2BS by specifying five target ratings, namely 1, 2, 3, 4, and 5. Both our model and S2BS are fed the sentences in the testing dataset. For each sentence, both models are required to generate five revised sentences, each satisfying one of the target ratings.

	MAE					Edit Distance				
	T=1	T=2	T=3	T=4	T=5	T=1	T=2	T=3	T=4	T=5
Original	2.2182	1.2379	0.8259	0.9279	1.7818	N/A	N/A	N/A	N/A	N/A
S2BS	1.6839	0.9444	0.7567	0.7572	1.3024	6.6439	5.342	4.9390	5.005	6.2290
Our Model	1.4162	0.6298	0.7408	0.5377	0.9408	7.9191	4.7	3.4505	4.13	8.0094

Table 3: MAE and Edit Distance for our proposed model and S2BS. T refers to the target outcome.

We evaluate the MAE between the target outcome and the outcome of the revised sentences. Each model is trained for three times and the average results are reported in Table 3. "Original" refers to the MAE between the targets and the ratings of input sentences. We can observe that the MAE values of both our model and S2BS are smaller than Original. It demonstrates that both models are able to revise the sentences towards the outcome targets. Furthermore, compared with S2BS, our model achieves smaller MAE values. One major reason is that we disentangle a content factor and an outcome factor, and design three components to leverage pseudo-parallel sentences. By modeling the wording difference, our model captures the keywords that cause the difference in the outcome. By enforcing the content factors of pseudo-parallel sentences to be similar, the model is capable to generate the content factor more precisely. Moreover, the dual reconstruction can guide the editing procedure with the hints from the parallel sentences. In contrast, S2BS only disentangles one factor for capturing both content and outcome properties, and thus it cannot perform the same operations on sentence pairs. The MAE for T=5 is smaller than that for T=1. This is partially due to the fact that the outcomes of the test sentences are closer to 5, refer to Table 2. We also report the average Edit Distance between the input sentences and the generated sentences to measure the degree of revisions. For T=1 and T=5, our model conducts more editing than S2BS, which brings in better MAE, while for T={2, 3, 4}, our model generates more accurate sentences (i.e. better MAE) with less editing. This observation coincides with the fact that we need more editing to revise a sentence towards an extreme target (i.e., 1 and 5), such as including degree adverbs "very" and "extremely".

We also compare our model with TST for sentiment polarity transfer. We employ the same evaluation metric as used in (Shen et al., 2017): the sentiment accuracy of the transferred sentences. We define the revised sentences with ratings larger than 3 as positive, smaller than 3 as negative. The results are given in Table 4, where two accuracy values are reported: negative to positive, and the reverse. The results show that our model achieves better accuracy than TST in both transfer directions. One reason is that our method models the associations between each sentence and its outcome, and thus captures the sentiment wordings better. Our model is far better for transferring negative sentences into positive, moreover, both models achieve better performance for this transfer direction. We can probably attribute the reason to the imbalanced training data: 55% positive sentences v.s. 45% negative sentences.

	Neg. to Pos.	Pos. to Neg.
TST	0.7280	0.7097
Our Model	0.8836	0.7862

Table 4: Accuracy comparison with TST.

	Content Preservation (Range: [0, 2])	Fluency (Range: [1, 4])
TST	1.02	2.56
S2BS	0.70	2.53
Our Model	1.38	2.48

Table 5: Manual evaluation.

4.5 Manual Evaluation

We hire five workers to manually evaluate the quality of 500 sentences generated by each of our model and the compared models. The result is shown in Table 5. "Content Preservation" measures whether the generated sentence preserves the content of the input sentence. The score range is {2: fully preserved, 1: partially preserved, 0: not preserved}. "Content Preservation" is an important metric in this task since it is required that the output sentence and the original sentence should describe the same content subject. "Fluency" measures the grammatical quality of a sentence, which

	Generated sentence
E.g. 1	this tire center is amazing .
T=1	this tire center is horrible .
T=3	this tire center is really good .
T=5	this tire center is amazing .
E.g. 2	horrible food !
T=1	horrendous
T=3	their food amazing !
T=5	amazing delicious food ! recommend !
E.g. 3	decent food and wine selection , but nothing i will rush back for .
T=1	decent food and wine selection , but nothing i will rush for no .
T=3	decent food and wine selection , but i will never look back for .
T=5	decent food and wine selection , but excellent service, will return !
E.g. 4	our first time and we had a great meal , wonderful service .
T=1	our first time and we had a terrible meal , stale service .
T=3	our first time and we had a great meal , we have service .
T=5	our first time and we had a great meal , wonderful service .
E.g. 5	food is very addiction tasty !
T=1	food is just horrible here ?
T=3	food is just addiction here !
T=5	food is very yummy addiction !

Table 6: Case study.

ranges from 1 (bad) to 4 (good), by following the definition in TST (Shen et al., 2017).

The result shows that our model achieves the best content preservation score. Our editing procedure explicitly fixes the content factor and only modifies the outcome factor, which helps better preserve the content. In contrast, S2BS and TST include only one shared factor for both the content and the outcome, thus fail to distinguish one from the other. For the "Fluency" metric, S2BS and TST are slightly better than our model. Generally speaking, it is because our model introduces more powerful components for modeling the outcome differences between pseudo-parallel sentences, so as to achieve our goal of editing an input sentence to satisfy the expected outcome. However, these components do not contribute to the language quality of generated sentences.

4.6 Case Study

We show some examples produced by our model in Table 6. For each input, we specify three targets: 1, 3, and 5. For the first and the forth examples, the original sentences are not revised when the target rating is set to 5 (i.e., T=5) since the original sentences are already quite positive. For the first example, when T=3, "amazing" is revised to a relatively less positive phrase "really good". This case demonstrates that our model is able to capture the subtle difference in word sentiments, so that it can revise sentences reasonably according to the quantifiable rating guidance. Moreover, for the second example, we notice that our model revises the original sentence "horrible food !" to "amazing delicious food ! recommend !" for T=5. This case shows that our model not only changes one word with another having different sentiment, e.g. "horrible" to "amazing delicious", but also creatively introduces words from a new perspective, e.g. "recommend".

4.7 Ablation and Tuning Behavior Discussions

Recall that our model is a combination of a revised VAE, which disentangles two factors from a sentence to enable the subsequent design, and a coupling component modeling pseudo-parallel sentence pairs. For the three losses of the coupling component, we show their effects under the MAE metric in Table 7. "None" refers to all three losses are removed, and it is basically worse than S2BS, which implies only using the revised VAE does not work well. As more losses added, the performance is gradually improved. Moreover, the dual reconstruction is more effective than the others.

	T=1	T=3	T=5
S2BS	1.6839	0.7567	1.3024
None	1.6639	0.7684	1.5434
$\mathcal{L}_{sim}$	1.6090	0.8258	1.5233
$\mathcal{L}_{diff}$	1.6793	0.8017	1.3140
$\mathcal{L}_{d-rec}$	1.5191	0.7784	1.1218
$\mathcal{L}_{sim}, \mathcal{L}_{diff}$	1.4991	0.8218	1.3705
$\mathcal{L}_{sim}, \mathcal{L}_{d-rec}$	1.4101	0.8027	1.1246
$\mathcal{L}_{diff}, \mathcal{L}_{d-rec}$	1.3879	0.7786	1.1413
ALL	1.4162	0.7408	0.9408

Table 7: Ablation study.

In the weight tuning, the first step only tunes the weights of $\mathcal{L}_{rec}$ and $\mathcal{L}_{mse}$. We observe that solely minimizing $\mathcal{L}_{rec}$ and $\mathcal{L}_{mse}$ also decreases $\mathcal{L}_{sim}$, because in this process, the encoder E_2 becomes more capable of disentangling the content factor, and thus z and z' become similar as they come from two similar input sentences, i.e. pseudo-parallel sentences. Another observation is that solely minimizing $\mathcal{L}_{rec}$ and $\mathcal{L}_{mse}$ increases $\mathcal{L}_{d-rec}$ after some training epochs. To analyze the reason, let us assume there is a sentence x in the training set. Thus, the loss of reconstructing x from y and z is included in $\mathcal{L}_{rec}$. Assume that x is also included in a pseudo-parallel pair, and thus the loss of reconstructing x from y and z' is included in $\mathcal{L}_{d-rec}$. The only difference between the two losses lies in the content factors z and z'. Given that z and z' are not enforced to resemble each other when $\mathcal{L}_{sim}$ is excluded from this tuning step, $\mathcal{L}_{rec}$ and $\mathcal{L}_{d-rec}$ cannot be minimized simultaneously. Moreover, when we minimize $\mathcal{L}_{sim}$ in the second step with the weights of $\mathcal{L}_{rec}$ and $\mathcal{L}_{mse}$ fixed, we observe that $\mathcal{L}_{d-rec}$ also decreases, which complies with the above analysis.

5 Related Works

Inspired by the task of image style transfer (Gatys et al., 2016; Liu and Tuzel, 2016), researchers proposed the task of text style transfer and obtained some encouraging results (Fu et al., 2018; Hu et al., 2017; Jhamtani et al., 2017; Melnyk et al., 2017; Zhang et al., 2018; Li et al., 2018; Prabhumoye et al., 2018; Niu and Bansal, 2018). Existing studies on text style transfer mainly aim at transferring text from an original style into a target style, e.g., from negative to positive, from male to female, from rude/normal to polite; from modern text to Shakespeare style, etc. In contrast, our proposed task QuaSE assumes each sentence is associated with an outcome pertaining to continues values, and the editing is under the guidance of a specific target.

To transfer the style of a sentence, the paradigm of most works (Shen et al., 2017; Mueller et al., 2017; Prabhumoye et al., 2018) first learns the latent representation of the original sentence and then applies a decoder to generate the transferred sentence. A line of works (Shen et al., 2017; Mueller et al., 2017), including the studied task in this paper, assume that only non-parallel data is available for training. In such settings, VAEs (Kingma and Welling, 2013) are employed to learn the latent representations of sentences. Shen et al. (2017) assume a shared latent content distribution across text corpora belonging to different styles, and leverages the alignment of latent representations from different styles to perform style transfer. Mueller et al. (2017) associate the latent representations with a numerical outcome, which is a measurement of the style. A transferred sentence is generated from a modified latent representation. Different from the aforementioned works based on latent representations, Li et al. (2018) propose a simpler method that achieves attribute transfer by changing a few attribute marker words or phrases in the sentence that are indicative of a particular attribute, while leaving the rest of the sentence largely unchanged. The simple method is able to generate better-quality sentences than the aforementioned works. Besides style transfer, sentence editing models can be developed for other tasks. For example, Schmaltz et al. (2017) propose neural sequence-labelling models for correcting the grammatical errors of sentences.

6 Conclusions

We proposed a new task namely Quantifiable Sequence Editing (QuaSE), where a model needs to edit an input sentences towards the direction of a numerical outcome target. To tackle this task, we proposed a novel framework that simultaneously exploits the single sentences and pseudo-parallel sentence pairs. For evaluation, we prepared a dataset with Yelp sentences and their ratings. Experimental results show that our framework outperforms the compared methods under the measures of sentiment polarity accuracy and target value errors. Case studies show that our framework can generate some interesting sentences.

References

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2016. Variational inference: A review for statisticians. *arXiv*, abs/1601.00670.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Yifan Gao, Jianan Wang, Lidong Bing, Irwin King, and Michael R. Lyu. 2018. Difficulty controllable question generation for reading comprehension. *CoRR*, abs/1807.03586.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of 2016 Conference on Computer Vision and Pattern Recognition*, pages 2414–2423.

K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics (TACL)*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1587–1596.

Harsh Jhamtani, Varun Gangal, Eduard H. Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence-to-sequence models. *arXiv*, abs/1707.01161.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv*, abs/1312.6114.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *The 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ming-Yu Liu and Oncel Tuzel. 2016. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 469–477.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Igor Melnyk, Cícero Nogueira dos Santos, Kahini Wadhawan, Inkit Padhi, and Abhishek Kumar. 2017. Improved neural text attribute transfer with non-parallel data. *arXiv*, abs/1711.09395.

Jonas Mueller, David Gifford, and Tommi Jaakkola. 2017. Sequence to better sequence: Continuous revision of combinatorial structures. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2536–2544.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics (TACL)*.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Allen Schmaltz, Yoon Kim, Alexander M. Rush, and Stuart M. Shieber. 2017. Adapting sequence models for sentence correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2807–2813.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval- and generation-based dialog systems. *arXiv*, abs/1610.07149.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Yufei Wang, Zhe Lin, Xiaohui Shen, Scott Cohen, and Garrison W. Cottrell. 2017. Skeleton key: Image captioning by skeleton-attribute decomposition. In *Proceedings of 2017 Conference on Computer Vision and Pattern Recognition*, pages 7378–7387.

Tong Xiao, Jingbo Zhu, Chunliang Zhang, and Tongran Liu. 2016. Syntactic skeleton-based translation. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2856–2862.

Ye Zhang, Nan Ding, and Radu Soricut. 2018. Shaped: Shared-private encoder-decoder for text style adaptation. In *The 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Paraphrase Generation with Deep Reinforcement Learning

Zichao Li[1], Xin Jiang[1], Lifeng Shang[1], Hang Li[2]
[1]Noah's Ark Lab, Huawei Technologies
{li.zichao, jiang.xin, shang.lifeng}@huawei.com
[2]Toutiao AI Lab
lihang.lh@bytedance.com

Abstract

Automatic generation of paraphrases from a given sentence is an important yet challenging task in natural language processing (NLP). In this paper, we present a deep reinforcement learning approach to paraphrase generation. Specifically, we propose a new framework for the task, which consists of a *generator* and an *evaluator*, both of which are learned from data. The generator, built as a sequence-to-sequence learning model, can produce paraphrases given a sentence. The evaluator, constructed as a deep matching model, can judge whether two sentences are paraphrases of each other. The generator is first trained by deep learning and then further fine-tuned by reinforcement learning in which the reward is given by the evaluator. For the learning of the evaluator, we propose two methods based on *supervised learning* and *inverse reinforcement learning* respectively, depending on the type of available training data. Experimental results on two datasets demonstrate the proposed models (the generators) can produce more accurate paraphrases and outperform the state-of-the-art methods in paraphrase generation in both automatic evaluation and human evaluation.

1 Introduction

Paraphrases refer to texts that convey the same meaning but with different expressions. For example, "*how far is Earth from Sun*", "*what is the distance between Sun and Earth*" are paraphrases. Paraphrase generation refers to a task in which given a sentence the system creates paraphrases of it. Paraphrase generation is an important task in NLP, which can be a key technology in many applications such as retrieval based question answering, semantic parsing, query reformulation in web search, data augmentation for dialogue system. However, due to the complexity of natural language, automatically generating accurate and diverse paraphrases is still very challenging. Traditional symbolic approaches to paraphrase generation include rule-based methods (McKeown, 1983), thesaurus-based methods (Bolshakov and Gelbukh, 2004; Kauchak and Barzilay, 2006), grammar-based methods (Narayan et al., 2016), and statistical machine translation (SMT) based methods (Quirk et al., 2004; Zhao et al., 2008, 2009).

Recently, neural network based sequence-to-sequence (Seq2Seq) learning has made remarkable success in various NLP tasks, including machine translation, short-text conversation, text summarization, and question answering (e.g., Cho et al. (2014); Wu et al. (2016); Shang et al. (2015); Vinyals and Le (2015); Rush et al. (2015); Yin et al. (2016)). Paraphrase generation can naturally be formulated as a Seq2Seq problem (Cao et al., 2017; Prakash et al., 2016; Gupta et al., 2018; Su and Yan, 2017). The main challenge in paraphrase generation lies in the definition of the evaluation measure. Ideally the measure is able to calculate the semantic similarity between a generated paraphrase and the given sentence. In a straightforward application of Seq2Seq to paraphrase generation one would make use of cross entropy as evaluation measure, which can only be a loose approximation of semantic similarity. To tackle this problem, Ranzato et al. (2016) propose employing reinforcement learning (RL) to guide the training of Seq2Seq and using lexical-based measures such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) as a reward function. However, these lexical measures may not perfectly represent semantic similarity. It is likely that a correctly generated sequence gets a low ROUGE score due to lexical mismatch. For instance, an input sentence "*how far is Earth from Sun*" can be paraphrased as "*what is the distance between Sun and Earth*", but it will

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3865–3878
Brussels, Belgium, October 31 - November 4, 2018.

get a very low ROUGE score given "*how many miles is it from Earth to Sun*" as a reference.

In this work, we propose taking a data-driven approach to train a model that can conduct evaluation in learning for paraphrasing generation. The framework contains two modules, a generator (for paraphrase generation) and an evaluator (for paraphrase evaluation). The generator is a Seq2Seq learning model with attention and copy mechanism (Bahdanau et al., 2015; See et al., 2017), which is first trained with cross entropy loss and then fine-tuned by using policy gradient with supervisions from the evaluator as rewards. The evaluator is a deep matching model, specifically a decomposable attention model (Parikh et al., 2016), which can be trained by supervised learning (SL) when both positive and negative examples are available as training data, or by inverse reinforcement learning (IRL) with outputs from the generator as supervisions when only positive examples are available. In the latter setting, for the training of evaluator using IRL, we develop a novel algorithm based on max-margin IRL principle (Ratliff et al., 2006). Moreover, the generator can be further trained with non-parallel data, which is particularly effective when the amount of parallel data is small.

We evaluate the effectiveness of our approach using two real-world datasets (Quora question pairs and Twitter URL paraphrase corpus) and we conduct both automatic and human assessments. We find that the evaluator trained by our methods can provide accurate supervisions to the generator, and thus further improve the accuracies of the generator. The experimental results indicate that our models can achieve significantly better performances than the existing neural network based methods.

It should be noted that the proposed approach is not limited to paraphrase generation and can be readily applied into other sequence-to-sequence tasks such as machine translation and generation based single turn dialogue. Our technical contribution in this work is of three-fold:

1. We introduce the generator-evaluator framework for paraphrase generation, or in general, sequence-to-sequence learning.
2. We propose two approaches to train the evaluator, i.e., supervised learning and inverse reinforcement learning.
3. In the above framework, we develop several techniques for learning of the generator and evaluator.

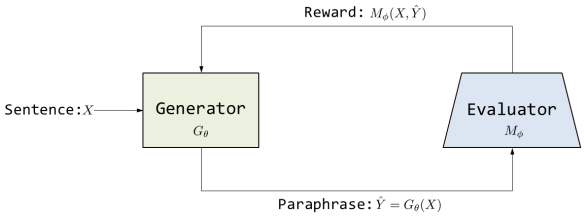

Figure 1: Framework of RbM (Reinforced by Matching).

Section 2 defines the models of generator and evaluator. In section 3, we formalize the problem of learning the models of generator and evaluator. In section 4, we report our experimental results. In section 5, we introduce related work.

2 Models

This section explains our framework for paraphrase generation, containing two models, the generator and evaluator.

2.1 Problem and Framework

Given an input sequence of words $X = [x_1, \ldots, x_S]$ with length S, we aim to generate an output sequence of words $Y = [y_1, \ldots, y_T]$ with length T that has the same meaning as X. We denote the pair of sentences in paraphrasing as (X, Y). We use $Y_{1:t}$ to denote the subsequence of Y ranging from 1 to t and use $\hat{Y}$ to denote the sequence generated by a model.

We propose a framework, which contains a generator and an evaluator, called **RbM** (Reinforced by Matching). Specifically, for the generator we adopt the Seq2Seq architecture with attention and copy mechanism (Bahdanau et al., 2015; See et al., 2017), and for the evaluator we adopt the decomposable attention-based deep matching model (Parikh et al., 2016). We denote the generator as G_θ and the evaluator as M_ϕ, where θ and ϕ represent their parameters respectively. Figure 1 gives an overview of our framework. Basically the generator can generate a paraphrase of a given sentence and the evaluator can judge how semantically similar the two sentences are.

2.2 Generator: Seq2Seq Model

In this work, paraphrase generation is defined as a sequence-to-sequence (Seq2Seq) learning problem. Given input sentence X, the goal is to learn a model G_θ that can generate a sentence

$\hat{Y} = G_\theta(X)$ as its paraphrase. We choose the *pointer-generator* proposed by See et al. (2017) as the generator. The model is built based on the encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014), both of which are implemented as recurrent neural networks (RNN). The encoder RNN transforms the input sequence X into a sequence of hidden states $H = [h_1, \ldots, h_S]$. The decoder RNN generates an output sentence Y on the basis of the hidden states. Specifically it predicts the next word at each position by sampling from $\hat{y}_t \sim p(y_t|Y_{1:t-1}, X) = g(s_t, c_t, y_{t-1})$, where s_t is the decoder state, c_t is the context vector, y_{t-1} is the previous word, and g is a feed-forward neural network. Attention mechanism (Bahdanau et al., 2015) is introduced to compute the context vector as the weighted sum of encoder states:

$$c_t = \sum_{i=1}^{S} \alpha_{ti} h_i, \quad \alpha_{ti} = \frac{\exp \eta(s_{t-1}, h_i)}{\sum_{j=1}^{S} \exp \eta(s_{t-1}, h_j)},$$

where α_{ti} represents the attention weight and η is the *attention function*, which is a feed-forward neural network.

Paraphrasing often needs copying words from the input sentence, for instance, named entities. The pointer-generator model allows either generating words from a vocabulary or copying words from the input sequence. Specifically the probability of generating the next word is given by a mixture model:

$$p_\theta(y_t|Y_{1:t-1}, X) = q(s_t, c_t, y_{t-1}) g(s_t, c_t, y_{t-1}) + (1 - q(s_t, c_t, y_{t-1})) \sum\nolimits_{i: y_t = x_i} \alpha_{ti},$$

where $q(s_t, c_t, y_{t-1})$ is a binary neural classifier deciding the probability of switching between the generation mode and the copying mode.

2.3 Evaluator: Deep Matching Model

In this work, paraphrase evaluation (identification) is casted as a problem of learning of sentence matching. The goal is to learn a real-valued function $M_\phi(X, Y)$ that can represent the matching degree between the two sentences as paraphrases of each other. A variety of learning techniques have been developed for matching sentences, from linear models (e.g., Wu et al. (2013)) to neural network based models (e.g., Socher et al. (2011); Hu et al. (2014)). We choose a simple yet effective neural network architecture, called the *decomposable-attention* model (Parikh et al., 2016), as the evaluator. The evaluator can calculate the semantic similarity between two sentences:

$$M_\phi(X, Y) = H(\sum_{i=1}^{S} G([e(x_i), \bar{x}_i]), \sum_{j=1}^{T} G([e(y_j), \bar{y}_j])),$$

where $e(\cdot)$ denotes a word embedding, $\bar{x}_i$ and $\bar{y}_j$ denote inter-attended vectors, H and G are feed-forward networks. We refer the reader to Parikh et al. (2016) for details. In addition, we add *positional encodings* to the word embedding vectors to incorporate the order information of the words, following the idea in Vaswani et al. (2017).

3 Learning

This section explains how to learn the generator and evaluator using deep reinforcement learning.

3.1 Learning of Generator

Given training data (X, Y), the generator G_θ is first trained to maximize the conditional log likelihood (negative cross entropy):

$$\mathcal{L}_{\text{Seq2Seq}}(\theta) = \sum\nolimits_{t=1}^{T} \log p_\theta(y_t|Y_{1:t-1}, X). \quad (1)$$

When computing the conditional probability of the next word as above, we choose the previous word y_{t-1} in the ground-truth rather than the word $\hat{y}_{t-1}$ generated by the model. This technique is called *teacher forcing*.

With teacher forcing, the discrepancy between training and prediction (also referred to as *exposure bias*) can quickly accumulate errors along the generated sequence (Bengio et al., 2015; Ranzato et al., 2016). Therefore, the generator G_θ is next fine-tuned using RL, where the reward is given by the evaluator.

In the RL formulation, generation of the next word represents an *action*, the previous words represent a *state*, and the probability of generation $p_\theta(y_t|Y_{1:t-1}, X)$ induces a stochastic *policy*. Let r_t denote the *reward* at position t. The goal of RL is to find a policy (i.e., a generator) that maximizes the expected cumulative reward:

$$\mathcal{L}_{RL}(\theta) = \mathbb{E}_{p_\theta(\hat{Y}|X)} \sum_{t=1}^{T} r_t(X, \hat{Y}_{1:t}). \quad (2)$$

We define a positive reward at the end of sequence ($r_T = R$) and a zero reward at the other positions. The reward R is given by the evaluator M_ϕ. In particular, when a pair of input sentence X and generated paraphrase $\hat{Y} = G_\theta(X)$ is given, the reward is calculated by the evaluator $R = M_\phi(X, \hat{Y})$.

We can then learn the optimal policy by employing policy gradient. According to the policy gradient theorem (Williams, 1992; Sutton et al., 2000), the gradient of the expected cumulative reward can be calculated by

$$\nabla_\theta \mathcal{L}_{RL}(\theta) = \sum_{t=1}^{T} [\nabla_\theta \log p_\theta(\hat{y}_t | \hat{Y}_{1:t-1}, X)] r_t. \tag{3}$$

The generator can thus be learned with stochastic gradient descent methods such as Adam (Kingma and Ba, 2015).

3.2 Learning of Evaluator

The evaluator works as the reward function in RL of the generator and thus is essential for the task. We propose two methods for learning the evaluator in different settings. When there are both positive and negative examples of paraphrases, the evaluator is trained by *supervised learning* (SL). When only positive examples are available (usually the same data as the training data of the generator), the evaluator is trained by *inverse reinforcement learning* (IRL).

Supervised Learning

Given a set of positive and negative examples (paraphrase pairs), we conduct supervised learning of the evaluator with the pointwise cross entropy loss:

$$\mathcal{J}_{\mathrm{SL}}(\phi) = -\log M_\phi(X, Y) - \log(1 - M_\phi(X, Y^-)), \tag{4}$$

where Y^- represents a sentence that is not a paraphrase of X. The evaluator M_ϕ here is defined as a classifier, trained to distinguish negative example (X, Y^-) from positive example (X, Y).

We call this method **RbM-SL** (Reinforced by Matching with Supervised Learning). The evaluator M_ϕ trained by supervised learning can make a judgement on whether two sentences are paraphrases of each other. With a well-trained evaluator M_ϕ, we further train the generator G_θ by reinforcement learning using M_ϕ as a reward function. Figure 2a shows the learning process of RbM-SL. The detailed training procedure is shown in Algorithm 1 in Appendix A.

Inverse Reinforcement Learning

Inverse reinforcement learning (IRL) is a sub-problem of reinforcement learning (RL), about learning of a reward function given *expert demonstrations*, which are sequences of states and actions from an expert (optimal) policy. More specifically, the goal is to find an optimal reward function R^* with which the expert policy $p_{\theta^*}(Y|X)$ really becomes optimal among all possible policies, i.e.,

$$\mathbb{E}_{p_{\theta^*}(Y|X)} R^*(Y) \geq \mathbb{E}_{p_\theta(\hat{Y}|X)} R^*(\hat{Y}), \;\; \forall \theta.$$

In the current problem setting, the problem becomes learning of an optimal reward function (evaluator) given a number of paraphrase pairs given by human experts (expert demonstrations).

To learn an optimal reward (matching) function is challenging, because the expert demonstrations might not be optimal and the reward function might not be rigorously defined. To deal with the problem, we employ the maximum margin formulation of IRL inspired by Ratliff et al. (2006).

The maximum margin approach ensures the learned reward function has the following two desirable properties in the paraphrase generation task: (a) given the same input sentence, a reference from humans should have a higher reward than the ones generated by the model; (b) the margins between the rewards should become smaller when the paraphrases generated by the model get closer to a reference given by humans. We thus specifically consider the following optimization problem for learning of the evaluator:

$$\mathcal{J}_{\mathrm{IRL}}(\phi) = \max(0, 1 - \zeta + M_\phi(X, \hat{Y}) - M_\phi(X, Y)), \tag{5}$$

where ζ is a slack variable to measure the agreement between $\hat{Y}$ and Y. In practice we set $\zeta = \text{ROUGE-L}(\hat{Y}, Y)$. Different from RbM-SL, the evaluator M_ϕ here is defined as a ranking model that assigns higher rewards to more plausible paraphrases.

Once the reward function (evaluator) is learned, it is then used to improve the policy function (generator) through policy gradient. In fact, the generator G_θ and the evaluator M_ϕ are trained alternatively. We call this method **RbM-IRL** (Reinforced by Matching with Inverse Reinforcement Learning). Figure 2b shows the learning process of RbM-IRL. The detailed training procedure is shown in Algorithm 2 in Appendix A.

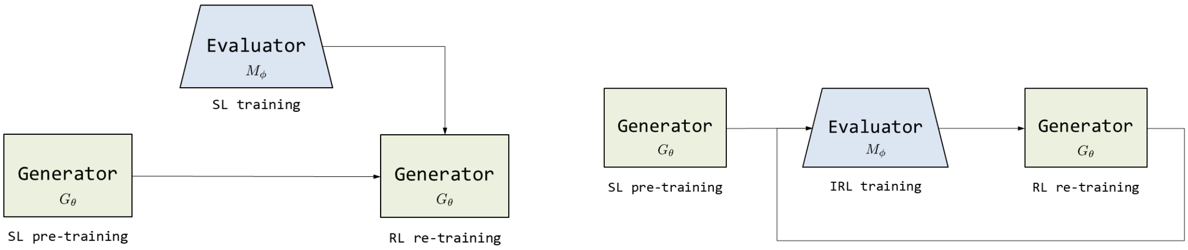

(a) RbM-SL

(b) RbM-IRL

Figure 2: Learning Process of RbM models: (a) RbM-SL, (b) RbM-IRL.

We can formalize the whole learning procedure as the following optimization problem:

$$\min_{\phi} \max_{\theta} \mathbb{E}_{p_\theta(\hat{Y}|X)} \mathcal{J}_{\text{IRL}}(\phi). \tag{6}$$

RbM-IRL can make effective use of sequences generated by the generator for training of the evaluator. As the generated sentences become closer to the ground-truth, the evaluator also becomes more discriminative in identifying paraphrases.

It should be also noted that for both RbM-SL and RbM-IRL, once the evaluator is learned, the reinforcement learning of the generator only needs *non-parallel sentences* as input. This makes it possible to further train the generator and enhance the generalization ability of the generator.

3.3 Training Techniques

Reward Shaping

In the original RL of the generator, only a positive reward R is given at the end of sentence. This provides sparse supervision signals and can make the model greatly degenerate. Inspired by the idea of reward shaping (Ng et al., 1999; Bahdanau et al., 2017), we estimate the intermediate cumulative reward (value function) for each position, that is

$$Q_t = \mathbb{E}_{p_\theta(Y_{t+1:T}|\hat{Y}_{1:t},X)} R(X, [\hat{Y}_{1:t}, Y_{t+1:T}]),$$

by Monte-Carlo simulation, in the same way as in Yu et al. (2017):

$$Q_t = \begin{cases} \frac{1}{N}\sum_{n=1}^{n=N} M_\phi(X, [\hat{Y}_{1:t}, \widehat{Y}^n_{t+1:T}]), & t < T \\ M_\phi(X, \hat{Y}), & t = T, \end{cases} \tag{7}$$

where N is the sample size and $\widehat{Y}^n_{t+1:T} \sim p_\theta(Y_{t+1:T}|\hat{Y}_{1:t}, X)$ denotes simulated sub-sequences randomly sampled starting from the $(t+1)$-th word. During training of the generator, the reward r_t in policy gradient (3) is replaced by Q_t estimated in (7).

Reward Rescaling

In practice, RL algorithms often suffer from instability in training. A common approach to reduce the variance is to subtract a baseline reward from the value function. For instance, a simple baseline can be a moving average of historical rewards. While in RbM-IRL, the evaluator keeps updating during training. Thus, keeping track of a baseline reward is unstable and inefficient. Inspired by Guo et al. (2018), we propose an efficient reward rescaling method based on ranking. For a batch of D generated paraphrases $\{\hat{Y}^d\}_{d=1}^D$, each associated with a reward $R^d = M_\phi(X^d, \hat{Y}^d)$, we rescale the rewards by

$$\bar{R}^d = \sigma(\delta_1 \cdot (0.5 - \frac{\text{rank}(d)}{D})) - 0.5, \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function, $\text{rank}(d)$ is the rank of R^d in $\{R^1, ..., R^D\}$, and δ_1 is a scalar controlling the variance of rewards. A similar strategy is applied into estimation of in-sequence value function for each word, and the final rescaled value function is

$$\bar{Q}^d_t = \sigma(\delta_2 \cdot (0.5 - \frac{\text{rank}(t)}{T})) - 0.5 + \bar{R}^d, \tag{9}$$

where $\text{rank}(t)$ is the rank of Q^d_t in $\{Q^d_1, ..., Q^d_T\}$.

Reward rescaling has two advantages. First, the mean and variance of $\bar{Q}^d_t$ are controlled and hence they make the policy gradient more stable, even with a varying reward function. Second, when the evaluator M_ϕ is trained with the ranking loss as in RbM-IRL, it is better to inform which paraphrase is better, rather than to provide a scalar reward in a range. In our experiment, we find that this method can bring substantial gains for RbM-SL and RbM-IRL, but not for RL with ROUGE as reward.

Curriculum Learning

RbM-IRL may not achieve its best performance if all of the training instances are included in training at the beginning. We employ a curriculum learning strategy (Bengio et al., 2009) for it. During the training of the evaluator M_ϕ, each example k is associated with a weight w^k, i.e.

$$\mathcal{J}^k_{\text{IRL-CL}}(\phi) = w^k \max(0,1 - \zeta^k + M_\phi(X^k, \hat{Y}^k) - M_\phi(X^k, Y^k)) \quad (10)$$

In curriculum learning, w^k is determined by the difficulty of the example. At the beginning, the training procedure concentrates on relatively simple examples, and gradually puts more weights on difficult ones. In our case, we use the edit distance $\mathcal{E}(X, Y)$ between X and Y as the measure of difficulty for paraphrasing. Specifically, w^k is determined by $w^k \sim \text{Binomial}(p^k, 1)$, and $p^k = \sigma(\delta_3 \cdot (0.5 - \frac{\text{rank}(\mathcal{E}(X^k,Y^k))}{K}))$, where K denotes the batch size for training the evaluator. For δ_3, we start with a relatively high value and gradually decrease it. In the end each example will be sampled with a probability around 0.5. In this manner, the evaluator first learns to identify paraphrases with small modifications on the input sentences (e.g. "*what 's*" and "*what is*"). Along with training it gradually learns to handle more complicated paraphrases (e.g. "*how can I*" and "*what is the best way to*").

4 Experiment

4.1 Baselines and Evaluation Measures

To compare our methods (RbM-SL and RbM-IRL) with existing neural network based methods, we choose five baseline models: the attentive Seq2Seq model (Bahdanau et al., 2015), the stacked Residual LSTM networks (Prakash et al., 2016), the variational auto-encoder (VAE-SVG-eq) (Gupta et al., 2018) [1], the pointer-generator (See et al., 2017), and the reinforced pointer-generator with ROUGE-2 as reward (RL-ROUGE) (Ranzato et al., 2016).

We conduct both automatic and manual evaluation on the models. For the automatic evaluation, we adopt four evaluation measures: ROUGE-1, ROUGE-2 (Lin, 2004), BLEU (Papineni et al., 2002) (up to at most bi-grams) and METEOR (Lavie and Agarwal, 2007). As pointed out, ideally it would be better not to merely use a lexical measure like ROUGE or BLEU for evaluation of paraphrasing. We choose to use them for reproducibility of our experimental results by others. For the manual evaluation, we conduct evaluation on the generated paraphrases in terms of relevance and fluency.

4.2 Datasets

We evaluate our methods with the **Quora** question pair dataset [2] and **Twitter** URL paraphrasing corpus (Lan et al., 2017). Both datasets contain positive and negative examples of paraphrases so that we can evaluate the RbM-SL and RbM-IRL methods. We randomly split the Quora dataset in two different ways obtaining two experimental settings: Quora-I and Quora-II. In Quora-I, we partition the dataset by question pairs, while in Quora-II, we partition by question ids such that there is no shared question between the training and test/validation datasets. In addition, we sample a smaller training set in Quora-II to make the task more challenging. Twitter URL paraphrasing corpus contains two subsets, one is labeled by human annotators while the other is labeled automatically by algorithm. We sample the test and validation set from the labeled subset, while using the remaining pairs as training set. For RbM-SL, we use the labeled subset to train the evaluator M_ϕ. Compared to Quora-I, it is more difficult to achieve a high performance with Quora-II. The Twitter corpus is even more challenging since the data contains more noise. The basic statistics of datasets are shown in Table 1.

Table 1: Statistics of datasets.

	Generator			Evaluator (RbM-SL)	
Dataset	#Train	#Test	#Validation	#Positive	#Negative
Quora-I	100K	30K	3K	100K	160K
Quora-II	50K	30K	3K	50K	160K
Twitter	110K	5K	1K	10K	40K

4.3 Implementation Details

Generator We maintain a fixed-size vocabulary of 5K shared by the words in input and output, and truncate all the sentences longer than 20 words. The model architecture, word embedding size and LSTM cell size are as the same as reported in See et al. (2017). We use Adadgrad optimizer (Duchi et al., 2011) in the supervised pre-training and Adam optimizer in the reinforcement learning, with the batch size of 80. We also fine-tune the

---

[1] We directly present the results reported in Gupta et al. (2018) on the same dateset and settings.

[2] `https://www.kaggle.com/c/quora-question-pairs`

Table 2: Performances on Quora datasets.

	Quora-I				Quora-II			
Models	ROUGE-1	ROUGE-2	BLEU	METEOR	ROUGE-1	ROUGE-2	BLEU	METEOR
Seq2Seq	58.77	31.47	36.55	26.28	47.22	20.72	26.06	20.35
Residual LSTM	59.21	32.43	37.38	28.17	48.55	22.48	27.32	22.37
VAE-SVG-eq	-	-	-	25.50	-	-	-	22.20
Pointer-generator	61.96	36.07	40.55	30.21	51.98	25.16	30.01	24.31
RL-ROUGE	63.35	37.33	41.83	30.96	54.50	27.50	32.54	25.67
RbM-SL (ours)	**64.39**	**38.11**	**43.54**	**32.84**	**57.34**	**31.09**	**35.81**	**28.12**
RbM-IRL (ours)	64.02	37.72	43.09	31.97	56.86	29.90	34.79	26.67

Table 3: Performances on Twitter corpus.

	Twitter			
Models	ROUGE-1	ROUGE-2	BLEU	METEOR
Seq2Seq	30.43	14.61	30.54	12.80
Residual LSTM	32.50	16.86	33.90	13.65
Pointer-generator	38.31	21.22	40.37	17.62
RL-ROUGE	40.16	22.99	42.73	18.89
RbM-SL (ours)	41.87	24.23	44.67	19.97
RbM-IRL (ours)	**42.15**	**24.73**	**45.74**	**20.18**

Table 4: Human evaluation on Quora datasets.

	Quora-I		Quora-II	
Models	Relevance	Fluency	Relevance	Fluency
Pointer-generator	3.23	4.55	2.34	2.96
RL-ROUGE	3.56	4.61	2.58	3.14
RbM-SL (ours)	**4.08**	4.67	**3.20**	3.48
RbM-IRL (ours)	4.07	**4.69**	2.80	**3.53**
Reference	4.69	4.95	4.68	4.90

Seq2Seq baseline models with Adam optimizer for a fair comparison. In supervised pre-training, we set the learning rate as 0.1 and initial accumulator as 0.1. The maximum norm of gradient is set as 2. During the RL training, the learning rate decreases to 1e-5 and the size of Monte-Carlo sample is 4. To make the training more stable, we use the ground-truth with reward of 0.1.

Evaluator We use the pretrained GoogleNews 300-dimension word vectors [3] in Quora dataset and 200-dimension GloVe word vectors [4] in Twitter corpus. Other model settings are the same as in Parikh et al. (2016). For evaluator in RbM-SL we set the learning rate as 0.05 and the batch size as 32. For the evaluator of M_ϕ in RbM-IRL, the learning rate decreases to 1e-2, and we use the batch size of 80.

We use the technique of reward rescaling as mentioned in section 3.3 in training RbM-SL and RbM-IRL. In RbM-SL, we set δ_1 as 12 and δ_2 as 1. In RbM-IRL, we keep δ_2 as 1 all the time and decrease δ_1 from 12 to 3 and δ_3 from 15 to 8 during curriculum learning. In ROUGE-RL, we take the exponential moving average of historical rewards as baseline reward to stabilize the training:

$$b_m = \lambda \overline{Q}_{m-1} + (1-\lambda) b_{m-1},\ b_1 = 0$$

where b_m is the baseline b at iteration m, $\overline{Q}$ is the mean value of reward, and we set λ as 0.1 by grid search.

4.4 Results and Analysis

Automatic evaluation Table 2 shows the performances of the models on Quora datasets. In both settings, we find that the proposed RbM-SL and RbM-IRL models outperform the baseline models in terms of all the evaluation measures. Particularly in Quora-II, RbM-SL and RbM-IRL make significant improvements over the baselines, which demonstrates their higher ability in learning for paraphrase generation. On Quora dataset, RbM-SL is constantly better than RbM-IRL for all the automatic measures, which is reasonable because RbM-SL makes use of additional labeled data to train the evaluator. Quora datasets contains a large number of high-quality non-paraphrases, i.e., they are literally similar but semantically different, for instance "*are analogue clocks better than digital*" and "*is analogue better than digital*". Trained with the data, the evaluator tends to become more capable in paraphrase identification. With additional evaluation on Quora data, the evaluator used in RbM-SL can achieve an accuracy of 87% on identifying positive and negative pairs of paraphrases.

Table 3 shows the performances on the Twitter corpus. Our models again outperform the baselines in terms of all the evaluation measures. Note that RbM-IRL performs better than RbM-SL in this case. The reason might be that the evaluator

---

[3] `https://code.google.com/archive/p/word2vec/`

[4] `https://nlp.stanford.edu/projects/glove/`

of RbM-SL might not be effectively trained with the relatively small dataset, while RbM-IRL can leverage its advantage in learning of the evaluator with less data.

In our experiments, we find that the training techniques proposed in section 3.3 are all necessary and effective. Reward shaping is by default employed by all the RL based models. Reward rescaling works particularly well for the RbM models, where the reward functions are learned from data. Without reward rescaling, RbM-SL can still outperform the baselines but with smaller margins. For RbM-IRL, curriculum learning is necessary for its best performance. Without curriculum learning, RbM-IRL only has comparable performance with ROUGE-RL.

Human evaluation We randomly select 300 sentences from the test data as input and generate paraphrases using different models. The pairs of paraphrases are then aggregated and partitioned into seven random buckets for seven human assessors to evaluate. The assessors are asked to rate each sentence pair according to the following two criteria: *relevance* (the paraphrase sentence is semantically close to the original sentence) and *fluency* (the paraphrase sentence is fluent as a natural language sentence, and the grammar is correct). Hence each assessor gives two scores to each paraphrase, both ranging from 1 to 5. To reduce the evaluation variance, there is a detailed evaluation guideline for the assessors in Appendix B. Each paraphrase is rated by two assessors, and then averaged as the final judgement. The agreement between assessors is moderate (kappa=0.44).

Table 4 demonstrates the average ratings for each model, including the ground-truth references. Our models of RbM-SL and RbM-IRL get better scores in terms of relevance and fluency than the baseline models, and their differences are statistically significant (paired t-test, p-value < 0.01). We note that in human evaluation, RbM-SL achieves the best relevance score while RbM-IRL achieves the best fluency score.

Case study Figure 3 gives some examples of generated paraphrases by the models on Quora-II for illustration. The first and second examples show the superior performances of RbM-SL and RbM-IRL over the other models. In the third example, both RbM-SL and RbM-IRL capture accurate paraphrasing patterns, while the other models wrongly segment and copy words from the input sentence. Compared to RbM-SL with an error of repeating the word *scripting*, RbM-IRL generates a more fluent paraphrase. The reason is that the evaluator in RbM-IRL is more capable of measuring the fluency of a sentence. In the fourth example, RL-ROUGE generates a totally non-sense sentence, and pointer-generator and RbM-IRL just cover half of the content of the original sentence, while RbM-SL successfully rephrases and preserves all the meaning. All of the models fail in the last example, because the word *ducking* is a rare word that never appears in the training data. Pointer-generator and RL-ROUGE generate totally irrelevant words such as *UNK* token or *victory*, while RbM-SL and RbM-IRL still generate topic-relevant words.

5 Related Work

Neural paraphrase generation recently draws attention in different application scenarios. The task is often formalized as a sequence-to-sequence (Seq2Seq) learning problem. Prakash et al. (2016) employ a stacked residual LSTM network in the Seq2Seq model to enlarge the model capacity. Cao et al. (2017) utilize an additional vocabulary to restrict word candidates during generation. Gupta et al. (2018) use a variational auto-encoder framework to generate more diverse paraphrases. Ma et al. (2018) utilize an attention layer instead of a linear mapping in the decoder to pick up word candidates. Iyyer et al. (2018) harness syntactic information for controllable paraphrase generation. Zhang and Lapata (2017) tackle a similar task of sentence simplification withe Seq2Seq model coupled with deep reinforcement learning, in which the reward function is manually defined for the task. Similar to these works, we also pretrain the paraphrase generator within the Seq2Seq framework. The main difference lies in that we use another trainable neural network, referred to as evaluator, to guide the training of the generator through reinforcement learning.

There is also work on paraphrasing generation in different settings. For example, Mallinson et al. (2017) leverage bilingual data to produce paraphrases by pivoting over a shared translation in another language. Wieting et al. (2017); Wieting and Gimpel (2018) use neural machine translation to generate paraphrases via back-translation of bilingual sentence pairs. Buck et al. (2018) and Dong et al. (2017) tackle the problem of QA-specific paraphrasing with the guidance from an external

Figure 3: Examples of the generated paraphrases by different models on Quora-II.

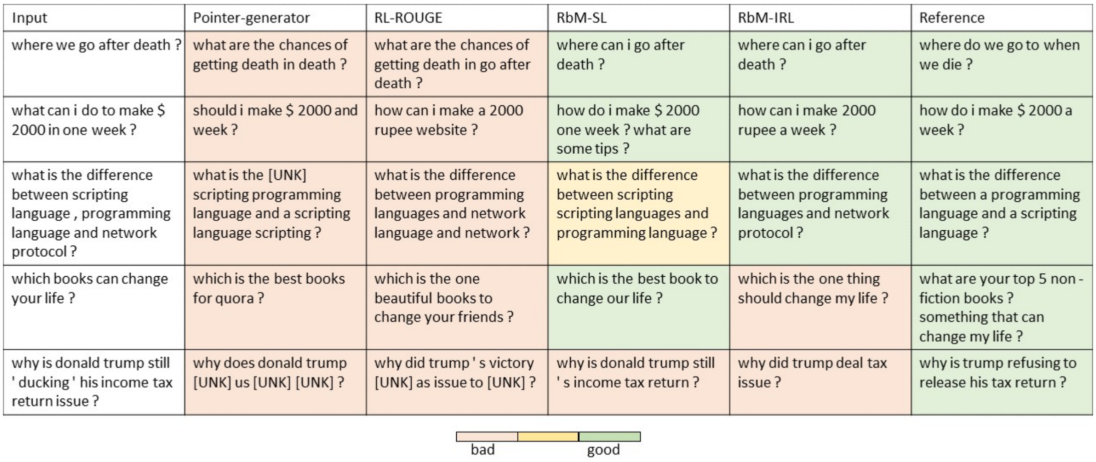

Input	Pointer-generator	RL-ROUGE	RbM-SL	RbM-IRL	Reference
where we go after death ?	what are the chances of getting death in death ?	what are the chances of getting death in go after death ?	where can i go after death ?	where can i go after death ?	where do we go to when we die ?
what can i do to make $ 2000 in one week ?	should i make $ 2000 and week ?	how can i make a 2000 rupee website ?	how do i make $ 2000 one week ? what are some tips ?	how can i make 2000 rupee a week ?	how do i make $ 2000 a week ?
what is the difference between scripting language , programming language and network protocol ?	what is the [UNK] scripting programming language and a scripting language scripting ?	what is the difference between programming languages and network language and network ?	what is the difference between scripting scripting languages and programming language ?	what is the difference between programming languages and network protocol ?	what is the difference between a programming language and a scripting language ?
which books can change your life ?	which is the best books for quora ?	which is the one beautiful books to change your friends ?	which is the best book to change our life ?	which is the one thing should change my life ?	what are your top 5 non - fiction books ? something that can change my life ?
why is donald trump still ' ducking ' his income tax return issue ?	why does donald trump [UNK] us [UNK] [UNK] ?	why did trump ' s victory [UNK] as issue to [UNK] ?	why is donald trump still ' s income tax return ?	why did trump deal tax issue ?	why is trump refusing to release his tax return ?

QA system and an associated evaluation metric.

Inverse reinforcement learning (IRL) aims to learn a reward function from expert demonstrations. Abbeel and Ng (2004) propose *apprenticeship learning*, which uses a feature based linear reward function and learns to match feature expectations. Ratliff et al. (2006) cast the problem as structured maximum margin prediction. Ziebart et al. (2008) propose max entropy IRL in order to solve the problem of expert suboptimality. Recent work involving deep learning in IRL includes Finn et al. (2016b) and Ho et al. (2016). There does not seem to be much work on IRL for NLP. In Neu and Szepesvári (2009), parsing is formalized as a feature expectation matching problem. Wang et al. (2018) apply adversarial inverse reinforcement learning in visual story telling. To the best of our knowledge, our work is the first that applies deep IRL into a Seq2Seq task.

Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) is a family of unsupervised generative models. GAN contains a generator and a discriminator, respectively for generating examples from random noises and distinguishing generated examples from real examples, and they are trained in an adversarial way. There are applications of GAN on NLP, such as text generation (Yu et al., 2017; Guo et al., 2018) and dialogue generation (Li et al., 2017). RankGAN (Lin et al., 2017) is the one most similar to RbM-IRL that employs a ranking model as the discriminator. However, RankGAN works for text generation rather than sequence-to-sequence learning, and training of generator in RankGAN relies on parallel data while the training of RbM-IRL can use non-parallel data.

There are connections between GAN and IRL as pointed by Finn et al. (2016a); Ho and Ermon (2016). However, there are significant differences between GAN and our RbM-IRL model. GAN employs the discriminator to distinguish generated examples from real examples, while RbM-IRL employs the evaluator as a reward function in RL. The generator in GAN is trained to maximize the loss of the discriminator in an adversarial way, while the generator in RbM-IRL is trained to maximize the expected cumulative reward from the evaluator.

6 Conclusion

In this paper, we have proposed a novel deep reinforcement learning approach to paraphrase generation, with a new framework consisting of a generator and an evaluator, modeled as sequence-to-sequence learning model and deep matching model respectively. The generator, which is for paraphrase generation, is first trained via sequence-to-sequence learning. The evaluator, which is for paraphrase identification, is then trained via supervised learning or inverse reinforcement learning in different settings. With a well-trained evaluator, the generator is further fine-tuned by reinforcement learning to produce more accurate paraphrases. The experiment results demonstrate that the proposed method can significantly improve the quality of paraphrase generation upon the baseline methods. In the future, we plan to apply the framework and training techniques into other tasks, such as machine translation and dialogue.

Acknowledgments

This work is supported by China National 973 Program 2014CB340301.

References

Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *ICML*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

Igor Bolshakov and Alexander Gelbukh. 2004. Synonymous paraphrasing using wordnet and internet. *Natural Language Processing and Information Systems*, pages 189–200.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *ICLR*.

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase. In *AAAI*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *EMNLP*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. 2016a. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *NIPS 2016 Workshop on Adversarial Training*.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016b. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *AAAI*.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *AAAI*.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *NIPS*.

Jonathan Ho, Jayesh Gupta, and Stefano Ermon. 2016. Model-free imitation learning with policy optimization. In *ICML*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL*.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *NAACL*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *EMNLP*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *EMNLP*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL-04 workshop*.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *NIPS*.

Shuming Ma, Xu Sun, Wei Li, Sujian Li, Wenjie Li, and Xuancheng Ren. 2018. Word embedding attention network: Generating words by querying distributed word representations for paraphrase generation. In *NAACL*.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *EACL*.

Kathleen R McKeown. 1983. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10.

Shashi Narayan, Siva Reddy, and Shay B Cohen. 2016. Paraphrase generation from latent-variable pcfgs for semantic parsing. In *INLG*.

Gergely Neu and Csaba Szepesvári. 2009. Training parsers by inverse reinforcement learning. *Machine learning*, 77(2):303–337.

Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual lstm networks. In *COLING*.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *EMNLP*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2006. Maximum margin planning. In *ICML*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *ACL*.

John Wieting and Kevin Gimpel. 2018. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *ACL*.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. In *EMNLP*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wei Wu, Zhengdong Lu, and Hang Li. 2013. Learning bilinear model for matching queries and documents. *The Journal of Machine Learning Research*, 14(1):2519–2548.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *IJCAI*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *EMNLP*.

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *ACL*.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008. Combining multiple resources to improve smt-based paraphrasing model. In *ACL*.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*.

A Algorithms of RbM-SL and RbM-IRL

Algorithm 1: Training Procedure of RbM-SL

```
Input : A corpus of paraphrase pairs $\{(X, Y)\}$, a corpus of non-paraphrase pairs $\{(X, Y^-)\}$, a corpus of (non-parallel) sentences $\{X\}$.
Output: Generator $G_{\theta'}$
Train the evaluator $M_\phi$ with $\{(X, Y)\}$ and $\{(X, Y^-)\}$;
Pre-train the generator $G_\theta$ with $\{(X, Y)\}$;
Init $G_{\theta'} := G_\theta$;
while not converge do
    Sample a sentence $X = [x_1, \dots, x_S]$ from the paraphrase corpus or the non-parallel corpus;
    Generate a sentence $\hat{Y} = [\hat{y}_1, \dots, \hat{y}_T]$ according to $G_{\theta'}$ given input $X$;
    Set the gradient $g_{\theta'} = 0$;
    for $t = 1$ to $T$ do
        Run $N$ Monte Carlo simulations: $\{\widehat{Y}^1_{t+1:T}, ...\widehat{Y}^N_{t+1:T}\} \sim p_{\theta'}(Y_{t+1:T}|\hat{Y}_{1:t}, X)$;
        Compute the value function by
            $Q_t = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} M_\phi(X, [\hat{Y}_{1:t}, \widehat{Y}^n_{t+1:T}]), & t < T \\ M_\phi(X, \hat{Y}), & t = T. \end{cases}$
        Rescale the reward to $\bar{Q}_t$ by (8);
        Accumulate $\theta'$-gradient: $g_{\theta'} := g_{\theta'} + \nabla_\theta \log p_{\theta'}(\hat{y}_t|\hat{Y}_{1:t-1}, X)\bar{Q}_t$
    end
    Update $G_{\theta'}$ using the gradient $g_{\theta'}$ with learning rate $\gamma_G$: $G_{\theta'} := G_{\theta'} + \gamma_G g_{\theta'}$
end
Return $G_{\theta'}$
```

Algorithm 2: Training Procedure of RbM-IRL

```
Input : A corpus of paraphrase pairs $\{(X, Y)\}$, a corpus of (non-parallel) sentences $\{X\}$.
Output: Generator $G_{\theta'}$, evaluator $M_{\phi'}$
Pre-train the generator $G_\theta$ with $\{(X, Y)\}$;
Init $G_{\theta'} := G_\theta$ and $M_{\phi'}$;
while not converge do
    while not converge do
        Sample a sentence $X = [x_1, \dots, x_S]$ from the paraphrase corpus;
        Generate a sentence $\hat{Y} = [\hat{y}_1, \dots, \hat{y}_T]$ according to $G_{\theta'}$ given input $X$;
        Calculate $\phi'$-gradient: $g_{\phi'} := \nabla_\phi \mathcal{J}_{\text{IRL-CL}}(\phi)$;
        Update $M_{\phi'}$ using the gradient $g_{\phi'}$ with learning rate $\gamma_M$: $M_{\phi'} := M_{\phi'} - \gamma_M g_{\phi'}$
    end
    Train $G_{\theta'}$ with $M_{\phi'}$ as in Algorithm 1;
end
Return $G_{\theta'}$, $M_{\phi'}$
```

B Human Evaluation Guideline

Please judge the paraphrases from the following two criteria:

(1) **Grammar and Fluency**: the paraphrase is acceptable as natural language text, and the grammar is correct;

(2) **Coherent and Consistent**: please view from the perspective of the original poster, to what extent the answer of paraphrase is helpful for you with respect to the original question. Specifically, you can consider following aspects:

- Relatedness: it should be topically relevant to the original question.
- Type of question: the type of the original question remains the same in paraphrase.
- Informative: no information loss in paraphrase.

For each paraphrase, give two separate score ranking from 1 to 5. The meaning of specific score is as following:

- Grammar and Fluency
  - 5: Without any grammatical error;
  - 4: Fluent and has one minor grammatical error that does not affect understanding, e.g. *what is the best ways to learn programming*;
  - 3: Basically fluent and has two or more minor grammatical errors or one serious grammatical error that does not have strong impact on understanding, e.g. *what some good book for read*;
  - 2: Can not understand what it means but it is still in the form of human language, e.g. *what is the best movie of movie*;
  - 1: Non-sense composition of words and not in the form of human language, e.g. *how world war iii world war*.
- Coherent and Consistent
  - 5: Accurate paraphrase with exact the same meaning of the source sentence;
  - 4: Basically the same meaning of the source sentence but does not cover some minor content, e.g. *what are some good places to visit in hong kong during summer → can you suggest some places to visit in hong kong*;
  - 3: Cover part of the content of source sentence and has serious information loss, e.g. *what is the best love movie by wong ka wai → what is the best movie*;
  - 2: Topic relevant but fail to cover most of the content of source sentence, e.g. *what is some tips to learn english → when do you start to learn english*;
  - 1: Topic irrelevant or even can not understand what it means.

There is token *[UNK]* that stands for unknown token in paraphrase. Ones that contains *[UNK]* should have both grammar and coherent score lower than 5. The grammar score should depend on other tokens in the paraphrase. The specific coherent score depends on the impact of *[UNK]* on that certain paraphrase. Here are some paraphrase examples given original question *how can robot have human intelligence ?*:

- *paraphrase:* how can [UNK] be intelligent ?
  coherent score: 1
  This token prevent us from understanding the question and give proper answer. It causes serious information loss here;

- *paraphrase:* how can robot [UNK] intelligent ?
  coherent score: 3
  There is information loss, but the unknown token does not influence our understanding so much;

- *paraphrase:* how can robot be intelligent [UNK] ?
  coherent score: 4
  [UNK] basically does not influence understanding.

NOTED:

- Please decouple *grammar* and *coherent* as possible as you can. For instance, given a sentence *is it true that girls like shopping*, the paraphrase *do girls like go go shopping* can get a *coherent* score of 5 but a *grammar* score of only 3. But for the one you even can not understand, e.g., *how is the go shopping of girls*, you should give both of low *grammar* score and low *coherent* score, even it contains some topic-relevant words.

- Do a Google search when you see any strange entity name such that you can make more appropriate judgement.

Operation-guided Neural Networks for High Fidelity Data-To-Text Generation

Feng Nie[1]*, **Jinpeng Wang**[2], **Jin-Ge Yao**[2], **Rong Pan**[1], **Chin-Yew Lin**[2]
[1]Sun Yat-Sen University [2]Microsoft Research Asia
fengniesysu@gmail.com, {jinpwa, jinge.yao, cyl}@microsoft.com, panr@sysu.edu.cn

Abstract

Recent neural models for data-to-text generation are mostly based on data-driven end-to-end training over encoder-decoder networks. Even though the generated texts are mostly fluent and informative, they often generate descriptions that are not consistent with the input structured data. This is a critical issue especially in domains that require inference or calculations over raw data. In this paper, we attempt to improve the fidelity of neural data-to-text generation by utilizing pre-executed symbolic operations. We propose a framework called **Op**eration-guided **Att**ention-based sequence-to-sequence network (OpAtt), with a specifically designed gating mechanism as well as a quantization module for operation results to utilize information from pre-executed operations. Experiments on two sports datasets show our proposed method clearly improves the fidelity of the generated texts to the input structured data.

1 Introduction

Data-to-text generation is a classic language generation task that takes structured data (e.g., a table of statistics or a set of event records) as input, aiming at automatically producing texts that informatively, correctly and fluently describe the data (Kukich, 1983; Reiter and Dale, 1997; Angeli et al., 2010; Konstas and Lapata, 2012; Perez-Beltrachini and Gardent, 2017). Traditionally, a data-to-text generation system should pay attention to the problem of content selection (i.e., *what to say*) and surface realization (i.e., *how to say*) (Reiter and Dale, 1997; Gatt and Krahmer, 2018). Modern neural generation systems avoid the distinction of these aspects by building over a standard encoder-decoder architecture (Sutskever et al., 2014) with the attention mechanism over input content (Bahdanau et al., 2015) and train the whole system in an end-to-end fashion. As a result, end-to-end neural text generation has drawn increasing attention from the natural language research community (Mei et al., 2016; Lebret et al., 2016; Wiseman et al., 2017; Kiddon et al., 2016).

Input Data				
Row	Team	Points	Rebound	City
1	Heat	94	44	Miami
2	Hawks	95	40	Atlanta
Generated Description				
Hawks edges the Heat with 95 - 94				

Table 1: An example of generated texts from structured data. In this example, the wining team is not indicated explicitly, but can be inferred from the scores for hte two teams. The words with underlining and wave lines are based on the facts from the input data and the results of inferring, respectively.

However, a critical issue for neural text generation has been largely overlooked. In domains such as sports, finance or medical care, language generation should adhere to facts which are supported by or can be derived from the input data through analysis or inference. For instance, the sentence "Hawks edges the Heat with 95-94" describing the result of a basketball game should always conform to the original data in team names and the scoreline. More importantly, the word "edges" in the description is an inferred fact that the scores between the two competing teams are rather close, while "Hawks" is the winner that scores the slightly higher point total of "95". Since current neural models do not have special treatment for such data analytics, they are likely to generate spurious and incorrect statements. This problem has already been pointed out in recent studies (Wiseman et al., 2017). Related studies on neural program induction have shown that cur-

---

[*] Contribution during internship at Microsoft.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3879–3889
Brussels, Belgium, October 31 - November 4, 2018.

rent neural models have difficulties in learning arithmetic operations such as addition and comparisons (Joulin and Mikolov, 2015; Neelakantan et al., 2016).

A straightforward way to improve the fidelity of neural text generation is to separate symbolic operations out from the neural models. More specifically, it is viable to pre-execute a few symbolic operations before generation, and then use the results of execution to guide the whole generation process. However, there are two major challenges for incorporating pre-defined operations: (1) if we apply operations exhaustively on all fields with compatible value types in the table, it would create a huge search space in which mention worthy results are rare events and (2) it is difficult to establish the correspondences between specific spans of numeric results and lexical choices. For example, the word "edges" corresponds to the slight difference in score, i.e. 1, in Table. 1.

Inspired by recent work that separates neural representations and symbolic operations (Liang et al., 2017), we propose a framework for neural data-to-text generation that is able to utilize information from pre-computed operations on raw data. Based on a standard sequence-to-sequence model with an attention and copying mechanism, we design a gating mechanism for the neural model to decide which part of the execution results should be used for generation. To address the second challenge, we also design a quantization layer to map numerical execution results into bins to guide different lexical choices according to different quantities of values.

To examine the effectiveness of our proposed model, we collect a large dataset of sports headline generation for NBA basketball games[1]. We also evaluate the models on the ROTOWIRE dataset released by Wiseman et al. (2017) which targets at generating short paragraphs. Experiments show that our model outperforms current state-of-the-art neural methods in terms of both fluency and fidelity. In summary, we make the following contributions in this paper:

- We propose a neural data-to-text framework that generate texts by additional processing over input data. Based on a basic sequence-to-sequence model with attention and copying, we design a gating mechanism to enable the model to decide which part of the executed results should be utilized. We also propose a novel quantization layer to map specific numerical values onto different spans to affect lexical choices under different conditions.
- To focus our study on correct text generation, we collect a challenging dataset for NBA headline generation.
- We conduct experiments on the NBA headline dataset as well as the ROTOWIRE dataset from previous work. Results show improvements on both correctness and fluency from our proposed framework over baseline systems.

2 Background: Attention-Based Neural Sequence-to-Sequence Model

In this section, we briefly introduce the architecture of the attention-based sequence-to-sequence (Seq2Seq) (Cho et al., 2014b; Bahdanau et al., 2015) model with a copy mechanism (See et al., 2017), which is the basis of our proposed model.

2.1 RNN Encoder-Decoder

The goal of data-to-text generation is to generate a natural language description for a given set of data records $S = \{r_j\}_{j=1}^{K}$. Usually, a Seq2Seq model consists of an encoder and a decoder with recurrent neural networks (RNN). First, each input record r_j is encoded into a hidden vector $\mathbf{h}_j$ with $j \in \{1, ..., K\}$ using a bidirectional RNN. The decoder generates the description word by word using another RNN.

In the training phase, given a record set and its corresponding natural language description (S, y), the Seq2Seq model maximizes the conditional probability as follows:

$$P(y|S) = \prod_{t=1}^{T} P(y_t|y_{<t}, S) \qquad (1)$$

where y_t is the t-th word in the description and T is the length of the description. The conditional probability $P(y_t|y_{<t}, S)$ is computed as:

$$P(y_t|y_{<t}, S) = \mathrm{softmax}(f(\mathbf{d}_t, y_{t-1}, \mathbf{c}_t)) \qquad (2)$$

where $f(\cdot)$ is a non-linear function and $\mathbf{d}_t$ is the hidden state of the decoder at step t:

$$\mathbf{d}_t = g(\mathbf{d}_{t-1}, y_{t-1}, \mathbf{c}_{t-1}) \qquad (3)$$

---

[1] Available at `https://github.com/janenie/espn-nba-data`

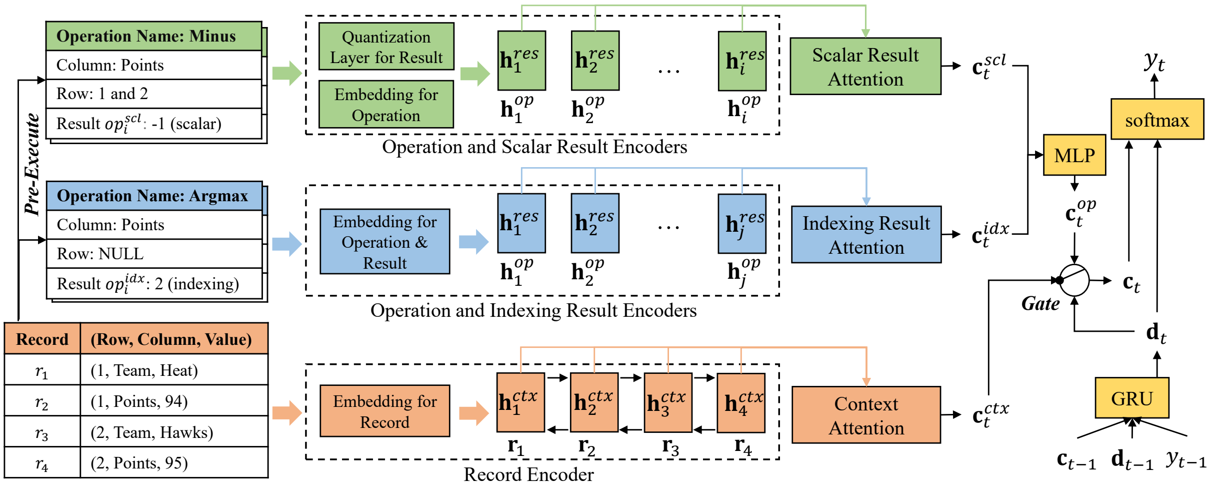

Figure 1: A diagram of the operation guided neural data-to-text generation. The input record table is converted from the first 3 columns of Table1. First, a set of operations are applied to the input records. Then, the records, operations and pre-excuted operation results are encoded. Finally, an attention-equipped GRU decoder with a gating mechanism decides which part of the execution results and context should be used for generation.

where $g(\cdot)$ is a non-linear function. We adopt the Gated Recurrent Unit (GRU) (Cho et al., 2014a) as the recurrent unit for the encoder and decoder. $\mathbf{c}_\mathrm{t}$ in Eq. 2 is the context vector at timestep t, computed as a weighted hidden vectors $\mathbf{h}_j$:

$$\mathbf{c}_t = \sum_{j=1}^{K} \alpha_{t,j} \mathbf{h}_j \tag{4}$$

where $\alpha_{t,j}$ is computed by an attention scheme, typically implemented as a softmax distribution over scores calculated with a multi-layer perceptron (Bahdanau et al., 2015).

2.2 Copy Mechanism

Recent work augments Seq2Seq models to copy words directly from the source information on which they are conditioned (Gu et al., 2016; See et al., 2017). These models usually introduce an additional binary variable z_t into per-timestep target word distribution, which indicates whether the target word y_t is copied from the source or is generated from the recurrent hidden states. We use the pointer-generator network (See et al., 2017) for the copy mechanism. Specifically, the binary variable z_t is calculated from the context vector $\mathbf{c}_t$, the decoder state $\mathbf{d}_t$ and the decoder input y_{t-1}:

$$p_{gen} = \sigma(\mathbf{w}_c^\top \mathbf{c}_t + \mathbf{w}_d^\top \mathbf{d}_t + \mathbf{w}_y^\top y_{t-1} + b_{ptr}) \tag{5}$$

where vectors $\mathbf{w}_c$, $\mathbf{w}_d$, $\mathbf{w}_y$ and the scalar b_{ptr} are learnable parameters, and σ is the sigmoid function. The joint probability for generating y_t is formulated as follows:

$$\begin{aligned} P_{copy}(y_t|y_{<t}, S) &= p_{gen} P(y_t|y_{<t}, S) \\ &+ (1 - p_{gen}) \sum_{i:r_i = y_t} \alpha_{t,i} \end{aligned} \tag{6}$$

3 The Proposed Model

In this paper, we propose to utilize information from pre-executed operations on the input data to guide the generation. As shown in Fig. 1, our model consists of a record encoder, an operation encoder and an operation result encoder, and an attention-equipped GRU decoder with a gating mechanism. First, a set of operations are applied to all valid records in the input data, yielding their corresponding pre-executed results. The pre-executed results act as facts inferred from input data to guide the generation. Then, the records, operation and pre-executed operation results are encoded into corresponding representation. Finally, we design a gating mechanism for the GRU decoder to decide which part of the inferred facts should be used for generation. Moreover, to address the challenge in establishing correspondences between specific numeric results and lexical choices, a quantization layer maps the results into several segmentations to guide the lexical choices.

3.1 Notation

Given the input data and description pair (S, y), where each target description $y = y_1, ..., y_T$ consists of T words, and each input data is stored in a table (e.g., Table 1), where each row is an entity and each column is a field of this entity. The input data can be transferred into K records $S = \{r_i\}_{i=1}^{K}$, where each record r is a triple $(r.idx, r.f, r.v)$. For r_4 in the table of Fig. 1, $r.idx$ $r.f$ and $r.v$ refer to the row index (e.g., row 2), the field name (e.g., column Points) and value (e.g., cell value 95), respectively. We also define a set of operations $\{op_i\}$, and the operations are applied to the input records S to produce corresponding results at the preprocessing stage. The results of operations can be categorized into two types: op_i^{scl} denotes results with a type of scalar value and op_i^{idx} denotes results with a type of indexing value.

3.2 Encoding Records

We map each record $r \in S$ into a vector $\mathbf{r}$ by concatenating the embedding of $r.idx$ (e.g., row 2), $r.f$ (e.g., column Points) and $r.v$ (e.g., cell value 95), denoted as $\mathbf{r} = [\mathbf{e}^{idx}, \mathbf{e}^{f}, \mathbf{e}^{v}]^\top$, where $\mathbf{e}^{idx}$, $\mathbf{e}^{f}$, $\mathbf{e}^{v}$ are trainable word embeddings of $r.idx$, $r.f$ and $r.v$ respectively, similar to (Yang et al., 2017). We feed a set of record vectors $\mathbf{r}_1, ..., \mathbf{r}_K$ to a bidirectional GRU and yield the final record representations $\mathbf{h}_1^{ctx}, ..., \mathbf{h}_K^{ctx}$ as introduced in Section 2. We leave the exploring of different encoding methods as future work, as it would affect the performance.

3.3 Encoding Operations

As shown in Fig. 1, each operation op_i consists of: a) the name of the operation $op_i.t$ (e.g., minus); b) the column $op_i.c$ to which the operation applies (e.g., Points); and c) the row to which the operation applies, denoted as $op_i.arg = \{r_i.idx\}_{i=1}^{A}$, where A is the count of arguments. We then encode each operation op_i by concatenating the representation of these three components and feed them into a nonlinear layer to represent each operation as follows:

$$\mathbf{h}_i^{op} = \tanh(\mathbf{W}_{op}[\mathbf{op}_i^{t}, \mathbf{op}_i^{c}, \mathbf{op}_i^{arg}]^\top{}_i + b_{op}), \quad (7)$$

where $\mathbf{op}_i^{t}$ is the embedding of $op_i.t$; $\mathbf{op}_i^{c}$ is the embedding of column $op_i.c$ which shares the same parameters of embedding with record column $r.f$. For $op_i.arg$, it may contain multiple arguments, so we apply a nonlinear layer to get a fixed length representation as follows:

$$\mathbf{op}_i^{arg} = \tanh(\sum_{k \in arg_i} \mathbf{W}_k^{arg} \mathbf{e}_k^{idx} + b_{arg}), \quad (8)$$

where $\mathbf{e}_k^{idx}$ is the same embedding as used to encode the row index $r.idx$, and $\mathbf{W}_k^{arg}$ and b_{arg} are learnable parameters. For operations which are applied in the entire column (e.g., argmax) without specific rows, the representation of arguments is a special vector which stands for ALL.

3.4 Encoding Operation Results

The operations produce two types of results, one is scalar results (e.g., the minus operation returns -1), the other is indexing results (e.g., the argmax operation returns the row number 2), and two encoders are designed to encode these results respectively.

Scalar Results Representation In Table. 1, the word "edges" is generated based on the fact that the points gap of the two teams is -1. In fact, other value likes -2 or -3 is close to -1, and the word "edges" is also applicable to them. However, directly establishing the lexical choices on various sparse numeric values is not easy (Reiter et al., 2005; Smiley et al., 2016; Zarrieß and Schlangen, 2016). Reiter et al. (2005) use consistent data-to-word rules for time-series weather forecast summary generation. In this paper, we aim to capture the data-to-word mapping automatically by a simple quantization unit. A *quantization layer* is designed to map the scalar values into several bins, namely quantization units. Specifically, we feed each scalar value op_i^{scl} to a softmax layer, and its representation $\mathbf{h}_i^{res}$ is computed as the weighted sum of all quantization embeddings:

$$\mathbf{q}_i = \mathbf{W}_q op_i^{scl} + b_q, \quad (9)$$

$$\mu_{i,l} = \frac{\exp(q_{i,l})}{\sum_{j=1}^{L} \exp(q_{i,j})}, \quad (10)$$

$$\mathbf{h}_i^{res} = \sum_{l=1}^{L} \mu_{i,l}\, \mathbf{e}_l^{scl} \quad (11)$$

where $\mathbf{W}_q$ and b_q are trainable parameters, $\mathbf{e}^{scl}$ is the quantization embedding and L is the size of quantization units. Note that L is much smaller than the unique number of scalar results. We set L to 5 in this paper.

Indexing Results Representation Some operations produce the row number of records (denoted as idx_i) as a result. For instance, the argmax operation in Fig. 1 returns row 2. We then look up the row embedding of the selected record defined in Section 3.2 to represent the result. Defined as $\mathbf{h}_i^{res} = \mathbf{e}_i^{idx}$.

3.5 Decoder

Comparing with the Seq2Seq model described in Section 2 and our model, the main difference is in the context vector $\mathbf{c}_t$. Different from Eq. 4, our model has both records and operations as input. We design two attention layers to summarize information from both parts respectively, the overall context vector $\mathbf{c}_t$ is balanced by a dynamic gate λ_t.

$$\mathbf{c}_t = (1-\lambda_t)\mathbf{c}_t^{op} + \lambda_t \mathbf{c}_t^{ctx}, \quad (12)$$

$$\lambda_t = \sigma(\mathbf{W}_g \mathbf{d}_t + b_g), \quad (13)$$

where $\mathbf{c}_t^{op}$ and $\mathbf{c}_t^{ctx}$ are the context vector of operation results and records, respectively.

As there are two types of operation results which have quite different meanings, their context vectors are calculated separately and then put together by a nonlinear layer. The context vectors $\mathbf{c}_t^{scl}$ of operation results with scalar value at timestep t are constructed as (Luong et al., 2015):

$$\mathbf{c}_t^{scl} = \sum_{j=1}^{N} \alpha_{t,j}^{scl} * \mathbf{h}_j^{res} \quad (14)$$

$$\beta_{t,j}^{scl} = \text{MLP}(\mathbf{d}_{t-1}, \mathbf{h}_j^{op}), \quad (15)$$

$$\alpha_{t,j}^{scl} = \frac{\exp(\beta_{t,j}^{scl})}{\sum_k \exp(\beta_{t,k}^{scl})} \quad (16)$$

where MLP stands for standard 1-layer perceptron (with $\tanh$ nonlinearity), and $\alpha_{t,j}^{scl}$ refers to the importance of j-th operations at the current timestep t. Eq. 14 is based on the attention mechanism which can be treated as mapping a query and a set of key-value pairs to an output. The output $\mathbf{c}_t^{scl}$ is computed as a weighted sum of the values $\mathbf{h}_j^{res}$, where the weight assigned to each value is computed by a compatibility function of the query $\mathbf{d}_{t-1}$ with the corresponding key $\mathbf{h}_j^{op}$. In this way, we also construct $\mathbf{c}_t^{idx}$. Then the context vector of operation results at time step t is computed by putting these two context vectors together:

$$\mathbf{c}_t^{op} = \text{MLP}([\mathbf{c}_t^{scl}, \mathbf{c}_t^{idx}]^\top) \quad (17)$$

The context vector representation $\mathbf{c}_t^{ctx}$ for records is constructed by replacing $\mathbf{h}_j^{res}$ with $\mathbf{h}_j^{ctx}$ in Eq. 14 and replacing $\mathbf{h}_j^{op}$ with $\mathbf{h}_j^{ctx}$ in Eq. 15.

After obtaining $\mathbf{c}_t$, the word distribution for generation can be calculated by substituting the $\mathbf{c}_t$ in Eq. 2. For the copy probability defined in Eq. 6, to copy words based on the information of both operations and records at current time step t, we need to update the attention weights for Eq. 6 based on the newly computed context vector $\mathbf{c}_t$ and decoding state $\mathbf{d}_t$:

$$\beta_{t,j}^{new} = \text{MLP}(\mathbf{h}_j^{ctx}, [\mathbf{d}_{t-1}, \mathbf{c}_t]^\top) \quad (18)$$

$$\alpha_{t,j}^{new} = \frac{\exp(\beta_{t,j}^{new})}{\sum_k \exp(\beta_{t,k}^{new})} \quad (19)$$

	ESPN	ROTOWIRE	WIKIBIO
Vocab	3.3K	11.3K	400K
Tokens	114.3K	1.6M	19M
Examples	15.1K	4.9K	728K
Avg Len	9.5	337.1	26.1
Input facts	62.7%	61.2%	72.1%
Inferred facts	29.1%	11.7%	7.4%
Unsupported	8.2%	27.1%	20.5%

Table 2: Dataset statistics. For each dataset, we also manually label the source for the facts mentioned in 20 descriptions, and report the percentage of facts based on the input data, inferred facts and unsupported facts.

3.6 Training

As the results of operations are pre-computed in an offline stage, our proposed model is fully differentiable and can be optimized in an end-to-end manner using back propagation. Given the batches of records $\{S\}_N$ and the standard natural language descriptions $\{Y\}_N$, the objective function is to minimize the negative log-likelihood:

$$L = -\frac{1}{N}\sum_{k=1}^{N}\sum_{t=1}^{T_k} \log p(y_t^k | y_{<t}^k, S^k) \quad (20)$$

where the superscript k indicates the index of the records-description pair, and T_k is the length of the k-th description.

4 Experiments

4.1 Datasets

Several benchmark datasets have been used in recent years for data-to-text generation (Liang et al., 2009; Chen and Mooney, 2008; Lebret et al., 2016). For instance, Lebret et al. (2016) have built a biography generation dataset from Wikipedia. However, a recent study by Perez-Beltrachini and Gardent (2017) shows that existing datasets have a few missing properties such as lacking syntactic and semantic diversity. To check whether the facts mentioned in the descriptions are based on input data, we identify the text spans which contain facts (e.g., in table 1, "Hawks" is a span contain fact) from the descriptions and divide each span into three categories: a) input facts (facts that can be directly found from the input), b) inferred facts (facts that can not be directly found from the input but can be derived), c) unsupported facts (facts that can not be found or derived from input data). Table 2 shows that WikiBio dataset requires inference on only 5.4% of its data. To better demonstrate the effectiveness of our approach, we adopt the following datasets which require substantially more inference based on the input data:

ROTOWIRE We use the dataset and its standard splits released by Wiseman et al. (2017), which consists of 4,853 human written NBA basketball game summaries aligned with their corresponding game statistics. Table 2 shows that 11.7% of facts in the game summaries can be inferred based on the input data. However, this dataset focuses on generating long text and 27.1% of facts are unsupported[2], which brings difficulties to the analysis of fidelity for the generated text.

ESPN We collect 15,054 NBA game result headlines during 2006-2017 from the ESPN website, paired with their corresponding game statistics. These headlines are professional and concise, e.g., the description in Fig. 1. The percentage of inferred facts is 29.1% while unsupportive facts is only 8%, so we can focus on generation for the inferred facts. We split the dataset into 12,043 (80%) for training, 1,505 (10%) for development and 1,506 (10%) for testing respectively.

4.2 Instantiation

In the following experiments, we define two operations, the minus operation which returns the scalar result and the argmax operation which returns a id of a row. These operations are applied to all columns and rows whose record values are numeric numbers. The number of pre-executed results increases with the number of operations, arguments and the size of input data, which will impact the efficiency of our model. The unnecessary operation arguments can be pruned, e.g., only apply operations to the arguments co-mentioned in descriptions on the training set. We will leave this part of research for our future work.

4.3 Experiment Setup

In the main experiments, we compare our model with the following methods: (a) Template: a problem-specific template-based generator which fills structured data into corresponding placeholders to generate texts[3], (b) Seq2Seq+copy: Seq2Seq model with pointer network copy mechanism introduced in Section 2. It is one of the state-of-the-art methods, (c) Seq2Seq+op: Seq2Seq+copy plus the results of operations, where results are directly treated as extra records and fed to the record encoder introduced in Section 3.2 with the original input together, (d) Seq2Seq+op+quanti: We apply the quantization layer Eq. 9-11 to the results of minus operation on the basis of Seq2Seq+op. For completeness, we also report the results of Wiseman et al. (2017) on the ROTOWIRE dataset. The difference between this baseline and Seq2Seq+copy is that the former uses an LSTM rather than GRU for decoding and an additional copying loss. All the experiments use a beam size of 5 in decoding[4].

For model training, we use the stochastic gradient descent algorithm and the AdaDelta optimizer (Zeiler, 2012). The dimension of trainable word embeddings are set to 256 except for the dimension of input record row embedding, which is set to 32; and the dimension of hidden units in GRUs are all set to 512. All the parameters are initialized using a normal distribution with zero mean and a variance of $\sqrt{6/(d_{in}+d_{out})}$, where d_{in} is the dimension of the input layer and d_{out} is the dimension of the output layer (Glorot and Bengio, 2010). Training converges after 40 epochs.

4.4 Main Results

We adopt both automatic evaluation and human evaluation to evaluate the proposed model. **Automatic Evaluation** We employ BLEU-4 as the metric for automatic evaluation. Table 4 gives the automatic evaluation results for generation on two datasets. Our proposed model OpAtt outperforms neural network baselines (See et al., 2017; Wiseman et al., 2017). The results show that our method which incorporates the operations enables generating texts that are fidelity to facts and therefore yields the best performance. Seq2Seq+op+quant outper-

---

[2] e.g., injuries, rankings in the league, team schedule, etc.

[3] For the ROTOWIRE dataset, we adopt Wiseman et al. (2017)'s templates. For the ESPN dataset, we use Dou et al. (2018)'s system to extract templates. The template is constructed by emitting teams and players information in a sentence: `<team1>` beats `<team2>` with `<point1>`-`<point2>`.

[4] The authors have updated the dataset to fix some mistakes recently, so we cannot use the result which is reported in their paper and rerun this baseline with the authors' code.

	ESPN			ROTOWIRE		
	#Cont./#Supp. (input facts)	#Cont./#Supp. (inferred facts)	#Cont. (unsupported)	#Cont./#Supp. (input facts)	#Cont./#Supp. (inferred facts)	#Cont. (unsupported)
Ref	0.00 / 4.90	0.00 / 1.12	0.51	0.00 / 12.87	0.00 / 3.07	3.20
Seq2Seq+copy	0.44 / 4.61	0.16 / 1.25	0.25	3.75 / 14.44	0.89 / 2.20	2.82
Seq2Seq+op	0.24 / 3.97	0.07 / 1.08	0.76	5.55 / 18.13	0.42 / 2.53	1.93
Seq2Seq+op+quant	0.21 / 4.88	0.03 / 1.10	0.32	3.47 / 16.02	0.53 / 2.02	2.13
OpAtt	0.04 / 5.00	0.02 / 1.27	0.19	2.24 / 16.56	0.18 / 2.84	2.07

Table 3: Average annotators judgment for the count of facts contradicting (#Cont.) and supporting (#Supp.) on facts based on input data, inferred facts and unsupported facts respectively.

	ESPN		ROTOWIRE	
	Dev	Test	Dev	Test
Template	13.75	14.27	8.97	8.93
Wiseman's	-	-	13.57	13.62
Seq2Seq+copy	15.63	15.30	13.72	13.47
Seq2Seq+op	14.07	13.74	13.52	13.44
Seq2Seq+op+quant	15.68	15.49	14.05	13.88
OpAtt	**17.19***	**18.00***	**14.96***	**14.74***

Table 4: BLEU scores (%) over two datasets. Statistical significant is indicated with *($p < 0.05$) with respect to Seq2Seq+copy.

forms the baseline method Seq2Seq+copy, but is not as good as our method. The result confirms that our proposed method with specialized operation encoder and gating mechanism utilizes the information of operations more effectively. Moreover, Seq2Seq+op+quant outperforms Seq2Seq+op showing the effectiveness of the quantization layer.

Human Evaluation Because of the approximate nature of the automated metric BLEU, we also conduct human evaluation to examine the fidelity of the generated texts. We randomly select some games from testing set, and entrust a professional crowdsourcing company to annotate the generated texts[5]. Specifically, three native English workers who are familiar with NBA games are hired. They are first required to identify the text spans which contain facts from the generated texts, then categorize the text spans into one of three facts listed in Table 2, and finally judge whether the span is supported or contradicted by the input data.

Table 3 shows the annotation results. Our method talks more about the inferred facts in the generated texts while includes less contradictions. In addition, all methods produce some unsupported facts which affect the fidelity of the generated texts. We leave this issue for future work.

	Dev	Test
Seq2Seq + copy	15.63	15.30
OpAtt	**17.19**	**18.00**
OpAtt w/o argmax op	15.71	15.97
OpAtt w/o quantization	16.35	16.70
OpAtt w/o gate	16.35	16.15

Table 5: BLEU scores (%) of model ablation.

Reference	horford 's dunk helps hawks edge nets , 114 - 111
Seq2Seq +copy	nets **rally** from 17 down to **top** nets 111 - 111
OpAtt w/o argmax op	hawks **rally** from 17 down to beat nets 114 - 111
OpAtt	horford scores 24 as hawks beat nets 114 - 111

Table 6: The generated texts by introducing different operations. The words with underline, wavy line and dot line are input facts, inferred facts and unsupported fact, respectively. And the **bold** words are contradicted facts.

4.5 Analysis

As discussed in Section 4.1, the ESPN dataset is rich in inferred facts. Therefore, the model analysis is based on this dataset, and all case studies are made on the development set.

4.5.1 Effect of Operations

We examine the necessity and the benefit of introducing operations by removing the argmax operation (see "OpAtt w/o argmax op" in Table 5). Comparing to Seq2Seq+copy, the results show that our full model and "OpAtt w/o argmax op" which incorporates results of operations both work well in terms of BLEU, and the improvements increase with the number of operations.

To better illustrate that our model can generate factually correct text, we show the texts generated by different models in Table 6. The game

---

[5] The Fleiss' kappa score of the annotation is 0.782 for ESPN and 0.761 for ROTOWIRE respectively. For the ESPN dataset, we select 50 games and each with one generated sentence. For ROTOWIRE, by following (Wiseman et al., 2017), we select 15 games and each with 3 randomly selected sentences.

results mentioned in the text generated by the Seq2Seq+copy model are wrong, which shows the inability for existing neural models on inferring facts from the structured data. After adding the minus operation, “OpAtt w/o argmax op” is able to infer the game result by applying the minus operation on the points of the two competing teams, therefore its generated text conforms to the game results. The results confirm the necessity of introducing operations to ensure factually correct generation. Furthermore, our full model generates text with the correct point leader and game result based on the results of operation argmax and operation minus respectively.

4.5.2 Effect of Quantization

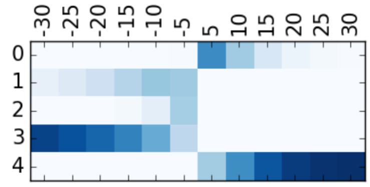

Figure 2: Weights of the quantization softmax layer when mapping the points gap of two competing teams to five bins. X-axis is points gap and Y-axis is quantization bin.

The quantization layer maps the numerical execution results into several bins to enable different lexical choices according to different quantities of values. Compared to our full model, “OpAtt w/o quantization” in Table 5 which removes the quantization layer decreases the BLEU performance, which shows the effectiveness of the quantization layer in the lexical choices during generation.

In Fig. 2, we visualize the weights of quantization softmax layer $\mu_{i,l}$ produced by Eq. 10 when mapping the points gap of two competing teams to five bins. We can see that the points gaps with close numerical values are mapped to the same bin, so the decoder can choose similar words for them in generation. When the absolute value of the points gap is small, the weights distribution over the points gap is dispersive. At this time, the decoder tends to generate general words. This distribution becomes more centralized with the increase of the absolute value of the points gap, to generate more unique words. Moreover, we show the distribution of words that describes the winning relationship of games over different intervals of game points gap. As shown in Table 7, we can clearly see that apart from three most common word “beat”, “past”, “win over”, our proposed quantization layer can choose specific words according to the points gap. The word “edge” and “hold off” will only be chosen when the points gap is small, while the word “rout” and “blow out” will appear when the points gap is larger than 10.

Points Gap	Words describes winning relationship
[0, 5)	beat, past, win over, edge, hold off, survive
[5, 10)	beat, past, win over, out last, hold off
[10, 20)	beat, past, win over, blow out, top, pull away, rout
>= 20	beat, past, win over, power, rout, easy win over, roll past

Table 7: The words that describing the winning relationship of games over different intervals of game points gap.

4.5.3 Effect of Gating Mechanism

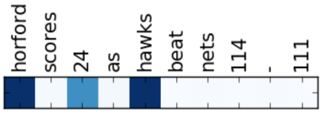

Figure 3: The gating weights at different time steps.

We design a gating mechanism to decide when to incorporate the results of operations to guide the process of generation. From Table 5, “OpAtt w/o gate” stands for the method which replaces the balancing weight λ in Eq. 12 to 0.5, which is a special case of our proposed gating mechanism. The performance of this ablation is worse than our full model, which demonstrates that the gating mechanism is an essential component. Fig. 3 shows an example of the gating weights at each time step in generation, where a darker cell means the incorporation of more information from operation results for decoding corresponding word. We can see that the gate weights are reasonable, as the gate values are large when deciding the team leader “horford” and the winner of the game “hawks”.

5 Related Work

Data-to-text generation is a task of natural language generation (NLG) (Gatt and Krahmer, 2018). Previous research has focused on individual content selection (Kukich, 1983; Reiter and Dale, 1997; Duboué and McKeown, 2003; Barzilay and Lapata, 2005) and surface realization (Goldberg et al., 1994; Soricut and Marcu, 2006; Wong and Mooney, 2007).

Recent work avoids the distinction of the content selection and sentence realization. Chen and Mooney (2008) use an SMT based approach to learn alignments between comments and their corresponding event records. Angeli et al. (2010) transform the problem into a sequence of local decisions using a log-linear model. Konstas and Lapata (2012) employ a PCFG to simultaneously optimize the content selection and surface realization problem.

In the field of neural text generation, Mei et al. (2016) uses a neural encoder-decoder approach for end-to-end training. Some have focused on conditional language generation based on tables (Yang et al., 2017), short biographies generation from Wikipedia tables (Lebret et al., 2016; Chisholm et al., 2017) and comments generation based on stock prices (Murakami et al., 2017). However, none of these methods consider incorporating the facts that can be inferred from the input data to guide the process of generation. Murakami et al. (2017) post-process the price by extending the copy mechanism and replacing numerical values with defined arithmetic operations after generation. While our model, OpAtt utilizes information from pre-computed operations on raw data to guide the generation.

Our work is related to research areas on deep learning models for program induction and question answering from a knowledge base (Neelakantan et al., 2016; Liang et al., 2017; Ling et al., 2017). Neelakantan et al. (2016) solve the problem of semantic parsing from structured data and generate programs using pre-defined arithmetic operations. Liang et al. (2017) design a set of executable operators and obtain the answers by the generated logic forms. Ling et al. (2017) design a set of operators to generate the latent program for math problem solving. However, data-to-text is a different task. The operations for these methods are designed to find the answers, while we use the operations to guide the process of generation.

6 Conclusion and Future Work

In this work, we address the problem of generating consistent text from structured data in a neural data-to-text generation framework, where we extract facts that can be inferred in the given data by applying several executable symbolic operations to guide the generation. Moreover, we design a special quantization layer to operations whose result type is numeric value and establish the correspondence between the numeric values and lexical choice in generation. Experiments show that our method, OpAtt, outperforms existing state-of-the-art neural methods, in both fluency and fidelity evaluations.

As applying operations on a large number of records greatly increases the search space for the attention mechanism, we will extend our model to automatically detect the relevant operations to reduce computing complexity. We will also extend the set of operations to accommodate historical data, graph data and detect the unsupported facts in the generation within the single framework.

7 Acknowledgement

We thank the anonymous reviewers for their helpful comments. We also thank Zhirui Zhang, Shuangzhi Wu and Yu Wu for helpful conversations and comments on the work. The contact author of this paper, according to the meaning given to this role by Sun Yat-Sen University, is Rong Pan.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 502–512.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 141–148.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 128–135.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. *CoRR*, abs/1702.06235.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Longxu Dou, Guanghui Qin, Jinpeng Wang, Jin-Ge Yao, and Chin-Yew Lin. 2018. Data2text studio: Automated text generation from structured data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, Brussels, Belgium, October 11- November 4, 2018*.

Pablo Ariel Duboué and Kathleen R. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2003, Sapporo, Japan, July 11-12, 2003*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Eli Goldberg, Norbert Driedger, and Richard I. Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Armand Joulin and Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 190–198.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 329–339.

Ioannis Konstas and Mirella Lapata. 2012. Concept-to-text generation via discriminative reranking. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 369–378.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, June 15-17, 1983.*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1203–1213.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada. Association for Computational Linguistics.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 91–99.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 720–730.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1374–1384.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. *ICLR*.

Laura Perez-Beltrachini and Claire Gardent. 2017. Analysing data-to-text generation benchmarks. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 238–242.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artif. Intell.*, 167(1-2):137–169.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Charese Smiley, Vassilis Plachouras, Frank Schilder, Hiroko Bretz, Jochen Leidner, and Dezhao Song. 2016. When to plummet and when to soar: Corpus based verb selection for natural language generation. In *Proceedings of the 9th International Natural Language Generation conference*.

Radu Soricut and Daniel Marcu. 2006. Stochastic language generation using widl-expressions and its application in machine translation and summarization. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263.

Yuk Wah Wong and Raymond J. Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, April 22-27, 2007, Rochester, New York, USA*, pages 172–179.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1850–1859.

Sina Zarrieß and David Schlangen. 2016. Towards generating colour terms for referents in photographs: Prefer the expected or the unexpected? In *Proceedings of the 9th International Natural Language Generation conference*.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Generating Classical Chinese Poems via Conditional Variational Autoencoder and Adversarial Training

Juntao Li[1,2,†], **Yan Song**[3], **Haisong Zhang**[3],
Dongmin Chen[1], **Shuming Shi**[3], **Dongyan Zhao**[1,2], **Rui Yan**[1,2,*]
[1]Beijing Institute of Big Data Research,
Academy for Advanced Interdisciplinary Studies, Peking University, Beijing, China
[2]Institute of Computer Science and Technology, Peking University, Beijing, China
[3]Tencent AI Lab
{lijuntao,zhaody,dongminchen,ruiyan}@pku.edu.cn
{clksong,hansonzhang,shumingshi}@tencent.com

Abstract

It is a challenging task to automatically compose poems with not only fluent expressions but also aesthetic wording. Although much attention has been paid to this task and promising progress is made, there exist notable gaps between automatically generated ones with those created by humans, especially on the aspects of term novelty and thematic consistency. Towards filling the gap, in this paper, we propose a conditional variational autoencoder with adversarial training for classical Chinese poem generation, where the autoencoder part generates poems with novel terms and a discriminator is applied to adversarially learn their thematic consistency with their titles. Experimental results on a large poetry corpus confirm the validity and effectiveness of our model, where its automatic and human evaluation scores outperform existing models.

1 Introduction

In mastering concise, elegant wordings with aesthetic rhythms in fixed patterns, classical Chinese poem is a special cultural heritage to record personal emotions and political views, as well as document daily or historical events. Being a fascinating art, writing poems is an attractive task that researchers of artificial intelligence are interested in (Tosa et al., 2008; Wu et al., 2009; Netzer et al., 2009; Oliveira, 2012; Yan et al., 2013, 2016a; Ghazvininejad et al., 2016, 2017; Singh et al., 2017; Xu et al., 2018), partially for the reason that poem generation and its related research could benefit other constrained natural language generation tasks. Conventionally, rule-based models (Zhou et al., 2010) and statistical machine translation (SMT) models (He et al., 2012) are proposed for this task. Recently, deep neural models are employed to generate fluent and natural poems (Wang et al., 2016a; Yan, 2016; Zhang et al., 2017a). Although these models look promising, they are limited in many aspects, e.g., previous studies generally fail to keep thematic consistency (Wang et al., 2016c; Yang et al., 2017) and improve term[1] novelty (Zhang et al., 2017a), which are important characteristics of poems.

In classical Chinese poem composing, thematic consistency and term novelty are usually mutually exclusive conditions to each other, i.e., consistent lines may bring duplicated terms while intriguing choices of characters could result in thematic diversities. On one hand, thematic consistency is essential for poems; it is preferred that all lines concentrate on the same theme throughout a poem. Previous work mainly focused on using keywords (Wang et al., 2016c; Hopkins and Kiela, 2017) to plan a poem so as to generate each line with a specific keyword. Such strategy is risky for the reason that the keywords are not guaranteed consistent in a topic, especially when they are generated or extracted from an inventory (Wang et al., 2016c). On the other hand, Chinese poems are generally short in length, with every character carefully chosen to be concise and elegant. Yet, prior poem generation models with recurrent neural networks (RNN) are likely to generate high-frequency characters (Zhang et al., 2017a), and the resulted poems are trivial and boring. The reason is that RNN tends to be entrapped within local word co-occurrences, they normally fail to capture global characteristic such as topic or hierarchical semantic properties (Bowman et al., 2016).

To address the aforementioned shortcomings, RNN is extended to autoencoder (Dai and Le, 2015) for improving sequence learning, which has

---

*Corresponding author: Rui Yan (ruiyan@pku.edu.cn)
†Work was partially done at Tencent AI Lab.

[1] We use term and character interchangeably in this paper.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3890–3900
Brussels, Belgium, October 31 - November 4, 2018.

been proven to be appealing in explicitly modeling global properties such as syntactic, semantic, and discourse coherence (Li et al., 2015). Moreover, boosting autoencoder with variational inference (Kingma and Welling, 2014), known as variational autoencoder (VAE), can generate not only consistent but also novel and fluent term sequences (Bowman et al., 2016). To generalize VAE for versatile scenarios, conditional variational autoencoders (CVAE) are proposed to supervise a generation process with certain attributes while maintaining the advantages of VAE. It is verified in supervised dialogue generation (Serban et al., 2017; Shen et al., 2017; Zhao et al., 2017) that CVAE can generate better responses with given dialogue contexts. Given the above background and to align it with our expectations for poem generation, it is worth trying to apply CVAE to create poems. In the meantime, consider that modeling thematic consistency with adversarial training is proven to be promising in controlled text generation (Hu et al., 2017), models for semantic matching can be potentially improved with an explicit discriminator (Wu et al., 2017), so does poem generation.

In this paper, we propose a novel poem generation model (CVAE-D) using CVAE to generate novel terms and a discriminator (D) to explicitly control thematic consistency with adversarial training. To the best of our knowledge, this is the first work of generating poems with the combination of CVAE and adversarial training. Experiments on a large classical Chinese poetry corpus confirm that, through encoding inputs with latent variables and explicit measurement of thematic information, the proposed model outperforms existing ones in various evaluations. Quantitative and qualitative analysis indicate that our model can generate poems with not only distinctive terms, but also consistent themes to their titles.

2 Preliminaries

2.1 VAE and CVAE

In general, VAE consists of an encoder and a decoder, which correspond to the encoding process where input x is mapped to a latent variable z, i.e., $x \mapsto z$, and the decoding process where the latent variable z is reconstructed to the input x, i.e., $z \mapsto x$. In detail, the encoding process computes a posterior distribution $q_\theta(z \mid x)$ given the input x. Similarly, the decoding process can be formulated as $p_\theta(x \mid z)$, representing the probability distribution of generating input x conditioned on z, where z has a regularized prior distribution $p_\theta(z)$, i.e. a standard Gaussian distribution. Herein θ represents the parameters of both encoder and decoder. Importantly, presented by Kingma and Welling (2014), on the condition of large datasets and intractable integral of the marginal likelihood $p_\theta(x)$, the true posterior $q_\theta(z \mid x)$ is simulated by a variational approximation $q_\phi(z \mid x)$ in modeling the encoding process, where ϕ is the parameters for q.

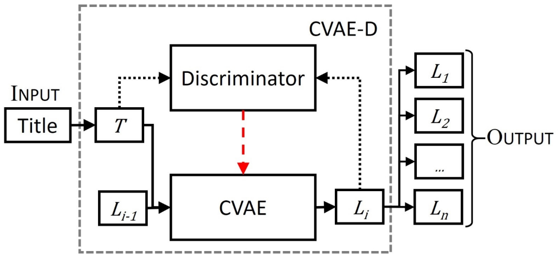

Figure 1: The overall framework of our poem generation model. Solid arrows present the generation process of each line L_i on the condition of the previous line L_{i-1} and title T. Black dotted arrows represent the adversarial learning for thematic consistency. The red dashed arrow refers to the back-propagation of the discriminator to the CVAE.

In learning a VAE, its objective is to maximize the log-likelihood $\log p_\theta(x)$ over input x. To facilitate learning, one can target on pushing up the variational lower bound of $\log p_\theta(x)$:

$$\begin{aligned}\mathbb{L}(\theta, \phi; x) = &- \mathrm{KL}(q_\phi(z|x) \parallel p_\theta(z)) \\ &+ \mathbf{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]\end{aligned} \tag{1}$$

such that the original $\log p_\theta(x)$ is also optimized. Herein the KL-divergence term $\mathrm{KL}(\cdot)$ can be viewed as the regularization for encouraging the approximated posterior $q_\phi(z|x)$ to be close to the prior $p_\theta(z)$, e.g. standard Gaussian distribution. $\mathrm{E}[\cdot]$ is the reconstruction loss conditioned on the approximation posterior $q_\phi(z|x)$, which reflects how well the decoding process goes.

CVAE extends VAE with an extra condition c to supervise the generation process by modifying the. The objective of CVAE is thus to maximize the reconstruction log-likelihood of the input x under the condition of c. Following the operation for VAE, we have the corresponding variational lower bound of $p_\theta(x|c)$ formulated as

$$\begin{aligned}\mathbb{L}(\theta, \phi; x, c) = &- \mathrm{KL}(q_\phi(z|x, c) \parallel p_\theta(z|c)) \\ &+ \mathbf{E}_{q_\phi(z|x,c)}[\log p_\theta(x|z, c)]\end{aligned} \tag{2}$$

which is similar to Eq.1 except that all items are introduced with c, such as $q_\phi(z|x, c)$ and $p_\theta(z|c)$, referring to the conditioned approximate posterior and the conditioned prior, respectively.

2.2 Problem Formulation

Following the text-to-text generation paradigm (Ranzato et al., 2015; Kiddon et al., 2016; Hu et al., 2017; Ghosh et al., 2017), our task has a similar problem setting with conventional studies (Zhang and Lapata, 2014; Wang et al., 2016c), where a poem is generated in a line-by-line manner that each line serves as the input for the next one, as illustrated in Figure 1. To formulate this task, we separate its input and output with necessary notations as follows.

The **INPUT** of the entire model is a title, $T = (e_1, e_2, \ldots, e_N)$, functionalized as the theme of the target poem[2], where e_i refers to i-the character's embedding and N is the length of the title. The first line L_1 is generated only conditioned on the title T, once this step is done, the model takes the input of the previous generated line as well as the title at each subsequent step, until the entire poem is completed.

The overall **OUTPUT** is an n-line poem, formulated as $(L_1, L_2, \ldots, L_n)$, where $L_i = (e_{i,1}, e_{i,2}, \ldots, e_{i,m})$ denotes each line in the poem, with $e_{i,j}$ referring to the embedding of a character at i-th line on j-th position, $\forall\ 1 \leq i \leq n$, $1 \leq j \leq m$. Particularly for classic Chinese poems, there are strict patterns, which require $m = 5$ or $m = 7$, and $n = 4$[3] or $n = 8$[4]. Once a template is chosen, m and n are fixed. In this paper, we mainly focus on $n = 4$.

3 The Model

As illustrated in Figure 1, our CVAE-D consists of two parts, CVAE and a discriminator, where their details are elaborated in the following subsections.

3.1 The CVAE

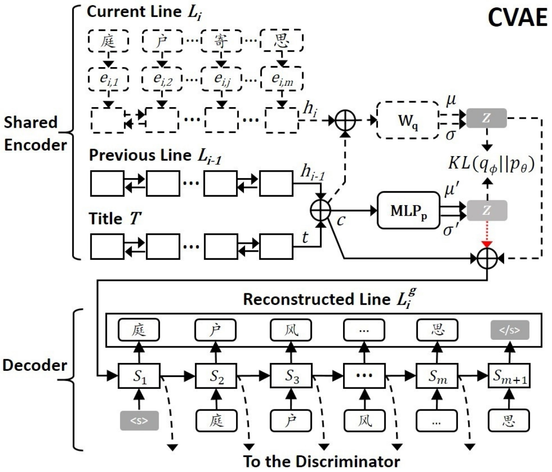

Figure 2: The CVAE for poem generation. $\oplus$ denotes the vector concatenation operation. Only the part with solid lines and the red dotted arrow is applied in prediction, while the entire CVAE is used in training process except the red dotted arrow part.

The CVAE includes an encoder and a decoder, plays as the core part in our model that generates classic Chinese poems. The encoder encodes both the title and lines with shared parameters by a bidirectional RNN (Schuster and Paliwal, 1997) with gated recurrent units (GRU) (Chung et al., 2014). Through the encoder, at each step, the previous line L_{i-1}, current line L_i and title T are represented as concatenated forward and backward[5] vectors $h_{i-1} = [\overrightarrow{h}_{i-1}, \overleftarrow{h}_{i-1}]$, $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$ and $t = [\overrightarrow{t}, \overleftarrow{t}]$, respectively. Note that h_i corresponds to x, while the concatenation of h_{i-1} and t functionalized as c in Eq. 2, i.e., $c = [h_{i-1}, t]$. Following previous work (Kingma and Welling, 2014; Zhao et al., 2017; Yang et al., 2017), we assume that the variational approximate posterior is a multivariate Gaussian $\mathcal{N}$ with a diagonal covariance structure $q_\phi(z|x, c) = \mathcal{N}(\mu, \sigma^2 \boldsymbol{I})$. Thus μ and σ are the key parameters[6] to be learned, and they are computed by

$$\begin{bmatrix} \mu \\ \log\left(\sigma^2\right) \end{bmatrix} = W_q \begin{bmatrix} x \\ c \end{bmatrix} + b_q \tag{3}$$

where W_q and b_q are trainable parameters. Similarly, the prior $p_\theta(z|c)$ can be formulated as another multivariate Gaussian $\mathcal{N}(\mu', \sigma'^2 \boldsymbol{I})$; its parameters are then calculated by a single-layer fully-connected neural network (denoted as MLP) with the $tanh(\cdot)$ activation function,

$$\begin{bmatrix} \mu' \\ \log\left(\sigma'^2\right) \end{bmatrix} = \mathrm{MLP}_p\left(c\right) \tag{4}$$

The decoder uses a one-layer RNN with GRU that takes $[z, c]$ as the input to predict each line L_i. The hidden states of the GRU, $(s_1, s_2, \ldots, s_m)$[7],

---

[2] We directly treat the title as the theme for each poem in this paper instead of transferring it to a few keywords as that was done in Yang et al. (2017).

[3] The quatrain.

[4] The eight-line regulated verse.

[5] $\rightarrow$ and $\leftarrow$ refer to forward and backward, respectively.

[6] μ and σ^2 represent the mean and variance of $\mathcal{N}(\mu, \sigma^2 \boldsymbol{I})$.

[7] Note that s_{m+1} is not passed to the discriminator.

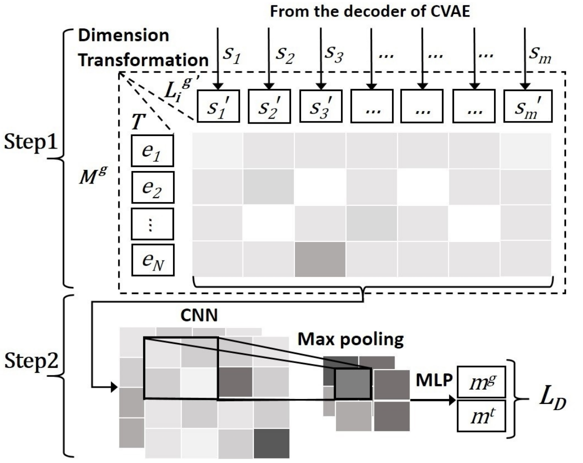

Figure 3: The Discriminator.

are not only used to generate the reconstructed lines, but also passed to the discriminator for learning thematic consistency.

The entire encoder and the decoder are used throughout the training process, with only part of the encoder (objects with solid lines in Figure 2) and the decoder applied in prediction. It is worth noting that, θ and ϕ mentioned in §2.1 are not explicitly corresponded to any particular neural networks described in this section. Instead, the probability process denoted by θ corresponds to the decoding and part of the encoding process, so does ϕ, i.e., $\phi = \{W_q, b_q\}$.

3.2 The Discriminator

The discriminator is introduced in our model to evaluate thematic consistency between the input title and the generated poem lines. The loss from this discriminator is then back-propagated to the decoder of the CVAE to enhance its training. In this paper, we employ a procedure that consists of two steps. First, we compute an interaction (or matching) matrix according to a generated line ${L_i}^g$ and the title T, where ${L_i}^g$ is the reconstructed result of L_i. Then, we utilize a convolutional neural network (CNN) to learn the matching score between ${L_i}^g$ and T, where the score is interpreted as the degree of thematic consistency. Specifically, in the discriminator, we treat ${L_i}^g$ and L_i as the negative and positive instance, referring to thematically inconsistent and consistent case, respectively.

In detail, for the first step, we use the state sequence[8] of the decoder to represent ${L_i}^g$, i.e., ${L_i}^g = (s_1, s_2, \ldots, s_m)$. A dimension transformation is then conducted on ${L_i}^g$, to align ${L_i}^g$ and T:

$$s_i^{'} = ReLU(W_d s_i + b_d) \tag{5}$$

where $ReLU$ is the rectified linear units activation function (Nair and Hinton, 2010), with trainable parameters W_d and b_d. In doing so, the dimension of $s_i^{'}$ is identical to character embeddings. The transformed line is then denoted as ${L_i}^{g'} = (s_1^{'}, s_2^{'}, \ldots, s_m^{'})$. Thus the interaction matrix between ${L_i}^{g'}$ and T is then formulated as

$$M^g = {L_i}^{g'} \cdot T \tag{6}$$

where $M^g \in \mathbb{R}^{N \times m}$; "$\cdot$" denotes the matrix multiplication.

In the second step, a CNN is used to extract features from the interaction matrix. The resulted feature matrix is calculated by $\mathcal{F} = CNN(M^g)$. Then, we apply a max-overtime pooling (Collobert et al., 2011) over $\mathcal{F}$ to capture the most salient information. After this operation, an MLP with one hidden layer is used to flatten the feature matrix and generate the final matching score $m^g \in (0, 1)$ via a $sigmoid$ activation function.

In addition to m^g, the matching score m^t between the positive sample L_i and T is computed in a process similar to the above procedure, except the dimension transformation because character embeddings in both title T and L_i share the same dimension.[9]

Finally, following the routine of generative adversarial networks (GAN) (Goodfellow et al., 2014), the discriminator is trained to measure the thematic consistency of generated lines and the ground truth lines according to the matching scores m^g and m^t, with the objective function

$$\mathcal{L}_D = \log(m^g) + \log(1 - m^t) \tag{7}$$

minimized. Note that the discriminator is only applied during the training process, where the parameters of the encoder and decoder are enhanced by the feedback of the discriminator.

---

[8] Following previous work (Goyal et al., 2016; Hu et al., 2017) using adversarial training, using state sequence instead of the outputs is because the discrete nature of the outputs hinders gradient calculation.

[9] Different from ${L_i}^g$, L_i is represented directly by its sequence of character embeddings, for the reason that the discriminator is only connected with the decoder while L_i does not go through it. Otherwise, if the encoder states are passed to the discriminator, the loss would be back-propagated to the encoder and disturb CVAE training accordingly.

	Poem #	Line #	Vocab #	Token #
PTD	56,549	371,754	7,685	2,093,740
PSD	253,237	1,497,348	9,959	9,008,418
Total	309,786	1,869,102	10,306	11,102,158

Table 1: Corpus statistics of PTD and PSD. **Vocab #** and **Token #** refer to vocabulary size and total number of tokens, respectively, in terms of character.

3.3 Training the Model

The overall objective of CVAE-D is to minimize

$$\mathcal{L}_{\text{CVAE-D}} = \mathcal{L}_{CVAE} - \lambda \mathcal{L}_D \tag{8}$$

with respect to parameters of the CVAE, where $\mathcal{L}_{CVAE}$ is the loss of CVAE, corresponding to $-\mathbb{L}(\theta, \phi; x, c)$. In doing so, $\mathcal{L}_D$ is maximized with regard to parameters of the discriminator, referring to that the generated poems are thematic consistent and able to confuse the discriminator. Herein λ is a balancing parameter. We train the CVAE and the discriminator alternatively in a two-step adversarial fashion similar to that was done in Zhang et al. (2017c). This training strategy is repeated until the $\mathcal{L}_{\text{CVAE-D}}$ is converged.

4 Experiment Setup

4.1 Datasets

To learn our poem generation model, we collect two corpora for experiments: a collection of classic Chinese poems from Tang dynasty (PTD), and the other from Song dynasty (PSD). Statistics of the two corpora are reported in Table 1. Note that for classical Chinese poem, the dominant genres are quatrain and eight-line regulated verse with either 5 or 7 characters in each line. As a result, our model is targeted to generate poems within these two genres, especially the quatrain. All titles of poems are treated as their themes. We randomly choose 1,000 and 2,000 poems for validation and test, respectively, with the rest poems for training.

4.2 Baselines

In addition to our CVAE-D, several highly related and strong methods are conducted as baselines in our experiments, including:

S2S, the conventional sequence-to-sequence model (Sutskever et al., 2014), which has proven to be successful in neural machine translation (NMT) and other text generation tasks.

AS2S and its extension **Key-AS2S** and **Mem-AS2S**, where AS2S is the S2S model integrated with attention mechanism (Bahdanau et al., 2014). Key-AS2S and Mem-AS2S are AS2S with keywords planning (Wang et al., 2016c) and a memory module (Zhang et al., 2017a), respectively. Particularly, they are dedicated models designed for Chinese poem generation.

Criterion	Description
Consistency	Whether a poem displays a consistent theme.
Fluency	Whether a poem is grammatically satisfied.
Meaning	How meaningful the content of a poem is.
Poeticness	Whether a poem has the attributes of poetry.
Overall	Average scores of the above four criteria.

Table 2: Human evaluation criteria.

GAN, a basic implementation of generative adversarial networks (Goodfellow et al., 2014) for this task on top of S2S. This baseline is added to investigate the performance of introducing a discriminator to simple structures other than CVAE.

CVAE[10] and its extension **CVAE-Key**, where the former is the conventional CVAE model and the latter refers to the combination of CVAE and keywords planning (Yang et al., 2017). The CVAE baseline is used for investigating how poem generation can be done with only CVAE, while CVAE-Key aims to provide a comparison to our model with a different technique for thematic control.

4.3 Model Settings

All baselines and the CVAE-D are trained with the following hyper-parameters. The dimension of character embedding is set to 300 for the most frequent 10,000 characters in our vocabulary. The hidden state sizes of the GRU encoder and decoder are set to 500. All trainable parameters, e.g., W_q and W_d, are initialized from a uniform distribution $[-0.08, 0.08]$. We set the mini-batch size to 80 and employ the Adam (Kingma and Ba, 2014) for optimization. We utilize the gradient clipping strategy (Pascanu et al., 2013) to avoid gradient explosion, with the gradient clipping value set to 5.

In addition to the shared hyper-parameters, we have particular settings for CVAE-D. The layer size of MLP_p is set to 400. The dimension of latent variable z is set to 300. For the CNN used in the discriminator, its kernel size is set to $(5, 5)$, with the stride size k to 2. We follow the conventional setting (Hu et al., 2017; Creswell et al.,

---

[10] We do not include VAE as our baseline since VAE cannot perform a supervised generation process.

	Automatic Evaluation							Human Evaluation				
Model	**BLEU-1**	**BLEU-2**	**Sim**	**Dist-1**	**Dist-2**	**Dist-3**	**Dist-4**	**Con.**	**Flu.**	**Mea.**	**Poe.**	**Ovr.**
S2S	13.8	2.48	14.7	2.50	16.2	34.9	50.0	1.79	1.84	1.71	1.60	1.74
AS2S	15.5	2.59	14.8	2.30	15.2	31.4	44.3	1.92	1.71	1.80	1.74	1.79
Key-AS2S	15.8	1.92	19.8	3.00	16.3	33.0	45.6	2.21	2.15	1.92	2.23	2.13
MeM-AS2S	16.0	1.48	22.0	3.40	51.4	87.9	96.8	1.70	2.23	2.09	2.89	2.23
GAN	17.7	2.54	22.5	2.50	16.8	35.3	49.6	2.36	2.08	2.01	2.08	2.13
CVAE	17.0	1.73	13.7	4.70	52.3	90.6	99.0	1.69	2.16	2.14	2.58	2.14
CVAE-Key	16.4	1.83	31.0	4.31	43.0	80.6	95.8	1.83	2.29	2.08	2.53	2.18
CVAE-D	**18.1**	**2.85**	**36.3**	**5.20**	**59.2**	**94.2**	**99.8**	**2.58**	**2.35**	**2.34**	**2.96**	**2.56**

Table 3: Results of automatic and human evaluations. **BLEU-1** and **BLEU-2** are BLEU scores on unigrams and bigrams ($p < 0.01$); **Sim** refer to the similarity score; **Dist-**n corresponds to the distinctness of n-gram, with $n = 1$ to 4; **Con., Flu., Mea., Poe., Ovr.** represent consistency, fluency, meaning, poeticness, and overall, respectively.

2017) to set the balancing parameter λ to 0.1.[11]

4.4 Evaluation Metrics

To comprehensively evaluate the generated poems, we employ the following metrics:

BLEU: The BLEU score (Papineni et al., 2002) is an effective metric, widely used in machine translation, for measuring word overlapping between ground truth and generated sentences. In poem generation, BLEU is also utilized as a metric in previous studies (Zhang and Lapata, 2014; Wang et al., 2016a; Yan, 2016; Wang et al., 2016b). We follow their settings in this paper.

Similarity: For thematic consistency, it is challenging to automatically evaluate different models. We adopt the embedding average metric to score sentence-level similarity as that was applied in Wieting et al. (2015). In this paper, we accumulate the embeddings of all characters from the generated poems and that from the given title, and use $cosine$ to compute the similarity between the two accumulated embeddings.

Distinctness: As an important characteristic, poems use novel and unique characters to maintain their elegance and delicacy. Similar to that proposed for dialogue systems (Li et al., 2016), this evaluation is employed to measure character diversity by calculating the proportion of distinctive [1,4]-grams[12] in the generated poems, where final distinctness values are normalized to [0,100].

Human Evaluation: Since writing poems is a complicated task, there always exist incoordinations between automatic metrics and human experiences. Hence, we conduct human evaluation to assess the performance of different models. In doing so, each poem is assessed by five annotators who are well educated and have expertise in Chinese poetry. The evaluation is conducted in a blind review manner, where each annotator has no information about the generation method that each poem belongs to. Following previous work (He et al., 2012; Zhang and Lapata, 2014; Wang et al., 2016c; Zhang et al., 2017a), we evaluate generated poems by four criteria, namely, consistency, fluency, meaning, and poeticness. Each criterion is rated from 1 to 3, representing bad, normal, good, respectively. The details are illustrated in Table 2.

5 Experimental Results

5.1 Quantitative Analysis

Table 3 reports the results of both automatic and human evaluations. We analyze the results from the following aspects.

5.1.1 The effect of CVAE

This study is to investigate whether using latent variable and variational inference can improve the diversity and novelty of terms in generated poems. There are two main observations.

CVAE significantly improves term novelty. As illustrated in Table 3, CVAE outperforms all baselines significantly in terms of distinctness. With diversified terms, the aesthetics scores also confirm that CVAE can generate poems that correspond to better user experiences. Although Mem-AS2S can generate a rather high distinctness score, it requires a more complicated structure in learning and generating poems. The results confirm the effectiveness of CVAE in addressing the issue of term duplications that occurred in RNN.

CVAE cannot control thematic consistency of generated poems. Recall that thematic consistency and term diversity are usually mutually ex-

---

[11] We tried different values for λ, varying from 0.001 to 1, which result in similar performance of the CVAE-D.

[12] Defined as the number of distinctive n-grams divided by the total number of n-grams, shown as Dist-1, Dist-2, Dist-3, Dist-4 in Table 3.

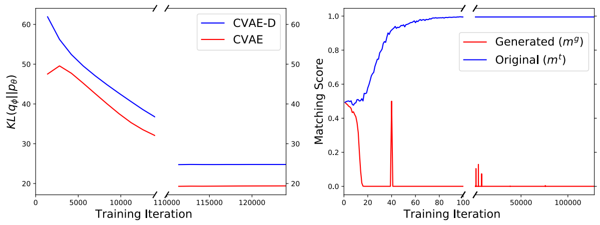

(a) KL divergence (b) Matching score

Figure 4: KL divergences of CVAE and CVAE-D (a) and matching scores of the generated and original lines from the discriminator (b). All curves are drawn against training iterations.

clusive, CVAE produces the worst result in thematic consistency, which is confirmed in Table 3 by the similarity score in automatic evaluation and the consistency score in human evaluation.

5.1.2 The Influence of the Discriminator

As previously stated, introducing a discriminator with adversarial training is expected to bring positive effect on thematic consistency. We investigate the influence of discriminator with two groups of comparison, i.e., CVAE-D v.s. CVAE, GAN v.s. S2S. Following observations are made in this investigation, which confirm that adversarial learning is an effective add-on to existing models for thematics control, without affecting other aspects.

The discriminator effectively enhances poem generation with thematic information. When the discriminator is introduced, CVAE and S2S model are capable of generating thematically consistent poems, as illustrated by the similarity and meaning scores in Table 3. The BLEU results also confirm that the discriminator can improve the overlapping between generated poems and the ground truth, which serves as thematic consistent cases.

The extra discriminator does not affect base models on irrelevant merits. For any base model, e.g., S2S and CVAE, when adding a discriminator, it is expected that it can bring help on thematic consistency while limiting any inferior effects on other evaluations. This is confirmed in the results, e.g., for distinctness, CVAE-D and GAN are comparable to CVAE and S2S.

5.1.3 The Performance of CVAE-D

Overall, the CVAE-D model substantially outperforms all other models in all metrics. Especially for term novelty and thematic consistency, CVAE-D illustrates an extraordinary balance between them, with observable improvements on both sides. This balance is mainly contributed from the proposed framework that seamlessly integrates CVAE and the discriminator. Except for the automatic and human evaluation scores, the fact is also supported by the training loss of $\mathrm{KL}(q_\phi(z|x,c) \parallel p_\theta(z|c))$ and $\mathcal{L}_\mathcal{D}$ as shown in Figure 4, where 1) the KL-divergence of CVAE-D has an analogous trend with CVAE, referring to that the CVAE part in CVAE-D is trained as good as an independent CVAE; 2) the discriminator captures the distinctness of thematic consistency between the generated lines and the ground truth lines at the very early stage of training.

书窗碧桃 Daydream in my garden
庭户风光寄所思， The view in the garden brings up the fantasy, 伊人重过惜残枝。 As if my love dances in the scenery. 窗前花开不知味， Hence blossom can never arouse my curiosity, 唯有落红入我诗。 With only fading memory in the poetry.

Figure 5: An example poem generated by the CVAE-D model. Note that the translation is performed in delivering the meaning instead of the verbatim manner.

5.2 Qualitative Analysis

In addition to evaluating CVAE-D with quantitative results, we also conduct case studies to illustrate its superiority. Figure 5 gives an example of the CVAE-D generated poems, which well demonstrates the capability of our model. The entire poem elegantly expresses a strong theme of "missing my love".[13] It is clearly shown that the choices of the characters, such as 庭 (yard), 枝 (branch), 花 (flower), 红 (red), etc., match with the given title to a certain extent with no one repetitively used. To further investigate how different models perform on thematic consistency, we visualize the correspondence between generated poems (the first two lines) and the given title with heatmaps in Figure 6, where Figure 6(a) and Figure 6(b) illustrate the results yielded by CVAE and CVAE-D, respectively.[14] Obviously, the overall color in Figure 6(a) is lighter than that in Figure 6(b), which

---

[13] "Seeing an object makes one miss someone" is a popular theme in Classical Chinese poems.

[14] Grids in the heatmap represent the correlations between the fine-tuned embeddings of the characters in the title and the generated lines. Since the embeddings are updated in the training process, a better model leads to higher correlations among the embeddings of related characters.

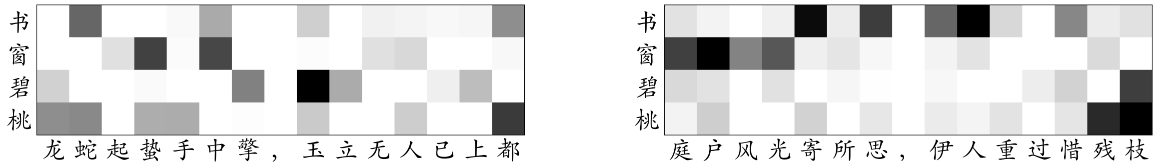

(a) CVAE

(b) CVAE-D

Figure 6: Heatmaps derived from CVAE (a) and CVAE-D (b), in illustrating the correlation between the characters in lines and the title. The horizontal axis refers to characters of the first two lines generated by different models; the vertical axis corresponds to characters in the title. Darker color indicates higher thematic consistency.

may indicate that most of the characters generated by CVAE are not addressed with thematic attentions over the given title. On the opposite, CVAE-D presents darker color in the grids on all related characters, which further reveals the effectiveness of CVAE-D in improving thematic consistency of a poem with respect to its title.

It is observed that there are also inferior cases generated by our model. A notable example pattern is that some fine-grained attributes, e.g., sentiment, emotion, are not well aligned across lines, where some lines may deliver different mood from others. Since our model does not explicitly control such attributes, thus one potential solution to address this issue is to introduce other features to model such information, which requires a special design to adjust the current model. We also notice there exists a few extraordinary bad cases where their basic characteristics, such as wording, fluency, etc., are unacceptable. This phenomenon is randomly observed with no patterns, which could be explained by the complexity of the model and the fragile natural of adversarial training (Goodfellow et al., 2014; Li et al., 2017). Careful parameter setting and considerate module assemble could mitigate this problem, thus lead to potential future work of designing more robust frameworks.

6 Related Work

Deep Generative Models. This work can be seen as an extension of research on deep generative models (Salakhutdinov and Hinton, 2009; Bengio et al., 2014), where most of the previous work, including VAE and CVAE, focused on image generation (Sohn et al., 2015; Yan et al., 2016b). Since GAN (Goodfellow et al., 2014) is also a successful generative model, there are studies tried to integrate VAE and GAN (Larsen et al., 2016). In natural language processing, many recent deep generative models are applied to dialogue systems Serban et al. (2017); Shen et al. (2017); Zhao et al. (2017) and text generation with (Hu et al., 2017; Yu et al., 2017; Lin et al., 2017; Zhang et al., 2017b; Guo et al., 2018). To the best of our knowledge, this work is the first one integrating CVAE and adversarial training with a discriminator for text generation, especially in a particular text genre, poetry.

Automatic Poem Generation. According to methodology, previous approaches can be roughly classified into three categories: 1) rule and template based methods (Tosa et al., 2008; Wu et al., 2009; Netzer et al., 2009; Zhou et al., 2010; Oliveira, 2012; Yan et al., 2013); 2) SMT approaches (Jiang and Zhou, 2008; Greene et al., 2010; He et al., 2012); 3) deep neural models (Zhang and Lapata, 2014; Wang et al., 2016b; Yan, 2016). Compared to rule-based and SMT models, neural models are able to learn more complicated representations and generate smooth poems. Most recent studies followed this paradigm. For example, Wang et al. (2016c) proposed a modified encoder-decoder model with keyword planning; Zhang et al. (2017a) adopted memory-augmented RNNs to dynamically choose each term from RNN output or a reserved inventory. To improve thematic consistency, Yang et al. (2017) combined CVAE and keywords planning. Compared to them, our approach offers an alternative way for poem generation that can produce novel terms and consistent themes via an integrated framework, without requiring special designed modules or post-processing steps.

7 Conclusions

In this paper, we proposed an effective approach that integrates CVAE and adversarial training for classical Chinese poem generation. Specifically, we used CVAE to generate each line of a poem with novel and diverse terms. A discriminator was then applied with adversarial training to explicitly control thematic consistency. Experiments conducted on a large Chinese poem corpus illustrated that through the proposed architecture with CVAE and the discriminator, substantial improvement was observed on the results from our generated poems over those from the existing models. Further qualitative study on given examples and some brief error analyses also confirmed the validity and effectiveness of our proposed approach.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61672058; NSFC No. 61876196). Rui Yan was sponsored by CCF-Tencent Open Research Fund and Microsoft Research Asia (MSRA) Collaborative Research Program.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Eric Thibodeau-Laufer, Guillaume Alain, and Jason Yosinski. 2014. Deep Generative Stochastic Networks Trainable by Backprop. In *International Conference on Machine Learning*, pages 226–234, Beijing, China.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555*.

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

Antonia Creswell, Anil A Bharath, and Biswa Sengupta. 2017. Conditional Autoencoders with Adversarial Information Factorization. *arXiv preprint arXiv:1711.05175*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised Sequence Learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, Montreal, Canada.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating Topical Poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, Austin,USA.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an Interactive Poetry Generation System. *Proceedings of ACL 2017, System Demonstrations*, pages 43–48.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-LM: A Neural Language Model for Customizable Affective Text Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 634–642, Vancouver, Canada.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in neural information processing systems*, pages 2672–2680, Montreal, Canada.

Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. 2016. Professor Forcing: A new Algorithm for Training Recurrent Networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, Barcelona, Spain.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic Analysis of Rhythmic Poetry with Applications to Generation and Translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 524–533, Massachusetts, USA.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long Text Generation via Adversarial Training with Leaked Information. *AAAI*.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating Chinese Classical Poems with Statistical Machine Translation Models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1650–1656, Toronto, Canada.

Jack Hopkins and Douwe Kiela. 2017. Automatically Generating Rhythmic Verse with Neural Networks. In *Meeting of the Association for Computational Linguistics*, pages 168–178, Vancouver, Canada.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward Controlled Generation of Text. In *International Conference on Machine Learning*, pages 1587–1596, Sydney, Australia.

Long Jiang and Ming Zhou. 2008. Generating Chinese Couplets using a Statistical MT Approach. In *COLING 2008, International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, Uk*, pages 377–384, Manchester, United Kingdom.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally Coherent Text Generation with Neural Checklist Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, USA.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *stat*, 1050:10.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. 2016. Autoencoding Beyond Pixels Using a Learned Similarity Metric. In *International Conference on Machine Learning*, pages 1558–1566, New York, USA.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, USA.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A Hierarchical Neural Autoencoder for Paragraphs and Documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1106–1115, Beijing, China.

Jiwei Li, Will Monroe, Tianlin Shi, Sèbastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial Learning for Neural Dialogue Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169, Copenhagen, Denmark.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial Ranking for Language Generation. In *Advances in Neural Information Processing Systems*, pages 3155–3165.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th international conference on machine learning*, pages 807–814, Haifa, Israel.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku: Generating Haiku with Word Associations Norms. *Computational Approaches to Linguistic Creativity*, page 32.

Hugo Gonçalo Oliveira. 2012. PoeTryMe: a Versatile Platform for Poetry Generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL*, pages 311–318, Philadelphia, USA.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the Difficulty of Training Recurrent Neural Networks. In *International Conference on Machine Learning*, pages 1310–1318, Atlanta,USA.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence Level Training with Recurrent Neural Networks. *arXiv preprint arXiv:1511.06732*.

Ruslan Salakhutdinov and Geoffrey E Hinton. 2009. Deep Boltzmann Machines. In *AISTATS*, pages 448–455.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. In *AAAI*, pages 3295–3301, San Francisco, USA.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A Conditional Variational Framework for Dialog Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 504–509, Vancouver, Canada.

Divya Singh, Margareta Ackerman, and Rafael Pérez y Pérez. 2017. A Ballad of the Mexicas: Automated Lyrical Narrative Writing. In *Eighth International Conference on Computational Creativity, ICCC, Atlanta*.

Kihyuk Sohn, Xinchen Yan, and Honglak Lee. 2015. Learning Structured Output Representation using Deep Conditional Generative Models. In *International Conference on Neural Information Processing Systems*, pages 3483–3491, Montreal, Canada.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in neural information processing systems*, pages 3104–3112, Montreal, Canada.

Naoko Tosa, Hideto Obara, and Michihiko Minoh. 2008. Hitch Haiku: An Interactive Supporting System for Composing Haiku Poem. In *International Conference on Entertainment Computing*, pages 209–216. Springer.

Qixin Wang, Tianyi Luo, and Dong Wang. 2016a. Can Machine Generate Traditional Chinese Poetry? A Feigenbaum Test. In *International Conference on Brain Inspired Cognitive Systems*, pages 34–46.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016b. Chinese Song Iambics Generation with Neural Attention-based Model. In *International Joint Conference on Artificial Intelligence*, pages 2943–2949, New York, 2016.

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016c. Chinese Poetry Generation with Planning based Neural Network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1051–1060, Osaka, Japan.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards Universal Paraphrastic Sentence Embeddings. *arXiv preprint arXiv:1511.08198*.

Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. 2009. New Hitch Haiku: An Interactive Renku Poem Composition Supporting Tool Applied for Sightseeing Navigation System. In *International Conference on Entertainment Computing*, pages 191–196. Springer.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 496–505, Vancouver, Canada.

Linli Xu, Liang Jiang, Chuan Qin, Zhe Wang, and Dongfang Du. 2018. How Images Inspire Poems: Generating Classical Chinese Poetry from Images with Memory Networks. *arXiv preprint arXiv:1803.02994*.

Rui Yan. 2016. i, Poet: Automatic Poetry Composition through Recurrent Neural Networks with Iterative Polishing Schema. In *International Joint Conference on Artificial Intelligence*, pages 2238–2244, New York, USA.

Rui Yan, Han Jiang, Mirella Lapata, Shou De Lin, Xueqiang Lv, and Xiaoming Li. 2013. i, Poet: Automatic Chinese Poetry Composition through a Generative Summarization Framework under Constrained Optimization. In *International Joint Conference on Artificial Intelligence*, pages 2197–2203, Beijing, China.

Rui Yan, Cheng-Te Li, Xiaohua Hu, and Ming Zhang. 2016a. Chinese couplet generation with neural network structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2347–2357.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016b. Attribute2image: Conditional Image Generation from Visual Attributes. In *European Conference on Computer Vision*, pages 776–791, Amsterdam, Netherlands. Springer.

Xiaopeng Yang, Xiaowen Lin, Shunda Suo, and Ming Li. 2017. Generating Thematic Chinese Poetry using Conditional Variational Autoencoder. *arXiv preprint arXiv:1711.07632*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*, pages 2852–2858, San Francisco, USA.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017a. Flexible and Creative Chinese Poetry Generation Using Neural Memory. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1364–1373, Vancouver, Canada.

Xingxing Zhang and Mirella Lapata. 2014. Chinese Poetry Generation with Recurrent Neural Networks. In *Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Doha, Qatar.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017b. Adversarial Feature Matching for Text Generation. In *International Conference on Machine Learning*, pages 4006–4015.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017c. Aspect-augmented Adversarial Networks for Domain Adaptation. *Transactions of the Association of Computational Linguistics*, 5(1):515–528.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 654–664, Vancouver, Canada.

Chang Le Zhou, Wei You, and Xiao Jun Ding. 2010. Genetic Algorithm and Its Implementation of Automatic Generation of Chinese SONGCI. *Journal of Software*, 21(3):427–437.

Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks

Yao Zhao **Xiaochuan Ni** **Yuanyuan Ding** **Qifa Ke**
Microsoft AI & Research
Sunnyvale, California
{yaozhao, xiaon, yuand, qke}@microsoft.com

Abstract

Question generation, the task of automatically creating questions that can be answered by a certain span of text within a given passage, is important for question-answering and conversational systems in digital assistants such as Alexa, Cortana, Google Assistant and Siri. Recent sequence to sequence neural models have outperformed previous rule-based systems. Existing models mainly focused on using one or two sentences as the input. Long text has posed challenges for sequence to sequence neural models in question generation – worse performances were reported if using the whole paragraph (with multiple sentences) as the input. In reality, however, it often requires the whole paragraph as context in order to generate high quality questions. In this paper, we propose a maxout pointer mechanism with gated self-attention encoder to address the challenges of processing long text inputs for question generation. With sentence-level inputs, our model outperforms previous approaches with either sentence-level or paragraph-level inputs. Furthermore, our model can effectively utilize paragraphs as inputs, pushing the state-of-the-art result from 13.9 to 16.3 (BLEU_4).

1 Introduction

Question generation (QG), aiming at creating questions from natural language text, e.g. a sentence or paragraph, is an important area in natural language processing (NLP). It is receiving increasing interests in recent years from both industrial and academic communities, due to the booming of Question-and-Answer (QnA) and conversation systems, including Alexa, Cortana, Google Assistant and Siri, the advancement of QnA or machine comprehension technologies together with the releases of datasets like SQuAD (Rajpurkar et al., 2016) and MS MARCO (Nguyen et al., 2016), and the success of language generation technologies for tasks like machine translation (Wu et al., 2016) and text summarization (See et al., 2017) in NLP. A conversational system can be proactive by asking the user questions (Shum et al., 2018), while a QnA system can benefit from a large scale question-answering corpus which can be created by an automated QG system (Duan et al., 2017). Education is another key application where QG can help with reading comprehension (Heilman and Smith, 2010).

In NLP, QG has been mainly tackled by two approaches: 1) rule-based approach, e.g. (Heilman and Smith, 2010; Mazidi and Nielsen, 2014; Labutov et al., 2015) 2) neural QG approach: end-to-end training a neural network using the sequence to sequence (also called encoder-decoder) framework, e.g. (Du et al., 2017; Yuan et al., 2017; Song et al., 2017; Zhou et al., 2017). In this paper, we adopt the second approach.

More specifically, we focus on an answer-aware QG problem, which takes a passage and an answer as inputs, and generates a question that targets the given answer. It is also assumed the answer is comprised of certain spans of the text from the given passage. This is the exact setting of SQuAD, and similar problems have been addressed in, e.g. (Zhou et al., 2017; Yuan et al., 2017; Wang et al., 2017a).

A paragraph often contains much richer context than a sentence, as shown in Figure 1. (Du et al., 2017) pointed out that about 20% questions in SQuAD require paragraph-level information to be asked and using the whole paragraph can improve QG performance on those questions. However, a paragraph can contain irrelevant information w.r.t. the answer for generating the question. The challenge is thus how to effectively utilize relevant information at paragraph-level for QG. Existing neural QG works conducted on SQuAD use

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3901–3910
Brussels, Belgium, October 31 - November 4, 2018.

1 **Paragraph**: carolina suffered a major setback when thomas davis , an 11-year veteran who had already overcome three acl tears in his career , went down with a broken arm in the nfc championship game . *despite this , he insisted he would still find a way to play in the super bowl* . his prediction turned out to be accurate
Human generated: what game did thomas davis say he would play in , despite breaking a bone earlier on ?
Sentence-level QG: what sports game did spielberg decide to play in ?
Paragraph-level QG: what competition did thomas davis think he would play in ?

2 **Paragraph**: walt disney and his brother roy contacted goldenson at the end of 1953 for abc to agree to finance part of the disneyland project in exchange for producing a television program for the network . *walt wanted abc to invest $ 500,000 and accrued a guarantee of $ 4.5 million in additional loans , a third of the budget intended for the park .* around 1954 , abc agreed to finance disneyland in exchange for the right to broadcast a new sunday night program , disneyland , which debuted on the network on october 27 , 1954 as the first of many anthology television programs that disney would broadcast over the course of the next 50 years
Human generated: how much did walt disney want abc to invest in disneyland ?
Sentence-level QG: how much money did walt wanted to invest ?
Paragraph-level QG: how much money did walt wanted to invest in 1953 ?

3 **Paragraph**: following the peterloo massacre of 1819 , poet percy shelley wrote the political poem the mask of anarchy later that year , that begins with the images of what he thought to be the unjust forms of authority of his timeand then imagines the stirrings of a new form of social action . *it is perhaps the first modern [vague] statement of the principle of nonviolent protest .* a version was taken up by the author henry david thoreau in his essay civil disobedience , and later by gandhi in his doctrine of satyagraha . gandhi 's satyagraha was partially influenced and inspired by shelley 's nonviolence in protest and political action . in particular , it is known that gandhi would often quote shelley 's masque of anarchy to vast audiences during the campaign for a free india
Human generated: his poem is considered the first kind of what type of protest ?
Sentence-level QG: what is the principle of the protest ?
Paragraph-level QG: what type of protest did percy shelley write ?

4 **Paragraph**: the victoria and albert museum (often abbreviated as the v & a) , london , is the world 's largest museum of decorative arts and design , housing a permanent collection of over 4.5 million objects *. it was founded in 1852 and named after queen victoria and prince albert.* the v & a is located in the brompton district of the royal borough of kensington and chelsea , in an area that has become known as " albertopolis " because of its association with prince albert , the albert memorial and the major cultural institutions with which he was associated . these include the natural history museum , the science museum and the royal albert hall . the museum is a non-departmental public body sponsored by the department for culture , media and sport . like other national british museums , entrance to the museum has been free since 2001
Human generated: when was the victoria and albert museum founded ?
Sentence-level QG: when was prince albert and prince albert founded ?
Paragraph-level QG: when was the victoria and albert museum founded ?

Figure 1: Examples where paragraph-level information is required to ask right questions. *Sentences* contain *answers* are in *italic* font, while *answers* are underscored. QG results are generated by model *s2s-a-ct-mp-gsa*.

only a sentence as context, e.g. (Du et al., 2017; Song et al., 2017; Zhou et al., 2017); when applied to paragraph-level context (Yuan et al., 2017) we observed large gaps compared to state-of-the-art results achieved by using sentence-level context.

In this paper, we extend previous sequence to sequence attention model with a maxout pointer mechanism and a gated self-attention encoder which outperforms existing neural QG approaches with either sentence or paragraph as inputs. Furthermore, with paragraph-level inputs, it outperforms the results of previous approaches with sentence-level inputs, improving state-of-the-art result from 13.9 to 16.3 (BLEU_4). This is the first model that demonstrates large improvement with paragraph as input over sentence as input. In addition, our model is more concise compared to most of existing ones, e.g. (Yuan et al., 2017; Song et al., 2017). Techniques like incorporating rich features (Zhou et al., 2017) and policy gradient (Song et al., 2017; Yuan et al., 2017) are orthogonal to ours and can be leveraged for further performance improvement in the future.

2 Our Model

2.1 Problem Definition

We use P and A to represent input passage and answer respectively, and use Q to represent the generated question. ”Passage” in this section can represent either a sentence or a paragraph. The task is to find $\bar{Q}$ that:

$$\bar{Q} = \underset{Q}{\operatorname{argmax}}\, Prob(Q|P, A)$$

where passage is comprised of sequence of words:$P = \{x_t\}_{t=1}^{M}$, answer A must be sub spans of the passage. Words generated in $Q = \{y_t\}_{t=1}^{K}$ are either from the input passage, $\{x_t\}_{t=1}^{M}$, or from a vocabulary V.

Figure 2 illustrates the end-to-end structure of our model proposed in this paper.

2.2 Passage and Answer Encoding

Different types of encoders are designed for various domains (Chung et al., 2014; Hochreiter and Schmidhuber, 1997). We are agnostic to the form of the encoder and simply use recurrent neural network (RNN) to present the encoding process:

$$\mathbf{u}_t = RNN^E(\mathbf{u}_{t-1}, [\mathbf{e}_t, \mathbf{m}_t]) \qquad (1)$$

Answer Tagging. In Eq. 1, $\mathbf{u}_t$ represents the RNN hidden state at time step t, $\mathbf{e}_t$ is the word embedding representation of word x_t in passage P. $\mathbf{m}_t$ is the meta-word representation of whether word x_t is in or outside the answer. $[\mathbf{a}, \mathbf{b}]$ represents the concatenation of vector $\mathbf{a}$ and $\mathbf{b}$. We call this approach answer tagging which is similar to the techniques in (Zhou et al., 2017; Yuan et al., 2017). For applications, it is essential to be able to generate question that is coherent to an answer.

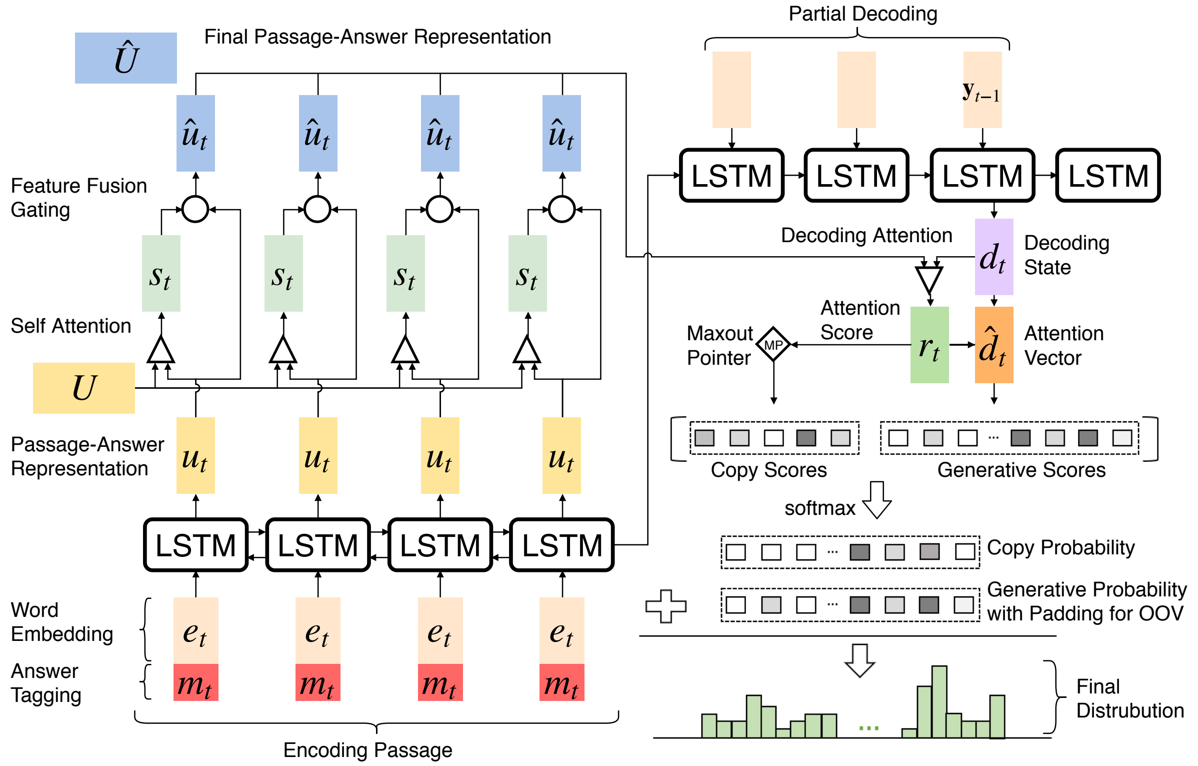

Figure 2: End-to-end diagram for the model with answer tagging, gated self-attention and maxout pointer mechanism.

If RNN^E is bi-directional, $\mathbf{u}$ is the concatenated representation of the forward and backward passes: $\mathbf{U} = \{[\overrightarrow{\mathbf{u}_t}, \overleftarrow{\mathbf{u}_t}]\}_{t=1}^{M}$.

Gated Self-attention. Our gated self-attention mechanism is designed to aggregate information from the whole passage and embed intra-passage dependency to refine the encoded passage-answer representation at every time step. It has two steps: 1) taking encoded passage-answer representation $\mathbf{u}$ as input and conducting matching against itself to compute self matching representation; (Wang et al., 2017b) 2) combining the input with self matching representation using a feature fusion gate (Gong and Bowman, 2017).

$$\mathbf{a^s}_t = softmax(\mathbf{U}^\mathsf{T}\mathbf{W}^s\mathbf{u}_t) \tag{2}$$

$$\mathbf{s}_t = \mathbf{U} \cdot \mathbf{a^s}_t \tag{3}$$

Step 1. In Eq. 2, $\mathbf{W}^s$ is a trainable weight matrix. In Eq. 3, $\mathbf{s}_t$ is the weighted sum of all words' encoded representation in passage based on their corresponding matching strength to current word at t. $\mathbf{s} = \{\mathbf{s}_t\}_{t=1}^{M}$ is the final self matching representation.

$$\mathbf{f}_t = tanh(\mathbf{W}^f[\mathbf{u}_t, \mathbf{s}_t]) \tag{4}$$

$$\mathbf{g}_t = sigmoid(\mathbf{W}^g[\mathbf{u}_t, \mathbf{s}_t]) \tag{5}$$

$$\hat{\mathbf{u}}_t = \mathbf{g}_t \odot \mathbf{f}_t + (1 - \mathbf{g}_t) \odot \mathbf{u}_t \tag{6}$$

Step 2. The self matching representation $\mathbf{s}_t$ is combined with original passage-answer representation $\mathbf{u}_t$ as the new self matching enhanced representation $\mathbf{f}_t$, Eq. 4. A learnable gate vector $\mathbf{g}_t$, Eq. 5, chooses the information between the original passage-answer representation and the new self matching enhanced representation to form the final encoded passage-answer representation $\hat{\mathbf{u}}_t$, Eq. 6, where $\odot$ is the element-wise multiplication.

2.3 Decoding with Attention and Maxout Pointer

In the decoding stage, the decoder is another RNN that generates words sequentially conditioned on the encoded input representation and the previously decoded words.

$$\mathbf{d}_t = RNN^D(\mathbf{d}_{t-1}, \mathbf{y}_{t-1}) \tag{7}$$

$$p(y_t|\{y_{<t}\}) = softmax(\mathbf{W}^V\mathbf{d}_t) \tag{8}$$

In Eq. 7, $\mathbf{d}_t$ represents the hidden state of the RNN at time t where $\mathbf{d}_0$ is passed from the final hidden state of the encoder. y_t stands for the word generated at time t. The bold font $\mathbf{y}_t$ is used to represent y_t's corresponding word embedding representation. In Eq. 8, first an affine layer projects $\mathbf{d}_t$ to a space with vocabulary-size dimensions, then a $softmax$ layer computes a probability distribution over all words in a fixed vocabulary V . $\mathbf{W}^V$ is a trainable weight matrix.

Attention. Attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) has been used to improve sequence to sequence models' performance and has became a default setting for many applications.

We use Luong attention mechanism to compute raw attention scores $\mathbf{r}_t$, Eq. 9. An attention layer, Eq. 12 is applied above the concatenation of decoder state $\mathbf{d}_t$ and the attention context vector $\mathbf{c}_t$ and its output is used as the new decoder state.

$$\mathbf{r}_t = \hat{\mathbf{U}}^\mathsf{T} \mathbf{W}^a \mathbf{d_t} \tag{9}$$

$$\mathbf{a^d}_t = softmax(\mathbf{r}_t) \tag{10}$$

$$\mathbf{c}_t = \hat{\mathbf{U}} \cdot \mathbf{a^d}_t \tag{11}$$

$$\hat{\mathbf{d}}_t = tanh(\mathbf{W}^b[\mathbf{d}_t, \mathbf{c}_t]) \tag{12}$$

Copy/Pointer. Copy mechanism (Gu et al., 2016) or pointer network (See et al., 2017; Vinyals et al., 2015) was introduced to allow both copying words from input via pointing, and generating words from a predefined vocabulary during decoding.

Similar to (Gu et al., 2016), our pointer mechanism directly leverages raw attention scores $\mathbf{r}_t = \{r_{t,k}\}_{k=1}^{M}$ over the input sequence which has a vocabulary of χ. Words at every time step (a pointer) are treated as unique copy targets and the final score on one word is calculated as the sum of all scores pointing to the same word, Eq. 13, where x_k and y_t stand for word vocabulary indices of the kth word in input and the tth word in decoded sequence respectively. The scores of the nonoccurence words are set to negative infinity which will be masked out by the downstream $softmax$ function.

$$\mathbf{sc}^{copy}(y_t) = \begin{cases} \sum\limits_{k, where\ x_k = y_t} r_{t,k}, & y_t \in \chi \\ -inf, & otherwise \end{cases} \tag{13}$$

We then concatenate $\mathbf{sc}_t^{copy}$ with the generative scores (from Eq. 8 before $softmax$), $[\mathbf{sc}_t^{gen}, \mathbf{sc}_t^{copy}]$, which has dimension: $|V| + |\chi|$. Then we perform $softmax$ on the concatenated vectors and sum up the probabilities pointing to same words. Taking $softmax$ on the concatenated score vector enforces copy and generative modes to compete with each other due to the shared normalization denominator. Another popular solution is to do $softmax$ independently to the scores from each mode, and then combine their output probabilities with a dynamic weight which is generated by a trainable network (See et al., 2017). We have tested both and didn't find significant difference in terms of accuracy on our QG task. We choose the former copy approach mainly because it doesn't add extra trainable parameters.

Maxout Pointer. Despite the outstanding performance of existing copy/pointer mechanisms, we observed that repeated occurrence of words in the input sequence tends to cause repetitions in output sequence, especially when the input sequence is long, e.g. a paragraph. This issue exacerbates the repetition problem which has already been commonly observed in sequence to sequence models (Tu et al., 2016; See et al., 2017). In this paper, we propose a new maxout pointer mechanism to address this issue and improve the metrics for QG task. Related works (Goodfellow et al., 2013) have explored MLP maxout with dropout.

Instead of combining all the scores to calculate the probability, We limit the magnitude of scores of repeated words to their maximum value, as shown in Eq. 14. The rest remains the same as in the previous copy mechanism.

$$\mathbf{sc}^{copy}(y_t) = \begin{cases} \max\limits_{k, where\ x_k = y_t} r_{t,k}, & y_t \in \chi \\ -inf, & otherwise \end{cases} \tag{14}$$

3 Experiments

In our experiments, we study the proposed model on the QG task on SQuAD (Rajpurkar et al., 2016) and MS MARCO (Nguyen et al., 2016) dataset, demonstrate the performance of proposed components on both sentence and paragraph inputs, and compare the model with existing approaches.

3.1 Dataset

SQuAD

The SQuAD dataset contains 536 Wikipedia articles and more than 100K questions posed about the articles by crowd-workers. Answers are also

	BLEU_1		**BLEU_2**		**BLEU_3**		**BLEU_4**		**METEOR**		**ROUGE-L**	
Model	**Sen.**	**Par.**	**Sen.**	**Par.**	**Sen.**	**Par.**	**Sen.**	**Par.**	**Sen.**	**Par.**	**Sen.**	**Par.**
s2s	30.41	28.49	12.68	10.43	6.33	4.70	3.44	2.38	11.98	10.69	29.93	27.32
s2s-a	34.46	31.26	18.07	14.37	11.20	8.02	7.42	4.80	14.95	12.52	34.69	30.11
s2s-a-at	40.57	40.56	24.30	24.23	16.40	16.33	11.54	11.46	18.35	18.42	40.76	40.40
s2s-a-at-cp	42.15	41.66	26.28	25.52	18.35	17.48	13.37	12.43	18.02	17.76	41.97	41.30
s2s-a-at-mcp	**43.65**	44.22	**28.23**	28.56	20.33	20.57	15.23	15.43	19.19	19.55	43.60	43.65
s2s-a-at-mcp-gsa	43.47	**45.07**	**28.23**	**29.58**	**20.40**	**21.60**	**15.32**	**16.38**	**19.29**	**20.25**	**43.91**	**44.48**

Table 1: Performance of our models on **Split1** with both sentence-level input and paragraph-level input. **Sen.** means sentence, while **Par.** means paragraph.

provided to the questions, which are spans of tokens in the articles.

Following (Du et al., 2017; Zhou et al., 2017; Song et al., 2017), our experiments are conducted using the accessible part of SQuAD: train and development (dev*) sets. To be able to directly compare with their works, we adopt two types of data split: 1) **Split1**: similar to (Du et al., 2017), we use dev* set as test set, and split train set into train and dev sets randomly with ratio 90%-10%. The split is done at article level. However, we keep all samples instead of only keeping the sentence-question pairs that have at least one non-stop-word in common (with 6.7% pairs dropped) as in (Du et al., 2017). This makes our dataset harder for training and evaluation. 2) **Split2**: similar to (Zhou et al., 2017), we split dev* set into dev and test sets randomly with ratio 50%-50%. The split is done at sentence level.

MS MARCO

MS MARCO datasets contains 100,000 queries with corresponding answers and passages. All questions are sampled from real anonymized user queries and context passages are extracted from real web documents. We picked a subset of MS MARCO data where answers are sub-spans within the passages, and use dev set as test set (7k), and split train set with ratio 90%-10% into train (51k) and dev (6k) sets.

3.2 Implementation Details

We used 2 layers LSTM as the RNN cell for both encoding and decoding. For encoding, bi-directional LSTM was used. The cell hidden size was 600. Dropout with probability 0.3 was applied between vertical LSTM stacks. For word embedding, we used pre-trained GloVe word vectors with 300 dimensions (Pennington et al., 2014), and froze them during training. Dimension of answer tagging meta-word embedding was 3. Both encoder and decoder shared the same vocabulary of the most frequent 45k GloVe words. For optimization, we used SGD with momentum (Qian, 1999; Nesterov, 1983). Learning rate was initially set to 0.1 and halved since epoch 8 at every 2 epochs afterwards. Models were totally trained with 20 epochs. The mini-batch size for parameter update was 64. After training, we looked at the 4 models with lowest perplexities and selected the one which used the most number of epochs as final model. During decoding for prediction, we used beam search with the beam size of 10, and stopped decoding when every beam in the stack generates the EOS token.

3.3 Evaluation

We conduct automatic evaluation with metrics: BLEU_1, BLEU_2, BLEU_3, BLEU_4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and ROUGE-L (Lin, 2004), and use evaluation package released by (Sharma et al., 2017) to compute them.

4 Results and Analysis

4.1 Comparison of Techniques

Table 1 shows evaluation results for different models on SQuAD Split1. Results with both sentence-level and paragraph-level inputs are included. Similar results also have been observed on SQuAD Split2. The definitions of the models under comparison are:

s2s: basic sequence to sequence model
s2s-a: *s2s* + attention mechanism
s2s-a-at: *s2s-a* + answer tagging
s2s-a-at-cp: *s2s-a-at* + copy mechanism

s2s-a-at-mp: *s2s-a-at* + maxout pointer mechanism
s2s-a-at-mp-gsa: *s2s-a-at-mp* + gated self-attention

Attention Mechanism

s2s-a vs. *s2s*: we can see attention brings in large improvement on both sentence and paragraph inputs. The lower performance on paragraph indicates the challenge of encoding paragraph-level information.

Answer Tagging

s2s-a-at vs. *s2s-a*: answer tagging dramatically boosts the performance, which confirms the importance of answer-aware QG: to generate good question, we need to control/learn which part of the context the generated question is asking about. More importantly, answer tagging clearly reduces the gap between sentence and paragraph inputs, which could be explained with: by providing guidance on answer words, we can make the model learn to neglect noise when processing a long context.

Copy Mechanism

s2s-a-at-cp vs. *s2s-a-at*: as expected, copy mechanism further improves the performance on the QG task. (Du et al., 2017) pointed out most of the sentence-question pairs in SQuAD have over 50% overlaps in non-stop-words. Our results prove that sequence to sequence models with copy mechanism can very well learn when to generate a word and when to copy one from input on such QG task. More interestingly, the performance is lower when paragraph is given as input than sentence as input. The gap, again, reveals the challenge of leveraging longer context. We found that, when paragraph is given, the model tends to generate more repetitive words, and those words (often entities/concepts) usually appear multiple times in the context, Figure 3. The repetition issue can also be seen for sentence input, but it is more severe for paragraph.

Maxout Pointer

s2s-a-at-mp vs. *s2s-a-at-cp*: Maxout pointer is designed to resolve the repetition issue brought by the basic copy mechanism, for example Figure 3. The maxout pointer mechanism outperforms the basic copy mechanism in all metrics. Moreover, the effectiveness of maxout pointer is more significant when paragraph is given as the model input, as it reverses the performance gap between models trained with sentence and paragraph inputs, Table 1.

Paragraph: a problem is regarded as inherently difficult if its solution requires significant resources , whatever the algorithm used . the theory formalizes this intuition , by introducing mathematical models of computation to study these problems and quantifying the amount of resources needed to solve them , such as time and storage . *other complexity measures are also used , such as the amount of communication (used in communication complexity) , the number of gates in a circuit (used in circuit complexity) and the number of processors (used in parallel computing) .* one of the roles of computational complexity theory is to determine the practical limits on what computers can and can not do
Human generated: what unit is measured to determine circuit complexity ?
Basic copy QG: what is an example of a circuit complexity in complexity complexity ?
Maxout Pointer QG: what is another name for circuit complexity ?

Figure 3: Example for maxout pointer vs. basic copy/pointer.

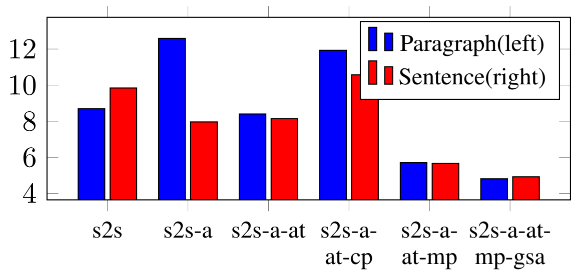

Figure 4: Word duplication rates of QG on Split1, duplication rate on human generated question is **3.53**.

To demonstrate that maxout pointer reduces repetitions in generated questions, we present the word duplication rates in generated questions for various models in Figure 4. Word duplication rate was computed by counting the number of words which appear more than once, and then taking a ratio of them over the total word counts. As shown in Figure 4, both the attention mechanism and the basic copy mechanism introduce more repetitions, although they improve overall accuracy according to Table 1. For models trained with paragraph inputs, where the duplication rates are much higher, maxout pointer reduces the duplication rates to half of their values in the basic copy and to the same level as model trained with sentence inputs.

Such repetition issue was also observed in other sequence to sequence applications, e.g. (See et al., 2017) who proposed a coverage model and coverage loss to resolve this issue. We implemented and tested their approach on our QG task. Even though the duplication ratio dropped as expected, we observed a slight decline in the accuracy when coverage loss was added.

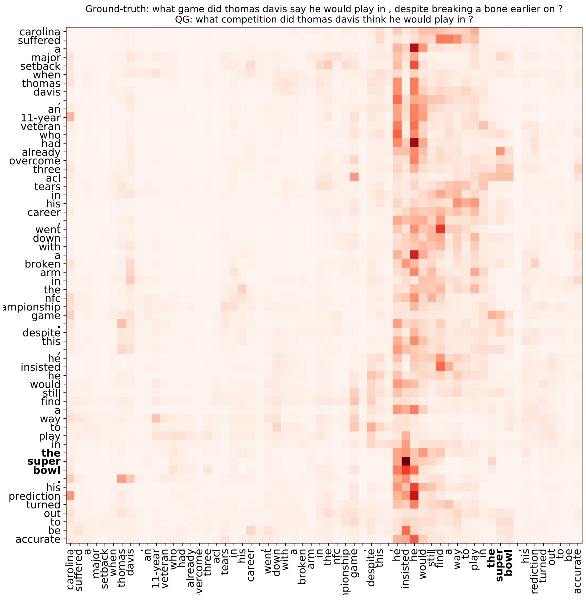

Figure 5: Self-attention alignments map: each row represents an alignment vector of self-attention.

Gated Self-attention

s2s-a-at-mp-gsa vs. *s2s-a-at-mp*: the results demonstrate the effectiveness of gated self-attention, in particular, when working with paragraph inputs. This is the first time, as we know, taking paragraph as input is better than sentence for neural QG tasks. The observation is consistent across all metrics. Gated self-attention helps refine encoded context by fusing important information with the context's self representation properly, especially when the context is long.

To better understand how gated self-attention works, we visualize the self alignment vectors at each time step of the encoded sequence for one example, in Figure 5. This example corresponds to the example 1 in Figure 1. We can see the alignments distribution concentrates near the answer sequence and the most relevant context: "thomas davis" in this example. Such alignments in turn would help to promote the most valuable information for decoding.

Beam Search

Beam search is commonly used for decoding for predictions. Existing neural QG works, e.g. (Du et al., 2017; Zhou et al., 2017), evaluated their models with beam search decoding. However, so far, none of them have reported the comparison between beam search decoding with greedy decoding. In this paper, we give such comparison for our best model: *s2s-a-at-mp-gsa* with both sentence and paragraph inputs in Table 2. We can clearly see beam search decoding boosts all metrics for both sentence and paragraph inputs. The effectiveness of beam search has also been demonstrated for other tasks, like neural machine translation in (Wu et al., 2016).

	Beam_Search		**Greedy**	
Metric	**Sen.**	**Par.**	**Sen.**	**Par.**
BLEU_1	43.47	45.07	42.25	43.48
BLEU_2	28.23	29.58	26.36	27.67
BLEU_3	20.40	21.60	18.35	19.64
BLEU_4	15.32	16.38	13.28	14.50
METEOR	19.29	20.25	18.25	19.17
ROUGE-L	43.91	44.48	42.58	43.62

Table 2: Comparison of beam search and greedy decodings for model **s2s-a-ct-mp-gsa** on Split1.

4.2 Comparison with Existing Neural Question Generation Works

On SQuAD dataset, we compare the BLEU, METEOR, ROUGE-L scores of our best model, *s2s-a-at-mp-gsa*, with the numbers in the existing works in Table 3. Comparison with (Du et al., 2017) and (Song et al., 2017) is conducted on SQuAD data Split1, while comparison with (Zhou et al., 2017) and (Song et al., 2017) is conducted on data SQuAD split2. Because (Song et al., 2017) had results on both splits, we compare with both of them.

On MS MARCO dataset, we compare the BLEU_4 scores reported by (Duan et al., 2017) in Table 4.

Our model with maxout pointer and gated self-attention achieves the state-of-the-art results in QG. Note that all those existing works in SQuAD encoded only sentence-level information, the results from our model surpass them on the same sentence input while achieving much higher numbers when working with paragraph.

4.3 Case Study

In Figure 1, we present some examples for which paragraph-level information is needed to ask good/correct questions. Generated questions from model *s2s-a-ct-mp-gsa* are also presented for both sentence and paragraph inputs. Those examples demonstrate that generated questions contain richer information when paragraphs are provided

	Model	BLEU_1	BLEU_2	BLEU_3	BLEU_4	METEOR	ROUGE-L
Split1	(Du et al., 2017)	43.09	25.96	17.50	12.28	16.62	39.75
	(Song et al., 2017)	x	x	x	13.98	18.77	42.72
	ours, sentence	43.47	28.23	20.40	15.32	19.29	43.91
	ours, paragraph	**45.07**	**29.58**	**21.60**	**16.38**	**20.25**	**44.48**
Split2	(Zhou et al., 2017)	x	x	x	13.27	x	x
	(Song et al., 2017)	x	x	x	13.91	x	x
	ours, sentence	44.51	29.07	21.06	15.82	19.67	44.24
	ours, paragraph	**45.69**	**30.25**	**22.16**	**16.85**	**20.62**	**44.99**

Table 3: Comparison of results on SQuAD dataset.

	Model	BLEU_4
MSMARCO	(Du et al., 2017)	10.46
	(Duan et al., 2017)	11.46
	ours, sentence	16.02
	ours, paragraph	**17.24**

Table 4: Comparison of results on MSMARCO dataset.

instead of sentences.

In example 1 and 3, name ”thomas davis” and ”percy shelley” appear in the paragraphs not the sentences contain the answers.

In example 2, paragraph-level QG can generate richer description ”in 1953” from the paragraph, although human generated is ”disneyland”. Both generated questions lack one relative important piece of information ”want **abc** to invest”.

In example 4, paragraph-level QG correctly identifies ”it” is referring to the museum which is out of the sentence.

5 Related Work

QG has been mainly tackled with two types of approaches. One is built on top of heuristic rules that creates questions with manually constructed template and ranks the generated results, e.g. (Heilman and Smith, 2010; Mazidi and Nielsen, 2014; Labutov et al., 2015). Those approaches heavily depend on human effort, which makes them hard to scale up to many domains. The other one, which is becoming increasingly popular, is to train an end-to-end neural network from scratch by using sequence to sequence or encoder-decoder framework, e.g. (Du et al., 2017; Yuan et al., 2017; Song et al., 2017; Zhou et al., 2017). The second one is more related to us, so we will focus on describing those approaches.

(Du et al., 2017) pioneered the work of automatic QG using an end-to-end trainable sequence to sequence neural model. Automatic and human evaluation results showed that their system outperformed the previous rule-based systems (Heilman and Smith, 2010; Rus et al., 2010). However, in their study, there was no control about which part of the passage the generated question was asking about.

Answer-aware sequence to sequence neural QG systems (Zhou et al., 2017; Subramanian et al., 2017; Yuan et al., 2017) encoded answer location information using an annotation vector corresponding to the answer word positions. (Zhou et al., 2017) utilized rich features of the passage including answer positions. (Subramanian et al., 2017) deployed a two-stage neural model that detects key phrases and subsequently generates questions conditioned on them. (Yuan et al., 2017) combined supervised and reinforcement learning in the training of their model using policy gradient techniques to maximize several rewards that measure question quality. Instead of using an annotation vector to tag the answer locations, (Song et al., 2017) proposed a unified framework for QG and question answering by encoding both the answer and the passage with a multi-perspective matching mechanism.

(Tang et al., 2017; Wang et al., 2017a) proposed joint models to address QG and question answering together. (Duan et al., 2017) conducted QG for improving question answering. Due to the mixed objectives including question answering, their approaches’ performance on QG were lower than the state-of-the-art results.

6 Conclusion and Future Work

In this paper, we proposed a new sequence to sequence network which contains a gated self-attention encoder and a maxout pointer decoder to address the answer-aware QG problem for long text input. We demonstrated the model can effectively utilize paragraph-level context, and outperformed the results with sentence-level context. The new model exceeded state-of-the-art approaches with either paragraph or sentence inputs.

We would like to discuss some potential challenges when applying the QG model in practice. 1) Answer spans aren't provided as input. One straight-forward method is to extract entities or noun phrases and use them as potential answer spans. A neural entity selection model can also be leveraged to extract good answer candidates to improve the precision as proposed in (Subramanian et al., 2017). 2) An input passage does not contain any eligible answers. In such case, we do not expect the model to output valid questions. We could remove questions with low generation probability, while a better approach could be running entity selection or quality detection model before question generation step to eliminate ineligible passages. 3) An answer could be shared by different questions. We could output multiple questions using beam search. However, beam search does not guarantee to output diversified candidates. We would need to explicitly model diversity among candidates during generation, for example, leveraging the approach described in (Li and Jurafsky, 2016).

Our future work lands in the following directions: incorporate rich features, such as POS and entity, in input passages; directly optimize sequence-level metrics with policy gradient; relax the constraint on answer to accept abstractive answers; jointly model question generation and question answering; ask multiple questions simultaneously with diverse perspectives.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Yichen Gong and Samuel R Bowman. 2017. Ruminating reader: Reasoning with gated multi-hop attention. *arXiv preprint arXiv:1704.07415*.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Michael Heilman and Noah A Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics.

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

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, page 889898. Association for Computational Linguistics.

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Stan Szpakowicz Marie-Francine Moens, editor, Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Karen Mazidi and Rodney D. Nielsen. 2014. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, page 321326. Association for Computational Linguistics.

Yurii Nesterov. 1983. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, volume 269, pages 543–547.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *Proceedings of 30th Conference on Neural Information Processing Systems (NIPS)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

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

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 251–257. Association for Computational Linguistics.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.

Heung-Yeung Shum, Xiaodong He, and Di Li. 2018. From eliza to xiaoice: challenges and opportunities with social chatbots. *arXiv preprint arXiv:1801.01957*.

Linfeng Song, Zhiguo Wang, and Wael Hamza. 2017. A unified auery-based generative model for question generation and question answering. *arXiv preprint arXiv:1709.01058*.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, and Adam Trischler. 2017. Neural models for key phrase detection and question generation. *arXiv preprint arXiv:1706.04560*.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Association for computation linguistics*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Neural Information Processing Systems*.

Tong Wang, Xingdi (Eric) Yuan, and Adam Trischler. 2017a. A joint model for question answering and question generation. In *Learning to generate natural language workshop, ICML 2017*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017b. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer.

Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task

Tao Yu   Rui Zhang   Kai Yang   Michihiro Yasunaga
Dongxu Wang   Zifan Li   James Ma   Irene Li
Qingning Yao   Shanelle Roman   Zilin Zhang   Dragomir R. Radev
Department of Computer Science, Yale University
`{tao.yu, r.zhang, k.yang, michihiro.yasunaga, dragomir.radev}@yale.edu`

Abstract

We present *Spider*, a large-scale complex and cross-domain semantic parsing and text-to-SQL dataset annotated by 11 college students. It consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains. We define a new complex and cross-domain semantic parsing and text-to-SQL task so that different complicated SQL queries and databases appear in train and test sets. In this way, the task requires the model to generalize well to both new SQL queries and new database schemas. Therefore, Spider is distinct from most of the previous semantic parsing tasks because they all use a single database and have the exact same program in the train set and the test set. We experiment with various state-of-the-art models and the best model achieves only 9.7% exact matching accuracy on a database split setting. This shows that Spider presents a strong challenge for future research. Our dataset and task with the most recent updates are publicly available at `https://yale-lily.github.io/seq2sql/spider`.

1 Introduction

Semantic parsing (SP) is one of the most important tasks in natural language processing (NLP). It requires both understanding the meaning of natural language sentences and mapping them to meaningful executable queries such as logical forms, SQL queries, and Python code.

Recently, some state-of-the-art methods with Seq2Seq architectures are able to achieve over 80% exact matching accuracy even on some complex benchmarks such as ATIS and GeoQuery. These models seem to have already solved most problems in this field.

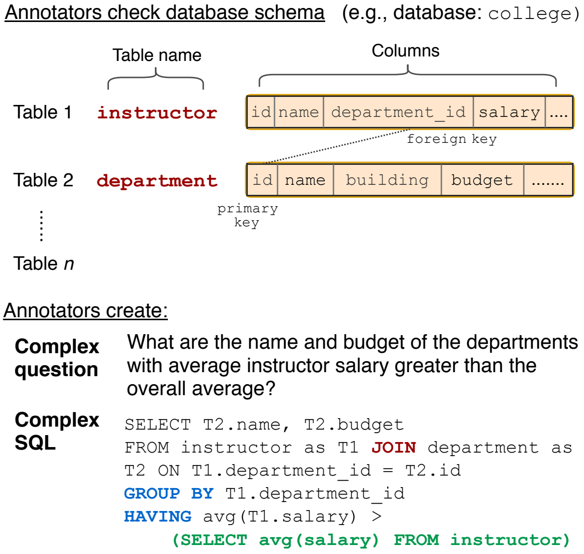

Complex question What are the name and budget of the departments with average instructor salary greater than the overall average?

Complex SQL

```
SELECT T2.name, T2.budget
FROM instructor as T1 JOIN department as
T2 ON T1.department_id = T2.id
GROUP BY T1.department_id
HAVING avg(T1.salary) >
    (SELECT avg(salary) FROM instructor)
```

Figure 1: Our corpus annotates complex questions and SQLs. The example contains joining of multiple tables, a `GROUP BY` component, and a nested query.

However, previous tasks in this field have a simple but problematic task definition because most of these results are predicted by semantic "matching" rather than semantic parsing. Existing datasets for SP have two shortcomings. First, those that have complex programs (Zelle and Mooney, 1996; Li and Jagadish, 2014; Yaghmazadeh et al., 2017a; Iyer et al., 2017) are too small in terms of number of programs for training modern data-intensive models and have only a single dataset, meaning that the same database is used for both training and testing the model. More importantly, the number of logic forms or SQL labels is small and each program has about 4-10 paraphrases of natural language problem to expand the size of the dataset. Therefore, the exact same target programs appear in both the train and test sets. The models can achieve decent performances even on very complex programs by memorizing the patterns of question and program pairs during training and decoding the programs exactly the same way as it saw in the training set during testing. Finegan-Dollak et al. (2018) split the dataset by programs

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3911–3921
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

so that no two identical program would be in both the train and test sets. They show that the models built on this question-splitting data setting fail to generalize to unseen programs. Second, existing datasets that are large in terms of the number of programs and databases such as WikiSQL (Zhong et al., 2017) contain only simple SQL queries and single tables. In order to test a model's real semantic parsing performance on unseen complex programs and its ability to generalize to new domains, an SP dataset that includes a large amount of complex programs and databases with multiple tables is a must.

However, compared to other large, realistic datasets such as ImageNet for object recognition (Deng et al., 2009) and SQuAD for reading comprehension (Rajpurkar et al., 2016), creating such SP dataset is even more time-consuming and challenging in some aspects due to the following reasons. First, it is hard to find many databases with multiple tables online. Second, given a database, annotators have to understand the complex database schema to create a set of questions such that their corresponding SQL queries cover all SQL patterns. Moreover, it is even more challenging to write different complex SQL queries. Additionally, reviewing and quality-checking of question and SQL pairs takes a significant amount of time. All of these processes require very specific knowledge in databases.

To address the need for a large and high-quality dataset for a new complex and cross-domain semantic parsing task, we introduce *Spider*, which consists of 200 databases with multiple tables, 10,181 questions, and 5,693 corresponding complex SQL queries, all written by 11 college students spending a total of 1,000 man-hours. As Figure 1 illustrates, given a database with multiple tables including foreign keys, our corpus creates and annotates complex questions and SQL queries including different SQL clauses such as joining and nested query. In order to generate the SQL query given the input question, models need to understand both the natural language question and relationships between tables and columns in the database schema.

In addition, we also propose a new task for text-to-SQL problem. Since Spider contains 200 databases with foreign keys, we can split the dataset with complex SQL queries in a way that no database overlaps in train and test, which overcomes the two shortcomings of prior datasets, and defines a new semantic parsing task in which the model needs to generalize not only to new programs but also to new databases. Models have to take questions and database schemas as inputs and predict unseen queries on new databases.

To assess the task difficulty, we experiment with several state-of-the-art semantic parsing models. All of them struggle on this task. The best model achieves only 9.7% exact matching accuracy in the database split setting. This suggests that there is a large room for improvement.

2 Related Work and Existing Datasets

Several semantic parsing datasets with different queries have been created. The output can be in many formats, e.g., logic forms. These datasets include ATIS (Price, 1990; Dahl et al., 1994), GeoQuery (Zelle and Mooney, 1996), and JOBS (Tang and Mooney, 2001a). They have been studied extensively (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Das et al., 2010; Liang et al., 2011; Banarescu et al., 2013; Artzi and Zettlemoyer, 2013; Reddy et al., 2014; Berant and Liang, 2014; Dong and Lapata, 2016). However, they are domain specific and there is no standard label guidance for multiple SQL queries.

Recently, more semantic parsing datasets using SQL as programs have been created. Iyer et al. (2017) and Popescu et al. (2003a) labeled SQL queries for ATIS and GeoQuery datasets. Other existing text-to-SQL datasets also include Restaurants (Tang and Mooney, 2001b; Popescu et al., 2003a), Scholar (Iyer et al., 2017), Academic (Li and Jagadish, 2014), Yelp and IMDB (Yaghmazadeh et al., 2017b), Advising (Finegan-Dollak et al., 2018), and WikiSQL (Zhong et al., 2017). These datasets have been studied for decades in both the NLP community (Warren and Pereira, 1982; Popescu et al., 2003b, 2004; Li et al., 2006; Giordani and Moschitti, 2012; Wang et al., 2017; Iyer et al., 2017; Zhong et al., 2017; Xu et al., 2017; Yu et al., 2018; Huang et al., 2018; Wang et al., 2018; Dong and Lapata, 2018; McCann et al., 2018) and the Database community (Li and Jagadish, 2014; Yaghmazadeh et al., 2017b). We provide detailed statistics on these datasets in Table 1.

Most of the previous work train their models without schemas as inputs because they use a single database for both training and testing. Thus, they do not need to generalize to new domains. Most importantly, these datasets have a limited number of labeled logic forms or SQL queries. In order to expand the size of these datasets and apply neural network approaches, each logic form or SQL query has about 4-10 paraphrases for the natural language input. Most previous studies follow the standard question-based train and test split (Zettlemoyer and Collins, 2005). This way, the exact same target queries (with similar paraphrases) in the test appear in training set as well. Utilizing this assumption, existing models can achieve decent performances even on complex programs by memorizing database-specific SQL templates. However, this accuracy is artificially inflated because the model merely needs to decide which template to use during testing. Finegan-Dollak et al. (2018) show that template-based approaches can get even higher results. To avoid getting this inflated result, Finegan-Dollak et al. (2018) propose a new, program-based splitting evaluation, where the exact same queries do not appear in both training and testing. They show that under this framework, the performance of all the current state-of-the-art semantic parsing systems drops dramatically even on the same database, indicating that these models fail to generalize to unseen queries. This indicates that current studies in semantic parsing have limitations.

We also want the model to generalize not only to unseen queries but also to unseen databases. Zhong et al. (2017) published the WikiSQL dataset. In their problem definition, the databases in the test set do not appear in the train or development sets. Also, the task needs to take different table schemas as inputs. Therefore, the model has to generalize to new databases. However, in order to generate about 90,000 questions and SQL pairs for about 26,000 databases, Zhong et al. (2017) made simplified assumptions about the SQL queries and databases. Their SQL labels only cover single `SELECT` column and aggregation, and `WHERE` conditions. Moreover, all the databases only contain single tables. No `JOIN`, `GROUP BY`, and `ORDER BY`, etc. are included.

Recently, researchers have constructed some datasets for code generation including IFTTT (Quirk et al., 2015), DJANGO (Oda et al., 2015), HEARTHSTONE (Ling et al., 2016), NL2Bash (Lin et al., 2018), and CoNaLa (Yin et al., 2018).

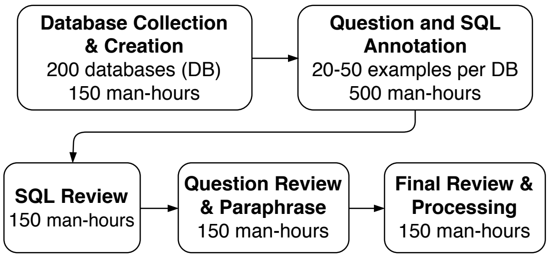

Figure 2: The annotation process of our Spider corpus.

These tasks parse natural language descriptions into a more general-purpose programming language such as Python (Allamanis et al., 2015; Ling et al., 2016; Rabinovich et al., 2017; Yin and Neubig, 2017).

3 Corpus Construction

All questions and SQL queries were written and reviewed by 11 computer science students who were native English speakers.As illustrated in Figure 2, we develop our dataset in five steps, spending around 1,000 hours of human labor in total: §3.1 Database Collection and Creation, §3.2 Question and SQL Annotation, §3.3 SQL Review, §3.4 Question Review and Paraphrase, §3.5 Final Question and SQL Review.

3.1 Database Collection and Creation

Collecting databases with complex schemas is hard. Although relational databases are widely used in industry and academia, most of them are not publicly available. Only a few databases with multiple tables are easily accessible online.

Our 200 databases covering 138 different domains are collected from three resources. First, we collected about 70 complex databases from different college database courses, SQL tutorial websites, online csv files, and textbook examples. Second, we collected about 40 databases from the DatabaseAnswers[1] where contains over 1,000 data models across different domains. These data models contain only database schemas. We converted them into SQLite, populated them using an online database population tool[2], and then manually corrected some important fields so that the table contents looked natural. Finally, we created the remaining 90 databases based on WikiSQL. To ensure the domain diversity, we select about 500 tables in about 90 different domains to create these 90 databases. To create each database,

---

[1] `http://www.databaseanswers.org/`

[2] `http://filldb.info/`

we chose several related tables from WikiSQL dev or test splits, and then created a relational database schema with foreign keys based on the tables we selected. We had to create some intersection tables in order to link several tables together. For most other cases, we did not need to populate these databases since tables in WikiSQL are from Wikipedia, which already had real world data stored.

We manually corrected some database schemas if they had some column names that did not make sense or missed some foreign keys. For table and column names, it is common to use abbreviations in databases. For example, 'student_id' might be represented by 'stu_id'. For our task definition, we manually changed each column name back to regular words so that the system only handled semantic parsing issues.

3.2 Question and SQL Annotation

For each database, we ask eight computer science students proficient in SQL to create 20-50 natural questions and their SQL labels. To make our questions diverse, natural, and reflective of how humans actually use databases, we did not use any template or script to generate question and SQL queries. Our annotation procedure ensures the following three aspects.

A) SQL pattern coverage. We ensure that our corpus contains enough examples for all common SQL patterns. For each database, we ask annotators to write SQL queries that cover all the following SQL components: `SELECT` with multiple columns and aggregations, `WHERE`, `GROUP BY`, `HAVING`, `ORDER BY`, `LIMIT`, `JOIN`, `INTERSECT`, `EXCEPT`, `UNION`, `NOT IN`, `OR`, `AND`, `EXISTS`, `LIKE` as well as nested queries. The annotators made sure that each table in the database appears in at least one query.

B) SQL consistency. Some questions have multiple acceptable SQL queries with the same result. However, giving totally different SQL labels to similar questions can hinder the training of semantic parsing models. To avoid this issue, we designed the annotation protocol so that all annotators choose the same SQL query pattern if multiple equivalent queries are possible. More detail is explained in our appendix.

C) Question clarity. We did not create questions that are (1) vague or too ambiguous, or (2) require knowledge outside the database to answer.

First, ambiguous questions refer to the questions that do not have enough clues to infer which columns to return and which conditions to consider. For example, we would not ask "What is the most popular class at University X?" because the definition of "popular" is not clear: it could mean the rating of the class or the number of students taking the course. Instead, we choose to ask "What is the name of the class which the largest number of students are taking at University X?". Here, "popular" refers to the size of student enrollment. Thus, the "student_enrollment" column can be used in condition to answer this question. We recognize that ambiguous questions appear in real-world natural language database interfaces.

We agree that future work needs to address this issue by having multi-turn interactions between the system and users for clarification. However, our main aim here is to develop a corpus to tackle the problem of handling complex queries and generalizing across databases, which no existing semantic parsing datasets could do. Moreover, the low performances of current state-of-the-art models already show that our task is challenging enough, without ambiguous questions. In addition, questions are required to contain the specific information to return. Otherwise, we don't know if class id is also acceptable in the previous case. Most of questions in the existing semantic parsing datasets are ambiguous. This is not a big problem if we use one single dataset because we have enough data domain specific examples to know which columns are default. However, it would be a serious problem in cross domain tasks since the default return values differ cross domain and people.

Second, humans sometimes ask questions that require common sense knowledge outside the given database. For instance, when people ask "Display the employee id for the employees who report to John", the correct SQL is

```
SELECT employee_id
FROM employees
WHERE manager_id = (
SELECT employee_id
FROM employees
WHERE first_name = 'John')
```

which requires the common knowledge that "X reports to Y" corresponds to an "employee-manager" relation. we do not include such questions and leave them as a future research direction.

Annotation tools We open each database on a web-based interface powered by the sqlite_web[3] tool. It allows the annotators to see the schema and content of each table, execute SQL queries, and check the returned results. This tool was extremely helpful for the annotators to write executable SQL queries that reflect the true meaning of the given questions and return correct answers.

3.3 SQL Review

Once the database is labeled with question-query pairs, we ask a different annotator to check if the questions are clear and contain enough information to answer the query. For a question with multiple possible SQL translations, the reviewers double check whether the SQL label is correctly chosen under our protocol. Finally, the reviewers check if all the SQL labels in the current database cover all the common SQL clauses.

3.4 Question Review and Paraphrase

After SQL labels are reviewed, native English speakers review and correct each question. They first check if the question is grammatically correct and natural. Next, they make sure that the question reflects the meaning of its corresponding SQL label. Finally, to improve the diversity in questions, we ask annotators to add a paraphrased version to some questions.

3.5 Final Review

Finally, we ask the most experienced annotator to conduct the final question and SQL review. This annotator makes the final decision if multiple reviewers are not sure about some annotation issues. Also, we run a script to execute and parse all SQL labels to make sure they are correct.

4 Dataset Statistics and Comparison

We summarize the statistics of Spider and other text-to-SQL datasets in Table 1. Compared with other datasets, Spider contains databases with multiple tables and contains SQL queries including many complex SQL components. For example, Spider contains about twice more nested queries and 10 times more `ORDER BY (LIMIT)` and `GROUP BY (HAVING)` components than the total of previous text-to-SQL datasets. Spider has 200 distinct databases covering 138 different domains such as college, club, TV show, government, etc. Most domains have one database, thus containing 20-50 questions, and a few domains such as flight information have multiple databases with more than 100 questions in total. On average, each database in Spider has 28 columns and 9 foreign keys. The average question length and SQL length are about 13 and 21 respectively. Our task uses different databases for training and testing, evaluating the cross-domain performance. Therefore, Spider is the *only one* text-to-SQL dataset that contains both databases with multiple tables in different domains and complex SQL queries It tests the ability of a system to generalize to not only new SQL queries and database schemas but also new domains.

5 Task Definition

On top of the proposed dataset, we define a text-to-SQL task that is more realistic than prior work. Unlike most of the previous semantic parsing or text-to-SQL tasks, models will be tested on *both different complex SQL queries and different complex databases in different domains* in our task. It aims to ensure that models can only make the correct prediction when they truly understand the semantic meaning of the questions, rather than just memorization. Also, because our databases contain different domains, our corpus tests model's ability to generalize to new databases. In this way, model performance on this task can reflect the real semantic parsing ability.

In order to make the task feasible and to focus on the more fundamental part of semantic parsing, we make the following assumptions:

- In our current task, we do not evaluate model performance on generating values. Predicting correct SQL structures and columns is more realistic and critical at this stage based on the low performances of various current state-of-the-art models on our task. In a real world situation, people need to double check what condition values are and finalize them after multiple times. It is unrealistic to predict condition values without interacting with users. In reality, most people know what values to ask but do not know the SQL logic. A more reasonable way is to ask users to use an interface searching the

---

[3] `https://github.com/coleifer/sqlite-web`

Dataset	# Q	# SQL	# DB	# Domain	# Table /DB	`ORDER BY`	`GROUP BY`	`NESTED`	`HAVING`
ATIS	5,280	947	1	1	32	0	5	315	0
GeoQuery	877	247	1	1	6	20	46	167	9
Scholar	817	193	1	1	7	75	100	7	20
Academic	196	185	1	1	15	23	40	7	18
IMDB	131	89	1	1	16	10	6	1	0
Yelp	128	110	1	1	7	18	21	0	4
Advising	3,898	208	1	1	10	15	9	22	0
Restaurants	378	378	1	1	3	0	0	4	0
WikiSQL	80,654	77,840	26,521	-	1	0	0	0	0
Spider	10,181	5,693	200	138	5.1	1335	1491	844	388

Table 1: Comparisons of text-to-SQL datasets. **Spider** is the *only one* text-to-SQL dataset that contains both databases with multiple tables in different domains and complex SQL queries. It was designed to test the ability of a system to generalize to not only new SQL queries and database schemas but also new domains.

values, then ask more specific questions. Also, other previous work with value prediction uses one single database in both train and test which makes it possible to overfit. However, in our task, we have different databases of different domains in train and test.

- As mentioned in the previous sections, we exclude some queries that require outside knowledge such as common sense inference and math calculation. For example, imagine a table with birth and death year columns. To answer the questions like "How long is X's life length?", we use `SELECT death_year - birth_year`. Even though this example is easy for humans, it requires some common knowledge of the life length definition and the use of a math operation, which is not the focus of our dataset.

- We assume all table and column names in the database are clear and self-contained. For example, some databases use database specific short-cut names for table and column names such as "stu_id", which we manually converted to "student id" in our corpus.

6 Evaluation Metrics

Our evaluation metrics include Component Matching, Exact Matching, and Execution Accuracy. In addition, we measure the system's accuracy as a function of the difficulty of a query. Since our task definition does not predict value string, our evaluation metrics do not take value strings into account.

We will release the official evaluation script along with our corpus so that the research community can share the same evaluation platform.

Component Matching To conduct a detailed analysis of model performance, we measure the average exact match between the prediction and ground truth on different SQL components. For each of the following components:

- `SELECT`
- `WHERE`
- `GROUP BY`
- `ORDER BY`
- `KEYWORDS` (including all SQL keywords without column names and operators)

we decompose each component in the prediction and the ground truth as bags of several subcomponents, and check whether or not these two sets of components match exactly. To evaluate each `SELECT` component, for example, consider `SELECT avg(col1), max(col2), min(col1)`, we first parse and decompose into a set `(avg, min, col1), (max, col2)`, and see if the gold and predicted sets are the same. Previous work directly compared decoded SQL with gold SQL. However, some SQL components do not have order constraints. In our evaluation, we treat each component as a set so that for example, `SELECT avg(col1), min(col1), max(col2)` and `SELECT avg(col1), max(col2), min(col1)` would be treated as the same query. To report a model's overall performance on each component, we compute F1 score on exact set matching.

Exact Matching We measure whether the predicted query as a whole is equivalent to the gold query. We first evaluate on the SQL clauses as described in the last section. The predicted query is correct only if all of the components are correct. Because we conduct set comparison in each clause, this exact matching metric can handle the "ordering issue" (Xu et al., 2017).

Execution Accuracy[4] Since Exact Matching can create false negative evaluation when the semantic parser generates novel and correct syntax structures, we also consider Execution Accuracy. All our databases have executable SQLite files, so we can measure execution accuracy as well. However, it is also important to note that Execution Accuracy can create false positive evaluation as a predicted SQL could return the same result (for example, 'NULL') as the gold SQL when they are semantically different. So we can use both to complement each other.

Finally, our evaluation also considers multiple acceptable keys if `JOIN` and `GROUP` are in the query. For example, suppose "stu_id" in one table refers to "stu_id" in another table, `GROUP BY` either is acceptable.

SQL Hardness Criteria To better understand the model performance on different queries, we divide SQL queries into 4 levels: easy, medium, hard, extra hard. We define the difficulty based on the number of SQL components, selections, and conditions, so that queries that contain more SQL keywords (`GROUP BY`, `ORDER BY`, `INTERSECT`, nested subqueries, column selections and aggregators, etc) are considered to be harder. For example, a query is considered as hard if it includes more than two `SELECT` columns, more than two `WHERE` conditions, and `GROUP BY` two columns, or contains `EXCEPT` or nested queries. A SQL with more additions on top of that is considered as extra hard. Figure 3 shows examples of SQL queries in 4 hardness levels.

Easy

What is the number of cars with more than 4 cylinders?

```
SELECT COUNT(*)
FROM cars_data
WHERE cylinders > 4
```

Meidum

For each stadium, how many concerts are there?

```
SELECT T2.name, COUNT(*)
FROM concert AS T1 JOIN stadium AS T2
ON T1.stadium_id = T2.stadium_id
GROUP BY T1.stadium_id
```

Hard

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name
FROM countries AS T1 JOIN continents
AS T2 ON T1.continent = T2.cont_id
JOIN car_makers AS T3 ON
T1.country_id = T3.country
WHERE T2.continent = 'Europe'
GROUP BY T1.country_name
HAVING COUNT(*) >= 3
```

Extra Hard

What is the average life expectancy in the countries where English is not the official language?

```
SELECT AVG(life_expectancy)
FROM country
WHERE name NOT IN
    (SELECT T1.name
     FROM country AS T1 JOIN
     country_language AS T2
     ON T1.code = T2.country_code
     WHERE T2.language = "English"
       AND T2.is_official = "T")
```

Figure 3: SQL query examples in 4 hardness levels.

7 Methods

In order to analyze the difficulty and demonstrate the purpose of our corpus, we experiment with several state-of-the-art semantic parsing models. As our dataset is fundamentally different from the prior datasets such as Geoquery and WikiSQL, we adapted these models to our task as follows. We created a 'big' column list by concatenating columns in all tables of the database together as a input to all models. Also, for each model, we limit the column selection space for each question example to all column of the database which the question is asking instead of all column names in the whole corpus.

Seq2Seq Inspired by neural machine translation (Sutskever et al., 2014), we first apply a basic sequence-to-sequence model, **Seq2Seq**. Then, we also explore **Seq2Seq+Attention** from (Dong and Lapata, 2016) by adding an attention mechanism (Bahdanau et al., 2015). In addition, we include **Seq2Seq+Copying** by adding an attention-based copying operation similar to (Jia and Liang, 2016).

The original model does not take the schema into account because it has the same schema in both train and test. We modify the model so that it

---

[4] We will provide the results in the later version. Please check our website for the latest updates on the task at https://yale-lily.github.io/seq2sql/spider

Method	Easy	Medium	Hard	Extra Hard	All
Example Split					
Seq2Seq	24.3%	9.5%	6.3%	1.5%	11.2%
Seq2Seq+Attention (Dong and Lapata, 2016)	31.2%	13.9%	11.6%	3.3%	15.5%
Seq2Seq+Copying	30.0%	12.6%	10.0%	3.3%	14.8%
Iyer et al. (2017)	18.6%	12.0%	9.0%	1.8%	9.8%
SQLNet (Xu et al., 2017)	36.2%	15.6%	7.9%	4.9%	17.4%
TypeSQL (Yu et al., 2018)	48.6%	38.2%	18.1%	19.8%	34.3%
Database Split					
Seq2Seq	17.9%	2.7%	1.3%	0.6%	5.4%
Seq2Seq+Attention (Dong and Lapata, 2016)	17.9%	2.9%	1.8%	1.3%	5.7%
Seq2Seq+Copying	15.1%	3.4%	1.0%	1.3%	5.2%
Iyer et al. (2017)	7.9%	2.1%	1.3%	1.1%	3.1%
SQLNet (Xu et al., 2017)	23.7%	5.9%	2.3%	0.3%	8.3%
TypeSQL (Yu et al., 2018)	29.6%	6.1%	2.3%	0.3%	9.7%

Table 2: Accuracy of Exact Matching on SQL queries with different hardness levels.

Method	`SELECT`	`WHERE`	`GROUP BY`	`ORDER BY`	`KEYWORDS`
Example Split					
Seq2Seq	25.8%	8.2%	18.9%	13.0%	21.8%
Seq2Seq+Attention	30.0%	11.4%	24.1%	19.3%	22.4%
Seq2Seq+Copying	28.7%	8.7%	24.2%	21.6%	19.7%
Iyer et al. (2017)	19.2%	8.1%	16.1%	9.6%	13.6%
SQLNet	46.1%	32.5%	30.6%	61.4%	77.6%
TypeSQL	68.7%	55.4%	40.5%	67.0%	73.6%
Database Split					
Seq2Seq	13.7%	3.7%	3.2%	4.9%	8.9%
Seq2Seq+Attention	14.0%	5.0%	3.2%	6.1%	9.4%
Seq2Seq+Copying	12.0%	2.7%	5.2%	6.9%	6.7%
Iyer et al. (2017)	6.3%	1.9%	3.0%	3.6%	3.5%
SQLNet	24.0%	18.0%	11.8%	47.1%	61.9%
TypeSQL	36.2%	14.7%	6.4%	49.5%	59.4%

Table 3: F1 scores of Component Matching on all SQL queries.

considers the table schema information by passing a vocabulary mask that limits the model to decode the words from SQL key words, table and column names in current database.

(Iyer et al., 2017) Iyer et al. (2017) apply an attention based sequence-to-sequence model similar to (Luong et al., 2015) to SQL datasets with automatic dataset expansion through paraphrasing and SQL templates. In addition, they show how user interactions improve results consistently. In our case, we did not consider the user interaction part.

SQLNet introduced by (Xu et al., 2017) uses column attention and employs a sketch-based method and generates SQL as a slot-filling task. This fundamentally avoids the sequence-to-sequence structure when ordering does not matter in SQL query conditions. Because it is originally designed for WikiSQL, we also extend its `SELECT` and `WHERE` modules to other components.

TypeSQL is the state-of-the-art model on the WikiSQL task (Yu et al., 2018). It improves upon SQLNet by proposing a different training procedure and utilizing types extracted from either knowledge graph or table content to help model better understand entities and numbers in the question. In our experiment, we use the question type info extracted from database content. Also, we extend their modules to other components.

8 Experimental Results and Discussion

We summarize the performance of all models on our test set including accuracy of exact matching in Table 2 and F1 scores of component matching in Table 3. For the final training dataset, we also select and include 752 queries and 1659 questions that follow our annotation protocol from six existing datasets: Restaurants, GeoQuery, Scholar, Academic, IMDB, and Yelp. We report results on two different settings for all models: (1) Example split where examples are randomly split into 7862 train, 1831 dev, 2147 test. Questions for the same database can appear in both train and test. (2) Database split where 206 databases are randomly split into 130 train, 36 dev, and 40 test. All questions for the same database are in the same split.

Overall Performance The performances of the Seq2Seq-based models including Seq2Seq, Seq2Seq+Attention, Seq2Seq+Copying, and Iyer et al. (2017) are very low. However, they are able to generate nested and complex queries. Thus, they can get a few hard and extra hard examples correct. But in the vast majority of cases, they predict invalid SQL queries with grammatical errors. The attention and copying mechanisms do not help much either. In contrast, SQLNet and TypeSQL that utilize SQL structure information to guide the SQL generation process significantly outperform other Seq2Seq model. While they can produce valid queries, however, they are unable to generate nested queries or queries with keywords such as `EXCEPT` and `INTERSECT`.

In general, the overall performances of all models are low, indicating that our task is challenging and there is still a large room for improvement.

Example Split vs Database Split As discussed in Section 5, another challenge of the dataset is to generalize to new databases. To study this, in Table 2 and Table 3 we compare model performances under the two settings. For all models, the performance under database split is much lower than that under example split. In addition, we observe that all models perform poorly on column selection. This shows that our dataset presents a challenge for the model to generalize to new databases.

Complexity of Database Schema In order to show how the complexity of the database schema affects model performance, Figure 4 plots the exact matching accuracy as a function of the number of foreign keys in a database. The performance decreases as the database has more foreign keys. The first reason is because the model has to choose column and table names from many candidates in a complex database schema. Second, a complex database schema presents a great challenge for the model to capture the relationship between different tables with foreign keys. It indicates that this task requires more effective methods to encode the relation of tables with foreign keys.

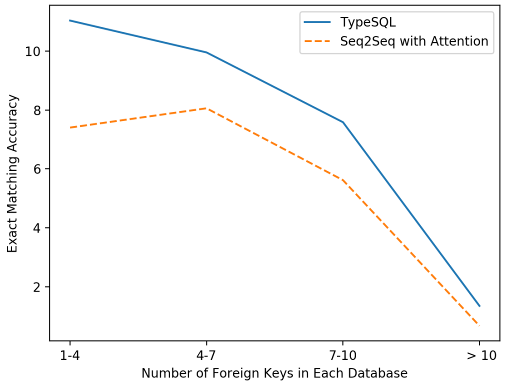

Figure 4: Exact matching accuracy as a function of the number of foreign keys.

9 Conclusion

In this paper we introduce Spider, a large, complex and cross-domain semantic parsing and text-to-SQL dataset, which directly benefits both NLP and DB communities. Based on Spider, we define a new challenging and realistic semantic parsing task. Experimental results on several state-of-the-art models on this task suggests plenty space of improvement.

Acknowledgement

We thank Graham Neubig, Tianze Shi, Catherine Finegan-Dollak, and three anonymous reviewers for their discussion and feedback. We also thank Barry Williams for providing datasets from the DatabaseAnswers.

References

Miltiadis Allamanis, Daniel Tarlow, Andrew D. Gordon, and Yi Wei. 2015. Bimodal modelling of source code and natural language. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2123–2132. JMLR.org.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association forComputational Linguistics*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic frame-semantic parsing. In *NAACL*.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan Dhanalakshmi Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *ACL 2018*. Association for Computational Linguistics.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to sql queries with generative parsers discriminatively reranked. In *COLING (Posters)*, pages 401–410.

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen tau Yih, and Xiaodong He. 2018. Natural language to structured query generation via meta-learning. In *NAACL*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *CoRR*, abs/1704.08760.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *VLDB*.

Yunyao Li, Huahai Yang, and HV Jagadish. 2006. Constructing a generic natural language interface for an xml database. In *EDBT*, volume 3896, pages 737–754. Springer.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. 2018. Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. In *LREC*.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *ACL (1)*. The Association for Computer Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (t). In *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, ASE '15.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *Proceedings of the 20th international conference on Computational Linguistics*, page 141. Association for Computational Linguistics.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003a. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003b. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM.

P. J. Price. 1990. Evaluation of spoken language systems: the atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden*

Valley, Pennsylvania, June 24-27,1990, pages 91–95.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 878–888.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *ACL (1)*, pages 1139–1149. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Lappoon R. Tang and Raymond J. Mooney. 2001a. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, pages 466–477, Freiburg, Germany.

Lappoon R Tang and Raymond J Mooney. 2001b. Using multiple clause constructors in inductive logic programming for semantic parsing. In *ECML*, volume 1, pages 466–477. Springer.

Chenglong Wang, Alvin Cheung, and Rastislav Bodik. 2017. Synthesizing highly expressive sql queries from input-output examples. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 452–466. ACM.

Chenglong Wang, Po-Sen Huang, Alex Polozov, Marc Brockschmidt, and Rishabh Singh. 2018. Execution-guided neural program decoding. In *ICML workshop on Neural Abstract Machines and Program Induction v2 (NAMPI)*.

David HD Warren and Fernando CN Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics*, 8(3-4):110–122.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-2007)*, Prague, Czech Republic.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017a. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017b. Sqlizer: Query synthesis from natural language. *Proc. ACM Program. Lang.*, 1(OOPSLA):63:1–63:26.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to mine aligned code and natural language pairs from stack overflow. In *International Conference on Mining Software Repositories (MSR)*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *ACL (1)*, pages 440–450. Association for Computational Linguistics.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of NAACL*. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR. AAAI Press/MIT Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Unsupervised Natural Language Generation with Denoising Autoencoders

Markus Freitag and **Scott Roy**
Google AI
`{freitag,hsr}@google.com`

Abstract

Generating text from structured data is important for various tasks such as question answering and dialog systems. We show that in at least one domain, without any supervision and only based on unlabeled text, we are able to build a Natural Language Generation (NLG) system with higher performance than supervised approaches. In our approach, we interpret the structured data as a corrupt representation of the desired output and use a denoising auto-encoder to reconstruct the sentence. We show how to introduce noise into training examples that do not contain structured data, and that the resulting denoising auto-encoder generalizes to generate correct sentences when given structured data.

1 Introduction

Natural Language Generation (NLG) is the task of generating text from structured data. Recent success in Deep Learning motivated researchers to use neural networks instead of human designed rules and templates to generate meaningful sentences from structured information. However, these supervised models work well only when provided either massive amounts of labeled data or when restricted to a limited domain. Unfortunately, labeled examples are costly to obtain and are non-existent for many domains. Conversely, large amount of unlabeled data are often freely available in many languages.

A labeled example is given in Table 1. One labeled example consists of a set of slot pairs and at least one golden target sequence. Each slot pair has a slot name (e.g. ”name”) and a slot value (e.g. ”Loch Fyne”). In this work, we present an unsupervised NLG approach that learns its parameters without the slot pairs on target sequences only. We use the approach of a denoising auto-encoder (DAE) (Vincent et al., 2008) to train our model. During training, we use corrupt versions of each target sequence as input and learn to reconstruct the correct sequence using a sequence-to-sequence network (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). We show how to introduce noise into the training data in such a way that the resulting DAE is capable of generating sentences out of a set of slot pairs. Taking advantage of using unlabeled data only, we also incorporate out-of-domain data into the training process to improve the quality of the generated text.

name	type	food	family friendly
Loch Fyne	restaurant	Indian	yes

Table 1: Three possible correct target sequences for the structured data above: (a) There is an Indian restaurant that is kids friendly. It is Loch Fyne. (b) Loch Fyne is a well-received restaurant with a wide range of delicious Indian food. It also delivers a fantastic service to young children. (c) Loch Fyne is a family friendly restaurant providing Indian food.

2 Network

In all our experiments, we use our in-house attention-based sequence-to-sequence (seq2seq) implementation which is similar to Bahdanau et al. (2015). The approach is based on an encoder-decoder network. The encoder employs a bi-directional RNN to encode the input words $\mathbf{x} = (x_1, ..., x_l)$ into a sequence of hidden states $\mathbf{h} = (h_1, ..., h_l)$, where l is the length of the input sequence. Each h_i is a concatenation of a left-to-right $\overrightarrow{h_i}$ and a right-to-left $\overleftarrow{h_i}$ RNN:

$$h_i = \begin{bmatrix} \overleftarrow{h}_i \\ \overrightarrow{h}_i \end{bmatrix} = \begin{bmatrix} \overleftarrow{f}(x_i, \overleftarrow{h}_{i+1}) \\ \overrightarrow{f}(x_i, \overrightarrow{h}_{i-1}) \end{bmatrix}$$

where $\overleftarrow{f}$ and $\overrightarrow{f}$ are two gated recurrent units (GRU) proposed by Cho et al. (2014).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3922–3929
Brussels, Belgium, October 31 - November 4, 2018.

Given the encoded $\mathbf{h}$, the decoder predicts the target sentence by maximizing the conditional log-probability of the correct output $\mathbf{y}^* = (y_1^*, ...y_m^*)$, where m is the length of the target. At each time t, the probability of each word y_t from a target vocabulary V_y is:

$$p(y_t|\mathbf{h}, y_{t-1}^*..y_1^*) = g(s_t, y_{t-1}^*, H_t), \qquad (1)$$

where g is a two layer feed-forward neural network over the embedding of the previous target word y_{t-1}^*, the hidden state s_t, and the weighted sum of $\mathbf{h}$ (H_t).

Before we compute s_t and H_t, we first covert s_{t-1} and the embedding of y_{t-1}^* into an intermediate state s_t' with a GRU u as:

$$s_t' = u(s_{t-1}, y_{t-1}^*). \qquad (2)$$

Then we have s_t as:

$$s_t = q(s_t', H_t) \qquad (3)$$

where q is a GRU, and the H_t is computed as:

$$H_t = \begin{bmatrix} \sum_{i=1}^{l} (\alpha_{t,i} \cdot \overleftarrow{h}_i) \\ \sum_{i=1}^{l} (\alpha_{t,i} \cdot \overrightarrow{h}_i) \end{bmatrix}, \qquad (4)$$

The attention weights, α in H_t, are computed with a two layer feed-forward neural network r:

$$\alpha_{t,i} = \frac{\exp\{r(s_t', h_i)\}}{\sum_{j=1}^{l} \exp\{r(s_t', h_j)\}} \qquad (5)$$

3 Unsupervised Approach

Our unsupervised model is based on the same training idea as a denoising auto-encoder (DAE) similar to Vincent et al. (2008). The original DAEs were feedforward nets applied to (image) data. In our experiments, the model architecture is a seq2seq model similar to Bahdanau et al. (2015). The idea of a DAE is to train a model that is able to reconstruct each training example from a partially destroyed input. This is done by first corrupting each training sequence x_i to get a partially destroyed version $\tilde{x_i}$.

In our unsupervised experiments, we generate the training data with the following corrupting process, parameterized by the desired percentage p of deletion: for each target sequence x_i, a fixed percentage p of words are removed at random, while the others are left untouched. We sample a new corrupt version $\tilde{x_i}$ in each training epoch. Instead of always removing a fixed percentage of words, we sample p for each sequence separately from a Gaussian distribution with mean $p = 0.6$ and variance 0.1. We chose $p = 0.6$ based on the average length ratio between the slot values and the target sequences in our labeled training data.

This corruption approach is motivated by the fact that many NLG problems are facing a similar task to the one the DAE is solving. Given some structured information, the task is to generate a target sequence that includes all the information. If we map the structured information to phrases that should be in the desired output, then the structured data problem resembles the DAE problem. For instance, if we have the following structured example: *name: Aromi - family friendly: yes → Aromi has a family friendly atmosphere.* , we convert it into the input *Aromi family friendly* that we can feed to the DAE. To preprocess the structured data, we convert the boolean feature *family friendly* into a meaningful phrase (”family friendly”) by using the slot name. For all non boolean slot pairs, we just use the slot values as meaningful phrases. Please keep in mind this transformation is only needed during inference as the training data has no slot pairs and only consists of pairs of corrupt and correct target sequences.

Nevertheless, there are two major differences between the training procedure of a DAE and an inference instance in NLG: First, we do not need to predict any content information in NLG as all of the content information is already provided by the structured data. On the other hand, a DAE training instance can also remove content words from the sentence. To align the two much closer, we restrict the words which the DAE is allowed to remove and apply the following heuristic to the corruption process of the DAE: Given the absolute counts $N(v_i)$ for each word v_i in our vocabulary, we only allow v_i to be removed when its count $N(v_i)$ is larger than a threshold. This heuristic is motivated by the fact that the corpus frequency of content words like a restaurant name is most likely low and the corpus frequency of non-content words like ”the” is most likely high. The corpus frequencies can be either calculated on the training data itself or on a different corpora. The latter one has the advantage that domain specific content words that are frequent in the training data will have a low frequency in an out-of-domain corpora.

The second difference is that in a DAE training

	original	Loch Fyne is a family friendly restaurant providing Indian food .
(a)	remove random 60%	Fyne is restaurant food .
(b)	remove only words w_i with $N(w_i) > 100$	Loch Fyne family friendly Indian
(c)	shuffle words	family friendly Indian Loch Fyne

Table 2: Training data generation heuristics. (a): random 60% of the words are removed. (b): 60% of the words are removed, but only words that occur more than 100 times in the training data. Our assumption is that these are the non-content words. (c): On top of (b), all words are shuffled while keeping all word pairs (e.g. Loch Fyne) together that also occur in the original sentence.

instance, the words in a corrupt input occur in the same order as in the desired target. For an NLG inference instance, the order of the structured input does not need to match the order of the words in the output. To overcome this issue, we shuffle the words within the corrupt sentence while not splitting bigrams that also exist in the original sentence. An example of all three heuristics is given In Table 2.

4 Supervised Approach

For comparison, we train a supervised baseline based on the vanilla seq2seq model as described in Section 2. To make better use of the structured data, we found that the input word embeddings (wemb) of the seq2seq network should be represented together by the slot name and value. We split the word embedding vector into two parts and use the upper half for a word embedding of the slot name and the lower half for the word embedding of the slot value. If a slot value has multiple words, we build separate word embeddings for each word, but all having the same upper part (slot name). An example for the slot pairs of Table 1 is given in Figure 1.

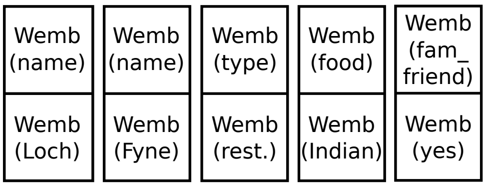

Figure 1: Example input word embeddings for our supervised baseline (Section 4) from the training Example of Table 1. The upper half of the word embedding is used for the slot names; the lower half for the slot values.

5 Data Sets

The **E2E data set** (Novikova et al., 2017) contains reviews in the restaurant domain. Given up to 8 different pieces of information about a restaurant, the task is to generate an English sentence that includes all of the provided structured information. The dataset comes with 42061 training examples. We split the provided dev set (547 examples) into validation (tail 268 examples) and test (head 279 examples), each having between 3 and 42 (on average 8) reference sentences. An example of an E2E training instance is given in Table 1.

The **news-commentary** data set is a parallel corpus of news provided by the WMT conference (Bojar et al., 2017) for training machine translation (MT) systems. For our unsupervised experiments, we use the English **news-commentary** part of the corpora only which contains 256,715 sentences.

All corpora are tokenized and we remove sentences that are longer than 60 tokens. In addition to tokenization, we also apply byte-pair encoding (Sennrich et al., 2016) when **news-commentary** is included in the setup.

dataset	task	# examples
E2E	NLG	42,061
news-commentary	MT (De-En)	256,715

Table 3: Training data statistics. Each training instance in NLG contains structured information and one reference sequence. Each training instance in MT contains one sentence written in both German and English.

6 Experiments

6.1 Model Parameters

For all of our experiments we utilize the seq2seq implementation as described in Section 2. We run inference with a beam size of 5. We use a hidden layer size of 1024 and a word embedding size of 620 and use SGD with an initial learning rate of 0.5. We halve the learning rate every other epoch starting from the 5th epoch. We evaluate the generated text with BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004), and NIST (Doddington, 2002) and use the evaluation tool provided by the E2E organizers to calculate the scores.

6.2 Automatic Scores

Our experimental results are summarized in Table 4. We list two supervised baselines: The first one is from the organizers of the E2E challenge, the second one is from our supervised setup (Section 4). Our baseline yields better performance on BLEU and ROUGE-L while reaching similar performance in NIST. Our third (unsupervised) *baseline copy input* just runs the evaluation metrics on the input (slot values of the structured data). This system performs much worse, but serves as a lower bound for our unsupervised experiments.

We report results on different unsupervised setups as described in Section 3. The system *randomly drop* just randomly drops 60% of the words, but still yields 57.3 BLEU, 65.9 ROUGE-L and 7.3 NIST points. You can easily detect a lot of extra information in the output that can not be explained by the structured input. Further, the output sounds very machine generated as the output depends on the order of the structured data. The heuristics + *only words w/ count >100 (on ind data)* and + *only words w/ count >100 (on ood data)* forbid removing words in the corruption phase that appear less than 100 times in the in-domain data or out-of-domain (ood) data, respectively. The latter setup uses the out-of-domain data for generating the word counts only and yields an improvement of 5.2 BLEU, 1.3 ROUGE-L and 0.2 NIST points compared to just randomly dropping words. The output still sounds very machine generated, but stops hallucinating additional information. We further improve the performance by 3.5 BLEU, 3.9 ROUGE-L, and 0.2 NIST points when shuffling the words in the corrupted input and using the out-of-domain data also as training examples. We use the English side of the 256,715 sentences from the news-commentary dataset as out-of-domain data only. We did not see any further improvements by adding more out-of-domain training data.

Finally, we build a semi supervised system that in addition to the unlabeled data includes the labeled information for some of the training examples. For these, we remove the slot names from the structured data and use a concatenation of all slot values as input to learn the correct output. By jointly using both unlabeled and labeled data, we yield an additional improvement of 1.0 BLEU points compared to our best fully unsupervised system. In our semi supervised setup, we only use the slot values as input even for the labeled examples. This explains the drop in performance when comparing to the supervised setups. All supervised setups also include the slot names in their input representation.

6.3 Human Evaluation

In addition to automatic scores, we ran human assessment of the generated text as none of the automatic metrics correlates well with human judgment (Belz and Reiter, 2006).

To collect human rankings, we presented 3 outputs generated by 3 different systems side-by-side to crowd-workers, who were asked to score each sentence on a 6-point scale for:

- **fluency**: How do you judge the overall naturalness of the utterance in terms of its grammatical correctness and fluency?

For the next questions, we presented in addition to the 3 different system outputs, the structured representations of each example. We asked the crowd-worker to score the following two questions on a 5-point scale:

- **all information**: How much of the given information is mentioned in the text?
- **bad/ false information**: How much false or extra information is mentioned in the text?

Each task has been given to three different raters. Consequently, each output has a separate score for each question that is the average of 3 different ratings. The human evaluation results are summarized in Table 5. We included the two supervised baselines and our best unsupervised setup in the human evaluation. The unsupervised setup outperforms the supervised setups in fluency. One explanation is that our unsupervised system includes additional unlabeled data that can not be included in a supervised setup. Due to our unsupervised learning approach that all words in the structured data need to be included in the final output, the unsupervised system did not miss any information. Further, all three outputs included little false or wrong information that was not included in the structured data. All in all the output of the

setup	model	BLEU	ROUGE-L	NIST
supervised	baseline E2E challenge (Dušek and Jurcıcek, 2016)	70.2	72.4	8.3
	baseline vanilla seq2seq (Section 4)	72.7	75.1	8.3
unsupervised	baseline copy input	27.7	56.4	3.2
	randomly drop	57.3	65.9	7.3
	+ only words w/ count $>$100 (on ind data)	59.5	66.4	7.3
	+ only words w/ count $>$100 (on ood data)	62.5	67.2	7.5
	+ shuffle pos	64.8	69.4	7.6
	+ ood data	66.0	71.1	7.7
semi-supervised	+ labeled ind data	67.0	72.1	7.8

Table 4: Results on the E2E dataset. The terms ood and ind are abbreviations for out-of-domain and in-domain respectively. The baseline systems as well as the unsupervised systems that are not labeled with + *ood data* are trained on the E2E in-domain training data only.

system	fluency	all information	extra/ false information
baseline E2E challenge (Dušek and Jurcıcek, 2016)	4.01	4.89	0.05
baseline vanilla seq2seq (Section 4)	4.46	4.91	0.08
unsupervised (random drop + words w/ count $>$100 (ood data) + shuffle pos + ood data)	4.70†	5.00†	0.05

Table 5: Human evaluation results: We generated 279 output sequences for each of the 3 listed systems. Each sequence has been evaluated by 3 different raters and the score is the average of 837 ratings per system. For each task and sequence, the raters where asked to give a score between 0 and 5. A score of 5 for *fluency* means that the text is fluent and grammatical correct. A score of 5 for *all information* means that all information from the structured data is mentioned. A score of 0 for *extra/ false information* means that no information besides the structured data is mentioned in the sequence. Scores labeled with † are significant better than all other systems ($p < 0.0001$).

unsupervised system is better than the two supervised systems.

We used approximate randomization (AR) as our significance test, as recommended by (Riezler and Maxwell, 2005). Pairwise tests between results in Table 5 showed that our novel unsupervised approach is significantly better than both baselines regarding fluency and mentioning all information with the likelihood of incorrectly rejecting the null hypothesis of $p < 0.0001$.

7 Limitations

Our unsupervised approach has two limitations and is therefore not easily applicable to all NLG problems or datasets. First, we can only run our approach for datasets where the input meaning representation either overlaps with target texts or we need to generate rules that map the structured data to target words. Unfortunately, the needed pattern can be very complicated and the effort of writing rules can be similar to the one of building a template based system.

Second, to be able to generate text from structured data during inference, the original structured input is converted to an unstructured one by discarding the slot names. This can be problematic in scenarios where the slot name itself contributes to the meaning representation, but the slot name should not be in the target text. For instances the structured data of a WEBNLG (Gardent et al., 2017) training example consists of several subject-predicate-object tuple features. Many of the features for one example have the same subject, but different predicates and objects. But yet in the final output, we prefer to have the subject only once.

8 Related Work

8.1 Neural Language Generation

Due to the recent success in Deep Learning, researchers started to use end-to-end systems to jointly model the traditional separated tasks of content selection, sentence planning and surface realization in one system. Recently, RNNs (Wen et al., 2015b), attention-based methods (Mei et al.,

2016) or LSTMs (Wen et al., 2015a) were successfully applied for the task of NLG. Liu et al. (2018) introduced a modified LSTM that adds a field gate into the LSTM to incorporate the structured data. Further, they used a dual attention mechanism that combines attention of both the slot names and the actual slot content. Sha et al. (2017) extended this approach and integrated a linked matrix in their model that learns the desired order of the slots in the target text. Further, Dušek and Jurcıcek (2016) reranked the n-best output from a seq2seq model to penalize sentences that miss required information or add irrelevant ones. Instead of RNNs, Lebret et al. (2016) introduced a neural feed-forward language model conditioned on both the full structured data and the structured information of the previous generated words. In addition, the authors introduced a copy mechanism for boosting the words given by the structured data.

In contrast to the above mentioned related work, we train our model in a fully unsupervised fashion. Although, all our experiments have been conducted with the seq2seq model, our unsupervised approach can be applied on top of all of the different network architectures that are introduced by the above mentioned papers.

8.2 DAE and Unsupervised Learning

Denoising auto-encoders and unsupervised training have been applied to various other NLP tasks. Vincent et al. (2008) introduced denoising one-layer auto-encoders that are optimized to reconstruct input data from random corruption. The outputs of the intermediate layers of these denoisers are then used as input features for subsequent learning tasks such as supervised classification (Lee et al., 2009; Glorot et al., 2011). They showed that transforming data into DAE representations (as a pre-training or initialization step) gives more robust (supervised) classification performance. Lample et al. (2018) used a denoising auto-encoder to build an unsupervised Machine Translation model. Hill et al. (2016) trained a denoising auto-encoder on a seq2seq network architecture for training sentence and paragraph representations from the output of the intermediate layers. They showed that using noise in the encoder step is helpful to learn a better sentence representation.

In contrast to the above mentioned related work, we train a DAE directly on a task and do not take the intermediate hidden states of a DAE as sentence representation to help learning a different task. Further, none of the related work applied DAEs on the task of generating sentences out of structured data. In addition, we modify the original DAE corruption process by introducing heuristics that remove non-content words only to match the input representation of a supervised NLG training instance.

9 Conclusion

We showed how to train a denoising auto-encoder that is able to generate correct English sentences from structured data. By applying several heuristics to the corruption phase of the auto-encoder, we reach better performance compared to two fully supervised systems. As no labeled data is needed in our approach, we further successfully improve the quality by incorporating out-of-domain data into the training phase. We run a human evaluation for the two supervised baselines and our best unsupervised setup. We see that the output of our unsupervised setup not only includes 100% of the structured information, but also outperforms both supervised baselines in terms of fluency and grammatical correctness.

The unsupervised training scheme gives us the option to incorporate any unlabeled data. One possible addition to our approach would be to incorporate text in different languages into our system, so that we can generate the output in any language from the same structured data.

Our approach is appropriate only for NLG problems where the goal is to include all the information from the structured data in the output. In future work, we will focus on the semi-supervised approach to make the DAE also suitable for problems where instead of all, only a subset of the structured information should be included in the output.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations (ICLR)*.

Anja Belz and Ehud Reiter. 2006. Comparing Automatic and Human Evaluation of NLG Systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. 2017. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR*, abs/1409.1259.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research (HLT)*, pages 138–145. Morgan Kaufmann Publishers Inc.

Ondrej Dušek and Filip Jurcıcek. 2016. Sequence-to-Sequence Generation for Spoken Dialogue via Deep Syntax Trees and Strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 45–51.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating Training Corpora for Micro-Planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 513–520.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1367–1377.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised Machine Translation Using Monolingual Corpora Only. In *International Conference on Learning Representations (ICLR)*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1203–1213, Austin, Texas.

Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. 2009. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1096–1104.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text Generation by Structure-aware Seq2seq Learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 720–730.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E Dataset: New Challenges For End-to-End Generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*, pages 201–206.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Stefan Riezler and John T Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for mt. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 57–64.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1715–1725.

L. Sha, L. Mou, T. Liu, P. Poupart, S. Li, B. Chang, and Z. Sui. 2017. Order-Planning Neural Text Generation From Structured Data. *ArXiv e-prints*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103. ACM.

Tsung-Hsien Wen, Milica Gašic, Dongho Kim, Nikola Mrkšic, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 275.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1711–1721.

Answer-focused and Position-aware Neural Question Generation

Xingwu Sun[1], **Jing Liu**[1], **Yajuan Lyu**[1], **Wei He**[1], **Yanjun Ma**[1], **Shi Wang**[2]
[1]Baidu Inc., Beijing, China
[2]Institute of Computing Technology Chinese Academy of Sciences, Beijing, China
`{sunxingwu,liujing46,lvyajuan,hewei06,mayanjun02}@baidu.com`
`wangshi@ict.ac.cn`

Abstract

In this paper, we focus on the problem of question generation (QG). Recent neural network-based approaches employ the sequence-to-sequence model which takes an answer and its context as input and generates a relevant question as output. However, we observe two major issues with these approaches: (1) The generated interrogative words (or question words) do not match the answer type. (2) The model copies the context words that are far from and irrelevant to the answer, instead of the words that are close and relevant to the answer. To address these two issues, we propose an answer-focused and position-aware neural question generation model. (1) By answer-focused, we mean that we explicitly model question word generation by incorporating the answer embedding, which can help generate an interrogative word matching the answer type. (2) By position-aware, we mean that we model the relative distance between the context words and the answer. Hence the model can be aware of the position of the context words when copying them to generate a question. We conduct extensive experiments to examine the effectiveness of our model. The experimental results show that our model significantly improves the baseline and outperforms the state-of-the-art system.

1 Introduction

The task of question generation (QG) aims to generate questions for a given text, and it can benefit several real applications: (1) In the area of education, QG can help generate questions for reading comprehension materials (Du et al., 2017). (2) QG can enable the machine to actively ask questions in a dialogue system. (3) QG can also aid in the development of question answering datasets (Duan et al., 2017). Typically, QG includes two subtasks: (1) what to say, i.e. determining the targets (e.g. sentences, phrases or words) that should be asked; (2) how to say, i.e. producing the surface-form of the question. In this paper, we focus on the sub-task of surface-form realization of questions by assuming the targets are given.

Previous work of QG can be classified into two categories: rule-based and neural network-based. Compared to the rule-based approach, the neural network-based approach does not rely on hand-crafted rules, and it is instead data-driven and trainable in an end-to-end fashion. The recent release of large-scale machine reading comprehension datasets, e.g. SQuAD (Rajpurkar et al., 2016) and MARCO (Nguyen et al., 2016), also drives the development of neural question generation.

Recent neural question generation approaches (Du et al., 2017; Zhou et al., 2017) employ sequence-to-sequence model that takes an answer and its context as input and outputs a relevant question. Zhou et al. (2017) further enrich the sequence-to-sequence model with rich features (e.g. answer position and lexical features) to generate answer focused questions, and incorporate copy mechanism that allows to copy words from the context when generating questions. To the best of our knowledge, it achieve the best results on SQuAD dataset so far (Zhou et al., 2017). In this paper, we implement this approach and carefully study its generation results. Specifically, we randomly sample 130 questions generated by the approach, and manually judge their quality by comparing them with the references. We find 54 out of 130 questions are ill generated, and we observe two major issues with the 54 questions: (1) 20 (37.04% errors) questions contain the question words that do not match the answer type, though the answer position feature has been incorporated. Because the model does not pay much attention to the answer that is a key to question word generation. Table 1 gives an example. A when-question

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3930–3939
Brussels, Belgium, October 31 - November 4, 2018.

Context: The tax collector who arrested him rose to higher political office , and Thoreau 's essay was not published until after **the end of the Mexican War**.
Answer: the end of the Mexican War
Question generated by the baseline: **Why** was Thoreau's essay published ?
Reference: **When** was Thoreau's essay published ?

Table 1: A bad case where the generated question word does not match the answer type. A when-question should be triggered for answer "the end of the Mexican War", while a why-question is generated by the baseline.

Context: This mechanism is still the **leading theory** today; however, a **second theory** suggests that most cpdna is actually linear and replicates through **homologous recombination**.
Answer: homologous recombination
Question generated by the baseline: What is the **leading theory** today ?
Reference: How does the **second theory** say most cpdna replicates ?

Table 2: A bad case where the model copies the context words far away from and irrelevant to the answer. The baseline copies "leading theory" that is far away from and unrelated to the answer "homologous recombination", but neglects the phrase "second theory" that is close and relevant to the answer.

should be triggered for answer "the end of the Mexican War" while a why-question is generated by the model. (2) 11 (20.37% errors) questions copies the context words that are far from and irrelevant to the answer, instead of the words that are close and relevant to the answer. Because the model is not aware of the positions of context words. Table 2 gives an example. The baseline model copies "leading theory" that is far away from and unrelated to the answer "homologous recombination", but neglects the phrase "second theory" that is close and relevant to the answer.

To address these two issues, we propose an answer-focused and position-aware neural question generation model. (1) By answer-focused, we mean that we explicitly model question word generation by incorporating the answer embedding, which can help generate a question word matching the answer type. (2) By position-aware, we mean that we model the relative distance between the context words and the answer. The relative distance is encoded as position embedding, on which a position-aware attention is generated. The position-aware attention help the model copy the context words that are relatively close and relevant to the answer. We further conduct extensive experiments on SQuAD and MARCO dataset to examine the effectiveness of the answer-focused model and position-aware model, respectively. The experimental results show that the combination of our proposed answer-focused model and position-aware model significantly improves the baseline and outperforms the state-of-the-art system.

The contributions of this paper can be summarized as follows:

- We analyze the generation results by the state-of-the-art neural model, and find two major issues with the model: (a) the generated question words do not match the answer type; (b) the model copies the context words that are far from and irrelevant to the answer.
- To deal with these two issues, we propose an answer-focused and position-aware neural question generation model.
- We conduct extensive experiments to examine the effectiveness of our proposed model.

2 Related Work

Question Generation Previous work of QG can be classified into two categories: rule-based and neural network-based. Regardless of the approach taken, QG usually includes two sub-tasks: (1) what to say, i.e. selecting the targets that should be asked. (2) how to say, i.e. formulating the structure of the question and producing the surface realization. This is similar to other natural language generation tasks. In this paper, we focus on the second sub-task, i.e. surface-form realization of questions by assuming the targets are given.

The rule-based approaches usually include the following steps: (1) Preprocess the given text by applying natural language processing techniques, including syntactic parsing, sentence simplification and semantic role labeling. (2) Identify the targets that should be asked by using rules or semantic roles. (3) Generate questions using transformation rules or templates. (4) Rank the over generated questions by well-designed features (Heilman and Smith, 2009, 2010; Chali and Hasan, 2015). The major drawbacks of rule-based approaches include: (1) they rely on rules or templates that are expensive to manually create; (2) the rules or templates lack diversity; (3) the targets that they can deal with are limited.

To tackle the issues of rule-based approaches, the neural network-based approaches are applied to the task of QG. The neural network-based approaches do not rely on hand-crafted rules, and they are instead data driven and trainable in an end-to-end fashion. Serban et al. (2016) firstly introduce an encoder-decoder framework with attention mechanism to generate factoid questions for the facts (i.e. each fact is a triple composed of a subject, a predicate and an object) from FreeBase. Du et al. (2017) introduce sequence-to-sequence model with attention mechanism to generate questions for the text from SQuAD dataset, which contains large-scale manually annotated triples composed of question, answer and the context (i.e. the passage). Zhou et al. (2017) enrich the sequence-to-sequence model with rich features, e.g. answer position and lexical features, and incorporate copy mechanism that allows it to copy words from the context when generating a question. Their experiments show the effectiveness of the rich features and the copy mechanism. Duan et al. (2017) propose to combine templates and sequence-to-sequence model. Specifically, they mine question patterns from a question answering community and apply sequence-to-sequence to generate question patterns for a given text. Tang et al. (2017) model question answering and question generation as dual tasks. It helps generate better questions when training these two tasks together.

In this paper, we observe two major issues with the exiting neural models: (1) The generated question words do not match the answer type, since the models do not pay much attention to the answers that are critical to generate question words. (2) The model copies the context words that are far from and irrelevant to the answer, instead of the words that are close and relevant to the answer, since the models are not aware the positions of the context words. To address these two issues, we propose an answer-focused and position-aware neural question generation model. As to position-aware models, Zeng et al. (2014); Zhang et al. (2017) introduce position feature in the task of relation extraction. They apply this feature to encode the relative distance to the target noun pairs. In the task of QG, Zhou et al. (2017) apply BIO scheme to label answer position, which is a weak representation of relative distance between answer and its context words.

Sequence-to-sequence In recent years, the sequence-to-sequence model has been widely used in the area of natural language generation, including the tasks of abstractive text summarization, response generation in dialogue, poetry generation, etc. Sutskever et al. (2014) propose a sequence-to-sequence model and apply it to the task of machine translation. Bahdanau et al. (2014) introduce attention mechanism to the sequence-to-sequence model and it greatly improves the model performance on the task of machine translation. To deal with the out of vocabulary issue, several variants of the sequence-to-sequence model have been proposed to copy words from source text (Gu et al., 2016; Gulcehre et al., 2016; Cao et al., 2017; See et al., 2017).

3 Our Models

In this section, we describe the details of our models. We first describe the baseline model, a feature-enriched pointer-generator model. Then, we elaborate the proposed answer-focused model and position-aware model to deal with the two issues discussed in previous section. Finally, a hybrid model is introduced to combine these two models.

3.1 Baseline (Feature-enriched Pointer-generator Model)

Our baseline model is an attention-based pointer-generator model (See et al., 2017) enhanced with various rich features proposed by (Zhou et al., 2017). These features include: named entity (NE), part-of-speech (POS) and answer position in the embedding layer of the encoder.

The encoder in our baseline is a bidirectional LSTM, which takes the joint embedding of word, answer position and lexical features

(NE, POS) as input $(w_1, w_2, ..., w_{T_x})\ with\ w_i \in \mathbb{R}^{d_w+d_a+d_n+d_p}$, where T_x is the input length and d_w, d_a, d_n, d_p is the dimensionality of word embedding, answer position embedding, NE embedding and POS embedding respectively. It produces a sequence of hidden states $(h_1, h_2, ..., h_{T_x})$ to represent its input, each of which is a concatenation of a forward and a backward LSTM representation:

$$\begin{aligned} h_i &= [\overleftarrow{h}_i; \overrightarrow{h}_i], \\ \overleftarrow{h}_i &= \text{LSTM}(w_i, \overleftarrow{h}_{i+1}), \\ \overrightarrow{h}_i &= \text{LSTM}(w_i, \overrightarrow{h}_{i-1}) \end{aligned} \tag{1}$$

where $\overleftarrow{h}_i$, $\overrightarrow{h}_i$ are all d_h-dimensional vectors.

The decoder is a unidirectional LSTM conditioned on all encoded hidden states. At decoding step t, the decoder reads an input word embedding w_t, previous attentional context vector c_{t-1} and its previous hidden state s_{t-1} to update its current hidden state $s_t \in \mathbb{R}^{d_h}$:

$$s_t = \text{LSTM}([w_t; c_{t-1}], s_{t-1}) \tag{2}$$

The context vector c_t together with an attention distribution α_t are generated via attention mechanism (Bahdanau et al., 2014). Attention can be regarded as a semantic match between encoder hidden states and the decoder hidden state. It describes how the model spread out the amount it cares about different encoder hidden states during decoding. At step t, the context vector c_t and the attention distribution α_t are calculated as follows:

$$c_t = \sum_{i=1}^{T_x} \alpha_{ti} h_i \tag{3}$$

$$\alpha_{ti} = \text{softmax}(e_{ti}) \tag{4}$$

$$e_{ti} = v^T \tanh(W_h^T h_i + W_s^T s_t + b) \tag{5}$$

where $W_h, W_s, b\ and\ v$ are all trainable parameters.

Our baseline is based on a pointer-generator framework, which includes two complementary modes: a generation mode and a copy mode. The former mode generates words from a given vocabulary as the vanilla sequence-to-sequence model:

$$P_{vocab} = \text{softmax}\,(g(s_t, c_t)) \tag{6}$$

where $g(\cdot)$ is a two-layer feed-forward network with a maxout internal activation. $P_{vocab} \in \mathbb{R}^{|V|}$ denotes the vocabulary distribution with a vocabulary size of $|V|$.

The latter mode copies words directly from the source sequence. As the attention weights already measure the relevance of each input word to the partial decoding state, we treat α_t as the copy probability i.e. $P_{copy}(w) = \alpha_t$. Both modes are switched via a generation probability p_{gen} as follows:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) P_{copy}(w) \tag{7}$$

where p_{gen} is computed from the context vector c_t, decoder hidden state s_t and the decoder input w_t:

$$p_{gen} = \sigma(f(c_t, s_t, w_t)) \tag{8}$$

$f(\cdot)$ indicates a simple feed-forward neural network that emits a single scalar value. The whole network is thus trained end-to-end according to the negative log likelihood loss of target word probability $P(w^*)$. In the baseline, we apply BIO scheme to label answer position, where B,I,O denote the begin of an answer, the non-begin of the answer and the non-answer context words respectively. Besides, we introduce dropout (Srivastava et al., 2014) with maxout (Goodfellow et al., 2013) to tackle over-fitting problem and pre-trained global vectors (Glove) for word representation (Pennington et al., 2014). All these techniques have been verified in their effectiveness in our model.

3.2 Answer-focused Model

The mismatch between generated question words and answer type is a major issue in neural question generation (NQG). Even though answer is a key to question word generation, most NQG models do not focus on answer or weakly emphasis on it when generating question words. As Table 1 shows, a when-question should be triggered for the answer “the end of the Mexican War”, but an answer-irrelevant why-question is generated by the baseline. According to our analysis in Section 1, we discover that nearly 37% bad cases from our baseline fall into this category. To deal with this issue, we develop an answer-focused model.

We observe that the generation of question words is mainly related to the answer and its surrounding words. For example in Table 1, the answer and its context “until after the end of the Mexican War” already involve the essential information to generate a question word “when”. This

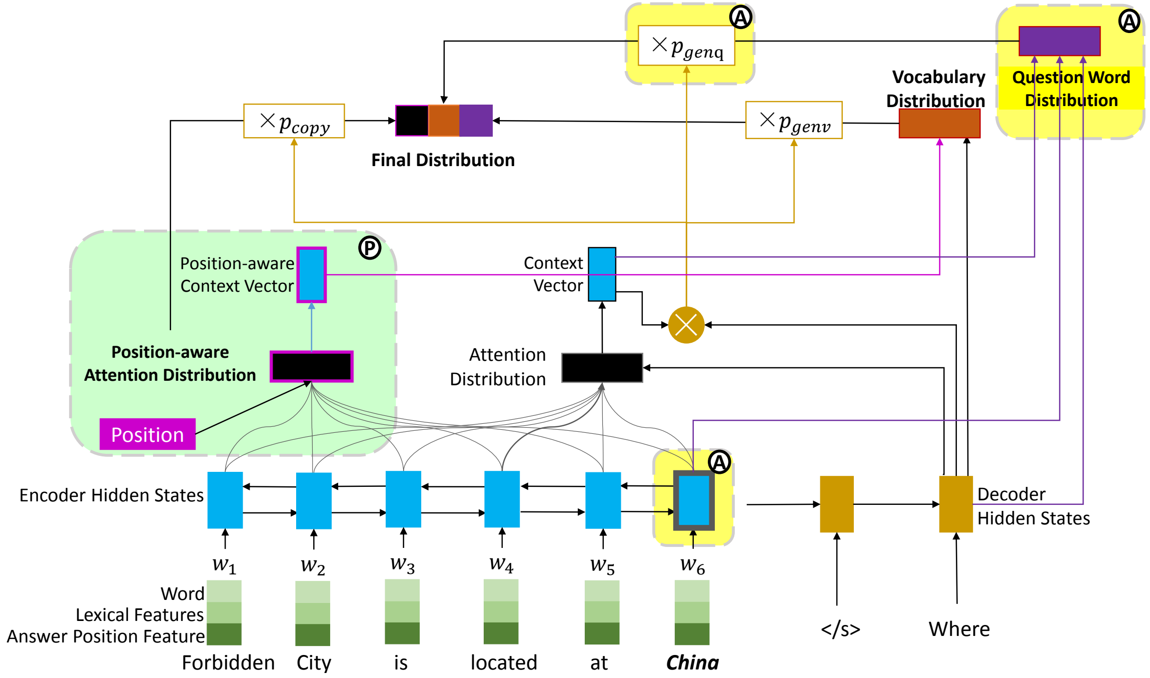

Figure 1: The modules marked with Ⓐ and colored yellow describe the answer-focused model, while the ones marked with Ⓟ and colored green describe the position-aware model. In answer-focused model, comparing to pointer-generator model, we introduce a question word generation mode and generate question words in a restricted vocabulary of question words. In position-aware model, we incorporate word position embeddings to gain a position-aware attention for further generation. The hybrid model has two types of attention distribution: position-aware and non-position-aware attention distribution and two types of context vector accordingly. Question word distribution is generated from non-position-aware context vector while vocabulary distribution is calculated from position-aware one.

suggests that the answer and its context can benefit the generation of question words. We also observe that the number of question words is limited. Therefore, we introduce a specific vocabulary of question words to directly and explicitly model the generation of question words.

As depicted in Figure 1, comparing to pointer-generator model, we introduce an additional question word generation mode to generate question words based on a restricted vocabulary of question words. This mode produces a question word distribution based on an answer embedding v_{ans}, the decoder state s_t and the context vector c_t:

$$P_{question\_word} = \text{softmax}\left(g(v_{ans}, s_t, c_t)\right) \qquad (9)$$

where $P_{question\_word}$ is a $|V_{qw}|$-dimensional probability distribution, and $|V_{qw}|$ is the size of vocabulary of question words. We employ the encoded hidden state at the answer start position as the answer embedding, i.e. $v_{ans} = h_{answer\_start}$. We argue that under bidirectional encoding, this answer embedding has already memorized both the left and the right contexts around the answer region, making it a desired choice. To control the balance among different modes, we introduce a 3-dimensional switch probability:

$$p_{genv}, p_{genq}, p_{copy} = \text{softmax}(f(c_t, s_t, w_t)) \qquad (10)$$

The final probability distribution is calculated via a weighted sum of the three mode probability distributions:

$$\begin{aligned} P(w) = {} & p_{genv}P_{vocab}(w) + p_{copy}P_{copy}(w) \\ & + p_{genq}P_{question\_word}(w) \end{aligned} \qquad (11)$$

3.3 Position-aware Model

Another main issue of NQG is that the generated question copies the context words that are distant from and irrelevant to the answer, instead of the words that are close and relevant to the answer. In other words, attention is distracted by irrelevant words far away from the answer. As shown in Table 2, the baseline model copies "leading theory" that are far away from and unrelated to the answer "homologous recombination", but neglects the words "second theory" and "linear and replicates through" that are closer to the answer. In

Section 1, we analyze the bad cases of our baseline, and find that about 20% suffer from this phenomenon. Further analysis on these cases indicates that the closer a word is to the answer, the more likely it should be copied.

Based on these evidences, we believe that an important reason for the above phenomenon lies in the lack of word position information inside the NQG model. Following this direction, we propose a position-aware model. The model aims at enforcing local attention, and adapting the attention weights so as to put more emphasis on answer-surrounded context words by incorporating word position embeddings.

A straightforward solution to inject position information is to directly incorporate relative word position embeddings. This can inform the model where the answer is and what context words are close to it. As depicted in Figure 1, we achieve this by feeding position embeddings $(d_{p_1}, d_{p_2}, ..., d_{p_{T_x}})$ into the computation of attention distribution through a single layer network:

$$e_{ti} = v^T \tanh(W_d d_{p_i} + W_h h_i + W_s s_t + b) \quad (12)$$

$$\alpha_{ti} = \text{softmax}(e_{ti}) \quad (13)$$

where p_i is the relative distance between the i-th word and the answer, d_{p_i} is the embedding of p_i, which we call word position embedding. By optimizing the attention parameters, our model is expected to discover the correlation between target words and their relative distance from the answer. As a result, distracted attention on irrelevant input words can be avoided.

3.4 A Hybrid Model (Answer-focused and Position-aware)

In Section 3.2 and 3.3, we describe the answer-focused model and position-aware model respectively. In this section, we combine these two models to get a both answer-focused and position-aware hybrid model. As depicted in Figure 1, the hybrid model generates two types of attention distribution: position-aware and non-position-aware distribution. Accordingly, the model has two types of context vector: position-aware and non-position-aware context vector. At time step t, we calculate position-aware attention distribution α'_t as equation (12), (13), and non-position-aware one α_t as equation (4), (5). And we use c_t, c'_t to represent non-position-aware and position-aware context vector calculated as equation (3) respectively.

The hybrid model has three modes as in the answer-focused model. The question word distribution is computed from non-position-aware attention distribution as equation (9), while vocabulary distribution is calculated from position-aware one. The final distribution is weighted sum of the three mode probability distributions:

$$P_{vocab} = \text{softmax}\left(g(s_t, c'_t)\right) \quad (14)$$

$$\begin{aligned} P(w) = {} & p_{genv} P_{vocab}(w) + p_{copy} P_{copy}(w) \\ & + p_{genq} P_{question\_word}(w) \end{aligned} \quad (15)$$

where $P_{copy}(w)$ is the position-aware attention distribution α'_t.

4 Experiments

4.1 Experiment Settings

Dataset In this paper, we conduct the experiments on SQuAD and MARCO. Since the test sets of both data sets are not publicly available, we follow Zhou et al. (2017) to randomly split the development set into two parts and use them as the development set and test set for the task of question generation. In SQuAD, there are $86,635$, $8,965$ and $8,964$ question-answer pairs in our training set, development set and test set, respectively. We directly use the extracted features [1] shared by Zhou et al. (2017). In MARCO, there are $74,097$, $4,539$ and $4,539$ question-answer pairs in our training set, development set and test set, respectively. We use Stanford CoreNLP [2] to extract lexical features. We attach all the processed data sets in the supplemental materials.

Implementation Details In this paper, we set the cutoff length of the input sequence as 100 words. The vocabulary contains the most frequent $20,000$ words in each training set. The vocabulary of question words contains 20 words. We use the pre-trained Glove word vectors [3] with 300 dimensions to initialize the word embeddings that will be further fine-tuned in the training stage. The representations of answer position feature and lexical features at the embedding layer of the encoder

---

[1] `https://res.qyzhou.me/redistribute.zip`

[2] `https://nlp.stanford.edu/software/`

[3] `http://nlp.stanford.edu/data/glove.6B.zip`

DataSet	SQuAD				MARCO			
Model	**BLEU-1**	**BLEU-2**	**BLEU-3**	**BLEU-4**	**BLEU-1**	**BLEU-2**	**BLEU-3**	**BLEU-4**
NQG++ (Zhou et al., 2017)	-	-	-	13.29	-	-	-	-
Pointer-generator model See et al. (2017)	32.32	18.04	12.06	8.60	42.40	29.37	20.71	15.16
Feature-enriched pointer-generator model	40.49	26.11	18.94	14.34	44.45	31.85	23.32	17.90
Answer-focused model	42.10	27.52	20.14	15.36	46.59	33.46	24.57	18.73
Position-aware model	42.16	27.37	20.00	15.23	47.16	34.20	24.40	18.19
Hybrid model	**43.02**	**28.14**	**20.51**	**15.64**	**48.24**	**35.95**	**25.79**	**19.45**

Table 3: The main experimental results of baselines, answer-focused model, position-aware model and a hybrid model on SQuAD and MARCO.

are randomly initialized as 32 dimensional vectors that are trainable during training stage. The hidden size of both the encoder and decoder is 512. We use dropout only in the encoder with a dropout rate 0.5. The size of answer embedding in answer-focused model is 512. The position, that indicates the relative distance between the context words and the answer, ranges from 0 to 80 and its embedding size in position-aware model is 16. We use the optimization algorithm Adagrad (Duchi et al., 2011) with learning rate 0.15, an initial accumulator value of 0.1 and batch size as 64. We use gradient clipping with a maximum gradient norm of 2. During training, we select the best model on development set.

Evaluation Metrics We report BLEU (Papineni et al., 2002) as the main evaluation metric of the question generation systems.

Baselines In the experiments, we have three baselines for comparisons:

- **NQG++ (Zhou et al., 2017)** It is the state-of-the-art neural question generation system on SQuAD that incorporates rich features to the embedding layer of a sequence-to-sequence model and introduces copy mechanism proposed by Gulcehre et al. (2016).
- **Pointer-generator model (See et al., 2017)** It is a sequence-to-sequence model with copy mechanism that has different architecture from the one proposed by Gulcehre et al. (2016). We choose this model since its copy mechanism shows better performance (See et al., 2017). Note that we do not enable the coverage mechanism in this model to have a fair comparison.
- **Feature-enriched pointer-generator model** We add the features to the embedding layer of the pointer-generator model as described in Section 3.1.

4.2 Main Results

Table 3 shows the main results, and we have the following observations:

- The feature-enriched pointer-generator model outperforms NQG++. Both of the two models employ the sequence-to-sequence model with copy mechanism and the same features. The major difference between them is that their copy mechanism has different architecture, and pointer-generator shows better performance.
- The pointer-generator model without the features does not perform well. This verifies the effectiveness of the features extracted by Zhou et al. (2017).
- Both answer-focused model and position-aware model outperform the feature-enriched pointer-generator model and NQG++. We will analyze the effectiveness of these two models in the following two sections, respectively.
- The hybrid model shows the best performance and it outperforms the two single models, answer-focused model and position-aware model.

4.3 Answer-focused Model Analysis

As we discussed in Section 1, one major issue with the neural question generation model is that the generated question word does not match the answer type. The design of answer-focused model is explicitly modeling question word generation by incorporating answer embedding. It is expected that the answer-focused model can reduce such errors. Recall the example shown in Table 1 (Section 1). The answer-focused model correctly predicts the question word, though it copies wrong context words that can be further corrected by the hybrid model. The outputs of the answer-focused model and the hybrid model for the case in Table 1 are as follows:

- **Answer-focused model:** ***When*** did Thoreau's essay come to higher political office ?
- **Answer-focused + Position-aware model:** ***When*** was Thoreau's essay published ?

We further evaluate different systems in terms of the same question word ratio (SQWR). This metric measures the ratio of the generated questions that have the same question words as the reference. Table 4 shows the SQWR of different systems on SQuAD. We can see that answer-focused model outperforms the strong baseline, feature-enriched pointer-generator model.

Model	SQWR
Pointer-generator model (See et al., 2017)	53.17%
Feature-enriched pointer-generator model	71.58%
Answer-focused model	**73.91%**

Table 4: The answer-focused model has the highest same question word ratio.

We re-analyze the 20 (37% of errors) questions (discussed in Section 1) that have the answer type mismatching problem. Our answer-focused model can correct 7 out of the 20 bad cases. All the resolved cases have a commonality that we can easily figure out the answer type from the answer itself, as the case shown in Table 1. We also analyze the remaining 13 of the 20 unresolved cases. (1) 4 cases show that the answer type is closely related to the context words other than the answer. But these context words are far from the answer, and the encoding of the answer by LSTM has little memory of them. As shown in Table 5, the answer "an attempt to avoid responsibility for her actions" is useless to generate a question word "why", while the useful context word "because" is far from the answer. (2) 3 cases show that the answer itself has ambiguity to generate the right question words. (3) 3 cases contain the answers with the wrong named entity labels. (4) 2 cases contain answers which are out of vocabulary. (5) 1 case is hard even for human.

4.4 Position-aware Model Analysis

As discussed in Section 1, another major issue with the neural question generation model is that the model copies the context words that are far from and irrelevant to the answer. The design of position-aware model is tackling this issue by modeling the relative distance between context words and the answer, so that the attention distribution for copy mechanism will be biased towards the context words that are close and relevant to the answer. Recall the example in Table 2 (Section 1). The question generated by the baseline copies the wrong context words "leading theory" instead of "second theory" and "linear and replicates through". The outputs of our position-aware and hybrid model for this case are as follows.

Context: This action was upheld because , according to the U.S. court of appeals for the first circuit , her statement suggested a lack of remorse , **an attempt to avoid responsibility for her actions** , and even a likelihood of repeating her illegal actions .

Answer: an attempt to avoid responsibility for her actions

Reference: Why is giving a defiant speech sometimes more harmful for the individual ?

Question generated by the answer-focused model: What did the court of appeals reject ?

Table 5: A bad case of baseline remains unresolved by applying answer-focused model because answer type is closely related to the context word "because" instead of the answer itself, but "because" is far from the answer. Thus, the encoding of answer has little memory of "because".

- **Position-aware model:** The ***second theory*** suggests that most cpdna is actually ***linear and replicates through*** what ?
- **Position-aware + Answer-focused model:** Most cpdna is actually ***linear and replicates through*** what ?

We can observe that the model can copy the correct context words after introducing the position embedding to the attention distribution. Figure 2 illustrates the attention distributions for copy mechanism before and after introducing position embedding of context words. We can see that "second" has much higher probability in the position-aware attention distribution.

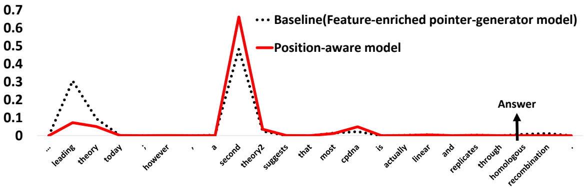

Figure 2: The two attention distributions for copy mechanism when generating questions for the case in Table 2. The position-aware model emphasizes more on the "second" that is close to the answer.

The design of position-aware model aims to help copy the context words close and relevant to the answer. We analyze the effects of position-aware model on copying out of vocabulary (OOV) words from the source sequence. We measure the effects from two perspectives: average precision and average recall. For one generated sequence, the precision is defined as the ratio of the number of OOV words appearing in both the generated question and the reference, and the number of OOV words in the generated question. The recall is defined as the ratio of the number of OOV words appearing in both the generated question and the reference, and the number of OOV words in the reference. The average precision (AP) is the mean of precision of all generated questions, and the average recall (AR) is the mean of recall of all generated questions. As Table 6 shows, our position-aware model can significantly improve the AP and AR, which indicates our position-aware model can help on copying OOV.

Model	AP	AR
Feature-enriched pointer-generator model	21.22%	18.87%
Position-aware model	**22.79%**	**20.62%**

Table 6: Our position-aware model can significantly improve the average precision and recall of copied OOV.

5 Conclusion

In this paper, we find two major issues with the existing neural question generation model. To tackle the two issues, we propose an answer-focused and position-aware model. We further conduct extensive experiments on SQuAD and MARCO dataset. The experimental results show that the combination of our proposed answer-focused model and position-aware model significantly improves the baseline and outperforms the state-of-the-art system.

6 Acknowledgments

This work is supported by the National Basic Research Program of China (973 program, No. 2014CB340505). We thank the anonymous reviewers for their constructive criticism of the manuscript.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase. In *AAAI*, pages 3152–3158.

Yllias Chali and Sadid A Hasan. 2015. Towards topic-to-question generation. *Computational Linguistics*, 41(1):1–20.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Michael Heilman and Noah A Smith. 2009. Question generation via overgenerating transformations and ranking. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA LANGUAGE TECHNOLOGIES INST.

Michael Heilman and Noah A Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer.

Diversity-Promoting GAN: A Cross-Entropy Based Generative Adversarial Network for Diversified Text Generation

Jingjing Xu, Xuancheng Ren, Junyang Lin, Xu Sun
MOE Key Lab of Computational Linguistics, School of EECS, Peking University
`{jingjingxu,renxc,linjunyang,xusun}@pku.edu.cn`

Abstract

Existing text generation methods tend to produce repeated and “boring” expressions. To tackle this problem, we propose a new text generation model, called Diversity-Promoting Generative Adversarial Network (DP-GAN). The proposed model assigns low reward for repeatedly generated text and high reward for “novel” and fluent text, encouraging the generator to produce diverse and informative text. Moreover, we propose a novel language-model based discriminator, which can better distinguish novel text from repeated text without the saturation problem compared with existing classifier-based discriminators. The experimental results on review generation and dialogue generation tasks demonstrate that our model can generate substantially more diverse and informative text than existing baselines.[1]

1 Introduction

Text generation is an important task in Natural Language Processing (NLP) as it lays the foundation for many applications, such as dialogue generation, machine translation (Ma et al., 2018b; Xu et al., 2018a), text summarization (Ma et al., 2018a), and table summarization (Liu et al., 2017). In these tasks, most of the systems are built upon the sequence-to-sequence paradigm (Sutskever et al., 2014), which is an end-to-end model that encodes a source sentence to a dense vector and then decodes the vector to a target sentence. The standard training method is based on Maximum Likelihood Estimation (MLE).

Although being widely applied, the conventional MLE training causes systems to repeatedly generate “boring” sentences, which usually are expressions with high frequency (e.g., “I am sorry” in dialogue generation (Li et al., 2016)). The major reason is that MLE encourages the model to overproduce high-frequency words.[2] The overestimation of high-frequency words discourages the model from generating low-frequency but meaningful words in real data, which makes generated text tend to be repeated and “boring”.

To tackle this problem, we propose a new model for diversified text generation, called DP-GAN. The key idea is to build a discriminator that is responsible for giving reward to the generator based on the novelty of generated text. We consider the text that is frequently generated by the generator as the low-novelty text and the text that is uncommon in the generated data as the high-novelty text. Considering most of the real-world sentences are novel and fluent, we treat the real-world text as the positive example and the generated text as the negative example to train the discriminator. Such training mechanism encourages the discriminator to give higher reward for the text that looks like real-world data. The reward is fed back to the generator, which promotes the generator to generate diverse and fluent text via policy gradient. In this framework, a good discriminator that can assign reasonable reward for the generator is a critical component.

However, directly applying a classifier as the discriminator like most existing GAN models (e.g., SeqGAN (Yu et al., 2017)) cannot achieve satisfactory performance. The main problem is that the reward given by the classifier cannot reflect the novelty of text accurately. First, most existing classifier-based discriminators take the probability of a sequence being true as the reward. When a sentence fits the distribution of real-world

---

[1] The code is available at `https://github.com/lancopku/DPGAN`

[2] For example, the frequency ratios of “the”, “and”, “was” are 4.2%, 3.2%, 1.5% in real data, and they go up to 7.1%, 4.6%, 5.3% in the MLE generated data on our review generation task.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3940–3949
Brussels, Belgium, October 31 - November 4, 2018.

text and is far from the generated data, the reward saturates and scarcely distinguishes the difference between these novel sentences. For example, for a sentence A with mildly high novelty and a sentence B with extremely high novelty, the classifier cannot tell the difference and gives them saturated reward: 0.997 and 0.998. Second, in our tasks, we find that a simple classifier can reach very high accuracy (almost 99%), which makes most generated text receive reward around zero because the discriminator can identify them with high confidence. It shows that the classifier also cannot distinguish the difference between low-novelty text. The reason for this problem is that the training objective of the classifier-based GAN is in fact minimizing the Jensen-Shannon Divergence (JSD) between the distributions of the real data and the generated data (Nowozin et al., 2016). If the accuracy of classifier is too high, JSD fails to measure the distance between the two distributions, and cannot give reasonable reward to the model for generating real and diverse text (Arjovsky et al., 2017).

Instead of using a classifier, we propose a novel language-model based discriminator and use the output of the language model, cross-entropy, as the reward. The main advantage of our model lies in that the cross-entropy based reward for novel text is high and does not saturate, while the reward for text with low novelty is small but discriminative. The analysis of the experimental results shows that our discriminator can better distinguish novel text compared with traditional classifier-based discriminators.

Our contributions are listed as follows:

- We propose a new model, called DP-GAN, for diversified text generation, which assigns low reward for repeated text and high reward for novel and fluent text.
- We propose a novel language-model based discriminator that can better distinguish novel text from repeated text without the saturation problem.
- The experimental results on review generation and dialogue generation tasks show that our method can generate substantially more diverse and informative text than existing methods.

2 Related Work

A great deal of attention has been paid to developing data-driven methods for natural language dialogue generation. Conventional statistical approaches tend to rely extensively on hand-crafted rules and templates, require interaction with humans or simulated users to optimize parameters, or produce conversation responses in an information retrieval fashion. Such properties prevent training on the large corpora that are becoming increasingly available, or fail to produce novel natural language responses.

Currently, a popular model for text generation is the sequence-to-sequence model (Sutskever et al., 2014; Cho et al., 2014). However, the sequence-to-sequence model tends to generate short, repetitive (Lin et al., 2018), and dull text (Luo et al., 2018). Recent researches have focused on developing methods to generate informative (Xu et al., 2018b) and diverse text (Li et al., 2017, 2016; Guu et al., 2017; Shao et al., 2017). Reinforcement learning is incorporated into the model of conversation generation to generate more human-like speeches (Li et al., 2017). Moreover, there are also other methods to improve the diversity of the generated text by using mutual-information, prototype editing, and self attention (Li et al., 2016; Guu et al., 2017; Shao et al., 2017).

In this paper, to handle this problem, we propose to use adversarial training (Goodfellow et al., 2014; Denton et al., 2015; Li et al., 2017), which has achieved success in image generation (Radford et al., 2015; Chen et al., 2016; Gulrajani et al., 2017; Berthelot et al., 2017). However, training GAN is a non-trivial task and there are some previous researches that investigate methods to improve training performance, such as Wasserstein GAN (WGAN) (Arjovsky et al., 2017) and Energy-based GAN (EGAN) (Salimans et al., 2016; Gulrajani et al., 2017; Zhao et al., 2017; Berthelot et al., 2017). GAN in text generation has not shown significant improvement as it has in computer vision. This is partially because text generation is a process of sampling in discrete space where the normal gradient descent solution is not available, which makes it difficult to train. There are some researches that focus on tackling this problem. SeqGAN (Yu et al., 2017) incorporates the policy gradient into the model by treating the procedure of generation as a stochastic policy in reinforcement learning. Ranzato et al. (2016)

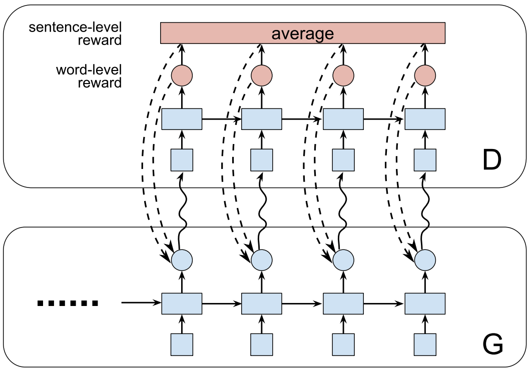

Figure 1: Illustration of DP-GAN. Lower: The generator is trained by policy gradient where the reward is provided by the discriminator. Upper: The discriminator is based on the language model trained over the real text and the generated text.

trains the sequence-to-sequence model with policy gradient for neural machine translation. Bahdanau et al. (2017) applies the actor-critic model on the same task.

3 Diversity-Promoting GAN

The basic structure of our DP-GAN contains a generator that is responsible for generating text and a discriminator that discriminates between the generated text and the real text. The sketch of DP-GAN is shown in Figure 1.

3.1 Overview

The generator G_θ is based on a sequence-to-sequence structure. Given a sentence as input, the generator is capable of generating long text, which contains multiple sentences of various lengths. To put it formally, given the input sentence $x_{1:m} = (x_1, x_2, x_3, ..., x_m)$ of m words from Γ, the vocabulary of words, the model generates the text of T sentences $Y_{1:T} = (y_1, ..., y_t, ..., y_T)$, where y_t from Λ, the set of candidate sentence. The term $y_t = (y_{t,1}, ..., y_{t,K})$ is the t^{th} sentence, where $y_{t,K}$ is the K^{th} word.

The discriminator D_ϕ is a language model. The output of the language model, cross entropy, is defined as the reward to train the generator. Our reward consists of two parts, the reward at the sentence level and that at the word level. With the discriminator and the reward function, we train the generator by reinforcement learning. A sketch of training DP-GAN is shown in Algorithm 1. The details are described as follows.

Algorithm 1 The adversarial reinforcement learning algorithm for training the generator G_θ and the discriminator D_ϕ.

```
Initialize $G_\theta$, $D_\phi$ with random weights $\theta$, $\phi$
Pre-train $G_\theta$ using MLE on a sequence dataset $\mathcal{D} = (X, Y)$
Generate samples using $G_\theta$ for training $D_\phi$
Pre-train $D_\phi$ by Eq. (1)
$N$ = number of training iterations
$M$ = number of training generator
$K$ = number of training discriminator
for each $i = 1, 2, ..., N$ do
    for each $j = 1, 2, ..., M$ do
        Generate a sequence $Y_{1:T} \sim G_\theta$
        Compute rewards by Eq. (2) and Eq. (3)
        Update generator via policy gradient Eq. (5)
        Sample a sequence $Y_{1:T} \sim \mathcal{D}$
        Compute rewards by Eq. (2) and Eq. (3)
        Update generator parameters via Eq. (5)
    end for
    for each $j = 1, 2, ..., K$ do
        Generate samples using $G_\theta$
        Train discriminator $D_\phi$ by Eq. (1)
    end for
end for
```

3.2 Generator

For the concern of real-world applications, this paper assumes that the output of the model can be long text made up of multiple sentences. In order to generate multiple sentences, we build a standard hierarchical LSTM decoder (Li et al., 2015). The two layers of the LSTM are structured hierarchically. The bottom layer decodes the sentence representation and the top layer decodes each word based on the output of the bottom layer. The attention mechanism is used for word decoding (Bahdanau et al., 2014; Luong et al., 2015).

3.3 Discriminator

Most existing GAN models use a binary classifier as the discriminator. The probability of being true is regarded as the reward (Li et al., 2016; Yu et al., 2017). Different from that, we propose a language-model based discriminator D_ϕ that builds on a unidirectional LSTM. We use the output of the language model, cross-entropy, as the reward. Specifically, given a sentence y_t, the cross-entropy based reward for the k^{th} word is calculated as

$$R(y_{t,k}) = -\log D_\phi(y_{t,k}|y_{t,<k})$$

We maximize the reward of real-world text and minimize the reward of generated text to train the discriminator. The reason of minimizing the reward of generated text is that, we expect the text that is repeatedly generated by the generator can be identified by the discriminator and get lower reward. The motivation of maximizing the reward of real-world data lies in that, we expect not only the uncommon text in the generated data can get high reward, but also low-quality text can be punished to some extend. Considering the real-world text is diverse and fluent, we maximize the reward of real-world text to encourage the discriminator to give high reward for the text that looks like the real-world data. Therefore, such training mechanism avoids the problem of novel but low-quality text getting high reward. The loss function of the discriminator is formulated as follows:

$$\begin{aligned} J(\phi) = \\ -\left(E_{Y\sim p_{data}}[R(Y)] - E_{Y\sim G_\theta}[R(Y)]\right) \end{aligned} \tag{1}$$

where $R(Y)$ stands for the averaged reward of Y.

3.4 Reward

Our reward function consists of two parts, the sentence-level reward and the word-level reward, which are illustrated as follows.

3.4.1 Sentence-Level Reward

For a sentence y_t of K words, the reward at the sentence level is the averaged reward of each word:

$$R(y_t) = -\frac{1}{K}\sum_{k=1}^{K} \log D_\phi(y_{t,k}|y_{t,<k}) \tag{2}$$

In contrast, the reward of the existing classifier-based discriminators (Li et al., 2016; Yu et al., 2017) is calculated as follows:

$$R(y_t) = D_\phi(true|y_t)$$

where D_ϕ is a binary classifier judging how likely y_t is from the real-world data.

The major problem of the classifier-based discriminator is that the reward cannot reflect the novelty of text accurately. First, the reward for high-novelty text is easy to saturate, which scarcely distinguishes the difference between novel text. Second, we find that the discriminator can easily achieve very high accuracy on identifying the generated text, which makes most of them get reward around zero. It shows that the classifier still cannot tell the difference between the text with low novelty.

On the contrary, the analysis of experimental result shows that our proposed discriminator can better distinguish high-novelty text from low-novelty text without the saturation problem. The reward for high-novelty text is high and does not saturate while the reward for low-novelty text is small but discriminative.

3.4.2 Word-Level Reward

Considering that the reward for different words in a sentence y_t should be different, we further propose to use the reward at the word level as follows:

$$R(y_{t,k}|y_{t,<k}) = -\log D_\phi(y_{t,k}|y_{t,<k}) \tag{3}$$

It can be found that the classifier-based discriminator only provides reward for the finished sequence. Thus, for a sequence of length T, to evaluate the action-value for a word at the time step t, Monte Carlo Search (MCS) with a roll-out policy G_θ is usually applied to sample the unknown last $T-t$ tokens (Yu et al., 2017). However, this could be computationally expensive because the time complexity is $O(T^2)$. On the contrary, our discriminator can calculate the reward of all words with the time complexity of $O(T)$, which is more computationally efficient.

3.5 Policy Gradient Training

The loss function of the generator (policy) is to maximize the reward from the start state s_0 to the end state (Sutton et al., 1999):

$$\begin{aligned} J(\theta) &= \sum_{t=1}^{T} E[R_{t,K}|s_{t-1},\theta] \\ &= \sum_{t=1}^{T}\sum_{y_{t,1}} G_\theta(y_{t,1}|s_{t-1}) Q_{D_\phi}^{G_\theta}(s_{t-1},y_{t,1}) \end{aligned} \tag{4}$$

where $R_{t,K} = \sum_{k=1}^{K}\gamma^{k-1}R(y_t)R(y_{t,k})$ is the total reward for a complete sentence, including both the sentence-level and the word-level rewards. The term $Q_{D_\phi}^{G_\theta}(s_{t-1},y_{t,1})$ is estimated by $R_{t,1}$. The term γ is the discount rate and s_t is the initial state.

In this paper, we use the policy gradient method (Williams, 1992). The gradient of Eq. (4) is approximated as follows:

$$\begin{aligned} &\nabla_\theta J(\theta) \simeq \\ &\sum_{t=1}^{T}\sum_{k=1}^{K}\gamma^{k-1}R_{t,k}\nabla_\theta \log G_\theta(y_{t,k}|y_{t,<k}) \end{aligned} \tag{5}$$

where $R_{t,k} = \sum_{i=k}^{K} \gamma^{i-1} R(y_t) R(y_{t,i})$ is the total reward starting from step k.

Following previous work (Li et al., 2017), we also use teacher forcing (Bengio et al., 2015) to train the generator. In teacher forcing, the decoder receives the real-world text as input at each time step. The loss function of teacher forcing is the same with that of policy gradient training. The only difference is that the text is generated from G_θ in policy gradient training but from the real data in teacher forcing.

4 Experiment

We evaluate DP-GAN on two real-world natural language generation tasks, review generation and dialogue generation. We first introduce the dataset, the training details, the baselines, and the evaluation metrics. Then, we compare our model with the state-of-the-art models. Finally, we show the experimental results and provide the detailed analysis.

4.1 Datasets

Yelp Review Generation Dataset (Yelp): This dataset is provided by Yelp Dataset Challenge.[3] In our version of review generation, the model should generate a paragraph based on a given sentence. We build a new dataset for this task by splitting the data into two parts. In each review, we take the first sentence as the input text, and the following sentences as the target text. The processed Yelp dataset contains 1,400K, 400K, and 12K pairs for training, validation, and testing, respectively.

Amazon Review Generation Dataset (Amazon): This dataset is provided by McAuley and Leskovec (2013). It consists of review information of fine foods from Amazon. Like Yelp, we process this dataset by extracting the first sentence as the source text and the rest as the target text. The processed Amazon dataset contains 400K, 100K, and 12K pairs for training, validation, and testing, respectively.

OpenSubtitles Dialogue Dataset (Dialogue): This dataset[4] is used for dialogue generation. Following previous work, we treat each turn in the dataset as the target text and the two previous sentences as the source text. We remove the pairs whose response is shorter than 5 words. We randomly sample 1,800K, 500K, and 12K turns for training, validation, and testing, respectively.

Yelp	Token	Dist-1	Dist-2	Dist-3	Dist-S
MLE	151.2K	1.2K	3.9K	6.6K	3.9K
PG-BLEU	131.1K	1.1K	3.3K	5.5K	3.1K
SeqGAN	140.5K	1.1K	3.5K	6.1K	3.6K
DP-GAN(S)	**438.6K**	1.7K	7.5K	15.7K	10.6K
DP-GAN(W)	271.9K	2.8K	14.8K	29.0K	12.6K
DP-GAN(SW)	406.8K	**3.4K**	**22.3K**	**49.6K**	**17.3K**

Amazon	Token	Dist-1	Dist-2	Dist-3	Dist-S
MLE	176.1K	0.6K	2.1K	3.5K	2.6K
PG-BLEU	124.5K	0.6K	1.9K	3.5K	2.3K
SeqGAN	217.3K	0.7K	2.6K	4.6K	3.2K
DP-GAN(S)	**467.6K**	0.8K	3.6K	7.6K	7.0K
DP-GAN(W)	279.4K	1.6K	8.9K	18.4K	9.6K
DP-GAN(SW)	383.6K	**1.9K**	**11.7K**	**26.3K**	**13.6K**

Dialogue	Token	Dist-1	Dist-2	Dist-3	Dist-S
MLE	81.1K	1.4K	4.4K	6.3K	4.1K
PG-BLEU	97.9K	1.2K	3.9K	5.5K	3.3K
SeqGAN	83.4K	1.4K	4.5K	6.5K	4.5K
DP-GAN(S)	**112.2K**	1.5K	5.2K	8.5K	5.6K
DP-GAN(W)	79.4K	1.9K	7.7K	11.4K	6.0K
DP-GAN(SW)	97.3K	**2.1K**	**10.8K**	**19.1K**	**8.0K**

Table 1: Performance of the DP-GAN and three baselines on review generation and dialogue generation tasks. Higher is better. DP-GAN(S), DP-GAN(W), and DP-GAN(SW) represent DP-GAN with only sentence-level reward, only word-level reward, and combined reward, respectively. *Token* represents the number of generated words. Dist-1, Dist-2, Dist-3, and Dist-S are respectively the number of distinct unigrams, bigrams, trigrms, and sentences in the generated text. For example, 1.2K in Dist-1 means 1200 distinct unigrams.

4.2 Baselines

We compare the proposed DP-GAN with the following baseline models:

MLE: The generator is a sequence-to-sequence model. The generator is trained with traditional MLE.

PG-BLEU: The generator is a sequence-to-sequence model. It is trained by policy gradient with the BLEU score of the generated text as the reward (Bahdanau et al., 2017). The advantage is that this model can directly optimize the task-specific score: BLEU.

SeqGAN: Sequence GAN (Yu et al., 2017) uses a binary classifier as the discriminator. Since it is originally for unconditional generation, for a fair comparison, we expand it to the version of conditional generation. We re-implement the generator by replacing a language model with a sequence-to-sequence model.

---

[3] `https://www.yelp.com/dataset/challenge`

[4] `http://opus.lingfil.uu.se/OpenSubtitles.php`

Yelp	Relevance	Diversity	Fluency	All
MLE	1.49	1.73	1.78	1.89
PG-BLEU	1.47	2.59	**1.38**	2.22
SeqGAN	1.48	2.40	1.54	2.12
DP-GAN	**1.32**	**1.23**	1.66	**1.51**
Amazon	Relevance	Diversity	Fluency	All
MLE	1.52	1.81	1.72	1.93
PG-BLEU	1.62	2.48	1.63	2.24
SeqGAN	1.56	2.37	**1.40**	1.97
DP-GAN	**1.31**	**1.25**	1.52	**1.50**
Dialogue	Relevance	Diversity	Fluency	All
MLE	1.19	1.84	1.37	1.87
PG-BLEU	**1.13**	1.85	1.21	1.75
SeqGAN	**1.13**	1.71	**1.20**	1.64
DP-GAN	**1.13**	**1.50**	1.30	**1.55**

Table 2: Results of human evaluation on the three datasets. The score represents the averaged ranking of each model and lower is better. *All* represents the ranking given by annotators based on a comprehensive consideration. It can be seen that DP-GAN results in the largest improvement in terms of diversity and relevance while slightly reducing fluency.

4.3 Training Details

For review generation, we set the number of generated sentences to 6 with the maximum length of 40 words for each generated sentence. Based on the performance on the validation set, we set the hidden size to 256, embedding size to 128, vocabulary size to 50K, and batch size to 64 for the proposed model and the baselines. We use the Adagrad (Duchi et al., 2011) optimizer with the initial learning rate 0.1. In adversarial training, the step for training the generator is 1K, the step for training the discriminator is 5K. Both the generator and the discriminator are pre-trained for 10 epochs before adversarial learning. In particular, for PG-BLEU and SeqGAN, before reinforcement learning or adversarial learning, we pre-train the sequence-to-sequence model for 10 epochs like DP-GAN. For dialogue generation, the settings are the same with review generation, except that we set the number of generated sentences to 1 with the maximum length of 40 words because there is only one sentence in the response.

4.4 Experimental Results

We conduct two kinds of evaluations in this work, automatic evaluation and human evaluation. The details of evaluation results are shown as follows.

4.4.1 Automatic Evaluation

We evaluate the proposed model in terms of several metrics that can reflect the diversity. The results are shown in Table 1. *Token* represents the total number of generated words. Dist-1, Dist-2, Dist-3, and Dist-S are respectively the number of distinct unigrams, bigrams, trigrms, and sentences. DP-GAN(S), DP-GAN(W), and DP-GAN(SW) represent DP-GAN with only sentence-level reward, only word-level reward, and combined reward, respectively. From the results, it is obvious that the proposed model substantially outperforms the existing models. PG-BLEU achieves slightly weaker results compared with MLE. The reason is that PG-BLEU uses BLEU score as the reward for reinforcement learning. However, the BLEU score is low for most of the generated text. The low reward makes it hard to learn from the real data. SeqGAN does not achieve better results, which suggests that the classifier-based discriminator fails to encourage the generator to produce diverse text.

In terms of the total number of generated words, DP-GAN(S) achieves better results than DP-GAN(W). Since the sentence-level reward reflects the novelty of the whole sentence, it gives repeated and short text low reward while novel and longer text high reward. Thus, the generator is encouraged to generate novel text. In terms of the number of distinct n-grams, DP-GAN(W) achieves better results than DP-GAN(S). It is because the word-level reward gives each word more precise score and novel n-grams could be better encouraged. As we can see, DP-GAN(SW), which combines the advantages of sentence-level and word-level rewards, generates not only more diverse n-grams than DP-GAN(S) but also longer text than DP-GAN(W). Since combining the word-level and sentence-level rewards achieves better results than using just one of them, we focus more on the combined reward in the following parts.

In review generation and dialogue generation tasks, it is a widely debated question how well the BLEU score against a single reference can reflect the quality of the generated text (Liu et al., 2016). Thus, although the proposed model achieves better BLEU scores compared with baselines, we omit the detailed comparisons in terms of BLEU for space.

4.4.2 Human Evaluation

We conduct a human evaluation on the test set. For all tasks, we randomly extract 200 samples from the test sets. Each item contains the input text and the text generated by the different systems. The items are distributed to three anno-

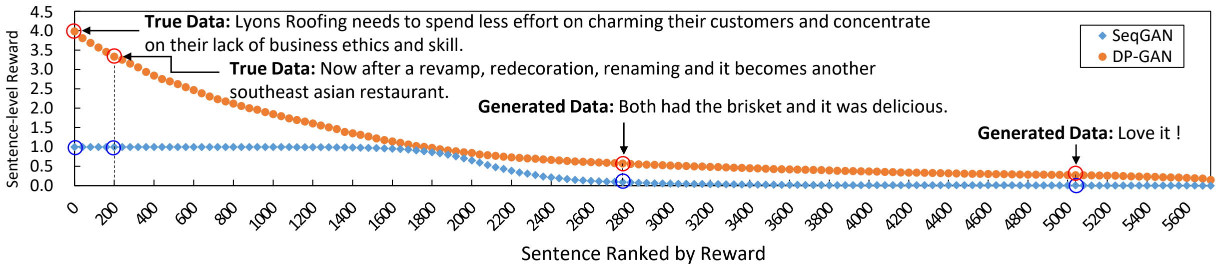
True Data: Lyons Roofing needs to spend less effort on charming their customers and concentrate on their lack of business ethics and skill.

True Data: Now after a revamp, redecoration, renaming and it becomes another southeast asian restaurant.

Generated Data: Both had the brisket and it was delicious.

Generated Data: Love it !

Figure 2: Distribution of rewards between SeqGAN and DP-GAN. The upper two sentences are sampled from the real-world data and the lower two sentences are sampled from the generated data. It is important to note that the sentence-level reward of DP-GAN is averaged word-level reward and a long sentence does not indicate a high score. As we can see, the reward distribution of SeqGAN saturates and cannot distinguish the novelty of the text accurately. In contrast, DP-GAN has a strong ability of resisting reward saturation and can give more precise reward for text in terms of novelty.

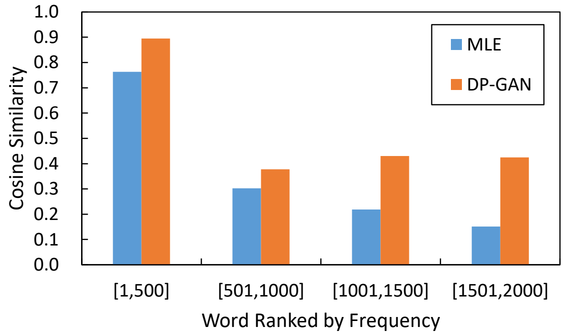

Figure 3: Cosine similarity between the real-world data distribution and the generated data distributions of various models. For example, the first column represents the cosine similarity on top 500 words with the highest frequencies in real-world data. As we can see, the generated data distribution of DP-GAN is closer to the real-world data distribution, especially considering the words of low frequency.

tators who have no knowledge about which system the text is from. Following the work of Li et al. (2017), we require them to rank the generated text considering relevance, diversity, and fluency. It is important to note that all the annotators have linguistic background. Relevance means that how likely the generated text is related to the input text. Diversity means that how much the generated text provides specific information, rather than "dull" and repeated information. Fluency means that how likely the generated text is produced by human. *All* represents the ranking given by annotators based on a comprehensive consideration of all human evaluation metrics. The results of human evaluation are shown in Table 2. It needs to be mentioned that in the special case that several pieces of generated text are exactly the same, they are given the same ranking. The inter-annotator agreement is satisfactory considering the difficulty in the human evaluation. The Pearson's correlation coefficient is 0.76 on diversity, 0.59 on fluency and 0.27 on relevance, with $p < 0.0001$. The p-value is all below 0.001, indicating high correlation and agreement. Table 2 shows that DP-GAN brings the largest improvement in terms of diversity and relevance while scarcely reducing fluency. First, DP-GAN significantly outperforms baselines in term of diversity. Second, DP-GAN achieves the best performance in terms of relevance. Diverse text generated by DP-GAN brings more specific expressions, increasing the performance of relevance. For example, given input sentence*"food is good"*, the relevance of *"love it, beef is the best"* is obviously better than that of *"love it"*. Third, the fluency of DP-GAN is slightly worse than that of baselines. It is mainly due to the fact that the text generated by DP-GAN is much longer. From Table 1, we can see that the number of words generated by DP-GAN is almost three times than that of baselines in the Yelp dataset.

4.4.3 Analysis: Why It Works

In this section, we provide detailed analysis to see why our proposed method works better.

In Figure 2, we demonstrate the reward distributions of our model and SeqGAN. It can be seen that the reward of SeqGAN cannot reflect the novelty of text accurately. First, when the novelty of text is relatively high, the reward given by SeqGAN saturates and cannot tell the difference between high-novelty text. Even with different degrees of novelty, the two high-novelty sentences sampled from the true data in the figure get nearly

Input: *One of my favorite places to eat.*
MLE: *Service is great.*
PG-BLEU: *Service is always good.*
SeqGAN: *Love the chicken and waffles. Service is always great.*
DP-GAN: *Love the fact that they have a large selection of food. Service is always great and the food is always fresh. I've been to this place a few times and have never been disappointed.*

Input: *Excellent service, clean, nice ambience.*
MLE: *Food is very good.*
PG-BLEU: *Food is very good.*
SeqGAN: *Food is very good.*
DP-GAN: *I had the chicken and beef and it was delicious. Food is good and the service is great. I would recommend this place to anyone who wants to eat here.*

Input: *I brought friends – seven of them – who'd never been to a Bar Louie before.*
MLE: *Food was mediocre, but the service was great. I will not be returning.*
PG-BLEU: *Food was terrible. Wouldn't recommend this place to anyone.*
SeqGAN: *Food was just okay. I wouldn't recommend this place.*
DP-GAN: *Ordered a chicken salad with a side of fries, and it was just ok. Food was good, but the service was slow. I had to ask for a refill on my drink, and the bartender was rude. I wouldn't recommend this place to anyone.*

Input: *I don't think it's too spicy, so i add a little hot curry spice to the sauce.*
MLE: *It's great for cooking. I have to say, I'm not sure if it is the same.*
PG-BLEU: *Love it!*
SeqGAN: *Love it! Love this stuff. I have used it for years and it is very good.*
DP-GAN:*I've tried many different brands of hot sauces but this one is the best. I've also used it in soups and stews. I also like that it is organic and has a great flavor.*

Table 3: Examples generated by DP-GAN and the state-of-the-art models on the Yelp dataset. It can be found that the text generated by baselines is more generic and repeated, while our model generates text with more specific details and higher diversity.

the same reward in SeqGAN. Second, most generated text receives reward around zero because of the high accuracy of classifier. It is hard for such reward to distinguish the difference between low-novelty text. For example, as shown in the figure, "*Both had the brisket and it was delicious*" is much more informative than "*Love it!*". The discriminator of SeqGAN gives them practically the same reward, while the proposed discriminator can better distinguish the two sentences in terms of novelty. In fact, the classifier in SeqGAN trained for 10 epochs can reach very high accuracy, that is, 98.35% and 99.63% for Yelp and Amazon, respectively. If the accuracy of classifier is too high, the classifier cannot give reasonable reward to the generator for generating real and diverse text (Arjovsky et al., 2017).

In contrast, the language-model based reward given by DP-GAN better reflect the novelty of the text. The novel text is given high reward that does not saturate. The generated data, which can be less novel, is given relatively low but nonzero reward that can encourage the generator to generate diverse expressions. The refined reward leads to more efficient training, thus resulting in better performance.

We also compare the cosine similarity between the real-world data distribution and the generated data distributions of various models. Figure 3 shows the results. We calculate the cosine distance between two vectors, where each element is the frequency of a word indexed by its rank in real-world data. For example, the first element in the vector means the frequency of the word that ranks first in real-world data. The word frequency vector is divided into 4 vectors to show the similarity of words of different frequencies. The distribution of words are more similar when they occur more frequently in real-world data. As DP-GAN promotes diversity, words of low frequency in real-world data are better learned and the similarity is much better than that of MLE. In all, the generated data distribution of DP-GAN is closer to the real-world data distribution in all intervals, especially considering the words of low frequency.

Table 3 presents the examples generated by different models on the Yelp dataset. It can be found that the text generated by MLE is more generic and repeated, while PG-BLEU and SeqGAN do not perform obviously better than MLE. Moreover, it can be clearly seen that our model generates text with more specific details and higher diversity.

5 Conclusions

In this paper, we propose a new model, called DP-GAN, to promote the diversity of the generated text. DP-GAN assigns low reward for repeated text and high reward for novel and fluent text, encouraging the generator to produce novel and diverse text. We evaluate DP-GAN on two tasks and the findings are concluded as follows: First, the proposed method substantially outperforms the baseline methods in automatic and human evaluations. It shows that DP-GAN is capable of producing more diverse and informative text. Second, the proposed discriminator can better distinguish novel text from repeated text with the saturation problem compared without traditional classifier-based discriminators. Third, with the improvement of diversity, the generated data distribution of DP-GAN is closer to the real-world data distribution compared with that of MLE.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028). We thank Wei Li and Bingzhen Wei for providing the thoughtful suggestions. Xu Sun is the corresponding author of this paper.

References

Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *ICML 2017*, pages 214–223.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR 2017*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2014*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS 2015*, pages 1171–1179.

David Berthelot, Tom Schumm, and Luke Metz. 2017. BEGAN: boundary equilibrium generative adversarial networks. *CoRR*, abs/1703.10717.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS 2016*, pages 2172–2180.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP 2014*, pages 1724–1734.

Emily L. Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS 2015*, pages 1486–1494.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS 2014*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. 2017. Improved training of wasserstein gans. In *NIPS 2017*, pages 5769–5779.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2017. Generating sentences by editing prototypes. *CoRR*, abs/1709.08878.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL 2016*, pages 110–119.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *ACL 2015*, pages 1106–1115.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *EMNLP 2017*, pages 2157–2169.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. *CoRR*, abs/1805.03989.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP 2016*, pages 2122–2132.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text generation by structure-aware seq2seq learning. *CoRR*, abs/1711.09724.

Liangchen Luo, Jingjing Xu, Junyang Lin, Qi Zeng, and Xu Sun. 2018. An auto-encoder matching model for learning utterance-level semantic dependency in dialogue generation. In *EMNLP, 2018*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP 2015*, pages 1412–1421.

Shuming Ma, Xu Sun, Junyang Lin, and Houfeng Wang. 2018a. Autoencoder as assistant supervisor: Improving text representation for chinese social media text summarization. *CoRR*, abs/1805.04869.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018b. Bag-of-words as target for neural machine translation. *CoRR*, abs/1805.04871.

Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW 2013*, pages 897–908.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-gan: Training generative neural samplers using variational divergence minimization. In *NIPS 2016*, pages 271–279.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR 2016*.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *NIPS 2016*, pages 2226–2234.

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *EMNLP 2017*, pages 2210–2219.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 2014*, pages 3104–3112.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS 1999*, pages 1057–1063.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018a. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *ACL, 2018*.

Jingjing Xu, Yi Zhang, Qi Zeng, Xuancheng Ren, Xiaoyan Cai, and Xu Sun. 2018b. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *EMNLP, 2018*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI 2017*, pages 2852–2858.

Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. 2017. Energy-based generative adversarial network. In *ICLR 2017*.

Towards a Better Metric for Evaluating Question Generation Systems

Preksha Nema[1,2] **Mitesh M. Khapra**[1,2]
[1]Department of Computer Science and Engineering, Indian Institute of Technology Madras
[2] Robert Bosch Center for Data Science and Artificial Intelligence,
Indian Institute of Technology Madras
`{preksha,miteshk}@cse.iitm.ac.in`

Abstract

There has always been criticism for using n-gram based similarity metrics, such as BLEU, NIST, *etc*, for evaluating the performance of NLG systems. However, these metrics continue to remain popular and are recently being used for evaluating the performance of systems which automatically generate questions from documents, knowledge graphs, images, *etc*. Given the rising interest in such automatic question generation (AQG) systems, it is important to objectively examine whether these metrics are suitable for this task. In particular, it is important to verify whether such metrics used for evaluating AQG systems focus on *answerability* of the generated question by preferring questions which contain all relevant information such as question type (Wh-types), entities, relations, *etc*. In this work, we show that current automatic evaluation metrics based on n-gram similarity do not always correlate well with human judgments about *answerability* of a question. To alleviate this problem and as a first step towards better evaluation metrics for AQG, we introduce a scoring function to capture *answerability* and show that when this scoring function is integrated with existing metrics, they correlate significantly better with human judgments. The scripts and data developed as a part of this work are made publicly available.[1]

1 Introduction

The advent of large scale datasets for document Question Answering (QA) (Rajpurkar et al., 2016; Nguyen et al., 2016; Joshi et al., 2017; Saha et al., 2018a) knowledge base driven QA (Bordes et al., 2015; Saha et al., 2018b) and Visual QA (Antol et al., 2015; Johnson et al., 2017) has enabled the development of end-to-end supervised models for QA. However, as is always the case, data-hungry neural network based solutions could benefit from even more training data, especially in specific domains which existing datasets do not cater to. Creating newer datasets for specific domains or augmenting existing datasets with more data is a tedious, time-consuming and expensive process. To alleviate this problem and create even more training data, there is growing interest in developing techniques that can automatically generate questions from a given source, say a document (Du et al., 2017; Du and Cardie, 2017), knowledge base (Reddy et al., 2017; Serban et al., 2016), or image (Li et al., 2017) . We refer to this task as Automatic Question Generation (AQG). For example, given the document in Table 1, the task is to automatically generate a question whose answer is also contained in the document.

Document: In **1648** before the term “genocide” had been coined , the Peace of Westphalia was established to protect ethnic, racial and in some instances religious groups.
Possible Question: In which year was the Peace of Westphalia established ?

Table 1: A sample question generated by a human.

Given the practical importance of AQG and its potential to influence research in QA, it is not surprising that there has been prolific work in this field in the past one year itself (Jain et al., 2017; Li et al., 2017; Zhang et al., 2017; Du et al., 2017; Duan et al., 2017). Before this field grows further, it is important that the community critically examines the current evaluation metrics being used for this task. In particular, there is a need to closely examine the utility of existing n-gram based similarity metrics such as BLEU (Papineni et al., 2002), METEOR (Lavie and Denkowski,

---

[1] `https://github.com/PrekshaNema25/Answerability-Metric`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3950–3959
Brussels, Belgium, October 31 - November 4, 2018.

2009), NIST (Doddington, 2002), *etc.* which have been adopted for this task. This work is a first step in that direction where we propose that apart from n-gram similarity, any metric for AQG should also take into account the *answerability* of the generated questions. With the help of a few examples below, we illustrate that *answerability* depends on the presence of relevant information such as question type (Wh-types), entities, relations, *etc*, and it is possible that a generated question has a high BLEU score but is still unanswerable and hence not useful.

To begin with, consider the task of answering questions from a Knowledge Base. Let us assume that the intended (gold standard) question is "Who was the director of Titanic?" and two different AQG systems generate the following questions "S1: director of Titanic?" and "S2: Who was the director of?". Any n-gram based evaluation metric would obviously assign a higher score to S2 (BLEU3: 81.9) than S1 (BLEU3: 36.8). However, as should be obvious S1 contains all the relevant information, and most humans would be easily able to understand and answer this question. A good evaluation metric should capture this notion of *answerability* and give more importance to relevant words in the question which brings us to the question "Which words are relevant?"

The above example might give the impression that *named entities* are essential but other words are not. However, this is misleading and may not always be the case. For example, consider these questions over an image: "Are the cats drinking milk?" v/s "How many cats are drinking milk?". These two questions have very different meaning indicating that even words like *are* and *how* are also crucial. Similarly, consider the task of answering questions from a passage titled "Matt Damon". In this case, most humans will be able to answer the question "What is the birth date of" even though the named entity is missing given that the passage only talks about "Matt Damon". Thus, in some cases, depending on the source (document, knowledge base, image) different portions of the question may be important.

To concretize the intuitions developed with the help of the above examples, we first collect human judgments. Specifically, we take questions from existing datasets for document QA, knowledge base QA and visual QA and add systematic noise to these questions. We show these questions to humans and ask them to assign scores to these questions based on the *answerability* and hence the usefulness of these questions (*i.e.*, whether the question contains enough information for them to be able to answer it correctly). We also compute various n-gram similarity metrics (BLEU, METEOR, NIST) comparing the noisy questions to the original questions and show that these metrics do not correlate well with human judgments. Similar studies (Callison-Burch et al., 2006; Liu et al., 2016) have already shown that these metrics do not correlate well with *fluency, adequacy, coherence* but in this work, we focus on *answerability*.

Based on the human evaluations, we propose to modify existing metrics to focus on *answerability* in addition to n-gram similarity. The idea is to make these metrics flexible such that, if needed, the weight assigned to *answerability* and n-gram similarity can be adjusted depending on the task (document QA, Knowledge-Base QA, Visual QA). Further, for capturing *answerability* we propose additional weights for different components of the question (question type, content words, function words, and named entities) These weights can be learned from a small amount of human annotated data and may differ from task to task.

2 Related Work

We have organized our literature survey into 2 parts: (i) question generation systems (ii) studies which analyze evaluation metrics used for NLG.

Question Generation: Early work on question generation used rule-based approaches to generate questions from declarative sentences (Heilman and Smith, 2010; Mostow and Chen, 2009; Lindberg et al., 2013; Labutov et al., 2015). More recent works use attention based neural models for question generation (Du and Cardie, 2017; Du et al., 2017). Some models (Yuan et al., 2017) feed the generated questions to a QA system and use the performance of the QA system as an indicator of the quality of the questions. A few models (Wang et al., 2017; Tang et al., 2017) treat question answering (QA) and question generation (QG) as complementary tasks and focus on jointly training for these two tasks. Other models focus only on the performance of the QA task (Yang et al., 2017; Duan et al., 2017) and not explicitly on the quality of the generated questions. Apart from generating questions from text there is also research on generating questions from images (Jain et al., 2017; Li et al., 2017; Zhang et al., 2017) and knowledge base (Serban et al., 2016; Reddy et al., 2017).

Evaluation metrics for NLG: Current popular metrics for NLG such as BLEU (Papineni et al., 2002), METEOR(Lavie and Denkowski, 2009), ROUGE (Lin, 2004) and NIST (Doddington, 2002) essentially compute the n-gram similarity between the reference sentence and the generated sentence. Though these metrics are very popular and are used for a wide range of NLG tasks including AQG, there has always been criticism for using these metrics (for example, see (Callison-Burch et al., 2006; R et al., 2007; Callison-Burch, 2009)). More recently, there has been criticism (Liu et al., 2016) for using such metrics for evaluating dialog systems eventually resulting in a new metric (Lowe et al., 2017). This new metric while very important, came a bit late in the day and much after several dialog systems were proposed, evaluated and compared using the above n-gram based metrics. It is very important to prevent a similar situation in question generation where many systems get proposed without evaluating them using the right metric. Our work is a first step in this direction, and we hope it will lead to more research in designing the right metrics for AQG.

3 Current Evaluation Metrics

We give a quick overview of the metrics which are currently used for evaluating AQG systems.

BLEU: BLEU is a precision-based evaluation metric which considers exact n-gram matches. For a given value of n, the precision is computed as the fraction of n-grams in the generated hypothesis which match some n-gram in the reference hypothesis. The final BLEU score is computed as the geometric mean of the n-gram precisions obtained by varying n from 1 to N where N is typically 3 or 4. It also contains a brevity penalty to penalize hypothesis that are too short.

METEOR: As opposed to BLEU, METEOR uses both precision and recall, *i.e.*, it computes the fraction of the hypothesis which matches the reference (precision) as well as the fraction of the reference which is contained in the hypothesis (recall). Further, unlike BLEU which only considers exact matches, METEOR also considers matches with stemmed words, synonyms, and paraphrases. It also gives different weightage to matches corresponding to function words and matches corresponding to content words. The final score is the harmonic mean of the precision and recall calculated based on these four matches. Additionally, it also includes a fragmentation penalty to account for gaps and differences in word order. In effect, METEOR is a parametric metric where the different parameters, *viz.*, (i) fragmentation penalty, (ii) weights of different matchers (exact, stemmed, synonyms, paraphrases) and (iii) weights of function and content words, are tuned to maximize correlation with human judgments.

NIST: NIST is a variant of the standard BLEU metric that takes into account the relative importance of each n-grams in the sentence. In particular, the metric gives a high weightage to n-grams which have a lower frequency in the corpus and hence are more informative as compared to very frequent n-grams which are less informative. Further, unlike BLEU which takes the geometric mean of n-gram precisions, NIST takes the arithmetic mean of these precisions. Additionally, they make a small change to the brevity penalty to minimize the impact of minor variations in the length of the hypothesis.

ROUGE: ROUGE is a set of evaluation metrics which were proposed in the context of automatic summarization. Typically, most studies use ROUGE-L, which is F-measure based on the Longest Common Subsequence (LCS) between a candidate and target sentence. Given two sequences, a common subsequence is the set of words which appear in both the sequences in the same order but unlike n-grams the common subsequence does not need to be contiguous. LCS is the longest of such common subsequences. For example, given the sentences candidate:"the boy went home" and reference:"the boy will go home", "the boy home" is the longest common subsequence even though it is not contiguous.

4 Human Judgments For Answerability

As mentioned earlier, for AQG, in addition to n-gram similarity, we also need to focus on the *answerability* of the generated questions. As illustrated in Section 1, answerability of a question depends on whether it contains all relevant information, such as question type (Wh-types), named entities and content words (often relations). Further, depending on the task (document QA, knowledge-base QA or visual QA) the importance of these words may vary. We perform human evaluations to ascertain the importance of each of these components across different QA tasks. These evaluations allow us to independently analyze the importance of each of these components for the 3 QA tasks. In the remainder of this section, we describe the (i) process of creating noisy questions (ii) instructions given to the evaluators and the (iii) inferences drawn from human evaluations.

4.1 Creating Noisy Questions

We took 1000 questions each from 3 popular QA datasets, *viz.*, SQuAD, WikiMovies, and VQA. SQuAD (Rajpurkar et al., 2016) is a reading comprehension dataset consisting of around 100K questions based on passages from around 500 Wikipedia articles. The WikiMovies dataset contains around 100K questions which can be answered from a movie knowledge graph containing 43K entities and 9 relations *(director, writer, actor, etc.)*. The VQA dataset is an image QA dataset containing $265,016$ images with around 5.4 questions on average per image.

We then created noisy versions of these questions using one of the following four methods:

Dropping function words: We refer to the list of English function words as defined in NLTK (Loper and Bird, 2002) and drop all such words from the question. Note that a noisy question with all function words dropped will have a very low BLEU score compared to the original question.

Dropping Named Entities: In our setup, identifying named entities in questions was easy because the questions were well formed and all named entities were capitalized. Alternately, we could have used the Stanford NER. However, on manual inspection, we found that marking the capitalized words as named entities were sufficient. We randomly dropped at most three named entities per question. This allows us to study how humans rate the output of an AQG system which does not contain the correct named entities.

Dropping Content Words: Words other than function words and named entities are also crucial for *answerability*. For example, "Who killed Jane?" and "Who married Jane?" lead to totally different answers. The word "killed/married" is very relevant to ascertain the correct answer. These words typically capture the relation between the entities involved (for example, *killed (John, Jane)*). We identify such important (content) words as ones which are neither question types (7-Wh questions) nor named entities nor stop-words. This perturbation allows us to study how humans rate an AQG system which does not produce the correct content (relation) words.

Changing the Question type: Changing the question type can lead to a different answer altogether or can make the question incoherent. For example the answers to "Who killed Jane?" and "What killed Jane?" are completely different. We create a noisy question by randomly changing the type of the question (for example. replace "who" with "what"). These question types are well defined (7-Wh questions including "*how*") and hence it is easy to identify and replace them. This allows us to study the importance of correct question type in the output of an AQG system.

Note that, an alternate way of collecting human judgments would have been to take the output of existing AQG systems and ask humans to assign answerability scores to these questions based on the presence/absence of the above mentioned relevant information. However, when we asked human evaluators to analyze 200 questions generated by an existing AQG system, they reported that the quality was poor. In particular, after having discussions with annotators, we found that using this output, it would be very difficult to conduct such a systematic study to assess the importance of different words in the question. Hence, we chose to use systematically simulated noisy questions.

4.2 Instructions

We asked the annotators to rate the *answerability* of the above noisy questions on a scale of 1-5. The annotators were clearly told whether the questions belonged to documents or knowledge bases or images. In our initial evaluations, we also tried showing the actual source (image or document) to the annotators. However, we realized that this did not allow us to do an unbiased evaluation of the quality of the questions. The annotators inferred missing information from the document or image and marked the question as answerable (even though the relevant entity *cat* is missing in the question). For example, consider the image of a cat drinking milk and the question "What is the drinking ?" If a human is shown the image then she can easily infer that the missing information is "cat" and hence mark the question as answerable. This clearly biases the study, and therefore we did not show the source to the evaluators.

Rating	**Description**	**Examples**
1	All important information is missing and it is impossible to answer the question	“What is against the ~~sign~~ ?”, “Why is using ~~O2~~ instead of ~~CO2~~ less efficient?”
2	Most of the important information is missing and I can’t infer the answer to the question	“Which films did ~~Lee H. Katzin~~ direct ?”, “Low doses of ~~anti-inflammatories~~ are sometimes used with what ~~classes~~ of drugs?”
3	Some important information is missing leading to multiple answers	“What Harvard Alumni ~~was the~~ Palestine Prime Minister?”, “What country ~~is the~~ teaching subject discussing?”
4	Most of the important information is present and I can infer the answer	“How ~~far~~ from the Yard is the Quad located?”,“what ~~films~~ did Melvin Van Peebles star in?”
5	All important information is present and I can answer the question	“What globally popular half marathon began ~~in~~ 1981?”, “What kind ~~of~~ vehicle ~~is~~ parked ~~the~~ sidewalk?”

Table 2: Instructions along with the examples. The striked out words were removed as a part of systematic noise from the original question.

Dataset	κ	Pearson	Spearman
SQuAD	0.63	0.823	0.795
WikiMovies	0.81	0.934	0.927
VQA	0.70	0.842	0.822

Table 3: Inter annotator agreement, Pearson and Spearman coefficients between Human Scores.

A total of 25 in-house annotators participated in our study, and we got each question evaluated by two annotators. The annotators were Computer Science graduates competent in English. We did an initial pilot using the instructions mentioned in Table 2, but due to the subjective nature of the task, it was difficult for the annotators to agree on the notion of *important information*. In particular, we found that the annotators disagreed between *most important information* and *all important information* (*i.e.*, they were confused between rating 1 v/s 2 and 4 v/s 5). We, therefore, did a small pilot with a group of 10 annotators and asked them to evaluate around 30 questions from each dataset and help us refine the guidelines to define the notion of importance clearly. Based on group discussions with the annotators we arrived at additional example based guidelines to help them distinguish between cases where *“all the”*, *“most of the”* and *“some of the”* important information is present. The original instructions and various examples (some of which are shown in Table 2) were then shared and explained to all the annotators, and they used these to provide their judgments.

4.3 Human-Human Correlation

In Table 3, we report the average inter-annotator agreement between the ratings using Cohen’s kappa (κ) score (Cohen, 1968). Based on guidelines in (McHugh, 2012) we note that we have a strong inter-annotator agreement for WikiMovies and moderate agreement for SQuAD and VQA. Figure 1 indicates that there is a linear correlation between the two ratings for each question and hence we measured the correlation using Pearson coefficient. For completeness, we also measure the monotonic correlation using Spearman coefficient. The Spearman coefficient is slightly lower than the Pearson coefficient because the inter-annotator agreement is stronger at the tail of the distribution *i.e.*, when the question is either very bad (Rating: 1) or very good (Rating: 5).

Metric	**SQuAD**		**WikiMovies**		**VQA**	
	Pearson	*Spearman*	*Pearson*	*Spearman*	*Pearson*	*Spearman*
BLEU1	0.167	0.165	0.179	0.144	-0.025*	-0.048*
BLEU2	0.100*	0.103*	0.072*	0.087*	-0.075*	-0.091*
BLEU3	0.080*	0.086*	0.036*	0.001*	-0.126	-0.114
BLEU4	0.065*	0.067*	-0.020*	-0.011*	-0.086*	-0.127
ROUGE-L	0.165	0.158	0.091*	0.043*	-0.009*	-0.053*
METEOR	0.107	0.124	0.198	0.214	-0.035*	0.009*
NIST	0.173	0.158	0.088*	-0.033*	0.158	0.169

Table 4: Correlation between existing metrics and human judgments. Note that the values with $*$ are **not** statistically significant (p-value > 0.01).

4.4 Correlation between human scores and existing evaluation metrics

We first compute BLEU, METEOR, NIST and ROUGE-L score for each noisy question by comparing it to the original question. We then compute the correlation of each of these scores with annotator ratings. Note that to compute correlation, the annotator ratings are combined to obtain a gold score. The ratings are normalized using the normalization method mentioned in (Blatz et al., 2004) and then averaged to obtain the gold score. For SQuAD and VQA, we observe that NIST which gives more weightage to informative n-grams correlates better than other metrics. For WikiMovies, METEOR which even allows non-

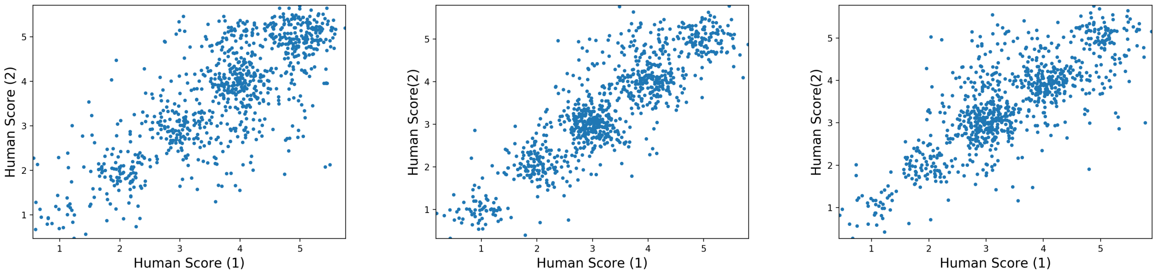

Figure 1: Human-Human Correlation for SQUAD, WikiMovies and VQA respectively.

exact word matches correlates better than other metrics. For SQuAD and WikiMovies, the correlation of human scores with the simple unigram based BLEU1 score is higher than that with other metrics. This is in line with the observation we made earlier that humans can understand and answer questions that are not well-formed, *e.g.*, "What birth-date Damon?".

5 Modifying existing metrics for AQG

The above study suggests that existing metrics do not correlate well with human judgments about *answerability*. We propose modifications to these metrics so that in addition to n-gram similarity they also account for *answerability*. Based on the human evaluations, we found that *answerability* mainly depends on the presence of 4 types of elements, *viz.*, *relevant content words*, *named entities* and *question types* and *function words*. As outlined in Section 4.1 it is easy to identify these elements in the question. Let $c(S_r), c(S_n), c(S_q)$ and $c(S_f)$ be the number of **r**elevant words, **n**amed entities, **q**uestion words and **f**unction words respectively in the noisy question which have corresponding matching words in the gold standard reference question. We can then compute the weighted average of the precision and recall of each of these elements as

$$P_{avg} = \sum_i w_i \frac{c(S_i)}{|l_i|} \qquad R_{avg} = \sum_i w_i \frac{c(S_i)}{|r_i|}$$

where $i \in \{r, n, q, f\}$, $\sum_i w_i = 1$ and $|l_i|$, $|r_i|$ is the number of the words belonging to i^{th} type of element in the noisy question and reference sentences respectively. Just to be clear r, n, q, f stand for *relevant content words*, *named entities* and *question types* and *function words* respectively. Note that w_i's are tunable weights and in Section 5.1, we explain how to tune these weights.

Datasets	w_{ner}	w_{imp}	w_{sw}	w_{qt}	δ
SQuAD	0.41	0.36	0.03	0.20	0.66
WikiMovies	0.55	0.31	0.02	0.11	0.83
VQA	0.04	0.59	0.15	0.21	0.75

Table 5: Coefficients learnt for Q-BLEU1 from human judgments across different datasets.

$$\text{Answerability} = 2.\frac{P_{avg} R_{avg}}{P_{avg} + R_{avg}}$$

We can combine this *answerability* score with any existing metric (say, BLEU4) to derive a modified metric for AQG as shown below:

$$Q\text{-BLEU4} = \delta \text{Answerability} + (1 - \delta)\text{BLEU4} \tag{1}$$

such that $\delta \in \{0, 1\}$ to make sure that Q-Metric ranges between 0 to 1. Similarly, we can derive Q-NIST, Q-METEOR and so on.

5.1 Tuning the weights w_i's and δ

We tuned the weights (w_i's and δ) using the human annotation data. For each source (document, knowledge-base, and images), annotators evaluated 1000 noisy questions. The annotator scores were first scaled between 0 to 1 using the normalization method in (Blatz et al., 2004), and the normalized scores were averaged to obtain the final gold score. For each source, we used 300 of these annotations and used bagging to find the optimal weights. In particular, we drew 200 samples randomly from the given set of 300 samples and did a grid search to find w_i's and δ such that the Q-METRIC computed using Equation 1 had maximum correlation with human scores. We repeated this process for $k = 20$ times and computed the optimal w_i's and δ each time. We found that for

Q-Metric	SQuAD		WikiMovies		VQA	
	Pearson	*Spearman*	*Pearson*	*Spearman*	*Pearson*	*Spearman*
Q-BLEU1	0.258	0.255	0.828	0.841	0.405	0.384
Q-BLEU2	0.244	0.243	0.825	0.835	0.390	0.360
Q-BLEU3	0.239	0.240	0.824	0.837	0.374	0.331
Q-BLEU4	0.233	0.232	0.826	0.837	0.373	0.311
Q-ROUGE-L	0.253	0.249	0.821	0.841	0.402	0.385
Q-METEOR	0.158	0.157	0.821	0.837	0.402	0.378
Q-NIST	0.246	0.248	0.824	0.845	0.384	0.346

Table 6: Correlation between proposed Q-Metric and human judgments. All the correlations have a p-value < 0.01 and hence statistically significant.

any given weight (w_i) the standard deviation was very low across these k experiments. For each w_i and δ we obtained the final value by taking an average of the values learned in each of the k experiments. We also observed that the weights did not change much even when we used more data for tuning. Also note that we tuned these weights separately for each metric (*i.e.*, Q-BLEU4, Q-NIST, Q-METEOR and so on). For illustration, we report these weights for Q-BLEU1 metric in Table 5. As expected, the weights depend on the source from which the question was generated. Note that for WikiMovies, named entities have the highest weight. For VQA content words are most important, as they provide information about the entity being referred to in the question. Note that for SQuAD and VQA, the original base metric also gets weightage comparable to other components, indicating that a fluent question makes it easier to understand thus making it answerable. The overall trend for the values of w_i's was similar for other Q-METRICs also (*i.e.*, for Q-NIST, Q-METEOR and so on).

5.2 Correlation between Human scores and different Q-METRICs

Once the weights are tuned, we fix these weights and compute the Q-METRIC for the remaining 600-700 examples and report the correlation with human judgments for the same set of examples (see Table 6). For a fair comparison, the correlation scores reported in Table 4 are also on the same 600-700 examples. The correlation scores obtained for different Q-METRICs are indeed encouraging. In particular, we observe that while the correlation of existing metrics with noisy questions generated was very low (Table 4), the correlation of the modified metrics is much higher. This suggests that adding the learnable component for *answerability* and tuning its weights indeed leads to a better-correlated metric. Note that for VQA and SQuAD the correlations are not as high as human-human correlations, but the correlations are still statistically significant. We acknowledge that there is clearly scope for further improvement and the proposed metric is perhaps only a first step towards designing an appropriate metric for AQG. Hopefully, the human evaluation data released as a part of this work will help to design even better metrics for AQG.

5.3 Qualitative Analysis

We have listed some examples in Table 7, which highlight some strengths and weakness of the proposed Q-METRIC. We categorize examples as positive/negative depending on the similarity between human scores for answerability and the Q-BLEU score. For the examples marked as positive, the Q-BLEU score is very close to the *answerability* score given by humans.

6 Extrinsic evaluation

So far we have shown that existing metrics do not always correlate well with human judgments and it is possible to design metrics which correlate better with human judgments by including a learnable component to focus on *answerability*. We would now like to propose an extrinsic way of evaluating the usefulness of the proposed metric. The motivation for this extrinsic evaluation comes from the fact that one of the intended purposes of the modified metrics is to use them for training QA systems. Suppose we use a particular metric for evaluating the quality of an AQG system and suppose this metric suggests that the questions generated by this system are poor. We would obviously

Dataset		Original Question	Modified Question	Human Scores	QBLEU
SQuAD	**Positive**	What is another type of accountant other than a CPA?	What is another type of accountant other than a ?	0.10	0.47
		In addition to schools, where else is popularly based authority effective?	In addition schools, where else popularly based authority effective?	0.85	0.83
	Negative	When did Tesla begin working for the Continental Edison Company?	When did begin working for the Continental Edison Company?	0.10	0.84
		What famous person congratulated him?	What person congratulated him?	0.85	0.17
VQA	**Positive**	What color is the monster truck?	What color monster truck?	0.92	0.81
		What is in the polythene ?	What is in the ?	0.10	0.14
	Negative	Why there are no leaves on the tree?	Why are leaves the tree?	0.35	0.73
		How are the carrots prepared in the plate?	How carrots prepared plate?	0.10	0.68
WikiMovies	**Positive**	what films does Ralf Harolde appear in ?	what films Ralf Harolde appear ?	0.97	0.91
		what is a film directed by Eddie Murphy ?	Which a film directed by Eddie Murphy ?	0.91	0.88
	Negative	what films does Gerard Butler appear in ?	how does Gerard Butler appear in ?	0.15	0.89
		John Conor Brooke appears in which movies ?	appears in which movies ?	0.03	0.44

Table 7: Human (Gold) and Q-Metric scores for some of the examples from the collected human-evaluation data.

Type of Noise	BLEU	QBLEU	Hit 1
None	100	100	76.5
Stop Words	25.4	84.0	75.6
Question Type	74.0	79.3	73.5
Content Words	29.4	64.3	54.7
Named Entity	41.9	48.5	17.97

Table 8: Performance obtained by training on different types of noisy questions (WikiMovies).

Noise	BLEU	QBLEU	F1
None	100	100	76.5
Question Type	80.1	66.1	69.0
Stop Words	24.2	61.0	70.4
Content Words	60.7	57.1	64.1
Named Entity	77.0	56.0	73.8

Table 9: Performance obtained by training on different types of noisy questions (SQuAD).

Noise	BLEU	QBLEU	Acc(%)
None	100	100	64.4
Content Words	49.4	58.2	60.21
Question Type	63.7	50.9	59.81
Stop Words	10.8	37.7	57.37

Table 10: Performance obtained by training on different types of noisy questions (VQA).

discard this system and not use the questions generated by it to train a QA system. However, if the metric itself is questionable, then it is possible that the questions were good enough, but the metric was not good to evaluate their quality. To study this effect, we create a noisy version of the training data of SQuAD, WikiMovies, and VQA using the same methods outlined in Section 4.1. We then train a state of the art model for each of these tasks on this noisy data and evaluate the trained model on the original test set of each of these datasets. The models that we considered were (Seo et al., 2016) for SQuAD, (Miller et al., 2016) for WikiMovies and (Ben-younes et al., 2017) for VQA.

The results of our experiments are summarized in Table 8 - 10. The first column for each table shows the manner in which the noisy training data was created. The second column shows the BLEU4 score of the noisy questions when compared to the original reference questions (thus it tells us the perceived quality of these questions under the BLEU4 metric). We consider BLEU4 because of all the current metrics used for AQG it is the most popular. Similarly, the third column tells us the perceived quality of these questions under the Q-BLEU4 metric. Ideally, we would want that the performance of the model should correlate better with the perceived quality of the training questions as identified by a given metric. We observe that the general trend is better *w.r.t.* the Q-BLEU4 metric than the BLEU4 metric (*i.e.*, in general, higher Q-BLEU4 indicates better performance and lower Q-BLEU4 indicates poor performance). In particular, notice that BLEU4 gives much importance to stop words, but these words hardly have any influence on the final performance. We believe that such an extrinsic evaluation should also be used while designing better metrics and it would help us get better insights.

7 Conclusion

The main aim of this work was to objectively examine the utility of existing metrics for AQG. Specifically, we wanted to see if existing metrics account for the *answerability* of the generated questions. To do so, we took noisy generated questions from three different tasks, *viz.*, document QA, knowledge base QA and visual QA, and showed that the *answerability* scores assigned by humans did not correlate well with existing metrics. Based on these studies, we proposed a modification for existing metrics and showed that with the proposed modification these metrics correlate better with human judgments. The proposed modification involves learnable weights which can be tuned (depending on the source) using the human judgments released as a part of this work. Finally, we propose an extrinsic evaluation with the aim of assessing the end utility of these metrics in selecting good AQG systems for creating training data for QA systems. Though the proposed metric correlates better with human judgments, there is still scope for improvement especially for document QA and visual QA. As future work, we would like to design better metrics for answerability and check if a non-linear combination of different elements in the Q-Metric leads to better correlation with human judgments.

8 Acknowledgements

We would like to thank Google for supporting Preksha Nema through their Google India Ph.D. Fellowship Program. We would also like to express our gratitude to the volunteers who participated in human evaluations.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In International Conference on Computer Vision (ICCV).

Hedi Ben-younes, Rémi Cadene, Matthieu Cord, and Nicolas Thome. 2017. Mutan: Multimodal tucker fusion for visual question answering. In The IEEE International Conference on Computer Vision (ICCV). volume 1, page 3.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In Proceedings of the 20th international conference on Computational Linguistics. Association for Computational Linguistics, page 315.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. CoRR abs/1506.02075.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using amazon's mechanical turk. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '09, pages 286–295.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluation the role of bleu in machine translation research. In EACL. The Association for Computer Linguistics.

J Cohen. 1968. Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. Psychological bulletin 70(4):213220.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In Proceedings of the Second International Conference on Human Language Technology Research. HLT '02, pages 138–145.

Xinya Du and Claire Cardie. 2017. Identifying where to focus in reading comprehension for neural question generation. In EMNLP. Association for Computational Linguistics, pages 2067–2073.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In ACL (1). Association for Computational Linguistics, pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In EMNLP. Association for Computational Linguistics, pages 866–874.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In HLT-NAACL. The Association for Computational Linguistics, pages 609–617.

Unnat Jain, Ziyu Zhang, and Alexander G. Schwing. 2017. Creativity: Generating diverse questions using variational autoencoders. In CVPR. IEEE Computer Society, pages 5415–5424.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In CVPR. IEEE Computer Society, pages 1988–1997.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In ACL (1). Association for Computational Linguistics, pages 1601–1611.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In ACL (1). The Association for Computer Linguistics, pages 889–898.

Alon Lavie and Michael J. Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. Machine Translation 23(2-3):105–115.

Yikang Li, Nan Duan, Bolei Zhou, Xiao Chu, Wanli Ouyang, and Xiaogang Wang. 2017. Visual question generation as dual task of visual question answering. CoRR abs/1709.07192.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Proc. ACL workshop on Text Summarization Branches Out. page 10.

David Lindberg, Fred Popowich, John C. Nesbit, and Philip H. Winne. 2013. Generating natural language questions to support learning on-line. In ENLG. The Association for Computer Linguistics, pages 105–114.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In EMNLP. The Association for Computational Linguistics, pages 2122–2132.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1. ETMTNLP '02, pages 63–70.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In ACL (1). Association for Computational Linguistics, pages 1116–1126.

Mary L McHugh. 2012. Interrater reliability: the kappa statistic. Biochemia medica: Biochemia medica 22(3):276–282.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In EMNLP. The Association for Computational Linguistics, pages 1400–1409.

Jack Mostow and Wei Chen. 2009. Generating instruction automatically for the reading strategy of self-questioning. In AIED. IOS Press, volume 200 of Frontiers in Artificial Intelligence and Applications, pages 465–472.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016..

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In ACL. ACL, pages 311–318.

Ananthakrishnan R, Pushpak Bhattacharyya, M Sasikumar, and Ritesh M Shah. 2007. Some issues in automatic evaluation of english-hindi mt: More blues for bleu. In International Conference on Natural Language Processing.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016. pages 2383–2392.

Sathish Reddy, Dinesh Raghu, Mitesh M. Khapra, and Sachindra Josh. 2017. Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model. In EACL (1). Association for Computational Linguistics, pages 376–385.

Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. 2018a. Duorc: Towards complex language understanding with paraphrased reading comprehension. In ACL (1). Association for Computational Linguistics, pages 1683–1693.

Amrita Saha, Vardaan Pahuja, Mitesh M. Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018b. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In AAAI. AAAI Press.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. CoRR abs/1611.01603.

Iulian Vlad Serban, Alberto García-Durán, Çaglar Gülçehre, Sungjin Ahn, Sarath Chandar, Aaron C. Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In ACL (1). The Association for Computer Linguistics.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. CoRR abs/1706.02027.

Tong Wang, Xingdi Yuan, and Adam Trischler. 2017. A joint model for question answering and question generation. CoRR abs/1706.01450.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W. Cohen. 2017. Semi-supervised QA with generative domain-adaptive nets. CoRR abs/1702.02206.

Xingdi Yuan, Tong Wang, Çaglar Gülçehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. In Rep4NLP@ACL. Association for Computational Linguistics, pages 15–25.

Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, Jianfeng Lu, and Anton van den Hengel. 2017. Asking the difficult questions: Goal-oriented visual question generation via intermediate rewards. CoRR abs/1711.07614.

Stylistic Chinese Poetry Generation via Unsupervised Style Disentanglement

Cheng Yang[1,2,3], **Maosong Sun**[1,2,4]*, **Xiaoyuan Yi**[1,2,3], **Wenhao Li**[1,2,3]
[1]Department of Computer Science and Technology, Tsinghua University
[2]Institute for Artificial Intelligence, Tsinghua University
[3]State Key Lab on Intelligent Technology and Systems, Tsinghua University
[4]Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University
`{cheng-ya14,yi-xy16,liwh16}@mails.tsinghua.edu.cn`
`sms@tsinghua.edu.cn`

Abstract

The ability to write diverse poems in different styles under the same poetic imagery is an important characteristic of human poetry writing. Most previous works on automatic Chinese poetry generation focused on improving the coherency among lines. Some work explored style transfer but suffered from expensive expert labeling of poem styles. In this paper, we target on stylistic poetry generation in a fully unsupervised manner for the first time. We propose a novel model which requires no supervised style labeling by incorporating mutual information, a concept in information theory, into modeling. Experimental results show that our model is able to generate stylistic poems without losing fluency and coherency.

1 Introduction

Classical Chinese poetries are great heritages of the history of Chinese culture. One of the most popular genres of classical Chinese poems, *i.e. quatrains*, contains four lines with five or seven characters each and additional rhythm and tune restrictions. During 1,000 years history of quatrains, various styles, *e.g.* pastoral, descriptive and romantic, have been developed to express different feelings of poets. In human poetry writing, poets are able to write completely different poems in diverse styles even given the same keyword or first sentence. For example, as shown in Fig. 1, when a poet mentioned "月" (the moon), she/he may write about the Great Wall in the battlefield style or the sleepless feeling in the romantic style. Such ability to write stylistic poems under the same poetic imagery is an important characteristic of human poetries.

Automatic poetry generation is one of the first attempts towards computer writing. Chinese quatrain generation has also attracted much attention in recent years. Early works inspired by statistical machine translation explored rule-based and template-based methods (He et al., 2012; Yan et al., 2013), while recent works (Zhang and Lapata, 2014; Wang et al., 2016; Yan, 2016; Zhang et al., 2017; Yang et al., 2017; Yi et al., 2017) employed neural network based sequence-to-sequence approaches which have shown their effectiveness in neural machine translation for poem generation. Most works target on improving the coherency among all lines and the conformity between the theme and subsequent lines by planning (Wang et al., 2016), polishing schema (Yan, 2016), poem block (Yi et al., 2017) and conditional variational autoencoder (Yang et al., 2017). Different from these previous works, we aim to learn the ability of diverse stylistic poetry generation which can generate multiple outputs (poems) in various styles under the same input (keywords or the first sentence). This ability enables a poetry generation system to be closer to a real poet and allows the model to generate more expressive and creative poems.

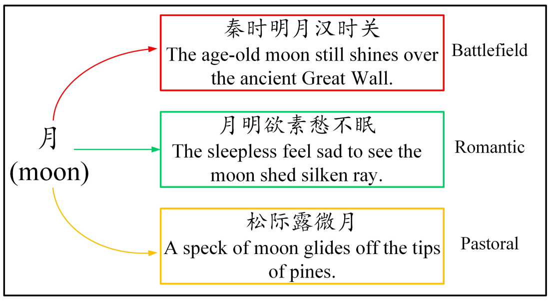

Figure 1: An example of poems in diverse styles under the same keyword.

However, there is no explicit label about what style or category a poem or a sentence is for thousands of poems in the database. Therefore, traditional supervised sequence-to-sequence mod-

---

* corresponding author: sms@tsinghua.edu.cn

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3960–3969
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

els (Zhang et al., 2017) are not capable to generate stylistic poems without expert labeling. To the best of our knowledge, we are the first effort at stylistic poetry generation in a fully unsupervised manner.

In this paper, we propose a novel poetry generation model which can disentangle the poems in different styles and generate style-specific outputs conditioned on the manually selected style input. We employ sequence-to-sequence model with attention mechanism (Bahdanau et al., 2014) as our basis and maximize the mutual information which measures the dependency between two random variables in information theory to strengthen the relationship between manually selected style inputs and generated style-specific outputs. Experimental results show that our model is able to generate poems in various styles without losing fluency and coherency.

To summarize, our effort provides the following three contributions:

- To the best of our knowledge, we are the first effort at stylistic poetry generation which is an important characteristic of human poetries in a fully unsupervised manner.
- We innovatively incorporate mutual information, a measurement in information theory, for unsupervised style disentanglement and style-specific generation.
- Experimental results show that our model is able to generate diverse poems in various styles without losing fluency and coherency.

2 Related Works and Motivations

Poetry generation is a classic task in computer writing (Gervás, 2001; Levy, 2001; Netzer et al., 2009; Oliveira, 2012). Chinese poetry generation has also attracted much attention during the last decade. Early works are mostly rule-based and template-based (Wu et al., 2009). (He et al., 2012) employed statistical machine translation and (Yan et al., 2013) adopted automatic summarization techniques for classical poem generation.

As neural network based approaches have been successfully applied in various applications such as machine translation (Bahdanau et al., 2014), people came up with the idea to apply sequence-to-sequence models for poem generation. (Zhang and Lapata, 2014) employed the Recurrent Neural Network (RNN) as their basis and further considered the global context using Convolutional Neural Network (CNN). They also incorporated other hand-crafted features into the model to improve the coherency. (Yan, 2016) proposed an iterative polishing schema based on two RNNs, which can refine the generated poems for several times. (Wang et al., 2016) generated poetries in a two-stage process: planning the keywords of each line first and then generating each line sequentially. (Yi et al., 2017) proposed poem blocks to learn semantic meaning within a single line and semantic relevance among lines in a poem. (Yang et al., 2017) employed a conditional variational autoencoder with augmented word2vec architecture to enhance the conformity between the theme and generated poems. All previous works above target on improving the coherency and conformity of poetry generation while a recent work (Zhang et al., 2017) was proposed to improve the novelty of generated poems using external memories. Their model can also transfer the style of a poem into three predefined topics. However, they need to manually label many poems in these three predefined topics to learn the patterns.

Formally, a traditional sequence-to-sequence model actually learns a conditional probability distribution $\Pr(s_{output}|s_{input})$ where s_{output} is the generated poems and s_{input} is the input keyword or the first sentence. Different from machine translation tasks where the output sentence s_{output} is rather certain and compact given input sentence s_{input}, the conditional probability distribution $\Pr(s_{output}|s_{input})$ has strong uncertainty in literary creation scenarios such as poem generation. To support this point, we further train a topic model based on LDA (Blei et al., 2003) by treating each poem as a document. We train a 10-topic and a 20-topic LDA model and then reanalyze the largest topic component of each sentence in a poem. Surprisingly, we find that only 20% and 10% consecutive sentences in a poem have the same largest topic components respectively though we assume that each poem is generated by the same topic when training the LDA model. The fact indicates that even given the former sentence, the style of the latter one could still be diverse and flexible. Intuitively, poets will choose a style or topic in their minds and then write the next sentence based on the style they choose in poetry creation. Therefore, we propose to learn a stylistic poetry generation model which can disentangle the poems in different styles and generate style-specific outputs as real human poets could do. Note that no explicit style labels are given to the training data and thus previous supervised algorithms cannot be adapted for the purpose easily.

Recently, a few works have been proposed for disentangled representation learning. InfoGAN (Chen et al., 2016) was proposed for generating continuous image data conditioned on image labels and disentangled text generation (Hu et al., 2017) focused on controllable generation in the semi-supervised setting. Though inspired by them, the motivation and proposed models of our work differ from these methods by a large margin. To the best of our knowledge, we are the first effort at stylistic poetry generation in a fully unsupervised manner.

3 Method

We hope our generation model can generate multiple outputs in various styles. Formally, our model takes two arguments as input: input sentence s_{input} and style id $k \in 1, 2 \ldots K$ where K is the total number of different styles. Then we can enumerate each style id k and generate style-specific output sentence s^k_{output} respectively.

In this section, we will start by introducing the background knowledge about mutual information and sequence-to-sequence model with attention mechanism. Then we propose our framework of decoder model for style disentanglement by taking the mutual information as an additional regularization term. Finally, we will present the details of our implementations of each component in the framework.

3.1 Mutual Information

Inspired by previous works on image generation (Chen et al., 2016) and semi-supervised generation (Hu et al., 2017), we propose to incorporate the concept of mutual information in information theory for style disentanglement. Given two random variables X and Y, the mutual information $I(X, Y)$ measures "the amount of information" obtained about one random variable given another one [1]. Mutual information can also be interpreted as a measurement about how similar the joint probability distribution $p(X, Y)$ is to the product of marginal distributions $p(X)p(Y)$. The definition of mutual information is

$$I(X, Y) = \int_Y \int_X p(X, Y) \log \frac{p(X, Y)}{p(X)p(Y)} dX dY. \quad (1)$$

3.2 Sequence-to-Sequence Model with Attention Mechanism

We employ a widely used Encoder-Decoder framework (Sutskever et al., 2014) which was firstly proposed in machine translation as our basis. Suppose sentence $X = (x_1 x_2 \ldots x_T)$ and $Y = (y_1 y_2 \ldots y_T)$ are the input sentence (source sentence) and output sentence (target sentence) respectively where x_i, y_i for $i = 1, 2 \ldots T$ are characters and T is the total number of characters in the sentence. We denote the character vocabulary as V.

Specifically, we use bidirectional LSTM (biLSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) model as the encoder to project the input sentence X into the vector space. Formally, the hidden state of LSTM are computed by

$$\overrightarrow{h_i} = LSTM_{forward}(\overrightarrow{h_{i-1}}, e(x_i)), \quad (2)$$

$$\overleftarrow{h_i} = LSTM_{backward}(\overleftarrow{h_{i-1}}, e(x_{T-i+1})), \quad (3)$$

for $i = 1, 2 \ldots T$ where $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ are the i-th hidden state of forward and backward LSTM respectively, $e(x_i) \in \mathcal{R}^d$ is the character embedding of character x_i and d is the dimension of character embeddings. Then we concatenate corresponding hidden states of forward and backward LSTM $h_i = [\overrightarrow{h_i}, \overleftarrow{h_{T-i+1}}]$ as the i-th hidden state of biLSTM. In specific, we use the last hidden state h_T as the embedding vector and feed it to the decoder.

The decoder module contains an LSTM decoder with attention mechanism (Bahdanau et al., 2014) which computes a context vector as a weighted sum of all encoder hidden states to represent the most relevant information at each stage. The character probability distribution when decoding the i-th character can be expressed as

$$p(y_i | y_1 y_2 \ldots y_{i-1}, X) = g(y_i | s_i), \quad (4)$$

where $g(\cdot)$ is a linear projection function with softmax regularization, s_i is the i-th hidden state in the decoder LSTM:

$$s_i = LSTM_{decoder}(s_{i-1}, [e(y_{i-1}), a_i]), \quad (5)$$

for $i = 2, \ldots T$ and $s_1 = h_T$, where $[\,]$ indicates concatenation operation, $e(y_{i-1})$ is character embedding of y_{i-1} and a_i is the context vector

---

[1] `https://en.wikipedia.org/wiki/Mutual_information`

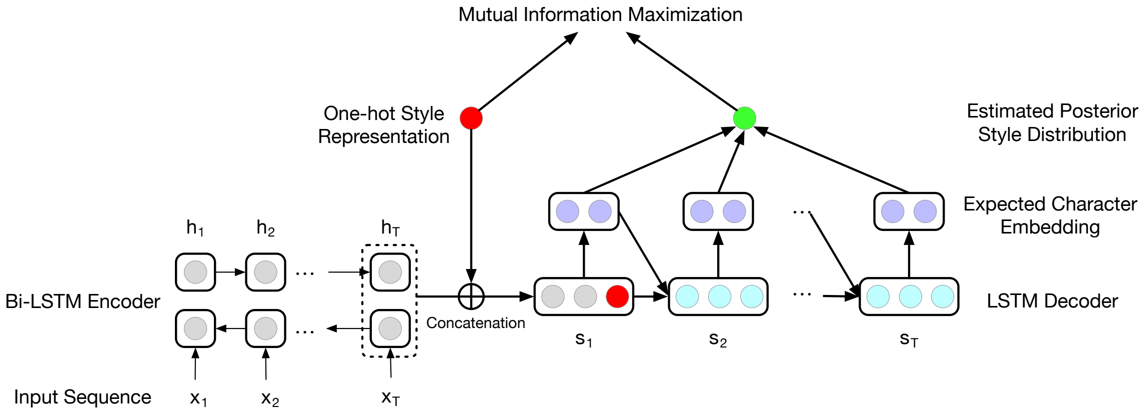

Figure 2: An overview of style disentanglement by mutual information maximization.

learned by attention mechanism (Bahdanau et al., 2014)

$$a_i = attention(s_{i-1}, h_{1:T}). \tag{6}$$

3.3 Decoder Model with Style Disentanglement

To accept two arguments, *i.e.*, input sentence X and style id k, we can directly concatenate the one-hot representation of style id $one\_hot(k)$ and the embedding vector h_T obtained by bi-LSTM encoder and then feed the concatenated vector $[one\_hot(k), h_T]$ instead of h_T into the decoder model without changing anything else.

However, there is no theoretical guarantee that the output sentence generated by the decoder is strongly correlated to the style id input $one\_hot(k)$. In other words, when the input style id is changed, the output sentence would probably be the same because no supervised loss is given to force the output sentences to follow the one-hot style representation. Therefore, we naturally come up with the idea to add a regularization term to force a strong dependency relationship between the input style id and generated sentence.

Without loss of generality, we assume that the input style id is a uniformly distributed random variable Sty and $\Pr(Sty = k) = \frac{1}{K}$ for $k = 1, 2 \ldots K$ where K is the total number of styles. Recall that mutual information quantifies the mutual dependency between two random variables. Hence we propose to maximize the mutual information between the style distribution $\Pr(Sty)$ and the generated sentence distribution $\Pr(Y; X)$ given input sentence X to strengthen the dependency between them as shown in Fig. 2. The mutual information is computed as

$$\begin{aligned}
&I(\Pr(Sty), \Pr(Y;X)) \\
=&\sum_{k=1}^{K}\Pr(Sty=k)\int_{Y|k;X}\log\frac{\Pr(Y,Sty=k;X)}{\Pr(Sty=k)\Pr(Y;X)}dY \\
=&\sum_{k=1}^{K}\Pr(Sty=k)\int_{Y|k;X}\log\frac{\Pr(Y,Sty=k;X)}{\Pr(Y;X)}dY \\
&-\sum_{k=1}^{K}\Pr(Sty=k)\log\Pr(Sty=k) \\
=&\sum_{k=1}^{K}\Pr(Sty=k)\int_{Y|k;X}\log\Pr(Sty=k|Y)dY+\log K \\
=&\int_{Y;X}\sum_{k=1}^{K}\Pr(Sty=k|Y)\log P(Sty=k|Y)dY+\log K.
\end{aligned} \tag{7}$$

Note that the input sequence X and style Sty are both input arguments and thus independent with each other. Therefore, posterior distribution $\Pr(Sty = k|Y; X) = \Pr(Sty = k|Y)$.

But the posterior probability distribution $\Pr(Sty = k|Y)$ is actually unknown, we cannot compute the integration directly. Fortunately, with the help of variational inference maximization (Barber and Agakov, 2003), we can train a parameterized function $Q(Sty = k|Y)$ which estimates the posterior distribution and maximize a lower bound of mutual information in Eq. 7 instead:

$$\begin{aligned}
&I(\Pr(Sty), \Pr(Y;X)) - \log K \\
=&\int_{Y;X}\sum_{k=1}^{K}\Pr(Sty=k|Y)\log\Pr(Sty=k|Y)dY \\
=&\int_{Y;X}\sum_{k=1}^{K}\Pr(Sty=k|Y)\log Q(Sty=k|Y)dY \\
&+\int_{Y;X}\sum_{k=1}^{K}\Pr(Sty=k|Y)\log\frac{\Pr(Sty=k|Y)}{Q(Sty=k|Y)}dY
\end{aligned}$$

$$
\begin{aligned}
&= \int_{Y;X} \sum_{k=1}^{K} \Pr(Sty = k|Y) \log Q(Sty = k|Y) dY \\
&\quad + \int_{Y;X} KL(\Pr(Sty|Y), Q(Sty|Y)) dY \\
&\geq \int_{Y;X} \sum_{k=1}^{K} \Pr(Sty = k|Y) \log Q(Sty = k|Y) dY \\
&= \sum_{k=1}^{K} \Pr(Sty = k) \int_{Y|k;X} \log Q(Sty = k|Y) dY.
\end{aligned} \tag{8}
$$

Here $KL(\Pr(\cdot)||Q(\cdot))$ indicates the KL-divergence distance between probability distribution $\Pr(\cdot)$ and $Q(\cdot)$. The inequality comes from the fact that the KL-divergence is always no less than zero and is tight when $\Pr(\cdot) = Q(\cdot)$.

We use the lower bound parameterized by function Q in Eq. 8 as an additional maximization term. Intuitively, the lower bound is maximized if we can perfectly infer the style id from generated style-specific outputs by inference function Q. It also indicates that generated outputs will heavily depend on the input style id and outputs generated by different styles are distinguishable, which follows our motivation of stylistic poetry generation. Now we will introduce how to design the posterior distribution estimation function Q and compute the integration in the lower bound.

3.4 Posterior Distribution Estimation

Given an output sequence Y, the function Q estimates the probability distribution of the style of sequence Y and therefore disentangles different styles. In this paper, we employ neural network to parametrize the posterior distribution estimation function Q. Specifically, we first compute the average character embeddings of sequence Y and then use a linear projection with softmax normalizer to get the style distribution. Formally, $Q(Sty|Y)$ is computed as

$$
Q(Sty|Y) = \mathit{softmax}(W \cdot \frac{1}{T} \sum_{i=1}^{T} e(y_i)), \tag{9}
$$

where $W \in \mathcal{R}^{K \times d}$ is the linear projection matrix.

Then the last thing to do is to compute the integration over $Y|k;X$. However, the search space of sequence Y is exponential to the size of vocabulary. Hence it's impossible to enumerate all possible sequence Y for computing the integration. Also, it is not differentiable if we sample sequence Y according to generation probability for approximation. Therefore, we propose to use expected character embedding to approximate the integration.

3.5 Expected Character Embedding

Inspired by previous works (Kočiský et al., 2016), we use expected character embedding to approximate the probabilistic space of output sequences: we only generate an expected embedding sequence and suppose $Y|k;X$ has one hundred percent probability generating this one. Formally, Eq. 4 gives a probability distribution of generating the i-th character given previous ones. Then the "expected" generated character embedding is

$$
\mathit{expect}(i;k,X) = \sum_{c \in V} g(c|s_i) e(c), \tag{10}
$$

where $\mathit{expect}(i;k,X) \in \mathcal{R}^d$ represents the expected character embedding at i-th output given style id k and input sequence X and $c \in V$ enumerates all characters in the vocabulary.

Then $\mathit{expect}(i;k,X)$ is fed into the LSTM in decoder to update the hidden state for generating next expected character embedding:

$$
s_{i+1} = LSTM_{decoder}(s_i, [\mathit{expect}(i;k,X), a_{i+1}]). \tag{11}
$$

Finally, we use the expected embeddings $\mathit{expect}(i;k,X)$ for $i = 1, 2 \ldots T$ as an approximation of the whole probability space of $Y|k;X$. The lower bound in Eq. 8 can be rewritten as

$$
\mathcal{L}_{reg} = \frac{1}{K} \sum_{k=1}^{K} \log\{\mathit{softmax}(W \cdot \frac{1}{T} \sum_{i=1}^{T} \mathit{expect}(i;k,X))[k]\}, \tag{12}
$$

where $x[j]$ represents the j-th dimension of vector x.

We add the lower bound $\mathcal{L}_{reg}$ to the overall training objective as a regularization term. The computing process is shown in Fig. 2. Then for each training pair (X,Y), we aim to maximize

$$
\mathit{Train}(X,Y) = \sum_{i=1}^{T} \log p(y_i|y_1 y_2 \ldots y_{i-1}, X) + \lambda \mathcal{L}_{reg}, \tag{13}
$$

where $p(y_i|y_1 y_2 \ldots y_{i-1}, X)$ is style irrelevant generation likelihood and computed by setting one-hot style representation to an all-zero vector, and λ is a harmonic hyperparameter to balance the log-likelihood of generating sequence Y and the lower bound of mutual information. The first term ensures that the decoder can generate fluent and coherent outputs and the second term guarantees the style-specific output has a strong dependency on the one-hot style representation input.

Note that unlike machine translation task where the trained model is desired to generate exactly the same as the target, poetry generation encourages novelty as an important requirement. In Eq. 13, we can also enumerate the style id representation as a one-hot vector instead of an all-zero one. However, this will force the generation of all styles to be close to the training target and discourage the diversity and novelty.

Moreover, our model is not task-specific: the regularization term $\mathcal{L}_{reg}$ can be added to other models conveniently for diverse or stylistic generations. A Chinese quatrain has at most 4*7=28 characters and clear rhyme requirements. As the first attempt on unsupervised style disentanglement, we find Chinese poetry generation is an ideal choice for evaluation because we can better focus on stylistic generation rather than dealing with genre requirements. We will explore the applicability of our model on other languages and tasks for future work.

4 Experiments

Following the experimental settings in previous works (Zhang et al., 2017), we conduct human judgment to evaluate the performance of our model and state-of-the-art baseline methods. We will first introduce the dataset, our model details, baseline methods and evaluation settings. Then we will present the experimental results and give further analysis about the evaluation. Finally, we will present some example generations for case study.

4.1 Dataset and Model Details

We collect 168,000 poems (half *wuyan*: five characters per line, half *qiyan*: seven characters per line) over 1,000 years history of classical Chinese poetries as our corpus. We randomly select 80% of the corpus as training set, 10% as validation set and leave the rest for test. We extract all consecutive sentences in a poem as training pairs (X, Y) and feed them into our model.

For hyperparameter settings of our model, *i.e.* stylistic poetry generation (SPG), we set the dimensions of character embeddings and encoder hidden states as $d = 512$. The total number of styles is set to $K = 10$. Hence the dimension of decoder hidden states is $512 + 512 + 10 = 1034$. We pack the training pairs to mini-batches and the size of mini-batches is set to 50. The harmonic hyperparameter is set to $\lambda = 0$ for the first $50,000$ mini-batches as pretraining and $\lambda = 1.0$ for subsequent batches. We use Adam optimizer (Kingma and Ba, 2014) for stochastic gradient descent. We also employ dropout (Srivastava et al., 2014) strategy with dropout rate 0.2 to avoid overfitting problem. We terminate the training process when the optimization loss on validation set is stable, *i.e.* $300,000$ mini-batches in total.

4.2 Baselines

We consider the following state-of-the-art poetry generation models for comparison:

- **Seq2seq** (Bahdanau et al., 2014) is the sequence-to-sequence model with attention mechanism. Note that seq2seq is the basis of our SPG model. We can better analyze the improvement by style disentanglement from the comparison with seq2seq.
- **Polish** (Yan, 2016) proposed an iterative schema to polish and refine the generated sentences for several times instead of a one-pass generation.
- **Memory** (Zhang et al., 2017) incorporated external memory into poem generation for better novelty. The memory can be seen as a regularization to constrain the behavior of the neural model.

Rule-based and template-based methods are not considered as our baselines as they have been already thoroughly compared in (He et al., 2012; Yan, 2016).

4.3 Evaluation Settings

Following the settings in (Zhang et al., 2017), we employ human judgment for evaluation. To better compare the ability to generate fluent and coherent poems, we fix the first sentence as input and let all models generate three subsequent sentences. The first sentences are randomly chosen from the poems in the test set. Therefore we also consider the original poems written by real poets for comparison.

Note that our SPG model need a manually specified style id as input. For fully automatic poem generation, we use the posterior style estimation function $Q(\cdot)$ to infer the style of the first sentence and then generate next three sentences sequentially using the same style.

As previous works (Manurung, 2004; Yi et al., 2017) did, we design four criteria for human judgment: **Fluency** (are the generated poems fluent and well-formed?), **Coherence** (is the topic of the whole quatrain consistent?), **Meaningfulness**

Group	Methods	Fluency		Coherence		Meaningfulness		Poeticness	
		wuyan	qiyan	wuyan	qiyan	wuyan	qiyan	wuyan	qiyan
Group 1	seq2seq	2.75	2.48	2.60	2.33	2.38	2.15	2.40	2.35
	SPG	**3.43**	**3.23**	**3.13**	**3.05**	**2.83**	**3.10**	**3.10**	**3.10**
Group 2	memory	2.50	2.60	2.30	2.38	2.18	2.25	2.05	2.40
	polish	2.53	2.90	2.28	2.55	2.15	2.55	2.20	2.50
	SPG	**3.38**	**3.53**	**3.38**	**3.30**	**3.13**	**3.25**	**3.20**	**3.20**
	human poet	3.85	3.68	4.05	3.83	4.00	3.83	3.53	3.75

Table 1: Human judgment results. We bold the best performing model in each group.

(does the poem convey some certain messages?) and **Poeticness** (does the poem have some poetic features?). Each criterion needs to be scored on a 5-point scale ranging from 1 to 5.

For each model, we generate 20 *wuyan* and 20 *qiyan* quatrains given the first sentence. We invite 10 experts who major in Chinese literature or are members of a poetry association to evaluate these quatrains. We divide the baseline methods into two groups: In the first group, we compare SPG with seq2seq (Bahdanau et al., 2014) to present the advantages of style disentanglement; In the second group, we compare SPG with state-of-the-art poetry generation methods, memory (Zhang et al., 2017) and polish (Yan, 2016), and the original poems written by poets to demonstrate the effectiveness of our algorithm. Note that the scores of human judgment are relative, not absolute. Thus we compare with seq2seq separately to avoid mutual interference.

4.4 Experimental Results

We report the average scores of expert judgments in Table 1. From the experimental results, we have the following observations:

Firstly, the experimental results in group 1 demonstrate that SPG outperforms seq2seq, the basis of our model, by learning a style-disentangled generation model. The only difference between seq2seq and our model is that we append a one-hot style id to the encoder state and add mutual information regularization in Eq. 12 to the loss function. Note that our model is fully unsupervised: the one-hot style id conveys no meaningful message unless the mutual information regularization is considered. In other words, the one-hot style id and mutual information regularization cannot be torn apart. Therefore, the improvements over seq2seq all come from the part of style disentanglement modeling because seq2seq and our model share all the other components. The comparisons in group 1 of Table 1 are sufficient to show the effectiveness of style modeling.

Secondly, the experimental results in group 2 show that SPG consistently outperforms two state-of-the-art poem generation methods. SPG is able to generate poetries in diverse styles without losing fluency and coherency. Our advantages are two-fold: On one hand, SPG better fits the diversity characteristic of human poetry writing. Intuitively, seq2seq learns a generation model that mixes poems in various styles and is more likely to generate meaningless common sentences. By disentangling poems in different styles, our model can learn a more compact generation model for each style and write more fluent and meaningful poems. On the other hand, by fixing the style when generating three lines in a quatrain, SPG is able to generate more coherent poems.

One can also imagine that the generation probability distribution actually consists of several peaks where each peak identifies a cluster of poems in similar styles. A traditional model mixes all the stuff together and learns a one-peak generation model which is more likely to generate meaningless common sentences. In contrast, our model disentangles different clusters (styles) and learns a more accurate and compact generation model for each cluster (style). Hence our model can generate poems with higher quality and beat the baselines in terms of human evaluation. This observation demonstrates the effectiveness and robustness of our model.

Finally, there is still a large margin between SPG and human poets. In this paper, we focus on poem generation in diverse styles. Other poetry characteristics are also important for the quality of generations. We will consider applying our style-disentangle regularization on other poem models for better performance in the future.

Style id	Keywords
1	loneliness, melancholy
2	the portrait of landscape
3	sorrow during roaming
4	hermit, rural scenes
5	grand scenery, regrets about old events
6	sorrow during drinking
7	emotions towards life experience
8	the portrait of hazy sceneries
9	reminiscence, homesickness
10	sadness about seasons

Table 2: Representative keywords for poems generated by each learned style.

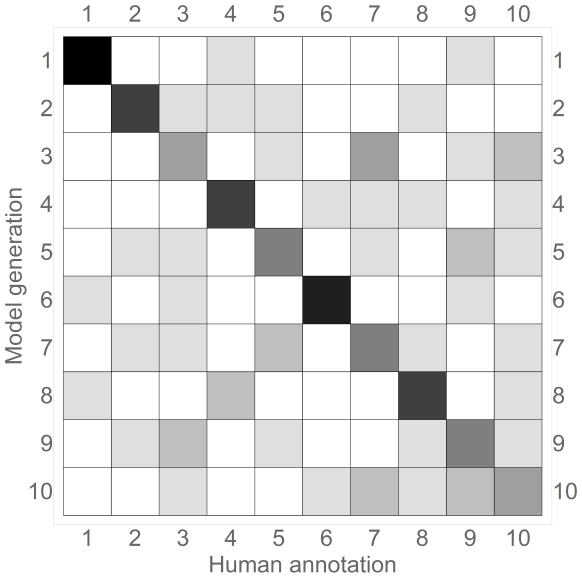

Figure 3: Experimental results on style recognition. Each row represents the human annotation of corresponding style generations. The diagonal blocks are correct classifications. Darker block indicates higher probability.

4.5 Interpretability of Learned Styles

In this subsection, we conduct experiments to understand the semantic meanings of each learned style. To this end, we first generate 50 poems in each style and then manually select two or three keywords to sketch the overall feelings of every style. The keywords of all 10 styles are listed in Table 2. Note that the learned styles may not strictly align the traditional writing styles recognized by human such as romantic or pastoral because our model is fully unsupervised.

Then we generate another 50 poems (5 poems per style) and ask the experts to classify these poems into the 10 styles according to the relationship between generated poems and style keywords in Table 2. The human annotation results are shown in Fig. 3.

We can see that many learned styles can be successfully recognized by human with a higher probability, e.g. the first style can be correctly classified by 80% probability. This observation indicates that the learned styles of SPG are meaningful and recognizable through only two or three keywords. Also, the generated poems are diverse otherwise they cannot be differentiated and correctly classified.

4.6 Case Study

We present three poems generated by SPG with the same first sentence for case study. We only list the results of three most representative styles in Fig. 4 and put other generation examples in supplementary materials (in Chinese). We can see that the poems generated by different style inputs differ a lot from each other and follow the style keywords. To conclude, SPG can generate fluent and coherent poetries in diverse styles.

5 Conclusion

In this paper, we propose a stylistic poetry generation (SPG) model to learn the ability to write poems in different styles under the same poetic imagery. To the best of our knowledge, we are the first effort at stylistic poetry generation in a fully unsupervised manner. Therefore, our model requires no expensive expert style annotation for thousands of poems in the database. We innovatively employ mutual information, a concept in information theory, to develop our model. Experimental results show that SPG is able to generate fluent and coherent poetries in diverse styles without losing fluency and coherency. The learned styles are meaningful and recognizable given only two or three keywords. Our algorithm has been incorporated into Jiuge[2] poetry generation system.

For future works, we will consider adopting the mutual information regularization for other text generation tasks which encourage stylistic generation or diversity to improve their performances. Another intriguing direction is to refine our model for more task-specific scenarios. Besides, our model simplifies the prior of style distribution as a uniform distribution. This assumption could be further improved by an iteratively updating approach.

---

[2] https://jiuge.thunlp.cn

浊酒一杯聊酩酊，
After a cup of unstrained wine,
I have been a little drunk
白云千里断鸿濛。
I saw the cloud split the sky apart.
马蹄踏破青山路，
On horseback, I pass through every road
across the mountain,
惆怅斜阳落日红。
but can only watch the red sun falling down
with sorrow.

(a) Style 1: “loneliness, melancholy”

浊酒一杯聊酩酊，
After a cup of unstrained wine,
I have been a little drunk
扁舟何处问渔樵。
With a narrow boat, where could I find
the hermits?
行人莫讶归来晚，
Friends, don't be surprised that I come
back so late,
万里春风到海潮。
I have seen the great tide and the grand
spring breeze.

(b) Style 4: “hermit, rural scenes”

浊酒一杯聊酩酊，
After a cup of unstrained wine,
I have been a little drunk
浮云何处觅仙踪。
I wonder on which cloud I can see the
presence of the gods.
迢迢十二峰头月，
The moon above the mount seems
farther and farther.
漠漠千山暮霭浓。
The mist among the hill becomes
thicker and thicker.

(c) Style 8: “the portrait of hazy sceneries”

Figure 4: Examples generated by style 1,4 and 8 given the same first sentence. The keywords of the three styles are listed for convenience.

Acknowledgments

We would like to thank Chongxuan Li, Huimin Chen, Jiannan Liang, Zhipeng Guo and anonymous reviewers for their insightful comments. This research is funded by the National 973 project (No. 2014CB340501). It is also partially supported by the NExT++ project, the National Research Foundation, Prime Minister's Office, Singapore under its IRC@Singapore Funding Initiative.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

David Barber and Felix Agakov. 2003. The im algorithm: a variational approach to information maximization. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, pages 201–208. MIT Press.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Neural Information Processing Systems*.

Pablo Gervás. 2001. An expert system for the composition of formal spanish poetry. *Knowledge-Based Systems*, 14(3-4):181–188.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating chinese classical poems with statistical machine translation models. In *AAAI*.

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Controllable text generation. *arXiv preprint arXiv:1703.00955*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Ling Wang, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087.

Robert P Levy. 2001. A computational model of poetic creativity with neural network as measure of adaptive fitness. In *Proceedings of the ICCBR-01 Workshop on Creative Systems*.

Hisar Manurung. 2004. An evolutionary algorithm approach to poetry generation.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku: Generating haiku with word associations norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 32–39. Association for Computational Linguistics.

Hugo Gonçalo Oliveira. 2012. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016. Chinese poetry generation with planning based neural network. *arXiv preprint arXiv:1610.09889*.

Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. 2009. New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. In *International Conference on Entertainment Computing*, pages 191–196. Springer.

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *IJCAI*, pages 2238–2244.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 2013. i, poet: Automatic chinese poetry composition through a generative summarization framework under constrained optimization. In *IJCAI*, pages 2197–2203.

Xiaopeng Yang, Xiaowen Lin, Shunda Suo, and Ming Li. 2017. Generating thematic chinese poetry with conditional variational autoencoder. *arXiv preprint arXiv:1711.07632*.

Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. 2017. Generating chinese classical poems with rnn encoder-decoder. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 211–223. Springer.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative chinese poetry generation using neural memory. *arXiv preprint arXiv:1705.03773*.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

Generating More Interesting Responses in Neural Conversation Models with Distributional Constraints

Ashutosh Baheti[1], **Alan Ritter**[1], **Jiwei Li**[2], **and Bill Dolan**[3]

[1]Ohio State University, OH, USA
[2]Shannon.AI and Renmin University of China, Beijing, China
[3]Microsoft Research, Redmond, WA, USA
`{baheti.3,ritter.1492}@osu.edu`
`jiweil@stanford.edu`
`billdol@microsoft.com`

Abstract

Neural conversation models tend to generate safe, generic responses for most inputs. This is due to the limitations of likelihood-based decoding objectives in generation tasks with diverse outputs, such as conversation. To address this challenge, we propose a simple yet effective approach for incorporating side information in the form of distributional constraints over the generated responses. We propose two constraints that help generate more content rich responses that are based on a model of syntax and topics (Griffiths et al., 2005) and semantic similarity (Arora et al., 2016). We evaluate our approach against a variety of competitive baselines, using both automatic metrics and human judgments, showing that our proposed approach generates responses that are much less generic without sacrificing plausibility. A working demo of our code can be found at `https://github.com/abaheti95/DC-NeuralConversation`.

1 Introduction

Recent years have seen growing interest in neural generation methods for data-driven conversation. This approach has the potential to leverage massive conversational datasets on the web to learn *open-domain* dialogue agents, without relying on hand-written rules or manual annotation. Such response generation models could be combined with traditional dialogue systems to enable more natural and adaptive conversation, in addition to new applications such as predictive response suggestion (Kannan et al., 2016), however many challenges remain.

A major drawback of neural conversation generation is that it tends to produce too many "safe" or generic responses, for example: *"I don't know"* or *"What are you talking about ?"*. This is a pervasive problem that has been independently reported by multiple research groups (Li et al., 2016a; Serban et al., 2016; Li et al., 2016c).[1] The effect is due to the use of conditional likelihood as a decoding objective – maximizing conditional likelihood is a suitable choice for text-to-text generation tasks such as machine translation, where the source and target are semantically equivalent, however, in conversation there are many acceptable ways to respond. Simply choosing most predictable reply often leads to very dull conversation.

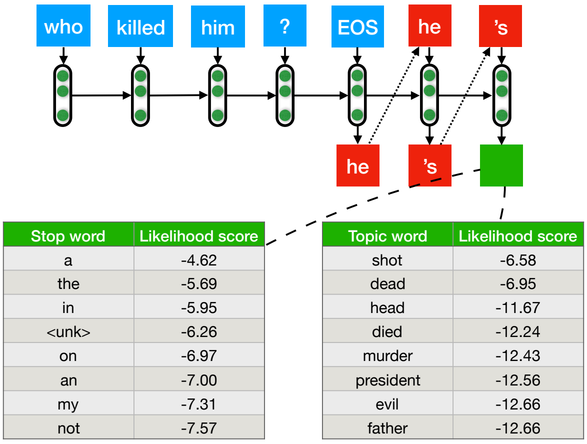

Stop word	Likelihood score
a	-4.62
the	-5.69
in	-5.95
<unk>	-6.26
on	-6.97
an	-7.00
my	-7.31
not	-7.57

Topic word	Likelihood score
shot	-6.58
dead	-6.95
head	-11.67
died	-12.24
murder	-12.43
president	-12.56
evil	-12.66
father	-12.66

Figure 1: Illustration of the dull response problem in maximum likelihood neural conversation generation using an example from the OpenSubtitles corpus. Function (stop) words tend to receive higher log probabilities than content (topic) words. The highest likelihood stop words and topic words in this context are listed.

Figure 1 illustrates the problem with conditional likelihood using an example. After encoding the source message using a bidirectional LSTM with attention, and fixing the first two words of the response, we show the highest ranked words (according to log-likelihood scores) taken from a list

---

[1] `https://research.googleblog.com/2015/11/computer-respond-to-this-email.html`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3970–3980
Brussels, Belgium, October 31 - November 4, 2018.

of stop words[2] in contrast to those selected from a list of topic words.[3] As illustrated in the figure, response generation that is based on maximum likelihood is biased towards stop-words and therefore results in responses that are *safe* (likely to be plausible in the context of the input), but also *bland* (don't contribute any new information to the conversation). This motivates the need for augmenting the decoding objective to encourage the use of more content words.

To address the *dull-response* problem in neural conversation, in this paper, we propose a new decoding objective that flexibly incorporates side-information in the form of distributional constraints. We explore two constraints, one which encourages the distribution over topics and syntax in the response to match that found in the user's input. To estimate these distributions, we leverage the unsupervised model of topics and syntax proposed by Griffiths and Steyvers (2005). The second constraint encourages generated responses to be semantically similar to the user's input; semantic similarity is measured using fixed-dimensional sentence embeddings (Arora et al., 2016).

After introducing distributional constraints into the decoding objective, we empirically demonstrate, in an evaluation that is based on human judgments, that our approach generates more content-rich responses when compared with two competitive baselines: Maximum Mutual Information (MMI) (Li et al., 2016a), in addition to an approach that conditions on topic models as additional context in neural conversation (Xing et al., 2017). While encouraging the model to generate less bland responses can be risky, we find that our approach achieves comparable plausibility while introducing significantly more content.

2 Neural Conversation Generation

As a starting point for our approach we leverage the Seq2Seq model (Sutskever et al., 2014; Bahdanau et al., 2014) which has been used as a basis for a broad range of recent work on neural conversation (Kannan et al., 2016; Li et al., 2016a; Serban et al., 2016; Shao et al., 2017). This model consists of two parts, an *encoder* and a *decoder* both of which are typically stacked LSTM layers. The encoder reads the input sequence and creates a hidden representation. The decoder conditions on this representation, using attention, and generates the response using a neural network language model (Bengio et al., 2003; Sutskever et al., 2011).

3 Distributional Topic and Semantic Similarity Constraints

Neural generation models select a response, $\hat{Y}$ by maximizing over a decoding objective, typically using greedy beam search from left to right over partially completed responses, which are scored using the decoder RNN language model. A commonly used decoding objective is the conditional likelihood of the target given the source, $P(Y|X)$:

$$\begin{aligned} \hat{Y} &= \arg\max_{Y} \{\log P(Y|X)\} \qquad (1) \\ &= \arg\max_{w_1,\ldots,w_n} \{\sum_{i=1}^{n} \log P(w_i|w_1,\ldots w_{i-1},X)\} \end{aligned}$$

As discussed in Section 1, models trained to maximize conditional likelihood tend to assign low probability to content words as compared to (more frequent) function words, leading to bland, generic responses most of the time. To ameliorate this, we introduce distributional constraints in the form of additional terms in the decoding objective that favor hypotheses containing more content words that are similar to the source in the *Topical* and *Semantic* sense.

For the constraint in the topic domain, we are interested in the topic probability distributions of the source, X, and target Y, $P(T|X)$ and $P(T|Y)$, where T is a random variable defined over k topics. Then we can modify the decoding objective from Eq 1:

$$\begin{aligned} \hat{Y}^T = \arg\max_{Y} \{ &\log P(Y|X) + \qquad (2) \\ &\alpha \times \Delta(P(T|X), P(T|Y))\} \end{aligned}$$

Here, Δ is a similarity function between the two probability distributions and α is a tunable hyper-parameter to adjust impact of this constraint.

Much recent work has investigated how to encode the semantic meaning of a sentence into a fixed high dimensional embedding space (Kiros et al., 2015; Wieting and Gimpel, 2017). Given such an embedding representation of X and Y, one can find the semantic similarity between the two and similar to Eq 2 we can add a semantic similarity constraint to the likelihood objective as

---

[2] `https://www.ranks.nl/stopwords`

[3] The top 10 topic words were taken from each of the 50 topics inferred by an HMM-LDA model (after removing stop words).

follows:

$$\hat{Y}^{Emb} = \arg\max_{Y}\{\log(P(Y|X)) + \beta \times \nabla(Emb(X), Emb(Y))\} \tag{3}$$

where, $Emb()$ is a function that maps an utterance to a semantic vector representation, ∇ is a function that computes similarity of the two embeddings and β is a tunable parameter.

Both of the constraint terms from Eq 2 and Eq 3 are additive in nature and thus can be combined in a straightforward fashion. This formulation allows us to systematically combine information from three different models to produce better responses in terms of topic and semantic relevance. Conceptually, the likelihood term governs the grammatical structure of the response while the topic and semantic constraints drive content selection (Nenkova and Passonneau, 2004; Barzilay and Lapata, 2005).

4 Decoding with Distributional Constraints

In Section 3, we defined two constraints (one topic constraint and one semantic) for use in the decoding objective. Incorporating these constraints during decoding requires that they factorize in a way that is compatible with left-to-right beam search over words in the response. The standard approach to computing posterior distributions in topic models requires a probabilistic inference procedure over the entire source and target. Furthermore, computing semantic representations can involve the use of complex neural architectures. Both of these proceedures are difficult to integrate into decoding, because they are computationally expensive and would need to be called repeatedly within the inner loop of the decoder. Furthermore, when performing left-to-right beam search, as is common practice in neural generation, the complete response is generally not available. To address these challenges, we propose using simple additive variants of these methods that factorize over words and which we found to enable efficient decoding without sacrificing performance.

4.1 Topic Similarity

Estimating the topic distribution of the source, $P(T|X)$, and response, $P(T|Y)$, is a key step in implementing the topic-similarity constraint. HMM-LDA is a generative model that is able to separate topic and syntax words, by inferring topic distributions in a corpus while flexibly modeling function words. We briefly summarize this model before describing our implementation.

4.1.1 Syntax-Topics model

Griffiths et. al. (2005) suggested an unsupervised generative model that simultaneously labels each word in a document with a syntax (c) and topic (z) state. They modify the Latent Dirichlet Allocation model to include a syntactic component akin to a Hidden Markov Model (HMM). In LDA, each topic (z) is associated with a probability distribution over the vocabulary $\phi^{(z)}$. HMM-LDA adds additional distributions over words for each syntactic class (c) as $\phi^{(c)}$. A special class, $c = 0$, is reserved for topics. The transition model between classes c_{i-1} to c_i follows a multinomial distribution distribution $\pi^{(c_{i-1})}$. Each document has an associated distribution over topics $\theta^{(d)}$; each word, w_j, in the document has an associated latent topic variable, z_j, that is drawn from $\theta^{(d)}$ and c_j is drawn from $\pi^{(c_{j-1})}$. If $c_j = 0$, then w_j is drawn from $\phi^{(z_j)}$, otherwise it is drawn from $\phi^{(c_j)}$. Markov Chain Monte Carlo inference (MCMC) is used to infer values for the hidden topic and syntax variables associated with a given document collection. To estimate topic and syntax distributions, we performed collapsed Gibbs sampling over our training corpus of conversations, where each conversation is treated as a document. One sample of the hidden variables was used to estimate model parameters after 2,500 iterations of burn in. Our code for training the HMM-LDA model is available online[4].

4.1.2 Estimating Topic Distributions with HMM-LDA

To compute distributional topic constraints in neural response generation, we first need an efficient method for estimating topic distributions that factorizes over words, given a point estimate of an HMM-LDA model's parameters. We would like to estimate topic distributions based on content words contained in a sentence and ignore function words. HMM-LDA provides us with topic, $\phi^{(z)}$, and syntax, $\phi^{(c)}$, distributions over the vocabulary of words, $w \in V$. Treating a sentence as a bag-of-words we can estimate its distribution over topics as a sum of topic distributions over all words normalized by sentence length. However, we found this approach does not to work well in

---

[4] `https://github.com/abaheti95/HMM-LDA`

practice because it gives equal weight to topic and syntax words. To address this issue, we weighted each word's topic distribution $P(T|w)$ by its probability of being generated by the topic component of the HMM-LDA model (i.e. $P(C = 0|w)$). The topic distribution of a sentence, S, is estimated as:

$$P(T|S) = \frac{1}{Z}\Sigma_{w \in S} P(T|w)P(C = 0|w) \qquad (4)$$

where $Z = \Sigma_{w \in S} P(C = 0|w)$ is a normalizing constant that corresponds to the expected number of content words in the sentence. As mentioned earlier, a more accurate estimate of the topic distribution could be obtained using MCMC inference or by applying the forward-backward algorithm. However, these methods are computationally expensive and not well-suited to the decoding framework used in neural generation.

The method described above allows us to efficiently compute the topic distribution of a sentence for use in the topic constraint in Eq 2. For a similarity function, Δ, we simply use the vector dot product, which is closely related to cosine similarity. This formulation has the advantage that it enables memoization during decoding. Another advantage is that it captures the ratio of topic to syntax words due to the weights $P(C = 0|w)$.[5] Therefore, the overall constraint has the effect of keeping the syntax-topics ratio in generated hypothesis similar to the source.

4.2 Semantic Similarity

To define the semantic similarity constraint we first encode a semantic representation of the source and target into a fixed dimensional embedding space. There are many sentence embedding methods that could be used, however we want this encoding to be relatively efficient as it will be used many times during beam search.

Arora et. al. (2016) recently proposed a simple sentence embedding method, which was shown to have competitive performance across a variety of tasks. Their approach uses a weighted average of word embeddings where each word is weighted by $\frac{a}{a+P(w)}$; here, $P(w)$ is the unigram probability and a is a hyperparameter. Such a weighting scheme reduces the impact of frequent words (typically function words) in the overall sentence embedding. Next the first principal component of all the sentence embeddings in the corpus is removed. (Arora et al., 2016) points that the first principal component has high cosine similarity with common function words. Removing this component gives sentence embeddings that encapsulate the semantic meaning of the sentence. We use this technique in our implementation of $Emb()$ in Eq 3. For the similarity function, Δ, we use the dot product. Analogous to the topic constraint described above, this approach to measuring semantic similarity also decomposes over words and works well in the decoding framework.

Parameter	Value
RNN Type	Bi-LSTM
Layers	4
Hidden layer dim.	1000
Learning rate	0.1
max. grad. norm.	1
Optimization	Adadelta
Parameter Init	(-0.08, 0.08) (uniform)

Table 1: Hyperparameter setting for training

Bucket	#dialogues	#test
b1 (3-6 words)	10994	334
b2 (7-15 words)	15794	333
b3 (16-25 words)	5167	333
total	31955	1000

Table 2: Test set from Cornell Movie Dialogue Corpus. Column 2 shows the total number of dialogues that we got after all pre-processing and Column 3 shows the number of sampled dialogues in the test set.

5 Datasets

For training purposes we use OpenSubtitles (Tiedemann, 2009), a large corpus of movie subtitles (roughly 60M-70M lines) that is freely available and has been used in a broad range of recent work on data-driven conversation. OpenSubtitles does not contain speaker annotations on the dialogue turns, so as previously noted when used for learning data-driven conversation models the data is somewhat noisy. Nonetheless, it is possible to create a useful corpus of conversations from this data by assuming each line corresponds to a full speaker turn. Although this assumption is often violated, prior work has successfully trained

---

[5] Assuming topic distribution of syntax words to be uniform, a sentence with more syntax words will dampen modes in the distribution. Alternately, with less syntax words the overall distribution will be more peaked.

and evaluated neural conversation models using this corpus. In our experiments we used a preprocessed version of this dataset distributed by Li et. al. (2016a).[6] The dataset contains large number of two turn dialogues out of which we sampled 23M to use as our training set and 10k as a validation set.

Due to the noisy nature of the OpenSubtitles conversations we do not use them for evaluation. Instead, we leverage the Cornell Movie Dialogue Corpus (Danescu-Niculescu-Mizil and Lee, 2011) which is much smaller but contains accurate speaker annotations. We extracted all two turn conversations (source target pair) from this corpus and removed those with less than three and more than 25 words. After this, we divided the remaining conversations into three buckets based on source length. The numbers can be found in Table 2. From each bucket we randomly sampled ≈333 dialogues for a total of 1000 dialogues in our test set. We evaluate all models on this test set. Since automatic metrics do not correlate with human judgment, we manually tuned the hyperparameters (α and β) on a small development set (4 dialogues from each bucket to create a small 12 sentence development set; disjoint from test set). We manually inspected the responses generated by the model on the development set for different values of α and β and choose those that performed best.

6 Experimental Conditions and Baselines

During learning we use the same hyperparameters for all models; these are displayed in Table 1, and are based on those reported by Li et. al. (2016a).[7] We compare our approach with the following baselines:

MMI: We re-implemented the MMI-bidi method proposed by Li et. al. (2016a). MMI is a particularly appropriate baseline for comparison, as it encourages responses that have higher relevance to the input in contrast to conditional likelihood, which tends to favor responses with higher unconditional probability. MMI-bidi generates B candidates using Beam search on a Seq2Seq model trained to maximize conditional likelihood of the target given the source, $P(Y|X)$, then re-ranks them using a separately trained source given target model, $P(X|Y)$. Combining both directions in this way has the effect of maximizing mutual information (Li et al., 2016a).

TA-Seq2Seq: Another relevant baseline is the TA-Seq2Seq model of Xing et. al. (2017) that integrates information from a pre-trained topic model into neural response generation using an attention mechanism to condition on relevant topic words. They evaluate their model on a dataset of Chinese forum posts. Unfortunately we could not use the code provided by the authors due to data-mismatch (their model makes use of user identities which are not available in the OpenSubtitles corpus). We therefore compare with a re-implementation of their approach in which we modify each source sentence to include a list of the 20 most relevant topic words from HMM-LDA and then train using the same Seq2Seq framework with attention. This enables the model to condition on the relevant topic words. In addition to incorporating attention over topics, Xing et. al. also introduced an approach to *biased generation* - to replicate this we add a constant factor to all topic words during the prediction.

7 Results and Analysis

Our proposed decoding objective constraints (topic and semantic) are complementary to the MMI objective, which encourages diversity and relevance to the source input. Therefore, in addition to comparing against the baselines described above, we evaluated three variants of our model: **(1)** maximum conditional likelihood combined with semantic and topic distributional constraints with a beam size of 10 (DC-10) **(2)** The same configuration with MMI-bidi re-ranking using a beam size of 10 DC-MMI10 and **(3)** MMI-bidi reranking with a beam size of 200 (DC-MMI200). We test all configurations on the 1000 conversations test set described in Section 5 and compare them on automatic metrics and also in a crowd-sourced human evaluation. We do not consider TA-200 (TA-Seq2Seq, Beam=200), DC-200 and MMI-10 for human evaluation as they appear to perform worse than other model variants in automatic metrics and also on our set of development sentences. Sample responses for all the remaining models are presented in Table 3.

7.1 Automatic Metrics

Following Li et. al. (2016a), we report distinct-1 and distinct-2, which measure the diversity of re-

---

[6] http://nlp.stanford.edu/data/OpenSubData.tar

[7] OpenNMT is used for training our models (Klein et al., 2017).

Source	Target (ground truth)	MMI	Approximate TA-Seq2Seq	Distributional Constraints
where does he work ?	in there , sir .	does he know where he is ?	what kind of place is this ?	he has work to do at the university
here 's your jacket !	uh , thanks ...	i don 't want it !	what kind of suit is this ?	you can stick your shirt in it !
what 's so damn funny ?	been to any good <unk>lately ?	what are you laughing at ?	what 's the matter with you ?	it 's sort of a funny thing you know ?
well , what exactly does our platoon do ? serve <unk>? process paperwork ?	our platoon has the best assignment of all . we 're the first into battle !	well as a matter of fact	the best thing to do ?	well the war is over but we don 't have time for this shit
left us here to rot . that 's what they 've done . heroes of the newspapers !	all right , all right . keep quiet .	they 'r e all dead !	it 's time to fight !	we are brothers and kings of war !
that 's it . you 're not setting foot off this ship until we return home .	but i haven 't even shown you my biology project ...	we 'r e not going anywhere	i hope you don 't mind	we have orders that the ship will return to earth immediately
i 'm han solo , captain of this vessel . who 's in charge then ?	i 'm ben kenobi. luke starkiller here is leading our expedition .	i don 't know sir sir	you 'r e a coward !	i am captain kirk commander of the ship
her grandmother said she 'd been threatening to run away . and i found the car at the miami bus terminal .	you don 't think sam could 've put it there ?	but she didn 't tell me	that 's one hell of a job	it was parked in a car crash near the road

Table 3: Sample responses of all the models on the dev set

sponses. These are the ratios of types to tokens for unigrams and bigrams, respectively. We also report BLEU-1 scores following previous work, however it should be noted that BLEU-1 is not generally accepted to correlate with human judgments in conversation generation tasks (Liu et al., 2016) as there are many acceptable ways to reply to an input which may not match a reference response. Lastly, we compare the percentage of stop-words[8] of the responses generated by each model (smaller values, that are closer to the distribution of human conversations are preferred). The automatic evaluation is presented in Table 4.

For brevity we define aliases for each system in the 2^{nd} column of Table 4 which are used in subsequent discussion. The human responses are diverse and also generally longer than automatically generated responses. MMI200 has higher diversity than TA-Seq2Seq in terms of distinct-1 and distinct-2. This illustrates the importance of re-ranking using MMI. Our approach produces almost twice as many distinct unigrams and bigrams. We also observe MMI200 and TA-Seq2Seq achieve higher BLEU scores than our models, however this is not surprising since our models are designed to generate more interesting responses containing rarer content words that are less likely to appear in reference responses.

---

[8] Long Stopword List from `https://www.ranks.nl/stopwords`. We appended punctuations to this list.

Model	Alias	distinct-1	distinct-2	BLEU -1	Avg. length	Stop-word%
Human responses	human	2381/0.176	7532/0.602	-	13.5	70.66
MMI (Beam=200)	MMI200	351/0.058	990/0.197	12.8	6.0	84.91
TA-Seq2Seq(Beam=10)	TA-10	237/0.036	524/0.095	12.9	6.5	79.40
Dist. Const. (Beam=10)	DC-10	710/0.097	2014/0.320	11.0	7.3	**72.04**
Dist. Const. + MMI (Beam=10)	DC-MMI10	732/0.099	2098/0.327	11.4	7.4	73.87
Dist. Const. + MMI (Beam=200)	DC-MMI200	850/**0.116**	2946/**0.465**	11.6	7.3	72.25

Table 4: Automatic metrics evaluation. The 3^{rd} and 4^{th} columns show the ratio of types to tokens for unigrams and bigrams respectively. 7^{th} Column shows the % of stop-words generated by the models in their responses.

Model Alias	No(%)	Unsure(%)	Yes(%)
Plausible?			
human	19.807	23.448	56.745
MMI200	27.623	26.445	45.931
TA-10	**26.981**	26.874	**46.146**
DC-MMI200	30.835	24.41	44.754
Content Richness?			
human	16.488	19.914	63.597
MMI200	23.662	32.976	43.362
TA-10	31.799	30.086	38.116
DC-MMI200	**20.021**	26.660	**53.319**

Table 5: Human judgments for Plausibility of the different models. Each numerical cell contains a percentage value corresponding to its row truncated to 2 decimal precision.

Model Alias	No (%)	Unsure (%)	Yes (%)
Plausible?			
DC-10	36.617	27.944	35.439
DC-MMI10	33.619	28.694	37.687
DC-MMI200	**30.835**	24.41	**44.754**
Content Richness?			
DC-10	19.272	26.017	54.711
DC-MMI10	**18.844**	26.231	**54.925**
DC-MMI200	20.021	26.660	53.319

Table 6: Comparing the model variation by reducing beam size to 10 and also comparing decoder constraints without MMI reranking

As expected we observe that MMI200 and TA-10 have a higher percentage of stop-words than human responses. According to the human evaluation discussed in Section 7.2, these models were also found to have lower content richness.

7.2 Human Evaluation

We conducted a survey on the crowd-sourcing platform, Amazon Mechanical Turk. Every model response is scored on 2 categories: 1) **P**lausibility - *is the response plausible for the given source?* and 2) **C**ontent Richness - *does the response add new information to the conversation?* We asked the evaluators to respond on a 5-point scale to the questions above (Strongly Agree, Agree, Unsure, Disagree, Strongly Disagree). These were later collapsed to 3 categories (Agree, Unsure, Disagree). The results for plausibility and content richness of our model in addition to the MMI and TA-Seq2Seq baselines and human responses are presented in Table 5.

We observe that MMI200 and TA-10 models achieve slightly better plausibility scores since they tend to generate safe, dull responses. However, we find that when using a beam size of 200 and MMI re-ranking, our approach which incorporates distributional constraints, DC-MMI200, achieves competitive plausibility, while achieving significantly higher content richness.

7.2.1 Statistical Significance of Results

To verify the statistical significance of our findings, we conducted a pairwise bootstrap test (Efron and Tibshirani, 1994; Berg-Kirkpatrick et al., 2012) comparing the difference between percentage of *Agree* annotations (Yes column in the Table 5). We computed p-values for each pair of models: MMI200 vs DC-MMI200 and TA vs DC-MMI200. For plausibility, we did not find a significant difference in either comparison (p-value $\approx$ 0.25) while for content richness, both differences were found to be significant (p-value $<10^{-4}$). To summarize: our model significantly beats both baselines in terms of content richness while the difference in plausibility was not found to be statistically significant.

7.2.2 Pairwise Evaluation of Interestingness

To further validate our claims we also did a side by side comparison study between MMI200 and DC-MMI200. For every test case, we showed Mechanical Turk workers the source sentence along with responses generated by both systems and asked them select which is more interesting. We observe that in 56% out of 1000 cases, DC-MMI200 was rated as the more interesting response. The result is statistically significant with p-value $<4 \times 10^{-4}$ (using an exact binomial test).

7.3 Model Variations

To see the effectiveness of our decoding constraints separately, we compare the best performing DC-MMI200 model with DC-10 and DC-MMI10, both of which use a beam size of 10 – DC-10 does not include MMI reranking. The results of Mechanical Turk evaluation, following the approach described in Section 7.2, are presented in Table 6. We observe that with a beam size of 10 our model is able to generate content rich responses, but suffers in terms of plausibility. The values in the table suggests the decoding constraints defined in this work successfully inject content words into candidate hypotheses and that MMI is able to effectively choose plausible candidates. In the case of DC-10 and DC-MMI10, both models generate the same candidates, but MMI is able to re-rank the results and thus improves plausibility.

8 Related Work

Conversational agents primarily fall into two categories: task oriented dialogue systems (Williams et al., 2013; Wen et al., 2015) and chatbots (Weizenbaum, 1966), although there have been some efforts to integrate the two (Dodge et al., 2015; Yu et al., 2017). Some of the earliest work on data-driven chatbots (Ritter et al., 2011) explored the use of phrase-based Statistical Machine Translation (SMT) on large numbers of conversations gathered from Twitter (Ritter et al., 2010). Subsequent progress on the use of neural networks in machine translation inspired the use of Sequence-to-Sequence (Seq2Seq) models for data-driven response generation (Shang et al., 2015; Sordoni et al., 2015; Li et al., 2016a).

Our approach, which incorporates distributional constraints into the decoding objective, is related to prior work on posterior regularization (Mann and McCallum, 2008; Ganchev et al., 2010; Zhu et al., 2014). Posterior regularization introduces similar distributional constraints on expectations computed over unlabeled data using a model's parameters. These are typically added to the learning objective for semi-supervised scenarios where available labeled data is limited. In contrast, our approach introduces distributional constraints into the decoding objective as a way to combine neural conversation models trained on large quantities of conversational data with separately trained models of topics and semantic similarity that can drive content selection.

There are numerous examples of related work on improving neural conversation models. Shao et. al. (2017) introduced a stochastic approach to beam search that does segment-by-segment reranking to promote diversity. Zhang et. al. (2018) develop models which converse while assuming a persona defined by a short description of attributes. Wang et. al. (2017) suggested decoding methods that influence the style and topic of the generated response. Bosselutet al. (2018) develop discourse-aware rewards with reinforcement learning (RL) to generate long and coherent texts. Li et. al. (2016c) applied deep reinforcement learning to dialogue generation to maximize long-term reward of the conversation, as opposed to directly maximizing likelihood of the response. This line of work was further extended with adversarial learning (Li et al., 2017) that rewards generated conversations that are indistinguishable from real conversations in the data. Lewis et. al. (2017) applied reinforcement learning with dialogue rollouts to generate replies that maximize expected reward, while learning to generate responses from a crowdsourced dataset of negotiation dialogues. Choi et. al. (2018) used crowd-workers to gather a corpus of 100K information-seeking QA dialogues that are answerable using text spans from Wikipedia. Niu and Bansal (2018) designed a number of weakly-supervised models that generate polite, neutral or rude responses. Their fusion model combines a language model trained on polite utterances with the decoder. In the second method they prepend the utterance with a politeness label and scale its embedding to vary politeness. The third model is Polite-RL which assigns a reward based on a politeness classifier. Gimpel et. al. (2013) explored methods for increasing the diversity of N-best lists in machine translation by introducing a pairwise dissimilarity function. Similar ideas have been explored in the context of neural generation models. (Vijayakumar et al., 2016; Li and Jurafsky, 2016; Li et al., 2016b)

Following previous work we evaluated our approach using a combination of automatic metrics and human judgments. Some recent work has explored the possibility of adversarial evaluation of neural conversation models (Lowe et al., 2017; Li et al., 2017).

9 Conclusions

We presented an approach to generate more interesting responses in neural conversation models by incorporating side information in the form of distributional constraints. When using maximum likelihood decoding objectives, neural conversation models tend to generate safe responses, such as "I don't know" for most inputs. Our proposed approach provides a flexible method of incorporating a broad range of distributional constraints into the decoding objective. We proposed and empirically evaluated two constraints that factorize over words, and therefore naturally fit into the commonly used left-to-right beam search decoding framework. The first encourages the use of more relevant topic words in the response the second encourages semantic similarity between the source and target. We empirically demonstrated, through human evaluation, that when taken together these constraints lead to responses that contribute significantly more information to the conversation, while maintaining plausibility in the context of the input.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1464128.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in nlp. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005. Association for Computational Linguistics.

Antoine Bosselut, Asli Celikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang, and Yejin Choi. 2018. Discourse-aware neural rewards for coherent text generation. *arXiv preprint arXiv:1805.03766*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac : Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2015. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111.

Thomas L Griffiths, Mark Steyvers, David M Blei, and Joshua B Tenenbaum. 2005. Integrating topics and syntax. In *Advances in neural information processing systems*, pages 537–544.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, et al. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining, pages 955–964. ACM.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Jiwei Li, Will Monroe, Tianlin Shi, Sèbastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1116–1126.

Gideon S Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. *Proceedings of ACL-08: HLT*, pages 870–878.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the human language technology conference of the north american chapter of the association for computational linguistics: Hlt-naacl 2004*.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics (TACL)*.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1577–1586.

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2210–2219.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jörg Tiedemann. 2009. News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2140–2150.

Joseph Weizenbaum. 1966. Elizaa computer program for the study of natural language communication between man and machine. *Communications of the ACM*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2078–2088.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, volume 17, pages 3351–3357.

Zhou Yu, Alexander Rudnicky, and Alan Black. 2017. Learning conversational systems that interleave task and non-task content. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

Jun Zhu, Ning Chen, and Eric P Xing. 2014. Bayesian inference with posterior regularization and applications to infinite latent svms. *The Journal of Machine Learning Research*, 15(1):1799–1847.

Better Conversations by Modeling, Filtering, and Optimizing for Coherence and Diversity

Xinnuo Xu, Ondřej Dušek, Ioannis Konstas and Verena Rieser
The Interaction Lab, School of Mathematical and Computer Sciences
Heriot-Watt University, Edinburgh
`xx6, o.dusek, i.konstas, v.t.rieser@hw.ac.uk`

Abstract

We present three enhancements to existing encoder-decoder models for open-domain conversational agents, aimed at effectively modeling coherence and promoting output diversity: (1) We introduce a measure of coherence as the GloVe embedding similarity between the dialogue context and the generated response, (2) we filter our training corpora based on the measure of coherence to obtain topically coherent and lexically diverse context-response pairs, (3) we then train a response generator using a conditional variational autoencoder model that incorporates the measure of coherence as a latent variable and uses a context gate to guarantee topical consistency with the context and promote lexical diversity. Experiments on the OpenSubtitles corpus show a substantial improvement over competitive neural models in terms of BLEU score as well as metrics of coherence and diversity.

1 Introduction

End-to-end neural response generation methods are promising for developing open domain dialogue systems as they allow to learn from very large unlabeled datasets (Shang et al., 2015; Sordoni et al., 2015; Vinyals and Le, 2015). However, these models have also been shown to generate generic, uninformative, and non-coherent replies (e.g., *"I don't know."* in Figure 1), mainly due to the fact that neural systems tend to settle for the most frequent options, thus penalizing length and favoring high-frequency word sequences (Sountsov and Sarawagi, 2016; Wei et al., 2017).

To address these problems, Li et al. (2016a) and Li et al. (2017a) attempt to promote diversity by improving the objective function, but do not model diversity explicitly. Serban et al. (2017) focus on model structure without any upgrades to the objective function. Other works control the style of the output by leveraging external resources (Hu et al. (2017): sentiment classifier, time annotation; Zhao et al. (2017): dialogue acts) or focus on well-structured input such as paragraphs (Li and Jurafsky, 2017).

Conversational history	**Response**
A: You stay out of this. B: So you want water, huh? A: That's right.	B-Coh: Well, I got water. B-Incoh: I don't know.
A: Where do we start? B: Kitchen. A: Definitely the kitchen.	B-Coh: Specifically the stove. B-Incoh: Let's go for a walk.

Figure 1: Examples of conversational history (left) with two alternative responses to follow it (right): (B-Coh) a more coherent, topical utterance, and (B-Incoh) a generic, inconsistent response.

This paper extends previous attempts to model diversity and coherence by enhancing all three aspects of the learning process: the data, the model, and the objective function. While previous research has addressed these aspects individually, this paper is the first to address all three in a unified framework. Instead of using existing linguistic knowledge or labeled datasets, we aim to control for coherence by learning directly from data, using a fully unsupervised approach. This is also the first work encoding and evaluating coherence explicitly in the dialogue generation task, as opposed to using diversity, style, or other properties of responses as a proxy.

In this work, given a dialogue history, we regard as a coherent response an utterance that is thematically correlated and naturally continuing from the previous turns, as well as lexically diverse. For example, in Figure 1 the response *"Specifically the stove."* is a very natural and coherent response, elaborating on the topic of *kitchen* introduced in

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3981–3991
Brussels, Belgium, October 31 - November 4, 2018.

the previous two utterances and containing rich thematic words, whereas the response *"Let's go for a walk."* is unrelated and uninteresting.

In order to obtain coherent responses, we present three generic enhancements to existing encoder-decoder (E-D) models:

1. We define a *measure of coherence* simply as the averaged word embedding similarity between the words of the context and the response computed using GloVe vectors (Pennington et al., 2014).
2. We filter a corpus of conversations based on our measure of coherence, which leaves us with context-response pairs that are both topically coherent and lexically diverse.
3. We train an E-D generator recast as a conditional Variational Autoencoder (cVAE; Zhao et al., 2017) model that incorporates two latent variables, one for encoding the context and another for conditioning on the measure of coherence, trained jointly as in Hu et al. (2017). We then decode using a context gate (Tu et al., 2017) to control the generation of words that directly relate to the most topical words of the context and promote coherence.

Experiments on the OpenSubtitles (Lison and Meena, 2016) corpus demonstrate the effectiveness of the overall approach. Our models achieve a substantial improvement over competitive neural models. We provide an ablation analysis, quantifying the contributions that come from effective modeling of coherence into our models. All our experimental code is freely available on GitHub.[1]

2 Coherence-based Dialogue Generation

Our model aims to generate responses given a dialogue context, incorporating measures of coherence estimated purely from the training data. We propose the following enhancements to the attention-based E-D architecture (Bahdanau et al., 2015; Luong et al., 2015):

- We introduce a stochastic latent variable $\mathbf{z}$ conditioned on previous dialogue context to store the global information about the conversation (Bowman et al., 2016; Chung et al., 2015; Li and Jurafsky, 2017; Hu et al., 2017).
- We force the model to condition on the measure of coherence explicitly by encoding a latent variable (code) c learned from data.
- We incorporate a context gate (Tu et al., 2017) that dynamically controls the ratio at which the generated words in the response derive directly from the coherence-enhanced dialogue context or the previously generated parts of the response.

In the rest of this section, we introduce the measure of coherence (Section 2.1), we present an overview of our model (Section 2.2), and finally describe the model in detail (Sections 2.3–2.4).

2.1 Measure of Dialogue Coherence

Semantic vector space models of language represent each word with a real-valued word embedding vector (Pennington et al., 2014). By simply taking a weighted average of all its word embeddings, a whole sentence can be mapped into the semantic vector space. We define the coherence of a dialogue as the average distance between semantic vectors of preceding dialogue context and its response.

Let $\mathbf{x} = \{x_1, \dots x_j, \dots x_J\}$ represent a dialogue context and $\mathbf{y} = \{y_1, \dots y_i, \dots y_I\}$ a response. J and I are the numbers of words in the dialogue context and its response, respectively. Semantic vector space models map each word x_j into embeddings $\mathbf{x}_j^{emb}$, and y_i into $\mathbf{y}_i^{emb}$. The semantic representation of a dialogue context $\mathbf{x}$ is then $\mathbf{x}^{emb} = \sum_{j=1}^{J} w_j \mathbf{x}_j^{emb}$; for a response $\mathbf{y}$, it is $\mathbf{y}^{emb} = \sum_{i=1}^{I} v_i \mathbf{y}_i^{emb}$. Here, w_j and v_i are importance weights for each word in the sentence.[2] The measure of coherence is then defined as the cosine distance of the two semantic vectors of the dialogue context and its response:

$$C(\mathbf{x}, \mathbf{y}) = \cos\left(\mathbf{x}^{emb}, \mathbf{y}^{emb}\right) \qquad (1)$$

2.2 Model Overview

End-to-end response generation for dialogue can be formalized as follows: Given a dialogue context $\mathbf{x}$, a dialogue generator generates the next utterance $\mathbf{y}$. During the training process, the aim for a dialogue generator is to maximize the probability $p(\mathbf{y}|\mathbf{x})$ over the training dataset. To encode

---

[1] `https://github.com/XinnuoXu/CVAE_Dial`

[2] We set the importance weights to 0 for a list of stop words (high-frequency words such as articles and prepositions, names, punctuation marks), 1 otherwise.

dialogue contexts that adequately incorporate coherence information, we build our generator based on the cVAE model of Hu et al. (2017), which has been used to control text generation with respect to linguistic properties, such as tense or sentiment.

In our model, the response $\mathbf{y}$ is generated conditioned on the previous conversation $\mathbf{x}$, a diversity-promoting latent variable $\mathbf{z}$, and a latent variable c indicating dialogue coherence; $\mathbf{z}$ and c are independent. The generation probability $p\left(\mathbf{y}|\mathbf{x}\right)$ is defined as:

$$\begin{aligned} p\left(\mathbf{y}|\mathbf{x}\right) &= \int_{\mathbf{z},c} p\left(\mathbf{y}|\mathbf{x},\mathbf{z},c\right) p\left(\mathbf{z},c|\mathbf{x}\right) \mathrm{d}\mathbf{z}\,\mathrm{d}c \\ &= \int_{\mathbf{z},c} p\left(\mathbf{y}|\mathbf{x},\mathbf{z},c\right) p\left(\mathbf{z}|\mathbf{x}\right) p\left(c|\mathbf{x}\right) \mathrm{d}\mathbf{z}\,\mathrm{d}c \end{aligned} \tag{2}$$

Unfortunately, optimizing Eq (2) during training is intractable; therefore, we apply variational inference and optimize instead the variational lower bound:

$$\begin{aligned} \log p\left(\mathbf{y}|\mathbf{x}\right) &= \log \int_{\mathbf{z},c} p\left(\mathbf{y}|\mathbf{x},\mathbf{z},c\right) p\left(\mathbf{z},c|\mathbf{x}\right) \mathrm{d}\mathbf{z}\,\mathrm{d}c \\ &\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})p(c|\mathbf{x},\mathbf{y})}\left[\log p\left(\mathbf{y}|\mathbf{x},\mathbf{z},c\right)\right] \\ &\quad - D_{KL}\left(q\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right) \parallel p\left(\mathbf{z}|\mathbf{x}\right)\right) \end{aligned} \tag{3}$$

where $p\left(\mathbf{y}|\mathbf{x},\mathbf{z},c\right)$ is the probability of generating utterance $\mathbf{y}$ given $\mathbf{x}$, $\mathbf{z}$ and c; $q\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right)$ stands for the approximate posterior distribution of the latent variable $\mathbf{z}$ conditioned on dialogue context $\mathbf{x}$ and the gold response $\mathbf{y}$; $p\left(c|\mathbf{x},\mathbf{y}\right)$ is the measure of coherence between context $\mathbf{x}$ and response $\mathbf{y}$; $p\left(\mathbf{z}|\mathbf{x}\right)$ is the true prior distribution of $\mathbf{z}$ conditioned only on dialogue context $\mathbf{x}$; $D_{KL}\left(\cdot|\cdot\right)$ denotes the KL-divergence. We assume that both $q\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right)$ and $p\left(\mathbf{z}|\mathbf{x}\right)$ are Gaussian with mean vectors μ_{appr}, μ_{true} and covariance matrices Σ_{appr}, Σ_{true}.

2.2.1 Model Details

Optimizing Eq (3) consists of two parts: (1) minimizing the KL-divergence between the approximate posterior distribution and the true prior distribution of $\mathbf{z}$, (2) maximizing the probability of generating the gold response $\mathbf{y}$ conditioned on dialogue context $\mathbf{x}$ and coherence factors $\mathbf{z}$ and c. Figure 2 shows the pipeline of the training procedure.

Encoder: First, we encode a dialogue context $\mathbf{x}$ into a hidden state $\mathbf{h}$ using the *context encoder*, which is based on Recurrent Neural Networks (RNNs). Then the *posterior network* encodes both dialogue context $\mathbf{x}$ and gold response $\mathbf{y}$ into a hidden state $\mathbf{h}_{appr}$ followed by two linear transformations $f_{appr}\left(\cdot\right)$ and $g_{appr}\left(\cdot\right)$ to map $\mathbf{h}_{appr}$ into mean vector μ_{appr} and covariance matrix Σ_{appr}. The latent variable $\mathbf{z}$ can be sampled from the distribution $N\left(\mu_{appr}, \Sigma_{appr}\right)$:

$$\begin{aligned} \mu_{appr} &= f_{appr}\left(\mathbf{h}_{appr}\right) \\ \Sigma_{appr} &= g_{appr}\left(\mathbf{h}_{appr}\right) \\ q\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right) &= N\left(\mu_{appr}, \Sigma_{appr}\right) \end{aligned} \tag{4}$$

The *prior network* in Figure 2 takes a form similar to the posterior network:

$$\begin{aligned} \mu_{true} &= f_{true}\left(\mathbf{h}_{true}\right) \\ \Sigma_{true} &= g_{true}\left(\mathbf{h}_{true}\right) \\ p\left(\mathbf{z}|\mathbf{x}\right) &= N\left(\mu_{true}, \Sigma_{true}\right) \end{aligned} \tag{5}$$

where $\mathbf{h}_{true}$ is the final hidden state of an RNN encoding only the dialogue context $\mathbf{x}$, and $f_{true}\left(\cdot\right)$, $g_{true}\left(\cdot\right)$ are linear transformations. Code c is given by the *coherence measure* from Eq (1).

Decoder: We build an attention-based decoder (Bahdanau et al., 2015; Luong et al., 2015) using RNNs to generate responses conditioned on encoded dialogue context $\mathbf{h}$, diversity signal $\mathbf{z}$, and coherence signal c. We concatenate the latent variables $\mathbf{z}$ and c to the context encoder hidden state $\mathbf{h}$ and feed them into the decoder as the initial hidden state $\mathbf{s}_0$, similar to Hu et al. (2017).

During the decoding process, tokens are generated sequentially under the following probability distribution:

$$\begin{aligned} p\left(\mathbf{y}|\mathbf{x},\mathbf{z},c\right) &= \prod_{i=1}^{I} p\left(y_i|y^{<i},\mathbf{x},\mathbf{z},c\right) \\ &= \prod_{i=1}^{I} g\left(y_{i-1},\mathbf{s}_i,\mathbf{a}_i\right) \end{aligned} \tag{6}$$

where I is the length of the produced response; $g\left(\cdot\right)$ is an RNN; $\mathbf{s}_i$ is the hidden state of the decoder at time step i which is conditioned on the previously generated token y_{i-1}, the previous hidden state $\mathbf{s}_{i-1}$, and the weighted attention vector

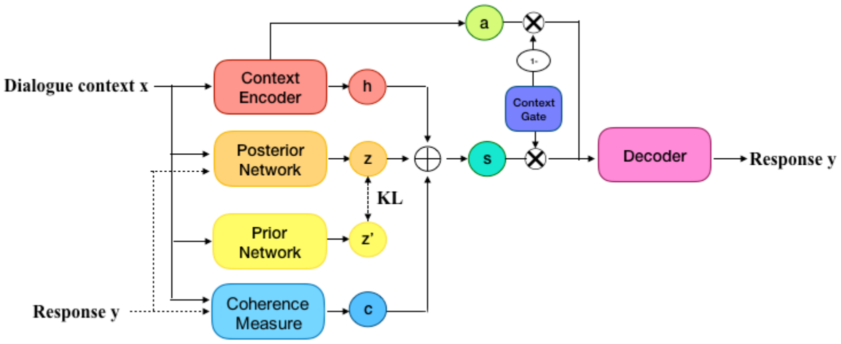

Figure 2: The training process of the generative model. First, the dialogue context is encoded: $\mathbf{h}$ is the final hidden state of the context encoder. Then we derive the diversity-promoting latent variable $\mathbf{z}$. Next, we compute the latent variable c that corresponds to the measure of coherence between the dialogue context x and the generated response y. We concatenate all three vectors into $\mathbf{s}$ to feed the decoder. $\mathbf{a}$ is the attention matrix calculated for every time step of the decoding process.

$\mathbf{a}_i$:

$$\mathbf{s}_i = f\left(y_{i-1}, \mathbf{s}_{i-1}, \mathbf{a}_i\right) \qquad (7)$$

$$\mathbf{a}_i = \sum_{j=1}^{J} \mathbf{w}_{ij}\mathbf{h}_i \qquad (8)$$

where J is the number of tokens of the dialogue context; $\mathbf{h}_i$ is the i^{th} hidden state of the encoder; the attention weight $\mathbf{w}_{ij}$ of each context hidden state $\mathbf{h}_i$ is computed following Luong et al. (2015).

Context Gate: To increase the influence of code c, we introduce the context gate k. Unlike Tu et al. (2017), whose context gate assigns an element-wise weight to the input signal deriving from the encoder RNN, we build the context gate conditioned only on the coherence signal:

$$k_i = \lambda\sigma\left(c - c_i'\right) \qquad (9)$$

where σ is the sigmoid function; λ is a bias term;[3] c is the target value of the measure of coherence, calculated by $C\left(\mathbf{x}, \mathbf{y}\right)$ (see Section 2.1); c_i' is the measure of coherence between the dialogue context and the generated prefix sentence at time step i, calculated by $C\left(\mathbf{x}, \mathbf{y}^{<i}\right)$. Now Eq (7) with the context gate applied to $\mathbf{s}_i$ can be rewritten as:

$$\mathbf{s}_i = f\left(y_{i-1}, (1 - k_i) \circ \mathbf{s}_{i-1}, k_i \circ \mathbf{a}_i\right) \qquad (10)$$

where $\circ$ denotes element-wise multiplication.

The coherence-informed context gate aims to dynamically control the ratio at which preceding dialogue context and previously generated tokens of the current response contribute to the generation of the next token in the response.

2.3 Training

Our generator is trained similarly to Hu et al. (2017). The objective function is a weighted combination of three losses (generation, coherence, and diversity):

$$\mathcal{L} = \mathcal{L}_G + \lambda_c\mathcal{L}_c + \lambda_{\mathbf{z}}\mathcal{L}_{\mathbf{z}} \qquad (11)$$

To teach the generator to produce responses close to the training data, we maximize the generation probability of the training response $\log p\left(\mathbf{y}|\mathbf{x}\right)$ given the dialogue context according to Eq (2). During training, we set $\mathcal{L}_G = -\log p\left(\mathbf{y}|\mathbf{x}\right)$ and minimize the following:

$$\begin{aligned}\mathcal{L}_G =& D_{KL}\left(q\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right) \parallel p\left(\mathbf{z}|\mathbf{x}\right)\right) \\ &- \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})p(c|\mathbf{x},\mathbf{y})}\left[\log p\left(\mathbf{y}|\mathbf{x},\mathbf{z},c\right)\right]\end{aligned} \qquad (12)$$

Apart from the generation loss, the coherence measure provides an extra learning signal $\mathcal{L}_c$ which pushes the generator to produce responses that match the coherence signal given by the latent variable c.

$$\mathcal{L}_c = -\mathbb{E}_{p(\mathbf{z}|\mathbf{x})p(c)}\left[\log p\left(c|\mathbf{x}, \widetilde{G}\left(\mathbf{x},\mathbf{z},c\right)\right)\right] \qquad (13)$$

In Eq (13), $p\left(c\right) = N\left(0,1\right)$ is the prior distribution of the coherence variable c. To ensure that the loss is differentiable, we cannot sample words from the response vocabulary. Instead we define $\widetilde{G}\left(\mathbf{x},\mathbf{z},c\right) = \mathbf{y}^s = \{\mathbf{y}_1^s, \ldots \mathbf{y}_i^s, \ldots \mathbf{y}_I^s\}$ as the sequence of output word probability distributions. $p\left(c|\mathbf{x}, \widetilde{G}\left(\mathbf{x},\mathbf{z},c\right)\right)$ is predicted by the coherence measure defined in Eq (1) with $\mathbf{y}^{emb}$ set as:

$$\mathbf{y}^{emb} = \sum_{i=1}^{I} \mathbf{y}_j^{s\top} M_{glv} \qquad (14)$$

---

[3] We set λ empirically against the development set.

where M_{glv} is the word embedding matrix trained using GloVe (Section 2.1).

The last component in Eq (11) is the independent constraint $\mathcal{L}_{\mathbf{z}}$ that forces the soft distribution over the generated response $\widetilde{G}$ to be diverse, so that it is able to faithfully reproduce the latent variable $\mathbf{z}$:

$$\mathcal{L}_{\mathbf{z}} = -\mathbb{E}_{p(\mathbf{z}|\mathbf{x})p(c)} \left[\log q \left(\mathbf{z} | \mathbf{x}, \widetilde{G} \left(\mathbf{x}, \mathbf{z}, c \right) \right) \right] \tag{15}$$

where $q \left(\mathbf{z} | \mathbf{x}, \widetilde{G} \left(\mathbf{x}, \mathbf{z}, c \right) \right)$ is predicted by the posterior network with $\mathbf{y}_j^s$ as the soft input to the RNN encoder at each time step j.

2.4 Inference

Figure 3 shows the inference process of the generative model. Given a dialogue context $\mathbf{x}$ and an expected coherence value c, the context encoder first encodes the dialogue context into a hidden state $\mathbf{h}$. The prior network then generates a sample $\mathbf{z}'$ conditioned on the dialogue context. The decoder is initialized with $\mathbf{s}$, i.e., the concatenation of $\mathbf{h}$, $\mathbf{z}$ and c. During decoding, the next word is generated via the context gate modulating between the attention-reweighted context and the previously generated words of the response.

3 Dataset and Filtering

Dataset for Generator

We train and evaluate our models on the OpenSubtitles corpus (Lison and Tiedemann, 2016) with automatic dialogue turn segmentation (Lison and Meena, 2016).[4] A training pair consists of a dialogue context and a corresponding response. We consider three consecutive turns as the dialogue context and the following turn as the response. From a total of 65M instances, we select those that have context and response lengths of less than 120 and 30 words, respectively. We create two datasets:

1. **OST** (plain OpenSubtitles) consists of 2M/4K/4K instances as our training/development/test sets, selected randomly from the whole corpus;
2. **fOST** (filtered OpenSubtitles) contains the same amount of instances, but randomly selected only among those that have a measure of coherence score $C(\mathbf{x}, \mathbf{y}) \geq 0.68$.[5]

Dataset	Coh	D-1%	D-2%	D-Sent%
OST	0.390	14.3	57.9	83.8
fOST	0.801	15.5	62.9	89.3

Table 1: Coherence and diversity metrics[7] for the OST and fOST datasets (see Section 3 for the datasets and Section 4.2 for metrics definition).

Filtering of the OpenSubtitles corpus is motivated by the fact that by removing the video and audio modalities which the subtitles originally accompanied, we are very often left with incomplete and incoherent dialogues. Therefore, by keeping dialogues with high coherence scores, we aim at building a high quality corpus with (1) more semantically coherent and topically related contexts and responses, and (2) fewer general and dull responses. Table 3 shows the coherence and diversity metrics (cf. Section 4.2) between OST and fOST. Unsurprisingly, coherence for fOST is much higher than OST, with a slightly higher diversity. We list dialogue examples for different coherence scores in Supplemental Material B.

Dataset for Coherence Measure

In order to accurately measure coherence on our domain using the semantic distance as defined in Section 2.1, we train GloVe embeddings on the full OpenSubtitles corpus (i.e. 100K movies).

4 Experiments

Our generator model, ablative variants, and baselines are implemented using the publicly available OpenNMT-py framework (Klein et al., 2017) based on Bahdanau et al. (2015) and Luong et al. (2015). We used the publicly available glove-python package[8] to implement our coherence measure.

We experiment on two versions of our model: (1) cVAE with the coherence context gate as described in Section 2.3 (*cVAE-XGate*), (2) cVAE with the original context gate implementation of

---

[4] `http://www.opensubtitles.org`

[5] The coherence score is calculated as shown in Eq (1). We observed that the scores on the training set follow a normal distribution with a slight tail on the negatively correlated side, so we fit a normal distribution to the data with parameters $N(0.25, 0.22)$ and set the cut-off to $+2\sigma$. A histogram of coherence scores is shown in Figure 5 in Supplemental Material A.

[7] Note that Distinct-1 and Distinct-2 are computed on a randomly selected subsets of 4k responses.

[8] `https://github.com/maciejkula/glove-python`

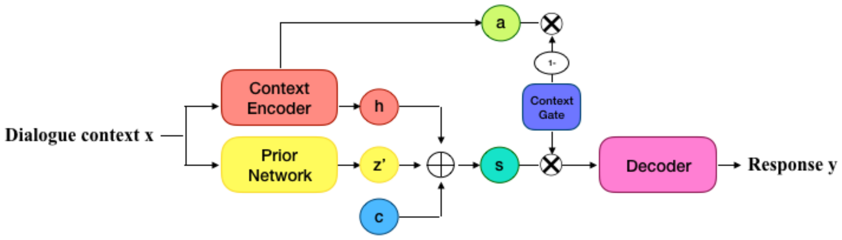

Figure 3: The inference process of the generative model, where the latent variable c is given as an input.

(Tu et al., 2017) (*cVAE-CGate*). For each of these, we consider the main variant where the input coherence measure c is preset to a fixed ideal value as estimated on development data (1.0 for OST and 0.95 for fOST), as well as an oracle variant where we use the true coherence measure between the context and the gold-standard response in the test set (indicated with "(C)" in Tables 2 and 3).

We compare against two baseline models: (1) a vanilla E-D with attention (*Attention*) (Luong et al., 2015); (2) an enhancement where output beams are rescored using the maximum mutual information anti-language model (MMI-antiLM) of Li et al. (2016a) (*MMI*).

4.1 Parameter Settings

We set our model parameters based on preliminary experiments on the development data.

We use 2-layer RNNs with LSTM cells (Hochreiter and Schmidhuber, 1997) with input/hidden dimension of 128 for both the context encoder and the decoder. The dropout rate is set to 0.2 and the Adam optimizer (Kingma and Ba, 2015) is used to update the parameters. A vocabulary of 25,000 words is shared between the encoder and the decoder.

Both the posterior network and prior network for the latent variable learning are built with 2-layer LSTM RNNs with input/hidden dimension of 64. The dimension of the latent variable $\mathbf{z}$ is set to 20. Same as for the encoder and decoder, the dropout rate is 0.2 and the Adam optimizer is used to update the parameters.

The window size for GloVe computation in our coherence measure is set to 10.

4.2 Evaluation metrics

We use a number of metrics to evaluate the outputs of our models:

- **BLEU, B1, B2, B3** – the word-overlap score against gold-standard responses (Papineni et al., 2002) used by the vast majority of recent dialogue generation works (Zhao et al., 2017; Yao et al., 2017; Li et al., 2017a, 2016c; Sordoni et al., 2015; Li et al., 2016a; Ghazvininejad et al., 2017). BLEU in this paper refers to the default BLEU-4, but we also report on lower n-gram scores (B1, B2, B3).[9]
- **Coh** – our novel GloVe-based coherence score calculated using Eq (1) showing the semantic distance of dialogue contexts and generated responses.
- **D-1, D-2, D-Sent** – common metrics used to evaluate the diversity of generated responses (e.g. Li et al., 2016a; Xu et al., 2017; Xing et al., 2017; Dhingra et al., 2017): the proportion of distinct unigrams, bigrams, and sentences in the outputs.

5 Results

All model variants described in Section 4 are trained on both OST and fOST datasets. Tables 2 and 3 present the scores of all models tested on the OST and fOST test sets, respectively. Note that in addition to testing the models on the respective test sections of their training datasets, we also test them on the other dataset (OST-trained models on fOST and vice-versa). This way, we can observe the performance of the fOST-trained models in more noisy contexts and see how good the OST-trained models are when evaluated against coherent responses only.

Given all the evaluated model variants, we can observe the effects and contributions of the individual components of our setup:

- *Data filtering:* The models trained on fOST consistently outperform the same models trained on OST – for all evaluation metrics and on both test sets. This shows that coherence-based training data filtering is generally beneficial.

---

[9] We use the Multi-BLEU script from OpenNMT to measure BLEU scores.

Training data	Model	BLEU%	B1%	B2%	B3%	Coh	D-1%	D-2%	D-Sent%
OST	Attention	1.32	10.92	3.85	2.10	0.293	3.4	14.2	25.6
	MMI	1.31	11.06	3.88	2.09	0.284	3.3	14.6	28.2
	cVAE-CGate (C)	1.58	11.86	4.45	2.48	0.311	4.1	15.0	28.2
	cVAE-XGate (C)	1.51	13.38	4.97	2.58	0.324	3.9	14.5	29.8
	cVAE-CGate (1.0)	**1.60**	**17.08**	**5.78**	**2.86**	0.404	**5.0**	**27.1**	79.7
	cVAE-XGate (1.0)	1.44	15.83	5.34	2.62	**0.413**	4.5	22.6	**80.2**
fOST	Attention	1.79	15.43	5.65	2.94	0.758	**11.9**	41.8	92.7
	MMI	1.99	16.24	**6.06**	3.22	0.764	**11.9**	44.5	95.8
	cVAE-CGate (C)	**2.10**	15.98	6.05	**3.35**	0.728	**11.9**	37.6	88.4
	cVAE-XGate (C)	1.85	**16.44**	5.94	3.07	0.706	10.3	31.2	80.4
	cVAE-CGate (0.95)	2.02	15.52	5.78	3.16	**0.767**	10.6	**44.8**	**98.7**
	cVAE-XGate (0.95)	1.64	14.43	5.20	2.70	0.745	9.0	36.9	**98.7**

Table 2: Evaluation results on the OST test set (see Section 4 for model description and Section 4.2 for metrics definition). Note that the cVAE-CGate(C) / cVAE-XGate(C) models use the true c value between the context and the gold response as input. Other cVAE-CGate / cVAE-XGate models use fixed values for c selected on dev sets shown in brackets. BLEU score reported here is BLEU-4; B1, B2 and B3 denote lower n-gram BLEU scores.

Training data	Model	BLEU%	B1%	B2%	B3%	Coh	D-1%	D-2%	D-Sent%
OST	Attention	0.86	8.34	2.79	1.45	0.284	3.6	14.6	29.4
	MMI	0.89	8.47	2.89	1.48	0.278	3.7	15.3	31.5
	cVAE-CGate (C)	1.64	10.20	4.17	2.40	0.329	5.1	19.4	35.8
	cVAE-XGate (C)	1.80	11.70	4.90	2.83	0.359	5.2	19.2	39.7
	cVAE-CGate (1.0)	2.25	16.82	6.81	3.70	0.422	**5.4**	**28.2**	81.0
	cVAE-XGate (1.0)	**2.41**	**18.62**	**7.56**	**4.09**	**0.434**	4.8	23.4	**84.0**
fOST	Attention	3.84	16.65	8.72	5.54	0.803	**12.8**	43.4	88.7
	MMI	3.84	16.81	8.78	5.57	0.803	12.6	42.5	88.8
	cVAE-CGate (C)	4.58	17.64	9.53	6.30	0.796	12.4	41.6	85.5
	cVAE-XGate (C)	4.33	18.43	9.59	6.11	0.783	10.7	33.1	78.8
	cVAE-CGate (0.95)	**4.98**	20.95	**10.93**	**7.02**	**0.814**	12.1	**51.4**	**98.2**
	cVAE-XGate (0.95)	4.47	**20.98**	10.43	6.50	0.797	10.4	42.5	97.6

Table 3: Evaluation results on the fOST test set (see Section 4 and Table 2 for model description; see Section 4.2 for metrics definition). BLEU score reported here is BLEU-4; B1, B2 and B3 denote lower n-gram BLEU scores.

- *cVAE-Context Gate models:* Nearly all cVAE-based models perform markedly better than the baselines w.r.t. BLEU, coherence, and diversity.[10]

  If we look at models trained on OST and tested on fOST (the top half of Table 3), we can see that all cVAE-based models, especially cVAE-XGate, are able to learn to produce coherent and diverse response even when trained on a noisy, incoherent corpus. Examples of responses generated by the baseline MMI model and by cVAE-XGate in Figure 4 show that cVAE-XGate mostly produces more diverse and coherent responses than MMI.

- *Preset c vs. oracle models with gold-standard c:* Table 2 shows that on the noisy OST test set, cVAE-based models using the gold-standard value of c achieve higher BLEU scores than models using preset c. This is expected since many gold-standard responses in the unfiltered set have a low coherence score; – the model can generate a more generic response if the gold-standard c is low. The models with preset c always attempt to generate coherent responses, which is apparent from the other metrics: Coh and D-Sent are consistently higher than for models using gold-standard c.

  On the fOST test set where only high-coherence responses are expected, models using fixed c consistently reach higher scores in all metrics including BLEU (see Table 3). This shows that in general, using a preset constant value of c works well, even better than using the gold-standard c.

In sum, using our coherence measure both for data filtering and inside the models leads to output performance improvements.

---

[10] We performed paired bootstrap re-sampling for the best cVAE model and the best baseline model in each experiments set (Table 2 and Table 3) as is done for MT (Koehn, 2004), which confirmed statistical significance at 99% confidence level for all cases except for models trained on fOST and tested on OST (bottom half of Table 2).

		Dialogue context	Response
1	A:	i have an audition at 4:00.	B-GT: you volunteered.
	B:	apparently now i'm cooking dinner.	B-MMI: i don 't know.
	A:	how did this all happen?	B-XGT: well , i'm going to have a dinner.
2	A:	it's not that easy.	B-GT: the waiting list would take too long.
	B:	we can't contact potential donors.	B-MMI: i don't know.
	A:	if we want to save frank, his brother is his only hope	B-XGT: we can't save him.
3	A:	that fish ate my fish!	B-GT: you'll wait till next year to win one of these.
	B:	you cost me £30,000	B-MMI: i'm sorry.
	A:	sorry, folks.	B-XGT: you owe me a little fish.
4	A:	was it what you wanted?	B-GT: you like model planes?
	B:	no.	B-MMI: i wanted to know.
	A:	i wanted an airplane.	B-XGT: i wanted a helicopter.
5	A:	great.	B-GT: yeah, a vanilla decaf latte.
	B:	thanks so much.	B-MMI: no.
	A:	vanilla decaf latte?	B-XGT: yeah , that 's a good coffee
6	A:	to make everybody happy.	B-GT: and stick it in your mouth.
	B:	which means that sometimes	B-MMI: and you got to take a shower.
	A:	you got to take a pile of poop	B-XGT: and you got to take a walk.
7	A:	now you'll knock again and say you forgot to ask	B-GT: should i knock and make her come here for that?
	B:	if she has a friend for your friend.	B-MMI: you're crazy!
	A:	are you crazy?	B-XGT: i'm crazy, i'm crazy.
8	A:	i knew you would say that.	B-GT: you are all way too pretty to be arguing.
	B:	mr. kerem is a charming and decent guy.	B-MMI: he's a very good man.
	A:	and he's really into you.	B-XGT: i didn't say that.

Figure 4: Generation output examples comparing two models trained on the OST dataset and tested on the fOST test set: the MMI baseline (B-MMI) and our cVAE-XGate (1.0) model (B-XGT). B-GT is the ground-truth response from the test set. The three sequential dialog turns on the left are the preceding dialogue context used to generate the responses. Corresponding topical phrases are underlined. We can see that cVAE-XGate (1.0) mostly produces markedly more coherent and specific outputs than MMI (1-5). In some cases, it is comparable with MMI (6-7) and occasionally, it is less coherent (8).

6 Related Work

Our work fits into the context of the very active area of end-to-end generative conversation models, where neural E-D approaches have been first applied by Vinyals and Le (2015) and extended by many others since.

Many works address the lack of diversity and coherence in E-D outputs (Sountsov and Sarawagi, 2016; Wei et al., 2017) but do not attempt to model coherence directly, unlike our work: Li et al. (2016a) use anti-LM reranking; Li et al. (2016c) modify the beam search decoding algorithm, similar to Shao et al. (2017) in addition to using a self-attention model. Mou et al. (2016) predict keywords for the output in a preprocessing step while Wu et al. (2018) preselect a vocabulary subset to be used for decoding. Li et al. (2016b) focus specifically on personality generation (using personality embeddings) and Wang et al. (2017) promote topic-specific outputs by language-model rescoring and sampling.

A lot of recent works explore the use of additional training signals and VAE setups in dialogue generation. In contrast to this paper, they do not focus explicitly on coherence: Asghar et al. (2017) use reinforcement learning with human-provided feedback, Li et al. (2017a) use a RL scenario with length as reward signal. Li et al. (2017b) add an adversarial discriminator to provide RL rewards (discriminating between human and machine outputs), Xu et al. (2017) use a full adversarial training setup. The most recent works explore the usage of VAEs: Cao and Clark (2017) explore a vanilla VAE setup conditioned on dual encoder (for contexts and responses) during training, the model of Serban et al. (2017) uses a VAE in a hierarchical E-D model. Shen et al. (2017) use a cVAE conditioned on sentiment and response genericity (based on a handwritten list of phrases). Shen et al. (2018) combine a cVAE with a plain VAE in an adversarial fashion.

We also draw on ideas from other areas than dialogue generation to build our models: Tu et al. (2017)'s context gates originate from machine translation and Hu et al. (2017)'s cVAE training stems from free-text generation.

7 Conclusions and Future Work

We showed that explicitly modeling coherence and optimizing towards coherence and diversity leads to better-quality outputs in dialogue response generation. We introduced three extensions to current encoder-decoder response generation models: (1) we defined a measure of coherence based on GloVe embeddings (Pennington et al., 2014), (2) we filtered the OpenSubtitles training corpus (Lison and Meena, 2016) based on this measure to obtain coherent and diverse training instances, (3) we trained a cVAE model based on (Hu et al., 2017) and (Tu et al., 2017) that uses our coherence measure as one of the training signals. Our experimental results showed a considerable improvement in the output quality over competitive models, which demonstrates the effectiveness of our approach.

In future work, we plan to replace the GloVe-based measure of coherence with a trained discriminator that distinguishes between coherent and incoherent responses (Li and Jurafsky, 2017). This will allow us to use extend the notion of coherence to account for phenomena such as topic shifts. We also plan to verify the results with a human evaluation study.

Acknowledgements

This research received funding from the EPSRC project MaDrIgAL (EP/N017536/1). The Titan Xp used for this research was donated by the NVIDIA Corporation.

References

Nabiha Asghar, Pascal Poupart, Xin Jiang, and Hang Li. 2017. Deep Active Learning for Dialogue Generation. In *6th Joint Conference on Lexical and Computational Semantics (*SEM) 2017*. ArXiv: 1612.03929.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations (ICLR2015)*, San Diego, CA, USA. ArXiv: 1409.0473.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Kris Cao and Stephen Clark. 2017. Latent Variable Dialogue Models and their Diversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 182–187, Valencia, Spain. Association for Computational Linguistics.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 484–495.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen tau Yih, and Michel Galley. 2017. A knowledge-grounded neural conversation model. *arXiv preprint arXiv:1702.01932*.

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, CA, USA. ArXiv: 1412.6980.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016b. A Persona-Based Neural

Conversation Model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany. ArXiv: 1603.06155.

Jiwei Li and Dan Jurafsky. 2017. Neural Net Models for Open-Domain Discourse Coherence. In *Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. ArXiv: 1606.01545.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016c. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017a. Learning to decode for future success. *arXiv preprint arXiv:1701.06549*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017b. Adversarial Learning for Neural Dialogue Generation. *arXiv:1701.06547 [cs]*. ArXiv: 1701.06547.

Pierre Lison and Raveesh Meena. 2016. Automatic Turn Segmentation for Movie & TV Subtitles. In *2016 IEEE Workshop on Spoken Language Technology*. IEEE conference proceedings.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. ArXiv: 1508.04025.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to Backward and Forward Sequences: A Content-Introducing Approach to Generative Short-Text Conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358, Osaka, Japan. The COLING 2016 Organizing Committee. ArXiv: 1607.00970.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Iulian Serban, Alessandro Sordoni, Ryan Joseph Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating Long and Diverse Responses with Neural Conversation Models. In *Empirical Methods in Natural Language Processing*, pages 2200–2009, Copenhagen, Denmark. ArXiv: 1701.03185.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A Conditional Variational Framework for Dialog Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, Vancouver, Canada. ArXiv: 1705.00316.

Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018. Improving Variational Encoder-Decoders in Dialogue Generation. In *AAAI 2018*. ArXiv: 1802.02032.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL*.

Pavel Sountsov and Sunita Sarawagi. 2016. Length bias in Encoder Decoder Models and a Case for Global Conditioning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1525, Austin, TX, USA. Association for Computational Linguistics.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context Gates for Neural Machine Translation. *Transactions of the Association for Computational Linguistics*, 5:87–99.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2140–2150, Copenhagen, Denmark.

Bolin Wei, Shuai Lu, Lili Mou, Hao Zhou, Pascal Poupart, Ge Li, and Zhi Jin. 2017. Why Do Neural Dialog Systems Generate Short and Meaningless Replies? A Comparison between Dialog and Translation. *arXiv:1712.02250 [cs]*. ArXiv: 1712.02250.

Yu Wu, Wei Wu, Dejian Yang, Can Xu, Zhoujun Li, and Ming Zhou. 2018. Neural Response Generation with Dynamic Vocabularies. In *AAAI 2018*, New Orleans, LA, USA. ArXiv: 1711.11191.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, pages 3351–3357.

Zhen Xu, Bingquan Liu, Baoxun Wang, SUN Chengjie, Xiaolong Wang, Zhuoran Wang, and Chao Qi. 2017. Neural response generation via gan with an approximate embedding layer. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 628–637.

Lili Yao, Yaoyuan Zhang, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. Towards implicit content-introducing for generative short-text conversation systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2180–2189.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *ACL*, Vancouver, Canada. ArXiv: 1703.10960.

Incorporating Background Knowledge into Video Description Generation

Spencer Whitehead[1], **Heng Ji**[1], **Mohit Bansal**[2], **Shih-Fu Chang**[3], **Clare R. Voss**[4]
[1] Computer Science Department, Rensselaer Polytechnic Institute
`{whites5,jih}@rpi.edu`
[2] Computer Science Department, University of North Carolina at Chapel Hill
`mbansal@cs.unc.edu`
[3] Department of Electrical Engineering, Columbia University
`sfchang@ee.columbia.edu`
[4]US Army Research Laboratory
`clare.r.voss.civ@mail.mil`

Abstract

Most previous efforts toward video captioning focus on generating generic descriptions, such as, “A man is talking.” We collect a news video dataset to generate enriched descriptions that include important background knowledge, such as named entities and related events, which allows the user to fully understand the video content. We develop an approach that uses video meta-data to retrieve topically related news documents for a video and extracts the events and named entities from these documents. Then, given the video as well as the extracted events and entities, we generate a description using a *Knowledge-aware Video Description* network. The model learns to incorporate entities found in the topically related documents into the description via an *entity pointer network* and the generation procedure is guided by the event and entity types from the topically related documents through a *knowledge gate*, which is a gating mechanism added to the model’s decoder that takes a one-hot vector of these types. We evaluate our approach on the new dataset of news videos we have collected, establishing the first benchmark for this dataset as well as proposing a new metric to evaluate these descriptions.

1 Introduction

Video captioning is a challenging task that seeks to automatically generate a natural language description of the content of a video. Many video captioning efforts focus on learning video representations that model the spatial and temporal dynamics of the videos (Yao et al., 2015; Venugopalan et al., 2016; Yu et al., 2017). Although the language generation component within this task is of great importance, less work has been done to enhance the contextual knowledge conveyed by the descriptions. The descriptions generated by previous methods tend to be “generic”, describing only what is evidently visible and lacking specific knowledge, like named entities and event participants, as shown in Figure 1a. In many situations, however, generic descriptions are uninformative as they do not provide contextual knowledge. For example, in Figure 1b, details such as *who is speaking* or *why they are speaking* are imperative to truly understanding the video, since contextual knowledge gives the surrounding circumstances or cause of the depicted events.

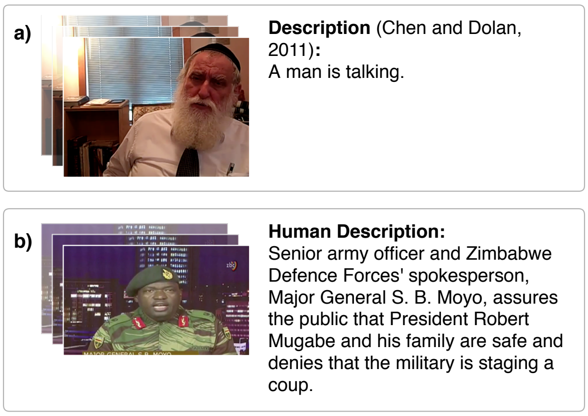

Description (Chen and Dolan, 2011)**:**
A man is talking.

Human Description:
Senior army officer and Zimbabwe Defence Forces' spokesperson, Major General S. B. Moyo, assures the public that President Robert Mugabe and his family are safe and denies that the military is staging a coup.

Figure 1: Comparison of machine (a) and human (b) generated descriptions.[1]

To address this problem, we collect a news video dataset, where each video is accompanied by meta-data (*e.g.*, tags and date) and a natural language description of the content in, and/or context around, the video. We create an approach to this task that is motivated by two observations.

First, the video content alone is insufficient to generate the description. Named entities or specific events are necessary to identify the participants, location, and/or cause of the video content. Although knowledge could potentially be mined from visual evidence (*e.g.*, recognizing the location), training such a system is exceedingly diffi-

---

[1] (a) https://goo.gl/2StcD8, (b) https://goo.gl/VFR5nw

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3992–4001
Brussels, Belgium, October 31 - November 4, 2018.

cult (Tran et al., 2016). Further, not all the knowledge necessary for the description may appear in the video. In Figure 2a, the video depicts much of the description content, but knowledge of the speaker ("*Carles Puigdemont*") is unavailable if limited to the visual evidence because the speaker never appears in the video, making it intractable to incorporate this knowledge into the description.

Second, one may use a video's meta-data to retrieve topically related news documents that contain the named entities or events that appear in the video's description, but these may not be specific to the video content. For example, in Figure 2b, the video discusses the "*heightened security*" and does not depict the arrest directly. Topically related news documents capture background knowledge about the attack that led to the "*heightened security*" as well as the arrest, but they may not describe the actual video content, which displays some of the increased security measures.

Thus, we propose to retrieve topically related news documents from which we seek to extract named entities (Pan et al., 2017) and events (Li et al., 2013) likely relevant to the video. We then propose to use this knowledge in the generation process through an *entity pointer network*, which learns to dynamically incorporate extracted entities into the description, and through a new *knowledge gate*, which conditions the generator on the extracted event and entity types. We include the video content in the generation by learning video representations using a spatio-temporal hierarchical attention that spatially attends to regions of each frame and temporally attends to different frames. We call the combination of these generation components the *Knowledge-aware Video Description* (KaVD) network. The contributions of this paper are as follows:

- We create a knowledge-rich video captioning dataset, which can serve as a new benchmark for future work.
- We propose a new *Knowledge-aware Video Description* network that can generate descriptions using the video and background knowledge mined from topically related documents.
- We present a knowledge reconstruction based metric, using entity and event F1 scores, to evaluate the correctness of the knowledge conveyed in the generated descriptions.

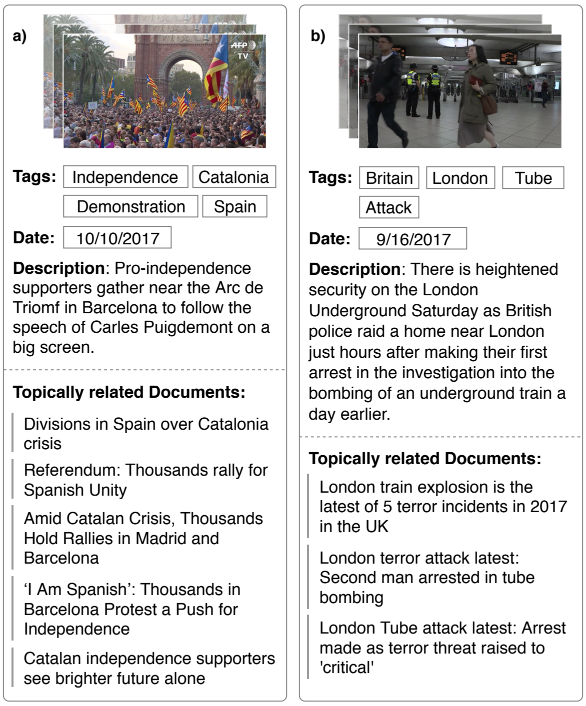

Tags: Independence | Catalonia | Demonstration | Spain

Date: 10/10/2017

Description: Pro-independence supporters gather near the Arc de Triomf in Barcelona to follow the speech of Carles Puigdemont on a big screen.

Topically related Documents:

Divisions in Spain over Catalonia crisis

Referendum: Thousands rally for Spanish Unity

Amid Catalan Crisis, Thousands Hold Rallies in Madrid and Barcelona

'I Am Spanish': Thousands in Barcelona Protest a Push for Independence

Catalan independence supporters see brighter future alone

Tags: Britain | London | Tube | Attack

Date: 9/16/2017

Description: There is heightened security on the London Underground Saturday as British police raid a home near London just hours after making their first arrest in the investigation into the bombing of an underground train a day earlier.

Topically related Documents:

London train explosion is the latest of 5 terror incidents in 2017 in the UK

London terror attack latest: Second man arrested in tube bombing

London Tube attack latest: Arrest made as terror threat raised to 'critical'

Figure 2: Examples from the news video dataset (video, meta-data, and description) with some retrieved topically related documents.[2]

2 Approach

Figure 3 shows our overall approach. We first retrieve topically related news documents using tags from the video meta-data. Next, we apply entity discovery and linking as well as event extraction methods to the documents, which yields a set of entities and events relevant to the video. We represent this background knowledge in two ways: 1) we encode the entities through entity embeddings and 2) we encode the event and entity typing information into a *knowledge gate vector*, which is a one-hot vector where each entry represents an entity or event type. Finally, with the video and these representations of the background knowledge, we employ our KaVD network, an encoder-decoder (Cho et al., 2014) style model, to generate the description.

2.1 Document Retrieval and Knowledge Extraction

We gather topically related news documents as a source of background knowledge using the video meta-data. For each video, we use the corresponding tags to perform a keyword search on documents from a number of popular news outlet web-

---

[2] (a) https://goo.gl/3cF1oU, (b) https://goo.gl/NkwHvJ

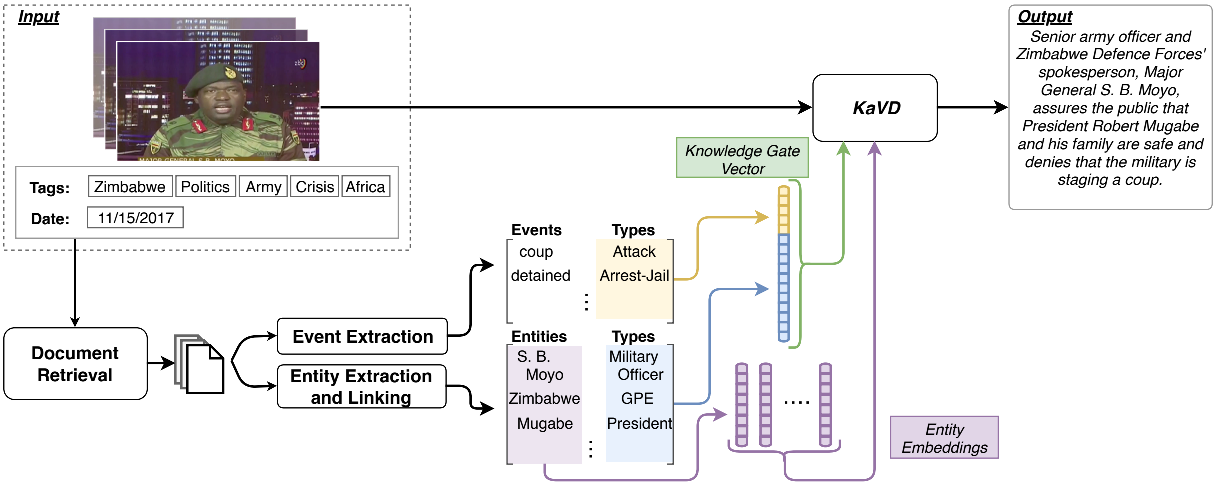

Figure 3: Overall pipeline of our approach.

sites.[3] We filter these documents by the date associated with video, only keeping documents that are written within d days before and after the video upload date.[4] The keyword search gathers documents that are at least somewhat topically relevant and filtering by date increases the likelihood that the documents reference the specific events and entities of the video, since the occurrences of entity and event mentions across news documents tend to be temporally correlated. We retrieve an average of 3.1 articles per video and find that on average 68.8% of the event types and 70.6% of the entities in the ground truth description also appear in corresponding news articles. In Figure 3, the retrieved background documents include the entity "*Mugabe*" and the event "*detained*", which are relevant to the video description.

We apply a high-performing, publicly available entity discovery and linking system (Pan et al., 2017) to extract named entities and their types. This system is able to discover entities and link them to rich knowledge bases that provide fine-grained types that we can exploit to better discern between entities in the news documents (*e.g.*, "`President`" versus "`Military Officer`").[5] Additionally, we use a high-performing event extraction system (Li et al., 2013) to extract events and their arguments. For example, in Figure 3, we get entities "*S. B. Moyo*", "*Zimbabwe*", and "*Mugabe*" with their respective types, "`Military Officer`", "`GPE`", and "`President`". Likewise, we obtain events "*coup*" and "*detained*" with their respective types, "`Attack`" and "`Arrest-Jail`". The entities and events along with their types provide valuable insight into the context of the video and can bias the decoder to generate the correct event mentions and incorporate the proper entities.

We encode the entities and events into representations that can be fed to the model. First, we obtain an entity embedding, $\mathbf{e}_m$, for each entity by averaging the embeddings of the words in the entity mention. Second, we encode the entity and event types into a one-hot knowledge gate vector, $\mathbf{k}_0$. Each element of $\mathbf{k}_0$ corresponds to an event or entity type (*e.g.*, "`Arrest-Jail`" event type or "`President`" entity type), so the j^{th} element, $k^{(j)}$, is 1 if the entity or event type is found in the related documents and 0 otherwise. $\mathbf{k}_0$ serves as the initial knowledge gate vector of the decoder (Section 2.2). The entity embeddings give the model access to semantic representations of the entities, while the knowledge gate vector aids the generation process by providing the model with the event and entity types.

2.2 KaVD Network

Our model learns video representations using hierarchical, or multi-level, attention (Yang et al., 2016; Qin et al., 2017). The encoder is comprised of a spatial attention (Xu et al., 2015) and bidirectional Long Short-Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997) temporal encoder. The spatial attention allows the model to attend to different locations of each frame (Figure 4), yielding frame representations

---

[3] BBC, CNN, and New York Times.

[4] $d = 3$ in our experiments.

[5] We only use types that appear in the training data and are within 4 steps from the top of the 7,309 type hierarchy *here*.

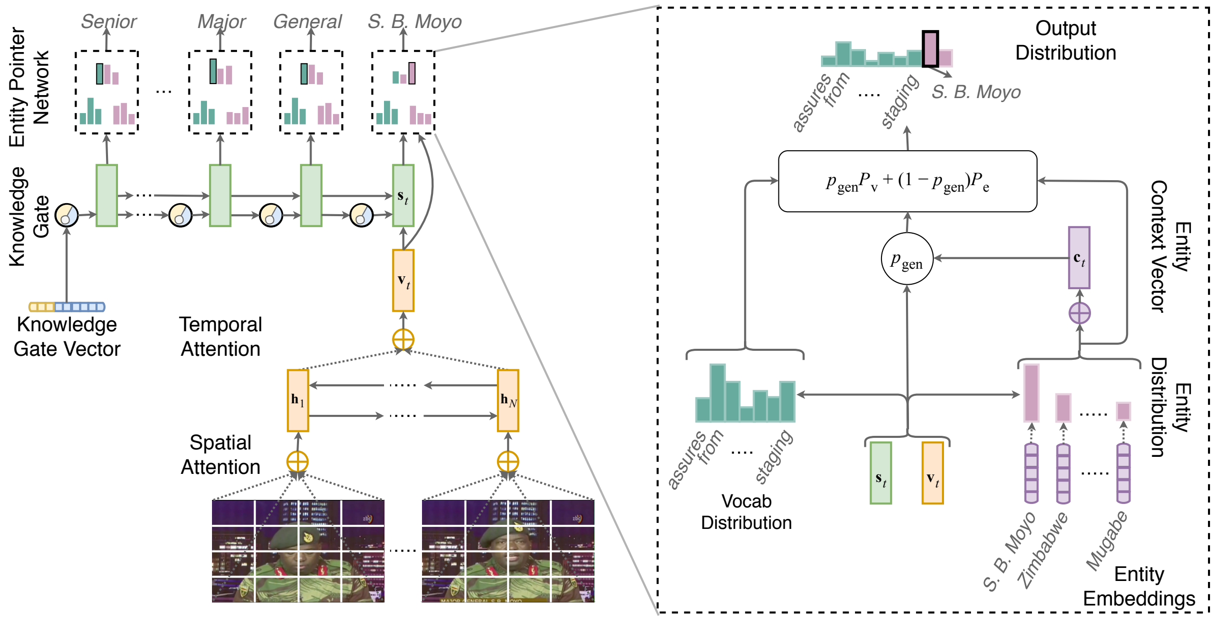

Figure 4: KaVD Network. At each decoder time step, the model computes p_{gen} to determine whether to emit a vocabulary word or a named entity from the topically related documents.

that emphasize the most important regions of each frame. The temporal encoder incorporates motion into the frame representations by encoding information from the preceding and subsequent frames (Yao et al., 2015). We use a LSTM decoder, which applies a temporal attention (Bahdanau et al., 2015) to the frame representations at each step. To generate each word, the decoder computes its hidden state, adjusts this hidden state with the *knowledge gate* output at the current time step, and determines the most probable word by utilizing the *entity pointer network* to decide whether to generate a named entity or vocabulary word. Pointer networks are effective at incorporating out-of-vocabulary (OOV) words in output sequences (Miao and Blunsom, 2016; See et al., 2017). In previous research, OOV words may appear in the input sequence, in which case they are copied into the output. Analogously, in our approach, named entities can be considered as OOV words that are from a separate set instead of the input sequence. In the following equations, where appropriate, we omit bias terms for brevity.

Encoder. The input to the encoder is a sequence of video frames, $\{F_1, ..., F_N\}$. First, we extract frame-level features by applying a Convolutional Neural Network (CNN) (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Ioffe and Szegedy, 2015; Szegedy et al., 2015, 2017) to each frame, F_i, and obtaining the response of a convolutional layer, $\{\mathbf{a}_{i,1}, ..., \mathbf{a}_{i,L}\}$, where $\mathbf{a}_{i,l}$ is a D-dimensional representation of the l^{th} location of the i^{th} frame (*e.g.*, the top left box of the first frame in Figure 4). We apply a spatial attention to these location representations, given by

$$\alpha_{i,l} = a_{\text{space}}\left(\mathbf{a}_{i,l}\right) \tag{1}$$

$$\xi_{i,l} = \text{softmax}\left(\alpha_{i,l}\right) \tag{2}$$

$$\mathbf{z}_i = \sum_{l=1}^{L} \xi_{i,l}\mathbf{a}_{i,l} \tag{3}$$

where a_{space} is a scoring function (Bahdanau et al., 2015). Frame representations $\{\mathbf{z}_1, ..., \mathbf{z}_N\}$ are input to a bi-directional LSTM, producing temporally encoded frame representations $\{\mathbf{h}_1, ..., \mathbf{h}_N\}$.

Decoder. The decoder is an attentive LSTM cell with the addition of a *knowledge gate* and *entity pointer network*. At each decoder step t, we apply a temporal attention to the frame representations,

$$\beta_{t,i} = a_{\text{time}}\left(\mathbf{h}_i, \mathbf{s}_{t-1}\right) \tag{4}$$

$$\eta_{t,i} = \text{softmax}\left(\beta_{t,i}\right) \tag{5}$$

$$\mathbf{v}_t = \sum_{i=1}^{N} \eta_{t,i}\mathbf{h}_i \tag{6}$$

where $\mathbf{s}_{t-1}$ is the previous decoder hidden state and a_{time} is another scoring function. This yields a single, spatio-temporally attentive video representation, $\mathbf{v}_t$. We then compute an intermediate hidden state, $\hat{\mathbf{s}}_t$, by applying the decoder LSTM to $\mathbf{s}_{t-1}$, $\mathbf{v}_t$, and previous word embedding, $\mathbf{x}_{t-1}$. The final decoder hidden state is determined after the knowledge gate computation.

The motivation for the knowledge gate is that it biases the model to generate sentences that contain specific knowledge relevant to the video and topically related documents, acting as a kind of coverage mechanism (Tu et al., 2016). For example, given the retrieved event types in Figure 3, the knowledge gate encourages the decoder to generate the event trigger "*coup*" due to the presence of the "`Attack`" event type. Inspired by the gating mechanisms from natural language generation (Wen et al., 2015; Tran and Nguyen, 2017), the knowledge gate, $\mathbf{g}_t$, is given by

$$\mathbf{g}_t = \sigma\left(\mathbf{W}_{g,v}[\mathbf{x}_{t-1}, \mathbf{v}_t] + \mathbf{W}_{g,s}\hat{\mathbf{s}}_t\right) \quad (7)$$

$$\mathbf{k}_t = \mathbf{g}_t \odot \mathbf{k}_{t-1} \quad (8)$$

where all $\mathbf{W}$ are learned parameters and $[\mathbf{x}_{t-1}, \mathbf{v}_t]$ is the concatenation of these two vectors. This gating step determines the amount of the entity and event type features contained in $\mathbf{k}_{t-1}$ to carry to the next step. With the updated $\mathbf{k}_t$, we compute the decoder hidden state, $\mathbf{s}_t$, as

$$\mathbf{s}_t = \hat{\mathbf{s}}_t + (\mathbf{o}_t \odot \tanh\left(\mathbf{W}_{s,k}\mathbf{k}_t\right)) \quad (9)$$

where $\mathbf{o}_t$ is the output gate of the LSTM and $\mathbf{W}_{s,k}$ is a learned parameter.

Our next step is to generate the next word. The model needs to produce named entities (*e.g.*, "*S. B. Moyo*" and "*Robert Mugabe*") throughout the generation process. These named entities tend to occur rarely if at all in many datasets, including ours. We overcome this issue by using the entity embeddings from the topically related documents as potential entities to incorporate in the description. We adopt a soft switch pointer network (See et al., 2017), as our entity pointer network, to perform the selection of generating words or entities.

For our entity pointer network to predict the next word, we first predict a vocabulary distribution, $P_{\text{v}} = \psi\left(\mathbf{s}_t, \mathbf{v}_t\right)$, where $\psi(\cdot)$ is a softmax output layer. $P_{\text{v}}(w)$ is the probability of generating word w from the decoder vocabulary. Next, we compute an entity context vector, $\mathbf{c}_t$, using a soft attention mechanism:

$$\gamma_{t,m} = a_{\text{entity}}\left(\mathbf{e}_m, \mathbf{s}_t, \mathbf{v}_t\right) \quad (10)$$

$$\epsilon_{t,m} = \text{softmax}\left(\gamma_{t,m}\right) \quad (11)$$

$$\mathbf{c}_t = \sum_{m=1}^{M} \epsilon_{t,m}\mathbf{e}_m \quad (12)$$

Here, a_{entity} is yet another scoring function. We use the scalars $\epsilon_{t,m}$ as our entity probability distribution, P_{e}, where $P_{\text{e}}\left(E_m\right) = \epsilon_{t,m}$ is the probability of generating entity mention E_m. We compute the probability of generating a word from the vocabulary, p_{gen}, as

$$p_{\text{gen}} = \sigma(\mathbf{w}_c^\top\mathbf{c}_t + \mathbf{w}_s^\top\mathbf{s}_t + \mathbf{w}_x^\top\mathbf{x}_{t-1} + \mathbf{w}_v^\top\mathbf{v}_t) \quad (13)$$

where all $\mathbf{w}$ are learned parameters. Finally, we predict the probability of word w by

$$P(w) = p_{\text{gen}}P_{\text{v}}(w) + (1 - p_{\text{gen}})P_{\text{e}}(w) \quad (14)$$

and select the word of maximum probability. In Equation 14, $P_{\text{e}}(w)$ is 0 when w is not a named entity. Likewise, P_{v} is 0 when w is an OOV word. For the example in Figure 4, the vocabulary distribution, P_{v}, has the word "*from*" as the most probable word and the entity distribution, P_{e}, has the entity "*S. B. Moyo*" as the most probable entity. However, by combining these two distribution using p_{gen}, the model switches to the entity distribution and correctly generates "*S. B. Moyo*".

3 News Video Dataset

Current datasets for video description generation focus on specific (Rohrbach et al., 2014) and general (Chen and Dolan, 2011; Xu et al., 2016) domains, but do not contain a large proportion of descriptions with specific knowledge like named entities as shown in Table 1. In our news video dataset, the descriptions are replete with important knowledge that is both necessary and challenging to incorporate into the generated descriptions.

Our news video dataset contains AFP international news videos from YouTube.[6] These videos are from October, 2015 to November, 2017 and cover a variety of topics, such as protests, attacks, natural disasters, trials, and political movements. The videos are "on-the-scene" and contain some depiction of the content in the description. For each video, we take the YouTube descriptions given by AFP News as the ground-truth descriptions we wish to generate. We collect the tags and meta-data (*e.g.*, upload date). We filter videos by length, with a cutoff of 2 minutes, and remove videos which are videographics or animations. For preprocessing, we tokenize each sentence, remove punctuation characters other than

---

[6] https://www.youtube.com/user/AFP

Dataset	Domain	#Videos	#Sentences	Vocab Size	Named Entities/Sentence
TACos M-L (Rohrbach et al., 2014)	Cooking	14,105	52,593	2,864	0.1×10^{-4}
MSVD (Chen and Dolan, 2011)	Multi-category	1,970	70,028†	13,010	0.4×10^{-2}
MSR-VTT-10K (Xu et al., 2016)	20 categories	10,000	200,000†	29,316	1.4×10^{-1}
News Video (Ours)	News	2,883	3,302	9,179	2.1

Table 1: Comparison of our news video dataset to other datasets. † indicates that the dataset has multiple, single-sentence reference descriptions for each video.

periods, commas, and apostrophes, and replace numerical quantities and dates/times with special tokens. We sample frames at a rate of $1fps$. We randomly select 400 videos for testing, 80 for validation, and 2,403 for training. We make the dataset publicly available: https://goo.gl/2jScKk.

4 Experiments

4.1 Model Comparisons

We test our method against the following baselines: **Article-only.** We use the summarization model of See et al. (2017) to generate the description by summarizing the topically related documents. **Video-only (VD).** We train a model that does not receive any background knowledge and generates the description directly from the video. **VD with knowledge gate only (VD+Knowledge Gate), VD with entity pointer network only (VD+Entity Pointer), and no-video (Entity Pointer+Knowledge Gate).** These test the effects of the knowledge gate, entity pointer network, and video encoder in isolation.

Each model uses a cross entropy loss. Video-based models are trained using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.0002 and have a hidden state size of 512 as well as an embedding size of 300. We use Google News pre-trained word embeddings (Mikolov et al., 2013) to initialize our word embeddings and compute entity embeddings. For visual features, we use the Conv3-512 layer response of VGGNet (Simonyan and Zisserman, 2014) pre-trained on ImageNet (Deng et al., 2009).

4.2 Evaluations

METEOR (Denkowski and Lavie, 2014) and ROUGE-L (Lin, 2004) are adopted as metrics for evaluating the generated descriptions. We choose METEOR because we only have one reference description per video and this metric accounts for stemming and synonym matching. We also use ROUGE-L for comparison to summarization work. These capture the coherence and relevance of the generated descriptions to the ground truth.

Generating these descriptions is concerned with not only generating fluent text, but also the amount of knowledge conveyed and the accuracy of the knowledge elements (*e.g.*, named entities or event structures). Previous work in natural language generation and summarization (Nenkova and Passonneau, 2004; Novikova et al., 2017; Wiseman et al., 2017; Pasunuru and Bansal, 2018) scores and/or assigns weights to overlapping text, salient phrases, or information units (*e.g.*, entity relations (Wiseman et al., 2017)). However, knowledge elements cannot be simply represented as a set of isolated information units since they are inherently interconnected through some structure.

Therefore, for this knowledge-centric generation task, we compute F1 scores on event and entity extraction results from the generated descriptions against the extraction results on the ground truth. For entities, we measure the F1 score of the named entities in the generated description compared to the ground truth. For events, given a generated description, w^s, and the ground truth description, w^c, we extract a set of event structures, $\mathcal{Y}^s$ and $\mathcal{Y}^c$, for both descriptions such that $\mathcal{Y} = \{(t_k, r_{k,1}, a_{k,1}, ..., r_{k,m}, a_{k,m})\}_{k=1}^{K}$ where there are K events extracted from the description, t_k is the k^{th} event type, $r_{k,m}$ is the m^{th} argument role of t_k, and $a_{k,m}$ is the m^{th} argument of t_k. For the description in Figure 2a, one may obtain:

$$\mathcal{Y} = \{(\texttt{Demonstrate}, \texttt{Entity}, \text{“\textit{Pro-independence supporters}”}, \texttt{Place}, \text{“\textit{Barcelona}”})\}$$

Next, we form event type, argument role, and argument triples $(t^s_k, r^s_{k,m}, a^s_{k,m})$ and $(t^c_j, r^c_{j,m}, a^c_{j,m})$ for each event structure in $\mathcal{Y}^s$ and $\mathcal{Y}^c$, respectively. We compute the F1 score of the triples, considering a triple correct if and only if it appears in the ground truth triples.[7] This metric enables us to evaluate how well a generated description captures the overall events, while still giving credit to partially correct event structures. We compute these F1 scores on 50 descriptions based on manually annotated event structures. We also perform automatic F1 score evaluation on the entire test set using the entity and event extraction systems of Pan et al. (2017) and Li et al. (2013), respectively. The manual evaluations offer accurate comparisons and control for correctness, while the automated evaluations explore the viability of using automated IE tools to measure performance, which is desirable for scaling to larger datasets for which manual evaluations are too expensive.

5 Results and Analysis

The KaVD network outperforms almost all of the baselines, as shown in Table 2, achieving statistically significant improvements in METEOR and ROUGE-L w.r.t. all other models besides the no-video model ($p < 0.05$).[8] The additions of the entity pointer network and knowledge gate are complementary and greatly improve the entity incorporation performance, increasing the entity F1 scores by at least 6% in both the manual and automatic evaluations. In Figure 5a, the entity pointer network is able to incorporate the entity "*Abdiaziz Abu Musab*", who is a leader of the group responsible for the attack. We find that the entity and event type features from the knowledge gate help generate more precise entities. However, noise in the article retrieval process and entity extraction system limits our entity incorporation capabilities, since on average only 70.6% of the entities in the ground truth description are retrieved from the articles. Lastly, the video encoder helps generate the correct events and offers qualitative benefits, such as allowing the model to generate more concise and diverse descriptions, though it negatively affects the entity incorporation performance.

The video alone is insufficient to generate the correct entities (Table 2). In Figure 5a, the VD baseline generates the correct event, but generates the incorrect location "*Kabul*". We observe that when the visual evidence is ambiguous, this model may fail to generate the correct events and entities. For example, if a video depicts the destruction of buildings after a hurricane, then the VD baseline may mistakenly describe the video as an explosion since the visual evidence is similar.

The article-only baseline tends to mention the correct entities as shown in Figure 5a, where the description is generally on topic but provides some irrelevant information. Indeed, this model can generate descriptions unrelated to the video itself. In Figure 5b, the article-only baseline's description contains some correct entities (*e.g.*, "*Colombia*"), but is not focused on the announcement depicted in the video. As See et al. (2017) discuss, this model can be more extractive than abstractive, copying many sequences from the documents. This can lead to irrelevant descriptions as the articles may not be specific to the video.

Our entity and event F1 score based metrics correlate well with the correctness of the knowledge conveyed in the generated description. The consistency in model rankings between the manual and automatic entity metrics shows the potential of using automated entity extraction approaches to evaluate with this metric. We observe discrepancies between the manual and automatic event metrics, in part, due to errors in the automated extraction and the addition of more test points. For example, in the generated sentence, "*Hundreds of people are to take to the streets of...*", the event extraction system mistakenly assigns a "`Transport`" event type instead of the correct "`Demonstrate`" event type. In contrast, such mistakes do not appear in the manual evaluations.

6 Related Work

Most previous video captioning efforts focus on learning video representations through different encoding techniques (Venugopalan et al., 2015a,b), using spatial or temporal attentions (Yao et al., 2015; Pan et al., 2016; Yu et al., 2016; Zanfir et al., 2016), using 3D CNN features (Tran et al., 2015; Yao et al., 2015; Pan et al., 2016), or easing the learning process via multi-task learning or reinforcement rewards (Pasunuru and Bansal, 2017a,b). Compared to other hierarchical models (Pan et al., 2016; Yu et al., 2016), each level of our hierarchy encodes a different dimension of the video, leveraging global temporal features and local spatial features, which are shown to be effective for different tasks (Ballas et al., 2015; Xu et al., 2015; Yu et al., 2017).

We move towards using datasets with captions that have specific knowledge rather than generic

---

[7] This criterion is used for computing precision and recall.

[8] Found via paired bootstrap resampling (Koehn, 2004).

Model	METEOR	ROUGE-L	Entity F1	Auto-Entity F1	Event F1	Auto-Event F1
Article-only	8.6	13.2	8.7	8.5	1.9	3.6
VD	9.1	17.9	2.5	1.5	1.0	7.3
VD+Entity Pointer	9.7	18.1	15.3	13.6	5.7	7.0
VD+Knowledge Gate	9.8	18.5	10.2	10.7	6.7	8.3
Entity Pointer+Knowledge Gate	10.1	18.7	**23.7**	**20.9**	2.2	**9.9**
KaVD	**10.2**	**18.9**	22.1	19.7	**9.6**	8.9

Table 2: METEOR, ROUGE-L, and manual/automated entity (Entity F1/Auto-Entity F1) and event (Event F1/Auto-Event F1) F1 score results of the baselines and KaVD network on our news video dataset.

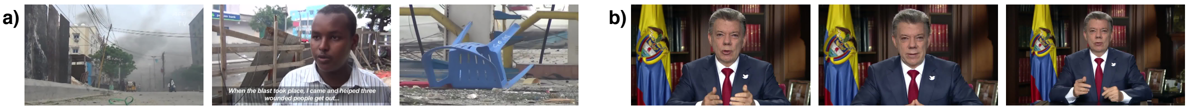

a) ***Title:*** Deadly Shabaab attack rocks Somali capital

b) ***Title:*** Santos: 'Green light' for referendum on Colombia peace deal

Model	**Description**
Article-only	somali capital mogadishu on saturday. at least 276 people have died and the government news agency sonna says only 111 of them have been identified. a turkish military but instead witnessed her burial. no group has yet said it was behind on instead he attended her burial. "anfa'a said she had spoken to her sister 20 minutes before on
VD	a suicide bomber killed # people in a bus carrying # people killed in a bus in central kabul.
VD+Entity Pointer	A suicide bomber killed # people were killed in a bus near the northern city of Mogadishu, police said.
VD +Knowledge Gate	At least # people were killed and # wounded when a busy bus station in Kabul, killing at least # people dead and others who died in the rubble of the deadliest attack in the country.
EntityPointer +Knowledge Gate	At least # people were killed in a suicide car bomb attack on a suicide car bomb attack on a police vehicle in Mogadishu, police said.
KaVD	A suicide bombing claimed by the Abdiaziz Abu Musab group `time` killed # people in Somalia's capital Mogadishu, killing # people, officials said.

Model	**Description**
Article-only	colombia's marxist rebels against her family. and last year, when given the leg of helena gonzlez's nephew years ago is still fresh the as pope francis arrived in colombia on wednesday for a six-day the
VD	president donald trump says that he will be talks to be to be talks to be talks in the country's country to be talks, saying he says he would be no evidence's state and kerry says.
VD+Entity Pointer	President Maduro says the FARC president warns that the ceasefire to Prime Minister says that he will be ready to help President Maduro says that he is no evidence of President Bashar talks in Bogota.
VD +Knowledge Gate	US Secretary of State John Kerry, who will not any maintain in Syria, after a ceasefire in Syria, saying that the United Nations says, it will not to be into a speech in its interview.
EntityPointer +Knowledge Gate	Venezuela's President FARC envoy to Colombia is a definitive ceasefire in the FARC conflict, with FARC rebels, the FARC rebels.
KaVD	Colombia's government, signed the peace agreement with the FARC peace accord in the FARC rebels.

Figure 5: Comparison of generated descriptions. The KaVD network generates the correct entities and correct events, while other models may contain some wrong entities or wrong events.

captions as in previous work (Chen and Dolan, 2011; Rohrbach et al., 2014; Xu et al., 2016). There are efforts in image captioning to personalize captions (Park et al., 2017), incorporate novel objects into captions (Venugopalan et al., 2016), and perform open domain captioning (Tran et al., 2016). To the best of our knowledge, our dataset is the first of its kind and offers challenges in entity and activity recognition as well as the generation low probability words. Datasets with captions rich in knowledge elements, like those in our dataset, take a necessary step towards increasing the utility of video captioning systems.

We employ similar approaches to those in automatic summarization, where pointer networks (Vinyals et al., 2015) and copy mechanisms (Gu et al., 2016) are used (Gulcehre et al., 2016; Nallapati et al., 2016; Miao and Blunsom, 2016; See et al., 2017), and natural language generation for dialogue systems (Wen et al., 2015; Tran and Nguyen, 2017). The KaVD network combines the copying capabilities of pointer networks (See et al., 2017) and semantic control of gating mechanisms (Wen et al., 2015; Tran and Nguyen, 2017) in a complementary fashion to address a new, multi-modal task.

7 Conclusions and Future Work

We collect a news video dataset with knowledge-rich descriptions and present a multi-modal approach to this task that uses a novel Knowledge-aware Video Description network, which can utilize background knowledge mined from topically related documents. We offer a new metric to measure a model's ability to incorporate named entities and specific events into the descriptions. We show the effectiveness of our approach and set a new benchmark for this dataset. In future work, we are increasing the size of dataset and exploring other knowledge-centric metrics for this task.

Acknowledgments

This work was supported by the U.S. DARPA AIDA Program No. FA8750-18-2-0014 and U.S. ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. 2015. Delving deeper into convolutional networks for learning video representations. In *ICLR*.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *ACL*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *WMT*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Text summarization branches out: ACL workshop*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *EMNLP*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshops at ICLR*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *NAACL*.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *EMNLP*.

Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. 2016. Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *ACL*.

Cesc Chunseong Park, Byeongchang Kim, and Gunhee Kim. 2017. Attend to you: Personalized image captioning with context sequence memory networks. In *CVPR*.

Ramakanth Pasunuru and Mohit Bansal. 2017a. Multi-task video captioning with video and entailment generation. In *ACL*.

Ramakanth Pasunuru and Mohit Bansal. 2017b. Reinforced video captioning with entailment rewards. In *EMNLP*.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *NAACL*.

Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. In *IJCAI*.

Anna Rohrbach, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. In *GCPR*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*.

Kenneth Tran, Xiaodong He, Lei Zhang, Jian Sun, Cornelia Carapcea, Chris Thrasher, Chris Buehler, and Chris Sienkiewicz. 2016. Rich image captioning in the wild. In *CVPR Workshops*.

Van-Khanh Tran and Le-Minh Nguyen. 2017. Natural language generation for spoken dialogue system using rnn encoder-decoder networks. In *CoNLL*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL*.

Subhashini Venugopalan, Lisa Anne Hendricks, Raymond Mooney, and Kate Saenko. 2016. Improving lstm-based video description with linguistic knowledge mined from text. In *EMNLP*.

Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015a. Sequence to sequence – video to text. In *ICCV*.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015b. Translating videos to natural language using deep recurrent neural networks. In *NAACL*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. In *EMNLP*.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *CVPR*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *ICCV*.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *CVPR*.

Youngjae Yu, Hyungjin Ko, Jongwook Choi, and Gunhee Kim. 2017. End-to-end concept word detection for video captioning, retrieval, and question answering. In *CVPR*.

Mihai Zanfir, Elisabeta Marinoiu, and Cristian Sminchisescu. 2016. Spatio-temporal attention models for grounded video captioning. In *ACCV*.

Multimodal Differential Network for Visual Question Generation

Badri N. Patro **Sandeep Kumar** **Vinod K. Kurmi** **Vinay P. Namboodiri**
Indian Institute of Technology, Kanpur
{badri,sandepkr,vinodkk,vinaypn}@iitk.ac.in

Abstract

Generating natural questions from an image is a semantic task that requires using visual and language modality to learn multimodal representations. Images can have multiple visual and language contexts that are relevant for generating questions namely places, captions, and tags. In this paper, we propose the use of exemplars for obtaining the relevant context. We obtain this by using a Multimodal Differential Network to produce natural and engaging questions. The generated questions show a remarkable similarity to the natural questions as validated by a human study. Further, we observe that the proposed approach substantially improves over state-of-the-art benchmarks on the quantitative metrics (BLEU, METEOR, ROUGE, and CIDEr).

1 Introduction

To understand the progress towards multimedia vision and language understanding, a visual Turing test was proposed by (Geman et al., 2015) that was aimed at visual question answering (Antol et al., 2015). Visual Dialog (Das et al., 2017) is a natural extension for VQA. Current dialog systems as evaluated in (Chattopadhyay et al., 2017) show that when trained between bots, AI-AI dialog systems show improvement, but that does not translate to actual improvement for Human-AI dialog. We believe that this is because, the questions generated by bots are not natural (human-like) and therefore does not translate to improved human dialog. Therefore an improvement in the quality of questions could enable dialog agents to perform well in human interactions. Further, (Ganju et al., 2017) show that unanswered questions can be used for improving VQA, Image captioning and Object Classification.

An interesting line of work in this respect is the work of (Mostafazadeh et al., 2016). Here the authors have proposed the challenging task of generating *natural* questions for an image. One aspect that is central to a question is the context that is relevant to generate it. However, this context changes for every image. As can be seen in Figure 1, an image with a person on a skateboard would result in questions related to the event. Whereas for a little girl, the questions could be related to age rather than the action. How can one have widely varying context provided for generating questions? To solve this problem, we use the context obtained by considering exemplars, specifically we use the difference between relevant and irrelevant exemplars. We consider different contexts in the form of Location, Caption, and Part of Speech tags.

Figure 1: Can you guess which among the given questions is human annotated and which is machine generated? [0]

Our method implicitly uses a differential context obtained through supporting and contrasting exemplars to obtain a differential embedding. This embedding is used by a question decoder to decode the appropriate question. As discussed further, we observe this implicit differential context to perform better than an explicit keyword

---

[0] The human annotated questions are (b) for the first image and (a) for the second image.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4002–4012
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

based context. The difference between the two approaches is illustrated in Figure 2. This also allows for better optimization as we can backpropagate through the whole network. We provide detailed empirical evidence to support our hypothesis. As seen in Figure 1 our method generates natural questions and improves over the state-of-the-art techniques for this problem.

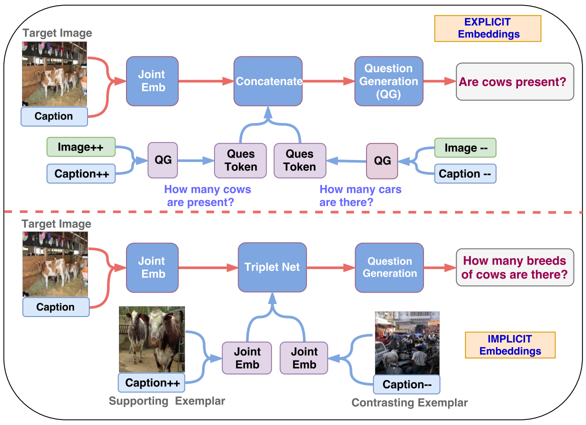

Figure 2: Here we provide intuition for using implicit embeddings instead of explicit ones. As explained in section 1, the question obtained by the implicit embeddings are natural and holistic than the explicit ones.

To summarize, we propose a multimodal differential network to solve the task of visual question generation. Our contributions are: (1) A method to incorporate exemplars to learn differential embeddings that captures the subtle differences between supporting and contrasting examples and aid in generating natural questions. (2) We provide Multimodal differential embeddings, as image or text alone does not capture the whole context and we show that these embeddings outperform the ablations which incorporate cues such as only image, or tags or place information. (3) We provide a thorough comparison of the proposed network against state-of-the-art benchmarks along with a user study and statistical significance test.

2 Related Work

Generating a natural and engaging question is an interesting and challenging task for a smart robot (like chat-bot). It is a step towards having a natural visual dialog instead of the widely prevalent visual question answering bots. Further, having the ability to ask natural questions based on different contexts is also useful for artificial agents that can interact with visually impaired people. While the task of generating question automatically is well studied in NLP community, it has been relatively less studied for image-related natural questions. This is still a difficult task (Mostafazadeh et al., 2016) that has gained recent interest in the community.

Recently there have been many deep learning based approaches as well for solving the text-based question generation task such as (Du et al., 2017). Further, (Serban et al., 2016) have proposed a method to generate a factoid based question based on triplet set {subject, relation and object} to capture the structural representation of text and the corresponding generated question.

These methods, however, were limited to text-based question generation. There has been extensive work done in the Vision and Language domain for solving image captioning, paragraph generation, Visual Question Answering (VQA) and Visual Dialog. (Barnard et al., 2003; Farhadi et al., 2010; Kulkarni et al., 2011) proposed conventional machine learning methods for image description. (Socher et al., 2014; Vinyals et al., 2015; Karpathy and Fei-Fei, 2015; Xu et al., 2015; Fang et al., 2015; Chen and Lawrence Zitnick, 2015; Johnson et al., 2016; Yan et al., 2016) have generated descriptive sentences from images with the help of Deep Networks. There have been many works for solving Visual Dialog (Chappell et al., 2004; Das et al., 2016, 2017; De Vries et al., 2017; Strub et al., 2017). A variety of methods have been proposed by (Malinowski and Fritz, 2014; Lin et al., 2014; Antol et al., 2015; Ren et al., 2015; Ma et al., 2016; Noh et al., 2016) for solving VQA task including attention-based methods (Zhu et al., 2016; Fukui et al., 2016; Gao et al., 2015; Xu and Saenko, 2016; Lu et al., 2016; Shih et al., 2016; Patro and Namboodiri, 2018). However, Visual Question Generation (VQG) is a separate task which is of interest in its own right and has not been so well explored (Mostafazadeh et al., 2016). This is a vision based novel task aimed at generating natural and engaging question for an image. (Yang et al., 2015) proposed a method for continuously generating questions from an image and subsequently answering those questions. The works closely related to ours are that of (Mostafazadeh et al., 2016) and (Jain et al., 2017). In the former work, the authors used an encoder-decoder based framework whereas in the latter work, the authors extend it by using a variational autoencoder based sequential routine to obtain natural questions by performing sampling of the latent variable.

3 Approach

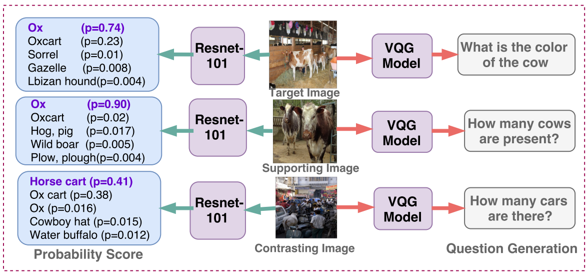

Figure 3: An illustrative example shows the validity of our obtained exemplars with the help of an object classification network, RESNET-101. We see that the probability scores of target and supporting exemplar image are similar. That is not the case with the contrasting exemplar. The corresponding generated questions when considering the individual images are also shown.

In this section, we clarify the basis for our approach of using exemplars for question generation. To use exemplars for our method, we need to ensure that our exemplars can provide context and that our method generates valid exemplars.

We first analyze whether the exemplars are valid or not. We illustrate this in figure 3. We used a pre-trained RESNET-101 (He et al., 2016) object classification network on the target, supporting and contrasting images. We observed that the supporting image and target image have quite similar probability scores. The contrasting exemplar image, on the other hand, has completely different probability scores.

Exemplars aim to provide appropriate context. To better understand the context, we experimented by analysing the questions generated through an exemplar. We observed that indeed a supporting exemplar could identify relevant tags (cows in Figure 3) for generating questions. We improve use of exemplars by using a triplet network. This network ensures that the joint image-caption embedding for the supporting exemplar are closer to that of the target image-caption and vice-versa. We empirically evaluated whether an explicit approach that uses the differential set of tags as a one-hot encoding improves the question generation, or the implicit embedding obtained based on the triplet network. We observed that the implicit multimodal differential network empirically provided better context for generating questions. Our understanding of this phenomenon is that both target and supporting exemplars generate similar questions whereas contrasting exemplars generate very different questions from the target question. The triplet network that enhances the joint embedding thus aids to improve the generation of target question. These are observed to be better than the explicitly obtained context tags as can be seen in Figure 2. We now explain our method in detail.

4 Method

The task in visual question generation (VQG) is to generate a natural language question $\hat{Q}$, for an image I. We consider a set of pre-generated context C from image I. We maximize the conditional probability of generated question given image and context as follows:

$$\hat{\theta} = \arg\max_{\theta} \sum_{(I,C,Q)} \log P(Q|\, I, C, \theta) \qquad (1)$$

where θ is a vector for all possible parameters of our model. Q is the ground truth question. The log probability for the question is calculated by using joint probability over $\{q_0, q_1, ..., q_N\}$ with the help of chain rule. For a particular question, the above term is obtained as:

$$\log P(\hat{Q}|I, C) = \sum_{t=0}^{N} \log P(q_t|I, C, q_0, .., q_{t-1})$$

where N is length of the sequence, and q_t is the t^{th} word of the question. We have removed θ for simplicity.

Our method is based on a sequence to sequence network (Sutskever et al., 2014; Vinyals et al., 2015; Bahdanau et al., 2014). The sequence to sequence network has a text sequence as input and output. In our method, we take an image as input and generate a natural question as output. The architecture for our model is shown in Figure 4. Our model contains three main modules, (a) Representation Module that extracts multimodal features (b) Mixture Module that fuses the multimodal representation and (c) Decoder that generates question using an LSTM-based language model.

During inference, we sample a question word q_i from the softmax distribution and continue sampling until the end token or maximum length for the question is reached. We experimented with both sampling and *argmax* and found out that *argmax* works better. This result is provided in the supplementary material.

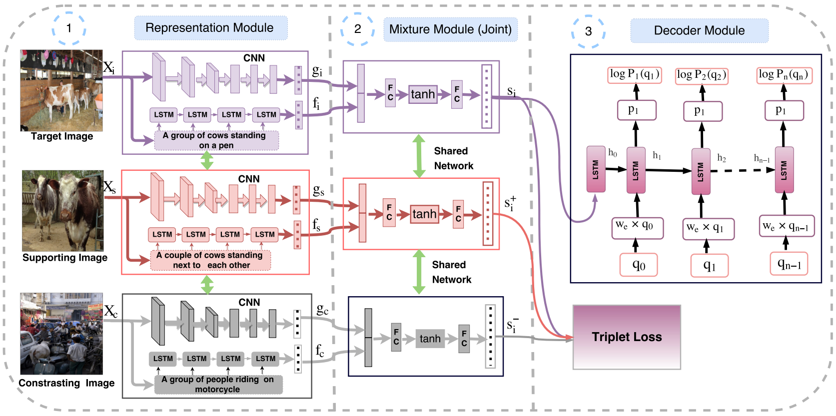

Figure 4: This is an overview of our Multimodal Differential Network for Visual Question Generation. It consists of a Representation Module which extracts multimodal features, a Mixture Module that fuses the multimodal representation and a Decoder that generates question using an LSTM based language model. In this figure, we have shown the Joint Mixture Module. We train our network with a Cross-Entropy and Triplet Loss.

4.1 Multimodal Differential Network

The proposed Multimodal Differential Network (MDN) consists of a representation module and a joint mixture module.

4.1.1 Finding Exemplars

We used an efficient KNN-based approach (k-d tree) with Euclidean metric to obtain the exemplars. This is obtained through a coarse quantization of nearest neighbors of the training examples into 50 clusters, and selecting the nearest as supporting and farthest as the contrasting exemplars. We experimented with ITML based metric learning (Davis et al., 2007) for image features. Surprisingly, the KNN-based approach outperforms the latter one. We also tried random exemplars and different number of exemplars and found that $k = 5$ works best. We provide these results in the supplementary material.

4.1.2 Representation Module

We use a triplet network (Frome et al., 2007; Hoffer and Ailon, 2015) in our representation module. We refereed a similar kind of work done in (Patro and Namboodiri, 2018) for building our triplet network. The triplet network consists of three subparts: target, supporting, and contrasting networks. All three networks share the same parameters. Given an image x_i we obtain an embedding g_i using a CNN parameterized by a function $G(x_i, W_c)$ where W_c are the weights for the CNN. The caption C_i results in a caption embedding f_i through an LSTM parameterized by a function $F(C_i, W_l)$ where W_l are the weights for the LSTM. This is shown in part 1 of Figure 4. Similarly we obtain image embeddings g_s & g_c and caption embeddings f_s & f_c.

$$\begin{aligned} g_i &= G(x_i, W_c) = CNN(x_i) \\ f_i &= F(C_i, W_l) = LSTM(C_i) \end{aligned} \tag{2}$$

4.1.3 Mixture Module

The Mixture module brings the image and caption embeddings to a joint feature embedding space. The input to the module is the embeddings obtained from the representation module. We have evaluated four different approaches for fusion viz., joint, element-wise addition, Hadamard and attention method. Each of these variants receives image features g_i & the caption embedding f_i, and outputs a fixed dimensional feature vector s_i. The Joint method concatenates g_i & f_i and maps them to a fixed length feature vector s_i as follows:

$$s_i = W_j^T * \tanh(W_{ij} g_i \frown (W_{cj} f_i + b_j)) \tag{3}$$

where g_i is the 4096-dimensional convolutional feature from the FC7 layer of pretrained VGG-19 Net (Simonyan and Zisserman, 2014). W_{ij}, W_{cj}, W_j are the weights and b_j is the bias for different layers. $\frown$ is the concatenation operator.

Similarly, we obtain context vectors s_i^+ & s_i^- for the supporting and contrasting exemplars. Details for other fusion methods are present in supplementary.The aim of the triplet network (Schroff et al., 2015) is to obtain context vectors that bring the supporting exemplar embeddings closer to the target embedding and vice-versa. This is obtained as follows:

$$\begin{aligned} &D(t(s_i), t(s_i^+)) + \alpha < D(t(s_i), t(s_i^-)) \\ &\forall (t(s_i), t(s_i^+), t(s_i^-)) \in M, \end{aligned} \tag{4}$$

where $D(t(s_i), t(s_j)) = ||t(s_i) - t(s_j)||_2^2$ is the Euclidean distance between two embeddings $t(s_i)$ and $t(s_j)$. M is the training dataset that contains all set of possible triplets. $T(s_i, s_i^+, s_i^-)$ is the triplet loss function. This is decomposed into two terms, one that brings the supporting sample closer and one that pushes the contrasting sample further. This is given by

$$T(s_i, s_i^+, s_i^-) = \texttt{max}(0, D^+ + \alpha - D^-) \quad (5)$$

Here D^+, D^- represent the Euclidean distance between the target and supporting sample, and target and opposing sample respectively. The parameter $\alpha(= 0.2)$ controls the separation margin between these and is obtained through validation data.

4.2 Decoder: Question Generator

The role of decoder is to predict the probability for a question, given s_i. RNN provides a nice way to perform conditioning on previous state value using a fixed length hidden vector. The conditional probability of a question token at particular time step q_t is modeled using an LSTM as used in machine translation (Sutskever et al., 2014). At time step t, the conditional probability is denoted by $P(q_t|I, C, q_0, ...q_{t-1}) = P(q_t|I, C, h_t)$, where h_t is the hidden state of the LSTM cell at time step t, which is conditioned on all the previously generated words $\{q_0, q_1, ...q_{N-1}\}$. The word with maximum probability in the probability distribution of the LSTM cell at step k is provided as an input to the LSTM cell at step $k+1$ as shown in part 3 of Figure 4. At $t = -1$, we are feeding the output of the mixture module to LSTM. $\hat{Q} = \{\hat{q_0}, \hat{q_1}, ...\hat{q_{N-1}}\}$ are the predicted question tokens for the input image I. Here, we are using $\hat{q_0}$ and $\hat{q_{N-1}}$ as the special token START and STOP respectively. The softmax probability for the predicted question token at different time steps is given by the following equations where LSTM refers to the standard LSTM cell equations:

$$
\begin{aligned}
&x_{-1} = S_i = \text{Mixture Module}(g_i, f_i)\\
&h_0 = \text{LSTM}(x_{-1})\\
&x_t = W_e * q_t, \forall t \in \{0, 1, 2, ...N-1\}\\
&h_{t+1} = \text{LSTM}(x_t, h_t), \forall t \in \{0, 1, 2, ...N-1\}\\
&o_{t+1} = W_o * h_{t+1}\\
&\hat{y}_{t+1} = P(q_{t+1}|I, C, h_t) = \text{Softmax}(o_{t+1})\\
&Loss_{t+1} = loss(\hat{y}_{t+1}, y_{t+1})
\end{aligned}
$$

Where $\hat{y}_{t+1}$ is the probability distribution over all question tokens. $loss$ is cross entropy loss.

4.3 Cost function

Our objective is to minimize the total loss, that is the sum of cross entropy loss and triplet loss over all training examples. The total loss is:

$$L = \frac{1}{M}\sum_{i=1}^{M}(L_{cross} + \gamma L_{triplet}) \quad (6)$$

where M is the total number of samples,γ is a constant, which controls both the loss. $L_{triplet}$ is the triplet loss function 5. L_{cross} is the cross entropy loss between the predicted and ground truth questions and is given by:

$$L_{cross} = \frac{-1}{N}\sum_{t=1}^{N} y_t \texttt{log} P(\hat{q_t}|I_i, C_i, \hat{q_0}, ..\hat{q_{t-1}})$$

where, N is the total number of question tokens, y_t is the ground truth label. The code for MDN-VQG model is provided [1].

4.4 Variations of Proposed Method

While, we advocate the use of multimodal differential network for generating embeddings that can be used by the decoder for generating questions, we also evaluate several variants of this architecture. These are as follows:

Tag Net: In this variant, we consider extracting the part-of-speech (POS) tags for the words present in the caption and obtaining a Tag embedding by considering different methods of combining the one-hot vectors. Further details and experimental results are present in the supplementary. This Tag embedding is then combined with the image embedding and provided to the decoder network.

Place Net: In this variant we explore obtaining embeddings based on the visual scene understanding. This is obtained using a pre-trained PlaceCNN (Zhou et al., 2017) that is trained to classify 365 different types of scene categories. We then combine the activation map for the input image and the VGG-19 based place embedding to obtain the joint embedding used by the decoder.

Differential Image Network: Instead of using multimodal differential network for generating embeddings, we also evaluate differential image network for the same. In this case, the embedding does not include the caption but is based only on

---

[1] The project page for MDN-VQG Model is `https://badripatro.github.io/MDN-VQG/`

Figure 5: These are some examples from the VQG-COCO dataset which provide a comparison between our generated questions and human annotated questions. (a) is the human annotated question for all the images. More qualitative results are present in the supplementary material.

the image feature. We also exeperimented with using multiple exemplars and random exemplars. Further details, pseudocode and results regarding these are present in the supplementary material.

4.5 Dataset

We conduct our experiments on Visual Question Generation (VQG) dataset (Mostafazadeh et al., 2016), which contains human annotated questions based on images of MS-COCO dataset. This dataset was developed for generating natural and engaging questions based on common sense reasoning. We use VQG-COCO dataset for our experiments which contains a total of 2500 training images, 1250 validation images, and 1250 testing images. Each image in the dataset contains five natural questions and five ground truth captions. It is worth noting that the work of (Jain et al., 2017) also used the questions from VQA dataset (Antol et al., 2015) for training purpose, whereas the work by (Mostafazadeh et al., 2016) uses only the VQG-COCO dataset. VQA-1.0 dataset is also built on images from MS-COCO dataset. It contains a total of 82783 images for training, 40504 for validation and 81434 for testing. Each image is associated with 3 questions. We used pretrained caption generation model (Karpathy and Fei-Fei, 2015) to extract captions for VQA dataset as the human annotated captions are not there in the dataset. We also get good results on the VQA dataset (as shown in Table 2) which shows that our method doesn't necessitate the presence of ground truth captions. We train our model separately for VQG-COCO and VQA dataset.

4.6 Inference

We made use of the 1250 validation images to tune the hyperparameters and are providing the results on test set of VQG-COCO dataset. During inference, We use the Representation module to find the embeddings for the image and ground truth caption without using the supporting and contrasting exemplars. The mixture module provides the joint representation of the target image and ground truth caption. Finally, the decoder takes in the joint features and generates the question. We also experimented with the captions generated by an Image-Captioning network (Karpathy and Fei-Fei, 2015) for VQG-COCO dataset and the result for that and training details are present in the supplementary material.

5 Experiments

We evaluate our proposed MDN method in the following ways: First, we evaluate it against other variants described in section 4.4 and 4.1.3. Second, we further compare our network with state-of-the-art methods for VQA 1.0 and VQG-COCO dataset. We perform a user study to gauge human opinion on naturalness of the generated question and analyze the word statistics in Figure 6. This is an important test as humans are the best

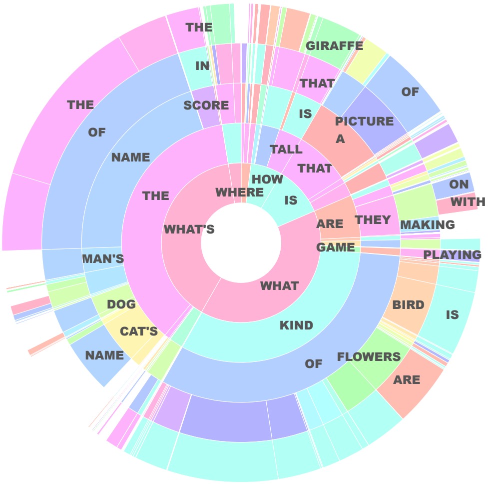

Figure 6: Sunburst plot for VQG-COCO: The i^{th} ring captures the frequency distribution over words for the i^{th} word of the generated question. The angle subtended at the center is proportional to the frequency of the word. While some words have high frequency, the outer rings illustrate a fine blend of words. We have restricted the plot to 5 rings for easy readability. Best viewed in color.

deciders of naturalness. We further consider the statistical significance for the various ablations as well as the state-of-the-art models. The quantitative evaluation is conducted using standard metrics like BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE (Lin, 2004), CIDEr (Vedantam et al., 2015). Although these metrics have not been shown to correlate with 'naturalness' of the question these still provide a reasonable quantitative measure for comparison. Here we only provide the BLEU1 scores, but the remaining BLEU-n metric scores are present in the supplementary. We observe that the proposed MDN provides improved embeddings to the decoder. We believe that these embeddings capture instance specific differential information that helps in guiding the question generation. Details regarding the metrics are given in the supplementary material.

5.1 Ablation Analysis

We considered different variations of our method mentioned in section 4.4 and the various ways to obtain the joint multimodal embedding as described in section 4.1.3. The results for the VQG-COCO test set are given in table 1. In this table, every block provides the results for one of the variations of obtaining the embeddings and different ways of combining them. We observe that the Joint Method (JM) of combining the embeddings works the best in all cases except the Tag Embeddings. Among the ablations, the proposed MDN method works way better than the other variants in terms of BLEU, METEOR and ROUGE metrics by achieving an improvement of 6%, 12% and 18% in the scores respectively over the best other variant.

Emb.	**Method**	**B1**	**METEOR**	**ROUGE**	**CIDEr**
Tag	AtM	22.4	8.6	22.5	20.8
Tag	HM	**24.4**	**10.8**	**24.3**	**55.0**
Tag	AM	24.4	10.6	23.9	49.4
Tag	JM	22.2	10.5	22.8	50.1
PlaceCNN	AtM	24.4	10.3	24.0	51.8
PlaceCNN	HM	24.0	10.4	24.3	49.8
PlaceCNN	AM	24.1	10.6	24.3	51.5
PlaceCNN	JM	**25.7**	**10.8**	**24.5**	**56.1**
Diff. Img	AtM	20.5	8.5	**24.4**	19.2
Diff. Img	HM	23.6	8.6	22.3	22.0
Diff. Img	AM	20.6	8.5	24.4	19.2
Diff. Img	JM	**30.4**	**11.7**	22.3	**22.8**
MDN	AtM	22.4	8.8	24.6	22.4
MDN	HM	26.6	12.8	30.1	31.4
MDN	AM	29.6	15.4	32.8	41.6
MDN(**Ours**)	JM	**36.0**	**23.4**	**41.8**	**50.7**

Table 1: Analysis of variants of our proposed method on VQG-COCO Dataset as mentioned in section 4.4 and different ways of getting a joint embedding (Attention (AtM), Hadamard (HM), Addition (AM) and Joint (JM) method as given in section 4.1.3) for each method. Refer section 5.1 for more details. B1 is BLEU1.

5.2 Baseline and State-of-the-Art

The comparison of our method with various baselines and state-of-the-art methods is provided in table 2 for VQA 1.0 and table 3 for VQG-COCO dataset. The comparable baselines for our method are the image based and caption based models in which we use either only the image or the caption embedding and generate the question. In both the tables, the first block consists of the current state-of-the-art methods on that dataset and the second contains the baselines. We observe that for the VQA dataset we achieve an improvement of 8% in BLEU and 7% in METEOR metric scores over the baselines, whereas for VQG-COCO dataset this is 15% for both the metrics. We improve over the previous state-of-the-art (Yang et al., 2015) for VQA dataset by around 6% in BLEU score and 10% in METEOR score. In the VQG-COCO dataset, we improve over (Mostafazadeh et al., 2016) by 3.7% and (Jain et al., 2017) by 3.5% in terms of METEOR scores.

5.3 Statistical Significance Analysis

We have analysed Statistical Significance (Demšar, 2006) of our MDN model for VQG for different variations of the mixture

Methods	BLEU1	METEOR	ROUGE	CIDEr
Sample(Yang,2015)	38.8	12.7	34.2	13.3
Max(Yang,2015)	59.4	17.8	49.3	33.1
Image Only	56.6	15.1	40.0	31.0
Caption Only	57.1	15.5	36.6	30.5
MDN-Attention	60.7	16.7	49.8	33.6
MDN-Hadamard	61.7	16.7	50.1	29.3
MDN-Addition	61.7	18.3	50.4	**42.6**
MDN-Joint (**Ours**)	**65.1**	**22.7**	**52.0**	33.1

Table 2: State-of-the-Art comparison on VQA-1.0 Dataset. The first block consists of the state-of-the-art results, second block refers to the baselines mentioned in section 5.2, third block provides the results for the variants of mixture module present in section 4.1.3.

Context	BLEU1	METEOR	ROUGE	CIDEr
Natural 2016	19.2	19.7	-	-
Creative 2017	35.6	19.9	-	-
Image Only	20.8	8.6	22.6	18.8
Caption Only	21.1	8.5	25.9	22.3
Tag-Hadamard	24.4	10.8	24.3	55.0
Place CNN-Joint	25.7	10.8	24.5	**56.1**
Diff.Image-Joint	30.4	11.7	26.3	38.8
MDN-Joint (**Ours**)	**36.0**	**23.4**	**41.8**	50.7
Humans 2016	**86.0**	**60.8**	**-**	-

Table 3: State-of-the-Art (SOTA) comparison on VQG-COCO Dataset. The first block consists of the SOTA results, second block refers to the baselines mentioned in section 5.2, third block shows the results for the best method for different ablations mentioned in table 1.

module mentioned in section 4.1.3 and also against the state-of-the-art methods. The Critical Difference (CD) for Nemenyi (Fišer et al., 2016) test depends upon the given α (confidence level, which is 0.05 in our case) for average ranks and N (number of tested datasets). If the difference in the rank of the two methods lies within CD, then they are not significantly different and vice-versa. Figure 7 visualizes the post hoc analysis using the CD diagram. From the figure, it is clear that MDN-Joint works best and is statistically significantly different from the state-of-the-art methods.

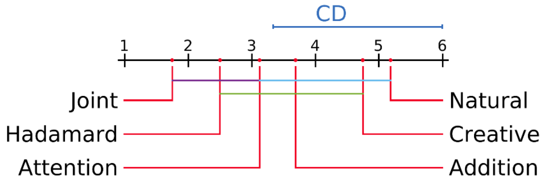

Figure 7: The mean rank of all the models on the basis of METEOR score are plotted on the x-axis. Here Joint refers to our MDN-Joint model and others are the different variations described in section 4.1.3 and Natural (Mostafazadeh et al., 2016), Creative (Jain et al., 2017). The colored lines between the two models represents that these models are not significantly different from each other.

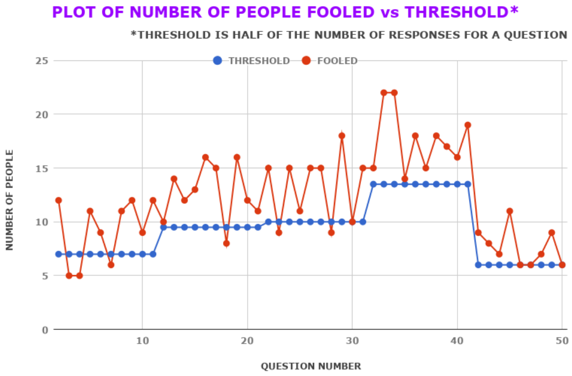

Figure 8: Perceptual Realism Plot for human survey. Here every question has different number of responses and hence the threshold which is the half of total responses for each question is varying. This plot is only for 50 of the 100 questions involved in the survey. See section 5.4 for more details.

5.4 Perceptual Realism

A human is the best judge of naturalness of any question, We evaluated our proposed MDN method using a 'Naturalness' Turing test (Zhang et al., 2016) on 175 people. People were shown an image with 2 questions just as in figure 1 and were asked to rate the naturalness of both the questions on a scale of 1 to 5 where 1 means 'Least Natural' and 5 is the 'Most Natural'. We provided 175 people with 100 such images from the VQG-COCO validation dataset which has 1250 images. Figure 8 indicates the number of people who were fooled (rated the generated question more or equal to the ground truth question). For the 100 images, on an average 59.7% people were fooled in this experiment and this shows that our model is able to generate natural questions.

6 Conclusion

In this paper we have proposed a novel method for generating natural questions for an image. The approach relies on obtaining multimodal differential embeddings from image and its caption. We also provide ablation analysis and a detailed comparison with state-of-the-art methods, perform a user study to evaluate the naturalness of our generated questions and also ensure that the results are statistically significant. In future, we would like to analyse means of obtaining composite embeddings. We also aim to consider the generalisation of this approach to other vision and language tasks.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72.

K Barnard, P Duygulu, and D Forsyth. 2003. N. de freitas, d. *Blei, and MI Jordan," Matching Words and Pictures", submitted to JMLR*.

Alan R Chappell, Andrew J Cowell, David A Thurman, and Judi R Thomson. 2004. Supporting mutual understanding in a visual dialogue between analyst and computer. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 48, pages 376–380. SAGE Publications Sage CA: Los Angeles, CA.

Prithvijit Chattopadhyay, Deshraj Yadav, Viraj Prabhu, Arjun Chandrasekaran, Abhishek Das, Stefan Lee, Dhruv Batra, and Devi Parikh. 2017. Evaluating visual conversational agents via cooperative human-ai games. In *Proceedings of the Fifth AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*.

Xinlei Chen and C Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2422–2431.

Abhishek Das, Harsh Agrawal, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. 2016. Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions? In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017. Visual Dialog. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. 2007. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM.

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proc. of CVPR*.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, et al. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *European conference on computer vision*, pages 15–29. Springer.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2016. Janes v0. 4: Korpus slovenskih spletnih uporabniških vsebin. *Slovenščina*, 2(4):2.

Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. 2007. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Siddha Ganju, Olga Russakovsky, and Abhinav Gupta. 2017. What's in a question: Using visual questions as a form of supervision. In *CVPR*.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. 2015. Are you talking to a machine? dataset and methods for multilingual image question. In *Advances in Neural Information Processing Systems*, pages 2296–2304.

Donald Geman, Stuart Geman, Neil Hallonquist, and Laurent Younes. 2015. Visual turing test for computer vision systems. *Proceedings of the National Academy of Sciences of the United States of America*, 112(12):3618–3623.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer.

Unnat Jain, Ziyu Zhang, and Alexander G Schwing. 2017. Creativity: Generating diverse questions using variational autoencoders. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4565–4574.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2011. Baby talk: Understanding and generating image descriptions. In *Proceedings of the 24th CVPR*. Citeseer.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out:Proceedings of the ACL-04 workshop*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Lin Ma, Zhengdong Lu, and Hang Li. 2016. Learning to answer questions from image using convolutional neural network. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1802–1813.

Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. 2016. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 30–38.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Badri Patro and Vinay P Namboodiri. 2018. Differential attention for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7680–7688.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems*, pages 2953–2961.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*.

Kevin J Shih, Saurabh Singh, and Derek Hoiem. 2016. Where to look: Focus regions for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4613–4621.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association of Computational Linguistics*, 2(1):207–218.

Florian Strub, Harm De Vries, Jeremie Mary, Bilal Piot, Aaron Courville, and Olivier Pietquin. 2017. End-to-end optimization of goal-driven and visually grounded dialogue systems. *arXiv preprint arXiv:1703.05423*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pages 451–466. Springer.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer.

Yezhou Yang, Yi Li, Cornelia Fermuller, and Yiannis Aloimonos. 2015. Neural self talk: Image understanding via continuous questioning and answering. *arXiv preprint arXiv:1512.03460*.

Richard Zhang, Phillip Isola, and Alexei A Efros. 2016. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4995–5004.

Entity-aware Image Caption Generation

Di Lu[1], **Spencer Whitehead**[1], **Lifu Huang**[1],**Heng Ji**[1], **Shih-Fu Chang**[2]
[1]Computer Science Department, Rensselaer Polytechnic Institute
`{lud2,whites5,huangl7,jih}@rpi.edu`
[2]Computer Science Department, Columbia University
`sfchang@cs.columbia.edu`

Abstract

Current image captioning approaches generate descriptions which lack specific information, such as named entities that are involved in the images. In this paper we propose a new task which aims to generate informative image captions, given images and hashtags as input. We propose a simple but effective approach to tackle this problem. We first train a convolutional neural networks - long short term memory networks (CNN-LSTM) model to generate a template caption based on the input image. Then we use a knowledge graph based collective inference algorithm to fill in the template with specific named entities retrieved via the hashtags. Experiments on a new benchmark dataset collected from Flickr show that our model generates news-style image descriptions with much richer information. Our model outperforms unimodal baselines significantly with various evaluation metrics. [1]

1 Introduction

As information regarding emergent situations disseminates through social media, the information is presented in a variety of data modalities (*e.g.* text, images, and videos), with each modality providing a slightly different perspective. Images have the capability to vividly represent events and entities, but without proper contextual information they become less meaningful and lose utility. While images may be accompanied by associated tags or other meta-data, which are inadequate to convey detailed events, many lack the descriptive text to provide such context. For example, there are 17,285 images on Flickr from the Women's March on January 21, 2017,[2] most of which contain only a few tags and lack any detailed text descriptions. The absence of context leaves individuals with no knowledge of details such as the purpose or location of the march.

Image captioning offers a viable method to directly provide images with the necessary contextual information through textual descriptions. Advances in image captioning (Xu et al., 2015; Fang et al., 2015; Karpathy and Fei-Fei, 2015; Vinyals et al., 2015) are effective in generating sentence-level descriptions. However, sentences generated by these approaches are usually generic descriptions of the visual content and ignore background information. Such generic descriptions do not suffice in emergent situations as they, essentially, mirror the information present in the images and do not provide detailed descriptions regarding events and entities present in, or related to, the images, which is imperative to understanding emergent situations. For example, given the image in Figure 1, the state-of-the-art automatically generated caption is '*Several women hold signs in front of a building.*', which is lacking information regarding relevant entities (*e.g.* ' *Junior doctors*', '*Tories*').

In our work, we propose an ambitious task: **entity-aware image caption generation**: automatically generate an image description that incorporates specific information such as named entities, relevant to the image, given limited text information, such as associated tags and meta-data (*e.g.* time of photo and geo-location). Our approach to this task generally follows three steps. First, instead of directly generating a sentence for an image, we generate a ***template*** sentence with fillable slots by training an image captioning architecture on image-caption pairs, where we replace the entities from the captions with slot types indicating the type of entity that should be used

---

[1] Datasets and programs: `https://github.com/dylandilu/Entity-aware-Image-Captioning`

[2] https://en.wikipedia.org/wiki/2017_Women%27s_March

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4013–4023
Brussels, Belgium, October 31 - November 4, 2018.

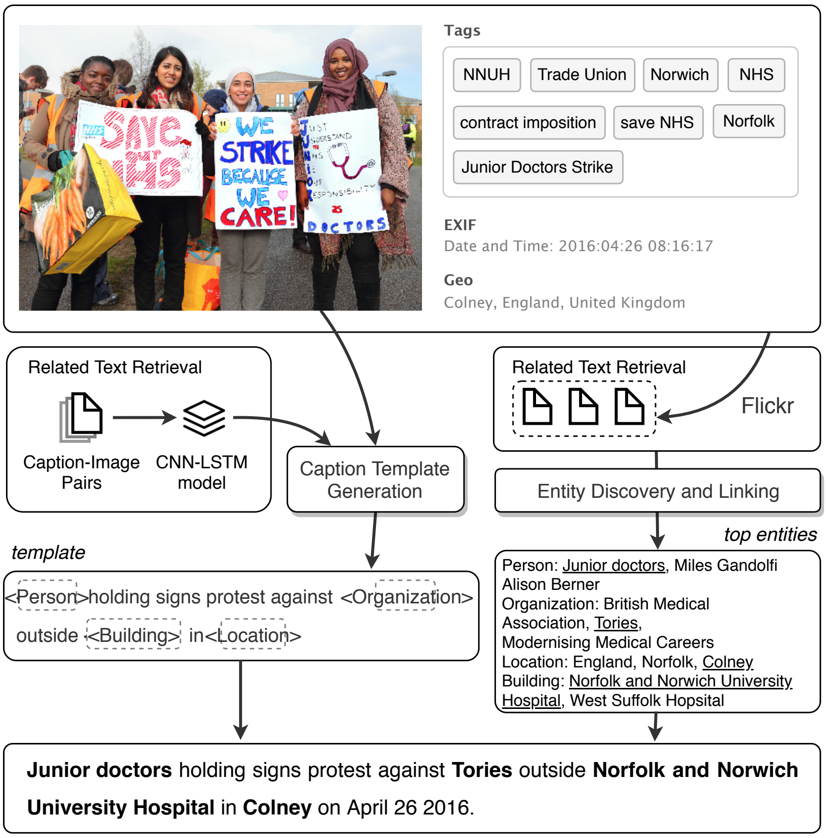

Figure 1: The overall framework.

to fill the slot. A ***template*** for the example in Figure 1 is: `<Person>` holding signs protest against `<Organization>` outside `<Building>` in `<Location>`.

Second, given the associated tags of an image, we apply entity discovery and linking (EDL) methods to extract specific entities from previous posts that embed the same tags. Finally, we select appropriate candidates for each slot based upon the entity type and frequency. For example, we select the person name '*Junior doctors*' to fill in the slot `<Person>` because it co-occurs frequently with other entities such as '*Tories*' in the text related to the tags $\#NHS, \#JuniorDoctorsStrike$. This framework offers a distinct advantage in that it is very flexible, so more advanced captioning or EDL methods as well as other data can be used almost interchangeably within the framework.

To the best of our knowledge, we are the first to incorporate contextual information into image captioning without large-scale training data or topically related news articles and the generated image captions are closer to news captions.

2 Approach Overview

Figure 1 shows the overall framework of our proposed model. Given an image with associated tags and other meta-data, such as geographical tags and EXIF data,[3] we first feed the image into a template caption generator to generate a sentence composed of context words, such as "*stand*", and slots, such as `<person>`, to represent missing specific information like named entities (Section 3). The template caption generator, which follows the encoder-decoder model (Cho et al., 2014) with a CNN encoder and LSTM decoder (Vinyals et al., 2015), is trained using news image-template caption pairs.

We then retrieve topically-related images from the Flickr database, which have the same tags as the input image. Next, we apply EDL algorithms to the image titles to extract entities and link them to external knowledge bases to retrieve their fine-grained entity types. Finally, for each slot generated by the template generator, we choose to fill the slot with the appropriate candidate based on entity type and frequency (Section 4).

---

[3] EXIF data contains meta-data tags of photos such as date, time, and camera settings.

3 Template Caption Generation

Language models (LM) are widely used to generate text (Wen et al., 2015; Tran and Nguyen, 2017) and play a crucial role in most of the existing image captioning approaches (Vinyals et al., 2015; Xu et al., 2015). These models, learned from large-scale corpora, are able to predict a probability distribution over a vocabulary. However, LM struggle to generate specific entities, which occur sparsely, if at all, within training corpora. Moreover, the desired entity-aware captions may contain information not directly present in the image alone. Unless the LM is trained or conditioned on data specific to the emergent situation of interest, the LM alone cannot generate a caption that incorporates the specific background information. We address this issue by only relying on the LM to generate abstract slot tokens and connecting words or phrases, while slot filling is used to incorporate specific information. This approach allows the LM to focus on generating words or phrases with higher probability, since each slot token effectively has a probability equal to the sum of all the specific entities that it represents, thereby circumventing the issue of generating lower probability or out-of-vocabulary (OOV) words.

In this section, we describe a novel method to train a model to automatically generate template captions with slots as '*placeholders*' for specific background information. We first present the schemas which define the slot types (Section 3.1) and the procedure to acquire training data for template generation (Section 3.2). Finally, we introduce the model for template caption generation (Section 3.3).

3.1 Template Caption Definition

Named entities are the most specific information which cannot be easily learned by LM. Thus, in this work, we define slots as placeholders for entities with the same types. We use the fine grained entity types defined in DBpedia [4] (Auer et al., 2007) to name the slots because these types are specific enough to differentiate between a wide range of entities and still general enough so that the slots have higher probabilities in the language model. For example, `Person` is further divided into `Athlete`, `Singer`, `Politician`, and so on. Therefore, a template caption like '`Athlete` celebrates after scoring.' can be generated by the language model through leveraging image features, where the slot `Athlete` means a sports player (e.g., Cristiano Ronaldo, Lionel Messi).

3.2 Acquisition of Image-Template Caption Pairs

High quality training data is crucial to train a template caption generator. However, the image-caption datasets used in previous work, such as Microsoft Common Objects in Context (MS COCO) (Lin et al., 2014) and Flickr30K (Rashtchian et al., 2010), are not suitable for this task because they are designed for non-specific caption generation and do not contain detailed, specific information such as named entities. Further, manual creation of captions is expensive. In this work, we utilize news image-caption pairs, which are well written and can be easily collected. We use the example in Figure 2 to describe our procedure to convert image-caption to image-template caption pairs: **preprocessing**, **compression**, **generalization**.

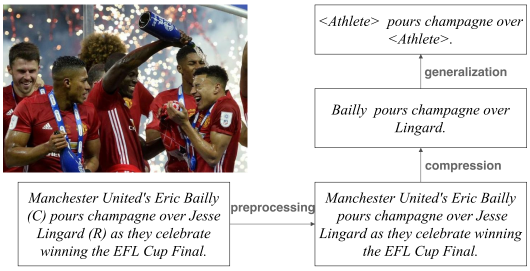

Figure 2: Procedure of Converting News Captions into Templates.

Preprocessing: We first apply the following pre-processing steps: (1) remove words in parentheses, such as *'(C)'* and *'(R)'* in Figure 2, because they usually represent auxiliary information and are not aligned with visual concepts in images; (2) if a caption includes more than one sentence, we choose the longer one. Based on our observation, shorter sentences usually play the role of background introduction, which are not aligned with the key content in images; (3) remove captions with less than 10 tokens because they tend to be not informative enough. The average length of the news image captions is 37 tokens.

Compression: The goal of compression is to make news captions short and aligned with images as much as possible by keeping informa-

---

[4] We use the sixth level entity types in Yago ontology (Wordnet types only).

tion related to objects and concepts in the images, which are usually subjects, verbs and objects in sentences. In this paper, we propose a simple but efficient compression method based on dependency parsing. We do not use other complicated compressions (Kuznetsova et al., 2014) because our simple method achieves comparative results on image caption dataset. We first apply the Stanford dependency parser (De Marneffe and Manning, 2008) on preprocessed captions. Then, we traverse the parse tree from the root (*e.g.'pours'*) via `<governor, grammatical relations, dependent>` triples using breadth-first search. We decide to keep a dependent or not based on its grammatical relation with the governor. Based on our observations, among the 50 grammatical relations in the Stanford dependency parser, we keep the dependents that have the following grammatical relations with their governors: *nsubj, obj, iobj, dobj, acomp, det, neg, nsubjpass, pobj, predet, prep, prt, vmod, nmod, cc*.

Generalization: The last step for preparing training data is to extract entities from captions and replace them with the slot types we defined in Section 3.1. We apply Stanford CoreNLP name tagger (Manning et al., 2014) to the captions to extract entity mentions of the following types: `Person`, `Location`, `Organization`, and `Miscellaneous`. Next, we use an English Entity Linking algorithm (Pan et al., 2015) to link the entity mentions to DBpedia and retrieve their fine-grained types.[5] We choose the higher level type if there are multiple fine-grained types for a name. For example, the entity types of *Manchester United*, *Eric Bailly*, and *Jesse Lingard* are *SoccerTeam*, *Athlete*, and *Athlete*, respectively. For entity mentions that cannot be linked to DBpedia, we use their coarse-grained entity types, which are the outputs of name tagger.

Finally, we replace the entities in the compressed captions with their corresponding slots:

Generalized Template: `<Athlete>` pours champagne over `<Athlete>`.

3.3 Generation Model

Using the template caption and image pairs *(S, I)* as training data, we regard the template caption generation as a regular image captioning task. Thus, we adapt the encoder-decoder architecture which is successful in the image captioning task (Vinyals et al., 2015; Xu et al., 2015). Our model (Figure 3) is most similar to the one proposed in (Vinyals et al., 2015). Note, other captioning methods may easily be used instead.

Encoder: Similar to previous work (Vinyals et al., 2015; Xu et al., 2015; Karpathy and Fei-Fei, 2015; Venugopalan et al., 2017), we encode images into representations using a ResNet (He et al., 2016) model pre-trained on the ImageNet dataset (Deng et al., 2009) and use the outputs before the last fully-connected layer.

Decoder: We employ a Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) based language model to decode image representations into template captions. We provide the LSTM the image representation, I, as the initial hidden state. At the t^{th} step, the model predicts the probabilities of words/slots, y_t, based on the word/slot generated at last time step, y_{t-1}, as well as the hidden state, s_t.

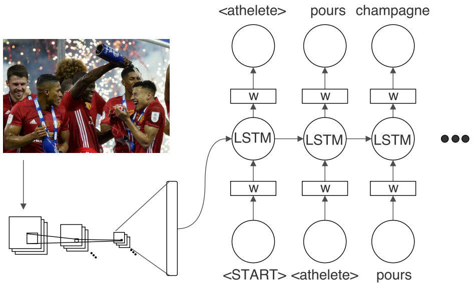

Figure 3: LSTM language generator.

4 Template Caption Entity Population

With the generated template captions, our next step is to fill in the slots with the appropriate specific entities to make the caption complete and entity-aware. In this section, we expand our method to extract candidate entities from contextual information (*i.e.*, images in Flickr with the same tags). Once we extract candidate entities, we apply the Quantified Collective Validation (QCV) algorithm (Wang et al., 2015), which constructs a number of candidate graphs and performs collective validation on those candidate graphs to choose the appropriate entities for the slots in the template caption.

---

[5] This yields a total of 95 types after manually cleaning.

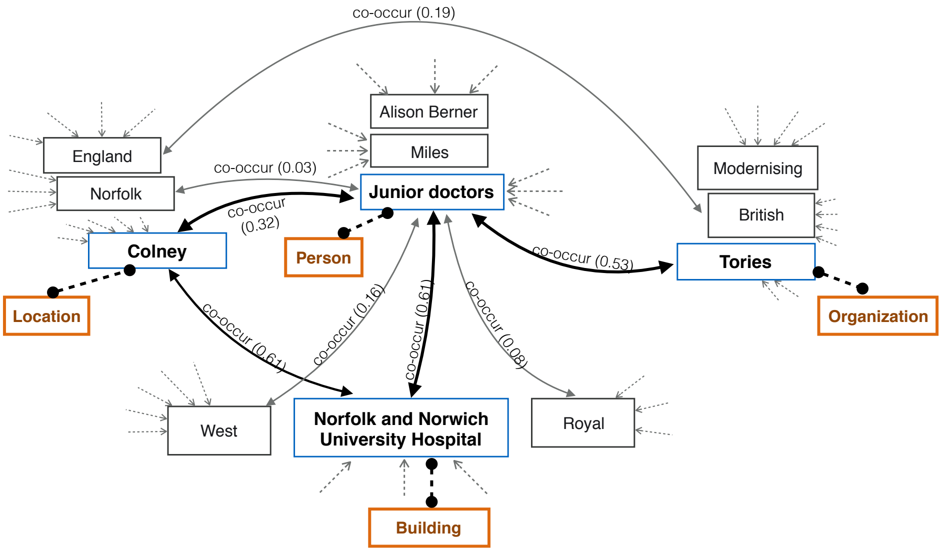

Figure 4: Knowledge graph construction.

4.1 Candidate Entity Retrieval

Users on social media typically post images with tags that are event-related (e.g. #occupywallstreet), entity-related (e.g. #lakers), or topic-related (e.g. #basketball). On our Flickr testing dataset, the average number of tags associated with an image is 11.5 and posts with the same tags likely share common entities. Therefore, given an image and its tags, we retrieve images from Flickr with the same tags by a window size of seven-day based on taken date of the photo, and then utilize the textual information accompanying the retrieved images as context. We filter out the high frequency hashtags($>$ 200 in testing dataset). Because some common tags, such as '#concert', appear in lots of posts related to different concerts.

Given the related text, we apply EDL algorithms (Pan et al., 2015) to extract named entities and link them to DBpedia to obtain their entity types. For each entity type, we rank the candidates based on their frequency in the context and only keep the top 5 candidate entities.

4.2 Quantified Collective Validation

Each slot in the template must be filled with an entity from its corresponding candidate entities. We can regard this step as an entity linking problem, whose goal is to choose an entity from several candidates given an entity mention. We utilize the QCV algorithm to construct a number of candidate graphs for a given set of slots (Wang et al., 2015), where each combination of candidate entities substituted into the slots yields a different graph (Figure 4). For each candidate combination graph, G_c^i, we compute the edge weights between each pair of candidates in the graph as

$$H_r = \frac{f_{c_h c_t}}{max(f_{c_h}, f_{c_t})} \tag{1}$$

where $r \in E(G_c^i)$ is an edge in G_c^i, c_h and c_t are the head candidate and tail candidate of the edge, $f_{c_h c_t}$ is the co-occurrence frequency of the pair of candidates, and f_{c_h} and f_{c_t} are the individual frequencies of head candidate and tail candidate, respectively. For example, in Figure 4, *Colney* (`Location`) and *Junior doctors* (`Person`) co-occur frequently, therefore the edge between them has a larger weight.

We compute the summed edge weight, $\omega(G_c^i)$, for each G_c^i by

$$\omega(G_c^i) = \sum_{r \in E(G_c^i)} H_r \tag{2}$$

and select the combination of candidates with the largest $\omega(G_c^i)$ to fill in the set of slots.

As a result of this process, given the template: '`<Person>` holding signs protest against `<Organization>` outside `<Building>` in `<Location>`.', we obtain an entity-aware caption: '***Junior doctors** holding signs protest against **Tories** outside **Norfolk and Norwich University Hospital** in **Colney***'.

4.3 Post-processing

Some images in Flickr have EXIF data which gives the date that an image is taken. We convert this information into the format such as '***April 26 2016***' and add it to the generated captions as post-processing, by which we obtain the complete caption: '*Junior doctors holding signs protest against Tories outside Norfolk and Norwich University Hospital in Colney on **April 26 2016**.*'. We leave the generated caption without adding date information if it is not available. For those slots that cannot be filled by names, we use general words to replace them, such as using the word *'Athlete'* to replace the slot `Athlete`.

5 Experiments

5.1 Data

We require images with well-aligned, news-style captions for training. However, we want to test our model on real social media data and it is difficult to collect these informative captions for social media data. Therefore, we acquire training data from news outlets and testing data from social media. We select two different topics, social events and sports events, as our case studies.

	Train	Dev	Test
Number of Images	29,390	4,306	3,688
Number of Tokens	1,086,350	161,281	136,784
Social Event	6,998	976	872
Sports Event	22,392	3,330	2,816
Person	32,878	4,539	4,156
Location	42,786	5,657	5,432
Organization	17,290	2,370	2,124
Miscellaneous	8,398	1,103	1,020

Table 1: Statistics of datasets for template generation.

Template Generator Training and Testing. To train the template caption generator, we collect 43,586 image-caption pairs from Reuters[6], using topically-related keywords[7] as queries. We do not use existing image caption datasets, such as MSCOCO (Lin et al., 2014), because they do not contain many named entities. After the compression and generalization procedures (Section 3.2) we keep 37,384 images and split them into train, development, and test sets. Table 1 shows the statistics of the datasets.

Entity-aware Caption Testing. Since news images do not have associated tags, for the purpose of testing our model in a real social media setting, we use images from Flickr for our caption evaluation[8], which is an image-centric, representative social media platform. We use the same keywords as for template generator training to retrieve multi-modal data with Creative Commons license[9], for social and sports events. We choose the images that already have news-style descriptions from users and manually confirm they are well-aligned. In total, we collect 2,594 images for evaluation. For each image, we also obtain the tags (30,148 totally) and meta-data, such as EXIF and geotag data, when they are available.

5.2 Models for Comparison

We compare our entity-aware model with the following baselines:

CNN-RNN-raw. We use the model proposed by Vinyals et al. (2015) to train an image captioning model on the raw news image-caption pairs, and apply to Flickr testing data directly.

CNN-RNN-compressed. We use the model proposed by Vinyals et al. (2015) to train a model on the compressed news image-caption pairs.

Text-summarization. We apply SumBasic summarization algorithms (Vanderwende et al., 2007), that is a summarization for multiple documents based on frequency of word and semantic content units, to text documents retrieved by hashtag.

Entity-aware. We apply trained template generator on Flickr testing data, and then fill in the slots with extracted background information.

5.3 Evaluation Metrics

We use three standard image captioning evaluation metrics, BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014), ROUGE (Lin, 2004) and CIDEr (Vedantam et al., 2015), to evaluate the quality of both the generated templates and generated captions. BLEU is a metric based on correlations at the sentence level. METEOR is a metric with recall weighted higher than precision and it takes into account stemming as well as synonym matching. ROUGE is proposed for evaluation of summarization and relies

---

[6] https://www.reuters.com/

[7] Social Events: concert, festival, parade, protest, ceremony; Sports Events: Soccer, Basketball, Soccer, Baseball, Ice Hockey

[8] https://www.flickr.com/

[9] https://creativecommons.org/licenses/

highly on recall. CIDEr metric downweights the n-grams common in all image captions, which are similar to tf-idf. Since the goal of this task is to generate entity-aware descriptions, we also measure the entity F1 scores for the final captions, where we do fuzzy matching to manually count the overlapped entities between the system output and reference captions. Besides, we do human evaluation with a score in range of 0 to 3 using the criteria as follows: Score 0: generated caption is not related to the ground-truth; Score 1: generated caption is topically-related to the ground-truth, but has obvious errors; Score 2: generated caption is topically-related to the ground-truth, and has overlapped named entities; Score 3: generated caption well describes the image.

5.4 Template Evaluation

Raw: new york giants kicker josh brown kicks a field goal during the first quarter against the san francisco 49ers at metlife stadium.

Generated Coarse Template: `<Person>` runs with ball against `<Location>` `<Location>` in the first half of their `<Miscellaneous>` football game in `<Location>`

Generated Fine Template: `<Footballteam>` kicker `<Player>` kicks a field goal out of the game against the `<Footballteam>` at `<Organization>` stadium

Figure 5: Example of generated template.

Table 2 shows the performances of template generator based on coarse-grained and fine-grained type respectively, and Figure 5 shows an example of the template generated. Coarse templates are the ones after we replace names with these coarse-grained types. Entity Linking classifies names into more fine-grained types, so the corresponding templates are fine templates. The generalization method of replacing the named entities with entity types can reduce the vocabulary size significantly, which reduces the impact of sparse named entities in training data. The template generation achieves close performance with state-of-the-art generic image captioning on MSCOCO dataset (Xu et al., 2015). The template generator based on coarse-grained entity type outperforms the one based on fine-grained entity type for two reasons: (1) fine template relies on EDL, and incorrect linkings import noise; (2) named entities usually has multiple types, but we only choose one during generalization. Thus the caption, '*Bob Dylan performs at the Wiltern Theatre in Los Angeles*', is generalized into '`<Writer>` *performs at the* `<Theater>` *in* `<Loaction>`', but the correct type for *Bob Dylan* in this context should be `Artist`.

5.5 Flickr Caption Results

Table 4 shows the comparison between our model and the baselines. The scores are much lower than traditional caption generation tasks such as COCO, because we use the real captions as ground-truth. Our model outperforms all the baselines on all metrics except BLEU-4, where Text-summarization model achieves better score. Generally, the model based on textual features (Text-summarization) has better performance than vision-based models (CNN-RNN-raw and CNN-RNN-compressed). It indicates textual summarization algorithm is more effective when it involves specific knowledge generation. Text-summarization model generates results from documents retrieved by hashtags, so it tends to include some long phrases common in those documents. However the templates generated by our model is based on the language model trained from the news captions, which has different style with Flickr captions. It results in that Text-summarization model achieves better BLEU-4 score. Our model improves CIDEr score more significantly compared with other metrics, because CIDEr downweights the n-grams common in all captions, where more specific information such as named entities contribute more to the scores. The experimental results demonstrate that our model is effective to generate image captions with specific knowledge.

5.6 Analysis

Figure 6 shows some examples of the captions generated by the entity-aware model.

Good Examples: (A) in Figure 6 describes the events in the images well ('*performs*') and include correct, well-placed entities, such as '*Little Dragon*' and '*House of Blues*'.

Approach	Vocabulary	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE	CIDEr
Raw-caption	10,979	15.1	11.7	9.9	8.8	8.8	24.2	34.7
Coarse Template	3,533	**46.7**	**36.1**	**29.8**	**25.7**	**22.4**	**43.5**	161.6
Fine Template	3,642	43.0	33.4	27.8	24.3	20.3	39.8	**165.3**

Table 2: Comparison of Template Generator with coarse/fine-grained entity type. Coarse Template is generalized by coarse-grained entity type (name tagger) and fine template is generalized by fine-grained entity type (EDL).

#littledragon#houseofblues
#cleveland#ohio#concert

EXIF: 2017-08-02 20:42:53
(A)

#mlb#baseball
#orioleparkatcamdenyards
#joekelly#bostonredsox

EXIF: 2016-06-01 19:25:01
(B)

#toronto#tiff#tiff17
#tiff2017#raptors#patrick
#patterson#patrickpatterson

EXIF: 2017-09-09 15:38:21
(C)

	Model	Caption
A	CNN-RNN-compressed	jack white from rock band the dead weather performs during the 44th montreux jazz festival in montreux
	Text-summarization	Little Dragon performing at the House of Blues in Cleveland, OH
	Entity-aware(ours)	singer **little dragon** performs at the **house of blues** in **cleveland** August 2 2017
	Human	**little dragon** performing at the **house of blues** in **cleveland, oh**
B	CNN-RNN-compressed	houston astros starting pitcher brett delivers in the second inning against the cleveland indians at progressive field
	Text-summarization	Red Sox at Orioles 6/2/16
	Entity-aware(ours)	**baltimore orioles** starting pitcher **joe kelly** pitches in the first inning against the **baltimore orioles** at **baltimore** June 1 2016
	Human	**joe kelly** of the **boston red sox** pitches in a game against the **baltimore orioles** at **oriole park** at camden yards on june 1, 2016 in **baltimore, maryland**
C	CNN-RNN-compressed	protestors gesture and hold signs during a protest against what demonstrators call police brutality in mckinney , texas .
	Text-summarization	Toronto, Canada September 9, 2017.
	Entity-aware(ours)	supporters of an ban protest outside the **toronto international film festival** in **toronto** September 9 2017
	Human	**patrick patterson** at the premiere of the **carter effect**, 2017 **toronto film festival**

Figure 6: Examples of generated entity-aware caption.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE	CIDEr	F1	Human*
CNN-RNN-raw	7.5	2.7	0.9	0.3	2.6	7.5	1.1	1.2	0.49
CNN-RNN-compress	8.3	3.3	1.1	0.5	3.0	9.2	1.5	2.1	0.50
Text-summarization	9.5	8.0	7.1	**6.5**	9.9	11.9	17.2	35.4	0.59
Entity-aware(ours)	**25.5**	**14.9**	**8.0**	4.7	**11.0**	**21.1**	**29.9**	**39.7**	**0.87**

Table 4: Comparison between our entity-aware model and baseline models on various topics. (* We make human evaluation on 259 images randomly selected.)

Relation Error of Filled Entities: Some of our errors result from ignoring of relations between entities. In Example (B) of Figure 6 our model generate a good template, but connects '*Joe Kelly*', who is actually a pitcher of '*Res Sox*', with '*Baltimore Orioles*' incorrectly. One possible solution is to incorporate relation information when the model fills in the slots with entities.

Template Error: Another category of errors results from wrong templates generated by our model. Examples (C) in Figure 6 is about a film festival, but the model generates a template about protest, which is not related to the image. One potential improvement is to incorporate the information from associated tags, such as the number of tags and the named entity types related to the tags, as features during template caption generation to make generated templates dynamically change according to the context.

6 Related Work

The goal of image captioning is to automatically generate a natural language sentence given an image. (Kulkarni et al., 2013; Yang et al., 2011; Mitchell et al., 2012) perform object recognition in images and fill hand-made templates with the recognized objects. (Kuznetsova et al., 2012, 2014) retrieve similar images, parse associated captions into phrases, and compose them into new sentences. Due to the use of static, handmade templates, these approaches are unable to generate a variety of sentence realizations, which can result in poorly generated sentences and requires one to manually create more templates to extend the generation. Our approach overcomes this by dynamically generating the output.

More recent work utilizes neural networks and applies an encoder-decoder model (Cho et al., 2014). Vinyals et al. (2015) use a CNN to encode images into a fixed size vector representation and a LSTM to decode the image representations into a sentence. Xu et al. (2015) incorporate an attention mechanism (Bahdanau et al., 2015) and attend to the output from a convolutional layer of a CNN to produce the image representations for the decoder. Instead of encoding a whole image as a vector, (Johnson et al., 2016) apply R-CNN object detection (Girshick et al., 2014), match text snippets to the regions of the image detected by the R-CNN, and use a recurrent neural network (RNN) language model, similar to (Vinyals et al., 2015), to generate a description of each region.

The surface realization for state-of-the-art neural approaches is impressive, but, in the context of generating entity-aware captions, these methods fall short as they heavily rely on training data for language modeling. (Tran et al., 2016) leverage face and landmark recognition to generate captions containing named persons, but such large-scale training is difficult. Consequently, OOV words like named entities, which are a quintessential aspect of entity-aware captions because OOV words typically represent entities or events, are difficult to generate due to low training probabilities. Some work has been done to incorporate novel objects into captions (Venugopalan et al., 2017), but this does not address the need to generate entity-aware captions and incorporate contextual information; rather, it gives the ability to generate more fine-grained entities within captions that still lack the necessary context. (Feng and Lapata, 2013) also generates a caption with named entities, but from associated news articles, in which there is much more textual context than our setting. Our approach uses neural networks to generate dynamic templates and then fills in the templates with specific entities. Thus, we are able to combine the sentence variation and surface realization quality of neural language modeling and the capability to incorporate novel words of template-based approaches.

7 Conclusions and Future Work

In this paper we propose a new task which aims to automatically generate entity-aware image descriptions with limited textual information. Experiments on a new benchmark dataset collected from Flickr show that our approach generates more informative captions compared to traditional image captioning methods. Moreover, our two-step approach can easily be applied to other language generation tasks involving specific information.

In the future, we will expand the entity-aware model to incorporate the relations between candidates when the model fills in the slots, which can avoid the cases such as *'Cristiano Ronaldo of Barcelona'*. We will also make further research on context-aware fine-grained entity typing to train a better template generator. Another research direction based on this work is to develop an end-to-end neural architecture to make the model more flexible without generating a template in the middle.

Acknowledgments

This work was supported by the U.S. DARPA AIDA Program No. FA8750-18-2-0014 and U.S. ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the third International Conference on Learning Representations*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*.

Yansong Feng and Mirella Lapata. 2013. Automatic caption generation for news images. *IEEE transactions on pattern analysis and machine intelligence*, 35(4):797–812.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE conference on computer vision and pattern recognition*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4565–4574.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*.

Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2013. Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2891–2903.

Polina Kuznetsova, Vicente Ordonez, Alexander C Berg, Tamara L Berg, and Yejin Choi. 2012. Collective generation of natural image descriptions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Polina Kuznetsova, Vicente Ordonez, Tamara L Berg, and Yejin Choi. 2014. Treetalk: Composition and compression of trees for image descriptions. *Transactions of the Association of Computational Linguistics*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Proceedings of the 2014 European Conference on Computer Vision*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Margaret Mitchell, Xufeng Han, Jesse Dodge, Alyssa Mensch, Amit Goyal, Alex Berg, Kota Yamaguchi, Tamara Berg, Karl Stratos, and Hal Daumé III. 2012. Midge: Generating image descriptions from computer vision detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics–Human Language Technologies*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics–Human Language Technologies Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Kenneth Tran, Xiaodong He, Lei Zhang, Jian Sun, Cornelia Carapcea, Chris Thrasher, Chris Buehler, and Chris Sienkiewicz. 2016. Rich image captioning in the wild. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Van-Khanh Tran and Le-Minh Nguyen. 2017. Natural language generation for spoken dialogue system using rnn encoder-decoder networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning*.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management*, 43(6):1606–1618.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2017. Captioning images with diverse objects. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*.

Han Wang, Jin Guang Zheng, Xiaogang Ma, Peter Fox, and Heng Ji. 2015. Language and domain independent entity linking with quantified collective validation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 2015 International Conference on Machine Learning*.

Yezhou Yang, Ching Lik Teo, Hal Daumé III, and Yiannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Learning to Describe Differences Between Pairs of Similar Images

Harsh Jhamtani, Taylor Berg-Kirkpatrick
Language Technologies Institute
Carnegie Mellon University
`{jharsh,tberg}@cs.cmu.edu`

Abstract

In this paper, we introduce the task of automatically generating text to describe the differences between two similar images. We collect a new dataset by crowd-sourcing difference descriptions for pairs of image frames extracted from video-surveillance footage. Annotators were asked to succinctly describe *all* the differences in a short paragraph. As a result, our novel dataset provides an opportunity to explore models that align language and vision, and capture visual salience. The dataset may also be a useful benchmark for coherent multi-sentence generation. We perform a first-pass visual analysis that exposes clusters of differing pixels as a proxy for object-level differences. We propose a model that captures visual salience by using a latent variable to align clusters of differing pixels with output sentences. We find that, for both single-sentence generation and as well as multi-sentence generation, the proposed model outperforms the models that use attention alone.

Man by yellow poles in after pic wasn't there before.
There are two people in middle of court that were not there earlier.
Person crossing crosswalk is no longer there

The blue truck is no longer there.
A car is approaching the parking lot from the right

Figure 1: Examples from Spot-the-diff dataset: We collect text descriptions of all the differences between a pair of images. Note that the annotations in our dataset are exhaustive wrt differences in the two images i.e. annotators were asked to describe all the visible differences. Thus, the annotations contain multi-sentence descriptions.

1 Introduction

The interface between human users and collections of data is an important application area for artificial intelligence (AI) technologies. Can we build systems that effectively interpret data and present their results concisely in natural language? One recent goal in artificial intelligence has been to build models that are able to interpret and describe visual data to assist humans in various tasks. For example, image captioning systems (Vinyals et al., 2015b; Xu et al., 2015; Rennie et al., 2017; Zhang et al., 2017) and visual question answering systems (Antol et al., 2015; Lu et al., 2016; Xu and Saenko, 2016) can help visually impaired people in interacting with the world. Another way in which machines can assist humans is by identifying meaningful patterns in data, selecting and combining salient patterns, and generating concise and fluent 'human-consumable' descriptions. For instance, text summarization (Mani and Maybury, 1999; Gupta and Lehal, 2010; Rush et al., 2015) has been a long standing problem in natural language processing aimed at providing a concise text summary of a collection of documents.

In this paper, we propose a new task and accompanying dataset that combines elements of image captioning and summarization: the goal of 'spot-the-diff' is to generate a succinct text description of *all* the salient differences between a pair of similar images. Apart from being a fun puzzle, solutions to this task may have applications in assisted surveillance, as well as computer assisted tracking of changes in media assets. We collect and release a novel dataset for this task, which will be potentially useful for both natural language and computer vision research communities. We

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4024–4034
Brussels, Belgium, October 31 - November 4, 2018.

used crowd-sourcing to collect text descriptions of differences between pairs of image frames from video-surveillance footage (Oh et al., 2011), asking annotators to succinctly describe *all* salient differences. In total, our datasets consist of descriptions for 13,192 image pairs. Figure 1 shows a sample data point - a pair of images along with a text description of the differences between the two images as per a human annotator.

There are multiple interesting modeling challenges associated with the task of generating natural language summaries of differences between images. First, not all low-level visual differences are sufficiently salient to warrant description. The dataset presents an interesting source of supervision for methods that attempt to learn models of visual salience (we additionally conduct exploratory experiments with a baseline salience model, as described later). Second, humans use different levels of abstraction when describing visual differences. For example, when multiple nearby objects have all moved in coordination between images in a pair, an annotator may refer to the group as a single concept (e.g. 'the row of cars'). Third, given a set of salient differences, planning the order of description and generating a fluent sequence of multiple sentences is itself a challenging problem. Together, these aspects of the proposed task make it a useful benchmark for several directions of research.

Finally, we experiment with neural image captioning based methods. Since salient differences are usually described at an object-level rather than at a pixel-level, we condition these systems on a first-pass visual analysis that exposes clusters of differing pixels as a proxy for object-level differences. We propose a model which uses latent discrete variables in order to directly align difference clusters to output sentences. Additionally we incorporate a learned prior that models the visual salience of these difference clusters. We observe that the proposed model which uses alignment as a discrete latent variable outperforms those that use attention alone.

2 'Spot-the-diff' Task and Dataset

We introduce 'spot-the-diff' dataset consisting of 13,192 image pairs along with corresponding human provided text annotations stating the differences between the two images. Our goal was to create a dataset wherein there are meaningful differences between two similar images. To achieve this, we work with image frames extracted from VIRAT surveillance video dataset (Oh et al., 2011), which consists of 329 videos across 11 frames of reference totalling to about 8.5 hours of videos.

Total number of annotations	$13,192$
Mean (std dev.) number of sentences per annotation	$1.86(1.01)$
Vocabulary size	2404
Frequent word types (>=5 occurrences)	1000
Word tokens that are frequent word types	97%
Mean (std dev.) number of words in sentence:	$10.96(4.97)$
% Long sentences ($>$ 20 words)	5%

Table 1: Summary statistics for spot-the-diff dataset

2.1 Extracting Pairs of Image Frames

To construct our dataset, we first need to identify image pairs such that some objects have changed positions or have entered or left in the second image compared to the first image. To achieve this, we first extract a certain number of randomly selected image frame pairs from a given video. Thereafter, we compute the L_2 distance between the two images in each pair (under RGB representation). Finally, we set a lower and a upper threshold on the L_2 distance values so calculated to filter out the image pairs with potentially too less or too many changes. These thresholds are selected based on manual inspection. The resulting image pairs are used for collecting the difference descriptions.

2.2 Human Annotation

We crowd-sourced natural language differences between images using Amazon Mechanical Turk. We restrict to annotators from primarily Anglophone countries: USA, Australia, United Kingdom, and Canada, as we are working with English language annotations. We limit to those participants which have lifetime HIT $> 80\%$. We award 5 cents per HIT (Human Intelligence Task) to participants. We provide the annotators with an example on how to work on the task. We request the annotators to write complete English sentences, with each sentence on a separate line. We collect a total of 13192 annotations.

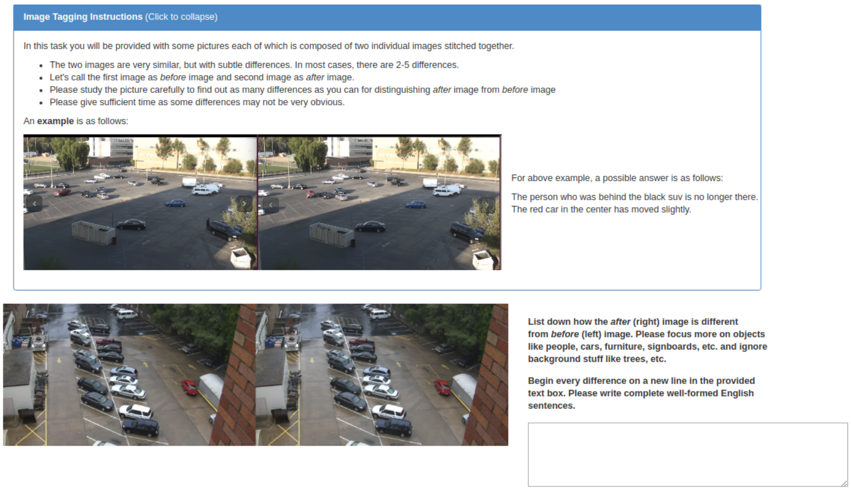

Image Tagging Instructions (Click to collapse)

In this task you will be provided with some pictures each of which is composed of two individual images stitched together.

- The two images are very similar, but with subtle differences. In most cases, there are 2-5 differences.
- Let's call the first image as *before* image and second image as *after* image.
- Please study the picture carefully to find out as many differences as you can for distinguishing *after* image from *before* image
- Please give sufficient time as some differences may not be very obvious.

An **example** is as follows:

For above example, a possible answer is as follows:

The person who was behind the black suv is no longer there.
The red car in the center has moved slightly.

List down how the *after* (right) image is different from *before* (left) image. Please focus more on objects like people, cars, furniture, signboards, etc. and ignore background stuff like trees, etc.

Begin every difference on a new line in the provided text box. Please write complete well-formed English sentences.

Figure 2: AMT (Amazon Mechanical Turk) HIT (Human Intelligence Task) setup for data collection. We provide the annotators with detailed instructions, along with an example showing how to perform the task. We request the annotators to write complete English sentences, with each sentence on a separate line. We collect a total of 13,192 annotations.

Dataset	BLEU-1/2/3/4	ROUGE-L
Spot-the-diff ($A = 3$)	$0.41/0.25/0.15/0.08$	0.31
MS-COCO ($A = 3$)	$0.38/0.22/0.13/0.08$	0.34
MS-COCO ($A = 5$)	$0.66/0.47/0.32/0.22$	0.48

Table 2: Human agreement for our dataset: We report measures such as BLEU and ROUGE when 'evaluating' one set of human generated captions against the remaining sets. $A = k$ represents k captions per data point, out of which 1 is chosen as hypothesis, while remaining $k - 1$ act as references.

2.3 Dataset statistics

Table 1 shows some summary statistics about the collected dataset. Since we deal with a focused domain, we observe a small vocabulary size. On an average there are 1.86 reported differences / sentences per image pair. We also report inter-annotator agreement as measured using text overlap of multiple annotations for the same image pair. We collect three sets of annotations for a small subset of the data (467 data points) for the purpose of reporting inter-annotator agreements. We thereby calculate BLEU and ROUGE-L scores by treating one set of annotations as 'hypothesis' while remaining two sets act as 'references'(Table 2). We repeat the same analysis for MS-COCO dataset and report these measures for reference. The BLEU and METEOR values for our dataset seem reasonable and are comparable to the values observed for MS-COCO dataset.

3 Modeling Difference Description Generation

We propose a neural model for describing visual difference based on the input pair of images that uses latent alignment variable to capture visual salience. Since most descriptions talk about higher-level differences rather than individual pixels, we first perform a visual analysis that precomputes a set of difference clusters in order to approximate object-level differences, as described next. The output of this analysis is treated as input to a neural encoder-decoder text generation model that incorporates a latent alignment variable and is trained on our new dataset.

3.1 Exposing Object-level Differences

We first analyze the input image pair for the pixel-level differences by computing a *pixel-difference mask*, followed by a local spatial analysis which segments the difference mask into clusters that approximate the set of object-level differences. Thereafter, we extract image features using convolutional neural models and use these as input to a neural text generation model, described later.

Pixel-level analysis: The lowest level of visual difference is individual differences between corresponding pixels in the input pair. Instead of requiring our description model to learn to compute pixel-level differences as a first step, we precompute and directly expose these to the model. Let $X = (\boldsymbol{I_1}, \boldsymbol{I_2})$ represent the image pair in a datum. For each such image pair in our dataset, we obtain a corresponding pixel-difference mask $\boldsymbol{M}$. M is a *binary-valued* matrix of the same dimensions (length and width) as each of the images in the corresponding image pair, wherein each element in the matrix is 1 (active) if the corresponding pixel is *different* between the input pair, and

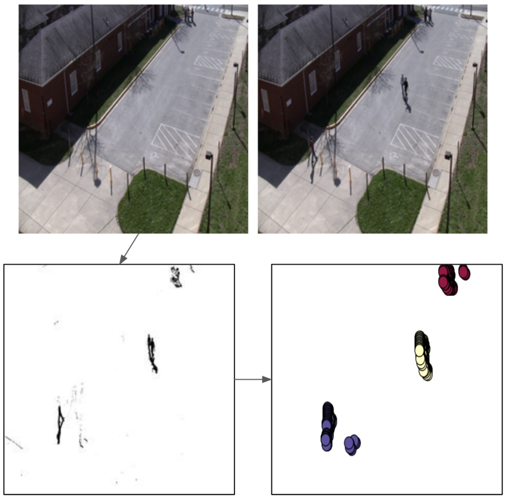

Figure 3: Exposing Object-level Differences: Before training a model to describe visual difference, we first compute pixel-level differences, as well as a segmentation of these differences into clusters, as a proxy for exposing object-level differences. The first row shows the original image pair. Bottom left depicts the pixel-difference mask, which represents extracted pixel-level differences. The segmentation of the pixel-difference mask into clusters is shown in the bottom right.

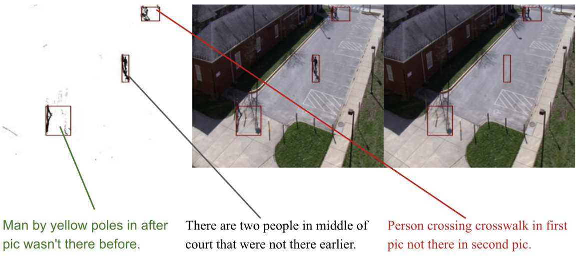

Figure 4: The figure shows the *pixel-difference* mask for the running example, along with the two original images, with bounding boxes around clusters. Typically one or more difference clusters are used to frame one reported difference / sentence, and it is rare for a difference cluster to participate in more than one reported difference.

0 otherwise. To decide whether a pair of corresponding pixels in the input image pair are sufficiently *different*, we calculate the L_2-distance between the vectors corresponding to each pixel's color value (three channels) and check whether this difference is greater than a threshold δ (set based on manual inspections).

While the images are extracted from supposedly still cameras, we do find some minor shifts in the camera alignment, which is probably due to occasional wind but may also be due to manual human interventions. These shifts are rare and small, and we align the images in the pair by iterating over a small range of vertical and horizontal shifts to find the shift with minimum corresponding L_2-distance between the two images.

Object-level analysis: Most visual descriptions refer to object-level differences rather than pixel-level differences. Again, rather than requiring the model to learn to group pixel differences into objects, we attempt to expose this to the model via pre-processing. As a proxy for object-level difference, we segment the pixel-level differences in the pixel-difference mask into clusters, and pass these clusters as additional inputs to the model. Based on manual inspection, we find that with the right clustering technique, this process results in groupings that roughly correspond to objects that have moved, appeared, and disappeared between the input pair. Here, we find that density based clustering algorithms like DBScan (Ester et al., 1996) work well in practice for this purpose. In our scenario, the DBScan algorithm predicts clusters of nearby active pixels, and marks outliers consisting of small groups of isolated active pixels, based on a calculation of local density. This also serves as a method for pruning any noisy pixel differences which may have passed through the pixel-level analysis.

As the output of DBScan, we obtain segmentation of the pixel difference matrix M into *difference clusters*. Let the number of *difference clusters* be represented by K (DBScan is a non-parametric clustering method, and as such the number of clusters K is different for each data point.). Now, let's define $\boldsymbol{C_k}$ as another binary-valued mask matrix such that the elements in matrix corresponding to the k^{th} difference cluster are 1 (active) while rest of the elements are 0.

3.2 Text Generation Model

We observe from annotated data that each individual sentence in a full description typically refers only to visual differences within a single cluster (see Figure 4). Further, on average, there are more clusters than there are sentences. While many uninteresting and *noisy* pixel-level differences get screened out in preprocessing, some uninteresting clusters are still identified. These are unlikely to be described by annotators because, even though they correspond to legitimate visual differences, they are not visually salient. Thus, we can roughly model description generation as a cluster selection process.

In our model, which is depicted in Figure 5, we

X=(I_1,I_2)	:	Image pair in the datum
M	:	Pixel-difference mask is a binary-valued matrix depicting pixel-level changes
F_1, F_2	:	Image feature tensors for I_1 and I_2 respectively
K	:	Number of segments
C_k	:	Cluster mask corresponding to k^{th} difference cluster
T	:	Number of reported differences / sentences
z_i	:	Discrete alignment variable for the i^{th} sentence. $z_i \in \{1, 2, ..., K\}$
$S_1, .., S_T$	:	List of T Sentences

Table 3: Summary of notation used in description of the method.

assume that each output description, which consists of sentences $S_1, \ldots, S_T$, is generated sentence by sentence conditioned on the input image pair $X = (I_1, I_2)$. Further, we let each sentence S_i be associated with a latent alignment variable, $z_i \in \{1, \ldots, K\}$, that chooses a cluster to focus on (Vinyals et al., 2015a). The choice of z_i is itself conditioned on the input image pair, and parameterized in a way that lets the model learn which types of clusters are visually salient and therefore likely to be described as sentences. Together, the probability of a description given an image pair is given by:

$$P(S_1, .., S_T|X) \\ = \sum_{z_1,..,z_T} \prod_{i=1}^{T} \underbrace{P(S_i|z_i, X; \theta))}_{\text{decoder}} \underbrace{P(z_i|X; w)}_{\text{alignment prior}} \quad (1)$$

The various components of this equation are described in detail in the next few sections. Here, we briefly summarize each. The term $P(z_i|X; w)$ represents the prior over the latent variable z_i and is parameterized in a way that lets the model learn which types of clusters are visually salient. The term $P(S_i|z_i, X; \theta)$ represents the likelihood of sentence S_i given the input image pair and alignment z_i. We employ masking and attention mechanisms to encourage this decoder to focus on the cluster chosen by z_i. Each of these components conditions on visual features produced by a pre-trained image encoder.

The alignment variable z_i for each sentence is chosen independently, and thus our model is similar to IBM Model 1 (Brown et al., 1993) in terms of its factorization structure. This will allow tractable learning and inference as described in Section 3.3. We refer to our approach as DDLA (**D**ifference **D**escription with **L**atent **A**lignment).

Alignment prior: We define a learnable prior over alignment variable z_i. In particular, we let the multinomial distribution on z_i be parameterized in a log-linear fashion using feature function $g(z_i)$. Specifically, we consider the following four features: the length, width, and area of the smallest rectangular region enclosing cluster z_i, and the number of active elements in mask C_{z_i}. Specifically, we let $P(z_i|X; w) \propto \exp(w^T g(z_i))$.

Visual encoder: We extract images features using ResNet (He et al., 2016) pre-trained on Imagenet data. Similar to prior work (Xu et al., 2015), we extract features using a lower level convolutional layer instead of fully connected layer. In this way, we obtain image features of dimensionality $14 * 14 * 2096$, where the first two dimensions correspond to a grid of coarse, spatially localized, feature vectors. Let F_1 and F_2 represent the extracted feature tensors for I_1 and I_2 respectively.

Sentence decoder: We use an LSTM decoder (Hochreiter and Schmidhuber, 1997) to generate the sequence of words in each output sentence, conditioned on the image pair and latent alignments. We use a matrix transformation of the extracted image features to initialize the hidden state of the LSTM decoder for each sentence, independent of the setting of z_i. Additionally, we use an attention mechanism over the image features at every decoding step, similar to the previous work (Xu et al., 2015). However, instead of considering attention over the entire image, we restrict attention over image features to the cluster mask determined by the alignment variable, C_{z_i}. Specifically, we project binary mask C_{z_i} from the input image dimensionality (224*224) to the dimensionality of the visual features (14*14). To achieve this, we use pyramid reduce downsampling on a smoothed version of cluster mask C_{z_i}. The resulting projection roughly corresponds to the subset of visual features with the cluster region in their receptive field. This projection is multiplied to attention weights.

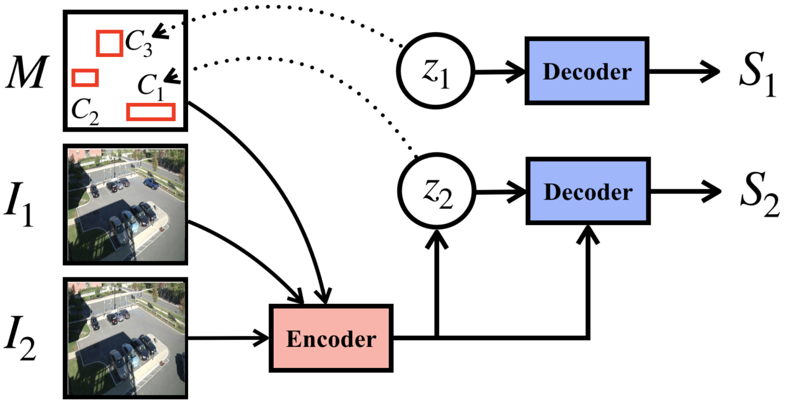

Figure 5: Model architecture for generating difference descriptions. We incorporate a discrete latent variable z which selects one of the clusters as a proxy for object-level focus. Conditioned on the cluster and visual features in the corresponding region, the model generates a sentence using an LSTM decoder. During training, each sentence in the full description receives its own latent alignment variable, z.

3.3 Learning and Decoding

Learning in our model is accomplished by stochastic gradient ascent on the marginal likelihood of each description with alignment variables marginalized out. Since alignment variables are independent of one another, we can marginalize over each z_i separately. This means running back-propagation through the decoder K times for each sentence, where K is the number of clusters. In practice K is relatively small and this direct approach to training is feasible. Following equation 1, we train both the generation and prior in an end-to-end fashion.

For decoding, we consider the following two problem settings. In the first setting, we consider the task of producing a single sentence in isolation. We evaluate in this setting by treating the sentences in the ground truth description as multiple reference captions. This setting is similar to the typical image captioning setting. In the second setting, we consider the full multi-sentence generation task where the system is required to produce a full description consisting of multiple sentences describing all differences in the input. Here, the generated multi-sentence text is directly evaluated against the multi-sentence annotation in the crowd-soured data.

Single-sentence decoding: For single sentence generation, we first select the value of z_i which maximizes the prior $P(z_i|X;w)$. Thereafter, we simply use greedy decoding to generate a sentence conditioned on the chosen z_i and the input image pair.

Multi-sentence decoding: Here, we first select a set of clusters to include in the output description, and then generate a single sentence for each cluster using greedy decoding. Since typically there are more clusters than sentences, we condition on the ground truth number of sentences and choose the corresponding number of clusters. We rank clusters by decreasing likelihood under the alignment prior and then choose the top T.

4 Experiments

We split videos used to create the dataset into train, test, and validation in the ratio 80:10:10. This is done to ensure that all data points using images from the same video are entirely in one split. We report quantitative metrics like CIDEr (Vedantam et al., 2015), BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), and ROUGE-L, as is often reported by works in image captioning. We report these measures for both sentence level setting and multi-sentence generation settings. Thereafter, we also discuss some qualitative examples. We implement our models in PyTorch (Paszke et al., 2017). We use mini-batches of size 8 and use Adam optimizer[1]. We use CIDEr scores on validation set as a criteria for early stopping.

Baseline models: We consider following baseline models: CAPT model considers soft attention over the input pair of images (This atten-

---

[1] Our data set can be obtained through `https://github.com/harsh19/spot-the-diff`

Model	Bleu 1/2/3/4	Meteor	Cider	Rouge-L	Perplexity
NN	0.226 0.111 0.057 0.026	0.102	0.120	0.201	-
CAPT	0.304 0.194 0.126 0.073	0.105	0.263	0.256	16.78
CAPT-MASKED	0.301 0.200 0.131 0.078	0.108	0.285	0.271	15.12
DDLA-UNIFORM	0.285 0.175 0.108 0.064	0.106	0.250	0.247	9.96
DDLA	0.343 0.221 0.140 0.085	0.120	**0.328**	0.286	9.73

Table 4: **Single sentence decoding:** We report automatic evaluation scores for various models under single sentence generation setting. DDLA model fares better scores than various baseline methods for all the considered measures. Both the DDLA models get much better perplexities than baseline methods.

Model	Bleu 1/2/3/4	Meteor	Cider	Rouge-L	LenRatio
NN-MULTI	0.223 0.109 0.056 0.026	0.087	0.105	0.181	1.035
CAPT-MULTI	0.262 0.146 0.081 0.045	0.094	0.235	0.174	1.042
DDLA-UNIFORM	0.243 0.143 0.085 0.051	0.094	0.217	0.213	0.778
DDLA	0.289 0.173 0.103 0.062	0.108	**0.297**	0.260	0.811

Table 5: **Multi-sentence decoding** We report automatic evaluation scores for various models under multi-sentence generation setting. DDLA model achieves better scores compared to the baseline methods. Note that these scores are not directly comparable with single sentence generation setting. **LenRatio** is the ratio of the average number of tokens in the prediction to the average number of tokens in the ground truth for the test set.

tion mechanism is similar to that used in prior image captioning works (Xu et al., 2015), except that we have two images instead of a single image input). We do not perform any masking in case of CAPT model, and simply ignore the cluster information. The model is trained to generate a single sentence. Thus, this model is similar to a typical captioning model but with soft attention over two images. CAPT-MASK model is similar to CAPT model except that it incorporates the masking mechanism defined earlier using the union of all the cluster masks in the corresponding image. We also consider a version of the CAPT model wherein the target prediction is the whole multi-sentence description – CAPT-MULTI – for this setting, we simply concatenate the sentences in any arbitrary order [2]. Additionally, we consider a nearest neighbor baseline (NN-MULTI), wherein we simply use the annotation of the closest matching training data point. We compute the closeness based on the extracted features of the image pair, and leverage sklearns (Pedregosa et al., 2011) Nearest-Neighbor module. For single sentence setting (NN), we randomly pick one of the sentences in the annotation.

We also consider a version of DDLA model with fixed uniform prior, and refer to this model as DDLA-UNIFORM . For single sentence generation, we sample z_j randomly from the uniform distribution and then perform decoding. For the multi-sentence generation setting, we employ simple heuristics to order the clusters at test time. One such heuristic we consider is to order the clusters as per the decreasing area of the bounding box (smallest rectangular area enclosing the cluster).

Results: We report various automated metrics for the different methods under single sentence generation and multi-sentence generation in Tables 4 and 5 respectively. For the single sentence generation setting, we observe that the DDLA model outperforms various baselines as per most of the scores on the test data split. DDLA-UNIFORM method performs similar to the CAPT baseline methods. For the multi-sentence generation, the DDLA model again outperforms other methods. This means that having a learned prior is useful in our proposed method. Figure 6 shows an example data point with predicted outputs by different methods.

5 Discussion and Analysis

Qualitative Analysis of Outputs We perform a qualitative analysis on the outputs to understand the drawbacks in the current methods. One apparent limitation of the current methods is the failure to explicitly model the movement of same object in the two images (Figure 7) – prior works on object tracking can be useful here. Sometimes the models get certain attributes of the objects wrong. e.g. 'blue car' instead of 'red car'. Some output predictions state an object to have 'appeared' instead of 'disappeared' and vice

---

[2] Note that we do not provide CAPT-MULTI with ground truth number of sentences

HUMAN: A white truck has appeared in the after image. A person is now walking on the footpath.
DDLA (multi-sentence): A white truck appeared on the road. There is a person walking in the after image
CAPT-multi (multi-sentence): There is a car. There is a person walking in the parking lot.

HUMAN: There are more people in the group.
DDLA (multi-sentence): There are more people in the after image
CAPT-multi (multi-sentence): The people in the right image.

Figure 6: Predictions from various methods for two input image pairs.

DDLA: The blue truck is gone.

Figure 7: Some drawbacks with the current models: One apparent drawback with the single cluster selection is that it misses opportunity to identify an object which has moved significantly- considering it as appeared or disappeared as the case may be. In this example, the blue truck moved, but the DDLA model predicts that the truck is no longer there.

versa.

Do models learn alignment between sentence and difference clusters? We performed a study on 50 image pairs by having two humans manually annotate gold alignments between sentences and difference clusters. We then computed alignment precision for the model's predicted alignments. To obtain model's predicted alignment for a given sentence S_i, we compute $argmax_k P(z_i = k|X)P(S_i|z_i = k, X)$. Our proposed model achieved a precision of 54.6%, an improvement over random chance at 27.4%.

Clustering for pre-processing Our generation algorithm assumed one sentence uses only one cluster and as such we tune the hyper-parameters of clustering method to get large clusters so that typically a cluster will entirely contain a reported difference. On inspecting randomly selected data points, we observe that in some cases too large clusters are marked by the clustering procedure. One way to mitigate this is to tune clustering parameters to get smaller clusters and update the generation part to use a subset of clusters. As mentioned earlier, we consider clustering as a means to achieve object level pre-processing. One possible future direction is to leverage pre-trained object detection models to detect cars, trucks, people, etc. and make these predictions readily available to the generation model.

Multi-sentence Training and Decoding As mentioned previously, we query the models for a desired number of 'sentences'. In future works we would like to relax this assumption and design models which can predict the number of sentences as well. Additionally, our proposed model doesn't not explicitly ensure consistency in the latent variables for different sentences of a given data point i.e the model does not make explicit use of the fact that sentences report non-overlapping visual differences. Enforcing this knowledge while retaining the feasibility of training is a potential future direction of work.

6 Related Work

Modeling pragmatics: The dataset presents an opportunity to test methods which can model pragmatics and reason about semantic, spatial and visual similarity to generate a textual description of what has changed from one image to another. Some prior work in this direction (Andreas and Klein, 2016; Vedantam et al., 2017) contrastively describe a target scene in presence of a distractor. In another related task – referring expression comprehension (Kazemzadeh et al., 2014; Mao et al., 2016; Hu et al., 2017) – the model has to identify which object in the image is being referred to by the given sentence. However, our proposed task comes with a pragmatic goal related to summarization: the goal is to identify and describe *all* the differences. Since the goal is well defined, it may be used to constrain models that attempt to learn how humans describe visual difference.

Natural language generation: Natural language generation (NLG) has a rich history of previous work, including, for example, recent works on biography generation (Lebret et al., 2016), weather report generation (Mei et al., 2016), and recipe generation (Kiddon et al., 2016). Our task can viewed as a potential benchmark for coherent multi-sentence text generation since it involves assembling multiple sentences to succinctly cover a set of differences.

Visual grounding: Our dataset may also provide a useful benchmark for training unsupervised and semi-supervised models that learn to align vision and language. Plummer et al. (2015) collected annotation for phrase-region alignment in an image captioning dataset, and follow up work has attempted to predict these alignments (Wang et al., 2016; Plummer et al., 2017; Rohrbach et al., 2016). Our proposed dataset poses a related alignment problem: attempting to align sentences or phrases to visual differences. However, since differences are contextual and depend on visual comparison, our new task may represent a more challenging scenario as modeling techniques advance.

Image change detection: There are some works on land use pattern change detection ((Radke et al., 2005)). These works are related since they try to screen out noise and mark the regions of change between two images of same area at different time stamps. Bruzzone and Prieto (2000) propose an unsupervised change detection algorithms aim to discriminate between changed and unchanged pixels for multi-temporal remote sensing images. Zanetti and Bruzzone (2016) propose a method that allows unchanged class to be more complex rather than having a single unchanged class. Though image diff detection is part of our pipeline, our end task is to generate natural language descriptors. Moreover, we observe that simple clustering seems to work well for our dataset.

Other relevant works: Maji (2012) aim to construct a lexicon of parts and attributes by formulating an annotation task where annotators are asked to describe differences between two images. Some other related works model phrases describing change in color (Winn and Muresan, 2018), move-by-move game commentary for describing change in game state (Jhamtani et al., 2018), and code commit message summarizing changes in code-base from one commit to another (Jiang et al., 2017). There exist some prior works on fine grained image classification and captioning (Wah et al., 2014; Nilsback and Zisserman, 2006; Khosla et al., 2011). The premise of such works is that it is difficult for machine to find discriminative features between similar objects e.g. birds of different species. Such works are relevant for us as the type of data we deal with are usually of same object or scene taken at a different time or conditions.

7 Conclusion

In this paper, we proposed the new task of describing differences between pairs of similar images and introduced a corresponding dataset. Compared to many prior image captioning datasets, text descriptions in the ‘Spot-the-diff’ dataset are often multi-sentence, consisting of all the differences in two similar images in most of the cases. We performed exploratory analysis of the dataset and highlighted potential research challenges. We discuss how our ’Spot-the-diff’ dataset is useful for tasks such as language vision alignment, referring expression comprehension, and multi-sentence generation. We performed pixel and object level preprocessing on the images to identify clusters of differing pixels. We observe that the proposed model which aligns clusters of differing pixels to output sentences performs better than the models which use attention alone. We also discuss some limitations of current methods and scope for future directions.

Acknowledgements

We are thankful to anonymous EMNLP reviewers for their valuable suggestions. We thank Eric Nyberg for discussions on dataset collection. We also acknowledge Nikita Duseja and Varun Gangal for helping with the proof-reading of the paper. We thank Luo (2017) for releasing a PyTorch implementation of many popular image captioning models. This project was supported in part by a Adobe Research gift. Opinions and findings in this paper are of the authors, and do not necessarily reflect the views of Adobe.

References

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4,2016*, pages 1173–1182.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Lorenzo Bruzzone and Diego F Prieto. 2000. Automatic analysis of the difference image for unsupervised change detection. *IEEE Transactions on Geoscience and Remote sensing*, 38(3):1171–1182.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2017. Modeling relationships in referential expressions with compositional modular networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4418–4427. IEEE.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Learning to generate move-by-move commentary for chess games from large-scale social forum data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1661–1671.

Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 135–146. IEEE.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. 2011. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, volume 2, page 1.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Ruotian Luo. 2017. An image captioning codebase in pytorch. `https://github.com/ruotianluo/ImageCaptioning.pytorch`.

Subhransu Maji. 2012. Discovering a lexicon of parts and attributes. In *Computer Vision–ECCV 2012. Workshops and Demonstrations*, pages 21–30. Springer.

Inderjeet Mani and Mark T Maybury. 1999. *Advances in automatic text summarization*. MIT press.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20.

Hongyuan Mei, TTI UChicago, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of NAACL-HLT*, pages 720–730.

M-E. Nilsback and A. Zisserman. 2006. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454.

Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. 2011. A large-scale benchmark dataset for event recognition in surveillance video. In *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*, pages 3153–3160. IEEE.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Bryan A Plummer, Arun Mallya, Christopher M Cervantes, Julia Hockenmaier, and Svetlana Lazebnik. 2017. Phrase localization and visual relationship detection with comprehensive image-language cues. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1928–1937.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2641–2649. IEEE.

Richard J Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. 2005. Image change detection algorithms: a systematic survey. *IEEE transactions on image processing*, 14(3):294–307.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, volume 1, page 3.

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *European Conference on Computer Vision*, pages 817–834. Springer.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1070–1079. IEEE.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Catherine Wah, Grant Van Horn, Steve Branson, Subhransu Maji, Pietro Perona, and Serge Belongie. 2014. Similarity comparisons for interactive fine-grained categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–866.

Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016. Learning deep structure-preserving image-text embeddings. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5005–5013.

Olivia Winn and Smaranda Muresan. 2018. lightercan still be dark: Modeling comparative color descriptions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 790–795.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pages 451–466. Springer.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81.

Massimo Zanetti and Lorenzo Bruzzone. 2016. A generalized statistical model for binary change detection in multispectral images. In *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*, pages 3378–3381. IEEE.

Li Zhang, Flood Sung, Feng Liu, Tao Xiang, Shaogang Gong, Yongxin Yang, and Timothy M Hospedales. 2017. Actor-critic sequence training for image captioning. *arXiv preprint arXiv:1706.09601*.

Object Hallucination in Image Captioning

Anna Rohrbach^{*1}**, Lisa Anne Hendricks**^{*1}**,**
Kaylee Burns^{1} **, Trevor Darrell**^{1}**, Kate Saenko**^{2}
^{1} UC Berkeley, ^{2} Boston University

Abstract

Despite continuously improving performance, contemporary image captioning models are prone to "hallucinating" objects that are not actually in a scene. One problem is that standard metrics only measure similarity to ground truth captions and may not fully capture image relevance. In this work, we propose a new image relevance metric to evaluate current models with veridical visual labels and assess their rate of object hallucination. We analyze how captioning model architectures and learning objectives contribute to object hallucination, explore when hallucination is likely due to image misclassification or language priors, and assess how well current sentence metrics capture object hallucination. We investigate these questions on the standard image captioning benchmark, MSCOCO, using a diverse set of models. Our analysis yields several interesting findings, including that models which score best on standard sentence metrics do not always have lower hallucination and that models which hallucinate more tend to make errors driven by language priors.

1 Introduction

Image captioning performance has dramatically improved over the past decade. Despite such impressive results, it is unclear to what extent captioning models actually rely on image content: as we show, existing metrics fall short of fully capturing the captions' relevance to the image. In Figure 1 we show an example where a competitive captioning model, Neural Baby Talk (NBT) (Lu et al., 2018), incorrectly generates the object "bench." We refer to this issue as object *hallucination*.

While missing salient objects is also a failure mode, captions are summaries and thus generally not expected to describe all objects in the scene. On the other hand, describing objects that are *not present* in the image has been shown to be less preferable to humans. For example, the LSMDC challenge (Rohrbach et al., 2017a) documents that correctness is more important to human judges than specificity. In another study, (MacLeod et al., 2017) analyzed how visually impaired people react to automatic image captions. They found that people vary in their preference of either coverage or correctness. For many visually impaired who value correctness over coverage, hallucination is an obvious concern.

NBT: A woman talking on a cell phone while sitting on a ***bench***.
CIDEr: **0.87**, METEOR: 0.23, SPICE: **0.22,** CHs: **1.00**, CHi: **0.33**

TopDown: A woman is talking on a cell phone.
CIDEr: 0.54, METEOR: **0.26**, SPICE: 0.13, CHs: **0.00**, CHi: **0.00**

Figure 1: Image captioning models often "hallucinate" objects that may appear in a given context, like e.g. a *bench* here. Moreover, the sentence metrics do not always appropriately penalize such hallucination. Our proposed metrics (CHAIRs and CHAIRi) reflect hallucination. For CHAIR *lower is better*.

Besides being poorly received by humans, object hallucination reveals an internal issue of a captioning model, such as not learning a very good representation of the visual scene or overfitting to its loss function.

In this paper we assess the phenomenon of object hallucination in contemporary captioning models, and consider several key questions. The

---

* Denotes equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4035–4045
Brussels, Belgium, October 31 - November 4, 2018.

first question we aim to answer is: *Which models are more prone to hallucination?* We analyze this question on a diverse set of captioning models, spanning different architectures and learning objectives. To measure object hallucination, we propose a new metric, *CHAIR (Caption Hallucination Assessment with Image Relevance)*, which captures image relevance of the generated captions. Specifically, we consider both ground truth object annotations (MSCOCO Object segmentation (Lin et al., 2014)) and ground truth sentence annotations (MSCOCO Captions (Chen et al., 2015)). Interestingly, we find that models which score best on standard sentence metrics do not always hallucinate less.

The second question we raise is: *What are the likely causes of hallucination?* While hallucination may occur due to a number of reasons, we believe the top factors include visual misclassification and over-reliance on language priors. The latter may result in memorizing which words "go together" regardless of image content, which may lead to poor generalization, once the test distribution is changed. We propose *image and language model consistency* scores to investigate this issue, and find that models which hallucinate more tend to make mistakes consistent with a language model.

Finally, we ask: *How well do the standard metrics capture hallucination?* It is a common practice to rely on automatic sentence metrics, e.g. CIDEr (Vedantam et al., 2015), to evaluate captioning performance during development, and few employ human evaluation to measure the final performance of their models. As we largely rely on these metrics, it is important to understand how well they capture the hallucination phenomenon. In Figure 1 we show how two sentences, from NBT with hallucination and from TopDown model (Anderson et al., 2018) – without, are scored by the standard metrics. As we see, hallucination is not always appropriately penalized. We find that by using additional ground truth data about the image in the form of object labels, our metric CHAIR allows us to catch discrepancies that the standard captioning metrics cannot fully capture. We then investigate ways to assess object hallucination risk with the standard metrics. Finally, we show that CHAIR is complementary to the standard metrics in terms of capturing human preference.

2 Caption Hallucination Assessment

We first introduce our image relevance metric, *CHAIR*, which assesses captions w.r.t. objects that are actually in an image. It is used as a main tool in our evaluation. Next we discuss the notions of *image and language model consistency*, which we use to reason about the causes of hallucination.

2.1 The CHAIR Metric

To measure object hallucination, we propose the *CHAIR (Caption Hallucination Assessment with Image Relevance)* metric, which calculates what proportion of words generated are actually in the image according to the ground truth sentences and object segmentations. This metric has two variants: per-instance, or what fraction of object instances are hallucinated (denoted as CHAIRi), and per-sentence, or what fraction of sentences include a hallucinated object (denoted as CHAIRs):

$$\text{CHAIR}_i = \frac{|\{\text{hallucinated objects}\}|}{|\{\text{all objects mentioned}\}|}$$

$$\text{CHAIR}_s = \frac{|\{\text{sentences with hallucinated object}\}|}{|\{\text{ all sentences}\}|}$$

For easier analysis, we restrict our study to the 80 MSCOCO objects which appear in the MSCOCO segmentation challenge. To determine whether a generated sentence contains hallucinated objects, we first tokenize each sentence and then singularize each word. We then use a list of synonyms for MSCOCO objects (based on the list from Lu et al. (2018)) to map words (e.g., "player") to MSCOCO objects (e.g., "person"). Additionally, for sentences which include two word compounds (e.g., "hot dog") we take care that other MSCOCO objects (in this case "dog") are not incorrectly assigned to the list of MSCOCO objects in the sentence. For each ground truth sentence, we determine a list of MSCOCO objects in the same way. The MSCOCO segmentation annotations are used by simply relying on the provided object labels.

We find that considering both sources of annotation is important. For example, MSCOCO contains an object "dining table" annotated with segmentation maps. However, humans refer to many different kinds of objects as "table" (e.g., "coffee table" or "side table"), though these objects are not annotated as they are not specifically "dining table". By using sentence annotations to scrape ground truth objects, we account for variation in how human annotators refer to different objects. Inversely, we find that frequently humans will not mention all objects in a scene. Qualitatively, we observe that both annotations are important to capture hallucination. Empirically, we verify that using only segmentation labels or only reference captions leads to higher hallucination (and practically incorrect) rates.

2.2 Image Consistency

We define a notion of *image consistency*, or how consistent errors from the captioning model are with a model which predicts objects based on an image alone. To measure image consistency for a particular generated word, we train an image model and record $P(w|I)$ or the probability of predicting the word given only the image. To score the image consistency of a caption we use the average of $P(w|I)$ for all MSCOCO objects, where higher values mean that errors are *more* consistent with the image model. Our image model is a multi-label classification model with labels corresponding to MSCOCO objects (labels determined the same way as is done for CHAIR) which shares the visual features with the caption models.

2.3 Language Consistency

We also introduce a notion of *language consistency*, i.e. how consistent errors from the captioning model are with a model which predicts words based only on previously generated words. We train an LSTM (Hochreiter and Schmidhuber, 1997) based language model which predicts a word w_t given previous words $w_{0:t-1}$ on MSCOCO data. We report language consistency as $1/R(w_t)$ where $R(w_t)$ is the rank of the predicted word in the language model. Again, for a caption we report average rank across all MSCOCO objects in the sentence and higher language consistency implies that errors are *more* consistent with the language model.

We illustrate image and language consistency in Figure 2, i.e. the hallucination error ("fork") is more consistent with the Language Model predictions than with the Image Model predictions. We use these consistency measures in Section 3.3 to help us investigate the causes of hallucination.

Image Model predictions:
bowl, broccoli, carrot, dining table

Language Model predictions for the last word:
fork, spoon, bowl

Generated caption: A plate of food with broccoli and a ***fork***.

Figure 2: Example of image and language consistency. The hallucination error ("fork") is more consistent with the Language Model.

3 Evaluation

In this section we present the findings of our study, where we aim to answer the questions posed in Section 1: *Which models are more prone to hallucination? What are the likely causes of hallucination? How well do the standard metrics capture hallucination?*

3.1 Baseline Captioning Models

We compare object hallucination across a wide range of models. We define two axes for comparison: model architecture and learning objective.

Model architecture. Regarding model architecture, we consider models both with and without attention mechanisms. In this work, we use "attention" to refer to any mechanism which learns to focus on different image regions, whether image regions be determined by a high level feature map, or by object proposals from a trained detector. All models are end-to-end trainable and use a recurrent neural network (LSTM (Hochreiter and Schmidhuber, 1997) in our case) to output text. For non-attention based methods we consider the **FC model** from Rennie et al. (2017) which incorporates visual information by initializing the LSTM hidden state with high level image features. We also consider **LRCN** (Donahue et al., 2015) which considers visual information at each time step, as opposed to just initializing the LSTM hidden state with extracted features.

For attention based models, we consider **Att2In** (Rennie et al., 2017), which is similar to the original attention based model proposed by (Xu et al., 2015), except the image feature is only input into the cell gate as this was shown to lead to better performance. We then consider the attention model proposed by (Anderson et al., 2018) which proposes a specific "top-down attention" LSTM as well as a "language" LSTM.

Generally attention mechanisms operate over high level convolutional layers. The attention mechanism from (Anderson et al., 2018) can be used on such feature maps, but Anderson et al. also consider feature maps corresponding to object proposals from a detection model. We consider both models, denoted as **TopDown** (feature map extracted from high level convolutional layer) and **TopDown-BB** (feature map extracted from object proposals from a detection model). Finally, we consider the recently proposed **Neural Baby Talk (NBT)** model (Lu et al., 2018) which explicitly uses object detections (as opposed to just bounding boxes) for sentence generation.

Learning objective. All of the above models are trained with the standard *cross entropy* (CE) loss as well as the *self-critical* (SC) loss proposed by Rennie et al. (2017) (with an exception of NBT, where only the CE version is included). The SC loss directly optimizes the CIDEr metric with a reinforcement learning technique. We additionally consider a model trained with a *GAN* loss (Shetty et al., 2017) (denoted **GAN**), which applies adversarial training to obtain more diverse and "human-like" captions, and their respective non-GAN baseline with the CE loss.

TopDown deconstruction. To better evaluate how each component of a model might influence hallucination, we "deconstruct" the TopDown model by gradually removing components until it is equivalent to the FC model. The intermediate networks are *NoAttention*, in which the attention mechanism is replaced by mean pooling, *NoConv* in which spatial feature maps are not input into the network (the model is provided with fully connected feature maps), *SingleLayer* in which only one LSTM is included in the model, and finally, instead of inputting visual features at each time step, visual features are used to initialize the LSTM embedding as is done in the FC model. By deconstructing the TopDown model in this way, we ensure that model design choices and hyperparameters do not confound results.

Implementation details. All the baseline models employ features extracted from the fourth layer of ResNet-101 (He et al., 2016), except for the GAN model which employs ResNet-152. Models without attention traditionally use fully connected layers as opposed to convolutional layers. However, as ResNet-101 does not have intermediate fully connected layers, it is standard to average pool convolutional activations and input these features into non-attention based description models. Note that this means the difference between the *NoAttention* and *NoConv* model is that the *NoAttention* model learns a visual embedding of spatial feature maps as opposed to relying on pre-pooled feature maps. All models except for TopDown-BB, NBT, and GAN are implemented in the same open source framework from Luo et al. (2018).[1]

Training/Test splits. We evaluate the captioning models on two MSCOCO splits. First, we consider the split from Karpathy et al. (Karpathy and Fei-Fei, 2015), specifically in that case the models are trained on the respective Karpathy Training set, tuned on Karpathy Validation set and the reported numbers are on the Karpathy Test set. We also consider the *Robust* split, introduced in (Lu et al., 2018), which provides a compositional split for MSCOCO. Specifically, it is ensured that the object pairs present in the training, validation and test captions do not overlap. In this case the captioning models are trained on the Robust Training set, tuned on the Robust Validation set and the reported numbers are on the Robust Test set.

3.2 Which Models Are More Prone To Hallucination?

We first present how well competitive models perform on our proposed CHAIR metric (Table 1). We report CHAIR at sentence-level and at instance-level (CHs and CHi in the table). In general, we see that models which perform better on standard evaluation metrics, perform better on CHAIR, though this is not always true. In particular, models which optimize for CIDEr frequently hallucinate more. Out of all generated captions on the Karpathy Test set, anywhere between 7.4% and 17.7% include a hallucinated object. When shifting to more difficult training scenarios in which new combinations of objects are seen at test time, hallucination consistently increases (Table 2).

Karpathy Test set. Table 1 presents object hallucination on the Karpathy Test set. All sentences are generated using beam search and a beam size of 5. We note a few important trends. First, models with attention tend to perform better on the CHAIR metric than models without attention. As we explore later, this is likely because they have

---

[1] `https://github.com/ruotianluo/self-critical.pytorch`

Model	Att.	Cross Entropy					Self Critical				
		S	M	C	CHs	CHi	S	M	C	CHs	CHi
LRCN*		17.0	23.9	90.8	17.7	12.6	16.9	23.5	93.0	17.7	12.9
FC*		17.9	24.9	95.8	15.4	10.9	18.4	25.0	103.9	14.4	10.1
Att2In*	✓	18.9	25.8	102.0	10.8	7.8	19.0	25.7	106.7	12.2	8.4
TopDown*	✓	19.9	26.7	107.6	8.4	6.0	20.4	27.0	117.2	13.6	8.8
TopDown-BB †	✓	20.4	27.1	113.7	8.3	5.9	21.4	27.7	120.6	10.4	6.9
NBT †	✓	19.4	26.2	105.1	7.4	5.4	-	-	-	-	
		Cross Entropy					GAN				
GAN ‡		18.7	25.7	100.4	10.6	7.6	16.6	22.7	79.3	8.2	6.5

Table 1: Hallucination analysis on the Karpathy Test set: Spice (S), CIDEr (C) and METEOR (M) scores across different image captioning models as well as CHAIRs (sentence level, CHs) and CHAIRi (instance level, CHi). All models are generated with beam search (beam size=5). * are trained/evaluated within the same implementation (Luo et al., 2018), † are trained/evaluated with implementation publicly released with corresponding papers, and ‡ sentences obtained directly from the author. For discussion see Section 3.2.

a better understanding of the image. In particular, methods that incorporate bounding box attention (as opposed to relying on coarse feature maps), consistently have lower hallucination as measured by our CHAIR metric. Note that the NBT model does not perform as well on standard captioning metrics as the TopDown-BB model but has lower hallucination. This is perhaps because bounding box proposals come from the MSCOCO detection task and are thus "in-domain" as opposed to the TopDown-BB model which relies on proposals learned from the Visual Genome (Krishna et al., 2017) dataset. Second, frequently training models with the self-critical loss actually increases the amount of hallucination. One hypothesis is that CIDEr does not penalize object hallucination sufficiently, leading to both increased CIDEr and increased hallucination. Finally, the LRCN model has a higher hallucination rate than the FC model, indicating that inputting the visual features only at the first step, instead of at every step, leads to more image relevant captions.

We also consider a GAN based model (Shetty et al., 2017) in our analysis. We include a baseline model (trained with CE) as well as a model trained with the GAN loss.[2] Unlike other models, the GAN model uses a stronger visual network (ResNet-152) which could explain the lower hallucination rate for both the baseline and the GAN model. Interestingly, when comparing the baseline and the GAN model (both trained with ResNet-152), standard metrics decrease substantially, even though human evaluations from (Shetty et al., 2017) demonstrate that sentences are of comparable quality. On the other hand, hallucination decreases, implying that the GAN loss actually helps decrease hallucination. Unlike the self critical loss, the GAN loss encourages sentences to be human-like as opposed to optimizing a metric. Human-like sentences are not likely to hallucinate objects, and a hallucinated object is likely a strong signal to the discriminator that a sentence is generated, and is not from a human.

	Att	S	M	C	CHs	CHi
FC*		15.5	22.7	76.2	21.3	15.3
Att2In*	✓	16.9	24.0	85.8	14.1	10.1
TopDown*	✓	17.7	24.7	89.8	11.3	7.9
NBT †	✓	18.1	24.8	94.5	6.8	4.6

Table 2: Hallucination Analysis on the Robust Test set: Spice (S), CIDEr (C) and METEOR (M) scores across different image captioning models as well as CHAIRs (sentence level, CHs) and CHAIRi (instance level, CHi). * are trained/evaluated within the same implementation (Luo et al., 2018), † are trained/evaluated with implementation publicly released with corresponding papers. All models trained with cross-entropy loss. See Section 3.2.

We also assess the effect of beam size on CHAIR. We find that generally beam search decreases hallucination. We use beam size of 5, and for all models trained with cross entropy, it outperforms lower beam sizes on CHAIR. However, when training models with the self-critical loss, beam size sometimes leads to worse performance on CHAIR. For example, on the Att2In model trained with SC loss, a beam size of 5 leads to 12.8 on CHAIRs and 8.7 on CHAIRi, while a beam size of 1 leads to 10.8 on CHAIRs and 8.1 on CHAIRi.

Robust Test set. Next we review the hallucination behavior on the Robust Test set (Table 2). For almost all models the hallucination increases on the Robust split (e.g. for TopDown from 8.4% to 11.4% of sentences), indicating that the issue of

---

[2] Sentences were procured directly from the authors.

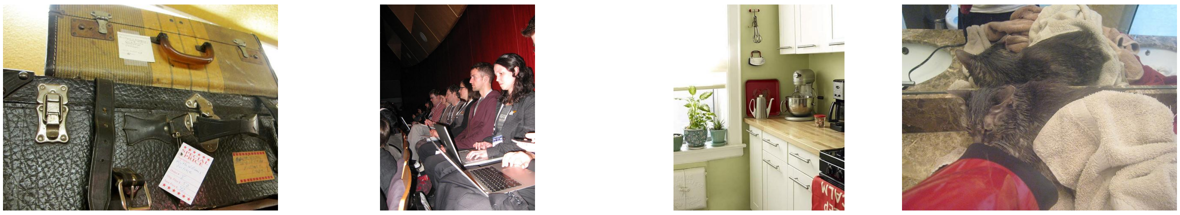

TopDown: A pile of luggage sitting on top of a ***table***.
NBT: Several pieces of luggage sitting on a ***table***.

TopDown: A group of people sitting around a ***table*** with laptops.
NBT: A group of people sitting around a ***table*** with laptop.

TopDown: A kitchen with a stove and a ***sink***.
NBT: A kitchen with a stove and a ***sink***.

TopDown: A couple of cats laying on top of a ***bed***.
NBT: A couple of cats laying on top of a ***bed***.

TopDown: A cat sitting on top of a ***laptop computer***.
NBT: A cat sitting on a table next to a ***computer***.

TopDown: A brown dog sitting on top of a ***chair***.
NBT: A brown and white dog sitting under an ***umbrella***.

TopDown: Aa man and a woman are playing with a ***frisbee***.
NBT: A man riding a skateboard down a street.

TopDown: A man standing on a beach holding a ***surfboard***.
NBT: A man standing on top of a sandy beach.

Figure 3: Examples of object hallucination from two state-of-the-art captioning models, TopDown and NBT, see Section 3.2.

hallucination is more critical in scenarios where test examples can not be assumed to have the same distribution as train examples. We again note that attention is helpful for decreasing hallucination. We note that the NBT model actually has lower hallucination scores on the robust split. This is in part because when generating sentences we use the detector outputs provided by Lu et al. (2018). Separate detectors on the Karpathy test and robust split are not available and the detector has access to images in the robust split during training. Consequently, the comparison between NBT and other models is not completely fair, but we include the number for completeness.

In addition to the Robust Test set, we also consider a set of MSCOCO in which certain objects are held out, which we call the *Novel Object split* (Hendricks et al., 2016). We train on the training set outlined in (Hendricks et al., 2016) and test on the Karpathy test split, which includes objects unseen during training. Similarly to the Robust Test set, we see hallucination increase substantially on this split. For example, for the TopDown model hallucination increases from 8.4% to 12.1% for CHAIRs and 6.0% to 9.1% for CHAIRi.

We find no obvious correlation between the average length of the generated captions and the hallucination rate. Moreover, vocabulary size does not correlate with hallucination either, i.e. models with *more diverse* descriptions may actually *hallucinate less*. We notice that hallucinated objects tend to be mentioned towards *the end of the sentence* (on average at position 6, with average sentence length 9), suggesting that some of the preceding words may have triggered hallucination. We investigate this below.

Which objects are hallucinated and in what context? Here we analyze which MSCOCO objects tend to be hallucinated more often and what are the common preceding words and image context. Across all models the super-category *Furniture* is hallucinated most often, accounting for $20-50\%$ of all hallucinated objects. Other common super-categories are *Outdoor objects*, *Sports* and *Kitchenware*. On the Robust Test set, *Animals* are often hallucinated. The *dining table* is the most frequently hallucinated object across all models (with an exception of GAN, where *person* is the most hallucinated object). We find that often words like “sitting” and “top” precede the “dining table” hallucination, implying the two common scenarios: a person “sitting at the table” and an object “sitting on top of the table” (Figure 3, row 1, examples 1, 2). Similar observations can be made for other objects, e.g. word “kitchen” often precedes “sink” hallucination (Figure 3, row 1, example 3) and “laying” precedes “bed” (Figure 3, row 1, example 4). At the same time, if we look at which objects are actually present in the image (based on MSCOCO object annotations), we can similarly identify that presence of a “cat” co-occurs with hallucinating a “laptop” (Figure 3, row 2, example 1), a “dog” – with a “chair” (Figure 3, row 2, example 2) etc. In most cases we observe that the hallucinated objects appear in the relevant scenes (e.g. “surfboard” on a beach), but

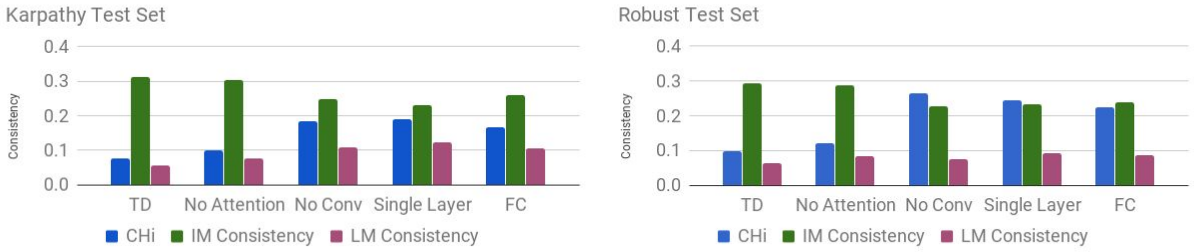

Figure 4: Image and Language model consistency (IM, LM) and CHAIRi (instance-level, CHi) on deconstructed TopDown models. Images with less hallucination tend to make errors consistent with the image model, whereas models with more hallucination tend to make errors consistent with the language model, see Section 3.3.

there are cases where objects are hallucinated out of context (e.g. "bed" in the bathroom, Figure 3, row 1, example 4).

3.3 What Are The Likely Causes Of Hallucination?

In this section we investigate the likely causes of object hallucination. We have earlier described how we deconstruct the TopDown model to enable a controlled experimental setup. We rely on the deconstructed TopDown models to analyze the impact of model components on hallucination.

First, we summarize the hallucination analysis on the deconstructed TopDown models (Table 3). Interestingly, the *NoAttention* model does not do substantially worse than the full model (w.r.t. sentence metrics and CHAIR). However, removing Conv input (*NoConv* model) and relying only on FC features, decreases the performance dramatically. This suggests that much of the gain in attention based models is primarily due to *access to feature maps with spatial locality*, not the actual attention mechanism. Also, similar to LRCN vs. FC in Table 1, initializing the LSTM hidden state with image features, as opposed to inputting image features at each time step, leads to lower hallucination (*Single Layer* vs. *FC*). This is somewhat surprising, as a model which has access to image information at each time step should be less likely to "forget" image content and hallucinate objects. However, it is possible that models which include image inputs at each time step with no access to spatial features overfit to the visual features.

Now we investigate what causes hallucination using the deconstructed TopDown models and the *image consistency* and *language consistency* scores, introduced in Sections 2.2 and 2.3 which capture how consistent the hallucinations errors are with image- / language-only models.

Karpathy Split	METEOR	CIDEr	SPICE	CHs	CHi
TD	26.10	103.40	19.50	10.80	7.40
No Attention	25.60	99.70	18.80	14.20	9.40
No Conv	22.90	81.30	15.70	25.70	17.70
Single Layer	22.70	80.20	15.50	25.60	18.00
FC	23.30	85.10	16.40	23.60	15.70

Table 3: Hallucination analysis on deconstructed TopDown models with sentence metrics, CHAIRs (sentence level, CHs) and CHAIRi (instance level, CHi). See Section 3.3.

Figure 4 shows the CHAIR metric, image consistency and language consistency for the deconstructed TopDown models on the Karpathy Test set (left) and the Robust Test set (right). We note that models with *less* hallucination tend to make errors consistent with the image model, whereas models with *more* hallucination tend to make errors consistent with the language model. This implies that models with less hallucination are better at integrating knowledge from an image into the sentence generation process. When looking at the Robust Test set, Figure 4 (right), which is more challenging, as we have shown earlier, we see that image consistency *decreases* when comparing to the same models on the Karpathy split, whereas language consistency is similar across all models trained on the Robust split. This is perhaps because the Robust split contains novel compositions of objects at test time, and all of the models are heavily biased by language.

Finally, we measure image and language consistency during training for the FC model and note that at the beginning of training errors are more consistent with the language model, whereas towards the end of training, errors are more consistent with the image model. This suggests that models first learn to produce fluent language before learning to incorporate visual information.

TD: A cat is sitting on a bed in a room.
S: 12.1 M: 23.8 C: 69.7
TD Restrict: A bed with a blanket and a pillow on it.
S: 23.5 M: 25.4 C: 52.5

TD: A cat laying on the ground with a frisbee.
S: 8.0 M: 13.1 C: 37.0
TD Restrict: A black and white animal laying on the ground.
S: 7.7 M: 15.9 C: 17.4

Figure 5: Examples of how TopDown (TD) sentences change when we enforce that objects cannot be hallucinated: SPICE (S), Meteor (M), CIDEr (C), see Section 3.4.

3.4 How Well Do The Standard Metrics Capture Hallucination?

In this section we analyze how well SPICE (Anderson et al., 2016), METEOR (Banerjee and Lavie, 2005), and CIDEr (Vedantam et al., 2015) capture hallucination. All three metrics do penalize sentences for mentioning incorrect words, either via an F score (METEOR and SPICE) or cosine distance (CIDEr). However, if a caption mentions enough words correctly, it can have a high METEOR, SPICE, or CIDEr score while still hallucinating specific objects.

Our first analysis tool is the TD-Restrict model. This is a modification of the TopDown model, where we enforce that MSCOCO objects which are not present in an image are *not generated* in the caption. We determine which words refer to objects absent in an image following our approach in Section 2.1. We then set the log probability for such words to a very low value. We generate sentences with the TopDown and TD-Restrict model with beam search of size 1, meaning all words produced by both models are the same, until the TopDown model produces a hallucinated word.

We compare which scores are assigned to such captions in Figure 5. TD-Restrict generates captions that do not contain hallucinated objects, while TD hallucinates a “cat” in both cases. In Figure 5 (left) we see that CIDEr scores the more correct caption much lower. In Figure 5 (right), the TopDown model incorrectly calls the animal a “cat.” Interestingly, it then correctly identifies the “frisbee,” which the TD-Restrict model fails to mention, leading to lower SPICE and CIDEr.

	CIDEr	METEOR	SPICE
FC	0.197	0.198	0.266
Att2In	0.177	0.178	0.246
TopDown	0.135	0.140	0.172

Table 4: Pearson correlation coefficients between 1-CHs and CIDEr, METEOR, and SPICE scores, see Section 3.4.

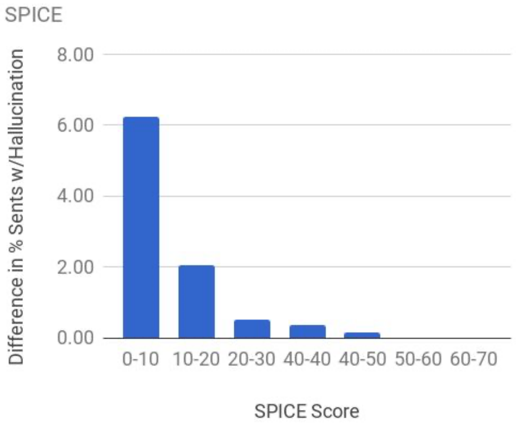

Figure 6: Difference in percentage of sentences with *no* hallucination for TopDown and FC models when SPICE scores fall into specific ranges. For sentences with low SPICE scores, the hallucination is generally larger for the FC model, even though the SPICE scores are similar, see Section 3.4.

In Table 4 we compute Pearson correlation coefficient between individual sentence scores and the *absence* of hallucination, i.e. $1-$CHAIRs; we find that SPICE consistently correlates higher with $1-$CHAIRs. E.g., for the FC model the correlation for SPICE is 0.27, while for METEOR and CIDEr – around 0.2.

We further analyze the metrics in terms of their predictiveness of hallucination risk. Predictiveness means that a certain score should imply a certain percentage of hallucination. Here we show the results for SPICE and the captioning models FC and TopDown. For each model and a score interval (e.g. $10-20$) we compute the percentage of captions *without* hallucination ($1-$CHAIRs). We plot the difference between the percentages from both models (TopDown - FC) in Figure 6. Comparing the models, we note that even when scores are similar (e.g., all sentences with SPICE score in the range of $10-20$), the TopDown model has fewer sentences with hallucinated objects. We see similar trends across other metrics. Consequently, object hallucination can *not* be always predicted based on the traditional sentence metrics.

Is CHAIR complementary to standard metrics? In order to measure usefulness of our proposed metrics, we have conducted the following

	Metric	Metric +(1-CHs)	Metric +(1-CHi)
METEOR	0.269	0.299	0.304
CIDEr	0.282	0.321	0.322
SPICE	0.248	0.277	0.281

Table 5: Pearson correlation coefficients between individual/combined metrics and human scores. See Section 3.4.

human evaluation (via the Amazon Mechanical Turk). We have randomly selected 500 test images and respective captions from 5 models: non-GAN baseline, GAN, NBT, TopDown and TopDown - Self Critical. The AMT workers were asked to score the presented captions w.r.t. the given image based on their preference. They could score each caption from 5 (very good) to 1 (very bad). We did not use ranking, i.e. different captions could get the same score; each image was scored by three annotators, and the average score is used as the final human score. For each image we consider the 5 captions from all models and their corresponding sentence scores (METEOR, CIDEr, SPICE). We then compute Pearson correlation between the human scores and sentence scores; we also consider a simple combination of sentence metrics and 1-CHAIRs or 1-CHAIRi by summation. The final correlation is computed by averaging across all 500 images. The results are presented in Table 5. Our findings indicate that a simple combination of CHAIRs or CHAIRi with the sentence metrics leads to an increased correlation with the human scores, showing the usefulness and complementarity of our proposed metrics.

Does hallucination impact generation of other words? Hallucinating objects impacts sentence quality not only because an object is predicted incorrectly, but also because the hallucinated word impacts generation of other words in the sentence. Comparing the sentences generated by TopDown and TD-Restrict allows us to analyze this phenomenon. We find that after the hallucinated word is generated, the following words in the sentence are different 47.3% of the time. This implies that hallucination impacts sentence quality beyond simply naming an incorrect object. We observe that one hallucination may lead to another, e.g. hallucinating a “cat” leading to hallucinating a “chair”, hallucinating a “dog” – to a “frisbee”.

4 Discussion

In this work we closely analyze hallucination in object captioning models. Our work is similar to other works which attempt to characterize flaws of different evaluation metrics (Kilickaya et al., 2016), though we focus specifically on hallucination. Likewise, our work is related to other work which aims to build better evaluation tools ((Vedantam et al., 2015), (Anderson et al., 2016), (Cui et al., 2018)). However, we focus on carefully quantifying and characterizing one important type of error: object hallucination.

A significant number of objects are hallucinated in current captioning models (between 5.5% and 13.1% of MSCOCO objects). Furthermore, hallucination does not always agree with the output of standard captioning metrics. For instance, the popular self critical loss increases CIDEr score, but also the amount of hallucination. Additionally, we find that given two sentences with similar CIDEr, SPICE, or METEOR scores from two different models, the number of hallucinated objects might be quite different. This is especially apparent when standard metrics assign a low score to a generated sentence. Thus, for challenging caption tasks on which standard metrics are currently poor (e.g., the LSMDC dataset (Rohrbach et al., 2017b)), the CHAIR metric might be helpful to tease apart the most favorable model. Our results indicate that CHAIR complements the standard sentence metrics in capturing human preference.

Additionally, attention lowers hallucination, but it appears that much of the gain from attention models is due to access to the underlying convolutional features as opposed the attention mechanism itself. Furthermore, we see that models with stronger *image consistency* frequently hallucinate fewer objects, suggesting that strong visual processing is important for avoiding hallucination.

Based on our results, we argue that the design and training of captioning models should be guided not only by cross-entropy loss or standard sentence metrics, but also by image relevance. Our CHAIR metric gives a way to evaluate the phenomenon of hallucination, but other image relevance metrics e.g. those that incorporate missed salient objects, should also be investigated. We believe that incorporating visual information in the form of ground truth objects in a scene (as opposed to only reference captions) helps us better understand the performance of captioning models.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and vqa. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge Belongie. 2018. Learning to evaluate image captioning. In *CVPR*.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, and Trevor Darrell. 2016. Deep compositional captioning: Describing novel object categories without paired training data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10.

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

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. 2016. Re-evaluating automatic metrics for image captioning. In *European Chapter of the Association for Computational Linguistics*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2018. Neural baby talk. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ruotian Luo, Brian Price, Scott Cohen, and Gregory Shakhnarovich. 2018. Discriminability objective for training descriptive captions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Haley MacLeod, Cynthia L. Bennett, Meredith Ringel Morris, and Edward Cutrell. 2017. Understanding blind peoples experiences with computer-generated captions of social media images. In *Proceedings of the 2017 SIGCHI Conference on Human Factors in Computing Systems*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Anna Rohrbach, Makarand Tapaswi, Atousa Torabi, Tegan Maharaj, Marcus Rohrbach, Sanja Fidler Christopher Pal, and Bernt Schiele. 2017a. The Joint Video and Language Understanding Workshop: MovieQA and The Large Scale Movie Description Challenge (LSMDC). `https://sites.google.com/site/describingmovies/lsmdc-2017`.

Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele. 2017b. Movie description. *International Journal of Computer Vision*, 123(1):94–120.

Rakshith Shetty, Marcus Rohrbach, Lisa Anne Hendricks, Mario Fritz, and Bernt Schiele. 2017. Speaking the same language: Matching machine to human captions by adversarial training. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pages 4566–4575.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Abstractive Text-Image Summarization Using Multi-Modal Attentional Hierarchical RNN

Jingqiang Chen[1], **Hai Zhuge**[1,2,3,4]
[1]Nanjing University of Posts and Telecommunications
[2]Aston University; [3]Guangzhou University
[4]Key Laboratory of Intelligent Information Processing, ICT,
University of Chinese Academy of Sciences, Chinese Academy of Sciences
cjq@njupt.edu.cn, haizhuge@gmail.com

Abstract

Rapid growth of multi-modal documents on the Internet makes multi-modal summarization research necessary. Most previous research summarizes texts or images separately. Recent neural summarization research shows the strength of the Encoder-Decoder model in text summarization. This paper proposes an abstractive text-image summarization model using the attentional hierarchical Encoder-Decoder model to summarize a text document and its accompanying images simultaneously, and then to align the sentences and images in summaries. A multi-modal attentional mechanism is proposed to attend original sentences, images, and captions when decoding. The DailyMail dataset is extended by collecting images and captions from the Web. Experiments show our model outperforms the neural abstractive and extractive text summarization methods that do not consider images. In addition, our model can generate informative summaries of images.

1 Introduction

Summarizing multi-modal documents to get multi-modal summaries is becoming an urgent need with rapid growth of multi-modal documents on the Internet. Text-Image summarization is to summarize a document with text and images to generate a summary with text and images. The summarization approach is different from pure text summarization. It is also different from image summarization which summarizes an image set to get a subset of images.

An image worths thousands of words (Rossiter, et al., 2012). Image plays an important role in information transmission. Incorporating images into text to generate text-image

Former Manchester United striker Alan Smith denies Liverpool fans attacked his ambulance after Anfield horror injury

Alan Smith has rubbished claims that Liverpool fans attacked the ambulance he was travelling to hospital in following the aftermath of his horror injury during Manchester United's FA Cup 1-0 defeat to their fierce rivals nine years ago.
Smith suffered a broken leg and a dislocated ankle while attempting to block a John Arne Riise free-kick during the fifth round clash at Anfield in February 2006.
At the time of the incident, reports circulated that Liverpool fans tried to disrupt Smith's journey to hospital by throwing bottles, beer glasses and stones at the ambulance as well as rocking the vehicle.

Alan Smith has denied claims that Liverpool fans attacked the ambulance he was travelling to hospital in during Manchester United's FA Cup fifth round exit nine years ago

The horror incident occurred after Smith (second right) blocked John Arne Riise's free-kick

But Smith, who made his comeback for United seven months later, insists that was not the case.
'It didn't happen – fans were still in the ground,' he told FourFourTwo exclusively in the May edition of their magazine.
'I went back to Liverpool a few years later with Newcastle and had a great reception.
'I had loads of mail from fans after the injury, including a lot from Liverpool. And Liverpool's medical staff were great. They were worried that because there was no blood flowing that I could have had a club foot.
'It wasn't Riise's fault and he came to see me. The dislocated ankle was worse than the leg break because I snapped ligaments and there were complications.
'I knew I was never going to be the same player. I've appreciated every game I've played since that injury, I know how close I was to being finished.'

Smith suffered a broken leg and a dislocated ankle as a result of the incident at Anfield

Smith was carried off on a stretcher before he was taken to hospital via an ambulance

Figure 1: An example of multi-modal news taken from the DailyMail corpora.

Liverpool beat Manchester United 1-0 in the FA Cup fifth round in 2006.
Alan Smith suffered a broken leg and a dislocated ankle during the game.

Reports claimed Liverpool fans attacked the ambulance with Smith in.

Figure 2: The manually generated text-image summary.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4046–4056
Brussels, Belgium, October 31 - November 4, 2018.

summaries can help people better understand, memorize, and express information. Most of recent research focuses on pure text summarization, or image summarization. Little has been done on text-image summarization.

Figure 1 and Figure 2 show an example of text-image summarization. Figure 1 is the original multi-modal news with text and images. The news has 17 sentences (with 322 words) and 4 images each of which has a caption. Figure 2 is the manually generated multi-modal summary. In the summary, the news is distilled to 3 sentences (with 36 words) and 2 images, and each summary sentence is aligned with an image.

To generate such a text-image summary, the following problems should be considered: How to generate the text part? How to measure the importance of images, and extract important images to form the image summary? How to align sentences with images?

In this paper, we propose a neural text-image summarization model based on the attentional hierarchical Encoder-Decoder model to solve the above problems. The attentional Encoder-Decoder model has been successfully used in sequence-to-sequence applications such as machine translation (Luong et al., 2015), text summarization (Cheng and Lapata, 2016; Tan et al., 2017), image captioning (Liu et al., 2017a), and machine reading comprehension (Cui et al., 2016).

At the encoding stage, we use the hierarchical bi-directional RNN to encode the sentences and the text document, use the RNN and the CNN to encode the image set. In the decoding stage, we combine text encoding and image encoding as the initial state, and use the attentional hierarchical decoder which attends original sentences, images and captions to generate the text summary. Each generated sentence is aligned with a sentence, an image, or a caption in the original document. Based on the alignment scores, images are selected and aligned with the generated sentences. In the inference stage, we adopt the multi-modal beam search algorithm which scores beams based on bigram overlaps of the generated sentences and the attended captions.

The main contributions are as follows:

1) We propose the text-image summarization task, and extend the standard DailyMail corpora by collecting images and captions of each news from the Web for the task.
2) We propose an RNN model to encode the ordered image set of the multi-model document as one of the initial states (the other is the text encoding) of the decoder.
3) We propose three multi-modal attentional mechanisms which attend the text and the images simultaneously when decoding.
4) Experiments show that attending images when decoding can improve text summarization, and that our model can generate informative image summaries.

2 Related Work

Recent research on text summarization focuses on neural methods. Attentional Encoder-Decoder model is first proposed in (Bahdanau et al., 2014) and (Luond et al., 2015) to align the original text and the translated text in machine translation. The attention model is applied to sentence summarization by considering the neural language model and the attention model when generating next words (Rush et al., 2015). A selective Encoder-Decoder model that uses a selective gate network to control information from the encoder to the decoder for sentence summarization is proposed (Zhou et al., 2017).

A neural document summarization model by extracting sentences and words is proposed (Cheng and Lapata, 2016). They use a CNN model to encode sentences, and then use a RNN model to encode documents. The model extracts sentences by computing the probability of sentences belonging to the summary based on an RNN model. The model extracts words from the original document based on an attentional decoder. An RNN-based extractive summarization named SummaRuNNer, treating summarization as a sentence classification problem is proposed (Nallapati et al., 2016). A logistic classifier is then applied using features computed based on the RNN model. A hierarchical Encoder-Decoder model, conserving the hierarchical structure of documents is proposed (Li et al., 2015). A graph-based attentional Encoder-Decoder model using a PageRank algorithm to compute the attention is proposed (Tan et al., 2017).

Image captioning generates a caption for an image. Text-image summarization is similar to image captioning in that both utilize image information to generate text. Images are encoded with CNN models such as VGGNet (Simonyan and Zisserman, 2014), AlexNet (Krizhevsky et al., 2012) and GoogleNet (Szegedy et al., 2014) by extracting the last full-connected layers. An attentional model is used in image captioning by splitting an image into multiple parts which is attended in the decoding process (Xu et al., 2015). Image tags was used as additional information, and semantic attention model which attends image tags when decoding was proposed（You et al., 2016). The attention-based alignment of image parts and text is studied（Liu et al., 2017a), and the results show that the alignments is in high accordance with manual alignments. An image to an ordered recognized object set is encoded, and the attentional decoder is applied to generate captions（Liu et al., 2017b).

Multi-modal summarization summarizes text, images, videos, and etc. It is an important branch of automatic summarization. Traditional multi-modal summarization inputs multi-modal documents or pure text documents, and outputs multi-modal documents (Wu, 2011; Greenbacker, 2011; Yan, 2012; Agrawal, 2011; Zhu, 2007; UzZaman, 2011). For example, Yan et al., (2012) generate multi-modal timeline summaries for news sets by constructing a bi-graph between text and images, and apply a heterogeneous reinforcement ranking algorithm. Strategies to summarizing texts with images and the notion of summarization of things are proposed in (Zhuge, 2016). The deep learning related work (Wang et al. 2016) treats text summarization as a sentence recommendation task and applies matrix factorization algorithm. They first retrieve images from Yahoo!, use the CNN to extract image features as the additional information of sentences, use Rouge maximization as the training object function which are trained with SGD. In test time, sentences are extracted based on the model and images are retrieved from the Search Engine.

3 Method

Figure 3 shows the framework, a multi-modal attentional hierarchical encoder-decoder model. The hierarchical encoder-decoder is proposed in (Li et al., 2015) and extended by (Tan et al. , 2017) for document summarization through bringing in the graph-based attentional model.

Our model consists of three parts: a hierarchical RNN to encode the original sentences and the captions, a CNN+RNN encoder to encode the image set, and a multi-modal attentional hierarchical RNN decoder.

The input of our model is a multi-modal document *MD* = {*D*, *PicSet*}, where *D* is the main text of the multi-modal document and *PicSet* is the image-caption set ordered by the occurring order of images in the document.

3.1 Main Text Encoder

The main text *D* consists of sentences, each of which consists of words. Let $D=[s_1, s_2, \ldots, s_{|d|}]$, and $s_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,|s_i|}]$ where $x_{i,j}$ is the word embedding of the j^{th} word in the s_i. We use word2vec (Mikolov et al., 2013) to create word embeddings. GRU is used as the RNN cell (Cho et al., 2014).

We use a hierarchical RNN encoder to encode the main text *D* to vector representation. The sentence encoder is adopted to encode sentences to vector representations. An <eos> token is appended to the end of each sentence. A bi-directional RNN is used as the sentence encoder:

$$\vec{h}_{i,j} = \overrightarrow{GRU}^{s}(\vec{h}_{i,j-1}, x_{i,j}) \quad (1)$$

$$\overleftarrow{h}_{i,j} = \overleftarrow{GRU}^{s}(\overleftarrow{h}_{i,j}, x_{i,j+1}) \quad (2)$$

$$enc^{sent}_{i} = [\vec{h}_{i,1}, \overleftarrow{h}_{i,-1}] \quad (3)$$

where enc^{sent}_{i} denotes the vector representation of s_i. It is the concatenation of $\vec{h}_{i,1}$ and $\overleftarrow{h}_{i,-1}$.

We use enc^{sent}_{i} as inputs to the document encoder to encode the main text to vector representations. A bi-directional RNN is adopted as the document encoder:

$$\vec{h}_{i} = \overrightarrow{GRU}^{d}(\vec{h}_{i-1}, enc^{sent}_{i}) \quad (4)$$

$$\overleftarrow{h}_{i} = \overleftarrow{GRU}^{d}(\overleftarrow{h}_{i+1}, enc^{sent}_{i}) \quad (5)$$

$$h_i = [\vec{h}_i, \overleftarrow{h}_i] \quad (6)$$

$$enc^{doc}_{d} = [\vec{h}_1, \overleftarrow{h}_{-1}] \quad (7)$$

where enc^{doc} denotes the vector representation of the *D*, and h_i is the concatenated hidden state of s_i.

3.2 CaptionSet and ImageSet Encoder

The ordered image-caption set *PicSet* consists of an ordered image set and an ordered caption set

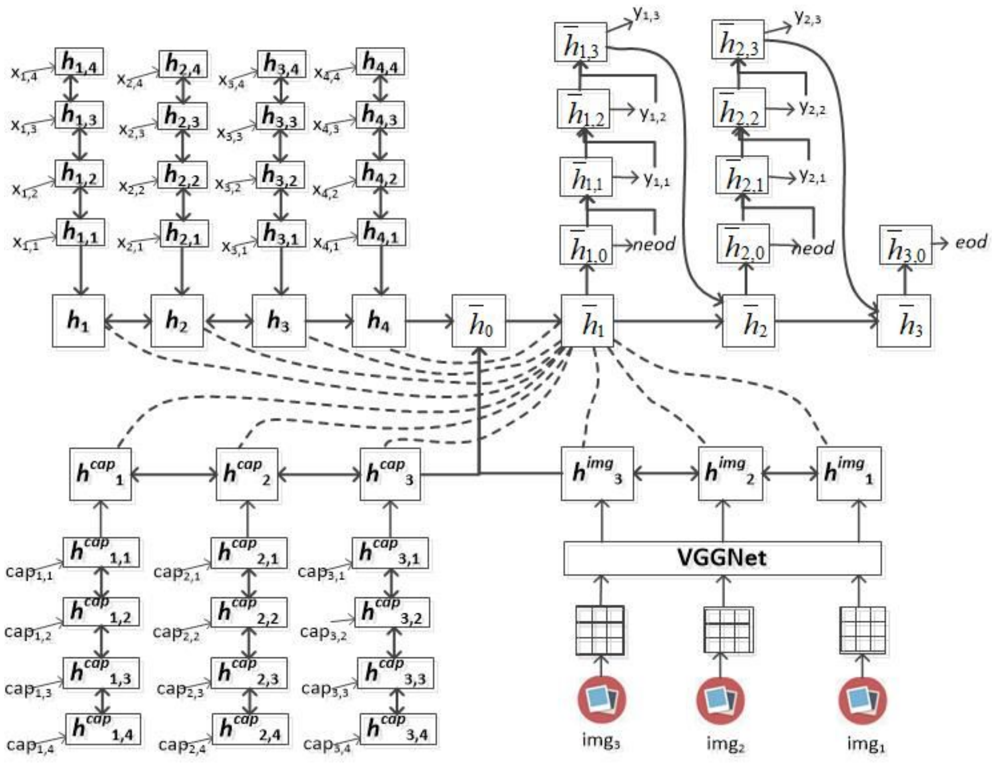

Figure 3: The framework of our neural text-image summarization model.

which are ordered by the occurring order in the multi-modal document. The image occurring order makes sense because images are often put near the most related sentences, and the sentences have strict order in the document.

We treat the ordered caption set as a document, and apply the sentence encoder and the document encoder to the caption document. Then, we get the hidden state h^{cap}_i and the vector representation enc^{cap} of the caption document.

We use the CNN model to extract the vector representation of each image, and then use the RNN model to encode the ordered image set to vector representation. The CNN model we adopted is 19-layer VGGNet (Simonyan and Zisserman, 2014). We drop the last dropout layer and keep the last full-connected layer as the image's vector representation, the dimension of which is 4096.

We then use a bi-directional RNN model to encode the ordered image set and the image features are used as inputs of the RNN model.

$$\overrightarrow{h}^{img}_i = \overrightarrow{GRU}^{img}(\overrightarrow{h}^{img}_{i-1}, img^{fea}_i) \quad (8)$$

$$\overleftarrow{h}^{img}_i = \overleftarrow{GRU}^{img}(\overleftarrow{h}^{img}_{i+1}, img^{fea}_i) \quad (9)$$

$$h^{img}_i = [\overrightarrow{h}^{img}_i, \overleftarrow{h}^{img}_i] \quad (10)$$

$$enc^{img} = [\overrightarrow{h}^{img}_1, \overleftarrow{h}^{img}_{-1}] \quad (11)$$

where img^{fea}_i is the vector representation of img_i, enc^{img} is the vector representation of the image set, and h^{img}_i is the hidden state of img_i when encoding the image set.

To our best knowledge, we are the first to adopt the RNN model to encode the image set.

3.3 Decoder

In the decoding state, we adopt the hierarchical RNN decoder to generate text summaries.

$$\begin{aligned}\bar{h}_0 = \tanh(&W^{dec\_doc} \times enc^{doc} \\ &+V^{dec\_img} \times enc^{img} \\ &+V^{dec\_cap} \times enc^{cap})\end{aligned} \quad (12)$$

$$\tilde{h}_i = GRU^{dec\_sent1}(\bar{h}_{i-1}, \bar{h}_{i-1,-1}) \quad (13)$$

$$\bar{h}_i = GRU^{dec\_sent2}(\tilde{h}_i, c_i) \quad (14)$$

$$\bar{h}_{i,j} = GRU^{dec\_word}(\bar{h}_{i,j-1}, y_{i,j-1}) \quad (15)$$

$$y_{i,j} \sim soft\max(W^{soft\max}\bar{h}_{i,j} + b) \quad (16)$$

Equation (12) to (16) are the equations of the hierarchical decoder which consists of a sentence decoder and a word decoder.

Equation (12) computes the initial state $\bar{h}_0$ for the sentence decoder by combining the decoding of the main text information and the decoding of the image information of the multi-modal document. To represent image information, we can use both of the image set decoding and the caption set decoding, or only use one of them, depending on the multi-modal attention mechanism introduced in the next subsection.

The sentence decoder uses a two-level hidden output model (Luong et al., 2015) to generate the representation of the next sentence through equation (13) and equation (14). The two-level hidden output model consistently improves the summarization performance on different datasets (Chen et al., 2016). In equation (14), the two-level model computes $\bar{h}_i$ by capturing a direct interaction between $\tilde{h}_i$ and c_i. $\tilde{h}_i$ is computed by equation (13) using the preceding sentence decoder hidden state $\bar{h}_{i-1}$ and the word decoder hidden state $\bar{h}_{i-1,-1}$ which is the last hidden state of the preceding word decoder. And c_i is the context of the sentence decoder computed based on the multi-modal attention model.

The word decoder uses the sentence representation generated by the sentence decoder as the initial state, and use the <sos> (start of sentence) token as the initial input. Equation (15) and equation (16) generate the next hidden state and the next word. The output of the word decoder in the first step is a switch sign which is either <neod> token or <eod> token. The token <neod> means “not end of document”, and the token <eod> means “end of document”. If the first output is <eod>, the whole decoding process is finished. If the first output is <neod>, the token is used as the next input of the word decoder. The word decoding process is finished when it generates the <eos> token. The last hidden state of the word decoder is treated as the vector representation of the generated sentence and is used as next input of the sentence decoder.

3.4 Multi-Modal Attention

We propose three multi-modal attention mechanisms to compute the sentence decoding context c_i.

Traditional attention mechanisms for text summarization computes the importance score of the sentence s_j in the original document based on the relationship between the decoding hidden state $\tilde{h}_i$ and the original sentence encoding hidden state h_j. We call the traditional attention model as **Text Attention (*attT* for short)**, which is computed by equation (17), (18) and (19):

$$att^T(\tilde{h}_i, h_j) = v^T \tanh(W^T \tilde{h}_i + U^T h_j) \quad (17)$$

$$\alpha^T(\tilde{h}_i, h_j) = \frac{\exp(att^T(\tilde{h}_i, h_j))}{\sum_{j=1}^{|D|} \exp(att^T(\tilde{h}_i, h_j))} \quad (18)$$

$$c^T(\tilde{h}_i) = \sum_{j=1}^{|D|} \alpha^T(\tilde{h}_i, h_j) h_j \quad (19)$$

where $att^T(\tilde{h}_i, h_j)$ is the attention (Banahama et al., 2014), $\alpha^T(\tilde{h}_i, h_j)$ is the normalized attention, and $c^T(\tilde{h}_i)$ is the context.

The problem is that the multi-modal document has images and captions besides the main text. Therefore, we propose three multi-modal attention mechanisms which take images and captions into consideration.

Text-Caption Attention (*attTC* for short). This attention model uses captions to represent the image information. *attTC* computes the attention score of the caption cap_j based on the relationship between the caption encoding hidden state $h^{cap}{}_i$ and the decoding hidden state $\tilde{h}_j$.

$$\alpha^{TC}(\tilde{h}_i, h_j) = \frac{\exp(att^{TC}(\tilde{h}_i, h_j))}{\sum_{j=1}^{|D|} \exp(att^{TC}(\tilde{h}_i, h_j)) + \sum_{j=1}^{|PicSet|} \exp(att^{TC}(\tilde{h}_i, h^{cap}{}_j))} \quad (20)$$

$$\alpha^{TC}(\tilde{h}_i, h^{cap}{}_j) = \frac{\exp(att^{TC}(\tilde{h}_i, h^{cap}{}_j))}{\sum_{j=1}^{|D|} \exp(att^{TC}(\tilde{h}_i, h_j)) + \sum_{j=1}^{|PicSet|} \exp(att^{TC}(\tilde{h}_i, h^{cap}{}_j))} \quad (21)$$

$$c^{TC}(\tilde{h}_i) = \sum_{j=1}^{|D|} \alpha^{TC}(\tilde{h}_i, h_j) h_j + \sum_{j=1}^{|PicSet|} \alpha^{TC}(\tilde{h}_i, h^{cap}{}_j) h^{cap}{}_j \quad (22)$$

Text-Image Attention (*attTI* for short). This attention model only uses images to represent the image information neglecting the captions. *attTI* computes the importance score of the image img_j based on the relationship between the image encoding hidden state $h^{img}{}_i$ and the decoding hidden state $\tilde{h}_j$.

$$\alpha^{TI}(\tilde{h}_i, h^{img}{}_j) = \frac{\exp(att^{TI}(\tilde{h}_i, h^{img}{}_j))}{\sum_{j=1}^{|D|}\exp(att^{TI}(\tilde{h}_i, h_j)) + \sum_{j=1}^{|PicSet|}\exp(att^{TI}(\tilde{h}_i, h^{img}{}_j))} \quad (23)$$

$$c^{TI}(\tilde{h}_i) = \sum_{j=1}^{|D|}\alpha^{TI}(\tilde{h}_i, h_j)h_j + \sum_{j=1}^{|PicSet|}\alpha^{TI}(\tilde{h}_i, h^{img}{}_j)h^{img}{}_j \quad (24)$$

Text-Image-Caption Attention (*attTIC* for short). This attention model uses both captions and images to represent the image information. *attTIC* computes the importance score of the caption cap_j and the importance score of the image img_j simultaneously, and then compute the context of the decoding hidden state $\tilde{h}_i$ using equation (25).

$$\begin{aligned} c^{TIC}(\tilde{h}_i) &= \sum_{j=1}^{|D|}\alpha^{TIC}(\tilde{h}_i, h_j)h_j \\ &+ \sum_{j=1}^{|PicSet|}\alpha^{TIC}(\tilde{h}_i, h^{cap}{}_j)h^{cap}{}_j + \alpha^{TIC}(\tilde{h}_i, h^{img}{}_j)h^{img}{}_j \end{aligned} \quad (25)$$

In the attention mechanisms, $\alpha(\tilde{h}_i, h_j)$ is the normalized attention score of h_j, $\alpha(\tilde{h}_i, h^{cap}{}_j)$ is the normalized attention score of $h^{cap}{}_j$, $\alpha(\tilde{h}_i, h^{img}{}_j)$ is the normalized attention score of $h^{img}{}_j$, and $c(\tilde{h}_i)$ is the context.

The initial state of the decoder is computed by Equation (12) which can be adjusted according to different attention models.

3.5 Model Training

Since there are no existing manual text-image summaries, and most of the existing training and testing data have pure text summaries, we decide to use pure text summaries as training data to train our models. **The sentence-image alignment relationships can be discovered through training the multi-modal attention models.**

The loss function L of our summarization model is the negative log likelihood of generating text summaries over the training multi-modal document set *MDS*.

$$L = \sum_{(D,PicSet,Y)\in MDS} -\log P(Y \mid D, PicSet) \quad (26)$$

where $Y=[y_1, y_2, \ldots, y_{|Y|}]$ is the word sequences of the summary corresponding to the main text D and the ordered image set *PicSet*, including the tokens <eos>, <neod> and <eod>.

$$\log P(Y \mid D, PicSet) = \sum_{t=1}^{|Y|}\log P(y_t \mid \{y_1, \ldots, y_{t-1}\}, c; \theta) \quad (27)$$

where $\log P(y_t \mid \{y_1, \ldots, y_{t-1}\}, c; \theta)$ is modeled by the multi-modal encoder-decoder model.

We use the Adam (Kingma and Ba, 2014) gradient-based optimization method to optimize the model parameters.

3.6 Multi-Modal Beam Search Algorithm

There are two major problems of the generation of summaries: one is the out-of-vocabulary problem, and the other is the low quality of the generated texts including information incorrectness and repetitions.

For the OOV problem, we use the words in the attended sentences or captions in the original document to replace OOV tokens in the generated summary. Previous research uses the attended words to replace OOV tokens in the flatten encoder-decoder model which attends the words of the original word sequence (Jean, et al., 2015). Our model is hierarchical and multi-modal, and attends sentences, images, and captions when decoding. We use the following algorithm to find the replacement for the j^{th} OOV in a generated sentence:

Step 1: Order the original sentences and captions by the attending scores in descending order.

Step 2: Return the j^{th} OOV word in the ordered sentences and captions as the replacement.

For the *attTI* mechanism that attends images neglecting captions, we use captions instead of the attended images in the algorithm.

For the low-quality generated text problem, we adopt the hierarchical beam search algorithm (Tan el al., 2017). We extend the algorithm by adding caption-level and image-level beam search. The multi-modal hierarchical beam search algorithm comprises *K*-best word-level beam search and *N*-best sentence-caption-level beam search. In particular, we use the corresponding captions instead of images in beam search algorithm for the *attTI* mechanism which attends images.

$$score(y_t) = p(y_t) + \gamma(ref(Y_{t-1} + y_t, s_*) - ref(Y_{t-1}, s_*)) \quad (28)$$

At the word-level search algorithm, we compute the score of generating word y_t using equation (28) where *ref* is a function calculating the ratio of bigram overlap between two texts, s_* is the attended sentence or caption, and γ is the weighting factor. The added term aims to increase the overlap of the generated summary and the original text.

At the sentence level and the caption level, we set the sentence beam width as *N*, and keep *N*-best previously un-referred sentences or captions which have highest attending scores. For each sentence beam, we try *M* sentences or captions and keep the one achieving best word-level scores.

3.7 Image Selection and Alignment

We rank the images, select several most important images as the image summary, and align each sentence with an image in the image summary. The score of images is computed by equation (29).

$$score(img_j) = \sum_{i=1}^{|TextSum|} \alpha_{i,j} \quad (29)$$

where $\alpha_{i,j}$ is the attention score of the j^{th} image when generating the i^{th} sentence of the text summary, and $|TextSum|$ is the number of summary sentences.

The images are ranked by the scores in descending order, and the top K images are selected to form the image summary *ImgSum*. We align each sentence i in *TextSum* to the image j in *ImgSum* such that $\alpha_{i,j}$ is the biggest.

4 Experiments

4.1 Data preparation

We extend the standard DailyMail corpora through extracting the images and the captions from the html-formatted documents. We call the corpora as E-DailyMail. The standard DailyMail and CNN datasets are two widely used datasets for neural document summarization, which are originally built in (Hermann et al., 2015) by collecting human generated highlights and news stories from the news websites. We only extend the DailyMail dataset because it has more images and is easier to collect than the CNN dataset does. We find that the text documents provided by the original DailyMail corpora contain captions. This is due to that all related texts are extracted from the html-formatted news when the corpora are created. We keep the original text documents unchanged in E-DailyMail. The split and statistics of E-DailyMail are shown in Table 1.

4.2 Implementation

We preprocess the text of the E-DailyMail corpora by tokenizing the text and replacing the digits with the <NUM> token. The 40k most frequent words in the corpora are kept and other words are replaced with OOV.

Our model is implemented by using Google's open-source seq2seq-master project written with Tensorflow. We use one layer of the GRU cell. The dimension of the hidden state of the RNN decoder is 512. The dimension of the word embedding vector is 128. The dimension of the hidden state of the bi-directional RNN encoder is 256. We initialize the word embeddings with Google's word2vec tools (Mikolov et al., 2013) trained in the whole text of DailyMail/CNN corpora. We extract the 4096-dimension full-connected layer of 19-layer VGGNet (Simonyan and Zisserman, 2014) as the vector representation of images. We set the parameters of Adam to those provided in (Kingma and Ba, 2014). The batch size is set to 5. Convergence is reached within 800k training steps. It takes about one day for training 40k ~ 50k steps depending on the models on a GTX-1080 TI GPU card. The sentence beam width and the word beam width are set as 2 and 5 respectively. M is set as 3. The parameter γ is set as 3 or 300 tuned on the validation set.

Train	**Dev**	**Test**		
196557	**12147**	**10396**		
D.L.	**S.L.**	**I.N**	**Sent.L**	**Cap.L**
26.0	**3.84**	**5.42**	**26.86**	**24.75**

Table 1: The split and statistics of the E-DailyMail corpora. D.L and S.L indicate the average number of sentences in the document and summary. I.N indicates the average number of images in the story. Sent.L and Cap.L indicates the average number of word in the sentence and the caption respectively.

To train the multi-modal attention mechanism such as *attTIC*, we concatenate the matrix of text representations, image representations, and caption representations to one matrix $M = [h_1, h_2, \ldots h_{|D|}, h^{cap}_1, h^{cap}_2, \ldots, h^{cap}_{|PicSet|}, h^{img}_1, h^{img}_2, \ldots, h^{img}_{|PicSet|}]$. The parameters of the attention mechanisms are trained simultaneously. This way the model training can converge faster.

4.3 Evaluation of Text Summarization

The widely used ROUGE (Lin, 2004) is adopted to evaluate text summaries.

We compare four attention models. *HNNattTC*-3, *HNNattTIC*-3, *HNNattTI*-3, and *HNNattT*-3 are our hierarchical RNN summarization models with the *attTC*, *attIC*, *attTI*, and *attT* attention mechanisms respectively, and 3 is the γ value. *HNNattT* is similar to the model introduced in (Tan et al., 2017) without the graph-based attention. We compare our models with *HNNattT* to show the influence of multi-modal attentions. The first 4 lines in Table 2 are the results with summary length of 75 bytes. The results show that *HNNattTI* has considerable improvement over *HNNattT*. An interesting observation is that *HNNattTC* and *HNNattTIC* are not better than *HNNattT*. One of the

Method	Rouge-1	Rouge-2	Rouge-L
***HNNattTI*-3**	**24.84**	8.7	**16.99**
HNNattTC-3	18.61	6.7	13.44
HNNTattTIC-3	21.17	8.1	15.24
HNNattT-3	22.09	7.9	15.97
Lead	21.9	7.2	11.6
NN-SE	22.7	8.5	12.5
SummaRuNNer-abs	23.8	9.6	13.3
LREG(500)	18.5	6.9	10.2
NN-ABS(500)	7.8	1.7	7.1
NN-WE(500)	15.7	6.4	9.8

Table 2: Comparison results on the DailyMail test set using Rouge recall at 75 bytes.

Method	Rouge-1	Rouge-2	Rouge-L
***HNNattTI*-300**	**32.64**	**12.02**	**23.88**
HNNattTC-300	26.75	10.12	19.42
HNNattTIC-300	30.52	11.04	21.81
HNNattT-300	31.34	11.81	22.93

Table 3: Comparison results on the DailyMail test set using full-length F1metric.

reasons is that the text documents provided by the DailyMail corpora contain captions. Captions are already parts of the text documents. The other reason is that captions distract attentions and cannot attract sufficient attentions from the original sentences, which will be discussed in the next subsection.

We compare our methods with state-of-the-art neural summarization methods reported in recent papers on the DailyMail corpora. Extractive models include Lead which is a strong baseline using the leading 3 sentences as the summary, NN-SE (Cheng and Lapata, 2016), and *SummaRuNNer-abs* (Nallapati et al., 2017) which is trained on the abstractive summaries. Abstractive models include *NN-ABS*, *NN-WE*, *LREG*, though they are tested on 500 samples of the test set. *LREG* is a feature-based method using linear regression. *NN-ABS* is a simple hierarchical extension of (Rush et al., 2015). *NN-WE* is the abstractive model restricting the generation of words from the original document. The results are shown in the last 6 rows in Table 2. Our method *HNNattTI* outperforms the three extractive models and the three abstractive models.

We compare our models under the full-length F1 metric by setting the γ value as 300. According to (Tan et al., 2017), a large γ makes the generated summary has more overlaps with the attended texts, and thus partly overcome the repeated sentences problem in the generated summary. We do not incorporate the attention distraction mechanism (Chen et al., 2016) into our model, because we want to focus on our own model to see whether considering images improves text summarization. Results in Table 3 also show that *HNNattTI* performs better than *HNNattT*, *HNNattTC*, and *HNNattTIC*.

Method	Rouge-1	Rouge-2	Rouge-L
***HNNattTI*-3-OOV**	24.03	8.2	16.52
HNNattTC-3-OOV	18.18	6.53	12.87
HNNTattTIC-3-OOV	20.50	7.67	14.36
HNNattT-3-OOV	21.60	7.82	15.05

Table 4: Comparison results using Rouge recall at 75 bytes without OOV replacement. *HNNattTI*-3-OOV is the version of *HNNattTI*-3 without the OOV replacement mechanism.

Method	Rouge-1	Rouge-2	Rouge-L
***HNNattTI*-300-OOV**	32.03	11.52	22.67
HNNattTC-300-OOV	26.13	9.87	19.03
HNNattTIC-300-OOV	30.11	10.87	21.12
HNNattT-300-OOV	30.74	11.21	22.28

Table 5: Comparison results using full-length F1 metric without OOV replacement. *HNNattTI*-300-OOV is the version of *HNNattTI*-300 without the OOV replacement mechanism.

To show the influence of our OOV replacement mechanism, we eliminate the mechanism from our models, and show the evaluation results in Table 4 and Table 5. We can see from the two tables that the scores are lower than the corresponding scores in Table 2 and Table 3. Our OOV replacement mechanism improves the summarization models, though the mechanism is relatively simple.

In short, combining and attending images in the neural summarization model improves document summarization.

4.4 Evaluation of Image Summarization

To evaluation the image summarization, the gold standard image summary is generated based on a greedy algorithm on the captions as follows: at each time i, choose img_k to maximize $Rouge(\{cap_1,\ldots cap_{i-1},cap_k\},\ Abs\_Sum)$ - $Rouge(\{cap_1,\ldots cap_{i-1}\},\ Abs\_Sum))$ where Abs_Sum is the ground truth text summary and

num	*HNNattTI*	*HNNattTC*	*HNNattTIC*	*Random*
1	**0.4978**	0. 4137	0. 4362	0. 4721
2	**0.4783**	0. 3998	0. 4230	0. 4517

Table 6: Image summarization using the recall metric for the 1-image or 2-images summary. γ is set as 300.

cap$_k$ is the caption of *img*$_k$. The average number of images in summaries is 2.15. The average Rouge-1, Rouge-2, and Rouge-L scores of the caption summaries with respect to the ground truth summaries are 43.85, 19.70, and 36.30 respectively.

We use the 1-image and 2-image random selected image summaries as the baselines which we compare our models with. The top 1 or 2 images ranked by our model are selected out to form the summaries. Results in Table 4 show that *HNNattTI* outperforms the random baseline, while *HNNattTC* and *HNNattTIC* perform worse. This implies that attending images can generate better sentence-image alignment in the multi-modal summaries than the model attending captions does. And this can also partly explain why our summarization model attending images when decoding can generate better text summaries than the one attending captions does.

4.5 Instance

Figure 4 shows the text-image summary of the example demonstrated in Figure 1 generated by the *HNNattTI* model. In the summary, there are 2 images and 3 generated sentences, and each sentence is aligned with an image. The image summary has one common image with Figure 2.

S2: smith suffered a broken leg and a dislocated ankle as a result of a broken leg as a result of the incident at anfield.

S1: smith was carried off on a stretcher before he was carried off on a stretcher in a stretcher.

S3: he was travelling to hospital during manchester united's fa cup fifth round exit nine years ago when he was travelling to the ambulance.

Figure 4: The generated text-image summary of the example in Figure 1.

	IMG1	IMG2	IMG3	IMG4
S1	0.0947	0.1089	0.1157	0.1194
S2	0.0893	0.1020	0.1070	0.1052
S3	0.0853	0.0769	0.0946	0.0969

Table 7: The sentence-image alignment scores of the generated summary for the news in Figure 1.

The sentences are named by S1, S2, and S3 respectively.

Table 7 shows the sentence-image alignment scores. The four images in the original document are numbered from top to bottom and left to right by IMG1, IMG2, IMG3, and IMG4. The summation of alignment scores for a summary sentence is less than 1, because the sentence is also aligned with the sentences in the original document.

5 Conclusions

This paper proposes the text-image summarization task to summarize and align texts and images simultaneously. Most previous research summarizes texts and images separately, and few has been done on text-image summarization. We propose the multi-modal attentional mechanism which attends original sentences, images, captions simultaneously in the hierarchical encoder-decoder model, use the RNN model to encode the ordered image set as the initial state of the decoder, and propose the multi-modal beam search algorithm which scores beams using the bigram overlaps of the generated sentences and the captions. The model is trained by using abstractive text summaries as the targets, and the attention scores of images are used to score images. The original DailyMail dataset is extended by collecting images and captions from the Web. Experiments show that our model attending images outperforms the models not attending images, three existing neural abstractive models and three existing extractive models. Experiments also show our model can generate informative summaries of images.

Acknowledgments

The research was sponsored by the National Natural Science Foundation of China (No.61806101, No.61876048) and the Natural Science Foundation of Jiangsu Province (BK20150862). We thank the anonymous reviewers for helpful comments. Professor Hai Zhuge is the corresponding author.

References

Agrawal, R., Gollapudi, S., Kannan, A., & Kenthapadi, K. (2011). Enriching textbooks with images. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 1847-1856). ACM.

Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. Computer Science, 2014.

Chen, Q., Zhu, X., Ling, Z., Wei, S., and Jiang, H. (2016, July). Distraction-based neural networks for modeling documents. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (pp. 2754-2760). AAAI Press.

Cheng, J. and Lapata, M., 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Cho, K., Van Merriënboer, B., Bahdanau, D. and Bengio, Y., 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Cui, Y., Chen, Z., Wei, S., Wang, S., Liu, T. and Hu, G., 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009, June. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 248-255). IEEE.

Gu, J., Lu, Z., Li, H. and Li, V.O., 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Vol. 1, pp. 1631-1640).

Greenbacker, C. F., 2011. Towards a framework for abstractive summarization of multimodal documents. In *Proceedings of the ACL 2011 Student Session* (pp. 75-80). Association for Computational Linguistics.

Hermann, K.M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M. and Blunsom, P., 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems* (pp. 1693-1701).

Jean, S., Cho, K., Memisevic, R. and Bengio, Y., 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky A., Sutskever I., Hinton G. E. ImageNet classification with deep convolutional neural networks. Communications of the Acm, 2013, 60(2):2012.

Li J., Luong M. T., Jurafsky D. A Hierarchical Neural Autoencoder for Paragraphs and Documents. Computer Science, 2015.

Lin, C.Y., 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Liu C., Mao J., Sha F., Yuille AL. 2017a. Attention Correctness in Neural Image Captioning. InAAAI 2017 Feb 4 (pp. 4176-4182).

Liu, C., Sun, F., Wang, C., Wang, F. and Yuille, A., 2017b. MAT: A multimodal attentive translator for image captioning. *arXiv preprint arXiv:1702.05658*.

Luong, M. T., Hieu P., and Christopher D. M.. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025* (2015).

Mikolov T., Sutskever I., Chen K., et al. Distributed Representations of Words and Phrases and their Compositionality. 2013, 26:3111-3119.

Nallapati R., Zhai F., Zhou B. SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents. 2016.

Rossiter M. J., Derwing T. M., Jones V. M. L. O. Is a Picture Worth a Thousand Words?. Tesol Quarterly, 2012, 42(2):325-329.

Rush A. M., Chopra S., Weston J. A Neural Attention Model for Abstractive Sentence Summarization. Computer Science, 2015.

Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. Computer Science, 2014.

Szegedy C., Liu W., Jia Y., et al. Going deeper with convolutions. CoRR, abs/1409.4842, 2014.

Tan, J., Wan, X. and Xiao, J., 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Vol. 1, pp. 1171-1181).

UzZaman, N., Bigham, J. P., & Allen, J. F. (2011). Multimodal summarization of complex sentences. In *Proceedings of the 16th international conference on Intelligent user interfaces* (pp. 43-52). ACM.

Wang W. Y., Mehdad Y., Radev D. R., et al. A Low-Rank Approximation Approach to Learning Joint

Embeddings of News Stories and Images for Timeline Summarization. NAACL. 2016:58-68.

Wu, P., & Carberry, S. (2011). Toward extractive summarization of multimodal documents. In *Proceedings of the Workshop on Text Summarization at the Canadian Conference on Artificial Intelligence* (pp. 53-61).

Xu K., Ba J., Kiros R., et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Computer Science, 2015:2048-2057.

Yan R., Wan X., et al. Visualizing timelines: evolutionary summarization via iterative reinforcement between text and image streams. CIKM2012. ACM, 2012:275-284.

You Q., Jin H., Wang Z., et al. Image Captioning with Semantic Attention. IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2016:4651-4659.

Zhou Q., Yang N., Wei F., et al. Selective Encoding for Abstractive Sentence Summarization. Meeting of the Association for Computational Linguistics. 2017:1095-1104.

Zhu, X., Goldberg, A. B., et al. (2007). A text-to-picture synthesis system for augmenting communication. In *AAAI* (Vol. 7, pp. 1590-1595).

Zhuge, H. Multi-Dimensional Summarization in Cyber-Physical Society, Morgan Kaufmann, 2016.

Keyphrase Generation with Correlation Constraints

Jun Chen[†], Xiaoming Zhang[◇], Yu Wu[†♡], Zhao Yan[‡], Zhoujun Li[†*]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[◇]School of Cyber Science and Technology, Beihang University, Beijing, China
[♡] The author is supported by AdeptMind Scholarship
[‡]Tencent Lingo Lab, Beijing, China
`{chenjun1994,yolixs,wuyu,lizj}@buaa.edu.cn`
`zhaoyan@tencent.com`

Abstract

In this paper, we study automatic keyphrase generation. Although conventional approaches to this task show promising results, they neglect correlation among keyphrases, resulting in duplication and coverage issues. To solve these problems, we propose a new sequence-to-sequence architecture for keyphrase generation named CorrRNN, which captures correlation among multiple keyphrases in two ways. First, we employ a coverage vector to indicate whether the word in the source document has been summarized by previous phrases to improve the coverage for keyphrases. Second, preceding phrases are taken into account to eliminate duplicate phrases and improve result coherence. Experiment results show that our model significantly outperforms the state-of-the-art method on benchmark datasets in terms of both accuracy and diversity.

1 Introduction

A keyphrase is a piece of text that is able to summarize a long document, organize contents and highlight important concepts, like "virtual organizations" in Table 1. It provides readers with a rough understanding of a document without going through its content, and has many potential applications, such as information retrieval, text summarization and document classification.

Keyphrase can be categorized into present keyphrase which appears in a source document, and absent keyphrase that does not appear in the document. Conventional approaches extract important text spans as candidate phrases and rank them as keyphrases (Hulth, 2003; Medelyan et al., 2008; Liu et al., 2011; Wu et al., 2015; Wang et al., 2016), that show promising results on the present keyphrases but cannot handle absent keyphrases.

Title: Resolving conflict and inconsistency in norm regulated virtual organizations.
Abstract: Norm governed virtual organizations define, govern and facilitate coordinated resource sharing and problem solving in societies of agents. With an explicit account of norms, openness in virtual organizations can be achieved new components, designed by various parties, can be seamlessly accommodated. We focus on virtual organizations realised as multi agent systems ...

Ground truth: virtual organizations; multi agent systems; agent; norm conflict; conflict prohibition; norm inconsistency; ...
Predicted keyphrases: virtual organizations; multi agent systems; artificial intelligence; inter agent; multi agent; action delegation; software agents; resource sharing; grid services; agent systems;

Table 1: The example shows the duplication and coverage issues of state-of-the-art model. The phrases in red are duplicate, and the underlined parts in the source document are not covered by the predicted results, while they are summarized by "norm conflict" and "norm inconsistency" in the golden list.

To predict absent keyphrases, generative methods have been proposed by Meng et al. (2017). The approach employs a sequence-to-sequence (Seq2Seq) framework (Sutskever et al., 2014) with a copy mechanism (Gu et al., 2016) to encourage rare word generation, in which the encoder compresses the text into a dense vector and the decoder generates a phrase with a Recurrent Neural Network (RNN) language model, achieving state-of-the-art performance. Since a document corresponds to multiple keyphrases, the approach divides it into multiple document-keyphrase pairs as training instances. This approach, however, neglects the correlation among target keyphrases since it does not model the one-to-many relationship between the document and keyphrases. Therefore, keyphrase prediction only depends on the source document, and ignores the keyphrases which have been generated. As a consequence, the generated keyphrases suffer from **duplication issue** and **coverage issue**. A duplication issue is defined as at least two phrases expressing the same meaning, hindering readers from obtaining more information from keyphrases. For example, three keyphrases have an identical meaning in Table 1,

---

[*] Corresponding Author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4057–4066
Brussels, Belgium, October 31 - November 4, 2018.

including ”multi agent systems”, ”multi agent” and ”agent systems”. A coverage issue means some key points in the document are not covered by the keyphrases, such as ”norm conflict” and ”norm inconsistency” in Table 1.

To mitigate such issues, we mimic human behavior in terms of how to assign keyphrases for an arbitrary document. Given a document in Table 1, an annotator will read it and generate keyphrases according to his understanding of the content, like ”virtual organizations”, ”multi agent systems”. After that, instead of generating duplicate phrases like ”agent systems” and ”multi agent”, the annotator will review the document and preceding keyphrases, then generate a phrase like ”norm conflict” to cover topics that have not been summarized by previous phrases. The iteration does not stop until all of a document’s topics are covered by keyphrases.

We propose a new sequence-to-sequence architecture CorrRNN, capable of capturing correlation among keyphrases. Notably, correlation constraints in this paper are defined as keyphrases that should cover all topics in the source document and differ from each other. Specifically, we employ a coverage mechanism (Tu et al., 2016) to memorize which parts in the source document have been covered by previous phrases. In this way, the document coverage is modeled explicitly, enabling the generated keyphrases to cover more topics. Furthermore, we propose a review mechanism that considers the previous keyphrases in the generation process, in order to avoid the repetition in the final results. Concretely, the review mechanism explicitly models the correlation between the keyphrases that have been generated and the keyphrase that is going to be generated with a novel architecture. It extends the existing Seq2Seq model and captures the one-to-many relationship in keyphrase generation. Augmented with the coverage mechanism and the review mechanism, CorrRNN does not only inherit the advantages of the Seq2Seq model, but also improves the coverage and diversity in the generation process.

We test our model on three benchmark datasets. The results show that our model outperforms state-of-the art methods by a large margin, demonstrating the effectiveness of the correlation constraints. In addition, our model is better than heuristic rules on improving diversity, since it instills the correlation knowledge to the model in an end-to-end fashion.

Our contributions in this paper are three-fold: (1) the proposal of modeling the one-to-many correlation for keyphrase generation, (2) the proposal of a new architecture CorrRNN for keyphrase generation, and (3) empirical verification of the effectiveness of CorrRNN on public datasets.

In the remainder of this paper, we will first review the related work in Section 2, then we elaborate on the proposed model in Section 3. After that, we list the experiment settings in Section 4, results and discussion follow in Section 5. Finally, the conclusion and future work in Section 6.

2 Related Work

How to assign keyphrases to a long document is a fundamental task, that has been studied intensively in previous works. Existing methods can be categorized into two groups: extraction based and generation based methods.

The former group extracts important keyphrases in a document which consists of two phases. The first phase is to construct a set of phrase candidates with heuristic methods, such as extracting important n-grams (Hulth, 2003; Medelyan et al., 2008; Hulth, 2003; Shang et al., 2017) and selecting text chunks with certain postags (Liu et al., 2011; Wang et al., 2016; Le et al., 2016; Liu et al., 2015). The second phase is to rank the candidates with machine learning methods. Specifically, some researchers (Frank et al., 1999; Witten et al., 1999; Hulth, 2003; Medelyan et al., 2009; Gollapalli and Caragea, 2014) formulate the keyphrase extraction as a supervised classification problem, while others apply unsupervised approaches (Mihalcea and Tarau, 2004; Grineva et al., 2009; Liu et al., 2009, 2010; Zhang et al., 2013; Bougouin et al., 2013, 2016) on this task. Besides, Tomokiyo and Hurst (2003) employ two statistical language models to measure the informativeness for phrases. Liu et al. (2011) use a word alignment model to learn translation probabilities between the words in documents and the words in keyphrases, which alleviates the problem of vocabulary gaps.

The latter group, generative methods, assigns keyphrases to a document with natural language generation techniques, and is capable of generating absent keyphrases. Owing to the development of neural networks (Bahdanau et al., 2014), Meng et al. (2017) apply an encoder-decoder framework (Sutskever et al., 2014) with a copy mechanism

(Gu et al., 2016) to this task, achieving state-of-the-art performance.

Our work is a generation based approach. The main difference of our model is that we consider the correlation among keyphrases. Our model proposes a new review mechanism to enhance keyphrase diversity, while employs a coverage mechanism that has proven effective for summarization (See et al., 2017) and machine translation (Tu et al., 2016) to guarantee keyphrase coverage. Some previous works on keyword extraction have already exploited the correlation problem with a re-rank strategy (Habibi and Popescu-Belis, 2013; Ni et al., 2012). In contrast, we model the correlation in an end-to-end fashion.

3 Keyphrase Generation with Correlation

3.1 Problem Formalization

Suppose that we have a data set $\mathcal{D} = \{\mathbf{x}_i, \mathbf{p}_i\}_{i=1}^{N}$, where $\mathbf{x_i}$ is a source text, $\mathbf{p}_i = \{p_{i,j}\}_{j=1}^{M_i}$ is the keyphrase set of $\mathbf{x_i}$, and N is the number of documents. Both the source text and target keyphrase are word sequences, donated as $\mathbf{x}_i = (x_1^i, x_2^i, ..., x_T^i)$ and $p_{i,j} = (y_1^{i,j}, y_2^{i,j}, ..., y_{L^i}^{i,j})$ respectively. T and L^i are the length of word sequences of $\mathbf{x_i}$ and $p_{i,j}$. Prior work aims to maximize the probability of $\prod_{i=1}^{N}\prod_{j=1}^{M_i} P(p_{i,j}|\mathbf{x}_i)$, while our model considers keyphrase correlation to address coverage and duplication issues by maximizing the probability of $\prod_{i=1}^{N}\prod_{j=1}^{M_i} P(p_{i,j}|\mathbf{x}_i, p_{i,l<j})$.

3.2 Seq2Seq Model with Copy Mechanism

A Seq2Seq model (Sutskever et al., 2014) is employed as backbone in this paper. The encoder converts the variable-length input sequence $x = (x_1, x_2, ..., x_T)$ into a set of hidden representation $h = (h_1, h_2, ..., h_T)$ by iterating along time t with the following equation:

$$h_t = f(x_t, h_{t-1}) \tag{1}$$

where f is a non-linear function.

Then the context vector c is computed as a weighted sum of hidden representation set h through an attention mechanism (Bahdanau et al., 2014), which next acts as the representation of the whole input x at time step t.

$$c_t = \sum_{j=1}^{T} \alpha_{tj} h_j \tag{2}$$

where α_{tj} is a coefficient which measures the match degree between the inputs around position j and the output at position t.

With the context vector c_t, decoder generates variable-length word sequence step by step, the generative process which is known as a language model:

$$s_t = f(y_{t-1}, s_{t-1}, c_t) \tag{3}$$

$$p_g(y_t|y_{<t}, x) = g(y_{t-1}, s_t, c_t) \tag{4}$$

where s_t denotes the hidden state of the decoder at time t. y_t is the predicted word from vocabulary based on the largest probability after $g(.)$.

Unfortunately, pure generative mode cannot generate any keyphrase (e.g. noun, entity) which contains out-of-vocabulary words. Thus we incorporate a copy mechanism (Gu et al., 2016) into the encoder-decoder model to predict out-of-vocabulary words by selecting appropriate words from source text. After incorporation, the probability of predicting a new word consists of two parts:

$$p(y_t|y_{<t}, x) = p_g(y_t|y_{<t}, x) + p_c(y_t|y_{<t}, x) \tag{5}$$

$$p_c(y_t|y_{<t}, x) = \frac{1}{Z} \sum_{j:x_j=y_t} e^{\sigma(h_j^T W_c)[y_{t-1}; s_t; c_t]}, y_t \in \mathcal{X} \tag{6}$$

where p_g and p_c denote the probability of generating and coping. $\mathcal{X}$ is the set of unique words in source sequence x, σ is a non-linear function. $W_c \in \mathbb{R}$ is a learned parameter matrix. Z is the sum for normalization. For more details, please see (Gu et al., 2016).

3.3 Model Correlation

Keyphrases should cover more topics and differ from each other, while previous work (Meng et al., 2017) ignored this correlation among multiple keyphrases, resulting in duplication and coverage issues. In this part, we focus on capturing the one-to-many correlation to alleviate above issues. On the one hand, we employ a coverage mechanism (Tu et al., 2016) that diversifies attention distributions to improve the topic coverage of keyphrases. On the other hand, we propose a review mechanism which makes use of contextual information of previous phrases (already generated) to avoid duplicate generation. For better display of the proposed model, the overall framework is illustrated in Figure 1 and the detailed process is described in Algorithm 1.

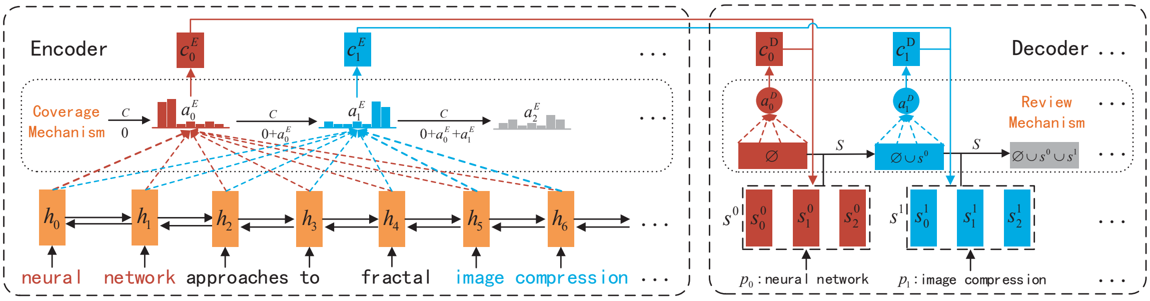

Figure 1: The overall framework structure. Note that p_i indicates a keyphrase (e.g. p_0 ="neural network"), s^i indicates the hidden state set of phrase p_i, coverage vector C and target-side review context S update and transmit along the process of decoding multiple keyphrases.

3.3.1 Coverage Mechanism

As is well known, multiple keyphrases usually correspond to multiple different positions of the source text (see Figure 1), the positions of words that have already been summarized should not be focused again since the attention mechanism automatically focuses on important areas of source text. To overcome the coverage issue, we incorporate a coverage mechanism (Tu et al., 2016) into our model, which diversifies the attention distributions of multiple keyphrases to make sure more important areas in the source document are attended and summarized into keyphrases.

Concretely, we maintain a coverage vector c^t which is the sum of attention distributions over all previous decoder time steps. Intuitively, c^t represents the degree of coverage that those words in the source text have received from the attention mechanism so far.

$$c^t = \sum_{i=0}^{t-1} a^i \tag{7}$$

Note that c^0 is a zero vector since no word in source text has be covered.

Later, the coverage vector c^t is an extra input for the attention mechanism, then the source context set h is read and weight averaged into a contextual representation c_t^E by the attention mechanism with a coverage vector, with Eqn.(2) transforming into Eqn.(8) as follows:

$$c_t^E = \sum_{j=1}^{T} \alpha_{tj} h_j; \quad \alpha_{tj} = \frac{exp(e_{tj}^E)}{\sum_{k=1}^{T} exp(e_{tk}^E)};$$
$$e_{tj}^E = v^T tanh(W_h h_j + W_s s_{t-1} + \mathbf{w_c} \mathbf{c_j^t} + b_{attn}) \tag{8}$$

where E is the encoder and $\mathbf{w_c}$ is a learned parameter with the same length as v.

With the coverage vector, the attention mechanism's decision for choosing where in source text to focus next is informed by a reminder of its previous decisions, which ensures that the attention mechanism avoids repeatedly attending to the same locations in the source text more easily, thus generated phrases cover more topics in the source document.

Algorithm 1 Training procedure of the proposed model.

```
Require: The train corpus set $\mathcal{D}$; The encoder $\varphi_E$; The decoder $\varphi_D$; The attention function $attention$;
for each $(X, (P_1, ..., P_M)) \in \mathcal{D}$ do
    compute source hidden states $H = \varphi_E(X)$;
    init target review context $S = \phi$;
    init coverage vector $C = \vec{0}$;
    for each $P_i = (y_1^i, y_2^i, ..., y_{T^i}^i) \in (P_1, ..., P_M)$ do
        init $s_0^i$;
        for $t = 1; t <= T^i; t++$ do
            $c_t^{iE}, a_t^i = attention(H, s_{t-1}^i, C)$ ;
            $c_t^{iD} = attention(S, s_{t-1}^i)$ ;
            $s_t^i = \varphi_D(y_{t-1}^i, s_{t-1}^i, c_t^{iE}, c_t^{iD})$ ;
            $S = S \cup \{s_{t-1}^i\}$;
            $C = C + a_t^i$ ;
        end for
        compute $loss$ for $P_i$;
    end for
    compute $gradient$ and $update$;
end for
```

3.3.2 Review Mechanism

Considering human behavior on assigning keyphrases that review previous phrases to avoid duplicate assignment, we construct a target side review context set which contains contextual information of generated phases. The target context with an attention mechanism can make use of contextual information of generated phrases to help predict the next phrase, which we call the review mechanism.

Like source context c_t^E described above, on the target side, the target context is defined as $s^t = \{s_1, s_2, ..., s_{t-1}\}$, which is the collection of hidden states before time step t. When decoding the word at t-th step, s^t is used to inform an extra contextual representation, thus target side attentive contexts are integrated into c_t^D:

$$c_t^D = \sum_{j=1}^{t-1} \beta_{tj} s_j; \quad \beta_{tj} = \frac{exp(e_{tj}^D)}{\sum_{k=1}^{t-1} exp(e_{tk}^D)};$$

$$e_{tj}^D = v^T tanh(W_h s_j + W_s s_{t-1}) \tag{9}$$

Afterwards, c_t^D is provided as an extra input to derive the hidden state s_t and later the probability distribution for choosing t-th word . The target context gets updated consequently as $s^{t+1} = s^t \cup \{s_t\}$ in the decoding progress.

$$s_t = f(y_{t-1}, s_{t-1}, c_t^E, \mathbf{c_t^D}) \tag{10}$$

$$p_g(y_t|y_{<t}, x) = g(y_{t-1}, s_t, c_t^E, \mathbf{c_t^D}) \tag{11}$$

$$p_c(y_t|y_{<t}, x) = \frac{1}{Z} \sum_{j:x_j=y_t} e^{\sigma(h_j^T W_c)[y_{t-1}; s_t; c_t^E; \mathbf{c_t^D}]}, y_t \in \mathcal{X} \tag{12}$$

Eqn.(10), Eqn.(11) and Eqn.(12) are transformed from Eqn.(3), Eqn.(4) and Eqn.(6) respectively. More mathematical details are displayed below to make Eqn.(10) more clear:

$$\begin{aligned}
r_t &= \sigma(W_r y_{t-1} + U_r s_{t-1} + C_r^E c_t^E + \mathbf{C_r^D c_t^D});\\
z_t &= \sigma(W_z y_{t-1} + U_z s_{t-1} + C_z^E c_t^E + \mathbf{C_z^D c_t^D});\\
\widetilde{s_t} &= tanh(W y_{t-1} + U[r_t \circ s_{t-1}] + C^E c_t^E + \mathbf{C^D c_t^D});\\
s_t &= (1 - z_t) \circ s_{t-1} + z_t \circ \widetilde{s_t}
\end{aligned} \tag{13}$$

where E, D indicate the encoder and decoder, W, U, C are learned parameters of the model, σ is a $sigmoid$ function, $\circ$ indicates an element-wise product.

With the contextual information of previous phrases, review mechanism ensures next predicted phrase less duplication and topic coherence. So far, we transmit and update the coverage vector and review context along the multi-target phrase decoding process to improve the coverage and diversity of keyphrases. We denote our model with coverage only and review only as CorrRNN$_C$ and CorrRNN$_R$, and empirically compare them in experiments. The objectives are to minimize the negative log-likelihood of the target words, given a data sample with source text x and corresponding phrases set $p = \{p_i\}_{i=0}^M$, loss is calculated as follows:

$$loss = -\frac{1}{M} \sum_{i=0}^{M} \sum_{t=0}^{T_i} log(p(y_t^i | y_{<t}^i, x, p_{j,j<i})) \tag{14}$$

4 Experiment Settings

4.1 Implementation Details

In the preprocessing phase, we follow (Meng et al., 2017) to preprocess the text with tokenization, lowercasing, and digit replacement to ensure fairness. Each article consists of one source text and corresponding multiple keyphrases, and the source text is the concatenation of its title and abstract. We set the max number of target phrases to 10 for an article in consideration of the device memory, thus those with more than 10 target phrases are cut into multiple articles. Finally, we have 558830 articles (text-keyphrases pair) for training.

In the training phase, we choose a bidirectional GRU for the encoder and another forward GRU for the decoder. The top 50000 frequent words are chosen as the vocabulary, the dimension of word embeddings is set to 150, the value of embedding is randomly initialized with uniform distribution in [-0.1, 0.1], and the dimension of the hidden layers is set to 300. Adam is adopted to optimize the model with initial learning rate=10^{-4}, gradient clipping=0.1 and dropout rate=0.5. The training is stopped once the loss on the validation set stops dropping for several iterations.

In the generation phase, we use beam search to generate multiple phrases. The beam depth is set to 6 and the beam size is set to 200. Source code will be released at `https://github.com/nanfeng1101/s2s-kg`.

4.2 Datasets

Following Meng et al. (2017), we train our model on the **KP20k** dataset (Meng et al., 2017), which contains articles collected from various online digital libraries. The dataset has 527,830 articles for training and 20000 articles for validation.

We evaluate our model on three benchmark datasets which are widely adopted in previous works, with the details described below:

- **NUS** (Vijayakumar et al., 2016): It contains 211 papers with author-assigned keyphrases, all of which we use as test data.

- **Semeval-2010** (Kim et al., 2010): 288 articles are collected from the ACM Digital Library. 100 of them are used for test data and the rest are added to the training set.
- **Krapivin** (Krapivin et al., 2008): This dataset contains 2304 papers. The first 400 papers in alphabetical order are used for evaluation and the rest are added to the training set.

4.3 Baseline Models

We compare our model with extractive and generative approaches. The extractive baselines include Tf-idf, TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), ExpandRank (Wan and Xiao, 2008), TopicRank[1] (Bougouin et al., 2013), KEA (Witten et al., 1999) and Maui (Medelyan et al., 2009). The generative baselines include RNN and CopyRNN (Meng et al., 2017). In these baselines, the first five are unsupervised and the later four are supervised. We set up all the baselines following optimal settings in (Hasan and Ng, 2010), (Bougouin et al., 2013) and (Meng et al., 2017).

To demonstrate the effectiveness of end-to-end learning, we compare CorrRNN to CopyRNN with post-processing. In post-precessing, we only keep the first appearence of keyphrase in duplicate predictions, duplication means that a phrase is a substring of another. The baseline can be seen as heuristic rules for improving the diversity, denoted as CopyRNN$_F$.

4.4 Evaluation Metrics

For a fair comparison, we evaluate the experiment results on present keyphrases and absent keyphrases separately, because extractive methods cannot generate absent keyphrases. Following Meng et al. (2017), we employ F1-measure for present keyphrases and recall for absent keyphrases. Here, we use **F1@K** and **R@K** to denote the F1 and recall score in the top K keyphrases. Note that we use Porter Stemmer for preprocessing to determine whether the two keyphrases are identical.

Furthermore, α-NDCG, which is widely used to measure the diversity of keyphrase generation (Habibi and Popescu-Belis, 2013) and information retrieval (Clarke et al., 2008), is adopted to evaluate the diversity of the generative methods, denoted as **N@K**. α is a trade-off between relevance and diversity in α-NDCG, which is set to equal weights of 0.5 according to Habibi and Popescu-Belis (2013). The higher α-NDCG is, the more diverse the results are. We re-implement CopyRNN with the source code[2] provided by the authors in order to evaluate it on the α-NDCG metric.

$$\alpha\text{-}NDCG[k] = \frac{DCG[k]}{DCG'[k]};$$
$$DCG[k] = \sum_{j=1}^{k} G[j]/log_2(j+1);$$
$$G[k] = \sum_{i=1}^{m} J(d_k, i)(1-\alpha)^{r_{i,k-1}}$$

where α is a parameter, m denotes the number of target phrases, k denotes the number of predicted phrases. $J(d_k, i) = 0$ or 1, which indicates whether the k-th predicted phrase is relevant to the i-th target phrase, and $r_{i,k-1}$ indicates how many predicted phrases are relevant to the i-th target phrase before the k-th predicted phrase. Note that relevance here is defined as whether the word set of a keyphrase is a subset of another keyphrase (e.g. ”multi agent” vs ”multi agent system”).

5 Results and Analysis

5.1 Present Phrase Prediction

Present phrase prediction is also known as keyphrase extraction in prior studies. We evaluate how well our model performs on this common task. The results are shown in Tables 2 and 3, which list the performance of the top 5 and top 10 results.

In terms of F1-measure, CorrRNN and CopyRNN outperform other baselines by a large margin, indicating the effectiveness of RNN with a copy mechanism. As we consider the correlation among multiple phrases, the overall results of CorrRNN are better than CopyRNN significantly ($t-$test with $p < 0.01$). This is mainly because CorrRNN alleviates the duplication and coverage issues in existing methods, with more correct phrases boosted in the top 10 results. Heuritic baseline CopyRNN$_F$ is even worse than CopyRNN, indicating that the heuristic rules may hurt the performance of generative approaches. It also proven that it is a better way to model the correlation among keyphrases in an end-to-end fashion.

Regarding α-NDCG, CorrRNN and its variants surpass other methods, demonstrating that incorporating correlation constraints can improve both

---

[1] https://github.com/adrien-bougouin/KeyBench

[2] https://github.com/memray/seq2seq-keyphrase

relevance and diversity. As the heuristic rules influence the relevance of CopyRNN, CopyRNN$_F$ performs a little better than CopyRNN on the α-NDCG.

Model	**NUS**		**SemEval**		**Krapivin**	
	F1@5	**F1@10**	**F1@5**	**F1@10**	**F1@5**	**F1@10**
Tf-idf	0.136	0.184	0.128	0.194	0.129	0.160
TextRank	0.195	0.196	0.176	0.187	0.189	0.162
SingleRank	0.140	0.173	0.135	0.176	0.189	0.162
ExpandRank	0.132	0.164	0.139	0.170	0.081	0.126
TopicRank	0.115	0.123	0.083	0.099	0.117	0.112
Maui	0.249	0.268	0.044	0.039	0.249	0.216
KEA	0.069	0.084	0.025	0.026	0.110	0.152
RNN	0.169	0.127	0.157	0.124	0.135	0.088
CopyRNN	0.334	0.326	0.291	0.304	0.311	0.266
CopyRNN$_F$	0.323	0.289	0.270	0.270	0.293	0.222
CorrRNN$_C$	**0.361**	**0.335**	0.296	0.319	0.311	0.273
CorrRNN$_R$	0.354	0.328	0.306	0.312	0.314	0.270
CorrRNN	0.358	0.330	**0.320**	**0.320**	**0.318**	**0.278**

Table 2: The F1 performance on present phrase prediction. From top to bottom, baselines are listed as unsupervised and supervised.

Model	**NUS**		**SemEval**		**Krapivin**	
	N@5	**N@10**	**N@5**	**N@10**	**N@5**	**N@10**
CopyRNN	0.740	0.713	0.682	0.667	0.622	0.625
CopyRNN$_F$	0.743	0.720	0.692	0.681	0.635	**0.657**
CorrRNN$_C$	**0.781**	0.747	0.728	0.694	0.649	0.642
CorrRNN$_R$	0.770	0.742	0.718	0.692	**0.669**	**0.657**
CorrRNN	0.771	**0.752**	**0.752**	**0.720**	0.659	0.647

Table 3: The α-NDCG performance on present phrase prediction.

5.2 Absent Phrase Prediction

We evaluate the performance of generative methods within the recall of the top 10 results, which is shown in Table 4. We can see that both CopyRNN and CorrRNN outperform RNN although the improvement is not as much as in present phrase prediction. It indicates that the copy mechanism is very helpful for predicting absent phrases. We can also see that CopyRNN and CorrRNN are comparable in terms of recall, but CorrRNN is better on diversity, proving that our model can address the duplicate issue in keyphrase generation.

Model	**NUS**		**SemEval**		**Krapivin**	
	R@10	**N@10**	**R@10**	**N@10**	**R@10**	**N@10**
RNN	0.050	N/A	0.041	N/A	0.095	N/A
CopyRNN	0.058	0.213	**0.043**	0.228	0.113	0.162
CopyRNN$_F$	0.057	0.216	**0.043**	0.233	0.112	0.164
CorrRNN$_C$	**0.064**	0.215	0.041	0.231	**0.121**	**0.168**
CorrRNN$_R$	0.054	0.223	0.041	**0.250**	0.103	0.163
CorrRNN	0.059	**0.229**	0.041	0.243	0.108	0.166

Table 4: The recall and α-NDCG performance on absent phrase prediction.

5.3 Generalization Ability

As described above, CorrRNN performed well on scientific publications. In this part, we construct our experiments on news domain to see if the proposed model works when transferring to a different domain with unfamiliar texts. We adopted the popular news article dataset: DUC-2001 (Wan and Xiao, 2008) for our experiments. The dataset contains 308 news articles with 2488 manually assigned keyphrases, and each article consists of 740 words on average, which is completely different from the datasets we used above (see Table 5).

Dataset	**Length**	**Dataset**	**Length**
NUS	219	SemEval	235
Krapivin	184	DUC-2001	740

Table 5: The average text length of test datasets.

We directly applied CorrRNN, which is trained on scientific publications, on predicting phrases for news articles without any adaptive adjustment. Experiment results from (Hasan and Ng, 2010), (Meng et al., 2017) and our experiments are shown in Table 6, from which we can see that the proposed model CorrRNN can extract a considerable portion of keyphrases correctly from unfamiliar texts. It outperforms TextRank (Mihalcea and Tarau, 2004), KeyCluster (Liu et al., 2009), TopicRank (Bougouin et al., 2013) and CopyRNN (Meng et al., 2017), but it falls behind the other three baselines because the test domain changes. The model should perform better if it is trained on news dataset.

When transferring to news domain, the vocabulary changes a lot, more unknown words occur, and the correlation also may not applicable, the model can still capture positional and syntactic features within the text to predict phrases despite the different text type and length. The experiment verifies the generalization ability of our model, thus we have good reasons to believe that our model has a great potential to be generalized to more domains after sufficient training.

Model	**F1@10**	**Model**	**F1@10**
Tf-idf	0.270	TopicRank	0.154
TextRank	0.097	KeyCluster	0.140
SingleRank	0.256	CopyRNN	0.164
ExpandRank	0.269	CorrRNN	0.173

Table 6: Performance on DUC-2001. CopyRNN and CorrRNN are supervised, and they are trained on scientific publications but evaluated on news.

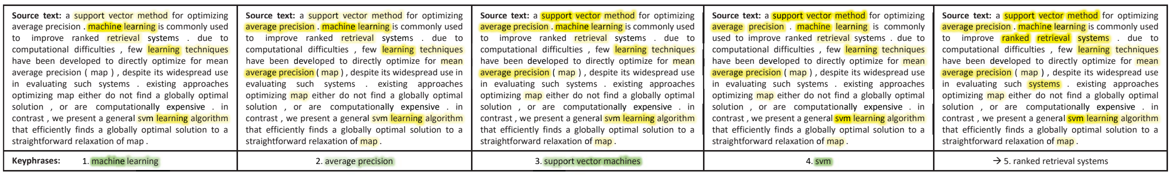

Source text: a support vector method for optimizing average precision . machine learning is commonly used to improve ranked retrieval systems . due to computational difficulties , few learning techniques have been developed to directly optimize for mean average precision (map) , despite its widespread use in evaluating such systems . existing approaches optimizing map either do not find a globally optimal solution , or are computationally expensive . in contrast , we present a general svm learning algorithm that efficiently finds a globally optimal solution to a straightforward relaxation of map .	**Source text:** a support vector method for optimizing average precision . machine learning is commonly used to improve ranked retrieval systems . due to computational difficulties , few learning techniques have been developed to directly optimize for mean average precision (map) , despite its widespread use in evaluating such systems . existing approaches optimizing map either do not find a globally optimal solution , or are computationally expensive . in contrast , we present a general svm learning algorithm that efficiently finds a globally optimal solution to a straightforward relaxation of map .	**Source text:** a support vector method for optimizing average precision . machine learning is commonly used to improve ranked retrieval systems . due to computational difficulties , few learning techniques have been developed to directly optimize for mean average precision (map) , despite its widespread use in evaluating such systems . existing approaches optimizing map either do not find a globally optimal solution , or are computationally expensive . in contrast , we present a general svm learning algorithm that efficiently finds a globally optimal solution to a straightforward relaxation of map .	**Source text:** a support vector method for optimizing average precision . machine learning is commonly used to improve ranked retrieval systems . due to computational difficulties , few learning techniques have been developed to directly optimize for mean average precision (map) , despite its widespread use in evaluating such systems . existing approaches optimizing map either do not find a globally optimal solution , or are computationally expensive . in contrast , we present a general svm learning algorithm that efficiently finds a globally optimal solution to a straightforward relaxation of map .	**Source text:** a support vector method for optimizing average precision . machine learning is commonly used to improve ranked retrieval systems . due to computational difficulties , few learning techniques have been developed to directly optimize for mean average precision (map) , despite its widespread use in evaluating such systems . existing approaches optimizing map either do not find a globally optimal solution , or are computationally expensive . in contrast , we present a general svm learning algorithm that efficiently finds a globally optimal solution to a straightforward relaxation of map .
Keyphrases: 1. machine learning	2. average precision	3. support vector machines	4. svm	→ 5. ranked retrieval systems

Figure 2: Visualization, deeper shading denotes higher value. Note that yellow shading and green shading indicate coverage vector and review attention respectively.

5.4 Discussion

5.4.1 Model Ablation

We investigate the effect of coverage mechanism and review mechanism in our model with CorrRNN$_C$ and CorrRNN$_R$ respectively, shown in Tables 2, 3 and 4. It is clear that both the coverage mechanism and review mechanism are helpful for improving the coverage and diversity of predicted phrases. We note the inconsistency of ablate models in our experiments. First, no ablate model achieves the best performance on all of the test datasets, the full model CorrRNN gets better perfermance on present phrase prediction, while CorrRNN$_C$ seems better than the others in absent phrase prediction. As the present phrases are the majority, the full model CorrRNN can achieve best overall performance in actual use. Second, proposed models perform better on dataset NUS and SemEval than Kravipin. This may be due to the difference of assignment quality among test datasets, keyphrase assignments with higher coverage and higher diversity benefiting more from our models.

5.4.2 Visualization

In Figure 2, we visualize the coverage vector and review attention with an example to further clarify how our model works. Due to space limitation, we only visualize top5 phrases in the example, they are already enough to support our analysis. For coverage vector, we can see that source attention transfers along the changes of coverage vector. At the first, only relevant words of ”machine learning” are attended. After that, the coverage vector informs attention mechanism to attend other positions instead of repetitive attention, that promotes the generation of later phrases like ”average precision”. After the last one being generated, it is clear that coverage vector basically covers all topics of the source document, including ”machine learning”, ”mean average precision”, ”svm” and ”ranked retrieval systems”. As for the review attention, it’s clear that contextual information of all previous phrases are attended for generating the last phrase ”ranked retrieval systems”, which verifies that the review mechanism helps to alleviate duplication and ensure coherence of results.

5.4.3 Comparison with Heuristic Rules

We design the baseline CopyRNN$_F$ with post-processing to explore whether heuristic rules can alleviate duplication and coverage issues. It is clear from Tables 2, 3 and 4 that the experiment results are negative. Compared to our model CorrRNN, heuristic rules can’t address duplication and coverage issues fundamentally. We offer two explanations for these observations. Firstly, heuristic rules can only handle those phrases which have already been generated in results, that shows no help for enabling phrases to cover more topics in source text. Secondly, although duplication can be reduced by heuristic rules forcibly, the remaining phrases are not guaranteed to be correct, thus it hurts the accuracy badly.

5.4.4 Complexity Analysis

According to our observations during experiment phase, the CopyRNN (Meng et al., 2017) model has 78835750 network parameters, while CorrRNN owns 94886750 parameters because of the incorporation of coverage mechanism (few) and review mechanism (most). However, benefiting from the consideration of one-to-many relationship, and correlation constraints among keyphrases, our model CorrRNN not only achieves better performance but also converges faster than CopyRNN.

5.4.5 Case Study

As shown in Table 7, we compare the phrases generated by CorrRNN and CopyRNN on an example article. Compared to CopyRNN, CorrRNN generates one more correct present phrase ”voip”

Title: Deployment issues of a voip conferencing system in a virtual conferencing environment. **Abstract**: Real time services have been supported by and large on circuitswitched networks. Recent trends favour services ported on packet switched networks. For audio conferencing, we need to consider many issues scalability, quality of the conference application, floor control and load on the clients servers to name a few. In this paper, we describe an audio service framework designed to provide a virtual conferencing environment (vce). The system is designed to accommodate a large number of end users speaking at the same time and spread across the internet. The framework is based on conference servers $\langle DIGIT \rangle$, which facilitate the audio handling, while we exploit the sip capabilities for signaling purposes. Client selection is based on a recent quantifier called loudness number that helps mimic a physical face to face conference. We deal with deployment issues of the proposed solution both in terms of scalability and interactivity, while explaining the techniques we use to reduce the traffic. We have implemented a conference server (cs) application on a campus wide network at our institute.
Present Phrase: **CopyRNN:** deployment; virtual conferencing; real time; distributed systems; **virtual conferencing environment**; client server; **conference server**; distributed applications; audio conferencing; floor control; **CorrRNN:** **voip**; virtual conferencing; voip conferencing; audio conferencing; audio service; real time services; real time; distributed systems; **conference server**; **virtual conferencing environment**;
Absent Phrase: **CopyRNN:** quality of service; distributed conferencing; virtual environments; internet conferencing; conference conferencing; load balancing; packet conferencing; real time systems; distributed computing; virtual server; **CorrRNN:** real time systems; real time voip; voip service; **real time audio**; wireless networks; conference conferencing; real time communications; packet conferencing; audio communication; quality of service;

Table 7: Top10 phrases generated by CopyRNN and CorrRNN. Phrases in bold are correct, and phrases underlined are duplicate.

and one more correct absent phrase ”real time audio” respectively, which covers two important topics, while CopyRNN loses these key points. Moreover, four ”conferencing (noun)” phrases are generated by CopyRNN, including ”distributed conferencing”, ”internet conferencing”, ”conference conferencing” and ”packet conferencing”, which hinders readers from obtaining more information, while CorrRNN only has two.

6 Conclusion and Future Work

In this paper, we propose a new Seq2Seq architecture that models correlation among multiple keyphrases in an end-to-end fashion by incorporating a coverage mechanism and a review mechanism. Comprehensive empirical studies demonstrate that our model can alleviate duplication and coverage issues effectively and improve diversity and coverage for keyphrase generation. To the best of our knowledge, this is the first use of encoder-decoder model for keyphrase generation in an one-to-many way. Our future work will focus on two areas: investigation on multi-document keyphrase generation, and incorporation of structure or syntax information in keyphrase generation.

Acknowledgements

We appreciate valuable comments from anonymous reviewers. Yu Wu is supported by Microsoft Fellowship Scholarship and AdeptMind Scholarship. This work was supported in part by the Natural Science Foundation of China (Grand Nos. U1636211,61672081,61370126), and Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001) and National Key R&D Program of China (No.2016QY04W0802).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. volume abs/1409.0473.

Adrien Bougouin, Florian Boudin, and Beatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *IJCNLP*, pages 543–551.

Adrien Bougouin, Florian Boudin, and Beatrice Daille. 2016. Keyphrase annotation with graph co-ranking. In *the 26th COLING*, pages 2945–2955.

Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Buttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. pages 659–666. SIGIR.

Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-specific keyphrase extraction.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *28th AAAI*, pages 1629–1635. AAAI Press.

Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multi-theme documents. In *18th WWW*, pages 661–670. ACM.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Maryam Habibi and Andrei Popescu-Belis. 2013. Diverse keyword extraction from conversations. In *ACL (2)*, pages 651–657.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *the 23rd COLING*, pages 365–373. Association for Computational Linguistics.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *EMNLP 2003*, pages 216–223.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *5th SemEval*, pages 21–26. Association for Computational Linguistics.

Mikalai Krapivin, Aliaksandr Autayeu, and Maurizio Marchese. 2008. Large dataset for keyphrases extraction. In *Technical Report DISI-09-055*.

Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. 2016. Unsupervised keyphrase extraction: Introducing new kinds of words to keyphrases. pages 665–671. Springer International Publishing, Cham.

Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *SIGMOD15*.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *15th CoNLL*, pages 135–144. Association for Computational Linguistics.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *EMNLP 2010*, pages 366–376. Association for Computational Linguistics.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *EMNLP 2009*, volume 1–1, pages 257–266. Association for Computational Linguistics.

Olena Medelyan, Eibe Frank, and Ian H Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *EMNLP 2009*, volume 3–3, pages 1318–1327. Association for Computational Linguistics.

Olena Medelyan, Ian H Witten, and David Milne. 2008. Topic indexing with wikipedia. In *Proceedings of the AAAI WikiAI workshop*, volume 1, pages 19–24.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *55th ACL*, volume 1, pages 582–592. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. pages 404–411. EMNLP.

Weijian Ni, Tong Liu, and Qingtian Zeng. 2012. Extracting keyphrase set with high diversity and coverage using structural svm. In *14th APWeb*, pages 122–133. Springer.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *55th ACL*, pages 1073–1083. Association for Computational Linguistics.

Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, and Jiawei Han. 2017. Automated phrase mining from massive text corpora. *CoRR*, abs/1702.04457.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Takashi Tomokiyo and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *ACL 2003*, volume 18, pages 33–40. Association for Computational Linguistics.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation.

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. volume abs/1610.02424.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. AAAI.

Minmei Wang, Bo Zhao, and Yihua Huang. 2016. Ptr: Phrase-based topical ranking for automatic keyphrase extraction in scientific publications. pages 120–128. Springer International Publishing, Cham.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 1999. Kea: Practical automatic keyphrase extraction. pages 254–255. ACM.

Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2015. Mining query subtopics from questions in community question answering. In *AAAI*, pages 339–345.

Fan Zhang, Lian'en Huang, and Bo Peng. 2013. Wordtopic-multirank : A new method for automatic keyphrase extraction. In *IJCNLP*, pages 10–18.

Closed-Book Training to Improve Summarization Encoder Memory

Yichen Jiang and **Mohit Bansal**
UNC Chapel Hill
{yichenj, mbansal}@cs.unc.edu

Abstract

A good neural sequence-to-sequence summarization model should have a strong encoder that can distill and memorize the important information from long input texts so that the decoder can generate salient summaries based on the encoder's memory. In this paper, we aim to improve the memorization capabilities of the encoder of a pointer-generator model by adding an additional 'closed-book' decoder without attention and pointer mechanisms. Such a decoder forces the encoder to be more selective in the information encoded in its memory state because the decoder can't rely on the extra information provided by the attention and possibly copy modules, and hence improves the entire model. On the *CNN/Daily Mail* dataset, our 2-decoder model outperforms the baseline significantly in terms of ROUGE and METEOR metrics, for both cross-entropy and reinforced setups (and on human evaluation). Moreover, our model also achieves higher scores in a test-only DUC-2002 generalizability setup. We further present a memory ability test, two saliency metrics, as well as several sanity-check ablations (based on fixed-encoder, gradient-flow cut, and model capacity) to prove that the encoder of our 2-decoder model does in fact learn stronger memory representations than the baseline encoder.

1 Introduction

Text summarization is the task of condensing a long passage to a shorter version that only covers the most salient information from the original text. Extractive summarization models (Jing and McKeown, 2000; Knight and Marcu, 2002; Clarke and Lapata, 2008; Filippova et al., 2015) directly pick words, phrases, and sentences from the source text to form a summary, while an abstractive model generates (samples) words from a fixed-size vocabulary instead of copying from text directly.

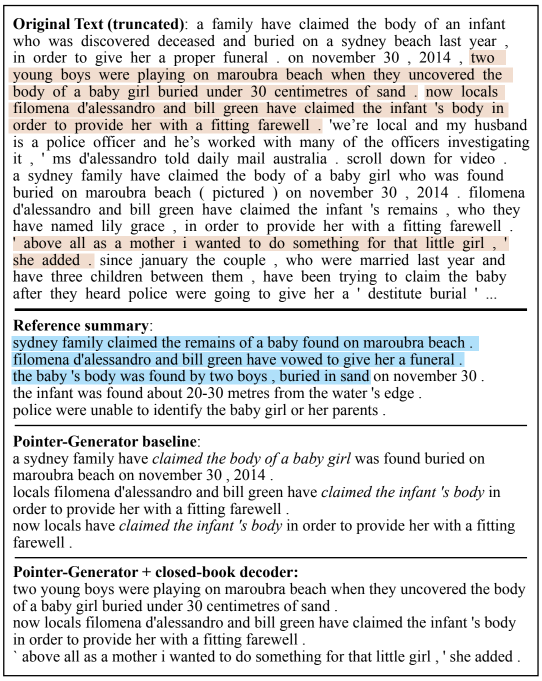

Original Text (truncated): a family have claimed the body of an infant who was discovered deceased and buried on a sydney beach last year , in order to give her a proper funeral . on november 30 , 2014 , two young boys were playing on maroubra beach when they uncovered the body of a baby girl buried under 30 centimetres of sand . now locals filomena d'alessandro and bill green have claimed the infant 's body in order to provide her with a fitting farewell . 'we're local and my husband is a police officer and he's worked with many of the officers investigating it , ' ms d'alessandro told daily mail australia . scroll down for video . a sydney family have claimed the body of a baby girl who was found buried on maroubra beach (pictured) on november 30 , 2014 . filomena d'alessandro and bill green have claimed the infant 's remains , who they have named lily grace , in order to provide her with a fitting farewell . ' above all as a mother i wanted to do something for that little girl , ' she added . since january the couple , who were married last year and have three children between them , have been trying to claim the baby after they heard police were going to give her a ' destitute burial ' ...

Reference summary:
sydney family claimed the remains of a baby found on maroubra beach .
filomena d'alessandro and bill green have vowed to give her a funeral .
the baby 's body was found by two boys , buried in sand on november 30 .
the infant was found about 20-30 metres from the water 's edge .
police were unable to identify the baby girl or her parents .

Pointer-Generator baseline:
a sydney family have *claimed the body of a baby girl* was found buried on maroubra beach on november 30 , 2014 .
locals filomena d'alessandro and bill green have *claimed the infant 's body* in order to provide her with a fitting farewell .
now locals have *claimed the infant 's body* in order to provide her with a fitting farewell .

Pointer-Generator + closed-book decoder:
two young boys were playing on maroubra beach when they uncovered the body of a baby girl buried under 30 centimetres of sand .
now locals filomena d'alessandro and bill green have claimed the infant 's body in order to provide her with a fitting farewell .
` above all as a mother i wanted to do something for that little girl , ' she added .

Figure 1: Baseline model repeats itself twice (italic), and fails to find all salient information (highlighted in red in the original text) from the source text that is covered by our 2-decoder model. The summary generated by our 2-decoder model also recovers most of the information mentioned in the reference summary (highlighted in blue in the reference summary).

The last few years have seen significant progress on both extractive and abstractive approaches, of which a large number of studies are fueled by neural sequence-to-sequence models (Sutskever et al., 2014). One popular formulation of such models is an RNN/LSTM encoder that encodes the source passage to a fixed-size memory-state vector, and another RNN/LSTM decoder that generates the summary from this memory state. This paradigm is enhanced by attention mechanism (Bahdanau et al., 2015) and pointer network (Vinyals et al., 2015), such that the decoder can refer to (and weigh) all the encod-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4067–4077
Brussels, Belgium, October 31 - November 4, 2018.

ing steps' hidden states or directly copy words from the source text, instead of relying solely on encoder's final memory state for all information about the source passage. Recent studies (Rush et al., 2015; Nallapati et al., 2016; Chopra et al., 2016; Zeng et al., 2016; Gu et al., 2016b; Gulcehre et al., 2016; See et al., 2017) have demonstrated success with such seq-attention-seq and pointer models in summarization tasks.

While the advantage of attention and pointer models compared to vanilla sequence-to-sequence models in summarization is well supported by previous studies, these models still struggle to find the most salient information in the source text when generating summaries. This is because summarization, being different from other text-to-text generation tasks (where there is an almost one-to-one correspondence between input and output words, e.g., machine translation), requires the sequence-attention-sequence model to additionally decide where to attend and where to ignore, thus demanding a strong encoder that can determine the importance of different words, phrases, and sentences and flexibly encode salient information in its memory state. To this end, we propose a novel 2-decoder architecture by adding another 'closed book' decoder without attention layer to a popular pointer-generator baseline, such that the 'closed book' decoder and pointer decoder share an encoder. We argue that this additional 'closed book' decoder encourages the encoder to be better at memorizing salient information from the source passage, and hence strengthen the entire model. We provide both intuition and evidence for this argument in the following paragraphs.

Consider the following case. Two students are learning to do summarization from scratch. During training, both students can first scan through the passage once (encoder's pass). Then student A is allowed to constantly look back (attention) at the passage when writing the summary (similar to a pointer-generator model), while student B has to occasionally write the summary without looking back (similar to our 2-decoder model with a non-attention/copy decoder). During the final test, both students can look at the passage while writing summaries. We argue that student B will write more salient summaries in the test because s/he learns to better distill and memorize important information in the first scan/pass by not looking back at the passage in training.

In terms of back-propagation intuition, during the training of a seq-attention-seq model (e.g., See et al. (2017)), most gradients are back-propagated from the decoder to the encoder's hidden states through the attention layer. This encourages the encoder to correctly encode salient words at the corresponding encoding steps, but does make sure that this information is not forgotten (overwritten in the memory state) by the encoder afterward. However, for a plain LSTM (closed-book) decoder without attention, its generated gradient flow is back-propagated to the encoder through the memory state, which is the only connection between itself and the encoder, and this, therefore, encourages the encoder to encode only the salient, important information in its memory state. Hence, to achieve this desired effect, we jointly train the two decoders, which share one encoder, by optimizing the weighted sum of their losses. This approximates the training routine of student B because *the sole encoder has to perform well for both decoders*. During inference, we only employ the pointer decoder due to its copying advantage over the closed-book decoder, similar to the situation of student B being able to refer back to the passage during the test for best performance (but is still trained hard to do well in both situations). Fig. 1 shows an example of our 2-decoder summarizer generating a summary that covers the original passage with more saliency than the baseline model.

Empirically, we test our 2-decoder architecture on the *CNN/Daily Mail* dataset (Hermann et al., 2015; Nallapati et al., 2016), and our model surpasses the strong pointer-generator baseline significantly on both ROUGE (Lin, 2004) and METEOR (Denkowski and Lavie, 2014) metrics, as well as based on human evaluation. This holds true both for a cross-entropy baseline as well as a stronger, policy-gradient based reinforcement learning setup (Williams, 1992). Moreover, our 2-decoder models (both cross-entropy and reinforced) also achieve reasonable improvements on a test-only generalizability/transfer setup on the DUC-2002 dataset.

We further present a series of numeric and qualitative analysis to understand whether the improvements in these automatic metric scores are in fact due to the enhanced memory and saliency strengths of our encoder. First, by evaluating the representation power of the encoder's final memory state after reading long passages (w.r.t.

the memory state after reading ground-truth summaries) via a cosine-similarity test, we prove that our 2-decoder model indeed has a stronger encoder with better memory ability. Next, we conduct three sets of ablation studies based on fixed-encoder, gradient-flow cut, and model capacity to show that the stronger encoder is the reason behind the significant improvements in ROUGE and METEOR scores. Finally, we show that summaries generated by our 2-decoder model are qualitatively better than baseline summaries as the former achieved higher scores on two saliency metrics (based on cloze-Q&A blanks and a keyword classifier) than the baseline summaries, while maintaining similar length and better avoiding repetitions. This directly demonstrates our 2-decoder model's enhanced ability to memorize and recover important information from the input document, which is our main contribution in this paper.

2 Related Work

Extractive and Abstractive Summarization: Early models for automatic text summarization were usually extractive (Jing and McKeown, 2000; Knight and Marcu, 2002; Clarke and Lapata, 2008; Filippova et al., 2015). For abstractive summarization, different early non-neural approaches were applied, based on graphs (Giannakopoulos, 2009; Ganesan et al., 2010), discourse trees (Gerani et al., 2014), syntactic parse trees (Cheung and Penn, 2014; Wang et al., 2013), and a combination of linguistic compression and topic detection (Zajic et al., 2004). Recent neural-network models have tackled abstractive summarization using methods such as hierarchical encoders and attention, coverage, and distraction (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Chen et al., 2016; Takase et al., 2016) as well as various initial large-scale, short-length summarization datasets like DUC-2004 and Gigaword. Nallapati et al. (2016) adapted the *CNN/Daily Mail* (Hermann et al., 2015) dataset for long-text summarization, and provided an abstractive baseline using attentional sequence-to-sequence model.

Pointer Network for Summarization: Pointer networks (Vinyals et al., 2015) are useful for summarization models because summaries often need to copy/contain a large number of words that have appeared in the source text. This provides the advantages of both *extractive* and *abstractive* approaches, and usually includes a gating function to model the distribution for the extended vocabulary including the pre-set vocabulary and words from the source text (Zeng et al., 2016; Nallapati et al., 2016; Gu et al., 2016b; Gulcehre et al., 2016; Miao and Blunsom, 2016). See et al. (2017) used a soft gate to control model's behavior of copying versus generating. They further applied coverage mechanism and achieved the state-of-the-art results on *CNN/Daily Mail* dataset.

Memory Enhancement: Some recent works (Wang et al., 2016; Xiong et al., 2018; Gu et al., 2016a) have studied enhancing the memory capacity of sequence-to-sequence models. They studied this problem in Neural Machine Translation by keeping an external memory state analogous to data in the Von Neumann architecture, while the instructions are represented by the sequence-to-sequence model. Our work is novel in that we aim to improve the internal long-term memory of the encoder LSTM by adding a closed-book decoder that has no attention layer, yielding a more efficient internal memory that encodes only important information from the source text, which is crucial for the task of long-document summarization.

Reinforcement Learning: Teacher forcing style maximum likelihood training suffers from exposure bias (Bengio et al., 2015), so recent works instead apply reinforcement learning style policy gradient algorithms (REINFORCE (Williams, 1992)) to directly optimize on metric scores (Henß et al., 2015; Paulus et al., 2018). Reinforced models that employ this method have achieved good results in a number of tasks including image captioning (Liu et al., 2017; Ranzato et al., 2016), machine translation (Bahdanau et al., 2016; Norouzi et al., 2016), and text summarization (Ranzato et al., 2016; Paulus et al., 2018).

3 Models

3.1 Pointer-Generator Baseline

The pointer-generator network proposed in See et al. (2017) can be seen as a hybrid of *extractive* and *abstractive* summarization models. At each decoding step, the model can either sample a word from its vocabulary, or copy a word directly from the source passage. This is enabled by the attention mechanism (Bahdanau et al., 2015), which includes a distribution a_i over all encoding steps, and a context vector c_t that is the weighted sum of encoder's hidden states. The attention mechanism

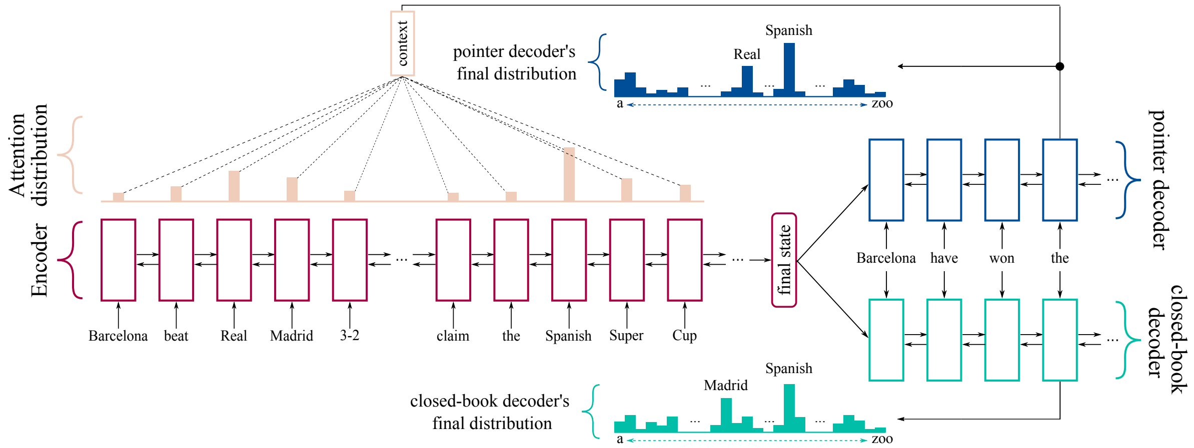

Figure 2: Our 2-decoder summarization model with a pointer decoder and a closed-book decoder, both sharing a single encoder (this is during training; next, at inference time, we only employ the memory-enhanced encoder and the pointer decoder).

is modeled as:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{attn})$$
$$a_i^t = \mathrm{softmax}(e_i^t); \quad c_t = \sum_i a_i^t h_i \tag{1}$$

where v, W_h, W_s, and b_{attn} are learnable parameters. h_i is encoder's hidden state at i^{th} encoding step, and s_t is decoder's hidden state at t^{th} decoding step. The distribution a_i^t can be seen as the amount of attention at decode step t towards the i^{th} encoder state. Therefore, the context vector c_t is the sum of the encoder's hidden states weighted by attention distribution a^t.

At each decoding step, the previous context vector c_{t-1} is concatenated with current input x_t, and fed through a non-linear recurrent function along with the previous hidden state s_{t-1} to produce the new hidden state s_t. The context vector c_t is then calculated according to Eqn. 1 and concatenated with the decoder state s_t to produce the logits for the vocabulary distribution P_{vocab} at decode step t: $P_{vocab}^t = \mathrm{softmax}(V_2(V_1[s_t, c_t]+b_1)+b_2)$, where V_1, V_2, b_1, b_2 are learnable parameters. To enable copying out-of-vocabulary words from source text, a pointer similar to Vinyals et al. (2015) is built upon the attention distribution and controlled by the generation probability p_{gen}:

$$p_{gen}^t = \sigma(U_c c_t + U_s s_t + U_x x_t + b_{ptr})$$
$$P_{attn}^t(w) = p_{gen}^t P_{vocab}^t(w) + (1 - p_{gen}^t) \sum_{i:w_i=w} a_i^t$$

where U_c, U_s, U_x, and b_{ptr} are learnable parameters. x_t and s_t are the input token and decoder's state at t^{th} decoding step. σ is the sigmoid function. We can see p_{gen} as a soft gate that controls the model's behavior of copying from text with attention distribution a_i^t versus sampling from vocabulary with generation distribution P_{vocab}^t.

3.2 Closed-Book Decoder

As shown in Eqn. 1, the attention distribution a_i depends on decoder's hidden state s_t, which is derived from decoder's memory state c_t. If c_t does not encode salient information from the source text or encodes too much unimportant information, the decoder will have a hard time to locate relevant encoder states with attention. However, as explained in the introduction, most gradients are back-propagated through attention layer to the encoder's hidden state h_t, not directly to the final memory state, and thus provide little incentive for the encoder to memorize salient information in c_t.

Therefore, to enhance encoder's memory, we add a closed-book decoder, which is a unidirectional LSTM decoder without attention/pointer layer. The two decoders share a single encoder and word-embedding matrix, while out-of-vocabulary (OOV) words are simply represented as [UNK] for the closed-book decoder. The entire 2-decoder model is represented in Fig. 2. During training, we optimize the weighted sum of negative log likelihoods from the two decoders:

$$\mathcal{L}_{XE} = \frac{1}{T} \sum_{t=1}^{T} -\left((1-\gamma) \log P_{attn}^t(w|x_{1:t}) + \gamma \log P_{cbdec}^t(w|x_{1:t})\right) \tag{2}$$

where P_{cbdec} is the generation probability from the closed-book decoder. The mix ratio γ is tuned on the validation set.

	ROUGE			MTR
	1	2	L	Full
PREVIOUS WORKS				
*(Nallapati16)	35.46	13.30	32.65	
pg (See17)	36.44	15.66	33.42	16.65
OUR MODELS				
pg (baseline)	36.70	15.71	33.74	16.94
pg + cbdec	38.21	16.45	34.70	18.37
RL + pg	37.02	15.79	34.00	17.55
RL + pg + cbdec	**38.58**	**16.57**	**35.03**	**18.86**

Table 1: ROUGE F1 and METEOR scores (*non-coverage*) on CNN/Daily Mail test set of previous works and our models. 'pg' is the pointer-generator baseline, and 'pg + cbdec' is our 2-decoder model with closed-book decoder(cbdec). The model marked with $\star$ is trained and evaluated on the anonymized version of the data.

	ROUGE			MTR
	1	2	L	Full
PREVIOUS WORKS				
pg (See17)	39.53	17.28	36.38	18.72
RL* (Paulus17)	39.87	15.82	36.90	
OUR MODELS				
pg (baseline)	39.22	17.02	35.95	18.70
pg + cbdec	40.05	17.66	36.73	19.48
RL + pg	39.59	17.18	36.16	19.70
RL + pg + cbdec	**40.66**	**17.87**	**37.06**	**20.51**

Table 2: ROUGE F1 and METEOR scores (*with-coverage*) on the *CNN/Daily Mail* test set. Coverage mechanism (See et al., 2017) is used in all models except the RL model (Paulus et al., 2018). The model marked with $\star$ is trained and evaluated on the anonymized version of the data.

	ROUGE			MTR
	1	2	L	Full
pg (See17)	37.22	15.78	33.90	13.69
pg (baseline)	37.15	15.68	33.92	13.65
pg + cbdec	37.59	16.84	34.43	13.82
RL + pg	39.92	16.71	36.13	15.12
RL + pg + cbdec	**41.48**	**18.69**	**37.71**	**15.88**

Table 3: ROUGE F1 and METEOR scores on DUC-2002 (test-only transfer setup).

3.3 Reinforcement Learning

In the reinforcement learning setting, our summarization model is the policy network that generates words to form a summary. Following Paulus et al. (2018), we use a self-critical policy gradient training algorithm (Rennie et al., 2016; Williams, 1992) for both our baseline and 2-decoder model. For each passage, we sample a summary $y^s = w^s_{1:T+1}$, and greedily generate a summary $\hat{y} = \hat{w}_{1:T+1}$ by selecting the word with the highest probability at each step. Then these two summaries are fed to a reward function r, which is the ROUGE-L scores in our case. The RL loss function is:

$$\mathcal{L}_{RL} = \frac{1}{T}\sum_{t=1}^{T}(r(\hat{y}) - r(y^s)) \log P^t_{attn}(w^s_{t+1}|w^s_{1:t}) \tag{3}$$

where the reward for the greedily-generated summary ($r(\hat{y})$) acts as a baseline to reduce variance. We train our reinforced model using the mixture of Eqn. 3 and Eqn. 2, since Paulus et al. (2018) showed that a pure RL objective would lead to summaries that receive high rewards but are not fluent. The final mixed loss function for RL is: $\mathcal{L}_{XE+RL} = \lambda\mathcal{L}_{RL} + (1-\lambda)\mathcal{L}_{XE}$, where the value of λ is tuned on the validation set.

4 Experimental Setup

We evaluate our models mainly on *CNN/Daily Mail* dataset (Hermann et al., 2015; Nallapati et al., 2016), which is a large-scale, long-paragraph summarization dataset. It has online news articles (781 tokens or ~40 sentences on average) with paired human-generated summaries (56 tokens or 3.75 sentences on average). The entire dataset has 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs. We use the same version of data as See et al. (2017), which is the original text with no preprocessing to replace named entities. We also use DUC-2002, which is also a long-paragraph summarization dataset of news articles. This dataset has 567 articles and 1~2 summaries per article.

All the training details (e.g., vocabulary size, RNN dimension, optimizer, batch size, learning rate, etc.) are provided in the supplementary materials.

5 Results

We first report our evaluation results on *CNN/Daily Mail* dataset. As shown in Table 1, our 2-decoder model achieves statistically significant improvements[1] upon the pointer-generator baseline (pg), with +1.51, +0.74, and +0.96 points advantage in ROUGE-1, ROUGE-2 and ROUGE-L (Lin, 2004), and +1.43 points advantage in METEOR (Denkowski and Lavie, 2014). In the reinforced setting, our 2-decoder model still maintains significant ($p < 0.001$)

---

[1] Our improvements in Table 1 are statistically significant with $p < 0.001$ (using bootstrapped randomization test with 100k samples (Efron and Tibshirani, 1994)) and have a 95% ROUGE-significance interval of at most ±0.25.

Reference summary:
mitchell moffit and greg brown from asapscience present theories.
different personality traits can vary according to expectations of parents.
beyoncé, hillary clinton and j. k. rowling are all oldest children.

Pointer-Gen baseline:
the kardashians are a strong example of a large celebrity family where the siblings share very different personality traits.
on asapscience on youtube, the pair discuss how being the first, middle, youngest, or an only child affects us.

Pointer-Gen + closed-book decoder:
the kardashians are a strong example of a large celebrity family where the siblings share very different personality traits.
on asapscience on youtube , the pair discuss how being the first, middle, youngest, or an only child affects us.
the personality traits are also supposedly affected by whether parents have high expectations and how strict they were.

Figure 3: The summary generated by our 2-decoder model covers salient information (highlighted in red) mentioned in the reference summary, which is not presented in the baseline summary.

Model	Score
2-Decoder Wins	49
Pointer-Generator Wins	31
Non-distinguishable	20

Table 4: Human evaluation for our 2-decoder model versus the pointer-generator baseline.

advantage in all metrics over the pointer-generator baseline.

We further add the coverage mechanism as in See et al. (2017) to both baseline and 2-decoder model, and our 2-decoder model (pg + cbdec) again receives significantly higher[2] scores than the original pointer-generator (pg) from See et al. (2017) and our own pg baseline, in all ROUGE and METEOR metrics (see Table 2). In the reinforced setting, our 2-decoder model (RL + pg + cbdec) outperforms our strong RL baseline (RL + pg) by a considerable margin (stat. significance of $p < 0.001$). Fig. 1 and Fig. 3 show two examples of our 2-decoder model generating summaries that cover more salient information than those generated by the pointer-generator baseline (see supplementary materials for more example summaries).

We also evaluate our 2-decoder model with coverage on the DUC-2002 test-only generalizability/transfer setup by decoding the entire dataset with our models pre-trained on *CNN/Daily Mail*, again achieving decent improvements (shown in Table 3) over the single-decoder baseline as well as See et al. (2017), in both a cross-entropy and a reinforcement learning setup.

	similarity
pg (baseline)	0.817
pg + cbdec ($\gamma = \frac{1}{2}$)	0.869
pg + cbdec ($\gamma = \frac{2}{3}$)	**0.889**
pg + cbdec ($\gamma = \frac{5}{6}$)	0.872
pg + cbdec ($\gamma = \frac{10}{11}$)	0.860

Table 5: Cosine-similarity between memory states after two forward passes.

5.1 Human Evaluation

We also conducted a small-scale human evaluation study by randomly selecting 100 samples from the *CNN/DM* test set and then asking human annotators to rank the baseline summaries versus the 2-decoder's summaries (randomly shuffled to anonymize model identity) according to an overall score based on readability (grammar, fluency, coherence) and relevance (saliency, redundancy, correctness). As shown in Table 4, our 2-decoder model outperforms the pointer-generator baseline (stat. significance of $p < 0.03$).

6 Analysis

In this section, we present a series of analysis and tests in order to understand the improvements of the 2-decoder models reported in the previous section, and to prove that it fulfills our intuition that the closed-book decoder improves the encoder's ability to encode salient information in the memory state.

6.1 Memory Similarity Test

To verify our argument that the closed-book decoder improves the encoder's memory ability, we design a test to numerically evaluate the representation power of encoder's final memory state. We perform two forward passes for each encoder (2-decoder versus pointer-generator baseline). For the first pass, we feed the entire article to the encoder and collect the final memory state; for the second pass we feed the ground-truth summary to the encoder and collect the final memory state. Then we calculate the cosine similarity between these two memory-state vectors. For an optimal summarization model, its encoder's memory state after reading the entire article should be highly similar to its memory state after reading the ground truth summary (which contains all the important information), because this shows that when reading a long passage, the model is only encoding important information in its memory and

---

[2] All our improvements in Table 2 are statistically significant with $p < 0.001$, and have a 95% ROUGE-significance interval of at most ±0.25.

forgets the unimportant information. The results in Table 5 show that the encoder of our 2-decoder model achieves significantly ($p < 0.001$) higher article-summary similarity score than the encoder of a pointer-generator baseline. This observation verifies our hypothesis that the closed-book decoder can improve the memory ability of the encoder.

6.2 Ablation Studies and Sanity Check

Fixed-Encoder Ablation: Next, we conduct an ablation study in order to prove the qualitative superiority of our 2-decoder model's encoder to the baseline encoder. To do this, we train two pointer-generators with randomly initialized decoders and word embeddings. For the first model, we restore the pre-trained encoder from our pointer-generator baseline; for the second model, we restore the pre-trained encoder from our 2-decoder model. We then fix the encoder's parameters for both models during the training, only updating the embeddings and decoders with gradient descent. As shown in the upper half of Table 6, the pointer-generator with our 2-decoder model's encoder receives significantly higher ($p < 0.001$) scores in ROUGE than the pointer-generator with baseline's encoder. Since these two models have the exact same structure with only the encoders initialized according to different pre-trained models, the significant improvements in metric scores suggest that our 2-decoder model does have a stronger encoder than the pointer-generator baseline.

Gradient-Flow-Cut Ablation: We further design another ablation test to identify how the gradients from the closed-book decoder influence the entire model during training. Fig. 4 demonstrates the forward pass (solid line) and gradient flow (dashed line) between encoder, decoders, and embeddings in our 2-decoder model. As we can see, the closed-book decoder only depends on the word embeddings and encoder. Therefore it can affect the entire model during training by influencing either the encoder or the word-embedding matrix. When we stop the gradient flow between the encoder and closed-book decoder (① in Fig. 4), and keep the flow between closed-book decoder and embedding matrix (② in Fig. 4), we observe non-significant improvements in ROUGE compared to the baseline. On the other hand, when we stop the gradient flow at ② and keep ①, the improvements are statistically significant ($p < 0.01$) (see the lower half of Table 6). This proves that the gradients back-propagated from closed-book decoder to the encoder can strengthen the entire model, and hence verifies the gradient-flow intuition discussed in introduction (Sec. 1).

	ROUGE		
	1	2	L
FIXED-ENCODER ABLATION			
pg baseline's encoder	37.59	16.27	34.33
2-decoder's encoder	**38.44**	**16.85**	**35.17**
GRADIENT-FLOW-CUT ABLATION			
pg baseline	37.73	16.52	34.49
stop ①	37.72	16.58	34.54
stop ②	**38.35**	**16.79**	**35.13**

Table 6: ROUGE F1 scores of ablation studies, evaluated on *CNN/Daily Mail* validation set.

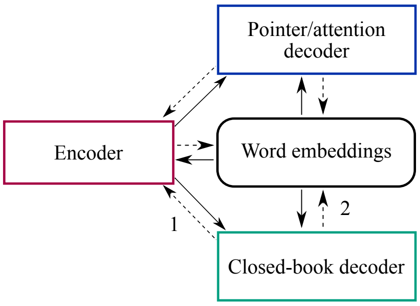

Figure 4: Solid lines represent the forward pass, and dashed lines represent the gradient flow in back-propagation. For the two ablation tests, we stop the gradient at ① and ② respectively.

Model Capacity: To validate and sanity-check that the improvements are the result of the inclusion of our closed-book decoder and not due to some trivial effects of having two decoders or larger model capacity (more parameters), we train a variant of our model with two duplicated (initialized to be different) attention-pointer decoders. We also evaluate a pointer-generator baseline with 2-layer encoder and decoder (pg-2layer) and increase the LSTM hidden dimension and word embedding dimension of the pointer-generator baseline (pg-big) to exceed the total number of parameters of our 2-decoder model (34.5M versus 34.4M parameters). Table 7 shows that neither of these variants can match our 2-decoder model in terms of ROUGE and METEOR scores, and hence proves that the improvements of our model are indeed because of the closed-book decoder rather than due to simply having more parameters.[3]

Mixed-loss Ratio Ablation: We also present eval-

---

[3] It is also important to point out that our model is not a 2-decoder ensemble, because we use only the pointer decoder during inference. Therefore, the number of parameters used for inference is the same as the pointer-generator baseline.

	ROUGE		
	1	2	L
pg baseline	37.73	16.52	34.49
pg + ptrdec	37.66	16.50	34.47
pg-2layer	37.92	16.48	34.62
pg-big	38.03	16.71	34.84
pg + cbdec	**38.87**	**16.93**	**35.38**

Table 7: ROUGE F1 and METEOR scores of sanity check ablations, evaluated on *CNN/DM* validation set.

	ROUGE		
	1	2	L
$\gamma = 0$	37.73	16.52	34.49
$\gamma = 1/2$	38.09	16.71	34.89
$\gamma = 2/3$	**38.87**	**16.93**	**35.38**
$\gamma = 5/6$	38.21	16.69	34.81
$\gamma = 10/11$	37.99	16.39	34.7

Table 8: ROUGE F1 scores on *CNN/DM* validation set, of 2-decoder models with different values of the closed-book-decoder:pointer-decoder mixed loss ratio.

uation results (on the *CNN/Daily Mail* validation set) of our 2-decoder models with different closed-book-decoder:pointer-decoder mixed-loss ratio (γ in Eqn. 2) in Table 8. The model achieves the best ROUGE and METEOR scores at $\gamma = \frac{2}{3}$. Comparing Table 8 with Table 5, we observe a similar trend between the increasing ROUGE/METEOR scores and increasing memory cosine-similarities, which suggests that the performance of a pointer-generator is strongly correlated with the representation power of the encoder's final memory state.

6.3 Saliency and Repetition

Finally, we show that our 2-decoder model can make use of this better encoder memory state to summarize more salient information from the source text, as well as to avoid generating unnecessarily lengthy and repeated sentences besides achieving significant improvements on ROUGE and METEOR metrics.

	saliency 1	saliency 2
pg (See17)	60.4%	27.95%
our pg baseline	59.6%	28.95%
pg + cbdec	62.1%	29.97%
RL + pg	62.5%	30.17%
RL + pg + cbdec	**66.2%**	**31.40%**

Table 9: Saliency scores based on *CNN/Daily Mail* cloze blank-filling task and a keyword-detection approach (Pasunuru and Bansal, 2018). All models in this table are trained with coverage loss.

	3-gram	4-gram	5-gram	sent
pg (baseline)	13.20%	12.32%	11.60%	8.39%
pg + cbdec	**9.66%**	**9.02%**	**8.55%**	**6.72%**

Table 10: Percentage of repeated 3, 4, 5-grams and sentences in generated summaries.

Saliency: To evaluate saliency, we design a keyword-matching test based on the original *CNN/Daily Mail* cloze blank-filling task (Hermann et al., 2015). Each news article in the dataset is marked with a few cloze-blank keywords that represent salient entities, including names, locations, etc. We count the number of keywords that appear in our generated summaries, and found that the output of our best teacher-forcing model (pg+cbdec with coverage) contains 62.1% of those keywords, while the output provided by See et al. (2017) has only 60.4% covered. Our reinforced 2-decoder model (RL + pg + cbdec) further increases this percentage to 66.2%. The full comparison is shown in the first column of Table 9. We also use the saliency metric in Pasunuru and Bansal (2018), which finds important words detected via a keyword classifier (trained on the SQuAD dataset (Rajpurkar et al., 2016)). The results are shown in the second column of Table 9. Both saliency tests again demonstrate our 2-decoder model's ability to memorize important information and address them properly in the generated summary. Fig. 1 and Fig. 3 show two examples of summaries generated by our 2-decoder model compared to baseline summaries.

Summary Length: On average, summaries generated by our 2-decoder model have 66.42 words per summary, while the pointer-generator-baseline summaries have 65.88 words per summary (and the same effect holds true for RL models, where there is less than 1-word difference in average length). This shows that our 2-decoder model is able to achieve higher saliency with similar-length summaries (i.e., it is not capturing more salient content simply by generating longer summaries).

Repetition: We observe that out 2-decoder model can generate summaries that are less redundant compared to the baseline, when both models are not trained with coverage mechanism. Table 10 shows the percentage of repeated n-grams/sentences in summaries generated by the pointer-generator baseline and our 2-decoder model.

Abstractiveness: Abstractiveness is another major challenge for current abstractive summarization models other than saliency. Since our baseline is an abstractive model, we measure the percentage of novel n-grams (n=2, 3, 4) in our generated summaries, and find that our 2-decoder model generates 1.8%, 4.8%, 7.6% novel n-grams while our baseline summaries have 1.6%, 4.4%, 7.1% on the same test set. Even though generating more abstractive summaries is not our focus in this paper, we still show that our improvements in metric and saliency scores are not obtained at the cost of making the model more extractive.

6.4 Discussion: Connection to Multi-Task Learning

Our 2-decoder model somewhat resembles a Multi-Task Learning (MTL) model, in that both try to improve the model with extra knowledge that is not available to the original single-task baseline. While our model uses MTL-style parameter sharing to introduce extra knowledge from the same dataset, traditional Multi-Task Learning usually employs additional/out-of-domain auxiliary tasks/datasets as related knowledge (e.g., translation with 2 language-pairs). Our 2-decoder model is more about how to learn to do a single task from two different points of view, as the pointer decoder is a hybrid of extractive and abstractive summarization models (primary view), and the closed-book decoder is trained for abstractive summarization only (auxiliary view). The two decoders share their encoder and embeddings, which helps enrich the encoder's final memory state representation.

Moreover, as shown in Sec. 6.2, our 2-decoder model (pg + cbdec) significantly outperforms the 2-duplicate-decoder model (pg + ptrdec) as well as single-decoder models with more layers/parameters, hence proving that our design of the auxiliary view (closed-book decoder doing abstractive summarization) is the reason behind the improved performance, rather than some simplistic effects of having a 2-decoder ensemble or higher #parameters.

7 Conclusion

We presented a 2-decoder sequence-to-sequence architecture for summarization with a closed-book decoder that helps the encoder to better memorize salient information from the source text. On *CNN/Daily Mail* dataset, our proposed model significantly outperforms the pointer-generator baselines in terms of ROUGE and METEOR scores (in both a cross-entropy (XE) setup and a reinforcement learning (RL) setup). It also achieves improvements in a test-only transfer setup on the DUC-2002 dataset in both XE and RL cases. We further showed that our 2-decoder model indeed has a stronger encoder with better memory capabilities, and can generate summaries with more salient information from the source text. To the best of our knowledge, this is the first work that studies the "representation power" of the encoders final state in an encoder-decoder model. Furthermore, our simple, insightful 2-decoder architecture can also be useful for other tasks that require long-term memory from the encoder, e.g., long-context QA/dialogue and captioning for long videos.

8 Acknowledgement

We thank the reviewers for their helpful comments. This work was supported by DARPA (YFA17-D17AP00022), Google Faculty Research Award, Bloomberg Data Science Research Grant, Nvidia GPU awards, and Amazon AWS. The views contained in this article are those of the authors and not of the funding agency.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Third International Conference on Learning Representations*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *IJCAI*.

Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 775–786.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*, pages 340–348. ACL.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, volume 14, pages 1602–1613.

George Giannakopoulos. 2009. Automatic summarization from multiple documents. *Ph. D. dissertation*.

Jiatao Gu, Baotian Hu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016a. Guided sequence-to-sequence learning with external rule memory. In *International Conference on Learning Representations*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016b. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Stefan Henß, Margot Mieskes, and Iryna Gurevych. 2015. A reinforcement learning approach for adaptive single-and multi-document summarization. In *GSCL*, pages 3–12.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Hongyan Jing and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 178–185, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy. 2017. Improved Image Captioning via Policy Gradient optimization of SPIDEr. In *International Conference on Computer Vision*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Ramesh Nallapati, Bowen Zhou, Çağlar Gu̇lçehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Computational Natural Language Learning*.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representation*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. In *Computer Vision and Pattern Recognition (CVPR)*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Empirical Methods in Natural Language Processing*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *ACL*.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Hao Xiong, Zhongjun He, Xiaoguang Hu, and Hua Wu. 2018. Multi-channel encoder for neural machine translation. In *AAAI*.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston*, pages 112–119.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*.

Improving Neural Abstractive Document Summarization with Structural Regularization[*]

Wei Li[1,2,3] **Xinyan Xiao**[2] **Yajuan Lyu**[2] **Yuanzhuo Wang**[1]
[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]Baidu Inc., Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
`weili.ucas.ict@gmail.com`, `{xiaoxinyan,lvyajan}@baidu.com`,
`wangyuanzhuo@ict.ac.cn`

Abstract

Recent neural sequence-to-sequence models have shown significant progress on short text summarization. However, for document summarization, they fail to capture the long-term structure of both documents and multi-sentence summaries, resulting in information loss and repetitions. In this paper, we propose to leverage the structural information of both documents and multi-sentence summaries to improve the document summarization performance. Specifically, we import both *structural-compression* and *structural-coverage* regularization into the summarization process in order to capture the information compression and information coverage properties, which are the two most important structural properties of document summarization. Experimental results demonstrate that the structural regularization improves the document summarization performance significantly, which enables our model to generate more informative and concise summaries, and thus significantly outperforms state-of-the-art neural abstractive methods.

1 Introduction

Document summarization is the task of generating a fluent and condensed summary for a document while retaining the gist information. Recent success of neural sequence-to-sequence (seq2seq) architecture on text generation tasks like machine translation (Bahdanau et al., 2014) and image caption (Vinyals et al., 2015), has attracted growing attention to abstractive summarization research. Huge success has been witnessed in abstractive sentence summarization (Rush et al., 2015; Takase et al., 2016; Chopra et al., 2016; Cao et al., 2017; Zhou et al., 2017), which builds one-sentence summaries from one or two-sentence input. However, the extension of sentence abstractive methods to document summarization task is not straightforward.

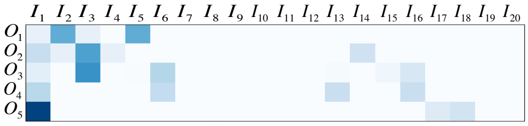

(a) Gold Reference

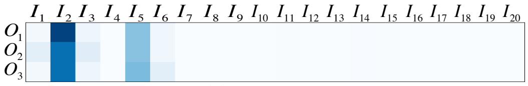

(b) Seq2Seq-baseline

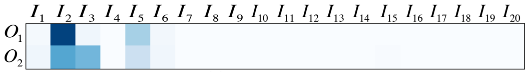

(c) Point-gen-cov

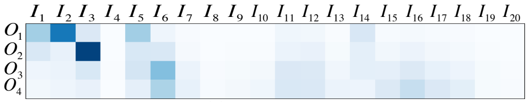

(d) Hierarchical-baseline

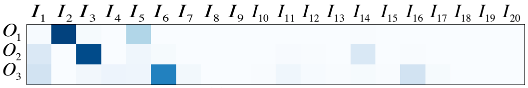

(e) Our Method

Figure 1: Comparison of sentence-level attention distributions for the summaries in Table 1 on a news article. (a) is the heatmap for the gold reference summary, (b) is for the *Seq2seq-baseline* system, (c) is for the *Point-gen-cov* (See et al., 2017) system, (d) is for the *Hierarchical-baseline* system and (e) is for our system. I_i and O_i indicate the i-th sentence of the input and output, respectively. Obviously, the seq2seq models, including the *Seq2seq-baseline* model and the *Point-gen-cov* model, lose much salient information of the input document and focus on the same set of sentences repeatedly. The *Hierarchical-baseline* model fails to detect several specific sentences that are salient and relevant for each summary sentence and focuses on the same set of sentences repeatedly. On the contrary, our method with structural regularizations focuses on different sets of source sentences when generating different summary sentences and discovers more salient information from the document.

As long-distance dependencies are difficult to be captured in the recurrent framework (Bengio et al., 1994), the seq2seq models are not yet able to achieve convincing performance in encoding and decoding for a long sequence of multiple sentences (Chen et al., 2017; Koehn and Knowles,

---

[*] This work was done while the first author was doing internship at Baidu Inc.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4078–4087
Brussels, Belgium, October 31 - November 4, 2018.

Original Text (truncated): the family of conjoined twin sisters who died 19 days after they were born have been left mortified$_{(2)}$ after they arrived at their gravesite to find cemetery staff had cleared the baby section of all mementos and tossed them in the rubbish$_{(3)}$. faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition ~~known as disrosopus~~ $_{(1)}$. they died in hospital less than a month after they were born and their parents ~~, simon howie and renee young ,~~ laid them to rest at pinegrove memorial park in sydney 's west$_{(2)}$. scroll down for video . faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition ~~known as disrosopus~~$_{(1)}$. family members have visited the grave every week to leave mementos and flowers for faith and hope , but when mr howie and ms young arrived on thursday they found the site completely bare$_{(3)}$. ' we took renee's aunts to see the girls for the first time and we found everything had been stripped away , ' mr howie told daily mail australia . ' we were devastated and mortified . we 've had a little shrine set up and we 've been adding to it since the funeral . ' it 's heartbreaking to know we 've set this up and it has been treated like rubbish . ' faith and hope were buried in a pink coffin and their family and friends released doves and pink and white balloons at their funeral . their family and friends had built up a small memorial with pink and white statues , flowers , pebbles and toys over the past 11 months . when they arrived on thursday , everything had been removed apart from a bunch of flowers . the twins were buried at pinegrove memorial park in western sydney after they died after just 19 days$_{(2)}$. their family and friends had built a small shrine at their gravesite , which they have added to since the funeral . family members have visited the grave every week to leave mementos and flowers for faith and hope , but when parents simon howie and renee young arrived on thursday they found the site completely bare$_{(3)}$.
Gold Reference: faith and hope howie were born with one body and two faces on may 8. they tragically died in hospital just 19 days after they were born . parents simon howie and renee young visit their grave at pinegrove in western sydney fortnightly . they arrived on thursday to find the grave bare of all the girls ' mementos . staff had cleared entire baby section and thrown belongings in rubbish .
Seq2seq-baseline: faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition known as disrosopus . faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition known as disrosopus . faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition known as disrosopus .
Point-cov (See et al., 2017): faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition known as disrosopus . they died in hospital less than a month after they were born and their parents , simon howie and renee young , laid them to rest at pinegrove memorial park in sydney 's west.
Hierarchical-baseline: faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition . they died in hospital less than a month after they died in hospital less than a month after they were born and laid them to rest at pinegrove memorial park in sydney 's west . family members have visited the grave every week to leave mementos and flowers for faith and hope , but when they were born on thursday they found the site completely bare . family members have visited the grave every week to leave mementos and flowers for faith and hope , but when they found the site completely bare .
Our Method: faith and hope howie were dubbed the miracle twins when they were born on may 8 last year with one body and two faces due to an extremely rare condition$_{(1)}$. they died in hospital less than a month after they were born and their parents laid them to rest at pinegrove memorial park in sydney 's west$_{(2)}$. family members have visited the grave every week to leave mementos and flowers for faith and hope , but when mr howie and ms young arrived on thursday they found the site completely bare$_{(3)}$.

Table 1: Comparison of the generated summaries of four abstractive summarization models and the gold reference summary on a news article. The summaries generated by the seq2seq models, both the Seq2seq-baseline model and the Point-cov model, lose some salient information. The Seq2seq-baseline model even contains serious information repetitions. The Hierarchical-baseline model not only contains serious repetitions, but also makes non-grammatical or non-coherent sentences. On the contrary, the summary generated by our model contains more salient information and is more concise. Our model also shows the ability to generate a summary sentence by compressing several source sentences, such as ~~shortening a long sentence~~.

2017). In document summarization, it is also difficult for the seq2seq models to discover important information from too much input content of a document (Tan et al., 2017a,b). The summary generated by the seq2seq models usually loses salient information of the original document or even contains repetitions (see Table 1).

In fact, both document and summary naturally have document-sentence hierarchical structure, instead of being a flat sequence of words. It is widely aware that the hierarchical structure is necessary and useful for neural document modeling. Hierarchical neural models have already been successfully used in document-level language modeling (Lin et al., 2015) and document classification (Yang et al., 2016). However, few work makes use of the hierarchical structure of document and multi-sentence summary in document summarization. The basic hierarchical encoder-decoder model (Li et al., 2015) is also not yet able to capture the structural properties of both document and summary (see Figure 1[1]), resulting in more serious repetitions and even nonsensical sentences (see Table 1).

In document summarization, information compression and information coverage are the two most important structural properties. Based on the hierarchical structure of document and summary, they can be realized at the sentence-level as: (1) *Structural-compression*: each summary sentence is generated by compressing several specific source sentences; (2) *Structural-coverage*: different summary sentences usually focus on different sets of source sentences to cover more salient information of the original document. Figure 1(a) intuitively shows the two properties in human-written gold reference summaries. We import both structural-compression and structural-coverage regularizations into the document summarization process based on a hierarchical encoder-decoder with hybrid sentence-word attention model. Typically, we design an effective learning and inference algorithm to explicitly model the structural-compression and structural-coverage properties of document summarization process, so as to generate more informative and concise summaries (see Table 1).

We conduct our experiments on benchmark datasets and the results demonstrate that properly modeling the *structural-compression* and *structural-coverage* properties based on the hier-

---

[1] To simulate the sentence-level attention mechanism on the gold reference summary, we compute the words-matching similarities (based on TF-IDF cosine similarity) between a reference-summary sentence and the corresponding source document sentences and normalize them into attention distributions. The sentence-level attention distributions of the Seq2seq-baseline model and the Point-gen-cov model are computed by summing the attention weights of all words in each sentence and then normalized across sentences.

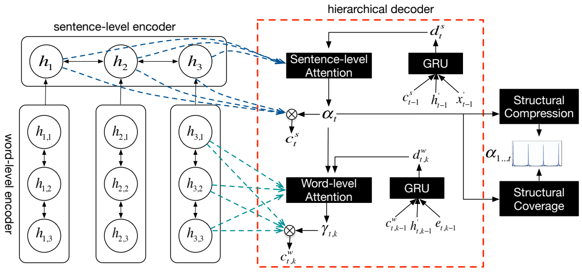

Figure 2: Our hierarchical encoder-decoder model with structural regularization for abstractive document summarization.

archical structure of document and summary, improves document summarization performance significantly. Our model is able to generate more informative and concise summaries by enhancing sentences compression and coverage, and significantly outperforms state-of-the-art seq2seq-based abstractive methods, especially on summarizing long documents with long summaries.

2 Hierarchical Encoder-Decoder Model

In this section, we introduce our baseline hierarchical encoder-decoder model which consists of two parts: a hierarchical encoder and a hierarchical decoder, as shown in Figure 2. Similar to (Li et al., 2015), both the encoder and decoder consists of two levels: a sentence level and a word level. The main distinction is that we design a hybrid sentence-word attention mechanism on the hierarchical decoder to help organize summary content and realize summary sentences.

2.1 Hierarchical Encoder

The goal of the encoder is to map the input document to a hidden vector representation. We consider a source document X as a sequence of sentences: $X = \{s_i\}$, and each sentence s_i as a sequence of words: $s_i = \{w_{ij}\}$. The word-level encoder encodes the words of a sentence into a sentence representation, and the sentence-level encoder encodes the sentences of a document into the document representation. In this work, both the word-level encoder and sentence-level encoder use the bidirectional Gated Recurrent Unit (BiGRU) (Chung et al., 2014). The word-level encoder sequentially updates its hidden state upon each received word, as $h_{i,j} = BiGRU(h_{i,j-1}, e_{i,j})$ where $h_{i,j}$ and $e_{i,j}$ denote the hidden state and the embedding of word $w_{i,j}$, respectively. The concatenation of the forward and backward final hidden states in the word-level encoder is indicated as the vector representation r_i of sentence s_i, which is used as input to the sentence-level encoder. The sentence-level encoder updates its hidden state after receiving each sentence representation, as $h_i = BiGRU(h_{i-1}, r_i)$ where h_i denotes the hidden state of sentence s_i. The concatenation of the forward and backward final states in the sentence-level encoder is used as the vector representation of document $\mathbf{d}$.

In the hierarchical encoder architecture, long dependency problem will be largely reduced at both the sentence level and the word level, so it can better capture the structural information of the input document.

2.2 Hierarchical Decoder with Hybrid Sentence-Word Attention

The goal of the decoder is to generate output summary according to the representation of the input document. Let $Y = \{s'_i\}$ indicates a candidate summary of document X, and each sentence s'_i consists of a sequence of words $s'_i = \{w'_{ij}\}$. The hierarchical decoder organizes summary Y sentence by sentence, and realizes each sentence word by word. In this work, both the sentence-level decoder and word-level decoder use a single layer of unidirectional GRU. The sentence-level decoder receives the document representation $\mathbf{d}$ as initial state h'_0 and predicts sentence representations sequentially by $h'_t = GRU(h'_{t-1}, r'_{t-1})$, where h'_t denotes the hidden state of the tth summary sentence s'_t and r'_{t-1} denotes the encoded representation of the previously generated sentence s'_{t-1}. The word-level decoder receives a sentence representation h'_t as initial state $h'_{t,0}$ and predicts word representations sequentially by $h'_{t,k} = GRU(h'_{t,k-1}, e_{t,k-1})$ where $h'_{t,k}$ denotes the hidden state of word $w'_{t,k}$ in sentence s'_t and $e_{t,k-1}$ denotes the embedding of previously generated word $w'_{t,k-1}$ in sentence s'_t.

In this work, we design a hybrid sentence-word attention mechanism based on the hierarchical encoder-decoder architecture, which contains both **sentence-level attention** and **word-level attention**, to better exploit both the sentence-level information and word-level information from the input document and the output summary.

2.2.1 Sentence-level Attention

The sentence-level attention mechanism is designed on the sentence-level encoder and decoder, which is used to help our model to detect important and relevant source sentences in each sentence generation step. α_t^i indicates how much the t-th summary sentence attends to the i-th source sentence, which is computed by $\alpha_t^i = e^{f(h_i,h'_t)}/\sum_l e^{f(h_l,h'_t)}$ where f is the function modeling the relation between h_i and h'_t. We use the function $f(\mathbf{a},\mathbf{b}) = v^T tanh(W_a\mathbf{a} + W_b\mathbf{b})$, where v, W_a, W_b are all learnable parameters. Then the sentence level context vector c_t^s when generating the tth sentence s'_t can be computed as: $c_t^s = \sum_i \alpha_t^i h_i$, which is incorporated into the sentence-level decoding process.

2.2.2 Word-level Attention with Sentence-level Normalization

The word-level attention is designed on the word-level encoder and decoder, which is used to help our model to realize the summary sentence by locating relevant words in the selected source sentences in each word generation step. Let $\beta_{t,k}^{i,j}$ denotes how much the j-th word in source sentence s_i contributes to generating the k-th word in summary sentence s'_t, which is computed by $\beta_{t,k}^{i,j} = e^{f(h_{i,j},h'_{t,k})}/\sum_l e^{f(h_{i,l},h'_{t,k})}$.

Since the word-level attention above is within each source sentence, we normalize it by sentence-level attentions to get word attention over all source words, as $\gamma_{t,k}^i = \beta_{t,k}^i \alpha_t^i$. Then the word-level context vector when generating word $w'_{t,k}$ can be computed as: $\mathbf{c}_{t,k}^w = \sum_i \sum_j \gamma_{t,k}^{i,j} h_{i,j}$, which is also incorporated into the word-level decoding process.

At each word generation step, the vocabulary distribution is calculated from the context vector $\mathbf{c}_{t,k}^w$ and the decoder state $h_{t,k}^{'}$ by:

$$P_{vocab}(w'_{t,k}) = softmax(W_v(W_c[h_{t,k}^{'}, c_{t,k}^w] + b_c) + b_v) \qquad (1)$$

where W_v, W_c, b_c and b_v are learned parameters. We also incorporate the copy mechanism (See et al., 2017) based on the normalized word-level attention to help generate out-of-vocabulary (OOV) words during the sentence realization process.

3 Structural Regularization

Although the above hierarchical encoder-decoder model is designed based on the document-sentence hierarchical structure, it can't capture the basic structural properties of document summarization (see Figure 1(d) and Table 1). However, the hierarchical architecture makes it possible for importing structural regularization to capture the sentence-level characteristics of document summarization process. In this work, we propose to model the *structural-compression* and *structural-coverage* properties based on the hierarchical encoder-decoder model by adding structural regularization during both the model learning phase and inference phase.

3.1 Structural Compression

Compression is a basic property of document summarization, which has been widely explored in traditional document summarization research, such as sentence compression-based methods which shorten sentences by removing non-salient parts (Li et al., 2013; Durrett et al., 2016) and sentence fusion-based methods which merge information from several different source sentences (Barzilay and McKeown, 2005; Cheung and Penn, 2014). As shown in Figure 1, each summary sentence in the human-written reference summary is also created by compressing several specific source sentences.

In this paper, we propose to model the *structural-compression* property of document summarization based on sentence-level attention distributions by:

$$strCom(\alpha_t) = 1 - \frac{1}{logN}\sum_{i=1}^{N} \alpha_t^i log\alpha_t^i \qquad (2)$$

where α_t denotes the sentence-level attention distribution when generating the tth summary sentence and N denotes the length of distribution α_t. The right part in the above formula is actually the entropy of the distribution α_t. As the attention distribution becomes sparser, the entropy of the distribution becomes lower, so the value of $strCom(\alpha_t)$ defined above will become larger. Sparse sentence-level attentions help the model compress and generalize several specific source sentences which are salient and relevant in the sentence generation process. Note that, $0 \leq strCom(\alpha_t) \leq 1$.

3.2 Structural Coverage

A good summary should have the ability to cover most of the important information of an input document. As shown in Figure 1, the human-written reference summary covers the information of many source sentences. Coverage has been used as a measure in many traditional document summarization research, such as the submodular-based methods which optimize the information coverage of the summary with similarity-based coverage metrics (Lin and Bilmes, 2011; Chali et al., 2017).

In this work, we simply model the *structural-coverage* property of summary based on the hierarchical architecture by encouraging different summary sentences to focus on different sets of source sentences so that the summary can cover more salient sentences of the input document. We measure the *structural-coverage* of summary based on the sentence-level attention distributions:

$$strCov(\alpha_t) = 1 - \sum_i \min(\alpha_t^i, \sum_{t'=0}^{t-1} \alpha_{t'}^i) \quad (3)$$

which is used to encourage different summary sentences to focus on different sets of source sentences during the summary generation process. As the sentence-level attention distributions of different summary sentences become more diversified, the summary will cover more source sentences, which is effective to improve the informativeness and conciseness of summaries. Note that, $0 \leq strCov(\alpha_t) \leq 1$.

3.3 Model Learning

Experimental results reveal that the properties of *structural-compression* and *structural-coverage* are hard to be captured by both the seq2seq models and the hierarchical encoder-decoder baseline model, which largely restricts their performance (Section 4). In this work, we model them explicitly by regulating the sentence-level attention distributions based on the hierarchical encoder-decoder framework. The loss function $\mathcal{L}$ of the model is the mix of negative log-likelihood of generating summaries over training set $\mathcal{T}$, the structural-compression loss and the structural-coverage loss:

$$\mathcal{L} = \sum_{(X,Y)\in\mathcal{T}} \{-logP(Y|X;\theta) + \underbrace{\lambda_1 \sum_t strCom(\alpha_t)}_{structural-compression\ loss} + \underbrace{\lambda_2 \sum_t strCov(\alpha_t)}_{structural-coverage\ loss} \} \quad (4)$$

where λ_1 and λ_2 are hyper-parameters tuned on the validation set. We use Adagrad (Duchi et al., 2011) with learning rate 0.1 and an initial accumulator value 0.1 to optimize the model parameters θ.

3.4 Hierarchical Decoding Algorithm

The traditional beam search algorithm that widely used for text generation can only help generate fluent sentence, and is not easy to extend to the sentence level. The reason is that the K-best sentences generated by a word decoder will mostly be similar to each other (Li et al., 2016; Tan et al., 2017a). We propose a hierarchical beam search algorithm with *structural-compression* and *structural-coverage* regularization.

The hierarchical decoding algorithm has two levels: K-best word-level beam search and N-best sentence-level beam search. At the word-level, the vanilla beam search algorithm is used to maximize the accumulated score $\hat{P}(s'_t)$ of generating current summary sentence s'_t. At the sentence-level, N-best beam search is realized by maximizing the accumulated score $score_t$ of all the sentences generated, including the sentences generation score, structural-compression score and structural-coverage score, which are defined as:

$$score_t = \sum_{t'=0}^{t} \{\hat{P}(s'_{t'}) + \zeta_1 strCom(\alpha_{t'}) + \zeta_2 strCov(\alpha_{t'})\} \quad (5)$$

where ζ_1 and ζ_2 are factors introduced to control the influence of structural regularization during the decoding process.

4 Experiments

4.1 Experimental Settings

We conduct our experiments on the *CNN/Daily Mail* dataset (Hermann et al., 2015), which has been widely used for exploration on summarizing documents with multi-sentence summaries (Nallapati et al., 2016; See et al., 2017; Tan et al., 2017a; Paulus et al., 2017). The CNN/DailyMail dataset contains input sequences of about 800 tokens in average and multi-sentence summaries of up to 200 tokens. The average number of sentences in documents and summaries are 42.1 and 3.8, respectively. We use the same version of non-anonymized data (the original text without pre-processing) as See et al. (2017), which has 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs.

For all experiments, the word-level encoder and decoder both use 256-dimensional hidden states, and the sentence-level encoder and decoder both

Method	Rouge-1	Rouge-2	Rouge-L
SummaRuNNer-abs	37.5	14.5	33.4
SummaRuNNer	39.6	16.2	35.3
Seq2seq-baseline	36.64	15.66	33.42
ABS-temp-attn	35.46	13.30	32.65
Graph-attention	38.1	13.9	34.0
Point-cov	39.53	17.28	36.38
Hierachical-baseline	34.95	14.79	32.68
Our Model	**40.30**	**18.02**	**37.36**

Table 2: Rouge F_1 scores on the test set. All our ROUGE scores have a **95% confidence interval of at most ± 0.25** as reported by the official ROUGE script.

length	Method	Rouge-1	Rouge-2	R.-L
< 100	Our M.	**39.66**	**17.28**	**36.69**
(94.47%)	Point-cov	39.44	17.20	36.30
$[100, 125)$	Our M.	**43.07**	**19.96**	**39.47**
(4.00%)	Point-cov	41.78	19.00	38.41
$[125, 150)$	Our M.	**43.25**	**19.21**	**40.08**
(1.07%)	Point-cov	41.31	18.02	37.75
> 150	Our M.	**40.64**	**18.30**	**38.00**
(0.46%)	Point-cov	35.64	17.76	33.12

Table 3: Comparison results w.r.t different length of reference summary. < 100 indicates the reference summary has less than 100 words (occupy 94.47% of test set).

use 512-dimensional hidden states. The dimension of word embeddings is 128, which is learned from scratch during training. We use a vocabulary of 50k words for both the encoder and decoder.

We trained our model on a single Tesla K40m GPU with a batch size of 16 and an epoch is set containing 10,000 randomly sampled documents. Convergence is reached within 300 epochs. After tuning on the validation set, parameters λ_1, λ_2, ζ_1 and ζ_2, are set as -0.5, -1.0, 1.2 and 1.4, respectively. At the test time, we use the hierarchical decoding algorithm with sentence-level beam size 4 and word-level beam size 8.

4.2 Evaluation

ROUGE Evaluation. We evaluate our models with the widely used ROUGE (Lin, 2004) toolkit. We compare our system's results with the results of state-of-the-art neural summarization approaches reported in recent papers, including both abstractive models and extractive models. The extractive models include **SummaRuNNer** (Nallapati et al., 2017) and **SummaRuNNer-abs** which is similar to SummaRuNNer but is trained directly on the abstractive summaries. The abstractive models include:

1) **Seq2seq-baseline**, which uses the basic seq2seq encoder-decoder architecture with attention mechanism, and incorporates with copy mechanism (See et al., 2017) to alleviate the OOV problem.
2) **ABS-temp-attn** (Nallapati et al., 2016), which uses *Temporal Attention* on the seq2seq architecture to overcome the repetition problem.
3) **Point-cov** (See et al., 2017), which is an extension of the Seq2seq-baseline model by importing word-coverage mechanism to reduce repetitions in summary.
4) **Graph-attention** (Tan et al., 2017a), which uses a graph-ranking based attention mechanism based on a hierarchical architecture to identify important sentences.
5) **Hierachical-baseline**, which just uses the basic hierarchical encoder-decoder with hybrid attention model proposed in this paper.

Results in Table 2 show that our model significantly outperforms all the neural abstractive baselines and extractive baselines. An interesting observation is that the performance of the **Hierarchical-baseline** model are lower than the **Seq2seq-baseline** model, which demonstrates the difficulty for a traditional model to identify the structural properties of document summarization process. Our model outperforms the **Hierarchical-baseline** model by more than 4 ROUGE points, which demonstrates that the structural regularization improves the document summarization performance significantly.

To verify the superiority of our model on generating long summaries, we also compare our method with the best seq2seq model **Point-cov** (See et al., 2017) by evaluating them on a test set w.r.t. different length of reference summaries. The results are shown in Table 3, which demonstrate that our model is better at generating long summary than the seq2seq model. As the summary becomes longer, our system will obtain larger advantages over the baseline (from +0.22 Rouge-1, +0.08 Rouge-2 and +0.39 Rouge-L for summary less than 100 words, rising to +5.00 Rouge-1, +0.54 Rouge-2 and +4.88 Rouge-L for summaries more than 150 words).

Human Evaluation. In addition to the ROUGE evaluation, we also conducted human evaluation on 50 random samples from CNN/DailyMail test set and compared the summaries generated by our method with the outputs of **Seq2seq-baseline** and **Point-cov** (See et al., 2017). Three data annotators were asked to compare the generated summaries

Method	Informat.	Concise	Coherent	Fluent
Seq2seq-b.	2.79*	2.52*	2.68*	3.57
Point-cov	3.17*	2.92*	3.00*	3.54
Our Model	3.67	3.39	3.51	3.70

Table 4: Human evaluation results. * indicates the difference between **Our Model** and other models are statistic significant ($p < 0.05$) by two-tailed t-test.

Method	R-1	R-2	R-L	strCom	strCov
Hierarchical-b.	34.95	14.79	32.68	0.22	0.31
+*strCom*	37.03	16.21	34.44	0.64	0.71
+*strCov*	39.52	17.12	36.44	0.65	0.87
+hierD	**40.30**	**18.02**	**37.36**	**0.68**	**0.93**

Table 5: Results of adding different components of our method in terms of ROUGE-1, ROUGE-2, ROUGE-L, strCom (Equation 1) and strCov (Equation 2) scores.

with the human summaries, and assess each summary from four independent perspectives: (1) **Informative**: How informative the summary is? (2) **Concise**: How concise the summary is? (3) **Coherent**: How coherent (between sentences) the summary is? (4) **Fluent**: How fluent, grammatical the sentences of a summary are? Each property is assessed with a score from 1(worst) to 5(best). The average results are presented in Table 4.

The results show that our model consistently outperforms the **Seq2seq-baseline** model and the previous state-of-the-art method **Point-cov**. As shown in Table 1, the summary generated by **Seq2Seq-Baseline** usually contains repetition of sentences or phrases, which seriously affects its informativeness, conciseness as well as coherence. The **Point-cov** model effectively alleviates the information repetition problem, however, it usually loses some salient information and mainly copies original sentences directly from the input document. The summaries generated by our method obviously contains more salient information and are more concise through sentences compression, which shows the effectiveness of the structural regularization in our model. The results also show that the sentence-level modeling of document and summary in our model makes the generated summaries achieve better inter-sentence coherence.

5 Discussion

5.1 Model Validation

To verify the effectiveness of each component in our model, we conduct several ablation experiments. Based on the *Hierarchical-baseline* model, several different structural regularizations are added one by one: +*strCom* indicates adding structural-compression regularization during model learning, +*strCov* indicates adding structural-coverage regularization during model learning, +*hierD* indicates using the hierarchical decoding algorithm with both structural-compression and structural-coverage regularizations during inference.

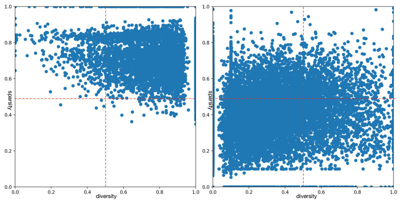

(a) gold reference summary

(b) Seq2seq-baseline

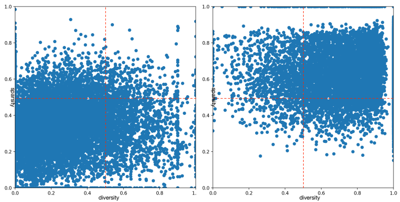

(c) Hierarchical-baseline

(d) Our model with structural regularizations

Figure 3: Comparisons of structural-compression and structural-coverage analysis results on random samples from CNN/Daily Mail datasets, which demonstrate that both the Seq2seq-baseline model and the Hierarchical-baseline model are not yet able to capture them properly, but our model with structural regularizations achieves similar behavior with the gold reference summary.

Results on the test set are shown in Table 5. Our method much outperforms all the compared systems, which verifies the effectiveness of each component of our model. Note that, both the structural-compression and structural-coverage regularization significantly affect the summarization performance. The higher structural-compression and structural-coverage scores will lead to higher ROUGE scores. Therefore, we can conclude that the structural-compression and structural-coverage regularization based on our hierarchical model have significant contributions to the increase of ROUGE scores.

5.2 Structural Properties Analysis

We further compare the ability of different models in capturing the structural-compression and structural-coverage properties of document summarization. Figure 3 shows the comparison results of 4000 document-summary pairs with 14771 reference-summary sentences sampled from CNN/Daily Mail dataset. Figure 3(a) shows that most samples (over 95%) fall into the right-top area in human-made summaries, which indicates high structural-compression and structural-

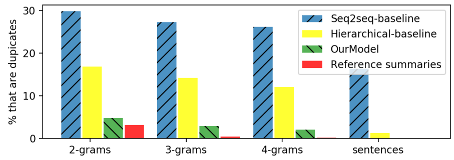

Figure 4: The structural regularization reduces undesirable repetitions while summaries from the Seq2seq-baseline and the Hierarchical-baseline contains many n-gram repetitions.

coverage scores. However, Figure 3 (b) and (c) show that in both the Seq2seq-baseline model and the Hierarchical-baseline model, most samples fall into the left-bottom area (low structural-compression and structural-coverage), and only about 13% and 7% samples fall into the right-top area, respectively. Figure 3 (d) shows that our system with structural regularization achieves similar behaviors to human-made summaries (over 80% samples fall into the right-top area). The results demonstrate that the structural-compression and structural-coverage properties are common in document summarization, but both the seq2seq models and the basic hierarchical encoder-decoder models are not yet able to capture them properly.

5.3 Effects of Structural Regularization

The structural regularization based on our hierarchical encoder-decoder with hybrid attention model improves the quality of summaries from two aspects: (1) The summary covers more salient information and contains very few repetitions, which can be seen both qualitatively (Table 1 and Figure 1) and quantitatively (Table 5 and Figure 4). (2) The model has the ability to shorten a long sentence to generate a more concise one or compress several different sentences to generate a more informative one by merging the information from them. Table 6 shows several examples of abstractive summaries produced by sentence compression in our model.

6 Related Work

Recently some work explored the seq2seq models on document summarization, which exhibit some undesirable behaviors, such as inaccurately reproducing factual details, OOVs and repetitions. To alleviate these issues, copying mechanism (Gu et al., 2016; Gulcehre et al., 2016; Nallapati et al., 2016) has been incorporated into the encoder-decoder architecture to help generate information correctly. Distraction-based attention model (Chen et al., 2016) and word-level coverage mechanism (See et al., 2017) have also been investigated to alleviate the repetition problem. Reinforcement learning has also been studied to improve the document summarization performance from global sequence level (Paulus et al., 2017).

Original Text: luke lazarus ~~, a 23-year-old former private school boy ,~~ was jailed for at least three years on march 27 for raping an 18-year-old virgin in an alleyway outside his father 's soho nightclub ~~in kings cross , inner sydney~~ in may 2013 .(...) **Summary:** luke lazarus was jailed for at least three years on march 27 for raping an 18-year-old virgin in an alleyway outside his father 's soho nightclub in may 2013 .
Original Text: (...) amy wilkinson , 28 , claimed housing benefit and council tax benefit even though she was living in a home owned by her mother and her partner ~~, who was also working~~ .wilkinson ~~, who was a british airways cabin crew attendant ,~~ was ordered to pay back a total of 17,604 that she claimed over two years when she appeared at south and east cheshire magistrates court last week . (...) **Summary:** amy wilkinson , 28 , claimed housing benefit and council tax benefit even though she was living in a home owned by her mother and her partner . *she* was ordered to pay back a total of 17,604 that she claimed over two years when she appeared at south and east cheshire magistrates court last week .
Original Text: (...) a grand jury charged durst with possession of a firearm by a felon , and possession of both a firearm and an illegal drug : 5 ounces of marijuana , said assistant district attorney chris bowman , spokesman for the district attorney . millionaire real estate heir robert durst was indicted wednesday on the two weapons charges that have kept him in new orleans even though his lawyers say he wants to go to los angeles as soon as possible to face a murder charge there . his arrest related to those charges has kept durst from being extradited to los angeles , where he 's charged in the december 2000 death of longtime friend susan berman .(...) **Summary:** *durst entered his plea during an arraignment in a new orleans court on weapons charges that accused him of possessing both a firearm and an illegal drug , marijuana . the weapons arrest has kept durst in new orleans even though he is charged in the december 2000 death of a longtime friend* .

Table 6: Examples of sentences compression or fusion by our model. The ~~link-through~~ denotes deleting the non-salient part of the original text. The *italic* denotes novel words or sentences generated by sentences fusion or compression.

Hierarchical Encoder-Decoder architecture is first proposed by Li et al. (2015) to train an auto-encoder to reconstruct multi-sentence paragraphs. In summarization field, hierarchical encoder has first been used to alleviate the long dependency problem for long inputs (Cheng and Lapata, 2016; Nallapati et al., 2016). Tan et al. (2017b) also propose to use a hierarchical encoder to encode multiple summaries produced by several extractive summarization methods, and then decode them into a headline. However, these models don't model the decoding process hierarchically.

Tan et al. (2017a) first use the hierarchical encoder-decoder architecture on generating multi-sentences summaries. They mainly focus on incorporating sentence ranking into abstractive document summarization to help detect important sentences. Different from that, our work mainly tends to verify the necessity of leveraging document structure in document summarization and studies how to properly capture the structural properties of document summarization based on the hierarchical architecture to improve the performance of document summarization.

7 Conclusions

In this paper we analyze and verify the necessity of leveraging document structure in document summarization, and explore the effectiveness of capturing structural properties of document summarization by importing both *structural-compression* and *structural-coverage* regularization based on the proposed hierarchical encoder-decoder with hybrid attention model. Experimental results demonstrate that the structural regularization enables our model to generate more informative and concise summaries by enhancing sentences compression and coverage. Our model achieves considerable improvement over state-of-the-art seq2seq-based abstractive methods, especially on long document with long summary.

Acknowledgments

This work was supported by National Key Research and Development Program of China under grants 2016YFB1000902 and 2017YFC0820404, and National Natural Science Foundation of China under grants 61572469, 91646120, 61772501 and 61572473. We thank the anonymous reviewers for their helpful comments about this work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2017. Faithful to the original: Fact aware neural abstractive summarization. *arXiv preprint arXiv:1711.04434*.

Yllias Chali, Moin Tanvee, and Mir Tafseer Nayeem. 2017. Towards abstractive multi-document summarization using submodular function-based framework, sentence compression and merging. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 418–424.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. *arXiv preprint arXiv:1707.05436*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. *arXiv preprint arXiv:1610.08462*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 775–786.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. *arXiv preprint arXiv:1603.08887*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proceedings of the 2013 Conference on*

Empirical Methods in Natural Language Processing, pages 490–500.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *AAAI*, 1:1.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *EMNLP*, pages 1054–1059.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017a. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1171–1181.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017b. From neural sentence summarization to headline generation: A coarse-to-fine approach. *IJCAI*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. *arXiv preprint arXiv:1704.07073*.

Iterative Document Representation Learning Towards Summarization with Polishing

Xiuying Chen[1], **Shen Gao**[1], **Chongyang Tao**[1], **Yan Song**[2], **Dongyan Zhao**[1] **and Rui Yan**[1*]
[1]Institute of Computer Science and Technology, Peking University, Beijing, China
[2]Tencent AI Lab
{xy-chen,shengao,chongyangtao,zhaody,ruiyan}@pku.edu.cn
clksong@tencent.com

Abstract

In this paper, we introduce Iterative Text Summarization (ITS), an iteration-based model for supervised extractive text summarization, inspired by the observation that it is often necessary for a human to read an article multiple times in order to fully understand and summarize its contents. Current summarization approaches read through a document only once to generate a document representation, resulting in a sub-optimal representation. To address this issue we introduce a model which iteratively polishes the document representation on many passes through the document. As part of our model, we also introduce a selective reading mechanism that decides more accurately the extent to which each sentence in the model should be updated. Experimental results on the CNN/DailyMail and DUC2002 datasets demonstrate that our model significantly outperforms state-of-the-art extractive systems when evaluated by machines and by humans.

1 Introduction

A summary is a shortened version of a text document which maintains the most important ideas from the original article. Automatic text summarization is a process by which a machine gleans the most important concepts from an article, removing secondary or redundant concepts. Nowadays as there is a growing need for storing and digesting large amounts of textual data, automatic summarization systems have significant usage potential in society.

Extractive summarization is a technique for generating summaries by directly choosing a subset of salient sentences from the original document to constitute the summary. Most efforts made towards extractive summarization either rely on human-engineered features such as sentence length, word position, and frequency (Cohen, 2002; Radev et al., 2004; Woodsend and Lapata, 2010; Yan et al., 2011a,b, 2012) or use neural networks to automatically learn features for sentence selection (Cheng and Lapata, 2016; Nallapati et al., 2016a).

Although existing extractive summarization methods have achieved great success, one limitation they share is that they generate the summary after only one pass through the document. However, in real-world human cognitive processes, people read a document multiple times in order to capture the main ideas. Browsing through the document only once often means the model cannot fully get at the document's main ideas, leading to a subpar summarization. We share two examples of this. (1) Consider the situation where we almost finish reading a long article and forget some main points in the beginning. We are likely to go back and review the part that we forget. (2) To write a good summary, we usually first browse through the document to obtain a general understanding of the article, then perform a more intensive reading to select salient points to include in the summary. In terms of model design, we believe that letting a model read through a document multiple times, polishing and updating its internal representation of the document can lead to better understanding and better summarization.

To achieve this, we design a model that we call Iterative Text Summarization (ITS) consisting of a novel "iteration mechanism" and "selective reading module". ITS is an iterative process, reading through the document many times. There is one encoder, one decoder, and one iterative unit in each iteration. They work together to polish document representation. The final labeling part uses outputs from all iterations to generate summaries. The selective reading module we design is a modi-

---

[*] Corresponding author: Rui Yan (ruiyan@pku.edu.cn)

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4088–4097
Brussels, Belgium, October 31 - November 4, 2018.

fied version of a Gated Recurrent Unit (GRU) network, which can decide how much of the hidden state of each sentence should be retained or updated based on its relationship with the document.

Overall, our contribution includes:

1. We propose Iterative Text Summarization (ITS), an iteration based summary generator which uses a sequence classifier to extract salient sentences from documents.
2. We introduce a novel iterative neural network model which repeatedly polishes the distributed representation of document instead of generating that once for all. Besides, we propose a selective reading mechanism, which decides how much information should be updated of each sentence based on its relationship with the polished document representation. Our entire architecture can be trained in an end-to-end fashion.
3. We evaluate our summarization model on representative CNN/DailyMail corpora and benchmark DUC2002 dataset. Experimental results demonstrate that our model outperforms state-of-the-art extractive systems when evaluated automatically and by human.

2 Related Work

Our research builds on previous works in two fields: summarization and iterative modeling.

Text summarization can be classified into extractive summarization and abstractive summarization. Extractive summarization aims to generate a summary by integrating the most salient sentences in the document. Abstractive summarization aims to generate new content that concisely paraphrases the document from scratch.

With the emergence of powerful neural network models for text processing, a vast majority of the literature on document summarization is dedicated to abstractive summarization. These models typically take the form of convolutional neural networks (CNN) or recurrent neural networks (RNN). For example, Rush et al. (2015) propose an encoder-decoder model which uses a local attention mechanism to generate summaries. Nallapati et al. (2016b) further develop this work by addressing problems that had not been adequately solved by the basic architecture, such as keyword modeling and capturing the hierarchy of sentence-to-word structures. In a follow-up work, Nallapati et al. (2017) propose a new summarization model which generates summaries by sampling a topic one sentence at a time, then producing words using an RNN decoder conditioned on the sentence topic. Another related work is by See et al. (2017), where the authors use “pointing” and “coverage” techniques to generate more accurate summaries.

Despite the focus on abstractive summarization, extractive summarization remains an attractive method as it is capable of generating more grammatically and semantically correct summaries. This is the method we follow in this work. In extractive summarization, Cheng and Lapata (2016) propose a general framework for single-document text summarization using a hierarchical article encoder composed with an attention-based extractor. Following this, Nallapati et al. (2016a) propose a simple RNN-based sequence classifier which outperforms or matches the state-of-art models at the time. In another approach, Narayan et al. (2018) use a reinforcement learning method to optimize the Rouge evaluation metric for text summarization. The most recent work on this topic is (Wu and Hu, 2018), where the authors train a reinforced neural extractive summarization model called RNES that captures cross-sentence coherence patterns. Due to the fact that they use a different dataset and have not released their code, we are unable to compare our models with theirs.

The idea of iteration has not been well explored for summarization. One related study is Xiong et al. (2016)’s work on dynamic memory networks, which designs neural networks with memory and attention mechanisms that exhibit certain reasoning capabilities required for question answering. Another related work is (Yan, 2016), where they generate poetry with iterative polishing sn chema. Similiar method can also be applied on couplet generation as in (Yan et al., 2016). We take some inspiration from their work but focus on document summarization. Another related work is (Singh et al., 2017), where the authors present a deep network called Hybrid MemNet for the single document summarization task, using a memory network as the document encoder. Compared to them, we do not borrow the memory network structure but propose a new iterative architecture.

3 Methodology

3.1 Problem Formulation

In this work, we propose Iterative Text Summarization (ITS), an iteration-based supervised model for extractive text summarization. We treat the extractive summarization task as a sequence labeling problem, in which each sentence is visited sequentially and a binary label that determines whether or not it will be included in the final summary is generated.

ITS takes as input a list of sentences $s = \{s_1, \ldots, s_{n_s}\}$, where n_s is the number of sentences in the document. Each sentence s_i is a list of words: $s_i = \{w_1^i, \ldots, w_{n_w}^i\}$, where n_w is the word length of the sentence. The goal of ITS is to generate a score vector $\boldsymbol{y} = \{y_1, \ldots, y_{n_s}\}$ for each sentence, where each score $y_i \in [0, 1]$ denotes the sentence's *extracting probability*, that is, the probability that the corresponding sentence s_i will be extracted to be included in the summary. We train our model in a supervised manner, using a corresponding gold summary written by human experts for each document in training set. We use an unsupervised method to convert the human-written summaries to gold label vector $\boldsymbol{y'} = \{y'_1, ..., y'_{n_s}\}$, where $y'_i \in \{0, 1\}$ denotes whether the i-th sentence is selected (1) or not (0). Next, during training process, the cross entropy loss is calculated between $\boldsymbol{y}$ and $\boldsymbol{y'}$, which is minimized to optimize $\boldsymbol{y}$. Finally, we select three sentences with the highest score according to $\boldsymbol{y}$ to be the extracted summary. We detail our model below.

3.2 Model Architecture

ITS is depicted in Fig.1. It consists of multiple iterations with one encoder, one decoder, and one iteration unit in each iteration. We combine the outputs of decoders in all iterations to generate the extracting probabilities in the final labeling module.

Our encoder is illustrated in the shaded region in the left half of Fig.1. It takes as input all sentences as well as the document representation from the previous unit $\boldsymbol{D_{k-1}}$, processes them through several neural networks, and outputs the final state to the iterative unit module which updates the document representation.

Our decoder takes the form of a bidirectional RNN. It takes the representation of sentence generated by the encoder as input, and its initial state is the polished document representation $\boldsymbol{D_k}$. Our last module, the sentence labeling module, concatenates the hidden states of all decoders together to generate an integrated score for each sentence.

As we apply supervised training, the objective is to maximize the likelihood of all sentence labels $y' = \{y'_1, ..., y'_{n_s}\}$ given the input document s and model parameters θ:

$$\log p(y'|s;\theta) = \sum_{i=1}^{n_s} \log p(y'_i|s;\theta) \qquad (1)$$

4 Our Model

4.1 Encoder

In this subsection, we describe the encoding process of our model. For brevity, we drop the superscript k when focusing on a particular layer. All the $\boldsymbol{W}$'s and $\boldsymbol{b}$'s in this section with different superscripts or subscripts are the parameters to be learned.

Sentence Encoder: Given a discrete set of sentences $s = \{s_1, \ldots, s_{n_s}\}$, we use a word embedding matrix $\boldsymbol{M} \in \mathbb{R}^{V \times D}$ to embed each word w^i in sentence s_i into continuous space $\boldsymbol{\hat{w}^i}$, where V is the vocabulary size, D is the dimension of word embedding.

The sentence encoder can be based on a variety of encoding schemes. Simply taking the average of embeddings of words in a sentence will cause too much information loss, while using GRUs or Long Short-Term Memory (LSTM) requires more computational resources and is prone to overfitting. Considering above, we select *positional encoding* described in (Sukhbaatar et al., 2015) as our sentence encoding method. Each sentence representation $\boldsymbol{\hat{s}_i}$ is calculated by $\boldsymbol{\hat{s}_i} = \sum_{j=1}^{n_w} \boldsymbol{l_j} \circ \boldsymbol{\hat{w}_j^i}$, where $\circ$ is element-wise multiplication, $\boldsymbol{l_j}$ is a column vector computed as $l_{j,d} = (1 - \frac{j}{n_w}) - (\frac{d}{D})(1 - \frac{2j}{n_w})$, $l_{j,d}$ denotes the d-th dimension of $\boldsymbol{l_j}$.

Note that throughout this study, we use GRUs as our RNN cells since they can alleviate the overfitting problem as confirmed by our experiments. As our selective reading mechanism (which will be explained later) is a modified version of original GRU cell, we give the details of the GRU here. GRU is a gating mechanism in recurrent neural networks, introduced in (Cho et al., 2014). Their performance was found to be similar to that of LSTM cell but using fewer parameters as described in (Hochreiter and Schmidhuber, 1997). The GRU cell consists of an update gate vector

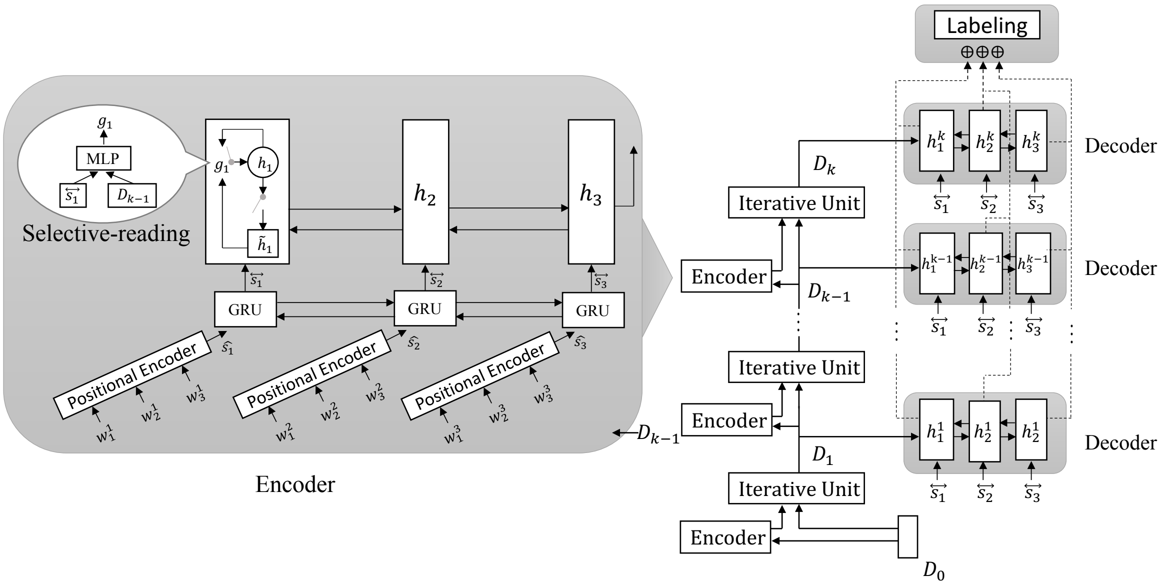

Figure 1: Model Structure: There is one encoder, one decoder and one iterative unit (which is used to polish document representation) in each iteration. The final labeling part is used to generating the extracting probabilities for all sentences combining hidden states of decoders in all iterations. We take a document consists of three sentences for example here.

$\boldsymbol{u_i}$, a reset gate vector $\boldsymbol{r_i}$, and an output vector $\boldsymbol{h_i}$. For each time step i with input $\boldsymbol{x_i}$ and previous hidden state $\boldsymbol{h_{i-1}}$, the updated hidden state $\boldsymbol{h_i} = \mathrm{GRU}(\boldsymbol{x_i}, \boldsymbol{h_{i-1}})$ is computed by:

$$\boldsymbol{u_i} = \sigma(\boldsymbol{W^{(u)}}\boldsymbol{x_i} + \boldsymbol{U^{(u)}}\boldsymbol{h_{i-1}} + \boldsymbol{b^{(u)}}) \quad (2)$$

$$\boldsymbol{r_i} = \sigma(\boldsymbol{W^{(r)}}\boldsymbol{x_i} + \boldsymbol{U^{(r)}}\boldsymbol{h_{i-1}} + \boldsymbol{b^{(r)}}) \quad (3)$$

$$\tilde{\boldsymbol{h_i}} = \tanh(\boldsymbol{W^{(h)}}\boldsymbol{x_i} + \boldsymbol{r_i} \circ \boldsymbol{U}\boldsymbol{h_{i-1}} + \boldsymbol{b^{(h)}}) \quad (4)$$

$$\boldsymbol{h_i} = \boldsymbol{u_i} \circ \tilde{\boldsymbol{h_i}} + (1 - \boldsymbol{u_i}) \circ \boldsymbol{h_{i-1}} \quad (5)$$

where σ is the sigmoid activation function, $\boldsymbol{W^{(u)}}, \boldsymbol{W^{(r)}}, \boldsymbol{W^{(h)}} \in \mathbb{R}^{n_H \times n_I}, \boldsymbol{U^{(u)}}, \boldsymbol{U^{(r)}}, \boldsymbol{U} \in \mathbb{R}^{n_H \times n_H}, n_H$ is the hidden size, n_I is the size of input $\boldsymbol{x_i}$.

To further study the interactions and information exchanges between sentences, we establish a Bi-directional GRU (Bi-GRU) network taking the sentence representation as input:

$$\overrightarrow{\boldsymbol{s_i}} = \mathrm{GRU}_{\mathrm{fwd}}(\hat{\boldsymbol{s}}_{\boldsymbol{i}}, \overrightarrow{\boldsymbol{s_{i-1}}}) \quad (6)$$

$$\overleftarrow{\boldsymbol{s_i}} = \mathrm{GRU}_{\mathrm{bwd}}(\hat{\boldsymbol{s}}_{\boldsymbol{i}}, \overleftarrow{\boldsymbol{s_{i-1}}}) \quad (7)$$

$$\overleftrightarrow{\boldsymbol{s_i}} = \overrightarrow{\boldsymbol{s_i}} + \overleftarrow{\boldsymbol{s_i}} \quad (8)$$

where $\hat{\boldsymbol{s}}_{\boldsymbol{i}}$ is the sentence representation input at time step i, $\overrightarrow{\boldsymbol{s_i}}$ is the hidden state of the forward GRU at time step i, and $\overleftarrow{\boldsymbol{s_i}}$ is the hidden state of the backward GRU. This architecture allows information to flow back and forth to generate new sentence representation $\overleftrightarrow{\boldsymbol{s_i}}$.

Document Encoder: We must initialize a document representation before polishing it. Generating the document representation from sentence representations is a process similar to generating the sentence representation from word embeddings. This time we need to compress the whole document, not just a sentence, into a vector. Because the information a vector can contain is limited, rather than to use another neural network, we simply use a non-linear transformation of the average pooling of the concatenated hidden states of the above Bi-GRU to generate the document representation, as written below:

$$\boldsymbol{D_0} = \tanh(\boldsymbol{W}\frac{1}{n_s}\sum_{i=1}^{n_s}[\overrightarrow{\boldsymbol{s_i}}; \overleftarrow{\boldsymbol{s_i}}] + \boldsymbol{b}) \quad (9)$$

where '[·;·]' is the concatenation operation.

Selective Reading module: Now we can formally introduce the selective reading module in Fig.1. This module is a bidirectional RNN consisting of modified GRU cells whose input is the sentence representation $\overleftrightarrow{\boldsymbol{s}} = \{\overleftrightarrow{\boldsymbol{s_1}}, ..., \overleftrightarrow{\boldsymbol{s_{n_s}}}\}$. In the original version of GRU, the update gate $\boldsymbol{u_i}$ in Equation 2 is used to decide how much of hidden state should be retained and how much should be updated. However, due to the way $\boldsymbol{u_i}$ is calculated, it is sensitive to the position and ordering of sentences, but loses information captured by the polished document representation.

Herein, we propose a modified GRU cell that replace the $\boldsymbol{u_i}$ with the newly computed update gate $\boldsymbol{g_i}$. The new cell takes in two inputs, the sentence representation and the document representation from the last iteration, rather than merely the sentence representation. For each sentence, the selective network generates an update gate vector $\boldsymbol{g_i}$ in the following way:

$$\boldsymbol{f_i} = [\overleftrightarrow{\boldsymbol{s_i}} \circ \boldsymbol{D_{k-1}}; \overleftrightarrow{\boldsymbol{s_i}}; \boldsymbol{D_{k-1}}] \tag{10}$$

$$\boldsymbol{F_i} = \boldsymbol{W^{(2)}}\text{tanh}(\boldsymbol{W^{(1)}}\boldsymbol{f_i} + \boldsymbol{b^{(1)}}) + \boldsymbol{b^{(2)}} \tag{11}$$

$$\boldsymbol{g_i} = \frac{\exp(\boldsymbol{F_i})}{\sum_{j=1}^{n_s}\exp(\boldsymbol{F_j})} \tag{12}$$

where $\overleftrightarrow{\boldsymbol{s_i}}$ is the i-th sentence representation, $\boldsymbol{D_{k-1}}$ is the document representation from last iteration. Equation 5 now becomes:

$$\boldsymbol{h_i} = \boldsymbol{g_i} \circ \tilde{\boldsymbol{h_i}} + (1 - \boldsymbol{g_i}) \circ \boldsymbol{h_{i-1}} \tag{13}$$

We use this “selective reading module” to automatically decide to which extent the information of each sentence should be updated based on its relationship with the polished document. In this way, the modified GRU network can grasp more accurate information from the document.

4.2 Iterative Unit

After each sentence passes through the selective reading module, we wish to update the document representation $\boldsymbol{D_{k-1}}$ with the newly constructed sentence representations. The iterative unit (also depicted above in Fig.1) is designed for this purpose. We use a GRU$_{\text{iter}}$ cell to generate the polished document representation, whose input is the final state of the selective reading network from the previous iteration, $\boldsymbol{h_{n_s}}$ and whose initial state is set to the document representation of the previous iteration, $\boldsymbol{D_{k-1}}$. The updated document representation is computed by:

$$\boldsymbol{D_k} = \text{GRU}_{\text{iter}}(\boldsymbol{h_{n_s}}, \boldsymbol{D_{k-1}}) \tag{14}$$

4.3 Decoder

Next, we describe our decoders, which are depicted shaded in the right part of Fig.1. Following most sequence labeling task (Xue and Palmer, 2004; Carreras and Màrquez, 2005) where they learn a feature vector for each sentence, we use a bidirectional GRU$_{\text{dec}}$ network in each iteration to output features so as to calculate extracting probabilities. For k-th iteration, given the sentence representation $\overleftrightarrow{\boldsymbol{s}}$ as input and the document representation $\boldsymbol{D_k}$ as the initial state, our decoder encodes the features of all sentences in the hidden state $\boldsymbol{h^k} = \{\boldsymbol{h_0^k}, ..., \boldsymbol{h_{n_s}^k}\}$:

$$\boldsymbol{h_i^k} = \text{GRU}_{\text{dec}}(\overleftrightarrow{\boldsymbol{s}}, \boldsymbol{h_{i-1}^k}) \tag{15}$$

$$\boldsymbol{h_0^k} = \boldsymbol{D_k} \tag{16}$$

4.4 Sentence Labeling Module

Next, we use the feature of each sentence to generate corresponding extracting probability. Since we have one decoder in each iteration, if we directly transform the hidden states in each iteration to extracting probabilities, we will end up with several scores for each sentence. Either taking the average or summing them together by specific weights is inappropriate and inelegant. Hence, we concatenate hidden states of all decoders together and apply a multi-layer perceptron to them to generate the extracting probabilities:

$$\boldsymbol{y} = \boldsymbol{W^{(4)}}\text{tanh}(\boldsymbol{W^{(3)}}[\boldsymbol{h^1}; ...; \boldsymbol{h^k}] + \boldsymbol{b^{(3)}}) + \boldsymbol{b^{(4)}} \tag{17}$$

where $\boldsymbol{y} = \{y_1, ..., y_{n_s}\}$, y_i is the extracting probability for each setence. In this way, we let the model learn by itself how to utilize the outputs of all iterations and assign to each hidden state a reliable weight. In section 6, we will show that this labeling method outperforms other methods.

5 Experiment Setup

In this section, we present our experimental setup for training and estimating our summarization model. We first introduce the datasets used for training and evaluation, and then introduce our experimental details and evaluation protocol.

5.1 Datasets

In order to make a fair comparison with our baselines, we used the CNN/Dailymail corpus which was constructed by Hermann et al. (2015). We used the standard splits for training, validation and testing in each corpus (90,266/1,220/1,093 documents for CNN and 196,557/12,147/10,396 for DailyMail). We followed previous studies in using the human-written story highlight in each article as a gold-standard abstractive summary. These highlights were used to generate gold labels when training and testing our model using the greedy search method similar to (Nallapati et al., 2016a).

We also tested ITS on an out-of-domain corpus, DUC2002, which consists of 567 documents. Documents in this corpus belong to 59 various clusters and each cluster has a unique topic. Each document has two gold summaries written by human experts of length around 100 words.

5.2 Implementation Details

We implemented our model in Tensorflow (Abadi et al., 2016). The code for our models is available online[1]. We mostly followed the settings in (Nallapati et al., 2016a) and trained the model using the Adam optimizer (Kingma and Ba, 2014) with initial learning rate 0.001 and anneals of 0.5 every 6 epochs until reaching 30 epochs. We selected three sentences with highest scores as summary. After preliminary exploration, we found that arranging them according to their scores consistently achieved the best performance. Experiments were performed with a batch size of 64 documents. We used 100-dimension GloVe (Pennington et al., 2014) embeddings trained on Wikipedia 2014 as our embedding initialization with a vocabulary size limited to 100k for speed purposes. We initialized out-of-vocabulary word embeddings over a uniform distribution within [-0.2,0,2]. We also padded or cut sentences to contain exactly 70 words. Each GRU module had 1 layer with 200-dimensional hidden states and with either an initial state set up as described above or a random initial state. To prevent overfitting, we used dropout after each GRU network and embedding layer, and also applied L2 loss to all unbiased variables. The iteration number was set to 5 if not specified. A detailed discussion about iteration number can be found in section 7.

5.3 Baselines

On all datasets we used the Lead-3 method as a baseline, which simply chooses the first three sentences in a document as the gold summary. On DailyMail datasets, we report the performance of SummaRuNNer in (Nallapati et al., 2016a) and the model in (Cheng and Lapata, 2016), as well as a logistic regression classifier (LReg) that they used as a baseline. We reimplemented the Hybrid MemNet model in (Singh et al., 2017) as one of our baselines since they only reported the performance of 500 samples in their paper. Also, Narayan et al. (2018) released their code[2] for the REFRESH model, we used their code to produce Rouge recall scores on the DailyMail dataset as they only reported results on CNN/DailyMail joint dataset. Baselines on CNN dataset are similar. On DUC2002 corpus, we compare our model with several baselines such as Integer Linear Programming (ILR) and LReg. We also report the performance of the newest neural networks model including (Nallapati et al., 2016a; Cheng and Lapata, 2016; Singh et al., 2017).

5.4 Evaluation

In the evaluation procedure, we used the Rouge scores, i.e. Rouge-1, Rouge-2, and Rouge-L, corresponding to the matches of unigram, bigrams, and Longest Common Subsequence (LCS) respectively, to estimate our model. We obtained our Rouge scores using the standard pyrouge package[3]. To compare with other related works, we used full-length F1 score on the CNN corpus, limited length of 75 bytes and 275 bytes recall score on DailyMail corpus. As for the DUC2002 corpus, following the official guidelines, we examined the Rouge recall score at the length of 75 words. All results in our experiment are statistically significant using 95% confidence interval as estimated by Rouge script.

Schluter (2017) noted that only using the Rouge metric to evaluate summarization quality can be misleading. Therefore, we also evaluated our model using human evaluation. Five highly educated participants were asked to rank 40 summaries produced by four models: the Lead-3 baseline, Hybrid MemNet, ITS, and human-authored highlights. We chose Hybrid MemNet as one of the human evaluation baselines since its performance is relatively high compared to other baselines. Judging criteria included informativeness and coherence. Test cases were randomly sampled from DailyMail test set.

6 Experiment analysis

Table 1 shows the performance comparison of our model with other baselines on the DailyMail dataset with respect to Rouge score at 75 bytes and 275 bytes of summary length. Our model performs consistently and significantly better than other models on 75 bytes, while on 275 bytes, the improvement margin is smaller. One possible interpretation is that our model has high precision on top rank outputs, but the accuracy is lower for lower rank sentences. In addition, (Cheng and Lapata, 2016) used additional supervised training

---

[1] `https://github.com/yingtaomj/Iterative-Document-Representation-Learning-Towards-Summarization-with-Polishing`

[2] `https://github.com/EdinburghNLP/Refresh`

[3] `https://pypi.python.org/pypi/pyrouge/0.1.0`

DailyMail	b75			b275		
	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
Lead-3	21.9	7.2	11.6	40.5	14.9	32.6
LReg(500)	18.5	6.9	10.2	-	-	-
Cheng *et.al*'16	22.7	8.5	12.5	42.2	17.3	**34.8**
SummaRuNNer	26.2	10.8	14.4	42	16.9	34.1
REFRESH	24.1	11.5	12.5	40.3	15.1	32.9
Hybrid MemNet	26.3	11.2	15.5	41.4	16.7	33.2
ITS	**27.4**	**11.9**	**16.1**	**42.4**	**17.4**	34.1

Table 1: Comparison with other baselines on **DailyMail** test dataset using Rouge recall score with respect to the abstractive ground truth at 75 bytes and at 275 bytes.

CNN	Rouge-1	Rouge-2	Rouge-L
Lead-3	29.1	11.1	25.9
Cheng *et.al*'16	28.4	10.0	25.0
Hybrid MemNet	29.9	11.3	26.4
REFRESH	30.4	11.7	26.9
ITS	**30.8**	**12.6**	**27.6**

Table 2: Comparison with other baselines on **CNN** test dataset using full-length F1 variants of Rouge.

DUC2002	Rouge-1	Rouge-2	Rouge-L
Lead-3	43.6	21.0	40.2
LReg	43.8	20.7	40.3
ILP	45.4	21.3	42.8
Cheng *et.al*'16	47.4	23.0	**43.5**
SummaRuNNer	46.6	23.1	43.0
Hybrid MemNet	46.9	23.0	43.1
ITS	**47.6**	**23.4**	**43.5**

Table 3: Comparison with other baselines on **DUC2002** dataset using Rouge recall score with respect to the abstractive ground truth at 75 bytes.

Variations	Rouge-1	Rouge-2	Rouge-L
ITS	27.4	11.9	16.1
w/o selective reading	27.1	11.6	15.4
w/o iteration	26.9	11.6	15.8
w/o concatenation	27.2	11.7	15.9

Table 4: Ablation study on DailyMail test dataset with respect to the abstractive ground truth at 75 bytes.

to create sentence-level extractive labels to train their model, while our model uses an unsupervised greedy approximation instead.

We also examined the performance of our model on CNN dataset as listed in Table 2. To compare with other models, we used full-length Rouge F1 metric as reported by Narayan et al. (2018). Results demonstrate that our model has a consistently best performance on different datasets.

In Table 3, we present the performance of ITS on the out of domain DUC dataset. Our model outperforms or matches other basic models including LReg and ILR as well as neural network baselines such as SummaRuNNer with respect to the ground truth at 75 bytes, which shows that our model can be adapted to different copora maintaining high accuracy.

In order to explore the impact of internal structure of ITS, we also conducted an ablation study in Table 4. The first variation is the same model without the selective reading module. The second one sets the iteration number to one, that is, a model without iteration process. The last variation is to apply MLP on the output from the last iteration instead of concatenating the hidden states of all decoders. All other settings and parameters are the same. Performances of these models are worse than that of ITS in all metrics, which demonstrates the preeminence of ITS. More importantly, by this controlled experiment, we can verify the contribution of different module in ITS.

7 Further discussion

Analysis of iteration number: We did a broad sweep of experiments to further investigate the influence of iteration process on the generated summary quality. First, we studied the influence of iteration number. In order to make a fair comparison between models with different iteration number, we trained all models for same epochs without tuning. Fig.2 illustrates the relationship between iteration number and the Rouge score at 75 bytes of summary length on DailyMail test dataset. The result shows that the Rouge score increases with the number of iteration to begin with. After reaching the upper limit it begins to drop. Note that

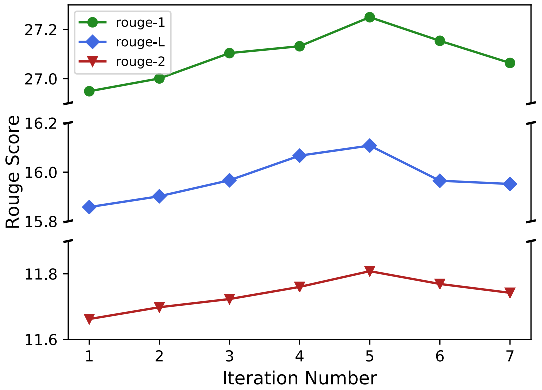

Figure 2: Relationship between number of iteration and Rouge score on DailyMail test dataset with respect to the ground truth at 75 bytes.

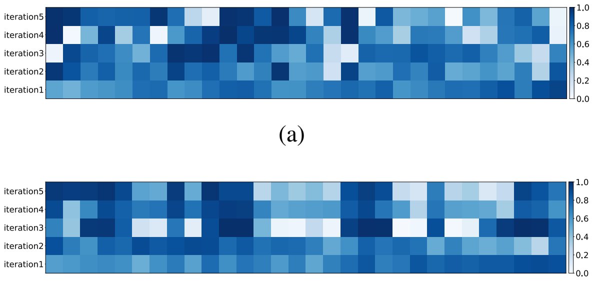

(a)

(b)

Figure 3: The predicted extracting probabilities for each sentence calculated by the output of each iteration.

Models	1st	2nd	3rd	4th
Lead-3	0.12	0.11	0.25	**0.52**
Hybrid MemNet	0.24	0.25	**0.28**	0.23
ITS	0.31	**0.34**	0.23	0.12
Gold	**0.33**	0.30	0.24	0.13

Table 5: System ranking comparison with other baselines on DailyMail corpus. Rank 1 is the best and Rank 4 is the worst. Each score represents the percentage of the summary under this rank.

the result of training the model for only one epoch outperforms the state-of-the-art in (Singh et al., 2017), which demonstrates that our selective reading module is effective. The fact that continuing this process increase the performance confirms that the iteration idea behind our model is useful in practice. Based on above observation, we set the default iteration number to be 5.

Analysis of polishing process: Next, to fully investigate how the iterative process influences the extracting results, we draw heatmaps of the extracting probabilities for each decoder at each iteration. We pick two representative cases in Fig.3, where the x-axis represents the sentence index and y-axis is the iteration number, x-axis labels are omitted. The darker the color is, the higher the extracting probability is. In Fig.3(a), it can be seen that when the iteration begins, most sentences have similar probabilities. As we increase the number of iteration, some probabilities begin to fall and others saturate. This means that the model already has preferred sentences to select. Another interesting feature we found is that there is a transitivity between iterations as shown in Fig.3(b). To be specific, the sentences which are not preferred by iteration 3 remain low probabilities in the next two iterations, while sentences with relatively high scores are still preferred by iteration 4 and 5.

Human Evaluation: We gave human evaluators three system-generated summaries, generated by Lead-3, Hybrid MemNet, ITS, as well as the human-written gold standard summary, and asked them to rank these summaries based on summary informativeness and coherence. Table 5 shows the percentages of summaries of different models under each rank scored by human experts. It is not surprising that gold standard has the most summaries of the highest quality. Our model has the most summaries under 2nd rank, thus can be considered 2nd best, following are Hybrid MemNet and Lead-3, as they are ranked mostly 3rd and 4th. By case study, we found that a number of summaries generated by Hybrid MemNet have two sentences the same as ITS out of three, however, the third distinct sentence from our model always leads to a better evaluation result considering overall informativeness and coherence. Readers can refer to the appendix to see our case study.

8 Conclusion

In this work, we introduce ITS, an iteration based extractive summarization model, inspired by the observation that it is often necessary for a human to read the article multiple times to fully understand and summarize it. Experimental results on CNN/DailyMail and DUC corpora demonstrate the effectiveness of our model.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. We would also like to thank Jin-ge Yao and Zhengyuan Ma for their valuable advice on this project. This work was supported by the National Key Research and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61876196, No. 61672058). Rui Yan was sponsored by CCF-Tencent Open Research Fund and Microsoft Research Asia (MSRA) Collaborative Research Program.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 152–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*.

Kevin Bretonnel Cohen. 2002. Natural language processing for online applications: Text retrieval, extraction and categorization (review). *Language*, 80(3):510–511.

Karl Moritz Hermann, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. pages 1693–1701.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

Ramesh Nallapati, Igor Melnyk, Abhishek Kumar, and Bowen Zhou. 2017. Sengen: Sentence generating neural variational topic model.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016a. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira Dos Santos, Caglar Gulcehre, and Bing Xiang. 2016b. Abstractive text summarization using sequence-to-sequence rnns and beyond.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning.

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

Dragomir R Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, et al. 2004. Mead-a platform for multidocument multilingual text summarization. In *LREC*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *Computer Science*.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 41–45.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

Abhishek Kumar Singh, Manish Gupta, and Vasudeva Varma. 2017. Hybrid memnet for extractive summarization. pages 2303–2306.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *Computer Science*.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Meeting of the Association for Computational Linguistics*, pages 565–574.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. *arXiv preprint arXiv:1804.07036*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *IJCAI*, pages 2238–2244.

Rui Yan, Cheng Te Li, Xiaohua Hu, and Ming Zhang. 2016. Chinese couplet generation with neural network structures. In *Meeting of the Association for Computational Linguistics*, pages 2347–2357.

Rui Yan, Jian Yun Nie, and Xiaoming Li. 2011a. Summarize what you are interested in: An optimization framework for interactive personalized summarization. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John Mcintyre Conference Centre, Edinburgh, Uk, A Meeting of Sigdat, A Special Interest Group of the ACL*, pages 1342–1351.

Rui Yan, Xiaojun Wan, Mirella Lapata, Wayne Xin Zhao, Pu-Jen Cheng, and Xiaoming Li. 2012. Visualizing timelines: Evolutionary summarization via iterative reinforcement between text and image streams. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 275–284. ACM.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011b. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 745–754.

Bottom-Up Abstractive Summarization

Sebastian Gehrmann **Yuntian Deng** **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
`{gehrmann, dengyuntian, srush}@seas.harvard.edu`

Abstract

Neural network-based methods for abstractive summarization produce outputs that are more fluent than other techniques, but which can be poor at content selection. This work proposes a simple technique for addressing this issue: use a data-efficient content selector to over-determine phrases in a source document that should be part of the summary. We use this selector as a *bottom-up* attention step to constrain the model to likely phrases. We show that this approach improves the ability to compress text, while still generating fluent summaries. This two-step process is both simpler and higher performing than other end-to-end content selection models, leading to significant improvements on ROUGE for both the CNN-DM and NYT corpus. Furthermore, the content selector can be trained with as little as 1,000 sentences, making it easy to transfer a trained summarizer to a new domain.

Source Document

german chancellor angela merkel [did] not [look] too pleased about the weather during her [annual] easter holiday [in italy.] as britain [basks] in [sunshine] and temperatures of up to 21c, mrs merkel and her husband[, chemistry professor joachim sauer,] had to settle for a measly 12 degrees. the chancellor and her [spouse] have been spending easter on the small island of ischia, near naples in the mediterranean for over a [decade.]

[not so sunny:] angela merkel [and] her husband[, chemistry professor joachim sauer,] are spotted on their [annual] easter trip to the island of ischia[,] near naples[. the] couple [traditionally] spend their holiday at the five-star miramare spa hotel on the south of the island [, which comes] with its own private beach [, and balconies overlooking the] ocean [.]...

Reference

- angela merkel and husband spotted while on italian island holiday.

  . . .

Baseline Approach

- angela merkel and her husband, chemistry professor joachim sauer, are spotted on their annual easter trip to the island of ischia, near naples.

  . . .

Bottom-Up Summarization

- angela merkel and her husband are spotted on their easter trip to the island of ischia, near naples.

  . . .

Figure 1: Example of two sentence summaries with and without bottom-up attention. The model does not allow copying of words in [gray], although it can generate words. With bottom-up attention, we see more explicit sentence compression, while without it whole sentences are copied verbatim.

1 Introduction

Text summarization systems aim to generate natural language summaries that compress the information in a longer text. Approaches using neural networks have shown promising results on this task with end-to-end models that encode a source document and then decode it into an abstractive summary. Current state-of-the-art neural abstractive summarization models combine extractive and abstractive techniques by using pointer-generator style models which can copy words from the source document (Gu et al., 2016; See et al., 2017). These end-to-end models produce fluent abstractive summaries but have had mixed success in content selection, i.e. deciding what to summarize, compared to fully extractive models.

There is an appeal to end-to-end models from a modeling perspective; however, there is evidence that when summarizing people follow a two-step approach of first selecting important phrases and then paraphrasing them (Anderson and Hidi, 1988; Jing and McKeown, 1999). A similar argument has been made for image captioning. Anderson et al. (2017) develop a state-of-the-art model with a two-step approach that first pre-computes bounding boxes of segmented objects and then applies attention to these regions. This so-called *bottom-up* attention is inspired by neuroscience research describing attention based on properties in-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4098–4109
Brussels, Belgium, October 31 - November 4, 2018.

herent to a stimulus (Buschman and Miller, 2007).

Motivated by this approach, we consider *bottom-up* attention for neural abstractive summarization. Our approach first selects a selection mask for the source document and then constrains a standard neural model by this mask. This approach can better decide which phrases a model should include in a summary, without sacrificing the fluency advantages of neural abstractive summarizers. Furthermore, it requires much fewer data to train, which makes it more adaptable to new domains.

Our full model incorporates a separate content selection system to decide on relevant aspects of the source document. We frame this selection task as a sequence-tagging problem, with the objective of identifying tokens from a document that are part of its summary. We show that a content selection model that builds on contextual word embeddings (Peters et al., 2018) can identify correct tokens with a recall of over 60%, and a precision of over 50%. To incorporate bottom-up attention into abstractive summarization models, we employ masking to constrain copying words to the selected parts of the text, which produces grammatical outputs. We additionally experiment with multiple methods to incorporate similar constraints into the training process of more complex end-to-end abstractive summarization models, either through multi-task learning or through directly incorporating a fully differentiable mask.

Our experiments compare bottom-up attention with several other state-of-the-art abstractive systems. Compared to our baseline models of See et al. (2017) bottom-up attention leads to an improvement in ROUGE-L score on the CNN-Daily Mail (CNN-DM) corpus from 36.4 to 38.3 while being simpler to train. We also see comparable or better results than recent reinforcement-learning based methods with our MLE trained system. Furthermore, we find that the content selection model is very data-efficient and can be trained with less than 1% of the original training data. This provides opportunities for domain-transfer and low-resource summarization. We show that a summarization model trained on CNN-DM and evaluated on the NYT corpus can be improved by over 5 points in ROUGE-L with a content selector trained on only 1,000 in-domain sentences.

2 Related Work

There is a tension in document summarization between staying close to the source document and allowing compressive or abstractive modification. Many non-neural systems take a select and compress approach. For example, Dorr et al. (2003) introduced a system that first extracts noun and verb phrases from the first sentence of a news article and uses an iterative shortening algorithm to compress it. Recent systems such as Durrett et al. (2016) also learn a model to select sentences and then compress them.

In contrast, recent work in neural network based data-driven extractive summarization has focused on extracting and ordering full sentences (Cheng and Lapata, 2016; Dlikman and Last, 2016). Nallapati et al. (2016b) use a classifier to determine whether to include a sentence and a selector that ranks the positively classified ones. These methods often over-extract, but extraction at a word level requires maintaining grammatically correct output (Cheng and Lapata, 2016), which is difficult. Interestingly, key phrase extraction while ungrammatical often matches closely in content with human-generated summaries (Bui et al., 2016).

A third approach is neural abstractive summarization with sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2014). These methods have been applied to tasks such as headline generation (Rush et al., 2015) and article summarization (Nallapati et al., 2016a). Chopra et al. (2016) show that attention approaches that are more specific to summarization can further improve the performance of models. Gu et al. (2016) were the first to show that a copy mechanism, introduced by Vinyals et al. (2015), can combine the advantages of both extractive and abstractive summarization by copying words from the source. See et al. (2017) refine this pointer-generator approach and use an additional coverage mechanism (Tu et al., 2016) that makes a model aware of its attention history to prevent repeated attention.

Most recently, reinforcement learning (RL) approaches that optimize objectives for summarization other than maximum likelihood have been shown to further improve performance on these tasks (Paulus et al., 2017; Li et al., 2018b; Celikyilmaz et al., 2018). Paulus et al. (2017) approach the coverage problem with an intra-attention in which a decoder has an attention over previously generated words. However RL-based training can be difficult to tune and slow to train. Our method does not utilize RL training, although in theory this approach can be adapted to RL methods.

Several papers also explore multi-pass extractive-abstractive summarization. Nallapati et al. (2017) create a new source document comprised of the important sentences from the source and then train an abstractive system. Liu et al. (2018) describe an extractive phase that extracts full paragraphs and an abstractive one that determines their order. Finally Zeng et al. (2016) introduce a mechanism that reads a source document in two passes and uses the information from the first pass to bias the second. Our method differs in that we utilize a completely abstractive model, biased with a powerful content selector.

Other recent work explores alternative approaches to content selection. For example, Cohan et al. (2018) use a hierarchical attention to detect relevant sections in a document, Li et al. (2018a) generate a set of keywords that is used to guide the summarization process, and Pasunuru and Bansal (2018) develop a loss-function based on whether salient keywords are included in a summary. Other approaches investigate the content-selection at the sentence-level. Tan et al. (2017) describe a graph-based attention to attend to one sentence at a time, Chen and Bansal (2018) first extract full sentences from a document and then compress them, and Hsu et al. (2018) modulate the attention based on how likely a sentence is included in a summary.

3 Background: Neural Summarization

Throughout this paper, we consider a set of pairs of texts $(\mathcal{X}, \mathcal{Y})$ where $x \in \mathcal{X}$ corresponds to source tokens $x_1, \ldots, x_n$ and $y \in \mathcal{Y}$ to a summary $y_1, \ldots, y_m$ with $m \ll n$.

Abstractive summaries are generated one word at a time. At every time-step, a model is aware of the previously generated words. The problem is to learn a function $f(x)$ parametrized by θ that maximizes the probability of generating the correct sequences. Following previous work, we model the abstractive summarization with an attentional sequence-to-sequence model. The attention distribution $p(a_j|x, y_{1:j-1})$ for a decoding step j, calculated within the neural network, represents an embedded soft distribution over all of the source tokens and can be interpreted as the current focus of the model.

The model additionally has a copy mechanism (Vinyals et al., 2015) to copy words from the source. Copy models extend the decoder by predicting a binary soft switch z_j that determines whether the model copies or generates. The copy distribution is a probability distribution over the source text, and the joint distribution is computed as a convex combination of the two parts of the model,

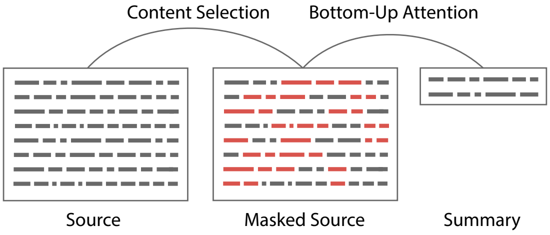

Figure 2: Overview of the selection and generation processes described throughout Section 4.

$$\begin{aligned} &p(y_j \mid y_{1:j\text{-}1}, x) = \\ &p(z_j = 1 \mid y_{1:j\text{-}1}, x) \times p(y_j \mid z_j = 1, y_{1:j\text{-}1}, x) + \\ &p(z_j = 0 \mid y_{1:j\text{-}1}, x) \times p(y_j \mid z_j = 0, y_{1:j\text{-}1}, x) \end{aligned} \tag{1}$$

where the two parts represent copy and generation distribution respectively. Following the *pointer-generator* model of See et al. (2017), we reuse the attention $p(a_j|x, y_{1:j-1})$ distribution as copy distribution, i.e. the copy probability of a token in the source w through the copy attention is computed as the sum of attention towards all occurrences of w. During training, we maximize marginal likelihood with the latent switch variable.

4 Bottom-Up Attention

We next consider techniques for incorporating a content selection into abstractive summarization, illustrated in Figure 2.

4.1 Content Selection

We define the content selection problem as a word-level extractive summarization task. While there has been significant work on custom extractive summarization (see related work), we make a simplifying assumption and treat it as a sequence tagging problem. Let $t_1, \ldots, t_n$ denote binary tags for each of the source tokens, i.e. 1 if a word is copied in the target sequence and 0 otherwise.

While there is no supervised data for this task, we can generate training data by aligning the summaries to the document. We define a word x_i as copied if (1) it is part of the longest possible subsequence of tokens $s = x_{i-j:i:i+k}$, for integers $j \leq i; k \leq (n-i)$, if $s \in x$ and $s \in y$, and (2) there exists no earlier sequence u with $s = u$.

We use a standard bidirectional LSTM model trained with maximum likelihood for the sequence labeling problem. Recent results have shown that better word representations can lead to significantly improved performance in sequence tagging tasks (Peters et al., 2017). Therefore, we first map each token w_i into two embedding channels $e_i^{(\mathrm{w})}$ and $e_i^{(\mathrm{c})}$. The $e^{(\mathrm{w})}$ embedding represents a static channel of pre-trained word embeddings, e.g. GLoVE (Pennington et al., 2014). The $e^{(\mathrm{c})}$ are contextual embeddings from a pretrained language model, e.g. ELMo (Peters et al., 2018) which uses a character-aware token embedding (Kim et al., 2016) followed by two bidirectional LSTM layers $h_i^{(1)}$ and $h_i^{(2)}$. The contextual embeddings are fine-tuned to learn a task-specific embedding $e_i^{(\mathrm{c})}$ as a linear combination of the states of each LSTM layer and the token embedding,

$$e_i^{(\mathrm{c})} = \gamma \times \sum_{\ell=0}^{2} s_j \times h_i^{(\ell)},$$

with γ and $s_{0,1,2}$ as trainable parameters. Since these embeddings only add four additional parameters to the tagger, it remains very data-efficient despite the high-dimensional embedding space.

Both embeddings are concatenated into a single vector that is used as input to a bidirectional LSTM, which computes a representation h_i for a word w_i. We can then calculate the probability q_i that the word is selected as $\sigma(W_s h_i + b_s)$ with trainable parameters W_s and b_s.

4.2 Bottom-Up Copy Attention

Inspired by work in bottom-up attention for images (Anderson et al., 2017) which restricts attention to predetermined bounding boxes within an image, we use these attention masks to limit the available selection of the pointer-generator model.

As shown in Figure 1, a common mistake made by neural copy models is copying very long sequences or even whole sentences. In the baseline model, over 50% of copied tokens are part of copy sequences that are longer than 10 tokens, whereas this number is only 10% for reference summaries. While bottom-up attention could also be used to modify the source encoder representations, we found that a standard encoder over the full text was effective at aggregation and therefore limit the bottom-up step to attention masking.

Concretely, we first train a pointer-generator model on the full dataset as well as the content selector defined above. At inference time, to generate the mask, the content selector computes selection probabilities $q_{1:n}$ for each token in a source document. The selection probabilities are used to modify the copy attention distribution to only include tokens identified by the selector. Let a_j^i denote the attention at decoding step j to encoder word i. Given a threshold ϵ, the selection is applied as a hard mask, such that

$$p(\tilde{a}_j^i | x, y_{1:j-1}) = \begin{cases} p(a_j^i | x, y_{1:j-1}) & q_i > \epsilon \\ 0 & \text{ow.} \end{cases}$$

To ensure that Eq. 1 still yields a correct probability distribution, we first multiply $p(\tilde{a}_j | x, y_{1:j-1})$ by a normalization parameter λ and then renormalize the distribution. The resulting normalized distribution can be used to directly replace a as the new copy probabilities.

4.3 End-to-End Alternatives

Two-step BOTTOM-UP attention has the advantage of training simplicity. In theory, though, standard copy attention should be able to learn how to perform content selection as part of the end-to-end training. We consider several other end-to-end approaches for incorporating content selection into neural training.

Method 1: (MASK ONLY): We first consider whether the alignment used in the bottom-up approach could help a standard summarization system. Inspired by Nallapati et al. (2017), we investigate whether aligning the summary and the source during training and fixing the gold copy attention to pick the "correct" source word is beneficial. We can think of this approach as limiting the set of possible copies to a fixed source word. Here the training is changed, but no mask is used at test time.

Method 2 (MULTI-TASK): Next, we investigate whether the content selector can be trained alongside the abstractive system. We first test this hypothesis by posing summarization as a multi-task problem and training the tagger and summarization model with the same features. For this setup, we use a shared encoder for both abstractive summarization and content selection. At test time, we apply the same masking method as bottom-up attention.

Method 3 (DIFFMASK): Finally we consider training the full system end-to-end with the mask during training. Here we jointly optimize both objectives, but use predicted selection probabilities to softly mask the copy attention $p(\tilde{a}_j^i|x, y_{1:j-1}) = p(a_j^i|x, y_{1:j-1}) \times q_i$, which leads to a fully differentiable model. This model is used with the same soft mask at test time.

5 Inference

Several authors have noted that longer-form neural generation still has significant issues with incorrect length and repeated words than in short-form problems like translation. Proposed solutions include modifying models with extensions such as a coverage mechanism (Tu et al., 2016; See et al., 2017) or intra-sentence attention (Cheng et al., 2016; Paulus et al., 2017). We instead stick to the theme of modifying inference, and modify the scoring function to include a length penalty lp and a coverage penalty cp, and is defined as $s(x, y) = \log p(y|x)/lp(x) + cp(x; y)$.

Length: To encourage the generation of longer sequences, we apply length normalizations during beam search. We use the length penalty by Wu et al. (2016), which is formulated as

$$lp(y) = \frac{(5 + |y|)^\alpha}{(5 + 1)^\alpha},$$

with a tunable parameter α, where increasing α leads to longer summaries. We additionally set a minimum length based on the training data.

Repeats: Copy models often repeatedly attend to the same source tokens, generating the same phrase multiple times. We introduce a new summary specific coverage penalty,

$$cp(x; y) = \beta \left(-n + \sum_{i=1}^{n} \max\left(1.0, \sum_{j=1}^{m} a_i^j\right)\right).$$

Intuitively, this penalty increases whenever the decoder directs more than 1.0 of total attention within a sequence towards a single encoded token. By selecting a sufficiently high β, this penalty blocks summaries whenever they would lead to repetitions. Additionally, we follow (Paulus et al., 2017) and restrict the beam search to never repeat trigrams.

6 Data and Experiments

We evaluate our approach on the CNN-DM corpus (Hermann et al., 2015; Nallapati et al., 2016a), and the NYT corpus (Sandhaus, 2008), which are both standard corpora for news summarization. The summaries for the CNN-DM corpus are bullet points for the articles shown on their respective websites, whereas the NYT corpus contains summaries written by library scientists. CNN-DM summaries are full sentences, with on average 66 tokens ($\sigma = 26$) and 4.9 bullet points. NYT summaries are not always complete sentences and are shorter, with on average 40 tokens ($\sigma = 27$) and 1.9 bullet points. Recent work has used both the anonymized and the non-anonymized versions of on CNN-DM, so direct comparison can be difficult. Following See et al. (2017), we use the non-anonymized version of this corpus and truncate source documents to 400 tokens and the target summaries to 100 tokens in training and validation sets. For experiments with the NYT corpus, we use the preprocessing described by Paulus et al. (2017), and additionally remove author information and truncate source documents to 400 tokens instead of 800. These changes lead to an average of 326 tokens per article, a decrease from the 549 tokens with 800 token truncated articles. The target (non-copy) vocabulary is limited to 50,000 tokens for all models.

The content selection model uses pre-trained GloVe embeddings of size 100, and ELMo with size 1024. The bi-LSTM has two layers and a hidden size of 256. Dropout is set to 0.5, and the model is trained with Adagrad, an initial learning rate of 0.15, and an initial accumulator value of 0.1. We limit the number of training examples to 100,000 on either corpus, which only has a small impact on performance. For the jointly trained content selection models, we use the same configuration as the abstractive model.

For the base model, we re-implemented the Pointer-Generator model as described by See et al. (2017). To have a comparable number of parameters to previous work, we use an encoder with 256 hidden states for both directions in the one-layer LSTM, and 512 for the one-layer decoder. The embedding size is set to 128. We found that increasing model size or changing the model to the Transformer (Vaswani et al., 2017) can lead to slightly improved performance, but at the cost of increased training time and parameters. The

Method	R-1	R-2	R-L
Pointer-Generator (See et al., 2017)	36.44	15.66	33.42
Pointer-Generator + Coverage (See et al., 2017)	39.53	17.28	36.38
ML + Intra-Attention (Paulus et al., 2017)	38.30	14.81	35.49
ML + RL (Paulus et al., 2017)	39.87	15.82	36.90
Saliency + Entailment reward (Pasunuru and Bansal, 2018)	40.43	18.00	37.10
Key information guide network (Li et al., 2018a)	38.95	17.12	35.68
Inconsistency loss (Hsu et al., 2018)	40.68	17.97	37.13
Sentence Rewriting (Chen and Bansal, 2018)	40.88	17.80	**38.54**
Pointer-Generator (our implementation)	36.25	16.17	33.41
Pointer-Generator + Coverage Penalty	39.12	17.35	36.12
Pointer-Generator + Mask Only	37.70	15.63	35.49
Pointer-Generator + Multi-Task	37.67	15.59	35.47
Pointer-Generator + DiffMask	38.45	16.88	35.81
Bottom-Up Summarization	**41.22**	**18.68**	38.34

Table 1: Results of abstractive summarizers on the CNN-DM dataset.[2] The first section shows encoder-decoder abstractive baselines trained with cross-entropy. The second section describes reinforcement-learning based approaches. The third section presents our baselines and the attention masking methods described in this work.

model is trained with the same Adagrad configuration as the content selector. Additionally, the learning rate halves after each epoch once the validation perplexity does not decrease after an epoch. We do not use dropout and use gradient-clipping with a maximum norm of 2.

All inference parameters are tuned on a 200 sentence subset of the validation set. Length penalty parameter α and copy mask ϵ differ across models and baselines, with α ranging from 0.6 to 1.4, and ϵ ranging from 0.1 to 0.2. The minimum length of the generated summary is set to 35 for CNN-DM and 6 for NYT. While the Pointer-Generator uses a beam size of 5 and does not improve with a larger beam, we found that bottom-up attention requires a larger beam size and set it to 10. The coverage penalty parameter β is set to 10, and the copy attention normalization parameter λ to 2 for both approaches.

We use AllenNLP (Gardner et al., 2018) for the content selector, and the abstractive models are implemented in OpenNMT-py (Klein et al., 2017).[3].

7 Results

Table 1 shows our main results on the CNN-DM corpus, with abstractive models shown in the top, and bottom-up attention methods at the bottom. We first observe that using a coverage inference penalty scores the same as a full coverage mechanism, without requiring any additional model parameters. We found that none of our end-to-end models lead to improvements, indicating that it is difficult to apply the masking during training without hurting the training process. The *Mask Only* model with increased supervision on the copy mechanism performs very similar to the *Multi-Task* model. On the other hand, bottom-up attention leads to a major improvement across all three scores. While we would expect better content selection to primarily improve ROUGE-1, the fact all three increase hints that the fluency is not being hurt specifically. Our cross-entropy trained approach even outperforms all of the reinforcement-learning based approaches in ROUGE-1 and 2, while the highest reported ROUGE-L score by Chen and Bansal (2018) falls within the 95% confidence interval of our results.

Table 2 shows experiments with the same systems on the NYT corpus. We see that the 2 point improvement compared to the baseline Pointer-Generator maximum-likelihood approach carries over to this dataset. Here, the model outperforms

---

[3] Code and reproduction instructions can be found at `https://github.com/sebastianGehrmann/bottom-up-summary`

[3] These results compare on the non-anonymized version of this corpus used by (See et al., 2017). The best results on the anonymized version are R1:41.69 R2:19.47 RL:37.92 from (Celikyilmaz et al., 2018). We compare to their DCA model on the NYT corpus.

Method	R-1	R-2	R-L
ML*	44.26	27.43	40.41
ML+RL*	47.03	30.72	**43.10**
DCA[†]	**48.08**	31.19	42.33
Point.Gen. + Coverage Pen.	45.13	30.13	39.67
Bottom-Up Summarization	47.38	**31.23**	41.81

Table 2: Results on the NYT corpus, where we compare to RL trained models. * marks models and results by Paulus et al. (2017), and † results by Celikyilmaz et al. (2018).

the RL based model by Paulus et al. (2017) in ROUGE-1 and 2, but not L, and is comparable to the results of (Celikyilmaz et al., 2018) except for ROUGE-L. The same can be observed when comparing ML and our Pointer-Generator. We suspect that a difference in summary lengths due to our inference parameter choices leads to this difference, but did not have access to their models or summaries to investigate this claim. This shows that a bottom-up approach achieves competitive results even to models that are trained on summary-specific objectives.

The main benefit of bottom-up summarization seems to be from the reduction of mistakenly copied words. With the best Pointer-Generator models, the precision of copied words is 50.0% compared to the reference. This precision increases to 52.8%, which mostly drives the increase in R1. An independent-samples t-test shows that this improvement is statistically significant with t=14.7 ($p < 10^{-5}$). We also observe a decrease in average sentence length of summaries from 13 to 12 words when adding content selection compared to the Pointer-Generator while holding all other inference parameters constant.

Domain Transfer While end-to-end training has become common, there are benefits to a two-step method. Since the content selector only needs to solve a binary tagging problem with pretrained vectors, it performs well even with very limited training data. As shown in Figure 3, with only 1,000 sentences, the model achieves an AUC of over 74. Beyond that size, the AUC of the model increases only slightly with increasing training data.

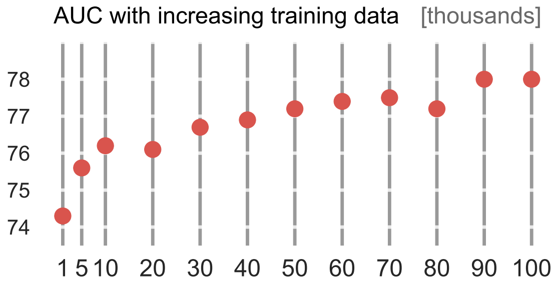

Figure 3: The AUC of the content selector trained on CNN-DM with different training set sizes ranging from 1,000 to 100,000 data points.

	AUC	R-1	R-2	R-L
CNNDM		25.63	11.40	20.55
+1k	80.7	30.62	16.10	25.32
+10k	83.6	32.07	17.60	26.75
+100k	86.6	33.11	18.57	27.69

Table 3: Results of the domain transfer experiment. AUC numbers are shown for content selectors. ROUGE scores represent an abstractive model trained on CNN-DM and evaluated on NYT, with additional copy constraints trained on 1/10/100k training examples of the NYT corpus.

To further evaluate the content selection, we consider an application to domain transfer. In this experiment, we apply the Pointer-Generator trained on CNN-DM to the NYT corpus. In addition, we train three content selectors on 1, 10, and 100 thousand sentences of the NYT set, and use these in the bottom-up summarization. The results, shown in Table 3, demonstrates that even a model trained on the smallest subset leads to an improvement of almost 5 points over the model without bottom-up attention. This improvement increases with the larger subsets to up to 7 points. While this approach does not reach a comparable performance to models trained directly on the NYT dataset, it still represents a significant increase over the not-augmented CNN-DM model and produces summaries that are quite readable. We show two example summaries in Appendix A. This technique could be used for low-resource domains and for problems with limited data availability.

8 Analysis and Discussion

Extractive Summary by Content Selection? Given that the content selector is effective in conjunction with the abstractive model, it is interesting to know whether it has learned an effective extractive summarization system on its own. Table 4 shows experiments comparing content selec-

Method	R-1	R-2	R-L
LEAD-3	40.1	17.5	36.3
NEUSUM (Zhou et al., 2018)	41.6	19.0	38.0
Top-3 sents (Cont. Select.)	40.7	**18.0**	37.0
Oracle Phrase-Selector	67.2	37.8	58.2
Content Selector	**42.0**	15.9	**37.3**

Table 4: Results of extractive approaches on the CNN-DM dataset. The first section shows sentence-extractive scores. The second section first shows an oracle score if the content selector selected all the correct words according to our matching heuristic. Finally, we show results when the Content Selector extracts all phrases above a selection probability threshold.

tion to extractive baselines. The LEAD-3 baseline is a commonly used baseline in news summarization that extracts the first three sentences from an article. Top-3 shows the performance when we extract the top three sentences by average copy probability from the selector. Interestingly, with this method, only 7.1% of the top three sentences are not within the first three, further reinforcing the strength of the LEAD-3 baseline. Our naive sentence-extractor performs slightly worse than the highest reported extractive score by Zhou et al. (2018) that is specifically trained to score combinations of sentences. The final entry shows the performance when all the words above a threshold are extracted such that the resulting summaries are approximately the length of reference summaries. The oracle score represents the results if our model had a perfect accuracy, and shows that the content selector, while yielding competitive results, has room for further improvements in future work.

This result shows that the model is quite effective at finding important words (ROUGE-1) but less effective at chaining them together (ROUGE-2). Similar to Paulus et al. (2017), we find that the decrease in ROUGE-2 indicates a lack of fluency and grammaticality of the generated summaries. A typical example looks like this:

> a man food his first hamburger wrongfully for 36 years. michael hanline, 69, was convicted of murder for the shooting of truck driver jt mcgarry in 1980 on judge charges.

This particular ungrammatical example has a ROUGE-1 of 29.3. This further highlights the benefit of the combined approach where bottom-up predictions are chained together fluently by the abstractive system. However, we also note that the abstractive system requires access to the full source document. Distillation experiments in which we tried to use the output of the content-selection as training-input to abstractive models showed a drastic decrease in model performance.

Data	%Novel	Verb	Noun	Adj
Reference	14.8	30.9	35.5	12.3
Vanilla S2S	6.6	14.5	19.7	5.1
Pointer-Generator	2.2	25.7	39.3	13.9
Bottom-Up Attention	0.5	53.3	24.8	6.5

Table 5: %Novel shows the percentage of words in a summary that are not in the source document. The last three columns show the part-of-speech tag distribution of the novel words in generated summaries.

Analysis of Copying While Pointer-Generator models have the ability to abstract in summary, the use of a copy mechanism causes the summaries to be mostly extractive. Table 5 shows that with copying the percentage of generated words that are not in the source document decreases from 6.6% to 2.2%, while reference summaries are much more abstractive with 14.8% novel words. Bottom-up attention leads to a further reduction to only a half percent. However, since generated summaries are typically not longer than 40-50 words, the difference between an abstractive system with and without bottom-up attention is less than one novel word per summary. This shows that the benefit of abstractive models has been less in their ability to produce better paraphrasing but more in the ability to create fluent summaries from a mostly extractive process.

Table 5 also shows the part-of-speech-tags of the novel generated words, and we can observe an interesting effect. Application of bottom-up attention leads to a sharp decrease in novel adjectives and nouns, whereas the fraction of novel words that are verbs sharply increases. When looking at the novel verbs that are being generated, we notice a very high percentage of tense or number changes, indicated by variation of the word "say", for example "said" or "says", while novel nouns are mostly morphological variants of words in the source.

Figure 4 shows the length of the phrases that are being copied. While most copied phrases in the

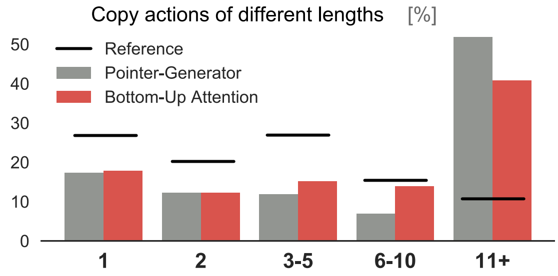

Figure 4: For all copied words, we show the distribution over the length of copied phrases they are part of. The black lines indicate the reference summaries, and the bars the summaries with and without bottom-up attention.

reference summaries are in groups of 1 to 5 words, the Pointer-Generator copies many very long sequences and full sentences of over 11 words. Since the content selection mask interrupts most long copy sequences, the model has to either generate the unselected words using only the generation probability or use a different word instead. While we observed both cases quite frequently in generated summaries, the fraction of very long copied phrases decreases. However, either with or without bottom-up attention, the distribution of the length of copied phrases is still quite different from the reference.

Inference Penalty Analysis We next analyze the effect of the inference-time loss functions. Table 6 presents the marginal improvements over the simple Pointer-Generator when adding one penalty at a time. We observe that all three penalties improve all three scores, even when added on top of the other two. This further indicates that the unmodified Pointer-Generator model has already learned an appropriate representation of the abstractive summarization problem, but is limited by ineffective content selection and inference methods.

9 Conclusion

This work presents a simple but accurate content selection model for summarization that identifies phrases within a document that are likely included in its summary. We showed that this content selector can be used for a bottom-up attention that restricts the ability of abstractive summarizers to copy words from the source. The combined bottom-up summarization system leads to improvements in ROUGE scores of over two points on both the CNN-DM and NYT corpora. A comparison to end-to-end trained methods showed that this particular problem cannot be easily solved with a single model, but instead requires fine-tuned inference restrictions. Finally, we showed that this technique, due to its data-efficiency, can be used to adjust a trained model with few data points, making it easy to transfer to a new domain. Preliminary work that investigates similar bottom-up approaches in other domains that require a content selection, such as grammar correction, or data-to-text generation, have shown some promise and will be investigated in future work.

Data	R-1	R-2	R-L
Pointer Generator	36.3	16.2	33.4
+ Length Penalty	38.0	16.8	35.0
+ Coverage Penalty	38.9	17.2	35.9
+ Trigram Repeat	39.1	17.4	36.1

Table 6: Results on CNN-DM when adding one inference penalty at a time.

Acknowledgements

We would like to thank Barbara J. Grosz for helpful discussions and feedback on early stages of this work. We further thank the three anonymous reviewers. This work was supported by a Samsung Research Award. YD was funded in part by a Bloomberg Research Award. SG was funded in part by NIH grant 5R01CA204585-02.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2017. Bottom-up and top-down attention for image captioning and vqa. *arXiv preprint arXiv:1707.07998*.

Valerie Anderson and Suzanne Hidi. 1988. Teaching students to summarize. *Educational leadership*, 46(4):26–28.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Duy Duc An Bui, Guilherme Del Fiol, John F Hurdle, and Siddhartha Jonnalagadda. 2016. Extractive text summarization system to aid data extraction from full text in systematic review development. *Journal of biomedical informatics*, 64:265–272.

Timothy J Buschman and Earl K Miller. 2007. Top-down versus bottom-up control of attention in the prefrontal and posterior parietal cortices. *science*, 315(5820):1860–1862.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1662–1675.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 615–621.

Alexander Dlikman and Mark Last. 2016. Using machine learning methods and linguistic features in single-document extractive summarization. In *DMNLP@ PKDD/ECML*, pages 1–8.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*, pages 1–8. Association for Computational Linguistics.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. *arXiv preprint arXiv:1603.08887*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. *arXiv preprint arXiv:1805.06266*.

Hongyan Jing and Kathleen R McKeown. 1999. The decomposition of human-written summary sentences. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 129–136.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018a. Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 55–60.

Piji Li, Lidong Bing, and Wai Lam. 2018b. Actor-critic based training framework for abstractive summarization. *arXiv preprint arXiv:1803.11070*.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016a. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. 2016b. Classify or select: Neural architectures for extractive document summarization. *arXiv preprint arXiv:1611.04244*.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, Volume 2 (Short Papers), volume 2, pages 646–653.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1171–1181.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–663.

Examples	Generated summary
Reference	green bay packers successful season is largely due to quarterback brett favre
S2S	ahman green rushed for 000 yards in 00-00 victory over the giants . true , dorsey levens , good enough to start for most teams but now green 's backup , contributed kickoff returns of 00 , 00 and 00 yards .
Content Selection	playoff-bound green bay packers beat the giants in the 00-00 victory . the packers won three games and six of each other .
Reference	paul byers , pioneer of visual anthropology , dies at age 00
S2S	paul byers , an early practitioner of mead , died on dec. 00 at his home in manhattan . he enlisted in the navy , which trained him as a cryptanalyst and stationed him in australia .
Content Selection	paul byers , an early practitioner of anthropology , pioneered with margaret mead .

Table 7: Domain-transfer examples.

A Domain Transfer Examples

We present two generated summaries for the CNN-DM to NYT domain transfer experiment in Table 7. S2S refers to a Pointer-Generator with Coverage Penalty trained on CNN-DM that scores 20.6 ROUGE-L on the NYT dataset. The content-selection improves this to 27.7 ROUGE-L without any fine-tuning of the S2S model.

Controlling Length in Abstractive Summarization Using a Convolutional Neural Network

Yizhu Liu[1] **Zhiyi Luo**[2] **Kenny Q. Zhu**[3]
Shanghai Jiao Tong University, Shanghai, China
{[1]liuyizhu, [2]jessherlock}@sjtu.edu.cn, [3]kzhu@cs.sjtu.edu.cn

Abstract

Convolutional neural networks (CNNs) have met great success in abstractive summarization, but they cannot effectively generate summaries of desired lengths. Because generated summaries are used in difference scenarios which may have space or length constraints, the ability to control the summary length in abstractive summarization is an important problem. In this paper, we propose an approach to constrain the summary length by extending a convolutional sequence to sequence model. The results show that this approach generates high-quality summaries with user defined length, and outperforms the baselines consistently in terms of ROUGE score, length variations and semantic similarity.

1 Introduction

Great progress (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017; Paulus et al., 2017) has been made recently on abstractive summarization. Many use sequence-to-sequence model based on RNN and attention mechanism (Rush et al., 2015), which was originally used for machine translation (Sutskever et al., 2014; Bahdanau et al., 2014). Recently, Gehring et al. (2017) proposed a convolutional sequence to sequence model equipped with Gated Linear Units (Dauphin et al., 2017), residual connections (He et al., 2016) and attention mechanism. Such a convolutional model achieves state-of-the-art accuracies in abstractive summarization on single sentence summarization, and it is much faster than the previous recurrent models as it can be easily parallelized. Furthermore, unlike recurrent models, the convoluational model has more stable gradients because of its backpropagation path.

Constraining summary length, while largely neglected in the past, is actually an important aspect of abstractive summarization. For example, given the same input document, if the summary is to be displayed on mobile devices, or within a fixed area of advertisement slot on a website, we may want to produce a much shorter summary. Unfortunately, most existing abstractive summarization models are not trained to react to summary length constraints. When the constraint is given at test time, the current practice is **i)** to truncate the generated summary after N tokens are generated when you want the summaries of length no more than N, and **ii)** ignore EOS (end of summary) token until the first M tokens are generated when you want the summaries of length at least M. Such a crude way of controlling summary length makes the output summary incomplete or incoherent.

Previous research on controlling length of abstractive summary has been scarce. Fan et al. (2017), who applies convolutional sequence to sequence model on multi-sentence summarization, converts length range as some special markers which are predefined and fixed. These markers are included in the training vocabulary. At training time, the model prepends the input of the summarizer with marker indicating the length of input sequence. At test time, it controls the length of the generated summary also by prepending length marker indicating the desired length. Unfortunately, this approach can not generate summaries of arbitrary lengths. It only generates summaries in predefined ranges of length, thus only meets the length constraints approximately. This is shown in Table 1. The above truncation practice can be used in conjunction with any of the length control methods but the excessive parts (red) will be truncated leaving incomplete sentences.

In our work, we extend the convolutional sequence to sequence model (Gehring et al., 2017) by controlling the length of summarization. Our approach seeks to generate summaries of any de-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4110–4119
Brussels, Belgium, October 31 - November 4, 2018.

Table 1: Example summaries generated by different models with a desired length of 10 (red parts exceed the 10 token limit).

Reference summary (53 tokens)
david de gea and victor valdes enjoyed an afternoon off at a theme park . spanish duo donned shades as they made the most of the rare sunshine . it has certainly been a rollercoaster season for manchester united . united are third in the premier league after an impressive recent run .
Basic CNN summary (35 tokens)
david de gea and victor valdes made the most of the rare english sun with a trip to a theme park . david de gea and victor valdes enjoyed some fun in the sun .
(Fan et al., 2017) summary (30 tokens)
david de gea and victor valdes enjoyed a trip to a theme park . the pair enjoyed a relaxing time just days after united 's win against manchester city .
Our Length Control summary (LC) (10 tokens)
david de gea and victor valdes enjoy some fun .

sired number of tokens (also shown in Table 1). To do this, a length constraint is added to each convolutional block of the initial layer of the model. This information is propagated layer by layer during training. Our contributions are as follows:

1. We propose a simple but effective method to generate summaries with arbitrary desired length (Section 2.2).
2. Our approach outperforms the state-of-art baseline methods substantially by all evaluation metrics, i.e., ROUGE scores, length variation and semantic similarity (Section 3).
3. The generated summaries from our model are natural and complete, especially when the desired length is short (Section 3).

Next, we present the basic convolutional sequence to sequence model and our extension, followed by the evaluation of our approach and a discussion of related work.

2 Methodology

In this section, we will describe the model architecture used for our experiments and propose our length control method which is implemented by extending the basic model.

For summarization problems based on seq2seq model, given a sequence of tokens $\mathbf{x} = (x_1, x_2, ..., x_m)$ in the source document and a sequence of tokens $\mathbf{y} = (y_1, y_2, ..., y_n)$ in the target summary (i.e. $m > n$), the goal is to estimate the conditional probability $p(\mathbf{y}|\mathbf{x})$:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t}^{T} p(y_t|y_1, y_2, ..., y_{t-1}, \mathbf{x}) \quad (1)$$

We aim at getting the above conditional probabaility which can generate summaries with arbitrary desired length.

2.1 Basic CNN seq2seq Model

Our basic model consists of a multi-layer convolutional sequence to sequence model (CNN seq2seq)[1] (Gehring et al., 2017; LeCun et al., 1989) and an attention mechanism. Figure 1 illustrates the model.

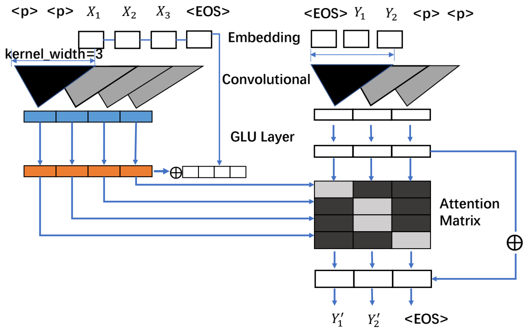

Figure 1: CNN seq2seq model

In the CNN seq2seq model, we obtain the input sequence $\mathbf{X} = (X_1, ..., X_m)$ and output sequence $\mathbf{Y} = (Y_1, ..., Y_n)$ after combining word vectors with their absolute positions in the document. We use $\mathbf{z} = (z_1^l, z_2^l, ..., z_m^l)$ and $\mathbf{h} = (h_1^l, h_2^l, ..., h_n^l)$ to denote the convolutional output of the encoder and decoder in l-th layer. Each element of the output sequence generated by the decoder network is fed back into the next layer of decoder network. Next, we add GLU (Dauphin et al., 2017) and residual connections (He et al., 2016) in each layer:

$$h_i^l = GLU(W^l[h_s^{l-1}, ..., h_t^{l-1}] + b^l) + h_i^{l-1} \quad (2)$$

where $[h_s^{l-1}, ..., h_t^{l-1}]$ corresponds to the h_i^l in the convolutional layers. The choice of s and t is based on kernel width and the padding method used to match the output of convolutional layers to the input length. We compute the probability distribution of generating the next elements y_{i+1} based on the current state and transform the top decoder output h_i^L via softmax:

$$p(y_{i+1}|y_1, ..., y_i, \mathbf{x}) = softmax(W_o h_i^L + b_o) \quad (3)$$

---

[1] `https://github.com/facebookresearch/fairseq-py`.

In addition, a multi-step attention mechanism that connects the encoder and decoder is used in each decoder layer. We define the decoder state d_i^l for attention as following:

$$d_i^l = W_d^l h_i^l + b_d^l + Y_i \tag{4}$$

The attention c_i^l is a weighted sum of the encoder outputs. The weights a_{ij}^l are based on the decoder states.

$$a_{ij}^l = \frac{\exp(d_i^l \cdot z_j^u)}{\sum_{t=1}^{m} \exp(d_i^l \cdot z_t^u)} \tag{5}$$

$$c_i^l = \sum_{j=1}^{m} a_{ij}^l (z_j^u + X_j) \tag{6}$$

At last, we add c_i^l to the current decoder elements h_i^l, which forms the final output or the input of the next layer in the decoder.

2.2 Modified Model with Length Control (LC)

We propose an approach which can control the summary length in CNN seq2seq model. The model can generate different summaries by setting desired length. It has the ability to generate the EOS tag at the appropriate time point in a natural manner.

To produce a summary of a given desired length, we modify the basic model by feeding the desired length as a parameter into the decoder of the CNN seq2seq model. At training time, we use the true length of the gold summary as the desired length. At test time, we can give any desired length len to the model and obtain a summary with length approximate to len. The modified decoder is shown in Figure 2.

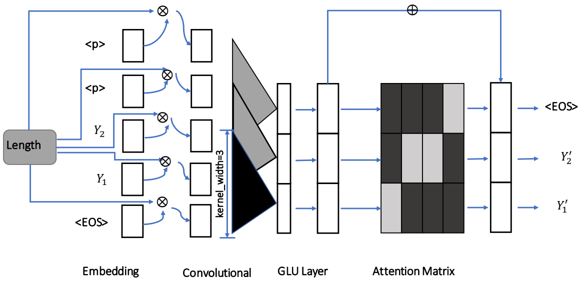

Figure 2: Modified Decoder

The CNN seq2seq model creates hierarchical structure over the input sequence. It is capable of capturing the correlation between elements over short distances at lower layers and between elements over long distances at higher layers. The useful information among the elements is aggregated after GLU. Therefore, we set the desired length as an input to the initial state of the decoder:

$$h_i^1 = v(W^1[h_s^0, ..., h_t^0] + b^1) + h_i^0 * len \tag{7}$$

where W is a trainable parameter, len is the desired length, v is GLU funciton and h_i^0 is the i-th element in the initial layer.

In the above function, we add length information at first layer in CNN model. GLU is like a gate. It can filter some information from a particular unit in each layer. The information attenuation occurs in GLU layer by layer. Different desired lengths have different degrees of information attenuation. Therefore the model is able to learn the probability of generating EOS with its own length information attenuation. This operation enables the model to produce a natural and complete summary for a given length constraint naturally.

3 Evaluation

In this section, our benchmark is the CNN/Daily Mail DMQA dataset (Hermann et al., 2015; Nallapati et al., 2016; See et al., 2017) [2], consisting of pairs of a single source document and a multi-sentence summary. The dataset includes 286,817 training pairs, 13,368 validation pairs and 11,487 test pairs. We follow the same pre-processing step used by See et al. (2017), and fill in the blanks with answer named entities. We show an example of such pairs in Table 4(a).

We compare our length constrained summarization model with the basic CNN seq2seq model and the state-of-the-art length controllable summarization model (Fan et al., 2017) [3]. Following Fan et al., we distribute the dataset into a set of disjoint buckets that correspond to summaries of different lengths. Each bucket contains roughly equal number of documents. The distribution is shown in Figure 3.

All competing methods have three flavors: *free*, *truncated* and *exact*. In the free version(Free), given the desired length N, each method generates summaries naturally until an EOS is generated. In the truncated version(Trunc), each method

---

[2] https://cs.nyu.edu/kcho/DMQA/

[3] All datasets, source code and generated summaries can be downloaded from `http://202.120.38.146/sumlen`.

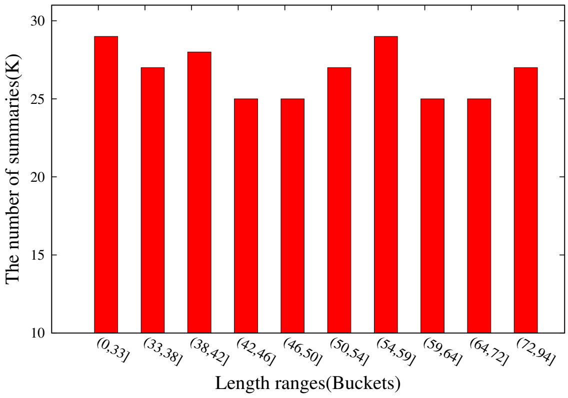

Figure 3: The buckets distribution of the dataset

artificially inserts an EOS if EOS has not been generated in the first N tokens. In the exact version(Exact), each method generates N non-EOS tokens by assigning a score of -∞ to the EOS and inserts an EOS after the N-th token. The purpose of Free version is to evaluate the method's ability to generate summaries with desired length; the purpose of the other two versions is to enable fair comparison of the summaries in terms of their content given that the summaries are of equal length.

3.1 Experimental Setup

In the following experiments, all the competing models have 8 convolutional layers in both encoder and decoder parts with kernel width as 3. For each convolutional layer, we set the hidden vector size as 512 and the embedding size as 256. To alleviate the overfitting problem, we add the dropout ($p = 0.2$) layer for all convolutional layers and fully connected layers.

To optimize the proposed model, we use Nesterov's accelerated gradient method (Sutskever et al., 2013) with gradient clipping 0.1 (Pascanu et al., 2013), momentum 0.99, and learning rate 0.2. We terminate the training process when the learning rate drops below $10e$-5. We set beam size as 5 for the beam search algorithm in the testing step. Next, we introduce the evaluation metrics in the following experiments:

1. *ROUGE* scores (F1 score) of the produced summaries, including ROUGE-1(R-1), ROUGE-2(R-2) and ROUGE-L(R-L) (Lin, 2004). ROUGE-2 is the most popular metric for summarization.

2. *Variance*(Var) of the summary lengths against the desired length len:

$$var = 0.001 * \frac{1}{n}\sum_{i=0}^{n}|l_i - len|^2, \quad (8)$$

where n is the number of pairs in the dataset, and l_i is the length of the generated summary i. We introduce the variance to evaluate the ability of exact control of the output length.

3. *Similarity*(Sim) between generated summaries and their corresponding reference summaries:

$$sim = \frac{1}{n}\sum_{i=0}^{n}\frac{y_i \cdot y_i'}{||y_i||||y_i'||} \quad (9)$$

where n is the number of pairs. y_i is the vector representation of the reference summary i and y_i' is vector of the corresponding generated summary i. Both y_i and y_i' are the sum of GloVe [4] word vectors of the words in these summaries.

We introduce the similarity metric here to complement the ROUGE scores because Yao et al. (2017a) showed that the standard ROUGE scores cannot capture semantic similarity beyond n-grams. Given the same source document, abstractive summarization may create summaries that don't share many words but mean the same. To show the effectiveness of this Sim metric, we design a dataset from the summarization tasks of TAC 2010∼2011[5]. The TAC dataset consists of 90 topics in total, each with 2 subset. Each subset has 4 reference summaries by different humans. We assume reference summaries about the same topic to be semantically similar to each other, while summaries across topics are unrelated. Thus we created 2,160 pairs of similar summaries as postive data and 2,160 pairs of unrelated summaries as negative data. We then compute the Pearson correlation between the ROUGE score and the ground truth as well as between Sim and the ground truth and show the results in Table 2. Sim metric certainly resembles semantic similarity better than ROUGE by this experiment.

In this paper, we don't use manual evaluation as the major metric. The reason is that Lin (2004) showed that the manual evaluation is unstable and

---

[4] https://nlp.stanford.edu/projects/glove/.

[5] https://tac.nist.gov/

Table 2: Pearson correlation with the true semantic relatedness

	TAC 2010	TAC 2011
R-2	0.6107	0.6034
Sim	**0.6653**	**0.7165**

the inter-human agreement is low due to the variety in abstractive summaries. The ROUGE scores and Similarity scores can respectively measure the syntactic similarity and semantic similarity. They are complementary to each other and give better quantitative assessment of the summarization quality.

3.2 Experiment 1: Gold Summary Lengths

In the first experiment, for each test document-summary pair, we set the desired length as the length of the gold summary and ask the competing methods to generate a summary with the desired length. As shown in Table 3, the proposed model (LC) outperforms the other models on all of the evaluation metrics. The ROUGE score shows the accuracy of these models. Lower variance reflects better length control of the model. Higher similarity reflects better quality of generated summaries from the semantic point of view.

Table 3: Desired Length: Gold Summary Lengths

		R-1	R-2	R-L	Var	Sim
Free	CNN	34.49	14.38	25.78	0.3465	0.9220
	Fan	34.53	14.40	25.78	0.3446	0.9216
	LC	**35.45**	**14.50**	**26.02**	**0.0005**	**0.9272**
Trunc	CNN	34.76	**14.53**	26.00	0.3045	0.9201
	Fan	34.74	14.52	25.97	0.3031	0.9197
	LC	**35.44**	14.48	**26.02**	**0.0002**	**0.9268**
Exact	CNN	35.39	14.43	**26.07**	0.0	0.9249
	Fan	35.37	14.42	26.03	0.0	0.9246
	LC	**35.44**	**14.50**	26.02	0.0	**0.9268**

The LC model achieves the highest ROUGE and similarity scores as well as the lowest variance in both Free and Exact version, which shows the effectiveness of LC for generating high quality summaries under length constraint. In the Trunc version, the LC model outperforms the other comparable models on all evaluation metrics except for the ROUGE score. Note that, the ROUGE scores of LC model are very stable, indicating its effective length control. As for the other two models, they have better ROUGE score on Trunc version. However, as the example shown in Table 4[6], higher ROUGE scores do not necessarily mean high quality abstractive summaries.

The ROUGE score consists of Recall(R), Precision(P) and F1-measure(F). The summary tends to achieve a better ROUGE score when the length of generated summary is slightly shorter than the desired length. In Table 4(b), the CNN model has the same R score as LC model and a higher P score than LC model because of its slightly shorter length. We can see that the CNN model achieve a higher F score even its generated summary is not good. Moreover, for the basic model, the generated summary always repeats the sentences when the length of generated summary is longer than the desired length. In Table 4(c), the P score of its Trunc version would be improved by a large margin. Thus, the ROUGE score for the Trunc version biases toward the models with weak length control. The generated summaries of the LC model in Table 4(d), which capture the semantic of the reference summary and satisfy the constraint length very well, are better than the other two models even with a slightly lower ROUGE score. The topic of this example is that Louis Jordan, who is the son of Frank Jordan, got lost during sailing and was finally rescued from his boat. Our model generates the summary with correct information, but other two models get the Louis Jordan and Frank Jordan mixed up. This is correctly measured by the similarity scores.

3.3 Experiment 2: Arbitrary Lengths

In the second experiment, we ask the methods to generate summaries with arbitrary lengths. We report the results of all three methods with five arbitrary lengths: 10, 30, 50, 70 and 90. We show the performance of each model with different length constraints in Table 5, Table 6, Figure 4 and Figure 5. The basic CNN model has the same ROUGE scores in the Free version since it cannot control the length of generated summaries on its own. For Fan et al. (2017), the desired length is mapped to the model's predefined fixed length range(s) that *contains* the desired length before it produces its summaries. For example, the desired length 10 is mapped to the first bucket $(0, 33]$.

To demonstrate the effectiveness of LC model and further illustrate the results, we show an example of generated summaries by LC(Free) model with different lengths. As shown in Table 7, when the desired length (e.g., 10) is very different from the length of the reference summary, the ROUGE

---

[6] The entities in different color indicate two important roles in the text. The words in bold type mean correct content.

Table 4: Example summaries generated in Experiment 1.

(a) Source document and reference summary (36 tokens)

Source document
the last time frank jordan spoke with his son, louis jordan was fishing on **a sailboat a few miles off the south carolina coast**. the next time ... more than two months had passed and the younger jordan was on a contrainer ship 200 miles from north carolina, just **rescued from his disabled boat** . “**i thought i lost you**,”the relieved father said. louis jordan, 37, took his sailboat out in **late january** and **hadn’t been heard from in 66 days** ... the younger jordan said **he took his sailboat out to the gulf stream to find some better fishing** ... the boat capsized two more times before he was **rescued**, according to jordan.
Reference summary
louis jordan says his sailboat capsized three times . he **survived** by collecting rainwater and eating raw fish . frank jordan told cnn **his son** is n’t an experienced sailor but has a strong will .

(b) Free summary(29 tokens), Trunc summary(29 tokens) and Exact summary of CNN

		Summary	R	P	F	Var	Sim
CNN	Free	frank jordan **took his sailboat out to the gulf stream to find some better fishing** , jordan says . “ it took so long , ” jordan says .	6.06	9.09	7.27	0.049	0.9217
	Trunc	frank jordan **took his sailboat out to the gulf stream to find some better fishing** , jordan says . “ it took so long , ” jordan says .	6.06	**9.09**	7.27	0.049	0.9217
	Exact	frank jordan **took his sailboat out to the gulf stream to find some better fishing** , jordan says . jordan says he took his sailboat out to the gulf stream to find some better fishing .	6.06	**6.25**	6.15	-	0.9254

(c) Free summary(50 tokens), Trunc summary(36 tokens) and Exact summary of Fan

		Summary	R	P	F	Var	Sim
Fan	Free	frank jordan **took his sailboat out to the gulf stream to find some better fishing .** jordan says he took his sailboat out to the gulf stream to find some better fishing . jordan says he took his sailboat out to the gulf stream to find some better fishing .	6.06	**4.35**	5.06	0.196	0.9215
	Trunc	frank jordan took his sailboat out to the gulf stream to find some better fishing . **his son** , louis jordan **took his sailboat out to the gulf stream to find some better fishing .**	12.12	**12.90**	12.50	0.0	0.9194
	Exact	frank jordan took his sailboat out to the gulf stream to find some better fishing . **his son** , louis jordan **took his sailboat out to the** gulf stream to find some **better fishing .**	12.12	12.90	12.50	-	0.9194

(d) Free summary(36 tokens), Trunc summary(36 tokens) and Exact summary of LC(ours)

		Summary	R	P	F	Var	Sim
LC	Free	louis jordan was on **a sailboat a few miles off the south carolina coast** . he **had n’t been heard from in 66 days** when he was **rescued** . he was **rescued from his boat** .	6.06	6.06	6.06	0.0	**0.9293**
	Trunc	louis jordan was on **a sailboat a few miles off the south carolina coast** . he **had n’t been heard from in 66 days** when he was **rescued** . he was **rescued from his boat** .	6.06	6.06	6.06	0.0	**0.9293**
	Exact	louis jordan was on **a sailboat a few miles off the south carolina coast** . he **had n’t been heard from in 66 days** when he was **rescued** . he was **rescued from his boat** .	6.06	6.06	6.06	-	**0.9293**

score may not be good even though the generated summary matches the reference quite well semantically. The generated summaries from LC model are *natural* and *complete*. The summaries with short desired length on Trunc and Exact version would be more vulnerable to the incomplete problem. We randomly sample 100 summaries generated by each model under Trunc and Exact with desired length of 10 and 30, and manually inspect their readibility. This is a simplified human-evaluation of summarization, which just determines whether the sentences in summaries under length control are complete or not. If complete, the score is 1; if not, it is 0. It is easier to accomplish and more reliable than other sophisticated human-evaluation. Table 8 shows that the LC model has a clear advantage over the other two models in terms of summary fluency.

In this experiment, the desired length is fixed for all the documents which is independent from the corresponding lengths of reference summaries such that the generated summaries may include more versatile words and phrases different from the reference summaries. Thus, the similar-

Table 5: Desired Length: 10, 30, 50, 70, 90

(a) **Free** version

Free	10			30			50			70			90		
	CNN	Fan	LC	CNN	Fan	LC	CNN	Fan	LC	CNN	Fan	LC	CNN	Fan	LC
R-1	**34.49**	34.28	19.03	**34.49**	34.28	32.26	34.49	34.60	**34.71**	34.49	**34.65**	33.83	**34.49**	30.56	32.17
R-2	**14.38**	14.18	8.45	**14.38**	14.18	13.60	14.38	**14.41**	14.24	14.38	**14.50**	13.67	**14.38**	12.20	13.00
R-L	**25.78**	25.60	16.47	**25.78**	25.60	24.64	25.78	**25.79**	25.62	25.78	**25.82**	24.67	**25.78**	22.08	23.28

(b) **Trunc** version

Trunc	10			30			50			70			90		
	CNN	Fan	LC	CNN	Fan	LC	CNN	Fan	LC	CNN	Fan	LC	CNN	Fan	LC
R-1	**20.14**	20.12	18.77	32.96	**32.99**	32.25	35.14	35.07	**35.60**	34.49	**34.67**	33.83	31.27	**34.70**	32.16
R-2	**9.27**	9.22	8.31	**14.06**	14.04	13.60	**14.46**	14.40	14.30	14.38	**14.50**	13.67	12.40	**14.55**	13.00
R-L	**17.35**	17.34	16.28	**25.11**	25.06	24.62	**26.09**	26.05	25.90	25.78	**25.82**	24.67	22.69	**25.86**	23.29

(c) **Exact** version

Exact	10			30			50			70			90		
	CNN	Fan	LC	CNN	Fan	LC	CNN	Fan	LC	CNN	Fan	LC	CNN	Fan	LC
R-1	**20.14**	20.14	20.06	**33.05**	32.83	32.94	34.71	34.72	**34.81**	32.24	33.35	**33.82**	31.27	31.37	**32.04**
R-2	**9.27**	9.23	9.23	**14.08**	13.79	14.00	**14.78**	14.17	14.23	13.33	13.39	**13.59**	12.41	12.47	**12.86**
R-L	**17.36**	17.36	17.30	**25.15**	24.87	25.02	**25.65**	25.63	25.60	24.31	24.35	**24.56**	22.69	22.76	**23.14**

Table 6: Similarity of different length

(a) **Free** version

	CNN	Fan	LC
10	**0.9220**	0.9205	0.8124
30	**0.9220**	0.9214	0.9092
50	0.9220	0.9216	**0.9263**
70	0.9220	0.9222	**0.9323**
90	0.9220	0.9234	**0.9256**

(b) **Trunc** version

	CNN	Fan	LC
10	0.7966	0.7968	**0.8003**
30	0.9079	0.9080	**0.9085**
50	0.9236	0.9231	**0.9286**
70	0.9219	0.9222	**0.9323**
90	0.9325	0.9329	**0.9353**

(c) **Exact** version

	CNN	Fan	LC
10	0.7968	0.7961	**0.7975**
30	0.9083	0.9073	**0.9090**
50	0.9248	0.9245	**0.9251**
70	0.9299	0.9230	**0.9320**
90	0.9325	0.9327	**0.9347**

Table 7: Generated summaries of LC (Free) model

10	the younger jordan was **rescued from his disabled boat** .
30	louis jordan was **rescued from his disabled** . he **boat had n't been heard from in 66 days in late january** . he was rescued from his disabled boat .
50	" **i thought i lost you** , " jordan says . the younger jordan was on **a sailboat a few miles off the south carolina coast** . " i thought i lost you , " jordan tells **his son** . jordan says he was grateful to the people .

Table 8: The proportion of summaries that are natural and complete with desired length 10 and 30

	Trunc		Exact	
	10	30	10	30
CNN	0.41	0.37	0.48	0.47
Fan	0.50	0.42	0.53	0.57
LC	**0.62**	**0.59**	**0.88**	**0.86**

ity score is more reasonable for evaluation than ROUGE score. As shown in Table 6 and Figure 4, the LC model achieves the highest similarity score except for the length of 10 and 30 in the Free version. The reason is that there is only 5% of testing data with the length of reference summary shorter than 30. Due to the effective length control of LC model, the lengths of generated summaries from LC model are usually much shorter than those from the other models and the length of corresponding reference summaries when we set the desired length as 10 or 30. This leads to a relative lower similarity score shown in Figure 4(b) and Figure 4(c). As shown in Figure 5, the LC model achieves the lowest variance. In Figure 5(a), as the length of most summaries is around 50 and the number of summaries with a length of 10 or 90 is small, the CNN model and Fan model has lowest variance at 50 and highest variance at 90. In Figure 5(b), the length of generated summaries in Trunc version is no more than desired length. So the variances of CNN model and Fan model are incremental. Besides, we can find that the LC model is stable under all conditions because of its effective length control model.

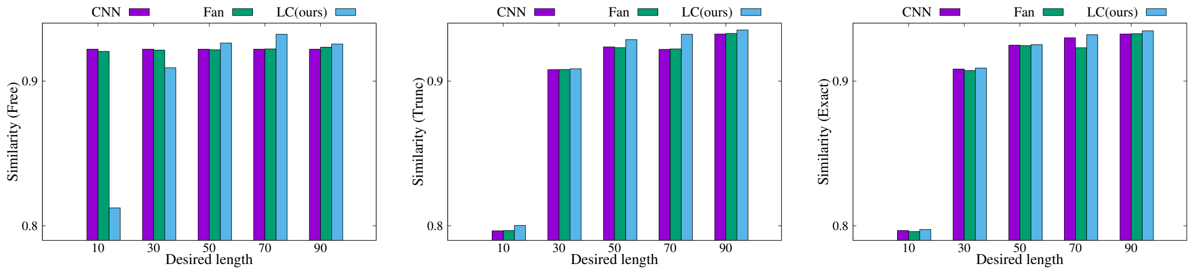

(a) Similarity in Free version (b) Similarity in Trunc version (c) Similarity in Exact version

Figure 4: Similarity of different length

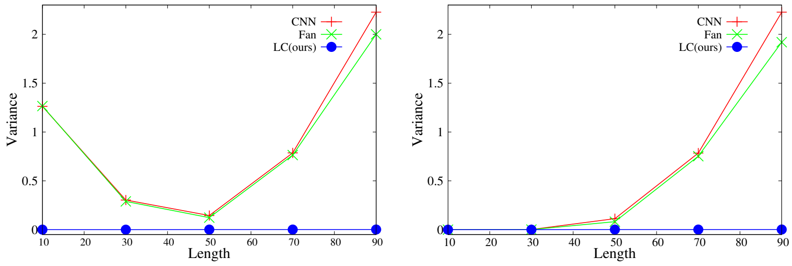

(a) Variance in Free version (b) Variance in Trunc version

Figure 5: Variance of different length

3.4 Significance Test on Similarity Result

We use significance test to prove that similarity metric is reliable even though the numerical difference of similarity scores in experiment is little. Because the similarity scores of generated summaries do not follow normal distribution, we take Kruskal-Wallis test (Loukina et al., 2014; Albert, 2017) as our significance test to measure that the difference of similarity results of three methods is significant or not. As shown in Table 9, all p-values are less than 0.05. The smaller p-value, the higher significant. Thus, the difference of the similarity results is significant.

Table 9: p-value of significance test

	Free	Trunc	Exact
Exp.1	3.4e-32	2.12e-45	0.01
Exp.2	0.0	4.6e-39	1.0e-4

4 Related Work

In this section, we discuss some previous work on length control in abstractive summarization and explain why we choose CNN as our basic summarization model.

4.1 Length Control for Abstractive Summarization

When summarizing a document, it is desirable to be able to control the length of summary so as to cater to different users and scenarios. Most abstractive summarization systems are based on encoder-decoder models and generate summaries whose length depends on the training summaries. Due to the variability of the sequence generation models, such as the different structures and functions, it is hard to design a length constraint method on all summarization models.

Previous methods control summary length by generating EOS token at a particular time. Rush et al. (2015) used an ad-hoc method, in which the system is inhibited from generating the EOS tag by assigning a score of $-\infty$ to the tag and generats a fixed number of words. Kikuchi et al. (2016) proposed two different methods for RNN seq2seq model which can control the summary length by taking length embedding as an additional input for the LSTM and adding desired length into initial memory cell for the LSTM. In this model, they use the Gigawords as dataset and focus on the abstractive summarization in sentence level which generates one sentence as the summary. For CNN seq2seq model, Fan et al. (2017) put some special markers into the vocabulary which denote different length ranges. It prepends the input of the summarizer with the marker during training and testing. These special markers are predefined and fixed. In this paper, we aim at generating complete summaries with arbitrary desired length naturally for CNN seq2seq model. We use multi-layers CNN seq2seq model on both encoder and decoder. We set the length constraint at the first layer of decoder to implement the length control of the summarization. Compared with other methods, our approach can effectively control the length of generated summary in a natural manner. Meanwhile, it can generate summaries with length approximate to the desired length without semantic losing in less time.

4.2 Encoder-Decoder for Abstractive Summarization

Automatic document summarization generates short summaries for original documents. A summary should cover the key topics of the original document(s). A good summary should be coherent, non-redundant and readable(Yao et al., 2017b). The research in abstractive summarization with encoder-decoder model (Sutskever et al., 2014; Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017; Paulus et al., 2017; Fan et al., 2017) has made some progress.

Most of them use RNN with different attention mechanisms (Nallapati et al., 2016; See et al., 2017; Paulus et al., 2017). Rush et al. (2015) used RNN with soft-attention, while Paulus et al. (2017) used the RNN with intra-attention. Recently, research on CNN based summarization has gained momentum. Gehring et al. (2017) proposed the CNN seq2seq model with multi-step attention, which was extended in (Fan et al., 2017). Bai et al. (2018) showed that CNN is more powerful than RNN for sequence modeling. What's more, CNN enables much faster training and more stable gradients than RNN. Therefore we select CNN seq2seq model as our basic model and do not compare our model with RNN seq2seq model.

5 Conclusion

We presented a simple approach to modify existing CNN seq2seq model with a summary length input and were able to train a model that produces summaries of desired length that are fluent and coherent. This is a better solution than the current practice of summary truncation. Compared with the existing summarization methods, we show that our model has the ability to control the output length on its own using its internal state without losing semantic information or sacrificing the ROUGE score.

Acknowledgment

Kenny Q. Zhu is the contact author and was supported by NSFC grants 91646205 and 61373031. Thanks to the anonymous reviewers for their valuable feedback.

References

Corbin Albert. 2017. Exploring teacher forcing techniques for sequence-to-sequence abstractive headline summarization.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 93–98.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 933–941.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *arXiv preprint arXiv:1711.05217*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1243–1252.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1328–1338.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 396–404.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Anastassia Loukina, Klaus Zechner, and Lei Chen. 2014. Automatic evaluation of spoken summaries: the case of language assessment. In *Proceedings of the ninth workshop on innovative use of NLP for building educational applications*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 379–389.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1139–1147.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Jin-Ge Yao, Xiaojun Wan, and Jianguo Xiao. 2017a. Recent advances in document summarization. *Knowl. Inf. Syst.*

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2017b. Recent advances in document summarization. *Knowl. Inf. Syst.*, 53(2):297–336.

APRIL: Interactively Learning to Summarise by Combining Active Preference Learning and Reinforcement Learning

Yang Gao, Christian M. Meyer, Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
`https://www.ukp.tu-darmstadt.de/`

Abstract

We propose a method to perform automatic document summarisation without using reference summaries. Instead, our method interactively learns from users' *preferences*. The merit of preference-based interactive summarisation is that preferences are easier for users to provide than reference summaries. Existing preference-based interactive learning methods suffer from high sample complexity, i.e. they need to interact with the oracle for many rounds in order to converge. In this work, we propose a new objective function, which enables us to leverage active learning, preference learning and reinforcement learning techniques in order to reduce the sample complexity. Both simulation and real-user experiments suggest that our method significantly advances the state of the art. Our source code is freely available at `https://github.com/UKPLab/emnlp2018-april`.

1 Introduction

With the rapid growth of text-based information on the Internet, *automatic document summarisation* attracts increasing research attention from the Natural Language Processing (NLP) community (Nenkova and McKeown, 2012). Most existing document summarisation techniques require access to reference summaries to train their systems. However, obtaining reference summaries is very expensive: Lin (2004) reported that 3,000 hours of human effort were required for a simple evaluation of the summaries for the Document Understanding Conferences (DUC). Although previous work has proposed heuristics-based methods to summarise without reference summaries (Ryang and Abekawa, 2012; Rioux et al., 2014), the gap between their performance and the upper bound is still large: the ROUGE-2 upper bound of .212 on DUC'04 (P.V.S. and Meyer, 2017) is, for example, twice as high as Rioux et al.'s (2014) .114.

The *Structured Prediction from Partial Information* (SPPI) framework has been proposed to learn to make structured predictions without access to gold standard data (Sokolov et al., 2016b). SPPI is an interactive NLP paradigm: It interacts with a user for multiple rounds and learns from the user's feedback. SPPI can learn from two forms of feedback: *point-based* feedback, i.e. a numeric score for the presented prediction, or *preference-based* feedback, i.e. a preference over a pair of predictions. Providing preference-based feedback yields a lower cognitive burden for humans than providing ratings or categorical labels (Thurstone, 1927; Kendall, 1948; Kingsley and Brown, 2010; Zopf, 2018). Preference-based SPPI has been applied to multiple NLP applications, including text classification, chunking and machine translation (Sokolov et al., 2016a; Kreutzer et al., 2017). However, SPPI has prohibitively high sample complexities in the aforementioned NLP tasks, as it needs at least hundreds of thousands rounds of interaction to make near-optimal predictions, even with simulated "perfect" users. Figure 1a illustrates the workflow of the preference-based SPPI.

To reduce the sample complexity, in this work, we propose a novel preference-based interactive learning framework, called *APRIL* (Active Preference ReInforcement Learning). APRIL goes beyond SPPI by proposing a new objective function, which divides the preference-based interactive learning problem into two phases (illustrated in Figure 1b): an *Active Preference Learning* (APL) phase (the right cycle in Figure 1b), and a *Reinforcement Learning* (RL) phase (the left cycle). We show that this separation enables us to query preferences more effectively and to use the collected preferences more efficiently, so as to reduce the sample complexity.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4120–4130
Brussels, Belgium, October 31 - November 4, 2018.

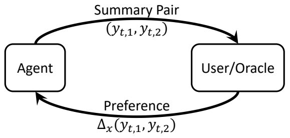

(a) Workflow of preference-based SPPI

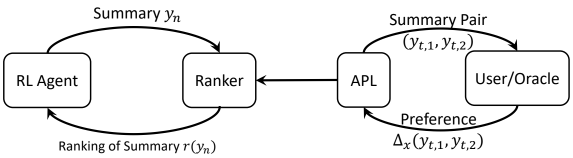

(b) Workflow of APRIL

Figure 1: A comparison of workflows of SPPI (a) and APRIL (b) in the EMDS use case. Notation details, e.g., Δ_x and $r(y_n)$, are discussed in §3.

We apply APRIL to *Extractive Multi-Document Summarisation* (EMDS). The task of EMDS is to extract sentences from the original documents to build a summary under a length constraint. We accommodate multiple APL and RL techniques in APRIL and compare their performance under different simulation settings. We also compare APRIL to a state-of-the-art SPPI implementation using both automatic metrics and human evaluation. Our results suggest that APRIL significantly outperforms SPPI.

2 Related Work

RL has been previously used to perform EMDS without using reference summaries. Ryang and Abekawa (2012) formulated EMDS as a *Markov Decision Process* (MDP), designed a heuristics-based reward function considering both information coverage rate and redundancy level, and used the *Temporal Difference* (TD) algorithm (Sutton, 1984) to solve the MDP. In a follow-up work, Rioux et al. (2014) proposed a different reward function, which also did not require reference summaries; their experiments suggested that using their new reward function improved the summary quality. Henß et al. (2015) proposed a different RL formulation of EMDS and jointly used supervised learning and RL to perform the task. However, their method requires the access to reference summaries. More recent works applied encoder-decoder-based RL to document summarisation (Ranzato et al., 2015; Narayan et al., 2018; Paulus et al., 2017; Pasunuru and Bansal, 2018). These works outperformed standard encoder-decoder as RL can directly optimise the ROUGE scores and can tackle the *exposure bias* problems. However, these neural RL methods all used ROUGE scores as their rewards, which in turn relied on reference summaries. APRIL can accommodate these neural RL techniques in its RL phase by using a ranking of summaries instead of the ROUGE scores as rewards. We leave neural APRIL for future study.

P.V.S. and Meyer (2017) proposed a bigram-based interactive EMDS framework. They asked users to label important bigrams in candidate summaries and used *integer linear programming* (ILP) to extract sentences covering as many important bigrams as possible. Their method requires no access to reference summaries, but it requires considerable human effort during the interaction: in simulation experiments, their system needed to collect up to 350 bigram annotations from a (simulated) user. In addition, they did not consider noise in users' annotations but simulated perfect oracles.

Preference learning aims at obtaining the ranking (i.e. total ordering) of objects from pairwise preferences (Fürnkranz and Hüllermeier, 2010). Simpson and Gurevych (2018) proposed to use an improved *Gaussian process preference learning* (Chu and Ghahramani, 2005) for learning to rank arguments in terms of convincingness from crowd-sourced annotations. However, such Bayesian methods can hardly scale and suffer from high computation time. Zopf (2018) recently proposed to learn a sentence ranker from preferences. The resulting ranker can be used to identify the important sentences and thus to evaluate the quality of the summaries. His study also suggests that providing sentence preferences takes less time than writing reference summaries. APRIL not only learns a ranking over summaries from pairwise preferences, but also uses the ranking to "guide" our RL agent to generate good summaries.

There is a recent trend in machine learning to combine active learning, preference learning and RL, for learning to perform complex tasks from preferences (Wirth et al., 2017). The resulting algorithm is termed *Preference-based RL* (PbRL), and has been used in multiple applications, including training robots (Wirth et al., 2016) and Atari-playing agents (Christiano et al., 2017). SPPI and APRIL can both be viewed as PbRL algorithms. But unlike most PbRL methods that learn a utility function of the predictions (in EMDS, predictions are summaries) to guide the RL agent, APRIL is able to directly use a ranking of predictions to guide the RL agent without making assumptions about the underlying structure of the utility functions. This also enables APRIL to use non-utility-based preference learning techniques (e.g., Maystre and Grossglauser, 2017).

3 Background

In this section, we recap necessary details of SPPI, RL and preference learning, and adapt them to the EMDS use case, laying the foundation for APRIL.

3.1 The SPPI Framework

Let $\mathcal{X}$ be the input space and let $\mathcal{Y}(x)$ be the set of possible outputs for input $x \in \mathcal{X}$. In EMDS, $x \in \mathcal{X}$ is a cluster of documents and $\mathcal{Y}(x)$ is the set of all possible summaries for cluster x. The function $\Delta_x\colon \mathcal{Y}(x) \times \mathcal{Y}(x) \to \{0, 1\}$ is the *preference function* such that $\Delta_x(y_i, y_j) = 1$ if the user believes y_j is better than y_i (denoted by $y_j \succ y_i$ or equivalently $y_i \prec y_j$), and 0 otherwise. Throughout this paper we assume that users do not equally prefer two different items. For a given x, the expected loss is:

$$\mathcal{L}^{\text{SPPI}}(w|x) = \mathbb{E}_{p_w(y_i,y_j|x)}[\Delta_x(y_i, y_j)] \\ = \sum_{y_i,y_j \in \mathcal{Y}(x)} \Delta_x(y_i, y_j)\, p_w(y_i, y_j|x), \qquad (1)$$

where $p_w(y_i, y_j|x)$ is the probability of querying the pair (y_i, y_j). Formally,

$$p_w(y_i, y_j|x) \\ = \frac{\exp[w^\intercal(\phi(y_i|x) - \phi(y_j|x))]}{\sum\limits_{y_p,y_q \in \mathcal{Y}(x)} \exp[w^\intercal(\phi(y_p|x) - \phi(y_q|x))]}, \qquad (2)$$

where $\phi(y|x)$ is the vector representation of y given x, and w is the weight vector to be learnt. Eq. (2) is a Gibbs sampling strategy: $w^\intercal(\phi(y_i|x) - \phi(y_j|x))$ can be viewed as the "utility gap" between y_i and y_j. The sampling strategy p_w encourages querying pairs with large utility gaps.

To minimise $\mathcal{L}^{\text{SPPI}}$, SPPI uses gradient descent to update w incrementally. Alg. 1 presents the pseudo code of our adaptation of SPPI to EMDS. In the supplementary material, we provide a detailed derivation of $\nabla_w \mathcal{L}^{\text{SPPI}}(w|x)$.

3.2 Reinforcement Learning

RL amounts to efficient algorithms for searching optimal solutions in MDPs. MDPs are widely used to formulate *sequential decision making* problems, which EMDS falls into: in EMDS, the summariser has to sequentially select sentences from the original documents and add them to the draft summary. An (episodic) MDP is a tuple (S, A, P, R, T). S is the set of *states*, A is the set of *actions*, $P\colon S \times A \times S \to \mathbb{R}$ is the *transition function* with $P(s'|s, a)$ yielding the probability of performing action a in state s and being transited to a new state s'. $R\colon S \times A \to \mathbb{R}$ is the *reward function* with $R(s, a)$ giving the immediate reward for performing action a in state s. $T \subseteq S$ is the set of *terminal states*; visiting a terminal state terminates the current episode.

```
Input  : sequence of learning rates $\gamma_t$; query
         budget $T$; document cluster $x$
initialise $w_0$;
while $t = 0 \ldots T$ do
    sample $(y_i, y_j)$ according to Eq. (2);
    obtain feedback $\Delta_x(y_i, y_j)$;
    $w_{t+1} := w_t - \gamma_t \nabla_w \mathcal{L}^{\text{SPPI}}(w|x)$
end
Output: $y^* = \arg\max_{y \in Y(x)} w_{T+1}^\intercal \phi(y, x)$
```

Algorithm 1: SPPI for preference-based interactive document summarisation (adjusted from Alg. 2 in (Sokolov et al., 2016a)).

In EMDS, we follow the same MDP formulation as Ryang and Abekawa (2012) and Rioux et al. (2014). Given a document cluster, a state s is a draft summary, A includes two types of actions, *concatenate* a new sentence to the current draft summary, or *terminate* the draft summary construction. The transition function P in EMDS is trivial because given the current draft summary and an action, the next state can be easily inferred. The reward function R returns an evaluation score of the summary once the action *terminate* is performed; otherwise it returns 0 because the summary is still under construction and thus not ready to be evaluated. Providing non-zero rewards before the action *terminate* can lead to even worse result, as reported by Rioux et al. (2014).

A *policy* $\pi\colon S \times A \to \mathbb{R}$ in an MDP defines how actions are selected: $\pi(s, a)$ is the probability of selecting action a in state s. In EMDS, a policy corresponds to a strategy to build summaries for a given document cluster. We let $\mathcal{Y}_\pi(x)$ be the set of all possible summaries the policy π can construct in the document cluster x, and we slightly abuse the notation by letting $\pi(y|x)$ denote the probability of policy π generating a summary y in cluster x. Then the expected reward of a policy is:

$$\begin{aligned}\mathcal{R}^{\mathrm{RL}}(\pi|x) &= \mathbb{E}_{y\in\mathcal{Y}_\pi(x)} R(y|x)\\ &= \sum_{y\in\mathcal{Y}_\pi(x)} \pi(y|x)R(y|x), \qquad (3)\end{aligned}$$

where $R(y|x)$ is the reward for summary y in document cluster x. The goal of an MDP is to find the optimal policy π^* that has the highest expected reward: $\pi^* = \arg\max_\pi \mathcal{R}^{\mathrm{RL}}(\pi)$.

Note that the loss function in SPPI (Eq. (1)) and the expected reward function in RL (Eq. (3)) are in similar forms: if we view the pair selection probability p_w in Eq. (2) as a policy, and view the preference function Δ_x in Eq. (1) as a negative reward function, we can view SPPI as an RL problem. The major difference between SPPI and RL is that SPPI selects and evaluates pairs of outputs, while RL selects and evaluates single outputs. We will exploit their connection to propose our new objective function and the APRIL framework.

3.3 Preference Learning

The linear Bradley-Terry (BT) model (Bradley and Terry, 1952) is one of the most widely used methods in preference learning. Given a set of items $\mathcal{Y}$, suppose we have observed T preferences: $Q = \{q_1(y_{1,1}, y_{1,2}), \cdots, q_T(y_{T,1}, y_{T,2})\}$, where $y_{i,1}, y_{i,2} \in \mathcal{Y}$, and $q_i \in \{\prec, \succ\}$ is the oracle's preference in the i^{th} round. The BT model minimises the following cross-entropy loss:

$$\begin{aligned}\mathcal{L}^{\mathrm{BT}}(w) = -\sum_{q_i(y_{i,1},y_{i,2})\in Q} [\,&\mu_{i,1}\log\mathcal{P}_w(y_{i,1}\succ y_{i,2})\\ &+ \mu_{i,2}\log\mathcal{P}_w(y_{i,2}\succ y_{i,1})\,], \qquad (4)\end{aligned}$$

where $\mathcal{P}_w(y_i \succ y_j) = (1 + \exp[w^\intercal(\phi(y_j) - \phi(y_i))])^{-1}$, and $\mu_{i,1}$ and $\mu_{i,2}$ indicate the direction of preferences: if $y_{i,1} \succ y_{i,2}$ then $\mu_{i,1} = 1$ and $\mu_{i,2} = 0$. Let $w^* = \arg\min_w \mathcal{L}^{\mathrm{BT}}(w)$, then w^* can be used to rank all items in $\mathcal{Y}$: for any $y_i, y_j \in \mathcal{Y}$, the ranker prefers y_i over y_j if $w^{*\intercal}\phi(y_i) > w^{*\intercal}\phi(y_j)$.

4 APRIL: Decomposing SPPI into Active Preference Learning and RL

A major problem of SPPI is its high sample complexity. We believe this is due to two reasons. First, SPPI's sampling strategy is inefficient: From Eq. (2) we can see that SPPI tends to select pairs with large quality gaps for querying the user. This strategy can quickly identify the relatively good and relatively bad summaries, but needs many rounds of interaction to find the top summaries. Second, SPPI uses the collected preferences ineffectively: In Alg. 1, each preference is used only once for performing the gradient descent update and is forgotten afterwards. SPPI does not generalise or re-use collected preferences, wasting the useful and expensive information.

These two weaknesses of SPPI motivate us to propose a new learning paradigm that can query and generalise preferences more efficiently. Recall that in EMDS, the goal is to find the optimal summary for a given document cluster x, namely the summary that is preferred over all other possible summaries in $\mathcal{Y}(x)$. Based on this understanding, we define a new expected reward function $\mathcal{R}^{\mathrm{APRIL}}$ for policy π as follows:

$$\begin{aligned}&\mathcal{R}^{\mathrm{APRIL}}(\pi|x) = \mathbb{E}_{y_j\sim\pi}[\frac{1}{|\mathcal{Y}(x)|}\sum_{y_i\in\mathcal{Y}(x)}\Delta_x(y_i, y_j)]\\ &= \frac{1}{|\mathcal{Y}(x)|}\sum_{y_j\in\mathcal{Y}_\pi(x)}\pi(y_j|x)\sum_{y_i\in\mathcal{Y}(x)}\Delta_x(y_i, y_j)\\ &= \sum_{y\in\mathcal{Y}_\pi(x)}\pi(y|x)\, r(y|x), \qquad (5)\end{aligned}$$

where $r(y|x) = \sum_{y_i\in\mathcal{Y}(x)}\Delta_x(y_i, y_j)/|\mathcal{Y}(x)|$. Note that $\Delta_x(y_i, y_j)$ equals 1 if y_j is preferred over y_i and equals 0 otherwise (see §3.1). Thus, $r(y|x)$ is the relative position of y in the (ascending) sorted $\mathcal{Y}(x)$, and it can be approximated by preference learning. The use of preference learning enables us to generalise the observed preferences to a ranker (see §3.3), allowing more effective use of the collected preferences. Also, we can use active learning to select summary pairs for querying more effectively. In addition, the resemblance of $\mathcal{R}^{\mathrm{APRIL}}$ and RL's reward function $\mathcal{R}^{\mathrm{RL}}$ (in Eq. (3)) enables us to use a wide range of RL algorithms to maximise $\mathcal{R}^{\mathrm{APRIL}}$ (see §2).

Based on the new objective function, we split the preference-based interactive learning into two phases: an *Active Preference Learning* (APL) phase (the right cycle in Fig. 1b), responsible for querying preferences from the oracle and approximating the ranking of summaries, and an *RL* phase (the left cycle in Fig. 1b), responsible for learning to summarise based on the learned ranking. The resulting framework APRIL allows for integrating any active preference learning and RL techniques. Note that only the APL phase is online (i.e. in-

```
Input  : query budget $T$; document cluster $x$;
         RL episode budget $N$
/* Phase 1: active preference learning */
while $t = 0 \dots T$ do
    sample a summary pair $(y_i, y_j)$ using any APL strategy;
    obtain feedback $\Delta_x(y_i, y_j)$;
    update ranker according to Eq. (4) ;
end
/* Phase 2: RL-based summarisation */
initialise an arbitrary policy $\pi_0$;
while $n = 0 \dots N$ do
    evaluate policy $\pi_n$ according to Eq. (5);
    update policy $\pi_n$ using any RL algorithm;
end
Output: $y^* = \arg\max_{y \in \mathcal{Y}_{\pi_N}(x)} \pi_N(y|x)$
```

Algorithm 2: Pseudo code of APRIL

Dataset	Lang	# Topic	# Doc	# Token/Doc
DUC '01	EN	30	308	781
DUC '02	EN	59	567	561
DUC '04	EN	50	500	587

Table 1: Statistics of the datasets. The target summary length is 100 tokens in all three datasets.

volving humans in the loop) while the RL phase can be performed offline, helping to improve the real-time responsiveness. Also, the learned ranker can provide an unlimited number of rewards (i.e. $r(y|x)$ in Eq. (5)) to the RL agent, enabling us to perform many episodes of RL training with a small number of collected preferences – unlike in SPPI where each collected preference is used to train the system for one round and is forgotten afterwards. Alg. 2 shows APRIL in pseudo code.

5 Experimental Setup

Datasets. We perform experiments on DUC '04 to find the best performing APL and RL techniques. Then we combine the best-performing APL and RL to complete APRIL and compare it against SPPI on the DUC '01, DUC '02 and DUC '04 datasets.[1] Some statistics of these datasets are summarised in Table 1.

Simulated Users. Existing preference-based interactive learning techniques assume that the oracle has an *intrinsic evaluation function* U^* and provides preferences consistent with U^* by preferring higher valued candidates. We term this a *Perfect Oracle* (PO). We believe that assuming a PO is unrealistic for real-world applications, because sometimes real users tend to misjudge the preference direction, especially when the presented candidates have similar quality. In this work, besides PO, we additionally consider two types of noisy oracles based on the user-response models proposed by Viappiani and Boutilier (2010):

- **Constant noisy oracle (CNO)**: with probability $c \in [0, 1]$, this oracle randomly selects which summary is preferred; otherwise it provides preferences consistent with U^*. We consider CNOs with $c = 0.1$ and $c = 0.3$.
- **Logistic noisy oracle (LNO)**: for two summaries y_i and y_j in cluster x, the oracle prefers y_i over y_j with probability $p_{U^*}(y_i \succ y_j|x; m) = (1 + \exp[(U^*(y_j|x) - U^*(y_i|x))/m])^{-1}$. This oracle reflects the intuition that users are more likely to misjudge the preference direction when two summaries have similar quality. Note that the parameter $m \in \mathbb{R}^+$ controls the "noisiness" of the user's responses: higher values of m result in a less steep sigmoid curve, and the resulting oracle is more likely to misjudge. We use LNOs with $m = 0.3$ and $m = 1$.

As for the intrinsic evaluation function U^*, recent work has suggested that human preferences over summaries have high correlations to ROUGE scores (Zopf, 2018). Therefore, we define:

$$U^*(y|x) = \frac{R_1(y|x)}{0.47} + \frac{R_2(y|x)}{0.22} + \frac{R_S(y|x)}{0.18} \quad (6)$$

where R_1, R_2 and R_S stand for ROUGE-1, ROUGE-2 and ROUGE-SU4, respectively. The real values (0.47, 0.22 and 0.18) are used to balance the weights of the three ROUGE scores. We choose them to be around the EMDS upper-bound ROUGE scores reported by P.V.S. and Meyer (2017). As such, an optimal summary's U^* value should be around 3.

Implementation. All code is written in Python and runs on a desktop PC with 8 GB RAM and an i7-2600 CPU. We use NLTK (Bird et al., 2009) to perform sentence tokenisation. Our source code is freely available at `https://github.com/UKPLab/emnlp2018-april`.

6 Simulation Results

We first study the APL phase (§6.1) and the RL phase (§6.2)) separately by comparing the perfor-

---

[1] `http://duc.nist.gov/`

mance of multiple APL and RL algorithms in each phase. Then, in §6.3, we combine the best performing APL and RL algorithm to complete Alg. 2 and compare APRIL against SPPI.

6.1 APL Phase Performance

Recall that the task of APL is to output a ranking of all summaries in a cluster. In this subsection, we test multiple APL techniques and compare the quality of their resulting rankings. Two metrics are used: Kendall's τ (Kendall, 1948) and Spearman's ρ (Spearman, 1904). Both metrics are valued between -1 and 1, with higher values suggesting higher rank correlation. Because the number of possible summaries in a cluster is huge, instead of evaluating the ranking quality on all possible summaries, we evaluate rankings on 10,000 randomly sampled summaries, denoted $\hat{\mathcal{Y}}(x)$. During querying, all candidate summaries presented to the oracle are also selected from $\hat{\mathcal{Y}}(x)$. Sampling $\hat{\mathcal{Y}}(x)$ a priori helps us to reduce the response time to under 500 ms for all APL techniques we test. We compare four active learning strategies under two query budgets, $T = 10$ and $T = 100$:

- **Random Sampling (RND)**: Randomly select two summaries from $\hat{\mathcal{Y}}(x)$ to query.
- **SPPI Sampling (SBT)**: Select summary pairs from $\hat{\mathcal{Y}}(x)$ according to the SPPI strategy in Eq. (2). After each round, the weight vector w is updated according to Eq. (4).
- **Uncertainty Sampling (Unc)**: Query the most *uncertain* summary pairs. In line with P.V.S. and Meyer (2017), the uncertainty of a summary is evaluated as follows: first, we estimate the probability of a summary y being the optimal summary in cluster x as $p_{\text{opt}}(y|x) = (1 + \exp(-w_t^{*\intercal}\phi(x, y)))^{-1}$, where w_t^* is the weights learned by the BT model (see §3.3) in round t. Given $p_{\text{opt}}(y|x)$, we let the uncertainty score $unc(y|x) = 1 - p_{\text{opt}}(y|x)$ if $p_{\text{opt}}(y|x) \geq 0.5$ and $unc(y|x) = p_{\text{opt}}(y|x)$ otherwise.
- **J&N** is the *robust query selection algorithm* proposed by Jamieson and Nowak (2011). It assumes that the items' preferences are dependent on their distances to an unknown reference point in the embedding space: the farther an item to the reference point, the more preferred the item is. After each round of interaction, the algorithm uses all collected preferences to locate the area where the reference point may fall into, and identify the query pairs which can reduce the size of this area, termed *ambiguous query pairs*. To combat noise in preferences, the algorithm selects the most-likely-correct ambiguous pair to query the oracle in each round.

After all preferences are collected, we obtain the ranker as follows: for any $y_i, y_j \in \mathcal{Y}(x)$, the ranker prefers y_i over y_j if

$$\begin{aligned}&\alpha w^{*\intercal}\phi(y_i|x) + (1-\alpha)HU(y_i|x) > \\ &\alpha w^{*\intercal}\phi(y_j|x) + (1-\alpha)HU(y_j|x), \qquad (7)\end{aligned}$$

where w^* is the weights vector learned by the BT model (see Eq. (4)), HU is the heuristics-based summary evaluation function proposed by Ryang and Abekawa (2012), and $\alpha \in [0, 1]$ is a parameter. The aim of using HU and α is to trade off between the *prior knowledge* (i.e. heuristics-based HU) and the *posterior observation* (i.e. the BT-learnt w^*), so as to combat the *cold-start* problem. Based on some preliminary experiments, we set $\alpha = 0.3$ when the query budget is 10, and $\alpha = 0.7$ when the query budget is 100. The intuition is to put more weight to the posterior with increasing rounds of interaction. More systematic research of α can yield better results; we leave it for future work. For the vector $\phi(y|x)$, we use the same bag-of-bigram embeddings as Rioux et al. (2014), and we let its length be 200.

In Table 2, we compare the performance of the four APL methods on the DUC'04 dataset. The baseline we compared against is the prior ranking. We find that Unc significantly[2] outperforms all other APL methods, except when the oracle is LNO-1, where the advantage of Unc to SBT is not significant. Also, both Unc and SBT are able to significantly outperform the baseline under all settings. The competitive performance of SBT, especially with LNO-1, is due to its unique sampling strategy: LNO-1 is more likely to misjudge the preference direction when the presented summaries have similar quality, but SBT has high probability to present summaries with large quality gaps (see Eq. (2)), effectively reducing the chance that LNOs misjudge preference directions. However, SBT is more "conservative" compared to Unc because it tends to exploit the existing

---

[2] In this paper we use double-tailed student t-test to compute p-values, and we let significance level be $p < 0.01$.

	RND		SBT		Unc		J&N	
Oracle	τ	ρ	τ	ρ	τ	ρ	τ	ρ
Query budget T = 10, $\alpha = 0.3$:								
PO	.211	.310	.241	.353	**.253***	**.370***	.217	.319
CNO-0.1	.208	.307	.231	.339	**.240***	**.351***	.211	.311
CNO-0.3	.210	.309	.218	.320	**.229***	**.337***	.205	.302
LNO-0.3	.210	.309	.216	.318	**.231***	**.339***	.209	.307
LNO-1	.206	.303	.210	.308	**.211**	**.310**	.207	.305
Query budget T = 100, $\alpha = 0.7$:								
PO	.258	.377	.340	.490	**.418***	**.587***	.255	.372
CNO-0.1	.248	.363	.317	.459	**.386***	**.549***	.247	.362
CNO-0.3	.212	.312	.271	.396	**.330***	**.476***	.232	.340
LNO-0.3	.231	.339	.277	.404	**.324***	**.467***	.229	.336
LNO-1	.210	.309	**.225**	.330	**.225**	**.331**	.213	.313
Baseline, $\alpha = 0, T = 0$: $\tau = .206, \rho = .304$								

Table 2: Performance of multiple APL algorithms (columns) using different oracles and query budgets (rows). The baseline is the purely prior ranking. All results except the baseline are averaged over 50 document clusters in DUC'04. Asterisk: significant advantage over other active learning strategies given the same oracle and budget T.

ranking to select one good and one bad summary to query, while Unc performs more exploration by querying the summaries that are least confident according to the current ranking. We believe this explains the strong overall performance of Unc.

Additional experiments suggest that when we only use the posterior ranking (i.e. letting $\alpha = 1$), no APL we test can surpass the baseline when $T = 10$. Detailed results are presented in the supplementary material. This observation reflects the severity of the cold-start problem, confirms the effectiveness of our prior-posterior trade-off mechanism in combating cold-start, and indicates the importance of tuning the α value (see Eq. (7)). This opens up exciting avenues for future work.

6.2 RL Phase Performance

We compare two RL algorithms: TD(λ) (Sutton, 1984) and LSTD(λ) (Boyan, 1999). TD(λ) has been used in previous RL-based EMDS work (Ryang and Abekawa, 2012; Rioux et al., 2014). LSTD(λ) is chosen, because it is an improved TD algorithm and has been used in the state-of-the-art PbRL algorithm by Wirth et al. (2016). We let the learning round (see Alg. 2) $N = 5,000$, which we found to yield good results in reasonable time (less than 1 minute to generate a summary for one document cluster). Letting $N = 3,000$ will result in a significant performance drop, while increasing N to 10,000 will only bring marginal improvement at the cost of doubling the runtime. The learning parameters we use for TD(λ) are the same as those by Rioux et al. (2014). For LSTD(λ), we let $\lambda = 1$ and initialise its square matrix as a diagonal matrix with random numbers between 0 and 1, as suggested by Lagoudakis and Parr (2003). The rewards we use are the U^* function introduced in §5. Note that this serves as the upper-bound performance, because U^* relies on the reference summaries (see Eq. (6)), which are not available in the interactive setting. As a baseline, we also present the upper-bound performance of integer linear programming (ILP) reported by P.V.S. and Meyer (2017), optimised for bigram coverage.

Method	R_1	R_2	R_L	R_{SU4}
TD(λ)	**.484**	.184	**.388**	**.199**
LSTD(λ)	.458	.159	.366	.185
ILP	.470	**.212**	N/A	.185

Table 3: Upper-bound performance comparison. Results are averaged over all clusters in DUC'04.

Table 3 shows the performance of RL and ILP on the DUC'04 dataset. TD(λ) significantly outperforms LSTD(λ) in terms of all ROUGE scores we consider. Although the least-square RL algorithms (which LSTD belongs to) have been proved to achieve better performance than standard TD methods in large-scale problems (see Lagoudakis and Parr, 2003), their performance is sensitive to many factors, e.g., initialisation values in the diagonal matrix, regularisation parameters, etc. We note that a similar observation about the inferior performance of least-square RL in EMDS is reported by Rioux et al. (2014).

TD(λ) also significantly outperforms ILP in terms of all metrics except ROUGE-2. This is not surprising, because the bigram-based ILP is optimised for ROUGE-2, whereas our reward function U^* considers other metrics as well (see Eq. (6)). Since ILP is widely used as a strong baseline for EMDS, these results confirm the advantage of using RL for EMDS problems.

6.3 Complete Pipeline Performance

Finally, we combine the best techniques of the APL and RL phase (namely Unc and TD(λ), respectively) to complete APRIL, and compare it against SPPI. As a baseline, we use the heuristic-based rewards HU to train both TD(λ) (ranking-based training, i.e. using HU to produce $r(y|x)$ in Eq. (5) to train) and SPPI (preference-based training, i.e. using HU for generating pairs to train

			DUC '01				DUC '02				DUC '04			
Oracle	Method	T	R_1	R_2	R_L	R_{SU4}	R_1	R_2	R_L	R_{SU4}	R_1	R_2	R_L	R_{SU4}
PO	SPPI	10	.332	.075	.264	.104	.357	.083	.280†	.116	.378	.098	.299	.129
	APRIL	10	**.357**	.087	**.283**	**.119**	**.390**	**.108**	**.306**	**.133**	**.410**	**.116**	**.325**	**.149**
	SPPI	100	.353	.091	.284	.119	.391	.104	.306	.136	.392	.106	.312	.140
	APRIL	100	**.363**	**.091**	**.283**	**.118**	**.393**	**.107**	**.310**	**.137**	**.415**	**.118**	**.325**	**.151**
CNO-0.1	SPPI	10	.331	.081	.265	.103	.358	.081	.279†	.114†	.372†	.093†	.295†	.125†
	APRIL	10	**.351**	.081	**.276**	.112	**.376**	**.102**	**.296**	**.126**	**.403**	**.111**	**.320**	**.145**
	SPPI	100	.350	.089	.279	.117	.377	.100	.294	.129	.390	.107	.309	.138
	APRIL	100	**.353**	.084	**.280**	**.115**	**.385**	**.103**	**.302**	**.134**	**.411**	**.117**	**.325**	**.151**
CNO-0.3	SPPI	10	.320†	.063†	.253†	.096†	.354†	.080	.278†	.113†	.370†	.093†	.295†	125†
	APRIL	10	.339	**.076**	**.266**	**.108**	**.370**	**.091**	.290	**.124**	**.394**	**.104**	**.312**	**.138**
	SPPI	100	.345	.079	.270	.111	.373	.094	.295	.125	.386	.104	.307	.136
	APRIL	100	**.349**	**.081**	**.275**	**.109**	**.376**	**.097**	**.296**	**.127**	**.404**	**.114**	**.320**	**.146**
LNO-0.3	SPPI	10	.319†	.067†	.253†	.096†	.354†	.083	.280†	.113†	.375†	.095†	.294†	.127†
	APRIL	10	**.347**	**.084**	**.275**	**.109**	**.370**	**.095**	**.289**	**.125**	**.398**	**.108**	**.311**	**.141**
	SPPI	100	.321†	.068†	.252†	.097†	.352†	.080	.278†	.112†	.387	.104	.309	.136
	APRIL	100	**.350**	**.086**	**.277**	**.123**	**.380**	**.079**	**.296**	**.129**	**.407**	**.112**	**.321**	**.147**
LNO-1	SPPI	10	.314†	.058†	.250†	.092†	.348†	.076†	.273†	.110†	.373†	.096†	.297†	.126†
	APRIL	10	**.337**	**.072**	**.266**	**.104**	**.362**	**.085**	**.286**	**.119**	**.388**	**.102**	**.307**	**.134**
	SPPI	100	.320†	.064†	.255†	.097†	.351†	.078†	.273†	.113†	.381	.099	.301	.132
	APRIL	100	**.347**	**.080**	**.274**	**.109**	**.369**	**.089**	**.286**	**.123**	**.391**	**.101**	**.308**	**.136**
Baselines	SPPI	0	.323	.068	.259	.098	.350	.077	.278	.112	.372	.093	.293	.125
	TD(λ)	0	.324	.069	.256	.099	.350	.081	.276	.113	.372	.086	.292	.122

Table 4: Comparison of APRIL and SPPI. All results are averaged over all clusters in each dataset. Baselines: HU-trained SPPI and TD(λ), without any interaction (i.e. $T = 0$). Boldface: Comparable (i.e. no significant gaps exist) or significantly better than SPPI with 100 rounds of interaction, under the same oracle. Superscript †: Comparable or significantly worse than the corresponding baseline.

	DUC'01	DUC'02	DUC'04	Overall
APRIL	**3.57±.30**	**4.14±.14**	**3.86±.40**	**3.86±.17**
SPPI	2.29±.29	2.14±.14	3.14±.34	2.52±.18

Table 5: Human ratings for the summaries generated by APRIL and SPPI (mean±standard error).

SPPI) for up to 5,000 episodes. The baseline results are presented in the bottom rows of Table 4.

We make the following observations from Table 4. **(i)** Given the same oracle, the performance of APRIL with 10 rounds of interaction is comparable or even superior than that of SPPI after 100 rounds of interaction (see boldface in Table 4), suggesting the strong advantage of APRIL to reduce sample complexity. **(ii)** APRIL can significantly improve the baseline with either 10 or 100 rounds of interaction, but SPPI's performance can be even worse than the baseline (marked by † in Table 4), especially under the high-noise low-budget settings (i.e., CNO-0.3, LNO-0.3, and LNO-1 with $T = 10$). This is because SPPI lacks a mechanism to balance between prior and posterior ranking, while APRIL can adjust this trade-off by tuning α (Eq. (7)). This endows APRIL with better noise robustness and lower sample complexity in high-noise low-budget settings. Note that the above observations also hold for the other two datasets, indicating the consistently strong performance of APRIL across different datasets.

As for the overall runtime, when budget $T = 100$, APRIL on average takes 2 minutes to interact with an oracle and output a summary, while SPPI takes around 15 minutes due to its expensive gradient descent computation (see §3.1).

7 Human Evaluation

Finally, we invited real users to compare and evaluate the quality of the summaries generated by SPPI and APRIL. We randomly selected three topics (d19 from DUC'01, d117i from DUC'02 and d30042 from DUC'04), and let both SPPI and our best-performing APRIL interact with PO for 10 rounds on these topics. The resulting 100-word summaries, shown in Figure 2, were presented to seven users, who had already read two background texts to familiarize with the topic. The users were asked to provide their preference on the presented

Topic d30042 (DUC'04), SPPI: After meeting Libyan leader Moammar Gadhafi in a desert tent, U.N. Secretary-General Kofi Annan said he thinks an arrangement for bringing two suspects to trial in the bombing of a Pan Am airliner could be secured in the "not too distant future." TRIPOLI, Libya (AP) U.N. Secretary-General Kofi Annan arrived in Libya Saturday for talks aimed at bringing to trial two Libyan suspects in the 1988 Pan Am bombing over Lockerbie, Scotland. Secretary General Kofi Annan said Wednesday he was extending his North African tour to include talks with Libyan authorities. Annan's one-day, 2nd graf pvs During his Algerian stay,	**Topic d30042 (DUC'04), APRIL**: TRIPOLI, Libya (AP) U.N. Secretary-General Kofi Annan arrived in Libya Saturday for talks aimed at bringing to trial two Libyan suspects in the 1988 Pan Am bombing over Lockerbie, Scotland. Annan's one-day visit to meet with Libyan leader Col. Moammar Gadhafi followed reports in the Libyan media that Gadhafi had no authority to hand over the suspects. The 60-year-old Annan is trying to get Libya to go along with a U.S.-British plan to try the two suspects before a panel of Scottish judges in the Netherlands for the Dec. 21, 1988, bombing over Lockerbie, Scotland. Sirte is 400 kilometers (250 miles) east of the Libyan capital Tripoli. During his Algerian stay,
Topic d117i (DUC'02), SPPI: The Booker Prize is sponsored by Booker, an international food and agriculture business. The novel, a story of Scottish low-life narrated largely in Glaswegian dialect, is unlikely to prove a popular choice with booksellers, who have damned all six books shortlisted for the prize as boring, elitist and- worst of all- unsaleable. The shortlist of six for the Pounds 20,000 Booker Prize for fiction, announced yesterday, immediately prompted the question 'Who ? ' Japanese writer Kazuo Ishiguro won the 1989 Booker Prize, Britain's top literary award, for his novel "The Remains of the Day," judges announced Thursday. He didn't win.	**Topic d117i (DUC'02), APRIL**: Australian novelist Peter Carey was awarded the coveted Booker Prize for fiction Tuesday night for his love story, "Oscar and Lucinda." The Booker Prize is sponsored by Booker, an international food and agriculture business, and administered by The Book Trust. British publishers can submit three new novels by British and Commonwealth writers. Six novels have been nominated for the Booker Prize, Britain's most prestigious fiction award, and bookmakers say the favorite is "The Remains of the Day" by Japanese author Kazuo Ishiguro. On the day of the Big Event, Ladbroke, the large British betting agency, posted the final odds.
Topic d19 (DUC'01), SPPI: The issue cuts across partisan lines in the Senate, with Minority Leader Bob Dole (R-Kan.) arguing against the White House position on grounds that including illegal aliens in the census is unfair to American citizens.. Loss of Seats Cited. Shelby's amendment says only that the secretary is to "make such adjustments in total population figures as may be necessary, using such methods and procedures as the secretary determines feasible and appropriate" to keep illegal aliens from being counted in congressional reapportionment. "Some states will lose congressional seats because of illegal aliens," Dole argued. But there's nothing simple about it.	**Topic d19 (DUC'01), APRIL**: In a blow to California and other states with large immigrant populations, the Senate voted Friday to bar the Census Bureau from counting illegal aliens in the 1990 population count. But the Senate already has voted to force the Census Bureau to exclude illegal immigrants in preparing tallies for congressional reapportionment. said that Georgia and Indiana both lost House seats after the 1980 Census, and California and New York- centers of illegal immigration- each gained seats. A majority of the members of the House of Representatives has signaled support. The national head count will be taken April 1, 1990.

Figure 2: Summaries generated by SPPI and APRIL used in the human evaluation experiments.

summary pairs and rate the summaries on a 5-point Likert scale with higher scores for better summaries. All users are fluent in English.

In all three topics, all users prefer the APRIL-generated summaries over the SPPI-generated summaries. Table 5 shows the users' ratings. The APRIL-generated summaries consistently receive higher ratings. These results are consistent with our simulation experiments and confirm the significant advantage of APRIL over SPPI.

8 Conclusion

We propose a novel preference-based interactive learning formulation named APRIL (Active Preference ReInforcement Learning), which is able to make structured predictions without referring to the gold standard data. Instead, APRIL learns from preference-based feedback. We designed a novel objective function for APRIL, which naturally splits APRIL into an *active preference learning* (APL) phase and a *reinforcement learning* (RL) phase, enabling us to leverage a wide spectrum of active learning, preference learning and RL algorithms to maximise the output quality with a limited number of interaction rounds. We applied APRIL to the Extractive Multi-Document Summarisation (EMDS) problem, simulated the users' preference-giving behaviour using multiple user-response models, and compared the performance of multiple APL and RL techniques. Simulation experiments indicated that APRIL significantly improved the summary quality with just 10 rounds of interaction (even with high-noise oracles), and significantly outperformed SPPI in terms of both sample complexity and noise robustness. Human evaluation results suggested that real users preferred the APRIL-generated summaries over the SPPI-generated ones.

We identify two major lines of future work. On the technical side, we plan to employ more advanced APL and RL algorithms in APRIL, such as sample-efficient Bayesian-based APL algorithms (e.g., Simpson and Gurevych, 2018) and neural RL algorithms (e.g. Mnih et al., 2015) to further reduce the sample complexity of APRIL. On the experimental side, a logical next step is to implement an interactive user interface for APRIL and conduct a larger evaluation study comparing the summary quality before and after the interaction. We also plan to apply APRIL to more NLP applications, including machine translation, information exploration and semantic parsing.

Acknowledgements

This work has been supported by the German Research Foundation as part of the QA-EduInf project (grant GU 798/18-1 and grant RI 803/12-1). We thank the researchers and students from TU Darmstadt who participated in our human evaluation experiments. We also thank Johannes Fürnkranz, Christian Wirth and the anonymous reviewers for their helpful comments.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly.

Justin A. Boyan. 1999. Least-squares temporal difference learning. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), June 27–30, 1999, Bled, Slovenia*, pages 49–56.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems 30: 31st Annual Conference on Neural Information Processing Systems, December 4–9, 2017, Long Beach, CA, USA*, pages 4302–4310.

Wei Chu and Zoubin Ghahramani. 2005. Preference learning with Gaussian processes. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), August 7–11, 2005, Bonn, Germany*, pages 137–144.

Johannes Fürnkranz and Eyke Hüllermeier. 2010. Preference learning: An introduction. In *Preference Learning*, pages 1–17. Berlin/Heidelberg: Springer.

Stefan Henß, Margot Mieskes, and Iryna Gurevych. 2015. A reinforcement learning approach for adaptive single- and multi-document summarization. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology (GSCL 2015), September 30–October 2, 2015, University of Duisburg-Essen, Germany*, pages 3–12.

Kevin G. Jamieson and Robert D. Nowak. 2011. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems, December 12–14, 2011, Granada, Spain*, pages 2240–2248.

M.G. Kendall. 1948. *Rank correlation methods*. C. Griffin.

David C Kingsley and Thomas C Brown. 2010. Preference uncertainty, preference refinement and paired comparison choice experiments. *Land Economics*, 86(3):530–544.

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. 2017. Bandit structured prediction for neural sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017), Volume 1: Long Papers, July 30–August 4, 2017, Vancouver, Canada*, pages 1503–1513.

Michail G Lagoudakis and Ronald Parr. 2003. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL, July 21–26, 2004, Barcelona, Spain*, pages 74–81.

Lucas Maystre and Matthias Grossglauser. 2017. Just sort it! A simple and effective approach to active preference learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017), August 6–11, 2017, Sydney, Australia*, pages 2344–2353.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018),Volume 1 (Long Papers), June 1-6, 2018, New Orleans, LA, USA*, pages 1747–1759.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 43–76. Boston: Springer.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018), Volume 2 (Short Papers), June 1-6, 2018, New Orleans, LA, USA*, pages 646–653.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Avinesh P.V.S. and Christian M. Meyer. 2017. Joint optimization of user-desired content in multi-document summaries by learning from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017): Volume 1: Long Paper, July 30–August 4, 2017, Vancouver, Canada*, pages 1353–1363.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Cody Rioux, Sadid A. Hasan, and Yllias Chali. 2014. Fear the REAPER: A system for automatic multi-document summarization with reinforcement learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), October 25–29, 2014, Doha, Qatar*, pages 681–690.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012), July 12–14, 2012, Jeju Island, Korea*, pages 256–265.

Edwin D. Simpson and Iryna Gurevych. 2018. Finding convincing arguments using scalable bayesian preference learning. *Transactions of the Association for Computational Linguistics*, 6:357–371.

Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. 2016a. Learning structured predictors from bandit feedback for interactive NLP. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016): Volume 1: Long Papers, August 7–12, 2016, Berlin, Germany*, pages 1610–1620.

Artem Sokolov, Julia Kreutzer, Stefan Riezler, and Christopher Lo. 2016b. Stochastic structured prediction under bandit feedback. In *Advances in Neural Information Processing Systems 29: 30th Annual Conference on Neural Information Processing Systems, December 5–10, 2016, Barcelona, Spain*, pages 1489–1497.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101.

R. S. Sutton. 1984. *Temporal Credit Assignment in Reinforcement Learning*. Ph.D. thesis, University of Massachusetts, Amherst.

Louis Leon Thurstone. 1927. A Law of Comparative Judgement. *Psychological Review*, 34:278–286.

Paolo Viappiani and Craig Boutilier. 2010. Optimal Bayesian recommendation sets and myopically optimal choice query sets. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems, December 6-9 , Vancouver, British Columbia, Canada*, pages 2352–2360.

Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. 2017. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18:4945–4990.

Christian Wirth, Johannes Fürnkranz, and Gerhard Neumann. 2016. Model-free preference-based reinforcement learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12–17, 2016, Phoenix, AZ, USA*, pages 2222–2228.

Markus Zopf. 2018. Estimating summary quality with pairwise preferences. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018), Volume 1 (Long Papers), June 1–8, 2018, New Orleans, LA, USA*, pages 1687–1696.

Adapting the Neural Encoder-Decoder Framework from Single to Multi-Document Summarization

Logan Lebanoff, Kaiqiang Song and Fei Liu
Department of Computer Science
University of Central Florida, Orlando, FL 32816, USA
`{loganlebanoff, kqsong}@knights.ucf.edu` `feiliu@cs.ucf.edu`

Abstract

Generating a text abstract from a set of documents remains a challenging task. The neural encoder-decoder framework has recently been exploited to summarize single documents, but its success can in part be attributed to the availability of large parallel data automatically acquired from the Web. In contrast, parallel data for multi-document summarization are scarce and costly to obtain. There is a pressing need to adapt an encoder-decoder model trained on single-document summarization data to work with multiple-document input. In this paper, we present an initial investigation into a novel adaptation method. It exploits the maximal marginal relevance method to select representative sentences from multi-document input, and leverages an abstractive encoder-decoder model to fuse disparate sentences to an abstractive summary. The adaptation method is robust and itself requires no training data. Our system compares favorably to state-of-the-art extractive and abstractive approaches judged by automatic metrics and human assessors.

1 Introduction

Neural abstractive summarization has primarily focused on summarizing short texts written by single authors. For example, *sentence summarization* seeks to reduce the first sentence of a news article to a title-like summary (Rush et al., 2015; Nallapati et al., 2016; Takase et al., 2016; Song et al., 2018); *single-document summarization* (**SDS**) focuses on condensing a news article to a handful of bullet points (Paulus et al., 2017; See et al., 2017). These summarization studies are empowered by large parallel datasets automatically harvested from online news outlets, including Gigaword (Rush et al., 2015), CNN/Daily Mail (Hermann et al., 2015), NYT (Sandhaus, 2008), and Newsroom (Grusky et al., 2018).

To date, *multi-document summarization* (**MDS**) has not yet fully benefited from the development of neural encoder-decoder models. MDS seeks to condense a set of documents likely written by multiple authors to a short and informative summary. It has practical applications, such as summarizing product reviews (Gerani et al., 2014), student responses to post-class questionnaires (Luo and Litman, 2015; Luo et al., 2016), and sets of news articles discussing certain topics (Hong et al., 2014). State-of-the-art MDS systems are mostly extractive (Nenkova and McKeown, 2011). Despite their promising results, such systems cannot perform text abstraction, e.g., paraphrasing, generalization, and sentence fusion (Jing and McKeown, 1999). Further, annotated MDS datasets are often scarce, containing only hundreds of training pairs (see Table 1). The cost to create ground-truth summaries from multiple-document inputs can be prohibitive. The MDS datasets are thus too small to be used to train neural encoder-decoder models with millions of parameters without overfitting.

DATASET	SOURCE	SUMMARY	#PAIRS
Gigaword (Rush et al., 2015)	**the first sentence** of a news article	**8.3 words** title-like	**4 Million**
CNN/Daily Mail (Hermann et al., 2015)	**a news article**	**56 words** multi-sent	**312 K**
TAC (08-11) (Dang et al., 2008)	**10 news articles** related to a topic	**100 words** multi-sent	**728**
DUC (03-04) (Over and Yen, 2004)	**10 news articles** related to a topic	**100 words** multi-sent	**320**

Table 1: A comparison of datasets available for sent. summarization (Gigaword), single-doc (CNN/DM) and multi-doc summarization (DUC/TAC). The labelled data for multi-doc summarization are much less.

A promising route to generating an abstractive summary from a multi-document input is to apply a neural encoder-decoder model trained for single-document summarization to a "*mega-document*" created by concatenating all documents in the set at test time. Nonetheless, such a model may not scale well for two reasons. First, identifying important text pieces from a mega-document can be challenging for the encoder-decoder model, which is trained on single-document summarization data

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4131–4141
Brussels, Belgium, October 31 - November 4, 2018.

where the summary-worthy content is often contained in the first few sentences of an article. This is not the case for a mega-document. Second, redundant text pieces in a mega-document can be repeatedly used for summary generation under the current framework. The attention mechanism of an encoder-decoder model (Bahdanau et al., 2014) is position-based and lacks an awareness of semantics. If a text piece has been attended to during summary generation, it is unlikely to be used again. However, the attention value assigned to a similar text piece in a different position is not affected. The same content can thus be repeatedly used for summary generation. These issues may be alleviated by improving the encoder-decoder architecture and its attention mechanism (Cheng and Lapata, 2016; Tan et al., 2017). However, in these cases the model has to be re-trained on large-scale MDS datasets that are not available at the current stage. There is thus an increasing need for a lightweight adaptation of an encoder-decoder model trained on SDS datasets to work with multi-document inputs at test time.

In this paper, we present a novel adaptation method, named PG-MMR, to generate abstracts from multi-document inputs. The method is robust and requires no MDS training data. It combines a recent neural encoder-decoder model (PG for Pointer-Generator networks; See et al., 2017) that generates abstractive summaries from single-document inputs with a strong extractive summarization algorithm (MMR for Maximal Marginal Relevance; Carbonell and Goldstein, 1998) that identifies important source sentences from multi-document inputs. The PG-MMR algorithm iteratively performs the following. It identifies a handful of the most important sentences from the mega-document. The attention weights of the PG model are directly modified to focus on these important sentences when generating a summary sentence. Next, the system re-identifies a number of important sentences, but the likelihood of choosing certain sentences is reduced based on their similarity to the partially-generated summary, thereby reducing redundancy. Our research contributions include the following:

- we present an investigation into a novel adaptation method of the encoder-decoder framework from single- to multi-document summarization. To the best of our knowledge, this is the first attempt to couple the maximal marginal relevance algorithm with pointer-generator networks for multi-document summarization;
- we demonstrate the effectiveness of the proposed method through extensive experiments on standard MDS datasets. Our system compares favorably to state-of-the-art extractive and abstractive summarization systems measured by both automatic metrics and human judgments.

2 Related Work

Popular methods for multi-document summarization have been extractive. Important sentences are extracted from a set of source documents and optionally compressed to form a summary (Daume III and Marcu, 2002; Zajic et al., 2007; Gillick and Favre, 2009; Galanis and Androutsopoulos, 2010; Berg-Kirkpatrick et al., 2011; Li et al., 2013; Thadani and McKeown, 2013; Wang et al., 2013; Yogatama et al., 2015; Filippova et al., 2015; Durrett et al., 2016). In recent years neural networks have been exploited to learn word/sentence representations for single- and multi-document summarization (Cheng and Lapata, 2016; Cao et al., 2017; Isonuma et al., 2017; Yasunaga et al., 2017; Narayan et al., 2018). These approaches remain extractive; and despite encouraging results, summarizing a large quantity of texts still requires sophisticated abstraction capabilities such as generalization, paraphrasing and sentence fusion.

Prior to deep learning, abstractive summarization has been investigated (Barzilay et al., 1999; Carenini and Cheung, 2008; Ganesan et al., 2010; Gerani et al., 2014; Fabbrizio et al., 2014; Pighin et al., 2014; Bing et al., 2015; Liu et al., 2015; Liao et al., 2018). These approaches construct domain templates using a text planner or an open-IE system and employ a natural language generator for surface realization. Limited by the availability of labelled data, experiments are often performed on small domain-specific datasets.

Neural abstractive summarization utilizing the encoder-decoder architecture has shown promising results but studies focus primarily on single-document summarization (Nallapati et al., 2016; Kikuchi et al., 2016; Chen et al., 2016; Miao and Blunsom, 2016; Tan et al., 2017; Zeng et al., 2017; Zhou et al., 2017; Paulus et al., 2017; See et al., 2017; Gehrmann et al., 2018). The pointing mechanism (Gulcehre et al., 2016; Gu et al., 2016) allows a summarization system to both copy words from the source text and generate new words from the vocabulary. Reinforcement learning is exploited to directly optimize evaluation metrics (Paulus et al., 2017; Kryściński et al., 2018; Chen and Bansal, 2018). These studies focus on summarizing single documents in part because the training data are abundant.

The work of Baumel et al. (2018) and Zhang et al. (2018) are related to ours. In particular, Baumel et al. (2018) propose to extend an abstractive summarization system to generate query-focused summaries; Zhang et al. (2018) add a document set encoder to their hierarchical summarization framework. With these few exceptions, little research has been dedicated to investigate the feasibility of extending the encoder-decoder framework to generate abstractive summaries from multi-document inputs, where available training data are scarce.

This paper presents some first steps towards the goal of extending the encoder-decoder model to a multi-document setting. We introduce an adaptation method combining the pointer-generator (PG) networks (See et al., 2017) and the maximal marginal relevance (MMR) algorithm (Carbonell and Goldstein, 1998). The PG model, trained on SDS data and detailed in Section §3, is capable of generating document abstracts by performing text abstraction and sentence fusion. However, if the model is applied at test time to summarize multi-document inputs, there will be limitations. Our PG-MMR algorithm, presented in Section §4, teaches the PG model to effectively recognize important content from the input documents, hence improving the quality of abstractive summaries, all without requiring any training on multi-document inputs.

3 Limits of the Encoder-Decoder Model

The encoder-decoder architecture has become the *de facto* standard for neural abstractive summarization (Rush et al., 2015). The encoder is often a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) converting the input text to a set of hidden states $\{\mathbf{h}_i^e\}$, one for each input word, indexed by i. The decoder is a unidirectional LSTM that generates a summary by predicting one word at a time. The decoder hidden states are represented by $\{\mathbf{h}_t^d\}$, indexed by t. For sentence and single-document summarization (Nallapati et al., 2016; Paulus et al., 2017; See et al., 2017), the input text is treated as a sequence of words, and the model is expected to capture the source syntax inherently.

$$e_{t,i} = \mathbf{v}^\top \tanh(\mathbf{W}^e[\mathbf{h}_t^d||\mathbf{h}_i^e||\widetilde{\alpha}_{t,i}] + \mathbf{b}^e) \quad (1)$$
$$\alpha_{t,i} = \text{softmax}(e_{t,i}) \quad (2)$$
$$\widetilde{\alpha}_{t,i} = \textstyle\sum_{t'=0}^{t-1} \alpha_{t',i} \quad (3)$$

The attention weight $\alpha_{t,i}$ measures how important the i-th input word is to generating the t-th output word (Eq. (1-2)). Following (See et al., 2017), $\alpha_{t,i}$ is calculated by measuring the strength of interaction between the decoder hidden state $\mathbf{h}_t^d$, the encoder hidden state $\mathbf{h}_i^e$, and the *cumulative* attention $\widetilde{\alpha}_{t,i}$ (Eq. (3)). $\widetilde{\alpha}_{t,i}$ denotes the cumulative attention that the i-th input word receives up to time step t-1. A large value of $\widetilde{\alpha}_{t,i}$ indicates the i-th input word has been used prior to time t and it is unlikely to be used again for generating the t-th output word.

A context vector ($\mathbf{c}_t$) is constructed (Eq. (4)) to summarize the semantic meaning of the input; it is a weighted sum of the encoder hidden states. The context vector and the decoder hidden state ($[\mathbf{h}_t^d||\mathbf{c}_t]$) are then used to compute the vocabulary probability $P_{vcb}(w)$ measuring the likelihood of a vocabulary word w being selected as the t-th output word (Eq. (5)).[1]

$$\mathbf{c}_t = \textstyle\sum_i \alpha_{t,i}\mathbf{h}_i^e \quad (4)$$
$$P_{vcb}(w) = \text{softmax}(\mathbf{W}^y[\mathbf{h}_t^d||\mathbf{c}_t] + \mathbf{b}^y) \quad (5)$$

In many encoder-decoder models, a "switch" is estimated ($p_{gen} \in [0,1]$) to indicate whether the system has chosen to select a word from the vocabulary or to copy a word from the input text (Eq. (6)). The switch is computed using a feedforward layer with σ activation over $[\mathbf{h}_t^d||\mathbf{c}_t||\mathbf{y}_{t-1}]$, where $\mathbf{y}_{t-1}$ is the embedding of the output word at time t-1. The attention weights ($\alpha_{t,i}$) are used to compute the copy probability (Eq. (7)). If a word w appears once or more in the input text, its copy probability ($\sum_{i:w_i=w} \alpha_{t,i}$) is the sum of the attention weights over all its occurrences. The final probability $P(w)$ is a weighted combination of the vocabulary probability and the copy probability. A cross-entropy loss function can often be used to train the model end-to-end.

$$p_{gen}=\sigma(\mathbf{w}^z[\mathbf{h}_t^d||\mathbf{c}_t||\mathbf{y}_{t-1}])+b^z) \quad (6)$$
$$P(w)=p_{gen}P_{vcb}(w)+(1-p_{gen})\sum_{i:w_i=w}\alpha_{t,i} \quad (7)$$

To thoroughly understand the aforementioned encoder-decoder model, we divide its model parameters into four groups. They include

- parameters of the encoder and the decoder;
- $\{\mathbf{w}^z, b^z\}$ for calculating the "switch" (Eq. (6));

---

[1] Here $[\cdot||\cdot]$ represents the concatenation of two vectors. The pointer-generator networks (See et al., 2017) use two linear layers to produce the vocabulary distribution $P_{vcb}(w)$. We use $\mathbf{W}^y$ and $\mathbf{b}^y$ to denote parameters of both layers.

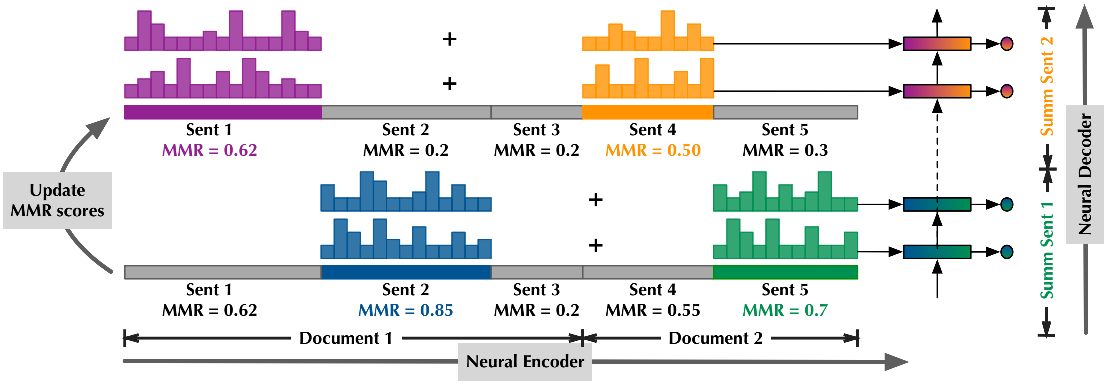

Figure 1: System framework. The PG-MMR system uses K highest-scored source sentences (in this case, K=2) to guide the PG model to generate a summary sentence. All other source sentences are "muted" in this process. Best viewed in color.

- $\{\mathbf{W}^y, \mathbf{b}^y\}$ for calculating $P_{vcb}(w)$ (Eq. (5));
- $\{\mathbf{v}, \mathbf{W}^e, \mathbf{b}^e\}$ for attention weights (Eq. (1)).

By training the encoder-decoder model on single-document summarization (SDS) data containing a large collection of news articles paired with summaries (Hermann et al., 2015), these model parameters can be effectively learned.

However, at test time, we wish for the model to generate abstractive summaries from *multi-document inputs*. This brings up two issues. First, the parameters are ineffective at identifying salient content from multi-document inputs. Humans are very good at identifying representative sentences from a set of documents and fusing them into an abstract. However, this capability is not supported by the encoder-decoder model. Second, the attention mechanism is based on input word positions but not their semantics. It can lead to redundant content in the multi-document input being repeatedly used for summary generation. We conjecture that both aspects can be addressed by introducing an "external" model that selects representative sentences from multi-document inputs and dynamically adjusts the sentence importance to reduce summary redundancy. This external model is integrated with the encoder-decoder model to generate abstractive summaries using selected representative sentences. In the following section we present our adaptation method for multi-document summarization.

4 Our Method

Maximal marginal relevance. Our adaptation method incorporates the maximal marginal relevance algorithm (MMR; Carbonell and Goldstein, 1998) into pointer-generator networks (PG; See et al., 2017) by adjusting the network's attention values. MMR is one of the most successful extractive approaches and, despite its straightforwardness, performs on-par with state-of-the-art systems (Luo and Litman, 2015; Yogatama et al., 2015). At each iteration, MMR selects one sentence from the document (D) and includes it in the summary (S) until a length threshold is reached. The selected sentence (s_i) is the most important one amongst the remaining sentences and it has the least content overlap with the current summary. In the equation below, $\text{Sim}_1(s_i, D)$ measures the similarity of the sentence s_i to the document. It serves as a proxy of sentence importance, since important sentences usually show similarity to the centroid of the document. $\max_{s_j \in S} \text{Sim}_2(s_i, s_j)$ measures the maximum similarity of the sentence s_i to each of the summary sentences, acting as a proxy of redundancy. λ is a balancing factor.

$$\underset{s_i \in D \setminus S}{\text{argmax}} \Big[\underbrace{\lambda \text{Sim}_1(s_i, D)}_{\text{importance}} - \underbrace{(1-\lambda) \max_{s_j \in S} \text{Sim}_2(s_i, s_j)}_{\text{redundancy}} \Big]$$

Our PG-MMR describes an iterative framework for summarizing a multi-document input to a summary consisting of multiple sentences. At each iteration, PG-MMR follows the MMR principle to select the K highest-scored source sentences; they serve as the basis for PG to generate a summary sentence. After that, the scores of all source sentences are updated based on their importance and redundancy. Sentences that are highly similar to the partial summary receive lower scores. Selecting K sentences via the MMR algorithm helps the PG system to effectively identify salient source content that has not been included in the summary.

Muting. To allow the PG system to effectively utilize the K source sentences without retraining the neural model, we dynamically adjust the PG attention weights ($\alpha_{t,i}$) at test time. Let S_k represent a selected sentence. The attention weights of the words belonging to $\{\mathrm{S}_k\}_{k=1}^K$ are calculated as before (Eq. (2)). However, words in other sentences are forced to receive zero attention weights ($\alpha_{t,i}$=0), and all $\alpha_{t,i}$ are renormalized (Eq. (8)).

$$\alpha_{t,i}^{\text{new}} = \begin{cases} \alpha_{t,i} & i \in \{\mathrm{S}_k\}_{k=1}^K \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

It means that the remaining sentences are "muted" in this process. In this variant, the sentence importance does not affect the original attention weights, other than muting.

In an alternative setting, the sentence salience is multiplied with the word salience and renormalized (Eq. (9)). PG uses the reweighted alpha values to predict the next summary word.

$$\alpha_{t,i}^{\text{new}} = \begin{cases} \alpha_{t,i}\text{MMR}(\mathrm{S}_k) & i \in \{\mathrm{S}_k\}_{k=1}^K \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Sentence Importance. To estimate sentence importance $\text{Sim}_1(s_i, D)$, we introduce a supervised regression model in this work. Importantly, the model is trained on single-document summarization datasets where training data are abundant. At test time, the model can be applied to identify important sentences from multi-document input. Our model determines sentence importance based on four indicators, inspired by how humans identify important sentences from a document set. They include (a) sentence length, (b) its absolute and relative position in the document, (c) sentence quality, and (d) how close the sentence is to the main topic of the document set. These features are considered to be important indicators in previous extractive summarization framework (Galanis and Androutsopoulos, 2010; Hong et al., 2014).

Regarding the sentence quality (c), we leverage the PG model to build the sentence representation. We use the bidirectional LSTM encoder to encode any source sentence to a vector representation. $[\overrightarrow{\mathbf{h}}_N^e || \overleftarrow{\mathbf{h}}_1^e]$ is the concatenation of the last hidden states of the forward and backward passes. A document vector is the average of all sentence vectors. We use the document vector and the cosine similarity between the document and sentence vectors as indicator (d). A support vector regression model is trained on (sentence, score) pairs where the training data are obtained from the CNN/Daily Mail dataset. The target importance score is the ROUGE-L recall of the sentence compared to the ground-truth summary. Our model architecture leverages neural representations of sentences and documents, they are data-driven and not restricted to a particular domain.

Algorithm 1 The PG-MMR algorithm for summarizing multi-document inputs.

```
Input: SDS data; MDS source sentences $\{\mathrm{S}_i\}$
Train the PG model on SDS data
▶ $\mathcal{I}(\mathrm{S}_i)$ and $\mathcal{R}(\mathrm{S}_i)$ are the importance and redundancy scores of the source sentence $\mathrm{S}_i$
$\mathcal{I}(\mathrm{S}_i) \leftarrow \text{SVR}(\mathrm{S}_i)$ for all source sentences
$\text{MMR}(\mathrm{S}_i) \leftarrow \lambda\mathcal{I}(\mathrm{S}_i)$ for all source sentences
Summary $\leftarrow \{\}$
$t \leftarrow$ index of summary words
while $t < L_{\text{max}}$ do
    Find $\{\mathrm{S}_k\}_{k=1}^K$ with highest MMR scores
    Compute $\alpha_{t,i}^{\text{new}}$ based on $\{\mathrm{S}_k\}_{k=1}^K$ (Eq. (8))
    Run PG decoder for one step to get $\{w_t\}$
    Summary $\leftarrow$ Summary + $\{w_t\}$
    if $w_t$ is the period symbol then
        $\mathcal{R}(\mathrm{S}_i) \leftarrow \text{Sim}(\mathrm{S}_i, \text{Summary}), \forall i$
        $\text{MMR}(\mathrm{S}_i) \leftarrow \lambda\mathcal{I}(\mathrm{S}_i) - (1-\lambda)\mathcal{R}(\mathrm{S}_i), \forall i$
    end if
end while
```

Sentence Redundancy. To calculate the redundancy of the sentence ($\max_{s_j \in S} \text{Sim}_2(s_i, s_j)$), we compute the ROUGE-L precision, which measures the longest common subsequence between a source sentence and the partial summary (consisting of all sentences generated thus far by the PG model), divided by the length of the source sentence. A source sentence yielding a high ROUGE-L precision is deemed to have significant content overlap with the partial summary. It will receive a low MMR score and hence is less likely to serve as basis for generating future summary sentences.

Alg. 1 provides an overview the PG-MMR algorithm and Fig. 1 is a graphical illustration. The MMR scores of source sentences are updated after each summary sentence is generated by the PG model. Next, a different set of highest-scored sentences are used to guide the PG model to generate the next summary sentence. "Muting" the remaining source sentences is important because it helps the PG model to focus its attention on the most significant source content. The code for our model is publicly available to further MDS research.[2]

5 Experimental Setup

Datasets. We investigate the effectiveness of the PG-MMR method by testing it on standard multi-document summarization datasets (Over and Yen,

---

[2] https://github.com/ucfnlp/multidoc_summarization

2004; Dang and Owczarzak, 2008). These include DUC-03, DUC-04, TAC-08, TAC-10, and TAC-11, containing 30/50/48/46/44 topics respectively. The summarization system is tasked with generating a concise, fluent summary of 100 words or less from a set of 10 documents discussing a topic. All documents in a set are chronologically ordered and concatenated to form a mega-document serving as input to the PG-MMR system. Sentences that start with a quotation mark or do not end with a period are excluded (Wong et al., 2008). Each system summary is compared against 4 human abstracts created by NIST assessors. Following convention, we report results on DUC-04 and TAC-11 datasets, which are standard test sets; DUC-03 and TAC-08/10 are used as a validation set for hyperparameter tuning.[3]

The PG model is trained for single-document summarization using the CNN/Daily Mail (Hermann et al., 2015) dataset, containing single news articles paired with summaries (human-written article highlights). The training set contains 287,226 articles. An article contains 781 tokens on average; and a summary contains 56 tokens (3.75 sentences). During training we use the hyperparameters provided by See et al. (2017). At test time, the maximum/minimum decoding steps are set to 120/100 words respectively, corresponding to the max/min lengths of the PG-MMR summaries. Because the focus of this work is on multi-document summarization (MDS), we do not report results for the CNN/Daily Mail dataset.

Baselines. We compare PG-MMR against a broad spectrum of baselines, including state-of-the-art extractive ('*ext-*') and abstractive ('*abs-*') systems. They are described below.[4]

- *ext-***SumBasic** (Vanderwende et al., 2007) is an extractive approach assuming words occurring frequently in a document set are more likely to be included in the summary;
- *ext-***KL-Sum** (Haghighi and Vanderwende, 2009) greedily adds source sentences to the summary if it leads to a decrease in KL divergence;
- *ext-***LexRank** (Erkan and Radev, 2004) uses a graph-based approach to compute sentence importance based on eigenvector centrality in a graph representation;
- *ext-***Centroid** (Hong et al., 2014) computes the importance of each source sentence based on its cosine similarity with the document centroid;
- *ext-***ICSISumm** (Gillick et al., 2009) leverages the ILP framework to identify a globally-optimal set of sentences covering the most important concepts in the document set;

System	DUC-04 R-1	R-2	R-SU4
SumBasic (Vanderwende et al., 2007)	29.48	4.25	8.64
KLSumm (Haghighi et al., 2009)	31.04	6.03	10.23
LexRank (Erkan and Radev, 2004)	34.44	7.11	11.19
Centroid (Hong et al., 2014)	35.49	7.80	12.02
ICSISumm (Gillick and Favre, 2009)	37.31	9.36	13.12
DPP (Taskar, 2012)	38.78	9.47	13.36
Extract+Rewrite (Song et al., 2018)	28.90	5.33	8.76
Opinosis (Ganesan et al., 2010)	27.07	5.03	8.63
PG-Original (See et al., 2017)	31.43	6.03	10.01
PG-MMR w/ SummRec	34.57	7.46	11.36
PG-MMR w/ SentAttn	36.52	8.52	12.57
PG-MMR w/ Cosine (*default*)	36.88	**8.73**	12.64
PG-MMR w/ BestSummRec	36.42	**9.36**	13.23

Table 2: ROUGE results on the DUC-04 dataset.

System	TAC-11 R-1	R-2	R-SU4
SumBasic (Vanderwende et al., 2007)	31.58	6.06	10.06
KLSumm (Haghighi et al., 2009)	31.23	7.07	10.56
LexRank (Erkan and Radev, 2004)	33.10	7.50	11.13
Extract+Rewrite (Song et al., 2018)	29.07	6.11	9.20
Opinosis (Ganesan et al., 2010)	25.15	5.12	8.12
PG-Original (See et al., 2017)	31.44	6.40	10.20
PG-MMR w/ SummRec	35.06	8.72	12.39
PG-MMR w/ SentAttn	37.01	10.43	13.85
PG-MMR w/ Cosine (*default*)	37.17	**10.92**	14.04
PG-MMR w/ BestSummRec	40.44	**14.93**	17.61

Table 3: ROUGE results on the TAC-11 dataset.

- *ext-***DPP** (Taskar, 2012) selects an optimal set of sentences per the determinantal point processes that balance the coverage of important information and the sentence diversity;
- *abs-***Opinosis** (Ganesan et al., 2010) generates abstractive summaries by searching for salient paths on a word co-occurrence graph created from source documents;
- *abs-***Extract+Rewrite** (Song et al., 2018) is a recent approach that scores sentences using LexRank and generates a title-like summary for each sentence using an encoder-decoder model trained on Gigaword data.
- *abs-***PG-Original** (See et al., 2017) introduces an encoder-decoder model that encourages the system to copy words from the source text via pointing, while retaining the ability to produce novel words through the generator.

6 Results

Having described the experimental setup, we next compare the PG-MMR method against the baselines on standard MDS datasets, evaluated by both automatic metrics and human assessors.

ROUGE (Lin, 2004). This automatic metric measures the overlap of unigrams (R-1), bigrams (R-2) and skip bigrams with a maximum distance of 4 words (R-SU4) between the system summary and a set of reference summaries. ROUGE scores of various systems are presented in Table 2 and 3 respectively for the DUC-04 and TAC-11 datasets.

We explore variants of the PG-MMR method.

---

[3] The hyperparameters for all PG-MMR variants are K=7 and λ=0.6; except for "w/ BestSummRec" where K=2.

[4] We are grateful to Hong et al. (2014) for providing the summaries generated by Centroid, ICSISumm, DPP systems. These are only available for the DUC-04 dataset.

They differ in how the importances of source sentences are estimated and how the sentence importance affects word attention weights. "***w/ Cosine***" computes the sentence importance as the cosine similarity score between the sentence and document vectors, both represented as sparse TF-IDF vectors under the vector space model. "***w/ SummRec***" estimates the sentence importance as the predicted R-L recall score between the sentence and the summary. A support vector regression model is trained on sentences from the CNN/Daily Mail datasets (≈33K) and applied to DUC/TAC sentences at test time (see §4). "***w/ BestSummRec***" obtains the best estimate of sentence importance by calculating the R-L recall score between the sentence and reference summaries. It serves as an upper bound for the performance of "w/ SummRec." For all variants, the sentence importance scores are normalized to the range of [0,1]. "***w/ SentAttn***" adjusts the attention weights using Eq. (9), so that words in important sentences are more likely to be used to generate the summary. The weights are otherwise computed using Eq. (8).

As seen in Table 2 and 3, our PG-MMR method surpasses all unsupervised extractive baselines, including SumBasic, KLSumm, and LexRank. On the DUC-04 dataset, ICSISumm and DPP show good performance, but these systems are trained directly on MDS datasets, which are not utilized by the PG-MMR method. PG-MMR exhibits superior performance compared to existing abstractive systems. It outperforms Opinosis and PG-Original by a large margin in terms of R-2 F-scores (5.03/6.03/**8.73** for DUC-04 and 5.12/6.40/**10.92** for TAC-11). In particular, ***PG-Original*** is the original pointer-generator networks with multi-document inputs at test time. Compared to it, PG-MMR is more effective at identifying summary-worthy content from the input. "w/ Cosine" is used as the default PG-MMR and it shows better results than "w/ SummRec." It suggests that the sentence and document representations obtained from the encoder-decoder model (trained on CNN/DM) are suboptimal, possibly due to a vocabulary mismatch, where certain words in the DUC/TAC datasets do not appear in CNN/DM and their embeddings are thus not learned during training. Finally, we observe that "w/ BestSummRec" yields the highest performance on both datasets. This finding suggests that there is a great potential for improvements of the PG-MMR method as its "extractive" and "abstractive" components can be separately optimized.

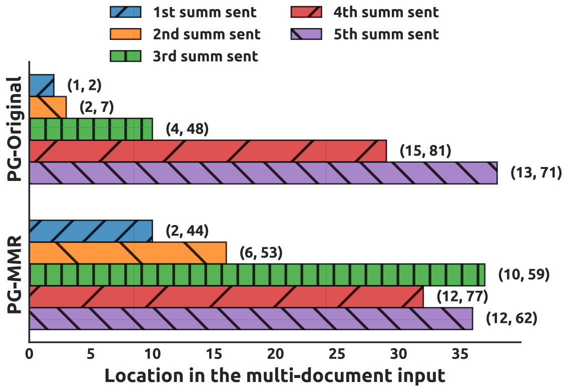

Figure 2: The median location of summary n-grams in the multi-document input (and the lower/higher quartiles). The n-grams come from the 1st/2nd/3rd/4th/5th summary sentence and the location is the source sentence index. (TAC-11)

System	1-grams	2-grams	3-grams	Sent
Extr+Rewrite	89.37	54.34	25.10	6.65
PG-Original	99.64	96.28	88.83	47.67
PG-MMR	99.74	97.64	91.57	59.13
Human Abst.	84.32	45.22	18.70	0.23

Table 4: Percentages of summary n-grams (or the entire sentences) appear in the multi-document input. (TAC-11)

Location of summary content. We are interested in understanding why PG-MMR outperforms PG-Original at identifying summary content from the multi-document input. We ask the question: where, in the source documents, does each system tend to look when generating their summaries? Our findings indicate that PG-Original gravitates towards early source sentences, while PG-MMR searches beyond the first few sentences.

In Figure 2 we show the median location of the first occurrences of summary n-grams, where the n-grams can come from the 1st to 5th summary sentence. For PG-Original summaries, n-grams of the 1st summary sentence frequently come from the 1st and 2nd source sentences, corresponding to the lower/higher quartiles of source sentence indices. Similarly, n-grams of the 2nd summary sentence come from the 2nd to 7th source sentences. For PG-MMR summaries, the patterns are different. The n-grams of the 1st and 2nd summary sentences come from source sentences of the range (2, 44) and (6, 53), respectively. Our findings suggest that PG-Original tends to treat the input as a single-document and identifies summary-worthy content from the beginning of the input, whereas PG-MMR can successfuly search a broader range of the input for summary content. This capability is crucial for multi-document input where important content can come from any article in the set.

Degree of extractiveness. Table 4 shows the

	Linguistic Quality			Rankings (%)			
System	Fluency	Inform.	NonRed.	1st	2nd	3rd	4th
Extract+Rewrite	2.03	2.19	1.88	5.6	11.6	11.6	**71.2**
LexRank	**3.29**	3.36	3.30	30.0	28.8	32.0	9.2
PG-Original	3.20	3.30	3.19	29.6	26.8	**32.8**	10.8
PG-MMR	3.24	**3.52**	**3.42**	**34.8**	**32.8**	23.6	8.8

Table 5: Linguistic quality and rankings of system summaries. (DUC-04)

Human Abstract	PG-Original Summary
• Boeing 737-400 plane with 102 people on board crashed into a mountain in the West Sulawesi province of Indonesia, on Monday, January 01, 2007, killing at least 90 passengers, with 12 possible survivors. • The plane was Adam Air flight KI-574, departing at 12:59 pm from Surabaya on Java bound for Manado in northeast Sulawesi. • The plane crashed in a mountainous region in Polewali, west Sulawesi province. • There were three Americans on board, it is not know if they survived. • The cause of the crash is not known at this time but it is possible bad weather was a factor.	• Adam Air Boeing 737-400 crashed Monday after vanishing off air traffic control radar screens between the Indonesian islands of Java and Sulawesi. • Up to 12 people were thought to have survived, with rescue teams racing to the crash site near Polewali in West Sulawesi , some 180 kilometres north of the South Sulawesi provincial capital Makassar. • It was the worst air disaster since Sept. 5, 2005, when a Mandala Airline's Boeing 737-200 crashed shortly after taking off from the North Sumatra's airport, killing 103 people. • Earlier on Friday, a ferry carrying 628 people sank off the Java coast.
Extract+Rewrite Summary	**PG-MMR Summary**
• Plane with 102 people on board crashes. • Three Americans among 102 on board plane in Indonesia. • Rescue team arrives in Indonesia after plane crash. • Plane with 102 crashes in West Sulawesi, killing at least 90. • No word on the fate of Boeing 737-400. • Plane carrying 96 passengers loses contact with Makassar. • Plane crashes in Indonesia , killing at least 90. • Indonesian navy sends two planes to carry bodies of five. • Indonesian plane carrying 102 missing. • Indonesian lawmaker criticises slow deployment of plane. • Hundreds of kilometers plane crash.	• The Adam Air Boeing 737-400 crashed Monday afternoon, but search and rescue teams only discovered the wreckage early Tuesday. • The Indonesian rescue team arrived at the mountainous area in West Sulawesi province where a passenger plane with 102 people onboard crashed into a mountain in Polewali, West Sulawesi province. • Air force rear commander Eddy Suyanto told-Shinta radio station that the plane – operated by local carrier Adam Air – had crashed in a mountainous region in Polewali province on Monday. • There was no word on the fate of the remaining 12 people on board the boeing 737-400.

Table 6: Example system summaries and human-written abstract. The sentences are manually de-tokenized for readability.

percentages of summary n-grams (or entire sentences) appearing in the multi-document input. PG-Original and PG-MMR summaries both show a high degree of extractiveness, and similar findings have been revealed by See et al. (2017). Because PG-MMR relies on a handful of representative source sentences and mutes the rest, it appears to be marginally more extractive than PG-Original. Both systems encourage generating summary sentences by stitching together source sentences, as about 52% and 41% of the summary sentences do not appear in the source, but about 90% the n-grams do. The Extract+Rewrite summaries (§5), generated by rewriting selected source sentences to title-like summary sentences, exhibits a high degree of abstraction, close to that of human abstracts.

Linguistic quality. To assess the linguistic quality of various system summaries, we employ Amazon Mechanical Turk human evaluators to judge the summary quality, including PG-MMR, LexRank, PG-Original, and Extract+Rewrite. A turker is asked to rate each system summary on a scale of 1 (worst) to 5 (best) based on three evaluation criteria: ***informativeness*** (to what extent is the meaning expressed in the ground-truth text preserved in the summary?), ***fluency*** (is the summary grammatical and well-formed?), and ***non-redundancy*** (does the summary successfully avoid repeating information?). Human summaries are used as the ground-truth. The turkers are also asked to provide an overall ranking for the four system summaries. Results are presented in Table 5. We observe that the LexRank summaries are highest-rated on fluency. This is because LexRank is an extractive approach, where summary sentences are directly taken from the input. PG-MMR is rated as the best on both informativeness and non-redundancy. Regarding overall system rankings, PG-MMR summaries are frequently ranked as the 1st- and 2nd-best summaries, outperforming the others.

Example summaries. In Table 6 we present example summaries generated by various systems. PG-Original cannot effectively identify important content from the multi-document input. Extract+Rewrite tends to generate short, title-like sentences that are less informative and carry substantial redundancy. This is because the system is trained on the Gigaword dataset (Rush et al., 2015) where the target summary length is 7 words. PG-MMR generates summaries that effectively condense the important source content.

7 Conclusion

We describe a novel adaptation method to generate abstractive summaries from multi-document inputs. Our method combines an extractive summarization algorithm (MMR) for sentence extraction and a recent abstractive model (PG) for fusing source sentences. The PG-MMR system demonstrates competitive results, outperforming strong extractive and abstractive baselines.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tal Baumel, Matan Eyal, and Michael Elhadad. 2018. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *arXiv preprint arXiv:1801.07704*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J. Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of ACL*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2017. Improving multi-document summarization via text classification. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Giuseppe Carenini and Jackie Chi Kit Cheung. 2008. Extractive vs. NLG-based abstractive summarization of evaluative text: The effect of corpus controversiality. In *Proceedings of the Fifth International Natural Language Generation Conference (INLG)*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of ACL*.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 update summarization task. In *Proceedings of Text Analysis Conference (TAC)*.

Hal Daume III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*.

Giuseppe Di Fabbrizio, Amanda J. Stent, and Robert Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. *Proceedings of the 8th International Natural Language Generation Conference (INLG)*.

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Proceedings of NAACL-HLT*.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the NAACL Workshop on Integer Linear Programming for Natural Langauge Processing*.

Dan Gillick, Benoit Favre, Dilek Hakkani-Tur, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The ICSI/UTD summarization system at TAC 2009. In *Proceedings of TAC*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of Neural Information Processing Systems (NIPS)*.

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

Kai Hong, John M Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*.

Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. 2017. Extractive summarization using multi-task learning with document classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hongyan Jing and Kathleen McKeown. 1999. The decomposition of human-written summary sentences. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of EMNLP*.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract meaning representation for multi-document summarization. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Proceedings of ACL Workshop on Text Summarization Branches Out*.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Wencan Luo and Diane Litman. 2015. Summarizing student responses to reflection prompts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Wencan Luo, Fei Liu, Zitao Liu, and Diane Litman. 2016. Automatic summarization of student course feedback. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*.

Paul Over and James Yen. 2004. An introduction to DUC-2004. *National Institute of Standards and Technology*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Daniele Pighin, Marco Cornolti, Enrique Alfonseca, and Katja Filippova. 2014. Modelling events through memory-based, open-ie patterns for abstractive summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for sentence summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kaiqiang Song, Lin Zhao, and Fei Liu. 2018. Structure-infused copy mechanisms for abstractive summarization. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alex Kuleszaand Ben Taskar. 2012. *Determinantal Point Processes for Machine Learning*. Now Publishers Inc.

Kapil Thadani and Kathleen McKeown. 2013. Sentence compression with joint structural inference. In *Proceedings of CoNLL*.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing and Management*, 43(6):1606–1618.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of ACL*.

Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Dani Yogatama, Fei Liu, and Noah A. Smith. 2015. Extractive summarization by maximizing semantic volume. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management*.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2017. Efficient summarization with read-again and copy mechanism. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jianmin Zhang, Jiwei Tan, and Xiaojun Wan. 2018. Towards a neural network approach to abstractive multi-document summarization. *arXiv preprint arXiv:1804.09010*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Semi-Supervised Learning for Neural Keyphrase Generation

Hai Ye[*] and **Lu Wang**
College of Computer and Information Science
Northeastern University
Boston, MA 02115
`hye.me@outlook.com` `luwang@ccs.neu.edu`

Abstract

We study the problem of generating keyphrases that summarize the key points for a given document. While sequence-to-sequence (seq2seq) models have achieved remarkable performance on this task (Meng et al., 2017), model training often relies on large amounts of labeled data, which is only applicable to resource-rich domains. In this paper, we propose semi-supervised keyphrase generation methods by leveraging both labeled data and large-scale unlabeled samples for learning. Two strategies are proposed. First, unlabeled documents are first tagged with synthetic keyphrases obtained from unsupervised keyphrase *extraction* methods or a self-learning algorithm, and then combined with labeled samples for training. Furthermore, we investigate a multi-task learning framework to jointly learn to generate keyphrases as well as the titles of the articles. Experimental results show that our semi-supervised learning-based methods outperform a state-of-the-art model trained with labeled data only.

1 Introduction

Keyphrase extraction concerns the task of selecting a set of phrases from a document that can indicate the main ideas expressed in the input (Turney, 2000; Hasan and Ng, 2014). It is an essential task for document understanding because accurate identification of keyphrases can be beneficial for a wide range of downstreaming natural language processing and information retrieval applications. For instance, keyphrases can be leveraged to improve text summarization systems (Zhang et al., 2004; Liu et al., 2009a; Wang and Cardie, 2013), facilitate sentiment analysis and opinion mining (Wilson et al., 2005; Berend, 2011), and help with document clustering (Hammouda et al., 2005). Though relatively easy to implement, extract-based approaches fail to generate keyphrases that do not appear in the source document, which are frequently produced by human annotators as shown in Figure 1. Recently, Meng et al. (2017) propose to use a sequence-to-sequence model (Sutskever et al., 2014) with copying mechanism for *keyphrase generation*, which is able to produce phrases that are not in the input documents.

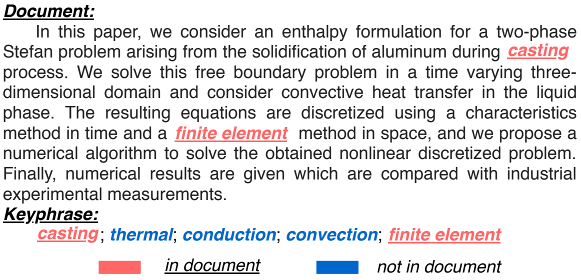

Document:

In this paper, we consider an enthalpy formulation for a two-phase Stefan problem arising from the solidification of aluminum during ***casting*** process. We solve this free boundary problem in a time varying three-dimensional domain and consider convective heat transfer in the liquid phase. The resulting equations are discretized using a characteristics method in time and a ***finite element*** method in space, and we propose a numerical algorithm to solve the obtained nonlinear discretized problem. Finally, numerical results are given which are compared with industrial experimental measurements.

Keyphrase:

casting; ***thermal***; ***conduction***; ***convection***; ***finite element***

🟥 *in document* 🟦 *not in document*

Figure 1: Sample document with labeled keyphrases.

While seq2seq model demonstrates good performance on keyphrase generation (Meng et al., 2017), it heavily relies on massive amounts of labeled data for model training, which is often unavailable for new domains. To overcome this drawback, in this work, *we investigate semi-supervised learning for keyphrase generation, by leveraging abundant unlabeled documents along with limited labeled data*. Intuitively, additional documents, though unlabeled, can provide useful knowledge on universal linguistic features and discourse structure, such as context information for keyphrases and that keyphrases are likely to be noun phrases or main verbs. Furthermore, learning with unlabeled data can also mitigate the overfitting problem, which is often caused by small size of labeled training data, and thus improve model generalizability and enhance keyphrase generation performance on unseen data.

Concretely, two major approaches are proposed for leveraging unlabeled data. For the first method,

---

[*] Work was done while visiting Northeastern University.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4142–4153
Brussels, Belgium, October 31 - November 4, 2018.

unlabeled documents are first tagged with synthetic keyphrases, then mixed with labeled data for model pre-training. Synthetic keyphrases are acquired through existing unsupervised keyphrase *extraction* methods (e.g., TF-IDF or TextRank (Mihalcea and Tarau, 2004)) or a self-learning algorithm. The pre-trained model will further be fine-tuned on the labeled data only. For the second approach, we propose a multi-task learning (MTL) framework[1] by jointly learning the main task of keyphrase generation based on labeled samples, and an auxiliary task of title generation (Rush et al., 2015) on unlabeled documents. Here one encoder is shared among the two tasks. Importantly, we test our proposed methods on a seq2seq framework, however, we believe they can be easily adapted to other encoder-decoder-based systems.

Extensive experiments are conducted in scientific paper domain. Results on five different datasets show that all of our semi-supervised learning-based models can uniformly significantly outperform a state-of-the-art model (Meng et al., 2017) as well as several competitive unsupervised and supervised keyphrase extraction algorithms based on F_1 and recall scores. We further carry out a cross-domain study on generating keyphrases for news articles, where our models yield better F_1 than a model trained on labeled data only. Finally, we also show that training with unlabeled samples can further produce performance gain even when a large amount of labeled data is available.

2 Related Work

Keyphrase Extraction and Generation. Early work mostly focuses on the keyphrase extraction task, and a two-step strategy is typically designed. Specifically, a large pool of candidate phrases are first extracted according to predefined syntactic templates (Mihalcea and Tarau, 2004; Wan and Xiao, 2008; Liu et al., 2009b, 2011) or their estimated importance scores (Hulth, 2003). In the second step, re-ranking is applied to yield the final keyphrases, based on supervised learning (Frank et al., 1999; Witten et al., 1999; Hulth, 2003; Lopez and Romary, 2010; Kim and Kan, 2009), unsupervised graph algorithms (Mihalcea and Tarau, 2004; Wan and Xiao, 2008; Bougouin et al., 2013), or topic modelings (Liu et al., 2009c, 2010). Keyphrase generation is studied in more recent work. For instance, Liu et al. (2011) propose to use statistic machine translation model to learn word-alignments between documents and keyphrases, enabling the model to generate keyphrases which do not appear in the input. Meng et al. (2017) train seq2seq-based generation models (Sutskever et al., 2014) on large-scale labeled corpora, which may not be applicable to a new domain with minimal human labels.

Neural Semi-supervised Learning. As mentioned above, though significant success has been achieved by seq2seq model in many NLP tasks (Luong et al., 2015; See et al., 2017; Dong and Lapata, 2016; Wang and Ling, 2016; Ye et al., 2018), they often rely on large amounts of labeled data, which are expensive to get. In order to mitigate the problem, semi-supervised learning has been investigated to incorporate unlabeled data for modeling training (Dai and Le, 2015; Ramachandran et al., 2017). For example, neural machine translation community has studied the usage of source-side or target-side monolingual data to improve translation quality (Gülçehre et al., 2015), where generating synthetic data (Sennrich et al., 2016; Zhang and Zong, 2016), multi-task learning (Zhang and Zong, 2016), and autoencoder-based methods (Cheng et al., 2016) are shown to be effective. Multi-task learning is also examined for sequence labeling tasks (Rei, 2017; Liu et al., 2018). In our paper, we study semi-supervised learning for keyphrase generation based on seq2seq models, which has not been explored before. Besides, we focus on leveraging source-side unlabeled articles to enhance performance with synthetic keyphrase construction or multi-task learning.

3 Neural Keyphrase Generation Model

In this section, we describe the neural keyphrase generation model built on a sequence-to-sequence model (Sutskever et al., 2014) as illustrated in Figure 2. We denote the input source document as a sequence $\mathbf{x} = x_1 \cdots x_{|\mathbf{x}|}$ and its corresponding keyphrase set as $a = \{a_i\}_{i=1}^{|a|}$, with a_i as one keyphrase.

Keyphrase Sequence Formulation. Different from the setup by Meng et al. (2017), where input article is paired with each keyphrase a_i to consist a training sample, we concatenate the keyphrases in a into a keyphrase sequence $\mathbf{y} =$

---

[1] We use "semi-supervised learning" as a broad term to refer to the two methods proposed in this paper.

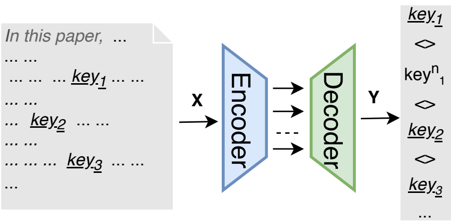

Figure 2: Neural keyphrase generation model built on sequence-to-sequence framework. Input is the document, and output is the keyphrase sequence consisting of phrases present (key_i) or absent (key_i^n) in the input.

$a_1 \diamondsuit a_2 \diamondsuit \cdots \diamondsuit a_{|a|}$, where $\diamondsuit$ is a segmenter to separate the keyphrases[2]. With this setup, the seq2seq model is capable to generate all possible keyphrases in one sequence as well as capture the contextual information between the keyphrases from the same sequence.

Seq2Seq Attentional Model. With source document $\mathbf{x}$ and its keyphrase sequence $\mathbf{y}$, an encoder encodes $\mathbf{x}$ into context vectors, from which a decoder then generates $\mathbf{y}$. We set the encoder as one-layer bi-directional LSTM model and the decoder as another one-layer LSTM model (Hochreiter and Schmidhuber, 1997). The probability of generating target sequence $p(\mathbf{y}|\mathbf{x})$ is formulated as:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t|\mathbf{y}_{<t}, \mathbf{x}) \tag{1}$$

where $\mathbf{y}_{<t} = y_1 \cdots y_{t-1}$.

Let $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$ denote the hidden state vector in the encoder at time step t, which is the concatenation of forward hidden vector $\overrightarrow{\mathbf{h}}_t$ and backward hidden vector $\overleftarrow{\mathbf{h}}_t$. Specifically, $\overrightarrow{\mathbf{h}}_t = \mathbf{f}_{\mathrm{LSTM_e}}(x_t, \overrightarrow{\mathbf{h}}_{t-1})$ and $\overleftarrow{\mathbf{h}}_t = \mathbf{f}_{\mathrm{LSTM_e}}(x_t, \overleftarrow{\mathbf{h}}_{t+1})$, where $\mathbf{f}_{\mathrm{LSTM_e}}$ is an LSTM unit in encoder.

Decoder hidden state is calculated as $\mathbf{s}_t = \mathbf{f}_{\mathrm{LSTM_d}}(y_{t-1}, \mathbf{s}_{t-1})$, where $\mathbf{f}_{\mathrm{LSTM_d}}$ is an LSTM unit in decoder. We apply global attention mechanism (Luong et al., 2015) to calculate the context vector:

$$\begin{aligned} \mathbf{c}_t &= \sum_{i=1}^{|\mathbf{x}|} \alpha_{t,i} \mathbf{h}_i \\ \alpha_{t,i} &= \frac{\exp(\mathbf{W}_{\mathrm{att}}[\mathbf{s}_t; \mathbf{h}_i])}{\sum_{k=1}^{|x|} \exp(\mathbf{W}_{\mathrm{att}}[\mathbf{s}_t \mathbf{h}_k])} \end{aligned} \tag{2}$$

Algorithm 1 Keyphrase Ranking

```
Input: Generated top R keyphrase sequences $\mathcal{S} = [\mathbf{y}_1, \cdots, \mathbf{y}_{\mathrm{R}}]$ ranked with generation possibility from high
    to low with beam search
Output: Ranked keyphrase set $\mathcal{A}$ with importance from
    high to low
    function KEY-RANK($\mathcal{S}$)
        $\mathcal{A} \leftarrow$ list()                    ▷ set $\mathcal{A}$ as a empty list.
        $\mathcal{Q} \leftarrow$ dict()      ▷ to skip keyword that is already in $\mathcal{A}$.
        for $\mathbf{y}_i \in \mathcal{S}$ do
            $y_i \leftarrow \mathbf{y}_i$.split("$\diamondsuit$");               ▷ split $\mathbf{y}_i$ by "$\diamondsuit$"
            for $a \in y_i$ do
                ▷ if $a$ has not been merged in $\mathcal{A}$, then keep it.
                if not $\mathcal{Q}$.has_key($a$) then
                    $\mathcal{A}$.append($a$)
                    $\mathcal{Q}$.update(dict({$a$ : ""}))
```

where $\alpha_{t,i}$ is the attention weight; $\mathbf{W}_{\mathrm{att}}$ contains learnable parameters. In this paper, we omit the bias variables to save space.

The probability to predict y_t in the decoder at time step t is factorized as:

$$\begin{aligned} p_{\mathrm{vocab}}(y_t|\mathbf{y}_{<t}, \mathbf{c}_t) = \mathbf{f}_{\mathrm{softmax}}(\mathbf{W}_{\mathrm{d_1}} \cdot \\ \tanh(\mathbf{W}_{\mathrm{d_2}}[\mathbf{s}_t; \mathbf{c}_t])) \end{aligned} \tag{3}$$

where $\mathbf{f}_{\mathrm{softmax}}$ is the softmax function and $\mathbf{W}_{\mathrm{d_1}}$ and $\mathbf{W}_{\mathrm{d_2}}$ are learnable parameters.

Pointer-generator Network. Similar to Meng et al. (2017), we utilize copying mechanism via pointer-generator network (See et al., 2017) to allow the decoder to directly copy words from input document, thus mitigating out-of-vocabulary (OOV) problem. At time step t, the generation probability p_{gen} is calculated as:

$$p_{\mathrm{gen}} = \mathbf{f}_{\mathrm{sigmoid}}(\mathbf{W}_{\mathrm{c}}\mathbf{c}_t + \mathbf{W}_{\mathrm{s}}\mathbf{s}_t + \mathbf{W}_{\mathrm{y}}y_t) \tag{4}$$

where $\mathbf{f}_{\mathrm{sigmoid}}$ is a sigmoid function; $\mathbf{W}_{\mathrm{c}}$, $\mathbf{W}_{\mathrm{s}}$ and $\mathbf{W}_{\mathrm{y}}$ are learnable parameters. p_{gen} plays a role of switcher to choose to generate a word from a fixed vocabulary with probability p_{vocab} or directly copy a word from the source document with the attention distribution α_t. With a combination of a fixed vocabulary and the extended source document vocabulary, the probability to predict y_t is:

$$p(y_t) = p_{\mathrm{gen}} p_{\mathrm{vocab}}(y_t|\mathbf{y}_{<t}, \mathbf{c}_t) + (1 - p_{\mathrm{gen}}) \sum_{i: y_i = y_t} \alpha_{t,i} \tag{5}$$

where if y_t does not appear in the fixed vocabulary, then the first term will be zero; and if y_t is outside source document, the second term will be zero.

---

[2] We concatenate keyphrases following the original keyphrase order in the corpora, and we set $\diamondsuit$ as ";" in our implementation. The effect of keyphrase ordering will be studied in the future work.

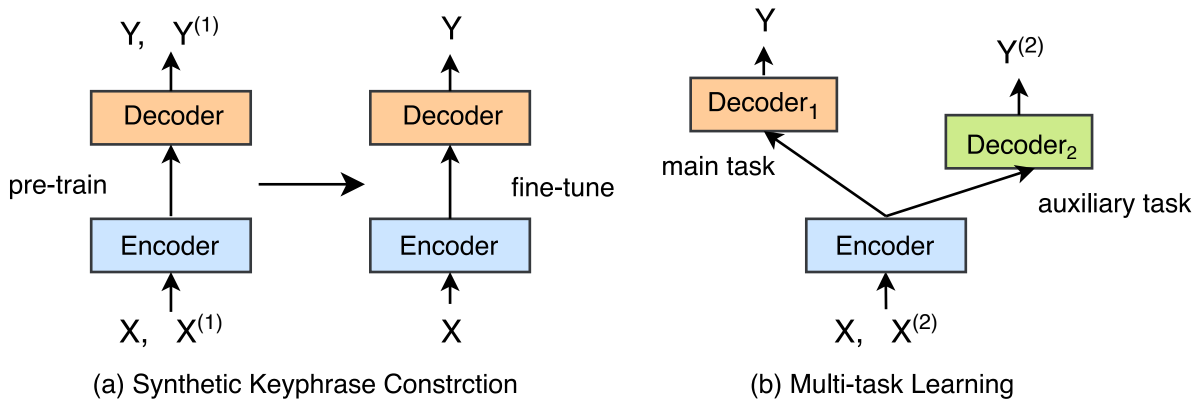

Figure 3: Our two semi-supervised learning methods which are based on (a) synthetic keyphrase construction, and (b) multi-task learning. (X, Y) represents labeled sample. $\mathrm{X}^{(1)}$ and $\mathrm{X}^{(2)}$ denotes unlabeled documents. $\mathrm{Y}^{(1)}$ refers to synthetic keyphrases of $\mathrm{X}^{(1)}$ and $\mathrm{Y}^{(2)}$ means the title of $\mathrm{X}^{(2)}$. For method of synthetic keyphrase construction, model will be pre-trained on the mixture of gold-standards and synthetic data, then fine-tuned on labeled data. For multi-task learning, model parameters of main task and auxiliary task will be jointly updated. Encoder parameters of the two tasks are shared.

Supervised Learning. With a labeled dataset $\mathcal{D}_p = \{\mathbf{x}_{(i)}, \mathbf{y}_{(i)}\}_{i=1}^N$, the loss function of seq2seq model is as follows:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N} \log p(\mathbf{y}_{(i)}|\mathbf{x}_{(i)}; \theta) \tag{6}$$

where θ contains all model parameters.

Keyphrase Inference and Ranking Strategy. Beam search is utilized for decoding, and the top R keyphrase sequences are leveraged for producing the final keyphrases. Here we use a beam size of 50, and R as 50. We propose a ranking strategy to collect the final set of keyphrases. Concretely, in sequence we collect unique keyphrases from the top ranked beams to lower ranked beams, and keyphrases in the same sequence are ordered as in the generation process. Intuitively, higher ranked sequences are likely of better quality. As for keyphrases in the same sequence, we find that more salient keyphrases are usually generated first. The ranking method is presented in Algorithm 1.

4 Semi-Supervised Learning for Keyphrase Generation

As illustrated in Figure 3, two methods are proposed to leverage abundant unlabeled data. The first is to provide synthetic keyphrases using unsupervised keyphrase extraction methods or self-learning algorithm, then mixed with labeled data for model training, which is described in Section 4.1. Furthermore, we introduce multi-task learning that jointly generates keyphrases and the title of the document (Section 4.2). We denote the large-scale unlabeled documents as $\mathcal{D}_u = \{\mathbf{x}'_{(i)}\}_{i=1}^M$ and labeled data as $\mathcal{D}_p = \{\mathbf{x}_{(i)}, \mathbf{y}_{(i)}\}_{i=1}^N$, where $M \gg N$.

4.1 Synthetic Keyphrase Construction

The first proposed technique is to construct synthetic labeled data by assigning keyphrases for unlabeled documents, and then mix the synthetic data with human labeled data for modeling training. Intuitively, adding training samples with synthetic keyphrases has two potentially benefits: (1) the encoder is exposed to more documents in training, and (2) the decoder also benefit from additional information from identifying contextual information for keyphrases. We propose and compare two methods to extract synthetic keyphrases.

Unsupervised Learning Methods. Unsupervised learning methods on keyphrase extraction have been long studied in previous work (Mihalcea and Tarau, 2004; Wan and Xiao, 2008). Here we select two effective and widely used methods to select keyphrases on unlabeled dataset $\mathcal{D}_u$, which are TF-IDF and TextRank (Mihalcea and Tarau, 2004). We combine the two methods into a hybrid approach, in which we first adopt the two methods to separately select top K keyphrases from the document, we then take the union with duplicate removal. To construct the keyphrase se-

Algorithm 2 Training Procedures for Synthetic Keyphrase Construction

```
Input: $\mathcal{D}_p$, $\mathcal{D}_s$, $\theta$
Output: $\theta$
  function PRE-TRAIN($\mathcal{D}_p$, $\mathcal{D}_s$, $\theta$)
      $\mathcal{D}_{p+s} \leftarrow \mathcal{D}_p \cup \mathcal{D}_s$
      Shuffle $\mathcal{D}_{p+s}$ randomly
      Update $\theta$ on $\mathcal{D}_{p+s}$ until converge
  function FINE-TUNE($\mathcal{D}_p$, $\theta$)
      Set $\theta$ as the best parameters from PRE-TRAIN
      Update $\theta$ on $\mathcal{D}_p$ until converge
```

quence, we concatenate the terms from TF-IDF and then from TextRank, following the corresponding ranking order. We set K as 5 in our experiments.

Self-learning Algorithm. Inspired by prior work (Zhang and Zong, 2016; Sennrich et al., 2016), we adopt self-learning algorithm to boost training data. Concretely, we first build a baseline model by training the seq2seq model on the labeled corpus $\mathcal{D}_p$. Then the trained baseline model is utilized to generate synthetic keyphrase sequence $\mathbf{y}'$ given a unlabeled document $\mathbf{x}'$. We adopt beam search to generate the synthetic keyphrase sequences and beam size is set as 10. The top one beam is selected.

Training Procedure. After the synthetic data $\mathcal{D}_s = \{\mathbf{x}'_{(i)}, \mathbf{y}'_{(i)}\}_{i=1}^{M}$ is obtained by either of the aforementioned methods, we mix labeled data $\mathcal{D}_p$ with $\mathcal{D}_s$ to train the seq2seq model. As described in Algorithm 2, we combine $\mathcal{D}_p$ with $\mathcal{D}_s$ into $\mathcal{D}_{p+s}$ and shuffle $\mathcal{D}_{p+s}$ randomly, then we pre-train the model on $\mathcal{D}_{p+s}$, in which no network parameters are frozen during the training process. The best performing model is selected based on validation set, then fine-tuned on $\mathcal{D}_p$ until converge.

4.2 Multi-task Learning with Auxiliary Task

The second approach to leverage unlabeled documents is to employ a multi-task learning framework which combines the main task of keyphrase generation with an auxiliary task through parameter sharing strategy. Similar to the model structure in Zhang and Zong (2016), our main task and the auxiliary task share an encoder network but have different decoders. Multi-task learning will benefit from the source-side information to improve the model generality of encoder.

In most domains such as scientific papers or news articles, a document usually contains a title that summarizes the core topics or ideas, with a similar spirit as keyphrases. We thus choose title generation as auxiliary task, which has been studied as a summarization problem (Rush et al., 2015; Colmenares et al., 2015). Let $\mathcal{D}'_u = \{\mathbf{x}'_{(i)}, \mathbf{q}_{(i)}\}_{i=1}^{M}$ denote the dataset which is assigned with titles for unlabeled data $\mathcal{D}_u$, the loss function of multi-task learning is factorized as:

Dataset	TRAIN	VALID
Small-scale		
Document-Keyphrase	40,000	5,000
Document-SyntheticKeyphrase	400,000	N/A
Document-Title	400,000	15,000
Large-scale		
Document-Keyphrase	130,000	5,000

Avg. #Tokens in Train	LABELED	SYN.	MTL
Small-scale			
Document	176.3	175.9	165.5
Keyphrase Sequence	23.3	23.5	N/A
Title	N/A	N/A	10.4

Table 1: Statistics of datasets used in our experiments.

$$\mathcal{L}(\theta^e, \theta_1^d, \theta_2^d) = -\sum_{i=1}^{N} \log p(\mathbf{y}_{(i)} | \mathbf{x}_{(i)}; \theta^e, \theta_1^d) - \sum_{i=1}^{M} \log p(\mathbf{q}_{(i)} | \mathbf{x}'_{(i)}; \theta^e, \theta_2^d) \quad (7)$$

where θ^e indicates encoder parameters; θ_1^d and θ_2^d are the decoder parameters.

Training Procedure. We adopt a simple alternating training strategy to switch training between the main task and the auxiliary task. Specifically, we first estimate parameters on auxiliary task with $\mathcal{D}'_u$ for one epoch, then train model on the main task with $\mathcal{D}_p$ (labeled dataset) for T epochs. We follow this training procedure for several times until the model of our main task converges. We set T as 3.

5 Experiments

5.1 Datasets

Our major experiments are conducted on scientific articles which have been studied in previous work (Hulth, 2003; Nguyen and Kan, 2007; Meng et al., 2017). We use the dataset from Meng et al. (2017) which is collected from various online digital libraries, e.g. ScienceDirect, ACM Digital Library, Wiley, and other portals.

As indicated in Table 1, we construct a relatively small-scale labeled dataset with 40K document-keyphrase[3] pairs, and a large-scale

---

[3] Here keyphrase refers to the keyphrase sequence.

Model	INSPEC		KRAPIVIN		NUS		SEMEVAL		KP20K	
	F_1@5	F_1@10	F_1@5	F_1@10	F_1@5	F_1@10	F_1@5	F_1@10	F_1@5	F_1@10
Comparisons										
TF-IDF	0.223	$0.304^\dagger$	0.113	0.143	0.139	0.181	0.120	$0.184^\dagger$	0.105	0.130
TEXTRANK	0.229	0.275	0.173	0.147	0.195	0.190	0.172	0.181	0.181	0.150
SINGLERANK	0.214	0.297	0.096	0.137	0.145	0.169	0.132	0.169	0.099	0.124
EXPANDRANK	0.211	0.295	0.096	0.136	0.137	0.162	0.135	0.163	N/A	N/A
MAUI	0.041	0.033	0.243	$0.208^\dagger$	0.249	$0.261^\dagger$	0.045	0.039	0.265	$0.227^\dagger$
KEA	0.109	0.129	0.120	0.131	0.068	0.081	0.027	0.027	0.180	0.163
SEQ2SEQ-COPY	0.269	0.234	0.274	0.207	0.345	0.282	0.278	0.226	0.291	0.215
Semi-supervised										
SYN.UNSUPER.	**0.326**$*$	**0.334**$*$	0.283	0.239$*$	**0.356**	0.317$*$	0.306	**0.294**$*$	0.300$*$	**0.245**$*$
SYN.SELF-LEARN.	0.310$*$	0.301$*$	0.289	0.236$*$	0.339	0.305	0.295	0.282$*$	0.301$*$	0.240$*$
MULTI-TASK	**0.326**$*$	0.309$*$	**0.296**	**0.240**$*$	0.354	**0.320**$*$	**0.322**	0.289$*$	**0.308**$*$	0.243$*$

Table 2: Results of *present* keyphrase generation with metrics F_1@5 and F_1@10. $*$ marks numbers that are significantly better than SEQ2SEQ-COPY ($p < 0.01$, F-test). Due to the high time perplexity, no result is reported by ExpandRank on KP20K, as done in Meng et al. (2017).

dataset of 400K unlabeled documents. Each document contains an abstract and a title of the paper. In multi-task learning setting, the auxiliary task is to generate the title from the abstract. For the validation set, we collect 5K document-keyphrase pairs for the process of pre-training and fine-tuning for methods based on synthetic data construction. For multi-task learning, we use the same 5K document-keyphrase pairs for the main task training, another 15K document-title pairs for the auxiliary task. We further conduct experiments on a 130K large-scale labeled dataset, which includes the small-scale labeled data.

Similar to Meng et al. (2017), we test our model on five widely used public datasets from the scientific domain: INSPEC (Hulth, 2003), NUS (Nguyen and Kan, 2007), KRAPIVIN (Krapivin et al., 2009), SEMEVAL-2010 (Kim et al., 2010) and KP20K (Meng et al., 2017).

5.2 Experimental Settings

Data preprocessing is implemented as in (Meng et al., 2017). The texts are first tokenized by NLTK (Bird and Loper, 2004) and lowercased, then the numbers are replaced with <digit>. We set maximal length of source text as 200, 40 for target text. Encoder and decoder both have a vocabulary size of 50K. The word embedding size is set to 128. Embeddings are randomly initialized and learned during training. The size of hidden vector is 512. Dropout rate is set as 0.3. Maximal gradient normalization is 2. Adagrad (Duchi et al., 2011) is adopted to train the model with learning rate of 0.15 and the initial accumulator rate is 0.1.

For synthetic data construction, we first use batch size of 64 for model pre-training and then reduce to 32 for model fine-tuning. For both training stages, after 8 epochs, learning rate be decreased with a rate of 0.5. For multi-task learning, batch size is set to 32 and learning rate is reduced to half after 20 training epochs. To build baseline seq2seq model, we set batch size as 32 and decrease learning rate after 8 epochs. For self-learning algorithm, beam size is set to 10 to generate target sequences for unlabeled data and the top one is retained.

5.3 Comparisons with Baselines

Evaluation Metrics. Following (Liu et al., 2011; Meng et al., 2017), we adopt top-N macro-averaged *precision*, *recall* and *F-measure* (F_1) for model evaluation. *Precision* means how many top-N extracted keywords are correct and *recall* means how many target keyphrases are extracted in top-N candidates. Porter Stemmer is applied before comparisons.

Baselines. We train a baseline seq2seq model on the small-scale labeled dataset. We further compare with four unsupervised learning methods: TF-IDF, TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), ExpandRank (Wan and Xiao, 2008), and two supervised learning methods of Maui (Medelyan et al., 2009) and KEA (Witten et al., 1999). We follow Meng et al. (2017) for baselines setups.

Results. Here we show results for *present* and *absent* keyphrase generation separately[4]. From the results of present keyphrase generation as shown in Table 2, although trained on small-scale

---

[4] Recall that *present* means the keyphrase appears in the document, otherwise, it is *absent*.

		Semi-supervised		
Dataset	SEQ.	SYN.UN.	SYN.SELF.	MULTI.
INSPEC	0.012	0.018	0.013	**0.022**
KRAPIVIN	0.01	0.008	0.018	**0.021**
NUS	0.002	0.011	0.003	**0.013**
SEMEVAL	0.001	**0.008**	0.003	0.006
KP20K	0.01	0.016	0.013	**0.021**

Table 3: Results of *absent* kephrase generation based on Recall@10.

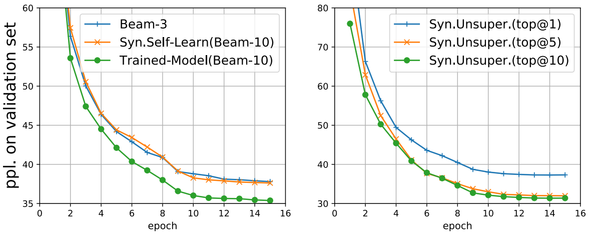

Figure 4: *Pre-training* curves with perplexity on validation set with various options for synthetic keyphrases construction. Left is for options for self-learning algorithm and right is for unsupervised learning methods.

labeled corpora, our baseline seq2seq model with copying mechanism still achieves better F_1@5 scores, compared to other baselines. This demonstrates that a baseline seq2seq model has learned the mapping patterns from source text to target keyphrases to some degree. However, small-scale labeled data still hinders the potential of seq2seq model according to the poor performance of F_1@10. By leveraging large-scale unlabeled data, our semi-supervised learning methods achieve siginifcant improvement over seq2seq baseline, as well as exhibit the best performance in both F_1@5 and F_1@10 on almost all datasets.

We further compare seq2seq based models on the task of generating keyphrases beyond input article vocabulary. Illustrated by Table 3, semi-supervised learning significantly improves the absent generation performance, compared to the baseline seq2seq. Among our models, the multi-task learning method is more effective at generating absent keyphrases than the two methods by leveraging synthetic data. The main reason may lie in that synthetic keyphrases potentially introduce noisy annotations, while the decoder in multi-task learning setting focuses on learning from gold-standard keyphrases. We can also see that the overall performances by all models are low, due to the intrinsic difficulty of absent keyphrase generation (Meng et al., 2017). Moreover, we only employ 40K labeled data for training, which is rather limited for training. Besides, we believe better evaluation methods should be used instead of exact match, e.g., by considering paraphrases. This will be studied in the future work.

5.4 Effect of Synthetic Keyphrase Quality

In this section, we conduct experiments to further study the effect of synthetic keyphrase quality on model performance. Two sets of experiments are undertaken, one for evaluating unsupervised learning and one for self-learning algorithm.

For self-learning algorithm, we further generate synthetic keyphrases using following options:

- Beam-size-3: Based on our baseline model trained with labeled data, we use beam search with a smaller beam size of 3 to generate synthetic data[5].

- Trained-model: We adopt the model which has been trained with self-learning algorithm on 40K labeled data and 400K unlabeled data, to generate the top one keyphrase sequence with beam size of 10.

For unsupervised learning method, we originally merge top-K (K = 5) keyphrases from TF-IDF and TextRank, here we use options where K is set as 1 or 10 to extract keyphrases:

- Top@1: Using TF-IDF or TextRank, we only keep top 1 extraction from each, then take the union of the two.

- Top@10: Similarly, we keep top 10 extracted terms from TF-IDF or TextRank, then take the union.

As illustrated in Figure 4, when models are pre-trained with synthetic keyphrases of better quality, results by "Trained-model" consistently produce better performance (i.e., lower perplexity). Similar phenomenon can be observed when "top@5" and "top@10" are applied for extraction in unsupervised learning setting. Furthermore, after models are pre-trained and then fine-tuned, the results in Figure 5 show that the difference among baselines becomes insignificant—the quality of synthetic keyphrases have limited effect on final scores. The reason might be that though synthetic

---

[5] We also experiment with greedy search (i.e. beam size of 1), however, unknown words are frequently generated.

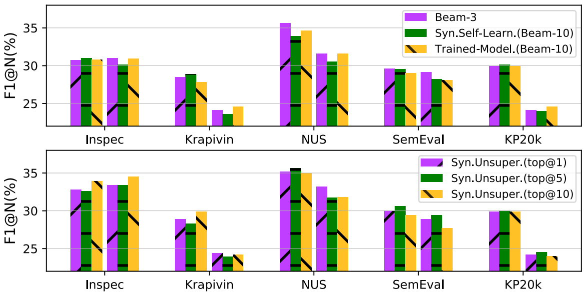

Figure 5: Effect of synthetic data quality on *present* keyphrase generation (models are pre-trained and fine-tuned) based on $F_1@5$ (left three columns) and $F_1@10$ (right three columns), on five datasets. The upper is for self-learning algorithm and the bottom is for unsupervised learning method.

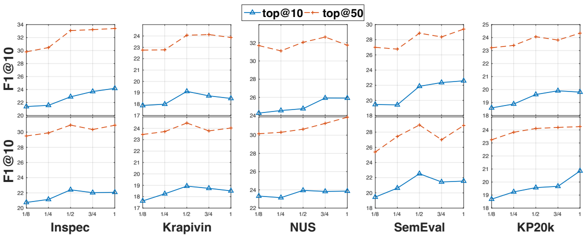

Figure 6: Effect of various amounts of unlabeled data for training on *present* keyphrase generation with $F_1@10$. Upper is for synthetic data construction method with unsupervised learning. Bottom is for multi-task learning algorithm.

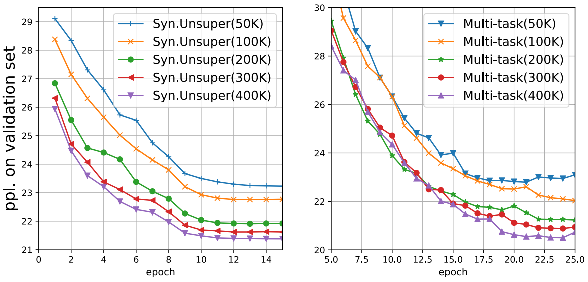

Figure 7: Perplexity on validation set with varying amounts of unlabeled data for training. Left is for fine-tuning procedure based on models trained with synthetic data constructed with unsupervised learning. Right is for multi-task learning procedure with performance on the main task.

Model	F_1	Model	F_1
Our Models		***Unsupervised***	
SEQ2SEQ	0.056	TF-IDF	0.270
SYN.UNSUPER.	0.083	TEXTRANK	0.097
SYN.SELF-LEARN.	0.065	SINGLERANK	0.256
MULTI-TASK	**0.109**	EXPANDRANK	0.269

Table 4: Results of keyphrase generation for news from DUC dataset with F_1. Results of unsupervised learning methods are adopted from Hasan and Ng (2010).

keyphrases potentially introduce noisy information for decoder training, the encoder is still well trained. In addition, after fine-tuning on labeled data, the decoder acquires additional knowledge, thus leading to better performance and minimal difference among the options.

5.5 Effect of Amount of Unlabeled Data

In this section, we further evaluate whether varying the amount of unlabeled data will affect model performance. We conduct experiments based on methods of synthetic data construction with unsupervised learning and multi-task learning. We further carry out experiments with randomly selected $50K(1/8)$, $100K(1/4)$, $200K(1/2)$ and $300K(3/4)$ unlabeled documents from the pool of $400K$ unlabeled data. After models being trained, we adopt beam search to generate keyphrase sequences with beam size of 50. We keep top N keyphrase sequences to yield the final keyphrases using Algorithm 1. $F_1@10$ is adopted to illustrate the model performances. N is set as 10 or 50.

The *present* keyphrase generation results are shown in Figure 6, from which we can see that when increasing the amount of unlabeled data, model performance is further improved. This is because additional unlabeled data can provide with more evidence on linguistic or context features and thus make the model, especially the encoder, have better generalizability. This finding echoes with the training procedure illustrated in Figure 7, where more unlabeled data uniformly leads to better performance. Therefore, we believe that leveraging more unlabeled data for model training can boost model performance.

5.6 A Pilot Study for Cross-Domain Test

Up to now, we have demonstrated the effectiveness of leveraging unlabeled data for in-domain experiments, but is it still effective when being tested on a different domain? We thus carry out a pilot cross-domain test on news articles. The widely used DUC dataset (Wan and Xiao, 2008) is utilized, consisting of 308 articles with 2,048 labeled keyphrases.

The experimental results are shown in Table 4 which indicate that: 1) though trained on scientific papers, our models still have the ability to gener-

Model	**INSPEC**		**KRAPIVIN**		**NUS**		**SEMEVAL**		**KP20K**	
	$F_1@5$	**$F_1@10$**	**$F_1@5$**	**$F_1@10$**	**$F_1@5$**	**$F_1@10$**	**$F_1@5$**	**$F_1@10$**	**$F_1@5$**	**$F_1@10$**
SEQ2SEQ-COPY	**0.34**	0.329	0.308	0.251	0.36	0.327	0.301	0.285	0.318	0.251
Semi-supervised										
SYN.UNSUPER.	0.338	**0.34**	0.316	**0.255**	**0.365**	0.335	**0.337**	0.308	0.322	0.261$*$
SYN.SELF-LEARN.	0.33	0.326	0.304	**0.255**	0.359	**0.336**	0.304	0.304	0.321	0.263$*$
MULTI-TASK	0.328	0.318	**0.323**	0.254	**0.365**	0.326	0.319	**0.312**	**0.328$*$**	**0.264$*$**

Table 5: Results of *present* keyphrase generation on large-scale labeled data with $F_1@5$ and $F_1@10$. $*$ indicates significant better performance than SEQ2SEQ-COPY with $p < 0.01$ (F-test).

ate keyphrases for news articles, illustrating that our models have learned some universal features between the two domains; and 2) semi-supervised learning by leveraging unlabeled data improves the generation performances more, indicating that our proposed method is reasonably effective when being tested on cross-domain data. Though unsupervised methods are still superior, for future work, we can leverage unlabeled out-of-domain corpora to improve cross-domain keyphrase generation performance, which could be a promising direction for domain adaption or transfer learning.

			Semi-supervised	
Dataset	SEQ.	SYN.UN.	SYN.SELF.	MULTI.
INSPEC	0.021	0.024	0.032	**0.033**
KRAPIVIN	0.02	0.031	0.043	**0.047**
NUS	0.009	0.026	0.024	**0.036**
SEMEVAL	0.011	0.014	0.015	**0.02**
KP20K	0.021	0.034	0.039	**0.046**

Table 6: Performance on *absent* keyphrase generation by Recall@10 with large-scale labeled training data.

5.7 Training on Large-scale Labeled Data

Finally, it would be interesting to study whether unlabeled data can still improve performance when the model is trained on a larger scaled labeled data. We conduct experiments on a larger labeled dataset with 130K pairs, along with the 400K unlabeled data. Here the baseline seq2seq model is built on the 130K dataset.

From the *present* keyphrase generation results in Table 5, it can be seen that unlabeled data is still helpful for model training on a large-scale labeled dataset. This implies that we can also leverage unlabeled data to enhance generation performance even in a resource-rich setting. Referring to the *absent* keyphrase generation results shown in Table 6, semi-supervised learning also boosts the scores. From Table 6, training on large-scale labeled data, *absent* generation is significantly improved, compared to being trained on a small-scale labeled data (see Table 3).

6 Conclusion and Future Work

In this paper, we presented a semi-supervised learning framework that leverages unlabeled data for keyphrase generation built upon seq2seq models. We introduced synthetic keyphrases construction algorithm and multi-task learning to effectively leverage abundant unlabeled documents. Extensive experiments demonstrated the effectiveness of our methods, even in scenario where large-scale labeled data is available.

For future work, we will 1) leverage unlabeled data to study domain adaptation or transfer learning for keyphrase generation; and 2) investigate novel models to improve *absent* keyphrase generation when limited labeled data is available based on semi-supervised learning.

Acknowledgements

This research is based upon work supported in part by National Science Foundation through Grants IIS-1566382 and IIS-1813341, and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract # FA8650-17-C-9116. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. We thank three anonymous reviewers for their insightful suggestions on various aspects of this work.

References

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*, pages 1162–1170.

Steven Bird and Edward Loper. 2004. NLTK: the natural language toolkit. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain, July 21-26, 2004 - Poster and Demonstration*.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 543–551.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Carlos A. Colmenares, Marina Litvak, Amin Mantrach, and Fabrizio Silvestri. 2015. HEADS: headline generation as sequence prediction using an abstract feature-rich space. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 133–142.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3079–3087.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 668–673.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535.

Khaled M Hammouda, Diego N Matute, and Mohamed S Kamel. 2005. Corephrase: Keyphrase extraction for document clustering. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 265–274. Springer.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art. In *COLING 2010, 23rd International Conference on Computational Linguistics, Posters Volume, 23-27 August 2010, Beijing, China*, pages 365–373.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1262–1273.

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

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2003, Sapporo, Japan, July 11-12, 2003*.

Su Nam Kim and Min-Yen Kan. 2009. Re-examining automatic keyphrase extraction approaches in scientific articles. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications, MWE@IJCNLP 2009, Singapore, August 6, 2009*, pages 9–16.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, pages 21–26.

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction. Technical report, University of Trento.

Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. 2009a. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 620–628. Association for Computational Linguistics.

Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. 2009b. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA*, pages 620–628.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning, CoNLL 2011, Portland, Oregon, USA, June 23-24, 2011*, pages 135–144.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 366–376.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009c. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 257–266.

Patrice Lopez and Laurent Romary. 2010. HUMB: automatic key term extraction from scientific articles in GROBID. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, pages 248–251.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1318–1327.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 582–592.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 404–411.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers, 10th International Conference on Asian Digital Libraries, ICADL 2007, Hanoi, Vietnam, December 10-13, 2007, Proceedings*, pages 317–326.

Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 383–391.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2121–2130.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 379–389.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2(4):303–336.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 855–860.

Lu Wang and Claire Cardie. 2013. Domain-Independent Abstract Generation for Focused Meeting Summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1395–1405, Sofia, Bulgaria. Association for Computational Linguistics.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 47–57.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM conference on Digital Libraries, August 11-14, 1999, Berkeley, CA, USA*, pages 254–255.

Hai Ye, Xin Jiang, Zhunchen Luo, and Wenhan Chao. 2018. Interpretable charge predictions for criminal cases: Learning to generate court views from fact descriptions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1854–1864.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1535–1545.

Yongzheng Zhang, A. Nur Zincir-Heywood, and Evangelos E. Milios. 2004. World wide web site summarization. *Web Intelligence and Agent Systems*, 2(1):39–53.

MSMO: Multimodal Summarization with Multimodal Output

Junnan Zhu[1,2], **Haoran Li**[1,2], **Tianshang Liu**[1,2], **Yu Zhou**[1,2], **Jiajun Zhang**[1,2] and **Chengqing Zong**[1,2,3]
[1] National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
`{junnan.zhu, yzhou, jjzhang, cqzong}@nlpr.ia.ac.cn`

Abstract

Multimodal summarization has drawn much attention due to the rapid growth of multimedia data. The output of the current multimodal summarization systems is usually represented in texts. However, we have found through experiments that multimodal output can significantly improve user satisfaction for informativeness of summaries. In this paper, we propose a novel task, multimodal summarization with multimodal output (MSMO). To handle this task, we first collect a large-scale dataset for MSMO research. We then propose a multimodal attention model to jointly generate text and select the most relevant image from the multimodal input. Finally, to evaluate multimodal outputs, we construct a novel multimodal automatic evaluation (MMAE) method which considers both intramodality salience and intermodality relevance. The experimental results show the effectiveness of MMAE.

1 Introduction

Text summarization is to extract the important information from source documents. With the increase of multimedia data on the internet, some researchers (Li et al., 2016b; Shah et al., 2016; Li et al., 2017) focus on multimodal summarization in recent years. Existing experiments (Li et al., 2017, 2018a) have proven that, compared to text summarization, multimodal summarization can improve the quality of generated summary by using information in visual modality.

However, the output of existing multimodal summarization systems is usually represented in a single modality, such as textual or visual (Li et al., 2017; Evangelopoulos et al., 2013; Mademlis et al., 2016). In this paper, we argue that multimodal output[1] is necessary for the following three reasons: 1) It is much easier and faster for users to get critical information from the images (Li et al., 2017). 2) According to our experiments, the multimodal output (text+image) increases users' satisfaction by 12.4% compared to the single-modality output (text) (more details can be found in Sec. 4.2). 3) Images help users to grasp events while texts provide more details related to the events. Thus the images and text can complement each other, assisting users to gain a more visualized understanding of events (Bian et al., 2013). We give an example in Fig. 1 to illustrate this phenomenon. For the output with only the text summary, user will be confused about the description of "four-legged creatures"; while with a relevant image, user will have a clearer understanding of the text.

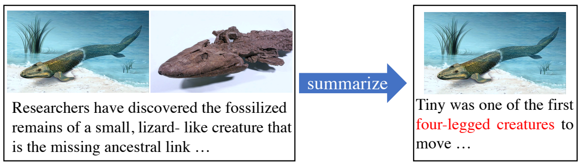

Figure 1: The illustration of our proposed task – Multimodal Summarization with Multimodal Output (MSMO). The image can help better understand the text in the red font.

In recent years, some researchers(Bian et al., 2013, 2015; Wang et al., 2016) focus on incorporating multimedia contents into the output of summarization which all treat the image-text pair as a basic summarization unit. But in our work, our input comes from a document and a collection of images where there is no alignment between texts and images. So our biggest challenge is how to bridge the semantic gaps between texts and images. Based on the above discussion, in this work, we propose a novel task which we refer to as Multimodal Summarization with Multimodal Output (MSMO). To explore this task, we focus on the

---

[1] Note that in this work, the multimodal output refers to a pictorial summary which contains one image (for the sake of simplicity, we first consider only one image) and a piece of text. We leave the other multimodal content (like videos) as future work.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4154–4164
Brussels, Belgium, October 31 - November 4, 2018.

following three questions: 1) how to acquire the relevant data; 2) how to generate the multimodal output; 3) how to automatically evaluate the quality of the multimodal output in MSMO.

For the first question, similar to Hermann et al. (2015), we collect a large-scale multimodal dataset[2] from *Daily Mail* website and annotate some pictorial summaries. For the second question, we propose a multimodal attention model to jointly generate text and the most relevant image, in which the importance of images is determined by the visual coverage vector. For the last question, we construct a novel multimodal automatic evaluation (MMAE) which jointly considers salience of text, salience of image, and image-text relevance.

Our main contributions are as follows:

- We present a novel multimodal summarization task, which takes the news with images as input, and finally outputs a pictorial summary. We construct a large-scale corpus for MSMO studying.
- We propose an abstractive multimodal summarization model to jointly generate summary and the most relevant image.
- We propose a multimodal automatic evaluation (MMAE) method which mainly considers three aspects: salience of text, salience of image, and relevance between text and image.

2 Our Models

2.1 Overview

We begin by defining the MSMO task. The input of the task is a document and a collection of images and the output is a pictorial summary. As shown in Fig. 2, our proposed model consists of four modules: text encoder, image encoder, multimodal attention layer, and summary decoder. The text encoder is a BiLSTM used to encode text. Our image encoder is VGG19[3] pretrained on ImageNet (Simonyan and Zisserman, 2015) used to extract global or local features. The multimodal attention layer aims to fuse textual and visual information during decoding. Our summary decoder, which is a unidirectional LSTM, makes use of information from two modalities to generate the text summary and select the most relevant image according to visual coverage vector. Our text encoder and summary decoder are based on pointer-generator network which we will describe in Sec. 2.2. We then describe image encoder and multimodal attention layer in our multimodal attention model (Sec. 2.3).

2.2 Pointer-Generator Network

See et al. (2017) propose a pointer-generator network which allows both copying words from the source text and generating words from a fixed vocabulary, achieving the best performance on *CNN/Daily mail* dataset. Their model consists of an encoder (a single-layer bidirectional LSTM) and an attentive decoder (a unidirectional LSTM). The encoder maps the article to a sequence of encoder hidden states h_i. During decoding, the decoder receives the embedding of the previous word and reaches a new decoder state s_t. Then the context vector c_t is computed by the attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) as calculated in Eq. 1 and 2. To alleviate the problem of repetition, See et al. (2017) maintain a coverage vector cov^t, which is the sum of attention distributions over all previous decoding timesteps (initialized to zero vector at timestep 0): $cov^t = \sum_{\tilde{t}=0}^{t-1} \alpha^{\tilde{t}}$. The coverage vector is used as an extra input to the attention vector (Eq. 1) and is also used to calculate the coverage loss (Eq. 6). Next, the attention distribution is used to calculate the context vector as follows.

$$e_i^t = v^T \tanh(\mathbf{W}_h h_i + \mathbf{W}_s s_t + \mathbf{W}_c cov^t) \quad (1)$$

$$\alpha^t = \mathrm{softmax}(e^t) \quad (2)$$

$$c_t = \sum_i \alpha_i^t h_i \quad (3)$$

The important part in this model is the calculation of the generation probability p_g. It represents the probability of generating a word from the vocabulary distribution p_v, and $(1 - p_g)$ represents the probability of copying a word from the source by sampling from the attention distribution α^t. p_g is determined by c_t, s_t, and the decoder input x_t in Eq. 4. The final probability distribution over the extended vocabulary, which denotes the union of the vocabulary and all words in the source, is calculated in Eq. 5. Finally, the loss for timestep t is the sum of the negative log likelihood of the target

---

[2] Our dataset has been released to the public, which can be found in `http://www.nlpr.ia.ac.cn/cip/jjzhang.htm`.

[3] `http://www.robots.ox.ac.uk/~vgg/research/very_deep`

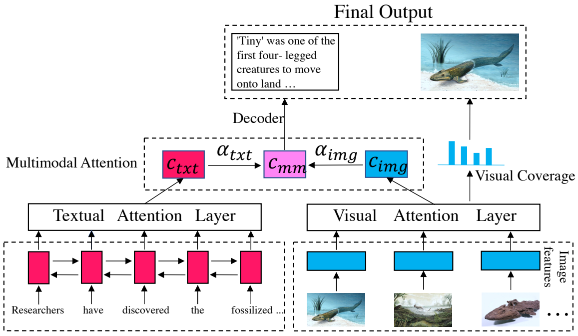

Figure 2: The framework of our model.

word w_t^* and the coverage loss (Eq. 6):

$$p_g = \sigma(\mathbf{W}_h^* c_t + \mathbf{W}_s^* s_t + \mathbf{W}_x x_t) \quad (4)$$

$$p_w = p_g p_v(w) + (1 - p_g) \sum_{w_i = w} \alpha_i^t \quad (5)$$

$$L_t = -\mathrm{log} p_{w_t^*} + \sum_i \min(\alpha_i^t, cov_i^t) \quad (6)$$

2.3 Multimodal Attention Model

We incorporate visual information into the pointer-generator network and propose a novel multimodal attention model. As shown in Fig. 2, there are three main differences between our model and pointer-generator network: 1) We have an extra image encoder and a corresponding visual attention layer; 2) To achieve the fusion of textual and visual information, we introduce a multimodal attention mechanism; 3) We add a visual coverage (Li et al., 2018a) to both alleviate visual repetition and measure the salience of image. More details are as follows.

Image Encoder. We apply the VGG19 to extract global and local image feature vectors for all images. The global features g are 4096-dimensional activations of the pre-softmax fully-connected layer `fc7`. The local features l are the $7 \times 7 \times 512$ feature maps of the last pooling layer (`pool5`). We flatten the local feature into a matrix $\mathbf{A} = (a_1, \cdots, a_L)(L = 49)$ where $a_l \in \mathbb{R}^{512}$ corresponds to a patch of an image.

Visual Attention. The attention mechanism is learned to focus on different parts of input text while decoding. Attention mechanisms have also shown to work with other modalities, like images, where they can learn to attend the salient parts of an image (Xu et al., 2015). We then explore to use images with a visual attention to learn text-image alignment. Concretely, we extend attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) to visual attention mechanisms, which attend visual signals. There are three variants of our visual attention mechanisms: 1) attention on global features (ATG), 2) attention on local features (ATL), and 3) hierarchical visual attention on local features (HAN). We take the calculation of ATG as an example. To attend to the salient parts of a collection of images with size M, we flatten the global feature set g into a matrix $g\prime = (g_1, \cdots, g_M)$. In addition to calculating the text context vector in Sec. 2.2, we also obtain a visual context vector. We first project the image feature into the same dimension as the text context vector. The visual attention is calculated as follows:

$$g^* = \mathbf{W}_I^2(\mathbf{W}_I^1 g + b_I^1) + b_I^2 \quad (7)$$

$$e_a^t = v_a^T \tanh(\mathbf{W}_a g_i^* + \mathbf{U}_a s_t + cov_a^t) \quad (8)$$

$$\alpha_a^t = \mathrm{softmax}(e_a^t) \quad (9)$$

where $\mathbf{W}_I^1 \in \mathbb{R}^{4096 \times 4096}$ and $\mathbf{W}_I^2 \in \mathbb{R}^{4096 \times d_h}$ are the image transformation matrices, $b_I^1 \in \mathbb{R}^{4096}$ and $b_I^2 \in \mathbb{R}^{d_h}$ are bias vectors, and cov_a^t denotes the visual coverage vector and is initialized to zero vector in the beginning. Then the visual attention distribution α_a^t is used to obtain the visual context vector c_{img}^t through $c_{img}^t = \sum_i \alpha_{a,i}^t g_i^*$. Similar is the ATL, we flatten the local feature set A into a matrix $A' = (a_1, \cdots, a_{M \times 49})$. The calculation of attention in ATL is the same as in ATG. There is a bit difference in the HAN model, which first attend to the 49 image patches and get an intermediate visual context vector to represent the image, and then attend to the intermediate visual context vectors to get the visual context vector.

Multimodal Attention. To fuse the text and visual context information, we add a multimodal attention layer (Li et al., 2018a), as shown in Fig. 2. And the attention distribution is calculated as follows:

$$e_{txt}^t = v_{txt}^T(\mathbf{W}_{txt} c_{txt}^t + \mathbf{U}_{txt} s_t) \quad (10)$$

$$e_{img}^t = v_{img}^T(\mathbf{W}_{img} c_{img}^t + \mathbf{U}_{img} s_t) \quad (11)$$

$$\alpha_{txt}^t = \mathrm{softmax}(e_{txt}^t) \quad (12)$$

$$\alpha_{img}^t = \mathrm{softmax}(e_{img}^t) \quad (13)$$

$$c_{mm}^t = \alpha_{txt}^t c_{txt}^t + \alpha_{img}^t c_{img}^t \quad (14)$$

where α_{txt}^t is the attention weight for text context vector and α_{img}^t is the attention weight for visual context vector.

Visual Coverage. In addition to the calculation of the text coverage vector as in Sec. 2.2, we also obtain a visual coverage vector cov_{img}^t, which is the sum of visual attention distributions. To help reduce repeated attention to multimodal information, we incorporate a text coverage loss and a visual coverage loss into the loss function. The final loss function is as follows:

$$L_t = -\log p_{w_t^*} + \sum_i \min(\alpha_i^t, cov_i^t) + \sum_j \min(\alpha_j^t, cov_{img,j}^t) \quad (15)$$

The attention mechanism can attend the salient parts of texts or images. Meanwhile, the coverage mechanism sums up all the historical attention distributions. Therefore, we regard the coverage vector as a global salience measure of the source being attended. We then use the visual coverage vector in the last decoding timestep to select the most relevant image. Concretely, we choose the image whose coverage score is the largest. The process is a bit different for the local features. An image corresponds to 49 patches, the coverage scores of these patches are summed up to get the salience score of the image as follows:

$$\mathrm{S}_j = \sum_{patch} cov_{patch,j}^{t^*} \quad (16)$$

where $\mathrm{S_j}$ denotes the salience of the j-th image and $cov_{patch,j}^{t^*}$ denotes the coverage score of each corresponding image patch in the last decoding timestep $t*$. For the HAN, we introduce an extra coverage vector for the image patches attention and calculate coverage loss for it as follows:

$$L_t = -\log p_{w_t^*} + \sum_k \min(\alpha_k^t, cov_{patch,k}^t) + \sum_i \min(\alpha_i^t, cov_i^t) + \sum_j \min(\alpha_j^t, cov_{img,j}^t) \quad (17)$$

3 Multimodal Automatic Evaluation

To evaluate the quality of a pictorial summary, we propose the MMAE method which is defined as $y = f(m_1, m_2, m_3)$. In this definition, m_1, m_2, and m_3 denote scores measured by three metrics which consider salience of text (Sec. 3.1), salience of image (Sec. 3.2), and image-text relevance (Sec. 3.3) respectively, $f(\cdot)$ denotes a mapping function, and y denotes the score of the pictorial summary.

In our experiments, the reference pictorial summary consists of a text summary and a reference image set[4] $\mathrm{ref}_{\mathrm{img}}$. In MMAE, m_1 is obtained by comparing the text summary in reference with that in model output, m_2 is obtained by comparing the image set in reference with the image in model output, and m_3 considers the image-text similarity in model output. To learn MMAE, we choose three simple methods to fit y with human judgment scores. These methods include Linear Regression (LR), and two nonlinear methods: Logistic Regression (Logis), and Multilayer Perceptron (MLP).

3.1 Salience of Text

ROUGE (Lin, 2004b) is widely used to automatically assess the quality of text summarization systems. It has been shown that ROUGE correlates well with human judgments (Lin, 2004a; Owczarzak et al., 2012; Over and Yen, 2004). Therefore, we directly apply ROUGE to assess the salience of the text units.

3.2 Salience of Image

We propose a metric, namely, image precision (**IP**), to measure the salience of image. The image precision is defined as follows:

$$\mathrm{IP} = \frac{|\{\mathrm{ref}_{\mathrm{img}}\} \cap \{\mathrm{rec}_{\mathrm{img}}\}|}{|\{\mathrm{rec}_{\mathrm{img}}\}|} \quad (18)$$

where ref_{img}, rec_{img} denote reference images and recommended images by MSMO systems respectively. The reasons for this metric are as follows.

A good summary should have good coverage of the events for both texts and images. The image in the output should be closely related to the events. So we formulate the image selection process as an image recommendation —instead of recommending items to users as in a recommendation system, we recommend the most salient image to an event. It can also be viewed as an image retrieval task, which retrieves the image most relevant to an event. Precision and recall are commonly used to evaluate recommendation systems (Karypis, 2001) and information retrieval task (Zuva and Zuva, 2012). However, we only care about whether the image appears in the reference image set. Thus in our case, we are only interested in calculating precision metric. Therefore, we adapt the precision here as **IP** to measure image salience.

---

[4] More details can be found in Sec. 4.1

3.3 Image-Text Relevance

A prerequisite for a pictorial summary to help users accurately acquire information is that the image must be related to the text. Therefore, we regard the image-text relevance as one of metrics to measure the quality of the pictorial summary. We consider using visual-semantic embedding (Faghri et al., 2018; Wang et al., 2018) to calculate the cosine similarity between visual feature and textual feature, which we use as image-text relevance. Visual-semantic embedding has been widely used in cross-modal retrieval (Kiros et al., 2014) and image captioning (Karpathy and Fei-Fei, 2015).

We apply VSE0 model of Faghri et al. (2018), which achieves state-of-the-art performance for image-caption retrieval task on the Flickr30K dataset (Young et al., 2014). The difference is that instead of training a CNN model to encode the image, we use the pretrained VGG19 to extract global features. The text is encoded by a unidirectional Gated Recurrent Unit (GRU) to a sequence of vector representations. Then we apply the max-over-time pooling (Collobert et al., 2011) to get a single vector representation. Next, the visual features and text features are projected to a joint semantic space by two feed-forward neural networks. The whole network is trained using a max-margin loss:

$$\begin{aligned} L = \sum_{\hat{c}} \max(\beta - s(i,c) + s(i,\hat{c}), 0) \\ + \sum_{\hat{i}} \max(\beta - s(i,c) + s(\hat{i},c), 0) \end{aligned} \quad (19)$$

The loss comprises two symmetric terms, with i and c being images and captions repectively. The first term is taken over negative captions $\hat{c}$ image i in a batch. The second is over negative images $\hat{i}$ given caption c. If i and c are closer to each other in the joint embedding space than to any other negative pairs, by a margin β, the loss is zero. We choose to use image-caption pairs in our dataset to train the VSE0 model.

4 Experiments

We conduct the following five sets of experiments: 1) To verify our motivation of the multimodal output (pictorial summary), we design an experiment for user satisfaction test (Sec. 4.2); 2) We compare our multimodal summarization with text summarization from both ROUGE score and manual evaluation (Sec. 4.3); 3) To verify the effectiveness of our evaluation metrics, we calculate the correlation between these metrics and human judgments (Sec. 4.4); 4) We conduct two experiments to show the effectiveness of our proposed MMAE and the generalization of MMAE respectively (Sec. 4.5); 5) Finally, we evaluate our multimodal attention model with MMAE (Sec. 4.6).

The hyperparameters in our model are similar to See et al. (2017), except that we set the maximum number of images to 10, 7, and 7 for ATG, ATL, and HAN respectively, because different articles have the image collection of different sizes. The images are sorted in the order of the position in the article.

4.1 Dataset

There is no large-scale benchmark dataset for MSMO. We follow Hermann et al. (2015) to construct a corpus from *Daily Mail* website[5]. Similar to Hermann et al. (2015), we use the manually-written highlights offered by *Daily Mail* as a reference text summary. From *Daily Mail*, we randomly select articles within a week and find that 2,917 out of 2,930 articles contain images. More details are illustrated in Table 1.

	train	valid	test
#Documents	293,965	10,355	10,261
#ImgCaps	1,928,356	68,520	71,509
#AvgTokens(S)	720.87	766.08	730.80
#AvgTokens(R)	70.12	70.02	72.16
#AvgCapTokens	22.07	22.64	22.34
#AvgImgCaps	6.56	6.62	6.97

Table 1: Corpus statistics. Each image on the website is paired with a caption. *#ImgCaps* denotes the number of image-caption pairs. *#AvgTokens(S)*, *#AvgTokens(R)* and *#AvgCapTokens* denote the average number of tokens in articles, highlights, and captions respectively.

To get the pictorial reference, we employ 10 graduate students to select the relevant images from the article for each reference text summary. We allow annotators to select up to three images to reduce the difference between different annotators. If the annotators find that there is no relevant image, they will select none of them. Each article is annotated by at least two students[6]. Since we use the text reference to guide the generation of the pictorial summary, we do not use the reference image during training. Therefore, we only conduct the annotation on the test set.

---

[5] `http://www.dailymail.co.uk`

[6] A third annotator will be asked to decide the final annotation for the case of divergence for the first two annotators.

4.2 User Satisfaction Test

We conduct an experiment to investigate whether a pictorial summary can improve the user satisfaction for the informativeness of the summary. For a fair comparison, we propose a novel strategy to compare text summaries and pictorial summaries. We take an example to illustrate our strategy. Given 100 source news pages, we have their corresponding reference text summaries and pictorial summaries. We divide them into two parts of the same size, part 1 and part 2. In part 1, human annotator A evaluates the text summaries according to the input news, and human annotator B evaluates the pictorial summaries. In part 2, annotator A evaluates the pictorial summaries and annotator B evaluates the text summaries. All annotators will give a score of 1 to 5. The input news is the same for annotator A and annotator B.

Format	$\text{Annotator}_{\text{A}}$	$\text{Annotator}_{\text{B}}$	Overall
Text	3.67	3.75	3.71
Pictorial	4.14	4.20	4.17

Table 2: User satisfaction test results. In total, we use the strategy mentioned in Section 4.2 to evaluate 400 randomly selected source news pages. *Overall* denotes the average score on these 400 samples.

Table 2 shows our results for user satisfaction test. User ratings of pictorial summaries are 12.4% higher than text summaries. It shows that users prefer this way of presenting information. It also confirms our motivation for MSMO.

4.3 Comparison with Text Summarization

Our user satisfaction test in Sec. 4.2 is done in an ideal situation, comparing the text reference with the pictorial reference. To show the effectiveness of our model, we also compare our model with text summarization from ROUGE and human judgment scores. We compare several abstractive summarization methods with our multimodal summarization methods. **PGC**[7] (See et al., 2017) refers to the pointer-generator network (Sec. 2.2). **AED** (Nallapati et al., 2016) uses an attentional encoder-decoder framework and adds some linguistic features such as POS, named-entities, and TF-IDF into the encoder. We also implement a seq2seq model with attention (**S2S+attn**). To compare the multimodal output with our multimodal model, we propose an extractive method based on GuideRank (**GR**) (Li et al., 2016a, 2018b). GuideRank applies LexRank (Erkan and Radev, 2004) with guidance strategy. In this strategy, captions recommend the sentences related to them. The rankings of sentences and captions are obtained through GR; we extract sentences that satisfy the length limit as a text summary according to the ranking of text. We select an image whose caption ranks the first in the captions. And finally, the pictorial summary is obtained. We evaluate different summarization models with the standard ROUGE metric, reporting the F1 scores for ROUGE-1, ROUGE-2, and ROUGE-L. Our ROUGE results are given in Table 3, and human judgment scores are given in Table 4.

	Model	ROUGE-1	ROUGE-2	ROUGE-L
Base	S2S+attn	32.32	12.44	29.65
	AED	34.78	13.10	32.24
	PGC	**41.11**	**18.31**	**37.74**
MM	ATG	40.63	18.12	37.53
	ATL	**40.86**	18.27	**37.75**
	HAN	40.82	**18.30**	37.70
	GR	37.13	15.03	30.21

Table 3: ROUGE F1 scores on our test set. All our ROUGE scores are reported by official ROUGE script.

Model	PGC	ATG	ATL	HAN
HS	3.07	3.30	3.22	3.20

Table 4: Human judgment scores for our multimodal model and PGC. We randomly select 400 articles and use the same strategy as Sec. 4.2. HS denotes the average human judgment scores.

From Table 3, all multimodal models lead to a decrease in ROUGE scores which can attribute to the following reasons. There are 6.56 images on average in each article and not every image is closely related to the event of the article. In other words, some images are noise. On the other hand, our text input is long text, and it contains enough information for text generation. In Table 4, multimodal models are better than text model in human judgments. It further illustrates our motivation, and also proves the effectiveness of our models.

4.4 Correlation Test

To illustrate the effectiveness of our evaluation metrics, we conduct an experiment on correlations between these metrics and human judgment scores. Human annotators give a score which ranges from 1 to 5 to a pictorial summary accord-

---

[7] `https://github.com/abisee/pointer-generator`

ing to the reference[8]. The reference consists of a text summary and up to three relevant images selected by humans. We randomly extract the pictorial summaries from the output of different systems. In response to the three aspects we proposed in Section 3, we propose some related metrics respectively. For text salience, we apply **ROUGE-1**, **ROUGE-2**, **ROUGE-L**, and **BLEU**. For image-text relevance of candidate pictorial summaries, we propose two ways. One is to calculate the similarity (**Img-Sum**) between the image and the whole text summary. The other is to calculate the similarities between the image and each sentence in the text summary. Then we take the maximum and average values as two metrics: **MAX$_{\text{sim}}$** and **AVG$_{\text{sim}}$**. For image salience, in addition to the **IP** metric mentioned in Section 3.2, we try to calculate the similarity between the candidate image and each reference image in three ways: 1) **I-I**: similarities between the global `fc7` features, 2) **Hist**: Bhattacharyya distance[9] for histogram comparison (Bhattacharyya, 1943), and 3) **Temp**: Fourier Analysis template matching (Briechle and Hanebeck, 2001).

We employ annotators to evaluate 600 samples (randomly selected from the outputs of each model on the validation set). Each sample is scored by two persons and we take the average score as the final score. We use 450 of them as training set to train the MMAE model in Sec. 4.5, the rest is used as test set. The scores calculated by each evaluation metric are then tested on the training set to see how well they correlate with human judgments. The correlation is evaluated with three metrics, including 1) Pearson correlation coefficient (r), 2) Spearman rank coefficient (ρ), and 3) Kendall rank coefficient (τ). Our results of correlation test are given in Table 5.

As shown in Table 5, **IP** (Image Precision) correlates best with human assessments according to the three correlation coefficients. It illustrates that people pay more attention to images when assessing pictorial summaries. If we choose the right image for the summary, people are more likely to assign a high score. We also note that the correlation score of **IP** is significantly higher than four text metrics. Because it is easy for a person to judge the importance of images based on reference, such as to see whether the image appears in reference. However, measuring the semantic similarity of two texts is difficult. The four metrics all measure the degree of n-gram overlap which cannot accurately measure semantic similarity.

	Metric	r	ρ	τ
Text	BLEU	.1949	.1542	.1198
	ROUGE-1	.3006	.2941	.2152
	ROUGE-2	.2735	.2742	.2002
	ROUGE-L	**.3144**	**.3087**	**.2272**
Image-Text	AVG$_{\text{sim}}$	.2662	.2388	.1774
	MAX$_{\text{sim}}$	**.2849**	**.2749**	**.2033**
	Img-Sum	.2380	.2075	.1556
Image	I-I (max)	.0169	.0258	.0196
	I-I (avg)	-.0262	-.0140	-.0113
	Hist$_{\text{avg}}$	.4688	.5077	.3725
	Hist$_{\text{max}}$	.5974	.6388	.5149
	Temp$_{\text{avg}}$	.4913	.4944	.3631
	Temp$_{\text{max}}$	.5967	.6435	.5080
	IP	**.6407**	**.6482**	**.5789**

Table 5: Correlation with human judgment scores (training set), measured with Pearson r, Spearman ρ, and Kendall τ coefficients. The max and avg denote the maximum and average value of the scores.

For the image-text relevance, **MAX$_{\text{sim}}$** performs best and is comparable to the several ROUGE metrics. It shows that in a good pictorial summary, the image and text should be relevant. In some cases, even though the generated text is not so important, the image is closely related to the text. At this time, people can also be satisfied. On the other hand, our VSE0 (Sec. 3.3) model can capture some fluency of sentences by adopting GRU. Compare **MAX$_{\text{sim}}$**, **AVG$_{\text{sim}}$**, and **Img-Sum**, this is very intuitive. Once people find a sentence (or a part) relevant to the image, they will think the image is related to the text. Besides, the worst performance of **Img-Sum** metric is probably because the average length of captions used to train VSE0 model is about 22, far less than the length of the summary. We find the **I-I (max)** and **I-I (avg)** nearly do not correlate with human assessments. It shows that the visual features extracted from VGG19 are not suitable for calculating the similarity between news images. The analysis of **Hist (Temp)$_{\text{avg}}$** and **Hist (Temp)$_{\text{max}}$** is similar to the analysis of **MAX$_{\text{sim}}$** and **AVG$_{\text{sim}}$** above.

4.5 Effectiveness and Generalization of MMAE

We then select the best-performing metrics separately from the three sets of metrics, namely

---

[8] Some articles are annotated with no relevant images (about 3.9%), we directly skipped these articles without manual scoring

[9] In statistics, the Bhattacharyya distance measures the similarity of two discrete or continuous probability distributions. For a distance d, we take $(1-d)$ as the similarity.

ROUGE-L, **MAX$_{\text{sim}}$**, and **IP**. We apply LR, MLP, and Logis to learn our MMAE model that combines the three metrics. We calculate the three coefficients for the three metrics on the test set as a comparison. The correlation results are given in Table 6.

Metric	r	ρ	τ
ROUGE-L	.3488	.3554	.2669
MAX$_{\text{sim}}$	.2541	.2339	.1773
IP	**.5982**	**.5966**	**.5485**
MMAE$_{\text{LR}}$	**.6646**	.6644	.5265
MMAE$_{\text{MLP}}$	.6632	.6646	.5265
MMAE$_{\text{Logis}}$	.6630	**.6653**	**.5277**

Table 6: Correlation with human judgment scores (test set).

As shown in Table 6, the MMAE learned by three methods correlates better with human judgments. Although MMAE$_{\text{Logis}}$ gets a slightly higher correlation score according to Spearman and Kendall coefficients, we choose the MMAE$_{\text{LR}}$[10] as our final MMAE model due to Occam's Razor[11].

It is crucial that MMAE can generalize for a previously unseen system. To test the generalization of MMAE, we use MMAE to evaluate a new system and calculate the correlation with human judgment scores. The new system is a naive model which applies LexRank to extract sentences and randomly select an image from source. We can observe that MMAE still correlates well with human judgment scores, as shown in Table 7. It illustrates that MMAE generalize well for a new model. We give some examples of MMAE in supplementary material.

Metric	r	ρ	τ
ROUGE-L	.3223	.3514	.2615
MMAE	.6352	.6318	.4728

Table 7: Correlation results for the new model on the same 150 test samples as in Sec. 4.4.

4.6 Model Performances

According to our analyses above, we have proved MMAE can evaluate multimodal output. In this section, we report the MMAE scores for our proposed multimodal attention model, as shown in Table 8.

Model	ROUGE-L	MAX$_{\text{sim}}$	IP	MMAE
ATG	40.76	25.82	59.28	**3.35**
ATL	40.80	13.26	**62.44**	3.26
HAN	**40.82**	12.22	61.83	3.25
GR	30.20	**26.60**	61.70	3.20

Table 8: Results evaluated by our MMAE method. We skipped the articles that are labeled as no relevant images. Finally, only 9,851 of the 10,261 articles are left.

Surprisingly, the model ATG achieves the highest MMAE score despite the mediocre performance in three individual metrics. The **MAX$_{\text{sim}}$** score of ATG is much higher than ATL and HAN. It shows the global features can help to learn better image-text alignments. Since GR itself makes use of the image-caption pairs, it is natural to get a high image-text relevance score. Our proposed multimodal attention models all achieves higher performance than the extractive baseline GR, which further indicate the effectiveness of our models.

5 Related Work

Different from text summarization (Wan and Yang, 2006; Rush et al., 2015; Zhu et al., 2017; See et al., 2017; Celikyilmaz et al., 2018; Paulus et al., 2018), Multimodal Summarization is a task to generate a condensed text summary or a few keyframes to help acquire the gist of multimedia data. One of the most significant advantages of the task is that it does not rely solely on text information, but it can also utilize the rich visual content from the images.

In recent years, much work has focused on multimodal summarization. Evangelopoulos et al. (2013) detect salient events in a movie based on the saliency of individual features for aural, visual, and linguistic representations. Li et al. (2017) generate the text summary from an asynchronous collection of text, image, audio, and video. There has also been some work (Bian et al., 2013, 2015; Wang et al., 2016; Qian et al., 2016) focused on producing multimodal output for summarization. Bian et al. (2013, 2015) aim to produce a visualized summary for microblogs. Wang et al. (2016) generate a pictorial storyline for summarization. Qian et al. (2016) generate the multimedia topics for social events. But these researches all treat image-text pairs, in which texts and images are aligned, as a basic summarization unit. For example, the images are aligned with the text in a mi-

---

[10] The weight for ROUGE-L, MAX$_{\text{sim}}$, and IP is 1.641, 0.854, 0.806 respectively and the intercept is 1.978.

[11] `https://en.wikipedia.org/wiki/Occam%27s_razor`

croblog post; Wang et al. (2016) obtain the image-text pairs by using image search engine. None of the above works focuses on generating multimodal output from a collection of texts and images that are not explicitly aligned. This is one of the goals in this paper. Another difference is that they separately evaluate texts and images when evaluating the final results. In our work, we propose a new automatic evaluation which jointly considers two aspects of textual and visual modalities.

6 Conclusion

In this paper, we focus on a novel task which aims to automatically generate a multimodal summary from multimodal news, where the images and the texts are not explicitly aligned. We provide a multimodal summarization method to jointly generate text and the most relevant image, which can be referred as the baseline for further study. Our proposed metrics have been proved to be effective in evaluating the multimodal output. Moreover, the idea of constructing our MMAE can be easily extended to other modalities. That is, we both consider the intramodality salience and the intermodality relevance.

7 Acknowledgments

The research work described in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2016QY02D0303. We thank the three anonymous reviewers for their careful reading of our paper and their many insightful comments and suggestions. We also thank Shaonan Wang, Xiaomian Kang, and Yang Zhao for helpful discussions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Anil Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109.

Jingwen Bian, Yang Yang, and Tat-Seng Chua. 2013. Multimedia summarization for trending topics in microblogs. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1807–1812.

Jingwen Bian, Yang Yang, Hanwang Zhang, and Tat-Seng Chua. 2015. Multimedia summarization for social events in microblog stream. *IEEE Transactions on multimedia*, 17(2):216–228.

Kai Briechle and Uwe D Hanebeck. 2001. Template matching using fast normalized cross correlation. In *Optical Pattern Recognition XII*, volume 4387, pages 95–103.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1662–1675.

Ronan Collobert, Jason Weston, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, 12(1):2493–2537.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Georgios Evangelopoulos, Athanasia Zlatintsi, Alexandros Potamianos, Petros Maragos, Konstantinos Rapantzikos, Georgios Skoumas, and Yannis Avrithis. 2013. Multimodal saliency and fusion for movie summarization based on aural, visual, and textual attention. *IEEE Transactions on Multimedia*, 15(7):1553–1568.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. Vse++: Improving visual-semantic embeddings with hard negatives. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 1693–1701.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3128–3137.

George Karypis. 2001. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Haoran Li, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2016a. Guiderank: A guided ranking graph model for multilingual multi-document summarization. In *The Fifth Conference on Natural Language Processing and Chinese Computing & The Twenty Fourth International Conference on Computer Processing of Oriental Languages (NLPCC-ICCPOL)*, pages 608–620.

Haoran Li, Junnan Zhu, Tianshang Liu, Jiajun Zhang, and Chengqing Zong. 2018a. Multi-modal sentence summarization with modality attention and image filtering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4152–4158.

Haoran Li, Junnan Zhu, Cong Ma, Jiajun Zhang, and Chengqing Zong. 2017. Multi-modal summarization for asynchronous collection of text, image, audio and video. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1092–1102.

Haoran Li, Junnan Zhu, Cong Ma, Jiajun Zhang, and Chengqing Zong. 2018b. Read, watch, listen and summarize: Multi-modal summarization for asynchronous text, image, audio and video. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*.

Zechao Li, Jinhui Tang, Xueming Wang, Jing Liu, and Hanqing Lu. 2016b. Multimedia news summarization in search. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(3):33.

Chin-Yew Lin. 2004a. Looking for a few good metrics: Automatic summarization evaluation-how many samples are enough? In *Proceedings of NII Testbeds and Community for information access Research (NTCIR)*.

Chin-Yew Lin. 2004b. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421.

Ioannis Mademlis, Anastasios Tefas, Nikos Nikolaidis, and Ioannis Pitas. 2016. Multimodal stereoscopic movie summarization conforming to narrative characteristics. *IEEE Transactions on Image Processing*, 25(12):5828–5840.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Paul Over and James Yen. 2004. An introduction to duc 2004 intrinsic evaluation of generic new text summarization systems. In *Proceedings of the Document Understanding Conference (DUC)*.

Karolina Owczarzak, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Shengsheng Qian, Tianzhu Zhang, and Changsheng Xu. 2016. Multi-modal multi-view topic-opinion mining for social event analysis. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 2–11.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 379–389.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1073–1083.

Rajiv Ratn Shah, Yi Yu, Akshay Verma, Suhua Tang, Anwar Dilawar Shaikh, and Roger Zimmermann. 2016. Leveraging multimodal information for event summarization and concept-level sentiment analysis. *Knowledge-Based Systems*, 108:102–109.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Xiaojun Wan and Jianwu Yang. 2006. Improved affinity graph based multi-document summarization. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 181–184.

Liwei Wang, Yin Li, Jing Huang, and Svetlana Lazebnik. 2018. Learning two-branch neural networks for image-text matching tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

William Yang Wang, Yashar Mehdad, Dragomir R Radev, and Amanda Stent. 2016. A low-rank approximation approach to learning joint embeddings of news stories and images for timeline summarization. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 58–68.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, pages 2048–2057.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics (TACL)*, 2:67–78.

Junnan Zhu, Long Zhou, Haoran Li, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2017. Augmenting neural sentence summarization through extractive summarization. In *Proceedings of the 6th Conference on Natural Language Processing and Chinese Computing (NLPCC)*, pages 16–28.

Keneilwe Zuva and Tranos Zuva. 2012. Evaluation of information retrieval systems. *International Journal of Computer Science & Information Technology*, 4(3):35.

Frustratingly Easy Model Ensemble for Abstractive Summarization

Hayato Kobayashi
Yahoo Japan Corporation / RIKEN AIP
`hakobaya@yahoo-corp.jp`

Abstract

Ensemble methods, which combine multiple models at decoding time, are now widely known to be effective for text-generation tasks. However, they generally increase computational costs, and thus, there have been many studies on compressing or distilling ensemble models. In this paper, we propose an alternative, simple but effective unsupervised ensemble method, *post-ensemble*, that combines multiple models by selecting a majority-like output in post-processing. We theoretically prove that our method is closely related to kernel density estimation based on the von Mises-Fisher kernel. Experimental results on a news-headline-generation task show that the proposed method performs better than the current ensemble methods.

1 Introduction

Recent success in deep learning, especially encoder-decoder models (Sutskever et al., 2014; Bahdanau et al., 2015), has dramatically improved the performance of various text-generation tasks, such as translation (Johnson et al., 2017), summarization (Ayana et al., 2017), question-answering (Choi et al., 2017), and dialogue response generation (Dhingra et al., 2017). In these studies on neural text generation, it has been known that a model-ensemble method, which predicts output text by averaging multiple text-generation models at decoding time, is effective even for text-generation tasks, and many state-of-the-art results have been obtained with ensemble models. However, an ensemble method has a clear drawback in that it increases computational costs, i.e., the increase in time as the number of models increases, since it averages the word-prediction probabilities of all models in each decoding step. Therefore, there have been many studies on model compression or distillation for ensemble methods, each of which has successfully shrunk an ensemble model (Hinton et al., 2015; Chebotar and Waters, 2016; Kuncoro et al., 2016; Kim and Rush, 2016; Stahlberg and Byrne, 2017; Freitag et al., 2017).

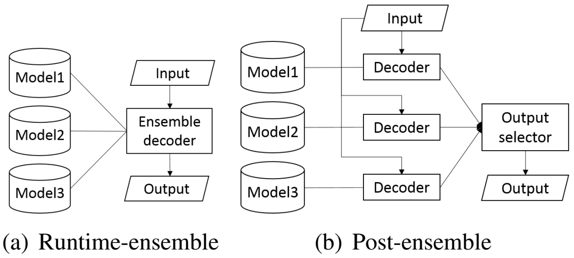

Figure 1: Flow charts of current runtime-ensemble (a) and our proposed post-ensemble (b).

In this paper, we propose an alternative method for model ensemble inspired by the majority vote in classification tasks (Littlestone and Warmuth, 1994). Majority vote is a method that selects the most frequent label from the predicted labels of multiple classifiers in post-processing. Similarly, our method involves selecting a majority-like output from the generated outputs of multiple text-generation models in post-processing as in Fig. 1(b), instead of averaging models at decoding time as in Fig. 1(a). The difference between a classification task and text-generation task is that we need to consider a sequence of labels for each model output in a text-generation task, although we consider only one label in a classification task. This means a majority output may not exist since each output will be basically different from other outputs, which are generated from different models. To overcome this problem, we propose an unsupervised method for selecting a majority-like output close to the other outputs by using cosine similarity. The idea is quite simple, but experiments showed that our method is more effective than the current ensemble methods.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4165–4176
Brussels, Belgium, October 31 - November 4, 2018.

Our work can open up a new direction for two research communities: model ensemble and hypotheses reranking (see Sec. 6 for detailed descriptions of the related studies). For the first, we suggest a new category of ensemble algorithms that corresponds to the output selection in classification tasks. In classification tasks, there are roughly three approaches for model ensemble: model selection in preprocessing, model average at runtime, and output selection in post-processing. In text generation studies, model selection by cross-validation and model average with an ensemble decoder have been frequently used, but output selection as typified by majority vote has received less attention because of the fact that a majority output may not exist, as described above. Therefore, there is enough room to study this direction in the future. Since our algorithm in this paper is quite simple, we expect that more sophisticated methods can improve the results even over our approach.

For the hypotheses reranking research community, we suggest a new category of reranking tasks, where we need to select the best output from the generated outputs of multiple models, instead of the N-best hypotheses of a single model. Hypotheses reranking for a text-generation model is related to our task, but in this case, a reranking method based on a language model is frequently used and is basically enough to correct the scoring of a beam search with a single model (Chen et al., 2006; Vaswani et al., 2013; Luong and Popescu-Belis, 2016) since the purpose is to obtain a fluent output and remove erroneous outputs, assuming the model can generate good outputs. A clear difference between our task and the reranking task is that we should consider all outputs to decide the goodness of an output because a fluent output is not always appropriate in this task. This is similar to extractive summarization (Erkan and Radev, 2004) but is significantly different from our task in that our output candidates have almost the same meaning.

Our contributions in this paper are as follows.

- We propose a simple, fast, and effective method for unsupervised ensembles of text generation models, where (i) the implementation is "frustratingly easy" without any modification of model code (Alg. 1), (ii) the computational time is enough for practical use (Sec. 5.3), i.e., an ensemble time of 3.7 ms per sentence against a decoding time of 44 ms, and (iii) the performance is competitive with the state-of-the-art results (Sec. 5.2), i.e., our method (ensemble of 32 models) for 37.52 ROUGE-1 against the state-of-the-art method (single model) for 37.27 ROUGE-1 on a news-headline-generation task.
- We prove that our method is an approximation of finding the maximum density point by kernel density estimation based on the von Mises-Fisher kernel (Sec. 4). In addition, we derive a formula of the error bound of this approximation.
- We will release the 128 prepared models used in this paper (Sec. 5.1), each of which was trained for more than two days, as a new dataset to improve ensemble methods.

2 Preliminaries

In Sec. 2.1, we briefly explain an encoder-decoder model for text generation, and in Sec. 2.2, we discuss the current ensemble methods for combining multiple text generation models at decoding time.

2.1 Encoder-Decoder Model

An encoder-decoder model is a conditional language generation model, which can learn rules for generating an appropriate output sequence corresponding to an input sequence by using the statistics of many correct pairs of input and output sequences, e.g., news articles and their headlines. When training this model, we calculate a conditional likelihood,

$$p(y \mid x) = \prod_{t=1}^{T-1} p(y_{t+1} \mid y_{\le t}, x), \qquad (1)$$

with respect to each pair (x, y) of input sequence $x = x_1 \cdots x_S$ and output sequence $y = y_1 \cdots y_T$, where $y_{\le t} = y_1 \cdots y_t$, and maximize its mean. The model $p(y \mid x)$ in Eq. (1) is achieved by combining two recurrent neural networks, called an *encoder* and *decoder*. The former reads an input sequence x to recognize its content, and the latter predicts an output sequence y corresponding to the content.

After training, we can obtain an output y from an input x by using a learned model $p(y \mid x)$. Since the calculation of an optimal output is clearly intractable, most studies used a beam search, which is a greedy search algorithm that keeps a limited number of best partial solutions, whose size is called the *beam size*. Formally, a set of best partial solutions of beam size b at step t is represented as $Y^b_{\le t}$, which is recursively defined as the top b elements with respect to $p(y_{\leq t} \mid x)$, where $y_{\leq t} \in Y^b_{\leq t-1} \times Y$. The Y is a set of available elements for y_i, or a target dictionary. Let start and goal meta symbols be <s> and </s>, respectively. A beam search procedure starts from $Y_{\leq 0} = \{$<s>$\}$ and finishes when the last symbols of all elements in $Y^b_{\leq t}$ are the goal element </s> or when its length t becomes larger than some threshold.

2.2 Runtime-Ensemble

In a text-generation task, model ensemble is a method of predicting a next word by averaging the word-prediction probabilities of multiple text-generation models at decoding time. Fig. 1(a) shows a flow chart of the current ensemble methods, which we call *runtime-ensemble* to distinguish them from our method. There are mainly two variants of runtime-ensemble using arithmetic mean $\overline{p}_{\mathrm{a}}$ and geometric mean $\overline{p}_{\mathrm{g}}$, which are defined as

$$\overline{p}_{\mathrm{a}}(y_{\leq t} \mid x) = \tfrac{1}{|M|} \textstyle\sum_{p \in M} p(y_{\leq t} \mid x), \tag{2}$$

$$\overline{p}_{\mathrm{g}}(y_{\leq t} \mid x) = \left(\textstyle\prod_{p \in M} p(y_{\leq t} \mid x)\right)^{\frac{1}{|M|}}, \tag{3}$$

where M is a set of learned models. We call the former `EnsSum` and the latter `EnsMul`. Although there have been no comparative experiments, `EnsMul` is usually used since most decoding programs keep $\log p$ and calculating $\sum_{p \in M} \log p$ is enough to obtain the top b words with respect to $\overline{p}_{\mathrm{g}}$ for a beam search procedure.

3 Post-Ensemble

Our alternative ensemble method combines multiple text-generation models by selecting a majority-like output close to the other outputs, which is calculated with a similarity function such as cosine similarity. We call this method *post-ensemble* since it is executed in post-processing, i.e., after a decoding process. Fig. 1(b) shows a flow chart of post-ensemble, and Alg. 1 shows its algorithm. When our method receives an input x, a normal decoder calculates the output s of each model p from the input in parallel (lines 2–4), and the output selector selects the majority-like output y from all outputs (lines 6–9). In line 7, we calculate the score c of each output s by using a similarity function K, where $K(s, s')$ represents the similarity between s and s'. A higher score means that the output s is in a denser part in the output space since the score c means the average similarity in other outputs.

```
   Input: Input text $x$, set $M$ of learned models, and
          similarity function $K$, such as cos.
   Output: Output prediction $y$.
1  $S \leftarrow \emptyset$;
2  foreach $p \in M$ do
3      $s \leftarrow$ output of model $p$ for input $x$;
4      $S \leftarrow S \cup \{s\}$;
5  $C \leftarrow \{\}$; // as a hash map
6  foreach $s \in S$ do
7      $c \leftarrow \frac{1}{|S|} \sum_{s' \in S} K(s, s')$;
8      $C[s] \leftarrow c$;
9  $y = \mathrm{argmax}_{s \in S}\, C[s]$;
10 return $y$
```

Algorithm 1: Post-ensemble procedure.

The post-ensemble procedure has two main advantages compared with the current runtime-ensemble procedure. One is that we do not need to develop an ensemble decoder by modifying a decoding program on a deep learning framework. The concept of runtime-ensemble is simple, but its implementation is not that simple in recent sophisticated open source software. For example, we need to modify about 100 lines to add an ensemble feature to the decoding program of an open source neural machine translator, OpenNMT[1], which requires understanding the overall mechanism of the software. The other advantage is that we can easily parallelize decoding processes in our method since each output can be calculated by using a single model. If we have a server program for text generation, we can improve its performance with all our machine resources (ideally) by assigning a server to each model and allowing the output selector to communicate with it.

One drawback of our method is that its expressive power is basically the same as that of each single model. However, this alternatively means that the lower bound of the quality of each output is guaranteed with the worst case of the outputs of single models, while the current runtime-ensemble method can perform worse than each single model for the worst case input. Furthermore, experiments showed our post-ensemble method is more effective than the current runtime-ensemble methods.

4 Theoretical Analysis

In this section, we prove that when $K(s, s') = \cos(s, s')$, Alg. 1 is an approximation of find-

---

[1] https://github.com/OpenNMT/OpenNMT-py

ing the maximum density point by kernel density estimation based on the von Mises-Fisher kernel. First, we briefly explain kernel density estimation and how to apply it to our method in Sec. 4.1. Then, we introduce the von Mises-Fisher kernel used in this analysis and later experiments in Sec. 4.2. Finally, we prove a theorem that guarantees the approximation error in Sec. 4.3.

4.1 Kernel Density Estimation

Kernel density estimation is a non-parametric method for estimating the probability density function of a random variable. Let $(X_1, \cdots, X_n)$ be an independent and identically distributed (i.i.d.) sample that was drawn from a distribution with an unknown density function f. The kernel density estimator based on the sample is defined as

$$\tilde{f}(X) = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} K(X, X_i). \tag{4}$$

Using an appropriate kernel such as the Gaussian kernel, this estimator $\tilde{f}$ converges to the true density f, and it can be proved that there is no non-parametric estimator that converges faster than this kernel density estimator (Wahba, 1975).

Here, let us consider our outputs $(s_1, \cdots, s_n)$, which correspond to S in Alg. 1. They are generated from text generation models $(p_1, \cdots, p_n)$, which correspond to M in Alg. 1. We assume that these models are trained with randomly initialized parameters $(\theta_1, \cdots, \theta_n)$, each of which includes a random seed for the optimizer, and the other settings are deterministic. In this case, we can construct a function $F : P \to O$ that maps the parameter space P onto the output space O. In other words, if each parameter θ_i is an i.i.d. random variable, the corresponding output $s_i = F(\theta_i)$ is also an i.i.d. random variable. Therefore, Eq. (4) can be directly used for line 7 in Alg. 1.

Our method can be regarded as a heuristic approach based on the characteristics of our encoder-decoder model, where there are many local solutions for optimization. We expect that our method can be applied to other models on the basis of a theoretical study (Kawaguchi, 2016), that showed that deep neural networks can have many local optima, but there are no poor local optima (formally, every local minimum of deep neural networks is a global minimum under a certain condition). We do not consider this direction since theoretical justification is beyond our scope.

4.2 von Mises-Fisher Kernel

The von Mises-Fisher kernel (Hall et al., 1987) is a natural extension of the Gaussian kernel to a unit hypersphere. This kernel is especially useful for directional or angular statistics, so it is expected to be compatible with the cosine similarity frequently used in natural language processing. The definition is

$$K_{\text{vmf}}(s, s') = C_q(\kappa) \exp(\kappa \cos(s, s')), \tag{5}$$

where κ is a smoothing factor called the *concentration parameter*, and cos is a cosine similarity, i.e., $\cos(s, s') = \frac{s \cdot s'}{||s||_2 ||s'||_2}$. $C_q(\kappa)$ is the normalization constant, which is defined as

$$C_q(\kappa) = \left(\kappa^{\frac{q-1}{2}}\right) \Big/ \left((2\pi)^{\frac{q+1}{2}} \mathcal{I}_{\frac{q-1}{2}}(\kappa)\right), \tag{6}$$

where $\mathcal{I}_v$ is the modified Bessel function of the first kind at order v, and q is the dimension of directional data (angular expression of data).

In the experiments described later, we implemented Alg. 1 with this kernel by using the log-sum-exp trick (Nielsen and Sun, 2016) to avoid overflow/underflow problems since $\text{argmax} \sum \exp(x) = \text{argmax} \log \sum \exp(x)$. In addition, we used Garcia-Portugues's rule (Garcia-Portugues, 2013) to adjust the concentration parameter $\kappa = \hat{h}^{-2}$, defined as

$$\hat{h} = \left(\frac{4\pi^{\frac{1}{2}} \mathcal{I}_{\frac{q-1}{2}}(\tilde{\kappa})^2}{\tilde{\kappa}^{\frac{q+1}{2}} \left(2q \mathcal{I}_{\frac{q+1}{2}}(2\tilde{\kappa}) + (2+q)\tilde{\kappa} \mathcal{I}_{\frac{q+3}{2}}(2\tilde{\kappa}) \right) n} \right)^{\frac{1}{4+q}} \tag{7}$$

where $\tilde{\kappa}$ is an approximation of κ derived from the maximum likelihood estimation (Sra, 2012), defined as $\tilde{\kappa} = \frac{\tilde{\mu}(q-\tilde{\mu})}{1-\tilde{\mu}^2}$, where $\tilde{\mu}$ is the sample mean of the directional data in a unit hypersphere.

4.3 Approximation Error Analysis

We prove an approximation error bound of Alg. 1 when $K(s, s') = \cos(s, s')$, as shown in the following theorem.

Theorem 1. The output y of Alg. 1 with $K(s, s') = \cos(s, s')$ is equivalent to the maximization of the first order Taylor series approximation $\tilde{p}$ of the kernel density estimator p based on the von Mises-Fisher kernel, i.e.,

$$\tilde{p}(y) = \max\nolimits_{s \in S} \tilde{p}(s), \tag{8}$$

where the approximation error R^* of the output y with respect to the true density estimator p, i.e,. $R^* = \max_{s \in S} p(s) - p(y)$, is bounded by

$$R^* \leq C_q(\kappa) \kappa^2 \exp(\kappa) (\sigma^2 + \mu^2), \tag{9}$$

where $\mu = \max_{s \in S} \mathbb{E}_{s'}[\cos(s, s')]$, and $\sigma^2 = \max_{s \in S} \mathbb{V}_{s'}[\cos(s, s')]$.

Proof sketch. Eq. (8) can be obtained by using the first order Taylor series approximation at 0 of $\exp(x)$, i.e., $\exp(x) \approx 1 + x$, and the nature of argmax, i,e., $\operatorname{argmax}(1 + \kappa x) = \operatorname{argmax} x$. Eq. (9) can be derived by the Lagrange error bound $\tilde{R}(x)$ for $\exp(x) \approx 1+x$, where $x = \kappa \cos(s, s')$, and $-\kappa \le x \le \kappa$, as

$$\tilde{R}(x) = \frac{\max_{x'} \exp(x')}{2!} x^2 \le \frac{\exp(\kappa)}{2} x^2. \quad (10)$$

See Appendix A for the complete proof. ☐

This theorem implies that the approximation error becomes smaller as κ becomes smaller. Since κ is the concentration parameter, the shape of the density estimation will be smooth when κ is small, while it will be a peak when κ is large. This means that, when κ is large, the density estimation is almost the same as the majority vote. Therefore, we can naturally choose a small value for κ for our purpose. In fact, the concentration parameter was set as $\kappa = 0.69$ by using Garcia-Portugues's rule in our experiments. The normalization constant using κ was calculated as $C_q(\kappa) = 0.14$, and the average values of μ and σ with respect to the set S of output candidates were $\mathbb{E}_S[\sigma] = 0.30$ and $\mathbb{E}_S[\mu] = 0.78$, respectively. In this case, the theoretical average approximation error was calculated as $\mathbb{E}_S[R^*] \le 0.093 = 0.14 \times 0.69^2 \times \exp(0.69) \times (0.78^2 + 0.30^2)$. This is quite small in view of the approximation error for a probability. In addition, the actual average approximation error can be much smaller, and it was about 1.95×10^{-7} in our experiments. The accuracy defined by the rate at which the approximate maximum is the true maximum, i.e., $p(y) = \max_{s \in S} p(s)$, was 96.36%. Detail on the settings of our experiments will be given in the next section.

5 Experiments

We first explain the basic settings of our experiments in Sec. 5.1 and report a comparative experiment and analysis on the news-headline-generation task in Sec. 5.2. Then, we discuss the change in some of the settings to conduct an experiment by changing the number of models and the settings of model preparation in Sec. 5.3 and Sec. 5.4, respectively.

5.1 Basic Settings

Dataset: We used a well-known dataset `Gigaword` of a news-headline-generation task, which was prepared by Rush et al. (2015). This dataset has been extensively used in recent studies on abstractive summarization (Takase et al., 2016; Chopra et al., 2016; Kiyono et al., 2017; Zhou et al., 2017; Suzuki and Nagata, 2017; Cao et al., 2018). The `Gigaword` dataset was created from the English Gigaword corpus[2], in which the input is the first sentence in a news article, and the output is the headline of the article. The training, validation, and test sets included 3.8M, 189K, and 2K sentences, respectively. The preprocessed data are publicly available[3]. The dataset is also used to train official pretrained models of OpenNMT[4].

Model and Training: We basically used the default PyTorch implementation of OpenNMT[5] on June 11, 2017 throughout our experiments, but the unidirectional long short-term memory (LSTM) for the encoder was replaced with a bidirectional one to obtain nearly state-of-the-art results. The basic settings are as follows. Our model consisted of a bidirectional LSTM for the encoder and a stacked LSTM with input feeding for the decoder. These LSTMs had two layers with 500-dimensional hidden layers whose dropout rates were 0.3, and their input vectors were created by a 500-dimensional word-embedding layer.

The model was trained with a stochastic gradient descent method with a learning rate of 1.0, where the mini-batch size was set to 64. The learning process ended in 13 epochs, decaying the learning rate with a decay factor of 0.5 in each epoch after 8 epochs. These training settings are the same as the training of the official pretrained models of OpenNMT, and we confirmed that these settings performed better than training with Adam (Kingma and Ba, 2014) in our preliminary experiments. We prepared 10 learned models by random initialization for the ensemble methods in our experiments.

Decoding and Evaluation: When decoding input sequences, we used a beam-search algorithm with a beam width of 5. The maximum size of decoded sequences was 100. The generated unknown token <unk> was replaced by the source word with the highest attention weight.

To evaluate decoded sequences, we calculated ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004), mainly used in the headline-generation-task (Rush et al., 2015). ROUGE-1 and ROUGE-

---

[2] https://catalog.ldc.upenn.edu/LDC2012T21

[3] https://github.com/harvardnlp/sent-summary

[4] http://opennmt.net/Models/

[5] SHA: c13a558767cbc19b612968eb4d01a1f26d5df688

2 are the co-occurrence rates of unigrams and bigrams, respectively, between a generated headline and its reference. ROUGE-L is the rate of the longest common subsequence between them to the reference length. We used a Python wrapper of the ROUGE-1.5.5.pl script[6] and took the average value of 10 times, each of which used 10 models.

Compared Methods: We compared the following methods. `Single` is a baseline with a single model. `EnsSum` and `EnsMul` are strong baselines with runtime-ensemble. `MaxLik` and `MajVote` are weak baselines with naive post-processing. `LexRank` and `LMRank` are simple unsupervised methods from two other related tasks, extractive summarization and hypotheses reranking, respectively. `PostCosE` and `PostCosB` are variants of the proposed method with post-ensemble. `PostVmfE` and `PostVmfB` are true density estimators corresponding to `PostCosE` and `PostCosB`, respectively. Their descriptions are listed below in detail.

- `Single` decodes an output by using the best single model with respect to the word level accuracy on a validation set.
- `EnsSum` and `EnsMul` decode an output averaging multiple models with Eq. (2) and Eq. (3), respectively.
- `MaxLik` selects an output with the maximum likelihood, which is calculated by the corresponding model p in Alg. 1, from candidate outputs generated by multiple models.
- `MajVote` selects an output by majority vote based on exact matching, i.e., $y = \mathrm{argmax}_{s \in S} |\{s' \in S \mid s = s'\}|$.
- `LexRank` selects an output with the LexRank algorithm (Erkan and Radev, 2004). We used a Python implementation[7], where a graph is constructed on the basis of cosine similarities between the tf-idf vectors (without stop-words) of candidate outputs. The idf weights are calculated from the training set.
- `LMRank` selects an output that maximizes the likelihood of a (non-conditional) language model p_{LM}, i.e., $y = \mathrm{argmax}_{s \in S}\, p_{\mathrm{LM}}(s)$, as in (Vaswani et al., 2013). We used the decoder part of the encoder-decoder model described in Sec. 5.1, which was trained with both source and target sentences in the training set. This allows this model to learn the fluency in both normal and headline-like sentences.
- `PostCosE` and `PostVmfE` select an output on the basis of Alg. 1 with the cosine similarity i.e., $K(s, s') = \cos(s, s')$, and the von Mises-Fisher kernel, i.e., $K(s, s') = K_{\mathrm{vmf}}(s, s')$ in Eq. (5), respectively. The feature of each output is the average of pretrained 300-dimensional word embeddings[8].
- `PostCosB` and `PostVmfB` are variants of `PostCosE` and `PostVmfE` with simple bag-of-words features (sparse vectors), respectively.

In addition, we used the following measurements for analysis. `MaxRef` represents the upper bound for the performance of our method. `Mean`, `Max`, and `Min` represent the performance statistics of the single models.

- `MaxRef` selects the best output with respect to ROUGE-1, which is calculated by using the references in the test set.
- `Mean`, `Max`, and `Min` are the mean, maximum, and minimum of the (non-ensemble) ROUGE-1 values for the 10 models, respectively. The difference between `Single` and `Max` is that the former uses the validation set, while the latter uses the test set.

5.2 Main Results

We conducted a comparative experiment on the news-headline-generation task to verify the effectiveness of our post-ensemble method compared with the current runtime-ensemble methods.

Tab. 1 shows the experimental results for the `Gigaword` dataset, including the results of our method with 32 models and other previous results. First of all, we can see that the variant of our post-ensemble method, `PostCosB`, clearly outperformed the runtime-ensemble methods (strong baselines), `EnsSum` and `EnsMul`, and the other baselines. The differences between our best method `PostCosB` and the best baseline `EnsSum` were all statistically significant on the basis of a one-tailed, paired t-test ($p < 0.05$). Comparing with the recent results of Cao et al. (2018) obtained with open information extraction and dependency parse technologies and the other previous results[9], our method with 32 models also performed better, although the algorithm of our

---

[6] https://github.com/pltrdy/files2rouge

[7] https://github.com/wikibusiness/lexrank

[8] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

[9] We did not present the results of Raffel et al. (2017) since Kiyono et al. (2017) pointed out that their settings are different from the previous studies.

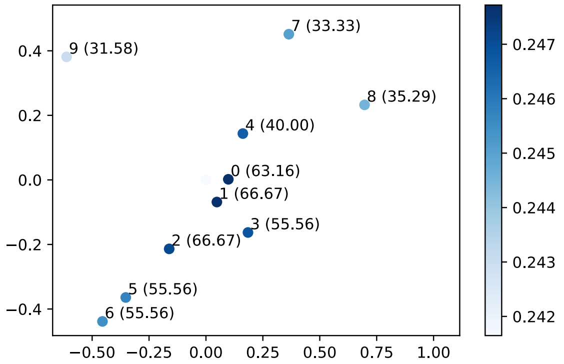

0: **interpol asks** members **to** devise **rules for policing** ...
1: **interpol asks** members **to** devise **rules for policing**
2: **interpol asks** members **to** devise **rules for policing** at ...
3: **interpol asks** members **to** devise **rules** on **policing**
4: **interpol asks** members **to** devise **rules** and procedures ...
5: **interpol** seeks **rules for policing** of **global** level
6: **interpol** seeks **rules for policing** at **global** level
7: **interpol asks** members **to** act against wanted fugitives
8: **interpol asks** members **to** help fight fugitives
9: **interpol asks for** legal status **for** red corner notices

Figure 2: Left scatter-plot shows two-dimensional visualization of outputs generated from 10 models on basis of multi-dimensional scaling (Cox and Cox, 2008), and right list shows their contents. Each point in plot represents sentence embedding of corresponding output, and label indicates model ID and ROUGE-1, i.e., "ID (ROUGE)." Color intensity means score of kernel density estimation of `PostCosE` (see right color bar), and outputs are sorted by scores. Reference and input are as follows. Each bold word in above list means co-occurrence with reference below.

Reference: *interpol asks world govts to make rules for global policing*

Input: *top interpol officers on wednesday asked its members to devise rules and procedures for policing at the global level and providing legal status to red corner notices against wanted fugitives .*

	R-1	R-2	R-L
`Single`	35.57	17.47	33.19
`EnsSum`	36.55	18.48	34.24
`EnsMul`	36.47	18.35	34.16
`MaxLik`	35.04	17.37	32.80
`MajVote`	35.97	18.09	33.67
`LexRank`	36.03	17.64	33.60
`LMRank`	35.07	17.12	32.61
`PostCosE`	37.02	18.46	34.54
`PostVmfE`	**37.06**	18.53	34.60
`PostCosB`	37.05	**18.59**	**34.61**
`PostVmfB`	37.02	18.58	34.59
`MaxRef`*	45.40	24.61	42.09
`Mean`*	35.57	17.48	33.19
`Max`*	36.03	17.83	33.63
`Min`*	35.00	17.08	32.67
`PostCosE` (32 models)	37.52	18.55	34.86
`PostCosB` (32 models)	37.48	18.76	34.99
(Rush et al., 2015)♯	31.00	12.65	28.34
(Takase et al., 2016)♯	31.64	12.94	28.54
(Chopra et al., 2016)♯	33.78	15.96	31.15
(Kiyono et al., 2017)♯	35.79	17.84	33.34
(Zhou et al., 2017)♯	36.15	17.54	33.63
(Suzuki and Nagata, 2017)♯	36.30	17.31	33.88
(Cao et al., 2018)♯	37.27	17.65	34.24

Table 1: F-measure ROUGE-1, ROUGE-2, and ROUGE-L scores (%) for news-headline-generation task. Bold and underlined scores represent best scores for ensembles of 10 models and all methods excluding measurements with "$*$," respectively. Results with "♯" are taken from corresponding papers.

method is quite simple. Note that our method can be easily applied to their models to improve their results. Looking at the row for `MaxRef`, the results imply that our post-ensemble method still has room for improvement without any changes to model structure. Although we also conducted an experiment by changing the settings of model preparation, the results had a similar tendency to those of the main results (see Sec. 5.4).

Fig. 2 illustrates how our method worked with kernel density estimation (see the figure caption for detailed descriptions). The left scatter-plot shows a two-dimensional visualization of 10 outputs generated from the 10 models and the estimated densities (represented by color intensity in the right bar). Looking at the center part of the plot, we can see that there are many good outputs with high ROUGE-1 results (noted in brackets in the plot) in the dense part. The right list shows the corresponding outputs of the points in the left plot, where these outputs are sorted by the estimated density. The list shows that our method successfully obtained the majority-like output (model ID of 0) in the dense part of the output space, although there are no exact match outputs. Looking at the bottom part of the list, we can see that our method clearly eliminated unpromising outputs (model ID of 7, 8, and 9) with less information, since they are scattered.

5.3 Effect of Number of Models

We compared the effect of changing the number of models on the performance of our best method `PostCosB` and the best baseline `EnsSum`. We pre-

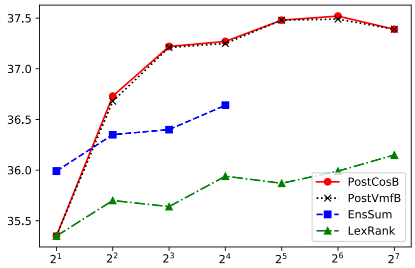

Figure 3: F-measure ROUGE-1 performance (%) vs. number of models for news-headline-generation task. X-axis is log scale (2^1–2^7).

pared 128 models, in which each training took more than two days. The ROUGE-1 performance was measured by varying the number of models, i.e., 2, 4, $\cdots$, 128.

Fig. 3 shows the performance of our best method `PostCosB`, the corresponding true estimator `PostVmfB`, the best baseline `EnsMul`, and the most widely-used baseline `LexRank` versus the number of models. Note that we could not calculate the results of `EnsMul` for more than 16 models due to out of memory errors. The figure shows that `PostCosB` performed better than `EnsMul` even for these 16 models. We obtained a 37.48 ROUGE-1 score with 32 models, which was better than the state-of-the-art results in Tab. 1, but the performance seems to be saturated with more than 32 models. Looking at `PostCosB` and `PostVmfB`, we can see that the performances are almost the same, which also supports our theoretical analysis in Sec. 4. `LexRank` did not work well even though the number of models was large.

The complexity of the post-ensemble procedure in Alg. 1 is $O(\beta\nu+\delta\nu^2)$, where ν is the number of models, δ is the dimension of the output space, and β is the number of operations of the beam-search. We can reduce it to $O(\beta+\delta\nu)$ by simply parallelizing lines 2–4 and 6–8 in Alg. 1 without any change to the model code on the deep learning framework. Since the operations of β includes all matrix calculations in the model, we can basically assume $\beta \gg \delta\nu$. In fact, the actual calculation times of `PostCosE` and `PostCosB` with a naive implementation in Python were 0.0097 and 0.0037 seconds per sentence when $\nu = 32$, respectively. They are enough for practical use in comparison with the decoding speeds of 0.044 (GPU) and 0.49 (CPU) seconds per sentence. In addition, the complexity of the runtime-ensemble is $O(\beta\nu)$, which cannot be parallelized without modifying more than a hundred lines of code after understanding a whole system.

	Random	Self	Hetero	Bagging
`Single`	35.57	35.34	35.67	34.87
`EnsSum`	36.55	35.46	36.42	36.25
`EnsMul`	36.47	35.22	36.49	35.80
`MaxLik`	35.04	34.21	34.86	34.95
`MajVote`	35.97	35.49	35.89	35.22
`LexRank`	36.03	33.57	35.91	35.72
`LMRank`	35.07	33.47	34.71	34.39
`PostCosE`	37.02	**35.91**	36.57	**36.89**
`PostVmfE`	**37.06**	35.72	36.69	36.84
`PostCosB`	37.05	35.74	**36.76**	36.78
`PostVmfB`	37.02	35.75	36.75	36.81
`MaxRef`*	45.40	43.37	45.32	46.44
`Mean`*	35.57	34.43	35.28	34.85
`Max`*	36.03	35.34	35.96	35.31
`Min`*	35.00	33.43	34.49	34.36

Table 2: F-measure ROUGE-1 scores (%) of random-ensemble (Random), self-ensemble (Self), hetero-ensemble (Hetero), and bagging-ensemble (Bagging) for news-headline-generation task. Bold scores represent best scores for all methods excluding measurements with “$*$.”

5.4 Effect of Model Preparation

We conducted experiments to verify the effect of changing the model preparation on post-ensemble performance. In addition to random initialization (*random-ensemble*), we address three variations of model preparation: *self-ensemble*, *hetero-ensemble*, and *bagging-ensemble*. The first one, self-ensemble, is a method of extracting models from “checkpoints” saved in each epoch in a training. We prepared the models of self-ensemble by using 10 checkpoints from 4–13 epochs. The second one, hetero-ensemble, is a method of training models varying in model structure. We prepared 10 models for hetero-ensemble, consisting of 8 models prepared by changing the number of layers in the LSTM encoder/decoder in $\{2, 3\}$, the size of LSTM hidden states in $\{250, 500\}$, and the size of word embedding in $\{250, 500\}$, and two models prepared by replacing the bidirectional encoder with a unidirectional encoder and a bidirectional encoder with a different merge action, i.e., summation instead of concatenation. The third one, bagging-ensemble, is a method of training models by bagging of training data. We randomly extracted 80% of the training data 10 times and prepared 10 models for bagging-ensemble. We used the same dictionary of the original data for these models, since the runtime-ensemble methods, `EnsSum` and `EnsMul`, failed to average the models with different dictionaries.

Note that the outputs for self-ensemble and hetero-ensemble cannot be regarded as i.i.d samples, but we believe the basic idea can be practically applied.

Tab. 2 shows the F-measure ROUGE-1 scores for the `Gigaword` dataset of the above three variations, self-, hetero-, and bagging-ensembles, as well as random-ensemble. The table indicates that all variants of our post-ensemble method performed better than the current runtime-ensemble methods, `EnsSum` and `EnsMul`, for all variations of model preparation. Looking at the row for `PostCosE`, random-ensemble was the most effective, while self-ensemble was the worst, as expected. Bagging-ensemble was relatively effective for post-ensemble according to the relative improvement from `Single`, despite the fact that we trained the models with 80% of the training data. Hetero-ensemble performed worse than random-ensemble for these settings, but we expect that if the model structure can be randomly chosen, hetero-ensemble will perform better.

6 Related Work

Distillation techniques for an ensemble of multiple models have been widely studied (Kuncoro et al., 2016; Chebotar and Waters, 2016; Kim and Rush, 2016; Freitag et al., 2017; Stahlberg and Byrne, 2017), especially after a study by Hinton et al. (2015). Kuncoro et al. (2016) and Chebotar and Waters (2016) studied distillation techniques for ensembles of multiple dependency parsers and speech recognition models, respectively. There are several ensemble methods for ensembles of machine translation models (Kim and Rush, 2016; Freitag et al., 2017; Stahlberg and Byrne, 2017). For example, Stahlberg and Byrne (2017) proposed a method of unfolding an ensemble of multiple translation models into a single large model once and shrinking it down to a small one. However, all methods require extra implementation on a deep-learning framework, and it is not easy to apply them to other models. Our post-ensemble method does not require such coding skills. In addition, since the predictions of post-ensemble can be regarded as a teacher model, these distillation techniques should be combined with a teacher model based on post-ensemble.

Hypotheses reranking of language generation has been extensively studied, but most studies focused on discriminative training using costly annotated data (Shen et al., 2004; White and Rajkumar, 2009; Duh et al., 2010; Kim and Mooney, 2013; Mizumoto and Matsumoto, 2016). The main stream of our focused unsupervised approach was a reranking method based on a language model (Chen et al., 2006; Vaswani et al., 2013; Luong and Popescu-Belis, 2016), and other approaches include reranking methods based on key phrase extraction (Boudin and Morin, 2013), dependency analysis (Hasan et al., 2010), and search results (Peng et al., 2013). All of the above described studies were not used for model ensemble. Tomeh et al. (2013) used an ensemble learning, but the purpose was to improve the performance of the reranking model for hypotheses reranking of a single model. Li et al. (2009), which work is the most related one, proposed a reranking algorithm for model ensemble. However, their method was constructed to perform at decoding time, so it can be regarded as runtime-ensemble.

The term "frustratingly easy" in this paper is borrowed from "frustratingly easy" papers (Daumé III, 2007; Daumé III et al., 2010; Tommasi and Caputo, 2013; Sun et al., 2016; Kim et al., 2016).

7 Conclusion

We proposed a simple but effective model-ensemble method, called *post-ensemble*, for abstractive-summarization models, i.e., encoder-decoder models. We verified the effectiveness of our method on the news-headline-generation task.

We will release the 128 prepared models used in this paper[10], each of which was trained for more than two days, as a new dataset for improving ensemble methods. For example, future research includes applying learning-to-rank regarding all outputs as features, conducting active learning to select a new model setting online, and developing boosting-like-ensemble based on the bagging of training data.

Acknowledgments

The author would like to thank the anonymous reviewers of all versions of this paper for sparing their valuable time and leaving many insightful comments.

---

[10] https://research-lab.yahoo.co.jp/en/software/

References

Ayana, Shi-Qi Shen, Yan-Kai Lin, Cun-Chao Tu, Yu Zhao, Zhi-Yuan Liu, and Mao-Song Sun. 2017. Recent Advances on Neural Headline Generation. *Journal of Computer Science and Technology*, 32(4):768–784.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Florian Boudin and Emmanuel Morin. 2013. Keyphrase Extraction for N-best Reranking in Multi-Sentence Compression. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 298–305. Association for Computational Linguistics.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the Original: Fact Aware Neural Abstractive Summarization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, pages 4784–4791. AAAI Press.

Yevgen Chebotar and Austin Waters. 2016. Distilling Knowledge from Ensembles of Neural Networks for Speech Recognition. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association*, pages 3439–3443.

Yi Chen, Ming Zhou, and Shilong Wang. 2006. Reranking Answers for Definitional QA Using Language Modeling. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 1081–1088. Association for Computational Linguistics.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-Fine Question Answering for Long Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 209–220. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 93–98. Association for Computational Linguistics.

Michael A. A. Cox and Trevor F. Cox. 2008. *Multidimensional Scaling*. Springer Berlin Heidelberg.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2017)*, pages 256–263. Association for Computational Linguistics.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly Easy Semi-Supervised Domain Adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59. Association for Computational Linguistics.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 484–495. Association for Computational Linguistics.

Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, Hideki Isozaki, and Masaaki Nagata. 2010. N-best reranking by multitask learning. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 375–383. Association for Computational Linguistics.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *Journal of Artificial Intelligence Research (JAIR)*, 22(1):457–479.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble Distillation for Neural Machine Translation. *CoRR*, abs/1702.01802.

Eduardo Garcia-Portugues. 2013. Exact risk improvement of bandwidth selectors for kernel density estimation with directional data. *Electronic Journal of Statistics*, 7:1655–1685.

Peter Hall, G. S. Watson, and Javier Cabrera. 1987. Kernel Density Estimation with Spherical Data. *Biometrika*, 74(4):751–762.

Sasa Hasan, Oliver Bender, and Hermann Ney. 2010. Reranking Translation Hypotheses Using Structural Properties. In *Proceedings of the EACL'06 Workshop on Learning Structured Information in Natural Language Applications*, pages 41–48. Association for Computational Linguistics.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR*, abs/1503.02531.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernand a Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Kenji Kawaguchi. 2016. Deep Learning without Poor Local Minima. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 586–594. Curran Associates, Inc.

Joohyun Kim and Raymond Mooney. 2013. Adapting Discriminative Reranking to Grounded Language Learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 218–227. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1317–1327. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly Easy Neural Domain Adaptation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 387–396. The COLING 2016 Organizing Committee.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Shun Kiyono, Sho Takase, Jun Suzuki, Naoaki Okazaki, Kentaro Inui, and Masaaki Nagata. 2017. Source-side Prediction for Neural Headline Generation . *CoRR*, abs/1712.08302.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1744–1753. Association for Computational Linguistics.

Mu Li, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009. Collaborative Decoding: Partial Hypothesis Re-ranking Using Translation Consensus between Decoders. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP 2009)*, pages 585–592. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the ACL Workshop on Text Summarization Branches Out*.

Nick Littlestone and Manfred K. Warmuth. 1994. The Weighted Majority Algorithm. *Information and Computation*, 108(2):212–261.

Ngoc Quang Luong and Andrei Popescu-Belis. 2016. A Contextual Language Model to Improve Machine Translation of Pronouns by Re-ranking Translation Hypotheses. *Baltic Journal of Modern Computing*, 4(2):292–304.

Tomoya Mizumoto and Yuji Matsumoto. 2016. Discriminative reranking for grammatical error correction with statistical machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 1133–1138. Association for Computational Linguistics.

Frank Nielsen and Ke Sun. 2016. Guaranteed Bounds on Information-Theoretic Measures of Univariate Mixtures Using Piecewise Log-Sum-Exp Inequalities. *Entropy*, 18(12).

F. Peng, S. Roy, B. Shahshahani, and F. Beaufays. 2013. Search results based N-best hypothesis rescoring with maximum entropy classification. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 422–427.

Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and Linear-Time Attention by Enforcing Monotonic Alignments. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, pages 2837–2846.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389. Association for Computational Linguistics.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative Reranking for Machine Translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004)*, pages 177–184. Association for Computational Linguistics.

Suvrit Sra. 2012. A Short Note on Parameter Approximation for Von Mises-Fisher Distributions: And a Fast Implementation of I s (x). *Computational Statistics*, 27(1):177–190.

Felix Stahlberg and Bill Byrne. 2017. Unfolding and Shrinking Neural Machine Translation Ensembles. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 1946–1956. Association for Computational Linguistics.

Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of Frustratingly Easy Domain Adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, pages 2058–2065. AAAI Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112. Curran Associates, Inc.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 291–297. Association for Computational Linguistics.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural Headline Generation on Abstract Meaning Representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1054–1059. Association for Computational Linguistics.

Nadi Tomeh, Nizar Habash, Ryan Roth, Noura Farra, Pradeep Dasigi, and Mona Diab. 2013. Reranking with Linguistic and Semantic Features for Arabic Optical Character Recognition. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 549–555. Association for Computational Linguistics.

Tatiana Tommasi and Barbara Caputo. 2013. Frustratingly Easy NBNN Domain Adaptation. In *IEEE International Conference on Computer Vision (ICCV 2013)*, pages 897–904.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-Scale Neural Language Models Improves Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1387–1392. Association for Computational Linguistics.

Grace Wahba. 1975. Optimal Convergence Properties of Variable Knot, Kernel, and Orthogonal Series Methods for Density Estimation. *The Annals of Statistics*, 3(1):15–29.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron Reranking for CCG Realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 410–419. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1095–1104. Association for Computational Linguistics.

Automatic Pyramid Evaluation Exploiting EDU-based Extractive Reference Summaries

Tsutomu Hirao[1] and **Hidetaka Kamigaito**[2] and **Masaaki Nagata**[1]
[1]NTT Communication Science Laboratories, NTT Corporation
[2]Institute of Innovative Research, Tokyo Institute of Technology
[1]`{hirao.tsutomu,nagata.masaaki}@lab.ntt.co.jp`
[2]`kamigaito@lr.pi.titech.ac.jp`

Abstract

This paper tackles automation of the pyramid method, a reliable manual evaluation framework. To construct a pyramid, we transform human-made reference summaries into extractive reference summaries that consist of Elementary Discourse Units (EDUs) obtained from source documents and then weight every EDU by counting the number of extractive reference summaries that contain the EDU. A summary is scored by the correspondences between EDUs in the summary and those in the pyramid. Experiments on DUC and TAC data sets show that our methods strongly correlate with various manual evaluations.

1 Introduction

To develop high quality summarization systems, we need accurate automatic content evaluation. Although, various evaluation measures have been proposed, ROUGE-N (Lin, 2004), Basic Elements (BE) (Hovy et al., 2006) remain the de facto standard measures since they strongly correlate with various manual evaluations and are easy to use. However, the evaluation scores computed by these automatic measures are not so useful for improving system performance because they merely confirm if the summary contains small textual fragments and so they do not address semantic correctness.

The pyramid method was proposed as a manual evaluation that well supports the improvement of summarization systems (Nenkova and Passonneau, 2004; Nenkova et al., 2007). First, the method identifies conceptual contents, Summary Content Units (SCUs), in reference summaries and then constructs a pyramid by collecting semantically equivalent SCUs. The weight of an SCU in the pyramid is defined as the number of reference summaries that contain the SCU. Thus, an SCU shared by many reference summaries is given higher weight. Second, a system summary is scored by the correspondences between SCUs in the summary and the pyramid. Its results are very useful for system improvement, *i.e.*, we can know which important SCUs the system could or could not include in the summary. Although the pyramid method is reliable, it requires considerable cost and effort.

To address the weaknesses, automatic pyramid evaluation, Pyramid Evaluation via Automated Knowledge Extraction (PEAK) was proposed (Yang et al., 2016). Since SCU is the conceptual content of the text, it is difficult to automatically extract them from reference summaries by systems. Thus, PEAK regards subject-predicate-object triples as alternatives to SCUs and constructs a pyramid by clustering semantically equivalent triples. However, the performance of subject-predicate-object triples extraction is not satisfying for the practical demands and semantic similarity utilized for clustering the triples does not correlate well with human judgment (see Section 2). As a result, the resultant pyramid is unreliable. Actually, PEAK is significantly inferior to ROUGE and BE (see Section 4.3) in terms of correlation.

To cope with the above problems, this paper proposes yet another automatic pyramid evaluation method. Its key feature is constructing a pyramid that consists of Elementary Discourse Units (EDUs), clause-like text units introduced in Rhetorical Structure Theory (Mann, William Charles and Thompson, Sandra Annear, 1988), in the source documents. In other words, we regard EDUs as alternatives to SCUs. To construct the pyramid, we transform human-made reference summaries into EDU-based extractive reference summaries and then weight every EDU by counting the number of the extractive reference summaries that contain the EDU. The rea-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4177–4186
Brussels, Belgium, October 31 - November 4, 2018.

son why we derive extractive reference summaries whose SCUs are EDUs is as follows. First, Li et al. (2016) reported that EDUs are very similar to SCUs. Second, the performance of EDU segmenter is sufficient to satisfy practical requirements (see Section 2). Third, we do not need measure any semantic similarity to identify EDUs common to the extractive reference summaries. We also examine two types of extractive reference summary. One is based on the alignment between EDUs in reference summary and source documents. The other is based on the *extractive oracle summary* (Hirao et al., 2017). We conducted experiments on the Document Understanding Conference (DUC) 2003 to 2007 data sets and Text Analysis Conference (TAC) 2008 to 2011 data sets. The results showed that our methods exhibit strong correlation with manual evaluations.

2 Background and Related Work

The pyramid method (Nenkova and Passonneau, 2004; Nenkova et al., 2007), a manual evaluation framework, was developed to measure the content coverage of summaries. The pyramid method consists of two steps: (1) pyramid construction, and (2) summary scoring based on the pyramid. First, human annotators identify Summary Content Units (SCUs), conceptual content units in the reference summaries. They then construct a pyramid by clustering and weighting SCUs. The weight of an SCU is defined as the number of reference summaries that contain the SCU. As a result, if there are K reference summaries, the upper bound weight of an SCU in the pyramid is K and the lower bound is 1. Second, the score for a summary is determined by the correspondences between SCUs in the summary and those in the pyramid. Thus, the score is defined as the sum of weights of SCUs that correspond to those in the pyramid in the summary divided by the sum of SCU weight possible for an average-length reference summary. The pyramid method has two advantages over conventional manual evaluations: (1) the score is not intuitive but is systematically computed, *i.e.*, the score can be explained as the sum of weights of SCUs in the pyramid, (2) the correspondences between the SCUs in a summary and the pyramid tell us whether the summary contains important SCUs or not. Thus, the results explicitly tell us why a summary was given a good or bad score.

During the past few years, studies have focused on the automatic scoring of summaries based on manually generated pyramids. Harnly et al. (2005) proposed a scoring method that matches SCUs in the pyramid with possible textual fragments in the summary. They enumerate all possible textual fragments within a sentence in the summary and compute similarity scores between the fragments and the SCUs in the pyramid based on unigram overlap. Then, they find the optimal correspondences between SCUs and the fragments that maximize the sum of similarity scores. Passonneau et al. (2013) extended the method by introducing distributional semantics to compute the similarity scores between SCUs and the fragments.

Recently, Yang et al. (2016) proposed the first automatic pyramid method, Pyramid Evaluation via Automated Knowledge Extraction (PEAK). PEAK employs subject-predicate-object triples extracted by ClausIE (Del Corro and Gemulla, 2013) as SCUs, and constructs pyramids by cutting a graph whose vertices represent the triples and whose edges represent semantic similarity scores between the triples computed by Align, Disambiguate and Walk (ADW) (Pilehvar et al., 2013). When evaluating a summary, PEAK constructs a weighted bipartite graph whose vertices represent subject-predicate-object triples extracted from the pyramid and the summary, respectively; the edges represent the similarity scores between the triples as computed by ADW. It scores the summary by solving the Linear Assignment Problem which involves maximizing the sum of the similarity scores on the bipartite graph.

The major difference between PEAK and our method is that the former regards the reference summary as a set of subject-predicate-object triples while the latter regards a reference summary as a set of EDUs obtained from the source documents. Thus, to construct high quality pyramids, PEAK is required to not only accurately extract the triples but also measure the semantic similarity between them accurately. However, in general, both extracting the triples and measuring the semantic similarity are still challenging NLP tasks. The performances are not always achieved in practical use. Actually, the F-measure of ClausIE is around 0.6 (Del Corro and Gemulla, 2013) and the correlation coefficients between the semantic similarity obtained from ADW and human judgment lie in the range of 0.55 to 0.88

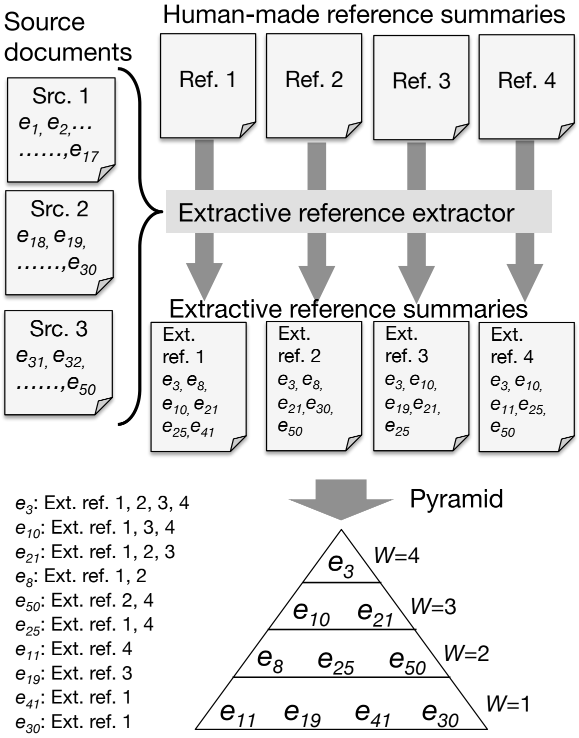

Figure 1: Overview of our pyramid construction.

(Pilehvar et al., 2013). As a result, the resultant pyramids have insufficient quality to be practical. Clearly, further improvement is necessary.

While our method is required to decompose a document into EDUs accurately, the EDU segmenter offers accurate decomposition performance; existing EDU boundary detection methods have F-measures over 0.9 (Fisher and Roark, 2007; Feng and Hirst, 2014). Moreover, since extractive reference summaries are set of EDUs from the source documents, we do not need semantic similarity to identify EDUs that have the same meaning. Thus, we can easily construct a pyramid by simply counting the number of extractive reference summaries that contains each EDU.

3 Automatic Pyramid Evaluation

First, we transform human-made reference summaries into extractive reference summaries; the EDUs in the source documents are used as the atomic units. Second, we construct a pyramid by weighting EDUs in the extractive reference summaries. EDU weights are defined as the number of reference summaries that contain each EDU (see Figure 1). In addition, we propose two techniques for deriving the extractive reference summaries.

3.1 Extractive Reference Summaries based on Alignment between EDUs

When similarity scores between EDUs in a reference summary and those in the source documents are available, we can regard extractive reference summary derivation as an optimal alignment problem with a length constraint, an extension of Linear Assignment Problem. We assume that a bipartite graph in which the vertices represent EDUs in the reference summary and source documents, and the edges represent similarity scores between the EDUs. The optimal alignment is obtained by solving following ILP problem:

$$\text{maximize} \sum_{j=1}^{|\mathcal{E}|} \sum_{k=1}^{|\mathcal{M}|} \phi(e_j, m_k) a_{j,k} \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{|\mathcal{E}|} \sum_{k=1}^{|\mathcal{M}|} \ell(e_j) a_{j,k} \leq L_{\max} \tag{2}$$

$$\sum_{j=1}^{|\mathcal{E}|} a_{j,k} \leq 1 \qquad \forall k \tag{3}$$

$$\sum_{k=1}^{|\mathcal{M}|} a_{j,k} \leq 1 \qquad \forall j \tag{4}$$

$$a_{j,k} \in \{0, 1\} \qquad \forall j, k. \tag{5}$$

$\mathcal{E}$ is the set of all EDUs in the source documents and $\mathcal{M}$ is the set of all EDUs in the reference summary. $\ell()$ returns the length (the number of words) of a textual unit. $\phi(e_j, m_k)$ returns the similarity score between the j-th EDU in the source documents and the k-th EDU in the reference summary as follows:

$$\phi(e_j, m_k) = \frac{\ell(\text{LCS}(e_j, m_k))}{\ell(m_k)}. \tag{6}$$

LCS$(\cdot, \cdot)$ returns the Longest Common Subsequence between e_j and m_k. $a_{j,k}$ is a binary indicator, and $a_{j,k} = 1$ denotes that the j-th EDU e_j in the source documents is aligned to the k-th EDU in the reference summary, *i.e.*, e_j is included in the extractive reference summary. Equation (2) ensures the the length of the extractive reference summary is less than $L_{\max}$, the length of the human-made reference summary. After solving the ILP problem, we can obtain the extractive reference summaries by collecting EDUs according to $a_{j,k} = 1$.

3.2 Extractive Reference Summaries based on Extractive Oracle Summaries

As another extractive reference summary, we can utilize extractive oracle summary (Hirao et al., 2017). The extractive oracle summary is defined as the set of consequential textual fragments within a sentence obtained from the source documents that has the maximum automatic evaluation score. Since we regard EDUs as SCUs and employ ROUGE/BE as an automatic evaluation measure, an extractive reference summary is a summary that consists of EDUs in the source documents and has maximum ROUGE/BE score.

For a given reference summary R , the extractive oracle summary is defined as follows:

$$\begin{aligned} O = &\mathop{\arg\max}_{E \subseteq \mathcal{E}} f(R, S) \\ \text{s.t.} \quad & \ell(S) \leq L_{\max}. \end{aligned} \tag{7}$$

$f()$ denotes an automatic evaluation measure (ROUGE/BE) and is defined as follows:

$$f(R, S) = \frac{\sum_{i=1}^{|U|} \min\{N(u_i, R), N(u_i, S)\}}{\sum_{i=1}^{|U|} N(u_i, R)}. \tag{8}$$

S is a system summary and U is the set of all atomic units in the reference summary. N-grams are utilized as the units used in computing ROUGE and head-modifier-relation triples are utilized in computing BE. $N(u_i, R)$; $N(u_i, S)$ returns the number of occurrences of the units in the reference and system summary, respectively.

Since the extractive oracle summaries in Hirao et al. (2017) are based on sentences, we extend the method to obtain EDU-based extractive oracle summaries. The ILP formulation that returns an extractive oracle summary is defined as follows:

$$\text{maximize} \sum_{i=1}^{|U|} z_i - \sum_{m=1}^{|S|} s_m \tag{9}$$

$$\text{s.t.} \quad \sum_{j=1}^{|\mathcal{E}|} \ell(e_j) x_j \leq L_{\max} \tag{10}$$

$$N(u_i, R) \geq z_i \qquad \forall i \tag{11}$$

$$\sum_{n \in V_i} d_n \geq z_i \qquad \forall i \tag{12}$$

$$x_{\text{left}(n)} \geq d_n \qquad \forall n \tag{13}$$

$$x_{\text{right}(n)} \geq d_n \qquad \forall n \tag{14}$$

$$s_{\text{c}(j)} \geq x_j \qquad \forall j \tag{15}$$

$$d_n \in \{0, 1\} \qquad \forall n \tag{16}$$

$$x_j \in \{0, 1\} \qquad \forall j \tag{17}$$

$$z_i \in \mathbb{Z}_+ \qquad \forall i. \tag{18}$$

z_i is the count of the i-th unit in the oracle summary. x_j is a binary indicator, $x_j = 1$ denotes that the j-th EDU, e_j, is included in the oracle summary. s_m is a binary indicator, $s_m = 1$ denotes that EDU(s) in m-th sentence is included in the oracle summary. The value of $\sum_{m=1}^{|S|} s_m$ is equal to the number of sentence whose EDU(s) is used in oracle summary. Thus, an oracle summary that consist of fewer sentences tends to obtain a higher objective value. Therefore, we can avoid generating fragmented oracle summaries with low readability. This objective function is inspired by the work of compressive summarization method (Morita et al., 2013). V_i is the set of indices indicating the position of the i-th unit, and d_n is a binary indicator indicating whether the n-th unit is contained in the oracle summary or not. Function left($\cdot$) and right($\cdot$) return the index of EDU that contains a word on the left in the unit, and the index of EDU that contains a word on the right in the unit, respectively. Function c($\cdot$) returns the index of sentence that contains j-th EDU.

Figure 2 shows examples. Suppose that the 10-th triple in U is "<has,computer,rcmod>". From the figure, the indices of the triple corresponding to "<has,computer,rcmod>" are 6 and 21. Thus, $V_{10} = \{6, 21\}$. The word on the left in the triples is "computer" and the word on the right is "has". For the first triple, the index of the EDU that contains "computer" is 1 and the index of the EDU that contains "has" is 2. For the second triple, the index of the EDU that corresponds to "computer" is 4, while that of "has" is 5. Thus, $\text{left}(6) = 1$ and $\text{right}(6) = 2$, $\text{left}(21) = 4$, and $\text{right}(21) = 5$.

After solving the ILP problem, we construct the extractive oracle summary by collecting EDUs according to $x_j = 1$.

3.3 Pyramid Construction: EDU Weighting

By deriving extractive references, we can easily construct a pyramid. The weight of an EDU is defined as the number of extractive references that contain the EDU. Here, $\mathcal{P}$ is a complete set of all EDUs in K extractive reference summaries, *i.e.*, $\mathcal{P} = \bigcup_{i=1}^{K} E_i$. E_i is the set of EDUs obtained from the i-th extractive reference summary. The weight

Index of word/triple	word	Index of dependant	Relation	Index of EDU
1	**We**	**2**	**nsubj**	**1**
2	**need**	0	ROOT	**1**
3	a	4	det	1
4	**computer**	2	dobj	**1**
5	that	6	nsubj	2
6	**has**	**4**	**rcmod**	**2**
7	an	9	det	2
8	excellent	9	amod	2
9	**CPU**	6	dobj	**2**
10	to	11	aux	3
11	**implement**	**9**	**vmod**	**3**
12	the	13	det	3
13	algorithm.	11	dobj	3

Index of word/triple	word	Index of dependant	Relation	Index of EDU
14	**We**	**17**	**nsubj**	**4**
15	do	17	aux	4
16	not	17	neg	4
17	**need**	0	ROOT	**4**
18	a	18	det	4
19	**computer**	17	dobj	**4**
20	that	21	nsubj	5
21	**has**	**19**	**rcmod**	**5**
22	an	24	det	5
23	excellent	24	amod	5
24	GPU.	21	dobj	5

u_{10}=<has, computer, rcmod>
V_{10}={6, 21}
left(6)=1 left(21)=4
right(6)=2 right(21)=5

u_{15}=<We, need, nsubj>
V_{15}={1, 14}
left(1)=1 left(14)=4
right(1)=1 right(14)=4

u_{20}=<implement, CPU, vmod>
V_{20}={11}
left(11)=2
right(11)=3

Figure 2: Examples of head-modifier-relation triples.

of the j-th EDU in $\mathcal{P}$ is defined as follows:

$$w_j = C(p_j). \quad (19)$$

$C()$ returns the number of extractive reference summaries that contain p_j, *i.e.*, the maximum score of $C(p_j)$ is K and its minimum score is 1. Since all EDUs in the source documents are assigned an integer score in the range of 0[1] to K, the scoring can be regard as a variant of relative utility score (Radev and Tam, 2003).

3.4 Automatic Scoring of Summaries

Based on the pyramid, we compute a score for a summary by aligning EDUs in the pyramid and EDUs in the system summary. By following PEAK, we find the optimal alignment by solving the Linear Assignment Problem. That is, we compute all similarity scores between EDUs in the summary and pyramid and then find the maximal score so that each EDU in the system summary is matched to at most one EDU in the pyramid. The ILP formulation of the problem is as follows:

$$\text{maximize} \quad \sum_{i=1}^{|\mathcal{C}|} \sum_{j=1}^{|\mathcal{P}|} g(c_i, p_j) w_j \alpha_{i,j} \quad (20)$$

$$\text{s.t.} \quad \sum_{i=1}^{|\mathcal{C}|} \alpha_{i,j} \leq 1 \quad \forall i \quad (21)$$

$$\sum_{j=1}^{|\mathcal{P}|} \alpha_{i,j} \leq 1 \quad \forall j \quad (22)$$

$$\alpha_{i,j} \in \{0, 1\} \quad \forall i, j (23)$$

$\mathcal{C}$ is the set of EDUs in the system summary. Function $g()$ indicates a binary function based on the similarity score between EDUs as follows:

$$g(c, p) = \begin{cases} 1 & \phi(c, p) \geq t \\ 0 & \text{otherwise.} \end{cases} \quad (24)$$

We set $t = 0.55$ in our experiments (Section 4).

$\alpha_{i,j}$ is a binary indicator, $\alpha_{i,j} = 1$ denotes that the i-th EDU in the system summary is aligned to j-th EDU in the pyramid.

The optimal solution of the objective function in the ILP problem (19)-(22) is not normalized. Since the unnormalized score is not suitable for comparing systems, we propose a normalization method. To normalize the score to satisfy the range of 0 to 1, we divide the score by the maximum score of sum of the EDU weights. Since every EDU in the pyramid has both length (the number of the word) and weight, the maximum score is derived by solving the *knapsack problem*:

$$\text{maximize} \quad \sum_{j=1}^{|\mathcal{P}|} w_j x_j \quad (25)$$

$$\text{s.t.} \quad \sum_{j=1}^{|\mathcal{P}|} \ell(p_j) \leq L_{\max} \quad (26)$$

---

[1] The EDUs that are not included in pyramid have scores of zero.

	2003	2004	2005	2006		2007	
Manual evaluation	Cov.	Cov.	Resp.	Resp.	Pyr.	Resp.	Pyr.
# of Topics	30	50	50	50	20	45	23
# of Systems	16	17	32	35	22	32	13
# of References	4	4	4,9	4		4	
Summary length	100	100	250	250		250	
Summary type	Generic	Generic	Query-focused	Query-focused		Query-focused	

Table 1: Statistics of the data sets (DUC-2003 to 2007).

	2008	2009	2010	2011
Manual evaluation	Resp. Pyr.	Resp. Pyr.	Resp. Pyr.	Resp Pyr.
# of Topics	48	44	46	44
# of Systems	58	55	43	50
# of References	4	4	4	4
Summary length	100	100	100	100
Summary type	Update Initial Update	Update Initial Update	Guided Initial Update	Guided Initial Update

Table 2: Statistics of the data sets (TAC-2008 to 2011).

$$x_j \in \{0, 1\} \qquad \forall j \qquad (27)$$

x_j is a binary indicator, and $x_j = 1$ denotes that the j-th EDU is included in the *knapsack*.

Finally, the score is defined as $\mathrm{Pyramid}(\mathcal{P}, S) = \mathrm{OTP}_{\mathrm{LAP}}/\mathrm{OPT}_{\mathrm{KP}}$. $\mathrm{OPT}_{\mathrm{LAP}}$ and $\mathrm{OPT}_{\mathrm{LAP}}$ denote maximum score of Equation (20) and maximum score of Equation (25), respectively.

4 Experiments

To investigate the effectiveness of our automatic evaluation method, we compare the correlation coefficients yielded by our methods with those obtained from strong baselines, ROUGE-2, ROUGE-SU4 and BE. We employ ROUGE toolkit version 1.5.5 to compute ROUGE/BE scores and Stanford Parser (de Marneffe et al., 2006) to obtain head-modifier-relation triples. In addition, we examine two types of oracle summaries for our method. One is ROUGE-2-based, the other is BE-based.

We evaluate automatic evaluation measures by Pearson's correlation r, Spearman's rank correlation ρ and Kendall's rank correlation τ. Correlation coefficients are computed by average automatic score and average manual evaluation score for all topics.

4.1 Data Sets

We conducted experiments on the data sets developed for multi document summarization tasks in DUC-2003 to 2007 and TAC-2008 to 2011. Table 1 and Table 2 show the statistics of the data sets.

DUC-2003 and 2004 were used for a generic summarization task with 100 word limit; mean coverage was used in a manual evaluation. DUC-2005 to 2007 were used for a query-focused summarization task with 250 word limit; responsiveness was used in a manual evaluation. The number of topics varied from 30 to 50 and the participating systems from 16 to 35. Note that the pyramid method was applied to small sets of topics in DUC-2006 and 2007.

TAC-2008 and 2009 were used for an update summarization task while TAC-2010 and 2011 were employed for a guided summarization task. For both tasks, the participating systems required two types of summaries, *initial summary* and *update summary* with 100 word limit. Both pyramid method and responsiveness were used in manual evaluations. In particular, TAC-2008 to 2011 have large numbers of participating systems, from 44 to 48.

4.2 EDU Segmenter

We regard decomposing a sentence into EDUs as a sequential tagging problem and implement a neural EDU segmenter that classifies each word in a sentence as the boundary of EDU or not based on 3-layer bi-LSTM (Wang et al., 2015). The size of word embeddings and hidden layers of the LSTM were set to 100 and 256, respectively. To handle low-frequency words, all words are encoded to 40 dimension hidden state by using character-based bi-LSTM (Lample et al., 2016). To utilize entire words in a corpus, we integrated word dropout (Iyyer et al., 2015) into our models with smoothing rate, 1.0. Moreover, to avoid overfitting the training data, dropout layer was adopted to the input of the LSTMs with the ratio 0.3.

The segmenter was trained by utilizing the training data of RST Discourse Treebank corpus (Carlson et al., 2001). The macro-averaged F-measure of boundary detection on the test data of the corpus is 0.917. The source documents, system summaries and reference summaries utilized

	2003	2004	2005	2006		2007	
	Cov.	Cov.	Resp.	Resp.	Pyr.	Resp.	Pyr.
ROUGE-2	.906/.821/.617	.909/.838/.691	.932/.931/.792	.836/.767/.584	.905/.884/.740	.880/.873/.715	.979/.989/.949
ROUGE-SU4	.782/.774/.600	.854/.772/.559	.925/.893/.731	.849/.790/.601	.885/.850/.706	.835/.832/.650	.961/.973/.897
BE	.927/.862/.617	.936/.868/.721	.897/.867/.714	.834/.757/.584	.883/.837/.680	.891/.890/.732	.982/.973/.897
PEAK	–	–	–	.617/.640/–	.508/.538/–	–	–
Prop(BE)	.936/.909/.750	.929/.892/.750	.845/.819/.657	.786/.716/.516	.877/.833/.687	.885/.881/.715	.936/.967/.897
Prop(ROUGE)	.908/.874/.750	.938/.814/.676	.864/.809/.629	.740/.670/.465	.871/.818/.662	.853/.845/.679	.943/.951/.872
Prop(AL)	.831/.841/.633	.904/.855/.735	.821/.757/.567	.762/.667/.465	.801/.772/.584	.814/.793/.610	.958/.962/.872

Table 3: Evaluation results from DUC-2003 to 2007.

		Inital		Update	
		Pyr.	Resp.	Pyr.	Resp.
2008	ROUGE-2	.908/.909/.757	.830/.868/.677	.943/.942/.800	.910/.888/.728
	ROUGE-SU4	.888/.885/.733	.803/.834/.636	.926/.933/.783	.902/.895/.725
	BE	.913/.903/.732	.817/.818/.627	.944/.939/.799	.913/.880/.712
	Prop(BE)	.926/.905/.734	.867/.852/.663	.940/.918/.779	.922/.899/.736
	Prop(ROUGE)	.895/.891/.708	.851/.840/.648	.912/.871/.702	.901/.872/.699
	Prop(AL)	.833/.792/.598	.779/.794/.602	.929/.895/.746	.909/.905/.750
		Pyr.	Resp.	Pyr.	Resp.
2009	ROUGE-2	.911/.950/.823	.757/.844/.674	.939/.896/.742	.717/.755/.600
	ROUGE-SU4	.920/.925/.786	.767/.805/.631	.939/.857/.701	.729/.719/.568
	BE	.856/.931/.784	.692/.838/.670	.924/.929/.798	.695/.816/.670
	Prop(BE)	.867/.932/.782	.854/.848/.670	.855/.917/.782	.866/.810/.656
	Prop(ROUGE)	.886/.917/.770	.858/.819/.639	.864/.890/.741	.822/.735/.586
	Prop(AL)	.901/.872/.689	.881/.821/.666	.886/.857/.704	.830/.743/.594
		Pyr.	Resp.	Pyr.	Resp.
2010	ROUGE-2	.978/.917/.787	.967/.924/.801	.963/.911/.758	.953/.890/.742
	ROUGE-SU4	.968/.947/.830	.954/.952/.837	.910/.885/.727	.900/.878/.727
	BE	.965/.942/.817	.943/.907/.749	.953/.911/.775	.928/.872/.713
	Prop(BE)	.949/.872/.713	.953/.867/.720	.954/.912/.764	.957/.913/.774
	Prop(ROUGE)	.952/.854/.673	.959/.859/.702	.938/.873/.713	.936/.860/.711
	Prop(AL)	.928/.882/.697	.929/.891/.720	.898/.853/.676	.900/.845/.691
		Pyr.	Resp.	Pyr.	Resp.
2011	ROUGE-2	.955/.888/.734	.930/.776/.592	.862/.789/.616	.879/.831/.665
	ROUGE-SU4	.976/.888/.726	.943/.778/.585	.857/.824/.642	.892/.865/.689
	BE	.934/.900/.736	.903/.757/.554	.880/.828/.670	.842/.783/.610
	Prop(BE)	.905/.857/.690	.917/.832/.640	.891/.880/.693	.889/.868/.694
	Prop(ROUGE)	.925/.883/.708	.924/.847/.673	.864/.864/.689	.870/.862/.683
	Prop(AL)	.934/.891/.713	.920/.792/.618	.843/.787/.601	.865/.799/.607

Table 4: Evaluation results from TAC-2008 to 2011.

in our experiments were decomposed into EDUs by the segmenter.

4.3 Results and Discussion

Table 3 and 4 list the correlation coefficients between manual evaluation and automatic evaluation for DUC-2003 to 2007 and TAC-2008 to 2011, respectively. In the tables, the coefficients are written in the order "Pearson's r/ Spearman's ρ/ Kendall's τ". The rows of Prop(BE), Prop(ROUGE) and Prop(AL) denote our method with BE-based oracle summaries as extractive reference summaries, with ROUGE-2-based oracle summaries, and with extractive reference summaries based on alignment, respectively. "Cov.", "Resp." and "Pyr." denote mean coverage, responsiveness and manual pyramid, respectively.

With regard to mean coverage on DUC-2003 to 2004, Prop(BE) achieved the best correlation coefficients. The correlation coefficients indicate very strong correlation with the manual evaluation. Prop(ROUGE) and Prop(AL) attained comparable correlation coefficients with the baseline methods. The correlation coefficients still indicate strong correlation.

With regard to responsiveness, our methods achieved lower correlation coefficients on DUC-2005 to 2007 than on DUC-2003 to 2004. Although our methods are outperformed by the baseline methods, both r and ρ of Prop(BE) still exceed 0.8 except for responsiveness on DUC-2006. Since our methods mimic manual pyramid evaluation, correlation coefficients against manual pyramid on DUC-2006 to 2007 are better than those against responsiveness and the scores are comparable to those of the baselines.

Moreover, we compare our methods with PEAK on the DUC-2006 data set. For manual pyramid, r and ρ are 0.508 and 0.538, respectively, while for responsiveness they are 0.617 and 0.640, respectively. These scores are significantly lower than those attained by our methods and baselines. Note that these results are obtained by running the code from the author's web page `http://www.larayang.com/peak/`. The results demonstrated that our methods are superior to PEAK.

For manual pyramid on TAC-2008 to 2011, all methods attained quite strong correlation. The scores achieved were around 0.9 and better than those on DUC-2003 to 2007. In particular, Prop(BE) achieved the best scores in some cases. Although, responsiveness yielded lower correlation coefficients than manual pyramid, Prop(BE) still retains strong correlation , e.g., ρ are in the range of 0.857 to 0.932 against manual pyramid, 0.810 to 0.913 against responsiveness. The average correlation coefficients across all data sets on TAC are shown in Table 5. The average correlation coefficients of Prop(BE) slightly lower than those of ROUGE-2 and BE against manual pyramid. On the ohter hand, Prop(BE) achieved the best correlation coefficients against responsiveness. The results imply that Prop(BE) achieves comparable performance to baseline methods.

In a comparison of our methods, Prop(BE) attained the best results while Prop(ROUGE) showed better results than Prop(AL) in many cases. These results imply that extractive oracle summaries are helpful as extractive reference summaries and BE is better objective function to generate them.

In addition, we show SCUs and corresponding EDUs obtained from a human-made pyramid and Prop(BE) in Figure 3. They are obtained from topic "Earthquake Sichuan (ID:D1110B)" from TAC-2011 Guided Summarization Task, the topic type is categorized as "Accidents and Natural Disasters". Summarizers are required to generate a summary that includes following aspects: (1) WHAT: what happend, (2) WHEN: date, time, other temporal placement makers, (3) WHERE: physical location, (4) WHY: reasons for accident/disaster, (5) WHO_AFFECTED: casualties (death, injury), or individuals otherwise negatively affected by the accident/disaster, (6) DAMAGES: damages caused by the accident/disaster, (7) COUNTERMEASURES: countermeasures, rescue efforts, prevention efforts, other reactions to the accident/disaster. From the figure, we can see that the EDUs are not always identical to human-generated SCUs at word-level but are identical at concept-level.

	Pyr.	Resp.
ROUGE-2	.932/.900/.752	.868/.847/.685
ROUGE-SU4	.923/.893/.741	.861/.841/.675
BE	.921/.910/.764	.842/.834/.663
Prop(BE)	.911/.899/.742	.903/.861/.694
Prop(ROUGE)	.905/.880/.713	.890/.837/.668
Prop(AL)	.894/.856/.678	.877/.824/.656

Table 5: Average correlation coefficients across data sets (TAC-2008 to TAC-2011)

In short, these results imply that our methods have at least comparable performance to the baselines. Although our methods are outperformed by the baselines in some cases, the correlation coefficients are high enough against manual evaluation. Moreover, our methods have a significant advantage over the baselines methods because our methods clearly indicate whether the output of the text summarization system failed to include important SCUs. Thus, our automatic pyramid method enhanced with extractive oracle summaries is helpful for further improvement of summarization systems.

5 Conclusion

This paper proposed an automatic pyramid evaluation method that allows us to scrutinize the failure analysis of systems. To construct a pyramid, we transform human-made reference summaries into extractive reference summaries whose atomic units are EDUs obtained from the source documents. Then, we weight every EDU by counting the number of extractive reference summaries that contain the EDU. When evaluating a summary, we determine the correspondences between EDUs in the pyramid to those in the summary by solving Linear Assignment Problem and give a score to the summary based on the correspondences. We also proposed two types of extractive reference summaries. The first is the alignment-based extractive reference summary. The second is the extractive

SCUs obtained from human-made pyramid

$w = 4$ The 7.8-magnitude earthquake struck
$w = 4$ Sichuan Province of China
$w = 4$ No warning signs detected
$w = 4$ Over 8,500 killed
$w = 4$ China allocated 200 million yuan ($29 Million) disaster relief
$w = 1$ Rain is forecast, could hamper relief efforts
$w = 1$ Quake also affected Gansu, Shaanxi provinces, and Chongqing municipality

EDUs obtained from pyramid of Prop(BE)

$w = 1$ The 7.8-magnitude earthquake struck Sichuan province shortly before 2:30 pm
$w = 2$ in aid for earthquake victims in Sichuan Province of China
$w = 4$ Chinese authorities did not detect any warning signs ahead of Monday's earthquake
$w = 1$ leaving at least 12,000 people died
$w = 2$ China has allocated 200 million yuan
$w = 1$ Rain in the coming days in Sichuan is expected to hamper earthquake relief efforts, as well as increase risks of landslides
$w = 1$ 50 in the municipality of Chongqing, 61 in Shaanxi province, and one in southwestern Yunnan

Figure 3: Examples of SCUs obtained from pyramids.

oracle summary.

To demonstrate the effectiveness of our methods, we conducted experiments on DUC-2003 to 2007 and TAC-2008 to 2011 data sets. The results demonstrated that our method yielded results that well correlated with various manual evaluations. The correlation coefficients are at least comparable to those obtained from strong baselines, ROUGE-2, ROUGE-SU and BE and significantly higher than those obtained from previous automatic pyramid evaluation, PEAK.

References

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proc. of the Second SIGdial Workshop on Dialogue and Discourse*, pages 1–10.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366.

Vanessa Wei Feng and Graeme Hirst. 2014. Two-pass discourse segmentation with pairing and global features. *CoRR*, abs/1407.8215.

Seeger Fisher and Brian Roark. 2007. The utility of parse-derived features for automatic discourse segmentation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 488–495.

Aaron Harnly, Ani Nenkova, Rebecca Passonneau, and Owen Rambow. 2005. Automation of summary evaluation by the pyramid method. In *Proc. of the International Conference on Recent Advances in Natural Language Processing (RANLP 2005)*, pages 226–232.

Tsutomu Hirao, Masaaki Nishino, Jun Suzuki, and Masaaki Nagata. 2017. Enumeration of extractive oracle summaries. In *Proc. of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 386–396.

Eduard Hovy, Chin-Yew Lin, Liang Zhou, and Junichi Fukumoto. 2006. Automated summarization evaluation with basic elements. In *Proc. of the 5th International Conference Language Resource and Evaluation (LREC06)*, pages 899–902.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1681–1691.

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

Junyi Jessy Li, Kapil Thadani, and Amanda Stent. 2016. The role of discourse units in near-extractive summarization. In *Proc. of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 137–147.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proc. of Workshop on Text Summarization Branches Out*, pages 74–81.

Mann, William Charles and Thompson, Sandra Annear. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *In Proceedings of International Conference on Language Resources and Evaluation (LREC)*, pages 449–454.

Hajime Morita, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2013. Subtree extractive summarization via submodular maximization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1023–1032.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proc. of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics:HLT-NAACL 2004*, pages 145–152.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(2).

Rebecca J. Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated pyramid scoring of summaries using distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 143–147.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1341–1351.

Dragomir R. Radev and Daniel Tam. 2003. Summarization evaluation using relative utility. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pages 508–511.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *CoRR*, abs/1510.06168.

Qian Yang, Rebecca Passonneau, and Gerard de Melo. 2016. Peak: Pyramid evaluation via automated knowledge extraction. In *Proc. of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, pages 2673–2679.

Learning to Encode Text as Human-Readable Summaries using Generative Adversarial Networks

Yau-Shian Wang
National Taiwan University
`king6101@gmail.com`

Hung-Yi Lee
National Taiwan University
`tlkagkb93901106@gmail.com`

Abstract

Auto-encoders compress input data into a latent-space representation and reconstruct the original data from the representation. This latent representation is not easily interpreted by humans. In this paper, we propose training an auto-encoder that encodes input text into human-readable sentences, and unpaired abstractive summarization is thereby achieved. The auto-encoder is composed of a generator and a reconstructor. The generator encodes the input text into a shorter word sequence, and the reconstructor recovers the generator input from the generator output. To make the generator output human-readable, a discriminator restricts the output of the generator to resemble human-written sentences. By taking the generator output as the summary of the input text, abstractive summarization is achieved without document-summary pairs as training data. Promising results are shown on both English and Chinese corpora.

1 Introduction

When it comes to learning data representations, a popular approach involves the auto-encoder architecture, which compresses the data into a latent representation without supervision. In this paper we focus on learning text representations. Because text is a sequence of words, to encode a sequence, a sequence-to-sequence (seq2seq) auto-encoder (Li et al., 2015; Kiros et al., 2015) is usually used, in which a RNN is used to encode the input sequence into a fixed-length representation, after which another RNN is used to decode the original input sequence given this representation.

Although the latent representation learned by the seq2seq auto-encoder can be used in downstream applications, it is usually not human-readable. A human-readable representation should comply the rule of human grammar and can be comprehended by human. Therefore, in this work, we use comprehensible natural language as a latent representation of the input source text in an auto-encoder architecture. This human-readable latent representation is shorter than the source text; in order to reconstruct the source text, it must reflect the core idea of the source text. Intuitively, the latent representation can be considered a summary of the text, so unpaired abstractive summarization is thereby achieved.

The idea that using human comprehensible language as a latent representation has been explored on text summarization, but only in a semi-supervised scenario. Previous work (Miao and Blunsom, 2016) uses a prior distribution from a pre-trained language model to constrain the generated sequence to natural language. However, to teach the compressor network to generate text summaries, the model is trained using labeled data. In contrast, in this work we need no labeled data to learn the representation.

As shown in Fig. 1, the proposed model is composed of three components: a generator, a discriminator, and a reconstructor. Together, the generator and reconstructor form a text auto-encoder. The generator acts as an encoder in generating the latent representation from the input text. Instead of using a vector as latent representation, however, the generator generates a word sequence much shorter than the input text. From the shorter text, the reconstructor reconstructs the original input of the generator. By minimizing the reconstruction loss, the generator learns to generate short text segments that contain the main information in the original input. We use the seq2seq model in modeling the generator and reconstructor because both have input and output sequences with different lengths.

However, it is very possible that the generator's output word sequence can only be processed and recognized by the reconstructor but is

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4187–4195
Brussels, Belgium, October 31 - November 4, 2018.

not readable by humans. Here, instead of regularizing the generator output with a pre-trained language model (Miao and Blunsom, 2016), we borrow from adversarial auto-encoders (Makhzani et al., 2015) and cycle GAN (Zhu et al., 2017) and introduce a third component – the discriminator – to regularize the generator's output word sequence. The discriminator and the generator form a generative adversarial network (GAN) (Goodfellow et al., 2014). The discriminator discriminates between the generator output and human-written sentences, and the generator produces output as similar as possible to human-written sentences to confuse the discriminator. With the GAN framework, the discriminator teaches the generator how to create human-like summary sentences as a latent representation. However, due to the non-differential property of discrete distributions, generating discrete distributions by GAN is challenging. To tackle this problem, in this work, we proposed a new kind of method on language generation by GAN.

By achieving unpaired abstractive text summarization, machine is able to unsupervisedly extract the core idea of the documents. This approach has many potential applications. For example, the output of the generator can be used for the downstream tasks like document classification and sentiment classification. In this study, we evaluate the results on an abstractive text summarization task. The output word sequence of the generator is regarded as the summaries of the input text. The model is learned from a set of documents without summaries. As most documents are not paired with summaries, for example the movie reviews or lecture recordings, this technique makes it possible to learn summarizer to generate summaries for these documents. The results show that the generator generates summaries with reasonable quality on both English and Chinese corpora.

2 Related Work

Abstractive Text Summarization

Recent model architectures for abstractive text summarization basically use the sequence-to-sequence (Sutskever et al., 2014) framework in combination with various novel mechanisms. One popular mechanism is attention (Bahdanau et al., 2015), which has been shown helpful for summarization (Nallapati et al., 2016; Rush et al., 2015; Chopra et al., 2016). It is also possible to directly optimize evaluation metrics such as ROUGE (Lin, 2004) with reinforcement learning (Ranzato et al., 2016; Paulus et al., 2017; Bahdanau et al., 2016). The hybrid pointer-generator network (See et al., 2017) selects words from the original text with a pointer (Vinyals et al., 2015) or from the whole vocabulary with a trained weight. In order to eliminate repetition, a coverage vector (Tu et al., 2016) can be used to keep track of attended words, and coverage loss (See et al., 2017) can be used to encourage model focus on diverse words. While most papers focus on supervised learning with novel mechanisms, in this paper, we explore unsupervised training models.

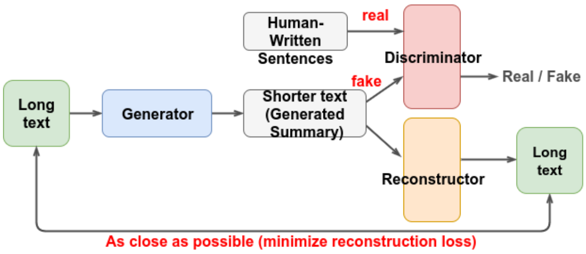

Figure 1: Proposed model. Given long text, the generator produces a shorter text as a summary. The generator is learned by minimizing the reconstruction loss together with the reconstructor and making discriminator regard its output as human-written text.

GAN for Language Generation

In this paper, we borrow the idea of GAN to make the generator output human-readable. The major challenge in applying GAN to sentence generation is the discrete nature of natural language. To generate a word sequence, the generator usually has non-differential parts such as *argmax* or other sample functions which cause the original GAN to fail.

In (Gulrajani et al., 2017), instead of feeding a discrete word sequence, the authors directly feed the generator output layer to the discriminator. This method works because they use the earth mover's distance on GAN as proposed in (Arjovsky et al., 2017), which is able to evaluate the distance between a discrete and a continuous distribution. SeqGAN (Yu et al., 2017) tackles the sequence generation problem with reinforcement learning. Here, we refer to this approach as adversarial REINFORCE. However, the discriminator only measures the quality of whole sequence, and thus the rewards are extremely sparse and the rewards assigned to all the generation steps are all the same. MC search (Yu et al., 2017) is proposed to evaluate the approximate reward at each time step, but this method suffers from high time complexity. Following this idea, (Li et al., 2017) proposes partial evaluation approach to evaluate the expected reward at each time step. In this paper, we propose the self-critical adversarial REINFORCE algorithm as another way to evaluate the expected reward at each time step. The performance between original WGAN and proposed adversarial REINFORCE is compared in experiment.

3 Proposed Method

The overview of the proposed model is shown in Fig. 2. The model is composed of three components: generator G, discriminator D, and reconstructor R. Both G and R are seq2seq hybrid pointer-generator networks (See et al., 2017) which can decide to copy words from encoder input text via pointing or generate from vocabulary. They both take a word sequence as input and output a sequence of word distributions. Discriminator D, on the other hand, takes a sequence as input and outputs a scalar. The model is learned from a set of documents x and human-written sentences y^{real}.

To train the model, a training document $x = \{x_1, x_2, ..., x_t, ..., x_T\}$, where x_t represents a word, is fed to G, which outputs a sequence of word distributions $G(x) = \{y_1, y_2, ..., y_n, ..., y_N\}$, where y_n is a distribution over all words in the lexicon. Then we randomly sample a word y_n^s from each distribution y_n, and a word sequence $y^s = \{y_1^s, y_2^s, ..., y_N^s\}$ is obtained according to $G(x)$. We feed the sampled word sequence y^s to reconstructor R, which outputs another sequence of word distributions $\hat{x}$. The reconstructor R reconstructs the original text x from y^s. That is, we seek an output of reconstructor $\hat{x}$ that is as close to the original text x as possible; hence the loss for training the reconstructor, R_{loss}, is defined as

$$R_{loss} = \sum_{k=1}^{K} l_s(x, \hat{x}), \tag{1}$$

where the reconstruction loss $l_s(x, \hat{x})$ is the cross-entropy loss computed between the reconstructor output sequence $\hat{x}$ and the source text x, or the negative conditional log-likelihood of source text x given word sequence y^s sampled from $G(x)$. The reconstructor output sequence $\hat{x}$ is teacher-forced by source text x. The subscript $_s$ in $l_s(x, \hat{x})$ indicates that $\hat{x}$ is reconstructed from y^s. K is the number of training documents, and (1) is the summation of the cross-entropy loss over all the training documents x.

In the proposed model, the generator G and reconstructor R form an auto-encoder. However, the reconstructor R does not directly take the generator output distribution $G(x)$ as input [1]. Instead, the reconstructor takes a sampled discrete sequence y^s as input. Due to the non-differentiable property of discrete sequences, we apply the REINFORCE algorithm, which is described in Section 4.

In addition to reconstruction, we need the discriminator D to discriminate between the real sequence y^{real} and the generated sequence y^s to regularize the generated sequence satisfying the summary distribution. D learns to give y^{real} higher scores while giving y^s lower scores. The loss for training the discriminator D is denoted as D_{loss}; this is further described in Section 5.

G learns to minimize the reconstruction loss R_{loss}, while maximizing the loss of the discriminator D by generating a summary sequence y^s that cannot be differentiated by D from the real thing. The loss for the generator G_{loss} is

$$G_{loss} = \alpha R_{loss} - D'_{loss} \tag{2}$$

where D'_{loss} is highly related to D_{loss} – but not necessary the same[2] – and α is a hyper-parameter. After obtaining the optimal generator by minimizing (2), we use it to generate summaries.

Generator G and discriminator D together form a GAN. We use two different adversarial training methods to train D and G; as shown in Fig. 2, these two methods have their own discriminators 1 and 2. Discriminator 1 takes the generator output layer $G(x)$ as input, whereas discriminator 2 takes the sampled discrete word sequence y^s as input. The two methods are described respectively in Sections 5.1 and 5.2.

4 Minimizing Reconstruction Loss

Because discrete sequences are non-differentiable, we use the REINFORCE algorithm. The generator is seen as an agent whose reward given the source text x is $-l_s(x, \hat{x})$. Maximizing the reward is equivalent to minimizing the reconstruction loss R_{loss} in (1). However, the reconstruction

---

[1] We found that if the reconstructor R directly takes $G(x)$ as input, the generator G learns to put the information about the input text in the distribution of $G(x)$, making it difficult to sample meaningful sentences from $G(x)$.

[2] D'_{loss} has different formulations in different approaches. This will be clear in Sections 5.1 and 5.2.

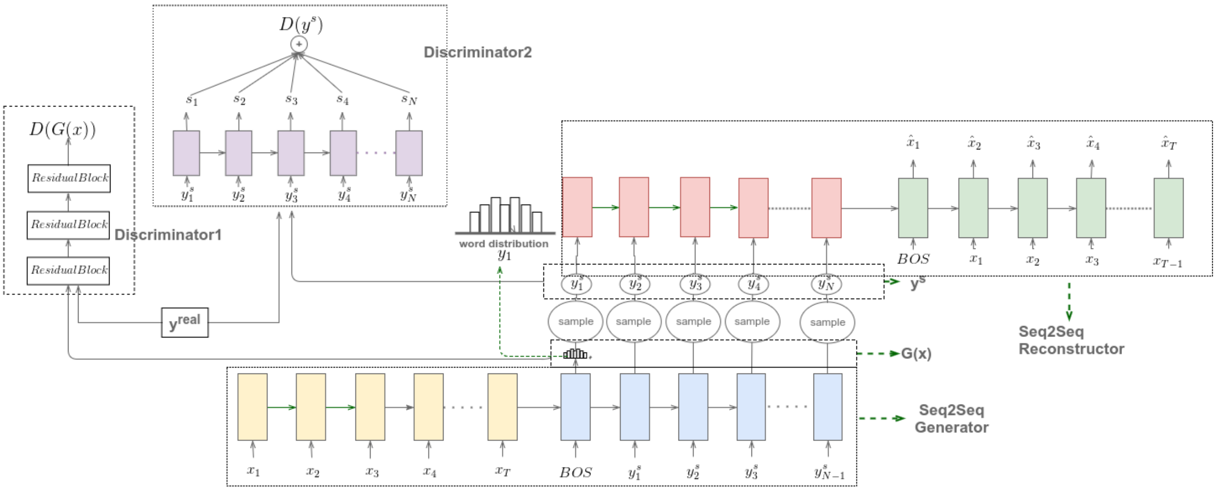

Figure 2: Architecture of proposed model. The generator network and reconstructor network are a seq2seq hybrid pointer-generator network, but for simplicity, we omit the pointer and the attention parts.

loss varies widely from sample to sample, and thus the rewards to the generator are not stable either. Hence we add a baseline to reduce their difference. We apply self-critical sequence training (Rennie et al., 2017); the modified reward $r^R(x, \hat{x})$ from reconstructor R with the baseline for the generator is

$$r^R(x, \hat{x}) = -l_s(x, \hat{x}) - (-l_a(x, \hat{x}) - b) \quad (3)$$

where $-l_a(x, \hat{x}) - b$ is the baseline. $l_a(x, \hat{x})$ is also the same cross-entropy reconstruction loss as $l_s(x, \hat{x})$, except that $\hat{x}$ is obtained from y^a instead of y^s. y^a is a word sequence $\{y_1^a, y_2^a, ..., y_n^a, ..., y_N^a\}$, where y_n^a is selected using the *argmax* function from the output distribution of generator y_n. As in the early training stage, the sequence y^s barely yields higher reward than sequence y^a, to encourage exploration we introduce the second baseline score b, which gradually decreases to zero. Then, the generator is updated using the REINFORCE algorithm with reward $r^R(x, \hat{x})$ to minimize R_{loss}.

5 GAN Training

With adversarial training, the generator learns to produce sentences as similar to the human-written sentences as possible. Here, we conduct experiments on two kinds of methods of language generation with GAN. In Section 5.1 we directly feed the generator output probability distributions to the discriminator and use a Wasserstein GAN (WGAN) with a gradient penalty. In Section 5.2, we explore adversarial REINFORCE, which feeds sampled discrete word sequences to the discriminator and evaluates the quality of the sequence from the discriminator for use as a reward signal to the generator.

5.1 Method 1: Wasserstein GAN

In the lower left of Fig. 2, the discriminator model of this method is shown as **discriminator1** D_1. D_1 is a deep CNN with residual blocks, which takes a sequence of word distributions as input and outputs a score. The discriminator loss D_{loss} is

$$D_{loss} = \frac{1}{K}\sum_{k=1}^{K} D_1(G(x^{(k)})) - \frac{1}{K}\sum_{k=1}^{K} D_1(y^{real(k)}) + \beta_1 \frac{1}{K}\sum_{k=1}^{K}(\Delta_{y^{i(k)}} D_1(y^{i(k)}) - 1)^2,$$

where K denotes the number of training examples in a batch, and k denotes the k-th example. The last term is the gradient penalty (Gulrajani et al., 2017). We interpolate the generator output layer $G(x)$ and the real sample y^{real}, and apply the gradient penalty to the interpolated sequence y^i. β_1 determines the gradient penalty scale. In Equation (2), for WGAN, the generator maximizes D'_{loss}:

$$D'_{loss} = \frac{1}{K}\sum_{k=1}^{K} D_1(G(x^{(k)})). \quad (4)$$

5.2 Method 2: Self-Critic Adversarial REINFORCE

In this section, we describe in detail the proposed adversarial REINFORCE method. The core idea is we use the LSTM discriminator to evaluate the current quality of the generated sequence

$\{y_1^s, y_2^s, ..., y_i^s\}$ at each time step i. The generator knows that compared to the last time step, as the generated sentence either improves or worsens, it can easily find the problematic generation step in a long sequence, and thus fix the problem easily.

5.2.1 Discriminator 2

As shown in Fig. 2, the **discriminator2** D_2 is a unidirectional LSTM network which takes a discrete word sequence as input. At time step i, given input word y_i^s it predicts the current score s_i based on the sequence $\{y_1, y_2, ..., y_i\}$. The score is viewed as the quality of the current sequence. An example of discriminator regularized by weight clipping(Arjovsky et al., 2017) is shown in Fig. 3.

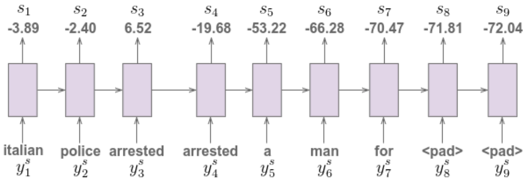

Figure 3: When the second *arrested* appears, as the sentence becomes ungrammatical, the discriminator determines that this example comes from the generator. Hence, after this time-step, it outputs low scores.

In order to compute the discriminator loss D_{loss}, we sum the scores $\{s_1, s_2, ..., s_N\}$ of the whole sequence y^s to yield

$$D_2(y^s) = \frac{1}{N}\sum_{n=1}^{N} s_n.$$

where N denotes the generated sequence length. Then, the loss of discriminator is

$$D_{loss} = \frac{1}{K}\sum_{k=1}^{K} D_2(y^{s(k)}) - \frac{1}{K}\sum_{k=1}^{K} D_2(y^{real(k)})$$
$$+\beta_2 \frac{1}{K}\sum_{k=1}^{K}(\Delta_{y^{i(k)}} D_2(y^{i(k)}) - 1)^2,$$

Similar to previous section, the last term is gradient penalty term. With the loss mentioned above, the discriminator attempts to quickly determine whether the current sequence is real or fake. The earlier the timestep discriminator determines whether the current sequence is real or fake, the lower its loss.

5.2.2 Self-Critical Generator

Since we feed a discrete sequence y^s to the discriminator, the gradient from the discriminator cannot directly back-propagate to the generator. Here, we use the policy gradient method. At timestep i, we use the $i-1$ timestep score s_{i-1} from the discriminator as its self-critical baseline. The reward r_i^D evaluates whether the quality of sequence in timestep i is better or worse than that in timestep $i-1$. The generator reward r_i^D from D_2 is

$$r_i^D = \begin{cases} s_i & \text{if i = 1} \\ s_i - s_{i-1} & \text{otherwise.} \end{cases}$$

However, some sentences may be judged as bad sentences at the previous timestep, but at later timesteps judged as good sentences, and vice versa. Hence we use the discounted expected reward d with discount factor γ to calculate the discounted reward d_i at time step i as

$$d_i = \sum_{j=i}^{N} \gamma^{j-i} r_j^D.$$

To maximize the expected discounted reward d_i, the loss of generator is:

$$G'_{loss} = -E_{y_i^s \sim p_G(y_i^s|y_1^s,...,y_{i-1}^s,x)}[d_i]. \qquad (5)$$

We use the likelihood ratio trick to approximate the gradient to minimize (5).

6 Experiment

Our model was evaluated on the English/Chinese Gigaword datasets and CNN/Daily Mail dataset. In Section 6.1,6.2 and 6.4, the experiments were conducted on English Gigaword, while the experiments were conducted on CNN/Daily Mail dataset and Chinese Gigaword dataset respectively in Sections 6.3 and 6.6. We used ROUGE(Lin, 2004) as our evaluation metric.[3] During testing, when using the generator to generate summaries, we used beam search with beam size=5, and we eliminated repetition. We provide the details of the implementation and corpus re-processing respectively in Appendix A and B.

Before jointly training the whole model, we pre-trained the three major components – generator, discriminator, and reconstructor – separately. First, we pre-trained the generator in an unsupervised manner so that the generator would be able to somewhat grasp the semantic meaning of the source text. The details of the pre-training are in Appendix C. We pre-trained the discriminator and reconstructor respectively with the pre-trained generator's output to ensure that these two critic networks provide good feedback to the generator.

---

[3] We used pyrouge package with option -m -n 2 -w 1.2 to compute ROUGE score for all experiments.

Task	Labeled	Methods	R-1	R-2	R-L
(A)Supervised	3.8M	(A-1)Supervised training on generator	33.19	14.21	30.50
		(A-2) (Rush et al., 2015)†	29.76	11.88	26.96
		(A-3) (Chopra et al., 2016)†	33.78	15.97	31.15
		(A-4) (Zhou et al., 2017)†	**36.15**	**17.54**	**33.63**
(B) Trivial baseline	0	(B-1) Lead-8	21.86	7.66	20.45
(C) Unpaired	0	(C-1) Pre-trained generator	21.26	5.60	18.89
		(C-2) WGAN	28.09	9.88	25.06
		(C-3) **Adversarial REINFORCE**	**28.11**	**9.97**	**25.41**
(D) Semi-supervised	10K	(D-1) WGAN	29.17	10.54	26.72
		(D-2) **Adversarial REINFORCE**	**30.01**	**11.57**	**27.61**
	500K	(D-3)(Miao and Blunsom, 2016)†	30.14	12.05	27.99
		(D-4) WGAN	32.50	13.65	29.67
		(D-5) **Adversarial REINFORCE**	**33.33**	**14.18**	**30.48**
	1M	(D-6)(Miao and Blunsom, 2016)†	31.09	12.79	28.97
		(D-7) WGAN	33.18	14.19	30.69
		(D-8) **Adversarial REINFORCE**	**34.21**	**15.16**	**31.64**
(E) Transfer learning	0	(E-1) Pre-trained generator	21.49	6.28	19.34
		(E-2) **WGAN**	25.11	7.94	23.05
		(E-3) **Adversarial REINFORCE**	**27.15**	**9.09**	**24.11**

Table 1: Average F1 ROUGE scores on English Gigaword. R-1, R-2 and R-L refers to ROUGE 1, ROUGE 2 and ROUGE L respectively. Results marked with † are obtained from corresponding papers. In part (A), the model was trained supervisedly. In row (B-1), we select the article's first eight words as its summary. Part (C) are the results obtained without paired data. In part (D), we trained our model with few labeled data. In part (E), we pre-trained generator on CNN/Diary and used the summaries from CNN/Diary as real data for the discriminator.

6.1 English Gigaword

The English Gigaword is a sentence summarization dataset which contains the first sentence of each article and its corresponding headlines. The preprocessed corpus contains 3.8M training pairs and 400K validation pairs. We trained our model on part of or fully unparalleled data on 3.8M training set. To have fair comparison with previous works, the following experiments were evaluated on the 2K testing set same as (Rush et al., 2015; Miao and Blunsom, 2016). We used the sentences in article headlines as real data for discriminator[4]. As shown in the following experiments, the headlines can even come from another set of documents not related to the training documents.

The results on English Gigaword are shown in Table 1. WGAN and adversarial REINFORCE refer to the adversarial training methods mentioned in Sections 5.1 and 5.2 respectively. Results trained by full labeled data are in part (A). In row (A-1), We trained our generator by supervised training. Compared with the previous work (Zhou et al., 2017), we used simpler model and smaller vocabulary size. We did not try to achieve the state-of-the-art results because the focus of this work is unsupervised learning, and the proposed approach is independent to the summarization models used. In row (B-1), we simply took the first eight words in a document as its summary.

The results for the pre-trained generator with method mentioned in Appendix.C is shown in row (C-1). In part (C), we directly took the sentences in the summaries of Gigaword as the training data of discriminator. Compared with the pre-trained generator and the trivial baseline , the proposed approach (rows (C-2) and (C-3)) showed good improvement. In Fig. 4, we provide a real example. More examples can be found in the Appendix.D.

6.2 Semi-Supervised Learning

In semi-supervised training, generator was pre-trained with few available labeled data. During training, we conducted teacher-forcing with labeled data on generator after several updates without labeled data. With 10K, 500K and 1M la-

---

[4] Instead of using general sentences as real data for discriminator, we chose sentences from headlines because they have their own unique distribution.

beled data, the teacher-forcing was conducted every 25, 5 and 3 updates without paired data, respectively. In teacher-forcing, given source text as input, the generator was teacher-forced to predict the human-written summary of source text. Teacher-forcing can be regarded as regularization of unpaired training that prevents generator from producing unreasonable summaries of source text. We found that if we teacher-forced generator too frequently, generator would overfit on training data since we only used very few labeled data on semi-supervised training.

The performance of semi-supervised model in English Gigaword regarding available labeled data is shown in Table 1 part (D). We compared our results with (Miao and Blunsom, 2016) which was the previous state-of-the-art method on semi-supervised summarization task under the same amount of labeled data. With both 500K and 1M labeled data, our method performed better. Furthermore, with only 1M labeled data, using adversarial REINFORCE even outperformed supervised training in Table 1 (A-1) with the whole 3.8M labeled data.

Source Text:
global one communications will launch its global intranet <unk> -lrb- virtual private networks -rrb- service in new zealand by the end of the year , the new zealand infotech weekly reported monday .

Ground Truth: international intranet service destined for new zealand	**(A-1)Supervised Result:** global one communications to launch global networks in new zealand
(C-2)WGAN: global communications launch global service in new zealand	**(C-3)Adversarial REINFORCE:** global communications launch global virtual private networks in new zealand
(E-2)WGAN: global one communications will launch its global networks -rrb- service	**(E-3)Adversarial REINFORCE:** global one communications will launch its global virtual networks

Figure 4: Real examples with methods referred in Table 1. The proposed methods generated summaries that grasped the core idea of the articles.

6.3 CNN/Daily Mail dataset

The CNN/Daily Mail dataset is a long text summarization dataset which is composed of news articles paired with summaries. We evaluated our model on this dataset because it's a popular benchmark dataset, and we want to know whether the proposed model works on long input and long output sequences. The details of corpus preprocessing can be found in Appendix.B . In unpaired training, to prevent the model from directly matching the input articles to its corresponding summaries, we split the training pairs into two equal sets, one set only supplied articles and the other set only supplied summaries.

The results are shown in Table 2. For supervised approaches in part (A), although our seq2seq model was similar to (See et al., 2017), due to the smaller vocabulary size (we didn't tackle out-of-vocabulary words), simpler model architecture, shorter output length of generated summaries, there was a performance gap between our model and the scores reported in (See et al., 2017). Compared to the lead-3 baseline in part (B) which took the first three sentences of articles as summaries, the seq2seq models fell behind. That was because news writers often put the most important information in the first few sentences, and thus even the best abstractive summarization model only slightly beat the lead-3 baseline on ROUGE scores. However, during pre-training or training we didn't make assumption that the most important sentences are in first few sentences.

We observed that our unpaired model yielded decent ROUGE-1 score, but it yielded lower ROUGE-2 and ROUGE-L score. That was probably because the length of our generated sequence was shorter than ground truth, and our vocabulary size was small. Another reason was that the generator was good at selecting the most important words from the articles, but sometimes failed to combine them into reasonable sentences because it's still difficult for GAN to generate long sequence. In addition, since the reconstructor only evaluated the reconstruction loss of whole sequence, as the generated sequence became long, the reconstruction reward for generator became extremely sparse. However, compared to pre-trained generator (rows (C-2), (C-3) v.s. (C-1)), our model still enhanced the ROUGE score. An real example of generated summary can be found at Appendix.D Fig.11 .

6.4 Transfer Learning

The experiments conducted up to this point required headlines unpaired to the documents but in the same domain to train discriminator. In this subsection, we generated the summaries from English Gigaword (target domain), but the summaries for discriminator were from CNN/Daily Mail dataset (source domain).

The results of transfer learning are shown in Table. 1 part (E). Table 1 (E-1) is the result of pre-

Methods		R-1	R-2	R-L
(A)Supervised	(A-1)Supervised training on our generator	38.89	13.74	29.42
	(A-2) (See et al., 2017)†	39.53	17.28	36.38
(B)Lead-3 baseline (See et al., 2017)†		40.34	17.70	36.57
(C) Unpaired	(C-1) Pre-trained generator	29.86	5.14	14.66
	(C-2) WGAN	35.14	**9.43**	**21.04**
	(C-3) **Adversarial REINFORCE**	**35.51**	9.38	20.98

Table 2: F1 ROUGE scores on CNN/Diary Mail dataset. In row (B), the first three sentences were taken as summaries. Part (C) are the results obtained without paired data. The results with symbol † are directly obtained from corresponding papers.

Methods		R-1	R-2	R-L
(A) Training with paired data (supervised)		49.62	34.10	46.42
(B)Lead-15 baseline		30.08	18.24	27.74
(C) Unpaired	(C-1) Pre-trained generator	28.36	16.73	26.48
	(C-2) WGAN	38.15	24.60	35.27
	(C-3) **Adversarial REINFORCE**	**41.25**	**26.54**	**37.76**

Table 3: F1 ROUGE scores on Chinese Gigaword. In row (B), we selected the article's first fifteen words as its summary. Part (C) are the results obtained without paired data.

trained generator and the poor pre-training result indicates that the data distributions of two datasets are quite different. We find that using sentences from another dataset yields lower ROUGE scores on the target testing set (parts (E) v.s. (C)) due to the mismatch word distributions between the summaries of the source and target domains. However, the discriminator still regularizes the generated word sequence. After unpaired training, the model enhanced the ROUGE scores of the pre-trained model (rows (E-2), (E-3) v.s. (E-1)) and it also surpassed the trivial baselines in part (B).

6.5 GAN Training

In this section, we discuss the performance of two GAN training methods. As shown in the Table 1, in English Gigaword, our proposed adversarial REINFORCE method performed better than WGAN. However, in Table 2, our proposed method slightly outperformed by WGAN. In addition, we find that when training with WGAN, convergence is faster. Because WGAN directly evaluates the distance between the continuous distribution from generator and the discrete distribution from real data, the distribution was sharpened at an early stage in training. This caused generator to converge to a relatively poor place. On the other hand, when training with REINFORCE, generator keeps seeking the network parameters that can better fool discriminator. We believe that training GAN on language generation with this method is worth exploring.

6.6 Chinese Gigaword

The Chinese Gigaword is a long text summarization dataset composed of paired headlines and news. Unlike the input news in English Gigaword, the news in Chinese Gigaword consists of several sentences. The results are shown in Table 3. Row (A) lists the results using 1.1M document-summary pairs to directly train the generator without the reconstructor and discriminator: this is the upper bound of the proposed approach. In row (B), we simply took the first fifteen words in a document as its summary. The number of words was chosen to optimize the evaluation metrics. Part (C) are the results obtained in the scenario without paired data. The discriminator took the summaries in the training set as real data. We show the results of the pre-trained generator in row (C-1); rows (C-2) and (C-3) are the results for the two GAN training methods respectively. We find that despite the performance gap between the unpaired and supervised methods (rows (C-2), (C-3) v.s. (A)), the proposed method yielded much better performance than the trivial baselines (rows (C-2), (C-3) v.s. (B)).

7 Conclusion and Future Work

Using GAN, we propose a model that encodes text as a human-readable summary, learned without document-summary pairs. In future work, we hope to use extra discriminators to control the style and sentiment of the generated summaries.

References

Martin Arjovsky, Soumith Chintala, and Lon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *HLT-NAAC*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *NIPS*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *ACL*.

Jiwei Li, Will Monroe, Tianlin Shi, Sbastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *In Text summarization branches out: ACL workshop*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. *EMNLP*.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *EMNLP*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. *CVPR*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *EMNLP*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *NIPS*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *ACL*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *NIPS*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. *AAAI*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. *ACL*.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*.

Joint Multitask Learning for Community Question Answering Using Task-Specific Embeddings

Shafiq Joty[†], **Lluís Màrquez**[*] **and Preslav Nakov**[‡]
[†]Nanyang Technological University, Singapore
[⋆]Amazon, Barcelona, Spain
[‡]Qatar Computing Research Institute, HBKU, Qatar
`srjoty@ntu.edu.sg, lluismv@amazon.com, pnakov@qf.org.qa`

Abstract

We address jointly two important tasks for Question Answering in community forums: given a new question, (*i*) *find related existing questions*, and (*ii*) *find relevant answers to this new question*. We further use an auxiliary task to complement the previous two, i.e., (*iii*) *find good answers with respect to the thread question in a question-comment thread*. We use deep neural networks (DNNs) to learn meaningful task-specific embeddings, which we then incorporate into a conditional random field (CRF) model for the multitask setting, performing joint learning over a complex graph structure. While DNNs alone achieve competitive results when trained to produce the embeddings, the CRF, which makes use of the embeddings and the dependencies between the tasks, improves the results significantly and consistently across a variety of evaluation metrics, thus showing the complementarity of DNNs and structured learning.

1 Introduction and Motivation

Question answering web forums such as StackOverflow, Quora, and Yahoo! Answers usually organize their content in topically-defined forums containing multiple *question–comment threads*, where a question posed by a user is often followed by a possibly very long list of comments by other users, supposedly intended to answer the question. Many forums are not moderated, which often results in noisy and redundant content.

Within community Question Answering (cQA) forums, two subtasks are of special relevance when a user poses a new question to the website (Hoogeveen et al., 2018; Lai et al., 2018): (*i*) finding similar questions (*question-question relatedness*), and (*ii*) finding relevant answers to the new question, if they already exist (*answer selection*).

Both subtasks have been the focus of recent research as they result in end-user applications. The former is interesting for a user who wants to explore the space of similar questions in the forum and to decide whether to post a new question. It can also be relevant for the forum owners as it can help detect redundancy, eliminate question duplicates, and improve the overall forum structure. Subtask (*ii*) on the other hand is useful for a user who just wants a quick answer to a specific question, without the need of digging through the long answer threads and winnowing good from bad comments or without having to post a question and then wait for an answer.

Obviously, the two subtasks are interrelated as the information needed to answer a new question is usually found in the threads of highly related questions. Here, we focus on jointly solving the two subtasks with the help of yet another related subtask, i.e., determining whether a comment within a question-comment thread is a good answer to the question heading that thread.

An example is shown in Figure 1. A new question q is posed for which several potentially related questions are identified in the forum (e.g., by using an information retrieval system); q_i in the example is one of these existing questions. Each retrieved question comes with an associated thread of comments; c^i_m represents one comment from the thread of question q_i. Here, c^i_m is a *good* answer for q_i, q_i is indeed a question *related* to q, and consequently c^i_m is a *relevant* answer for the new question q. This is the setting of SemEval-2016 Task 3, and we use its benchmark datasets.

Our approach has two steps. First, a deep neural network (DNN) in the form of a feed-forward neural network is trained to solve each of the three subtasks separately, and the subtask-specific hidden layer activations are taken as embedded feature representations to be used in the second step.

---

[*] Work conducted while this author was at QCRI, HBKU.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4196–4207
Brussels, Belgium, October 31 - November 4, 2018.

q: **"How can I extend a family visit visa?"**

q_i: "Dear All; I wonder if anyone knows the procedure how I can extend the family visit visa for my wife beyond 6 months. I already extended it for 5 months and is 6 months running. I would like to get it extended for couple of months more.Any suggestion is highly appreciable.Thanks"

c_m^i: "You can get just another month's extension before she completes 6 months by presenting to immigration office a confirmed booking of her return ticket which must not exceed 7 months."

Figure 1: Example of the three pieces of information in the cQA problems addressed in this paper.

Then, a conditional random field (CRF) model uses these embeddings and performs joint learning with global inference to exploit the dependencies between the subtasks.

A key strength of DNNs is their ability to learn nonlinear interactions between underlying features through specifically-designed hidden layers, and also to learn the features (e.g., vectors for words and documents) automatically. This capability has led to gains in many unstructured output problems. DNNs are also powerful for structured output problems. Previous work has mostly relied on recurrent or recursive architectures to propagate information through hidden layers, but has been disregarding the modeling strength of structured conditional models, which use global inference to model consistency in the output structure (i.e., the class labels of all nodes in a graph). In this work, we explore the idea that combining simple DNNs with structured conditional models can be an effective and efficient approach for cQA subtasks that offers the best of both worlds.

Our experimental results show that: (*i*) DNNs already perform very well on the question-question similarity and answer selection subtasks; (*ii*) strong dependencies exist between the subtasks under study, especially answer-goodness and question-question-relatedness influence answer-selection significantly; (*iii*) the CRFs exploit the dependencies between subtasks, providing sizeably better results that are on par or above the state of the art. In summary, we demonstrate the effectiveness of this marriage of DNNs and structured conditional models for cQA subtasks, where a feed-forward DNN is first used to build vectors for each individual subtask, which are then "reconciled" in a multitask CRF.

2 Related Work

Various neural models have been applied to cQA tasks such as *question-question similarity* (dos Santos et al., 2015; Lei et al., 2016; Wang et al., 2018) and *answer selection* (Wang and Nyberg, 2015; Qiu and Huang, 2015; Tan et al., 2015; Chen and Bunescu, 2017; Wu et al., 2018). Most of this work used advanced neural network architectures based on convolutional neural networks (CNN), long short-term memory (LSTM) units, attention mechanism, etc. For instance, dos Santos et al. (2015) combined CNN and bag of words for comparing questions. Tan et al. (2015) adopted an attention mechanism over bidirectional LSTMs to generate better answer representations, and Lei et al. (2016) combined recurrent and CNN models for question representation. In contrast, here we use a simple DNN model, i.e., a feed-forward neural network, which we only use to generate task-specific embeddings, and we defer the joint learning with global inference to the structured model.

From the perspective of modeling cQA subtasks as structured learning problems, there is a lot of research trying to exploit the correlations between the comments in a question–comment thread. This has been done from a feature engineering perspective, by modeling a comment in the context of the entire thread (Barrón-Cedeño et al., 2015), but more interestingly by considering a thread as a structured object, where comments are to be classified as *good* or *bad* answers collectively. For example, Zhou et al. (2015) treated the answer selection task as a sequence labeling problem and used recurrent convolutional neural networks and LSTMs. Joty et al. (2015) modeled the relations between pairs of comments at any distance in the thread, and combined the predictions of local classifiers using graph-cut and Integer Linear Programming. In a follow up work, Joty et al. (2016) also modeled the relations between all pairs of comments in a thread, but using a fully-connected pairwise CRF model, which is a joint model that integrates inference within the learning process using global normalization. Unlike these models, we use DNNs to induce task-specific embeddings, and, more importantly, we perform multitask learning of three different cQA subtasks, thus enriching the relational structure of the graphical model.

We solve the three cQA subtasks jointly, in a multitask learning framework. We do this using the datasets from the SemEval-2016 Task 3 on *Community Question Answering* (Nakov et al., 2016b), which are annotated for the three subtasks, and we compare against the systems that participated in that competition. In fact, most of these systems did not try to exploit the interaction between the subtasks or did so only as a pipeline. For example, the top two systems, SUPER TEAM (Mihaylova et al., 2016) and KELP (Filice et al., 2016), stacked the predicted labels from two subtasks in order to solve the main answer selection subtask using SVMs. In contrast, our approach is neural, it is based on joint learning and task-specific embeddings, and it is also lighter in terms of features.

In work following the competition, Nakov et al. (2016a) used a triangulation approach to answer ranking in cQA, modeling the three types of similarities occurring in the triangle formed by the original question, the related question, and an answer to the related comment. However, theirs is a pairwise ranking model, while we have a joint model. Moreover, they focus on one task only, while we use multitask learning. Bonadiman et al. (2017) proposed a multitask neural architecture where the three tasks are trained together with the same representation. However, they do not model comment-comment interactions in the same question-comment thread nor do they train task-specific embeddings, as we do.

The general idea of combining DNNs and structured models has been explored recently for other NLP tasks. Collobert et al. (2011) used Viterbi inference to train their DNN models to capture dependencies between word-level tags for a number of sequence labeling tasks: part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. Huang et al. (2015) proposed an LSTM-CRF framework for such tasks. Ma and Hovy (2016) included a CNN in the framework to compute word representations from character-level embeddings. While these studies consider tasks related to constituents in a sentence, e.g., words and phrases, we focus on methods to represent comments and to model dependencies between comment-level tags. We also experiment with arbitrary graph structures in our CRF model to model dependencies at different levels.

3 Learning Approach

Let q be a newly-posed question, and c_m^i denote the m-th comment ($m \in \{1, 2, \ldots, M\}$) in the answer thread for the i-th potentially related question q_i ($i \in \{1, 2, \ldots, I\}$) retrieved from the forum. We can define three cQA subtasks: (*A*) classify each comment c_m^i in the thread for question q_i as *Good* vs. *Bad* with respect to q_i; (*B*) determine, for each retrieved question q_i, whether it is *Related* to the new question q in the sense that a good answer to q_i might also be a good answer to q; and finally, (*C*) classify each comment c_m^i in each answer thread as either *Relevant* or *Irrelevant* with respect to the new question q.

Let $y_{i,m}^a \in \{Good, Bad\}$, $y_i^b \in \{Related, Not\text{-}related\}$, and $y_{i,m}^c \in \{Relevant, Irrelevant\}$ denote the corresponding output labels for subtasks A, B, and C, respectively. As argued before, subtask C depends on the other two subtasks. Intuitively, if c_m^i is a good comment with respect to the existing question q_i, and q_i is related to the new question q (subtask A), then c_m^i is likely to be a relevant answer to q. Similarly, subtask B can benefit from subtask C: if comment c_m^i in the answer thread of q_i is relevant with respect to q, then q_i is likely to be related to q.

We propose to exploit these inherent correlations between the cQA subtasks as follows: (*i*) by modeling their interactions in the input representations, i.e., in the feature space of (q, q_i, c_m^i), and more importantly, (*ii*) by capturing the dependencies between the output variables $(y_{i,m}^a, y_i^b, y_{i,m}^c)$. Moreover, we cast each cQA subtask as a structured prediction problem in order to model the dependencies between output variables of the same type. Our intuition is that if two comments c_m^i and c_n^i in the same thread are similar, then they are likely to have the same labels for both subtask A and subtask C, i.e., $y_{i,m}^a \approx y_{i,n}^a$, and $y_{i,m}^c \approx y_{i,n}^c$. Similarly, if two pre-existing questions q_i and q_j are similar, they are also likely to have the same labels, i.e., $y_i^b \approx y_j^b$.

Our framework works in two steps. First, we use a DNN, specifically, a feed-forward NN, to learn task-specific embeddings for the three subtasks, i.e., output embeddings $\mathbf{x}_{i,m}^a$, $\mathbf{x}_i^b$ and $\mathbf{x}_{i,m}^c$ for subtasks A, B and C (Figure 2a). The DNN uses syntactic and semantic embeddings of the input elements, their interactions, and other similarity features between them and, as a by-product, learns the output embeddings for each subtask.

In the second step, a structured conditional model operates on subtask-specific embeddings from the DNNs and captures the dependencies between the

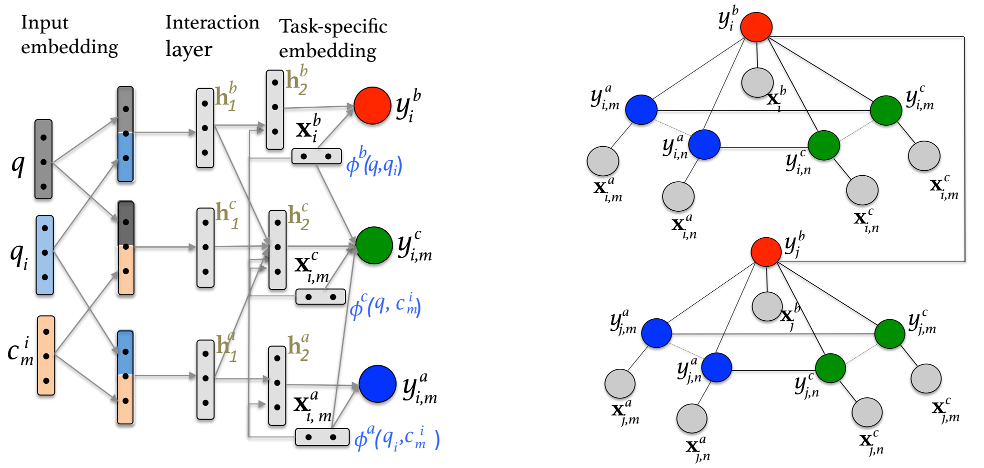

(a) Our feed-forward neural networks

(b) Our globally normalized joint conditional model

Figure 2: Graphical representation of our cQA framework. On the left (a), we have three feed-forward neural networks to learn task-specific embeddings for the three cQA subtasks. On the right (b), a global conditional random field (CRF) models intra- and inter-subtask dependencies.

subtasks, between existing questions, and between comments for an existing question (Figure 2b). Below, we describe the two steps in detail.

3.1 Neural Models for cQA Subtasks

Figure 2a depicts our complete neural framework for the three subtasks. The input is a tuple (q, q_i, c^i_m) consisting of a new question q, a retrieved question q_i, and a comment c^i_m from q_i's answer thread. We first map the input elements to fixed-length vectors $(\mathbf{z}_q, \mathbf{z}_{q_i}, \mathbf{z}_{c^i_m})$ using their syntactic and semantic *embeddings*. Depending on the requirements of the subtasks, the network then models the interactions between the inputs by passing their embeddings through non-linear hidden layers $\nu(\cdot)$. Additionally, the network also considers *pairwise* similarity features $\phi(\cdot)$ between two input elements that go directly to the output layer, and also through the last hidden layer. The pairwise features together with the activations at the final hidden layer constitute the task-specific embeddings for each subtask t: $\mathbf{x}^t_i = [\nu^t(\cdot), \phi^t(\cdot)]$. The final layer defines a Bernoulli distribution for each subtask $t \in \{a, b, c\}$:

$$p(y^t_i|q, q_i, c^i_m, \theta) = \mathrm{Ber}(y^t_i|\,\mathrm{sig}(\mathbf{w}^T_t \mathbf{x}^t_i)) \qquad (1)$$

where $\mathbf{x}^t_i$, $\mathbf{w}_t$, and y^t_i are the task-specific embedding, the output layer weights, and the prediction variable for subtask t, respectively, and $\mathrm{sig}(\cdot)$ refers to the sigmoid function.

We train the models by minimizing the cross-entropy between the predicted distribution and the gold labels. The main difference between the models is how they compute the task-specific embeddings $\mathbf{x}^t_i$ for subtask t.

Neural Model for Subtask A. The feed-forward network for subtask A is shown in the lower part of Figure 2a. To determine whether a comment c^i_m is *good* with respect to the thread question q_i, we model the interactions between c^i_m and q_i by merging their embeddings $\mathbf{z}_{c^i_m}$ and $\mathbf{z}_{q_i}$, and passing them through a hidden layer:

$$\mathbf{h}^a_1 = f(U^a[\mathbf{z}_{q_i}, \mathbf{z}_{c^i_m}]) \qquad (2)$$

where U^a is the weight matrix from the inputs to the first hidden units, f is a non-linear activation function. The activations are then fed to a final subtask-specific hidden layer, which combines these signals with the pairwise similarity features $\phi^a(q_i, c^i_m)$. Formally,

$$\mathbf{h}^a_2 = f(V^a[\mathbf{h}^a_1, \phi^a(q_i, c^i_m)]) \qquad (3)$$

where V^a is the weight matrix. The task-specific output embedding is formed by merging $\mathbf{h}^a_2$ and $\phi^a(q_i, c^i_m)$; $\mathbf{x}^a_{i,m} = [\mathbf{h}^a_2, \phi^a(q_i, c^i_m)]$.

Neural Model for Subtask B. To determine whether an existing question q_i is *related* to the new question q, we model the interactions between q and q_i using their embeddings and pairwise similarity features similarly to subtask A.

The upper part of Figure 2a shows the network. The transformation is defined as follows:

$$\mathbf{h}^b_1 = f(U^b[\mathbf{z}_q, \mathbf{z}_{q_i}]); \mathbf{h}^b_2 = f(V^b[\mathbf{h}^b_1, \phi^b(q, q_i)])$$

where U^b and V^b are the weight matrices in the first and second hidden layer. The task-specific embedding is formed by $\mathbf{x}_i^b = [\mathbf{h}_2^b, \phi^b(q, q_i)]$.

Neural Model for Subtask C. The network for subtask C is shown in the middle of Figure 2a. To decide if a comment c_m^i in the thread of q_i is *relevant* to q, we consider how related q_i is to q, and how useful c_m^i is to answer q_i. Again, we model the direct interactions between q and c_m^i using pairwise features $\phi^c(q, c_m^i)$ and a hidden layer transformation $\mathbf{h}_1^c = f(U^c[\mathbf{z}_q, \mathbf{z}_{c_m^i}])$, where U^c is a weight matrix. We then include a second hidden layer to combine the activations from different inputs and pairwise similarity features. Formally,

$$\mathbf{h}_2^c = f(V^c[\mathbf{h}_1^a, \mathbf{h}_1^b, \mathbf{h}_1^c, \phi^a(q_i, c_m^i), \phi^b(q, q_i), \phi^c(q, c_m^i)])$$

The final task-specific embedding for subtask C is formed as $\mathbf{x}_i^c = [\mathbf{h}_2^c, \phi^a(q_i, c_m^i), \phi^b(q, q_i), \phi^c(q, c_m^i)]$.

3.2 Joint Learning with Global Inference

One simple way to exploit the interdependencies between the subtask-specific embeddings ($\mathbf{x}_{i,m}^a$, $\mathbf{x}_i^b$, $\mathbf{x}_{i,m}^c$) is to precompute the predictions for some subtasks (A and B), and then to use the predictions as features for the other subtask (C). However, as shown later in Section 6, such a *pipeline* approach propagates errors from one subtask to the subsequent ones. A more robust way is to build a joint model for all subtasks.

We could use the full DNN network in Figure 2a to learn the classification functions for the three subtasks *jointly* as follows:

$$p(y_{i,m}^a, y_i^b, y_{i,m}^c|\theta) = p(y_{i,m}^a|\theta_a)p(y_i^b|\theta_b)p(y_{i,m}^c|\theta_c) \tag{4}$$

where $\theta = [\theta_a, \theta_b, \theta_c]$ are the model parameters.

However, this has two key limitations: (*i*) it assumes conditional independence between the subtasks given the parameters; (*ii*) the scores are normalized locally, which leads to the so-called *label bias* problem (Lafferty et al., 2001), i.e., the features for one subtask would have no influence on the other subtasks.

Thus, we model the dependencies between the output variables by learning (globally normalized) node and edge factor functions that jointly optimize a global performance criterion. In particular, we represent the cQA setting as a large undirected graph $G{=}(V, E){=}(V_a{\cup}V_b{\cup}V_c, E_{aa}{\cup}E_{bb}{\cup}E_{cc}{\cup}E_{ac}{\cup}E_{bc}{\cup}E_{ab})$. As shown in Figure 2b, the graph contains six subgraphs: $G_a{=}(V_a, E_{aa})$, $G_b{=}(V_b, E_{bb})$ and $G_c{=}(V_c, E_{cc})$ are associated with the three subtasks, while the bipartite subgraphs $G_{ac}{=}(V_a \cup V_c, E_{ac})$, $G_{bc}{=}(V_b \cup V_c, E_{bc})$ and $G_{ab}{=}(V_a \cup V_b, E_{ab})$ connect nodes across tasks.

We associate each node $u \in V_t$ with an input vector $\mathbf{x}_u$, representing the embedding for subtask t, and an output variable y_u, representing the class label for subtask t. Similarly, each edge $(u, v) \in E_{st}$ is associated with an input feature vector $\mu(\mathbf{x}_u, \mathbf{x}_v)$, derived from the node-level features, and an output variable $y_{uv} \in \{1, 2, \cdots, L\}$, representing the state transitions for the pair of nodes.[1] For notational simplicity, here we do not distinguish between comment and question nodes, rather we use u and v as general indices. We define the following joint conditional distribution:

$$p(\mathbf{y}|\theta, \mathbf{x}) = \frac{1}{Z(\theta, \mathbf{x})} \prod_{t \in \tau} \Big[\prod_{u \in V_t} \psi_n(y_u|\mathbf{x}, \mathbf{w}_n^t) \Big] \prod_{(s,t) \in \tau \times \tau} \Big[\prod_{(u,v) \in E_{st}} \psi_e(y_{uv}|\mathbf{x}, \mathbf{w}_e^{st}) \Big] \tag{5}$$

where $\tau = \{a, b, c\}$, $\psi_n(\cdot)$ and $\psi_e(\cdot)$ are node and edge *factors*, respectively, and $Z(\cdot)$ is a global normalization constant. We use log-linear factors:

$$\psi_n(y_u|\mathbf{x}, \mathbf{w}_n^t) = \exp(\sigma(y_u, \mathbf{x})^T \mathbf{w}_n^t) \tag{6}$$

$$\psi_e(y_{uv}|\mathbf{x}, \mathbf{w}_e^{st}) = \exp(\sigma(y_{uv}, \mathbf{x})^T \mathbf{w}_e^{st}) \tag{7}$$

where $\sigma(\cdot)$ is a feature vector derived from the inputs and the labels.

This model is essentially a pairwise conditional random field (Murphy, 2012). The global normalization allows CRFs to surmount the *label bias* problem, allowing them to take long-range interactions into account. The objective in Equation 5 is a convex function, and thus we can use gradient-based methods to find the global optimum. The gradients have the following form:

$$f'(\mathbf{w}_n^t) = \sum_{u \in V_t} \sigma(y_u, \mathbf{x}) - \mathbb{E}[\sigma(y_u, \mathbf{x})] \tag{8}$$

$$f'(\mathbf{w}_e^{st}) = \sum_{(u,v) \in E_{st}} \sigma(y_{uv}, \mathbf{x}) - \mathbb{E}[\sigma(y_{uv}, \mathbf{x})] \tag{9}$$

where $\mathbb{E}[\phi(\cdot)]$ is the expected feature vector.

Training and Inference. Traditionally, CRFs have been trained using offline methods like LBFGS (Murphy, 2012). Online training using

---

[1] To avoid visual clutter, the input features and the output variables for the edges are not shown in Figure 2b.

first-order methods such as stochastic gradient descent was proposed by Vishwanathan et al. (2006). Since our DNNs are trained with the RMSprop online adaptive algorithm (Tieleman and Hinton, 2012), in order to compare our two models, we use RMSprop to train our CRFs as well.

For our CRF models, we use Belief Propagation, or BP, (Pearl, 1988) for inference. BP converges to an exact solution for trees. However, exact inference is intractable for graphs with loops. Despite this, Pearl (1988) advocated for the use of BP in loopy graphs as an approximation. Even though BP only gives approximate solutions, it often works well in practice for loopy graphs (Murphy et al., 1999), outperforming other methods such as mean field (Weiss, 2001).

Variations of Graph Structures. A crucial advantage of our CRFs is that we can use arbitrary graph structures, which allows us to capture dependencies between different types of variables: (*i*) *intra-subtask*, for variables of the same subtask, e.g., y_i^b and y_j^b in Figure 2b, and (*ii*) *across-subtask*, for variables of different subtasks.

For intra-subtask, we explore *null* (i.e., no connection between nodes) and *fully-connected* relations. For subtasks A and C, the intra-subtask connections are restricted to the nodes inside a thread, e.g., we do not connect $y_{i,m}^c$ and $y_{j,m}^c$ in Figure 2b.

For across-subtask, we explored three types of connections depending on the subtasks involved: (*i*) *null* or no connection between subtasks, (*ii*) *1:1* connection for A-C, where the corresponding nodes of the two subtasks in a thread are connected, e.g., $y_{i,m}^a$ and $y_{i,m}^c$ in Figure 2b, and (*iii*) *M:1* connection to B, where we connect all the nodes of C or A to the thread-level B node. Each configuration of intra- and across-connections yields a different CRF model. Figure 2b shows one such model for two threads each containing two comments, where all subtasks have fully-connected intra-subtask links, 1:1 connection for A-C, and M:1 for C-B and A-B.

4 Features for the DNN Models

We have two types of features: (*i*) *input embeddings*, for q, q_i and c_m^i, and (*ii*) *pairwise features*, for (q, q_i), (q, c_m^i), and (q_i, c_m^i) — see Figure 2a.

4.1 Input Embeddings

We use three types of pre-trained vectors to represent a question (q or q_i) or a comment (c_m^i):

GOOGLE VECTORS. 300-dimensional embedding vectors, trained on 100 billion words from Google News (Mikolov et al., 2013). The embedding for a question (or comment) is the average of the word embeddings it is composed of.

SYNTAX. We parse the question (or comment) using the Stanford neural parser (Socher et al., 2013), and we use the final 25-dimensional vector produced internally as a by-product of parsing.

QL VECTORS. We use fine-tuned word embeddings pretrained on all the available in-domain Qatar Living data (Mihaylov and Nakov, 2016).

4.2 Pairwise Features

We extract pairwise features for each of (q, q_i), (q, c_m^i), and (q_i, c_m^i) pairs. These include:

COSINES. We compute cosines using the above vectors: $\cos(q, q_i)$, $\cos(q, c_m^i)$ and $\cos(q_i, c_m^i)$.

MT FEATURES. We use the following machine translation evaluation metrics: (1) BLEU (Papineni et al., 2002); (2) NIST (Doddington, 2002); (3) TER v0.7.25 (Snover et al., 2006); (4) METEOR v1.4 (Lavie and Denkowski, 2009); (5) Unigram PRECISION; (6) Unigram RECALL.

BLEU COMPONENTS. We further use various components involved in the computation of BLEU:[2] n-gram precisions, n-gram matches, total number of n-grams (n=1,2,3,4), lengths of the hypotheses and of the reference, length ratio between them, and BLEU's brevity penalty.

QUESTION-COMMENT RATIO. (1) question-to-comment count ratio in terms of sentences/tokens/nouns/verbs/adjectives/adverbs/pronouns; (2) question-to-comment count ratio of words that are not in WORD2VEC's Google News vocabulary.

4.3 Node Features

COMMENT FEATURES. These include number of (1) nouns/verbs/adjectives/adverbs/pronouns, (2) URLs/images/emails/phone numbers, (3) tokens/sentences, (4) positive/negative smileys, (5) single/double/triple exclamation/interrogation symbols, (6) interrogative sentences, (7) 'thank' mentions, (8) words that are not in WORD2VEC's Google News vocabulary. Also, (9) average number of tokens, and (10) word type-to-token ratio.

META FEATURES. (1) is the person answering the question the one who asked it; (2) reciprocal rank of comment c_m^i in the thread of q_i, i.e., $1/m$;

---

[2] BLEU FEATURES and BLEU COMPONENTS (Guzmán et al., 2016a,b) are ported from an MT evaluation framework (Guzmán et al., 2015; Guzmán et al., 2017) to cQA.

	batch	dropout	reg. str	inter. layer	task-spec. layer
A	16	0.3	0.001	10	125
B	25	0.2	0.05	5	75
C	32	0.3	0.0001	15	50

Table 1: Best setting for DNNs, as found on DEV.

Subtask A			
System	MAP	AvgRec	MRR
Random order	52.80	66.52	58.71
Chronological order	59.53	72.60	67.83
ConvKN (second at SE-2016)	77.66	88.05	84.93
Kelp (best at SE-2016)	79.19	88.82	86.42
DNN$_\mathbf{A}$ (subtask A network)	**76.20**	**86.52**	**84.95**
Subtask B			
System	MAP	AvgRec	MRR
Random order	46.98	67.92	50.96
IR order	74.75	88.30	83.79
ConvKN (second at SE-2016)	76.02	90.70	84.64
UH-PRHLT (best at SE-2016)	76.70	90.31	83.02
DNN$_\mathbf{B}$ (subtask B network)	**76.27**	**90.27**	**83.57**
DNN$_B$ + A gold labels	76.10	89.96	83.62
DNN$_B$ + C gold labels	77.19	90.78	83.73
DNN$_B$ + A and C gold labels	77.12	90.71	83.73
Subtask C			
System	MAP	AvgRec	MRR
Random order	15.01	11.44	15.19
IR+Chron. order	40.36	45.97	45.83
Kelp (second at SE-2016)	52.95	59.27	59.23
SUper team (best at SE-2016)	55.41	60.66	61.48
DNN$_\mathbf{C}$ (subtask C network)	**54.24**	**58.30**	**61.47**
DNN$_C$ + A gold labels	61.14	66.67	66.86
DNN$_C$ + B gold labels	56.29	61.11	62.67
DNN$_C$ + A and B gold labels	63.49	71.16	68.19

Table 2: Results for our DNN models on all cQA subtasks, compared to the top-2 systems from SemEval-2016 Task 3. Inter-subtask dependencies are explored using gold output labels.

(3) reciprocal rank of c_m^i in the list of comments for q, i.e., $1/[m+10\times(i-1)]$; and (4) reciprocal rank of question q_i in the list for q, i.e., $1/i$.

5 Data and Settings

We experiment with the data from SemEval-2016 Task 3 (Nakov et al., 2016b). Consistently with our notation from Section 3, it features three subtasks: subtask A (i.e., whether a comment c_m^i is a good answer to the question q_i in the thread), subtask B (i.e., whether the retrieved question q_i is related to the new question q), and subtask C (i.e., whether the comment c_m^i is a relevant answer for the new question q). Note that the two main subtasks we are interested in are B and C.

DNN Setting. We preprocess the data using min-max scaling. We use RMSprop[3] for learning, with parameters set to the values suggested by Tieleman and Hinton (2012). We use up to 100 epochs with patience of 25, rectified linear units (ReLU) as activation functions, l_2 regularization on weights, and dropout (Srivastava et al., 2014) of hidden units. See Table 1 for more detail.

CRF Setting. For the CRF model, we initialize the node-level weights from the output layer weights of the DNNs, and we set the edge-level weights to 0. Then, we train using RMSprop with loopy BP. We regularize the node parameters according to the best settings of the DNN: 0.001, 0.05, and 0.0001 for A, B, and C, respectively.

6 Results and Discussion

Below, we first present the evaluation results using DNN models (Section 6.1). Then, we discuss the performance of the joint models (Section 6.2).

6.1 Results for the DNN Models

Table 2 shows the results for our individual DNN models (rows in boldface) for subtasks A, B and C on the TEST set.

We report three ranking-based measures that are commonly accepted in the IR community: mean average precision (MAP), which was the official evaluation measure of SemEval-2016, average recall (AvgRec), and mean reciprocal rank (MRR).

For each subtask, we show two baselines and the results of the top-2 systems at SemEval. The first baseline is a random ordering of the questions/comments, assuming no knowledge about the subtask. The second baseline keeps the chronological order of the comments for subtask A, of the question ranking from the IR engine for subtask B, and both for subtask C.

We can see that the individual DNN models for subtasks B and C are very competitive, falling between the first and the second best at SemEval-2016. For subtask A, our model is weaker, but, as we will see below, it can help improve the results for subtasks B and C, which are our focus here.

Looking at the results for subtask C, we can see that sizeable gains are possible when using gold labels for subtasks A and B as features to DNN$_C$, e.g., adding gold A labels yields +6.90 MAP points.

Similarly, using gold labels for subtask B adds +2.05 MAP points absolute. Moreover, the gain is cumulative: using the two gold labels together

---

[3] Other adaptive algorithms such as ADAM (Kingma and Ba, 2014) or ADADELTA (Zeiler, 2012) were slightly worse.

yields +9.25 MAP points. The same behavior is observed for the other evaluation measures. Of course, as we use gold labels, this is an upper bound on performance, but it justifies our efforts towards a joint multitask learning model.

6.2 Results for the Joint Model

Below we discuss the evaluation results for the joint model. We focus on subtasks B and C, which are the main target of our study.

Results for Subtask C. Table 3 compares several variants of the CRF model for joint learning, which we described in Section 3.2 above.

Row 1 shows the results for our individual DNN_C model. The following rows 2–4 present a *pipeline* approach, where we first predict labels for subtasks A and B and then we add these predictions as features to DNN_C. This is prone to error propagation, and improvements are moderate and inconsistent across the evaluation measures.

The remaining rows correspond to variants of our CRF model with different graph structures. Overall, the improvements over DNN_C are more sizeable than for the pipeline approach (with one single exception out of 24 cases); they are also more consistent across the evaluation measures, and the improvements in MAP over the baseline range from +0.96 to +1.76 points absolute.

Rows 5–8 show the impact of adding connections to subtasks A and B when solving subtask C (see Figure 2b). Interestingly, we observe the same pattern as with the gold labels: the A-C and B-C connections help individually and in combination, with A-C being more helpful. Yet, further adding A-B does not improve the results (row 8).

Note that the locally normalized joint model in Eq. 4 yields much lower results than the globally normalized CRF_{all} (row 8): 54.32, 59.87, and 61.76 in MAP, AvgRec and MRR (figures not included in the table for brevity). This evinces the problems with the conditional independence assumption and the local normalization in the model.

Finally, rows 9–12 explore variants of the best system from the previous set (row 7), which has connections between subtasks only. Rows 9–12 show the results when using subgraphs for A, B and C that are fully connected (i.e., for all pairs).
We can see that none of these variants yields improvements over the model from row 7, i.e., the fine-grained relations between comments in the threads and between the different related questions do not seem to help solve subtask C in the joint model. Note that our scores from row 7 are better than the best results achieved by a system at SemEval-2016 Task 3 subtask C: 56.00 vs. 55.41 on MAP, and 63.25 vs. 61.48 on MRR.

Results for Subtask B. Next, we present in Table 4 similar experiments, but this time with subtask B as the target, and we show some more measures (accuracy, precision, recall, and F_1).

Given the insights from Table 2 (where we used gold labels), we did not expect to see much improvements for subtask B. Indeed, as rows 2–4 show, using the pipeline approach, the IR measures are basically unaltered. However, classification accuracy improves by almost one point absolute, recall is also higher (trading for lower precision), and F_1 is better by a sizeable margin.

Coming to the joint models (rows 6–9), we can see that the IR measures improve consistently over the pipeline approach, even though not by much. The effect on accuracy-P-R-F_1 is the same as observed with the pipeline approach but with larger differences.[4] In particular, accuracy improves by more than two points absolute, and recall increases, which boosts F_1 to almost 60.

Row 5 is a special case where we only consider subtask B, but we do the learning and the inference over the set of ten related questions, exploiting their relations. This yields a slight increase in all measures; more importantly, it is crucial for obtaining better results with the joint models.

Rows 6–9 show results for various variants of the A-C and B-C architecture with fully connected B nodes, playing with the fine-grained connection of the A and C nodes. The best results are in this block, with increases over DNN_B in MAP (+0.61), AvgRec (+0.69) and MRR (+1.05), and especially in accuracy (+2.18) and F_1 (+11.25 points). This is remarkable given the low expectation we had about improving subtask B.

Note that the best architecture for subtask C from Table 3 (A-C and B-C with no fully connected B layer) does not yield good results for subtask B.
We speculate that subtask B is overlooked by the architecture, which has many more connections and parameters on the nodes for subtasks A and C (ten comments are to be classified for both subtask

---

[4] Note that we have a classification approach, which favors accuracy-P-R-F_1; if we want to improve the ranking measures, we should optimize for them directly.

#	*System*	*Comments*	*MAP* (Δ)	*AvgRec* (Δ)	*MRR* (Δ)
1	DNN$_C$	Subtask C network	54.24	58.30	61.47
2	DNN$_{C+PA}$	DNN$_C$ with A predicted labels	55.21 (+0.97)	58.36 (+0.06)	62.69 (+1.22)
3	DNN$_{C+PB}$	DNN$_C$ with B predicted labels	54.17 (-0.04)	58.17 (-0.13)	62.55 (+1.08)
4	DNN$_{C+PA+PB}$	DNN$_C$ with A and B predicted labels	55.11 (+0.90)	58.69 (+0.39)	60.10 (-1.37)
5	CRF$_{AC}$	CRF with A-C connections	55.42 (+1.18)	58.69 (+0.39)	63.25 (+1.78)
6	CRF$_{BC}$	CRF with B-C connections	55.20 (+0.96)	58.87 (+0.57)	62.30 (+0.83)
7	CRF$_{ACBC}$	CRF with A-C and B-C connections	**56.00** (+1.76)	60.20 (+1.90)	**63.25** (+1.78)
8	CRF$_{all}$	CRF with all pairwise connections	55.81 (+1.57)	60.15 (+1.85)	62.68 (+1.21)
9	CRF$_{ACBC,Cf}$	CRF$_{ACBC}$ with fully connected C	55.73 (+1.49)	59.77 (+1.47)	62.80 (+1.33)
10	CRF$_{ACBC,AfCf}$	CRF$_{ACBC}$ with fully connected A and C	55.54 (+1.30)	59.86 (+1.56)	62.54 (+1.07)
11	CRF$_{ACBC,BfCf}$	CRF$_{ACBC}$ with fully connected B and C	55.67 (+1.43)	**60.22** (+1.92)	62.80 (+1.33)
12	CRF$_{ACBC,f}$	CRF$_{ACBC}$ with all layers fully connected	55.81 (+1.57)	60.15 (+1.85)	**63.25** (+1.78)

Table 3: Performance of the *pipeline* and of the joint learning models on subtask C. The best results for each measure are in bold, and the gains over the single neural network (DNN$_C$) are shown in parentheses.

#	*System*	*Comments*	*MAP*	*AvgRec*	*MRR*	*Acc*	*P*	*R*	F_1
1	DNN$_B$	Subtask B network	76.27	90.27	83.57	76.39	**89.53**	33.05	48.28
2	DNN$_{B+PA}$	DNN$_B$ with A predicted labels	76.08	89.99	83.38	77.40	86.41	38.20	52.98
3	DNN$_{B+PC}$	DNN$_B$ with C predicted labels	76.33	90.38	83.62	77.40	83.19	40.34	54.34
4	DNN$_{B+PA+PC}$	DNN$_B$ with A and C predicted labels	76.43	90.34	83.62	77.11	78.74	42.92	55.56
5	CRF$_{Bf}$	CRF with fully connected B	76.41	90.34	83.81	77.00	84.62	37.76	52.23
6	CRF$_{ACBC,Bf}$	CRF$_{ACBC}$ with fully connected B	**76.89**	90.87	84.19	77.86	76.00	**48.93**	**59.53**
7	CRF$_{ACBC,AfBf}$	CRF$_{ACBC}$ with fully connected A and B	76.51	90.64	84.19	78.29	83.47	43.35	57.06
8	CRF$_{ACBC,BfCf}$	CRF$_{ACBC}$ with fully connected B and C	76.87	**90.96**	84.44	77.86	78.68	45.92	58.00
9	CRF$_{ACBC,f}$	CRF$_{ACBC}$ with all layers fully connected	76.25	90.38	**84.62**	**78.57**	81.20	46.35	59.02

Table 4: Performance of the *pipeline* and of the joint models on subtask B (best results in boldface).

A and C, while only one decision is to be made for the related question B).

Finally, note that our best results for subtask B are also slightly better than those for the best system at SemEval-2016 Task 3, especially on MRR.

7 Conclusion

We have presented a framework for multitask learning of two community Question Answering problems: question-question relatedness and answer selection. We further used a third, auxiliary one, i.e., finding the good comments in a question-comment thread. We proposed a two-step framework based on deep neural networks and structured conditional models, with a feed-forward neural network to learn task-specific embeddings, which are then used in a pairwise CRF as part of a multitask model for all three subtasks.

The DNN model has its strength in generating compact embedded representations for the subtasks by modeling interactions between different input elements.

On the other hand, the CRF is able to perform global inference over arbitrary graph structures accounting for the dependencies between subtasks to provide globally good solutions. The experimental results have proven the suitability of combining the two approaches. The DNNs alone already yielded competitive results, but the CRF was able to exploit the task-specific embeddings and the dependencies between subtasks to improve the results consistently across a variety of evaluation metrics, yielding state-of-the-art results.

In future work, we plan to model text complexity (Mihaylova et al., 2016), veracity (Mihaylova et al., 2018), speech act (Joty and Hoque, 2016), user profile (Mihaylov et al., 2015), trollness (Mihaylov et al., 2018), and goodness polarity (Balchev et al., 2016; Mihaylov et al., 2017). From a modeling perspective, we want to strongly couple CRF and DNN, so that the global errors are backpropagated from the CRF down to the DNN layers. It would be also interesting to extend the framework to a cross-domain (Shah et al., 2018) or a cross-language setting (Da San Martino et al., 2017; Joty et al., 2017). Trying an ensemble of neural networks with different initial seeds is another possible research direction.

Acknowledgments

The first author would like to thank the funding support from MOE Tier-1.

References

Daniel Balchev, Yasen Kiprov, Ivan Koychev, and Preslav Nakov. 2016. PMI-cool at SemEval-2016 Task 3: Experiments with PMI and goodness polarity lexicons for community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 844–850, San Diego, California, USA.

Alberto Barrón-Cedeño, Simone Filice, Giovanni Da San Martino, Shafiq Joty, Lluís Màrquez, Preslav Nakov, and Alessandro Moschitti. 2015. Thread-level information for comment classification in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 687–693, Beijing, China.

Daniele Bonadiman, Antonio Uva, and Alessandro Moschitti. 2017. Effective shared representations with multitask learning for community question answering. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, EACL '17, pages 726–732, Valencia, Spain.

Charles Chen and Razvan Bunescu. 2017. An exploration of data augmentation and RNN architectures for question ranking in community question answering. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, IJCNLP '17, pages 442–447, Taipei, Taiwan.

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

Giovanni Da San Martino, Salvatore Romeo, Alberto Barrón-Cedeño, Shafiq Joty, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2017. Cross-language question re-ranking. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1145–1148, Tokyo, Japan.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Francisco, California, USA.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning semantic relations between questions and answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 1116–1123, San Diego, California, USA.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2015. Pairwise neural machine translation evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 805–814, Beijing, China.

Francisco Guzmán, Shafiq R. Joty, Lluís Màrquez, and Preslav Nakov. 2017. Machine translation evaluation with neural networks. *Computer Speech & Language*, 45:180–200.

Francisco Guzmán, Lluís Màrquez, and Preslav Nakov. 2016a. Machine translation evaluation meets community question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 460–466, Berlin, Germany.

Francisco Guzmán, Preslav Nakov, and Lluís Màrquez. 2016b. MTE-NN at SemEval-2016 Task 3: Can machine translation evaluation help community question answering? In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, pages 887–895, San Diego, California, USA.

Doris Hoogeveen, Li Wang, Timothy Baldwin, and Karin M. Verspoor. 2018. Web forum retrieval and text analytics: A survey. *Foundations and Trends in Information Retrieval*, 12(1):1–163.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Shafiq Joty, Alberto Barrón-Cedeño, Giovanni Da San Martino, Simone Filice, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2015. Global thread-level inference for comment classification in community question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP '15, pages 573–578, Lisbon, Portugal.

Shafiq Joty and Enamul Hoque. 2016. Speech act modeling of written asynchronous conversations with task-specific embeddings and conditional structured models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 1746–1756, Berlin, Germany.

Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2016. Joint learning with global inference for comment classification in community question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '16, pages 703–713, San Diego, California, USA.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language learning with adversarial neural networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, CoNLL '17, pages 226–37, Vancouver, Canada.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, California, USA.

Tuan Manh Lai, Trung Bui, and Sheng Li. 2018. A review on deep learning techniques applied to answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 2132–2144, Santa Fe, New Mexico, USA.

Alon Lavie and Michael Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23(2–3):105–115.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez. 2016. Semi-supervised question retrieval with gated convolutions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '16, pages 1279–1289, San Diego, California, USA.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 1064–1074, Berlin, Germany.

Todor Mihaylov, Daniel Balchev, Yasen Kiprov, Ivan Koychev, and Preslav Nakov. 2017. Large-scale goodness polarity lexicons for community question answering. In *Proceedings of the 40th International Conference on Research and Development in Information Retrieval*, SIGIR '17, Tokyo, Japan.

Todor Mihaylov, Georgi Georgiev, and Preslav Nakov. 2015. Finding opinion manipulation trolls in news community forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, CoNLL '15, pages 310–314, Beijing, China.

Todor Mihaylov, Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Georgi Georgiev, and Ivan Koychev. 2018. The dark side of news community forums: Opinion manipulation trolls. *Internet Research*.

Todor Mihaylov and Preslav Nakov. 2016. SemanticZ at SemEval-2016 Task 3: Ranking relevant answers in community question answering using semantic similarity based on fine-tuned word embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 879–886, San Diego, California, USA.

Tsvetomila Mihaylova, Pepa Gencheva, Martin Boyanov, Ivana Yovcheva, Todor Mihaylov, Momchil Hardalov, Yasen Kiprov, Daniel Balchev, Ivan Koychev, Preslav Nakov, Ivelina Nikolova, and Galia Angelova. 2016. SUper Team at SemEval-2016 Task 3: Building a feature-rich system for community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 836–843, San Diego, California, USA.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadjov, and James Glass. 2018. Fact checking in community forums. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI '18, pages 879–886, New Orleans, Louisiana, USA.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '13, pages 746–751, Atlanta, Georgia, USA.

Kevin Murphy. 2012. *Machine Learning A Probabilistic Perspective*. The MIT Press.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 467–475, Stockholm, Sweden.

Preslav Nakov, Lluís Màrquez, and Francisco Guzmán. 2016a. It takes three to tango: Triangulation approach to answer ranking in community question answering. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP '16, pages 1586–1597, Austin, Texas, USA.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, abed Alhakim Freihat, James Glass, and Bilal Randeree. 2016b. SemEval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 525–545, San Diego, California, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meting of the Association for Computational Linguistics*, ACL '02, pages 311–318, Philadelphia, Pennsylvania, USA.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, California, USA.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of International Joint Conference on Artificial Intelligence*, IJCAI '15, pages 1305–1311, Buenos Aires, Argentina.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 694–699, Beijing, China.

Darsh J Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. 2018. Adversarial domain adaptation for duplicate question detection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, Brussels, Belgium.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas*, AMTA '06, pages 223–231, Cambridge, Massachusetts, USA.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 455–465, Sofia, Bulgaria.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. LSTM-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

T. Tieleman and G Hinton. 2012. *RMSprop*. COURSERA: Neural Networks

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. 2006. Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 969–976, Pittsburgh, Pennsylvania, USA.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 707–712, Beijing, China.

Pengwei Wang, Lei Ji, Jun Yan, Dejing Dou, Nisansa De Silva, Yong Zhang, and Lianwen Jin. 2018. Concept and attention-based cnn for question retrieval in multi-view learning. *ACM Trans. Intell. Syst. Technol.*, 9(4):41:1–41:24.

Yair Weiss. 2001. Comparing the mean field method and belief propagation for approximate inference in MRFs. In *Advanced Mean Field Methods*, pages 229–239, Cambridge, Massachusetts, USA. MIT Press.

Wei Wu, Xu SUN, and Houfeng WANG. 2018. Question condensing networks for answer selection in community question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 1746–1755, Melbourne, Australia.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Xiaoqiang Zhou, Baotian Hu, Qingcai Chen, Buzhou Tang, and Xiaolong Wang. 2015. Answer sequence learning with neural networks for answer selection in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 713–718, Beijing, China.

What Makes Reading Comprehension Questions Easier?

Saku Sugawara[♠], **Kentaro Inui**[♣◇], **Satoshi Sekine**[◇], **Akiko Aizawa**[♡♠]
[♠]The University of Tokyo, [♣]Tohoku University,
[◇]RIKEN AIP, [♡]National Institute of Informatics
`saku@is.s.u-tokyo.ac.jp`, `inui@ecei.tohoku.ac.jp`,
`satoshi.sekine@riken.jp`, `aizawa@nii.ac.jp`

Abstract

A challenge in creating a dataset for machine reading comprehension (MRC) is to collect questions that require a sophisticated understanding of language to answer beyond using superficial cues. In this work, we investigate what makes questions easier across recent 12 MRC datasets with three question styles (answer extraction, description, and multiple choice). We propose to employ simple heuristics to split each dataset into *easy* and *hard* subsets and examine the performance of two baseline models for each of the subsets. We then manually annotate questions sampled from each subset with both validity and requisite reasoning skills to investigate which skills explain the difference between easy and hard questions. From this study, we observed that (i) the baseline performances for the hard subsets remarkably degrade compared to those of entire datasets, (ii) hard questions require knowledge inference and multiple-sentence reasoning in comparison with easy questions, and (iii) multiple-choice questions tend to require a broader range of reasoning skills than answer extraction and description questions. These results suggest that one might overestimate recent advances in MRC.

Article: Spectre (2015 film) on Wikipedia
Context: (s_1) In *November 2014*, **Sony Pictures** Entertainment was targeted by **hackers** who released details of confidential **e-mails** between **Sony** executives regarding [...]. (s_2) Included within these were several memos relating to the production [...]. (s_3) Eon Productions later issued a statement [...].
Question: When $_{(k=1)}$ did hackers get into the Sony Pictures e-mail system?
Prediction for the full question: *November 2014*
Prediction for the $k = 1$ question: *November 2014*
Uni-gram overlaps between s_i and the question:
s_1: 5, s_2: 0, s_3: 0

Figure 1: Example from the SQuAD dataset (Rajpurkar et al., 2016). The baseline system can answer the token-limited question and, even if there are other candidate answers, it can easily attend to the answer-contained sentence (s_1) by watching word overlaps.

1 Introduction

Evaluating natural language understanding (NLU) systems is a long-established problem in AI (Levesque, 2014). One approach to doing so is the machine reading comprehension (MRC) task, in which a system answers questions about given texts (Hirschman et al., 1999). Although recent studies have made advances (Yu et al., 2018), it is still unclear to what precise extent questions require understanding of texts (Jia and Liang, 2017).

In this study, we examine MRC datasets and discuss what is needed to create datasets suitable for the detailed testing of NLU. Our motivation originates from studies that demonstrated unintended biases in the sourcing of other NLU tasks, in which questions contain simple patterns and systems can recognize these patterns to answer them (Gururangan et al., 2018; Mostafazadeh et al., 2017).

We conjecture that a situation similar to this occurs in MRC datasets. Consider the question shown in Figure 1, for example. Although the question, starting with *when*, requires an answer that is expressed as a moment in time, there is only one such expression (i.e., *November 2014*) in the given text (we refer to the text as the *context*). In other words, the question has only a single candidate answer. The system can solve it merely by *recognizing the entity type* required by *when*. In addition to this, even if another expression of time appears in other sentences, only one sentence (i.e., s_1) appears to be related to the question; thus, the system can easily determine the correct answer by *attention*, that is, by matching the words appearing both in the context and the ques-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4208–4219
Brussels, Belgium, October 31 - November 4, 2018.

tion. Therefore, this kind of question does not require a complex understanding of language—e.g., multiple-sentence reasoning, which is known as a more challenging task (Richardson et al., 2013).

In Section 3, we define two heuristics, namely *entity-type recognition* and *attention*. We specifically analyze the differences in the performance of baseline systems for the following two configurations: (i) questions answerable or unanswerable with the first k tokens; and (ii) questions whose correct answer appears or does not appear in the context sentence that is most similar to the question (henceforth referred to as *the most similar sentence*). Although similar heuristics are proposed by Weissenborn et al. (2017), ours are utilized for question filtering, rather than system development; Using these simple heuristics, we split each dataset into *easy* and *hard* subsets for further investigation of the baseline performance.

After conducting the experiments, we analyze the following two points in Section 4. First, we consider which questions are valid for testing, i.e., reasonably solvable. Second, we consider what reasoning skills are required and whether this exposes any differences among the subsets. To investigate these two concerns, we manually annotate sample questions from each subset in terms of validity and required reasoning skills, such as word matching, knowledge inference, and multiple sentence reasoning.

We examine 12 recently proposed MRC datasets (Table 1), which include answer extraction, description, and multiple-choice styles. We also observe differences based on these styles. For our baselines, we use two neural-based systems, namely, the Bidirectional Attention Flow (Seo et al., 2017) and the Gated-Attention Reader (Dhingra et al., 2017).

In Section 5, we describe the advantages and disadvantages of different question styles with regard to evaluating NLU systems. We also interpret our heuristics for constructing realistic MRC datasets.

Our contributions are as follows:

- This study is the first large-scale investigation across recent 12 MRC datasets with three question styles.
- We propose to employ simple heuristics to split each dataset into *easy* and *hard* subsets and examine the performance of two baseline models for each of the subsets.
- We manually annotate questions sampled from each subset with both validity and requisite reasoning skills to investigate which skills explain the difference between easy and hard questions.

Answer extraction (select a context span)
1. SQuAD (v1.1) (Rajpurkar et al., 2016)
2. AddSent (Jia and Liang, 2017)
3. NewsQA (Trischler et al., 2017)
4. TriviaQA (Wikipedia set) (Joshi et al., 2017)
5. QAngaroo (WikiHop) (Welbl et al., 2018)
Description (generate a free-form answer)
6. MS MARCO (v2) (Nguyen et al., 2016)
7. NarrativeQA (summary) (Kočiský et al., 2018)
Multiple choice (choose from multiple options)
8. MCTest (160 + 500) (Richardson et al., 2013)
9. RACE (middle + high) (Lai et al., 2017)
10. MCScript (Ostermann et al., 2018)
11. ARC Easy (ARC-E) (Clark et al., 2018)
12. ARC Challenge (ARC-C) (Clark et al., 2018)

Table 1: Examined datasets.

We observed the following:

- The baseline performances for the hard subsets remarkably degrade compared to those of entire datasets.
- Our annotation study shows that hard questions require knowledge inference and multiple-sentence reasoning in comparison with easy questions.
- Compared to questions with answer extraction and description styles, multiple-choice questions tend to require a broader range of reasoning skills while exhibiting answerability, multiple answer candidates, and unambiguity.

These findings suggest that one might overestimate recent advances in MRC systems. They also emphasize the importance of considering simple answer-seeking heuristics when sourcing questions, in that a dataset could be easily biased unless such heuristics are employed.[1]

2 Examined Datasets and Baselines

2.1 Datasets

We analyzed 12 MRC datasets with three question styles: answer extraction, description, and

---

[1] All scripts used in this study, along with the subsets of the datasets and the annotation results, are available at `https://github.com/Alab-NII/mrc-heuristics`.

multiple choice (Table 1). Our aim was to select datasets varying in terms of corpus genre, context length, and question sourcing methods.[2] Other datasets that are not covered in our study, but can be analyzed using the same method, include: QA4MRE (Sutcliffe et al., 2013), CNN/Daily Mail (Hermann et al., 2015), Children's Book Test (Hill et al., 2016), bAbI (Weston et al., 2015), WikiReading (Hewlett et al., 2016), LAMBADA (Paperno et al., 2016), Who-did-What (Onishi et al., 2016), ProPara (Dalvi et al., 2018), MultiRC (Khashabi et al., 2018), CliCR (Suster and Daelemans, 2018), SQuAD (v2.0) (Rajpurkar et al., 2018), and DuoRC (Saha et al., 2018).

2.2 Baseline Systems

We employed the following two widely used baselines.

Bidirectional Attention Flow (BiDAF) (Seo et al., 2017) was used for the answer extraction and description datasets. BiDAF models bi-directional attention between the context and question. It achieved state-of-the-art performance on the SQuAD dataset.

Gated-Attentive Reader (GA) (Dhingra et al., 2017) was used for the multiple-choice datasets. GA has a multi-hop architecture with an attention mechanism. It achieved state-of-the-art-performance on the CNN/Daily Mail and Who-did-What datasets.

Why we used different baseline systems: The multiple-choice style can be transformed to answer extraction, as mentioned in Clark et al. (2018). However, in some datasets, many questions have no textual overlap to determine the correct answer span in the context. Therefore, in order to avoid underestimating the baseline performance of those datasets, we used the GA system which is applicable to multiple choice questions.

We scored the performance using exact match (EM)/F1 (Rajpurkar et al., 2016), Rouge-L (Lin, 2004), and accuracy for the answer extraction, description, and multiple-choice datasets, respectively (henceforth, we refer to these collectively as the *score*, for simplicity). For the description datasets, we determined in advance the answer span of the context that gives the highest Rouge-L score to the human-generated gold answer. We computed the Rouge-L score between the predicted span and the gold answer.[3]

Reproduction of the baseline performance: We used the same architecture as the official baseline systems unless specified otherwise. All systems were trained on the training set and tested on the development/test set of each dataset. We also used different hyperparameters for each dataset according to characteristics such as context length (see Appendix A for details). We show the baseline performance of both the official results and those from our implementations in Tables 2 and 3. Our implementations outperformed or showed comparable performance to the official baseline on most datasets. However, in TriviaQA, MCTest, RACE, and ARC-E, our baseline performance did not reach that of the official baseline, due to differences in architecture or the absence of reported hyperparameters in the literature.

3 Two Filtering Heuristics

The first goal of this paper is to determine whether there are unintended biases of the kind exposed in Figure 1 in MRC datasets. We examined the influence of the two filtering heuristics: (i) entity type recognition (Section 3.1) and (ii) attention (Section 3.2). We then investigated the performance of the baseline systems on the questions filtered by the defined heuristics (Section 3.3).

3.1 Entity Type-based Heuristic

The aim of this heuristic was to detect questions that can be solved based on (i) the existence of a single candidate answer that is restricted by expressions such as "wh-" and "how many," and (ii) lexical patterns that appear around the correct answer. Because the query styles are not uniform across datasets (e.g., MARCO uses search engine queries), we could not directly use interrogatives. Instead, we simply provided the first k tokens of questions to the baseline systems. We chose smaller values for k than the (macro) average of the question length across the datasets (= 12.2 tokens). For example, for $k = 4$ of the question *will I qualify for OSAP if I'm new in Canada* (excerpted from MARCO), we use *will I qualify for*. Even if the tokens do not have an interrogative, the system may recognize lexical patterns around the correct answer. Questions that can be solved

---

[2] The ARC Easy and Challenge were collected using different methods; hence, we treated them as different datasets (see Clark et al. (2018) for further details).

[3] We used the official evaluation scripts of SQuAD and MS MARCO to compute the EM/F1 and Rouge-L, respectively.

	Dataset	SQuAD	AddSent	NewsQA	TriviaQA	QAngaroo	MARCO	NarraQA
Statistics	Question style (metrics)	answer extraction (exact match / F1)					description (Rouge-L)	
	Question sourcing	reading context	reading context	reading headline	trivia / quiz	chaining knowledge[1]	search query[1]	reading summary
	Context genre	Wikipedia	Wikipedia	news	Wikipedia	Wikipedia	web	moviescript
	Split examined	dev	dev	test	dev[2]	dev	dev	test
	# questions	10570	3560	5126	430	5129	55578[3]	10557
	Avg. # context tokens	150.1	163.3	698.8	783.4	1545.5	625.7	664.5
	Avg. # question tokens	11.8	12.3	8.0	19.0	3.6	6.1	9.9
	Avg. # sents in context	5.2	5.8	30.3	28.5	57.2	31.5	27.6
Baseline performance	Official baseline	67.7/77.3	28.2/34.3	34.1/48.2	47.5/53.7	42.9/-	17.7[4]	36.30
	Our BiDAF baseline	**67.9/77.2**	**42.6/50.4**	**40.2/56.4**	**44.0/49.3**	**43.8/49.3**	**36.42**[3]	**43.66**
	Q first tokens (k=4)	30.7/44.6	19.2/29.7	30.4/44.4	20.5/25.0	43.6/49.1	32.61	25.23
	(k=2)	14.0/25.0	9.4/17.8	19.4/30.3	14.4/18.5	42.6/48.0	25.13	13.00
	(k=1)	7.0/14.9	4.2/10.6	13.5/23.8	8.6/12.5	42.0/47.5	21.67	8.45
	% of # Q ($\geq$0.5 for k=2)	22.4	15.8	29.7	20.0	49.8	17.9	10.3
	Ans in sim sent	71.4/80.6	50.2/58.2	42.9/59.7	58.0/65.1	41.7/49.2	38.96	45.17
	only with sim sent	73.3/82.8	71.4/81.1	52.8/70.9	64.8/72.7	66.7/74.2	45.30	58.56
	Ans not in sim sent	56.6/66.4	28.1/35.5	37.8/53.5	40.4/45.2	43.9/49.3	35.84	41.99
	% of # Q (ans in sim)	76.3	65.7	46.3	20.5	4.2	18.6	52.6
	***Hard* subset**	**38.7/45.2**	**18.2/23.4**	**27.9/40.9**	**30.0/32.5**	**2.3/2.6**	**15.42**	**39.61**
	% of *hard*	**15.7**	**25.4**	**30.0**	**59.8**	**36.9**	**12.5**	**28.2**

Table 2: Statistics from the answer extraction and description datasets and their baselines. *Dev* represents a development set. *Ans in sim sent* refers to questions whose answer appears in the sentence that is most similar to the question. [1]The questions are not complete sentences and may start with more specific words than interrogatives. [2]Verified set. [3]No answer questions were removed. [4]The Passage Ranking model (Nguyen et al., 2016).

by examining these patterns were also of interest when filtering.

Results: Tables 2 and 3 present the results for $k = 1, 2, 4$. In addition, to know the exact ratio of the questions that are solved rather than the scores for the answer extraction and description styles, we counted questions with $k = 2$ that achieved the score ≥ 0.5.[4] As k decreased, so too did the baseline performance on all datasets in Table 2 except QAngaroo. By contrast, in QAngaroo and the multiple-choice datasets, the performance did not degrade so strongly. In particular, the difference between the scores on the full and $k = 1$ questions in QAngaroo was 1.8. Because the questions in QAngaroo are not complete sentences, but rather knowledge-base entries that have a blank, such as *country_of_citizenship Henry VI of England,* this result implies that the baseline system can infer the answer merely by the first token of questions, i.e., the type of knowledge-base entry.

In most multiple-choice datasets, the $k = 1$ scores were significantly higher than random-choice scores. Given that multiple-choice questions offer multiple options that are of valid entity/event types, this gap was not necessarily caused by the limited number of candidate answers, as in the case with the answer extraction datasets. Therefore, we inferred that in the solved questions, incorrect options appeared less than the correct option did or did not appear at all in the context (such questions were regarded as solvable exclusively using the word match skill, which we analyzed in Section 4). Remarkably, although we failed to achieve a higher baseline performance, the score for the complete questions in MCTest was lower than that of the $k = 1$ questions. This result showed that the MCTest questions were sufficiently difficult such that it was not especially useful for the baseline system to consider the entire question statement.

3.2 Attention-based Heuristic

Next, we examined in each dataset (i) how many questions have their correct answers in the most similar sentence and (ii) whether a performance gap exists for such questions (i.e., whether such questions are easier than the others).

We used uni-gram overlap as a similarity mea-

---

[4] We considered that this threshold is sufficient to judge that the system attends to the correct span because of the potential ambiguity of these styles (see Section 4).

	Dataset	MCTest	RACE	MCScript	ARC-E	ARC-C
	Style (metrics)	multiple choice (accuracy)				
	Q sourcing	reading context	English exam	script scenario	science exam	
	Genre	narrative	various	narrative	textbook	
Statistics	Split examined	test	test	dev	dev	dev
	# questions	840	4934	1411	2376	1171
	Avg. # C tokens	249.9	339.3	195.2	142.0	138.3
	Avg. # Q tokens	9.4	11.5	7.8	21.8	25.4
	Avg. # sents	18.4	17.9	11.5	8.1	8.2
Baseline performance	Random	25.0	25.0	50.0	25.0	25.0
	Official baseline	43.2[1]	44.1	72.0	62.6[2]	20.3[2]
	Our GA baseline	**34.3**	**42.7**	**75.5**	**43.9**	**30.1**
	Q tokens (k=4)	36.1	38.4	73.7	38.8	30.6
	(k=2)	33.9	37.7	71.1	37.0	29.0
	(k=1)	34.9	36.4	70.9	35.3	28.6
	Ans in sim sent	33.1	40.8	74.0	47.5	31.6
	only w/ sim	32.4	40.4	74.4	48.5	28.9
	Ans not in sim	34.9	43.3	75.8	40.4	29.4
	% of # Q (in sim)	33.5	23.2	17.7	48.7	34.8
	***Hard* subset**	**4.3**	**23.5**	**28.7**	**20.6**	**15.6**
	% of *hard*	**62.4**	**58.8**	**27.1**	**53.9**	**66.4**

Table 3: Statistics from the multiple-choice datasets and their baselines. [1]The Attentive Reader (Hermann et al., 2015) from Yin et al. (2016). [2]An information retrieval system from Clark et al. (2018).

sure.[5] We counted how many times question words appeared in each sentence, where question words were stemmed and stopwords were dropped. We then checked whether the correct answer appeared in the most similar sentence. For the multiple-choice datasets, we selected the text span that provided the highest Rouge-L score with the correct option as the correct answer.

Results: Tables 2 and 3 show the results. Considering the average number of context sentences, most datasets contained a significantly high proportion of questions whose answers were in the most similar sentence.

In the answer extraction and description datasets, except QAngaroo, the baseline performance improved when the correct answer appeared in the most similar sentence, and gaps were found between the performances on these questions and the others. These gaps indicated that the dataset may lack balance for testing NLU. If these questions tend to require the word matching skill exclusively, attending the other portion is useful in studying a more realistic NLU, e.g., common-sense reasoning and discourse understanding. Therefore, we investigated whether these questions merely require word matching (see Section 4).

Meanwhile, in the first three multiple-choice datasets, the performance differences were marginal or inversed, implying that although the baseline performance was not especially high, the difficulty of these questions for the baseline system was not affected by whether their correct answers appeared in the most similar sentence.

We further analyzed the baseline performance after removing the context and leaving only the most similar sentence. In AddSent and QAngaroo, the scores remarkably improved (>20 F1). From this result, we can infer that on these datasets the baseline systems were distracted by other sentences in the context. This observation was supported by the results from the AddSent dataset (Jia and Liang, 2017), which contains manually injected distracting sentences (i.e., adversarial examples).

3.3 Performance on *Hard* Subsets

In the previous two sections, we observed that in the examined datasets (i) some questions were solved by the baseline systems merely with the first k tokens and/or (ii) the baseline performances increased for questions whose answers were in the most similar sentence. We were concerned that these two will become dominant factors in measuring the baseline performance using the datasets; Hence, we split each development/test set into *easy* and *hard* subsets for further investigation.

***Hard* subsets:** A *hard* subset comprised questions (i) whose score is not positive when $k = 2$ *and* (ii) whose correct answer does not appear in the most similar sentence. The easy subsets comprised the remaining questions. We aimed to investigate the gap of the performance values between the *easy* and *hard* subsets. If the gap is large, the dataset may be strongly biased toward questions that are solved by recognizing entity types or lexical patterns and may not be suitable for measuring the system's ability for complex reasoning.

Results and clarification: The bottom row of Tables 2 and 3 shows that the baseline performances on the *hard* subset remarkably decreased in almost all examined datasets. These results revealed that we may overestimate the ability of the baseline systems previously perceived. How-

---

[5] Although there are other similarity measures, we used this basic measure to obtain an intuitive result.

ever, we clarify that our intention is not to remove the questions solved or mitigated by our defined heuristics to create a new *hard* subset because this may generate new biases as indicated in Gururangan et al. (2018). Rather, we would like to emphasize the importance of the defined heuristics when sourcing questions. Indeed, ill attention to these heuristics can lead to unintended biases.

4 Annotating Question Validity and Required Skills

4.1 Annotation Specifications

Objectives: To complement the observations in the previous sections, we annotated sampled questions from each subset of the datasets. Our motivation can be summarized as follows: (i) How many questions are valid in each dataset? That is, the *hard* questions may not in fact be hard, but just unsolvable, as indicated in Chen et al. (2016). (ii) What kinds of reasoning skills explain the *easy*/*hard* questions? (iii) Are there any differences among the datasets and the question styles?

We annotated the minimum skills required to choose the correct answer among other candidates. We assumed that the solver knows what type of entity or event is entailed by the question.

Annotation labels: Our annotation labels (Table 4) were inspired by previous works such as Chen et al. (2016), Trischler et al. (2017), and Lai et al. (2017). The major modifications were twofold: (i) detailed question validity, including a number of reasonable candidate answers and answer ambiguity, and (ii) posing multiple-sentence reasoning as a skill compatible with other skills.

Reasoning types indeed have other classifications. For instance, Lai et al. (2017) defined five reasoning types, including *attitude analysis* and *whole-picture reasoning*. We incorporated them into the *knowledge* and *meta/whole* classes. Clark et al. (2018) proposed detailed knowledge and reasoning types, but these were specific to science exams and, thus, omitted from our study.

Independent of the abovementioned reasoning types, we checked whether the question required multiple-sentence reasoning to answer the questions. As another modification, we extended the notion of "sentence" in our annotation and considered a subordinate clause as a sentence. This modification was intended to deal with the internal complexity of a sentence with multiple clauses, which can also render a question difficult.

Validity

1. *Unsolvable* – the context coupled with the question does not reasonably give the answer.
2. *Single candidate* – the question does not have multiple candidate answers.
3. *Ambiguous* – the question does not have a unique, decidable answer, or, multiple possible answers are not covered by the gold answers.

Reasoning skill

4. *Word matching* – matching the context and question words.
5. *Paraphrasing* – using lexical and grammatical knowledge.
6. *Knowledge* – inference using commonsense and/or world knowledge.
7. *Meta/Whole* – understanding meta terms, such as "author" and "writer," and comprehending the general context.
8. *Math/Logic* – using mathematical and logical knowledge, includeing multiple-choice questions that ask "which option is not true."

Multiple-sentence reasoning

9. (i) coreference (ii) causal relation (iii) spatial–temporal relations (iv) none – gathering cues from multiple sentences/clauses.

Table 4: Annotation labels. One of the reasoning skills is annotated with the questions that are "no" in all validity labels. Multiple sentence reasoning is independent of reasoning skills and annotated with all valid questions.

Settings: For each subset of the datasets, 30 questions were annotated. Therefore we obtained annotations for $30 \times 2 \times 12 = 720$ questions. The annotation was performed by the authors. The annotator was given the context, question, and candidate answers for multiple-choice questions along with the correct answer. To reduce bias, the annotator did not know which *easy* or *hard* subset the questions were in, and was not told the predictions and scores of the respective baseline systems.

4.2 Annotation Results

Tables 5 and 6 show the annotation results.

Validity: TriviaQA, QAngaroo, and ARCs revealed a relatively high *unsolvability*, which seemed to be caused by the unrelatedness between the questions and their context. For example, QAngaroo's context was gathered from Wikipedia articles that were not necessarily related to the questions.[6] The context passages in ARCs were

---

[6] Nonetheless, it is remarkable that even though the dataset was automatically constructed, the remaining valid *hard*

	Dataset	SQuAD		AddSent		NewsQA		TriviaQA		QAngaroo		MARCO		NarraQA	
	Subset	easy	hard	easy	hard	easy	hard	easy	hard	easy	hard	easy	hard	easy	hard
	F1/Rouge-L	80.9	37.6	61.5	29.5	52.7	30.3	70.6	33.4	71.1	3.5	49.4	21.5	54.9	51.2
Validity	Unsolvable	0.0	0.0	0.0	0.0	0.0	6.7	16.7	16.7	33.3	43.3	0.0	0.0	0.0	0.0
	Single cand.	23.3	10.0	6.7	3.3	10.0	3.3	3.3	6.7	6.7	3.3	0.0	0.0	6.7	0.0
	Ambiguous	3.3	13.3	3.3	13.3	43.3	30.0	13.3	13.3	13.3	20.0	6.7	3.3	0.0	0.0
	Valid	73.3	76.7	90.0	83.3	46.7	60.0	66.7	63.3	46.7	33.3	93.3	96.7	93.3	100.0
Skill	Word match	59.1	21.7	55.6	24.0	42.9	66.7	45.0	26.3	35.7	20.0	89.3	44.8	46.4	43.3
	Paraphrasing	18.2	26.1	11.1	36.0	21.4	11.1	5.0	10.5	7.1	20.0	0.0	10.3	25.0	20.0
	Knowledge	22.7	47.8	33.3	40.0	35.7	22.2	50.0	63.2	57.1	60.0	10.7	44.8	28.6	33.3
	Meta/Whole	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.3
	Math/Logic	0.0	4.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Relation	Multi sent.	22.7	17.4	25.9	36.0	35.7	16.7	35.0	36.8	57.1	80.0	7.1	13.8	28.6	46.7
	Coreference	18.2	17.4	14.8	32.0	21.4	16.7	35.0	31.6	50.0	50.0	7.1	13.8	14.3	33.3
	Causal	0.0	0.0	3.7	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	14.3	6.7
	Space/Temp.	4.5	0.0	7.4	0.0	14.3	0.0	0.0	5.3	7.1	30.0	0.0	0.0	0.0	6.7

Table 5: Annotation results for the answer extraction and description datasets.

	Dataset	MCTest		RACE		MCScript		ARC-E		ARC-C	
	Subset	easy	hard	easy	hard	easy	hard	easy	hard	easy	hard
	Accuracy	83.3	13.3	76.7	30.0	93.3	26.7	60.0	16.7	43.3	10.0
Validity	Unsolvable	0.0	0.0	0.0	0.0	0.0	0.0	3.3	30.0	46.7	33.3
	Single cand.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Ambiguous	0.0	0.0	3.3	0.0	0.0	0.0	3.3	0.0	3.3	3.3
	Valid	100.0	100.0	96.7	100.0	100.0	100.0	93.3	70.0	50.0	63.3
Skill	Word match	56.7	46.7	17.2	6.7	36.7	46.7	71.4	52.4	33.3	15.8
	Paraphrasing	6.7	10.0	13.8	6.7	20.0	6.7	14.3	19.0	20.0	31.6
	Knowledge	30.0	26.7	34.5	43.3	20.0	36.7	14.3	23.8	40.0	42.1
	Meta/Whole	3.3	3.3	31.0	33.3	20.0	10.0	0.0	0.0	0.0	0.0
	Math/Logic	3.3	13.3	3.4	10.0	3.3	0.0	0.0	4.8	6.7	10.5
Relation	Multi sent.	46.7	73.3	58.6	76.7	0.0	30.0	7.1	14.3	0.0	10.5
	Coreference	33.3	56.7	44.8	60.0	0.0	16.7	7.1	9.5	0.0	0.0
	Causal	6.7	6.7	3.4	13.3	0.0	3.3	0.0	0.0	0.0	0.0
	Space/Temp.	6.7	10.0	10.3	3.3	0.0	10.0	0.0	4.8	0.0	10.5

Table 6: Annotation results for the multiple-choice datasets.

Label	r	p
Single cand (BiDAF)	0.150	0.002
Ambiguous (BiDAF)	0.098	0.044
Word matching (BiDAF)	0.266	0.000
Knowledge (BiDAF)	-0.288	0.000
Multi sent (BiDAF)	-0.120	0.035
Unsolvable (GA)	-0.119	0.039

Table 7: Pearson's correlation coefficients (r) between the annotation labels and the baseline scores with $p < 0.05$.

curated from textbooks that may not provide sufficient information to answer the questions.[7] Note that it is possible for unsolvable questions to be permitted, and that the system must indicate them in some datasets, such as QA4MRE, NewsQA, MARCO, and SQuAD (v2.0).

However, for *single candidate*, we found that few questions had only single-candidate answers. Furthermore, there were even fewer single-candidate answers in AddSent than in SQuAD. This result supported the claim that the adversarial examples augmented the number of possible candidate answers, thereby degrading the baseline performance.

In our annotation, *ambiguous* questions were

---

questions were difficult for the baseline system.

[7] Our analysis was not intended to undermine the quality of these questions. We refer readers to Clark et al. (2018).

ID: ./cnn/stories/4ca29639845a40551a62d10212a46aec7caf3369.story-2
Context: [...] This plot of land is scheduled to house the permanent United Airlines Flight 93 memorial. [...]
Question: What was the name of the flight?
Answer: 93
Possible answers: United Airlines Flight 93, Flight 93

Figure 2: Example of an *ambiguous* question from NewsQA (Trischler et al., 2017).

found to be those with multiple correct spans. Figure 2 shows an example. In this case, several answers aside from "93" were correct. Ambiguity is an important feature insofar because it can lead to unstable scoring in EM/F1.

The multiple-choice datasets mostly comprised valid questions, with the exception of the unsolvable questions in the ARC datasets.

Reasoning skills: We can see that *word matching* was more important in the *easy* subsets, and *knowledge* was more pertinent to the *hard* subsets in 10 of the 12 datasets. These results confirmed that the manner by which we split the subsets was successful at filtering questions that were relatively easy in terms of reasoning skills. However, we did not observe this trend with *paraphrasing*, which seemed difficult to distinguish from *word matching* and *knowledge*. With regard to *meta/whole* and *math/logic*, we can see that these skills were needed less in the answer extraction and description datasets. They were more pertinent to the multiple-choice datasets.

Multiple-sentence reasoning: Multiple-sentence reasoning was more correlated with the *hard* subsets in 10 of the 12 datasets. Although NewsQA showed the inverse tendency for *word matching*, *knowledge*, and *multiple-sentence reasoning*, we suspect that this was caused by annotation variance and filtering a large portion of ambiguous questions. For relational types, we did not see a significant trend in any particular type.

Correlation of labels and baseline scores: Across all examined datasets, we analyzed the correlations between the annotation labels and the scores of each baseline system in Table 7. In spite of the small size of the annotated samples, we derived statistically significant correlations for six labels. These results confirmed that BiDAF performed well for the *word matching* questions and relatively poorly with the *knowledge* questions. By contrast, we did not observe this trend in GA.

5 Discussion

In this section, we discuss the advantages and disadvantages of the question styles. We also interpret the defined heuristics in terms of constructing more realistic MRC datasets.

Differences among the question styles: The biggest advantage to the answer extraction style is its ease in generating questions, which enables us to produce large-scale datasets. In contrast, a disadvantage to this style is that it rarely demands *meta/whole* and *math/logic* skills, which can require answers not contained in the context. Moreover, as observed in Section 4, it seems difficult to guarantee that all possible answer spans are given as the correct answers. By contrast, the description and multiple-choice styles have the advantage of having no such restrictions on the appearance of candidate answers (Kočiský et al., 2018; Khashabi et al., 2018). Nonetheless, the description style is difficult to evaluate because the Rouge-L and BLEU scores are insufficient for testing NLU. Whereas it is easy to evaluate the performance on multiple-choice questions, generating multiple reasonable options requires considerable effort.

Interpretation of our heuristics: When we regard the MRC task as recognizing textual entailment (RTE) (Dagan et al., 2006), the task requires the reader to construct one or more premises from the context and form the most reasonable hypothesis from the question and candidate answer (Sachan et al., 2015). Thus, easier questions are those (i) where the reader needs to generate only one hypothesis, and (ii) where the premises directly describe the correct hypothesis. Our two heuristics can also be seen as the formalizations of these criteria. Therefore, to make questions more realistic, we need to create multiple hypotheses that require complex reasoning to be distinguished. Moreover, the integration of premises should be complemented by external knowledge to provide sufficient information to verify the correct hypothesis.

6 Related Work

Our heuristics and annotation were motivated by *unintended biases* (Levesque, 2014) and *evaluation overfitting* (Whiteson et al., 2011), respectively.

Unintended biases: The MRC task tests a reading process that involves retrieving stored information and performing inferences (Sutcliffe et al., 2013). However, constructing datasets that comprehensively require those skills is difficult. As Levesque (2014) discussed as a desideratum for testing AI, we should avoid creating questions that can be solved by matching patterns, using unintended biases, and selectional restrictions. For the unintended biases, one suggestive example is the Story Cloze Test (Mostafazadeh et al., 2016), in which a system chooses a sentence among candidates to conclude a given paragraph of the story. A recent attempt at this task showed that recognizing superficial features in the correct candidate is critical to achieve the state of the art (Schwartz et al., 2017).

Similarly, in MRC, Weissenborn et al. (2017) proposed *context/type matching heuristic* to develop a simple neural system. Min et al. (2018) observed that, in SQuAD, 92% of answerable questions can be answered only using a single context sentence. In visual question answering, Agrawal et al. (2016) analyzed the behavior of models with the variable length of the first question words. Khashabi et al. (2018) more recently proposed a dataset with questions for multi-sentence reasoning.

Evaluation overfitting: The theory behind evaluating AI distinguishes between task- and skill-oriented approaches (Hernández-Orallo, 2017). In the task-oriented approach, we usually develop a system and test it on a specific dataset. The developed system sometimes lacks generality but achieves the state of the art for that specific dataset. Further, it becomes difficult to verify and explain the solution to tasks. The situation in which we are biased to the specific tasks is called evaluation overfitting (Whiteson et al., 2011). By contrast, with the skill-oriented approach, we aim to interpret the relationships between tasks and skills. This orientation can encourage the development of more realistic NLU systems.

As One of our goals was to investigate whether easy questions are dominant in recent datasets, it did not necessarily require a detailed classification of reasoning types. Nonetheless, we recognize there are more fine-grained classifications of the required skills for NLU. For example, Weston et al. (2015) defined 20 skills as a set of toy tasks. Sugawara et al. (2017) also organized 10 prerequisite skills for MRC. LoBue and Yates (2011) and Sammons et al. (2010) analyzed entailment phenomena using detailed classifications in RTE. For the ARC dataset, Boratko et al. (2018) proposed knowledge and reasoning types.

7 Conclusion

This study examined MRC questions from 12 datasets to determine what makes such questions easier to answer. We defined two heuristics that limit candidate answers and thereby mitigate the difficulty of questions. Using these heuristics, the datasets were split into easy and hard subsets. We further annotated the questions with their validity and the reasoning skills needed to answer them. Our experiments revealed that the baseline performance degraded with the hard questions, which required knowledge inference and multiple-sentence reasoning compared to easy questions. These results suggest that one might overestimate the ability of the baseline systems. They also emphasize the importance of analyzing and reporting the properties of new datasets when released. One limitation of this work was the heavy cost of the annotation. In future research, we plan to explore a method for automatically classifying reasoning types. This will enable us to evaluate systems through a detailed organization of the datasets.

Acknowledgments

We would like to thank Rajarshi Das, Shehzaad Dhuliawala, and anonymous reviewers for their insightful comments. This work was supported by JSPS KAKENHI Grant Numbers 18H03297 and 18J12960.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the behavior of visual question answering models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1955–1960, Austin, Texas. Association for Computational Linguistics.

Michael Boratko, Harshit Padigela, Divyendra Mikkilineni, Pritish Yuvraj, Rajarshi Das, Andrew McCallum, Maria Chang, Achille Fokoue-Nkoutche, Pavan Kapanipathi, Nicholas Mattei, Ryan Musa, Kartik Talamadupula, and Michael Witbrock. 2018. A systematic classification of knowledge, reasoning, and context within the arc dataset. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 60–70, Melbourne, Australia. Association for Computational Linguistics.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the

CNN/Daily Mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2358–2367. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1595–1604. Association for Computational Linguistics.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1693–1701.

José Hernández-Orallo. 2017. Evaluation in artificial intelligence: from task-oriented to ability-oriented measurement. *Artificial Intelligence Review*, 48(3):397–447.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. WikiReading: A novel large-scale language understanding task over wikipedia. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1545, Berlin, Germany. Association for Computational Linguistics.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *the International Conference on Learning Representations*.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 325–332. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2011–2021, Copenhagen, Denmark. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611. Association for Computational Linguistics.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262. Association for Computational Linguistics.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 796–805, Copenhagen, Denmark. Association for Computational Linguistics.

Hector J. Levesque. 2014. On our best behaviour. *Artificial Intelligence*, 212:27 – 35.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Peter LoBue and Alexander Yates. 2011. Types of common-sense knowledge needed for recognizing textual entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 329–334, Portland, Oregon, USA. Association for Computational Linguistics.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1725–1735. Association for Computational Linguistics.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did What: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235. Association for Computational Linguistics.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018. MCScript: A novel dataset for assessing machine comprehension using script knowledge. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA).

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1525–1534. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Mrinmaya Sachan, Kumar Dubey, Eric Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 239–249. Association for Computational Linguistics.

Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. 2018. DuoRC: Towards complex language understanding with paraphrased reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1693, Melbourne, Australia. Association for Computational Linguistics.

Mark Sammons, V.G.Vinod Vydiswaran, and Dan Roth. 2010. "Ask not what textual entailment can do for you...". In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1199–1208. Association for Computational Linguistics.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A. Smith. 2017. Story cloze task: UW NLP system. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 52–55, Valencia, Spain. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. 2017. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Proceedings of the 55th Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 806–817. Association for Computational Linguistics.

Simon Suster and Walter Daelemans. 2018. CliCR: a dataset of clinical case reports for machine reading comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1551–1563. Association for Computational Linguistics.

Richard Sutcliffe, Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Corina Forascu, Yassine Benajiba, and Petya Osenova. 2013. Overview of QA4MRE main task at CLEF 2013. *Working Notes, CLEF*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200. Association for Computational Linguistics.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280, Vancouver, Canada. Association for Computational Linguistics.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-complete question answering: a set of prerequisite toy tasks. In *the International Conference on Learning Representations*.

Shimon Whiteson, Brian Tanner, Matthew E Taylor, and Peter Stone. 2011. Protecting against evaluation overfitting in empirical reinforcement learning. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, pages 120–127. IEEE.

Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. 2016. Attention-based convolutional neural network for machine comprehension. In *Proceedings of the Workshop on Human-Computer Question Answering*, pages 15–21. Association for Computational Linguistics.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations*.

A Hyperparameters of the Baseline Systems

We used different hyperparameters for each dataset because of the different characteristics of the datasets, e.g., the context length. Tables 8 and 9 show the hyperparameters.

Dataset	b	h	q	d
SQuAD	60	100	400	20
AddSent	60	100	400	20
NewsQA	32	100	1000	20
TriviaQA	32	100	400	20
QAngaroo	16	50	4096	20
MARCO	20	40	1600	30
NarrativeQA	60	50	1000	20

Table 8: Hyperparameters (batch size b, hidden layer size h, document size threshold d, question size threshold q) of the Bidirectional Attention Flow (Seo et al., 2017) for each dataset. The other settings basically followed the original implementation. In TriviaQA, we followed a method for the dataset preparation used in Joshi et al. (2017).

Dataset	b	h	n	dr	lr
MCTest	10	32	1	0.5	0.01
RACE	32	128	1	0.2	0.1
MCScript	25	64	1	0.5	0.2
ARC-E	32	256	1	0.5	0.3
ARC-C	32	256	1	0.5	0.3

Table 9: Hyperparameters (batch size b, hidden layer size h, number of attention layers n, dropout rate dr, learning rate lr) of the Gated-Attentive Reader (Dhingra et al., 2017) for each dataset. The other settings basically followed the implementation in Lai et al. (2017).

Commonsense for Generative Multi-Hop Question Answering Tasks

Lisa Bauer∗ **Yicheng Wang**∗ **Mohit Bansal**
UNC Chapel Hill
`{lbauer6, yicheng, mbansal}@cs.unc.edu`

Abstract

Reading comprehension QA tasks have seen a recent surge in popularity, yet most works have focused on fact-finding extractive QA. We instead focus on a more challenging multi-hop generative task (NarrativeQA), which requires the model to reason, gather, and synthesize disjoint pieces of information within the context to generate an answer. This type of multi-step reasoning also often requires understanding implicit relations, which humans resolve via external, background commonsense knowledge. We first present a strong generative baseline that uses a multi-attention mechanism to perform multiple hops of reasoning and a pointer-generator decoder to synthesize the answer. This model performs substantially better than previous generative models, and is competitive with current state-of-the-art span prediction models. We next introduce a novel system for selecting grounded multi-hop relational commonsense information from ConceptNet via a pointwise mutual information and term-frequency based scoring function. Finally, we effectively use this extracted commonsense information to fill in gaps of reasoning between context hops, using a selectively-gated attention mechanism. This boosts the model's performance significantly (also verified via human evaluation), establishing a new state-of-the-art for the task. We also show that our background knowledge enhancements are generalizable and improve performance on QAngaroo-WikiHop, another multi-hop reasoning dataset.

1 Introduction

In this paper, we explore the task of machine reading comprehension (MRC) based QA. This task tests a model's natural language understanding capabilities by asking it to answer a question based on a passage of relevant content. Much progress has been made in reasoning-based MRC-QA on the bAbI dataset (Weston et al., 2016), which contains questions that require the combination of multiple disjoint pieces of evidence in the context. However, due to its synthetic nature, bAbI evidences have smaller lexicons and simpler passage structures when compared to human-generated text.

There also have been several attempts at the MRC-QA task on human-generated text. Large scale datasets such as CNN/DM (Hermann et al., 2015) and SQuAD (Rajpurkar et al., 2016) have made the training of end-to-end neural models possible. However, these datasets are fact-based and do not place heavy emphasis on multi-hop reasoning capabilities. More recent datasets such as QAngaroo (Welbl et al., 2018) have prompted a strong focus on multi-hop reasoning in very long texts. However, QAngaroo is an extractive dataset where answers are guaranteed to be spans within the context; hence, this is more focused on fact finding and linking, and does not require models to synthesize and generate new information.

We focus on the recently published NarrativeQA generative dataset (Kočiský et al., 2018) that contains questions requiring multi-hop reasoning for long, complex stories and other narratives, which requires the model to go beyond fact linking and to synthesize non-span answers. Hence, models that perform well on previous reasoning tasks (Dhingra et al., 2018) have had limited success on this dataset. In this paper, we first propose the Multi-Hop Pointer-Generator Model (MHPGM), a strong baseline model that uses multiple hops of bidirectional attention, self-attention, and a pointer-generator decoder to effectively read and reason within a long passage and synthesize a coherent response. Our model achieves 41.49 Rouge-L and 17.33 METEOR on the summary

---

∗ Equal contribution (published at EMNLP 2018).
We publicly release all our code, models, and data at: `https://github.com/yicheng-w/CommonSenseMultiHopQA`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4220–4230
Brussels, Belgium, October 31 - November 4, 2018.

subtask of NarrativeQA, substantially better than the performance of previous generative models.

Next, to address the issue that understanding human-generated text and performing long-distance reasoning on it often involves intermittent access to missing hops of external commonsense (background) knowledge, we present an algorithm for selecting useful, grounded multi-hop relational knowledge paths from ConceptNet (Speer and Havasi, 2012) via a pointwise mutual information (PMI) and term-frequency-based scoring function. We then present a novel method of inserting these selected commonsense paths between the hops of document-context reasoning within our model, via the Necessary and Optional Information Cell (NOIC), which employs a selectively-gated attention mechanism that utilizes commonsense information to effectively fill in gaps of inference. With these additions, we further improve performance on the NarrativeQA dataset, achieving 44.16 Rouge-L and 19.03 METEOR (also verified via human evaluation). We also provide manual analysis on the effectiveness of our commonsense selection algorithm.

Finally, to show the effectiveness and generalizability of our multi-hop reasoning and commonsense methods, we also tested our MHPGM and MHPGM+NOIC models on QAngaroo-WikiHop (Welbl et al., 2018), which is an extractive dataset for multi-hop reasoning on human-generated documents. We found that our background commonsense knowledge enhanced model achieved 1.5% higher accuracy than our strong baseline.

2 Related Work

Machine Reading Comprehension: MRC has long been a task used to assess a model's ability to understand and reason about language. Large scale datasets such as CNN/Daily Mail (Hermann et al., 2015) and SQuAD (Rajpurkar et al., 2016) have encouraged the development of many advanced, high performing attention-based neural models (Seo et al., 2017; Dhingra et al., 2017). Concurrently, datasets such as bAbI (Weston et al., 2016) have focused specifically on multi-step reasoning by requiring the model to reason with disjoint pieces of information. On this task, it has been shown that iteratively updating the query representation with information from the context can effectively emulate multi-step reasoning (Sukhbaatar et al., 2015).

More recently, there has been an increase in multi-paragraph, multi-hop inference QA datasets such as QAngaroo (Welbl et al., 2018) and NarrativeQA (Kočiskỳ et al., 2018). These datasets have much longer contexts than previous datasets, and answering a question often requires the synthesis of multiple discontiguous pieces of evidence. It has been shown that models designed for previous tasks (Seo et al., 2017; Kadlec et al., 2016) have limited success on these new datasets. In our work, we expand upon Gated Attention Network (Dhingra et al., 2017) to create a baseline model better suited for complex MRC datasets such as NarrativeQA by improving its attention and gating mechanisms, expanding its generation capabilities, and allowing access to external commonsense for connecting implicit relations.

Commonsense/Background Knowledge: Commonsense or background knowledge has been used for several tasks including opinion mining (Cambria et al., 2010), sentiment analysis (Poria et al., 2015, 2016), handwritten text recognition (Wang et al., 2013), and more recently, dialogue (Young et al., 2018; Ghazvininejad et al., 2018). These approaches add commonsense knowledge as relation triples or features from external databases. Recently, large-scale graphical commonsense databases such as ConceptNet (Speer and Havasi, 2012) use graphical structure to express intricate relations between concepts, but effective goal-oriented graph traversal has not been extensively used in previous commonsense incorporation efforts. Knowledge-base QA is a task in which systems are asked to find answers to questions by traversing knowledge graphs (Bollacker et al., 2008). Knowledge path extraction has been shown to be effective at the task (Bordes et al., 2014; Bao et al., 2016). We apply these techniques to MRC-QA by using them to extract useful commonsense knowledge paths that fully utilize the graphical nature of databases such as ConceptNet (Speer and Havasi, 2012).

Incorporation of External Knowledge: There have been several attempts at using external knowledge to boost model performance on a variety of tasks: Chen et al. (2018) showed that adding lexical information from semantic databases such as WordNet improves performance on NLI; Xu et al. (2017) used a gated recall-LSTM mechanism to incorporate commonsense information into token representations in dialogue.

In MRC, Weissenborn et al. (2017) integrated external background knowledge into an NLU model by using contextually-refined word embeddings which integrated information from ConceptNet (single-hop relations mapped to unstructured text) via a single layer bidirectional LSTM. Concurrently to our work, Mihaylov and Frank (2018) showed improvements on a cloze-style task by incorporating commonsense knowledge via a context-to-commonsense attention, where commonsense relations were extracted as triples. This work represented commonsense relations as key-value pairs and combined context representation and commonsense via a static gate.

Differing from previous works, we employ multi-hop commonsense paths (multiple connected edges within ConceptNet graph that give us information beyond a single relationship triple) to help with our MRC model. Moreover, we use this in tandem with our multi-hop reasoning architecture to incorporate different aspects of the commonsense relationship path at each hop, in order to bridge different inference gaps in the multi-hop QA task. Additionally, our model performs synthesis with its external, background knowledge as it generates, rather than extracts, its answer.

3 Methods

3.1 Multi-Hop Pointer-Generator Baseline

We first rigorously state the problem of generative QA as follows: given two sequences of input tokens: the context, $X^C = \{w_1^C, w_2^C, \ldots, w_n^C\}$ and the query, $X^Q = \{w_1^Q, w_2^Q, \ldots, w_m^Q\}$, the system should generate a series of answer tokens $X^a = \{w_1^a, w_2^a, \ldots, w_p^a\}$. As outlined in previous sections, an effective generative QA model needs to be able to perform several hops of reasoning over long and complex passages. It would also need to be able to generate coherent statements to answer complex questions while having the ability to copy rare words such as specific entities from the reading context. With these in mind, we propose the Multi-Hop Pointer-Generator Model (MHPGM) baseline, a novel combination of previous works with the following major components:

- **Embedding Layer**: The tokens are embedded into both learned word embeddings and pretrained context-aware embeddings (ELMo (Peters et al., 2018)).
- **Reasoning Layer**: The embedded context is then passed through k reasoning cells, each of which iteratively updates the context representation with information from the query via BiDAF attention (Seo et al., 2017), emulating a single reasoning step within the multi-step reasoning process.
- **Self-Attention Layer**: The context representation is passed through a layer of self-attention (Cheng et al., 2016) to resolve long-term dependencies and co-reference within the context.
- **Pointer-Generator Decoding Layer**: A attention-pointer-generator decoder (See et al., 2017) that attends on and potentially copies from the context is used to create the answer.

The overall model is illustrated in Fig. 1, and the layers are described in further detail below.

Embedding layer: We embed each word from the context and question with a learned embedding space of dimension d. We also obtain context-aware embeddings for each word via the pretrained embedding from language models (ELMo) (1024 dimensions). The embedded representation for each word in the context or question, $\mathbf{e}_i^C$ or $\mathbf{e}_i^Q \in \mathbb{R}^{d+1024}$, is the concatenation of its learned word embedding and ELMo embedding.

Reasoning layer: Our reasoning layer is composed of k reasoning cells (see Fig. 1), where each incrementally updates the context representation. The t^{th} reasoning cell's inputs are the previous step's output ($\{\mathbf{c}_i^{t-1}\}_{i=1}^n$) and the embedded question ($\{\mathbf{e}_i^Q\}_{i=1}^m$). It first creates step-specific context and query encodings via cell-specific bidirectional LSTMs:

$$\mathbf{u}^t = \text{BiLSTM}(\mathbf{c}^{t-1}); \qquad \mathbf{v}^t = \text{BiLSTM}(\mathbf{e}^Q)$$

Then, we use bidirectional attention (Seo et al., 2017) to emulate a hop of reasoning by focusing on relevant aspects of the context. Specifically, we first compute context-to-query attention:

$$S_{ij}^t = W_1^t \mathbf{u}_i^t + W_2^t \mathbf{v}_j^t + W_3^t(\mathbf{u}_i^t \odot \mathbf{v}_j^t)$$

$$p_{ij}^t = \frac{\exp(S_{ij}^t)}{\sum_{k=1}^m \exp(S_{ik}^t)}$$

$$(\mathbf{c_q})_i^t = \sum_{j=1}^m p_{ij}^t \mathbf{v}_j^t$$

where W_1^t, W_2^t, W_3^t are trainable parameters, and $\odot$ is elementwise multiplication. We then compute a query-to-context attention vector:

$$m_i^t = \max_{1 \leq j \leq m} S_{ij}^t$$

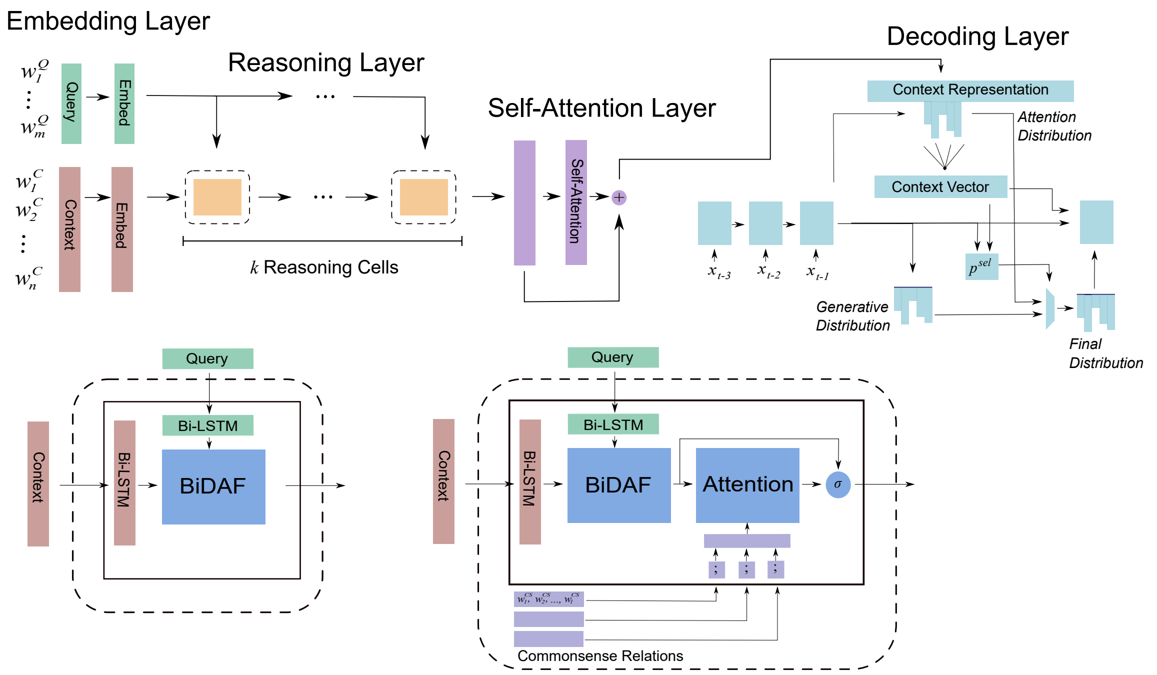

Baseline Reasoning Cell

NOIC Reasoning Cell

Figure 1: Architecture for our Multi-Hop Pointer-Generator Model, and the NOIC commonsense reasoning cell.

$$p_i^t = \frac{\exp(m_i^t)}{\sum_{j=1}^{n} \exp(m_j^t)}$$

$$\mathbf{q_c}^t = \sum_{i=1}^{n} p_i^t \mathbf{u}_i^t$$

We then obtain the updated context representation:

$$\mathbf{c}_i^t = [\mathbf{u}_i^t; (\mathbf{c_q})_i^t; \mathbf{u}_i^t \odot (\mathbf{c_q})_i^t; \mathbf{q_c}^t \odot (\mathbf{c_q})_i^t]$$

where ; is concatenation, $\mathbf{c}^t$ is the cell's output.

The initial input of the reasoning layer is the embedded context representation, i.e., $\mathbf{c}^0 = \mathbf{e}^C$, and the final output of the reasoning layer is the output of the last cell, $\mathbf{c}^k$.

Self-Attention Layer: As the final layer before answer generation, we utilize a residual static self-attention mechanism (Clark and Gardner, 2018) to help the model process long contexts with long-term dependencies. The input of this layer is the output of the last reasoning cell, $\mathbf{c}^k$. We first pass this representation through a fully-connected layer and then a bi-directional LSTM to obtain another representation of the context $\mathbf{c}^{SA}$. We obtain the self attention representation $\mathbf{c}'$:

$$S_{ij}^{SA} = W_4\mathbf{c}_i^{SA} + W_5\mathbf{c}_j^{SA} + W_6(\mathbf{c}_i^{SA} \odot \mathbf{c}_j^{SA})$$

$$p_{ij}^{SA} = \frac{\exp(S_{ij}^{SA})}{\sum_{k=1}^{n} \exp(S_{ik}^{SA})}$$

$$\mathbf{c}'_i = \sum_{j=1}^{n} p_{ij}^{SA}\mathbf{c}_j^{SA}$$

where W_4, W_5, and W_6 are trainable parameters.

The output of the self-attention layer is generated by another layer of bidirectional LSTM.

$$\mathbf{c}'' = \text{BiLSTM}([\mathbf{c}'; \mathbf{c}^{SA}; \mathbf{c}' \odot \mathbf{c}^{SA}])$$

Finally, we add this residually to $\mathbf{c}^k$ to obtain the encoded context $\mathbf{c} = \mathbf{c}^k + \mathbf{c}''$.

Pointer-Generator Decoding Layer: Similar to the work of See et al. (2017), we use a pointer-generator model attending on (and potentially copying from) the context.

At decoding step t, the decoder receives the input $\mathbf{x}_t$ (embedded representation of last timestep's output), the last time step's hidden state $\mathbf{s}_{t-1}$ and context vector $\mathbf{a}_{t-1}$. The decoder computes the current hidden state $\mathbf{s}_t$ as:

$$\mathbf{s}_t = \text{LSTM}([\mathbf{x}_t; \mathbf{a}_{t-1}], \mathbf{s}_{t-1})$$

This hidden state is then used to compute a probability distribution over the generative vocabulary:

$$P_{gen} = \text{softmax}(W_{gen}\mathbf{s}_t + \mathbf{b}_{gen})$$

We employ Bahdanau attention mechanism (Bahdanau et al., 2015) to attend over the context ($\mathbf{c}$ being the output of self-attention layer):

$$\alpha_i = \mathbf{v}^\intercal \tanh(W_c\mathbf{c}_i + W_s\mathbf{s}_t + b_{attn})$$

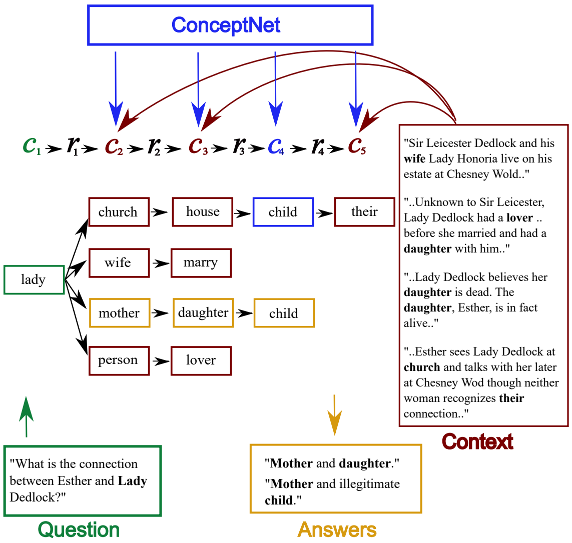

Figure 2: Commonsense selection approach.

$$\hat{\alpha}_i = \frac{\exp(\alpha_i)}{\sum_{j=1}^{n} \exp(\alpha_j)}$$

$$\mathbf{a}_t = \sum_{i=1}^{n} \hat{\alpha}_i \mathbf{c}_i$$

We utilize a pointer mechanism that allows the decoder to directly copy tokens from the context based on $\hat{\alpha}_i$. We calculate a selection distribution $\mathbf{p}^{sel} \in \mathbb{R}^2$, where p_1^{sel} is the probability of generating a token from P_{gen} and p_2^{sel} is the probability of copying a word from the context:

$$\mathbf{o} = \sigma(W_a \mathbf{a}_t + W_x \mathbf{x}_t + W_s \mathbf{s}_t + b_{ptr})$$

$$\mathbf{p}^{sel} = \text{softmax}(\mathbf{o})$$

Our final output distribution at timestep t is a weighted sum of the generative distribution and the copy distribution:

$$P_t(w) = p_1^{sel} P_{gen}(w) + p_2^{sel} \sum_{i:w_i^C = w} \hat{\alpha}_i$$

3.2 Commonsense Selection and Representation

In QA tasks that require multiple hops of reasoning, the model often needs knowledge of relations not directly stated in the context to reach the correct conclusion. In the datasets we consider, manual analysis shows that external knowledge is frequently needed for inference (see Table 1).

Dataset	Outside Knowledge Required
WikiHop	11%
NarrativeQA	42%

Table 1: Qualitative analysis of commonsense requirements. WikiHop results are from Welbl et al. (2018); NarrativeQA results are from our manual analysis (on the validation set).

Even with a large amount of training data, it is very unlikely that a model is able to learn every nuanced relation between concepts and apply the correct ones (as in Fig. 2) when reasoning about a question. We remedy this issue by introducing *grounded* commonsense (background) information using relations between concepts from ConceptNet (Speer and Havasi, 2012)[1] that help inference by introducing useful connections between concepts in the context and question.

Due to the size of the semantic network and the large amount of unnecessary information, we need an effective way of selecting relations which provides novel information while being grounded by the context-query pair. Our commonsense selection strategy is twofold: (1) collect potentially relevant concepts via a tree construction method aimed at selecting with high recall candidate reasoning paths, and (2) rank and filter these paths to ensure both the quality and variety of added information via a 3-step scoring strategy (initial node scoring, cumulative node scoring, and path selection). We will refer to Fig. 2 as a running example throughout this section.[2]

3.2.1 Tree Construction

Given context C and question Q, we want to construct paths grounded in the pair that emulate reasoning steps required to answer the question. In this section, we build 'prototype' paths by constructing trees rooted in concepts in the query with the following branching steps[3] to emulate multi-hop reasoning process. For each concept c_1 in the question, we do:

Direct Interaction: In the first level, we select relations r_1 from ConceptNet that directly link c_1 to a concept within the context, $c_2 \in C$, e.g., in Fig. 2, we have *lady* → *church*, *lady* → *mother*, *lady* → *person*.

Multi-Hop: We then select relations in ConceptNet r_2 that link c_2 to another concept in the context, $c_3 \in C$. This emulates a potential reason-

---

[1] A semantic network where the nodes are individual concepts (words or phrases) and the edges describe directed relations between them (e.g., ⟨island, UsedFor, vacation⟩).

[2] We release all our commonsense extraction code and the extracted commonsense data at: `https://github.com/yicheng-w/CommonSenseMultiHopQA`

[3] If we are unable to find a relation that satisfies the condition, we keep the steps up to and including the node.

ing hop within the context of the MRC task, e.g., *church* $\rightarrow$ *house*, *mother* $\rightarrow$ *daughter*, *person* $\rightarrow$ *lover*.

Outside Knowledge: We then allow an unconstrained hop into c_3's neighbors in ConceptNet, getting to $c_4 \in \text{nbh}(c_3)$ via r_3 ($\text{nbh}(v)$ is the set of nodes that can be reached from v in one hop). This emulates the gathering of useful external information to complete paths within the context, e.g., *house* $\rightarrow$ *child*, *daughter* $\rightarrow$ *child*.

Context-Grounding: To ensure that the external knowledge is indeed helpful to the task, and also to explicitly link 2nd degree neighbor concepts within the context, we finish the process by grounding it again into context by connecting c_4 to $c_5 \in C$ via r_4, e.g., *child* $\rightarrow$ *their*.

3.2.2 Rank and Filter

This tree building process collects a large number of potentially relevant and useful paths. However, this step also introduces a large amount of noise. For example, given the question and full context (not depicted in the figure) in Fig. 2, we obtain the path "*between* $\rightarrow$ *hard* $\rightarrow$ *being* $\rightarrow$ *cottage* $\rightarrow$ *country*" using our tree building method, which is not relevant to our question. Therefore, to improve the precision of useful concepts, we rank these knowledge paths by their relevance and filter out noise using the following 3-step scoring method:

Initial Node Scoring: We want to select paths with nodes that are important to the context, in order to provide the most useful commonsense relations. We approximate importance and saliency for concepts in the context by their term-frequency, under the heuristic that important concepts occur more frequently. Thus we score $c \in \{c_2, c_3, c_5\}$ by: $\text{score}(c) = \text{count}(c)/|C|$, where $|C|$ is the context length and count() is the number of times a concept appears in the context. In Fig. 2, this ensures that concepts like *daughter* are scored highly due to their frequency in the context.

For c_4, we use a special scoring function as it is an unconstrained hop into ConceptNet. We want c_4 to be a logically consistent next step in reasoning following the path of c_1 to c_3, e.g., in Fig. 2, we see that *child* is a logically consistent next step after the partial path of *mother* $\rightarrow$ *daughter*. We approximate this based on the heuristic that logically consistent paths occur more frequently. Therefore, we score this node via Pointwise Mutual Information (PMI) between the partial path c_{1-3} and c_4: $\text{PMI}(c_4, c_{1-3}) = \log(\mathbb{P}(c_4, c_{1-3})/\mathbb{P}(c_4)\mathbb{P}(c_{1-3}))$, where

$$\mathbb{P}(c_4, c_{1-3}) = \frac{\text{\# of paths connecting } c_1, c_2, c_3, c_4}{\text{\# of distinct paths of length 4}}$$

$$\mathbb{P}(c_4) = \frac{\text{\# of nodes that can reach } c_4}{|\text{ConceptNet}|}$$

$$\mathbb{P}(c_{1-3}) = \frac{\text{\# of paths connecting } c_1, c_2, c_3}{\text{\# of paths of length 3}}$$

Further, it is well known that PMI has high sensitivity to low-frequency values, thus we use normalized PMI (NPMI) (Bouma, 2009): $\text{score}(c_4) = \text{PMI}(c_4, c_{1-3})/(-\log \mathbb{P}(c_4, c_{1-3}))$.

Since the branching at each juncture represents a hop in the multi-hop reasoning process, and hops at different levels or with different parent nodes do not 'compete' with each other, we normalize each node's score against its siblings:

$$\text{n-score}(c) = \text{softmax}_{\text{siblings}(c)}(\text{score}(c)).$$

Cumulative Node Scoring: We want to add commonsense paths consisting of multiple hops of relevant information, thus we re-score each node based not only on its relevance and saliency but also that of its tree descendants.

We do this by computing a cumulative node score from the bottom up, where at the leaf nodes, we have c-score $=$ n-score, and for c_l not a leaf node, we have $\text{c-score}(c_l) = \text{n-score}(c_l) + f(c_l)$ where f of a node is the average of the c-scores of its top 2 highest scoring children.

For example, given the paths *lady* $\rightarrow$ *mother* $\rightarrow$ *daughter*, *lady* $\rightarrow$ *mother* $\rightarrow$ *married*, and *lady* $\rightarrow$ *mother* $\rightarrow$ *book*, we start the cumulative scoring at the leaf nodes, which in this case are *daughter*, *married*, and *book*, where *daughter* and *married* are scored much higher than *book* due to their more frequent occurrences. Then, to cumulatively score $mother$, we would take the average score of its two highest scoring children (in this case *married* and *daughter*) and compound that with the score of *mother* itself. Note that the poor scoring of the irrelevant concept *book* does not affect the scoring of *mother*, which is quite high due to the concept's frequent occurrence and the relevance of its top scoring children.

Path Selection: We select paths in a top-down breath-first fashion in order to add information relevant to different parts of the context. Starting at the root, we recursively take two of its children with the highest cumulative scores until we reach a leaf, selecting up to $2^4 = 16$ paths. For example, if we were at node *mother*, this allows us to select the child node *daughter* and *married* over the child node *book*. These selected paths, as well as their partial sub-paths, are what we add as external information to the QA model, i.e., we add the complete path ⟨lady, AtLocation, church, RelatedTo, house, RelatedTo, child, RelatedTo, their⟩, but also truncated versions of the path, including ⟨lady, AtLocation, church, RelatedTo, house, RelatedTo, child⟩. We directly give these paths to the model as sequences of tokens.[4]

Overall, our sampling strategy provides the knowledge that a *lady* can be a *mother* and that *mother* is connected to *daughter*. This creates a logical connection between *lady* and *daughter* which helps highlight the importance of our second piece of evidence (see Fig. 2). Likewise, the commonsense information we extracted create a similar connection in our third piece of evidence, which states the explicit connection between *daughter* and *Esther*. We also successfully extract a more story context-centric connection, in which commonsense provides the knowledge that a *lady* is at the location *church*, which directs to another piece of evidence in the context. Additionally, this path also encodes a relation between *lady* and *child*, by way of *church*, which is how *lady* and *Esther* are explicitly connected in the story.

3.3 Commonsense Model Incorporation

Given the list of commonsense logic paths as sequences of words: $X^{CS} = \{w_1^{CS}, w_2^{CS}, \dots, w_l^{CS}\}$ where w_i^{CS} represents the list of tokens that make up a single path, we first embed these commonsense tokens into the learned embedding space used by the model, giving us the embedded commonsense tokens, $\mathbf{e}_{ij}^{CS} \in \mathbb{R}^d$. We want to use these commonsense paths to fill in the gaps of reasoning between hops of inference. Thus, we propose Necessary and Optional Information Cell (NOIC), a variation of our base reasoning cell used in the reasoning layer that is capable of incorporating optional helpful information.

NOIC This cell is an extension to the base reasoning cell that allows the model to use commonsense information to fill in gaps of reasoning. An example of this is on the bottom left of Fig. 1, where we see that the cell first performs the operations done in the base reasoning cell and then adds optional, commonsense information.

At reasoning step t, after obtaining the output of the base reasoning cell, $\mathbf{c}^t$, we create a cell-specific representation for commonsense information by concatenating the embedded commonsense paths so that each path has a single vector representation, $\mathbf{u}_i^{CS}$. We then project it to the same dimension as $\mathbf{c}_i^t$: $\mathbf{v}_i^{CS} = \text{ReLU}(W\mathbf{u}_i^{CS} + b)$ where W and b are trainable parameters.

We use an attention layer to model the interaction between commonsense and the context:

$$S_{ij}^{CS} = W_1^{CS}\mathbf{c}_i^t + W_2^{CS}\mathbf{v}_j^{CS} + W_3^{CS}(\mathbf{c}_i^t \odot \mathbf{v}_j^{CS})$$

$$p_{ij}^{CS} = \frac{\exp(S_{ij}^{CS})}{\sum_{k=1}^{l}\exp(S_{ij}^{CS})}$$

$$\mathbf{c}_i^{CS} = \sum_{j=1}^{l} p_{ij}^{CS}\mathbf{v}_j^{CS}$$

Finally, we combine this commonsense-aware context representation with the original $\mathbf{c}_i^t$ via a sigmoid gate, since commonsense information is often not necessary at every step of inference:

$$\mathbf{z}_i = \sigma(W_z[\mathbf{c}_i^{CS}; \mathbf{c}_i^t] + b_z)$$

$$(\mathbf{c_o})_i^t = \mathbf{z}_i \odot \mathbf{c}_i^t + (1 - \mathbf{z}_i) \odot \mathbf{c}_i^{CS}$$

We use ${\mathbf{c_o}}^t$ as the output of the current reasoning step instead of $\mathbf{c}^t$. As we replace each base reasoning cell with NOIC, we selectively incorporate commonsense at every step of inference.

4 Experimental Setup

Datasets: We report results on two multi-hop reasoning datasets: generative NarrativeQA (Kočiský et al., 2018) (summary subtask) and extractive QAngaroo WikiHop (Welbl et al., 2018). For multiple-choice WikiHop, we rank candidate responses by their generation probability. Similar to previous works (Dhingra et al., 2018), we use the non-oracle, unmasked and not-validated dataset.

Evaluation Metrics: We evaluate NarrativeQA on the metrics proposed by its original authors: Bleu-1, Bleu-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and Rouge-L (Lin, 2004). We also evaluate on CIDEr (Vedantam et al., 2015) which emphasizes annotator consensus. For WikiHop, we evaluate on accuracy.[5]

More dataset, metric, and all other training details are in the supplementary.

---

[4] In cases where more than one relation can be used to make a hop, we pick one at random.

[5] Due to the 2-week evaluation wait-time on the non-public test set, we instead train our model on a sub-section of the training set, pick hyperparameters based on a small

Model	BLEU-1	BLEU-4	METEOR	Rouge-L	CIDEr
Seq2Seq (Kočiskỳ et al., 2018)	15.89	1.26	4.08	13.15	-
ASR (Kočiskỳ et al., 2018)	23.20	6.39	7.77	22.26	-
BiDAF† (Kočiskỳ et al., 2018)	33.72	15.53	15.38	36.30	-
BiAttn + MRU-LSTM† (Tay et al., 2018)	36.55	19.79	17.87	41.44	-
MHPGM	40.24	17.40	17.33	41.49	139.23
MHPGM+ NOIC	**43.63**	**21.07**	**19.03**	**44.16**	**152.98**

Table 2: Results across different metrics on the test set of NarrativeQA-summaries task. † indicates span prediction models trained on the Rouge-L retrieval oracle.

Model	Acc (%)
BiDAF (Welbl et al., 2018)	42.09
Coref-GRU (Dhingra et al., 2018)	56.00
MHPGM	56.74
MHPGM+ NOIC	**58.22**

Table 3: Results of our models on WikiHop dataset.

5 Results

5.1 Main Experiment

The results of our model on both NarrativeQA and WikiHop with and without commonsense incorporation are shown in Table 2 and Table 3. We see empirically that our model outperforms all generative models on NarrativeQA, and is competitive with the top span prediction models. Furthermore, with the NOIC commonsense integration, we were able to further improve performance ($p < 0.001$ on all metrics[6]), establishing a new state-of-the-art for the task. We also see that our model performs well on WikiHop,[7] and is further improved via the addition of commonsense ($p < 0.001$), demonstrating the generalizability of both our model and commonsense addition techniques.[8]

5.2 Model Ablations

We also tested the effectiveness of each component of our architecture as well as the effectiveness of adding commonsense information on the NarrativeQA validation set, with results shown in Table 4. Experiment 1 and 5 are our models presented in Table 2. Experiment 2 demonstrates the importance of multi-hop attention by showing that if we only allow one hop of attention (even with all other components of the model, including ELMo embeddings) the model's performance decreases by over 12 Rouge-L points. Experiment 3 and 4 demonstrate the effectiveness of other parts of our model. We see that ELMo embeddings (Peters et al., 2018) were also important for the model's performance and that self-attention is able to contribute significantly to performance on top of other components of the model. Finally, we see that effectively introducing external knowledge via our commonsense selection algorithm and NOIC can improve performance even further on top of our strong baseline.

5.3 Commonsense Ablations

We also conducted experiments testing the effectiveness of our commonsense selection and incorporation techniques. We first tried to naively add ConceptNet information by initializing the word embeddings with the ConceptNet-trained embeddings, NumberBatch (Speer and Havasi, 2012) (we also change embedding size from 256 to 300). Then, to verify the effectiveness of our commonsense selection and grounding algorithm, we test our best model on in-domain noise by giving each context-query pair a set of random relations grounded in other context-query pairs. This should teach the model about general commonsense relations present in the domain of NarrativeQA but does not provide grounding that fills in specific hops of inference. We also experimented with a simpler commonsense extraction method of using a single hop from the query to the context. The results of these are shown in Table 5, where we see that neither NumberBatch nor random-relationships nor single-hop commonsense offer statistically significant improvements[9],

---

(500 examples) held-out part of the training set, and test on the original validation set (by treating it as an unseen test set). We will promptly include the non-public test set results in the next version and at: `https://github.com/yicheng-w/CommonSenseMultiHopQA`

[6] Stat. significance computed using bootstrap test with 100K iterations (Noreen, 1989; Efron and Tibshirani, 1994).

[7] Note that we compare our model's performance to other models' tuned performance on the development set and ours is still equal or better.

[8] All results here are for the standard (non-oracle) unmasked and not-validated dataset. Welbl et al. (2018) has reported higher numbers on different data settings which are not comparable to our results.

[9] The improvement in Rouge-L and METEOR for all three ablation approaches have $p \geq 0.15$ with the bootstrap test.

#	Ablation	B-1	B-4	M	R	C
1	-	42.3	18.9	18.3	44.9	151.6
2	$k = 1$	32.5	11.7	12.9	32.4	95.7
3	- ELMo	32.8	12.7	13.6	33.7	103.1
4	- Self-Attn	37.0	16.4	15.6	38.6	125.6
5	+ NOIC	**46.0**	**21.9**	**20.7**	**48.0**	**166.6**

Table 4: Model ablations on NarrativeQA val-set.

Commonsense	B-1	B-4	M	R	C
None	42.3	18.9	18.3	44.9	151.6
NumberBatch	42.6	19.6	18.6	44.4	148.1
Random Rel.	43.3	19.3	18.6	45.2	151.2
Single Hop	42.1	19.9	18.2	44.0	148.6
Grounded Rel.	**45.9**	**21.9**	**20.7**	**48.0**	**166.6**

Table 5: Commonsense ablations on NarrativeQA val-set.

	Commonsense Required	
	Yes	No
Relevant CS Extracted	34%	14%
Irrelevant CS Extracted	16%	36%

Table 6: NarrativeQA's commonsense requirements and effectiveness of commonsense selection algorithm.

MHPGM+NOIC better	23%
MHPGM better	15%
Indistinguishable (Both-good)	41%
Indistinguishable (Both-bad)	21%

Table 7: Human evaluation on the output quality of the MHPGM+NOIC vs. MHPGM in terms of correctness.

whereas our commonsense selection and incorporation mechanism improves performance significantly across all metrics. We also present several examples of extracted commonsense and its model attention visualization in the supplementary.

6 Human Evaluation Analysis

We also conduct human evaluation analysis on both the quality of the selected commonsense relations, as well as the performance of our final model.

Commonsense Selection: We conducted manual analysis on a 50 sample subset of the NarrativeQA test set to check the effectiveness of our commonsense selection algorithm. Specifically, given a context-query pair, as well as the commonsense selected by our algorithm, we conduct two independent evaluations: (1) was any external commonsense knowledge necessary for answering the question?; (2) were the commonsense relations provided by our algorithm relevant to the question? The result for these two evaluations as well as how they overlap with each other are shown in Table 6, where we see that 50% of the cases required external commonsense knowledge, and on a majority (34%) of those cases our algorithm was able to select the correct/relevant commonsense information to fill in gaps of inference. We also see that in general, our algorithm was able to provide useful commonsense 48% of the time.

Model Performance: We also conduct human evaluation to verify that our commonsense incorporated model was indeed better than MHPGM. We randomly selected 100 examples from the NarrativeQA test set, along with both models' predicted answers, and for each datapoint, we asked 3 external human evaluators (fluent English speakers) to decide (without knowing which model produced each response) if one is strictly better than the other, or that they were similar in quality (both-good or both-bad). As shown in Table 7, we see that the human evaluation results are in agreement with that of the automatic evaluation metrics: our commonsense incorporation has a reasonable impact on the overall correctness of the model. The inter-annotator agreement had a Fleiss $\kappa = 0.831$, indicating 'almost-perfect' agreement between the annotators (Landis and Koch, 1977).

7 Conclusion

We present an effective reasoning-generative QA architecture that is a novel combination of previous work, which uses multiple hops of bidirectional attention and a pointer-generator decoder to effectively perform multi-hop reasoning and synthesize a coherent and correct answer. Further, we introduce an algorithm to select grounded, useful paths of commonsense knowledge to fill in the gaps of inference required for QA, as well a Necessary and Optional Information Cell (NOIC) which successfully incorporates this information during multi-hop reasoning to achieve the new state-of-the-art on NarrativeQA.

Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by DARPA (YFA17-D17AP00022), Google Faculty Research Award, Bloomberg Data Science Research Grant, and NVidia GPU awards. The views contained in this article are those of the authors and not of the funding agency.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

Erik Cambria, Amir Hussain, Tariq Durrani, Catherine Havasi, Chris Eckl, and James Munro. 2010. Sentic computing for patient centered applications. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 1279–1282. IEEE.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2406–2417. Association for Computational Linguistics.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pages 845–855.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 42–48.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1832–1846.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *AAAI*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 908–918.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832. Association for Computational Linguistics.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Soujanya Poria, Erik Cambria, Alexander Gelbukh, Federica Bisio, and Amir Hussain. 2015. Sentiment data flow analysis by means of dynamic linguistic patterns. *IEEE Computational Intelligence Magazine*, 10(4):26–36.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Federica Bisio. 2016. Sentic LDA: Improving on LDA with semantic similarity for aspect-based sentiment analysis. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 4465–4473. IEEE.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in ConceptNet 5. In *LREC*, pages 3679–3686.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2440–2448.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-range reasoning for machine comprehension. *arXiv preprint arXiv:1803.09074*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575.

Qiu-Feng Wang, Erik Cambria, Cheng-Lin Liu, and Amir Hussain. 2013. Common sense knowledge for handwritten chinese text recognition. *Cognitive Computation*, 5(2):234–242.

Dirk Weissenborn, Tomáš Kočiský, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural NLU systems. *arXiv preprint arXiv:1706.02596*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *ICLR*.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2017. Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling. In *International Joint Conference on Neural Networks*.

Tom Young, Erik Cambria, Iti Chaturvedi, Minlie Huang, Hao Zhou, and Subham Biswas. 2018. Augmenting end-to-end dialog systems with commonsense knowledge. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text

Haitian Sun* **Bhuwan Dhingra*** **Manzil Zaheer** **Kathryn Mazaitis**
Ruslan Salakhutdinov **William W. Cohen**
School of Computer Science
Carnegie Mellon University
`{haitians,bdhingra,manzilz,krivard,rsalakhu,wcohen}@cs.cmu.edu`

Abstract

Open Domain Question Answering (QA) is evolving from complex pipelined systems to end-to-end deep neural networks. Specialized neural models have been developed for extracting answers from either text alone or Knowledge Bases (KBs) alone. In this paper we look at a more practical setting, namely QA over the combination of a KB and entity-linked text, which is appropriate when an *incomplete* KB is available with a large text corpus. Building on recent advances in graph representation learning we propose a novel model, GRAFT-Net, for extracting answers from a question-specific subgraph containing text and KB entities and relations. We construct a suite of benchmark tasks for this problem, varying the difficulty of questions, the amount of training data, and KB completeness. We show that GRAFT-Net is competitive with the state-of-the-art when tested using either KBs or text alone, and vastly outperforms existing methods in the combined setting.

1 Introduction

Open domain Question Answering (QA) is the task of finding answers to questions posed in natural language. Historically, this required a specialized pipeline consisting of multiple machine-learned and hand-crafted modules (Ferrucci et al., 2010). Recently, the paradigm has shifted towards training end-to-end deep neural network models for the task (Chen et al., 2017; Liang et al., 2017; Raison et al., 2018; Talmor and Berant, 2018; Iyyer et al., 2017). Most existing models, however, answer questions using a *single* information source, usually either text from an encyclopedia, or a single knowledge base (KB).

Intuitively, the suitability of an information source for QA depends on both its *coverage* and the *difficulty* of extracting answers from it. A large text corpus has high coverage, but the information is expressed using many different text patterns. As a result, models which operate on these patterns (e.g. BiDAF (Seo et al., 2017)) do not generalize beyond their training domains (Wiese et al., 2017; Dhingra et al., 2018) or to novel types of reasoning (Welbl et al., 2018; Talmor and Berant, 2018). KBs, on the other hand, suffer from low coverage due to their inevitable incompleteness and restricted schema (Min et al., 2013), but are easier to extract answers from, since they are constructed precisely for the purpose of being queried.

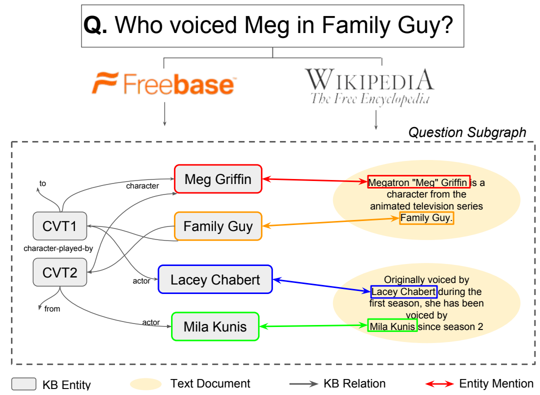

Figure 1: To answer a question posed in natural language, GRAFT-Net considers a heterogeneous graph constructed from text and KB facts, and thus can leverage the rich relational structure between the two information sources.

In practice, some questions are best answered using text, while others are best answered using KBs. A natural question, then, is how to effectively combine both types of information. Surprisingly little prior work has looked at this problem. In this paper we focus on a scenario in which a large-scale KB (Bollacker et al., 2008; Auer et al., 2007) and a text corpus are available, but neither is sufficient alone for answering all questions.

---

[*] Haitian Sun and Bhuwan Dhingra contributed equally to this work.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4231–4242
Brussels, Belgium, October 31 - November 4, 2018.

A naïve option, in such a setting, is to take state-of-the-art QA systems developed for each source, and aggregate their predictions using some heuristic (Ferrucci et al., 2010; Baudiš, 2015). We call this approach *late fusion*, and show that it can be sub-optimal, as models have limited ability to aggregate evidence across the different sources (§ 5.4). Instead, we focus on an *early fusion* strategy, where a single model is trained to extract answers from a *question subgraph* (Fig 1) containing relevant KB facts as well as text sentences. Early fusion allows more flexibility in combining information from multiple sources.

To enable early fusion, in this paper we propose a novel graph convolution based neural network, called GRAFT-Net (Graphs of Relations Among Facts and Text Networks), specifically designed to operate over heterogeneous graphs of KB facts and text sentences. We build upon recent work on graph representation learning (Kipf and Welling, 2016; Schlichtkrull et al., 2017), but propose two key modifications to adopt them for the task of QA. First, we propose *heterogeneous update rules* that handle KB nodes differently from the text nodes: for instance, LSTM-based updates are used to propagate information into and out of text nodes (§ 3.2). Second, we introduce a *directed propagation method*, inspired by personalized Pagerank in IR (Haveliwala, 2002), which constrains the propagation of embeddings in the graph to follow paths starting from seed nodes linked to the question (§ 3.3). Empirically, we show that both these extensions are crucial for the task of QA.

We evaluate these methods on a new suite of benchmark tasks for testing QA models when both KB and text are present. Using WikiMovies (Miller et al., 2016) and WebQuestionsSP (Yih et al., 2016), we construct datasets with a varying amount of training supervision and KB completeness, and with a varying degree of question complexity. We report baselines for future comparison, including Key Value Memory Networks (Miller et al., 2016; Das et al., 2017c), and show that our proposed GRAFT-Nets have superior performance across a wide range of conditions (§ 5). We also show that GRAFT-Nets are competitive with the state-of-the-art methods developed specifically for text-only QA, and state-of-the art methods developed for KB-only QA (§ 5.4)[1].

2 Task Setup

2.1 Description

A knowledge base is denoted as $\mathcal{K} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ is the set of entities in the KB, and the edges $\mathcal{E}$ are triplets (s, r, o) which denote that relation $r \in \mathcal{R}$ holds between the subject $s \in \mathcal{V}$ and object $o \in \mathcal{V}$. A text corpus $\mathcal{D}$ is a set of documents $\{d_1, \dots, d_{|\mathcal{D}|}\}$ where each document is a sequence of words $d_i = (w_1, \dots, w_{|d_i|})$. We further assume that an (imperfect) entity linking system has been run on the collection of documents whose output is a set $\mathcal{L}$ of links (v, d_p) connecting an entity $v \in \mathcal{V}$ with a word at position p in document d, and we denote with $\mathcal{L}_d$ the set of all entity links in document d. For entity mentions spanning multiple words in d, we include links to all the words in the mention in $\mathcal{L}$.

The task is, given a natural language question $q = (w_1, \dots, w_{|q|})$, extract its answers $\{a\}_q$ from $\mathcal{G} = (\mathcal{K}, \mathcal{D}, \mathcal{L})$. There may be multiple correct answers for a question. In this paper, we assume that the answers are entities from either the documents or the KB. We are interested in a wide range of settings, where the KB $\mathcal{K}$ varies from highly incomplete to complete for answering the questions, and we will introduce datasets for testing our models under these settings.

To solve this task we proceed in two steps. First, we extract a subgraph $\mathcal{G}_q \subset \mathcal{G}$ which contains the answer to the question with high probability. The goal for this step is to ensure high recall for answers while producing a graph small enough to fit into GPU memory for gradient-based learning. Next, we use our proposed model GRAFT-Net to learn node representations in $\mathcal{G}_q$, conditioned on q, which are used to classify each node as being an answer or not. Training data for the second step is generated using distant supervision. The entire process mimics the search-and-read paradigm for text-based QA (Dhingra et al., 2017).

2.2 Question Subgraph Retrieval

We retrieve the subgraph $\mathcal{G}_q$ using two parallel pipelines – one over the KB $\mathcal{K}$ which returns a set of entities, and the other over the corpus $\mathcal{D}$ which returns a set of documents. The retrieved entities and documents are then combined with entity links to produce a fully-connected graph.

KB Retrieval. To retrieve relevant entities from the KB we first perform entity linking on the ques-

---

[1] Source code and data are available at `https://github.com/OceanskySun/GraftNet`

tion q, producing a set of *seed entities*, denoted S_q. Next we run the Personalized PageRank (PPR) method (Haveliwala, 2002) around these seeds to identify other entities which might be an answer to the question. The edge-weights around S_q are distributed equally among all edges of the same type, and they are weighted such that edges relevant to the question receive a higher weight than those which are not. Specifically, we average word vectors to compute a relation vector $v(r)$ from the surface form of the relation, and a question vector $v(q)$ from the words in the question, and use cosine similarity between these as the edge weights. After running PPR we retain the top E entities $v_1, \ldots, v_E$ by PPR score, along with any edges between them, and add them to $\mathcal{G}_q$.

Text Retrieval. We use Wikipedia as the corpus and retrieve text at the sentence level, i.e. documents in $\mathcal{D}$ are defined along sentences boundaries[2]. We perform text retrieval in two steps: first we retrieve the top 5 most relevant Wikipedia articles, using the weighted bag-of-words model from DrQA (Chen et al., 2017); then we populate a Lucene[3] index with sentences from these articles, and retrieve the top ranking ones $d_1, \ldots, d_D$, based on the words in the question. For the sentence-retrieval step, we found it beneficial to include the title of the article as an additional field in the Lucene index. As most sentences in an article talk about the title entity, this helps in retrieving relevant sentences that do not explicitly mention the entity in the question. We add the retrieved documents, along with any entities linked to them, to the subgraph $\mathcal{G}_q$.

The final question subgraph is $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_q, \mathcal{R}^+)$, where the vertices $\mathcal{V}_q$ consist of all the retrieved entities and documents, i.e. $\mathcal{V}_q = \{v_1, \ldots, v_E\} \cup \{d_1, \ldots, d_D\}$. The edges are all relations from $\mathcal{K}$ among these entities, plus the entity-links between documents and entities, i.e.

$$\begin{aligned}\mathcal{E}_q = &\{(s, o, r) \in \mathcal{E} : s, o \in \mathcal{V}_q, r \in \mathcal{R}\} \\ &\cup \{(v, d_p, r_L) : (v, d_p) \in \mathcal{L}_d, d \in \mathcal{V}_q\},\end{aligned}$$

where r_L denotes a special "linking" relation. $\mathcal{R}^+ = \mathcal{R} \cup \{r_L\}$ is the set of all edge types in the subgraph.

3 GRAFT-Nets

The question q and its answers $\{a\}_q$ induce a labeling of the nodes in $\mathcal{V}_q$: we let $y_v = 1$ if $v \in \{a\}_q$ and $y_v = 0$ otherwise for all $v \in \mathcal{V}_q$. The task of QA then reduces to performing binary classification over the nodes of the graph $\mathcal{G}_q$. Several graph-propagation based models have been proposed in the literature which learn node representations and then perform classification of the nodes (Kipf and Welling, 2016; Schlichtkrull et al., 2017). Such models follow the standard gather-apply-scatter paradigm to learn the node representation with homogeneous updates, i.e. treating all neighbors equally.

The basic recipe for these models is as follows:

1. Initialize node representations $h_v^{(0)}$.
2. For $l = 1, \ldots, L$ update node representations

$$h_v^{(l)} = \phi\left(h_v^{(l-1)}, \sum_{v' \in N_r(v)} h_{v'}^{(l-1)}\right),$$

where $N_r(v)$ denotes the neighbours of v along incoming edges of type r, and ϕ is a neural network layer.

Here L is the number of *layers* in the model and corresponds to the maximum length of the paths along which information should be propagated in the graph. Once the propagation is complete the final layer representations $h_v^{(L)}$ are used to perform the desired task, for example link prediction in knowledge bases (Schlichtkrull et al., 2017).

However, there are two differences in our setting from previously studied graph-based classification tasks. The first difference is that, in our case, the graph $\mathcal{G}_q$ consists of *heterogeneous* nodes. Some nodes in the graph correspond to KB entities which represent symbolic objects, whereas other nodes represent textual documents which are variable length sequences of words. The second difference is that we want to condition the representation of nodes in the graph on the natural language question q. In §3.2 we introduce heterogeneous updates to address the first difference, and in §3.3 we introduce mechanisms for conditioning on the question (and its entities) for the second.

3.1 Node Initialization

Nodes corresponding to entities are initialized using fixed-size vectors $h_v^{(0)} = x_v \in \mathbb{R}^n$, where

---

[2] The term *document* will always refer to a sentence in the rest of this paper.

[3] `https://lucene.apache.org/`

x_v can be pre-trained KB embeddings or random, and n is the embedding size. Document nodes in the graph describe a variable length sequence of text. Since multiple entities might link to different positions in the document, we maintain a variable length representation of the document in each layer. This is denoted by $H_d^{(l)} \in \mathbb{R}^{|d| \times n}$. Given the words in the document $(w_1, \ldots, w_{|d|})$, we initialize its hidden representation as:

$$H_d^{(0)} = \text{LSTM}(w_1, w_2, \ldots),$$

where LSTM refers to a long short-term memory unit. We denote the p-th row of $H_d^{(l)}$, corresponding to the embedding of p-th word in the document d at layer l, as $H_{d,p}^{(l)}$.

3.2 Heterogeneous Updates

Figure 2 shows the update rules for entities and documents, which we describe in detail here.

Entities. Let $M(v) = \{(d,p)\}$ be the set of positions p in documents d which correspond to a mention of entity v. The update for entity nodes involves a single-layer feed-forward network (FFN) over the concatenation of four states:

$$h_v^{(l)} = \text{FFN}\left(\begin{bmatrix} h_v^{(l-1)} \\ h_q^{(l-1)} \\ \sum_r \sum_{v' \in N_r(v)} \alpha_r^{v'} \psi_r(h_{v'}^{(l-1)}) \\ \sum_{(d,p) \in M(v)} H_{d,p}^{(l-1)} \end{bmatrix}\right). \quad (1)$$

The first two terms correspond to the entity representation and question representation (details below), respectively, from the previous layer.

The third term aggregates the states from the entity neighbours of the current node, $N_r(v)$, after scaling with an attention weight $\alpha_r^{v'}$ (described in the next section), and applying relation specific transformations ψ_r. Previous work on Relational-Graph Convolution Networks (Schlichtkrull et al., 2017) used a linear projection for ψ_r. For a batched implementation, this results in matrices of size $O(B|\mathcal{R}_q||\mathcal{E}_q|n)$, where B is the batch size, which can be prohibitively large for large subgraphs[4]. Hence in this work we use *relation vectors* x_r for $r \in \mathcal{R}_q$ instead of matrices, and compute the update along an edge as:

$$\psi_r(h_{v'}^{(l-1)}) = pr_{v'}^{(l-1)} \text{FFN}\left(x_r, h_{v'}^{(l-1)}\right). \quad (2)$$

Here $pr_{v'}^{(l-1)}$ is a PageRank score used to control the propagation of embeddings along paths starting from the seed nodes, which we describe in detail in the next section. The memory complexity of the above is $O(B(|\mathcal{F}_q| + |\mathcal{E}_q|)n)$, where $|\mathcal{F}_q|$ is the number of facts in the subgraph $\mathcal{G}_q$.

The last term aggregates the states of all tokens that correspond to mentions of the entity v among the documents in the subgraph. Note that the update depends on the positions of entities in their containing document.

Documents. Let $L(d,p)$ be the set of all entities linked to the word at position p in document d. The document update proceeds in two steps. First we aggregate over the entity states coming in at each position separately:

$$\tilde{H}_{d,p}^{(l)} = \text{FFN}\left(H_{d,p}^{(l-1)}, \sum_{v \in L(d,p)} h_v^{(l-1)}\right). \quad \text{(3a)}$$

Here $h_v^{(l-1)}$ are normalized by the number of outgoing edges at v. Next we aggregate states within the document using an LSTM:

$$H_d^{(l)} = \text{LSTM}(\tilde{H}_d^{(l)}). \quad \text{(3b)}$$

3.3 Conditioning on the Question

For the parts described thus far, the graph learner is largely agnostic of the question. We introduce dependence on question in two ways: by attention over relations, and by personalized propagation.

To represent q, let $w_1^q, \ldots, w_{|q|}^q$ be the words in the question. The initial representation is computed as:

$$h_q^{(0)} = \text{LSTM}(w_1^q, \ldots, w_{|q|}^q)_{|q|} \in \mathbb{R}^n, \quad (4)$$

where we extract the final state from the output of the LSTM. In subsequent layers the question representation is updated as $h_q^{(l)} = \text{FFN}\left(\sum_{v \in S_q} h_v^{(l)}\right)$, where S_q denotes the seed entities mentioned in the question.

Attention over Relations. The attention weight in the third term of Eq. (1) is computed using the question and relation embeddings:

$$\alpha_r^{v'} = \text{softmax}(x_r^T h_q^{(l-1)}),$$

where the softmax normalization is over all outgoing edges from v', and x_r is the relation vector for relation r. This ensures that embeddings are propagated more along edges relevant to the question.

---

[4] This is because we have to use adjacency matrices of size $|\mathcal{R}_q| \times |\mathcal{E}_q| \times |\mathcal{E}_q|$ to aggregate embeddings from neighbours of all nodes simultaneously.

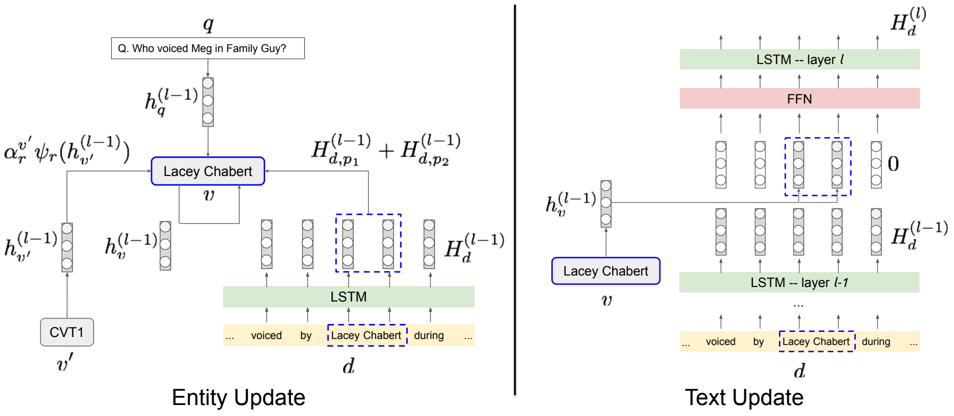

Figure 2: Illustration of the heterogeneous update rules for entities (**left**) and text documents (**right**)

Directed Propagation. Many questions require multi-hop reasoning, which follows a path from a seed node mentioned in the question to the target answer node. To encourage such a behaviour when propagating embeddings, we develop a technique inspired from personalized PageRank in IR (Haveliwala, 2002). The propagation starts at the seed entities S_q mentioned in the question. In addition to the vector embeddings $h_v^{(l)}$ at the nodes, we also maintain scalar "PageRank" scores $pr_v^{(l)}$ which measure the total weight of paths from a seed entity to the current node, as follows:

$$pr_v^{(0)} = \begin{cases} \frac{1}{|S_q|} & \text{if} \quad v \in S_q \\ 0 & \text{o.w.} \end{cases},$$

$$pr_v^{(l)} = (1-\lambda)pr_v^{(l-1)} + \lambda \sum_r \sum_{v' \in N_r(v)} \alpha_r^{v'} pr_{v'}^{(l-1)}.$$

Notice that we reuse the attention weights $\alpha_r^{v'}$ when propagating PageRank, to ensure that nodes along paths relevant to the question receive a high weight. The PageRank score is used as a scaling factor when propagating embeddings along the edges in Eq. (2). For $l = 1$, the PageRank score will be 0 for all entities except the seed entities, and hence propagation will only happen outward from these nodes. For $l = 2$, it will be non-zero for the seed entities and their 1-hop neighbors, and propagation will only happen along these edges. Figure 3 illustrates this process.

3.4 Answer Selection

The final representations $h_v^{(L)} \in \mathbb{R}^n$, are used for binary classification to select the answers:

$$\Pr\left(v \in \{a\}_q | \mathcal{G}_q, q\right) = \sigma(w^T h_v^{(L)} + b), \tag{5}$$

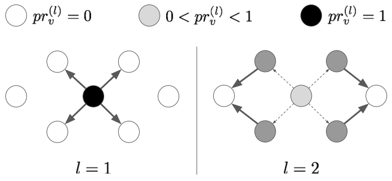

Figure 3: Directed propagation of embeddings in GRAFT-Net. A scalar *PageRank* score $pr_v^{(l)}$ is maintained for each node v across layers, which spreads out from the seed node. Embeddings are only propagated from nodes with $pr_v^{(l)} > 0$.

where σ is the sigmoid function. Training uses binary cross-entropy loss over these probabilities.

3.5 Regularization via Fact Dropout

To encourage the model to learn a robust classifier, which exploits all available sources of information, we randomly drop edges from the graph during training with probability p_0. We call this *fact-dropout*. It is usually easier to extract answers from the KB than from the documents, so the model tends to rely on the former, especially when the KB is complete. This method is similar to DropConnect (Wan et al., 2013).

4 Related Work

The work of Das et al. (2017c) attempts an early fusion strategy for QA over KB facts and text. Their approach is based on Key-Value Memory Networks (KV-MemNNs) (Miller et al., 2016) coupled with a universal schema (Riedel et al., 2013) to populate a memory module with representations of KB triples and text snippets independently. The key limitation for this model is that it ignores the rich relational structure between the facts and text snippets. Our graph-based method, on the other hand, explicitly uses this structure for the propagation of embeddings. We compare the two approaches in our experiments (§5), and show that GRAFT-Nets outperform KV-MemNNs over all tasks.

Non-deep learning approaches have been also attempted for QA over both text assertions and KB facts. Gardner and Krishnamurthy (2017) use traditional feature extraction methods of open-vocabulary semantic parsing for the task. Ryu et al. (2014) use a pipelined system aggregating evidence from both unstructured and semi-structured sources for open-domain QA.

Another line of work has looked at learning combined representations of KBs and text for relation extraction and Knowledge Base Completion (KBC) (Lao et al., 2012; Riedel et al., 2013; Toutanova et al., 2015; Verga et al., 2016; Das et al., 2017b; Han et al., 2016). The key difference in QA compared to KBC is that in QA the inference process on the knowledge source has to be conditioned on the question, so different questions induce different representations of the KB and warrant a different inference process. Furthermore, KBC operates under the fixed schema defined by the KB before-hand, whereas natural language questions might not adhere to this schema.

The GRAFT-Net model itself is motivated from the large body of work on graph representation learning (Scarselli et al., 2009; Li et al., 2016; Kipf and Welling, 2016; Atwood and Towsley, 2016; Schlichtkrull et al., 2017). Like most other graph-based models, GRAFT-Nets can also be viewed as an instantiation of the Message Passing Neural Network (MPNN) framework of Gilmer et al. (2017). GRAFT-Nets are also *inductive* representation learners like GraphSAGE (Hamilton et al., 2017), but operate on a heterogeneous mixture of nodes and use retrieval for getting a subgraph instead of random sampling. The recently proposed Walk-Steered Convolution model uses random walks for learning graph representations (Jiang et al., 2018). Our personalization technique also borrows from such random walk literature, but uses it to localize propagation of embeddings.

Tremendous progress on QA over KB has been made with deep learning based approaches like memory networks (Bordes et al., 2015; Jain, 2016) and reinforcement learning (Liang et al., 2017; Das et al., 2017a). But extending them with text, which is our main focus, is non-trivial. In another direction, there is also work on producing parsimonious graphical representations of textual data (Krause et al., 2016; Lu et al., 2017); however in this paper we use a simple sequential representation augmented with entity links to the KB which works well.

For QA over text only, a major focus has been on the task of reading comprehension (Seo et al., 2017; Gong and Bowman, 2017; Hu et al., 2017; Shen et al., 2017; Yu et al., 2018) since the introduction of SQuAD (Rajpurkar et al., 2016). These systems assume that the answer-containing passage is known apriori, but there has been progress when this assumption is relaxed (Chen et al., 2017; Raison et al., 2018; Dhingra et al., 2017; Wang et al., 2018, 2017; Watanabe et al., 2017). We work in the latter setting, where relevant information must be retrieved from large information sources, but we also incorporate KBs into this process.

5 Experiments & Results

5.1 Datasets

WikiMovies-10K consists of $10K$ randomly sampled training questions from the WikiMovies dataset (Miller et al., 2016), along with the original test and validation sets. We sample the training questions to create a more difficult setting, since the original dataset has $100K$ questions over only 8 different relation types, which is unrealistic in our opinion. In § 5.4 we also compare to the existing state-of-the-art using the full training set.

We use the KB and text corpus constructed from Wikipedia released by Miller et al. (2016). For entity linking we use simple surface level matches, and retrieve the top 50 entities around the seeds to create the question subgraph. We further add the top 50 sentences (along with their article titles) to the subgraph using Lucene search over the text corpus. The overall answer recall in our constructed subgraphs is 99.6%.

WebQuestionsSP (Yih et al., 2016) consists of 4737 natural language questions posed over Freebase entities, split up into 3098 training and 1639 test questions. We reserve 250 training questions for model development and early stopping. We use the entity linking outputs from S-MART[5] and retrieve 500 entities from the neighbourhood around the question seeds in Freebase to populate the

---

[5] `https://github.com/scottyih/STAGG`

question subgraphs[6]. We further retrieve the top 50 sentences from Wikipedia with the two-stage process described in §2. The overall recall of answers among the subgraphs is 94.0%.

Table 1 shows the combined statistics of all the retreived subgraphs for the questions in each dataset. These two datasets present varying levels of difficulty. While all questions in WikiMovies correspond to a single KB relation, for WebQuestionsSP the model needs to aggregate over two KB facts for ∼30% of the questions, and also requires reasoning over constraints for ∼7% of the questions (Liang et al., 2017). For maximum portability, QA systems need to be robust across several degrees of KB availability since different domains might contain different amounts of structured data; and KB completeness may also vary over time. Hence, we construct an additional 3 datasets each from the above two, with the number of KB facts downsampled to 10%, 30% and 50% of the original to simulate settings where the KB is incomplete. We repeat the retrieval process for each sampled KB.

5.2 Compared Models

KV-KB is the Key Value Memory Networks model from Miller et al. (2016); Das et al. (2017c) but using only KB and ignoring the text. **KV-EF** (early fusion) is the same model with access to both KB and text as memories. For text we use a BiLSTM over the entire sentence as keys, and entity mentions as values. This re-implementation shows better performance on the text-only and KB-only WikiMovies tasks than the results reported previously[7] (see Table 4). **GN-KB** is the GRAFT-Net model ignoring the text. **GN-LF** is a late fusion version of the GRAFT-Net model: we train two separate models, one using text only and the other using KB only, and then ensemble the two[8]. **GN-EF** is our main GRAFT-Net model with early fusion. **GN-EF+LF** is an ensemble over the GN-EF and GN-LF models, with the same ensembling method as GN-LF. We report Hits@1, which is the accuracy of the top-predicted answer from the model, and the F1 score. To compute the F1 score we tune a threshold on the development set to select answers based on binary probabilities for each node in the subgraph.

5.3 Main Results

Table 2 presents a comparison of the above models across all datasets. GRAFT-Nets (GN) shows consistent improvement over KV-MemNNs on both datasets in all settings, including KB only (-KB), text only (-EF, Text Only column), and early fusion (-EF). Interestingly, we observe a larger relative gap between the Hits and F1 scores for the KV models than we do for our GN models. We believe this is because the attention for KV is normalized over the memories, which are KB facts (or text sentences): hence the model is unable to assign high probabilities to multiple facts at the same time. On the other hand, in GN, we normalize the attention over *types of relations* outgoing from a node, and hence can assign high weights to all the correct answers.

We also see a consistent improvement of early fusion over late fusion (-LF), and by ensembling them together we see the best performance across all the models. In Table 2 (right), we further show the improvement for KV-EF over KV-KB, and GN-LF and GN-EF over GN-KB, as the amount of KB is increased. This measures how effective these approaches are in utilizing text plus a KB. For KV-EF we see improvements when the KB is highly incomplete, but in the full KB setting, the performance of the fused approach is worse. A similar trend holds for GN-LF. On the other hand, GN-EF with text improves over the KB-only approach in all settings. As we would expect, though, the benefit of adding text decreases as the KB becomes more and more complete.

5.4 Comparison to Specialized Methods

In Table 4 we compare GRAFT-Nets to state-of-the-art models that are specifically designed and tuned for QA using either only KB or only text. For this experiment we use the full WikiMovies dataset to enable direct comparison to previously reported numbers. For DrQA (Chen et al., 2017), following the original paper, we restrict answer spans for WebQuestionsSP to match an entity in Freebase. In each case we also train GRAFT-Nets using only KB facts or only text sentences. In three out of the four cases, we find that GRAFT-Nets ei-

---

[6] A total of 13 questions had no detected entities. These were ignored during training and considered as incorrect during evaluation.

[7] For all KV models we tuned the number of layers $\{1, 2, 3\}$, batch size $\{10, 30, 50\}$, model dimension $\{50, 80\}$. We also use fact dropout regularization in the KB+Text setting tuned between $\{0, 0.2, 0.4\}$.

[8] For ensembles we take a weighted combination of the answer probabilities produced by the models, with the weights tuned on the dev set. For answers only in text or only in KB, we use the probability as is.

Dataset	# train / dev / test	# entity nodes	# edge types	# document nodes	# question vocab
WikiMovies-10K	10K / 10K / 10K	43,233	9	79,728	1759
WebQuestionsSP	2848 / 250 / 1639	528,617	513	235,567	3781

Table 1: Statistics of all the retrieved subgraphs $\cup_q \mathcal{G}_q$ for WikiMovies-10K and WebQuestionsSP.

Model	Text Only	KB + Text			
		10 %	30%	50%	100%
WikiMovies-10K					
KV-KB	–	15.8 / 9.8	44.7 / 30.4	63.8 / 46.4	94.3 / 76.1
KV-EF	50.4 / 40.9	53.6 / 44.0	60.6 / 48.1	75.3 / 59.1	93.8 / 81.4
GN-KB	–	19.7 / 17.3	48.4 / 37.1	67.7 / 58.1	**97.0 / 97.6**
GN-LF	**73.2 / 64.0**	74.5 / 65.4	78.7 / 68.5	83.3 / 74.2	96.5 / 92.0
GN-EF		75.4 / 66.3	82.6 / 71.3	87.6 / 76.2	96.9 / 94.1
GN-EF+LF		**79.0 / 66.7**	**84.6 / 74.2**	**88.4 / 78.6**	**96.8 / 97.3**
WebQuestionsSP					
KV-KB	–	12.5 / 4.3	25.8 / 13.8	33.3 / 21.3	46.7 / 38.6
KV-EF	23.2 / 13.0	24.6 / 14.4	27.0 / 17.7	32.5 / 23.6	40.5 / 30.9
GN-KB	–	15.5 / 6.5	34.9 / 20.4	47.7 / 34.3	66.7 / 62.4
GN-LF	**25.3 / 15.3**	29.8 / 17.0	39.1 / 25.9	46.2 / 35.6	65.4 / 56.8
GN-EF		31.5 / 17.7	40.7 / 25.2	49.9 / 34.7	67.8 / 60.4
GN-EF+LF		**33.3 / 19.3**	**42.5 / 26.7**	**52.3 / 37.4**	**68.7 / 62.3**

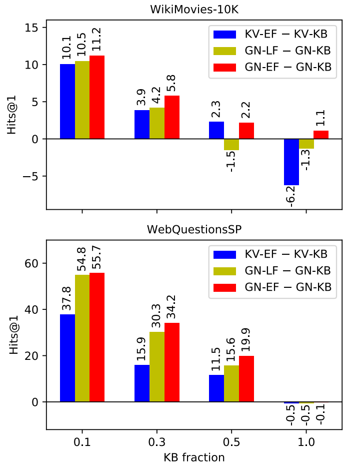

Table 2: **Left:** Hits@1 / F1 scores of GRAFT-Nets (GN) compared to KV-MemNN (KV) in KB only (-KB), early fusion (-EF), and late fusion (-LF) settings. **Right:** Improvement of early fusion (-EF) and late fusion (-LF) over KB only (-KB) settings as KB completeness increases.

ther match or outperform the existing state-of-the-art models. We emphasize that the latter have no mechanism for dealing with the fused setting.

The one exception is the KB-only case for WebQuestionsSP where GRAFT-Net does 6.2% F1 points worse than Neural Symbolic Machines (Liang et al., 2017). Analysis suggested three explanations: (1) In the KB-only setting, the recall of subgraph retrieval is only 90.2%, which limits overall performance. In an oracle setting where we ensure the answers are part of the subgraph, the F1 score increases by 4.8%. (2) We use the same probability threshold for all questions, even though the number of answers may vary significantly. Models which parse the query into a symbolic form do not suffer from this problem since answers are retrieved in a deterministic fashion. If we tune separate thresholds for each question the F1 score improves by 7.6%. (3) GRAFT-Nets perform poorly in the few cases where there is a constraint involved in picking out the answer (for example, "who *first* voiced Meg in Family Guy"). If we ignore such constraints, and consider all entities with the same sequence of relations to the seed as correct, the performance improves by 3.8% F1. Heuristics such as those used by Yu et al. (2017) can be used to improve these cases. Figure 3 shows examples where GRAFT-Net fails to predict the correct answer set exactly.

5.5 Effect of Model Components

Heterogeneous Updates. We tested a non-heterogeneous version of our model, where instead of using fine-grained entity linking information for updating the node representations ($M(v)$ and $L(d,p)$ in Eqs. 1, 3a), we aggregate the document states across all its positions as $\sum_p H_{d,p}^{(l)}$ and use this combined state for all updates. Without the heterogeneous update, all entities $v \in L(d,\cdot)$ will receive the same update from document d. Therefore, the model cannot disambiguate different entities mentioned in the same document. The result in Table 5 shows that this version is consistently worse than the heterogeneous model.

Conditioning on the Question. We performed an ablation test on the directed propagation method and attention over relations. We observe that both components lead to better performance. Such effects are observed in both complete and incomplete KB scenarios, e.g. on WebQuestionsSP dataset, as shown in Figure 4 (left).

Fact Dropout. Figure 4 (right) compares the performance of the early fusion model as we vary

Question	Correct Answers	Predicted Answers
what language do most people speak in afghanistan	Pashto language, Farsi (Eastern Language)	Pashto language
what college did john stockton go to	Gonzaga University	Gonzaga University, Gonzaga Preparatory School

Table 3: Examples from WebQuestionsSP dataset. **Top:** The model misses a correct answer. **Bottom:** The model predicts an extra incorrect answer.

Method	WikiMovies (full)		WebQuestionsSP	
	kb	doc	kb	doc
MINERVA	**97.0 / –**	–	–	–
R2-AsV	–	85.8 / –	–	–
NSM	–	–	**– / 69.0**	–
DrQA*	–	–	–	21.5 / –
R-GCN#	**96.5 / 97.4**	–	37.2 / 30.5	–
KV	93.9 / –	76.2 / –	– / –	– / –
KV#	95.6 / 88.0	80.3 / 72.1	46.7 / 38.6	23.2 / 13.0
GN	**96.8 / 97.2**	**86.6 / 80.8**	67.8 / 62.8	**25.3 / 15.3**

Table 4: Hits@1 / F1 scores compared to SOTA models using only KB or text: MINERVA (Das et al., 2017a), R2-AsV (Watanabe et al., 2017), Neural Symbolic Machines (NSM) (Liang et al., 2017), DrQA (Chen et al., 2017), R-GCN (Schlichtkrull et al., 2017) and KV-MemNN (Miller et al., 2016). *DrQA is pretrained on SQuAD. #Re-implemented.

	0 KB	0.1 KB	0.3 KB	0.5 KB	1.0 KB
NH	22.7 / 13.6	28.7 / 15.8	35.6 / 23.2	47.2 / 33.3	66.5 / 59.8
H	25.3 / 15.3	31.5 / 17.7	40.7 / 25.2	49.9 / 34.7	67.8 / 60.4

Table 5: Non-Heterogeneous (NH) vs. Heterogeneous (H) updates on WebQuestionsSP

the rate of fact dropout. Moderate levels of fact dropout improve performance on both datasets. The performance increases as the fact dropout rate increases until the model is unable to learn the inference chain from KB.

6 Conclusion

In this paper we investigate QA using text combined with an incomplete KB, a task which has received limited attention in the past. We introduce several benchmark problems for this task by modifying existing question-answering datasets, and discuss two broad approaches to solving this problem—"late fusion" and "early fusion". We show that early fusion approaches perform better.

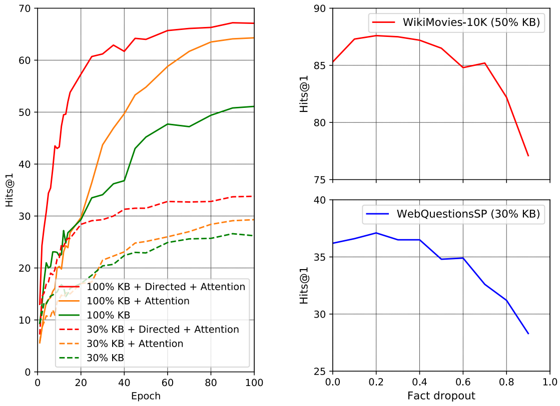

Figure 4: **Left:** Effect of directed propagation and query-based attention over relations for the WebQuestionsSP dataset with 30% KB and 100% KB. **Right:** Hits@1 with different rates of fact-dropout on and WikiMovies and WebQuestionsSP.

We also introduce a novel early-fusion model, called GRAFT-Net, for classifying nodes in subgraph consisting of both KB entities and text documents. GRAFT-Net builds on recent advances in graph representation learning but includes several innovations which improve performance on this task. GRAFT-Nets are a single model which achieve performance competitive to state-of-the-art methods in both text-only and KB-only settings, and outperform baseline models when using text combined with an incomplete KB. Current directions for future work include – (1) extending GRAFT-Nets to pick spans of text as answers, rather than only entities and (2) improving the subgraph retrieval process.

Acknowledgments

Bhuwan Dhingra is supported by NSF under grants CCF-1414030 and IIS-1250956 and by grants from Google. Ruslan Salakhutdinov is supported in part by ONR grant N000141812861, Apple, and Nvidia NVAIL Award.

References

James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Advances in Neu-*

ral Information Processing Systems, pages 1993–2001.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Petr Baudiš. 2015. Yodaqa: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*, pages 1156–1165.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017a. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017b. Chains of reasoning over entities, relations, and text using recurrent neural networks. *EACL*.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017c. Question answering on knowledge bases and text using universal schema and memory networks. *ACL*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Bhuwan Dhingra, Danish Pruthi, and Dheeraj Rajagopal. 2018. Simple and effective semi-supervised question answering. *NAACL*.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

Matt Gardner and Jayant Krishnamurthy. 2017. Open-vocabulary semantic parsing with both distributional statistics and formal knowledge. In *AAAI*, pages 3195–3201.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *ICML*.

Yichen Gong and Samuel R Bowman. 2017. Ruminating reader: Reasoning with gated multi-hop attention. *arXiv preprint arXiv:1704.07415*.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2016. Joint representation learning of text and knowledge for knowledge graph completion. *arXiv preprint arXiv:1611.04125*.

Taher H Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic reader for machine comprehension. *arXiv preprint arXiv:1705.02798*.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1821–1831.

Sarthak Jain. 2016. Question answering over knowledge base using factual memory networks. In *Proceedings of the NAACL Student Research Workshop*, pages 109–115.

Jiatao Jiang, Zhen Cui, Chunyan Xu, Chengzheng Li, and Jian Yang. 2018. Walk-steered convolution for graph classification. *arXiv preprint arXiv:1804.05837*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Sebastian Krause, Leonhard Hennig, Andrea Moro, Dirk Weissenborn, Feiyu Xu, Hans Uszkoreit, and Roberto Navigli. 2016. Sar-graphs: A language resource connecting linguistic knowledge with semantic relations from knowledge graphs. *Web Semantics: Science, Services and Agents on the World Wide Web*, 37:112–131.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1017–1026. Association for Computational Linguistics.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated graph sequence neural networks. *ICLR*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *ACL*.

Zhengdong Lu, Haotian Cui, Xianggen Liu, Yukun Yan, and Daqi Zheng. 2017. Object-oriented neural programming (oonp) for document understanding. *arXiv preprint arXiv:1709.08853*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *EMNLP*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782.

Martin Raison, Pierre-Emmanuel Mazaré, Rajarshi Das, and Antoine Bordes. 2018. Weaver: Deep co-encoding of questions and documents for machine reading. *arXiv preprint arXiv:1804.10490*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Pum-Mo Ryu, Myung-Gil Jang, and Hyun-Ki Kim. 2014. Open domain question answering using wikipedia-based knowledge model. *Information Processing and Management*, 50(5):683 – 692.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ICLR*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

A. Talmor and J. Berant. 2018. The web as a knowledge-base for answering complex questions. In *North American Association for Computational Linguistics (NAACL)*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. *NAACL*.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R^3: Reinforced reader-ranker for open-domain question answering. *AAAI*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017. Evidence aggregation for answer re-ranking in open-domain question answering. *arXiv preprint arXiv:1711.05116*.

Yusuke Watanabe, Bhuwan Dhingra, and Ruslan Salakhutdinov. 2017. Question answering from unstructured text by retrieval and comprehension. *arXiv preprint arXiv:1703.08885*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*.

Georg Wiese, Dirk Weissenborn, and Mariana Neves. 2017. Neural domain adaptation for biomedical question answering. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 281–289, Vancouver, Canada. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 201–206.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *ICLR*.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. *ACL*.

A Nil-Aware Answer Extraction Framework for Question Answering

Souvik Kundu and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
`{souvik, nght}@comp.nus.edu.sg`

Abstract

Recently, there has been a surge of interest in reading comprehension-based (RC) question answering (QA). However, current approaches suffer from an impractical assumption that every question has a valid answer in the associated passage. A practical QA system must possess the ability to determine whether a valid answer exists in a given text passage. In this paper, we focus on developing QA systems that can extract an answer for a question if and only if the associated passage contains an answer. If the associated passage does not contain any valid answer, the QA system will correctly return *Nil*. We propose a nil-aware answer span extraction framework that is capable of returning *Nil* or a text span from the associated passage as an answer in a single step. We show that our proposed framework can be easily integrated with several recently proposed QA models developed for reading comprehension and can be trained in an end-to-end fashion. Our proposed nil-aware answer extraction neural network decomposes pieces of evidence into relevant and irrelevant parts and then combines them to infer the existence of any answer. Experiments on the NewsQA dataset show that the integration of our proposed framework significantly outperforms several strong baseline systems that use pipeline or threshold-based approaches.

1 Introduction

Machine comprehension (MC) systems mimic the process of reading comprehension (RC) by answering questions after understanding natural language text. Several datasets and resources have been developed recently. Richardson et al. (2013) developed a small-scale multiple-choice question answering (QA) dataset. Hermann et al. (2015) created a large cloze-style MC dataset based on CNN and Daily Mail news article summaries. However, Chen et al. (2016) reported that the task is not challenging enough and hence, advanced models had to be evaluated on more realistic datasets. Subsequently, SQuAD (Rajpurkar et al., 2016) was released, where, unlike previous datasets, the answers to different questions can vary in length.

In previous datasets, questions and answers are formulated given text passages. Hence, a valid answer can always be found in the associated passage for every question created. Trischler et al. (2017) proposed a more challenging and realistic dataset, NewsQA, where the questions were formed using CNN article summaries without accessing the original full texts. As such, some questions have no valid answers in the associated passages (referred to as nil questions).

Recently, several neural models for answer span extraction have been proposed (Wang and Jiang, 2017; Seo et al., 2017; Yang et al., 2017; Xiong et al., 2017; Weissenborn et al., 2017; Wang et al., 2017; Shen et al., 2017b; Chen et al., 2017; Kundu and Ng, 2018). However, none of the models considered nil questions, although it is crucial for a practical QA system to be able to determine whether a text passage contains a valid answer for a question. In this paper, we focus on developing QA systems that extract an answer for a question if and only if the associated passage contains a valid answer. Otherwise, they are expected to return *Nil* as answer. We propose a nil-aware answer extraction framework which returns *Nil* or a span of text as answer, when integrated with end-to-end neural MC models. Our proposed framework is based on evidence decomposition-aggregation, where the evidence vectors derived by a higher level encoding layer are first decomposed into relevant and irrelevant components and later aggregated to infer the existence of a valid answer. In addition, we develop several baseline models with pipeline and threshold-based approaches. In a

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4243–4252
Brussels, Belgium, October 31 - November 4, 2018.

pipeline model, detection of nil questions is carried out separately before answer span extraction. In a threshold-based model, the answer span extraction model is entirely trained on questions that have valid answers, and *Nil* is returned based on a confidence threshold.

The contributions of this paper are as follows: (1) We propose a nil-aware answer span extraction framework to return *Nil* or an exact answer span to a question, in a single step, depending on the existence of a valid answer. (2) Our proposed framework can be readily integrated with many recently proposed neural machine comprehension models. In this paper, we extend four machine comprehension models, namely BiDAF (Seo et al., 2017), R-Net (Wang et al., 2017), DrQA (Chen et al., 2017), and AMANDA (Kundu and Ng, 2018), with our proposed framework, and show that they achieve significantly better results compared to the corresponding pipeline and threshold-based models on the NewsQA dataset.

2 Task Definition

Given a passage and a question, we propose models that can extract an answer if and only if the passage contains an answer. When the passage does not contain any answer, the models return *Nil* as the answer. A valid answer is denoted as two pointers in the passage, representing the start and end tokens of the answer span. Let $\mathcal{P}$ be a passage with tokens $(\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_T)$ and $\mathcal{Q}$ be a question with tokens $(\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_U)$, where T and U are the length of the passage and question respectively. A system needs to determine whether the answer is *Nil* or comprises two pointers, b and e, such that $1 \leq b \leq e \leq T$.

3 Proposed Framework

In this section, we first describe our proposed evidence decomposition-aggregation framework for nil-aware answer extraction. Then, we provide a detailed description of how we extend a state-of-the-art model AMANDA (Kundu and Ng, 2018) to NAMANDA[1] (nil-aware AMANDA). We also provide brief descriptions of how we integrate our proposed framework with the other three models.

3.1 Nil-Aware Answer Extraction

Decomposition of lexical semantics over sentences has been successfully used in the past for sentence similarity learning (Wang et al., 2016). Most of the recently proposed machine reading comprehension models can be generalized based on a common pattern observed in their network architecture. They have a question-passage joint encoding layer (also known as question-aware passage encoding layer) followed by an evidence encoding layer. In this work, we decompose the evidence vectors for each passage word obtained from the evidence encoding layer with respect to question-passage joint encoding vectors to derive semantically relevant and irrelevant components. We decompose the evidence vectors for each passage word, because passage vectors can be partially supported by the corresponding question-passage joint encoding vectors, and based on the level of support, it either increases or decreases the chance of finding a valid answer. When we aggregate the orthogonally decomposed evidence vectors, it combines both the supportive and unsupportive pieces of evidence for a particular passage word. To obtain the most impactful portions, we perform a max-pooling operation over all the aggregated vectors. The resulting vector is denoted as the Nil vector. As the training set contains both nil questions (with no valid answers) and non-nil questions (with valid answers), the model automatically learns when to pool unsupportive (for nil questions) and supportive (for non-nil questions) portions to construct the Nil vector. In this way, the model is able to induce a strong bias towards the nil pointer when there is no answer present due to the dominance of unsupportive components in the nil vector.

The proposed method in Wang et al. (2016) was developed for sentence similarity learning tasks, such as answer sentence selection. They decompose an answer sentence with respect to a question and vice versa. The decomposed vectors are then aggregated to obtain a single vector which is used to derive the similarity score. Although our proposed method (developed for the more complex task of answer span extraction) is inspired from the idea of lexical decomposition and composition, one major difference is that we decompose the evidence vectors with respect to question-passage joint encoding vectors. Another important advance is how it is adopted to return nil or a span

---

[1] Our source code is available at `https://github.com/nusnlp/namanda`

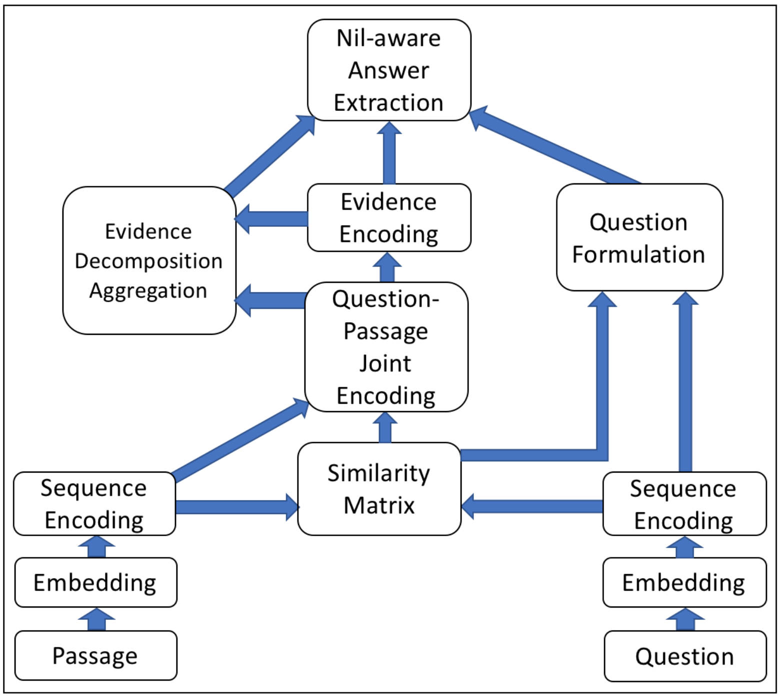

Figure 1: Overview of the architecture of Nil-aware AMANDA (NAMANDA).

of text from the passage in a single step.

3.2 Nil-Aware AMANDA

The architecture of Nil-aware AMANDA (NAMANDA) is given in Figure 1.

3.2.1 Embeddings

To obtain the embeddings, we concatenate word and character-level embedding vectors. We use pre-trained vectors from GloVe (Pennington et al., 2014) for word-level embeddings. For character embeddings, a trainable character-based lookup table is used followed by a convolutional neural network (CNN) and max-pooling (Kim, 2014).

3.2.2 Sequence Encoding

We use bi-directional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997) on the embedding vectors to incorporate contextual information. We represent the outputs as $\mathbf{D} \in \mathbb{R}^{T \times H}$ and $\mathbf{Q} \in \mathbb{R}^{U \times H}$ for passage and question respectively. H is the number of hidden units of the BiLSTMs.

3.2.3 Similarity Matrix

The similarity matrix is obtained by computing the dot product of passage and question sequence-level encoding vectors. The similarity matrix $\mathbf{A} \in \mathbb{R}^{T \times U}$ can be expressed as $\mathbf{A} = \mathbf{D}\ \mathbf{Q}^{\top}$, where $A_{i,j}$ is the similarity between the ith passage word and the jth question word.

3.2.4 Question Formulation

To aggregate the most relevant parts of the question, column-wise maximum values of $\mathbf{A}$ are normalized using a softmax function to obtain $\mathbf{k} \in \mathbb{R}^{U}$. Then, the question vectors in $\mathbf{Q}$ are aggregated by $\mathbf{q}_{ma} = \mathbf{k}\ \mathbf{Q}$. The question type information is incorporated via $\mathbf{q}_f \in \mathbb{R}^{2H}$, by concatenating the sequence-level question encoding vectors of the first wh-word $\mathbf{q}_{t_{wh}}$ and its following word $\mathbf{q}_{t_{wh}+1}$. It can be given as $\mathbf{q}_f = \mathbf{q}_{t_{wh}} \,||\, \mathbf{q}_{t_{wh}+1}$, where $||$ denotes the concatenation operation. The set of wh-words we used is $\{$*what, who, how, when, which, where, why*$\}$. The final question representation, $\tilde{\mathbf{q}} \in \mathbb{R}^{H}$, is formulated by applying a feed-forward neural network on the concatenated representation of $\mathbf{q}_{ma}$ and $\mathbf{q}_f$.

3.2.5 Question-Passage Joint Encoding

In this step, we jointly encode the passage and question. We apply a row-wise softmax function on $\mathbf{A}$ to obtain $\mathbf{R} \in \mathbb{R}^{T \times U}$. Now, for all the passage words, the aggregated question representation $\mathbf{G} \in \mathbb{R}^{T \times H}$ is computed by $\mathbf{G} = \mathbf{R}\ \mathbf{Q}$. The aggregated question vectors corresponding to the passage words are then concatenated with the sequence-level passage vectors to obtain $\mathbf{S} \in \mathbb{R}^{T \times 2H}$. We apply another BiLSTM to obtain a combined representation $\mathbf{V} \in \mathbb{R}^{T \times H}$.

3.2.6 Evidence Decomposition-Aggregation

First, multi-factor self-attentive encoding is applied to accumulate evidence from the entire passage. The use of multiple factors while calculating self attention helps to obtain meaningful information from a long context with fine-grained inference. If m represents the number of factors, multi-factor attention $\mathbf{F}^{[1:m]} \in \mathbb{R}^{T \times m \times T}$ is formulated as:

$$\mathbf{F}^{[1:m]} = \mathbf{V}\, \mathbf{W}_f^{[1:m]}\, \mathbf{V}^\top \ , \tag{1}$$

where $\mathbf{W}_f^{[1:m]} \in \mathbb{R}^{H \times m \times H}$ is a 3-way tensor. Now, to refine the evidence, a max-pooling operation is performed on $\mathbf{F}^{[1:m]}$ over the number of factors, resulting in the self-attention matrix $\mathbf{F} \in \mathbb{R}^{T \times T}$. We normalize $\mathbf{F}$ by applying a row-wise softmax function, resulting in $\tilde{\mathbf{F}} \in \mathbb{R}^{T \times T}$. Now the self-attentive encoding $\mathbf{M} \in \mathbb{R}^{T \times H}$ can be given as $\mathbf{M} = \tilde{\mathbf{F}}\ \mathbf{V}$. The self-attentive encoding vectors are then concatenated with the question-dependent passage word encoding vectors ($\mathbf{V}$), and a feed-forward neural network-based gating is applied to control the overall impact, resulting in $\mathbf{Y} \in \mathbb{R}^{T \times 2H}$.

Then we decompose the evidence vector for every passage word with orthogonal decomposition. Each row of $\mathbf{Y}$, $\mathbf{y}_t \in \mathbb{R}^{2H}$, is decomposed into its parallel and perpendicular components with respect to the corresponding question-passage joint encoding ($\mathbf{S}$) vector, $\mathbf{s}_t \in \mathbb{R}^{2H}$. The parallel components represent the relevant parts of the accumulated evidence and the orthogonal components represent the irrelevant counterparts. If the parallel component of $\mathbf{y}_t$ is represented as $\mathbf{y}_t^{=} \in \mathbb{R}^{2H}$ and the perpendicular component is represented as $\mathbf{y}_t^{\perp} \in \mathbb{R}^{2H}$, then

$$\mathbf{y}_t^{=} = \frac{\mathbf{y}_t\, \mathbf{s}_t^\top}{\mathbf{s}_t\, \mathbf{s}_t^\top}\, \mathbf{s}_t \tag{2}$$

$$\mathbf{y}_t^{\perp} = \mathbf{y}_t - \mathbf{y}_t^{=} \tag{3}$$

Similarly, we derive the parallel and orthogonal vectors for all the passage words. We denote parallel components with $\mathbf{Y}^{=} \in \mathbb{R}^{T \times 2H}$ and perpendicular components with $\mathbf{Y}^{\perp} \in \mathbb{R}^{T \times 2H}$.

In the aggregation step, the parallel and orthogonal components are fed to a feed-forward linear layer. $\mathbf{Y}^a \in \mathbb{R}^{T \times H}$ denotes the output of the linear layer and $\mathbf{y}_t^a \in \mathbb{R}^H$ is its tth row:

$$\mathbf{y}_t^a = \tanh(\mathbf{y}_t^{=}\mathbf{W}_a + \mathbf{y}_t^{\perp}\mathbf{W}_a + \mathbf{b}_a) \ , \tag{4}$$

where $\mathbf{W}_a \in \mathbb{R}^{2H \times H}$ and $\mathbf{b}_a \in \mathbb{R}^H$ are the weight matrix and bias vector respectively. Then we apply a max-pooling operation over all the words to obtain the *Nil* vector representation denoted as $\hat{\mathbf{n}}$. Now we derive the score for the *Nil* pointer which will be shared for normalizing the beginning and ending pointers later. The *Nil* pointer score is given as:

$$n_s = \hat{\mathbf{n}}\mathbf{w}_n^\top \ , \tag{5}$$

where $\mathbf{w}_n \in \mathbb{R}^H$ is a learnable weight vector.

3.2.7 Nil-Aware Pointing

Two stacked BiLSTMs are used on top of $\mathbf{Y}$ to determine the beginning and ending pointers. Let the hidden unit representations of these two BiLSTMs be $\mathbf{B} \in \mathbb{R}^{T \times H}$ and $\mathbf{E} \in \mathbb{R}^{T \times H}$. We measure the similarity scores between the previously derived question vector $\tilde{\mathbf{q}}$ and the contextual encoding vectors in $\mathbf{B}$ and $\mathbf{E}$. If $\mathbf{s}_b \in \mathbb{R}^T$ and $\mathbf{s}_e \in \mathbb{R}^T$ are the scores for the beginning and ending pointers, then

$$\mathbf{s}_b = \tilde{\mathbf{q}}\, \mathbf{B}^\top \ , \ \mathbf{s}_e = \tilde{\mathbf{q}}\, \mathbf{E}^\top \tag{6}$$

We prepend the *nil* score n_s to $\mathbf{s}_b$ and $\mathbf{s}_e$ for shared normalization. The updated scores $\hat{\mathbf{s}}_b \in \mathbb{R}^{T+1}$ and $\hat{\mathbf{s}}_e \in \mathbb{R}^{T+1}$ can be represented as:

$$\hat{\mathbf{s}}_b = [n_s, \mathbf{s}_b] \ , \ \hat{\mathbf{s}}_e = [n_s, \mathbf{s}_e] \tag{7}$$

The beginning and ending pointer probability distributions for a given passage $\mathcal{P}$ and a question $\mathcal{Q}$ is given as:

$$\begin{aligned} \Pr(b \mid \mathcal{P}, \mathcal{Q}) &= \text{softmax}(\hat{\mathbf{s}}_b) \\ \Pr(e \mid \mathcal{P}, \mathcal{Q}) &= \text{softmax}(\hat{\mathbf{s}}_e) \end{aligned} \tag{8}$$

The joint probability distribution for answer a is given as:

$$\Pr(a \mid \mathcal{P}, \mathcal{Q}) = \Pr(b \mid \mathcal{P}, \mathcal{Q})\, \Pr(e \mid \mathcal{P}, \mathcal{Q}) \tag{9}$$

For training, we minimize the cross entropy loss summing over all training instances. During prediction, we select the locations in the passage for which the product of $\Pr(b)$ and $\Pr(e)$ is maximized, where $1 \le b \le e \le T + 1$. If the value of b is 1, we assign the answer as *Nil*.

3.3 Nil-Aware DrQA

We extend DrQA (Chen et al., 2017) to NDrQA by integrating our proposed nil-aware answer extraction framework. In DrQA, the embeddings of passage tokens consist of pretrained word vectors from Glove, several syntactic features, and passage-question joint embedding (aligned question embedding). The syntactic features include exact match of passage words with question in surface, lowercase, and lemma form. They also used part-of-speech tags, named entity tags, and term frequency values for each passage word. Subsequently, a stack of BiLSTMs is used for encoding. The outputs of the stacked BilSTMs are used as evidence vectors to help extract the answer span. We decompose those stacked BiLSTM output vectors with respect to the passage embedding and generate the *nil* pointer score as given in Eqs (2–5). The question vector formulation in DrQA is performed by applying a stack of BilSTMs on question embedding. The nil-aware pointing mechanism is the same as that given in Section 3.2.7 except an additional bi-linear term is used for each $\mathbf{s}_b$ and $\mathbf{s}_e$ in Eq (6).

3.4 Nil-Aware R-Net

In R-Net (Wang et al., 2017), after embedding and encoding of the passage and question words, a gated recurrent network is used to obtain the question-passage joint representation. Subsequently, a self-matching attentive encoding is used to accumulate evidence from the entire passage. In the output layer, an answer recurrent pointer network is used to predict the boundary of an answer span. To extend R-Net to nil-aware R-Net (NR-Net), we decompose the output vectors of the self-matching layer with respect to the question-passage joint encoding vectors, and then aggregate them to obtain the *nil* pointer score as illustrated in Eqs (2–5). In the output layer, we combine the *nil* pointer score to the beginning and ending pointer unnormalized scores, and jointly normalize them using softmax function as given in Eqs (7–8).

3.5 Nil-Aware BiDAF

In BiDAF (Seo et al., 2017), an attention flow layer is used to jointly encode the passage and question. Then, a modeling layer is used to capture the interaction among the question-aware passage vectors. The output of the modeling layer serves as evidence to help extract the answer span in the output layer. To extend the BiDAF model to nil-aware BiDAF (NBiDAF), we decompose the output of the modeling layer with respect to the question-passage joint encoding vectors, and then aggregate them to derive the *nil* pointer score (similar to Eqs (2–5). Similar to the other nil-aware models, we concatenate the *nil* pointer score to the start and end pointer unnormalized scores derived in the output layer, and then jointly normalize them.

4 Baseline Models

For comparison, we propose two types of baseline approaches for nil-aware answer extraction.

4.1 Pipeline Approach

Here, two models are used in a pipeline:

Nil detector: Given a pair of passage and question, a nil detector model determines whether a valid answer is present in the passage.

Answer span extractor: If the nil detector model predicts the presence of a valid answer, the answer span extractor then extracts the answer.

For nil detection, we developed a logistic regression (LR) model with manually defined features and four neural models. For the LR model, we extract four different features which capture the similarity between a passage and a question. Let $\mathcal{P}$ be the passage and $\mathcal{Q}$ be the question (consisting of U' tokens excluding stop words). If $f(\mathcal{P}, \mathcal{Q}_i)$ is the frequency of the ith question word in passage $\mathcal{P}$, then the first feature η is defined as:

$$\eta = \sum_{i=1}^{U'} \log(1 + f(\mathcal{P}, \mathcal{Q}_i)) \qquad (10)$$

The second feature is the same as η, except that the lemma form is considered for both passage and question tokens instead of the surface form. Additionally, we include word overlap count features in both surface and lemma forms.

We also develop several advanced neural network architectures for nil detection. After embedding (the same as Section 3.2.1), we apply sequence-level encoding with either BiLSTM or CNN. For CNN, we use equal numbers of unigram, bigram, and trigram filters and the outputs are concatenated to obtain the final encoding. Next, we apply either global max-pooling (MP) or attentive pooling (AP) over all the sequence vectors to obtain an aggregated vector representation. Let the sequence encoding of a passage be $\mathbf{P}^{nd} \in \mathbb{R}^{T\times H}$, and $\mathbf{p}_t^{nd}$ be the tth row of $\mathbf{P}^{nd}$. The aggregated vector $\tilde{\mathbf{p}}_{nd} \in \mathbb{R}^{H}$ for AP can be obtained as:

$$a_t^{nd} \propto \exp(\mathbf{p}_t^{nd} \mathbf{w}^\top) \tag{11}$$

$$\tilde{\mathbf{p}}_{nd} = \mathbf{a}^{nd} \mathbf{P}^{nd} \ , \tag{12}$$

where $\mathbf{w} \in \mathbb{R}^H$ is a learnable vector. Similarly, we derive the aggregated question vector $\tilde{\mathbf{q}}_{nd}$. For nil detection, we compute the similarity score (s_{nd}) between the aggregated vectors:

$$s_{nd} = \text{sigmoid}(\tilde{\mathbf{p}}_{nd} \ \tilde{\mathbf{q}}_{nd}^\top) \tag{13}$$

We experimented with four state-of-the-art answer span extractor models, namely BiDAF (Seo et al., 2017), R-Net (Wang et al., 2017), DrQA (Chen et al., 2017), and AMANDA (Kundu and Ng, 2018). Note that the answer extraction models are trained entirely on passage-question pairs which always have valid answers.

4.2 Threshold-Based Approach

Here, we do not use any nil questions to train the neural answer span extraction model. This approach assumes that when there is a valid answer, the probability distributions of the beginning and ending pointers will have lower entropy. This results in a higher maximum joint probability of the beginning and ending pointers. In contrast, when an answer is not present in the associated passage, the output probability distributions have higher entropy, resulting in a lower value of maximum joint probability. We set the maximum joint probability *threshold* based on the best Nil F1 score on the nil questions in the development set. Now, for a given test passage and question, we first compute the maximum of all the joint probabilities associated with all the answer spans. Let a_{span} be the answer span with highest joint probability p_{max}. We assign the final *answer* as follows:

$$answer = \begin{cases} Nil, & \text{if } p_{max} \leq threshold \\ a_{span}, & \text{otherwise} \end{cases} \tag{14}$$

5 Experiments

5.1 Experimental Settings

We use the NewsQA dataset with nil questions (Trischler et al., 2017) in our experiments. Its training, development, and test sets consist of 10,938, 638, and 632 passages respectively and every passage is associated with some questions. In each subset, there are some questions which have no answers in the corresponding associated passages (i.e., the nil questions). The detailed statistics of the dataset are given in Table 1.

Dataset		#Passages	#Questions
Train	NewsQA	10,938	92,549
	+Nil Qs		107,673
Dev	NewsQA	638	5,166
	+Nil Qs		5,988
Test	NewsQA	632	5,126
	+Nil Qs		5,971

Table 1: Statistics of the NewsQA dataset.

We compute exact match (EM) and F1 score for questions with valid answers. For questions without any valid answers, we compute Nil precision, recall, and F1 scores as follows:

$$\text{Nil precision} = \frac{\#\text{Correctly predicted Nil}}{\#\text{predicted Nil}} \tag{15}$$

$$\text{Nil recall} = \frac{\#\text{Correctly predicted Nil}}{\#\text{Nil questions}} \tag{16}$$

$$\text{Nil F1} = 2 \times \frac{\text{Nil precision} \times \text{Nil recall}}{\text{Nil precision} + \text{Nil recall}} \tag{17}$$

To compute the overall EM and F1 scores, we consider *Nil* as correct for the questions which do not have any valid answers. All evaluation scores reported in this paper are in %.

All the neural network models are implemented in PyTorch[2]. We use the default hyper-parameters for all the answer span extractor models. We use the open source implementation of DrQA[3]. We use a third party implementation of R-Net[4] whose performance is very similar to the original scores. We reimplemented BiDAF[5] and AMANDA[6] to easily integrate our proposed nil-aware answer extraction framework and make the training faster. We integrate the nil-aware answer span extraction framework with each model keeping all the hyper-parameters unchanged. For nil-detection models, we use the same settings as (N)AMANDA. We use 300 hidden units for BiLSTMs and a total of 300 filters for the CNN-based models. We use dropout (Srivastava et al., 2014) with probability 0.3 for every trainable layer. We use binary cross-entropy loss and the Adam optimizer (Kingma and Ba, 2015) for training the nil-detection models.

---

[2] http://pytorch.org

[3] https://github.com/facebookresearch/DrQA

[4] `https://github.com/HKUST-KnowComp/MnemonicReader/blob/master/r_net.py`

[5] Our implementation gives 3% lower F1 score compared to the reported results in Seo et al. (2017) on the SQuAD development set.

[6] Our implementation gives 0.5% higher F1 score compared to the reported scores in Kundu and Ng (2018) on the NewsQA test set.

Answer Extractor	**Nil Detector**	**Test Set (w/o Nil Questions)**		**Test Set (with Nil Questions)**				
		EM	**F1**	**Nil Precision**	**Nil Recall**	**Nil F1**	**Overall EM**	**Overall F1**
BiDAF	-	42.5	57.5	-	-	-	36.5	49.4
	LR	39.6	53.2	33.1	28.9	30.9	38.1	49.7
	MP-BiLSTM	40.1	54.2	52.5	48.3	50.3	41.3	53.4
	MP-CNN	42.3	57.2	73.8	15.0	24.9	38.4	51.2
	AP-BiLSTM	40.5	54.7	55.3	47.5	51.1	41.5	53.7
	AP-CNN	40.1	54.3	50.8	39.9	44.7	40.1	52.3
NBiDAF		40.8	54.7	48.0	59.6	53.2	43.5	55.4
R-Net	-	49.9	64.0	-	-	-	42.8	54.8
	LR	46.4	58.8	33.1	28.9	30.9	43.9	54.6
	MP-BiLSTM	47.3	60.3	52.5	48.3	50.3	47.5	58.6
	MP-CNN	49.7	63.6	73.8	15.0	24.9	44.8	56.7
	AP-BiLSTM	47.6	60.7	55.3	47.5	51.1	47.6	58.8
	AP-CNN	47.2	60.4	50.8	39.9	44.7	46.2	57.5
NR-Net		47.0	60.8	53.6	57.6	55.5	48.5	60.3
DrQA	-	50.0	64.0	-	-	-	42.9	54.8
	LR	46.3	58.8	33.1	28.9	30.9	43.8	54.6
	MP-BiLSTM	47.1	60.2	52.5	48.3	50.3	47.3	58.5
	MP-CNN	49.6	63.5	73.8	15.0	24.9	44.7	56.7
	AP-BiLSTM	47.4	60.6	55.3	47.5	51.1	47.4	58.8
	AP-CNN	47.0	60.2	50.8	39.9	44.7	46.0	57.3
NDrQA		48.5	61.8	53.5	57.2	55.3	49.8	61.1
AMANDA	-	49.2	64.2	-	-	-	42.2	55.1
	LR	45.4	58.8	33.1	28.9	30.9	43.1	54.5
	MP-BiLSTM	46.2	60.2	52.5	48.3	50.3	46.5	58.5
	MP-CNN	48.3	63.3	73.8	15.0	24.9	43.6	56.5
	AP-BiLSTM	46.3	60.6	55.3	47.5	51.1	46.5	58.7
	AP-CNN	45.9	60.0	50.8	39.9	44.7	45.1	57.1
NAMANDA		48.6	62.2	57.1	56.7	56.9	49.7	61.5

Table 2: Performance Comparison with pipeline approaches on the NewsQA test set.

5.2 Results

Tables 2 and 3 compare results of the nil-aware answer span extractor models with several pipeline and threshold-based models, respectively. We also include the results of four standalone answer span extraction models on the test set without nil questions. Table 2 shows that the end-to-end nil-aware models achieve the highest overall EM and F1 scores compared to all the corresponding pipeline systems. Note that the MP-BiLSTM nil detection model achieves higher Nil F1 scores compared to LR and MP-CNN. This is because BiLSTM is able to capture long-range contextual information to infer the existence of valid answers. Furthermore, AP-based models perform better compared to MP-based models as the attention mechanism used in AP-based models inherently identifies important contextual information. Due to this, the performance gap between AP-CNN and AP-BiLSTM is lower than the performance gap between MP-CNN and MP-BiLSTM. In addition to achieving higher Nil F1 score than the strong nil detection baseline systems, nil-aware models manage to achieve competitive scores compared to the corresponding standalone answer span extractors on the test set where there are no nil questions.

Table 3 shows that the nil-aware models outperform the corresponding threshold-based models. Note that all four answer span extraction models, when used in a threshold-based approach for nil detection, produce low Nil precision and relatively higher Nil recall. The low precision significantly degrades performance on the test set without nil questions. These models often return *Nil* since it is critical to find suitable values for the required *threshold*. This is because NewsQA passages are often very long and as a result, probability distributions with higher entropy for answer pointer selection lead to irregular maximum joint probability *threshold* values.

We perform statistical significance tests using paired t-test and bootstrap resampling. Performances of all the nil-aware models (in terms of overall EM and F1) are significantly better ($p < 0.01$) than the corresponding best pipeline models and threshold-based approaches.

Answer Extractor	Nil Answer Handling	Test Set (w/o Nil Questions)		Test Set (with Nil Questions)				
		EM	F1	Nil Precision	Nil Recall	Nil F1	Overall EM	Overall F1
BiDAF	No	42.5	57.5	-	-	-	36.5	49.4
	Yes	37.9	48.3	25.8	60.2	36.1	41.0	50.0
NBiDAF		40.8	54.7	48.0	59.6	53.2	43.5	55.4
R-Net	No	49.9	64.0	-	-	-	42.8	54.8
	Yes	45.3	54.7	25.5	53.6	34.6	46.5	54.5
NR-Net		47.0	60.8	53.6	57.6	55.5	48.5	60.3
DrQA	No	50.0	64.0	-	-	-	42.9	54.8
	Yes	40.8	48.1	23.1	68.0	34.5	44.6	51.0
NDrQA		48.5	61.8	53.5	57.2	55.3	49.8	61.1
AMANDA	No	49.2	64.2	-	-	-	42.2	55.1
	Yes	42.2	51.3	24.0	59.5	34.2	44.6	52.5
NAMANDA		48.6	62.2	57.1	56.7	56.9	49.7	61.5

Table 3: Performance comparison with threshold-based approaches on the NewsQA test set.

5.3 Analysis

For better understanding, we present further experiments and analysis of one of the proposed models, NAMANDA.

In addition to linear aggregation, we experiment with BiLSTM-based and CNN-based aggregation models. When we use BiLSTM aggregation, Eq. (4) is modified to $\mathbf{y}_t^a = \mathbf{h}_t^{=} + \mathbf{h}_t^{\perp}$, where

$$\begin{aligned} \mathbf{h}_t^{=} &= \text{BiLSTM}(\mathbf{y}_t^{=}, \mathbf{h}_{t-1}^{=}, \mathbf{h}_{t+1}^{=}) \quad (18) \\ \mathbf{h}_t^{\perp} &= \text{BiLSTM}(\mathbf{y}_t^{\perp}, \mathbf{h}_{t-1}^{\perp}, \mathbf{h}_{t+1}^{\perp}) \end{aligned}$$

We use equal numbers of unigram, bigram, and trigram filters for CNN-based aggregation. Similar to BiLSTM-based aggregation, we add the CNN outputs for $\mathbf{Y}^{=}$ and $\mathbf{Y}^{\perp}$. Table 4 shows that linear aggregation achieves the highest overall F1 score despite using the least number of parameters.

Table 5 shows the results of NAMANDA on the NewsQA development set when different components are removed such as character embeddings, question-passage joint encoding, and the second LSTM for the answer-ending pointer. When question-passage joint encoding is removed, self-attentive encoding is formed as well as decomposed with respect to sequence-level passage encoding. When we remove the second LSTM for the answer-ending pointer, a feed-forward network is used instead. It is clear from Table 5 that question-passage joint encoding has the highest impact.

Aggregation Model	w/o Nil	with Nil	
	EM (F1)	Nil Prec/ Rec (F1)	Overall EM (F1)
BiLSTM	47.6 (60.9)	56.5/53.5 (55.0)	48.4 (59.9)
CNN	46.2 (60.0)	52.7/54.5 (53.6)	47.3 (59.2)
Linear (NAMANDA)	47.8 (60.5)	51.2/57.2 (54.0)	49.1 (60.0)

Table 4: Effect of different aggregation models on the NewsQA dev set.

Model	w/o Nil	with Nil	
	EM (F1)	Nil Prec/ Rec (F1)	Overall EM (F1)
– character embeddings	46.3 (59.2)	51.9/54.1 (53.0)	47.4 (58.5)
– q-passage joint encoding	32.5 (43.9)	41.4/58.8 (48.6)	36.1 (45.9)
– second LSTM	47.6 (60.3)	56.7/51.0 (53.7)	48.0 (59.0)
NAMANDA	47.8 (60.5)	51.2/57.2 (54.0)	49.1 (60.0)

Table 5: Ablation studies on the NewsQA dev set.

Figure 2(a) and Figure 2(b) show the results of NAMANDA on different question (excluding the stop words) and passage lengths respectively on the NewsQA development set. With increasing question length, the Nil F1 score also improves. This is because with more information in a question, it becomes easier to detect whether the associated passage contains a valid answer. Increasing Nil F1 scores also help to improve the overall F1 scores. However, the overall F1 score degrades with increasing length of the associated passage. When the associated passage is long, it is difficult for the answer span extractor to extract an answer for a question which has a valid answer, due to the increasing amount of potentially distracting information. The Nil F1 scores remain similar for passages consisting of not more than 1,200 tokens. Beyond that, the Nil F1 score degrades a little

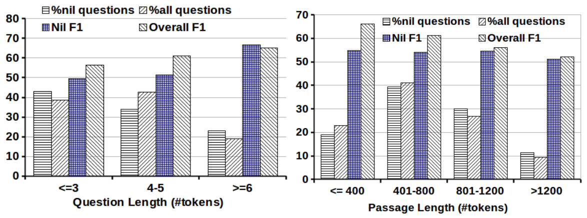

(a) (b)

Figure 2: Results for different (a) question and (b) passage lengths on NewsQA dev set.

as it becomes very challenging to infer the existence of a valid answer accurately with increasing amount of potentially distracting information present in the passage.

Nil detection is itself a very challenging task. Performances of the nil-aware models are worse than the corresponding answer extractor models on the test set without nil questions as Nil precision is lower than 100%. We carried out an experiment to evaluate the performance of NAMANDA on development sets with varying number of nil questions. As the proportion of nil questions in a set increases, NAMANDA outperforms AMANDA by a larger margin on overall scores.

6 Related Work

In some years of the question answering track at the Text Retrieval Conference (TREC)[7], some questions were considered as nil questions for which no valid answers could be found in the entire corpus. Participating teams were required to return *Nil* as answer for those questions. Many teams used threshold-based methods to determine whether any of the retrieved answers for a given question was valid or not. If none of the answers had high confidence, *Nil* was returned as answer. To evaluate the performance on the nil questions, TREC used Nil precision, recall and F1 scores.

In recent years, research on question answering has witnessed substantial progress with rapid advances in neural network architectures. For example, on the answer sentence selection task, where a system has to choose the correct answer sentence from a pool of candidate sentences for a given question, the introduction of attention-based neural models has rapidly advanced the state of the art (Tan et al., 2015; Yang et al., 2016; dos Santos et al., 2016; Wang et al., 2016; Bian et al., 2017; Shen et al., 2017a).

However, in the answer sentence selection task, the answer is always a full sentence. Rajpurkar et al. (2016) released a reading comprehension-based QA dataset SQuAD, where given a passage and a question, a system needs to find the exact answer span rather than a sentence. Although SQuAD became very popular and served as a good test set to develop advanced end-to-end neural network architectures, it does not include any nil questions. In practical QA, it is critical to decide whether or not a passage contains a valid answer for a given question. Subsequently, the NewsQA (Trischler et al., 2017) dataset has been released which attempts to overcome this deficiency. However, all the proposed models for NewsQA so far have excluded nil questions during evaluation. Contrary to prior work, we focus on developing models for nil-aware answer span extraction. Very recently, Rajpurkar et al. (2018) released the SQUADRUN dataset by augmenting the SQuAD dataset with unanswerable questions. The unanswerable questions are written adversarially by crowdworkers to look similar to the answerable ones.

7 Conclusion

In this paper, we have focused on nil-aware answer span extraction for RC-based QA. A nil-aware QA system only extracts a span of text from the associated passage as an answer to a given question if and only if the passage contains a valid answer. We have proposed a nil-aware answer span extraction framework based on evidence decomposition and aggregation that can be easily integrated with several recently proposed neural answer span extraction models. We have also developed several pipeline and threshold-based models using advanced neural architectures for comparison. Experiments on the NewsQA dataset show that our proposed framework, when integrated with the answer span extraction models, achieves better performance compared to all the corresponding pipeline and threshold-based models. Employing such a nil-aware answer span extractor in practical IR-style QA tasks will be interesting future work.

References

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model

---

[7] `https://trec.nist.gov/data/qa.html`

with dynamic-clip attention for answer selection. In *Proceedings of CIKM*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN / Daily Mail reading comprehension task. In *Proceedings of ACL*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of ACL*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in NIPS*.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Souvik Kundu and Hwee Tou Ng. 2018. A question-focused multi-factor attention network for question answering. In *Proceedings of AAAI*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of EMNLP*.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR*.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017a. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of EMNLP*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017b. ReasoNet: Learning to stop reading in machine comprehension. In *Proceedings of KDD*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. LSTM-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, ACL*.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using Match-LSTM and answer pointer. In *Proceedings of ICLR*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of ACL*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Proceedings of CoNLL*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *Proceedings of ICLR*.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. 2016. aNMM: Ranking short answer texts with attention-based neural matching model. In *Proceedings of CIKM*.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. 2017. Words or characters? Fine-grained gating for reading comprehension. In *Proceedings of ICLR*.

Exploiting Deep Representations for Neural Machine Translation

Zi-Yi Dou
Nanjing University
`douzy@smail.nju.edu.cn`

Zhaopeng Tu[*]
Tencent AI Lab
`zptu@tencent.com`

Xing Wang
Tencent AI Lab
`brightxwang@tencent.com`

Shuming Shi
Tencent AI Lab
`shumingshi@tencent.com`

Tong Zhang
Tencent AI Lab
`bradymzhang@tencent.com`

Abstract

Advanced neural machine translation (NMT) models generally implement encoder and decoder as multiple layers, which allows systems to model complex functions and capture complicated linguistic structures. However, only the top layers of encoder and decoder are leveraged in the subsequent process, which misses the opportunity to exploit the useful information embedded in other layers. In this work, we propose to simultaneously expose all of these signals with layer aggregation and multi-layer attention mechanisms. In addition, we introduce an auxiliary regularization term to encourage different layers to capture diverse information. Experimental results on widely-used WMT14 English⇒German and WMT17 Chinese⇒English translation data demonstrate the effectiveness and universality of the proposed approach.

1 Introduction

Neural machine translation (NMT) models have advanced the machine translation community in recent years (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014). NMT models generally consist of two components: an *encoder* network to summarize the input sentence into sequential representations, based on which a *decoder* network generates target sentence word by word with an attention model (Bahdanau et al., 2015; Luong et al., 2015).

Nowadays, advanced NMT models generally implement encoder and decoder as multiple layers, regardless of the specific model architectures such as RNN (Zhou et al., 2016; Wu et al., 2016), CNN (Gehring et al., 2017), or Self-Attention Network (Vaswani et al., 2017; Chen et al., 2018). Several researchers have revealed that different layers are able to capture different types of syntax and semantic information (Shi et al., 2016; Peters et al., 2018; Anastasopoulos and Chiang, 2018). For example, Shi et al. (2016) find that both local and global source syntax are learned by the NMT encoder and different types of syntax are captured at different layers.

However, current NMT models only leverage the top layers of encoder and decoder in the subsequent process, which misses the opportunity to exploit useful information embedded in other layers. Recently, aggregating layers to better fuse semantic and spatial information has proven to be of profound value in computer vision tasks (Huang et al., 2017; Yu et al., 2018). In natural language processing community, Peters et al. (2018) have proven that simultaneously exposing all layer representations outperforms methods that utilize just the top layer for transfer learning tasks.

Inspired by these findings, we propose to exploit deep representations for NMT models. Specifically, we investigate two types of strategies to better fuse information across layers, ranging from layer aggregation to multi-layer attention. While layer aggregation strategies combine hidden states at the same position across different layers, multi-layer attention allows the model to combine information in different positions. In addition, we introduce an auxiliary objective to encourage different layers to capture diverse information, which we believe would make the deep representations more meaningful.

We evaluated our approach on two widely-used WMT14 English⇒German and WMT17 Chinese⇒English translation tasks. We employed TRANSFORMER (Vaswani et al., 2017) as the baseline system since it has proven to outperform other architectures on the two tasks (Vaswani et al., 2017; Hassan et al., 2018). Experimen-

---

[*] Zhaopeng Tu is the corresponding author of the paper. This work was conducted when Zi-Yi Dou was interning at Tencent AI Lab.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4253–4262
Brussels, Belgium, October 31 - November 4, 2018.

tal results show that exploiting deep representations consistently improves translation performance over the vanilla TRANSFORMER model across language pairs. It is worth mentioning that TRANSFORMER-BASE with deep representations exploitation outperforms the vanilla TRANSFORMER-BIG model with only less than half of the parameters.

2 Background: Deep NMT

Deep representations have proven to be of profound value in machine translation (Meng et al., 2016; Zhou et al., 2016). Multiple-layer encoder and decoder are employed to perform the translation task through a series of nonlinear transformations from the representation of input sequences to final output sequences. The layer can be implemented as RNN (Wu et al., 2016), CNN (Gehring et al., 2017), or Self-Attention Network (Vaswani et al., 2017). In this work, we take the advanced Transformer as an example, which will be used in experiments later. However, we note that the proposed approach is generally applicable to any other type of NMT architectures.

Specifically, the encoder is composed of a stack of L identical layers, each of which has two sub-layers. The first sub-layer is a self-attention network, and the second one is a position-wise fully connected feed-forward network. A residual connection (He et al., 2016) is employed around each of the two sub-layers, followed by layer normalization (Ba et al., 2016). Formally, the output of the first sub-layer $\mathbf{C}_e^l$ and the second sub-layer $\mathbf{H}_e^l$ are calculated as

$$\begin{aligned}\mathbf{C}_e^l &= \textsc{Ln}\big(\textsc{Att}(\mathbf{Q}_e^l, \mathbf{K}_e^{l-1}, \mathbf{V}_e^{l-1}) + \mathbf{H}_e^{l-1}\big),\\ \mathbf{H}_e^l &= \textsc{Ln}\big(\textsc{Ffn}(\mathbf{C}_e^l) + \mathbf{C}_e^l\big),\end{aligned} \qquad (1)$$

where $\textsc{Att}(\cdot)$, $\textsc{Ln}(\cdot)$, and $\textsc{Ffn}(\cdot)$ are self-attention mechanism, layer normalization, and feed-forward networks with ReLU activation in between, respectively. $\{\mathbf{Q}_e^l, \mathbf{K}_e^{l-1}, \mathbf{V}_e^{l-1}\}$ are query, key and value vectors that are transformed from the (*l-1*)-th encoder layer $\mathbf{H}_e^{l-1}$.

The decoder is also composed of a stack of L identical layers. In addition to two sub-layers in each decoder layer, the decoder inserts a third sublayer $\mathbf{D}_d^l$ to perform attention over the output of the encoder stack $\mathbf{H}_e^L$:

$$\begin{aligned}\mathbf{C}_d^l &= \textsc{Ln}\big(\textsc{Att}(\mathbf{Q}_d^l, \mathbf{K}_d^{l-1}, \mathbf{V}_d^{l-1}) + \mathbf{H}_d^{l-1}\big),\\ \mathbf{D}_d^l &= \textsc{Ln}\big(\textsc{Att}(\mathbf{C}_d^l, \mathbf{K}_e^{L}, \mathbf{V}_e^{L}) + \mathbf{C}_d^{l}\big),\\ \mathbf{H}_d^l &= \textsc{Ln}\big(\textsc{Ffn}(\mathbf{D}_d^l) + \mathbf{D}_d^l\big),\end{aligned} \qquad (2)$$

where $\{\mathbf{Q}_d^l, \mathbf{K}_d^{l-1}, \mathbf{V}_d^{l-1}\}$ are transformed from the (*l-1*)-th decoder layer $\mathbf{H}_d^{l-1}$, and $\{\mathbf{K}_e^L, \mathbf{V}_e^L\}$ are transformed from the top layer of the encoder. The top layer of the decoder $\mathbf{H}_d^L$ is used to generate the final output sequence.

Multi-layer network can be considered as a strong feature extractor with extended receptive fields capable of linking salient features from the entire sequence (Chen et al., 2018). However, one potential problem about the vanilla Transformer, as shown in Figure 1a, is that both the encoder and decoder stack layers in sequence and only utilize the information in the top layer. While studies have shown deeper layers extract more semantic and more global features (Zeiler and Fergus, 2014; Peters et al., 2018), these do not prove that the last layer is the ultimate representation for any task. Although residual connections have been incorporated to combine layers, these connections have been "shallow" themselves, and only fuse by simple, one-step operations (Yu et al., 2018). We investigate here how to better fuse information across layers for NMT models.

In the following sections, we simplify the equations to $\mathbf{H}^l = \textsc{Layer}(\mathbf{H}^{l-1})$ for brevity.

3 Proposed Approaches

In this section, we first introduce how to exploit deep representations by simultaneously exposing all of the signals from all layers (Sec 3.1). Then, to explicitly encourage different layers to incorporate various information, we propose one way to measure the diversity between layers and add a regularization term to our objective function to maximize the diversity across layers (Sec 3.2).

3.1 Deep Representations

To exploit deep representations, we investigate two types of strategies to fuse information across layers, from layer aggregation to multi-layer attention. While layer aggregation strategies combine hidden states at the same position across different layers, multi-layer attention allows the model to combine information in different positions.

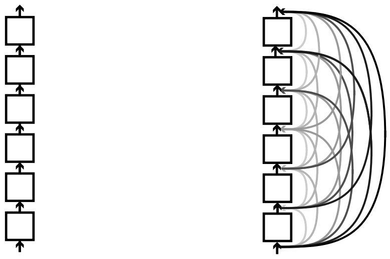

(a) Vanilla (b) Dense

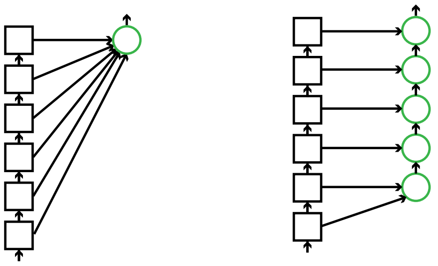

(c) Linear (d) Iterative

Figure 1: Illustration of (a) vanilla model without any aggregation and (b,c,d) layer aggregation strategies. Aggregation nodes are represented by green circles.

3.1.1 Layer Aggregation

While the aggregation strategies are inspired by previous work, there are several differences since we have simplified and generalized from the original model, as described below.

Dense Connection. The first strategy is to allow all layers to directly access previous layers:

$$\mathbf{H}^l = f(\mathbf{H}^1, \ldots, \mathbf{H}^{l-1}). \tag{3}$$

In this work, we mainly investigate whether densely connected networks work for NMT, which have proven successful in computer vision tasks (Huang et al., 2017). The basic strategy of densely connected networks is to connect each layer to every previous layer with a residual connection:

$$\mathbf{H}^l = \text{Layer}(\mathbf{H}^{l-1}) + \sum_{i=1}^{l-1} \mathbf{H}^i. \tag{4}$$

Figure 1b illustrates the idea of this approach. Our implementation differs from (Huang et al., 2017) in that we use an addition instead of a concatenation operation in order to keep the state size constant. Another reason is that concatenation operation is computationally expensive, while residual connections are more efficient.

While dense connection directly feeds previous layers to the subsequent layers, the following mechanisms maintain additional layers to aggregate standard layers, from shallow linear combination, to deep non-linear aggregation.

Linear Combination. As shown in Figure 1c, an intuitive strategy is to linearly combine the outputs of all layers:

$$\widehat{\mathbf{H}} = \sum_{l=1}^{L} \mathbf{W}_l \mathbf{H}^l, \tag{5}$$

where $\{\mathbf{W}_1, \ldots, \mathbf{W}_L\}$ are trainable matrices. While the strategy is similar in spirit to (Peters et al., 2018), there are two main differences: (1) they use normalized weights while we directly use parameters that could be either positive or negative numbers, which may benefit from more modeling flexibility. (2) they use a scalar that is shared by all elements in the layer states, while we use learnable matrices. The latter offers a more precise control of the combination by allowing the model to be more expressive than scalars (Tu et al., 2017).

We also investigate strategies that *iteratively* and *hierarchically* merge layers by incorporating more depth and sharing, which have proven effective for computer vision tasks (Yu et al., 2018).

Iterative Aggregation. As illustrated in Figure 1d, iterative aggregation follows the iterated stacking of the backbone architecture. Aggregation begins at the shallowest, smallest scale and then iteratively merges deeper, larger scales. The iterative deep aggregation function I for a series of layers $\mathbf{H}_1^l = \{\mathbf{H}^1, \cdots, \mathbf{H}^l\}$ with increasingly deeper and semantic information is formulated as

$$\widehat{\mathbf{H}}^l = I(\mathbf{H}_1^l) = \textsc{Agg}(\mathbf{H}^l, \widehat{\mathbf{H}}^{l-1}), \tag{6}$$

where we set $\widehat{\mathbf{H}}^1 = \mathbf{H}^1$ and $\textsc{Agg}(\cdot, \cdot)$ is the aggregation function:

$$\textsc{Agg}(x, y) = \textsc{Ln}(\textsc{Ff}([x; y]) + x + y). \tag{7}$$

As seen, in this work, we first concatenate x and y into $z = [x; y]$, which is subsequently fed to a feed-forward network with a sigmoid activation in between. Residual connection and layer normalization are also employed. Specifically, both x and y have residual connections to the output. The choice of the aggregation function will be further studied in the experiment section.

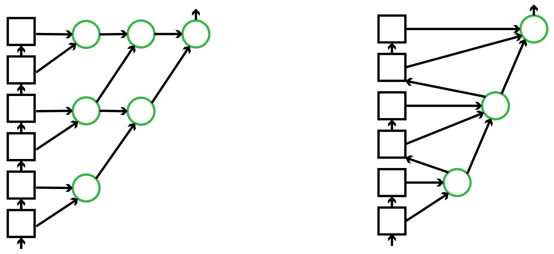

(a) CNN-like tree (b) Hierarchical

Figure 2: Hierarchical aggregation (b) that aggregates layers through a tree structure (a).

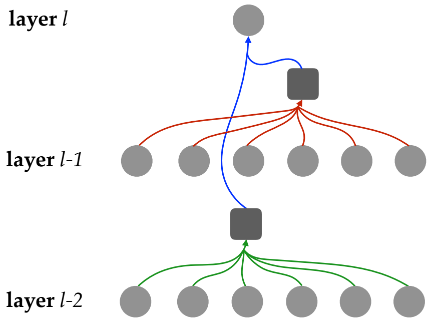

Figure 3: Multi-layer attention allows the model to attend multiple layers to construct each hidden state. We use two-layer attention for illustration, while the approach is applicable to any layers lower than l.

Hierarchical Aggregation. While iterative aggregation deeply combines states, it may still be insufficient to fuse the layers for its sequential architecture. Hierarchical aggregation, on the other hand, merges layers through a tree structure to preserve and combine feature channels, as shown in Figure 2. The original model proposed by Yu et al. (2018) requires the number of layers to be the power of two, which limits the applicability of these methods to a broader range of NMT architectures (e.g. six layers in (Vaswani et al., 2017)). To solve this problem, we introduce a CNN-like tree with the filter size being two, as shown in Figure 2a. Following (Yu et al., 2018), we first merge aggregation nodes of the same depth for efficiency so that there would be at most one aggregation node for each depth. Then, we further feed the output of an aggregation node back into the backbone as the input to the next sub-tree, instead of only routing intermediate aggregations further up the tree, as shown in Figure 2b. The interaction between aggregation and backbone nodes allows the model to better preserve features.

Formally, each aggregation node $\widehat{\mathbf{H}}^i$ is calculated as

$$\widehat{\mathbf{H}}^i = \begin{cases} \textsc{Agg}(\mathbf{H}^{2i-1}, \mathbf{H}^{2i}), & i = 1 \\ \textsc{Agg}(\mathbf{H}^{2i-1}, \mathbf{H}^{2i}, \widehat{\mathbf{H}}^{i-1}), & i = 2, 3 \end{cases}$$

where $\textsc{Agg}(\mathbf{H}^{2i-1}, \mathbf{H}^{2i})$ is computed via Eqn. 7, and $\textsc{Agg}(\mathbf{H}^{2i-1}, \mathbf{H}^{2i}, \widehat{\mathbf{H}}^{i-1})$ is computed as

$$\textsc{Agg}(x, y, z) = \textsc{Ln}(\textsc{Ff}([x; y; z]) + x + y + z).$$

The aggregation node at the top layer $\widehat{\mathbf{H}}^{L/2}$ serves as the final output of the network.

3.1.2 Multi-Layer Attention

Partially inspired by Meng et al. (2016), we also propose to introduce a multi-layer attention mechanism into deep NMT models, for more power of transforming information across layers. In other words, for constructing each hidden state in any layer-l, we allow the self-attention model to attend any layers lower than l, instead of just layer *l-1*:

$$\begin{aligned} \mathbf{C}^l_{-1} &= \textsc{Att}(\mathbf{Q}^l, \mathbf{K}^{l-1}, \mathbf{V}^{l-1}), \\ \mathbf{C}^l_{-2} &= \textsc{Att}(\mathbf{Q}^l, \mathbf{K}^{l-2}, \mathbf{V}^{l-2}), \\ &\dots \\ \mathbf{C}^l_{-k} &= \textsc{Att}(\mathbf{Q}^l, \mathbf{K}^{l-k}, \mathbf{V}^{l-k}), \\ \mathbf{C}^l &= \textsc{Agg}(\mathbf{C}^l_{-1}, \dots, \mathbf{C}^l_{-k}), \end{aligned} \tag{8}$$

where $\mathbf{C}^l_{-i}$ is sequential vectors queried from layer *l-i* using a separate attention model, and $\textsc{Agg}(\cdot)$ is similar to the pre-defined aggregation function to transform k vectors $\{\mathbf{C}^l_{-1}, \dots, \mathbf{C}^l_{-k}\}$ to a d-dimension vector, which is subsequently used to construct the encoder and decoder layers via Eqn. 1 and 2 respectively. Note that multi-layer attention only modifies the self-attention blocks in both encoder and decoder, while does not revises the encoder-decoder attention blocks.

3.2 Layer Diversity

Intuitively, combining layers would be more meaningful if different layers are able to capture diverse information. Therefore, we explicitly add a *regularization* term to encourage the diversities between layers:

$$\mathcal{L} = \mathcal{L}_{\text{likelihood}} + \lambda \mathcal{L}_{\text{diversity}}, \tag{9}$$

where λ is a hyper-parameter and is set to 1.0 in this paper. Specifically, the regularization term measures the average of the distance between every two adjacent layers:

$$\mathcal{L}_{diversity} = \frac{1}{L-1} \sum_{l=1}^{L-1} D(\mathbf{H}^l, \mathbf{H}^{l+1}). \qquad (10)$$

Here $D(\mathbf{H}^l, \mathbf{H}^{l+1})$ is the averaged *cosine-squared distance* between the states in layers $\mathbf{H}^l = \{\mathbf{h}_1^l, \ldots, \mathbf{h}_N^l\}$ and $\mathbf{H}^{l+1} = \{\mathbf{h}_1^{l+1}, \ldots, \mathbf{h}_N^{l+1}\}$:

$$D(\mathbf{H}^l, \mathbf{H}^{l+1}) = \frac{1}{N} \sum_{n=1}^{N} (1 - \cos^2(\mathbf{h}_n^l, \mathbf{h}_n^{l+1})).$$

The cosine-squared distance between two vectors is maximized when two vectors are linearly independent and minimized when two vectors are linearly dependent, which satisfies our goal.[1]

4 Experiments

4.1 Setup

Dataset. To compare with the results reported by previous work (Gehring et al., 2017; Vaswani et al., 2017; Hassan et al., 2018), we conducted experiments on both Chinese⇒English (Zh⇒En) and English⇒German (En⇒De) translation tasks. For the Zh⇒En task, we used all of the available parallel data with maximum length limited to 50, consisting of about 20.62 million sentence pairs. We used newsdev2017 as the development set and newstest2017 as the test set. For the En⇒De task, we trained on the widely-used WMT14 dataset consisting of about 4.56 million sentence pairs. We used newstest2013 as the development set and newstest2014 as the test set. Byte-pair encoding (BPE) was employed to alleviate the Out-of-Vocabulary problem (Sennrich et al., 2016) with 32K merge operations for both language pairs. We used 4-gram NIST BLEU score (Papineni et al., 2002) as the evaluation metric, and *sign-test* (Collins et al., 2005) to test for statistical significance.

Models. We evaluated the proposed approaches on advanced Transformer model (Vaswani et al., 2017), and implemented on top of an open-source toolkit – THUMT (Zhang et al., 2017). We followed Vaswani et al. (2017) to set the configurations and train the models, and have reproduced their reported results on the En⇒De task. The parameters of the proposed models were initialized by the pre-trained model. We tried $k = 2$ and $k = 3$ for the multi-layer attention model, which allows to attend to the lower two or three layers.

We have tested both *Base* and *Big* models, which differ at hidden size (512 vs. 1024), filter size (2048 vs. 4096) and the number of attention heads (8 vs. 16).[2] All the models were trained on eight NVIDIA P40 GPUs where each was allocated with a batch size of 4096 tokens. In consideration of computation cost, we studied model variations with *Base* model on En⇒De task, and evaluated overall performance with *Big* model on both Zh⇒En and En⇒De tasks.

4.2 Results

Table 1 shows the results on WMT14 En⇒De translation task. As seen, the proposed approaches improve the translation quality in all cases, although there are still considerable differences among different variations.

Model Complexity Except for dense connection, all other deep representation strategies introduce new parameters, ranging from 14.7M to 33.6M. Accordingly, the training speed decreases due to more efforts to train the new parameters. Layer aggregation mechanisms only marginally decrease decoding speed, while multi-layer attention decreases decoding speed by 21% due to an additional attention process for each layer.

Layer Aggregation (Rows 2-5): Although dense connection and linear combination only marginally improve translation performance, iterative and hierarchical aggregation strategies achieve more significant improvements, which are up to +0.99 BLEU points better than the baseline model. This indicates that deep aggregations outperform their shallow counterparts by incorporating more depth and sharing, which is consistent with the results in computer vision tasks (Yu et al., 2018).

Multi-Layer Attention (Rows 6-7): Benefiting from the power of attention models, multi-layer attention model can also significantly outperform baseline, although it only attends to one or two additional layers. However, increasing the number of lower layers to be attended from $k = 2$ to

---

[1] We use cosine-squared distance instead of cosine distance, since the latter is maximized when two vectors are in opposite directions. In such case, the two vectors are in fact linearly dependent, while we aim at encouraging the vectors independent from each other.

[2] Here “filter size” refers to the hidden size of the feed-forward network in the Transformer model.

#	Model	# Para.	Train	Decode	BLEU	△
1	Transformer-Base	88.0M	1.82	1.33	27.64	–
2	+ Dense Connection	+0.0M	1.69	1.28	27.94	+0.30
3	+ Linear Combination	+14.7M	1.59	1.26	28.09	+0.45
4	+ Iterative Aggregation	+31.5M	1.32	1.22	28.61	+0.97
5	+ Hierarchical Aggregation	+23.1M	1.46	1.25	28.63	+0.99
6	+ Multi-Layer Attention (k=2)	+33.6M	1.19	1.05	28.58	+0.94
7	+ Multi-Layer Attention (k=3)	+37.8M	1.12	1.00	28.62	+0.98
8	+ Iterative Aggregation + $\mathcal{L}_{diversity}$	+31.5M	1.28	1.22	28.76	+1.12
9	+ Hierarchical Aggregation + $\mathcal{L}_{diversity}$	+23.1M	1.41	1.25	28.78	+1.14
10	+ Multi-Layer Attention (k=2)+ $\mathcal{L}_{diversity}$	+33.6M	1.12	1.05	28.75	+1.11

Table 1: Evaluation of translation performance on WMT14 English⇒German ("En⇒De") translation task. "# Para." denotes the number of parameters, and "Train" and "Decode" respectively denote the training speed (steps/second) and decoding speed (sentences/second) on Tesla P40.

System	Architecture	En⇒De		Zh⇒En	
		# Para.	BLEU	# Para.	BLEU
Existing NMT systems					
(Wu et al., 2016)	RNN with 8 layers	N/A	26.30	N/A	N/A
(Gehring et al., 2017)	CNN with 15 layers	N/A	26.36	N/A	N/A
(Vaswani et al., 2017)	Transformer-Base	65M	27.3	N/A	N/A
	Transformer-Big	213M	28.4	N/A	N/A
(Hassan et al., 2018)	Transformer-Big	N/A	N/A	N/A	24.2
Our NMT systems					
this work	Transformer-Base	88M	27.64	108M	24.13
	+ Deep Representations	111M	28.78†	131M	24.76†
	Transformer-Big	264M	28.58	304M	24.56
	+ Deep Representations	356M	29.21†	396M	25.10†

Table 2: Comparing with existing NMT systems on WMT14 English⇒German and WMT17 Chinese⇒English tasks. "+ Deep Representations" denotes "+ Hierarchical Aggregation + $\mathcal{L}_{diversity}$". "†" indicates statistically significant difference ($p < 0.01$) from the Transformer baseline.

$k = 3$ only gains marginal improvement, at the cost of slower training and decoding speeds. In the following experiments, we set set $k = 2$ for the multi-layer attention model.

Layer Diversity (Rows 8-10): The introduced diversity regularization consistently improves performance in all cases by encouraging different layers to capture diverse information. Our best model outperforms the vanilla Transformer by +1.14 BLEU points. In the following experiments, we used hierarchical aggregation with diversity regularization (Row 8) as the default strategy.

Main Results Table 2 lists the results on both WMT17 Zh⇒En and WMT14 En⇒De translation tasks. As seen, exploiting deep representations consistently improves translation performance across model variations and language pairs, demonstrating the effectiveness and universality of the proposed approach. It is worth mentioning that Transformer-Base with deep representations exploitation outperforms the vanilla Transformer-Big model, with only less than half of the parameters.

4.3 Analysis

We conducted extensive analysis from different perspectives to better understand our model. All results are reported on the En⇒De task with Transformer-Base.

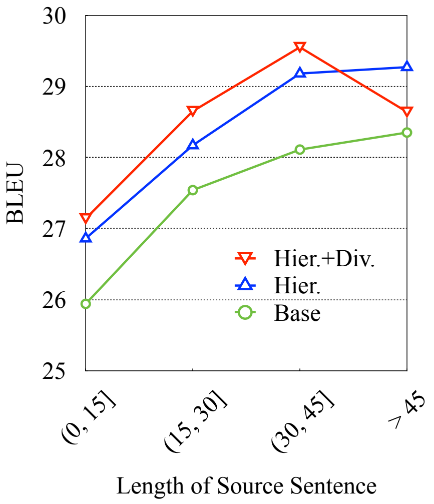

Figure 4: BLEU scores on the En⇒De test set with respect to various input sentence lengths. “Hier.” denotes hierarchical aggregation and “Div.” denotes diversity regularization.

4.3.1 Length Analysis

Following Bahdanau et al. (2015) and Tu et al. (2016), we grouped sentences of similar lengths together and computed the BLEU score for each group, as shown in Figure 4. Generally, the performance of TRANSFORMER-BASE goes up with the increase of input sentence lengths, which is superior to the performance of RNN-based NMT models on long sentences reported by (Bentivogli et al., 2016). We attribute this to the strength of self-attention mechanism to model global dependencies without regard to their distance.

Clearly, the proposed approaches outperform the baseline model in all length segments, while there are still considerable differences between the two variations. Hierarchical aggregation consistently outperforms the baseline model, and the improvement goes up on long sentences. One possible reason is that long sentences indeed require deep aggregation mechanisms. Introducing diversity regularization further improves performance on most sentences (e.g. ≤ 45), while the improvement degrades on long sentences (e.g. > 45). We conjecture that complex long sentences may need to store duplicate information across layers, which conflicts with the diversity objective.

4.3.2 Effect on Encoder and Decoder

Both encoder and decoder are composed of a stack of L layers, which may benefit from the proposed approach. In this experiment, we investigated how our models affect the two components, as shown in Table 3. Exploiting deep representations of encoder or decoder individually consistently outperforms the vanilla baseline model, and exploiting both components further improves the performance. These results provide support for the claim that exploiting deep representations is useful for both understanding input sequence and generating output sequence.

Model	**Applied to**		**BLEU**
	Encoder	*Decoder*	
BASE	N/A	N/A	26.13
OURS	✓	×	26.32
	×	✓	26.41
	✓	✓	**26.69**

Table 3: Experimental results of applying hierarchical aggregation to different components on En⇒De validation set.

4.3.3 Impact of Aggregation Choices

Model	**RESIDUAL**	**AGGREGATE**	**BLEU**
BASE	N/A	N/A	26.13
OURS	None	SIGMOID	25.48
	Top		26.59
	All		**26.69**
		RELU	26.56
		ATTENTION	26.54

Table 4: Impact of residual connections and aggregation functions for hierarchical layer aggregation.

As described in Section 3.1.1, the function of hierarchical layer aggregation is defined as

$$\mathrm{AGG}(x, y, z) = \mathrm{LN}(\mathrm{FF}([x; y; z]) + x + y + z),$$

where $\mathrm{FF}(\cdot)$ is a feed-forward network with a sigmoid activation in between. In addition, all the input layers $\{x, y, z\}$ have residual connections to the output. In this experiment, we evaluated the impact of residual connection options, as well as different choices for the aggregation function, as shown in Table 4.

Concerning residual connections, if none of the input layers are connected to the output layer (“None”), the performance would decrease. The translation performance is improved when the output is connected to only the top level of the input layers (“Top”), while connecting to all input layers (“All”) achieves the best performance. This indicates that cross-layer connections are necessary to avoid the gradient vanishing problem.

Besides the feed-forward network with sigmoid activation, we also tried two other aggregation functions for $\textsc{Ff}(\cdot)$: (1) A feed-forward network with a ReLU activation in between; and (2) multi-head self-attention layer that constitutes the encoder and decoder layers in the TRANSFORMER model. As seen, all the three functions consistently improve the translation performance, proving the robustness of the proposed approaches.

4.4 Visualization of Aggregation

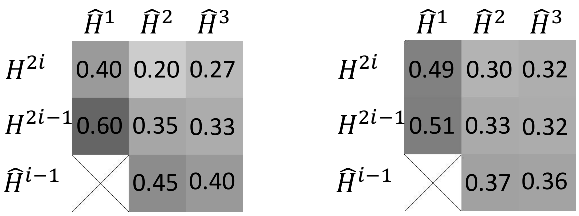

(a) No regularization. (b) With regularization.

Figure 5: Visualization of the exploitation of input representations for hierarchical aggregation. x-axis is the aggregation node and y-axis is the input representation. $\widehat{H}^i$ denotes the i-th aggregation layer, and H^i denotes the i-th encoder layer. The rightmost and topmost position in x-axis and y-axis respectively represent the highest layer.

To investigate the impact of diversity regularization, we visualized the exploitation of the input representations for hierarchical aggregation in encoder side, as shown in Figure 5. Let $\mathbf{H}^i = \{H^{2i}, H^{2i-1}, \widehat{H}^{i-1}\}$ be the input representations, we calculated the exploitation of the j-th input as

$$s_j = \frac{\sum_{w \in W_j} |w|}{\sum_{H_{j'} \in \mathbf{H}} \{\sum_{w' \in W_{j'}} |w'|\}}, \qquad (11)$$

where W_j is the parameter matrix associated with the input H_j. The score s_j is a rough estimation of the contribution of H_j to the aggregation $\widehat{H}^i$.

We have two observations. First, the model tends to utilize the bottom layer more than the top one, indicating the necessity of fusing information across layers. Second, using the diversity regularization in Figure 5(b) can encourage each layer to contribute more equally to the aggregation. We hypothesize this is because of the diversity regularization term encouraging the different layers to contain diverse and equally important information.

5 Related Work

Representation learning is at the core of deep learning. Our work is inspired by technological advances in representation learning, specifically in the field of *deep representation learning* and *representation interpretation*.

Deep Representation Learning Deep neural networks have advanced the state of the art in various communities, such as computer vision and natural language processing. One key challenge of training deep networks lies in how to transform information across layers, especially when the network consists of hundreds of layers.

In response to this problem, ResNet (He et al., 2016) uses skip connections to combine layers by simple, one-step operations. Densely connected network (Huang et al., 2017) is designed to better propagate features and losses through skip connections that concatenate all the layers in stages. Yu et al. (2018) design structures iteratively and hierarchically merge the feature hierarchy to better fuse information in a deep fusion.

Concerning machine translation, Meng et al. (2016) and Zhou et al. (2016) have shown that deep networks with advanced connecting strategies outperform their shallow counterparts. Due to its simplicity and effectiveness, skip connection becomes a standard component of state-of-the-art NMT models (Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017). In this work, we prove that deep representation exploitation can further improve performance over simply using skip connections.

Representation Interpretation Several researchers have tried to visualize the representation of each layer to help better understand what information each layer captures (Zeiler and Fergus, 2014; Li et al., 2016; Ding et al., 2017). Concerning natural language processing tasks, Shi et al. (2016) find that both local and global source syntax are learned by the NMT encoder and different types of syntax are captured at different layers. Anastasopoulos and Chiang (2018) show that higher level layers are more representative than lower level layers. Peters et al. (2018) demonstrate that higher-level layers capture context-dependent aspects of word meaning while lower-level layers model aspects of syntax. Inspired by these observations, we propose to expose all of these representations to better fuse information across layers. In addition, we introduce a regularization to encourage different layers to capture diverse information.

6 Conclusion

In this work, we propose to better exploit deep representations that are learned by multiple layers for neural machine translation. Specifically, the hierarchical aggregation with diversity regularization achieves the best performance by incorporating more depth and sharing across layers and by encouraging layers to capture different information. Experimental results on WMT14 English⇒German and WMT17 Chinese⇒English show that the proposed approach consistently outperforms the state-of-the-art TRANSFORMER baseline by +0.54 and +0.63 BLEU points, respectively. By visualizing the aggregation process, we find that our model indeed utilizes lower layers to effectively fuse the information across layers.

Future directions include validating our approach on other architectures such as RNN (Bahdanau et al., 2015) or CNN (Gehring et al., 2017) based NMT models, as well as combining with other advanced techniques (Shaw et al., 2018; Shen et al., 2018; Yang et al., 2018; Li et al., 2018) to further improve the performance of TRANSFORMER.

Acknowledgments

We thank the anonymous reviewers for their insightful comments.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *NAACL*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *EMNLP*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, Goerge Foster, Llion Jones, Parmar Niki, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In *ACL*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *ACL*.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *ACL*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *CVPR*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018. Multi-head attention with disagreement regularization. In *EMNLP*.

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John E Hopcroft. 2016. Convergent learning: Do different neural networks learn the same representations? In *ICLR*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Fandong Meng, Zhengdong Lu, Zhaopeng Tu, Hang Li, and Qun Liu. 2016. A deep memory-based architecture for sequence-to-sequence learning. In *ICLR Workshop*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. DiSAN: directional self-attention network for RNN/CNN-free language understanding. In *AAAI*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *TACL*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *EMNLP*.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. 2018. Deep layer aggregation. In *CVPR*.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *ECCV*.

Jiacheng Zhang, Yanzhuo Ding, Shiqi Shen, Yong Cheng, Maosong Sun, Huanbo Luan, and Yang Liu. 2017. THUMT: An Open Source Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1706.06415*.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *TACL*.

Why Self-Attention?
A Targeted Evaluation of Neural Machine Translation Architectures

Gongbo Tang[1*] **Mathias Müller**[2] **Annette Rios**[2] **Rico Sennrich**[2,3]

[1]Department of Linguistics and Philology, Uppsala University

[2]Institute of Computational Linguistics, University of Zurich

[3]School of Informatics, University of Edinburgh

Abstract

Recently, non-recurrent architectures (convolutional, self-attentional) have outperformed RNNs in neural machine translation. CNNs and self-attentional networks can connect distant words via shorter network paths than RNNs, and it has been speculated that this improves their ability to model long-range dependencies. However, this theoretical argument has not been tested empirically, nor have alternative explanations for their strong performance been explored in-depth. We hypothesize that the strong performance of CNNs and self-attentional networks could also be due to their ability to extract semantic features from the source text, and we evaluate RNNs, CNNs and self-attention networks on two tasks: subject-verb agreement (where capturing long-range dependencies is required) and word sense disambiguation (where semantic feature extraction is required). Our experimental results show that: 1) self-attentional networks and CNNs do not outperform RNNs in modeling subject-verb agreement over long distances; 2) self-attentional networks perform distinctly better than RNNs and CNNs on word sense disambiguation.

1 Introduction

Different architectures have been shown to be effective for neural machine translation (NMT), ranging from recurrent architectures (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2015; Sutskever et al., 2014; Luong et al., 2015) to convolutional (Kalchbrenner and Blunsom, 2013; Gehring et al., 2017) and, most recently, fully self-attentional (Transformer) models (Vaswani et al., 2017). Since comparisons (Gehring et al., 2017; Vaswani et al., 2017; Hieber et al., 2017) are mainly carried out via BLEU (Papineni et al., 2002), it is inherently difficult to attribute gains in BLEU to architectural properties.

Recurrent neural networks (RNNs) (Elman, 1990) can easily deal with variable-length input sentences and thus are a natural choice for the encoder and decoder of NMT systems. Modern variants of RNNs, such as GRUs (Cho et al., 2014) and LSTMs (Hochreiter and Schmidhuber, 1997), address the difficulty of training recurrent networks with long-range dependencies. Gehring et al. (2017) introduce a neural architecture where both the encoder and decoder are based on CNNs, and report better BLEU scores than RNN-based NMT models. Moreover, the computation over all tokens can be fully parallelized during training, which increases efficiency. Vaswani et al. (2017) propose Transformer models, which are built entirely with attention layers, without convolution or recurrence. They report new state-of-art BLEU scores for EN→DE and EN→FR. Yet, the BLEU metric is quite coarse-grained, and offers no insight as to which aspects of translation are improved by different architectures.

To explain the observed improvements in BLEU, previous work has drawn on theoretical arguments. Both Gehring et al. (2017) and Vaswani et al. (2017) argue that the length of the paths in neural networks between co-dependent elements affects the ability to learn these dependencies: the shorter the path, the easier the model learns such dependencies. The papers argue that Transformers and CNNs are better suited than RNNs to capture long-range dependencies.

However, this claim is based on a theoretical argument and has not been empirically tested. We argue other abilities of non-recurrent networks could be responsible for their strong performance. Specifically, we hypothesize that the improvements in BLEU are due to CNNs and Transformers being strong semantic feature extractors.

---

[*] Work carried out during a visit to the machine translation group at the University of Edinburgh.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4263–4272
Brussels, Belgium, October 31 - November 4, 2018.

In this paper, we evaluate all three popular NMT architectures: models based on RNNs (referred to as *RNNS2S* in the remainder of the paper), based on CNNs (referred to as *ConvS2S*) and self-attentional models (referred to as Transformers). Motivated by the aforementioned theoretical claims regarding path length and semantic feature extraction, we evaluate their performance on a subject-verb agreement task (that requires modeling long-range dependencies) and a word sense disambiguation (WSD) task (that requires extracting semantic features). Both tasks build on test sets of contrastive translation pairs, *Lingeval97* (Sennrich, 2017) and *ContraWSD* (Rios et al., 2017).

The main contributions of this paper can be summarized as follows:

- We test the theoretical claims that architectures with shorter paths through networks are better at capturing long-range dependencies. Our experimental results on modeling subject-verb agreement over long distances do not show any evidence that Transformers or CNNs are superior to RNNs in this regard.
- We empirically show that the number of attention heads in Transformers impacts their ability to capture long-distance dependencies. Specifically, many-headed multi-head attention is essential for modeling long-distance phenomena with only self-attention.
- We empirically show that Transformers excel at WSD, indicating that they are strong semantic feature extractors.

2 Related work

Yin et al. (2017) are the first to compare CNNs, LSTMs and GRUs on several NLP tasks. They find that CNNs are better at tasks related to semantics, while RNNs are better at syntax-related tasks, especially for longer sentences.

Based on the work of Linzen et al. (2016), Bernardy and Lappin (2017) find that RNNs perform better than CNNs on a subject-verb agreement task, which is a good proxy for how well long-range dependencies are captured. Tran et al. (2018) find that a Transformer language model performs worse than an RNN language model on a subject-verb agreement task. They, too, note that this is especially true as the distance between subject and verb grows, even if RNNs resulted in a higher perplexity on the validation set. This result of Tran et al. (2018) is clearly in contrast to the general finding that Transformers are better than RNNs for NMT tasks.

Bai et al. (2018) evaluate CNNs and LSTMs on several sequence modeling tasks. They conclude that CNNs are better than RNNs for sequence modeling. However, their CNN models perform much worse than the state-of-art LSTM models on some sequence modeling tasks, as they themselves state in the appendix.

Tang et al. (2018) evaluate different RNN architectures and Transformer models on the task of historical spelling normalization which translates a historical spelling into its modern form. They find that Transformer models surpass RNN models only in high-resource conditions.

In contrast to previous studies, we focus on the machine translation task, where architecture comparisons so far are mostly based on BLEU.

3 Background

3.1 NMT Architectures

We evaluate three different NMT architectures: RNN-based models, CNN-based models, and Transformer-based models. All of them have a bipartite structure in the sense that they consist of an encoder and a decoder. The encoder and the decoder interact via a soft-attention mechanism (Bahdanau et al., 2015; Luong et al., 2015), with one or multiple attention layers.

In the following sections, h_i^l is the hidden state at step i of layer l, h_{i-1}^l represents the hidden state at the previous step of layer l, h_i^{l-1} means the hidden state at i of $l-1$ layer, E_{x_i} represents the embedding of x_i, and $e_{pos,i}$ denotes the positional embedding at position i.

3.1.1 RNN-based NMT

RNNs are stateful networks that change as new inputs are fed to them, and each state has a direct connection only to the previous state. Thus, the path length of any two tokens with a distance of n in RNNs is exactly n. Figure 1 (a) shows an illustration of RNNs.

$$h_i^l = h_i^{l-1} + f_{rnn}(h_i^{l-1}, h_{i-1}^l) \tag{1}$$

In deep architectures, two adjacent layers are commonly connected with residual connections. In the lth encoder layer, h_i^l is generated by Equation 1,

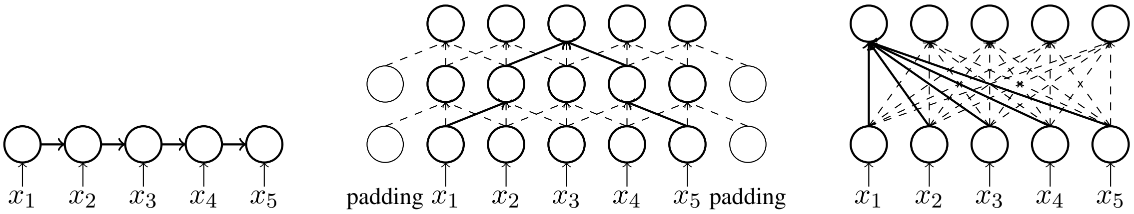

(a) RNN (b) CNN (c) Self-attention

Figure 1: Architectures of different neural networks in NMT.

where f_{rnn} is the RNN (GRU or LSTM) function. In the first layer, $h_i^0 = f_{rnn}(E_{x_i}, h_{i-1}^0)$.

In addition to the connection between the encoder and decoder via attention, the initial state of the decoder is usually initialized with the average of the hidden states or the last hidden state of the encoder.

3.1.2 CNN-based NMT

CNNs are hierarchical networks, in that convolution layers capture local correlations. The local context size depends on the size of the kernel and the number of layers. In order to keep the output the same length as the input, CNN models add padding symbols to input sequences. Given an L-layer CNN with a kernel size k, the largest context size is $L(k-1)$. For any two tokens in a local context with a distance of n, the path between them is only $\lceil n/(k-1) \rceil$.

As Figure 1 (b) shows, a 2-layer CNN with kernel size 3 "sees" an effective local context of 5 tokens. The path between the first token and the fifth token is only 2 convolutions.[1] Since CNNs do not have a means to infer the position of elements in a sequence, positional embeddings are introduced.

$$h_i^l = h_i^{l-1} + f_{cnn}(W^l[h_{i-\lfloor k/2 \rfloor}^{l-1}; ...; h_{i+\lfloor k/2 \rfloor}^{l-1}] + b^l) \quad (2)$$

The hidden state h_i^l shown in Equation 2 is related to the hidden states in the same convolution and the hidden state h_i^{l-1} from the previous layer. k denotes the kernel size of CNNs and f_{cnn} is a nonlinearity. *ConvS2S* chooses Gated Linear Units (GLU) which can be viewed as a gated variation of ReLUs. W^l are called convolutional filters. In the input layer, $h_i^0 = E_{x_i} + e_{pos,i}$.

3.1.3 Transformer-based NMT

Transformers rely heavily on self-attention networks. Each token is connected to any other token in the same sentence directly via self-attention. Moreover, Transformers feature attention networks with multiple *attention heads*. Multi-head attention is more fine-grained, compared to conventional 1-head attention mechanisms. Figure 1 (c) illustrates that any two tokens are connected directly: the path length between the first and the fifth tokens is 1. Similar to CNNs, positional information is also preserved in positional embeddings.

The hidden state in the Transformer encoder is calculated from all hidden states of the previous layer. The hidden state h_i^l in a self-attention network is computed as in Equation 3.

$$h_i^l = h_i^{l-1} + f(\text{self-attention}(h_i^{l-1})) \quad (3)$$

where f represents a feedforward network with ReLU as the activation function and layer normalization. In the input layer, $h_i^0 = E_{x_i} + e_{pos,i}$. The decoder additionally has a multi-head attention over the encoder hidden states.

3.2 Contrastive Evaluation of Machine Translation

Since we evaluate different NMT architectures explicitly on subject-verb agreement and WSD (both happen implicitly during machine translation), BLEU as a measure of overall translation quality is not helpful. In order to conduct these targeted evaluations, we use contrastive test sets.

Sets of contrastive translations can be used to analyze specific types of errors. Human reference translations are paired with one or more contrastive variants, where a specific type of error is introduced automatically.

The evaluation procedure then exploits the fact that NMT models are conditional language models. By virtue of this, given any source sentence S

---

[1] Note that the decoder employs masking to avoid conditioning the model on future information, which reduces the effective context size to $L\frac{k-1}{2}$.

and target sentence T, any NMT model can assign to them a probability $P(T|S)$. If a model assigns a higher score to the correct target sentence than to a contrastive variant that contains an error, we consider it a correct decision. The accuracy of a model on such a test set is simply the percentage of cases where the correct target sentence is scored higher than all contrastive variants.

Contrastive evaluation tests the sensitivity of NMT models to specific translation errors. The contrastive examples are designed to capture specific translation errors rather than evaluating the global quality of NMT models. Although they do not replace metrics such as BLEU, they give further insights into the performance of models, on specific linguistic phenomena.

3.2.1 *Lingeval97*

Lingeval97 has over 97,000 English→German contrastive translation pairs featuring different linguistic phenomena, including subject-verb agreement, noun phrase agreement, separable verb-particle constructions, transliterations and polarity. In this paper, we are interested in evaluating the performance on long-range dependencies. Thus, we focus on the subject-verb agreement category which consists of 35,105 instances.

In German, verbs must agree with their subjects in both grammatical number and person. Therefore, in a contrastive translation, the grammatical number of a verb is swapped. Table 1 gives an example.

English:	[...] plan will be approved
German:	[...] **Plan** verabschiedet **wird**
Contrast:	[...] **Plan** verabschiedet **werden**

Table 1: An example of a contrastive pair in the subject-verb agreement category.

3.2.2 *ContraWSD*

In *ContraWSD*, given an ambiguous word in the source sentence, the correct translation is replaced by another meaning of the ambiguous word which is incorrect. For example, in a case where the English word *line* is the correct translation of the German source word *Schlange*, *ContraWSD* replaces *line* with the other translations of *Schlange*, such as *snake*, *serpent*, to generate contrastive translations.

For German→English, *ContraWSD* contains 84 different German word senses. It has 7,200 German→English lexical ambiguities, each lexical ambiguity instance has 3.5 contrastive translations on average. For German→French, it consists of 71 different German word senses. There are 6,700 German→French lexical ambiguities, with an average of 2.2 contrastive translations each lexical ambiguity instance. All the ambiguous words are nouns so that the disambiguation is not possible simply based on syntactic context.

4 Subject-verb Agreement

The subject-verb agreement task is the most popular choice for evaluating the ability to capture long-range dependencies and has been used in many studies (Linzen et al., 2016; Bernardy and Lappin, 2017; Sennrich, 2017; Tran et al., 2018). Thus, we also use this task to evaluate different NMT architectures on long-range dependencies.

4.1 Experimental Settings

Different architectures are hard to compare fairly because many factors affect performance. We aim to create a level playing field for the comparison by training with the same toolkit, *Sockeye* (Hieber et al., 2017) which is based on MXNet (Chen et al., 2015). In addition, different hyperparameters and training techniques (such as label smoothing or layer normalization) have been found to affect the performance (Chen et al., 2018). We apply the same hyperparameters and techniques for all architectures except the parameters of each specific architecture. Since the best hyperparameters for different architectures may be diverse, we verify our hyperparameter choice by comparing our results to those published previously. Our models achieve similar performance to that reported by Hieber et al. (2017) with the best available settings. In addition, we extend *Sockeye* with an interface that enables scoring of existing translations, which is required for contrastive evaluation.

All the models are trained with 2 GPUs. During training, each mini-batch contains 4096 tokens. A model checkpoint is saved every 4,000 updates. We use *Adam* (Kingma and Ba, 2015) as the optimizer. The initial learning rate is set to 0.0002. If the performance on the validation set has not improved for 8 checkpoints, the learning rate is multiplied by 0.7. We set the early stopping patience to 32 checkpoints. All the neural networks have 8 layers. For *RNNS2S*, the encoder has 1 bi-directional LSTM and 6 stacked uni-directional LSTMs, and the decoder is a stack of 8 uni-directional LSTMs. The size of embeddings and hidden states is 512. We apply layer normalization and label smoothing (0.1) in all models. We tie the source and target embeddings. The dropout rate of embeddings and Transformer blocks is set to 0.1. The dropout rate of RNNs and CNNs is 0.2. The kernel size of CNNs is 3. Transformers have an 8-head attention mechanism.

To test the robustness of our findings, we also test a different style of RNN architecture, from a different toolkit. We evaluate bi-deep transitional RNNs (Miceli Barone et al., 2017) which are state-of-art RNNs in machine translation. We use the bi-deep RNN-based model (*RNN-bideep*) implemented in *Marian* (Junczys-Dowmunt et al., 2018). Different from the previous settings, we use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$. The initial learning rate is 0.0003. We tie target embeddings and output embeddings. Both the encoder and decoder have 4 layers of LSTM units, only the encoder layers are bi-directional. LSTM units consist of several cells (deep transition): 4 in the first layer of the decoder, 2 cells everywhere else.

We use training data from the WMT17 shared task.[2] We use *newstest2013* as the validation set, and use *newstest2014* and *newstest2017* as the test sets. All BLEU scores are computed with *SacreBLEU* (Post, 2018). There are about 5.9 million sentence pairs in the training set after preprocessing with Moses scripts. We learn a joint BPE model with 32,000 subword units (Sennrich et al., 2016). We employ the model that has the best perplexity on the validation set for the evaluation.

4.2 Overall Results

Table 2 reports the BLEU scores on *newstest2014* and *newstest2017*, the perplexity on the validation set, and the accuracy on long-range dependencies.[3] *Transformer* achieves the highest accuracy on this task and the highest BLEU scores on both *newstest2014* and *newstest2017*. Compared to *RNNS2S*, *ConvS2S* has slightly better results regarding BLEU scores, but a much lower accuracy on long-range dependencies. The *RNN-bideep* model achieves distinctly better BLEU scores and a higher accuracy on long-range dependencies. However, it still cannot outperform Transformers on any of the tasks.

Model	2014	2017	PPL	Acc(%)
RNNS2S	23.3	25.1	6.1	95.1
ConvS2S	23.9	25.2	7.0	84.9
Transformer	**26.7**	**27.5**	**4.5**	**97.1**
RNN-bideep	24.7	26.1	5.7	96.3

Table 2: The results of different NMT models, including the BLEU scores on *newstest2014* and *newstest2017*, the perplexity on the validation set, and the accuracy of long-range dependencies.

Figure 2 shows the performance of different architectures on the subject-verb agreement task. It is evident that *Transformer*, *RNNS2S*, and *RNN-bideep* perform much better than *ConvS2S* on long-range dependencies. However, *Transformer*, *RNNS2S*, and *RNN-bideep* are all robust over long distances. *Transformer* outperforms *RNN-bideep* for distances 11-12, but *RNN-bideep* performs equally or better for distance 13 or higher. Thus, we cannot conclude that Transformer models are particularly stronger than RNN models for long distances, despite achieving higher average accuracy on distances above 10.

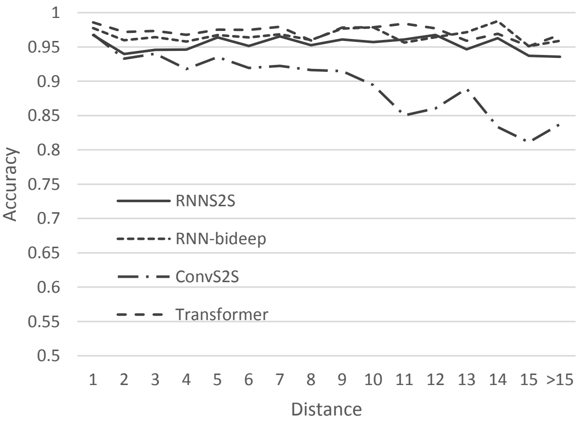

Figure 2: Accuracy of different NMT models on the subject-verb agreement task.

4.2.1 CNNs

Theoretically, the performance of CNNs will drop when the distance between the subject and the verb exceeds the local context size. However, *ConvS2S* is also clearly worse than *RNNS2S* for subject-verb agreement within the local context size.

In order to explore how the ability of *ConvS2S* to capture long-range dependencies depends on the local context size, we train additional systems,

---

[2] `http://www.statmt.org/wmt17/translation-task.html`

[3] We report average accuracy on instances where the distance between subject and verb is longer than 10 words.

varying the number of layers and kernel size. Table 3 shows the performance of different *ConvS2S* models. Figure 3 displays the performance of two 8-layer CNNs with kernel size 3 and 7, a 6-layer CNN with kernel size 3, and *RNNS2S*. The results indicate that the accuracy increases when the local context size becomes larger, but the BLEU score does not. Moreover, *ConvS2S* is still not as good as *RNNS2S* for subject-verb agreement.

Layer	K	Ctx	2014	2017	Acc(%)
4	3	4	22.9	24.2	81.1
6	3	6	23.6	25.0	82.5
8	3	8	**23.9**	**25.2**	84.9
8	5	16	23.5	24.7	89.7
8	7	24	23.3	24.6	**91.3**

Table 3: The performance of *ConvS2S* with different settings. *K* means the kernel size. The *ctx* column is the theoretical largest local context size in the masked decoder.

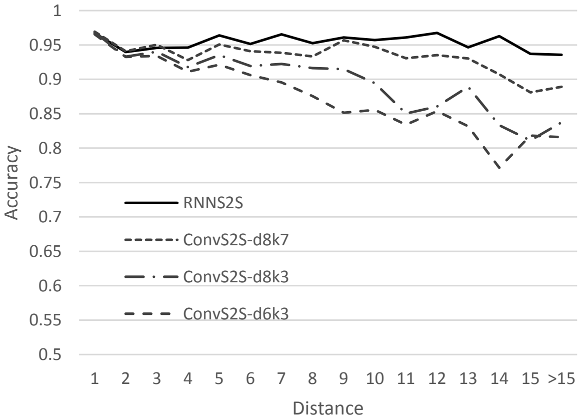

Figure 3: Results of *ConvS2S* models and the *RNNS2S* model at different distances.

Regarding the explanation for the poor performance of *ConvS2S*, we identify the limited context size as a major problem. One assumption to explain the remaining difference is that, scale invariance of CNNs is relatively poor (Xu et al., 2014). Scale-invariance is important in NLP, where the distance between arguments is flexible, and current recurrent or attentional architectures are better suited to handle this variance.

Our empirical results do not confirm the theoretical arguments in Gehring et al. (2017) that CNNs can capture long-range dependencies better with a shorter path. The BLEU score does not correlate well with the targeted evaluation of long-range distance interactions. This is due to the locality of BLEU, which only measures on the level of n-grams, but it may also indicate that there are other trade-offs between the modeling of different phenomena depending on hyperparameters. If we aim to get better performance on long-range dependencies, we can take this into account when optimizing hyperparameters.

4.2.2 RNNs vs. *Transformer*

Even though *Transformer* achieves much better BLEU scores than *RNNS2S* and *RNN-bideep*, the accuracies of these architectures on long-range dependencies are close to each other in Figure 2.

Our experimental result contrasts with the result from Tran et al. (2018). They find that Transformers perform worse than LSTMs on the subject-verb agreement task, especially when the distance between the subject and the verb becomes longer. We perform several experiments to analyze this discrepancy with Tran et al. (2018).

A first hypothesis is that this is caused by the amount of training data, since we used much larger datasets than Tran et al. (2018). We retrain all the models with a small amount of training data similar to the amount used by Tran et al. (2018), about 135K sentence pairs. The other training settings are the same as in Section 4.1. We do not see the expected degradation of *Transformer-s*, compared to *RNNS2S-s* (see Figure 4). In Table 4, the performance of *RNNS2S-s* and *Transformer-s* is similar, including the BLEU scores on *newstest2014, newstest2017*, the perplexity on the validation set, and the accuracy on the long-range dependencies.

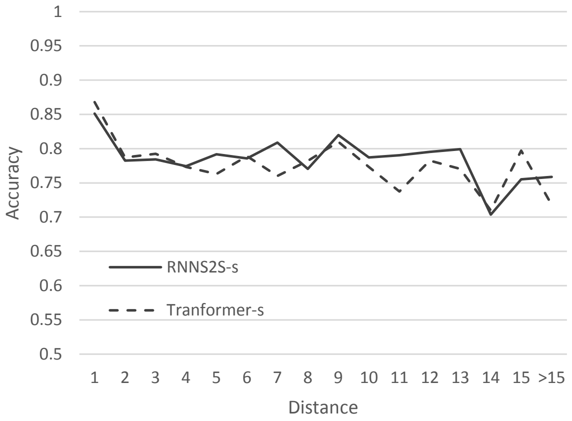

Figure 4: Results of a Transformer and *RNNS2S* model trained on a small dataset.

A second hypothesis is that the experimental settings lead to the different results. In order to investigate this, we do not only use a small training set, but also replicate the experimental settings of Tran et al. (2018). The main changes are neural network layers (8→4); embedding size (512→128); multi-head size (8→2); dropout rate (0.1→0.2); checkpoint save frequency (4,000→1,000), and initial learning rate (0.0002→0.001).

Model	2014	2017	PPL	Acc(%)
RNNS2S-s	7.3	7.8	47.8	77.3
Trans-s	7.2	8.0	44.6	74.6
RNNS2S-re	9.2	10.5	39.2	77.7
Trans-re-h2	9.6	10.7	36.9	71.9
Trans-re-h4	9.5	11.9	35.8	73.8
Trans-re-h8	9.4	10.4	36.0	75.3

Table 4: The results of different models with small training data and replicate settings. *Trans* is short for Transformer. Models with the suffix "*-s*" are models trained with small data set. Models with the suffix "*-re*" are models trained with replicate settings. "*h2, h4, h8*" indicates the number of attention heads for Transformer models.

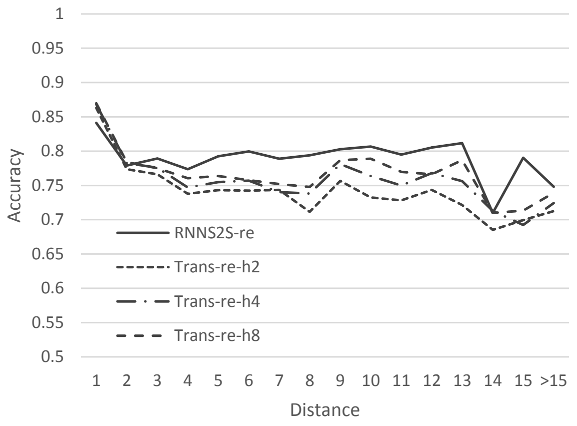

Figure 5: Results of the models with replicate settings, varying the number of attention heads for the Transformer models.

In the end, we get a result that is similar to Tran et al. (2018). In Figure 5, *Transformer-re-h2* performs clearly worse than *RNNS2S-re* on long-range dependencies. By increasing the number of heads in multi-head attention, subject-verb accuracy over long distances can be improved substantially, even though it remains below that of *RNNS2S-re*. Also, the effect on BLEU is small.

Our results suggest that the importance of multi-head attention with a large number of heads is larger than BLEU would suggest, especially for the modeling of long-distance phenomena, since multi-head attention provides a way for the model to attend to both local and distant context, whereas distant context may be overshadowed by local context in an attention mechanism with a single or few heads.

Although our study is not a replication of Tran et al. (2018), who work on a different task and a different test set, our results do suggest an alternative interpretation of their findings, namely that the poor performance of the Transformer in their experiments is due to hyperparameter choice. Rather than concluding that RNNs are superior to Transformers for the modeling of long-range dependency phenomena, we find that the number of heads in multi-head attention affects the ability of Transformers to model long-range dependencies in subject-verb agreement.

5 WSD

Our experimental results on the subject-verb agreement task demonstrate that CNNs and Transformer are not better at capturing long-range dependencies compared to RNNs, even though the paths in CNNs and Transformers are shorter. This finding is not in accord with the theoretical argument in both Gehring et al. (2017) and Vaswani et al. (2017). However, these architectures perform well empirically according to BLEU. Thus, we further evaluate these architectures on WSD, to test our hypothesis that non-recurrent architectures are better at extracting semantic features.

5.1 Experimental settings

We evaluate all architectures on *ContraWSD* on both DE→EN and DE→FR. We reuse the parameter settings in Section 4.1, except that: the initial learning rate of *ConvS2S* is reduced from 0.0003 to 0.0002 in DE→EN; the checkpoint saving frequency is changed from 4,000 to 1,000 in DE→FR because of the training data size.

For DE→EN, the training set, validation set, and test set are the same as the other direction EN→DE. For DE→FR, we use around 2.1 million sentence pairs from Europarl (v7) (Tiedemann, 2012)[4] and News Commentary (v11) cleaned by Rios et al. (2017)[5] as our training set. We use *newstest2013* as the evaluation set, and use *newstest2012* as the test set. All the data is preprocessed with Moses scripts.

---

[4] `http://opus.nlpl.eu/Europarl.php`

[5] `http://data.statmt.org/ContraWSD/`

Model	DE→EN				DE→FR		
	PPL	2014	2017	Acc(%)	PPL	2012	Acc(%)
RNNS2S	5.7	29.1	30.1	84.0	7.06	16.4	72.2
ConvS2S	6.3	29.1	30.4	82.3	7.93	16.8	72.7
Transformer	**4.3**	**32.7**	33.7	**90.3**	**4.9**	**18.7**	**76.7**
uedin-wmt17	–	–	**35.1**	87.9	–	–	–
TransRNN	5.2	30.5	31.9	86.1	6.3	17.6	74.2

Table 5: The results of different architectures on *newstest* sets and *ContraWSD*. *PPL* is the perplexity on the validation set. *Acc* means accuracy on the test set.

In addition, we also compare to the best result reported for DE→EN, achieved by *uedin-wmt17* (Sennrich et al., 2017), which is an ensemble of 4 different models and reranked with right-to-left models.[6] *uedin-wmt17* is based on the bi-deep RNNs (Miceli Barone et al., 2017) that we mentioned before. To the original 5.9 million sentence pairs in the training set, they add 10 million synthetic pairs with back-translation.

5.2 Overall Results

Table 5 gives the performance of all the architectures, including the perplexity on validation sets, the BLEU scores on *newstest*, and the accuracy on *ContraWSD*. Transformers distinctly outperform *RNNS2S* and *ConvS2S* models on DE→EN and DE→FR. Moreover, the Transformer model on DE→EN also achieves higher accuracy than *uedin-wmt17*, although the BLEU score on *newstest2017* is 1.4 lower than *uedin-wmt17*. We attribute this discrepancy between BLEU and WSD performance to the use of synthetic news training data in *uedin-wmt17*, which causes a large boost in BLEU due to better domain adaptation to *newstest*, but which is less helpful for ContraWSD, whose test set is drawn from a variety of domains.

For DE→EN, *RNNS2S* and *ConvS2S* have the same BLEU score on *newstest2014*, *ConvS2S* has a higher score on *newstest2017*. However, the WSD accuracy of *ConvS2S* is 1.7% lower than *RNNS2S*. For DE→FR, *ConvS2S* achieves slightly better results on both BLEU scores and accuracy than *RNNS2S*.

The Transformer model strongly outperforms the other architectures on this WSD task, with a gap of 4–8 percentage points. This affirms our hypothesis that Transformers are strong semantic features extractors.

5.3 Hybrid Encoder-Decoder Model

In recent work, Chen et al. (2018) find that hybrid architectures with a Transformer encoder and an RNN decoder can outperform a pure Transformer model. They speculate that the Transformer encoder is better at encoding or extracting features than the RNN encoder, whereas the RNN is better at conditional language modeling.

For WSD, it is unclear whether the most important component is the encoder, the decoder, or both. Following the hypothesis that Transformer encoders excel as semantic feature extractors, we train a hybrid encoder-decoder model (*TransRNN*) with a Transformer encoder and an RNN decoder.

The results (in Table 5) show that *TransRNN* performs better than *RNNS2S*, but worse than the pure Transformer, both in terms of BLEU and WSD accuracy. This indicates that WSD is not only done in the encoder, but that the decoder also affects WSD performance. We note that Chen et al. (2018); Domhan (2018) introduce the techniques in Transformers into RNN-based models, with reportedly higher BLEU. Thus, it would be interesting to see if the same result holds true with their architectures.

6 Conclusion

In this paper, we evaluate three popular NMT architectures, *RNNS2S*, *ConvS2S*, and Transformers, on subject-verb agreement and WSD by scoring contrastive translation pairs.

We test the theoretical claims that shorter path lengths make models better capture long-range dependencies. Our experimental results show that:

- There is no evidence that CNNs and Transformers, which have shorter paths through networks, are empirically superior to RNNs in modeling subject-verb agreement over long distances.

---

[6] `https://github.com/a-rios/ContraWSD/tree/master/baselines`

- The number of heads in multi-head attention affects the ability of a Transformer to model long-range dependencies in the subject-verb agreement task.
- Transformer models excel at another task, WSD, compared to the CNN and RNN architectures we tested.

Lastly, our findings suggest that assessing the performance of NMT architectures means finding their inherent trade-offs, rather than simply computing their overall BLEU score. A clear understanding of those strengths and weaknesses is important to guide further work. Specifically, given the idiosyncratic limitations of recurrent and self-attentional models, combining them is an exciting line of research. The apparent weakness of CNN architectures on long-distance phenomena is also a problem worth tackling, and we can find inspiration from related work in computer vision (Xu et al., 2014).

Acknowledgments

We thank all the anonymous reviews and Joakim Nivre who give a lot of valuable and insightful comments. We appreciate the grants provided by Erasmus+ Programme and Anna Maria Lundin's scholarship committee. GT is funded by the Chinese Scholarship Council (grant number `201607110016`). MM, AR and RS have received funding from the Swiss National Science Foundation (grant number `105212_169888`).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Jean-Philippe Bernardy and Shalom Lappin. 2017. Using Deep Neural Networks to Learn Syntactic Agreement. *LiLT (Linguistic Issues in Language Technology)*, 15(2):1–15.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86. Association for Computational Linguistics.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Tobias Domhan. 2018. How much attention do you need? a granular analysis of neural machine translation architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1799–1808. Association for Computational Linguistics.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1243–1252, Sydney, Australia. PMLR.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1712.05690*.

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

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast Neural Machine Translation in C++. *arXiv preprint arXiv:1804.00344*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Antonio Valerio Miceli Barone, Jindřich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017. Deep architectures for Neural Machine Translation. In *Proceedings of the Second Conference on Machine Translation*, pages 99–107, Copenhagen, Denmark. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Annette Rios, Laura Mascarell, and Rico Sennrich. 2017. Improving Word Sense Disambiguation in Neural Machine Translation with Sense Embeddings. In *Proceedings of the Second Conference on Machine Translation*, pages 11–19, Copenhagen, Denmark. Association for Computational Linguistics.

Rico Sennrich. 2017. How Grammatical is Character-level Neural Machine Translation? Assessing MT Quality with Contrastive Translation Pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The University of Edinburgh's Neural MT Systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399, Copenhagen, Denmark. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., Montréal, Canada.

Gongbo Tang, Fabienne Cap, Eva Pettersson, and Joakim Nivre. 2018. An evaluation of neural machine translation models on historical spelling normalization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1320–1331. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The Importance of Being Recurrent for Modeling Hierarchical Structure. *arXiv preprint arXiv:1803.03585*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Yichong Xu, Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, and Zheng Zhang. 2014. Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369*.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Simplifying Neural Machine Translation with Addition-Subtraction Twin-Gated Recurrent Networks

Biao Zhang[1], **Deyi Xiong**[2], **Jinsong Su**[1]*, **Qian Lin**[1] and **Huiji Zhang**[3]
Xiamen University, Xiamen, China[1]
Soochow University, Suzhou, China[2]
Xiamen Meiya Pico information Co.,Ltd. Xiamen, China[3]
{zb,qianl}@stu.xmu.edu.cn, dyxiong@suda.edu.cn, jssu@xmu.edu.cn
zhanghj@300188.cn

Abstract

In this paper, we propose an addition-subtraction twin-gated recurrent network (ATR) to simplify neural machine translation. The recurrent units of ATR are heavily simplified to have the smallest number of weight matrices among units of all existing gated RNNs. With the simple addition and subtraction operation, we introduce a twin-gated mechanism to build input and forget gates which are highly correlated. Despite this simplification, the essential non-linearities and capability of modeling long-distance dependencies are preserved. Additionally, the proposed ATR is more transparent than LSTM/GRU due to the simplification. Forward self-attention can be easily established in ATR, which makes the proposed network interpretable. Experiments on WMT14 translation tasks demonstrate that ATR-based neural machine translation can yield competitive performance on English-German and English-French language pairs in terms of both translation quality and speed. Further experiments on NIST Chinese-English translation, natural language inference and Chinese word segmentation verify the generality and applicability of ATR on different natural language processing tasks.

1 Introduction

Neural machine translation (NMT), typically with an attention-based encoder-decoder framework (Bahdanau et al., 2015), has recently become the dominant approach to machine translation and already been deployed for online translation services (Wu et al., 2016). Recurrent neural networks (RNN), e.g., LSTMs (Hochreiter and Schmidhuber, 1997) or GRUs (Chung et al., 2014), are widely used as the encoder and decoder for NMT. In order to alleviate the gradient vanishing issue found in simple recurrent neural networks (SRNN) (Elman, 1990), recurrent units in LSTMs or GRUs normally introduce different gates to create shotcuts for gradient information to pass through.

Notwithstanding the capability of these gated recurrent networks in learning long-distance dependencies, they use remarkably more matrix transformations (i.e., more parameters) than SRNN. And with many non-linear functions modeling inputs, hidden states and outputs, they are also less transparent than SRNN. These make NMT which is based on these gated RNNs suffer from not only inefficiency in training and inference due to recurrency and heavy computation in recurrent units (Vaswani et al., 2017) but also difficulty in producing interpretable models (Lee et al., 2017). These also hinder the deployment of NMT models particularly on memory- and computation-limited devices.

In this paper, our key interest is to simplify recurrent units in RNN-based NMT. In doing so, we want to investigate how further we can advance RNN-based NMT in terms of the number of parameters (i.e., memory consumption), running speed and interpretability. This simplification shall preserve the capability of modeling long-distance dependencies in LSTMs/GRUs and the expressive power of recurrent non-linearities in SRNN. The simplification shall also reduce computation load and physical memory consumption in recurrent units on the one hand and allow us to take a good look into the inner workings of RNNs on the other hand.

In order to achieve this goal, we propose an addition-subtraction twin-gated recurrent network (ATR) for NMT. In the recurrent units of ATR, we only keep the very essential weight matrices: one over the input and the other over the history (similar to SRNN). Comparing with previous

---

*Corresponding author.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4273–4283
Brussels, Belgium, October 31 - November 4, 2018.

RNN variants (e.g., LSTM or GRU), we have the smallest number of weight matrices. This will reduce the computation load of matrix multiplication. ATR also uses gates to bypass the vanishing gradient problem so as to capture long-range dependencies. Specifically, we use the addition and subtraction operations between the weighted history and input to estimate an input and forget gate respectively. These add-sub operations not only distinguish the two gates so that we do not need to have different weight matrices for them, but also make the two gates dynamically correlate to each other. Finally, we remove some non-linearities in recurrent units.

Due to these simplifications, we can easily show that each new state in ATR is an unnormalized weighted sum of previous inputs, similar to recurrent additive networks (Lee et al., 2017). This property not only allows us to trace each state back to those inputs which contribute more but also establishes unnormalized forward self-attention between the current state and all its previous inputs. The self-attention mechanism has already proved very useful in non-recurrent NMT (Vaswani et al., 2017).

We build our NMT systems on the proposed ATR with a single-layer encoder and decoder. Experiments on WMT14 English-German and English-French translation tasks show that our model yields competitive results compared with GRU/LSTM-based NMT. When we integrate an orthogonal context-aware encoder (still single layer) into ATR-based NMT, our model (yielding 24.97 and 39.06 BLEU on English-German and English-French translation respectively) is even comparable to deep RNN and non-RNN NMT models which are all with multiple encoder/decoder layers. In-depth analyses demonstrate that ATR is more efficient than LSTM/GRU in terms of NMT training and decoding speed.

We adapt our model to other language translation and natural language processing tasks, including NIST Chinese-English translation, natural language inference and Chinese word segmentation. Our conclusions still hold on all these tasks.

2 Related Work

The most widely used RNN models are LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Chung et al., 2014), both of which are good at handling gradient vanishing problem, a notorious bottleneck of the simple RNN (Elman, 1990). The design of gates in our model follows the gate philosophy in LSTM/GRU.

Our work is closely related to the recurrent additive network (RAN) proposed by Lee et al. (2017). They empirically demonstrate that many non-linearities commonly used in RNN transition dynamics can be removed, and that recurrent hidden states computed as purely the weighted sum of input vectors can be quite efficient in language modeling. Our work follows the same spirit of simplifying recurrent units as they do. But our proposed ATR is significantly different from RAN in three aspects. First, ATR is simpler than RAN with even fewer parameters. There are only two weight matrices in ATR while four different weight matrices in the simplest version of RAN (two for each gate in RAN). Second, since the only difference between the input and forget gate in ATR is the addition/subtraction operation between the history and input, the two gates can be learned to be highly correlated as shown in our analysis. Finally, although RAN is verified effective in language modeling, our experiments show that ATR is better than RAN in machine translation in terms of both speed and translation quality.

To speed up RNN models, a line of work has attempted to remove recurrent connections. For example, Bradbury et al. (2016) propose the quasi-recurrent neural network (QRNN) which uses convolutional layers and a minimalist recurrent pooling function to improve parallelism. Very recently, Lei and Zhang (2017) propose a simple recurrent unit (SRU). With the cuDNN optimization, their RNN model can be trained as fast as CNNs. However, to obtain promising results, QRNN and SRU have to use deep architectures. In practice, 4-layer QRNN encoder and decoder are used to gain translation quality that is comparable to that of single-layer LSTM/GRU NMT. In particular, our one-layer model achieves significantly higher performance than a 10-layer SRU system.

Finally, our work is also related to the efforts in developing alternative architectures for NMT models. Zhou et al. (2016) introduce fast-forward connections between adjacent stacked RNN layers to ease gradient propagation. Wang et al. (2017a) propose a linear associate unit to reduce the gradient propagation length along layers in deep NMT. Gehring et al. (2017b) and Vaswani et al. (2017) explore purely convolutional and attentional archi-

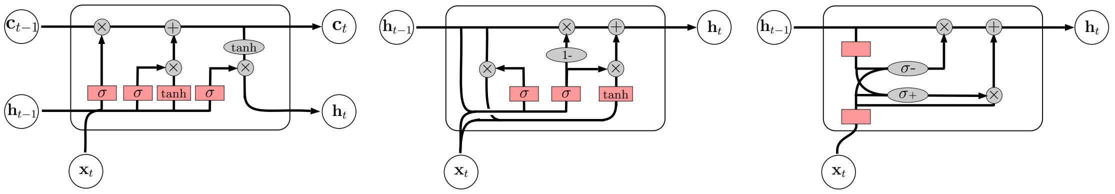

(a) LSTM (b) GRU (c) ATR

Figure 1: Architecture for LSTM, GRU and ATR. $\mathbf{c}_*$ indicates the memory cell specific to the LSTM network. $\mathbf{x}_*$ and $\mathbf{h}_*$ denote the input and output hidden states respectively.

tectures as alternatives to RNNs for neural translation. With careful configurations, their deep models achieve state-of-the-art performance on various datasets.

3 Addition-Subtraction Twin-Gated Recurrent Network

Given a sequence $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$, RNN updates the hidden state $\mathbf{h}_t$ recurrently as follows:

$$\mathbf{h}_t = \phi(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{1}$$

where $\mathbf{h}_{t-1}$ is the previous hidden state, which is considered to store information from all previous inputs, and $\mathbf{x}_t$ is the current input. The function $\phi(\cdot)$ is a non-linear recurrent function, abstracting away from details in recurrent units.

GRU can be considered as a simplified version of LSTM. In this paper, theoretically, we use GRU as our benchmark and propose a new recurrent unit to further simplify it. The GRU function is defined as follows (see Figure 1b):

$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1}) \tag{2}$$
$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1}) \tag{3}$$
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \tag{4}$$
$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t \tag{5}$$

where $\odot$ denotes an element-wise multiplication. The *reset* gate $\mathbf{r}_t$ and *update* gate $\mathbf{z}_t$ enable manageable information flow from the history and the current input to the new state respectively. Despite the success of these two gates in handling gradient flow, they consume extensive matrix transformations and weight parameters.

We argue that many of these matrix transformations are not essential. We therefore propose an addition-subtraction twin-gated recurrent unit (ATR), formulated as follows (see Figure 1c):

$$\mathbf{p}_t = \mathbf{W}_h\mathbf{h}_{t-1}, \ \mathbf{q}_t = \mathbf{W}_x\mathbf{x}_t \tag{6}$$
$$\mathbf{i}_t = \sigma(\mathbf{p}_t + \mathbf{q}_t) \tag{7}$$
$$\mathbf{f}_t = \sigma(\mathbf{p}_t - \mathbf{q}_t) \tag{8}$$
$$\mathbf{h}_t = \mathbf{i}_t \odot \mathbf{q}_t + \mathbf{f}_t \odot \mathbf{h}_{t-1} \tag{9}$$

The hidden state $\mathbf{h}_t$ in ATR is a weighted mixture of both the current input $\mathbf{q}_t$ and the history $\mathbf{h}_{t-1}$ controlled by an *input* gate $\mathbf{i}_t$ and a *forget* gate $\mathbf{f}_t$ respectively. Notice that we use the transformed representation $\mathbf{q}_t$ for the current input rather than the raw vector $\mathbf{x}_t$ due to the potential mismatch in dimensions between $\mathbf{h}_t$ and $\mathbf{x}_t$.

Similar to GRU, we use gates, especially the forget gate, to control the back-propagated gradient flow to make sure gradients will neither vanish nor explode. We also preserve the non-linearities of SRNN in ATR but only in the two gates.

There are three significant differences of ATR from GRU. Some of these differences are due to the simplifications introduced in ATR. First, we squeeze the number of weight matrices in gate calculation from four to two (see Equation (2&3) and (7&8)). In all existing gated RNNs, the inputs to gates are weighted sum of the previous hidden state and input. In order to distinguish these gates, the weight matrices over the previous hidden state and the current input should be different for different gates. The number of different weight matrices in gates is therefore 2|#gates| in previous gated RNNs. Different from them, ATR introduces different operations (i.e., addition and subtraction) between the weighted history and input to distinguish the input and forget gate. Therefore, the weight matrices over the previous state/input in the two gates can be the same in ATR. Second, we keep the very essential non-linearities, only in the two gates. In ATR, the role of $\mathbf{q}_t$ is similar to that of $\tilde{\mathbf{h}}_t$ in GRU (see Equation (4)). However, we completely wipe out the recurrent non-linearity

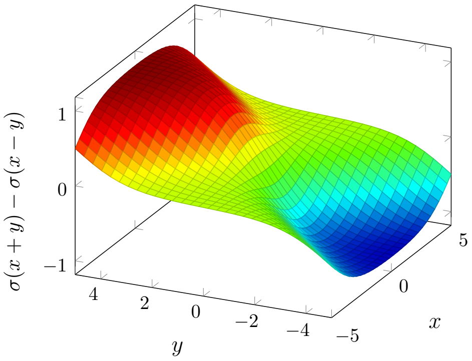

Figure 2: Visualization of the difference between $\sigma(x+y)$ (the input gate) and $\sigma(x-y)$ (the forget gate). Here, we set $x, y \in [-5, 5]$.

of $\tilde{\mathbf{h}}_t$ in $\mathbf{q}_t$ (i.e., $\mathbf{q}_t = \mathbf{W}_x\mathbf{x}_t$). Lee et al. (2017) show that this non-linearity is not necessary in language modeling. We further empirically demonstrate that it is neither essential in machine translation. Third, in GRU the gates for $\tilde{\mathbf{h}}_t$ and $\mathbf{h}_{t-1}$ are coupled and normalized to 1 while we do not explicitly associate the two gates in ATR. Instead, they can be learned to be correlated in an implicit way, as shown in the next subsection and our empirical analyis in Section 5.1.

3.1 Twin-Gated Mechanism

Unlike GRU, we use an addition and subtraction operation over the transformed current input $\mathbf{q}_t$ and history $\mathbf{p}_t$ to differentiate the two gates in ATR. As the two gates have the same weights for their input components with only a single difference in the operation between the input components, they act like twins. We term the two gates in ATR as twin gates and the procedure, shown in Equation (7&8), as the *twin-gated mechanism*. This mechanism endows our model with the following two advantages: 1) Both addition and subtraction operations are completely linear so that fast computation can be expected; and 2) No other weight parameters are introduced for gates so that our model is more memory-compact.

A practical question for the twin-gated mechanism is whether twin gates are really capable of dynamically weighting the input and history information. To this end, we plot the surface of one-dimensional $\sigma(x+y) - \sigma(x-y)$ in Figure 2. It is clear that both gates are highly non-linearly correlated, and that there are regions where $\sigma(x+y)$ is equal to, greater or smaller than $\sigma(x-y)$. In other words, by adapting the distribution of input and forget gates, the twin-gated mechanism has the potential to automatically seek suitable regions in Figure 2 to control its preference between the new and past information. We argue that the input and forget gates are negatively correlated after training, and empirically show their actual correlation in Section 5.1.

Model	**# WM**	**# MT**
LSTM	8	8
GRU	6	6
RAN	4	4
ATR	2	2

Table 1: Comparison of LSTM, GRU, RAN and ATR in terms of the number of weight matrices (WM) and matrix transformations (MT).

3.2 Computation Analysis

Here we provide a systematical comparison of computations in LSTM, GRU, RAN and our ATR with respect to the number of weight matrices and matrix transformations. Notice that all these units are building blocks of RNNs so that the total computational complexity and the minimum number of sequential operations required are unchanged, i.e. $\mathcal{O}(n \cdot d^2)$ and $\mathcal{O}(n)$ respectively where n is the sequence length and d is the dimensionality of hidden states. However, the actual number of matrix transformations in the unit indeed significantly affects the running speed of RNN in practice.

We summarize the results in Table 1. LSTM contains three different gates and a cell state, including 4 different neural layers with 8 weight matrices and transformations. GRU simplifies LSTM by removing a gate, but still involves two gates and a candidate hidden state. It includes 3 different neural layers with 6 weight matrices and transformations. RAN further simplifies GRU by removing the non-linearity in the state transition and therefore contains 4 weight matrices in its simplest version. Although our ATR also has two gates, however, there are only 2 weight matrices and transformations, accounting for only a third and a quarter of those in GRU and LSTM respectively. To the best of our knowledge, ATR has the smallest number of weight transformations in existing gated RNN units. We provide a detailed and empirical analysis on the speed in Section 5.2.

3.3 Interpretability Analysis of Hidden States

An appealing property of the proposed ATR is its interpretability. This can be demonstrated by rolling out Equation (9) as follows:

$$\begin{aligned}\mathbf{h}_t &= \mathbf{i}_t \odot \mathbf{q}_t + \mathbf{f}_t \odot \mathbf{h}_{t-1} \\ &= \mathbf{i}_t \odot \mathbf{W}_t \mathbf{x}_t + \sum_{k=1}^{t-1} \mathbf{i}_k \odot \left(\prod_{l=1}^{t-k} \mathbf{f}_{k+l} \right) \odot \mathbf{W}_x \mathbf{x}_k \\ &\approx \sum_{k=1}^{t} \mathbf{g}_k \odot \mathbf{W}_x \mathbf{x}_k \end{aligned} \tag{10}$$

where $\mathbf{g}_k$ can be considered as an approximate weight assigned to the k-th input. Similar to the RAN model (Lee et al., 2017), the hidden state in ATR is a component-wise weighted sum of the inputs. This not only enables ATR to build up essential dependencies between preceding inputs and the current hidden state, but also allows us to easily detect which previous words have the promising impacts on the current state. This desirable property obviously makes ATR highly interpretable.

Additionally, this form of weighted sum is also related to self-attention (Vaswani et al., 2017). It can be considered as a forward unnormalized self-attention where each hidden state attends to all its previous positions. As the self-attention mechanism has proved very useful in NMT (Vaswani et al., 2017), we conjecture that such property of ATR partially contributes to its success in machine translation as shown in our experiments. We visualize the dependencies captured by Equation (10) in Section 5.3.

4 Experiments

4.1 Setup

We conducted our main experiments on WMT14 English-German and English-French translation tasks. Translation quality is measured by case-sensitive BLEU-4 metric (Papineni et al., 2002). Details about each dataset are as follows:

English-German To compare with previous reported results (Luong et al., 2015b; Jean et al., 2015; Zhou et al., 2016; Wang et al., 2017a), we used the same training data of WMT 2014, which consist of 4.5M sentence pairs. We used the newstest2013 as our dev set, and the newstest2014 as our test set.

English-French We used the WMT 2014 training data. This corpora contain 12M selected sentence pairs. We used the concatenation of newstest2012 and newstest2013 as our dev set, and the newstest2014 as our test set.

The used NMT system is an attention-based encoder-decoder system, which employs a bidirectional recurrent network as its encoder and a two-layer hierarchical unidirectional recurrent network as its decoder, companied with an additive attention mechanism (Bahdanau et al., 2015). We replaced the recurrent unit with our proposed ATR model. More details are given in Appendix A.1.

We also conducted experiments on Chinese-English translation, natural language inference and Chinese word segmentation. Details and experiment results are provided in Appendix A.2.

4.2 Training

We set the maximum length of training instances to 80 words for both English-German and English-French task. We used the *byte pair encoding* compression algorithm (Sennrich et al., 2016) to reduce the vocabulary size as well as to deal with the issue of rich morphology. We set the vocabulary size of both source and target languages to 40K for all translation tasks. All out-of-vocabulary words were replaced with a token "*unk*".

We used 1000 hidden units for both encoder and decoder. All word embeddings had dimensionality 620. We initialized all model parameters randomly according to a uniform distribution ranging from -0.08 to 0.08. These tunable parameters were then optimized using Adam algorithm (Kingma and Ba, 2015) with the two momentum parameters set to 0.9 and 0.999 respectively. Gradient clipping 5.0 was applied to avoid the gradient explosion problem. We trained all models with a learning rate $5e^{-4}$ and batch size 80. We decayed the learning rate with a factor of 0.5 between each training epoch. Translations were generated by a beam search algorithm that was based on log-likelihood scores normalized by sentence length. We used a beam size of 10 in all the experiments. We also applied dropout for English-German and English-French tasks on the output layer to avoid over-fitting, and the dropout rate was set to 0.2.

To train deep NMT models, we adopted the GNMT architecture (Wu et al., 2016). We kept all the above settings, except the dimensionality

System	Architecture	Vocab	*tok* BLEU	*detok* BLEU
Buck et al. (2014)	WMT14 winner system phrase-based + large LM	-	-	20.70
Existing deep NMT systems (perhaps different tokenization)				
Zhou et al. (2016)	LSTM with 16 layers + F-F connections	160K	20.60	-
Lei and Zhang (2017)	SRU with 10 layers	50K	20.70	-
Antonino and Federico (2018)	SR-NMT with 4 layers	32K	23.32	-
Wang et al. (2017a)	GRU with 4 layers + LAU + PosUnk	80K	23.80	-
Wang et al. (2017a)	GRU with 4 layers + LAU + PosUnk + ensemble	80K	26.10	-
Wu et al. (2016)	LSTM with 8 layers + RL-refined WPM	32K	24.61	-
Wu et al. (2016)	LSTM with 8 layers + RL-refined ensemble	80K	26.30	-
Vaswani et al. (2017)	Transformer with 6 layers + base model	37K	27.30	-
Comparable NMT systems (the same tokenization)				
Luong et al. (2015a)	LSTM with 4 layers + local att. + unk replace	50K	20.90	-
Zhang et al. (2017a)	GRU with gated attention + BPE	40K	23.84	-
Gehring et al. (2017b)	CNN with 15 layers + BPE	40K	25.16	-
Gehring et al. (2017b)	CNN with 15 layers + BPE + ensemble	40K	26.43	-
Zhang et al. (2018a)	Transformer with 6 layers + aan + base model	32K	26.31	-
Our end-to-end NMT systems				
this work	RNNSearch + GRU + BPE	40K	22.54	22.06
	RNNSearch + LSTM + BPE	40K	22.96	22.39
	RNNSearch + RAN + BPE	40K	22.14	21.40
	RNNSearch + ATR + BPE	40K	22.48	21.99
	RNNSearch + ATR + CA + BPE	40K	23.31	22.70
	GNMT + ATR + BPE	40K	24.16	23.59
	RNNSearch + ATR + CA + BPE + ensemble	40K	24.97	24.33

Table 2: Tokenized (*tok*) and detokenized (detok) case-sensitive BLEU scores on the WMT14 English-German translation task. "*unk replace*" and "*PosUnk*" denotes the approach of handling rare words in Jean et al. (2015) and Luong et al. (2015a) respectively. "*RL*" and "*WPM*" is the reinforcement learning optimization and word piece model used in Wu et al. (2016). "*CA*" is the context-aware recurrent encoder (Zhang et al., 2017b). "*LAU*" and "*F-F*" denote the linear associative unit and the fast-forward architecture proposed by Wang et al. (2017a) and Zhou et al. (2016) respectively. "*aan*" denotes the average attention network proposed by Zhang et al. (2018a).

of word embedding and hidden state which we set to be 512.

4.3 Results on English-German Translation

The translation results are shown in Table 2. We also provide results of several existing systems that are trained with comparable experimental settings to ours. In particular, our single model yields a detokenized BLEU score of 21.99. In order to show that the proposed model can be orthogonal to previous methods that improve LSTM/GRU-based NMT, we integrate a single-layer context-aware (CA) encoder (Zhang et al., 2017b) into our system. The ATR+CA system further reaches 22.7 BLEU, outperforming the winner system (Buck et al., 2014) by a substantial improvement of 2 BLEU points. Enhanced with the deep GNMT architecture, the GNMT+ATR system yields a gain of 0.89 BLEU points over the RNNSearch+ATR+CA and 1.6 BLEU points over the RNNSearch + ATR. Notice that different from our system which was trained on the parallel corpus alone, the winner system used a huge monolingual text to enhance its language model.

Compared with the existing LSTM-based (Luong et al., 2015a) deep NMT system, our shallow/deep model achieves a gain of 2.41/3.26 tokenized BLEU points respectively. Under the same training condition, our ATR outperforms RAN by a margin of 0.34 tokenized BLEU points, and achieves competitive results against its GRU/LSTM counterpart. This suggests that although our ATR is much simpler than GRU, LSTM and RAN, it still possesses strong modeling capacity.

In comparison to several advanced deep NMT models, such as the Google NMT (8 layers, 24.61 tokenized BLEU) (Wu et al., 2016) and the LAU-connected NMT (4 layers, 23.80 tokenized BLEU) (Wang et al., 2017a), the performance of our shallow model (23.31) is competitive. Particularly, when replacing LSTM in the Google NMT with our ATR model, the GNMT+ATR system achieves a BLEU score of 24.16, merely 0.45 BLEU points lower. Notice that although all systems use the same training data of WMT14, the

System	Architecture	Vocab	*tok* BLEU	*detok* BLEU
	Existing end-to-end NMT systems			
Jean et al. (2015)	RNNSearch (GRU) + unk replace + large vocab	500K	34.11	-
Luong et al. (2015b)	LSTM with 6 layers + PosUnk	40K	32.70	-
Sutskever et al. (2014)	LSTM with 4 layers	80K	30.59	-
Shen et al. (2016)	RNNSearch (GRU) + MRT + PosUnk	30K	34.23	-
Zhou et al. (2016)	LSTM with 16 layers + F-F connections + 36M data	80K	37.70	-
Wu et al. (2016)	LSTM with 8 layers + RL-refined WPM + 36M data	32K	38.95	-
Wang et al. (2017a)	RNNSearch (GRU) with 4 layers + LAU	30K	35.10	-
Gehring et al. (2017a)	Deep Convolutional Encoder 20 layers with kernel width 5	30K	35.70	-
Vaswani et al. (2017)	Transformer with 6 layers + 36M data + base model	32K	38.10	-
Gehring et al. (2017b)	ConvS2S with 15 layers + 36M data	40K	40.46	-
Vaswani et al. (2017)	Transformer with 6 layers + 36M data + big model	32K	41.80	-
Wu et al. (2016)	LSTM with 8 layers + RL WPM + 36M data + ensemble	32K	41.16	-
	Our end-to-end NMT systems			
this work	RNNSearch + GRU + BPE	40K	35.89	33.41
	RNNSearch + LSTM + BPE	40K	36.95	34.15
	RNNSearch + ATR + BPE	40K	36.89	34.00
	RNNSearch + ATR + CA + BPE	40K	37.88	34.96
	GNMT + ATR + BPE	40K	38.59	35.67
	RNNSearch + ATR + CA + BPE + ensemble	40K	39.06	36.06

Table 3: Tokenized (*tok*) and detokenized (detok) case-sensitive BLEU scores on the WMT14 English-French translation task. "*12M data*" indicates the same training data as ours, while "*36M data*" is a significant larger dataset that contains the *12M data*.

tokenization of these work might be different from ours. However, the overall results can indicate the competitive strength of our model. In addition, SRU (Lei and Zhang, 2017), a recent proposed efficient recurrent unit, obtains a BLEU score of 20.70 with 10 layers, far more behind ATR's.

We further ensemble eight likelihood-trained models with different random initializations for the ATR+CA system. The variance in the tokenized BLEU scores of these models is 0.07. As can be seen from Table 2, the ensemble system achieves a tokenized and detokenized BLEU score of 24.97 and 24.33 respectively, obtaining a gain of 1.66 and 1.63 BLEU points over the single model. The final result of the ensemble system, to the best of our knowledge, is a very promising result that can be reached by single-layer NMT systems on WMT14 English-German translation.

4.4 Results on English-French Translation

Unlike the above translation task, the WMT14 English-French translation task provides a significant larger dataset. The full training data have approximately 36M sentence pairs, from which we only used 12M instances for experiments following previous work (Jean et al., 2015; Gehring et al., 2017a; Luong et al., 2015b; Wang et al., 2017a). We show the results in Table 3.

Our shallow model achieves a tokenized BLEU score of 36.89 and 37.88 when it is equipped with the CA encoder, outperforming almost all the listed systems, except the Google NMT (Wu et al., 2016), the ConvS2S (Gehring et al., 2017b) and the Transformer (Vaswani et al., 2017). Enhanced with the deep GNMT architecture, the GNMT+ATR system reaches a BLEU score of 38.59, which beats the base model version of the Transformer by a margin of 0.49 BLEU points. When we use four ensemble models (the variance in the tokenized BLEU scores of these ensemble models is 0.16), the ATR+CA system obtains another gain of 0.47 BLEU points, reaching a tokenized BLEU score of 39.06, which is comparable with several state-of-the-art systems.

5 Analysis

5.1 Analysis on Twin-Gated Mechanism

We provide an illustration of the actual relation between the learned input and forget gate in Figure 3. Clearly, these two gates show strong negative correlation. When the input gate opens with high values, the forget gate prefer to be close. Quantitatively, on the whole test set, the Pearson's r of the input and forget gate is -0.9819, indicating a high correlation.

5.2 Analysis on Speed and Model Parameters

As mentioned in Section 3.2, ATR has much fewer model parameters and matrix transformations. We

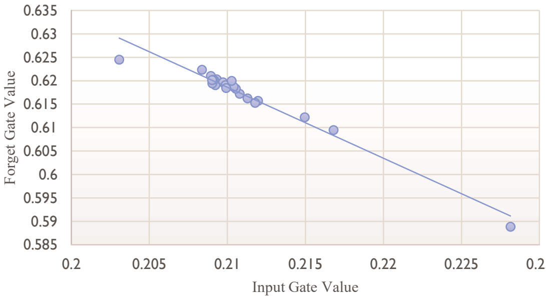

Figure 3: Visualization of correlation between the input and forget gate learned in Equation (13). For each gate, we record its mean value $(\frac{1}{d_h}\sum_i \mathbf{i}_{t,i}/\frac{1}{d_h}\sum_i \mathbf{f}_{t,i})$ with its position on newstest2014, and then average all gate values for each position over all test sentences. This figure illustrates the position-wise input and forget gate values.

Model	#PMs	Train	Test
RNNSearch+GRU	83.5M	1996	168 (0.133)
RNNSearch+LSTM	93.3M	1919	167 (0.139)
RNNSearch+RAN	79.5M	2192	170 (0.129)
DeepRNNSearch	143.0M	894	70 (0.318)
Transformer	70.2M	4961	44 (0.485)
RNNSearch+ATR	67.8M	2518	177 (0.123)
RNNSearch+ATR+CA	63.1M	3993	186 (0.118)

Table 4: Comparison on the training and decoding speed and the number of model parameters of different NMT models on WMT14 English-German translation task with beam size 1. **#PMs**: the number of model parameters. **Train/Test**: the number of words in one second processed during training/testing. The number in bracket indicates the average decoding time per source sentence (in seconds).

provide more details in this section by comparing against the following two NMT systems:

- *DeepRNNSearch (GRU)*: a deep GRU-equipped RNNSearch model (Wu et al., 2016) with 5 layers. We set the dimension of word embedding and hidden state to 620 and 1000 respectively.
- *Transformer*: a purely attentional translator (Vaswani et al., 2017). We set the dimension of word embedding and filter size to 512 and 2048 respectively. The model was trained with a minibatch size of 256.

We also compare with the GRU and LSTM-based RNNSearch. Without specific mention, all other experimental settings for all these models are the same as for our model. We implement all these models using the *Theano* library, and test the speed on one GeForce GTX TITAN X GPU card. We show the results on Table 4.

We observe that the Transformer achieves the best training speed, processing 4961 words per second. This is reasonable since the Transformer can be trained in full parallelization. On the contrary, DeepRNNSearch is the slowest system. As RNN performs sequentially, stacking more layers of RNNs inevitably reduces the training efficiency. However, this situation becomes the reverse when it comes to the decoding procedure. The Transformer merely generates 44 words per second while DeepRNNSearch reaches 70. This is because during decoding, all these beam search-based systems must generate translation one word after another. Therefore the parallelization advantage of the Transformer disappears. In comparison to DeepRNNSearch, the Transformer spends extra time on performing self-attention over all previous hidden states.

Our model with the CA structure, using only 63.1M parameters, processes 3993 words per second during training and generates 186 words per second during decoding, which yields substantial speed improvements over the GRU- and LSTM-equipped RNNSearch. This is due to the light matrix computation in recurrent units of ATR. Notice that the speed increase of ATR over GRU and LSTM does not reach 3x. This is because at each decoding step, there are mainly two types of computation: recurrent unit and softmax layer. The latter consumes the most calculation, which, however, is the same for different models (LSTM/GRU/ATR).

5.3 Analysis on Dependency Modeling

As shown in Section 3.3, a hidden state in our ATR can be formulated as a weighted sum of the previous inputs. In this section, we quantitatively analyze the weights $\mathbf{g}_k$ in Equation (10) induced from Equation (13). Inspired by Lee et al. (2017), we visualize the captured dependencies of an example in Figure 4 where we connect each word to the corresponding previous word with the highest weight $\mathbf{g}_k$.

Obviously, our model can discover strong local dependencies. For example, the token "*unglück@@*" and "*lichen*" should be a

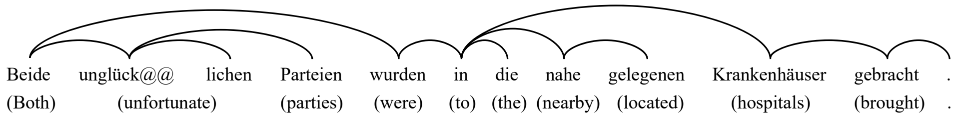

Figure 4: Visualization of dependencies on a target (German) sentence (selected from newstest2014). "@@" indicates a separator that splits one token into two pieces of sub-words.

single word. Our model successfully associates "*unglück@@*" closely to the generation of "*lichen*" during decoding. In addition, our model can also detect non-consecutive long-distance dependencies. Particularly, the prediction of "*Parteien*" relies heavily on the token "*unglücklichen*", which actually entails an *amod* linguistic dependency relationship. These captured dependencies make our model more interpretable than LSTM/GRU.

6 Conclusion and Future Work

This paper has presented a twin-gated recurrent network (ATR) to simplify neural machine translation. There are only two weight matrices and matrix transformations in recurrent units of ATR, making it efficient in physical memory usage and running speed. To avoid the gradient vanishing problem, ATR introduces a twin-gated mechanism to generate an input gate and forget gate through linear addition and subtraction operation respectively, without introducing any additional parameters. The simplifications allow ATR to produce interpretable results.

Experiments on English-German and English-French translation tasks demonstrate the effectiveness of our model. They also show that ATR can be orthogonal to and applied with methods that improve LSTM/GRU-based NMT, indicated by the promising performance of the ATR+CA system. Further analyses reveal that ATR can be trained more efficiently than GRU. It is also able to transparently model long-distance dependencies.

We also adapt our ATR to other natural language processing tasks. Experiments show encouraging performance of our model on Chinese-English translation, natural language inference and Chinese word segmentation, demonstrating its generality and applicability on various NLP tasks.

In the future, we will continue to examine the effectiveness of ATR on different neural models for NMT, such as the hierarchical NMT model (Su et al., 2018b) as well as the generative NMT model (Su et al., 2018a). We are also interested in adapting our ATR to summarization, semantic parsing etc.

Acknowledgments

The authors were supported by National Natural Science Foundation of China (Grants No. 61672440, 61622209 and 61861130364), the Fundamental Research Funds for the Central Universities (Grant No. ZK1024), and Scientific Research Project of National Language Committee of China (Grant No. YB135-49). Biao Zhang greatly acknowledges the support of the Baidu Scholarship. We also thank the reviewers for their insightful comments.

References

M. Antonino and M. Federico. 2018. Deep Neural Machine Translation with Weakly-Recurrent Units. *ArXiv e-prints*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proc. of EMNLP*. Association for Computational Linguistics.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proc. of ACL*, pages 1466–1477.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *CoRR*, abs/1611.01576.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the common crawl. In *Proc. of LREC*, pages 3579–3584, Reykjavik, Iceland.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proc. of EMNLP*, pages 1197–1206.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proc. of EMNLP*, pages 551–561.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. 2017a. A convolutional encoder model for neural machine translation. In *Proc. of ACL*, pages 123–135.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017b. Convolutional sequence to sequence learning.

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

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proc. of ACL-IJCNLP*, pages 1–10.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Proc. of ICLR*.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. *CoRR*, abs/1705.07393.

T. Lei and Y. Zhang. 2017. Training RNNs as Fast as CNNs. *ArXiv e-prints*.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*, pages 1412–1421.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proc. of ACL-IJCNLP*, pages 11–19.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proc. of ACL*, pages 293–303, Baltimore, Maryland. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proc. of ICLR*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*, pages 1715–1725.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proc. of ACL*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

Richard Sproat and Thomas Emerson. 2003. The first international chinese word segmentation bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing - Volume 17*, SIGHAN '03, pages 133–143.

Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. 2018a. Variational recurrent neural machine translation. *arXiv preprint arXiv:1801.05119*.

Jinsong Su, Jiali Zeng, Deyi Xiong, Yang Liu, Mingxuan Wang, and Jun Xie. 2018b. A hierarchy-to-sequence attentional neural machine translation model. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(3):623–632.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *CoRR*, abs/1511.06361.

M. Wang, Z. Lu, J. Zhou, and Q. Liu. 2017. Deep Neural Machine Translation with Linear Associative Unit. *ArXiv e-prints*.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017a. Deep neural machine translation with linear associative unit. In *Proc. of ACL*, pages 136–145, Vancouver, Canada.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proc. of NAACL*, pages 1442–1451.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017b. Bilateral multi-perspective matching for natural language sentences. *CoRR*, abs/1702.03814.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2017a. A gru-gated attention model for neural machine translation. *CoRR*, abs/1704.08430.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018a. Accelerating neural transformer via an average attention network. In *Proc of ACL*, pages 1789–1798. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, Jinsong Su, and Hong Duan. 2017b. A context-aware recurrent encoder for neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, PP(99):1–1.

Jinchao Zhang, Mingxuan Wang, Qun Liu, and Jie Zhou. 2017c. Incorporating word reordering knowledge into attention-based neural machine translation. In *Proc. of ACL*, pages 1524–1534, Vancouver, Canada. Association for Computational Linguistics.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018b. Asynchronous bidirectional decoding for neural machine translation. *CoRR*, abs/1801.05122.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *Pro. of EMNLP*, pages 647–657.

Zaixiang Zheng, Hao Zhou, Shujian Huang, Lili Mou, Xinyu Dai, Jiajun Chen, and Zhaopeng Tu. 2017. Modeling past and future for neural machine translation. *CoRR*, abs/1711.09502.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383.

Speeding Up Neural Machine Translation Decoding by Cube Pruning

Wen Zhang[1,2] **Liang Huang**[3,4] **Yang Feng**[1,2] **Lei Shen**[1,2] **Qun Liu**[5,1]

[1]Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[2]University of Chinese Academy of Sciences, Beijing, China
{zhangwen,fengyang,shenlei17z}@ict.ac.cn

[3]Oregon State University, Corvallis, OR, USA [4]Baidu Research, Sunnyvale, CA, USA
liang.huang.sh@gmail.com

[5]Huawei Noah's Ark Lab, Hong Kong, China
qun.liu@huawei.com

Abstract

Although neural machine translation has achieved promising results, it suffers from slow translation speed. The direct consequence is that a trade-off has to be made between translation quality and speed, thus its performance can not come into full play. We apply cube pruning, a popular technique to speed up dynamic programming, into neural machine translation to speed up the translation. To construct the equivalence class, similar target hidden states are combined, leading to less RNN expansion operations on the target side and less $\mathrm{softmax}$ operations over the large target vocabulary. The experiments show that, at the same or even better translation quality, our method can translate faster compared with naive beam search by $3.3\times$ on GPUs and $3.5\times$ on CPUs.

1 Introduction

Neural machine translation (NMT) has shown promising results and drawn more attention recently (Kalchbrenner and Blunsom, 2013; Cho et al., 2014b; Bahdanau et al., 2015; Gehring et al., 2017a,b; Vaswani et al., 2017). A widely used architecture is the attention-based encoder-decoder framework (Cho et al., 2014b; Bahdanau et al., 2015) which assumes there is a common semantic space between the source and target language pairs. The encoder encodes the source sentence to a representation in the common space with the recurrent neural network (RNN) (Hochreiter and Schmidhuber, 1997) and the decoder decodes this representation to generate the target sentence word by word. To generate a target word, a probability distribution over the target vocabulary is drawn based on the attention over the entire source sequence and the target information rolled by another RNN. At the training time, the decoder is forced to generate the ground truth sentence, while at inference, it needs to employ the beam search algorithm to search through a constrained space due to the huge search space.

Even with beam search, NMT still suffers from slow translation speed, especially when it works not on GPUs, but on CPUs, which are more common practice. The first reason for the inefficiency is that the generation of each target word requires extensive computation to go through all the source words to calculate the attention. Worse still, due to the recurrence of RNNs, target words can only be generated *sequentially* rather than in parallel. The second reason is that large vocabulary on target side is employed to avoid unknown words (UNKs), which leads to a large number of normalization factors for the $\mathrm{softmax}$ operation when drawing the probability distribution. To accelerate the translation, the widely used method is to trade off between the translation quality and the decoding speed by reducing the size of vocabulary (Mi et al., 2016a) or/and the number of parameters, which can not realize the full potential of NMT.

In this paper, we borrow ideas from phrase-based and syntax-based machine translation where cube pruning has been successfully applied to speed up the decoding (Chiang, 2007; Huang and Chiang, 2007). Informally, cube pruning "coarsens" the search space by clustering similar states according to some equivalence relations. To apply this idea to NMT, however, is much more involved. Specifically, in the process of beam search, we cluster similar target hidden states to construct equivalence classes, the three dimensions of which are target words in the target vocabulary, part translations retained in the beam search and different combinations of similar target hidden states, respectively. The clustering operation

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4284–4294
Brussels, Belgium, October 31 - November 4, 2018.

can directly decrease the number of target hidden states in the following calculations, together with cube pruning, resulting in less RNN expansion operations to generate the next hidden state (related to the first reason) and less softmax operations over the target vocabulary (related to the second reason). The experiment results show that, when receiving the same or even better translation quality, our method can speed up the decoding speed by 3.3× on GPUs and 3.5× on CPUs.

2 Background

The proposed strategy can be adapted to optimize the beam search algorithm in the decoder of various NMT models. Without loss of generality, we take the attention-based NMT (Bahdanau et al., 2015) as an example to introduce our method. In this section, we first introduce the attention-based NMT model and then the cube pruning algorithm.

2.1 The Attention-based NMT Model

The attention-based NMT model follows the encoder-decoder framework with an extra attention module. In the following parts, we will introduce each of the three components. Assume the source sequence and the observed translation are $\mathbf{x} = \{x_1, \cdots, x_{|\mathbf{x}|}\}$ and $\mathbf{y} = \{y_1^*, \cdots, y_{|\mathbf{y}|}^*\}$.

Encoder The encoder uses a bidirectional GRU to obtain two sequences of hidden states. The final hidden state of each source word is got by concatenating the corresponding pair of hidden states in those sequences. Note that e_{x_i} is employed to represent the embedding vector of the word x_i.

$$\overrightarrow{h}_i = \overrightarrow{\mathbf{GRU}}\left(e_{x_i}, \overrightarrow{h}_{i-1}\right) \quad (1)$$

$$\overleftarrow{h}_i = \overleftarrow{\mathbf{GRU}}\left(e_{x_i}, \overleftarrow{h}_{i+1}\right) \quad (2)$$

$$h_i = \left[\overrightarrow{h}_i; \overleftarrow{h}_i\right] \quad (3)$$

Attention The attention module is designed to extract source information (called context vector) which is highly related to the generation of the next target word. At the j-th step, to get the context vector, the relevance between the target word y_j^* and the i-th source word is firstly evaluated as

$$r_{ij} = \mathbf{v}_a^T \tanh\left(\mathbf{W}_a s_{j-1} + \mathbf{U}_a h_i\right) \quad (4)$$

Then, the relevance is normalized over the source sequence, and all source hidden states are added weightedly to produce the context vector.

$$\alpha_{ij} = \frac{\exp\left(r_{ij}\right)}{\sum_{i'=1}^{|\mathbf{x}|} \exp\left(r_{i'j}\right)}; \; c_j = \sum\nolimits_{i=1}^{|\mathbf{x}|} \alpha_{ij} h_i \quad (5)$$

Decoder The decoder also employs a GRU to unroll the target information. The details are described in Bahdanau et al. (2015). At the j-th decoding step, the target hidden state s_j is given by

$$s_j = f\left(e_{y_{j-1}^*}, s_{j-1}, c_j\right) \quad (6)$$

The probability distribution $\mathcal{D}_j$ over all the words in the target vocabulary is predicted conditioned on the previous ground truth words, the context vector c_j and the unrolled target information s_j.

$$t_j = g\left(e_{y_{j-1}^*}, c_j, s_j\right) \quad (7)$$

$$o_j = \mathbf{W}_o t_j \quad (8)$$

$$\mathcal{D}_j = \mathrm{softmax}\left(o_j\right) \quad (9)$$

where g stands for a linear transformation, $\mathbf{W}_o$ is used to map t_j to o_j so that each target word has one corresponding dimension in o_j.

2.2 Cube Pruning

The cube pruning algorithm, proposed by Chiang (2007) based on the k-best parsing algorithm of Huang and Chiang (2005), is actually an accelerated extension based on the naive beam search algorithm. Beam search, a heuristic dynamic programming searching algorithm, explores a graph by expanding the most promising nodes in a limited set and searches approximate optimal results from candidates. For the sequence-to-sequence learning task, given a pre-trained model, the beam search algorithm finds a sequence that approximately maximizes the conditional probability (Graves, 2012; Boulanger-Lewandowski et al., 2013). Both Sutskever et al. (2014) and Bahdanau et al. (2015) employed the beam search algorithm into the NMT decoding to produce translations with relatively larger conditional probability with respect to the optimized model parameters. Remarkably, Huang and Chiang (2007) successfully applied the cube pruning algorithm to the decoding of SMT. They found that the beam search algorithm in SMT can be extended, and they utilized the cube pruning and some variants to optimize the search process in the decoding phase of phrase-based (Och and Ney, 2004) and syntax-based (Chiang, 2005; Galley et al., 2006) systems,

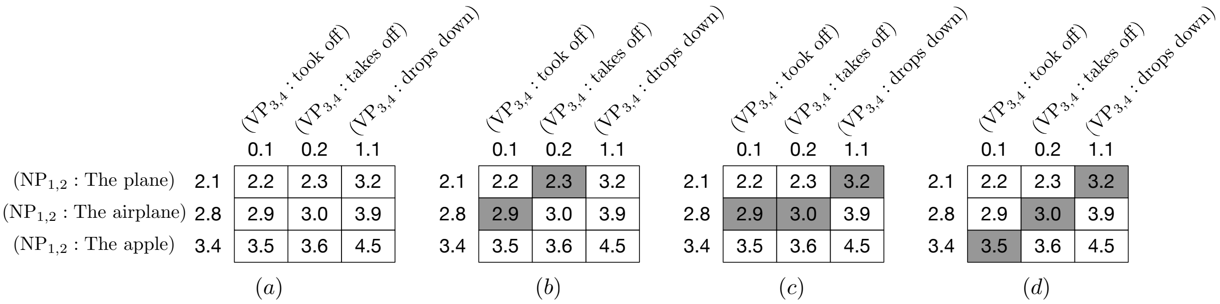

Figure 1: Cube pruning in SMT decoding. (a): the values in the grid denote the negative log-likelihood cost of the terminal combinations on both dimensions, and each dimension denotes a translation candidate in this example; (b)-(d): the process of popping the best candidate of top three items.

	GPU		**CPU**	
Calculation Units	**Time(s)**	**Percentage**	**Time(s)**	**Percentage**
Eq. (6): $s_j = f(e_{y^*_{j-1}}, s_{j-1}, c_j)$	551.07	75.73%	1370.92	19.42%
Eq. (7): $t_j = g(e_{y^*_{j-1}}, c_j, s_j)$	88.25	12.13%	277.76	3.93%
Eq. (8): $o_j = \mathbf{W}_o t_j$	25.33	3.48%	2342.53	33.18%
Eq. (9): $\mathcal{D}_j = \mathrm{softmax}(o_j)$	63.00	8.66%	3069.25	43.47%

Table 1: Time cost statistics for decoding the whole MT03 testset on GPUs and CPUs with beam size 10.

which decreased a mass of translation candidates and achieved a significant speed improvement by reducing the size of complicated search space, thereby making it possible to actualize the thought of improving the translation performance through increasing the beam size.

In the traditional SMT decoding, the cube pruning algorithm aims to prune a great number of partial translation hypotheses without computing and storing them. For each decoding step, those hypotheses with the same translation rule are grouped together, then the cube pruning algorithm is conducted over the hypotheses. We illustrate the detailed process in Figure 1.

3 NMT Decoder with Cube Pruning

3.1 Definitions

We define the related storage unit tuple of the i-th candidate word in the j-th beam as $n^i_j = (c^i_j, s^i_j, y^i_j, bp^i_j)$, where c^i_j is the negative log-likelihood (NLL) accumulation in the j-th beam, s^i_j is the decoder hidden state in the j-th beam, y^i_j is the index of the j-th target word in large vocabulary and bp^i_j is the backtracking pointer for the j-th decoding step. Note that, for each source sentence, we begin with calculating its encoded representation and the first hidden state s^0_0 in decoder, then searching from the initial tuple $(0.0, s^0_0, 0, 0)$ existing in the first beam[1].

It is a fact that Equation (9) produces the probability distribution of the predicted target words over the target vocabulary V. Cho et al. (2014b) indicated that whenever a target word is generated, the $\mathrm{softmax}$ function over V computes probabilities for all words in V, so the calculation is expensive when the target vocabulary is large. As such, Bahdanau et al. (2015) (and many others) only used the top-$30k$ frequent words as target vocabulary, and replaced others with UNK. However, the final normalization operation still brought high computation complexity for forward calculations.

3.2 Time Cost in Decoding

We conducted an experiment to explore how long each calculation unit in the decoder would take. We decoded the MT03 test dataset by using naive beam search with beam size of 10 and recorded the time consumed in the computation of Equation (6), (7), (8) and (9), respectively. The statistical results in Table 1 show that the recurrent calculation unit consumed the most time on GPUs, while

---

[1] The initial target word index y^0_0 equals to 0, which actually corresponds to the Beginning Of Sentence (BOS) token in target vocabulary.

the $\mathrm{softmax}$ computation also took lots of time. On CPUs, the most expensive computational time cost was caused by the $\mathrm{softmax}$ operation over the entire target vocabulary[2]. In order to avoid the time-consuming normalization operation in testing, we introduced *self-normalization* (denoted as **SN**) into the training.

3.3 Self-normalization

Self-normalization (Devlin et al., 2014) was designed to make the model scores which are produced by the output layer be approximated by the probability distribution over the target vocabulary without normalization operation. According to Equation (9), for an observed target sentence $\mathbf{y} = \{y_1^*, \cdots, y_{|\mathbf{y}|}^*\}$, the Cross-Entropy (CE) loss could be written as

$$\begin{aligned} \mathrm{L}_\theta &= -\sum_{j=1}^{|\mathbf{y}|} \log \mathcal{D}_j[y_j^*] \\ &= -\sum_{j=1}^{|\mathbf{y}|} \log \frac{\exp\left(o_j[y_j^*]\right)}{\sum_{y' \in V} \exp\left(o_j[y']\right)} \\ &= \sum_{j=1}^{|\mathbf{y}|} \log \sum_{y' \in V} \exp\left(o_j[y']\right) - o_j[y_j^*] \end{aligned} \tag{10}$$

where o_j is the model score generated by Equation (8) at the j-th step, we marked the $\mathrm{softmax}$ normalizer $\sum_{y' \in V} \exp\left(o_j[y']\right)$ as Z.

Following the work of Devlin et al. (2014), we modified the CE loss function into

$$\begin{aligned} \mathrm{L}_\theta &= -\sum_{j=1}^{|\mathbf{y}|} \left(\log \mathcal{D}_j[y_j^*] - \alpha(\log Z - 0)^2\right) \\ &= -\sum_{j=1}^{|\mathbf{y}|} \left(\log \mathcal{D}_j[y_j^*] - \alpha \log^2 Z\right) \end{aligned} \tag{11}$$

The objective function, shown in Equation (11), is optimized to make sure $\log Z$ is approximated to 0, equally, make Z close to 1 once it converges. We chose the value of α empirically. Because the $\mathrm{softmax}$ normalizer Z is converged to 1 in inference, we just need to ignore Z and predict the target word distribution at the j-th step only with o_j:

$$\mathcal{D}_j = o_j \tag{12}$$

3.4 Cube Pruning

Table 1 clearly shows that the equations in the NMT forward calculation take lots of time. Here, according to the idea behind the cube pruning algorithm, we tried to reduce the time of time-consuming calculations, e.g., Equation (6), and further decrease the search space by introducing the cube pruning algorithm.

3.4.1 Integrating into NMT Decoder

Extended from the naive beam search in the NMT decoder, cube pruning, treated as a pruning algorithm, attempts to reduce the search space and computation complexity by merging some similar items in a beam to accelerate the naive beam search, keeping the 1-best searching result almost unchanged or even better by increasing the beam size. Thus, it is a fast and effective algorithm to generate candidates.

Assume that T restores the set of the finished translations. For each step in naive beam search process, $\mathrm{beamsize}-|\mathrm{T}|$ times forward calculations are required to acquire $\mathrm{beamsize}-|\mathrm{T}|$ probability distributions corresponding to each item in the previous beam (Bahdanau et al., 2015). while for each step in cube pruning, in terms of some constraints, we merge all similar items in the previous beam into one *equivalence class* (called a sub-cube). The constraint we used here is that items being merged in the previous beam should have the same target words. Then, for the sub-cube, only one forward calculation is required to obtain the approximate predictions by using the loose hidden state. Elements in the sub-cube are sorted by previous accumulated NLL along the columns (the first dimension of beam size) and by the approximate predictions along the rows (the second dimension of vocabulary size). After merging, one beam may contain several sub-cubes (the third dimension), we start to search from item in the upper left corner of each sub-cube, which is the best one in the sub-cube, and continue to spread out until enough candidates are found. Once a item is selected, the exact hidden state will be used to calculate its exact NLL.

Through all above steps, the frequency of forward computations decreases. We give an example to dive into the details in Figure 2.

Assume that the beam size is 4. Given the 10^{th}

---

[2] Note that, identical to Bahdanau et al. (2015), we only used $30k$ as the vocabulary size.

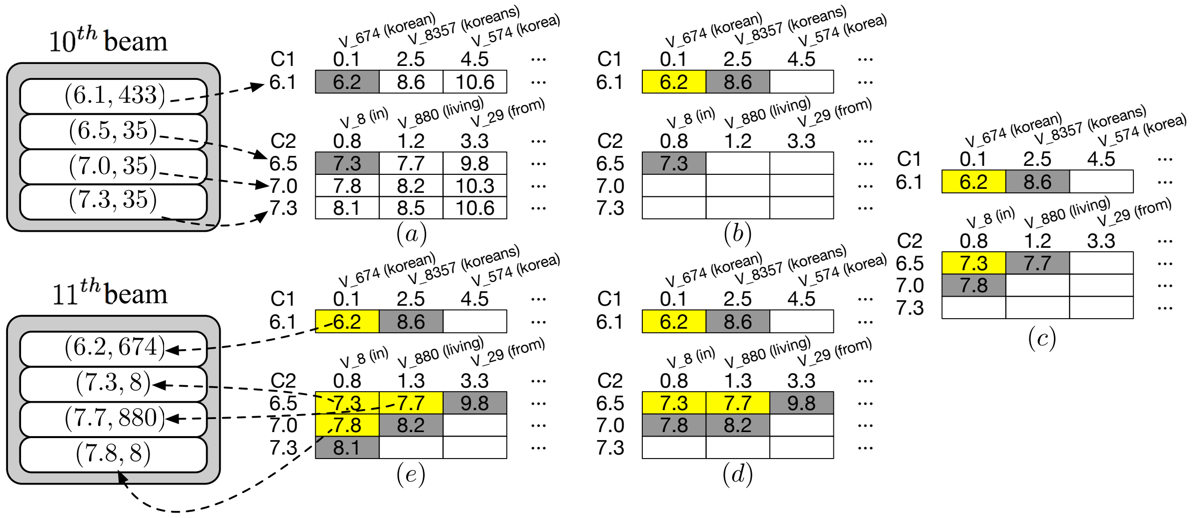

Figure 2: Cube pruning diagram in beam search process during NMT decoding. We only depict the accumulated NLL and the word-level candidate for each item in the beam (in the bracket). Assume the beam size is 4, we initialize a heap for the current step, elements in the 10^{th} beam are merged into two sub-cubes C1 and C2 according to the previous target words; (a) the two elements located in the upper-left corner of the two sub-cubes are pushed into the heap; (b) minimal element $(6.2, 674)$ is popped out, meanwhile, its neighbor $(8.6, 8357)$ is pushed into the heap; (c) minimal element $(7.3, 8)$ is popped out, its right-neighbor $(7.7, 880)$ and lower-neighbor $(7.8, 8)$ are pushed into the heap; (d) minimal element $(7.7, 880)$ is popped out, its right-neighbor $(9.8, 29)$ and down-neighbor $(8.2, 880)$ are pushed into the heap; (e) minimal element $(7.8, 8)$ is popped out, then its down-neighbor $(8.1, 8)$ is pushed into the heap. 4 elements have been popped out, we use them to construct the 11^{th} beam. Yellow boxes indicate the 4-best word-level candidates to be pushed into the 11^{th} beam.

beam, we generate the 11^{th} beam. Different from the naive beam search, we first group items in the previous beam into two sub-cubes C1 and C2 in term of the target word y_{j-1}. As shown in part (a) of Figure 2, $(6.1, 433)$ constructs the sub-cube C1; $(6.5, 35)$, $(7.0, 35)$ and $(7.3, 35)$ are put together to compose another sub-cube C2. Items in part (a) are ranked in ascending order along both row and column dimension according to the accumulated NLL. For each sub-cube, we use the first state vector in each sub-cube as the approximate one to produce the next probability distribution and the next state. At beginning, each upper-left corner element in each sub-cube is pushed into a minimum heap, after popping minimum element from the heap, we calculate and restore the exact NLL of the element, then push the right and lower ones alongside the minimum element into heap. At this rate, the searching continues just like the "diffusion" in the sub-cube until 4 elements are popped, which are ranked in terms of their exact NLLs to construct the 11^{th} beam. Note that once an element is popped, we calculate its exact NLL. From the step (e) in Figure 2, we can see that 4 elements have been popped from C1 and C2, and then ranked in terms of their exact NLLs to build the 11^{th} beam.

We refer above algorithm as the naive cube pruning algorithm (called **NCP**)

3.4.2 Accelerated Cube Pruning

In each step of the cube pruning algorithm, after merging the items in the previous beam, some similar candidates are grouped together into one or more sub-cube(s). We also try to predict the approximate distribution for each sub-cube only according to the top-1 state vector (the first row in the sub-cube in Figure 2), and select next candidates after ranking. The predicted probability distribution will be very similar to that of the naive beam search. Besides, Each sub-cube only requires one forward calculation. Thus, it could save more search space and further reduce the computation complexity for the decoder. Unlike the naive cube pruning algorithm, accelerated cube pruning pops each item, then still use the approximate NLL instead of the exact one. We denote this kind of accelerated cube pruning algorithm as **ACP**.

4 Experiments

We verified the effectiveness of proposed cube pruning algorithm on the Chinese-to-English (Zh-En) translation task.

4.1 Data Preparation

The Chinese-English training dataset consists of 1.25M sentence pairs[3]. We used the NIST 2002 (MT02) dataset as the validation set with 878 sentences, and the NIST 2003 (MT03) dataset as the test dataset, which contains 919 sentences.

The lengths of the sentences on both sides were limited up to 50 tokens, then actually 1.11M sentence pairs were left with 25.0M Chinese words and 27.0M English words. We extracted $30k$ most frequent words as the source and target vocabularies for both sides.

In all the experiments, case-insensitive 4-gram BLEU (Papineni et al., 2002) was employed for the automatic evaluation, we used the script *mteval-v11b.pl*[4] to calculate the BLEU score.

4.2 System

The system is an improved version of attention-based NMT system named RNNsearch (Bahdanau et al., 2015) where the decoder employs a conditional GRU layer with attention, consisting of two GRUs and an attention module for each step[5]. Specifically, Equation (6) is replaced with the following two equations:

$$\tilde{s}_j = \mathbf{GRU}_1(e_{y^*_{j-1}}, s_{j-1}) \tag{13}$$

$$s_j = \mathbf{GRU}_2(c_j, \tilde{s}_j) \tag{14}$$

Besides, for the calculation of relevance in Equation (4), s_{j-1} is replaced with $\tilde{s}_{j-1}$. The other components of the system keep the same as RNNsearch. Also, we re-implemented the beam search algorithm as the naive decoding method, and naive searching on the GPU and CPU server were conducted as two baselines.

4.3 Training Details

Specially, we employed a little different settings from Bahdanau et al. (2015): Word embedding sizes on both sides were set to 512, all hidden sizes in the GRUs of both encoder and decoder were also set to 512. All parameter matrices, including bias matrices, were initialized with the uniform distribution over $[-0.1, 0.1]$. Parameters were updated by using mini-batch Stochastic Gradient Descent (SGD) with batch size of 80 and the learning rate was adjusted by AdaDelta (Zeiler, 2012) with decay constant ρ=0.95 and denominator constant ϵ=1e-6. The gradients of all variables whose L2-norm are larger than a pre-defined threshold 1.0 were normalized to the threshold to avoid gradient explosion (Pascanu et al., 2013). Dropout was applied to the output layer with dropout rate of 0.5. We exploited length normalization (Cho et al., 2014a) strategy on candidate translations in beam search decoding.

The model whose BLEU score was the highest on the validation set was used to do testing. Maximal epoch number was set to 20. Training was conducted on a single Tesla K80 GPU, it took about 2 days to train a single NMT model on the Zh-En training data. For *self-normalization*, we empirically set α as 0.5 in Equation (11)[6].

4.4 Search Strategies

We conducted experiments to decode the MT03 test dataset on the GPU and CPU server respectively, then compared search quality and efficiency among following six search strategies under different beam sizes.

NBS-SN: Naive Beam Search without **SN**

NBS+SN: Naive Beam Search with **SN**

NCP-SN: Cube Pruning without **SN**

NCP+SN: Cube Pruning with **SN**

ACP-SN: Accelerated Cube Pruning without **SN**

ACP+SN: Accelerated Cube Pruning with **SN**

4.5 Comparison of Average Merging Rate

We first give the definition of the Average Merging Rate (denoted as AMR). Given a test dataset, we counted the total word-level candidates (noted as N_w) and the total sub-cubes (noted as N_c) during the whole decoding process, then the AMR can be simply computed as

$$\overline{m} = \mathrm{N}_w / \mathrm{N}_c \tag{15}$$

The MT03 test dataset was utilized to compare the trends of the AMR values under all

---

[3] These sentence pairs are mainly extracted from LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

[4] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/mteval-v11b.pl`

[5] `https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf`

[6] Following Devlin et al. (2014), we had tried 0.01, 0.1, 0.5 1.0 and 10.0 for the value of α, we found that 0.5 produced the best result.

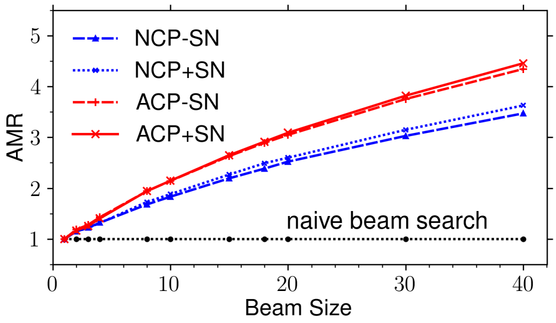

Figure 3: AMR comparison on the MT03 test dataset. Decoding the MT03 test dataset on a single GeForce GTX TITAN X GPU server under the different searching settings. y-axis represents the AMR on the test dataset in the whole searching process and x-axis indicates beam size. Unsurprisingly, we got exactly the same results on the CPU server, not shown here.

six methods. We used the pre-trained model to translate the test dataset on a single GeForce GTX TITAN X GPU server. Beam size varies from 1 to 40, values are included in the set $\{1, 2, 3, 4, 8, 10, 15, 18, 20, 30, 40\}$. For each beam size, six different searching settings were applied to translate the test dataset respectively. The curves of the AMRs during the decoding on the MT03 test dataset under the proposed methods are shown in Figure 3. Note that the AMR values of **NBS** are always 1 whether there is **SN** or not.

Comparing the curves in the Figure 3, we could observe that the naive beam search does not conduct any merging operation in the whole searching process, while the average merging rate in the cube pruning almost grows as the beam size increases. Comparing the red curves to the blue ones, we can conclude that, in any case of beam size, the AMR of the accelerated cube pruning surpasses the basic cube pruning by a large margin. Besides, self-normalization could produces the higher average merging rate comparing to the counterpart without self-normalization.

4.6 Comparison on the GPU Server

Intuitively, as the value of the AMR increases, the search space will be reduced and computation efficiency improves. We compare the two proposed searching strategies and the naive beam search in two conditions (with self-normalization and without self-normalization). Figure 4 demonstrates the results of comparison between the proposed searching methods and the naive beam search baseline in terms of search quality and search efficiency under different beam sizes.

By fixing the beam size and the dataset, we compared the changing trend of BLEU scores for the three distinct searching strategies under two conditions. Without self-normalization, Figure 4a shows the significant improvement of the search speed, however the BLEU score drops about 0.5 points. We then equipped the search algorithm with self-normalization. Figure 4b shows that the accelerated cube pruning search algorithm only spend about one-third of the time of the naive beam search to achieve the best BLEU score with beam size 30. Concretely, when the beam size is set to be 30, **ACP+SN** is 3.3 times faster than the baseline on the MT03 test dataset, and both performances are almost the same.

4.7 Comparison on the CPU Server

Similar to the experiments conducted on GPUs, we also translated the whole MT03 test dataset on the CPU server by using all six search strategies under different beam sizes. The trends of the BLEU scores over those strategies are shown in Figure 5.

The proposed search methods gain the similar superiority on CPUs to that on GPUs, and the decoding speed is obviously slower than that on GPUs. From the Figure 5a, we can also clearly see that, compared with the **NBS-SN**, **NCP-SN** only speeds up the decoder a little, **ACP-SN** produces much more acceleration. However, when we did not introduce self-normalization, the proposed search methods will also result in a loss of about 0.5 BLEU score. The self-normalization made the **ACP** strategy faster than the baseline by about 3.5×, in which condition the **NBS+SN** got the best BLEU score 38.05 with beam size 30 while the **ACP+SN** achieved the highest score 38.12 with beam size 30. The results could be observed in Figure 5b. Because our method is on the algorithmic level and platform-independent, it is reasonable that the proposed method can not only perform well on GPUs, but also accelerate the decoding significantly on CPUs. Thus, the accelerated cube pruning with self-normalization could improve the search quality and efficiency stably.

4.8 Decoding Time

In this section, we only focus on the consuming time of translating the entire MT03 test dataset.

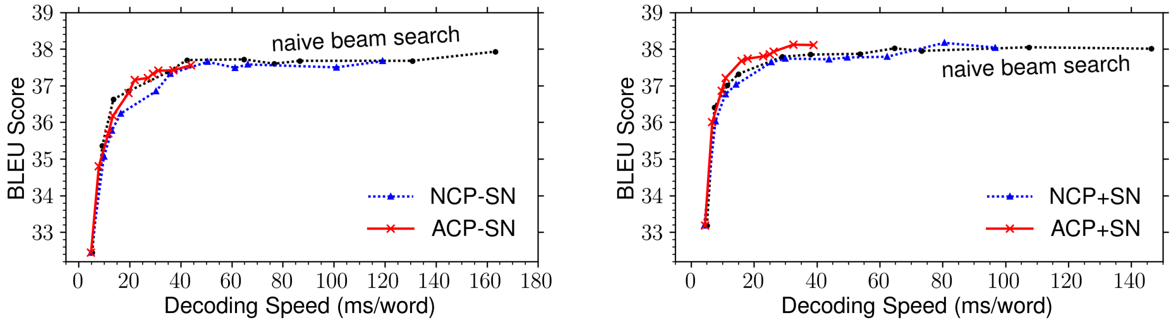

(a) BLEU vs. decoding speed, without self-normalization

(b) BLEU vs. decoding speed, with self-normalization

Figure 4: Comparison among the decoding results of the MT03 test dataset on the single GeForce GTX TITAN X GPU server under the three different searching settings. y-axis represents the BLEU score of translations, x-axis indicates that how long it will take for translating one word on average.

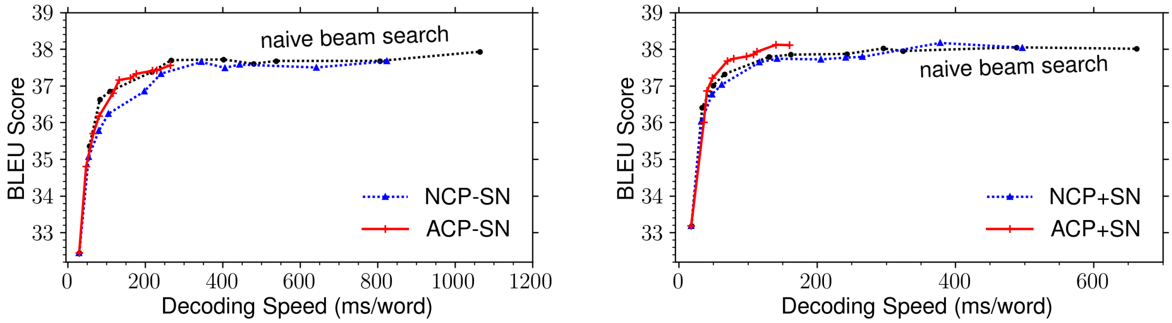

(a) BLEU vs. decoding speed, without self-normalization

(b) BLEU vs. decoding speed, with self-normalization

Figure 5: Comparison among the decoding results of the MT03 test dataset on the single AMD Opteron(tm) Processor under the three different searching settings. y-axis represents the BLEU score of translations, x-axis indicates that how long it will take for translating one word on average.

Under the two conditions, we calculated the times spent on translating the entire test dataset for different beam sizes, then draw the curves in Figure 6 and 7. From the Figure 6a and 6b, we could observe that accelerated cube pruning algorithm speeds up the decoding by about 3.8× on GPUs when the beam size is set to 40. Figure 7a and 7b show that the accelerated cube pruning algorithm speeds up the decoding by about 4.2× on CPU server with the beam size 40.

5 Related Work

Recently, lots of works devoted to improve the efficiency of the NMT decoder. Some researchers employed the way of decreasing the target vocabulary size. Jean et al. (2015) improved the decoding efficiency even with the model using a very large target vocabulary but selecting only a small subset of the whole target vocabulary. Based on the work of Jean et al. (2015), Mi et al. (2016b) introduced sentence-level and batch-level vocabularies as a very small subset of the full output vocabulary, then predicted target words only on this small vocabulary, in this way, they only lost 0.1 BLEU points, but reduced target vocabulary substantially.

Some other researchers tried to raise the efficiency of decoding from other perspectives. Wu et al. (2016) introduced a coverage penalty α and length normalization β into beam search decoder to prune hypotheses and sped up the search process by 30%∼40% when running on CPUs. Hu et al. (2015) used a priority queue to choose the best hypothesis for the next search step, which drastically reduced search space.

Inspired by the works of Mi et al. (2016b) and Huang and Chiang (2007), we consider pruning hypothesis in NMT decoding by using cube pruning algorithm, but unlike traditional SMT decoding where dynamic programming was used to merge equivalent states (e.g., if we use phrase-

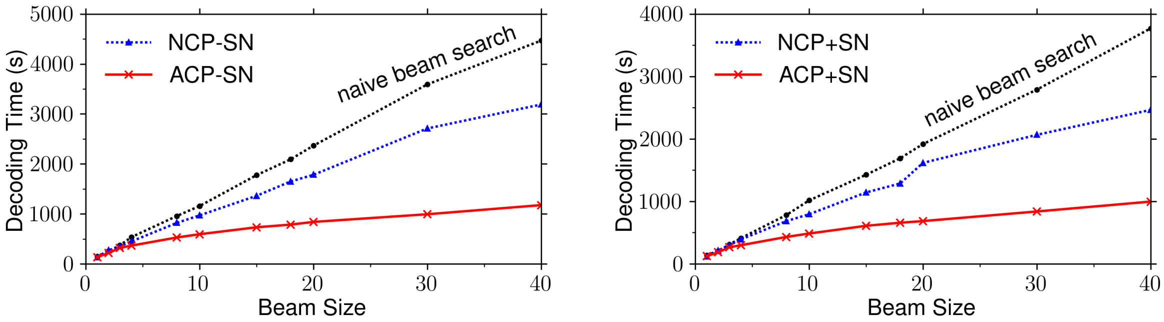

(a) Time spent on translating MT03 test dataset for different beam sizes without self-normalization

(b) Time spent on translating MT03 test dataset for different beam sizes with self-normalization

Figure 6: Comparison among the decoding results of the MT03 test dataset on the single GeForce GTX TITAN X GPU server under the three different searching settings. y-axis represents the BLEU score of translations, x-axis indicates that how long it will take for translating one word on average.

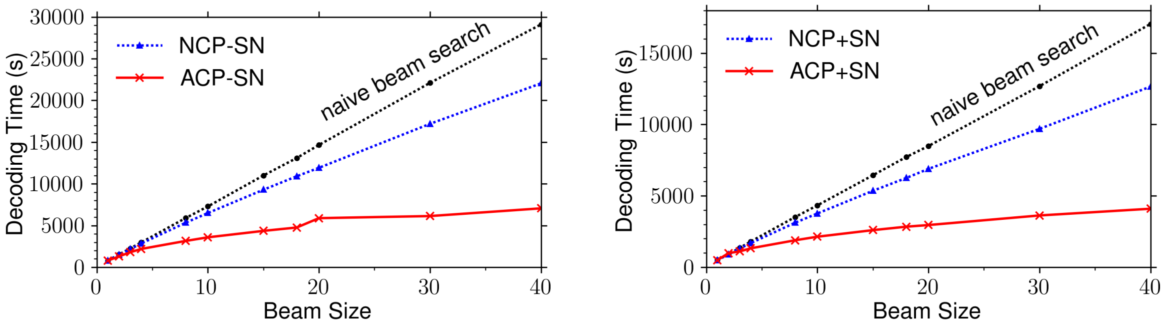

(a) Time spent on translating MT03 test dataset for different beam sizes without self-normalization

(b) Time spent on translating MT03 test dataset for different beam sizes with self-normalization

Figure 7: Comparison among the decoding results of the MT03 test dataset on the single AMD Opteron(tm) Processor under the three different searching settings. y-axis represents the BLEU score of translations, x-axis indicates that how long it will take for translating one word on average.

based decoding with trigram language model, we can merge states with same source-side coverage vector and same previous two target words). However, this is not appropriate for current NMT decoding, since the embedding of the previous target word is used as one input of the calculation unit of each step in the decoding process, we could group equivalence classes containing the same previous target word together.

6 Conclusions

We extended cube pruning algorithm into the decoder of the attention-based NMT. For each step in beam search, we grouped similar candidates in previous beam into one or more equivalence class(es), and bad hypotheses were pruned out. We started searching from the upper-left corner in each equivalence class and spread out until enough candidates were generated. Evaluations show that, compared with naive beam search, our method could improve the search quality and efficiency to a large extent, accelerating the NMT decoder by 3.3× and 3.5× on GPUs and CPUs, respectively. Also, the translation precision could be the same or even better in both situations. Besides, self-normalization is verified to be helpful to accelerate cube pruning even further.

Acknowledgements

We thank the three anonymous reviewers for their comments, Kai Zhao and Haitao Mi for uggestions. This work is supported in part by NSF IIS-1817231 & IIS-1656051, and is also supported in part by National Natural Science Foundation of China (No. 61472428 & No. 61662077).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR 2015*.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2013. Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340. Citeseer.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017a. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135, Vancouver, Canada. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017b. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

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

Xiaoguang Hu, Wei Li, Xiang Lan, Hua Wu, and Haifeng Wang. 2015. Improved beam search with constrained softmax for nmt. *Proceedings of MT Summit XV*, page 297.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 53–64, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126, Sofia, Bulgaria. Association for Computational Linguistics.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016a. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, Austin, Texas. Association for Computational Linguistics.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016b. Vocabulary manipulation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 124–129, Berlin, Germany. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Revisiting Character-Based Neural Machine Translation with Capacity and Compression

Colin Cherry*, **George Foster***, **Ankur Bapna**, **Orhan Firat**, **Wolfgang Macherey**
Google AI
`colincherry,fosterg,ankurbpn,orhanf,wmach@google.com`

Abstract

Translating characters instead of words or word-fragments has the potential to simplify the processing pipeline for neural machine translation (NMT), and improve results by eliminating hyper-parameters and manual feature engineering. However, it results in longer sequences in which each symbol contains less information, creating both modeling and computational challenges. In this paper, we show that the modeling problem can be solved by standard sequence-to-sequence architectures of sufficient depth, and that deep models operating at the character level outperform identical models operating over word fragments. This result implies that alternative architectures for handling character input are better viewed as methods for reducing computation time than as improved ways of modeling longer sequences. From this perspective, we evaluate several techniques for character-level NMT, verify that they do not match the performance of our deep character baseline model, and evaluate the performance versus computation time tradeoffs they offer. Within this framework, we also perform the first evaluation for NMT of conditional computation over time, in which the model learns which timesteps can be skipped, rather than having them be dictated by a fixed schedule specified before training begins.

1 Introduction

Neural Machine Translation (NMT) has largely replaced the complex pipeline of Phrase-Based MT with a single model that is trained end-to-end. However, NMT systems still typically rely on pre- and post-processing operations such as tokenization and word fragmentation through byte-pair encoding (BPE; Sennrich et al., 2016). Although these are effective, they involve hyperparameters that should ideally be tuned for each language pair and corpus, an expensive step that is frequently omitted. Even when properly tuned, the representation of the corpus generated by pipelined external processing is likely to be sub-optimal. For instance, it is easy to find examples of word fragmentations, such as *fling* → *fl* + *ing*, that are linguistically implausible. NMT systems are generally robust to such infelicities—and can be made more robust through subword regularization (Kudo, 2018)—but their effect on performance has not been carefully studied. The problem of finding optimal segmentations becomes more complex when an NMT system must handle multiple source and target languages, as in multilingual translation or zero-shot approaches (Johnson et al., 2017).

Translating characters instead of word fragments avoids these problems, and gives the system access to all available information about source and target sequences. However, it presents significant modeling and computational challenges. Longer sequences incur linear per-layer cost and quadratic attention cost, and require information to be retained over longer temporal spans. Finer temporal granularity also creates the potential for attention jitter (Gulcehre et al., 2017). Perhaps most significantly, since the meaning of a word is not a compositional function of its characters, the system must learn to memorize many character sequences, a different task from the (mostly) compositional operations it performs at higher levels of linguistic abstraction.

In this paper, we show that a standard LSTM sequence-to-sequence model works very well for characters, and given sufficient depth, consistently outperforms identical models operating over word fragments. This result suggests that a productive line of research on character-level models is to seek architectures that approximate standard sequence-to-sequence models while being compu-

---

*Equal contributions

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4295–4305
Brussels, Belgium, October 31 - November 4, 2018.

tationally cheaper. One approach to this problem is *temporal compression*: reducing the number of state vectors required to represent input or output sequences. We evaluate various approaches for performing temporal compression, both according to a fixed schedule; and, more ambitiously, learning compression decisions with a Hierarchical Multiscale architecture (Chung et al., 2017). Following recent work by Lee et al. (2017), we focus on compressing the encoder.

Our contributions are as follows:

- The first large-scale empirical investigation of the translation quality of standard LSTM sequence-to-sequence architectures operating at the character level, demonstrating improvements in translation quality over word fragments, and quantifying the effect of corpus size and model capacity.
- A comparison of techniques to compress character sequences, assessing their ability to trade translation quality for increased speed.
- A first attempt to learn how to compress the source sequence during NMT training by using the Hierarchical Multiscale LSTM to dynamically shorten the source sequence as it passes through the encoder.

2 Related Work

Early work on modeling characters in NMT focused on solving the out-of-vocabulary and softmax bottleneck problems associated with word-level models (Ling et al., 2015; Costa-jussà and Fonollosa, 2016; Luong and Manning, 2016). These took the form of word-boundary-aware hierarchical models, with word-level models delegating to character-level models to generate representations in the encoder and words in the decoder. Our work will not assume fixed word boundaries are given in advance.

With the advent of word-fragment approaches, interest in character-level processing fell off, but has recently been reignited with the work of Lee et al. (2017). They propose a specialized character-level encoder, connected to an unmodified character-level RNN decoder. They address the modeling and efficiency challenges of long character sequences using a convolutional layer, max-pooling over time, and highway layers. We agree with their conclusion that character-level translation is effective, but revisit the question of whether their specific encoder produces a desirable speed-quality tradeoff in the context of a much stronger baseline translation system. We draw inspiration from their pooling solution for reducing sequence length, along with similar ideas from the speech community (Chan et al., 2016), when devising fixed-schedule reduction strategies in Section 3.3.

One of our primary contributions is an extensive invesigation of the efficacy of a typical LSTM-based NMT system when operating at the character-level. The vast majority of existing studies compare a specialized character-level architecture to a distinct word-level one. To the best of our knowledge, only a small number of papers have explored running NMT unmodified on character sequences; these include: Luong and Manning (2016) on WMT'15 English-Czech, Wu et al. (2016) on WMT'14 English-German, and Bradbury et al. (2016) on IWSLT German-English. All report scores that either trail behind or reach parity with word-level models. Only Wu et al. (2016) compare to word fragment models, which they show to outperform characters by a sizeable margin. We revisit the question of character- versus fragment-level NMT here, and reach quite different conclusions.

3 Methods

3.1 Baseline Sequence-to-Sequence Model

We adopt a simplified version of the LSTM architecture of Chen et al. (2018) that achieves state-of-the-art performance on the competitive WMT14 English-French and English-German benchmarks. This incorporates bidirectional LSTM (BiLSTM) layers in the encoder, concatenating the output from forward and backward directions before feeding the next layer. Output from the top encoder layer is projected down to the decoder dimension and used in an additive attention mechanism computed over the bottom decoder layer. The decoder consists of unidirectional layers, all of which use the encoder context vectors computed from attention weights over the bottom layer. For both encoder and decoder we use layer normalization (Ba et al., 2016) and residual connections beginning at the third layer. We do not apply a non-linearity to LSTM output. We regularize with dropout applied to embeddings and to the output of each LSTM layer.

In the interests of simplicity and reproducibility, we depart from Chen et al. (2018) in several ways: we do not use multi-headed attention, feed encoder context vectors to the softmax, regularize with label smoothing or weight decay, nor apply dropout to the attention mechanism.

Our baseline character models and BPE models both use this architecture, differing only in whether the source and target languages are tokenized into sequences of characters or BPE word fragments. We describe BPE briefly below.

3.2 Byte-Pair Encoding

Byte-Pair Encoding (BPE) offers a simple interpolation between word- and character-level representations (Sennrich et al., 2016). It creates a vocabulary of frequent words and word fragments in an iterative greedy merging process that begins with characters and ends when a desired vocabulary size is reached. The source and target language are typically processed together in order to exploit lexical similarities. Given a vocabulary, BPE re-tokenizes the corpus into word fragments in a greedy left-to-right fashion, selecting the longest possible vocabulary match, and backing off to characters when necessary.

Since each BPE token consists of one or more characters, BPE-tokenized sequences will be shorter than character sequences. Viewed as a mechanism to reduce sequence length, BPE differs from the solutions we will discuss subsequently in that it increases the vocabulary size, delegating the task of creating representations for word fragments to the embedding table. Also, despite being data-driven, its segmentation decisions are fixed before NMT training begins.

3.3 Fixed stride Temporal Pooling

We explore using fixed stride temporal pooling within the encoder to compress the source character sequence. These solutions are characterized by pooling the contents of two or more contiguous timesteps to create a single vector that summarizes them, and will replace them to shorten the sequence in the next layer. These approaches can learn to interpret the raw character sequence in service to their translation objective, but any such interepretation must fit into the pooling schedule that was specified during network construction. We evaluate two methods in this family: a re-implementation of Lee et al. (2017), and a version of our baseline with interspersed pooling layers.

As mentioned earlier, Lee et al. (2017) propose a specialized character encoder that combines convolutional layers to accumulate local context, max-pooling layers to reduce sequence lengths, highway layers to increase network capacity, followed by bidirectional GRU layers to generate globally aware contextual source representations. This strategy is particularly efficient because all reductions happen before the first recurrent layer. We re-implement their approach faithfully, with the exceptions of using LSTMs in place of GRUs,[1] and modifying the batch sizes to accomodate our multi-GPU training scheme.

While pooling based approaches are typically employed in association with convolutional layers, we can also intersperse pooling layers into our high capacity baseline encoder. This means that after each BiLSTM layer, we have the option to include a fixed-stride pooling layer to compress the sequence before it is processed by the next BiLSTM layer. This is similar to the pyramidal LSTM encoders used for neural speech recognition (Chan et al., 2016). This general strategy affords considerable flexibility to the network designer, leaving the type of pooling (concatenation, max, mean), and the strides with which to pool as design decisions that can be tuned to fit the task.

3.4 Learned Temporal Compression

It is unsatisfying to compress a sequence on a fixed schedule; after all, the characters in a sentence do not each carry an identical amount of information. The goal of this section is to explore data-driven reduction methods that are optimized to the NMT system's objective, and which learn to compress as a part of training.

Any strategy for performing temporal compression will necessarily make discrete decisions, since sentence length is discrete. Examples of such strategies include sparse attention (Raffel et al., 2017) and discrete auto-encoders (Kaiser et al., 2018). For our initial exploration, we chose the hierarchical multiscale (HM) architecture of Chung et al. (2017), which we briefly describe.

3.4.1 Hierarchical Multiscale LSTM

The HM is a bottom-up temporal subsampling approach, with each layer selecting the timesteps that will survive to the layer above. At a given timestep t and layer ℓ, the network makes a binary decision,

---

[1] Development experiments indicated that using LSTMs over GRUs resulted in a slight improvement.

z_t^ℓ, to determine whether or not it should send its output up to layer $\ell + 1$. The preactivation for this decision, $\tilde{z}_t^\ell$, is a function of the current node's inputs from below and from the previous hidden state, similar to an LSTM gate. However, z_t^ℓ's activation is a binary step function in the forward pass, to enable discrete decisions, and a hard sigmoid in the backward pass, to allow gradients to flow through the decision point.[2] The z_t^ℓ decision affects both the layer above, and the next timestep of the current layer:

- $z_t^\ell = 1$, *flow up*: the node above (t, ℓ+1) performs a normal LSTM update; the node to the right (t+1, ℓ) performs a modified update called a *flush*, which ignores the LSTM internal cell at (t, ℓ), and redirects the incoming LSTM hidden state from (t, ℓ) to (t, $\ell + 1$).
- $z_t^\ell = 0$, *flow right*: the node above (t, ℓ+1) simply copies the cell and hidden state values from ($t-1$, ℓ+1); the node to the right (t+1, ℓ) performs a normal LSTM update.

Conceptually, when $z_t^\ell = 0$, the node above it becomes a placeholder and is effectively removed from the sequence for that layer. Shorter upper layers save computation and facilitate the left-to-right flow of information for the surviving nodes.

Typically, one uses the top hidden state h_t^L from a stack of L RNNs to provide the representation for a timestep t. But for the HM, the top layer may be updated much less frequently than the layers below it. To enable tasks that need a distinct representation for each timestep, such as language modeling, the HM employs a *gated output module* to mix hidden states across layers. This learned module combines the states $h_t^1, h_t^2, \ldots, h_t^L$ using scaling and projection operators to produce a single output h_t.

3.4.2 Modifying the HM for NMT

We would like sequences to become progressively shorter as we move upward through the layers. As originally specified, the HM calculates z_t^ℓ independently for every t and ℓ, including copied nodes, meaning that a "removed" timestep could reappear in a higher layer when a copied node (t, ℓ) sets $z_t^\ell = 1$. This is easily addressed by locking $z_t^\ell = 0$ for copied nodes, creating a hierarchical structure in which upper layers never increase the amount of computation.

We also found that the flush component of the original architecture, which modifies the LSTM update at (t+1, ℓ) to discard the LSTM's internal cell, provided too much incentive to leave z_t^ℓ at 0, resulting in degenerate configurations which collapsed to having very few tokens in their upper layers. We addressed this by removing the notion of a flush from our architecture. The node to the right (t+1, ℓ) always performs a normal LSTM update, regardless of z_t^ℓ. This modification is similar to one proposed independently by Kádár et al. (2018), who simplified the flush operation by removing the connection to (t, $\ell + 1$).

We found it useful to change the initial value of the bias term used in the calculation of $\tilde{z}_t^\ell$, which we refer to as the z-bias. Setting z-bias to 1, which is the saturation point for the hard sigmoid with slope 1, improves training stability by encouraging the encoder to explore configurations where most timesteps survive through all layers, before starting to discard them.

Even with these modifications, we observed degenerate behavior in some settings. To discourage this, we added a compression loss component similar to that of Ke et al. (2018) to penalize z activation rates outside a specified range α_1, α_2: $\mathcal{L}_c = \sum_l \max(0,\ Z^l - \alpha_1 T,\ \alpha_2 T - Z^l)$, where T is source sequence length and $Z^l = \sum_{t=1}^{T} z_t^l$.

To incorporate the HM into our NMT encoder, we replace the lowest BiLSTM layer with unidirectional HM layers.[3] We adapt any remaining BiLSTM layers to copy or update according to the z-values calculated by the top HM layer.

4 Experimental Design

4.1 Corpora

We adopt the corpora used by Lee et al (2017), with the exception of WMT15 Russian-English.[4] To measure performance on an "easy" language pair, and to calibrate our results against recent benchmarks, we also included WMT14 English-French. Table 1 gives details of the corpora used. All corpora are preprocessed using Moses tools.[5]

---

[2] This disconnect between forward and backward activations is known as a straight-through estimator (Bengio et al., 2013).

[3] The flush operation makes the original HM inherently left-to-right. Since we have dropped flushes from our current version, it should be straightforward to devise a bidirectional variant, which we leave to future work.

[4] Due to licence restrictions.

[5] Scripts and arguments:
`remove-non-printing-char.perl`

corpus	train	dev	test
WMT15 Finnish-En	2.1M	1500	1370
WMT15 German-En	4.5M	3003	2169
WMT15 Czech-En	14.8M	3003	2056
WMT14 En-French	39.9M	3000	3003

Table 1: Corpora, with linecounts. Test sets are WMT14-15 newstest. Dev sets are newsdev 2015 (Fi) and newstest 2013 (De, Fr), and 2014 (Cs).

Dev and test corpora are tokenized, but not filtered or cleaned. Our character models use only the most frequent 496 characters across both source and target languages; similarly, BPE is run across both languages, with a vocabulary size of 32k.

4.2 Model sizes, training, and inference

Except where noted below, we used 6 bidirectional layers in the encoder, and 8 unidirectional layers in the decoder. All vector dimensions were 512.

Models were trained using sentence-level cross-entropy loss. Batch sizes are capped at 16,384 tokens, and each batch is divided among 16 NVIDIA P100s running synchronously.

Parameters were initialized with a uniform (0.04) distribution. We use the Adam optimizer, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-6}$ (Kingma and Ba, 2014). Gradient norm is clipped to 5.0. The initial learning rate is 0.0004, and we halve it whenever dev set perplexity has not decreased for 2k batches, with at least 2k batches between successive halvings. Training stops when dev set perplexity has not decreased for 8k batches.

Inference uses beam search with 8 hypotheses, coverage penalty of 0.2 (Tu et al., 2016), and length normalization of 0.2 (Wu et al., 2016).

4.3 Tuning and Evalution

When comparing character-level and BPE models, we tuned dropout independently for each setting, greedily exploring increments of 0.1 in the range 0.1–0.5, and selecting based on dev-set BLEU. This expensive strategy is crucial to obtaining valid conclusions, since optimal dropout values tend to be lower for character models.

Our main evaluation metric is Moses-tokenized case-sensitive BLEU score. We report test-set scores on the checkpoints having highest dev-set BLEU. To facilitate comparison with future work we also report SacreBLEU scores (Post, 2018) for key results, using the Moses detokenizer.

	Tokenized BLEU			SacreBLEU
Language	BPE	Char	Delta	Char
EnFr	38.8	39.2	0.4	38.1
CsEn	24.8	25.9	1.1	25.6
DeEn	29.7	31.6	1.9	31.6
FiEn	17.5	19.3	1.8	19.5

Table 2: Character versus BPE translation.

Comparison Point			Ref	Ours
Chen et al. (2018)	BPE	EnFr	41.0	38.8
Wu et al. (2016)	BPE	EnFr	39.0	
Lee et al. (2017)	Char	CsEn	22.5	25.9
		DeEn	25.8	31.6
		FiEn	13.1	19.3

Table 3: Comparisons with some recent points in the literature. Scores are tokenized BLEU.

5 Results

5.1 Character-level translation

We begin with experiments to compare the standard RNN architecture from Section 3.1 at the character and BPE levels, using our full-scale model with 6 bidirectional encoder layers and 8 decoder layers. The primary results of our experiments are presented in Table 2, while Table 3 positions the same results with respect to recent points from the literature.

There are a number of observations we can draw from this data. First, from the EnFr results in Table 3, we are in line with GNMT (Wu et al., 2016), and within 2 BLEU points of the RNN and Transformer models investigated by Chen et al. (2018). So, while we are not working at the exact state-of-the-art, we are definitely in a range that should be relevant to most practitioners.

Also from Table 3, we compare quite favorably with Lee et al. (2017), exceeding their reported scores by 3-6 points, which we attribute to having employed much higher model capacity, as they use a single bidirectional layer in the encoder and a two-layer decoder. We investigate the impact of model capacity in Section 5.1.1.

Finally, Table 2 clearly shows the character-level systems outperforming BPE for all language pairs. The dominance of character-level methods in Table 2 indicates that RNN-based NMT architectures are not only capable of translating charac-

---

`tokenize.perl`

`clean-corpus-n.perl -ratio 9 1 100`

ter sequences, but actually benefit from them. This is in direct contradiction to the few previously reported results on this matter, which can in most cases be explained by our increased model capacity. The exception is GNMT (Wu et al., 2016), which had similar depth. In this case, possible explanations for the discrepancy include our use of a fully bidirectional encoder, our translating into English instead of German, and our model-specific tuning of dropout.

5.1.1 Effect of model capacity

Character-level NMT systems have a more difficult sequence-modeling task, as they need to infer the meaning of words from their constituent characters, where models with larger tokens instead delegate this task to the embedding table. Therefore, we hypothesize that increasing the model's capacity by adding layers will have a greater impact on character-level models. Figure 1 tests this hypothesis by measuring the impact of three model sizes on test BLEU score. For each of our four language pairs, the word-fragment model starts out ahead, and quickly loses ground as architecture size increases. For the languages with greater morphological complexity—German, Czech and Finnish—the slope of the character model's curve is notably steeper than that of the BPE system, indicating that these systems could benefit from yet more modeling capacity.

5.1.2 Effect of corpus size

One of the most compelling arguments for working with characters (and to a lesser extent, word-fragments) is improved generalization. Through morphological generalizations, the system can better handle low-frequency and previously unseen words. It stands to reason that as the training corpus increases in size, the importance of these generalization capabilities will decrease. We test this hypothesis by holding the language pair constant, and varying the training corpus size by downsampling the full training corpus. We choose EnFr because it has by far the most available data. We compare four sizes: 2M, 4M, 14M and 40M.

The results are shown in Figure 2. As expected, the gap between character and word-fragment modeling decreases as corpus size increases. From the slopes of the curves, we can infer that the advantage of character-level modeling will disappear completely as we reach 60-70M sentence pairs. However, there is reason to expect this break-even point to be much higher for more morphologically complex languages. It is also important to recall that relatively few language-pairs can assemble parallel corpora of this size.

5.1.3 Speed

The performance advantage of working with characters comes at a significant computational cost. With our full-sized architecture, character models trained roughly 8x more slowly than BPE models.[6] Figure 3 shows that training time grows linearly with number of layers in the model, and that character models have a much higher per-layer cost: roughly 0.38 msec/sentence versus 0.04 for BPE. We did not directly measure the difference in attention cost, but it cannot be greater than the difference in total cost for the smallest number of layers. Therefore, we can infer from Figure 3 that processing 5 layers in a character model incurs roughly the same time cost as attention. This is surprising given the quadratic cost of attention, and indicates that efforts to speed up character models cannot focus exclusively on attention.

5.1.4 Qualitative comparison

To make a qualitative comparison between word fragments (BPE) and characters for NMT, we examined 100 randomly selected sentence pairs from the DeEn test set. One author examined the sentences, using a display that showed the source[7] and the reference, along with the output of BPE and character models. Any differences between the two outputs were highlighted. They then assigned tags to both system outputs indicating broad error categories, such as lexical choice, word order and German compound handling.[8] Tags were restricted to cases where one system made a mistake that the other did not.

Of the 100 sentences, 47 were annotated as being identical or of roughly the same quality. The remaining 53 exhibited a large variety of differences. Table 4 summarizes the errors that were most easily characterized. BPE and character sys-

---

[6] Recall that we use batches containing 16,384 tokens—corresponding to a fixed memory budget—for both character and BPE models. Thus character models are slowed not only by having longer sentences, but also by parallelizing across fewer sentences in each batch.

[7] The annotating author does not speak German.

[8] Our annotator also looked specifically for agreement and negation errors, as studied by Sennrich (2017) for English-to-German character-level NMT. However, neither system exhibited these error types with sufficient frequency to draw meaningful conclusions.

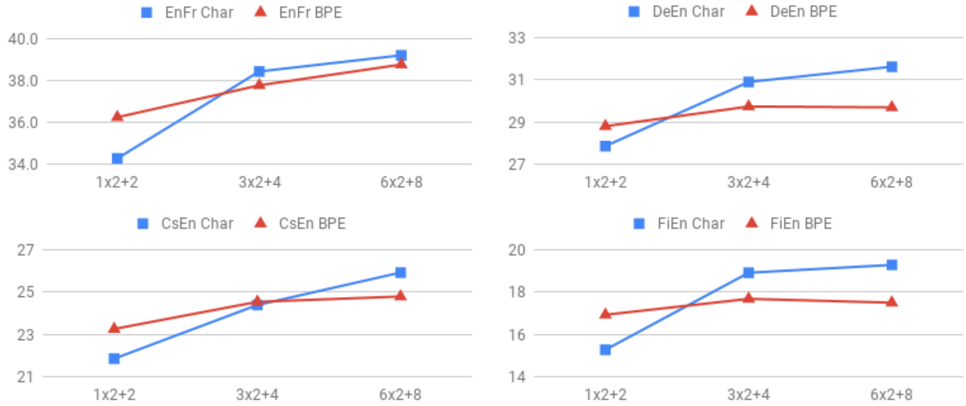

Figure 1: Test BLEU for character and BPE translation as architectures scale from 1 BiLSTM encoder layer and 2 LSTM decoder layers (1×2+2) to our standard 6×2+8. The y-axis spans 6 BLEU points for each language pair.

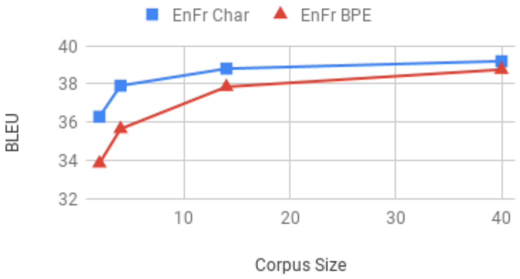

Figure 2: BLEU versus training corpus size in millions of sentence pairs, for the EnFr language-pair.

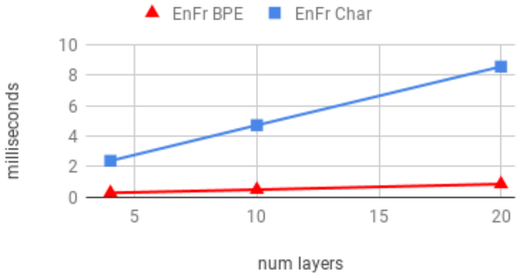

Figure 3: Training time per sentence versus total number of layers (encoder plus decoder) in the model.

Error Type	BPE	Char
Lexical Choice	19	8
Compounds	13	1
Proper Names	2	1
Morphological	2	2
Other lexical	2	4
Dropped Content	7	0

Table 4: Error counts out of 100 randomly sampled examples from the DeEn test set.

tems differ most in the number of lexical choice errors, and in the extent to which they drop content. The latter is surprising, and appears to be a side-effect of a general tendency of the character models to be more faithful to the source, verging on being overly literal. An example of dropped content is shown in Table 5 (top).

Regarding lexical choice, the two systems differ not only in the number of errors, but in the nature of those errors. In particular, the BPE model had more trouble handling German compound nouns. Table 5 (bottom) shows an example which exhibits two compound errors, including one where the character system is a strict improvement, translating *Bunsenbrenner* into *bunsen burner* instead of *bullets*. The second error follows another common pattern, where both systems mishandle the German compound (*Chemiestunden / chemistry lessons*), but the character system fails in a more useful way.

We also found that both systems occasionally mistranslate proper names. Both fail by attempting to translate when they should copy over, but the BPE system's errors are harder to understand as they involve semantic translation, rendering *Britta Hermann* as *Sir Leon*, and *Esme Nussbaum* as *smiling walnut*.[9] The character system's one observed error in this category was phonetic rather than semantic, rendering *Schotten* as *Scottland*.

Interestingly, we also observed several instances where the model correctly translates the German 24-hour clock into the English 12-hour clock; for example, *19.30* becomes *7:30 p.m.*. This deterministic transformation is potentially in reach for both models, but we observed it only for the character system in this sample.

---

[9] The BPE segmentations for these names were: `_Britt a _Herr mann` and `_Es me _N uss baum`

Src	Für diejenigen, die in ländlichen und abgelegenen Regionen **des Staates** lebten, . . .
Ref	Those living in regional and remote areas **of the state** . . .
BPE	For those who lived in rural and remote regions, . . .
Char	For those who lived in rural and remote regions **of the state**, . . .
Src	Überall im Land, in Tausenden von **Chemiestunden**, haben Schüler ihre **Bunsenbrenner** auf Asbestmatten abgestellt.
Ref	Up and down the country, in myriad **chemistry lessons**, pupils have perched their **Bunsen burners** on asbestos mats.
BPE	Across the country, thousands of **chemists** have turned their **bullets** on asbestos mats.
Char	Everywhere in the country, in thousands of **chemical hours, students** have parked their **bunsen burners** on asbestos mats.

Table 5: Examples of BPE and character outputs for two sentences from the DeEn test set, demonstrating dropped content (top) and errors with German compounds (bottom).

5.2 Compressing the Source Sequence

At this point we have established that character-level NMT benefits translation quality, but incurs a large computational cost. In this section, we evaluate the speed-quality tradeoffs of various techniques for reducing the number of state vectors required to represent the source sentence. All experiments are conducted on our DeEn language pair, chosen for having a good balance of morphological complexity and training corpus size.

5.2.1 Optimizing the BPE vocabulary

Recall that BPE interpolates between word- and character-level processing by tokenizing consecutive characters into word fragments; larger BPE vocabulary sizes result in larger fragments and shorter sequences. If character-level models outperform BPE with a vocabulary size of 32k, then is there a smaller BPE vocabulary size that reaps the benefits of character-level processing, while still substantially reducing the sequence length?

To answer this question, we test a number of BPE vocabularies, as shown in Table 6. For each vocabulary, we measure BLEU and sequence compression rate, defined as the average size of the source sequence in characters divided by its size in word fragments (the ratio for the target sequence was similar). Unfortunately, even at just 1k vocabulary items, BPE has already lost a BLEU point with respect to the character model. When comparing these results to the other methods in this section, it is important to recall that BPE is compressing both the source and target sequence (by approximately the same amount), doubling its effective compression rate.

Encoder	BPE Size	BLEU	Comp.
BiLSTM	Char	31.6	1.00
BiLSTM	1k	30.5	0.44
BiLSTM	2k	30.4	0.35
BiLSTM	4k	30.0	0.29
BiLSTM	8k	29.6	0.25
BiLSTM	16k	30.0	0.22
BiLSTM	32k	29.7	0.20
Lee et. al. reimpl	Char	28.0	0.20
BiLSTM + pooling	Char	30.0	0.47
HM, 3-layer	Char	31.2	0.77
HM, 2-layer	Char	30.9	0.89

Table 6: Compression results on WMT15 DeEn. The *Comp.* column shows the ratio of total computations carried out in the encoder.

5.2.2 Fixed Stride Compression

The goal of these experiments is to determine whether using fixed schedule compression is a feasible alternative to BPE. We evaluate our reimplementation of the pooling model of Lee et al. (2017) and our pooled BiLSTM encoder, both described in Section 3.3. For the pooled BiLSTM encoder, development experiments led us to introduce two mean-pooling layers, a stride 3 layer after the second BiLSTM, and a stride 2 layer after the third. Therefore, the final output of the encoder is compressed by a factor of 6.

The results are also shown in Table 6. Note that for the pooled BiLSTM, different encoder layers have different lengths: 2 full length layers, followed by 1 at $\frac{1}{3}$ length and 3 at $\frac{1}{6}$ length. Therefore, we report the average compression across layers here and for the HM in Section 5.2.3.

Our implementation of Lee et al. (2017) outperforms the original results by more than 2 BLEU

Model	BLEU	Comp.
LSTM	28.9	1.00
HM, no-fl	27.3	0.63
HM, no-fl, hier	28.5	0.65
HM, no-fl, hier, zb1, anneal	28.8	0.65

Table 7: HM small-scale results on WMT15 DeEn. The *Comp.* column is the proportion of layer-wise computation relative to the full LSTM.

points. We suspect most of these gains result from better optimization of the model with large batch training. However, our attempts to scale this encoder to larger depths, and therfore to the level of performance exhibited by our other systems, did not result in any significant improvements. This is possibly due to difficulties with optimizing a deeper stack of diverse layers.

Comparing the performance of our Pooled BiLSTM model against BPE, we notice that for a comparable level of compression (BPE size of 1k), BPE out-performs the pooled model by around 0.5 BLEU points. At a similar level of performance (BPE size of 4k), BPE has significantly shorter sequences. Although fixed-stride pooling does not yet match the performance of BPE, we remain optimistic about its potential. The appeal of these models derives from their simplicity; they are easy to optimize, perform reasonably well, and remove the complication of BPE preprocessing.

5.2.3 Hierarchical Multiscale Compression

We experimented with using the Hierarchical Multiscale (HM; Section 3.4.1) architecture to learn compression decisions for the encoder.

For initial exploration, we used a scaled-down architecture consisting of 3 unidirectional HM encoder layers and 2 LSTM decoder layers, attending over the HM's gated output module. Comparisons to an equivalent LSTM are shown in table 7. The first two HM lines justify the no-flush and hierarchical modifications described in Section 3.4.1, yielding incremental gains of 27.3 (the flush variant failed to converge), and 1.2 respectively. Initializing z-bias to 1 and annealing the slope of the hard binarizer from 1.0 to 5.0 over 80k minibatches gave further small gains, bringing the HM to parity with the LSTM while saving approximately 35% of layer-wise computations. Interestingly, we found that, over a wide range of training conditions, each layer tended to reduce computation by roughly 60% relative to the layer below.[10]

For full-scale experiments, we stacked 5 BiLSTM layers on top of 2 or 3 HM layers, as described in section 3.4.1, using only the top HM layer (rather than the gated output module) as input to the lowest BiLSTM layer. To stabilize the 3-HM configuration we used a compression penalty with a weight of 2, and α_1 and α_2 of 0.1 and 0.9. Given the tendency of HM layers to reduce computation by a roughly constant proportion, we expect fewer z-gates to be open in the 3-HM configuration, but this is achieved at the cost of one extra layer relative to our standard 12-layer encoder. As shown in table 6, the 3-HM configuration achieves much better compression even when this is accounted for, and also gives slightly better performance than 2-HM. In general, HM gating results in less compression but better performance than the fixed-stride techniques.

Although these preliminary results are promising, it should be emphasized that the speed gains they demonstrate are conceptual, and that realizing them in practice comes with significant engineering challenges.

6 Conclusion

We have demonstrated the translation quality of standard NMT architectures operating at the character-level. Our experiments show the surprising result that character NMT can substantially out-perform BPE tokenization for all but the largest training corpora sizes, and the less surprising result that doing so incurs a large computational cost. To address this cost, we have explored a number of methods for source-sequence compression, including the first application of the Hierarchical Multiscale LSTM to NMT, which allows us to learn to dynamically compress the source sequence.

We intend this paper as a call to action. Character-level translation is well worth doing, but we do not yet have the necessary techniques to benefit from this quality boost without suffering a disproportionate reduction in speed. We hope that these results will spur others to revisit the question of character-level translation as an interesting testbed for methods that can learn to process, summarize or compress long sequences.

---

[10] For instance, the 2nd and 3rd layer of the best configuration shown had on average 60% and 36% of z gates open, yielding the computation ratio of $(1 + 0.6 + 0.36)/3 = 0.65$.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (To Appear)*.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical multiscale recurrent neural networks. In *International Conference on Learning Representations*, Toulon, France.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany. Association for Computational Linguistics.

Caglar Gulcehre, Francis Dutil, Adam Trischler, and Yoshua Bengio. 2017. Plan, attend, generate: Character-level neural machine translation with planning. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 228–234, Vancouver, Canada. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernand a Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*.

Nan Rosemary Ke, Konrad Zolna, Alessandro Sordoni, Zhouhan Lin, Adam Trischler, Yoshua Bengio, Joelle Pineau, Laurent Charlin, and Chris Pal. 2018. Focused hierarchical RNNs for conditional sequence processing. *arXiv preprint arXiv:1806.04342*.

Diederik P Kingma and Jimmy Lei Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (To Appear)*.

Ákos Kádár, Marc-Alexandre Côté, Grzegorz Chrupała, and Afra Alishahi. 2018. Revisiting the hierarchical multiscale LSTM. In *COLING*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. *arXiv preprint arXiv:1804.08771v1*.

Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *International Conference on Machine Learning*, Sydney, Australia.

Rico Sennrich. 2017. How Grammatical is Character-level Neural Machine Translation? Assessing MT Quality with Contrastive Translation Pairs. In *EACL*, pages 376–382, Valencia, Spain.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawaand Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

A Skeleton-Based Model for Promoting Coherence Among Sentences in Narrative Story Generation

Jingjing Xu[1], **Xuancheng Ren**[1], **Yi Zhang**[1], **Qi Zeng**[1], **Xiaoyan Cai**[2], **Xu Sun**[1]
[1]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[2]School of Automation, Northwestern Polytechnical University
`{jingjingxu,renxc,zhangyi16,pkuzengqi,xusun}@pku.edu.cn`
`xiaoyanc@nwpu.edu.cn`

Abstract

Narrative story generation is a challenging problem because it demands the generated sentences with tight semantic connections, which has not been well studied by most existing generative models. To address this problem, we propose a skeleton-based model to promote the coherence of generated stories. Different from traditional models that generate a complete sentence at a stroke, the proposed model first generates the most critical phrases, called skeleton, and then expands the skeleton to a complete and fluent sentence. The skeleton is not manually defined, but learned by a reinforcement learning method. Compared to the state-of-the-art models, our skeleton-based model can generate significantly more coherent text according to human evaluation and automatic evaluation. The G-score is improved by 20.1% in human evaluation.[1]

1 Introduction

We focus on the problem of narrative story generation, a special kind of story generation (Li et al., 2013). It requires systems to generate a narrative story based on a short description of a scene or an event, as shown in Table 1. In general, a narrative story is described with several inter-related scenes. Different from traditional text generation tasks, this task is more challenging because it demands the generated sentences with tight semantic connections. Currently, most state-of-the-art approaches (Jain et al., 2017; Liu et al., 2017; Fan et al., 2018; Ma et al., 2018a; Xu et al., 2018b) are largely based on Sequence-to-Sequence (Seq2Seq) models (Sutskever et al., 2014), which generate a sentence at a stroke in a left-to-right manner.

However, we find it hard for these approaches to model the semantic dependency among sentences, which causes low-quality generated stories where the scenes are irrelevant. In fact, as shown in Figure 1, we observe that the connection among sentences is mainly reflected through key phrases, such as predicates, subjects, objects and so on. In this work, we regard the phrases that express the key meanings of a sentence as a skeleton. The other words (e.g., modifiers) not only are redundant for understanding semantic dependency, but also make the dependency sparse. Therefore, generating all information at a stroke makes it difficult to learn the dependency of sentences. In contrast, the sentences written by humans are closely tied and the whole story is more coherent and fluent. It is mainly attributed to the way of human writing where we often first come up with a skeleton and then reorganize them into a fluent sentence.

Task Description
Input: A short description of a scene or an event. Output: A relevant narrative story following the input.
Examples
Input: *Fans came together to celebrate the opening of a new studio for an artist.* **Output**: *The artist provided champagne in flutes for everyone. Friends toasted and cheered the artist as she opened her new studio.*
Input: *Last week I attended a wedding for the first time.* **Output**: *There were a lot of families there. They were all taking pictures together. Everyone was very happy. The bride and groom got to ride in a limo that they rented.*

Table 1: An illustration of narrative story generation.

Therefore, motivated by the way of human writing, we propose a skeleton-based model to improve the coherence of generated text. The key idea is to first generate a skeleton and then expand the skeleton to a complete sentence. As a simplified sentence representation, the skeleton can help machines learn the dependency of sentences by avoiding the interference of irrelevant information. Our model contains two parts: a skeleton-

---

[1] The code is available at `https://github.com/lancopku/Skeleton-Based-Generation-Model`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4306–4315
Brussels, Belgium, October 31 - November 4, 2018.

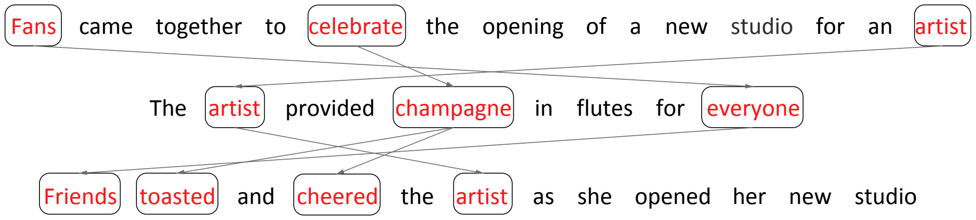

Figure 1: The semantic dependency among sentences in a narrative story. It can be seen that the connection among sentences is mainly reflected through key phrases (shown in red). In this work, we regard such key phrases as a skeleton.

based generative module and a skeleton extraction module.

The generative module consists of an input-to-skeleton component and a skeleton-to-sentence component. The input-to-skeleton component learns to associate inputs and skeletons, and the skeleton-to-sentence component learns to expand a skeleton to a sentence. In our model, a good skeleton that can capture key semantic information is a critical supervisory signal.

The skeleton extraction module is used to generate sentence skeletons. In real-world datasets, the human-annotated skeleton is usually unavailable. In addition, it is difficult to define the unified rules of extracting skeletons, because different sentences have different focuses. To address this problem, we build a skeleton extraction module to automatically explore sentence skeletons. Considering the discrete choice of skeleton words causes the loss function to be non-differentiable, we use a reinforcement learning method to build the connection between the skeleton extraction module and the generative module.

Our contributions are listed as follows:

- A skeleton-based model is proposed to promote the coherence of generated stories.
- The proposed model contains a skeleton-based generative module and a skeleton extraction module. Two modules are connected by a reinforcement learning method to automatically explore sentence skeletons.
- The experimental results on automatic evaluation and human evaluation show that our model can generate significantly more coherent text compared to the state-of-the-art models.

2 Related Work

Strictly speaking, the story generation task requires systems to generate a story from scratch without any external materials. However, for simplification, many existing story generation models rely on their given materials, such as short text descriptions (Harrison et al., 2017; Jain et al., 2017), visual images (Charles et al., 2001; Huang et al., 2016), and so on. Different from these studies, we get rid of external materials and consider the complete story generation task (McIntyre and Lapata, 2009). For this task, the widely used models are based on Seq2Seq models. However, although they can generate a fluent sentence (Xu et al., 2018a), these models still perform badly on generating inter-related sentences, which are necessary for a coherent story.

To address this problem, there are several models that build the mid-level sentence semantic representation to simplify the dependency among sentences. Clark et al. (2018) extract the entities in sentences, and combine the entity context and text context together when generating a target sentence. Cao et al. (2018) encode the words with specific pre-defined dependency labels to a mid-level sentence representation. Martin et al. (2018) use additional knowledge bases to get a generalized sentence representation. Ma et al. (2018b) use the bag-of-words which occur in all references as a representation of the correct translation. Luo et al. (2018) propose to use two auto-encoders to learn the semantic representation of utterance in dialogue. However, although these models reduce the dependency sparsity to some extent, the unified rules are non-flexible and tend to generate over-simplified representations, resulting in the loss of key information.

Different from these models, we use a reinforcement learning method to automatically extract sentence skeletons for simplifying the dependency of sentences, rather than manual rules. Therefore, our proposed skeleton-based model is more flexible and can adaptively determine the appropriate granularity of sentence representations for a balance between keeping key semantics and

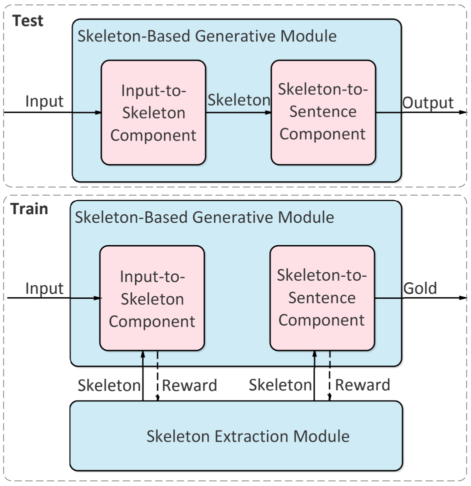

Figure 2: An illustration of the proposed model. Top: Testing. Bottom: Training. The "input" means the existing text, including the source input and the already generated text. The "skeleton" means the skeleton of the output. The skeleton extraction module first extracts skeletons from gold outputs. The pairs of input-skeleton and skeleton-gold are used to train the input-to-skeleton component and the skeleton-to-sentence component. In return, the generative module can be used to evaluate the quality of extracted skeletons. Therefore, we use the feedback of the generative module to reward extracted skeletons. By cooperation, the two modules can promote each other until convergence.

simplifying sentence representations.

3 Skeleton-Based Model

An overview of our proposed skeleton-based model is presented in Section 3.1. The details of the skeleton-based generative module and the skeleton extraction module are shown in Section 3.2 and Section 3.3. The reinforcement learning method is explained in Section 3.4.

3.1 Overview

As shown in Figure 2, our model consists of two parts, a skeleton-based generative module G_ϕ and a skeleton extraction module E_γ. The generative module consists of an input-to-skeleton component and a skeleton-to-sentence component.

The generative module generates a story sentence by sentence. When decoding a sentence, the input-to-skeleton component first generates a skeleton based on the existing text, including the source input and the already generated text, and then the skeleton-to-sentence component expands and reorganizes the skeleton into a complete sentence. We keep running this process until the generative module generates an ending symbol.

In the training process, we first use a weakly supervised method to assign the skeleton extraction module with initial extraction ability. Then, we use extracted skeletons to train the input-to-skeleton component and the skeleton-to-sentence component. In return, the generative module can be used to evaluate the quality of extracted skeletons. Therefore, we use the feedback of the generative module to reward extracted skeletons. The reward refines the skeleton extraction module. The improved skeleton extraction module further enhances the generative module. By cooperation, the two modules can promote each other until convergence.

3.2 Skeleton-Based Generative Module

The skeleton-based generative module G_ϕ consists of two parts: an input-to-skeleton component Q_α and a skeleton-to-sentence component D_θ.

3.2.1 Input-to-Skeleton Component

The input-to-skeleton component Q_α builds on a Seq2Seq structure with a hierarchical encoder (Li et al., 2015) and an attention-based decoder (Bahdanau et al., 2014). It is responsible for learning the dependency between inputs and skeletons. In the encoding process, we first obtain sentence representations via a word-level Long Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997), and then generate a compressed vector h via a sentence-level LSTM network. Finally, given the compressed vector h, the attention-based decoder is responsible for imagining a skeleton.

Given the training pair of input $\boldsymbol{c}$ and skeleton $\boldsymbol{s} = \{s_1, \cdots, s_i, \cdots, s_T\}$, the cross-entropy loss is computed as

$$L_\alpha = -\sum_{i=1}^{T} P_Q(s_i|\boldsymbol{c}, \alpha) \qquad (1)$$

where α refers to the parameters of the input-to-skeleton component. The skeleton $\boldsymbol{s}$ is extracted by the skeleton extraction module. The extracting details will be introduced in Section 3.3.

3.2.2 Skeleton-to-Sentence Component

The skeleton-to-sentence component D_θ builds on a Seq2Seq structure. Both the encoder and the decoder are one-layer LSTM networks with the attention mechanism. Given a skeleton $\boldsymbol{s}$, the encoder first generates a compressed representation, which is then used to generate a detailed and polished sentence via the decoder.

Given the training pair of skeleton $\boldsymbol{s}$ and target sentence $\boldsymbol{y} = \{y_1, \cdots, y_i, \cdots, y_M\}$, the cross-entropy loss is computed as

$$L_\theta = -\sum_{i=1}^{M} P_D(y_i|\boldsymbol{s}, \theta) \tag{2}$$

where θ refers to the parameters of the skeleton-to-sentence component.

3.3 Skeleton Extraction Module

Given a sentence $\boldsymbol{x}$, the skeleton extraction module E_γ is responsible for extracting its skeleton that only preserves the key information. Specially, we use the Seq2Seq model with the attention mechanism as the implementation. Both the encoder and the decoder are based on LSTM structures.

Since the extracted skeletons are treated as supervisory signals for the generative module, the extraction ability needs to be initialized. To pre-train the skeleton extraction module, we propose a weakly supervised method. We reformulate skeleton extraction as a sentence compression problem and use a sentence compression dataset to train this module. In sentence compression, the compressed sentence is required to be grammatical and convey the most important information. From the aspect of keeping important information, the sentence compression dataset can be used to help the training of the skeleton extraction module. However, since the style of the sentence compression dataset is very different from that of the narrative story dataset, it is difficult for the pre-trained module to give narrative text accurate compression results. Therefore, the supervisory signals are noisy and need to be further improved.

Given the training pair of the original text $\boldsymbol{x}$ and the compressed version $\boldsymbol{s} = \{s_1, \cdots, s_i, \cdots, s_T\}$, we use the following cross-entropy loss to pre-train E_γ:

$$L_\gamma = -\sum_{i=1}^{T} P_E(s_i|\boldsymbol{x}, \gamma) \tag{3}$$

where γ is the parameters of the skeleton extraction module.

3.4 Reinforcement Learning Method

We propose a reinforcement learning method to build the connection between the skeleton extraction module and the skeleton-based generative module for exploring better skeletons. The detailed training process is shown in Algorithm 1.

Algorithm 1 The reinforcement learning method for training the generative module G_ϕ and the skeleton extraction module E_γ.

```
Initialize the generative module $G_\phi$ and the skeleton extraction module $E_\gamma$ with random weights $\phi, \gamma$
Pre-train $E_\gamma$ using MLE based on Eq. 3
for each iteration $j = 1, 2, ..., J$ do
    Generate a skeleton $\boldsymbol{s}_j$ based on $E_\gamma$
    Given $\boldsymbol{s}_j$, train $G_\phi$ based on Eq. 1 and Eq. 2.
    Compute the reward $R_c$ based on Eq. 5
    Compute the gradient of $E_\gamma$ based on Eq. 4
    Update the model parameter $\gamma$
end for
```

Due to the discrete choice of words in skeletons, the loss is no longer differentiable over the skeleton extraction module. Therefore, we use policy gradient (Sutton et al., 1999) to train the skeleton extraction module.

First, we calculate a reward R_c based on the feedback of the generative module. The details of calculation process will be introduced in Section 3.4.1. Then, we optimize the parameters through policy gradient by maximizing the expected reward to train the skeleton extraction module. According to the policy gradient theorem, the gradient for the skeleton extraction module is

$$\nabla J(\gamma) = \mathbb{E}[R_c \cdot \nabla \log(P_E(\boldsymbol{s}|\boldsymbol{x}), \gamma)] \tag{4}$$

where $\boldsymbol{x}$ is the original sentence, $\boldsymbol{s}$ is the skeleton generated by a sampling mechanism.

3.4.1 Reward

To design an appropriate rewarding function, there is a critical question that needs to be considered: what will good/bad skeletons bring to the generative module.

We first define what is a good (or bad) skeleton. A good skeleton is expected to contain all key information and ignore other information. In contrast, the skeletons that contain too much detailed information or lack necessary information are considered as bad skeletons and should be punished. For ease of analysis, we classify possible scenarios into three categories: good skeletons, incomplete skeletons, and redundant skeletons.

If a skeleton contains too little information, it will get harder for the skeleton-to-sentence component to reconstruct the original sentence based on the skeleton. Therefore, the cross-entropy loss of this example will be higher compared with other skeleton-sentence pairs.

If a skeleton contains too much redundant information, the input-skeleton relation will be sparse. Therefore, the cross-entropy loss of this example will be higher compared with other input-skeleton pairs.

For a good skeleton that contains an appropriate amount of information, it will benefit the two components and will get balanced losses from the two components.

Therefore, to encourage good skeletons and punish bad skeletons, we use the multiplication of the cross-entropy loss in the input-to-skeleton component and that in the skeleton-to-sentence component as the reward:

$$R_c = [K - (R_1 \times R_2)^{\frac{1}{2}}] \tag{5}$$

where K is the upper bound of the reward, R_1 and R_2 are cross-entropy losses in the input-to-skeleton component and the skeleton-to-sentence component, respectively. Only if two components both output lower cross-entropy losses, the extracted skeleton can be rewarded.

4 Experiment

In this section, we evaluate our model on a narrative story generation dataset. We first introduce the dataset, the training details, the baselines, and the evaluation metrics. Then, we compare our model with the state-of-the-art models. Finally, we show the experimental results and provide the detailed analysis.

4.1 Dataset

We use the recently introduced visual storytelling dataset (Huang et al., 2016) in our experiments. This dataset contains the pairs of photo sequences and the associated coherent narrative of events through time written by humans. We only use the text data for our experiments. In our version of narrative story generation, the model should generate a coherent story based on a given sentence. We build a new dataset for this task by splitting the data into two parts. In each story, we take the first sentence as the input text, and the following sentences as the target text. The processed dataset contains 40153, 4990, and 5054 stories for training, validation, and testing, respectively. The maximum number of sentences in each story is 6. In total, the number of training sentences is over 20K and the number of training words is over 2M.

To pre-train the skeleton extraction module, we use a sentence compression dataset (Filippova and Altun, 2013). In this dataset, every compression is a subsequence of tokens from the input. The dataset contains 16999, 1000, and 1998 pairs for training, validation, and testing, respectively.

4.2 Baselines

We compare our proposed model with the following the state-of-the-art models.

Entity-Enhanced Seq2Seq Model (EE-Seq2Seq) (Clark et al., 2018). It regards entities as important context needed for coherent stories. When decoding a sentence, it combines entity context and text context together to reduce dependency sparsity.

Dependency-Tree Enhanced Seq2Seq Model (DE-Seq2Seq) (Cao et al., 2018). It defines some manual rules based on dependency parsing labels to find a simplified sentence representation. Following this work, we treat the extracted words based on the predefined rules as the skeleton.

Generalized-Template Enhanced Seq2Seq Model (GE-Seq2Seq) (Martin et al., 2018). It takes advantages of existing knowledge bases to get a generalized sentence representation. Following this work, we treat the generalized sentence representation as the skeleton.

4.3 Training Details

For narrative story generation, we set the number of generated sentences to 6 with the maximum length of 40 words for each generated sentence. Based on the performance on the validation set, we set the hidden size to 128, embedding size to 50, vocabulary size to 20K, and batch size to 10 for the proposed model and the state-of-the-art models. We use the Adagrad (Duchi et al., 2011) optimizer with the initial learning rate 0.6. All of the gradients are clipped when the norm exceeds 2. Both the generative module and skeleton extractive module are pre-trained for 30 and 40 epochs before reinforcement learning. The K in Equation 5 is set to 1. Due to the lack of annotated entities and dependency parsing labels, we use a popular natural language processing toolkit, Spacy[2], to extract entities and dependency parsing labels in the EE-Seq2Seq and DE-Seq2Seq models.

---

[2] https://spacy.io/

Models	BLEU
EE-Seq2Seq	0.0029
DE-Seq2Seq	0.0027
GE-Seq2Seq	0.0022
Proposed Model	**0.0042 (+44.8%)**

Table 2: Automatic evaluations of the proposed model and the state-of-the-art models.

Models	Fluency	Coherence	G-Score
EE-Seq2Seq	6.28	5.14	5.68
DE-Seq2Seq	8.48	3.54	5.48
GE-Seq2Seq	9.48	3.58	5.82
Proposed Model	8.69	5.62	**6.99 (+20.1%)**

Table 3: Human evaluations of the proposed model and the state-of-the-art models.

4.4 Evaluation Metrics

We conduct two kinds of evaluations in this work, automatic evaluation and human evaluation. The details of evaluation metrics are shown as follows.

4.4.1 Automatic Evaluation

Following the previous work (Li et al., 2016; Martin et al., 2018), we use the BLEU score to measure the quality of generated text. BLEU (Papineni et al., 2002) is originally designed to automatically judge the machine translation quality. The key point is to compare the similarity between the results created by the machine and the references provided by the human. Currently, it is widely used in many generation tasks, such as dialogue generation, story generation, summarization, and so on. For precise results, we remove all stop words, like “the”, “a”, before computing BLEU scores.

4.4.2 Human Evaluation

Although the quantitative evaluation generally indicates the quality of generated stories, it can not accurately evaluate the generated text. Therefore, we also perform a human evaluation on the test set. We randomly choose 100 items for human evaluation. Each item contains the stories generated by different models given the same source sentence. The items are distributed to the annotators who have no knowledge about which model the story is from. It is important to note that all the annotators have linguistic background. They are asked to score the generated stories in terms of fluency and coherence. Fluency represents whether each sentence in the generated story is correct in grammar. Coherence evaluates whether the generated story is coherent. The score ranges from 1 to 10 (1 is very bad and 10 is very good). To evaluate the overall performance, we use the geometric mean of fluency and coherence as an evaluation metric.

4.5 Experimental Results

Table 2 shows the results of automatic evaluation. The proposed model performs the best according to BLEU. In particular, the differences between the existing state-of-the-art models are within 0.07, while the proposed model supersedes the best of them by 0.13.

As we previously explained, the best evaluation for narrative story generation is human evaluation. The human evaluation results are listed in Table 3.[3]

As for fluency, the proposed model receives the score of 8.69, second to the GE-Seq2Seq model. It is expected that the generalized templates can constrain the search space in generation and the model achieves higher fluency by loss of expressive power. In particular, we find that only 0.48%, 1.01%, and 1.20% of the unigrams, bigrams, and trigrams are unique in the stories generated by the GE-Seq2Seq model, while the percentages are 3.16%, 15.33%, and 29.67% in the stories generated by our proposed model. Nonetheless, the proposed model outperforms the other two existing models by a substantial margin. In terms of coherence, the proposed model is better than all the existing models. We need to point out that the GE-Seq2Seq model is scored the lowest in coherence, while highest in fluency. It indicates that the GE-Seq2Seq model does not learn the dependency among sentences effectively, which results from the constraint of the templates. It also needs to be noted that the models are all scored below 6 in coherence, meaning that there is still a long way to go before the generated stories satisfy the requirement of humans. Overall, the proposed model is arguably better than the existing models in that it achieves a balance between coherence and fluency, with a G-score improvement of 20.1%.

Table 4 presents the examples generated by different models. Compared with the existing models, the sentences generated by our proposed model are connected more logically. For the EE-Seq2Seq model, while it connects *park* with *plants*

---

[3] The inter-annotator agreement is satisfactory considering the difficulty in the human evaluation. The Pearson’s correlation coefficient is 0.37 on coherence and 0.26 on fluency, with $p < 0.0001$.

and *rocks* successfully (4th ex.), it insists on telling *getting married* when it sees *[male]* or *[female]* (1st and 2nd ex.). Such examples suggest that some entities (e.g. *park*) embody semantics more independently, while for others (e.g. *male*), we have to associate them in the specific context. The rest of the models try to generalize the target sentences. The DE-Seq2Seq model uses the core dependency arguments as the skeleton. However, the results demonstrate the generated sentences are quite irrelevant. The sentence may have links such as *walked through* to *came out* (1st ex.), but the objects in the generated stories are hardly related. The GE-Seq2Seq model replaces the specific words with more general concepts and generates some good examples, e.g. the second one in the table. However, there can be over-generalizations. For example, as for the third example, the GE-Seq2Seq model associates *driving* with *car show*, causing the incoherent description. In the last example, the generated story completely diverges from the input. These results prove the drawbacks of static rule-based skeletons. The proposed model uses a skeleton extraction module to adaptively determine the appropriate granularity of skeletons. The skeleton keeps the main semantic of a sentence, which can be a whole sentence, phrases, or even segments. It makes the model learn the dependency of sentences more effectively so that the generated stories are much more coherent.

Input 1: *[Female] and her friend had a night out on the town.*
EE-Seq2Seq: *They were very happy to see us. [Male] and [female] were getting married today.*
DE-Seq2Seq: *We walked through the streets of the city and found a place to sit. The whole family came out to support their families.*
GE-Seq2Seq: *My friends and I went to a bar last night. [Female] was so happy to be there.*
Proposed Model: *They went to the bar. They had a great time.*

Input 2: *[Male] and [male] are talented artists.*
EE-Seq2Seq: *They were very happy to see us. [Male] and [female] were getting married today.*
DE-Seq2Seq:*The kids get to play with the kids. We had a great time.*
GE-Seq2Seq:*The art was very subjective. The artist had a keen eye.*
Proposed Model: *They are discussing the importance of the lecture.*

Input 3: *The bride was driving to the wedding.*
EE-Seq2Seq: *The family was so happy to be there. They were very happy to see us.*
DE-Seq2Seq: *The view from the top was amazing. There were a lot of people there.*
GE-Seq2Seq: *I went to the car show last weekend. People are getting ready to go.*
Proposed Model: *The groom and groomsmen watched the bride. They kissed and shared their pictures.*

Input 4: *The park was filled with beauty.*
EE-Seq2Seq: *The plants were so beautiful. The rocks were very pretty.*
DE-Seq2Seq: *The family was so happy to be there.*
GE-Seq2Seq: *There were a lot of people there. We had a great time.*
Proposed Model: *The trees were very sparse. There were also some flowers.*

Table 4: Examples generated by the proposed model and the state-of-the-art models.

4.6 Incremental Analysis

In this section, we conduct a series of experiments to evaluate the contributions of our key components. The results are listed in Table 5. The Seq2Seq model is scored the lowest according to BLEU. With the skeleton extraction module, the BLEU score is slightly improved, which suggests that the model starts to learn the connection of longer segments. Finally, with reinforcement learning, the BLEU score significantly overpasses the Seq2Seq model by 40%.

Models	BLEU
Seq2Seq	0.0028
+Skeleton Extraction Module	0.0029
+Reinforcement Learning	**0.0042**

Table 5: Automatic evaluations of key components.

Table 6 shows the human evaluation results. The slight improvement with the skeleton extraction module in BLEU reflects as the decreases in both fluency and coherence. It suggests the necessity of human evaluation. The decreased results can be explained by the fact that the style of the dataset for pre-training the skeleton extraction module is very different from the narrative story dataset. While it may help extract some useful skeletons, it is likely that many of them are not suitable for learning the dependency of sentences. Finally, when the skeleton extraction module is trained on the target domain using reinforcement learning, the human evaluation is improved significantly by 14% on G-score.

Table 7 further shows the results of the skeleton extraction module. As we can see, the module keeps only the essential parts of the sentence. Most of the adjectival phrases and adverbial phrases are removed. Furthermore, we can find that for longer sentences that contain too detailed information, it only extracts the key information. For shorter sentences where all information is nec-

Models	Fluency	Coherence	G-Score
Seq2Seq	7.54	4.98	6.13
+Skeleton Extraction Module	7.26	4.32	5.60
+Reinforcement Learning	8.69	5.62	**6.99**

Table 6: Human evaluations of the key components.

1) *There was a small power station on the side of the building.*
2) *The lady wearing the pink shirt decided to stop playing the video and chatted with other guests.*
3) *At the end of the night, guests taking pictures before saying goodbye to each other.*
4) *Afterwards, we celebrated with some drinks and watched the rest of the parade.*
5) *A few miles away was a lake that we really enjoyed watching.*
6) *Some of the guests partied harder than others.*
7) *The bride was driving to the wedding.*

Table 7: Analysis of the skeleton extraction module. Given a whole sentence as input, the words in red are the extracted skeleton.

essary, it choose to keep all words. It proves that the skeleton extraction module is effective and is expert in only removing detailed information that is not needed.

Furthermore, it is not quite surprising to see that on our dataset, the Seq2Seq model beats the existing state-of-the-art models (DE-Seq2Seq and GE-Seq2Seq) in human evaluation and automatic evaluation. It is mainly attributed to the over-simplification of sentences. For narrative sentences, the key information is usually expressed in a complicated way. It can be a segment, a phrase, or a whole sentence. The simple rules lead to the excessive loss of key information while our proposed model can adaptively determine the appropriate granularity.

4.7 Error Analysis

Although the proposed model outperforms the state-of-the-art models, it needs to be noted that the highest coherence score, 5.62, is a moderate result in human evaluation, indicating that there is still a long way to go before the generated stories reach the human level. Therefore, in this subsection, we give a detailed error analysis to explore what factors affect the performance.

First, we classify the generated stories with scores below 6 that are considered less coherent. We conclude 4 types of errors from these outputs and the distribution of error types are shown in Figure 3. It is expected that the irrelevant scenes make up most of the errors. In addition, there are several examples that are hard to be understood due to chaotic syntax. For the type of chaotic timeline, the model neglects the time order of scenes and the generated stories goes backward in time. The repeated scenes mean that the generated stories just describe the input again. The above errors show that there are many dimensions in coherence, including scene-specific relevance, temporal connection, and non-recurrence. Modeling such dimensions is still a hard problem.

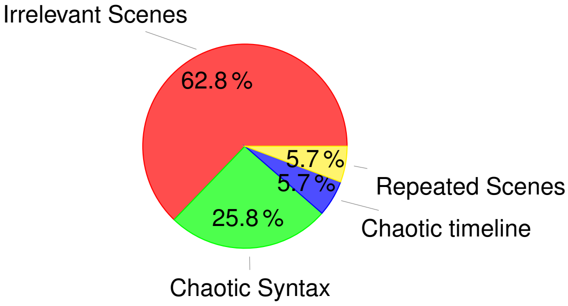

Figure 3: The distribution of error types.

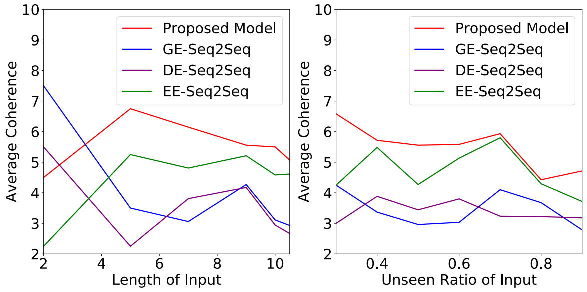

Figure 4: An illustration of how the performance is affected by the length of input (left) and the unseen ratio of input (right).

Furthermore, we explore how the performance is affected by the length of input and the unseen ratio of input. The results are shown in Figure 4. "Unseen ratio" is the percentage of the phrases that are not seen in the training data. We use the gap between 1 and the BLEU score with the training data as the reference to compute it. When the input is short and the model often sees the input, the generated story tends to have high coherence. However, when the length of input increases and the model is not familiar with the input, the coherence goes down. Since our model extracts the key semantics better, the dependency of sentences can be easier to learned, which brings the smaller decrease in coherence.

5 Conclusion and Future Work

In this work, we propose a new skeleton-based model for generating coherent narrative stories.

Different from traditional models, the proposed model first generates a skeleton that contains the key information of a sentence, and then expands the skeleton to a complete sentence. Experimental results show that our model significantly improves the quality of generated stories, especially in coherence. However, even with the best human evaluation results, the error analysis shows that there are still many challenges in narrative story generation, which we would like to explore in the future.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028). We thank all the reviewers for providing the constructive suggestions. Xu Sun is the corresponding author of this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Fred Charles, Steven J Mead, and Marc Cavazza. 2001. Character-driven story generation in interactive storytelling. In *Virtual Systems and Multimedia, 2001. Proceedings. Seventh International Conference on*, pages 609–615. IEEE.

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018. Neural text generation in stories using entity representations as context. In *NAACL HLT 2018, The 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *CoRR*, abs/1805.04833.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1481–1491.

Brent Harrison, Christopher Purdy, and Mark O Riedl. 2017. Toward automated story generation with markov chain monte carlo methods and deep neural networks. In *Proceedings of the 2017 Workshop on Intelligent Narrative Technologies*.

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

Ting-Hao K. Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Aishwarya Agrawal, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*.

Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban Laha, and Karthik Sankaranarayanan. 2017. Story generation from sequence of independent short descriptions. *CoRR*, abs/1707.05501.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. 2013. Story generation with crowdsourced plot graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1106–1115.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1192–1202.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text generation by structure-aware seq2seq learning. *CoRR*, abs/1711.09724.

Liangchen Luo, Jingjing Xu, Junyang Lin, Qi Zeng, and Xu Sun. 2018. An auto-encoder matching model for learning utterance-level semantic dependency in dialogue generation. In *EMNLP, 2018*.

Shuming Ma, Xu Sun, Junyang Lin, and Houfeng Wang. 2018a. Autoencoder as assistant supervisor: Improving text representation for chinese social media text summarization. *CoRR*, abs/1805.04869.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018b. Bag-of-words as target for neural machine translation. *CoRR*, abs/1805.04871.

Lara J. Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Neil Duncan McIntyre and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 217–225.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 1057–1063.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018a. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *EMNLP, 2018*.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018b. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *ACL, 2018*.

NEXUS Network: Connecting the Preceding and the Following in Dialogue Generation

Xiaoyu Shen[13]*; **Hui Su**[2]*; **Wenjie Li**[4] **and Dietrich Klakow**[3]
[1]Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
[2]Pattern Recognition Center, Wechat, Tencent, China
[3]Spoken Language Systems (LSV), Saarland University, Germany
[4]The Hong Kong Polytechnic University, Hong Kong
`xshen@mpi-inf.mpg.de, aaronsu@tencent.com`

Abstract

Sequence-to-Sequence (seq2seq) models have become overwhelmingly popular in building end-to-end trainable dialogue systems. Though highly efficient in learning the backbone of human-computer communications, they suffer from the problem of strongly favoring short generic responses. In this paper, we argue that a good response should smoothly connect both the preceding dialogue history and the following conversations. We strengthen this connection through mutual information maximization. To sidestep the non-differentiability of discrete natural language tokens, we introduce an auxiliary continuous code space and map such code space to a learnable prior distribution for generation purpose. Experiments on two dialogue datasets validate the effectiveness of our model, where the generated responses are closely related to the dialogue context and lead to more interactive conversations.

1 Introduction

With the availability of massive online conversational data, there has been a surge of interest in building open-domain chatbots with data-driven approaches. Recently, the neural network based sequence-to-sequence (seq2seq) framework (Sutskever et al., 2014; Cho et al., 2014) has been widely adopted. In such a model, the encoder, which is typically a recurrent neural network (RNN), maps the source tokens into a fixed-sized continuous vector, based on which the decoder estimates the probabilities on the target side word by word. The whole model can be efficiently trained by maximum likelihood (MLE) and has demonstrated state-of-the-art performance in various domains. However, this architecture is not suitable for modeling dialogues. Recent research has found that while the seq2seq model generates syntactically well-formed responses, they are prone to being off-context, short, and generic. (e.g., "I dont know" or "I am not sure") (Li et al., 2016a; Serban et al., 2016). The reason lies in the one-to-many alignments in human conversations, where one dialogue context is open to multiple potential responses. When optimizing with the MLE objective, the model tends to have a strong bias towards safe responses as they can be literally paired with arbitrary dialogue context without semantical or grammatical contradictions. These safe responses break the dialogue flow without bringing any useful information and people will easily lose interest in continuing the conversation.

A_1: Do you know the movie Star Wars?
B_1: Only a bit. You can tell me about it!
A_2: Of course! This is about ...

Figure 1: A conversation in real life

In this paper, we propose NEXUS Network which aims at producing more on-topic responses to maintain an interactive conversation flow. Our assumption is that a good response should serve as a "nexus": connecting and being informative to both the preceding dialogue context and the follow-up conversations. For example, in Figure 1, the response from B_1 is a smooth connection, where the first half indicates the preceding context is a "Do you know" question and the second half informs that the follow-up would be an introduction about *Star Wars*. We establish this connection by maximizing the mutual information (MMI) of the current utterance with both the past and future contexts. In this way, generic responses can be largely discouraged as they contain no valuable information and thus have only weak correlations

---

*Indicates equal contribution. X. Shen focuses on algorithm and H. Su is responsible for experiments.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4316–4327
Brussels, Belgium, October 31 - November 4, 2018.

with the surrounding context. To enable efficient training, two challenges exist.

The first challenge comes from the discrete nature of language tokens, hindering efficient gradient descent. One strategy is to estimate the gradient by methods like Gumbel-Softmax (Maddison et al., 2017; Jang et al., 2017) or REINFORCE algorithm (Williams, 1992), which has been applied in many NLP tasks (He et al., 2016; Shetty et al., 2017; Gu et al., 2018; Paulus et al., 2018), but the trade-off between bias and variance of the estimated gradient is hard to reconcile. The resulting model usually strongly relies on sensitive hyper-parameter tuning, careful pre-train and task-specific tricks. Li et al. (2016a); Wang et al. (2017) avoid this non-differentiability problem by learning a separate backward model to rerank candidate responses in the testing phase while still adhering to the MLE objective for training. However, the candidate set normally suffers from low diversity and a huge sample size is needed for good performance (Li et al., 2016b).

The second challenge relates to the unknown future context in the testing phase. In our framework, both the history and future context need to be explicitly observed in order to compute the mutual information. When applying it to generating tasks where only the history context is given, there is no way to explicitly take into account the future information. Therefore, reranking-based models do not apply here. (Li et al., 2016c) addresses future information by policy learning, but the model suffers from high variance due to the enormous sequential search space. Serban et al. (2017); Zhao et al. (2017); Shen et al. (2017) adopt the variational inference strategy to reduce the training variance by optimizing over latent continuous variables. However, they all stick to the original MLE objective and no connection with the surrounding context is considered.

In this work, we address both challenges by introducing an auxiliary continuous code space which is learned from the whole dialogue flow. At each time step, instead of directly optimizing discrete utterances, the current, past and future utterances are all trained to maximize the mutual information with this code space. Furthermore, a learnable prior distribution is simultaneously optimized to predict the corresponding code space, enabling efficient sampling in the testing phase without getting access to the ground-truth future conversation. Extensive experiments have been conducted to validate the superiority of our framework. The generated responses clearly demonstrate better performance with respect to both coherence and diversity.

2 Model Structure

2.1 Motivation

Let u_i be the ith utterance within a dialogue flow. The dialogue history H_{i-1} contains all the preceding context $u_1, u_2, \ldots, u_{i-1}$ and F_{i+1} denotes the future conversations $u_{i+1}, \ldots, u_T$. The objective of our model is to find the decoding probability $p_\theta(u_i|H_{i-1}, F_{i+1})$ that maximizes the mutual information $I(H_{i-1}, u_i)$ and $I(u_i, F_{i+1})$. Formally, the objective is:

$$\begin{aligned} &\max_\theta \lambda_1 I(H_{i-1}, u_i) + \lambda_2 I(u_i, F_{i+1}) \\ &\qquad u_i \sim p_\theta(u_i|H_{i-1}, F_{i+1}) \end{aligned} \tag{1}$$

λ_1 and λ_2 adjusts the relative weight. Mutual information is defined over $p_\theta(u_i|H_{i-1}, F_{i+1})$ and the empirical distribution $p(H_{i-1}, F_{i+1})$. Now we assume the future context F_{i+1} is known to us when training the decoding probability, we will address the unknown future problem later.

Directly optimizing with this objective is unfortunately infeasible because the exact computation of mutual information is intractable, and back-propagating through sampled discrete sequences is notoriously difficult to train. The discontinuity prevents the direct application of the reparameterization trick (Kingma and Welling, 2014). Low-variance relaxations like Gumbel-Softmax (Jang et al., 2017), semantic hashing (Kaiser et al., 2018) or vector quantization (van den Oord et al., 2017) lead to biased gradient estimations, which are accumulated as the sequence becomes longer. The Monte-Carlo-Simulation is unbiased but suffers from high variances. Designing a reasonable control variate for variance reduction is an extremely tricky task (Mnih and Gregor, 2014; Tucker et al., 2017). For this sake, we propose replacing u_i with a continuous code space c learned from the whole dialogue flow.

2.2 Continuous Code Space

We define the continuous code space c to follow the Gaussian probability distribution with a diagonal covariance matrix conditioning on the whole

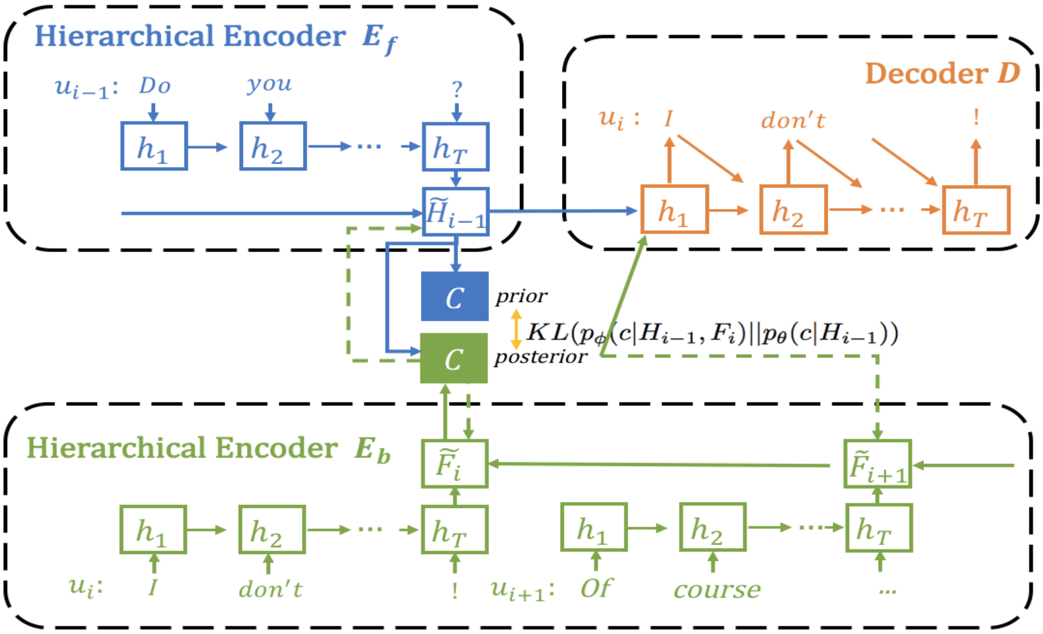

Figure 2: Framework of NEXUS Networks. Full line indicates the generative model to generate the continuous code and corresponding responses. Dashed line indicates the inference model where the posterior code is trained to infer the history, current and future utterances. Both parts are simultaneously trained by gradient descent.

dialogue:

$$c \sim p_\phi(c|H_{i-1}, F_i) = \mathcal{N}(\mu_c, \sigma_c^2 \mathbb{I}|H_{i-1}, F_i) \quad (2)$$

The dialogue history H_{i-1} is encoded into vector $\tilde{H}_{i-1}$ by a forward hierarchical GRU model E_f as in (Serban et al., 2016). The future conversation, including the current utterance, is encoded into $\tilde{F}_i$ by a backward hierarchical GRU E_b. $\tilde{H}_{i-1}$ and $\tilde{F}_i$ are concatenated and a multi-layer perceptron is built on top of them to estimate the Gaussian mean and covariance parameters. The code space is trained to infer the encoded history $\tilde{H}_{i-1}$ and future $\tilde{F}_{i+1}$. The full optimizing objective is:

$$\begin{aligned}
\mathcal{L}(c) = \max_\phi \mathbb{E}_{p_\phi(H_{i-1},F_i,c)}[\lambda_1 \log p_\phi(\tilde{H}_{i-1}|c) \\
+\lambda_2 \log p_\phi(\tilde{F}_{i+1}|c)] \\
p_\phi(H_{i-1}, F_i, c) = p(H_{i-1}, F_i) p_\phi(c|H_{i-1}, F_i) \\
p_\phi(\tilde{H}_{i-1}|c) = \mathcal{N}(\mu_{H_i}, \sigma^2_{H_i}\mathbb{I}|c) \\
p_\phi(\tilde{F}_{i+1}|c) = \mathcal{N}(\mu_{F_{i+1}}, \sigma^2_{F_{i+1}}\mathbb{I}|c)
\end{aligned} \quad (3)$$

where $\tilde{H}_{i-1}$ and $\tilde{F}_{i+1}$ are also assumed to be Gaussian distributed given c with mean and covariance estimated from multi-layer perceptrons. We infer the encoded vectors instead of the original sequences for three reasons. Firstly, inferring dense vectors is parallelizable and computationally much cheaper than autoregressive decoding, especially when the context sequences could be unlimitedly long. Secondly, sequence vectors can capture more holistic semantic-level similarity than individual tokens. Lastly, It can also help alleviate the posterior collapsing issue (Bowman et al., 2016) when training variational inference models on text (Chen et al., 2017; Shen et al., 2018), which we will use later. It can be shown that the above objective maximizes a lower bound of $\lambda_1 I(H_{i-1}, c) + \lambda_2 I(c, F_{i+1})$, given the conditional probability $p_\phi(c|H_{i-1}, F_i)$. The proof is a direct extension of the derivation in (Chen et al., 2016), followed by the Data Processing Inequality (Beaudry and Renner, 2012) that the encoding function can only reduce the mutual information. As the sampling process contains only Gaussian continuous variables, the above objective can be trained through the reparameterization trick (Kingma and Welling, 2014), which is a low-variance, unbiased gradient estimator (Burda et al., 2015). After training, samples from $p_\phi(c|H_{i-1}, F_i)$ hold high mutual information with both the history and future context. The next step is then transferring the continuous code space to reasonable discrete natural language utterances.

2.3 Decoding from Continuous Space

Our decoder transfers the code space c into the ground-truth utterance u_i by defining the probability distribution $p(u_i|H_{i-1}, c)$, which is implemented as a GRU decoder going through u_i word by word to estimate the output probability. The encoded history $\tilde{H_{i-1}}$ and code space c are concatenated as an extra input at each time step. The loss function for the decoder is then:

$$\begin{gathered}\mathcal{L}(d) = \max_{\phi} \mathbb{E}_{p_\phi(H_{i-1},F_i,c)} \log p_\phi(u_i|H_{i-1},c) \\ p_\phi(H_{i-1},F_i,c) = p(H_{i-1},F_i)p_\phi(c|H_{i-1},F_i)\end{gathered} \tag{4}$$

which can be proved to be the lower bound of the conditional mutual information $I(u_i,c|H_{i-1})$. By maximizing the conditional mutual information, c_i is trained to maintain as much information about the target sequence u_i as possible.

Combining Eq. 3 and 4, our model until now can be viewed as optimizing a lower bound of the following objective:

$$\begin{gathered}\max_{\phi} \lambda_1 I(H_{i-1},c) + \lambda_2 I(c,F_{i+1}) + I(u_i,c|H_{i-1}) \\ c \sim p_\phi(c|H_{i-1},F_i)\end{gathered} \tag{5}$$

Compared with the original motivation in Eq. 1, we sidestep the non-differentiability problem by replacing u_i with a continuous code space c, then forcing u_i to contain the same information as maintained in c by additionally maximizing the mutual information between them.

Nonetheless, Eq. 5 and Eq. 1 might lead to different optimums as mutual information does not satisfy the transitive law. In the extreme case, different dimensions of c could individually maintain information about history, current and future conversations and the conversations themselves do not share any dependency relation. To avoid this issue, we restrict the dimension of c to be smaller than that of the encoded vectors. In this case, optimizing Eq. 5 will favor utterances having stronger correlations with the surrounding context to achieve a higher total mutual information.

2.4 Learnable Prior Distribution for Unknown Future

The last problem is the sampling mechanism of c in Eq. 2, which conditions on the ground-truth future conversation. In the testing phase, when we have no access to it, we cannot perform the decoding process as in Eq. 4. To allow for decoding with only the history context, we need to learn an appropriate prior distribution $p_\theta(c|H_{i-1})$ for c. In the ideal case, we would like

$$p_\theta(c|H_{i-1}) = \sum_{F_i} p_\phi(c|H_{i-1},F_i) = p_\phi(c|H_{i-1}) \tag{6}$$

However, $p_\phi(c|H_{i-1})$ is intractable as it integrates over all possible future conversations. We apply variational inference on c to maximize the variational lower bound (Jordan et al., 1999):

$$\begin{gathered}\mathcal{L}(p) = \max_{\theta,\phi} \mathbb{E}_{p_\phi(c|H_{i-1},F_i)} \log p_\theta(\tilde{F_i}|H_{i-1},c) \\ -KL(p_\phi(c|H_{i-1},F_i)||p_\theta(c|H_{i-1})) \\ p_\theta(\tilde{F_i}|H_{i-1},c) \sim \mathcal{N}(\mu_{F_i},\sigma^2_{F_i}\mathbb{I}|H_{i-1},c) \\ p_\theta(c|H_{i-1}) \sim \mathcal{N}(\mu_{prior},\sigma^2_{prior}\mathbb{I}|H_{i-1}))\end{gathered} \tag{7}$$

It can be reformulated as maximizing:

$$\begin{gathered}\mathbb{E}_{p_\phi(c|H_{i-1})} KL(p_\phi(\tilde{F_i}|H_{i-1},c)||p_\theta(\tilde{F_i}|H_{i-1},c)) \\ -KL(p_\phi(c|H_{i-1})||p_\theta(c|H_{i-1}))\end{gathered} \tag{8}$$

We can see it implicitly matches $p_\phi(c|H_{i-1})$ to a tractable Gaussian distribution $p_\theta(c|H_{i-1})$ by minimizing the KL divergence between them. It also functions as a regularizer to prevent overfitting when learning $p_\phi(c|H_{i-1},F_i)$. In the testing phase, we can sample c from the learned prior distribution $p_\theta(c|H_{i-1})$, then generate a response based on it.

2.5 Summary

To sum up, the total objective function of our model is:

$$\mathcal{L} = \mathcal{L}(c) + \mathcal{L}(d) + \mathcal{L}(p) \tag{9}$$

Weighting can be added to individual loss functions for better performance, but we find it enough to maintain equal weights and avoid extra hyperparameters. All the parameters are simultaneously updated by gradient descent except for the encoders E_f and E_b, which only accept gradients from $\mathcal{L}(d)$ since otherwise the model can easily learn to encode no information for a lower reconstruction loss in $\mathcal{L}(c)$ and $\mathcal{L}(p)$. An overview of our training procedure is depicted in Fig. 2.

3 Relationship to Existing Methods

MMI decoding MMI decoder was proposed by (Li et al., 2016a) and further extended in (Wang et al., 2017). The basic idea is the same as our model by maximizing the mutual information with the dialogue context. However, the MMI principle is applied only at the testing phase rather than the training phase. As a result, it can only be used to evaluate the quality of a generation by estimating its mutual information with the context. To apply it in a generative task, we have to first sample some candidate responses with the seq2seq model, then rerank them by accounting for the MMI score. Our model differs from it in that we directly estimate the decoding probability thus no post-sampling rerank is needed. Moreover, we further include the future context to strengthen the connection role of the current utterances.

Conditional Variational Autoencoder The idea of learning an appropriate prior distribution in Eq. 7 is essentially a conditional variational autoencoder (Sohn et al., 2015) where the accumulated posterior distribution is trained to stay close to a prior distribution. It has also been applied in dialogue generation (Serban et al., 2017; Zhao et al., 2017). However, all the above methods stick to the MLE objective function and do not optimize with respect to the mutual information. As we will show in the experiment, they fail to learn the correlation between the utterance and its surrounding context. The generation diversity of these models comes more from the sampling randomness of the prior distribution rather than from the correct understanding of context correlation. Moreover, they suffer from the posterior collapsing problem (Bowman et al., 2016) and require special tricks like KL-annealing, BOW loss or word drop-out (Shen et al., 2018). Our model does not have such problems.

Deep Reinforcement Learning Dialogue Generation (Li et al., 2016c) first considered future success in dialogue generation and applied deep reinforcement learning to encourage more interactive conversations. However, the reward functions are intuitively hand-crafted. The relative weight for each reward needs to be carefully tuned and the training stage is unstable due to the huge search space. In contrast, our model maximizes the mutual information in the continuous space and trains the prior distribution through the reparamaterization trick. As a result, our model can be more easily trained with a lower variance. Throughout our experiment, the training process of NEXUS network is rather stable and much less data-hungry. The MMI objective of our model is theoretically more sound and no manually-defined rules need to be specified.

4 Experiments

4.1 Dataset and Training Details

We run experiments on the DailyDialog (Li et al., 2017b) and Twitter corpus (Ritter et al., 2011). DailyDialog contains 13118 daily conversations under ten different topics. This dataset is crawled from various websites for English learner to practice English in daily life, which is high-quality, less noisy but relatively smaller. In contrast, the Twitter corpus is significantly larger but contains more noise. We obtain the dataset as used in Serban et al. (2017) and filter out tweets that have already been deleted, resulting in about 750,000 multi-turn dialogues. The contents have more informal, colloquial expressions which makes the generation task harder. These two datasets are randomly separated into training/validation/test sets with the ratio of 10:1:1.

In order to keep our model comparable with the state-of-the-art, we keep most parameter values the same as in (Serban et al., 2017). We build our vocabulary dictionary based on the most frequent 20,000 words for both corpus and map other words to a UNK token. The dimensionality of the code space c is 100. We use a learning rate of 0.001 for DailyDialog and 0.0002 for Twitter corpus. The batch size is fixed to 128. The word vector dimension is 300 and is initialized with the public Word2Vec (Mikolov et al., 2013) embeddings trained on the Google News Corpus. The probability estimators for the Gaussian distributions are implemented as 3-layer perceptrons with the hyperbolic tangent activation function. As mentioned above, when training NEXUS models, we block the gradient from $\mathcal{L}(c)$ and $\mathcal{L}(p)$ with respect to E_f and E_b to encourage more meaningful encodings. The UNK token is prevented from being generated in the test phase. We implemented all the models with the open-sourced Python library Pytorch (Paszke et al., 2017) and optimized using the Adam optimizer (Kingma and Ba, 2015).

4.2 Compared Models

We conduct extensive experiments to compare our model against several representative baselines.

Seq2Seq: Following the same implementation as in (Vinyals and Le, 2015), the seq2seq model serves as a baseline. We try both greedy decoding

Model	DailyDialog			Twitter		
	Average	Greedy	Extreme	Average	Greedy	Extreme
Greedy	0.443	0.376	0.328	0.510	0.341	0.356
Beam	0.437	0.350	0.369	0.505	0.345	0.352
MMI	0.457	0.371	0.371	0.518	0.353	0.365
RL	0.405	0.329	0.305	0.460	0.349	0.323
VHRED	**0.491**	0.375	0.313	0.525	0.389	0.372
NEXUS-H	0.479	0.381*	**0.385***	**0.558***	0.392	0.373
NEXUS-F	0.476	0.383*	0.373	0.549*	0.393	0.386*
NEXUS	0.488	**0.392***	0.384*	0.556*	**0.397***	**0.391***

Table 1: Results of embedding-based metrics. * indicates statistically significant difference ($p < 0.05$) from the best baselines. The same mark is used in Table 2

and beam search (Graves, 2012) with beam size set to 5 when testing.

MMI: We implemented the bidirectional-MMI decoder as in Li et al. (2016a), which showed better performance over the anti-LM model. The hyperparameter λ is set to 0.5 as suggested. 200 candidates per context are sampled for re-ranking.

VHRED: The VHRED model is essentially a conditional variational autoencoder with hierarchical encoders (Serban et al., 2017; Zhao et al., 2017). To alleviate the posterior collapsing problem, we apply the KL-annealing trick and early stop with the step set as 12,000 for the DailyDialog and 75,000 for the Twitter corpus.

RL: Deep reinforcement learning chatbot as in (Li et al., 2016c). We use all the three reward functions mentioned in the paper and keep the relative weights the same as in the original paper. Policy network is initialized with the above-mentioned MMI model.

NEXUS-H: NEXUS network maximizing mutual information only with the history ($\lambda_2 = 0$).

NEXUS-F: NEXUS network maximizing mutual information only with the future ($\lambda_1 = 0$).

NEXUS: NEXUS network maximizing mutual information with both the history and future.

NEXUS-H and NEXUS-F are implemented to help us better analyze the effects of different components in our model. The hyperparameters λ_1 and λ_2 in NEXUS are set to be 0.5 and 1 respectively as we find history vector is consistently easier to be reconstructed than the future vector (A.6).

4.3 Metric-based Performance

Embedding Score We conducted three embedding-based evaluations (average, greedy and extrema) (Liu et al., 2016), which map responses into vector space and compute the cosine similarity (Rus and Lintean, 2012). The embedding-based metrics can to a large extent capture the semantic-level similarity between generated responses and ground truth. We represent words using Word2Vec embeddings trained on the Google News Corpus. We also measure the uncertainty of the score by assuming each data point is independently Gaussian distributed. The standard deviation yields the 95% confidence interval (Barany et al., 2007). Table 1 reports the embedding scores on both datasets. NEXUS network significantly outperforms the best baseline model in most cases. Notably, NEXUS can absorb the advantages from both NEXUS-H and NEXUS-F. The history and future information seem to help the model from different perspectives. Taking into account both of them does not create a conflict and the combination leads to an overall improvement. RL performs rather poorly on this metric, which is understandable as it does not target the ground-truth responses during training (Li et al., 2016c).

BLEU Score BLEU is a popular metric that measures the geometric mean of the modified n-gram precision with a length penalty (Papineni et al., 2002). Table 2 reports the BLEU 1-3 scores. Compared with embedding-based metrics, the BLEU score quantifies the word-overlap between generated responses and the ground-truth. One challenge of evaluating dialogue generation by BLEU score is the difficulty of accessing multiple references for the one-to-many alignment relation. Following Sordoni et al. (2015); Zhao et al.

Model	**DailyDialog**			**Twitter**		
	BLEU-1	**BLEU-2**	**BLEU-3**	**BLEU-1**	**BLEU-2**	**BLEU-3**
Greedy	0.394	0.245	0.157	0.340	0.203	0.116
Beam	0.386	0.251	0.163	0.338	0.205	0.112
MMI	0.407	0.269	0.172	0.347	0.208	0.118
RL	0.298	0.186	0.075	0.314	0.199	0.103
VHRED	0.395	**0.281**	0.190	0.355	0.211	0.124
NEXUS-H	0.418	0.279	**0.199***	**0.366***	0.212	0.126
NEXUS-F	0.399	0.260	0.167	0.359	0.213	0.123
NEXUS	**0.424***	0.276	0.198*	0.363*	**0.220***	**0.131***

Table 2: Results of BLEU score. It is computed based on the smooth BLEU algorithm (Lin and Och, 2004). p-value interval is computed base on the altered bootstrap resampling algorithm (Riezler and Maxwell, 2005)

(2017); Shen et al. (2018), for each context, 10 more candidate references are acquired by using information retrieval methods (see Appendix A.4 for more details). All candidates are then passed to human annotators to filter unsuitable ones, resulting in 6.74 and 5.13 references for DailyDialog and Twitter dataset respectively. The human annotation is costly, so we evaluate it on 1000 sampled test cases for each dataset. As the BLEU score is not the simple mean of individual sentence scores, we compute the 95% significance interval by bootstrap resampling (Koehn, 2004; Riezler and Maxwell, 2005). As can be seen, NEXUS network achieves best or near-best performances with only greedy decoders. NEXUS-H generally outperforms NEXUS-F as the connection with future context is not explicitly addressed by the BLEU score metric. MMI and VHRED bring minor improvements over the seq2seq model. Even when evaluated on multiple references, RL still performs worse than most models.

Connecting the preceding We define two metrics to evaluate the model's capability of "connecting the preceding context": **AdverSuc** and **Neg-PMI**. AdverSuc measures the coherence of generated responses with the provided context by learning an adversarial discriminator (Li et al., 2017a) on the same corpus to distinguish coherent responses from randomly sampled ones. We encode the context and response separately with two different LSTM neural networks and output a binary signal indicating coherent or not[1]. The AdverSuc value is reported as the success rate that the model fools the classifier into believing its false generations ($p(generated = coherent) > 0.5$). Neg-PMI measures the negative pointwise mutual information value $-\log p(c|r)/p(c)$ between the generated response r and the dialogue context c. $p(c|r)$ is estimated by training a separate backward seq2seq model. As $p(c)$ is a constant, we ignore it and only report the value of $-\log p(c|r)$. A good model should achieve a higher AdverSuc and a lower Neg-PMI. The results are listed in Table 3. We can see there is still a big gap between ground-truth and synthesized responses. As expected, NEXUS-H leads to the most significant improvement. MMI model also performs remarkably well, but it requires post-reranking thus the sampling process is much slower. VHRED and NEXUS-F do not help much here, sometimes even slightly degrade the performance. We also tried removing the history context when computing the posterior distribution in VHRED, the resulting model has similar performance among all metrics, which suggests VHRED itself cannot actually learn the correlation pattern with the preceding context. Surprisingly, though RL explicitly set the coherence score as a reward function, its performance is far from satisfying. We assume RL requires much more data to learn the appropriate policy than other models and the training process suffers from a higher variance. The result is thus hard to be guaranteed.

Connecting the following We measure the model's capability of "connecting the following context" from two perspectives: number of the simulated turns and diversity of generated re-

---

[1] We apply the same architecture as in Lu et al. (2017). In our experiment, the discriminator performs reasonably well in the 4 scenarios outlined in Li et al. (2017a) and thus can be used as a fair evaluation metric.

Model	AdverSuc	Neg-PMI	#Turns	Distinct-1	Distinct-2	Pri	Post	Flu
Greedy	0.21|0.13	47.4|45.8	0.2|0.6	.019|.017	.096|.072	0.45	0.04	0.92
Beam	0.16|0.12	47.2|45.3	0.2|0.7	.026|.019	.103|.086	0.52	0.06	0.90
MMI	0.30|0.19	45.6|43.2	1.1|1.6	.042|.025	.247|.117	0.56	0.13	0.89
RL	0.13|0.11	45.0|42.6	2.3|2.3	.048|.033	.324|.287	0.46	0.15	0.69
VHRED	0.19|0.16	46.8|44.7	1.7|1.1	.255|.106	.431|.311	0.42	0.22	0.92
NEXUS-H	**0.36**|**0.21**	**44.1**|41.8	2.0|1.8	.263|.108	.454|.306	0.66	0.20	0.92
NEXUS-F	0.22|0.12	47.1|45.9	2.6|2.2	**.288**|.117	.466|.325	0.51	0.31	**0.94**
NEXUS	0.35|0.18	44.6|**41.4**	**2.8**|**2.5**	.282|**.119**	**.470**|**.329**	**0.70**	**0.33**	0.93
GROUND	0.87|0.73	40.5|38.1	4.8|4.0	.390|.215	.522|.495	0.92	0.67	0.97

Table 3: Coherence, diversity and human evaluations. Left: DailyDialog results, right: Twitter results

sponses. We apply all models to generate multiple turns until a generic response is reached. The set of generic responses is manually examined to include all utterances providing only passive dull replies[2]. The number of generated turns can reflect the time that a model can maintain an interactive conversation. The results are reflected in the **#Turns** column in Table 3. As in (Li et al., 2016a), we measure the diversity by the percentage of distinct unigrams (**Distinct-1**) and bigrams (**Distinct-2**) in all generated responses. Intuitively a higher score on these three metrics implies a more interactive generation system that can better connect the future context. Again, NEXUS network dominates most fields. NEXUS-F brings more impact than NEXUS-H as it explicitly encourages more interactive turns. Most seq2seq models fail to provide an informative response in the first turn. The MMI-decoder does not change much, possibly because the sampling space is not large enough, a more diverse sampling mechanism (Vijayakumar et al., 2018) might help. NEXUS network can effectively continue the conversation for 2.8 turns for DailyDialog and 2.5 turns for Twitter, which is closest to the ground truth (4.8 and 4.0 turns respectively). It also achieves the best diversity score in both datasets. It is worth mentioning that NEXUS-H also improves over baselines, though not as significantly as NEXUS-F, so NEXUS is not a trade-off but more like an enhanced version from NEXUS-H and NEXUS-F.

In summary, NEXUS network clearly generates higher-quality responses in both coherence and diversity, even in a rather small dataset like DailyDialog. NEXUS-H contributes more to the coherence and NEXUS-F more to the diversity.

4.4 Human Evaluation

We also employed crowdsourced judges to provide evaluations for a random sample of 500 items in the DailyDialog test dataset. Participants are asked to assign a binary score to each context-response pair from three perspectives: whether the response coincides with its preceding context (Pri), whether the response is interesting enough for people to continue (Post) and whether the response itself is a fluent natural sentence (Flu). Each sample gets one point if judged as yes and zero otherwise. Each pair is judged by three participants and the score supported by most people is adopted. We also evaluated the inter-annotator consistency by Fleiss'k score(Fleiss, 1971) and obtained k scores of 0.452 for Pri, 0.459 for Post (moderate agreement) and 0.621 for Flu (substantial agreement), which implies most context-response pairs reach a consensus on the evaluation task. We compute the average human score for each model. Unlike metric-based scores, the human evaluation is conducted only on the DailyDialog corpus as it contains less noise and can be more fairly evaluated by human judges. Table 3 shows the result in the last three columns. As can be seen, the pri and post human scores are highly correlated with the automatic evaluation metric "coherence" and "#turns", verifying the validity of these two metrics. As for fluency, there is no significant difference among most models. As we also manually examined, fluency is not a major problem and all models produce mostly well-formed sentences. Overall, NEXUS network does produce responses that are more acceptable to human judges.

Table 4 presents some randomly sampled context-response pairs provided by MMI,

---

[2] We use a simple rule matching method (see Appendix A.5). We manually inspect it on a validation subset and find the accuracy is more than 90%. Similar methods are adopted in (Li et al., 2016c).

Context	**Response**
Hi , Alice , what are you going to do this sunday ? __eou__	**MMI:** Oh , that' s great ! **VHRED :** Well , it seems that i'm going to be late . **RL :** How long he it be ? **NEXUS:** Nothing really , what ' s up ?
Did you watch the Oscars on Sunday night ? __eou__	**MMI:** I have no idea . **VHRED:** That's not too bad . **RL :** It's my mum's birthday on Sunday . **NEXUS:** No , nothing much. Could you help me ?
Let's go for a drink ! __eou__	**MMI:** Thanks . **VHRED:** all right . let me see you then . **RL :** Thank you inviting me . **NEXUS:** Sounds good . Where are we going ?
You won't want to miss our webinar tomorrow ! __eou__	**MMI:** lol ! **VHRED:** That ' s terrific ! **RL :** Sure, where where ? **NEXUS:** Thanks for your invitation! I'm free !
How I read the article ? __eou__	**MMI:** there!!! **VHRED:** What ' s good with it ? **RL :** : I dont know what to do with it ! **NEXUS:** Maybe the force is yourself !

Table 4: Examples of context-response pairs. __eou__ denotes end-of-utterance. First three rows are from DailyDialog and the last two rows are from Twitter

VHRED, RL and NEXUS model. We see NEXUS network does generate more interactive outputs than the other three. Though reranked by the bidirectional language model, the MMI decoder still produces quite a few generic responses. VHRED's utterances are more diverse, but it only cares about answering to the immediate query and makes no efforts to bring about further topics. Moreover, it also generates more inappropriate responses than the others. RL provides diverse responses but sometimes not fluent or coherent enough. We do observe that NEXUS sometimes generate over-complex questions which are not very natural, as in the second example. But in most cases, it outperforms the others.

5 Conclusion

In this paper, we propose "NEXUS Network" to enable more interactive human-computer conversations. The main goal of our model is to strengthen the "nexus" role of the current utterance, connecting both the preceding and the following dialogue context. We compare our model with MMI, reinforcement learning and CVAE-based models. Experiments show that NEXUS network consistently produces higher-quality responses. The model is easier to train, requires no special tricks and demonstrates remarkable generalization capability even in a very small dataset.

Our model can be considered as combining the objective of MMI and CVAE and is compatible with current improving techniques. For example, mutual information can be maximized under a tighter bound using Donsker-Varadhan or f-divergence representation (Donsker and Varadhan, 1983; Nowozin et al., 2016; Belghazi et al., 2018). Extending the code space distribution to more than Gaussian by importance weighted autoencoder (Burda et al., 2015), inverse autoregressive flow (Kingma et al., 2016) or VamPrior (Tomczak and Welling, 2018) should also help with the performance.

Acknowledgments

We thank all anonymous reviewers, Gerhard Weikum, Jie Zhou, Cheng Niu and the dialogue system team of Wechat AI for valuable comments. Xiaoyu Shen is supported by IMPRS-CS fellowship. This work is partially funded by DFG collaborative research center SFB 1102 and Research Grants Council of Hong Kong (PolyU 152036/17E, 152040/18E).

References

Imre Barany, Van Vu, et al. 2007. Central limit theorems for gaussian polytopes. *The Annals of Probability*, 35(4):1593–1621.

Normand J Beaudry and Renato Renner. 2012. An intuitive proof of the data processing inequality. *Quantum Information & Computation*, 12(5-6):432–441.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Devon Hjelm, and Aaron Courville. 2018. Mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 531–540, Stockholmsmssan, Stockholm Sweden. PMLR.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. 2015. Importance weighted autoencoders. *CoRR*, abs/1509.00519.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017. Variational lossy autoencoder. *ICLR*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Monroe D Donsker and SR Srinivasa Varadhan. 1983. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711.

Jiatao Gu, Daniel Jiwoong Im, and Victor OK Li. 2018. Neural machine translation with gumbel-greedy decoding. *AAAI*, pages 5125–5132.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *ICLR*.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. 1999. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2390–2399, Stockholmsmssan, Stockholm Sweden. PMLR.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *ICLR*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A simple, fast diverse decoding algorithm for neural generation. *CoRR*, abs/1611.08562.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Jiwei Li, Will Monroe, Tianlin Shi, Sèbastien Jean, Alan Ritter, and Dan Jurafsky. 2017a. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017b. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 986–995.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, page 501. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Yichao Lu, Phillip Keung, Shaonan Zhang, Jason Sun, and Vikas Bhardwaj. 2017. A practical approach to dialogue response generation in closed domains. *CoRR*, abs/1703.09439.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. *ICLR*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ICLR workshop*.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pages 1791–1799.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279.

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS workshop*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *ICLR*.

Stefan Riezler and John T Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for mt. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 57–64.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162. Association for Computational Linguistics.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3783. AAAI Press.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 3295–3301.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 504–509.

Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018. Improving variational encoder-decoders in dialogue generation. *AAAI*, pages 5456–5463.

Rakshith Shetty, Marcus Rohrbach, Lisa Anne Hendricks, Mario Fritz, and Bernt Schiele. 2017. Speaking the same language: Matching machine to human captions by adversarial training. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jakub Tomczak and Max Welling. 2018. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. 2017. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pages 2627–2636.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasaath R Selvaraju, Qing Sun, Stefan Lee, David J Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. *AAAI*, pages 7371–7379.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2140–2150.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–664.

A Neural Local Coherence Model for Text Quality Assessment

Mohsen Mesgar [*]
Heidelberg Institute for
Theoretical Studies (HITS) and
Research Training Group AIPHES

Michael Strube
Heidelberg Institute for
Theoretical Studies (HITS)
`michael.strube@h-its.org`

Abstract

We propose a local coherence model that captures the flow of what semantically connects adjacent sentences in a text. We represent the semantics of a sentence by a vector and capture its state at each word of the sentence. We model what relates two adjacent sentences based on the two most similar semantic states, each of which is in one of the sentences. We encode the perceived coherence of a text by a vector, which represents patterns of changes in salient information that relates adjacent sentences. Our experiments demonstrate that our approach is beneficial for two downstream tasks: Readability assessment, in which our model achieves new state-of-the-art results; and essay scoring, in which the combination of our coherence vectors and other task-dependent features significantly improves the performance of a strong essay scorer.

1 Introduction

Coherence is a key factor that distinguishes well-written texts from random collections of sentences. A potential application of coherence models is text quality assessment. Examples include readability assessment (Pitler and Nenkova, 2008; Li and Hovy, 2014) and essay scoring (Miltsakaki and Kukich, 2004; Burstein et al., 2010). Here, we address the problem of local coherence modeling, which captures text relatedness at the level of sentence-to-sentence transitions.

Several approaches to local coherence modeling have been proposed. Entity-based methods principally relate adjacent sentences by means of entities, which are mentioned as noun phrases, NPs, in sentences (Barzilay and Lapata, 2008; Elsner and Charniak, 2011; Guinaudeau and Strube, 2013; Tien Nguyen and Joty, 2017). Lexical models connect sentences based on semantic relations between words in sentences (Beigman Klebanov and Shamir, 2006; Heilman et al., 2007; Mesgar and Strube, 2016). Both of these approaches suffer from different weaknesses. The entity-based models require an entity detection system, a coreference model, and a syntactic parser. These subsystems need to be perfect to gain the best performance of entity-based coherence models. The weakness of the lexical models is that they consider words independently, i.e. regardless of context in that words appear. More concretely, such lexical models take sentences as a bag of words. Recent deep learning coherence work (Li and Hovy, 2014; Li and Jurafsky, 2017) adopts recursive and recurrent neural networks for computing semantic vectors for sentences. Coherence models that use recursive neural networks suffer from a severe dependence on external resources, e.g. a syntactic parser to construct their recursion structure. Coherence models that purely rely on the recurrent neural networks process words sequentially within a text. However, in such models, long-distance dependencies between words cannot be captured effectively due to the limits of the memorization capability of recurrent networks.

Our motivation is to overcome these limitations. We use the advantages of distributional representations in order to, first, identify and represent salient semantic information that connects sentences, and second, extract patterns of changes in such information as a text progresses. By representing words of sentences with their pre-trained embeddings, we take lexical semantic relations between words into account. We employ a Recurrent Neural Network (RNN) layer to combine information in word embeddings and actual context information of words in sentences. Our model encodes salient information that relates two adja-

---

[*] This author is currently employed by the Ubiquitous Knowledge Processing (UKP) Lab, Technische Universität Darmstadt, `https://www.ukp.tu-darmstadt.de`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4328–4339
Brussels, Belgium, October 31 - November 4, 2018.

cent sentences based on the two most similar RNN states in sentences. We accumulate two identified RNN states to represent semantic information that connects two adjacent sentences. We encode pattern of semantic information changes across sentences in a text by a convolutional neural network to represent coherence. Our end-to-end coherence model is superior to previous work because it relates sentences based on two semantic information states in sentences that are highly similar. So it does not need extra tools such as coreference resolution systems. Furthermore, our model incorporates words in their sentence context and models (roughly) distant relations between words.

We evaluate our model on two tasks: readability assessment and essay scoring. Both have been frequently used for coherence evaluation (Barzilay and Lapata, 2008; Miltsakaki and Kukich, 2004). Readability assessment is a ranking task where we compare the rankings given by the model against human judgments. Essay scoring is a regression task, in which we investigate if the combination of coherence vectors produced by our model and other essay scoring features proposed by Phandi et al. (2015) improves the performance of the essay scorer. The experimental results show that our model achieves the state-of-the-art result for readability assessment on the examined dataset (De Clercq and Hoste, 2016); and the combination of our coherence features with other essay scoring features significantly improves the performance of the examined essay scorer (Phandi et al., 2015).

2 Related Work

Early work on coherence captures different types of relations: entity-based (Grosz et al., 1995; Barzilay and Lapata, 2008), lexical-based (Beigman Klebanov and Flor, 2013; Somasundaran et al., 2014; Zhang et al., 2015), etc. Among these models, the entity-grid model (Barzilay and Lapata, 2005, 2008) has received a lot of attention. In this model, entities are defined, heuristically, by applying a string match over head nouns of all NPs in a text. The model, then, defines all possible changes over syntactic roles of entities in adjacent sentences as coherence patterns. The entity-grid model has been extended both by expanding its entity extraction phase (Elsner and Charniak, 2011; Feng and Hirst, 2012) and by defining other types of patterns (Lin et al., 2011; Louis and Nenkova, 2012; Ji and Eisenstein, 2014; Guinaudeau and Strube, 2013). Recently, Tien Nguyen and Joty (2017) fed entity grid representations of texts to a convolutional neural network (CNN) in order to overcome the limitation of predefined coherence patterns and extract patterns automatically. However, all of these models limit relations between sentences to entities that are shared by sentences. This makes the performance of these models dependent on the performance of other tools like coreference resolution systems and syntactic parsers. Our coherence model, in contrast, is based on relations between any embedded semantic information in sentences, and does not require entity annotations. A similar approach to ours is proposed by Mesgar and Strube (2016). Their approach encodes lexical relations between sentences in a text via a graph. Sentences are encoded by nodes, and lexical semantic relations between sentences are represented by edges. Coherence patterns are obtained by applying a subgraph mining method to graph representations of all texts in a corpus. This model involves words individually and independent of their sentence context. Our model uses a RNN layer over words in sentences to incorporate context information. Our approach for extracting coherence patterns also differs from this model as we employ CNNs rather than graph mining. Li and Hovy (2014) model sentences as vectors derived from RNNs and train a feed-forward neural network that takes an input window of sentence vectors and assigns a probability which represents the coherence of the sentences in the window. Text coherence is evaluated by sliding the window over sentences and aggregating their coherence probabilities. Similarly, Li and Jurafsky (2017) study the same model at a larger scale and use a sequence-to-sequence approach in which the model is trained to generate the next sentence given the current sentence and vice versa. Our approach differs from these methods; we represent coherence by a vector of coherence patterns. Moreover, our model takes distant relations between words in a text into account by relating two semantic states of sentences that are highly similar. Lai and Tetreault (2018) compare the performance of the aforementioned coherence models on texts from different domains. They conclude that the neural coherence models, which are explained above, surpass examined non-neural coherence models such as the entity-based models and the lexical-based model. Unlike their evaluation method, which predicts the coherence level of a text, we rank two texts with respect to their coherence levels for the readability assessment task. We also show that integrating our coherence model into an essay scorer improves its performance.

An important task for evaluating a coherence model is readability assessment (Li and Hovy, 2014; Petersen et al., 2015; Todirascu et al., 2016). The more coherent a text, the faster to read and easier to understand it is. Early readability formulas were based on superficial text features such as average word lengths (Kincaid et al., 1975). These formulas systematically ignore many important factors that affect readability such as discourse coherence (Barzilay and Lapata, 2008). Schwarm and Ostendorf (2005) and Feng et al. (2010) recast readability assessment as a ranking task, and employ different semantic (e.g. language model perplexity scores) and syntactic (e.g. the average number of NPs) features to solve this task. Pitler and Nenkova (2008) show that discourse coherence features are more informative than other features for ranking texts with respect to their readability. Following the related work on coherence modeling (Barzilay and Lapata, 2008; Mesgar and Strube, 2015), we evaluate our coherence model on this task.

Another popular task for evaluating coherence models is essay scoring (Beigman Klebanov and Flor, 2013; Somasundaran et al., 2014). Miltsakaki and Kukich (2004) employ an essay scoring system to examine whether local coherence features, as defined by a measure of Centering Theory's Rough-Shift transitions (Grosz et al., 1995), might be a significant contributor to the evaluation of essays. They show that adding such features to their essay scorer improves its performance significantly. Burstein et al. (2010) specifically focus on the impact of entity transition features, as proposed by the entity-grid model for coherence modeling, on the essay scoring task. They demonstrate that by combining these features with other features related to grammar errors and word usage, the performance of their automated essay scoring system improves. Likewise, we combine our coherence vectors with other features that are used by a strong essay scorer (Phandi et al., 2015) and show that our coherence vectors improve the performance of this system significantly.

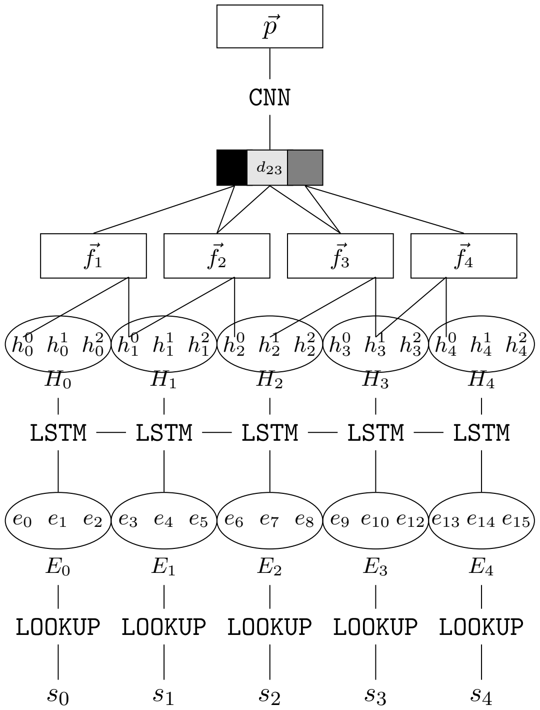

Figure 1: An illustration of our model. e_k is word embeddings associated with the k^{th} word in an input text. h_i^j depicts the j^{th} hidden state in LSTM states of sentence s_i. Two states in LSTM states of sentence s_i and sentence s_{i-1} that have the highest similarity are selected to connect sentences. Vector $\vec{f_i}$ captures information about the salient topic that relates sentence s_i to sentence s_{i-1}. d_{23} represents the similarity between $\vec{f_2}$ and $\vec{f_3}$ or the degree of continuity of the topic over adjacent sentences. Different shades of gray show different degrees of similarity. The `CNN` encodes patterns of changes as coherence vector $\vec{p}$.

3 Coherence Model

In this section, we describe details of our model. First, we explain how we encode words in their context (Section 3.1). Then we show how we relate sentences (Section 3.2), and finally we explain how we represent coherence based on sentence relations (Section 3.3). A general formulation of our model is a parametric function, $\vec{p} = L_\theta(d)$, where d is an input document, θ indicates parameters of neural modules, and $\vec{p}$ is a vector representation for the coherence of d. Figure 1 illustrates our model.

3.1 Word and Context Representations

We use a lookup table to associate all words in the vocabulary with word embeddings. The lookup table is initialized by existing pre-trained word embeddings because they capture lexical semantic relations between words. For sentence s_i, the lookup table returns matrix E_i whose rows are embeddings of words in s_i. A weakness of former lexical coherence models (Somasundaran et al., 2014; Mesgar and Strube, 2016) is that they only rely on semantic relations between words in sentences, regardless of the current context of words. In order to overcome this limitation, we use a standard unidirectional[1] RNN with Long Short-Term Memory (LSTM) cells to encode the current context of words in sentences. For embedding matrix E_i:

$$H_i = \texttt{LSTM}\left(E_i, h_{i-1}^{n-1}\right),$$

where H_i is a list of LSTM states, and h_{i-1}^{n-1} is the last LSTM state of sentence s_{i-1}. Parameter n is the number of words in a sentence. We take state vector $h_i^j \in H_i$ as a representation of its input word embedding, e_j, that is combined with its preceding word vectors in sentence s_i. For sake of brevity, the details of `LSTM` formulations are explained in Appendix A.

3.2 Sentence Relation Representations

The relation between sentences is encoded by the most similar semantic states of sentences. Given two adjacent sentences, two of their LSTM states that have the highest similarity are selected to connect them. Those LSTM states refer to the salient semantic information that is shared between sentences. To model this, we follow attention components in neural language models (Bahdanau et al., 2014; Vaswani et al., 2017) where the similarity between the last LSTM state and each of its preceding states is computed to measure the amount of attention that the model should give to its preceding context for generating the next word. More formally, for two adjacent sentences s_i and s_{i-1}, one LSTM state in H_i and one LSTM state in H_{i-1} that have the maximum similarity are selected to represent the relation between the sentences:

$$(\vec{u}, \vec{v}) = \underset{\substack{(\vec{h}_m \in H_i) \\ (\vec{h}_n \in H_{i-1})}}{\operatorname{argmax}} \left(sim(\vec{h}_m, \vec{h}_n)\right),$$

where H_i and H_{i-1} are LSTM states corresponding to sentences s_i and s_{i-1}. The similarity function, sim, returns the absolute value of the dot product between input vectors,

$$sim(\vec{h}_m, \vec{h}_n) = |\vec{h}_m \cdot \vec{h}_n|, \tag{1}$$

where the function $|.|$ computes the absolute value of its input[2]. We use the dot product function because in practice it enables our model to calculate the above equations efficiently in parallel and in matrix-space, i.e., directly on H_i and H_{i-1}. Since this is the details of implementation, we explain matrix-based equations in Appendix B. The absolute value in the similarity function is used to encode semantic relatedness between associated information with vectors, which is independent of the sign of the similarity function (Manning and Schütze, 1999).

We represent semantic information that relates two adjacent sentences by accumulating its selected LSTM states in the corresponding sentences. Since averaging in the vector space is an effective way to accumulate information represented in some vectors (Iyyer et al., 2015; Wieting et al., 2016), we compute the average of two identified vectors among the LSTM states of two adjacent sentences to represent semantic information shared by the sentences. More concretely, the vector representation of what relates sentence s_i to its immediately preceding sentence is obtained by averaging a vector of H_i and a vector of H_{i-1} that are identified as highly similar:

$$\vec{f_i} = avg(\vec{u}, \vec{v}) = \frac{\vec{u} + \vec{v}}{2},$$

where $\vec{u}$ and $\vec{v}$ are selected vectors. $\vec{f_i}$ is the vector representation of what connects s_i to its immediately preceding sentence.

3.3 Coherence Representations

Since sentences in a coherent text are about similar topics and share some semantic information, we compute semantic similarity between adjacent information states, i.e. $\vec{f_i}$s, to capture how they are changing through a text. We propose to encode changes by a continuous value between 0 and 1, where 1 shows that there is no change and 0 indicates that there is a big semantic drift in a text. Any value in between depicts how far a text is semantically changing. Given two adjacent vectors $\vec{f_i}$ and $\vec{f}_{i+1}$, the degree of continuity between them is:

---

[1] We use unidirectional RNN to model the way that an English text is read.

[2] In practice, the absolute function is implemented as $g(z) = \max(0, z) - \min(0, z)$ to be differentiable.

$$d = \frac{sim(\vec{f_i}, \vec{f_{i+1}})}{l},$$

where l is the length of input vectors, which is used to prevent large numbers (Vaswani et al., 2017), and sim is the similarity function (Section 3.2). The task of this layer is to check if the salient information that is shared by two adjacent sentences is salient in the subsequent sentence or not.

The last layer of our model is a convolutional layer to automatically extract and represent patterns of semantic changes in a text. CNNs have proven useful for various NLP tasks (Collobert et al., 2011; Kim, 2014; Kalchbrenner et al., 2014; Cheng and Lapata, 2016) because of their effectiveness in identifying patterns in their input (Xu et al., 2015). In the case of coherence, the convolution layer can identify coherence patterns that correlate with final tasks (Tien Nguyen and Joty, 2017). We use a temporal narrow convolution by applying a kernel filter k of width h to a window of h adjacent transitions over sentences to produce a new coherence feature. This filter is applied to each possible window of transitions in a text to produce a feature map $\vec{p}$, which is a coherence vector. Since we use a standard convolution layer, we explain details of the CNN formulations in Appendix C.

3.4 Variants of Our Model

In our experiments, we consider two variants of our model: **CohLSTM** that is the full version of our model as described above; and **CohEmb** that is an ablation. CohEmb has no RNN layer, so the model is built directly on word embeddings. In this model, relations between sentences are made over only content words by eliminating all stop words.

4 Implementation Details

Model configurations. Our model is implemented in PyTorch[3] with CUDA 8.0 support. In preprocessing we apply zero-padding to all sentences and documents to make their length equal. The vocabulary is limited to the 4000 most frequent words in the training data and all other words are replaced with the unknown token. We use the pre-trained word embeddings released by Zou et al. (2013), which are employed by state-of-the-art essay scoring systems. The dimensions of word embeddings and LSTM cells are 50 and 300, respectively. The convolution layer uses one filter with size 4. However, optimizing hyperparameters for each task may lead to better performance. For selecting two vectors with the highest similarity from the LSTM states of two adjacent sentences, we capture the similarity between any pair of LSTM states of the sentences as an element in a vector, and then apply a max-pooling layer to this vector of similarities to identify the pair with maximally similar LSTM states. Selected LSTM states are used for representing salient information shared by the sentences. In CohEmb, stop words are removed by the SMART English stop word list (Salton, 1971).

Training setup. We set the mini-batch size to 32 and train the network for 100 epochs. At each epoch we evaluate the model on the validation set and select the one with the best performance for test evaluations. We optimize with *Adam*, with an initial learning rate of 0.01. Word vectors are updated during training. The dropout method with rate 0.5 is employed for regularization. Loss functions are specifically defined for each task.

5 Experiments

We evaluate our model on two downstream tasks: readability assessment (Section 5.1), in which coherence representations of documents are mapped to coherence scores, and then documents are ranked based on these scores; and essay scoring (Section 5.2), in which the coherence representation of an essay is combined with other features for essay scoring to quantify the quality of the essay.

5.1 Readability Assessment

Readability assessment – How difficult is a text to read and understand? – depends on many factors one of which is coherence. Texts that are more coherent are supposed to be faster to read and easier to understand. Following earlier research on local coherence (Barzilay and Lapata, 2008; Pitler and Nenkova, 2008; Guinaudeau and Strube, 2013), we evaluate our coherence model on this task by ranking texts with respect to readability, instead of predicting readability scores. More formally, we approach readability assessment as follows: Given a text-pair, which text is easier to read?

---

[3] https://pytorch.org

Compared models. We compare the two variants of our model as described in Section 3.4 with two following state-of-the-art systems:

Mesgar and Strube (2016). This is a graph-based coherence model, in which nodes of a graph indicate sentences of a text, and an edge between two sentence nodes represents the existence of a lexico-semantic relation between two words in the sentences. Semantic relations between words are measured by the absolute value of the cosine function over their corresponding pre-trained word embeddings. If the similarity value for two word vectors is below a certain threshold[4] then the connection between these two words is omitted. Given the graph representation of a text, its coherence is encoded as a vector whose elements are frequencies of different subgraphs in the graph. The size of subgraphs is defined by the number of their nodes and is set to five. Subgraphs are extracted by a random sampling approach. We choose this model for comparison because its intuition is similar to our model. However, this model suffers from the following limitations: word embeddings are considered independently, not in their current context; and a manual threshold is used for connection filtering. We overcome these two weaknesses using the RNN and CNN layers in our model, respectively.

De Clercq and Hoste (2016). This is the state-of-the-art readability system on the examined dataset. It uses a rich set of readability features ranging from surface to semantic text features. The ranking is performed by LibSVM in their model. We report their best performance that is achieved by extensive feature engineering and SVM's parameter optimization.

Experimental setup. In Section 3, we formulated our model as $\vec{p} = L_\theta(d)$ where θ represents parameters of the neural modules (i.e. the CNN and RNN layers) in our model. For this task, we use an output layer to map coherence vector $\vec{p}$ to score s which quantifies the degree of the perceived coherence of document d. Formally, the output layer is $s_d = \vec{u} \cdot \vec{p} + b$ where $\vec{u}$ and b are the weight vector and bias, respectively. Let document d be more readable than document d', then the model should ideally produce $s_d > s'_d$. We train the parameters of the model by a pairwise ranking approach and define the loss function as:

$$loss = \max\{0, 1 - s_d + s_{d'}\}.$$

The parameters of the model are shared to obtain the scores for texts in pair (d, d').

Data. We use the readability dataset proposed by De Clercq et al. (2014). It consists of 105 texts collected from the British National Corpus and Wikipedia in four different genres: administrative (e.g. reports and surveys), informative (e.g. articles of newspapers and Wikipedia entries), instructive (e.g. user manuals and guidelines), and miscellaneous (e.g., very technical texts and children's literature). The average number of sentences is about 12 per text. $10,907$ pairs of texts are labeled with five fine-grained categories: $\{-100, -50, 0, 50, 100\}$ indicating that the first text in a pair is respectively *much easier*, *somewhat easier*, *equally difficult*, *somewhat difficult*, *more difficult* to read than the second text in the pair. Labels of text-pairs are assigned by human judges. Similar to De Clercq and Hoste (2016), we evaluate on the positive and negative labels as two sets of classes resulting in $6,290$ text-pairs in total. The original readability dataset does not provide any standard training/validation/test sets. We apply 5-fold cross-validation over this dataset.

Evaluation metric. The quality of a model is measured in terms of accuracy, which is the fraction of pairs that are correctly ranked by a model divided by the total number of document-pairs. We report the average accuracy over all runs of cross-validation as the final result. We perform a paired t-test to determine if improvements are statistically significant ($p < .05$).

Results. Table 1 summarizes the results of different systems for the readability assessment task.

Model	**Accuracy (%)**
Mesgar and Strube (2016)	85.70*
CohEmb	92.17*
CohLSTM	**97.77**
De Clercq and Hoste (2016)	96.88

Table 1: Results on readability assessment. The first system is the state-of-the-art coherence model on this dataset. The last one is a full readability system. "$*$" indicates statistically significant difference with the bold result.

CohEmb significantly outperforms the graph-based coherence model proposed by

---

[4] Like Mesgar and Strube (2016), we set this threshold to 0.9.

Mesgar and Strube (2016) by a large margin (6%), showing that our model captures coherence better than their model. In our model, the CNN layer automatically learns which connections are important to be considered for coherence patterns, whereas this is performed in Mesgar and Strube (2016) by defining a threshold for eliminating connections.

CohLSTM significantly outperforms both the coherence model proposed by Mesgar and Strube (2016) and the CohEmb model by 11% and 5%, respectively, and defines a new state-of-the-art on this dataset. CohLSTM, unlike Mesgar and Strube (2016)'s model and CohEmb, considers words of sentences in their sentence context. This supports our intuition that actual context information of words contributes to the perceived coherence of texts.

CohLSTM, which captures exclusively local coherence, even outperforms the readability system proposed by De Clercq and Hoste (2016), which relies on a wide range of lexical, syntactic and semantic features.

5.2 Essay Scoring

One part of the student assessment process is essay writing where students are asked to write an essay about a given topic known as a prompt. An essay scoring system assigns an essay a score reflecting the quality of the essay. The quality of an essay depends on various factors including coherence. Following previous studies (Miltsakaki and Kukich, 2004; Lei et al., 2014; Somasundaran et al., 2014; Zesch et al., 2015), we approach this task by combining the coherence vector produced by our model and the feature vector developed by an open-source essay scorer to represent an essay by a vector. The final vector representation of an essay, $\vec{x}$, is mapped to a score by a simple neural regression method as follows:

$$s = sigmoid(\vec{u} \cdot \vec{x} + b),$$

where $\vec{u}$ and b are the weight vector and the bias, respectively. We exactly define vector $\vec{x}$ for different examined systems, where we explain compared models for essay scoring.

Compared models. We compare variations of our model (Section 3.4) with the following models:

EASE (BLRR). As a baseline we use an open-source essay scoring system, Enhanced AI Scoring Engine[5] (EASE) (Phandi et al., 2015). This system was ranked third among all 154 participating teams in the Automated Student Assessment Prize (ASAP) competition and is the best among all open-source participating systems. It employs Bayesian Linear Ridge Regression (BLRR) as its regression method applied to a set of linguistic features grouped in four categories: *(i) Frequency-based features*: such as the number of characters, the number of words, the number of commas, etc; *(ii) POS-based features*: the number of POS n-grams; *(iii) Word overlap with prompt*; *(iv) Bag of n-grams*: the number of uni-grams and bi-grams.

EASE. The difference between this system and *EASE (BLRR)* is in the employed regression method. This system uses a neural regression method as described above. In order to have a similar experimental settings for this task, here, we use feature vectors generated by Phandi et al. (2015) to train our neural regression system. The input of the neural regression function is a nonlinear transformation of the feature vector produced by EASE, $\vec{f}$. Therefore $\vec{x} = tanh(\vec{w} \cdot \vec{f} + b)$.

EASE & CohEmb. This model combines the feature vector computed by EASE, $\vec{f}$, and the coherence vector produced by CohEmb, $\vec{p}$, to have a more reliable representation of an essay. More concretely, the input to our regression function, $\vec{x}$, is obtained as follows:

$$\vec{h}_1 = tanh(\vec{w}_1 \cdot \vec{f} + b_1),$$
$$\vec{h}_2 = \vec{h}_1 \oplus \vec{p},$$
$$\vec{x} = tanh(\vec{w}_2 \cdot \vec{h}_2 + b_2),$$

where $\oplus$ indicates the concatenation operatation.

EASE & CohLSTM. The structure of this model is the same as the *EASE & CohEmb* structure. But the input coherence vector, $\vec{p}$, is produced by CohLSTM.

Dong et al. (2017). It is a sentence-document model, which is especially designed for this task. It first encodes each sentence by a vector, which represents whole sentence meanings, and then use an RNN to embed vectors of sentences into a document vector.

Experimental setup. The size of the input vector for the regression method, $\vec{x}$, is fixed to 100 and its output size is fixed to 1. Dimensions of other parameters, $\vec{w}_1$ and $\vec{w}_2$, are set accordingly. The

---

[5] `https://github.com/edx/ease`

loss function is the Mean Squared Error (MSE) between human scores, $\vec{H}$, and scores predicted by our system, $\vec{S}$:

$$MSE(\vec{H}, \vec{S}) = \frac{1}{N} \sum (\vec{H} - \vec{S})^2.$$

The models are compared for each prompt by running 5-fold cross-validation (Dong et al., 2017).

Data. We apply our model to a dataset used in the Automated Student Assessment Prize (ASAP) competition run by Kaggle[6]. The essays are associated with scores given by humans and categorized in eight prompts. Each prompt can be interpreted as a different essay topic along with different genres. Table 2 summarizes some properties of this dataset.

Prompt	# Essays	Genre	Avg. Len.
1	1783	argumentative	350
2	1800	argumentative	350
3	1726	response	150
4	1772	response	150
5	1805	response	150
6	1800	response	150
7	1569	narrative	250
8	723	narrative	650

Table 2: Some properties of the dataset used for the essay scoring experiment.

Evaluation metric. ASAP adopted Quadratic Weighted Kappa (QWK) as the official evaluation metric. This metric measures the agreement between scores predicted by a system and scores assigned by humans. QWK considers chance agreements and penalizes large disagreements more than small agreements. We use an implementation of QWK that is described in Taghipour and Ng (2016). The formulation of QWK are explained in Appendix D. The final reported QWK is the average over QWKs of all prompts. We perform a paired t-test to determine if improvements are statistically significant ($p < .05$).

Results. Table 3 shows the results of different systems for the essay scoring task.

Both EASE & CohEmb, and EASE & CohLSTM significantly improve EASE, confirming that our proposed representation for coherence is beneficial for essay scoring and improves the performance of the examined essay scoring system. Our model does not beat the state-of-the-art essay scoring system (Dong et al., 2017), which is especially designed for this task and is tuned on this dataset. This model learns a vector representation for an input essay so that the vector performs the best for this regression task. In contrast, the core of our best performing essay scoring system, i.e. EASE & CohLSTM, is the feature vector generated by EASE, which has less modeling capacity than a deep learning model like the model proposed by Dong et al. (2017). The reason that we combine our coherence model with EASE, rather than the model proposed by Dong et al. (2017), is that EASE has no notion of coherence. By combining our coherence model with it, we examine if our coherence vector improves its performance or not.

Surprisingly, EASE & CohLSTM works on par with EASE & CohEmb. To gain a better insight, we ablate EASE feature vectors and compare the performance of the coherence models, i.e., CohLSTM, and CohEmb. Of course, coherence vectors on their own are not sufficient for predicting essay scores but this setup shows how much each variant of our model contributes to this task. The two last rows in Table 3 show the results.

CohLSTM outperforms CohEmb on all prompts, which matches the results for readability assessment. This confirms our intuition that integrating the information of the current context of words contributes to coherence measurement.

In terms of average QWK, CohLSTM works similar to EASE; however they behave differently on different prompts. The largest improvement for CohLSTM, with respect to EASE, is obtained on prompt 7 and 8. These two prompts ask for stories about *laughter* and *patience*, so corresponding essays can be categorized in the narrative genre (see Table 2). The guidelines of these two prompts, which are publicly available in the Kaggle data, ask human annotators to assign the highest score to essays that are coherent and hold the attention of readers through an essay. This is what our model captures: the sequence of semantic changes in a text, or coherence.

On prompt 5, in contrast, we see the largest deterioration in performance of CohLSTM in comparison to EASE. This prompt asks students to describe the mood created by the author of a memoir. Essays are expected to contain specific infor-

---

[6] `https://www.kaggle.com/c/asap-aes/data`

Model	Prompts								Avg QWK
	1	2	3	4	5	6	7	8	
EASE (BLRR)	0.761	0.606	0.621	0.742	0.784	0.775	0.730	0.617	0.705
Dong et al. (2017)	0.822	0.682	0.672	0.814	0.803	0.811	0.801	0.705	0.764
EASE	0.702	0.572	0.620	0.731	0.752	0.758	0.648	0.530	0.664
EASE & CohEmb	0.783	0.646	**0.664**	0.776	0.777	0.776	0.744	0.632	0.725*
EASE & CohLSTM	**0.784**	**0.654**	0.663	**0.788**	**0.793**	**0.794**	**0.756**	**0.646**	**0.728***
CohEmb	0.625	0.523	0.501	0.570	0.581	0.578	0.661	0.472	0.564
CohLSTM	0.669	0.634	0.591	0.710	0.639	0.716	0.729	0.641	0.666

Table 3: Results on essay scoring. "$*$" shows significant improvements with respect to the underlined score ($p < .05$). Bold numbers show the best results among different variants of our model.

mation from the memoir so that an essay with the highest score has the highest coverage of all relevant and specific information from the memoir. Therefore, mentioning the details of the memoir in essays of prompt 5 is more important than coherence for this prompt. This also shows that our model exclusively captures the coherence of a text, which is the goal of this paper.

6 Conclusions

We developed a local coherence model that encodes patterns of changes in what semantically relates adjacent sentences. The main novelty of our approach is that it defines sentence connections based on any semantic concept in sentences. In this sense, our model goes beyond entity-based coherence models, which need extra dependencies such as coreference resolution systems. Moreover, in contrast to lexical cohesion models, which take words individually, our model encodes words in their sentence context. Our model relates sentences by means of distant relations between word representations. The most similar LSTM states in two adjacent sentences are selected to encode the salient semantic concept that relates the sentences. The model finally employs a convolutional layer to extract and represent patterns of topic changes across sentences in a text as a coherence vector.

We evaluate coherence vectors generated by our model on the readability assessment and essay scoring tasks. On the former, our model achieves new state-of-the-art results. On the latter, it significantly improves the performance of a strong essay scorer. We believe the reason that our system works is that it learns which semantic concepts of sentences should be used to relate sentences, and which information about concepts is required to model sentence-to-sentence transitions. In future work we intend to run qualitative experiments on patterns that are extracted by our model to see if they are also linguistically interpretable.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1 and the Klaus Tschira Foundation, Heidelberg, Germany. We thank Mohammad Taher Pilehvar, Ines Rehbein, and Mark-Christoph Müller for their valuable feedback on earlier drafts of this paper. We also thank anonymous reviewers for their useful suggestions for improving the quality of the paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics,* Ann Arbor, Mich., 25–30 June 2005, pages 141–148.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Beata Beigman Klebanov and Michael Flor. 2013. Word association profiles and their use for automated scoring of essays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 1148–1158.

Beata Beigman Klebanov and Eli Shamir. 2006. Reader-based exploration of lexical cohesion. *Language Resources and Evaluation*, 40(2):109–126.

Jill Burstein, Joel Tetreault, and Slava Andreyev. 2010. Using entity-based features to model coherence in student essays. In *Proceedings of Human Language Technologies 2010: The Conference of the North American Chapter of the Association for Computational Linguistics,* Los Angeles, Cal., 2–4 June 2010, pages 681–684.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016, pages 484–494.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(8):2493–2537.

Orphée De Clercq and Véronique Hoste. 2016. All mixed up? Finding the optimal feature set for general readability prediction and its application to English and Dutch. *Computational Linguistics*, 42(3):457–490.

Orphée De Clercq, Véronique Hoste, Bart Desmet, Philip Van Oosten, Martine De Cock, and Lieve Macken. 2014. Using the crowd for readability prediction. *Natural Language Engineering*, 20(3):293–325.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning,* Vancouver, Canada, 3–4 August 2017, pages 153–162.

Micha Elsner and Eugene Charniak. 2011. Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Portland, Oreg., 19–24 June 2011, pages 125–129.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. 2010. A comparison of features for automatic readability assessment. In *Proceedings of Coling 2010: Poster Volume,* Beijing, China, 23–27 August 2010, pages 276–284.

Vanessa Wei Feng and Graeme Hirst. 2012. Extending the entity-based coherence model with multiple ranks. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics,* Avignon, France, 23–27 April 2012, pages 315–324.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 93–103.

Michael J. Heilman, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics,* Rochester, N.Y., 22–27 April 2007, pages 460–467.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Beijing, China, 26–31 July 2015, pages 1681–1691.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Baltimore, Md., 22–27 June 2014, pages 13–24.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Baltimore, Md., 22–27 June 2014, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 1746–1751.

J. Peter Kincaid, Robert P. Jr. Fishburne, Richard L. Rogers, and Brad S. Chisson. 1975. Derivation of new readability formulas (automated readability index, Fog count and Flesch reading ease formula) for navy enlisted personnel. Technical Report 8-75, Naval Technical Training Command, Naval Air Station Memphis-Millington, Tenn.

Alice Lai and Joel Tetreault. 2018. Discourse coherence in the wild: A dataset, evaluation and methods. In *Proceedings of the SIGdial 2018 Conference: The 19th Annual Meeting of the Special Interest Group on Discourse and Dialogue,* Melbourne, Australia, 12–14 July 2018, pages 214–223.

Chi-Un Lei, Ka Lok Man, and To Ting. 2014. Using learning analytics to analyze writing skills of students: A case study in a technological common core curriculum course. *IAENG International Journal of Computer Science*, 41(3):41–45.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 2039–2048.

Jiwei Li and Dan Jurafsky. 2017. Neural net models of open-domain discourse coherence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* Copenhagen, Denmark, 7–11 September 2017, pages 198–209.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Portland, Oreg., 19–24 June 2011, pages 997–1006.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 1157–1168.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Mass.

Mohsen Mesgar and Michael Strube. 2015. Graph-based coherence modeling for assessing readability. In *Proceedings of STARSEM 2015: The Fourth Joint Conference on Lexical and Computational Semantics,* Denver, Col., 4–5 June 2015, pages 309–318.

Mohsen Mesgar and Michael Strube. 2016. Lexical coherence graph modeling using word embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* San Diego, Cal., 12–17 June 2016, pages 1414–1423.

Eleni Miltsakaki and Karen Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering*, 10(1):25–55.

Casper Petersen, Christina Lioma, Jakob Grue Simonsen, and Birger Larsen. 2015. Entropy and graph based modelling of document coherence using discourse entities: An application to IR. In *Proceedings of the ACM SIGIR International Conference on the Theory of Information Retrieval,* Northhampton, Mass., 27-30 September 2015, pages 191–200.

Peter Phandi, Kian Ming A. Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* Lisbon, Portugal, 17–21 September 2015, pages 431–439.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Honolulu, Hawaii, 25–27 October 2008, pages 186–195.

Gerard Salton. 1971. *The SMART Retrieval System – Experiments in Automatic Document Processing*. Englewood Cliffs, N.J.: Prentice Hall.

Sarah E. Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics,* Ann Arbor, Mich., 25–30 June 2005, pages 523–530.

Swapna Somasundaran, Jill Burstein, and Martin Chodorow. 2014. Lexical chaining for measuring discourse coherence quality in test-taker essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers,* Dublin, Ireland, 23–29 August 2014, pages 950–961.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* Austin, Tex., 1–5 November 2016, pages 1882–1891.

Dat Tien Nguyen and Shafiq Joty. 2017. A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Vancouver, Canada, 30 July – 4 August, 2017, pages 1320–1330.

Amalia Todirascu, Thomas Francois, Delphine Bernhard, Nuria Gala, and Anne-Laure Ligozat. 2016. Are cohesive features relevant for text readability evaluation? In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,* Osaka, Japan, 11–17 December 2016, pages 987–997.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS 2017),* Long Beach, CA., USA, 4–9 December 2017, pages 6000–6010.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* Austin, Tex., 1–5 November 2016, pages 1504–1515.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell:

Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (Volume 37)* , Lille, France, 07–09 July 2015, pages 2048–2057.

Torsten Zesch, Michael Wojatzki, and Dirk Scholten-Akoun. 2015. Task-independent features for automated essay grading. In *Proceedings of the 10th Workshop on Innovative Use of NLP for Building Educational Applications,* Denver, Col., 4 June 2015, pages 224–232.

Muyu Zhang, Vanessa Wei Feng, Bing Qin, Graeme Hirst, Ting Liu, and Jingwen Huang. 2015. Encoding world knowledge in the evaluation of local coherence. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Denver, Col., 31 May – 5 June 2015, pages 1087–1096.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 1393–1398.

Deep Attentive Sentence Ordering Network

Baiyun Cui, Yingming Li,* Ming Chen, and Zhongfei Zhang
College of Information Science and Electronic Engineering,
Zhejiang University, China
`baiyunc@yahoo.com,{yingming,funkyblack,zhongfei}@zju.edu.cn`

Abstract

In this paper, we propose a novel deep attentive sentence ordering network (referred as ATTOrderNet) which integrates self-attention mechanism with LSTMs in the encoding of input sentences. It enables us to capture global dependencies among sentences regardless of their input order and obtains a reliable representation of the sentence set. With this representation, a pointer network is exploited to generate an ordered sequence. The proposed model is evaluated on Sentence Ordering and Order Discrimination tasks. The extensive experimental results demonstrate its effectiveness and superiority to the state-of-the-art methods.

1 Introduction

Modeling a coherent text is one of the key problems in natural language processing. A well-organized text with a logical structure is much easier for people to read and understand. Sentence ordering task (Barzilay and Lapata, 2008) has been proposed to cope with this problem. It aims to organize a set of sentences into a coherent text with a logically consistent order and has wide applications in natural language generation such as concept-to-text generation (Konstas and Lapata, 2012a,b, 2013), retrieval-based question answering (Yu et al., 2018; Verberne, 2011), and extractive multi-document summarization (Barzilay and Elhadad, 2002; Galanis et al., 2012; Nallapati et al., 2017), where the improper ordering of sentences would introduce ambiguity and degrade readability. An example of this task is shown in Table 1.

A set of sentences		An ordered text	
4	Finally, the parser is evaluated.	1	We develop a useful parser.
1	We develop a useful parser.	2	We first describe the older one.
3	Then we present our parser.	3	Then we present our parser.
2	We first describe the older one.	4	Finally, the parser is evaluated.

Table 1: An example to illustrate the sentence ordering task. The set of sentences is confusing in its current order. We aim to reorganize them with a more coherent order.

Traditional methods developed for this task employ handcrafted linguistic features to model the document structure such as Entity Grid (Barzilay and Lapata, 2008), Content Model (Barzilay and Lee, 2004), and Probabilistic Model (Lapata, 2003). However, manual feature engineering heavily relies on linguistic knowledge and also limits these systems to be domain specific. Inspired by the success of deep learning, data-driven approaches based on neural networks have been proposed including Pairwise Ranking Model (Chen et al., 2016) which learns the relative order of sentence pairs to predict the pairwise ordering of sentences, and Window network (Li and Hovy, 2014) sliding a window over the text to evaluate the coherence.

Recently, hierarchical RNN-based approaches (Gong et al., 2016; Logeswaran et al., 2018) have been proposed to deal with this task. Such methods exploit LSTMs based paragraph encoder to compute a context representation for the whole sequential sentences and then adopt a pointer network (Vinyals et al., 2015) as the decoder to predict their order. However, since LSTM works sequentially, paragraph encoder only based on LSTMs suffers from the incorrect input sentence order and has difficulty in capturing a logically reliable representation through the recurrent connections, which makes trouble for the decoder to find the correct order.

To overcome the above limitation, in this work, we develop a novel deep attentive sentence ordering network (referred as ATTOrderNet) by inte-

---

*Corresponding author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4340–4349
Brussels, Belgium, October 31 - November 4, 2018.

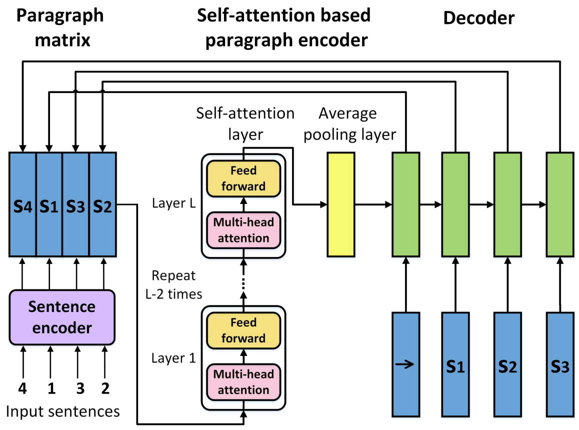

Figure 1: Architecture of deep attentive Sentence Ordering Network.

grating self-attention mechanism (Vaswani et al., 2017) with LSTMs to learn a relatively reliable paragraph representation for subsequent sentence ordering. In particular, the bidirectional LSTM is first adopted as a sentence encoder to map the input sentences to the corresponding distributed vectors, and then a self-attention based paragraph encoder is introduced to capture structural relationships across sentences and to obtain a hierarchical context representation of the entire set of sentences. Consequently, based on the learned paragraph vector, a pointer network is applied to perform sentence ordering by decoding an ordered sequence. Figure 1 shows the architecture of ATTOrderNet, where self-attention mechanism is introduced to capture the dependencies among sentences.

In contrast to the previous paragraph encoders with LSTMs, self-attention mechanism is less sensitive to the input order of sentence sequence and is effective in modeling the accurate relationships across sentences, which reduces the influence of the original order of the input sentences and perfectly meets the requirement of our task. Further, unlike Transformer (Vaswani et al., 2017), we do not add any positional encodings in our model to minimize the influence of the unclear order information.

Extensive evaluations are conducted on the sentence ordering task and order discrimination task to investigate the performances of ATTOrderNet. The experimental results on seven public sentence ordering datasets show the superior performances of the framework to the competing models. Meanwhile, the visualization of the attention layer in paragraph encoder is provided for better understanding of the effectivity of self-attention mechanism. Besides, in the Order Discrimination task, our model also achieves the state-of-the-art performance with remarkable improvements on two benchmark datasets.

2 Deep Attentive Sentence Ordering Network

In this section, we first formulate the sentence ordering problem and then describe the proposed model ATTOrderNet, which is based on the encoder-decoder architecture applying self-attention mechanism as paragraph encoder and a pointer network as the decoder. This combination effectively captures the intrinsic relations across a set of sentences with the desirable property of being invariant to the sentence order, which directly helps address the difficulty of this task.

2.1 Problem formulation

The sentence ordering task aims to order a set of sentences as a coherent text. Specifically, a set of n sentences with the order $\mathbf{o} = [o_1, o_2, \cdots, o_n]$ can be described as $\mathbf{s} = [s_{o_1}, s_{o_2}, \cdots, s_{o_n}]$. The goal is to find the correct order $\mathbf{o}^*$ for them, $\mathbf{o}^* = [o_1^*, o_2^*, \cdots, o_n^*]$, with which the whole sentences have the highest coherence probability:

$$P(\mathbf{o}^*|\mathbf{s}) > P(\mathbf{o}|\mathbf{s}), \forall \mathbf{o} \in \psi \qquad (1)$$

where $\mathbf{o}$ indicates any order of these sentences and ψ denotes the set of all possible orders. For instance, in Table 1, the current order $\mathbf{o}$ is $[4, 1, 3, 2]$ and $\mathbf{o}^* = [1, 2, 3, 4]$ is the correct order for these sentences.

2.2 Intuition and Model architecture

Given a set of sentences, the existing hierarchical RNN-based models first transform each sentence into a distributed vector with a sentence encoder and then these sentence embeddings are fed to a LSTMs-based paragraph encoder. Consequently, based on the learned paragraph vector, a pointer network is exploited to decode the order of the input sentences. However, since LSTM works sequentially while the order of these sentences is unknown and quite possibly wrong in this problem, LSTMs-based paragraph encoder has difficulty in capturing a convincing representation through the recurrent connections, which influences the performance of sentence ordering.

In this paper, we use self-attention mechanism for paragraph encoder instead. In particular, we employ this mechanism without encoding any positional information of the sentences. Ignoring the current order of the sentences, self-attention based paragraph encoder perfectly meets the requirement of the sentence ordering task and learns a logically reliable representation of the whole paragraph by globally capturing the relationships across sentences. With this learned representation vector, a decoder is then designed to generate a coherent order assignment for the input sentences.

In the following, we elaborate on the main building blocks of our ATTOrderNet in details: a sentence encoder, a self-attention mechanism based paragraph encoder, and a decoder.

2.3 Sentence Encoder

For a sentence, we first apply word embedding matrix to translate the raw words in the sentence into distributional representations, and then adopt bidirectional LSTMs to learn a sentence-level representation for summarizing its high level semantic concepts.

Specifically, assume that a sentence s_{o_i} containing n_w raw words as $s_{o_i} = [w_1, \cdots, w_{n_w}]$, these words are transformed to dense vectors through a word embedding matrix $\mathbf{W}_e$: $\mathbf{x}_t = \mathbf{W}_e w_t$, $t \in [1, n_w]$. The sequence of vectors $[\mathbf{x}_1, \cdots, \mathbf{x}_{n_w}]$ is then fed into bidirectional LSTMs sequentially to compute a semantic representation of the sentence.

Long Short-term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) is capable of learning long-term dependencies and alleviating the problems of gradient vanishment and exploding. Here, we adopt bidirectional LSTMs (Bi-LSTMs) to take full advantages of additional backward information and enhance the memory capability. In particular, the Bi-LSTMs contain the forward LSTMs which process the sentence s_{o_i} from w_1 to w_{n_w} and backward LSTMs which read s_{o_i} in the reversed direction:

$$\begin{aligned} \overrightarrow{\mathbf{h}}_t, \overrightarrow{\mathbf{c}}_t &= \mathrm{LSTM}(\overrightarrow{\mathbf{h}}_{t-1}, \overrightarrow{\mathbf{c}}_{t-1}, \mathbf{x}_t) \\ \overleftarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{c}}_t &= \mathrm{LSTM}(\overleftarrow{\mathbf{h}}_{t+1}, \overleftarrow{\mathbf{c}}_{t+1}, \mathbf{x}_t) \\ \mathbf{h}_t &= [\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t] \end{aligned} \tag{2}$$

where $\mathbf{h}_t$ denotes the representation of position t by concatenating the forward hidden state $\overrightarrow{\mathbf{h}}_t$ and backward hidden state $\overleftarrow{\mathbf{h}}_t$ together. The output of the last hidden state of the Bi-LSTMs is taken to be the sentence representation vector as $\mathbf{s}_{o_i} = \mathbf{h}_{n_w}$, which incorporates the contextual information from both directions in the sentence.

So far, we have obtained a syntactic and semantic representation for a single sentence. In the following, a self-attention based paragraph encoder is proposed to obtain a high level representation for all given sentences by capturing sequential structures and logical relationships among them.

2.4 Paragraph Encoder

2.4.1 Self-attention mechanism

We start by introducing the scaled dot-product attention, which is the foundation of self-attention mechanism used in ATTOrderNet. Given a matrix of n query vectors $\mathbf{Q} \in \mathbb{R}^{n \times d}$, keys $\mathbf{K} \in \mathbb{R}^{n \times d}$, and values $\mathbf{V} \in \mathbb{R}^{n \times d}$, the scaled dot-product attention computes the output matrix as:

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{\mathrm{d}}})\mathbf{V} \tag{3}$$

The multi-head attention with h parallel heads is employed, where each head is an independent scaled dot-product attention. The mathematical formulation is shown below:

$$\mathbf{M}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_{\mathrm{i}}^{\mathrm{Q}}, \mathbf{K}\mathbf{W}_{\mathrm{i}}^{\mathrm{K}}, \mathbf{V}\mathbf{W}_{\mathrm{i}}^{\mathrm{V}}) \tag{4}$$

$$\mathrm{MH}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathbf{M}_1, \cdots, \mathbf{M}_h)\mathbf{W} \tag{5}$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d \times d_a}$ with $d_a = d/h$ are the projection matrices for the i-th head and $\mathbf{W} \in \mathbb{R}^{hd_a \times d}$.

Self-attention (Vaswani et al., 2017; Tan et al., 2017; Shen et al., 2017) is a special case of attention mechanism that only requires a single sequence to compute its representation where queries, keys, and values are all from the same place.

2.4.2 Self-attention based Paragraph Encoder

The paragraph encoder is composed of multiple self-attention layers followed by an average pooling layer.

Sentence vectors encoded by the sentence encoder are first packed together into a paragraph matrix $\mathbf{S} = [\mathbf{s}_{o_1}, \mathbf{s}_{o_2}, \cdots, \mathbf{s}_{o_n}]$ as $\mathbf{E}^0$. This paragraph matrix $\mathbf{S} \in \mathbb{R}^{n \times d}$ is then fed forward to L self-attention layers, where each layer learns a representation $\mathbf{E}^{l+1} = \mathrm{U}(\mathbf{E}^l)$ by taking the output from the previous layer l:

$$\mathrm{U}(\mathbf{E}^l) = \Phi(\mathrm{FN}(\mathrm{D}(\mathbf{E}^l)), \mathrm{D}(\mathbf{E}^l)) \quad (6)$$

$$\mathrm{D}(\mathbf{E}^l) = \Phi(\mathrm{MH}(\mathbf{E}^l, \mathbf{E}^l, \mathbf{E}^l), \mathbf{E}^l) \quad (7)$$

$$\Phi(\mathbf{v}, \mathbf{w}) = \mathrm{LayerNorm}(\mathbf{v} + \mathbf{w}) \quad (8)$$

$$\mathrm{FN}(\mathbf{x}) = \mathrm{ReLU}(\mathbf{x}\mathbf{W}_1^l + \mathbf{b}_1^l)\mathbf{W}_2^l + \mathbf{b}_2^l \quad (9)$$

where $\Phi(\cdot)$ performs layer normalization (Ba et al., 2016) on the residual output to preserve the auto-regressive property, and $\mathrm{FN}(\cdot)$ represents the fully connected feed-forward networks which consists of two linear layers with ReLU nonlinearity in the middle. $\mathbf{W}_1^l \in \mathbb{R}^{d \times d_f}$, $\mathbf{b}_1^l \in \mathbb{R}^{d_f}$, $\mathbf{W}_2^l \in \mathbb{R}^{d_f \times d}$, and $\mathbf{b}_2^l \in \mathbb{R}^d$ are trainable parameters. We set $d_f = 1024$ in all our experiments.

Self-attention mechanism adopted in the paragraph encoder directly relate sentences at different positions from the text by computing the attention score (relevance) between each pair of sentences. This allows each sentence to build links with all other sentences in the text, which enables the encoder to exploit latent dependency relationships among sentences without regarding to their input order. Then, attention mechanism uses weighted sum operation to establish a higher level representation for the entire sentence set. As we see, there is no order information used in the encoding process which prevents the model from being affected by the incorrect sentence order. Therefore, self-attention based paragraph encoder is efficient in modeling of dependencies while being invariant to the sentence order.

The final paragraph representation $\mathbf{v}$ is obtained in the average pooling layer by averaging the output matrix $\mathbf{E}^L \in \mathbb{R}^{n \times d}$ from the last self-attention layer: $\mathbf{v} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{e}^{L_i}$, where n is the number of sentences and $\mathbf{e}^{L_i}$ denotes the i-th row in $\mathbf{E}^L$. This learned representation vector can be viewed as a hierarchical encoding of the entire set of sentences which will then be used as the input of the decoder to perform sentence ordering.

2.5 Decoder

The aim of decoder is to predict a consistent order for the input set of sentences.

Following the previous approaches (Gong et al., 2016; Logeswaran et al., 2018), the coherence probability of given sentences $\mathbf{s}$ with the order $\mathbf{o}$ is formalized as:

$$P(\mathbf{o}|\mathbf{s}) = \prod_{i=1}^{n} P(o_i|o_{i-1}, \cdots, o_1, \mathbf{s}) \quad (10)$$

The higher the probability, the more coherent sentences assignment is.

To calculate $P(\mathbf{o}|\mathbf{s})$, we employ the pointer network architecture (Vinyals et al., 2015) as our decoder which consists of LSTMs cells (Equation 11-13). The LSTM takes the embedding of the previous sentence as the input to decoder step. During training, the correct order of sentences $\mathbf{o}^*$ is known, so the input sequence $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] = [\mathbf{s}_{o_1^*}, \mathbf{s}_{o_2^*}, \cdots, \mathbf{s}_{o_n^*}]$. For step i, the input to the decoder is $\mathbf{x}_{i-1} = \mathbf{s}_{o_{i-1}^*}$. At test time, the predicted sentence assignment $\mathbf{s}_{\hat{o}_{i-1}}$ is used instead.

The initial state of the decoder LSTM is initialized with the final paragraph vector from the encoder: $\mathbf{h}_0 = \mathbf{v}^T$. And the input at the first step in decoder $\mathbf{x}_0 \in \mathbb{R}^d$ is a vector of zeros. The mathematical formulation for the i-th step in decoder is as follows:

$$\mathbf{h}_i, \mathbf{c}_i = \mathrm{LSTM}(\mathbf{h}_{i-1}, \mathbf{c}_{i-1}, \mathbf{x}_{i-1}) \quad (11)$$

$$\mathbf{u}_j^i = \mathbf{g}^T \tanh(\mathbf{W}_1 \mathbf{s}_{o_j} + \mathbf{W}_2 \mathbf{h}_i) \quad (12)$$

$$P(o_i|o_{i-1}, \cdots, o_1, \mathbf{s}) = \mathrm{softmax}(\mathbf{u}^i) \quad (13)$$

where $\mathbf{g} \in \mathbb{R}^d$, $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$, and $\mathbf{W}_2 \in \mathbb{R}^{d \times d}$ are learnable parameters and $j \in (1, \cdots, n)$. The softmax function normalizes the vector $\mathbf{u}^i \in \mathbb{R}^n$ to produce an output distribution over all input sentences. And $P(o_i|o_{i-1}, \cdots, o_1, \mathbf{s})$ can be interpreted as the coherence probability for the current output sequence when s_{o_i} being the sentence choice at position i conditioned on the previous sentences assignment.

Order Prediction: The predicted order $\hat{\mathbf{o}} = [\hat{o}_1, \hat{o}_2, \cdots, \hat{o}_n]$ is the one with the highest coherence probability:

$$\hat{\mathbf{o}} = \underset{\mathbf{o}}{\mathrm{argmax}}\, P(\mathbf{o}|\mathbf{s}) \quad (14)$$

In this work, we use beam search strategy to find a sub optimal result.

2.6 Training

For each ordered document, we use one random permutation of sentences as the input sample at each epoch during the training and testing process. Assume that there are K documents in the training set. We define $(q_j, y_j)_{j=1}^K$, where y_j is in the correct order $\mathbf{o}^*$ of original document j and q_j denotes the set of sentences with a specific permutation of y_j. $P(y_j|q_j) = P(\mathbf{o}^*|\mathbf{s} = q_j)$ can be interpreted as the probability that sentences are assigned in the correct order when given sentences q_j.

Dataset	Length statistics			Data split			Vocabulary
	min	mean	max	train	valid	test	
Accident	6	11.5	19	100	-	100	4501
Earthquake	3	10.4	32	100	-	99	3022
NIPS abstract	2	6	15	2248	409	402	16721
AAN abstract	1	5	20	8569	962	2626	34485
NSF abstract	2	8.9	40	96070	10185	21580	334090
arXiv abstract	2	5.38	35	884912	110614	110615	64557
SIND caption	5	5	5	40155	4990	5055	30861

Table 2: Statistics of seven datasets used in our experiments.

We aim to train the overall model to maximize this probability by minimizing the loss function:

$$L = -\frac{1}{K}\sum_{j=1}^{K} \log P(y_j|q_j;\theta) + \frac{\lambda}{2}||\theta||_2^2 \qquad (15)$$

where θ represents all trainable parameters in the networks and λ is a regularization parameter.

3 Experiments

3.1 Datasets

Accident, Earthquake: two datasets obtained from (Barzilay and Lapata, 2008). The first one is a collection of aviation accident reports from the National Transportation Safety Board and the second one comprises Associated Press articles related to earthquake. Since the original datasets do not provide validation samples, we follow the setup in (Louis and Nenkova, 2012; Li and Hovy, 2014) and use 10-fold cross validation on the training data.

NIPS abstract, AAN abstract, NSF abstract: these three datasets are from (Logeswaran et al., 2018) containing abstracts from NIPS papers, ACL papers, and the NSF Research Award Abstracts dataset respectively.

arXiv abstract, SIND caption: we further consider two datasets used in (Gong et al., 2016). The former consists of abstracts from papers on arXiv website (Chen et al., 2016) and the other contains captions from SIND dataset (Huang et al., 2016).

Further statistics about seven datasets are illustrated in Table 2.

3.2 Training setup

We use pre-trained 100 dimensional GloVe word embeddings (Pennington et al., 2014). And all the out-of-vocabulary words are replaced with <UNK>, whose embeddings are updated during training process. The nltk sentence tokenizer is used for word tokenization.[1] Parameter optimization is performed using stochastic gradient descent. We adopt Adadelta (Zeiler, 2012) as the optimizer with $\epsilon = 10^6$ and $\rho = 0.95$. The learning rate is initialized to 1.0, the batch size is 16, and the beam size is set to 64. The hidden layer size of LSTMs in sentence encoder is 256, and is 512 in the decoder. The number of attention layers in the paragraph encoder is 6 for AAN abstract, 4 for NSF abstract and arXiv abstract, and 2 for the rest of datasets. We employ 8 parallel heads throughout all self-attention layers and use L2 weight decay on the trainable variables with regularization parameter $\lambda = 10^{-5}$. The model is implemented with TensorFlow[2]. Hyperparameters are chosen using the validation set.

3.3 Sentence Ordering

We first evaluate our model on the sentence ordering task, as proposed by Barzilay and Lapata (2008). Given a set of permuted sentences, our goal is to return the original order for them which is considered to be the most coherent.

3.3.1 Baselines

We compare ATTOrderNet against a random baseline and all the competing models. These baseline methods can be categorized into three classes and results are reported in (Soricut and Marcu, 2006; Gong et al., 2016; Logeswaran et al., 2018).

(1) Traditional approaches: Probabilistic Model (Lapata, 2003); Content Model (Barzilay and Lee, 2004); Utility-Trained model (Soricut and Marcu, 2006); Entity Grid (Barzilay and Lapata, 2008). These four methods employ handcrafted features in modeling the document structure.

(2) Data-driven methods: Window network (Li and Hovy, 2014); Seq2seq (Li and Jurafsky,

---

[1] NLTK implementation: http://www.nltk.org/

[2] https://www.tensorflow.org/

Models	Accident	Earthquake	NIPS abstract		AAN abstract		NSF abstract		arXiv abstract		SIND caption	
	τ	τ	Acc	τ	Acc	τ	Acc	τ	PMR	τ	PMR	τ
Random	0	0	15.59	0	19.36	0	9.46	0	8.07	0	6.05	0
Probabilistic Model	0.07	0.48	-	-	-	-	-	-	-	-	-	-
Content Model	0.44	0.81	-	-	-	-	-	-	-	-	-	-
Utility-Train Model	0.50	0.47	-	-	-	-	-	-	-	-	-	-
Entity Grid	0.12	0.19	20.10	0.09	21.82	0.10	-	-	-	-	-	-
Seq2seq	-	-	27.18	0.27	36.62	0.40	13.68	0.10	-	-	-	-
Window network	-	-	41.76	0.59	50.87	0.65	18.67	0.28	-	-	-	-
Pairwise Ranking Model	-	-	-	-	-	-	-	-	33.43	0.66	-	-
RNN Decoder	-	-	48.22	0.67	52.06	0.66	25.79	0.48	-	-	-	-
Varient-LSTM+PtrNet	-	-	51.55	**0.72**	58.06	**0.73**	28.33	0.51	-	-	-	-
CNN+PtrNet	0.58	0.84	48.64	0.66	58.21	0.69	33.22	0.52	39.28	0.71	12.32	0.48
LSTM+PtrNet	0.58	0.85	50.87	0.67	58.20	0.69	32.45	0.52	40.44	0.72	12.34	0.48
ATTOrderNet (ATT)	0.59	0.87	52.17	0.71	60.95	0.70	37.07	0.52	41.03	0.72	12.77	0.48
ATTOrderNet (CNN)	0.61	0.89	54.69	0.72	62.97	0.72	37.59	0.54	42.09	0.73	13.95	0.49
ATTOrderNet	**0.64**	**0.92**	**56.09**	**0.72**	**63.24**	**0.73**	**37.72**	**0.55**	**42.19**	**0.73**	**14.01**	**0.49**

Table 3: Experimental results for different methods on the Sentence Ordering task.

2017); Pairwise Ranking Model (Chen et al., 2016). These three approaches capture the local coherence of text based on the neural networks.
(3) Hierarchical RNN-based models: Varient-LSTM+PtrNet, RNN Decoder (Logeswaran et al., 2018); CNN+PtrNet, LSTM+PtrNet (Gong et al., 2016). These architectures adopt RNN based approaches to obtain the representation for the input set of sentences and employ the pointer network as the decoder to predict order. The main difference between ATTOrderNet and these models lies in the design of paragraph encoder.

For thorough comparison, besides the models proposed in the existing literature, we further implement two variants of ATTOrderNet.

ATTOrderNet (ATT): The sentence encoder in this model is also entirely based on self-attention mechanism with 4 self-attention layers and 5 heads. Different from the paragraph encoder, the positional encoding method proposed by Vaswani et al. (2017) is applied here to encode temporal information of each input word.

ATTOrderNet (CNN): This model employs convolutional neural networks to model sentences. In experiment, the number of feature maps is set to 512 and the width of convolution filter is 4.

3.3.2 Evaluation Metrics

To provide assessments on the quality of the orderings we predict in this task, we use the following three metrics:

Kendall's tau (τ): Kendall's tau is one of the most frequently used metrics for the automatic evaluation of document coherence (Lapata, 2003; Logeswaran et al., 2018; Li and Jurafsky, 2017). It could be formalized as: $\tau = 1 - 2\times$ (number of inversions) $/\binom{n}{2}$, where n is the length of the sequence and the number of inversions denotes the number of pairs in the predicted sequence with incorrect relative order. This metric ranges from -1 (the worst) to 1 (the best).

Accuracy (Acc): We follow (Logeswaran et al., 2018) in employing Accuracy to measure how often the absolute position of a sentence was correctly predicted. Compared to τ, it penalizes correctly predicted subsequences that are shifted.

Perfect Match Ratio (PMR): Perfect match ratio (Gong et al., 2016) is the most stringent measurement in this task. It calculates the radio of exactly matching orders: PMR=$\frac{1}{K}\sum_{i=1}^{K}\mathbf{1}(\widehat{\mathbf{o}}^i = \mathbf{o}^{i*})$, where $\widehat{\mathbf{o}}^i$ and $\mathbf{o}^{i*}$ are predicted and correct orders of the i-th text respectively.

3.3.3 Results

The experimental results on all datasets are reported in Table 3. Results show that ATTOrderNet gives the best performance across most datasets and under most evaluation measurements.

The improvement is regardless of data sizes. In particular, for smaller datasets such as Accident and Earthquake datasets, ATTOrderNet outperforms the previous best baseline methods by 6% and 7% tau score respectively. As for medium size datasets including NIPS abstract and AAN abstract, ATTOrderNet shows absolute improvements of 4.54% and 5.03% accuracy score over the previous state-of-the-art. Such finding is consistent across larger datasets. ATTOrderNet outperforms the previous state-of-the-art systems by 4.50% accuracy score with 3% tau score on NSF abstract, 1.75% PMR score with 1% tau score on arXiv abstract, and 1.67% PMR score with 1% tau score on SIND caption. Interestingly, ATTOrderNet reaches 42.19% PMR score on arXiv abstract, which means that more than 2/5 texts in the test set can be ordered exactly right. This performance clearly demonstrates the adaptability and flexibility of the proposed model.

As shown in Table 3, ATTOrderNet performs much better than data-driven methods by a significant margin on all corresponding datasets. It proves the importance of exploiting the context by self-attention mechanism as these competing models only consider the local coherence in the text. Among the traditional ordering approaches, Content Model (Barzilay and Lee, 2004) representing topics as states and capturing possible orderings for global coherence performs better than other methods with the tau score of 0.81 on Earthquake dataset, which also demonstrates that global context is important to sentence ordering. However, Content Model requires manual feature engineering that costs great human efforts. In contrast, the self-attention mechanism used in ATTOrderNet directly captures the global dependences for the whole text while requiring no linguistic knowledge anymore and enables ATTOrderNet to further improve tau score to 0.92 on the same dataset.

In addition, hierarchical RNN-based models capture the global coherence among sentences with LSTMs and outperform the traditional methods and data-driven approaches in most cases. However, these models still suffer from the permutation of sentences within the document since LSTM works sequentially. ATTOrderNet achieves superior performances to them by adopting the self-attention mechanism to reduce the influence of the permutation of sentences.

Further, ATTOrderNet (CNN) has better performances than ATTOrderNet (ATT) on most of the datasets. We conjecture that this is due to the limitation of data size. Since ATTOrderNet (ATT) applies self-attention mechanism in both sentence and paragraph encoders requiring more data to train the model, however the size of the datasets used in this task is smaller than those in other tasks such as document classification (Yang et al., 2016). Given larger datasets in the future, we believe ATTOrderNet (ATT) would perform much better. Among three sentence encoders, ATTOrderNet presents a superior performance across the board. This indicates that LSTM is more efficient in learning semantic representation for sentence level in this task. ATTOrderNet becomes more competitive through combining both advantages of LSTMs and self-attention mechanism.

Models	arXiv abstract		SIND caption	
	head	tail	head	tail
Random	23.06	23.16	22.78	22.56
Pairwise Ranking Model	84.85	62.37	-	-
CNN+PtrNet	89.43	65.36	73.53	53.26
LSTM+PtrNet	90.47	66.49	74.66	53.30
ATTOrderNet (ATT)	89.68	65.75	75.88	54.30
ATTOrderNet (CNN)	90.86	67.85	75.95	54.37
ATTOrderNet	**91.00**	**68.08**	**76.00**	**54.42**

Table 4: The performance of correctly predicting the first and the last sentences on arXiv abstract and SIND caption datasets.

Since the first and the last sentences of the text are more special to discern (Chen et al., 2016; Gong et al., 2016), we also evaluate the ratio of correctly predicting the first and the last sentences. Table 4 summarizes our performances on arXiv abstract and SIND caption. As we see, all models show fair well in predicting the first sentence, and the prediction accuracy declines for the last one. It is observed that ATTOrderNet still achieves a boost in predicting two positions compared to the previous state-of-the-art system on both datasets.

3.3.4 Visualization of attention

The aim of this section is to visualize the relationship between sentences captured by self-attention mechanism and understand how it helps perform the sentence ordering task. A technique for visualizing attention mechanism in neural networks is proposed by Vaswani et al. (2017) [3]. Inspired by this work, we select a text from AAN abstract dataset to visualize the hierarchical attention layer from the paragraph encoder of ATTOrderNet in Figure 2. Different from visualizing the dependencies of one word with the other words in the sentence (Vaswani et al., 2017), our visualization

---

[3] https://github.com/tensorflow/tensor2tensor

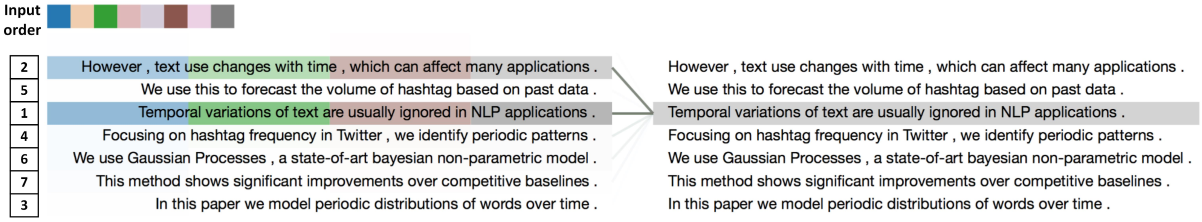

Figure 2: An example of the attention mechanism describing sentence dependencies in the encoder self-attention in layer 5 of 6 for the text from AAN abstract. Four attention heads attend to a dependency of the selected sentence *"Temporal variations of text are ..."*, providing important clues for the correct order prediction. Attention here shown is only for this sentence. Different color indicate different heads. Best viewed in color.

shows the dependencies between each sentence and all other sentences in the text.

For the example in Figure 2, the left text is the input sample to our model which contains a set of permuted sentences with the correct order besides it. The right side is a copy of the input text, which is presented for showing the relevance between each pair of sentences more clearly. The line with grey color on the right text is an example sentence chosen to visualize the attention weights with other sentences. On the top of the text are 8 colored squares representing 8 different attention heads used in the paragraph encoder. Colored columns on the left text show the performance of their corresponding heads. The darkness of the color in column denotes the normalized distribution of the attention weight for the example sentence in the head. Sentences in darker shades show more attention weight which reflects stronger links they have with the example sentence.

In particular, Figure 2 shows the attention distribution for the first sentence in the original document. We present the weight distribution in four heads as an instance. It is interesting to see that all of them showing significant higher attention weights on the true second sentence *"However, text use changes ..."* than other sentences in the text. This indicates that these heads are able to learn the latent dependency relationships from sentences and can successfully distinguish which one is the true next following among all sentence candidates. These heads build much stronger links between this sentence with the chosen one in order to keep structural information for higher level representation, such as paragraph representation.

3.4 Order Discrimination

In this section, we assess ATTOrderNet on another common evaluation task which is usually adopted in the existing literature: order discrimination task.

Order discrimination (Barzilay and Lapata, 2008; Elsner and Charniak, 2011, 2008) aims to compare a document to a randomly permuted version of it. Models are evaluated with Pairwise Accuracy: the ratio of correctly identifying the original document with higher coherence probability (defined in Equation 10) than the probability of its permutation.

Among seven datasets mentioned above, we use two of them to assess the performance of ATTOrderNet on the order discrimination task: Accident and Earthquake datasets. These two have been widely used for this task in the previous literature (Li and Hovy, 2014; Logeswaran et al., 2018). This gives us the convenience of directly comparing the result of the proposed model against the reported results. Following the setup in (Barzilay and Lapata, 2008), a maximum of 20 random permutations were generated for each training and testing article to create the pairwise data. There are 1986 and 1956 test pairs in Accident and Earthquake datasets respectively.

3.4.1 Baselines

To demonstrate that ATTOrderNet truly improves the order discrimination performance, we compare ATTOrderNet with the following representative models: Graph from (Guinaudeau and Strube, 2013), HMM and HMM+Entity from (Louis and Nenkova, 2012), Entity Grid from (Barzilay and Lapata, 2008), Recurrent and Recursive from (Li and Hovy, 2014), Discriminative model from (Li and Jurafsky, 2017), Varient-LSTM+PtrNet from (Logeswaran et al., 2018), CNN+PtrNet and LSTM+PtrNet from (Gong et al., 2016). The results of the last two methods were obtained by training their models on two datasets.

Models	Accident	Earthquake
Random	50.0	50.0
Graph	84.6	63.5
HMM+Entity	84.2	91.1
HMM	82.2	93.8
Entity Grid	90.4	87.2
Recurrent	84.0	95.1
Recursive	86.4	97.6
Discriminative Model	93.0	99.2
Varient-LSTM+PtrNet	94.4	99.7
CNN+PtrNet	93.5	99.4
LSTM+PtrNet	93.7	99.5
ATTOrderNet (ATT)	95.4	99.6
ATTOrderNet (CNN)	95.8	99.7
ATTOrderNet	**96.2**	**99.8**

Table 5: Experimental results of Pairwise Accuracy for different approaches on two datasets in the Order Discrimination task.

3.4.2 Results

Table 5 reports the results of ATTOrderNet and currently competing architectures in this evaluation task. ATTOrderNet also achieves the state-of-the-art performance, showing a remarkable advancement of about 1.8% gain on Accident dataset and further improving the pairwise accuracy to 99.8 on Earthquake dataset.

LSTM+PtrNet and CNN+ PtrNet (Gong et al., 2016) fall short of Varient-LSTM+PtrNet (Logeswaran et al., 2018) in performance. This could also be blamed for their paragraph encoder. Documents in both datasets are much longer than those in others, which brings more trouble for LSTMs in paragraph encoder to build logical representations. Compared to the result in the sentence ordering task, Entity Grid (Barzilay and Lapata, 2008) achieves a good performance in this task and even outperforms Recurrent neural networks and Recursive neural networks (Li and Hovy, 2014) on Accident dataset. However, Entity Grid requires hand-engineered features and heavily relies on linguistic knowledge which restrain the model to be adapted to other tasks.

4 Conclusion

In this paper, we develop a novel deep attentive sentence ordering model (referred as ATTOrderNet) integrating self-attention mechanism with LSTMs. It enables us to directly capture logical relationships among sentences regardless of their input order and obtain a reliable representation of the sentence set. With this representation, a pointer network is applied to generate an ordered sequence. ATTOrderNet is evaluated on Sentence Ordering and Order Discrimination tasks. The experimental results demonstrate its effectiveness and show promising improvements over existing models across most datasets.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61702448, 61672456) and the Fundamental Research Funds for the Central Universities (No. 2017QNA5008, 2017FZA5007). We thank all reviewers for their valuable comments.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Regina Barzilay and Noemie Elhadad. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL*, pages 113–120.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Neural sentence ordering. *arXiv preprint arXiv:1607.06952*.

Micha Elsner and Eugene Charniak. 2008. Coreference-inspired coherence modeling. In *ACL*, pages 41–44.

Micha Elsner and Eugene Charniak. 2011. Extending the entity grid with entity-specific features. In *ACL*, pages 125–129.

Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. 2012. Extractive multi-document summarization with integer linear programming and support vector regression. *COLING*, pages 911–926.

Jingjing Gong, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. End-to-end neural sentence ordering using pointer network. *arXiv preprint arXiv:1611.04953*.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *ACL*, volume 1, pages 93–103.

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

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *NAACL*, pages 1233–1239.

Ioannis Konstas and Mirella Lapata. 2012a. Concept-to-text generation via discriminative reranking. In *ACL*, pages 369–378. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2012b. Unsupervised concept-to-text generation with hypergraphs. In *NAACL*, pages 752–761. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *EMNLP*, pages 1503–1514.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *ACL*, pages 545–552. Association for Computational Linguistics.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *EMNLP*, pages 2039–2048.

Jiwei Li and Dan Jurafsky. 2017. Neural net models of open-domain discourse coherence. In *EMNLP*, pages 198–209.

Lajanugen Logeswaran, Honglak Lee, and Dragomir R. Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *AAAI*.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *EMNLP-CONLL*, pages 1157–1168. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *COLING/ACL*, pages 803–810. Association for Computational Linguistics.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2017. Deep semantic role labeling with self-attention. *arXiv preprint arXiv:1712.01586*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

Suzan Verberne. 2011. Retrieval-based question answering for machine reading evaluation. In *CLEF (Notebook Papers/Labs/Workshop)*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*, pages 2692–2700.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*, pages 1480–1489.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Getting to “Hearer-old”: Charting Referring Expressions Across Time

Ieva Staliūnaitė[‡] **Hannah Rohde**[‡‡] **Bonnie Webber**[†] **Annie Louis**[†,¶]
‡Utrecht Institute of Linguistics, Utrecht University, Netherlands
‡‡Linguistics and English Language, University of Edinburgh, UK
†School of Informatics, University of Edinburgh, UK
¶The Alan Turing Institute, London, UK
`i.r.staliunaite@students.uu.nl`
`{hannah.rohde, bonnie.webber, annie.louis}@ed.ac.uk`

Abstract

When a reader is first introduced to an entity, its referring expression must describe the entity. For entities that are widely known, a single word or phrase often suffices. This paper presents the first study of how expressions that refer to the same entity develop over time. We track thousands of person and organization entities over 20 years of *New York Times* (NYT). As entities move from *hearer-new* (first introduction to the NYT audience) to *hearer-old* (common knowledge) status, we show empirically that the referring expressions along this trajectory depend on the type of the entity, and exhibit linguistic properties related to becoming common knowledge (e.g., shorter length, less use of appositives, more definiteness). These properties can also be used to build a model to predict how long it will take for an entity to reach hearer-old status. Our results reach 10-30% absolute improvement over a majority-class baseline.

1 Introduction

While today the company Google is so well known that its name can even be used as a verb, in 2002, it was referred to in *The New York Times*[1] as “Google, the company behind the popular Web search engine”. The appositive told the readers what the company does, whereas now such elaboration is needed rarely, if at all. This paper presents a first computational study that relates the form of an entity’s referring expressions (RE) in articles written at different times to the entity’s changing information status.[2]

Previous work has focused on predicting how REs for an entity vary *within a single text*. This type of information status can improve coreference resolution (Recasens et al., 2013) and help generate references in automatic summaries (Nenkova and McKeown, 2003). But there has been little exploration of the change in REs to an entity *over time* and *across articles*, as the entity is accepted into common knowledge. The current work is driven by linguistic interest in characterizing REs over time. In addition, knowing the current acceptance of an entity can help in generating time-appropriate expressions. From a social science perspective, there is also great interest in capturing the birth, acceptance into common parlance, but also possible death, and subsequent reintroductions of entities.

In this paper, we disambiguate and track thousands of person (PER) and organization (ORG) entities in the *New York Times Annotated Corpus* (Sandhaus, 2008) across 20 years of news. We extract and analyze hundreds of thousands of REs for these entities, and provide the first empirical evidence that the expressions used to refer to an entity grow shorter over time, and that properties such as definiteness increase. The properties of RE form are also not uniform over entities, and we identify systematic differences between PER and ORG.

We also present a model that predicts the future information status of an entity. The model takes the REs in a small snapshot (a month’s span) from anywhere in an entity’s timeline, and predicts how long it will take the entity to reach hearer-old (common knowledge) status. Features related to mention frequency, and content, syntax, and topic of the REs are highly predictive, giving accuracies in the 60-80% range. We also make our corpus available for future work on REs over time.

2 Background

The choice of an RE depends on the availability and novelty of an entity (Prince, 1992). A mention may be first or later *within a text* (discourse-new

---

[1] article 1386221 from Sandhaus (2008)

[2] Corpus available at `http://groups.inf.ed.ac.uk/cup/ref/`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4350–4359
Brussels, Belgium, October 31 - November 4, 2018.

and discourse-old respectively), or either newly introduce the entity to an audience (hearer-new) or be part of common knowledge (hearer-old). According to Prince (1992), hearer-old entities are more often mentioned with definite expressions, since the hearer can pick out the unique referent based on background knowledge. Corpus studies (Nenkova and McKeown, 2003; Yoshida, 2011) have corroborated similar trends that subsequent mentions to established entities *within a discourse* tend to be reduced or definite noun phrases.

In computational work, many studies (Nissim, 2006; Rahman and Ng, 2011; Markert et al., 2012) classify entities in a text as discourse-new/old. Here, hearer-old entities are a separate *mediated* class (not introduced in the text but which readers infer based on common knowledge), and predicted using features such as definiteness and the entity name itself. In the context of text summarization, the system of Siddharthan et al. (2011) classifies entities in source documents as hearer-old or not, based on frequency, syntax, and coreference (within the documents). The predicted status is then used in a rule-based algorithm to generate references in summaries of the source. Earlier work (Radev, 1998) modeled the choice of the best expression (from a lexicon) to fit the specific semantic context during text generation. Supplementing these efforts, we model the *progression* of entities' status from hearer-new to hearer-old as it changes across, rather than within, documents.

In the social sciences, Graus et al. (2017) analyzed distributions of entity mentions, and intervals between mentions, to identify patterns of entities becoming common knowledge, but without looking at the content of the REs. The coinage and subsequent acceptance/extinction of lexical innovations is another domain that models expressions over time, often by mapping properties of the speakers who use them within a community (see Tredici and Fernandez, 2018 and work reviewed therein). Our focus here is on REs specifically.

In what follows, we explain our RE extraction (Section 3) and linguistic features (Section 4). Analysis of the REs and the model for information status prediction are in Sections 5 and 6.

3 Extracting REs over time

We use the *New York Times Annotated Corpus* (NYTAC) (Sandhaus, 2008), containing the 1.8M articles published in the *New York Times* over the period 1987–2007 (20 years).

Given the complexity in identifying potentially interesting entities and disambiguating references to them over time, we limited our scope to person (PER) and organization (ORG) entities, which we could disambiguate with high accuracy. We also set aside years 2003–2007 for future validation, and used 1990–2002 for all our training, validation, and testing. Across these 13 years[3], we collected 52,338 unique entities (74% PER and 26% ORG) that were mentioned 284,064 times (65% PER mentions and 35% ORG).

3.1 RE span detection

We used the Stanford CoreNLP toolkit (Manning et al., 2014) (version 2018-01-31) to obtain constituency parse trees, and perform coreference resolution on the articles.[4]

We then extract all noun phrases (NP), including NPs that are nested in other NPs. For each NP, we identify if it is a *proper name* RE by matching the structure of its children to a set of six patterns (Table 1) of syntactic structures that can be used in describing an entity: *pre-modifier, relative clause, appositive, participle clause, adjective/adverb clause and prepositional phrase.*

Pattern 1 (pre-modifier) is used to identify NPs which are headed (rightmost leaf, except for possessives) by a proper noun (NNP). If an NP matches pattern 1, we have found a proper noun RE. Otherwise, if the NP matches one of the other patterns, we recursively match the patterns for the head NP phrase (bolded in Table 1) until one of them matches pattern 1. For *appositives*, there is no consensus about which NP should be the head of the phrase, as the entity name can be syntactically realized as the first or the second NP. Here we process both NPs. If there is no match for pattern 1 in this process, we discard the RE because the entity name is not a proper noun phrase. If we find multiple overlapping RE spans headed by the same noun, we only keep the largest NP.

After identifying a proper noun NP, the sequence of NNP children in the component NP span that matched pattern 1 is treated as the *base expression* (the entity name). The rest of the RE is the *descriptor* (description of the entity). For example, the phrase 'the pilot, First Lieut. Kelly Flinn' matches the *appositive* pattern.

---

[3] Our corpus contains trajectories for 20 years.

[4] The shift-reduce parser (Zhu et al., 2013) has an F1 score of 90% on the Wall Street Journal corpus (results from 2014).

Type	**Pattern for the children of an NP**	**Example NP**	**Base expression**
1. Pre-modifier	DT? (JJ | JJR | JJS | VBG | CD | QP | NP | NN | NNS | NNP | NNPS | PRP | , | CC | HYPH | SYM)* POS? *NNP*+ POS?	The tedious, complicated ABC	ABC
2. Relative clause	**NP** ,? SBAR ,?	The International Business Machines Corporation, which is the second-biggest advertiser on the Internet	International Business Machines
3. Appositive	**NP** (, | :)? **NP** (, | :)?	International Business Machines Corporation, the worlds largest computer company	International Business Machines Corporation
4. Participle clause	**NP** ,? VP ,?	International Business Machines Corporation, based in Armonk, N.Y.	International Business Machines Corporation
5. Adjective or adverb clause	**NP** ,? (ADJP | ADVP) ,?	Western Resources Inc., worth $1.7 billion	Western Resources Inc.
6. Prepositional phrase	**NP** ,? PP ,?	The National Basketball Association in New York	National Basketball Association

Table 1: Regular expressions (regex) for finding RE spans within an NP. The regex use Penn Treebank (Marcus et al., 1993) tags.

The full phrase is the RE, the string of NNPs ‘First Lieut. Kelly Flinn’ that comprise the embedded NP is the base expression, and the remaining NP ‘the pilot’ is the descriptor.

Certain adjustments had to be made. For example, NPs of the form ‘NNP of NNP’ (e.g. “University of Virginia”) are treated as base expressions. In addition, some connectives and symbols were included in the base to accommodate names such as “Food and Drug Administration”. These adjustments lead to their own errors. For instance, for “Dan Zegart of Titusville”, our exception rule will mark the full expression as the base, while we would prefer just the name. As expected, our RE spans are also subject to parsing errors.

But we found that the REs mined are largely robust. We manually annotated 100 randomly selected REs for correctness of the full RE span and base expression. 92 were correctly identified. Most errors involve prepositional phrase (PP) attachment, so PP modifiers are ignored in further analysis (except PPs within base expressions).

3.2 Finding cross-document entity chains

We next identify ORG and PER entities, and link the REs for the same entities over the entire span of the NYTAC. While cross-document coreference is usually a hard problem, we took advantage of NYTAC metadata to identify and link mentions to the same ORG and PER entity with high precision.

STEP 1: Our analysis focuses on salient and repeatedly appearing entities in the news. Therefore using position in the lead paragraph as a proxy for salience, we only include entities mentioned at least once in the first three sentences of each article. We also filter out entities that appear in fewer than two documents in our corpus.

Then within an article, we extract coreference chains to identify unique entities and their REs. For each entity, we then identify a *single RE* in the article which is indicative of its hearer-old/new status. One would expect the first mention to be performing this task as later mentions are discourse-old, which itself affects the form of the REs. But we found that entities are not necessarily introduced in the first mention, hence for each entity we take the *longest descriptor* among the first three mentions in an article.

STEP 2: Next we link mentions across articles. Each NYTAC article has metadata tags which *inter alia* name *salient* people and organizations appearing in the article. These tags uniquely identify an entity every time it appears in the corpus.

We match these tags to the article REs. The tags contain normalized entity names which may not match the article’s REs (from STEP 1) exactly. So we perform matches at the level of coreference chains. A chain is matched to a tagged name if the words from the base expressions in that chain overlap highly with the tag. In the case of people, the last name of the tagged person had to match the last proper noun in one of the base expressions. For ORG, at most one word in the base expression could be missing in the tag, unless an acronym of the tagged name was used as the base expression. We manually annotated the correctness of tag matches for a random sample of 25 each of PER and ORG entities. 88% of PER and 96% of ORG matches were accurate indicating high precision. All the PER match mistakes involved two people with the same last name mentioned in same article, e.g. Bill Clinton and Hillary Clinton.[5] As an estimate of recall, we find that 61.1% of all PER metadata tags, and 59.5% of ORG tags were matched to an entity, which is reasonable. Note also that we only match when an entity is mentioned within the first three sentences of an article.

Once tags are matched to article REs, the linking of REs across the corpus is also complete since the tags are already linked. The extracted REs for each entity are ordered by the publication date of the article, creating a chain of time-ordered REs for that entity across the entire corpus.

For an example of the result of our complete timeline extraction method, consider the entity Boris Yeltsin. Mr. Yeltsin became leader of the Russian parliament in May 1990, and subsequently the President in June 1991. Both events are within the timeframe of our corpus. Below we list some expressions taken from different timepoints in Mr. Yeltsin's path to presidency and later.

Spring 1990	Mr. Yeltsin, the popular chairman of the Russian Parliament who has emerged as the champion of radical reform and decentralization and as the prime political rival to the Soviet President, Mikhail S. Gorbachev
Fall 1990	Boris Yeltsin, champion of the slender insurgent minority at the Communist Party congress
Spring 1991	Boris Yeltsin, the president of the Russian federated republic and chief opposition critic of Mr. Gorbachev
Fall 1991	Boris Yeltsin, the president of the Russian republic
Spring 1992	President Boris Yeltsin
Spring 1993	Mr. Yeltsin
Spring 2000	Former Russian President Boris Yeltsin

Figure 1 shows the average length of descriptors for Mr. Yeltsin: 5 or 6 words up until Spring 1991, after which the average length is 1 word or below, indicating a significant shift in information status.

3.3 Defining hearer-old status

Next we designate an entity's mentions as hearer-new, mentions which are hearer-old, and those whose information status lies in between.

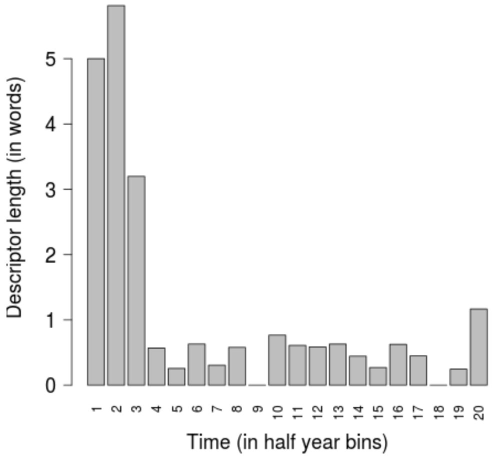

Figure 1: Descriptor lengths for Mr. Boris Yeltsin starting Spring 1990

Hearer-new mention. The NYTAC spans the years 1987–2007. To identify entities which are new, we treat the years 1987 to 1989 as a prior background corpus. When an entity is mentioned in the news from 1990 onwards but which never appeared in the prior corpus, the entity is considered as 'hearer-new' at the first mention.[6]

Mr. Yeltsin (our previous example) in fact resigned from the Politburo in 1987 before emerging again as leader of Parliament in 1990. Interestingly, our method (possibly) correctly identifies Mr. Yeltsin as hearer-new in 1990 based on comparison with the prior corpus of 1987-89.

Entity mention trajectories. From the previous section, we have the trajectories of REs for any identified hearer-new entity up to the end of our data (2002). We exclude entities that have a gap of more than 6 months between consecutive mentions, because such long gaps require the author to reintroduce an entity in case readers had forgotten it. Figure 2 shows how the length of descriptors tends to increase with greater time gaps.

Also, different entities are introduced at different times in the span from 1990 to 2002. To normalize their introduction time, we define the idea of *age* for an entity at a certain time point, so

---

[5] Coreference chains are also noisy in these cases.

[6] We noticed that sometimes multiple unique metadata tags exist for the same entity, mostly reflecting major changes in what the entity is known for, for example an entity before and after becoming a President. This pattern is not consistent in the editorial conventions however, so we leave improvements to cross-document coreference for future work.

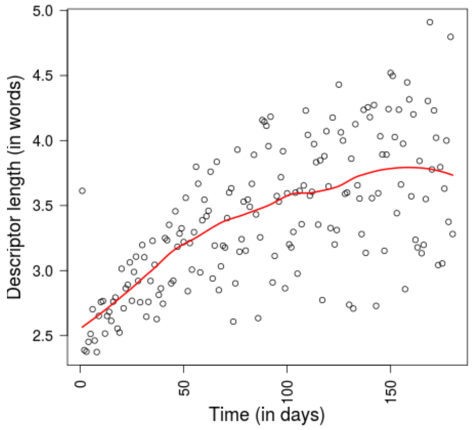

Figure 2: Mean descriptor length vs. time since previous mention

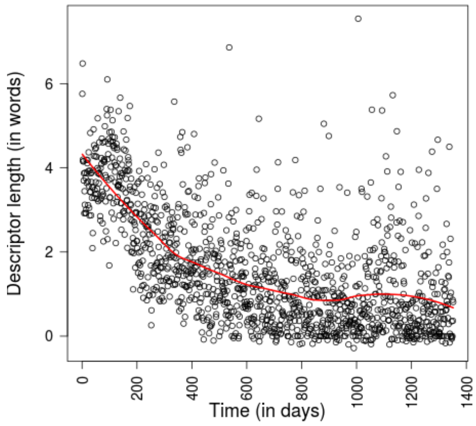

Figure 3: Mean descriptor length vs 'age' of entity

that all entities start out at time zero. For every later mention of the entity, we record the *age* as the number of days since first mention. Figure 3 shows a clear decrease in length of descriptors with increasing age of the entity.

Hearer-old mentions. We hypothesize that hearer-old entities are referred to by bare names only, and only occasionally by longer REs. Thus, we define *acceptance* of a hearer-old entity as the time after which its descriptors (the additional words besides the base expression) do not exceed a length of n words on average. Also rather than define this time point based on a single mention (which would be fragile), we bin the age values into month spans. The information status of an entity will undergo little change within a month.

First, we calculate the average length of all descriptors (entire timeline) for a given entity. If it is less than n words, then we designate the entity as hearer-old within the first month. Otherwise, we find the first month in the timeline after which the average length of descriptors of all remaining mentions is below n, and use it as time of acceptance. If the entity never reaches the threshold, we conclude that it is not accepted in the time span of the corpus. We tested values for n of 0.5 words, 1 word, and 2 words, and chose the 1-word threshold which performed best (in our linear model and classification experiments which follow).

Table 2 shows the distribution of the acceptance age for entities in our corpus. While PERs are accepted faster, more of the PER entities are also never accepted within the corpus timespan.

Below we list some examples, and qualitative observations of PER entities from each bin. The frequency of mentions is indicated within [].

1 month bin involves prominent public figures:

Bill Clinton [6005]; Michael Jordan [8]; Michael Bloomberg [546]; Nelson Mandela [3]

within 1 year are authors, journalists, local politicians (senators, mayors):

Judith Kaye (a centrist on the New York State Court of Appeals) [6]; (Senator-elect) Hillary Rodham Clinton [86]; Jurate Kazickas (a freelance writer) [2]

1-13 years involve politicians and their families, people linked to famous criminal cases:

Lisa Olson (a reporter for the Boston Herald) [14]; Laura Bush (the wife of president-elect George W. Bush) [47]; Boris Yeltsin (President of the powerful Russian Republic, whom Mr. Gorbachev portrayed as a destructive opportunist) [1229]; Joseph Gambino (a convicted heroin trafficker) [9]; Abner Louima (the Haitian immigrant who the prosecutors say was tortured by New York City police officers in a Brooklyn station house) [276]

longer than timespan involves those mentioned along a longer period but not famous (lawyers, economists, doctors), or people mentioned a lot within a short period (wedding announcements, serial killers, accident victims) and people reintroduced from the past:

Mildred Natwick (a versatile actress who created an engaging gallery of eccentric, whimsical and spunky characters in plays, films and television for more than 60 years) [2], Irene Neal (a 53 year old sculptor and painter) [2]

4 Features to characterize REs

We group the mentions of an entity within each month, and treat the *group* as one example. In this

Accepted after	less than 1 month	1 to 12 months	1 to 13 years	longer than timespan of corpus
People (entity level)	48.39%	6.42%	0.24%	44.95%
Organizations (entity level)	57.61%	8.66%	0.86%	32.89%
People (RE level)	46.92%	7.59%	2.47%	43.02%
Organizations (RE level)	49.71%	10.77%	6.67%	32.85%

Table 2: Distribution of acceptance ages. The 'entity level' rows record *one age* for each entity since its first mention. 'RE level' is the distribution when the remaining age is calculated from each RE mention. The timelines are truncated at year 2002. The later years are kept as a test set.

ORGANIZATION	**PERSON**
Positive indefinite article, length 0, length 0-3, length 3-10, length 10-20, average length, gap between mentions, topic:transport	**Positive** definite article, pre-modifier, age, length 0-3, length 3-10, length 10-20, average length, section:cars and lifestyle, topic:sports, topic: entertainment
Negative relative clause, definite article, named entity, topic:international relations, topic:military and politics	**Negative** relative clause, appositive, possessive, named entity, length 0, topic:transport, topic:religion, topic:awards

Table 3: Significant main effects in linear model

way, the example contains RE choices for the entity at *a certain snapshot in time*. A set of 61 features are computed for each example to characterize the REs (descriptor part only) it contains. *None* of the features involve the identity of the entity.

Descriptor contents: include the number of PER and ORG named entities if any in the descriptor text, type/token ratio, the counts of date and money, adjectives, superlative adjectives, verbs, and honorifics (based on a list with words such as Judge, President, Dr. etc). All counts are normalized by the number of REs in the example.

Syntactic form: They include average number per RE of definite articles, indefinite articles and possessive constructions, appositives, participle clauses, relative clauses, adjective or adverb clauses, and pre-modification. We also record the length of the descriptors using 5 bins (0 words, 1-3, 3-10, 10-20 and >20 words). The feature value is the proportion of REs in a bin. A binary feature also indicates whether the average length of descriptors is below 0.5 words.

Frequency of mentions: We include the number of mentions within the one month bin, and the average time gap between consecutive mentions (multiplied by the $\log$ of number of mentions to compensate for frequency). We also include the entity's current age (time since introduction).

Context: We employ the topic metadata from NYTAC to capture a notion of world context of an entity's mentions. Every article has topic tags (sports, finance, technology, politics, etc.) and also a section label (travel, economics, culture, etc.). We clustered the thousands of topic tags into 20 broad topics by using the Glove word embeddings (Pennington et al., 2014), and K-means clustering. A small set of 17 clusters were also created for the newspaper sections. The count of mentions belonging to articles in each cluster is a feature.

5 Features versus acceptance time

We built a linear model (LM) to test which features are significantly predictive of time to acceptance.

An example is the set of REs for an entity from a one month bin. The dependent variable is the *time left until acceptance*, i.e. the value (in months) from the current age of the entity (month were the mentions were taken from) until the acceptance age. For our corpus, the possible values are 0 to 156. Entities that do not become well known within the span of our dataset are given a label of 160 months. Entities which fall out of use (never mentioned in the corpus after a certain time point but also have not reached acceptance threshold at their found last mentions) have a label of 161.[7]

We also performed a variance analysis over the LM with ANOVA to account for possible correlation between variables. We scale the feature values into z-scores which show how many standard deviations each example is away from the mean.

We used *lm* and *Anova* (Fox and Weisberg, 2011) functions within R (R Core Team, 2013). The adjusted R^2 value is 43% for PER and 37%

---

[7] One could also perform survival analysis for this problem, we have not explored it yet.

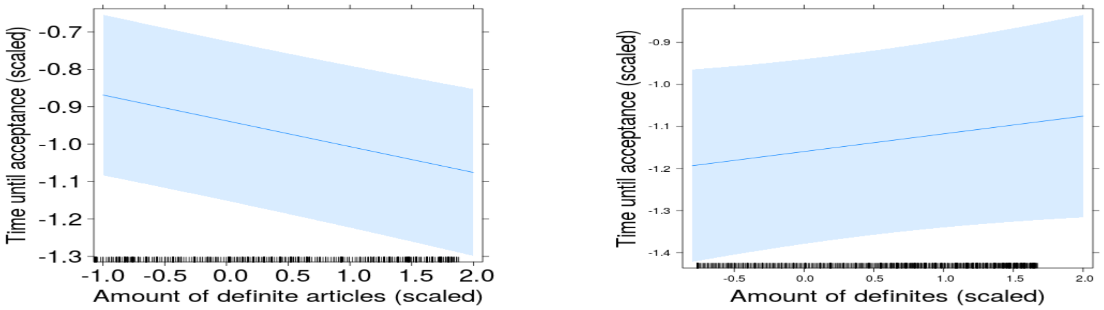

(a) Definite articles in ORG descriptors

(b) Definite articles in PER descriptors

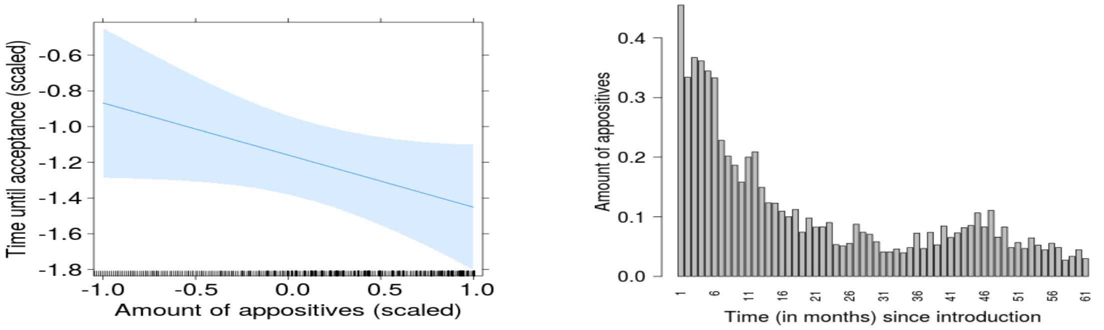

(c) Appositives in PER descriptors

(d) Proportion of PER mentions with appositive descriptors, at different ‘ages’ of an entity (within the first 5 years)

Figure 4: Main effects for definite articles and appositives

for ORG entities. The significant features ($p < 0.05$) from the ANOVA are in Table 3. The features are divided into positive and negative ones, reflecting whether a higher value of the feature was predictive of a higher (positive) or lower (negative) value of the time until acceptance. More features are significant for PER (18) compared to ORG (13) expressions. Also for persons, the significant features are more varied compared to ORG, indicating that people may have more varied roles and characterizations presented in the media. Note also that we have more data for PER entities.

Below we describe some of the most interesting findings from the model.

5.1 Main effects

Definite articles are a significant feature for both PER and ORG entities. In Figures 4a and 4b, we plot the variation in *time until acceptance* versus number of definite articles, holding other variables at their means. The bands show the 0.95 level confidence interval. We used the *effects* package (Fox, 2003) in R, and the graphs show scaled values.

Assuming that the use of definite articles indicates definite expressions roughly, for ORG entities, the results confirm the hypothesis that REs become more definite the longer the entity is in use. REs such as “Ebay, an online auction site” become “Ebay, the largest of the auction sites” and “Ebay, the auction site” before settling to “Ebay”.

The pattern for people is opposite, with more definite articles used with early mentions. Upon closer observation, we found that in fact, the expressions are more often definite when they are closer to acceptance, but with definiteness not expressed by the article. The definite article is excluded, and the possessive is used to produce a concise phrase e.g. “Russia’s acting president” instead of “the acting president of Russia”. The possessive phrase is short and also has fewer function words, which is consistent with psycholinguistic findings that people tend to reduce highly predictable phrases by dropping function words (Jaeger and Levy, 2007). Moreover, we observed that for ORG, an article may still describe the entity after acceptance e.g. “The FDA”, and understandably this pattern does not exist for persons.

Appositives are not significant predictors for ORG; for PER, more appositives is associated with closeness to acceptance (Figure 4c). Yet early mentions involve more appositives than later ones (Figure 4d). One interpretation of these contrast-

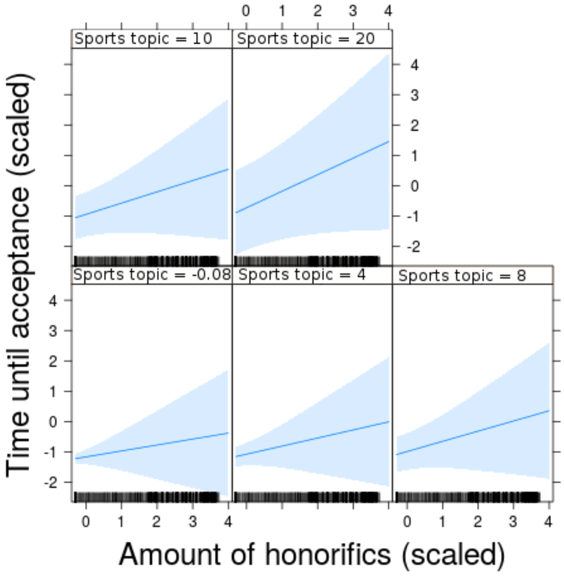

Figure 5: Interaction between the effect of sports topic and honorifics (both scaled). Sub-figures represent proportions of mentions in sports topics.

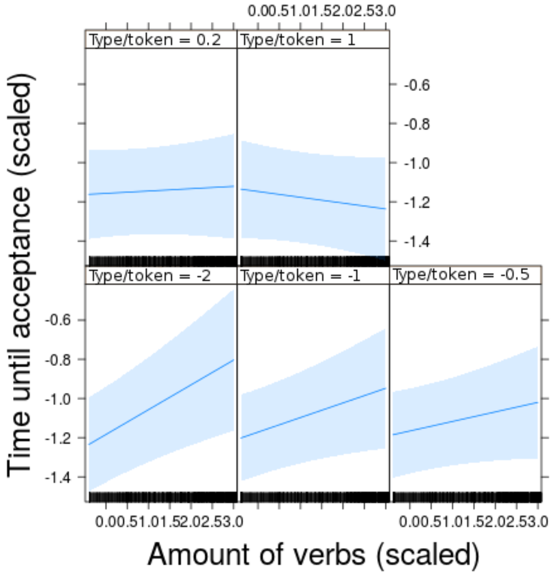

Figure 6: Interaction between the effect of type/token ratio and verb use (both scaled). Sub-figures represent values of the type/token ratio.

ing tendencies could be that there are two types of entities. The first are those important enough to be introduced with lots of appositives at the beginning of their 'lifespan'. These entities could be accepted quickly, and this behavior may manifest as having more appositives close to acceptance. The other set is that of non-salient entities which are probably not introduced with appositives anyway, and show slow acceptance. This analysis is supported by our observations that entities which are not as important as the context they appear in, such as a lesser known football player who scores a goal or an architect of a famous building, are often introduced without an appositive.

5.2 Interactions

We also found significant interactions in the model highlighting differences between various types of entities. For example, there is no main effect for honorifics but there is a significant positive effect when sports entities are involved (Figure 5). It is possible that REs such as "the number one player" indicate closeness to acceptance more than "the president of the National Hockey League", as honorifics (e.g. 'president') are not used with most known people in the sports domain.

The presence of verbs (in the descriptor) also does not have a main effect, but becomes meaningful when the type/token ratio of descriptors of a person is low (see Figure 6). We consider the type/token ratio as reflecting the different guises that a person is mentioned with, and verbs in the descriptor as associating the person with an event rather than their role in society. When the type/token ratio is low, it suggests the person is known for one or a few aspects. Here acceptance time and verbs are positively related, implying that when the few aspects are events (more verbs), the entities have a slower path to hearer-old. Whereas, if the aspects were titles such as CEO (less verbs), the entity is closer to acceptance.

6 Predicting future information-status

Given the significant correlations between our features and the time to acceptance, we now build a predictive model to identify acceptance time given a snapshot of REs for an entity. Every example contains a certain entity's mentions over a month's span, similar to the linear model. The target class indicates whether or not the entity was accepted within a period, x years from the sample time.

The classification models were built separately for PER and ORG expressions. We divide the data into 70% training, 10% validation and 20% test. For number of data points for training/validation/test, ORG has 8,768/1,252/2,506 data points; PER has 26,190/3,742/7,483. Note that examples are month-sized bins of each en-

ENTITY	Baseline (%) (majority-class)	SVM			MLP			
		C	Gamma	Accuracy (%)	Layers	Units	Alpha	Accuracy (%)
4-CLASS								
Org	35.17	10	0.01	62.69	2	400	100	68.36
People	42.01	100	0.01	70.77	3	300	100	74.18
BINARY								
Org	69.80	10	0.01	78.41	3	100	100	79.41
People	59.16	10	0.01	79.14	3	300	100	80.66

Table 4: Results of the SVM and MLP classifiers. C and Alpha are regularization parameters, Gamma parameterizes the RBF kernel. Hidden layers and hidden units in the MLP are also shown.

tity's REs, rather than unique entities. We explore both an SVM classifier with an RBF kernel, and a Multi Layer Perceptron (MLP) classifier. The MLP uses a BFGS solver with ReLU activation function. We used implementations from scikit-learn (Pedregosa et al., 2011), and the parameters of both classifiers were tuned on the development set using grid search.

We build two types of classifiers (results in Table 4). The baseline is majority-class assignment.

4-CLASS is a four-label classifier for predicting *when* an entity will become hearer-old. The four classes reflect the distribution of acceptance ages (see Table 2—RE level): already accepted (at the sample month), will be accepted within the year, between 1 and 13 years from the sample, and will not be accepted within the time frame we have.

BINARY is a classifier performing a simpler binary division of whether an entity is hearer-old or hearer-new *after 2 years* from the sampling time. About half of both entities in our data are accepted within 2 years after introduction.

The 4-CLASS MLP model is better than SVM reaching 68% accuracy for ORG and 74% for people. The improvement is over 30% absolute value, indicating the significant effect of the model and features. Still there is scope for improvement, given that the performance is less than 75%.

Both SVM and MLP perform similarly for the binary tasks. The overall binary classification accuracy is 80% for both PER and ORG entities, a 10% increase for ORG and 20% for PER.

We also performed an ablation study to identity the most useful individual classes of features. Syntax features (Section 4) had the biggest impact when removed, lowering performing by 5-8% for all models. But since both our classifiers are non-linear, they can capture useful interactions between all our feature classes.[8]

7 Conclusion

In this work, we have shown empirically that the path to hearer-old status displays detectable and interesting linguistic features, and that entities of a certain type exhibit distinctive properties. These significant differences have allowed for a first model which predicts how long it will take for an entity to be accepted as common knowledge.

There are a number of directions which we plan to explore. While we have focused on predicting whether an entity will be accepted after a certain period of time has passed, we have not modeled the RE tokens themselves, or their generation. During our analysis, it was clear that the REs of a (named) entity may also change semantically over time, reflecting current interest in it, rather than the same RE content just growing shorter over time. We have also not explored the RE timelines of common noun entities such as *organic food* or *cryptocurrency*. It is possible that they follow a different trend than named entities, and require a different set of feature indicators. We also plan to improve upon our current models and assumptions. Currently, we have ignored entities with long time gaps and requiring reintroduction to an audience. Learning to predict when an entity will go out of use, and indicators for reintroduction will add strength to our analyses. The definition of acceptance and fine-grained models for prediction will also be developed. We are also releasing a corpus of chained REs from the NYTAC (represented as byte-span sets) to enable other researchers to study these aspects of REs.

Acknowledgments

Staliūnaitė was funded under EU COST Action IS1312 (*TextLink*). Rohde was supported by a Leverhulme Trust Prize in Languages & Literatures. We also thank Jan Odijk, Mark Steedman, and our anonymous reviewers for their comments.

---

[8] The *context* class uses corpus-specific metadata, but unsupervised topic modeling could likely approximate it.

References

John Fox. 2003. Effect displays in R for generalised linear models. *Journal of Statistical Software*, 8(15):1–27.

John Fox and Sanford Weisberg. 2011. *An R Companion to Applied Regression*, second edition. Sage, Thousand Oaks CA.

David Graus, Daan Odijk, and Maarten de Rijke. 2017. The birth of collective memories: Analyzing emerging entities in text streams. *arXiv preprint arXiv:1701.04039*.

Florian T. Jaeger and Roger P. Levy. 2007. Speakers optimize information density through syntactic reduction. In *Proceedings of Advances in Neural Information Processing Systems*, pages 849–856.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Katja Markert, Yufang Hou, and Michael Strube. 2012. Collective classification for fine-grained information status. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pages 795–804.

Ani Nenkova and Kathleen McKeown. 2003. References to named entities: A corpus study. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Short Papers - Volume 2*, pages 70–72.

Malvina Nissim. 2006. Learning information status of discourse entities. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 94–102.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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

Ellen F Prince. 1992. The ZPG letter: Subjects, definiteness, and information-status. *In Sandra Thompson and William Mann, editors, Discourse description: diverse analyses of a fund raising text*, pages 295–325.

R Core Team. 2013. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Dragomir R Radev. 1998. Learning correlations between linguistic indicators and semantic constraints: Reuse of context-dependent descriptions of entities. In *Proceedings of the 17th International Conference on Computational linguistics-Volume 2*, pages 1072–1078.

Altaf Rahman and Vincent Ng. 2011. Learning the information status of noun phrases in spoken dialogues. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1069–1080.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 627–633.

Evan Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2011. Information status distinctions and referring expressions: An empirical study of references to people in news summaries. *Computational Linguistics*, 37(4):811–842.

Marco Del Tredici and Raquel Fernandez. 2018. The road to success: Assessing the fate of linguistic innovations in online communities. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Etsuko Yoshida. 2011. *Referring expressions in English and Japanese: Patterns of use in dialogue processing*, volume 208. John Benjamins Publishing.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 434–443.

Making “fetch” happen: The influence of social and linguistic context on nonstandard word growth and decline

Ian Stewart and Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30318
{istewart6, jacobe}@gatech.edu

Abstract

In an online community, new words come and go: today’s *haha* may be replaced by tomorrow’s *lol*. Changes in online writing are usually studied as a social process, with innovations diffusing through a network of individuals in a speech community. But unlike other types of innovation, language change is shaped and constrained by the grammatical system in which it takes part. To investigate the role of social and structural factors in language change, we undertake a large-scale analysis of the frequencies of nonstandard words in Reddit. Dissemination across many linguistic contexts is a predictor of success: words that appear in more linguistic contexts grow faster and survive longer. Furthermore, social dissemination plays a less important role in explaining word growth and decline than previously hypothesized.

1 Introduction

Stop trying to make “fetch” happen! It’s not going to happen! – Regina George (*Mean Girls*, 2005)

With the fast-paced and ephemeral nature of online discourse, language change in online writing is both prevalent (Androutsopoulos, 2011) and noticeable (Squires, 2010). In social media, new words emerge constantly to replace even basic expressions such as laughter: today’s *haha* is tomorrow’s *lol* (Tagliamonte and Denis, 2008). Why do some nonstandard words, like *lol*, succeed and spread to new contexts, while others, like *fetch*, fail to catch on? Can a word’s growth be predicted from patterns of usage during its early days?

Language change can be treated like other social innovations, such as the spread of hyperlinks (Bakshy et al., 2011) or hashtags (Romero et al., 2011; Tsur and Rappoport, 2015). A key aspect of the adoption of a new practice is its *dissemination*: is it used by many people, and in many social contexts? High dissemination enables words to achieve greater exposure among social groups (Altmann et al., 2011), and may signal that the innovation is positively evaluated.

In addition to social constraints, language change is also shaped by grammatical constraints (D’Arcy and Tagliamonte, 2015). New words and phrases rarely change the rules of the game but must instead find their place in a competitive ecosystem with finely-differentiated linguistic roles, or “niches” (MacWhinney, 1989). Some words become valid in a broad range of linguistic contexts, while others remain bound to a small number of fixed expressions. We therefore posit a structural analogue to social dissemination, which we call *linguistic dissemination*.

We compare the fates of such words to determine how linguistic and social dissemination each relate to word growth, focusing on the adoption of nonstandard words in the popular online community Reddit. The following hypotheses are evaluated:

- **H1: Nonstandard words with higher initial social dissemination are more likely to grow.** Following the intuition that words require a large social base to succeed, we hypothesize a positive correlation between social dissemination and word growth.
- **H2-weak: Nonstandard words with higher linguistic dissemination in the early phase of their history are more likely to grow.** This follows from work in corpus linguistics showing that words and grammatical patterns with a higher diversity of collocations are more likely to be adopted (Ito and Tagliamonte, 2003; Partington, 1993).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4360–4370
Brussels, Belgium, October 31 - November 4, 2018.

- **H2-strong: Nonstandard words with higher linguistic dissemination are more likely to grow, even after controlling for social dissemination.** This follows from the intuition that linguistic context and social context contribute differently to word growth.

To address H2, we develop a novel metric for characterizing linguistic dissemination, by comparing the observed number of n-gram contexts to the number of contexts that would be predicted based on frequency alone. Our analysis of word growth and decline includes: (1) prediction of frequency change in growth words (as in prior work); (2) causal inference of the influence of dissemination on probability of word growth; (3) binary prediction of future growth versus decline; and (4) survival analysis, to determine the factors that predict when a word's popularity begins to decline. All tests indicate that linguistic dissemination plays an important role in explaining the growth and decline of nonstandard words.

2 Related Work

Lexical change online Language changes constantly, and one of the most notable forms of change is the adoption of new words (Metcalf, 2004), sometimes referred to as lexical entrenchment (Chesley and Baayen, 2010). New nonstandard words may arise through the mutation of existing forms by processes such as truncation (e.g, *favorite* to *fave*; Grieve et al., 2016) and blending (e.g., *web+log* to *weblog* to *blog*; Cook and Stevenson, 2010). The fast pace and interconnected nature of online communication is particularly conducive to innovation, and social media provides a "birds-eye view" on the process of change (Danescu-Niculescu-Mizil et al., 2013; Kershaw et al., 2016; Tsur and Rappoport, 2015).

The most closely related work is a contemporaneous study that explored the role of weak social ties in the dissemination of linguistic innovations on Reddit, which also proposed the task of quantitatively predicting the success or failure of lexical innovations (Tredici and Fernández, 2018). One distinguishing feature of our work is the emphasis on *linguistic* (rather than social) context in explaining these successes and failures. In addition to predicting the binary distinction between success and failure, we also take on the more fine-grained task of predicting the length of time that each nonstandard word will survive.

Social dissemination Language changes as a result of transmission across generations (Labov, 2007) as well as diffusion across individuals and social groups (Bucholtz, 1999). Such diffusion can be quantified with *social dissemination*, which Altmann et al. (2011) define as the count of social units (e.g., users) who have adopted a word, normalized by the expected count under a null model in which the word is used with equal frequency across the entire population. Altmann et al. (2011) use dissemination of words across forum users and threads to predict the words' change in frequency in Usenet, finding a positive correlation between frequency change and both kinds of social dissemination. In contrast, Garley and Hockenmaier (2012) use the same metric to predict the growth of English loanwords on German hip-hop forums, and find that social dissemination has less predictive power than expected. We seek to replicate these prior findings, and to extend them to the broader context of Reddit.

Linguistic dissemination In historical linguistics, the distribution of a new word or construction across lexical contexts can signal future growth (Partington, 1993). Furthermore, grammatical and lexical factors can explain a speaker's choice of linguistic variant (Ito and Tagliamonte, 2003; Cacoullos and Walker, 2009) and can provide more insight than social factors alone. Our study proposes a generalizable method of measuring the dissemination of a word across lexical contexts with *linguistic* dissemination and compares social and linguistic dissemination as predictors of language change.

3 Data

Our study examines the adoption of words on social media, and we focus on Reddit as a source of language change. Reddit is a social content sharing site separated into distinct sub-communities or "subreddits" that center around particular topics (Gilbert, 2013). Reddit is a socially diverse and dynamic online platform, making it an ideal environment for research on language change (Kershaw et al., 2016). Furthermore, because Reddit data is publicly available we expect that this study can be more readily replicated than a similar study on other platforms such as Facebook or Twitter, whose data is less easily obtained.

	Total	Monthly mean
Comments	1,625,271,269	45,146,424
Tokens	56,674,728,199	1,574,298,006
Subreddits	333,874	48,786
Users	14,556,010	2,302,812
Threads	102,908,726	3,079,780

Table 1: Data summary statistics.

We analyze a set of public monthly Reddit comments[1] posted between 1 June 2013 and 31 May 2016, totalling $T = 36$ months of data. This dataset has been analyzed in prior work (Hessel et al., 2016; Tan and Lee, 2015) and has been noted to have some missing data (Gaffney and Matias, 2018), although this issue should not affect our analysis. To reduce noise in the data, we filter all comments generated by known bots and spam users[2] and filter all comments created in well-known non-English subreddits.[3] The final data collected is summarized in Table 1.

We replace all references to subreddits and users (marked by the convention *r/subreddit* and *u/user*) with *r/SUB* and *u/USER* tokens, and all hyperlinks with a *URL* token. We also reduce all repeated character sequences to a maximum length of three (e.g., *loooool* to *loool*). The final vocabulary includes the top 100,000 words by frequency.[4] We replace all OOV words with UNK tokens, which comprise 3.95% of the total tokens.

3.1 Finding growth words

Our work seeks to study the growth of nonstandard words, which we identify manually instead of relying on pre-determined lists (Tredici and Fernández, 2018).To detect such words, we first compute the Spearman correlation coefficient between the time steps $\{1...T\}$ and each word w's frequency time series $f^{(w)}_{(1:T)}$ (frequency normalized and log-transformed). The Spearman correlation coefficient captures monotonic, gradual growth that characterizes the adoption of nonstandard words (Grieve et al., 2016; Kershaw et al., 2016).

The first set of words is filtered by a Spearman correlation coefficient above the 85th percentile ($N = 15,017$). From this set of words, one of the authors manually identified 1,120 words in set $\mathcal{G}$ ("growth") that are neither proper nouns (*berniebot, killary, drumpf*) nor standard words (*election, voting*).[5] These words were removed because their growth may be due to exogenous influence. A "standard" word is one that can plausibly be found in a newspaper article, which follows from the common understanding of newspaper text as a more formal and standard register. Therefore, a "nonstandard" word is one that cannot plausibly be found in a newspaper article, a judgment often used by linguists to determine what counts as slang (Dumas and Lighter, 1978). In ambiguous cases, one of the authors inspected a sample of comments that included the word. We validate this process by having both authors annotate the top 200 growth candidates for standard/proper versus nonstandard (binary), obtaining inter-annotator agreement of κ=0.79.

3.2 Finding decline words

To determine what makes the growth words successful, we need a control group of "decline" words, which are briefly adopted and later abandoned. Although these words may have been successful before the time period investigated, their decline phase makes them a useful comparison for the growth words. We find such words by fitting two parametric models to the frequency series.

Piecewise linear fit We fit a two-phase piecewise linear regression on each word's frequency time series $f_{(1:T)}$, which splits the time series into $f_{(1:\hat{t})}$ and $f_{(\hat{t}+1:T)}$. The goal is to select a split point $\hat{t}$ to minimize the sum of the squared error between observed frequency f and predicted frequency $\hat{f}$:

$$\hat{f}(m_1, m_2, b, t) = \begin{cases} b + m_1 t & t \leq \hat{t} \\ b + m_1\hat{t} + m_2(t - \hat{t}) & t > \hat{t}, \end{cases} \tag{1}$$

where b is the intercept, m_1 is the slope of the first phase, and m_2 is the slope of the second phase. Decline words $\mathcal{D}_p$ ("piecewise decline") display

---

[1] From `http://files.pushshift.io/reddit/comments/` (Accessed 1 October 2016).

[2] The same list used in Tan and Lee (2015): `https://chenhaot.com/data/multi-community/README.txt` (Accessed 1 October 2016).

[3] We randomly sampled 100 posts from the top 500 subreddits and labelled a subreddit as non-English if fewer than 90% of its posts were identified by `langid.py` (Lui and Baldwin, 2012) as English.

[4] We restricted the vocabulary because of the qualitative analysis required to identify nonstandard words.

[5] Code and word lists available at: `https://github.com/ianbstewart/nonstandard_word_dissemination`.

Word set	Examples
$\mathcal{G}$	*idk, lmao, shitpost, tbh, tho*
$\mathcal{D}_l$	*atty, eyebleach, iifym, obeasts, trashy*
$\mathcal{D}_p$	*brojob, nparent, rekd, terpers, wot*

Table 2: Examples of nonstandard words in all word sets: growth ($\mathcal{G}$), logistic decline ($\mathcal{D}_l$) and piecewise decline ($\mathcal{D}_p$).

growth in the first phase ($m_1 > 0$), decline in the second phase ($m_2 < 0$), and a strong fit between observed and predicted data, indicated by $R^2(f, \hat{f})$ above the 85^{th} percentile (36.1%); this filtering yields 14,995 candidates.

Logistic fit To account for smoother growth-decline trajectories, we also fit the growth curve to a logistic distribution, which is a continuous unimodal distribution with support over the non-negative reals. We identify the set of candidates $\mathcal{D}_l$ ("logistic decline") as words with a strong fit to this distribution, as indicated by R^2 above the 99^{th} percentile (82.4%), yielding 998 candidates. The logistic word set partially overlaps with the piecewise set, because some words' frequency time series show a strong fit to both the piecewise function and the logistic distribution.

Combined set We combine the sets $\mathcal{D}_p$ and $\mathcal{D}_l$ to produce a set of decline word candidates ($N = 15,665$). Next, we filter this combined set to exclude standard words and proper nouns, yielding a total of 530 decline words in set $\mathcal{D}$. Each word is assigned a split point $\hat{t}$ based on the estimated time of switch between the growth phase and the decline phase, which is the split point $\hat{t}$ for piecewise decline words and the center of the logistic distribution $\hat{\mu}$ for the logistic decline words.

Examples of both growth and decline words are shown in Table 2. The growth words include several acronyms (*tbh*, "to be honest"; *lmao*, "laughing my ass off"), while the decline words include clippings (*atty*, "atomizer"), respellings (*rekd*, "wrecked"; *wot*, "what") and compounds (*nparent*, "narcissistic parent").

We also provide a distribution of the words across word generation categories in Table 3, including compounds and clippings in similar proportions to prior work (Kulkarni and Wang, 2018). Because the growth and decline words exhibit similar proportions of category counts, we do not expect that this will be a significant confound in differentiating growth from decline.

	Clipping	Compound	Respelling	Other	Total
$\mathcal{G}$	198 (17.7%)	334 (29.8%)	83 (7.4%)	505 (45.1%)	1,120
$\mathcal{D}$	53 (10.0%)	100 (18.9%)	108 (20.4%)	269 (50.8%)	530

Table 3: Word formation category counts in growth ($\mathcal{G}$) and decline ($\mathcal{D}$) word sets.

4 Predictors

We now outline the predictors used to measure the degree of **social** and **linguistic** dissemination in the growth and decline words.

4.1 Social dissemination

We rely on the dissemination metric proposed by Altmann et al. (2011) to measure the degree to which a word occupies a specific social niche (e.g., low dissemination implies limited niche). To compute user dissemination D^U for word w at time t, we first compute the number of individual users who used word w at time t, written $U_t^{(w)}$. We then compare this with the expectation $\tilde{U}_t^{(w)}$ under a model in which word frequency is identical across all users. The user dissemination is the log ratio,

$$\log \frac{U_t^{(w)}}{\tilde{U}_t^{(w)}} = \log U_t^{(w)} - \log \tilde{U}_t^{(w)}. \qquad (2)$$

Following Altmann et al. (2011), the expected count $\tilde{U}_t^{(w)}$ is computed as,

$$\tilde{U}_t^{(w)} = \sum_{u \in \mathcal{U}_t} (1 - e^{-f_t^{(w)} m_t^{(u)}}), \qquad (3)$$

where $m_t^{(u)}$ equals the total number of words contributed by user u in month t, and $\mathcal{U}_t$ is the set of all users active in month t. This corresponds to a model in which each token from a user has identical likelihood $f_t^{(w)}$ of being word w. In this way, we compute dissemination for all users (D^U), subreddits (D^S) and threads (D^T) for each month $t \in \{1...T\}$.

4.2 Linguistic dissemination

Linguistic dissemination captures the diversity of linguistic contexts in which a word appears, as measured by unique n-gram counts. We compute

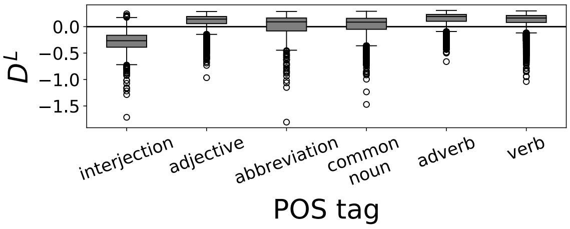

Figure 1: Distribution of mean linguistic dissemination (D^L) across part of speech groups.

the log count of unique trigram[6] contexts for all words (C^3) using all possible trigram positions: in the sentence "*that's cool af haha*", the term *af* appears in three unique trigrams, *that's cool af, cool af haha, af haha <END>*.

The unique log number of trigram contexts is strongly correlated with log word frequency ($\rho(C^3, f) = 0.904$), as implied by Heaps' law (Egghe, 2007). We therefore adjust this statistic by comparing with its expected value $\tilde{C}^3$. At each timestep t, we fit a linear regression between log-frequency and log-unique n-gram counts, and then compute the residual between the observed log count of unique trigrams and its expectation, $D^L = C^3_t - \tilde{C}^3_t$. The residual D^L, or *linguistic dissemination*, identifies words with a higher or lower number of lexical contexts than expected.

Linguistic dissemination can separate words by grammatical category, as shown in Figure 1 where the mean D^L values are computed for words across common part-of-speech categories. Part-of-speech tags were computed over the entire corpus using a Twitter-based tagger (Gimpel et al., 2011), and each word type was assigned the most likely POS tag to provide an approximate distribution of tags over the vocabulary. Interjections have a lower median D^L than other word categories due to the tendency of interjections to occur in limited lexical contexts. Conversely, verbs have a higher median D^L due to the flexibility of verbs' arguments (e.g., subject and object may both be open-class nouns).

5 Results

The hypotheses about social and linguistic dissemination are tested under four analyses: correlation against frequency change in growth words; causal inference on probability of word growth; binary prediction of word growth; and survival analysis of decline words.

	Variance explained	Lower, upper 95%
f_{t-12}	10.8%	[10.2%, 11.5%]
D^L_{t-12}	0.584%	[0.461%, 0.777%]
D^U_{t-12}	0.307%	[0.251%, 0.398%]
D^S_{t-12}	0.120%	[0.0852%, 0.191%]
D^T_{t-12}	0.246%	[0.171%, 0.379%]
f_{t-24}	21.4%	[20.4%, 22.4%]
D^L_{t-24}	1.29%	[1.05%, 1.64%]
D^U_{t-24}	0.400%	[0.346%, 0.493%]
D^S_{t-24}	0.287%	[0.201%, 0.392%]
D^T_{t-24}	0.272%	[0.226%, 0.380%]

Table 4: Percent of variance explained in frequency change, computed over all growth words $\mathcal{G}$. $N = 26,880$ for $k = 12$, $N = 13,440$ for $k = 24$.

5.1 Correlational analysis

To test the relative importance of the linguistic and social context on word growth, we correlate these metrics with frequency change ($\Delta_{f_t} = f_t - f_{t-k}$) across all growth words. This replicates the methodology in prior work by Altmann et al. (2011) and Garley and Hockenmaier (2012), who analyzed different internet forums. Focusing on long-term change with $k = 12$ (one year) and $k = 24$ (two years), we compute the proportion of variance in frequency change explained by the covariates using a relative importance regression (Kruskal, 1987).[7]

The results of the regression are shown in Table 4. All predictors have relative importance greater than zero, according to a bootstrap method to produce confidence intervals (Tonidandel et al., 2009). Frequency is the strongest predictor (f_{t-12}, f_{t-24}), because words with low initial frequency often show the most frequency change. In both short- and long-term prediction, linguistic dissemination (D^L_{t-12}, D^L_{t-24}) has a higher relative importance than each of the social dissemination metrics. The social dissemination metrics have less explanatory power, in comparison with the other predictors and in comparison to the prior results of Garley and Hockenmaier (2012), who found 1.5% of variance explained by D^U and 1.9% for D^T at $k = 24$. Our results were robust to the exclusion of the predictor D^L, meaning that

---

[6] Pilot analysis with bigram contexts gave similar results.

[7] Relative importance regression implemented in the *relaimpo* package in R: `https://cran.r-project.org/package=relaimpo`

a model with only the social dissemination metrics as predictors resulted in a similar proportion of variance explained. The weakness of social dissemination could be due to the fragmented nature of Reddit, compared to more intra-connected forums. Since users and threads are spread across many different subreddits, and users may not visit multiple subreddits, a higher social dissemination for a particular word may not lead to immediate growth.

5.2 Causal analysis

While correlation can help explain the relationship between dissemination and frequency change, it only addresses the weak version of H2: it does not distinguish the causal impact of linguistic and social dissemination. To test the strong version of H2, we turn to a causal analysis, in which the *outcome* is whether a nonstandard word grows or declines, the *treatment* is a single dissemination metric such as linguistic dissemination, and the *covariates* are the remaining dissemination metrics. The goal of this analysis is to test the impact of each dissemination metric, while holding the others constant.

Causal inference typically uses a binary treatment/control distinction (Angrist et al., 1996), but in this case the treatment is continuous. We therefore turn to an adapted model known as the *average dose response function* to measure the causal impact of dissemination (Imbens, 2000). To explain the procedure for estimating the average dose response, we adopt the following terminology: Z for treatment variable, X for covariates, Y for outcome.[8]

1. A linear model is fit to estimate the treatment from the covariates,

$$Z_i \mid X_i \sim \mathcal{N}(\beta^\top X_i, \sigma^2). \tag{4}$$

The output of this estimation procedure is a vector of weights $\hat{\beta}$ and a variance $\hat{\sigma}^2$.

2. The generalized propensity score (GPS) R is the likelihood of observing the treatment given the covariates, $P(Z_i \mid X_i)$. It is computed from the parameters estimated in the previous step:

$$\hat{R}_i = \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} \exp\left(-\frac{(Z_i - \hat{\beta}^\top X_i)^2}{2\hat{\sigma}^2}\right). \tag{5}$$

3. A logistic model is fit to predict the outcome Y_i using the treatment Z_i and the GPS $\hat{R}_i$:

$$\hat{Y}_i = \text{Logistic}(\hat{\alpha}_0 + \hat{\alpha}_1 Z_i + \hat{\alpha}_2 \hat{R}_i). \tag{6}$$

This involves estimating the parameters $\{\hat{\alpha}_0, \hat{\alpha}_1, \hat{\alpha}_2.\}$ By incorporating the generalized propensity score $\hat{R}_i$ into this predictive model over the outcome, it is possible to isolate the causal effect of the treatment from the other covariates (Hirano and Imbens, 2004).

4. The range of treatments is divided into levels (quantiles). The *average dose response* for a given treatment level s_z is the mean estimated outcome for all instances at that treatment level,

$$\hat{\mu}(s_z) = \frac{1}{|s_z|} \sum_{z_i \in s_z} \hat{Y}_i. \tag{7}$$

The average dose response function is then plotted for all treatment levels.

Each dissemination metric is considered separately as a treatment. We consider all other dissemination metrics and frequency as covariates: e.g., for treatment variable D^L, the covariates are set to $[f, D^U, D^S, D^T]$. We bootstrap the above process 100 times with different samples to produce confidence intervals. To balance the outcome classes, we sample an equal number of growth and decline words for each bootstrap iteration.

The average dose response function curves in Figure 2 show that linguistic dissemination (D^L) produces the most dramatic increase in word growth probability. For linguistic dissemination, the lowest treatment quantile (0%-10%) yields a growth probability below 40% (significantly less than chance), as compared to the highest treatment quantile (90-100%), which yields a growth probability nearly at 70% (significantly greater than chance). This supports the strong form of H2, which states that linguistic dissemination is predictive of growth, even after controlling for the frequency and the other dissemination metrics. Subreddit dissemination also shows a mild causal effect on word growth, up to 60% in the highest treatment quantile. The other social dissemination metrics prove to have less effect on word growth.

---

[8] Average dose response function implemented in the *causaldrf* package in R: `https://cran.r-project.org/package=causaldrf`

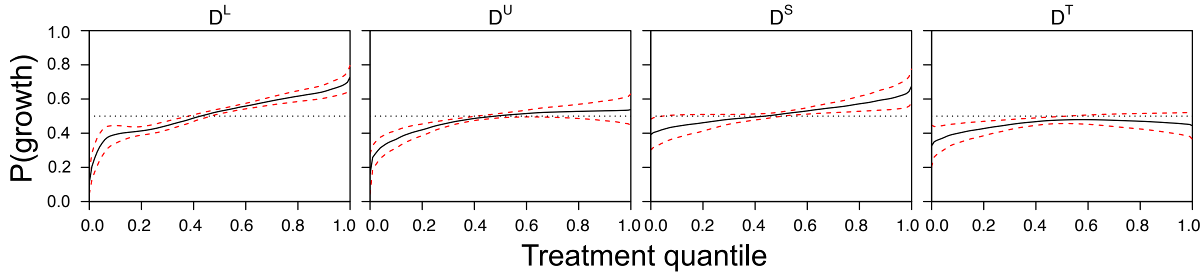

Figure 2: Average dose response function for all treatment variables, where outcome is probability of word growth. 95% confidence intervals plotted in red, chance rate of 50% marked with dotted black line.

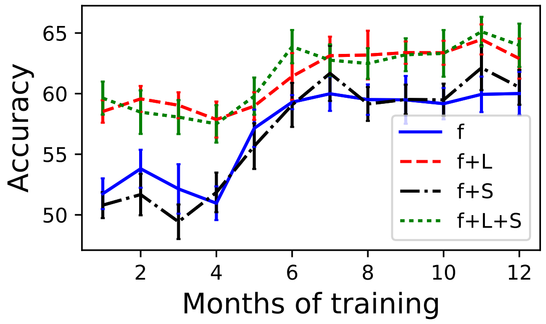

Figure 3: Prediction accuracy for different feature sets using $k = 1...12$ months of training data. f indicates frequency-only, $f+L$ frequency plus linguistic dissemination, $f+S$ frequency plus social dissemination, $f+L+S$ all features.

5.3 Predictive analysis

We now turn to prediction to determine the utility of linguistic and social dissemination: using the first k months of data, can we predict whether a word will grow or decline in popularity? This is similar to previous work in predicting the success of lexical innovations (Kooti et al., 2012), but our goal is to compare the relative predictive power of various dissemination metrics, rather than to maximize accuracy.

We use logistic regression with 10-fold cross-validation over four different feature sets: frequency-only (`f`), frequency plus linguistic dissemination (`f+L`), frequency plus social dissemination (`f+S`) and all features (`f+L+S`). Each fold is balanced for classes so that the baseline accuracy is 50%. Figure 3 shows that linguistic dissemination provides more predictive power than social dissemination: the accuracy is consistently higher for the models with linguistic dissemination than for the frequency-only and social dissemination models. The accuracies converge as the training data size increases, which suggests that frequency is a useful predictor if provided sufficient historical trajectory.

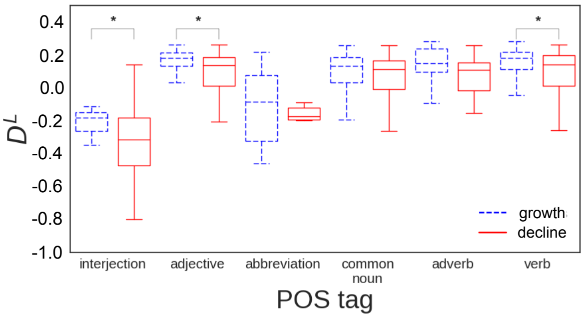

Figure 4: Distribution of D^L values across growth and decline words, grouped by part of speech tag. * indicates $p < 0.05$ in one-tailed t-test between growth and decline D^L values.

Part-of-speech robustness check Considering the uneven distribution of linguistic dissemination across part-of-speech groups (Figure 1), the prediction results may be explained by an imbalance of word categories between the growth and decline words. This issue is addressed through two robustness checks: within-group comparison and prediction.

First, we compare the distribution of linguistic dissemination values between growth and decline words, grouped by the most common POS tags (computed in § 4.2). Each decline word is matched with a growth word based on similar mean frequency in the first $k = 12$ months, and their mean linguistic dissemination values during that time period are compared, grouped within POS tag groups. The differences in Figure 4 show that across all POS tags, the growth words show a tendency toward higher linguistic dissemination with significant ($p < 0.05$) differences in the interjections, adjectives and verbs.

Next, we add POS tags as additional features to the frequency-only model in the binary prediction task. The accuracy of a predictive model with access to frequency and POS features at $k = 1$ is 54.8%, which is substantially lower than the accuracy of the model with frequency and linguistic dissemination (cf. Figure 3).[9] Thus, linguistic dissemination thus contributes predictive power beyond what is contributed by part-of-speech alone.

5.4 Survival analysis

Having investigated what separates growth from decline, we now focus on the factors that precede a decline word's "death" phase (Drouin and Dury, 2009).

Predicting the time until a word's decline can be framed as survival analysis (Klein and Moeschberger, 2005), in which a word is said to "survive" until the beginning of its decline phase at split point $\hat{t}$. In the Cox proportional hazards model (Cox, 1972), the hazard of death λ at each time t is modeled as a linear function of a vector of predictors,

$$\lambda_i(t) = \lambda_0(t) \exp(\beta \cdot \mathbf{x}_i), \tag{8}$$

where $\mathbf{x}_i$ is the vector of predictors for word i, and β is the vector of coefficients. Each cell $x_{i,j}$ is set to the mean value of predictor j for word i over the training period $t = \{1...k\}$ where $k = 3$.

For words which begin to decline in popularity in our dataset, we treat the point of decline as the "death" date. The remaining words are viewed as *censored* instances: they may begin to decline in popularity at some point in the future, but this time is outside our frame of observation. We use frequency, social dissemination and linguistic dissemination as predictors in a Cox regression model.[10]

The estimated coefficients from the regression are shown in Table 5. We find a negative coefficient for linguistic dissemination ($\beta = -0.330, p < 0.001$), which mirrors the results from § 5.2: higher D^L indicates a lower hazard of word death, and therefore a higher likelihood of survival. We also find that higher subreddit dissemination has a weak but insignificant correlation with a lower likelihood of word death ($\beta = -0.156, p > 0.05$). Both of these results lend additional support to the strong form of the hypothesis H2.

Predictor	β	std. error	Z	p
f	-0.207	0.0492	-4.21	***
D^{L}	-0.330	0.0385	-8.56	***
D^{U}	0.0053	0.0518	0.102	
D^{S}	-0.156	0.0807	-1.928	
D^{T}	0.0825	0.0662	1.25	

Table 5: Cox regression results for predicting word death with all predictors (`f+L+S`) averaged over first $k = 3$ months. *** indicates $p < 0.001$, otherwise $p > 0.05$.

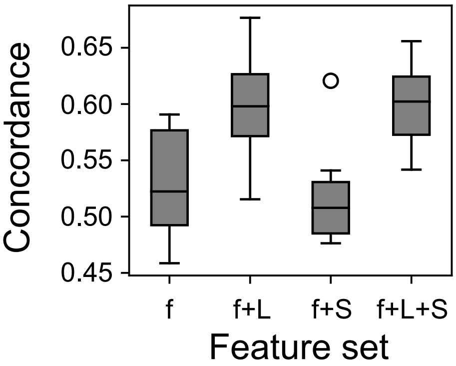

Figure 5: Distribution of concordance scores (10-fold cross-validation) of the Cox regression models across feature sets.

The predictive accuracy of survival analysis can be quantified by a *concordance score*. A score of 1.0 on heldout data indicates that the model perfectly predicts the order of death times; a score of 0.5 indicates that the predictions are no better than a chance ordering. We perform 10-fold cross-validation of the survival analysis model, and plot the results in Figure 5. The model with access to linguistic dissemination (`f+L`) consistently achieves higher concordance than the baseline frequency-only model (`f`), ($t = 4.29, p < 0.001$), and the model with all predictors `f+L+S` significantly outperforms the model with access only to frequency and social dissemination `f+S` ($t = 4.64, p < 0.001$). The result is reinforced by testing the goodness-of-fit for each model with model *deviance*, or difference from the null model. The `f+L` model has lower deviance, i.e. better fit, than the null model ($\chi^2 = 93.3, p < 0.01$), and the `f+L+S` does not have a significantly lower deviance than the `f+L` model ($\chi^2 = 4.6, p = 0.80$),

---

[9] Higher k values yield similar results.

[10] Cox regression implemented in the *lifelines* package in Python: `https://lifelines.readthedocs.io/en/latest/`.

suggesting that adding social dissemination does not significantly improve model fit.

6 Discussion

All four analyses support H2: linguistic dissemination was the strongest predictor of monthly frequency changes in growth words, the best differentiator of growth and decline words in causal and predictive tasks, and the most effective warning sign that a word is about to decline. Linguistic dissemination can be related to theories such as the FUDGE factors (Chesley and Baayen, 2010; Cook, 2010; Metcalf, 2004), in which a word's growth depends on frequency (F), unobtrusiveness (U), diversity of users and situations (D), generation of other forms and meanings (G), and endurance (E). Linguistic dissemination provides an example of "diversity of situation."

The effectiveness of linguistic dissemination is exemplified in pairs of semantically similar growth and decline words. In the first $k = 3$ months of growth, the growth word *kinda* has a relatively high ratio of linguistic to frequency ($\frac{D^L}{f} = 0.270$) as compared with the semantically similar decline word *sorta* (0.055). This pattern holds for other pairs of semantically similar growth/decline words: *fuckwit* and *fuckboy*; *lolno* and *lmao*; *yup* and *yas*. While not exhaustive, such a trend suggests that the growth words were able to reach a wider range of lexical contexts and therefore succeed where the decline words failed.

Regarding H1, we generally found a positive role for social dissemination as well, although these results were not consistent across all metrics and tests, particularly in the survival analysis. This matches the conclusion from Garley and Hockenmaier (2012), who argued that social dissemination is less predictive of word adoption than Altmann et al. (2011) originally suggested. One possible explanation is the inclusion of word categories such as proper nouns in the analysis of Altmann et al. (2011); the adoption of such terms may rely on social dynamics more than the adoption of nonstandard terms. The lower predictive power of thread and user dissemination is also interesting and suggests that subreddits are more socially salient in terms of exposing nonstandard words to potential adopters.

Limitations One limitation in the study was the exclusion of orthographic and morphological features such as affixation, which has been noted as a predictor of word growth (Kershaw et al., 2016). Future work should incorporate these features as additional predictors. Our study also omitted borrowings, unlike prior work in word adoption that has focused on borrowings (Chesley and Baayen, 2010; Garley and Hockenmaier, 2012). Our early language-filtering steps eliminated most non-English words from the vocabulary, although it would have been interesting to examine loanword use in English-language posts. Finally, our study was limited by the focus on nonstandard words rather than memetic phrases (e.g., *like a boss*) which may show a similar correlation between dissemination, growth and decline (Bybee, 2006).

Future work We approximate linguistic dissemination using trigram counts, because they are easy to compute and they generalize across word categories. In future work, a more sophisticated approach might estimate linguistic dissemination with syntactic features such as appearance across different phrase heads (Kroch, 1989; Ito and Tagliamonte, 2003) or across nouns of different semantic classes (D'Arcy and Tagliamonte, 2015). Future work should also investigate more semantically-aware definitions of linguistic dissemination. The existence of semantic "neighbors" occurring in similar contexts (e.g., the influence of standard intensifier *very* on nonstandard intensifier *af*) may prevent a new word from reaching widespread popularity (Grieve, 2018).

Acknowledgments

The authors thank the anonymous reviewers, the audience at NWAV 46 for discussion of an early presentation of this study, and the members of Georgia Tech's Computational Linguistics Lab for their help throughout the project. This research was supported by NSF award IIS-1452443, AFOSR award FA9550-14-1-0379, and NIH award R01-GM112697-03.

References

Eduardo Altmann, Janet Pierrehumbert, and Adilson Motter. 2011. Niche as a determinant of word fate in online groups. *PLoS ONE*, 6(5):1–12.

Jannis Androutsopoulos. 2011. Language change and digital media: a review of conceptions and evidence. In Kristiansen Tore and Nikolas Coupland, editors, *Standard Languages and Language Standards in* *a Changing Europe*, pages 145–159. Novus Press, Oslo.

Joshua D Angrist, Guido W Imbens, and Donald B Rubin. 1996. Identification of Causal Effects Using Instrumental Variables. *Source Journal of the American Statistical Association*, 91(434):444–455.

Eytan Bakshy, Jake Hofman, Winter Mason, and Duncan Watts. 2011. Everyone's an influencer: quantifying influence on Twitter. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 65–74.

Mary Bucholtz. 1999. "Why be normal?": Language and identity practices in a community of nerd girls. *Language in Society*, 28:203–223.

Joan L. Bybee. 2006. From Usage to Grammar: The Mind's Response to Repetition. *Language*, 82(4):711–733.

Rena Torres Cacoullos and James Walker. 2009. The Present of the English Future: Grammatical Variation and Collocations in Discourse. *Language*, 85(2):321–354.

Paula Chesley and Harald Baayen. 2010. Predicting new words from newer words: Lexical borrowings in French. *Linguistics*, 48(6):1343–1374.

Paul Cook. 2010. *Exploiting linguistic knowledge to infer properties of neologisms*. Ph.D. thesis, University of Toronto.

Paul Cook and Suzanne Stevenson. 2010. Automatically identifying the source words of lexical blends in English. *Computational Linguistics*, 36(1):129–149.

David Cox. 1972. Regression models and life tables. *Journal of the Royal Statistical Society*, 34:187–220.

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, and Christopher Potts. 2013. No Country for Old Members: User Lifecycle and Linguistic Change in Online Communities. *Proceedings of the International Conference on World Wide Web*, pages 307–317.

Alexandra D'Arcy and Sali Tagliamonte. 2015. Not always variable: probing the vernacular grammar. *Language Variation and Change*, 27(3):255–285.

Patrick Drouin and Pascaline Dury. 2009. When terms disappear from a specialized lexicon: A semi-automatic investigation into necrology. In *Actes de la conférence internationale "Language for Special Purposes"*.

BK Dumas and Jonathan Lighter. 1978. Is slang a word for linguists? *American Speech*, 53(1):5–17.

Leo Egghe. 2007. Untangling Herdan's law and Heaps' Law: Mathematical and informetric arguments. *Journal of the American Society for Information Science and Technology*, 58(5):702–709.

Devin Gaffney and J. Nathan Matias. 2018. Caveat emptor, computational social science: Large-scale missing data in a widely-published reddit corpus. *PLoS ONE*, 13(7).

Matt Garley and Julia Hockenmaier. 2012. Beefmoves: dissemination, diversity, and dynamics of English borrowings in a German hip hop forum. In *Proceedings of the Association of Computational Linguistics*, pages 135–139.

Eric Gilbert. 2013. Widespread Underprovision on Reddit. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 803–808.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 42–47.

Jack Grieve. 2018. Natural selection in the modern English Lexicon. In *International Conference on Language Evolution*, pages 153–157.

Jack Grieve, Andrea Nini, and Diansheng Guo. 2016. Analyzing lexical emergence in Modern American English online. *English Language and Linguistics*, 20(2):1–29.

Jack Hessel, Chenhao Tan, and Lillian Lee. 2016. Science, AskScience, and BadScience: On the coexistence of highly related communities. In *Proceedings of the International Conference on Social and Web Media*, pages 171–180.

Keisuke Hirano and Guido W Imbens. 2004. The propensity score with continuous treatments. In Andrew Gelman and Xiao-Li Meng, editors, *Applied Bayesian modeling and causal inference from incomplete-data perspectives*, pages 73–84. Wiley, Chichester.

Guido W Imbens. 2000. The role of the propensity score in estimating dose-response functions. *Biometrika*, 87(3):706–710.

Rika Ito and Sali Tagliamonte. 2003. Well weird, right dodgy, very strange, really cool: Layering and recycling in English intensifiers. *Language in Society*, 32:257–279.

Daniel Kershaw, Matthew Rowe, and Patrick Stacey. 2016. Towards Modelling Language Innovation Acceptance in Online Social Networks. *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 553–562.

John Klein and Melvin Moeschberger. 2005. *Survival analysis: techniques for censored and truncated data*. Springer Science & Business Media, New York.

Farshad Kooti, Winter A Mason, Krishna P Gummadi, and Meeyoung Cha. 2012. Predicting emerging social conventions in online social networks. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 445–454.

Anthony S Kroch. 1989. Reflexes of grammar in patterns of language change. *Language Variation and Change*, 1(3):199–244.

William Kruskal. 1987. Relative importance by averaging over orderings. *The American Statistician*, 41(1):6–10.

Vivek Kulkarni and William Yang Wang. 2018. Simple Models for Word Formation in English Slang. In *In Proceedings of the North American Association of Computational Linguistics*, pages 1424–1434.

William Labov. 2007. Transmission and Diffusion. *Language*, 83(2):344–387.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the Association of Computational Linguistics*, pages 25–30.

Brian MacWhinney. 1989. Competition and Lexical Categorization. *Linguistic Categorization*, pages 195–242.

Allan Metcalf. 2004. *Predicting new words: The secrets of their success*. Houghton Mifflin, New York.

Alan Partington. 1993. Corpus evidence of language change: The case of the intensifier. In Mona Baker, Gill Francis, and Elena Tognini-Bonelli, editors, *Text and Technology: In Honour of John Sinclair*, pages 177–192. John Benjamins Publishing, Philadelphia.

Daniel Romero, Brendan Meeder, and Jon Kleinberg. 2011. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on Twitter. In *Proceedings of the International Conference on World Wide Web*, pages 695–704. ACM.

Lauren Squires. 2010. Enregistering internet language. *Language in Society*, 39:457–492.

Sali Tagliamonte and Derek Denis. 2008. Linguistic ruin? LOL! Instant messaging and teen language. *American Speech*, 83(1):3–34.

Chenhao Tan and Lillian Lee. 2015. All who wander: On the prevalence and characteristics of multi-community engagement. In *Proceedings of the International Conference on World Wide Web*, pages 1056–1066.

Scott Tonidandel, James LeBreton, and Jeff Johnson. 2009. Determining the statistical significance of relative weights. *Psychological methods*, 14(4):387.

Marco Del Tredici and Raquel Fernández. 2018. The Road to Success: Assessing the Fate of Linguistic Innovations in Online Communities. In *Proceedings of the International Conference on Computational Linguistics*, pages 1591–1603.

Oren Tsur and Ari Rappoport. 2015. Don't Let Me Be #Misunderstood: Linguistically Motivated Algorithm for Predicting the Popularity of Textual Memes. In *Proceedings of the International Conference on Social and Web Media*, pages 426–435.

Analyzing Correlated Evolution of Multiple Features Using Latent Representations

Yugo Murawaki
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
`murawaki@i.kyoto-u.ac.jp`

Abstract

Statistical phylogenetic models have allowed the quantitative analysis of the evolution of a single categorical feature and a pair of binary features, but correlated evolution involving multiple discrete features is yet to be explored. Here we propose latent representation-based analysis in which (1) a sequence of discrete surface features is projected to a sequence of independent binary variables and (2) phylogenetic inference is performed on the latent space. In the experiments, we analyze the features of linguistic typology, with a special focus on the order of subject, object and verb. Our analysis suggests that languages sharing the same word order are not necessarily a coherent group but exhibit varying degrees of diachronic stability depending on other features.

1 Introduction

Research on structural properties (typological features) of language, such as the order of subject, object and verb (examples are `SOV` and `SVO`) and the presence or absence of tone, is largely synchronic in nature. Since languages of the world exhibit an astonishing diversity, the sample of languages used in a typical typological study is selected from a diverse set of language families and from various geographical regions. Not surprisingly, most of them lack historical documentation that allows us to directly trace their evolutionary history.

At the same time, however, typologists have long struggled to *dynamicize* synchronic typology, or to infer diachronic universals of change from current cross-linguistic variation (Greenberg, 1978; Nichols, 1992; Maslova, 2000; Bickel, 2013). They have also tried to uncover deep historical relations between languages (Nichols, 1992).

One of the main developments in diachronic typology in the last decade has been the application of powerful statistical tools borrowed from the field of evolutionary biology (Dediu, 2010; Greenhill et al., 2010; Dunn et al., 2011; Maurits and Griffiths, 2014; Greenhill et al., 2017). As illustrated in Figure 1, the key idea is that if a phylogenetic tree is given, we can infer the ancestral states with varying degrees of confidence, and by extension, can induce diachronic universals of change. To perform statistical inference, we assume that each feature evolves along the branches of the tree according to a continuous-time Markov chain (CTMC) model, which is controlled by a *transition rate matrix* (TRM). Once TRMs are estimated, we can gain insights from them, for example, by simulating language evolution (Maurits and Griffiths, 2014).

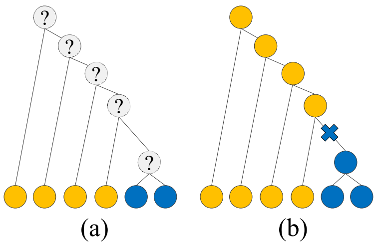

Figure 1: Phylogenetic comparative methods. Each node denotes a language with its state, or the value of its feature, indicated by color. (a) The task settings. A tree and the states of the leaf nodes are observed. The states of the internal nodes are latent variables to be inferred. (b) A result of inference. The cross sign on the branch indicates a change of the state.

One problem in previous studies is that they do not adequately model a characteristic of typological features that has been central to linguistic typology, that is, the fact that these features are not independent but depend on each other (Greenberg, 1963; Daumé III and Campbell, 2007). For example, if a language takes a verb before an object (`VO`), then it takes postnominal relative clauses

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4371–4382
Brussels, Belgium, October 31 - November 4, 2018.

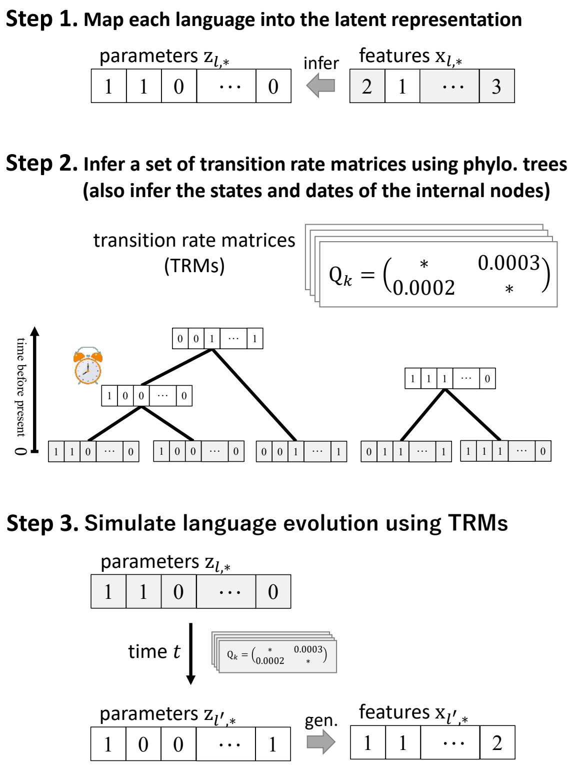

Figure 2: Overview of our framework. Observed and latent variables are marked in gray and white, respectively.

(`NRel`) (`VO` → `NRel`, in shorthand), and a related universal, `RelN` → `OV`, also holds (Dryer, 2011). Despite the long-standing interest in inter-feature dependencies, most statistical models assume independence between features (Daumé III, 2009; Dediu, 2010; Greenhill et al., 2010, 2017; Murawaki, 2016; Murawaki and Yamauchi, 2018). A rare exception is Dunn et al. (2011), who extended Greenberg's idea by applying a phylogenetic model of correlated evolution (Pagel and Meade, 2006). However, the model adopted by Dunn et al. (2011) can only handle the dependency between a pair of binary features. Typological features have two or more possible values in general, and more importantly, the dependencies between features are not limited to a pair (Itoh and Ueda, 2004). For example, the order of relative clauses has connections to the order of adjective and noun (`AdjN` or `NAdj`), in addition to the order of object and verb, as two universals, `RelN` → `AdjN` and `NAdj` → `NRel`, are known to hold well (Dryer, 2011).

In this paper, we propose latent representation-based analysis of diachronic typology. Figure 2 shows an overview of our framework. Following Murawaki (2017), we assume that a sequence of discrete surface features that represents a language is generated from a sequence of binary latent variables called *parameters* (Step 1). Parameters are, by assumption, independent of each other and switching one parameter entails multiple changes of surface features in general. Thus, by performing phylogenetic inference on the latent space, we can handle the dependencies of all available features in an implicit manner (Step 2). The latent parameter representation can be projected back to the surface feature representation when needed for analysis. Like Maurits and Griffiths (2014), we run simulation experiments to interpret the estimated model parameters (Step 3).

What we propose is a general framework with which we can analyze any discrete feature, but as a proof-of-concept demonstration, we follow Maurits and Griffiths (2014) in focusing on the order of subject, object and verb (hereafter simply referred to as basic word order or BWO).[1] In the dataset we use, the BWO feature has 7 possible values, 6 logically possible orders plus the special value `No dominant order` (Dryer, 2013b), meaning that it cannot be analyzed directly with Dunn et al.'s model. We show that languages sharing the same word order are not a coherent group but exhibit varying degrees of diachronic stability depending on other features.

2 Related Work

2.1 Statistical Diachronic Typology

The building block of statistical phylogenetic models[2] is a *time-tree*, which places nodes on an axis of time. In their standard applications to language (Gray and Atkinson, 2003; Bouckaert et al., 2012), time-trees are inferred from cognate

---

[1] We chose the BWO feature because it is appealing to a wider audience. We are aware that Matthew S. Dryer, who provided language data for the BWO feature, favors binary classifications (`OV` vs. `VO` and `SV` vs. `VS`) over the six-way classification (Dryer, 1997, 2013a). He argues that the binary classifications are more fundamental than the six-way classification, but our latent representation-based analysis does not require feature values to be primitive in nature because it reorganizes feature values into various latent parameters.

[2] Statistical phylogenetic models can be either distance-based and character-based. Character-based models are classified into parsimony-based and likelihood-based. In this paper, we focus on likelihood-based Bayesian models for their ability to date internal nodes. However, it is worth noting that attempts to overcome the limitations of the tree model mostly rely on non-likelihood-based models (Nakhleh et al., 2005; Nelson-Sathi et al., 2010).

	Dep.	# of language families	Tree sources	Dating	Abs.
Dediu (2010)	single	1	experts	yes	no
Greenhill et al. (2010)	single	1	cognates	no	yes
Maurits and Griffiths (2014)	single	1 or 7 combined	other	no	yes
Dunn et al. (2011)	bin. pair	1	cognates	no	yes
Ours	all	309 incl. 154 isolates	experts	yes	yes

Table 1: A comparison of time-tree-based approaches to diachronic typology. (1) Feature dependencies modeled: independent (single), a pair of binary features (bin. pair) or all dependencies considered (all). (2) The number of language families (i.e., trees) used for each run of phylogenetic inference. (3) The sources of the trees used in inference: tree topologies established by historical linguists (experts), time-trees reconstructed by phylogenetic models using cognate data (cognates), or time-trees obtained by distance-based clustering based on geographical coordinates or others (other). (4) Whether the dates of internal nodes are inferred (yes) or given a priori (no). (5) Whether absolute dates are obtained. If no, only dates relative to the root node are inferred.

data (Dyen et al., 1992; Greenhill et al., 2008).[3] However, if a tree is given a priori, phylogenetic models can also be used to estimate the parameters of a TRM, which controls how languages change their feature values over time. This is how typological features are analyzed in previous studies.

Dediu (2010) aggregated TRMs taken from various families to measure the stability of features. Greenhill et al. (2010) compared typological data with cognate data in terms of stability. Maurits and Griffiths (2014) focused on the BWO feature and analyzed how it had changed in the past and was likely to change in the future. Dunn et al. (2011) estimated TRMs for pairs of binary features and found that perceived correlated evolution was mostly lineage-specific rather than universal.

Taking a closer look at these studies, we can see that they vary as to how to prepare trees, as summarized in Table 1. Leaf nodes are assumed to be at the present date $t = 0$, but how can we assign backward dates t to internal nodes? A popular approach (Greenhill et al., 2010; Dunn et al., 2011) is to construct a time-tree with absolute (calendar) dates, using binary-coded lexical cognate data, and then to fit each trait of interest independently on the time-tree.[4]

However, cognate data are available only for a handful of language families such as Indo-European, Austronesian and Niger-Congo (or its mammoth Bantu branch). Moreover, phylogenetic inference was performed separately one after the other. This marks a sharp contrast with the long tradition of testing against a worldwide sample. In fact, it is suggested that sample diversity and aggregate time depth are not large enough to draw meaningful conclusions (Croft et al., 2011; Levy and Daumé III, 2011).

For this reason, we take another approach, which was employed by Dediu (2010). He used language families established by historical linguists. Because such tree topologies are not associated with dates, he inferred the dates of internal nodes together with the states of internal nodes and TRMs. This was possible because he jointly fitted a sequence of traits, instead of fitting each trait independently. If multiple traits are combined, they provide considerable information on a branch length, or the time elapsing from a parent to a child, because the elapsed time is roughly inversely proportional to the similarity between the two nodes.[5]

Our approach differs from Dediu's mainly in two points. First, whereas Dediu (2010) performed posterior inference separately for each language family, we tie a single set of TRMs to all available language families. Second, Dediu (2010) only inferred relative dates because he did not perform *calibration* (Drummond and Bouckaert, 2015). In order to assign calendar dates to nodes, we use multiple calibration points (the clock in Figure 2 indicates a calibration point). As is com-

---

[3] See Pereltsvaig and Lewis (2015) for a criticism of computational approaches to historical linguistics and Chang et al. (2015) for an elegant solution to a set of problems commonly found in inferred time-trees.

[4] To be precise, a set of tree *samples* given by MCMC sampling is usually employed to account for uncertainty.

[5] Although some previous studies adopted relaxed clock models, in which different branches have different rates of evolution (Drummond and Bouckaert, 2015), we use the simple strict clock model because our calibration points are not large enough in number to harness the very flexible models.

monly done in the cognate-based reconstruction of a time-tree (Bouckaert et al., 2012), we set the Gaussian, Gaussian mixture, log-normal and uniform distributions as priors on the dates of the corresponding internal nodes.

2.2 Latent Representations of Languages

While previous studies analyzed the evolution of a single categorical feature (Dediu, 2010; Greenhill et al., 2010; Maurits and Griffiths, 2014) or a pair of binary features (Dunn et al., 2011), we capture the dependencies of all available features by mapping each language to a sequence of independent latent variables. To our knowledge, Murawaki (2015) was the first to introduce latent representations to typological features. Pointing out several critical problems, however, Murawaki (2017) superseded the earlier model. The present study is built on top of a slightly modified version of the Bayesian model presented by Murawaki (2017).

Like the present study, Murawaki (2015) performed phylogenetic inference on the latent space. However, since this model lacks the notion of time, it does not have descriptive power beyond clustering. Borrowing statistical models from the field of evolutionary biology, we perform time-aware inference.

3 Proposed Method

3.1 Latent Representations of Languages

Central to our framework of diachronic analysis are the latent representations of languages (Murawaki, 2017). Each language l is represented as a sequence of N discrete features $\mathrm{x}_{l,*} = (x_{l,1}, \cdots, x_{l,N}) \in \mathbb{N}_0^N$. $x_{l,n}$ can take a binary value ($x_{l,n} \in \{0,1\}$) or categorical value ($x_{l,n} \in \{1, 2, \cdots, F_n\}$, where F_n is the number of distinct values). We assume that $\mathrm{x}_{l,*}$ is stochastically generated from its latent representation, $\mathrm{z}_{l,*} = (z_{l,1}, \cdots, z_{l,K}) \in \{0,1\}^K$, where K is the number of binary parameters, which is given a priori.

Dependencies between surface features are captured by weight matrix $W \in \mathbb{R}^{K \times M}$. M will be described below. In the generative story, we first calculate feature score vector $\tilde{\theta}_{l,*} = (\mathrm{z}_{l,*}^{\mathrm{T}} W)^{\mathrm{T}} \in \mathbb{R}^M$. We then obtain model parameter vector $\theta_{l,*} \in (0,1)^M$ by normalizing $\tilde{\theta}_{l,*}$ for each feature type n. We use the sigmoid function for binary features,

$$\theta_{l,f(n,1)} = \frac{1}{1 + \exp(-\tilde{\theta}_{l,f(n,1)})}, \tag{1}$$

and the softmax function for categorical features,

$$\theta_{l,f(n,i)} = \frac{\exp(\tilde{\theta}_{l,f(n,i)})}{\sum_{i'=1}^{F_n} \exp(\tilde{\theta}_{l,f(n,i')})}. \tag{2}$$

Note that while a binary feature corresponds to one model parameter, categorical feature n is tied to F_n model parameters. We use function $f(n,i) \in \{1, \cdots, m, \cdots, M\}$ to map feature n to the corresponding model parameter index. Finally, we draw a binary feature from $\mathrm{Bernoulli}(\theta_{l,f(n,1)})$, and a categorical feature from $\mathrm{Categorical}(\theta_{l,f(n,1)}, \cdots, \theta_{l,f(n,F_n)})$.

To gain an insight into how W captures inter-feature dependencies, suppose that for parameter k, a certain group of languages take $z_{l,k} = 1$. If two categorical feature values (n_1, i_1) and (n_2, i_2) have large positive weights (i.e., $w_{k,f(n_1,i_1)} > 0$ and $w_{k,f(n_2,i_2)} > 0$), then the pair must often co-occur in these languages because W raises both $\theta_{l,f(n_1,i_1)}$ and $\theta_{l,f(n_2,i_2)}$. Likewise, the fact that two feature values do not co-occur can be encoded as a positive weight for one value and a negative weight for the other.

The remaining question is how $z_{l,k}$ is generated. We draw $\mathrm{z}_{*,k} = (z_{1,k}, \cdots, z_{L,k})$ from an autologistic model (Besag, 1974) that incorporates the observation that phylogenetically or areally close languages tend to take the same value.

To complete the generative story, let X and Z be the matrices of languages in the surface and latent representations, respectively, and let A be a set of latent variables controlling K autologistic models. The joint distribution is defined as

$$P(A, Z, W, X) = P(A)P(Z|A)P(W)P(X|Z, W),$$

where hyperparameters are omitted for brevity. For prior probabilities $P(A)$ and $P(W)$, please refer to Murawaki (2017).

Even if less than 30% of the items of X are present, this model has been demonstrated to recover missing values reasonably well. Also, when plotted on a world map, some parameters appear to retain phylogenetic and areal signals observed for surface features, indicating that they are not mere statistical artifacts (Murawaki, 2017).

3.2 Transition Rate Matrices (TRMs)

We assume that each parameter k independently evolves along the branches of trees according to a continuous-time Markov chain (CTMC) model (Drummond and Bouckaert, 2015). The CTMC is a continuous extension to the more familiar discrete-time Markov chain. It is is controlled by a TRM Q_k. If the number of states (possible values) is 2, then Q_k is a 2×2 matrix:

$$Q_k = \begin{pmatrix} -\alpha_k & \alpha_k \\ \beta_k & -\beta_k \end{pmatrix}.$$

We set Gamma priors on $\alpha_k, \beta_k > 0$.

Q_k can be used to calculate the *transition probability*, or the probability of language l taking value b for parameter k conditioned on l's parent $\pi(l)$ and t, the time span between the two:

$$P(z_{l,k} = b | z_{\pi(l),k} = a, t) = \exp(tQ_k)_{a,b}. \quad (3)$$

The matrix exponential $\exp(tQ_k)$ can be solved analytically if Q_k is a 2×2 matrix:

$$\exp(tQ_k) = \begin{pmatrix} \frac{\beta_k + \alpha_k e^{-(\alpha_k+\beta_k)t}}{\alpha_k+\beta_k} & \frac{\alpha_k - \alpha_k e^{-(\alpha_k+\beta_k)t}}{\alpha_k+\beta_k} \\ \frac{\beta_k - \beta_k e^{-(\alpha_k+\beta_k)t}}{\alpha_k+\beta_k} & \frac{\alpha_k + \beta_k e^{-(\alpha_k+\beta_k)t}}{\alpha_k+\beta_k} \end{pmatrix}.$$

As t approaches to infinity, we obtain the *stationary probability* $(\frac{\beta_k}{\alpha_k+\beta_k}, \frac{\alpha_k}{\alpha_k+\beta_k})^{\mathrm{T}}$. We can see that α_k and β_k control both the speed of change (the larger the higher) and the stationary distribution.

A root node has no parent by definition. We draw the state of a root node from the stationary distribution. Thus, language isolates do have impact on posterior inference of TRMs.

3.3 Posterior Inference of Time-trees

To estimate TRMs, we need to specify the generative model of time-trees and an inference algorithm. In the generative story, each tree topology is drawn from some uniform distribution. The dates of its nodes are determined next. If the node in question is not a calibration point, its date is drawn from some uniform distribution, subject to the ancestral ordering constraint: a node must be older than its descendants. If the node is a calibration point, its date is drawn from the corresponding prior distribution.[6] TRM parameters, α_k and β_k, are generated from Gamma priors. For the root node, the value of parameter k is drawn from the corresponding stationary distribution. The states of the non-root nodes are generated using Eq. (3).

Given tree topologies, the states of the leaf nodes and calibration points, we need to infer (1) the dates of the internal nodes, (2) the states of the internal nodes, and (3) TRM parameters, α_k and β_k, for each latent parameter k. Gibbs sampling updates of these variables are as follows:

Update dates We update the dates of the internal nodes one by one. The time span in which the target node can move is bound by its parent (if there is) and its eldest child. We use slice sampling (Neal, 2003) to update the date. In addition, we use a Metropolis-Hastings operator that multiplies the dates of all the internal nodes of a tree by a rate drawn from a log-normal distribution.

Update states For each parameter k, we block-sample a whole given tree. Specifically, we implement a Bayesian version of Felsenstein's tree-pruning algorithm, which is akin to the forward filtering-backward sampling algorithm for Bayesian hidden Markov models.

Update α_k and β_k We jointly sample α_k and β_k for each k. Since both the transition and stationary probabilities can be obtained analytically for binary traits, we use Hamiltonian Monte Carlo to exploit gradient information (Neal, 2011).

3.4 Three-Step Analysis

Now we are ready to elaborate on the proposed framework of diachronic analysis (Figure 2).

Step 1 We map each language, represented as a sequence of N discrete surface features, to a sequence of K binary latent parameters. Let feature matrix X be decomposed into observed and missing portions, X_{obs} and X_{mis}, respectively. Given X_{obs}, we use Gibbs sampling to infer A, parameter matrix Z, weight matrix W, and X_{mis} (Murawaki, 2017).[7]

In the present study, we set $K = 100$. We run 5 independent MCMC chains. For each chain, we start with 1,000 burn-in iterations. We then obtain 10 samples with an interval of 10 iterations. Note that after burn-in iterations, we fix W and only sample A, Z, and X_{mis} to avoid the identifiability problems (e.g., label-switching). For each item of Z and X_{mis}, we output the most frequent value among the 10 samples. We do this to reduce uncertainty.

---

[6] This model is slightly leaky because some priors (e.g., Gaussian) assign non-zero probabilities to illogical time-trees that violate the ancestral ordering constraint.

[7] We employ a slightly modified Metropolis-Hastings operator to improve the mobility of Z.

Step 2 We fit a set of K TRMs on family trees around the world. Formally, what are observed are tree topologies, the states of the leaf nodes (i.e., sequences of latent parameters), and multiple calibration points. Given these, we infer TRM parameters, α_k and β_k, for each latent parameter k, as well as the dates and states of the internal nodes. We, again, use Gibbs sampling as explained in Section 3.3.

We collect 10 samples with an interval of 10 iterations after 1,000 burn-in iterations. We do this for each of the 5 samples obtained in Step 1. As a result, we obtain 50 samples in total.

Step 3 We analyze the TRMs by simulating language evolution. Given the latent representation of language l, we stochastically generate its descendant l' after some time span t. Specifically, we draw $z_{l',k}$ according to the transition probability of Eq. (3) for each parameter k. Using weight matrix W, we then project the latent representation $\mathrm{z}_{l',*}$ back to the surface representation $\mathrm{x}_{l',*}$. To be precise, we use model parameter vector $\theta_{l',*}$, instead of $\mathrm{x}_{l',*}$, for further analysis. For each of the 50 samples obtained in Step 2, we simulate the evolution of a given language 100 times (5,000 samples in total).

4 Experiments

4.1 Data and Preprocessing

The database of typological features we used is the online edition[8] of the *World Atlas of Language Structures* (WALS) (Haspelmath et al., 2005). We preprocessed the database as was done in Murawaki (2017), with different thresholds. As a result, we obtained a language–feature matrix consisting of $L = 2{,}607$ languages and $N = 152$ features. Only 19.98% of items in the matrix were present. We manually classified features into binary and categorical ones. The number of model parameters, M, was 760.

We used Glottolog 3.2 (Hammarström et al., 2018) as the source of family trees. Glottolog has three advantages over Ethnologue (Lewis et al., 2014), another commonly-used catalog of the world's languages. (1) Glottolog makes explicit that it adopts a genealogical classification, rather than hierarchical clusterings of modern languages. (2) It reflects more recent research. (3) Mapping between Glottolog and WALS is easy because WALS provides Glottolog's language codes (glottocodes) when available.

After Step 1 of Section 3.4, we dropped languages from WALS that could not be mapped to Glottolog. As a result, 2,557 languages remained. We subdivided a Glottolog node if multiple languages from WALS shared the same glottocode. We removed leaf nodes that were not present in WALS and repeatedly dropped internal nodes that had only one child. We obtained 309 language families among which 154 had only one node (i.e., language isolates).

We collected 50 calibration points from secondary literature (Holman et al., 2011; Bouckaert et al., 2012; Gray et al., 2009; Maurits and Griffiths, 2014; Grollemund et al., 2015). For example, we set a Gaussian prior with mean 2,500 BP (before present) and standard deviation 500 on the date of (Proto-)Hmong-Mien. See Table S.1 of the supplementary materials for details. Our calibration points are by no means definitive or exhaustive but should be seen as a first step toward world-scale dating.

4.2 Case Study: Basic Word Order (BWO)

As a proof-of-concept demonstration of the proposed framework, we investigate the BWO feature, or WALS's Feature 81A (Dryer, 2013b). The cross-linguistic variation of BWO attracts attention not only from typologists but from psycholinguists (see Maurits and Griffiths (2014) for a brief review). Some claim that the fact that `SOV` is the most frequent order indicates its optimality, presumably in terms of functionality. Some others point to an apparent historical trend of `SOV` changing to `SVO`, but not vice versa (Gell-Mann and Ruhlen, 2011), which might imply (1) the functional superiority of `SVO` over `SOV` and (2) an even higher prevalence of `SOV` in the past. `SOV` preferences in emerging sign languages (Sandler et al., 2005) and in elicited pantomime (Goldin-Meadow et al., 2008) are also reported.

Maurits and Griffiths (2014) fitted a 6 × 6 TRM[9] on large language families. However, we suspect that singling out BWO is oversimplification. Given its profound effect on the whole grammatical system, a BWO change can hardly occur independently of other features. In fact, Mithun (1995) lists a variety of morphological factors that have

---

[8] `http://wals.info/`

[9] The special value `No dominant order` was removed in their experiments.

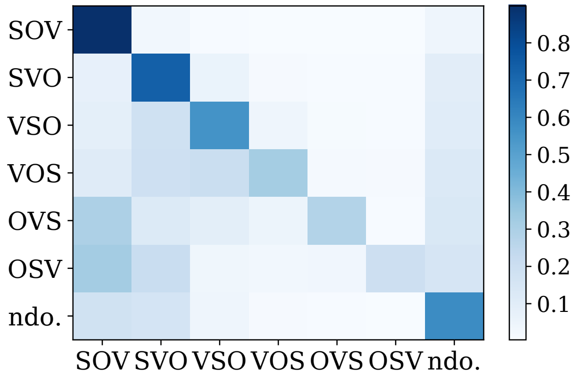

Figure 3: Transition probability of BWO. An item of the matrix (a, b) indicates how probable a language with word order a will take word order b after 2,000 years. Note that this covers the scenario in which the language switches to another word order c before changing to b.

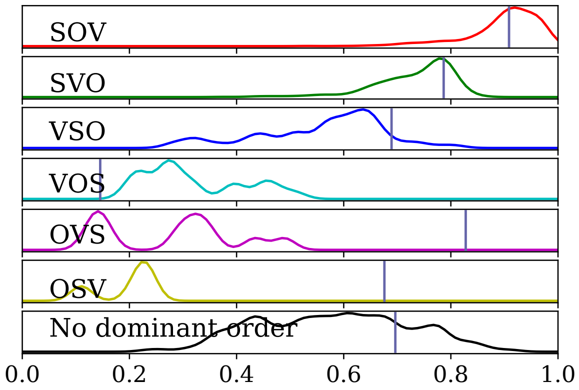

Figure 4: Variability in the probability of keeping the same word order. For word order i, the x-axis indicates the average of $\theta_{l',f(n,i)}$, the probability of taking word order i after 2,000 years. The y-axis indicates the relative frequency of the corresponding probability among the languages with word order i. Kernel density estimation is used for smoothing. The pulse in each box shows the corresponding probability estimated by directly fitting the 7 × 7 TRM of the surface feature.

diachronically reduced the rigidity of `SOV` order in Native American languages. Her analysis suggests that languages sharing the same word order might not be a monolithic group.

Here, we use latent representation-based analysis to answer questions: how variable language sharing the same BWO are with respect to diachronic stability, and what kind of features are correlated with BWO stability?

4.3 Variability of Diachronic Stability

Among the 2,557 modern languages, we chose 1,357 languages for which the BWO feature was present. We simulated evolution with $t = 2{,}000$, as described in Section 3.4. Let n be the index of the BWO feature. For the 5,000 samples of each simulated language l', we averaged the BWO probability vectors, $\theta_{l',f(n,1)}, \cdots, \theta_{l',f(n,F_n)}$.

Before going into inter-language variability, let us take a look at the overall trend. We took the average of the BWO probability vectors for each word order. The result is shown in Figure 3. Our findings largely agree with those of Maurits and Griffiths (2014): (1) `SOV` is the most diachronically stable word order, which is followed by `SVO`, (2) `SOV` prefers changing to `SVO` over `VSO` (although hardly visually recognizable), and (3) `VSO` is more likely to change to `SVO` than to `SOV`, just to name a few.

Next, the variability is visualized in Figure 4. We can see that languages sharing the same word order differ considerably in terms of diachronic stability. For comparison, we fitted the 7 × 7 TRM of the BWO feature on the samples of time-trees obtained in Step 2 of Section 3.4. For `SOV` and `SVO`, the probabilities based on the surface feature pointed to the modal probabilities based on the latent representations. This is somewhat surprising because we anticipated that the combination of the stochastic surface-to-latent and latent-to-surface mappings would amplify uncertainty of estimation.

The two least common word orders, `OVS` (0.8%) and `OSV` (0.3%) exhibited huge gaps between the two types of probabilities. The probabilities based on the surface feature were consistently larger (i.e., more stable). Surface feature-based estimation had no other way to explain the presence of these uncommon word orders than slowing down the convergence to the stationary distribution (otherwise they go extinct). Maurits and Griffiths (2014) also reported some counter-intuitive results regarding `OVS` and `OSV`. By contrast, latent parameter-based estimation appears to have explained the low frequencies partly with the stochasticity of observation associated with the latent-to-surface mapping.

Now we attempt to explain the variability of diachronic stability. Although we have all model parameters in hand, it is not easy to manually ana-

Weight	Explanatory variable (feature: value)
0.01527	59A Possessive Classification: More than five classes
0.01297	90A Order of Relative Clause and Noun: Relative clause-Noun
0.01138	85A Order of Adposition and Noun Phrase: Postpositions
0.00998	94A Order of Adverbial Subordinator and Clause: Final subordinator word
0.00889	51A Position of Case Affixes: Case suffixes
−0.02507	93A Position of Interrogative Phrases in Content Questions: Initial interrogative phrase
−0.02576	26A Prefixing vs. Suffixing in Inflectional Morphology: Weakly prefixing
−0.02738	143E Preverbal Negative Morphemes: NegV
−0.03169	85A Order of Adposition and Noun Phrase: Inpositions
−0.08360	85A Order of Adposition and Noun Phrase: No dominant order

Table 2: Regression analysis of variability for `SOV`. 10 among 94 variables are shown.

Weight	Explanatory variable (feature: value)
0.01797	4A Voicing in Plosives and Fricatives: In both plosives and fricatives
0.01403	33A Coding of Nominal Plurality: Plural prefix
0.01095	85A Order of Adposition and Noun Phrase: Prepositions
0.00995	92A Position of Polar Question Particles: Final
0.00856	85A Order of Adposition and Noun Phrase: No adpositions
0.00670	26A Prefixing vs. Suffixing in Inflectional Morphology: Strong prefixing
−0.01401	87A Order of Adjective and Noun: No dominant order
−0.01440	87A Order of Adjective and Noun: Adjective-Noun
−0.01974	57A Position of Pronominal Possessive Affixes: Possessive suffixes
−0.03226	51A Position of Case Affixes: Case suffixes

Table 3: Regression analysis of variability for `SVO`. 10 among 52 variables are shown.

Prob.	Language	Family	Aff.
0.854	Fyam	Atlantic-Congo	`WS`
0.828	Younuo Bunu	Hmong-Mien	`LA`
0.825	Czech	Indo-European	`WS`
0.825	Tetum	Austronesian	`LA`
0.824	Stieng	Austroasiatic	`LA`
0.824	Alune	Austronesian	`EQ`
0.822	Berom	Atlantic-Congo	`SP`
0.822	Paulohi	Austronesian	`EQ`
0.821	South-Central Kikongo	Atlantic-Congo	`SP`
0.820	Abun	(isolate)	`LA`

Table 4: Some of the most stable `SVO` languages with the values of the affixation feature (Dryer, 2013c). The first column indicates the probabilities of keeping the `SVO` order after 2,000 years. Names of languages and language families are taken from Glottolog. The values of the affixation feature are `EQ` (Equal prefixing and suffixing), `LA` (Little affixation), `SP` (Strong prefixing), and `WS` (Weakly suffixing).

lyze their complex dependencies. The approach we adopt in the present study is to let a simpler model explain the model's complex behavior. Specifically, we used linear regression with L1 regularization (i.e., lasso). The hyperparameter was tuned using 3-fold cross-validation. For each word order i, the target variable was the average of $\theta_{l',f(n,i)}$ while explanatory variables were the current surface features, $x_{l,1}, \cdots, x_{l,N}$. For better interpretability, we excluded from explanatory variables surface features that trivially depended on the BWO feature (Takamura et al., 2016). Note that missing values were imputed in Step 1 of Section 3.4.

Tables 2 and 3 show the results of regression analysis for `SOV` and `SVO` languages, respectively. As expected, feature values typically associated with the specified word order had positive weights while negative weights indicate inconsistency. A stable `SOV` language may use prenominal relative clauses, postpositions and/or case suffixes. The trend was less clear for `SVO` languages, but those characterized by heavy use of prefixes were stable too. Interestingly, Feature 85A (Order of Adposition and Noun Phrase) had two positively weighted values: `Prepositions` and `No adpositions`. We speculate that `SVO` order is suitable for analytic languages that rely heavily on word ordering to encode syntactic structure (e.g., English and languages of Mainland Southeast Asia) but is not necessarily so for languages with rich morphological devices for marking syntactic structure so that word ordering can relatively freely convey information structure.

4.4 Language-Specific Analysis

In Section 4.3, we suggested that stable `SVO` languages do not form a coherent group but can be grouped into at least two clusters. This can be confirmed in Table 4, where most of the most stable `SVO` languages exhibit either (1) little affixation or (2) strong prefixation. To analyze these languages in detail, we performed language-specific simulations. We chose Tetum and South-Central Kikongo as the examples of analytic and strongly prefixing languages, respectively.

Figure 5 shows the word order probabilities of

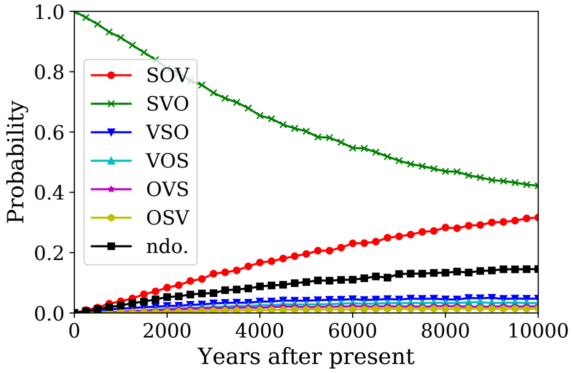

Figure 5: Simulated evolution of Tetum BWO.

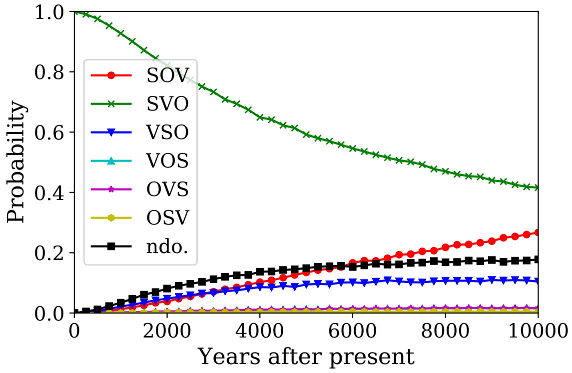

Figure 6: Simulated evolution of South-Central Kikongo BWO.

Tetum as a function of time. For each time t, we performed simulation 500 times for each of the 50 samples and took the average of the BWO probability vectors, $\theta_{l',f(n,1)}, \cdots, \theta_{l',f(n,F_n)}$. According to our analysis, Tetum will remain `SVO` with a probability of 81.1% at $t = 2{,}000$. `SVO` was followed by `SOV` (8.4%) and `No dominant order` (5.3%).

What will the Austronesian language of East Timor look like in the future, if it switches to `SOV`? To answer this question, we performed regression analysis again with $t = 2{,}000$. For each word order i, the target variable was $\theta_{l',f(n,i)}$ of each sample of simulated language l' whereas explanatory variables were the items of the probability vector $\theta_{l',*}$. In other words, we aimed at finding out features that were characteristic of the specified word order. As before, we removed surface features with trivial dependencies on the BWO feature (Takamura et al., 2016) as well as the BWO feature itself.

Table 5 shows the result of regression analysis. If the relatively analytic language switches to `SOV`, Tetum will be characterized by a holistic reconfiguration. It is likely to develop suffixes and to replace prepositions with postpositions. South-Central Kikongo is analyzed in the same manner, as shown in Figure 6 and Table 6. The Bantu language of Africa is markedly different from Tetum as it is characterized by a higher tendency to switch to `No dominant order`.

Weight	Explanatory variable (feature: value)
0.1256	85A Order of Adposition and Noun Phrase: Postpositions
0.1086	16A Weight Factors in Weight-Sensitive Stress Systems: Coda consonant
0.0687	69A Position of Tense-Aspect Affixes: Tense-aspect suffixes
0.0552	35A Plurality in Independent Personal Pronouns: Number-indifferent pronouns
0.0518	2A Vowel Quality Inventories: Large (7-14)
0.0506	122A Relativization on Subjects: Non-reduction

Table 5: Regression analysis of the Tetum changing to `SOV` order. Top 6 out of 262 variables.

Weight	Explanatory variable (feature: value)
0.1048	15A Weight-Sensitive Stress: Left-oriented: One of the first three
0.0940	85A Order of Adposition and Noun Phrase: Postpositions
0.0821	16A Weight Factors in Weight-Sensitive Stress Systems: Coda consonant
0.0715	7A Glottalized Consonants: Ejectives, implosives, and glottalized resonants
0.0661	100A Alignment of Verbal Person Marking: Accusative
0.0636	64A Nominal and Verbal Conjunction: Both expressed by juxtaposition

Table 6: Regression analysis of the South-Central Kikongo changing to `SOV` order. Top 6 out of 408 variables.

5 Conclusion

In this paper, we presented a new framework of latent representation-based analysis of diachronic typology, which enables us to investigate correlated evolution of multiple surface features in an exploratory manner. We focused on the order of subject, object and verb as a proof-of-concept demonstration, but investigating other features would be fruitful too. We analyzed the estimated model parameters with simulation experiments. In the future, we would like to investigate the inferred trees in detail.[10] The source code is publicly available at `https://github.com/murawaki/lattyp`.

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Number 18K18104.

---

[10] A preliminary analysis is presented in Section S.2 of the supplementary materials.

References

Julian Besag. 1974. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236.

Balthasar Bickel. 2013. Distributional biases in language families. In David A. Peterson and Alan Timberlake, editors, *Language Typology and Historical Contingency*, pages 415–444. John Benjamins.

Remco Bouckaert, Philippe Lemey, Michael Dunn, Simon J. Greenhill, Alexander V. Alekseyenko, Alexei J. Drummond, Russell D. Gray, Marc A. Suchard, and Quentin D. Atkinson. 2012. Mapping the origins and expansion of the Indo-European language family. *Science*, 337(6097):957–960.

Will Chang, Chundra Cathcart, David Hall, and Andrew Garrett. 2015. Ancestry-constrained phylogenetic analysis supports the Indo-European steppe hypothesis. *Language*, 91(1):194–244.

William Croft, Tanmoy Bhattacharya, Dave Kleinschmidt, D. Eric Smith, and T. Florian Jaeger. 2011. Greenbergian universals, diachrony, and statistical analyses. *Linguistic Typology*, 15(2):433–453.

Hal Daumé III. 2009. Non-parametric Bayesian areal linguistics. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 593–601.

Hal Daumé III and Lyle Campbell. 2007. A Bayesian model for discovering typological implications. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 65–72.

Dan Dediu. 2010. A Bayesian phylogenetic approach to estimating the stability of linguistic features and the genetic biasing of tone. *Proceedings of the Royal Society of London B: Biological Sciences*, 278(1704):474–479.

Alexei J. Drummond and Remco R. Bouckaert. 2015. *Bayesian Evolutionary Analysis with BEAST*. Cambridge University Press.

Matthew S. Dryer. 1997. On the six-way word order typology. *Studies in Language. International Journal sponsored by the Foundation "Foundations of Language"*, 21(1):69–103.

Matthew S. Dryer. 2011. The evidence for word order correlations: a response to Dunn, Greenhill, Levinson and Gray's paper in Nature. *Linguistic Typology*, 15:335–380.

Matthew S. Dryer. 2013a. On the six-way word order typology, again. *Studies in Language. International Journal sponsored by the Foundation "Foundations of Language"*, 37(2):267–301.

Matthew S. Dryer. 2013b. Order of subject, object and verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology.

Matthew S. Dryer. 2013c. Prefixing vs. suffixing in inflectional morphology. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology.

Michael Dunn, Simon J. Greenhill, Stephen C. Levinson, and Russell D. Gray. 2011. Evolved structure of language shows lineage-specific trends in word-order universals. *Nature*, 473(7345):79–82.

Isidore Dyen, Joseph B. Kruskal, and Paul Black. 1992. An Indoeuropean classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5):1–132.

Joseph Felsenstein. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376.

Murray Gell-Mann and Merritt Ruhlen. 2011. The origin and evolution of word order. *Proceedings of the National Academy of Sciences*, 108(42):17290–17295.

Susan Goldin-Meadow, Wing Chee So, Aslı Özyürek, and Carolyn Mylander. 2008. The natural order of events: How speakers of different languages represent events nonverbally. *Proceedings of the National Academy of Sciences*, 105(27):9163–9168.

Russell D. Gray and Quentin D. Atkinson. 2003. Language-tree divergence times support the Anatolian theory of Indo-European origin. *Nature*, 426(6965):435–439.

Russell D. Gray, Alexei J. Drummond, and Simon J. Greenhill. 2009. Language phylogenies reveal expansion pulses and pauses in Pacific settlement. *science*, 323(5913):479–483.

Joseph H. Greenberg. 1963. Some universals of grammar with particular reference to the order of meaningful elements. In Joseph H. Greenberg, editor, *Universals of Language*, pages 73–113. MIT Press.

Joseph H. Greenberg. 1978. Diachrony, synchrony and language universals. In Joseph H. Greenberg, Charles A. Ferguson, and Edith A. Moravesik, editors, *Universals of Human Language*, volume 1. Stanford University Press.

Simon J. Greenhill, Quentin D. Atkinson, Andrew Meade, and Russel D. Gray. 2010. The shape and tempo of language evolution. *Proceedings of the Royal Society B: Biological Sciences*, 277(1693):2443–2450.

Simon J. Greenhill, Robert Blust, and Russell D. Gray. 2008. The Austronesian basic vocabulary database: From bioinformatics to lexomics. *Evolutionary Bioinformatics*, 4:271–283.

Simon J. Greenhill, Chieh-Hsi Wu, Xia Hua, Michael Dunn, Stephen C. Levinson, and Russell D. Gray. 2017. Evolutionary dynamics of language systems. *Proceedings of the National Academy of Sciences*, 114(42):E8822–E8829.

Rebecca Grollemund, Simon Branford, Koen Bostoen, Andrew Meade, Chris Venditti, and Mark Pagel. 2015. Bantu expansion shows that habitat alters the route and pace of human dispersals. *Proceedings of the National Academy of Sciences*, 112(43):13296–13301.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank, editors. 2018. *Glottolog 3.2*. Max Planck Institute for the Science of Human History.

Martin Haspelmath, Matthew Dryer, David Gil, and Bernard Comrie, editors. 2005. *The World Atlas of Language Structures*. Oxford University Press.

Eric W. Holman, Cecil H. Brown, Søren Wichmann, André Müller, Viveka Velupillai, Harald Hammarström, Sebastian Sauppe, Hagen Jung, Dik Bakker, Pamela Brown, Oleg Belyaev, Matthias Urban, Robert Mailhammer, Johann-Mattis List, and Dmitry Egorov. 2011. Automated dating of the world's language families based on lexical similarity. *Current Anthropology*, 52(6):841–875.

Yoshiaki Itoh and Sumie Ueda. 2004. The Ising model for changes in word ordering rules in natural languages. *Physica D: Nonlinear Phenomena*, 198(3):333–339.

Roger Levy and Hal Daumé III. 2011. Computational methods are invaluable for typology, but the models must match the questions. *Linguistic Typology*, 15:393–399.

M. Paul Lewis, Gary F. Simons, and Charles D. Fennig, editors. 2014. *Ethnologue: Languages of the World, 17th Edition*. SIL International. Online version: http://www.ethnologue.com.

Elena Maslova. 2000. A dynamic approach to the verification of distributional universals. *Linguistic Typology*, 4(3):307–333.

Luke Maurits and Thomas L. Griffiths. 2014. Tracing the roots of syntax with Bayesian phylogenetics. *Proceedings of the National Academy of Sciences*, 111(37):13576–13581.

Marianne Mithun. 1995. Morphological and prosodic forces shaping word order. In Pamela A. Downing and Michael Noonan, editors, *Word Order in Discourse*, pages 387–423. John Benjamins Publishing.

Yugo Murawaki. 2015. Continuous space representations of linguistic typology and their application to phylogenetic inference. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 324–334.

Yugo Murawaki. 2016. Statistical modeling of creole genesis. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1329–1339.

Yugo Murawaki. 2017. Diachrony-aware induction of binary latent representations from typological features. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 451–461. Asian Federation of Natural Language Processing.

Yugo Murawaki and Kenji Yamauchi. 2018. A statistical model for the joint inference of vertical stability and horizontal diffusibility of typological features. *Journal of Language Evolution*, 3(1):13–25.

Luay Nakhleh, Don Ringe, and Tandy Warnow. 2005. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. *Language*, pages 382–420.

Radford M. Neal. 2003. Slice sampling. *Annals of Statistics*, 31(3):705–767.

Radford M. Neal. 2011. MCMC using Hamiltonian dynamics. In Steve Brooks, Andrew Gelman, Galin L. Jones, and Xiao-Li Meng, editors, *Handbook of Markov Chain Monte Carlo*, pages 113–162. CRC Press.

Shijulal Nelson-Sathi, Johann-Mattis List, Hans Geisler, Heiner Fangerau, Russell D. Gray, William Martin, and Tal Dagan. 2010. Networks uncover hidden lexical borrowing in Indo-European language evolution. *Proceedings of the Royal Society B: Biological Sciences*.

Johanna Nichols. 1992. *Linguistic Diversity in Space and Time*. University of Chicago Press.

Mark Pagel and Andrew Meade. 2006. Bayesian analysis of correlated evolution of discrete characters by reversible-jump Markov chain Monte Carlo. *The American Naturalist*, 167(6):808–825.

Asya Pereltsvaig and Martin W. Lewis. 2015. *The Indo-European Controversy: Facts and Fallacies in Historical Linguistics*. Cambridge University Press.

Wendy Sandler, Irit Meir, Carol Padden, and Mark Aronoff. 2005. The emergence of grammar: Systematic structure in a new language. *Proceedings of the National Academy of Sciences*, 102(7):2661–2665.

Hiroya Takamura, Ryo Nagata, and Yoshifumi Kawasaki. 2016. Discriminative analysis of linguistic features for typological study. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 69–76.

Capturing Regional Variation with Distributed Place Representations and Geographic Retrofitting

Dirk Hovy
Bocconi University
`dirk.hovy@unibocconi.it`

Christoph Purschke
University of Luxembourg
`christoph.purschke@uni.lu`

Abstract

Dialects are one of the main drivers of language variation, a major challenge for natural language processing tools. In most languages, dialects exist along a continuum, and are commonly discretized by combining the extent of several preselected linguistic variables. However, the selection of these variables is theory-driven and itself insensitive to change. We use Doc2Vec on a corpus of 16.8M anonymous online posts in the German-speaking area to learn continuous document representations of cities. These representations capture continuous regional linguistic distinctions, and can serve as input to downstream NLP tasks sensitive to regional variation. By incorporating geographic information via retrofitting and agglomerative clustering with structure, we recover dialect areas at various levels of granularity. Evaluating these clusters against an existing dialect map, we achieve a match of up to 0.77 V-score (harmonic mean of cluster completeness and homogeneity). Our results show that representation learning with retrofitting offers a robust general method to automatically expose dialectal differences and regional variation at a finer granularity than was previously possible.

1 Introduction

People actively use dialects to mark their regional origin (Shoemark et al., 2017a,b), making them one of the main drivers of language variation. Accounting for this variation is a challenge for NLP systems (see for example the failed attempts of people with accents trying to use dialogue systems. Accounting for variation can significantly improve performance in machine translation (Mirkin and Meunier, 2015; Östling and Tiedemann, 2017), geolocation (Rahimi et al., 2017a,b) and help personalize applications and search.

However, regional variation involves a complex set of grammatical, lexical, and phonological features, all of them continuously changing. Consequently, dialects are not static discrete entities, but exist along a continuum in most languages. Variational linguistics and dialectology typically discretize this continuum by using a set of preselected features (Trudgill, 2000), often including outdated vocabulary. The resulting dialect areas are highly accurate, but extremely time-consuming to construct and inflexible (the largest and to date most comprehensive evaluation of German dialects, the Wenker-Atlas (Rabanus et al., 2010) is almost 150 years old and took decades to complete). Work in dialectometry has shown that computational methods, such as clustering (Nerbonne and Heeringa, 1997; Prokić and Nerbonne, 2008; Szmrecsanyi, 2008, inter alia) and dimensionality reduction (Nerbonne et al., 1999; Shackleton Jr, 2005) can instead be used to identify dimensions of variation in manually constructed discrete feature vectors. However, the success of such approaches depends on precise prior knowledge of variation features (Lameli, 2013).

Distributed representations, as unsupervised methods, can complement these methods by capturing similarities between words and documents (here: cities) along various latent dimensions, including syntactic, semantic, and pragmatic aspects. These representations are therefore more compact, less susceptible to data sparsity than latent variable models, and allow us to represent a large number of possible clusters than feature-based representations (cf. Luong et al. (2013)). These properties also allow us to measure similarities on a continuous scale, which makes represen-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4383–4394
Brussels, Belgium, October 31 - November 4, 2018.

tation learning especially useful for the study of regional language variation along several linguistic dimensions.

We use a corpus of 16.8 million anonymous German online posts, cast cities as document labels, and induce document embeddings for these cities via Doc2Vec (Le and Mikolov, 2014). We first show that the resulting city embeddings capture regional linguistic variation at a more fine-grained, continuous regional distinction than previous approaches (Bamman et al., 2014; Östling and Tiedemann, 2017), which operated at a state or language level.[1] We also show that the embeddings can serve as input to a geolocation task, outperforming a bag-of-words model, and producing competitive results.

However, such representations are susceptible to linguistic data bias, ignore geographic factors, and are hard to evaluate with respect to their fit with existing linguistic distinctions. We address these problems by including geographic information via retrofitting (Faruqui et al., 2015; Hovy and Fornaciari, 2018): we use administrative region boundaries to modify the city embeddings, and evaluate the resulting vectors in a clustering approach to discover larger dialect regions.

In contrast to most dialectometric approaches (Nerbonne et al., 1999; Prokić and Nerbonne, 2008), and in line with common NLP practice (Doyle, 2014; Grieve, 2016; Huang et al., 2016; Rahimi et al., 2017a), we also evaluate the clustered dialect areas quantitatively. Rather than testing the geographic extent of individual words against known dialect areas (Doyle, 2014), we compare the match of entire geographic regions to a recent German dialect map (Lameli, 2013). We use cluster evaluation metrics to measure how well our clusters match the known dialect regions.

The results show that our method automatically captures existing (manually determined) dialect distinctions well, and even goes beyond them in that it also allows for a more fine-grained qualitative analysis. Our research shows that representation learning is well suited to the study of language variation, and demonstrates the potential of incorporating non-linguistic information via retrofitting. For an application of our methodology to a larger Twitter data set over multiple languages, see (Hovy et al., In Preparation).

Contributions In this paper, we make the following contributions to linguistic insights, performance improvements, and algorithmic contributions. We show:

1. how Doc2Vec can be used to learn distributed representations of cities that capture continuous regional linguistic variation. The approach is general and can be applied to other languages and data sets;
2. that the city representations capture enough distinction to produce competitive results in geolocation, even this was not the main focus;
3. that retrofitting can be used to incorporate geographic information into the embeddings, extending the original algorithm's applications;
4. that the clusterings match with a sociolinguistic dialect map (Lameli, 2013), measuring their homogeneity, completeness, and their harmonic mean (V-measure), and reach a V-measure of 0.77, beating an informed baseline;

We publicly release the data, code, and map files for future research at https://github.com/Bocconi-NLPLab.

2 Data

2.1 Source

We use data from the social media app Jodel,[2] a mobile chat application that lets people anonymously talk to other users within a 10km-radius around them. The app was first published in 2014, and has seen substantial growth since its beginning. It has several million users in the German-speaking area (GSA), and is expanding to France, Italy, Scandinavia, Spain, and lately the United States. Users can post and answer to posts within the radius around their own current location. All users are anonymous. Answers to an initial post are organized in threads. The vast majority of posts in Jodel are written in standard German, but since it is conceptually spoken langauge (Koch and Oesterreicher, 1985; Eisenstein, 2013), regional and dialectal forms are common, especially in Switzerland, Austria, and rural areas in Southern Germany. The data therefore reflects current

---

[1] Han et al. (2014) has used city-level representations, but have not applied them to the identification of dialect areas.

[2] `https://jodel.com/`

developments in language dynamics to mark regionality (Purschke, 2018).

We used a publicly available API to collect data between April and June 2017 from 123 initial locations: 79 German cities with a population over 100k people, all 17 major cities in Austria ("Mittel- und Oberzentren"), and 27 cities in Switzerland (the 26 cantonal capitals plus Lugano in the very south of the Italian-speaking area). Due to the 10km radius, posts from other nearby cities get collected as well. We include these additional cities if they have more than 200 threads, thereby growing the total number of locations.[3] Ultimately, this results in 408 cities (333 in Germany, 27 in Austria, 48 in Switzerland). The resulting locations are spread relatively evenly across the entire GSA, albeit with some gaps in parts of Germany with low population density. In total, we collect 2.3 million threads, or 16.8 million posts.

We treat each thread as a document in our representation learning setup, labeled with the name of the city in which the thread took place.

2.2 Preprocessing

We preprocess the data to minimize vocabulary size, while maintaining regional discriminative power. We lowercase the input and restrict ourselves to content words, based on the part-of-speech (nouns, verbs, adjectives, adverbs, and proper names), using the `spacy`[4] tagger.

Prior studies showed that many regionally-distributed content words are topically driven (Eisenstein et al., 2010; Salehi et al., 2017). People talk more about their own region than about others, so the most indicative words include place names (the own city, or specific places within that city), and other local culture terms, such as sports teams. We try to minimize the effect of such regional topics, by excluding all named entities, as well as the names of all cities in our list, to instead focus on dialectal lexical variation.

We use `NLTK`[5] to remove German stop words, and to lemmatize the words. While this step removes the inflectional patterns found in German, which could have regional differences, we focus here on lexical differences, and lemmatization greatly reduces vocabulary size, leading to better representations. While both POS-tagging and NER can introduce noise, they are more flexible and exhaustive than pre-defined word lists.[6] Finally, we concatenate collocations based on the PMI of the adjacent words in the cleaned corpus. The average instance length is about 40 words after cleaning.

2.3 Data Statement

The corpus was selected to represent informal, everyday online speech across the German-speaking area in Europe, and to capture regional distinctions. The data was acquired via the publicly available API. The language is mainly standard German, but with a substantial amount of dialectal entries, mainly from southern German varieties, as well as some French and Italian, which could not be removed without losing dialect. The platform is anonymous, but mainly used by young people, as indicated by a prevalence of college-related topics. It contains spontaneous, written, asynchronous interactions in a chat platform organized by threads. Anonymous reference to prior interlocutors is possible. The app is mainly used to discuss everyday topics, entertainment, flirting, venting, and informal surveys.

3 Methodology

3.1 Representation Learning

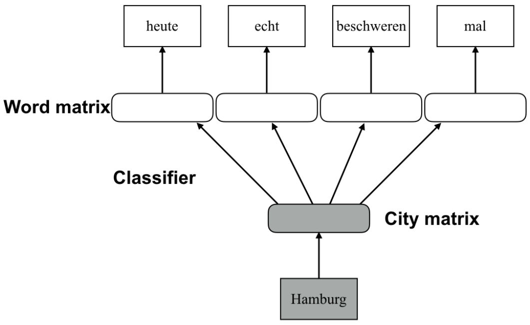

Figure 1: Doc2vec model example for window size 4.

To learn both word and city representations, we use the Doc2Vec implementation of para-

---

[3] The number of threads differs widely even between cities, ranging from dozens to over 40k in cities like Munich, Vienna, or Berlin.

[4] https://spacy.io/

[5] http://www.nltk.org/

[6] Note that stopwords and place names are more reliably detected in their standard form than in regional variants of abbreviations, meaning the standard forms are more reliably excluded if posts are written in High German, than if posts are written in dialect. This may lead to higher coherence for regions with a higher amount of non-standard tokens (as in Switzerland), thereby actually supporting our goal of detecting regional variation.

graph2vec (Le and Mikolov, 2014) in `gensim`.[7] The model is conceptually similar to `word2vec` (Mikolov et al., 2013), but also learns document label representations (in our case city names), embedded in the same space as the words. We use distributed bag-of-words (DBOW) training. The model parameters are fitted by predicting randomly sampled context words from a city vector. The objective is to maximize the log probability of the prediction,

$$y = \arg\max_{W} \log \sum_{i=1}^{N} log(p(w_i|k))$$

where k is a city, and $W = w_{i...N}$ a sequence of N randomly sampled words from the thread (see Figure 1 for a schematic representation).

During training, semantically similar words end up closer together in vector space, as do words "similar" to a particular city, and cities that are linguistically similar to each other.

Due to the nature of our task, we unfortunately do not have gold data (i.e., verified cluster labels) to tune parameters.We therefore follow the settings described in (Lau and Baldwin, 2016) for the parameters, and set the vector dimensions to 300, window size to 15, minimum frequency to 10, negative samples to 5, downsampling to 0.00001, and run for 10 iterations.

3.2 Visualization

In order to examine whether the city embeddings capture the continuous nature of dialects, we visualize them. If our assumption holds, we expect to see gradual continuous change between cities and regions.

We use non-negative matrix factorization (NMF) on the 300-dimensional city representation matrix to find the first three principal components, normalize them each to values 0.0–1.0 and interpret those as RGB values.[8] I.e., we assume the first principal component signals the amount of red, the second component the amount of green, and the third component the amount of blue. This triple can be translated into a single color value. E.g., 0.5 red, 0.5 green, and 0.5 blue translates into medium gray. This transformation translates city representations into color values that preserve linguistic similarities. Similar hues correspond to similar representations, and therefore, by extension, linguistic similarity.

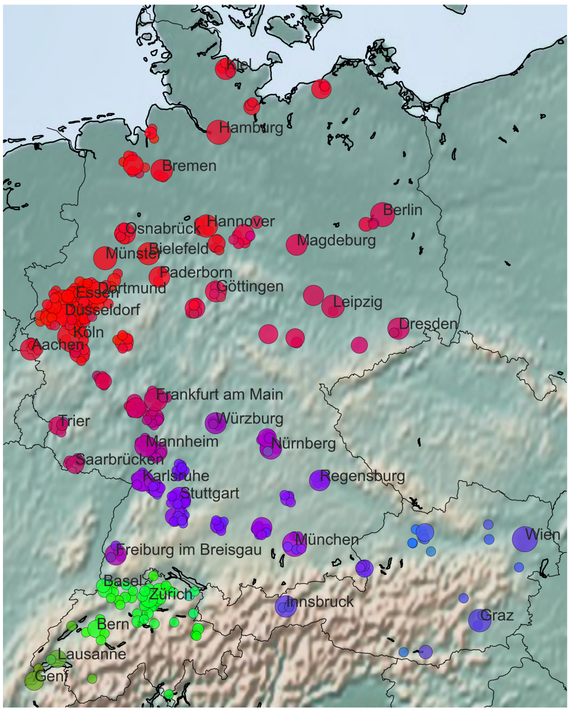

Figure 2: Gradient color map of first three components of city embeddings, interpreted as RGB, for all cities above 200 threads. Color reflects linguistic similarity.

NMF tries to find a decomposition of a given i-by-k matrix W into d components by a i-by-d row-representation V and a d-by-k column representation H. In our case, $d = 3$. Since we are only interested in a reduced representation of the cities, V, we discard H.

The result is indeed a continuous color gradient over the cities over 200 threads, see Figure 2. The circle size for every city indicates the relative number of threads per location.

In order to get reliable statistics, we restrict ourselves to cities with more than 200 observed conversations (about 2.1M conversations: 1.82M in Germany, 173k in Austria, and 146k in Switzerland). Including cities with fewer conversations adds more data points, but induces noise, as many of those representations are based on too little data, resulting in inaccurate vectors.

Even without in-depth linguistic analysis, we can already see differences between Switzerland (green color tones) and the rest of the GSA. Within

---

[7] `https://radimrehurek.com/gensim/models/doc2vec.html`

[8] Note that instead learning 3-dimensional embeddings would not amount to the same, as those are likely *not* equivalent of the three first principal components, and thus not as useful. 300 dimensions capture other degrees of variation, increasing the chance to capture meaningful latent dimensions.

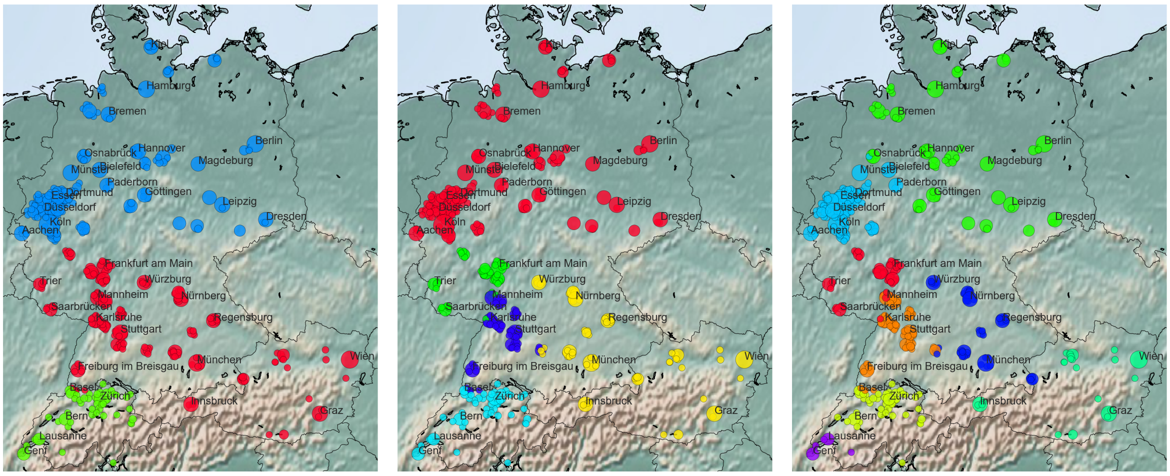

Figure 3: Clustering solutions of retrofit city embeddings for entire GSA with 3, 5, and 8 clusters. Colors denote clusters, assigned randomly.

Switzerland, we see a distinction between the German (lighter green) and the French-speaking area around Lausanne and Geneva (darker tones). On the other hand, we find a continuous transition from red over purple to bluish colors in Germany and Austria. These gradients largely correspond to the dimensions North→South(East): red→blue and West→East: intense tones →pale tones. These dimensions mirror the well-known strong linguistic connection between the southeast of Germany and Austria, and between most cities in the north of Germany.

3.3 Clustering

The visualization in the last section already suggests that we capture the German dialect continuum, and the existence of larger dialect areas. However, in order to evaluate against existing dialect maps, we need to discretize the continuous representation. We use hierarchical agglomerative clustering (Ward Jr, 1963) with Ward linkage, Euclidean affinity, and structure to discover dialect areas. We compare the agglomerative clustering results to a k-means approach.

Agglomerative clustering starts with each city in its own cluster, and recursively merges pairs into larger clusters, until we have reached the required number. Pairs are chosen to minimize the increase in linkage distance (for Ward linkage, this measure is the new cluster's variance). We use cities with 50–199 threads (66 cities) to tune the clustering parameters (linkage function and affinity), and report results obtained on cities with more than 200 threads.

Since the city representations are indirectly based on the words used in the respective cities, the clustering essentially captures regional similarity in vocabulary. If the clusters we find in our data match existing dialect distinctions, this provides a compelling argument for the applicability of our methodology.

3.4 Including geographic knowledge

While we capture regional variation by means of linguistic similarities here, it does include a geographic component as well. The embeddings we learn do not include this component, though. This can produce undesirable clustering results. Large cities, due to their "melting-pot" function, often use similar language, so their representations are close in embedding space. This is an example of Galton's problem (Naroll, 1961): Munich and Berlin are not linguistically similar because they belong to the same dialect, but due to some outside factor (in this case, shared vocabulary through migration).

To address geography, we experiment with two measures: clustering with structure, and retrofitting (Faruqui et al., 2015; Hovy and Fornaciari, 2018).

Structure To introduce geographic structure into clustering, we use a connectivity matrix over the inverse distance between cities (i.e., geographically close cities have a higher number), which is used as weight during the merging. This weight makes close geographic neighbors more likely to be merged before distant cities are.

Note, though, that this geographic component does *not* predetermine the clustering outcome: geographically close cities that are linguistically different still end up in separate clusters, as we will see. The Spearman ρ correlation between the geographic distance and the cosine-similarity of cities is positive, but does not fully explain the similarities (Austria 0.40, Germany 0.42, Switzerland 0.72). The stronger correlation for Switzerland suggests a localized effect of regional varieties. Geographic structure in clustering does, however, provide speedups, regional stability, and more stable clustering solutions than unstructured clustering. We will see this in comparison to k-means.

Retrofitting Faruqui et al. (2015) introduced retrofitting of vectors based on external knowledge. We take the idea proposed for word vectors and semantic resources and extend it following Hovy and Fornaciari (2018) to apply it to city representations and membership in geographic regions. We construct a set Ω with tuples of cities (c_i, c_j) such that there exists a region R where $c_i \in R$ and $c_j \in R$. We use the NUTS2 regions (Nomenclature of Territorial Units for Statistics, a EuroStats geocoding standard) to determine R. In Germany, NUTS2 has 39 regions, corresponding to government regions.

To include the geographic knowledge, we retrofit the existing city embeddings C. The goal is to make the representations of cities that are in the same region more similar to each other than to cities in other regions, resulting in a retrofit embeddings matrix $\hat{C}$. For a retrofit city vector $\hat{c}_i$, the update equation is

$$\hat{c}_i = \alpha c_i + \beta \frac{\sum_{j:(i,j)\in\Omega} \hat{c}_j}{N}$$

where $\hat{c}_i$ is the original city vector, and α and β are tradeoff parameters to control the influence of the geographic vs. the linguistic information. See Faruqui et al. (2015) and Hovy and Fornaciari (2018) for more details.

4 Evaluation

In order to evaluate our methodology, we measure both its ability to match German dialect distinctions, and the performance of the learned embeddings in a downstream geolocation task.

Figure 3 provides examples of different clustering solutions after retrofitting. Note that colors are assigned randomly and do *not* correspond to the linguistic similarity from Figure 2. Switzerland immediately forms a separate cluster (the 2-cluster solution separates Switzerland vs. everything else), and further clusters first separate out more southern German varieties before distinguishing the northern varieties. This is in line with sociolinguistic findings (Plewnia and Rothe, 2012) about ubiquity of dialect use (more common in the south, therefore more varied regions, reflected in our clustering). Due to space constraints, we have to omit further clustering stages, but find linguistically plausible solutions beyond the ones shown here. For an in-depth qualitative analysis of the different clustering solutions and the socio-demographic and linguistic factors, see Purschke and Hovy (In Preparation).

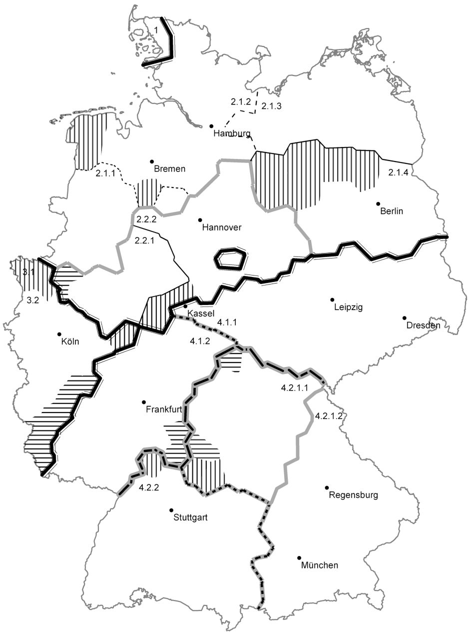

Figure 4: German dialect Regions after Lameli (2013). Shaded areas denote dialect overlap.

Dialect match We use the map of German dialects and their regions by Lameli (2013) (see Figure 4) and its 14 large-scale areas[9] as gold standard to measure how well the various clustering-solutions correspond to the dialect boundaries. This map is based on empirical quantitative analysis of German dialects, albeit based on data from the 19th century, and therefore naturally on different domains and media than our study.

Note that we can only assess the cities within modern-day Germany (clusters formed in Austria or Switzerland are not covered). We therefore rerun the clusterings on the subset of German cities, so results differ slightly from the clusters induced

---

[9] Some areas partially overlap with each other.

	ORIGINAL						RETROFIT					
	K-MEANS			AGGLOMERATIVE			K-MEANS			AGGLOMERATIVE		
#	V-score	H	C	V-score	H	C	V-score	H	C	V-score	H	C
2	0.41	0.27	0.89	0.41	0.27	0.83	0.43	0.28	0.94	0.44	0.28	0.95
3	0.53	0.39	0.84	0.46	0.33	0.73	0.57	0.42	0.87	0.54	0.40	0.85
4	0.61	0.49	0.80	0.59	0.48	0.76	0.66	0.53	0.86	0.68	0.56	0.88
5	0.61	0.50	0.79	0.63	0.54	0.74	0.69	0.59	0.83	0.71	0.62	0.84
6	0.65	0.56	0.76	0.64	0.58	0.72	0.72	0.64	0.82	0.72	0.64	0.82
7	0.64	0.57	0.74	0.66	0.61	0.72	0.72	0.65	0.80	0.69	0.64	0.76
8	0.62	0.56	0.70	0.66	0.61	0.71	0.70	0.67	0.74	0.73	0.70	0.76
9	0.70	0.65	0.76	0.70	0.68	0.72	0.70	0.67	0.73	0.73	0.70	0.75
10	0.68	0.66	0.70	0.70	0.68	0.72	0.71	0.70	0.72	0.74	0.72	0.75
11	0.69	0.67	0.71	**0.72**	0.71	0.72	0.74	0.75	0.74	0.74	0.74	0.74
12	0.66	0.65	0.67	0.70	0.72	0.68	0.71	0.72	0.69	0.75	0.78	0.72
13	0.67	0.68	0.66	0.70	0.73	0.67	0.73	0.75	0.71	0.74	0.78	0.70
14	0.67	0.68	0.66	0.69	0.73	0.65	0.71	0.76	0.66	0.74	0.79	0.70
15	0.66	0.68	0.64	0.71	0.77	0.66	0.74	0.80	0.70	0.76	0.82	0.70
16	0.67	0.71	0.64	0.71	0.78	0.66	0.73	0.80	0.67	**0.77**	0.85	0.71
17	0.67	0.70	0.63	0.70	0.78	0.64	0.73	0.81	0.67	0.76	0.85	0.68
18	0.65	0.68	0.62	0.70	0.78	0.64	0.74	0.83	0.66	0.75	0.85	0.66
19	0.64	0.68	0.59	0.70	0.79	0.63	0.72	0.82	0.64	0.75	0.87	0.67
20	0.65	0.71	0.59	0.69	0.80	0.61	0.74	0.85	0.66	0.75	0.87	0.66

Table 1: Evaluation of the fit of various cluster solutions against the reference dialect map Lameli (2013). k-means results averaged over 5 runs. Agglomerative clustering with structure. Retrofitting on NUTS2 regions. Baseline: 0.74 V-score, 0.93 homogeneity, 0.62 completeness.

on the entire GSA.

We report *homogeneity* (whether a cluster contains only data points from a single region) and *completeness* (how many data points of a NUTS region are in the same cluster), as well as their harmonic mean, the *V-score*. This corresponds to precision/recall/F1 scores used in classification. Note that we will not be able to faithfully reconstruct Lameli's distinctions, since Lameli's map contains overlapping regions, whose data points therefore already violate perfect homogeneity.

The outline of dialect regions in Lameli's map is based on the NUTS2 regions, so we compare all clustering solutions to an informed baseline that assigns each city the NUTS2 region it is located in. Except for regions in dialect overlaps, each NUTS region is completely contained in one dialect region, so the baseline can achieve almost perfect homogeneity.

Downstream task geolocation For the geolocation task, we randomly select 100 cities with at least 200 threads from each country (7 in Austria, 82 in Germany, 11 in Switzerland). We then collect threads with at least 100 words from these cities for each country (11,240 threads from Austria, 137,081 from Germany, and 18,590 from Switzerland). Each thread is a training instance, i.e., we have 166,911 instances. We use the Doc2Vec model from before to induce a document representation for each instance and use the vector as input to a logistic regression model that predicts the city name.

For testing, we sample 5,000 threads from the same cities (maintaining the same proportional distribution and word count constraint), but from a *separate* data set, collected two months after the original sample. We again use the Doc2Vec model to induce representations, and evaluate the classifier on this data.

We measure accuracy, accuracy at 161km (100 miles), and the median distance between prediction and target. We compare the model with Doc2Vec representations to a bag-of-words (BOW) model with the same parameters. Since the representation here is based on words, we can not apply retrofitting. As baseline, we report the most-frequent city prediction.

5 Results

Dialect match Table 1 shows the results of clustering solutions up to 20 clusters for both retrofit and original embeddings. Irrespective of the clustering approach, retrofit representations perform markedly better.

Homogeneity increases substantially the more clusters we induce (in the limit, each data point becomes a single cluster, resulting in perfect homogeneity), whereas completeness decreases slightly with more clusters (they increase the likelihood that a region is split up into several clusters). We achieve the best V-score, 0.77, with 16 clusters.

Averaged k-means (over 5 runs) is much less consistent, due to random initialization, but presumably also because it cannot incorporate the geographic information. For few clusters, its performance is better than agglomerative clustering, but as the number of clusters increases (and the geographic distribution of the cities becomes more intricate), k-means stops improving.

The baseline achieves almost perfect homogeneity, as expected (the only outliers are NUTS regions in overlap areas). Completeness is lower than almost all clustering solutions, though. The V-score, 0.74, is therefore lower than the best clustering solution.

Both the cluster evaluation metrics and the visual correspondence suggest that our method captures regional variation at a lexical level well.

MODEL	↑ACC	↑ACC@161	↓MED. DIST.
baseline	0.03	0.31	269.33
BOW	0.21	0.50	156.17
D2V	**0.26**	**0.52**	**145.16**

Table 2: Geolocation performance for city embeddings and bag-of-word vectors on held-out data set. Baseline predicts most frequent city from training data.

Downstream Evaluation: Geolocation Table 2 shows the results of the geolocation downstream task. Despite the fact that the representation learning setup was *not* designed for this task and excluded all the most informative words for it (Salehi et al., 2017), the induced embeddings capture enough pertinent regional differences to achieve reasonable performance (albeit slightly below state of the art, which typically has a median distance around 100km, and an accuracy@161 of 0.54, see cf. Rahimi et al. (2017b)) and decidedly outperform the BOW model and most-frequent-city baseline on all measures.

6 Analysis

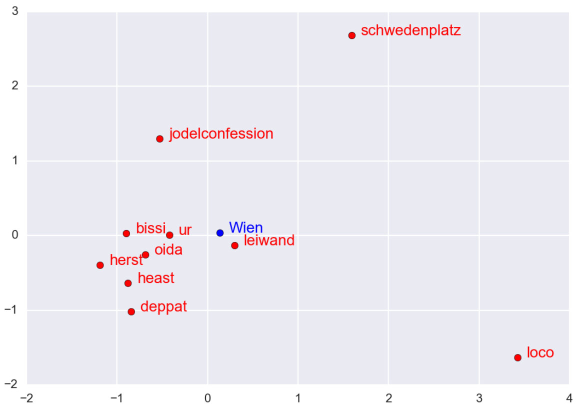

Figure 5: Visualization of city representation for Wien (Vienna) and its 10 nearest word neighbors in two dimensions. The closest seven words are all Austrian dialect words

Because both words and cities are represented in the same embeddings space (at least before retrofitting), we can compare the vectors of cities to each other (asking: which cities are linguistically most similar to each other, which is what we have done above) and words to cities (asking: which words are most similar to/indicative of a city). The latter allows us to get a qualitative sense of how descriptive the words are for each city.

Figure 5 shows an example of word and city similarity for the city representation of Vienna.

We can also use the cluster centroid of several city vectors to represent entire regions. The new vector no longer represents a real location, but is akin to the theoretic linguistic center of a dialect region. We can then find the most similar words to this centroid. For the solution with 3 clusters (cf. Figure 3), we get the solutions in Table 3. As expected, the regional prototypes do not overlap, but feature dialectal expressions in the south, and general standard German expressions in the north.

Again, for an in-depth qualitative analysis and discussion of the socio-linguistic correlations, see Purschke and Hovy (In Preparation).

7 Related Work

Dialectometric studies, exploring quantitative statistical models for regional variation, range from

CLUSTER	PROTOTYPES	TRANSLATION
Switzerland	esch, ond, vell, gaht, wüki, nöd, besch, emmer, nor, au nöd	*is, and, many, goes, really, not, (you) are, always, just, also not*
Northern Germany	ja gut, erstmal, sieht, drauf, vielleicht, mehr, gut, sehen, schonmal, Ahnung	*well yes, first, sees, onto, maybe, more, good, see, already, idea*
Southern Germany & Austria	afoch, voi, nd, i a, oda, möppes, nimma, is a, mei, gscheid	*easy, full, and, me too, or, girl (*SLANG*), no more, is also, well, smart*

Table 3: Prototypical words (10 nearest neighbors) for each of 3 clusters.

work on dialect data in Dutch (Nerbonne and Heeringa, 1997; Prokić and Nerbonne, 2008; Wieling et al., 2011, inter alia) and British English (Szmrecsanyi, 2008), to Twitter-based approaches for American dialect distinctions (Grieve et al., 2011; Huang et al., 2016) and the regional differentiation of African American Vernacular English (Jones, 2015). While these papers rely on existing dialect maps for comparison, they rarely quantitatively evaluate against them, as we do.

Recently, NLP has seen increased interest in *computational sociolinguistics* (Nguyen et al., 2016). These works examine the correlation of socio-economic attributes with linguistic features, including regional distribution of lexical and phonological differences (Eisenstein et al., 2010; Doyle, 2014; Bamman et al., 2014), syntactic variation (Johannsen et al., 2015), diachronic variation (Danescu-Niculescu-Mizil et al., 2013; Kulkarni et al., 2015; Hamilton et al., 2016), and correlation with socio-demographic attributes (Eisenstein et al., 2011; Eisenstein, 2015). Other have further explored regional variation on social media, and showed the prevalence of regional lexical variants (Hovy et al., 2015; Hovy and Johannsen, 2016; Donoso and Sánchez, 2017). Several works include quantitative comparisons to measure the empirical fit of their findings (Peirsman et al., 2010; Han et al., 2014; Huang et al., 2016; Grieve, 2016; Kulkarni et al., 2016), albeit not on entire existing dialect maps.

The use of representation learning is new and relatively limited, especially given its prevalence in other areas of NLP. Bamman et al. (2014) have shown how regional meaning differences can be learned from Twitter via distributed word representations between US states, but not for individual cities. More recently, Kulkarni et al. (2016); Rahimi et al. (2017a) and Rahimi et al. (2017b) have shown how neural models can exploit regional lexical variation for geolocation, while also enabling dialectological insights, whereas our goals are exactly reversed. Östling and Tiedemann (2017) have shown how distributed representations of entire national languages capture typological similarities that improve translation quality. Most of these papers focus on downstream performance that accounts for regional variation, rather than on explicitly modeling variation. We include a downstream performance, but also evaluate the cluster composition quantitatively.

8 Conclusion

We use representation learning, structured clustering, and geographic retrofitting on city embeddings to study regional linguistic variation in German. Our approach captures gradual linguistic differences, and matches an existing German dialect map, achieving a V-score of 0.77. The learned city embeddings also capture enough regional distinction serve as input to a downstream geolocation task, outperforming a BOW baseline and producing competitive results. Our findings indicate that city embeddings capture regional linguistic variation, which can be further enriched with geographic information via retrofitting. They also suggest that traditional ideas of regionality persist online. Our methodology is general enough to be applied to other languages that lack dialect maps (e.g., Switzerland), and to other tasks studying regional variation. We publicly release our data and code.

Acknowledgements

We would like to thank the anonymous reviewers of this paper and Barbara Plank, who helped to strengthen and clarify our findings.

References

David Bamman, Chris Dyer, and Noah A. Smith. 2014. Distributed Representations of Geographically Situated Language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 828–834. Proceedings of ACL.

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. No country for old members: User lifecycle and linguistic change in online communities. In *Proceedings of the 22nd international conference on World Wide Web*, pages 307–318. ACM.

Gonzalo Donoso and David Sánchez. 2017. Dialectometric analysis of language variation in Twitter. *VarDial 2017*, page 16.

Gabriel Doyle. 2014. Mapping Dialectal Variation by Querying Social Media. In *EACL*, pages 98–106.

Jacob Eisenstein. 2013. What to do about bad language on the Internet. In *Proceedings of the 2013 conference of the North American Chapter of the association for computational linguistics: Human language technologies*, pages 359–369.

Jacob Eisenstein. 2015. Systematic patterning in phonologically-motivated orthographic variation. *Journal of Sociolinguistics*, 19(2):161–188.

Jacob Eisenstein, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287. Association for Computational Linguistics.

Jacob Eisenstein, Noah A Smith, and Eric P Xing. 2011. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1365–1374. Association for Computational Linguistics.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615.

Jack Grieve. 2016. *Regional variation in written American English*. Cambridge University Press.

Jack Grieve, Dirk Speelman, and Dirk Geeraerts. 2011. A statistical method for the identification and aggregation of regional linguistic variation. *Language Variation and Change*, 23(2):193–221.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Proceedings of the 54th Meeting of the Association for Computational Linguistics*, pages 1489–1501. Association for Computational Linguistics.

Bo Han, Paul Cook, and Timothy Baldwin. 2014. Text-based Twitter user geolocation prediction. *Journal of Artificial Intelligence Research*, 49:451–500.

Dirk Hovy and Tommaso Fornaciari. 2018. Increasing In-Class Similarity by Retrofitting Embeddings with Demographic Information. In *Proceedings of the 2018 conference on Empirical Methods in Natural Language Processing*.

Dirk Hovy and Anders Johannsen. 2016. Exploring Language Variation Across Europe–A Web-based Tool for Computational Sociolinguistics. In *Proceedings of LREC*.

Dirk Hovy, Anders Johannsen, and Anders Søgaard. 2015. User review-sites as a source for large-scale sociolinguistic studies. In *Proceedings of WWW*.

Dirk Hovy, Afhsin Rahimi, Tim Baldwin, and Julian Brooke. In Preparation. Visualizing Regional Language Variation Across Europe on Twitter. In Stanley D. Brunn and Roland Kehrein, editors, *Handbook of the Changing World Language Map*. Dordrecht: Springer.

Yuan Huang, Diansheng Guo, Alice Kasakoff, and Jack Grieve. 2016. Understanding US regional linguistic variation with Twitter data analysis. *Computers, Environment and Urban Systems*, 59:244–255.

Anders Johannsen, Dirk Hovy, and Anders Søgaard. 2015. Cross-lingual syntactic variation over age and gender. In *Proceedings of CoNLL*.

Taylor Jones. 2015. Toward a description of African American Vernacular English dialect regions using "Black Twitter". *American Speech*, 90(4):403–440.

Peter Koch and Wulf Oesterreicher. 1985. Sprache der Nähe-Sprache der Distanz. Mündlichkeit und Schriftlichkeit im Spannungsfeld von Sprachtheorie und Sprachgeschichte. *Romanistisches Jahrbuch*, 36:15–43.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635. International World Wide Web Conferences Steering Committee.

Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2016. Freshman or Fresher? Quantifying the Geographic Variation of Language in Online Social Media. In *ICWSM*, pages 615–618.

Alfred Lameli. 2013. *Strukturen im Sprachraum: Analysen zur arealtypologischen Komplexität der Dialekte in Deutschland*, volume 54. Walter de Gruyter.

Jey Han Lau and Timothy Baldwin. 2016. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. page 78.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Shachar Mirkin and Jean-Luc Meunier. 2015. Personalized machine translation: Predicting translational preferences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal. Association for Computational Linguistics*.

Raoul Naroll. 1961. Two solutions to Galton's problem. *Philosophy of Science*, 28(1):15–39.

John Nerbonne and Wilbert Heeringa. 1997. Measuring dialect distance phonetically. In *Proceedings of the Third Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON-97)*, pages 11–18.

John Nerbonne, Wilbert Heeringa, and Peter Kleiweg. 1999. Edit distance and dialect proximity. *Time Warps, String Edits and Macromolecules: The theory and practice of sequence comparison*, 15.

Dong Nguyen, A Seza Doğruöz, Carolyn P Rosé, and Franciska de Jong. 2016. Computational sociolinguistics: A survey. *Computational linguistics*.

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 644–649, Valencia, Spain. Association for Computational Linguistics.

Yves Peirsman, Dirk Geeraerts, and Dirk Speelman. 2010. The automatic identification of lexical variation between language varieties. *Natural Language Engineering*, 16(4):469–491.

Albrecht Plewnia and Astrid Rothe. 2012. Sprache – Einstellungen – Regionalität. In Ludwig M. Eichinger, Albrecht Plewnia, Christiane Schoel, Dagmar Stahlberg, and Gerhard Stickel, editors, *Sprache und Einstellungen. Spracheinstellungen aus sprachwissenschaftlicher und sozialpsychologischer Perspektive. Mit einer Sprachstandserhebung zum Deutschen*.

Jelena Prokić and John Nerbonne. 2008. Recognising groups among dialects. *International journal of humanities and arts computing*, 2(1-2):153–172.

Christoph Purschke. 2018. Language regard and cultural practice: Variation, evaluation, and change in the German regional languages. In Betsy Evans, Erica Benson, and James Stanford, editors, *Language regard: Methods, variation, and change*, pages 245–261. Cambridge University Press, Cambridge.

Christoph Purschke and Dirk Hovy. In Preparation. Lörres, Möppes, and the Swiss. (Re)Discovering Regional Patterns in Anonymous Social Media Data. *Journal of Linguistic Geography*.

Stefan Rabanus, Roland Kehrein, and Alfred Lameli. 2010. Creating digital editions of historical maps. *Language and space*, 2:375–385.

Afshin Rahimi, Timothy Baldwin, and Trevor Cohn. 2017a. Continuous representation of location for geolocation and lexical dialectology using mixture density networks. In *Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2017b. A neural model for user geolocation and lexical dialectology. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL2017)*. Association for Computational Linguistics (ACL2017).

Bahar Salehi, Dirk Hovy, Eduard Hovy, and Anders Søgaard. 2017. Huntsville, hospitals, and hockey teams: Names can reveal your location. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 116–121.

Robert G Shackleton Jr. 2005. English-American speech relationships: A quantitative approach. *Journal of English Linguistics*, 33(2):99–160.

Philippa Shoemark, James Kirby, and Sharon Goldwater. 2017a. Topic and audience effects on distinctively Scottish vocabulary usage in Twitter data. In *Proceedings of VarDial Workshop*. Association for Computational Linguistics.

Philippa Shoemark, Debnil Sur, Luke Shrimpton, Iain Murray, and Sharon Goldwater. 2017b. Aye or naw, whit dae ye hink? Scottish independence and linguistic identity on social media. In *Proceedings of EACL*. Association for Computational Linguistics.

Benedikt Szmrecsanyi. 2008. Corpus-based dialectometry: aggregate morphosyntactic variability in British English dialects. *International Journal of Humanities and Arts Computing*, 2(1-2):279–296.

Peter Trudgill. 2000. *Sociolinguistics: An introduction to language and society*. Penguin UK.

Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Martijn Wieling, John Nerbonne, and R Harald Baayen. 2011. Quantitative social dialectology: Explaining linguistic variation geographically and socially. *PloS one*, 6(9):e23613.

Characterizing Interactions and Relationships between People

Farzana Rashid and **Eduardo Blanco**
Human Intelligence and Language Technologies Lab
University of North Texas
Denton, TX, 76203
farzanarashid@my.unt.edu, eduardo.blanco@unt.edu

Abstract

This paper presents a set of dimensions to characterize the association between two people. We distinguish between interactions (when somebody refers to somebody in a conversation) and relationships (a sequence of interactions). We work with dialogue scripts from the TV show Friends, and do not impose any restrictions on the interactions and relationships. We introduce and analyze a new corpus, and present experimental results showing that the task can be automated.

1 Introduction

People interact with each other and as a result form relationships. These relationships range from weak (e.g., John talking to a waiter to order a drink) to strong (e.g., John and his best friend discussing career options). Traditionally, information extraction systems target, among others, relationships between people, e.g., PARENT, SIBLING, OTHER-PERSONAL, OTHER-PROFESSIONAL (Doddington et al., 2004).

Extracting a label describing the general relationship between two entities—often called *relation type*—is useful for tasks such as summarization (Jijkoun et al., 2004) and question answering (White et al., 2001). Only assigning a relation type, however, does not account for nuances in the relationship between two individuals. First, a relationship can be characterized beyond a relation type. For example, people having a PROFESSIONAL relationship may be *spatially near* or *distant* (working at the same or different offices), and have an *equal* or *hierarchical* relationship (two software developers or a developer and the CEO). Second, relationships are defined by multiple interactions, and the fine-grained characteristics of interactions do not necessarily mirror the characteristics of the corresponding relationship. For example, software developers having a *cooperative* PROFESSIONAL relationship may have a *heated* interaction in a meeting that does not affect the long-term PROFESSIONAL relationship. Similarly, the same software developers having a *task-oriented* PROFESSIONAL relationship may have occasionally *pleasure oriented* interactions (e.g., when they go out for drinks on Fridays).

This kind of fine-grained characteristics of interactions and relationships are called *dimensions* in social science (Wish et al., 1976). Social scientists have also studied language usage and how people interact with each other depending on their relationship. For example, Gibbs (2000) studies irony (sarcasm, hyperbole, understatement, etc.) in communications among friends, and Snyder and Stukas Jr (1999) analyze the expectations in social interactions (e.g., interactions between strangers tend to be more formal) as well as the consequences of breaking expectations. In the social sciences, however, researchers mostly focus on how people act (e.g., how they talk to each other and about each other) and how people perceive interactions and relationships. In general, they do not attempt to automatically characterize interactions or relationships from language usage.

In this paper, we characterize the interactions between people and the resulting relationships. The main contributions of this paper are: (a) a set of dimensions to characterize interactions and relationships, including dimensions previously defined in the social sciences and two novel dimensions; (b) annotations of these dimensions for all interactions and relationships in 24 episodes of the TV show Friends (Season 1); (c) corpus analysis including agreements, label distributions and correlations; and (d) experimental results showing that classifiers grounded on language usage (and discarding the names of people being considered) are successful at automating the task.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4395–4404
Brussels, Belgium, October 31 - November 4, 2018.

2 Previous Work

Within natural language processing, there have been several recent efforts working with relationships between people. Voskarides et al. (2015) extract human-readable descriptions of relations in a knowledge graph. Unlike the work presented here, they experiment with a proprietary knowledge graph and rely heavily on features extracted from the graph. Iyyer et al. (2016) propose an unsupervised algorithm to extract relationship trajectories of fictional characters. Bracewell et al. (2012) introduce social acts (e.g., agreement, undermining) designed to characterize relationships exhibiting adversarial and collegial behavior (similar to our cooperative vs. competitive dimension). None of these works distinguish between interactions and relationships, characterize interactions and relationships with dimensions, or consider all interactions between two people.

In our previous work, we characterize interpersonal relationships with dimensions (Rashid and Blanco, 2017). In this paper, we improve upon our previous effort as follows. First, we distinguish between interpersonal interactions and relations. Second, we work with dialogues thus the same people interact with each other many times.

There have been a few studies on analyzing language usage when people communicate. For example, Danescu-Niculescu-Mizil et al. (2012) study how power differences affect language style in online communities, and Prabhakaran and Rambow (2014) present a classifier to detect power relationships in email threads. Similarly, Gilbert (2012) explores how people in hierarchical relationships communicate through email, and Bramsen et al. (2011) focus on identifying power relationships in social networks. Politeness in online forums has also been studied (Danescu-Niculescu-Mizil et al., 2013). While power (similar to our equal vs. hierarchical dimension, Section 3) and politeness could be considered dimensions, these works exploit structural and linguistic features derived from communications between two individuals. Unlike all of them, we distinguish between and characterize interactions and relationships, and automate the task using only information derived from language usage.

Information extraction systems target, among others, relationships between people. There have been many evaluations (Grishman and Sundheim, 1996; Doddington et al., 2004; Kulick et al., 2014; Surdeanu and Heng, 2014), and there are two main approaches. Traditionally, relationships are defined before training takes place (e.g., PARENT, FRIENDS), and systems are trained using supervised machine learning (Yu and Lam, 2010; Nguyen et al., 2016; West et al., 2014). On the other hand, open information extraction (Wu and Weld, 2010; Angeli et al., 2015) has emerged as an unsupervised domain-independent approach to extract relations. Regardless of details, these previous works extract explicit relationships and do not attempt to characterize instances of relationships with dimensions. Additionally, they do not distinguish between interactions and relationships.

3 Interpersonal Interactions and Relationships

In this paper, we work with transcripts of conversations and define interaction and relationship as follows. An *interaction* between two people x and y exists for each conversation turn by either x or y referring to the other person. A *relationship* between two people x and y exists if there is at least one interaction between them. One could understand a relationship between x and y as the association defined by a sequence of interactions between x and y. Beyond these definitions, we do not impose any restriction on what constitutes an interaction or relationship: interactions occur each time two people refer to each other in their speech (even if they are not talking to each other), and one or more interactions constitute a relationship.

Interactions and relationships between people have been extensively studied in psychology and social sciences in general. The *right* set of dimensions is not agreed upon (Wish et al., 1976; Deutsch, 2011; Adamopoulos, 2012), and we argue that it depends on the domain of interest (e.g., personal diaries vs. news articles covering politics). We note that dimensions apply to interactions and relationships between specific people (i.e., instances of interactions and relationships), not relation types. For example, a KINSHIP relationship between x and y could be *intense* or *superficial* (depending on x and y) and a particular interaction of that relationship may be *spatially near* or *distant* (even for the same x and y).

The dimensions we work with are briefly summarized in Table 1 and described below. All but two dimensions are defined in previous work in the social sciences (see references in Table 1).

		Other descriptors	Ref.
Interaction	Cooperative vs. Competitive	friendly vs. hostile, promotive vs. contrient	[1]
	Active vs. Passive	direct vs. indirect, unequivocal vs. equivocal	[2]
	Concurrent vs. Nonconcurrent	convergent vs. divergent, synchronous vs. asynchronous	New
	Spatially Near vs. Distant	nearby vs. faraway, attached vs. detached	New
Relationship	Equal vs. Hierarchical	autonomy vs. control, submission, dominance	[1]
	Intense vs. Superficial	important vs. insignificant, influential vs. trivial	[1]
	Pleasure vs. Task Oriented	emotionally involved vs. detached	[1]
	Intimate vs. Unintimate	close vs. distant, indifference, randomness	[3]
	Temporary vs. Enduring	momentary vs. lasting, provisional vs. permanent	[4]

Table 1: Dimensions of interpersonal interactions and relationships. [1] stands for (Wish et al., 1976), [2] for (Kelley, 2013), [3] for (Adamopoulos, 2012), and [4] for (Deutsch, 2011). *New* indicates a dimension defined after analyzing several examples and doing pilot annotations.

Dimensions of Interactions. We consider four dimension of interactions between people, i.e., when a speaker refers to a person in a conversation:

- *Cooperative vs. Competitive*. A interaction is *cooperative* if both people (a) have a common interest or goal, (b) like each other, (c) benefit from the interaction, or (d) think alike or share similar views. Otherwise, the interaction is *competitive*.
- *Active vs. Passive*. An interaction is *active* if both people communicate directly. Otherwise, the interaction is *passive*. For example, when John talks to Bill about Mary, John and Bill have an active interaction, and John and *Mary* have a passive interaction.
- *Concurrent vs. Nonconcurrent*. An interaction is *concurrent* if both people are involved in an event or action at the same time (the event does not need to be a communicating event). Otherwise, the interaction is *nonconcurrent*. For example, when John talks to Bill about a trip with Mary, John and Mary have a *concurrent* interaction, but when John talks to Bill about Paul's house, John and Paul have a nonconcurrent interaction.
- *Spatially Near vs. Distant*. An interaction is *spatially near* (or *near* for short) if the interaction is concurrent and both people are at the same location during the event in which they are involved. Otherwise, the interaction is *spatially distant* (or *distant* for short).

Dimensions of Relationships. We consider five dimensions of relationships between people:

- *Equal vs. Hierarchical*. A relationship is *equal* if both people (a) have the same social status, (b) are at the same level in the power structure, (c) share similar responsibilities, or (d) have the same role. Otherwise, the relationship is *hierarchical*.
- *Intense vs. Superficial*. A relationship is *intense* if both people interact with each other frequently, i.e., they are involved repeatedly. Otherwise, the relationship is *superficial*.
- *Pleasure vs. Task Oriented*. A relationship is *pleasure oriented* if both people interact socially and their relationship is not bound by professional rules or regulations. Otherwise, the relationship is *task oriented*.
- *Intimate vs. Unintimate*. A relationship is *intimate* if both people are emotionally close and warm to each other. Otherwise, the relationship in *unintimate*.
- *Temporary vs. Enduring*. A relationship is *temporary* if it lasts less than a day. A relationship is *enduring* if it last over a month. Otherwise (if it lasts more than a day and less than a month), this dimension is undefined.

4 Annotating Dimensions of Interactions and Relationships

Annotating dimensions of interactions and relationships requires a corpus in which the same people interact several times. We augment an existing corpus of scripts from the TV show Friends (Chen and Choi, 2016). More specifically, we work with the 24 episodes from Season 1 because they:

- contain a large number of conversation turns (9,168, see counts per episode in Table 2);
- involve many characters (42 characters speak at least 100 conservation turns, see the characters that interact the most in Table 2, and the full list in the supplementary materials);

- include speaker information (i.e., we have access to who says what); and
- include annotations linking each mention of people in each conversation turn to the actual person (the name of the person).

Beyond size, the main motivation to use this corpus is the last item above: starting from scratch with another corpus of dialogues would require a substantially larger annotation effort. We refer the reader to the afore-cited paper for details, but the original corpus clusters mentions to people such as *guy*, *my brother* and *he* together with other mentions of the same person. The original corpus is publicly available,[1] and we release our annotations as stand-alone annotations.[2]

4.1 Selecting Pairs of People

The corpus we start with makes it straightforward to select pairs of people whose interactions and corresponding relationships will be annotated. We consider as interactions all instances of somebody mentioning (or referring to) somebody else in a conversation turn. We consider as relationships individuals who interact at least once. Note that we do not (a) distinguish between x mentioning y and y mentioning x, and (b) consider as an interaction x talking to y unless the conversation turn contains a mention to y (the mention need not be the actual name, it could be a pronoun or any nominal mention). Our rationale is as follows. First, all dimensions of interactions are symmetric; and all dimensions of relationships are symmetric except *equal* vs. *hierarchical*. Second, the characters of Friends refer to each other explicitly at least once in most conversations and scenes, either using first names or the pronoun *you*. Thus we are considering as an interaction most verbal exchanges.

Table 2 shows basic counts per episode. We show the number of interactions, unique relationships (i.e., interactions between unique pairs of people), and the pair of people who interact the most. The supplementary materials include an extended table listing the number of times each pair of people interact per episode.

Episode	#Turns	#Inters	#Rels	People with most inters.
1	544	143	21	Monica-Rachel (23)
2	358	105	19	Rachel-Barry (14)
3	475	98	20	Joey-Chandler (12)
4	359	64	17	Monica-Rachel (15)
5	290	76	18	Chandler-Janice (14)
6	353	82	16	Chandler-Aurora (32)
7	291	76	15	Ross-Rachel (13)
8	371	64	19	Monica- Mrs. Geller (14)
9	375	75	18	Monica-Rachel (13)
10	273	69	19	Phoebe-David (22)
11	447	159	24	Chandler-Ross (22)
12	351	109	23	Phoebe-Paolo (14)
13	328	120	24	Joey-Mr.Tribbianni (21)
14	294	239	21	Chandler-Janice (28)
15	430	78	17	Joey-Ross (15)
16	470	95	31	Chandler-Nina (14)
17	526	109	17	Monica-Rachel(28)
18	306	42	14	Ross-Rachel (9)
19	183	29	10	Ross-Rachel (9)
20	467	111	20	Rachel-Mindy (30)
21	403	85	24	Monica-Fake Monica (23)
22	468	92	22	Phoebe-Chandler (32)
23	457	104	26	Ross-Susan (19)
24	325	107	18	Ross-Rachel (30)

Table 2: Basic corpus counts. We show the number of conversation turns, interactions (i.e., one person referring to another one), unique relationships (i.e., unique pairs of people who interact with each other), and the pairs of people with most interactions.

4.2 Annotation Process

The annotations were done one episode at a time. Annotators were presented with the full transcript of the episode including speaker information (who speaks what?) and the names of the individuals mentioned in each conversation turn (who do speakers talk about?). Annotators read each episode from the beginning, and annotate dimensions of interactions and relationships after each interaction. Regarding interactions, they were instructed to annotate dimensions taking into account the language of the current conversation turn. Regarding relationships, they were instructed to annotate dimensions taking into account all previous conversation turns within the same episode. For example, if previous turns state that *Rachel* and *Monica* are best friends, the relationship will continue to be annotated *intense* even if an interaction does not indicate so (until a turn indicates that they are not friends, if applicable).

---

[1] `https://github.com/emorynlp/character-mining`

[2] `http://www.cse.unt.edu/~blanco/`

	Speaker	2nd party	Cooperative (I)
1: Wait, does he eat chalk?	Phoebe	Paul	-1
2: Hey, hey, hey, that's not the rule and you know it.	Ross	Woman	-1
3: She is so peaceful.	Monica	Phoebe	1
	Speaker	2nd party	Concurrent (I)
4: Well, then can we meet him?	Rachel	Alan	-1
5: Hey, buddy, what's up!	Chandler	Alan	1
6: Ma, I'm sorry.	Joey	Mrs. Tribbiani	1
	Speaker	2nd party	Equal (R)
7: Oh, ah-the kid has it.	Joey	the kid	-1
8: Happy birthday, pal!	Chandler	Ross	1
9: Hey dad, what's up?	Monica	Mr. Geller	-1
	Speaker	2nd party	Intense (R)
10: Um, has uh Dr. Franzblau been by?	Rachel	Dr. Franzblau	-1
11: There's a beautiful woman at eight, nine, ten o'clock!	Chandler	Aurora	-1
12: Uh, Rach, it's the Visa card people.	Monica	Rachel	1

Table 3: Annotation examples. We show examples of contrasting values for selected dimensions. The first party is always the speaker, and the second party is underlined. I stands for interactions, and R for relationship.

		raw	κ
Interaction	Cooperative	93.4%	0.82
	Active	95.4%	0.89
	Concurrent	93.7%	0.84
	Spat. Near	95.7%	0.87
Relationship	Equal	94.6%	0.85
	Intense	88.8%	0.80
	Pleasure Or.	97.4%	0.77
	Intimate	92.3%	0.85
	Temporary	94.8%	0.78

Table 4: Inter-annotator agreement (raw agreement and Cohen's κ). κ values between 0.6 and 0.8 indicate *substantial* agreement, κ values over 0.8 indicate *perfect* agreement (Artstein and Poesio, 2008).

We discovered during pilot annotations that the value for a dimension sometimes cannot be determined. For example, if the first interaction between *Rachel* and *Monica* is *Rachel: How are [you]$_{\text{Monica}}$ doing?*, we cannot tell if the relationship is *temporary* or *enduring*. We note, however, that all interaction after we find out that they are best friends (as long as they remain best friends) will be annotated *enduring*. Hereafter, we refer to dimensions by the first descriptor in Table 1, and use `1` if the first descriptor of a dimension is true, `-1` if the second descriptor is true, and `0` if neither the first nor the second descriptor can be chosen.

Annotation Quality. The annotations were done by two graduate students in computational linguistics. First, they did pilot annotations to better define the dimensions (Section 3). After several iterations, both of them annotated 3 episodes (15% of all interactions). Table 4 presents the inter-annotator agreements. Cohen's κ range between 0.77 and 0.89, and most (7 out of 9) are above 0.80, which is considered perfect agreement. Values between 0.60 and 0.80 are considered substantial (Artstein and Poesio, 2008). The remaining episodes were annotated once.

4.3 Annotation Examples

We present annotation examples in Table 3. The interactions in conversation turns 1 and 2 are *competitive*: Phoebe is ridiculing Paul by asking Monica if he eats chalk, and Ross is confronting an unnamed woman. In turn (3), Monica refers to Phoebe with affection (as the latter sleeps), thus the interaction is *cooperative*. Turns (4–6) exemplify *concurrent* vs. *nonconcurrent*. In (4), Rachel is inquiring whether she can meet Alan (Monica's boyfriend), and Rachel and Alan are not involved in the same event (at this point, the meeting may or may not happen). In (5–6), however the speaker and second party are involved directly in a communication event. In examples (4–6), the values for *active* are the same as for *concurrent*.

Turns 7–12 present examples for dimensions of relationships. Examples 7 and 9 are fairly straightforward: previous interactions reveal that *Joey* is

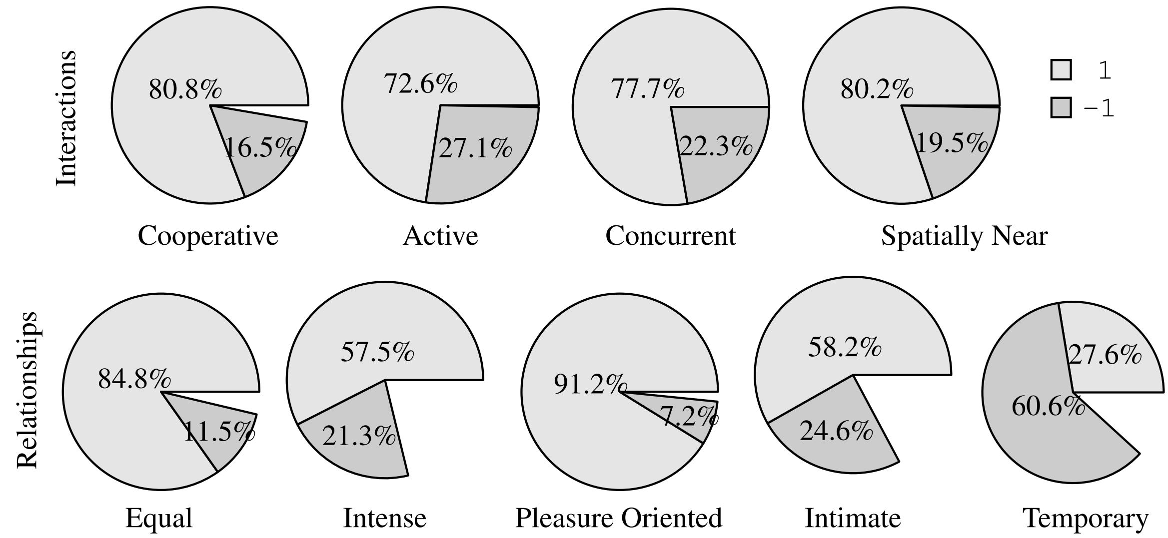

Figure 1: Label percentages per dimension. The missing portion of pie charts correspond to label `0` (unknown).

an adult and Monica's father is indeed *Mr. Geller*. Example 8 is annotated *equal*, as Chandler and Ross are friends based on previous interactions (the use of *pal* also helps). Examples 10–12 require more explanation, as additional information beyond the current conversation turn is required (recall that dimensions of relationships are annotated taking into account the previous turns within the same episode, Section 4.2). Dr. Franzblau is the doctor of a friend's ex-wife, so Monica and him have a *superficial* relationship (Turn 10). At the point Turn (11) is spoken by Chandler, she and *Aurora* are strangers, so they have a *superficial* relationship. In (12), previous conversations reveal that Monica and Rachel are close friends, and they interact often (*intense*).

5 Corpus Analysis

The pie charts in Figure 1 present the label distributions per dimension. Regarding interactions, we note that (a) values for all dimensions can be determined almost always (the percentages of `0` (unknown) are almost zero), and (b) the first descriptor is much more common in all dimensions. These percentages do not represent the distribution of interactions between people in general: the scripts of the TV show Friends mostly contain conversation between friends. Regarding relationships, we observe a larger percentage of `0` (unknown) although values of all dimension can be determined most of the time (labels `1` and `-1`, indicating that the first or second descriptor apply). Most dimensions are biased towards `1` (the only exception is *temporary*, as most relationships are *enduring*), especially *pleasure oriented* and *equal* (91.2% and 84.8%). Again, these distributions would be different if we worked with other sources of dialogue than the TV show Friends.

	Cooperative (I)	Active (I)	Concurrent (I)	Spat. Near (I)	Equal (R)	Intense (R)	Pleasure Or. (R)	Intimate (R)
Active	.06	–						
Conc.	.02	.81	–					
Near	.01	.71	.89	–				
Equal	-.01	.08	.06	.05	–			
Intense	.08	.28	.24	.22	.28	–		
Pleasure	-.05	.01	.08	.08	.50	.13	–	
Intimate	.12	.28	.26	.22	.32	.75	.28	–
Temp.	-.02	-.33	-.31	-.28	-.24	-.52	-.27	-.54

Table 5: Pearson correlations between pairs of dimensions of interactions and dimensions.

Many of the dimensions we consider in this work are intuitively correlated. For example, *concurrent* interactions must be *active*, and *pleasure oriented* interactions are probably also *equal*. We note, however, that interactions can be *passive* and *concurrent*, e.g., in *(Monica talking to Joey) [He]*$_{\text{Paul}}$ *is just [a guy]*$_{\text{Paul}}$ *I am dating!*, Monica and Paul have a *passive* and *concurrent* interaction (they are dating, but they are not talking to each other). Table 5 shows Pearson correlations between all dimensions of interactions and relationships. Most correlations are under 0.3 (29 out

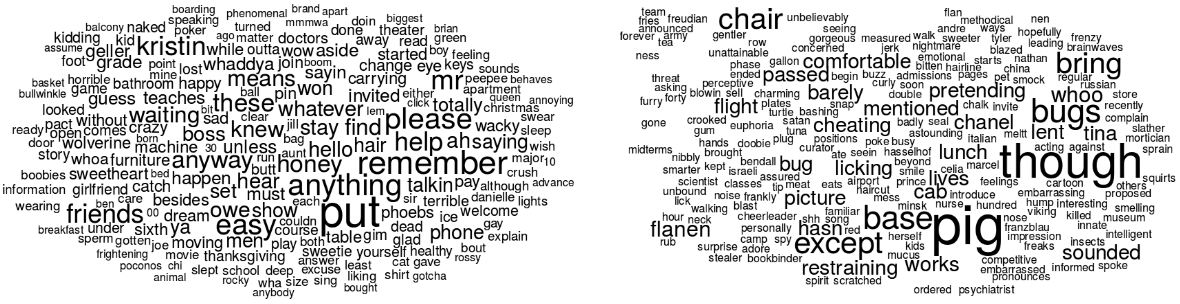

Spatially Near vs. Spatially Distant (Interaction)

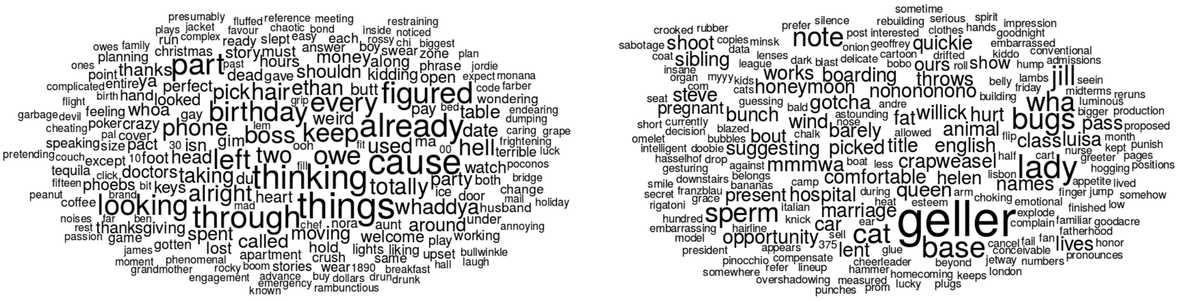

Intense vs. Superficial (Relationship)

Figure 2: Most salient words (calculated with tf-idf) for dimensions *spatially near* and *intense*.

of 36), although some pairs do have high correlations. In particular, *active* interactions tend to be both *concurrent* (0.81) and *spatially near* (0.71), and *spatially near* interactions tend to be concurrent (0.89). Regarding relationships, *intimate* correlates with *intense* (0.75), *pleasure oriented* with *equal* (0.50), and *temporary* with both *superficial* (0.52) and *intimate* (0.54).

Finally, Figure 2 shows the most salient words of dimensions *spatially near* and *intense*. We calculated salience using tf-idf (Schütze et al., 2008). Interactions containing derogatory words (e.g., *pig*, *bugs*, *pretending*, *cheating*) tend to be *distant*, and *near* interactions contain mostly neutral and nicer words such as *friends*, *sweatheart* and *please*. We also note that cognitive verbs and nouns (e.g., *thinking*, *figured (out)*, *looking (into)*, *cause*), as well as important events (*birthday*, *thanksgiving*) and slang usage (e.g., *whaddya*) signal *intense* relationships.

6 Experiments and Results

We experimented with SVM classifiers with RBF kernel to predict dimensions of interactions and relationships. We divided the 24 episodes into train (episodes 1–20) and test (21–24), and trained one classifier per dimension using scikit-learn (Pedregosa et al., 2011). Each classifier is trained with three labels: `1` (1st descriptor), `-1` (2nd descriptor) and `0` (unknown). The SVM parameters (C and γ) were tuned using 10-fold cross-validation with the train split, and results are reported using the test split.

		P	R	F
Majority Baseline		0.54	0.73	0.62
SVM	first word	0.62	0.71	0.65
	BOW current	0.66	0.73	0.67
	+ sentiment	0.65	0.72	0.66
	+ other	0.71	0.75	0.70
	+ BOW previous	0.73	0.76	0.72

Table 6: Results obtained with the test set with several systems (average of all dimensions). *Previous* refers to the previous conversation in which the same pair of people interacted not the immediately previous turn).

Note that different pairs of people interact more or less in each episode (Table 2). Thus, the classifiers are grounded on general language usage and not modeling who talks and who is talked about. We also experimented with LSTMs taking as input the current conversation turn and previous turns, but do not report results because SVM classifiers yielded better results.

		`1` (1st descriptor)			`0` (unknown)			`-1` (2nd descriptor)			All		
		P	R	F	P	R	F	P	R	F	P	R	F
Interaction	Cooperative	0.83	0.96	0.89	0.00	0.00	0.00	0.31	0.08	0.13	0.73	0.80	0.75
	Active	0.92	0.90	0.92	n/a	n/a	n/a	0.75	0.76	0.75	0.87	0.87	0.87
	Concurrent	0.92	0.92	0.92	n/a	n/a	n/a	0.70	0.69	0.70	0.87	0.87	0.87
	Spat. Near	0.89	0.94	0.91	n/a	n/a	n/a	0.69	0.53	0.60	0.85	0.86	0.85
Relationship	Equal	0.86	0.95	0.90	0.00	0.00	0.00	0.15	0.07	0.10	0.76	0.83	0.79
	Intense	0.70	0.81	0.75	0.28	0.11	0.16	0.38	0.46	0.41	0.56	0.60	0.57
	Pleasure Or.	0.82	1.00	0.90	0.00	0.00	0.00	1.00	0.02	0.04	0.82	0.83	0.75
	Intimate	0.61	0.85	0.71	0.26	0.05	0.09	0.42	0.36	0.39	0.48	0.56	0.49
	Temporary	0.53	0.42	0.47	0.67	0.17	0.27	0.62	0.85	0.72	0.60	0.60	0.57

Table 7: Results obtained per dimension with the best system (all features, Table 6). The results under *All* the weighted averages for all labels, recall that the label distribution is biased (Figure 1).

Feature Set. We use a combination of features extracted directly from the conversation turn, sentiment lexica and context. Specifically, we extract (a) the first word in the conversation turn, (b) bag-of-words features (binary flags and tf-idf scores), and (c) the root verb, and flags indicating the presence of exclamation, question marks and negation cues from (Morante and Daelemans, 2012) (other). Regarding sentiment, we extract flags indicating whether the turn has a positive, negative or neutral word in the list by Hamilton et al. (2016), the sentiment score of the turn (summation of sentiment scores per token over number of tokens in the turn), and a flag indicating whether the turn contains a negative word from the list by Hu and Liu (2004). Regarding context, we extract bag-of-words features from the previous conversation turn in which the same people interact (not necessarily the preceding turn).

6.1 Results

Table 6 shows the overall results (average of all dimensions) obtained with the majority baseline and several feature combinations. All feature combinations outperform the baseline. Sentiment features are not beneficial, leading to the conclusion that sentiment does not correlate with dimensions of interactions and relationships between people. This may look surprising at first sight, but recall that our dimensions capture much more than if two people get along (Table 1). Finally the bag-of-words features from the previous turn in which the same people interacted bring a small improvement (F: 0.70 vs. 0.72).

We show results per dimension for the best feature combination (all) in Table 7. Despite the label distributions are biased (Figure 1), the system predicts most labels for most dimensions except the very biased ones (*cooperative*, *equal* and *pleasure oriented*). Note that `0` (unknown) does not allow us to determine the value of a dimension, and the low results with this label are not a concern.

7 Conclusions

This paper presents the task of characterizing interactions and relationships between people. We work with dialogue transcripts, and define an *interaction* as a speaker referring to somebody else, and a *relationship* as a sequence of one or more interactions. Unlike previous work (Section 2), we target all interactions and relationships, and use dimensions that are applicable to any interaction or relationship regardless of the underlying type (e.g., SIBLINGS, FRIENDS, DOCTOR-PATIENT).

We have presented an annotation effort on 24 episodes of the popular TV show Friends (Season 1). The total number of conversation turns is 9,168, and the total number of interactions is 2,331. The label distribution per dimension shows that the labels are unbalanced, but a relatively straightforward SVM is able to outperform the majority baseline (F: 0.62 vs. 0.72, Table 6). Features extracted using well-known sentiment lexica yield no improvements, leading to the conclusion that the dimensions we work with capture information beyond whether two people get along.

Crucially, values for the dimensions we work with can be determined most of the time (Figure 1, labels `1` and `-1`). Since we do not impose any restriction on the interactions or relationships we work with, we conclude that these dimensions may be universally applicable.

References

John Adamopoulos. 2012. The emergence of social meaning: A theory of action construal. In *Handbook of Social Resource Theory*, pages 255–272. Springer.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Comput. Linguist.*, 34(4):555–596.

David B. Bracewell, Marc T. Tomlinson, Mary Brunson, Jesse Plymale, Jiajun Bracewell, and Daniel Boerger. 2012. Annotation of adversarial and collegial social actions in discourse. In *Proceedings of the Sixth Linguistic Annotation Workshop*, LAW VI '12, pages 184–192, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philip Bramsen, Martha Escobar-Molano, Ami Patel, and Rafael Alonso. 2011. Extracting social power relationships from natural language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 773–782, Portland, Oregon, USA. Association for Computational Linguistics.

Yu-Hsin Chen and Jinho D. Choi. 2016. Character identification on multiparty conversation: Identifying mentions of characters in tv shows. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 90–100. Association for Computational Linguistics.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 699–708, New York, NY, USA. ACM.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 250–259, Sofia, Bulgaria. Association for Computational Linguistics.

Morton Deutsch. 2011. Interdependence and psychological orientation. In *Conflict, Interdependence, and Justice*, pages 247–271. Springer.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ace) program tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC-2004)*, Lisbon, Portugal. European Language Resources Association (ELRA). ACL Anthology Identifier: L04-1011.

Raymond W Gibbs. 2000. Irony in talk among friends. *Metaphor and symbol*, 15(1-2):5–27.

Eric Gilbert. 2012. Phrases that signal workplace hierarchy. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1037–1046, New York, NY, USA. ACM.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 466–471, Stroudsburg, PA, USA. Association for Computational Linguistics.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Austin, Texas. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544, San Diego, California. Association for Computational Linguistics.

Valentin Jijkoun, Maarten de Rijke, and Jori Mur. 2004. Information extraction for question answering: Improving recall through syntactic patterns. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Harold H Kelley. 2013. *Personal relationships: Their structures and processes*. Psychology Press.

Seth Kulick, Ann Bies, and Justin Mott. 2014. Inter-annotator agreement for ere annotation. In *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 21–25, Baltimore, Maryland, USA. Association for Computational Linguistics.

Roser Morante and Walter Daelemans. 2012. Conandoyle-neg: Annotation of negation cues and their scope in conan doyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 1563–1568, Istanbul, Turkey. European Language Resources Association (ELRA). ACL Anthology Identifier: L12-1077.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466, San Diego, California. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Vinodkumar Prabhakaran and Owen Rambow. 2014. Predicting power relations between participants in written dialog from a single thread. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 339–344, Baltimore, Maryland. Association for Computational Linguistics.

Farzana Rashid and Eduardo Blanco. 2017. Dimensions of interpersonal relationships: Corpus and experiments. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2297–2306, Copenhagen, Denmark. Association for Computational Linguistics.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press.

Mark Snyder and Arthur A Stukas Jr. 1999. Interpersonal processes: The interplay of cognitive, motivational, and behavioral activities in social interaction. *Annual review of psychology*, 50(1):273–303.

Mihai Surdeanu and Ji Heng. 2014. Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In *Proceedings of the TAC-KBP 2014 Workshop*.

Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. 2015. Learning to explain entity relationships in knowledge graphs. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 564–574, Beijing, China. Association for Computational Linguistics.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 515–526, New York, NY, USA. ACM.

Michael White, Tanya Korelsky, Claire Cardie, Vincent Ng, David Pierce, and Kiri Wagstaff. 2001. Multidocument summarization via information extraction. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Myron Wish, Morton Deutsch, and Susan J Kaplan. 1976. Perceived dimensions of interpersonal relations. *Journal of Personality and Social Psychology*, 33(4):409.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1399–1407, Stroudsburg, PA, USA. Association for Computational Linguistics.

Why Swear? Analyzing and Inferring the Intentions of Vulgar Expressions

Eric Holgate[†] **Isabel Cachola**[‡] **Daniel Preoţiuc-Pietro**[◇] **Junyi Jessy Li**[†]
[†]Department of Linguistics, [‡]Department of Mathematics,
The University of Texas at Austin
`{holgate@,isabelcachola@,jessy@austin.}utexas.edu`
[◇]Bloomberg LP
`dpreotiucpie@bloomberg.net`

Abstract

Vulgar words are employed in language use for several different functions, ranging from expressing aggression to signaling group identity or the informality of the communication. This versatility of usage of a restricted set of words is challenging for downstream applications and has yet to be studied quantitatively or using natural language processing techniques. We introduce a novel data set of 7,800 tweets from users with known demographic traits where all instances of vulgar words are annotated with one of the six categories of vulgar word use. Using this data set, we present the first analysis of the pragmatic aspects of vulgarity and how they relate to social factors. We build a model able to predict the category of a vulgar word based on the immediate context it appears in with 67.4 macro F1 across six classes. Finally, we demonstrate the utility of modeling the type of vulgar word use in context by using this information to achieve state-of-the-art performance in hate speech detection on a benchmark data set.

1 Introduction

Vulgarity is a common element of conversation (Jay, 2009; Mehl et al., 2007) and is used even more frequently in social networks such as Twitter (Wang et al., 2014). Understanding the motivation behind the choice to be vulgar and the way in which vulgarity is manifested in naturally occurring environments is of interdisciplinary interest. Pragmatic functions that dictate patterns of vulgarity usages may interact with speaker cultural background and demographics. This makes them appealing—and challenging—to model in NLP applications

Yet, to date, there has been no empirical study on the type of vulgar word usage. Research in linguistics and psychology has identified several types of usage for vulgar words (Andersson and Trudgill, 1990; Pinker, 2007; Wang, 2013). These range from use as an intensifier for an opinion or emotion, to offend others, or simply as a way of speaking or to signal the level of (in)formality in a conversation (Pinker, 2007). Table 1 shows example tweets with the six general functions of vulgar word usage.

Function	**Tweet**
Express aggression	<USER> You are an **ass** Your industry is full of assholes and you do nothing to improve (...)
Express emotion	There are so many things I want to do, But investing in equipment is a pain in the **ass**
Emphasise	today is a good **ass** day <URL>
Auxiliary	Wish <USER> could save my **ass** on these exams like he used to
Signal Group Identity	Now this is a group of **ass** kickers!
Non-vulgar	Kick **Ass** 2 - Red Band Trailer <URL>

Table 1: Examples of tweets containing the vulgar word **ass** with six different functions.

We notice that in one of the examples, the vulgar word *ass* is used to verbally abuse another user, while the same word can also be employed to emphasize a feeling ('good *ass* day') or to express an emotion ('pain in the *ass*'). Hence, explicitly modeling vulgar words use is expected to positively impact the performance of practical tasks such as hate speech detection or the way in which profanity filtering is performed.

The goal of our study is to present a comprehensive and multi-faceted analysis of the types of vulgar word usage. To this end, this paper presents:

1. The first data set of written utterances that contain vulgar words, where each vulgar word is labeled for one of six functions of use[1]
2. A quantitative analysis of vulgar word usage across different user demographic traits
3. A machine learning approach to predicting one of six types of vulgar word usage from context
4. Experiments demonstrating that modeling the type of vulgar word usage in context can im-

---

[1] `https://github.com/ericholgate/VulgarFunctionsTwitter`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4405–4414
Brussels, Belgium, October 31 - November 4, 2018.

prove predictive performance of the hate speech prediction task on a benchmark data set

Our novel data set contains 7,800 tweets with 8,524 vulgar word labels annotated for one of six functions by seven annotators. We find that the way in which vulgarity is used interacts with user demographic variables such as age or political ideology. We then build a model for predicting the usage type of each vulgar word in the tweet using the tweet context. Finally, we explicitly model the vulgar word usage type in the task of hate speech detection to discriminate between hate speech and tweets including profanity, demonstrating an improvement in predictive accuracy of 3.7 F1. This demonstrates that using insights into vulgar word usage developed in linguistics and psychology, we can achieve quantitative improvements on downstream NLP applications and inform the way models are built and tailored to the task.

2 Related Work

Vulgar language and its uses and pragmatic functions have been studied in several linguistic and psychological studies and the phenomenon has many names. In this paper, we will use *vulgarity*, *profanity*, and *swear/curse words* interchangeably.

Vulgar words were found to be very versatile, with a vulgar word being able to perform different interpersonal functions according to different contexts. Four types of usage are identified in Andersson and Trudgill (1990), including abusive (intended to harm the hearer), expletive (used to express emotions; not directed towards others), humorous (looks like abusive swearing, but has the opposite function) and auxiliary (swearing as a way of speaking, often or always non-emphatic). The five functions of swear words suggested in Pinker (2007) are: dysphemistic (conveying negative sentiment), idiomatic (signaling informality or simply used as a manner of speaking), abusive (intending to offend or harm), emphatic (intending to stress a claim or intensify emotive content) and cathartic (communicating pain). Finally, Wang (2013) identifies four pragmatic roles for profanity with a considerable degree of overlap with Pinker: emoting, emphasizing, aggressing, and group identity signaling.

For the scope of this study, we aim to cover all the functions identified by past research that can be identified from text with restricted content and context such as tweets. Thus, we dropped the cathartic function from Pinker (2007), which is an instantaneous reaction more specific to speech to relieve the effect of physical pain (Stephens et al., 2009). This would thus be very rarely – if ever – expressed through social media and would be very hard to annotate with textual content alone while lacking the broader context of its utterance. We also considered the abuse and aggression functions as equivalent across categorizations, as they imply a face-threatening act (Brown and Levinson, 1987). We considered the auxiliary and idiomatic categories as equivalent across classes, but maintained signaling group identity as in Wang (2013). We also created a non-vulgar classification in case the vulgar word is used in a non-vulgar context (e.g., a name that doubles as a vulgar term).

Due to their affective impact, vulgar words are often used as *strange* synonyms by substituting each other in context or idioms, even when they have no affinity in syntax or meaning (Quang, 1971; Pinker, 2007) (e.g., for God's sake – for fuck's sake; 'I don't give a damn/fuck/shit). This heavily contributes to vulgar word volatility across different functions and higher ambiguity in context. However, this type of usage can allow computational approaches that model the immediate context around a word to generalize across words to functions. To date, there has been no research on quantitatively modeling the **function** of vulgar words in context.

The overall frequency of usage of vulgar words has been quantitatively studied in social media and online communities. For example, Wang et al. (2014) estimates that vulgar posts constitute upwards of 1.15% of tweets and examines vulgar token frequency and how it varies with time, geolocation, and gender. An analysis of profanity across gender and age also appears in Gauthier et al. (2015).

In fact, gender is the most studied sociodemographic factor in relation to the use of profanity. Many studies have shown that male-identifying users employ vulgar terms more frequently than female-identifying users (e.g. Selnow, 1985; Wang et al., 2014).

Jay and Janschewitz (2008) demonstrate that profanity is moderated by pragmatic or contextual factors that go beyond gender, including occupation, social status, and even the nature of the relationship between interlocutors – though this last point, proves difficult to explore via Twitter where the identity of the audience is at least partially obfuscated. Other social factors such as age, religiosity or social status have also been shown to vary with vulgar frequency (McEnery, 2004), as has political ideology (Sylwester and Purver, 2015; Preoţiuc-Pietro et al., 2017). It is thus likely that sociodemographic factors also influence the functions with which vulgar words are used.

This study expands the scope of this type of research, by going beyond simple frequency of usage to pragmatic functions of vulgar words and how they are used differently by different sociodemographic groups.

3 Data

We use social media as our data source as this contains a high level of expression of thoughts, opinions and emotions (Java et al., 2007; Kouloumpis et al., 2011) and represents a platform for observing written interactions and conversations between users (Ritter et al., 2010).

Social media and Twitter in particular provide vast volumes of text which are more informal and less curated compared to other domains such as newswire. An additional advantage of Twitter data is that it allows us to study the sociodemographic context.

3.1 Identifying Vulgar Tweets

We use the corpus of tweets utilized to construct the Vulgar Twitter corpus introduced in prior work (Cachola et al., 2018). Every tweet in this corpus contains at least one vulgar term. We then annotate each instance of a vulgar token for type of use. Note that we use the full dataset of 7,800 tweets which contains 1K more tweets[2] than the released version of the Vulgar Twitter corpus.

The Vulgar Twitter corpus was constructed by identifying tweets containing vulgarity through use of the vulgarity lexicon available at `www.noswearing.com`. A total of 82 tokens were removed from this list as they were deemed not to be unambiguously vulgar after manual inspection.[3] Regular expressions were utilized to identify common intentional spelling variations (e.g., vowel reduplication such as *fuuuuuck* or self-censorship such as *a$$*).

For the complete description of the composition and construction of the Vulgar Twitter corpus, we refer the interested reader to the original paper (Cachola et al., 2018).

3.2 Data Sampling

The Vulgar Twitter corpus overlaps with Preoţiuc-Pietro et al. (2017) which allows us to consider the relationship between sociodemographic features and vulgar functionality. The tweets are compiled from up to 3,200 most recent tweets (per Twitter Developer API) of 4,132 twitter users who provided sociodemographic information via self-report in an online survey (Preoţiuc-Pietro et al., 2017). This sociodemographic data has been utilized in our previous research Cachola et al. (2018); for a full description of its collection, we refer the reader to Preoţiuc-Pietro et al. (2017).

3.3 Demographic Variables and Coding

Demographic information (including gender, age, level of education, level of annual income, faith, an political ideology) was was self-reported via online survey. To control for cultural variation, data was only solicited from residents of the United States. All demographic variables are ordinal with the exception of gender, which is binary.

- **Gender:** a binary[4] variable (Female as 1; Male as 0).
- **Age:** an ordinal, integer-valued variable [13-90].
- **Income:** An ordinal variable [1-8]; the lowest level (1) stands for '< $20,000' and the highest (8) stands for '> $200,000'.
- **Education:** An ordinal variable [1-6]; the lowest level (1) stands for 'no high school degree' and the highest (6) stands for 'Advanced Degree (e.g., Ph.D.)'.
- **Political ideology:** an ordinal variable on the liberal-conservative spectrum (a common form of representation of US political ideology (Ellis and Stimson, 2012)). Reporting options ranged from 'Very Conservative'

---

[2] These tweets are excluded due to low sentiment agreement in Cachola et al. (2018).

[3] These terms were largely anatomical words or general verbs like *penis*, *vagina*, and *blow*, but some identity descriptors like *gay*, *queer*, and *lesbian* were also excluded after manual review of a large sampling of uses revealed they were not overwhelmingly employed as slurs.

[4] Users were asked to identify their gender as Male, Female, or via an open field. Users who did not respond as either Female or Male were excluded from data collection as there was not a sufficient population to be confident that data would be representative.

Function	**Definition**	**Freq.**
Express aggression (**Agr**)	The word is used in order to harm the person or group the tweet is about.	15.2%
Express emotion (**Emo**)	The word is used to express emotions (positive or negative) related to the users internal states, exclamations, feelings or attitudes towards an object. If removing the vulgar term, the expressed emotion is lacking.	24.8%
Emphasise (**Emp**)	The word is used to emphasize a statement or feeling.	29.8%
Auxiliary (**Aux**)	The use of this word is simply a manner of speaking and does not fit any of the above descriptions. Descriptions of external emotions (those of someone else) fall into this category.	17.0%
Signal Group Identity (**Sig**)	This word is used as a marker of identity in a specific social group.	4.7%
Non-vulgar Use (**Non**)	The use of this word is not vulgar (e.g., named entities that involve vulgar words).	8.2%

Table 2: Functions of vulgar words, their definition presented to the annotators and their frequency in the final data set.

	Agr	**Emo**	**Emp**	**Aux**	**Sig**	**Non**
Agr	0.63	0.11	0.09	0.07	0.10	0.01
Emo	0.07	0.59	0.20	0.13	0.01	0.01
Emp	0.04	0.18	0.68	0.07	0.01	0.02
Aux	0.07	0.16	0.15	0.56	0.03	0.03
Sig	0.17	0.06	0.07	0.11	0.57	0.02
Non	0.02	0.04	0.04	0.10	0.02	0.77

Table 3: Confusion matrix between aggregated function (row) and individual annotations (column), normalized by row.

(1) to ‘Moderate’ (4) to ’Very Liberal’ (7). Two additional responses, ‘Other’ (8) and ‘Apathetic’(9) were included to cover the full breadth of the ideological spectrum, but users selecting these options were excluded from our dataset (1,290 in total) in order to maintain an ordinal scale.

- **Faith:** an ordinal variable [1-6]; users were asked to report the average number of religious services attended. Available responses ranged from ‘Never’ (1) to ‘Multiple times per week’ (6).

3.4 Data Processing

We follow the same preprocessing procedure as in Cachola et al. (2018). URL’s and usernames are replaced by <URL> and <USER> tokens respectively to protect user privacy. Punctuation is then removed and all words are lowercased.

3.5 Annotation

We have collected annotations via Amazon Mechanical Turk (MTurk) for vulgar word usage type for 8,524 instances of vulgar words across the 7,800 tweets present in the Vulgar Twitter corpus (Cachola et al., 2018).

The task guidelines follow previous research from linguistics and psychology described in Section 2. For generality, we use a union of the different classes proposed and grouped classes where it was possible. The final guidelines include six different functions of vulgar words described in Table 2.

For quality control, we asserted the following qualifications on MTurk: locale=US, approval rate >90%, number of HITs approved >100. Further, we removed all ratings from users that have a Cohen’s Kappa of lower than 0.2 when compared to the majority rating of the other six annotations resulting in the removal of 8,430 ratings (14% of the total number) from 150 out of 663 users. These users were banned and annotations were recollected until 7 ratings were obtained for all instances.

We measured inter-annotator agreement using Krippendorf’s Alpha as this can handle cases where each item was labeled by different groups of users. The overall Krippendorfs Alpha is 0.506 despite there being a large number of classes (6). This alpha value (0.506) is regarded as a moderate level of agreement (Artstein and Poesio, 2008). To reduce uncertainty, we aggregate our labels across seven different annotators. In cases where no majority class emerged from the seven annotations (10.6% of the instances), the tie was broken by one of the authors of the paper, who have significant training and experience in linguistic annotation.

The distribution of the final vulgar word type is presented in the last column from Table 2. Table 3 shows the confusion matrix between aggregated function (row) and individual annotations (column); each cell is normalized by the row sum. Some patterns in disagreements include: (1) Vulgar words used to signal group identity are sometimes confused with aggression as annotating these may require additional social context about the user (e.g. if they are female, African-American, etc.) or about social relationships (e.g. *“lmao yeah cause a bitch can’t sing”*). (2) Emotion confused with auxiliary usage in idioms or where there is a lack on context about what is

Word	Rank	Entropy	Agr	Emo	Emp	Aux	Sig	Non
bitchy	35	1.547	3	5	2	3	2	0
dicks	33	1.442	7	2	0	6	2	2
bastard	26	1.307	18	3	1	7	1	3
fuck	4	1.272	246	345	190	75	0	0
pussy	21	1.256	17	3	2	33	1	7
ass	5	1.250	116	45	222	352	7	5
hell	2	1.220	16	242	602	71	0	238
dick	9	1.208	36	4	5	87	0	67
bitch	7	1.194	296	23	20	60	110	3
shit	1	1.170	59	555	200	488	1	1

Table 4: Top vulgar words sorted by entropy. Higher entropy indicates a more evenly distributed usage is across functions (maximum entropy over six values = 1.791, minimum entropy over all functions = 0). Rank represents the rank of the word by frequency in the data set.

the author's intent or target (e.g. *"Stop cryin.. Damn you got the foul"*). (3) Auxiliary use of vulgar words in an emotional tweet (e.g, *"ok knicks. we're winning. dont fuck it up."*). (4) Short tweets lacking context drive confusion about the target of vulgarity or if an emotion is expressed (e.g. *"Fuck yeah"*).

4 Analysis

We start with a quantitative analysis of our data. First, we examine the extent to which the same vulgar word is used for different functions. Then, we identify if sociodemographic factors impact the functions with which vulgar words are used.

4.1 Vulgar Word Analysis

To quantitatively measure which vulgar words are most used with different functions, we first compute its distribution over the six functions in our entire data set. Then, we compute the entropy as a measure of how evenly distributed the distribution over functions of each word is. To avoid uncertainly associated computing statistics over distributions with low counts, we keep only words that appear more than 10 times in our data set (43 words) after collapsing variants of the same word (e.g. fuck – f*ck – fuuuck).

The average entropy of all vulgar words is $\mu = 0.835$ ($\sigma = 0.36$), with 0 being the minimum entropy (i.e., all words are used with one function) and 1.791 being the maximum entropy (i.e., all words are used with the same frequency with all six functions). The words with the highest entropy are presented in Table 4.

The table shows that four of the most frequent five words are in the top ten words by entropy, with all of them having significant numbers of occurrences in at least three vulgar functions. Actually, the average entropy of words used at least 100 times in our data set (15 words) is 0.930 compared to 0.835 for words used at least 10 times.

We see that all words in the table are used significantly with three or more functions. On average, in the entire data set, each word is used at least once with $\mu = 4.00$ functions ($\sigma = 1.34$).

This highlights both the challenges in modeling vulgar word functions and the opportunity of using the function to improve practical applications.

In contrast, Table 5 shows the vulgar words which are most likely to be used with each of the six functions.

4.2 Demographic Analysis

Sociodemographic factors may impact the distribution with which each function of vulgar words is used. To measure this, we compute for each user a normalized distribution over the functions of vulgar words used in our data set. Then, we compute Pearson correlation where the dependent variable is the fraction of each vulgar word function and the independent variables are the user sociodemographic trait values. Following previous work (Schwartz and et al., 2013; Preoţiuc-Pietro et al., 2017), for all analyses we consider gender and age basic traits and control for potential data skew by introducing both variables as controls in partial correlation. When studying age and gender, we use the other trait as the control. Since we are running 36 tests at once without pre-stated hypotheses, we correct the correlations for multiple comparisons using Bonferroni correction. Results of these analyses are presented in Table 6.

The results show several vulgar word functions are specific of age. Younger users of Twitter are more likely to use vulgar words to signal group identity and to express emotion. Older age is more likely to be related to use of words that are vulgar with non-vulgar functions. These correlations show that there are differences in how younger generations are using vulgar words, even if tweets were posted in the same time interval, signaling a diachronic change in usage.

The analysis shows that the only significant correlation with the other five demographic variables is between both political ideology and faith and using vulgar words for emphasis and in non-vulgar functions. Liberals are more likely to use vulgar words for emphasis and less likely to use them with non-vulgar functions. Previous research

Aggression		Express Emotion		Emphasis		Auxiliary		Signal Group Identity		Non-Vulgar	
Word	**Freq**	**Word**	**Freq**	**Word**	**Freq**	**Word**	**Freq**	**Word**	**Freq**	**Word**	**Freq**
cunt	86.9%	pissed	84.4%	fucking	84.7%	asses	73.9%	bitches	88.9%	mick	100%
asshole	86.3%	bullshit	64.2%	fuckin	84.0%	shitting	69.2%	nigga	85.7%	cracker	97.5%
asshit	83.0%	fucked	61.3%	goddamn	70.0%	arse	69.2%	slut	26.0%	dyke	92.8%
faggot	81.8%	shitty	52.6%	damn	62.3%	cock	62.9%	whore	25.0%	coon	92.8%
fag	73.3%	shit	42.5%	hell	51.4%	pussy	52.3%	hoe	23.8%	ho	88.3%

Table 5: Vulgar words most used with each of the six functions.

Trait	Agr	Emo	Emp	Aux	Sig	Non
Gender	.011	-.005	.004	-.044	.051	.011
Age	-.013	-.085*	-.046	-.036	-.100**	.227**
Education	-.030	.009	-.006	-.007	-.032	.027
Income	.037	.002	.027	-.035	-.045	.032
Faith	-.031	-.047	-.112**	-.066	.014	.224**
Political Ideology	.009	.050	.092**	-.022	.003	-.124**

Table 6: Pearson correlation between user demographic traits and usage of the different functions of vulgar words. All correlations are significant at (*) $p < .05$, (**) $p < .01$, two-tailed t-test, **Bonferroni corrected** for multiple comparisons. Results for education, income and religiosity are controlled for age and gender.

showed that liberals are more likely to use more vulgarity overall in social media (Sylwester and Purver, 2015; Preoţiuc-Pietro et al., 2017) and are perceived by others to use more frequently than they do vulgar words (Carpenter et al., 2016), but this analysis shows this is especially due to vulgar word use to emphasise. The results are reversed for faith, which is known to be strongly correlated to conservative political ideology. Controlling for faith and political ideology with partial correlation does not alter the significance of this result.

Intriguingly, all other traits (gender, education and income) are not significantly correlated with an increased usage in any of the functions.

The vulgar word functions of aggression and auxiliary usage, which are more standard and traditional usages of vulgar words, do not show any significant differences with any sociodemographic trait.

5 Modeling Vulgar Word Use

The previous section showed that the same vulgar words can be used with several different functions. In this section, we use machine learning approaches to explicitly predict the function of a vulgar word given the tweet it appears in as context.

5.1 Method

We use logistic regression[5] to build six one vs. all binary classifiers for each of the six functions using information from the immediate lexical and syntactic context surrounding the word and general usage of the word in training data.

5.2 Features

We use the following feature types in our experiments:

Intention Distribution –We include six features encoding the distribution over intentional classes of the target word in training data, as some words use only several functions and some more predominantly than others.

Tweet Content –We derive a tweet-level representation of the entire content of the tweet by averaging vector representations of its constituent words. We utilize 200-dimensional GloVe embeddings pre-trained on 2B tweets (Pennington et al., 2014).

Sentiment Content –We include two features which represent the number of positive and negative valence words in the tweet, normalized by tweet length. For this feature group, we utilize the opinion lexicon introduced in Hu and Liu (2004).

Part of Speech Context –We encode the part of speech of the target word, the previous word and the next word as one-hot vectors as we expect syntactic information to be an indicator of different functions in context. We extract parts of speech using the Twitter version of the Stanford POS tagger which demonstrated good results on tagging tweets and uses the finer grained Penn Treebank tagset (Derczynski et al., 2013).

Brown Clusters –Finally, we include two one-hot feature groups which indicate the Brown Cluster (Brown et al., 1992) membership of word immediately before and immediately after the vulgar term.

---

[5] In preliminary experiments, we attempted to utilize a BiLSTM to encode tweet context, but it did significantly worse than the logistic regression model, possibly due to many parameters and classes compared to the size of the training data.

Method	Precision	Recall	F1
Most Frequent Class	5.05	16.6	7.76
All Features	**68.8**	**66.4**	**67.4**
– Intention Distribution	58.3	53.8	55.3
– Tweet Content	67.9	64.0	65.6
– Sentiment Content	68.6	66.3	67.3
– Part of Speech Context	67.8	64.6	65.9
– Brown Clusters	68.6	64.9	66.3

Table 7: Performance statistics for our baseline, predictive model with all features and with ablating each feature group. Precision, Recall and F1 score are macro-averaged across the six classes.

Brown Clusters are obtained by hierarchical clustering tokens based on contexts in which they immediately co-occur. We use the precomputed cluster representations as seen in Turian et al. (2010). We also experimented with Twitter-specific clusters (Owoputi et al., 2012), but found they did not perform as well on our development set.

Additionally, we experimented with a personal pronoun indicator feature in a three word window around the target, a one-hot lexical feature encoding the target vulgar item, and NRC emotion scores (Mohammad and Turney, 2013), but found there to be no improvement in performance as a result. We did not experiment with using the demographic variables as features as these are generally unavailable for use in predictive systems.

5.3 Experimental Results

We split our data into a training set of 6,883 tweets and a testing set of 1,087 tweets, and held out a set of 554 tweets as a validation set on which to test different hyperparameter settings.

Performance statistics for our predictive model are presented in Table 7, as well as ablation experiments for each feature group.

Our predictive model vastly outperforms the most frequent baseline, which uniformly selects the most frequent class overall (emphasis) and scores very low due to the very even distribution over functions. Our best model achieves a macro-averaged F1 score of 67.4 across the six classes.

In the ablation experiments, we see by removing one feature group at a time, which feature type adds most predictive value over others. Withholding the intention distribution feature group from the model shows the greatest loss in performance (12.1 macro F1). This is somewhat expected, as this feature gives a prior distribution over functions for the target word based on training data and, as most words are rarely used with some functions, allows the model to downweigh them. However, even with no word function prior, the predictive performance is still relatively high (55.3 macro F1 across six classes), showing that only the content and context is substantially predictive for the function of a vulgar word

Removing tweet content features or part-of-speech context introduce a similar drop in predictive performance, showing that the overall tweet content and the local syntactic context of the mention play complimentary roles in inference. The sentiment feature groups are the least informative, yielding an negligible increase in performance of only 0.1 macro F1.

The predictive model's F1 scores by class is as follows: *Aggression* – 65.6, *Emotion* – 62.4, *Emphasis* – 76.4, *Signal Group Identity* – 56.5, *Auxiliary* – 62.2, *Not Vulgar* – 81.4.

The highest predictive performance is obtained for vulgar words used in a non vulgar context, which is due to the different tweet content of these tweets and the restricted sets of words which are usually used as non-vulgar. The emphasis functions is the second most accurately predictable using our model, due to the very distinctive syntactic patterns of usage of this function (usually as an adjective). The least predictable function is signaling group identity. We observed that this function is usually used as part of larger conversational context and often relies on a shared social context.

6 Hate Speech Prediction

Finally, we aim to show that modeling the function of vulgar words explicitly has practical implications by using this in a downstream application.

6.1 Task

Automatic hate-speech detection on social media is the task defined as generally identifying abusive speech targeting specific group characteristics, such as ethnic origin, religion, gender, or sexual orientation (Warner and Hirschberg, 2012) with a clear intention to incite harm, or to promote hatred (Zhang and Luo, 2018). Several data sets and approaches to automatic hate speech detection have been recently proposed (Djuric et al., 2015; Burnap and Williams, 2015; Waseem and Hovy, 2016; Nobata et al., 2016; Davidson et al., 2017).

The task of predicting hate-speech is challenging for natural language processing using lexical information as it aims to predict the intention of the message and several words used in conveying hate speech can have other common uses (Davidson et al., 2017; Malmasi and Zampieri, 2018).

Hate speech is very often confused with offensive language, as highlighted in the error analyses of past hate speech detection papers (Davidson et al., 2017). Quantitative analysis of the machine learning models suggest that obscene words are very informative for both the hate speech and offensive classes of tweets (Malmasi and Zampieri, 2018), hinting that the functions of vulgar words usage are a major source of ambiguity.

Our hypothesis is that explicitly modeling the function a vulgar word has in context will benefit the hate speech prediction task, by differentiating between aggression and other usages.

6.2 Experiments

Data. We use the dataset introduced in Davidson et al. (2017) as this is publicly available, contains tweets collected using vulgar words and explicitly differentiates between offensive tweets and tweets containing hate speech. The three classes in this data set are hate speech, offensive, and neither.

Another public dataset on tweets, introduced in Waseem and Hovy (2016), focuses on specific forms of hate speech (sexist and racist), but is collected with a restricted set of keywords, has low coverage of vulgar words and does not explicitly distinguish between hate speech and other offensive language. Other datasets for this task are not publicly available e.g., Nobata et al. (2016).

Setup. In order to directly measure the impact on predicting performance introduced by explicitly modeling the function of the vulgar word in the tweet, we follow the same methodology to identify hate speech as described in Davidson et al. (2017), as implemented through the openly available code provided by the authors.[6] We thus train three one-vs-all logistic regression classifiers with L2 regularization as implemented in scikit-learn (Pedregosa et al., 2011). Features used in the model include unigram to trigram TF-IDF weighted word features, Part-of-Speech unigram to trigrams, reading level, sentiment words, Twitter specific features (e.g., hashtags, mentions, retweets, and URLs) as well as generic tweet-level features (e.g., number of characters, words, and syllables in each tweet).

	Method	
Class	Davidson et al.	+ vulgar features
Hate Speech	33.6	**39.7**
Offensive	92.1	**93.5**
Neither	82.2	**85.7**
Average	69.3	**73.0**

Table 8: F1 score per class and the macro average.

Vulgar Function Features. We directly and explicitly include the function of the vulgar word present in the tweet by introducing six new features to the hate speech detection model which represent the scores with which the vulgar word is associated with the six functions. If multiple vulgar words exist in a tweet, we use the average predictions over the six functions.

Metrics. We run the model from Davidson et al. (2017) using the provided code on 10-fold cross validation and report the average F1 score for each class as well as the macro-averaged F1 score across all ten folds. Using the available code, we could not reproduce exactly results presented in the Davidson et al. (2017) paper. For predicting the function of the vulgar words from context, we use our best predictive model described in Section 5. We also re-scale our six function features by multiplying them with a large exponent in order to make them significant in model training.

Results. As presented in Table 8, the addition of the six vulgar function features improves the F1 score for each of three classes to up to 6.1 F1 for the hate speech class, which had the lowest performance. This results in an improvement of the macro-F1 score for the entire classification task of 3.7 in F1. This demonstrates the importance of the proposed vulgar function modeling task in detecting hate speech.

7 Conclusion

This paper presents the first empirical study on the pragmatic functions of vulgar words. We created a novel, freely available data set of 7,800 vulgar tweets having 8,524 instances of vulgar words labeled for one of six functions by seven annotators and expert adjudication. We quantitatively showed, leveraging research in linguistics and psychology, that vulgar words are frequently used with different functions and, in the first quantitative analysis on this topic, uncovered that vulgar words are used with different functions by younger users to signal group identity and for expressing emotions.

We have built the first machine learning model

---

[6] `https://github.com/t-davidson/hate-speech-and-offensive-language`

for predicting vulgar word function from context, achieving a performance of 67.4 macro F1, demonstrating the practical feasibility of this task. We showed the usefulness of this task, by integrating predicted vulgar word function in the downstream task of hate speech detection, achieving an improvement of 3.7 in F1 on a benchmark data set.

This study showed that modeling pragmatic function is of practical importance. Future work will use this linguistic information to inform more complex machine learning models, e.g., deep neural networks, in an attempt to increase predictive gains. As two of the most used functions of vulgar words relate to expressing sentiment or emotions, we will also explore collecting sentiment annotations for joint sentiment and vulgar word function inference and use this to improve the task of sentiment analysis using multi-task methods.

References

Lars-Gunnar Andersson and Peter Trudgill. 1990. *Bad language*. Penguin.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Penelope Brown and Stephen C Levinson. 1987. *Politeness: Some Universals in Language Usage*, volume 4. Cambridge University Press.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Pete Burnap and Matthew L Williams. 2015. Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making. *Policy & Internet*, 7(2):223–242.

Isabel Cachola, Eric Holgate, Daniel Preoţiuc-Pietro, and Junyi Jessy Li. 2018. Expressively Vulgar: The Socio-Dynamics of Vulgarity and its Effects on Sentiment Analysis in Social Media. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING, pages 2927–2938.

Jordan Carpenter, Daniel Preoţiuc-Pietro, Lucie Flekova, Salvatore Giorgi, Courtney Hagan, Margaret Kern, Anneke Buffone, Lyle Ungar, and Martin Seligman. 2016. Real Men don't say 'cute': Using Automatic Language Analysis to Isolate Inaccurate Aspects of Stereotypes. *Social Psychological and Personality Science*, 8.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM, pages 512–515.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter Part-of-Speech Tagging for all: Overcoming Sparse and Noisy Data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, RANLP, pages 198–206.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate Speech Detection with Comment Embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, WWW, pages 29–30.

Christopher Ellis and James A Stimson. 2012. *Ideology in America*. Cambridge University Press.

Michael Gauthier, Adrien Guille, A Deseille, and Fabien Rico. 2015. Text Mining and Twitter to Analyze British Swearing Habits. *Handbook of Twitter for Research*.

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD, pages 168–177.

Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. 2007. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65.

Timothy Jay. 2009. The Utility and Ubiquity of Taboo Words. *Perspectives on Psychological Science*, 4(2):153–161.

Timothy Jay and Kristin Janschewitz. 2008. The Pragmatics of Swearing. *Journal of Politeness Research. Language, Behaviour, Culture*, 4(2):267–288.

Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. 2011. Twitter Sentiment Analysis: The Good the Bad and the OMG! In *Proceedings of the 5th International AAAI Conference on Web and Social Media*, ICWSM, pages 538–541.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(2):187–202.

Tony McEnery. 2004. *Swearing in English: Bad Language, Purity and Power from 1586 to the Present*. Routledge.

Matthias R Mehl, Simine Vazire, Nairán Ramírez-Esparza, Richard B Slatcher, and James W Pennebaker. 2007. Are Women Really more Talkative than Men? *Science*, 317(5834):82–82.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29(3):436–465.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW, pages 145–153.

Olutobi Owoputi, Brendan OConnor, Chris Dyer, Kevin Gimpel, and Nathan Schneider. 2012. Part-of-Speech Tagging for Twitter: Word Clusters and Other Advances.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, EMNLP, pages 1532–1543.

Steven Pinker. 2007. *The stuff of thought: Language as a window into human nature*. Penguin.

Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. 2017. Beyond Binary Labels: Political Ideology Prediction of Twitter Users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 729–740.

Phuc Dong Quang. 1971. English Sentences without Overt Grammatical Subject. pages 3–10.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised Modeling of Twitter Conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL, pages 172–180.

H Andrew Schwartz and et al. 2013. Personality, Gender, and Age in the Language of Social Media: The Open-vocabulary Approach. *PloS ONE*, 8(9).

Gary W Selnow. 1985. Sex differences in uses and perceptions of profanity. *Sex Roles*, 12(3-4):303–312.

Richard Stephens, John Atkins, and Andrew Kingston. 2009. Swearing as a Response to Pain. *Neuroreport*, 20(12):1056–1060.

Karolina Sylwester and Matthew Purver. 2015. Twitter Language Use Reflects Psychological Differences between Democrats and Republicans. *PLoS ONE*, 10(9).

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 384–394.

Na Wang. 2013. An Analysis of the Pragmatic Functions of 'swearing' in Interpersonal Talk. *Griffith Working Papers in Pragmatics and Intercultural Communication*, 6:71–79.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2014. Cursing in English on Twitter. In *Proceedings of the 17th ACM conference on Computer Supported Cooperative Work & Social Computing*, CSCW, pages 415–425.

William Warner and Julia Hirschberg. 2012. Detecting Hate Speech on the World Wide Web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26.

Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL student research workshop*, NAACL, pages 88–93.

Ziqi Zhang and Lei Luo. 2018. Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter. *Semantic Web*.

Is it Time to Swish?
Comparing Deep Learning Activation Functions Across NLP tasks

Steffen Eger, Paul Youssef, Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science
Technische Universität Darmstadt
`www.ukp.tu-darmstadt.de`

Abstract

Activation functions play a crucial role in neural networks because they are the nonlinearities which have been attributed to the success story of deep learning. One of the currently most popular activation functions is ReLU, but several competitors have recently been proposed or 'discovered', including LReLU functions and swish. While most works compare newly proposed activation functions on few tasks (usually from image classification) and against few competitors (usually ReLU), we perform the first large-scale comparison of 21 activation functions across eight different NLP tasks. We find that a largely unknown activation function performs most stably across all tasks, the so-called penalized tanh function. We also show that it can successfully replace the sigmoid and tanh gates in LSTM cells, leading to a 2 percentage point (pp) improvement over the standard choices on a challenging NLP task.

1 Introduction

Activation functions are a crucial component of neural networks because they turn an otherwise linear classifier into a non-linear one, which has proven key to the high performances witnessed across a wide range of tasks in recent years. While different activation functions such as sigmoid or tanh are often equivalent on a theoretical level, in the sense that they can all approximate arbitrary continuous functions (Hornik, 1991), different activation functions often show very diverse behavior in practice.

For example, sigmoid, one of the activation functions dominating in neural network practice for several decades eventually turned out less suitable for learning because (according to accepted wisdom) of its small derivative which may lead to vanishing gradients. In this respect, the so-called ReLU function (Glorot et al., 2011) has proven much more suitable. It has an identity derivative in the positive region and is thus claimed to be less susceptible to vanishing gradients. It has therefore (arguably) become the most popular activation function. The recognition of ReLU's success has led to various extensions proposed (Maas et al., 2013; He et al., 2015; Klambauer et al., 2017), but none has reached the same popularity, most likely because of ReLU's simplicity and because the gains reported tended to be inconsistent or marginal across datasets and models (Ramachandran et al., 2017).

Activation functions have been characterized by a variety of properties deemed important for successful learning, such as ones relating to their derivatives, monotonicity, and whether their range is finite or not. However, in recent work, Ramachandran et al. (2017) employed automatic search to find high-performing novel activation functions, where their search space contained compositions of elementary unary and binary functions such as $\max$, $\min$, $\sin$, $\tanh$, or $\exp$. They found many functions violating properties deemed as useful, such as non-monotonic activation functions or functions violating the gradient-preserving property of ReLU. Indeed, their most successful function, which they call swish, violates both of these conditions. However, as with previous works, they also only evaluated their newly discovered as well as their (rectifier) baseline activation functions on few different datasets, usually taken from the image classification community such as CIFAR (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015), and using few types of different networks, such as the deep convolutional networks abounding in the image classification community (Szegedy et al., 2016).

To our best knowledge, there exists no large-scale empirical comparison of different activations

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4415–4424
Brussels, Belgium, October 31 - November 4, 2018.

across a variety of tasks and network architectures, and even less so within natural language processing (NLP).[1] Thus, the question which activation function really performs best and most stably across different NLP tasks and popular NLP models remains unanswered to this date.

In this work, we fill this gap. We compare (i) 21 different activation functions, including the 6 top performers found from automatic search in Ramachandran et al. (2017), across (ii) three popular NLP task types (sentence classification, document classification, sequence tagging) comprising 8 individual tasks, (iii) using three different popular NLP architectures, namely, MLPs, CNNs, and RNNs. We also (iv) compare all functions across two different dimensions, namely: top vs. average performance.

We find that a largely unknown activation function, penalized tanh (Xu et al., 2016), performs most stably across our different tasks. We also find that it can successfully replace tanh and sigmoid activations in LSTM cells. We further find that the majority of top performing functions found in Ramachandran et al. (2017) do not perform well for our tasks. An exception is swish, which performed well across several tasks, but less stably than penalized tanh and other functions.[2]

2 Theory

Activation functions We consider 21 activation functions, 6 of which are "novel" and proposed in Ramachandran et al. (2017). The functional form of these 6 is given in Table 1, together with the sigmoid function.

The remaining 14 are: tanh, sin, relu, lrelu-0.01, lrelu-0.30, maxout-2, maxout-3, maxout-4, prelu, linear, elu, cube, penalized tanh, selu. We briefly describe them: lrelu-0.01 and lrelu-0.30 are the so-called leaky relu (LReLU) functions (Maas et al., 2013); the idea behind them is to avoid zero activations/derivatives in the negative region of relu. Their functional form is given in Table 1. prelu (He et al., 2015) generalizes the LReLU functions by allowing the slope in the negative region to be a learnable parameter. The maxout functions (Goodfellow et al., 2013) are different in that they introduce additional parameters and do not operate on a single scalar input. For example, maxout-2 is the operation that takes the maximum of two inputs: $\max\{\mathbf{xW}+\mathbf{b},\mathbf{xV}+\mathbf{c}\}$, so the number of learnable parameters is doubled. maxout-3 is the analogous function that takes the maximum of three inputs. As shown in Goodfellow et al. (2013), maxout can approximate any convex function. sin is the standard sine function, proposed in neural network learning, e.g., in Parascandolo et al. (2016), where it was shown to enable faster learning on certain tasks than more established functions. penalized tanh (Xu et al., 2016) has been defined in analogy to the LReLU functions, which can be thought of as "penalizing" the identity function in the negative region. The reported good performance of penalized tanh on CIFAR-100 (Krizhevsky, 2009) lets the authors speculate that the slope of activation functions near the origin may be crucial for learning. linear is the identity function, $f(x) = x$. cube is the function $f(x) = x^3$, proposed in Chen and Manning (2014) for an MLP used in dependency parsing. elu (Clevert et al., 2015) has been proposed as (yet another) variant of relu that assumes negative values, making the mean activations more zero-centered. selu is a scaled variant of elu used in Klambauer et al. (2017) in the context of so-called self-normalizing neural nets.

sigmoid	$f(x) = \sigma(x) = 1/(1+\exp(-x))$
swish	$f(x) = x \cdot \sigma(x)$
maxsig	$f(x) = \max\{x, \sigma(x)\}$
cosid	$f(x) = \cos(x) - x$
minsin	$f(x) = \min\{x, \sin(x)\}$
arctid	$f(x) = \arctan(x)^2 - x$
maxtanh	$f(x) = \max\{x, \tanh(x)\}$
lrelu-0.01	$f(x) = \max\{x, 0.01x\}$
lrelu-0.30	$f(x) = \max\{x, 0.3x\}$
penalized tanh	$f(x) = \begin{cases} \tanh(x) & x > 0, \\ 0.25\tanh(x) & x \leq 0 \end{cases}$

Table 1: Top: sigmoid activation function as well as 6 top performing activation functions from Ramachandran et al. (2017). Bottom: the LReLU functions with different parametrizations as well as penalized tanh.

Properties of activation functions Many properties of activation functions have been speculated to be crucial for successful learning. Some of these are listed in Table 2, together with brief de-

---

[1] An exception may be considered Xu et al. (2015), who, however, only contrast the rectifier functions on image classification datasets.

[2] Accompanying code to reproduce our experiments is available from `https://github.com/UKPLab/emnlp2018-activation-functions`.

Property	Description	Problems	Examples
derivative	f'	>1 exploding gradient (e) <1 vanishing (v)	sigmoid (v), tanh (v), cube (e)
zero-centered	range centered around zero?	if not, slower learning	tanh (+), relu (−)
saturating	finite limits	vanishing gradient in the limit	tanh, penalized tanh, sigmoid
monotonicity	$x > y \implies f(x) \geq f(y)$	unclear	exceptions: sin, swish, minsin

Table 2: Frequently cited properties of activation functions

.

scriptions and illustrations.

Graphs of all activation functions can be found in the appendix.

3 Experiments

We conduct experiments using three neural network types and three types of NLP tasks, described in §3.1, §3.2, and §3.3 below.

3.1 MLP & Sentence Classification

Model We experiment with a multi-layer perceptron (MLP) applied to sentence-level classification tasks. That is, input to the MLP is a sentence or short text, represented as a fixed-size vector embedding. The output of the MLP is a label which classifies the sentence or short text. We use two sentence representation techniques, namely, Sent2Vec (Pagliardini et al., 2018), of dimensionality 600, and InferSent (Conneau et al., 2017), of dimensionality 4096. Our MLP has the form:

$$\mathbf{x}_i = f(\mathbf{x}_{i-1} \cdot \mathbf{W}_i + \mathbf{b}_i)$$
$$\mathbf{y} = \text{softmax}(\mathbf{x}_N \mathbf{W}_{N+1} + \mathbf{b}_{N+1})$$

where $\mathbf{x}_0$ is the input representation, $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are hidden layer representations, and $\mathbf{y}$ is the output, a probability distribution over the classes in the classification task. Vectors $\mathbf{b}$ and matrices $\mathbf{W}$ are the learnable parameters of our network. The activation function is given by f and ranges over the choices described in §2.

Data We use four sentence classification tasks, namely: movie review classification (MR), subjectivitiy classification (SUBJ), question type classification (TREC), and classifying whether a sentence contains an argumentation structure of a certain type (claim, premise, major claim) or else is non-argumentative (AM). The first three datasets are standard sentence classification datasets and contained in the SentEval framework.[3] We choose the AM dataset for task diversity, and derive it by projecting token-level annotations in the dataset from Stab and Gurevych (2017) to the sentence level. In the rare case (<5% of the cases) when a sentence contains multiple argument types, we choose one based on the ordering Major Claim (MC) > Claim (C) > Premise (P). Datasets and examples are listed in Table 3.

Approach We consider 7 "mini-experiments":

- (1): MR dataset with Sent2Vec-unigram embeddings as input and 1% of the full data as training data; (2): the same mini-experiment with 50% of the full data as training data. In both cases, the dev set comprises 10% of the full data and the rest is for testing.
- (3,4): SUBJ with InferSent embeddings and likewise 1% and 50% of the full data as train data, respectively.
- (5): The TREC dataset with original split in train and test; 50% of the train split is used as dev data.
- (6): The AM dataset with original split in train, dev, and test (Eger et al., 2017), and with InferSent input embeddings. (7): the same mini-experiment with Sent2Vec-unigram embeddings.

We report accuracy for mini-experiments (1-5) and macro-F1 for (6-7). We report macro-F1 for (6-7) because the AM dataset is imbalanced.

The motivation behind choosing different input embeddings for different tasks was to investigate a wider variety of conditions. Choosing subsets of the full data had the same intention.

For all 7 mini-experiments, we draw the same 200 randomly chosen hyperparameters from the ranges indicated in Table 4. All experiments are conducted in keras.[4]

For each of the 21 different activation functions detailed in §2, we conduct each mini-experiment with the 200 randomly chosen hyperparameters.

---

[3] `https://github.com/facebookresearch/SentEval`

[4] `https://keras.io/`

Task	Type	Size	C	Example
AM	Argumentation	7k	4	Not cooking fresh food will lead to lack of nutrition. *(claim)*
MR	Sentiment	11k	2	Too slow for a younger crowd , too shallow for an older one. *(neg)*
SUBJ	Subjectivity	10k	2	A movie that doesnt aim too high , but doesnt need to. *(subj)*
TREC	Question-types	6k	6	What's the Olympic Motto? *(description)*
NG	Doc classification	18k	20	[...] You can add "dark matter" and quarks [...] (*sci.space*)
R8	Doc classification	7k	8	bowater industries profit exceed [...] (*earn*)
POS	POS tagging	204k	17	What/*PRON* to/*PART* feed/*VERB* my/*PRON* dog/*NOUN* [...]
TL-AM	Token-level AM	148k	7	[...] I/*O* firmly/*O* believe/*O* that/*O* we/*B-MC* should/*I-MC* [...]

Table 3: Evaluation tasks used in our experiments, grouped by task type (sentence classification, document classification, sequence tagging), with statistics and examples. C is the number of classes to predict.

All activation functions use the same hyperparameters and the same train, dev, and test splits.

We store two results for each mini-experiment, namely: (i) the test result corresponding to the **best** (`best`) dev performance; (ii) the **average** (`mean`) test result across all hyperparameters. The `best` result scenario mirrors standard optimization in machine learning: it indicates the score one can obtain with an activation function when the MLP is well-optimized. The `mean` result scenario is an indicator for what one can expect when hyperparameter optimization is 'shallow' (e.g., because computing times are prohibitive): it gives the average performance for randomly chosen hyperparameters. We note that we run each hyperparameter combination with 5 different random weight initializations and all the reported scores (best dev score, best `best`, best `mean`) are averages over these 5 random initializations.

Finally, we set the following hyperparameters for all MLP experiments: patience of 10 for early stopping, batch size 16, 100 epochs for training.

Results Figure 1 shows `best` and `mean` results, averaged over all 7 mini-experiments, for each activation function. To make individual scores comparable across mini-experiments, we perform max normalization and divide each score by the maximum score achieved in any given mini-experiment (for `best` and `mean`, respectively) before averaging.[5]

For `best`, the top performers are the rectifier functions (relu, lrelu-0.01, prelu) as well as maxout and penalized tanh. The newly discovered activation functions lag behind, with the best of them being minsin and swish. linear is worst, together with elu and cube. Overall, the difference between the best activation function, relu, and the worst, linear, is only roughly 2pp, however. This means that if hyperparameter search is done carefully, the choice of activation function is less important for these sentence classification tasks. Particularly the (binary) tasks MR and SUBJ appear robust against the choice of activation function, with the difference between the best and worst function being always less than 1pp, in all settings. For TREC and AM, the situation is slightly different: for TREC, the difference is 2pp (swish vs. maxsig) and for AM, it is 3pp using InferSent embeddings (swish vs. cube) and 12pp using Sent2Vec embeddings (relu vs. linear). It is noteworthy that swish wins 2 out of 3 cases in which the choice of activation function really matters.

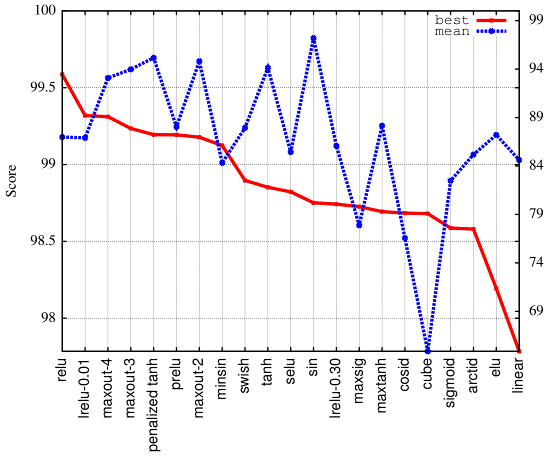

Figure 1: Sentence Classification. Left y-axis: `best`. Right y-axis: `mean`. Score on y-axes is the average over all mini-experiments.

`mean` results are very different from `best` results. Here, somewhat surprisingly, the oscillating

---

[5] We chose max normalization so that certain tasks/mini-experiments would not unduly dominate our averaged scores. Overall, our averaged scores are not (much) affected by this decision, however: the Spearman correlation of rankings of activation functions under max normalization and under no max normalization are above 0.98 in all our three classification scenarios considered in §3.1,§3.2,§3.3.

Model	Hyperparameter	Range
(a) MLP	(1) optimizer	$\{$Adam,RMSprop,Adagrad,Adadelta,Adamax,Nadam,sgd$\}$
	(2) #hidden layers N	$\{1, 2, 3, 4\}$
	(3) dropout value	$[0.1, 0.75]$
	(4) hidden units	$[30, 500]$
	(5) learning rate	$\mathcal{N}(m, m/5)$
	(6) weight initializer	$\{$random-n, random-u, varscaling, orthogonal, lecun-u, glorot-n, glorot-u, he-n, he-u$\}$
(b) CNN	(a) (1,3,5,6)	same as MLP
	embedding dimension	$[40, 200]$
	number of filters n_k	$[30, 500]$
	#hidden layers N	$\{1, 2, 3\}$
	filter size h	$\{1, 2, 2, 3, 3, 3, 4\}$
(c) RNN/LSTM	(a) (1-5)	same as MLP
	recurrent initializer	same as (a) (6) plus identity matrix

Table 4: Hyperparameter ranges for each network type. Hyperparameters are drawn using a discrete or continuous uniform distribution from the indicated ranges. Repeated values indicate multi-sets. $\mathcal{N}(\mu, s)$ is the normal distribution with mean μ and std s; $\mu = m$ is the default value from keras for the specific optimizer (if drawn learning rate is < 0, we choose it to be m).

sin function wins, followed by penalized tanh, maxout and swish. The difference between the best `mean` function, sin, and the worst, cube, is more than 30pp. This means that using cube is much riskier and requires more careful hyperparameter search compared to sin and the other top performers.

3.2 CNN & Document Classification

Model Our second paradigm is document classification using a CNN. This approach has been popularized in NLP by the ground-breaking work of Kim (2014). Even though shallow CNNs do not reach state-of-the-art results on large datasets anymore (Johnson and Zhang, 2017), simple approaches like (shallow) CNNs are still very competitive for smaller datasets (Joulin et al., 2016).

Our model operates on token-level and first embeds a sequence of tokens $x_1, \dots, x_n$, represented as 1-hot vectors, into learnable embeddings $\mathbf{x}_1, \dots, \mathbf{x}_n$. The model then applies 1D-convolution on top of these embeddings. That is, a filter $\mathbf{w}$ of size h takes h successive embeddings $\mathbf{x}_{i:i+h-1}$, performs a scalar product and obtains a feature c_i:

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b).$$

Here, f is the activation function and b is a bias term. We take the number n_k of different filters as a hyperparameter. When our network has multiple layers, we stack another convolutional layer on top of the first (in total we have n_k outputs at each time step), and so on. Our penultimate layer is a global max pooling layers that selects the maximum from each feature map. A final softmax layer terminates the network.

Data We use two document classification tasks, namely: 20 Newsgroup (NG) and Reuters-21578 R8 (R8). Both datasets are standard document classification datasets. In NG, the goal is to classify each document into one of 20 newsgroup classes (alt.atheism, sci.med, sci.space, etc.). In R8, the goal is to classify Reuters news text into one of eight classes (crude, earn, grain, interest, etc.). We used the preprocessed files from `https://www.cs.umb.edu/~smimarog/textmining/datasets/` (in particular, stopwords are removed and the text is stemmed).

Approach We consider 4 mini-experiments:

- (1,2) NG dataset with 5% and 50%, respectively of the full data as train data. In both cases, 10% of the full data is used as dev data, and the rest as test data.
- (3,4) Same as (1,2) for R8.

We report accuracy for all experiments. We use a batch size of 64, 50 epochs for training, and a patience of 10. For all mini-experiments, we again draw 200 randomly chosen hyperparameters from the ranges indicated in Table 4. The hyperparameters and train/dev/test splits are the same for all activation functions.

Results Figure 2 shows `best` and `mean` results, averaged over all mini-experiments. This time, the winners for `best` are elu, selu (again two members from the rectifier family), and maxout-3, but the difference between maxout-3 and several lower ranked functions is minimal. The cube function is again worst and sigmoid and cosid have similarly bad performance. Except for minsin, the newly proposed activation functions from Ramachandran et al. (2017) again considerably lag behind. The most stable activation functions are the maxout functions as well as penalized tanh, tanh and sin.

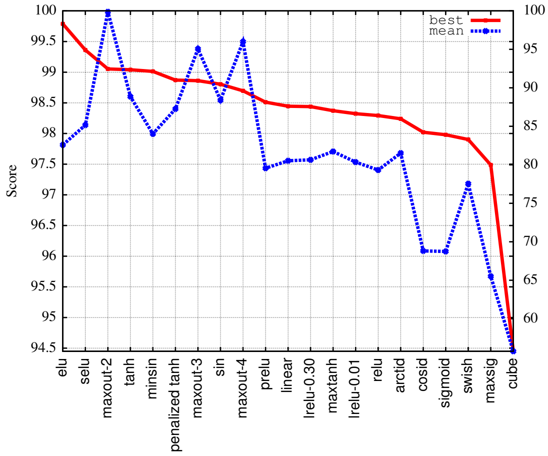

Figure 2: Doc classification.

3.3 RNN & Sequence Tagging

Model Our third paradigm is sequence tagging, a ubiquitous model type in NLP. In sequence tagging, a sequence of input tokens $w_1, \dots, w_K$ is mapped to a sequence of labels $y_1, \dots, y_K$. Classical sequence tagging tasks include POS tagging, chunking, NER, discourse parsing (Braud et al., 2017), and argumentation mining (Eger et al., 2017; Schulz et al., 2018). We use a standard recurrent net for sequence tagging, whose form is:

$$\mathbf{h}_i = f(\mathbf{h}_{i-1}\mathbf{W} + \mathbf{w}_i \cdot \mathbf{U} + \mathbf{b})$$
$$\mathbf{y}_i = \text{softmax}(\mathbf{h}_i\mathbf{V} + \mathbf{c})$$

Here, $\mathbf{w}_i$ are (pre-trained) word embeddings of words w_i. Vectors $\mathbf{b}, \mathbf{c}$ and matrices $\mathbf{U}, \mathbf{V}, \mathbf{W}$ are parameters to be learned during training. The above describes an RNN with only one hidden layer, $\mathbf{h}_i$, at each time step, but we consider the generalized form with $N \geq 1$ hidden layers; we also choose a bidirectional RNN in which the hidden outputs of a forward RNN and a backward RNN are combined. RNNs are particularly deep networks—indeed, the depth of the network corresponds to the length of the input sequence—which makes them particularly susceptible to the vanishing gradient problem (Pascanu et al., 2012).

Initially, we do not consider the more popular LSTMs here for reasons indicated below. However, we include a comparison after discussing the RNN performance.

Data We use two sequence tagging tasks, namely: English POS tagging (POS), and token-level argumentation mining (TL-AM) using the same dataset (consisting of student essays) as for the sentence level experiments. In token-level AM, we tag each token with a BIO-label plus the component type, i.e., the label space is $\mathcal{Y} = \{\mathrm{B}, \mathrm{I}\} \times \{\mathrm{MC}, \mathrm{C}, \mathrm{P}\} \cup \{\mathrm{O}\}$, where 'O" is a label for non-argumentative tokens. The motivation for using TL-AM is that, putatively, AM has more long-range dependencies than POS or similar sequence tagging tasks such as NER, because argument components are much longer than named entities and component labels also depend less on the current token.

Approach We consider 6 mini-experiments:

- (1): TL-AM with Glove-100d word embeddings and 5% of the original training data as train data; (2) the same with 30% of the original training data as train data. In both cases, dev and test follow the original train splits (Eger et al., 2017).
- (3,4) Same as (1) and (2) but with 300d Levy word embeddings (Levy and Goldberg, 2014).
- (5,6): POS with Glove-100d word embeddings and 5% and 30%, respectively, of the train data of a pre-determined train/dev/test split (13k/13k/178k tokens). Dev and test are fixed in both cases.

We report macro-F1 for mini-experiments (1-4) and accuracy for (5-6). For our RNN implementations, we use the accompanying code of (the state-of-the-art model of) Reimers and Gurevych (2017), which is implemented in keras. The network uses a CRF layer as an output layer. We use a batch size of 32, train for 50 epochs and use a patience of 5 for early stopping.

Results Figure 3 shows `best` and `mean` results, averaged over all 6 mini-experiments, for each activation function. We exclude prelu and the maxout functions because the keras implementation does not natively support these activation functions for RNNs. We also exclude the cube function because it performed very badly.

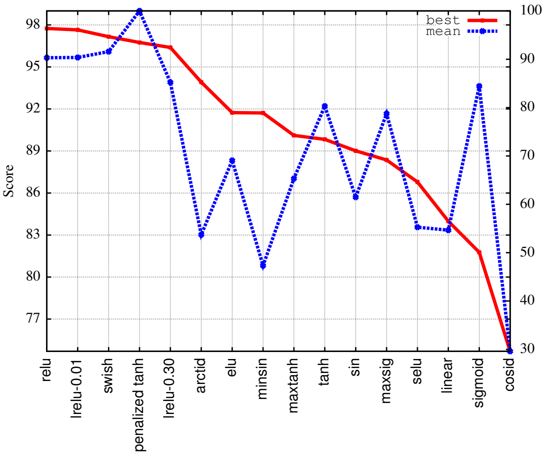

Figure 3: Sequence tagging.

Unlike for sentence classification, there are much larger differences between the activation functions. For example, there is almost 20pp difference between the best `best` activation functions: relu, lrelu-0.01, swish, penalized tanh, and the worst ones: linear, cosid, and sigmoid (the differences were larger had we included cube). Interestingly, this difference is mostly due to the TL-AM task: for POS, there is only 3pp difference between the best function (sigmoid (sic!), though with almost zero margin to the next best ones) and the worst one (linear), while this difference is almost 40pp for TL-AM. This appears to confirm our concerns regarding the POS tagging task as not being challenging enough due to lack of, e.g., long-range dependencies.

The four best `best` activation functions in Figure 3 are also the functions with the best `mean` results, i.e., they are most stable over different hyperparameters. The clear winner in this category is penalized tanh with 100% `mean` score, followed by swish with 91%. Worst is cosid with 30%. It is remarkable how large the difference between tanh and penalized tanh is both for `best` and `mean`—7pp and 20pp, respectively, which is much larger than the differences between the analogous pair of LReLU and relu. This appears to make a strong case for the importance of the slope around the origin, as suggested in Xu et al. (2016).

LSTM vs. RNN Besides an RNN, we also implemented a more popular RNN model with (bidirectional) LSTM blocks in place of standard hidden layers. Standard LSTM units follow the equations (simplified):

$$\begin{aligned}
\mathbf{f}_t &= \sigma([\mathbf{h}_{t-1};\mathbf{x}_t]\cdot\mathbf{W}_f),\\
\mathbf{i}_t &= \sigma([\mathbf{h}_{t-1};\mathbf{x}_t]\cdot\mathbf{W}_i),\\
\mathbf{o}_t &= \sigma([\mathbf{h}_{t-1};\mathbf{x}_t]\cdot\mathbf{W}_o)\\
\mathbf{c}_t &= \mathbf{f}_t\odot\mathbf{c}_{t-1}+\mathbf{i}_t\odot\tau([\mathbf{h}_{t-1};\mathbf{x}_t]\cdot\mathbf{W}_c)\\
\mathbf{h}_t &= \mathbf{o}_t\odot\tau(\mathbf{c}_t),
\end{aligned}$$

where $\mathbf{f}_t$ and $\mathbf{i}_t$ are perceived of as *gates* that control information flow, $\mathbf{x}_t$ is the input at time t and $\mathbf{h}_t$ is the hidden layer activation. In keras (and most standard references), σ is the (hard) sigmoid function, and τ is the tanh function.

We ran an LSTM on the TL-AM dataset with Levy word embeddings and 5% and 30% data size setup. We varied σ and τ independently, keeping the respective other function at its default.

We find that the top two choices for τ are penalized tanh and tanh (margin of 10pp), given that σ is sigmoid. For $\tau =$ tanh, the best choices are $\sigma =$ penalized tanh, sigmoid, and tanh. All other functions perform considerably worse. Thus, the top-performers are all saturating functions, indicating the different roles activation functions play in LSTMs—those of gates—compared to standard layers. It is worth mentioning that choosing σ or τ as penalized tanh is on average better than the standard choices for σ and τ. Indeed, choosing $\tau = \sigma =$ penalized tanh is on average 2pp better than the default choices of τ, σ.

It is further worth mentioning that the best `best` results for the LSTM are roughly 5pp better (absolute) than the best corresponding choices for the simple RNN.

4 Analysis & Discussion

Winner statistics Each of the three meta-tasks sentence classification, document classification, and sequence tagging was won, on average, by a member from the rectifier family, namely, relu (2) and elu, for `best`. Also, in each case, cube and cosid were among the worst performing activation functions. The majority of newly proposed functions from Ramachandran et al. (2017) ranked somewhere in the mid-field, with swish and minsin performing best in the `best` category. For the `mean` category, we particulary had the maxout functions as well as penalized tanh and sin regularly as top performers.

To get further insights, we computed a winner statistic across all 17 mini-experiments, counting how often each activation function was among the top 3. Table 5 shows the results, excluding prelu and the maxout functions because they were not considered in all mini-experiments.

`best`	penalized tanh (6), swish (6), elu (4), relu (4), lrelu-0.01 (4)
`mean`	penalized tanh (16), tanh (13) sin (10)

Table 5: Top-3 winner statistics. In brackets: number of times within top-3, keeping only functions with four or more top-3 rankings.

We see that penalized tanh and swish win here for `best`, followed by further rectifier functions. The `mean` category is clearly won by saturating activation functions with finite range. If this comparison were restricted to sentence and document classification, where we also included the maxout functions, then penalized tanh would have been outperformed by maxout for `mean`.

This appears to yield the conclusion that functions with limited range behave more stably across hyperparameter settings while non-saturating functions tend to yield better top-performances. The noteworthy exception to this rule is penalized tanh which excels in both categories (the more expensive maxout functions would be further exceptions). If the slope around the origin of penalized tanh is responsible for its good performance, then this could also explain why cube is so bad, since it is very flat close to the origin.

Influence of hyperparameters To get some intuition about how hyperparameters affect our different activation functions, we regressed the score of the functions on the test set on all the employed hyperparameters. For example, we estimated:

$$y = \alpha_l \cdot \log(n_l) + \alpha_d \cdot d + \cdots \qquad (1)$$

where y is the score on the test set, n_l is the number of layers in the network, d is the dropout value, etc. The coefficients α_k for each regressor k is what we want to estimate (in particular, their size and their sign). We logarithmized certain variables whose scale was substantially larger than those of others (e.g., number of units, number of filters). For discrete regressors such as the optimizer we used binary dummy variables. We estimated Eq. (1) independently for each activation function and for each mini-experiment. Overall, there was a very diverse pattern of outcomes, preventing us from making too strong conclusions. Still, we observed that while all models performed on average better with fewer hidden layers, particularly swish was robust to more hidden layers (small negative coefficient α_l), but also, to a lesser degree, penalized tanh. In the sentence classification tasks, sin and the maxout functions were particulary robust to an increase of hidden layers. Since penalized tanh is a saturating function and sin even an oscillating one, we therefore conclude that preserving the gradient (derivative close to one) is not a necessary prerequisite to successful learning in deeper neural networks.

5 Concluding remarks

We have conducted the first large scale comparison of activation functions across several different NLP tasks (and task types) and using different popular neural network types. Our main focus was on so-called scalar activation functions, but we also partly included the more costly 'many-to-one' maxout functions.

Our findings suggest that the rectifier functions (and the similarly shaped swish) can be top performers for each task, but their performance is unstable and cannot be predicted a priori. One of our major findings is that, in contrast, the saturating penalized tanh function performs much more stably in this respect and can with high probability be expected to perform well across tasks as well as different choices of hyperparameters. This appears to make it the method of choice particularly when hyperparameter optimization is costly. When hyperparameter optimization is cheap, we recommend to consider the activation function as another hyperparameter and choose it, e.g., from the range of functions listed in Table 5 along with maxout.

Another major advantage of the penalized tanh function is that it may also take the role of a gate (because of its finite range) and thus be employed in more sophisticated neural network units such as LSTMs, where the rectifiers fail completely. In this context, we noticed that replacing sigmoid and tanh in an LSTM cell with penalized tanh leads to a 2pp increase on a challenging NLP sequence tagging task. Exploring whether this holds across more NLP tasks should be scope for future work. Additionally, our research suggests it is worthwhile to further explore penalized tanh, an arguably marginally known activation function. For instance, other scaling factors than 0.25 (default value from Xu et al. (2016)) should be explored. Similarly as for prelu, the scaling factor can also be made part of the optimization problem.

Finally, we found that except for swish none of the newly discovered activation functions found in Ramachandran et al. (2017) made it in our top categories, suggesting that automatic search of activation functions should be made across multiple tasks in the future.

Acknowledgments

We thank Teresa Bötschen, Nils Reimers and the anonymous reviewers for helpful comments. This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 01UG1816B (CEDIFOR).

References

Chloé Braud, Ophélie Lacroix, and Anders Søgaard. 2017. Cross-lingual and cross-domain discourse segmentation of entire documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 237–243.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 681–691. Association for Computational Linguistics.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22, Vancouver, Canada. Association for Computational Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1319–III–1327. JMLR.org.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1026–1034, Washington, DC, USA. IEEE Computer Society.

Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2):251–257.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 562–570.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 971–980. Curran Associates, Inc.

A. Krizhevsky. 2009. Learning multiple layers of features from tiny images. Master's thesis, Computer Science Department, University of Toronto.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308. Association for Computational Linguistics.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning (ICML)*.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. 2016. Taming the waves: sine as activation function in deep neural networks.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. 2017. Searching for activation functions. *CoRR*, abs/1710.05941.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multitask learning for argumentation mining in low-resource settings. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page to appear. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261.

Bing Xu, Ruitong Huang, and Mu Li. 2016. Revise saturated activation functions. *CoRR*, abs/1602.05980.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853.

Hard Non-Monotonic Attention for Character-Level Transduction

Shijie Wu[Э] **Pamela Shapiro**[Э] **Ryan Cotterell**[Э,fi]
[Э]Department of Computer Science, Johns Hopkins University, Baltimore, USA
[fi]The Computer Laboratory, University of Cambridge, Cambridge, UK
`{shijie.wu, pshapiro, ryan.cotterell}@jhu.edu`

Abstract

Character-level string-to-string transduction is an important component of various NLP tasks. The goal is to map an input string to an output string, where the strings may be of different lengths and have characters taken from different alphabets. Recent approaches have used sequence-to-sequence models with an attention mechanism to learn which parts of the input string the model should focus on during the generation of the output string. Both soft attention and hard monotonic attention have been used, but hard non-monotonic attention has only been used in other sequence modeling tasks such as image captioning (Xu et al., 2015) and has required a stochastic approximation to compute the gradient. In this work, we introduce an exact, polynomial-time algorithm for marginalizing over the exponential number of non-monotonic alignments between two strings, showing that hard attention models can be viewed as neural reparameterizations of the classical IBM Model 1. We compare soft and hard non-monotonic attention experimentally and find that the exact algorithm significantly improves performance over the stochastic approximation and outperforms soft attention.

1 Introduction

Many natural language tasks are expressible as string-to-string transductions operating at the character level. Probability models with recurrent neural parameterizations currently hold the state of the art on many such tasks. On those string-to-string transduction tasks that involve a mapping between two strings of different lengths, it is often necessary to resolve which input symbols are related to which output symbols. As an example, consider the task of transliterating a Russian word into the Latin alphabet. In many cases, there exists a one-to-two mapping between Cyrillic and Latin letters: in Хурщёв (*Khrushchev*), the Russian Х can be considered to generate the Latin letters *Kh*. Supervision is rarely, if ever, provided at the level of character-to-character alignments—this is the problem that attention seeks to solve in neural models.

With the rise of recurrent neural networks, this problem has been handled with "soft" **attention** rather than traditional hard **alignment**. Attention (Bahdanau et al., 2015) is often described as "soft," as it does not clearly associate a single input symbol with each output symbol, but rather offers a fuzzy notion of what input symbols may be responsible for which symbols in the output. In contrast, an alignment directly associates a given input symbol with a given output symbol. To express uncertainty, practitioners often place a distribution over the exponential number of hard non-monotonic alignments, just as a probabilistic parser places a distribution over an exponential number of trees. The goal, then, is to learn the parameters of this distribution over all non-monotonic alignments through backpropagation. Incorporating hard alignment into probabilistic transduction models dates back much farther in the NLP literature; arguably, originating with the seminal paper by Brown et al. (1993). Some neural approaches have moved back towards this approach of a more rigid alignment, referring to it as "hard attention." We will refer to this as "hard attention" and to more classical approaches as "alignment."

This paper offers two insights into the usage of hard alignment. First, we derive a dynamic program for the exact computation of the likelihood in a neural model with latent hard alignment: Previous work has used a stochastic algorithm to approximately sum over the exponential number of alignments between strings. In so doing, we go on to relate neural hard alignment models to the classical IBM Model 1 for alignment in machine translation. Second, we provide an experimental comparison that indicates hard attention models outperform soft attention models on three character-level string-to-string transduction tasks: grapheme-to-phoneme conversion, named-entity transliteration and morphological inflection.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4425–4438
Brussels, Belgium, October 31 - November 4, 2018.

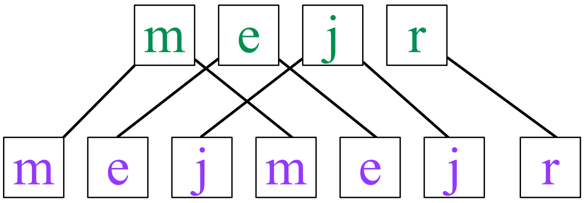

Figure 1: Example of a non-monotonic character-level transduction from the Micronesian language of Pingelapese. The infinitive *mejr* is mapped through a reduplicative process to its gerund *mejmejr* (Rehg and Sohl, 1981). Each input character is drawn in green and each output character is drawn in purple, connected with a line to the corresponding input character.

2 Non-Monotonic Transduction

This paper presents a novel, neural, probabilistic latent-variable model for **non-monotonic** transduction. As a concrete example of a non-monotonic transduction, consider the mapping of a Pingelapese infinitive to its gerund, as shown in Fig. 1. The mapping requires us to generate the output string left-to-right, bouncing around the input string out-of-order to determine the characters to transduce from. As the non-monotonic alignment is the latent variable, we will face a combinatorial problem: summing over all non-monotonic alignments. The algorithmic contribution of this paper is the derivation of a simple dynamic program for computing this sum in polynomial time that still allows for very rich recurrent neural featurization of the model. With respect to the literature, our paper represents the first instance of exact marginalization for a neural transducer with hard non-monotonic alignment; previous methods, such as Rastogi et al. (2016) and Aharoni and Goldberg (2017), are exclusively monotonic.

Non-monotonic methods dominate character-level transduction. Indeed, the state of art in classic character-level NLP tasks such as grapheme-to-phoneme conversion (Yao and Zweig, 2015), transliteration (Rosca and Breuel, 2016) and morphological inflection generation (Kann and Schütze, 2016) is held by the soft non-monotonic method of Bahdanau et al. (2015). Even though non-monotonicity is more common in word-level tasks, it also exists in character-level transduction tasks, as evidenced by our example in Fig. 1 and the superior performance of non-monotonic methods. Our error analysis in §8.4 sheds some light on why non-monotonic methods are the state of the art in a seemingly monotonic task.

A Note on the Character-level Focus. A natural question at this point is why we are not experimenting with word-level transduction tasks, such as machine translation. As we show in the §3.2 our method is often an order of magnitude slower, since it will involve a mixture of softmaxes. Thus, the exact marginalization scheme is practically unworkable for machine translation; we discuss future extensions for machine translation in §6. However, the slow-down is no problem for character-level tasks and we show empirical gains in §8.

3 Hard Non-Monotonic Alignment

3.1 The Latent-Variable Model

An alphabet is a finite, non-empty set. Given two alphabets $\Sigma_{\mathrm{x}} = \{x_1, \ldots, x_{|\Sigma_{\mathrm{x}}|}\}$ and $\Sigma_{\mathrm{y}} = \{y_1, \ldots, y_{|\Sigma_{\mathrm{y}}|}\}$, probabilistic approaches to the problem attempt to estimate a probability distribution $p(\mathbf{y} \mid \mathbf{x})$ where $\mathbf{y} \in \Sigma_{\mathrm{y}}^*$ and $\mathbf{x} \in \Sigma_{\mathrm{x}}^*$. Foreshadowing, we will define the parameters of p to be, in part, the parameters of a recurrent neural network, in line with the state-of-the-art models. We define the set $A = \{1, \ldots, |\mathbf{x}|\}^{|\mathbf{y}|}$, which has an interpretation as the set of all (potentially non-monotonic) alignments from $\mathbf{x}$ to $\mathbf{y}$ with the restriction that each output symbol y_i aligns to exactly one symbol in $\mathbf{x} \in \Sigma_{\mathrm{x}}^*$. In other words, A is the set of all many-to-one alignments between $\mathbf{x}$ and $\mathbf{y}$ where many may be as few as zero. We remark that $|A| = |\mathbf{x}|^{|\mathbf{y}|}$, which is exponentially large in the length of the target string $\mathbf{y}$. For an $\mathbf{a} \in A$, $a_i = j$ refers to the event that y_i, the i^{th} component of $\mathbf{y}$, is aligned to x_j, the j^{th} component of $\mathbf{x}$.

We define a probability distribution over output strings $\mathbf{y}$ conditioned on an input string $\mathbf{x}$ where we marginalize out *unobserved* alignments $\mathbf{a}$:

$$p(\mathbf{y} \mid \mathbf{x}) = \sum_{\mathbf{a} \in A(\mathbf{x},\mathbf{y})} p(\mathbf{y}, \mathbf{a} \mid \mathbf{x}) \tag{1}$$

$$= \underbrace{\sum_{\mathbf{a} \in A} \prod_{i=1}^{|\mathbf{y}|} p(y_i \mid a_i, \mathbf{y}_{<i}, \mathbf{x})\, p(a_i \mid \mathbf{y}_{<i}, \mathbf{x})}_{\textit{exponential number of terms}} \tag{2}$$

$$= \underbrace{\prod_{i=1}^{|\mathbf{y}|} \sum_{a_i=1}^{|\mathbf{x}|} p(y_i \mid a_i, \mathbf{y}_{<i}, \mathbf{x})\, p(a_i \mid \mathbf{y}_{<i}, \mathbf{x})}_{\textit{polynomial number of terms}} \tag{3}$$

$$= \prod_{i=1}^{|\mathbf{y}|} \sum_{j=1}^{|\mathbf{x}|} \mathbb{1}_{a_{i,j}}\, \alpha_j(i)\, p(y_i \mid a_i, \mathbf{y}_{<i}, \mathbf{x}) \tag{4}$$

where we define $\alpha_j(i) = p(a_i \mid \mathbf{y}_{<i}, \mathbf{x})$ in order to better notationally compare our model to that

of Bahdanau et al. (2015) in §5. Each distribution $p(y_i \mid a_i, \mathbf{y}_{<i}, \mathbf{x})$ in the definition of the model has a clean interpretation as a distribution over the output vocabulary Σ_y, given an input string $\mathbf{x} \in \Sigma_\mathrm{x}^*$, where y_i is aligned to x_j. Thus, one way of thinking about this hard alignment model is as a product of mixture models, one mixture at each step, with mixing coefficients $\alpha_j(i)$.

Why Does Dynamic Programming Work? Our dynamic program to compute the likelihood, fully specified in eq. (3), is quite simple: The non-monotonic alignments are independent of each other, i.e., $\alpha_j(i)$ is independent of $\alpha_j(i-1)$, conditioned on the observed sequence $\mathbf{y}$. This means that we can cleverly rearrange the terms in eq. (2) using the distributive property. Were this not the case, we could not do better than having an exponential number of summands. This is immediately clear when we view our model as a graphical model, as in Fig. 2: There is no active trail from a_i to a_k where $k > i$, ignoring the dashed lines. Note that this is no different than the tricks used to achieve exact inference in n^{th}-order Markov models—one makes an independence assumption between the current bit of structure and the previous bits of structure to allow an efficient algorithm. For a proof of eq. (2)–eq. (3), one may look in Brown et al. (1993). Foreshadowing, we note that certain parameterizations make use of input feeding, which breaks this independence; see §5.1.

Relation to IBM Model 1. The derivation above is similar to that of the IBM alignment model 1. We remark, however, two key generalizations that will serve our recurrent neural parameterization well in §4. First, traditionally, derivations of IBM Model 1 omit a prior over alignments $p(a_i \mid \mathbf{x})$, taking it to be uniform. Due to this omission, an additional multiplicative constant $\varepsilon/|\mathbf{x}|^{|\mathbf{y}|}$ is introduced to ensure the distribution remains normalized (Koehn, 2009). Second, IBM Model 1 does not condition on previously generated words on the output side. In other words, in their original model, Brown et al. (1993) assume that $p(y_i \mid a_i, \mathbf{y}_{<i}, \mathbf{x}) = p(y_i \mid a_i, \mathbf{x})$, forsaking dependence on $\mathbf{y}_{<i}$. We note that there is no reason why we need to make this independence assumption—we will likely want a target-side language model in transduction. Indeed, subsequent statistical machine translation systems, e.g., MOSES (Koehn et al., 2007), integrate a language model into the decoder. It is of note that many models in NLP have made similar independence assumptions, e.g., the emission distribution hidden Markov models (HMMs) are typically taken to be independent of all previous emissions (Rabiner, 1989). These assumptions are generally not necessary.

3.2 Algorithmic Analysis: Time Complexity

Let us assume that the requisite probability distributions are computable in $\mathcal{O}(1)$ time and the softmax takes $\mathcal{O}(\Sigma_\mathrm{y})$. Then, by inspection, the computation of the distribution in eq. (4) is $\mathcal{O}\left(|\mathbf{x}| \cdot |\mathbf{y}| \cdot |\Sigma_\mathrm{y}|\right)$, as the sum in eq. (3) contains this many terms thanks to the dynamic program that allowed us to rearrange the sum and the product. While this "trick" is well known in the NLP literature—it dates from the seminal work in statistical machine translation by Brown et al. (1993)—it has been forgotten in recent formulations of hard alignment (Xu et al., 2015), which use stochastic approximation to handle the exponential summands. As we will see in §5, we can compute the soft-attention model of Bahdanau et al. (2015) in $\mathcal{O}\left(|\mathbf{x}| \cdot |\mathbf{y}| + |\mathbf{y}| \cdot |\Sigma_\mathrm{y}|\right)$ time. When Σ_y is large, for example in case of machine translation with tens of thousands of Σ_y at least, we can ignore $|\mathbf{x}| \cdot |\mathbf{y}|$ in soft-attention model, and the exact marginalization has an extra $|\mathbf{x}|$-factor compared to soft-attention model. In practice, Shi and Knight (2017) show the bottleneck of a NMT system is the softmax layer, making the extra $|\mathbf{x}|$-factor practically cumbersome.

4 Recurrent Neural Parameterization

How do we parameterize $p(y_i \mid a_i, \mathbf{y}_{<i}, \mathbf{x})$ and $\alpha_j(i)$ in our hard, non-monotonic transduction model? We will use a neural network identical to the one proposed in the attention-based sequence-to-sequence model of Luong et al. (2015) *without input feeding* (a variant of Bahdanau et al. (2015)).

4.1 Encoding the Input

All models discussed in this exposition will make use of the same mechanism for mapping a source string $\mathbf{x} \in \Sigma_\mathrm{x}^*$ into a fixed-length representation in $\mathbb{R}^{d_h}$. This mapping will take the form of a bidirectional recurrent neural network encoder, which works as follows: each element of Σ_x is mapped to an embedding vector of length d_e through a mapping: $\mathbf{e} : \Sigma_\mathrm{x} \to \mathbb{R}^{d_e}$. Now, the RNN folds the

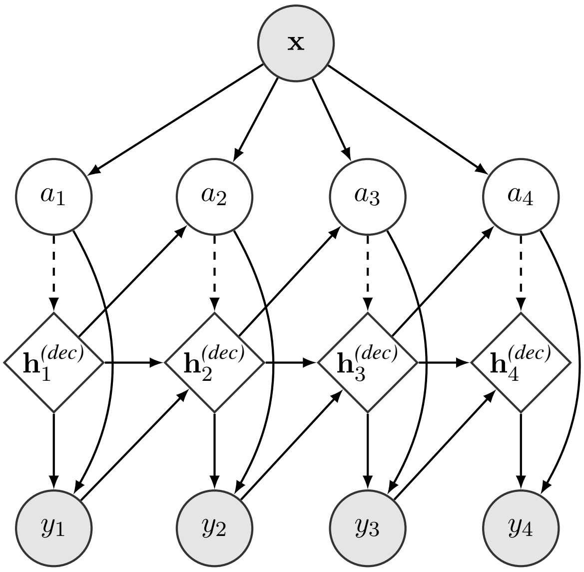

Figure 2: Our hard-attention model *without input feeding* viewed as a graphical model. Note that the circular nodes are random variables and the diamond nodes deterministic variables ($\mathbf{h}^{(dec)}_i$ is first discussed in §4.3). The independence assumption between the alignments a_i when the y_i are *observed* becomes clear. Note that we have omitted arcs from $\mathbf{x}$ to y_1, y_2, y_3 and y_4 for clarity (to avoid crossing arcs). We alert the reader that the dashed edges show the additional dependencies added in the *input feeding version*, as discussed in §5.1. Once we add these in, the a_i are no longer independent and break exact marginalization. Note the hard-attention model does not enforce an exactly one-to-one constraint. Each source-side word is free to align to many of the target-side words, independent of context. In the latent variable model, the x variable is a vector of source words, and the alignment may be over more than one element of x.

following recursion over the string $\mathbf{x}$ left-to-right:

$$\overrightarrow{\mathbf{h}}^{(enc)}_j = \tanh\Big(\overrightarrow{\mathbf{U}}^{(enc)}\,\mathbf{e}^{(enc)}(x_j) + \overrightarrow{\mathbf{V}}^{(enc)}\,\overrightarrow{\mathbf{h}}^{(enc)}_{j-1} + \overrightarrow{\mathbf{b}}^{(enc)}\Big) \tag{5}$$

where we fix the 0^{th} hidden state $\mathbf{h}^{(enc)}_0$ to the zero vector and the matrices $\overrightarrow{\mathbf{U}}^{(enc)} \in \mathbb{R}^{d_h \times d_e}$, $\overrightarrow{\mathbf{V}}^{(enc)} \in \mathbb{R}^{d_h \times d_h}$ and the bias term $\overrightarrow{\mathbf{b}}^{(enc)} \in \mathbb{R}^{d_h}$ are parameters to be learned. Performing the same procedure on the reversed string and using an RNN with different parameters, we arrive at hidden state vectors $\overleftarrow{\mathbf{h}}^{(enc)}_j$. The final hidden states from the encoder are the concatenation of the two, i.e., $\mathbf{h}^{(enc)}_j = \overrightarrow{\mathbf{h}}^{(enc)}_j \oplus \overleftarrow{\mathbf{h}}^{(enc)}_j$, where $\oplus$ is vector concatenation.

As has become standard, we will use an extension to this recursion: we apply the long short-term memory (LSTM; (Hochreiter and Schmidhuber, 1997)) recursions, rather than those of a vanilla RNN (Elman network; Elman (1990)).

4.2 Parameterization.

Now we define the alignment distribution $\alpha_j(i)$.

$$\alpha_j(i) = \frac{\exp(e_{ij})}{\sum_{j'=1}^{|\mathbf{x}|} \exp(e_{ij'})} \tag{6}$$

$$e_{ij} = \mathbf{h}^{(dec)\top}_i\, \mathbf{T}\, \mathbf{h}^{(enc)}_j \tag{7}$$

where $\mathbf{T} \in \mathbb{R}^{d_h \times 2d_h}$ and $\mathbf{h}^{(dec)}_i$, the decoder RNN's hidden state, is defined in §4.3. Importantly, the alignment distribution $\alpha_j(i)$ at time step i will *only* depend on the prefix of the output string $\mathbf{y}_{<i}$ generated so far. This is clear since the output-side decoder is a unidirectional RNN.

We also define $p(y_i \mid a_i, \mathbf{y}_{<i}, \mathbf{x})$.

$$p(y_i \mid a_i, \mathbf{y}_{<i}, \mathbf{x}) = \text{softmax}\Big(\mathbf{W}\,\mathbf{f}(\mathbf{h}^{(dec)}_i, \mathbf{h}^{(enc)}_{a_i})\Big) \tag{8}$$

The function $\mathbf{f}$ is a non-linear and vector-valued; one popular choice of $\mathbf{f}$ is a multilayer perceptron with parameters to be learned. We define:

$$\mathbf{f}(\mathbf{h}^{(dec)}_i, \mathbf{h}^{(enc)}_{a_i}) = \tanh\Big(\mathbf{S}\,(\mathbf{h}^{(dec)}_i \oplus \mathbf{h}^{(enc)}_{a_i})\Big) \tag{9}$$

where $\mathbf{S} \in \mathbb{R}^{d_s \times 3d_h}$.

4.3 Updating the hidden state $\mathbf{h}^{(dec)}_i$

The hidden state $\mathbf{h}^{(dec)}_i$ is also updated through the LSTM recurrences (Hochreiter and Schmidhuber, 1997). The RNN version of the recurrence mirrors that of the encoder,

$$\mathbf{h}^{(dec)}_i = \tanh\Big(\mathbf{U}^{(dec)}\,\mathbf{e}^{(dec)}(y_{i-1}) + \mathbf{V}^{(dec)}\,\mathbf{h}^{(dec)}_{i-1} + \mathbf{b}^{(dec)}\Big) \tag{10}$$

where $\mathbf{e}^{(dec)} : \Sigma_{\mathrm{y}} \to \mathbb{R}^{d_e}$ produces an embedding of each of the symbols in the output alphabet. What is crucial about this RNN, like the $\alpha_j(i)$, is that it only summarizes the characters decoded so far *independently of the previous attention weights*. In other words, the attention weights at time step i will have no influence from the attention weights at previous time steps, shown in Fig. 2. This is what allows for dynamic programming.

5 Transduction with Soft Attention

In order to contrast it with the hard alignment mechanism we develop, we here introduce Luong attention (Luong et al., 2015) for recurrent neural sequence to sequence models (Sutskever et al., 2014). Note that this model will also serve as an experimental baseline in §8.

The soft-attention transduction model defines a distribution over the output Σ_{y}^*, much like the hard-attention model, with the following expression:

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{|\mathbf{y}|} p(y_i \mid \mathbf{y}_{<i}, \mathbf{x}) \tag{11}$$

where we define each conditional distribution as

$$\begin{aligned} & p(y_i \mid \mathbf{y}_{<i}, \mathbf{x}) && (12) \\ & \quad = \text{softmax}\left(\mathbf{W}\, \mathbf{f}(\mathbf{h}_i^{(dec)}, \mathbf{c}_i)\right) \end{aligned}$$

We reuse the function $\mathbf{f}$ in eq. (9). The hidden state $\mathbf{h}_i^{(dec)}$, as before, is the i^{th} state of a target-side language model that summarizes the prefix of the string decoded so far; this is explained in §4.3. And, finally, we define the **context vector**

$$\mathbf{c}_i = \sum_{j=1}^{|\mathbf{x}|} \alpha_j(i)\, \mathbf{h}_j^{(enc)} \tag{13}$$

using the same alignment distribution as in §4.2. In the context of the soft-attention model, this distribution is referred to as the **attention weights**.

Inspection shows that there is only a small difference between the soft-attention model presented here and and our hard non-monotonic attention model. The difference is where we place the probabilities $\alpha_j(i)$. In the soft-attention version, we place them inside the softmax (and the function $\mathbf{f}$), as in eq. (12), and we have a mixture of the encoder's hidden states, the context vector, that we feed into the model. On the other hand, if we place them outside the softmax, we have a mixture of softmaxes, as shown in eq. (3). Both models have *identical* set of parameters.

5.1 Input Feeding: What's That?

The equations in eq. (10), however, are not the only approach. **Input-feeding** is another popular approach that is, perhaps, standard at this point (Luong et al., 2015). Input feeding refers to the setting where the architecture designer additionally feeds the attention weights into the update for the decoder's hidden state. This yields the recursion

$$\begin{aligned} \mathbf{h}_i^{(dec)} = \tanh\Big(&\mathbf{U}^{(dec)}\,(\mathbf{e}^{(dec)}(y_{i-1}) \oplus \bar{\mathbf{c}}_{i-1}) + \\ &\mathbf{V}^{(dec)}\,\mathbf{h}_{i-1}^{(dec)} + \mathbf{b}^{(dec)}\Big) \end{aligned} \tag{14}$$

where $\bar{\mathbf{c}}_{i-1} = \mathbf{f}(\mathbf{h}_{i-1}^{(dec)}, \mathbf{c}_{i-1})$. Note that this requires that $\mathbf{U}^{(dec)} \in \mathbb{R}^{d_h \times (d_e + d_s)}$. This is the architecture discussed in Bahdanau et al. (2015, §3.1). In contrast to the architecture above, this architecture has attention weights that do depend on previous attention weights due to the feeding in of the context vector $\mathbf{c}_i$. See Cohn et al. (2016) for an attempt to incorporate structural biases into the manner in which the attention distribution is influenced by previous attention distributions.

5.2 Combining Hard Non-Monotonic Attention with Input Feeding

To combine hard attention with input feeding, Xu et al. (2015) derive a variational lower bound on the log-likelihood though Jensen's inequality:

$$\begin{aligned} \log p(\mathbf{y} \mid \mathbf{x}) &= \log \sum_{\mathbf{a} \in A(\mathbf{x},\mathbf{y})} p(\mathbf{y}, \mathbf{a} \mid \mathbf{x}) \\ &= \log \sum_{\mathbf{a} \in A(\mathbf{x},\mathbf{y})} p(\mathbf{a} \mid \mathbf{x}) \cdot p(\mathbf{y} \mid \mathbf{x},) \\ &\geq \sum_{\mathbf{a} \in A(\mathbf{x},\mathbf{y})} p(\mathbf{a} \mid \mathbf{x}) \log p(\mathbf{y} \mid \mathbf{x}, \mathbf{a}) \end{aligned}$$

Note that we have omitted the dependence of $p(\mathbf{a} \mid \mathbf{x})$ on the appropriate prefix of $\mathbf{y}$; this was done for notational simplicity. Using this bound, Xu et al. (2015) derive an efficient approximation to the gradient using the REINFORCE trick of Williams (1992). This sampling-based gradient estimator is then used for learning, but suffers from high variance. We compare to this model in §8.

6 Future Work

Just as Brown et al. (1993) started with IBM Model 1 and build up to richer models, we can do the same. Extensions, resembling those of IBM Model 2 and the HMM aligner (Vogel et al., 1996)

	source	**target**
Grapheme-to-phoneme conversion	`a c t i o n`	`AE K SH AH N`
Named-entity transliteration	`A A C H E N`	아헨
Morphological inflection	`N AT+ALL SG l i p u k e`	`l i p u k k e e l l e`

Table 1: Example of source and target string for each task as processed by the model

that generalize IBM Model 1, are easily bolted on to our proposed model as well. If we are willing to perform approximate inference, we may also consider fertility as found in IBM Model 4.

In order to extend our method to machine translation (MT) in any practical manner, we require an approximation to the softmax. Given that the softmax is already the bottleneck of neural MT models (Shi and Knight, 2017), we can not afford ourselves a $\mathcal{O}(|\mathbf{x}|)$ slowdown during training. Many methods have been proposed for approximating the softmax (Goodman, 2001; Bengio et al., 2003; Gutmann and Hyvärinen, 2010). More recently, Chen et al. (2016) compared methods on neural language modeling and Grave et al. (2017) proposed a GPU-friendly method.

7 The Tasks

The empirical portion of the paper focuses on character-level string-to-string transduction problems. We consider three tasks: G: grapheme-to-phoneme conversion, T: named-entity transliteration, and I: morphological inflection. We describe each briefly in turn and we give an example of a source and target string for each task in Tab. 1.

Grapheme-to-Phoneme Conversion. We use the standard grapheme-to-phoneme conversion (G2P) dataset: the Sphinx-compatible version of CMUDict (Weide, 1998) and NetTalk (Sejnowski and Rosenberg, 1987). G2P transduces a word, a string of graphemes, to its pronunciation, a string of phonemes. We evaluate with word error rate (WER) and phoneme error rate (PER) (Yao and Zweig, 2015). PER is equal to the edit distance divided by the length of the string of phonemes.

input-fed		**soft attention**	**hard alignment**
	yes	① Bahdanau et al. (2015); Luong et al. (2015)	② Xu et al. (2015)
	no	③ Luong et al. (2015) *without input feeding*	④ *This work*

Table 2: The 4 architectures considered in the paper.

Named-Entity Transliteration. We use the NEWS 2015 shared task on machine transliteration (Zhang et al., 2015) as our named-entity transliteration dataset. It contains 14 language pairs. Transliteration transduces a named entity from its source language to a target language—in other words, from a string in the source orthography to a string in the target orthography. We evaluate with word accuracy in percentage (ACC) and mean F-score (MFS) (Zhang et al., 2015). For completeness, we include the definition of MFS in App. A.

Morphological Inflection. We consider the high-resource setting of task 1 in the CoNLL–SIGMORPHON 2017 shared task (Cotterell et al., 2017) as our morphological inflection dataset. It contains 51 languages in the high resource setting. Morphological inflection transduces a lemma (a string of characters) and a morphological tag (a sequence of subtags) to an inflected form of the word (a string of characters). We evaluate with word accuracy (ACC) and average edit distance (MLD) (Cotterell et al., 2017).

8 Experiments

The goal of the empirical portion of our paper is to perform a controlled study of the different architectures and approximations discussed up to this point in the paper. §8.1 exhibits the neural architectures we compare and the main experimental results[1] are in Tab. 3. In §8.2, we present the experimental minutiae, e.g. hyperparameters. In §8.3, we analyze our experimental findings. Finally, in §8.4, we perform error analysis and visualize the soft attention weight and hard alignment distribution.

8.1 The Architectures

The four architectures we consider in controlled comparison are: ①: soft attention *with* input feeding, ②: hard attention *with* input feeding, ③: soft attention *without* input feeding and ④: **hard attention *without* input feeding** (our system). They are also shown in Tab. 2. As a fifth system, we compare to the monotonic system Ⓜ: Aharoni and Goldberg (2017). Additionally, we present Ⓤ, a variant of ① where the number of parameters is not

---

[1] Because we do not have access to the test set of T, we only report development performance.

	Grapheme-to-Phoneme Conversion (G)				Named-Entity Transliteration (T)				Morphological Inflection (I)			
	Small		**Large**		**Small**		**Large**		**Small**		**Large**	
	WER	**PER**	**WER**	**PER**	**ACC**	**MFS**	**ACC**	**MFS**	**ACC**	**MLD**	**ACC**	**MLD**
①	33.7	0.080	30.8	0.074	38.9	0.890	39.9	0.893	91.4	0.183	91.1	0.201
Ⓤ	30.6	0.074	30.4	0.073	39.8	0.891	40.3	0.894	91.0	0.185	91.0	0.212
②	32.3	0.079	33.1	0.081	36.3	0.881	30.8	0.837	91.0	0.193	89.3	0.322
③	30.3	0.074	28.6	0.070	**40.1**	0.891	40.5	0.894	92.0	0.163	92.2	0.166
④	**29.6**	**0.072**	**28.2**	**0.068**	39.8	0.891	**41.1**	0.894	**92.6**	**0.151**	**93.6**	**0.128**
Ⓡ	30.7	0.076	29.7	0.074	37.1	0.882	36.9	0.863	91.2	0.190	92.8	0.151
Ⓜ	33.9	0.082	29.9	0.072	38.8	**0.959**	40.1	**0.960**	91.7	0.160	92.8	0.141

Table 3: Average test performance on G, T and I averaged across datasets and languages. See App. B fCor full breakdown.

	Small	Large	Search range
Emb. dim.	100	200	{50,100,200,300}
Enc. dim.	200	400	{100,200,400,600}
Enc. layer	1	2	{1,2,3}
Dec. dim.	200	400	{100,200,400,600}
Dec. layer	1	1	{1,2,3}
Dropout	0.2	0.4	{0,0.1,0.2,0.4}
# param.	1.199M	8.621M	N/A

Table 4: Model hyperparameters and search range

controlled for, and Ⓡ, a variant of ④ trained using REINFORCE instead of exact marginalization.

8.2 Experimental Details

We implement the experiments with PyTorch (Paszke et al., 2017) and we port the code of Aharoni and Goldberg (2017) to admit batched training. Because we did not observe any improvements in preliminary experiments when decoding with beam search [2], all models are decoded greedily.

Data Preparation. For G, we sample 5% and 10% of the data as development set and test set, respectively. For T, we only run experiments with 11 out of 14 language pairs[3] because we do not have access to all the data.

Model Hyperparameters. The hyperparameters of all models are in Tab. 4. The hyperparameters of the large model are tuned using the baseline ③ on selected languages in I, and the search range is shown in Tab. 4. All three tasks have the same two sets of hyperparameters. To ensure that ① has the same number of parameters as the other models, we decrease d_s in eq. (9) while for the rest of the models $d_s = 3d_h$. Additionally, we use a linear mapping to merge $\mathbf{e}^{(dec)}(y_{i-1})$ and $\bar{\mathbf{c}}_{i-1}$ in eq. (14) instead of concatenation. The output of the linear mapping has the same dimension as $\mathbf{e}^{(dec)}(y_{i-1})$, ensuring that the RNN has the same size.

Ⓜ has quite a different architecture: The input of the decoder RNN is the concatenation of the previously predicted word embedding, the encoder's hidden state at a specific step, and in the case of I, the encoding of the morphological tag. Differing from Aharoni and Goldberg (2017), we concatenate all attributes' embeddings (0 for attributes that are not applicable) and merge them with a linear mapping. The dimension of the merged vector and attributes vector are d_e. To ensure that it has the same number of parameters as the rest of the model, we increase the hidden size of the decoder RNN.

Optimization. We train the model with Adam (Kingma and Ba, 2014) with an initial learning rate of 0.001. We halve the learning rate whenever the development log-likelihood doesn't improve. We stop after the learning rate dips to 1×10^{-5}. We save all models after each epoch and select the model with best development performance. We train the model for at most 50 epochs, though all the experiments stop early. We train on G, T, and I with batch sizes of 20, 50 and 20, respectively. We notice in the experiments that the training of ① and Ⓤ is quite unstable with the large model, probably because of the longer chain of gradient information flow. We apply gradient clipping to the large model with maximum gradient norm 5.

REINFORCE. In the REINFORCE training of Ⓡ and ②, we sample 2 and 4 positions at each time step for the small and large model, respectively. The latter is tuned on selected languages in I with search range {2,3,4,5}. To stabilize the training, we apply a baseline with a moving average reward and

---

[2] Compared to greedy decoding with an average error rate of 20.1% and an average edit distance of 0.385, beam search with beam size 5 gets a slightly better edit distance of 0.381 while hurting the error rate with 20.2%.

[3] Ar–En, En–Ba, En–Hi, En–Ja, En–Ka, En–Ko, En–Pe, En–Ta, En–Th, Jn–Jk and Th–En.

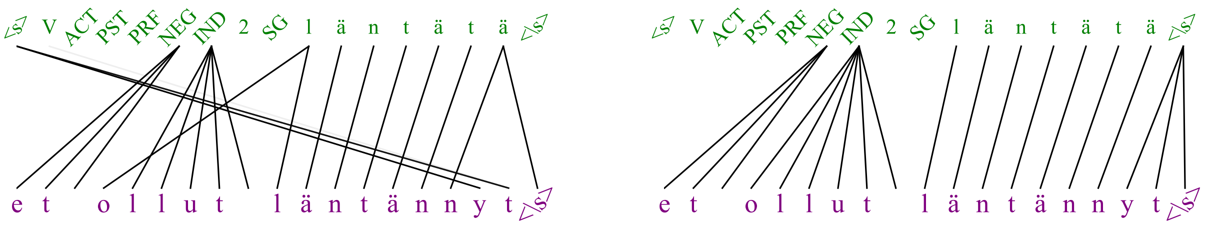

Figure 3: Attention-weight (③; left) and alignment distribution (④; right) of Finnish in I. Both models predict correctly.

discount factor of 0.9, similar to Xu et al. (2015).

8.3 Experimental Findings

Finding #1: Effect of Input Feeding. By comparing ③ and ④ against ① and ② in Tab. 3, we find input feeding hurts performance in all settings and all tasks. This runs in contrast to the reported results of Luong et al. (2015), but they experiment on machine translation, rather than character-level transduction. This validates our independence assumption about the alignment distribution.

Finding #2: Soft Attention vs. Hard Attention. Training with REINFORCE hurts the performance of the hard attention model; compare ① and ② (trained with REINFORCE), in Tab. 3. On the other hand, training with exact marginalization causes the hard attention model to outperform soft attention model in nearly all settings; compare ③ and ④ in Tab. 3. This comparison shows that hard attention outperforms soft attention in character-level string transduction when trained with exact marginalization.

Finding #3: Non-monotonicity vs. Monotonicity. The monotonic model Ⓜ underperforms compared to non-monotonic models ③ in Tab. 3 except for one setting. It performs slightly worse on T and G due to the many-to-one alignments in the data and the fact that Aharoni and Goldberg (2017) can only use the hidden vector of the final element of the span in a many-to-one alignment to directly predict the one target element. The current state-of-the-art systems for character-level string transduction are non-monotonic models, despite the tasks' seeming monotonicity; see §8.4.

Finding #4: Approximate Hard Attention. Given our development of an exact marginalization method for neural models with hard attention, a natural question to ask is how much exact marginalization helps during learning. By comparing ④ and Ⓡ in Tab. 3, we observe that training with exact marginalization clearly outperforms training under stochastic approximation in every setting and on every dataset. We also observe that exact marginalization allows faster convergence, since training with REINFORCE is quite unstable where some runs seemingly to get stuck.

	NETtalk				CMUDict			
	③		④		③		④	
	✓	✗	✓	✗	✓	✗	✓	✗
Monotonic	18742	1230	18823	1172	95824	17294	96176	17159
Non-monotonic	31	5	12	1	158	162	37	66

Table 5: Breakdown of correct and incorrect predictions of monotonic and non-monotonic alignments of ③ and ④ in G, derived from the soft attention weights and the hard alignment distribution

Finding #5: Controlling for Parameters. Input feeding yields a more expressive model, but also leads to an increase in the number of parameters. Here, we explore what effect this has on the performance of the models. In their ablation, Luong et al. (2015) did not control the number of parameters when adding input feeding. The total number of parameters of Ⓤ is 1.679M for the small setting and 10.541M for the large setting, which has 40% and 22.3% more parameters than the controlled setting. By comparing ① and Ⓤ in Tab. 3, we find that the increase in parameters, rather than the increase in expressivity explains the success of input feeding.

8.4 Visualization and Error Analysis

We hypothesize that even though the model is non-monotonic, it can learn monotonic alignment with flexibility if necessary, giving state-of-the-art results on many seemingly monotonic character-level string transduction task. To show more insights, we compare the best soft attention model (③) against the best hard alignment model (④) on G by showing the confusion matrix of each model in Tab. 5. An alignment is non-monotonic when alignment edges predicted by the model cross. There is an edge connecting x_j and y_i if the attention weight or hard alignment distribution $\alpha_j(i)$ is larger than 0.1. We find that the better-performing transducers are more monotonic, and most learned alignments are monotonic. The results indicate that there are a few transductions that are indeed non-monotonic in the dataset. However, the number is so few that this does not entirely explain why non-monotonic models outperform the monotonic models. We speculate this lies in the architecture of Aharoni and Goldberg (2017), which does not permit many-to-one alignments, while monotonic alignment learned by the non-monotonic model is more flexible. Future work will investigate this.

In Fig. 3, we visualize the soft attention weights (③) and the hard alignment distribution (④) side by side. We observe that the hard alignment distribution is more interpretable, with a clear boundary when predicting the prefixes.

9 Conclusion

We exhibit an efficient dynamic program for the exact marginalization of all non-monotonic alignments in a neural sequence-to-sequence model. We show empirically that exact marginalization helps over approximate inference by REINFORCE and that models with hard, non-monotonic alignment outperform those with soft attention.

Acknowledgements. The last author was supported by an NDSEG fellowship and a Facebook Fellowship.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, volume abs/1409.0473.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1975–1985.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 876–885, San Diego, California. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 561–564. IEEE.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for gpus. In *ICML*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. *Neural Computation*, 9(8):1735–1780.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 555–560, Berlin, Germany. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn. 2009. *Statistical Machine Translation*. Cambridge University Press.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633, San Diego, California. Association for Computational Linguistics.

Kenneth L. Rehg and Damian G. Sohl. 1981. *Ponapean Reference Grammar*. University of Hawaii Press.

Mihaela Rosca and Thomas Breuel. 2016. Sequence-to-sequence neural network models for transliteration. *arXiv preprint arXiv:1610.09565*.

Terrence J. Sejnowski and Charles R. Rosenberg. 1987. Parallel networks that learn to pronounce english text. *Complex Systems*, 1.

Xing Shi and Kevin Knight. 2017. Speeding up neural machine translation decoding by shrinking run-time vocabulary. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 574–579.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 3104–3112.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA.

R.L. Weide. 1998. The carnegie mellon pronouncing dictionary.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, pages 2048–2057.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. In *INTERSPEECH 2015*, pages 3330–3334, Dresden, Germany.

Min Zhang, Haizhou Li, Rafael E. Banchs, and A. Kumaran. 2015. Whitepaper of news 2015 shared task on machine transliteration. In *NEWS@ACL*.

A MFS

$$
\begin{aligned}
LCS(c, r) &= \frac{1}{2}(|c| + |r| - ED(c, r)) \\
R_i &= \frac{LCS(c_i, r_i)}{|r_i|} \\
P_i &= \frac{LCS(c_i, r_i)}{|c_i|} \\
FS_i &= 2\frac{R_i \times P_i}{R_i + P_i}
\end{aligned}
$$

where r_i and c_i is the i-th reference and prediction,

B Full breakdown of experiments

A full breakdown of G and T can be found in Tab. 6 and Tab. 7, respectively. A full breakdown of I can be found in Tab. 8 and Tab. 9.

	Small													
	①		Ⓤ		②		③		④		Ⓡ		Ⓜ	
	WER	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**
CMUDict	36.2	0.086	31.0	0.074	35.1	0.083	30.8	0.073	30.5	0.072	31.2	0.074	32.0	0.075
NETtalk	31.2	0.075	30.2	0.075	29.6	0.074	29.8	0.074	28.8	0.073	30.3	0.078	35.7	0.088
	Large													
	①		Ⓤ		②		③		④		Ⓡ		Ⓜ	
	WER	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**
CMUDict	32.3	0.076	31.4	0.073	36.7	0.087	30.5	0.073	29.8	0.071	31.8	0.077	30.5	0.072
NETtalk	29.3	0.071	29.4	0.072	29.5	0.075	26.8	0.068	26.6	0.066	27.7	0.071	29.3	0.072

Table 6: Full breakdown of G2P

	Small													
	①		Ⓤ		②		③		④		Ⓡ		Ⓜ	
	ACC	**MFS**	**ACC**	**MFS**	**ACC**	**MFS**	**ACC**	**MFS**	**ACC**	**MFS**	**ACC**	**MFS**	**ACC**	**MFS**
ArEn	54.9	0.954	54.8	0.953	53.9	0.954	53.5	0.951	56.6	0.954	53.9	0.950	60.1	0.980
EnBa	38.4	0.916	38.9	0.915	38.5	0.914	37.6	0.918	39.7	0.918	38.0	0.909	37.4	0.961
EnHi	42.4	0.922	44.0	0.925	40.8	0.921	46.1	0.927	43.8	0.926	43.6	0.924	43.0	0.967
EnJa	40.5	0.871	40.6	0.868	35.4	0.853	41.2	0.872	41.3	0.872	35.2	0.852	39.4	0.952
EnKa	34.8	0.910	35.5	0.912	33.1	0.909	37.8	0.913	36.0	0.909	34.9	0.907	35.6	0.960
EnKo	52.4	0.861	51.8	0.857	49.1	0.850	54.4	0.861	54.9	0.867	48.9	0.849	47.5	0.958
EnPe	28.3	0.899	32.6	0.908	30.8	0.903	34.8	0.911	30.5	0.901	29.6	0.898	34.7	0.964
EnTa	36.8	0.921	38.7	0.925	36.7	0.918	38.5	0.923	36.2	0.923	37.5	0.921	37.5	0.963
EnTh	42.4	0.909	42.6	0.907	37.0	0.892	42.8	0.907	42.1	0.906	37.1	0.890	40.1	0.954
JnJk	18.1	0.717	18.1	0.717	13.4	0.693	15.3	0.706	18.1	0.716	15.5	0.705	12.7	0.933
ThEn	39.2	0.912	40.2	0.913	31.1	0.882	38.9	0.911	38.6	0.912	33.6	0.897	38.3	0.960
	Large													
	①		Ⓤ		②		③		④		Ⓡ		Ⓜ	
	ACC	**MFS**	**ACC**	**MFS**	**ACC**	**MFS**	**ACC**	**MFS**	**ACC**	**MFS**	**ACC**	**MFS**	**ACC**	**MFS**
ArEn	52.2	0.954	54.3	0.954	0.7	0.682	55.4	0.954	55.6	0.953	57.1	0.955	59.7	0.979
EnBa	39.0	0.914	39.2	0.918	40.4	0.917	38.5	0.916	38.2	0.917	37.8	0.912	37.7	0.962
EnHi	42.0	0.923	43.0	0.926	38.9	0.914	45.7	0.929	46.1	0.928	40.7	0.916	45.0	0.968
EnJa	40.8	0.873	40.9	0.872	37.8	0.860	41.6	0.875	40.6	0.872	39.1	0.864	41.1	0.953
EnKa	36.2	0.913	37.9	0.914	35.0	0.909	37.5	0.913	38.6	0.915	38.2	0.913	39.2	0.961
EnKo	53.3	0.868	53.1	0.865	50.7	0.858	53.9	0.866	55.3	0.868	49.7	0.850	50.1	0.961
EnPe	34.0	0.911	34.6	0.913	32.5	0.906	34.2	0.912	35.3	0.911	33.4	0.911	34.3	0.964
EnTa	39.1	0.925	37.6	0.922	32.5	0.901	38.5	0.925	40.2	0.927	37.1	0.919	40.3	0.965
EnTh	43.6	0.910	43.7	0.909	32.5	0.869	43.7	0.909	43.9	0.910	40.0	0.897	41.3	0.955
JnJk	18.1	0.721	18.4	0.721	0.1	0.483	17.2	0.721	17.6	0.720	0.2	0.458	12.6	0.934
ThEn	40.2	0.915	40.3	0.915	37.7	0.909	39.2	0.915	40.3	0.916	32.9	0.897	39.8	0.962

Table 7: Full breakdown of NEWS2015

	Small													
	①		Ⓤ		②		③		④		Ⓡ		Ⓜ	
	ACC	**MLD**	**ACC**	**MLD**	**ACC**	**MLD**	**ACC**	**MLD**	**ACC**	**MLD**	**ACC**	**MLD**	**ACC**	**MLD**
albanian-high	97.2	0.048	98.5	0.023	97.9	0.043	98.1	0.031	98.1	0.045	98.5	0.029	94.5	0.150
arabic-high	89.2	0.396	79.1	0.792	91.0	0.419	90.4	0.360	91.7	0.377	90.2	0.438	89.1	0.340
armenian-high	94.6	0.106	95.3	0.086	90.1	0.214	95.2	0.080	95.5	0.080	93.7	0.126	94.4	0.108
basque-high	100.0	0.000	100.0	0.000	100.0	0.000	99.0	0.010	97.0	0.060	100.0	0.000	95.0	0.140
bengali-high	98.0	0.060	98.0	0.060	99.0	0.030	99.0	0.050	97.0	0.090	98.0	0.080	98.0	0.040
bulgarian-high	88.9	0.165	88.3	0.188	96.1	0.067	94.1	0.101	93.9	0.115	95.4	0.077	96.5	0.058
catalan-high	96.8	0.083	96.9	0.083	96.8	0.075	97.5	0.063	97.2	0.073	97.3	0.068	96.3	0.076
czech-high	90.3	0.170	92.3	0.145	90.6	0.167	91.9	0.157	92.2	0.139	78.2	0.468	91.4	0.152
danish-high	88.9	0.166	88.9	0.174	90.1	0.151	89.1	0.170	90.2	0.148	91.4	0.132	92.6	0.118
dutch-high	93.7	0.112	94.8	0.100	95.2	0.090	95.2	0.090	94.9	0.086	94.2	0.097	94.8	0.093
english-high	96.1	0.077	96.1	0.071	90.4	0.203	96.5	0.074	95.7	0.078	95.3	0.087	96.3	0.062
estonian-high	95.2	0.109	96.0	0.078	96.4	0.070	96.3	0.090	96.8	0.079	96.2	0.091	93.0	0.145
faroese-high	79.9	0.420	79.2	0.390	78.4	0.449	79.5	0.413	82.9	0.365	81.6	0.392	82.5	0.333
finnish-high	86.6	0.318	81.9	0.278	86.5	0.216	88.2	0.202	90.2	0.271	84.1	0.311	86.1	0.227
french-high	84.5	0.291	85.3	0.270	85.2	0.292	83.8	0.317	85.7	0.262	82.3	0.332	86.5	0.253
georgian-high	97.6	0.039	95.4	0.083	97.8	0.037	97.9	0.113	97.5	0.046	98.2	0.039	97.3	0.038
german-high	89.6	0.244	88.4	0.272	88.1	0.282	89.6	0.257	89.3	0.179	83.7	0.381	88.9	0.276
haida-high	97.0	0.040	98.0	0.030	98.0	0.030	99.0	0.020	97.0	0.040	98.0	0.030	92.0	0.150
hebrew-high	99.0	0.010	98.8	0.013	99.1	0.010	97.5	0.027	97.8	0.027	97.8	0.026	98.7	0.016
hindi-high	95.1	0.482	99.9	0.002	100.0	0.000	100.0	0.000	100.0	0.000	100.0	0.000	99.4	0.014
hungarian-high	83.4	0.338	83.9	0.336	85.2	0.333	82.5	0.372	82.3	0.381	83.2	0.444	83.0	0.367
icelandic-high	82.2	0.333	82.0	0.350	84.1	0.305	84.5	0.304	86.3	0.286	84.4	0.296	84.5	0.300
irish-high	87.4	0.387	84.4	0.454	89.0	0.333	87.9	0.351	90.6	0.289	88.3	0.332	88.5	0.335
italian-high	96.1	0.101	87.2	0.251	96.0	0.099	95.5	0.111	95.7	0.106	95.5	0.105	94.6	0.120
khaling-high	99.2	0.008	98.9	0.018	98.7	0.016	98.0	0.030	98.7	0.018	98.7	0.024	98.1	0.028
kurmanji-high	94.2	0.123	93.7	0.101	92.3	0.184	93.2	0.126	93.8	0.098	93.1	0.143	94.0	0.074
latin-high	65.4	0.578	65.9	0.591	69.6	0.516	70.1	0.476	72.1	0.458	68.6	0.503	70.7	0.450
latvian-high	94.7	0.084	95.3	0.071	93.7	0.104	95.1	0.090	95.5	0.081	94.2	0.101	93.6	0.100
lithuanian-high	87.0	0.178	87.8	0.247	84.9	0.233	86.9	0.196	89.1	0.162	87.4	0.201	80.9	0.258
lower-sorbian-high	94.6	0.111	93.7	0.112	94.8	0.100	93.4	0.138	95.2	0.096	94.8	0.103	94.2	0.108
macedonian-high	94.1	0.088	94.2	0.089	95.3	0.073	90.7	0.164	93.6	0.102	94.7	0.087	94.9	0.094
navajo-high	84.9	0.446	84.6	0.461	81.2	0.468	86.2	0.332	88.5	0.268	85.1	0.356	79.8	0.450
northern-sami-high	93.9	0.112	94.2	0.099	94.8	0.125	93.6	0.145	95.4	0.089	93.2	0.143	91.8	0.154
norwegian-bokmal-high	86.4	0.220	87.6	0.293	88.4	0.193	89.7	0.172	90.0	0.158	88.2	0.198	90.9	0.156
norwegian-nynorsk-high	71.8	0.454	78.1	0.363	76.5	0.392	77.9	0.378	77.4	0.379	81.0	0.324	88.4	0.197
persian-high	99.4	0.013	99.4	0.013	99.4	0.012	99.1	0.016	98.9	0.017	99.2	0.017	96.8	0.064
polish-high	89.8	0.245	86.5	0.306	82.2	0.424	90.7	0.237	89.9	0.258	88.9	0.297	90.8	0.198
portuguese-high	98.0	0.032	96.3	0.060	96.4	0.056	98.5	0.034	98.9	0.024	97.9	0.036	98.0	0.034
quechua-high	99.4	0.013	97.9	0.045	98.7	0.040	98.2	0.053	99.6	0.020	98.2	0.058	96.4	0.087
romanian-high	83.3	0.649	83.6	0.454	75.6	0.711	81.6	0.482	84.8	0.432	83.8	0.473	84.8	0.404
russian-high	89.4	0.263	86.8	0.303	71.2	0.960	90.1	0.271	90.1	0.242	87.7	0.312	89.8	0.233
serbo-croatian-high	89.6	0.209	86.8	0.241	89.0	0.244	87.2	0.256	89.5	0.236	89.9	0.225	91.8	0.159
slovak-high	90.5	0.155	89.8	0.168	87.2	0.212	91.3	0.145	89.6	0.163	90.2	0.158	92.6	0.128
slovene-high	94.7	0.101	94.1	0.110	95.3	0.094	96.3	0.070	96.2	0.080	94.9	0.114	95.9	0.068
sorani-high	89.3	0.131	90.0	0.124	89.2	0.127	87.7	0.148	88.4	0.149	88.9	0.143	84.0	0.222
spanish-high	95.9	0.080	94.3	0.104	96.0	0.083	95.1	0.105	95.5	0.099	94.6	0.118	94.2	0.099
swedish-high	87.1	0.212	87.2	0.214	87.8	0.208	88.9	0.167	88.7	0.185	69.6	0.880	90.2	0.165
turkish-high	97.1	0.069	96.9	0.078	95.6	0.104	96.8	0.073	95.4	0.099	95.5	0.127	93.3	0.146
ukrainian-high	90.4	0.171	91.1	0.157	89.0	0.188	90.3	0.168	91.8	0.141	91.6	0.139	92.9	0.111
urdu-high	99.4	0.009	99.6	0.007	99.2	0.012	99.5	0.009	99.4	0.009	99.7	0.005	98.2	0.027
welsh-high	96.0	0.070	96.0	0.080	98.0	0.040	96.0	0.070	99.0	0.030	98.0	0.040	98.0	0.040

Table 8: Full breakdown of SIGMORPHON2017 with small model

	Large													
	①		Ⓤ		②		③		④		Ⓡ		Ⓜ	
	ACC	**MLD**	**ACC**	**MLD**	**ACC**	**MLD**	**ACC**	**MLD**	**ACC**	**MLD**	**ACC**	**MLD**	**ACC**	**MLD**
albanian-high	97.9	0.037	96.3	0.066	98.5	0.024	98.0	0.030	98.5	0.028	98.8	0.021	96.1	0.123
arabic-high	90.4	0.496	89.8	0.397	88.4	0.456	89.8	0.358	92.3	0.352	91.9	0.437	90.8	0.267
armenian-high	94.3	0.112	95.6	0.166	94.3	0.106	94.9	0.078	95.8	0.075	94.6	0.095	93.6	0.110
basque-high	100.0	0.000	100.0	0.000	99.0	0.030	100.0	0.000	100.0	0.000	100.0	0.000	99.0	0.020
bengali-high	99.0	0.020	99.0	0.050	99.0	0.050	99.0	0.050	99.0	0.050	95.0	0.150	98.0	0.050
bulgarian-high	95.0	0.079	91.9	0.129	93.6	0.095	96.5	0.059	96.8	0.052	96.5	0.061	96.8	0.057
catalan-high	95.8	0.102	96.9	0.074	97.1	0.068	96.9	0.064	97.9	0.056	97.4	0.064	96.3	0.074
czech-high	89.2	0.184	87.4	0.209	87.8	0.241	90.7	0.176	92.7	0.133	89.7	0.188	92.0	0.140
danish-high	88.3	0.167	87.0	0.282	89.3	0.166	88.6	0.159	91.9	0.121	91.7	0.127	92.2	0.121
dutch-high	93.9	0.109	93.3	0.120	94.2	0.107	94.6	0.099	95.7	0.082	95.8	0.075	95.6	0.078
english-high	95.9	0.082	95.7	0.239	93.1	0.151	95.5	0.081	96.3	0.069	96.0	0.078	96.4	0.055
estonian-high	95.5	0.235	96.9	0.069	96.7	0.069	96.3	0.082	97.6	0.064	96.1	0.086	92.7	0.140
faroese-high	80.8	0.388	80.7	0.468	80.1	0.374	83.0	0.365	84.3	0.327	82.3	0.371	84.5	0.314
finnish-high	89.7	0.323	86.5	0.196	26.2	4.562	88.5	0.212	92.2	0.137	90.4	0.169	88.0	0.197
french-high	82.2	0.329	83.3	0.316	84.0	0.305	82.9	0.335	85.5	0.274	84.7	0.296	85.6	0.273
georgian-high	97.5	0.043	96.8	0.060	96.9	0.056	96.8	0.064	98.2	0.027	98.4	0.022	98.6	0.018
german-high	87.7	0.459	82.4	0.505	76.5	0.751	87.4	0.309	91.3	0.141	88.7	0.233	89.4	0.244
haida-high	98.0	0.030	98.0	0.030	98.0	0.030	97.0	0.040	98.0	0.030	98.0	0.030	95.0	0.100
hebrew-high	98.5	0.019	99.1	0.011	98.5	0.017	98.6	0.015	98.4	0.018	98.7	0.016	98.5	0.018
hindi-high	100.0	0.000	99.7	0.099	99.9	0.001	99.8	0.002	99.9	0.003	100.0	0.000	99.8	0.006
hungarian-high	82.7	0.361	82.0	0.362	81.7	0.400	79.0	0.437	84.1	0.347	82.0	0.386	84.7	0.329
icelandic-high	82.6	0.341	84.4	0.309	82.9	0.338	84.7	0.301	88.0	0.244	87.6	0.248	87.6	0.245
irish-high	85.8	0.377	86.4	0.391	81.8	0.512	88.7	0.423	90.5	0.254	88.3	0.344	89.6	0.304
italian-high	95.8	0.101	96.1	0.089	95.6	0.113	96.6	0.081	96.5	0.084	96.2	0.091	96.0	0.104
khaling-high	99.4	0.006	99.2	0.009	99.4	0.006	99.3	0.009	99.5	0.005	99.4	0.009	97.8	0.031
kurmanji-high	93.0	0.162	93.2	0.268	91.3	0.183	93.1	0.128	93.0	0.089	92.7	0.146	94.0	0.076
latin-high	71.6	0.518	69.6	0.513	70.4	0.485	78.0	0.371	78.4	0.361	76.2	0.392	74.7	0.378
latvian-high	80.8	0.524	91.5	0.133	94.6	0.095	94.3	0.097	96.3	0.056	95.7	0.078	95.2	0.068
lithuanian-high	86.4	0.368	85.4	0.394	87.9	0.173	89.7	0.150	90.6	0.126	90.7	0.139	89.4	0.149
lower-sorbian-high	93.9	0.118	93.7	0.208	95.2	0.094	95.2	0.094	95.1	0.100	96.3	0.073	95.6	0.083
macedonian-high	90.8	0.126	92.7	0.111	94.4	0.085	93.7	0.097	95.9	0.067	94.6	0.085	93.8	0.110
navajo-high	88.1	0.268	86.9	0.312	90.8	0.198	88.3	0.435	91.3	0.201	88.6	0.279	84.5	0.359
northern-sami-high	94.9	0.129	95.8	0.082	95.3	0.121	95.9	0.087	97.5	0.075	96.4	0.103	95.0	0.090
norwegian-bokmal-high	84.4	0.331	85.1	0.403	88.4	0.272	88.1	0.199	88.7	0.190	89.6	0.178	91.0	0.152
norwegian-nynorsk-high	73.3	0.440	76.2	0.408	78.2	0.376	79.4	0.354	80.4	0.345	82.6	0.314	89.3	0.182
persian-high	99.6	0.006	99.2	0.011	99.3	0.108	99.5	0.014	99.3	0.014	99.6	0.011	96.7	0.066
polish-high	85.3	0.406	85.7	0.369	88.5	0.282	88.4	0.366	89.7	0.248	89.2	0.251	90.2	0.193
portuguese-high	97.4	0.041	97.4	0.042	98.2	0.034	97.9	0.037	98.3	0.032	98.9	0.023	98.8	0.028
quechua-high	98.9	0.019	97.9	0.059	97.8	0.048	98.6	0.037	98.9	0.032	98.8	0.037	97.7	0.057
romanian-high	84.9	0.594	83.4	0.568	47.1	2.794	85.1	0.457	86.7	0.400	86.8	0.422	86.4	0.450
russian-high	88.0	0.353	85.8	0.458	86.9	0.324	89.8	0.432	90.5	0.244	90.3	0.244	91.2	0.220
serbo-croatian-high	84.9	0.307	85.0	0.277	88.8	0.323	88.8	0.236	90.9	0.187	82.0	0.426	91.4	0.185
slovak-high	87.7	0.214	89.4	0.270	89.2	0.264	89.6	0.186	92.1	0.126	90.4	0.163	92.6	0.129
slovene-high	94.6	0.096	93.2	0.222	94.6	0.102	95.2	0.078	96.1	0.073	94.9	0.097	96.4	0.063
sorani-high	88.8	0.138	88.4	0.144	88.9	0.133	89.4	0.123	90.3	0.121	90.1	0.120	85.9	0.180
spanish-high	94.3	0.092	95.8	0.163	96.1	0.072	96.1	0.075	96.7	0.056	96.2	0.072	95.4	0.079
swedish-high	85.7	0.237	85.0	0.329	79.4	0.529	85.9	0.235	90.9	0.157	89.6	0.170	92.2	0.132
turkish-high	95.9	0.103	94.9	0.226	94.8	0.118	92.4	0.172	97.0	0.063	94.8	0.121	94.3	0.128
ukrainian-high	90.4	0.170	90.3	0.161	90.8	0.148	92.5	0.126	92.9	0.116	93.1	0.114	91.4	0.144
urdu-high	99.3	0.012	99.4	0.010	99.0	0.015	99.6	0.007	99.3	0.014	99.6	0.006	99.1	0.013
welsh-high	96.0	0.060	98.0	0.050	98.0	0.040	97.0	0.050	97.0	0.070	97.0	0.060	97.0	0.060

Table 9: Full breakdown of SIGMORPHON2017 with large model

Speed Reading: Learning to Read ForBackward via Shuttle

Tsu-Jui Fu
Academia Sinica
No. 128, Sec. 2, Academia Rd.
Taipei 11529, Taiwan
`s103062110@m103.nthu.edu.tw`

Wei-Yun Ma
Academia Sinica
No. 128, Sec. 2, Academia Rd.
Taipei 11529, Taiwan
`ma@iis.sinica.edu.tw`

Abstract

We present LSTM-Shuttle, which applies human speed reading techniques to natural language processing tasks for accurate and efficient comprehension. In contrast to previous work, LSTM-Shuttle not only reads shuttling forward but also goes back. Shuttling forward enables high efficiency, and going backward gives the model a chance to recover lost information, ensuring better prediction. We evaluate LSTM-Shuttle on sentiment analysis, news classification, and cloze on IMDB, Rotten Tomatoes, AG, and Children's Book Test datasets. We show that LSTM-Shuttle predicts both better and more quickly. To demonstrate how LSTM-Shuttle actually behaves, we also analyze the shuttling operation and present a case study.

1 Introduction

Recently, recurrent neural networks (RNNs) and long short-term memory (LSTM) cells and gate recurrent unit (GRU) cells have achieved great success and are increasingly being applied in nature language processing tasks, e.g., part-of-speech (POS) tagging (Wang et al., 2015), named-entity recognition (Chiu and Nichols, 2015), sentiment analysis (Zhang et al., 2018), document classification (Kim, 2014; Le and Mikolov, 2014a), cloze (Srinivasan et al., 2018), machine translation (Bahdanau et al., 2015), dialogue modeling (Mei et al., 2017), document summarization (Allahyari et al., 2017), automatic knowledge extraction (Durme and Schubert, 2008), and question answering (Chen et al., 2017).

Those tasks all call for text comprehension techniques. To solve these tasks, the proposed models read all the text available. That is, models read every token or word of the text from beginning to end. However, for some classification tasks, it is not necessary to treat each individual word equally. Take, for example, sentiment analysis: sentences such as "this movie is amazing" or "too boring" are sufficient to judge a sentiment without reading the entire comment. In addition, the fact that texts are often written redundantly also motivates reading selectively, especially for certain NLP tasks.

In terms of human reading habits, although people tend to skim text when reading a newspaper or a novel, this does not significantly impair comprehension. Speed reading, a reading technique, is used to improve one's ability to read quickly. Work has been done on modeling skimming behavior along with the original sequence modeling RNN. LSTM-Jump (Yu et al., 2017) predicts how many words to skim based on the RNN hidden state. They show that neglecting some words in a document does not greatly harm prediction accuracy but does significantly accelerate the process. They also show that for certain tasks such as cloze, skimming even outperforms traditional methods. In addition, (Yu et al., 2018) use RNN hidden states to decide when to stop. If the RNN judges it has achieved sufficient comprehension of the context, it stops early and produce the answer directly.

However, strictly speakly, simply skimming and stopping early is not speed reading. For example, imagine that during a reading test, we first read the question and then the long article. We read quickly and skip some information. What do we do if we encounter text that we don't understand? We go back, read the previous text, and try to fill in the gaps in our understanding. For speed reading, going back – or "reading backward" – is likewise important as it helps us to recover lost information or correct misunderstandings, leading to better comprehension of long documents. In fact, reading backward increases rather than decreases reading speed: given the opportunity to go back to correct

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4439–4448
Brussels, Belgium, October 31 - November 4, 2018.

misunderstandings, we skim more words and thus read faster without reducing our comprehension.

In this paper, we propose LSTM-Shuttle, which teaches the RNN model to speed read by modeling both forward and backward reading behavior. We evaluate the proposed method on sentiment analysis, document classification, and cloze tasks. We use IMDB (L. et al., 2011) and Rotten Tomatoes (Pang and Lee, 2005) as sentiment analysis datasets, AG New (Shang et al., 2015) as a document classification dataset, and Facebook Children's Book Test (Hill et al., 2015) as a cloze dataset. The experiments show that the proposed method achieves better prediction performance and reads faster at the same time, in comparison with the LSTM baseline (Hochreiter and Schmidhuber, 1997) and LSTM-Jump. We also analyze the shuttling behavior under different settings, proving that reading forward and backward does help in reading.

2 Related Work

The proposed method is inspired mainly by LSTM-Jump (Yu et al., 2017), which predicts how many words should be neglected, accelerating the reading process using RNN. Both their work and ours is related to the idea of learning visual attention per (Mnih et al., 2015), where a recurrent model is used to decide which image part to watch seriatim. They train the model end-to-end using the REINFORCE algorithm (Williams, 1992) and sample from a continuous Gaussian distribution. The difference between their and our methods is that we sample from a discrete distribution to reflect the properties of text and image.

Many recent natural language processing applications have explored the idea of filtering irrelevant content. As in our work, instead of skimming some words, (Seo et al., 2018) consider all words but use a small RNN for irrelevant words and the original large RNN for relevant ones. (Campos et al., 2018) also attempt to dynamically control the RNN's computational costs, but they instead control the number of units to update at each time step. In our method, in contrast, we skim words, directly setting the amount of computation to be zero, which streamlines the reading process.

From another perspective, (Yu et al., 2018) attempt to model early stopping behavior, deciding whether the model can answer confidently based on the hidden states. This is very effective for tasks such as question answering. To ensure accuracy, (Shen et al., 2016) focuses on early stopping after multiple passes, and (Shen et al., 2016) also using reinforcement learning to attempt to learn to reason. Both early stopping and our method adequately take into account research on sufficient comprehension. While, early stopping is not fast enough for classification, we can do better with text shuttling.

3 Main Idea

In this section, we describe the proposed LSTM-Shuttle, first presenting its architecture. Then, we show that due to the nondifferentiability of the shuttle mechanism, we apply a policy gradient (Sutton et al., 1999) to train it end-to-end. Finally, we present the implementation details and the inference approach.

3.1 Overview

As Fig. 1 illustrates, LSTM-Shuttle is based on an LSTM recurrent neural network to which is added an additional fully connected layer to predict forward or backward steps after a softmax distribution.

Given a text which denoted as $x_1, x_2, \ldots, x_L$ or $x_{1:L}$, LSTM-Shuttle first reads a fixed number of words sequentially and outputs the hidden state. Then, based on the hidden state, LSTM-Shuttle computes the shuttle softmax distribution over the forward or backward steps on $[-K, K]$. Given a negative step value, LSTM-Shuttle goes back to correct misunderstandings, and with a positive step value, LSTM-Shuttle speed reads, skimming unimportant words. After shuttling, LSTM-Shuttle reads words sequentially and then proceeds to shuttle again, iteratively. This process continues until one of the following occurs:

- The shuttle softmax samples a 0
- The number of shuttles exceeds the predefined upper limit
- The model has arrived at the last word

After stopping, the last hidden state is used to predict the desired task. The post-processing depends on the task. For instance, for classification, the hidden state generates a softmax distribution over the target class, and for cloze, it is used to find the correlation between the question article and each candidate answer. The detailed settings of each task are described in Section 4.

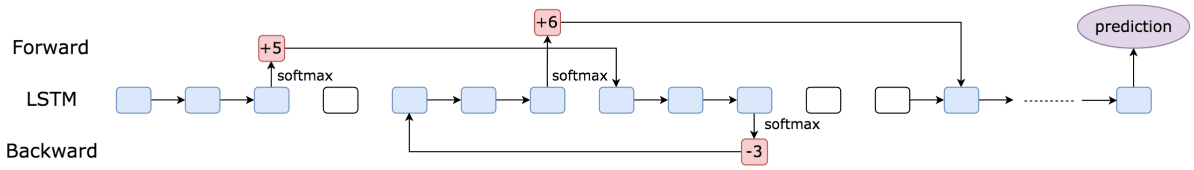

Figure 1: An overview of LSTM-Shuttle. In this example, we set the number of words read sequentially before shuttling $R = 2$, and the maximum shuttle size $K = 10$. The shuttle action is sampled from $[-K, K]$. After reading the entire text, the last hidden state is used to answer prediction. Note that when going backward, the shuttle step is counted before reading sequentially.

As with LSTM-Jump (Yu et al., 2017), we use the following notation:

- N: total number of allowed shuttles
- R: number of words to read before shuttling
- K: maximum shuttle size

Whereas K is a fixed hyperparameter during training, N and R can vary between training and testing. Note that as LSTM-Shuttle proceeds not only forwards but also backwards, the output shuttle size is $2K + 1$: K forward, K backward, and 1 for stopping. In contrast to LSTM-Jump (Yu et al., 2017), when going back, our shuttle step is counted before reading sequentially. In the example in Fig. 1, we set $R = 2$ and $K = 10$.

3.2 Training via Policy Gradient

In LSTM-Shuttle, there are two main parameter sets to compute: θ_R and θ_U. θ_R includes the RNN along with the output prediction parameters, and θ_U represents the parameters of the shuttle mechanism.

We compute θ_R via backpropagation directly by minimizing $J_1(\theta_R)$, the cross entropy loss, which is differentiable over θ_R and is the target objective function of the classification task.

However, this does not work for θ_U. Since cross entropy isn't differentiable over θ_U, we cannot use backpropagation to compute θ_U. Therefore, we recast it as a reinforcement learning problem and apply a policy gradient to train it: we seek to maximize the reward function over θ_U via the following formulation.

We first denote $s_{1:T}$ as the shuttle action sequence when training with text $x_{1:L}$. Assuming that h_i is the hidden state of LSTM before the i-th shuttle s_i, it is a function of $s_{i:i-1}$ and thus can be denoted as $h_i(s_{1:i-1})$. Also, the shuttle action can be sampled from the distribution of $p(s_t|h_i(s_{1:t-1}); \theta_U)$, which is determined by the shuttle softmax. We have the final prediction after LSTM-Shuttle processes text $x_{1:L}$ under the current θ_U shuttle strategy. As with (Yu et al., 2017), we set the reward to 1 if the prediction is correct, and -1 otherwise.

$$R = \begin{cases} 1 & \text{if predicted correctly} \\ -1 & \text{otherwise} \end{cases}$$

The objective function of θ_U we seek to maximize is the expected reward under the distribution over θ_U shuttle strategy, i.e.,

$$J_2(\theta_U) = \mathbb{E}_{p(s_{1:T};\theta_U)}[R], \tag{1}$$

where $p(s_{1:T;\theta_U}) = \prod_i p(s_{1:i}|h_i(s_{1:i-1}; \theta_U)$.

To maximize the objective function, we must compute the gradient of Eq. (1). We compute an approximate gradient by running M examples with the REINFORCE algorithm (Williams, 1992):

$$\nabla_{\theta_U} J_2(\theta_U) = \sum_{i=1}^{T} \mathbb{E}_{p(s_{1:T;\theta_U})}[\nabla_{\theta_U} \log p(s_{1:i}|h_i; \theta_U) R]$$
$$\approx \frac{1}{M} \sum_{m=1}^{M} \sum_{i=1}^{T} [\nabla_{\theta_U} \log p(s^m_{1:i}|h^m_i; \theta_U) R^m],$$

where the superscript m denotes that it belongs to the m-th example. Eventually, the term $\nabla_{\theta_U} \log p(s_{1:i}|h_i; \theta_U)$ is computed by backpropagation as usual.

Though the approximation of $\nabla_{\theta_U} J_2(\theta_U)$ is unbiased, it may have very high variance (Williams, 1992). One common way to reduce this variance is to subtract a baseline value b from the reward function R, transforming the approximated gradient into

$$\nabla_{\theta_U} J_2(\theta_U) \approx$$
$$\frac{1}{M} \sum_{m=1}^{M} \sum_{i=1}^{T} [\nabla_{\theta_U} \log p(s^m_{1:i}|h^m_i; \theta_U) R^m - b^m_i].$$

Here we apply same bias strategy as (Lewis et al., 2017), treating the bias value b as the average reward from then until now.

The final objective function for LSTM-Shuttle to minimize is

$$J(\theta_R, \theta_U) = J_1(\theta_R) - J_2(\theta_U),$$

which is entirely differentiable and can be computed by standard backpropagation.

3.3 Implementation detail and Inference

To simulate negative step selection, which corresponds to reading backward in the shuttle action, we set the shuttle output dimension to $[0, 2K]$, where 0 maps to $-K$, 1 maps to $-(K-1)$, $\ldots$, K maps to $0, \ldots, 2K-1$ maps to $+(K-1)$, and $2K$ maps to $+K$.

We used the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 10^{-3} for all experiments. For a fair comparison with (Yu et al., 2017), the dropout rate between LSTM layers was set to 0.2 and the embedding dropout rate to 0.1. We implemented LSTM-Shuttle in PyTorch (Adam et al., 2017) on a GTX 1080Ti gpu.

During inference, we apply greedy sampling: we select the most probable shuttle step from the shuttle softmax distribution.

4 Experiments

In this section, we evaluate the proposed LSTM-Shuttle on three different tasks: sentiment analysis, news classification, and cloze on four different datasets. We use IMDB (L. et al., 2011) and Rotten Tomatoes (Pang and Lee, 2005) for sentiment analysis, AG (Shang et al., 2015) for news article classification, and Children's Book Test (Hill et al., 2015) for cloze. Table 1 contains statistics for the tasks and datasets in our experiments.

To show the improvement in not only accuracy but also efficiency, we compared LSTM-Shuttle with three baselines: vanilla LSTM (Hochreiter and Schmidhuber, 1997), LSTM-Jump (Yu et al., 2017), and bi-directional LSTM-Jump, as shown in Fig. 2. For a fair comparison, we trained LSTM-Shuttle with the same LSTM settings as LSTM-Jump. For example, for sentiment analysis on IMDB, LSTM-Jump was trained with R=20, K=40, and N=80; we trained LSTM-Shuttle with the same parameters. Vanilla LSTM is the traditional recurrent neural network using LSTM cells which reads the entire text and then outputs the prediction. LSTM-Jump has a skim mechanism which neglects some text. Bi-directional LSTM-Jump applies LSTM-Jump twice but starting from different directions, and concatenates the last hidden state for answer prediction. To shorten the presentation, for LSTM-Jump we selected only two results from the original paper directly on each dataset: one with the best accuracy and the other with the highest efficiency. This is to show the difference between reading in two directions and shuttling. The quantitative result of each dataset is shown in the following sections.

In addition to the quantitative results, we sought to investigate how the shuttle mechanism progresses in reality. We present the shuttle statistics for different K settings, and show that because of the backward mechanism which affords a chance to recover lost information, LSTM-Shuffle shuttles with larger steps, increasing the shuttle step size as it grows more and more confident in its prediction.

4.1 Sentiment Analysis on IMDB and Rotten Tomatoes

Sentiment analysis is a classic natural language processing task, in which we read an article and predict its latent sentiment as positive or negative. It is widely implied in many forms or questionnaires such as satisfaction surveys. Here we use IMDB (L. et al., 2011) and Rotten Tomatoes (Pang and Lee, 2005) as our sentiment analysis datasets.

4.1.1 IMDB Results

IMDB (L. et al., 2011), a well-known movie information website, also includes audience comments and their sentiments. It contains 25,000 training data and 25,000 testing data, where the average length is 241 words. Both baselines and LSTM-Shuttle used a single layer and 128 hidden units as LSTM cells. We used pre-trained word2vec embeddings (Le and Mikolov, 2014b) as initial word embeddings and trained it along with LSTM. For a comparison with the baselines, all models were trained under $R = 20$, $K = 40$, and $N = 5$. We also show the result of a larger shuttle step ($K = 75$) version of LSTM-Shuttle.

Table 2 shows the experimental results for IMDB, where the speedup ratio is compared with vanilla LSTM, conducted on a machine with a single GTX 1080Ti GPU. For bi-directional LSTM-Jump we used our own implementation, and the

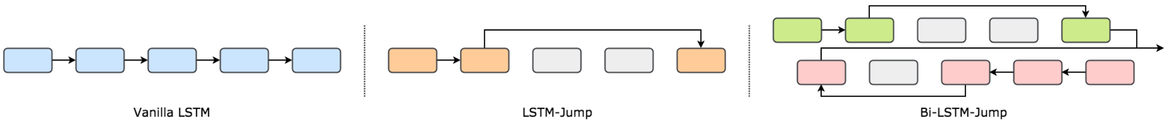

Figure 2: Baseline architectures. **Left:** Vanilla LSTM: reads entire text sequentially. **Middle:** LSTM-Jump: skims to neglect some words. **Right:** Bi-direction LSTM-Jump: skims in two directions, concatenates two latest hidden states.

Task	Dataset	Level	AvgLen	#train	#test	#class
Sentiment analysis	IMDB	word	241 words	21,143	25,000	2
Sentiment analysis	Rotten Tomatoes	word	22 words	8,835	1,030	2
News classification	AG	character	200 characters	101,851	7,600	4
Cloze	CBT-NE	sentence	20 sentences	120,769	2,500	10

Table 1: Tasks and datasets

Method	(R, K, N)	Accuracy	Speedup	Backward
LSTM	-	89.1%	1x	-
LSTM-Jump	(80, 40, 8)	89.4%*	1.64x	-
LSTM-Jump	(100, 40, 1)	88.0%	**2.54x***	-
Bi-LSTM-Jump	(80, 40, 8)	89.6%*	1.12x	-
Bi-LSTM-Jump	(100, 40, 1)	88.4%	2.33x*	-
LSTM-Shuttle	(20, 40, 5)	88.9%	2.43x	0.23
LSTM-Shuttle	(60, 40, 6)	**89.9**%*	2.08x	0.26
LSTM-Shuttle	(80, 40, 8)	89.7%	1.49x	0.27
LSTM-Shuttle	(100, 40, 1)	88.6%	2.46x*	0.12
LSTM-Shuttle	(50, **75**, 4)	89.7%*	2.27x	0.29
LSTM-Shuttle	(50, **75**, 2)	89.1%	2.45x*	0.21

Table 2: Sentiment analysis results on IMDB. * means the best accuracy or highest speedup for each method given the same setting of K.

"Backward" column in the table shows the frequency ratio of backward shuttles.

Under the same (R, K, N) setting, LSTM-Shuttle is a little slower than LSTM-Jump since the softmax size of K is larger, but the former yields better prediction. For bi-directional LSTM-Jump, since it applies LSTM-Jump twice, it predicts better than the original. However, we doubt whether it is worth sacrificing so much efficiency for such a small increase in prediction accuracy $(+0.2\%)$. Under the same (R, K, N), LSTM-Shuttle is more accurate and faster than bi-directional LSTM-Jump, even though $(80, 40, 8)$ is not in fact the best setting for LSTM-Shuttle. LSTM-Shuttle achieves the highest accuracy (89.9%) with 2.08× acceleration under $(60, 40, 6)$. Due to the shuttle mechanism that can go back, LSTM-Shuttle does not need to read many words before each shuttle, thus accelerating the overall reading process.

In general, the combination of (R, N) represents a trade-off between accuracy and efficiency. If we use a larger (R, N), the model reads more words and predicts better but more slowly. Otherwise, for a smaller (R, N), the model reads faster but yields predictions that are not as accurate. A similar tendency is found when it comes to the backward ratio. A smaller N means LSTM-Shuttle shuttles less often, so the model tends to read through as much as possible, making for a lower backward ratio. On the other hand, LSTM-Shuttle can shuttle many times so it is willing to go back to correct misunderstandings.

We also show the result for $K = 75$. With the larger shuttle step, fewer words are read before shuttling, which accelerates reading but has little impact on accuracy. LSTM-Shuttle achieves 89.7% with $2.27\times$ speedup under $(50, 75, 4)$ and 89.1% with $2.45\times$ times speedup under $(50, 75, 2)$: both settings yield both high accuracy and efficiency.

4.1.2 Rotten Tomatoes Results

The Rotten Tomatoes dataset (Pang and Lee, 2005) is to IMDB. We chose to use a two-layer LSTM and 256 hidden units, and again used the pre-trained word2vec embeddings (Le and Mikolov, 2014b). We trained all models under $R = 8$, $K = 10$, and $N = 3$.

The experimental results are shown in Table 3. LSTM-Shuttle achieves an accuracy of 79.5% with $1.55\times$ speedup; a higher efficiency version accelerates to $1.89\times$. These results demonstrate a similar trade-off tendency with different (R, N) combinations as those for IMDB. Since the average comment length in Rotten Tomatoes is short, we used lower shuttle times ($N = 2$) for better speed but also maintained high accuracy. Because

Method	(R, K, N)	Accuracy	Speedup	Backward
LSTM	-	79.1%	1x	-
LSTM-Jump	(7, 10, 4)	79.3%*	1.56x	-
LSTM-Jump	(9, 10, 2)	78.3%	1.94x*	-
Bi-LSTM-Jump	(7, 10, 4)	79.4%*	1.32x	-
Bi-LSTM-Jump	(9, 10, 2)	78.9%	1.54x*	-
LSTM-Shuttle	(7, 10, 4)	79.5%	1.52x	0.41
LSTM-Shuttle	(8, 10, 3)	79.5%*	1.55x	0.41
LSTM-Shuttle	(9, 10, 2)	79.3%	1.89x*	0.45
LSTM-Shuttle	(6, **20**, 3)	**79.8%***	1.74x	0.39
LSTM-Shuttle	(6, **20**, 2)	79.2%	**1.97x***	0.48

Table 3: Sentiment analysis results for Rotten Tomatoes. * means the best accuracy or highest speedup for each method given the same setting of K.

Method	(R, K, N)	Accuracy	Speedup	Backward
LSTM	-	88.1%	1x	-
LSTM-Jump	(30, 40, 5)	88.5%*	1.24x*	-
LSTM-Jump	(40, 40, 6)	87.4%	0.83x	-
Bi-LSTM-Jump	(30, 40, 5)	89.5%*	1.08x*	-
Bi-LSTM-Jump	(40, 40, 6)	88.4%	0.81x	-
LSTM-Shuttle	(20, 40, 5)	**90.1%***	1.34x*	0.27
LSTM-Shuttle	(30, 40, 5)	88.9%	1.16x	0.30
LSTM-Shuttle	(40, 40, 6)	88.4%	0.82x	0.34
LSTM-Shuttle	(20, **80**, 4)	89.8%	**1.63x***	0.26
LSTM-Shuttle	(30, **80**, 4)	**90.1%***	1.29x	0.28

Table 4: News classification result on AG. * means the best accuracy or highest speedup for each method given the same setting of K.

of the shorter comments and lower shuttle times, LSTM-Shuttle prefers to shuttle over almost the entire text from the beginning, trying to see the last part of a comment, and then goes back to the middle part; thus the backward ratio is much higher.

We also show the results under a larger K ($K =$ 20). On Rotten Tomatoes, a larger shuttle step seems more suitable. LSTM-Shuttle achieves the best accuracy (79.8%) with 1.74× high efficiency. A setting with fewer shuttles further accelerates up to 1.97× while maintaining a high accuracy of 79.2%.

4.2 News Article Classification on AG dataset

News classification is a common application of document comprehension. Given a news article, the model must recognize which field it belongs to. Modern topic classification is applied on different target sources such as blog posts. We used AG (Shang et al., 2015) as news article classification dataset in our experiments.

4.2.1 Result on AG dataset

We used the subset constructed by (Shang et al., 2015) for classification at the character level. AG contains news covering four topics (World, Sports, Business, Sci/Tech), each of which includes 30,000 training and 1,900 testing documents. We used a single-layer LSTM with 64 hidden units. We trained the character embedding with 16 dimensions for 70 characters in total, per LSTM-Jump (Yu et al., 2017). We trained all models using $R =$ 30, $K =$ 40, and $N =$ 5.

As shown in Table 4, LSTM-Shuttle still yields improvement at the character level. For both LSTM-Jump and bidirectional LSTM-Jump, the speedup effect is not obvious because of the computational overhead of skim being larger than when processing the entire text directly. On the other hand, LSTM-Shuttle yields accurate predictions even when reading fewer characters before shuttling, and it clearly yields accelerated performance. We reach an accuracy of 90.1% with 1.34× speedup, both far from LSTM-Jump. Interestingly, $(20, 40, 5)$ reads fewer words than $(30, 40, 5)$, but the former predicts better. This may be due to the character-level nature of the task and because many characters actually mislead the model. This can be seen with LSTM-Jump as well. The backward ratio on the AG dataset is nearly to that of IMDB: almost three times forward with once backward.

We also show that a larger setting of K ($K =$ 80) yields further acceleration, even at the character level, achieving the highest (1.63×) speedup.

4.3 Cloze on Children's Book Test Name Entity dataset

For the cloze task, we must supply the missing words in an article. In the Children's Book Test (CBT) (Hill et al., 2015), the question includes a complete article and a query from which a specific word is deleted. The model must determine which of the ten candidate words is most suitable. In contrast to previous tasks, which have a fixed class type, CBT provides different candidate words for each question. Thus we cannot train it as with a normal classification problem. Inspired by (Chen et al., 2016), we formulate the task as

$$\text{softmax}(CWh_o) \in \mathbb{R}^{10}, \qquad (2)$$

where $C \in \mathbb{R}^{10\times d}$ is the word embedding matrix, h_o is the latest LSTM hidden state, and W is a trainable weight variable. The output of the above equation is taken as the index of the answer word. We train LSTM-Shuttle to maximize the distribution over a one-hot answer index. Therefore for

Method	(R, K, N)	Accuracy	Speedup	Backward
LSTM	-	45.3%	1x	-
LSTM-Jump	(1, 5, 5)	46.8%*	3.05x	-
LSTM-Jump	(1, 5, 1)	45.2%	**6.28x***	-
Bi-LSTM-Jump	(1, 5, 5)	47.0%*	2.64x	-
Bi-LSTM-Jump	(1, 5, 1)	45.3%	6.19x*	-
LSTM-Shuttle	(1, 5, 5)	**47.2%***	2.98x	0.31
LSTM-Shuttle	(1, 5, 1)	46.0%	6.16x*	0.18
LSTM-Shuttle	(1, **10**, 5)	47.1%*	2.91x	0.36
LSTM-Shuttle	(1, **10**, 1)	46.6%	6.13x*	0.27

Table 5: Cloze result on CBT-NE. * means the best accuracy or highest speedup for each method given the same setting of K.

different candidate words, we concatenate them as an embedding matrix, feed this into the above equation, and generate the prediction distribution. We used the named-entity (NE) part of CBT as the cloze dataset when evaluating LSTM-Shuttle.

4.3.1 Result on CBT-NE

CBT-NE includes 120,769 questions for training and 2,500 for testing. We trained all models using Eq. (2) with a two-layer LSTM and 256 hidden units. Pre-trained word2vec embeddings were again applied directly. We trained them under $R = 1$, $K = 5$, and $N = 5$ at the sentence level, which means that LSTM-Shuttle read one sentence and shuttled several sentences five times. Vanilla LSTM, LSTM-Jump, Bi-LSTM-Jump, and LSTM-Shuttle all read the query, but only vanilla LSTM read the entire question article. Others decided how to skim or shuttle.

The result is reported in Table 5. LSTM-Shuttle's best accuracy is 47.2% with 2.98× speedup, and the highest efficiency version achieved 46.0% accuracy with 6.16× speedup. Despite the modest acceleration effect, LSTM-Shuttle yields consistently better prediction accuracy with minimal drops in efficiency. Acceleration is only modest because the average number of article sentences was only 20, and it thus did not need to read many sentences ($R = 1$) before shuttling or shuttling so many times ($N = 5$). That is also why the backward ratio here is low in CBT-NE.

LSTM-Shuttle maintains accurate prediction and high efficiency under a larger K: 46.6% accuracy with 6.13× speedup. To demonstrate the proposed method, we offer a case study in Section 4.5.

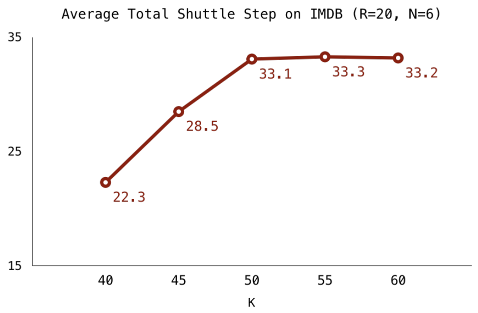

Figure 3: Average total shuttle steps

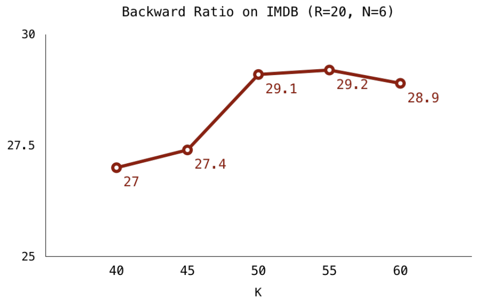

Figure 4: Backward ratio

4.4 Analysis of Shuttle Mechanism

Here we analyze how LSTM-Shuttle actually operates. We compute the total average shuttle steps and the average shuttle steps for each shuttle action under same $R = 20$ and $N = 6$ but different K on the IMDB datasets.

Fig. 3 shows the average total shuttle steps for different K, taking into consideration both forward and backward steps. For backward shuttling, we use the absolute value as its shuttle steps. For instance, shuttling -5 means it goes back 5 words, and the shuttle step is 5 indeed. We can see that a larger K, and thus a larger shuttle space, tends to shuttle larger steps, but also converges for large enough values of K. We see the same thing in the backward ratio. Fig. 4 shows the backward ratio

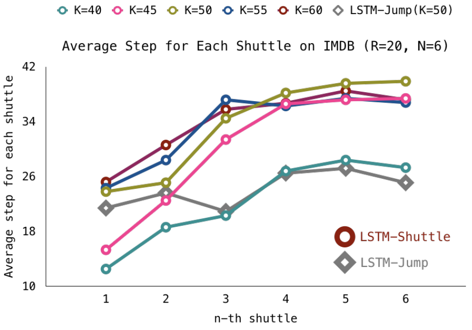

Figure 5: Average steps per shuttle

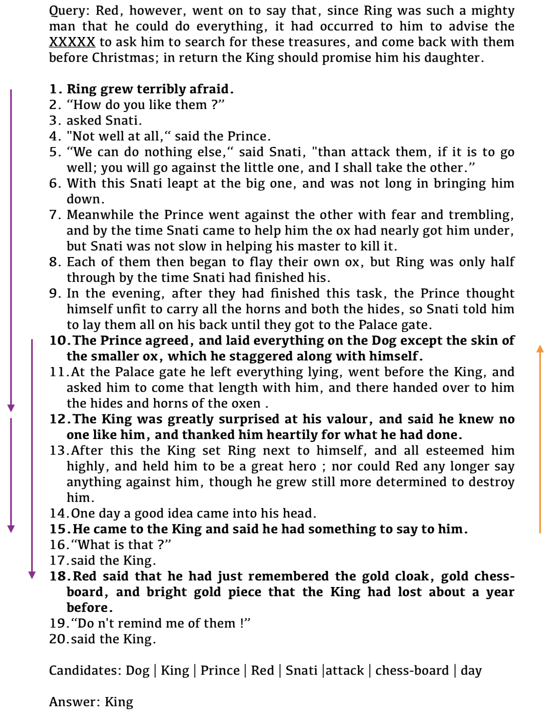

Query: Red, however, went on to say that, since Ring was such a mighty man that he could do everything, it had occurred to him to advise the XXXXX to ask him to search for these treasures, and come back with them before Christmas; in return the King should promise him his daughter.

1. **Ring grew terribly afraid.**
2. “How do you like them ?”
3. asked Snati.
4. "Not well at all,“ said the Prince.
5. “We can do nothing else,“ said Snati, "than attack them, if it is to go well; you will go against the little one, and I shall take the other.”
6. With this Snati leapt at the big one, and was not long in bringing him down.
7. Meanwhile the Prince went against the other with fear and trembling, and by the time Snati came to help him the ox had nearly got him under, but Snati was not slow in helping his master to kill it.
8. Each of them then began to flay their own ox, but Ring was only half through by the time Snati had finished his.
9. In the evening, after they had finished this task, the Prince thought himself unfit to carry all the horns and both the hides, so Snati told him to lay them all on his back until they got to the Palace gate.
10. **The Prince agreed, and laid everything on the Dog except the skin of the smaller ox, which he staggered along with himself.**
11. At the Palace gate he left everything lying, went before the King, and asked him to come that length with him, and there handed over to him the hides and horns of the oxen .
12. **The King was greatly surprised at his valour, and said he knew no one like him, and thanked him heartily for what he had done.**
13. After this the King set Ring next to himself, and all esteemed him highly, and held him to be a great hero ; nor could Red any longer say anything against him, though he grew still more determined to destroy him.
14. One day a good idea came into his head.
15. **He came to the King and said he had something to say to him.**
16. “What is that ?”
17. said the King.
18. **Red said that he had just remembered the gold cloak, gold chess-board, and bright gold piece that the King had lost about a year before.**
19. “Do n't remind me of them !”
20. said the King.

Candidates: Dog | King | Prince | Red | Snati |attack | chess-board | day

Answer: King

Figure 6: Example 1 for $R = 1$, $K = 15$, and $N = 4$, where read sentences are shown in bold

under different K. Since a larger K shuttles more both forward and backward, it has more chances to go back. Also, it converges when K is large enough.

In addition to the total average, we seek to understand how LSTM-Shuttle shuttles for each. As above, both forward and backward are taken into consideration and the absolute value is used for the backward steps. We show each shuttle record for a total of 6 shuttles under K between $[40, 50]$. As shown in Fig. 5, all settings of LSTM-Shuttle shuttle more steps after shuttling more times, but LSTM-Jump skims at an almost fixed frequency. Thus the model reads more words after more shuttles since it tends to read sequential words before each shuttle, in turn yielding better comprehension for the model. For LSTM-Shuttle, with its backward mechanism to recover lost information, it shuttles with larger and larger steps. However, for LSTM-Jump, as it cannot go back, it reads more carefully and maintains a constant skim step.

4.5 Case Study

Below, we show two examples of LSTM-Shuttle shuttling on the CBT-NE dataset. Example 1 in Fig. 6 illustrates a simple case. Based only on the query, “King should promise him his daughter”, the deleted word clearly should be “King”. LSTM-Shuttle shows more confidence providing the answer given only this query, so it shuttles with large steps to read the last part of the article, from sentence 1 to 12. Also, it goes back to confirm its prediction: from sentence 15 to 10. Example 2 in Fig. 7 is a more difficult task because the answer word “Nora” appears only once in the entire article. Thus LSTM-Shuttle must read more carefully. The shuttle steps are all smaller than 8 before discovering the answer in sentence 18, after which it goes back to see if it missed something. From these examples, we see that the shuttle mechanism is used in diverse manners for queries with different difficulties. For simple queries, LSTM-Shuttle shuttles in large steps, while for difficult queries, it shuttles more conservatively. In both cases we witness the ability to go back if necessary to make sure it understands correctly.

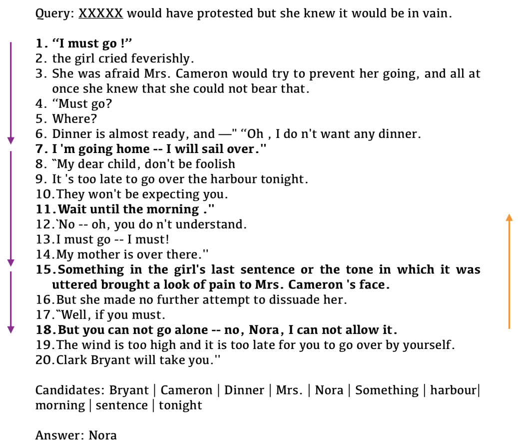

Query: XXXXX would have protested but she knew it would be in vain.

1. **“I must go !”**
2. the girl cried feverishly.
3. She was afraid Mrs. Cameron would try to prevent her going, and all at once she knew that she could not bear that.
4. “Must go?
5. Where?
6. Dinner is almost ready, and —" “Oh , I do n't want any dinner.
7. **I 'm going home -- I will sail over."**
8. ˝My dear child, don't be foolish
9. It 's too late to go over the harbour tonight.
10. They won't be expecting you.
11. **Wait until the morning ."**
12. `No -- oh, you do n't understand.
13. I must go -- I must!
14. My mother is over there.''
15. **Something in the girl's last sentence or the tone in which it was uttered brought a look of pain to Mrs. Cameron 's face.**
16. But she made no further attempt to dissuade her.
17. ˝Well, if you must.
18. **But you can not go alone -- no, Nora, I can not allow it.**
19. The wind is too high and it is too late for you to go over by yourself.
20. Clark Bryant will take you.''

Candidates: Bryant | Cameron | Dinner | Mrs. | Nora | Something | harbour| morning | sentence | tonight

Answer: Nora

Figure 7: Example 2 for $R = 1$, $K = 15$, and $N = 4$, where read sentences are shown in bold

5 Conclusion

We present LSTM-Shuttle to use human speed reading techniques for text comprehension. In addition to reading forward and skimming over text to accelerate, LSTM-Shuttle goes back to recover lost information or double-check its grasp of the text's meaning. We evaluate LSTM-Shuttle on sentiment analysis, news classification, and cloze on IMDB, Rotten Tomatoes, AG, and Children's Book Test datasets. We show that LSTM-Shuttle predicts better on all datasets with higher efficiency. We also analyze LSTM-Shuttle's behavior under different shuttle step restrictions, and provide case studies that reveal the specific shuttle operations; these show how the model comprehends the context to achieve specific goals.

References

Paszke Adam, Gross Sam, Chintala Soumith, Chanan Gregory, Yang Edward, DeVito Zachary, Lin Zeming, Desmaison Alban, Antiga Luca, and Lerer Adam. 2017. Automatic differentiation in pytorch. In *Proceedings of Neural Information Processing Systems*.

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text summarization techniques: A brief survey. In *arXiv preprint arXiv:1707.02268*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Victor Campos, Brendan Jou, Xavier Giro i Nieto, Jordi Torres, and Shih-Fu Chang. 2018. Skip rnn: Learning to skip state updates in recurrent neural networks. In *International Conference on Learning Representations*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Meeting of the Association for Computational Linguistics*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Meeting of the Association for Computational Linguistics*.

Jason P.C. Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. In *Transactions of the Association for Computational Linguistics*.

Benjamin Van Durme and Lenhart Schubert. 2008. Open knowledge extraction through compositional language processing. In *Conference on Semantics in Text Processing*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. In *arXiv preprint arXiv:1511.02301*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. In *Neural computation*.

Yoon Kim. 2014. Convolutional neural net- works for sentence classification. In *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*.

Maas Andrew L., Daly Raymond E., Pham Peter T., Huang Dan, Ng Andrew Y., and Potts Christopher. 2011. Learning word vectors for sentiment analysis. In *Meeting of the Association for Computational Linguistics*.

Quoc V. Le and Tomas Mikolov. 2014a. Distributed representations of sentences and documents. In *International Conference on Machine Learning*.

Quoc V. Le and Tomas Mikolov. 2014b. Distributed representations of sentences and documents. In *International Conference on Machine Learning*.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of Empirical Methods in Natural Language Processing*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2017. Coherent dialogue with attention-based language models. In *Association for the Advancement of Artificial Intelligence*.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2015. Recurrent models of visual attention. In *Proceedings of Neural Information Processing Systems*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Meeting of the Association for Computational Linguistics*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Neural speed reading via skimrnn. In *International Conference on Learning Representations*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Meeting of the Association for Computational Linguistics*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. In *Knowledge Discovery and Data Mining*.

Siddarth Srinivasan, Richa Arora, and Mark Riedl. 2018. A simple and effective approach to the story cloze test. In *arXiv preprint arXiv:1803.05547*.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of Neural Information Processing Systems*.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. In *Meeting of the Association for Computational Linguistics*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*.

Adams Wei Yu, Hongrae Lee, and Quoc V. Le. 2017. Learning to skim text. In *Meeting of the Association for Computational Linguistics*.

Keyi Yu, Yang Liu, Alexander G. Schwing, and Jian Peng. 2018. Fast and accurate text classification: Skimming, rereading and early stopping. In *International Conference on Learning Representations*.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis : A survey. In *arXiv preprint arXiv:1801.07883*.

Modeling Localness for Self-Attention Networks

Baosong Yang
University of Macau
nlp2ct.baosong@gmail.com

Zhaopeng Tu[*]
Tencent AI Lab
zptu@tencent.com

Derek F. Wong[*]
University of Macau
derekfw@umac.mo

Fandong Meng
Tencent AI Lab
fandongmeng@tencent.com

Lidia S. Chao
University of Macau
lidiasc@umac.mo

Tong Zhang
Tencent AI Lab
bradymzhang@tencent.com

Abstract

Self-attention networks have proven to be of profound value for its strength of capturing global dependencies. In this work, we propose to model localness for self-attention networks, which enhances the ability of capturing useful local context. We cast localness modeling as a learnable Gaussian bias, which indicates the central and scope of the local region to be paid more attention. The bias is then incorporated into the original attention distribution to form a revised distribution. To maintain the strength of capturing long distance dependencies and enhance the ability of capturing short-range dependencies, we only apply localness modeling to lower layers of self-attention networks. Quantitative and qualitative analyses on Chinese⇒English and English⇒German translation tasks demonstrate the effectiveness and universality of the proposed approach.

1 Introduction

Recently, a new simple architecture, the TRANSFORMER (Vaswani et al., 2017), that based solely on attention mechanisms has attracted increasing attention in machine translation community. Instead of using complex recurrent or convolutional neural networks, TRANSFORMER implements encoder and decoder as self-attention networks to draw global dependencies between input and output. By further parallel performing (multi-head) and stacking (multi-layer) attentive functions, TRANSFORMER has achieved state-of-the-art performance on various translation tasks (Shaw et al., 2018; Hassan et al., 2018).

One strong point of self-attention networks is the strength of capturing long-range dependencies by explicitly attending to all the signals. In this way, a representation is allowed to build a direct relation with another long-distance representation. Accordingly, it can serve as the role of RNN and CNN to capture both the short- and long-range relations among the representations.

Self-attention networks fully take into account all the signals with a weighted averaging operation. We argue that such operation disperses the distribution of attention, which results in overlooking the relation of neighboring signals. Recent works have shown that self-attention networks benefit from locality modeling. For example, Shaw et al. (2018) introduced relative position encoding to consider the relative distances between sequence elements, which produces substantial improvements on the translation task. Sperber et al. (2018) modeled the local information by restricting self-attention model to neighboring representations, which boosts performance on long-sequence acoustic modeling. Although not for self-attention, Luong et al. (2015) proposed a local attention model for translation task, which looks at only a subset of source words at a time. Inspired by these studies, we propose more flexible strategies for modeling localness for self-attention networks in this work.

Specifically, we cast the localness modeling as a learnable Gaussian bias, in which a central position (i.e. mean of the position) and a dynamic window (i.e. deviation of the distribution) are predicted with the intermediate representations in the self-attention network. Intuitively, the central position and the window respectively denote the center and the scope of the locality to be paid more attention. The learned Gaussian bias is then incorporated into the original attention distribution to form a revised distribution, which considers the expected local context.

Some researchers may doubt that self-attention networks augmented localness modeling focuses

---

[*] Zhaopeng Tu and Derek F. Wong are the co-corresponding authors of the paper. This work was conducted when Baosong Yang was interning at Tencent AI Lab.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4449–4458
Brussels, Belgium, October 31 - November 4, 2018.

leanings toward local context, which weakens its strength of capturing long-range dependencies. Our extensive analyses can dispel such doubt by showing that the potential problem is compensated by multi-layer multi-head self-attention networks. First, multi-head attention attends to local regions centered at different positions, which can constitute the complete information of an input sequence. Second, we found that self-attention models tend to capture short-range dependencies among neighboring words in lower layers, while capture long-range dependencies beyond phrase boundaries in higher layers. Accordingly, we only apply localness modeling to lower layers.

We conducted experiments on two widely-used WMT14 English⇒German and WMT17 Chinese⇒English translation tasks. The proposed approach consistently improves translation performance over the strong TRANSFORMER baseline, demonstrating its effectiveness and universality. In addition, our approach is complementary to the relative position encoding (Shaw et al., 2018), and combining them can further improve translation performance.

2 Background

Attention model has recently been a basic module of most deep learning models. The mechanism allows to dynamically select related representations as needed. In particular, it is very useful for generation models such as machine translation (Bahdanau et al., 2015; Luong et al., 2015; Yang et al., 2017) and image captioning (Xu et al., 2015).

2.1 Self-Attention Model

Recently, self-attention networks (Vaswani et al., 2017; Shaw et al., 2018; Shen et al., 2018a) have attracted increasing attention due to their flexibility in parallel computation and dependency modeling. Self-attention networks calculate attention weights between each pair of tokens in a single sequence, thus can capture long-range dependency more directly than their RNN counterpart.

Formally, given an input sequence $\mathbf{x} = \{x_1, \ldots, x_I\}$, each hidden state in the l-th layer is constructed by attending to the states in the $(l-1)$-th layer.[1] Specifically, the $(l-1)$-th layer $H^{l-1} \in \mathbb{R}^{I\times d}$ is first transformed into the queries $Q \in \mathbb{R}^{I\times d}$, the keys $K \in \mathbb{R}^{I\times d}$, and the values $V \in \mathbb{R}^{I\times d}$ with three separate weight matrices. The l-th layer is calculated as:

$$H^l = \textsc{Att}(Q,K)\,V, \tag{1}$$

where $\textsc{Att}(\cdot)$ is a dot-product attention model, defined as:

$$\textsc{Att}(Q,K) = softmax(energy) \tag{2}$$

$$energy = \frac{QK^T}{\sqrt{d}}, \tag{3}$$

where $\sqrt{d}$ is the scaling factor with d being the dimensionality of layer states.

2.2 Motivation

The self-attention network models the global dependencies without regard to their distances, by directly attending to all the positions in an input sequence (i.e. Equation 3). We argue that self-attention can be further improved by taking into account the local context. However, since the conventional self-attention models consider all of the words in a sequence, the weighted averaging inhibits the relation among the neighboring words.

From a linguistic intuition, when a word x_i is aligned to another word x_j, we also expect x_i to align mainly to the neighboring words of x_j, so as to capture the phrasal patterns that contain useful local context information. Take Figure 1 as an example, if "*Bush*" is aligned to "*held*" with high probability, we expect the self-attention model to pay more attention to the neighboring words "*a talk*". Consequently, the model is guided to capture the phrase "*held a talk*".

3 Localness Modeling

Figure 1 shows an example. We first learn a Gaussian bias, which is centered around the word "*talk*" (it is not necessary to be consistent with the original attention distribution), with a window size being 2 (in practice, it is a float number in our model). The distribution of attention is then regularized with the learned Gaussian bias to produce the final distribution, which pays more attention to the local context around the word "*talk*".

3.1 Localness Modeling as a Gaussian Bias

Specifically, a Gaussian bias G is placed to mask the logit similarity $energy$ in Equation 2, namely:

$$\textsc{Att}(Q,K) = softmax(energy + G). \tag{4}$$

The first term is the original dot product self-attention model. $G \in \mathbb{R}^{I\times I}$ is a favor alignment

---

[1] The first layer is the word embedding layer.

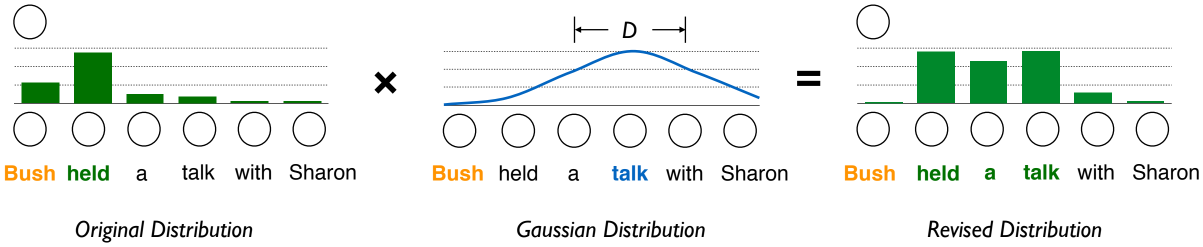

Figure 1: Illustration of the proposed approach. In this example, window size of 2 is used ($D = 2$).

position matrix (I denotes the sequence length). The element $G_{i,j} \in [0, -\infty)$ measures the tightness between the word x_j and the predicted central position P_i:

$$G_{i,j} = -\frac{(j - P_i)^2}{2{\sigma_i}^2}, \tag{5}$$

where σ_i denotes the standard deviation which is empirically set as $\sigma_i = \frac{D_i}{2}$, and D_i is a window size. Note that, due to the exponential operation in $softmax$ function, adding the logit similarity $energy$ with a bias $\in [0, -\infty)$ approximates to multiplying the attention distribution by a weight $\in [1, 0)$. The position and window size can be calculated as:

$$\begin{bmatrix} P_i \\ D_i \end{bmatrix} = I \cdot \text{sigmoid}(\begin{bmatrix} p_i \\ z_i \end{bmatrix}). \tag{6}$$

The scalar factor I is used to regulate P_i and D_i to real value numbers between 0 and the length of input sequence. The predictions are conditioned on two scalar p_i and z_i respectively.

3.2 Central Position Prediction

Since the prediction of each central position depends on its corresponding query vector,[2] we simply apply a feed-forward network to transform Q_i into a positional hidden state, which is then mapped into the scalar p_i by a linear projection $U_p \in \mathbb{R}^d$, namely:

$$p_i = {U_p}^T \tanh(W_p Q_i), \tag{7}$$

where $W_p \in \mathbb{R}^{d \times d}$ is the model parameter.

3.3 Window Size Prediction

Several alternative strategies are proposed to select the window size. Except a non-parametric approach, the other two define parametric windows. Among the parametric methods, the first strategy assigns a unified window size to all the hidden states in a layer, so as to consider the context of the sequence, while the second one calculates a distinct window size for each hidden state.

Fixed-Window A simple choice is to use a predefined window size D, which is a constant number throughout the whole training and testing process. In this study, following the common practice (Luong et al., 2015), D is set to 10.

Layer-Specific Window Furthermore, an interpretable way to select the window size is to account for the context of the sequence by summarizing the information from all the representations in a layer. In this study, we assign the mean of keys $\overline{\mathbf{K}}$ to represent the semantic context. Thus, the unified scalar z of a layer is defined as:

$$z = U_d^T \tanh(W_d \overline{\mathbf{K}}), \tag{8}$$

where $W_d \in \mathbb{R}^{d \times d}$ and $U_d \in \mathbb{R}^d$ are learnable parameters.

Query-Specific Window The last strategy provides a more flexible manner to differentiate the scope by conditioning on each query. Similar to the prediction of the central position (Equation 7), the query-specific window can be formally expressed as:

$$z_i = {U_d}^T \tanh(W_p Q_i). \tag{9}$$

Here, $U_d \in \mathbb{R}^d$ is a trainable linear projection. Note that, Equations 7 and 9 share same parameter W_p but use different U_p and U_d. The intuition behind this design is that the central position and window size interdependently locate the local scope, hence condition on the same hidden state. The distinct linear projections U_p and U_d are sufficient in distinguishing the two scalars, resulting in a smaller parameter size and faster computational speed than that of the layer-specific model.

---

[2] For the input of feed-forward network, we also tried an additive term to consider the weighted context O_i (Equation 1), namely: $\tanh(W_p Q_i + W_o O_i)$. Our experimental results showed that there is no progressive improvement.

3.4 Incorporating into TRANSFORMER

We evaluate our model on the advanced TRANSFORMER model (Vaswani et al., 2017), which builds an encoder-decoder framework merely using attention networks. Both the encoder and decoder are composed of a stack of $L = 6$ layers, each of which has two sub-layers. The first is a multi-head self-attention layer, and the second is a position-wise fully connected feed-forward layer. In this section, we describe how to apply our approach to TRANSFORMER by adapting to *multi-head* and *multi-layer* self-attention networks.

Adapting to Multi-Head Self-Attention Instead of performing a single attention function, the multi-head mechanism employs M separate attention models with distinct parameters to jointly attend to the information from different representation subspaces at different positions. Accordingly, we assign a distinct Gaussian bias to each attention head, and rewrite Equation 6 as:

$$\begin{bmatrix} P_i^m \\ D_i^m \end{bmatrix} = I \cdot \text{sigmoid}(\begin{bmatrix} p_i^m \\ z_i^m \end{bmatrix}), \qquad (10)$$

where p_i^m and z_i^m are trained with distinct parameters to predict the central position and window size for the m-th attention head.

We argue that multi-head self-attention may benefit more from localness modeling. Multi-head attention captures different features by attending to different positions, which complements the localness modeling that may potentially ignore the global information. Experimental results in Table 5 confirm our hypothesis by showing that localness modeling achieves more significant improvement when working with multi-head attention than its single-head counterpart.

Adapting to Multi-Layer Self-Attention Recent work shows that different layers capture different types of features. Anastasopoulos and Chiang (2018) indicated that higher-level layers are more representative than lower-level layers, while Peters et al. (2018) showed that higher-level layers capture context-dependent aspects of word meaning while lower-level layers model aspects of syntax. One question naturally arises: *is it necessary to model localness for all layers?*

In this work, we investigate which levels of layers benefit most from the localness modeling. In addition, we visualize the Gaussian biases across layers, to better understand the behaviors of different attentive layers.

4 Experiments

4.1 Setup

To compare with the results reported by previous work (Gehring et al., 2017; Vaswani et al., 2017; Hassan et al., 2018), we conducted experiments on both Chinese⇒English (Zh⇒En) and English⇒German (En⇒De) translation tasks. For the Zh⇒En task, the models were trained using all of the available parallel corpus from WMT17 dataset with maximum length limited to 50, consisting of about 20.62 million sentence pairs. We used newsdev2017 as the development set and newstest2017 as the test set. For the En⇒De task, we trained on the widely-used WMT14 dataset consisting of about 4.56 million sentence pairs. The models were validated on newstest2013 and examined on newstest2014. The Chinese sentences were segmented by the word segmentation toolkit *Jieba,*[3] and the English and German sentences were tokenized using the scripts provided in Moses. Then, all tokenized sentences were processed by byte-pair encoding (BPE) to alleviate the Out-of-Vocabulary problem (Sennrich et al., 2016) with 32K merge operations for both language pairs. The 4-gram NIST BLEU score (Papineni et al., 2002) is used as the evaluation metric.

We evaluated the proposed approaches on advanced TRANSFORMER model (Vaswani et al., 2017), and implemented on top of an open-source toolkit – THUMT[4] (Zhang et al., 2017). We followed Vaswani et al. (2017) to set the configurations and reproduced their reported results on the En⇒De task. We tested both the *Base* and *Big* models, which differ at the layer size (512 vs. 1024) and the number of attention heads (8 vs. 16). All the models were trained on eight NVIDIA P40 GPUs, each of which is allocated a batch of 4096 tokens. In consideration of the computation cost, we studied the variations of the *Base* model on Zh⇒En task, and evaluated the overall performance with the *Big* model on both Zh⇒En and En⇒De translation tasks.

4.2 Ablation Study

In the first series of experiments, we evaluated the impact of different components on the Zh⇒En validation set using the TRANSFORMER-BASE. First, we investigated the effect of different strategies to predict the localness window. Then, we

---

[3] `https://github.com/fxshy/jieba`

[4] `https://github.com/thumt/THUMT`

Model	Speed	Dev	△
Baseline	1.20	22.59	-
Fixed	1.14	23.07	+ 0.48
Layer-Spec.	1.07	23.13	+ 0.54
Query-Spec.	1.11	23.13	+ 0.54

Table 1: Evaluation of various window prediction strategies for localness modeling, which is only applied to encoder-side self-attention network. "Speed" denotes training speed measured in steps per second.

examined whether it is necessary to apply localness modeling to all the layers. Finally, given that TRANSFORMER consists of encoder and decoder side self-attention as well as encoder-decoder attention networks, we checked which types of attention networks benefit most from the localness modeling. To eliminate the influence of control variables, we conducted the first two ablation studies on encoder-side self-attention networks only.

Window Prediction Strategies As shown in Table 1, all the proposed window prediction strategies consistently improve the model performance over the baseline, validating the importance of localness modeling in self-attention networks. Among them, layer-specific and query-specific window outperform[5] their fixed counterpart, showing the benefit that flexible mechanism is able to capture varying local context according to layer and query information. Moreover, the flexible strategy does not reply on the handcrafted parameters (e.g. the pre-defined window size), which makes model robustly applicable to other language pairs and NLP tasks. Considering the training speed, we use the query-specific prediction mechanism as the default setting in subsequent experiments.

Layers to be Applied In this experiment, we investigated the question of which layers should be applied with the localness modeling. Recent works show that different layers tend to capture different features, thus there may have different needs for the local context. We applied localness modeling to different combinations of layers, as shown in Table 2. Clearly, modeling the localness for part of the layers consistently outperforms all layers in terms of the training speed and translation quality, which again validates our claim.

#	Layers	Speed	Dev	△
1	[1-6]	1.11	23.13	-
2	[1-1]	1.18	23.20	+ 0.07
3	[1-2]	1.17	23.23	+ 0.10
4	[1-3]	1.15	23.29	+ 0.16
5	[1-4]	1.14	23.26	+ 0.13
6	[4-6]	1.15	23.22	+ 0.09

Table 2: Evaluation of different layers in the encoder, which are implemented as self-attention with query-specific localness modeling.

Enc	Dec	Enc-Dec	Speed	Dev
✓	×	×	1.15	23.29
✓	✓	×	1.10	23.27
✓	×	✓	1.08	23.33
✓	✓	✓	1.02	23.19

Table 3: Effect of localness modeling on different types of attention networks. "Enc" and "Dec" denote the encoder and decoder side self-attention networks respectively, while "Enc-Dec" represents the encoder-decoder attention network.

Interestingly, the performance generally goes up with the increase of layers from bottom to top (Rows 2-4), while the trend does not hold when reaching the 4th-layer (Row 5). In addition, the lower three layers benefit more from the localness modeling than that of the higher three layers (Rows 4 and 6). These results reveal that lower-level layers benefit more from the local context. Accordingly, we only model the localness in the lower three layers in the following experiments.

Attention Networks to be Applied Table 3 lists the results of localness modeling on different types of attention networks. As observed, modeling localness for decoder-side self-attention and encoder-decoder attention networks only marginally improves or even harms the translation quality. We attribute the marginal improvement of the encoder-decoder attention network to the fact that it exploits the top-layer of encoder representations, which already embeds useful local context. Concerning decoder-side self-attention network, Zhang et al. (2018) pointed out

---

[5] Although the differences are not always significant, the flexible strategy consistently outperforms its fixed counterpart across language pairs. For example, the query-specific strategy improves performance over the fixed-window model by +0.07 and +0.23 BLEU points on Zh-En and En-De validation sets, respectively.

System	Architecture	Zh⇒En		En⇒De	
		# Para.	BLEU	# Para.	BLEU
	Existing NMT systems				
(Wu et al., 2016)	GNMT	n/a	n/a	n/a	26.30
(Gehring et al., 2017)	ConvS2S	n/a	n/a	n/a	26.36
(Vaswani et al., 2017)	Transformer-Base	n/a	n/a	65M	27.3
	Transformer-Big	n/a	n/a	213M	28.4
(Hassan et al., 2018)	Transformer-Big	n/a	24.2	n/a	n/a
	Our NMT systems				
this work	Transformer-Base	107.9M	24.13	88.0M	27.64
	+ Rel_Pos (Shaw et al., 2018)	108.0M	24.53	88.1M	27.94
	+ Localness	108.7M	24.77$^{\Uparrow}$	88.8M	28.11$^{\uparrow}$
	+ Localness + Rel_Pos	108.8M	24.96$^{\Uparrow}$	88.9M	28.54$^{\Uparrow}$
	Transformer-Big	303.9M	24.56	264.1M	28.58
	+ Localness	307.2M	25.03$^{\uparrow}$	267.4M	28.89
	+ Localness + Rel_Pos	307.3M	25.28$^{\Uparrow}$	267.5M	29.18$^{\Uparrow}$

Table 4: Comparing with the existing NMT systems on WMT17 Zh⇒En and WMT14 En⇒De test sets. "# Para." denotes the trainable parameter size of each model (M = million). "$\uparrow$ / $\Uparrow$": significant over the conventional self-attention counterpart ($p < 0.05/0.01$), tested by bootstrap resampling (Koehn, 2004).

that it tends to only focus on its nearby representation, which poses difficulties to modeling localness on the decoder side. In the main experiments, we only applied localness modeling to the lower three layers of the encoder, which employs a query-specific window prediction strategy.

4.3 Main Results

In this section, we evaluated the proposed approach on both WMT17 Zh⇒En and WMT14 En⇒De translation tasks, as listed in Table 4. Our baseline models, both Transformer-Base and Transformer-Big, outperform the reported results on the same data, which we believe make the evaluation convincing. As seen, modeling localness ("Localness") consistently achieves improvement across language pairs and model variations, demonstrating the efficiency and universality of the proposed approach.

We also re-implemented the relative position encoding ("Rel_Pos") that recently proposed by Shaw et al. (2018), which considers the relative distances between sequence elements. Both Shaw et al. (2018) and our work have shown that explicitly modeling locality for self-attention networks can improve the model performance. This indicates that it is necessary to enhance the locality modeling for Transformer. Besides, our approach is complementary to theirs, and combining them is able to further improve the translation performance. We attribute this to the fact that the two models modeling localness from two different aspects: First, the position embeddings are the same across different positions (if the absolute positions or relative positions are the same) and training examples, our model assigns a distinct localness bias to each position from layer to layer. Second, contrast to position encoding which learns the locality through the positional information in embeddings, our model directly revises the attention distribution to focus on a local space.

5 Analysis

We conducted extensive analyses to better understand our model in terms of its compatibility with multi-head and multi-layer attention networks, as well as building the ability of capturing phrasal patterns. All the results are reported on Zh⇒En development set with Transformer-Base, unless otherwise stated.

5.1 Compatibility with Multi-Head Attention

In this section, we investigated whether multi-head attention and localness modeling are compatible from two perspectives: (1) whether multi-head attention benefits more from the localness modeling than its single-head counterpart; and (2) how does multi-head attention work together with localness modeling?

Model	1-Head		8-Head	
	Dev	△	Dev	△
BASE	22.05	-	22.59	-
OURS	22.18	+ 0.13	23.29	+ 0.70

Table 5: Evaluation of localness modeling on top of single and multiple attention heads.

Multi-Head vs. Single-Head The single-head attention and multi-head attention differ at: the former uses a single 512-dimension attention head while the latter uses eight 64-dimension heads. The results in Table 5 confirm our claim by showing that multi-head attention indeed benefits more from our model than the single-head model (+0.70 vs. +0.13). It should be noted that our model marginally improves the performance under single-head setting. One possible reason is that our model focuses more on local context thus may ignore global information, which cannot be complemented by the single-head attention.

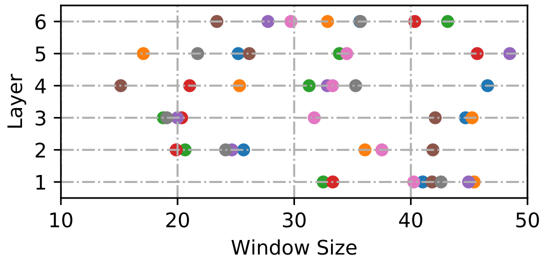

Figure 2: Instructions of the learned window size by head-specific parametric model, where colors distinguish the heads.

Can Multi-Head Separate Locality? To simplistically visualize how heads cooperate in modeling localness, we propose an additional parametric model which is assigned a learnable but unified window size for each head, namely **head-specific**. As a result, the window size D^m of the m-th head is calculated as:

$$D^m = N \cdot \text{sigmoid}(z^m), \qquad (11)$$

where the scalar z^m is a trainable parameter, $N = 50$ denotes a pre-defined constant number.

Figure 2 visualizes the distribution of the learned window size of each head, verifying that multi-head attention is able to capture diverse information by selecting suitable window sizes for different heads. For example, in the middle-level layers, heads are assigned to consider both the global and local information by regulating the different window sizes. One interesting finding is that the distributions of window size are not exactly same in different layers, which is explored in more details in the next section.

5.2 Analysis on Multi-Layer Attention

In this section, we try to answer how does each layer learn the localness. We first investigated how the window size varies across layers. Then we checked the specific behavior of the first word embedding layer, which is inconsistent with the trend of other layers.

The Higher Layer, The Larger Scope Shi et al. (2016) and Vaswani et al. (2017) have shown that different layers have the abilities to distinguish and capture diverse syntactic context (e.g. the dependents between words). Figure 3 shows the distribution of local scopes predicted by each layer. Except the first layer, the higher layers are more likely to pay attention to larger scopes, indicating that self-attention models tend to capture short-term dependencies among neighboring words in lower layers, while capture long-range dependencies beyond phrase boundaries in higher layers.

The Special First Layer Inconsistent with the intuition which the lower layers may focus on local information, in common, the first layer is assigned with large scopes of local context. The same phenomenon has also occurred for head-specific model (Figure 2). Since the first layer represents word embeddings that are deficient in context, we argue that the self-attention model at first layer has to encode the representations with global context. In addition, experimental results in Table 2 (Row 2) show that despite its large local size, modeling localness at the first layer is still valid.

5.3 Analysis on Phrasal Pattern

As aforementioned, one intuition of our approach is to capture useful phrase patterns. To evaluate the accuracy of phrase translations, we calculate the improvement of the proposed approaches over multiple N-grams, as shown in Figure 4. Although our models underperform the baseline on unigram translations, they consistently outperform the baseline on larger granularities, indicating that modeling locality can raise the ability of

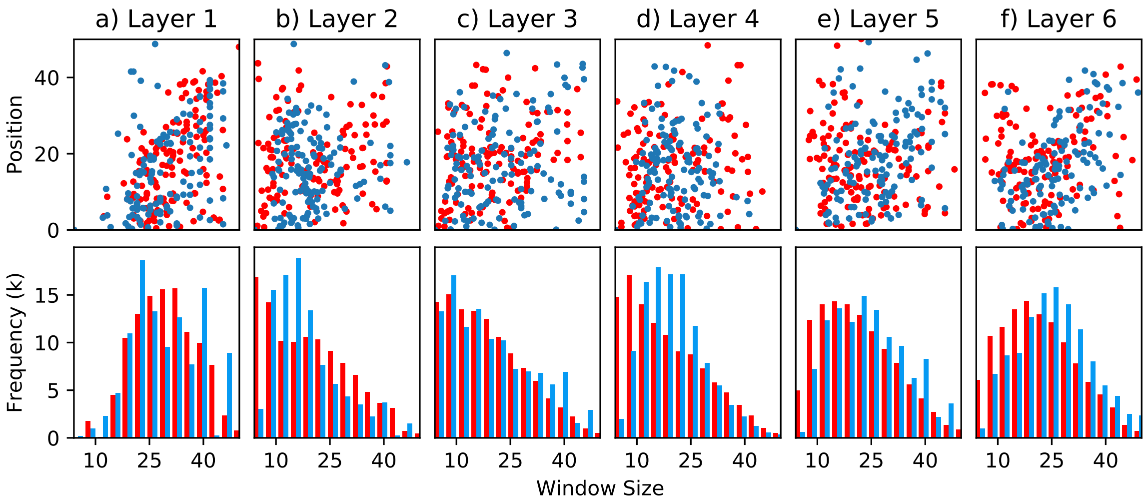

Figure 3: Distribution of the local scopes learned by each attentive layer. The upper figures illustrate the distribution of the predicted pairs of central position (Y-axis) and its correspond window size (X-axis) in each layer, the samples are randomly selected from the development set. The lower figures show the distribution of the window size in each layer. Blue color represents the layer-specific parametric approach, while the query-specific parametric method is indicated in red.

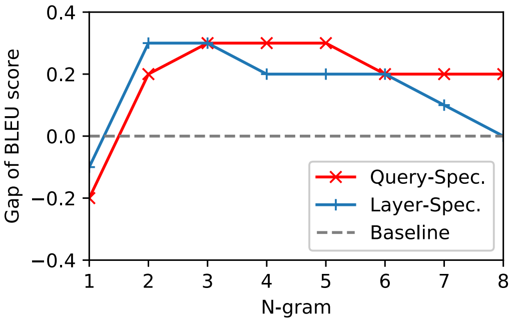

Figure 4: Performance improvement according to N-gram. Y-axis denotes the gap of BLEU score between our model and baseline.

self-attention model on capturing the phrasal information. Concerning the two variations, query-specific localness modeling surpasses its layer-specific counterpart on large phrases (i.e. 4-grams to 8-grams). We attribute this to the more modeling flexibility of query-specific strategy to differentiate the scope by conditioning on each query.

6 Related Work

A successful extension of neural language model is attention mechanism, which can directly capture long-distance dependencies by attending to previously generated words. Daniluk et al. (2017) proposed a *key-value-predict* attention to separate the key addressing, value reading, and word predicting functions explicitly. Im and Cho (2017) and Sperber et al. (2018) adopted self-attention networks for acoustic modeling and natural language inference tasks, respectively.

Vaswani et al. (2017) applied the idea of self-attention to neural machine translation. Shen et al. (2018a) and Shen et al. (2018b) proposed to improve the self-attention model with directional masks and multi-dimensional features. Although the standard self-attention model can give more bias toward localness,[6] several studies show that explicitly modeling localness for self-attention model can further improve performance. For example, Sperber et al. (2018) showed that restricting the self-attention model on the neighboring representations performs better for longer sequences in acoustic modeling and natural language inference tasks. Closely related to this work, Shaw et al. (2018) introduced relative position encoding to consider the relative distances between sequence elements. While they modeled localness from *static position embedding*, we improve locality modeling from *dynamically revising attention distribution*. Experimental results show

---

[6] As pointed out by one reviewer, in the original self-attention model, there are some considerations about given more bias toward the localness. For example, base on the definition of the positional embeddings, the adjacent words will have more similar positional embeddings compared with more further words. After summing word embeddings and corresponding positional embeddings together, the model would prefer the local words.

that the two models are complementary to each other, and combining them can further improve performance.

Several researches have shown that explicitly modeling phrases is useful for neural machine translation (Wang et al., 2017; Huang et al., 2018). Our results confirm these findings. Concerning attention models, Luong et al. (2015) proposed a modification to look at only a subset of input words at a time. This can be regarded as a "hard" variation of our fixed-window strategy. In this study, we propose more flexible strategies for placing and zooming the local scope, which yield better results than the fixed scope.

7 Conclusion

In this work, we enhanced the ability of capturing local context for self-attention networks with a learnable Gaussian bias. We proposed several strategies to learn the scope of the local context, and found that a query-specific mechanism yielded the best result due to its more modeling flexibility. Experimental results on widely-used English⇒German and Chinese⇒English translation tasks demonstrate the effectiveness and universality of the proposed approach. By visualizing the scopes of the learned Gaussian biases, we found that the higher the layer, the larger scope the bias, which is consistent with the findings in previous work (Shi et al., 2016; Peters et al., 2018).

As our approach is not limited to specific tasks, it is interesting to validate our model in other tasks, such as reading comprehension, language inference, and stance classification (Xu et al., 2018). Another promising direction is to design more powerful localness modeling techniques, such as incorporating linguistic knowledge (e.g. phrases and syntactic categories). It is also interesting to combine with other techniques (Shaw et al., 2018; Shen et al., 2018a; Dou et al., 2018; Li et al., 2018) to further improve the performance of Transformer.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61672555), the Joint Project of Macao Science and Technology Development Fund and National Natural Science Foundation of China (Grant No. 045/2017/AFJ) and the Multiyear Research Grant from the University of Macau (Grant No. MYRG2017-00087-FST). We thank the anonymous reviewers for their insightful comments.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied Multitask Learning for Neural Speech Translation. In *NAACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. In *ICLR*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting Deep Representations for Neural Machine Translation. In *EMNLP*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *ICML*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving Human Parity on Automatic Chinese to English News Translation. *arXiv:1803.05567*.

Po Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards Neural Phrase-based Machine Translation. In *ICLR*.

Jinbae Im and Sungzoon Cho. 2017. Distance-based Self-Attention Network for Natural Language Inference. *arXiv:1712.02047*.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *EMNLP*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018. Multi-Head Attention with Disagreement Regularization. In *EMNLP*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *ACL*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding. In *AAAI*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018b. Bi-Directional Block Self-Attention for Fast and Memory-Efficient Sequence Modeling. In *ICLR*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does String-based Neural MT Learn Source Syntax? In *EMNLP*.

Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. 2018. Self-Attentional Acoustic Models. In *Interspeech*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *NIPS*.

Xing Wang, Zhaopeng Tu, Deyi Xiong, and Min Zhang. 2017. Translating Phrases in Neural Machine Translation. In *EMNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation. *arXiv:1609.08144*.

Chang Xu, Cecile Paris, Surya Nepal, and Ross Sparks. 2018. Cross-Target Stance Classification with Self-Attention Networks. In *ACL*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *ICML*.

Baosong Yang, Derek F. Wong, Tong Xiao, Lidia S. Chao, and Jingbo Zhu. 2017. Towards Bidirectional Hierarchical Representations for Attention-based Neural Machine Translation. In *EMNLP*.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating Neural Transformer via an Average Attention Network. In *ACL*.

Jiacheng Zhang, Yanzhuo Ding, Shiqi Shen, Yong Cheng, Maosong Sun, Huanbo Luan, and Yang Liu. 2017. THUMT: An Open Source Toolkit for Neural Machine Translation. *arXiv:1706.06415*.

Chargrid: Towards Understanding 2D Documents

Anoop R Katti[*] **Christian Reisswig**[*] **Cordula Guder** **Sebastian Brarda**
Steffen Bickel **Johannes Höhne** **Jean Baptiste Faddoul**

SAP SE
Machine Learning R&D, Berlin Germany
{anoop.raveendra.katti | christian.reisswig | cordula.guder | sebastian.brarda | steffen.bickel | johannes.hoehne | jean.baptiste.faddoul}@sap.com

Abstract

We introduce a novel type of text representation that preserves the 2D layout of a document. This is achieved by encoding each document page as a two-dimensional grid of characters. Based on this representation, we present a generic document understanding pipeline for structured documents. This pipeline makes use of a fully convolutional encoder-decoder network that predicts a segmentation mask and bounding boxes. We demonstrate its capabilities on an information extraction task from invoices and show that it significantly outperforms approaches based on sequential text or document images.

1 Introduction

Textual information is often represented through structured documents which have an inherent 2D structure. This is even more so the case with the advent of new types of media and communications such as presentations, websites, blogs and formatted notebooks. In such documents, the layout, positioning, and sizing might be crucial to understand its semantic content and provide a strong guidance to the human perception.

NLP addresses the task of processing and understanding natural language texts through subtasks like language modeling, classification, information extraction, summarization, translation, and question answering among others. NLP methods typically operate on serialized text, which is a 1D sequence of characters. Such methods have been proven very successful for various tasks on unformatted text (e.g. books, reviews, news articles, short text snippets). However, when processing structured and formatted documents in which the relation between words is impacted not only by the sequential order, but also by the document layout, NLP can result in significant shortcomings.

Computer vision algorithms, on the other hand, are designed to exploit 2D information in the visual domain. Images are commonly processed with convolutional neural networks (LeCun et al., 1998; Krizhevsky et al., 2012; Ren et al., 2015b; Pinheiro et al., 2016) (or likes) that preserve and exploit the 2D correlation between neighboring pixels. While it is feasible to convert structured documents into images and then apply computer vision algorithms, this approach is not optimal for understanding their semantics as it is driven mostly by the visual content and not by the textual content. As a result, a machine learning model would first need to extract the text from the image followed by learning the semantics. This purely visual approach requires a more complex model and significantly larger training data compared to text-based approaches.

We propose a novel paradigm for processing and understanding structured documents. Instead of serializing a document into a 1D text, the proposed method, named *chargrid*, preserves the spatial structure of the document by representing it as a sparse 2D grid of characters. We then formulate the document understanding task as instance-level semantic segmentation on *chargrid*. More precisely, the model predicts a segmentation mask with pixel-level labels and object bounding boxes to group multiple instances of the same class. We apply the chargrid paradigm on an information extraction task from invoices and demonstrate that this method is superior to both, state-of-the-art NLP algorithms as well as computer vision algorithms.

The rest of the document is organized in three parts: we first introduce the *chargrid* paradigm; we then describe a specific application of information extraction from documents; finally, we present experimental results and conclusion.

---

[*] Equal contribution

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4459–4469
Brussels, Belgium, October 31 - November 4, 2018.

2 Related Work

NLP focuses on understanding natural language text through tasks like classification (Kim, 2014), translation (Bahdanau et al., 2014), summarization (Rush et al., 2015), and named entity recognition (Lample et al., 2016). Such methods expect unformatted text as input and, therefore, do not assume any intrinsic 2D structure.

Document Analysis, on the other hand, deals largely with problems such as recognizing printed/hand-written characters from a variety of documents (Graves and Schmidhuber, 2009), processing document images for document localization (Javed and Shafait, 2018), binarization (Tensmeyer and Martinez, 2017), and layout segmentation (Chen et al., 2015). As a result, it does not focus on understanding the character- and/or word-level semantics of the document the same way as NLP.

Within computer vision, problems such as scene text detection and recognition (Goodfellow et al., 2013), semantic segmentation (Badrinarayanan et al., 2017) as well as object detection (Gupta et al., 2014; Lin et al., 2017) can be considered as related problems to ours, but applied on a different domain, i.e., processing natural images instead of documents as input.

The closest to our work is Yang et al. (2017) that performs pixel-wise layout segmentation on a structured document, using sentence embeddings as additional input to an encoder-decoder network architecture. For each pixel inside the area of a sentence, the sentence embedding is appended to the visual feature embedding at the last layer of the decoder. The authors show that the layout segmentation accuracy can be significantly improved when using the textual features. Another related work is Palm et al. (2017) which extracts key-value information from structured documents (invoices) using a recurrent neural network (RNN). Their work addresses the problem of document understanding, however, the RNN operates on serialized 1D text.

Combining approaches from computer vision, NLP, and document analysis, our work is the first to systematically address the task of understanding 2D documents the same way as NLP while still retaining the 2D structure in structured documents.

3 Document Understanding with Chargrid

A human observer comprehends a document by understanding the semantic content of characters, words, paragraphs, and layout components. We encapsulate all such tasks under the common umbrella of *document understanding*. Therefore, we can formulate this problem as an instance segmentation task of characters on the page. In the following sections, we describe a new approach for solving that task.

3.1 Chargrid

Chargrid is a novel representation of a document that preserves its 2D layout. A chargrid can be constructed from *character boxes*, i.e., bounding boxes that each surround a single character somewhere on a given document page. This positional information can come from an optical character recognition (OCR) engine, or can be directly extracted from the layout information in the document as provided by, e.g., PDF or HTML. The coordinate space of a character box is defined by page height H and width W, and is usually measured in units of pixels.

The complete text of a document page can thus be represented as a set of tuples $\mathcal{D} = \{(c_k, \mathbf{b}_k) \mid k = 0, ..., n\}$, where c_k denotes the k-th character in the page and $\mathbf{b}_k$ the associated character box of the k-th character, which is formalized by the top-left pixel position, width and height, thus $\mathbf{b}_k = (x_k, y_k, w_k, h_k)$.

We can now construct the chargrid $g \in \mathbb{N}^{H \times W}$ of the original document page, and its *character-pixel* g_{ij} from the set $\mathcal{D}$ with

$$g_{ij} = \begin{cases} E(c_k) & \text{if} \quad (i,j) \prec \mathbf{b}_k \\ 0 \end{cases} \tag{1}$$

where $\prec$ means 'overlaps with', and where each point (i, j) corresponds to some pixel in the original document page pixel coordinate space defined by (H, W). $E(c_k)$ is some encoding of the character in the k-th character box, i.e. the value of character c_k may be mapped to a specific integer index. For instance, we may map the alphabet (or any character of interest) to non-zero indices $\{a, b, c, ...\} \rightarrow \{1, 2, 3, ...\}$. Note that we assume that character boxes cannot overlap, i.e. each character on a document page occupies a unique region on the page. In practice, it may happen that the

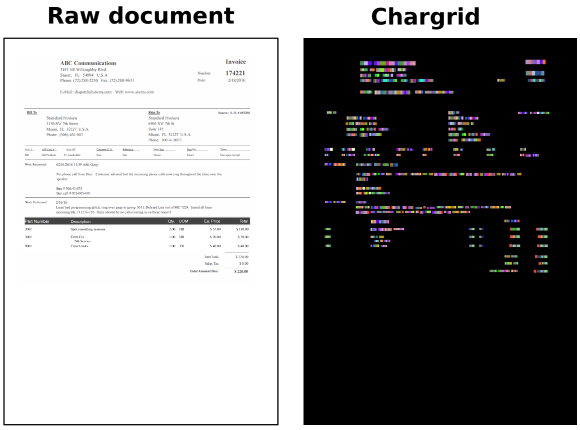

Figure 1: Example for a document page (left) and corresponding chargrid representation g (right).

corners and edges of a character box may overlap with other closeby characters. We solve such corner cases by assigning the character-pixel to the box that has the closest box center.

In other words, the chargrid representation is constructed as follows: for each character c_k at location $\mathbf{b}_k$, the area covered by that character is filled with some constant index value $E(c_k)$. All remaining character-pixels corresponding to empty regions on the original document page are initialized with 0. Figure 1 visualizes the chargrid representation of an example input document.

The advantage of the new chargrid representation is twofold: (i) we directly encode a character by a single scalar value rather than by a granular collection of grayscale pixels as is the case for images, thus making it easy for the subsequent document analysis algorithms to understand the document, and (ii), because the group of pixels that belonged to a given character are now all mapped to the same constant value, we can significantly downsample the chargrid representation without loss of any information. For instance, if the smallest character occupied a 10×10 pixel region in the original document, we can downsample the chargrid representation by a factor of 10×10. This significantly reduces the computational time of subsequent processing steps, such as training a machine learning model on this data representation.

We point out that a character can occupy a region spanning several character pixels. This is in contrast to traditional NLP where each character is represented by exactly one token. In our representation, therefore, the larger a given character, the more character-pixels represent that single character. We do not find this to be a problem, instead, it even helps since it implicitly encodes additional information, for example, the font size, that would otherwise not be available.

Before the chargrid representation is used as input to, e.g., a neural network (see Sec. 3.2), we apply 1-hot encoding to the chargrid g. Thus, the original chargrid representation $g \in \mathbb{N}^{H \times W}$ becomes a vector representation $\tilde{g} \in \mathbb{R}^{H \times W \times N_C}$, where N_C denotes the number of characters in the vocabulary including a padding/background character (in our case this is mapped to 0) and an unknown character (all characters that are not mapped by our encoding E will be mapped to this character).

We note that similar to using characters for constructing the chargrid, one can also use words to construct a *wordgrid* in the same way. In that case, rather than using 1-hot encoding, one may use a word embedding like word2vec or GloVe. While the construction of a wordgrid seems straightforward, we have not experimented with it in the present work as our dataset contains too many unusual words and spans multiple languages (see Sec. 4).

3.2 Network Architecture

We use the 1-hot encoded chargrid representation $\tilde{g}$ as input to a fully convolutional neural network to perform semantic segmentation on the chargrid and predict a class label for each character-pixel on the document. As there can be multiple and an unknown number of instances of the same class, we further perform instance segmentation. This means, in addition to predicting a segmentation mask, we may also predict bounding boxes using the techniques from object detection. This allows the model to assign characters from the same segmentation class to distinct instances.

Our model is described in Figure 2. It is comprised of two main parts: The encoder network and the decoder network. The decoder network is further made up of two branches: The segmentation branch and the bounding box regression branch. The encoder boils down to a VGG-type network (Simonyan and Zisserman, 2014) with dilated convolutions (Yu and Koltun, 2016), batch normalization (Ioffe and Szegedy, 2015), and spatial dropout (Tompson et al., 2015). Essentially, the encoder consists of five blocks where each block consists of three 3×3 convolutions (which

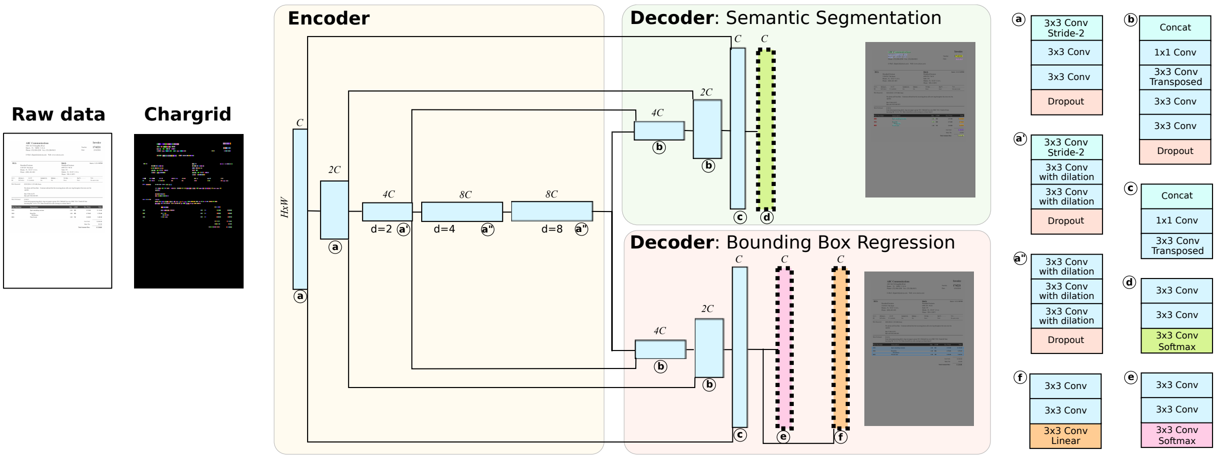

Figure 2: Network architecture for document understanding, the *chargrid-net*. Each convolutional block in the network is represented as a box. The height of a box is a proxy for feature map resolution while the width is a proxy for the number of output channels. C corresponds to the number of 'base' channels, which in turns corresponds to the number of output channels in the first encoder block. d denotes dilation rate.

themselves are made of convolution, batch normalization, ReLU activation) followed by spatial dropout at the end of a block. The first convolution in a block is a stride-2 convolution to downsample the input to that block. Whenever we downsample, we increase the number of output channels C of each convolution by a factor of two. We have found stride-2 convolutions to yield slightly better results compared to max pooling. In block four and five of the encoder, we do not apply any downsampling, and we leave the number of channels at 512 (the first block has $C = 64$ channels). We use dilated convolutions in block three, four, five with rates $d = 2, 4, 8$, respectively.

The decoder for semantic segmentation and for bounding box regression are both made of convolutional blocks which essentially reverse the downsampling of the encoder via stride-2 transposed convolutions. Each block first concatenates features from the encoder via lateral connections followed by 1×1 convolutions (Ronneberger et al., 2015; Pinheiro et al., 2016). Subsequently, we upsample via a 3×3 stride-2 transposed convolution. This is followed by two 3×3 convolutions. Note that whenever we upsample, we decrease the number of channels by a factor of two.

The two decoder branches are identical in architecture up to the last convolutional block. The decoder for semantic segmentation has an additional convolutional layer without batch normalization, but with bias and with softmax activation. The number of output channels of the last convolution corresponds to the number of classes. Together with the encoder, the decoder for the bounding box regression task forms a one-stage detector which makes use of focal loss (Lin et al., 2017). We also make use of the anchor box representation and corresponding bounding box regression targets as discussed in Ren et al. (2015a). The anchor-box representation allows us to handle bounding boxes that vary widely with respect to size and aspect ratios. Moreover, the anchor-box representation allows us to detect boxes of different classes. The number of output channels are $2N_a$ for the box mask (foreground versus background) and $4N_a$ for the four box coordinates, where N_a is the number of anchors per pixel. The weights of all layers are initialized following He et al. (2015), except for the last ones, which are initialized with a small constant value $1e-3$ for stabilization purposes.

In total, we have three equally contributing loss terms:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{seg}} + \mathcal{L}_{\text{boxmask}} + \mathcal{L}_{\text{boxcoord}}, \qquad (2)$$

where $\mathcal{L}_{\text{seg}}$ is the cross entropy loss for segmentation, e.g. (Ronneberger et al., 2015), $\mathcal{L}_{\text{boxmask}}$ is the binary cross entropy loss for box masks, and $\mathcal{L}_{\text{boxcoord}}$ is the Huber loss for box coordinate regression (Ren et al., 2015a). Both cross entropy loss terms are augmented following the focal loss idea (Lin et al., 2017). We also make use of aggressive static class weighting in both cross entropy loss terms mainly to counter the strong class imbalance between irrelevant and ”easy” pixels (the ”background” class) versus relevant and ”hard” pixels (all other classes; see Sec. 4.2 for further details).

We refer to the network depicted in Figure 2 as the *chargrid-net*.

4 Information Extraction from Invoices

As a concrete example for understanding structured 2D documents, we extract key-value information from invoices. We make no assumption on the format of the invoice, the country of origin (and consequently taxes, date formats, amount formats, currency etc.) or the language. In addition to that, real-world invoices often contain incomplete sentences, nouns, and abbreviations.

We want our model to parse an invoice and to extract 5 *header fields* (i.e., Invoice Number, Invoice Date, Invoice Amount, Vendor Name and Vendor Address) as well as the list of product items purchased, referred to as the *line-items*. Line-items include details for each item such as Line-item Description, Line-item Quantity and Line-item Amount. Together with the background class, this yields 9 classes and each character on the invoice is associated to exactly one class. We note that while header fields may only appear once on an invoice (are unique), line-items may occur in multiple instances.

To extract the values for each field, we collect all characters that are classified as belonging to the corresponding class. For line-items, we further group the characters by the predicted item bounding boxes.

4.1 Data

Our invoice dataset consists of 12k scanned sample invoices from a large variety of different vendors and languages. We assign 10k samples for training, 1k for validation and 1k for test on which we report our results (see Sec. 5). We ensure that vendors that appear in one set do not appear in any other set. This is more restrictive than necessary but gives a good estimate on how well the model generalizes to unseen invoice layouts. Figure 3 shows the distribution over vendors and languages. From the vendor distribution, it can be seen that most vendors contribute only one invoice to the dataset with at most 6 invoices coming from a single vendor. From the language distribution, it can be seen that while English is the predominant language, there are large representations from French, Spanish, Norwegian, and German.

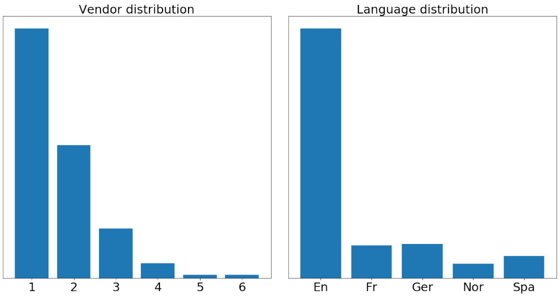

Figure 3: The left image shows a histogram of number of vendors over the number of contributing invoices in the dataset. Most vendors appear only once in the dataset. The right image shows a distribution over languages, illustrating the diversity of the invoice data.

For all invoices, we collected manual annotations with bounding boxes around the fields of interest. Considerable efforts were spent to ensure that the labels are correct. In particular, each invoice was analyzed by three annotators plus a reviewer. Over 35k invoices were investigated and finally only clean set of 12k invoices were selected. Furthermore, detailed instructions were given to the annotators for each field. Figure 4 visualizes the locations of annotated boxes on invoices for Invoice Amount and Line-item Quantity. It can be seen that while some regions are denser, the occurrences are spread widely over the whole invoice thereby illustrating the diversity in the invoice format.

4.2 Implementation Details

The invoices were processed with an open-source engine for OCR, Tesseract (v4), to extract the text from the documents.

We limit the number of distinct characters in our encoding E (in Eq. 1) to the most frequent ones, which in our present case means $N_C = 54$ different characters including the background/padding and unknown character. Thus, the 1-hot encoded chargrid $\tilde{g}$ has 54 channels. Each page of an invoice is processed independently.

Once we created the chargrid representation of a document page, we downsample to a fixed resolution using nearest neighbor interpolation to ensure that all chargrid representations have the same resolution for training (and inference). Note that this downsampling operation is still performed in the token space (i.e., before 1-hot encoding). To further ensure that no tokens in the chargrid representation are lost due to nearest neighbor interpolation, we downsample to twice the target resolution. This resolution is determined by the smallest characters in the training set. A second downsampling step, now in the 1-hot encoding to the final target resolution of 336x256 in our case, is performed using bilinear interpolation and fed into the network. Note that we could have also first performed 1-hot encoding on the document and applied bilinear interpolation directly to the target resolution. Computationally, however, this two-stage downsampling is more efficient.

We handle landscape documents by simply squeezing all input pages into our target resolution using interpolation (similar to image re-sizing). We have found that this approach is not harmful and for simplicity stick to it.

Line-items can occur in an unknown number of distinct instances. Therefore, we require instance segmentation of characters on the document. To accomplish this, the model is trained to predict bounding boxes that span across the entire row of one instance of a line-item, while the segmentation mask classifies those characters belonging to given column classes (such as, e.g., Line-item Quantity, or Line-item Description) of that line-item instance.

We implemented our model in TensorFlow 1.4. We use SGD with momentum $\beta = 0.9$ and learning rate $\alpha = 0.05$. We used weight decay of $\lambda = 10^{-4}$, and spatial dropout with probability $P = 0.1$. We perform random cropping of the chargrid for data augmentation (that is we pad by 16 character pixels in each direction after downsampling and then crop with a random offset in range $(16, 16)$).

We use aggressive class weighting in the cross-entropy loss for semantic segmentation and for the bounding box mask. We have found this to be more effective than the focal loss (which can bee seen as a form of dynamic class weighting). We implement class weighting following (Paszke et al., 2016) with a constant of $c = 1.04$. In early stages of our experiments not yet using class weights, we noted poor performance on the bounding box regression task as well as the segmentation task.

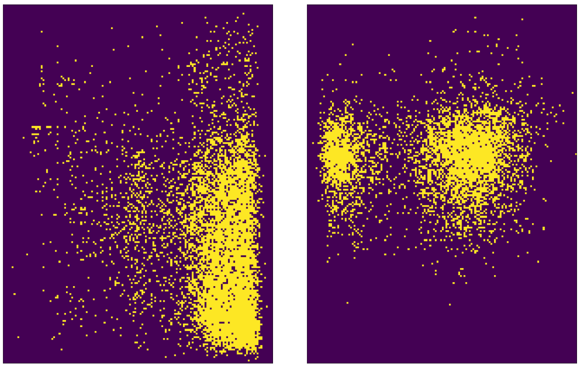

Figure 4: Spatial distribution of Invoice Amount (left) and Line-item Quantity (right) over the invoice. This depicts the variation in the invoice layouts contained in our dataset.

The distribution of line-items on invoices in our dataset reveals that around 50% of all invoices only contain less than three line-items. We found that repeating those invoices with more than three line-items during training more often than those with only few line-items significantly boosts bounding box regression accuracy. With a mini batch size of 7, each model took around $360k$ iterations and 3 days to fully converge on a single Nvidia Tesla V100.

4.3 Evaluation Measure

For evaluating our model, we would like to measure how much work would be saved by using the extraction system, compared to performing the field extraction manually. To capture this, we use a measure similar to the word error rate (Prabhavalkar et al., 2017) used in speech recognition or translation tasks.

For a given field, we count the number of insertions, deletions, and modifications of the predicted instances (pooled across the entire test set) to match the ground truth instances. Evaluations are made on the string level. We compute this measure as follows:

$$1 - \frac{\#[\text{insertions}] + \#[\text{deletions}] + \#[\text{modifications}]}{N}$$

where N is the total number of instances occurring in the ground truth of the entire test set. This measure can be negative, meaning that it would be less work to perform the extraction manually. In our present case, the error caused by the OCR engine does not affect this measure, because the same errors are present in the prediction and in the ground truth and are not considered a mismatch.

Figure 5: Sample invoices and their corresponding network predictions: the top row shows the chargrid input, the bottom row shows the predicted segmentation mask with overlaid bounding box predictions. Our model is able to handle a large diversity of layouts. The encoding of the characters on the chargrid has been scrambled to preserve privacy.

5 Experiments and Results

Figure 5 shows some sample predictions. It can be seen that the model can successfully extract key-value information on sample invoices despite significant diversity and complexity in the invoice layouts.

We show the quantitative results in Table 1. We compare the proposed model, chargrid-net (Sec. 3.2), against four other models. The first one is a *sequential* model based purely on text. It is a stack of bi-directional GRUs (Cho et al., 2014) taking a sequence of characters as input, and producing a sequence of labels as output. This model is our implementation of Palm et al. (2017) and serves as a baseline comparison against a more traditional NLP approach which is based on sequential input. We note that we have also experimented with a extension of this model that along with the characters also takes as input the position of each character on the document, however, with negligible returns. Therefore, we stick to this simpler model.

The second *image-only* model is identical to chargrid-net (Figure 2), except that we directly take the original image of the document page as network input rather than the chargrid $\tilde{g}$. This model serves as a baseline comparison to a purely image-based approach using directly the raw pixel information as input. We note that we downsample the image to the same input resolution as the chargrid representation, that is 336x256.

The third and fourth models are both a *hybrid* between the chargrid-net, and the image-only model. In both models, we replicate the encoder: one encoder for the chargrid input $\tilde{g}$, and one encoder for the image of the document page. Information from the two encoders is concatenated in the decoder: whenever a lateral connection from the encoder of the original chargrid-net is concatenated in a decoder block, we now concatenate lateral connections from the two encoders in a decoder block.

We distinguish two configurations of the hybrid model: model *chargrid-hybrid-C64* where chargrid and image encoders both have $C = 64$ base channels, and model *chargrid-hybrid-C32* where chargrid and image encoders have $C = 32$ base channels while the decoder still retains $C = 64$ base channels. The latter model is used to assess the influence on the number of encoder parameters since the hybrid model effectively has more parameters due to the two encoders.

It can be seen that compared to the purely text-based approach, the chargrid-net performs equivalently on single-instance single-word fields where the 2D structure is not as important. Examples of single-instance, single-word fields are 'Invoice Number', 'Invoice Amount', and 'Invoice Date'. The extraction of these fields could essentially also be easily tackled with named entity recognition approaches based on serialized text (Gillick et al., 2015; Lample et al., 2016).

The chargrid-net, however, significantly outper-

Model/Field	Invoice Number	Invoice Amount	Invoice Date	Vendor Name	Vendor Address	Line-item Description	Line-item Quantity	Line-item Amount
sequential	80.98%	79.13%	83.98%	28.97%	16.94%	-0.01%	-0.18%	0.22%
image-only	47.79%	68.91%	45.67%	19.68%	13.99%	49.50%	46.79%	63.49%
chargrid-net	80.48%	80.74%	83.78%	36.00%	39.13%	52.80%	65.20%	65.57%
chargrid-hybrid-C32	74.85%	77.93%	80.40%	32.00%	31.48%	46.27%	64.04%	63.25%
chargrid-hybrid-C64	82.49%	80.14%	84.28%	34.27%	36.83%	48.81%	64.59%	64.53%

Table 1: Accuracy measure (c.f. Sec. 4.3) for an 8-class information extraction problem on invoices. The proposed chargrid models perform consistently well on all extracted fields compared to sequential and image models.

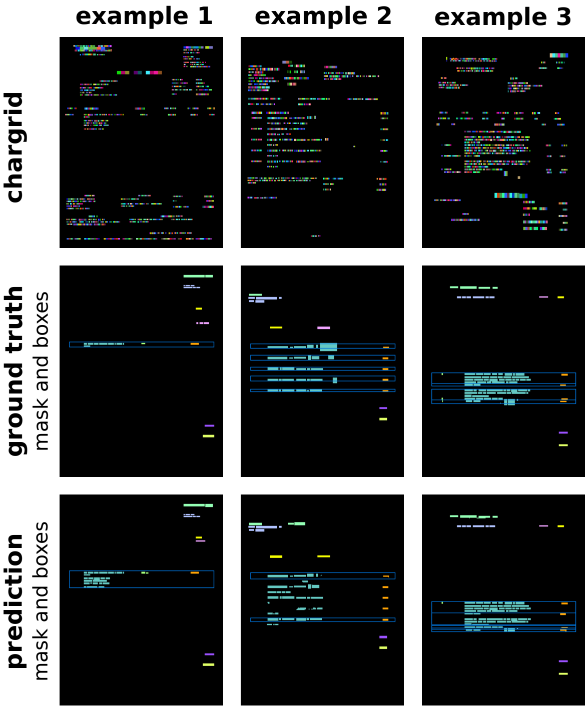

Figure 6: In example 1, chargrid-net attributes unrelated rows of text to the line-item's description. Moreover, some character-pixels are misclassified such that the predicted header field above the line-item is incorrect. Thus, we observe errors in both, the segmentation mask and the bounding box predictions. In example 2, the model predicts the segmentation mask mostly correct, but it fails to predict boxes for two of the four line-items. In example 3, the model fails to separate adjacent multi-row line-items from each other correctly. Please note that the ground truth is debatable in some cases; for example, it may be hard to decide which information belongs to a line-item's description and which does not.

forms the sequential model on multi-instance or multi-word fields where 2D relationships between text entities are important. Examples of such fields are Line-item Description, Line-item Quantity and Line-item Amount, which are all grouped as *sub-fields* to a specific line-item. Each of those fields may span a varying number of rows per line-item, see Figure 5. The sequential model fails to correctly identify those line-item fields which is manifested in a negative accuracy measure (Sec. 4.3). This implies that it is better to perform manual extraction over using automatic extraction. This is understandable since the line-item fields have a strong 2D structure that the sequential approach is not designed to capture.

In comparison with the image-only model, chargrid-net still performs much better. This is especially true for smaller fields which need to be *read* to be accurately localized. On the other hand, for larger fields like Line-item Description, which can be localized by only vision, the gap is much smaller.

The values for the hybrid models are a bit more interesting. One could expect that combining two complementary inputs such as the chargrid representation and image - one capturing the content and the other capturing, e.g. table delineations - would boost the accuracy. It turns out, however, that at least in the case of our invoice dataset, the additional image encoder does not bring additional benefits. Model chargrid-hybrid-C64, where the chargrid encoder branch has the same number of base channels $C = 64$ as the original chargrid-net, is essentially as accurate as chargrid-net. Model chargrid-hybrid-C32, where the image and chargrid encoder combined have the same number of channels (that is the chargrid encoder branch only has $C = 32$ base channels), the accuracy is significantly reduced.

We conclude that at least in our present extraction problem, most of the discriminative information comes from the chargrid encoder branch and thus from the chargrid representation.

Error examples of the chargrid-net model are depicted in Figure 6. One frequent error-type is that the model may fail to disentangle line-items, if they have a peculiar structure. While this is also challenging for human experts, other erroneous predictions are observed in samples for which the ground truth annotations are debatable.

6 Discussion

We introduce a new way of modeling documents by using a character grid as document representation. The chargrid allows models to capture 2D relationships between characters, words, and larger units of text. The idea of the chargrid paradigm is inspired by the human perception which is heavily guided by 2D shapes and structures for understanding this type of documents. Therefore, the chargrid allows to encode the positioning, size and alignment for textual components in a meaningful manner. While the chargrid paradigm could be applied to various kinds of NLP tasks, we demonstrate its potential on an information extraction task from invoices. We train a deep neural network with an encoder-decoder architecture and we show that the network computes accurate segmentation masks and bounding boxes, which pinpoint the relevant information on the invoice.

We evaluate the accuracy of the model and compare it to state-of-the-art NLP and computer vision approaches. While those baseline models achieve accurate predictions for individual fields, only the chargrid performs well on all information extraction tasks. Some fields such as Invoice Number are relatively easy to detect for a model that operates on serialized text, as discriminative keywords are commonly preceding the words to be extracted. However, more complex extraction tasks (e.g. Vendor Address or Line-item Quantity) cannot be performed accurately as they require to exploit both, textual components and 2D layout structures. The traditional image-only computer vision model yields accurate predictions only for large visual columns (such as Line-item Amount) and it fails to locate extractions that require to understand the text.

However, compared to traditional sequential neural NLP models, the benefits in accuracy come at a larger computational cost compared. Even though the proposed model is fully convolutional and parallelizes very well on a single (or even multiple) GPU(s), the added complexity introduced by using a 2D data representation can significantly increase the total data dimensionality. In our current use case, a chargrid-net training requires up to three days until full convergence, whereas our sequential model converges after only a few hours.

Comparing chargrid to semantic segmentation on natural images, one should note that character-pixels, unlike pixels of a gray-scale image, are categorical. This requires the character-pixels to be encoded as 1-hot. This yields a highly sparse data representation. While such representation is very common for NLP problems, it is new for segmentation networks that have previously only been applied to image segmentation.

7 Conclusion

Chargrid is a generic representation for 2D text. Using this as a base, one could solve any task on a 2D text such as document classification, named-entity recognition, information extraction, parts-of-speech tagging etc. Furthermore,chargrid is highly beneficial for scenarios where text and natural images are blended. One could imagine performing all the above NLP tasks also on such inputs. This work demonstrates the advantages of chargrid for an information extraction task but we believe that it is only the first step towards incorporating the 2D document structure into document understanding tasks. Some follow-up direction could be solving other NLP tasks on structured documents using chargrid or experimenting with other computer vision algorithms on chargrid. Furthermore, it may be interesting to use word embeddings rather than 1-hot encoded characters, i.e. a wordgrid, as 2D text input representation.

Acknowledgments

In acknowledgement of the partnership between Nvidia and SAP this publication has been prepared using Nvidia DGX-1. We thank Zbigniew Jerzak and Markus Noga for their support and fruitful discussions.

References

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 39(12):2481–2495.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

K. Chen, M. Seuret, M. Liwicki, J. Hennebert, and R. Ingold. 2015. Page segmentation of historical document images with convolutional autoencoders. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. pages 1011–1015. https://doi.org/10.1109/ICDAR.2015.7333914.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gülcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR* abs/1406.1078. http://arxiv.org/abs/1406.1078.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103* .

Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay D. Shet. 2013. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *CoRR* abs/1312.6082. http://arxiv.org/abs/1312.6082.

Alex Graves and Jürgen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, Curran Associates, Inc., pages 545–552. http://papers.nips.cc/paper/3449-offline-handwriting-recognition-with-multidimensional-recurrent-neural-networks.pdf.

Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. 2014. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*. Springer, pages 345–360.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. pages 1026–1034.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* .

K. Javed and F. Shafait. 2018. Real-time document localization in natural images by recursive application of a cnn. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. volume 01, pages 105–110. https://doi.org/10.1109/ICDAR.2017.26.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR* abs/1408.5882. http://arxiv.org/abs/1408.5882.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. pages 1097–1105.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* .

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002* .

Rasmus Berg Palm, Ole Winther, and Florian Laws. 2017. Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. *arXiv preprint arXiv:1708.07403* .

Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. 2016. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147* .

Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. 2016. Learning to refine object segments. In *European Conference on Computer Vision*. Springer, pages 75–91.

Rohit Prabhavalkar, Tara N Sainath, Yonghui Wu, Patrick Nguyen, Zhifeng Chen, Chung-Cheng Chiu, and Anjuli Kannan. 2017. Minimum word error rate training for attention-based sequence-to-sequence models. *arXiv preprint arXiv:1712.01818* .

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015a. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. pages 91–99.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015b. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR* abs/1506.01497. http://arxiv.org/abs/1506.01497.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, pages 234–241.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR* abs/1509.00685. http://arxiv.org/abs/1509.00685.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Chris Tensmeyer and Tony Martinez. 2017. Document image binarization with fully convolutional neural networks. *CoRR* abs/1708.03276. http://arxiv.org/abs/1708.03276.

Tesseract. v4 https://github.com/tesseract-ocr/tesseract.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 648–656.

Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C Lee Giles. 2017. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. *arXiv preprint arXiv:1706.02337* .

Fisher Yu and Vladlen Koltun. 2016. Multi-scale context aggregation by dilated convolutions. In *ICLR*.

Simple Recurrent Units for Highly Parallelizable Recurrence

Tao Lei[1] **Yu Zhang**[2] **Sida I. Wang**[1,3] **Hui Dai**[1] **Yoav Artzi**[1,4]
[1]ASAPP Inc. [2]Google Brain [3]Princeton University [4]Cornell University
[1]`{tao, hd}@asapp.com` [2]`ngyuzh@google.com`
[3]`sidaw@cs.princeton.edu` [4]`yoav@cs.cornell.edu`

Abstract

Common recurrent neural architectures scale poorly due to the intrinsic difficulty in parallelizing their state computations. In this work, we propose the Simple Recurrent Unit (SRU), a light recurrent unit that balances model capacity and scalability. SRU is designed to provide expressive recurrence, enable highly parallelized implementation, and comes with careful initialization to facilitate training of deep models. We demonstrate the effectiveness of SRU on multiple NLP tasks. SRU achieves 5–9x speed-up over cuDNN-optimized LSTM on classification and question answering datasets, and delivers stronger results than LSTM and convolutional models. We also obtain an average of 0.7 BLEU improvement over the Transformer model (Vaswani et al., 2017) on translation by incorporating SRU into the architecture.[1]

1 Introduction

Recurrent neural networks (RNN) are at the core of state-of-the-art approaches for a large number of natural language tasks, including machine translation (Cho et al., 2014; Bahdanau et al., 2015; Jean et al., 2015; Luong et al., 2015), language modeling (Zaremba et al., 2014; Gal and Ghahramani, 2016; Zoph and Le, 2016), opinion mining (Irsoy and Cardie, 2014), and situated language understanding (Mei et al., 2016; Misra et al., 2017; Suhr et al., 2018; Suhr and Artzi, 2018). Key to many of these advancements are architectures of increased capacity and computation. For instance, the top-performing models for semantic role labeling and translation use eight recurrent layers, requiring days to train (He et al., 2017; Wu et al., 2016b). The scalability of these models has become an important problem that impedes NLP research.

The difficulty of scaling recurrent networks arises from the time dependence of state computation. In common architectures, such as Long Short-term Memory (LSTM; Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU; Cho et al., 2014), the computation of each step is suspended until the complete execution of the previous step. This sequential dependency makes recurrent networks significantly slower than other operations, and limits their applicability. For example, recent translation models consist of non-recurrent components only, such as attention and convolution, to scale model training (Gehring et al., 2017; Vaswani et al., 2017).

In this work, we introduce the Simple Recurrent Unit (SRU), a unit with light recurrence that offers both high parallelization and sequence modeling capacity. The design of SRU is inspired by previous efforts, such as Quasi-RNN (QRNN; Bradbury et al., 2017) and Kernel NN (KNN; Lei et al., 2017), but enjoys additional benefits:

- SRU exhibits the same level of parallelism as convolution and feed-forward nets. This is achieved by balancing sequential dependence and independence: while the state computation of SRU is time-dependent, each state dimension is independent. This simplification enables CUDA-level optimizations that parallelize the computation across hidden dimensions and time steps, effectively using the full capacity of modern GPUs. Figure 1 compares our architecture's runtimes to common architectures.

- SRU replaces the use of convolutions (i.e., n-gram filters), as in QRNN and KNN, with more recurrent connections. This retains modeling capacity, while using less computation (and hyper-parameters).

---

[1] Our code is available at `https://github.com/taolei87/sru`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4470–4481
Brussels, Belgium, October 31 - November 4, 2018.

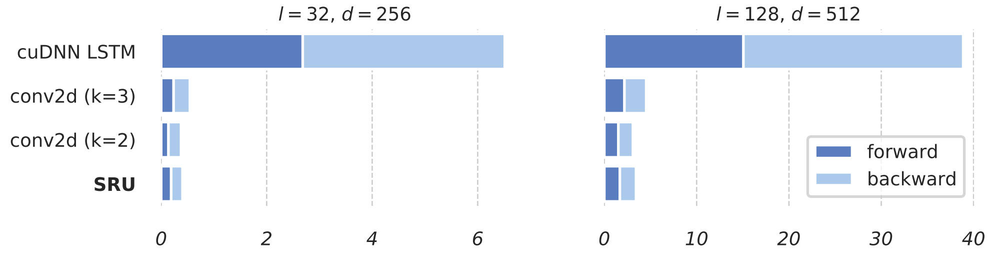

Figure 1: Average processing time in milliseconds of a batch of 32 samples using cuDNN LSTM, word-level convolution `conv2d` (with filter width $k = 2$ and $k = 3$), and the proposed SRU. We vary the number of tokens per sequence (l) and feature dimension (d).

- SRU improves the training of deep recurrent models by employing highway connections (Srivastava et al., 2015) and a parameter initialization scheme tailored for gradient propagation in deep architectures.

We evaluate SRU on a broad set of problems, including text classification, question answering, translation and character-level language modeling. Our experiments demonstrate that light recurrence is sufficient for various natural language tasks, offering a good trade-off between scalability and representational power. On classification and question answering datasets, SRU outperforms common recurrent and non-recurrent architectures, while achieving 5–9x speed-up compared to cuDNN LSTM. Stacking additional layers further improves performance, while incurring relatively small costs owing to the cheap computation of a single layer. We also obtain an average improvement of 0.7 BLEU score on the English to German translation task by incorporating SRU into Transformer (Vaswani et al., 2017).

2 Related Work

Improving on common architectures for sequence processing has recently received significant attention (Greff et al., 2017; Balduzzi and Ghifary, 2016; Miao et al., 2016; Zoph and Le, 2016; Lee et al., 2017). One area of research involves incorporating word-level convolutions (i.e. n-gram filters) into recurrent computation (Lei et al., 2015; Bradbury et al., 2017; Lei et al., 2017). For example, Quasi-RNN (Bradbury et al., 2017) proposes to alternate convolutions and a minimalist recurrent pooling function and achieves significant speed-up over LSTM. While Bradbury et al. (2017) focus on the speed advantages of the network, Lei et al. (2017) study the theoretical characteristics of such computation and possible extensions. Their results suggest that simplified recurrence retains strong modeling capacity through layer stacking. This finding motivates the design of SRU for both high parallelization and representational power. SRU also relates to IRNN (Le et al., 2015), which uses an identity diagonal matrix to initialize hidden-to-hidden connections. SRU uses point-wise multiplication for hidden connections, which is equivalent to using a diagonal weight matrix. This can be seen as a constrained version of diagonal initialization.

Various strategies have been proposed to scale network training (Goyal et al., 2017) and to speed up recurrent networks (Diamos et al., 2016; Shazeer et al., 2017; Kuchaiev and Ginsburg, 2017). For instance, Diamos et al. (2016) utilize hardware infrastructures by stashing RNN parameters on cache (or fast memory). Shazeer et al. (2017) and Kuchaiev and Ginsburg (2017) improve the computation via conditional computing and matrix factorization respectively. Our implementation for SRU is inspired by the cuDNN-optimized LSTM (Appleyard et al., 2016), but enables more parallelism – while cuDNN LSTM requires six optimization steps, SRU achieves more significant speed-up via two optimizations.

The design of recurrent networks, such as SRU and related architectures, raises questions about representational power and interpretability (Chen et al., 2018; Peng et al., 2018). Balduzzi and Ghifary (2016) applies type-preserving transformations to discuss the capacity of various simplified RNN architectures. Recent work (Anselmi et al., 2015; Daniely et al., 2016; Zhang et al., 2016; Lei et al., 2017) relates the capacity of neural networks to deep kernels. We empirically demonstrate SRU can achieve compelling results by stacking multiple layers.

3 Simple Recurrent Unit

We present and explain the design of Simple Recurrent Unit (SRU) in this section. A single layer of SRU involves the following computation:

$$\begin{aligned}
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f\mathbf{x}_t + \mathbf{v}_f \odot \mathbf{c}_{t-1} + \mathbf{b}_f\right) && (1)\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1-\mathbf{f}_t) \odot (\mathbf{W}\mathbf{x}_t) && (2)\\
\mathbf{r}_t &= \sigma\left(\mathbf{W}_r\mathbf{x}_t + \mathbf{v}_r \odot \mathbf{c}_{t-1} + \mathbf{b}_r\right) && (3)\\
\mathbf{h}_t &= \mathbf{r}_t \odot \mathbf{c}_t + (1-\mathbf{r}_t) \odot \mathbf{x}_t && (4)
\end{aligned}$$

where $\mathbf{W}$, $\mathbf{W}_f$ and $\mathbf{W}_r$ are parameter matrices and $\mathbf{v}_f$, $\mathbf{v}_r$, $\mathbf{b}_f$ and $\mathbf{b}_v$ are parameter vectors to be learnt during training. The complete architecture decomposes to two sub-components: a *light recurrence* (Equation 1 and 2) and a *highway network* (Equation 3 and 4).

The light recurrence component successively reads the input vectors $\mathbf{x}_t$ and computes the sequence of states $\mathbf{c}_t$ capturing sequential information. The computation resembles other recurrent networks such as LSTM, GRU and RAN (Lee et al., 2017). Specifically, a forget gate $\mathbf{f}_t$ controls the information flow (Equation 1) and the state vector $\mathbf{c}_t$ is determined by adaptively averaging the previous state $\mathbf{c}_{t-1}$ and the current observation $\mathbf{W}\mathbf{x}_t$ according to $\mathbf{f}_t$ (Equation 2).

One key design decision that differs from previous gated recurrent architectures is the way $\mathbf{c}_{t-1}$ is used in the sigmoid gate. Typically, $\mathbf{c}_{t-1}$ is multiplied with a parameter matrix to compute $\mathbf{f}_t$, e.g., $\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{V}_f\mathbf{c}_{t-1} + \mathbf{b}_f)$. However, the inclusion of $\mathbf{V}_f\mathbf{c}_{t-1}$ makes it difficult to parallelize the state computation: each dimension of $\mathbf{c}_t$ and $\mathbf{f}_t$ depends on all entries of $\mathbf{c}_{t-1}$, and the computation has to wait until $\mathbf{c}_{t-1}$ is fully computed. To facilitate parallelization, our light recurrence component uses a point-wise multiplication $\mathbf{v}_f \odot \mathbf{c}_{t-1}$ instead. With this simplification, each dimension of the state vectors becomes independent and hence parallelizable.

The highway network component (Srivastava et al., 2015) facilitates gradient-based training of deep networks. It uses the reset gate $\mathbf{r}_t$ (Equation 3) to adaptively combine the input $\mathbf{x}_t$ and the state $\mathbf{c}_t$ produced from the light recurrence (Equation 4), where $(1-\mathbf{r}_t) \odot \mathbf{x}_t$ is a skip connection that allows the gradient to directly propagate to the previous layer. Such connections have been shown to improve scalability (Wu et al., 2016a; Kim et al., 2016; He et al., 2016; Zilly et al., 2017).

The combination of the two components makes the overall architecture simple yet expressive, and easy to scale due to enhanced parallelization and gradient propagation.

3.1 Parallelized Implementation

Despite the parallelization friendly design of SRU, a naive implementation which computes equations (1)–(4) for each step t sequentially would not achieve SRU's full potential. We employ two optimizations to enhance parallelism. The optimizations are performed in the context of GPU / CUDA programming, but the general idea can be applied to other parallel programming models.

We re-organize the computation of equations (1)–(4) into two major steps. First, given the input sequence $\{\mathbf{x}_1 \cdots \mathbf{x}_L\}$, we batch the matrix multiplications across all time steps. This significantly improves the computation intensity (e.g. GPU utilization). The batched multiplication is:

$$\mathbf{U}^\top = \begin{pmatrix} \mathbf{W} \\ \mathbf{W}_f \\ \mathbf{W}_r \end{pmatrix} [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L] \ ,$$

where L is the sequence length, $\mathbf{U} \in \mathbb{R}^{L \times 3d}$ is the computed matrix and d is the hidden state size. When the input is a mini-batch of B sequences, $\mathbf{U}$ would be a tensor of size $(L, B, 3d)$.

The second step computes the remaining point-wise operations. Specifically, we compile all point-wise operations into a single fused CUDA kernel and parallelize the computation across each dimension of the hidden state. Algorithm 1 shows the pseudo code of the forward function. The complexity of this step is $O(L \cdot B \cdot d)$ per layer, where L is the sequence length and B is the batch size. In contrast, the complexity of LSTM is $O(L \cdot B \cdot d^2)$ because of the hidden-to-hidden multiplications (e.g. $\mathbf{V}\mathbf{h}_{t-1}$), and each dimension can not be independently parallelized. The fused kernel also reduces overhead. Without it, operations such as sigmoid activation would each invoke a separate function call, adding kernel launching latency and more data moving costs.

The implementation of a bidirectional SRU is similar: the matrix multiplications of both directions are batched, and the fused kernel handles and parallelizes both directions at the same time.

3.2 Initialization

Proper parameter initialization can reduce gradient propagation difficulties and hence have a positive

Algorithm 1 Mini-batch version of the forward pass defined in Equations (1)–(4).

```
Indices: Sequence length $L$, mini-batch size $B$, hidden state dimension $d$.
Input: Input sequences batch $\mathbf{x}[l,i,j]$; grouped matrix multiplication $\mathbf{U}[l,i,j']$;
  initial state $\mathbf{c}_0[i,j]$; parameters $\mathbf{v}_f[j]$, $\mathbf{v}_r[j]$, $\mathbf{b}_f[j]$ and $\mathbf{b}_r[j]$.
Output: Output $\mathbf{h}[\cdot,\cdot,\cdot]$ and internal $\mathbf{c}[\cdot,\cdot,\cdot]$ states.
  Initialize $\mathbf{h}[\cdot,\cdot,\cdot]$ and $\mathbf{c}[\cdot,\cdot,\cdot]$ as two $L \times B \times d$ tensors.
  for $i = 1,\cdots,B; j = 1,\cdots,d$ do          // Parallelize each example $i$ and dimension $j$
    $c = \mathbf{c}_0[i,j]$
    for $l = 1,\cdots,L$ do
      $f = \sigma\,(\,\mathbf{U}[l,i,j+d] + \mathbf{v}_f[j] \times c + \mathbf{b}_f[j]\,)$
      $c = f \times c + (1-f) \times \mathbf{U}[l,i,j]$
      $r = \sigma\,(\,\mathbf{U}[l,i,j+d\times 2] + \mathbf{v}_r[j] \times c + \mathbf{b}_r[j]\,)$
      $h = r \times c + (1-r) \times \mathbf{x}[l,i,j]$
      $\mathbf{c}[l,i,j] = c$
      $\mathbf{h}[l,i,j] = h$
  return $\mathbf{h}[\cdot,\cdot,\cdot]$ and $\mathbf{c}[\cdot,\cdot,\cdot]$
```

impact on the final performance. We now describe an initialization strategy tailored for SRU.

We start by adopting common initializations derived for feed-forward networks (Glorot and Bengio, 2010; He et al., 2015). The weights of parameter matrices are drawn with zero mean and $1/d$ variance, for instance, via the uniform distribution $[-\sqrt{3/d}, +\sqrt{3/d}]$. This ensures the output variance remains approximately the same as the input variance after the matrix multiplication.

However, the light recurrence and highway computation would still reduce the variance of hidden representations by a factor of $1/3$ to $1/2$:

$$\frac{1}{3} \le \frac{\mathrm{Var}[\mathbf{h}_t]}{\mathrm{Var}[\mathbf{x}_t]} \le \frac{1}{2} \ ,$$

and the factor converges to $1/2$ in deeper layers (see Appendix A). This implies the output $\mathbf{h}_t$ and the gradient would vanish in deep models. To offset the problem, we introduce a *scaling correction* constant α in the highway connection

$$\mathbf{h}_t \ = \ \mathbf{r}_t \odot \mathbf{c}_t \ + \ (1-\mathbf{r}_t) \odot \mathbf{x}_t \cdot \alpha \ \ ,$$

where α is set to $\sqrt{3}$ such that $\mathrm{Var}[\mathbf{h}_t] \approx \mathrm{Var}[\mathbf{x}_t]$ at initialization. When the highway network is initialized with a non-zero bias $\mathbf{b}_r = b$, the scaling constant α can be accordingly set as:

$$\alpha \ = \ \sqrt{1 + \exp(b) \times 2} \ \ .$$

Figure 2 compares the training progress with and without the scaling correction. See Appendix A for the derivation and more discussion.

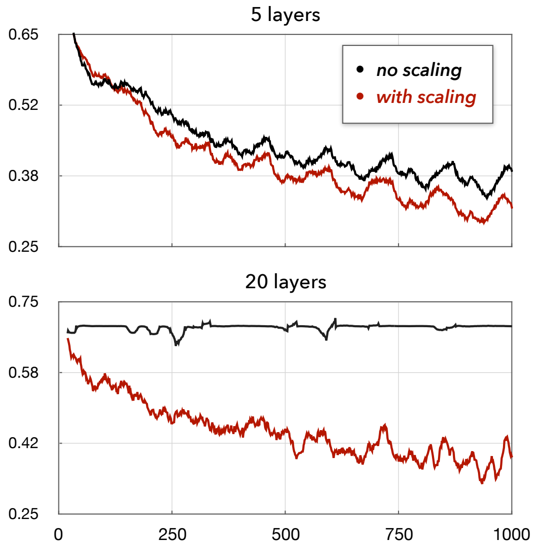

Figure 2: Training curves of SRU on classification. The x-axis is the number of training steps and the y-axis is the training loss. Scaling correction improves the training progress, especially for deeper models with many stacked layers.

4 Experiments

We evaluate SRU on several natural language processing tasks and perform additional analyses of the model. The set of tasks includes text classification, question answering, machine translation, and character-level language modeling. Training time on these benchmarks ranges from minutes (classification) to days (translation), providing a variety of computation challenges.

The main question we study is the performance-speed trade-off SRU provides in comparison to

Model	Size	CR	SUBJ	MR	TREC	MPQA	SST	Time
Best reported results:								
Wang and Manning (2013)		82.1	93.6	79.1	-	86.3	-	-
Kalchbrenner et al. (2014)		-	-	-	93.0	-	86.8	-
Kim (2014)		85.0	93.4	81.5	93.6	89.6	88.1	-
Zhang and Wallace (2017)		84.7	93.7	81.7	91.6	89.6	85.5	-
Zhao et al. (2015)		86.3	**95.5**	**83.1**	92.4	**93.3**	-	-
Our setup (default Adam, fixed word embeddings):								
CNN	360k	83.1±1.6	92.7±0.9	78.9±1.3	93.2±0.8	89.2±0.8	85.1±0.6	417
LSTM	352k	82.7±1.9	92.6±0.8	79.8±1.3	93.4±0.9	89.4±0.7	88.1±0.8	2409
QRNN (k=1)	165k	83.5±1.9	93.4±0.6	82.0±1.0	92.5±0.5	90.2±0.7	88.2±0.4	345
QRNN (k=1) + highway	204k	84.0±1.9	93.4±0.8	82.1±1.2	93.2±0.6	89.6±1.2	88.9±0.2	371
SRU (2 layers)	204k	84.9±1.6	93.5±0.6	82.3±1.2	94.0±0.5	90.1±0.7	89.2±0.3	**320**
SRU (4 layers)	303k	85.9±1.5	93.8±0.6	82.9±1.0	**94.8**±0.5	90.1±0.6	**89.6**±0.5	510
SRU (8 layers)	502k	**86.4**±1.7	93.7±0.6	**83.1**±1.0	94.7±0.5	90.2±0.8	88.9±0.6	879

Table 1: Test accuracies on classification benchmarks (Section 4.1). The first block presents best reported results of various methods. The second block compares SRU and other baselines given the same setup. For the SST dataset, we report average results of 5 runs. For other datasets, we perform 3 independent trials of 10-fold cross validation (3×10 runs). The last column compares the wall clock time (in seconds) to finish 100 epochs on the SST dataset.

other architectures. We stack multiple layers of SRU to directly substitute other recurrent, convolutional or feed-forward modules. We minimize hyper-parameter tuning and architecture engineering for a fair comparison. Such efforts have a non-trivial impact on the results, which are beyond the scope of our experiments. Unless noted otherwise, the hyperparameters are set identical to prior work.

4.1 Text Classification

Dataset We use six sentence classification benchmarks: movie review sentiment (MR; Pang and Lee, 2005), sentence subjectivity (SUBJ; Pang and Lee, 2004), customer reviews polarity (CR; Hu and Liu, 2004), question type (TREC; Li and Roth, 2002), opinion polarity (MPQA; Wiebe et al., 2005), and the Stanford sentiment treebank (SST; Socher et al., 2013).[2]

Following Kim (2014), we use word2vec embeddings trained on 100 billion Google News tokens. For simplicity, all word vectors are normalized to unit vectors and are fixed during training.

Setup We stack multiple SRU layers and use the last output state to predict the class label for a given sentence. We train for 100 epochs and use the validation (i.e., development) set to select the best training epoch. We perform 10-fold cross validation for datasets that do not have a standard train-evaluation split. The result on SST is averaged over five independent trials. We use Adam (Kingma and Ba, 2014) with the default learning rate 0.001, a weight decay 0 and a hidden dimension of 128.

We compare SRU with a wide range of methods on these datasets, including various convolutional models (Kalchbrenner et al., 2014; Kim, 2014; Zhang and Wallace, 2017) and a hierarchical sentence model (Zhao et al., 2015) reported as the state of the art on these datasets (Conneau et al., 2017). Their setups are not exactly the same as ours, and may involve more tuning on word embeddings and other regularizations. We use the setup of Kim (2014) but do not fine-tune word embeddings and the learning method for simplicity. In addition, we directly compare against three baselines trained using our code base: a re-implementation of the CNN model of Kim (2014), a two-layer LSTM model and Quasi-RNN (Bradbury et al., 2017). We use the official implementation of Quasi-RNN and also implement a version with highway connection for a fair comparison. These baselines are trained using the same hyper-parameter configuration as SRU.

Results Table 1 compares the test results on the six benchmarks. We select the best number re-

---

[2] We use the binary version of SST dataset.

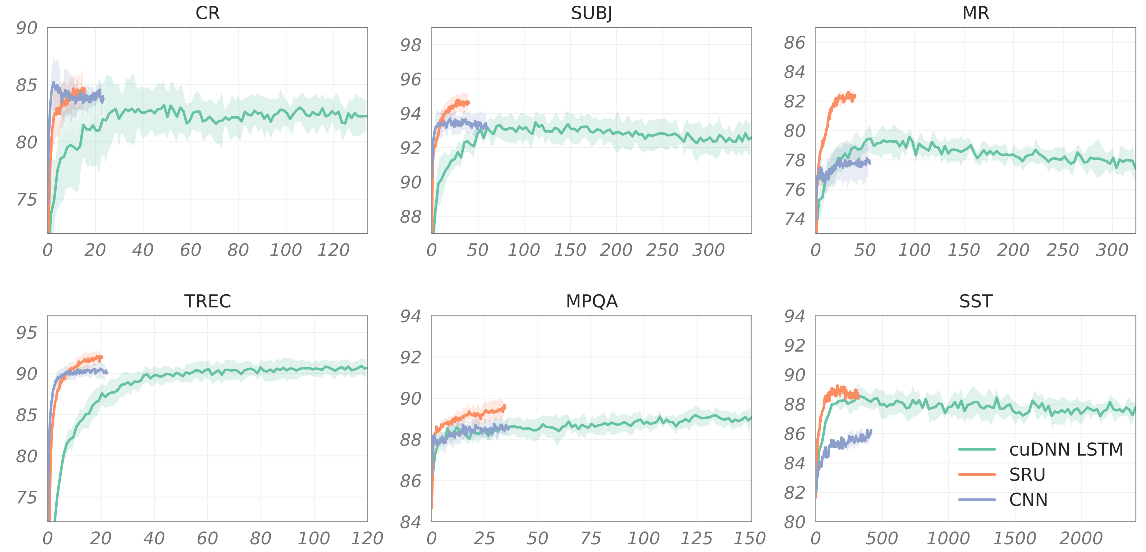

Figure 3: Mean validation accuracies (y-axis) and standard deviations of the CNN, 2-layer LSTM and 2-layer SRU models. We plot the curves of the first 100 epochs. X-axis is the training time used (in seconds). Timings are performed on NVIDIA GeForce GTX 1070 GPU, Intel Core i7-7700K Processor and cuDNN 7003.

ported in previous methods when multiple model variants were explored in their experiments. Despite our simple setup, SRU outperforms most previous methods and achieves comparable results compared to the state-of-the-art but more sophisticated model of Zhao et al. (2015). Figure 3 shows validation performance relative to training time for SRU, cuDNN LSTM and the CNN model. Our SRU implementation runs 5–9 times faster than cuDNN LSTM, and 6–40% faster than the CNN model of Kim (2014). On the movie review (MR) dataset for instance, SRU completes 100 training epochs within 40 seconds, while LSTM takes over 320 seconds.

4.2 Question Answering

Dataset We use the Stanford Question Answering Dataset (SQuAD; Rajpurkar et al., 2016). SQuAD is a large machine comprehension dataset that includes over 100K question-answer pairs extracted from Wikipedia articles. We use the standard train and development sets.

Setup We use the Document Reader model of Chen et al. (2017) as our base architecture for this task. The model is a combination of word-level bidirectional RNNs and attentions, providing a good testbed to compare our bidirectional SRU implementation with other RNN components.[3]

We use the open source implementation of Document Reader in our experiments.[4] We train models for up to 100 epochs, with a batch size of 32 and a hidden dimension of 128. Following the author suggestions, we use the Adamax optimizer (Kingma and Ba, 2014) and variational dropout (Gal and Ghahramani, 2016) during training. We compare with two alternative recurrent components: the bidirectional LSTM adopted in the original implementation of Chen et al. (2017) and Quasi-RNN with highway connections for improved performance.

Results Table 2 summarizes the results on SQuAD. SRU achieves 71.4% exact match and 80.2% F1 score, outperforming the bidirectional LSTM model by 1.9% (EM) and 1.4% (F1) respectively. SRU also exhibits over 5x speed-up over LSTM and 53–63% reduction in total training time. In comparison with QRNN, SRU obtains 0.8% improvement on exact match and 0.6% on F1 score, and runs 60% faster. This speed improvement highlights the impact of the fused ker-

---

[3] The current state-of-the-art models (Seo et al., 2016; Wang et al., 2017) make use of additional components such as character-level embeddings, which are not directly comparable to the setup of Chen et al. (2017). However, these models can potentially benefit from SRU since RNNs are incorporated in the model architecture.

[4] `https://github.com/hitvoice/DrQA`

Model	# layers	Size	Dev EM	Dev F1	Time per epoch RNN	Time per epoch Total
LSTM (Chen et al., 2017)	3	4.1m	69.5	78.8	316s	431s
QRNN (k=1) + highway	4	2.4m	70.1 ± 0.1	79.4 ± 0.1	113s	214s
	6	3.2m	70.6 ± 0.1	79.6 ± 0.2	161s	262s
SRU	3	2.0m	**70.2** ± 0.3	**79.3** ± 0.1	58s	159s
SRU	4	2.4m	**70.7** ± 0.1	**79.7** ± 0.1	72s	173s
SRU	6	3.2m	**71.4** ± 0.1	**80.2** ± 0.1	100s	201s

Table 2: Exact match (EM) and F1 scores of various models on SQuAD (Section 4.2). We also report the total processing time per epoch and the time spent in RNN computations. SRU outperforms other models, and is more than five times faster than cuDNN LSTM.

nel (Algorithm 1). While the QRNN baseline involves a similar amount of computation, assembling all element-wise operations of both directions in SRU achieves better GPU utilization.

4.3 Machine Translation

Dataset We train translation models on the WMT English→German dataset, a standard benchmark for translation systems (Peitz et al., 2014; Li et al., 2014; Jean et al., 2015). The dataset consists of 4.5 million sentence pairs. We obtain the pre-tokenized dataset from the OpenNMT project (Klein et al., 2017). The sentences were tokenized using the word-piece model (Wu et al., 2016b), which generates a shared vocabulary of about 32,000 tokens. Newstest-2014 and newstest-2017 are provided and used as the validation and test sets.[5]

Setup We use the state-of-the-art Transformer model of Vaswani et al. (2017) as our base architecture. In the base model, a single Transformer consists of a multi-head attention layer and a bottleneck feed-forward layer. We substitute the feed-forward network using our SRU implementation:

$$\begin{aligned} \text{base:} \quad & \mathbf{W} \cdot \text{ReLU\_layer}(\mathbf{x}) + \mathbf{b} \\ \text{ours:} \quad & \mathbf{W} \cdot \text{SRU\_layer}(\mathbf{x}) + \mathbf{b} \ . \end{aligned}$$

The intuition is that SRU can better capture sequential information as a recurrent network, and potentially achieve better performance while requiring fewer layers.

We keep the model configuration the same as Vaswani et al. (2017): the model dimension is $d_{\text{model}} = 512$, the feed-forward and SRU layer has inner dimensionality $d_{\text{ff}} = d_{\text{sru}} = 2048$, and positional encoding (Gehring et al., 2017) is applied on the input word embeddings. The base model without SRU has 6 layers, while we set the number of layers to 4 and 5 when SRU is added. Following the original setup, we use a dropout probability 0.1 for all components, except the SRU in the 5-layer model, for which we use a dropout of 0.2 as we observe stronger over-fitting in training.

We use a single NVIDIA Tesla V100 GPU for each model. The published results were obtained using 8 GPUs in parallel, which provide a large effective batch size during training. To approximate the setup, we update the model parameters every 5×5120 tokens and use 16,000 warm-up steps following OpenNMT suggestions. We train each model for 40 epochs (250,000 steps), and perform 3 independent trials for each model configuration. A single run takes about 3.5 days with a Tesla V100 GPU.

Results Table 3 shows the translation results. When SRU is incorporated into the architecture, both the 4-layer and 5-layer model outperform the Transformer base model. For instance, our 5-layer model obtains an average improvement of 0.7 test BLEU score and an improvement of 0.5 BLEU score by comparing the best results of each model achieved across three runs. SRU also exhibits more stable performance, with smaller variance over 3 runs. Figure 4 further compares the validation accuracy of different models. These results confirm that SRU is better at sequence modeling compared to the original feed-forward network (FFN), requiring fewer layers to achieve sim-

---

[5] `https://github.com/OpenNMT/OpenNMT-tf/tree/master/scripts/wmt`

Model	# layers	Size	BLEU score Valid	BLEU score Test	Speed (toks/sec)	Hours per epoch
Transformer (base)	6	76m	26.6±0.2 (26.9)	27.6±0.2 (27.9)	20k	2.0
Transformer (+SRU)	4	79m	26.7±0.1 (26.8)	27.8±0.1 (28.3)	**22k**	**1.8**
Transformer (+SRU)	5	90m	**27.1**±0.0 (**27.2**)	**28.3**±0.1 (**28.4**)	19k	2.1

Table 3: English→German translation results (Section 4.3). We perform 3 independent runs for each configuration. We select the best epoch based on the valid BLEU score for each run, and report the average results and the standard deviation over 3 runs. In addition, we experiment with averaging model checkpoints and use the averaged version for evaluation, following (Vaswani et al., 2017). We show the best BLEU results achieved in brackets.

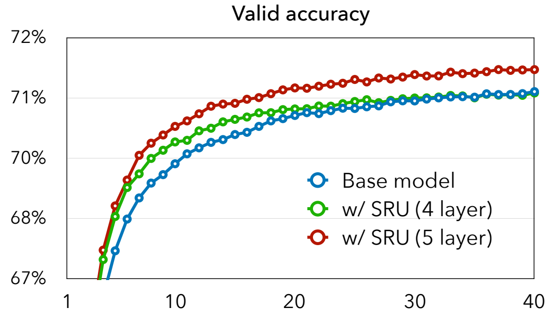

Figure 4: Mean validation accuracy (y-axis) of different translation models after each training epoch (x-axis).

ilar accuracy. Finally, adding SRU does not affect the parallelization or speed of Transformer – the 4-layer model exhibits 10% speed improvement, while the 5-layer model is only 5% slower compared to the base model. We present more results and discussion in Appendix B.3.

4.4 Character-level Language Modeling

Dataset We use Enwik8, a large dataset for character-level language modeling. Following standard practice, we use the first 90M characters for training and the remaining 10M split evenly for validation and test.

Setup Similar to previous work, we use a batch size of 128 and an unroll size of 100 for truncated backpropagation during training. We also experiment with an unroll size of 256 and a batch size of 64 such that each training instance has longer context. We use a non-zero highway bias $\mathbf{b}_r = -3$ that is shown useful for training language model (Zilly et al., 2017). Previous methods employ different optimizers and learning rate schedulers for training. For simplicity and consistency, we use the Adam optimizer and the same learning rate scheduling (i.e., Noam scheduling) as the translation experiments. We train a maximum of 100 epochs (about 700,000 steps).

We compare various recurrent models and use a parameter budget similar to previous methods. In addition, we experiment with the factorization trick (Kuchaiev and Ginsburg, 2017) to reduce the total number of parameters without decreasing the performance. See details in Appendix B.

Results Table 4 presents the results of SRU and other recurrent models. The 8-layer SRU model achieves validation and test bits per character (BPC) of 1.21, outperforming previous best reported results of LSTM, QRNN and recurrent highway networks (RHN). Increasing the layer of SRU to 12 and using a longer context of 256 characters in training further improves the BPC to 1.19

4.5 Ablation Analysis

We perform ablation analyses on SRU by successively disabling different components:

(1) Remove the point-wise multiplication term $\mathbf{v} \odot \mathbf{c}_{t-1}$ in the forget and reset gates. The resulting variant involves less recurrence and has less representational capacity.

(2) Disable the scaling correction by setting the constant $\alpha = 1$.

(3) Remove the skip connections.

We train model variants on the classification and question answering datasets. Table 5 and Figure 5 confirm the impact of our design decisions – removing these components result in worse classification accuracies and exact match scores.

Model	Size	# layers	Unroll size	Valid	Test	Time
MI-LSTM (Wu et al., 2016c)	17m	1	100	-	1.44	-
HM-LSTM (Chung et al., 2016)	35m	3	100	-	1.32	-
LSTM (Melis et al., 2017)	46m	4	50	1.28	1.30	-
RHN (Zilly et al., 2017)	46m	10	50	-	1.27	-
FS-LSTM (Mujika et al., 2017)	47m	4	100	-	1.25	-
QRNN (Merity et al., 2018)	26m	4	200	-	1.33	-
LSTM (Merity et al., 2018)	47m	3	200	-	1.23	-
SRU	37m	6	100	1.29	1.30	28min
SRU	37m	10	100	1.26	1.27	29min
SRU (with projection)	37m	6	100	1.25	1.26	29min
SRU (with projection)	47m	8	100	1.21	1.21	39min
SRU (with projection)	49m	12	256	1.19	1.19	41min

Table 4: Validation and test BPCs of different recurrent models on Enwik8 dataset. The last column presents the training time per epoch. For SRU with projection, we set the projection dimension to 512.

Model	4 layers	6 layers
SRU (full)	70.7	71.4
− remove $\mathbf{v} \odot \mathbf{c}_{t-1}$	70.6	71.1
− remove α-scaling	70.3	71.0
− remove highway	69.4	69.1

Table 5: Ablation analysis on SQuAD. Components are successively removed and the EM scores are averaged over 4 runs.

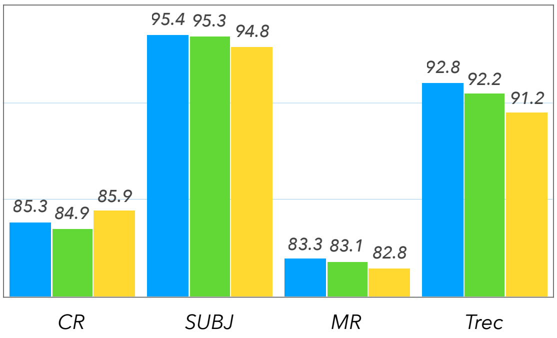

Figure 5: Ablation analysis on the classification datasets. Average validation results are presented. We compare the full SRU implementation (left blue), the variant without $\mathbf{v} \odot \mathbf{c}_{t-1}$ multiplication (middle green) and the variant without highway connection (right yellow).

5 Discussion

This work presents Simple Recurrent Unit (SRU), a scalable recurrent architecture that operates as fast as feed-forward and convolutional units. We confirm the effectiveness of SRU on multiple natural language tasks ranging from classification to translation. We open source our implementation to facilitate future NLP and deep learning research.

Trading capacity with layers SRU achieves high parallelization by simplifying the hidden-to-hidden dependency. This simplification is likely to reduce the representational power of a single layer and hence should be balanced to avoid performance loss. However, unlike previous work that suggests additional computation (e.g., n-gram filters) within the layer (Balduzzi and Ghifary, 2016; Bradbury et al., 2017), we argue that increasing the depth of the model suffices to retain modeling capacity. Our empirical results on various tasks confirm this hypothesis.

Acknowledgement

We thank Alexander Rush and Yoon Kim for help with machine translation experiments, and Danqi Chen for help with SQuAD experiments. We thank Adam Yala, Howard Chen, Jeremy Wohlwend, Lili Yu, Kyle Swanson and Kevin Yang for providing useful feedback on the paper and the SRU implementation. A special thanks to Hugh Perkins for his support on the experimental environment setup and Runqi Yang for answering questions about his code.

References

Fabio Anselmi, Lorenzo Rosasco, Cheston Tan, and Tomaso A. Poggio. 2015. Deep convolutional networks are hierarchical kernel machines. *CoRR*, abs/1508.01084.

Jeremy Appleyard, Tomás Kociský, and Phil Blunsom. 2016. Optimizing performance of recurrent neural networks on gpus. *CoRR*, abs/1604.01946.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

David Balduzzi and Muhammad Ghifary. 2016. Strongly-typed recurrent neural networks. In *International Conference on Machine Learning*.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural networks. In *Proceedings of the International Conference on Learning Representations*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yining Chen, Sorcha Gilroy, Kevin Knight, and Jonathan May. 2018. Recurrent neural networks as weighted language recognizers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Kyunghyun Cho, Bart van Merrienboer, ÃĞaglar GülÃğehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *CoRR*, abs/1609.01704.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Amit Daniely, Roy Frostig, and Yoram Singer. 2016. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*.

Greg Diamos, Shubho Sengupta, Bryan Catanzaro, Mike Chrzanowski, Adam Coates, Erich Elsen, Jesse Engel, Awni Hannun, and Sanjeev Satheesh. 2016. Persistent rnns: Stashing recurrent weights on-chip. In *International Conference on Machine Learning*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the international conference on artificial intelligence and statistics*.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch SGD: Training imagenet in 1 hour. *CoRR*, abs/1706.02677.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutnx00EDk, Bas R. Steunebrink, and Jx00FCrgen Schmidhuber. 2017. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Empirical Methods in Natural Language Processing*.

Yoon Kim, Yacine Jernite, David A Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*.

Oleksii Kuchaiev and Boris Ginsburg. 2017. Factorization tricks for lstm networks. *CoRR*, abs/1703.10722.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941.

Kenton Lee, Omer Levy, and Luke S. Zettlemoyer. 2017. Recurrent additive networks. *CoRR*, abs/1705.07393.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2017. Deriving neural architectures from sequence and graph kernels. *International Conference on Machine Learning*.

Liangyou Li, Xiaofeng Wu, Santiago Cortes Vaillo, Jun Xie, Andy Way, and Qun Liu. 2014. The DCU-ICTCAS MT system at WMT 2014 on german-english translation task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and R. Matthew Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *CoRR*, abs/1707.05589.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *CoRR*, abs/1803.08240.

Yajie Miao, Jinyu Li, Yongqiang Wang, Shi-Xiong Zhang, and Yifan Gong. 2016. Simplifying long short-term memory acoustic models for fast training and decoding. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Asier Mujika, Florian Meier, and Angelika Steger. 2017. Fast-slow recurrent neural networks. In *Advances in Neural Information Processing Systems*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the annual meeting on Association for Computational Linguistics*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the annual meeting on association for computational linguistics*.

Stephan Peitz, Joern Wuebker, Markus Freitag, and Hermann Ney. 2014. The RWTH aachen german-english machine translation system for wmt 2014. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. Rational recurrences. In *Empirical Methods in Natural Language Processing*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks:

The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

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

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in neural information processing systems*.

Alane Suhr and Yoav Artzi. 2018. Situated mapping of sequential instructions to actions with single-step reward observation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Sida Wang and Christopher Manning. 2013. Fast dropout training. In *International Conference on Machine Learning*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*.

Huijia Wu, Jiajun Zhang, and Chengqing Zong. 2016a. An empirical exploration of skip connections for sequential tagging. In *Proceedings of the International Conference on Computational Linguisticss*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ÅĄukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016b. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan R Salakhutdinov. 2016c. On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Yingjie Zhang and Byron C. Wallace. 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the International Joint Conference on Natural Language Processing*.

Yuchen Zhang, Jason D. Lee, and Michael I. Jordan. 2016. ℓ_1-regularized neural networks are improperly learnable in polynomial time. In *International Conference on Machine Learning*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2017. Recurrent highway networks. In *International Conference on Machine Learning*.

Barret Zoph and Quoc V. Le. 2016. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578.

NPRF: A Neural Pseudo Relevance Feedback Framework for Ad-hoc Information Retrieval

Canjia Li[1], **Yingfei Sun**[1], **Ben He**[1,3✉], **Le Wang**[1,4], **Kai Hui**[2],
Andrew Yates[5], **Le Sun**[3], **Jungang Xu**[1]
[1] University of Chinese Academy of Sciences, Beijing, China [2] SAP SE, Berlin, Germany
[3] Institute of Software, Chinese Academy of Sciences, Beijing, China
[4] Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
[5] Max Planck Institute for Informatics, Saarbrücken, Germany
{licanjia17, wangle315}@mails.ucas.ac.cn, {yfsun, benhe✉, xujg}@ucas.ac.cn
kai.hui@sap.com, ayates@mpi-inf.mpg.de, sunle@iscas.ac.cn

Abstract

Pseudo relevance feedback (PRF) is commonly used to boost the performance of traditional information retrieval (IR) models by using top-ranked documents to identify and weight new query terms, thereby reducing the effect of query-document vocabulary mismatches. While neural retrieval models have recently demonstrated strong results for ad-hoc retrieval, combining them with PRF is not straightforward due to incompatibilities between existing PRF approaches and neural architectures. To bridge this gap, we propose an end-to-end neural PRF framework that can be used with existing neural IR models by embedding different neural models as building blocks. Extensive experiments on two standard test collections confirm the effectiveness of the proposed NPRF framework in improving the performance of two state-of-the-art neural IR models.

1 Introduction

Recent progress in neural information retrieval models (NIRMs) has highlighted promising performance on the ad-hoc search task. State-of-the-art NIRMs, such as DRMM (Guo et al., 2016), HiNT (Fan et al., 2018), (Conv)-KNRM (Xiong et al., 2017; Dai et al., 2018), and (Co)-PACRR (Hui et al., 2017, 2018), have successfully implemented insights from traditional IR models using neural building blocks. Meanwhile, existing IR research has already demonstrated the effectiveness of incorporating relevance signals from top-ranked documents through pseudo relevance feedback (PRF) models (Buckley and Robertson, 2008; Diaz et al., 2016). PRF models expand the query with terms selected from top-ranked documents, thereby boosting ranking performance by reducing the problem of vocabulary mismatch between the original query and documents (Rocchio, 1971). Existing neural IR models do not have a mechanism for treating expansion terms differently from the original query terms, however, making it non-trivial to combine them with existing PRF approaches. In addition, neural IR models differ in their architectures, making the development of a widely-applicable PRF approach a challenging task.

To bridge this gap, we propose a generic neural pseudo relevance feedback framework, coined NPRF, that enables the use of PRF with existing neural IR models. Given a query and a target document, the top-ranked documents from the initial ranking are consumed by NPRF, which expands the query by interpreting it from different perspectives. Given a target document to evaluate, NPRF produces a final relevance score by considering the target document's relevance to these top-ranked documents and to the original query.

The proposed NPRF framework can directly incorporate different established neural IR models, which serve as the concrete scorers in evaluating the relevance of a document relative to the top-ranked documents and to the query, without changing their architectures. We instantiate the NPRF framework using two state-of-the-art neural IR models, and we evaluate their performance on two widely-used TREC benchmark datasets for ad-hoc retrieval. Our results confirm that the NPRF framework can substantially improve the performance of both models. Moreover, both neural models perform similarly inside the NPRF framework despite the fact that without NPRF one model performed substantially worse than the other model. The contributions of this work are threefold: 1) the novel NPRF framework; 2) two instantiations of the NPRF framework using two state-of-the-art neural IR models; and 3) the experiments that confirm the effectiveness of the NPRF framework.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4482–4491
Brussels, Belgium, October 31 - November 4, 2018.

The rest of this paper is organized as follows. Section 2 presents the proposed NPRF framework in details. Following that, Section 3 describes the setup of the evaluation, and reports the results. Finally, Section 4 recaps existing literature, before drawing conclusions in Section 5.

2 Method

In this section, we introduce the proposed neural framework for pseudo relevance feedback (NPRF). Recall that existing unsupervised PRF models (Rocchio, 1971; Lavrenko and Croft, 2001; Ye et al., 2009) issue a query to obtain an initial ranking, identify promising terms from the top-m documents returned, and expand the original query with these terms. Rather than selecting the expanded terms within the top-m documents, NPRF uses these documents directly as expansion queries by considering the interactions between them and a target document. Thus, each document's ultimate relevance score depends on both its interactions with the original query and its interactions with these feedback documents.

2.1 Overview

Given a query q, NPRF estimates the relevance of a target document d relative to q as described in the following steps. The architecture is summarized in Figure 1. Akin to the established neural IR models like DRMM (Guo et al., 2016), the description is based on a query-document pair, and a ranking can be produced by sorting the documents according to their scores.

- **Create initial ranking.** Given a document corpus, a ranking method $rel_q(q, d)$ is applied to individual documents to obtain the top-m documents, denoted as D_q for q.

- **Extract document interactions.** To evaluate the relevance of d, each d_q in D_q is used to expand q, where d is compared against each d_q, using a ranking method $rel_d(d_q, d)$.

- **Combine document interactions.** The relevance scores $rel_d(d_q, d)$ for individual $d_q \in D_q$ are further weighted by $rel_q(q, d_q)$, which serves as an estimator for the confidence of the contribution of d_q relative to q. The weighted combination of these relevance scores is used to produce a relevance score for d, denoted as $rel_D(q, D_q, d)$.

While the same ranking model can be used for both $rel_q(.,.)$ and $rel_d(.,.)$, we denote them separately in the architecture. In our experiments, the widely-used unsupervised ranking method BM25 (Robertson et al., 1995) serves as $rel_q(.,.)$; meanwhile two state-of-the-art neural IR relevance matching models, namely, DRMM (Guo et al., 2016) and K-NRM (Xiong et al., 2017), serve as the ranking method $rel_d(.,.)$. However, it is worth noting that in principle rel_q and rel_d can be replaced with any ranking method, and the above choices mainly aim to demonstrate the effectiveness of the NPRF framework.

2.2 Model Architecture

The NPRF framework begins with an initial ranking for the input query q determined by $rel_q(.,.)$, which forms D_q, the set of the top-m documents D_q. The ultimate query-document relevance score $rel_D(q, D_q, d)$ is computed as follows.

Extracting document interactions. Given the target document d and each feedback document $d_q \in D_q$, $rel_d(.,.)$ is used to evaluate the relevance between d and d_q, resulting in m real-valued relevance scores, where each score corresponds to the estimated relevance of d according to one feedback document d_q.

As mentioned, two NIRMs are separately used to compute $rel_d(d_q, d)$ in our experiments. Both models take as input the cosine similarities between each pair of terms in d_q and d, which are computed using pre-trained word embeddings as explained in Section 3.1. Given that both models consider only unigram matches and do not consider term dependencies, we first summarize d_q by retaining only the top-k terms according to their *tf-idf* scores, which speeds up training by reducing the document size and removing noisy terms. In our pilot experiments, the use of top-k *tf-idf* document summarization did not influence performance. For different $d_q \in D_q$, the same model is used as $rel_d(.,.)$ for different pairs of (d_q, d) by sharing model weights.

Combining document interactions. When determining the relevance of a target document d, there exist two sources of relevance signals to consider: the target document's relevance relative to the feedback documents D_q and its relevance relative to the query q itself. In this step, we combine $rel_d(d_q, d)$ for each $d_q \in D_q$ into an overall feedback document relevance score

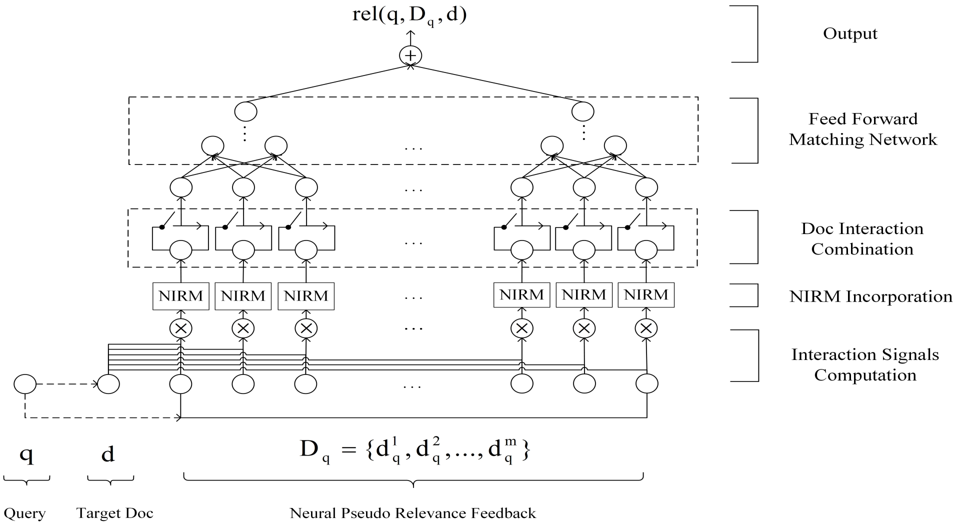

Figure 1: Architecture of the proposed neural pseudo relevance feedback (NPRF) framework.

$rel_D(q, D_q, d)$. When combining the relevance scores, the agreement between q and each d_q is also important, since d_q may differ from q in terms of information needs. The relevance of d_q from the initial ranking $rel_q(q, d_q)$ is employed to quantify this agreement and weight each $rel_d(d_q, d)$ accordingly.

When computing such agreements, it is necessary to remove the influence of the absolute ranges of the scores from the initial ranker. For example, ranking scores from a language model (Ponte and Croft, 1998) and from BM25 (Robertson et al., 1995) can differ substantially in their absolute ranges. To mitigate this, we use a smoothed min-max normalization to rescale $rel_q(q, d_q)$ into the range $[0.5, 1]$. The min-max normalization is applied by considering $min(rel_q(q, d_q)|d_q \in D_q)$ and $max(rel_q(q, d_q)|d_q \in D_q)$. Hereafter, $rel_q(q, d_q)$ is used to denote this relevance score after min-max normalization for brevity. The (normalized) relevance score is smoothed and then weighted by the relevance evaluation of d_q, producing a weighted document relevance score $rel_d{}'(d_q, d)$ for each $d_q \in D_q$ that reflects the relevance of d_q relative to q. This computation is described in the following equation.

$$rel_d{}'(d_q, d) = rel_d(d_q, d)(0.5 + 0.5 \times rel_q(q, d_q)) \quad (1)$$

As the last step, we propose two variants for combining the $rel_d{}'(d_q, d)$ for different d_q into a single score $rel_D(q, D_q, d)$: (i) performing a direct summation and (ii) using a feed forward network with a hyperbolic tangent ($tanh$) non-linear activation. Namely, the first variant simply sums up the scores, whereas the second takes the ranking positions of individual feedback documents into account.

2.3 Optimization and Training

Each training sample consists of a query q, a set of m feedback documents D_q, a relevant target document d^+ and a non-relevant target document d^- according to the ground truth. The Adam optimizer (Kingma and Ba, 2014) is used with a learning rate 0.001 and a batch size of 20. Training normally converges within 30 epochs, with weights uniformly initialized. A hinge loss is employed for training as shown below.

$$loss(q, D_q, d^+, d^-) =$$
$$max(0, 1 - rel(q, D_q, d^+) + rel(q, D_q, d^-))$$

3 Evaluation

3.1 Evaluation Setup

Dataset. We evaluate our proposed NPRF framework on two standard test collections, namely, TREC1-3 (Harman, 1993) and Robust04 (Voorhees, 2004). TREC1-3 consists of 741,856 documents with 150 queries used in the TREC 1-3 ad-hoc search tasks (Harman, 1993, 1994, 1995). Robust04 contains 528,155 documents and 249 queries used in the TREC 2004 Robust track (Voorhees, 2004). We use those two collections to balance between the number of queries and the TREC pooling depth, i.e., 100 on both collections, allowing for sufficient training data. Manual relevance judgments are available on both collections, where both the relevant and non-relevant documents are labeled for each query.

Two versions of queries are included in our experiments: a short keyword query (*title query*), and a longer description query that restates the corresponding keyword query's information need in terms of natural language (*description query*). We evaluate each type of query separately using the metrics Mean Average Precision at 1,000 (MAP), Precision at 20 (P@20) (Manning et al., 2008), and NDCG@20 (Järvelin and Kekäläinen, 2002).

Preprocessing. Stopword removal and Porter's stemmer are applied (Manning et al., 2008). The word embeddings are pre-trained based on a pool of the top 2,000 documents returned by BM25 for individual queries as suggested by (Diaz et al., 2016). The implementation of Word2Vec[1] from (Mikolov et al., 2013) is employed. In particular, we employ CBOW with the dimension set to 300, window size to 10, minimum count to 5, and a subsampling threshold of 10^{-3}. The CBOW model is trained for 10 iterations on the target corpus.

Unsupervised ranking models serve as baselines for comparisons. We use the open source Terrier platform's (Macdonald et al., 2012) implementation of these ranking models:

- **BM25** (Robertson et al., 1995), a classical probabilistic model, is employed as an unsupervised baseline. The hyper-parameters b and k_1 are tuned by grid search. As mentioned in Sec. 2.1, BM25 also generates the initial rankings D_q, serving as $rel_q(.,.)$ in the NPRF framework.

- On top of BM25, we use an adapted version of Rocchio's query expansion (Ye et al., 2009), denoted as **BM25+QE**. Note that, as demonstrated in the results, BM25+QE's performance is comparable with the base neural IR models, including DRMM, K-NRM and PACRR. This illustrates the difficulty in making improvements on the TREC benchmarks through the uses of deep learning methods. The hyper-parameters, including the number of feedback documents and the number of expansion terms, are optimized using grid search on training queries.

- In addition, **QL+RM3**, the query likelihood language model with the popular RM3 PRF (Lavrenko and Croft, 2001), is used as another unsupervised baseline.

Neural IR models are used for $rel_d(.,.)$. As mentioned in Section 2.1, two unigram neural IR models are employed in our experiments:

- **DRMM**. We employ the variant with the best effectiveness on Robust04 according to (Guo et al., 2016), namely, DRMM$_{LCH\times IDF}$ with the original configuration.

- **K-NRM**. Due to the lack of training data compared with the commercial data used by (Xiong et al., 2017), we employ a K-NRM variant with a frozen word embedding layer. To compensate for this substantial reduction in the number of learnable weights, we add an additional fully connected layer to the model. These changes lead to a small but competitive K-NRM variant, as demonstrated in (Hui et al., 2018).

- We additionally implement **PACRR** (Hui et al., 2017) for the purpose of performing comparisons, but do not use PACRR to compute $rel_d(.,.)$ due to the computational costs. In particular, PACRR-firstk is employed where the first $1,000$ terms are used to compute the similarity matrices, and the original configuration from (Hui et al., 2017) is used.

- **NIRM(QE)** uses the modified query generated by the query expansion of BM25+QE (Ye et al., 2009) as input to the neural IR model. Both DRMM and K-NRM are used to instantiate NIRM(QE).

- **Variants of the proposed NPRF approach.** As indicated in Section 2.2, NPRF includes two variants that differ in the combination of the relevance scores from different $d_q \in D_q$: the variant **NPRF$_{\mathbf{ff}}$** uses a feed forward network with a hidden layer with five neurons to compute $rel(d, D_q)$, and the other variant **NPRF$_{\mathbf{ds}}$** performs a direct summation of the different relevance scores. For the purposes of comparison, we additionally introduce another variant coined **NPRF$_{\mathbf{ff'}}$**, where the relevance of d_q to q is not considered in the combination by directly setting $rel'_d(d, d_q) = rel_d(d, d_q)$ in place of Equation 1, thereafter combining the scores with a fully connected layer as in NPRF$_{ff}$. We combine each of the three NPRF variants with

---

[1] https://code.google.com/p/word2vec/

the DRMM and K-NRM models, and report results for all six variants. Our implementation of the NPRF framework is available to enable future comparisons[2].

Akin to (Guo et al., 2016; Xiong et al., 2017; Hui et al., 2017), the NIRM baselines and the proposed NPRF are employed to re-rank the search results from BM25. In particular, the top-10 documents from the unsupervised baseline are used as the pseudo relevance feedback documents D_q as input for NPRF, where each $d_q \in D_q$ is represented by its top-20 terms with the highest tf-idf weights. As illustrated later in Section 3.3, NPRF's performance is stable over a wide range of settings for both parameters.

Cross-validation. Akin to (Hui et al., 2018), owing to the limited number of labeled data, five-fold cross-validation is used to report the results by randomly splitting all queries into five equal partitions. In each fold, three partitions are used for training, one for validation, and one for testing. The model with the best MAP on the validation set is selected. We report the average performance on all *test* partitions. A two-tailed paired t-test is used to report the statistical significance at 95% confidence interval.

3.2 Results

Comparison to BM25. We first compare the proposed NPRF models with the unsupervised BM25. The results are summarized in Tables 1 and 2, where the best result in each column is highlighted in bold. From Tables 1 and 2, it can be seen that the proposed NPRF variants obtain significant improvement relative to BM25 on both test collections with both kinds of test queries. Moreover, the results imply that the use of different query types does not affect the effectiveness of NPRF, which consistently outperforms BM25.

Comparison to neural IR models. NPRF is further compared with different neural IR models, as summarized in Tables 3 & 4. It can be seen that NPRF regularly improves on top of the NIRM baselines. For both types of queries, NPRF-DRMM outperforms DRMM and NPRF-KNRM outperforms K-NRM when re-ranking BM25. Remarkably, the proposed NPRF is able to improve the weaker NIRM baseline. For instance, on Robust04, when using the description queries, DRMM and K-NRM obtain highly different results, with MAPs of 0.2630 and 0.1687 after reranking the initial results from BM25, respectively. When NPRF is used in conjunction with the NIRM models, however, the gap between the two models is closed; that is, MAP=0.2801 for NRFF$_{ds}$-DRMM and MAP=0.2800 for NRFF$_{ds}$-KNRM (see Table 4). This finding highlights that our proposed NPRF is robust with respect to the use of the two embedded NIRM models. A possible explanation for the poor performance of K-NRM on two TREC collections is the lack of training data, as suggested in (Dai et al., 2018). While K-NRM could be improved by introducing weak supervision (Dai et al., 2018), we achieve the same goal by incorporating pseudo relevance feedback information without extra training data.

While the six NPRF variants exhibit similar results across both kinds of queries, NPRF$_{ds}$-DRMM in general achieves the best performance on Robust04, and NPRF$_{ds}$-KNRM appears to be the best variant on TREC1-3. In the meantime, NPRF$_{ds}$ outperforms NPRF$_{ff}$ variants. One difference between the two methods is that NPRF$_{ff}$ considers the position of each d_q in the D_q ranked documents, whereas NPRF$_{ds}$ simply sums up the scores regardless of the positions. The fact that NPRF$_{ds}$ performs better suggests that the ranking position within the D_q documents may not be a useful signal. In the remainder of this paper, we mainly report on the results obtained by NPRF$_{ds}$.

Comparison to query expansion baselines. In Table 5, the proposed NPRF model is compared with three kinds of query expansion baselines, namely, the unsupervised BM25+QE (Ye et al., 2009), QL+RM3 (Lavrenko and Croft, 2001), and DRMM/K-NRM(QE), the neural IR models using expanded queries as input. According to Table 5, the unsupervised BM25+QE baseline appears to achieve better performance in terms of MAP@1k, owing to its use of query expansion to match relevant documents containing the expansion terms from the whole collection. On the other hand, NPRF$_{ds}$, which reranks the top-1000 documents returned by BM25, outperforms the query expansion baselines in terms of early precision, as measured by either NDCG@20 or P@20. These measures on shallow rankings are particularly important for general IR applications where the quality of the top-ranked results is crucial to the user satisfaction. Moreover, our NPRF outperforms

---

[2] https://github.com/ucasir/NPRF

	Title						Description					
Model	MAP		P@20		NDCG@20		MAP		P@20		NDCG@20	
BM25	0.2408	-	0.4803	-	0.4947	-	0.2094	-	0.4613	-	0.4838	-
NPRF$_{ff}$-DRMM	0.2669†	10.85%	0.5010	4.31%	0.5119	3.47%	0.2509†	19.80%	0.5257†	13.95%	0.5393†	11.46%
NPRF$_{ff'}$-DRMM	0.2671†	10.93%	0.5023†	4.59%	0.5116	3.42%	0.2504†	19.58%	0.5163†	11.93%	0.5291†	9.37%
NPRF$_{ds}$-DRMM	0.2698†	12.03%	0.5187†	7.99%	0.5282†	6.77%	**0.2527**†	20.67%	**0.5283**†	14.53%	0.5444†	12.52%
NPRF$_{ff}$-KNRM	0.2633†	9.34%	0.5033	4.80%	0.5171	4.52%	0.2486†	18.71%	0.5240†	13.59%	0.5398†	11.58%
NPRF$_{ff'}$-KNRM	0.2654†	10.22%	0.5077†	5.70%	0.5216†	5.44%	0.2462†	17.60%	0.5197†	12.65%	0.5363†	10.84%
NPRF$_{ds}$-KNRM	**0.2707**†	12.41%	**0.5303**†	10.42%	**0.5406**†	9.29%	0.2505†	19.61%	0.5270†	14.24%	**0.5460**†	12.87%

Table 1: Comparisons between NPRF and *BM25* on *TREC1-3* dataset. Relative performances compared with *BM25* are in percentages. Significant improvements relative to the baselines are marked with †.

	Title						Description					
Model	MAP		P@20		NDCG@20		MAP		P@20		NDCG@20	
BM25	0.2533	-	0.3612	-	0.4158	-	0.2479	-	0.3514	-	0.4110	-
NPRF$_{ff}$-DRMM	0.2823†	11.46%	0.3941†	9.11%	0.4350†	4.62%	0.2766†	11.58%	0.3908†	11.21%	0.4421†	7.56%
NPRF$_{ff'}$-DRMM	0.2837†	12.00%	0.3928†	8.74%	0.4377†	5.27%	0.2774†	11.90%	0.3984†	13.38%	0.4493†	9.32%
NPRF$_{ds}$-DRMM	**0.2904**†	14.66%	**0.4064**†	12.52%	**0.4502**†	8.28%	**0.2801**†	12.95%	**0.4026**†	14.57%	**0.4559**†	10.92%
NPRF$_{ff}$-KNRM	0.2809†	10.90%	0.3851†	6.62%	0.4287	3.11%	0.2720†	9.71%	0.3867†	10.06%	0.4356†	5.99%
NPRF$_{ff'}$-KNRM	0.2815†	11.13%	0.3882†	7.48%	0.4264	2.55%	0.2737†	10.39%	0.3892†	10.74%	0.4382†	6.61%
NPRF$_{ds}$-KNRM	0.2846†	12.36%	0.3926†	8.69%	0.4327	4.06%	0.2800†	12.95%	0.3972†	13.03%	0.4477†	8.94%

Table 2: Comparisons between NPRF and *BM25* on the *Robust04* dataset. Relative performances compared with *BM25* are in percentages. Significant improvements relative to the baselines are marked with †.

NIRM(QE) in most cases, indicating the benefit brought by wrapping up the feedback information in a document-to-document matching framework as in NPRF, as opposed to directly adding unweighted expansion terms to the query. Recall that, it is not straightforward to incorporate these expanded terms within the existing NIRMs' architectures because the NIRMs do not distinguish between them and the original query terms.

3.3 Analysis

Parameter sensitivity. Moreover, we analyze factors that may influence NPRF's performance. We report results on NPRF$_{ds}$ using title queries on Robust04 for the sake of brevity, but similar observations also hold for the other NPRF variants, as well as on TREC1-3. Figure 2 illustrates the sensitivity of NPRF relative to two parameters: the number of feedback documents m within D_q and the number of terms k that are used to summarize each $d_q \in D_q$. Specifically, Figure 2 shows the performance of NPRF$_{ds}$ as the number of feedback documents m varies (top), and as the number of top terms k varies (bottom). The effectiveness of NPRF appears to be stable over a wide range of the parameter configurations, where the proposed model consistently outperforms the BM25 baseline.

Case study. A major advantage of the proposed NPRF over existing neural IR models is that it allows for soft-matching query-related terms that are missing from both the query and the target document. Table 6 presents an illustrative example of soft matching in NPRF. From Table 6, it can be seen that there exist query-related terms in the top-10 documents returned by BM25 in the initial ranking. However, since those query-related terms are missing in both the query and the target document, they are not considered in the document-to-query matching and, consequently, the target document is ranked 122^{nd} by BM25 despite the facts that it was judged relevant by a human assessor. In contrast, the NPRF framework allows for the soft-matching of terms that are missing in both the query and target document. As a result, the matching signals for the query terms and query-related terms in the target document are enhanced. This leads to enhanced effectiveness with the target document now ranked in the 5^{th} position.

In summary, the evaluation on two standard TREC test collections shows promising results obtained by our proposed NPRF approach, which outperforms state-of-the-art neural IR models in most cases. Overall, NPRF provides effective retrieval performance that is robust with respect to the two embedded neural models used for encoding the document-to-document interactions, the two kinds of queries with varied length, and wide range of parameter configurations.

Model	Title						Description					
	MAP		P@20		NDCG@20		MAP		P@20		NDCG@20	
DRMM	0.2469	-	0.4833	-	0.4919	-	0.2111	-	0.4423	-	0.4546	-
K-NRM	0.2284	-	0.4410	-	0.4530	-	0.1763	-	0.3753	-	0.3854	-
PACRR-firstk	0.2393	-	0.4620	-	0.4782	-	0.1702	-	0.3577	-	0.3666	-
NPRF$_{ff}$-DRMM	0.2669†	8.12%	0.5010	3.66%	0.5119	4.06%	0.2509†	18.83%	0.5257†	18.84%	0.5393†	18.63%
NPRF$_{ff'}$-DRMM	0.2671†	8.19%	0.5023	3.94%	0.5116	4.01%	0.2504†	18.61%	0.5163†	16.73%	0.5291†	16.40%
NPRF$_{ds}$-DRMM	0.2698†	9.26%	0.5187†	7.32%	0.5282†	7.38%	**0.2527**†	19.69%	**0.5283**†	19.44%	0.5444†	19.76%
NPRF$_{ff}$-KNRM	0.2633†	15.28%	0.5033†	14.13%	0.5171†	14.14%	0.2486†	40.97%	0.5240†	39.61%	0.5398†	40.06%
NPRF$_{ff'}$-KNRM	0.2654†	16.20%	0.5077†	15.12%	0.5216†	15.15%	0.2462†	39.65%	0.5197†	38.45%	0.5363†	39.13%
NPRF$_{ds}$-KNRM	**0.2707**†	18.51%	**0.5303**†	20.26%	**0.5406**†	19.35%	0.2505†	42.04%	0.5270†	40.41%	**0.5460**†	41.67%

Table 3: Comparisons between NPRF and neural IR models on *TREC1-3*. Relative performances of NPRF-DRMM(KNRM) compared with DRMM (K-NRM) are in percentages, and statistically significant improvements are marked with †.

Model	Title						Description					
	MAP		P@20		NDCG@20		MAP		P@20		NDCG@20	
DRMM	0.2688	-	0.3713	-	0.4297	-	0.2630	-	0.3558	-	0.4135	-
K-NRM	0.2464	-	0.3510	-	0.3989	-	0.1687	-	0.2301	-	0.2641	-
PACRR-firstk	0.2540	-	0.3631	-	0.4082	-	0.2087	-	0.2962	-	0.3362	-
NPRF$_{ff}$-DRMM	0.2823	5.03%	0.3941†	6.14%	0.4350	1.24%	0.2766†	5.17%	0.3908†	9.84%	0.4421†	6.92%
NPRF$_{ff'}$-DRMM	0.2837†	5.55%	0.3928	5.78%	0.4377	1.87%	0.2774†	5.48%	0.3984†	11.97%	0.4493†	8.67%
NPRF$_{ds}$-DRMM	**0.2904**†	8.05%	**0.4064**†	9.46%	**0.4502**	4.78%	**0.2801**†	6.46%	**0.4026**†	13.15%	**0.4559**†	10.26%
NPRF$_{ff}$-KNRM	0.2809†	14.00%	0.3851†	9.72%	0.4287†	7.48%	0.2720†	61.22%	0.3867†	68.08%	0.4356†	64.96%
NPRF$_{ff'}$-KNRM	0.2815†	14.25%	0.3882†	10.60%	0.4264†	6.90%	0.2737†	62.21%	0.3892†	69.12%	0.4382†	65.93%
NPRF$_{ds}$-KNRM	0.2846†	15.50%	0.3926†	11.85%	0.4327†	8.47%	0.2800†	65.98%	0.3972†	72.62%	0.4477†	69.55%

Table 4: Comparisons between NPRF and neural IR models on *Robust04*. Relative performances of NPRF-DRMM(KNRM) compared with DRMM (K-NRM) are in percentages, and statistically significant improvements are marked with †.

4 Related Work

Recently, several neural IR models (NIRMs) have been proposed to apply deep learning techniques in ad-hoc information retrieval. One of the essential ideas from prior work is to model the document-to-query *interaction* via neural networks, based on a matrix of document-to-query embedding term similarities, incorporating both the "exact matching" of terms appearing in both the document and query and the "soft matching" of different query and document term pairs that are semantically related.

DSSM, one of the earliest NIRMs proposed in (Huang et al., 2013), employs a multi-layer neural network to project queries and document into a common semantic space. The cosine similarity between a query and a document (document title) is used to produce a final relevance score for the query-document pair. CDSSM is a convolutional version of DSSM, which uses the convolutional neural network (CNN) and max-pooling strategy to extract semantic matching features at the sentence level (Shen et al., 2014). (Pang et al., 2016) also employ a CNN to construct the MatchPyramid model, which learns hierarchical matching patterns between local interactions of document-query pair. (Guo et al., 2016) argue that both DSSM and CDSSM are representation-focused models, and thus are better suited to capturing semantic matching than relevance matching (i.e., lexical matching), and propose the interaction-focused relevance model named DRMM. DRMM maps the local interactions between a query-document pair into a fixed-length histogram, from which the exact matching signals are distinguished from the other matching signals. These signals are fed into a feed forward network and a term gating network to produce global relevance scores. Similar to DRMM, K-NRM (Xiong et al., 2017) builds its model on top of a matrix of local interaction signals, and utilizes multiple Gaussian kernels to obtain multi-level exact/soft matching features that are input into a ranking layer to produce the final ranking score. K-NRM is later improved by Conv-KNRM, which employs CNN filters to capture n-gram representations of queries and documents (Dai et al., 2018). DeepRank (Pang et al., 2017) models the relevance generation process by identifying query-centric contexts, processing them with a CNN or LSTM, and aggregating them to produce a final relevance score. Building upon DeepRank, (Fan et al., 2018) propose to model diverse relevance patterns by a data-driven method to allow rele-

	TREC1-3						Robust04					
	Title			Description			Title			Description		
Model	MAP	P@20	NDCG@20	MAP	P@20	NDCG@20	MAP	P@20	NDCG@20	MAP	P@20	NDCG@20
BM25+QE	**0.2873**	0.5200	0.5330	**0.2601**	0.4973	0.5093	**0.2966**	0.3839	0.4353	**0.2926**	0.3817	0.4340
QL+RM3	0.2734	0.5093	0.5198	0.2421	0.4627	0.4801	0.2842	0.3878	0.4398	0.2686	0.3506	0.4150
DRMM (QE)	0.2741	0.5183	0.5345	0.2380	0.5077	0.5229	0.2876	0.4002	**0.4549**	0.2711	0.3822	0.4392
K-NRM (QE)	0.2633	0.5127	0.5235	0.2307	0.4877	0.5039	0.2521	0.3644	0.4062	0.2380	0.3304	0.3785
NPRF$_{ds}$-DRMM	0.2698	0.5187	0.5282	0.2527	**0.5283**	0.5444†	0.2904	**0.4064**	0.4502	0.2801	**0.4026**†	**0.4559**†
NPRF$_{ds}$-KNRM	0.2707	**0.5303**	**0.5406**	0.2505	0.5270	**0.5460**†	0.2846	0.3926	0.4327	0.2800	0.3972†	0.4477

Table 5: Comparisons between NPRF and *query expansion baselines* on *TREC1-3* and *Robust04*. Significant improvements over the best baseline is marked with †.

TREC Query 341: airport security	
Terms in doc at rank *i*	Terms in target document FBIS3-23332
1. terrorist detect passenger check police scan; 2. heathrow terrorist armed aviation police; 3. detect airline passenger police scan flight weapon; 4. aviation; 5. detect baggage passenger; 6. passenger bomb baggage terrorist explosive aviation scan flight weapon; 7. baggage airline detect passenger scan flight weapon; 8. baggage airline passenger flight; 9. passenger police aviation; 10. airline baggage aviation flight	transec semtex airline ditma **security** baggage heathrow test device lockerbie klm bomb virgin **airport** loaded blobby transport detect inspector terrorist identify atlantic depressing passenger fail aircraft dummy check inert patchy stein norwich doll regard rupert lapse busiest loophole employee campaign blew procedure traveler passport reconcile glasgow investigate boeing bags bag harry successive smuggle conscious reconciliation tragedy board wire hidden...

Table 6: An illustrative example of soft matching in NPRF. The target document FBIS3-23332, judged relevant, is ranked 122^{nd} by BM25 for query 341 on Robust04, and is promoted to the 5^{th} by NPRF$_{ds}$-DRMM. The NPRF mechanism increases the chances of soft-matching query-related terms that appear in the top-ranked documents (terms in blue), but are missing in both the query and the target document. Subsequently, the matching signals with the query terms (in **bold**) and the query-related terms (in red) in the target document are enhanced.

vance signals at different granularities to compete with each other for the final relevance assessment.

Duet (Mitra et al., 2017) employs two separate deep neural networks to build a relevance ranking model, in which a local model estimates the relevance score according to exact matches between query and document terms, and a distributed model estimates relevance by learning dense lower-dimensional representations of query and document text. (Zamani et al., 2018) extends the Duet model by considering different fields within a document.

(Hui et al., 2017) propose the PACRR model based on the idea that an appropriate combination of convolutional kernels and pooling operations can be used to successfully identify both unigram and n-gram query matches. PACRR is later improved upon by Co-PACRR, a context-aware variant that takes the local and global context of matching signals into account through the use of three new components (Hui et al., 2018).

(Ran et al., 2017) propose a document-based neural relevance model that utilizes complemented medical records to address the mismatch problem in clinical decision support. (Nogueira and Cho, 2017) propose a reinforcement learning approach to reformulating a task-specific query. (Li et al., 2018) propose DAZER, a CNN-based neural model upon interactions between seed words and words in a document for zero-shot document filtering with adversarial learning. (Ai et al., 2018) propose to refine document ranking by learning a deep listwise context model.

In summary, most existing neural IR models are based on query-document interaction signals and do not provide a mechanism for incorporating relevance feedback information. This work proposes an approach for incorporating relevance feedback information by embedding neural IR models within a neural pseudo relevance feedback framework, where the models consume feedback information via document-to-document interactions.

5 Conclusions

In this work we proposed a neural pseudo relevance feedback framework (NPRF) for incorporating relevance feedback information into existing neural IR models (NIRM). The NPRF framework uses feedback documents to better estimate relevance scores by considering individual feedback documents as different interpretations of the user's information need. On two standard TREC datasets, NPRF significantly improves the performance of two state-of-the-art NIRMs. Furthermore, NPRF was able to improve their perfor-

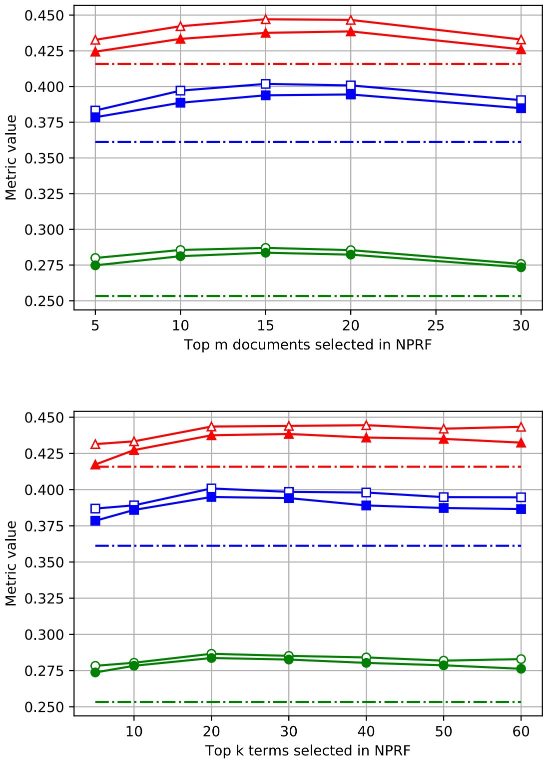

Figure 2: Performance of NPRF$_{ds}$ with different numbers of PRF documents (top) and different umber of terms which are used to summarize the feedback documents (bottom). The ○, □, △ correspond to results measured by MAP, P@20 and NDCG@20 respectively, and the empty or solid symbols correspond to those for NPRF$_{ds}$-DRMM and NPRF$_{ds}$-KNRM. The three dotted lines, from bottom to top, are the BM25 baseline evaluated by MAP, P@20 and NDCG@20, respectively.

mance across two kinds of query tested (namely, short queries and the verbal queries in natural language). Finally, our analysis demonstrated the robustness of the NPRF framework over different parameter configurations.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (61433015/61472391), and the Beijing Natural Science Foundation under Grant No. (4162067/4142050).

References

Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a deep listwise context model for ranking refinement. In *SIGIR*, pages 135–144. ACM.

Chris Buckley and Stephen E. Robertson. 2008. Relevance feedback track overview: Trec 2008. In *TREC*. NIST.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *WSDM*, pages 126–134. ACM.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. In *ACL*. The Association for Computer Linguistics.

Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *SIGIR*, pages 375–384. ACM.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *CIKM*, pages 55–64. ACM.

Donna Harman. 1993. Overview of the first text retrieval conference. In *SIGIR*, pages 36–47. ACM.

Donna Harman. 1994. Overview of the third text retrieval conference (TREC-3). In *TREC*, volume Special Publication 500-225, pages 1–20. National Institute of Standards and Technology (NIST).

Donna Harman. 1995. Overview of the second text retrieval conference (TREC-2). *Inf. Process. Manage.*, 31(3):271–289.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, pages 2333–2338. ACM.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A position-aware neural IR model for relevance matching. In *EMNLP*, pages 1049–1058. Association for Computational Linguistics.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-pacrr: A context-aware neural IR model for ad-hoc retrieval. In *WSDM*, pages 279–287. ACM.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-based language models. In *SIGIR*, pages 120–127. ACM.

Chenliang Li, Wei Zhou, Feng Ji, Yu Duan, and Haiqing Chen. 2018. A deep relevance model for zero-shot document filtering. In *ACL*, pages 2300–2310. The Association for Computer Linguistics.

Craig Macdonald, Richard McCreadie, Rodrygo Santos, and Iadh Ounis. 2012. From puppy to maturity: Experiences in developing terrier. In *OSIR at SIGIR*, pages 60–63.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *WWW*, pages 1291–1299. ACM.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *EMNLP*, pages 574–583. Association for Computational Linguistics.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016. A study of matchpyramid models on ad-hoc retrieval. *CoRR*, abs/1606.04648.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *CIKM*, pages 257–266. ACM.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *SIGIR*, pages 275–281. ACM.

Yanhua Ran, Ben He, Kai Hui, Jungang Xu, and Le Sun. 2017. A document-based neural relevance model for effective clinical decision support. In *BIBM*, pages 798–804. IEEE Computer Society.

Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Mike Gatford, and A. Payne. 1995. Okapi at TREC-4. In *TREC*, volume Special Publication 500-236. National Institute of Standards and Technology (NIST).

J. Rocchio. 1971. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART retrieval system: experiments in automatic document processing*, pages 313–323. Prentice Hall, Englewood, Cliffs, New Jersey.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW (Companion Volume)*, pages 373–374. ACM.

Ellen M. Voorhees. 2004. Overview of the TREC 2004 robust track. In *TREC*, volume Special Publication 500-261. National Institute of Standards and Technology (NIST).

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *SIGIR*, pages 55–64. ACM.

Zheng Ye, Xiangji Huang, Ben He, and Hongfei Lin. 2009. York university at TREC 2009: Relevance feedback track. In *TREC*, volume Special Publication 500-278. National Institute of Standards and Technology (NIST).

Hamed Zamani, Bhaskar Mitra, Xia Song, Nick Craswell, and Saurabh Tiwary. 2018. Neural ranking models with multiple document fields. In *WSDM*, pages 700–708. ACM.

Co-Stack Residual Affinity Networks with Multi-level Attention Refinement for Matching Text Sequences

Yi Tay$^{\dagger *}$, **Luu Anh Tuan**$^{\psi *}$, **Siu Cheung Hui**$^{\phi}$
$^{\dagger\phi}$Nanyang Technological University, Singapore
$^{\psi}$Institute for Infocomm Research, A*Star Singapore
ytay017@e.ntu.edu.sg†, at.luu@i2r.a-star.edu.sg$^{\psi}$
asschui@ntu.edu.sg$^{\phi}$

Abstract

Learning a matching function between two text sequences is a long standing problem in NLP research. This task enables many potential applications such as question answering and paraphrase identification. This paper proposes Co-Stack Residual Affinity Networks (CSRAN), a new and universal neural architecture for this problem. CSRAN is a deep architecture, involving stacked (multi-layered) recurrent encoders. Stacked/Deep architectures are traditionally difficult to train, due to the inherent weaknesses such as difficulty with feature propagation and vanishing gradients. CSRAN incorporates two novel components to take advantage of the stacked architecture. Firstly, it introduces a new bidirectional alignment mechanism that learns affinity weights by fusing sequence pairs across stacked hierarchies. Secondly, it leverages a multi-level attention refinement component between stacked recurrent layers. The key intuition is that, by leveraging information across all network hierarchies, we can not only improve gradient flow but also improve overall performance. We conduct extensive experiments on six well-studied text sequence matching datasets, achieving state-of-the-art performance on all.

1 Introduction

Determining the semantic affinity between two text sequences is a long standing research problem in natural language processing research. This is understandable, given that technical innovations in this domain would naturally bring benefits to a diverse plethora of applications ranging from paraphrase detection to standard document retrieval. This work focuses on short textual sequences, focusing on a myriad of applications such as natural language inference, question answering, reply prediction and paraphrase detection. This paper presents a new deep matching model for universal text matching.

Neural networks are dominant state-of-the-art approaches for many of these matching problems (Gong et al., 2017; Shen et al., 2017; Wang et al., 2017; Chen et al., 2017). Fundamentally, neural networks operate via a concept of feature hierarchy, in which hierarchical representations are constructed as sequences propagate across the network. In the context of matching, representations are often (1) encoded, (2) matched, and then (3) aggregated for prediction. Each key step often comprises several layers, which consequently adds to the overall depth of the network.

Unfortunately, it is a well established fact that deep networks are difficult to train. This is attributed to not only vanishing/exploding gradients but also an instrinsic difficulty pertaining to feature propagation. To this end, commonly adopted solutions include Residual connections (He et al., 2016) and/or Highway layers (Srivastava et al., 2015). The key idea in these approaches is to introduce additional (skip/residual) connections, propagating shallower layers to deeper layers via shortcuts. To the best of our knowledge, these techniques are generally applied to single sequences and therefore the notion of pairwise residual connections have not been explored.

This paper presents Co-Stack Residual Affinity Networks (CSRAN), a stacked multi-layered recurrent architecture for general purpose text matching. Our model proposes a new co-stacking mechanism that computes bidirectional affinity scores by leveraging all feature hierarchies between text sequence pairs. More concretely, word-by-word affinity scores are not computed just from the final encoded representations but across all the entire feature hierarchy.

There are several benefits to our co-stacking

---

[*] Denotes equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4492–4502
Brussels, Belgium, October 31 - November 4, 2018.

mechanism. Firstly, co-stacking acts as a form of residual connector, alleviating the instrinsic issues with network depth. Secondly, there are more extensive matching interfaces between text sequences as the affinity matrix is not computed by just one representation but multiple representations instead. Naturally, increasing the opportunities for interactions between sequences is an intuitive method for improving performance.

Additionally, our model incorporates a Multi-level Attention Refinement (MAR) architecture in order to fully leverage the stacked recurrent architecture. The MAR architecture is a multi-layered adaptation and extension of the CAFE model (Tay et al., 2017c), in which attention is computed, compressed and then re-fed into the input sequence. In our approach, we use CAFE blocks to repeatedly *refine* representations at each level of the stacked recurrent encoder.

The overall outcome of the above-mentioned architectural synergies is a highly competitive model that establishes state-of-the-art performance on six well-known text matching datasets such as SNLI and TrecQA. The overall contributions of this work are summarized as follows:

- We propose a new deep stacked recurrent architecture for matching text sequences. Our model is based on a new co-stacking mechanism which learns to align by exploiting matching across feature hierarchies. This can be interpreted as a new way to incorporate shortcut connections within neural models for sequence matching. Additionally, we also propose a multi-level attention refinement scheme to leverage our stacked recurrent model.
- While stacked architectures can potentially lead to considerable improvements in performance, our experiments show that in the absence of our proposed CSRA (Co-stack Residual Affinity) mechanism, stacking may conversely lead to performance degradation. As such, this demonstrates that our proposed techniques are essential for harnessing the potential of deep architectures.
- We conduct extensive experiments on four text matching tasks across six well-studied datasets, i.e., *Natural Language Inference* (SNLI (Bowman et al., 2015), SciTail (Khot et al., 2018)), *Paraphrase Identification* (Quora, TwitterURL (Lan et al., 2017)), *Answer Sentence Selection* (Wang et al., 2007) and *Utterance-Response Matching* (Ubuntu (Lowe et al., 2015)). Our model achieves state-of-the-art performance on all datasets.

2 Co-Stack Residual Affinity Networks

In this section, we introduce our proposed model architecture for general/universal text matching. The key idea of this architecture is to leverage deep stacked layers, while mitigating the inherent weaknesses of going deep. As such, our network is in similar spirit to highway networks, residual networks and DenseNets, albeit tailored specifically for pairwise architectures. Figure 1 illustrates a high-level overview of our proposed model architecture.

2.1 Input Encoder

The inputs to our model are standard sequences of words A and B which represent sequence a and sequence b respectively. In the context of different applications, a and b take different roles such as premise/hypothesis or question/answer. Both sequences are converted into word representations (via pretrained word embeddings) and character-based representations. Character embeddings are trainable parameters and a final character-based word representation of d dimensions is learned by passing all characters into a Bidirectional LSTM encoder. This is standard, following many works such as (Wang et al., 2017). Word embeddings and character-based word representations are then concatenated to form the final word representation. Next, the word representation is passed through a (optional and tuned as a hyperparameter) 2-layered highway network of d dimensions.

2.2 Stacked Recurrent Encoders

Next, word representations are passed into a stacked recurrent encoder layer. Specifically, we use Bidirectional LSTM encoders at this layer. Let k be the number of layers of the stacked recurrent encoder layer.

$$h^i_t = \text{BiLSTM}_i(h_{t-1}) \;\; \forall t \in [1, 2 \cdots \ell] \quad (1)$$

where BiLSTM_i represents the i-th BiLSTM layer and h^i_t represents the t-th hidden state of the i-th BiLSTM layer. ℓ is the sequence length. Note that the parameters are shared for both a and b.

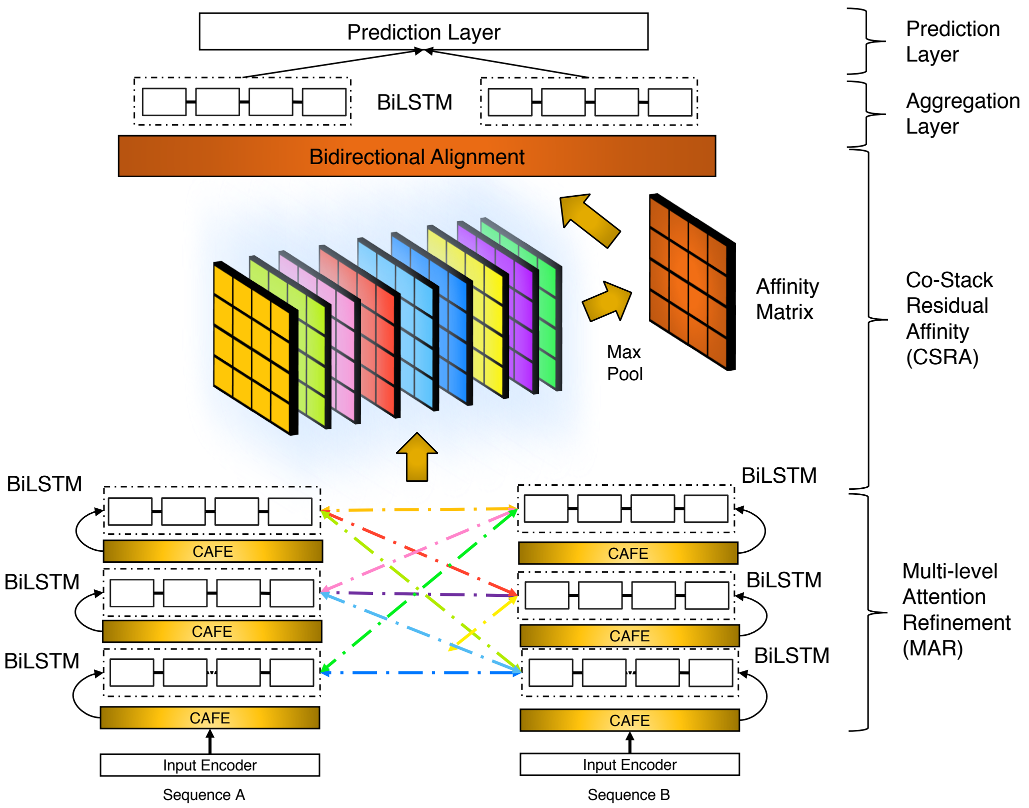

Figure 1: Illustration of the proposed Co-Stack Residual Affinity Network (CSRAN) architecture. Each color coded matrix represents the interactions between two layers of sequence A and sequence B. (Best viewed in color)

2.3 Multi-level Attention Refinement (MAR)

Inspired by CAFE (Tay et al., 2017c) (Compare-Align-Factorized Encoders), a top performing model on the SNLI benchmark, we utilize CAFE blocks between the BiLSTM layers. Each CAFE block returns *six* features, which are generated by a factorization operation using factorization machines (FM). While the authors in (Tay et al., 2017c) simply use this operation in a single layer, we utilize this in a multi-layered fashion which we found to have worked well. This constitutes our multi-level attention refinement mechanism. More concretely, we apply the CAFE operation to the outputs of each BiLSTM layer, allowing the next BiLSTM layer to process the 'augmented' representations. The next layer retains its dimensionality by projecting the augmented representation back to its original size using the BiLSTM encoder. This can be interpreted as repeatedly refining representations via attention. As such, adding CAFE blocks is a very natural fit to the stacked recurrent architecture.

CAFE Blocks This section describes the operation of each CAFE block. The key idea behind CAFE blocks is to align a and b, and compress alignment vectors such as $b' - a$ (subtraction), $b' \odot a$ (element-wise multiplication) and $[b'; a]$ (concatenation) into scalar features. These scalar features are concatenated to the original input embedding, which can be pased into another BiLSTM layer for refining representations. Firstly, a, b are modeled aligned via $E_{ij} = F(a)^\top F(b)$ and then aligned via:

$$A' = E^\top B \text{ and } B' = AE^\top \tag{2}$$

Given aligned pairs (A', B) and (B', A), we generate *three* matching vectors for the concatenation ($[a'_i; b_i]$), element-wise multiplication ($a'_i \odot b_i$) and subtraction vectors ($a'_i - b_i$) of each pair. After which, we apply a factorization machine (Rendle, 2010) $M(x)$ on each matching vector.

$$M(x) = w_0 + \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle\, x_i\, x_j \tag{3}$$

where $v \in \mathbb{R}^{d \times k}$, $w_0 \in \mathbb{R}$ and $w_i \in \mathbb{R}^d$. The output $M(x)$ is a scalar. Intuitively, this layer tries to learn pairwise interactions between every x_i and x_j using factorized (vector) parameters v.

Factorization machines model low-rank structure within the matching vector, producing a scalar feature. This enables efficient propagation of these matching features to the next layer. The output of each CAFE block is the original input to the CAFE module, augmented with the output of the factorization machines. As such, if the input sequence is of d dimensions, then the output is $d + 3$ dimensions. Additionally, intra-attention is applied in similar fashion as above to generate *three* more features for each sequence. As a result, the output dimensions for each word becomes $d + 6$.

2.4 Co-Stack Residual Affinity (CSRA)

This layer is the cornerstone of our proposed approach and is represented as the middle segment of Figure 1 (the colorful matrices).

Co-Stacking Co-stacking refers to the fusion of a and b across multiple hierarchies. Recall that the affinity score between two words is typically computed by $s_{ij} = a^\top b$. We extend this to a residual formulation. More concretely, the affinity score between both words is now computed as the *maximum* influence it has over all layers.

$$s_{ij} = \max \sum_p \sum_q a_{pi}^\top \, b_{qj} \tag{4}$$

where a_{pi} is the i-th word for the p-th stacked layer for a and b_{qj} is the j-th word for the q-th stacked layer for b. The choice of the maximum operator is intuitive and is strongly motivated by the fact that we would like to give a high affinity for each word pair that shows a strong match at any of different hierarchical stages of learning representations. Note that this layer can be interpreted as constructing a matching tensor based on multi-hierarchical information and selecting the most informative match across all representation hierarchies.

Bidirectional Alignment In order to learn (bidirectionally) attentive representations, we first concatenate all stacked outputs to form a $\ell \times kd$ vector. Next, we apply the following operations to $A \in \mathbb{R}^{\ell_a \times kd}$ and $B \in \mathbb{R}^{\ell_b \times kd}$.

$$\bar{A} = S^\top B \text{ and } \bar{B} = AS^\top \tag{5}$$

where $\bar{A} \in \mathbb{R}^{\ell_b \times kd}, \bar{B} \in \mathbb{R}^{\ell_a \times kd}$ are the attentive (aligned) representations.

2.4.1 Matching and Aggregation Layer

Next, we match the attentive (aligned) representations using the subtraction, element-wise multiplication and concatenation of each aligned word. Subsequently, we pass this matching vector into a k layered BiLSTM layer.

$$a_i' = \text{BiLSTM}_k([\bar{b}_i - a_i, \bar{b}_i \odot a_i, \bar{b}_i, a_i]) \tag{6}$$

$$b_i' = \text{BiLSTM}_k([\bar{a}_i - b_i, \bar{a}_i \odot b_i, \bar{a}_i, b_i]) \tag{7}$$

The final feature representation is learned via the summation across the temporal dimension as follows:

$$z = [\sum_{i=1}^{\ell_a} a_i' \; ; \; \sum_{i=1}^{\ell_b} b_i'] \tag{8}$$

where $[.;.]$ is the concatenation operator.

2.5 Output and Prediction Layer

Our model predicts using the feature vector z for every given sequence pair. At this layer, we utilize standard fully connected layers. The number of output layers is typically 2-3 and is a tuned hyperparameter. Softmax is applied onto the final layer. The final layer is application specific, e.g., k classes for classification tasks and a two-class softmax for pointwise ranking. For all datasets, we optimize the cross entropy loss.

3 Experimental Evaluation

In this section, we introduce our experimental setup, baselines and results.

3.1 Datasets and Competitor Baselines

We use six public benchmark datasets for evaluating our proposed approach. This section briefly introduces each dataset, along with several state-of-the-art approaches that we compare against. Table 1 provides a summary of the datasets used in our experiments.

Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) is a well-known dataset for *entailment classification* (or natural language inference). The task is to determine if two sequences entail/contradict or are neutral to each other. This task is a three-way classification problem. On this dataset, we compare with several state-of-the-art models such as BiMPM (Wang et al., 2017), ESIM (Chen et al., 2017), DIIN (Gong et al., 2017), DR-BiLSTM (Ghaeini et al., 2018) and CAFE (Tay et al., 2017c).

Dataset	Task	$\|C\|$	Pairs
SNLI	Premise-Hypothesis	3	570K
Scitail	Premise-Hypothesis	2	27K
Quora	Question-Question	2	400K
Twitter	Tweet-Tweet	2	51K
TrecQA	Question-Answer	R	56K
Ubuntu	Utterance-Response	R	1M

Table 1: Statistics of datasets used in our experiment. $|C|$ denotes the number of classes and R denotes a ranking formulation. Twitter stands for the TwitterURL dataset.

Science Entailment (SciTail) (Khot et al., 2018) is a new entailment classification dataset that was constructed from science questions and answers. This dataset involves two-way classification (entail or non-entail). We compare with DecompAtt (Parikh et al., 2016), ESIM, DGEM (Khot et al., 2018) and CAFE.

Quora Duplicate Detection is a well-studied paraphrase identification dataset[1]. We use the splits provided by (Wang et al., 2017). The task is to determine if two questions are paraphrases of each other. This task is formulated as a binary classication problem. We compare with L.D.C (Wang et al., 2016b), BiMPM, the DecompAtt implementation by (Tomar et al., 2017) (word and char level) and DIIN.

TwitterURL (Lan et al., 2017) is another dataset for paraphrase identification. It was constructed using Tweets referring to news articles. This task is also a binary classification problem. We compare with (1) MultiP (Xu et al., 2014), a strong baseline, (2) the implementation of (He and Lin, 2016) by (Lan et al., 2017) and (3) the Subword + LM model from (Lan and Xu, 2018).

TrecQA (Wang et al., 2007) is a well-studied dataset for *answer sentence selection task* (or question-answer matching). The goal is to rank answers given a question. This task is formulated as a pointwise learning-to-rank problem. Baselines include HyperQA (Tay et al., 2017a), Ranking-based Multi-Perspective CNN (He et al., 2015) implementation by (Rao et al., 2016), BiMPM, the compare-aggregate (Wang and Jiang, 2016a) model extension by (Bian et al., 2017) (we denote this model as CA), IWAN (Shen et al., 2017) and the recent MCAN model, i.e., Multi-Cast Attention Networks (Tay et al., 2018c). A leaderboard is maintained at `https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)`.

Ubuntu (Lowe et al., 2015) is a dataset for *Utterance-Response Matching* and comprises 1-million utterance-response pairs. This dataset is based on the Ubuntu dialogue corpus. The goal is to predict the response to a message. We use the same setup as (Wu et al., 2016). Baselines include CNTN (Qiu and Huang, 2015), APLSTM (dos Santos et al., 2016), MV-LSTM (Wan et al., 2016a) and KEHNN (Wu et al., 2016). Results are reported from (Wu et al., 2016).

Metrics For all datasets, we follow the evaluation procedure from all the original papers. The metric for SNLI, SciTail and Quora is the accuracy metric. The metric for the TwitterURL dataset is the F1 score. The metric for TrecQA is the Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) metric. The metric for Ubuntu is the Recall@K for $k = 1, 2, 5$ (given 9 negative samples) and the binary classification accuracy score.

3.2 Experimental Setup

All baselines are reported from the respective papers. All models are trained with the Adam optimizer (Kingma and Ba, 2014) with learning rates tuned amongst $\{0.001, 0.0003, 0.0004\}$. Batch size is tuned amongst $\{32, 64, 128, 256\}$. The dimensions of the BiLSTM encoders are tuned amongst $\{64, 100, 200, 300\}$ and the number of hidden dimensions of the prediction layers are tuned amongst $\{100, 200, 300, 600\}$. The number of stacked recurrent layers is tuned from $[2, 5]$ and the number of aggregation BiLSTM layers is tuned amongst $\{1, 2\}$. The number of prediction layers is tuned from [1,3]. Parameters are initialized using glorot uniform (Glorot and Bengio, 2010). All unspecified activation functions are ReLU activations. Word embeddings are initialized with GloVe (Pennington et al., 2014) and fixed during training. We implement our model in Tensorflow (Abadi et al., 2015) and use the cuDNN implementation for all BiLSTM layers.

3.3 Experimental Results

Overall, our proposed CSRAN architecture achieves state-of-the-art performance on all six well-established datasets.

---

[1] `https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`

On SNLI (Table 2), CSRAN achieves the best[2] single model performance to date on the well-established dataset. This demonstrates the effectiveness of CSRAN, taking into consideration of the inherent competitiveness of this well-known benchmark. On SciTail (Table 3), CSRAN similarly achieves the best performance to date on this dataset, outperforming the existing CAFE model by +3.4% absolute accuracy.

On Quora (Table 4), CSRAN also achieves the best single model score, outperforming strong baselines such as BiMPM (+1.1%) and DIIN (+0.2%). Moreover, there is also considerable performance improvement on the TwitterURL dataset (Table 5) as CSRAN outperforms the existing state-of-the-art Subword + LM model (+8%) and Deep Pairwise Word (+9.1%).

On TrecQA (Table 6), CSRAN achieves the best performance on this dataset. CSRAN outperforms the existing state-of-the-art model, IWAN (+3.2%/ + 4.6%). CSRAN also outperforms strong competitive baselines such as BiMPM (+5.2%/+3.6%) and MPCNN (+5.3%/+5.8%). Finally, on Ubuntu (Table 7), CSRAN also outperforms many competitive models such as CNTN, APLSTM and KEHNN. Performance improvement over all metrics are $\approx 9\% - 10\%$ compared to the existing state-of-the-art.

Overall, CSRAN achieves state-of-the-art performance on six well-studied datasets. On several datasets, our achieved performance is not only the highest reported score but also outperforms the existing state-of-the-art models by a considerable margin.

Model	Acc
BiMPM (Wang et al., 2017)	87.5
ESIM (Chen et al., 2017)	88.0
DIIN (Gong et al., 2017)	88.0
DR-BiLSTM (Ghaeini et al., 2018)	88.5
CAFE (Tay et al., 2017c)	88.5
CSRAN	**88.7**

Table 2: Experimental results on single model SNLI dataset.

Model	Acc
DecompAtt (Parikh et al., 2016)	72.3
ESIM (Chen et al., 2017)	70.6
DGEM (Khot et al., 2018)	77.3
CAFE (Tay et al., 2017c)	83.3
CSRAN	**86.7**

Table 3: Experimental results on SciTail dataset.

Model	Acc
L.D.C (Wang et al., 2016b)	87.5
Word DecompAtt (Tomar et al., 2017)	87.5
BiMPM (Wang et al., 2017)	88.1
Char DecompAtt (Tomar et al., 2017)	88.4
DIIN (Gong et al., 2017)	89.0
CSRAN	**89.2**

Table 4: Experimental results on Quora Duplicate Detection dataset.

Model	F1
MultiP (Xu et al., 2014)	0.536
DeepPairwiseWord (He and Lin, 2016)	0.749
Subword + LM (Lan and Xu, 2018)	0.760
CSRAN	**0.840**

Table 5: Experimental results on TwitterURL paraphrase dataset.

Model	MAP/MRR
HyperQA (Tay et al., 2017a)	0.784/0.865
MPCNN (Rao et al., 2016)	0.801/0.877
BiMPM (Wang et al., 2017)	0.802/0.899
CA (Bian et al., 2017)	0.821/0.899
IWAN (Shen et al., 2017)	0.822/0.889
MCAN (Tay et al., 2018c)	0.838/0.904
CSRAN	**0.854/0.935**

Table 6: Experimental results on TrecQA dataset.

Model	Acc	R@1	R@2	R@5
CNTN	0.743	0.349	0.512	0.797
LSTM	0.725	0.361	0.494	0.801
APLSTM	0.758	0.381	0.545	0.801
MV-LSTM	0.767	0.410	0.565	0.800
KEHNN	0.786	0.460	0.591	0.819
MCAN	0.834	0.551	0.684	0.875
CSRAN	**0.839**	**0.556**	**0.692**	**0.880**

Table 7: Experimental results on the Ubuntu dataset for utterance-response matching. Baseline results are reported from (Wu et al., 2016).

---

[2] For fair comparison, we do not compare with (1) models that use external contextualized word embeddings, e.g., CoVe (McCann et al., 2017) / ELMo (Peters et al., 2018) / generative pretraining (Radford et al.) and (2) ensemble systems. As either (1) and/or (2) would also intuitively boost the performance of the base CSRAN model.

3.3.1 Training Efficency

With many BiLSTM layers, it is natural to be skeptical about the training efficiency of our model. However, since we use the CUDNN implementation of the BiLSTM model, the runtime is actually very manageable. On SNLI, with a batch size of 128, our model with 3 stacked recurrent layers and 2 aggregation BiLSTM layers runs at ≈17 minutes per epoch and converges in less than 20 epochs. On SciTail, our model runs at ≈ 2 minutes per epoch with a batch size of 32. This is benchmarked on a TitanXP GPU. While our model is targetted at performance and not efficiency, this section serves as a reassurance that our model is not computationally prohibitive.

3.4 Ablation Study

In order to study the effectiveness of the key components in our proposed architecture, we conduct an extensive ablation study. Table 8 reports the results on several ablation baselines. There are three key ablation baselines as follows: (1) we removed MAR from the stacked recurrent network, (2) we removed CSRA from the network and finally (3) we removed both MAR and CSRA from the network. All ablation baselines reported are stacked with 3 layers. Firstly, we observe that both MAR and CSRA are critical components in our model, i.e., removing any of them would result in a drop in performance. Secondly, we observe that the relative utility of CSRA and MAR depends on the dataset. Removing MAR significantly reduces performance on SciTail and TrecQA. On the other hand, removing CSRA degrades the performance more than MAR on SNLI. Finally, it is good to note that, while performance degradation on SNLI development set may not seem significant, the *w/o MAR and CSRA* ablation performance baseline achieved only 87.7% accuracy on the test set, compared to 88.7% of the original model. This is equivalent to dropping from state-of-the-art to the 5th ranked model. Overall, we are able to conclude that the CSRA and MAR make meaningful improvements to our model architecture.

Ablation	SNLI	SciTail	TrecQA
Original	88.6	88.0	0.86/0.90
w/o MAR	88.4	82.5	0.79/0.85
w/o CSRA	88.1	86.2	0.84/0.89
w/o MAR/CSRA	88.0	83.0	0.79/0.84

Table 8: Ablation study (development score) of our key model components on three datasets.

3.5 Effect of Stack Depth

In this section, our goals are twofold - (1) studying the effect of stack depth on model performance and (2) determining if the proposed CSRAN model indeed helps with enabling deeper stack depths. In order to do so, we compute the development set performance of two models. The first is the full CSRAN architecture and the second is a baseline stacked model architecture. Note that the bidirectional alignment layer and remainder of the model architecture (highway layers, etc.) remain completely identical to CSRAN to make this study as fair as possible.

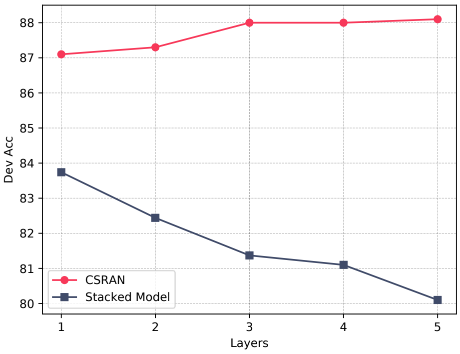

Figure 2: Relative effect of stack depth on CSRAN and the baseline Stacked Model on SciTail dataset.

Figure 2 illustrates the model performance with varying stack depth. As expected, the performance of the stacked model declines when increasing the stack depth. On the other hand, the performance of CSRAN improves by adding additional layers. The largest gain is when jumping from 2 layers to 3 layers. The subsequent performance improvement from 3-5 layers is marginal. From this study, the takeaway is that standard stacked architectures are insufficient. As such, our proposed CSRA mechanism can aid in enabling deeper models which can result in stronger model performance[3].

Next, we study the general effect of stack depth (number of layers) on model performance. Figure 3 reports the model performance (dev accuracy) of our CSRAN architecture on Quora and

---

[3] The best result on Scitail was obtained with 5 layers. Moreover, the difference in test performance between stacked and single-layered model was considerably high (+2.5%) even though dev performance increased by +1%.

SNLI datasets. We observe that a stacked architecture with 3 layers is significantly better than a single-layered architecture. The optimal development score is 3-4 layers for SNLI and 3 layers for Quora. However, we observe the performance of Quora declines after 3 layers (notably it is still higher than an unstacked model). However, the performance on SNLI remains relatively stable.

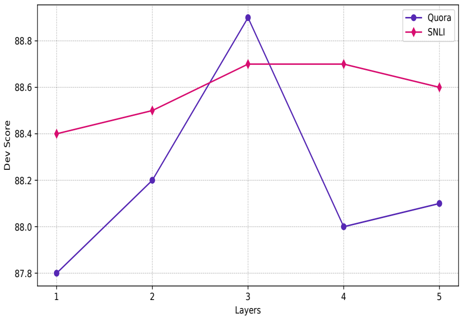

Figure 3: Effect of stack depth on CSRAN performance on Quora and SNLI datasets.

4 Related Work

Learning to matching text sequences is a core and fundamental research problem in NLP and Information Retrieval. A wide range of NLP applications fall under this paradigm such as natural language inference (Bowman et al., 2015; Khot et al., 2018), paraphrase identification (Lan and Xu, 2018), question answering (Severyn and Moschitti, 2015), document search (Shen et al., 2014; Hui et al., 2017), social media search (Rao et al., 2018) and entity linking (Phan et al., 2017). As such, universal text matching algorithms are generally very attractive, in lieu of the prospects of potentially benefitting an entire suite of NLP applications.

Neural networks have been the prominent choice for text matching. Earlier works are mainly concerned with learning a matching function between RNN/CNN encoded representations (Severyn and Moschitti, 2015; Yu et al., 2014; Qiu and Huang, 2015; Tay et al., 2017b, 2018b). Models such as Recursive Neural Networks have also been explored (Wan et al., 2016b). Subsequently, attention-based models were adopted (Rocktäschel et al., 2015; Wang et al., 2016a; Parikh et al., 2016), demonstrating superior performance relative to their non-attentive counterparts.

Today, the dominant state-of-the-art approaches for text matching are mostly based on neural models configured with bidirectional attention layers (Shen et al., 2017; Tay et al., 2017c). Bidirectional attention comes in various flavours which can be known as soft alignment (Shen et al., 2017; Chen et al., 2017), decomposable attention (Parikh et al., 2016), attentive pooling (dos Santos et al., 2016) and even complex-valued attention (Tay et al., 2018a). The key idea is to jointly soft align text sequences such that they can be compared at the index level. To this end, various comparison functions have been utilized, ranging from feed-forward neural networks (Parikh et al., 2016) to factorization machines (Tay et al., 2017c). Notably, these attention (and bi-attention) mechanisms are also widely adopted (or originated) from many related sub-fields of NLP such as machine translation (Bahdanau et al., 2014) and reading comprehension (Xiong et al., 2016; Seo et al., 2016; Wang and Jiang, 2016b).

Many text matching neural models are heavily grounded in the compare-aggregate architecture (Wang and Jiang, 2016a). In these models, matching and comparisons occur between text sequences, aggregating features for making the final prediction. Recent state-of-the-art models such as BiMPM (Wang et al., 2017) and DIIN (Gong et al., 2017) are representative of such architectural paradigm, utilizing an attention-based matching scheme and then a CNN or LSTM-based feature aggregator. Earlier works (Wan et al., 2016a; He et al., 2015; He and Lin, 2016) exploit a similar paradigm, albeit without the usage of attention.

Across many NLP and machine learning applications, utilizing stacked architectures is a common way to enhance representation capability of the encoder (Sutskever et al., 2014; Graves et al., 2013; Zhang et al., 2016; Nie and Bansal, 2017), leading to performance improvement. Deep networks suffer from inherent difficulty in feature propagation and/or vanishing/exploding gradients. As a result, residual strategies have often been employed (He et al., 2016; Srivastava et al., 2015; Huang et al., 2017). However, to the best of our knowledge, this work presents a new way of residual connections, leveraging on the fact that pairwise formulation of the text matching task.

5 Conclusion

We proposed a deep stacked recurrent architecture for general-purpose text sequence matching. We proposed a new co-stack residual affinity mechanism for matching sequence pairs, leveraging multi-hierarchical information for learning bidirectional alignments. Our proposed CSRAN model achieves state-of-the-art performance across six well-studied benchmark datasets and four different problem domains.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1987–1990, New York, NY, USA. ACM.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1657–1668.

Reza Ghaeini, Sadid A Hasan, Vivek Datla, Joey Liu, Kathy Lee, Ashequl Qadir, Yuan Ling, Aaditya Prakash, Xiaoli Z Fern, and Oladimeji Farri. 2018. Dr-bilstm: Dependent reading bidirectional lstm for natural language inference. *arXiv preprint arXiv:1802.05577*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *CoRR*, abs/1709.04348.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Hua He, Kevin Gimpel, and Jimmy J. Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1576–1586.

Hua He and Jimmy J. Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 937–948.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. Pacrr: A position-aware neural ir model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. *arXiv preprint arXiv:1708.00391*.

Wuwei Lan and Wei Xu. 2018. The importance of subword embeddings in sentence pair modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 41–45.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Minh C Phan, Aixin Sun, Yi Tay, Jialong Han, and Chenliang Li. 2017. Neupl: Attention-based semantic matching and pair-linking for entity disambiguation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1667–1676. ACM.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1305–1311.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Jinfeng Rao, Hua He, and Jimmy J. Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 1913–1916.

Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. 2018. Multi-perspective relevance matching with hierarchical convnets for social media search. *arXiv preprint arXiv:1805.08159*.

Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 373–382.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017a. Enabling efficient question answer retrieval via hyperbolic neural networks. *CoRR*, abs/1707.07847.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018a. Hermitian co-attention networks for text matching in asymmetrical domains. In *IJCAI*, pages 4425–4431.

Yi Tay, Minh C. Phan, Anh Tuan Luu, and Siu Cheung Hui. 2017b. Learning to rank question answer pairs with holographic dual LSTM architecture. In *Proceedings of the 40th International ACM SIGIR*

Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017, pages 695–704.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017c. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018b. Cross temporal recurrent networks for ranking question answer pairs. In *Proceedings of AAAI 2018*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018c. Multi-cast attention networks. In *Proceedings of KDD 2018*, KDD '18, pages 2299–2308, New York, NY, USA. ACM.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural paraphrase identification of questions with noisy pretraining. *arXiv preprint arXiv:1704.04565*.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016a. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2835–2841.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016b. Match-srnn: Modeling the recursive matching structure with spatial rnn. *arXiv preprint arXiv:1604.04378*.

Bingning Wang, Kang Liu, and Jun Zhao. 2016a. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? A quasi-synchronous grammar for QA. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 22–32.

Shuohang Wang and Jing Jiang. 2016a. A compare-aggregate model for matching text sequences. *CoRR*, abs/1611.01747.

Shuohang Wang and Jing Jiang. 2016b. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*.

Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2016. Knowledge enhanced hybrid neural network for text matching. *arXiv preprint arXiv:1611.04684*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. 2016. Highway long short-term memory rnns for distant speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5755–5759. IEEE.

Spherical Latent Spaces for Stable Variational Autoencoders

Jiacheng Xu and **Greg Durrett**
Department of Computer Science
The University of Texas at Austin
`{jcxu,gdurrett}@cs.utexas.edu`

Abstract

A hallmark of variational autoencoders (VAEs) for text processing is their combination of powerful encoder-decoder models, such as LSTMs, with simple latent distributions, typically multivariate Gaussians. These models pose a difficult optimization problem: there is an especially bad local optimum where the variational posterior always equals the prior and the model does not use the latent variable at all, a kind of "collapse" which is encouraged by the KL divergence term of the objective. In this work, we experiment with another choice of latent distribution, namely the von Mises-Fisher (vMF) distribution, which places mass on the surface of the unit hypersphere. With this choice of prior and posterior, the KL divergence term now only depends on the variance of the vMF distribution, giving us the ability to treat it as a fixed hyperparameter. We show that doing so not only averts the KL collapse, but consistently gives better likelihoods than Gaussians across a range of modeling conditions, including recurrent language modeling and bag-of-words document modeling. An analysis of the properties of our vMF representations shows that they learn richer and more nuanced structures in their latent representations than their Gaussian counterparts.[1]

1 Introduction

Recent work has established the effectiveness of deep generative models for a range of tasks in NLP, including text generation (Hu et al., 2017; Yu et al., 2017), machine translation (Zhang et al., 2016), and style transfer (Shen et al., 2017; Zhao et al., 2017a). Variational autoencoders, which have been explored in past work for text modeling (Miao et al., 2016; Bowman et al., 2016), posit a continuous latent variable which is used to capture latent structure in the data. Typical VAE implementations assume the prior of this latent space is a multivariate Gaussian; during training, a Kullback-Leibler (KL) divergence term in loss function encourages the variational posterior to approximate the prior. One major limitation of this approach observed by past work is that the KL term may encourage the posterior distribution of the latent variable to "collapse" to the prior, effectively rendering the latent structure unused (Bowman et al., 2016; Chen et al., 2016).

In this paper, we propose to use the von Mises-Fisher (vMF) distribution rather than Gaussian for our latent variable. vMF places a distribution over the unit hypersphere governed by a mean parameter μ and a concentration parameter κ. Our prior is a uniform distribution over the unit hypersphere ($\kappa = 0$) and our family of posterior distributions treats κ as a *fixed* model hyperparameter. Since the KL divergence only depends on κ, we can structurally prevent the KL collapse and make our model's optimization problem easier. We show that this approach is actually more robust than trying to flexibly learn κ, and a wide range of settings for fixed κ lead to good performance. Our model systematically achieves better log likelihoods than analogous Gaussian models while having higher KL divergence values, showing that it more successfully makes use of the latent variables at the end of training.

Past work has suggested several other techniques for dealing with the KL collapse in the Gaussian case. Annealing the weight of KL term (Bowman et al., 2016) still leaves us with brittleness in the optimization process, as we show in Section 2. Other prior work (Yang et al., 2017; Semeniuta et al., 2017) focuses on using CNNs rather than RNNs as the decoder in order to weaken the model and encourage the use of the

---

[1] The code and dataset are available at: `https://github.com/jiacheng-xu/vmf_vae_nlp`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4503–4513
Brussels, Belgium, October 31 - November 4, 2018.

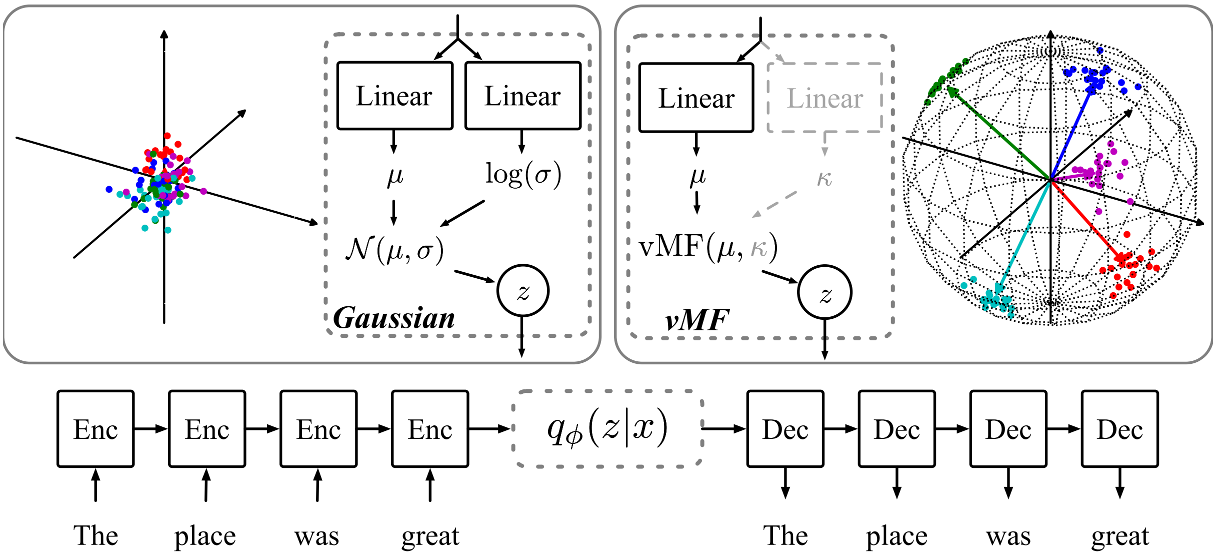

Figure 1: The Neural Variational RNN (NVRNN) language model based on a Gaussian prior (left) and a vMF prior (right). The encoder model first computes the parameters for the variational approximation $q_\phi(z|x)$ (see dotted box); we then sample z and generate the word sequence x given z. We show samples from $\mathcal{N}(0, I)$ and vMF($\cdot$,$\kappa = 100$); the latter samples lie on the surface of the unit sphere. While κ can be predicted from the encoder network, we find experimentally that fixing it leads to more stable optimization and better performance.

latent code, but the gains are limited and changing the decoder in this way requires ad hoc model engineering and careful tuning of various decoder capacity parameters. Our method is orthogonal to the choice of the decoder and can be combined with any of these approaches. Using vMF distributions in VAEs also leaves us the flexibility to modify the prior in other ways, such as using a product distribution with a uniform (Guu et al., 2018) or piecewise constant term (Serban et al., 2017a).

We evaluate our approach in two generative modeling paradigms. For both RNN language modeling and bag-of-words document modeling, we find that vMF is more robust than a Gaussian prior, and our model learns to rely more on the latent variable while achieving better held-out data likelihoods. To better understand the contrast between these models, we design and conduct a series of experiments to understand the properties of the Gaussian and vMF latent code spaces, which make different structural assumptions. Unsurprisingly, these latent code distributions capture much of the same information as in a bag of words, but we show that vMF can more readily go beyond this, capturing ordering information more effectively than a Gaussian code.

2 Variational Autoencoders for Text

Bowman et al. (2016) propose a variational autoencoder model for generative text modeling inspired by Kingma and Welling (2013). Instead of modeling $p(x)$ directly as in vanilla language models, VAEs introduce a continuous latent variable z and take the form $p(z)p(x|z)$. To train a VAE, we optimize the marginal likelihood $p(x) = \int p_\theta(z)p(x|z)dz$. The marginal log likelihood can be written as:

$$\log p_\theta(x) = \mathrm{KL}(q_\phi(z|x)||p_\theta(z|x)) + \mathcal{L}(\theta, \phi; x)$$

$$\begin{aligned}\mathcal{L}(\theta, \phi; x) = &-\mathrm{KL}(q_\phi(z|x)||p_\theta(z)) \\ &+ \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) \quad (1)\end{aligned}$$

$q_\phi(z|x)$, a variational approximation to the posterior $p(z|x)$, can be variously interpreted as a *recognition model* or encoder, parameterized by a neural network to encode the sentence x into a dense *code* z. $\mathcal{L}(\theta, \phi; x)$ is often called the evidence lower bound (ELBO). The first term of ELBO is the KL divergence of the approximate posterior from prior and the second term is an expected reconstruction error.

Since KL divergence is always non-negative, we can use $\mathcal{L}(\theta, \phi; x)$ as a lower bound of marginal likelihood $\log p_\theta(x)$. We optimize $\mathcal{L}(\theta, \phi; x)$, jointly learning the recognition model parameters ϕ and generative model parameters θ.

As the choice of prior $p(z)$, most previous work uses a centered multivariate Gaussian $p_\theta(z) =$

$\mathcal{N}(z; 0, I)$. Since Gaussians are a location-scale family of distributions, using them for both the prior and posterior allows us to apply the reparameterization trick and differentiate through the sampling stage $z \sim \mathbb{E}_{q_\phi(z|x)}$ when optimizing ELBO in practice (Kingma and Welling, 2013).

2.1 Case Study: NVRNN

A Neural Variational RNN (NVRNN) for language modeling is described in Bowman et al. (2016) and depicted in Figure 1. The goal of the NVRNN model is to extract a high level representation of a sentence into z and reconstruct the sentence with a neural language model.

We denote a sequence of words as $x = \{x_1, x_2, \cdots, x_n\}$. Unlike in vanilla language modeling, an NVRNN conditions on the latent variable z at each step of the generation $p_\theta(x|z) = p_\theta(x_1|z) \prod_{i=1}^{n} p(x_i|x_1, \ldots, x_{i-1}, z)$. This probability distribution is modeled using a recurrent model like an LSTM (Hochreiter and Schmidhuber, 1997) as illustrated in Figure 1. There is nothing unique about this choice; other recurrent sequence models like a CNN or a Transformer (Vaswani et al., 2017) could be used.

2.2 Posterior Collapse

When training a VAE, we update θ and ϕ simultaneously. Optimizing Eq. 1 gives two gradient terms: an update from the reconstruction loss (likelihood of the correct labels) and an update from the KL divergence. While the reconstruction loss term encourages the z to convey useful information to this model, the KL term consistently tries to regularize $q(z|x)$ towards the prior on every gradient update. This may trap the model in a bad local optimum where $q_\phi(z|x) = p_\theta(z)$ for all x: in this case, z is simply a noise source, which is useless to the model, so the model has learned to ignore it and will not make large enough gradient updates to break $q(z|x)$ out of this optimum.

Bowman et al. (2016) termed this issue *KL collapse* and proposed an annealing schedule to handle it, where the weight of the KL term is increased over the course of training.[2] In this way, the model initially learns to use the latent code but is then regularized towards the prior as training progresses. However, this trick is not sufficient to avert KL collapse in all scenarios, particularly when strong decoders are used and z has a minor impact on $p_\theta(x|z)$.

	No annealing		Sigmoid annealing	
	3-layer	1-layer	3-layer	1-layer
KL	0.00	3.37	1.05	**6.52**
NLL	135	129	132	**125**

Table 1: Development set KL and NLL values for two NVRNN models trained on the Penn Treebank with and without the annealing technique of Bowman et al. (2016). The higher-capacity 3-layer model collapses when no annealing is used, and while annealing works to improve performance, it still does not perform as well as the 1-layer variant. By contrast, a variant of the 1-layer model with vMF gives an NLL value of 117 and a KL of 18.6, a stronger result relying more heavily on the latent variable.

Table 1 shows experiments in a similar setup to that of Bowman et al. (2016). We train an NVRNN model on the Penn Treebank with four different hyperparameter settings. We either use a 3-layer LSTM encoder or a 1-layer LSTM and use or do not use a sigmoid annealing schedule (increase the KL weight from 0 to 1 over the first 20 epochs). We observe the best performance using the 1-layer model with annealing. One might conclude from this table that the annealing trick has worked since both models achieve better performance when annealing is used. But in fact, a vMF-based model can do better than either (NLL of 117), and moreover, we have no way of knowing that a better annealing scheme might not achieve even higher performance after training. Furthermore, the higher-capacity 3-layer model can theoretically do anything the 1-layer model can, so its lower performance indicates that our training is derailed either by overfitting or getting stuck in a local optimum where the latent variable is unused.[3]

Getting the best performance out of a VAE is, therefore, a challenging problem that requires careful tuning of the objective function and optimization procedure (Bowman et al., 2016; Zhao et al., 2017b; Higgins et al., 2017). Beyond the well-documented problem of KL collapse, an optimizer may simply get stuck in a local optimum during training and as a result, fail to find a model that most effectively exploits the latent variable.

The solution we advocate for in this paper is

---

[2] Reweighting the KL term is also used in methods like β-VAE (Higgins et al., 2017) and InfoVAE (Zhao et al., 2017b).

[3] In our experiments, we found significant variance in collapse frequency due to other hyperparameters including whether the encoder is a unidirectional or bidirectional LSTM.

to change the distribution for the latent space and simplify the optimization problem. In the next section, we describe the von Mises-Fisher distribution and its use in VAE, where it forces the model to put the latent representations on the surface of the unit hypersphere rather than squeezing everything to the origin. Critically, this distribution lets us fix the value of the KL term by fixing the distribution's concentration parameter κ; this averts the KL collapse and leads to good model performance across two generative modeling paradigms.

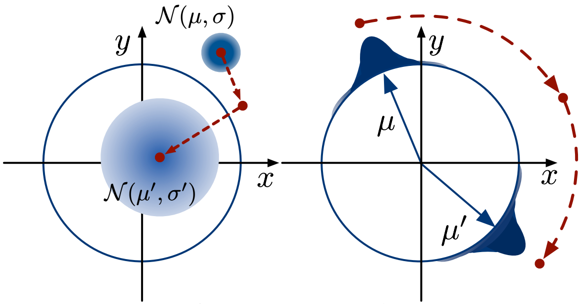

(a) Gaussian (b) vMF

Figure 2: Visualization of optimization of how q varies over time for a single example during learning. In the Gaussian case, the KL term tends to pull the model towards the prior (moving from μ, σ to μ', σ'), whereas in the vMF case there is no such pressure towards a single distribution.

3 von Mises-Fisher VAE

The von Mises-Fisher distribution is a distribution on the $(d-1)$-dimensional sphere in $\mathbb{R}^d$. The vMF distribution is defined by a direction vector μ with $||\mu||= 1$ and a concentration parameter $\kappa \geq 0$. The PDF of the vMF distribution for the d-dimensional unit vector x is defined as:

$$f_d(x; \mu, \kappa) = C_d(\kappa) \exp(\kappa \mu^T x) \tag{2}$$

$$C_d(\kappa) = \frac{\kappa^{d/2-1}}{(2\pi)^{d/2} I_{d/2-1}(\kappa)} \tag{3}$$

where I_v stands for the modified Bessel function of the first kind at order v.

Figure 1 shows samples from vMF distributions with various μ vectors (arrows), $d = 3$, and $\kappa = 100$. This is a high κ value, leading to samples that are tightly clustered around μ, which is the mean and mode of the distribution. When $\kappa = 0$, the distribution degenerates to a uniform distribution over the hypersphere independent of μ.

Past work has used vMF as an emission distribution in unsupervised clustering models (Banerjee et al., 2005), VAE for other domains (Davidson et al., 2018; Hasnat et al., 2017), and a generative editing model for text (Guu et al., 2018). We focus specifically on the empirical properties of vMF for text modeling and conduct a systematic examination of how this prior affects VAE models compared to using a Gaussian.

VAE with vMF We will use vMF as both our prior and variational posterior in our VAE models. Otherwise, the setup for our VAE remains the same as in the Gaussian case established in Section 2. Our prior is the uniform distribution vMF$(\cdot, \kappa = 0)$. Since true posterior $p_\theta(z|x)$ is intractable, we will approximate it with a variational posterior $q_\phi(z|x) = \text{vMF}(z; \mu, \kappa)$ where the mean direction μ is the output of encoding neural networks (Figure 1, right side) and κ is treated as a constant.

Before we can implement a VAE, we need to derive an expression for KL divergence in order to optimize ELBO (Equation 1) and give a sampling algorithm that admits the reparameterization trick (Kingma and Welling, 2013).

KL divergence With vMF$(\cdot, 0)$ as our prior, the KL divergence is:[4]

$$\begin{aligned} \text{KL}(\text{vMF}(\mu, \kappa)||\text{vMF}(\cdot, 0)) &= \kappa \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)} \\ + \left(\frac{d}{2} - 1\right) \log \kappa - \frac{d}{2} \log(2\pi) &- \log I_{d/2-1}(\kappa) \\ + \frac{d}{2} \log \pi + \log 2 &- \log \Gamma \left(\frac{d}{2}\right) \end{aligned}$$

Critically, this only depends on κ, not on μ. κ will be treated as a fixed hyperparameter, so this term will be constant for our model; KL collapse will therefore be rendered impossible.

Figure 2 shows a visualization of the learning trajectories of Gaussian and vMF VAE. For the Gaussian VAE, the KL divergence in the objective function tends to pull the posterior towards the prior centered at the origin and, therefore, make the optimization difficult as mentioned before. For the vMF VAE, given fixed κ, there is no such vacuous state and μ can vary freely.

---

[4] Our KL divergence agrees with that of Davidson et al. (2018) (see their appendix for a derivation), and we have verified it empirically. The equation in Guu et al. (2018) gives slightly different KL values, though differences are small (<5%) for most κ and dimension values we encounter.

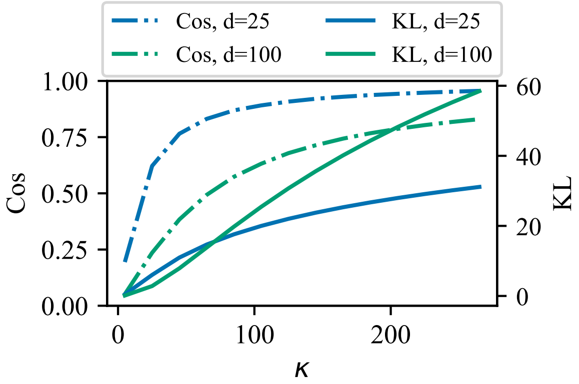

Figure 3: Visualization of the interaction between κ, KL, and dimensionality in vMF. Cos represents the cosine similarity between μ and samples from $\text{vMF}_d(\mu, \kappa)$ which reflects how disperse the distribution is. KL is defined as KL with a uniform vMF prior, $\text{KL}(\text{vMF}_d(\mu, \kappa) || \text{vMF}(\cdot, 0))$. Higher κ values yield higher cosine similarities, but also higher KL costs.

Figure 3 shows the KL value and concentration of $\text{vMF}(\mu, \kappa)$ for two different dimensionalities. KL increases monotonically with κ, as does concentration measured by cosine similarity. To get a fixed cosine dispersion as dimensionality increases, higher κ values are needed, resulting in higher KL values.

Sampling from vMF Following the implementation of Guu et al. (2018), we use the rejection sampling scheme of Wood (1994) to sample a "change magnitude" w. Our sample is then given by $z = w\mu + v\sqrt{1 - w^2}$, where v is a randomly sampled unit vector tangent to the hypersphere at μ. Neither v nor w depends on μ, so we can now take gradients of z with respect to μ as required.

4 Experiments on Language Modeling

We first evaluate our vMF approach in the NVRNN setting. We will return to this model and analyze its properties further in Sections 6 and 7 after showing experiments on document modeling.

Dataset For NVRNN, we use the Penn Treebank (Marcus et al., 1993), also used in Bowman et al. (2016), and Yelp 2013 (Xu et al., 2016). Examples in the Yelp dataset are much longer and more diverse than those from PTB, requiring more understanding of high-level semantics to generate a coherent sequence. Yelp has a long tail of very long reviews, so we truncate the examples to a maximum length of 50 words; this still gives an average length over twice as long as in the PTB setting.

Name	Train	Dev	Test	Len	Vocab
PTB	42068	3370	3761	21.1	10K
Yelp	62522	7773	8671	49.5	15K
20NG	11268	-	7505	96.1	2K
RC	794414	-	10000	116.8	10K

Table 2: Statistics of the datasets used in our experiments. Len stands for the average length of an example. Vocab is the vocabulary size; these follow prior work.

Statistics about all datasets used in this paper are shown in Table 2.

Settings We evaluate our NVRNN as in Bowman et al. (2016) and explore two different settings. In the **Standard** setting, the input to the RNN at each time step is the concatenation of the latent code z and the ground truth word from the last time step, while the **Inputless** setting does not use the prior word. The more powerful decoder of the Standard setting makes the latent representations inherently less useful. In the Inputless setting, the decoder needs to predict the whole sequence with only the help of given latent code. In this case, a high-quality representation of the sentence is badly needed and the model is driven to learn it.

Our implementation of VAE uses a one layer unidirectional LSTM as both encoder and decoder. We use an embedding size of 100 and hidden units of size 400 in the LSTM. The dimension of the latent code is chosen from $\{25, 50, 100\}$ by tuning on the development set. We use SGD to optimize all models with decayed learning rate and gradient clipping. For Yelp, the sentiment bit, which ranges from 1 to 5, is also embedded into a 50 dimension vector and input for every time step of the decoding phase.

Results Experimental results of the NVRNN are shown in Table 3. We report negative log likelihood (NLL)[5] and perplexity (PPL) on the test set. We follow the implementation reported in Bowman et al. (2016) where the KL term weight is annealed for the Gaussian VAE; vMF VAE works well without weight annealing. The vMF distribution gives a performance boost in all datasets in both the Standard and Inputless settings. Even in the Standard setting, our model is able to successfully use nonzero KL values to achieve better

---

[5] Reported values are actually a lower bound on the true NLL, computed from ELBO by sampling z.

Model	PTB				Yelp			
	Standard		Inputless		Standard		Inputless	
	NLL	PPL	NLL	PPL	NLL	PPL	NLL	PPL
RNNLM (2016)	100 (–)	116	135 (–)	>600	–	–	–	–
G-VAE (2016)	101 (2)	119	125 (15)	380	–	–	–	–
RNNLM (Ours)	100 (–)	114	134 (–)	596	199 (–)	55	300 (–)	432
G-VAE (Ours)	99 (4.4)	109	125 (6.3)	379	199 (0.5)	55	274 (13.4)	256
vMF-VAE (Ours)	**96 (5.7)**	**98**	**117 (18.6)**	**262**	**198 (6.4)**	**54**	**242 (48.5)**	**134**

Table 3: Experimental results of NVRNN on the test sets of PTB and Yelp. The upper RNNLM and G-VAE shows the result from Bowman et al. (2016). KL divergence is shown in the parenthesis, along with total NLL. Best results are in bold. vMF consistently uses higher KL term weights but achieves comparable or better NLL and perplexity values across all four settings.

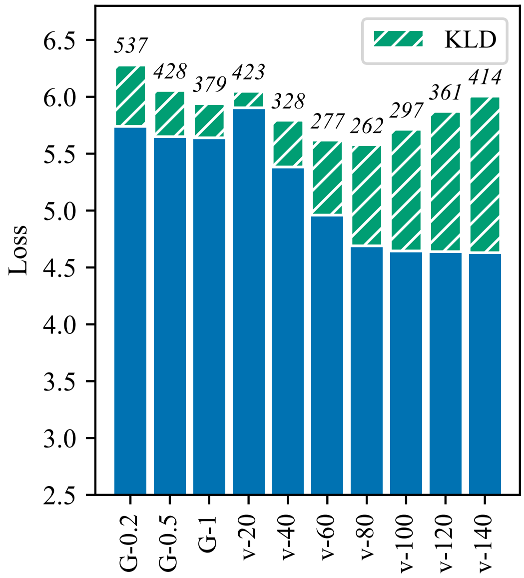

Figure 4: Comparison of Gaussian- and vMF-NVRNN with different hyper-parameters. All models are trained on PTB in the Inputless setting where the latent dimension is 50. G-α indicates Gaussian VAE with KL annealed by the given constant α, and V-κ indicates VAE with κ set to the given value. The green bar reflects the amount of KL loss while the total height reflects the whole objective. *Numbers* above bars are perplexity. vMF is more highly tunable and also achieves stronger results across a wide range of κ values.

perplexities, and even when KL collapse does not appear to be the case (e.g., G-VAE on the PTB-Standard setting), a Gaussian family of distributions results in lower KLs and worse log likelihoods, possibly due to optimization challenges. In the Inputless setting, we see large gains: vMF VAE reduces PPL from 379 to 262 in PTB, and from 256 to 134 in Yelp compared to Gaussian VAE.

Trade-off Comparison Besides the overall perplexity, we are also interested in the trade-off between reconstruction loss and KL, and the contribution of KL to the whole objective. Figure 4 shows the ability of our model to explicitly control the balance between the KL and the reconstruction term. First, we "permanently" anneal the Gaussian VAE by setting the weight of the KL term to a constant smaller than 1 (0.2 and 0.5 in our case). We find that this trick does mitigate the KL collapse, but the overall performance is worse. Therefore, this is not only a numerical game about the KL vs. NLL trade-off but a deeper challenge of how to structure models to learn effective latent representations.

For vMF VAE, when we gradually increase the value of κ, the concentration of the distribution around the mean direction μ is higher and samples from vMF are closer to μ. The model achieves the best perplexity when $\kappa = 80$. The reconstruction error is bounded around 4.5 due to the difficulty of the task and limited capacity of LSTM decoder. While κ is a hyperparameter that needs to be tuned, the model is overall not very sensitive to it, and we show in Section 7 that reasonable κ values transfer across similar tasks.

5 Experiments on Document Modeling

We also investigate how vMF VAE performs in a different setting, one less plagued by the KL collapse issue. Specifically, the Neural Variational Document Model (NVDM), proposed by Miao et al. (2016), is a VAE-based unsupervised document model. This model follows the VAE framework introduced in Section 2. Our document representation is an indicator vector x of word presence or absence in the document. Since this is a fixed-size representation, we use 2-layer MLPs with 400 hidden units for both the encoder $q(z|x)$ and decoder $p(x|z)$; the decoder places a simple

Model	Dim	20NG	RCV1
fDARN (2014)	50	917	724
	200	-	598
G-NVDM (2016)	50	836	563
	200	852	550
v-NVDM (Ours)	25	**793**	558
	50	830	**529**
	200	851	609

Table 4: Test set perplexities for the document modeling task. Feedforward Deep Auto Regressive Neural Network (fDARN) is implemented by Mnih and Gregor (2014). Gaussian-based NVDM (G-NVDM) is proposed in Miao et al. (2016). Dim indicates the dimension of the latent code. Our v-NVDM model outperforms past models by a substantial margin.

multinomial distribution over words in the vocabulary, and the probability of a document is the product of the probabilities of its words.

Dataset For NVDM, we use two standard news corpus, 20 News Groups (20NG) and the Reuters RCV1-v2, which were used in Miao et al. (2016).[6]

Results Experimental results[7] are shown in Table 4. In contrast with NVRNN, the NVDM fully relies on the power of latent code to predict the word distribution, so we never observe a KL collapse, yet vMF still achieves better performance than Gaussian. As shown in Figure 3, in order to keep the same amount of dispersion in samples from the variational posterior, larger latent dimensions need larger κ values and correspondingly larger KL term values. For 20NG, which is much smaller than RCV1, smaller dimensions therefore give better performance. For both datasets, the settings of $\kappa = 100, \text{dim} = 25$ and $\kappa = 150, \text{dim} \in \{50, 200\}$ work well.

6 What do our VAEs encode?

We design more probing tasks to demonstrate what is encoded in latent representations induced by vMF VAE. One additional model variant we explore here is the NVRNN-BoW model. This is a variant of NVRNN where the decoder additionally conditions on the vector $BoW = \frac{1}{n}\sum_{i=1}^{n} e(x_i)$, the average word embedding value of the sentence x. While an artificial setting, this lets us see how effectively the latent code can capture information other than simple word choice by making a form of this information independently available. Table 5 shows results in this setting, where once again we see the KL collapse problem for the Gaussian models and better performance from vMF on perplexity in both the Standard and Inputless settings.

$P(x\|z, \text{BoW})$	Standard NLL	Standard PPL	Inputless NLL	Inputless PPL
RNNLM	79 (–)	43	106 (–)	152
G-VAE	79 (0.0)	43	106 (0.4)	153
v-VAE	**73 (0.2)**	**33**	**93 (11.4)**	**82**

Table 5: Experimental results of NVRNN-BoW on PTB; i.e., the decoder also conditions on a bag of words representation of the sentence to generate. In this case, the Gaussian models exhibit KL collapse but vMF can still learn effectively.

Is the latent code more than a bag of words? For all of these models, one hypothesis is that the encoder may be learning to memorize the bag of words and then preferentially generate words in that bag from the decoder. To verify this, we investigate whether the BoW representation and the learned latent code can be reconstructed from each other. Specifically, given a sentence x we can compute BoW as defined above and $\mu = \text{enc}(x)$, the latent encoding of x as represented by the mean vector output by the encoder. We can use a simple multilayer perceptron to to try to map from the bag of words to the latent code: $\hat{\mu} = MLP(BoW)$, then learn the parameters of the MLP by minimizing $\|\hat{\mu} - \mu\|^2$ on a sample. The same process can be used to learn a mapping from μ back to the bag of words.

Table 6 shows averaged cosine similarities of our reconstructions under both Gaussian and vMF models. For vMF, μ can reconstruct the bag-of-words more accurately than the bag-of-words can reconstruct μ, indicating that the latent code in vMF captures more information beyond the bag of words.

We repeat this experiment in a separate NVRNN model where the decoder can explicitly condition on the BoW vector described above. The results are shown in the right column of Table 6. Our model, v-VAE, achieves a lower cosine

---

[6] The preprocessed version can be downloaded from `https://github.com/ysmiao/nvdm`

[7] We do not compare to results from Serban et al. (2017a). Compared to our current results, that work reports very strong performance on 20NG and very weak performance on RCV1; we believe they either used different preprocessing or made a mistake in reporting the results, but could not confirm this with the authors.

Model	NVRNN		NVRNN-BoW
Setting	$\mu \rightarrow$ BoW	BoW $\rightarrow \mu$	$\mu \rightarrow$ BoW
G-VAE	0.74	0.74	0.32
v-VAE	0.77	0.57	0.23

Table 6: Average cosine similarity when trying to reconstruct the latent code μ from the bag of words and vice versa. In vMF, the latent code contains more information beyond the bag of words, as shown by the lower cosine similarity when predicting BoW $\rightarrow$ μ (0.57). When the latent code is learned in a model conditioned on the bag of words (right column), it predicts the bag of words much less well, indicating that the model successfully learns orthogonal information.

similarity than G-VAE (0.23 vs. 0.32), indicating that it capturing less redundant information and using the latent space to more efficiently model other properties of the data.

Sensitivity to word order Table 6 shows that NVRNN with vMF encodes information beyond the bag of words; a natural hypothesis is that it is encoding word order. We can more directly investigate this in the context of both NVRNN and NVRNN-BoW settings. Inspired by Zhao et al. (2017a), we propose an experiment probing the sensitivity to randomly swapping adjacent pairs of words for the encoding in the Inputless setting on PTB. We vary the probability of swapping each word pair and see how the latent code changes as the number of swaps increases. Ideally, our models should capture ordering information and therefore be sensitive to this change.

Figure 5 shows the results. v-VAE's representations are more sensitive than those of the G-VAE: they change faster as swaps become more likely.[8] In the NVRNN-BoW setting, we see that the models are even more sensitive. vMF enables us to more easily learn this kind of desirable information in our sentence encodings.

7 Controlling Variance with κ

A core aspect of our approach so far has been treating κ as a fixed hyperparameter. Fixing κ is beneficial from an optimization standpoint: it makes it more difficult for the model to get stuck in local optima. But it also reduces the model's flexibility, since we can no longer predict per-example κ values, and it introduces another parameter that the system designer must tune.

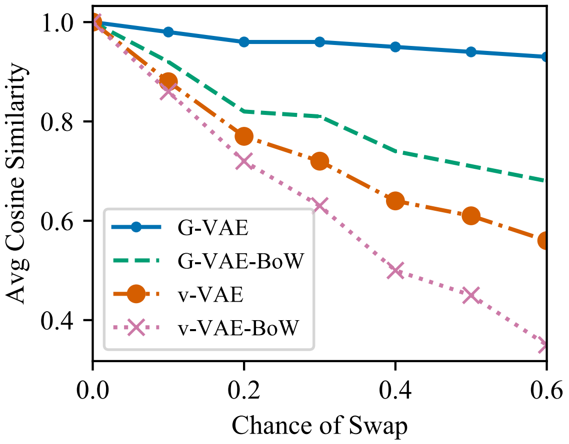

Figure 5: Sensitivity of latent codes to swapping adjacent words of encoding sequence. Cosine similarity is measured between the latent code (encoded mean vector) of the original sentence and the sentence after swaps are applied. We see that vMF is more highly sensitive to swaps in both the NVRNN and NVRNN-BoW settings, indicating that its latent space likely encodes more ordering information.

Fortunately, a wide range of κ values appear to work well for the tasks we consider. Figure 6 shows how the concentration parameter κ changes the results on PTB when the latent dimension and other hyperparameters are held fixed. We have ordered the tasks left-to-right from "hardest" to "easiest" in terms of necessity of latent representation: the Inputless setting needs heavy information from the latent code to reconstruct the sentence, whereas the Standard-BoW setting has an extremely strong decoder to predict the next word. We see that in each of these cases, a wide range of κ values works, and moreover reasonable κ values transfer between the two Standard and between the two Inputless settings, indicating that the overall approach is not highly sensitive to these hyperparameter values.

Brittleness of Learning κ Throughout this work, we have treated κ as a fixed parameter. However, we can treat κ in the same way as σ in the Gaussian case and learn it on a per-instance basis. The KL divergence of vMF is differentiable with respect to κ given gradients of the modified Bessel function of the first kind,[9] allowing us to change the concentration on a per-instance basis. However, this reintroduces the issue of KL collapse: the KL term will encourage κ to be as low

---

[8] The Gaussian VAE here makes very little use of the latent variable, hence why the representations change very little.

[9] $\nabla_\kappa I_d(\kappa) = \frac{1}{2}(I_{d-1}(\kappa) + I_{d+1}(\kappa))$

Figure 6: Perplexity of v-VAE in different settings with different κ values when the latent dimension is 50. Darker colors correspond to perplexity values closer to the best observed for that setting. For each task, we see that there is a range of κ values that work well, and these transfer between comparable tasks.

as possible, potentially making the latent variable vacuous.

In practice, we observe that it is necessary to clip κ values to a certain range for numerical reasons. Within this range, the model gravitates towards the smallest κ values and performs substantially worse than models trained with our fixed κ approach. This indicates that even with the vMF model, the optimization problem posed by ELBO is simply a hard one and the approach of fixing KL divergence is a surprisingly good optimization technique.

8 Related Work

Applications of VAE in NLP Deep generative models have achieved impressive successes in domains adjacent to NLP such as image generation (Gregor et al., 2015; Oord et al., 2016a) and speech generation (Chung et al., 2015; Oord et al., 2016b). VAEs specifically (Kingma and Welling, 2013; Rezende et al., 2014) have been a popular model variant in NLP. They have been applied to tasks including document modeling (Miao et al., 2016), language modeling (Bowman et al., 2016), and dialogue generation (Serban et al., 2017b). VAEs can be also be applied for semi-supervised classification (Xu et al., 2017). Recent twists on the standard VAE approach including combining VAE and holistic attribute discriminators for conditional generation (Hu et al., 2017) and using a more flexible latent space regularized by an adversarial method (Zhao et al., 2017a).

VAE Objective Several pieces of recent work have highlighted the issues with optimizing the VAE objective. Alemi et al. (2018) shed light on the problem from the perspective of information theory. Zhao et al. (2017b) and Higgins et al. (2017) both propose various reweightings of the objective along with theoretical and empirical justification.

Choices of Priors for VAE Some past work has explored various priors for VAE. Serban et al. (2017a) proposed a piecewise constant distribution which deals with multiple modes, but which sacrifices the property of continuous interpolation. Guu et al. (2018) also applied vMF in a VAE model, but used theirs specifically in the sentence-editing case. Davidson et al. (2018) explored vMF in a VAE model for MNIST and a link prediction task. Hasnat et al. (2017) applied the vMF distribution for facial recognition. Other past work has used different decoders, including CNNs (Yang et al., 2017) and CNN-RNN hybrids (Semeniuta et al., 2017). Changing the decoder is a change largely orthogonal to changing the prior: it can alleviate the KL vanishing issue, but it does not necessarily scale to new settings and does not give explicit control over utilization of the latent code.

9 Conclusion

In this paper, we propose the use of a von Mises-Fisher VAE to resolve optimization issues in variational autoencoders for text. This choice of distribution allows us to explicitly control the balance between the capacity of the decoder and the utilization of the latent representation in a principled way. Experimental results demonstrate that the proposed model has better performance than a Gaussian VAE across a range of settings. Further analysis shows that vMF VAE is more sensitive to word order information and makes more effective use of the latent code space.

Acknowledgments

This work was partially supported by NSF Grant IIS-1814522, a Bloomberg Data Science Grant, and an equipment grant from NVIDIA. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources used to conduct this research. Thanks as well to the anonymous reviewers for their helpful comments.

References

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif Saurous, and Kevin Murphy. 2018. Fixing a Broken ELBO. *International Conference on Machine Learning*.

Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, and Suvrit Sra. 2005. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Variational Lossy Autoencoder. *arXiv preprint arXiv:1611.02731*.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A Recurrent Latent Variable Model for Sequential Data. *Advances in Neural Information Processing Systems*.

Tim Davidson, Luca Falorsi, Nicola Cao, Thomas Kipf, and Jakub Tomczak. 2018. Hyperspherical Variational Auto-Encoders. *34th Conference on Uncertainty in Artificial Intelligence*.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. 2015. DRAW: A Recurrent Neural Network For Image Generation. *International Conference on Machine Learning*.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating Sentences by Editing Prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Md Hasnat, Julien Bohn, Jonathan Milgram, Stphane Gentric, and Liming Chen. 2017. von Mises-Fisher Mixture Model-based Deep learning: Application to Face Verification. *arXiv preprint arXiv:1706.04264*.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *International Conference on Learning Representations*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. *International Conference on Machine Learning*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.*, 19(2):313–330.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural Variational Inference for Text Processing. *International Conference on Machine Learning*.

Andriy Mnih and Karol Gregor. 2014. Neural Variational Inference and Learning in Belief Networks. *International Conference on Machine Learning*.

Aaron Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016a. Pixel Recurrent Neural Networks. *International Conference on Machine Learning*.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016b. WaveNet: A Generative Model for Raw Audio. *arXiv preprint arXiv:1609.03499*.

Danilo Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *International Conference on Machine Learning*.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A Hybrid Convolutional Variational Autoencoder for Text Generation. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Iulian Serban, Alexander G Ororbia, Joelle Pineau, and Aaron Courville. 2017a. Piecewise Latent Variables for Neural Variational Text Processing. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017b. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. *AAAI Conference on Artificial Intelligence*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style Transfer from Non-Parallel Text by Cross-Alignment. *Advances in Neural Information Processing Systems*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems*.

Andrew Wood. 1994. Simulation of the von Mises Fisher Distribution. *Communications in Statistics - Simulation and Computation*, 23(1):157–164.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Cached Long Short-Term Memory Neural Networks for Document-Level Sentiment Classification. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational Autoencoder for Semi-Supervised Text Classification. *AAAI Conference on Artificial Intelligence*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. *International Conference on Machine Learning*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *AAAI Conference on Artificial Intelligence*.

Biao Zhang, Deyi Xiong, Hong Duan, and Min Zhang. 2016. Variational Neural Machine Translation. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander M Rush, and Yann LeCun. 2017a. Adversarially Regularized Autoencoders. *arXiv*.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2017b. InfoVAE: Information Maximizing Variational Autoencoders. *arXiv*.

Learning Universal Sentence Representations with Mean-Max Attention Autoencoder

Minghua Zhang, Yunfang Wu[*]**, Weikang Li** and **Wei Li**
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University, Beijing, China
`{zhangmh,wuyf,wavejkd,liweitj47}@pku.edu.cn`

Abstract

In order to learn universal sentence representations, previous methods focus on complex recurrent neural networks or supervised learning. In this paper, we propose a mean-max attention autoencoder (mean-max AAE) within the encoder-decoder framework. Our autoencoder rely entirely on the MultiHead self-attention mechanism to reconstruct the input sequence. In the encoding we propose a mean-max strategy that applies both mean and max pooling operations over the hidden vectors to capture diverse information of the input. To enable the information to steer the reconstruction process dynamically, the decoder performs attention over the mean-max representation. By training our model on a large collection of unlabelled data, we obtain high-quality representations of sentences. Experimental results on a broad range of 10 transfer tasks demonstrate that our model outperforms the state-of-the-art unsupervised single methods, including the classical skip-thoughts (Kiros et al., 2015) and the advanced skip-thoughts+LN model (Ba et al., 2016). Furthermore, compared with the traditional recurrent neural network, our mean-max AAE greatly reduce the training time. [1]

1 Introduction

To automatically get the distributed representations of texts (words, phrases and sentences) is a fundamental task for natural language processing (NLP). There have been efficient learning algorithms to acquire the representations of words (Mikolov et al., 2013a), which have shown to provide useful features for various tasks. Interestingly, the acquired word representations reflect some observed aspects of human conceptual organization (Hill et al., 2015). In recent years, learning sentence representations has attracted much attention, which is to encode sentences into fixed-length vectors that could capture the semantic and syntactic properties of sentences and can then be transferred to a variety of other NLP tasks.

The most widely used method is to employ an encoder-decoder architecture with recurrent neural networks (RNN) to predict the original input sentence or surrounding sentences given an input sentence (Kiros et al., 2015; Ba et al., 2016; Hill et al., 2016; Gan et al., 2017). However, the RNN becomes time consuming when the sequence is long. The problem becomes more serious when learning general sentence representations that needs training on a large amount of data. For example, it took two weeks to train the skip-thought (Kiros et al., 2015). Moreover, the traditional RNN autoencoder generates words in sequence conditioning on the previous ground-truth words, i.e., teacher forcing training (Williams and Zipser, 1989). This teacher forcing strategy has been proven important because it forces the output of the RNN to stay close to the ground-truth sequence. However, at each time step, allowing the decoder solely to access the previous ground-truth words weakens the encoder's ability to learn the global information of the input sequence.

Some other approaches (Conneau et al., 2017; Cer et al., 2018; Subramanian et al., 2018) attempt to use the labelled data to build a generic sentence encoder, such as the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015), but such large-scale high-quality labelled data appropriate for training sentence representations is generally not available in other languages.

In this paper, we are interested in learning universal sentence representations based on a large amount of naturally occurring corpus, without using any labelled data. We propose a mean-max

---

[*] Corresponding author.

[1] Our code is publicly available at `https://github.com/Zminghua/SentEncoding`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4514–4523
Brussels, Belgium, October 31 - November 4, 2018.

attention autoencoder (mean-max AAE) to model sentence representations. Specifically, an encoder performs the MultiHead self-attention on an input sentence, and then the combined mean-max pooling operation is employed to produce the latent representation of the sentence. The representation is then fed into a decoder to reconstruct the input sequence, which also depends entirely on the MultiHead self-attention. At each time step, the decoder performs attention operations over the mean-max encoding, which on the one hand, enables the decoder to utilize the global information of the input sequence rather than generating words solely conditioning on the previous ground-truth words, and on the other hand, allows the decoder to attend to different representation subspaces dynamically.

We train our autoencoder on a large collection of unlabelled data, and evaluate the sentence embeddings across a diverse set of 10 transfer tasks. The experimental results show that our model outperforms the state-of-the-art unsupervised single models, and obtains comparable results with the combined models. Our mean-max representations yield consistent performance gain over the individual mean and max representations. At the same time, our model can be efficiently parallelized and so achieves significant improvement in computational efficiency.

In summary, our contributions are as follows:

- We apply the MultiHead self-attention mechanism to train autoencoder for learning universal sentence representations, which allows our model to do processing parallelization and thus greatly reduce the training time in large unlabelled data.
- we adopt a mean-max representation strategy in the encoding and then the decoder conducts attention over the latent representations, which can well capture the global information of the input from different views.
- After training only on naturally occurring unordered sentences, we obtain a simple and fast sentence encoder, which is an unsupervised single model and achieves state-of-the-art performance on various transfer tasks.

2 Related Work

With the flourishing of deep learning in NLP research, a variety of approaches have been developed for mapping word embeddings to fixed-length sentence representations. The methods generally fall into the following categories.

Unsupervised training with unordered sentences. This kind of methods depends only on naturally occurring individual sentences. Le and Mikolov (2014) propose the paragraph vector model, which incorporates a global context vector into the log-linear neural language model (Mikolov et al., 2013b), but at test time, inference needs to be performed to compute a new vector. Arora et al. (2017) propose a simple but effective Smooth Inverse Frequency (SIF) method, which represents sentence by a weighted average of word embeddings. Hill et al. (2016) introduce sequential denoising autoencoders (SDAE), which employ the denoising objective to predict the original source sentence given a corrupted version. They also implement bag-of-words models such as word2vec-SkipGram, word2vec-CBOW. Our model belongs to this group, which has no restriction on the required training data and can be trained on sets of sentences in arbitrary order.

Unsupervised training with ordered sentences. This kind of method is trained to predict the surrounding sentences of an input sentence, based on the naturally occurring coherent texts. Kiros et al. (2015) propose the skip-thoughts model, which uses an encoder RNN to encode a sentence and two decoder RNN to predict the surrounding sentences. The skip-thought vectors perform well on several tasks, but training this model is very slow, requiring several days to produce meaningful results. Ba et al. (2016) further obtain better results by adding layer-norm regularization on the skip-thoughts model. Gan et al. (2017) explore a hierarchical model to predict multiple future sentences, using a convolutional neural network (CNN) encoder and a long-short term memory (LSTM) decoder. Logeswaran and Lee (2018) reformulate the problem of predicting the context in which a sentence appears as a classification task. Given a sentence and its context, a classifier distinguishes context sentences from other contrastive sentences based on their vector representations.

Supervised learning of sentence representations. Hill et al. (2016) implement models trained on supervised data, including dictionary definitions, image captions from the COCO dataset (Lin et al., 2014) and sentence-aligned translated texts. Conneau et al. (2017) attempt to exploit the SNLI dataset for building generic sentence encoders. Through examining 7 different model schemes, they show that a bi-directional LSTM network with the max pooling yields excellent performance. Cer et al. (2018) apply multi-task learning to train sentence encoders, including a skip-thought like task, a conversational input-response task and classification tasks from the SNLI dataset. They also explore combining the sentence and word level transfer models. Subramanian et al. (2018) also present a multi-task learning framework for sentence representations, and train their model on several data resources with multiple training objectives on over 100 million sentences.

3 Model Description

Our model follows the encoder-decoder architecture, as shown in Figure 1. The input sequence is compressed into a latent mean-max representation via an encoder network, which is then used to reconstruct the input via a decoder network.

3.1 Notation

In our model, we treat the input sentence as one sequence of tokens. Let S denote the input, which is comprised of a sequence of tokens $\{w_1, w_2, \ldots, w_N\}$, where N denotes the length of the sequence. An additional "$</S>$" token is appended to each sequence. Each word w_t in S is embedded into a k-dimensional vector $e_t = W_e[w_t]$, where $W_e \in \mathbb{R}^{d_w \times V}$ is a word embedding matrix, w_t indexes one element in a V-dimensional set (vocabulary), and $W_e[v]$ denotes the v-th column of matrix W_e.

In order for the model to take account of the sequence order, we also add "positional encodings" (Vaswani et al., 2017) to the input embeddings:

$$p_t[2i] = sin(\frac{t}{10000^{2i/d_w}}) \tag{1}$$

$$p_t[2i+1] = cos(\frac{t}{10000^{2i/d_w}}) \tag{2}$$

where t is the position and i is the dimension. Each dimension of the positional encoding corresponds to a sinusoid. Therefore, the input of our model can be represented as $x_t = e_t + p_t$.

In the following description, we use h_t^e and h_t^d to denote the hidden vectors of the encoder and decoder respectively, the subscripts of which indicate timestep t, and the superscripts indicate operations at the encoding or decoding stage.

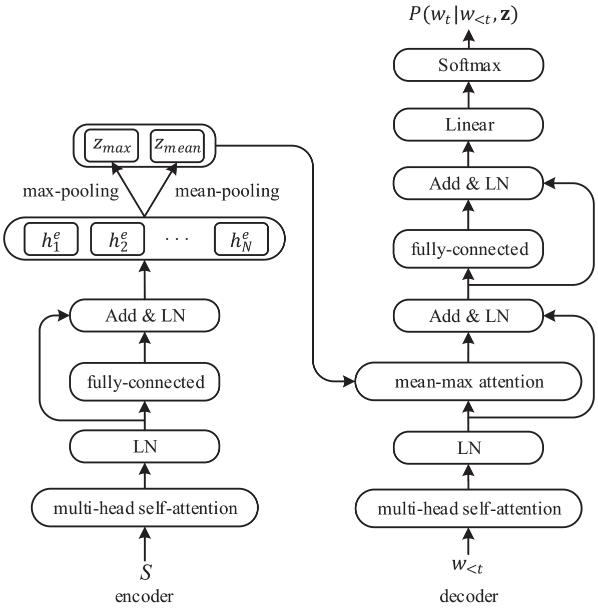

Figure 1: Illustration of the mean-max attention autoencoder.

3.2 MultiHead Self-Attention

In this subsection, we give a quick overview of MultiHead Self-Attention mechanism (Vaswani et al., 2017). The attention is to map a query q and a set of key-value pairs (K, V) to an output. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed based on the query and the corresponding key. The MultiHead mechanism applies multiple attention operations in parallel. Given q and (K, V), we can obtain the attention vector a by:

$$\begin{aligned} a &= MultiHead(q, K, V) && (3)\\ &= concat(head_1, \ldots, head_l) && (4)\\ head_i &= attention(\overline{q}, \overline{K}, \overline{V}) && (5)\\ &= softmax(\frac{\overline{q}\overline{K}^T}{\sqrt{d_k}})\overline{V} && (6) \end{aligned}$$

where

$$\overline{q}, \overline{K}, \overline{V} = qW_i^q, KW_i^K, VW_i^V \tag{7}$$

W_i^q, W_i^K and W_i^V are parameter matrices; $\overline{q} \in \mathbb{R}^{d_k}$, $\overline{K} \in \mathbb{R}^{n_k \times d_k}$ and $\overline{V} \in \mathbb{R}^{n_k \times d_v}$; d_k and d_v are the dimensions of $\overline{K}$ and $\overline{V}$ respectively; n_k is the number of key-value pairs.

The MultiHead self-attention allows the model to jointly attend to information from different positions. Due to the reduced dimension of each head and parallel operations, the total computational cost is similar to that of a single-head attention.

3.3 Attention Encoder

The encoder has two sub-layers. The first is a MultiHead self-attention mechanism, and the second is a position-wise fully connected feed-forward network which consists of two linear transformations with a ReLU activation in between. Different from Vaswani et al. (2017), we remove the residual connections in the MultiHead self-attention layer and only employ a residual connection in the fully connected layer, allowing the model to expand the dimension of hidden vectors to incorporate more information.

Given the input $\mathbf{x} = (x_1, \ldots, x_N)$, the hidden vector h_t^e at time-step t is computed by:

$$a_t^e = MultiHead(x_t, \mathbf{x}, \mathbf{x}) \tag{8}$$
$$\overline{a}_t^e = LN(a_t^e) \tag{9}$$
$$\overline{h}_t^e = max(0, \overline{a}_t^e W_1^e + b_1^e) W_2^e + b_2^e \tag{10}$$
$$h_t^e = LN(\overline{h}_t^e + \overline{a}_t^e) \tag{11}$$

where $W_1^e \in \mathbb{R}^{d_m \times d_f}$ and $W_2^e \in \mathbb{R}^{d_f \times d_m}$ are parameter matrices; $b_1^e \in \mathbb{R}^{d_f}$ and $b_2^e \in \mathbb{R}^{d_m}$ are bias vectors; d_m and d_f are the dimensions of hidden vector and fully connected inner layer respectively; LN denotes layer normalization.

Our model can be efficiently parallelized over the whole input. We can obtain all hidden vectors $(h_1^e, \ldots, h_N^e)$ simultaneously for an input sequence, thus greatly reducing the computational complexity compared with the sequential processing of LSTM.

3.4 Mean-Max Representation

Given the varying number of hidden vectors $\{h_t^e\}_{t=[1,\ldots,N]}$, we need to transform these local hidden vectors into a global sentence representation. We would like to apply the pooling strategy, which makes the extracted representation independent of the length of the input sequence and obtains a fixed-length vector. Conneau et al. (2017) examine BiLSTM with mean and max pooling for fixed-size sentence representation, and they conclude that the max pooling operation performs better on transfer tasks.

In this work, we propose to apply mean and max pooling simultaneously. The max pooling takes the maximum value over the sequence, which tries to capture the most salient property while filtering out less informative local values. On the other hand, the mean pooling does not make sharp choices on which part of the sequence is more important than others, and so it captures general information while not focusing too much on specific features. Obviously, the two pooling strategies can complement each other. The mean-max representation is obtained by:

$$z_{max}[i] = \max_t h_{ti}^e \tag{12}$$
$$z_{mean} = \frac{1}{N} \sum_t h_t^e \tag{13}$$
$$\mathbf{z} = [z_{max}, z_{mean}] \tag{14}$$

Through combining two different pooling strategies, our model enjoys the following advantages. First, in the encoder, we can summarize the hidden vectors from different perspectives and so capture more diverse features of the input sequence, which will bring robustness on different transfer tasks. Second, in the decoder (as described in the next subsection), we can perform attention over the mean-max representation rather than over the local hidden vectors step by step, which would potentially make the autoencoder objective trivial.

3.5 Attention Decoder

As with the encoder, the decoder also applies the MultiHead self-attention to reconstruct the input sequence. As shown in Figure 1, the encoder and decoder are connected through a mean-max attention layer, which performs attention over the mean-max representation generated by the encoder.

To facilitate expansion of the hidden size, we employ residual connections in the mean-max attention layer and the fully connected layer, but not in the MultiHead self-attention layer. Given $\mathbf{y} = (x_1, \ldots, x_{t-1})$ and $\mathbf{z}$ as the decoder input, the hidden vector h_t^d at time step t is obtained by:

$$a_t^d = MultiHead(y_t, \mathbf{y}, \mathbf{y}) \tag{15}$$
$$\overline{a}_t^d = LN(a_t^d) \tag{16}$$
$$a_t^z = MultiHead(\overline{a}_t^d, \mathbf{z}, \mathbf{z}) \tag{17}$$
$$\overline{a}_t^z = LN(a_t^z + \overline{a}_t^d) \tag{18}$$
$$\overline{h}_t^d = max(0, \overline{a}_t^z W_1^d + b_1^d) W_2^d + b_2^d \tag{19}$$
$$h_t^d = LN(\overline{h}_t^d + \overline{a}_t^z) \tag{20}$$

where $W_1^d \in \mathbb{R}^{d_m \times d_f}$ and $W_2^d \in \mathbb{R}^{d_f \times d_m}$ are parameter matrices; $b_1^d \in \mathbb{R}^{d_f}$ and $b_2^d \in \mathbb{R}^{d_m}$ are bias vectors. $\mathbf{z}$ in Equation (17) is the mean-max representation generated by Equation (14).

Given the hidden vectors $(h_1^d, \ldots, h_N^d)$, the probability of generating a sequence S with length-N is defined as:

$$P(w_t|w_{<t}, \mathbf{z}) \propto exp(W_3^d h_t^d + b_3^d) \tag{21}$$

$$J(\theta) = \sum_t logP(w_t|w_{<t}, \mathbf{z}) \tag{22}$$

The model learns to reconstruct the input sequence by optimizing the objective in Equation (22).

4 Evaluating Sentence Representations

In the previous work, researchers evaluated the distributed representations of sentences by adding them as features in transfer tasks (Kiros et al., 2015; Gan et al., 2017; Conneau et al., 2017). We use the same benchmarks and follow the same procedure to evaluate the capability of sentence embeddings produced by our generic encoder.

4.1 Transfer Tasks

We conduct extensive experiments on 10 transfer tasks. We first study the classification task on 6 benchmarks: movie review sentiment (MR, SST) (Pang and Lee, 2005; Socher et al., 2013), customer product reviews (CR) (Hu and Liu, 2004), subjectivity/objectivity classification (SUBJ) (Pang and Lee, 2004), opinion polarity (MPQA) (Wiebe et al., 2005) and question type classification (TREC) (Li and Roth, 2002).

We also consider paraphrase detection on the Microsoft Research Paraphrase Corpus (MRPC) (Dolan et al., 2004), where the evaluation metrics are accuracy and F1 score.

We then evaluate on the SICK dataset (Marelli et al., 2014) for both textual entailment (SICK-E) and semantic relatedness (SICK-R). The evaluation metric is Pearson correlation for SICK-R. We also evaluate on the SemEval task of STS14 (Agirre et al., 2014), where the evaluation metrics are Pearson and Spearman correlations.

The processing on each task is as follows: 1) Employ the pre-trained attention autoencoder to encode all sentences into the latent mean-max representations. 2) Using the representations as features, apply the open SentEval with a logistic regression classifier (Conneau et al., 2017) to automatically evaluate on all the tasks. For a fair comparison of the plain sentence embeddings, we adopt all the default settings.

4.2 Experiment Setup

We train our model on the open Toronto Books Corpus (Zhu et al., 2015), which was also used to train the skip-thoughts (Kiros et al., 2015) and skip-thoughts+LN (Ba et al., 2016). The Toronto Book Corpus consists of 70 million sentences from over $7,000$ books, which is not biased towards any particular domain or application.

The dimensions of hidden vectors and fully connected inner layer are set to $2,048$ and $4,096$ respectively. Hence, our mean-max AAE represents sentences with $4,096$ dimensional vectors. We set $l = 8$ parallel attention heads according to the development data.

We use the Adam algorithm (Kingma and Ba, 2014) with learning rate 2×10^{-4} for optimization. Gradient clipping is adopted by scaling gradients when the norm of the parameter vector exceeds a threshold of 5. We perform dropout (Srivastava et al., 2014) and set the dropout rate to 0.5. Minibatches of size 64 are used. Our model learns until the reconstruction accuracy in the development data stops improving.

Our aim is to learn a generic sentence encoder that could encode a large number of words. There always are some words that haven't been seen during training, and so we use the publicly available GloVe vectors [2] to expand our encoder's vocabulary. We set the word vectors in our models as the corresponding word vectors in GloVe, and do not update the word embeddings during training. Thus, any word vectors from GloVe can be naturally used to encode sentences. Our models are trained with a vocabulary of $21,583$ top frequent words in the Toronto Book corpus. After vocabulary expansion, we can now successfully cover $2,196,017$ words.

All experiments are implemented in Tensorflow (Abadi et al., 2016), using a NVIDIA GeForce GTX 1080 GPU with 8GB memory.

4.3 Evaluation Results

A summary of our experimental results on 10 tasks is given in Table 1, in which the evaluation metric of the first 8 tasks is accuracy. To make a clear comparison of the overall performance, we compute the "macro" and "micro" average of accuracy

---

[2] `https://nlp.stanford.edu/projects/glove/`

Method	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	SICK-E	SICK-R	STS14
Unsupervised training of single model										
ParagraphVec (DBOW)	60.2	66.9	76.3	70.7	-	59.4	72.9/81.1	-	-	.42/.43
SDAE	74.6	78.0	90.8	86.9	-	78.4	73.7/80.7	-	-	.37/.38
SIF (GloVe + WR)	-	-	-	-	82.2	-	-	**84.6**	-	**.69/ -**
word2vec BOW†	77.7	79.8	90.9	88.3	79.7	83.6	72.5/81.4	78.7	0.803	**.65/.64**
GloVe BOW†	**78.7**	78.5	91.6	87.6	79.8	83.6	72.1/80.9	78.6	0.800	.54/.56
FastSent	70.8	78.4	88.7	80.6	-	76.8	72.2/80.3	-	-	.63/.64
FastSent+AE	71.8	76.7	88.8	81.5	-	80.4	71.2/79.1	-	-	.62/.62
uni-skip	75.5	79.3	92.1	86.9	-	**91.4**	73.0/81.9	-	-	-
bi-skip	73.9	77.9	92.5	83.3	-	89.4	71.2/81.2	-	-	-
hierarchical-CNN	75.3	79.3	91.9	88.4	-	90.4	74.2/**82.7**	-	-	-
composite-CNN	77.1	80.6	92.1	88.6	-	**91.2**	**74.8**/82.2	-	-	-
skip-thoughts+LN†	**79.4**	**83.1**	**93.7**	**89.3**	**82.9**	88.4	-	79.5	**0.858**	.44/.45
mean-max AAE	**78.7**	**82.3**	**93.4**	**88.8**	**83.8**	**91.4**	**75.5/82.6**	**83.5**	**0.854**	.58/.56
Unsupervised training of combined model										
combine-skip†	76.5	80.1	93.6	87.1	82.0	92.2	73.0/82.0	82.3	0.858	.29/.35
combine-CNN	77.7	82.0	93.6	89.3	-	92.6	76.4/83.7	-	-	-
Trained on supervised data										
BiLSTM-Max (on SST)	*	83.7	90.2	89.5	*	86.0	72.7/80.9	83.1	0.863	.55/.54
BiLSTM-Max (on SNLI)	79.9	84.6	92.1	89.8	83.3	88.7	75.1/82.3	86.3	0.885	.68/.65
BiLSTM-Max (on AllNLI)	81.1	86.3	92.4	90.2	84.6	88.2	76.2/83.1	86.3	0.884	.70/.67

Table 1: Performance of sentence representation models on 10 transfer tasks. Top 2 results of unsupervised single models are shown in bold. The results with † are extracted from Conneau et al. (2017).

Method	Macro/Micro		
	8_tasks	7_tasks	6_tasks
uni-skip	-	-	83.0/83.7
bi-skip	-	-	81.4/82.1
hierarchical-CNN	-	-	83.3/84.0
composite-CNN	-	-	84.1/84.8
skip-thoughts+LN	-	85.2/85.9	-
mean-max AAE	**84.7/85.6**	**86.0/86.0**	85.0/**85.9**
combine-skip	83.4/84.2	-	-
combine-CNN	-	-	**85.3**/85.8
BiLSTM-Max(SST)	-	-	-
BiLSTM-Max(SNLI)	85.0/86.2	-	-
BiLSTM-Max(AllNLI)	85.7/86.9	-	-

Table 2: The macro and micro average accuracy across different tasks. The bold is the highest score among all unsupervised models.

in Table 2, where the micro average is weighted by the number of test samples in each task. In the previous work, different approaches conduct experiments on different benchmarks. Therefore we report the average scores on 6 tasks, 7 tasks and 8 tasks, respectively.

We divide related models into three groups. The first group contains unsupervised single models, including the Paragraph Vector model (Le and Mikolov, 2014), the SDAE method (Hill et al., 2016), the SIF model (Arora et al., 2017), the FastSent (Hill et al., 2016), the skip-thoughts (uni-skip and bi-skip) (Kiros et al., 2015), the CNN encoder (hierarchical-CNN and composite-CNN) (Gan et al., 2017) and skip-thoughts+LN (Ba et al., 2016). Our mean-max attention autoencoder sits in this group. The second group consists of unsupervised combined models, including combine-skip (Kiros et al., 2015) and combine-CNN (Gan et al., 2017). In the third group, we list the results from the work of Conneau et al. (2017) only for reference, since it is trained on labelled data.

Comparison with skip-thoughts+LN. The skip-thoughts+LN is the best model among the existing single models. Compared with the skip-thoughts+LN, our method obtains better results on 4 datasets (SST, TREC, SICK-E, STS14) and comparable results on 3 datastes (SUBJ, MPQA, SICK-R). Looking at the STS14 results, we observe that the cosine metrics in our representation space is much more semantically informative than in skip-thoughts+LN representation space (pearson score of 0.58 compared to 0.44). Considering the overall performance shown in Table 2, our model obtains better results both in the macro and micro average accuracy across 7 considered tasks. In view of the required training data, the skip-thoughts+LN needs coherent texts while our model needs only individual sentences. Moreover, we train our model in less than 5 hours on a single GPU compared to the best skip-thoughts+LN network trained for a month.

Unsupervised combined models. The results of the individual models (Kiros et al., 2015; Gan et al., 2017) are not promising. To get better performance, they train two separate models on the same corpus and then combine the latent representations together. As shown in Table 2, our mean-max attention autoencoder outperforms the classical combine-skip model by 1.3 points in the average performance across 8 considered tasks. Specially, the pearson correlation of our model is 2 times over the combine-skip model on the STS14 task. Looking at the overall performance of 6 tasks, our model gets comparable results with the combine-CNN, which combines the hierarchical and composite approaches to exploit the intra-sentence and inter-sentence information. Obviously, our model is simple and fast to implement compared with the combined methods.

Supervised representation training. It is unfair to directly compare our totally unsupervised model with the supervised representation learning method. Conneau et al. (2017) train the BiLSTM-Max (on ALLNLI) on the high-quality natural language inference data. Our model even performs better than the BiLSTM-Max (on ALLNLI) on the SUBJ and TREC tasks. More importantly, our model can be easily adapted to other low-resource languages.

4.4 Model Analysis

Our model contains three main modules: the mean-max attention layer, the combined pooling strategy and the encoder-decoder network. We make a further study on these components. The experimental results are shown in Table 3.

In our model, the mean-max attention layer allows the decoder to pay attention to the encoding representation of the full sentence at each time step dynamically. To summarize the contribution of the mean-max attention layer, we compare with traditional baselines, including the sequential denoising autoencoder (SDAE) with LSTM networks (Hill et al., 2016) and the CNN-LSTM autoencoder (Gan et al., 2017), both of which only use the encoding representation to set the initial state of the decoder and follow the teacher forcing strategy.

We employ both the mean and max pooling operations over the local hidden vectors to obtain sentence embeddings. To validate the effectiveness of our mean-max representations, we train two additional models: (i) an attention autoencoder only with max pooling (max AAE) and (ii) an attention autoencoder only with mean pooling (mean AAE). The dimension of hidden vectors is also set to $2,048$.

Our encoder-decoder network depends on the MultiHead self-attention mechanism to reconstruct the input sequence. To test the effect of the MultiHead self-attention mechanism, we replace it with RNN and implement a mean-max RNN autoencoder (mean-max RAE) training on the same Toronto Books Corpus. A bidirectional LSTM computes a set of hidden vectors on an input sentence, and then the mean and max pooling operations are employed to generate the latent mean-max representation. The representation is then fed to a LSTM decoder to reconstruct the input sequence through attention operation over the latent representation. The parameter configurations are consistent with our other models. Moreover, we also train two additional models with different pooling strategies: mean RAE and max RAE.

Analysis on the mean-max attention layer. Our mean-max attention layer brings significant performance gain over the previous autoencoders. Compared the mean RAE with LSTM-SDAE, both of which use the RNN-RNN encoder-decoder network to reconstruct the input sequence, our mean RAE consistently obtains better performance than LSTM-SDAE across all considered tasks. In particular, it yields a performance gain of 10.4 on the TREC dataset and 29 on the STS14 dataset. Compared with another CNN-LSTM autoencoder, our mean RAE also gets better performance for all but one task. It demonstrates that the mean-max attention layer enables the decoder to attend to the global information of the input sequence, thus go beyond the “teacher forcing training”.

Analysis on the pooling strategy. Considering the overall performance, our mean-max representations outperform the individual mean and max representations both in the attention and RNN networks. In our attention autoencoder, the macro average score of the mean-max AAE is more than 0.6 over the individual pooling strategy. In the RNN autoencoder, the combined pooling strategy

Method	Macro	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	SICK-E	SICK-R	STS14
mean AAE	84.1	78.6	81.6	93.3	88.6	83.5	90.2	74.8/82.3	82.2	0.851	.57/.55
max AAE	84.1	78.6	81.4	93.1	88.7	82.8	90.6	75.1/82.5	82.2	0.851	.57/.55
mean-max AAE	84.7	78.7	82.3	93.4	88.8	83.8	91.4	75.5/82.6	83.5	0.854	.58/.56
mean RAE	83.1	77.1	80.2	92.4	88.4	81.9	88.8	74.3/82.2	81.9	0.836	.66/.64
max RAE	83.0	77.2	80.1	92.0	88.4	81.0	88.6	74.3/82.0	82.0	0.851	.68/.66
mean-max RAE	83.6	77.4	80.1	92.2	88.5	82.3	90.6	74.6/82.0	83.2	0.853	.68/.65
LSTM SDAE	-	74.6	78.0	90.8	86.9	-	78.4	73.7/80.7	-	-	.37/.38
CNN-LSTM AE	-	75.5	79.0	92.0	88.0	-	89.8	73.6/82.1	-	-	-

Table 3: Performance of different pooling strategies and different encoder-decoder networks on 10 transfer tasks. Macro is the macro average over the first 8 tasks whose metric is accuracy.

yields a performance gain of 0.5 over the mean pooling and 0.6 over the max pooling. The results indicate that our mean-max pooling captures more diverse information of the input sequence, which is robust and effective in dealing with various transfer tasks.

Comparison with RNN-based autoencoder. As shown in Table 3, our MultiHead self-attention network obtains obvious improvement over the RNN network in different sets of pooling strategies, and it yields a performance gain of 1.1 when applying the best combined mean-max pooling operation. The results demonstrate that the MultiHead self-attention mechanism enables the sentence representations to capture more useful information about the input sequence.

Analysis on computational complexity. A self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires $O(n)$ sequential operations. Therefore, our model greatly reduces the computational complexity. Excluding the number of parameters used in the word embeddings, the skip-thought model (Kiros et al., 2015) contains 40 million parameters, while our mean-max AAE has approximately 39 million parameters. It took nearly 50.4 and 25.4 minutes to train the skip-thought model (Kiros et al., 2015) and the skip-thoughts+LN (Ba et al., 2016) per 1000 mini-batches respectively. Both the skip-thought and skip-thought+LN are implemented in Theano. A recent implementation of the skip-thoughts model was released by Google [3], which took nearly 25.9 minutes to train 1000 mini-batches on a GTX 1080 GPU. In our experiment, it took 3.3 minutes to train the mean-max AAE model every 1000 mini-batches.

Figure 2: Two examples illustrate that our mean-max attention layer could attend to the two different representations dynamically.

4.5 Attention Visualization

The above experimental results have proven the effectiveness of the mean-max attention mechanism in the decoding. We further inspect the attention distributions captured by the mean-max attention layer, as shown in Figure 2. The side-by-side heat illustrates how much the decoder pay attention to the mean representation and max representation respectively at each decoding step. We can see that the attention layer learns to selectively retrieve the mean or max representations dynamically, which relieve the decoder from the burden of generating words solely conditioning on the previous ground-truth words. Also, the two different representations can complement each other, and the mean representation plays a greater role.

---

[3] https://github.com/tensorflow/models/tree/master/research/skip_thoughts

5 Conclusion

In this paper, we present a mean-max AAE to learn universal sentence representations from unlabelled data. Our model applies the MultiHead self-attention mechanism both in the encoder and decoder, and employs a mean-max pooling strategy to capture more diverse information of the input. To avoid the impact of "teacher forcing training", our decoder performs attention over the encoding representations dynamically. To evaluate the effectiveness of sentence representations, we conduct extensive experiments on 10 transfer tasks. The experimental results show that our model obtains state-of-the-art performance among the unsupervised single models. Furthermore, it is fast to train a high-quality generic encoder due to the paralleling operation. In the future, we will adapt our mean-max AAE to other low-resource languages for learning universal sentence representations.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work is supported by the National Key Basic Research Program of China (2014CB340504) and National Natural Science Foundation of China (61773026, 61572245). The corresponding author of this paper is Yunfang Wu.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *International Workshop on Semantic Evaluation*, pages 81–91.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *International Conference on Computational Linguistics*, page 350.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2017. Learning generic sentence representations using convolutional neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2380–2390.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of NAACL-HLT*, pages 1367–1377.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, Usa, August*, pages 168–177.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Xin Li and Dan Roth. 2002. Learning question classifiers. *Coling*, 12(24):556–562.

Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Language Resources and Evaluation Conference*, pages A–696.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Bo Pang and Lillian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Meeting on Association for Computational Linguistics*, page 271.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv preprint arXiv:1506.06724*.

Word Mover's Embedding:
From Word2Vec to Document Embedding

Lingfei Wu
IBM Research
wuli@us.ibm.com

Ian En-Hsu Yen
Carnegie Mellon University
eyan@cs.cmu.edu

Kun Xu
IBM Research
kun.xu1@ibm.com

Fangli Xu
College of William and Mary
fxu02@email.wm.edu

Avinash Balakrishnan
IBM Research
avinash.bala@us.ibm.com

Pin-Yu Chen
IBM Research
Pin-yu.chen@ibm.com

Pradeep Ravikumar
Carnegie Mellon University
pradeepr@cs.cmu.edu

Michael J. Witbrock
IBM Research
witbrock@us.ibm.com

Abstract

While the celebrated Word2Vec technique yields semantically rich representations for individual words, there has been relatively less success in extending to generate unsupervised sentences or documents embeddings. Recent work has demonstrated that a distance measure between documents called *Word Mover's Distance* (WMD) that aligns semantically similar words, yields unprecedented KNN classification accuracy. However, WMD is expensive to compute, and it is hard to extend its use beyond a KNN classifier. In this paper, we propose the *Word Mover's Embedding* (WME), a novel approach to building an unsupervised document (sentence) embedding from pre-trained word embeddings. In our experiments on 9 benchmark text classification datasets and 22 textual similarity tasks, the proposed technique consistently matches or outperforms state-of-the-art techniques, with significantly higher accuracy on problems of short length.

1 Introduction

Text representation plays an important role in many NLP-based tasks such as document classification and clustering (Zhang et al., 2018; Gui et al., 2016, 2014), sense disambiguation (Gong et al., 2017, 2018a), machine translation (Mikolov et al., 2013b), document matching (Pham et al., 2015), and sequential alignment (Peng et al., 2016, 2015). Since there are no explicit features in text, much work has aimed to develop effective text representations. Among them, the simplest bag of words (BOW) approach (Salton and Buckley, 1988) and its term frequency variants (e.g. TF-IDF) (Robertson and Walker, 1994) are most widely used due to simplicity, efficiency and often surprisingly high accuracy (Wang and Manning, 2012). However, simply treating words and phrases as discrete symbols fails to take into account word order and the semantics of the words, and suffers from frequent near-orthogonality due to its high dimensional sparse representation. To overcome these limitations, Latent Semantic Indexing (Deerwester et al., 1990) and Latent Dirichlet Allocation (Blei et al., 2003) were developed to extract more meaningful representations through singular value decomposition (Wu and Stathopoulos, 2015) and learning a probabilistic BOW representation.

A recent empirically successful body of research makes use of distributional or contextual information together with simple neural-network models to obtain vector-space representations of words and phrases (Bengio et al., 2003; Mikolov et al., 2013a,c; Pennington et al., 2014). A number of researchers have proposed extensions of these towards learning semantic vector-space representations of sentences or documents. A simple but often effective approach is to use a weighted average over some or all of the embeddings of words in the document. While this is simple, important information could easily be lost in such a document representation, in part since it does not consider word order. A more sophisticated approach (Le and Mikolov, 2014; Chen, 2017) has focused on jointly learning embeddings for both words and paragraphs using models similar to Word2Vec. However, these only use word order within a small context window; moreover, the quality of word embeddings learned

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4524–4534
Brussels, Belgium, October 31 - November 4, 2018.

in such a model may be limited by the size of the training corpus, which cannot scale to the large sizes used in the simpler word embedding models, and which may consequently weaken the quality of the document embeddings.

Recently, Kusner et al. (Kusner et al., 2015) presented a novel document distance metric, Word Mover's Distance (WMD), that measures the dissimilarity between two text documents in the Word2Vec embedding space. Despite its state-of-the-art KNN-based classification accuracy over other methods, combining KNN and WMD incurs very high computational cost. More importantly, WMD is simply a distance that can be only combined with KNN or K-means, whereas many machine learning algorithms require a fixed-length feature representation as input.

A recent work in building kernels from distance measures, D2KE (distances to kernels and embeddings) (Wu et al., 2018a) proposes a general methodology of the derivation of a positive-definite kernel from a given distance function, which enjoys better theoretical guarantees than other distance-based methods, such as k-nearest neighbor and distance substitution kernel (Haasdonk and Bahlmann, 2004), and has also been demonstrated to have strong empirical performance in the time-series domain (Wu et al., 2018b).

In this paper, we build on this recent innovation D2KE (Wu et al., 2018a), and present the *Word Mover's Embedding* (WME), an unsupervised generic framework that learns continuous vector representations for text of variable lengths such as a sentence, paragraph, or document. In particular, we propose a new approach to first construct a positive-definite *Word Mover's Kernel* via an infinite-dimensional feature map given by the Word Mover's distance (WMD) to random documents from a given distribution. Due to its use of the WMD, the feature map takes into account alignments of individual words between the documents in the semantic space given by the pre-trained word embeddings. Based on this kernel, we can then derive a document embedding via a Random Features approximation of the kernel, whose inner products approximate exact kernel computations. Our technique extends the theory of *Random Features* to show convergence of the inner product between WMEs to a positive-definite kernel that can be interpreted as a soft version of (inverse) WMD.

The proposed embedding is more efficient and flexible than WMD in many situations. As an example, WME with a simple linear classifier reduces the computational cost of WMD-based KNN *from cubic to linear* in document length and *from quadratic to linear* in number of samples, while simultaneously improving accuracy. WME is extremely easy to implement, fully parallelizable, and highly extensible, since its two building blocks, Word2Vec and WMD, can be replaced by other techniques such as GloVe (Pennington et al., 2014; Wieting et al., 2015b) or S-WMD (Huang et al., 2016). We evaluate WME on 9 real-world text classification tasks and 22 textual similarity tasks, and demonstrate that it consistently matches or outperforms other state-of-the-art techniques. Moreover, WME often achieves orders of magnitude speed-up compared to KNN-WMD while obtaining the same testing accuracy. Our code and data is available at `https://github.com/IBM/WordMoversEmbeddings`.

2 Word2Vec and Word Mover's Distance

We briefly introduce Word2Vec and WMD, which are the key building blocks of our proposed method. Here are some notations we will use throughout the paper. Given a total number of documents N with a vocabulary $\mathcal{W}$ of size $|\mathcal{W}| = n$, the Word2vec embedding gives us a d-dimensional vector space $\mathcal{V} \subseteq \mathbb{R}^d$ such that any word in the vocabulary set $w \in \mathcal{W}$ is associated with a semantically rich vector representation $\boldsymbol{v}_w \in \mathbb{R}^d$. Then in this work, we consider each document as a collection of word vectors $x := (\boldsymbol{v}_j)_{j=1}^{L}$ and denote $\mathcal{X} := \bigcup_{L=1}^{L_{\max}} \mathcal{V}^L$ as the space of documents.

Word2Vec. In the celebrated Word2Vec approach (Mikolov et al., 2013a,c), two shallow yet effective models are used to learn vector-space representations of words (and phrases), by mapping those that co-occur frequently, and consequently with plausibly similar meaning, to nearby vectors in the embedding vector space. Due to the model's simplicity and scalability, high-quality word embeddings can be generated to capture a large number of precise syntactic and semantic word relationships by training over hundreds of billions of words and millions of named entities. The advantage of document representations building on top of word-level embeddings is that one can make full use of high-quality pre-trained word embeddings. Throughout this paper we use Word2Vec as our first building block but other (unsupervised or supervised) word

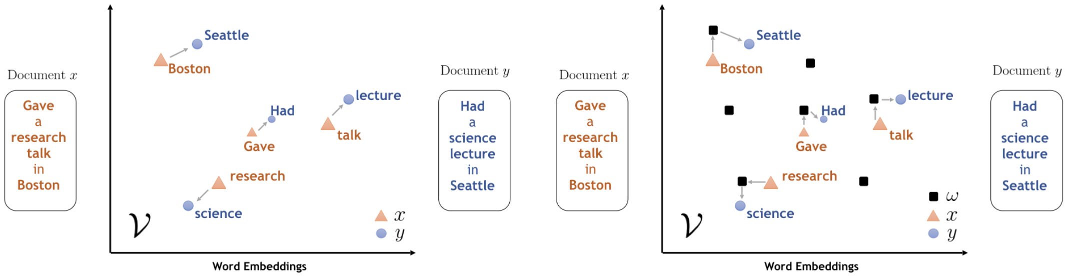

(a) WMD

(b) WME

Figure 1: An illustration of the WMD and WME. All non-stop words are marked as bold face. WMD measures the distance between two documents. WME approximates a kernel derived from WMD with a set of random documents.

embeddings (Pennington et al., 2014; Wieting et al., 2015b) could also be utilized.

Word Mover's Distance. Word Mover's Distance was introduced by (Kusner et al., 2015) as a special case of the Earth Mover's Distance (Rubner et al., 2000), which can be computed as a solution of the well-known transportation problem (Hitchcock, 1941; Altschuler et al., 2017). WMD is a distance between two text documents x, $y \in \mathcal{X}$ that takes into account the alignments between words. Let $|x|$, $|y|$ be the number of distinct words in x and y. Let $\boldsymbol{f}_x \in \mathbb{R}^{|x|}$, $\boldsymbol{f}_y \in \mathbb{R}^{|y|}$ denote the normalized frequency vectors of each word in the documents x and y respectively (so that $\boldsymbol{f}_x^\intercal \mathbf{1} = \boldsymbol{f}_y^\intercal \mathbf{1} = 1$). Then the WMD distance between documents x and y is defined as:

$$\begin{aligned} \mathrm{WMD}(x,y) &:= \min_{F\in\mathbb{R}_+^{|x|\times|y|}} \langle C, F\rangle, \\ s.t., F\mathbf{1} &= \boldsymbol{f}_x, \quad F^\intercal\mathbf{1} = \boldsymbol{f}_y. \end{aligned} \tag{1}$$

where F is the transportation flow matrix with F_{ij} denoting the amount of flow traveling from i-th word x_i in x to j-th word y_j in y, and C is the transportation cost with $C_{ij} := \mathrm{dist}(\boldsymbol{v}_{x_i}, \boldsymbol{v}_{y_j})$ being the distance between two words measured in the Word2Vec embedding space. A popular choice is the Euclidean distance $\mathrm{dist}(\boldsymbol{v}_{x_i}, \boldsymbol{v}_{y_j}) = \|\boldsymbol{v}_{x_i} - \boldsymbol{v}_{y_j}\|_2$. When $\mathrm{dist}(\boldsymbol{v}_{x_i}, \boldsymbol{v}_{y_j})$ is a *metric*, the WMD distance in Eq. (1) also qualifies as a metric, and in particular, satisfies the triangle inequality (Rubner et al., 2000). Building on top of Word2Vec, WMD is a particularly useful and accurate for measure of the distance between documents with semantically close but syntactically different words as illustrated in Figure 1(a).

The WMD distance when coupled with KNN has been observed to have strong performance in classification tasks (Kusner et al., 2015). However, WMD is expensive to compute with computational complexity of $O(L^3 \log(L))$, especially for long documents where L is large. Additionally, since WMD is just a document distance, rather than a document representation, using it within KNN incurs even higher computational costs $O(N^2 L^3 \log(L))$.

3 Document Embedding via Word Mover's Kernel

In this section, we extend the framework in (Wu et al., 2018a), to derive a positive-definite kernel from an alignment-aware document distance metric, which then gives us an unsupervised semantic embeddings of texts of variable length as a by-product through the theory of *Random Feature Approximation* (Rahimi and Recht, 2007).

3.1 Word Mover's Kernel

We start by defining the *Word Mover's Kernel*:

$$\begin{aligned} k(x,y) &:= \int p(\omega)\phi_\omega(x)\phi_\omega(y)d\omega, \\ \text{where} \ \ \phi_\omega(x) &:= \exp(-\gamma \mathrm{WMD}(x,\omega)). \end{aligned} \tag{2}$$

where ω can be interpreted as a random document $\{\boldsymbol{v}_j\}_{j=1}^D$ that contains a collection of random word vectors in $\mathcal{V}$, and $p(\omega)$ is a distribution over the space of all possible random documents $\Omega := \bigcup_{D=1}^{D_{max}} \mathcal{V}^D$. $\phi_\omega(x)$ is an possibly infinite-dimensional feature map derived from the WMD between x and all possible documents $\omega \in \Omega$.

An insightful interpretation of this kernel (2):

$$k(x,y) := \exp(-\gamma \, \mathit{softmin}_{p(\omega)} f(\omega))$$

where

$$softmin_{p(\omega)}\ f(\omega) := -\frac{1}{\gamma}\log\int p(\omega)e^{-\gamma f(\omega)}d\omega,$$

and $f(\omega) = \{\text{WMD}(x,\omega) + \text{WMD}(\omega,y)\}$, is a version of soft minimum function parameterized by $p(\omega)$ and γ. Comparing this with the usual definition of soft minimum $softmin_i f_i := -softmax(-f_i) = -\log\sum_i e^{-f_i}$, it can be seen that the soft-min-variant in the above Equations uses a weighting of the objects ω via the probability density $p(\omega)$, and moreover has the additional parameter γ to control the degree of smoothness. When γ is large and $f(\omega)$ is Lipschitz-continuous, the value of the soft-min-variant is mostly determined by the minimum of $f(\omega)$.

Note that since WMD is a metric, by the triangular inequality we have

$$\text{WMD}(x,y) \ \leq \ \min_{\omega\in\Omega}\ (\text{WMD}(x,\omega) + \text{WMD}(\omega,y))$$

and the equality holds if we allow the length of random document $D_{\max}$ to be not smaller than L. Therefore, the kernel (2) serves as a good approximation to the WMD between any pair of documents x, y as illustrated in Figure 1(b), while it is *positive-definite* by the definition.

3.2 Word Mover's Embedding

Given the Word-Mover's Kernel in Eq. (2), we can then use the Monte-Carlo approximation:

$$k(x,y) \approx \langle Z(x), Z(y)\rangle = \frac{1}{R}\sum_{i=1}^{R}\phi_{\omega_i}(x)\phi_{\omega_i}(y) \tag{3}$$

where $\{\omega_i\}_{i=1}^{R}$ are i.i.d. random documents drawn from $p(\omega)$ and $Z(x) := (\frac{1}{\sqrt{R}}\phi_{\omega_i}(x))_{i=1}^{R}$ gives a vector representation of document x. We call this random approximation *Word Mover's Embedding*. Later, we show that this Random Features approximation in Eq. (3) converges to the exact kernel (2) uniformly over all pairs of documents (x, y) .

Distribution $p(\omega)$**.** A key ingredient in the Word Mover's Kernel and Embedding is the distribution $p(\omega)$ over random documents. Note that $\omega \in \mathcal{X}$ consists of sets of words, each of which lies in the Word2Vec embedding space; the characteristics of which need to be captured by $p(\omega)$ in order to generate (sets of) "meaningful" random words. Several studies have found that the word vectors $\boldsymbol{v}$ are roughly uniformly dispersed in the word embedding space (Arora et al., 2016, 2017). This is also consistent with our empirical findings, that the uniform distribution centered by the mean of all word vectors in the documents is generally applicable for various text corpora. Thus, if d is the dimensionality of the pre-trained word embedding space, we can draw a random word $u \in \mathbb{R}^d$ as $u_j \sim \text{Uniform}[v_{\min}, v_{\max}]$, for $j = 1, \ldots, d$, and where $v_{\min}$ and $v_{\max}$ are some constants.

Given a distribution over random words, the remaining ingredient is the length D of random documents. It is desirable to set these to a small number, in part because this length is indicative of the number of hidden global topics, and we expect the number of such global topics to be small. In particular, these global topics will allow short random documents to align with the documents to obtain "topic-based" discriminatory features. Since there is no prior information for global topics, we choose to uniformly sample the length of random documents as $D \sim \text{Uniform}[1, D_{\max}]$, for some constant $D_{\max}$. Stitching the distributions over words, and over the number of words, we then get a distribution over random documents. We note that our WME embedding allows potentially other random distributions, and other types of word embeddings, making it a flexible and powerful feature learning framework to utilize state-of-the-art techniques.

Algorithm 1 Word Mover's Embedding: An Unsupervised Feature Representation for Documents

```
Input: Texts $\{x_i\}_{i=1}^N$, $D_{\max}$, $R$.
Output: Matrix $Z_{N\times R}$, with rows corresponding to text embeddings.
Compute $v_{\max}$ and $v_{\min}$ as the maximum and minimum values, over all coordinates of the word vectors $\boldsymbol{v}$ of $\{x_i\}_{i=1}^N$, from any pre-trained word embeddings (e.g. Word2Vec, GloVe or PSL999).
for $j = 1, \ldots, R$ do
    Draw $D_j \sim \text{Uniform}[1, D_{\max}]$.
    Generate a random document $\omega_j$ consisting of $D_j$ number of random words drawn as $\omega_{j\ell} \sim \text{Uniform}[v_{\min}, v_{\max}]^d$, $\ell = 1, \ldots, D_j$.
    Compute $\boldsymbol{f}_{x_i}$ and $\boldsymbol{f}_{\omega_j}$ using a popular weighting scheme (e.g. NBOW or TF-IDF).
    Compute the WME feature vector $Z_j = \phi_{\omega_j}(\{x_i\}_{i=1}^N)$ using WMD in Equation (2).
end for
Return $Z(\{x_i\}_{i=1}^N) = \frac{1}{\sqrt{R}}[Z_1\ Z_2\ \ldots\ Z_R]$
```

Algorithm 1 summarizes the overall procedure to generate feature vectors for text of any length such as sentences, paragraphs, and documents.

KNN-WMD, which uses the WMD distance together with KNN based classification, requires $O(N^2)$ evaluations of the WMD distance, which in turn has $O(L^3 \log(L))$ complexity, assuming that documents have lengths bounded by L, leading to an overall complexity of $O(N^2 \, L^3 \, \log(L)$. In contrast, our WME approximation only requires super-linear complexity of $O(NRLlog(L))$ when D is constant. This is because in our case each evaluation of WMD only requires $O(D^2 \, L \log(L))$ (Bourgeois and Lassalle, 1971), due to the short length D of our random documents. This dramatic reduction in computation significantly accelerates training and testing when combined with empirical risk minimization classifiers such as SVMs. A simple yet useful trick is to pre-compute the word distances to avoid redundant computations since a pair of words may appear multiple times in different pairs of documents. Note that the computation of the ground distance between each pair of word vectors in documents has a $O(L^2 \, d)$ complexity, which could be close to one WMD evaluation if document length L is short and word vector dimension d is large. This simple scheme leads to additional improvement in runtime performance of our WME method that we show in our experiments.

3.3 Convergence of WME

In this section, we study the convergence of our embedding (3) to the exact kernel (2) under the framework of Random Features (RF) approximation (Rahimi and Recht, 2007). Note that the standard RF convergence theory applies only to the shift-invariant kernel operated on two vectors, while our kernel (2) operates on two documents $x, y \in \mathcal{X}$ that are sets of word vectors. In (Wu et al., 2018a), a general RF convergence theory is provided for any *distance-based kernel* as long as a finite covering number is given w.r.t. the given distance. In the following lemma, we provide the covering number for all documents of bounded length under the *Word Mover's Distance*. Without loss of generality, we will assume that the word embeddings $\{\boldsymbol{v}\}$ are normalized s.t. $\|\boldsymbol{v}\| \leq 1$.

Lemma 1. *There exists an ϵ-covering $\mathcal{E}$ of $\mathcal{X}$ under the WMD metric with Euclidean ground distance, so that:*

$$\forall x \in \mathcal{X}, \exists x_i \in \mathcal{E}, \; \mathit{WMD}(x, x_i) \leq \epsilon,$$

that has size bounded as $|\mathcal{E}| \leq (\frac{2}{\epsilon})^{d\,L}$, where L is a bound on the length of document $x \in \mathcal{X}$.

Equipped with Lemma 1, we can derive the following convergence result as a simple corollary of the theoretical results in (Wu et al., 2018a). We defer the proof to the appendix A.

Theorem 1. *Let $\Delta_R(x, y)$ be the difference between the exact kernel (2) and the random approximation (3) with R samples, we have uniform convergence*

$$P\left\{\max_{x,y \in \mathcal{X}} |\Delta_R(x,y)| > 2t\right\} \leq 2\left(\frac{12\gamma}{t}\right)^{2dL} e^{-Rt^2/2}.$$

where d is the dimension of word embedding and L is a bound on the document length. In other words, to guarantee $|\Delta_R(x, y)| \leq \epsilon$ with probability at least $1 - \delta$, it suffices to have

$$R = \Omega\left(\frac{dL}{\epsilon^2}\log(\frac{\gamma}{\epsilon}) + \frac{1}{\epsilon^2}\log(\frac{1}{\delta})\right).$$

4 Experiments

We conduct an extensive set of experiments to demonstrate the effectiveness and efficiency of the proposed method. We first compare its performance against 7 unsupervised document embedding approaches over a wide range of text classification tasks, including sentiment analysis, news categorization, amazon review, and recipe identification. We use 9 different document corpora, with 8 of these drawn from (Kusner et al., 2015; Huang et al., 2016); Table 1 provides statistics of the different datasets. We further compare our method against 10 unsupervised, semi-supervised, and supervised document embedding approaches on the 22 datasets from SemEval semantic textual similarity tasks. Our code is implemented in Matlab, and we use C Mex for the computationally intensive components of WMD (Rubner et al., 2000).

4.1 Effects of R and D on WME

Setup. We first perform experiments to investigate the behavior of the WME method by varying the number of Random Features R and the length D of random documents. The hyper-parameter γ is set via cross validation on training set over the range $[0.01, \; 10]$. We simply fix the $D_{min} = 1$, and vary D_{max} over the range $[3, \; 21]$. Due to limited space, we only show selected subsets of our results, with the rest listed in the Appendix B.2.

Table 1: Properties of the datasets

Dataset	C:Classes	N:Train	M:Test	BOW Dim	L:Length	Application
BBCSPORT	5	517	220	13243	117	BBC sports article labeled by sport
TWITTER	3	2176	932	6344	9.9	tweets categorized by sentiment
RECIPE	15	3059	1311	5708	48.5	recipe procedures labeled by origin
OHSUMED	10	3999	5153	31789	59.2	medical abstracts (class subsampled)
CLASSIC	4	4965	2128	24277	38.6	academic papers labeled by publisher
REUTERS	8	5485	2189	22425	37.1	news dataset (train/test split)
AMAZON	4	5600	2400	42063	45.0	amazon reviews labeled by product
20NEWS	20	11293	7528	29671	72	canonical user-written posts dataset
RECIPE_L	20	27841	11933	3590	18.5	recipe procedures labeled by origin

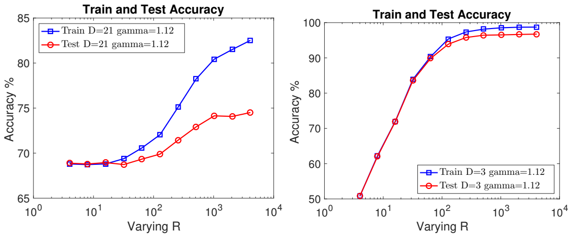

(a) TWITTER (b) CLASSIC

Figure 2: Train (Blue) and Test (Red) accuracy when varying R with fixed D.

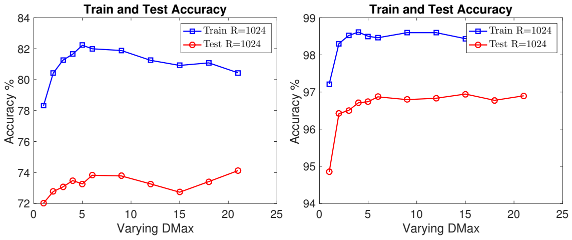

(a) TWITTER (b) CLASSIC

Figure 3: Train (Blue) and Test (Red) accuracy when varying D with fixed R.

Effects of R. We investigate how the performance changes when varying the number of Random Features R from 4 to 4096 with fixed D. Fig. 2 shows that both training and testing accuracies generally converge very fast when increasing R from a small number ($R = 4$) to a relatively large number ($R = 1024$), and then gradually reach to the optimal performance. This confirms our analysis in Theory 1 that the proposed WME can guarantee the fast convergence to the exact kernel.

Effects of D. We further evaluate the training and testing accuracies when varying the length of random document D with fixed R. As shown in Fig. 3, we can see that near-peak performance can usually be achieved when D is small (typically $D \leq 6$). This behavior illustrates two important aspects: (1) using very few random words (e.g. $D = 1$) is not enough to generate useful Random Features when R becomes large; (2) using too many random words (e.g. $D \geq 10$) tends to generate similar and redundant Random Features when increasing R. Conceptually, the number of random words in a random document can be thought of as the number of the global topics in documents, which is generally small. This is an important desired feature that confers both a performance boost as well as computational efficiency to the WME method.

4.2 Comparison with KNN-WMD

Baselines. We now compare two WMD-based methods in terms of testing accuracy and total training and testing runtime. We consider two variants of WME with different sizes of R. WME(LR) stands for WME with large rank that achieves the best accuracy (using R up to 4096) with more computational time, while WME(SR) stands for WME with small rank that obtains comparable accuracy in less time. We also consider two variants of both methods where +P denotes that we precompute the ground distance between each pair of words to avoid redundant computations.

Setup. Following (Kusner et al., 2015; Huang et al., 2016), for datasets that do not have a predefined train/test split, we report average and standard deviation of the testing accuracy and average run-time of the methods over five 70/30 train/test splits. For WMD, we provide the results (with respect to accuracy) from (Kusner et al., 2015); we also reran the experiments of KNN-WMD and found them to be consistent with the reported results. For all methods, we perform 10-fold cross validation to search for the best parameters on the training documents. We employ a linear SVM implemented using LIBLINEAR (Fan et al., 2008) on WME since it can isolate the effectiveness of the feature representation from the power of the nonlinear learning solvers. For additional results on all KNN-based methods, please refer to Appendix B.3.

Results. Table 2 corroborates the significant advan-

Table 2: Test accuracy, and total training and testing time (in seconds) of WME against KNN-WMD. Speedups are computed between the best numbers of KNN-WMD+P and these of WME(SR)+P when achieving similar testing accuracy. Bold face highlights the best number for each dataset.

Classifier	KNN-WMD		KNN-WMD+P	WME(SR)		WME(SR)+P	WME(LR)		WME(LR)+P	
Dataset	Accu	Time	Time	Accu	Time	Time	Accu	Time	Time	Speedup
BBCSPORT	95.4 ± 1.2	147	122	95.5 ± 0.7	3	1	**98.2 ± 0.6**	92	34	**122**
TWITTER	71.3 ± 0.6	25	4	72.5 ± 0.5	10	2	**74.5 ± 0.5**	162	34	**2**
RECIPE	57.4 ± 0.3	448	326	57.4 ± 0.5	18	4	**61.8 ± 0.8**	277	61	**82**
OHSUMED	55.5	3530	2807	55.8	24	7	**64.5**	757	240	**401**
CLASSIC	**97.2 ± 0.1**	777	520	96.6 ± 0.2	49	10	97.1 ± 0.4	388	70	**52**
REUTERS	96.5	814	557	96.0	50	24	**97.2**	823	396	**23**
AMAZON	92.6 ± 0.3	2190	1319	92.7 ± 0.3	31	8	**94.3 ± 0.4**	495	123	**165**
20NEWS	73.2	37988	32610	72.9	205	69	**78.3**	1620	547	**472**
RECIPE_L	71.4 ± 0.5	5942	2060	72.5 ± 0.4	113	20	**79.2 ± 0.3**	1838	330	**103**

Table 3: Testing accuracy of WME against Word2Vec and Doc2Vec-based methods.

Dataset	SIF(GloVe)	Word2Vec+nbow	Word2Vec+tf-idf	PV-DBOW	PV-DM	Doc2VecC	WME
BBCSPORT	97.3 ± 1.2	97.3 ± 0.9	96.9 ± 1.1	97.2 ± 0.7	97.9 ± 1.3	90.5 ± 1.7	**98.2 ± 0.6**
TWITTER	57.8 ± 2.5	72.0 ± 1.5	71.9 ± 0.7	67.8 ± 0.4	67.3 ± 0.3	71.0 ± 0.4	**74.5 ± 0.5**
OHSUMED	**67.1**	63.0	60.6	55.9	59.8	63.4	64.5
CLASSIC	92.7 ± 0.9	95.2 ± 0.4	93.9± 0.4	97.0 ± 0.3	96.5 ± 0.7	96.6 ± 0.4	**97.1 ± 0.4**
REUTERS	87.6	96.9	95.9	96.3	94.9	96.5	**97.2**
AMAZON	94.1 ± 0.2	94.0 ± 0.5	92.2 ± 0.4	89.2 ± 0.3	88.6 ± 0.4	91.2 ± 0.5	**94.3 ± 0.4**
20NEWS	72.3	71.7	70.2	71.0	74.0	78.2	**78.3**
RECIPE_L	71.1 ± 0.5	74.9 ± 0.5	73.1 ± 0.6	73.1 ± 0.5	71.1 ± 0.4	76.1 ± 0.4	**79.2 ± 0.3**

tages of WME compared to KNN-WMD in terms of both accuracy and runtime. First, WME(SR) can consistently achieve better or similar accuracy compared to KNN-WMD while requiring order-of-magnitude less computational time on all datasets. Second, both methods can benefit from precomputation of the ground distance between a pair of words but WME gains much more from prefetch (typically 3-5x speedup). This is because the typical length D of random documents is very short where computing ground distance between word vectors may be even more expensive than the corresponding WMD distance. Finally, WME(LR) can achieve much higher accuracy compared to KNN-WMD while still often requiring less computational time, especially on large datasets like 20NEWS and relatively long documents like OHSUMED.

4.3 Comparisons with Word2Vec & Doc2Vec

Baselines. We compare against 6 document representations methods: 1) *Smooth Inverse Frequency* (SIF) (Arora et al., 2017): a recently proposed simple but tough to beat baseline for sentence embeddings, combining a new weighted scheme of word embeddings with dominant component removal; 2) *Word2Vec+nbow*: a weighted average of word vectors using NBOW weights; 3) *Word2Vec+tf-idf*: a weighted average of word vectors using TF-IDF weights; 4) *PV-DBOW* (Le and Mikolov, 2014): distributed bag of words model of Paragraph Vectors; 5) *PV-DM* (Le and Mikolov, 2014): distributed memory model of Paragraph Vectors; 6) *Doc2VecC* (Chen, 2017): a recently proposed document-embedding via corruptions, achieving state-of-the-art performance in text classification.

Setup. *Word2Vec+nbow*, *Word2Vec+tf-idf* and WME use pre-trained Word2Vec embeddings while SIF uses its default pre-trained GloVe embeddings. Following (Chen, 2017), to enhance the performance of *PV-DBOW*, *PV-DM*, and *Doc2VecC* these methods are trained transductively on both train and test, which is indeed beneficial for generating a better document representation (see Appendix B.4). In contrast, the hyperparameters of WME are obtained through a 10-fold cross validation only on training set. For a fair comparison, we run a linear SVM using LIBLINEAR on all methods.

Results. Table 3 shows that WME consistently outperforms or matches existing state-of-the-art document representation methods in terms of testing accuracy on all datasets except one (OHSUMED). The first highlight is that simple average of word embeddings often achieves better performance than *SIF(Glove)*, indicating that removing the first principle component could hurt the expressive power of the resulting representation for some of classification tasks. Surprisingly, these two methods often achieve similar or better performance than *PV-DBOW* and *PV-DM*, which may be because of the high-quality pre-trained word embeddings. On

Table 4: Pearson's scores of WME against other unsupervised, semi-supervised, and supervised methods on 22 textual similarity tasks. Results are collected from (Arora et al., 2017) except our approach.

Approaches	Supervised						Unsupervised					Semi-supervised	
WordEmbeddings	PSL						GloVe					PSL	
Tasks	PP	Dan	RNN	iRNN	LSTM(no)	LSTM(o.g.)	ST	nbow	tf-idf	SIF	WME	SIF	WME
STS'12	58.7	56.0	48.1	58.4	51.0	46.4	30.8	52.5	58.7	56.2	60.6	59.5	**62.8**
STS'13	55.8	54.2	44.7	56.7	45.2	41.5	24.8	42.3	52.1	56.6	54.5	**61.8**	56.3
STS'14	70.9	69.5	57.7	70.9	59.8	51.5	31.4	54.2	63.8	68.5	65.5	**73.5**	68.0
STS'15	75.8	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.6	71.7	61.8	**76.3**	64.2
SICK'14	71.6	70.7	61.2	71.2	63.9	59.0	49.8	65.9	69.4	72.2	68.0	**72.9**	68.1
Twitter'15	52.9	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	33.8	48.0	41.6	49.0	47.4

the other hand, *Doc2VecC* achieves much better testing accuracy than these previous methods on two datasets (20NEWS, and RECIPE_L). This is mainly because that it benefits significantly from transductive training (See Appendix B.4). Finally, the better performance of WME over these strong baselines stems from fact that WME is empowered by two important building blocks, WMD and Word2Vec, to yield a more informative representation of the documents by considering both the word alignments and the semantics of words.

We refer the readers to additional results on the Imdb dataset in Appendix B.4, which also demonstrate the clear advantage of WME even compared to the supervised RNN method as well as the aforementioned baselines.

4.4 Comparisons on textual similarity tasks

Baselines. We compare WME against 10 supervised, simi-sepervised, and unsupervised methods for performing textual similarity tasks. Six supervised methods are initialized with Paragram-SL999(PSL) word vectors (Wieting et al., 2015b) and then trained on the PPDB dataset, including: 1) *PARAGRAM-PHRASE* (PP) (Wieting et al., 2015a): simple average of refined PSL word vectors; 2) *Deep Averaging Network* (DAN) (Iyyer et al., 2015); 3) *RNN*: classical recurrent neural network; 4) *iRNN*: a variant of RNN with the activation being the identify; 5) *LSTM(no)* (Gers et al., 2002): LSTM with no output gates; 6) *LSTM(o.g.)* (Gers et al., 2002): LSTM with output gates. Four unsupervised methods are: 1) *Skip-Thought Vectors* (ST) (Kiros et al., 2015): an encoder-decoder RNN model for generalizing Skip-gram to the sentence level; 2) *nbow*: simple averaging of pre-trained GloVe word vectors; 3) *tf-idf*: a weighted average of GloVe word vecors using TF-IDF weights; 4) *SIF* (Arora et al., 2017): a simple yet strong method on textual similarity tasks using GloVe word vecors. Two semi-supervised methods use PSL word vectors, which are trained using labeled data (Wieting et al., 2015b).

Setup. There are total 22 textual similarity datasets from STS tasks (2012-2015) (Agirre et al., 2012, 2013, 2014, 2015), SemEval 2014 Semantic Relatedness task (Xu et al., 2015), and SemEval 2015 Twitter task (Marelli et al., 2014). The goal of these tasks is to predict the similarity between two input sentences. Each year STS usually has 4 to 6 different tasks and we only report the averaged Pearson's scores for clarity. Detailed results on each dataset are listed in Appendix B.5.

Results. Table 4 shows that WME consistently matches or outperforms other unsupervised and supervised methods except the *SIF* method. Indeed, compared with *ST* and *nbow*, WME improves Pearson's scores substantially by 10% to 33% as a result of the consideration of word alignments and the use of TF-IDF weighting scheme. *tf-idf* also improves over these two methods but is slightly worse than our method, indicating the importance of taking into account the alignments between the words. *SIF* method is a strong baseline for textual similarity tasks but WME still can beat it on STS'12 and achieve close performance in other cases. Interestingly, WME is on a par with three supervised methods *RNN*, *LSTM(no)*, and *LSTM(o.g.)* in most cases. The final remarks stem from the fact that, WME can gain significantly benefit from the supervised word embeddings similar to *SIF*, both showing strong performance on PSL.

5 Related Work

Two broad classes of *unsupervised* and *supervised* methods have been proposed to generate sentence and document representations. The former primarily generate general purpose and domain independent embeddings of word sequences (Socher et al., 2011; Kiros et al., 2015; Arora et al., 2017); many unsupervised training research efforts have focused on either training an auto-encoder to learn the latent structure of a sentence (Socher et al., 2013), a paragraph, or document (Li et al., 2015); or generalizing Word2Vec models to predict words in a paragraph (Le and Mikolov, 2014; Chen, 2017) or in neighboring sentences (Kiros et al., 2015). However, some important information could be lost in the resulting document representation without considering the word order. Our proposed WME overcomes this difficulty by considering the alignments between each pair of words.

The other line of work has focused on developing compositional supervised models to create a vector representation of sentences (Kim et al., 2016; Gong et al., 2018b). Most of this work proposed composition using recursive neural networks based on parse structure (Socher et al., 2012, 2013), deep averaging networks over bag-of-words models (Iyyer et al., 2015; Wieting et al., 2015a), convolutional neural networks (Kim, 2014; Kalchbrenner et al., 2014; Xu et al., 2018), and recurrent neural networks using long short-term memory (Tai et al., 2015; Liu et al., 2015). However, these methods are less well suited for domain adaptation settings.

6 Conclusion

In this paper, we have proposed an alignment-aware text kernel using WMD for texts of variable lengths, which takes into account both word alignments and pre-trained high quality word embeddings in learning an effective semantics-preserving feature representation. The proposed WME is simple, efficient, flexible, and unsupervised. Extensive experiments show that WME consistently matches or outperforms state-of-the-art models on various text classification and textual similarity tasks. WME embeddings can be easily used for a wide range of downstream supervised and unsupervised tasks.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *SemEval@ NAACL-HLT*, pages 252–263.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *SemEval@ COLING*, pages 81–91.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In *In* SEM 2013: The Second Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics*. Citeseer.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics.

Jason Altschuler, Jonathan Weed, and Philippe Rigollet. 2017. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In *Advances in Neural Information Processing Systems*, pages 1964–1974.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

François Bourgeois and Jean-Claude Lassalle. 1971. An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12):802–804.

Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. 1995. Automatic query expansion using smart: Trec 3. *NIST special publication sp*, pages 69–69.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. In *ICLR*.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *Proceedings of the 29th international conference on Machine learning*.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Hongyu Gong, Suma Bhat, and Pramod Viswanath. 2018a. Embedding syntax and semantics of prepositions via tensor decomposition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 896–906.

Hongyu Gong, Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. Prepositions in context. *arXiv preprint arXiv:1702.01466*.

Hongyu Gong, Tarek Sakakini, Suma Bhat, and JinJun Xiong. 2018b. Document similarity for texts of varying lengths via hidden topics. In *ACL*, volume 1, pages 2341–2351.

Tom Griffiths and Mark Steyvers. 2007. Probabilistic topic models. *Latent Semantic Analysis: A Road to Meaning*.

Jie Gui, Tongliang Liu, Dacheng Tao, Zhenan Sun, and Tieniu Tan. 2016. Representative vector machines: a unified framework for classical classifiers. *IEEE transactions on cybernetics*, 46(8):1877–1888.

Jie Gui, Zhenan Sun, Jun Cheng, Shuiwang Ji, and Xindong Wu. 2014. How to estimate the regularization parameter for spectral regression discriminant analysis and its kernel version? *IEEE Transactions on Circuits and Systems for Video Technology*, 24(2):211–223.

Bernard Haasdonk and Claus Bahlmann. 2004. Learning with distance substitution kernels. In *Joint Pattern Recognition Symposium*, pages 220–227. Springer.

Frank L Hitchcock. 1941. The distribution of a product from several sources to numerous localities. *Studies in Applied Mathematics*, 20(1-4):224–230.

Gao Huang, Chuan Guo, Matt J Kusner, Yu Sun, Fei Sha, and Kilian Q Weinberger. 2016. Supervised word mover's distance. In *Advances in Neural Information Processing Systems*, pages 4862–4870.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2326–2335.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *SemEval@ COLING*, pages 1–8.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Xi Peng, Rogerio S Feris, Xiaoyu Wang, and Dimitris N Metaxas. 2016. A recurrent encoder-decoder network for sequential face alignment. In *European conference on computer vision*, pages 38–56. Springer, Cham.

Xi Peng, Shaoting Zhang, Yu Yang, and Dimitris N Metaxas. 2015. Piefa: Personalized incremental and ensemble face alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 3880–3888.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Hieu Pham, Minh-Thang Luong, and Christopher D Manning. 2015. Learning distributed representations for multilingual text sequences. In *Proceedings of NAACL-HLT*, pages 88–94.

Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, page 5.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *ACM SIGIR conference on Research and development in information retrieval*.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems*, pages 801–809.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015a. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015b. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL (TACL)*.

Lingfei Wu, Eloy Romero, and Andreas Stathopoulos. 2017. Primme_svds: A high-performance preconditioned svd solver for accurate large-scale computations. *SIAM Journal on Scientific Computing*, 39(5):S248–S271.

Lingfei Wu and Andreas Stathopoulos. 2015. A preconditioned hybrid svd method for accurately computing singular triplets of large matrices. *SIAM Journal on Scientific Computing*, 37(5):S365–S388.

Lingfei Wu, Ian En-Hsu Yen, Fnagli Xu, Pradeep Ravikumar, and Witbrock Michael. 2018a. D2ke: From distance to kernel and embedding. *https://arxiv.org/abs/1802.04956*.

Lingfei Wu, Ian En-Hsu Yen, Jinfeng Yi, Fangli Xu, Qi Lei, and Michael Witbrock. 2018b. Random warping series: A random features method for time-series embedding. In *International Conference on Artificial Intelligence and Statistics*, pages 793–802.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Wei Xu, Chris Callison-Burch, and Bill Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). In *SemEval@ NAACL-HLT*, pages 1–11.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state lstm for text representation. *arXiv preprint arXiv:1805.02474*.

Multilingual Clustering of Streaming News

Sebastião Miranda[◇] **Artūrs Znotiņš**[†⋆] **Shay B. Cohen**[♡] **Guntis Barzdins**[†⋆]
[◇]Priberam Labs, Alameda D. Afonso Henriques, 41, 2°, 1000-123 Lisboa, Portugal
[†]Innovation Labs LETA, Marijas Str. 2, Riga LV-1050, Latvia
[⋆]University of Latvia, IMCS, Rainis Blvd. 29, Riga LV-1459, Latvia
[♡]School of Informatics, University of Edinburgh, Edinburgh EH8 9AB
`sebastiao.miranda@priberam.pt`, `arturs.znotins@leta.lv`,
`scohen@inf.ed.ac.uk`, `guntis.barzdins@lu.lv`

Abstract

Clustering news across languages enables efficient media monitoring by aggregating articles from multilingual sources into coherent stories. Doing so in an online setting allows scalable processing of massive news streams. To this end, we describe a novel method for clustering an incoming stream of multilingual documents into monolingual and crosslingual story clusters. Unlike typical clustering approaches that consider a small and known number of labels, we tackle the problem of discovering an ever growing number of cluster labels in an online fashion, using real news datasets in multiple languages. Our method is simple to implement, computationally efficient and produces state-of-the-art results on datasets in German, English and Spanish.

1 Introduction

Following developing news stories is imperative to making real-time decisions on important political and public safety matters. Given the abundance of media providers and languages, this endeavor is an extremely difficult task. As such, there is a strong demand for automatic clustering of news streams, so that they can be organized into stories or themes for further processing. Performing this task in an online and efficient manner is a challenging problem, not only for newswire, but also for scientific articles, online reviews, forum posts, blogs, and microblogs.

A key challenge in handling document streams is that the story clusters must be generated on the fly in an online fashion: this requires handling documents one-by-one as they appear in the document stream. In this paper, we provide a treatment to the problem of *online document clustering*, i.e. the task of clustering a stream of documents into themes. For example, for news articles, we would want to cluster them into related news stories.

To this end, we introduce a system which aggregates news articles into fine-grained story clusters across different languages in a completely online and scalable fashion from a continuous stream. Our clustering approach is part of a larger *media monitoring* project to solve the problem of monitoring massive text and TV/Radio streams (speech-to-text). In particular, media monitors write intelligence reports about the most relevant events, and being able to search, visualize and explore news clusters assists in gathering more insight about a particular story. Since relevant events may be spawned from any part of the world (and from many multilingual sources), it becomes imperative to cluster news across different languages.

In terms of granularity, the type of story clusters we are interested in are the group of articles which, for example : (i) Narrate recent air-strikes in Eastern Ghouta (Syria); (ii) Describe the recent launch of Space X's Falcon Heavy rocket.

Main Contributions While most existing news clustering approaches assume a monolingual document stream – a non-realistic scenario given the diversity of languages on the Web – we assume a general, multilingual, document stream. This means that in our problem-formulation story documents appear in multiple languages and we need to cluster them to crosslingual clusters. Our main contributions are as follows:

- We develop a system that aggregates news articles into fine-grained story clusters across different languages in a completely online and scalable fashion from a continuous stream. As discussed in the introduction, this is a highly relevant task for the use-case of media monitoring.
- We formulate the problem of online multilingual document clustering and the representation that such clustering takes by interlacing the problem

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4535–4544
Brussels, Belgium, October 31 - November 4, 2018.

of monolingual clustering with crosslingual clustering. The representation of our clusters is interpretable, and similarly to topic models, consists of a set of keywords and weights associated with the relevant cluster. In our formulation, a monolingual cluster is a group of documents, and a crosslingual cluster is a group of monolingual clusters in different languages.

- We compare our approach to our own implementation of a state-of-the-art streaming method, and show much superior results for a dataset in English, Spanish and German.

2 Problem Formulation

We focus on clustering of a stream of documents, where the number of clusters is not fixed and learned automatically. We denote by D a (potentially infinite) space of multilingual documents. Each document d is associated with a language in which it is written through a function $L\colon D \to \mathcal{L}$ where $\mathcal{L}$ is a set of languages. For example, $L(d)$ could return English, Spanish or German. (In the rest of the paper, for an integer n, we denote by $[n]$ the set $\{1, \ldots, n\}$.)

We are interested in associating each document with a monolingual cluster via the function $C(d) \in \mathbb{N}$, which returns the cluster label given a document. This is done independently for each language, such that the space of indices we use for each language is separate. Furthermore, we interlace the problem of monolingual clustering with crosslingual clustering. This means that as part of our problem formulation we are also interested in a function $E\colon \mathbb{N} \times \mathcal{L} \to \mathbb{N}$ that associates each monolingual cluster with a crosslingual cluster, such that each crosslingual cluster only groups one monolingual cluster per different language, at a given time. The crosslingual cluster for a document d is $E(C(d), L(d))$. As such, a crosslingual cluster groups together monolingual clusters, at most one for each different language.

Intuitively, building both monolingual and crosslingual clusters allows the system to leverage high-precision monolingual features (e.g., words, named entities) to cluster documents of the same language, while simplifying the task of crosslingual clustering to the computation of similarity scores across monolingual clusters - which is a smaller problem space, since there are (by definition) less clusters than articles. We validate this choice in §5.

3 The Clustering Algorithm

Each document d is represented by two vectors in $\mathbb{R}^{k_1}$ and $\mathbb{R}^{k_2}$. The first vector exists in a "monolingual space" (of dimensionality k_1) and is based on a bag-of-words representation of the document. The second vector exists in a "crosslingual space" (of dimensionality k_2) which is common to all languages. More details about these representations are discussed in §4.

Online Clustering With our clustering algorithm, we maintain two types of centroid functions for each monolingual cluster. The first is a centroid function $H\colon \mathbb{N} \times \mathcal{L} \to \mathbb{R}^{k_1} \cup \{\perp\}$ that assists in associating each document with a monolingual cluster. The second is a centroid function $G\colon \mathbb{N} \to \mathbb{R}^{k_2} \cup \{\perp\}$ that assists in associating each monolingual cluster with a crosslingual cluster. The $\perp$ symbol is reserved to denote documents which are not associated with any cluster yet.

In our algorithm, we need to incrementally construct the functions H, G (the two centroid functions), C (the monolingual clustering function) and E (the crosslingual clustering function). Informally, we do so by first identifying a monolingual cluster for an incoming document by finding the closest centroid with the function H, and then associate that monolingual cluster with the crosslingual cluster that is closest based on the function G. The first update changes C and the second update changes E. Once we do that, we also update H and G to reflect the new information that exists in the new incoming document.

Example Figure 1 depicts the algorithm and the state it maintains. A document in some language (d_9) appears in the stream, and is clustered into one of the monolingual clusters (circles) that group together documents about the same story (for example, $\langle c_2, \mathrm{DE}\rangle$ could be a German cluster about a recent political event). Then, following this monolingual update, the online clustering algorithm updates the crosslingual clusters (round rectangles), each grouping together a set of monolingual clusters, one per language at the most. The centroids for the monolingual clusters are maintained by the function H. For example, $H(2, \mathrm{English})$ gives the centroid of the upper left English monolingual cluster. The function G maintains the crosslingual clusters. Considering the upper-left most crosslingual cluster, a_1, then $G(1)$ returns its centroid.

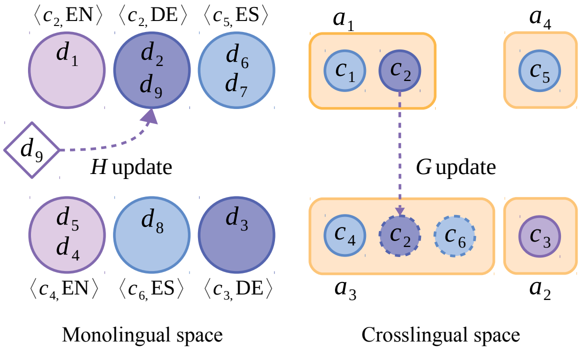

Figure 1: A pictorial description of the algorithm and the state it maintains. The algorithm maintains a monolingual cluster space, in which each cluster is a set of documents in a specific language. The algorithm also maintains a crosslingual cluster space, in which a cluster is a set of monolingual clusters in different languages. Documents are denoted by d_i, monolingual clusters by c_i (circles) and crosslingual clusters by a_i.

Algorithm To be more precise, the online clustering process works as follows. H and G start with just returning $\perp$ for any cluster number, both monolingual and crosslingual. With a new incoming document d, represented as a vector, we compute a similarity metric $\Gamma_0 \colon \mathbb{R}^{k_1} \times \mathbb{R}^{k_1} \rightarrow \mathbb{R}$ between the document vector and each of the existing centroids $\{i \mid H(i, L(d)) \neq \perp\}$. If the largest similarity exceeds a threshold τ for cluster index j, then we set $C(d) = j$. In that case, we also update the value of $H(i, L(d))$ to include new information from document d, as detailed below under "H update." If none of the similarity values exceed a threshold τ, we find the first i such that $H(i, L(d)) = \perp$ (the first cluster id which is still unassigned), and set $C(d) = i$, therefore creating a new cluster. We again follow an "H update" – this time for starting a new cluster.

In both cases, we also update the function G, by selecting the best crosslingual cluster for the recently updated (or created) monolingual cluster. To this end, we use another similarity metric $\Gamma_1 \colon \mathbb{R}^{k_2} \times \mathbb{R}^{k_2} \rightarrow \mathbb{R}$. Accordingly, we compute the similarity (using Γ_1) between the updated (or created) monolingual cluster and all monolingual clusters in each candidate crosslingual cluster, in the crosslingual feature space. The crosslingual cluster with highest sum of similarity scores is then selected. We also experimented computing this similarity by considering just the monolingual cluster of a particular "pivot language". The pivot language is a language that serves as the main indicator for a given crosslingual cluster. In our experiments, we mostly use English as the pivot language.

H Update To update H, we maintain a centroid for each cluster that is created as the average of all monolingual representations of documents that belong to that cluster. This is done for each language separately. This update can be done in $O(k_1)$ time in each step. Similarly, the update of G can be done in $O(k_2)$ time. In principle, we consider an "infinite" stream of documents, which means the number of documents in each cluster can be large. As such, for efficiency purposes, updates to H are immutable, which means that when a document is assigned to a monolingual cluster, that assignment is never changed.

G Update As described, updates to function G result in associating a monolingual cluster with a crosslingual cluster (and consequently, other monolingual clusters). Therefore, errors committed in updating G are of a higher magnitude than those committed in H, since they involve groups of documents. We also note that the best crosslingual cluster for a particular monolingual cluster might not be found right at the beginning of the process. We experiment with two types of updates to G. One which is immutable, in which changes to G are not reversed (and are described above), and one in which we introduce a novel technique to make a sequence of changes to G if necessary, as a mechanism to self-correct past clustering decisions. When a past decision is modified, it may result in a chaining of consequent modifications ("toppling of dominoes") which need to be evaluated. We coin this method "domino-toppling".

The motivation behind this technique is the change in news stories over time. The technique allows the method to modify past crosslingual clustering decisions and enables higher quality clustering results. When a past decision is modified, it may result in a chain of consequent modifications which need to be evaluated.

Our method of "domino-toppling" works by making (potentially sequences) of changes to previous clustering decisions for the crosslingual clusters, at each step placing a residual monolingual cluster in a crosslingual cluster that is most similar to it. Figure 2 gives the pseudocode for domino toppling.

This "domino-toppling" technique could have in principle a quadratic complexity in the number

```
Inputs: A monolingual cluster $c$ and a list of pairs $\langle a_j, \Gamma_1(c, a_j)\rangle$, $j \in [N]$.

Algorithm:
  • For all pairs $\langle a_j, \Gamma_1(c, a_j)\rangle$, $j \in [N]$, ordered by the second coordinate:
      • If $L(c)$ is not in $a_j$, add $c$ to $a_j$ and break.
      • Otherwise, let $y \leftarrow M(a_j, L(c))$. If $\Gamma_1(c, a_j) > \Gamma_1(y, a_j)$ then:
          • Add $c$ to $a_j$, remove $y$ from $a_j$ and call domino toppling with $y$ playing the role of $c$ and break.
  • If $c$ is left unassigned, create $a_{N+1}$ and add $c$ to it.
```

Figure 2: Crosslingual "domino-toppling". a_j is the jth crosslingual cluster (out of total N clusters) and Γ_1 is the similarity between them as in §4. $L(c)$ is the language for cluster c. $M(a, \ell)$ returns the monolingual cluster for language $\ell \in \mathcal{L}$ in crosslingual cluster a. See text for details.

of crosslingual clusters. However, we have verified that in practice it converges very fast, and in our evaluation dataset only 1% of the crosslingual updates result in topples. We apply this technique only to update G (and not H) because reversing cluster assignments in G can be done much more efficiently than in H – the total number of monolingual clusters (the clustered elements in G) is significantly smaller than the number of documents (the clustered elements in H). Crosslingual clustering is also a harder problem, which motivated the additional effort of developing this algorithm.

4 Document Representation

In this section, we give more details about the way we construct the document representations in the monolingual and crosslingual spaces. In particular, we introduce the definition of the similarity functions Γ_0 and Γ_1 that were referred in §3.

Monolingual Representation The monolingual representation for each document d in language $L(d)$ is a vector in $\mathbb{R}^{k_1}$ constructed from several TF-IDF subvectors with words, word lemmas and named entities. Each subvector is repeated for different sections of the document, the title, the body and both of them together. Besides these text fields and document timestamps, no other metadata was used. To detect named entities, we used Priberam's Text Analysis (Amaral et al., 2008) for English and Spanish, and Turbo Parser (Martins et al., 2013) for German. The extracted entities consist of people, organizations, places and other types.

Crosslingual Representation In the crosslingual space, a document representation is a vector in $\mathbb{R}^{k_2}$. Let $e(d, i)$ be a crosslingual embedding of word i in the document d, which is a vector of length m . Then the document representation $v(d)$ of d consists of subvectors of the form $v(d) = \sum_{i=1}^{n} t_i e(d, i)$, where t_i is the TF-IDF score of the ith word in the relevant section of the document (title, body or both). As detailed further in §5 we compute IDF values from a large pretraining dataset. Furthermore, for both the monolingual and crosslingual cases, we also experiment with using document timestamp features, as explained in §4.1. We use a new set of diverse timestamp features in addition to the simple absolute difference (in hours) between timestamps used by Rupnik et al. (2016).

4.1 Similarity Metrics

Our similarity metric computes weighted cosine similarity on the different subvectors, both in the case of monolingual clustering and crosslingual clustering. Formally, for the monolingual case, the similarity is given by a function defined as:

$$\Gamma_0(d_j, c_l) = \sum_{i=1}^{K} \phi_i(d_j, c_l) \cdot q_i^0 + \sum_{i=1}^{3} \gamma_i(d_j, c_l) \cdot q_i^1. \tag{1}$$

and is computed on the TF-IDF subvectors where K is the number of subvectors for the relevant document representation. For the crosslingual case, we discuss below the function Γ_1, which has a similar structure.

Here, d_j is the jth document in the stream and c_l is a monolingual cluster. The function $\phi_i(d_j, c_l)$ returns the cosine similarity between the document representation of the jth document and the centroid for cluster c_l. The vector q^0 denotes the weights through which each of the cosine similarity values for each subvectors are weighted, whereas q^1 denotes the weights for the timestamp features, as detailed further. Details on learning the weights q^0 and q^1 are discussed in §4.2.

The function $\gamma(d, c)$ that maps a pair of document and cluster to $\mathbb{R}^3$ is defined as follows. Let

$$f(t) = \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right) \tag{2}$$

for a given μ and $\sigma > 0$. For each document d and cluster c, we generate the following three-dimensional vector $\gamma(d, c) = (s_1, s_2, s_3)$:

- $s_1 = f(t(d) - n_1(c))$ where $t(d_j)$ is the timestamp for document d and $n_1(c)$ is the timestamp for the newest document in cluster c.
- $s_2 = f(t(d) - n_2(c))$ where $n_2(c)$ is the average timestamp for all documents in cluster c.
- $s_3 = f(t(d) - n_3(c))$ where $n_3(c)$ is the timestamp for the oldest document in cluster c.

These three timestamp[1] features model the time aspect of the online stream of news data and help disambiguate clustering decisions, since time is a valuable indicator that a news story has changed, even if a cluster representation has a reasonable match in the textual features with the incoming document. The same way a news story becomes popular and fades over time (Lerman and Hogg, 2010), we model the probability of a document belonging to a cluster (in terms of timestamp difference) with a probability distribution.

For the case of crosslingual clustering, we introduce Γ_1, which has a similar definition to Γ_0, only instead of passing document/cluster similarity feature vectors, we pass cluster/cluster similarities, across all language pairs. Furthermore, the features are the crosslingual embedding vectors of the sections title, body and both combined (similarly to the monolingual case) and the timestamp features. For denoting the cluster timestamp, we use the average timestamps of all articles in it.

4.2 Learning to Rank Candidates

In §4.1 we introduced q^0 and q^1 as the weight vectors for the several document representation features. We experiment with both setting these weights to just 1 ($q^0_i = 1\,\forall i$ and $q^1_j = 1\,\forall j \in [3]$) and also learning these weights using support vector machines (SVMs). To generate the SVM training data, we simulate the execution of the algorithm on a training data partition (which we do not get evaluated on) and in which the gold standard labels are given. We run the algorithm using only the first subvector $\phi_1(d_j, c_l)$, which is the TF-IDF vector with the words of the document in the body and title. For each incoming document, we create a collection of positive examples, for the document and the clusters which share at least one document in the gold labeling. We then generate 20 negative examples for the document from the 20 best-matching clusters which are not correct. To find out the best-matching clusters, we rank them according to their similarity to the input document using only the first subvector $\phi_1(d_j, c_l)$.

Using this scheme we generate a collection of ranking examples (one for each document in the dataset, with the ranking of the best cluster matches), which are then trained using the SVM-Rank algorithm (Joachims, 2002). We run 5-fold cross-validation on this data to select the best model, and train both a separate model for each language according to Γ_0 and a crosslingual model according to Γ_1.

5 Experiments

Our system was designed to cluster documents from a (potentially infinite) real-word data stream. The datasets typically used in the literature (TDT, Reuters) have a small number of clusters (≈ 20) with coarse topics (economy, society, etc.), and therefore are not relevant to the use case of media monitoring we treat - as it requires much more fine-grained story clusters about particular events. To evaluate our approach, we adapted a dataset constructed for the different purpose of binary classification of joining cluster pairs.[2] We processed it to become a collection of articles annotated with monolingual and crosslingual cluster labels.[3]

Statistics about this dataset are given in Table 1. As described further, we tune the hyper-parameter τ on the development set. As for the hyper-parameters related to the timestamp features, we fixed $\mu = 0$ and tuned σ on the development set, yielding $\sigma = 72$ hours (3 days).[4] To compute IDF scores (which are global numbers computed across a corpus), we used a different and much larger dataset that we collected from Deutsche Welle's news website (`http://www.dw.com/`). The dataset consists of 77,268, 118,045 and 134,243 documents for Spanish, English and German, respectively.

The conclusions from our experiments are: (a) the weighting of the similarity metric features using SVM significantly outperforms unsupervised baselines such as CluStream (Table 2); (b) the SVM approach significantly helps to learn when to create a new cluster, compared to simple grid search

---

[1] Timestamps are given in hours since 1970.

[2] `https://github.com/rupnikj/jair_paper`

[3] The code and data we used is available at `https://github.com/priberam/news-clustering`.

[4] This shows a relative robustness to reordering of the articles – articles within 3 days of each other could appear anywhere in that window, and the algorithm would still perform well.

	Dataset	Size	Avg. L.	C	Avg. S.
train	English	12,233	434	593	21
	German	4,043	282	377	11
	Spanish	4,527	355	416	11
test	English	8,726	521	222	39
	German	2,101	440	118	18
	Spanish	2,177	392	149	15

Table 1: Statistics for the development and evaluation datasets, constructed from the dataset in Rupnik et al. (2016), as explained in §5. "Size" denotes the number of documents in the collection, "Avg. L." is the average number of words in a document, "C" denotes the number of clusters in the collection and "Avg. S." is the average number of documents in each cluster.

	algorithm	F_1	P	R
English	CluStream	79.0	**98.6**	65.9
	Tokens+Lemmas+Ents	92.7	92.9	**92.5**
	+TS	**94.1**	98.2	90.3
German	CluStream	89.7	**99.9**	81.3
	Tokens+Lemmas+Ents	90.7	99.7	83.2
	+TS	**97.1**	**99.9**	**94.5**
Spanish	CluStream	78.1	73.4	83.5
	Tokens+Lemmas+Ents	88.8	95.9	82.7
	+TS	**94.2**	**97.0**	**91.6**

Table 2: Clustering results on the labeled dataset. We compare our algorithm (with and without timestamps) with the online micro-clustering routine of Aggarwal and Yu (2006) (denoted by CluStream). The F_1 values are for the precision (P) and recall (R) in the following columns. See Table 3 for a legend of the different models. Best result for each language is in bold.

for the optimal τ (Table 4); (c) separating the feature space into one for monolingual clusters in the form of keywords and the other for crosslingual clusters based on crosslingual embeddings significantly helps performance.

Evaluation Method We evaluate clustering in the following manner: let tp be the number of correctly clustered-together document pairs, let fp be the number of incorrectly clustered-together document pairs and let fn be the number of incorrectly not-clustered-together document pairs. Then we report precision as $\frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}}$, recall as $\frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fn}}$ and F_1 as the harmonic mean of the precision and recall measures. We do the same to evaluate crosslingual clustering, but on a higher level: we count tp, fn and fp for the decisions of clustering clusters, as crosslingual clusters are groups of monolingual gold clusters.

5.1 Monolingual Results

In our first set of experiments, we report results on monolingual clustering for each language separately. Monolingual clustering of a stream of documents is an important problem that has been inspected by others, such as by Ahmed et al. (2011) and by Aggarwal and Yu (2006). We compare our results to our own implementation of the online micro-clustering routine presented by Aggarwal and Yu (2006), which shall be referred to as CluStream. We note that CluStream of Aggarwal and Yu (2006) has been a widely used state-of-the-art system in media monitoring companies as well as academia, and serves as a strong baseline to this day.

In our preliminary experiments, we also evaluated an online latent semantic analysis method, in which the centroids we keep for the function H (see §3) are the average of reduced dimensional vectors of the incoming documents as generated by an incremental singular value decomposition (SVD) of a document-term matrix that is updated after each incoming document. However, we discovered that online LSA performs significantly worse than representing the documents the way is described in §4. Furthermore, it was also significantly slower than our algorithm due to the time it took to perform singular value decomposition.[5]

Clustering experiments Table 2 gives the final monolingual results on the three datasets. For English, we see that the significant improvement we get using our algorithm over the algorithm of Aggarwal and Yu (2006) is due to an increased recall score. We also note that the trained models surpass the baseline for all languages, and that the timestamp feature (denoted by TS), while not required to beat the baseline, has a very relevant contribution in all cases. Although the results for both the baseline and our models seem to differ across languages, one can verify a consistent improvement from the latter to the former, suggesting that the score differences should be mostly tied to the different difficulty found across the datasets for each language. The presented scores show that our learning framework generalizes well to different languages and enables high quality clustering results.

To investigate the impact of the timestamp fea-

---

[5] More specifically, we used an object of type `lsimodel` from the GenSim package that implements algorithms from Řehůřek (2010). The GenSim package can be found at `https://pypi.python.org/pypi/gensim`.

feature	accuracy
TOKENS	85.5
TOKENS+LEMMAS	85.9
TOKENS+LEMMAS+ENTS	86.5
TOKENS+LEMMAS+ENTS+TS	96.9

Table 3: Accuracy of the SVM ranker on the English training set. TOKENS are the word token features, LEMMAS are the lemma features for title and body, ENTS are named entity features and TS are timestamp features. All features are described in detail in §4, and are listed for both the title and the body.

tures, we ran an additional experiment using only the same three timestamp features as used in the best model on the English dataset. This experiment yielded scores of $F_1 = 61.1$, $P = 44.5$ and $R = 97.6$, which lead us to conclude that while these features are not competitive when used alone (hence temporal information by itself is not sufficient to predict the clusters), they contribute significantly to recall with the final feature ensemble.

We note that as described in §3, the optimization of the τ parameter is part of the development process. The parameter τ is a similarity threshold used to decide when an incoming document should merge to the best cluster or create a new one. We tune τ on the development set for each language, and the sensitivity to it is demonstrated in Figure 3 (this process is further referred to as τ_{search}). Although applying grid-search on this parameter is the most immediate approach to this problem, we experimented with a different method which yielded superior results: as described further, we discuss how to do this process with an additional classifier (denoted SVM-merge), which captures more information about the incoming documents and the existing clusters.

Additionally, we also experimented with computing the monolingual clusters with the same embeddings as used in the crosslingual clustering phase, which yielded poor results. In particular, this system achieved F_1 score of 74.8 for English, which is below the bag-of-words baseline presented in Table 2. This result supports the approach we then followed of having two separate feature spaces for the monolingual and crosslingual clustering systems, where the monolingual space is discrete and the crosslingual space is based on embeddings.

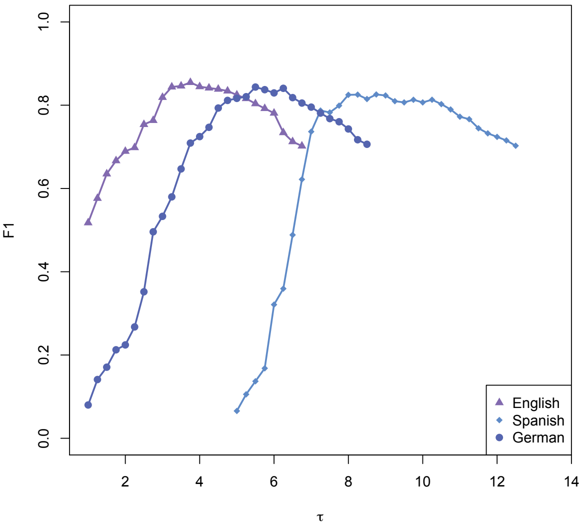

Figure 3: The F_1 score of the different language development sets as a function of the threshold τ. The first point for each language is identified using binary search.

model	F_1	P	R
τ_{search}	82.8	96.5	72.4
SVM-merge	94.1	98.2	90.3

Table 4: Comparison of two different cluster decision techniques for the English SVM model with all features (see Table 2). The first method, τ_{search}, corresponds to executing grid-search to find the optimal clustering τ parameter (see §3). SVM-merge is an alternative method in which we train an SVM binary classifier to decide if a new cluster should be created or not, where we use as features the maximal value of each coordinate for each document in a cluster.

SVM ranker experiments To investigate the importance of each feature, we now consider in Table 3 the accuracy of the SVM ranker for English as described in §4.1. We note that adding features increases the accuracy of the SVM ranker, especially the timestamp features. However, the timestamp feature actually interferes with our optimization of τ to identify when new clusters are needed, although they improve the SVM reranking accuracy. We speculate this is true because high accuracy in the reranking problem does not necessarily help with identifying when *new* clusters need to be opened. To investigate this issue, we experimented with a different technique to learn when to create a new cluster. To this end, we trained another SVM classifier just to learn this decision, this time a binary classifier using LIBLINEAR (Fan et al., 2008), by passing the max of the similarity of each feature between the incoming document and the current clustering pool as the input feature vector. This way, the classifier learns when the current clusters,

crosslingual model	F_1	P	R
τ_{search} (global)	72.7	89.8	61.0
τ_{search} (pivot)	84.0	83.0	85.0

Table 5: Crosslingual clustering results when considering two different approaches to compute distances across crosslingual clusters on the test set for Spanish, German and English. See text for details.

as a whole, are of a different news story than the incoming document. As presented in Table 4, this method, which we refer to as SVM-merge, solved the issue of searching for the optimal τ parameter for the SVM-rank model with timestamps, by greatly improving the F_1 score in respect to the original grid-search approach (τ_{search}).

5.2 Crosslingual Results

As mentioned in §3, crosslingual embeddings are used for crosslingual clustering. We experimented with the crosslingual embeddings of Gardner et al. (2015) and Ammar et al. (2016). In our preliminary experiments we found that the former worked better for our use-case than the latter.

We test two different scenarios for optimizing the similarity threshold τ for the crosslingual case. Table 5 shows the results for these experiments. First, we consider the simpler case of adjusting a global τ parameter for the crosslingual distances, as also described for the monolingual case. As shown, this method works poorly, since the τ grid-search could not find a reasonable τ which worked well for every possible language pair.

Subsequently, we also consider the case of using English as a pivot language (see §3), where distances for every other language are only compared to English, and crosslingual clustering decisions are made only based on this distance.[6] This yielded our best crosslingual score of F_1=84.0, confirming that crosslingual similarity is of higher quality between each language and English, for the embeddings we used. This score represents only a small degradation in respect to the monolingual results, since clustering across different languages is a harder problem.

6 Related Work

Early research efforts, such as the TDT program (Allan et al., 1998), have studied news clustering for some time. The problem of online monolingual clustering algorithms (for English) has also received a fair amount of attention in the literature. One of the earlier papers by Aggarwal and Yu (2006) introduced a two-step clustering system with both offline and online components, where the online model is based on a streaming implementation of k-means and a bag-of-words document representation. Other authors have experimented with distributed representations, such as Ahmed et al. (2011), who cluster news into *storylines* using Markov chain Monte Carlo methods, Řehůřek and Sojka (2010) who used incremental Singular Value Decomposition (SVD) to find relevant topics from streaming data, and Sato et al. (2017) who used the paragraph vector model (Le and Mikolov, 2014) in an offline clustering setting.

More recently, crosslingual linking of clusters has been discussed by Rupnik et al. (2016) in the context of linking existing clusters from the Event Registry (Leban et al., 2014) in a batch fashion, and by Steinberger (2016) who also present a batch clustering linking system. However, these are not "truly" online crosslingual clustering systems since they only decide on the linking of already-built monolingual clusters. In particular, Rupnik et al. (2016) compute distances of document pairs across clusters using nearest neighbors, which might not scale well in an online setting. As detailed before, we adapted the cluster-linking dataset from Rupnik et al. (2016) to evaluate our online crosslingual clustering approach. Preliminary work makes use of deep learning techniques (Xie et al., 2016; Guo et al., 2017) to cluster documents while learning their representations, but not in an online or multilingual fashion, and with a very small number of cluster labels (4, in the case of the text benchmark).

In our work, we studied the problem of monolingual and crosslingual clustering, having experimented several directions and methods and the impact they have on the final clustering quality. We described the first system which aggregates news articles into fine-grained story clusters across different languages in a completely online and scalable fashion from a continuous stream.

---

[6] In this case, all crosslingual clusters will have at least one pivot monolingual cluster, except for clusters which might stay unmerged as single-language degenerated crosslingual clusters.

7 Conclusion

We described a method for monolingual and crosslingual clustering of an incoming stream of documents. The method works by maintaining centroids for the monolingual and crosslingual clusters, where a monolingual cluster groups a set of documents and a crosslingual cluster groups a set of monolingual clusters. We presented an online crosslingual clustering method which auto-corrects past decisions in an efficient way. We showed that our method gives state-of-the-art results on a multilingual news article dataset for English, Spanish and German. Finally, we discussed how to leverage different SVM training procedures for ranking and classification to improve monolingual and crosslingual clustering decisions. Our system is integrated in a larger *media monitoring* project (Liepins et al., 2017; Germann et al., 2018) and solving the use-cases of monitors and journalists, having been validated with qualitative user testing.

Acknowledgments

We would like to thank Esma Balkır, Nikos Papasarantopoulos, Afonso Mendes, Shashi Narayan and the anonymous reviewers for their feedback. This project was supported by the European H2020 project SUMMA, grant agreement 688139 (see `http://www.summa-project.eu`) and by a grant from Bloomberg.

References

Charu C. Aggarwal and Philip S. Yu. 2006. A framework for clustering massive text and categorical data streams. In *SDM*, pages 479–483. SIAM.

Amr Ahmed, Qirong Ho, Jacob Eisenstein, Eric Xing, Alexander J Smola, and Choon Hui Teo. 2011. Unified analysis of streaming news. In *Proceedings of the 20th international conference on World wide web*, pages 267–276. ACM.

James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, Yiming Yang, et al. 1998. Topic detection and tracking pilot study: Final report. In *Proceedings of the DARPA broadcast news transcription and understanding workshop*.

Carlos Amaral, Adán Cassan, Helena Figueira, André Martins, Afonso Mendes, Pedro Mendes, José Pina, and Cláudia Pinto. 2008. Priberam's question answering system in QA@ CLEF 2008. In *Proceedings of the Workshop of the Cross-Language Evaluation Forum for European Languages*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Matt Gardner, Kejun Huang, Evangelos Papalexakis, Xiao Fu, Partha Talukdar, Christos Faloutsos, Nicholas Sidiropoulos, and Tom Mitchell. 2015. Translation invariant word embeddings. In *Proceedings of EMNLP*.

Ulrich Germann, Renars Liepins, Guntis Barzdins, Didzis Gosko, Sebastião Miranda, and David Nogueira. 2018. The summa platform: A scalable infrastructure for multi-lingual multi-media monitoring. *Proceedings of ACL 2018, System Demonstrations*.

Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. 2017. Improved deep embedded clustering with local structure preservation. In *Proceedings of IJCAI*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings ACM SIGKDD*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*.

Gregor Leban, Blaz Fortuna, Janez Brank, and Marko Grobelnik. 2014. Event registry: learning about world events from news. In *Proceedings of WWW*.

Kristina Lerman and Tad Hogg. 2010. Using a model of social dynamics to predict popularity of news. In *Proceedings of WWW*.

Renars Liepins, Ulrich Germann, Guntis Barzdins, Alexandra Birch, Steve Renals, Susanne Weber, Peggy van der Kreeft, Herve Bourlard, João Prieto, Ondrej Klejch, Peter Bell, Alexandros Lazaridis, Alfonso Mendes, Sebastian Riedel, Mariana S. C. Almeida, Pedro Balage, Shay B. Cohen, Tomasz Dwojak, Philip N. Garner, Andreas Giefer, Marcin Junczys-Dowmunt, Hina Imran, David Nogueira, Ahmed Ali, Sebastião Miranda, Andrei Popescu-Belis, Lesly Miculicich Werlen, Nikos Papasarantopoulos, Abiola Obamuyide, Clive Jones, Fahim Dalvi, Andreas Vlachos, Yang Wang, Sibo Tong, Rico Sennrich, Nikolaos Pappas, Shashi Narayan, Marco Damonte, Nadir Durrani, Sameer Khurana, Ahmed Abdelali, Hassan Sajjad, Stephan Vogel, David Sheppey, Chris Hernon, and Jeff Mitchell. 2017. The SUMMA platform prototype. In *Proceedings of the Software Demonstrations of EACL*.

André F. T. Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of ACL*.

Radim Řehůřek. 2010. Fast and faster: A comparison of two streamed matrix decomposition algorithms.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*.

Jan Rupnik, Andrej Muhic, Gregor Leban, Primoz Skraba, Blaz Fortuna, and Marko Grobelnik. 2016. News across languages-cross-lingual document similarity and event tracking. *Journal of Artificial Intelligence Research*, 55:283–316.

Motoki Sato, Austin J Brockmeier, Georgios Kontonatsios, Tingting Mu, John Y Goulermas, Jun'ichi Tsujii, and Sophia Ananiadou. 2017. Distributed document and phrase co-embeddings for descriptive clustering. In *Proceedings of EACL*.

Josef Steinberger. 2016. MediaGist: A cross-lingual analyser of aggregated news and commentaries. In *Proceedings of ACL*.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *Proceedings of ICML*.

Multi-Task Label Embedding for Text Classification

Honglun Zhang[1], **Liqiang Xiao**[1], **Wenqing Chen**[1], **Yongkun Wang**[2], **Yaohui Jin**[1]
[1]State Key Lab of Advanced Optical Communication System and Network
MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[2]Network and Information Center, Shanghai Jiao Tong University
{jinyh}@sjtu.edu.cn

Abstract

Multi-task learning in text classification leverages implicit correlations among related tasks to extract common features and yield performance gains. However, a large body of previous work treats labels of each task as independent and meaningless one-hot vectors, which cause a loss of potential label information. In this paper, we propose **Multi-Task Label Embedding** to convert labels in text classification into semantic vectors, thereby turning the original tasks into vector matching tasks. Our model utilizes semantic correlations among tasks and makes it convenient to scale or transfer when new tasks are involved. Extensive experiments on five benchmark datasets for text classification show that our model can effectively improve the performances of related tasks with semantic representations of labels and additional information from each other.

1 Introduction

Text classification is a common *Natural Language Processing* (NLP) issue that tries to infer the most appropriate label for a given sentence or document, for example, sentiment analysis, topic classification and so on. With the developments and prosperities of *Deep Learning* (Bengio et al., 2013), many neural network based models have been exploited by a large body of literature and achieved inspiring performance gains on various text classification tasks. These models are robust at feature engineering and can represent word sequences as fixed-length vectors with rich semantic information, which are notably ideal for subsequent NLP tasks.

Due to numerous parameters to train, neural network based models rely heavily on adequate amounts of annotated corpora, which cannot always be met as constructions of large-scale high-quality labeled datasets are extremely time-consuming and labor-intensive. *Multi-Task Learning* (**MTL**) solves this problem by jointly training multiple related tasks and leveraging potential correlations among them to increase corpora size implicitly, extract common features and yield performance gains. Inspired by (Caruana, 1997), there is a large body of research dedicated to MTL with neural network based models (Collobert and Weston, 2008; Liu et al., 2015b, 2016a,b; Zhang et al., 2017). These models usually contain a pre-trained lookup layer that map words into dense, low-dimension and real-value vectors with semantic implications, namely known as *Word Embedding* (Mikolov et al., 2013b), and utilize some lower layers to capture common features that are further fed to follow-up task-specific layers. However, many existing models may have at least one of the following three disadvantages:

- Lack of **Label Information**. Labels of each task are represented by independent and meaningless one-hot vectors, for example, *positive* and *negative* in sentiment analysis encoded as $[1, 0]$ and $[0, 1]$, which may cause a loss of potential label information.
- Inability to **Scale**. Network structures are elaborately designed to model various correlations for MTL, but many of them are structurally fixed and some can only deal with interactions between two tasks, namely pairwise interactions. When new tasks are involved, the network structures have to be modified and all networks have to be trained again.
- Inability to **Transfer**. Human beings can handle a completely new task without anymore efforts after learning with several related tasks, which can be called the ability

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4545–4553
Brussels, Belgium, October 31 - November 4, 2018.

of *Transfer Learning* (Ling et al., 2008). As discussed above, the network structures of many previous models are fixed and there are no layers specially designed for unannotated new tasks.

In this paper, we proposed **Multi-Task Label Embedding** (**MTLE**) to map labels of each task into semantic vectors, similar to how Word Embedding deals with word sequences, thereby turning the original tasks into vector matching tasks. The idea of embedding label is not new and is primarily designed to improve general neural text classification models (Bengio et al., 2010; Norouzi et al., 2013; Ma et al., 2016), but here we mainly focus on integrating label embedding to enhance MTL. MTLE utilizes semantic correlations among tasks and effectively solves the problems of scaling and transferring when new tasks are involved.

We conduct extensive experiments on five benchmark datasets for text classification. Compared to learning separately, jointly learning multiple related tasks based on MTLE demonstrates significant performance gains for each task.

Our contributions are four-fold:

- Our model efficiently leverages label information of each task by mapping labels into dense, low-dimension and real-value vectors with semantic implications.
- It is particularly convenient for our model to scale for new tasks. The network structures need no modifications and only samples from the new tasks require training.
- After training on several related tasks, our model can also naturally transfer to deal with new tasks without anymore training, while still achieving appreciable performances.
- We consider different scenarios of MTL and demonstrate strong results on several benchmark datasets for text classification. Our model outperforms most of the state-of-the-art baselines.

2 Problem Statements

2.1 Single-Task Learning

For a text classification task, the **Input** is a word sequence $x = \{x_1, x_2, ..., x_T\}$, where T is the sequence length and we need to output the most appropriate **Label** from the set $\{y_1, y_2, ..., y_C\}$, or their one-hot representation $\{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_C\}$, where C is the number of classes. In the supervised case, we have the **Annotation**, that is, the corresponding ground truth label $\tilde{\mathbf{y}}$ for each input, while in the unsupervised case, we only know the label set but lack the specific annotations.

A pre-trained lookup layer is used to get the embedding vector $\mathbf{x}_t \in \mathbb{R}^d$ for each word x_t. A text classification model f is trained to produce the predicted distribution $\hat{\mathbf{y}}$ for each $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$.

$$f(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T) = \hat{\mathbf{y}} \tag{1}$$

and the learning objective is to minimize the overall cross-entropy on the training set.

$$l = -\sum_{i=1}^{N}\sum_{j=1}^{C} \tilde{y}_{ij} \log \hat{y}_{ij} \tag{2}$$

where N denotes the number of training samples.

2.2 Multi-Task Learning

Given K supervised text classification tasks, $T_1, T_2, ..., T_K$, a multi-task learning model F is trained to transform each $\mathbf{x}^{(k)}$ from T_k into multiple predicted distributions $\{\hat{\mathbf{y}}^{(1)}, ..., \hat{\mathbf{y}}^{(K)}\}$.

$$F(\mathbf{x}_1^{(k)}, ..., \mathbf{x}_T^{(k)}) = \{\hat{\mathbf{y}}^{(1)}, ..., \hat{\mathbf{y}}^{(K)}\} \tag{3}$$

where only $\hat{\mathbf{y}}^{(k)}$ is used for loss computation. The overall training loss is a weighted combination of costs for each task.

$$L = -\sum_{k=1}^{K} \lambda_k \sum_{i=1}^{N_k}\sum_{j=1}^{C_k} \tilde{y}_{ij}^{(k)} \log \hat{y}_{ij}^{(k)} \tag{4}$$

where λ_k, N_k and C_k denote the weight, the number of training samples and the number of classes for each task T_k.

2.3 Ability to Scale

When new tasks are involved, for example, given K' more supervised tasks $T_{K+1}, T_{K+2}, ..., T_{K+K'}$, the MTL model F, which has been trained on the old K tasks, should be able to scale with few structural modifications.

$$F(\mathbf{x}_1^{(k)}, ..., \mathbf{x}_T^{(k)}) = \{\hat{\mathbf{y}}^{(1)}, ..., \hat{\mathbf{y}}^{(K+K')}\} \tag{5}$$

where $\mathbf{x}^{(k)}$ denotes samples from not only the old tasks but also the new ones.

There are mainly two scaling methods for F to deal with the new tasks. We can continue training F and further tune the model parameters based on samples from the new tasks, which we define as **Hot Update**, or re-train F again based on training samples from all tasks, which is defined as **Cold Update**.

2.4 Ability to Transfer

Given K'' more completely unannotated new tasks $T^{\circ}_{K+1}, T^{\circ}_{K+2}, ..., T^{\circ}_{K+K''}$, F should be able to transfer what it learned from the old tasks, which is defined as **Zero Update**, as no more training samples are available and the model parameters are not updated anymore.

For each $\mathbf{x}^{(k)}$ from the K'' tasks, F should produce the corresponding predicted distributions for the new tasks, even if the annotations of these new tasks are not provided at all.

$$F(\mathbf{x}^{(k)}_1, ..., \mathbf{x}^{(k)}_T) = \{\hat{\mathbf{y}}^{(K+1)}, ..., \hat{\mathbf{y}}^{(K+K'')}\} \quad (6)$$

which requires that F should be able to understand the meanings of each label from the new tasks and infer the most appropriate one.

3 Methodology

In text classification tasks, labels can be single-word or double-word, for example, *positive* and *negative* in binary sentiment classification, *very positive*, *positive*, *neutral*, *negative* and *very negative* in 5-category sentiment classification, but there are few labels that contain three words or more. Inspired by Word Embedding, we propose **Multi-Task Label Embedding (MTLE)** to convert labels of each task of MTL into dense, low-dimension and real-value vectors with semantic implications, thereby disclosing potential intra-task and inter-task correlations from both texts and labels.

3.1 Architecture

Figure 1 illustrates the general idea of MTLE, which mainly consists of three parts, the **Input Encoder**, the **Label Encoder** and the **Matcher**.

In the Input Encoder, each input sequence $x^{(k)}$ from T_k is transformed into embedding representation $\mathbf{x}^{(k)} = \{\mathbf{x}^{(k)}_1, \mathbf{x}^{(k)}_2, ..., \mathbf{x}^{(k)}_T\}$ by the Embedding Layer (E_I). The Learning Layer (L_I) is applied to recurrently comprehend $\mathbf{x}^{(k)}$ and generate a fixed-length vector $\mathbf{X}^{(k)}$, which can be regarded as an overall representation of the original input sequence $x^{(k)}$.

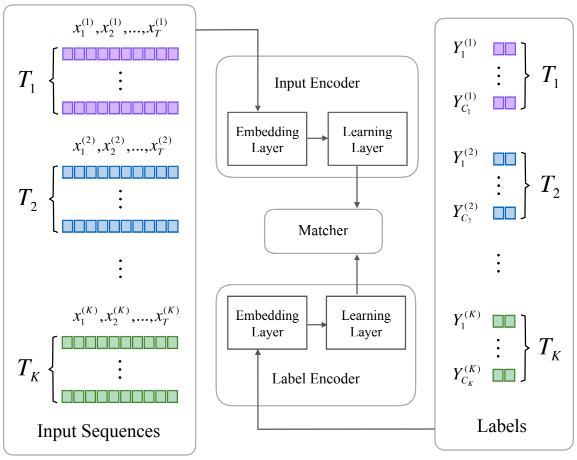

Figure 1: General Architecture of MTLE

In the Label Encoder, there are C_k labels in T_k, where each $y^{(k)}_j (1 \leq j \leq C_k)$ consists of one word or two words, and is mapped into the vector representation $\mathbf{y}^{(k)}_j$ by the Embedding Layer (E_L). The Learning Layer (L_L) absorbs $\mathbf{y}^{(k)}_j$ to generate a fixed-length vector $\mathbf{Y}^{(k)}_j$, which can be utilized as an overall semantic representation of the original label $y^{(k)}_j$.

In order to classify a sample $x^{(k)}$ from T_k, the Matcher obtains the corresponding $\mathbf{X}^{(k)}$ from the Input Encoder, all $\mathbf{Y}^{(k)}_j (1 \leq j \leq C_k)$ from the Label Encoder, and conducts vector matching to select the most appropriate class label.

In MTLE, E_I and E_L obtain understandings of words from texts and labels respectively, while L_I and L_L achieve representation abilities of vector sequences. All these four modules are learned within and shared among tasks.

3.2 Implementation Details

We can explore different embedding methods and neural networks to achieve E_I, E_L and L_I, L_L respectively, but here we choose to implement them in an easier way and spend more efforts to investigate the effectiveness of MTLE.

The implementation details of MTLE are illustrated in Figure 2. L_I and L_L are both fully-connection layers with the weights $\mathbf{W}_I$ and $\mathbf{W}_L$ of $|V| \times d$, where $|V|$ denotes the vocabulary size and d is the embedding size. We can get a pre-trained lookup table based on open domain corpora to initialize $\mathbf{W}_I, \mathbf{W}_L$ and further tune them during training.

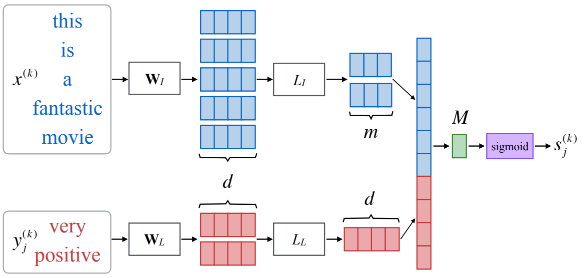

Figure 2: Implementation Details of MTLE

L_I and L_L should be trainable models that can transform a vector sequence of arbitrary length into a fixed-length vector, which can be implemented by a *Bi-directional Long Short-Term Memory Network* (**BiLSTM**) that can recurrently process vector sequences and learn long-term dependencies.

While there are numerous variants of the standard LSTM (Hochreiter and Schmidhuber, 1997), here we follow the implementation of (Graves, 2013). At each time step t, states of the LSTM can be fully described by five vectors in $\mathbb{R}^m$, an input gate $\mathbf{i}_t$, a forget gate $\mathbf{f}_t$, an output gate $\mathbf{o}_t$, the hidden state $\mathbf{h}_t$ and the memory cell $\mathbf{c}_t$, which adhere to the following transition equations.

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{V}_i\mathbf{c}_{t-1} + \mathbf{b}_i) \quad (7)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{V}_f\mathbf{c}_{t-1} + \mathbf{b}_f) \quad (8)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{V}_o\mathbf{c}_{t-1} + \mathbf{b}_o) \quad (9)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1}) \quad (10)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \quad (11)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (12)$$

where $\mathbf{x}_t$ is the current input, σ denotes logistic sigmoid function and $\odot$ denotes element-wise multiplication.

A BiLSTM consists of two LSTM layers that process the input sequences in original and reversed orders, and its output is the concatenation of hidden states from the forward and backward LSTM at each time step.

$$\mathbf{h}_t = \overrightarrow{\mathbf{h}}_t \oplus \overleftarrow{\mathbf{h}}_t \quad (13)$$

where $\oplus$ denotes vector concatenation.

We apply the above equations to implement L_I with hidden size m. However, it is inappropriate to apply a BiLSTM for L_L, as most labels contain only one or two words. Instead, L_L accepts the embedding vectors of each word from a label and calculate the average.

For an input sample $x^{(k)}$ and all C_k labels $y_j^{(k)}$ from T_k, the corresponding $\mathbf{X}^{(k)} \in \mathbb{R}^{2m}$ and $\mathbf{Y}_j^{(k)} \in \mathbb{R}^d$ are calculated as follows.

$$\mathbf{X}^{(k)} = L_I(\mathbf{W}_I(x^{(k)})) \quad (14)$$

$$\mathbf{Y}_j^{(k)} = L_L(\mathbf{W}_L(y_j^{(k)})) \quad (15)$$

In this paper, we mainly focus on the idea and effects of MTLE, so rather than exploring some useful mechanisms like gating, external memory or attention for stronger abilities of sequence learning, we choose the vanilla BiLSTM for quick implementations and spend most of our efforts on investigating the effectiveness of MTLE.

We concatenate $\mathbf{X}^{(k)}, \mathbf{Y}_j^{(k)}$ and apply another fully-connection layer with only one neuron, denoted by M, to implement the Matcher, which accepts outputs from L_I and L_L to produce a score of matching. Given the matching scores of each label, we refer to the idea of cross-entropy and calculate the loss function for a sample $x^{(k)}$ from T_k as follows.

$$s_j^{(k)} = \sigma(M(\mathbf{X}^{(k)} \oplus \mathbf{Y}_j^{(k)})) \quad (16)$$

$$\hat{y}_j^{(k)} = \frac{\exp(s_j^{(k)})}{\sum_{c=1}^{C_k} \exp(s_c^{(k)})} \quad (17)$$

$$l^{(k)} = -\sum_{j=1}^{C_k} \tilde{y}_j^{(k)} \log \hat{y}_j^{(k)} \quad (18)$$

The overall training objective is to minimize the weighted linear combination of costs for each task.

$$L = -\sum_{k=1}^{K} \lambda_k \sum_{i=1}^{N_k} l_i^{(k)} \quad (19)$$

MTLE provides an easy and intuitive way to realize MTL, where input texts and class labels from different tasks are jointly learned and compactly fused. During training, E_I and E_L learn better understanding of word semantics for different tasks, L_I and L_L obtain stronger abilities of sequence representation, while M produces more accurate scores for vector matching.

3.3 Scaling and Transferring

When new tasks are involved, it is particularly convenient for MTLE to scale or transfer as the network structure needs no modifications. For input samples $x^{(k)}$ and class labels $y_j^{(k)}$ from the new tasks, we can apply E_I, E_L, L_I, L_L to get their vector representations $\mathbf{X}^{(k)}$ and $\mathbf{Y}_j^{(k)}$, calculate the matching scores and find the most appropriate label.

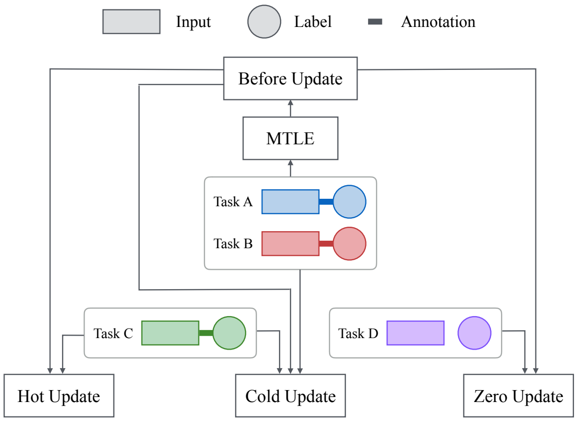

Figure 3: Three different updating methods

If the new tasks are annotated, we can apply Hot Update or Cold Update for scaling and better tune the parameters. If the new tasks are completely unannotated, we can use Zero Update for transferring and produce reasonable predictions.

The differences among Hot Update, Cold Update and Zero Update are illustrated in Figure 3, where **Before Update** denotes the state of MTLE trained on the old tasks before the new tasks are introduced. We will further compare these updating methods in the Experiment Section.

4 Experiment

In this section, we design extensive experiments with multi-task learning based on five benchmark datasets for text classification. We investigate the empirical performances of MTLE and compare them with existing state-of-the-art baselines.

4.1 Datasets

As Table 1 shows, we select five benchmark datasets for text classification, which are commonly used in a large body of research on MTL. We design three experiment scenarios to evaluate the performances of MTLE. Larger text classification datasets (Zhang et al., 2015) are not chosen as there are already enough samples to train and MTL may increase computation workloads. We use the accuracy for evaluations as samples from these datasets are mostly well balanced.

- **Multi-Cardinality** Movie review datasets with different average lengths and class numbers, including **SST-1** (Socher et al., 2013), **SST-2**, **IMDB** (Maas et al., 2011).
- **Multi-Domain** Product review datasets on different domains from *Multi-Domain Sentiment Dataset* (Blitzer et al., 2007).
- **Multi-Objective** Text classification datasets with different objectives, including **IMDB**, **RN** (Apté et al., 1994), **QC** (Li and Roth, 2002).

4.2 Hyperparameters and Training

Training of MTLE is conducted through back propagation with batch gradient descent (Amari, 1993). We obtain a pre-trained lookup table by applying *Word2Vec* (Mikolov et al., 2013a) on the Google News corpus, which contains more than 100B words with a vocabulary size of about 3M. During each epoch, we randomly divide training samples from different tasks into batches of fixed size. For each iteration, we randomly select one task and choose an untrained batch, calculate the gradient and update the parameters accordingly.

Parameters of the neural layers are randomly initialized with the Xavier initializer (Glorot and Bengio, 2010). We apply 10-fold cross-validation and different combinations of hyperparameters are investigated, of which the best one is described in Table 2.

4.3 Results of MTLE vs. Single Task

In Table 3, we compare the performances of MTLE with single-task learning, where only a BiLSTM layer is applied.

MTLE achieves significant performance gains with label information and additional correlations from related tasks. Multi-Domain, Multi-Cardinality and Multi-Objective benefit from MTLE with average improvements of 5.5%, 2.7% and 1.2%, as they contain increasingly weaker relevance among tasks. The result of IMDB in Multi-Cardinality is slightly better than that in Multi-Objective (91.3 against 90.9), as SST-1 and SST-2 share more semantically useful information with IMDB than RN and QC.

4.4 Abilities to Scale and Transfer

In order to investigate the abilities of MTLE to scale and transfer, we use $A + B \rightarrow C$ to denote the case where MTLE is trained on task A and B, while C is the newly involved one. We design three cases based on different scenarios and

Table 1: Five benchmark text classification datasets

Dataset	Description	Type	Average Length	Class	Objective
SST	Movie reviews in Stanford Sentiment Treebank including **SST-1** and **SST-2**	Sentence	19 / 19	5 / 2	Sentiment
IMDB	Internet Movie Database	Document	279	2	Sentiment
MDSD	Product reviews on books, DVDs, electronics and kitchen (**BDEK**)	Document	176 / 189 / 115 / 97	2	Sentiment
RN	Reuters Newswire topics classification	Document	146	46	Topics
QC	Question Classification	Sentence	10	6	Question Types

Table 2: Hyperparameter settings

Embedding size	$d = 300$
Hidden layer size of LSTM	$m = 100$
Batch size	$\delta = 32$
Initial learning rate	$\eta = 0.1$
Regularization weight	$\lambda = 10^{-5}$

compare the influences of Hot Update, Cold Update, Zero Update on each task.

- **Case 1** SST-1 + SST-2 → IMDB.
- **Case 2** B + D + E → K.
- **Case 3** RN + QC → IMDB.

where in Zero Update, we ignore the training set of C and just evaluate our model on the testing set.

As Table 4 shows, Before Update denotes the model trained on the old tasks before the new tasks are involved, so only evaluations on the old tasks are conducted.

Cold Update re-trains the model of Before Update with both the old tasks and the new tasks, thus achieving similar performances with the last line in Table 3. Different from Cold Update, Hot Update resumes training only on the new tasks, requires much less training time, while still obtains competitive results for all tasks. The new tasks like IMDB and Kitchen benefit more from Hot Update than the old tasks, as the parameters are further tuned according to annotations from these new tasks.

Zero Update provides inspiring possibilities for completely unannotated tasks. There are no more annotations for additional training from the new tasks, so we just apply the model of Before Update for evaluations on the testing sets of the new tasks. Zero Update achieves competitive performances in Case 1 (90.9 for IMDB) and Case 2 (86.7 for Kitchen), as tasks from these two cases all belong to sentiment datasets of different cardinalities or domains that contain rich semantic correlations with each other. However, the result for IMDB in Case 3 is only 74.2, as sentiment shares less relevance with topic and question type, thus leading to poor transferring performances.

4.5 Multi-Task or Label Embedding

MTLE mainly employs two mechanisms, label embedding and multi-task learning, so both implicit information from labels and potential correlations from other tasks make differences. In this section, we conduct experiments to explore the contributions of label embedding and multi-task learning respectively.

We choose the four tasks from Multi-Domain scenario and train MTLE on each task separately, so their performances are only influenced by label embedding. Then we re-train MTLE from scratch for every two tasks, every three tasks from them and record the performances of each task in different cases, where both label embedding and multi-task learning matter.

The results are illustrated in Figure 4. The first three graphs show the results of MTLE trained on every one, every two and every three tasks. In the first graph, the four tasks are trained separately and achieve improvements of 3.0%, 3.1%, 3.3%, 2.3% compared to the single task in Table 3. As more tasks are added step by step, MTLE produces increasing performance gains for each task and achieves an average improvement of 5.5% when all the four tasks are trained together. So it can

Table 3: Results of MTLE on different scenarios

Model	**Multi-Cardinality**			**Multi-Domain**				**Multi-Objective**			AvgΔ
	SST-1	SST-2	IMDB	B	D	E	K	IMDB	RN	QC	
Single Task	46.2	86.1	88.9	78.3	79.8	81.5	82.3	89.0	84.2	92.7	-
MTLE	**49.8**	**88.4**	**91.3**	**84.5**	**85.2**	**87.3**	**86.9**	**90.9**	**85.5**	**93.2**	+3.4

Table 4: Results of Hot Update, Cold Update and Zero Update in different cases

Model	**Case 1**			**Case 2**				**Case 3**		
	SST-1	SST-2	IMDB	B	D	E	K	RN	QC	IMDB
Before Update	48.6	87.6	-	83.7	84.5	85.9	-	84.8	**93.4**	-
Cold Update	**49.8**	**88.5**	91.2	**84.4**	**85.2**	**87.2**	86.9	**85.5**	93.2	91.0
Hot Update	49.6	88.1	**91.4**	84.2	84.9	87.0	**87.1**	85.2	92.9	**91.1**
Zero Update	-	-	90.9	-	-	-	86.7	-	-	74.2

be concluded that both information from labels as well as correlations from other tasks account for considerable parts of contributions.

In the last graph, diagonal cells denote improvements of every one task, while off-diagonal cells denote average improvements of every two tasks, so an off-diagonal cell of darker color indicates stronger correlation between the two tasks. An interesting finding is that Books is more related with DVDs and Electronics is more relevant to Kitchen. A possible reason may be that Books and DVDs are products targeted for reading or watching, while customers care more about appearances and functionalities when talking about Electronics and Kitchen.

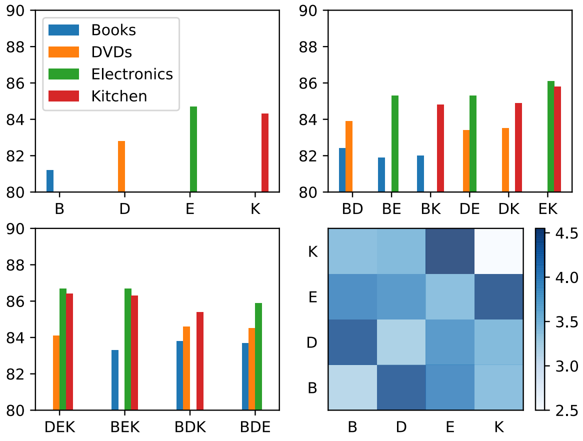

Figure 4: Performance gains of each task in different cases

4.6 Comparisons with State-of-the-art Models

We compare MTLE against the following models.

- **NBOW** Neural Bag-of-Words that sums up embedding vectors of all words and applies a non-linearity followed by a softmax layer.
- **PV** Paragraph Vectors followed by logistic regression (Le and Mikolov, 2014).
- **CNN** Convolutional Neural Networks for Sentence Classification (Kim, 2014).
- **MT-CNN** Multi-Task learning with Convolutional Neural Networks (Collobert and Weston, 2008) where lookup tables are partially shared.
- **MT-DNN** Multi-Task learning with Deep Neural Networks (Liu et al., 2015b) that utilizes bag-of-word representations and a hidden shared layer.
- **MT-RNN** Multi-Task learning with Recurrent Neural Networks with a shared-layer (Liu et al., 2016b).
- **DSM** Deep multi-task learning with Shared Memory (Liu et al., 2016a) where a external memory and a reading/writing mechanism are introduced.
- **GRNN** Gated Recursive Neural Network for sentence modeling and text classification (Chen et al., 2015).
- **Tree-LSTM** A generalization of LSTMs to tree-structured network topologies (Tai et al., 2015).

As Table 5 shows, MTLE achieves competitive or better performances on most tasks except

Table 5: Comparisons of MTLE against state-of-the-art models

Model	SST-1	SST-2	IMDB	Books	DVDs	Electronics	Kitchen	QC
NBOW	42.4	80.5	83.6	-	-	-	-	88.2
PV	44.6	82.7	**91.7**	-	-	-	-	91.8
CNN	48.0	88.1	-	-	-	-	-	93.6
MT-CNN	-	-	-	80.2	81.0	83.4	83.0	-
MT-DNN	-	-	-	79.7	80.5	82.5	82.8	-
MT-RNN	49.6	87.9	91.3	-	-	-	-	-
DSM	49.5	87.8	91.2	82.8	83.0	85.5	84.0	-
GRNN	47.5	85.5	-	-	-	-	-	**93.8**
Tree-LSTM	**50.6**	86.9	-	-	-	-	-	-
MTLE	49.8	**88.4**	91.3	**84.5**	**85.2**	**87.3**	**86.9**	93.2

for QC, as it contains less correlations with other tasks. Tree-LSTM outperforms our model on SST-1 (50.6 against 49.8), but it requires an external parser to get the sentence topological structure and utilizes treebank annotations. PV slightly surpasses MTLE on IMDB (91.7 against 91.3), as sentences from IMDB are much longer than SST and MDSD, which require stronger abilities of long-term dependency learning.

In this paper, we mainly focus the idea and effects of integrating label embedding to enhance multi-task learning, so we apply the BiLSTM to realize L_I, which can be further implemented by other more powerful sequence learning models (Liu et al., 2015a; Chen et al., 2015; Tai et al., 2015) and produce better performances. Explorations of other embedding layers and learning layers may be appreciated, but due to limited pages we choose to research these contents in future work.

5 Related Work

There is a large body of literature related to multi-task learning with neural networks in NLP (Collobert and Weston, 2008; Liu et al., 2015b, 2016a,b; Zhang et al., 2017).

(Collobert and Weston, 2008) use a shared lookup layer for common features, followed by task-specific layers for several traditional NLP tasks including part-of-speech tagging and semantic parsing. They use a fix-size window to solve the problem of variable-length input sequences, which can be better addressed by RNN.

(Liu et al., 2015b, 2016a,b; Zhang et al., 2017) all investigate MTL for text classification. (Liu et al., 2015b) apply bag-of-word representation, but information on word order is lost. (Liu et al., 2016a) introduce an external memory for information sharing with a reading/writing mechanism for communications. (Liu et al., 2016b) propose three different models for MTL with RNN and (Zhang et al., 2017) constructs a generalized architecture for RNN based MTL. However, models of these papers ignore essential information of labels and mostly can only address pairwise interactions between two tasks. Their network structures are also fixed, thereby failing to scale or transfer when new tasks are involved.

Different from the above work, MTLE maps labels of text classification tasks into semantic vectors and provide a more intuitive way to realize MTL with the abilities to scale and transfer. Input sequences from three or more tasks are jointly learned together with their labels, benefitting from each other and obtaining better sequence representations.

6 Conclusion and Future Work

In this paper, we propose Multi-Task Label Embedding to map labels of text classification tasks into semantic vectors. MTLE utilizes semantic correlations among tasks and effectively solves the problems of scaling and transferring when new tasks are involved. We explore three different scenarios of MTL and MTLE can improve performances of most tasks with semantic representations of labels and additional information from others in all scenarios.

In future work, we would like to explore other learning layers and generalize MTLE to address other NLP tasks, for example, sequence labeling and sequence-to-sequence learning.

References

Shun-ichi Amari. 1993. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(3):185–196.

Chidanand Apté, Fred Damerau, and Sholom M. Weiss. 1994. Automated Learning of Decision Rules for Text Categorization. *ACM Trans. Inf. Syst.*, 12(3):233–251.

Samy Bengio, Jason Weston, and David Grangier. 2010. Label embedding trees for large multi-class tasks. In *NIPS*, pages 163–171.

Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *ACL*.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28(1):41–75.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Shiyu Wu, and Xuanjing Huang. 2015. Sentence Modeling with Gated Recursive Neural Network. In *EMNLP*, pages 793–798.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, pages 160–167.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256.

Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR*, abs/1308.0850.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*, pages 1188–1196.

Xin Li and Dan Roth. 2002. Learning Question Classifiers. In *COLING*.

Xiao Ling, Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2008. Spectral domain-transfer learning. In *ACM SIGKDD*, pages 488–496.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015a. Multi-Timescale Long Short-Term Memory Neural Network for Modelling Sentences and Documents. In *EMNLP*, pages 2326–2335.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016a. Deep Multi-Task Learning with Shared Memory for Text Classification. In *EMNLP*, pages 118–127.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016b. Recurrent Neural Network for Text Classification with Multi-Task Learning. In *IJCAI*, pages 2873–2879.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015b. Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. In *NAACL HLT*, pages 912–921.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *COLING*, pages 171–180.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *NAACL HLT*, pages 142–150. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, pages 3111–3119.

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg Corrado, and Jeffrey Dean. 2013. Zero-shot learning by convex combination of semantic embeddings. *CoRR*, abs/1312.5650.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *EMNLP*, pages 1631–1642, Stroudsburg, PA. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, pages 1556–1566.

Honglun Zhang, Liqiang Xiao, Yongkun Wang, and Yaohui Jin. 2017. A generalized recurrent neural architecture for text classification with multi-task learning. In *IJCAI-17*, pages 3385–3391.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.

Semantic-Unit-Based Dilated Convolution for Multi-Label Text Classification

Junyang Lin[1,2], **Qi Su**[1], **Pengcheng Yang**[2], **Shuming Ma**[2], **Xu Sun**[2]
School of Foreign Languages, Peking University[1]
MOE Key Lab of Computational Linguistics, School of EECS, Peking University[2]
`{linjunyang, sukia, yang_pc, shumingma, xusun}@pku.edu.cn`

Abstract

We propose a novel model for multi-label text classification, which is based on sequence-to-sequence learning. The model generates higher-level semantic unit representations with multi-level dilated convolution as well as a corresponding hybrid attention mechanism that extracts both the information at the word-level and the level of the semantic unit. Our designed dilated convolution effectively reduces dimension and supports an exponential expansion of receptive fields without loss of local information, and the attention-over-attention mechanism is able to capture more summary relevant information from the source context. Results of our experiments show that the proposed model has significant advantages over the baseline models on the dataset RCV1-V2 and Ren-CECps, and our analysis demonstrates that our model is competitive to the deterministic hierarchical models and it is more robust to classifying low-frequency labels[1].

1 Introduction

Multi-label text classification refers to assigning multiple labels for a given text, which can be applied to a number of important real-world applications. One typical example is that news on the website often requires labels with the purpose of the improved quality of search and recommendation so that the users can find the preferred information with high efficiency with less disturbance of the irrelevant information. As a significant task of natural language processing, a number of methods have been applied and have gradually achieved satisfactory performance. For instance, a series of methods based on machine learning have been extensively utilized in the industries, such as Binary Relevance (Boutell et al., 2004). BR treats the task as multiple single-label classifications and can achieve satisfactory performance. With the development of Deep Learning, neural methods are applied to this task and achieved improvements (Zhang and Zhou, 2006; Nam et al., 2013; Benites and Sapozhnikova, 2015).

However, these methods cannot model the internal correlations among labels. To capture such correlations, the following work, including ML-DT (Clare and King, 2001), Rank-SVM (Elisseeff and Weston, 2002), LP (Tsoumakas and Katakis, 2006), ML-KNN (Zhang and Zhou, 2007), CC (Read et al., 2011), attempt to capture the relationship, which though demonstrated improvements yet simply captured low-order correlations. A milestone in this field is the application of sequence-to-sequence learning to multi-label text classification (Nam et al., 2017). Sequence-to-sequence learning is about the transformation from one type of sequence to another type of sequence, whose most common architecture is the attention-based sequence-to-sequence (Seq2Seq) model. The attention-based Seq2Seq (Sutskever et al., 2014) model is initially designed for neural machine translation (NMT) (Bahdanau et al., 2014; Luong et al., 2015). Seq2Seq is able to encode a given source text and decode the representation for a new sequence to approximate the target text, and with the attention mechanism, the decoder is competent in extracting vital source-side information to improve the quality of decoding. Multi-label text classification can be regarded as the prediction of the target label sequence given a source text, which can be modeled by the Seq2Seq. Moreover, it is able to model the high-order correlations among the source text as well as those among the label sequence with deep recurrent neural networks (RNN).

Nevertheless, we study the attention-based Seq2Seq model for multi-label text classification

---

[1] The code is available at `https://github.com/lancopku/SU4MLC`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4554–4564
Brussels, Belgium, October 31 - November 4, 2018.

(Nam et al., 2017) and find that the attention mechanism does not play a significant role in this task as it does in other NLP tasks, such as NMT and abstractive summarization. In Section 3, we demonstrate the results of our ablation study, which show that the attention mechanism cannot improve the performance of the Seq2Seq model. We hypothesize that compared with neural machine translation, the requirements for neural multi-label text classification are different. The conventional attention mechanism extracts the word-level information from the source context, which makes little contribution to a classification task. For text classification, human does not assign texts labels simply based on the word-level information but usually based on their understanding of the salient meanings in the source text.

For example, regarding the text “The young boys are playing basketball with great excitement and apparently they enjoy the fun of competition”, it can be found that there are two salient ideas, which are “game of the young” and “happiness of basketball game”, which we call “semantic units” of the text. The semantic units, instead of word-level information, mainly determine that the text can be classified into the target categories “youth” and “sports”.

Semantic units construct the semantic meaning of the whole text. To assign proper labels for text, the model should capture the core semantic units of the source text, the higher-level information compared with word-level information, and then assign the text labels based on its understanding of the semantic units. However, it is difficult to extract information from semantic units as the conventional attention mechanism focuses on extracting word-level information, which contains redundancy and irrelevant details.

In order to capture semantic units in the source text, we analyze the texts and find that the semantic units are often wrapped in phrases or sentences, connecting with other units with the help of contexts. Inspired by the idea of global encoding for summarization (Lin et al., 2018), we utilize the power of convolutional neural networks (CNN) to capture local interaction among words and generate representations of information of higher levels than the word, such as phrase or sentence. Moreover, to tackle the problem of long-term dependency, we design a multi-level dilated convolution for text to capture local correlation and long-term dependency without loss of coverage as we do not apply any form of pooling or strided convolution. Based on the annotations generated by our designed module and those by the original recurrent neural networks, we implement our hybrid attention mechanism with the purpose of capturing information at different levels, and furthermore, it can extract word-level information from the source context based on its attention on the semantic units.

In brief, our contributions are illustrated below:

- We analyze that the conventional attention mechanism is not useful for multi-label text classification, and we propose a novel model with multi-level dilated convolution to capture semantic units in the source text.
- Experimental results demonstrate that our model outperforms the baseline models and achieves the state-of-the-art performance on the dataset RCV1-v2 and Ren-CECps, and our model is competitive with the hierarchical models with the deterministic setting of sentence or phrase.
- Our analysis shows that compared with the conventional Seq2Seq model, our model with effective information extracted from the source context can better predict the labels of low frequency, and it is less influenced by the prior distribution of the label sequence.

2 Attention-based Seq2Seq for Multi-label Text Classification

As illustrated below, multi-label text classification has the potential to be regarded as a task of sequence prediction, as long as there are certain correlation patterns in the label data. Owing to the correlations among labels, it is possible to improve the performance of the model in this task by assigning certain label permutations for the label sequence and maximizing subset accuracy, which means that the label permutation and the corresponding attention-based Seq2Seq method are competent in learning the label classification and the label correlations. By maximizing the subset accuracy, the model can improve the performance of classification with the assistance of the information about the label correlations. Regarding label permutation, a straightforward method is to reorder the label data in accordance with the descending order by frequency, which shows satisfactory effects (Chen et al., 2017).

Multi-label text classification can be regarded as a Seq2Seq learning task, which is formulated as below. Given a source text $x = \{x_1, ..., x_i, ..., x_n\}$ and a target label sequence $y = \{y_1, ..., y_i, ..., y_m\}$, the Seq2Seq model learns to approximate the probability $P(y|x) = \prod_{t=1}^{m} P(y_t|y_{<t}, x)$, where $P(y_t|y_{<t}, x)$ is computed by the Seq2Seq model, which is commonly based on recurrent neural network (RNN).

The encoder, which is bidirectional Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1996), encodes the source text x from both directions and generates the source annotations h, where the annotations from both directions at each time step are concatenated ($h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$). To be specific, the computations of $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ are illustrated below:

$$\overrightarrow{h_i} = LSTM(x_i, \overrightarrow{h_{i-1}}, C_{i-1}) \quad (1)$$

$$\overleftarrow{h_i} = LSTM(x_i, \overleftarrow{h_{i-1}}, C_{i-1}) \quad (2)$$

We implement a unidirectional LSTM decoder to generate labels sequentially. At each time step t, the decoder generates a label y_t by sampling from a distribution of the target label set P_{vocab} until sampling the token representing the end of sentence, where:

$$P_{vocab} = g(y_{t-1}, c_t, s_{t-1}) \quad (3)$$

where $g(\cdot)$ refers to non-linear functions including the LSTM decoder, the attention mechanism as well as the softmax function for prediction. The attention mechanism generates c_t as shown in the following:

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i \quad (4)$$

$$\alpha_{t,i} = \frac{exp(e_{t,i})}{\sum_{j=1}^{n} exp(e_{t,j})} \quad (5)$$

$$e_{t,i} = s_{t-1}^{\top} W_a h_i \quad (6)$$

3 Problem

As we analyze the effects of the attention mechanism in multi-label text classification, we find that it contributes little to the improvement of the model's performance. To verify the effects of the attention mechanism, we conduct an ablation test to compare the performance of the Seq2Seq model without the attention mechanism and the attention-based SeqSeq model on the multi-label text classification dataset RCV1-v2, which is introduced in detail in Section 5.

Models	**HL(-)**	**P(+)**	**R(+)**	**F1(+)**
w/o attention	0.0082	0.883	0.849	0.866
+attention	0.0081	0.889	0.848	0.868

Table 1: Performances of the Seq2Seq models with and without attention on the RCV1-v2 test set, where HL, P, R, and F1 refer to hamming loss, micro-precision, micro-recall and micro-F_1. The symbol "+" indicates that the higher the better, while the symbol "-" indicates that the lower the better.

As is shown in Table 1, the Seq2Seq models with and without the attention mechanism demonstrate similar performances on the RCV1-v2 according to their scores of micro-F_1, a significant evaluation metric for multi-label text classification. This can be a proof that the conventional attention mechanism does not play a significant role in the improvement of the Seq2Seq model's performance. We hypothesize that the conventional attention mechanism does not meet the requirements of multi-label text classification. A common sense for such a classification task is that the classification should be based on the salient ideas of the source text. The semantic units, instead of word-level information, mainly determine that the text can be classified into the target categories "youth" and "sports". For each of a variety of texts, there are always semantic units that construct the semantic meaning of the whole text. Regarding an automatic system for multi-label text classification, the system should be able to extract the semantic units in the source text for better performance in classification. Therefore, we propose our model to tackle this problem.

4 Proposed Method

In the following, we introduce our proposed modules to improve the conventional Seq2Seq model for multi-label text classification. In general, it contains two components: multi-level dilated convolution (MDC) as well as hybrid attention mechanism.

4.1 Multi-level Dilated Convolution

On top of the representations generated by the original encoder, which is an LSTM in our model, we apply the multi-layer convolutional neural networks to generate representations of semantic units by capturing local correlations and long-term dependencies among words. To be specific, our CNN is a three-layer one-dimensional CNN. Fol-

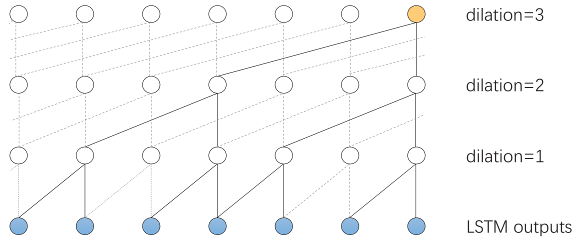

Figure 1: **Structure of Multi-level Dilated Convolution (MDC).** A example of MDC with kernel size $k = 2$ and dilation rates $[1, 2, 3]$. To avoid gridding effects, the dilation rates do not share a common factor other than 1.

lowing the previous work (Kalchbrenner et al., 2014) on CNN for NLP, we use one-dimensional convolution with the number of channels equal to the number of units of the hidden layer, so that the information at each dimension of a representation vector will not be disconnected as 2-dimension convolution does. Besides, as we are to capture semantic units in the source text instead of higher-level word representations, there is no need to use padding for the convolution.

A special design for the CNN is the implementation of dilated convolution. Dilation has become popular in semantic segmentation in computer vision in recent years (Yu and Koltun, 2015; Wang et al., 2018), and it has been introduced to the fields of NLP (Kalchbrenner et al., 2016) and speech processing (van den Oord et al., 2016). Dilated convolution refers to convolution inserted with "holes" so that it is able to remove the negative effects such as information loss caused by common down-sampling methods, such as max-pooling and strided convolution. Besides, it is able to expand the receptive fields at the exponential level without increasing the number of parameters. Thus, it becomes possible for dilated convolution to capture longer-term dependency. Furthermore, with the purpose of avoiding gridding effects caused by dilation (e.g., the dilated segments of the convolutional kernel can cause missing of vital local correlation and break the continuity between word representations), we implement a multi-level dilated convolution with different dilation rates at different levels, where the dilation rates are hyperparameters in our model.

Instead of using the same dilation rate or dilation rates with the common factor, which can cause gridding effects, we apply multi-level dilated convolution with different dilation rates, such as [1,2,3]. Following Wang et al. (2018), for N layers of 1-dimension convolution with kernel size K with dilation rates $[r_1, ..., r_N]$, the maximum distance between two nonzero values is $max(M_{i+1} - 2r_i, M_{i+1} - 2(M_{i+1} - r_i), r_i)$ with $M_N = r_N$, and the goal is $M_2 \leq K$. In our experiments, we set the dilation rates to [1, 2, 3] and K to 3, and we have $M_2 = 2$. The implementations can avoid the gridding effects and allows the top layer to access information between longer distance without loss of coverage. Moreover, as there may be irrelevant information to the semantic units at a long distance, we carefully design the dilation rates to [1, 2, 3] based on the performance in validation, instead of others such as [2, 5, 9], so that the top layer will not process the information among overlong distance and reduce the influence of unrelated information. Therefore, our model can generate semantic unit representations from the information at phrase level with small dilation rates and those at sentence level with large dilation rates.

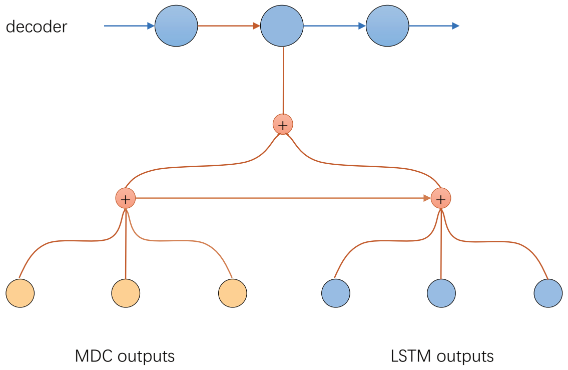

Figure 2: **Structure of Hybrid Attention.** The blue circles at the left bottom represent the source annotations generated by the LSTM encoder, the yellow circles at the right bottom represent the semantic unit representations generated by MDC, and the blue circles at the top represent the LSTM decoder outputs. At each decoding time step, the output of the LSTM attends to the semantic unit representations first, and then the new representation incorporated with high-level information attends to the source annotations.

4.2 Hybrid Attention

As we have annotations from the RNN encoder and semantic unit representations from the MDC, we design two types of attention mechanism to evaluate the effects of information of different levels. One is the common attention mechanism, which attends to the semantic unit representations instead of the source word annotations as the conventional does, the other is our designed hybrid attention mechanism to incorporate information of the two levels.

The idea of hybrid attention is motivated by memory networks (Sukhbaatar et al., 2015) and multi-step attention (Gehring et al., 2017). It can be regarded as the attention mechanism with multiple "hops", with the first hop attending to the higher-level semantic unit information and the second hop attending to the lower-level word unit information based on the decoding and the first attention to the semantic units. Details are presented below.

For the output of the decoder at each time step, it not only attends to the source annotations from the RNN encoder as it usually does but also attends to the semantic unit representations from the MDC. In our model, the decoder output first pays attention to the semantic unit representations from the MDC to figure out the most relevant semantic units and generates a new representation based on the attention. Next, the new representation with both the information from the decoding process as well as the attention to the semantic units attends to the source annotations from the LSTM encoder, so it can extract word-level information from the source text with the guidance of the semantic units, mitigating the problem of irrelevance and redundancy.

To be specific, for the source annotations from the LSTM encoder $h = \{h_1, ..., h_i, ..., h_n\}$ and the semantic unit representations $g = \{g_1, ..., g_i, ..., g_m\}$, the decoder output s_t first attends to the semantic unit representations g and generates a new representation s'_t. Then the new representation s'_t attends to the source annotations h and generates another representation $\tilde{s}_t$ following the identical attention mechanism as mentioned above. In the final step, the model generates o_t for the prediction of y_t, where:

$$o_t = s'_t \oplus \tilde{s}_t \tag{7}$$

For comparison, we also propose another type of attention called "additive attention", whose experimental results are in the ablation test. In this mechanism, instead of paying attention to the two types of representations step by step as mentioned above, the output of the LSTM decoder s_t attends to the semantic unit representations g and the source annotations h respectively to generate s'_t and $\tilde{s}_t$, which are finally added element-wisely for the final output o_t.

5 Experiment Setup

In the following, we introduce the datasets and our experiment settings as well as the baseline models that we compare with.

5.1 Datasets and Preprocessing

Reuters Corpus Volume I (RCV1-v2)[2]: The dataset (Lewis et al., 2004) consists of more than 800k manually categorized newswire stories by Reuters Ltd. for research purpose, where each story is assigned with multiple topics. The total number of topics is 103. To be specific, the training set contains around 802414 samples, while the development set and test set contain 1000 samples respectively. We filter the samples whose lengths are over 500 words in the dataset, which removes about 0.5% of the samples in the training, development and test sets. The vocabulary size is set to 50k words. Numbers as well as out-of-vocabulary words are masked by special tokens "#" and "UNK". For label permutation, we apply the descending order by frequency following Kim (2014).

Ren-CECps: The dataset is a sentence corpus collected from Chinese blogs, annotated with 8 emotion tags *anger*, *anxiety*, *expect*, *hate*, *joy*, *love*, *sorrow* and *surprise* as well as 3 polarity tags *positive*, *negative* and *neutral*. The dataset contains 35096 sentences for multi-label text classification. We apply preprocessing for the data similar to that for the RCV1-v2, which are filtering samples of over 500 words, setting the vocabulary size to 50k and applying the descending order by frequency for label permutation.

5.2 Experiment Settings

We implement our experiments in PyTorch on an NVIDIA 1080Ti GPU. In the experiments, the batch size is set to 64, and the embedding size and the number of units of hidden layers are 512. We use Adam optimizer (Kingma and Ba, 2014) with the default setting $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$. The learning rate is initialized to 0.0003 based on the performance on the development set, and it is halved after every epoch

---

[2] `http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm`

Models	HL(-)	P(+)	R(+)	F1(+)
BR	0.0086	0.904	0.816	0.858
CC	0.0087	0.887	0.828	0.857
LP	0.0087	0.896	0.824	0.858
CNN	0.0089	**0.922**	0.798	0.855
CNN-RNN	0.0085	0.889	0.825	0.856
S2S	0.0082	0.883	0.849	0.866
S2S+Attn	0.0081	0.889	0.848	0.868
Our Model	**0.0072**	0.891	**0.873**	**0.882**

Table 2: Performance on the RCV1-V2 test set. HL, P, R, and F1 denote hamming loss, micro-precision, micro-recall and micro-F_1, respectively ($p < 0.05$).

Models	HL(-)	P(+)	R(+)	F1(+)
BR	**0.1663**	**0.649**	0.472	0.546
CC	0.1828	0.572	0.551	0.561
LP	0.1902	0.556	0.517	0.536
CNN	0.1726	0.628	0.512	0.565
CNN-RNN	0.1876	0.576	0.538	0.556
S2S	0.1814	0.587	0.571	0.579
S2S+Attn	0.1793	0.589	0.573	0.581
Our Model	0.1782	0.593	**0.585**	**0.590**

Table 3: Performance of the models on the Ren-CECps test set. HL, P, R, and F1 denote hamming loss, micro-precision, micro-recall and micro-F_1, respectively ($p < 0.05$).

of training. Gradient clipping is applied with the range [-10, 10].

Following the previous studies (Zhang and Zhou, 2007; Chen et al., 2017), we choose hamming loss and micro-F_1 score to evaluate the performance of our model. Hamming loss refers to the fraction of incorrect prediction (Schapire and Singer, 1999), and micro-F_1 score refers to the weighted average F_1 score. For reference, the micro-precision as well as micro-recall scores are also reported. To be specific, the computations of Hamming Loss (HL) micro-F_1 score are illustrated below:

$$HL = \frac{1}{L} \sum \mathbb{I}(y \neq \hat{y}) \tag{8}$$

$$microF_1 = \frac{\sum_{j=1}^{L} 2tp_j}{\sum_{j=1}^{L} 2tp_j + fp_j + fn_j} \tag{9}$$

where tp_j, fp_j and fn_j refer to the number of true positive examples, false positive examples and false negative examples respectively.

5.3 Baseline Models

In the following, we introduce the baseline models for comparison for both datasets.

- **Binary Relevance (BR)** (Boutell et al., 2004) transforms the MLC task into multiple single-label classification problems.
- **Classifier Chains (CC)** (Read et al., 2011) transforms the MLC task into a chain of binary classification problems to model the correlations between labels.
- **Label Powerset (LP)** (Tsoumakas and Katakis, 2006) creates one binary classifier for every label combination attested in the training set.
- **CNN** (Kim, 2014) uses multiple convolution kernels to extract text feature, which is then input to the linear transformation layer followed by a sigmoid function to output the probability distribution over the label space.
- **CNN-RNN** (Chen et al., 2017) utilizes CNN and RNN to capture both global and local textual semantics and model label correlations.
- **S2S** and **S2S+Attn** (Sutskever et al., 2014; Bahdanau et al., 2014) are our implementation of the RNN-based sequence-to-sequence models without and with the attention mechanism respectively.

6 Results and Discussion

In the following sections, we report the results of our experiments on the RCV1-v2 and Ren-CECps. Moreover, we conduct an ablation test and the comparison with models with hierarchical models with the deterministic setting of sentence or phrase, to illustrate that our model with learnable semantic units possesses a clear advantage over the baseline models. Furthermore, we demonstrate that the higher-level representations are useful for the prediction of labels of low frequency in the dataset so that it can ensure that the model is not strictly learning the prior distribution of the label sequence.

6.1 Results

We present the results of our implementations of our model as well as the baselines on the RCV1-v2 on Table 2. From the results of the conventional baselines, it can be found that the classical methods for multi-label text classification still own competitiveness compared with the machine-learning-based and even deep-learning-based methods, instead of the Seq2Seq-based models. Regarding the Seq2Seq model, both the S2S and the S2S+Attn achieve improvements on the dataset, compared with the baselines above. However, as mentioned previously, the attention mechanism does not play a significant role in the Seq2Seq model for multi-label text classification. By contrast, our proposed mechanism, which is label-classification-oriented, can take both the information of semantic units and that of word units into consideration. Our proposed model achieves the best performance in the evaluation of Hamming loss and micro-F_1 score, which reduces 9.8% of Hamming loss and improves 1.3% of micro-F_1 score, in comparison with the S2S+Attn.

We also present the results of our experiments on Ren-CECps. Similar to the models' performance on the RCV1-v2, the conventional baselines except for Seq2Seq models achieve lower performance on the evaluation of micro-F_1 score compared with the Seq2Seq models. Moreover, the S2S and the S2S+Attn still achieve similar performance on micro-F_1 on this dataset, and our proposed model achieves the best performance with the improvement of 0.009 micro-F_1 score. An interesting finding is that the Seq2Seq models do not possess an advantage over the conventional baselines on the evaluation of Hamming Loss. We observe that there are fewer labels in the Ren-CECps than in the RCV1-v2 (11 and 103). As our label data are reordered according to the descending order of label frequency, the Seq2Seq model is inclined to learn the frequency distribution, which is similar to a long-tailed distribution. However, regarding the low-frequency labels with only a few samples, their amounts are similar, whose distributions are much more uniform than that of the whole label data. It is more difficult for the Seq2Seq model to classify them correctly while the model is approximating the long-tailed distribution compared with the conventional baselines. As Hamming loss reflects the average incorrect prediction, the errors in classifying into low-frequency labels will lead to a sharper increase in Hamming Loss, in comparison with micro-F_1 score.

6.2 Ablation Test

To evaluate the effects of our proposed modules, we present an ablation test for our model. We remove certain modules to control variables so that their effects can be fairly compared. To be specific, besides the evaluation of the conventional attention mechanism mentioned in Section 3, we evaluate the effects of hybrid attention and its modules. We demonstrate the performance of five models with different attention implementation for comparison, which are model without attention, one with only attention to the source annotations from LSTM, one with only attention to the semantic unit representations from the MDC, one with the attention to both the source annotations and semantic unit representations (additive) and hybrid attention, respectively. Therefore, the effects of each of our proposed modules, including MDC and hybrid attention, can be evaluated individually without the influence of the other modules.

Models	HL(-)	P(+)	R(+)	F1(+)
w/o attention	0.0086	**0.904**	0.816	0.871
attention	0.0087	0.887	0.828	0.869
MDC	0.0074	0.889	0.871	0.880
additive	0.0073	0.888	0.871	0.879
hybrid	**0.0072**	0.891	**0.873**	**0.882**

Table 4: Performance of the models with different attention mechanisms on the RCV1-V2 test set. HL, P, R, and F1 denote hamming loss, micro-precision, micro-recall and micro-F_1, respectively ($p < 0.05$).

Results in Table 4 reflect that our model still performs the best in comparison with models with the other types of attention mechanism. Except for the insignificant effect of the conventional attention mechanism mentioned above, it can be found that the high-level representations generated by the MDC contribute much to the performance of the Seq2Seq model for multi-label text classification, which improves about 0.9 micro-F_1 score. Moreover, simple additive attention mechanism, which is equivalent to the element-wise addition of the representations of MDC and those of the conventional mechanism, achieves similar performance to the single MDC, which also demonstrates that conventional attention mechanism in this task makes little contribution. As to our proposed hybrid attention, which is a relatively complex combination of the two mechanisms, can improve the performance of MDC. This shows that although conventional attention mechanism for word-level information does not influence the performance of the SeqSeq model significantly, the hybrid attention which extracts word-level infor-

Models	HL(-)	P(+)	R(+)	F1(+)
Hier-5	0.0075	0.887	0.869	0.878
Hier-10	0.0077	0.883	0.873	0.878
Hier-15	0.0076	0.879	0.879	0.879
Hier-20	0.0076	0.876	**0.881**	0.878
Our model	**0.0072**	**0.891**	0.873	**0.882**

Table 5: Performance of the hierarchical model and our model on the RCV1-V2 test set. Hier refers to hierarchical model, and the subsequent number refers to the length of sentence (word) for sentence-level representations ($p < 0.05$).

mation based on the generated high-level semantic information can provide some information about important details that are relevant to the most contributing semantic units.

6.3 Comparison with the Hierarchical Models

Another method that can extract high-level representations is a heuristic method that manually annotates sentences or phrases first and applies a hierarchical model for high-level representations. To be specific, the method does not only apply an RNN encoder to the word representations but also to sentence representations. In our reimplementation, we regard the representation from the LSTM encoder at the time step of the end of each sentence as the sentence representation, and we implement another LSTM on top of the original encoder that receives sentence representations as input so that the whole encoder can be hierarchical. We implement the experiment on the dataset RCV1-v2. As there is no sentence marker in the dataset RCV1-v2, we set a sentence boundary for the source text and we apply a hierarchical model to generate sentence representations.

Compared with our proposed MDC, the hierarchical model for the high-level representations is relatively deterministic since the sentences or phrases are predefined manually. However, our proposed MDC learns the high-level representations through dilated convolution, which is not restricted by the manually-annotated boundaries. Through the evaluation, we expect to see if our model with multi-level dilated convolution as well as hybrid attention can achieve similar or better performance than the hierarchical model. Moreover, we note that the number of parameters of the hierarchical model is more than that of our model, which are 47.24M and 45.13M respectively. Therefore, it is obvious that our model does not possess the advantage of parameter number in the comparison.

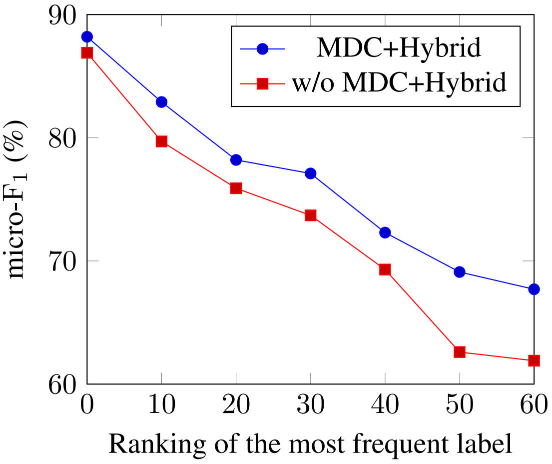

Figure 3: **Micro-F$_1$ scores of our model and the baseline on the evaluation of labels of different frequency.** The x-axis refers to the ranking of the most frequent label in the labels for classification, and the y-axis refers to the micro-F$_1$ score performance.

We present the results of the evaluation on Table 5, where it can be found that our model with fewer parameters still outperforms the hierarchical model with the deterministic setting of sentence or phrase. Moreover, in order to alleviate the influence of the deterministic sentence boundary, we compare the performance of different hierarchical models with different boundaries, which sets the boundaries at the end of every 5, 10, 15 and 20 words respectively. The results in Table 5 show that the hierarchical models achieve similar performances, which are all higher than the performances of the baselines. This shows that high-level representations can contribute to the performance of the Seq2Seq model on the multi-label text classification task. Furthermore, as these performances are no better than that of our proposed model, it can reflect that the learnable high-level representations can contribute more than deterministic sentence-level representations, as it can be more flexible to represent information of diverse levels, instead of fixed phrase or sentence level.

6.4 Error Analysis

Another finding in our experiments is that the model's performance on low-frequency label classification is lower than that on high-frequency label classification. This problem is also reflected in our report of the experimental results on the Ren-CECps. The decrease in performance is reasonable since the model is sensitive to the amount of data, especially on small datasets such as Ren-CECps. We also hypothesize that this error comes from the essence of the Seq2Seq model. As the frequency of our label data is similar to a long-tailed distribution and the data are organized by descending order of label frequency, the Seq2Seq model is inclined to model the distribution. As the frequency distribution of the low-frequency labels is relatively uniform, it is much harder for it to model the distribution.

In contrast, as our model is capable of capturing deeper semantic information for the label classification, we believe that it is more robust to the classification of low-frequency labels with the help of the information from multiple levels. We remove the top 10, 20, 30, 40, 50 and 60 most frequent labels subsequently, and we evaluate the performance of our model and the baseline Seq2Seq model on the classification of these labels. Figure 3 shows the results of the models on label data of different frequency. It is obvious that although the performances of both models decrease with the decrease of the label frequency, our model continues to perform better than the baseline on all levels of label frequency. In addition, the gap between the performances of the two models continues to increase with the decrease of label frequency, demonstrating our model's advantage over the baseline on classifying low-frequency labels.

7 Related Work

The current models for the multi-label classification task can be classified into three categories: problem transformation methods, algorithm adaptation methods, and neural network models.

Problem transformation methods decompose the multi-label classification task into multiple single-label learning tasks. The BR algorithm (Boutell et al., 2004) builds a separate classifier for each label, causing the label correlations to be ignored. In order to model label correlations, Label Powerset (LP) (Tsoumakas and Katakis, 2006) creates one binary classifier for every label combination attested in the training set and Classifier Chains (CC) (Read et al., 2011) connects all classifiers in a chain through feature space.

Algorithm adaptation methods adopt specific learning algorithms to the multi-label classification task without requiring problem transformations. Clare and King (2001) constructed decision tree based on multi-label entropy to perform classification. Elisseeff and Weston (2002) adopted a Support Vector Machine (SVM) like learning system to handle multi-label problem. Zhang and Zhou (2007) utilized the k-nearest neighbor algorithm and maximum a posteriori principle to determine the label set of each sample. Fürnkranz et al. (2008) made ranking among labels by utilizing pairwise comparison. Li et al. (2015) used joint learning predictions as features.

Recent studies of multi-label text classification have turned to the application of neural networks, which have achieved great success in natural language processing. Zhang and Zhou (2006) implemented the fully-connected neural networks with pairwise ranking loss function. Nam et al. (2013) changed the ranking loss function to the cross-entropy loss to better the training. Kurata et al. (2016) proposed a novel neural network initialization method to treat some neurons as dedicated neurons to model label correlations. Chen et al. (2017) incorporated CNN and RNN so as to capture both global and local semantic information and model high-order label correlations. (Nam et al., 2017) proposed to generate labels sequentially, and Yang et al. (2018); Li et al. (2018) both adopted the Seq2Seq, one with a novel decoder and one with a soft loss function respectively.

8 Conclusion

In this study, we propose our model based on the multi-level dilated convolution and the hybrid attention mechanism, which can extract both the semantic-unit-level information and word-level information. Experimental results demonstrate that our proposed model can significantly outperform the baseline models. Moreover, the analyses reflect that our model is competitive with the deterministic hierarchical models and it is more robust to classifying the low-frequency labels than the baseline.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028) and the National Thousand Young Talents Program. Xu Sun is the corresponding author of this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Fernando Benites and Elena Sapozhnikova. 2015. Haram: a hierarchical aram neural network for large-scale text classification. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 847–854. IEEE.

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. 2017. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *IJCNN 2017*, pages 2377–2383.

Amanda Clare and Ross D King. 2001. Knowledge discovery in multi-label phenotype data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer.

André Elisseeff and Jason Weston. 2002. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687.

Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. 2008. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML 2017*, pages 1243–1252.

Sepp Hochreiter and Jürgen Schmidhuber. 1996. LSTM can solve hard long time lag problems. In *NIPS, 1996*, pages 473–479.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR*, abs/1610.10099.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL 2014*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP 2014*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *NAACL HLT 2016*, pages 521–526.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Li Li, Houfeng Wang, Xu Sun, Baobao Chang, Shi Zhao, and Lei Sha. 2015. Multi-label text categorization with joint learning predictions-as-features method. In *EMNLP 2015*, pages 835–839.

Wei Li, Xuancheng Ren, Damai Dai, Yunfang Wu, Houfeng Wang, and Xu Sun. 2018. Sememe prediction: Learning semantic knowledge from unstructured textual wiki descriptions. *CoRR*, abs/1808.05437.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *ACL 2018*, pages 163–169.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP 2015*, pages 1412–1421.

Jinseok Nam, Jungi Kim, Iryna Gurevych, and Johannes Fürnkranz. 2013. Large-scale multi-label text classification - revisiting neural networks. *CoRR*, abs/1312.5419.

Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *NIPS 2017*, pages 5419–5429.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. In *ISCA Speech Synthesis Workshop 2016*, page 125.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3):333.

Robert E Schapire and Yoram Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS, 2015*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 2014*, pages 3104–3112.

Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3).

Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison W. Cottrell. 2018. Understanding convolution for semantic segmentation. In *WACV 2018*, pages 1451–1460.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: sequence generation model for multi-label classification. In *COLING 2018*, pages 3915–3926.

Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122.

Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

Min-Ling Zhang and Zhi-Hua Zhou. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048.

MCapsNet: Capsule Network for Text with Multi-Task Learning

Liqiang Xiao[1,2], **Honglun Zhang**[1,2], **Wenqing Chen**[1,2], **Yongkun Wang**[3], **Yaohui Jin**[1,2]
[1] State Key Lab of Advanced Optical Communication System and Network,
Shanghai Jiao Tong University
[2] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[3] Network and Information Center, Shanghai Jiao Tong University
{jinyh}@sjtu.edu.cn

Abstract

Multi-task learning has an ability to share the knowledge among related tasks and implicitly increase the training data. However, it has long been frustrated by the interference among tasks. This paper investigates the performance of capsule network for text, and proposes a capsule-based multi-task learning architecture, which is unified, simple and effective. With the advantages of capsules for feature clustering, proposed task routing algorithm can cluster the features for each task in the network, which helps reduce the interference among tasks. Experiments on six text classification datasets demonstrate the effectiveness of our models and their characteristics for feature clustering.

1 Introduction

Multi-task learning (MTL) has achieved a great success in the field of natural language processing, which can share the knowledge among multiple tasks, implicitly increasing the volume of training data. The combination of multi-task learning and deep neural networks (DNNs) generates a further synergy via the regularization effect on the DNNs (Collobert and Weston, 2008), which helps alleviate the overfitting and learn a more universal presentation.

Inspired by this, more DNN-based multi-task learning models are proposed to improve the performance. As depicted in Figure 1, they can be categorized into three groups by structure: tree scheme (Collobert and Weston, 2008; Liu et al., 2015), parallel scheme (Liu et al., 2016) and mediate scheme (Zhang et al., 2017). Tree scheme reuses some shallow layers of network and separates the higher layers for different tasks, which is the most common architecture for MTL but can only share the low-level knowledge. To share deeper level knowledge among the tasks, more layers are linked in parallel and mediate schemes. But this would severely suffer from the interference among tasks. Useless features following the helpful ones are fully shared among tasks, which may contaminate the feature spaces of tasks by useless ones. Besides, models under these two schemes usually employ multiple subnets in the structures, which would contain more parameters and are hard to train.

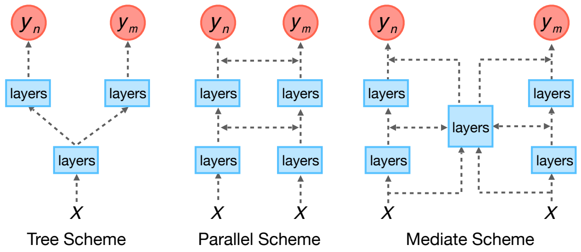

Figure 1: Three schemes of multi-task learning

Apparently, there is a contradiction between knowledge sharing and interference. Sharing too much between tasks would inevitably bring about the interference that feature space for each task may be contaminated by others. Shared useless features may mislead the prediction of network. This dilemma is caused by the lack of management for sharing process, in which the network can not discriminate the features and collect the appropriate features for each task.

Capsule network (Hinton et al., 2011; Mousa et al., 2017; Hinton et al., 2018) embeds the features into capsules and connects the neighbor layers via "routing-by-agreement". The *dynamic routing* algorithm has an ability to decide the routes of capsules, namely, to cluster the features for each category. So, intuitively this property of capsule network can be employed in MTL to discriminate the features for tasks.

In this paper, we explore the performance of capsule network for text (**CapsNet-1**, **CapsNet-1**) and show the benefits and potential of cap-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4565–4574
Brussels, Belgium, October 31 - November 4, 2018.

sule network for NLP. Then we mainly propose a capsule-based architecture for multi-task learning (**McapsNet**), which is unified, simple, effective and can cluster the feature for tasks. We designed a **Task Routing** algorithm to route the feature flows to tasks and vote for the classes, which can reduce the interference. In extensive experiences, our approach achieves competitive results in single-task scenario and shows obvious improvement in multi-task scenario, which proves our approach effective and its ability to reduce the interference among multiple tasks. Also, our visualization experiments intuitively show the feature clustering mechanism and how it helps make right predictions.

The contribution of this paper are three-folds:

- This paper investigates the performance of capsule network on text and designs two effective capsule-based models for text classification, which give clear improvement to several benchmarks.
- We novelly combine the capsule and multi-task learning, which can help reduce the interference among tasks.
- Proposed task routing algorithm can route the capsules to multiple tasks, by which the features is clustered into groups for tasks.

2 Convolutional Neural Network and Multi-Task Learning

Capsule network is based on the convolutional neural network (CNN) and uses a lot of convolution operations. The main differences between them are that capsule network uses vectors to represent the features and discards the pooling operation. CNN is good at feature extraction and can capture short and long range relations through the feature graph over text sequence (Kalchbrenner et al., 2014; Kim, 2014). In this section, we provide some formulations for CNN and some background knowledge for multi-task learning.

2.1 Single-Task CNN for Text Classification

Given a text sequence $x_{1:l} = x_1 x_2 \cdots x_l$ of length l, the target of CNN is to predict the category $\hat{y}$ of $x_{1:l}$ from a set $\{y_1, y_2, \cdots, y_C\}$, or a one-hot form of $\hat{\mathbf{y}}$, where C is the class number. Using $f(\cdot)$ denote the network, the prediction process can be formalized as $f(x_1, x_2, \cdots, x_l) = \hat{y}$.

For details, convolutional neural network $f(\cdot)$ first uses a lookup table to embed the word sequence $x_{1:l}$ into vectors $\mathbf{x}$. Then CNN produces the representation of the input sequence by stacking the layers of convolution, pooling and fully-connected in order.

$$\mathbf{F} = \mathbf{K} * \mathbf{x} \tag{1}$$

$$\hat{\mathbf{F}} = p(\mathbf{F}) \tag{2}$$

$$\hat{\mathbf{y}} = \mathbf{w}\hat{\mathbf{F}} + \mathbf{b}, \tag{3}$$

where $\mathbf{K}$ is the kernel of convolution operation $*$; $p(\cdot)$ denotes the pooling operation; $\mathbf{F}$ and $\hat{\mathbf{F}}$ represent the feature maps; $\mathbf{w}$ and $\mathbf{b}$ denote the weight and bias respectively in fully connected layer.

The parameters of the network are optimized via all kinds of SGD (stochastic gradient decent) algorithms to minimize the loss between prediction $\hat{\mathbf{y}}$ and ground truth label $\tilde{\mathbf{y}}$

$$l(\hat{\mathbf{y}}, \tilde{\mathbf{y}}) = -\sum_{i=1}^{N}\sum_{j=1}^{C} \tilde{\mathbf{y}}_j^i \log(\hat{\mathbf{y}}_j^i), \tag{4}$$

where i, j enumerate the training samples and classes respectively.

2.2 Multi-Task Learning

Multi-task learning model is usually the variant or combination of single-task ones (CNNs, RNNs or DNNs) like the architectures illustrated in Figure 1. Given K text classification tasks $\{T_1, T_2, \cdots, T_K\}$, a multi-task learning model $f(\cdot)$ shall have the ability to make prediction for samples $\mathbf{x}_i^{(k)}$ from each task T_k.

$$\hat{\mathbf{y}}_i^{(k)} = f(\mathbf{x}_i^{(k)}) \tag{5}$$

And the overall loss for all the tasks is usually a linear combination of the costs for each.

$$L = -\sum_{k=1}^{K} \lambda_k \sum_{i=1}^{N_k}\sum_{j=1}^{C_k} \tilde{\mathbf{y}}_{i,j}^{(k)} \log \hat{\mathbf{y}}_{i,j}^{(k)} \tag{6}$$

where λ_k,N_k and C_k denote the weight, number of training samples and class number of task T_k.

3 Capsule Networks for Text

Capsule network (CapsNet) is first proposed by Sabour et al. (2017) for image classification, which is position sensitive and shows strong performance on some classification tasks. As depicted in Figure 2, we propose several capsule networks for text, which are suitable for text representation and multi-task learning. They are comprised of convolutional layer, primary capsule layer and class capsule layer. In the rest of this section, we will first give the formulation of single-task capsule networks (CapsNet-1 and CapsNet-2) for text classification, and then transfer it into a multi-task version (McapsNet).

3.1 Primary Capsule Layer

Given an embedded sample of $\mathbf{x} \in \mathbb{R}^{l \times d}$ with length of l and word vectors of d-dimension, capsule network first employs a plain convolution layer to extract the local features from N-grams. Each kernel $\mathbf{K}_i$ with a bias $\mathbf{b}$ emits a feature maps $\mathbf{F}_i$ by convolution.

$$\mathbf{F}_i = \mathbf{x} * \mathbf{K}_i + \mathbf{b} \tag{7}$$

By assembling I feature maps together, we have a I-channel layer

$$\mathbf{F} = [\mathbf{F}_1, \mathbf{F}_2, \cdots, \mathbf{F}_I] \tag{8}$$

The generated feature maps are then fed into the primary capsule layer, piecing the instantiated parts together via another convolution. Primary capsules use vectors instead of scales to preserve the instantiated parameters for each feature, which can not only represent the intensity of activation but also record some details of the instantiated parts in input. In this way, capsule can be regarded as a short representation of instantiated parts that are detected by kernel.

Sliding over the feature map $\mathbf{F}$, each kernel $\mathbf{K}_j$ would output a series of capsules $p_j \in \mathbb{R}^d$ of d-dimension. These capsules comprise a channel $\mathbf{P}_j$ of primary capsule layer.

$$\mathbf{P}_j = g(\mathbf{K}_j * \mathbf{F} + \mathbf{b}) \tag{9}$$

where g is the nonlinear squash function and b is the capsule bias term. All the J channels can be arranged as

$$\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_J] \tag{10}$$

3.2 Connection Between Capsule Layers

Capsule network generates the capsules in next layer using "routing-by-agreement". This process takes the place of pooling operation that usually discards the location information, which helps augment the robust of the network and also helps cluster features for prediction.

Between two neighbor layers l and $l + 1$, a "prediction vectors" $\hat{\mathbf{u}}_{j|i}$ is first computed from the capsule $\mathbf{u}_i$ in lower layer l, by multiplying a weight matrix $\mathbf{W}_{ij}$

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i \tag{11}$$

Then, in the higher layer $l+1$ a capsule $\mathbf{s}_j$ is generated by the linear combination of all the prediction vectors with weights c_{ij}

$$\mathbf{s}_j = \sum_i c_{ij}\mathbf{u}_{j|i} \tag{12}$$

where c_{ij} are coupling coefficients decided by the iterative **dynamic routing** process.

Coupling coefficients are calculated by a "routing softmax" function on original logits b_{ij}, which are the log prior probability that capsule i should be coupled to capsule j.

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_j exp(b_{ij})} \tag{13}$$

This process of "routing softmax" guarantee the sum of all the coefficients for capsule j is 1.

The length of capsule represents the probability that the input sample has the object capsule describes, that is the activation of capsule. So the length of capsule is limited in range $[0, 1]$ with a non-linear squashing function.

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|^2} \tag{14}$$

By that, the short vectors are pushed to shrink to zero length, and long ones are pushed to one.

3.3 Dynamic Routing

Suppose capsule layer l has been generated. We have to decide the intensity of the connections between capsule i and j from l-th layer to $(l + 1)$-th layer, that is the coupling coefficient c_{ij}. The initial digit of coupling coefficient b_{ij} is updated with routing by agreement a_{ij}, which is calculated by a scale product between capsules in two layers.

$$a_{ij} = \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j \tag{15}$$

Value of agreement a_{ij} is added to the digit to calculated the capsules in the next layer.

$$b_{ij} \leftarrow b_{ij} + a_{ij} \tag{16}$$

And the whole process for update (Eq.(13)→(12) →(14)→(15)→(16)) is conducted iteratively to optimize the coupling coefficients and the capsules in the next layer.

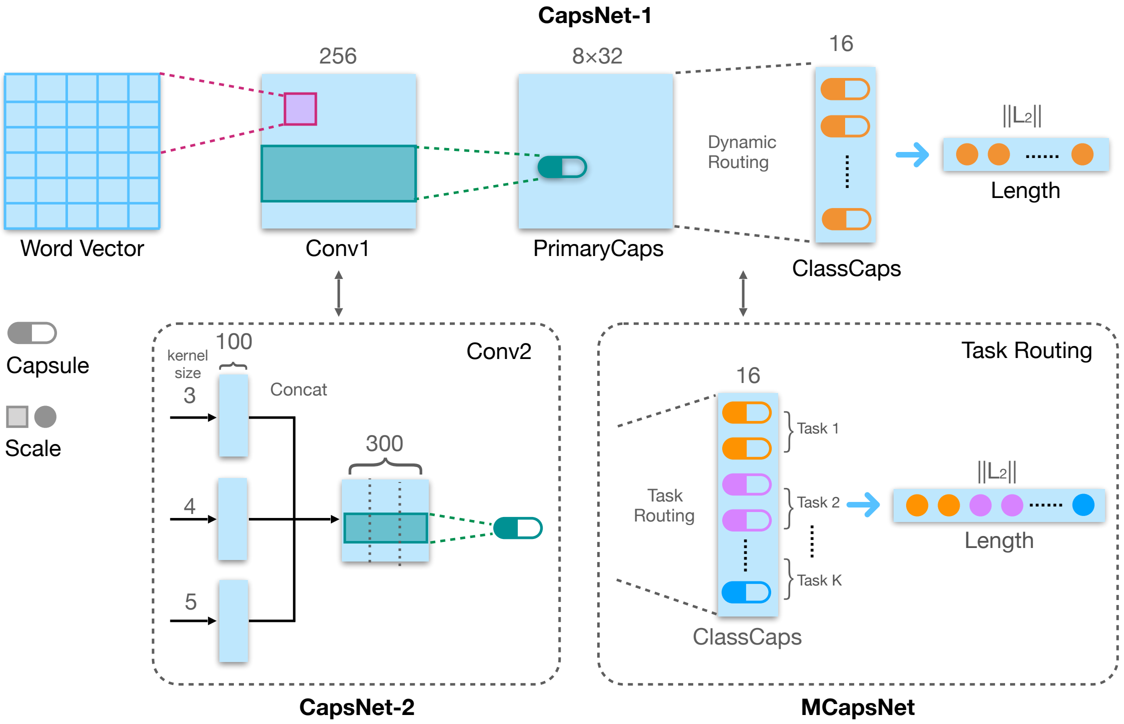

Figure 2: Architectures of capsule networks for text

3.4 Class Capsule Layer and Loss

Class capsule layer, as the top-level layer, is comprised of C class capsules, in which each one corresponds to a category. The length of instantiated parameters in each capsule represents the probability that the input sample belongs to this category, and the direction of each set of instantiated parameters preserves the characteristics of the features, which could be regarded as an encoded vector for the input sample.

Margin Loss To increase the difference between the lengths of classes, CapsNet utilizes a separated margin loss:

$$\begin{aligned} L_j =& G_j \max(0, m^+ - \|\mathbf{v}_j\|)^2 + \\ & \lambda(1-G_j)\max(0, m^- - \|\mathbf{v}_j\|)^2 \end{aligned} \tag{17}$$

where $\mathbf{v}_j$ is the capsule for class j; m^+ and m^- is the top and bottom margins respectively, which help push the length to shrink beyond two margins; $G_j = 1$ if and only if class j is the ground truth:

$$G_j = \begin{cases} 1 & \tilde{\mathbf{y}}_j = 1 \\ 0 & \tilde{\mathbf{y}}_j = 0 \end{cases} \tag{18}$$

λ is the weight for the absent classes, which reduces the weight of absent classes, avoiding shrinking the lengths of all the capsules too much at prophase training. In this paper, λ is set to 0.5.

Orphan Category A drawback of CapsNet is that it tends to account for everything in the input sampling, including some "background" information such as stop word and punctuations that would interfere the prediction. So an orphan category is added in class capsules in the output layer, which belongs to none of the categories of the task. The orphan category would help collect the less contributive capsules that contain too much "background" information, which reduces the interference for normal categories.

3.5 Substitutional Modules for Multi-Task

Task Routing

Dynamic routing algorithm is first proposed by Sabour et al. (2017), which displaces the pooling operation used in conventional convolution neural network. It maintains the position information for features, which is beneficial to both image and text representation. More importantly, this routing-by-agreement method has an ability to cluster the features into each class.

Inspirited by this, we employ this thought to cluster the features for different tasks and propose the **Task Routing** algorithm, which gives a simple and efficient solution to the problem that existing

Algorithm 1 Task Routing Algorithm

```
function ROUTING($\hat{\mathbf{u}}^{(k)}_{j|i}$, $a^{(k)}_{ij}$, $r$, $l$ )
    for $i = 0 \to r$ do
        for all capsule $i$ in layer $l$ and capsule $j$ in task $k$:
            $\mathbf{c}^{(k)}_i = softmax(\mathbf{b}^{(k)}_i)$
        for all capsule $j$ in layer $l + 1$:
            $\mathbf{v}^{(k)}_j = g(\sum_i c^{(k)}_{j|i} \hat{\mathbf{u}}^{(k)}_{j|i})$
        for all capsule $i$ in layer $l$ and capsule $j$ in task $k$:
            $b^{(k)}_{ij} = b^{(k)}_{ij} + a^{(k)}_{ij}$, $a^{(k)}_{ij} = \hat{\mathbf{u}}^{(k)}_{j|i} \cdot \mathbf{v}^{(k)}_j$
    end for
    return $\mathbf{v}^{(k)}_j$
end function
```

MTL models (Liu et al., 2017; Ruder et al., 2017; Fang et al., 2017) want to address: *"What feature should be shared and what should not among tasks?"* By that, network can decide the contribution of the features for each tasks and set the appropriate coupling coefficients between features and tasks.

More concretely, we introduce a coupling coefficient $c^{(k)}_{ij}$ between capsule i in l-th layer and capsule j in class capsule $(l + 1)$-th layer for task k, which is the result of a softmax function on $\mathbf{b}^{(k)}_{ij}$.

$$c^{(k)}_{ij} = softmax(\mathbf{b}^{(k)}_{ij}) \tag{19}$$

Then, instantiated parameter $\mathbf{v}^{(k)}_j$ of capsule j in task k is calculated by

$$\mathbf{v}^{(k)}_j = \sum_i c^{(k)}_{ij} \cdot \hat{\mathbf{u}}^{(k)}_{j|i} \tag{20}$$

where $\hat{\mathbf{u}}^{(k)}_{j|i} = \mathbf{W}_{k,j,i}\mathbf{u}_i$

Coupling coefficient $c^{(k)}_{ij}$ is restricted in range $[0, 1]$, which represents the probability that capsule i belongs to class capsule j in task k. And it is update by the algorithm is described in Algorithm 3.5.

Multi-Task Loss

The loss for each task is the sum of margin losses for all the classes $\sum_{j=1}^{C} L^{(k)}_j$. By linearly combining the loss for every task, we get multi-task loss

$$L = \sum_{k=1}^{K} \beta^{(k)} \sum_{j=1}^{C} L^{(k)}_j. \tag{21}$$

where $\beta^{(k)}$ is the weight for each loss and $\sum_{k=1}^{K} \beta^{(k)} = 1$. In this paper, all the $\beta^{(k)}$ are set to be $1/K$ to make a balance among K tasks.

Multi-Task Training

In order to juggle several tasks in a unified network, following (Collobert and Weston, 2008), each task is trained alternatively in a stochastic manner. The steps can be described as follows: 1. Pick up a task k randomly; 2. Select an arbitrary sample s from the task k; 3. Feed the sample s into the McapsNet and update the parameters; 4. Go back to 1.

3.6 Architectures of CapsNets for Text

As illustrated in Figure 2, we propose a capsule-based multi-task learning architecture McapsNet, which is base on the single-task structures CapsNet-1 and CapsNet-2. Architectures for them are detailed as following.

CapsNet-1 As depicted in Figure 2, CapsNet-1 is a fundamental framework with three layers. The first layer is a plain convolution operation with 256 kernels with window size of 3 and stride of 1. For activation function, we use ReLU to augment non-linearity. This layer helps extract local features from the input sequences, which is the base to construct primary capsules.

Primary capsule layer employs 32 kernels with window size of 3 and stride of 1. The emitted primary capsules are 8-dimensional, which have bigger respective field, helping reassemble the piece features into wholes.

Last one is the class capsule layer, which is comprised of 16-dimensional capsules for the classes. They are connected to PrimaryCaps with routing-by-agreement and the coupling coefficients are updated by dynamic routing algorithm.

CapsNet-2 On this basis of CapsNet-1, CapsNet-2 upgrades the convolutional layer and uses multiple kernel sizes, which enriches the features. And concatenating[1] them up allows primary capsule see the features with different kernel sizes in the same time.

MCapsNet McapsNet is a unified multi-task structure based on CapsNet-2. It replaces the dynamic routing with task routing (Algorithm 3.5), which enables the network to route the features to

---

[1] We use padding to ensure the sizes of feature maps are equal.

Dataset	Train	Dev	Test	Classes	Type
MR	9500	-	1100	2	review
SST-1	8544	1101	2210	5	sentiment
SST-2	6920	872	1821	2	sentiment
Subj	9000	-	1000	2	subjectivity
TREC	5900	-	500	6	question
AG's	120k	-	7600	4	news

Table 1: Statistics for six datasets

Dataset	MR	SST-1	SST-2	Subj	TREC	AG's
LSTM	75.9	45.9	80.6	89.3	86.8	86.1
BiLSTM	79.3	46.2	83.2	90.5	89.6	88.2
LR-LSTM	81.5	48.2	87.5	89.9	-	-
VD-CNN	-	-	-	-	-	91.3
DCNN	-	48.5	86.8	-	**93.0**	-
CNN-MC	81.1	47.4	**88.1**	93.2	92.2	-
CapsNet-1	**81.5**	48.1	86.4	**93.3**	91.8	91.1
CapsNet-2	**82.4**	**48.7**	87.8	**93.6**	92.9	**92.3**
- Orphan	81.9	48.3	87.2	93.4	92.6	91.7

Table 2: Single-task results. Row "- Orphan category" denotes a variant of CapsNet-2 without orphan category

each tasks. And the whole network is optimized in a stochastic way with multi-task training (Section 3.5).

Implement Details For word embedding, we use the word vectors in *Word2Vec* (Mikolov et al., 2013), which is 300-dimensional and has 3M vocabularies. And all the routing logits $b_{ij}^{(k)}$ is initialized to zero, so that all the capsules in adjacent layers ($\hat{\mathbf{u}}_{j|i}, \mathbf{v}_j$) are connected with equal possibility c_{ij}. The coupling coefficients are updated by routing with 3 iterations, which performs best for our approach. For training, we use Adam optimizer (Kingma and Ba, 2014) with exponentially decaying learning rate. Moreover, we use mini-batch with size of 8 for all the datasets.

4 Experiment

We test our capsule-based models on six datasets in both single-task and multi-task scenarios to demonstrate the effectiveness of our approaches. We also in this section conduct some investigations like ablation study and visualization to give a comprehensive understanding to the characteristics of our models.

4.1 Datasets

For both single-task and multi-task scenarios, we conduct extensive experiments on six benchmarks: movie reviews (MR) (Bo and Lee, 2005), Stanford Sentiment Treebank (SST-1 and SST-2) (Socher et al., 2013), subjectivity classification (Subj) (Pang et al., 2004), question dataset (TREC) (Li and Roth, 2002), AG's news corpus (Mousa et al., 2017). These datasets cover a wide range of text classification tasks, which can fully test a model and the details are listed in Table 1.

4.2 Competitors

To demonstrate the effectiveness of our capsule network, we compare the single-task architectures with several state-of-the-art models, involving LSTM/BiLSTM (Cho et al., 2014), LSTM regularized by linguistic knowledge (LR-LSTM) (Qian et al., 2016), very deep network (VD-CNN) (Conneau et al., 2016), dynamic CNN (DCNN) (Kalchbrenner et al., 2014), CNN with multiple channels (CNN-MC) (Kim, 2014). Also, we compare the multi-task architecture (Figure 2) with several strong baselines of multi-task learning, including a general architecture for multi-task learning (MT-GRNN) (Zhang et al., 2017), recurrent neural network based multi-task learning (MT-RNN) (Liu et al., 2016), convolutional neural network with multi-task learning (MT-DNN) (Collobert and Weston, 2008), deep neural network with multi-task learning (MT-CNN) (Liu et al., 2015).

4.3 Single-Task Learning Results

We first test our approach on six datasets for text classification under the scheme of single-task. As Table 2 shows, our single-task network enhanced by capsules is already a strong model. CapsNet-1 that has one kernel size obtains the best accuracy on 2 out of 6 datasets, and gets competitive results on the others. And CapsNet-2 with multiple kernel sizes further improves the performance and get best accuracy on 4 datasets. This proves our capsule networks are effective for text. Particularly, our capsule network outperforms conventional CNNs like DCNN, CNN-MC and VD-CNN with a large margin (by average 1.1%, 0.7% and 1.0% respectively), which shows the advantages of capsule network over conventional CNNs for clustering features and leveraging the position information.

Routing Iteration The coupling coefficients c_{ij} are updated by dynamic routing algorithm, which determines the connections between the capsules. To find the best updating iteration for coupling co-

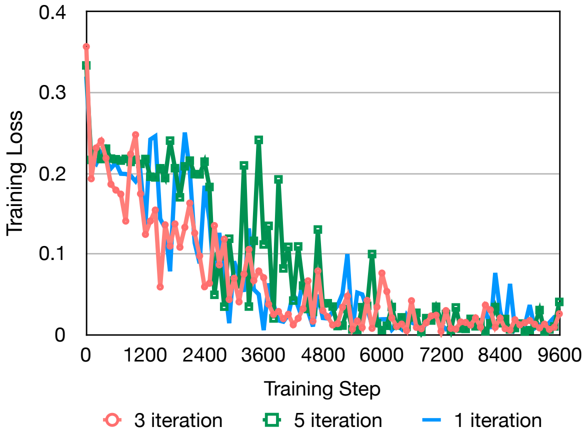

Figure 3: Influence of routing iteration

efficients, we test the CapsNet-2 with a series of iterations (1, 3 and 5) on MR dataset. As shown in Figure 3, network with 3 iterations convergences fast and performs best, which stays in line with the conclusion in (Sabour et al., 2017). So we utilize 3 iterations in all our experiments.

Ablation Study on Orphan Category Orphan category in class capsule layer helps collect the noise capsules that contain the 'background' information like stop words, punctuations or any unrelated words. We conduct the ablation experiment on orphan category, and result (Table 2) shows that network with orphan category perform better than the without one by 0.4%. This demonstrates the effectiveness of orphan category.

4.4 Multi-Task Learning Results

Up to now, we have obtained an optimized single-task architecture. In this section, we equip CapsNet-2 with the *task routing* and multi-task training procedure, namely the model MCapsNet, so that this capsule based architecture can learn several datasets in a unified network. Extensive experiments are conducted in this section to demonstrate the effectiveness of our multi-task learning architecture, as well as its ability for feature clustering.

Multi-Task Performance

We simultaneously train our model McapsNet on six tasks in Table 1 and compare it with single-task scenario (Table 3). We can see that our multi-task architecture clearly improves the performance over the single task models, which demonstrates the benefits of our multi-task architecture.

Dataset	MR	SST-1	SST-2	Subj	TREC	AG's	Avg.△
BiLSTM	79.3	46.2	83.2	90.5	89.6	88.2	+0
MT-GRNN	-	49.2	87.7	89.3	93.8	-	+2.6
MT-RNN	-	49.6	87.9	94.1	91.8	-	+3.5
MT-DNN	82.1	48.1	87.3	93.9	92.2	91.8	+2.9
MT-CNN	81.6	49.0	86.9	93.6	91.8	91.9	+3.0
CapsNet-1	81.5	48.1	86.4	93.3	91.8	91.1	+2.5
CapsNet-2	82.4	48.7	87.8	93.6	92.9	92.3	+3.3
MCapsNet	83.5	49.7	88.6	94.5	94.2	93.8	+4.6

Table 3: Multi-task results of MCapsNet. In column Avg.△, we use BiLSTM as baseline and calculate the average improvements over it.

As Table 3 shows, MCapsNet also outperforms the state-of-the-art multi-task learning models by at least 1.1%. This shows the advantages of our task routing algorithm, which can cluster the features for each task, instead of freely sharing the features among tasks.

4.5 Routing Visualization

To show the mechanism how capsule benefits the multi-task learning, we visualize the coupling coefficient $c_{ij}^{(k)} \in [0, 1]$ between primary and class capsules. We use kernel with size 1 for primary capsule layer so that every capsule represents only one 3-gram phrase. The strength of these connections indicates the importance of these 3-grams to their corresponding task and class.

We feed a random sample from the dataset MR into MCapsNet. In the first row of Table 4, we show the most important 3-gram phrases for two tasks MR and Subj (two classes for each) with word cloud. The sizes of the grams represent the weights of coupling coefficients. We can see that task routing algorithm helps lead the grams into the most related tasks, which allows each task only consider the helpful features for them. In another word, task routing builds a feature space for each task and avoids they contaminate each other. This demonstrates that MCapsNet has the ability of feature clustering, which can benefit MTL by reducing the interference.

We also illustrate the coupling coefficients sequentially for each task. The height of the blue and gray lines represents the polarity of positivity and subjectivity respectively. It is clear that MCapsNet can focus on the appropriate positions for each task, which helps make the final correct predication for every task.

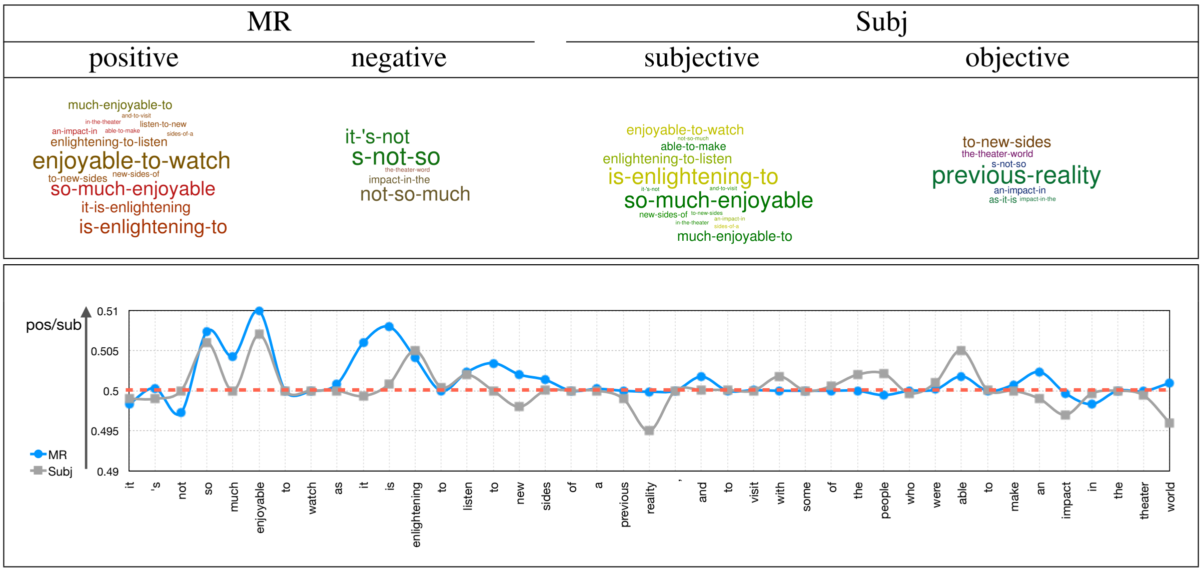

Table 4: Visualization of the task routing for a positive sample from MR, "it 's not so much enjoyable to watch as it is enlightening to listen to new sides of a previous reality , and to visit with some of the people who were able to make an impact in the theater world"

5 Related Work

Related work can be divided into two threads. The first thread is capsule network, which has been proven effective on many classification tasks.

Concept of capsule is first proposed by Hinton et al. (2011), which first use vector to describe the pose of object. This work improves the representation ability of the neural networks against the vanilla CNNs and also enhances the robust of network for transformation. Then dynamic routing algorithm is proposed in (Sabour et al., 2017), which is aimed to displace the pooling operation, building a part-whole relationship for object recognition. Dynamic routing can maintain the position information of features for objects that pooling operations generally discard. And the result shows the proposed method improves the state-of-the-art performance for MNIST dataset. Next, Hinton et al. (2018) employs the matrix to depict the pose and, based on EM algorithm designs a new routing procedure between capsule layers. This work shows strong ability for addressing transformation problem and gains significant improvement on smallNORB dataset.

All these methods are proposed for computer vision, while in this paper we investigate the benefits of capsules for text.

The other thread is about multi-task learning. The earliest idea can be traced back to (Caruana, 1997) and there have been some work completed in this field to augment the performance. Collobert and Weston (2008) develop a multi-task learning model based on CNN. It shares only one lookup table to train a better word embedding. And Liu et al. (2015) propose a DNN-based model for multi-task learning, which shares some low layers but separate the high-level layers to complete several different tasks.

Some models are proposed to share deeper layers of networks, which can exchange high-level knowledge among tasks and gain better performance. (Zhang et al., 2017) and (Liu et al., 2016) introduce some RNN architectures and design different schemes for knowledge sharing. These trials promote the performance of models, but they give no consideration to the interference in multi-task learning. Liu et al. (2017) add the adversarial losses in multi-task RNNs, which can alleviate the interference among tasks by finding a common feature space for tasks. However, the model has multiple subnets and various losses, which requires more computation and training skills.

Different from these methods, we use the thought of capsule in natural language processing (NLP) field. And proposed a capsule based multi-task learning architecture with task routing algorithm. This approach can cluster the features for each task, reducing the interference among them.

6 Conclusion and Future Work

This paper investigates the performance of capsule network for text representation, and proposes several effective architectures. By means of the characteristics of capsule network, we design a unified, sample yet effective architecture with task routing for multi-task learning, which has the ability to clustering the features, building a private feature space for every task.

In future work, we would like to investigate the relations of various tasks in multi-task learning by exploiting the potential of capsule network.

7 Acknowledge

We appreciate the valuable comments from anonymous reviewers. We also thank Xuan Luo for building and maintaining the GPU platforms. And this research was funded by Major State Research Development Program under Grant No. 2018YFC0830400.

References

Pang Bo and Lillian Lee. 2005. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. pages 115–124.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 160–167.

Alexis Conneau, Holger Schwenk, Loc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. pages 1107–1116.

Yuchun Fang, Zhengyan Ma, Zhaoxiang Zhang, Xu-Yao Zhang, and Xiang Bai. 2017. Dynamic multi-task learning with convolutional neural network. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1668–1674.

Geoffrey Hinton, Nicholas Frosst, and Sara Sabour. 2018. Matrix capsules with em routing.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2011. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

Xin Li and Dan Roth. 2002. Learning question classifiers. *Coling*, 12(24):556–562.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1–10.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119.

Amr Mousa, Bjrn Schuller, Amr Mousa, Bjrn Schuller, Amr Mousa, and Bjrn Schuller. 2017. Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. In *Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1023–1032.

Pang, Bo, Lee, and Lillian. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. *Proceedings of Acl*, pages 271–278.

Qiao Qian, Minlie Huang, and Xiaoyan Zhu. 2016. Linguistically regularized lstms for sentiment classification. *CoRR*, abs/1611.03949.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Sgaard. 2017. Sluice networks: Learning what to share between loosely related tasks.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank.

Honglun Zhang, Liqiang Xiao, Yongkun Wang, and Yaohui Jin. 2017. A generalized recurrent neural architecture for text classification with multi-task learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 3385–3391.

Uncertainty-aware generative models for inferring document class prevalence

Katherine A. Keith and **Brendan O'Connor**
College of Information and Computer Sciences
University of Massachusetts Amherst
`{kkeith,brenocon}@cs.umass.edu`

Abstract

Prevalence estimation is the task of inferring the relative frequency of classes of unlabeled examples in a group—for example, the proportion of a document collection with positive sentiment. Previous work has focused on aggregating and adjusting discriminative individual classifiers to obtain prevalence point estimates. But imperfect classifier accuracy ought to be reflected in uncertainty over the predicted prevalence for scientifically valid inference. In this work, we present (1) a generative probabilistic modeling approach to prevalence estimation, and (2) the construction and evaluation of prevalence confidence intervals; in particular, we demonstrate that an off-the-shelf discriminative classifier can be given a *generative re-interpretation*, by backing out an implicit individual-level likelihood function, which can be used to conduct fast and simple group-level Bayesian inference. Empirically, we demonstrate our approach provides better confidence interval coverage than an alternative, and is dramatically more robust to shifts in the class prior between training and testing.[1]

1 Introduction

The goal of *prevalence estimation* is to infer the relative frequency of classes y_i associated with unlabeled examples (e.g. documents) from a group, $x_i \in \mathcal{D}$. For example, one might want to estimate the proportion of blogs with a positive sentiment towards a political candidate (Hopkins and King, 2010), sentiment of responses to natural disasters on social media (Mandel et al., 2012), or prevalence of car types in street photos to infer neighborhood demographics (Gebru et al., 2017). Often, an analyst wants to compare prevalence between multiple groups, such as inferring prevalence variation over time (e.g., changes to online abuse content (Bissias et al., 2016)), or across other covariates (e.g., changes in police officers' "respect" when speaking to minorities (Voigt et al., 2017)). This problem has been re-introduced in many different fields: as "quantification" in data mining (Forman, 2005, 2008), "prevalence estimation" in statistics and epidemiology (Gart and Buck, 1966), and "class prior estimation" in machine learning (Vucetic and Obradovic, 2001; Saerens et al., 2002). In NLP, SemEval 2016 and 2017 included Twitter sentiment class prevalence tasks (Nakov et al., 2016; Rosenthal et al., 2017).

Prevalence estimation assumes access to a (potentially small) set of labeled examples to train a classifier; but unlike the task of individual classification, the goal is to estimate the proportion of a class among examples in a group. If a perfectly accurate classifier is available, it is trivial to construct a perfect prevalence estimate by counting the classification decisions (§3.1). In fact, most application papers in the previous paragraph use this or a similar aggregation rule to conduct their prevalence estimates. However, classifiers often exhibit errors from different sources, including:

- Shifts in the class distribution from training to testing ($P_{train}(y) \neq P_{test}(y)$). A classifier may be biased toward predicting $P_{train}(y)$.
- Difficult classification tasks (such as predicting sentiment or sarcasm) that result in low accuracy classifiers; this can be exacerbated by limited training data, as is common in social science or industry settings that require manual human annotation for labels.

It is typically assumed (and sometimes confirmed) that when an individual classifier has less than 100% accuracy, it can still give reasonable preva-

---

[1] Code available at `http://slanglab.cs.umass.edu/doc_prevalence` and `https://github.com/slanglab/doc_prevalence`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4575–4585
Brussels, Belgium, October 31 - November 4, 2018.

lence estimates.[2] However, there is relatively little understanding to what extent the quality of the document-level model impacts prevalence estimates. Imperfect classifier accuracy ought to be reflected in uncertainty over the predicted prevalence.

In this work, we tackle both of these challenges simultaneously, using a generative probabilistic modeling approach to prevalence estimation. This model directly parameterizes and conducts inference for the unknown prevalence, naturally accommodating shifts between training and testing, and also allows us to infer confidence intervals for the prevalence. We show that our best model can be seen as an *implicit likelihood* generative re-interpretation of an off-the-shelf discriminative classifier (§4.2); this unifies it with previous work, and also is easy for a practitioner to apply.

We additionally review several types of class prevalence estimators from the literature (§3), and conduct a robust empirical evaluation on sentiment analysis over hundreds of document groups, illustrating the methods' biases and robustness to class prior shift between training and testing. Our method provides better confidence interval coverage and is more robust to class prior shift than previous methods, and is substantially more accurate than an algorithm in widespread use in political science.

2 Problem definition

We consider two prevalence estimation problems: (1) point prediction and (2) confidence interval prediction. In this work, we are most interested in supervised learning for discrete-valued document labels, with access to a small to moderate number (e.g. around 1000) of labeled documents with text x and label y: $(x_i, y_i) \in \mathcal{D}^{train}$. We restrict attention to binary-valued labels $y \in \{0, 1\}$. At test time, there are one or more groups of unlabeled test documents, $\mathcal{D}^{(1)}, \cdots, \mathcal{D}^{(G)}$; for example, one group might be a set of tweets sent during a certain month, or a set of online reviews associated with a particular product. For each group $\mathcal{D}$, let $\theta^* \equiv (1/n) \sum_i^n y_i$ be the true proportion of positive labels (where $n = |\mathcal{D}|$).

The *prevalence point prediction* problem is to take an unlabeled document group $\mathcal{D}$ as input and infer an estimated $\hat{\theta} \in [0, 1]$. Ideally, this point estimate should be close to the true prevalence θ^*; we evaluate this by mean absolute error.

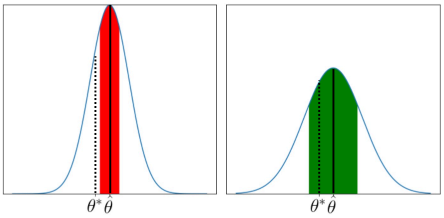

Figure 1: Example posterior distributions with MAP prevalence estimates, $\hat{\theta}$ (solid line) and the true prevalence, θ^* (dashed line). A desirable property is that confidence intervals, technically Bayesian credible intervals, (shaded regions) will be wider for more uncertain models. For example, the wider CI on the right (green) contains θ^* whereas the narrower CI interval on the left (red) does not.

In this work, we are the first (that we know of) to introduce the question of *uncertainty* in prevalence estimation. Since document classifiers are typically far from perfectly accurate, we should expect substantial error in prevalence prediction, and inference methods should quantify such uncertainty. We formalize this as a *prevalence confidence interval* (CI) inference, which takes as input a desired nominal coverage level $(1 - \alpha)$, and predicts a real-valued interval $[\hat{\theta}_{lo}, \hat{\theta}_{hi}] \subseteq [0, 1]$. Ideally, a CI prediction algorithm should have frequentist coverage semantics: over a large number of test groups,[3] $(1-\alpha)\%$ of the predicted intervals ought to contain the true value θ^*. If the problem is hard—for example, the relationship between document features and the label is not captured well by the model—the CI should be wide. We empirically evaluate coverage of CI-aware prevalence inference models. See Fig. 1 for an intuitive example.

3 Review and baselines: Discriminative individual classification aggregation

The most straightforward baseline approach to prevalence estimation is to build on discriminative, supervised learning for individual-level labels, such as binary logistic regression with bag-of-words features, randomized feature hashing

---

[2] For example, Bissias et al. find a relative mean absolute error of less than 0.01 when the individual classifier has ROC AUC of 0.91.

[3] Or in fact, across many experiments in which the model or algorithm is applied (Wasserman, 2011).

(Weinberger et al., 2009), or neural networks (Goldberg, 2016). Such a model defines an individual document's label probability $p_i \equiv p_\beta(y_i = 1 \mid x_i)$ where parameters β are fit by maximizing regularized likelihood on the labeled training data.

3.1 Classify and Count (CC)

For prevalence point estimation, Forman (2005) defines the "classify and count" (CC) method as simply averaging the most-likely individual label predictions,

$$\hat{\theta}^{CC} = \frac{1}{n} \sum_i 1\{p_i > 0.5\}. \tag{1}$$

This is the most obvious approach for practitioners, but it has at least two weaknesses, which have been addressed in different groups of prior work. First, the class proportions may change between training and test groups, which the Adjusted CC and ReadMe algorithms attempt to fix (§3.2–3.3). Second, it discards probabilistic information, which is remedied by the Probabilistic CC method, and an extension we propose (§3.4–3.5).

3.2 Adjusted Classify and Count (ACC)

CC may encounter problems if the test class distribution is different than the training's. The "adjusted classify-and-count" method (Gart and Buck, 1966; Forman, 2005) treats the classifier output as a proxy variable, and estimates a separate confusion model of classifier output $\hat{y}_i \equiv 1\{p_i > 0.5\}$ conditional on the true label, $p(\hat{y} \mid y)$, from cross-validation within the training set. Assuming the confusion model extends to the test data, a moment-matching approach is then used to infer the true label proportions, by first observing $p_{test}(\hat{y}) = \sum_y p(\hat{y} \mid y) p_{test}(y)$ and solving the linear system for $p_{test}(y)$, the test-time expected class prevalence. Using empirical estimates for the true positive rate TPR $= p(\hat{y} = 1 \mid y = 1)$, and false positive rate FPR $= p(\hat{y} = 1 \mid y = 0)$, and $\hat{\theta}^{CC} = p(\hat{y} = 1)$, it has the closed form

$$\hat{\theta}^{ACC} = \frac{\hat{\theta}^{CC} - \text{FPR}}{\text{TPR} - \text{FPR}}. \tag{2}$$

By design, ACC is more robust to a new test-time prevalence, but it relies on the accuracy of its TPR and FPR estimates, and its lack of probabilistic semantics makes it unclear how to infer confidence intervals.

3.3 ReadMe algorithm

An interesting extension to ACC is to remove the need for a discriminative classifier, by directly modeling text conditional on the latent document class. The ReadMe algorithm, developed in political science (Hopkins and King, 2010), extends ACC's linear system for every term type in a (subsampled and augmented) term vocabulary $\mathcal{V}$, and calculates their class-conditional probabilities from the training data. Assuming these conditional models also hold in the test data, that implies $p_{test}(w) = \sum_y \hat{p}(w \mid y) p_{test}(y)$; the algorithm infers $p_{test}(y)$ by minimizing the squared error of predicted versus empirical term frequencies in the test set. The open-source ReadMe software package[4] has been used in numerous political science studies, including inferring proportions of types of censored Chinese news (King et al., 2013), credit claiming in Congressional press releases (Grimmer et al., 2012), and voter intentions among Twitter messages (Ceron et al., 2015).

ReadMe is theoretically appealing in that it infers latent class prevalences to explain the test group's textual evidence; but as a non-probabilistic model, it does not directly imply a method for confidence intervals (Hopkins and King use the bootstrap). Furthermore, our experiments (§5), contra the original paper, show its implementation exhibits poor performance.

3.4 Probabilistic Classify and Count (PCC)

Both the CC and ACC methods discard uncertainty information from the classification model. In a difficult classification setting, for example, we might expect many probabilities to be near, say, 0.6, in which case the CC method may undercount the negative class. This suggests an alternative method, "probabilistic classify and count" (PCC):

$$\hat{\theta}^{PCC} = \frac{1}{n} \sum_i p_i \tag{3}$$

which is the expected prevalence, $(1/n) \sum_i y_i$, assuming each y_i is distributed according to the original probabilistic classifier.

3.5 PCC Poisson-Binomial distribution (PB-PCC)

If we assume each y_i is conditionally independent given text x_i and model parameters β, this

---

[4] `https://gking.harvard.edu/readme`

defines a fully probabilistic model for the class prevalence. Let the latent variable $S = \sum_i y_i$; its distribution is thus Poisson-Binomial (Chen and Liu, 1997). The modeled prevalence distribution $p(\frac{S}{n} \mid \mathcal{D})$ can be exactly inferred by Monte Carlo inference: each iteration samples every y_i and sums for an S sample. The S/n distribution over many iterations can be used to construct a Monte Carlo CDF $\hat{F}$, from which any $[\hat{F}(t), \hat{F}(t+1-\alpha)]$ is an $(1-\alpha)$-sized credible interval (where $0 \leq t \leq t+1-\alpha \leq 1$). This model has prevalence expectation $E[\frac{S}{n}] = \hat{\theta}^{PCC}$, and variance

$$\text{Var}\left[\frac{S}{n}\right] = \frac{1}{n^2}\sum_i p_i(1-p_i). \tag{4}$$

To a certain degree, this model captures uncertainty in the classifier since per-document variance, $p_i(1-p_i)$, is high when $p_i = 0.5$ and low when near 0 or 1. However, it also has a major weakness—the variance concentrates with a large test group size n, which is the wrong behavior when a classifier is truly noisy, for example, when a classifier is genuinely uncertain and predicts the same constant $p_i = q$ for each document. In this case, the correct behavior would be to maintain a flat, wide posterior belief about θ, which is better accomplished by the generative model we introduce in the subsequent section.

4 Our approach: generative probabilistic modeling

We turn to generative modeling, that seeks to to jointly model the probability of labels and text in both the training and test groups, by assuming a document's text is generated conditional on the document label. Language models have widespread use in natural language processing, and class-conditional models have been used for document classification (e.g. multinomial Naive Bayes; McCallum and Nigam (1998)). We use a similar generative setup to explicitly model a class prevalence for test group g, with a generative story for each (bag-of-words) document i in the group:

$$\theta_g \sim \text{Dist}(\alpha) \tag{5}$$

$$y_{i,g} \sim \text{Bernoulli}(\theta_g) \tag{6}$$

$$x_{i,g} \sim \text{Multinomial}(\phi_{y_{i,g}}) \tag{7}$$

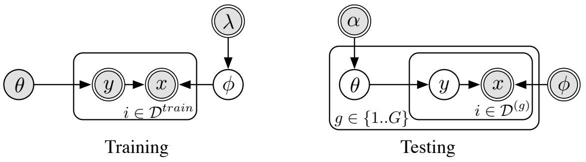

Training Testing

Figure 2: Our generative model for prevalence estimation. **Left:** Class-conditional language models (ϕ) are learned at training time. **Right:** Test-time inference for multiple groups' latent prevalences (θ).

The test group is assumed to have a latent class prior θ_g, which itself has a prior distribution (we assume $\text{Dist}(\alpha) = \text{Unif}(0,1)$ in this work). For each class k, ϕ_k is a class-conditional unigram language model, which is learned from the training data but fixed at test time. We then perform inference to find θ_g that gives a high probability to text data $\{x_i \in \mathcal{D}^{(g)}\}$. Figure 2 shows the probabilistic graphical model.

4.1 MNB and Loglin language models

We experiment with two explicit language models in this generative framework: (1) multinomial Naive Bayes (**MNB**), using a training-time symmetric Dirichlet prior $\phi_y \sim \text{Dir}(\lambda/V)$ for vocabulary size V and "pseudocount" λ, and (2) an additive log linear model (**Loglin**, a.k.a. SAGE (Eisenstein et al., 2011)). Loglin estimates words' probabilities as deviations from a background log-probability m,

$$\eta_{y,w} \sim \text{Laplace}(\lambda) \tag{8}$$

$$\phi_{y,w} = \exp(m_w + \eta_{y,w}) / \sum_j \exp(m_j + \eta_{y,j})$$

where m_w is the empirical log probability of a word w among all training documents, and $\eta_{y,w}$ denotes class-specific deviations of the log-probability of a word w, MAP estimated under a sparsity-inducing L1 penalty. Such sparse additive models have been used in both supervised and unsupervised document modeling; for example, as a document-level posterior classifier it outperforms MNB (Eisenstein et al., 2011), or even discriminative models (Taddy, 2013), and its sparsity helps interpretability for analyzing political, literary, and legal texts (Monroe et al., 2008; Sim et al., 2013; Bamman et al., 2014; Wang et al., 2012).

4.2 Implicit likelihoods from discriminative classifiers (LR-Implicit)

This generative formulation has a major advantage over the discriminative, CC-style aggregation models because it sets up a likelihood and posterior distribution over θ. But in terms of document modeling for classification purposes, the independence assumptions of the generative model are typically too strong, and for document-level classification, discriminative models tend to outperform similarly parameterized generative ones, especially when the training set is sufficiently large (Ng and Jordan, 2002). Thus, discriminative models may have information better suited to class prevalence inference. Also, since the most common practice for document classification is to use discriminative models, it would be helpful to more effectively use discriminative posteriors within our generative context.

In Naive Bayes-style generative document classification, the model defines $p_{gen}(x \mid y)$ and class prior $p(y)$, which are combined to calculate the posterior $p_{gen}(y \mid x) \propto p_{gen}(x \mid y)p(y)$. Discriminative models, by contrast, directly define a $p_{disc}(y \mid x)$. We can, however, expand this quantity via Bayes Rule:

$$p_{\text{disc}}(y \mid x) = p_{\text{implicit}'}(x \mid y) p_{\text{train}}(y)/p(x). \quad (9)$$

The "implicit document likelihood" $p_{\text{implicit}'}(x \mid y)$ is a likelihood function that, combined with a particular class prior $p(y)$, would have resulted in the same posterior predicted by the discriminative model. Given the discriminative posterior predictions and the training-time class prior $p_{\text{train}}(y) = \hat{\theta}_{\text{train}}$, an implicit likelihood function can be backed out for any particular document x; we define the "simple implicit" likelihood for document x to be:

$$p_{\text{implicit}}(x \mid y) = p_{\text{disc}}(y \mid x)/\hat{\theta}_{\text{train}}. \quad (10)$$

This takes the form of a correction of the discriminative posterior, by dividing out the training-time class prevalence.[5]

Our **LR-Implicit** generative model uses the same class prevalence and document label generation setup as before, but to calculate the individual documents' $p(x \mid y)$ probabilities, it uses p_{implicit} based on a logistic regression p_{disc}.[6]

This model is inspired by Saerens et al. (2002)'s EM algorithm for adjusting a classifier for a test set's class prior; they derive it differently by applying the assumption $p_{\text{train}}(x \mid y) = p_{\text{test}}(x \mid y)$, expanding each side with Bayes' Rule, solving for $p_{\text{test}}(y \mid x)$, then estimating $p_{\text{test}}(y)$ via EM. This in fact optimizes the same marginal likelihood function in the next section under the implicit-discriminative generative model; our formulation broadens it as a fully Bayesian or likelihood-based model.

4.3 Inference

To estimate class prevalence, we use the marginal log likelihood over θ to obtain a posterior over θ. For each each test group g, we have the marginal log probability of all document texts,

$$\begin{aligned} \text{MLL}_g(\theta) &\equiv \log p(\mathcal{D}^{(g)} \mid \theta) \quad (11) \\ &= \sum_{i \in \mathcal{D}^{(g)}} \log \sum_{y \in \{0,1\}} p(x_i, y_i = y \mid \theta) \\ &= \sum_{i \in \mathcal{D}^{(g)}} \log \left(\theta L_i^+ + (1-\theta) L_i^- \right), \end{aligned}$$

where we denote the class-conditional document text likelihoods $L_i^+ \equiv p(x_i \mid y_i = 1)$ and $L_i^- \equiv p(x_i \mid y_i = 0)$. The gradient for an individual document is $(L_i^+ - L_i^-)/(\theta L_i^+ + (1-\theta)L_i^-)$; intuitively, the sign of the numerator says that documents that are more likely under the positive than negative class encourage higher likelihood for larger values of θ. When the model is uncertain about a document—that is, when $L_i^+ \approx L_i^-$—that document contributes a relatively flat likelihood curve, expressing little preference for likely values of θ. If a model is more heavily regularized—for example, when the log-linear additive model is more dominated by the background language model—this condition tends to hold for the documents, leading to a flat, highly uncertain likelihood curve.

The marginal log likelihood is unimodal over $\theta \in [0,1]$, since it is concave, being a sum of concave log-linear functions, and having negative curvature:

$$\frac{\partial^2 \text{MLL}_g}{\partial \theta^2} = - \sum_{i \in \mathcal{D}^{(g)}} \left(\frac{L_i^+ - L_i^-}{\theta L_i^+ + (1-\theta)L_i^-} \right)^2. \quad (12)$$

Since it is concave and there is only one parameter, a very wide variety of techniques could be

---

[5] Technically, $p_{\text{implicit}'}$ is retrievable only up to a constant, and p_{implicit} is one particular compatible implicit likelihood, since it can be multiplied by any constant and is still consistent with Eq. 9, and would give rise to the same document- and group-level posteriors.

[6] The implicit likelihood still has the form of a logistic regression, adjusting its bias term: if $p_{\text{disc}}(y \mid x) = \sigma(\beta' x + \beta_0)$, then $p_{\text{implicit}}(x \mid y) = \sigma(\beta' x + \beta_0 - \log(\theta_{\text{train}}/(1-\theta_{\text{train}})))$.

used to reliably find a mode, including EM or first- or second-order methods. At least two approaches to inferring confidence intervals are possible. One is to use a central limit theorem-style approximation, assuming the sampling distribution is approximated by a normal with mean θ^{MLE} and variance $-[\partial^2 \text{MLL}_g / \partial\theta^2]^{-1}$. The second, which we focus on, is Bayesian estimation for $\log p(\theta_g \mid \mathcal{D}^{(g)}) \propto \log p(\theta_g) + \text{MLL}_g(\theta_g)$ by simply using a grid search over values $\theta \in \{0.001, 0.002, ...0.999\}$ to infer both the posterior mode θ^{MAP} as well as a 90% highest posterior density interval.[7] In small-scale experiments, this model had very similar results to the central limit theorem (with EM for θ^{MLE}).

5 Experiments

5.1 Data

In order to compare document class prevalence estimators, we desire datasets that (1) have natural document groups that correspond to realistic, real-world applications, (2) have a large number of test groups (hundreds or more), and (3) are freely available for academic research. It has been a challenge to fulfill these criteria in previous work. Nakov et al. (2016) conduct large-scale manual annotation of Twitter sentiment for SemEval 2016 Task 4, with topic-based test groups; unfortunately, redistribution is restricted to message IDs, making the original dataset difficult to reconstruct under Twitter's terms of service if messages have since been deleted. Bella et al. (2010) and Esuli and Sebastiani (2015) use large, pre-existing labeled document corpora, but they do not contain natural groups; evaluations utilize randomly sampled synthetic groups.

To better fulfill these criteria, we select the task of business review sentiment prevalence, where the goal is to estimate the proportion of reviews that are positive for one particular business; specifically, we use labeled data from the Yelp Dataset Challenge Round Nine[8] corpus, which consists of 4.1M reviews by 1M users for 144K businesses. We sample 500 businesses with at least 200 reviews each as the test groups. We treat the task as binary classification, and assign $y_i = 1$ to reviews with 3 or more stars. This task seems reasonably representative of real-world sentiment analysis problems, and this type of dataset can easily be collected and reproduced from Yelp or other widely available review data.

For training, we simulate a small-scale annotation project by sampling 2000 labeled documents from the rest of the corpus. This is a **natural** prevalence that on average is about the same as the test groups, though individual test groups may have a much different prevalence (ranging from 0.096 to 0.997, mean (stdev) 0.823 (0.136)). We also construct a **synthetic** training setting with a highly skewed class prior, selecting 2000 documents with a 0.1 class prevalence (i.e. 200 positive documents in the group). In each case, for every model, we re-run and average results over 10 different samples of the training set. For preprocessing, we tokenize with NLTK[9] and lowercase.

5.2 Model training

We use L1 regularization for logistic regression based on the vector of a documents' word counts, to be most directly comparable to the generative models; for each model, we select its hyperparameter (LR and Loglin's λ, or MNB's pseudocount) by minimizing cross-validated cross-entropy of individual document posteriors (within the labeled training set), over a grid search of powers of 2. The log-linear additive model is trained with OWL-QN (Andrew and Gao, 2007)[10] and the logistic regression model is trained with the default implementation in scikit-learn (Pedregosa et al., 2011).[11] We used ReadMe with its default parameters.[12]

5.3 Results

For each of the 500 test groups, we calculate a prevalence point estimate $\hat{\theta}$ with each method, and evaluate by averaging across groups for mean absolute error $\sum_g |\hat{\theta}_g - \theta^*_g|$ and bias $\sum_g (\hat{\theta}_g - \theta^*_g)$.[13] For the models that allow for confidence interval

---

[7] Since we use a uniform prior, this is just the MLE. Technically, we used a prior of Beta(1.0001, 1.0001) to avoid certain issues with tie-breaking, but it was not necessary.

[8] Downloaded June 2017 from `https://www.yelp.com/dataset_challenge`.

[9] `http://www.nltk.org/`

[10] Via `github.com/larsmans/pylbfgs`

[11] Version 0.18.2

[12] Version 0.99837 from `https://gking.harvard.edu/readme`, with default parameters features=15, n.subset=300, prob.wt=1. We bypass the ReadMe software's text preprocessing pipeline, and instead have it use nearly the same document-term matrices as the other models. Since it only handles binary document-term matrices, we transformed counts to indicators; with other models this change only made a minor difference in results.

[13] For the generative (MLL) models, $\hat{\theta}$ is the MAP estimate; the posterior mean gives similar results.

		Natural training prevalence ≈ 0.8				Synthetic training prevalence = 0.1			
		Point est.		CIs		Point est.		CIs	
		MAE	Bias	Cover.	Width	MAE	Bias	Cover.	Width
Const.	Pred. train mean	0.114	-0.045	—	—	0.723	-0.723	—	—
	Pred. 100%	0.177	0.177	—	—	0.177	0.177	—	—
	ReadMe	0.233	-0.222	—	—	0.383	-0.382	—	—
Disc. (LR)	CC	0.048	0.042	—	—	0.503	-0.503	—	—
	ACC	0.048	-0.001	—	—	0.132	-0.015	—	—
	PB-PCC	0.049	-0.017	0.283	0.044	0.464	-0.464	0.001	0.054
Gen. (MLL)	MNB	0.078	0.058	0.120	0.046	0.199	-0.199	0.022	0.073
	Loglin	0.089	-0.070	0.410	0.100	0.140	-0.036	0.510	0.273
	LR-Implicit	0.050	0.001	0.454	0.074	0.069	-0.051	0.439	0.082

Table 1: Mean absolute error (MAE), bias, nominal 90% confidence interval coverage, and average CI width for the 500 Yelp data test groups, averaged over 10 simulations of resampled training (2000 document) sets. We examine both the natural positive class training prevalence ($E[\theta_{train}] = 0.7783$), and a synthetic fixed prevalence of 0.1. Dashes indicate the methods that are not able to calculate confidence intervals.

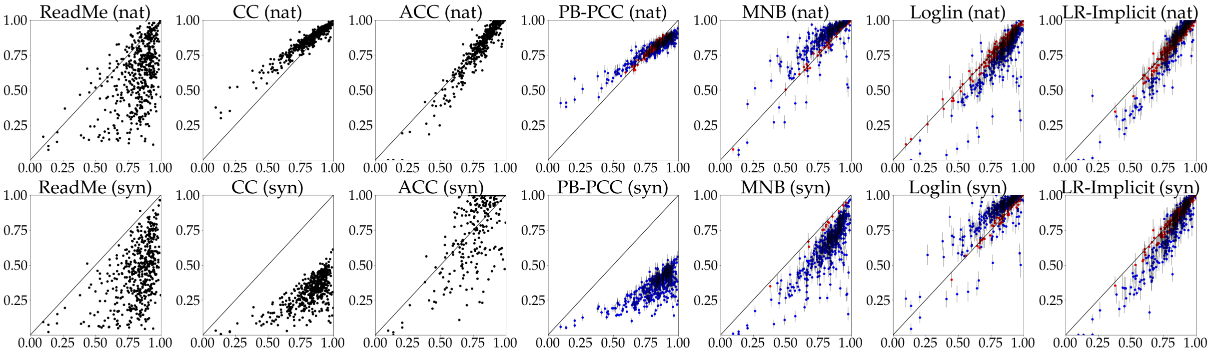

Figure 3: Gold prevalence θ^* (x-axis) versus predicted prevalence $\hat{\theta}$ (y-axis) for each of the 500 test groups with **natural** (nat) training prevalence (top row) and **synthetic** (syn) 0.1 training prevalence (bottom row). A black $y = x$ line is plotted for visualization. For the models that allow for confidence intervals, 90% CIs for each group are given by the faint grey lines. Blue dots indicate the CI does not contain θ^* and red dots indicate the CI does contain θ^*. For each setting, we show the the model with median MAE across training resamplings.

prediction, we infer 90% intervals and calculate coverage, which is best if it is 0.90. We also report average CI width; a narrower interval indicates more confidence (even if misplaced). Results are in Table 1; every result is averaged over 10 resamplings of the training set.

The ReadMe software did not have competitive performance; we hope in follow-up work to understand why Hopkins and King found it had considerably stronger performance than SVM-based CC.

For the natural training class prevalence setting (first column, Table 1), the discriminative-based models (CC, PCC and the adjusted variants ACC and LR-Implicit) all have very similar point estimate performance, outperforming the purely generative models (MNB and Loglin). For CI coverage, the log-linear and LR-Implicit generative models have significantly better coverage than the discriminative model (PB-PCC) or MNB. Future work is required to improve coverage to be closer to the nominal ideal of 90%.

By contrast, when the class prevalences are mismatched (second column, Table 1), the non-adjusted CC and PCC methods give extremely poor and biased point estimates, and PB-PCC has incredibly poor CI coverage. ACC and the generative models do much better, presumably because their models directly allow for variability in the test class prior. While Loglin has somewhat higher coverage in this setting, overall, LR-Implicit has

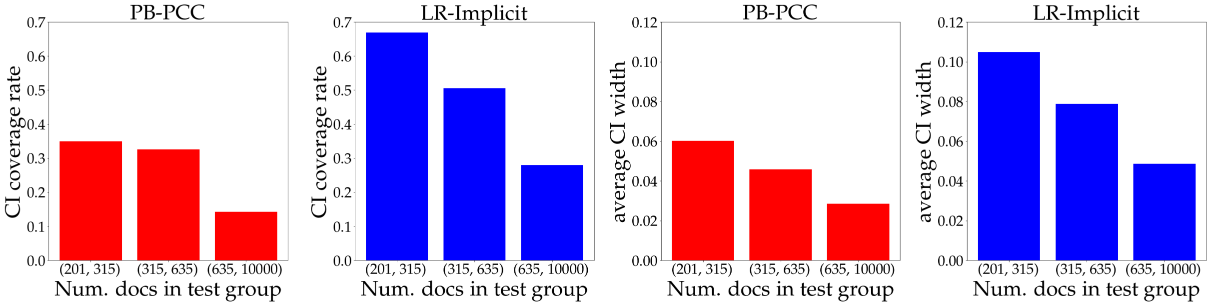

Figure 4: CI coverage rate (left two graphs) and average CI width (right two graphs) for three bins of the test groups, binned by number of documents.

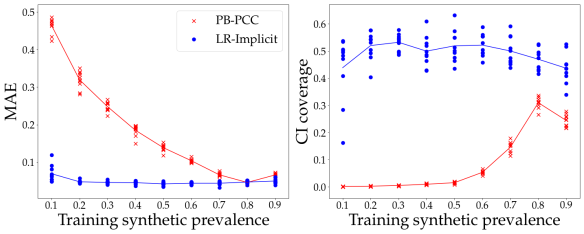

(a) Varying training prevalence

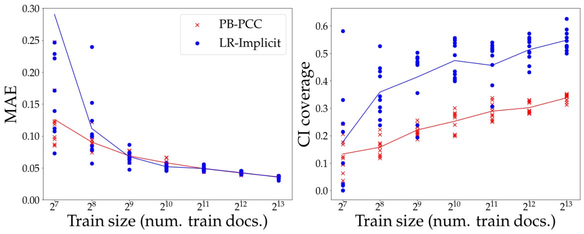

(b) Varying training size

Figure 5: MAE and 90% CI coverage for PB-PCC while varying **(a)** training prevalence (the proportion of the 2000 training documents with positive reviews) and **(b)** training size (number of documents in the training data) with natural prevalence. Lines are the averages over 10 resamplings of training sets and points represent one resampling.

consistently strong performance in both training settings, and for both point estimation and (relatively, at leas) confidence intervals.

Figure 3 shows θ^* versus $\hat{\theta}$ for each of the 500 test groups for each of the models, including predicted CIs. CC's and PCC's erroneous assumptions are directly viewable: in the natural prevalence setting, the slope shallower than 1, indicating a persistent under-sensitivity to the true class prevalence—unlike ACC and the generative models. In the synthetic training case, CC and PCC wildly underpredict, presumably because they are biased by the low training-time prevalence $\theta_{\text{train}} = 0.1$.

5.4 Comparison of PB-PCC and LR-Implicit

Since PB-PCC and LR-Implicit represent the strongest members of non-adjusted classification aggregation and generative modeling, respectively, we further compare their results. When varying synthetic training prevalence across 0.1 to 0.9 (Figure 5a), LR-Implicit has much better MAE in all settings except near the natural prevalence (the test groups have, on average, 0.82 positive prevalence), and consistently stronger CI coverage.

Figure 5b shows results for natural class prevalence when varying the training set size. Unfortunately, LR-Implicit is disadvantaged at very small test sizes—its MAE is higher when there are only a few hundred training documents ($\leq 2^8 = 256$), though performance converges after that. We suspect this may occur because, when textual evidence is weak, the classifier learns to more heavily rely on its bias term, which can be a useful form of bias when the training class prevalence matches the test groups (on average). However, at all levels, LR-Implicit's coverage is better.

Since we hypothesized that PB-PCC may be overconfident for large test groups (§3.5), we test this by binning test groups by the number of documents per group. Figure 4 confirms that PB-PCC exhibits overconfidence for larger groups (smaller CI width alongside lower CI coverage), but LR-Implicit suffers from the same problem as well.

6 Additional Related Work

González et al. (2017a) reviews the class prevalence estimation literature, and we note a few threads of work here. Bella et al. (2010) propose a probabilistic variant of ACC, and Esuli and Sebastiani (2015) compare many methods on news article topics (RCV1) and medical record subject heading (OHSUMED-S) class prevalence tasks, finding varying results among CC, ACC, and PCC. A number of other empirical evaluations were conducted in two SemEval Twitter sentiment prevalence shared tasks, with varying results among these and other methods with a range of classifiers (Nakov et al., 2016; Rosenthal et al., 2017); Nakov et al. note that CC was often one of the strongest methods. Esuli and Sebastiani as well as Xue and Weiss (2009) present semi-supervised loss-augmented classifier training methods to improve prevalence estimation. Tasche (2017) presents theoretical results for ACC and Saerens et al.'s EM method (what we call the LR-Implicit MLE), arguing they correctly predict θ^* under class prior shift; we confirm that those two methods are indeed better than many alternatives in our empirical evaluation. While we focus on inference of the test-time class prior as a class prevalence estimate, Saerens et al. (2002) also show their method can improve individual-level classification accuracy, which Sulc and Matas (2018) use for image classification. (From the viewpoint of individual classification, this phenomenon is known as prior probability shift (Moreno-Torres et al., 2012).) González et al. (2017b) and Card and Smith (2018), similarly to our results, find that CC is much poorer than ACC under class shift. Card and Smith also show that PCC can be sensitive to properties of the classifier, finding that well-calibrated classifiers can give strong performance. They argue that discriminative aggregation models are appropriate for tasks where humans respond to text. Jerzak et al. (2018) analyze issues in class prevalence estimation and propose the ReadMe2 algorithm, which adds external word embeddings, optimization-based dimension reduction, and similarity matching to ReadMe's moment-matching framework.

7 Conclusion

Document class prevalence estimation is a widespread and much understudied task. We show that simple and obvious classifier aggregation methods display consistent biases, especially under class prior shift. Given how widely some of the less effective methods are used, machine learning and natural language processing research could have real impact in this space.

We also call attention to the need for *uncertainty aware* inference—methods that give confidence intervals to summarize their uncertainty. While our method is a first step, future work is necessary to better understand the problem and develop methods with improved coverage. Also, our framework can accommodate a wide array of document and language models—while we focus on bag-of-words models, recent advances in sequence, neural, and attention-based document models could be added directly to our generative model, or used as a discriminative-implicit component. The overall framework could also be extended to multiclass, and potentially, structured prediction settings.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A Bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Antonio Bella, Cesar Ferri, José Hernández-Orallo, and Maria Jose Ramirez-Quintana. 2010. Quantification via probability estimators. In *IEEE 10th International Conference on Data Mining (ICDM)*.

George Bissias, Brian Levine, Marc Liberatore, Brian Lynn, Juston Moore, Hanna Wallach, and Janis Wolak. 2016. Characterization of contact offenders and child exploitation material trafficking on five peer-to-peer networks. *Child Abuse & Neglect*, 52:185 – 199.

Dallas Card and Noah A Smith. 2018. The importance of calibration for estimating proportions from annotations. In *Proceedings of Empirical Methods in Natural Language Processing*.

Andrea Ceron, Luigi Curini, and Stefano M Iacus. 2015. Using sentiment analysis to monitor electoral campaigns: Method matters—evidence from the United States and Italy. *Social Science Computer Review*, 33(1):3–20.

Sean X. Chen and Jun S. Liu. 1997. Statistical applications of the Poisson-binomial and conditional Bernoulli distributions. *Statistica Sinica*, pages 875–892.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proceedings of ICML*.

Andrea Esuli and Fabrizio Sebastiani. 2015. Optimizing text quantifiers for multivariate loss functions. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(4):27.

George Forman. 2005. Counting positives accurately despite inaccurate classification. In *European Conference on Machine Learning*.

George Forman. 2008. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17(2):164–206.

John J. Gart and Alfred A. Buck. 1966. Comparison of a screening test and a reference test in epidemiologic studies. II. A probabilistic model for the comparison of diagnostic tests. *American Journal of Epidemiology*, 83(3):593–602.

Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, Erez Lieberman Aiden, and Li Fei-Fei. 2017. Using deep learning and Google Street View to estimate the demographic makeup of neighborhoods across the United States. *Proceedings of the National Academy of Sciences*.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.

Pablo González, Alberto Castaño, Nitesh V. Chawla, and Juan José Del Coz. 2017a. A review on quantification learning. *ACM Computing Surveys*, 50(5):74:1–74:40.

Pablo González, Jorge Díez, Nitesh Chawla, and Juan José del Coz. 2017b. Why is quantification an interesting learning problem? *Progress in Artificial Intelligence*, 6(1):53–58.

Justin Grimmer, Solomon Messing, and Sean J Westwood. 2012. How words and money cultivate a personal vote: The effect of legislator credit claiming on constituent credit allocation. *American Political Science Review*, 106(4):703–719.

Daniel J. Hopkins and Gary King. 2010. A method of automated nonparametric content analysis for social science. *American Journal of Political Science*, 54(1):229–247.

Connor T. Jerzak, Gary King, and Anton Strezhnev. 2018. An improved method of automated nonparametric content analysis for social science. *Working paper*.

Gary King, Jennifer Pan, and Margaret E Roberts. 2013. How censorship in china allows government criticism but silences collective expression. *American Political Science Review*, 107(2):326–343.

Benjamin Mandel, Aron Culotta, John Boulahanis, Danielle Stark, Bonnie Lewis, and Jeremy Rodrigue. 2012. A demographic analysis of online sentiment during Hurricane Irene. In *Proceedings of the Second Workshop on Language in Social Media*, pages 27–36. Association for Computational Linguistics.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin' Words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372.

Jose G. Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. 2012. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 Task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.

Andrew Ng and Michael Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems (NIPS)*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 Task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Marco Saerens, Patrice Latinne, and Christine Decaestecker. 2002. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1):21–41.

Yanchuan Sim, Brice Acree, Justin H. Gross, and Noah A. Smith. 2013. Measuring ideological proportions in political speeches. In *Proceedings of EMNLP*.

Milan Sulc and Jiri Matas. 2018. Improving cnn classifiers by estimating test-time priors. *arXiv preprint arXiv:1805.08235*.

Matt Taddy. 2013. Multinomial inverse regression for text analysis. *Journal of the American Statistical Association*, 108(503):755–770.

Dirk Tasche. 2017. Fisher consistency for prior probability shift. *Journal of Machine Learning Research*, 18(95):1–32.

Rob Voigt, Nicholas P Camp, Vinodkumar Prabhakaran, William L Hamilton, Rebecca C Hetey, Camilla M Griffiths, David Jurgens, Dan Jurafsky, and Jennifer L Eberhardt. 2017. Language from police body camera footage shows racial disparities in officer respect. *Proceedings of the National Academy of Sciences*.

Slobodan Vucetic and Zoran Obradovic. 2001. Classification on data with biased class distribution. In *European Conference on Machine Learning*, pages 527–538. Springer.

William Yang Wang, Elijah Mayfield, Suresh Naidu, and Jeremiah Dittmar. 2012. Historical analysis of legal opinions with a sparse mixed-effects latent variable model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Larry Wasserman. 2011. *All of statistics*. Springer Science & Business Media.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*.

Jack Chongjie Xue and Gary M Weiss. 2009. Quantification and semi-supervised classification methods for handling changes in class distribution. In *Proceedings of KDD*.

Challenges of Using Text Classifiers for Causal Inference

Zach Wood-Doughty***, Ilya Shpitser**†**, Mark Dredze***†
Department of Computer Science
*Center for Language and Speech Processing
†Malone Center for Engineering in Healthcare
Johns Hopkins University, Baltimore, MD 21218
`{zach,ilyas,mdredze}@cs.jhu.edu`

Abstract

Causal understanding is essential for many kinds of decision-making, but causal inference from observational data has typically only been applied to structured, low-dimensional datasets. While text classifiers produce low-dimensional outputs, their use in causal inference has not previously been studied. To facilitate causal analyses based on language data, we consider the role that text classifiers can play in causal inference through established modeling mechanisms from the causality literature on missing data and measurement error. We demonstrate how to conduct causal analyses using text classifiers on simulated and Yelp data, and discuss the opportunities and challenges of future work that uses text data in causal inference.

1 Introduction

Most scientific analyses, in domains from economics to medicine, focus on low-dimensional structured data. Many such domains also have unstructured text data; advances in natural language processing (NLP) have led to an increased interest in incorporating language data into scientific analyses. While language is inherently unstructured and high dimensional, NLP systems can be used to process raw text to produce structured variables. For example, work on identifying undiagnosed side effects from electronic health records (EHR) uses text classifiers to produce clinical variables from the raw text (Hazlehurst et al., 2009).

NLP tools may also benefit the study of causal inference, which seeks to identify causal relations from observational data. Causal analyses traditionally use low-dimensional structured variables, such as clinical markers and binary health outcomes. Such analyses require assumptions about the data-generating process, which are often simpler with low-dimensional data. Unlike prediction tasks which are validated by held-out test sets, causal inference involves modeling counterfactual random variables (Neyman, 1923; Rubin, 1976) that represent the outcome of some hypothetical intervention. To rigorously reason about hypotheticals, we use causal models to link our counterfactuals to observed data (Pearl, 2009).

NLP provides a natural way to incorporate text data into causal inference models. We can produce low-dimensional variables using, for example, text classifiers, and then run our causal analysis. However, this straightforward integration belies several potential issues. Text classification is not perfect, and errors in a NLP algorithm may bias subsequent analyses. Causal inference requires understanding how variables influence one another and how correlations are confounded by common causes. Classic methods such as stratification provide a means for handling confounding of categorical or continuous variables, but it is not immediately obvious how such work can be extended to high-dimensional data.

Recent work has approached high-dimensional domains by using random forests (Wager and Athey, 2017) and other methods borrowed from machine learning (Chernozhukov et al., 2016). But even compared to an analysis that requires hundreds of confounders (Belloni et al., 2014), NLP models with millions of variables are very high-dimensional. While physiological symptoms reflect complex biological realities, many symptoms such as blood pressure are one-dimensional variables. While doctors can easily quantify the effect of high blood pressure on some outcome, can we use the “positivity” of a restaurant review to estimate a causal effect? More broadly, is it possible to employ text classification methods in a causal analysis?

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4586–4598
Brussels, Belgium, October 31 - November 4, 2018.

We explore methods for the integration of text classifiers into causal inference analyses that consider confounds introduced by imperfect NLP. We show what assumptions are necessary for causal analyses using text, and discuss when those assumptions may or may not be reasonable. We draw on the causal inference literature to consider two modeling aspects: missing data and measurement error. In the missing data formulation, a variable of interest is sometimes unobserved, and text data gives us a means to model the missingness process. In the measurement error formulation, we use a text classifier to generate a noisy proxy of the underlying variable.

We highlight practical considerations of a causal analysis with text data by conducting analyses with simulated and Yelp data. We examine the results of both formulations and show how a causal analysis which properly accounts for possible sources of bias produces better estimates than naïve methods which make unjustified assumptions. We conclude by examining how our approach may enable new research avenues for inferring causality with text data.

2 Causal Inference, Briefly

While randomized control trials (RCT) are the gold standard of determining causal effects of treatments on outcomes, they can be expensive or impossible in many settings. In contrast, the world is filled with observational data collected without randomization. While most studies simply report correlations from observational data, the field of causal inference examines what assumptions and analyses make it possible to identify causal effects.

We formalize a causal statement like "smoking causes cancer" as "if we were to conduct a RCT and assign smoking as a treatment, we would see a higher incidence of cancer among those assigned smoking than among the control group." In the framework of Pearl (1995), we consider a counterfactual variable of interest: what *would have* been the cancer incidence among smokers if smoking *had been* randomized? Specifically, we consider a causal effect as the counterfactual outcome of a hypothetical intervention on some treatment variable. If we denote smoking as our treatment variable A and cancer as our outcome variable Y, then we are interested in the counterfactual distribution, denoted $p(Y(a))$ or $p(Y \mid \text{do}(a))$. We interpret this as "the distribution over Y had A been set, possibly contrary to fact, to value a." For a binary treatment A, the causal effect of A on Y is denoted $\tau = E[Y(1)] - E[Y(0)]$; the average difference between if you had received the treatment and if you had not. Throughout, we use causal directed acyclic graphs (DAG), which assumes that an intervention on A is well-defined and results in a counterfactual variable $Y(a)$ (Pearl, 1995; Dawid, 2010).

Figure 1a shows an example of simple confounding. This is the simplest DAG in which counterfactual distribution $p(Y(a))$ is not simply $p(Y \mid A)$, as C influences both the treatment A and the outcome Y. To recover the counterfactual distribution $p(Y(a))$ that would follow an intervention upon A, we must "adjust" for C, applying the so-called "back-door criterion" (Pearl, 1995). We can then derive the counterfactual distribution $p(Y(a))$ and desired causal effect, τ_S as a function of the observed data, (Fig. 4 Eq. 1.) This derivation is shown in Appendix A.

Note that $p(Y(a))$ and τ_S require data on C, and if C is not in fact observed, it is impossible to recover the causal effect. Formally, we say that $p(Y(a))$ is *not identified* in the model, meaning there is no function f such that $p(Y(a)){=}f(p(A,Y))$. Identifiability is a primary concern of causal inference (Shpitser and Pearl, 2008).

Throughout, we assume for simplicity that A, C, and Y are binary variables. While text classifiers can convert high-dimensional data into binary variables for such analyses, we need to make further assumptions about how classification errors affect causal inferences. We cannot assume that the output of a text classifier can be treated as if it were ground truth. To conceptualize the ways in which a text classifier may be biased, we will consider them as a way to recover from missing data or measurement error.

3 Causal Models

Real-world observational data is messy and often imperfectly collected. Work in causal inference has studied how analyses can be made robust to missing data or data recorded with systematic measurement errors.

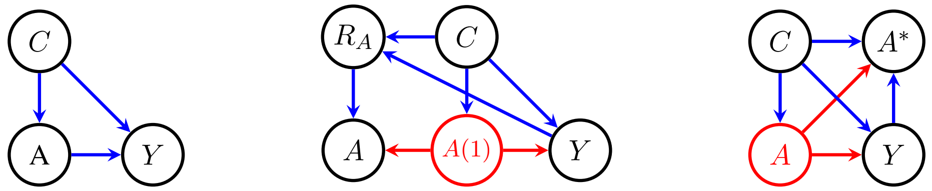

(a) Simple Confounding (b) Missing Data (c) Measurement Error

Figure 1: DAGs for causal inference without text data. Red variables are unobserved. A is a treatment, Y is an outcome, and C is a confounder.

A	C	Y
1	1	0
0	1	1
0	0	1
1	0	1

(a) Simple Confounding

R_A	A	C	Y
1	1	1	0
0	?	1	1
1	0	0	1
0	?	0	1

(b) Missing Data

A^*	C	Y
0	1	0
0	1	1
0	0	1
1	0	1

(c) Measurement Error

A^*	A
1	1
0	1
0	0
1	1

(d) Mismeasurement

Figure 2: Example data rows for causal inference without text data.

3.1 Missing Data

Our dataset has "missing data" if it contains individuals (instances) for which some variables are unobserved, even though these variables are typically available. This may occur if some survey respondents choose not to answer certain questions, or if certain variables are difficult to collect and thus infrequently recorded. Missing data is closely related to causal inference – both are interested in hypothetical distributions that we cannot directly observe (Robins et al., 2000; Shpitser et al., 2015).

Consider a causal model where A is sometimes missing (Figure 1b). The variable R_A is a binary indicator for whether A is observed ($R_A = 1$) or missing. The variable $A(R_A = 1)$, written as $A(1)$, represents the counterfactual value of A were it never missing. Finally, A is the observed proxy for $A(1)$: it has the same value as $A(1)$ if $R_A = 1$, and the special value "?" if $R_A = 0$.

Solving missingness can seen as intervening to set R_A to 1. Given $p(A, R_A, C, Y)$, we want to recover $p(A(1), C, Y)$. We may need to make a "Missing at Random" (MAR) assumption, which says that the missingness process is independent of the true missing values, conditional on observed values. Figure 1b reflects the MAR assumption; R_A is independent of $A(1)$ given fully-observed C and Y. If an edge existed from $A(1)$ to R_A, we have "Missing Not at Random" (MNAR) and would not be identified except in special cases (Shpitser et al., 2015).

3.2 Measurement Error

Sometimes a necessary variable is never observed, but is instead proxied by a variable which differs from the truth by some error. Consider the example of body mass index (BMI) as a proxy for obesity in a clinical study. Obesity is a known risk factor for many health outcomes, but has a complex clinical definition and is nontrivial to measure. BMI is a simple deterministic function of height and weight. To conduct a causal analysis of obesity on cancer when only BMI and cancer are measured, we can proceed as if we had measured obesity and then correct our analysis for the known error that comes from using BMI as a proxy for obesity (Hernán and Cole, 2009; Michels et al., 1998).

To generalize this concept, we can replace obesity with our ground truth variable A and replace BMI with a noisy proxy A^*. Figure 1c gives the DAG for this model. Unlike missing data problems, there is no hypothetical intervention which recovers the true data distribution $p(A, C, Y)$. Instead, we manipulate the observed distribution $p(A^*, C, Y)$ with the known relationship $p(A^*, A)$ to recover the desired $p(A, C, Y)$.

Unlike missing data, measurement error conceptualization can be used even when we never observe A (e.g. the table in Figure 2c) as long as we have knowledge about the error mechanism $p(A^*, A)$. Using this knowledge, we can correct for the error using 'matrix

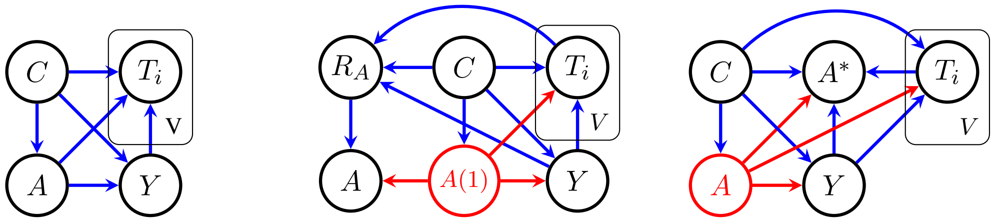

(a) Simple Confounding with Text (b) Missing Data with Text (c) Measurement Error with Text

Figure 3: DAGs for causal inference with text data. In the Yelp experiments we discuss, T_i influences Y and not the other way around.

$$\tau_S = \sum_C \left(p(Y=1 \mid A=1, C) - p(Y=1 \mid A=0, C)\right) p(C) \tag{1}$$

$$\begin{aligned}\tau_{\text{MD}} = \sum_C \Bigg(& \frac{p(A=1 \mid \mathbf{T}, C, Y=1, R_A=1)}{\sum_y' p(A=1 \mid \mathbf{T}, C, y', R_A=1) p(Y=y' \mid C)} \\ & - \frac{p(A=0 \mid \mathbf{T}, C, Y=1, R_A=1)}{\sum_y' p(A=0 \mid \mathbf{T}, C, Y=y', R_A=1) p(Y=y' \mid C)} \Bigg) p(Y=1, C)\end{aligned} \tag{2}$$

$$\tau_{\text{ME}} = \sum_C \left(\frac{\frac{-\delta_{c,y=1} q_{c,y=1}(0) + (1-\delta_{c,y=1}) q_{c,y=1}(1)}{(1-\epsilon_{c,y=1}-\delta_{c,y=1})}}{\sum_{y'} \frac{-\delta_{c,y'} q_{c,y'}(0) + (1-\delta_{c,y'}) q_{c,y'}(1)}{(1-\epsilon_{c,y'}-\delta_{c,y'})}} - \frac{\frac{(1-\epsilon_{c,y=1}) q_{c,y=1}(0) - \epsilon_{c,y=1} q_{c,y=1}(1)}{(1-\epsilon_{c,y=1}-\delta_{c,y=1})}}{\sum_{y'} \frac{(1-\epsilon_{c,y'}) q_{c,y'}(0) - \epsilon_{c,y'} q_{c,y'}(1)}{(1-\epsilon_{c,y'}-\delta_{c,y'})}} \right) p(C) \tag{3}$$

Define $\epsilon_{c,y} = p(A=0 \mid A^*=1, C=c, Y=y)$, $\delta c, y = p(A=1 \mid A^*=0, C=c, Y=y)$,
$q_{c,y}(0) = p(C=c, Y=y, A^*=0)$, and $q_{c,y}(1) = p(C=c, Y=y, A^*=1)$.

Figure 4: Functionals for the Causal Effects for Simple Confounding (τ_{SC}), Missing Data (τ_{MD}) and Measurement Error (τ_{ME}). Derivations are in Appendices A, B, and C.

adjustment' (Pearl, 2010). In practice we might learn $p(A^*, A)$ from data such as that found in Figure 2d. Recent work has also considered how multiple independent proxies of A could allow identification without any data on $p(A^*, A)$ (Kuroki and Pearl, 2014).

4 Causal Models for Text Data

We can use conceptualizations for missing data and measurement error to support causal analyses with text data. The choice of model depends on the assumptions we make about the data-generation process.

We add new variables to our models (Figure 1a) to represent text, which produces the data-generating distribution shown in Figure 3a. This model assumes that the underlying A, C, and Y variables are generated before the text variables; we use text to recover the true relationship between A and Y.

We represent text as an arbitrary set of V variables, which are independent of one another given the non-text variables. In our implemented analyses we will represent text as a bag-of-words, wherein each T_i is simply the binary indicator of the presence of the i-th word in our vocabulary of V words, and $\mathbf{T} = \cup_i T_i$. The restriction to simple text models allows us to explore connections to causal inference applications, though future work could relax assumptions of the text models to be inclusive of more sophisticated text models (e.g. neural sequence models (Lai et al., 2015; Zhang et al., 2015)), or consider causal relationships between two text variables.

To motivate our explanations, consider the task of predicting an individuals' smoking status from free-text hospital discharge notes (Uzuner et al., 2008; Wicentowski and Sydes, 2008). Some hospitals do not explicitly record patient smoking status as structured data, making it difficult to use such data in a study on the outcomes of smoking. We will suppose that we are given a dataset with patient data on lung cancer outcome (Y) and age (C), that our data on smoking status (A) is affected by either missing data or measurement error, but that we have text data ($\mathbf{T}$) from discharge records that will allow us to infer smoking status with reasonable accuracy.

4.1 Missing Data

To show how we might use text data to recover from missing data, we introduce missingness for A from Figure 3a to get the model in Figure 3b. The missing arrow from $A(1)$ to R_A encodes the MAR assumption, which is sufficient to make it possible to identify the full data distribution from the observed data.

Suppose our motivation is to estimate the causal effect of smoking status (A) on lung cancer (Y) adjusting for age (C). Imagine that missing data arises because hospitals sometimes – but not always – delete explicit data on smoking status from patient records. If we have access to patients' discharge notes ($\mathbf{T}$) and know whether a given patient had smoking status recorded (R_A), then the DAG in Figure 3b may be a reasonable model for our setting. Note that we must again assume that A does not directly affect R_A.

The causal effect of A on Y in Figure 3b is identified as τ_{MD}, given in Eq. 2 in Figure 4. The derivation is given in Appendix B.

4.2 Measurement Error

We model text data with measurement error by introducing a proxy A^* to the model in Figure 3c. We assume that the proxied value of A^* can depend upon all other variables, and that we will be able to estimate $p(A^*, A)$ given an external dataset, e.g. text classifier accuracy on held-out data.

Suppose we again want to estimate the causal effect from §4.1, but this time none of our hospital records contain explicit data on smoking status. However, imagine that we have a separate training dataset of medical discharge records annotated by expert pulmonologists for patients' smoking status. We could then train a classifier to predict smoking status using discharge record text[1].

Working from the derivation for matrix adjustment in binary models given by Pearl (2010), we identify the causal effect of A on Y (Figure 3c) as τ_{ME} (Eq 3 in Figure 4.) The derivation is in Appendix C.

5 Experiments

We now empirically evaluate the effectiveness of our two conceptualizations (missing data and measurement error) for including text data in causal analyses. We induce missingness or mismeasurement of the treatment variable and use text data to recover the true causal relationship of that treatment on the outcome. We begin with a simulation study with synthetic text data, and then conduct an analysis using reviews from `yelp.com`.

5.1 Synthetic Data

We select synthetic data so that we can control the entire data-generation process. For each data row, we first sample data on three binary variables (A, C, Y) and then sample V different binary variables T_i representing a V-vocabulary bag-of-words. A graphical model for this distribution appears in Figure 3a. We augment this distribution to introduce either missing data (Figure 3b) or measurement error (Figure 3c.) For measurement error, we sample two datasets. A small training set which gives data on $p(A, C, Y, \mathbf{T})$ and a large test set which gives data on $p(C, Y, \mathbf{T})$.

The full data generating process appears in Appendix D, and the implementation (along with all our code) is provided online[2].

5.2 Yelp Data

We utilize the 2015 Yelp Dataset Challenge[3] which provides 4.7M reviews of local businesses. Each review contains a one- to five-star rating, up to 5,000 characters of text. Yelp users can flag reviews as "Useful" as a mark of quality.

We extract treatment, outcome, and confounder variables from the structured data. The treatment is a binarized user rating that takes value 1 if the review has four or five stars and value 0 if the review has one or two stars. Three-star reviews are discarded from our analysis. The outcome is whether the review received at least one "Useful" flag. The confounder is whether the review's author has received at least two "Useful" flags across all reviews, according to their user object. In our data, 74.2% of reviews were positive, 42.6% of reviews were flagged as "Useful," and 56.7% users had received at least

---

[1] This is the precise setting of Uzuner et al. (2008).

[2] `github.com/zachwooddoughty/emnlp2018-causal`

[3] `yelp.com/dataset/challenge`

two such flags. We preprocess the text of each review by lowercasing, stemming, and removing stopwords, before converting to a bag-of-words representation with the 4,334 word vocabulary of all words which appeared at least 1000 times in a sample of 1M reviews.

Based on this $p(A, C, Y, \mathbf{T})$ distribution, we assume the data-generating process that matches Figure 3a and introduce missingness and mismeasurement as before, giving us data-generating processes matching Figures 3b and 3c.

Our intention is not to argue about a true real-world causal effect of Yelp reviews on peer behavior: we do not believe that our confounder is the only common cause of the author's rating and the platform's response. We leave for future work a case study that jointly addresses questions of identifiability and estimation of a real-world causal effect. In this work, our experiments focus on a simpler task: can a correctly-specified model that uses text data effectively estimate a causal effect in the presence of missing data or measurement error.

5.3 Models

We now introduce several baseline methods which, unlike our correctly specified models τ_{MD} and τ_{ME}, are not consistent estimators of our desired causal effect. We would expect that the theoretical bias in these estimators would result in poor performance in our experiments.

5.3.1 Baseline: Naïve Model

In both the missing data and measurement error settings, our models use some rows that are full observed. In missing data, these are rows where $R_A = 1$; in measurement error, the training set is sampled from the true distribution. The simplest approach to handling imperfect data is to throw away all rows without full data, and calculate Eq 1 from that data. In Figure 5, these are labeled as `*.naive`.

5.3.2 Baseline: Textless Model

In Figure 3b, if we do not condition on T_i to d-separate $A(1)$ from its missingness indicator, that influence may bias our estimate. While we know that ignoring text may introduce asymptotic bias into our estimates of the causal effect, we empirically evaluate how much bias is produced by this "Textless" model compared to a correct model. This is labeled as `*.no_text` in Figure 5 (a).

In principle, we could conduct a measurement error analysis using a model that does not include text. In practice, we found we could not impute A^* from C and Y alone. The non-textual classifier had such high error that the adjustment matrix was singular and we could not compute the effect. Thus, we have no such baseline in our measurement error results.

5.3.3 Baseline: `no_y` and `unadjusted` Models

In Figure 3b, we must also condition on C and Y to d-separate $A(1)$ from its missingness indicator. In our misspecified model for missing data, we do not condition on Y, leaving open a path for $A(1)$ to influence its missingness. In Figure 5 (a), this model is labeled as `*.no_y`.

When correcting for measurement error, a crucial piece of the estimation is the matrix adjustment using the known error between the proxy and the truth. A straightforward misspecified model for measurement error is to impute a proxy for each row in our dataset and then calculate the causal effect assuming no error between the proxy and truth. This approach, while simplistic, can be thought of as using a text classifier as a proxy without regard for the text classifier's biases. In Figure 5 (b), this approach is labeled as `*.unadjusted`.

5.3.4 Correct Models

Finally, we consider the estimation approaches presented in §4.1 and §4.2. For the missing data causal effect (τ_{MD} from Eq 2) we use a multiple imputation estimator which calculates the average effect across 20 samples from $p(A|\mathbf{T}, C, Y)$ for each row where $R_A = 0$. For the measurement error causal effect (τ_{ME} from Eq 3), we use the training set of $p(A, C, Y, \mathbf{T})$ data to estimate $\epsilon_{c,y}$ and $\delta c, y$ and the larger set of $p(C, Y, \mathbf{T})$ data to estimate $q_{c,y}$ and $p(C)$.

These models are displayed in Figure 5 (a) as `*.full` and in Figure 5 (b) `*.adjusted`.

5.4 Evaluation

Each model takes in a data sample with missingness or mismeasurement, and outputs an estimate of the causal effect of A on Y in the underlying data. Rather than comparing models' estimates against a population-level estimate, we compare against an estimate of the effect computed on the same data sample, but without any missing data or measurement error. This 'perfect data estimator' may still make errors given the finite data sample. We compare against this estimator to avoid a small-sample case where an estimator gets lucky. In Figure 5, we plot data sample size against the squared distance of each model's estimate from a perfect data estimator's estimate, averaged over ten runs. Figure 6 in Appendix E contains a second set of experiments using a larger vocabulary.

6 Results

Given that our correctly-specified models are proven to be asymptotically consistent, we would expect them to outperform misspecified models. However, for any given dataset, asymptotic consistency provides no guarantees.

6.1 Missing Data

The missing data (MD) experiments suggest that the correct `full` model does perform best. The `no_y` model performs approximately as well as the correct model on the synthetic data, but not on the Yelp data. The difference between the `no_y` and `full` missing data models is simply a function of the effect of Y on R_A. We could tweak our synthetic data distribution to increase the influence of Y to make the `no_y` model perform worse.

When we initially considered other data-generating distributions for missing data, we found that when we reduced the influence of the text variables on R_A, the `no_text` and `naive` models approached the performance of the correctly-specified model. While intuitive, this reinforces that the underlying distribution matters a great deal in how modeling choices may introduce biases if incorrectly specified.

6.2 Measurement error

The measurement error results tell a more interesting story. We see enormous fluctuations of the `adjusted` model, and in the synthetic data, the `unadjusted` model appears to be quite superior.

In the synthetic dataset, this is likely because our text classifier had near-perfect accuracy, and so simple approach of assuming its predictions were ground-truth introduced less bias. A broader issue with the `adjusted` model is that the matrix adjustment approach requires dividing by (potentially very small) probabilities, this sometimes resulted in huge over-corrections. In addition, since those probabilities are estimated from a relatively small training dataset, small changes to the error-estimate can propagate to huge changes in the final casual estimate.

This instability of the matrix adjustment approach may be a bigger problem for text and other high-dimensional data: unlike in our earlier example of BMI and obesity, there are likely no simple relationships between text and clinical variables. However, instead of using matrix adjustment as a way to recover the true effect, we may instead use it to bound the error our proxy may introduce. As mentioned by Pearl (2010), when $p(A \mid A^*)$ is not known exactly, we can use a Bayesian analysis to bound estimates of a causal effect. In a downstream task, this would let us explore the stability of our `adjusted` results.

7 Related Work

A few recent papers have considered the possibilities for combining text data with approaches from the causal inference literature. Landeiro and Culotta (2016) and Landeiro and Culotta (2017) explored text classification when the relationship between text data and class labels are confounded. Other work has used propensity scores as a way to extract features from text data (Paul, 2017) or to match social media users based on what words they write (De Choudhury et al., 2016). The only work we know of which seeks to estimate causal effects using text data focuses on effects *of* text or effects *on* text (Egami et al., 2018; Roberts et al., 2018). In our work, our causal effects do not include text variables: we use text variables to recover an underlying distribution and then estimate a causal effect within that distribution.

There is a conceptually related line of work in the NLP community on inferring causal relationships expressed in text (Girju, 2003; Kaplan and Berry-Rogghe, 1991). However, our work is fundamentally different. Rather than identify casual relations expressed via language, we are using text data in a causal model to identify the strength of an underlying causal effect.

Figure 5: Experimental results. Squared distance (y-axis, lower is better) of the estimated causal effect from τ_{SC} calculated from the full data with no missing data or measurement error. Error bars (negligible for larger datasets) are 1.96 times standard error across 10 experiments. Additional experiments with a larger vocabulary are shown in Appendix E.

8 Future Directions

While this paper addresses some initial issues arising from using text classifiers in causal analyses, many challenges remain. We highlight some of these issues as directions for future research.

We provided several proof-of-concept models for estimating effects, but our approach is flexible to more sophisticated models. For example, a semi-parametric estimator would make no assumptions about the text data distribution by wrapping the text classifier into an infinite-dimensional nuisance model (Tsiatis, 2007). This would enable estimators robust to partial model misspecification (Bang and Robins, 2005).

Choices in the design of statistical models of text consider issues like accuracy and tractability. Yet if these models are to be used in a causal framework, we need to understand how modeling assumptions introduce biases and other issues that can interfere with a downstream causal analysis. To take an example from the medical domain, we know that doctors write clinical notes throughout the healthcare process, but it is not obvious how to model this data-generating process. We could assume that the doctor's notes passively record a patient's progression, but in reality it may be that the content of the notes themselves actively change the patient's care; causality could work in either direction.

New lines of work in causality may be especially helpful for NLP. In this work, we used simple logistic regression on a bag-of-words representation of text; using state-of-the-art text models will likely require more causal assumptions. Nabi and Shpitser (2017) develops causality-preserving dimensionality reduction, which could help develop text representations that preserve causality.

Finally, we are interested in case studies on incorporating text classifiers into real-world causal analyses. Many health studies have used text classifiers to extract clinical variables from EHR data (Meystre et al., 2008). These works could be extended to study causal effects involving those extracted variables, but such extensions would require an understanding of the underlying assumptions. In any given study, the necessity and appropriateness of assumptions will hinge on domain expertise. The conceptualizations outlined in this paper, while far from solving all issues of causality and text, will help those using text classifiers to more easily consider research questions of cause and effect.

Acknowledgments

This work was in part supported by the National Institute of General Medical Sciences under grant number 5R01GM114771 and by the National Institute of Allergy and Infectious Diseases under grant number R01 AI127271-01A1. We thank the anonymous reviewers for their helpful comments.

References

Heejung Bang and James M Robins. 2005. Doubly robust estimation in missing data and causal inference models. *Biometrics*, 61(4):962–973.

Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. 2014. Inference on treatment effects after selection among high-dimensional controls. *The Review of Economic Studies*, 81(2):608–650.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, and Whitney K Newey. 2016. Double machine learning for treatment and causal parameters. Technical report, cemmap working paper, Centre for Microdata Methods and Practice.

A Philip Dawid. 2010. Beware of the dag! In *Causality: Objectives and Assessment*, pages 59–86.

Munmun De Choudhury, Emre Kiciman, Mark Dredze, Glen Coppersmith, and Mrinal Kumar. 2016. Discovering shifts to suicidal ideation from mental health content in social media. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 2098–2110. ACM.

Naoki Egami, Christian J Fong, Justin Grimmer, Margaret E Roberts, and Brandon M Stewart. 2018. How to make causal inferences using texts. *arXiv preprint arXiv:1802.02163*.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12*, pages 76–83. Association for Computational Linguistics.

Brian Hazlehurst, Allison Naleway, and John Mullooly. 2009. Detecting possible vaccine adverse events in clinical notes of the electronic medical record. *Vaccine*, 27(14):2077–2083.

Miguel A Hernán and Stephen R Cole. 2009. Invited commentary: causal diagrams and measurement bias. *American journal of epidemiology*, 170(8):959–962.

Randy M Kaplan and Genevieve Berry-Rogghe. 1991. Knowledge-based acquisition of causal relationships in text. *Knowledge Acquisition*, 3(3):317–337.

Manabu Kuroki and Judea Pearl. 2014. Measurement bias and effect restoration in causal inference. *Biometrika*, 101(2):423–437.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Virgile Landeiro and Aron Culotta. 2016. Robust text classification in the presence of confounding bias. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Virgile Landeiro and Aron Culotta. 2017. Controlling for unobserved confounds in classification using correlational constraints. *arXiv preprint arXiv:1703.01671*.

Stéphane M Meystre, Guergana K Savova, Karin C Kipper-Schuler, and John F Hurdle. 2008. Extracting information from textual documents in the electronic health record: a review of recent research. *Yearbook of medical informatics*, 17(01):128–144.

Karin B Michels, Sander Greenland, and Bernard A Rosner. 1998. Does body mass index adequately capture the relation of body composition and body size to health outcomes? *American Journal of Epidemiology*, 147(2):167–172.

Razieh Nabi and Ilya Shpitser. 2017. Semi-parametric causal sufficient dimension reduction of high dimensional treatments. *arXiv preprint arXiv:1710.06727*.

Jerzy Neyman. 1923. Sur les applications de la thar des probabilities aux experiences agaricales: Essay des principle. excerpts reprinted (1990) in English. *Statistical Science*, 5:463–472.

Michael J Paul. 2017. Feature selection as causal inference: Experiments with text classification. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 163–172.

Judea Pearl. 1995. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688.

Judea Pearl. 2009. *Causality*. Cambridge university press.

Judea Pearl. 2010. On measurement bias in causal inference. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 425–432. AUAI Press.

Margaret E Roberts, Brandon M Stewart, and Richard A Nielsen. 2018. Adjusting for confounding with text matching.

James M Robins, Andrea Rotnitzky, and Daniel O Scharfstein. 2000. Sensitivity analysis for selection bias and unmeasured confounding in missing data and causal inference models. In *Statistical models in epidemiology, the environment, and clinical trials*, pages 1–94. Springer.

D. B. Rubin. 1976. Causal inference and missing data (with discussion). *Biometrika*, 63:581–592.

Ilya Shpitser, Karthika Mohan, and Judea Pearl. 2015. Missing data as a causal and probabilistic problem. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 802–811. AUAI Press.

Ilya Shpitser and Judea Pearl. 2008. Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, 9(Sep):1941–1979.

Anastasios Tsiatis. 2007. *Semiparametric theory and missing data*. Springer Science & Business Media.

Özlem Uzuner, Ira Goldstein, Yuan Luo, and Isaac Kohane. 2008. Identifying patient smoking status from medical discharge records. *Journal of the American Medical Informatics Association*, 15(1):14–24.

Stefan Wager and Susan Athey. 2017. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*.

Richard Wicentowski and Matthew R Sydes. 2008. Using implicit information to identify smoking status in smoke-blind medical discharge summaries. *Journal of the American Medical Informatics Association*, 15(1):29–31.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

A Simple Confounding

$$p(Y(a)) = \sum_C p(Y(a) \mid C)p(C) \tag{4}$$
$$= \sum_C p(Y(a) \mid A, C)p(C) \tag{5}$$
$$= \sum_C p(Y \mid A, C)p(C) \tag{6}$$

Eq 5 holds because $Y(a) \perp A \mid C$, as seen in Figure 1a. Plugging this distribution into $\tau_S = E[Y(1)] - E[Y(0)]$ gives us the causal effect presented in Figure 4, Eq 1.

This assumes that an intervention on A is well-defined; if we *did* conduct a randomized control trial, we *could* assign $A = a$ and break A's dependence on C.

In general, this step requires that we condition on all "back-door" paths between the treatment and the outcome. In Figure 1(a), if we did not have data on C, we could not block the back-door path between A and Y.

Eq 6 holds due to consistency. We assume that, given we intervened to set $A = a$, if that individual would have been assigned $A = a$ in nature, then the distribution over Y is the same.

B Missing Data

Denote $p(Y(A(1) = a)) = p(Y(a))$.

First, we identify the causal effect in terms of the true $A(1)$.

$$p(Y(a))$$
$$= \sum_C p(Y(a) \mid C)p(C) \tag{7}$$
$$= \sum_C p(Y(a) \mid A(1), C)p(C) \tag{8}$$
$$= \sum_C p(Y \mid A(1), C)p(C) \tag{9}$$

Where 7 holds by chain rule, 8 holds by $A(1) \perp Y(a) \mid C$, and 9 by consistency.

Now, we identify $A(1)$ in terms of observed data.

$$p(A(1), C, Y)$$
$$= p(A(1) \mid C, Y)p(C, Y) \tag{10}$$
$$= p(A(1) \mid C, Y, R_A = 1)p(C, Y) \tag{11}$$
$$= p(A \mid C, Y, R_A = 1)p(C, Y) \tag{12}$$

Where 10 holds by chain rule, 11 by $A(1) \perp R_A \mid C, Y$, and 12 by consistency.

Now, use Eq 12 to identify $p(Y \mid A(1), C)$ from Eq 9 in terms of observed data.

$$p(Y \mid A(1), C)$$
$$= \frac{p(Y, A(1), C)}{p(A(1), C)} \tag{13}$$
$$= \frac{p(Y, A(1), C)}{\sum_Y p(Y, A(1), C)} \tag{14}$$
$$= \frac{p(A \mid C, Y, R_A = 1)p(C, Y)}{\sum_Y p(A \mid C, Y, R_A = 1)p(C, Y)} \tag{15}$$
$$= \frac{p(A \mid C, Y, R_A = 1)p(Y \mid C)}{\sum_Y p(A \mid C, Y, R_A = 1)p(Y \mid C)} \tag{16}$$

Where 13 holds by definition, 14 holds by marginalization, 15 holds by an application of 12 twice, and 16 holds by canceling out p(C).

If we include text in this derivation, we simply replace $p(A \mid C, Y, R_A = 1)$ with $p(A \mid \mathbf{T}, C, Y, R_A = 1)$, where $\mathbf{T}$ is all our text variables.

Finally, combine Eq 9 and Eq 16 to get:

$$p(Y(A(1) = a))$$
$$= \sum_C \frac{p(A \mid C, Y, R_A = 1)p(Y \mid C)}{\sum_Y p(A \mid C, Y, R_A = 1)p(Y \mid C)} p(C) \tag{17}$$

Plugging this distribution into $\tau_{\text{MD}} = E[Y(1)] - E[Y(0)]$ gives us the causal effect presented in Figure 4, Eq 2.

C Measurement Error

Define the following terms for convenience:

$$\epsilon_{c,y} = p(A = 0 \mid A^* = 1, C = c, Y = y) \tag{18}$$
$$\delta c, y = p(A = 1 \mid A^* = 0, C = c, Y = y) \tag{19}$$
$$q_{c,y}(0) = p(C = c, Y = y, A^* = 0) \tag{20}$$
$$q_{c,y}(1) = p(C = c, Y = y, A^* = 1) \tag{21}$$

Eq (5) and (7) from Pearl 2010 gives us:

$$p(A = 1, C = c, Y = y)$$
$$= \frac{-\delta_{c,y} q_{c,y}(0) + (1 - \delta_{c,y}) q_{c,y}(1)}{(1 - \epsilon_{c,y} - \delta_{c,y})} \tag{22}$$
$$p(A = 0, C = c, Y = y)$$
$$= \frac{(1 - \epsilon_{c,y}) q_{c,y}(0) - \epsilon_{c,y} q_{c,y}(1)}{(1 - \epsilon_{c,y} - \delta_{c,y})} \tag{23}$$

Now,

$$
\begin{aligned}
& p(Y \mid A=1, C) \\
& \quad = \frac{p(Y, A=1, C)}{p(A=1, C)} && (24) \\
& \quad = \frac{p(Y, A=1, C)}{\sum_Y p(Y, A=1, C)} && (25) \\
& \quad = \frac{\dfrac{-\delta_{c,y} q_{c,y}(0) + (1-\delta_{c,y}) q_{c,y}(1)}{(1-\epsilon_{c,y}-\delta_{c,y})}}{\sum_{y'} \dfrac{-\delta_{c,y'} q_{c,y'}(0) + (1-\delta_{c,y'}) q_{c,y'}(1)}{(1-\epsilon_{c,y'}-\delta_{c,y'})}} && (26)
\end{aligned}
$$

and then,

$$
\begin{aligned}
& p(Y \mid A=0, C) \\
& \quad = \frac{p(Y, A=0, C)}{p(A=0, C)} && (27) \\
& \quad = \frac{p(Y, A=0, C)}{\sum_Y p(Y, A=0, C)} && (28) \\
& \quad = \frac{\dfrac{(1-\epsilon_{c,y}) q_{c,y}(0) - \epsilon_{c,y} q_{c,y}(1)}{(1-\epsilon_{c,y}-\delta_{c,y})}}{\sum_{y'} \dfrac{(1-\epsilon_{c,y'}) q_{c,y'}(0) - \epsilon_{c,y'} q_{c,y'}(1)}{(1-\epsilon_{c,y'}-\delta_{c,y'})}} && (29)
\end{aligned}
$$

Plugging this distribution into $\tau_{\text{ME}} = E[Y(1)] - E[Y(0)]$ gives us the causal effect presented in Figure 4, Eq 3.

D Synthetic Data Distribution

In the distributions below, Ber(p) is used as the abbreviation a Bernoulli distribution with probability p.

Below, s_i, u_i and v_i are the effect of C, A, and Y on the probability of word T_i; each is drawn from $\mathcal{N}(0, \zeta)$, a parameter which controls how correlated words are with the underlying variables. When ζ is close to 0, the words are essentially random. When ζ is large, the words are essentially deterministic functions of the underlying variables. Similarly w_i is the effect of word T_i on R_A, and is drawn from $\mathcal{N}(0, \eta)$.

For both settings, we set vocabulary size to 4,334 (to match Yelp experiments) and ζ = 0.5. For the missing data setting, we set η = 0.1. We picked these constants by empirically finding a reasonable middle ground between the text data providing only noise and being a deterministic function of their parents. We picked all other constants such that the naïve correlation $p(Y \mid A)$ was a poor estimate of the counterfactual $p(Y(a))$ in the full-data setting.

D.1 Missing data data-generation

$$
\begin{aligned}
C &\sim \text{Ber}(0.4) \\
A(1) &\sim \text{Ber}(-0.3C + 0.4) \\
Y &\sim \text{Ber}(0.2C + 0.1A + 0.5) \\
T_i &\sim \text{Ber}(0.5 + u_i A + v_i C) \\
R_A &\sim \text{Ber}\left(0.7 + 0.2C - 0.4Y + \sum_i w_i T_i\right)
\end{aligned}
$$

D.2 Measurement error data-generation

$$
\begin{aligned}
C &\sim \text{Ber}(0.4) \\
A &\sim \text{Ber}(-0.3C + 0.4) \\
Y &\sim \text{Ber}(0.2C + 0.1A + 0.5) \\
T_i &\sim \text{Ber}(0.5 + s_i C + u_i A + v_i Y)
\end{aligned}
$$

Figure 6: Experimental results with a vocabulary of size 53,197. Squared distance (y-axis, lower is better) of the estimated causal effect from τ_{SC} calculated from the full data with no missing data or measurement error. Error bars (negligible for larger datasets) are 1.96 times standard error across 10 experiments.

E Additional Experiments

Figure 6 shows the results of a second set of experiments, which are identical to those described in §5 except the vocabulary size is now 53,197 instead of 4,334. For the Yelp data, the larger vocabulary consists of all words which appear at least ten times in a sample of 1M reviews. As the larger vocabulary introduced greater memory requirements, we did not run these experiments with as large of datasets.

The results of these experiments show roughly the same patterns as those seen in Figure 5. The `adjusted` measurement error models again appear erratic, generally performing worse than the `unadjusted` models though better than the `naive` models.

The `full` missing data model appeared to slightly outperform the `no_y` model on Yelp data but only perform as well on the synthetic data. Both these models appeared better than the `naive` and `no_text` models on both datasets.

Direct Output Connection for a High-Rank Language Model

Sho Takase† **Jun Suzuki**†‡ **Masaaki Nagata**†
†NTT Communication Science Laboratories
‡Tohoku University
{takase.sho, nagata.masaaki}@lab.ntt.co.jp
jun.suzuki@ecei.tohoku.ac.jp

Abstract

This paper proposes a state-of-the-art recurrent neural network (RNN) language model that combines probability distributions computed not only from a final RNN layer but also from middle layers. Our proposed method raises the expressive power of a language model based on the matrix factorization interpretation of language modeling introduced by Yang et al. (2018). The proposed method improves the current state-of-the-art language model and achieves the best score on the Penn Treebank and WikiText-2, which are the standard benchmark datasets. Moreover, we indicate our proposed method contributes to two application tasks: machine translation and headline generation. Our code is publicly available at: https://github.com/nttcslab-nlp/doc_lm.

1 Introduction

Neural network language models have played a central role in recent natural language processing (NLP) advances. For example, neural encoder-decoder models, which were successfully applied to various natural language generation tasks including machine translation (Sutskever et al., 2014), summarization (Rush et al., 2015), and dialogue (Wen et al., 2015), can be interpreted as conditional neural language models. Neural language models also positively influence syntactic parsing (Dyer et al., 2016; Choe and Charniak, 2016). Moreover, such word embedding methods as Skip-gram (Mikolov et al., 2013) and vLBL (Mnih and Kavukcuoglu, 2013) originated from neural language models designed to handle much larger vocabulary and data sizes. Neural language models can also be used as contextualized word representations (Peters et al., 2018). Thus, language modeling is a good benchmark task for investigating the general frameworks of neural methods in NLP field.

In language modeling, we compute joint probability using the product of conditional probabilities. Let $w_{1:T}$ be a word sequence with length T: $w_1, ..., w_T$. We obtain the joint probability of word sequence $w_{1:T}$ as follows:

$$p(w_{1:T}) = p(w_1) \prod_{t=1}^{T-1} p(w_{t+1}|w_{1:t}). \qquad (1)$$

$p(w_1)$ is generally assumed to be 1 in this literature, that is, $p(w_1) = 1$, and thus we can ignore its calculation. See the implementation of Zaremba et al. (2014)[1], for an example. RNN language models obtain conditional probability $p(w_{t+1}|w_{1:t})$ from the probability distribution of each word. To compute the probability distribution, RNN language models encode sequence $w_{1:t}$ into a fixed-length vector and apply a transformation matrix and the softmax function.

Previous researches demonstrated that RNN language models achieve high performance by using several regularizations and selecting appropriate hyperparameters (Melis et al., 2018; Merity et al., 2018). However, Yang et al. (2018) proved that existing RNN language models have low expressive power due to the *Softmax bottleneck*, which means the output matrix of RNN language models is low rank when we interpret the training of RNN language models as a matrix factorization problem. To solve the *Softmax bottleneck*, Yang et al. (2018) proposed *Mixture of Softmaxes* (MoS), which increases the rank of the matrix by combining multiple probability distributions computed from the encoded fixed-length vector.

In this study, we propose *Direct Output Connection* (DOC) as a generalization of MoS. For stacked RNNs, DOC computes the probability distributions from the middle layers including input embeddings. In addition to raising the rank, the

---

[1] https://github.com/wojzaremba/lstm

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4599–4609
Brussels, Belgium, October 31 - November 4, 2018.

proposed method helps weaken the vanishing gradient problem in backpropagation because DOC provides a shortcut connection to the output.

We conduct experiments on standard benchmark datasets for language modeling: the Penn Treebank and WikiText-2. Our experiments demonstrate that DOC outperforms MoS and achieves state-of-the-art perplexities on each dataset. Moreover, we investigate the effect of DOC on two applications: machine translation and headline generation. We indicate that DOC can improve the performance of an encoder-decoder with an attention mechanism, which is a strong baseline for such applications. In addition, we conduct an experiment on the Penn Treebank constituency parsing task to investigate the effectiveness of DOC.

2 RNN Language Model

In this section, we briefly overview RNN language models. Let V be the vocabulary size and let $P_t \in \mathbb{R}^V$ be the probability distribution of the vocabulary at timestep t. Moreover, let D_{h^n} be the dimension of the hidden state of the n-th RNN, and let D_e be the dimensions of the embedding vectors. Then the RNN language models predict probability distribution P_{t+1} by the following equation:

$$P_{t+1} = \mathrm{softmax}(W h_t^N), \quad (2)$$

$$h_t^n = f(h_t^{n-1}, h_{t-1}^n), \quad (3)$$

$$h_t^0 = E x_t, \quad (4)$$

where $W \in \mathbb{R}^{V \times D_{h^N}}$ is a weight matrix[2], $E \in \mathbb{R}^{D_e \times V}$ is a word embedding matrix, $x_t \in \{0,1\}^V$ is a one-hot vector of input word w_t at timestep t, and $h_t^n \in \mathbb{R}^{D_{h^n}}$ is the hidden state of the n-th RNN at timestep t. We define h_t^n at timestep $t = 0$ as a zero vector: $h_0^n = \mathbf{0}$. Let $f(\cdot)$ represent an abstract function of an RNN, which might be the Elman network (Elman, 1990), the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), the Recurrent Highway Network (RHN) (Zilly et al., 2017), or any other RNN variant. In this research, we stack three LSTM layers based on Merity et al. (2018) because they achieved high performance.

3 Language Modeling as Matrix Factorization

Yang et al. (2018) indicated that the training of language models can be interpreted as a matrix factorization problem. In this section, we briefly introduce their description. Let word sequence $w_{1:t}$ be context c_t. Then we can regard a natural language as a finite set of the pairs of a context and its conditional probability distribution: $\mathcal{L} = \{(c_1, P^*(X|c_1)), ..., (c_U, P^*(X|c_U))\}$, where U is the number of possible contexts and $X \in \{0,1\}^V$ is a variable representing a one-hot vector of a word. Here, we consider matrix $A \in \mathbb{R}^{U \times V}$ that represents the true log probability distributions and matrix $H \in \mathbb{R}^{U \times D_{h^N}}$ that contains the hidden states of the final RNN layer for each context c_t:

$$A = \begin{bmatrix} \log P^*(X|c_1) \\ \log P^*(X|c_2) \\ \dots \\ \log P^*(X|c_U) \end{bmatrix}; H = \begin{bmatrix} h_{c_1}^N \\ h_{c_2}^N \\ \dots \\ h_{c_U}^N \end{bmatrix}. \quad (5)$$

Then we obtain set of matrices $F(A) = \{A + \Lambda S\}$, where $S \in \mathbb{R}^{U \times V}$ is an all-ones matrix, and $\Lambda \in \mathbb{R}^{U \times U}$ is a diagonal matrix. $F(A)$ contains matrices that shifted each row of A by an arbitrary real number. In other words, if we take a matrix from $F(A)$ and apply the softmax function to each of its rows, we obtain a matrix that consists of true probability distributions. Therefore, for some $A' \in F(A)$, training RNN language models is to find the parameters satisfying the following equation:

$$HW^\top = A'. \quad (6)$$

Equation 6 indicates that training RNN language models can also be interpreted as a matrix factorization problem. In most cases, the rank of matrix $HW^\top$ is D_{h^N} because D_{h^N} is smaller than V and U in common RNN language models. Thus, an RNN language model cannot express true distributions if D_{h^N} is much smaller than $\mathrm{rank}(A')$.

Yang et al. (2018) also argued that $\mathrm{rank}(A')$ is as high as vocabulary size V based on the following two assumptions:

1. Natural language is highly context-dependent. In addition, since we can imagine many kinds of contexts, it is difficult to assume a basis that represents a conditional probability distribution for any contexts. In other words, compressing U is difficult.

2. Since we also have many kinds of semantic meanings, it is difficult to assume basic meanings that can create all other semantic meanings by such simple operations as addition and subtraction; compressing V is difficult.

---

[2] Actually, we apply a bias term in addition to the weight matrix but we omit it to simplify the following discussion.

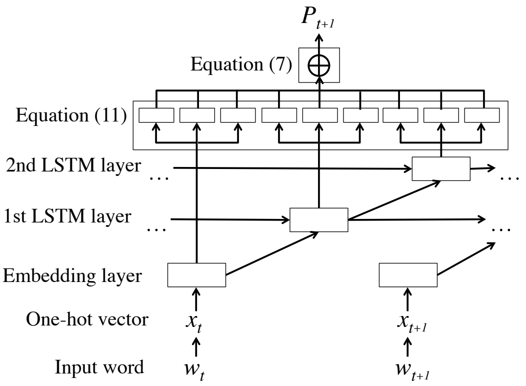

Figure 1: Overview of the proposed method: DOC. This figure represents the example of $N = 2$ and $i_0 = i_1 = i_2 = 3$.

In summary, Yang et al. (2018) indicated that D_{h^N} is much smaller than $\mathrm{rank}(A)$ because its scale is usually 10^2 and vocabulary size V is at least 10^4.

4 Proposed Method: Direct Output Connection

To construct a high-rank matrix, Yang et al. (2018) proposed Mixture of Softmaxes (MoS). MoS computes multiple probability distributions from the hidden state of final RNN layer h^N and regards the weighted average of the probability distributions as the final distribution. In this study, we propose Direct Output Connection (DOC), which is a generalization method of MoS. DOC computes probability distributions from the middle layers in addition to the final layer. In other words, DOC directly connects the middle layers to the output.

Figure 1 shows an overview of DOC, that uses the middle layers (including word embeddings) to compute the probability distributions. Figure 1 computes three probability distributions from all the layers, but we can vary the number of probability distributions for each layer and select some layers to avoid. In our experiments, we search for the appropriate number of probability distributions for each layer.

Formally, instead of Equation 2, DOC computes the output probability distribution at timestep $t+1$ by the following equation:

$$P_{t+1} = \sum_{j=1}^{J} \pi_{j,c_t} \, \mathrm{softmax}(\tilde{W} k_{j,c_t}), \qquad (7)$$

$$s.t. \sum_{j=1}^{J} \pi_{j,c_t} = 1, \qquad (8)$$

where π_{j,c_t} is a weight for each probability distribution, $k_{j,c_t} \in \mathbb{R}^d$ is a vector computed from each hidden state h^n, and $\tilde{W} \in \mathbb{R}^{V \times d}$ is a weight matrix. Thus, P_{t+1} is the weighted average of J probability distributions. We define the $U \times U$ diagonal matrix whose elements are weight $\pi_{j,c}$ for each context c as Φ. Then we obtain matrix $\tilde{A} \in \mathbb{R}^{U \times V}$:

$$\tilde{A} = \log \sum_{j=1}^{J} \Phi \, \mathrm{softmax}(K_j \tilde{W}^\top), \qquad (9)$$

where $K_j \in \mathbb{R}^{U \times d}$ is a matrix whose rows are vector $k_{j,c}$. $\tilde{A}$ can be an arbitrary high rank because the righthand side of Equation 9 computes not only the matrix multiplication but also a nonlinear function. Therefore, an RNN language model with DOC can output a distribution matrix whose rank is identical to one of the true distributions. In other words, $\tilde{A}$ is a better approximation of A' than the output of a standard RNN language model.

Next we describe how to acquire weight π_{j,c_t} and vector k_{j,c_t}. Let $\pi_{c_t} \in \mathbb{R}^J$ be a vector whose elements are weight π_{j,c_t}. Then we compute π_{c_t} from the hidden state of the final RNN layer:

$$\pi_{c_t} = \mathrm{softmax}(W_\pi h_t^N), \qquad (10)$$

where $W_\pi \in \mathbb{R}^{J \times D_{h^N}}$ is a weight matrix. We next compute k_{j,c_t} from the hidden state of the n-th RNN layer:

$$k_{j,c_t} = W_j h_t^n, \qquad (11)$$

where $W_j \in \mathbb{R}^{d \times D_{h^n}}$ is a weight matrix. In addition, let i_n be the number of k_{j,c_t} from h_t^n. Then we define the sum of i_n for all n as J; that is, $\sum_{n=0}^{N} i_n = J$. In short, DOC computes J probability distributions from all the layers, including the input embedding (h^0). For $i_N = J$, DOC becomes identical to MoS. In addition to increasing the rank, we expect that DOC weakens the vanishing gradient problem during backpropagation because a middle layer is directly connected to the output, such as with the auxiliary classifiers described in Szegedy et al. (2015).

For a network that computes the weights for several vectors, such as Equation 10, Shazeer et al. (2017) indicated that it often converges to a state where it always produces large weights for few vectors. In fact, we observed that DOC tends to assign large weights to shallow layers. To prevent this phenomenon, we compute the coefficient of variation of Equation 10 in each mini-batch as a regularization term following Shazeer et al. (2017). In other words, we try to adjust the sum of the weights for each probability distribution with identical values in each mini-batch. Formally, we compute the following equation for a mini-batch consisting of $w_b, w_{b+1}, ..., w_{\tilde{b}}$:

$$B = \sum_{t=b}^{\tilde{b}} \pi_{c_t} \tag{12}$$

$$\beta = \left(\frac{\mathrm{std}(B)}{\mathrm{avg}(B)}\right)^2, \tag{13}$$

where functions $\mathrm{std}(\cdot)$ and $\mathrm{avg}(\cdot)$ are functions that respectively return an input's standard deviation and its average. In the training step, we add λ_β multiplied by weight coefficient β to the loss function.

5 Experiments on Language Modeling

We investigate the effect of DOC on the language modeling task. In detail, we conduct word-level prediction experiments and show that DOC improves the performance of MoS, which only uses the final layer to compute the probability distributions. Moreover, we evaluate various combinations of layers to explore which combination achieves the best score.

5.1 Datasets

We used the Penn Treebank (PTB) (Marcus et al., 1993) and WikiText-2 (Merity et al., 2017) datasets, which are the standard benchmark datasets for the word-level language modeling task. Mikolov et al. (2010) and Merity et al. (2017) respectively published preprocessed PTB[3] and WikiText-2[4] datasets. Table 1 describes their statistics. We used these preprocessed datasets for fair comparisons with previous studies.

5.2 Hyperparameters

Our implementation is based on the averaged stochastic gradient descent Weight-Dropped LSTM (AWD-LSTM)[5] proposed by Merity et al. (2018). AWD-LSTM consists of three LSTMs with various regularizations. For the hyperparameters, we used the same values as Yang et al. (2018) except for the dropout rate for vector k_{j,c_t} and the non-monotone interval. Since we found that the dropout rate for vector k_{j,c_t} greatly influences β in Equation 13, we varied it from 0.3 to 0.6 with 0.1 intervals. We selected 0.6 because this value achieved the best score on the PTB validation dataset. For the non-monotone interval, we adopted the same value as Zolna et al. (2018). Table 2 summarizes the hyperparameters of our experiments.

		PTB	WikiText-2
Vocab		10,000	33,278
#Token	Train	929,590	2,088,628
	Valid	73,761	217,646
	Test	82,431	245,569

Table 1: Statistics of PTB and WikiText-2.

Hyperparameter	PTB	WikiText-2
Learning rate	20	15
Batch size	12	15
Non-monotone interval	60	60
D_e	280	300
D_{h^1}	960	1150
D_{h^2}	960	1150
D_{h^3}	620	650
Dropout rate for x_t	0.1	0.1
Dropout rate for h_t^0	0.4	0.65
Dropout rate for h_t^1, h_t^2	0.225	0.2
Dropout rate for h_t^3	0.4	0.4
Dropout rate for k_{j,c_t}	0.6	0.6
Recurrent weight dropout	0.50	0.50

Table 2: Hyperparameters used for training DOC.

#DOC						
i_3	i_2	i_1	i_0	λ_β	Valid	Test
15	0	0	0	0	56.54†	54.44†
20	0	0	0	0	56.88‡	54.79‡
15	0	0	5	0	56.21	54.28
15	0	5	0	0	55.26	53.52
15	5	0	0	0	54.87	53.15
15	5	0	0	0.0001	54.95	53.16
15	5	0	0	0.001	**54.62**	**52.87**
15	5	0	0	0.01	55.13	53.39
10	5	0	5	0	56.46	54.18
10	5	5	0	0	56.00	54.37

Table 3: Perplexities of AWD-LSTM with DOC on the PTB dataset. We varied the number of probability distributions from each layer in situation $J = 20$ except for the top row. The top row (†) represents MoS scores reported in Yang et al. (2018) as a baseline. ‡ represents the perplexity obtained by the implementation of Yang et al. (2018)[6] with identical hyperparameters except for i_3.

5.3 Results

Table 3 shows the perplexities of AWD-LSTM with DOC on the PTB dataset. Each value of columns i_n

---

[3] http://www.fit.vutbr.cz/ imikolov/rnnlm/

[4] https://einstein.ai/research/the-wikitext-long-term-dependency-language-modeling-dataset

[5] https://github.com/salesforce/awd-lstm-lm

[6] https://github.com/zihangdai/mos

λ_β	Valid	Test
0	0.276	0.279
0.0001	0.254	0.252
0.001	0.217	0.213
0.01	0.092	0.086

Table 4: Coefficient of variation of Equation 10: $\sqrt{\beta}$ in validation and test sets of PTB.

Model	Valid	Test
AWD-LSTM	401	401
AWD-LSTM-MoS	10000	10000
AWD-LSTM-DOC	10000	10000

Table 5: Rank of output matrix ($\tilde{A}$ in Equation 9) on the PTB dataset. D_3 of AWD-LSTM is 400.

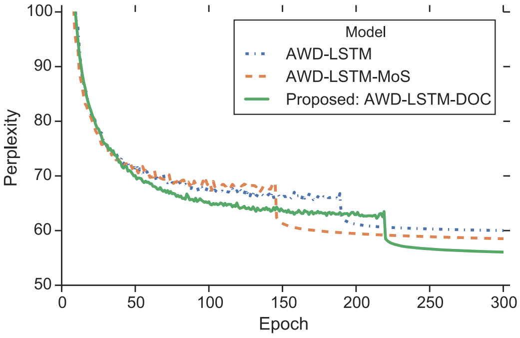

Figure 2: Perplexities of each method on the PTB validation set.

represents the number of probability distributions from hidden state h_t^n. To find the best combination, we varied the number of probability distributions from each layer by fixing their total to 20: $J = 20$. Moreover, the top row of Table 3 shows the perplexity of AWD-LSTM with MoS reported in Yang et al. (2018) for comparison. Table 3 indicates that language models using middle layers outperformed one using only the final layer. In addition, Table 3 shows that increasing the distributions from the final layer ($i_3 = 20$) degraded the score from the language model with $i_3 = 15$ (the top row of Table 3). Thus, to obtain a superior language model, we should not increase the number of distributions from the final layer; we should instead use the middle layers, as with our proposed DOC.

Table 3 shows that the $i_3 = 15, i_2 = 5$ setting achieved the best performance and the other settings with shallow layers have a little effect. This result implies that we need some layers to output accurate distributions. In fact, most previous studies adopted two LSTM layers for language modeling. This suggests that we need at least two layers to obtain high-quality distributions.

Model	Valid	Test
AWD-LSTM†	58.88	56.36
AWD-LSTM-MoS†	56.36	54.26
AWD-LSTM-MoS‡	55.67	53.75
AWD-LSTM-DOC	**54.62**	**52.87**
AWD-LSTM-DOC (fin)	**54.12**	**52.38**

Table 6: Perplexities of our implementations and reruns on the PTB dataset. We set the non-monotone interval to 60. † represents results obtained by original implementations with identical hyperparameters except for non-monotone interval. ‡ indicates the result obtained by our AWD-LSTM-MoS implementation with identical dropout rates as AWD-LSTM-DOC. For (fin), we repeated fine-tuning until convergence.

For the $i_3 = 15, i_2 = 5$ setting, we explored the effect of λ_β in $\{0, 0.01, 0.001, 0.0001\}$. Although Table 3 shows that $\lambda_\beta = 0.001$ achieved the best perplexity, the effect is not consistent. Table 4 shows the coefficient of variation of Equation 10, i.e., $\sqrt{\beta}$ in the PTB dataset. This table demonstrates that the coefficient of variation decreases with growth in λ_β. In other words, the model trained with a large λ_β assigns balanced weights to each probability distribution. These results indicate that it is not always necessary to equally use each probability distribution, but we can acquire a better model in some λ_β. Hereafter, we refer to the setting that achieved the best score ($i_3 = 15, i_2 = 5, \lambda_\beta = 0.001$) as AWD-LSTM-DOC.

Table 5 shows the ranks of matrices containing log probability distributions from each method. In other words, Table 5 describes $\tilde{A}$ in Equation 9 for each method. As shown by this table, the output of AWD-LSTM is restricted to D_3[7]. In contrast, AWD-LSTM-MoS (Yang et al., 2018) and AWD-LSTM-DOC outputted matrices whose ranks equal the vocabulary size. This fact indicates that DOC (including MoS) can output the same matrix as the true distributions in view of a rank.

Figure 2 illustrates the learning curves of each method on PTB. This figure contains the validation scores of AWD-LSTM, AWD-LSTM-MoS, and AWD-LSTM-DOC at each training epoch. We trained AWD-LSTM and AWD-LSTM-MoS by setting the non-monotone interval to 60, as with AWD-LSTM-DOC. In other words, we used hyperparameters identical to the original ones to train AWD-LSTM and AWD-LSTM-MoS, except for the non-monotone interval. We note that the opti-

---

[7] Actually, the maximum rank size of an ordinary RNN language model is $D_N + 1$ when we use a bias term.

Model	#Param	Valid	Test
LSTM (medium) (Zaremba et al., 2014)	20M	86.2	82.7
LSTM (large) (Zaremba et al., 2014)	66M	82.2	78.4
Variational LSTM (medium) (Gal and Ghahramani, 2016)	20M	81.9 ± 0.2	79.7 ± 0.1
Variational LSTM (large) (Gal and Ghahramani, 2016)	66M	77.9 ± 0.3	75.2 ± 0.2
Variational RHN (Zilly et al., 2017)	32M	71.2	68.5
Variational RHN + WT (Zilly et al., 2017)	23M	67.9	65.4
Variational RHN + WT + IOG (Takase et al., 2017)	29M	67.0	64.4
Neural Architecture Search (Zoph and Le, 2017)	54M	-	62.4
LSTM with skip connections (Melis et al., 2018)	24M	60.9	58.3
AWD-LSTM (Merity et al., 2018)	24M	60.0	57.3
AWD-LSTM + Fraternal Dropout (Zolna et al., 2018)	24M	58.9	56.8
AWD-LSTM-MoS (Yang et al., 2018)	22M	56.54	54.44
Proposed method: AWD-LSTM-DOC	23M	**54.62**	**52.87**
Proposed method: AWD-LSTM-DOC (fin)	23M	**54.12**	**52.38**
Proposed method (ensemble): AWD-LSTM-DOC × 5	114M	**49.99**	**48.44**
Proposed method (ensemble): AWD-LSTM-DOC (fin) × 5	114M	**48.63**	**47.17**

Table 7: Perplexities of each method on the PTB dataset.

Model	#Param	Valid	Test
Variational LSTM + IOG (Takase et al., 2017)	70M	95.9	91.0
Variational LSTM + WT + AL (Inan et al., 2017)	28M	91.5	87.0
LSTM with skip connections (Melis et al., 2018)	24M	69.1	65.9
AWD-LSTM (Merity et al., 2018)	33M	68.6	65.8
AWD-LSTM + Fraternal Dropout (Zolna et al., 2018)	34M	66.8	64.1
AWD-LSTM-MoS (Yang et al., 2018)	35M	63.88	61.45
Proposed method: AWD-LSTM-DOC	37M	**60.97**	**58.55**
Proposed method: AWD-LSTM-DOC (fin)	37M	**60.29**	**58.03**
Proposed method (ensemble): AWD-LSTM-DOC × 5	185M	**56.14**	**54.23**
Proposed method (ensemble): AWD-LSTM-DOC (fin) × 5	185M	**54.91**	**53.09**

Table 8: Perplexities of each method on the WikiText-2 dataset.

mization method converts the ordinary stochastic gradient descent (SGD) into the averaged SGD at the point where convergence almost occurs. In Figure 2, the turning point is the epoch when each method drastically decreases the perplexity. Figure 2 shows that each method similarly reduces the perplexity at the beginning. AWD-LSTM and AWD-LSTM-MoS were slow to decrease the perplexity from 50 epochs. In contrast, AWD-LSTM-DOC constantly decreased the perplexity and achieved a lower value than the other methods with ordinary SGD. Therefore, we conclude that DOC positively affects the training of language modeling.

Table 6 shows the AWD-LSTM, AWD-LSTM-MoS, and AWD-LSTM-DOC results in our configurations. For AWD-LSTM-MoS, we trained our implementation with the same dropout rates as AWD-LSTM-DOC for a fair comparison. AWD-LSTM-DOC outperformed both the original AWD-LSTM-MoS and our implementation. In other words, DOC outperformed MoS.

Since the averaged SGD uses the averaged parameters from each update step, the parameters of the early steps are harmful to the final parameters. Therefore, when the model converges, recent studies and ours eliminate the history of and then retrains the model. Merity et al. (2018) referred to this retraining process as fine-tuning. Although most previous studies only conducted fine-tuning once, Zolna et al. (2018) argued that two fine-tunings provided additional improvement. Thus, we repeated fine-tuning until we achieved no more improvements in the validation data. We refer to the model as AWD-LSTM-DOC (fin) in Table 6, which shows that repeated fine-tunings improved the perplexity by about 0.5.

Tables 7 and 8 respectively show the perplexities of AWD-LSTM-DOC and previous studies on PTB and WikiText-2[8]. These tables show that AWD-LSTM-DOC achieved the best perplexity. AWD-LSTM-DOC improved the perplexity by almost 2.0 on PTB and 3.5 on WikiText-2 from the state-of-the-art scores. The ensemble technique provided further improvement, as described in pre-

---

[8] We exclude models that use the statistics of the test data (Grave et al., 2017; Krause et al., 2017) from these tables because we regard neural language models as the basis of NLP applications and consider it unreasonable to know correct outputs during applications, e.g., machine translation. In other words, we focus on neural language models as the foundation of applications although we can combine the method using the statistics of test data with our AWD-LSTM-DOC.

vious studies (Zaremba et al., 2014; Takase et al., 2017), and improved the perplexity by at least 4 points on both datasets. Finally, the ensemble of the repeated finetuning models achieved 47.17 on the PTB test and 53.09 on the WikiText-2 test.

6 Experiments on Application Tasks

As described in Section 1, a neural encoder-decoder model can be interpreted as a conditional language model. To investigate the effect of DOC on an encoder-decoder model, we incorporate DOC into the decoder and examine its performance.

6.1 Dataset

We conducted experiments on machine translation and headline generation tasks. For machine translation, we used two kinds of sentence pairs (English-German and English-French) in the IWSLT 2016 dataset[9]. The training set respectively contains about 189K and 208K sentence pairs of English-German and English-French. We experimented in four settings: from English to German (En-De), its reverse (De-En), from English to French (En-Fr), and its reverse (Fr-En).

Headline generation is a task that creates a short summarization of an input sentence(Rush et al., 2015). Rush et al. (2015) constructed a headline generation dataset by extracting pairs of first sentences of news articles and their headlines from the annotated English Gigaword corpus (Napoles et al., 2012). They also divided the extracted sentence-headline pairs into three parts: training, validation, and test sets. The training set contains about 3.8M sentence-headline pairs. For our evaluation, we used the test set constructed by Zhou et al. (2017) because the one constructed by Rush et al. (2015) contains some invalid instances, as reported in Zhou et al. (2017).

6.2 Encoder-Decoder Model

For the base model, we adopted an encoder-decoder with an attention mechanism described in Kiyono et al. (2017). The encoder consists of a 2-layer bidirectional LSTM, and the decoder consists of a 2-layer LSTM with attention proposed by Luong et al. (2015). We interpreted the layer after computing the attention as the 3rd layer of the decoder. We refer to this encoder-decoder as EncDec. For the hyperparameters, we followed the setting of Kiyono et al. (2017) except for the sizes of hidden states and embeddings. We used 500 for machine translation and 400 for headline generation. We constructed a vocabulary set by using Byte-Pair-Encoding[10] (BPE) (Sennrich et al., 2016). We set the number of BPE merge operations at 16K for the machine translation and 5K for the headline generation.

Model	En-De	De-En	En-Fr	Fr-En
EncDec	23.05	28.18	34.37	34.07
EncDec+DOC ($i_3 = 2$)	23.62	29.12	36.09	34.41
EncDec+DOC ($i_3 = i_2 = 2$)	**23.97**	**29.33**	**36.11**	**34.72**

Table 9: BLEU scores on test sets in the IWSLT 2016 dataset. We report averages of three runs.

Model	RG-1	RG-2	RG-L
EncDec	46.77	24.87	43.58
EncDec+DOC ($i_3 = 2$)	46.91	24.91	43.73
EncDec+DOC ($i_3 = i_2 = 2$)	**46.99**	**25.29**	**43.83**
ABS (Rush et al., 2015)	37.41	15.87	34.70
SEASS (Zhou et al., 2017)	46.86	24.58	43.53
Kiyono et al. (2017)	46.34	24.85	43.49

Table 10: ROUGE F1 scores in headline generation test data provided by Zhou et al. (2017). RG in table denotes ROUGE. For our implementations (the upper part), we report averages of three runs.

In this experiment, we compare DOC to the base EncDec. We prepared two DOC settings: using only the final layer, that is, a setting that is identical to MoS, and using both the final and middle layers. We used the 2nd and 3rd layers in the latter setting because this case achieved the best performance on the language modeling task in Section 5.3. We set $i_3 = 2$ and $i_2 = 2, i_3 = 2$. For this experiment, we modified a publicly available encode-decoder implementation[11].

6.3 Results

Table 9 shows the BLEU scores of each method. Since an initial value often drastically varies the result of a neural encoder-decoder, we reported the average of three models trained from different initial values and random seeds. Table 9 indicates that EncDec+DOC outperformed EncDec.

Table 10 shows the ROUGE F1 scores of each method. In addition to the results of our implementations (the upper part), the lower part represents the published scores reported in previous studies. For the upper part, we reported the average of three models (as in Table 9). EncDec+DOC outperformed EncDec on all scores. Moreover, EncDec

---

[9] https://wit3.fbk.eu/

[10] https://github.com/rsennrich/subword-nmt

[11] https://github.com/mlpnlp/mlpnlp-nmt/

outperformed the state-of-the-art method (Zhou et al., 2017) on the ROUGE-2 and ROUGE-L F1 scores. In other words, our baseline is already very strong. We believe that this is because we adopted a larger embedding size than Zhou et al. (2017). It is noteworthy that DOC improved the performance of EncDec even though EncDec is very strong.

These results indicate that DOC positively influences a neural encoder-decoder model. Using the middle layer also yields further improvement because EncDec+DOC ($i_3 = i_2 = 2$) outperformed EncDec+DOC ($i_3 = 2$).

7 Experiments on Constituency Parsing

Choe and Charniak (2016) achieved high F1 scores on the Penn Treebank constituency parsing task by transforming candidate trees into a symbol sequence (S-expression) and reranking them based on the perplexity obtained by a neural language model. To investigate the effectiveness of DOC, we evaluate our language models following their configurations.

7.1 Dataset

We used the Wall Street Journal of the Penn Treebank dataset. We used the section 2-21 for training, 22 for validation, and 23 for testing. We applied the preprocessing codes of Choe and Charniak (2016)[12] to the dataset and converted a token that appears fewer than ten times in the training dataset into a special token *unk*. For reranking, we prepared 500 candidates obtained by the Charniak parser (Charniak, 2000).

7.2 Models

We compare AWD-LSTM-DOC with AWD-LSTM (Merity et al., 2018) and AWD-LSTM-MoS (Yang et al., 2018). We trained each model with the same hyperparameters from our language modeling experiments (Section 5). We selected the model that achieved the best perplexity on the validation set during the training.

7.3 Results

Model	F1 Base	F1 Rerank
Reranking with single model		
Choe and Charniak (2016)	89.7	92.6
AWD-LSTM	89.7	93.2
AWD-LSTM-MoS	89.7	93.2
AWD-LSTM-DOC	89.7	**93.3**
Reranking with model ensemble		
AWD-LSTM × 5 (ensemble)	89.7	93.4
AWD-LSTM-MoS × 5 (ensemble)	89.7	93.4
AWD-LSTM-DOC × 5 (ensemble)	89.7	**93.5**
AWD-LSTM-DOC × 5 (ensemble)	91.2	**94.29**
AWD-LSTM-DOC × 5 (ensemble)	93.12	**94.47**
State-of-the-art results		
Dyer et al. (2016)	91.7	93.3
Fried et al. (2017) (ensemble)	92.72	94.25
Suzuki et al. (2018) (ensemble)	92.74	94.32
Kitaev and Klein (2018)	95.13	-

Table 11: Bracketing F1 scores on the PTB test set (Section 23). This table includes reranking models trained on the PTB without external data.

Table 11 shows the bracketing F1 scores on the PTB test set. This table is divided into three parts by horizontal lines; the upper part describes the scores by single language modeling based rerankers, the middle part shows the results by ensembling five rerankers, and the lower part represents the current state-of-the-art scores in the setting without external data. The upper part also contains the score reported in Choe and Charniak (2016) that reranked candidates by the simple LSTM language model. This part indicates that our implemented rerankers outperformed the simple LSTM language model based reranker, which achieved 92.6 F1 score (Choe and Charniak, 2016). Moreover, AWD-LSTM-DOC outperformed AWD-LSTM and AWD-LSTM-MoS. These results correspond to the language modeling task.

The middle part shows that AWD-LSTM-DOC also outperformed AWD-LSTM and AWD-LSTM-MoS in the ensemble setting. In addition, we can improve the performance by exchanging the base parser with a stronger one. In fact, we achieved 94.29 F1 score by reranking the candidates from retrained Recurrent Neural Network Grammars (RNNG) (Dyer et al., 2016)[13], that achieved 91.2 F1 score in our configuration. Moreover, the lowest row of the middle part indicates the result by reranking the candidates from the retrained neural encoder-decoder based parser (Suzuki et al., 2018). Our base parser has two different parts from Suzuki et al. (2018). First, we used the sum of the hidden states of the forward and backward RNNs as the hidden layer for each RNN[14]. Second, we tied the embedding matrix to the weight matrix to compute

---

[12] https://github.com/cdg720/emnlp2016

[13] The output of RNNG is not in descending order because it samples candidates based on their scores. Thus, we prepared more candidates (i.e., 700) to be able to obtain correct instances as candidates.

[14] We used the deep bidirectional encoder described at http://opennmt.net/OpenNMT/training/models/ instead of a basic bidirectional encoder.

the probability distributions in the decoder. The retrained parser achieved 93.12 F1 score. Finally, we achieved 94.47 F1 score by reranking its candidates with AWD-LSTM-DOC. We expect that we can achieve even better score by replacing the base parser with the current state-of-the-art one (Kitaev and Klein, 2018).

8 Related Work

Bengio et al. (2003) are pioneers of neural language models. To address the curse of dimensionality in language modeling, they proposed a method using word embeddings and a feed-forward neural network (FFNN). They demonstrated that their approach outperformed n-gram language models, but FFNN can only handle fixed-length contexts. Instead of FFNN, Mikolov et al. (2010) applied RNN (Elman, 1990) to language modeling to address the entire given sequence as a context. Their method outperformed the Kneser-Ney smoothed 5-gram language model (Kneser and Ney, 1995; Chen and Goodman, 1996).

Researchers continue to try to improve the performance of RNN language models. Zaremba et al. (2014) used LSTM (Hochreiter and Schmidhuber, 1997) instead of a simple RNN for language modeling and significantly improved an RNN language model by applying dropout (Srivastava et al., 2014) to all the connections except for the recurrent connections. To regularize the recurrent connections, Gal and Ghahramani (2016) proposed variational inference-based dropout. Their method uses the same dropout mask at each timestep. Zolna et al. (2018) proposed fraternal dropout, which minimizes the differences between outputs from different dropout masks to be invariant to the dropout mask. Melis et al. (2018) used black-box optimization to find appropriate hyperparameters for RNN language models and demonstrated that the standard LSTM with proper regularizations can outperform other architectures.

Apart from dropout techniques, Inan et al. (2017) and Press and Wolf (2017) proposed the word tying method (WT), which unifies word embeddings (E in Equation 4) with the weight matrix to compute probability distributions (W in Equation 2). In addition to quantitative evaluation, Inan et al. (2017) provided a theoretical justification for WT and proposed the augmented loss technique (AL), which computes an objective probability based on word embeddings. In addition to these regularization techniques, Merity et al. (2018) used DropConnect (Wan et al., 2013) and averaged SGD (Polyak and Juditsky, 1992) for an LSTM language model. Their AWD-LSTM achieved lower perplexity than Melis et al. (2018) on PTB and WikiText-2.

Previous studies also explored superior architecture for language modeling. Zilly et al. (2017) proposed recurrent highway networks that use highway layers (Srivastava et al., 2015) to deepen recurrent connections. Zoph and Le (2017) adopted reinforcement learning to construct the best RNN structure. However, as mentioned, Melis et al. (2018) established that the standard LSTM is superior to these architectures. Apart from RNN architecture, Takase et al. (2017) proposed the input-to-output gate (IOG), which boosts the performance of trained language models.

As described in Section 3, Yang et al. (2018) interpreted training language modeling as matrix factorization and improved performance by computing multiple probability distributions. In this study, we generalized their approach to use the middle layers of RNNs. Finally, our proposed method, DOC, achieved the state-of-the-art score on the standard benchmark datasets.

Some studies provided methods that boost performance by using statistics obtained from test data. Grave et al. (2017) extended a cache model (Kuhn and De Mori, 1990) for RNN language models. Krause et al. (2017) proposed dynamic evaluation that updates parameters based on a recent sequence during testing. Although these methods might also improve the performance of DOC, we omitted such investigation to focus on comparisons among methods trained only on the training set.

9 Conclusion

We proposed *Direct Output Connection* (DOC), a generalization method of MoS introduced by Yang et al. (2018). DOC raises the expressive power of RNN language models and improves quality of the model. DOC outperformed MoS and achieved the best perplexities on the standard benchmark datasets of language modeling: PTB and WikiText-2. Moreover, we investigated its effectiveness on machine translation and headline generation. Our results show that DOC also improved the performance of EncDec and using a middle layer positively affected such application tasks.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2000)*, pages 132–139.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics (ACL 1996)*, pages 310–318.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 2331–2336.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 199–209.

Jeffrey L Elman. 1990. Finding Structure in Time. *Cognitive science*, 14(2):179–211.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 161–166.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving Neural Language Models with a Continuous Cache. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 2676–2686.

Shun Kiyono, Sho Takase, Jun Suzuki, Naoaki Okazaki, Kentaro Inui, and Masaaki Nagata. 2017. Source-side prediction for neural headline generation. *CoRR*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 1995)*, pages 181–184.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2017. Dynamic evaluation of neural sequence models. *CoRR*.

Roland Kuhn and Renato De Mori. 1990. A cache-based natural language model for speech recognition. 12:570–583.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1412–1421.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018).*

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018).*

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017).*

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, pages 1045–1048.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning Word Embeddings Efficiently with Noise-Contrastive Estimation. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 2265–2273.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2018)*, pages 2227–2237.

Boris T Polyak and Anatoli B Juditsky. 1992. Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 157–163.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 86–96.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. In *Proceedings of the Deep Learning Workshop in ICML 15*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112.

Jun Suzuki, Sho Takase, Hidetaka Kamigaito, Makoto Morishita, and Masaaki Nagata. 2018. An empirical study of building a strong baseline for constituency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 612–618.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*, pages 1–9.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2017. Input-to-output gate to improve rnn language models. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP 2017)*, pages 43–48.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. 2013. Regularization of Neural Networks using DropConnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, pages 1058–1066.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1711–1721.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014)*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1095–1104.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2017. Recurrent Highway Networks. *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, pages 4189–4198.

Konrad Zolna, Devansh Arpit, Dendi Suhubdy, and Yoshua Bengio. 2018. Fraternal dropout. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Disfluency Detection using Auto-Correlational Neural Networks

Paria Jamshid Lou
Macquarie University
Sydney, Australia
`paria.jamshid-lou@hdr.mq.edu.au`

Peter Anderson
Macquarie University
Sydney, Australia
`p.anderson@mq.edu.au`

Mark Johnson
Macquarie University
Sydney, Australia
`mark.johnson@mq.edu.au`

Abstract

In recent years, the natural language processing community has moved away from task-specific feature engineering, i.e., researchers discovering ad-hoc feature representations for various tasks, in favor of general-purpose methods that learn the input representation by themselves. However, state-of-the-art approaches to disfluency detection in spontaneous speech transcripts currently still depend on an array of hand-crafted features, and other representations derived from the output of pre-existing systems such as language models or dependency parsers. As an alternative, this paper proposes a simple yet effective model for automatic disfluency detection, called an auto-correlational neural network (ACNN). The model uses a convolutional neural network (CNN) and augments it with a new auto-correlation operator at the lowest layer that can capture the kinds of "rough copy" dependencies that are characteristic of repair disfluencies in speech. In experiments, the ACNN model outperforms the baseline CNN on a disfluency detection task with a 5% increase in f-score, which is close to the previous best result on this task.

1 Introduction

Disfluency informally refers to any interruptions in the normal flow of speech, including false starts, corrections, repetitions and filled pauses. Shriberg (1994) defines three distinct parts of a speech disfluency, referred to as the *reparandum*, *interregnum* and *repair*. As illustrated in Example 1, the reparandum *to Boston* is the part of the utterance that is replaced, the interregnum *uh, I mean* (which consists of a filled pause *uh* and a discouse marker *I mean*) is an optional part of a disfluent structure, and the repair *to Denver* replaces the reparandum. The fluent version is obtained by removing reparandum and interregnum words although disfluency detection models mainly deal with identifying and removing reparanda. The reason is that filled pauses and discourse markers belong to a closed set of words and phrases and are trivial to detect (Johnson and Charniak, 2004).

$$\begin{gathered}\textit{I want a flight } \overbrace{\textit{to Boston,}}^{\text{reparandum}} \\ \underbrace{\textit{uh, I mean}}_{\text{interregnum}} \underbrace{\textit{to Denver}}_{\text{repair}} \textit{ on Friday}\end{gathered} \tag{1}$$

In disfluent structures, the repair (e.g., *to Denver*) frequently seems to be a "rough copy" of the reparandum (e.g., *to Boston*). In other words, they incorporate the same or very similar words in roughly the same word order. In the Switchboard training set (Godfrey and Holliman, 1993), over 60% of the words in the reparandum are exact copies of words in the repair. Thus, this similarity is strong evidence of a disfluency that can help the model detect reparanda (Charniak and Johnson, 2001; Johnson and Charniak, 2004). As a result, models which are able to detect "rough copies" are likely to perform well on this task.

Currently, state-of-the-art approaches to disfluency detection depend heavily on hand-crafted pattern match features, specifically designed to find such "rough copies" (Zayats et al., 2016; Jamshid Lou and Johnson, 2017). In contrast to many other sequence tagging tasks (Plank et al., 2016; Yu et al., 2017), "vanilla" convolutional neural networks (CNNs) and long short-term memory (LSTM) models operating only on words or characters are surprisingly poor at disfluency detection (Zayats et al., 2016). As such, the task of disfluency detection sits in opposition to the ongoing trend in NLP away from task-specific feature engineering — i.e., researchers discovering ad-hoc feature representations for various tasks — in favor of general-purpose methods that

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4610–4619
Brussels, Belgium, October 31 - November 4, 2018.

learn the input representation by themselves (Collobert and Weston, 2008).

In this paper, we hypothesize that LSTMs and CNNs cannot not easily learn "rough copy" dependencies. We address this problem in the context of a CNN by introducing a novel auto-correlation operator. The resulting model, called an *auto-correlational neural network (ACNN)*, is a generalization of a CNN with an auto-correlation operator at the lowest layer. Evaluating the ACNN in the context of disfluency detection, we show that introducing the auto-correlation operator increases f-score by 5% over a baseline CNN. Furthermore, the ACNN — operating only on word inputs — achieves results which are competitive with much more complex approaches relying on hand-crafted features and outputs from pre-existing systems such as language models or dependency parsers. In summary, the main contributions of this paper are:

- We introduce the auto-correlational neural network (ACNN), a generalization of a CNN incorporating auto-correlation operations,
- In the context of disfluency detection, we show that the ACNN captures important properties of speech repairs including "rough copy" dependencies, and
- Using the ACNN, we achieve competitive results for disfluency detection without relying on any hand-crafted features or other representations derived from the output of pre-existing systems.

2 Related Work

Approaches to disfluency detection task fall into three main categories: noisy channel models, parsing-based approaches and sequence tagging approaches. Noisy channel models (NCMs) (Johnson and Charniak, 2004; Johnson et al., 2004) use complex tree adjoining grammar (TAG) (Shieber and Schabes, 1990) based channel models to find the "rough copy" dependencies between words. The channel model uses the similarity between the reparandum and the repair to allocate higher probabilities to exact copy reparandum words. Using the probabilities of TAG channel model and a bigram language model (LM) derived from training data, the NCM generates n-best disfluency analyses for each sentence at test time. The analyses are then reranked using a language model which is sensitive to the global properties of the sentence, such as a syntactic parser based LM (Johnson and Charniak, 2004; Johnson et al., 2004). Some works have shown that rescoring the n-best analyses with external n-gram (Zwarts and Johnson, 2011) and deep learning LMs (Jamshid Lou and Johnson, 2017) trained on large speech and non-speech corpora, and using the LM scores along with other features (i.e. pattern match and NCM ones) into a MaxEnt reranker (Johnson et al., 2004) improves the performance of the baseline NCM, although this creates complex runtime dependencies.

Parsing-based approaches detect disfluencies while simultaneously identifying the syntactic structure of the sentence. Typically, this is achieved by augmenting a transition-based dependency parser with a new action to detect and remove the disfluent parts of the sentence and their dependencies from the stack (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014; Yoshikawa et al., 2016). Joint parsing and disfluency detection can compare favorably to pipelined approaches, but requires large annotated treebanks containing both disfluent and syntatic structures for training.

Our proposed approach, based on an auto-correlational neural network (ACNN), belongs to the class of sequence tagging approaches. These approaches use classification techniques such as conditional random fields (Liu et al., 2006; Ostendorf and Hahn, 2013; Zayats et al., 2014; Ferguson et al., 2015), hidden Markov models (Liu et al., 2006; Schuler et al., 2010) and deep learning based models (Hough and Schlangen, 2015; Zayats et al., 2016) to label individual words as fluent or disfluent. In much of the previous work on sequence tagging approaches, improved performance has been gained by proposing increasingly complicated labeling schemes. In this case, a model with begin-inside-outside (BIO) style states which labels words as being inside or outside of edit region[1] is usually used as the baseline sequence tagging model. Then in order to come up with different pattern matching lexical cues for repetition and correction disfluencies, they extend the baseline state space with new explicit repair states to consider the words at repair region, in addition to edit region (Ostendorf and Hahn,

---

[1] For state labels, edit corresponds to reparandum.

2013; Zayats et al., 2014, 2016). A model which uses such labeling scheme may generate illegal label sequences at test time. As a solution, integer linear programming (ILP) constraints are applied to the output of classifier to avoid inconsistencies between neighboring labels (Georgila, 2009; Georgila et al., 2010; Zayats et al., 2016). This contrasts with our more straightforward approach, which directly labels words as being fluent or disfluent, and does not require any post-processing or annotation modifications.

The most similar work to ours is recent work by Zayats et al. (2016) that investigated the performance of a bidirectional long-short term memory network (BLSTM) for disfluency detection. Zayats et al. (2016) reported that a BLSTM operating only on words underperformed the same model augmented with hand-crafted pattern match features and POS tags by 7% in terms of f-score. In addition to lexically grounded features, some works incorporate prosodic information extracted from speech (Kahn et al., 2005; Ferguson et al., 2015; Tran et al., 2018). In this work, our primary motivation is to rectify the architectural limitations that prevent deep neural networks from automatically learning appropriate features from words alone. Therefore, our proposed model eschews manually engineered features and other representations derived from dependency parsers, language models or tree adjoining grammar transducers that are used to find "rough copy" dependencies. Instead, we aim to capture these kinds of dependencies automatically.

3 Convolutional and Auto-Correlational Networks

In this section, we introduce our proposed auto-correlation operator and the resulting auto-correlational neural network (ACNN) which is the focus of this work.

A convolutional or auto-correlational network computes a series of h feature representations $X^{(0)}, X^{(1)}, \ldots, X^{(h)}$, where $X^{(0)}$ is the input data, $X^{(h)}$ is the final (output) representation, and each non-input representation $X^{(k)}$ for $k > 0$, is computed from the preceding representation $X^{(k-1)}$ using a convolution or auto-correlation operation followed by an element-wise non-linear function.

Restricting our focus to convolutions in one dimension, as used in the context of language processing, each representation $X^{(k)}$ is a matrix of size (n, m_k), where n is the number of words in the input and m_k is the feature dimension of representation k, or equivalently it can be viewed as a sequence of n row vectors $X^{(k)} = (\boldsymbol{x}_1^{(k)}, \ldots, \boldsymbol{x}_n^{(k)})$, where $\boldsymbol{x}_t^{(k)}$ is the row vector of length m_k that represents the tth word at level k.

Consistent with the second interpretation, the input representation $X^{(0)} = (\boldsymbol{x}_1^{(0)}, \ldots, \boldsymbol{x}_n^{(0)})$ is a sequence of word embeddings, where m_0 is the length of the embedding vector and $\boldsymbol{x}_t^{(0)}$ is the word embedding for the tth word.

Each non-input representation $X^{(k)}, k > 0$ is formed by column-wise stacking the output of one or more convolution or auto-correlation operations applied to the preceding representation, and then applying an element-wise non-linear function. Formally, we define:

$$\begin{aligned} Y^{(k)} &= \left(F^{(k,1)}(X^{(k-1)}); \ldots; F^{(k,m_k)}(X^{(k-1)})\right) \\ X^{(k)} &= N^{(k)}(Y^{(k)}) \end{aligned} \tag{2}$$

where $F^{(k,u)}$ is the uth operator applied at layer k, and $N^{(k)}$ is the non-linear operation applied at layer k. Each operator $F^{(k,u)}$ (either convolution or auto-correlation) is a function from $X^{(k-1)}$, which is a matrix of size (n, m_{k-1}), to a vector of length n. A network that employs only convolution operators is a convolutional neural network (CNN). We call a network that utilizes a mixture of convolution and auto-correlation operators an *auto-correlational neural network* (ACNN). In our networks, the non-linear operation $N^{(k)}$ is always element-wise $ReLU$, except for the last layer, which uses a $softmax$ non-linearity.

3.1 Convolution Operator

A one-dimensional convolution operation maps an input matrix $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, where each $\boldsymbol{x}_t$ is a row vector of length m, to an output vector $\boldsymbol{y}$ of length n. The convolution operation is defined by a convolutional kernel A, which is applied to a window of words to produce a new output representation, and kernel width parameters ℓ and r, which define the number of words to the left and right of the target word included in the convolutional window. For example, assuming appropriate input padding where necessary, element y_t in the output vector $\boldsymbol{y}$ is computed as:

$$y_t \quad = \quad A \cdot X_{i:j} + \boldsymbol{b} \tag{3}$$

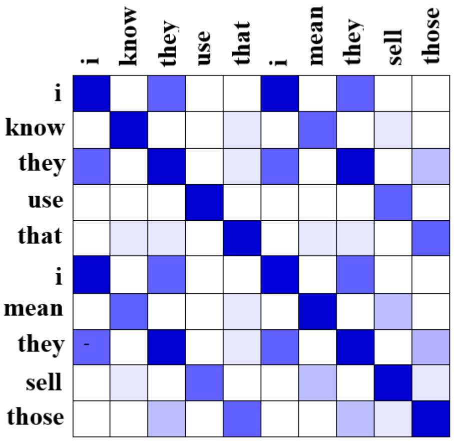

Figure 1: Cosine similarity between word embedding vectors learned by the ACNN model for the sentence "I know they use that I mean they sell those" (with disfluent words highlighted). In the figure, darker shades denote higher cosine values. "Rough copies" are clearly indicated by darkly shaded diagonals, which can be detected by our proposed auto-correlation operator.

where

A is a learned convolutional kernel of dimension $(\ell + r, m)$,

$X_{i:j}$ is the sub-matrix formed by selecting rows i to j from matrix X,

$\cdot$ is the dot product (a sum over elementwise multiplications),

i, j are given by $i = t - \ell$ and $j = t + r$, indicating the left and right extremities of the convolutional window effecting element y_t,

$\ell > 0$ is the left kernel width, and

$r > 0$ is right kernel width.

$\boldsymbol{b}$ is a learned bias vector of dimension n,

3.2 Auto-Correlation Operator

The auto-correlational operator is a generalisation of the convolution operator:

$$y_t = A \cdot X_{i:j} + B \cdot \hat{X}_{i:j,i:j} + \boldsymbol{b} \tag{4}$$

where y_t, A, X, $\boldsymbol{b}$, i and j are as in the convolution operator, and

$\hat{X}$ is a tensor of size (n, n, m) such that each vector $\hat{X}_{i,j,:}$ is given by $f(\boldsymbol{x}_i, \boldsymbol{x}_j)$,

$f(\boldsymbol{u}, \boldsymbol{v})$ is a binary operation on vectors, such as the Hadamard or element-wise product (i.e., $f(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u} \circ \boldsymbol{v}$), and

$\hat{X}_{i:j,i:j}$ is the sub-tensor formed by selecting indices i to j from the first two dimensions of tensor $\hat{X}$,

B is a learned convolutional kernel of dimension $(\ell + r, \ell + r, m)$.

Unlike convolution operations, which are linear, the auto-correlation operator introduces second-order interaction terms through the tensor $\hat{X}$ (since it multiplies the vector representations for each pair of input words). This naturally encodes the similarity between input words when applied at level $k = 1$ (or the co-activations of multiple CNN features, if applied at higher levels). As illustrated in Figure 1, blocks of similar words are indicative of "rough copies". We provide an illustration of the auto-correlation operation in Figure 2.

4 Experiments

4.1 Switchboard Dataset

We evaluate the proposed ACNN model for disfluency detection on the Switchboard corpus of conversational speech (Godfrey and Holliman, 1993). Switchboard is the largest available corpus (1.2×10^6 tokens) where disfluencies are annotated according to Shriberg's (1994) scheme:

[reparandum + {interregnum} repair]

where (+) is the interruption point marking the end of reparandum and {} indicate optional interregnum. We collapse this annotation to a binary classification scheme in which reparanda are labeled as disfluent and all other words as fluent. We disregard interregnum words as they are trivial to detect as discussed in Section 1.

Following Charniak and Johnson (2001), we split the Switchboard corpus into training, dev and test set as follows: training data consists of all sw[23]$*$.dff files, dev training consists of all sw4[5-9]$*$.dff files and test data consists of all sw4[0-1]$*$.dff files. We lower-case all text and remove all partial words and punctuations from the training data to make our evaluation both harder and more realistic (Johnson and Charniak, 2004). Partial words are strong indicators of disfluency; however, speech recognition models never generate them in their outputs.

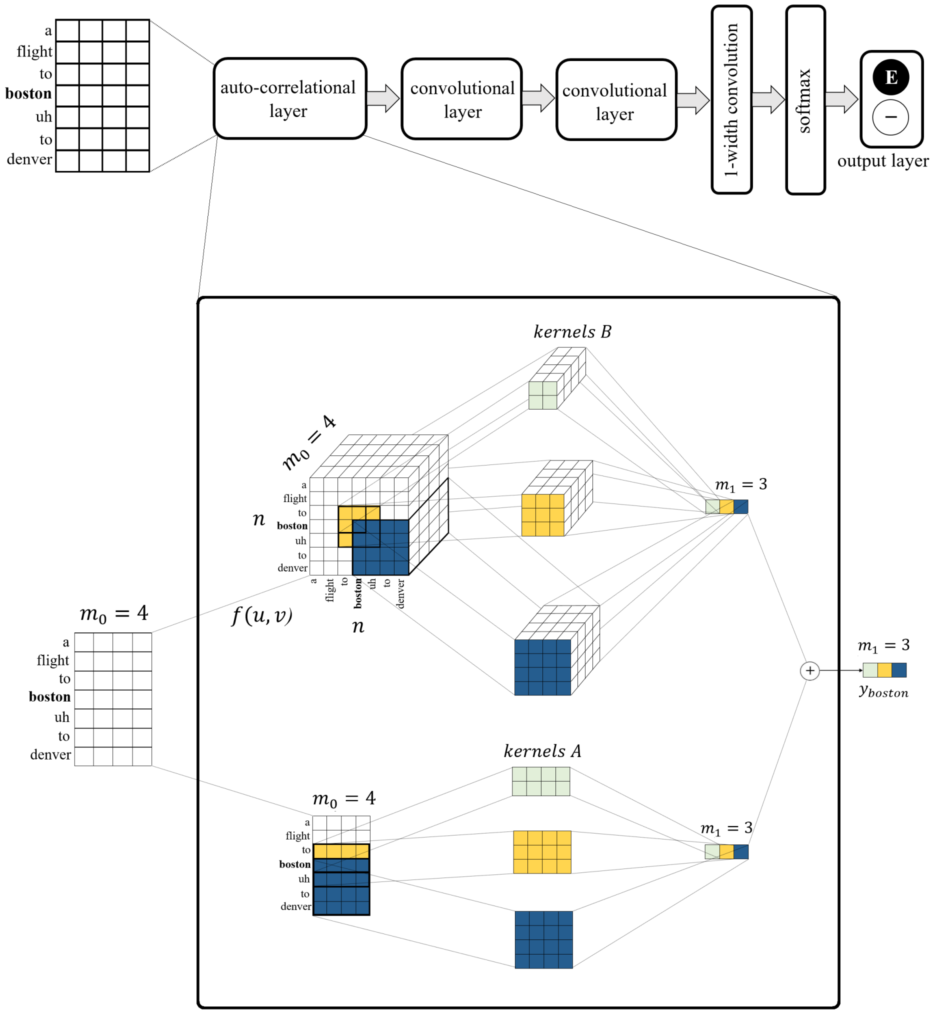

Figure 2: ACNN overview for labeling the target word "boston". A patch of words is fed into an auto-correlational layer. At inset bottom, the given patch of words is convolved with 2D kernels A of different sizes. At inset top, an auto-correlated tensor of size (n, n, m_0) is constructed by comparing each input vector $\boldsymbol{u} = \boldsymbol{x}_t$ with the input vector $\boldsymbol{v} = \boldsymbol{x}_{t'}$ using a binary function $f(\boldsymbol{u}, \boldsymbol{v})$. The auto-correlated tensor is convolved with 3D kernels B of different sizes. Each kernel group A and B outputs a matrix of size (n, m_1) (here, we depict only the row vector relating to the target word "boston"). These outputs are added element-wise to produce the feature representation that is passed to further convolutional layers, followed by a softmax layer. "E" = disfluent, "_" = fluent and m_0 = embedding size.

4.2 ACNN and CNN Baseline Models

We investigate two neural network models for disfluency detection; our proposed auto-correlational neural network (ACNN) and a convolutional neural network (CNN) baseline. The CNN baseline contains three convolutional operators (layers), followed by a width-1 convolution and a softmax output layer (to label each input word as either fluent or disfluent). The ACNN has the same general architecture as the baseline, except that we have replaced the first convolutional operator with an auto-correlation operator, as illustrated in Figure 2.

To ensure that equal effort was applied to the hyperparameter optimization of both models, we use randomized search (Bergstra and Bengio, 2012) to tune the optimization and architecture parameters separately for each model on the dev set, and to find an optimal stopping point for training. This results in different dimensions for each model. As indicated by Table 1, the resulting ACNN configuration has far fewer kernels at each layer than the CNN. However, as the auto-correlation kernels contain an additional dimension, both models have a similar number of parameters overall. Therefore, both models should have similar learning capacity except for their architectural differences (which is what we wish to investigate). Finally, we note that the resulting maximum right kernel width r_1 in the auto-correlational layer is 6. As illustrated in Figure 3, this is sufficient to capture almost all the "rough copies" in the Switchboard dataset (but could be increased for other datasets).

Configuration	CNN	ACNN
embedding dim	290	290
dropout rate	0.51	0.53
L_2 regularizer weight	0.13	0.23
#kernels at each layer	570	120
#kernel sizes at each layer	3	2
#words at left context ℓ_1	[0,1,4]	[5,3]
#words at left context ℓ_2	[1,2,3]	[4,2]
#words at left context ℓ_3	[0,1,2]	[3,2]
#words at right context r_1	[1,1,4]	[6,3]
#words at right context r_2	[1,2,4]	[5,3]
#words at right context r_3	[1,2,3]	[4,2]
#parameters	4.9M	4.9M

Table 1: Configuration of the CNN and ACNN models, where ℓ_k refers to the left kernel width at layer k, and r_k refers to the right kernel width at layer k. Both models have a similar total number of parameters.

For the ACNN, we considered a range of possible binary functions $f(\boldsymbol{u}, \boldsymbol{v})$ to compare the input vector $\boldsymbol{u} = \boldsymbol{x}_t$ with the input vector $\boldsymbol{v} = \boldsymbol{x}_{t'}$ in the auto-correlational layer. However, in initial experiments we found that the Hadamard or element-wise product (i.e. $f(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u} \circ \boldsymbol{v}$) achieved the best results. We also considered concatenating the outputs of kernels A and B in Equation 4, but we found that element-wise addition produced slightly better results on the dev set.

4.2.1 Implementation Details

In both models, we use $ReLU$ for the non-linear operation, all stride sizes are one word and there are no pooling operations. We randomly initialize the word embeddings and all weights of the model from a uniform distribution. The bias terms are initialized to be 1. To reduce overfitting, we apply dropout (Srivastava et al., 2014) to the input word embeddings and L_2 regularization to the weights of the width-1 convolutional layer. For parameter optimization, we use the Adam optimizer (Kingma and Ba, 2014) with a mini-batch size of 25 and an initial learning rate of 0.001.

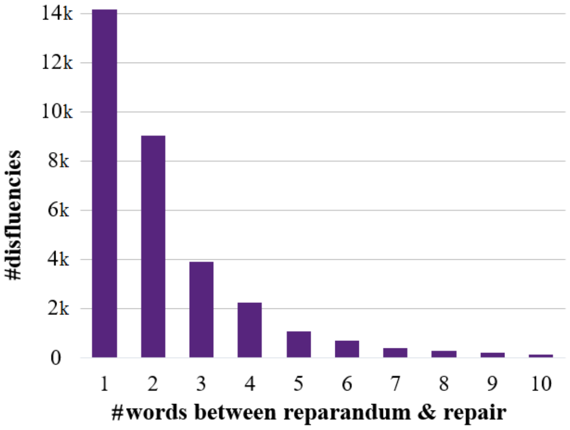

Figure 3: Distribution over the number of words in between the reparandum and the interregnum in the Switchboard training set (indicating the distance between "rough copies").

5 Results

As in previous work (Johnson and Charniak, 2004), we evaluate our model using precision, recall and f-score, where true positives are the words in the edit region (i.e., the reparandum words). As Charniak and Johnson (2001) observed, only 6% of words in the Switchboard corpus are disfluent, so accuracy is not a good measure of system performance. F-score, on the other hand, focuses more on detecting "edited" words, so it is more appropriate for highly skewed data.

Table 2 compares the dev set performance of the ACNN model against our baseline CNN, as well as the LSTM and BLSTM models proposed by Zayats et al. (2016) operating only on word inputs (i.e., without any disfluency pattern-match features). Our baseline CNN outperforms both the LSTM and the BLSTM, while the ACNN model clearly outperforms the baseline CNN, with a further 5% increase in f-score. In particular, the ACNN noticably improves recall without degrading precision.

model	P	R	F
BLSTM (words)*	87.8	71.1	78.6
LSTM (words)*	87.6	71.4	78.7
CNN	89.4	74.6	81.3
ACNN	**90.0**	**82.8**	**86.2**

Table 2: Precision (P), recall (R) and f-score (F) on the dev set for the BLSTM and LSTM models using words alone from *Zayats et al. (2016), as well as our baseline CNN and ACNN model.

To further investigate the differences between the two CNN-based models, we randomly select 100 sentences containing disfluencies from the Switchboard dev set and categorize them according to Shriberg's (1994) typology of speech repair disfluencies. *Repetitions* are repairs where the reparandum and repair portions of the disfluency are identical, while *corrections* are where the reparandum and repairs differ (so corrections are much harder to detect). *Restarts* are where the speaker abandons a sentence prefix, and starts a fresh sentence. As Table 3 shows, the ACNN model is better at detecting repetition and correction disfluencies than the CNN, especially for the more challenging correction disfluencies. On the other hand, the ACNN is no better than the baseline at detecting restarts, probably because the restart typically does not involve a rough copy dependency. Luckily restarts are much rarer than repetition and correction disfluencies.

model	**Rep.**	**Cor.**	**Res.**	**All**
CNN	93.3	66.0	57.1	80.4
ACNN	97.5	80.0	57.1	88.9

Table 3: F-scores for different types of disfluencies on a subset of the Switchboard dev set containing 140 disfluent structures — including 85 repetitions (Rep.), 51 corrections (Cor.) and 4 restarts (Res.).

We also repeated the analysis of (Zayats et al., 2014) on the dev data, so we can compare our models to their extended BLSTM model with a 17-state CRF output and hand-crafted features, including partial-word and POS tag features that enable it to capture some "rough copy" dependencies. As expected, the ACNN outperforms both the CNN and the extended BLSTM model, especially in the "Other" category that involve the non-repetition dependencies.

model	**Rep.**	**Other**	**Either**
CNN	92.2	66.7	81.3
BLSTM (17 states)*	94.1	66.7	85.8
ACNN	96.6	73.3	86.2

Table 4: F-scores for different types of disfluencies for the CNN, ACNN and BLSTM (17 states) *(Zayats et al., 2016) using the Switchboard dev set.

Finally, we compare the ACNN model to state-of-the-art methods from the literature, evaluated on the Switchboard test set. Table 5 shows that the ACNN model is competitive with recent models from the literature. The three models that score more highly than the ACNN all rely on hand-crafted features, additional information sources such as partial-word features (which would not be available in a realistic ASR application), or external resources such as dependency parsers and language models. The ACNN, on the other hand, only uses whole-word inputs and learns the "rough copy" dependencies between words without requiring any manual feature engineering.

model	**P**	**R**	**F**
Yoshikawa et al.(2016) ⋄	67.9	57.9	62.5
Georgila et al. (2010) †	77.4	64.6	70.4
Tran et al. (2018) ⊗ ⋆	-	-	77.5
Kahn et al. (2005) ⋆	-	-	78.2
Johnson et al. (2004) ≀	82.0	77.8	79.7
Georgila (2009) †	-	-	80.1
Johnson et al. (2004) †≀	-	-	81.0
Rasooli et al. (2013) ⋄	85.1	77.9	81.4
Zwarts et al. (2011) ⋈ ≀	-	-	83.8
Qian et al. (2013) ⋈	-	-	84.1
Honnibal et al. (2014) ⋄	-	-	84.1
ACNN	**89.5**	**80.0**	**84.5**
Ferguson et al. (2015) ⋆	90.0	81.2	85.4
Zayats et al. (2016) ⊗†	91.8	80.6	85.9
Jamshid Lou et al. (2017) ⋈ ≀	-	-	86.8

Table 5: Comparison of the ACNN model to the state-of-the-art methods on the Switchboard test set. The other models listed have used richer inputs and/or rely on the output of other systems, as well as pattern match features, as indicated by the following symbols: ⋄ dependency parser, † hand-crafted constraints/rules, ⋆ prosodic cues, ≀ tree adjoining grammar transducer, ⋈ refined/external language models and ⊗ partial words. P = precision, R = recall and F = f-score.

5.1 Qualitative Analysis

We conduct an error analysis on the Switchboard dev set to characterize the disfluencies that the ACNN model can capture and those which are difficult for the model to detect. In the following examples, the highlighted words indicate ground truth disfluency labels and the underlined ones are the ACNN predictions.

1. But if you let them yeah if you let them in a million at a time it wouldn't make that you know it wouldn't make that big a bulge in the

population

2. They’re handy uh they they come in handy at the most unusual times
3. My mechanics loved it because it was an old it was a sixty-five buick
4. Well I I I think we did I think we did learn some lessons that we weren’t uh we weren’t prepared for
5. Uh I have never even I have never even looked at one closely
6. But uh when I was when my kids were young I was teaching at a university
7. She said she’ll never put her child in a in a in a in a in a preschool
8. Well I think they’re at they’re they’ve come a long way
9. I I like a I saw the the the the tapes that were that were run of marion berry’s drug bust
10. But I know that in some I know in a lot of rural areas they’re not that good

According to examples 1-10, the ACNN detects *repetition* (e.g. 1, 5) and *correction* disfluencies (e.g. 3, 6, 10). It also captures complex structures where there are multiple or nested disfluencies (e.g. 2, 8) or stutter-like repetitions (e.g. 4, 7, 9).

11. My point was that there is for people who don’t want to do the military service it would be neat if there were an alternative . . .
12. I believe from what I remember of the literature they gave uh if you fail I believe they give you one more chance
13. Kind of a coarse kind of test
14. So we could pour concrete and support it with a a nice firm four by four posts
15. But uh I’m afraid I’m I’m probably in the minority
16. Same thing same thing that the her kids had
17. Did you you framed it in uh on on you framed in new square footage
18. And and and there needs to be a line drawn somewhere at reasonable and proper
19. . . . I think there’s a couple of levels of tests in terms of of drugs
20. See they have uh we have two the both c spans here

In some cases where repetitions are fluent, the model has incorrectly detected the first occurence of the word as disfluency (e.g. 13, 14, 15, 19). Moreover, when there is a long distance between reparandum and repair words (e.g. 11, 12), the model usually fails to detect the reparanda. In some sentences, the model is also unable to detect the disfluent words which result in ungrammatical sentences (e.g. 16, 17, 18, 20). In these examples, the undetected disfluencies “the”, “did”, “at” and “two the” cause the residual sentence to be ungrammatical.

We also discuss the types of disfluency captured by the ACNN model, but not by the baseline CNN. In the following examples, the ACNN predictions (underlined words) are the same as the ground truth disfluency labels (highlighted words). The **bolded** words indicate the CNN prediction of disfluencies.

21. Uh well **I** actually my dad’s my dad’s almost ninety
22. Not a man not a repair man but just a friend
23. we**’re** from a county we’re from the county they marched in
24. Now let’s now we’re done
25. And they’ve most of them have been pretty good
26. I do as far as uh as far as uh as far as immigration as a whole goes
27. No need to use this to play around with this space stuff anymore
28. We couldn’t survive **in a** in a juror in a trial system without a jury
29. You stay within your uh within your means
30. So **we’re** we’re part we’re actually part of MIT

The ACNN model has a generally better performance in detecting "rough copies" which are important indicator of *repetition* (e.g. 21, 29), *correction* (e.g. 22, 23, 24, 25, 27), and *stutter-like* (e.g. 26, 28, 30) disfluencies.

6 Conclusion

This paper presents a simple new model for disfluency detection in spontaneous speech transcripts. It relies on a new auto-correlational kernel that is designed to detect the "rough copy" dependencies that are characteristic of speech disfluencies, and combines it with conventional convolutional kernels to form an auto-correlational neural network (ACNN). We show experimentally that using the ACNN model improves over a CNN baseline on disfluency detection task, indicating that the auto-correlational kernel can in fact detect the rough copy dependencies between words in disfluencies. The addition of the auto-correlational kernel permits a fairly conventional architecture to achieve near state-of-the-art results without complex hand-crafted features or external information sources.

We expect that the performance of the ACNN model can be further improved in future by using more complex similarity functions and by incorporating similar kinds of external information (e.g. prosody) used in other disfluency models. In future work, we also intend to investigate other applications of the auto-correlational kernel. The auto-correlational layer is a generic neural network layer, so it can be used as a component of other architectures, such as RNNs. It might also be useful in very different applications such as image processing.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This research was supported by a Google award through the Natural Language Understanding Focused Program, and under the Australian Research Councils Discovery Projects funding scheme (project number DP160102156).

References

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(1):281–305.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies (NAACL'01)*, pages 118–126, Stroudsburg, USA.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML'17)*, pages 160–167, Helsinki, Finland.

James Ferguson, Greg Durrett, and Dan Klein. 2015. Disfluency detection with a semi-Markov model and prosodic features. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'15)*, pages 257–262, Denver, USA.

Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'09)*, pages 109–112, Boulder, USA.

Kallirroi Georgila, Ning Wang, and Jonathan Gratch. 2010. Cross-domain speech disfluency detection. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL'10)*, pages 237–240, Tokyo, Japan.

John Godfrey and Edward Holliman. 1993. Switchboard-1 release 2 LDC97S62. Published by: Linguistic Data Consortium, Philadelphia, USA.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics (TACL)*, 2(1):131–142.

Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. In *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH'15)*, pages 845–853, Dresden, Germany.

Paria Jamshid Lou and Mark Johnson. 2017. Disfluency detection using a noisy channel model and a deep neural language model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*, pages 547–553.

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL'04)*, pages 33–39, Barcelona, Spain.

Mark Johnson, Eugene Charniak, and Matthew Lease. 2004. An improved model for recognizing disfluencies in conversational speech. In *Proceedings of Rich Transcription Workshop*.

Jeremy Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective use of prosody in parsing conversational speech. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT'05)*, pages 233–240, Tallinn, Estonia.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Yang Liu, Elizabeth Shriberg, Andreas Stolckeand, Dustin Hillard, Mari Ostendorf, and Mary Harper. 2006. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 14(5):1526–1540.

Mari Ostendorf and Sangyun Hahn. 2013. A sequential repetition model for improved disfluency detection. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH'13)*, pages 2624–2628, Lyon, France.

Barbara Plank, Andersand Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*, pages 412–418, Berlin, Germany.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'13)*, pages 820–825, Atlanta, USA.

Mohammad Sadegh Rasooli and Joel Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, pages 124–129, Seattle, USA.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage parsing using human-like memory constraints. *Computational Linguistics*, 36(1):1–30.

Stuart Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th Conference on Computational Linguistics (COLING'90)*, pages 253–258, Helsinki, Finland.

Elizabeth Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis, University of California, Berkeley, USA.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Trang Tran, Shubham Toshniwal, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Mari Ostendorf. 2018. Parsing speech: A neural approach to integrating lexical and acoustic-prosodic information. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'18)*, pages 69–81, New Orleans, USA.

Masashi Yoshikawa, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Joint transition-based dependency parsing and disfluency detection for automatic speech recognition texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*, pages 1036–1041.

Xiang Yu, Agnieszka Falenska, and Ngoc Thang Vu. 2017. A general-purpose tagger with convolutional neural networks. In *Proceedings of the 1st Workshop on Subword and Character Level Models in NLP*, pages 124–129, Copenhagen, Denmark.

Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2014. Multi-domain disfluency and repair detection. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association (INTERSPEECH'15)*, pages 2907–2911, Singapore.

Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional LSTM. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (INTERSPEECH'16)*, pages 2523–2527, San Francisco, USA.

Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT'11)*, pages 703–711, Portland, USA.

Pyramidal Recurrent Unit for Language Modeling

Sachin Mehta[1], Rik Koncel-Kedziorski[1], Mohammad Rastegari[2], and Hannaneh Hajishirzi[1]

[1]University of Washington, Seattle, WA, USA
{sacmehta, kedzior, hannaneh}@uw.edu
[2]Allen Institute for AI and XNOR.AI, Seattle, WA, USA
mohammadr@allenai.org

Abstract

LSTMs are powerful tools for modeling contextual information, as evidenced by their success at the task of language modeling. However, modeling contexts in very high dimensional space can lead to poor generalizability. We introduce the Pyramidal Recurrent Unit (PRU), which enables learning representations in high dimensional space with more generalization power and fewer parameters. PRUs replace the linear transformation in LSTMs with more sophisticated interactions including pyramidal and grouped linear transformations. This architecture gives strong results on word-level language modeling while reducing the number of parameters significantly. In particular, PRU improves the perplexity of a recent state-of-the-art language model Merity et al. (2018) by up to 1.3 points while learning 15-20% fewer parameters. For similar number of model parameters, PRU outperforms all previous RNN models that exploit different gating mechanisms and transformations. We provide a detailed examination of the PRU and its behavior on the language modeling tasks. Our code is open-source and available at `https://sacmehta.github.io/PRU/`.

1 Introduction

Long short term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) are popular for many sequence modeling tasks and are used extensively in language modeling. A key to their success is their articulated gating structure, which allows for more control over the information passed along the recurrence. However, despite the sophistication of the gating mechanisms employed in LSTMs and similar recurrent units, the input and context vectors are treated with simple linear transformations prior to gating. Non-linear transformations such as convolutions (Kim et al., 2016) have been used, but these have not achieved the performance of well regularized LSTMs for language modeling (Melis et al., 2018).

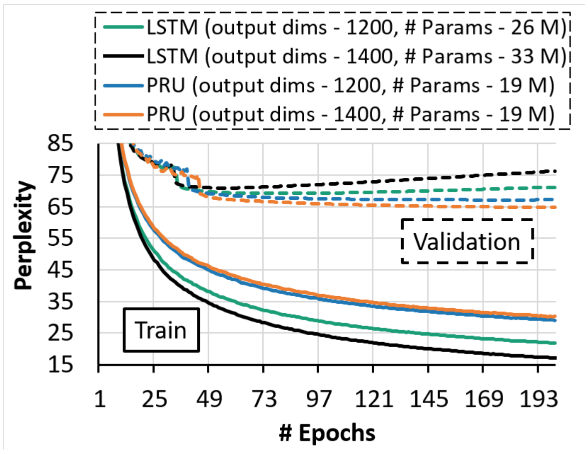

Figure 1: Comparison of training (solid lines) and validation (dashed lines) perplexities on the Penn Treebank with standard dropout for pyramidal recurrent units (PRU) and LSTM. PRUs learn latent representations in very high-dimensional space with good generalizability and fewer parameters. See Section 3 for more details about PRUs. Best viewed in color.

A natural way to improve the expressiveness of linear transformations is to increase the number of dimensions of the input and context vectors, but this comes with a significant increase in the number of parameters which may limit generalizability. An example is shown in Figure 1, where LSTMs performance decreases with the increase in dimensions of the input and context vectors. Moreover, the semantics of the input and context vectors are different, suggesting that each may benefit from specialized treatment.

Guided by these insights, we introduce a new recurrent unit, the Pyramidal Recurrent Unit (PRU), which is based on the LSTM gating structure. Figure 2 provides an overview of the PRU. At

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4620–4630
Brussels, Belgium, October 31 - November 4, 2018.

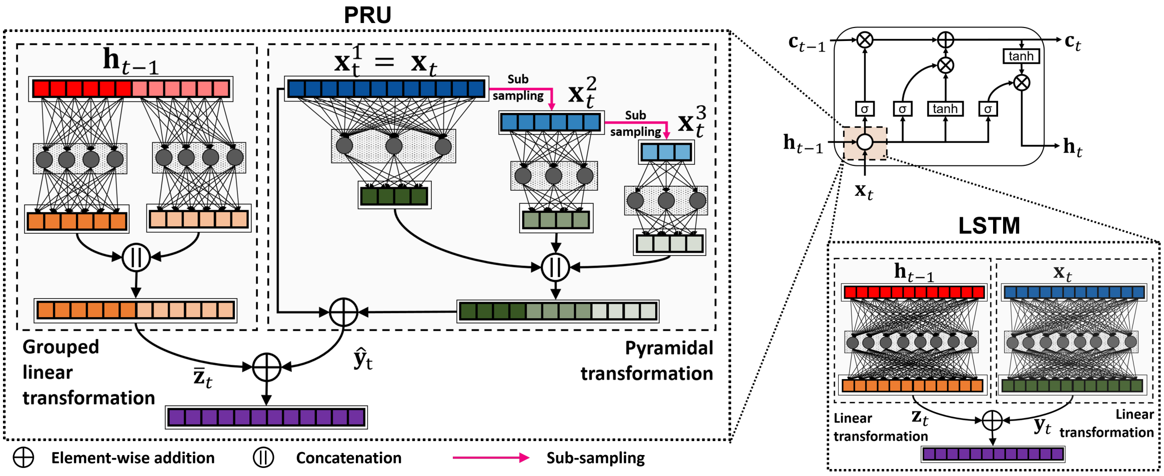

Figure 2: Block diagram visualizing the transformations in pyramidal recurrent unit (left) and the LSTM (bottom right) along with the LSTM gating architecture (top right). **Blue**, **red**, **green** (or **orange**), and **purple** signify the current input $\mathbf{x}_t$, output of the previous cell $\mathbf{h}_{t-1}$, the output of transformations, and the fused output, respectively. The color intensity is used to represent sub-sampling and grouping operations.

the heart of the PRU is the *pyramidal transformation* (PT), which uses subsampling to effect multiple views of the input vector. The subsampled representations are combined in a pyramidal fusion structure, resulting in richer interactions between the individual dimensions of the input vector than is possible with a linear transformation. Context vectors, which have already undergone this transformation in the previous cell, are modified with a *grouped linear transformation* (GLT) which allows the network to learn latent representations in high dimensional space with fewer parameters and better generalizability (see Figure 1).

We show that PRUs can better model contextual information and demonstrate performance gains on the task of language modeling. The PRU improves the perplexity of the current state-of-the-art language model (Merity et al., 2018) by up to 1.3 points, reaching perplexities of 56.56 and 64.53 on the Penn Treebank and WikiText2 datasets while learning 15-20% fewer parameters. Replacing an LSTM with a PRU results in improvements in perplexity across a variety of experimental settings. We provide detailed ablations which motivate the design of the PRU architecture, as well as detailed analysis of the effect of the PRU on other components of the language model.

2 Related work

Multiple methods, including a variety of gating structures and transformations, have been proposed to improve the performance of recurrent neural networks (RNNs). We first describe these approaches and then provide an overview of recent work in language modeling.

Gating-based mechanisms: The performance of RNNs have been greatly improved by gating mechanisms such as LSTMs (Hochreiter and Schmidhuber, 1997), GRUs (Chung et al., 2014), peep-hole connections (Gers and Schmidhuber, 2000), SRUs (Lei et al., 2018), and RANs (Lee et al., 2017). In this paper, we extend the gating architecture of LSTMs (Hochreiter and Schmidhuber, 1997), a widely used recurrent unit across different domains.

Transformations: Apart from the widely used linear transformation for modeling the temporal data, another transformation that has gained popularity is convolution (LeCun et al., 1995). Convolution-based methods have gained attention in computer vision tasks (Krizhevsky et al., 2012) as well as some of the natural language processing tasks including machine translation (Gehring et al., 2017). Convolution-based methods for language modeling, such as CharCNN (Kim et al., 2016), have not yet achieved the performance of well regularized LSTMs (Melis et al., 2018). We inherit ideas from convolution-based approaches, such as sub-sampling, to learn richer representations (Krizhevsky et al., 2012; Han et al., 2017).

Regularization: Methods such as dropout (Srivastava et al., 2014), variational dropout (Kingma et al., 2015), and weight dropout (Merity et al., 2018) have been proposed to regularize RNNs. These methods can be easily applied to PRUs.

Other efficient RNN networks: Recently, there has been an effort to improve the efficiency of RNNs. These approaches include quantization (Xu et al., 2018), skimming (Seo et al., 2018; Yu et al., 2017), skipping (Campos et al., 2018), and query reduction (Seo et al., 2017). These approaches extend standard RNNs and therefore, these approaches are complementary to our work.

Language modeling: Language modeling is a fundamental task for NLP and has garnered significant attention in recent years (see Table 1 for comparison with state-of-the-art methods). Merity et al. (2018) introduce regularization techniques such as weight dropping which, coupled with a non-monotonically triggered ASGD optimization, achieves strong performance improvements. Yang et al. (2018) extend Merity et al. (2018) with the mixture of softmaxes (MoS) technique, which increases the rank of the matrix used to compute next-token probabilities. Further, Merity et al. (2017) and Krause et al. (2018) propose methods to improve inference by adapting models to recent sequence history. Our work is complementary to these recent softmax layer and inference procedure improvements.

We extend state-of-the-art language model in Merity et al. (2018) by replacing the LSTM with the PRU. We show by experiments that the PRU improves the performance of Merity et al. (2018) while learning fewer parameters.

3 Pyramidal Recurrent Units

We introduce Pyramidal Recurrent Units (PRUs), a new RNN architecture which improves modeling of context by allowing for higher dimensional vector representations while learning fewer parameters. Figure 2 provides an overview of PRU. We first elaborate on the details of the pyramidal transformation and the grouped linear transformation. We then describe our recurrent unit, PRU.

3.1 Pyramidal transformation for input

The basic transformation in many recurrent units is a linear transformation $\mathcal{F}_L$ defined as:

$$\mathbf{y} = \mathcal{F}_L(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x}, \qquad (1)$$

where $\mathbf{W} \in \mathbb{R}^{N \times M}$ are learned weights that linearly map $\mathbf{x} \in \mathbb{R}^N$ to $\mathbf{y} \in \mathbb{R}^M$. To simplify notation, we omit the biases.

Motivated by successful applications of sub-sampling in computer vision (e.g., (Burt and Adelson, 1987; Lowe, 1999; Krizhevsky et al., 2012; Mehta et al., 2018)), we subsample input vector $\mathbf{x}$ into K pyramidal levels to achieve representation of the input vector at multiple scales. This sub-sampling operation produces K vectors, represented as $\mathbf{x}^k \in \mathbb{R}^{\frac{N}{2^{k-1}}}$, where 2^{k-1} is the sampling rate and $k = \{1, \cdots, K\}$. We learn scale-specific transformations $\mathbf{W}^k \in \mathbb{R}^{\frac{N}{2^{k-1}} \times \frac{M}{K}}$ for each $k = \{1, \cdots K\}$. The transformed sub-samples are concatenated to produce the pyramidal analog to $\mathbf{y}$, here denoted as $\bar{\mathbf{y}} \in \mathbb{R}^M$:

$$\bar{\mathbf{y}} = \mathcal{F}_P(\mathbf{x}) = \left[\mathbf{W}^1 \cdot \mathbf{x}^1, \cdots, \mathbf{W}^K \cdot \mathbf{x}^K\right], \qquad (2)$$

where $[\cdot, \cdot]$ indicates concatenation. We note that pyramidal transformation with $K = 1$ is the same as the linear transformation.

To improve gradient flow inside the recurrent unit, we combine the input and output using an element-wise sum (when dimension matches) to produce residual analog of pyramidal transformation, as shown in Figure 2 (He et al., 2016).

Sub-sampling: We sub-sample the input vector $\mathbf{x}$ into K pyramidal levels using the kernel-based approach (LeCun et al., 1995; Krizhevsky et al., 2012). Let us assume that we have a kernel κ with $2e + 1$ elements. Then, the input vector $\mathbf{x}$ can be sub-sampled as:

$$\mathbf{x}^k = \sum_{i=1}^{N/s} \sum_{j=-e}^{e} \mathbf{x}^{k-1}[si]\kappa[j], \qquad (3)$$

where s represents the stride and $k = \{2, \cdots, K\}$.

Reduction in parameters: The number of parameters learned by the linear transformation and the pyramidal transformation with K pyramidal levels to map $\mathbf{x} \in \mathbb{R}^N$ to $\bar{\mathbf{y}} \in \mathbb{R}^M$ are NM and $\frac{NM}{K} \sum\limits_{k=1}^{K} 2^{(1-k)}$ respectively. Thus, pyramidal transformation reduces the parameters of a linear transformation by a factor of $K(\sum_{k=1}^{K} 2^{(1-k)})^{-1}$. For example, the pyramidal transformation (with $K = 4$ and $N = M = 600$) learns 53% fewer parameters than the linear transformation.

3.2 Grouped linear transformation for context

Many RNN architectures apply linear transformations to both the input and context vector. However, this may not be ideal due to the differing semantics of each vector. In many NLP applications including language modeling, the input vector is a dense word embedding which is shared across all contexts for a given word in a dataset. In contrast, the context vector is highly contextualized by the current sequence. The differences between the input and context vector motivate their separate treatment in the PRU architecture.

The weights learned using the linear transformation (Eq. 1) are reused over multiple time steps, which makes them prone to over-fitting (Gal and Ghahramani, 2016). To combat over-fitting, various methods, such as variational dropout (Gal and Ghahramani, 2016) and weight dropout (Merity et al., 2018), have been proposed to regularize these recurrent connections. To further improve generalization abilities while simultaneously enabling the recurrent unit to learn representations at very high dimensional space, we propose to use grouped linear transformation (GLT) instead of standard linear transformation for recurrent connections (Kuchaiev and Ginsburg, 2017). While pyramidal and linear transformations can be applied to transform context vectors, our experimental results in Section 4.4 suggests that GLTs are more effective.

The linear transformation $\mathcal{F}_L : \mathbb{R}^N \to \mathbb{R}^M$ maps $\mathbf{h} \in \mathbb{R}^N$ linearly to $\mathbf{z} \in \mathbb{R}^M$. Grouped linear transformations break the linear interactions by factoring the linear transformation into two steps. First, a GLT splits the input vector $\mathbf{h} \in \mathbb{R}^N$ into g smaller groups such that $\mathbf{h} = \{\mathbf{h}^1, \cdots, \mathbf{h}^g\}, \forall\, \mathbf{h}^i \in \mathbb{R}^{\frac{N}{g}}$. Second, a linear transformation $\mathcal{F}_L : \mathbb{R}^{\frac{N}{g}} \to \mathbb{R}^{\frac{M}{g}}$ is applied to map $\mathbf{h}^i$ linearly to $\mathbf{z}^i \in \mathbb{R}^{\frac{M}{g}}$, for each $i = \{1, \cdots, g\}$. The g resultant output vectors $\mathbf{z}^i$ are concatenated to produce the final output vector $\bar{\mathbf{z}} \in \mathbb{R}^M$.

$$\bar{\mathbf{z}} = \mathcal{F}_G(\mathbf{h}) = \left[\mathbf{W}^1 \cdot \mathbf{h}^1, \cdots, \mathbf{W}^g \cdot \mathbf{h}^g\right] \quad (4)$$

GLTs learn representations at low dimensionality. Therefore, a GLT requires g fewer parameters than the linear transformation. We note that GLTs are subset of linear transformations. In a linear transformation, each neuron receives an input from each element in the input vector while in a GLT, each neuron receives an input from a subset of the input vector. Therefore, GLT is the same as a linear transformation when $g = 1$.

3.3 Pyramidal Recurrent Unit

We extend the basic gating architecture of LSTM with the pyramidal and grouped linear transformations outlined above to produce the Pyramidal Recurrent Unit (PRU), whose improved sequence modeling capacity is evidenced in Section 4.

At time t, the PRU combines the input vector $\mathbf{x}_t$ and the previous context vector (or previous hidden state vector) $\mathbf{h}_{t-1}$ using the following transformation function as:

$$\hat{\mathcal{G}}_v(\mathbf{x}_t, \mathbf{h}_{t-1}) = \hat{\mathcal{F}}_P(\mathbf{x}_t) + \mathcal{F}_G(\mathbf{h}_{t-1}), \quad (5)$$

where $v \in \{f, i, c, o\}$ indexes the various gates in the LSTM model, and $\hat{\mathcal{F}}_P(\cdot)$ and $\mathcal{F}_G(\cdot)$ represents the pyramidal and grouped linear transformations defined in Eqns. 2 and 4, respectively.

We will now incorporate $\hat{\mathcal{G}}_v(\cdot, \cdot)$ into LSTM gating architecture to produce PRU. At time t, a PRU cell takes $\mathbf{x}_t \in \mathbb{R}^N$, $\mathbf{h}_{t-1} \in \mathbb{R}^M$, and $\mathbf{c}_{t-1} \in \mathbb{R}^M$ as inputs to produce forget $\mathbf{f}_t$, input $\mathbf{i}_t$, output $\mathbf{o}_t$, and content $\hat{\mathbf{c}}_t$ gate signals. The inputs are combined with these gate signals to produce context vector $\mathbf{h}_t \in \mathbb{R}^M$ and cell state $\mathbf{c}_t \in \mathbb{R}^M$. Mathematically, the PRU with the LSTM gating architecture can be defined as:

$$\begin{aligned}
\mathbf{f}_t &= \sigma\left(\hat{\mathcal{G}}_f(\mathbf{x}_t, \mathbf{h}_{t-1})\right) \\
\mathbf{i}_t &= \sigma\left(\hat{\mathcal{G}}_i(\mathbf{x}_t, \mathbf{h}_{t-1})\right) \\
\hat{\mathbf{c}}_t &= tanh\left(\hat{\mathcal{G}}_c(\mathbf{x}_t, \mathbf{h}_{t-1})\right) \\
\mathbf{o}_t &= \sigma\left(\hat{\mathcal{G}}_o(\mathbf{x}_t, \mathbf{h}_{t-1})\right) \\
\mathbf{c}_t &= \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \hat{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \otimes tanh(\mathbf{c}_t)
\end{aligned} \quad (6)$$

where $\otimes$ represents the element-wise multiplication operation, and σ and $tanh$ are the sigmoid and hyperbolic tangent activation functions. We note that LSTM is a special case of PRU when g=K=1.

4 Experiments

To showcase the effectiveness of the PRU, we evaluate the performance on two standard datasets for word-level language modeling and compare with state-of-the-art methods. Additionally, we provide a detailed examination of the PRU and its behavior on the language modeling tasks.

4.1 Set-up

Dataset: Following recent works, we compare on two widely used datasets, the Penn Treebank (PTB) (Marcus et al., 1993) as prepared by Mikolov et al. (2010) and WikiText2 (WT-2) (Merity et al., 2017). For both datasets, we follow the same training, validation, and test splits as in Merity et al. (2018).

Language Model: We extend the language model, AWD-LSTM (Merity et al., 2018), by replacing LSTM layers with PRU. Our model uses 3-layers of PRU with an embedding size of 400. The number of parameters learned by state-of-the-art methods vary from 18M to 66M with majority of the methods learning about 22M to 24M parameters on the PTB dataset. For a fair comparison with state-of-the-art methods, we fix the model size to 19M and vary the value of g and hidden layer sizes so that total number of learned parameters is similar across different configurations. We use 1000, 1200, and 1400 as hidden layer sizes for values of g=1,2, and 4, respectively. We use the same settings for the WT-2 dataset. We set the number of pyramidal levels K to two in our experiments and use average pooling for subsampling. These values are selected based on our ablation experiments on the validation set (Section 4.4). We measure the performance of our models in terms of word-level perplexity. We follow the same training strategy as in Merity et al. (2018).

To understand the effect of regularization methods on the performance of PRUs, we perform experiments under two different settings: (1) *Standard dropout*: We use a standard dropout (Srivastava et al., 2014) with probability of 0.5 after embedding layer, the output between LSTM layers, and the output of final LSTM layer. (2) *Advanced dropout:* We use the same dropout techniques with the same dropout values as in Merity et al. (2018). We call this model as AWD-PRU.

4.2 Results

Table 1 compares the performance of the PRU with state-of-the-art methods. We can see that the PRU achieves the best performance with fewer parameters.

Standard dropout: PRUs achieve either the same or better performance than LSTMs. In particular, the performance of PRUs improves with the increasing value of g. At $g = 4$, PRUs outperform LSTMs by about 4 points on the PTB dataset and by about 3 points on the WT-2 dataset. This is explained in part by the regularization effect of the grouped linear transformation (Figure 1). With grouped linear and pyramidal transformations, PRUs learn rich representations at very high dimensional space while learning fewer parameters. On the other hand, LSTMs overfit to the training data at such high dimensions and learn $1.4\times$ to $1.8\times$ more parameters than PRUs.

Advanced dropouts: With the advanced dropouts, the performance of PRUs improves by about 4 points on the PTB dataset and 7 points on the WT-2 dataset. This further improves with finetuning on the PTB (about 2 points) and WT-2 (about 1 point) datasets.

Comparison with state-of-the-art: For similar number of parameters, the PRU with standard dropout outperforms most of the state-of-the-art methods by large margin on the PTB dataset (e.g. RAN (Lee et al., 2017) by 16 points with 4M less parameters, QRNN (Bradbury et al., 2017) by 16 points with 1M more parameters, and NAS (Zoph and Le, 2017) by 1.58 points with 6M less parameters). With advanced dropouts, the PRU delivers the best performance. On both datasets, the PRU improves the perplexity by about 1 point while learning 15-20% fewer parameters.

Inference: PRU is a drop-in replacement for LSTM, therefore, it can improve language models with modern inference techniques such as dynamic evaluation (Krause et al., 2018). When we evaluate PRU-based language models (only with standard dropout) with dynamic evaluation on the PTB test set, the perplexity of PRU ($g = 4, k = 2, M = 1400$) improves from 62.42 to 55.23 while the perplexity of an LSTM ($M = 1000$) with similar settings improves from 66.29 to 58.79; suggesting that modern inference techniques are equally applicable to PRU-based language models.

4.3 Analysis

It is shown above that the PRU can learn representations at higher dimensionality with more generalization power, resulting in performance gains for language modeling. A closer analysis of the impact of the PRU in a language modeling system reveals several factors that help explain how the PRU achieves these gains.

	WT-2			PTB		
Model	**Params**	**Val**	**Test**	**Params**	**Val**	**Test**
Variational LSTM (Gal and Ghahramani, 2016)	–	–	–	20 M	–	78.6
CharCNN (Kim et al., 2016)	–	–	–	19 M	–	78.9
Pointer Sentinel-LSTM (Merity et al., 2017)	–	–	–	19 M	72.4	70.9
RHN (Zilly et al., 2016)	–	–	–	23 M	67.9	65.4
NAS Cell (Zoph and Le, 2017)	–	–	–	25 M	–	64.0
Variational LSTM - (Inan et al., 2017)	28 M	91.5	87	24 M	75.7	73.2
SRU - 6 layers (Lei et al., 2018)	–	–	–	24 M	63.4	60.3
QRNN (Bradbury et al., 2017)	–	–	–	**18 M**	82.1	78.3
RAN (Lee et al., 2017)	–	–	–	22 M	–	78.5
4-layer skip-connection LSTM (Melis et al., 2018)	–	–	–	24 M	60.9	58.3
AWD-LSTM - (Merity et al., 2018)	33 M	69.1	66	24 M	60.7	58.8
AWD-LSTM - (Merity et al., 2018)-finetuned	33 M	68.6	65.8	24 M	60	57.3
Variational LSTM (Gal and Ghahramani, 2016)	–	–	–	66 M	–	73.4
NAS Cell (Zoph and Le, 2017)	–	–	–	54 M	–	62.4
Quantized LSTM - Full precision (Xu et al., 2018)	–	–	100.1	–	–	89.8
Quantized LSTM - 2 bit (Xu et al., 2018)	–	–	106.1	–	–	95.8
With standard dropout						
LSTM ($M = 1000$)	29 M	78.93	75.08	20 M	68.57	66.29
LSTM ($M = 1200$)	35 M	77.93	74.48	26 M	69.17	67.16
LSTM ($M = 1400$)	42 M	77.55	74.44	33 M	70.88	68.55
Ours -PRU ($g = 1, K = 2, M = 1000$)	28 M	79.15	76.59	19 M	69.8	67.78
Ours -PRU ($g = 2, K = 2, M = 1200$)	28 M	76.62	73.79	19 M	67.17	64.92
Ours -PRU ($g = 4, K = 2, M = 1400$)	28 M	75.46	72.77	19 M	64.76	62.42
With advanced dropouts						
Ours - AWD-PRU ($g = 1, K = 2, M = 1000$)	28 M	71.84	68.6	19 M	61.72	59.54
Ours - AWD-PRU ($g = 2, K = 2, M = 1200$)	28 M	68.57	65.7	19 M	60.81	58.65
Ours - AWD-PRU ($g = 4, K = 2, M = 1400$)	28 M	68.17	65.3	19 M	60.62	58.33
Ours - AWD-PRU ($g = 4, K = 2, M = 1400$)-finetuned	**28 M**	**67.19**	**64.53**	19 M	**58.46**	**56.56**

Table 1: Comparison of single model word-level perplexity of our model with state-of-the-art on validation and test sets of Penn Treebank and Wikitext-2 dataset. For evaluation, we select the model with minimum validation loss. Lower perplexity value represents better performance.

Confidence: As exemplified in Table 2a, the PRU tends toward more confident decisions, placing more of the probability mass on the top next-word prediction than the LSTM. To quantify this effect, we calculate the entropy of the next-token distribution for both the PRU and the LSTM using 3687 contexts from the PTB validation set. Figure 3 shows a histogram of the entropies of the distribution, where bins of size 0.23 are used to effect categories. We see that the PRU more often produces lower entropy distributions corresponding to higher confidences for next-token choices. This is evidenced by the mass of the red PRU curve lying in the lower entropy ranges compared to the blue LSTM's curve. The PRU can produce confident decisions in part because more information is encoded in the higher dimensional context vectors.

Variance in word embeddings: The PRU has the ability to model individual words at different resolutions through the pyramidal transform; which provides multiple paths for the gradient to the embedding layer (similar to multi-task learning) and improves the flow of information. When considering the embeddings by part of speech, we find that the pyramid level 1 embeddings exhibit higher variance than the LSTM across all POS categories (Figure 4), and that pyramid level 2 embeddings show extremely low variance[1]. We hypothesize that the LSTM must encode both coarse group similarities and individual word differences into the same vector space, reducing the space between individual words of the same category. The PRU can rely on the subsampled embeddings to

---

[1] POS categories are computed using NLTK toolkit.

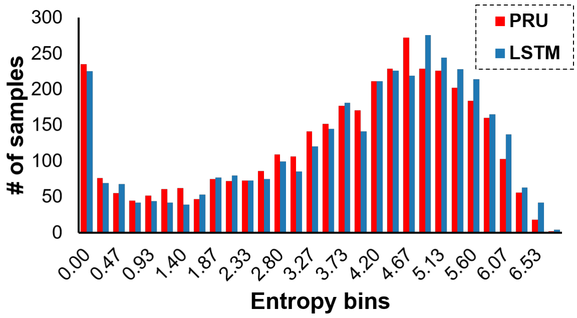

Figure 3: Histogram of the entropies of next-token distributions predicted by the PRU (mean 3.80) and the LSTM (mean 3.93) on the PTB validation set. Lower entropy values indicate higher confidence decisions, which is desirable if decisions are often correct.

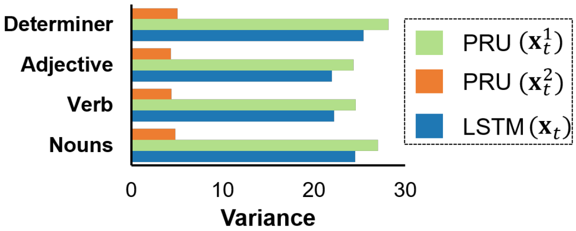

Figure 4: Variance of learned word embeddings for different categories of words on the PTB validation set. We compute the variance of a group of embeddings as the average squared euclidean distance to their mean. Higher variance may allow for better intra-category distinctions. The PRU with pyramid levels 1 and 2 is shown.

account for coarse-grained group similarities, allowing for finer individual word distinctions in the embedding layer. This hypothesis is strengthened by the entropy results described above: a model which can make finer distinctions between individual words can more confidently assign probability mass. A model that cannot make these distinctions, such as the LSTM, must spread its probability mass across a larger class of similar words.

Gradient-based analysis: Saliency analysis using gradients help identify relevant words in a test sequence that contribute to the prediction (Gevrey et al., 2003; Li et al., 2016; Arras et al., 2017). These approaches compute the relevance as the squared norm of the gradients obtained through back-propagation. Table 2a visualizes the heatmaps for different sequences. PRUs, in general, give more relevance to contextual words than LSTMs, such as southeast (sample 1), cost (sample 2), face (sample 4), and introduced (sample 5), which help in making more confident decisions. Furthermore, when gradients during back-propagation are visualized (Selvaraju et al., 2017) (Table 2b), we find that PRUs have better gradient coverage than LSTMs, suggesting PRUs use more features than LSTMs that contributes to the decision. This also suggests that PRUs update more parameters at each iteration which results in faster training. Language model in (Merity et al., 2018) takes 500 and 750 epochs to converge with PRU and LSTM as a recurrent unit, respectively.

4.4 Ablation studies

In this section, we provide a systematic analysis of our design choices. Our training methodology is the same as described in Section 4.1 with the standard dropouts. For a thorough understanding of our design choices, we use a language model with a single layer of PRU and fix the size of embedding and hidden layers to 600. The word-level perplexities are reported on the validation sets of the PTB and the WT-2 datasets.

Pyramidal levels K and groups g: The two hyper-parameters that control the trade-off between performance and number of parameters in PRUs are the number of pyramidal levels K and groups g. Figure 5 provides a trade-off between perplexity and recurrent unit (RU) parameters[2].

Variable K and fixed g: When we increase the number of pyramidal levels K at a fixed value of g, the performance of the PRU drops by about 1 to 4 points while reducing the total number of recurrent unit parameters by up to 15%. We note that the PRU with $K = 4$ at $g = 1$ delivers similar performance as the LSTM while learning about 15% fewer recurrent unit parameters.

Fixed K and variable g: When we vary the value of g at fixed number of pyramidal levels K, the total number of recurrent unit parameters decreases significantly with a minimal impact on the perplexity. For example, PRUs with $K = 2$ and $g = 4$ learns 77% fewer recurrent unit parameters while its perplexity (lower is better) increases by about 12% in comparison to LSTMs. Moreover, the decrease in number of parameters at higher value of g enables PRUs to learn the representations in high dimensional space with better generalizability (Table 1).

---

[2] # total params = # embedding params + # RU params

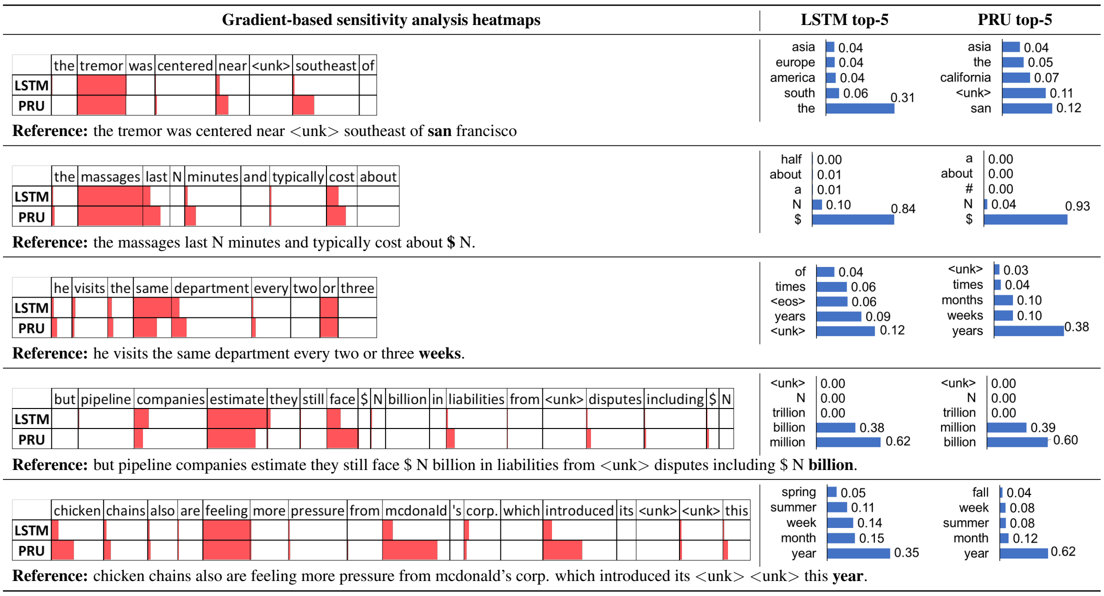

Reference: the tremor was centered near <unk> southeast of **san** francisco

Reference: the massages last N minutes and typically cost about **$** N.

Reference: he visits the same department every two or three **weeks**.

Reference: but pipeline companies estimate they still face $ N billion in liabilities from <unk> disputes including $ N **billion**.

Reference: chicken chains also are feeling more pressure from mcdonald's corp. which introduced its <unk> <unk> this **year**.

(a) Gradient-based saliency analysis. Salience score is proportional to cell coverage in red.

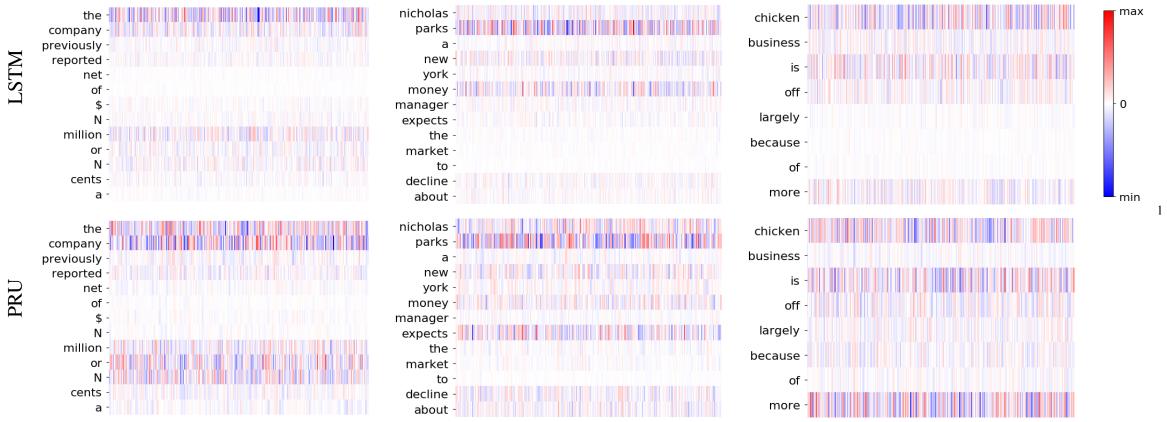

(b) Gradients during back-propagation for a test sequence (x-axis: dimensions of word vector, y-axis: test sequence)

Table 2: Qualitative comparison between the LSTM and the PRU: (a) Gradient-based saliency analysis along with top-5 predicted words. (b) Gradients during back-propagation. For computing the gradients for a given test sequence, the top-1 predicted word was used as the *true* predicted word. Best viewed in color.

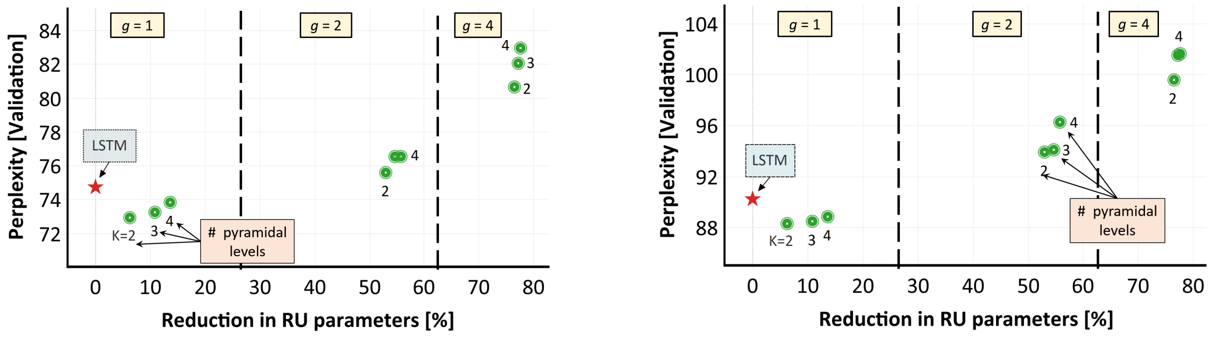

(a) PTB

(b) WT-2

Figure 5: Impact of number of groups g and pyramidal levels K on the perplexity. Reduction in recurrent unit (RU) parameters is computed with respect to LSTM. Lower perplexity value represents better performance.

Transformations: Table 3 shows the impact of different transformations of the input vector $\mathbf{x}_t$ and the context vector $\mathbf{h}_{t-1}$. We make following observations: (1) Using the pyramidal transformation for the input vectors improves the perplexity by about 1 point on both the PTB and WT-2 datasets while reducing the number of recurrent unit parameters by about 14% (see R1 and R4). We note that the performance of the PRU drops by up to 1 point when residual connections are not used (R4 and R6). (2) Using the grouped linear transformation for context vectors reduces the total number of recurrent unit parameters by about 75% while the performance drops by about 11% (see R3 and R4). When we use the pyramidal transformation instead of the linear transformation, the performance drops by up to 2% while there is no significant drop in the number of parameters (R4 and R5).

Subsampling: We set sub-sampling kernel κ (Eq. 3) with stride $s = 2$ and size of 3 ($e = 1$) in four different ways: (1) *Skip:* We skip every other element in the input vector. (2) *Convolution:* We initialize the elements of κ randomly from normal distribution and learn them during training the model. We limit the output values between -1 and 1 using $tanh$ activation function to make training stable. (3) *Avg. pool:* We initialize the elements of κ to $\frac{1}{3}$. (4) *Max pool:* We select the maximum value in the kernel window κ.

Table 4 compares the performance of the PRU with different sampling methods. Average pooling performs the best while skipping give comparable performance. Both of these methods enable the network to learn richer word representations while representing the input vector in different forms, thus delivering higher performance. Surprisingly, a convolution-based sub-sampling method does not perform as well as the averaging method. The $tanh$ function used after convolution limits the range of output values which are further limited by the LSTM gating structure, thereby impeding in the flow of information inside the cell. Max pooling forces the network to learn representations from high magnitude elements, thus distinguishing features between elements vanishes, resulting in poor performance.

	Transformations		**PTB**		**WT-2**	
	Context	Input	**PPL**	**# Params** (total/RU)	**PPL**	**# Params** (total/RU)
R1	LT	LT	74.80	8.8/2.9	89.30	22.8/2.9
R2	GLT	GLT	84.38	**6.5/0.5**	104.13	**20.46/0.5**
R3	GLT	PT	82.67	6.6/0.64	99.57	20.6/0.64
R4	LT	PT	**74.18**	8.5/2.5	**88.31**	22.5/2.5
R5	PT	PT	75.80	8.1/2.1	90.56	22.1/2.1
R6	LT	PT†	75.61	8.5/2.5	89.27	22.5/2.5

Table 3: Impact of different transformations used for processing input and context vectors (LT - linear transformation, PT - pyramidal transformation, and GLT - grouped linear transformation). Here, † represents that PT was used without residual connection, PPL represents word-level perplexity (lower is better), and the number of parameters are in million. We used K=g=4 in our experiments.

Dataset	**Skip**	**Max pool**	**Avg. Pool**	**Convolution**
PTB	75.12	87.6	**73.86**	81.56
WT-2	89.24	107.63	**88.88**	93.16

Table 4: Impact of different sub-sampling methods on the word-level perplexity (lower is better). We used g=1 and K=4 in our experiments.

5 Conclusion

We introduce the Pyramidal Recurrent Unit, which better model contextual information by admitting higher dimensional representations with good generalizability. When applied to the task of language modeling, PRUs improve perplexity across several settings, including recent state-of-the-art systems. Our analysis shows that the PRU improves the flow of gradient and expand the word embedding subspace, resulting in more confident decisions. Here we have shown improvements for language modeling. In future, we plan to study the performance of PRUs on different tasks, including machine translation and question answering. In addition, we will study the performance of the PRU on language modeling with more recent inference techniques, such as dynamic evaluation and mixture of softmax.

Acknowledgments

This research was supported by NSF (IIS 1616112, III 1703166), Allen Distinguished Investigator Award, and gifts from Allen Institute for AI, Google, Amazon, and Bloomberg. We are grateful to Aaron Jaech, Hannah Rashkin, Mandar Joshi, Aniruddha Kembhavi, and anonymous reviewers for their helpful comments.

References

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. In *8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis,*.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural networks. In *International Conference for Learning Representations (ICLR)*.

Peter J Burt and Edward H Adelson. 1987. The laplacian pyramid as a compact image code. In *Readings in Computer Vision*. Elsevier.

Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. 2018. Skip rnn: Learning to skip state updates in recurrent neural networks. In *International Conference for Learning Representations (ICLR)*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems (NIPS)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*.

Felix A Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*.

Muriel Gevrey, Ioannis Dimopoulos, and Sovan Lek. 2003. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological modelling*.

Dongyoon Han, Jiwhan Kim, and Junmo Kim. 2017. Deep pyramidal residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference for Learning Representations (ICLR)*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Diederik P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems (NIPS)*.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2018. Dynamic evaluation of neural sequence models. In *International Conference on Machine Learning (ICML)*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*.

Oleksii Kuchaiev and Boris Ginsburg. 2017. Factorization tricks for lstm networks. In *International Conference for Learning Representations (ICLR) Workshop*.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. *arXiv preprint arXiv:1705.07393*.

Tao Lei, Yu Zhang, and Yoav Artzi. 2018. Training rnns as fast as cnns. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

David G Lowe. 1999. Object recognition from local scale-invariant features. In *IEEE international conference on Computer vision (ICCV)*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*.

Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. 2018. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *European Conference in Computer Vision (ECCV)*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *International Conference for Learning Representations (ICLR)*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. In *International Conference for Learning Representations (ICLR)*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference for Learning Representations (ICLR)*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision (ICCV)*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. In *International Conference on Learning Representations (ICLR)*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Neural speed reading via skim-rnn. In *International Conference on Learning Representations (ICLR)*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*.

Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. 2018. Alternating multi-bit quantization for recurrent neural networks. In *International Conference for Learning Representations (ICLR)*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2018. Breaking the softmax bottleneck: a high-rank rnn language model. In *International Conference for Learning Representations (ICLR)*.

Adams Wei Yu, Hongrae Lee, and Quoc Le. 2017. Learning to skim text. In *Association for Computational Linguistics (ACL)*.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*.

Barret Zoph and Quoc V Le. 2017. Neural architecture search with reinforcement learning. In *International Conference for Learning Representations (ICLR)*.

On Tree-Based Neural Sentence Modeling

Haoyue Shi †,‡,* **Hao Zhou**‡ **Jiaze Chen**‡ **Lei Li**‡
†: School of EECS, Peking University, Beijing, China
`hyshi@pku.edu.cn`
‡: ByteDance AI Lab, Beijing, China
`{zhouhao.nlp, chenjiaze, lileilab}@bytedance.com`

Abstract

Neural networks with tree-based sentence encoders have shown better results on many downstream tasks. Most of existing tree-based encoders adopt syntactic parsing trees as the explicit structure prior. To study the effectiveness of different tree structures, we replace the parsing trees with trivial trees (*i.e.*, binary balanced tree, left-branching tree and right-branching tree) in the encoders. Though trivial trees contain no syntactic information, those encoders get competitive or even better results on all of the ten downstream tasks we investigated. This surprising result indicates that explicit syntax guidance may not be the main contributor to the superior performances of tree-based neural sentence modeling. Further analysis show that tree modeling gives better results when crucial words are closer to the final representation. Additional experiments give more clues on how to design an effective tree-based encoder. Our code is open-source and available at `https://github.com/ExplorerFreda/TreeEnc`.

1 Introduction

Sentence modeling is a crucial problem in natural language processing (NLP). Recurrent neural networks with long short term memory (Hochreiter and Schmidhuber, 1997) or gated recurrent units (Cho et al., 2014) are commonly used sentence modeling approaches. These models embed sentences into a vector space and the resulting vectors can be used for classification or sequence generation in the downstream tasks.

In addition to the plain sequence of hidden units, recent work on sequence modeling proposes to impose tree structure in the encoder (Socher et al., 2013; Tai et al., 2015; Zhu et al., 2015). These tree-based LSTMs introduce syntax tree as an intuitive structure prior for sentence modeling. They have already obtained promising results in many NLP tasks, such as natural language inference (Bowman et al., 2016; Chen et al., 2017c) and machine translation (Eriguchi et al., 2016; Chen et al., 2017a,b; Zhou et al., 2017). Li et al. (2015) empirically concludes that syntax tree-based sentence modeling are effective for tasks requiring relative long-term context features.

On the other hand, some works propose to abandon the syntax tree but to adopt the latent tree for sentence modeling (Choi et al., 2018; Yogatama et al., 2017; Maillard et al., 2017; Williams et al., 2018). Such latent trees are directly learned from the downstream task with reinforcement learning (Williams, 1992) or Gumbel Softmax (Jang et al., 2017; Maddison et al., 2017). However, Williams et al. (2018) empirically show that, Gumbel softmax produces unstable latent trees with the same hyper-parameters but different initializations, while reinforcement learning (Williams et al., 2018) even tends to generate left-branching trees. Neither gives meaningful latent trees in syntax, but each method still obtains considerable improvements in performance. This indicates that syntax may not be the main contributor to the performance gains.

With the above observation, we bring up the following questions: What does matter in tree-based sentence modeling? If tree structures are necessary in encoding the sentences, what mostly contributes to the improvement in downstream tasks? We attempt to investigate the driving force of the improvement by latent trees without syntax.

In this paper, we empirically study the effectiveness of tree structures in sentence modeling. We compare the performance of bi-LSTM and five tree LSTM encoders with different tree layouts, including the syntax tree, latent tree (from Gum-

---

[*] Now at Toyota Technological Institute at Chicago, `freda@ttic.edu`. This work was done when HS was an intern researcher at ByteDance AI Lab.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4631–4641
Brussels, Belgium, October 31 - November 4, 2018.

bel softmax) and three kinds of designed trivial trees (binary balance tree, left-branching tree and right-branching tree). Experiments are conducted on 10 different tasks, which are grouped into three categories, namely the single sentence classification (5 tasks), sentence relation classification (2 tasks), and sentence generation (3 tasks). These tasks depend on different granularities of features, and the comparison among them can help us learn more about the results. We repeat all the experiments 5 times and take the average to avoid the instability caused by random initialization of deep learning models.

We get the following conclusions:

- Tree structures are helpful to sentence modeling on classification tasks, especially for tasks which need global (long-term) context features, which is consistent with previous findings (Li et al., 2015).
- Trivial trees outperform syntactic trees, indicating that syntax may not be the main contributor to the gains of tree encoding, at least on the ten tasks we investigate.
- Further experiments shows that, given strong priors, tree based methods give better results when crucial words are closer to the final representation. If structure priors are unavailable, balanced tree is a good choice, as it makes the path distances between word and sentence encoding to be roughly equal, and in such case, tree encoding can learn the crucial words itself more easily.

2 Experimental Framework

We show the applied encoder-classifier/decoder framework for each group of tasks in Figure 1. Our framework has two main components: the encoder part and the classifier/decoder part. In general, models encode a sentence to a length-fixed vector, and then applies the vector as the feature for classification and generation.

We fix the structure of the classifier/decoder, and propose to use five different types of tree structures for the encoder part including:

- Parsing tree. We apply binary constituency tree as the representative, which is widely used in natural language inference (Bowman et al., 2016) and machine translation (Eriguchi et al., 2016; Chen et al., 2017a). Dependency parsing trees (Zhou et al., 2015, 2016a) are not considered in this paper.

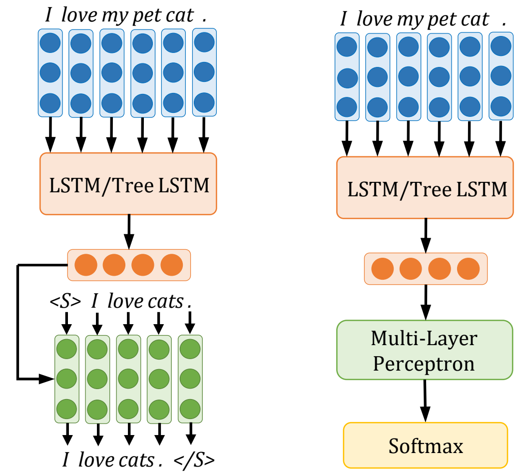

(a) Encoder-decoder framework for sentence generation.

(b) Encoder-classifier framework for sentence classification.

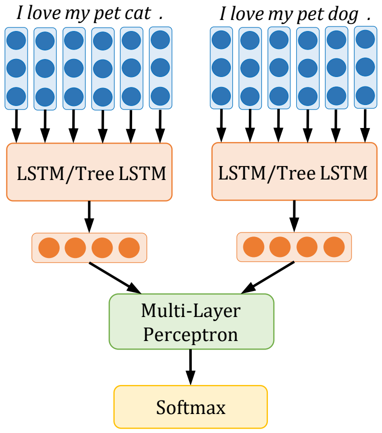

(c) Siamese encoder-classifier framework for sentence relation classification.

Figure 1: The encoder-classifier/decoder framework for three different groups of tasks. We apply multi-layer perceptron (MLP) for classification, and left-to-right decoders for generation in all experiments.

- Binary balanced tree. To construct a binary balanced tree, we recursively divide a group of n leafs into two contiguous groups with the size of $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$, until each group has only one leaf node left.
- Gumbel trees, which are produced by straight-forward Gumbel softmax models (Choi et al., 2018). Note that Gumbel trees are not stable to sentences (Williams et al., 2018), and we only draw a sample among all of them.
- Left-branching trees. We combine two nodes from left to right, to construct a left-branching tree, which is similar to those

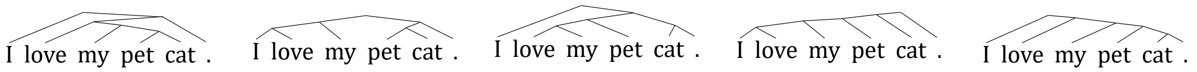

(a) Parsing tree. (b) Balanced tree. (c) Gumbel tree. (d) Left-branching tree. (e) Right-branching tree.

Figure 2: Examples of different tree structures for the encoder part.

generated by the reinforce based RL-SPINN model (Williams et al., 2018).

- Right-branching trees. In contrast to left-branching ones, nodes are combined from right to left to form a right-branching tree.

We show an intuitive view of the five types of tree structures in Figure 2. In addition, existing works (Choi et al., 2018; Williams et al., 2018) show that using hidden states of bidirectional RNNs as leaf node representations (bi-leaf-RNN) instead of word embeddings may improve the performance of tree LSTMs, as leaf RNNs help encode context information more completely. Our framework also support leaf RNNs for tree LSTMs.

3 Description of Investigated Tasks

We conduct experiments on 10 different tasks, which are grouped into 3 categories, namely the single sentence classification (5 tasks), sentence relation classification (2 tasks), and sentence generation (3 tasks). Each of the tasks is compatible to the encoder-classifier/decoder framework shown in Figure 1. These tasks cover a wide range of NLP applications, and depend on different granularities of features.

Note that the datasets may use articles or paragraphs as instances, some of which consist of only one sentence. For each dataset, we only pick the subset of single-sentence instances for our experiments, and the detailed meta-data is in Table 1.

3.1 Sentence Classification

First, we introduce four text classification datasets from Zhang et al. (2015), including AG's News, Amazon Review Polarity , Amazon Review Full and DBpedia. Additionally, noticing that parsing tree was shown to be effective (Li et al., 2015) on the task of word-level semantic relation classification (Hendrickx et al., 2009), we also add this dataset to our selections.

AG's News (AGN). Each sample in this dataset is an article, associated with a label indicating its topic: world, sports, business or sci/tech.

Amazon Review Polarity (ARP). The Amazon Review dataset is obtained from the Stanford Network Analysis Project (SNAP; McAuley and Leskovec, 2013). It collects a large amount of product reviews as paragraphs, associated with a star rate from 1 (most negative) to 5 (most positive). In this dataset, 3-star reviews are dropped, while others are classified into two groups: positive (4 or 5 stars) and negative (1 or 2 stars).

Amazon Review Full (ARF). Similar to the ARP dataset, the ARF dataset is also collected from Amazon product reviews. Labels in this dataset are integers from 1 to 5.

DBpedia. DBpedia is a crowd-sourced community effort to extract structured information from Wikipedia (Lehmann et al., 2015). Zhang et al. (2015) select 14 non-overlapping classes from DBpedia 2014 to construct this dataset. Each sample is given by the title and abstract of the Wikipedia article, associated with the class label.

Word-Level Semantic Relation (WSR) SemEval-2010 Task 8 (Hendrickx et al., 2009) is to find semantic relationships between pairs of nominals. Each sample is given by a sentence, of which two nominals are explicitly indicated, associated with manually labeled semantic relation between the two nominals. For example, the sentence "*My [apartment]*$_{e_1}$ *has a pretty large [kitchen]*$_{e_2}$ *.*" has the label *component-whole*(e_2, e_1). Different from retrieving the path between two labels (Li et al., 2015; Socher et al., 2013), we feed the entire sentence together with the nominal indicators (*i.e.*, tags of e_1 and e_2) as words to the framework. We also ignore the order of e_1 and e_2 in the labels given by the dataset. Thus, this task turns to be a 10-way classification one.

3.2 Sentence Relation Classification

To evaluate how well a model can capture semantic relation between sentences, we introduce the second group of tasks: sentence relation classification.

Natural Language Inference (NLI). The Stanford Natural Language Inference (SNLI) Corpus (Bowman et al., 2015) is a challenging dataset for sentence-level textual entailment. It has 550K training sentence pairs, as well as 10K for development and 10K for test. Each pair consists of two relative sentences, associated with a label which is one of *entailment, contradiction* and *neutral*.

Conjunction Prediction (Conj). Information about the coherence relation between two sentences is sometimes apparent in the text explicitly (Miltsakaki et al., 2004): this is the case whenever the second sentence starts with a conjunction phrase. Jernite et al. (2017) propose a method to create *conjunction prediction* dataset from unlabeled corpus. They create a list of phrases, which can be classified into nine types, as conjunction indicators. The object of this task is to recover the conjunction type of given two sentences, which can be used to evaluate how well a model captures the semantic meaning of sentences. We apply the method proposed by Jernite et al. (2017) on the Wikipedia corpus to create our conj dataset.

3.3 Sentence Generation

We also include the sentence generation tasks in our experiments, to investigate the representation ability of different encoders over global (long-term) context features. Note that our framework is based on encoding, which is different from those attention based approaches.

Paraphrasing (Para). Quora Question Pair Dataset is a widely applied dataset to evaluate paraphrasing models (Wang et al., 2017; Li et al., 2017b). [1] In this work, we treat the paraphrasing task as a sequence-to-sequence one, and evaluate on it with our sentence generation framework.

Machine Translation (MT). Machine translation, especially cross-language-family machine translation, is a complex task, which requires models to capture the semantic meanings of sentences well. We apply a large challenging English-Chinese sentence translation task for this investigation, which is adopted by a variety of neural translation work (Tu et al., 2016; Li et al., 2017a; Chen et al., 2017a). We extract the parallel data from the LDC corpora,[2] selecting 1.2M from them as our training set, 20K and 80K of them as our development set and test set, respectively.

Dataset	**#Sentence** Train	 Dev	 Test	**#Cls**	**Avg. Len**
Sentence Classification					
News	60K	6.7K	4.3K	4	31.5
ARP	128K	14K	16K	2	33.7
ARF	110K	12K	27K	5	33.8
DBpedia	106K	11K	15K	14	20.1
WSR	7.1K	891	2.7K	10	23.1
Sentence Relation					
SNLI	550K	10K	10K	3	11.2
Conj	552K	10K	10K	9	23.3
Sentence Generation					
Para	98K	2K	3K	N/A	10.2
MT	1.2M	20K	80K	N/A	34.1
AE	1.2M	20K	80K	N/A	34.1

Table 1: Meta-data of the downstream tasks we investigated. For each task, we list the quantity of instances in train/dev/test set, the average length (by words) of sentences (source sentence only for generation task), as well as the number of classes if applicable.

Auto-Encoding (AE). We extract the English part of the machine translation dataset to form a auto-encoding task, which is also compatible with our encoder-decoder framework.

4 Experiments

In this section, we present our experimental results and analysis. Section 4.1 introduces our set-up for all the experiments. Section 4.2 shows the main results and analysis on ten downstream tasks grouped into three classes, which can cover a wide range of NLP applications. Regarding that trivial tree based LSTMs perform the best among all models, we draw two hypotheses, which are i) right-branching tree benefits a lot from strong structural priors; ii) balanced tree wins because it fairly treats all words so that crucial information could be more easily learned by the LSTM gates automatically. We test the hypotheses in

---

[1] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[2] The corpora includes LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

	Sentence Classification					*Sentence Relation*		*Sentence Generation*		
Model	**AGN**	**ARP**	**ARF**	**DBpedia**	**WSR**	**NLI**	**Conj**	**Para**	**MT**	**AE**
Latent Trees										
Gumbel	91.8	87.1	48.4	98.6	66.7	80.4	51.2	20.4	17.4	39.5
+*bi-leaf-RNN*	91.8	**88.1**	**49.7**	98.7	69.2	**82.9**	53.7	20.5	22.3	75.3
(Constituency) Parsing Trees										
Parsing	91.9	87.5	49.4	**98.8**	66.6	81.3	52.4	19.9	19.1	44.3
+*bi-leaf-RNN*	92.0	88.0	49.6	**98.8**	68.6	82.8	53.4	20.4	22.2	72.9
Trivial Trees										
Balanced	92.0	87.7	49.1	98.7	66.2	81.1	52.1	19.7	19.0	49.4
+*bi-leaf-RNN*	**92.1**	87.8	**49.7**	**98.8**	**69.6**	82.6	**54.0**	20.5	22.3	76.0
Left-branching	91.9	87.6	48.5	98.7	67.8	81.3	50.9	19.9	19.2	48.0
+*bi-leaf-RNN*	91.2	87.6	48.9	98.6	67.7	82.8	53.3	20.6	21.6	72.9
Right-branching	91.9	87.7	49.0	**98.8**	68.6	81.0	51.3	20.4	19.7	54.7
+*bi-leaf-RNN*	91.9	87.9	49.4	98.7	68.7	82.8	53.5	**20.9**	**23.1**	**80.4**
Linear Structures										
LSTM	91.7	87.8	48.8	98.6	66.1	82.6	52.8	20.3	19.1	46.9
+*bidirectional*	91.7	87.8	49.2	98.7	67.4	82.8	53.3	20.2	21.3	67.0
Avg. Length	31.5	33.7	33.8	20.1	23.1	11.2	23.3	10.2	34.1	34.1

Table 2: Test results for different encoder architectures trained by a unified encoder-classifier/decoder framework. We report accuracy ($\times 100$) for classification tasks, and BLEU score (Papineni et al., 2002; word-level for English targets and char-level for Chinese targets) for generation tasks. Large is better for both of the metrics. The best number(s) for each task are in bold. In addition, average sentence length (in words) of each dataset is attached in the last row with underline.

Section 4.3. Finally, we compare the performance of linear and tree LSTMs with three widely applied pooling mechanisms in Section 4.4.

4.1 Set-up

In experiments, we fix the structure of the classifier as a two-layer MLP with ReLU activation, and the structure of decoder as GRU-based recurrent neural networks (Cho et al., 2014). [3] The hidden-layer size of MLP is fixed to 1024, while that of GRU is adapted from the size of sentence encoding. We initialize the word embeddings with 300-dimensional `GloVe` (Pennington et al., 2014) vectors.[4] We apply 300-dimensional bidirectional (600-dimensional in total) LSTM as leaf RNN when necessary. We use Adam (Kingma and Ba, 2015) optimizer to train all the models, with the learning rate of 1e-3 and batch size of 64. In the training stage, we drop the samples with the length of either source sentence or target sentence larger than 64. We do not apply any regularization or dropout term in all experiments except the task of WSR, on which we tune dropout term with respect to the development set. We generate the binary parsing tree for the datasets without parsing trees using ZPar (Zhang and Clark, 2011).[5] More details are summarized in supplementary materials.

4.2 Main Results

In this subsection, we aim to compare the results from different encoders. We do not include any attention (Wang et al., 2016; Lin et al., 2017) or pooling (Collobert and Weston, 2008; Socher et al., 2011; Zhou et al., 2016b) mechanism here, in order to avoid distractions and make the encoder structure affects the most. We will further analyze pooling mechanisms in Section 4.4.

Table 2 presents the performances of different

---

[3] We observe that ReLU can significantly boost the performance of Bi-LSTM on SNLI.

[4] `http://nlp.stanford.edu/data/glove.840B.300d.zip`

[5] `https://www.sutd.edu.sg/cmsresource/faculty/yuezhang/zpar.html`

encoders on a variety of downstream tasks, which lead to the following observations:

Tree encoders are useful on some tasks. We get the same conclusion with Li et al. (2015) that tree-based encoders perform better on tasks requiring long-term context features. Despiting the linear structured left-branching and right-branching tree encoders, we find that, tree-based encoders generally perform better than Bi-LSTMs on tasks of *sentence relation* and *sentence generation*, which may require relatively more long term context features for obtaining better performances. However, the improvements of tree encoders on **NLI** and **Para** are relatively small, which may be caused by that sentences of the two tasks are shorter than others, and the tree encoder does not get enough advantages to capture long-term context in short sentences.

Trivial tree encoders outperform other encoders. Surprisingly, binary balanced tree encoder gets the best results on most tasks of classification and right-branching tree encoder tends to be the best on sentence generation. Note that binary balanced tree and right-branching tree are only trivial tree structures, but outperform syntactic tree and latent tree encoders. The latent tree is really competitive on some tasks, as its structure is directly tuned by the corresponding tasks. However, it only beats the binary balanced tree by very small margins on NLI and ARP. We will give analysis about this in Section 4.3.

Larger quantity of parameters is not the only reason of the improvements. Table 2 shows that tree encoders benefit a lot from adding leaf-LSTM, which brings not only sentence level information to leaf nodes, but also more parameters than the bi-LSTM encoder. However, left-branching tree LSTM has a quite similar structure with linear LSTM, and it can be viewed as a linear LSTM-on-LSTM structure. It has the same amounts of parameters as other tree-based encoders, but still falls behind the balance tree encoder on most of the tasks. This indicates that larger quantity of parameters is at least not the only reason for binary balance tree LSTM encoders to gain improvements against bi-LSTMs.

4.3 Why Trivial Trees Work Better?

Binary balanced tree and right-branching are trivial ones, hardly containing syntax information. In this section, we analyze why these trees achieve high scores in deep.

4.3.1 Right Branching Tree Benefits from Strong Structural Prior

We argue that right-branching trees benefit from its strong structural prior. In sentence generation tasks, models generate sentences from left to right, which makes words in the left of the source sentence more important (Sutskever et al., 2014). If the encoder fails to memorize the left words, the information about right words would not help due to the error propagation. In right-branching trees, left words of the sentence are closer to the final representation, which makes the left words are more easy to be memorized, and we call this structure prior. Oppositely, in the case of left-branching trees, right words of the sentence are closer to the representation.

To validate our hypothesis, we propose to visualize the Jacobian as word-level saliency (Shi et al., 2018), which can be viewed as the contribution of each word to the sentence encoding:

$$\mathrm{J}(\mathbf{s},\mathbf{w}) = \|\nabla \mathbf{s}(\mathbf{w})\|_1 = \sum_{i,j} |\frac{\partial s_i}{\partial w_j}|$$

where $\mathbf{s} = (s_1, s_2, \cdots, s_p)^T$ denotes the embedding of a sentence, and $\mathbf{w} = (w_1, w_2, \cdots, w_q)^T$ denotes embedding of a word. We can compute the saliency score using backward propagation. For a word in a sentence, higher saliency score means more contribution to sentence encoding.

We present the visualization in Figure 3 using the visualization tool from Lin et al. (2017). It shows that right-branching tree LSTM encoders tend to look at the left part of the sentence, which is very helpful to the final generation performance, as left words are more crucial. Balanced trees also have this feature and we think it is because balance tree treats these words fairly, and crucial information could be more easily learned by the LSTM gates automatically.

However, bi-LSTM and left-branching tree LSTM also pay much attention to words in the right (especially the last two words), which maybe caused by the short path from the right words to the root representation, in the two corresponding tree structures.

Additionally, Table 3 shows that models trained with the same hyper-parameters but different initializations have strong agreement with each other.

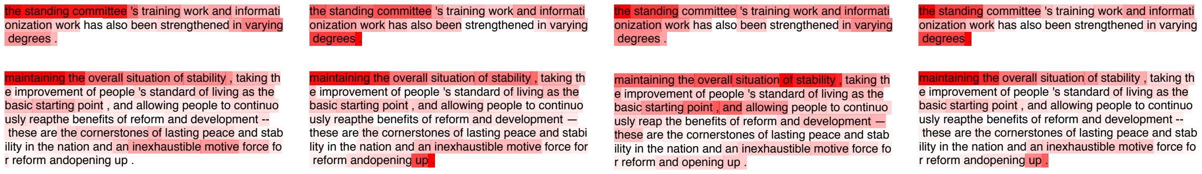

(a) Balanced tree, MT. (b) Left-branching tree, MT. (c) Right-branching, MT. (d) Bi-LSTM, MT.

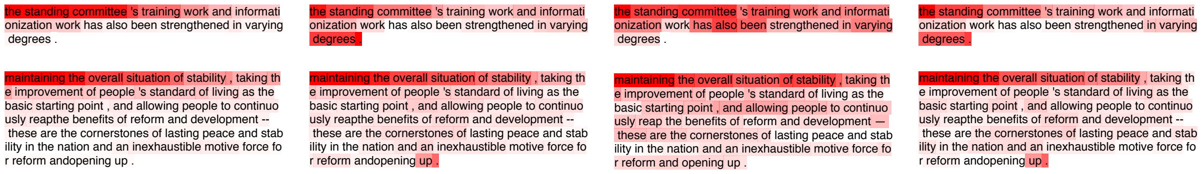

(e) Balanced tree, AE. (f) Left-branching tree, AE. (g) Right-branching, AE. (h) Bi-LSTM, AE.

Figure 3: Saliency visualization of words in learned MT and AE models. Darker means more important to the sentence encoding.

Model	**MT**	**AE**
Balanced (BiLRNN)	93.1	96.9
Left-Branching (BiLRNN)	94.2	95.4
Right-Branching (BiLRNN)	92.3	95.1
Bi-LSTM	96.4	96.1

Table 3: Mean average Pearson correlation $(\times 100)$ across five models trained with same hyper-parameters. For each testing sentence, we compute the saliency scores of words. Cross-model Pearson correlation can show the agreement of two models on one sentence, and average Pearson correlation is computed through all sentences. We report mean average Pearson correlation of the 5×4 model pairs.

Thus, "looking at the first words" is a stable behavior of balanced and right-branching tree LSTM encoders in sentence generation tasks. So is "looking at the first and the last words" for Bi-LSTMs and left-branching tree LSTMs.

4.3.2 Binary Balanced Tree Benefits from Shallowness

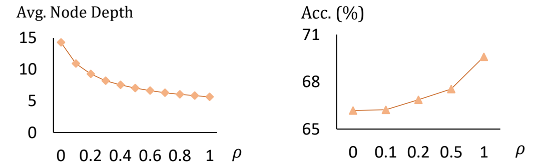

(a) ρ-depth line for WSR. (b) ρ-Acc. line for WSR.

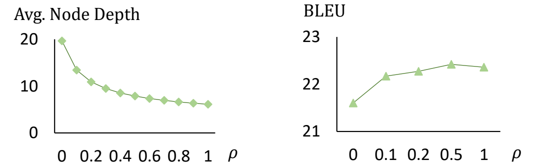

(c) ρ-depth line for MT. (d) ρ-BLEU line for MT.

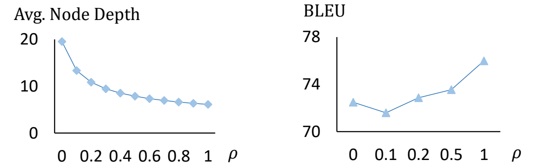

(e) ρ-depth line for AE. (f) ρ-BLEU line for AE.

Figure 4: ρ-depth and ρ-performance lines for three tasks. There is a trend that the depth drops and the performance raises with the growth of ρ.

Compared to syntactic and latent trees, the only advantage of balanced tree we can hypothesize is that, it is shallower and more balanced than others. Shallowness may lead to shorter path for information propagation from leafs to the root representation, and makes the representation learning more easy due to the reduction of errors in the propagation process. Balance makes the tree fairly treats all leaf nodes, which makes it more easily to automatically select the crucial information over all words in a sentence.

To test our hypothesis, we conduct the following experiments. We select three tasks, on which binary balanced tree encoder wins Bi-LSTMs with a large margin (WSR, MT and AE). We generate random binary trees for sentences, while con-

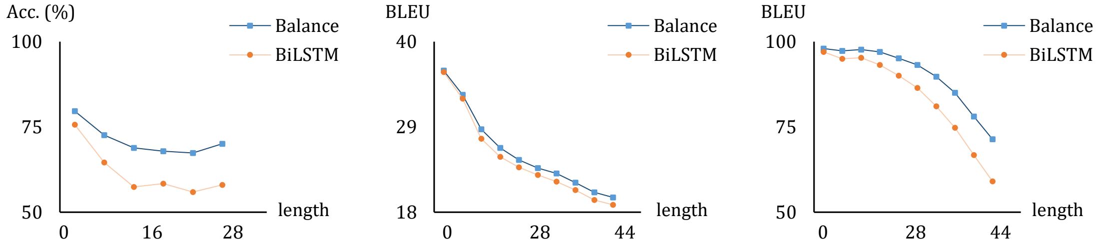

(a) Length-Accuracy lines for WSR. (b) Length-BLEU lines for MT. (c) Length-BLEU lines for AE.

Figure 5: Length-performance lines for the further investigated tasks. We divide test instances into several groups by length, and report the performance on each group respectively. Sentences with length in $[1, 8]$ are put to the first group, and the group $i (i \geq 2)$ covers the range of $[4i + 1, 4i + 4]$ in length.]

trolling the depth using a hyper-parameter ρ. We start by a group with all words (nodes) in the sentence. At each time, we separate n nodes to two continuous groups sized $(\lceil \frac{n}{2} \rceil, \lfloor \frac{n}{2} \rfloor)$ with probability ρ, while those sized $(n - 1, 1)$ with probability $1 - \rho$. Trees generated with $\rho = 0$ are exactly left-branching trees, and those generated with $\rho = 1$ are binary balanced trees. The expected node depth of the tree turns smaller with ρ varies from 0 to 1.

Figure 4 shows that, in general, trees with shallower node depth have better performance on all of the three tasks (for binary tree, shallower also means more balanced), which validates our above hypothesis that binary balanced tree gains the reward from its shallow and balanced structures.

Additionally, Figure 5 demonstrates that binary balanced trees work especially better with relative long sentences. As desired, on short-sentence groups, the performance gap between Bi-LSTM and binary balanced tree LSTM is not obvious, while it grows with the test sentences turning longer. This explains why tree-based encoder gives small improvements on NLI and Para, because sentences on these two tasks are much shorter than others.

4.4 Can Pooling Replace Tree Encoder?

Max pooling (Collobert and Weston, 2008; Zhao et al., 2015), mean pooling (Conneau et al., 2017) and self-attentive pooling (also known as self-attention; Santos et al., 2016; Liu et al., 2016; Lin et al., 2017) are three popular and efficient choices to improve sentence encoding. In this part, we will compare the performance of tree LSTMs and bi-LSTM on the tasks of WSR, MT and AE, with each pooling mechanism respectively, aiming to demonstrate the role that pooling plays in sentence modeling, and validate whether tree encoders can be replaced by pooling.

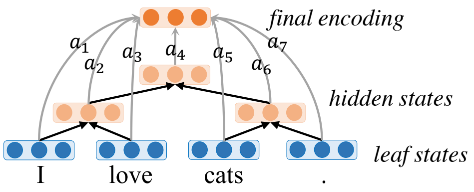

(a) Balanced tree.

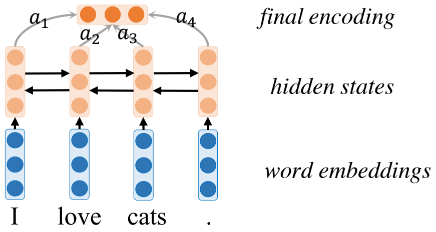

(b) Bi-LSTM.

Figure 6: An illustration of the investigated self-attentive pooling mechanism.

As shown in Figure 6, for linear LSTMs, we apply pooling mechanism to all hidden states; as for tree LSTMs, pooling is applied to all hidden states and leaf states of tree LSTMs. Implementation details are summarized in the supplementary materials.

Table 4 shows that max and attentive pooling improve all the structures on the task of WSR, but all the pooling mechanisms fail on MT and AE that require the encoding to capture complete information of sentences, while pooling mechanism may cause the loss of information through the procedure. The result indicates that, though pooling mechanism is efficient on some tasks, it cannot totally gain the advantages brought by tree structures. Additionally, we think the attention mech-

Model	WSR	MT	AE
Bi-LSTM	67.4	21.3	67.0
+*max-pooling*	71.8 ↑	21.6 ↑	48.0 ↓
+*mean-pooling*	64.3 ↓	21.8 ↑	47.8 ↓
+*self-attention*	**72.5** ↑	21.2 ↓	60.4 ↓
Parsing (BiLRNN)	68.6	22.2	72.9
+*max-pooling*	69.7 ↑	21.8 ↓	48.3 ↓
+*mean-pooling*	58.0 ↓	21.2 ↓	50.7 ↓
+*self-attention*	72.2 ↑	21.5 ↓	69.1↓
Balanced (BiLRNN)	69.6	22.3	76.0
+*max-pooling*	70.6 ↑	21.6 ↓	48.5 ↓
+*mean-pooling*	54.1 ↓	21.3 ↓	52.7 ↓
+*self-attention*	**72.5** ↑	21.6 ↓	69.5 ↓
Left (BiLRNN)	67.7	21.6	72.9
+*max-pooling*	71.2 ↑	20.5 ↓	47.6 ↓
+*mean-pooling*	67.3 ↓	21.4 ↓	51.8 ↓
+*self-attention*	72.1 ↑	21.6 –	70.2 ↓
Right (BiLRNN)	68.7	**23.1**	**80.4**
+*max-pooling*	71.6 ↑	21.6 ↓	48.4 ↓
+*mean-pooling*	67.2 ↓	22.1 ↓	53.9 ↓
+*self-attention*	72.4 ↑	21.6 ↓	68.9 ↓

Table 4: Performance of tree and linear-structured encoders with or without pooling, on the selected three tasks. We report accuracy ($\times 100$), char-level BLEU for MT and word-level BLEU for AE. All of the tree models have bidirectional leaf RNNs (BiLRNN). The best number(s) for each task are in bold. The top and down arrows indicate the increment or decrement of each pooling mechanism, against the baseline of pure tree based encoder with the same structure.

anism has the benefits of the balanced tree modeling, which also fairly treat all words and learn the crucial parts automatically. The path from representation to words in attention are even shorter than the balanced tree. Thus the fact that attentive pooling outperforms balanced trees on WSR is not surprising to us.

5 Discussions

Balanced tree for sentence modeling has been explored by Munkhdalai and Yu (2017) and Williams et al. (2018) in natural language inference (NLI). However, Munkhdalai and Yu (2017) focus on designing inter-attention on trees, instead of comparing balanced tree with other linguistic trees in the same setting. Williams et al. (2018) do compare balanced trees with latent trees, but balanced tree does not outperform the latent one in their experiments, which is consistent with ours. We analyze it in Section 4.2 that sentences in NLI are too short for the balanced tree to show the advantage.

Levy et al. (2018) argue that LSTM works for the gates ability to compute an element-wise weighted sum. In such case, tree LSTM can also be regarded as a special case of attention, especially for the balanced-tree modeling, which also automatically select the crucial information from all word representation. Kim et al. (2017) propose a tree structured attention networks, which combine the benefits of tree modeling and attention, and the tree structures in their model are also learned instead of the syntax trees.

Although binary parsing trees do not produce better numbers than trivial trees on many downstream tasks, it is still worth noting that we are not claiming the useless of parsing trees, which are intuitively reasonable for human language understanding. A recent work (Blevins et al., 2018) shows that RNN sentence encodings directly learned from downstream tasks can capture implicit syntax information. Their interesting result may explain why explicit syntactic guidance does not work for tree LSTMs. In summary, we still believe in the potential of linguistic features to improve neural sentence modeling, and we hope our investigation could give some sense to afterwards hypothetical exploring of designing more effective tree-based encoders.

6 Conclusions

In this work, we propose to empirically investigate what contributes mostly in the tree-based neural sentence encoding. We find that trivial trees without syntax surprisingly give better results, compared to the syntax tree and the latent tree. Further analysis indicates that the balanced tree gains from its shallow and balance properties compared to other trees, and right-branching tree benefits from its strong structural prior under the setting of left-to-right decoder.

Acknowledgements

We thank Hang Li, Yue Zhang, Lili Mou and Jiayuan Mao for their helpful comments on this work, and the anonymous reviewers for their valuable feedback.

References

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs Encode Soft Hierarchical Syntax. In *Proc. of ACL*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A Large Annotated Corpus for Learning Natural Language Inference. In *Proc. of EMNLP*.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A Fast Unified Model for Parsing and Sentence Understanding. In *Proc. of ACL*.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017a. Improved Neural Machine Translation with a Syntax-Aware Encoder and Decoder. In *Proc. of ACL*.

Kehai Chen, Rui Wang, Masao Utiyama, Lemao Liu, Akihiro Tamura, Eiichiro Sumita, and Tiejun Zhao. 2017b. Neural Machine Translation with Source Dependency Representation. In *Proc. of EMNLP*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017c. Enhanced LSTM for Natural Language Inference. In *Proc. of ACL*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proc. of EMNLP*.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to Compose Task-Specific Tree Structures. In *Proc. of AAAI*.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proc. of ICML*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proc. of EMNLP*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-Sequence Attentional Neural Machine Translation. In *Proc. of ACL*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *Proc. of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *Proc. of ICLR*.

Yacine Jernite, Samuel R. Bowman, and David Sontag. 2017. Discourse-based Objectives for Fast Unsupervised Sentence Representation Learning. *arXiv preprint arXiv:1705.00557*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured Attention Networks. In *Proc. of ICLR*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. DBpedia–A Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web*.

Omer Levy, Kenton Lee, Nicholas FitzGerald, and Luke Zettlemoyer. 2018. Long Short-Term Memory as a Dynamically Computed Element-wise Weighted Sum. In *Proc. of ACL*.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When Are Tree Structures Necessary for Deep Learning of Representations? In *Proc. of EMNLP*.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017a. Modeling Source Syntax for Neural Machine Translation. In *Proc. of ACL*.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2017b. Paraphrase Generation with Deep Reinforcement Learning. *arXiv preprint arXiv:1711.00279*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-Attentive Sentence Embedding. In *Proc. of ICLR*.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning Natural Language Inference using Bidirectional LSTM Model and Inner-Attention. *arXiv preprint arXiv:1605.09090*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proc. of ICLR*.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly Learning Sentence embeddings and Syntax with Unsupervised Tree-LSTMs. *arXiv preprint arXiv:1705.09189*.

Julian McAuley and Jure Leskovec. 2013. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *Proc. of the 7th ACM Conference on Recommender Systems*.

Eleni Miltsakaki, Rashmi Prasad, Aravind K. Joshi, and Bonnie L Webber. 2004. The Penn Discourse Treebank. In *Proc. of LREC*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural Tree Indexers for Text Understanding. In *Proc. of EACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proc. of EMNLP*.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive Pooling Networks. *arXiv preprint arXiv:1602.03609*.

Haoyue Shi, Jiayuan Mao, Tete Xiao, Yuning Jiang, and Jian Sun. 2018. Learning Visually-Grounded Semantics from Contrastive Adversarial Samples. In *Proc. of COLING*.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D. Manning, and Andrew Y. Ng. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Proc. of NIPS*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. In *Proc. of EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. of NIPS*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proc. of ACL-IJCNLP*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. In *Proc. of ACL*.

Yequan Wang, Minlie Huang, Li Zhao, and Xiaoyan Zhu. 2016. Attention-Based LSTM for Aspect-Level Sentiment Classification. In *Proc. of EMNLP*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. In *Proc. of IJCAI*.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do Latent Tree Learning Models Identify Meaningful Structure in Sentences? *Transaction of ACL*.

Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. In *Reinforcement Learning*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to Compose Words into Sentences with Reinforcement Learning. *Proc. of ICLR*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-Level Convolutional Networks for Text Classification. In *Proc. of NIPS*.

Yue Zhang and Stephen Clark. 2011. Syntactic Processing using the Generalized Perceptron and Beam Search. *Computational Linguistics*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-Adaptive Hierarchical Sentence Model. In *Proc. of IJCAI*.

Hao Zhou, Zhaopeng Tu, Shujian Huang, Xiaohua Liu, Hang Li, and Jiajun Chen. 2017. Chunk-Based Bi-Scale Decoder for Neural Machine Translation. In *Proc. of ACL*.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A Neural Probabilistic Structured-Prediction Model for Transition-based Dependency Parsing. In *Proc. of ACL*.

Hao Zhou, Yue Zhang, Shujian Huang, Junsheng Zhou, Xin-Yu Dai, and Jiajun Chen. 2016a. A Search-Based Dynamic Reranking Model for Dependency Parsing. In *Proc. of ACL*.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016b. Text Classification Improved by Integrating Bidirectional LSTM with Two-Dimensional Max Pooling. In *Proc. of COLING*.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long Short-Term Memory Over Recursive Structures. In *Proc. of ICML*.

Language Modeling with Sparse Product of Sememe Experts

Yihong Gu[1,2,*] **Jun Yan**[1,3,*] **Hao Zhu**[1,2,*] **Zhiyuan Liu**[1,2,†]
Ruobing Xie[4] **Maosong Sun**[1,2] **Fen Lin**[4] **Leyu Lin**[4]
[1]Institute for Artificial Intelligence
State Key Lab on Intelligent Technology and Systems
[2]Department of CST, [3]Department of EE, Tsinghua University, Beijing, China
[4]Search Product Center, WeChat Search Application Department, Tencent
`{gyh15,j-yan15,zhuhao15}@mails.tsinghua.edu.cn`,
`{lzy,sms}@tsinghua.edu.cn`, `xrbsnowing@163.com`,
`{felicialin,goshawklin}@tencent.com`

Abstract

Most language modeling methods rely on large-scale data to statistically learn the sequential patterns of words. In this paper, we argue that words are atomic language units but not necessarily atomic semantic units. Inspired by HowNet, we use sememes, the minimum semantic units in human languages, to represent the implicit semantics behind words for language modeling, named Sememe-Driven Language Model (SDLM). More specifically, to predict the next word, SDLM first estimates the sememe distribution given textual context. Afterwards, it regards each sememe as a distinct semantic expert, and these experts jointly identify the most probable senses and the corresponding word. In this way, SDLM enables language models to work beyond word-level manipulation to fine-grained sememe-level semantics, and offers us more powerful tools to fine-tune language models and improve the interpretability as well as the robustness of language models. Experiments on language modeling and the downstream application of headline generation demonstrate the significant effectiveness of SDLM. Source code and data used in the experiments can be accessed at `https://github.com/thunlp/SDLM-pytorch`.

1 Introduction

Language Modeling (LM) aims to measure the probability of a word sequence, reflecting its fluency and likelihood as a feasible sentence in a human language. Language Modeling is an essential component in a wide range of natural language processing (NLP) tasks, such as Machine Translation (Brown et al., 1990; Brants et al., 2007), Speech Recognition (Katz, 1987), Information Retrieval (Berger and Lafferty, 1999; Ponte and Croft, 1998; Miller et al., 1999; Hiemstra, 1998) and Document Summarization (Rush et al., 2015; Banko et al., 2000).

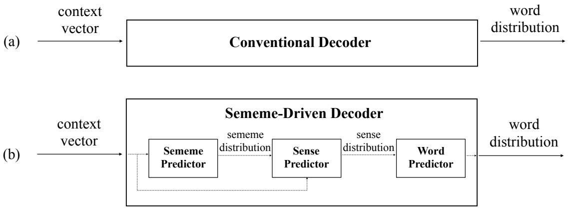

Figure 1: Decoder of (a) Conventional Language Model, (b) Sememe-Driven Language Model.

A probabilistic language model calculates the conditional probability of the next word given their contextual words, which are typically learned from large-scale text corpora. Taking the simplest language model for example, N-Gram estimates the conditional probabilities according to maximum likelihood over text corpora (Jurafsky, 2000). Recent years have also witnessed the advances of Recurrent Neural Networks (RNNs) as the state-of-the-art approach for language modeling (Mikolov et al., 2010), in which the context is represented as a low-dimensional hidden state to predict the next word.

Those conventional language models including neural models typically assume words as atomic symbols and model sequential patterns at word level. However, this assumption does not necessarily hold to some extent. Let us consider the following example sentence for which people want to predict the next word in the blank,

> *The U.S. trade deficit last year is initially estimated to be 40 billion* ______ .

People may first realize a *unit* should be filled in, then realize it should be a *currency unit*. Based on the country this sentence is talking about, the U.S., one may confirm it should be an *American cur-*

---

[*] Equal contribution.
[†] Correspondence author.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4642–4651
Brussels, Belgium, October 31 - November 4, 2018.

rency unit and predict the word *dollars*. Here, the *unit*, *currency*, and *American* can be regarded as basic semantic units of the word *dollars*. This process, however, has not been explicitly taken into consideration by conventional language models. That is, although in most cases words are atomic language units, words are not necessarily atomic semantic units for language modeling. We argue that explicitly modeling these atomic semantic units could improve both the performance and the interpretability of language models.

Linguists assume that there is a limited close set of atomic semantic units composing the semantic meanings of an open set of concepts (i.e. word senses). These atomic semantic units are named **sememes** (Dong and Dong, 2006).[i] Since sememes are naturally implicit in human languages, linguists have devoted much effort to explicitly annotate lexical sememes for words and build linguistic common-sense knowledge bases. HowNet (Dong and Dong, 2006) is one of the representative sememe knowledge bases, which annotates each Chinese word sense with its sememes. The philosophy of HowNet regards the parts and attributes of a concept can be well represented by sememes. HowNet has been widely utilized in many NLP tasks such as word similarity computation (Liu, 2002) and sentiment analysis (Fu et al., 2013). However, less effort has been devoted to exploring its effectiveness in language models, especially neural language models.

It is non-trivial for neural language models to incorporate discrete sememe knowledge, as it is not compatible with continuous representations in neural models. In this paper, we propose a Sememe-Driven Language Model (SDLM) to leverage lexical sememe knowledge. In order to predict the next word, we design a novel sememe-sense-word generation process: (1) We first estimate sememes' distribution according to the context. (2) Regarding these sememes as experts, we propose a sparse product of experts method to select the most probable senses. (3) Finally, the distribution of words could be easily calculated by marginalizing out the distribution of senses.

We evaluate the performance of SDLM on the language modeling task using a Chinese newspaper corpus People's Daily [ii] (Renmin Ribao), and also on the headline generation task using the Large Scale Chinese Short Text Summarization (LCSTS) dataset (Hu et al., 2015). Experimental results show that SDLM outperforms all those data-driven baseline models. We also conduct case studies to show that our model can effectively predict relevant sememes given context, which can improve the interpretability and robustness of language models.

2 Background

Language models target at learning the joint probability of a sequence of words $P(w^1, w^2, \cdots, w^n)$, which is usually factorized as $P(w^1, w^2, \cdots, w^n) = \prod_{t=1}^{n} P(w^t | w^{<t})$. Bengio et al. (2003) propose the first Neural Language Model as a feed-forward neural network. Mikolov et al. (2010) use RNN and a softmax layer to model the conditional probability. To be specific, it can be divided into two parts in series. First, a context vector $\mathbf{g}^t$ is derived from a deep recurrent neural network. Then, the probability $P(w^{t+1} | w^{\leq t}) = P(w^{t+1}; \mathbf{g}^t)$ is derived from a linear layer followed by a softmax layer based on $\mathbf{g}^t$. Let $\mathrm{RNN}(\cdot, \cdot; \boldsymbol{\theta}_{\mathrm{NN}})$ denote the deep recurrent neural network, where $\boldsymbol{\theta}_{\mathrm{NN}}$ denotes the parameters. The first part can be formulated as

$$\mathbf{g}^t = \mathrm{RNN}(\mathbf{x}_{w^t}, \{\mathbf{h}_l^{t-1}\}_{l=1}^{L}; \boldsymbol{\theta}_{\mathrm{NN}}). \qquad (1)$$

Here we use subscripts to denote layers and superscripts to denote timesteps. Thus $\mathbf{h}_l^t$ represents the hidden state of the L-th layer at timestep t. $\mathbf{x}_{w^t} \in \mathbb{R}^{H_0}$ is the input embedding of word w^t where H_0 is the input embedding size. We also have $\mathbf{g}^t \in \mathbb{R}^{H_1}$, where H_1 is the dimension of the context vector.

Supposing that there are N words in the language we want to model, the second part can be written as

$$P(w^{t+1}; \mathbf{g}^t) = \frac{\exp({\mathbf{g}^t}^{\mathrm{T}} \mathbf{w}_{w^{t+1}})}{\sum_{w'} \exp({\mathbf{g}^t}^{\mathrm{T}} \mathbf{w}_{w'})}, \qquad (2)$$

where $\mathbf{w}_w$ is the output embedding of word w and $\mathbf{w}_1, \mathbf{w}_2, \cdots \mathbf{w}_N \in \mathbb{R}^{H_2}$. Here H_2 is the output embedding size. For a conventional neural language model, H_2 always equals to H_1.

---

[i] Note that although sememes are defined as the minimum semantic units, there still exist several sememes for capturing syntactic information. For example, the word 和 "with" corresponds to one specific sememe 功能词 "FunctWord".

[ii] http://paper.people.com.cn/rmrb/

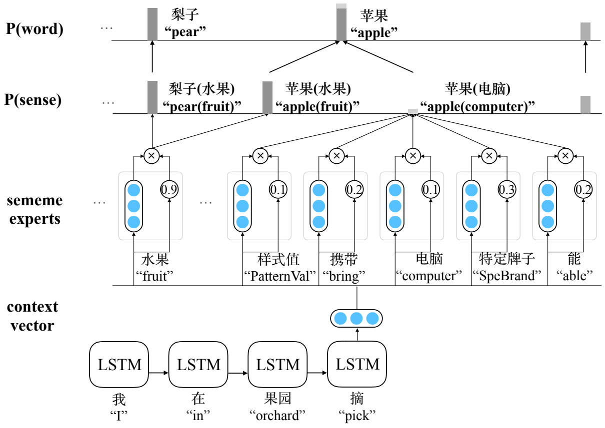

Figure 2: An example of the architecture of our model.

Given the corpus $\{w^t\}_{t=1}^{n}$, the loss function is defined by the negative log-likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{t=1}^{n}\log P(w^t|w^{<t};\boldsymbol{\theta}), \qquad (3)$$

where $\boldsymbol{\theta} = \{\{\mathbf{x}_i\}_{i=1}^{N}, \{\mathbf{w}_i\}_{i=1}^{N}, \boldsymbol{\theta}_{\text{NN}}\}$ is the set of parameters that are needed to be trained.

3 Methodology

In this section, we present our SDLM which utilizes sememe information to predict the probability of the next word. SDLM is composed of three modules in series: Sememe Predictor, Sense Predictor and Word Predictor. The Sememe Predictor first takes the context vector as input and assigns a weight to each sememe. Then each sememe is regarded as an expert and makes predictions about the probability distribution over a set of senses in the Sense Predictor. Finally, the probability of each word is obtained in the Word Predictor.

Here we use an example shown in Figure 2 to illustrate our architecture. Given context 我在果园摘 “In the orchard, I pick”, the actual next word could be 苹果 “apples”. From the context, especially the word 果园 “orchard” and 摘 “pick”, we can infer that the next word probably represents a kind of fruit. So the Sememe Predictor assigns a higher weight to the sememe 水果 “fruit” (0.9) and lower weights to irrelevant sememes like 电脑 “computer” (0.1). Therefore in the Sense Predictor, the sense 苹果 (水果) “apple (fruit)” is assigned a much higher probability than the sense 苹果 (电脑) “apple (computer)”. Finally, the probability of the word 苹果 “apple” is calculated as the sum of the probabilities of its senses 苹果 (水果) “apple(fruit)” and 苹果 (电脑) “apple (computer)”.

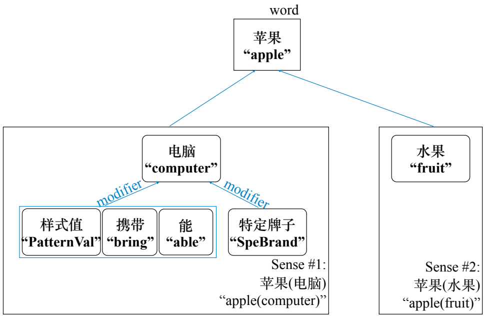

Figure 3: An example of the word-sense-sememe hierarchy.

In the following subsections, we first introduce the word-sense-sememe hierarchy in HowNet, and then give details about our SDLM.

3.1 Word-Sense-Sememe Hierarchy

We also use the example of “apple” to illustrate the word-sense-sememe hierarchy. As shown in Figure 3, the word 苹果 “apple” has two senses, one is the Apple brand, the other is a kind of fruit. Each sense is annotated with several sememes organized in a hierarchical structure. More specifically, in HowNet, sememes “PatternVal”, “bring”, “SpeBrand”, “computer” and “able” are annotated with the word “apple” and organized in a tree structure. In this paper, we ignore the structural relationship between sememes. For each word, we group all its sememes as an unordered set.

We present the notations that we use in the following subsections as follows. We define the overall sememe, sense, and word set as $\mathcal{E}$, $\mathcal{S}$ and $\mathcal{W}$. And we suppose the corpus contains $K = |\mathcal{E}|$ sememes, $M = |\mathcal{S}|$ senses and $N = |\mathcal{W}|$ words. For word $w \in \mathcal{W}$, we denote its corresponding sense set as $\mathcal{S}^{(w)}$. For sense $s \in \mathcal{S}^{(w)}$, we denote its corresponding sememes as an unordered set $\mathcal{E}^{(s)} = \{e_{n_1}, e_{n_2}, \cdots, e_{n_k}\} \subset \mathcal{E} = \{e_k\}_{k=1}^{K}$.

3.2 Sememe Predictor

The Sememe Predictor takes the context vector $\mathbf{g} \in \mathbb{R}^{H_1}$ as input and assigns a weight to each sememe. We assume that given the context $w^1, w^2, \cdots, w^{t-1}$, the events that word w^t contains sememe e_k ($k \in \{1, 2, \cdots, K\}$) are independent, since the sememe is the minimum semantic unit and there is no semantic overlap between any two different sememes. For simplicity, we ignore the superscript t. We design the Sememe Predictor as a linear decoder with the sigmoid activation function. Therefore, q_k, the probability that the next word contains sememe e_k, is formulated as

$$q_k = P(e_k|\mathbf{g}) = \sigma(\mathbf{g}^{\mathrm{T}}\mathbf{v}_k + b_k), \tag{4}$$

where $\mathbf{v}_k \in \mathbb{R}^{H_1}$, $b_k \in \mathbb{R}$ are trainable parameters, and $\sigma(\cdot)$ denotes the sigmoid activation function.

3.3 Sense Predictor and Word Predictor

The architecture of the Sense Predictor is motivated by Product of Experts (PoE) (Hinton, 1999). We regard each sememe as an expert that only makes predictions on the senses connected with it. Let $\mathcal{D}^{(e_k)}$ denote the set of senses that contain sememe e_k, the k-th expert. Different from conventional neural language models, which directly use the inner product of the context vector $\mathbf{g} \in \mathbb{R}^{H_1}$ and the output embedding $\mathbf{w}_w \in \mathbb{R}^{H_2}$ for word w to generate the score for each word, we use $\phi^{(k)}(\mathbf{g}, \mathbf{w})$ to calculate the score given by expert e_k. And we choose a bilinear function parameterized with a matrix $\mathbf{U}_k \in \mathbb{R}^{H_1 \times H_2}$ as a straight implementation of $\phi^{(k)}(\cdot, \cdot)$:

$$\phi^{(k)}(\mathbf{g}, \mathbf{w}) = \mathbf{g}^{\mathrm{T}}\mathbf{U}_k\mathbf{w}. \tag{5}$$

Let $\mathbf{w}_s$ denote the output embedding of sense s. The score of sense s provided by sememe expert e_k can be written as $\phi^{(k)}(\mathbf{g}, \mathbf{w}_s)$. Therefore, $P^{(e_k)}(s|\mathbf{g})$, the probability of sense s given by expert e_k, is formulated as

$$P^{(e_k)}(s|\mathbf{g}) = \frac{\exp(q_k C_{k,s}\phi^{(k)}(\mathbf{g}, \mathbf{w}_s))}{\sum_{s' \in \mathcal{D}^{(e_k)}} \exp(q_k C_{k,s'}\phi^{(k)}(\mathbf{g}, \mathbf{w}_{s'}))}, \tag{6}$$

where $C_{k,s}$ is a normalization constant because sense s is not connected to all experts (the connections are sparse with approximately λN edges, $\lambda < 5$). Here we can choose either $C_{k,s} = 1/|\mathcal{E}^{(s)}|$ (*left normalization*) or $C_{k,s} = 1/\sqrt{|\mathcal{E}^{(s)}||\mathcal{D}^{(e_k)}|}$ (*symmetric normalization*).

In the Sense Predictor, q_k can be viewed as a gate which controls the magnitude of the term $C_{k,s}\phi^{(k)}(\mathbf{g}, \mathbf{w}_{w_s})$, thus control the flatness of the sense distribution provided by sememe expert e_k. Consider the extreme case when $q_k \to 0$, the prediction will converge to the discrete uniform distribution. Intuitively, it means that the sememe expert will refuse to provide any useful information when it is not likely to be related to the next word.

Finally, we summarize the predictions on sense s by taking the product of the probabilities given by relevant experts and then normalize the result; that is to say, $P(s|\mathbf{g})$, the probability of sense s, satisfies

$$P(s|\mathbf{g}) \propto \prod_{e_k \in \mathcal{E}^{(s)}} P^{(e_k)}(s|\mathbf{g}). \tag{7}$$

Using Equation 5 and 6, we can formulate $P(s|\mathbf{g})$ as

$$P(s|\mathbf{g}) = \frac{\exp(\sum_{e_k \in \mathcal{E}^{(s)}} q_k C_{k,s}\mathbf{g}^{\mathrm{T}}\mathbf{U}_k\mathbf{w}_s)}{\sum_{s'} \exp(\sum_{e_k \in \mathcal{E}^{(s')}} q_k C_{k,s'}\mathbf{g}^{\mathrm{T}}\mathbf{U}_k\mathbf{w}_{s'})}. \tag{8}$$

It should be emphasized that all the supervision information provided by HowNet is embodied in the connections between the sememe experts and the senses. If the model wants to assign a high probability to sense s, it must assign a high probability to some of its relevant sememes. If the model wants to assign a low probability to sense s, it can assign a low probability to its relevant sememes. Moreover, the prediction made by sememe expert e_k has its own tendency because of its own $\phi^{(k)}(\cdot, \cdot)$. Besides, the sparsity of connections between experts and senses is also determined by HowNet itself. For our dataset, on average, a word is connected with 3.4 sememe experts and each sememe expert will make predictions about 22 senses.

As illustrated in Figure 2, in the Word Predictor, we get $P(w|\mathbf{g})$, the probability of word w, by summing up probabilities of corresponding s given by the Sense Predictor, that is

$$P(w|\mathbf{g}) = \sum_{s \in \mathcal{S}^{(w)}} P(s|\mathbf{g}). \tag{9}$$

3.4 Implementation Details

Basis Matrix Actually, HowNet contains $K \approx 2000$ sememes. In practice, we cannot directly introduce $K \times H_1 \times H_2$ parameters, which might be computationally infeasible and lead to overfitting. To address this problem, we apply a weight-sharing trick called the basis matrix. We use R basis matrices and their weighted sum to estimate $\mathbf{U}_k$:

$$\mathbf{U}_k = \sum_{r=1}^{R} \alpha_{k,r}\mathbf{Q}_r, \tag{10}$$

where $\mathbf{Q}_r \in \mathbb{R}^{H_1 \times H_2}$, $\alpha_{k,r} > 0$ are trainable parameters, and $\sum_{r=1}^{R} \alpha_{k,r} = 1$.

Weight Tying To incorporate the weight tying strategy (Inan et al., 2017; Press and Wolf, 2017), we use the same output embedding for multiple senses of a word. To be specific, the sense output embedding $\mathbf{w}_s$ for each $s \in S^{(w)}$ is the same as the word input embedding $\mathbf{x}_w$.

4 Experiments

We evaluate our SDLM on a Chinese language modeling dataset, namely People's Daily based on perplexity.[iii] Furthermore, to show that our SDLM structure can be a generic Chinese word-level decoder for sequence-to-sequence learning, we conduct a Chinese headline generation experiment on the LCSTS dataset. Finally, we explore the interpretability of our model with cases, showing the effectiveness of utilizing sememe knowledge.

4.1 Language Modeling

Dataset

We choose the People's Daily Corpus, which is widely used for Chinese NLP tasks, as the resource. It contains one month's news text from People's Daily (Renmin Ribao). Taking Penn Treebank (PTB) (Marcus et al., 1993) as a reference, we build a dataset for Chinese language modeling based on the People's Daily Corpus with 734k, 10k and 19k words in the training, validation and test set. After the preprocessing similar to (Mikolov et al., 2010) (see Appendix A), we get our dataset and the final vocabulary size is 13,476.

Baseline

As for baselines, we consider three kinds of neural language modeling architectures with LSTM cells: simple LSTM, Tied LSTM and AWD-LSTM.

LSTM and Tied LSTM Zaremba et al. (2014) use the dropout strategy to prevent overfitting for neural language models and adopt it to two-layer LSTMs with different embedding and hidden size: 650 for medium LSTM, and 1500 for large LSTM. Employing the weight tying strategy, we get Tied LSTM with better performance. We set LSTM and Tied LSTM of medium and large size as our baseline models and use the code from PyTorch examples[iv] as their implementations.

AWD-LSTM Based on several strategies for regularizing and optimizing LSTM-based language models, Merity et al. (2018) propose AWD-LSTM as a three-layer neural network, which serves as a very strong baseline for word-level language modeling. We build it with the code released by the authors[v].

Variants of Softmax Meanwhile, to compare our SDLM with other language modeling decoders, we set cHSM (Class-based Hierarchical Softmax) (Goodman, 2001), tHSM (Tree-based Hierarchical Softmax) (Mikolov et al., 2013) and MoS (Mixture of Softmaxes) (Yang et al., 2018) as the baseline add-on structures to the architectures above.

Experimental Settings

We apply our SDLM and other variants of softmax structures to the architectures mentioned above: LSTM (medium / large), Tied LSTM (medium / large) and AWD-LSTM. MoS and SDLM are only applied on the models that incorporate weight tying, while tHSM is only applied on the models without weight tying, since it is not compatible with this strategy.

For a fair comparison, we train these models with same experimental settings and conduct a hyper-parameter search for baselines as well as our models (the search setting and the optimal hyper-parameters can be found in Appendix C.1). We keep using these hyper-parameters in our SDLM for all architectures. It should be emphasized that we use the SGD optimizer for all architectures, and we decrease the learning rate by a factor of 2 if no improvement is observed on the validation set. We uniformly initialize the word embeddings, the class embeddings for cHSM and the non-leaf embeddings for tHSM in $[-0.1, 0.1]$. In addition, we set R, the number of basis matrices, to 5 in Tied LSTM architecture and to 10 in AWD-LSTM architecture. We choose the *left normalization* strategy because it performs better.

Experimental Results

Table 1 shows the perplexity on the validation and test set of our models and the baseline models. From Table 1, 2, and 3, we can observe that:

1. Our models outperform the corresponding baseline models of all structures, which indicates the effectiveness of our SDLM. Moreover, our SDLM not only consistently outperforms state-of-the-art MoS model, but also offers much better interpretability (as described in Sect. 4.3), which

---

[iii] Although we only conduct experiments on Chinese corpora, we argue that this model has the potential to be applied to other languages in the light of works on construction sememe knowledge bases for other languages, such as (Qi et al., 2018).

[iv] `https://github.com/pytorch/examples/tree/master/word_language_model`

[v] `https://github.com/salesforce/awd-lstm-lm`

makes it possible to interpret the prediction process of the language model. Note that under a fair comparison, we do not see MoS's improvement over AWD-LSTM while our SDLM outperforms it by 1.20 with respect to perplexity on the test set.

2. To further locate the performance improvement of our SDLM, we study the perplexity of the single-sense words and multi-sense words separately on Tied LSTM (medium) and Tied LSTM (medium) + SDLM. Improvements with respect to perplexity are presented in Table 2. The performance on both single-sense words and multi-sense words gets improved while multi-sense words benefit more from SDLM structure because they have richer sememe information.

3. In Table 3 we study the perplexity of words with different mean number of sememes. We can see that our model outperforms baselines in all cases and is expected to benefit more as the mean number of sememes increases.

Model	#Paras	Validation	Test
LSTM (medium)	24M	116.46	115.51
+ cHSM	24M	129.12	128.12
+ tHSM	24M	151.00	150.87
Tied LSTM (medium)	15M	105.35	104.67
+ cHSM	15M	116.78	115.66
+ MoS	17M	98.47	98.12
+ SDLM	17M	**97.75**	**97.32**
LSTM (large)	76M	112.39	111.66
+ cHSM	76M	120.07	119.45
+ tHSM	76M	140.41	139.61
Tied LSTM (large)	56M	101.46	100.71
+ cHSM	56M	108.28	107.52
+ MoS	67M	94.91	94.40
+ SDLM	67M	**94.24**	**93.60**
AWD-LSTM[iv]	26M	89.35	88.86
+ MoS	26M	92.98	92.76
+ SDLM	27M	**88.16**	**87.66**

Table 1: Single model perplexity on validation and test sets on the People's Daily dataset.

	#senses = 1	#senses > 1
Baseline ppl	93.21	121.18
SDLM ppl	87.22	111.88
Δppl	5.99	9.29
Δppl/Baseline ppl	6.4%	7.8%

Table 2: Perplexity of words with different number of senses on the test set.

We also test the robustness of our model by randomly removing 10% sememe-sense connections in HowNet. The test perplexity for Tied LSTM (medium) + SDLM slightly goes up to 97.67, compared to 97.32 with a complete HowNet, which shows that our model is robust to tiny incompleteness of annotations. However, the performance of out model is still largely dependent upon the accuracy of sememe annotations. As HowNet is continuously updated, we expect our model to perform better with sememe knowledge of higher quality.

	[1, 2)	[2, 4)	[4, 7)	[7, 14)
Baseline ppl	71.56	161.32	557.26	623.71
SDLM ppl	68.47	114.95	465.29	476.45
Δppl	3.09	16.36	91.98	147.25
Δppl/Baseline ppl	4.3%	10.1%	16.5%	23.61%

Table 3: Perplexity of words with different mean number of sememes on the test set.

4.2 Headline Generation

Dataset

We use the LCSTS dataset to evaluate our SDLM structure as the decoder of the sequence-to-sequence model. As its author suggests, we divide the dataset into the training set, the validation set and the test set, whose sizes are 2.4M, 8.7k and 725 respectively. Details can be found in Appendix B.

Models

For this task, we consider two models for comparison.

RNN-context As described in (Bahdanau et al., 2015), RNN-context is a basic sequence-to-sequence model with a bi-LSTM encoder, an LSTM decoder and attention mechanism adopted. The context vector is concatenated with the word embedding at each timestep when decoding. It's widely used for sequence-to-sequence learning, so we set it as the baseline model.

RNN-context-SDLM Based on RNN-context, we substitute the decoder with our proposed SDLM and name it RNN-context-SDLM.

Experimental Settings

We implement our models with PyTorch, on top of the OpenNMT libraries[v]. For both models, we set the word embedding size to 250, the hidden unit size to 250, the vocabulary size to 40000, and the beam size of the decoder to 5. For RNN-context-SDLM, we set the number of basis matrices to 3. We conduct a hyper-parameter search for both

---

[iv] We find that multi-layer AWD-LSTM has problems converging when adopting cHSM, so we skip that result.

[v] `http://opennmt.net`

models (see Appendix C.2 for settings and optimal hyper-parameters).

Experimental Results

Following previous works, we report the F1-score of ROUGE-1, ROUGE-2, and ROUGE-L on the test set. Table 4 shows that our model outperforms the baseline model on all metrics. We attribute the improvement to the use of SDLM structure.

Words in headlines do not always appear in the corresponding articles. However, words with the same sememes have a high probability to appear in the articles intuitively. Therefore, a probable reason for the improvement is that our model could predict sememes highly relevant to the article, thus generate more accurate headlines. This could be corroborated by our case study.

Model	Rouge-1	Rouge-2	Rouge-L
RNN-context	37.5	25.0	34.9
RNN-context-SDLM	**38.9**	**26.2**	**36.2**

Table 4: ROUGE scores of both models on the LCSTS test set.

4.3 Case Study

The above experiments demonstrate the effectiveness of our SDLM. Here we present some samples from the test set of the People's Daily Corpus in Table 5 as well as the LCSTS dataset in Table 6 and conduct further analysis.

For each example of language modeling, given the context of previous words, we list the Top 5 words and Top 5 sememes predicted by our SDLM. The target words and the sememes annotated with them in HowNet are blackened. Note that if the target word is an out-of-vocabulary (OOV) word, helpful sememes that are related to the target meaning are blackened.

Example (1)		
去年 美国 贸易逆差 初步 估计 为 <N> ______ 。		
The U.S. trade deficit last year is initially estimated to be <N> ______ .		
Top 5 word prediction		
美元 "dollar"	， ","	。 "."
日元 "yen"	和 "and"	
Top 5 sememe prediction		
商业 "commerce"	**金融 "finance"**	**单位 "unit"**
多少 "amount"	专 "proper name"	
Example (2)		
阿 总理 ______ 已 签署 了 一 项 命令 。		
Albanian Prime Minister ______ has signed an order.		
Top 5 word prediction		
内 "inside"	**<unk>**	在 "at"
塔 "tower"	和 "and"	
Top 5 sememe prediction		
政 "politics"	**人 "person"**	花草 "flowers"
担任 "undertake"	水域 "waters"	

Table 5: Some examples of word and sememe predictions on the test set of the People's Daily Corpus.

Sememes annotated with the corresponding sense of the target word 美元 "dollar" are 单位 "unit", 商业 "commerce", 金融 "finance", 货币 "money" and 美国 "US". In Example (1), the target word "dollar" is predicted correctly and most of its sememes are activated in the predicting process. It indicates that our SDLM has learned the word-sense-sememe hierarchy and used sememe knowledge to improve language modeling.

Example (2) shows that our SDLM can provide interpretable results on OOV word prediction with sememe information associated with it. The target word here should be the name of the Albanian prime minister, which is out of vocabulary. But with our model, one can still conclude that this word is probably relevant to the sememe "politics", "person", "flowers", "undertake" and "waters", most of which characterize the meaning of this OOV word – the name of a politician. This feature can be helpful when the vocabulary size is limited or there are many terminologies and names in the corpus.

For the example of headline generation, given the article and previous words, when generating the word 生 "student", except the sememe 预料 "predict", all other Top 5 predicted sememes have high relevance to either the predicted word or the context. To be specific, the sememe 学习 "study" is annotated with 生 "student" in HowNet. 考试 "exam" indicates "college entrance exam". 特定牌子 "brand" indicates "BMW". And 高等 "higher" indicates "higher education", which is the next step after this exam. We can conclude that with sememe knowledge, our SDLM structure can extract critical information from both the given article and generated words explicitly and produce better summarization based on it.

5 Related Work

Neural Language Modeling. RNNs have achieved state-of-the-art performance in the language modeling task since Mikolov et al. (2010) first apply RNNs for language modeling. Much work has been done to improve RNN-based language modeling. For example, a variety of work (Zaremba et al., 2014; Gal and Ghahramani, 2016; Merity et al., 2017, 2018) introduces many regularization and optimization methods for RNNs. Based on the observation that the word

Article
8 日 ， 阜 新 一 开 宝马 轿车 参加 高考 的 男 考生 考场 作弊 被 抓 ， 因 不满 监考 老师 没收 作弊 手机 ， 从 背后 一脚 将 女 监考 老师 从 最后 一排 踹到 讲台 。 并 口 出 狂 言 ： “ 你 知道 我 爸 是 谁 啊 ， 你 就 查 我 ？ ” 目前 ， 打人 考生 已 被拘留 。
On the 8th in Fuxin, a male student drove a BMW to take the college entrance exam and was caught cheating. Because the teacher confiscated his mobile phone, he kicked the teacher from the last row to the podium and shouted: ”Do you know who my dad is? How dare you catch me!” Currently, this student has been detained.
Gold
男生 高考 作弊 追打 监考 老师 ： 你 知道 我 爸 是 谁 ？
In the college entrance exam, a male student caught cheating hit the teacher: Do you know who my dad is?
RNN-context-SDLM
高考 生 作弊 被 抓 ： 你 知道 我 爸 是 谁 啊 ？
In the college entrance exam, a student was caught cheating: Do you know who my dad is?
Top 5 sememe prediction
考试 “exam” **学习 “study”** **特定牌子 “brand”** 预料 “predict” **高等 “higher”**

Table 6: An example of generated headlines on the LCSTS test set.

appearing in the previous context is more likely to appear again, some work (Grave et al., 2017a,b) proposes to use cache for improvements. In this paper, we mainly focus on the output decoder, the module between the context vector and the predicted probability distribution. Similar to our SDLM, Yang et al. (2018) propose a high-rank model which adopts a Mixture of Softmaxes structure for the output decoder. However, our model is sememe-driven with each expert corresponding to an interpretable sememe.

Hierarchical Decoder Since softmax computation on large vocabulary is time-consuming, therefore being a dominant part of the model's complexity, various hierarchical softmax models have been proposed to address this issue. These models can be categorized to class-based models and tree-based models according to their hierarchical structure. Goodman (2001) first proposes the class-based model which divides the whole vocabulary into different classes and uses a hierarchical softmax decoder to model the probability as $\mathbb{P}(\text{word}) = \mathbb{P}(\text{word}|\text{class})\mathbb{P}(\text{class})$, which is similar to our model. For the tree-based models, all words are organized in a tree structure and the word probability is calculated as the probability of always choosing the correct child along the path from the root node to the word node. While Morin and Bengio (2005) utilize knowledge from WordNet to build the tree, Mnih and Hinton (2008) build it in a bootstrapping way and Mikolov et al. (2013) construct a Huffman Tree based on word frequencies. Recently, Jiang et al. (2017) reform the tree-based structure to make it more efficient on GPUs. The major differences between our model and theirs are the purpose and the motivation. Our model targets at improving the performance and interpretability of language modeling using external knowledge in HowNet. Therefore, we take its philosophy of the word-sense-sememe hierarchy to design our hierarchical decoder. Meanwhile, the class-based and tree-based models are mainly designed to speed up the softmax computation in the training process.

Sememe. Recently, there are a lot of works concentrating on utilizing sememe knowledge in traditional natural language processing tasks. For example, Niu et al. (2017) use sememe knowledge to improve the quality of word embeddings and cope with the problem of word sense disambiguation. Xie et al. (2017) apply matrix factorization to predict sememes for words. Jin et al. (2018) improve their work by incorporating character-level information. Our work extends the previous works and tries to combine word-sense-sememe hierarchy with the sequential model. To be specific, this is the first work to improve the performance and interpretability of Neural Language Modeling with sememe knowledge.

Product of Experts. As Hinton (1999, 2002) propose, the final probability can be calculated as the product of probabilities given by experts.Gales and Airey (2006) apply PoE to the speech recognition where each expert is a Gaussian mixture model. Unlike their work, in our SDLM, each expert is mapped to a sememe with better interpretability. Moreover, as the final distribution is a categorical distribution, each expert is only responsible for making predictions on a subset of the categories (usually less than 10), so we call it Sparse Product of Experts.

Headline Generation. Headline generation is a kind of text summarization tasks. In recent years, with the advances of RNNs, a lot of works have been done in this domain. The encoder-decoder models (Sutskever et al., 2014; Cho et al., 2014) have achieved great success in sequence-to-sequence learning. Rush et al. (2015) propose a local attention-based model for abstractive sentence summarization. Gu et al. (2016) introduce the copying mechanism which is close to the rote memorization of the human being. Ayana et al. (2016) employ the minimum risk training strategy to optimize model parameters. Different from these works, we focus on the decoder of the sequence-to-sequence model, and adopt SDLM to utilize sememe knowledge for sentence generation.

6 Conclusion and Further Work

In this paper, we propose an interpretable Sememe-Driven Language Model with a hierarchical sememe-sense-word decoder. Besides interpretability, our model also achieves state-of-the-art performance in the Chinese Language Modeling task and shows improvement in the Headline Generation task. These results indicate that SDLM can successfully take advantages of sememe knowledge.

As for future work, we plan the following research directions: (1) In language modeling, given a sequence of words, a sequence of corresponding sememes can also be obtained. We will utilize the context sememe information for better sememe and word prediction. (2) Structural information about sememes in HowNet is ignored in our work. We will extend our model with the hierarchical sememe tree for more accurate relations between words and their sememes. (3) It is imaginable that the performance of SDLM will be significantly influenced by the annotation quality of sememe knowledge. We will also devote to further enrich the sememe knowledge for new words and phrases, and investigate its effect on SDLM.

Acknowledgement

This work is supported by the 973 Program (No. 2014CB340501), the National Natural Science Foundation of China (NSFC No. 61572273) and the research fund of Tsinghua University-Tencent Joint Laboratory for Internet Innovation Technology. This work is also funded by China Association for Science and Technology (2016QNRC001). Hao Zhu and Jun Yan are supported by Tsinghua University Initiative Scientific Research Program. We thank all members of Tsinghua NLP lab. We also thank anonymous reviewers for their careful reading and their insightful comments.

References

Shiqi Shen Ayana, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with minimum risk training. *arXiv preprint arXiv:1604.01904*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Michele Banko, Vibhu O Mittal, and Michael J Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of ACL*, pages 318–325. Association for Computational Linguistics.

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Adam Berger and John Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of SIGIR*, pages 222–229. ACM.

Thorsten Brants, Ashok C Popat, Peng Xu, Franz J Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP*.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Zhendong Dong and Qiang Dong. 2006. *Hownet and the computation of meaning (with Cd-rom)*. World Scientific.

Xianghua Fu, Guo Liu, Yanyan Guo, and Zhiqiang Wang. 2013. Multi-aspect sentiment analysis for chinese online social reviews based on topic modeling and hownet lexicon. *Knowledge-Based Systems*, 37:186–195.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NIPS*.

M. J. F. Gales and S. S. Airey. 2006. Product of gaussians for speech recognition. *Computer Speech and Language*, 20(1):22–40.

J Goodman. 2001. Classes for fast maximum entropy training. In *Proceedings of ICASSP*, pages 561–564 vol.1.

Edouard Grave, Moustapha Cisse, and Armand Joulin. 2017a. Unbounded cache model for online language modeling with open vocabulary. In *Proceedings of NIPS*.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017b. Improving neural language models with a continuous cache. In *Proceedings of ICLR*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*, pages 1631–1640.

Djoerd Hiemstra. 1998. A linguistically motivated probabilistic model of information retrieval. In *Proceedings of TPDL*, pages 569–584. Springer.

G. E Hinton. 1999. Products of experts. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on*, pages 1–6 vol.1.

G. E. Hinton. 2002. *Training products of experts by minimizing contrastive divergence.* MIT Press.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. In *Proceedings of EMNLP*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proceedings of ICLR*.

Nan Jiang, Wenge Rong, Min Gao, Yikang Shen, Zhang Xiong, Nan Jiang, Wenge Rong, Min Gao, Yikang Shen, and Zhang Xiong. 2017. Exploration of tree-based hierarchical softmax for recurrent language models. In *Proceedings of IJCAI*.

Huiming Jin, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin, and Leyu Lin. 2018. Incorporating chinese characters of words for lexical sememe prediction. In *Proceedings of ACL*, pages 2439–2449. Association for Computational Linguistics.

Dan Jurafsky. 2000. Speech & language processing. chapter 4. Pearson Education India.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

Qun Liu. 2002. Word similarity computing based on hownet. *Computational linguistics and Chinese language processing*, 7(2):59–76.

Mitchell P. Marcus, Beatrice Santorini, and Ann Marcinkiewicz, Mary. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *Proceedings of ICLR*.

Stephen Merity, Bryan Mccann, and Richard Socher. 2017. Revisiting activation regularization for language rnns. *arXiv preprint arXiv:1708.01009*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

David RH Miller, Tim Leek, and Richard M Schwartz. 1999. A hidden markov model information retrieval system. In *Proceedings of SIGIR*, pages 214–221. ACM.

Andriy Mnih and Geoffrey Hinton. 2008. A scalable hierarchical distributed language model. In *Proceedings of NIPS*, pages 1081–1088.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of AISTATS*.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In *Proceedings of ACL*, volume 1, pages 2049–2058.

Jay M Ponte and W Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of SIGIR*, pages 275–281. ACM.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of EACL*.

Fanchao Qi, Yankai Lin, Maosong Sun, Hao Zhu, Ruobing Xie, and Zhiyuan Liu. 2018. Cross-lingual lexical sememe prediction. In *Proceedings of EMNLP*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Ruobing Xie, Xingchi Yuan, Zhiyuan Liu, and Maosong Sun. 2017. Lexical sememe prediction via word embeddings and matrix factorization. In *Proceedings of IJCAI*, pages 4200–4206. AAAI Press.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank rnn language model. In *Proceedings of ICLR*.

Wojciech Zaremba, Ilya. Sutskever, and Oriol. Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Siamese Network-Based Supervised Topic Modeling

Minghui Huang[1,*], **Yanghui Rao**[1,†], **Yuwei Liu**[1], **Haoran Xie**[2], **Fu Lee Wang**[3]
[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
[2]Department of Mathematics and Information Technology,
The Education University of Hong Kong, Tai Po, Hong Kong
[3]School of Science and Technology,
The Open University of Hong Kong, Ho Man Tin, Kowloon, Hong Kong
huangmh25@mail2.sysu.edu.cn, raoyangh@mail.sysu.edu.cn,
liuyw23@mail2.sysu.edu.cn, hrxie2@gmail.com,
pwang@ouhk.edu.hk

Abstract

Label-specific topics can be widely used for supporting personality psychology, aspect-level sentiment analysis, and cross-domain sentiment classification. To generate label-specific topics, several supervised topic models which adopt likelihood-driven objective functions have been proposed. However, it is hard for them to get a precise estimation on both topic discovery and supervised learning. In this study, we propose a supervised topic model based on the Siamese network, which can trade off label-specific word distributions with document-specific label distributions in a uniform framework. Experiments on real-world datasets validate that our model performs competitive in topic discovery quantitatively and qualitatively. Furthermore, the proposed model can effectively predict categorical or real-valued labels for new documents by generating word embeddings from a label-specific topical space.

1 Introduction

As one of the most widely used text mining techniques, topic modeling can extract meaningful descriptions (i.e., topics) from a corpus (Blei, 2012). Most previous topic models, such as probabilistic Latent Semantic Analysis (pLSA) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2001) are unsupervised. In unsupervised topic models, each document is defined as a mixture distribution over topics and each topic is represented as a mixture distribution over words. Unsupervised topic models only exploit words in documents and do not incorporate the guidance of labels into learning processes. Therefore, these models fail to discover label-specific topics, which are important to support personality psychology (Weiner and Graham, 1990), aspect-level sentiment analysis (Liu, 2012), and cross-domain sentiment classification (He et al., 2011). For example, label-specific topics generated from sentimental texts can help to find attributions and causes for different sentiments by associating sentiments with real-world topics/events.

In light of this consideration, several supervised topic models are proposed to generate label-specific topics. One of the most representative models is the supervised Latent Dirichlet Allocation (sLDA) (Blei and McAuliffe, 2007), which restricts a document being associated with one real-valued response variable. To deal with categorical labels, multi-class sLDA (sLDAc) (Wang et al., 2009) and Labeled Latent Dirichlet Allocation (L-LDA) (Ramage et al., 2009) are proposed, but they are only applicable to classification. Recently, a supervised Neural Topic Model (sNTM) (Cao et al., 2015) is developed to tackle supervised tasks of both classification and regression. As a hybrid method, sNTM is in essence a neural network by following the document-topic distribution in topic models. Unfortunately, the label information has a little effect on topic generation since sNTM models documents and labels separately.

The above limitation motivates us to develop a supervised topic model which can jointly model documents and labels. Particularly, we propose a Siamese Labeled Topic Model (SLTM) to exploit the information of documents and labels based on the Siamese network (Bromley et al., 1993; Hu et al., 2014; Wang and Zhang, 2017), where weight matrices in SLTM represent conditional distributions. Therefore, by constraining weight matrices during the learning procedure, SLTM can follow probabilistic characteristics of topic models strictly. Compared to previous supervised topic models, the main advantages of our SLTM are summarized as follows. First, SLTM can gener-

---

[*] This work was finished when the first author was an undergraduate student of her final year.

[†] The corresponding author.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4652–4662
Brussels, Belgium, October 31 - November 4, 2018.

ate more coherent label-specific topics than others. This is because the supervision of labels is incorporated into topic modeling for SLTM. On the other hand, the mapping of topics to labels is unconstrained for most existing supervised topic models, which renders many coherent topics being generated outside labels. Second, strengths of neural networks are incorporated into SLTM to bootstrap its inference power on label prediction. Third, each word can be mapped to a topical embedding space and represented by a word embedding after generating label-specific topics.

To validate the effectiveness of the proposed model, we evaluate it on two real-world datasets in text mining. Experimental results indicate that our method is able to discover more coherent and label-specific topics than baseline models. Moreover, word embeddings learned by the proposed model can be used to predict labels for new documents effectively.

The remainder of this paper is organized as follows. We summarize related studies on supervised topic modeling in Section 2. For convenience of describing our model, we present the neural network view of topic models in Section 3. Then, we detail the proposed SLTM in Section 4. Experimental design and analysis of results are shown in Section 5. Finally, we present conclusions and future work in Section 6.

2 Related Work

Topic models, which focus on discovering unobserved class variables named "topics" statistically, have been widely used in text mining. One of the early topic models is pLSA (Hofmann, 1999). In pLSA, a document's word vector was decomposed into a mixture of topics, and a topic was represented as a probability distribution over words. LDA (Blei et al., 2001) extended pLSA by adding Dirichlet priors for a document's multinomial distribution over topics and a topic's multinomial distribution over words, which makes it suitable to generate topics for unseen documents.

The aforementioned models are unsupervised, which may be computationally costly to do some task-specific transformation when there is extra labeling information (Cao et al., 2015). To address this issue, several supervised topic models have been proposed to introduce the label guidance in learning processes. One of the most widely used supervised topic models is sLDA (Blei and McAuliffe, 2007). In sLDA, each document was paired with a response variable which obeys the Gaussian distribution. By extending the sLDA, BP-sLDA (Chen et al., 2015) applied back propagation over a deep architecture in conjunction with stochastic gradient/mirror descent for model parameter estimation, leading to scalable and end-to-end discriminative learning characteristics. Based on sLDA, multi-class sLDA (sLDAc) (Wang et al., 2009) was proposed to model documents with categorical labels by adding a softmax classifier rather than a linear regression in sLDA to a standard LDA. Another method of tackling corpora with discrete labels is L-LDA (Ramage et al., 2009), which associated each label with only one topic. To improve the performance of L-LDA in the classification task, Dependency-LDA (Dep-LDA) (Rubin et al., 2012) incorporated an extra topic model to capture the dependencies between labels and took the label dependencies into consideration when estimating topic distributions. Recently, a nonparametric supervised topic model (Li et al., 2018) was proposed to predict the response of interest (e.g., product ratings and sales). The limitation of above models is that they are only applicable to either discrete or continuous data.

In this paper, we propose a Siamese network-based supervised topic model named SLTM. The most relevant work to SLTM is the supervised Neural Topic Model (sNTM) for both classification and regression tasks (Cao et al., 2015), which constructed two hidden layers to generate the n-gram topic and document-topic representations. However, different from our SLTM using bag-of-words methods, sNTM adopted fixed embeddings trained on external resources (Mikolov et al., 2013). Thus, sNTM can not learn data-specific topics. Furthermore, sNTM is hard to follow probabilistic characteristics of the topic-word distribution in topic models, because a topic generated by sNTM is composed of an infinite number of n-grams. Finally, sNTM modeled documents and labels separately rather than uniformly in our SLTM.

3 Preliminaries

For convenience of describing the proposed model, we use hollow uppercase letters (e.g., $\mathbb{D}$) to represent collections, bold uppercase letters (e.g., $\mathbf{W}_1$) to represent matrices, bold lowercase letters (e.g., $\mathbf{y}_i$) to represent vectors, regular uppercase letters (e.g., M) to represent scalar constants,

Table 1: Frequently used notations.

Notation	Description
M	Number of documents
K	Number of topics
N	Size of the vocabulary
L	Size of the label set
$\mathbb{D}$	Document collection
$d_i \in \mathbb{D}$	The i-th document
$\mathbb{V}$	Vocabulary
$v_j \in \mathbb{V}$	The j-th word
$\mathbb{Z}$	Topic collection
$z_k \in \mathbb{Z}$	The k-th topic
$\mathbb{Y}$	Label collection
$\mathbf{y}_i \in \mathbb{R}^L$	Labels for document d_i
$p(v_j\|d_i)$	The probability of v_j given d_i

and regular lowercase letters (e.g., v_j) to represent scalar variables. Based on the above convention, frequently used notations are shown in Table 1. Given a document d_i with labels $\mathbf{y}_i$, our goal is to discover topics with a neural network framework. Therefore, we first describe the neural network view of topic models briefly.

Topic modeling is a popular latent variable inference method for co-occurrence data which associates unobserved classes with observations d_i and v_j, where v_j is a word in d_i. The conditional probability $p(v_j|d_i)$ is defined as:

$$p(v_j|d_i) = \sum_{k=1}^{K} p(v_j|z_k)p(z_k|d_i). \tag{1}$$

Let $\phi(v_j) = [p(v_j|z_1), \ldots, p(v_j|z_K)]$ and $\theta(d_i) = [p(z_1|d_i), \ldots, p(z_K|d_i)]$, then $p(v_j|d_i)$ in Equation 1 can be represented as the following vector form:

$$p(v_j|d_i) = \phi(v_j) \cdot \theta(d_i). \tag{2}$$

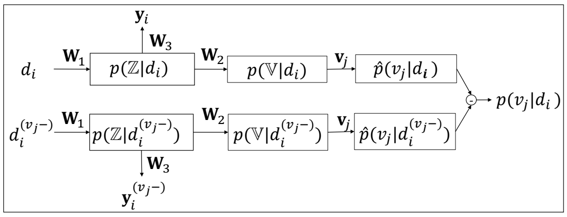

Figure 1: SLTM's word generation framework.

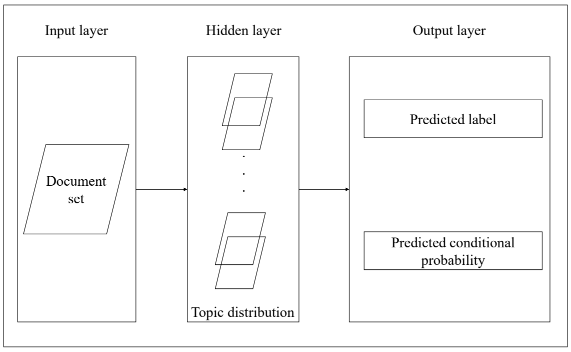

Figure 2: SLTM's architecture from the perspective of neural networks.

We represent horizontal stack by commas and vertical stack by semicolons, thus $\mathbf{W}_1 = [\theta(d_1)^T, \ldots, \theta(d_M)^T] \in \mathbb{R}^{K \times M}$ and $\mathbf{W}_2 = [\phi(v_1); \ldots; \phi(v_N)] \in \mathbb{R}^{N \times K}$, which are constrained by: $\mathbf{W}_1[k,m] \geq 0$, $\mathbf{W}_2[n,k] \geq 0$, $\sum_{k=1}^{K} \mathbf{W}_1[k,m] = 1$, and $\sum_{j=1}^{N} \mathbf{W}_2[j,k] = 1$, where $k \in [1,K]$, $m \in [1,M]$, and $n \in [1,N]$. Then, the vector form in Equation 2 can be extended to:

$$\begin{aligned} p(\mathbb{V}|\mathbb{D}) &= \begin{bmatrix} p(v_1|d_1) & \cdots & p(v_1|d_M) \\ \vdots & \ddots & \vdots \\ p(v_N|d_1) & \cdots & p(v_N|d_M) \end{bmatrix} \\ &= \begin{bmatrix} (\phi(v_1) \cdot \theta(d_1)) & \cdots & (\phi(v_1) \cdot \theta(d_M)) \\ \vdots & \ddots & \vdots \\ (\phi(v_N) \cdot \theta(d_1)) & \cdots & (\phi(v_N) \cdot \theta(d_M)) \end{bmatrix} \\ &= \mathbf{W}_2\mathbf{W}_1. \end{aligned} \tag{3}$$

With Equation 3, topic models can be viewed as neural networks, where $\mathbb{D}$ and $\mathbb{V}$ are input sets, $p(\mathbb{V}|\mathbb{D})$ is the output set, and $\mathbf{W}_1$ and $\mathbf{W}_2$ are parameter matrices of the neural network.

4 Siamese Labeled Topic Model

Similar to generative models such as pLSA, we propose a Siamese Labeled Topic Model (SLTM) based on the aforementioned neural network perspective of topic models. Figure 1 illustrates the framework of generating each word in SLTM, and the process is as follows. For a document d_i in $\mathbb{D}$, the topic distribution $p(\mathbb{Z}|d_i)$ is estimated by:

$$p(\mathbb{Z}|d_i) = \mathbf{W}_1\mathbf{d}_i, \tag{4}$$

where $\mathbf{d}_i$ is the indicator vector (Yang et al., 2013) of d_i, which means that the i-th entry of $\mathbf{d}_i$ is 1 and other entries are 0. Labels of d_i are $\mathbf{y}_i$, which are generated from the topic distribution of d_i as: $\mathbf{y}_i = \mathbf{W}_3 p(\mathbb{Z}|d_i)$. The above equation is constrained by $\mathbf{W}_3[l,k] \geq 0$, where $l \in [1, L]$, $k \in [1, K]$, and $\sum_{l=1}^{L} \mathbf{W}_3[l,k] = 1$ if $L > 1$. A topic z_k in $\mathbb{Z}$ has its word distribution $p(\mathbb{V}|z_k)$, which is computed by:

$$p(\mathbb{V}|z_k) = \mathbf{W}_2 \mathbf{z}_k, \tag{5}$$

where $\mathbf{z}_k$ is the indicator vector of z_k. Therefore, words can be generated from d_i as:

$$p(\mathbb{V}|d_i) = \mathbf{W}_2 \mathbf{W}_1 \mathbf{d}_i. \tag{6}$$

The architecture of SLTM from the perspective of neural networks is shown in Figure 2. With respect to the model optimization, we adopt the contrastive objective function used in previous works (Socher et al., 2014; Cui et al., 2014; Cao et al., 2015; He et al., 2017). For document d_i and every word v_j in d_i, we randomly sample a document from the document set $\mathbb{D}$ which does not contain v_j, as a negative sample document. The negative sample document is represented as $d_i^{(v_j-)}$ and has labels $\mathbf{y}_i^{(v_j-)}$. As shown in Figure 1, the lower sub-network, which takes $d_i^{(v_j-)}$ as input, has the same architecture as the the upper sub-network, which takes d_i as input. Because the document-topic distribution and the topic-word distribution of a corpus are fixed, $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{W}_3$ are shared among two sub-networks of our model. These two sub-networks are twin networks and thus the proposed model is essentially the Siamese network. Our objective is to make word v_j be learned by topics in document d_i, while not be learned by topics in the negative sampled document $d_i^{(v_j-)}$. Therefore, we only take word v_j in $\mathbb{V}$ into consideration during the learning procedure, which can be implemented as dot-multiplying $p(\mathbb{V}|d_i)$ with the indicator vector of v_j (i.e., $\mathbf{v}_j$) as: $\hat{p}(v_j|d_i) = p(\mathbb{V}|d_i) \cdot \mathbf{v}_j$. Particularly, the objective is to make the predicted conditional probability $\hat{p}(v_j|d_i)$ approach the observed conditional probability $p(v_j|d_i)$ (i.e., term frequency of word v_j in document d_i), while make the conditional probability $\hat{p}(v_j|d_i^{(v_j-)})$ approach zero. Thus, the loss function of predicted conditional probabilities and the observed conditional probability can be defined as:

Algorithm 1 Training Algorithm for SLTM

Input: $\mathbb{S} = \{\mathbb{D}, \mathbb{Y}\}$;

```
repeat
    for all $(d_i, \mathbf{y}_i) \in \mathbb{S}$ do
        for each word $v_j$ in document $d_i$ do
            Sample a document $d_i^{(v_j-)}$ which does not contain $v_j$;
            Calculate $loss(SLTM)$;
            if $loss(SLTM)$ is reducing then
                Update $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{W}_3$;
            end if
        end for
    end for
until convergence
```

$$\begin{aligned} &loss(d_i, d_i^{(v_j-)}) \\ &\quad = |p(v_j|d_i) - \hat{p}(v_j|d_i) + \hat{p}(v_j|d_i^{(v_j-)})|. \end{aligned} \tag{7}$$

We use another loss function $loss(\mathbf{y}_i, \mathbf{y}_i^{(v_j-)})$ to capture labels of d_i and $d_i^{(v_j-)}$, where $loss(\mathbf{y}_i, \mathbf{y}_i^{(v_j-)}) = loss(\mathbf{y}_i) + loss(\mathbf{y}_i^{(v_j-)})$. In the above, equations of $loss(\mathbf{y}_i)$ and $loss(\mathbf{y}_i^{(v_j-)})$ depend on the property of labels. For categorical and real-valued labels, the cross-entropy (Tang et al., 2014) and the mean absolute error (Willmott and Matsuura, 2005) are adopted, respectively.

The maximization of the weighted sum of conditional likelihoods is equivalent to minimize the losses of the weighted sum of loss functions, and these two loss functions are weighted by a hyperparameter α as in (Tang et al., 2014). Thus, the loss function of SLTM is:

$$\begin{aligned} loss(SLTM) &= \alpha \times loss(d_i, d_i^{(v_j-)}) \\ &\quad + (1-\alpha) \times loss(\mathbf{y}_i, \mathbf{y}_i^{(v_j-)}). \end{aligned} \tag{8}$$

The effect of α on predicting labels and discovering topics will be investigated in Section 5.6. Based on $loss(SLTM)$, three kinds of weights, i.e., $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ can be updated together by a vanilla back propagation (BP) algorithm with the early stopping criteria (Bengio, 2012). The training algorithm is shown in Algorithm 1.

After training, we obtain both document-topic and topic-word distributions. Then, each word can be mapped to a topic-level embedding space and represented as a word embedding. For instance, the word embedding of v_j is generated from the topic-word distribution $\mathbf{W}_2$ as:

$$\mathbf{e}(v_j) = \mathbf{W}_2[j, :]. \tag{9}$$

The generated word embeddings can be used for specific applications, such as label prediction. Particularly, we firstly represent a new document d_n by its document embeddings $\mathbf{e}(d_n)$, where $\mathbf{e}(d_n)$ is the sum of word embeddings of all words in d_n. Then, the predicted labels $\hat{\mathbf{y}}_n$ of document d_n can be estimated by:

$$\hat{\mathbf{y}}_n = f(\mathbf{W}_4 \mathbf{e}(d_n)), \tag{10}$$

where $\mathbf{W}_4$ denotes weights of each topic contributing to labels, and $f(.)$ is the activation function which depends on the type of labels. For categorical and normalized real-valued labels, we respectively adopt softmax and sigmoid as activation functions. Note that we do not predict labels for new documents based on $\mathbf{W}_3$ directly, because topic distributions of these documents can only be learned without the supervision of labels, i.e., new documents' topic distributions may be inconsistent to $\mathbf{W}_3$. Finally, we update $\mathbf{W}_4$ and word embeddings by RMSprop (Tieleman and Hinton, 2012) for label prediction.

5 Experiments

In this section, we firstly describe datasets and the setting of experiments. Secondly, we investigate the quality of generated topics by the topic coherence score and qualitative analysis. Thirdly, the quality of generated word embeddings is evaluated by label prediction and word similarity. Finally, the effect of the hyper-parameter α is evaluated on coherence of topics and label prediction.

5.1 Datasets and Setting

To evaluate the effectiveness of our method comprehensively, we conduct experiments on two real-world datasets with categorical and real-valued labels, respectively. The first corpus named ISEAR contains a collection of 7,666 sentences and each item is manually tagged with a categorical label over 7 emotions (Scherer and Wallbott, 1994). The second dataset YouTube[1] is often used for sentiment strength detection, which contains 3,407 comments on videos and each item is labeled with a real value between 0.1 (i.e., very negative sentiment) and 0.9 (i.e., very positive sentiment). These two datasets are selected for their similar word numbers in average. After removing stop words, the mean numbers of words in each document are 8.53 and 8.56 for ISEAR and YouTube. Besides, it is appropriate to evaluate the model performance on predicting emotions and sentiment strengths, because topics play an important role in understanding sentences or user comments (Liu, 2012). Since the proposed SLTM is suitable to both topic discovery and classification/regression tasks, we employ five kinds of baselines for comparison.

The first kind are the support vector machine (SVM), an efficient deep learning model for classification (i.e., fastText) (Grave et al., 2017), and the following supervised topic models which are confined to categorical labels:

- sLDAc (Wang et al., 2009): it models documents with categorical labels by adding a softmax classifier to a standard LDA.
- L-LDA (Ramage et al., 2009): it is a supervised model which associates labels with topics by one-to-one correspondence. Accordingly, the number of topics in L-LDA must equal the size of the label set.
- Dep-LDA (Rubin et al., 2012): it extends L-LDA by introducing a multinomial distribution over labels and capturing the dependencies between labels. Then, the label dependencies are used to sample topic distributions in supervised learning.

The second kind are the support vector regression (SVR), a state-of-the-art deep learning model for sentiment strength detection (i.e., HCNN) (Chen et al., 2017), and the following supervised topic models which are developed for predicting real-valued labels only:

- sLDA (Blei and McAuliffe, 2007): it is a classical supervised topic model, in which, each document is paired with a response variable, and the variable is defined as a Gaussian distribution with a mean value that is computed by a linear regression of topics.
- BP-sLDA (Chen et al., 2015): it applies back propagation over a deep architecture together with stochastic gradient/mirror descent for

---

[1] http://sentistrength.wlv.ac.uk/

Table 2: Topic coherence scores on ISEAR using different numbers of top words T.

	5	10	15
pLSA	0.0051	0.0024	-0.0013
LDA	0.0954	0.0492	0.0014
sLDAc	0.0014	0.0031	-0.0035
sNTM	-0.9267	-0.9508	-0.9667
SLTM	**0.1142**	**0.0680**	**0.0025**

Table 3: Topic coherence scores on YouTube using different numbers of top words T.

	5	10	15
pLSA	-0.0535	-0.2435	-0.3829
LDA	**-0.0154**	-0.2142	-0.3618
sLDA	-0.0627	-0.2502	-0.3962
BP-sLDA	-0.7021	-0.7670	-0.7900
sNTM	-0.9138	-0.9253	-0.9376
SLTM	-0.0993	**-0.1967**	**-0.3268**

parameter estimation of sLDA. The number of hidden layers is set to 3.

The third kind is a supervised n-gram model named sNTM, which is applicable to predict both categorical and real-valued labels for new documents (Cao et al., 2015). In sNTM, each n-gram is represented by a 300-dimensional embedding vector using the available tool word2vec[2]. By following (Cao et al., 2015), a large-scale Google News dataset with around 100 billion words is adopted for training. For topic discovery, two unsupervised topic models, pLSA (Hofmann, 1999) and LDA (Blei et al., 2001), are used as the fourth kind of baselines. Finally, we adopt two hybrid methods by combining LDA and supervised learning algorithms as baselines. In particular, a softmax classifier and a liner regression (LR) are used to predict categorical and real-valued labels for documents, respectively. Unless otherwise specified, we set α to 0.5 and adopt the stochastic gradient descent with batch size of 100 for training SLTM.

5.2 Coherence Score of Topics

To investigate the quality of topics discovered by SLTM quantitatively, we use the topic coherence score based on the normalised pointwise mutual information (Lau et al., 2014) as the evaluation metric. Intuitively, a topic coherence score that is larger indicates that the quality of topics is better. All unsupervised topic models (i.e., pLSA and LDA) and supervised methods which associate one label with multiple topics (i.e., sLDAc, sLDA, BP-sLDA, and sNTM) are adopted for comparison. Although L-LDA and Dep-LDA can identify label-specific topics on ISEAR, these models' one-to-one mapping of labels and topics makes them unsuitable in this evaluation. Particularly, L-LDA and Dep-LDA constraint each topic to words in certain documents with the same label, which renders their coherence scores being estimated by a subset of the corpus only. On the other hand, the quality of topics is evaluated on the whole corpus for SLTM and other baseline models.

Table 4: Each label's top 5 words on ISEAR.

Labels	Models	Top 5 words of label-specific topics
fear	sLDAc	home night car **afraid fear**
	L-LDA/Dep-LDA	night **afraid** car home **fear** felt
	SLTM	night **afraid fear** car home dark
joy	sLDAc	year passed heard exam university
	L-LDA/Dep-LDA	friend got time passed felt
	SLTM	**happy joy** passed got university
guilt	sLDAc	did didn't asked **guilty** said
	L-LDA/Dep-LDA	felt **guilty** friend did mother
	SLTM	**guilty** felt mother did friend
disgust	sLDAc	saw man **disgusted disgust** woman
	L-LDA/Dep-LDA	**disgusted** saw felt people friend
	SLTM	**disgusted** saw people man **disgust**
shame	sLDAc	know **ashamed** teacher happened lot
	L-LDA/Dep-LDA	**ashamed** felt friend time did
	SLTM	**ashamed** felt **shame** class teacher
anger	sLDAc	**angry** called new **anger** expected
	L-LDA/Dep-LDA	friend **angry** did time told
	SLTM	**angry** friend **anger** brother told
sadness	sLDAc	father close died away years
	L-LDA/Dep-LDA	died friend **sad** felt time
	SLTM	died **sad** death away friend

The average coherence scores of topics generated by different models on ISEAR and YouTube are respectively shown in Table 2 and Table 3, where the number of topics is 20, the number of top words T is set to 5, 10, and 15, and the best scores are highlighted in boldface. The results indicate that SLTM can discover more coherent topics than both unsupervised topic models and supervised methods, except for $T = 5$ on YouTube. It is also interesting to observe that supervised baseline models (i.e., sLDAc, sLDA, BP-sLDA, and sNTM) perform worse than pLSA and LDA for most cases, which validates that it is challenging to trade off label-specific word distributions with document-specific label distributions (Ramage et al., 2009).

5.3 Qualitative Analysis on Topics

In this part, we conduct qualitative inspection of 20 topics generated by SLTM. The ISEAR dataset

---

[2] https://code.google.com/p/word2vec/

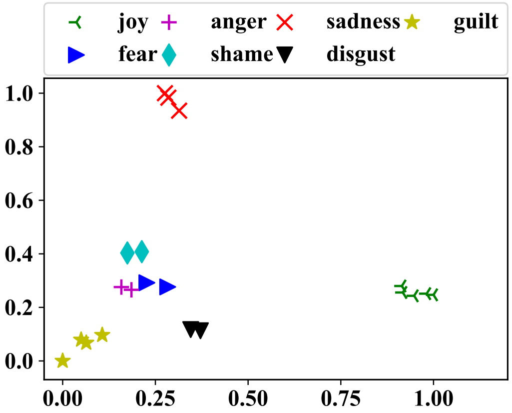

Figure 3: Scatter plot of topics identified by SLTM on ISEAR, where each point indicates a topic.

which contains multiple labels is used for illustration, since it is inappropriate to present the results on YouTube with a single real-valued label. For each model that is applicable to ISEAR, we show top 5 words of the generated label-specific topics in Table 4. It is worth to note that L-LDA and Dep-LDA achieve the same top words, since their difference only exists in the process of label prediction. The results indicate that although all models can learn meaningful topics, SLTM performs better than baseline models in label-specific topic discovery. For example, two words "happy" and "joy" which are strongly related to the label of "joy" are identified by SLTM with large probabilities. Similar results can be observed in other labels, thus topics discovered by our model are more convenient to be understood than others. Such a kind of performance enhancement is valuable to many real-world applications, e.g., personality education and psychotherapy, by producing human interpretable topics/events that evoke users' particular emotions.

For completeness, we also examine all topics generated by the baseline of sNTM. As mentioned earlier, sNTM is based on n-grams, instead of single words for SLTM and other baseline models. In the practical implementation, only unigrams and bigrams are considered since the embedding representation becomes less precise as n increases (Cao et al., 2015). The results indicate that sNTM can generate some topic bigrams such as "smelled disgusting" and "graduation exams", which are more appropriate to expressing a topic. However, only three topics are manually examined to be correlated with the seven emotions. This validates that sNTM is hard to introduce the guidance of labels in topic generation, because it models documents and labels separately.

Table 5: Classification performance on ISEAR.

	Accuracy	Cohen's kappa
SVM	0.5063	0.4240
fastText	0.5104	0.4298
LDA+softmax	0.1506	0.0089
sLDAc	0.1875	0.0540
L-LDA	0.4650	0.3758
Dep-LDA	0.4888	0.4036
sNTM	0.2478	0.1212
SLTM	**0.5213**	**0.4415**

To further evaluate the interpretability of topics extracted from SLTM, we firstly get topic embeddings by: $emb(z_k) = \mathbf{W}_1[k,:]$. Then, we map $emb(z_k)$ to a two-dimensional space via Principal Component Analysis (PCA). Figure 3 presents distributions of topics generated by SLTM over the ISEAR dataset. The scatter plot indicates that topics corresponding to the same label are closer than those of different labels. Furthermore, the distance between topics on correlated labels such as "fear" and "anger", is closer than that of topics on "joy" and other labels.

5.4 Evaluation on Label Prediction

We here evaluate the quality of word embeddings generated by SLTM on predicting categorical and real-valued labels based on ISEAR and YouTube, respectively. Since there are varied parameters for different models, we randomly select 60% of instances as the training set, 20% as the validation set, and the remaining 20% as the testing set. The values of parameters (e.g., the number of topics) for each model are all determined by the validation set. In label prediction, the main difference between SLTM and other supervised topic models is as follows. On one hand, a label-specific word embedding is introduced for predicting labels in SLTM according to Equation 10. On the other hand, other supervised topic models for both categorical and real-valued label prediction tasks infer labels for unlabeled documents by topic distributions directly, in which, topic distributions of unlabeled documents are learned without the supervision of labels.

For the task of categorical label prediction, the

Table 6: Regression performance on YouTube.

	MAE	pR^2
SVR	0.1424	-0.0591
HCNN	0.1112	0.3462
LDA+LR	0.1408	-0.0069
sLDA	0.1583	-0.2836
BP-sLDA	0.1394	-0.0208
sNTM	0.1342	0.0807
SLTM	**0.1005**	**0.4112**

accuracy and the Cohen's kappa score (Artstein and Poesio, 2008) are used as the evaluation metrics. Table 5 shows the classification performance of different models on ISEAR, where the best results are highlighted in boldface. For the prediction of real-valued labels on YouTube, we compare different models' regression performance by the mean absolute error (MAE) and the predictive R^2 (pR^2) (Blei and McAuliffe, 2007), as shown in Table 6. From the above results we can observe that SLTM achieves substantial performance improvement over baselines in predicting both categorical and real-valued labels, which indicates that word embeddings generated from labeled documents are more suitable for label prediction tasks than topic distributions generated from unlabeled documents without the guidance of labels.

5.5 Similarity of Word Embeddings

Word embeddings can reflect relations between words, and most methods of generating word embeddings are based on the local context information. This is because words with similar contexts may have similar semantics. However, a large-scale corpus is required to learn high quality word embeddings from the local context. Different from the previous word embedding generation methods, SLTM generates word embeddings based on the global label-specific topic information (i.e., the topical embedding space). Therefore, we further compare the quality of word embeddings learned by SLTM and three widely used methods: Word2Vec (W2V) (Mikolov et al., 2013), subword information Word2Vec (siW2V) (Joulin et al., 2017), and SSPMI (Levy and Goldberg, 2014). Among these baseline word embedding models, W2V and siW2V use the neural network framework, and SSPMI implicitly factorizes the pointwise mutual information (PMI) matrix of the local word co-occurrence patterns.

Table 7: Word similarity results on ISEAR.

	MEN	SimLex	Rare
W2V	0.002	-0.008	-0.119
siW2V	0.002	0.017	0.062
SSPMI	0.023	0.028	-0.004
SLTM	**0.169**	**0.037**	**0.089**

Table 8: Word similarity results on YouTube.

	MEN	SimLex	Rare
W2V	-0.018	0.004	-0.036
siW2V	-0.002	0.019	-0.051
SSPMI	-0.031	0.038	-0.026
SLTM	**0.048**	**0.040**	**0.068**

As our evaluation metric, the word similarity is estimated as follows. Firstly, we calculate cosine similarity scores for word pairs which occur in both the training set and the testing set. Secondly, word pairs are ranked according to their cosine similarities in the embedding space and human-assigned similarity scores, respectively. Finally, rankings of word similarity scores are evaluated by measuring the Spearman's rank correlation with rankings of human-assigned similarity scores. A higher correlation value indicates that it is more consistent to human judgements in word similarity. The following standard corpora which contain word pairs associated with human-assigned similarity scores are used for this evaluation: MEN (Bruni et al., 2014), SimLex-999 (SimLex) (Hill et al., 2015), and Rare (Luong et al., 2013).

We train W2V, siW2V, and SSPMI over each corpus by setting the number of context window size to 5. Furthermore, the dimension of word embeddings generated from all models is set to 50 according to (Lai et al., 2016). The values of word similarity on ISEAR and YouTube are respectively shown in Table 7 and Table 8, where the best results are highlighted in boldface. We can observe that SLTM outperforms baselines for all cases. The results indicate that word embeddings learned from the global label-specific topic information are better than those from the local context information without any external corpora.

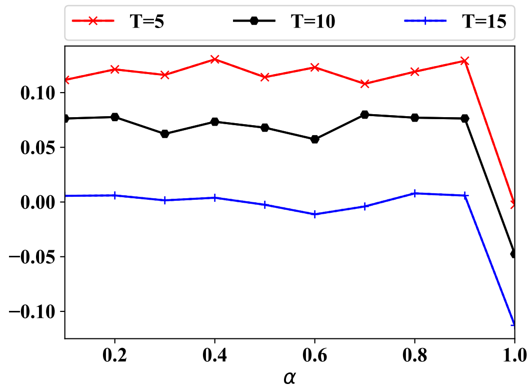

Figure 4: Topic coherence scores on ISEAR using different α values.

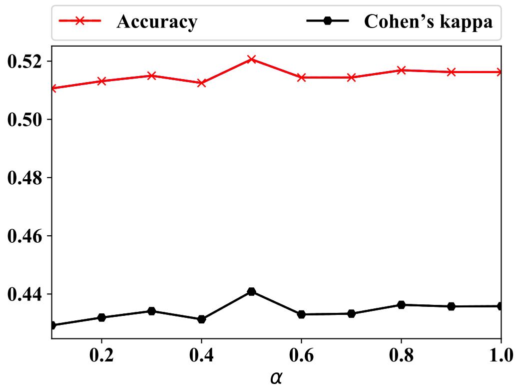

Figure 5: Label prediction performance on ISEAR using different α values.

5.6 Effect of the Hyper-parameter

After validating the effectiveness of SLTM on discovering topics and learning word embeddings, we now investigate the effect of the hyper-parameter in SLTM on these two aspects. According to Equation 8, the hyper-parameter α is used to weight two kinds of loss functions. Since $\mathbf{W}_2$ can be updated subject to $\alpha > 0$, we evaluate the performance of SLTM by varying α from 0.1 to 1 over the ISEAR dataset, as follows.

First, we evaluate the influence of hyper-parameter α on topic discovery by the coherence score of topics. To clearly illustrate the performance trend with different values of α, we set the number of top words T to 5, 10, and 15, and present topic coherence scores in Figure 4. The results indicate that SLTM performs stably under these α values on topic discovery, except for $\alpha = 1$ which ignores the label information totally. This validates the importance of label information in generating coherent topics.

Second, we use the learned word embeddings to predict document labels under different values of α. As shown in Figure 5, we can observe that when $\alpha = 0.5$, i.e., $loss(d_i, d_i^{(v_j-)})$ and $loss(\mathbf{y}_i, \mathbf{y}_i^{(v_j-)})$ are weighted equally, SLTM achieves the best performance in label prediction. The results indicate that the co-occurrence of documents and words as well as the label information are both important to generate good word embeddings. Furthermore, the label prediction performance of SLTM using any of these α values is better than that of most baselines (ref. Table 5). This validates the robustness of SLTM with different hyper-parameter values in supervised learning.

We also conduct experiments on YouTube using varied α values, which indicates that the hyper-parameter has a similar effect on both datasets.

6 Conclusion

In this paper, we proposed a supervised topic model named SLTM to discover label-specific topics by jointly modeling documents and labels. For the SLTM, weight matrices which represent document-topic and topic-word distributions can strictly follow probabilistic characteristics of topic models. Experiments were conducted on datasets with both categorical and real-valued labels, which validated that SLTM can not only discover more coherent topics, but also boost the performance of supervised learning tasks by learning high quality word embeddings. For future work, we plan to speed-up the training process of SLTM by GPUs and distributed algorithms. With the development of deep learning techniques, we also plan to de-emphasize irrelevant words with an attention mechanism.

Acknowledgments

We are grateful to the anonymous reviewers for their valuable comments on this manuscript. The research has been supported by the National Natural Science Foundation of China (61502545, U1711262, U1611264), a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (UGC/FDS11/E03/16), and the Innovation and Technology Fund (Project No. GHP/022/17GD) from the Innovation and Technology Commission of the Government of the Hong Kong Special Administrative Region.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 437–478. Springer-Verlag.

David M. Blei. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

David M. Blei and Jon D. McAuliffe. 2007. Supervised topic models. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 121–128.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent dirichlet allocation. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 601–608.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 737–744.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Artificial Intelligence Research*, 49:1–47.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2210–2216.

Huijun Chen, Xin Li, Yanghui Rao, Haoran Xie, Fu Lee Wang, and Tak-Lam Wong. 2017. Sentiment strength prediction using auxiliary features. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 5–14.

Jianshu Chen, Ji He, Yelong Shen, Lin Xiao, Xiaodong He, Jianfeng Gao, Xinying Song, and Li Deng. 2015. End-to-end learning of LDA by mirror-descent back propagation over a deep architecture. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 1765–1773.

Lei Cui, Dongdong Zhang, Shujie Liu, Qiming Chen, Mu Li, Ming Zhou, and Muyun Yang. 2014. Learning topic representation for SMT with neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 133–143.

Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 427–431.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 388–397.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 123–131.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 289–296.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 2042–2050.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 427–431.

Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 2177–2185.

Weifeng Li, Junming Yin, and Hsinchun Chen. 2018. Supervised topic modeling using hierarchical dirichlet process-based inverse regression: Experiments on e-commerce applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(6):1192–1205.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the 17th Conference on Computational Natural Language Learning*, pages 104–113.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 3111–3119.

Daniel Ramage, David Leo Wright Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 248–256.

Timothy N. Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. 2012. Statistical topic models for multi-label document classification. *Machine Learning*, 88(1-2):157–208.

Klaus R. Scherer and Harald G. Wallbott. 1994. Evidence for universality and cultural variation of differential emotion response patterning. *Personality and Social Psychology*, 66:310–328.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2):26–31.

Chong Wang, David M. Blei, and Fei-Fei Li. 2009. Simultaneous image classification and annotation. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1903–1910.

Zhongqing Wang and Yue Zhang. 2017. A neural model for joint event detection and summarization. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4158–4164.

Bernard Weiner and Sandra Graham. 1990. Attribution in personality psychology. *Handbook of personality: Theory and research*, pages 465–485.

Cort J Willmott and Kenji Matsuura. 2005. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30(1):79–82.

Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. 2013. Saliency detection via graph-based manifold ranking. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3166–3173.

GraphBTM: Graph Enhanced Autoencoded Variational Inference for Biterm Topic Model

Qile Zhu, Zheng Feng, Xiaolin Li
Large-scale Intelligent Systems Laboratory
NSF Center for Big Learning
University of Florida
{valder, fengzheng}@ufl.edu, andyli@ece.ufl.edu

Abstract

Discovering the latent topics within texts has been a fundamental task for many applications. However, conventional topic models suffer different problems in different settings. The Latent Dirichlet Allocation (LDA) may not work well for short texts due to the data sparsity (i.e., the sparse word co-occurrence patterns in short documents). The Biterm Topic Model (BTM) learns topics by modeling the word-pairs named biterms in the whole corpus. This assumption is very strong when documents are long with rich topic information and do not exhibit the transitivity of biterms. In this paper, we propose a novel way called GraphBTM to represent biterms as graphs and design Graph Convolutional Networks (GCNs) with residual connections to extract transitive features from biterms. To overcome the data sparsity of LDA and the strong assumption of BTM, we sample a fixed number of documents to form a mini-corpus as a training instance. We also propose a dataset called $All\ News$ extracted from (Thompson, 2017), in which documents are much longer than 20 Newsgroups. We present an amortized variational inference method for GraphBTM. Our method generates more coherent topics compared with previous approaches. Experiments show that the sampling strategy improves performance by a large margin.

1 Introduction

Topic model (Blei et al., 2003) is one of the most popular approaches to learn hidden representations of text. The broad applications of topic model range from recommender systems (Wang and Blei, 2011), computer vision (Fei-Fei and Perona, 2005), to bioinformatics (Rogers et al., 2005). Conventional topic models learning approaches are based on Gibbs Sampling (Griffiths and Steyvers, 2004) or Variational Expectation Maximization (VEM) algorithm (Blei et al., 2003). Both Gibbs Sampling and VEM are not directly applicable to new variations of the topic model. Specifically, the inference algorithm requires re-deriving for any minor changes to the model.

Recently, a neural network based topic model inference approach, the Autoencoded Variational Inference for Topic Model (AVITM), was proposed by (Srivastava and Sutton, 2017). This approach uses an inference network to directly map a document to its posterior distribution without any variational update steps. The proposed inference network is based on the Autoencoding Variational Bayes (AEVB) (Kingma and Welling, 2013), a stochastic variational inference algorithm over neural networks. Compared with the sampling based approaches, AVITM can scale to large datasets. Although it improves the model's robustness and reduces the computational cost, it still suffers from the data sparsity in short texts.

Biterm Topic Model (BTM) proposed by (Cheng et al., 2014) and (Yan et al., 2013) addresses the shortcoming of data sparsity for modeling the corpus of short texts. It explicitly models the patterns on top of word co-occurrence feature (Biterm, the unordered word-pair occurs in texts) from the corpus. It holds one topic distribution for the whole corpus rather than one document. BTM, therefore, is suitable for modeling short documents like tweets, and online QA texts. It also achieves better results than LDA in specific scenarios of the normal texts (Yan et al., 2013). However, using one topic distribution for all documents limits the model's expressiveness when the documents contain diverse topics.

To address the issue of data sparsity of LDA when modeling the short texts and the insufficient corpus-wise topic representation in BTM for normal texts, we strike a balance between these two approaches. Instead of modeling biterms in the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4663–4672
Brussels, Belgium, October 31 - November 4, 2018.

whole corpus, we extract biterms inside a fixed-length text window for every document and sample n documents to form an instance each time. As a result, we enhance the input feature with biterms that capture more word co-occurrence patterns than BOW and also avoid the insufficient corpus-wised topic representation in BTM. Another advantage of biterms is the transitivity. For example, we have two biterms (A, B) and (A, C). It is natural to think that B and C may share some similarities. This transitivity is similar to the graph structure data. So we model the biterms in a graph where the words are taken as nodes and the counts of biterms are the weight of edges. We extract the information from biterms explicitly by the graph convolutional network (Kipf and Welling, 2016).

In this paper, we propose a novel Graph-based inference network for the biterm topic model (GraphBTM). To the best of our knowledge, GraphBTM is the first AEVB inference approach for the biterm topic model with graph enhanced feature. Our model also strikes a good balance between LDA and BTM, leverages both advantages, and achieves better topics coherence scores than AVITM in two datasets. The main contributions of our work include:

- We are the first to apply the neural network based inference approach for the Biterm Topic Model, and achieve better results in *topic coherence* score than previous AEVB based inference method (AVITM) and online Variational Inference LDA.

- We propose a data argumentation method to enhance the input feature with word correlation from biterms in normal text and overcome the shortcoming of the data sparsity in LDA and the insufficient corpus-wise topic representation in BTM.

- We model the biterms as an undirected graph and adopt a novel graph convolutional network to encode word co-relationship in our inference network.

- We introduce a new dataset *All News* dataset containing 20,000 documents extracted from 15 news publishers (Thompson, 2017) for topic modeling. TThe documents are much longer compared with the 20 Newsgroups.

2 Background

2.1 Biterm Topic Model

BTM (Cheng et al., 2014) is proposed to solve the data sparsity problem in the scenario of short texts. Instead of modeling a single document, BTM considers the whole corpus as a mixture of topics. BTM collects all unordered word-pairs (biterms) from each short text or a fixed-length text window of normal texts. The generative progress of BTM can be described as follows, where α and β are two parameters of Dirichlet priors.

1. For each topic z
   (a) draw a topic-specific word distribution $\phi_z \sim Dir(\beta)$
2. Draw a topic distribution $\theta \sim Dir(\alpha)$ for the whole corpus
3. For each biterm b in the biterm set
   (a) draw a topic assignment $z \sim Multi(\theta)$
   (b) draw two words: $w_i, w_j \sim Multi(\phi_z)$

With the procedure above, the joint probability of a biterm $b = (w_i, w_j)$ can be written as:

$$P(b) = \sum_z P(z)P(w_i|z)P(w_j|z) \quad (1)$$

$$= \sum_z \theta_z \phi_{i|z} \phi_{j|z} \quad (2)$$

So the likelihood of the whole corpus B is:

$$P(B|\alpha, \beta) = \prod_{i,j} \sum_z \theta_z \phi_{i|z} \phi_{j|z} \quad (3)$$

2.2 Laplace Approximation of Dirichlet

Both LDA and BTM use the Dirichlet prior over the topic and word proportions. Wallach et al. (2009) showed that the Dirichlet prior is important to producing interpretable topics. However, it is hard to apply the Dirichlet prior to AEVB directly. AEVB uses the reparameterization trick (RT) to obtain a differentiable Monte Carlo estimator for the variational lower bound (details can be found in next section), and it is difficult to propose an effective RT for the Dirichlet prior.

Fortunately, we can approximate Dirichlet distribution with a logistic normal in the $softmax$ $basis$ by Laplace approximation (Hennig et al.,

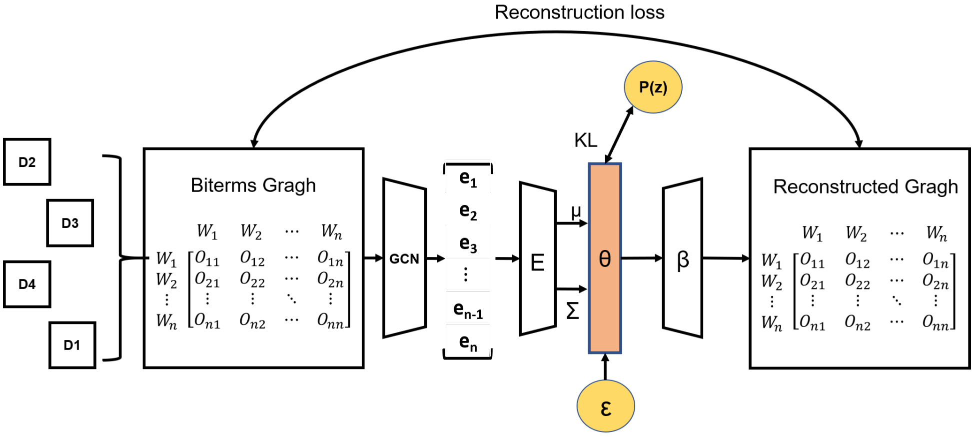

Figure 1: The overall structure of our proposed model GraphBTM. In this example, we sample 4 documents at a time and embed the aggregated biterm graph by GCNs. The graph embedding is sent to inference network E to produce the parameters for our variational distribution. We then use RT to generate the Monte Carlo samples. At last, we use the decoder network β to get the word probabilities and reconstruct the aggregated graph.

2012). MacKay (1998) gives the Dirichlet probability density function in the softmax basis over the variable x:

$$P(\pi|\alpha) = \frac{\Gamma(\sum_k^K(\alpha_k))}{\prod_k^K \Gamma(\alpha_k)} \prod_k \pi_k^{\alpha_k} g(1^T x) \tag{4}$$

where $\pi = \sigma(x)$ (softmax) and $g(1^T x)$ is an arbitrary density for integrability. Hennig et al. (2012) argued that the Eq. 4 could be approximately independent for large K (number of topics). So the covariance matrix of the Dirichlet prior becomes a diagonal matrix for large K. By this way, we can approximate the Dirichlet distribution with a multivariate normal with mean μ_k and covariance matrix Σ_{kk}:

$$\mu_k = log\alpha_k - \frac{1}{K}\sum_i^K log\alpha_i \tag{5}$$

$$\Sigma_{kk} = \frac{1}{\alpha_k}(1 - \frac{2}{K}) + \frac{1}{K^2}\sum_i^K \frac{1}{\alpha_i} \tag{6}$$

with this approximation in hand, we can easily apply RT by sampling from $\epsilon \sim \mathcal{N}(0, I)$ and compute probability $\pi_k = \sigma(\mu_k + \Sigma_{kk}^{1/2}\epsilon)$.

3 Graph Biterm Topic Model

Before getting into details of GraphBTM, we give an overall structure of GraphBTM, as shown in Fig. 1. We extract the biterms from a mini-corpus (aggregated sampled documents) and embed the whole biterm graph into a fixed length vector with the dimension of vocabulary size. Then we use this graph embedding as the input of our inference network and get the topic proportion. At last, we use the decoder network to get the word probabilities and reconstruct the biterm graph.

3.1 Model Biterms as Graphs

Commonly used input feature of topic models is bag-of-words (BOW) which implicitly capture the word co-occurrence patterns. BTM models the word co-occurrence explicitly by directly counting the word-pairs in a text window. However, using one-hot encoding for biterms may lose the transitive co-relations. We model collected biterms as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ (words as nodes and $|V|$ is the vocabulary size) and $\mathcal{E}$ (counts of corresponding biterms in the sample) are sets of nodes and edges, respectively. In this way, the adjacency matrix A ($A \in \mathbb{R}^{V \times V}$) denotes the counts of biterms in the sample. We also leverage the matrix A as the node feature matrix (A_i is the node feature for the word w_i).

We use GCNs proposed by (Kipf and Welling, 2016), which is a framework used for learning the graph structure data. Gilmer et al. (2017) presented a comprehensive overview. Consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a matrix $X \in \mathbb{R}^{n \times m}$ in where each row is a node feature $x_v \in \mathbb{R}^m$ ($v \in \mathcal{V}$). One layer GCN encodes information of a node with its immediate neighbors, defined as

$$h^{l+1} = f\left(\tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}(h^l W^l + b)\right) \tag{7}$$

where h^0 is the input features X, $\tilde{A} = A + I_N$

is the adjacency matrix of the graph with self-connections, I_N is the identity matrix, $\tilde{D}$ is the degree matrix of $\tilde{A}$, W^l is a trainable weight matrix in the layer and b is the bias. f denotes a non-linear activation function, such as ReLU. By stacking GCN layers, we can incorporate higher order neighborhoods. To represent the whole graph, we reduce the dimension of each node to one by using GCNs and concatenate them as the final representation of the biterm graph.

From another point of view, we can treat GCNs as a Laplacian smoothing. Repeatedly applying Laplacian smoothing may mix the features of vertices and make them indistinguishable (Li et al., 2018). On the other hand, the transitivity of words may not be meaningful when the number of hops (layers) of GCN increases. We solve this problem by adding shortcut connections between different layers inspired by Residual Networks (He et al., 2016). What's more, a recent study showed that adding the residual connection can help convergence (Li and Yuan, 2017).

3.2 AEVB for Biterm Graphs

Eq. 3 gives us the likelihood of the whole corpus based on $Multi(\phi_z)$. Here we rewrite the Eq. 3 with latent variables as

$$p(B|\alpha,\beta) = \int_\theta \left(\prod_{(i,j)} \sum_{z=1}^{k} \pi_i \pi_j p(z_n|\theta) \right) p(\theta|\alpha) d\theta \tag{8}$$

where $\pi_i = p(w_i|z_n,\beta)$. The inference of posterior $p(\theta, z|B,\alpha,\beta)$ over the hidden variables θ and z is intractable (Dickey, 1983). Many methods are proposed to solve this inference problem including Gibbs Sampling (Griffiths and Steyvers, 2004) and variational inference methods. Gibbs Sampling based approaches are computationally inefficient and varitional inference methods like mean field (Blei et al., 2003) scarify the topic quality for computational efficiency. Moreover, the major problem of these approximate inference algorithms is the inflexibility. Slight changes in model assumption may require designing a new inference algorithm. To alleviate this problem, we design an amortized approximate inference method similar to AVITM (Srivastava and Sutton, 2017). It is more flexible compared with other approximate inference methods and can be applied to any biterm graphs.

In Eq. 8, there are two latent variables θ and z, we introduce two free variational parameters γ over θ and ϕ over z. Our goal is to approximate the true posterior $p(\theta, z|B,\alpha,\beta)$ with variational distribution $q(\theta, z|\gamma,\phi) = q_\gamma(\theta)\prod_k q_\phi(z_k)$. Then we can transfer the inference problem as an optimization problem (Blei et al., 2003), which is to maximize

$$\begin{aligned} L(\gamma,\phi|\alpha,\beta) &= logp(B|\alpha,\beta) \\ &\quad - D_{KL}[q(\theta,z|\gamma,\phi)||p(\theta,z|B,\alpha,\beta)] \end{aligned} \tag{9}$$

L is a lower bound to the marginal log likelihood (ELBO). Following AEVB (Kingma and Welling, 2013), we rewrite the ELBO as

$$L(\gamma,\phi|\alpha,\beta) = -D_{KL} + R \tag{10}$$

where $R = E_q[logp(B|z,\theta,\alpha,\beta)]$. This form is intuitive. The first term is the KL divergence betweent the variational distribution and the prior on the latent variables, and the second term ensures that the latent variables are good at explaining and reconstructing the input data.

We use a neural network named $inference$ $network$ to compute the variational parameters. It takes the embedding of the biterm graph (sec. 3.1) as the input and outputs the parameters of the variational distribution. So the inference network can be defined as $(\mu_b, \Sigma_b) = f(b,\gamma)$, where μ_b and Σ_b are vectors of length k (topic numbers) and γ are the network parameters. In our setting, we use the logistic normal distribution which is an approximation of the Dirichlet prior to the variational distribution. We can choose the corresponding variational distribution $q_\gamma(\theta) = \mathcal{LN}(\theta|\mu_b, diag(\Sigma_b))$, where $diag(\cdot)$ converts a column vector to a diagonal matrix. One important advantage of using AEVB is that we couple the variational parameters for different inputs, unlike mean field variational inference, because they are computed from the same network.

Next is how to compute the expectations respect to q in R (Eq. 10). Kingma and Welling (2013) use a differential Monte Carlo estimator with the reparameterization trick. With RT, instead of sampling from the variational distribution directly, we sample from a simple distribution that is independent of all variational parameters. In this way, the gradient can be backpropagated through the variational parameters. For the logistic normal distribution, we can sample from a standard normal distribution $\epsilon \in \mathcal{N}(0, I)$.

Although the reparameterization trick helps us deal with θ, it is hard to deal with the discrete variable z. Fortunately, we can collapse the discrete variables z and only infer θ with collapsed inference method (Kurihara et al.) as

$$p(B|\alpha,\beta) = \int_{\theta}\left(\prod_{(i,j)}\pi_i\pi_j\right)p(\theta|\alpha)d\theta \qquad (11)$$

where $\pi_i = p(w_i|\beta,\theta)$, which is the probability of one word in the biterm. Now we only need to sampling from θ.

We can now get our final variational objective function as (to minimize the negative ELBO)

$$L = D_{KL} - E_{\epsilon}[\sum G_b \circ log(P^T P)] \qquad (12)$$

where G_b is the input biterm graph, $P = \sigma(\beta)\sigma(\mu + \Sigma^{1/2}\epsilon)$ is probabilities for all the words based on the input graph and $\circ$ denotes the element-wise production. The KL divergence between two logistic normal distritbutions are

$$\begin{aligned} D_{KL} =& \frac{1}{2}\{tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T\Sigma_1^{-1}(\mu_1 - \mu_0) \\ & - K + log\frac{|\Sigma_1|}{|\Sigma_0|}\} \end{aligned} \qquad (13)$$

3.3 Sample Mini-corpus

To alleviate the data sparsity problem of LDA (Zhu and Xing, 2012; Lin et al., 2014), BTM learns topics from the aggregated patterns in the whole corpus. In our observation, this assumption is too strong for normal texts. Other than BTM, some approaches in the literature addressed this problem by aggregating documents into a mini-corpus before training the topic model. For example, in tweets analysis, Weng et al. (2010) aggregated the tweets from one user into a document. Hong and Davison (2010) combined the tweets containing the same word. Inspired by these strategies, we make the same assumption for normal text. We first extract all the biterms in each document and randomly select n documents in the dataset as a mini-corpus. The biterms of the mini-corpus simply merge all the biterms from the n documents. Experiments show that a proper sampling number achieves the best performance.

3.4 Unnormalize the β

The topic-word distribution β is a mixture of multinomials. One drawback of this formulation is that it cannot predict something that is sharper than the distributions being mixed (Hinton and Salakhutdinov, 2009). This problem may result in some poor quality topics. Previous research (Srivastava and Sutton, 2017) shows that unnormalizing the parameters β and changing the conditional distribution of w_n as $w_n|\beta,\theta \sim Multinomial(1, \sigma(\beta\theta))$ can solve this problem.

With the unnormalized β, we can model it as a *decoder network* whose weight matrix $M = (m_1, \ldots, m_K)$ denotes the weight for all words under K topics. Applying softmax to row m_i will give us the probabilities under topic i.

4 Experiments

4.1 Datasets and Settings

We demonstrate our model on two datasets: 20 Newsgroups and *All News*. For All News, we use the data from kaggle collection (Thompson, 2017) [1], which collects documents from 15 main news publishers between 2016 and July 2017. Among these, we randomly select 20,000 documents. In our preprocessing of the texts, we follow the steps of tokenization, filtering out stop words, and non-UTF-8 characters in (Srivastava and Sutton, 2017). From the statistics summarized in Table 1. We can know that the 20 Newsgroups dataset is relatively sparse and the All News has rich information. The ratio of text lengths less or equal to 30 of the 20 Newsgroups dataset is 28%, which only 2% in the All News dataset. The average size differs a lot between these two datasets: 302 for the All News and 88 for the 20 Newsgroups.

For the generation of the matrix W_g, the selection of the window size of words is critical, a small size of window leads to very sparse W_g. Here, we choose an experience value 30 for the window size follows the (Yan et al., 2013). For the logistic normal approximation, we use the Dirichlet distribution with parameter α as 0.02. Our GraphBTM approach, including the GCN layers and the inference network are implemented with Pytorch-v0.4.0 (Paszke et al.). Parameters in our implemented model are optimized by the stochastic optimizer Adadelta (Zeiler, 2012) with learning rate 1. To embed the biterm graph, we use a 3-layer GCNs with size 1995-100, 100-100 and 100-1 for 20 Newsgroups and 5000-1000, 1000-100, 100-1 for ALL News. We use the *edge dropout* in GCN: when computing h^l, we ignore each node with a probability of 0.6. We use batch normalization

---

[1] https://www.kaggle.com/snapcrack/all-the-news

Datasets	Training instances	Ratio of <= 30	Avg size	Vocabulary	Avg Biterms #
20 News	11,259	28%	88	1995	1249
All News	20,000	2%	302	5000	5535

Table 1: Datasets statistics.

Dataset	# topics	GraphBTM	AVITM	LDA Online VI
20 News	50	**0.28**	0.25	0.10
	100	**0.26**	0.23	0.08
All news	50	**0.27**	0.24	0.14
	100	**0.26**	0.23	0.13

Table 2: Average topic coherence.

Datasets(# Topics)	# Samples	Score
20 News (k=50)	1	0.24
	3	**0.28**
	10	0.25
20 News (k=100)	1	0.21
	3	**0.26**
	10	0.25
All news (k=50)	1	**0.27**
	3	0.22
	5	0.20
All news (k=100)	1	**0.26**
	3	0.20
	5	0.17

Table 3: Results for different sampling numbers in different setting for the two datasets. Score denotes the topic coherence score.

(Ioffe and Szegedy, 2015) in $inference\ network$ with batch size 100 as same in (Srivastava and Sutton, 2017). We run each model 10 times and take the average results. Code is available at `https://github.com/valdersoul/GraphBTM`.

The perplexity has been used in the past works to measure the quality of the generated topics. However, the perplexity is not shown to be a good evaluation metric for the topics (Newman et al., 2010). What's more, our method models a mini-corpus instead of a real one and infer the topics through the pattern of biterms, so the perplexity is not suitable to measure the performance of our approach. To get a more objective measurement of the topics, we adopt "topic coherence" as our metric, proposed by (Mimno et al., 2011) to evaluate the quality of the topics. For a vector $V^{(z)} = (v_1^z, ..., v_T^z)$ as the top T words of the topic z, which ordered by the probability $p(w|z)$, the topic coherence is defined as:

$$C(z; V^{(z)}) = \sum_{t=2}^{T} \sum_{l=1}^{t} log \frac{D(v_m^{(z)}, v_l^{(z)}) + 1}{D(v_l^{(z)})}. \quad (14)$$

where $D(v)$ is the number of the documents that word v occurred, and $D(v, v')$ is the number of the documents that both the words v and v' occurred. The assumption of the topic coherence is the words with high frequency in a topic tend to appear in the same document. This measurement has been demonstrated to be highly consistent with the human evaluated quality of the topics.

4.2 Results and Discussions

Comparison with other approaches. We compare our GraphBTM approach with the AVITM (Srivastava and Sutton, 2017) and the LDA model (Blei et al., 2003). For AVITM, we use their results for 20 Newsgroups directly and run the model using the provided code[2]. We use the online variational inference for LDA (Hoffman et al., 2010) implementation by gensim library (Řehůřek and Sojka, 2010) as our LDA baseline. For both GraphBTM and AVITM, we run 200 iterations. Table 2 shows the average topic coherence for three models on the two datasets. The online VI LDA works worst in the three models and we find that on both datasets, the GraphBTM consistently outperforms other two models. We can verify the quality of the learned topics by displaying the topic examples in Table 4. The topics from GraphBTM are more coherent than the topics from both AVITM and LDA model.

---

[2] https://github.com/akashgit/autoencoding_vi_for_topic_models

Effect of the mini-corpus. To study the effect of our sampling strategy which has been discussed in section 3.3. Table 3 shows the performance of our model with different sample size for a mini-corpus. For the 20 Newsgroups dataset, the best performance is achieved when the sample size is 3. When we do not use our sample strategy (mini-corpus is 1), the performance drops by a large margin. From Table 1, we see that the average size of documents size in 20 Newsgroups is relatively short (88 compared with 302 in All News). Therefore, the 20 Newsgroups dataset may suffer from the sparsity problem. The experiment shows that our sampling strategy can help to overcome this problem. When sample size increases, the performance drops again. The biterm graph with large sample size may bring the same problem of the original BTM (insufficient topic representation). Compared to 20 Newsgroups dataset, documents in the All News dataset is longer and carried more topic information, so the best performance is achieved without sampling. We find that when the sample size is larger than an optimized value, the topic coherence starts to drop.

Effect of modeling biterms as graphs. To verify the effect of the graph modeling of biterms. We also do experiments on AVITM with the same sampling strategy. We use the sampling size 3 which achieves the best performance by GraphBTM in 20 Newsgroups to train AVITM model. The performance does not change a lot, with an average score of 0.25 which is the same as the score without sampling. It is not surprising to us because AVITM models the topic directly on the individual document with BOW feature. The BOW feature captures the word co-occurrence implicitly. So aggregating documents in AVITM can not enhance the input feature. However, our model uses GCN to capture the transitivity of biterms and can benefit from the sampling strategy a lot.

Residual connection. We add the residual connection between the first and second layer of GCN. On the other hand, it can also help convergence by adding residual connection (Li and Yuan, 2017). The residual can also help the network capture hierarchical information of the biterms. We remove the residual connection with the same setting which achieves the best performance in these two datasets and results in a **0.1** drop in performance.

5 Related Work

In this section, we briefly summarize the related work of the topic model into two categories: normal texts and short texts.

5.1 Normal Texts

The effort of uncovering the latent semantic representation of documents can be dated from the Latent Semantic Analysis (LSA) (Deerwester et al., 1990), which used the singular value decomposition of the document matrix to get the word patterns. The probabilistic latent semantic analysis (PLSA) (Hofmann, 1999) improved the LSA model by adding a probabilistic model based on a mixture decomposition. It assumed that a document could be presented as a mixture of topics and a topic is a distribution over words. LDA added the Dirichlet priors on topic and word distributions and proposed a complete generative model.

With the rising of deep learning (LeCun et al., 2015), researchers achieve significant improvement in many areas including image classification (He et al., 2016), speech recognition (Hinton et al., 2012) and named entity recognition (Ma and Hovy, 2016; Zhu et al., 2018). Many attempts have been made for topic models based on neural networks (Hinton and Salakhutdinov, 2009; Cao et al., 2015; Miao et al., 2016; Srivastava and Sutton, 2017). Cao et al. (2015) embedded multinomial relationships between documents, topics, and words in differentiable functions. However, they lost the stochasticity and Bayesian inference of prior functions. Miao et al. (2016) introduced the Neural Variational Document Model (NVDM), which used Gaussian distribution over topics and averaged over topic-word distribution in the logit space. Although they used the black-box variational inference (VAE), they did not approximate the Dirichlet prior. Srivastava and Sutton (2017) approximated the Dirichlet prior with logistic Gaussian using the Laplace approximation of Hennig et al. (2012) and collapsed the hidden topic value z with a mixture of experts (Hinton, 2002). This model (AVITM) significantly improved the topic coherence compared with the NVDM model. However, same as the LDA, AVITM suffers from the data sparsity problem.

Model	Topics
GraphBTM	attack ripem rsa encrypt cipher random key cryptography distribution encryption turkish turks greek greece armenian genocide turkey armenia armenians island season score player league game puck pitch win pitcher team israel lebanese israeli lebanon village attack zone arab territory civilian oname printf entry buf char contest stream output int remark
AVITM	ripem anonymous pgp rsa posting cipher atheism encrypt usenet atheist armenian genocide turks turkish muslim massacre turkey armenians armenia greek season nhl team hockey playoff puck league flyers defensive player israel israeli lebanese arab lebanon arabs civilian territory palestinian militia oname printf buf entry os char contest cpu stream remark
LDA	drug food health research medical test used development product computer system data software business personal ibm information technology offering common convertible proceeds co due used public filed agreement agreed acquisition acquire purchase sell subject subsidiary completed quarter earnings reported income expects fiscal loss per second

Table 4: Five selected topics from all models.

5.2 Short Texts

Early studies on short text topic model mainly focused on adding external knowledge to enrich the information of short texts. Phan et al. (2008) firstly learned hidden topics from substantial external resources to enrich the features in short text. Jin et al. (2011) leveraged the power of transfer learning to learn topics on short texts from auxiliary long text data. However, external knowledge in some domain may not be available.

Instead of adding external knowledge, one potential way is to add a sparse prior on the topic distribution. Chien and Chang (2014) used a spike model to control the sparsity of selected topics. Lin et al. (2014) used the same idea to add the sparsity on both topic and word distribution. Different from these approaches, some researchers tried to enhance data without external knowledge. Weng et al. (2010) aggregated the tweets from one user into a document. Hong and Davison (2010) combined the tweets containing the same words. Some other used non-probability topic model to solve this problem. Zhu and Xing (2012) proposed sparse topical coding, which relaxed the normalization constraint of admixture proportions and learned hierarchical latent representations.

6 Conclusion and Future Work

We proposed a Graph Enhanced Autoencoding Variational inference for Biterm Topic Model (GraphBTM). Our model used a black-box approximation inference approach to learn topics through the word co-occurrences (biterms). We modeled the biterms in the form of a graph where the nodes are the words and weighted edges are the counts of the corresponding biterms. On top of this graph representation, we designed a model by GCN layers with a residual connection to effectively extract node representations that preserve the missing connectivity. To overcome the problems of data sparsity in LDA and insufficient topic representation in BTM, we introduced a data argumentation approach by producing a mini-corpus with sampled documents. By setting a proper hyperparameter of sample size k, we achieved better topic coherence scores compared with previous works.

Our GCN model is based on spectral graph convolutions, which requires computing the graph Laplace for each sample. Compared to tasks with one graph input, we need to compute the graph Laplace for every input sample, causing substantial computational cost. It is critical to developing a memory efficient processing and storage strategy to handle the large-scale graph data when we generalize GraphBTM to complex tasks. Recently, fastGCN (Zhang et al., 2018) interpreted graph convolutions as integral transforms of functions under probability measures. Our following work will consider adopting fastGCN to speed up the process.

Acknowledgments

This work was partially supported by National Science Foundation (grant CNS-1842407), National Institutes of Health (grant R01GM110240), and industry members of NSF Center for Big Learning (`http://nsfcbl.org/index.php/partners/`).

References

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension.

Xueqi Cheng, Xiaohui Yan, Yanyan Lan, and Jiafeng Guo. 2014. Btm: Topic modeling over short texts. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):2928–2941.

Jen-Tzung Chien and Ying-Lan Chang. 2014. Bayesian sparse topic model. *Journal of Signal Processing Systems*, 74(3):375–389.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

James M Dickey. 1983. Multiple hypergeometric functions: Probabilistic interpretations and statistical uses. *Journal of the American Statistical Association*, 78(383):628–637.

Li Fei-Fei and Pietro Perona. 2005. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531. IEEE.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Philipp Hennig, David Stern, Ralf Herbrich, and Thore Graepel. 2012. Kernel topic models. In *Artificial Intelligence and Statistics*, pages 511–519.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2009. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*, pages 1607–1614.

Matthew Hoffman, Francis R. Bach, and David M. Blei. 2010. Online learning for latent dirichlet allocation. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 856–864. Curran Associates, Inc.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc.

Liangjie Hong and Brian D Davison. 2010. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*, pages 80–88. ACM.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ou Jin, Nathan N Liu, Kai Zhao, Yong Yu, and Qiang Yang. 2011. Transferring topical knowledge from auxiliary long texts for short text clustering. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 775–784. ACM.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kenichi Kurihara, Max Welling, and Yee Whye Teh. Collapsed variational dirichlet process mixture models.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. *arXiv preprint arXiv:1801.07606*.

Yuanzhi Li and Yang Yuan. 2017. Convergence analysis of two-layer neural networks with relu activation. In *Advances in Neural Information Processing Systems*, pages 597–607.

Tianyi Lin, Wentao Tian, Qiaozhu Mei, and Hong Cheng. 2014. The dual-sparse topic model: mining focused topics and focused terms in short text. In *Proceedings of the 23rd international conference on World wide web*, pages 539–550. ACM.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

David JC MacKay. 1998. Choice of basis for laplace approximation. *Machine learning*, 33(1):77–86.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the conference on empirical methods in natural language processing*, pages 262–272. Association for Computational Linguistics.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108. Association for Computational Linguistics.

Adam Paszke, Soumith Chintala, Ronan Collobert, Koray Kavukcuoglu, Clement Farabet, Samy Bengio, Iain Melvin, Jason Weston, and Johnny Mariethoz. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration, may 2017.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web*, pages 91–100. ACM.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Simon Rogers, Mark Girolami, Colin Campbell, and Rainer Breitling. 2005. The latent process decomposition of cdna microarray data sets. *IEEE/ACM transactions on computational biology and bioinformatics*, 2(2):143–156.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488*.

Andrew Thompson. 2017. All the news: 143,000 articles from 15 american publications. =https://www.kaggle.com/snapcrack/all-the-news.

Hanna M Wallach, David M Mimno, and Andrew McCallum. 2009. Rethinking lda: Why priors matter. In *Advances in neural information processing systems*, pages 1973–1981.

Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM.

Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. 2010. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456. ACM.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*.

Jun Zhu and Eric P Xing. 2012. Sparse topical coding. *arXiv preprint arXiv:1202.3778*.

Qile Zhu, Xiaolin Li, Ana Conesa, and Ccile Pereira. 2018. Gram-cnn: a deep learning approach with local context for named entity recognition in biomedical text. *Bioinformatics*, 34(9):1547–1554.

Modeling Online Discourse with Coupled Distributed Topics

Akshay Srivatsan[†] **Zachary Wojtowicz**[‡] **Taylor Berg-Kirkpatrick**[†]

[†]Language Technologies Institute, Carnegie Mellon University, `{asrivats,tberg}@cs.cmu.edu`
[‡]Social and Decision Sciences, Carnegie Mellon University, `zdw@andrew.cmu.edu`

Abstract

In this paper, we propose a deep, globally normalized topic model that incorporates structural relationships connecting documents in socially generated corpora, such as online forums. Our model (1) captures discursive interactions along observed reply links in addition to traditional topic information, and (2) incorporates latent distributed representations arranged in a deep architecture, which enables a GPU-based mean-field inference procedure that scales efficiently to large data. We apply our model to a new social media dataset consisting of 13M comments mined from the popular internet forum Reddit, a domain that poses significant challenges to models that do not account for relationships connecting user comments. We evaluate against existing methods across multiple metrics including perplexity and metadata prediction, and qualitatively analyze the learned interaction patterns.

1 Introduction

Topic models have become one of the most common unsupervised methods for uncovering latent semantic information in natural language data, and have found a wide variety of applications across the sciences. However, many common models - such as Latent Dirichlet Allocation (Ng and Jordan, 2003) - make an explicit exchangeability assumption that treats documents as independent samples from a generative prior, thereby ignoring important aspects of text corpora which are generated by non-ergodic, interconnected social systems. While the direct application of such models to datasets such as transcripts of The French Revolution (Barron et al., 2017) and discussions on Twitter (Zhao et al., 2011) have yielded sensible topics and exciting insights, their exclusion of document-to-document interactions imposes limitations on the scope of their applicability and the analyses they support. For instance, on many social media platforms, comments are short (the average Reddit comment is 10 words long), making them difficult to treat as full documents, yet they do cohere as a collection, suggesting that contextual relationships should be considered. Moreover, analysis of social data is often principally concerned with understanding relationships between documents (such as question-asking and -answering), so a model able to capture such features is of direct scientific relevance.

To address these issues, we propose a design that models representations of comments jointly along observed reply links. Specifically, we attach a vector of latent binary variables to each comment in a collection of social data, which in turn connect to each other according to the observed reply-link structure of the dataset. The inferred representations can provide information about the rhetorical moves and linguistic elements that characterize an evolving discourse. An added benefit is that while previous work such as Sequential LDA (Du et al., 2012) has focused on modeling a linear progression, the model we present applies to a more general class of acyclic graphs such as tree-structured comment threads ubiquitous on the web.

Online data can be massive, which presents a scalability issue for traditional methods. Our approach uses latent binary variables similar to a Restricted Boltzmann Machine (RBM); related models such as Replicated Softmax (RS) (Salakhutdinov and Hinton, 2009) have previously seen success in capturing latent properties of language, and found substantial speedups over previous methods due to their GPU amenable training procedure. RS was also shown to deal well with documents of significantly different length, another key characteristic of online data. While RBMs permit exact inference, the additional coupling potentials present in our model make inference intractable. However, the choice of bilinear potentials and latent

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4673–4682
Brussels, Belgium, October 31 - November 4, 2018.

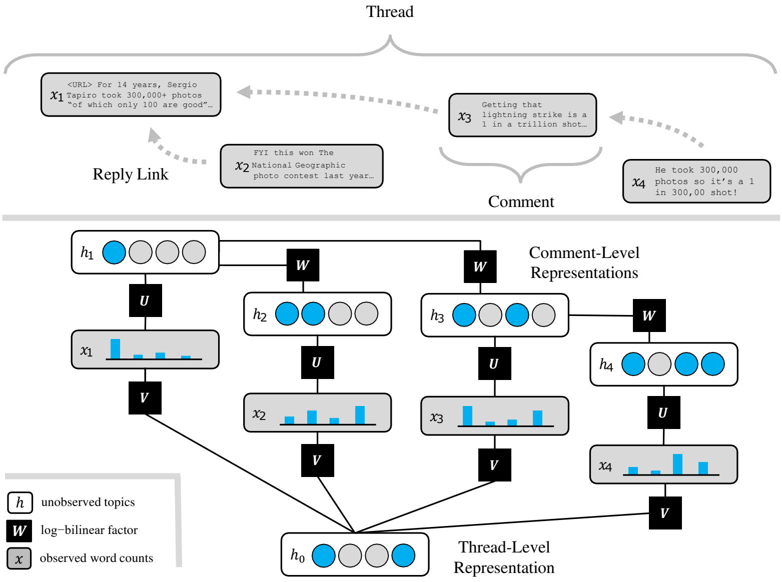

<URL> For 14 years, Sergio Tapiro took 300,000+ photos "of which only 100 are good"…

FYI this won The National Geographic photo contest last year…

Getting that lightning strike is a 1 in a trillion shot…

He took 300,000 photos so it's a 1 in 300,00 shot!

Figure 1: DDTM factor graph for an example thread. Each comment is modeled as an observed bag-of-words x with topics represented by a latent binary vector h. Log-bilinear factors connect the latent and observed variables of each comment, and the latent variables of parent-child comment pairs along observed reply links. Biases are omitted for clarity.

features admits a mean-field inference procedure which takes the form of a series of dense matrix multiplications followed by nonlinearities, which is particularly amenable to GPU computation and lets us scale efficiently to large data.

Our model outperforms LDA and RS baselines on perplexity and downstream tasks including metadata prediction and document retrieval when evaluated on a new dataset mined from Reddit. We also qualitatively analyze the learned topics and discuss the social phenomena uncovered.

2 Model

We now present an overview of our model. Specifically, it will take the probabilistic form of an undirected graphical model whose architecture mirrors the tree structure of the threads in our data.

2.1 Motivating Dataset

We evaluate on a corpus mined from Reddit, an internet forum which ranks as the fourth most trafficked site in the US (Alexa, 2018) and sees millions of daily comments (Reddit, 2015). Discourse on Reddit follows a branching pattern, shown in Figure 1. The largest unit of discourse is a thread, beginning with a link to external content or a natural language prompt, posted to a relevant subreddit based on its subject matter. Users comment in response to the original post (OP), or to any other comment. The result is a structure which splits at many points into more specific or tangential discussions that while locally coherent may differ substantially from each other. The data reflect features of the underlying memory and network structure of the generating process; comments are serially correlated and highly cross-referential. We treat individual comments as "documents" under the standard topic modeling paradigm, but use observed reply structure to induce a tree of documents for every thread.

2.2 Description of Discursive Distributed Topic Model

We now introduce the Discursive Distributed Topic Model (DDTM) (illustrated in Figure 1). For each comment in the thread, DDTM assigns a latent vector of binary random variables (or *bits*) that collectively form a distributed embedding of the topical content of that comment; for instance, one bit might represent sarcastic language while another might track usage of specific acronyms - a given comment could have any combination of those features. These representations are tied to those of parent and child comments via coupling potentials (see Section 2.3), which allow them to learn discursive properties by inducing a deep undirected network over the thread. In order to encourage the model to use these *comment-level* representations to learn discursive and stylistic patterns as opposed to simply topics of discussion, we incorporate a single additional latent vector for the entire thread that interacts with each comment, explaining word choices that are mainly topical rather than discursive or stylistic. As we demonstrate in our experiments (see Section 6) the *thread-level* embedding learns distributions more reminiscent of what a traditional topic model would uncover, while the comment-level embeddings model styles of speaking and mannerisms that do not directly indicate specific subjects of conversation. The joint probability is defined in terms of an energy function that scores latent embeddings and observed word counts across the tree of comments within a thread using log-bilinear potentials, and is globally normalized over all word count and embedding combinations.

2.3 Probability Model

More formally, consider a thread containing N comments each of size D_n with a vocabulary of size K. As depicted in Figure 1, each comment is viewed as a bag-of-words, densely connected via a log-bilinear potential to a latent embedding of size F. Let each comment be represented as as an integer vector $x_n \in \mathbb{Z}^K$ where x_{nk} is number of times word k was observed in comment n, and let $h_n = \{0,1\}^F$ be the topic embedding for each comment, and let $h_0 = \{0,1\}^F$ be the embedding for the entire thread. To model topic transitions, we score the embeddings of parent-child pairs with a separate coupling potential as shown in Figure 1 (comments with no parents or children receive additional start/stop biases respectively). Let replies be represented with sets R, P_N, and C_N where $(n,m) \in R$ and $n \in P_m$ and $m \in C_n$ if comment m is a reply to comment n. DDTM assigns probability to a specific configuration of x, h with an energy function scored by the emission (π_e) and coupling (π_c) potentials.

$$\begin{aligned}
E(x,h;\theta) &= \underbrace{\sum_{n=1}^{N} \pi_\text{e}(h,x,n)}_{\text{Emission Potentials}} + \underbrace{\sum_{(n,m)\in R} \pi_\text{c}(h,n,m)}_{\text{Coupling Potentials}} \\
\pi_\text{e}(h,x,n) &= h_n^\intercal U x_n + x_n^\intercal a + D_n h_n^\intercal b \\
&\quad + h_0^\intercal V x_n + D_n h_0^\intercal c \\
\pi_\text{c}(h,n,m) &= h_n^\intercal W h_m
\end{aligned} \tag{1}$$

Note that the bias on embeddings is scaled by the number of words in the comment, which controls for their highly variable length. The joint probability is computed by exponentiating the energy and dividing by a normalizing constant.

$$\begin{aligned}
p(x,h;\theta) &= \frac{\exp(E(x,h;\theta))}{Z(\theta)} \\
Z(\theta) &= \sum_{x',h'} \exp(E(x',h';\theta))
\end{aligned} \tag{2}$$

This architecture encourages the model to learn discursive maneuvers via the coupling potentials while separating within-thread variance and across-thread variance through the comment-level and thread-level embeddings respectively. The coupling of latent variables makes factored inference impossible, meaning that even the exact computation of the partition function is no longer tractable. This necessitates approximating the gradients for learning which we will now address.

3 Learning and Inference

Inference in this model class in intractable, so as has been done in previous work on topic modeling (Ng and Jordan, 2003) we rely on variational methods to approximate the gradients needed during training as well as the posteriors over the topic bit vectors. Specifically, we will need the gradients of the normalizer and the sum of the energy function over the hidden variables

$$E(x;\theta) = \log \sum_h \exp(E(x,h;\theta)) \tag{3}$$

which we refer to as the *marginal energy*. Following the approach described for undirected models by Eisner (2011), we approximate these quantities and their gradients with respect to the model parameters θ as we will now describe (thread-level embeddings are omitted in this section for clarity).

3.1 Normalizer Approximation

We aim to train our model to maximize the marginal likelihood of the observed comment word counts, conditioned on the reply links. To do this we must compute the gradient of the normalizer $Z(\theta)$. However, this quantity is computationally intractable, as it contains a summation over all exponential choices for every word in the thread. Therefore, we must approximate $Z(\theta)$. Observe that under Jensen's Inequality, we can form the following lower bound on the normalizer using an approximate joint distribution $q^{(\text{Z})}$.

$$\begin{aligned}
&\log Z(\theta) = \log \sum_{x,h} \exp(E(x,h;\theta)) \\
&\geq \mathbb{E}_{q^{(\text{Z})}}[E(x,h;\theta)] - \mathbb{E}_{q^{(\text{Z})}}[\log q^{(\text{Z})}(x,h;\phi,\gamma)]
\end{aligned} \tag{4}$$

We now define $q^{(\text{Z})}$ as depicted in Figure 2 as a mean-field approximation that treats all variables as independent. We parameterize $q^{(\text{Z})}$ with

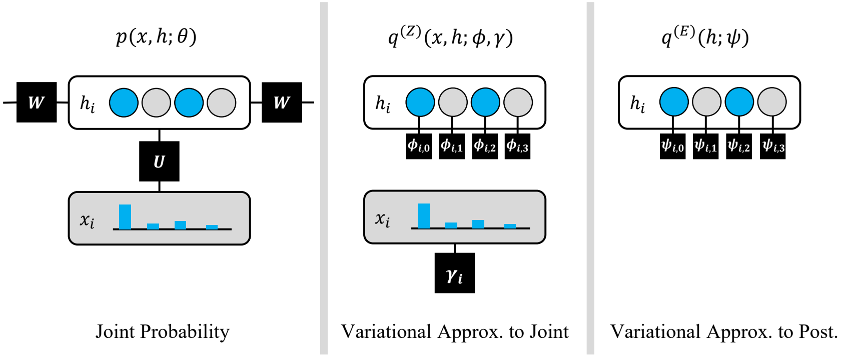

Figure 2: Factor graph of full joint compared to mean-field approximations to joint and posterior.

$\phi_{nf} \in [0,1]$, independent Bernoulli parameters representing the probability of h_{nf} being equal to 1, and γ_{nk} replicated softmaxes representing the probability of a word in x_n taking the value k. Note that all words in x_n are modeled as samples from this single distribution. The approximation then factors as follows:

$$
\begin{aligned}
q^{(\mathrm{Z})}(x,h;\phi,\gamma) &= q^{(\mathrm{Z})}(x;\gamma)\cdot q^{(\mathrm{Z})}(h;\phi)\\
q^{(\mathrm{Z})}(x;\gamma) &= \prod_{n=1}^{N}\prod_{k=1}^{K}(\gamma_{nk})^{x_{nk}}\\
q^{(\mathrm{Z})}(h;\phi) &= \prod_{n=1}^{N}\prod_{f=1}^{F}\left((\phi_{nf})^{h_{nf}}(1-\phi_{nf})^{(1-h_{nf})}\right)
\end{aligned}
\tag{5}
$$

We optimize the parameters of $q^{(\mathrm{Z})}$ to maximize its variational lower bound, via iterative mean-field updates, which allow us to perform coordinate ascent over the parameters of $q^{(\mathrm{Z})}$. Maximizing the lower bound with respect to particular ϕ_{nf} and γ_{nk} while holding all other parameters frozen, yields the following mean-field update equations (biases omitted for clarity):

$$
\begin{aligned}
\phi_{n\cdot} &= \sigma\left(U\gamma_n + \sum_{m\in C_n} W\phi_m + \phi_{P_n}^{\intercal}W\right)\\
\gamma_{n\cdot} &= \sigma\left(\phi_n^{\intercal}U\right)
\end{aligned}
\tag{6}
$$

We iterate over the parameters of $q^{(\mathrm{Z})}$ in an "upward-downward" manner; first updating ϕ for all comments with no children, then all comments whose children have been updated, and so on up to the root of the thread. Then we perform the same updates in reverse order. After updating all ϕ, we then update γ simultaneously (the components of γ are independent conditioned on ϕ). We iterate these upward-downward passes until convergence.

3.2 Marginal Energy Approximation

We can now approximate the normalizer, but still need the marginal data likelihood in order to take gradient steps on it and train our model. In order to recover the marginal likelihood, we must next approximate the marginal energy $E(x;\theta)$ as it too is intractable. This is due to the coupling potentials, which make the topics across comments dependent even when conditioned on the word counts. To do this, we form an additional variational approximation (see Figure 2) to the marginal energy, which we optimize similarly.

$$
\begin{aligned}
E(x;\theta) &= \log\sum_{h}\exp(E(x,h;\theta))\\
&\geq \mathbb{E}_{q^{(\mathrm{E})}}[E(x,h;\theta)] - \mathbb{E}_{q^{(\mathrm{E})}}[\log q^{(\mathrm{E})}(h;\psi)]
\end{aligned}
\tag{7}
$$

Since $q^{(\mathrm{E})}(h;\psi)$ need only model the hidden units h, we can parameterize it in the same manner as $q^{(\mathrm{Z})}(h;\phi)$. Note that while these distributions factor similarly, they do not share parameters, although we find that in practice, initializing $\phi \leftarrow \psi$ improves our approximation. We optimize the lower bound on $E(x;\theta)$ via a similar coordinate ascent strategy, where the mean-field updates take the following form (biases omitted for clarity):

$$
\psi_{n\cdot} = \sigma\left(Uh_n + \sum_{m\in C_n} W\psi_m + \psi_{P_n}^{\intercal}W\right) \tag{8}
$$

We can use $q^{(\mathrm{E})}$ to perform inference at test time in our model, as its parameters ψ directly correspond to the expected values of the hidden topic embeddings under our approximation.

3.3 Learning via Gradient Ascent

We train the parameters of our true model $p(x,h;\theta)$ via stochastic updates wherein we optimize both approximations on a single datum (i.e. thread) to compute the approximate gradient of its log-likelihood, and take a single gradient step on the model parameters (repeating on all training instances until convergence). That gradient is given by the difference in feature expectations under the approximations (entropy terms from the lower bounds are dropped as they do not depend on θ).

$$\begin{aligned}\nabla \log p(x;\theta) \approx\ & \mathbb{E}_{q^{(\mathrm{E})}(h;\psi)}\left[\nabla E(x,h;\theta)\right] \\ & - \mathbb{E}_{q^{(\mathrm{Z})}(x',h;\psi)}\left[\nabla E(x',h;\theta)\right]\end{aligned} \tag{9}$$

In summary, we use two separate mean-field approximations to compute lower bounds on the marginal energy $E(x,h;\theta)$, and its normalizer $Z(\theta)$, which lets us approximate the marginal likelihood $p(x;\theta)$. Note that as our estimate on the marginal likelihood is the difference between two lower bounds, it is not a lower bound itself, although in practice it works well for training.

3.4 Scalability and GPU Implementation

Given the magnitude of our dataset, it is essential to be able to train efficiently at scale. Many commonly used topic models such as LDA (Ng and Jordan, 2003) have difficulty scaling, particularly if trained via MCMC methods. Improvements have been shown from online training (Hoffman et al., 2010), but extending such techniques to model comment-to-comment connections and leverage GPU compute is nontrivial.

In contrast, our proposed model and mean-field procedure can be scaled efficiently to large data because they are amenable to GPU implementation. Specifically, the described inference procedure can be viewed as the output of a neural network. This is because DDTM is globally normalized with edges parameterized as log-bilinear weights, which results in the mean-field updates taking the form of matrix operations followed by nonlinearities. Therefore, a single iteration of mean-field is equivalent to a forward pass through a recursive neural network, whose architecture is defined by the tree structure of the thread. Multiple iterations are equivalent to feeding the output of the network back into itself in a recurrent manner, and optimizing for T iterations is achieved by unrolling the network over T timesteps. This property makes DDTM highly amenable to efficient training on a GPU, and allowed us to scale experiments to a dataset of over 13M total Reddit comments.

4 Experimental Setup

4.1 Data

We mined a corpus of Reddit threads pulled through the platform's API. Focusing on the twenty most popular subreddits (gifs, todayilearned, CFB, funny, aww, AskReddit, BlackPeopleTwitter, videos, pics, politics, The_Donald, soccer, leagueoflegends, nba, nfl, worldnews, movies, mildlyinteresting, news, gaming) over a one month period yielded $200,000$ threads consisting of $13,276,455$ comments total. The data was preprocessed by removing special characters, replacing URLs with a domain-specific token, stemming English words using a Snowball English Stemmer (Porter, 2001), removing stopwords, and truncating the vocabulary to only include the top $10,000$ most common words. OPs are modeled as a comment at the root of each thread to which all top-level comments respond. This dataset will be made available for public use after publication.

4.2 Baselines and Comparisons

We compare to baselines of Replicated Softmax (RS) (Salakhutdinov and Hinton, 2009) and Latent Dirichlet Allocation (LDA) (Ng and Jordan, 2003). RS is a distributed topic model similar to our own, albeit without any coupling potentials. LDA is a locally normalized topic model which defines topics as non-overlapping distributions over words. To ensure that DDTM does not gain an unfair advantage purely by having a larger embedding space, we divide the dimensions equally between comment- and thread-level. Unless specified 64 bits/topics were used. We experiment with RS and LDA treating either comments or full threads as documents.

4.3 Training and Initialization

SGD was performed using the Adam optimizer (Kingma and Ba, 2015). When running inference, we found convergence was reached in an average of 2 iterations of updates. Using a single NVIDIA Titan X (Pascal) card, we were able to train our model to convergence on the training set of 10M comments in less than 30 hours. It is worth noting that we found DDTM to be fairly sensitive to initialization. We found best results from Gaussian noise, with comment-level emissions at variance of 0.01, thread-level emissions at 0.0001, and transitions at 0. We initialized all biases to 0 except for the bias on word counts, which we set to the unigram log-probabilities from the train set.

	Perplexity (nats)			
Bits	32	64	96	128
RS (thr)	2240	2234	2233	2257
RS (cmt)	1675	1894	2245	2518
DDTM (-cpl)	2027	1704	1766	1953
DDTM	**1624**	**1590**	**1719**	**713**

Table 1: Perplexity of DDTM with and without coupling potentials (-cpl) vs. baselines trained at comment (cmt) or thread (thr) level across various numbers of topics and bits. For reference, a unigram model achieves 2644.

5 Results

5.1 Evaluating Perplexity

We compare models by perplexity on a held-out test set, a standard evaluation for generative and latent variables models.

Setup: Due to the use of mean-field approximations for both the marginal energy and normalizer we lose any guarantees regarding the accuracy of our likelihood estimate (both approximations are lower bounds, and therefore their difference is neither a strict lower bound nor guaranteed to be unbiased). To evaluate perplexity in a more principled way, we use Annealed Importance Sampling (AIS) to estimate the ratio between our model's normalizer and the tractable normalizer of a base model from which we can draw true independent samples as described by Salakhutdinov and Murray (2008). Note that since the marginal energy is intractable in our model, unlike a standard RBM, we must sample the joint - and not the marginal - intermediate distributions. This yields an unbiased estimate of the normalizer. The marginal energy must still be approximated via a lower bound, but given that AIS is unbiased and empirically low in variance, we can treat the overall estimate as a lower bound on likelihood for evaluation. Using 2000 intermediate distributions, and averaging over 20 runs, we evaluated per-word perplexity over a set of 50 unseen threads. Results are shown in Table 1.

Results: DDTM achieves the lowest perplexity at all dimensionalities. Note our ablation with the coupling potentials removed (-cpl), increases perplexity noticeably, indicating that modeling replies helps beyond simply modeling threads and comments jointly, particularly at larger embeddings. For reference, a unigram model achieves 2644. We find that LDA's approximate perplexity is even worse, likely due to slackness in its lower bound.

Task	**Upvote Regr.** (MSE)	**Deletion Pred.** (% acc.)
LDA (thr)	1.952	68.35
LDA (cmt)	2.047	59.26
RS (thr)	2.024	69.92
RS (cmt)	2.007	66.45
DDTM	**1.933**	**70.39**

Table 2: Performance of DDTM vs. Replicated Softmax (RS) and Latent Dirichlet Allocation (LDA) at predicting upvotes and child deletion.

5.2 Upvote Regression

To measure how well embeddings capture comment-level characteristics, we feed them into a linear regression model that predicts the number of upvotes the comment received. Upvotes provide a loose human-annotated measure of likability. We expect that context matters in determining how well received a comment is; the same comment posted in response to different parents may receive a very different number of upvotes. Hence, we expect comment-level embeddings to be more informative for this task when connected via our model's coupling potentials.

Setup: We trained a standard linear regressor for each model. The regressor was trained using ordinary least squares on the entire training set of comments using the model's computed topic embeddings as input, and the number of upvotes on the comment as the output to predict. As a preprocessing step, we took the log of the absolute number of votes before training. We compared models by mean squared error (MSE) on our test set. Results are shown in Table 2.

Results: DDTM achieves lowest MSE. To assess statistical significance, we performed a 500 sample bootstrap of our training set. The standard errors of these replications are small, and a two-sample t-test rejects the null hypothesis that DDTM has an average MSE equal to that of the next best method ($p < .001$). Note that our model outperforms both comment- and thread-level embeddings, suggesting that modeling these jointly, and modeling the effect of neighboring representations in the comment graph, more accurately learns information relevant to a comment's social impact.

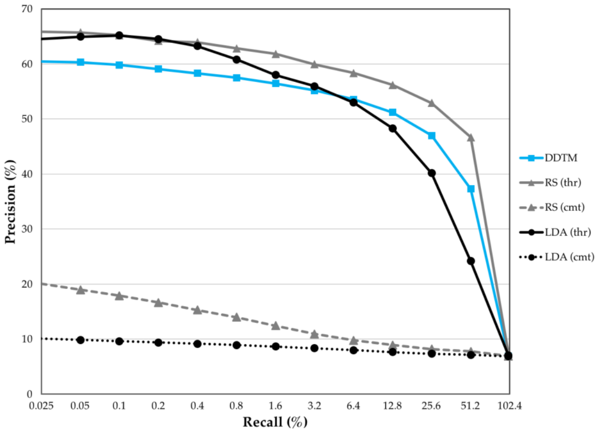

Figure 3: Precision vs. recall for document retrieval based on subreddit comparing various models for 1000 randomly selected held-out query comments.

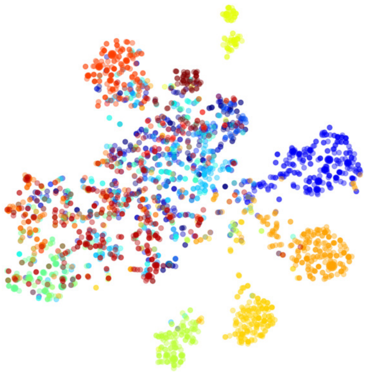

Figure 4: t-SNE visualization of a random sample of DDTM thread-level embeddings colored by subreddit (not observed in training)

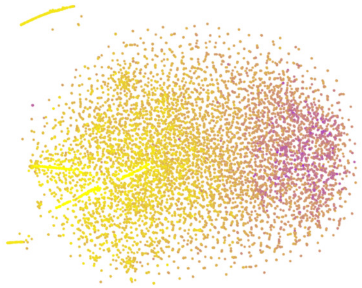

Figure 5: t-SNE visualization of a random sample of DDTM comment-level embeddings colored by log of comment length (darker is longer).

5.3 Deletion Prediction

Comments that are excessively provocative or in violation of site rules are often deleted, either by the author or a moderator. We can measure whether DDTM captures discursive interactions that lead to such intervention by training a logistic classifier that predicts whether any of a given comment's children have been deleted.

Setup: For each model, a logistic regression classifier was trained stochastically with the Adam optimizer on the entire training set of comments using the model's computed topic embeddings as input, and a binary label for whether the comment had any deleted children as the output to predict. We compared models by accuracy on our test set. Results are shown in Table 2.

Results: DDTM gets the highest accuracy. Interestingly, thread-level models do better than comment-level ones, which suggests that certain topics or even subreddits may correlate with comments being deleted. This makes sense given that subreddits vary in severity of moderation. DDTM's performance also demonstrates that modeling comment-to-comment interaction patterns is helpful in predicting when a comment will spawn a deleted future response, which strongly matches our intuition.

5.4 Document Retrieval

Finally, while DDTM is not designed to better capture topical structure, we evaluate the extent to which it can still capture this information by performing document retrieval, a standard evaluation, for which we treat the subreddit to which a thread was posted as a label for relevance. Note that every comment within the same thread belongs to the same subreddit, which gives thread-level models an inherent advantage at this task. We include this task purely for the purpose of demonstrating that by capturing discursive patterns, DDTM does not lose the ability to model thread-level topics as well.

Setup: Given a query comment from our held-out test set, we rank the training set by the Dice similarity of the hidden embeddings computed by the model. We consider a retrieved comment relevant to the query if they both originate from the same subreddit, which loosely categorizes the semantic content. Tuning the number of documents we return allows us to form precision recall curves, which we show in Figure 3.

Results: DDTM outperforms both comment-level baselines and is competitive with thread-level models, even beating LDA at high levels of recall. This indicates that despite using half of its dimensions to model comment-to-comment interactions DDTM can still do almost as good a job of modeling thread-level semantics as a model using its entire capacity

Bit #	**Associated Word Stems by Emission Weight (Higher Score → Lower Score)**
Comment-Level	
Bit 1	faq tldrs pms 165 til keyword questions feedback chat pm
2	irl riamverysmart legend omfg riski aboard favr madman skillset tunnel
3	lotta brah ouch spici oof bummer buildup viewership hd uncanni
4	funniest mah tfw teleport fav hoo plz bah whyd dumbest
5	handsom hipster texan hottest whore norwegian shittier scandinavian jealousi douch
Thread-Level	
Bit 1	btc gameplay tutori cyclist dev currenc kitti bitcoin rpg crypto
2	url_youtu url_leagueoflegends url_businessinsider url_twitter url_redd url_snopes
3	comey pede macron pg13 maga globalist ucf committe cuck distributor
4	maduro venezuelan ballot puerto catalonia rican quak skateboard venezuela quebec
5	nra scotus opioid cheney nevada metallica marijuana vermont colorado xanax

Table 3: Words with the highest emission weight for various comment-level and thread-level bits.

to do so. The gap between comment-level RS and LDA is also consistent with LDA's known issues dealing with sparse data (Sridhar, 2015), and lends credence to our theory that distributed topic representations are better suited to such domains.

6 Qualitative Analysis of Topics

We now offer qualitative analysis of the topic embeddings learned by our model. Note that since we use distributed embeddings, our bits are more akin to filters than complete distributions over words, and we typically observe as many as half of them active for a single comment. In a sense, we have an exponential number of topics, whose parameterization simply factors over the bits. Therefore, it can be difficult to interpret them as one would interpret topics learned by a model such as LDA. Furthermore, we find that in practice this effect is correlated with the topic embedding size; the more bits our model has, the less sparse and consequently less individually meaningful the bits become. Therefore for this analysis, we specifically focus on DDTM trained with 64 bits total.

6.1 Bits in Isolation

Directly inspecting the emission parameters, reveals that the comment-level and thread-level halves of our embeddings capture substantially different aspects of the data (shown in Table 3) akin to vertical, within-thread, and horizontal, across-thread sources of variance respectively. The comment-level topic bits tend to reflect styles of speaking, lingo, and memes that are not unique to a particular subject of discourse or even subreddit. For example, comment-level Bit 2 captures many words typical of taunting Reddit comments; replying with "/r/iamverysmart" (a subreddit dedicated to mocking people who make grandiose claims about their intellect) is a common way of jokingly implying that the author of the parent comment takes themselves too seriously — and thus corresponds to a certain kind of rhetorical move. Further, it is grouped with other words that indicate related rhetorical moves; calling a user "risky" or a "madman" is a common means of suggesting that they are engaging in a pointless act of rebellion. They also cluster at the coarsest level by length (see Figure 5) which we find to correlate with writing style.

By contrast, the thread-level bits are more indicative of specific topics of discussion, and unsurprisingly they cluster by subreddit (see Figure 4). For example, thread-level Bit 3 captures lexicon used almost exclusively by alt-right Donald Trump supporters as well as the names of various political figures. Bit 4 highlights words related to civil unrest in Spanish speaking parts of the world.

6.2 Bits in Combination

While these distributions over words (particularly for comment-level bits) can seem vague, when multiple bits are active, their effects compound to produce much more specific topics. One can think of the bits as defining soft filters over the space of words, that when stacked together carve out patterns not apparent in any of them individually. We now analyze a few sample topic embeddings. To do this, we perform inference as described on a held-out thread, and pass the comment-level topic embedding for a single sampled comment through our emission matrix and inspect the words with the highest corresponding weight (shown in Table 4). In generative terminology, these can be thought of as reconstructions of comments.

These topic embeddings capture more specific conversational and rhetorical moves. For example,

Sample #	**Associated Word Stems by Emission Weight (Higher Score → Lower Score)**
Comment-Level	
Sample 1	`grade grader math age 5th 9th 10th till mayb 7th`
2	`repost damn dope bamboozl shitload imagin cutest sad legendari awhil`
3	`heh dawg hmm spooki buddi aye m8 aww fam woah`
4	`hug merci bless tfw prayer pleas dear bear banana satan`
5	`chuckl cutest funniest yall bummer oooh mustv coolest ok oop`
6	`cutest heard coolest funniest havent seen ive craziest stupidest weirdest`
7	`reev keanu christoph murphi walken vincent chris til wick roger`
8	`moron douchebag stupid dipshit snitch jackass dickhead idioci hypocrit riddanc`
9	`technic actual realiz happen escal werent citat practic memo cba`
10	`reddit shill question background user subreddit answer relev discord guild`

Table 4: Words with the highest emission weight for sample held-out comment reconstructions.

Sample 6 displays supportive and interested reactionary language, which one might expect to see used in response to a post or comment linking to media or describing something intriguing. This is of note given that one of the primary aims of including coupling potentials was to encourage DDTM to learn "topics" that correspond to responses and interactive behavior, something existing methods are largely not designed for. By contrast, Sample 9 captures a variety of hostile language and insults, which unlike those discussed previously do not denote membership in a particular online community. As patterns of toxic and hateful behavior on Reddit are more well-studied (Chandrasekharan et al., 2017), it could be useful to have a tool to analyze precipitous contexts and parent comments, something which we hope systems based on coupling of comment embeddings have the capacity to provide. Sample 10 is of particular interest as it consists largely of Reddit terminology. Conversations about the meta of the site can manifest for example in users accusing each other of being "shills" (i.e. accounts paid to astroturf on behalf of external interests) or requesting/responding to "guilding", a feature which lets users purchase premium access for each other often in response to a particularly well made comment.

7 Related Work

Many topic models such as LDA (Ng and Jordan, 2003) treat documents as independent mixtures, yet this approach fails to model how comments interact with one another throughout a larger discourse if such connections exist in the data. Other work has considered modeling hierarchy in topics (Griffiths et al., 2004). These models form hierarchical representations of topics themselves, but still treat documents as independent. While this approach can succeed in learning topics of various granularities, it does not explicitly track how topics interact in the context of a nested conversation.

Some approaches such as Pairwise-Link-LDA and Link-PSLA-LDA (Nallapati et al., 2008) attempt to model interactions among documents in an arbitrary graph, albeit with important drawbacks. The former models every possible pairwise link between comments, and the latter models links as a bipartite graph, limiting its ability to scale to large tree-structured threads. Similar work on Topic-Link LDA (Liu et al., 2009) models link probabilities conditioned on both topic similarity and an authorship model, yet this approach is poorly suited to high volume, semi-anonymous online domains. Other studies have leveraged reply-structures on Reddit in the context of predicting persuasion (Hidey and McKeown), but DDTM differs in its generative, unsupervised approach.

DDTM's emission potentials are similar to those of Replicated Softmax (Salakhutdinov and Hinton, 2009), an undirected model based on a Restricted Boltzmann Machine. Unlike LDA-style models, RS does not assign a topic to each word, but instead builds a distributed representation. In this setting, a single word can be likely under two different topics, both of which are present, and lend probability mass to that word. LDA-style models by contrast would require the topics to compete for the word.

8 Conclusion

In this paper we introduce a novel way to learn topic interactions in observed discourse trees, and describe GPU-amenable learning techniques to train on large-scale data mined from Reddit. We demonstrate improvements over previous models on perplexity and downstream tasks, and offer qualitative analysis of learned discursive patterns. The dichotomy between the two levels of embeddings hints at applications in style-transfer.

References

Alexa. 2018. Reddit.com traffic, demographics and competitors. https://www.alexa.com/siteinfo/reddit.com. Accessed: 2018-02-22.

Alexander TJ Barron, Jenny Huang, Rebecca L Spang, and Simon DeDeo. 2017. Individuals, institutions, and innovation in the debates of the french revolution. *arXiv preprint arXiv:1710.06867*.

Eshwar Chandrasekharan, Umashanthi Pavalanathan, Anirudh Srinivasan, Adam Glynn, Jacob Eisenstein, and Eric Gilbert. 2017. You can't stay here: The efficacy of reddit's 2015 ban examined through hate speech. *Proceedings of the ACM on Human-Computer Interaction*, 1(CSCW):31.

Lan Du, Wray Buntine, Huidong Jin, and Changyou Chen. 2012. Sequential latent dirichlet allocation. *Knowledge and information systems*, 31(3):475–503.

Jason Eisner. 2011. High-level explanation of variational inference. https://www.cs.jhu.edu/~jason/tutorials/variational.html. Accessed: 2018-02-22.

Thomas L Griffiths, Michael I Jordan, Joshua B Tenenbaum, and David M Blei. 2004. Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems*, pages 17–24.

Christopher Hidey and Kathleen McKeown. Persuasive influence detection: The role of argument sequencing. In *Association for the Advancement of Artificial Intelligence*.

M. Hoffman, D. Blei, and F. Bach. 2010. Online learning for latent dirichlet allocation. *Neural Information Processing Systems*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*.

Yan Liu, Alexandru Niculescu-Mizil, and Wojciech Gryc. 2009. Topic-link lda: joint models of topic and author community. In *proceedings of the 26th annual international conference on machine learning*, pages 665–672. ACM.

Ramesh M Nallapati, Amr Ahmed, Eric P Xing, and William W Cohen. 2008. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–550. ACM.

D. Blei A. Ng and M. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Martin Porter. 2001. Snowball: A language for stem-ming algorithms. http://snowball.tartarus.org/texts. Accessed: 2018-02-22.

Reddit. 2015. Reddit in 2015. https://redditblog.com/2015/12/31/reddit-in-2015/. Accessed: 2018-02-22.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Replicated softmax: An undirected topic model. *Advances in Neural Information Processing Systems*, 22:1607–1614.

Ruslan Salakhutdinov and Iain Murray. 2008. On the quantitative analysis of deep belief networks. *Proceedings of the 25th International Conference on Machine Learning*.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised topic modeling for short texts using distributed representations of words. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 192–200.

Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing twitter and traditional media using topic models. In *European Conference on Information Retrieval*, pages 338–349. Springer.

Learning Disentangled Representations of Texts with Application to Biomedical Abstracts

Sarthak Jain
Northeastern University
`jain.sar@husky.neu.edu`

Edward Banner
Northeastern University
`ebanner@ccs.neu.edu`

Jan-Willem van de Meent
Northeastern University
`j.vandemeent@northeastern.edu`

Iain J. Marshall
King's College London
`iain.marshall@kcl.ac.uk`

Byron C. Wallace
Northeastern University
`b.wallace@northeastern.edu`

Abstract

We propose a method for learning *disentangled* representations of texts that code for distinct and complementary aspects, with the aim of affording efficient model transfer and interpretability. To induce disentangled embeddings, we propose an adversarial objective based on the (dis)similarity between triplets of documents with respect to specific aspects. Our motivating application is embedding biomedical abstracts describing clinical trials in a manner that disentangles the *populations*, *interventions*, and *outcomes* in a given trial. We show that our method learns representations that encode these clinically salient aspects, and that these can be effectively used to perform aspect-specific retrieval. We demonstrate that the approach generalizes beyond our motivating application in experiments on two multi-aspect review corpora.

1 Introduction

A classic problem that arises in (distributed) representation learning is that it is difficult to determine what information individual dimensions in an embedding encode. When training a classifier to distinguish between images of people and landscapes, we do not know *a priori* whether the model is sensitive to differences in color, contrast, shapes or textures. Analogously, in the case of natural language, when we calculate similarities between document embeddings of user reviews, we cannot know if this similarity primarily reflects user sentiment, the product discussed, or syntactic patterns. This lack of interpretability makes it difficult to assess whether a learned representations is likely to generalize to a new task or domain, hindering model transferability. Disentangled representations with known semantics could allow more efficient training in settings in which supervision is expensive to obtain (e.g., biomedical NLP).

Thus far in NLP, learned distributed representations have, with few exceptions (Ruder et al., 2016; He et al., 2017; Zhang et al., 2017), been *entangled*: they indiscriminately encode all aspects of texts. Rather than representing text via a monolithic vector, we propose to estimate multiple embeddings that capture complementary aspects of texts, drawing inspiration from the ML in vision community (Whitney, 2016; Veit et al., 2017a).

As a motivating example we consider documents that describe clinical trials. Such publications constitute the evidence drawn upon to support *evidence-based medicine* (EBM), in which one formulates precise clinical questions with respect to the Populations, Interventions, Comparators and Outcomes (PICO elements) of interest (Sackett et al., 1996).[1] Ideally, learned representations of such articles would factorize into embeddings for the respective PICO elements. This would enable aspect-specific similarity measures, in turn facilitating retrieval of evidence concerning a given condition of interest (i.e., in a specific patient population), regardless of the interventions and outcomes considered. Better representations may reduce the amount of supervision needed, which is expensive in this domain.

Our work is one of the first efforts to induce disentangled representations of texts,[2] which we believe may be broadly useful in NLP. Concretely, our contributions in this paper are as follows:

- We formalize the problem of learning disentangled representations of texts, and develop a relatively general approach for learning these from aspect-specific similarity judgments expressed as triplets $(s, d, o)_a$, which indicate that document d is more similar to document s than to document o, with respect to aspect a.

---

[1] We collapse I and C because the distinction is arbitrary.

[2] We review the few recent related works that do exist in Section 5.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4683–4693
Brussels, Belgium, October 31 - November 4, 2018.

- We perform extensive experiments that provide evidence that our approach yields disentangled representations of texts, both for our motivating task of learning PICO-specific embeddings of biomedical abstracts, and, more generally, for multi-aspect sentiment corpora.

2 Framework and Models

Recent approaches in computer vision have emphasized unsupervised learning of disentangled representations by incorporating information-theoretic regularizers into the objective (Chen et al., 2016; Higgins et al., 2017). These approaches do not require explicit manual annotations, but consequently they require post-hoc manual assignment of meaningful interpretations to learned representations. We believe it is more natural to use weak supervision to induce meaningful aspect embeddings.

2.1 Learning from Aspect Triplets

As a general strategy for learning disentangled representations, we propose exploiting aspect-specific document triplets $(s, d, o)_a$: this signals that s and d are *more* similar than are d and o, with respect to aspect a (Karaletsos et al., 2015; Veit et al., 2017b), i.e., $\text{sim}_a(d, s) > \text{sim}_a(d, o)$, where sim_a quantifies similarity w.r.t. aspect a.

We associate with each aspect an encoder enc_a (encoders share low-level layer parameters; see Section 2.2 for architecture details). This is used to obtain text embeddings $(\mathbf{e}_s^a, \mathbf{e}_d^a, \mathbf{e}_o^a)$. To estimate the parameters of these encoders we adopt a simple objective that seeks to maximize the similarity between $(\mathbf{e}_d^a, \mathbf{e}_s^a)$ and minimize similarity between $(\mathbf{e}_d^a, \mathbf{e}_o^a)$, via the following maximum margin loss

$$\mathcal{L}(\mathbf{e}_s^a, \mathbf{e}_d^a, \mathbf{e}_o^a) = max\{0, 1 - sim(\mathbf{e}_d^a, \mathbf{e}_s^a) + sim(\mathbf{e}_d^a, \mathbf{e}_o^a)\} \quad (1)$$

Where similarity between documents i and j with respect to a particular aspect a, $sim_a(i, j)$, is simply the cosine similarity between the aspect embeddings $\mathbf{e}_i^a$ and $\mathbf{e}_j^a$. This allows for the same documents to be similar with respect to some aspects while dissimilar in terms of others.

The above setup depends on the correlation between aspects in the training data. At one extreme, when triplets enforce identical similarities for all aspects, the model cannot distinguish between aspects at all. At the other extreme, triplets are present for only one aspect a, and absent for all other aspects a': In this case the model will use only the embeddings for aspect a to represent similarities. In general, we expect a compromise between these extremes, and propose using negative sampling to enable the model to learn targeted aspect-specific encodings.

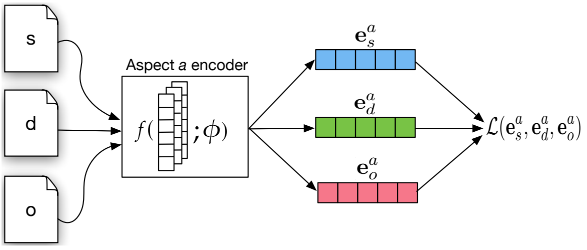

Figure 1: We propose associating aspects with encoders (low-level parameters are shared across aspects; this is not shown) and training these with triplets codifying aspect-wise relative similarities.

2.2 Encoder Architecture

Designing an aspect-based model requires specifying an encoder architecture. One consideration here is interpretability: a desirable property for aspect encoders is the ability to identify salient words for a given aspect. With this in mind, we propose using gated CNNs, which afford introspection via the token-wise gate activations.

Figure 2 schematizes our encoder architecture. The input is a sequence of word indices $d = (w_1, ..., w_N)$ which are mapped to m-dimensional word embeddings and stacked into a matrix $E = [\mathbf{e}_1, ..., \mathbf{e}_N]$. These are passed through sequential convolutional layers $C_1, ..., C_L$, which induce representations $H_l \in \mathbb{R}^{N \times k}$:

$$H_l = f_e(X * K_l + \mathbf{b}_l) \quad (2)$$

where $X \in \mathbb{R}^{N \times k}$ is the input to layer C_l (either a set of n-gram embeddings or H_{l-1}) and k is the number of feature maps. Kernel $K_l \in \mathbb{R}^{F \times k \times k}$ and $\mathbf{b}_l \in \mathbb{R}^k$ are parameters to be estimated, where F is the size of kernel window.[3] An activation function f_e is applied element-wise to the output of the convolution operations. We fix the size of $H_{l-1} \in \mathbb{R}^{N \times k}$ by zero-padding where necessary. Keeping the size of feature maps constant across layers allows us to introduce residual connections; the output of layer l is summed with the outputs of preceding layers before being passed forward.

We multiply the output of the last convolutional layer $H_L \in \mathbb{R}^{N \times k}$ with gates $\mathbf{g} \in \mathbb{R}^{N \times 1}$ to yield

---

[3] The input to C_1 is $E \in \mathbb{R}^{N \times m}$, thus $K_1 \in \mathbb{R}^{F \times m \times k}$.

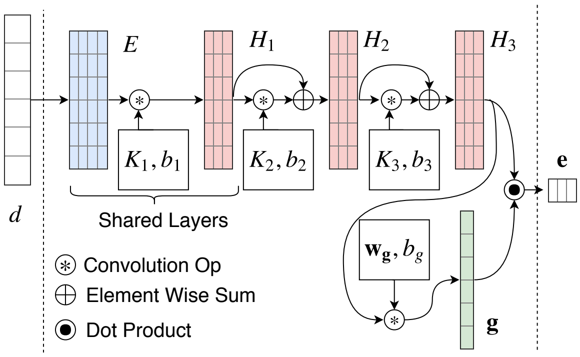

Figure 2: Schematic of our encoder architecture.

our final embedding $\mathbf{e}_d \in \mathbb{R}^{1\times k}$:

$$\begin{aligned} \mathbf{g} &= \sigma(H_L \cdot \mathbf{w}_g + b_g) \\ \mathbf{e}_d &= \mathbf{g}^T H_L \end{aligned} \tag{3}$$

where $\mathbf{w}_g \in \mathbb{R}^{k\times 1}$ and $b_g \in \mathbb{R}$ are learned parameters and σ is the sigmoid activation function. We impose a sparsity-inducing constraint on $\mathbf{g}$ via the ℓ_1 norm; this allows the gates to effectively serve as an attention mechanism over the input. Additionally, to capture potential cross-aspect correlation, weights in the embedding and first convolutional layers are shared between aspect encoders.

Alternative encoders. To assess the relative importance of the specific encoder model architecture used, we conduct experiments in which we fine-tune standard document representation models via triplet-based training. Specifically, we consider a single-layer MLP with BoW inputs, and a Neural Variational Document Model (NVDM) (Miao et al., 2016). For the NVDM we take a weighted sum of the original loss function and the triplet-loss over the learned embeddings, where the weight is a model hyperparameter.

3 Varieties of Supervision

Our approach entails learning from triplets that codify relative similarity judgments with respect to specific aspects. We consider two approaches to acquiring such triplets: the first exploits aspect-specific summaries written for texts, and the second assumes a more general scenario in which we solicit aspect-wise triplet judgments directly.

3.1 Deriving Triplets from Aspect Summaries

In the case of our motivating example – disentangled representations for articles describing clinical trials – we have obtained aspect-specific summaries from the *Cochrane Database of Systematic Reviews (CDSR)*. Cochrane is an international organization that creates and curates biomedical *systematic reviews*. Briefly, such reviews seek to formally synthesize all relevant articles to answer precise clinical questions, i.e., questions that specify a particular PICO frame. The CDSR consists of a set of reviews $\{R_i\}$. Reviews include multiple articles (studies) $\{S_{ij}\}$. Each study S consists of an abstract A and a set of free text summaries (s_P, s_I, s_O) written by reviewers describing the respective P, I and O elements in S.

Reviews implicitly specify PICO frames, and thus two studies in any given review may be viewed as equivalent with respect to their PICO aspects. We use this observation to derive document triplets. Recall that triplets for a given aspect include two comparatively similar texts (s, d) and one relatively dissimilar (o). Suppose the aspect of interest is the trial population. Here we match a given abstract (d) with its matched population summary from the CDSR (s); this encourages the encoder to yield similar embeddings for the abstract and the population description. The dissimilar o is constructed to distinguish the given abstract from (1) other aspect encodings (of interventions, outcomes), and, (2) abstracts for trials with different populations.

Concretely, to construct a triplet (s, d, o) for the PICO data, we draw two reviews R_1 and R_2 from the CDSR at random, and sample two studies from the first (s_1, s_1') and one from the second (s_2). Intuitively, s_2 will (very likely) comprise entirely different PICO elements than (s_1, s_1'), by virtue of belonging to a different review. To formalize the preceding description, our triplet is then: $(s = [s_1'^P], d = [s_1^{\text{abstract}}], o = [s_2^P | s_1'^I | s_1'^O])$, where s_1^{abstract} is the abstract for study s_1, and aspect summaries for studies are denoted by superscripts. We include a concrete example of triplet construction in the Appendix, Section D.

3.2 Learning Directly from Aspect-Wise Similarity Judgments

The preceding setup assumes a somewhat unique case in which we have access to aspect-specific summaries written for texts. As a more general setting, we also consider learning directly from triplet-wise supervision concerning relative similarity with respect to particular aspects (Amid and Ukkonen, 2015; Veit et al., 2017a; Wilber et al., 2014). The assumption is that such judgments can be solicited directly from annotators, and thus the approach may be applied to arbitrary domains, so long as meaningful aspects can be defined implicitly via pairwise similarities regarding them.

We do not currently have corpora with such judgments in NLP, so we constructed two datasets using aspect-specific sentiment ratings. Note that this highlights the flexibility of exploiting aspect-wise triplet supervision as a means of learning disentangled representations: existing annotations can often be repurposed into such triplets.

4 Datasets and Experiments

We present a series of experiments on three corpora to assess the degree to which the learned representations are disentangled, and to evaluate the utility of these embeddings in simple downstream retrieval tasks. We are particularly interested in the ability to identify documents similar w.r.t. a target aspect. All parameter settings for baselines are reported in the Appendix (along with additional experimental results). The code is available at `https://github.com/successar/neural-nlp`.

4.1 PICO (EBM) Domain

We first evaluate embeddings quantitatively with respect to retrieval performance. In particular, we assess whether the induced representations afford improved retrieval of abstracts relevant to a particular systematic review (Cohen et al., 2006; Wallace et al., 2010). We then perform two evaluations that explicitly assess the degree of disentanglement realized by the learned embeddings.

The PICO dataset comprises 41K abstracts of articles describing clinical trials extracted from the CDSR. Each abstract is associated with a review and three summaries, one per aspect (P/I/O). We keep all words that occur in ≥ 5 documents, converting all others to `unk`. We truncate documents to a fixed length (set to the 95th percentile).

4.1.1 Quantitative Evaluation

Baselines. We compare the proposed **P**, **I** and **O** embeddings and their concatenation $[\mathbf{P}|\mathbf{I}|\mathbf{O}]$ to the following. **TF-IDF**: standard TF-IDF representation of abstracts. **RR-TF**: concatenated TF-IDF vectors of sentences predicted to describe the respective PICO elements, i.e., sentence predictions made using the pre-trained model from (Wallace et al., 2016) — this model was trained using distant supervision derived from the CDSR. **doc2vec**: standard (entangled) distributed representations of abstracts (Le and Mikolov, 2014). **LDA**: Latent Dirichlet Allocation. **NVDM**: A generative model of text where the representation is a vector of log-frequencies that encode a topic (Miao et al., 2016). **ABAE**: An autoencoder model that discovers latent aspects in sentences (He et al., 2017). We obtain document embeddings by summing over constituent sentence embeddings. **DSSM**: A CNN based encoder trained with triplet loss over abstracts (Shen et al., 2014).

Hyperparameters and Settings. We use three layers for our CNN-based encoder (with 200 filters in each layer; window size of 5) and the PReLU activation function (He et al., 2015) as f_e. We use 200d word embeddings, initialized via pretraining over a corpus of PubMed abstracts (Pyysalo et al., 2013). We used the Adam optimization function with default parameters (Kingma and Ba, 2014). We imposed ℓ_2 regularization over all parameters, the value of which was selected from the range ($1e$-2, $1e$-6) as $1e$-5. The ℓ_1 regularization parameter for gates was chosen from the range ($1e$-2, $1e$-8) as $1e$-6. All model hyperparameters for our models and baselines were chosen via line search over a 10% validation set.

Metric. For this evaluation, we used a held out set of 15 systematic reviews (comprising 2,223 studies) compiled by Cohen et al. (2006). The idea is that good representations should map abstracts in the same review (which describe studies with the same PICO frame) relatively near to one another. To compute AUCs over reviews, we first calculate all pairwise study similarities (i.e., over all studies in the Cohen corpus). We can then construct an ROC for a given abstract a from a particular review to calculate its AUC: this measures the probability that a study drawn from the same review will be nearer to a than a study from a different review. A summary AUC for a review is taken as the mean of the study AUCs in that review.

Results. Table 1 reports the mean AUCs over individual reviews in the Cohen et al. (2006) corpus, and grand means over these (bottom row). In brief: The proposed PICO embeddings (concatenated) obtain an equivalent or higher AUC than baseline strategies on 12/14 reviews, and strictly higher AUCs in 11/14. It is unsurprising that we outperform unsupervised approaches, but we also best RR-TF, which was trained with the same CDSR corpus (Wallace et al., 2016), and DSSM (Shen et al., 2014), which exploits the same triplet loss

Study	TF-IDF	Doc2Vec	LDA	NVDM	ABAE	RR-TF	DSSM	P	I	O	[P|I|O]
ACEInhib.	0.81	0.74	0.72	0.85	0.81	0.67	0.85	0.83	0.88	0.84	**0.92**
ADHD	0.90	0.82	0.83	**0.93**	0.77	0.85	0.83	0.86	0.75	0.91	0.89
Antihist.	0.81	0.73	0.67	0.79	0.84	0.72	**0.91**	0.88	0.84	0.89	**0.91**
Antipsych.	0.75	0.85	0.88	0.89	0.81	0.63	0.96	0.91	0.93	**0.97**	**0.97**
BetaBlockers	0.67	0.65	0.61	0.76	0.70	0.56	0.68	0.71	0.75	0.77	**0.81**
CCBlockers	0.67	0.60	0.67	0.70	0.69	0.58	0.76	0.73	0.69	0.74	**0.77**
Estrogens	0.87	0.85	0.60	0.94	0.85	0.82	0.96	**1.00**	0.98	0.83	**1.00**
NSAIDS	0.85	0.77	0.73	0.9	0.77	0.74	0.89	0.94	**0.95**	0.8	**0.95**
Opioids	0.81	0.75	0.80	0.83	0.77	0.76	0.86	0.80	0.83	**0.92**	**0.92**
OHG	0.79	0.80	0.70	0.89	0.90	0.72	0.90	0.90	0.95	0.95	**0.96**
PPI	0.81	0.79	0.74	0.85	0.82	0.68	0.94	0.94	0.87	0.87	**0.95**
MuscleRelax.	0.60	0.67	0.74	0.75	0.61	0.57	0.77	0.68	0.62	**0.78**	0.75
Statins	0.79	0.76	0.66	0.87	0.77	0.68	0.87	0.82	**0.94**	0.87	**0.94**
Triptans	0.92	0.82	0.83	0.92	0.75	0.81	**0.97**	0.93	0.79	**0.97**	**0.97**
Mean	0.79	0.76	0.73	0.85	0.78	0.70	0.87	0.85	0.84	0.87	**0.91**

Table 1: AUCs achieved using different representations on the Cohen et al. corpus. Models to the right of the | are supervised; those to the right of || constitute the proposed disentangled embeddings.

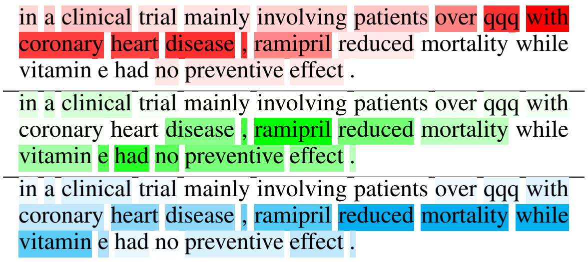

in a clinical trial mainly involving patients over qqq with coronary heart disease , ramipril reduced mortality while vitamin e had no preventive effect .
in a clinical trial mainly involving patients over qqq with coronary heart disease , ramipril reduced mortality while vitamin e had no preventive effect .
in a clinical trial mainly involving patients over qqq with coronary heart disease , ramipril reduced mortality while vitamin e had no preventive effect .

Table 2: Gate activations for each aspect in a PICO abstract. Note that because gates are calculated at the final convolution layer, activations are not in exact 1-1 correspondence with words.

as our model. We outperform the latter by an average performance gain of 4 points AUC (significant at 95% level using independent 2-sample t-test).

We now turn to the more important questions: are the learned representations actually disentangled, and do they encode the target aspects? Table 2 shows aspect-wise gate activations for PICO elements over a single abstract; this qualitatively suggests disentanglement, but we next investigate this in greater detail.

4.1.2 Qualitative Evaluation

To assess the degree to which our PICO embeddings are disentangled – i.e., capture complementary information relevant to the targeted aspects – we performed two qualitative studies.

First, we assembled 87 articles (not seen in training) describing clinical trials from a review on the effectiveness of decision aids (Stacey et al., 2014) for: women with, at risk for, and genetically at risk for, breast cancer (BCt, BCs and BCg, respectively); type II diabetes (D); menopausal women (MW); pregnant women generally (PW) and those who have undergone a C-section previously (PWc); people at risk for colon cancer (CC); men with and at risk of prostate cancer (PCt and PCs, respectively) and individuals with atrial fibrillation (AF). This review is unusual in that it *studies a single intervention (decision aids) across different populations*. Thus, if the model is successful in learning disentangled representations, the corresponding P vectors should roughly cluster, while the I/C should not.

Figure 3 shows a TSNE-reduced plot of the P, I/C and O embeddings induced by our model for these studies. Abstracts are color-coded to indicate the populations enumerated above. As hypothesized, P embeddings realize the clearest separation with respect to the populations, while the I and O embeddings of studies do not co-localize to the same degree. This is reflected quantitatively in the AUC values achieved using each aspect embedding (listed on the Figure). This result implies disentanglement along the desired axes.

Next we assembled 50 abstracts describing trials involving *hip replacement arthroplasty* (HipRepl). We selected this topic because HipRepl will either describe the trial population (i.e., patients who have received hip replacements) or it will be the intervention, but not both. Thus, we would expect that abstracts describing trials in which HipRepl describes the population cluster in the corresponding embedding space, but not in the intervention space (and vice-versa). To test this, we first manually annotated the 50 abstracts, associating HipRepl with either P or I. We used these labels to calculate pairwise AUCs, reported in Table 3. The results imply that the population embeddings discriminate between studies that

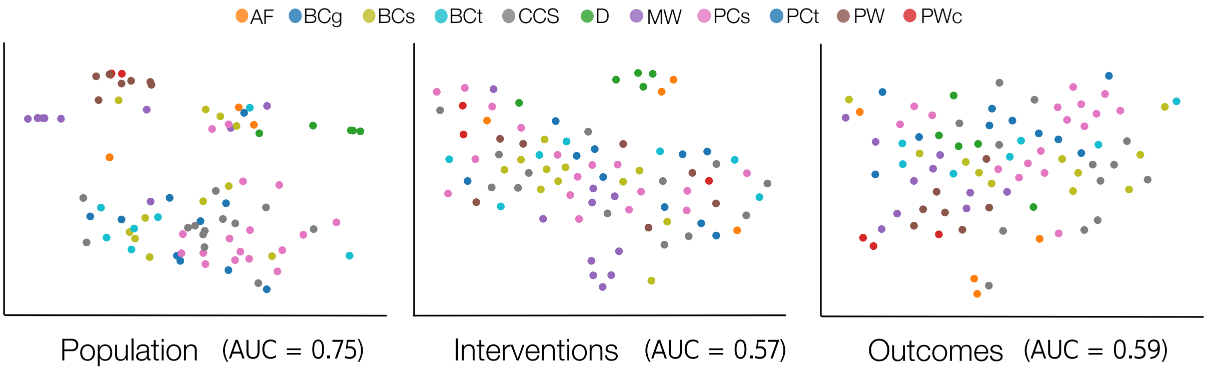

Figure 3: TSNE-reduced scatter of disentangled PICO embeddings of abstracts involving "decision aid" interventions. Abstracts are colored by known population group (see legend). Population embeddings for studies in the same group co-localize, more so than in the intervention and outcome space.

	HipRepl I	HipRepl P	Mean
Population	0.62	**0.68**	**0.66**
Intervention	**0.91**	0.46	0.57
Outcome	0.89	0.42	0.54

Table 3: AUCs realized over HipRepl studies using different embeddings. Column: Study label (HipRepl as P or I). Row: Aspect embedding used.

Population *american*, *area*, *breast*, *colorectal*, *diagnosis*, *inpatients*, *outpatients*, *stage*, *their*, *uk*

Intervention *adjunct*, *alone*, *an*, *discussion*, *intervention*, *methods*, *reduced*, *started*, *took*, *written*

Outcome *adults*, *area*, *either*, *eligible*, *importance*, *improve*, *mortality*, *pre*, *reduces*, *survival*

Table 4: Top ten most activated words, as determined by the gating mechanism.

Baseline	Look	Aroma	Palate	Taste
TF-IDF	0.63	0.62	0.62	0.61
LDA	0.73	0.73	0.73	0.73
Doc2Vec	0.61	0.61	0.61	0.61
NVDM	0.68	0.69	0.69	0.70
ABAE	0.50	0.50	0.50	0.50
BoW + Triplet	0.85	0.90	0.90	0.92
NVDM + Triplet	0.90	0.91	0.92	0.95
DSSM + Triplet	0.87	0.90	0.90	0.92
CNN + Triplet	**0.92**	**0.93**	**0.94**	**0.96**

Table 5: AUC results for different representations on the BeerAdvocate data. Models beneath the second line are supervised.

	Look	Aroma	Palate	Taste
Look	0.92	0.89	0.88	0.87
Aroma	0.90	0.93	0.91	0.92
Palate	0.89	0.92	0.94	0.95
Taste	0.90	0.94	0.95	0.96

Table 6: Cross AUC results for different representations on the BeerAdvocate data. Row: Embedding used. Column: Aspect evaluated against.

enrolled patients with HipRepl and other studies. Likewise, studies in which HipRepl was the intervention are grouped in the interventions embedding space, but not in the populations space.

Aspect words. In Table 4, we report the most activated unigrams for each aspect embedding on the decision aids corpus. To derive these we use the outputs of the gating mechanism (Eq. 3), which is applied to all words in the input text. For each word, we average the activations across all abstracts and find the top ten words for each aspect. The words align nicely with the PICO aspects, providing further evidence that our model learns to focus on aspect-specific information.

4.2 Multi-Aspect Reviews

We now turn from the specialized domain of biomedical abstracts to more general applications. In particular, we consider learning disentangled representations of beer, hotel and restaurant reviews. Learned embeddings should capture different aspects, e.g., taste or look in the case of beer.

4.2.1 Beer Reviews (BeerAdvocate)

We conducted experiments on the BeerAdvocate dataset (McAuley et al., 2012), which contains 1.5M reviews of beers that address four aspects: *appearance*, *aroma*, *palate*, and *taste*. Free-text reviews are associated with aspect-specific numerical ratings for each of these, ranging from 1 to 5. We consider ratings < 3 as negative, and > 3 as positive, and use these to generate triplets of reviews. For each aspect a, we construct triplets $(s, d, o)_a$ by first randomly sampling a review d. We then select s to be a review with the same sen-

	Look		Aroma		Palate		Taste	
Look	-	-	0.42	0.60	0.40	0.63	0.38	0.65
Aroma	0.33	0.69	-	-	0.41	0.59	0.41	0.60
Palate	0.32	0.70	0.46	0.54	-	-	0.49	0.52
Taste	0.23	0.80	0.35	0.66	0.33	0.67	-	-

Table 7: 'Decorrelated' cross-AUC results on the BeerAdvocate data, which attempt to mitigate confounding due to overall sentiment being correlated. Each cell reports metrics over subsets of reviews in which the sentiment differs between the row and column aspects. The numbers in each cell are the AUCs w.r.t. sentiment regarding the column aspect achieved using the row and column aspect representations, respectively.

timent with respect to a as d, and o to be a review with the opposite sentiment regarding a. We selected 90K reviews for experiments, such that we had an equal number of positive and negative reviews for each aspect. We only keep words appearing in at least 5 documents, converting all others to `unk`. We truncated reviews to 95 percentile length. We split our data into 80/10/10 ratio for training, validation and testing, respectively.

Baselines. We used the same baselines as for the PICO domain, save for *RR-TF*, which was domain-specific. Here we also evaluate the result of replacing the CNN-based encoder with NVDM, BoW and DSSM based encoders, respectively, each trained using triplet loss.

Hyperparameters and Settings. For the CNN-based encoder, we used settings and hyperparameters as described for the PICO domain. For the BoW encoder, we used 800d output embeddings and a PReLU activation function with ℓ_2 regularization set to $1e$-5. For the NVDM based encoder, we used 200d embeddings.

Metrics. We again performed an IR-type evaluation to assess the utility of representations. For each aspect k, we constructed an affinity matrix A^k such that $A^k_{ij} = sim_k(r_i, r_j)$ for beer reviews r_i and r_j. We consider two reviews similar under a given aspect k if they have the same (dichotomized) sentiment value for said aspect. We compute AUCs for each review and aspect using the affinity matrix A_k. The AUC values are averaged over reviews in the test set to obtain a final AUC metric for each aspect. We also report cross AUC measures in which we use embeddings for aspect k to distinguish reviews under aspect k'.

Results We report the AUC measures for each aspect on our test set using different representations in Table 5. Our model consistently outperforms baseline strategies over all aspects. Unsurprisingly, the model outperforms unsupervised approaches.[4] We realize consistent though modest improvement over triplet-supervised approaches that use alternative encoders.

In Table 6 we present cross AUC evaluations. Rows correspond to the embedding used and columns to the aspect evaluated against. As expected, aspect-embeddings perform better w.r.t. the aspects for which they code, suggesting some disentanglement. However, the reduction in performance when using one aspect representation to discriminate w.r.t. others is not as pronounced as above. This is because aspect ratings are highly correlated: if taste is positive, aroma is very likely to be as well. Effectively, here sentiment entangles all of these aspects.[5]

In Table 7, we evaluate cross AUC performance for beer by first 'decorrelating' the aspects. Specifically, for each cell (k, k') in the table, we first retrieve the subset of reviews in which the sentiment w.r.t. k differs from the sentiment w.r.t. k'. Then we evaluate the AUC similarity of these reviews on the basis of sentiment concerning k' using both k and k' embeddings, yielding a pair of AUCs (listed respectively). We observe that the using k' embeddings to evaluate aspect k' similarity yields better results than using k embeddings.

We present the most activated words for each aspect (as per the gating mechanism) in Table 8. And we present an illustrative review color-coded with aspect-wise gate activations in Table 9. For completeness, we reproduce the top words for aspects discovered using He et al. (2017) in the Appendix; these do not obviously align with the target aspects, which is unsurprising given that this is an unsupervised method.

4.2.2 Hotel & Restaurant Reviews

Finally, we attempt to learn embeddings that disentangle domain from sentiment in reviews. For this we use a combination of TripAdvisor and

---

[4] We are not sure why ABAE (He et al., 2017) performs so poorly on the review corpora. It may simply fail to prominently encode sentiment, which is important for these tasks. We note that this model performs reasonably well on the PICO data above, and qualitatively seems to recover reasonable aspects (though not specifically sentiment).

[5] Another view is that we are in fact inducing representations of <aspect, sentiment> pairs, and only the aspect varies across these; thus representations remain discriminative (w.r.t. sentiment) across aspects.

Look	*attractive*, *beautiful*, *fingers*, *pumpkin*, *quarter*, *received*, *retention*, *sheets*, *sipper*, *well-balanced*
Aroma	*beer*, *cardboard*, *cheap*, *down*, *follows*, *medium-light*, *rice*, *settled*, *skunked*, *skunky*
Palate	*bother*, *crafted*, *luscious*, *mellow*, *mint*, *range*, *recommended*, *roasted*, *tasting*, *weight*
Taste	*amazingly*, *down*, *highly*, *product*, *recommended*, *tasted*, *thoroughly*, *to*, *truly*, *wow*

Table 8: Most activated words for aspects on the beer corpus, as per the gating mechanism.

Yelp! ratings data. The former comprises reviews of hotels, the latter of restaurants; both use a scale of 1 to 5. We convert ratings into positive/negative labels as above. Here we consider aspects to be the domain (hotel or restaurant) and the sentiment (positive or negative). We aim to generate embeddings that capture information about only one of these aspects. We use 50K reviews from each dataset for training and 5K for testing.

Baselines. We use the same baselines as for the BeerAdvocate data, and similarly use different encoder models trained under triplet loss.

Evaluation Metrics. We perform AUC and cross-AUC evaluation as in the preceding section. For the domain aspect, we consider two reviews similar if they are from the same domain, irrespective of sentiment. Similarly, reviews are considered similar with respect to the sentiment aspect if they share a sentiment value, regardless of domain.

Results. In Table 10 we report the AUCs for each aspect on our test set using different representations. Baselines perform reasonably well on the domain aspect because reviews from different domains are quite dissimilar. Capturing sentiment information irrespective of domain is more difficult, and most unsupervised models fail in this respect. In Table 11, we observe that cross AUC results are much more pronounced than for the BeerAdvocate data, as the domain and sentiment are uncorrelated (i.e., sentiment is independent of domain).

5 Related Work

Work in representation learning for NLP has largely focused on improving word embeddings (Levy and Goldberg, 2014; Faruqui et al., 2015; Huang et al., 2012). But efforts have also been made to embed other textual units, e.g. characters (Kim et al., 2016), and lengthier texts including sentences, paragraphs, and documents (Le and Mikolov, 2014; Kiros et al., 2015).

Triplet-based judgments have been used in multiple domains, including vision and NLP, to estimate similarity information implicitly. For example, triplet-based similarity embeddings may be learned using 'crowdkernels' with applications to multi-view clustering (Amid and Ukkonen, 2015). Models combining similarity with neural networks mainly revolve around Siamese networks (Chopra et al., 2005) which use pairwise distances to learn embeddings (Schroff et al., 2015), a tactic we have followed here. Similarity judgments have also been used to generate document embeddings for IR tasks (Shen et al., 2014; Das et al., 2016).

Recently, He et al. (2017) introduced a neural model for aspect extraction that relies on an attention mechanism to identify aspect words. They proposed an autoencoder variant designed to tease apart aspects. In contrast to the method we propose, their approach is unsupervised; discovered aspects may thus not have a clear interpretation. Experiments reported here support this hypothesis, and we provide additional results using their model in the Appendix.

Other recent work has focused on text *generation* from factorized representations (Larsson et al., 2017). And Zhang et al. (2017) proposed a lightly supervised method for *domain adaptation* using aspect-augmented neural networks. They exploited *source* document labels to train a classifier for a *target* aspect. They leveraged sentence-level scores codifying sentence relevance w.r.t. individual aspects, which were derived from terms *a priori* associated with aspects. This supervision is used to construct a composite loss that captures both classification performance on the source task and a term that enforces *invariance* between source and target representations.

There is also a large body of work that uses probabilistic generative models to recover latent structure in texts. Many of these models derive from Latent Dirichlet Allocation (LDA) (Blei et al., 2003), and some variants have explicitly represented topics and aspects jointly for sentiment tasks (Brody and Elhadad, 2010; Sauper et al., 2010, 2011; Mukherjee and Liu, 2012; Sauper and Barzilay, 2013; Kim et al., 2013).

A bit more generally, *aspects* have also been interpreted as properties spanning entire texts, e.g., a perspective or theme which may then color the discussion of topics (Paul and Girju, 2010). This intuition led to the development of the *factorial LDA* family of topic models (Paul and Dredze, 2012; Wallace et al., 2014); these model individ-

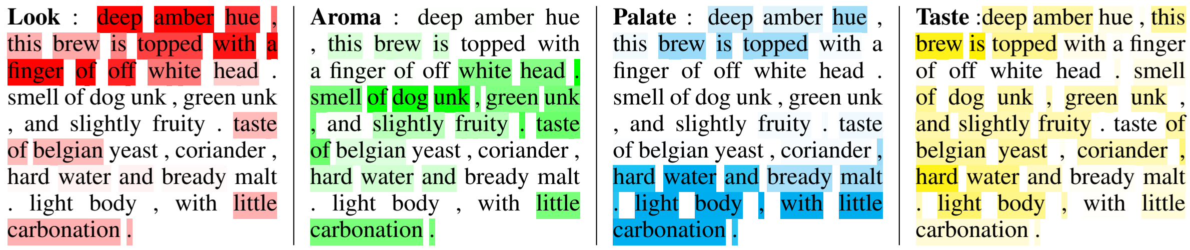

Look	**Aroma**	**Palate**	**Taste**
Look : deep amber hue , this brew is topped with a finger of off white head . smell of dog unk , green unk , and slightly fruity . taste of belgian yeast , coriander , hard water and bready malt . light body , with little carbonation .	**Aroma** : deep amber hue , this brew is topped with a finger of off white head . smell of dog unk , green unk , and slightly fruity . taste of belgian yeast , coriander , hard water and bready malt . light body , with little carbonation .	**Palate** : deep amber hue , this brew is topped with a finger of off white head . smell of dog unk , green unk , and slightly fruity . taste of belgian yeast , coriander , hard water and bready malt . light body , with little carbonation .	**Taste** :deep amber hue , this brew is topped with a finger of off white head . smell of dog unk , green unk , and slightly fruity . taste of belgian yeast , coriander , hard water and bready malt . light body , with little carbonation .

Table 9: Gate activations for each aspect in an example beer review.

Baseline	Domain	Sentiment
TF-IDF	0.59	0.52
Doc2Vec	0.83	0.56
LDA	0.90	0.62
NVDM	0.79	0.63
ABAE	0.50	0.50
BoW + Triplet	**0.99**	0.91
NVDM + Triplet	**0.99**	0.91
DSSM + Triplet	**0.99**	0.90
CNN + Triplet	**0.99**	**0.92**

Table 10: AUC results for different representations on the Yelp!/TripAdvisor Data. Models beneath the second line are supervised.

Baseline	Domain	Sentiment
Domain	0.988	0.512
Sentiment	0.510	0.917

Table 11: Cross AUC results for different representations for Yelp!/TripAdvisor Dataset.

ual word probability as a product of multiple latent factors characterizing a text. This is similar to the Sparse Additive Generative (SAGE) model of text proposed by Eisenstein et al. (2011).

6 Conclusions

We have proposed an approach for inducing disentangled representations of text. To learn such representations we have relied on supervision codified in aspect-wise similarity judgments expressed as document triplets. This provides a general supervision framework and objective. We evaluated this approach on three datasets, each with different aspects. Our experimental results demonstrate that this approach indeed induces aspect-specific embeddings that are qualitatively interpretable and achieve superior performance on information retrieval tasks.

Going forward, disentangled representations may afford additional advantages in NLP, e.g., by facilitating transfer (Zhang et al., 2017), or supporting aspect-focused summarization models.

7 Acknowledgements

This work was supported in part by National Library of Medicine (NLM) of the National Institutes of Health (NIH) award R01LM012086, by the Army Research Office (ARO) award W911NF1810328, and research funds from Northeastern University. The content is solely the responsibility of the authors.

References

Ehsan Amid and Antti Ukkonen. 2015. Multiview triplet embedding: Learning attributes in multiple maps. In *International Conference on Machine Learning*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan).

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *CVPR 2005. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.*

Aaron M Cohen, William R Hersh, K Peterson, and Po-Yin Yen. 2006. Reducing workload in systematic review preparation using automated citation classification. *Journal of the American Medical Informatics Association*, 13(2).

Arpita Das, Harish Yenala, Manoj Chinnakotla, and Manish Shrivastava. 2016. Together we stand: Siamese networks for similar question retrieval. In

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.

Jacob Eisenstein, Amr Ahmed, and Eric P Xing. 2011. Sparse Additive Generative Models of Text. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An Unsupervised Neural Attention Model for Aspect Extraction. In *Proceedings of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. Beta-Vae: Learning Basic Visual Concepts with a Constrained Variational Framework. *International Conference on Learning Representations*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Theofanis Karaletsos, Serge Belongie, and Gunnar Rätsch. 2015. Bayesian representation learning with oracle constraints. *arXiv preprint arXiv:1506.05011*.

Suin Kim, Jianwen Zhang, Zheng Chen, Alice H Oh, and Shixia Liu. 2013. A Hierarchical Aspect-Sentiment Model for Online Reviews. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*.

Maria Larsson, Amanda Nilsson, and Mikael Kågebäck. 2017. Disentangled Representations for Manipulation of Sentiment in Text. *arXiv preprint arXiv:1712.10066*.

Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the Association of Computational Linguistics (ACL)*.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *2012 IEEE 12th International Conference on Data Mining (ICDM)*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics.

Michael Paul and Mark Dredze. 2012. Factorial LDA: Sparse multi-dimensional text models. In *Advances in Neural Information Processing Systems*.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing. In *Proceedings of the 5th International Symposium on Languages in Biology and Medicine*.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. A Hierarchical Model of Reviews for Aspect-based Sentiment Analysis. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

David L Sackett, William M C Rosenberg, J A Muir Gray, R Brian Haynes, and W Scott Richardson. 1996. Evidence based medicine: what it is and what it isn't. *Bmj*, 312(7023).

Christina Sauper and Regina Barzilay. 2013. Automatic aggregation by joint modeling of aspects and values. *Journal of Artificial Intelligence Research*.

Christina Sauper, Aria Haghighi, and Regina Barzilay. 2010. Incorporating content structure into text analysis applications. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Christina Sauper, Aria Haghighi, and Regina Barzilay. 2011. Content models with attitude. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*.

D Stacey, F Légaré, N F Col, C L Bennett, M J Barry, K B Eden, H Thomas, A Lyddiatt, R Thomson, L Trevena, and J H C Wu. 2014. Decision aids for people facing health treatment or screening decisions. *Cochrane Database of Systematic Reviews*, 1.

Andreas Veit, Serge Belongie, and Theofanis Karaletsos. 2017a. Conditional Similarity Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Andreas Veit, Serge Belongie, and Theofanis Karaletsos. 2017b. Conditional Similarity Networks. In *Computer Vision and Pattern Recognition (CVPR)*.

Byron C Wallace, Joël Kuiper, Aakash Sharma, Mingxi Brian Zhu, and Iain J Marshall. 2016. Extracting PICO sentences from clinical trial reports using supervised distant supervision. *Journal of Machine Learning Research*, 17(132).

Byron C Wallace, Michael J Paul, Urmimala Sarkar, Thomas A Trikalinos, and Mark Dredze. 2014. A large-scale quantitative analysis of latent factors and sentiment in online doctor reviews. *Journal of the American Medical Informatics Association*, 21(6).

Byron C Wallace, Thomas A Trikalinos, Joseph Lau, Carla Brodley, and Christopher H Schmid. 2010. Semi-automated screening of biomedical citations for systematic reviews. *BMC bioinformatics*, 11(1).

William Whitney. 2016. Disentangled Representations in Neural Models. *arXiv preprint arXiv:1602.02383*.

Michael J Wilber, Iljung S Kwak, and Serge J Belongie. 2014. Cost-effective hits for relative similarity comparisons. In *Second AAAI Conference on Human Computation and Crowdsourcing*.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented Adversarial Networks for Domain Adaptation. In *Transactions of the Association for Computational Linguistics*.

Multi-Source Domain Adaptation with Mixture of Experts

Jiang Guo, Darsh J Shah and Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
`{jiang_guo, darsh, regina}@csail.mit.edu`

Abstract

We propose a mixture-of-experts approach for unsupervised domain adaptation from multiple sources. The key idea is to explicitly capture the relationship between a target example and different source domains. This relationship, expressed by a *point-to-set* metric, determines how to combine predictors trained on various domains. The metric is learned in an unsupervised fashion using *meta-training*. Experimental results on sentiment analysis and part-of-speech tagging demonstrate that our approach consistently outperforms multiple baselines and can robustly handle negative transfer.[1]

1 Introduction

Typical domain adaptation methods are designed to transfer supervision from a single source domain. However, in many practical applications, we have access to multiple sources. For instance, in sentiment analysis of product reviews, we can often transfer from a wide range of product domains, rather than one. This can be particularly promising for target domains which do not match any one available source well. For example, the *Kitchen* product domain may include reviews on *pans*, *cookbooks* or *electronic devices*, which cannot be perfectly aligned to a single source such as *Cookware*, *Books* or *Electronics*. By intelligently aggregating distinct and complementary information from multiple sources, we may be able to better fit the target distribution.

A straightforward approach to utilizing data from multiple sources is to combine them into a single domain. This strategy, however, does not account for distinct relations between individual sources and the target example. Constructing a common feature space for this heterogeneous collection may wash out informative characteristics of individual domains and also lead to *negative transfer* (Rosenstein et al., 2005).

Therefore, we propose to explicitly model the relationship between different source domains and target examples. We hypothesize that different source domains are aligned to different sub-spaces of the target domain. Specifically, in this paper, we model the domain relationship with a mixture-of-experts (MoE) approach (Jacobs et al., 1991b). For each target example, the predicted posterior is a weighted combination of all the experts' predictions. The weights reflect the proximity of the example to each source domain. Our model learns this *point-to-set* metric automatically, without additional supervision.

We define the *point-to-set* metric using Mahalanobis distance (Weinberger and Saul, 2009) between individual examples and a set (i.e. domain), which are computed within the hidden representation space of our model. The main challenge is to learn this metric in an unsupervised setting. We address it through a *meta-training* procedure, in which we create multiple *meta*-tasks of domain adaptation from the source domains. In each *meta*-task, we pick one of the source domains as *meta-target*, and the rest source domains as *meta-sources*. By minimizing the loss using the MoE predictions on *meta-target*, we are able to learn both the model and the metric simultaneously. To further improve transfer quality, we align the encoding space of our target and source domains via adversarial learning.

We evaluate our approach on sentiment analysis using the benchmark multi-domain Amazon reviews dataset (Chen et al., 2012; Ziser and Reichart, 2017) as well as on part-of-speech (POS) tagging using the SANCL dataset (Petrov and McDonald, 2012). Experiments show that our ap-

---

[1] Our code and data are available at `https://github.com/jiangfeng1124/transfer`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4694–4703
Brussels, Belgium, October 31 - November 4, 2018.

proach consistently improves the adaptation results over the best single-source model and a unified multi-source model. On average, we achieve a 7% relative error reduction on the Amazon reviews dataset, and a 13% on the SANCL dataset. Importantly, the POS tagging experiments on the SANCL dataset demonstrate that our method is able to robustly handle negative transfer from unrelated sources (e.g., *Twitter*) and utilize it effectively to consistently improve performance.

2 Related Work

Unsupervised domain adaptation Most existing domain adaptation methods focus on aligning the feature space between source and target domains to reduce the domain shift (Ben-David et al., 2007; Blitzer et al., 2007, 2006; Pan et al., 2010). Our approach is close to the representation learning approaches, such as the denoising autoencoder (Glorot et al., 2011), the marginalized stacked denoising autoencoders (Chen et al., 2012), and domain adversarial networks (Tzeng et al., 2014; Ganin et al., 2016; Zhang et al., 2017; Shen et al., 2018).

In contrast to these previous approaches, however, our approach not only learns a shared representation space that generalizes well to the target domain, but also captures informative characteristics of individual source domains.

Multi-Source domain adaptation The main challenge in using multiple sources for domain adaptation is in learning domain relations. Some approaches assume that all source domains are equally important to the target domain (Li and Zong, 2008; Luo et al., 2008; Crammer et al., 2008). Others learn a global domain similarity metric using labeled data in a supervised fashion (Yang et al., 2007; Duan et al., 2009; Yu et al., 2018). Alternatively, Mansour et al. (2009) and Bhatt et al. (2016) utilize unlabeled data of the target domain to find a distribution weighted combination of the source domains or to construct an auxiliary training set of the source domain instances close to the target domain instances. Recent adversarial methods on multi-source domain adaptation (Zhao et al., 2018; Chen and Cardie, 2018) align source domains to the target domains globally, without accounting for the distinct importance of each source with respect to a specific target example.

The work most related to ours is by Kim et al. (2017). They also model the *example-to-domain* relations, but use an attention mechanism. The attention module is learned using limited training data from the target domain in a supervised fashion. Our method, however, works in an unsupervised setting without utilizing any labeled data from the target domain.

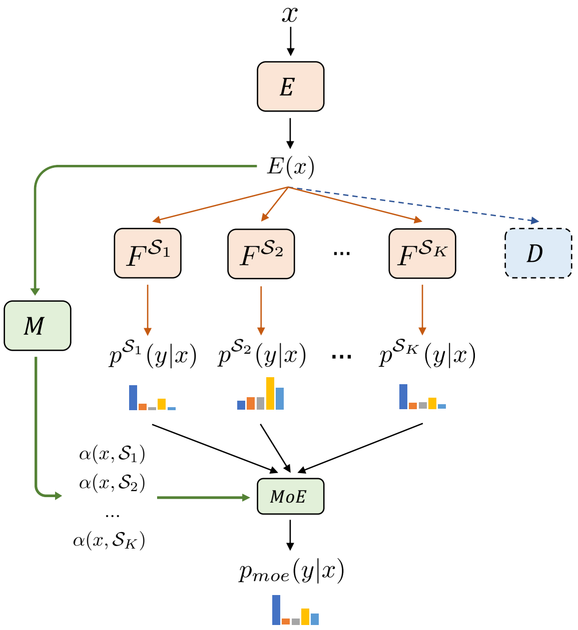

Figure 1: Architecture of the MoE model. E is the encoder which maps an input x to a hidden representation $E(x)$; $F^{\mathcal{S}_i}$ is the classifier on the i^{th} source domain; D is the critic that is only used during adversarial training. M is the metric learning component, which takes the encoding of x and source domains ($\mathcal{S}_{1:K}$) as input and computes α.

3 Methodology

Problem definition We follow the unsupervised multi-source domain adaptation setup, assuming access to labeled training data from K source domains: $\{\mathcal{S}_i\}_{i=1}^K$ where $\mathcal{S}_i \triangleq \{(x_t^{\mathcal{S}_i}, y_t^{\mathcal{S}_i})\}_{t=1}^{|\mathcal{S}_i|}$, and (optionally) unlabeled data from a target domain: $\mathcal{T} \triangleq \{x_t^{\mathcal{T}}\}_{t=1}^{|\mathcal{T}|}$. The goal is to learn a model using the source domain data, that generalizes well to the target domain.

Notations For the rest of the paper, we denote an individual example as x, and a batch of examples as $\mathbf{x}$. We use superscript to denote the domain from which an example is sampled, and use subscript to denote the index of an example.

3.1 Overview of Our Approach

We model the multiple source domains as a mixture of experts, and learn a *point-to-set* metric α to weight the experts for different target examples. The metric is learned in an unsupervised manner.

Our model consists of four key components as shown in Figure 1, namely the *encoder* (E), *classifier* (F), *metric* (M) and *adversary* (D). We use a typical neural multi-task learning architecture (Caruana, 1997), with a shared encoder across all sources, and domain-specific classifiers ($\{F^{\mathcal{S}_i}\}_{i=1}^{K}$). Each input is first encoded with E, and then fed to each classifier to obtain the domain-specific predictions (i.e. posteriors). The final predictions are then weighted based on the *metric* (see Equation 1).

We start by describing the representation learning component.

3.2 Representation

Our goal is to design an encoder that supports transfer, while maintaining source domain-specific information. Depending on different tasks and datasets, we select appropriate encoders — MLP, CNN or LSTM (see Section 4.3 for details).

We further add an adversarial module (D) on top of the encoder, in order to align the target domain with the sources. D is typically designed as a parameterized classifier in domain adversarial networks (Ganin et al., 2016; Zhang et al., 2017), which is trained jointly with the encoder and the classifiers through a minimax game. Here, we instead use Maximum Mean Discrepancy (MMD) (Gretton et al., 2012) as our adversary. This distance metric measures the discrepancy between two distributions explicitly in a non-parametric manner, greatly simplifying the training procedure compared to domain adversarial networks which use an additional domain classifier module.

3.3 Mixture of Experts

Given an example x from the target domain, we model its posterior distribution as a mixture of posteriors produced by models trained on different source domain data:

$$\begin{aligned} p_{moe}(y|x) &= \sum_{i=1}^{K} \alpha(x, \mathcal{S}_i) \cdot p^{\mathcal{S}_i}(y|x) \\ &= \sum_{i=1}^{K} \alpha(x, \mathcal{S}_i) \cdot \texttt{softmax}\big(\mathbf{W}^{\mathcal{S}_i} E(x)\big) \end{aligned} \tag{1}$$

$p^{\mathcal{S}_i}$ is the posterior distribution produced by the i^{th} source classifier $F^{\mathcal{S}_i}$ (the i^{th} expert). $\mathbf{W}^{\mathcal{S}_i}$ is the output layer weights of $F^{\mathcal{S}_i}$, α is a parameterized metric function that measures how much confidence we put in the specific source expert for a given example x.[2] To derive α, we first define a *point-to-set* Mahalanobis distance metric between an example x and a set $\mathcal{S}$:

$$d(x, \mathcal{S}) = \Big(\big(E(x) - \mu^{\mathcal{S}}\big)^{\top} \mathbf{M}^{\mathcal{S}} \big(E(x) - \mu^{\mathcal{S}}\big)\Big)^{\frac{1}{2}}$$

where $\mu^{\mathcal{S}}$ is the mean encoding of $\mathcal{S}$. In its original form, the matrix $\mathbf{M}^{\mathcal{S}}$ played the role of the inverse covariance matrix. However, computing the inverse of the covariance matrix is both time consuming and numerically unstable in practice. Here we allow $\mathbf{M}$ to denote any positive semi-definite matrix which is to be estimated during training (Weinberger and Saul, 2009). To guarantee the positive semi-definiteness of $\mathbf{M}$, we approximate $\mathbf{M}$ with $\mathbf{M} = \mathbf{U}\mathbf{U}^{\top}$, where $\mathbf{U} \in \mathbb{R}^{h \times r}$, h is the dimension of hidden representations and r is a hyper-parameter controlling the rank of $\mathbf{M}$.

Based on the distance metric, we further derive a confidence score $e(x, \mathcal{S}_i) = f\big(d(x, \mathcal{S}_i)\big)$ for each specific expert. The final metric values $\alpha(x, \mathcal{S}_i)$ are then obtained by normalizing these confidence scores:

$$\alpha(x, \mathcal{S}_i) = \frac{\exp\big(e(x, \mathcal{S}_i)\big)}{\sum_{j=1}^{K} \exp\big(e(x, \mathcal{S}_j)\big)} \tag{2}$$

Here, we explain our design of $e(x, \mathcal{S})$ on two tasks, respectively *binary classification* and *sequence tagging*, which are also used for evaluation in this paper (Section 4).

Binary classification The *point-to-set* Mahalanobis distance metric measures the distance between an example x and the mean encoding of $\mathcal{S}$, i.e. $\mu^{\mathcal{S}}$, while taking into account the (pseudo) covariance of $\mathcal{S}$. In binary classification, however, the mean vector $\mu^{\mathcal{S}}$ is likely to be located near the decision boundary, particularly under a balanced

---

[2] In traditional MoE frameworks (Jacobs et al., 1991a,b; Shazeer et al., 2017), α is commonly realized as a "gating network", which produces a normalized weight vector that determines the combination of experts depending solely on the input example x. However, simple gating networks do not yield promising results in our scenario. We hypothesize that both the input instance and the underlying domain distribution should be captured for determining the credit assignment.

setting. Therefore, a small $d(x, \mathcal{S})$ actually implies lower confidence of the corresponding classifier, which is counter-intuitive. To this end, we instead define the confidence $e(x, \mathcal{S})$ as the difference between the distances from x to each category of $\mathcal{S}$:

$$e(x, \mathcal{S}) = \left| d(x, \mathcal{S}^+) - d(x, \mathcal{S}^-) \right|$$

Here $\mathcal{S}^+$ and $\mathcal{S}^-$ stand for the positive space and negative space of $\mathcal{S}$ respectively. Consequently, if x is either far away from $\mathcal{S}$ (i.e., x is not in the manifold of $\mathcal{S}$) or near the classification boundary, we will get a small $e(x, \mathcal{S})$ indicating a low confidence to the corresponding prediction. On the contrary, if x is much closer to a specific category of $\mathcal{S}$ than other categories, the classifier will get a higher confidence.

Sequence tagging For sequence tagging tasks (e.g., POS tagging), we compute the distance metric at the *token level*.[3] Unlike in *binary classification*, the decision boundary here is more complicated, and the label distribution is typically imbalanced. The mean vector $\mu^{\mathcal{S}}$ is unlikely to be located at the decision boundary. So we directly use the (reverse) distance as the confidence value for each token x:

$$e(x, \mathcal{S}) = -d(x, \mathcal{S})$$

3.4 Training

Since we do not have annotated data in the target domain, we have to learn our model in an unsupervised fashion. Inspired by the recent progress on few-shot learning with metric-based models such as matching network (Vinyals et al., 2016; Yu et al., 2018) and prototypical network (Snell et al., 2017), we propose the following *meta-training* approach. Given K source domains, each source domain will be considered as a target, referred to as ***meta-target***, with the rest of the source domains as ***meta-sources***. This way, we obtain K (*meta-sources*, *meta-target*) training pairs for domain adaptation. Then, we apply our MoE formulation over these *meta-training* pairs to learn the metric. At testing time, the metric will be applied to all the K source domains for each example in the target domain.

We optimize two main objectives: the MoE objective and the multi-task learning (MTL) objective.

MoE objective For each example in each *meta-target* domain, we compute its MoE posterior using the corresponding *meta-sources*. Therefore, we get the following MoE loss over the entire multi-source training data:

$$\begin{aligned}\mathcal{L}_{moe} &= -\sum_{i=1}^{K} \sum_{j=1}^{|\mathcal{S}_i|} \log p_{moe}(y_j^{\mathcal{S}_i} | x_j^{\mathcal{S}_i}) \\ &= -\sum_{i=1}^{K} \sum_{j=1}^{|\mathcal{S}_i|} \log \sum_{l=1, l \neq i}^{K} \alpha(x, \mathcal{S}_l) \cdot p^{\mathcal{S}_l}(y_j^{\mathcal{S}_i} | x_j^{\mathcal{S}_i})\end{aligned} \tag{3}$$

Note that α is normalized over the *meta-sources* for each *meta-target*, rather than over all the K sources.

MTL objective For each *meta-target*, we further optimize a supervised cross-entropy loss using the corresponding labels. All supervised objectives are optimized jointly with the encoder being shared, resulting in the following multi-task learning objective:

$$\mathcal{L}_{mtl} = -\sum_{i=1}^{K} \sum_{j=1}^{|\mathcal{S}_i|} \log p^{\mathcal{S}_i}(y_j^{\mathcal{S}_i} | x_j^{\mathcal{S}_i}) \tag{4}$$

Adversary-augmented MoE We use MMD (Gretton et al., 2012) as the adversary to minimize the divergence between the marginal distribution of target domain and source domains. Specifically, at each training epoch, given the K batches $\{\mathbf{x}^{\mathcal{S}_1}, \mathbf{x}^{\mathcal{S}_2}, ..., \mathbf{x}^{\mathcal{S}_K}\}$ from all the source domains, we sample a batch (unlabeled) $\mathbf{x}^{\mathcal{T}}$ from our target domain, and minimize the MMD:

$$\mathcal{L}_{adv} = \texttt{MMD}^2(\mathbf{x}^{\mathcal{S}_1} \cup ... \cup \mathbf{x}^{\mathcal{S}_K}, \mathbf{x}^{\mathcal{T}}) \tag{5}$$

where

$$\texttt{MMD}(\mathcal{D}^{\mathcal{S}}, \mathcal{D}^{\mathcal{T}}) = \left\| \frac{1}{|\mathcal{D}^{\mathcal{S}}|} \sum_{x_s \in \mathcal{D}^{\mathcal{S}}} \phi(E(x_s)) - \frac{1}{|\mathcal{D}^{\mathcal{T}}|} \sum_{x_t \in \mathcal{D}^{\mathcal{T}}} \phi(E(x_t)) \right\|_{\mathcal{H}}$$

measures the discrepancy between $\mathcal{D}^{\mathcal{S}}$ and $\mathcal{D}^{\mathcal{T}}$ based on Reproducing Kernel Hilbert Space (RKHS). $\phi(\cdot)$ is the feature map induced by a universal kernel. We follow Bousmalis et al. (2016) and use a linear combination of multiple RBF kernels: $\kappa(\boldsymbol{h}_i, \boldsymbol{h}_j) = \Sigma_n \exp(-\frac{1}{2\sigma_n} \|\boldsymbol{h}_i - \boldsymbol{h}_j\|^2)$.

---

[3] This actually makes it a multi-class classification problem with respect to every token of a sequence.

Algorithm 1 Training Procedure

```
**Input**: multi-source domain data $\mathcal{S} = \{\mathcal{S}_i\}_{i=1}^{K}$, target domain data $\mathcal{T}$
**Hyper-parameters**: mini-batch size $m$, coefficients for different losses: $\lambda$, $\gamma$ and $\eta$
**repeat**
    Sample $K$ source mini-batches $\{(\mathbf{x}^{\mathcal{S}_i}, \mathbf{y}^{\mathcal{S}_i})\}_{i=1}^{K}$ from $\mathcal{S}$ and a target mini-batch $\mathbf{x}^{\mathcal{T}}$ from $\mathcal{T}$
    $\mathcal{L}_{mtl}, \mathcal{L}_{moe}, \mathcal{L}_{adv}, \mathcal{R}_h \leftarrow 0$
    **for** $t = 1$ to $K$ **do**
        Set ***meta-target*** as $\mathcal{T}^{meta} \triangleq \tilde{\mathcal{S}}_t \triangleq (\mathbf{x}^{\mathcal{S}_t}, \mathbf{y}^{\mathcal{S}_t})$
        Set ***meta-sources*** as $\mathcal{S}^{meta} \triangleq \{\tilde{\mathcal{S}}_i\}_{i=1, i\neq t}^{K}$, where $\tilde{\mathcal{S}}_i \triangleq (\mathbf{x}^{\mathcal{S}_i}, \mathbf{y}^{\mathcal{S}_i})$
        Compute cross-entropy loss over $\mathcal{T}^{meta}$, and add to $\mathcal{L}_{mtl}$
        Compute Mahalanobis metric $\alpha(x, \mathcal{S}')$ for each $x \in \mathcal{T}^{meta}$ and $\mathcal{S}' \in \mathcal{S}^{meta}$        ▷ Eq. (2)
        Compute MoE loss over $(\mathcal{S}^{meta}, \mathcal{T}^{meta})$ using $\boldsymbol{\alpha}$, and add to $\mathcal{L}_{moe}$
        Compute entropy of $\boldsymbol{\alpha}$ over $\mathcal{S}$, and add to $\mathcal{R}_h$
    **end for**
    Compute MMD between $\mathbf{x}^{\mathcal{T}}$ and $\cup_{i=1}^{K} \mathbf{x}^{\mathcal{S}_i}$, and add to $\mathcal{L}_{adv}$        ▷ Eq. (5)
    Update parameters via backpropagating gradients of the total loss $\mathcal{L}$        ▷ Eq. (7)
**until** converge
```

Entropy regularization In the *meta-training* process, for each example x in *meta-target*, we know exactly from which source x is sampled. This provides additional insight that the α distribution is skewed, which can be utilized as a soft constraint. Therefore, we propose to regularize the entropy of the α distribution over all the sources, rather than *meta-sources*:[4]

$$H\big(\boldsymbol{\alpha}(x, \cdot)\big) = -\sum_{l=1}^{K} \alpha(x, \mathcal{S}_l) \cdot \log \alpha(x, \mathcal{S}_l)$$

$$\mathcal{R}_h = \sum_{i=1}^{K} \sum_{j=1}^{|\mathcal{S}_i|} H\big(\boldsymbol{\alpha}(x_j^{\mathcal{S}_i}, \cdot)\big) \qquad (6)$$

Joint learning Our final objective is the weighted combination of each individual component loss:

$$\begin{aligned} \mathcal{L} = \lambda \cdot \mathcal{L}_{moe} &+ (1-\lambda) \cdot \mathcal{L}_{mtl} \\ &+ \gamma \cdot \mathcal{L}_{adv} \\ &+ \eta \cdot \mathcal{R}_h \end{aligned} \qquad (7)$$

where λ controls the balance of the MoE loss and MTL loss. γ is set to 0 in *non-adversarial* setting when unlabeled data from the target domain is not provided. Additionally, it would be straightforward to add an MoE loss for labeled data in the target domain if they are available, thus extending our framework to a setting where we have few-shot target annotations. The training process is shown in Algorithm 1.

4 Experimental Setup

4.1 Task and Dataset

Sentiment classification We use the multi-domain Amazon reviews dataset (Blitzer et al., 2007), one of the standard benchmark datasets for domain adaptation. It contains reviews on four domains: *Books* (B), *DVDs* (D), *Electronics* (E), and *Kitchen appliances* (K).

We follow the specific experiment settings proposed by Chen et al. (2012) (CHEN12) and Ziser and Reichart (2017) (ZISER17).

1. In CHEN12, each domain has 2,000 labeled examples for training (1,000 positive and 1,000 negative), and the target test set has 3,000 to 6,000 examples.[5]
2. In ZISER17, each domain also has 2,000 labeled examples (1,000 positive and 1,000 negative), sampled differently from CHEN12.

For each dataset, we conduct experiments by selecting the target domain in a round-robin fash-

---

[4] Alternatively, we can directly exploit this supervision and minimize the KL divergence of the α distribution and its ground truth one-hot distribution. In practice, however, we found it beneficial to allow examples from one domain to be attended to different sources. This observation may be attributed to the fact that each domain indeed consists of multiple latent sub-domains.

[5] This dataset has been processed by the author to TF-IDF representations, using the 5,000 most frequent unigram and bigram features, thus word order information is not available.

ion. Following the protocol in previous work, we use cross-validation over source domains for hyper-parameters selection for each adaptation task (Zhao et al., 2018). When training with an adversary, we use the 2,000 examples training set of the target domain as the unlabeled data in both the settings. In ZISER17, the same data is also used for test, resulting in a transductive setting.

Part-of-Speech tagging We further consider a sequence tagging task, where the metric is computed over the token-level encodings and multi-class predictions are made at the token (word) level. We use the SANCL dataset (Petrov and McDonald, 2012) which contains part-of-speech (POS) tagging annotations in 5 web domains: *Emails*, *Weblogs*, *Answers*, *Newsgroups*, and *Reviews*. Among these, *Newsgroups*, *Reviews*, and *Answers* have both a validation and a test set, and are used as target domains. The test set from *Weblogs* and *Emails* are used as individual source domains. The tagging is performed using the Universal POS tagset (Petrov et al., 2012). We also use *Twitter* (Liu et al., 2018) as an additional training source. Since it differs substantially from other sources and the target domain, we can assess our model's ability to handle negative transfer. We consider 750 sentences from each SANCL source domain for training, and up to 2,250 sentences from the *Twitter* dataset to magnify the negative transfer. The validation set in the standard split of each target domain is used for hyper-parameters selection and early-stopping in our experiments.

4.2 Baselines

We verify the efficacy of our approach (MoE) in *non-adversarial* and *adversarial* settings respectively. In both settings, we compare our approach against the following two baselines:

- **best-SS**: the best single-source adaptation model among all the sources.
- **uni-MS**: the unified multi-source adaptation model, which is trained using the combination of all the source domain data with single-source transfer methods. uni-MS is a common and strong baseline for multi-source domain adaptation (Zhao et al., 2018).

For the rest of the paper, we name the adversarial counterpart of the models as $*$**-A**.

In the adversarial setting on CHEN12, in addition to best-SS and uni-MS with adversarial loss, we further compare with the following two systems that also utilize unlabeled data from target domain.

- **mSDA**: the marginalized stacked denoising autoencoder (Chen et al., 2012). mSDA outperforms prior deep learning and shallow learning approaches such as structural correspondence learning (Blitzer et al., 2007) and denoising autoencoder (Glorot et al., 2011).
- **MDAN**: the multi-source domain adversarial network (Zhao et al., 2018). MDAN gives the state-of-the-art performance for multi-source domain adaptation on CHEN12. It generalizes the domain adversarial network to multiple source domain adaptation by selectively backpropagating the domain discrimination loss according to domain classification error.

4.3 Implementation Details

For CHEN12, since the dataset is in TF-IDF format and the word ordering information is not available, we use a multilayer perceptron (MLP) with an input layer of 5,000 dimensions and one hidden layer of 500 dimensions as our encoder. For ZISER17, we instead use a convolutional neural network (CNN) encoder with a combination of kernel widths 3 and 5 (Kim, 2014), each with one hidden layer of size 150, which are then concatenated to a 300 dimension representation.[6]

For the POS tagging encoder, we use a hierarchical bidirectional LSTM (BiLSTM) network, which contains a character-level BiLSTM for generating individual word representations, followed by a word-level BiLSTM that generates contextualized word representations.

For MMD, we follow Bousmalis et al. (2016) and use 19 RBF kernels with the standard deviation parameters ranging from 10^{-6} to 10^{6}.[7]

All the models were trained using Adam with weight decay. Learning rate is set to 10^{-4} for CHEN12 and 10^{-3} for ZISER17 and POS tagging. We use mini-batches of 32 samples from each domain. We tune the coefficients λ, η for each adaptation task. γ is set to 1 for all experiments.

---

[6] Note that with a more extensive architecture search, we are likely to achieve better results. This, however, is not the main focus of this work.

[7] Detailed values are presented in the supplementary material in Bousmalis et al. (2016).

SETTING	NON-ADVERSARIAL			ADVERSARIAL				
	best-SS	uni-MS	MoE	mSDA†	MDAN	best-SS-A	uni-MS-A	MoE-A
D,E,K–B	75.43	78.43	**79.42**	76.98	78.63	80.07	80.25	**80.87**
B,E,K–D	81.23	82.49	**83.35**	78.61	80.65	82.68	83.30	**83.99**
B,D,K–E	85.51	84.79*	**86.62**	81.98	85.34	86.32	85.96*	**86.38**
B,D,E–K	86.83	87.00	**87.96**	84.33	86.26	87.05	87.55	**88.06**
Average	*82.25*	*83.18*	***84.34***	*80.48*	*82.72*	*84.03*	*84.27*	***84.83***

Table 1: Multi-Source domain adaptation accuracy on Amazon dataset of CHEN12. * indicates negative transfer, i.e., the unified multi-source model underperforms the best single-source model. mSDA† is not an adversarial approach, but utilizes unlabeled data from target domain.

SETTING	NON-ADVERSARIAL			ADVERSARIAL		
	best-SS	uni-MS	MoE	best-SS-A	uni-MS-A	MoE-A
D,E,K–B	85.35	87.00	**87.55**	86.85	87.55	**87.85**
B,E,K–D	85.25	86.80	**87.85**	86.00	87.40	**87.65**
B,D,K–E	86.80	88.30	**89.20**	88.90	89.35	**89.50**
B,D,E–K	88.90	89.65	**90.45**	89.95	90.35	**90.45**
Average	*86.58*	*87.94*	***88.76***	*87.93*	*88.66*	***88.86***

Table 2: Multi-Source domain adaptation accuracy on Amazon dataset of ZISER17.

5 Results

5.1 Sentiment Analysis on Amazon Reviews

We report our results on the Amazon reviews datasets in Table 1 (CHEN12) and Table 2 (ZISER17). Our approach (MoE) consistently achieves the best performance across different settings and tasks.

The results clearly demonstrate the value of using multiple sources. In most cases, even a unified model performs better than the oracle best single source. By smartly combining all the sources, our model outperforms the unified model significantly. One exception is the task of "B,D,K-E" in CHEN12, where the unified multi-source model doesn't improve over the best single source model, constituting a *negative transfer* scenario. However, even in this scenario, our approach still performs significantly better, demonstrating its robustness in handling *negative transfer*.

Impact of adversarial adaptation We achieve consistent improvements over the baseline systems with the addition of the adversarial loss. In most cases, MoE also achieves additional improvement (e.g., 79.42% *vs.* 80.87% in "D,E,K-B"). We notice that in some cases, e.g., "B,D,K-E" in CHEN12 and "B,E,K-D" in ZISER17, the adversarial loss doesn't help MoE. This might be attributed to the fact that by aligning the target distribution with the source domains, the representation space becomes more compact, thus making it more difficult to capture source domain-specific characteristics and increasing the difficulty of metric learning in MoE.

Analysis on the metric (α) Figure 2 visualizes the distribution of α values, learned by our model in different tasks, across the source domains. The visualization is based on 200 examples for each domain randomly sampled from the corresponding test set. From the heatmap we can see that for a specific target domain, different examples may have different α distributions. Moreover, for most examples, the α distribution is skewed, indicating that our model draws on a few most informative source domains.

Figure 3 exemplifies the above point. For instance, the first review about "charger" and "battery" is closer to the *Electronics* source domain. This relation is successfully captured by the α distribution produced by our model.

We further investigate the impact of entropy regularization over α. Table 3 summarizes the ablation test results of entropy regularization ($\eta = 0$)

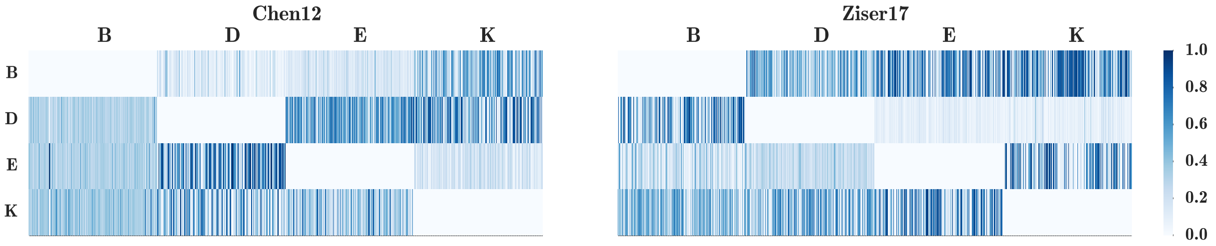

Figure 2: α distributions across source domains for randomly selected 200 examples in each target domain of Chen12 (left) and Ziser17 (right). Columns represent target domains and rows represent sources.

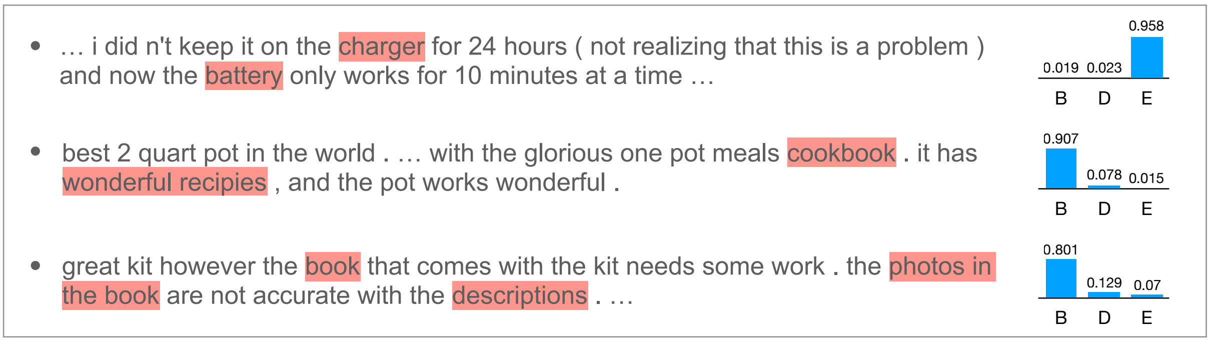

- … i did n't keep it on the charger for 24 hours (not realizing that this is a problem) and now the battery only works for 10 minutes at a time …
- best 2 quart pot in the world . … with the glorious one pot meals cookbook . it has wonderful recipies , and the pot works wonderful .
- great kit however the book that comes with the kit needs some work . the photos in the book are not accurate with the descriptions . …

Figure 3: Examples of *Kitchen* (K) reviews in Ziser17 and their α distribution over *Books* (B), *DVDs* (D) and *Electronics* (E). The manually highlighted words indicate the specific Kitchen products described in the reviews.

Setting	Chen12 (w/o $\mathcal{R}_h$)		Ziser17 (w/o $\mathcal{R}_h$)	
	MoE	MoE-A	MoE	MoE-A
D,E,K–B	*-0.70*	*-0.51*	*-0.75*	*-0.60*
B,E,K–D	*-0.67*	*-0.41*	*-0.05*	*-1.20*
B,D,K–E	*-1.93*	*-0.44*	*-0.70*	*-0.60*
B,D,E–K	*-0.49*	*-0.09*	*-0.50*	*+0.30*

Table 3: Ablation test of entropy regularizer on Chen12 and Ziser17 (decrease in accuracy).

on Chen12 and Ziser17. It shows that entropy regularization benefits our model under both non-adversarial and adversarial settings.

5.2 Part-of-Speech Tagging

Table 4 summarizes our results on POS tagging. Again, our approach consistently achieves the best performance across different settings and tasks. Adding *Twitter* as a source leads to a drop in performance for the unified model, as a result of negative transfer. Our method, however, robustly handles negative transfer and manages to even benefit from this additional source.

Impact of negative transfer Table 5 presents the α distribution learned by the metric, on average for all tokens of the target domain. As we can see, our model (MoE-A) effectively learns to decrease the weights on *Twitter*, demonstrating again its ability to alleviate negative transfer.

We further study the impact of this outlier source by varying the amount of *Twitter* data used during training. We gradually increase the number of *Twitter* instances by 750. As shown in Table 6, the increase of the *Twitter* data does not benefit the unified multi-source model (uni-MS-A), and even amplifies negative transfer for the *Answers* and *Reviews* domains. However, the performance of our MoE (MoE-A) model stays stable, consistently increasing with more *Twitter*, showing robustness in handling negative transfer.

6 Conclusion

In this paper, we propose a novel mixture-of-experts (MoE) approach for unsupervised domain adaptation from multiple diverse source domains. We model the domain relations through a *point-to-set* distance metric, and introduce a *meta-training* mechanism to learn this metric. Experimental results on sentiment classification and part-of-speech tagging demonstrate that our approach consistently outperforms various baselines and can robustly handle negative transfer. The effectiveness of our approach suggests its potential application to a broader range of domain adaptation tasks in NLP and other areas.

TARGET	NON-ADVERSARIAL				ADVERSARIAL			
	best-SS	uni-MS	uni-MS†	MoE	best-SS-A	uni-MS-A	uni-MS-A†	MoE-A
Answers	88.16	88.89	89.88	**90.26**	88.47	89.04	**89.99**	**89.80**
Reviews	87.15	87.45	88.91	**89.37**	87.26	87.90	88.94	**89.40**
Newsgroup	89.14	89.95	90.70	**91.03**	89.54	90.20	90.70	**91.13**
Average	*88.15*	*88.76*	*89.83*	***90.22***	*88.42*	*89.05*	*89.88*	***90.11***

Table 4: POS tagging results on SANCL data. Source domains include *Web*, *Emails*, *Twitter*. † indicates the unified multi-source model trained without *Twitter*, thus can be considered as the oracle performance (upper-bound) of uni-MS.

TARGET	SOURCE		
	Twitter	Emails	Web
Answers	0.0527	0.5941	0.3531
Reviews	0.0640	0.5250	0.4100
Newsgroup	0.0538	0.4960	0.4490

Table 5: Distribution of the metric values α on average for all tokens in the SANCL test set.

TARGET	MODEL (∗-**A**)	# Twitter instances		
		750	1,500	2,250
Answers	uni-MS	89.04	89.04	86.93
	MoE	89.80	91.22	90.90
Reviews	uni-MS	87.90	87.45	87.68
	MoE	89.40	90.23	91.14
Newsgroup	uni-MS	90.20	90.10	90.21
	MoE	91.13	91.32	91.82

Table 6: POS tagging accuracy with varying amounts of *Twitter* data in training.

Acknowledgments

We thank MIT NLP group and the anonymous reviewers for their helpful comments. We also thank Shiyu Chang and Mo Yu for insightful discussions on metric learning. This work is supported by the MIT-IBM Watson AI Lab. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*. pages 137–144.

Himanshu Sharad Bhatt, Manjira Sinha, and Shourya Roy. 2016. Cross-domain text classification with multiple domains and disparate label sets. In *Proceedings of the 54th ACL*. pages 1641–1650.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th ACL*. Prague, Czech Republic, pages 440–447.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*. pages 120–128.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*. pages 343–351.

Rich Caruana. 1997. Multitask learning. *Machine learning* 28(1):41–75.

Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*. Omnipress, USA, pages 1627–1634.

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. *arXiv preprint arXiv:1802.05694* .

Koby Crammer, Michael Kearns, and Jennifer Wortman. 2008. Learning from multiple sources. *Journal of Machine Learning Research* 9(Aug):1757–1774.

Lixin Duan, Ivor W Tsang, Dong Xu, and Tat-Seng Chua. 2009. Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings of the 26th International Conference on Machine Learning*. ACM, pages 289–296.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky.

2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research* 17(1):2096–2030.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning*. pages 513–520.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *Journal of Machine Learning Research* 13(Mar):723–773.

Robert A Jacobs, Michael I Jordan, and Andrew G Barto. 1991a. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive science* 15(2):219–250.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991b. Adaptive mixtures of local experts. *Neural computation* 3(1):79–87.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*. Doha, Qatar, pages 1746–1751.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Domain attention with an ensemble of experts. In *Proceedings of the 55th ACL*. volume 1, pages 643–653.

Shoushan Li and Chengqing Zong. 2008. Multi-domain sentiment classification. In *Proceedings of the 46th ACL*. pages 257–260.

Yijia Liu, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider, and Noah A Smith. 2018. Parsing tweets into universal dependencies. In *NAACL*.

Ping Luo, Fuzhen Zhuang, Hui Xiong, Yuhong Xiong, and Qing He. 2008. Transfer learning from multiple source domains via consensus regularization. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, pages 103–112.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation with multiple sources. In *Advances in neural information processing systems*. pages 1041–1048.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*. ACM, pages 751–760.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. Istanbul, Turkey, pages 2089–2096.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the first workshop on syntactic analysis of non-canonical language (sancl)*. volume 59.

Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. 2005. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*. volume 898, pages 1–4.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* .

Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2018. Wasserstein distance guided representation learning for domain adaptation. In *AAAI*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*. pages 4080–4090.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474* .

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*. pages 3630–3638.

Kilian Q Weinberger and Lawrence K Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research* 10(Feb):207–244.

Jun Yang, Rong Yan, and Alexander G Hauptmann. 2007. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*. ACM, pages 188–197.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Potdar Saloni, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *NAACL*.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics* 5:515–528.

Han Zhao, Shanghang Zhang, Guanhang Wu, Geoffrey J Gordon, et al. 2018. Multiple source domain adaptation with adversarial learning. *ICLR Workshop* .

Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada, pages 400–410.

A Neural Model of Adaptation in Reading

Marten van Schijndel
Department of Cognitive Science
Johns Hopkins University
`vansky@jhu.edu`

Tal Linzen
Department of Cognitive Science
Johns Hopkins University
`tal.linzen@jhu.edu`

Abstract

It has been argued that humans rapidly adapt their lexical and syntactic expectations to match the statistics of the current linguistic context. We provide further support to this claim by showing that the addition of a simple adaptation mechanism to a neural language model improves our predictions of human reading times compared to a non-adaptive model. We analyze the performance of the model on controlled materials from psycholinguistic experiments and show that it adapts not only to lexical items but also to abstract syntactic structures.

1 Introduction

Reading involves the integration of noisy perceptual evidence with probabilistic expectations about the likely contents of the text. Words that are consistent with these expectations are identified more quickly (Ehrlich and Rayner, 1981; Smith and Levy, 2013). For the reader's expectations to be maximally effective, they should not only reflect the reader's past experience with the language (Hale, 2001; MacDonald and Christiansen, 2002), but should also be *adapted* to the current context. Optimal adaptation would reflect properties of the text being read, such as genre, topic and writer identity, as well as the general tendency for recently used words and syntactic structures to be reused with higher probability (Bock, 1986; Church, 2000; Dubey et al., 2006).

Several studies have suggested that readers do in fact adapt their lexical and syntactic predictions to the current context (Otten and Van Berkum, 2008; Fine et al., 2013; Fine and Jaeger, 2016).[1] For example, Fine and Jaeger investigated the processing of "garden path" sentences such as (1):

(1) The experienced soldiers warned about the dangers conducted the midnight raid.

The word *warned* in (1) is initially ambiguous between a main verb interpretation (the soldiers were doing the warning) and a reduced relative clause interpretation (the soldiers were being warned). When the word *conducted* is reached, this ambiguity is resolved in favor of the reduced relative parse. Reduced relatives are infrequent constructions. This makes the disambiguating word *conducted* unexpected, causing it to be read more slowly than it would be in a context such as (2), in which the words *who were* indicate early on that only the relative clause parse is possible:

(2) The experienced soldiers who were warned about the dangers conducted the midnight raid.

Fine and Jaeger included a large proportion of reduced relatives in their experiment. As the experiment progressed, the cost of disambiguation in favor of the reduced relative interpretation decreased, suggesting that readers had come to expect a construction that is normally infrequent.

Human syntactic expectations have been successfully modeled with syntax-based language models (Hale, 2001; Levy, 2008; Roark et al., 2009). Recently, language models (LMs) based on recurrent neural networks (RNNs) have been shown to make adequate syntactic predictions (Linzen et al., 2016; Gulordava et al., 2018), and to make comparable reading time predictions to syntax-based LMs (van Schijndel and Linzen, 2018). In this paper, we propose a simple way to continuously adapt a neural LM, and test the method's psycholinguistic plausibility. We show that LM adaptation significantly improves our ability to predict human reading times using the LM. Follow-up experiments with controlled materials show that the LM adapts not only to specific

---

[1] Recently, Harrington Stack et al. (2018) questioned the robustness of the results of Fine et al. (2013).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4704–4710
Brussels, Belgium, October 31 - November 4, 2018.

vocabulary items but also to abstract syntactic constructions, as humans do.

2 Method

We use a simple method to adapt our LM: at the end of each new test sentence, we update the parameters of the LM based on its cross-entropy loss when predicting that sentence; the new weights are then used to predict the next test sentence.[2] Our baseline LM is a long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997) language model trained on 90 million words of English Wikipedia by Gulordava et al. (2018) (see Supplementary Materials for details). For adaptation, we keep the learning rate of 20 used by Gulordava et al. (the gradient is multiplied by this learning rate during weight updates). We examine the effect of this parameter in Section 5.2.

We tested the model on the Natural Stories Corpus (Futrell et al., 2018), which has 10 narratives with self-paced reading times from 181 native English speakers. There are two narrative genres in the corpus: fairy tales (seven texts) and documentary accounts (three texts).

3 Linguistic accuracy

We first measured how well the adaptive model predicted upcoming words. We report the model's perplexity, a quantity which is lower when the LM assigns higher probabilities to the words that in fact occurred. We adapted the model to the first k sentences of each text, then tested it on sentence $k+1$, for all k. Adaptation dramatically improved test perplexity compared to the non-adaptive version of the model (86.99 vs. 141.49).

We next adapted the model to each genre separately. If the model adapts to stylistic or syntactic patterns, we might expect adaptation to be more helpful in the fairy tale than the documentary genre: the Wikipedia corpus that the LM was originally trained on is likely to be more similar in style to the documentary genre. Consistent with this hypothesis, the documentary texts benefited less from adaptation (99.33 to 73.20) than the fairy tales (160.05 to 86.47), though the fact that both saw improvement from adaptation suggests that text-specific adaptation is beneficial even if the genre is similar to the training genre.

	$\hat{\beta}$	$\hat{\sigma}$	t
WITHOUT ADAPTIVE SURPRISAL:			
Sentence position	0.55	0.53	1.03
Word length	7.29	1.00	7.26
Non-adaptive surprisal	6.64	0.68	9.79
WITH ADAPTIVE SURPRISAL:			
Sentence position	0.29	0.53	0.55
Word length	6.42	1.00	6.40
Non-adaptive surprisal	-0.89	0.68	-1.31
Adaptive surprisal	8.45	0.63	13.42

Table 1: Fixed effect regression coefficients from fitting self-paced reading times. The top model lacks fixed and random effects of adaptive surprisal. In general, a coefficient is significant when $|t| > 2$.

Each genre consists of multiple texts. Does adaptation to a particular text lead to catastrophic forgetting (McCloskey and Cohen, 1989), such that the LM overfits to the text and forgets its more general knowledge acquired from the Wikipedia training corpus? This was not the case; in fact, adapting to the entirety of each genre without reverting to the baseline model after each text led to a very slightly *better* perplexity (fairytales: 86.47, documentaries: 73.20) compared with a setting in which the LM was reverted after each text (fairytales: 86.61, documentaries: 73.63).

4 Modeling human expectations

We next tested whether our adaptive LM matches human expectations better than a non-adaptive model. Since each reader saw the texts in a different order, we adapted the LM to each text separately: after each story, we reverted to the initial Wikipedia-trained LM and restarted adaptation on the next text. If anything, this likely resulted in a conservative estimate of the benefit of adaptation compared to a model that adapts continuously across multiple stories from the same genre, as humans might do.[3]

We used surprisal as a linking function between the LM's predictions and human reading times

---

[2] Our code is publicly available at: `https://github.com/vansky/neural-complexity.git`

[3] We do not distinguish between *priming* and *adaptation* in this paper. While it may be tempting to think of the LSTM memory cell as a model of priming and of the weight updates as a model of adaptation, Bock and Griffin (2000) provide evidence that priming cannot simply be a function of residual activation and that priming can be driven by longer-term learning (see Tooley and Traxler (2010) for more discussion on priming vs. adaptation).

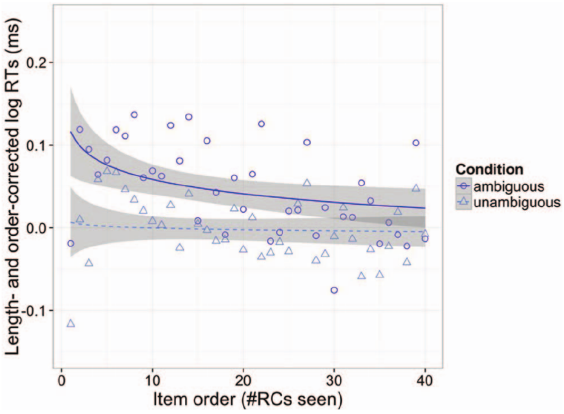

Figure 1: Mean length- and order-corrected reading times over the disambiguating region of the critical items in Fine and Jaeger (2016). Figure adopted from that paper.

(Hale, 2001; Smith and Levy, 2013). Surprisal quantifies how unpredictable each word (w_i) is given the preceding words:

$$\text{surprisal}(w_i) = -\log \text{P}(w_i \mid w_1 ... w_{i-1}) \quad (1)$$

We fit the self-paced reading times in the Natural Stories Corpus with linear mixed effects models (LMEMs), a generalization of linear regression (see Supplementary Materials for details).

In line with previous work, non-adaptive surprisal was a significant predictor of reading times ($p < 0.001$) when the model only included other baseline factors (Table 1, Top). Adaptive surprisal was a significant predictor of reading times ($p < 0.001$) over non-adaptive surprisal and all baseline factors (Table 1, Bottom). Crucially, non-adaptive surprisal was no longer a significant predictor of reading times once adaptive surprisal was included. This indicates that the predictions of the adaptive model subsume the predictions of the non-adaptive one.

5 Does the model adapt to syntax?

We have shown that LM adaptation improves our ability to model human expectations as reflected in a self-paced reading time corpus. How much of this improvement is due to adaptation of the model's syntactic representations (Bacchiani et al., 2006; Dubey et al., 2006) and how much is simply due to the model assigning a higher probability to words that have recently occurred (Kuhn and de Mori, 1990; Church, 2000)? We address this question using two syntactic phenomena: reduced relative clauses and the dative alternation.

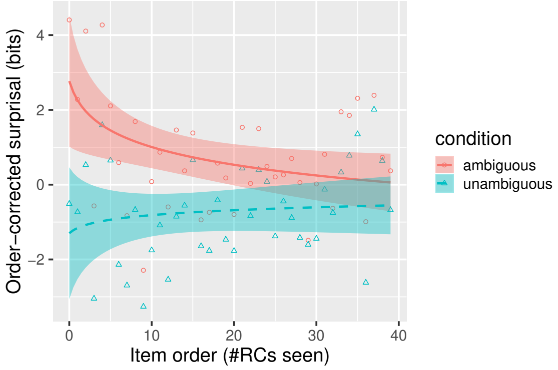

Figure 2: Mean order-corrected model surprisal over the disambiguating region of the critical items in Fine and Jaeger (2016).

5.1 Reduced relative clauses

We adapted the model independently to random orderings of the critical and filler stimuli used in Experiment 3 of Fine and Jaeger (2016);[4] this experiment (described in the Introduction) contained a much higher proportion of reduced relative clauses than their general distribution in English. We used surprisal as our proxy for reading times. Following Fine and Jaeger, we took the mean surprisal over three words in each ambiguous sentence: the disambiguating word and the following two words (e.g., *conducted the midnight* in example (1)). To estimate the magnitude of the syntactic disambiguation penalty while also controlling for lexical content, we subtracted this quantity from the mean surprisal over the exact same words in the paired unambiguous sentence (2). Linear regression showed that the disambiguation penalty decreased as the model was exposed to more critical items (item order coefficient: $\hat{\beta} = -0.0804$, $p < 0.001$), indicating that the LM was adapting to reduced relatives, a syntactic construction without any lexical content.

In order to compare our findings more directly with the results given by Fine and Jaeger (2016) (shown in Figure 1), we mimicked their method of plotting reading times. First, we fit a linear model of the mean surprisal of each disambiguating region with the number of trials the model had seen in the experiment thus far to account for a general trend of subjects speeding up over the course

---

[4] See details in the Supplementary Materials.

of the experiment. Then, we plotted the mean residual model surprisal that was left in the disambiguating region in both the ambiguous and unambiguous conditions as the experiment progressed. The shape of our model's adaptation to the reduced relative construction (upper curve in Figure 2) matched the human results reported by Fine and Jaeger. Like humans, the model showed an initially large adaptation effect, followed by more gradual adaptation thereafter. Both humans and our model continued to adapt over all the items rather than just at the beginning of the experiment. Also like humans, the model's response to unambiguous items did not change significantly over the course of the experiment ($p = 0.91$).

5.2 The dative alternation

Dative events can be expressed using two roughly equivalent English constructions:

(3) a. *Prepositional object (PO)*:
The boy threw the ball to the dog.
b. *Double object (DO)*:
The boy threw the dog the ball.

Work in psycholinguistics has shown that recent experience with one of these variants increases the probability of producing that variant (Bock, 1986; Kaschak et al., 2006) as well as the likelihood of predicting it in reading (Tooley and Bock, 2014). To test whether our adaptation method can reproduce this behavior, we generated 200 pairs of dative sentences similar to (3). We shuffled 100 DO sentences into 1000 filler sentences sampled from the Wikitext-2 training corpus (Merity et al., 2016) and adapted the model to these 1100 sentences. We then froze the weights of the adapted model and tested its predictions for two types of sentences: the PO counterparts of the DO sentences in the adaptation set, which shared the vocabulary of the adaptation set but differed in syntax; and 100 new DO sentences, which shared syntax but no content words with the adaptation set.[5]

An additional goal of this experiment was to examine the effect of learning rate on adaptation. During adaptation the model performs a single parameter update after each sentence and does not train until convergence with gradual reduction of the learning rate as would normally be the case during LM training. Consequently, the learning rate parameter crucially determines the amount of adaptation the model can undertake after each sentence. If the learning rate is very low, adaptation will not have any effect; if it is too high, either the model will overfit after each update and will not generalize well, or the model will forget its trained representation as it overshoots the targeted minima. The optimal rate may differ between lexical and syntactic adaptation. Our experiments thus far all used the same learning rate as our original model (20); here, we varied the learning rate on a logarithmic scale between 0.002 and 200.

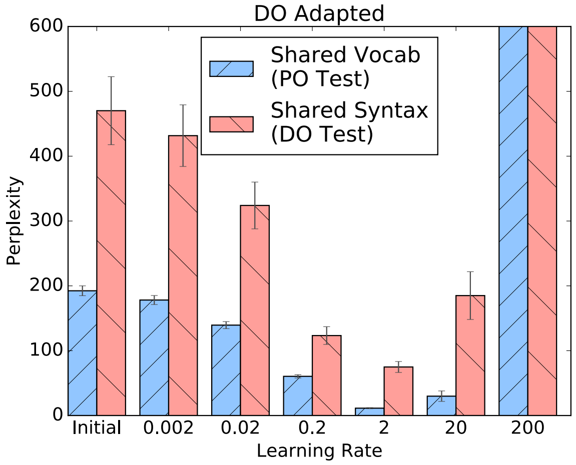

Figure 3: Learning rate influence over syntactic and lexical adaptation. The initial non-adaptive model performance is equivalent to the performance when using a learning rate of 0; the learning rate of 200 resulted in perplexity in the billions.

The results of this experiment are shown in Figure 3. The model successfully adapted to the DO construction as well as to the vocabulary of the adaptation sentences. This was the case for all of the learning rates except for 200, which resulted in enormous perplexity on both sentence types. Both lexical and syntactic adaptation were most successful when the learning rate was around 2, with perplexity reductions of 94% for lexical adaptation and 84% for syntactic adaptation.

Syntactic adaption was penalized at higher learning rates more than lexical adaptation (compare learning rates of 2 and 20). This fragility of syntactic adaptation likely stems from the fact that the model can directly observe the relevant vocabulary but syntax is latent and must be inferred from multiple similar sentences, a generalization which is impeded by overfitting at higher learning rates.

---

[5] For additional details as well as the reverse setting (adaptation to PO), see Supplementary Materials.

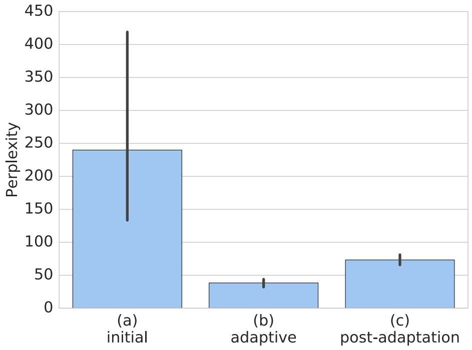

Figure 4: Perplexity on the held-out set of G_1 (a) before adaptation, (b) after adaptation to G_1, (c) after adapting to G_1 then adapting to G_2.

6 Testing for catastrophic forgetting

Our analysis of the Natural Stories corpus did not indicate that the model suffered from catastrophic forgetting. Yet the Natural Stories corpus contained only two genres; to address the issue of catastrophic forgetting more systematically, we used the premise sentences from the MultiNLI corpus (Williams et al., 2018) — a total of 2000 sentences for each of 10 genres.

For each genre pair G_1 and G_2 (omitting cases where $G_1 = G_2$), we first adapted the baseline Wikipedia model to 1000 sentences of G_1 using a learning rate of 2 (shown to be optimal in Section 5.2). We then adapted the model to 1000 sentences of G_2. Finally, we froze the model's weights and tested its perplexity on the 1000 held-out sentences from G_1.

The results averaged across all pairs of genres are plotted in Figure 4. Unsurprisingly, the model performed best on G_1 immediately after adapting to it (middle bar). Crucially, even after adapting to 1000 sentences of G_2 after its last exposure to G_1 (right bar), it still modeled G_1 much better than the non-adapted model (left bar). These results suggest that catastrophic forgetting is not a concern even with a relatively large amount of data.

7 Discussion

Adaptation greatly improved an RNN LM's word prediction accuracy, in line with other work on LM adaptation (Kneser and Steinbiss, 1993). We showed that the adapted model was psycholinguistically plausible, in two senses. First, it improved the correlation between surprisal derived from the model and human reading times, suggesting that the model generated more human-like expectations. Second, using materials that teased apart lexical content from syntax, we showed that the model adapted both its lexical and its syntactic predictions, in line with findings from human experiments. Finally, as in other neural-network based models in psychology (Chang et al., 2006), our gradient-based updates naturally incorporate the error-driven nature of syntactic adaptation; while we did not demonstrate this in the current paper, we hypothesize that our model will reproduce the finding that more surprising words lead to greater adaptation (Jaeger and Snider, 2013).

The simplicity of our adaptation method makes it attractive for use in modeling human expectations. Since adaptive surprisal is strictly superior to non-adaptive surprisal in modeling reading times, it would be a stronger baseline in analyses that aim to demonstrate the contribution of factors other than predictability.

We used a simple neural adaptation approach, where we performed continuous gradient updates based on the prediction error on the adaptation sentences (see also Krause et al., 2017). An alternative approach to neural LM adaptation uses recent RNN states in conjunction with the current state to make word predictions (Grave et al., 2017; Merity et al., 2017); a comparison of the two methods using our paradigms may provide insight into their relative strengths and weaknesses.

Finally, we reverted to the base model after the end of each text in our experiments, forgetting any text-specific adaptation. This mimics the effect of a participant leaving an experiment that had an unusual distribution of syntactic constructions and reverting to their standard expectations. In practice, however, humans are able to generalize from prior experience when they begin adapting to a new speaker or text if it is similar in some way to their previous experiences. For example, the model of Jaech and Ostendorf (2018) adapts to environmental factors, so it could potentially draw on independent experiences with female speakers and with lawyer speech in order to initialize a model of adaptation to a new female lawyer (see also Mikolov and Zweig, 2012; Kleinschmidt, 2018). The psycholinguistic plausibility of these models can be tested in future work.

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech & Language*, 20(1):41–68.

Kathryn Bock. 1986. Syntactic persistence in language production. *Cognitive Psychology*, 18(3):355–387.

Kathryn Bock and Zenzi M. Griffin. 2000. The persistence of structural priming: transient activation or implicit learning? *Journal of Experimental Psychology: General*, 129(2):177–192.

Franklin Chang, Gary S Dell, and Kathryn Bock. 2006. Becoming syntactic. *Psychological Review*, 113(2):234–272.

Kenneth W. Church. 2000. Empirical estimates of adaptation: the chance of two noriegas is closer to $p/2$ than p^2. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*, pages 180–186. Association for Computational Linguistics.

Amit Dubey, Frank Keller, and Patrick Sturt. 2006. Integrating syntactic priming into an incremental probabilistic parser, with an application to psycholinguistic modeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 417–424. Association for Computational Linguistics.

Susan F. Ehrlich and Keith Rayner. 1981. Contextual effects on word perception and eye movements during reading. *Journal of Verbal Learning and Verbal Behavior*, 20(6):641–655.

Alex B. Fine and T. Florian Jaeger. 2016. The role of verb repetition in cumulative structural priming in comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 42(9):1362–1376.

Alex B. Fine, T. Florian Jaeger, Thomas A. Farmer, and Ting Qian. 2013. Rapid expectation adaptation during syntactic comprehension. *PloS ONE*, 8(10):1–18.

Richard Futrell, Edward Gibson, Hal Tily, Anastasia Vishnevetsky, Steve Piantadosi, and Evelina Fedorenko. 2018. The natural stories corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 76–82.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In *Proceedings of the International Conference on Learning Representations (ICLR 2017)*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205. Association for Computational Linguistics.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Human Language Technologies*, pages 1–8, Pittsburgh, PA. Association for Computational Linguistics.

Caoimhe M. Harrington Stack, Ariel N. James, and Duane G. Watson. 2018. A failure to replicate rapid syntactic adaptation in comprehension. *Memory & Cognition*.

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

Aaron Jaech and Mari Ostendorf. 2018. Personalized language model for query auto-completion. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 700–705. Association for Computational Linguistics.

T. Florian Jaeger and Neal E. Snider. 2013. Alignment as a consequence of expectation adaptation: Syntactic priming is affected by the prime's prediction error given both prior and recent experience. *Cognition*, 127:57–83.

Michael P. Kaschak, Renrick A. Loney, and Kristin L Borreggine. 2006. Recent experience affects the strength of structural priming. *Cognition*, 99(3):B73–B82.

Dave F. Kleinschmidt. 2018. Structure in talker variability: How much is there and how much can it help? *Language, Cognition and Neuroscience*.

Reinhard Kneser and Volker Steinbiss. 1993. On the dynamic adaptation of stochastic language models. In *Proceedings of ICASSP-93*.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2017. Dynamic evaluation of neural sequence models. *arXiv preprint arXiv:1709.07432*.

Roland Kuhn and Renato de Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Maryellen C. MacDonald and Morten H. Christiansen. 2002. Reassessing working memory: Comment on Just and Carpenter (1992) and Waters and Caplan (1996). *Psychological Review*, 109(1):35–54.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower, editor, *The Psychology of Learning and Motivation: Volume 24*, 92, pages 109–165. San Diego: Academic Press.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Wikitext-2.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of the International Conference on Learning Representations (ICLR 2017)*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. *SLT*, 12:234–239.

Marte Otten and Jos J. A. Van Berkum. 2008. Discourse-based word anticipation during language processing: Prediction or priming? *Discourse Processes*, 45(6):464–496.

Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 324–333. Association for Computational Linguistics.

Nathaniel J. Smith and Roger Levy. 2013. The effect of word predictability on reading time is logarithmic. *Cognition*, 128:302–319.

Kristen M. Tooley and Kathryn Bock. 2014. On the parity of structural persistence in language production and comprehension. *Cognition*, 132(2):101–136.

Kristen M. Tooley and Matthew J. Traxler. 2010. Syntactic priming effects in comprehension: A critical review. *Language and Linguistics Compass*, 4(10):925–937.

Marten van Schijndel and Tal Linzen. 2018. Modeling garden path effects without explicit hierarchical syntax. In Tim Rogers, Marina Rau, Jerry Zhu, and Chuck Kalish, editors, *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 2600–2605. Cognitive Science Society, Austin, TX.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Understanding Deep Learning Performance through an Examination of Test Set Difficulty: A Psychometric Case Study

John P. Lalor[1*], **Hao Wu**[2], **Tsendsuren Munkhdalai**[3], **Hong Yu**[1,4]
[1]College of Information and Computer Sciences, University of Massachusetts, Amherst
[2] Department of Psychology and Human Development, Vanderbilt University
[3]Microsoft Research, Montréal, Québec
[4]Department of Computer Science, University of Massachusetts, Lowell
`*lalor@cs.umass.edu`

Abstract

Interpreting the performance of deep learning models beyond test set accuracy is challenging. Characteristics of individual data points are often not considered during evaluation, and each data point is treated equally. We examine the impact of a test set question's difficulty to determine if there is a relationship between difficulty and performance. We model difficulty using well-studied psychometric methods on human response patterns. Experiments on Natural Language Inference (NLI) and Sentiment Analysis (SA) show that the likelihood of answering a question correctly is impacted by the question's difficulty. As DNNs are trained with more data, easy examples are learned more quickly than hard examples.

1 Introduction

One method for interpreting deep neural networks (DNNs) is to examine model predictions for specific input examples, e.g. testing for shape bias as in Ritter et al. (2017). In the traditional classification task, the difficulty of the test set examples is not taken into account. The number of correctly-labeled examples is tallied up and reported. However, we hypothesize that it may be worthwhile to use difficulty when evaluating DNNs. For example, what does it mean if a trained model answers the more difficult examples correctly, but cannot correctly classify what are seemingly simple cases? Recent work has shown that for NLP tasks such as Natural Language Inference (NLI), models can achieve strong results by simply using the hypothesis of a premise-hypothesis pair and ignoring the premise entirely (Gururangan et al., 2016; Tsuchiya, 2018; Poliak et al., 2018).

In this work we consider understanding DNNs by looking at the difficulty of specific test set examples and comparing DNN performance under different training scenarios. Do DNN models learn examples of varying difficulty at different rates? If a model does well on hard examples and poor on easy examples, then can we say that it has really learned anything? In contrast, if a model does well on easy items, because a dataset is all easy, have we really "solved" anything?

To model difficulty we use Item Response Theory (IRT) from psychometrics (Baker and Kim, 2004). IRT models characteristics such as difficulty and discrimination ability of specific examples (called "items"[1]) in order to estimate a latent ability trait of test-takers. Here we use IRT to model the difficulty of test items to determine how DNNs learn items of varying difficulty. IRT provides a well-studied methodology for modeling item difficulty as opposed to more heuristic-based difficulty estimates such as sentence length. IRT was previously used to build a new test set for the NLI task (Lalor et al., 2016) and show that model performance is dependent on test set difficulty. In this work we use IRT to probe specific items to try to analyze model performance at a more fine-grained level, and expand the analysis to include the task of SA.

We train three DNNs models with varying training set sizes to compare performance on two NLP tasks: NLI and Sentiment Analysis (SA). Our experiments show that a DNN model's likelihood of classifying an item correctly is dependent on the item's difficulty. In addition, as the models are trained with more data, the odds of answering easy examples correctly increases at a faster rate than the odds of answering a difficult example correctly. That is, performance starts to look more human, in the sense that humans learn easy items faster than they learn hard items.

That the DNNs are better at easy items than hard items seems intuitive but is a surprising and interesting result since the item difficulties are modeled *from human data*. There is no underlying reason

---

[1] For the remainder of the paper we will refer to a single test set example as an "item" for consistency.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4711–4716
Brussels, Belgium, October 31 - November 4, 2018.

that the DNNs would find items that are easy for humans inherently easy. To our knowledge this is the first work to use a grounded measure of difficulty learned from human responses to understand DNN performance. Our contributions are as follows: (i) we use a well-studied methodology, IRT, to estimate item difficulty in two NLP tasks and show that this human-estimated difficulty is a useful predictor of DNN model performance, (ii) we show that as training size increases DNN performance trends towards expected human performance.[2]

2 Methods

2.1 Estimating Item Difficulty

To model item difficulty we use the Three Parameter Logistic (3PL) model from IRT (Baker, 2001; Baker and Kim, 2004; Lalor et al., 2016). The 3PL model in IRT models an individual's latent ability (θ) on a task as a function of three item characteristics: discrimination ability (a), difficulty (b), and guessing (c). For a particular item i, the probability that an individual j will answer item i correctly is a function of the individual's ability and the three item characteristics:

$$p_{ij}(\theta_j) = c_i + \frac{1 - c_i}{1 + e^{-a_i(\theta_j - b_i)}} \tag{1}$$

where a_i is the discrimination parameter (the value of the function slope at it's steepest point), b_i is the difficulty parameter (the value where $p_{ij}(\theta_j) = 0.5$), and c_i is the guessing parameter (the lower asymptote of the function). For a set of items I and a set of individuals J, the likelihood of each individual in J's responses to the items in I is:

$$L = \prod_{j=1}^{J} \prod_{i=1}^{I} p_{ij}(\theta_j)^{y_{ij}} q_{ij}(\theta_j)^{(1-y_{ij})} \tag{2}$$

where $q_{ij}(\theta_j) = 1 - p_{ij}(\theta_j)$ and $y_{ij} = 1$ if individual j answered item i correctly and $y_{ij} = 0$ otherwise. Item parameters and individual ability are jointly estimated from a set of individuals' response patterns using an Expectation-Maximization algorithm (Bock and Aitkin, 1981).

In this work we focus on the difficulty parameter b_i, which represents the latent ability level at which an individual has a 50% chance of answering item i correctly. Low values of b_i are associated with easier items (since an individual with low ability has a 50% chance of answering correctly), and higher values of b_i represent more difficult items.

2.2 Data

To estimate item difficulties for NLI, we used the pre-trained IRT models of Lalor et al. (2016) and extracted the difficulty item parameters. The data consists of approximately 1000 human annotator responses from Amazon Mechanical Turk (AMT) for a selection of 180 premise-hypothesis pairs from the SNLI data set (Bowman et al., 2015). Each AMT worker (Turker) was shown the premise-hypothesis pairs and was asked to indicate whether, if the premise was taken to be true, the hypothesis was (a) definitely true (*entailment*), (b) maybe true (*neutral*), or (c) definitely not true (*contradiction*).

For SA, we collected a new data set of labels for 134 examples randomly selected from the Stanford Sentiment Treebank (SSTB) (Socher et al., 2013), using a similar AMT setup as Lalor et al. (2016). For each randomly selected example, we had 1000 Turkers label the sentence as very negative, negative, neutral, positive, or very positive. We converted these responses to binary positive/negative labels and fit a new IRT 3PL model (§2.1) using the *mirt* R package (Chalmers et al., 2015). Very negative and negative labels were binned together, and neutral, positive, and very positive were binned together.

Tables 1 and 2 show examples of the items in our data sets, and the difficulty values estimated from the IRT models. The first example in Table 1 is a clear case of *entailment*, where if we assume that the premise is true, we can infer that the hypothesis is also true. The label of the second example in SNLI is *contradiction*, but in this case the result is not as clear. There are sports stadiums that offer lawn seating, and therefore this could potentially be a case of entailment (or neutral). Either way, one could argue that the second example here is more difficult than the first. Similarly, the first two examples of Table 2 are interesting. Both of these items are labeled as *negative* examples in the data set. The first example is clear, but the second one is more ambiguous. It could be considered a mild complement, since the author still endorses renting the movie. Therefore you could argue again that the second example is more difficult than the first. The learned difficulty parameters reflect this difference

---

[2] Code and data available at `http://jplalor.github.io`

Premise	Hypothesis	Label	Difficulty
A little girl eating a sucker	A child eating candy	Entailment	-2.74
People were watching the tournament in the stadium	The people are sitting outside on the grass	Contradiction	0.51
Two girls on a bridge dancing with the city skyline in the background	The girls are sisters.	Neutral	-1.92
Nine men wearing tuxedos sing	Nine women wearing dresses sing	Contradiction	0.08

Table 1: Examples of sentence pairs from the SNLI data sets, their corresponding gold-standard label, and difficulty parameter (b_i) as measured by IRT (§2.1).

Phrase	Label	Difficulty
The stupidest, most insulting movie of 2002's first quarter.	Negative	-2.46
Still, it gets the job done - a sleepy afternoon rental.	Negative	1.78
An endlessly fascinating, landmark movie that is as bold as anything the cinema has seen in years.	Positive	-2.27
Perhaps no picture ever made has more literally showed that the road to hell is paved with good intentions.	Positive	2.05

Table 2: Examples of phrases from the SSTB data set, their corresponding gold-standard label, and difficulty parameter (b_i) as measured by IRT (§2.1).

Dataset	Fleiss' κ
SNLI 4GS Contradiction	0.37
SNLI 4GS Entailment	0.48
SNLI 4GS Neutral	0.41
SNLI 5GS Contradiction	0.59
SNLI 5GS Entailment	0.63
SNLI 5GS Neutral	0.54
Sentiment Analysis	0.52

Table 3: Fleiss' κ scores for the NLI and SA annotations collected from AMT.

in difficulty in both cases.

Inter-rater reliability scores for the collected annotations are showin in Table 3. Scores for the NLI annotations were calculated when the original dataset was collected and are reproduced here (Lalor et al., 2016). Human annotations for the SA annotations were converted to binary before calculating the agreement. We see that the agreement scores are in the range of 0.4 to 0.6 which is considered moderate agreement (Landis and Koch, 1977). With the large number of annotators it is to be expected that there is some disagreement in the labels. However this disagreement can be interpreted as varying difficulty of the items, which is what we expect when we fit the IRT models.

2.3 Experiments

Our goal in this work is to understand how DNN performance on items of varying difficulty changes under different training scenarios. To test this, we trained three DNN models using subsets of the original SNLI and SSTB training data sets: (i) Long Short Term Memory Network (LSTM) (Bowman et al., 2015), (ii) Convolutional Neural Network (CNN) (Kim, 2014), and (iii) Neural Semantic Encoder (NSE), a type of memory-augmented RNN (Munkhdalai and Yu, 2017).[3] For each task (NLI and SA), we randomly sampled subsets of training data, from 100 examples up to and including the full training data sets.[4] We trained each model on the training data subsets, using the original development sets for early stopping to prevent overfitting. The IRT data with difficulty estimates were used as test sets for the trained models.

Once the models were trained and had classified the IRT data sets, we fit logistic regression models to predict whether a DNN model would label an item correctly, using the training set size and item difficulty as the dependent parameters.

3 Results

Figure 1 plots the contour plots of our learned regression models. The top row plots results for the NLI task, and the bottom row plots results for the SA task. From left to right in both rows, the plots show results for the LSTM, CNN, and NSE models. In each plot, the x-axis is the training set size, the y-axis is the item difficulty, and the contour lines represent the log-odds that the DNN model would classify an item correctly. As the plots show, item difficulty has a clear effect on classification. Easier items have higher odds of being classified correctly

---

[3] Please refer to the appendix for model details.

[4] We sampled 100, 1000, 2000, 5000, 10000, 50000, 100000, 200000, and 500000 examples for NLI, and sampled 100, 1000, 5000, 10000, 50000, and 75000 examples for SA.

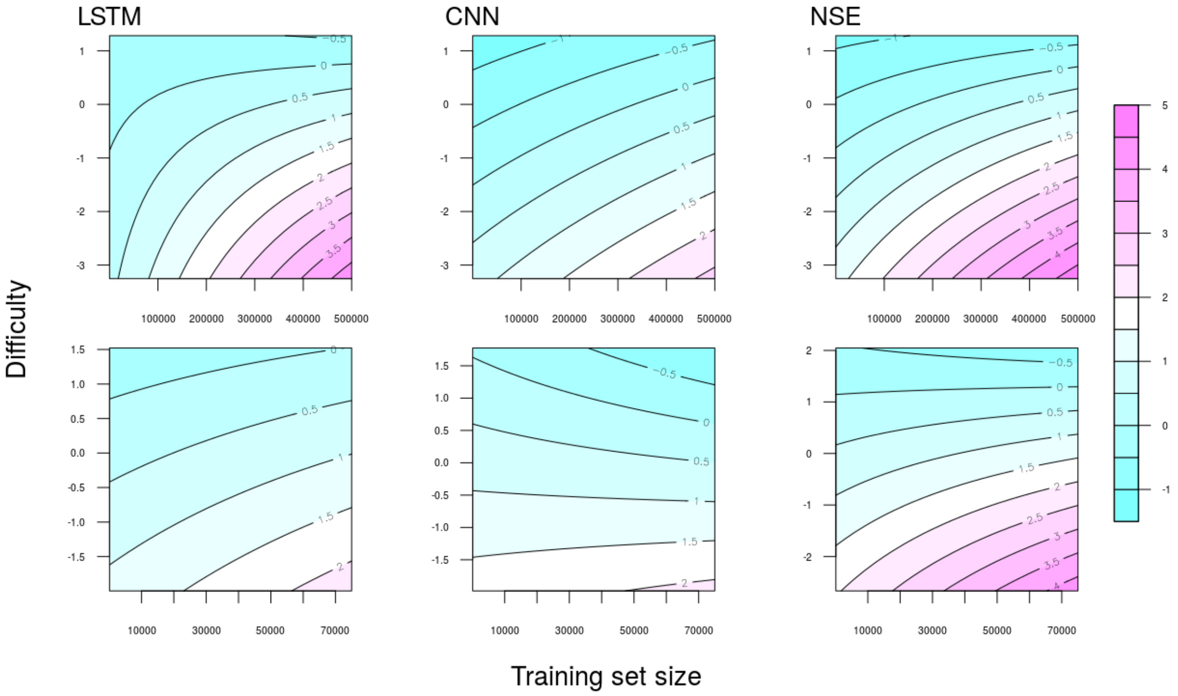

Figure 1: Contour plots showing log-odds of labeling an item correctly for NLI (top row) and SA (bottom row) as a function of training set size (x-axis) and item difficulty (y-axis). Each line in the plots represents a single log-odds value for labeling an item correctly. Blue indicates low log-odds of labeling an item correctly, and pink indicates high log-odds of labeling an item correctly. The contour colors are consistent across plots and log-odds values are shown in the legend on the right.

across all of the training set sizes. In addition, the slopes of the contour lines are steeper at lower levels of difficulty. This indicates that, moving left to right along the x-axis, a model's odds of answering an easy item correctly increase more quickly than the odds of answering a harder item correctly.

The contour plots for the CNN and NSE models on the SA task (Figure 1, second row middle and right plots) show that the easier items have higher likelihood of being classified correctly, but the odds for the most difficult items decrease as training size increases. This suggests that these models are learning in such a way that improves performance on easy items but has a negative effect on hard items. This result is important for interpretability, as it could inform stakeholder decisions if they need to have difficult examples classified.

The idea that easy items should be easier than hard items is consistent with learning strategies in humans. For example, when teaching new concepts to students, easier concepts are presented first so that the students can learn patterns and core information before moving to more difficult concepts (Collins et al., 1988; Arroyo et al., 2010). As students do more examples, all questions get easier, but easy questions get easier at a faster rate. Our result is also consistent with the key assumptions of curriculum learning (Bengio et al., 2009).

4 Related Work

Lalor et al. (2016) introduced the idea of applying IRT evaluation to NLP tasks. They built a set of scales using IRT for NLI and evaluated a single LSTM neural network to demonstrate the effectiveness of the evaluation, but did not evaluate other NLP models or tasks. Martínez-Plumed et al. (2016) consider IRT in the context of evaluating ML models, but they do not use a human population to calibrate the models, and obtain results that are difficult to interpret under IRT assumptions.

There has been work in the NLP community around modeling latent characteristics of data (Bruce and Wiebe, 1999) and annotators (Hovy et al., 2013), but none that apply the resulting metrics to interpret DNN models. Passonneau and Carpenter (2014) model the probability a label is correct with the probability of an annotator to label an item correctly according to the Dawid and Skene (1979) model, but do not consider difficulty or discriminatory ability of the data points.

One-shot learning is an attempt to build ML models that can generalize after being trained on one or a few examples of a class as opposed to a large training set (Lake et al., 2013). One-shot learning attempts to mimic human *learning behaviors* (i.e., generalization after being exposed to a small number of training examples) (Lake et al., 2016). Our work instead looks at comparisons to human *performance*, where any learning (on the part of models) has been completed beforehand. Our goal is to analyze DNN models and training set variations as they affect ability in the context of IRT.

5 Discussion

In this work we have shown that DNN model performance is affected by item difficulty as well as training set size. This is the first work that has used a well-established method for estimating difficulty to analyze DNN model performance as opposed to heuristics. DNN models perform better on easy items, and as more data is introduced in training, easy items are learned more quickly than hard items. Learning easy examples faster than harder examples is what would be expected when examining human response patterns as they learn more about a subject. However this has not previously been shown to be true in DNN models.

That the results are consistent across NLI and SA shows that the methods can be applied to a number of NLP tasks. The SA results do show that the odds of labeling a difficult item correctly decrease with more training data 1. It could be the case that these difficult items in the SA task are more subjective than the easier items, for example a review that is fairly neutral and is split between positive and negative annotations. These cases would be more difficult for a model to label, and are worth examining in more detail. By identifying items such as these as difficult makes it easier to see where the model is going wrong and allows for research on better way to represent these cases.

This result has implications for how machine learning models are evaluated across tasks. The traditional assumption that the test data is drawn from the same distribution as the training data, makes it difficult to understand how a model will perform in settings where that assumption does not hold. However, if the difficulty of test set data is known, we can better understand what kind of examples a given model performs well on, and specific instances where a model underperforms (e.g. the most difficult examples). In addition, researhers can build test sets that consist of a specific type of data (very easy, very hard, or a mix) to evaluate a trained model under specific assumptions to test generalization ability in a controlled way. This could allow for more confidence in model performance in more varied deployment settings, since there would be a set of tests a model would have to pass before being deployed.

It is important to note that the difficulty parameters were estimated *from a human population*, meaning that those items that are difficult for humans are in fact more difficult for the DNN models as well. This does not need to be the case given that DNNs learn very different patterns, etc. than humans. In fact there were exceptions in our results which shows that these models should be carefully examined using techniques like those described here. Future work can investigate why this is the case and how we can leverage this information to improve model performance and interpretability.

Acknowledgments

We thank the AMT Turkers who completed our annotation task. This work was supported in part by the HSR&D award IIR 1I01HX001457 from the United States Department of Veterans Affairs (VA). We also acknowledge the support of LM012817 from the National Institutes of Health. This work was also supported in part by the Center for Intelligent Information Retrieval. The contents of this paper do not represent the views of CIIR, NIH, VA, or the United States Government.

References

Ivon Arroyo, Hasmik Mehranian, and Beverly P Woolf. 2010. Effort-based tutoring: An empirical approach to intelligent tutoring. In *Educational Data Mining 2010*.

Frank B Baker. 2001. *The basics of item response theory*. ERIC.

Frank B. Baker and Seock-Ho Kim. 2004. *Item Response Theory: Parameter Estimation Techniques, Second Edition*. CRC Press.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

R Darrell Bock and Murray Aitkin. 1981. Marginal maximum likelihood estimation of item parameters:

Application of an em algorithm. *Psychometrika*, 46(4):443–459.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and D. Christopher Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Rebecca F. Bruce and Janyce M. Wiebe. 1999. Recognizing subjectivity: A case study in manual tagging. *Nat. Lang. Eng.*, 5(2):187–205.

Phil Chalmers, Joshua Pritikin, Alexander Robitzsch, and Mateusz Zoltak. 2015. mirt: Multidimensional Item Response Theory.

Allan Collins, John Seely Brown, and Susan E Newman. 1988. Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. *Thinking: The Journal of Philosophy for Children*, 8(1):2–10.

A. Philip Dawid and Allan M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2016. Annotation artifacts in natural language inference datg. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, Louisiana. Association for Computational Linguistics.

S Hochreiter and J Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Brenden M Lake, Ruslan R Salakhutdinov, and Josh Tenenbaum. 2013. One-shot learning by inverting a compositional causal process. In *Advances in neural information processing systems*, pages 2526–2534.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2016. Building machines that learn and think like people. *Behavioral and Brain Sciences*, pages 1–101.

John P. Lalor, Hao Wu, and Hong Yu. 2016. Building an evaluation scale using item response theory. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 648–657. Association for Computational Linguistics.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Fernando Martínez-Plumed, Ricardo B. C. Prudłncio, Adolfo Martnez Us, and Jos Hernndez-Orallo. 2016. Making sense of item response theory in machine learning. In *ECAI*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 1140–1148. IOS Press.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural semantic encoders. *EACL 2017*.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association of Computational Linguistics*, 2:311–326.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the 7th Joint Conference on Lexical and Computational Semantics (* SEM 2018)*.

Samuel Ritter, David GT Barrett, Adam Santoro, and Matt M Botvinick. 2017. Cognitive psychology for deep neural networks: A shape bias case study. In *ICML*.

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

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Lexicosyntactic Inference in Neural Models

Aaron Steven White
University of Rochester

Rachel Rudinger
Johns Hopkins University

Kyle Rawlins
Johns Hopkins University

Benjamin Van Durme
Johns Hopkins University

Abstract

We investigate neural models' ability to capture *lexicosyntactic inferences*: inferences triggered by the interaction of lexical and syntactic information. We take the task of event factuality prediction as a case study and build a factuality judgment dataset for all English clause-embedding verbs in various syntactic contexts. We use this dataset, which we make publicly available, to probe the behavior of current state-of-the-art neural systems, showing that these systems make certain systematic errors that are clearly visible through the lens of factuality prediction.

1 Introduction

The formal semantics literature has long been concerned with the complex array of inferences that different open class lexical items trigger (Kiparsky and Kiparsky, 1970; Karttunen, 1971a,b; Horn, 1972; Karttunen and Peters, 1979; Heim, 1992; Simons, 2001, 2007; Simons et al., 2010; Abusch, 2002, 2010; Gajewski, 2007; Anand and Hacquard, 2013, 2014). For example, why does (1a) give rise to the inference (2a), while the structurally identical (1b) triggers the inference (2b)?

(1) a. Jo doesn't believe that Bo left.
b. Jo doesn't know that Bo left.

(2) a. Jo believes that Bo didn't leave.
b. Bo left.
c. Bo didn't leave.

A major finding of this literature is that lexically triggered inferences are conditioned by surprising aspects of the syntactic context that a word occurs in. For example, while (3a), (3b), and (4a) trigger the inference (2b), (4b) triggers the inference (2c).

(3) a. Jo remembered that Bo left.
b. Jo didn't remember that Bo left.

(4) a. Bo remembered to leave.
b. Bo didn't remember to leave.

Accurately capturing such interactions – e.g. between clause-embedding verbs, negation, and embedded clause type – is important for any system that aims to do general natural language inference (MacCartney et al. 2008 *et seq*; cf. Dagan et al. 2006) or event extraction (see Grishman and Sundheim 1996 *et seq*), and it seems unlikely to be a trivial phenomenon to capture, given the complexity and variability of the inferences involved (see, e.g., Karttunen, 2012, 2013; Karttunen et al., 2014; van Leusen, 2012; White, 2014; Baglini and Francez, 2016; Nadathur, 2016, on implicatives).

In this paper, we investigate how well current state-of-the-art neural systems for a subtask of general event extraction – event factuality prediction (EFP; Nairn et al., 2006; Saurí and Pustejovsky, 2009, 2012; de Marneffe et al., 2012; Lee et al., 2015; Stanovsky et al., 2017; Rudinger et al., 2018) – capture inferential interactions between lexical items and syntactic context – *lexicosyntactic inferences* – when trained on current event factuality datasets. Probing these particular systems is useful for understanding neural systems' behavior more generally because (i) the best performing neural models for EFP (Rudinger et al., 2018) are simple instances of common baseline models; and (ii) the task itself is relatively constrained.

To do this, we substantially extend the MegaVeridicality1 dataset (White and Rawlins, 2018) to cover all English clause-embedding verbs in a variety of the syntactic contexts covered by recent psycholinguistic work (White and Rawlins, 2016), and we use the resulting dataset – MegaVeridicality2 – to probe these models' behavior. We focus on clause-embedding verbs because they show effectively every possible patterning of lexicosyntactic inference (Karttunen, 2012).

We discuss three findings: (i) Tree biLSTMs (T-biLSTMs) are better able to correctly predict lexicosyntactic inferences than linear-chain biLSTMs

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4717–4724
Brussels, Belgium, October 31 - November 4, 2018.

(L-biLSTMs); (ii) L-biLSTMs and T-biLSTMs capture different lexicosyntactic inferences, and thus ensembling their predictions can reliably improve performance; and (iii) even when ensembled, these models show systematic errors – e.g. performing well when the polarity of the matrix clause matches the polarity of the true inference, but poorly when these polarities mismatch.

We furthermore release MegaVeridicality2 at MegaAttitude.io as a benchmark for probing the ability of neural systems – whether for factuality prediction or for general natural language inference – to capture lexicosyntactic inference.

2 Data collection

We substantially extend the MegaVeridicality1 dataset (White and Rawlins, 2018), which contains factuality judgments for all English clause-embedding verbs that take tensed subordinate clauses. In White and Rawlins's annotation protocol, all verbs that are grammatical with such subordinate clauses – based on the MegaAttitude dataset (White and Rawlins, 2016) – are slotted into contexts either like (5a) or (5b), depending on whether they take a direct object or not.

(5) a. Someone {knew, didn't know} that a particular thing happened.
b. Someone {was, wasn't} told that a particular thing happened.

For each sentence generated in this way, 10 different annotators are asked to answer the question *did that thing happen?*: *yes*, *maybe or maybe not*, *no*.

There are two important aspects of these contexts to note. First, all lexical items besides the embedding verbs are semantically bleached to ensure that the measured lexicosyntactic inferences are only due to interactions between the embedding predicate – e.g. *know* or *tell* – and the syntactic context. Second, the matrix polarity – i.e. the presence or absence of *not* as a direct dependent of the embedding verb – is manipulated to create two sentences for each verb-context pair.

Our extension, MegaVeridicality2, includes judgments for a variety of infinitival subordinate clause types, exemplified in (6).[1] We investigate infinitival clauses because they can give rise to different lexicosyntactic inferences than finite subordinate clauses – e.g. compare (3) and (4).

Syntactic context	**# verbs**	**# sents**	**Ex.**
NP _ed that S	375	750	(5a)
NP was _ed that S	169	338	(5b)
NP _ed for NP to VP	184	368	(6a)
NP _ed NP to VP[+ev]	197	394	(6b)
NP _ed NP to VP[-ev]	128	256	(6c)
NP was _ed to VP[+ev]	278	556	(6d)
NP was _ed to VP[-ev]	256	512	(6e)
NP _ed to VP[+ev]	217	434	(6f)
NP _ed to VP[-ev]	165	330	(6g)
Total	1,969	3,938	

Table 1: Contexts and number of verbs for which annotations were collected: S = *something happened*, NP = *someone*, VP = *happen*, VP[+ev] = *do something*, VP[-ev] = *have something*. First two rows: MegaVeridicality1. All rows: MegaVeridicality2. The number of sentences is always twice the number of verbs, since matrix polarity is manipulated.

(6) a. Someone {needed, didn't need} for a particular thing to happen.
b. Someone {wanted, didn't want} a particular person to do, have a particular thing.
c. Someone {wanted, didn't want} a particular person to have a particular thing.
d. A particular person {was, wasn't} overjoyed to do a particular thing.
e. A particular person {was, wasn't} overjoyed to have a particular thing.
f. A particular person {managed, didn't manage} to do a particular thing.
g. A particular person {managed, didn't manage} to have a particular thing.

For each sentence, we also collect judgments from 10 different annotators, using the same question as White and Rawlins for context (6a) and modified questions for contexts (6b)-(6g): *did that person do that thing?* for (6b), (6d), and (6f); and *did that person have that thing?* for for (6c), (6e), and (6g). Table 1 shows the number of verb types for each syntactic context. With the polarity manipulation, this yields a total of 3,938 sentences.

To build a factuality prediction test set from these sentences, we combine MegaVeridicality1 with our dataset and replace each instance of *a particular person* or *a particular thing* with *someone* or *something* (respectively). Then, following White and Rawlins, we normalize the 10 responses for each sentence to a single real value using an ordinal mixed model-based procedure. We refer to the resulting dataset as MegaVeridicality2.

3 Model and evaluation

We use MegaVeridicality2 to evaluate the performance of three state-of-the-art neural models of

---

[1] We also explicitly manipulate two aspects of the subordinate clause in our extension of the MegaVeridicality dataset: (i) how NP embedded subjects are introduced; and (ii) whether the embedded clause contains an eventive predicate (*do*, *happen*) or a stative predicate (*have*). See Appendix A for details on the reasoning behind these manipulations.

event factuality (Rudinger et al., 2018): a linear-chain biLSTM (L-biLSTM), a dependency tree biLSTM (T-biLSTM), and a hybrid biLSTM (H-biLSTM) that ensembles the two. To predict the factuality of the event referred to by a particular predicate, these models pass the output state of the biLSTM at that predicate through a two-layer regression. In the case of the H-biLSTM, the output state of both the L- and T-biLSTMs are simply concatenated and passed through the regression.[2]

Following the multi-task training regime described by Rudinger et al. (2018), we train these models on four standard factuality datasets – FactBank (Saurí and Pustejovsky, 2009, 2012), UW (Lee et al., 2015), MEANTIME (Minard et al., 2016), and UDS (White et al., 2016; Rudinger et al., 2018) – with tied biLSTM weights but regression parameters specific to each dataset. We then use these trained models to predict the factuality of the embedded predicate in our dataset.

To understand how much of these models' performance on our dataset is really due to a correct computation of lexicosyntactic inferences, we also generate predictions for the sentences in our dataset with the embedding verbs UNKed. In this case, the model can rely only on the syntactic context surrounding the predicate to make its inferences. We refer to the models with lexical information as the LEX models and the ones without lexical information as the UNK models.

Each model produces four predictions, corresponding to the four different datasets it was trained on. We consider three different ways of ensembling these predictions using a cross-validated ridge regression: (i) ensembling the four predictions for each specific model (LEX or UNK); (ii) ensembling the predictions for the LEX version of a particular model with the UNK version of that same model (LEX+UNK); and (iii) ensembling the predictions across all models (LEX, UNK, or LEX+UNK). Each ensemble is evaluated in a 10-fold/10-fold nested cross-validation (see Cawley and Talbot, 2010). In each iteration of the outer cross-validation, a 10% test set is split off, and a 10-fold cross-validation to tune the regularization is conducted on the remaining 90%.

4 Results

Figure 1 shows the mean correlation between model predictions and true factuality on the outer fold test sets of the nested cross-validation described in §3. We note three aspects of this plot.

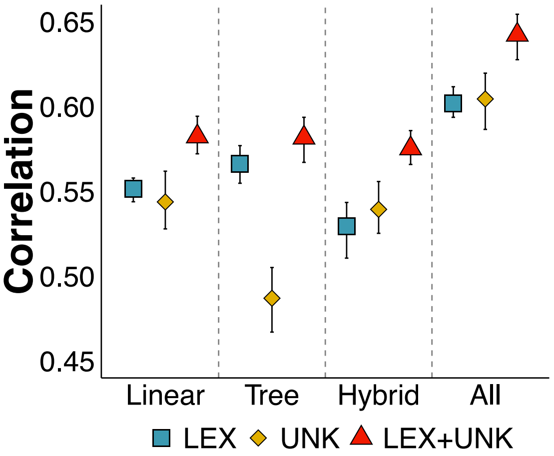

Figure 1: Mean correlation between model predictions and true factuality in nested cross-validation. Error bars show bootstrapped (iter=1,000) 95% confidence intervals for mean correlation across 10 outer folds.

First, among the LEX models, the T-biLSTM performs best, followed by the L-biLSTM, then the H-biLSTM. This is somewhat surprising, since Rudinger et al. find the opposite pattern of performance: the L- and H-biLSTMs vie for dominance, both outperforming the T-biLSTM. This indicates that T-biLSTMs are better able to represent the lexicosyntactic inferences relevant to this dataset, even though they underperform on more general datasets. This possibility is bolstered by the fact that, in contrast to the L- and H-biLSTMs, the LEX version of the T-biLSTMs performs significantly better than the UNK version, suggesting that the T-biLSTM is potentially more reliant on the lexical information than the other two.

Second, when the LEX and UNK version of each model is ensembled (LEX+UNK), we find comparable performance for all three biLSTMs – each outperforming the LEX version of the T-biLSTM. This indicates that each model captures similar amounts of information about lexicosyntactic inference, but this information is captured in the models' parameterizations in different ways.

Finally, when all three models are ensembled, we find that both the LEX and UNK version perform significantly better than any specific LEX+UNK model. This may indicate two things: (i) the models that only have access to syntax can perform just as well as ones that have access to both lexical information and syntax; but (ii) these models appear to capture different aspects of inference, since an ensemble of all models (All-LEX+UNK) performs significantly better than ei-

---

[2] See Appendix B for further details.

Someone ...	True	Pred.
faked that something happened	-3.15	0.86
was misinformed that something happened	-2.62	1.37
neglected to do something	-3.07	-0.02
pretended to have something	-2.96	0.05
was misjudged to have something	-2.46	0.55
forgot to have something	-3.18	-0.17
neglected to have something	-2.93	0.07
pretended that something happened	-2.11	0.86
declined to do something	-3.18	-0.22
was refused to do something	-3.16	-0.22
refused to do something	-3.12	-0.20
pretended to do something	-3.02	-0.11
disallowed someone to do something	-2.56	0.34
was declined to have something	-2.36	0.55
declined to have something	-3.12	-0.23
did n't hesitate to have something	1.84	-0.96
ceased to have something	-2.22	0.57
did n't hesitate to do something	1.86	-0.92
lied that something happened	-1.99	0.78
feigned to have something	-3.07	-0.31

Table 2: Sentences with the highest prediction errors.

ther the All-LEX or All-UNK ensembles alone.

Interestingly, however, even this ensemble performs more than 10 points worse than each model alone on FactBank, UW, and UDS. This raises the question of which lexicosyntactic inferences these models are missing – investigated below.

5 Analysis

We investigate two questions: (i) which inferences do all models do poorly on?; and (ii) what drives the differing strengths of each model?

Where do all models fail? Table 2 shows the 20 sentences with the highest prediction errors under the All-LEX+UNK ensemble. There are two interesting things to note about these sentences. First, most of them involve negative lexicosyntactic inferences that the model predicts to be either positive or near zero. Second, when the true inference is not positive, the matrix polarity of the original sentence is negative. This suggests that the models are not able to capture inferences whose polarity mismatches the matrix clause polarity.

One question that arises here is whether this inability affects all contexts equally. To answer this, we regress the absolute error of the predictions from this same ensemble (logged and standardized) against true factuality, matrix polarity, and context (as well as all of their two- and three-way interactions).[3] We find that the three-way interactions in this regression are reliable ($\chi^2(8)$=27.97, p < 0.001) – suggesting that there are nontrivial differences in these state-of-the-art factuality systems' ability to capture inferential interactions across verbs and syntactic contexts. The differences can be verified visually in Figure 2, which plots the factuality predicted by this ensemble against the true factuality from MegaVeridicality2.

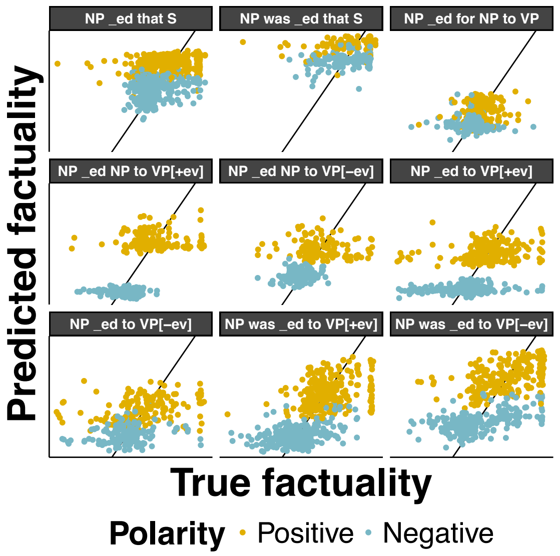

Figure 2: Factuality by syntactic context and polarity, each point a verb. Diagonals show perfect prediction.

To elaborate, the ensemble does best overall on contexts like (7a) and (7b), and worst overall on contexts like (7c). The contrast between (7b) and (7c) is particularly interesting because (i) (7c) is just the passivized form of (7b); and (ii) we do not observe similar behavior for contexts (7d) and (7e), which are analogous to (7b) and (7c), but replace the stative *have* with the eventive *do*.

(7) Someone...
  a. {_ed, didn't _} for something to happen.
  b. {_ed, didn't _} someone to have something.
  c. {was _ed, wasn't _ed} to have something.
  d. {_ed, didn't _} someone to do something.
  e. {was _ed, wasn't _ed} to do something.
  f. {_ed, didn't _} that something happened.

An additional nuance is that the ensemble does reliably better on the negative matrix polarity version of (7b) than on the positive, with the opposite true for (7e). This suggests these models do not capture an important inferential interaction between passivization and eventivity.

This suggestion is further bolstered by the fact that the ensemble's ability to predict cases where the matrix polarity mismatches the true factuality are reliably poorer in context (7c) but not in its minimal pairs (7e) and (7b), where the ensemble performs reliably poorer when the two match. Indeed, it is contexts (7c) and (7f) that drive the polarity mismatch effect evident in Table 2.

What drives differences between models? In §4, we noted two ways that the biLSTMs we in-

---

[3] See Appendix C for further details, including a summary of the regression on which the above discussion is based.

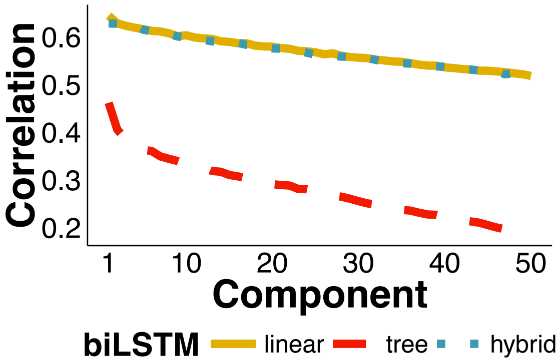

Figure 3: Canonical correlations between embedding verb embeddings and embedded verb hidden states.

vestigate differ: (i) the T-biLSTM appears to be more reliant on lexical information than L- and H-biLSTMs; and (ii) each model appears to encode information about lexicosyntactic inference in its parameterizations in different ways. We hypothesize that these two differences are related – specifically, that the T-biLSTM's heavier reliance on lexical information comes about as a consequence of stronger entanglement between lexical and syntactic information in its hidden states.

To probe this, we ask to what extent the embedding verb's embedding can be recovered from the embedded verb's hidden state using linear functions. If the lexical information is more strongly entangled with the syntactic information, it should be more difficult to construct a homomorphic (linear) function to decode the embedding verb's embedding from the embedded verb's hidden state. To measure this, we conduct a Canonical Correlation Analysis (CCA; Hotelling, 1936) between these two vector space representations for every sentence in our dataset. Given two matrices $\mathbf{X}$ (the embedding verb embeddings column stacked) and $\mathbf{Y}$ (the embedded verb hidden states column stacked), CCA constructs matrices $\mathbf{A}$ and $\mathbf{B}$, such that $\mathbf{a}_i, \mathbf{b}_i = \text{arg}_{\mathbf{a}', \mathbf{b}'}\text{max} \;\; \text{corr}(\mathbf{a}'\mathbf{X}, \mathbf{b}'\mathbf{Y})$ and $\text{corr}(\mathbf{a}_i\mathbf{X}, \mathbf{a}_j\mathbf{X}) = \text{corr}(\mathbf{b}_i\mathbf{Y}, \mathbf{b}_j\mathbf{Y}) = 0, \forall i < j$. This guarantees that the canonical correlation at component i, $\text{corr}(\mathbf{a}_i\mathbf{X}, \mathbf{b}_i\mathbf{Y})$, is nonincreasing in i, and thus the linearly decodable information about $\mathbf{Y}$ in $\mathbf{X}$ can be assessed using this function.

Figure 3 plots the canonical correlations for the first 50 components for each of the biLSTMs we investigated. We find that the canonical correlations associated with the T-biLSTM are substantially lower than those associated with the L- and H-biLSTMs across these first 50 components. This suggests that the T-biLSTM more strongly entangles lexical and syntactic information, perhaps explaining its apparently heavier reliance on lexical information, observed in §4.

Of note here is that the pattern seen in Figure 3 is probably at least partly a consequence of the different nonlinearities used for the L-biLSTM (tanh) and T-biLSTM (ReLU), and not the architectures themselves. But whether or not this pattern is due to the architectures, nonlinearities, or both, the entanglement hypothesis may still help explain the pattern of results discussed in §4.

6 Related work

This work is inspired by recent work in *recasting* various semantic annotations into natural language inference (NLI) datasets (White et al., 2017; Poliak et al., 2018a,b; Wang et al., 2018) to gain a better understanding of which phenomena standard neural NLI models (Bowman et al., 2015; Conneau et al., 2017) can capture – a line of work with deep roots (Cooper et al., 1996). The experimental setup – specifically, the idea of UNKing the embedding verb – was inspired by recent work that uses hypothesis-only baselines for a similar purpose (Gururangan et al., 2018; Poliak et al., 2018c; Tsuchiya, 2018). This work is also related to the broader investigation of sentence representations – particularly, tasks aimed at probing these representations' content (Pavlick and Callison-Burch, 2016; Adi et al., 2016; Conneau et al., 2018; Conneau and Kiela, 2018; Dasgupta et al., 2018).

7 Conclusion

We investigated neural models' ability to capture lexicosyntactic inference, taking the task of event factuality prediction (EFP) as a case study. We built a factuality judgment dataset for all English clause-embedding verbs in various syntactic contexts and used this dataset to probe current state-of-the-art EFP systems. We showed that these systems make certain systematic errors that are clearly visible through the lens of factuality.

Acknowledgments

This research was supported by the JHU HLTCOE, DARPA LORELEI and AIDA, NSF-BCS (1748969/1749025), and NSF-GRFP (1232825). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

References

Dorit Abusch. 2002. Lexical alternatives as a source of pragmatic presuppositions. *Semantics and Linguistic Theory*, 12:1–19.

Dorit Abusch. 2010. Presupposition triggering from alternatives. *Journal of Semantics*, 27(1):37–80.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

Pranav Anand and Valentine Hacquard. 2013. Epistemics and attitudes. *Semantics and Pragmatics*, 6(8):1–59.

Pranav Anand and Valentine Hacquard. 2014. Factivity, belief and discourse. In Luka Crnič and Uli Sauerland, editors, *The Art and Craft of Semantics: A Festschrift for Irene Heim*, volume 1, pages 69–90. MIT Working Papers in Linguistics, Cambridge, MA.

Rebekah Baglini and Itamar Francez. 2016. The implications of managing. *Journal of Semantics*, 33(3):541–560.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1466–1477, Berlin, Germany. Association for Computational Linguistics.

Željko Bošković. 1996. Selection and the Categorial Status of Infinitival Complements. *Natural Language & Linguistic Theory*, 14(2):269–304.

Željko Bošković. 1997. *The syntax of nonfinite complementation: An economy approach.* 32. MIT Press, Cambridge, MA.

Gavin C. Cawley and Nicola L.C. Talbot. 2010. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *J. Mach. Learn. Res.*, 11:2079–2107.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An Evaluation Toolkit for Universal Sentence Representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single \$&!#* vector: Probing sentence embeddings for linguistic properties. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1:2126–2136.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Josef Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman, Ted Briscoe, Holger Maier, and Karsten Konrad. 1996. Using the Framework. Technical Report LRE 62-051, The FRACAS Consortium.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The Pascal Recognising Textual Entailment Challenge. In *Machine learning challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising textual Entailment*, pages 177–190. Springer.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J. Gershman, and Noah D. Goodman. 2018. Evaluating Compositionality in Sentence Embeddings. *arXiv:1802.04302.*

Jon Robert Gajewski. 2007. Neg-raising and polarity. *Linguistics and Philosophy*, 30(3):289–328.

Thomas Angelo Grano. 2012. *Control and Restructuring at the Syntax-Semantics Interface*. Ph.D. thesis, University of Chicago.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference-6: A Brief History. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 466–471, Stroudsburg, PA, USA. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation Artifacts in Natural Language Inference Data. *arXiv:1803.02324.*

Irene Heim. 1992. Presupposition projection and the semantics of attitude verbs. *Journal of Semantics*, 9(3):183–221.

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

Laurence Robert Horn. 1972. *On the Semantic Properties of Logical Operators in English*. Ph.D. thesis, UCLA.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644.

Lauri Karttunen. 1971a. Implicative verbs. *Language*, pages 340–358.

Lauri Karttunen. 1971b. Some observations on factivity. *Papers in Linguistics*, 4(1):55–69.

Lauri Karttunen. 2012. Simple and phrasal implicatives. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 124–131. Association for Computational Linguistics.

Lauri Karttunen. 2013. You will be lucky to break even. In Tracy Holloway King and Valeria dePaiva, editors, *From Quirky Case to Representing Space: Papers in Honor of Annie Zaenen*, pages 167–180.

Lauri Karttunen and Stanley Peters. 1979. Conventional implicature. *Syntax and Semantics*, 11:1–56.

Lauri Karttunen, Stanley Peters, Annie Zaenen, and Cleo Condoravdi. 2014. The Chameleon-like Nature of Evaluative Adjectives. In *Empirical Issues in Syntax and Semantics 10*, pages 233–250. CSSP-CNRS.

Paul Kiparsky and Carol Kiparsky. 1970. Fact. In Manfred Bierwisch and Karl Erich Heidolph, editors, *Progress in Linguistics: A collection of papers*, pages 143–173. Mouton, The Hague.

Kenton Lee, Yoav Artzi, Yejin Choi, and Luke Zettlemoyer. 2015. Event Detection and Factuality Assessment with Non-Expert Supervision. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1648, Lisbon, Portugal. Association for Computational Linguistics.

Noor van Leusen. 2012. The Accommodation Potential of Implicative Verbs. In *Logic, Language and Meaning*, volume 7218 of *Lecture Notes in Computer Science*, pages 421–430. Springer.

Bill MacCartney, Michel Galley, and Christopher D Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of the conference on empirical methods in natural language processing*, pages 802–811. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2012. Did it happen? The pragmatic complexity of veridicality assessment. *Computational Linguistics*, 38(2):301–333.

Roger Martin. 2001. Null case and the distribution of PRO. *Linguistic Inquiry*, 32(1):141–166.

Roger Andrew Martin. 1996. *A minimalist theory of PRO and control*. Ph.D. thesis, University of Connecticut, Storrs.

Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begoña Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son. 2016. MEANTIME, the NewsReader Multilingual Event and Time Corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 23–28, Paris, France. European Language Resources Association (ELRA).

Makoto Miwa and Mohit Bansal. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Prerna Nadathur. 2016. Causal necessity and sufficiency in implicativity. *Semantics and Linguistic Theory*, 26:1002–1021.

Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. In *Proceedings of the Fifth International Workshop on Inference in Computational Semantics (ICoS-5)*, pages 20–21, Buxton, England. Association for Computational Linguistics.

Shinichi Nakagawa and Holger Schielzeth. 2013. A general and simple method for obtaining R2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution*, 4(2):133–142.

Ellie Pavlick and Chris Callison-Burch. 2016. Most "babies" are "little" and most "problems" are "huge": Compositional Entailment in Adjective-Nouns. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1:2164–2173.

David Pesetsky. 1991. Zero syntax: vol. 2: Infinitives.

Adam Poliak, Yonatan Belinkov, James Glass, and Benjamin Van Durme. 2018a. On the Evaluation of Semantic Phenomena in Neural Machine Translation Using Natural Language Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 513–523, New Orleans.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018b. Collecting Diverse Natural Language Inference Problems for Sentence Representation Evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018c. Hypothesis Only Baselines in Natural Language Inference. In *Proceedings of the 7th Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 180–191, New Orleans. Association for Computational Linguistics.

Rachel Rudinger, Aaron Steven White, and Benjamin Van Durme. 2018. Neural Models of Factuality. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 731–744, New Orleans. Association for Computational Linguistics.

Roser Saurí and James Pustejovsky. 2009. FactBank: a corpus annotated with event factuality. *Language Resources and Evaluation*, 43(3):227.

Roser Saurí and James Pustejovsky. 2012. Are you sure that this happened? assessing the factuality degree of events in text. *Computational Linguistics*, 38(2):261–299.

Mandy Simons. 2001. On the conversational basis of some presuppositions. *Semantics and Linguistic Theory*, 11:431–448.

Mandy Simons. 2007. Observations on embedding verbs, evidentiality, and presupposition. *Lingua*, 117(6):1034–1056.

Mandy Simons, Judith Tonhauser, David Beaver, and Craige Roberts. 2010. What projects and why. *Semantics and linguistic theory*, 20:309–327.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association of Computational Linguistics*, 2(1):207–218.

Gabriel Stanovsky, Judith Eckle-Kohler, Yevgeniy Puzikov, Ido Dagan, and Iryna Gurevych. 2017. Integrating Deep Linguistic Features in Factuality Prediction over Unified Datasets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 352–357. Association for Computational Linguistics.

Tim Stowell. 1982. The tense of infinitives. *Linguistic Inquiry*, 13(3):561–570.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Masatoshi Tsuchiya. 2018. Performance Impact Caused by Hidden Bias of Training Data for Recognizing Textual Entailment. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *arXiv:1804.07461*.

Aaron Steven White. 2014. Factive-implicatives and modalized complements. In *Proceedings of the 44th annual meeting of the North East Linguistic Society*, pages 267–278, University of Connecticut.

Aaron Steven White, Pushpendre Rastogi, Kevin Duh, and Benjamin Van Durme. 2017. Inference is Everything: Recasting Semantic Resources into a Unified Evaluation Framework. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 996–1005.

Aaron Steven White and Kyle Rawlins. 2016. A computational model of S-selection. *Semantics and Linguistic Theory*, 26:641–663.

Aaron Steven White and Kyle Rawlins. 2018. The role of veridicality and factivity in clause selection. In *Proceedings of the 48th Annual Meeting of the North East Linguistic Society*, page to appear, Amherst, MA. GLSA Publications.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, TX. Association for Computational Linguistics.

Susi Wurmbrand. 2014. Tense and aspect in English infinitives. *Linguistic Inquiry*, 45(3):403–447.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Dual Fixed-Size Ordinally Forgetting Encoding (FOFE) for Competitive Neural Language Models

Sedtawut Watcharawittayakul* and **Mingbin Xu*** and **Hui Jiang**
Department of Electrical Engineering and Computer Science
Lassonde School of Engineering, York University
4700 Keele Street, Toronto, Ontario, Canada
`{watchara, xmb, hj}@eecs.yorku.ca`

Abstract

In this paper, we propose a new approach to employ the fixed-size ordinally-forgetting encoding (FOFE) (Zhang et al., 2015b) in neural languages modelling, called *dual-FOFE*. The main idea behind dual-FOFE is that it allows to use two different forgetting factors so that it can avoid the trade-off in choosing either small or large values for the single forgetting factor in the original FOFE. In our experiments, we have compared the dual-FOFE based neural network language models (NNLM) against the original FOFE counterparts and various traditional NNLMs. Our results on the challenging Google Billion Words corpus show that both FOFE and dual FOFE yield very strong performance while significantly reducing the computational complexity over other NNLMs. Furthermore, the proposed dual-FOFE method further gives over 10% relative improvement in perplexity over the original FOFE model.

1 Introduction

Language modelling is an essential task for many natural language processing (NLP) applications including speech recognition, machine translation and text summarization. The goal of language modelling is to learn the distribution over a sequence of characters or words; this distribution may be utilized for encoding the language structure (e.g. the grammatical structure) as well as extracting information from the corpora (Jozefowicz et al., 2016). In the recent years, the popularity of neural networks (NN) has been a significant driving force for language modelling (LM) research; the well-known NN-LM models includes the feedforward NN-LMs (FNN-LMs) (Bengio et al., 2001, 2003), recurrent NN-LMs (RNN-LMs) (Mikolov et al., 2010; Mikolov and Zweig, 2012) and the long short-term memory (LSTM-LMs) (Hochreiter and Schmidhuber, 1997). Among all, FNN-LMs often have a simpler and more efficient learning process, but they tend to underperform the other NN-LMs due to the limited capability to memorize the long term dependency in natural languages (Zhang et al., 2015b). However this drawback could be addressed by applying the fixed-size ordinally-forgetting encoding (FOFE) to FNN's inputs. FOFE is an encoding method, which relies on the ordinally-forgetting mechanism to encode any word sequence based on the positions of words; this also allows the FOFE code to capture the long-term dependency (Zhang et al., 2015b). As shown in Zhang (2015b), FNN-LMs with FOFE can easily yield comparable performance as other NN-LMs. The key parameter in the FOFE method is the forgetting factor, which is responsible for determining the degree of sensitivity of the encoding with respect to the past context. However, the choice of a good value for the forgetting factor could be tricky since both small and large forgetting factors are offering different benefits.

In this paper, we propose a simple alteration to FOFE method, which allows to incorporate two forgetting factors into the fixed-size encoding of the variable-length word sequences. We name this approach as *dual-FOFE*. Our hypothesis is that by incorporating both the small and large forgetting factors in the FOFE encoding, the dual-FOFE is able to simultaneously optimize the abilities to capture the positional information as well as to model long term dependency. In our experiments, we have evaluated the proposed dual FOFE models on two large scale language modeling tasks, namely *enwiki9* and *Google Billion Words (GBW)* corpora. Experimental results have shown that both FOFE models yield very competitive performance on these tasks, comparable with the state-

---

*Equal contribution.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4725–4730
Brussels, Belgium, October 31 - November 4, 2018.

of-the-art systems but with significantly reduced learning complexity. Furthermore, the proposed dual-FOFE method further gives over 10% relative improvement in perplexity over the original FOFE model.

2 Related Work

In this section, we will briefly review the NN-LMs and the original FOFE method. The general idea behind NN-LM is to project the discrete words onto a continuous space, then learn to estimate the conditional probabilities of each known word within the projected space. The training of NN-LMs are often incredibly slow due to the inefficiency of softmax normalization when applied to the extremely large output layer. The solution currently used by many NN-LMs (including our models in this work) is to use noise contrastive estimation (NCE) (Gutmann and Hyvrinen, 2010). The basic idea of NCE is to reduce the probability estimation problem into a probabilistic binary classification problem (Mnih and Teh, 2012; Mnih and Kavukcuoglu, 2013).

2.1 Fixed-Size Ordinally Forgetting Encoding

Fixed-size ordinally-forgetting Encoding (FOFE) is an encoding method which generates a fixed-size representation, namely the FOFE code, for any variable-length word sequence (Zhang et al., 2015b). For a given word sequence $S = \{w_1, w_2, ..., w_T\}$, let e_t denote the one-hot representation of the word w_t, z_t for the FOFE code of the partial word sequence up to word w_t, z_t is computed as follows:

$$z_t = \alpha \cdot z_{t-1} + e_t \quad (1 \leq t \leq T) \tag{1}$$

where α ($0 < \alpha < 1$) denotes the forgetting factor, a parameter responsible for determining the degree of influence each time step of the past context has on the FOFE code. Obviously, FOFE can convert any variable-length sequence into a fixed-size code with length equal to the size of vocabulary.

In regard to uniqueness of FOFE code, the code is said to be (almost) unique under the two theorems (proven in Zhang (2015a)):

Theorem 1 *If $0 < \alpha \leq 0.5$, then FOFE code is guarantee uniqueness for any values of vocabulary's size and sequence's length.*

Theorem 2 *If $0.5 < \alpha < 1$, then FOFE code is guarantee almost uniqueness for any finite values of vocabulary's size and sequence's length, except for a finite set of countable choices of α.*

Furthermore, the chance of actually having any collisions for α between 0.5 and 1 is nearly impossible in practice, due to quantization errors in real computer systems. Hence in practice, it is safe to argue that FOFE is able to uniquely encodes any variable-length sequence into a fixed-size representation.

2.2 FOFE for FNN-LMs

The idea of FOFE based NN-LMs is to use FOFE to encode the partial history sequence of past words in a sentence, then feed this fixed-size FOFE code to a feedforward neural network as an input to predict next word. As shown in Figure 1, the FOFE code could be efficiently computed via time-delayed recursive structure, where the symbol z^{-1} in the figure represents a unit time delay (or equivalently a memory unit) from z_t to z_{t-1}.

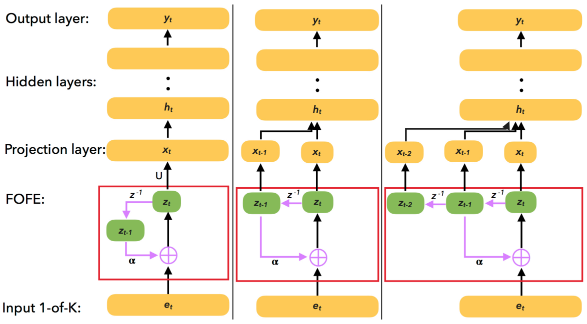

Figure 1: (Left) 1st-order FOFE FNN-LM; (Center) 2nd-order FOFE FNN-LM; (Right) 3rd-order FOFE FNN-LM.

The basic architecture for FOFE based FNN-LMs (called 1st-order) is the standard FNN architecture with an additional layer for encoding the input into FOFE code. However, in this work, we use the 2nd-order and 3rd-order FOFE FNN-LMs, which are shown to produce slightly better results (Zhang et al., 2015b). In a 2nd-order FOFE model, both the current partial sequences FOFE code (denoted as z_t) and the previous partial sequences FOFE code (denoted as z_{t-1}) are utilized to predict next word. In a 3rd-order FOFE model, all z_t, z_{t-1} and z_{t-2} are used as inputs to neural networks.

More recently, the FOFE methods have been successfully applied to many NLP tasks, including word embedding (Sanu et al., 2017), named entity recognition (Xu et al., 2017a), entity discovery and linking (Xu et al., 2016, 2017b).

3 Dual-FOFE

The main idea of dual-FOFE is to generate augmented FOFE encoding codes by concatenating two FOFE codes using two different forgetting factors. Each of these FOFE codes is still computed in the same way as the mathematical formulation shown in Equation (1). The difference between them is that we may select to use two different values for the forgetting factor (denoted as α) for additional modeling benefits.

3.1 Intuition behind Dual-FOFE

As mentioned in the subsection 2.1, the values in a FOFE code are used to encode both the content and the order information in a sequence. This is achieved by a recursive encoding method where at each recursive step the code will be multiplied by the forgetting factor (α) whose value is bounded by $0 < \alpha < 1$. In a practical computer with finite precision, this has an impact on the FOFE's abilities to precisely memorize the long-term dependency of past context as well as to properly represent the positional information.

The FOFE's ability to represent the positional information would improve with smaller forgetting factors. The reason is that that when α is small, the FOFE code (z_t) for each word vastly differs from its neighbour in magnitude. If α is too large (close to 1), the contribution of a word may not change too much no matter where it is. This may hamper the following neural networks to model the positional information. Conversely, the FOFE's ability to model the long-term dependency of the older context would improve with larger forgetting factors. This is because when α is small, the contribution of a word from the older history may quickly underflow to become irrelevant (i.e. forgotten) when computing the current word.

In the original FOFE with just a single forgetting factor, we would have to determine the best trade-off between these two benefits. On the other hand, the dual-FOFE does not face such issues since it is composed of two FOFE codes: the half of the dual-FOFE code using a smaller forgetting factor is solely optimized and responsible for representing the positional information of all words in the sequence; meanwhile the other half of the dual-FOFE code using a larger forgetting factor is optimized and responsible for maintaining the long-term dependency of past context.

3.2 Dual-FOFE based FNN-LM

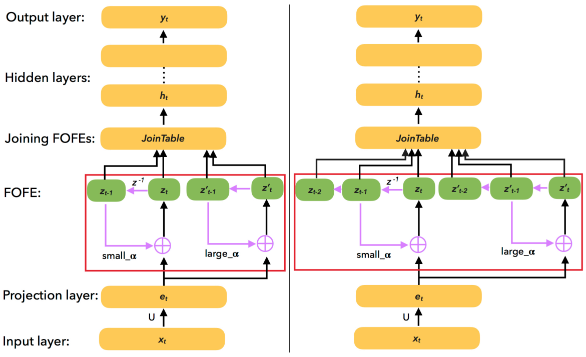

Figure 2: (Left) 2nd-order Dual-FOFE FNN-LM; (Right) 3rd-order Dual-FOFE FNN-LM.

As shown in Figure 2, the architecture of dual-FOFE based FNN-LMs is very similar to the original FOFE FNN-LMs.[1] In the Dual-FOFE FNN-LMs, the input word sequence would have to pass through two branches of the FOFE layers (using two different forgetting factors) and each encoding branch will produce a FOFE code representing the input sequence. These two FOFE codes are then joined to produce the dual-FOFE code, which would be fed to FNNs to predict the next word.

It might also be worth noting that in our implementation we do not explicitly reset FOFE codes, i.e. z_t value, at sentence boundaries. However, far-away histories will be gradually forgotten by the recursive calculation in FOFE due to $0 < \alpha < 1$ and finite precision in computers.

3.3 Dual-FOFE vs. Higher Order FOFE

As mentioned previously in the subsection 2.2, the higer order FOFE codes would utilize both the current and the previous sequence FOFE codes for prediction. Hence similar to dual-FOFE, the higher order FOFE could also maintain the sensitivity to both nearby and faraway context. Obviously a much higher order FOFE code may be required in order to achieve the same effect as dual-FOFE in terms of modelling long-term dependency. In this case, the higher order FOFE may also significantly increase the number of parameters in the input layer. At last, the dual FOFE

---

[1] The difference in the location of the projection layer between Figure 1 and 2 simply indicates two equivalent ways to do word projection. Figure 1 was originally from Zhang (2015b), but they mentioned in text (without a figure) that it is more efficient to do projection as in Figure 2 and both methods are mathematically equivalent since both projection and FOFE steps are linear.

and the higher order FOFE are largely complementary since we have observed consistent performance gains when combining dual FOFE with either 2nd-order or 3rd-order FOFE in our experiments.

4 Experiments

In this work, we have evaluated the proposed dual-FOFE based FNN-LMs against various traditional neural language models on two corpora: i) *enwik9* corpus: it consists of the first 1 billion bytes of English wikipedia dump, having total size of 170.8 million words; the corpus was divided into three parts: the training set (153M words), the test set (8.9M words), and the validation set (8.9M words); the vocabulary size is limited to 80k words (Zhang et al., 2015b). ii) Google Billion Words (GBW) corpus: it contains about 800 million words and the corpus is divided into two parts: the training set (792M words) and the test set (8M words); the vocabulary size for this corpus is limited to 800k words (Chelba et al., 2013).

4.1 Results on enwiki9

In the experiments on the enwiki9 corpus, we have trained three dual-FOFE FNN-LMs with different forgetting factor pairs, one FOFE FNN-LM, and one tri-FOFE FNN-LM. All five models adopt a 2nd-order FOFE structure, employing a word embeddings of 256 dimensions, three hidden layers of $400, 600, 600$ neurons and an output layer of 80k words (reflecting the vocabulary). [2] Note that the dual-FOFE FNN-LMs have to double the size of input context windows since dual-FOFE essentially contain two FOFE codes. But this increase only accounts for a negligible faction of total model parameters.

As shown in Table 1, all three dual-FOFE FNN-LMs, using three pairs of forgetting factors as (0.5, 0.7) and (0.7, 0.9) and (0.5, 0.9), can significantly outperform other traditional models previously reported on this corpus. We also note that it is beneficial to include a relatively large forgetting factor, such as 0.9, in the dual FOFE models since such a large alpha may help to memorize much longer context in the inputs. When compared with the original FOFE counterpart, the best dual-FOFE model using forgetting factors (0.5, 0.9) offers a relative gain of around 8% in test PPL.

It is worth noting that our dual-FOFE models can be extended to incorporate more than two alpha values. In fact after we have obtained a strong result supporting our dual-FOFE hypothesis, we have performed additional experiments using three alpha values, the so-called tri-FOFE model. The result on Table 1 has shown that the tri-FOFE FNN-LMs still slightly outperforms the dual-FOFE models. However, the gain is marginal. This leads us to believing that further extension of more alpha values in FOFE would be of limited use.

4.2 Results on Google Billion Words (GBW)

In the experiments on the GBW corpus, we have trained one dual-FOFE FNN-LM and one FOFE FNN-LM. Following the best dual-FOFE model configuration on the previous corpus, this dual-FOFE FNN-LM uses the same pair of dual forgetting factors $(0.5, 0.9)$. Both models adopt a 3rd-order structure, employing word embeddings of 256 dimensions, three hidden layers each of 4096 neurons, a compression layer with 720 neurons, and an output layer of 800k words (reflecting the vocabulary). Although dual-FOFE FNN-LM has doubled the size of input context windows of FOFE FNN-LM, the total number of model parameters in both models are almost equal, roughly 0.82 billion parameters.

As shown in Table 2, the dual-FOFE FNN-LM is able to produce a very competitive performance, comparable with the best previously reported results on this task, such as GCNN-13 (Dauphin et al., 2016) and LSTM-LM (Jozefowicz et al., 2016). The dual-FOFE FNN-LM are among the few single-model systems that are able to achieve test PPL below 40 on this task. Furthermore, our proposed dual FOFE model can significantly reduce the computational complexity, e.g., our model has a relatively smaller number of parameter (0.82B parameters) and it requires much less hardware resource to train (using only 1 GPU in our experiments). When compared with the original FOFE counterpart, the dual-FOFE FNN-LM is able to provide approximately 11% relative improvement in PPL.

5 Conclusions

In this paper, we have proposed a new approach of utilizing the fixed-size ordinally-forgetting encoding (FOFE) method for neural network lan-

---

[2] Comparing with Zhang (2015b), our single FOFE FNN-LM baseline use a slightly larger model, which lead to slightly better perplexity.

Table 1: Test PPL of various LMs on *enwiki9*.

Model	Architecture	PPL
KN 3-gram (Zhang et al., 2015b)	-	156
KN 5-gram (Zhang et al., 2015b)	-	132
FNN-LM 2-gram (Zhang et al., 2015b)	[2*200]-600-600-80k	150
FNN-LM 3-gram (Zhang et al., 2015b)	[3*200]-400-400-80k	131
FNN-LM 4-gram (Zhang et al., 2015b)	[4*200]-400-400-80k	125
RNN-LM (Zhang et al., 2015b)	[1*600]-80k	112
FOFE[α=0.7] FNN-LM (Zhang et al., 2015b)	[2*200]-600-600-80k	107
FOFE[α=0.7] FNN-LM	[2*256]-400-600-600-80k	104.8
Dual-FOFE[α=0.5,0.7] FNN-LM	[2*2*256]-400-600-600-80k	101.7
Dual-FOFE[α=0.7,0.9] FNN-LM	[2*2*256]-400-600-600-80k	97.0
Dual-FOFE[α=0.5,0.9] FNN-LM	[2*2*256]-400-600-600-80k	**96.6**
Tri-FOFE[α=0.5,0.7,0.9] FNN-LM	[3*2*256]-400-600-600-80k	**95.9**

Table 2: Test PPL of various LMs on Google Billion Words.

model	PPL	#param	hardware
Sigmoid-RNN-2048 (Ji et al., 2015)	68.3	4.1B	1 CPU
Interpolated KN 5-gram & 1.1B n-grams (Chelba et al., 2013)	67.6	1.8B	100 CPUs
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33B	-
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3	20B	24 GPUs
LSTM-1024-512 (Jozefowicz et al., 2016)	48.2	0.82B	40 GPUs
LSTM-2048-512 (Jozefowicz et al., 2016)	43.7	0.83B	40 GPUs
LSTM + CNN input (Jozefowicz et al., 2016)	30.0	1.04B	40 GPUs
GCNN-13 (Dauphin et al., 2016)	38.1	-	1 GPU
FOFE[α=0.7] FNN-LM [3*256]-4096*3-720-800k	43.6	0.82B	1 GPU
Dual-FOFE[α=0.5,0.9] FNN-LM [2*3*256]-4096*3-720-800k	**39.0**	**0.82B**	**1 GPU**

guage models (NN-LMs), known as dual-FOFE. As the name implies, this approach involves to produce a new fixed-sized representation for any variable-length sequence from a concatenation of two FOFE codes. This would have allowed us to select two values for the forgetting factors. One FOFE code with a smaller forgetting factor is responsible for representing the positional information of all words in the sequence while the other using a larger forgetting factor is responsible for modelling the even longer term dependency in far away history. Our experiments on both *enwiki9* and Google Billion Words (GBW) tasks have both demonstrated the effectiveness of the dual-FOFE modeling approach. Experimental results on the challenging GBW corpus have shown that the dual-FOFE FNN-LM has achieved over 10% improvement in perplexity over the original FOFE FNN-LM, without any significant drawback in model and learning complexity. When compared with other traditional neural language models, the dual-FOFE FNN-LM has achieved competitive performance with significantly lower computational complexity.

Acknowledgement

This work is supported mainly by a research donation from iFLYTEK Co., Ltd., Hefei, China, and partially by a discovery grant from Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In *Advances in Neural Information Processing Systems 13*, pages 932–938. MIT Press.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3:1137–1155.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083.

Michael Gutmann and Aapo Hyvrinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 297–304. Journal of Machine Learning Research - Proceedings Track.

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

Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. 2015. Blackout: Speeding up recurrent neural network language models with very large vocabularies. *CoRR*, abs/1511.06909.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network based language model. In *2012 IEEE Spoken Language Technology Workshop*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2265–2273, USA. Curran Associates Inc.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *ArXiv e-prints*.

Joseph Sanu, Mingbin Xu, Hui Jiang, and Quan Liu. 2017. Word embeddings based on fixed-size ordinally forgetting encoding. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 310–315, Copenhagen, Denmark. Association for Computational Linguistics.

Noam Shazeer, Joris Pelemans, and Ciprian Chelba. 2015. Sparse non-negative matrix language modeling for skip-grams. In *Proceedings of Interspeech 2015*, pages 1428–1432. International Speech Communication Association.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017a. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1237–1247, Vancouver, Canada. Association for Computational Linguistics.

Mingbin Xu, Nargiza Nosirova, Kelvin Jiang, Feng Wei, and Hui Jiang. 2017b. FOFE-based deep neural networks for entity discovery and linking. In *Proceedings of the Text Analysis Conference (TAC) 2017*.

Mingbin Xu, Feng Wei, Sedtawut Watcharawittayakul, Yuchen Kang, and Hui Jiang. 2016. The YorkNRM systems for trilingual EDL tasks at TAC KBP 2016. In *Proceedings of the Text Analysis Conference (TAC) 2016*.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Li-Rong Dai. 2015a. A fixed-size encoding method for variable-length sequences with its application to neural network language models. *CoRR*, abs/1505.01504.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015b. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*, pages 495–500. Association for Computational Linguistics.

The Importance of Being Recurrent for Modeling Hierarchical Structure

Ke Tran[1] **Arianna Bisazza**[2] **Christof Monz**[1]
[1] Informatics Institute, University of Amsterdam
[2] Leiden Institute of Advanced Computer Science, Leiden University
{m.k.tran,c.monz}@uva.nl a.bisazza@liacs.leidenuniv.nl

Abstract

Recent work has shown that recurrent neural networks (RNNs) can implicitly capture and exploit hierarchical information when trained to solve common natural language processing tasks (Blevins et al., 2018) such as language modeling (Linzen et al., 2016; Gulordava et al., 2018) and neural machine translation (Shi et al., 2016). In contrast, the ability to model structured data with non-recurrent neural networks has received little attention despite their success in many NLP tasks (Gehring et al., 2017; Vaswani et al., 2017). In this work, we compare the two architectures—*recurrent* versus *non-recurrent*—with respect to their ability to model hierarchical structure and find that recurrency is indeed important for this purpose. The code and data used in our experiments is available at `https://github.com/ketranm/fan_vs_rnn`

1 Introduction

Recurrent neural networks (RNNs), in particular Long Short-Term Memory networks (LSTMs), have become a dominant tool in natural language processing. While LSTMs appear to be a natural choice for modeling sequential data, recently a class of non-recurrent models (Gehring et al., 2017; Vaswani et al., 2017) have shown competitive performance on sequence modeling. Gehring et al. (2017) propose a fully convolutional sequence-to-sequence model that achieves state-of-the-art performance in machine translation. Vaswani et al. (2017) introduce Transformer networks that do not use any convolution or recurrent connections while obtaining the best translation performance. These non-recurrent models are appealing due to their highly parallelizable computations on modern GPUs. But do they have the same ability to exploit hierarchical structures *implicitly* in comparison to RNNs? In this work, we provide a first answer to this question.

Our interest here is the ability of capturing hierarchical structure without being equipped with explicit structural representations (Bowman et al., 2015b; Tran et al., 2016; Linzen et al., 2016). We choose Transformer as a non-recurrent model to study in this paper. We refer to Transformer as Fully Attentional Network (FAN) to emphasize this characteristic. Our motivation to favor FANs over convolutional neural networks (CNNs) is that FANs always have full access to the sequence history, making them more suited for modeling long distance dependencies than CNNs. Additionally, FANs promise to be more interpretable than LSTMs by visualizing attention weights.

The rest of the paper is organized as follows: We first highlight the differences between the two architectures (§2) and introduce the two tasks (§3). Then we provide setup and results for each task (§4 and §5) and discuss our findings (§6).

2 FAN versus LSTM

Conceptually, FANs differ from LSTMs in the way they utilize the previous input to predict the next output. Figure 1 depicts the main difference in terms of computation when each model is making predictions. At time step t, a FAN can access information from all previous time steps *directly* with $\mathcal{O}(1)$ computational operations. FANs do so by employing a self-attention mechanism to compute the weighted average of all previous input representations. In contrast, LSTMs compress at each time step all previous information into a single vector *recursively* based on the current input and the previous compressed vector. By their definition, LSTMs require $\mathcal{O}(d)$ computational operations to access the information at time step $t - d$.

For the details of self-attention mechanics in FANs, we refer to the work of Vaswani et al. (2017). We now proceed to measure both models' ability to

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4731–4736
Brussels, Belgium, October 31 - November 4, 2018.

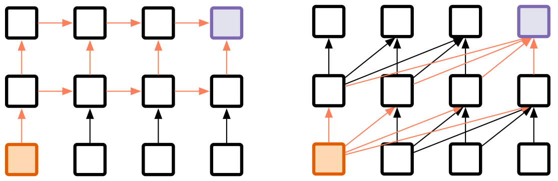

(a) LSTM (b) FAN

Figure 1: Diagram showing the main difference between a LSTM and a FAN. Purple boxes indicate the summarized vector at current time step t which is used to make prediction. Orange arrows indicate the information flow from a previous input to that vector.

learn hierarchical structure with a set of controlled experiments.

3 Tasks

We choose two tasks to study in this work: (1) subject-verb agreement, and (2) logical inference. The first task was proposed by Linzen et al. (2016) to test the ability of recurrent neural networks to capture syntactic dependencies in natural language. The second task was introduced by Bowman et al. (2015b) to compare tree-based recursive neural networks against sequence-based recurrent networks with respect to their ability to exploit hierarchical structures to make accurate inferences. The choice of tasks here is important to ensure that both models have to exploit hierarchical structural features (Jia and Liang, 2017).

4 Subject-Verb Agreement

Linzen et al. (2016) propose the task of predicting number agreement between subject and verb in naturally occurring English sentences as a proxy for the ability of LSTMs to capture hierarchical structure in natural language. We use the dataset provided by Linzen et al. (2016) and follow their experimental protocol of training each model using either (a) a general language model, i.e., next word prediction objective, and (b) an explicit supervision objective, i.e., predicting the number of the verb given its sentence history. Table 1 illustrates the training and testing conditions of the task.

Data: Following the original setting, we take 10% of the data for training, 1% for validation, and the rest for testing. The vocabulary consists of the 10k most frequent words, while the remaining words are replaced by their part-of-speech.

Table 1: Examples of training and test conditions for the two subject-verb agreement subtasks. The full input sentence is "The **keys** to the cabinet **are** on the table" where verb and subject are bold and intervening nouns are underlined.

	Input	Train	Test
(a)	the keys to the cabinet	are	$p(\text{are}) > p(\text{is})$?
(b)	the keys to the cabinet	plural	plural/singular?

Hyperparameters: To allow for a fair comparison, we find the best configuration for each model by running a grid search over the following hyperparameters: number of layers in $\{2, 3, 4\}$, dropout rate in $\{0.2, 0.3, 0.5\}$, embedding size and number of hidden units in $\{128, 256, 512\}$, number of heads (for FAN) in $\{2, 4\}$, and learning rate in $\{0.00001, 0.0001, 0.001\}$. The weights of the word embeddings and output layer are shared (Inan et al., 2017; Press and Wolf, 2017). Models are optimized by Adam (Kingma and Ba, 2015).

We first assess whether the LSTM and FAN models trained with respect to the language model objective assign higher probabilities to the correctly inflected verbs. As shown in Figures 2a and 2b, both models achieve high accuracies for this task, but LSTMs consistently outperform FANs. Moreover, LSTMs are clearly more robust than FANs with respect to task difficulty, measured both in terms of word distance and number of agreement attractors[1] between subject and verb. Christiansen and Chater (2016); Cornish et al. (2017) have argued that human memory limitations give rise to important characteristics of natural language, including its hierarchical structure. Similarly, our experiments suggest that, by compressing the history into a single vector before making predictions, LSTMs are forced to better learn the input structure. On the other hand, despite having direct access to all words in their history, FANs are less capable of detecting the verb's subject. We note that the validation perplexities of the LSTM and FAN are 67.06 and 69.14, respectively.

Secondly, we evaluate FAN and LSTM models explicitly trained to predict the verb number (Figures 2c and 2d). Again, we observe that LSTMs consistently outperform FANs. This is a particularly interesting result since the self-attention mechanism in FANs connects two words in any po-

---

[1] Agreement attractors are intervening nouns with the opposite number from the subject.

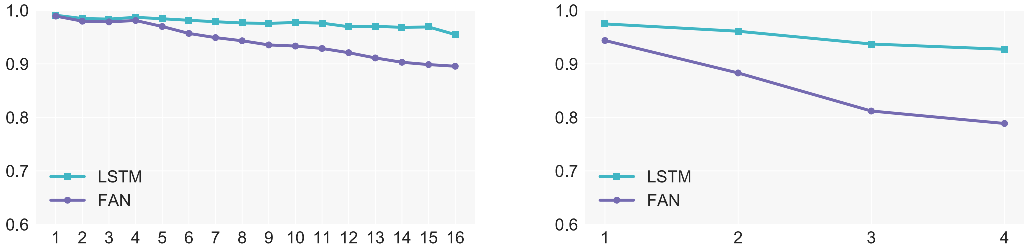

(a) Language model, breakdown by distance (b) Language model, breakdown by # attractors

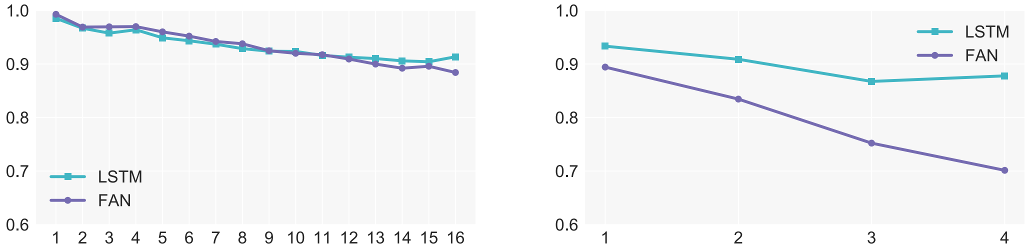

(c) Number prediction, breakdown by distance (d) Number prediction, breakdown by # attractors

Figure 2: Results of subject-verb agreement with different training objectives.

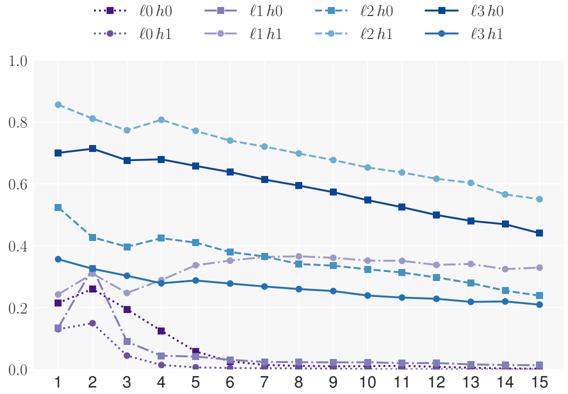

Figure 3: Proportion of times the subject is the most attended word by different heads at different layers ($\ell 3$ is the highest layer). Only cases where the model made a correct prediction are shown.

sition with a $\mathcal{O}(1)$ number of executed operations, whereas RNNs require more recurrent operations. Despite this apparent advantage of FANs, the performance gap between FANs and LSTMs increases with the distance and number of attractors.[2]

To gain further insights into our results, we examine the attention weights computed by FANs during verb-number prediction (supervised objective). Specifically, for each attention head at each layer of the FAN, we compute the percentage of times the subject is the most attended word among all words in the history. Figure 3 shows the results for all cases where the model made the correct prediction. While it is hard to interpret the exact role of attention for different heads and at different layers, we find that some of the attention heads at the higher layers ($\ell 2\ h1$, $\ell 3\ h0$) frequently point to the subject with an accuracy that decreases linearly with the distance between subject and verb.

5 Logical inference

In this task, we choose the artificial language introduced by Bowman et al. (2015b). The vocabulary of this language includes six word types {*a, b, c, d, e, f*} and three logical operators {*or*, *and*, *not*}. The task consists of predicting one of seven mutually exclusive logical relations that describe the relationship between a pair of sentences: entailment ($\sqsubset$, $\sqsupset$), equivalence ($\equiv$), exhaustive and non-exhaustive contradiction ($^\wedge$, $|$), and two types of semantic independence ($\#$, $\smile$). We generate 60,000 samples[3] with the number of logical operations ranging from 1 to 12. The train/dev/test dataset ratios are set to 0.8/0.1/0.1. Here are some samples of the training data:

(d (or f)) $\sqsupset$ (f (and a))
(d (and (c (or d)))) $\#$ (not f)
(not (d (or (f (or c))))) $\sqsubset$ (not (c (and (not d))))

Why artificial data? Despite the simplicity of the

---

[2] We note that our LSTM results are better than those in Linzen et al. (2016). Also surprising is that the language model objective yields higher accuracies than the number prediction objective. We believe this may be due to better model optimization and to the embedding-output layer weight sharing, but we leave a thorough investigation to future work.

[3] `https://github.com/sleepinyourhat/vector-entailment`

language, this task is not trivial. To correctly classify logical relations, the model must learn nested structures as well as the scope of logical operators. We verify the difficulty of the task by training three bag-of-words models followed by sum/average/max-pooling. The best of the three models achieve less than 59% accuracy on the logical inference versus 77% on the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015a). This shows that the SNLI task can be largely solved by exploiting shallow features without understanding the underlying linguistic structures, which has also been pointed out by recent work (Glockner et al., 2018; Gururangan et al., 2018).

Concurrently to our work Evans et al. (2018) proposed an alternative data set for logical inference and also found that a FAN model underperformed various other architectures including LSTMs.

5.1 Models

We follow the general architecture proposed in (Bowman et al., 2015b): Premise and hypothesis sentences are encoded by fixed-size vectors. These two vectors are then concatenated and fed to a 3-layer feed-forward neural network with ReLU non-linearities to perform 7-way classification of the logical relation.

The LSTM architecture used in this experiment is similar to that of Bowman et al. (2015b). We simply take the last hidden state of the top LSTM layer as a fixed-size vector representation of the sentence. Here, we use a 2-layer LSTM with skip connections. The FAN maps a sentence x of length n to $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_n] \in \mathbb{R}^{d \times n}$. To obtain a fixed-size representation $\mathbf{z}$, we use a self-attention layer with two trainable queries $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^{1 \times d}$:

$$\mathbf{z}_i = \text{softmax}\left(\frac{\mathbf{q}_i \mathbf{H}}{\sqrt{d}}\right) \mathbf{H}^\top \quad i \in \{1, 2\}$$

$$\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2]$$

We find the best hyperparameters for each model by running a grid search as explained in §4.

5.2 Results

Following the experimental protocol of Bowman et al. (2015b), the data is divided into 13 bins based on the number of logical operators. Both FANs and LSTMs are trained on samples with at most n logical operators and tested on all bins. Figure 4 shows the result of the experiments with $n \leq 6$ and $n \leq 12$. We see that FANs and LSTMs perform similarly when trained on the whole dataset (Figure 4a). However when trained on a subset of the data (Figure 4b), LSTMs obtain better accuracies on similar examples ($n \leq 6$) and generalize better on longer examples ($6 < n \leq 12$).

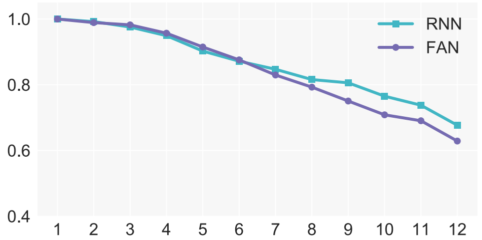

(a) $n \leq 12$

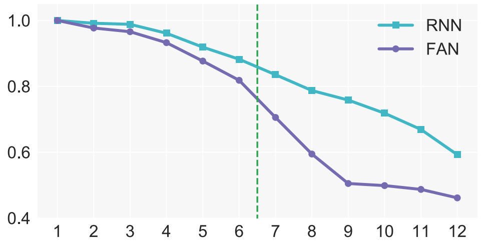

(b) $n \leq 6$

Figure 4: Results of logical inference when training on all data (a) or only on samples with at most n logical operators (b).

6 Discussion and Conclusion

We have compared a recurrent architecture (LSTM) to a non-recurrent one (FAN) with respect to the ability of capturing the underlying hierarchical structure of sequential data. Our experiments show that LSTMs slightly but consistently outperform FANs. We found that LSTMs are notably more robust with respect to the presence of misleading features in the agreement task, whether trained with explicit supervision or with a general language model objective. Secondly, we found that LSTMs generalize better than FANs to longer sequences in a logical inference task. These findings suggest that recurrency is a key model property which should not be sacrificed for efficiency when hierarchical structure matters for the task.

This does not imply that LSTMs should always be preferred over non-recurrent architectures. In fact, both FAN- and CNN-based networks have proved to perform comparably or better than LSTM-based ones on a very complex task like machine translation (Gehring et al., 2017; Vaswani et al., 2017). Nevertheless, we believe that the ability of capturing hierarchical information in sequential data remains a fundamental need for building intelligent systems that can understand and process language. Thus we hope that our insights will be useful towards building the next generation of neural networks.

Acknowledgments

This research was funded in part by the Netherlands Organization for Scientific Research (NWO) under project numbers 639.022.213, 612.001.218, and 639.021.646.

References

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep rnns encode soft hierarchical syntax. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 14–19, Melbourne, Australia. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015b. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches - Volume 1583*, COCO'15, pages 37–42.

Morten H. Christiansen and Nick Chater. 2016. The now-or-never bottleneck: A fundamental constraint on language. *Behavioral and Brain Sciences*, 39.

Hannah Cornish, Rick Dale, Simon Kirby, and Morten H Christiansen. 2017. Sequence memory constraints give rise to language-like structure through iterated learning. *PloS one*, 12(1):e0168532.

Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. 2018. Can neural networks understand logical entailment? In *ICLR*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. *ICLR*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, San Diego, CA, USA.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 321–331, San Diego, California. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Joint Learning for Targeted Sentiment Analysis

Dehong Ma[†], **Sujian Li**[†] **and Houfeng Wang**[†‡]
[†]MOE Key Lab of Computational Linguistics, Peking University, Beijing, 100871, China
[‡]Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, 221009, China
{madehong,lisujian,wanghf}@pku.edu.cn

Abstract

Targeted sentiment analysis (TSA) aims at extracting targets and classifying their sentiment classes. Previous works only exploit word embeddings as features and do not explore more potentials of neural networks when jointly learning the two tasks. In this paper, we carefully design the hierarchical multi-layer bidirectional gated recurrent units (HMBi-GRU) model to learn abstract features for both tasks, and we propose a HMBi-GRU based joint model which allows the target label of word to have influence on its sentiment label. Experimental results on two datasets show that our joint learning model can outperform other baselines and demonstrate the effectiveness of HMBi-GRU in learning abstract features.

1 Introduction

Targeted sentiment analysis (TSA) aims to extract targets in a text and simultaneously predict their sentiment classes (Hu and Liu, 2004; Jin et al., 2009; Li et al., 2010; Yang and Cardie, 2013). For example, given a sentence *"ESPN poll says Michael Jordan is the greatest basketball athlete"*, the targets are *ESPN* and *Michael Jordan* and their sentiment classes are *Neutral* and *Positive* respectively.

Targeted sentiment analysis can be seen as two tasks: target extraction and sentiment classification. Some researchers have tackled two tasks separately, e.g., target extraction (Liu et al., 2013; Wang et al., 2016a; Yin et al., 2016) and sentiment classification (Tang et al., 2016; Wang et al., 2016b; Ruder et al., 2016). Recently, some researches have attempted to conduct the two tasks jointly and generally see them as sequence labeling problems, where the *B/I/O* labels indicate target boundaries and the *Positive/Neutral/Negative* labels denote sentiment classes (Klinger and Cimiano, 2013; Yang and Cardie, 2013). Mitchell et al. (2013) explore labeling targets and their sentiment classes simultaneously by using the Conditional Random Fields (CRF) approach with traditional manual discrete features, and present three models: pipeline, joint and collapsed, according to different labeling processes of the two tasks. They find that the pipeline method outperforms the joint model on tweet dataset. Further, Zhang et al. (2015) introduce word embedding representations into the CRF framework and find that it is beneficial to integrate word embeddings into handcraft features in TSA regardless of pipeline, joint or collapsed methods.

With the success of deep learning techniques, neural networks have demonstrated their capability of sequence labeling (Collobert et al., 2011; Pei et al., 2014; Chen et al., 2015). However, Zhang et al. (2015) only use word embeddings to enrich features without taking full advantages of neural networks' potential in automatically capturing important sequence labeling features like long distance dependencies and character-level features.

To make better use of neural networks to explore appropriate character-level features and high-level semantic features for the two tasks, we design a hierarchical multi-layer bidirectional gated recurrent units networks (HMBi-GRU) which uses a multi-layer Bi-GRU to automatically learn character features (e.g. capitalization, noun suffix, etc) on letter sequence and model long distance dependencies between words on the concatenation of word embedding and its character features. The learned character features can also address out-of-vocabulary word problems.

In above example, the target label and sentiment label for *Michael Jordon* are "B-Person, I-Person" and "B-Positive, I-Positive", we can see that the boundary information (B, I) of target label and sentiment label is consistent. From the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4737–4742
Brussels, Belgium, October 31 - November 4, 2018.

view of human, we should first predict the target label and give corresponding sentiment label afterwards. Therefore, we introduce target label information into predicting sentiment label. In this way, our model can know about the target boundary information when predicting the sentiment label. Meanwhile, we also introduce transition matrix (Collobert et al., 2011) to model the dependencies between labels.

We conduct experiments on two datasets, and the performances show that our models outperform other baselines. This verifies the effectiveness of neural networks in TSA. In the experiments, we find that the target label information is important for predicting sentiment label. We also analyze the performance of multi-layer Bi-GRU and hierarchical architecture in learning character features and dependencies between words.

2 Model

We will detailedly introduce our model in this section, and our model is shown in Figure 1. Supposing that a sentence is composed of n words $[w_1, w_2, ..., w_n]$. For each word w_i consists of l_i characters $[c_1, c_2, ..., c_{l_i}]$ and l_i is the length of w_i. We embed all words and characters into low-dimensional real-value vectors which can be learned by language model (Bengio et al., 2003; Mikolov et al., 2013). We represent sentence as a matrix of word embeddings $W = [E_1, E_2, ..., E_n] \in R^{n \times d_w}$. Similarly, word w_i is denoted as a matrix of character embeddings $C_i \in R^{l_i \times d_c}$, and d_w and d_c are the size of word embedding and character embedding respectively.

First, we design a hierarchical two-layer architecture where each layer includes a multi-layer bidirectional Gated Recurrent Units (MBi-GRU). GRU is good at modeling a sequence with the benefits of avoiding the gradient vanishing and exploding problems. For a MBi-GRU, supposing that it has M layers of Bi-GRU, the hidden state on layer $m \in \{1, 2, ..., m\}$ at time $t \in \{1, 2, ..., n\}$ is recursively computed by:

$$h_t^m = \text{BiGRU}(h_t^{m-1}, h_{t-1}^m). \tag{1}$$

where the superscript of h denotes the corresponding layer of a MBi-GRU, and h^0 means the original inputs. BiGRU is bidirectional GRU which is defined as:

$$\text{BiGRU}(x_t, h_{t-1}) = \overrightarrow{h_t} \oplus \overleftarrow{h_t}; \tag{2}$$

$$\overrightarrow{h_t} = \text{GRU}(x_t, \overrightarrow{h_{t-1}}); \tag{3}$$

$$\overleftarrow{h_t} = \text{GRU}(x_t, \overleftarrow{h_{t-1}}). \tag{4}$$

where x_t is inputs which can be word embeddings or the hidden states of other BiGRU. $\oplus$ indicates the operation of concatenating two vectors.

With the matrix of character embeddings C_i as inputs, we utilize a MBi-GRU to learn character-level abstract features for word w_i based on its character embeddings. Through MBi-GRU, we can obtain the hidden states $[h_1^M, h_2^M, ..., h_{l_i}^M]$ on which a max-pooling operation is applied to output the character-level features $r_i \in R^{2d_c}$ for word w_i. The character features of all words in a sentence form a new matrix $C \in R^{n \times 2d_c}$. Next, We concatenate C with the matrix of word embeddings W and denote the concatenation as $F \in R^{n \times (d_w + 2d_c)}$. With F as input, We utilize another MBi-GRU to learn the hidden states $H = [h'^M_1, h'^M_2, ..., h'^M_n]$ as the final representations of the sentence. Therefore, the hierarchical two-layer MBi-GRU architecture can learn high-level abstract features with consideration of both character-level and word-level information.

After learning the final representations for sentence, we first project the features: $tf_i = h'^M_i$ of each word into target label space by:

$$y_t^i = f(tf_i \cdot W_p^t + b_p^t) \tag{5}$$

where W_p^t and b_p^t are weight matrix and bias.

As we know, the boundary of a target should be the same as that of its sentiment in sequence label. As the example in Section 1, the target label and sentiment label of *Michael Jordan* are "**B**-Person, **I**-Person" and "**B**-Positive, **I**-Positive" respectively. To learn this kind of consistency, we introduce the target label information into predicting sentiment label by:

$$y_s^i = f(sf_i \cdot W_p^s + b_p^s) \tag{6}$$

where $sf_i = h'^M_i \oplus y_t^i$, W_s^t and b_s^t are weight matrix and bias respectively. This makes our model know the target label information when predicting their sentiment.

For sequence labeling, there usually exist dependencies between labels. Take the target labeling task for example, label *I* will never follow label

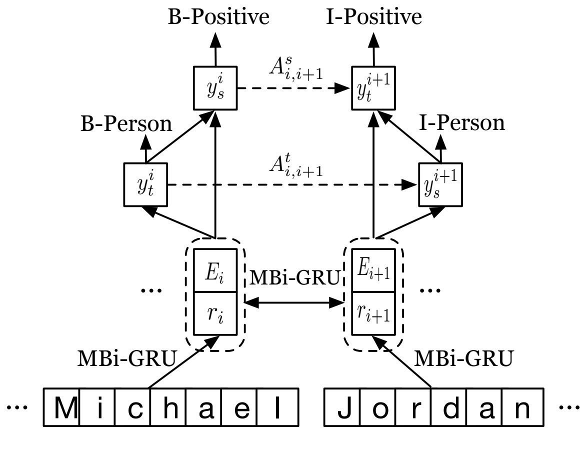

Figure 1: The overall architecture of our model.

O. To consider the influence of label dependencies, we introduce the transition matrix $A_{i,j}$ proposed by Collobert et al. (2011) which measures the probability of jumping from label i to label j.

Given the sentence $x = [w_1, w_2, ..., w_n]$ and the scores $y_t = [y_t^1, y_t^2, ..., y_t^n]$ and $y_s = [y_s^1, y_s^2, ..., y_s^n]$ computed by Eq. 5 and Eq. 6, we get the target labeling scores by summing up transition scores and the scores y_t^i:

$$s(y_t, x, \theta_t) = \sum\nolimits_{i=1}^{n} (A^t_{i-1,i} + y_t^i); \qquad (7)$$

where A^t is label transition matrix for target labeling. $\theta_t = \theta \cup \{A^t_{i,j}\}$, and θ denotes parameters of HMBi-GRUs.

Next, we normalize the target label scores over all possible labeling paths of target (i.e., Y_t) by a softmax function:

$$p_t(y_t|x) = \frac{e^{s(y_t,x,\theta_t)}}{\sum_{\hat{y}_t \in Y_t} e^{s(\hat{y}_t,x,\theta_t)}}; \qquad (8)$$

We can also use Eq. 7 and Eq. 8 to get the normalized sentiment label scores $p_s(y_s|x)$. To train our model, we define the loss function by:

$$\text{loss} = -\log(p_t(y_t|x)) - \log(p_s(y_s|x)). \qquad (9)$$

Finally, we obtain targets label sequence y_t^* and their sentiment label sequence y_s^* which have maximal score $y_t^* = \arg\max_{\hat{y} \in Y_t}(s(x, \hat{y}, \theta_t))$ $y_s^* = \arg\max_{\hat{y} \in Y_s}(s(x, \hat{y}, \theta_s))$. y_t^* and y_s^* can be computed by *Viterbi* algorithm.

3 Experiments

3.1 Setup

To validate the effectiveness of our model, we conduct experiments on two datasets, consisting of English tweets and Spanish tweets, which are constructed by Mitchell et al. (2013)[1]. Table 2 depicts the statistics of data, which contains sentence number, target number and the number of positive target, negative target and neutral target. To evaluate the system performance, we adopt *Precision*, *Recall* and *F-measure*. In our experiments, we evaluate the performance of detecting targets (DT) and targeted sentiment analysis (TSA) which a target is taken as correct only when the boundary and the sentiment are both correctly recognized. We also adopt *Precision*, *Recall* and *F-measure* used in Zhang et al. (2015) to evaluate our model. The reason why we don't compare with Mitchell et al. (2013) is that they only evaluate the beginning of targets along with the sentiment expressed towards it.

Datasets	#Sent	#Target	#Pos	#Neg	#Neu
English	2350	3288	707	275	2306
Spanish	5145	6658	1555	1007	4096

Table 1: Statistics of Datasets.

In our experiments, we use embeddings from Pennington et al. (2014)[2] and Cieliebak et al. (2017)[3] for English words and Spanish words respectively. The character embeddings are initialized by Xavier (Glorot and Bengio, 2010) and their dimension is 50. In our model, all unknown words, weight matrices and biases are initialized by Xavier Glorot and Bengio (2010). The dimensions of the character-level and word-level hidden states in MBi-GRU are set to 300 and 600 respectively. The layer number of multi-layer bidirectional GRU is set to 2. To avoid overfitting, we adopt dropout on embeddings, sf_i and tf_i, and the dropout rate is set to 0.5. The word embeddings and character embeddings will be tuned during training. Finally, we utilize Adam (Kingma and Ba, 2014) to optimize all parameters of our model.

3.2 Baselines

To investigate the performance of our joint model, we compare it with several baselines as follows:

- **Discrete** uses traditional discrete features as

---

[1] `http://www.m-mitchell.com/code/index.html`

[2] `https://nlp.stanford.edu/projects/glove/`

[3] `https://spinningbytes.com/resources/embeddings/`

Model	English						Spanish					
	DT			TSA			DT			TSA		
	P	R	F	P	R	F	P	R	F	P	R	F
Discrete	59.55	34.06	43.30	43.09	24.67	31.35	71.08	47.56	56.96	46.36	31.02	37.15
Neural	54.45	42.12	47.17	37.55	28.95	32.45	65.05	47.79	55.07	40.28	29.58	34.09
Integrate	**61.47**	49.28	54.59	44.62	35.84	39.67	**71.32**	61.11	65.74	46.67	39.99	43.02
Bi-GRU	58.13	43.46	49.62	45.76	32.29	37.73	65.24	53.02	58.45	46.33	37.50	41.45
MBi-GRU	58.27	49.01	53.24	45.80	35.21	39.81	66.14	60.07	62.95	45.61	40.04	42.64
HBi-GRU	57.24	**53.88**	55.41	44.94	38.60	41.52	68.24	61.81	64.82	46.53	42.21	44.18
No-Target	61.24	52.44	56.39	45.90	39.21	42.21	66.72	63.57	65.10	45.06	43.31	44.17
OURS	60.12	53.68	**56.98**	**46.52**	**39.99**	**42.87**	68.64	**63.66**	**66.01**	**48.09**	**43.44**	**45.61**

Table 2: Performance comparison of our models with the baselines.

inputs and multi-label CRF which contains two separate output clique potentials and two separate edge clique potentials for target extraction and sentiment classification respectively. There also exist links between target labels and sentiment labels for each word (Zhang et al., 2015).

- **Neural** uses word embeddings transformed with non-linear function as inputs, and others are the same as *Discrete* model (Zhang et al., 2015).

- **Integrated** integrates both discrete features and word embeddings into the same CRF framework and other settings are the same as *Discrete* (Zhang et al., 2015).

- **Bi-GRU** only uses word embeddings as inputs, and Bi-GRU is employed to learn representations for sentence.

- **MBi-GRU** also uses word embeddings as inputs, but MBi-GRU is utilized to model sentence.

- **HBi-GRU** first uses Bi-GRU to learn character level features for each word. Then, character level features and word embeddings are concatenated as inputs for another Bi-GRU to learn final representations for sentence.

- **No-Target** uses HMBi-GRU to learn representations for sentence, but h'^{M}_{i} (depicted in Section 2) are used to predict target label and sentiment label separately. *No-Target* doesn't let target label information to affect sentiment label. This is the biggest difference between *No-Target* and ours.

It is noticed that all of *Bi-GRU*, *MBi-GRU* and *HBi-GRU* use transition matrix to model the dependencies between labels and introduce target label information into predicting sentiment label.

3.3 Analysis

Table 2 displays the performance comparison of our models with the baselines. We can see that *Discrete* gets the worst results on English dataset, and *Neural* gets the worst results on Spanish dataset. The *Integrate* greatly improves the performances on both datasets because discrete features and word embeddings can complement each other.

Bi-GRU greatly improves the performance compared with *Discrete* and *Neural* but gets worse performance than *Integrate*. This verifies the effectiveness of neural networks in TAS. However, simple neural networks are not enough to acquire better results. *MBi-GRU* learns high-level features via multi-layer bidirectional GRU and achieves comparable results compared with *Integrate*.

Nevertheless, *Bi-GRU* and *MBi-GRU* do not make full use of character-level features. *HBi-GRU* incorporates character-level features by Bi-GRU on letter sequence of word. We can see that *HBi-GRU* improves about 1.85% and 1.16% in TSA on both datasets compared with *Integrate*. The performance of *HBi-GRU* demonstrates the importance of character-level features in TSA, and the hierarchical architecture is good at leaning multi-level (character-level, word-level) features.

Our model improves 3.20%, 2.59% in TSA and 2.39%, 0.27% in DT on both datasets compared with the existing best system: *Integrate*. Compared with *No-Target*, our model introduces target label information into predicting sentiment label and improves about 0.66%, 1.44% in TSA and 0.59%, 0.91% in DT on both datasets. The improvements demonstrate that target label information plays important roles in predicting sentiment label. It is noticed that the results of our model in

DT are also improved compared with *No-Target*. The reason may be that the gradients from sentiment loss have positive effects on detecting targets.

In a word, our model achieves state-of-the-art in DT and TSA on both datasets. Character-level features play great roles in DT and TSA, and HMBi-GRU is good at learning multi-level features. It is useful to learn boundary consistence by introducing target label information into predicting sentiment label.

3.4 Case Study

Here, we use a tweet from English Dataset as a case study, and the tweet is "Congratulations to our Champ Roger Federer ...". We apply *No-Target* and our model on the tweet. *No-Target* and our model get the same target labels: [O,O,O,O,B-Person,I-Person,...], and we can see that both models correctly extract the target: *Roger Federer*, and this results show the effectiveness of both models in detecting targets. Our model successfully obtains the correct sentiment labels: [O,O,O,O,B-Positive,I-Positive,...]. However, *No-Target* predicts a wrong sentiment label sequence: [O,O,O,B-Positive,I-Positive,O,...]. We can see that *No-Target* wrongly regard *Champ* as the beginning position and ignore *Federer*. The reasons are that the first letter of Champ is capitalized, which may mislead No-Target and there is no correlation between target and sentiment label. In our model, we incorporate target label information into predicting sentiment label. Therefore, our model tends to force target and sentiment label to have same boundary information.

This case study shows that the target label information plays important roles in predicting sentiment label because they share the same boundary information.

4 Related Work

Early works on target sentiment analysis were based on subjects and features. For example, Yi et al. (2003) extracted all references to the given subject and determined the sentiment of each reference. Hu and Liu (2004) first proposed several techniques to mine the product features that customers have expressed their opinions and determined their sentiment, and Popescu and Etzioni (2007) utilized unsupervised methods to identify opinions with respect to features and determine the polarity of opinions. Jin et al. (2009) proposed a novel lexicalized HMMs model to mine customer reviews of a product and extract highly specific product related entities which reviewers expressed their opinion, and they also identified the sentiment of opinion entities. The works of (Yang and Cardie, 2013) and (Li et al., 2010) are similar to (Jin et al., 2009). However, these works only take pre-defined features into account and can not find new features. To automatically extract targets and predict their sentiment, Mitchell et al. (2013) first proposed a conditional random fields (CRF) framework to jointly detect entities and identify their sentiment. Based on the work of (Mitchell et al., 2013), Zhang et al. (2015) explored the effect of word embeddings and automatic feature combinations by extending a CRF baseline using neural networks.

We propose a neural networks based joint model which extracts targets and their sentiments simultaneously. Our model takes full advantages of neural networks' potential in capturing sequence labeling features such as long distance dependencies and character-level features. Furthermore, Our model allows the target label to have positive effects on their sentiment label because target label shares boundary information with sentiment label.

5 Conclusion

In this paper, we propose a HMBi-GRU based joint model for targeted sentiment analysis. Our model will simultaneously extract targets and predict their sentiment. Furthermore, our model introduces target information into predicting corresponding sentiment label. Experiments show that the well-designed neural networks can greatly improve the result for targeted sentiment analysis, and target label information plays great roles in predicting sentiment label.

Acknowledgments

We thank anonymous reviewers for their insightful suggestions. Our work is supported by National Natural Science Foundation of China under Grant No.61433015 and the National Key Research and Development Program of China under Grant No.2017YFB1002101. The corresponding author of this paper is Houfeng Wang.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JMLR*, 3(Feb):1137–1155.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*, pages 1197–1206.

Mark Cieliebak, Jan Deriu, Dominic Egger, and Fatih Uzdilli. 2017. A twitter corpus and benchmark resources for german sentiment analysis. *SocialNLP*, page 45.

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

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *SIGKDD*, pages 168–177. ACM.

Wei Jin, Hung Hay Ho, and Rohini K Srihari. 2009. A novel lexicalized hmm-based learning framework for web opinion mining. In *ICML*, pages 465–472.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Roman Klinger and Philipp Cimiano. 2013. Bi-directional inter-dependencies of subjective expressions and targets and their value for a joint model. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 848–854.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *ACL*, pages 653–661.

Kang Liu, Heng Li Xu, Yang Liu, and Jun Zhao. 2013. Opinion target extraction using partially-supervised word alignment model. In *IJCAI*, volume 13, pages 2134–2140.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *ENMLP*, pages 1643–1654.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *ACL*, pages 293–303.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Ana-Maria Popescu and Orena Etzioni. 2007. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. A hierarchical model of reviews for aspect-based sentiment analysis. *arXiv preprint arXiv:1609.02745*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective lstms for target-dependent sentiment classification. In *International Conference on Computational Linguistics*, pages 3298–3307.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016a. Recursive neural conditional random fields for aspect-based sentiment analysis. *arXiv preprint arXiv:1603.06679*.

Yequan Wang, Minlie Huang, Li Zhao, and Xiaoyan Zhu. 2016b. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the conference on empirical methods in natural language processing*, pages 606–615.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1640–1649.

Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu, and Wayne Niblack. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 427–434. IEEE.

Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. *arXiv preprint arXiv:1605.07843*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *EMNLP*, pages 612–621.

Revisiting the Importance of Encoding Logic Rules in Sentiment Classification

Kalpesh Krishna♠♣ **Preethi Jyothi**♠ **Mohit Iyyer**♣

Indian Institute of Technology, Bombay♠
University of Massachusetts, Amherst♣
{kalpesh,miyyer}@cs.umass.edu
pjyothi@cse.iitb.ac.in

Abstract

We analyze the performance of different sentiment classification models on syntactically-complex inputs like *A-but-B* sentences. The first contribution of this analysis addresses reproducible research: to meaningfully compare different models, their accuracies must be averaged over far more random seeds than what has traditionally been reported. With proper averaging in place, we notice that the distillation model described in Hu et al. (2016), which incorporates explicit logic rules for sentiment classification, is ineffective. In contrast, using contextualized ELMo embeddings (Peters et al., 2018a) instead of logic rules yields significantly better performance. Additionally, we provide analysis and visualizations that demonstrate ELMo's ability to implicitly learn logic rules. Finally, a crowdsourced analysis reveals how ELMo outperforms baseline models even on sentences with ambiguous sentiment labels.

1 Introduction

In this paper, we explore the effectiveness of methods designed to improve sentiment classification (positive vs. negative) of sentences that contain complex syntactic structures. While simple bag-of-words or lexicon-based methods (Pang and Lee, 2005; Wang and Manning, 2012; Iyyer et al., 2015) achieve good performance on this task, they are unequipped to deal with syntactic structures that affect sentiment, such as contrastive conjunctions (i.e., sentences of the form *"A-but-B"*) or negations. Neural models that explicitly encode word order (Kim, 2014), syntax (Socher et al., 2013; Tai et al., 2015) and semantic features (Li et al., 2017) have been proposed with the aim of improving performance on these more complicated sentences. Recently, Hu et al. (2016) incorporate logical rules into a neural model and show that these rules increase the model's accuracy on sentences containing contrastive conjunctions, while Peters et al. (2018a) demonstrate increased overall accuracy on sentiment analysis by initializing a model with representations from a language model trained on millions of sentences.

In this work, we carry out an in-depth study of the effectiveness of the techniques in Hu et al. (2016) and Peters et al. (2018a) for sentiment classification of complex sentences. Part of our contribution is to identify an important gap in the methodology used in Hu et al. (2016) for performance measurement, which is addressed by averaging the experiments over several executions. With the averaging in place, we obtain three key findings: (1) the improvements in Hu et al. (2016) can almost entirely be attributed to just one of their two proposed mechanisms and are also less pronounced than previously reported; (2) contextualized word embeddings (Peters et al., 2018a) incorporate the "A-but-B" rules more effectively without explicitly programming for them; and (3) an analysis using crowdsourcing reveals a bigger picture where the errors in the automated systems have a striking correlation with the inherent sentiment-ambiguity in the data.

2 Logic Rules in Sentiment Classification

Here we briefly review background from Hu et al. (2016) to provide a foundation for our reanalysis in the next section. We focus on a logic rule for sentences containing an *"A-but-B"* structure (the only rule for which Hu et al. (2016) provide experimental results). Intuitively, the logic rule for such sentences is that the sentiment associated with the whole sentence should be the same as the sentiment associated with phrase *"B"*.[1]

---

[1] The rule is vacuously true if the sentence does not have this structure.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4743–4751
Brussels, Belgium, October 31 - November 4, 2018.

More formally, let $p_\theta(y|x)$ denote the probability assigned to the label $y \in \{+,-\}$ for an input x by the baseline model using parameters θ. A logic rule is (softly) encoded as a variable $r_\theta(x,y) \in [0,1]$ indicating how well labeling x with y satisfies the rule. For the case of *A-but-B* sentences, $r_\theta(x,y) = p_\theta(y|B)$ if x has the structure *A-but-B* (and 1 otherwise). Next, we discuss the two techniques from Hu et al. (2016) for incorporating rules into models: *projection*, which directly alters a trained model, and *distillation*, which progressively adjusts the loss function during training.

Projection. The first technique is to *project a trained model into a rule-regularized subspace*, in a fashion similar to Ganchev et al. (2010). More precisely, a given model p_θ is projected to a model q_θ defined by the optimum value of q in the following optimization problem:[2]

$$\min_{q,\xi\geq 0} \mathrm{KL}(q(X,Y)||p_\theta(X,Y)) + C\sum_{x\in X}\xi_x$$
$$\text{s.t. } \left(1 - \mathbb{E}_{y\leftarrow q(\cdot|x)}[r_\theta(x,y)]\right) \leq \xi_x$$

Here $q(X,Y)$ denotes the distribution of (x,y) when x is drawn uniformly from the set X and y is drawn according to $q(\cdot|x)$.

Iterative Rule Knowledge Distillation. The second technique is to transfer the domain knowledge encoded in the logic rules into a neural network's parameters. Following Hinton et al. (2015), a "student" model p_θ can learn from the "teacher" model q_θ, by using a loss function $\pi H(p_\theta, P_{\text{true}}) + (1-\pi)H(p_\theta, q_\theta)$ during training, where P_{true} denotes the distribution implied by the ground truth, $H(\cdot,\cdot)$ denotes the cross-entropy function, and π is a hyperparameter. Hu et al. (2016) computes q_θ after every gradient update by projecting the current p_θ, as described above. Note that both mechanisms can be combined: After fully training p_θ using the iterative distillation process above, the projection step can be applied one more time to obtain q_θ which is then used as the trained model.

Dataset. All of our experiments (as well as those in Hu et al. (2016)) use the SST2 dataset, a binarized subset of the popular Stanford Sentiment Treebank (SST) (Socher et al., 2013). The dataset includes phrase-level labels in addition to sentence-level labels (see Table 1 for detailed statistics); following Hu et al. (2016), we use both types of labels for the comparisons in Section 3.2. In all other experiments, we use only sentence-level labels, and our baseline model for all experiments is the CNN architecture from Kim (2014).

3 A Reanalysis

In this section we reanalyze the effectiveness of the techniques of Hu et al. (2016) and find that most of the performance gain is due to projection and not knowledge distillation. The discrepancy with the original analysis can be attributed to the relatively small dataset and the resulting variance across random initializations. We start by analyzing the baseline CNN by Kim (2014) to point out the need for an averaged analysis.

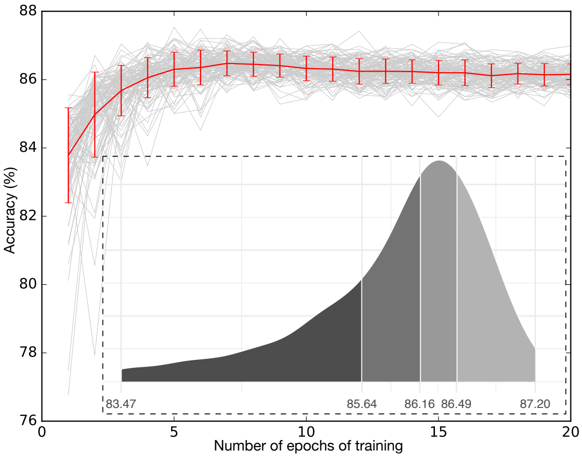

Figure 1: Variation in models trained on SST-2 (sentence-only). Accuracies of 100 randomly initialized models are plotted against the number of epochs of training (in gray), along with their average accuracies (in red, with 95% confidence interval error bars). The inset density plot shows the distribution of accuracies when trained with early stopping.

3.1 Importance of Averaging

We run the baseline CNN by Kim (2014) across 100 random seeds, training on sentence-level la-

Number of	Phrases	Train	Dev	Test
Instances	76961	6920	872	1821
A-but-B	3.5%	11.1%	11.5%	11.5%
Negations	2.0%	17.5%	18.3%	17.2%
Discourse	5.0%	24.6%	26.0%	24.5%

Table 1: Statistics of SST2 dataset. Here "Discourse" includes both *A-but-B* and negation sentences. The mean length of sentences is in terms of the word count.

---

[2] The formulation in Hu et al. (2016) includes another hyperparameter λ per rule, to control its relative importance; when there is only one rule, as in our case, this parameter can be absorbed into C.

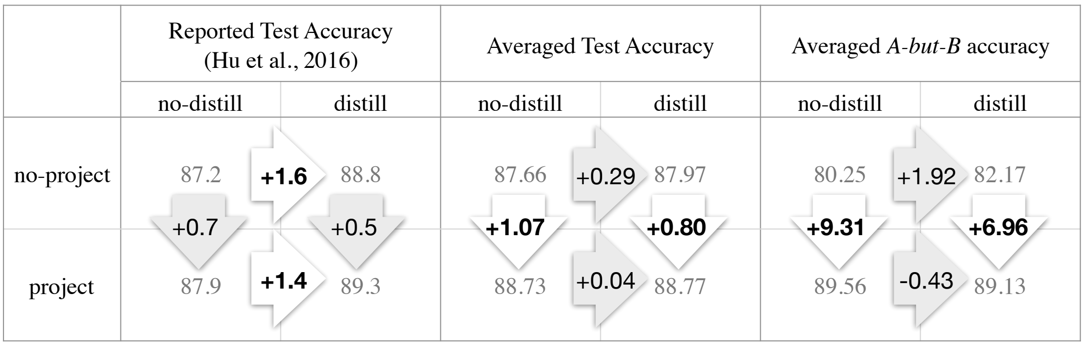

Figure 2: Comparison of the accuracy improvements reported in Hu et al. (2016) and those obtained by averaging over 100 random seeds. The last two columns show the (averaged) accuracy improvements for *A-but-B* style sentences. All models use the publicly available implementation of Hu et al. (2016) trained on phrase-level SST2 data.

bels. We observe a large amount of variation from run-to-run, which is unsurprising given the small dataset size. The inset density plot in Figure 1 shows the range of accuracies (83.47 to 87.20) along with 25, 50 and 75 percentiles.[3] The figure also shows how the variance persists even after the average converges: the accuracies of 100 models trained for 20 epochs each are plotted in gray, and their average is shown in red.

We conclude that, to be reproducible, only averaged accuracies should be reported in this task and dataset. This mirrors the conclusion from a detailed analysis by Reimers and Gurevych (2017) in the context of named entity recognition.

3.2 Performance of Hu et al. (2016)

We carry out an averaged analysis of the publicly available implementation[4] of Hu et al. (2016). Our analysis reveals that the reported performance of their two mechanisms (projection and distillation) is in fact affected by the high variability across random seeds. Our more robust averaged analysis yields a somewhat different conclusion of their effectiveness.

In Figure 2, the first two columns show the reported accuracies in Hu et al. (2016) for models trained with and without distillation (corresponding to using values $\pi = 1$ and $\pi = 0.95^t$ in the t^{th} epoch, respectively). The two rows show the results for models with and without a final projection into the rule-regularized space. We keep our hyper-parameters identical to Hu et al. (2016).[5]

The baseline system (no-project, no-distill) is identical to the system of Kim (2014). All the systems are trained on the phrase-level SST2 dataset with early stopping on the development set. The number inside each arrow indicates the improvement in accuracy by adding either the projection or the distillation component to the training algorithm. Note that the reported figures suggest that while both components help in improving accuracy, the distillation component is much *more* helpful than the projection component.

The next two columns, which show the results of repeating the above analysis after averaging over 100 random seeds, contradict this claim. The averaged figures show lower overall accuracy increases, and, more importantly, they attribute these improvements almost entirely to the projection component rather than the distillation component. To confirm this result, we repeat our averaged analysis restricted to only *"A-but-B"* sentences targeted by the rule (shown in the last two columns). We again observe that the effect of projection is pronounced, while distillation offers little or no advantage in comparison.

4 Contextualized Word Embeddings

Traditional context-independent word embeddings like `word2vec` (Mikolov et al., 2013) or `GloVe` (Pennington et al., 2014) are fixed vectors for every word in the vocabulary. In contrast, contextualized embeddings are dynamic representations, dependent on the current context of the word. We hypothesize that contextualized word embeddings might inherently capture these logic rules due to increasing the effective context size for the CNN layer in Kim (2014). Following the recent success of ELMo (Peters et al., 2018a) in sentiment analysis, we utilize the TensorFlow Hub implementation of ELMo[6] and feed these contextualized embeddings into our CNN model. We

---

[3] We use early stopping based on validation performance for all models in the density plot.

[4] `https://github.com/ZhitingHu/logicnn/`

[5] In particular, $C = 6$ for projection.

[6] `https://tfhub.dev/google/elmo/1`

fine-tune the ELMo LSTM weights along with the CNN weights on the downstream CNN task. As in Section 3, we check performance with and without the final projection into the rule-regularized space. We present our results in Table 2.

Switching to ELMo word embeddings improves performance by 2.9 percentage points on an average, corresponding to about 53 test sentences. Of these, about 32 sentences (60% of the improvement) correspond to *A-but-B* and negation style sentences, which is substantial when considering that only 24.5% of test sentences include these discourse relations (Table 1). As further evidence that ELMo helps on these specific constructions, the non-ELMo baseline model (no-project, no-distill) gets 255 sentences wrong in the test corpus on average, only 89 (34.8%) of which are *A-but-B* style or negations.

Statistical Significance: Using a two-sided Kolmogorov-Smirnov statistic (Massey Jr, 1951) with $\alpha = 0.001$ for the results in Table 2, we find that ELMo and projection each yield statistically significant improvements, but distillation does not. Also, with ELMo, projection is not significant. Specific comparisons have been added in the Appendix, in Table A3.

KL Divergence Analysis: We observe no significant gains by projecting a trained ELMo model into an *A-but-B* rule-regularized space, unlike the other models. We confirm that ELMo's predictions are much closer to the *A-but-B* rule's manifold than those of the other models by computing $\mathrm{KL}(q_\theta || p_\theta)$ where p_θ and q_θ are the original and projected distributions: Averaged across all *A-but-B* sentences and 100 seeds, this gives $0.27, 0.26$ and 0.13 for the Kim (2014), Hu et al. (2016) with distillation and ELMo systems respectively.

Intra-sentence Similarity: To understand the information captured by ELMo embeddings for *A-but-B* sentences, we measure the cosine similarity between the word vectors of every pair of words within the *A-but-B* sentence (Peters et al., 2018b). We compare the intra-sentence similarity for fine-tuned `word2vec` embeddings (baseline), ELMo embeddings without fine-tuning and finally fine-tuned ELMo embeddings in Figure 3. In the fine-tuned ELMo embeddings, we notice the words within the *A* and within the *B* part of the *A-but-B* sentence share the same part of the vector space. This pattern is less visible in the ELMo embeddings without fine-tuning and absent in the `word2vec` embeddings. This observation is indicative of ELMo's ability to learn specific rules for *A-but-B* sentences in sentiment classification. More intra-sentence similarity heatmaps for *A-but-B* sentences are in Figure A1.

Model		Test	*but*	*but* or *neg*
no-distill	no-project	85.98	78.69	80.13
no-distill	project	86.54	83.40	-
distill [7]	no-project	86.11	79.04	-
distill	project	86.62	83.32	-
ELMo	no-project	88.89	86.51	87.24
ELMo	project	88.96	87.20	-

Table 2: Average performance (across 100 seeds) of ELMo on the SST2 task. We show performance on *A-but-B* sentences ("*but*"), negations ("*neg*").

5 Crowdsourced Experiments

We conduct a crowdsourced analysis that reveals that SST2 data has significant levels of ambiguity even for human labelers. We discover that ELMo's performance improvements over the baseline are robust across varying levels of ambiguity, whereas the advantage of Hu et al. (2016) is reversed in sentences of low ambiguity (restricting to *A-but-B* style sentences).

Our crowdsourced experiment was conducted on Figure Eight.[8] *Nine* workers scored the sentiment of each *A-but-B* and negation sentence in the test SST2 split as 0 (negative), 0.5 (neutral) or 1 (positive). (SST originally had *three* crowdworkers choose a sentiment rating from 1 to 25 for every phrase.) More details regarding the crowd experiment's parameters have been provided in Appendix A.

We average the scores across all users for each sentence. Sentences with a score in the range $(x, 1]$ are marked as positive (where $x \in [0.5, 1)$), sentences in $[0, 1-x)$ marked as negative, and sentences in $[1-x, x]$ are marked as neutral. For instance, "*flat , but with a revelatory performance by michelle williams*" (score=0.56) is neutral when $x = 0.6$.[9] We present statistics of our dataset[10] in Table 3. Inter-annotator agree-

---

[7] Trained on sentences and not phrase-level labels for a fair comparison with baseline and ELMo, unlike Section 3.2.

[8] `https://www.figure-eight.com/`

[9] More examples of neutral sentences have been provided in the Appendix in Table A1, as well as a few "flipped" sentences receiving an average score opposite to their SST2 label (Table A2).

[10] The dataset along with source code can be found in

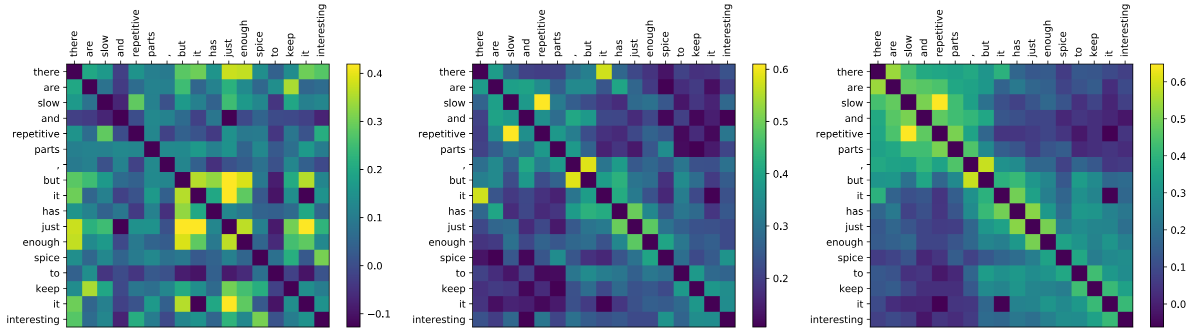

Figure 3: Heat map showing the cosine similarity between pairs of word vectors within a single sentence. The left figure has fine-tuned `word2vec` embeddings. The middle figure contains the original ELMo embeddings without any fine-tuning. The right figure contains fine-tuned ELMo embeddings. For better visualization, the cosine similarity between identical words has been set equal to the minimum value in the heat map.

ment was computed using Fleiss' Kappa (κ). As expected, inter-annotator agreement is higher for higher thresholds (less ambiguous sentences). According to Landis and Koch (1977), $\kappa \in (0.2, 0.4]$ corresponds to "fair agreement", whereas $\kappa \in (0.4, 0.6]$ corresponds to "moderate agreement".

We next compute the accuracy of our model for each threshold by removing the corresponding neutral sentences. Higher thresholds correspond to sets of less ambiguous sentences. Table 3 shows that ELMo's performance gains in Table 2 extends across all thresholds. In Figure 4 we compare all the models on the *A-but-B* sentences in this set. Across all thresholds, we notice trends similar to previous sections: 1) ELMo performs the best among all models on *A-but-B* style sentences, and projection results in only a slight improvement; 2) models in Hu et al. (2016) (with and without distillation) benefit considerably from projection; but 3) distillation offers little improvement (with or without projection). Also, as the ambiguity threshold increases, we see decreasing gains from projection on all models. In fact, beyond the 0.85 threshold, projection degrades the average performance, indicating that projection is useful for more ambiguous sentences.

Threshold	0.50	0.66	0.75	0.90
Neutral Sentiment	10	70	95	234
Flipped Sentiment	15	4	2	0
Fleiss' Kappa (κ)	0.38	0.42	0.44	0.58
no-distill, no-project	81.32	83.54	84.54	87.55
ELMo, no-project	87.56	90.00	91.31	93.14

Table 3: Number of sentences in the crowdsourced study (447 sentences) which got marked as neutral and which got the opposite of their labels in the SST2 dataset, using various thresholds. Inter-annotator agreement is computed using Fleiss' Kappa. Average accuracies of the baseline and ELMo (over 100 seeds) on non-neutral sentences are also shown.

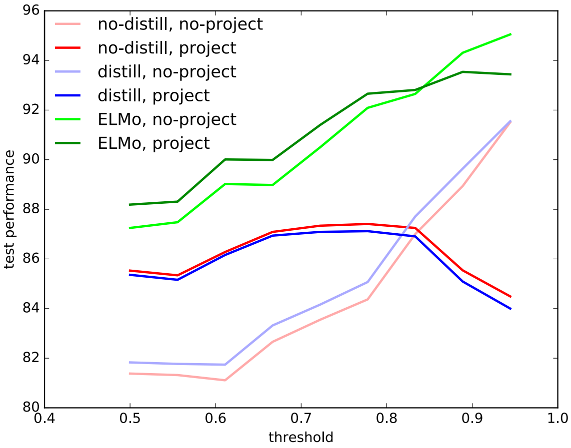

Figure 4: Average performance on the *A-but-B* part of the crowd-sourced dataset (210 sentences, 100 seeds)). For each threshold, only non-neutral sentences are used for evaluation.

6 Conclusion

We present an analysis comparing techniques for incorporating logic rules into sentiment classification systems. Our analysis included a meta-study highlighting the issue of stochasticity in performance across runs and the inherent ambiguity in the sentiment classification task itself, which was tackled using an averaged analysis and a crowdsourced experiment identifying ambiguous sentences. We present evidence that a recently proposed contextualized word embedding model (ELMo) (Peters et al., 2018a) implicitly learns logic rules for sentiment classification of complex sentences like *A-but-B* sentences. Future work includes a fine-grained quantitative study of ELMo word vectors for logically complex sentences along the lines of Peters et al. (2018b).

`https://github.com/martiansideofthemoon/logic-rules-sentiment`.

References

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Association for Computational Linguistics (ACL)*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics (ACL)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.

Shen Li, Zhe Zhao, Tao Liu, Renfen Hu, and Xiaoyong Du. 2017. Initializing convolutional filters with semantic features for text classification. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Association for Computational Linguistics (ACL)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

Matthew E. Peters, Mark Neumann, Wen tau Yih, and Luke Zettlemoyer. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Association for Computational Linguistics (ACL)*.

S. I. Wang and C. Manning. 2012. Baselines and bigrams: Simple, good sentiment and text classification. In *Association for Computational Linguistics (ACL)*.

Appendix

A Crowdsourcing Details

Crowd workers residing in five English-speaking countries (United States, United Kingdom, New Zealand, Australia and Canada) were hired. Each crowd worker had a Level 2 or higher rating on Figure Eight, which corresponds to a “group of more experienced, higher accuracy contributors”. Each contributor had to pass a test questionnaire to be eligible to take part in the experiment. Test questions were also hidden throughout the task and untrusted contributions were removed from the final dataset. For greater quality control, an upper limit of 75 judgments per contributor was enforced.

Crowd workers were paid a total of $1 for 50 judgments. An internal unpaid workforce (including the first and second author of the paper) of 7 contributors was used to speed up data collection.

# Judgments			Average	Sentence
Positive	Negative	Neutral		
1	1	7	0.50	the fight scenes are fun , but it grows tedious
3	2	4	0.56	it 's not exactly a gourmet meal but the fare is fair , even coming from the drive thru
2	3	4	0.44	propelled not by characters but by caricatures
4	2	3	0.61	not everything works , but the average is higher than in mary and most other recent comedies

Table A1: Examples of neutral sentences for a threshold of 0.66

# Judgments			Average	Original	Sentence
Positive	Negative	Neutral			
1	5	3	0.28	Positive	de niro and mcdormand give solid performances , but their screen time is sabotaged by the story 's inability to create interest
6	0	3	0.83	Negative	son of the bride may be a good half hour too long but comes replete with a flattering sense of mystery and quietness
0	5	4	0.22	Positive	wasabi is slight fare indeed , with the entire project having the feel of something tossed off quickly (like one of hubert 's punches) , but it should go down smoothly enough with popcorn

Table A2: Examples of flipped sentiment sentences, for a threshold of 0.66

Model 1		vs	Model 2		Significant
distill	no-project		distill	project	Yes
no-distill	no-project		no-distill	project	Yes
ELMo	no-project		ELMo	project	No
no-distill	no-project		distill	no-project	No
no-distill	project		distill	project	No
no-distill	no-project		ELMo	no-project	Yes
distill	no-project		ELMo	no-project	Yes
no-distill	project		ELMo	project	Yes
distill	project		ELMo	project	Yes

Table A3: Statistical significance using a two-sided Kolmogorov-Smirnov statistic (Massey Jr, 1951) with $\alpha = 0.001$.

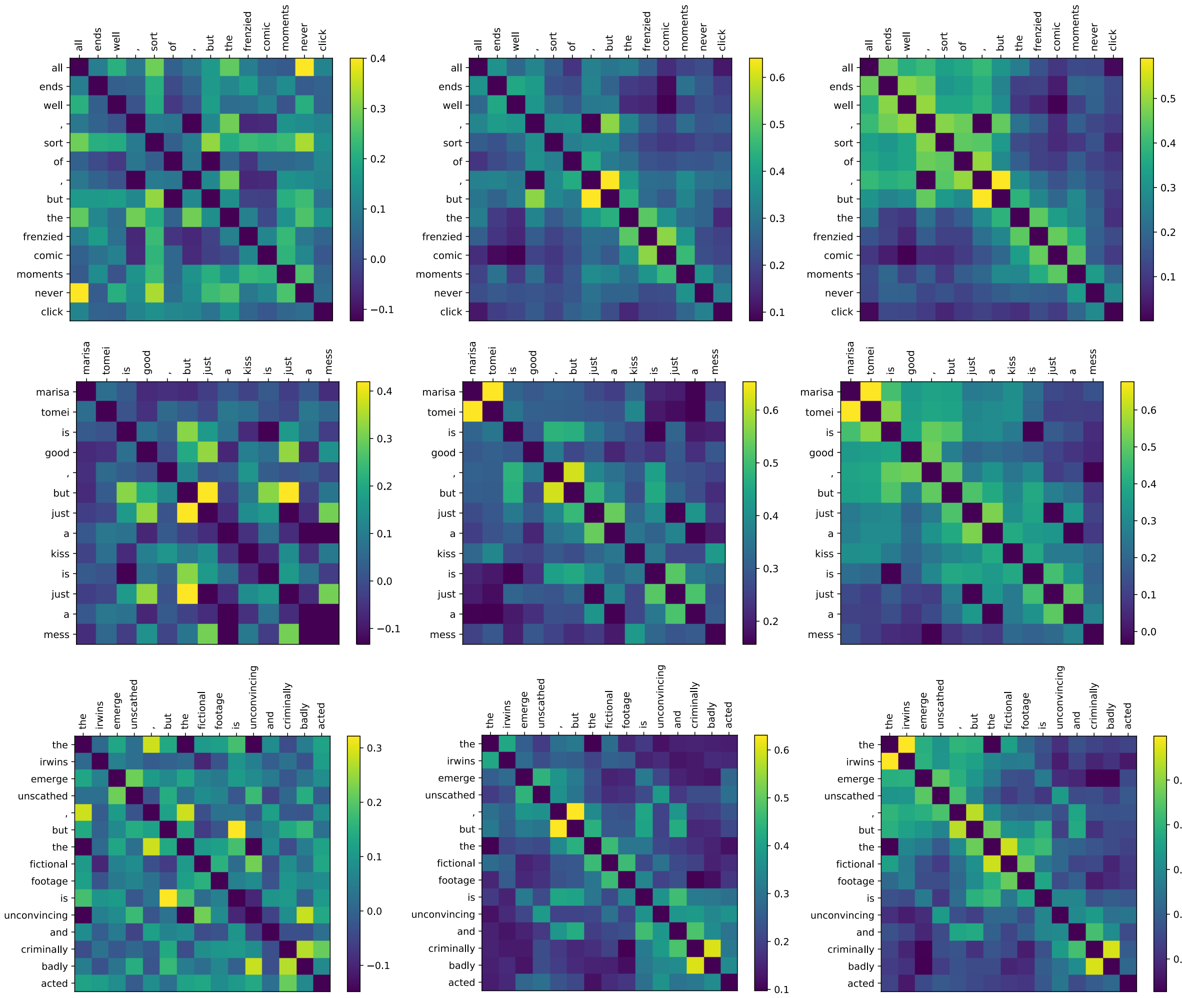

Figure A1: Heat map showing the cosine similarity between pairs of word vectors within a single sentence. The leftmost column has `word2vec` (Mikolov et al., 2013) embeddings, fine-tuned on the downstream task (SST2). The middle column contains the original ELMo embeddings (Peters et al., 2018a) without any fine-tuning. The representations from the three layers (token layer and two LSTM layers) have been averaged. The rightmost column contains ELMo embeddings fine-tuned on the downstream task. For better visualization, the cosine similarity between identical words has been set equal to the minimum value in the map.

A Co-Attention Neural Network Model for Emotion Cause Analysis with Emotional Context Awareness

Xiangju Li[1] **, Kaisong Song**[2] **, Shi Feng**[1] **, Daling Wang**[1] **, Yifei Zhang**[1]
[1]School of Computer Science and Engineering, Northeastern University, Shenyang, China
lixiangju100@163.com, (fengshi, wangdaling, zhangyifei)@cse.neu.edu.cn
[2]Alibaba Group, Hangzhou, China kaisong.sks@alibaba-inc.com

Abstract

Emotion cause analysis has been a key topic in natural language processing. Existing methods ignore the contexts around the emotion word which can provide an emotion cause clue. Meanwhile, the clauses in a document play different roles on stimulating a certain emotion, depending on their content relevance. Therefore, we propose a co-attention neural network model for emotion cause analysis with emotional context awareness. The method encodes the clauses with a co-attention based bi-directional long short-term memory into high-level input representations, which are further fed into a convolutional layer for emotion cause analysis. Experimental results show that our approach outperforms the state-of-the-art baseline methods.

1 Introduction

Emotion cause analysis (ECA) aims to identify the reasons behind a certain emotion expressed in text. Compared with emotion classification (Gao et al., 2013; Song et al., 2016, 2017; Feng et al., 2018), ECA is more challenging because it requires deep comprehension of text semantics and then extracts the cause of emotions accurately. Recently, Gui et al. (2017) considered ECA as a question answer problem and proposed a method which took a query (i.e., a single emotion word) as the input of a QA system. However, their method ignores an important fact that the context around the emotion word details the emotions and serves as an emotion cause clue. Therefore, the emotion clause consisting of the emotion word and its context should be treated as a whole unit for query.

Example 1 *In 2016* (c_1)*, Lakers' Kobe announced his retirement* (c_2)*. In the same year* (c_3)*, Spurs' Duncan also retired* (c_4)*. The Spurs' athletes and Duncan's fans are frustrated* (c_5)*.*

For example, a microblog user published a post in Twitter (see Example 1). From Example 1, we can find that the clause c_5 is the emotion clause which contains the emotion word "*frustrated*" and the clause c_4 is the cause clause of the emotion. However, it is difficult for previous works (Gui et al., 2017) to extract the expected emotion causes by only taking the emotion word "*frustrated*" as a query. In other words, the retirement of Kobe can also stimulate "*frustrated*" emotion, but it is not the cause of the clause c_5. Therefore, the context around the emotion word in the same emotion clause can serve as a useful emotion cause clue for the ECA task. In addition, the words shared by the emotion cause clause and the emotion clause (e.g., "*Spurs*" and "*Duncan*") usually reflect stronger emotion-cause associations and should be paid more attention when modeling.

In this paper, we propose a novel, extensive and effective Co-Attention Neural Network method (CANN) for this ECA task. The method incorporates the contextual information by a bi-directional long short-term memory encoder. Then, an attention matrix is constructed to model the mutual impacts of each candidate cause clause and the whole emotion clause. Meanwhile, the above clauses are paid different attention based on the transformed attention matrix. Finally, a convolutional neural network is utilized for classification. To sum up, the main contributions of our paper are three-fold: (1) We present a novel Co-Attention Neural Network model (CANN), which identifies the emotion cause from subjective texts by combining a co-attention mechanism, bi-directional long short-term memory units and convolutional neural network into a unified framework. (2) A co-attention mechanism is designed to capture the mutual impacts between each candidate cause clause and the emotion clause for enhancing the input representations of a neural network model. (3) Our

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4752–4757
Brussels, Belgium, October 31 - November 4, 2018.

comparative results on a public benchmark dataset show that CANN outperforms several state-of-the-art methods, which verifies the effectiveness of our proposed method.

2 Related Work

A variety of learning methods have been applied to emotion cause analysis and achieved plausible improvements since Lee et al. (2010) first gave the formal definition of this task and constructed a public available Chinese emotion cause annotated corpus based on manual annotation scheme. Chen et al. (2010) developed two sets of linguistic features based on linguistic cues and proposed a rule-based approach. Similarly, Gui et al. (2014) also proposed a rule based emotion cause detection method by designing 25 manually complied rules for this task and Gao et al. (2015) extended this method to deal with informal text such as microblog. Ghazi et al. (2015) built a Conditional Random Fields (CRF) learner to detect the emotion stimulus. However, this model is based on a simplistic assumption that emotion cause and emotion word co-exist in the same sentence, which limits its application scenario. More recently, Gui et al. (2016b) proposed a multi-kernel based method trained from a public SINA city news ECA Corpora. However, this method is still heavily dependent on the design of effective features.

With the apparent success of neural network methods and attention mechanisms, Gui et al. (2017) proposed a novel deep neural networks model which considers emotion cause identification as a question answering task. However, their method only takes a single emotion word as the query, which ignores important emotion-cause clues. Meanwhile, their method fails to utilize the mutual influences between each candidate cause clause and the emotion clause, which will depress the model performance. To address these issues, we propose a novel CANN model which not only considers the whole emotion clause as a query but also models the mutual influences between the candidate cause clause and the whole emotion clause via co-attention mechanism.

3 Methodology

In this section, we will first give the formal definition of the emotion cause analysis task. Then, we introduce our proposed method in detail.

3.1 The Definition of Emotion Cause Analysis

Given a document $D = \{c_1, c_2, \cdots, c_n\}$ about an emotion cause event, the goal of this ECA task is to identify which clause stimulates the emotion expression. For convenience, we suppose each document has a unique emotion clause (containing a single emotion word) and at least a corresponding emotion cause clause. Previous work considers emotion cause analysis as a question answering task by taking a single emotion word as a query to retrieve the emotion cause (Gui et al., 2017). This is not applicable to our observation that context around the emotion word in the same clause can serve as an important emotion-cause clue. Here, we are intended to take the whole emotion clause (i.e., the emotion word and its context words) as a query. Therefore, the emotion cause analysis task can be formulated as below:

$$P_x = \text{emotion-cause}\left(c^q, c^a\right)$$

where c^q is an emotion clause (i.e., query) and c^a is a candidate emotion cause clause (i.e., candidate answer)[1]. The probability P_x of identifying the clause c^a as the emotion cause of the emotion clause c^q is computed by the emotion-cause function ($x = 1$ if c^a stimulates the emotion expressed in the c^q, or $x = 0$ otherwise).

3.2 Co-Attention Neural Network for Emotion Cause Analysis

In this section, we propose a Co-Attention Neural Network (CANN) method, which contains four different layers: (1) an embedding layer for the input words; (2) a bi-directional long short-term memory (Bi-LSTM) layer; (3) a co-attention layer for generating the attention maps of the emotion clause and the candidate cause clause; and (4) a convolutional neural network layer to generate a probability distribution over the output classes. Figure 1 illustrates the details of our CANN.

(1) The embedding layer maps each word w_i into a d-dimensional vector $v_i \in \mathbb{R}^d$. Then, the clause c^q can be represented as a feature map $E^q = [v_1^q, v_2^q, ..., v_n^q]$. Symmetrically, we can also produce E^a for the emotion cause clause c^a.

(2) Bi-LSTM layer aims to encode contextual information with a Bi-LSTM (Wang et al., 2016). Let v_t^q be the t-th word vectors of the clause c^q. Each contextual embedding at time-step t of c^q can be formulated as $h_t^q = concat(\overrightarrow{h}_t^q, \overleftarrow{h}_t^q)$, where

---

[1] Note that the emotion clause c^q and the emotion cause clause c^a can be the same clause.

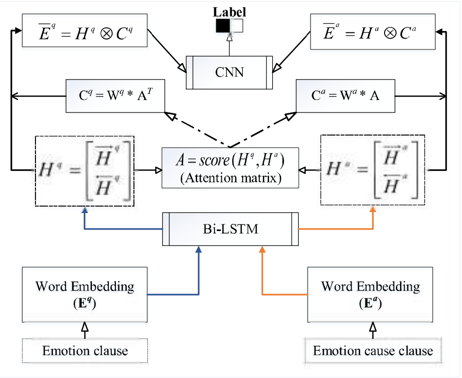

Figure 1: CANN architecture for ECA task.

$concat(\cdot,\cdot)$ is a concatenation operation, $\overrightarrow{h}_t^q = \overrightarrow{LSTM}(\overrightarrow{h}_{t-1}^q, v_t^q)$ and $\overleftarrow{h}_t^q = \overleftarrow{LSTM}(\overleftarrow{h}_{t+1}^q, v_t^q)$ are hidden states produced by a forward LSTM and a backward LSTM, respectively. Finally, a context embedding matrix $H^q \in \mathbb{R}^{n\times 2h}$ of the clause c^q can be obtained by concatenating all the context embeddings, where h is the number of hidden units. Similarly, we can also generate H^a for the emotion cause clause c^a.

(3) Co-attention layer contains three steps: construction of attention matrix, construction of attention feature maps and input transformation.

Firstly, we construct an attention matrix $A \in \mathbb{R}^{n\times n}$ which models mutual influences between the c^q and c^a via a measurement function: $A = score\big(H^q, H^a\big)$. Element $A_{i,j}$ represents the semantic similarity between the i-th word of the clause c^q and the j-th word of the c^a clause[2].

Next, two attention feature maps $C^q \in \mathbb{R}^{2h\times n}$ and $C^a \in \mathbb{R}^{2h\times n}$ can be derived from linearly transformed attention matrix A: $C^q = W^q \cdot A^\top$ and $C^a = W^a \cdot A$, where $W^q \in \mathbb{R}^{2h\times n}$ and $W^a \in \mathbb{R}^{2h\times n}$ are weight matrices.

Finally, we apply the attention feature maps to the original representations of c^q and c^a respectively: $\bar{E}^q = H^q \otimes C^q$, $\bar{E}^a = H^a \otimes C^a$.[3] They will be fed into CNN layer for future operations.

(4) In CNN layer, we applied three convolution filters with lengths 2, 3 and 4, each of which is followed by a max pooling operation to select most informative features. Two high-level feature vectors z^q and z^a are generated and then concatenated as the inputs of a fully interconnected softmax layer before classification.

Model Training: We use cross-entropy error between gold distribution and predicted distribution as the loss function. We use back-propagation to calculate the gradients of all the trainable parameters and update them with Adam (Kingma and Ba, 2015).

4 Experiments and Results

4.1 Dataset and Setup

Our experiments are conducted based on a Chinese emotion cause dataset[4]. The statistics are presented in Table 1. We evaluate the performance by precision (*P*), recall (*R*), and F-measure (*F*), which are commonly used evaluation metrics (Gui et al., 2016a). We set the number hidden units $h = 50$, the number of filters as $N = 100$, the dimension of word embedding $d = 30$, the initial learning rate $lr = 10^{-4}$, the dropout rate $p = 0.5$ and the batch size as 50 according to the best F value on the development set which is set aside 10% from the training set.

Item	Number	Item	Number
#Documents	2,105	#Cause$_1$	2,046
#Causes	2,167	#Cause$_2$	56
#Clauses	11,799	#Cause$_3$	3

Table 1: Statistics of the experimental dataset we used. Cause$_1$, Cause$_2$ and Cause$_3$ represent the documents with 1, 2 and 3 cause clauses, respectively.

4.2 Comparison of Different Methods

We compare CANN with some traditional and advanced baselines in Table 2, and separate results into three groups: (1) rule based or common-sense based methods; (2) machine learning methods; (3) deep neural network methods.

(1) **RB** is a rule based method, which generalizes two sets of linguistic rules for emotion cause analysis (Lee et al., 2010). **CB** is a knowledge based method (Russo et al., 2011), which uses the Chinese Emotion Cognition Lexicon (Xu et al., 2013) as the common-sense knowledge base.

(2) **RB+CB+ML** is a support vector machine (SVM) classifier trained on features including rules (Lee et al., 2010) and the Chinese Emotion Cognition Lexicon (Xu et al., 2013). **SVM** is a SVM classifier trained on unigrams, bigrams and

---

[2] function *score* is the cosine distance

[3] $\otimes$ denotes element-wise product.

[4] Dataset: http://hlt.hitsz.edu.cn/?page%20id=694

Methods	*P*	*R*	*F*
RB$^\diamond$	**0.6747**	0.4287	0.5243
CB$^\diamond$	0.2672	**0.7130**	0.3887
RB + CB$^\diamond$	0.5435	0.5307	**0.5370**
RB + CB + ML$^\diamond$	0.5921	0.5307	0.5597
SVM$^\diamond$	0.4200	0.4375	0.4285
Word2vec$^\diamond$	0.4301	0.4233	0.4136
Multi-kernel$^\diamond$	**0.6588**	**0.6927**	**0.6752**
CNN	0.5307	0.6427	0.5807
ConvMS-Memnet$^\diamond$	0.7076	0.6838	0.6955
CANN	**0.7721**	**0.6891**	**0.7266**

Table 2: Comparison among different methods. The results with superscript $\diamond$ are reported in (Gui et al., 2017). The best results in each group are highlighted.

trigrams features. **Word2vec** uses word representations obtained by Word2vec (Mikolov et al., 2013b) as features and then trains a SVM classifier. **Multi-kernel** (Gui et al., 2016a) uses the multi-kernel method to identify the emotion cause.

(3) **ConvMS-Memnet** is the current state-of-the-art method using the convolutional multiple-slot deep memory network to identify the emotion causes (Gui et al., 2017). **CNN** is the basic convolutional neural network proposed by (Kim, 2014). **CANN** is our co-attention neural network method.

From Group 1 in Table 2, we can observe that RB+CB outperforms RB and CB methods in F, which verifies that RB and CB are complementary to improve the model performance. In Group 2, Multi-kernel performs the best among the machine learning methods because it considers the structured context information, which plays an important role in this task. In Group 3, the best F value is achieved by our proposed CANN, which outperforms the state-of-the-art method ConvMS-Memnet by 3.11%. Meanwhile, the optimal values on P and R are also generated by CANN method. The results imply that considering context around the emotion word is helpful to capture useful emotion clues and detail context information. We can also find that CANN outperforms CNN method by 14.33% in F, which indicates that our co-attention mechanism has a significant positive effect for emotion cause identification by capturing the mutual impacts between the emotion clause (i.e., query) and each candidate cause clause (i.e., candidate answer).

4.3 Effects of Different CANN Components

We further validate the different roles of model components by comparing several sub-networks based on our proposed architectures: **Full** is our proposed CANN method which considers the whole emotion clause as a query unit. **Full-BCA** does not contain the Bi-LSTM encoder and the co-attention mechanism. **Full-CA** does not consider the co-attention mechanism. **Full-EA** only pays attention to the candidate cause clause.

Methods	*P*	*R*	*F*
Full-BCA	0.5307	0.6427	0.5807
Full-CA	0.7209	0.5405	0.6147
Full-EA	0.7714	0.6596	0.7100
Full	**0.7721**	**0.6891**	**0.7266**

Table 3: Effects of different CANN components.

From Table 3, we can find that all the partial configured models cannot compete with the **Full** model indicating that the co-attention mechanism is effective for the emotion cause analysis task. Meanwhile, the F value is improved remarkably when co-attention mechanism is considered. **Full-CA** performs better than **Full-BCA**, which verifies the effectiveness of Bi-LSTM in clause presentation. We also notice that **Full** gains nearly 12% improvements in F compared with **Full-CA** , which indicates that both emotion clause and emotion contribute to acquiring better performance.

4.4 Effects of Emotion Word and Context Words around Emotion Word

In this section, we further study the effects of context words around emotion word in the same emotion clause on the classification performance. **Full** is our proposed CANN method which considers the whole emotion clause as a query unit. **Full-emotion** does not consider the emotion word. **Full-context** only considers emotion word but ignores its context words. We compare above three different configurations of CANN in Table 4.

Methods	*P*	*R*	*F*
Full-emotion	0.6514	0.7203	0.6827
Full-context	0.5534	0.6815	0.6095
Full	**0.6939**	**0.7601**	**0.7240**

Table 4: Effects of emotion word and context words around emotion word.

As shown in Table 4, it is clear that the partial configurations cannot compete with the **Full** model indicating both emotion word and its context words are helpful. Specifically, emotion word can provide emotion clues directly and its context words can provide richer emotional details, both of which contribute to capturing intrinsic emotion causes. **Full-context** performs slightly better than **Full-emotion** because context words reflect the "skeleton" of emotion expression directly, such as "I feel so much _" where the whitespace "_" can be any emotion word (e.g., happy or sad). This verifies the advantages of context words.

4.5 Effects of Initialization Methods

We further study the influences of different initialization ways by applying different word embeddings into the input layer of our CANN method. **Random** initializes word embeddings randomly from a truncated standard normal distribution. **CBOW** and **Skip** represent the word embeddings which are pre-trained respectively by CBOW and Skip-Gram models (Mikolov et al., 2013a) based on the training set and then applied to parameter initialization.

Methods	*P*	*R*	*F*
Random	0.6519	0.5650	0.6022
CBOW	0.7207	**0.7245**	0.7226
Skip	**0.7721**	0.6891	**0.7266**

Table 5: Effects of different initialization methods.

As shown in Table 5, pre-trained initialization methods perform better than random initialization, which indicates that CANN is sensitive to the initial settings of the parameters. Meanwhile, **Skip** method performs slightly better than **CBOW** method by achieving best results in metrics P and F. Therefore, we adopt the pre-trained word embeddings generated by **Skip** method to initialize all the word vectors of our CANN in this work.

4.6 Case Study

In order to validate whether our CANN can focus on the words which describe the emotion causes. We select two clauses ($\mathbf{c_2}$ and $\mathbf{c_4}$) from Example 1 and visualize the attention feature map C^a in Figure 2. The horizontal coordinate represents the different dimensions of each word representation obtained by Bi-LSTM. The vertical coordinate represents the words of the clause. And the color depth represents different degrees of the attention weights of each word in emotion cause clause.

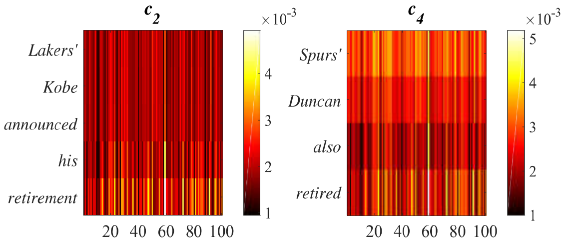

Figure 2: Visualization of Attention.

From Figure 2, we can easily find the word "*retirement*" in the clause $\mathbf{c_2}$ has much higher attention weights than other words in this emotion cause clause, which indicates that the cause is closely related to the word "*retirement*". In the clause $\mathbf{c_4}$, the words "*Spurs'*" and "*Duncan*" also have high attention weights expect "*retired*", which shows that the emotion cause is not only related to "*retirement*" and "*retired*" but also "*Spurs'*" and "*Duncan*". Above all, clause $\mathbf{c_4}$ is more likely to be identified as the emotion cause clause according to our method, which is consistent with our observation. We observe that words like *"Lakers'"*, *"his"* and *"also"* have low attention weights and they are irrelevant to the emotion word *"frustrated"*. These again verify the effectiveness of our proposed attention mechanism on the emotion cause analysis task.

5 Conclusions

This paper proposes an effective Co-Attention Neural Network to identify the emotion causes. The mutual impacts between the emotion clause and each candidate clause are modeled as an attention matrix which further transforms original inputs into high-level input features of a constitutional neural network for classification. Experimental results conducted on a real-word dataset shows that our method achieves substantial improvements against "state-of-the-art" baselines.

Acknowledgments

The work was supported by the National Key R&D Program of China under grant 2018YFB1004700, and National Natural Science Foundation of China (61772122, 61872074).

References

Ying Chen, Sophia Yat Mei Lee, Shoushan Li, and Churen Huang. 2010. Emotion cause detection with linguistic constructions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 179–187.

Shi Feng, Yaqi Wang, Kaisong Song, Daling Wang, and Ge Yu. 2018. Detecting multiple coexisting emotions in microblogs with convolutional neural networks. *Cognitive Computation*, 10(1):136–155.

Kai Gao, Hua Xu, and Jiushuo Wang. 2015. Emotion cause detection for chinese micro-blogs based on ecocc model. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 3–14.

Wei Gao, Shoushan Li, Sophia Yat Mei Lee, Guodong Zhou, and Churen Huang. 2013. Joint learning on sentiment and emotion classification. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1505–1508.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In *Computational Linguistics and Intelligent Text Processing - 16th International Conference*, pages 152–165.

Lin Gui, Jiannan Hu, Yulan He, Ruifeng Xu, Qin Lu, and Jiachen Du. 2017. A question answering approach to emotion cause extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1953–1602.

Lin Gui, Dongyin Wu, Ruifeng Xu, Qin Lu, and Yu Zhou. 2016a. Event-driven emotion cause extraction with corpus construction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1639–1649.

Lin Gui, Ruifeng Xu, Qin Lu, Dongyin Wu, and Yu Zhou. 2016b. Emotion cause extraction, a challenging task with corpus construction. In *Chinese National Conference on Social Media Processing*, pages 98–109.

Lin Gui, Li Yuan, Ruifeng Xu, Bin Liu, Qin Lu, and Yu Zhou. 2014. Emotion cause detection with linguistic construction in chinese weibo text. In *Natural Language Processing and Chinese Computing*, pages 457–464.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, pages 1–15.

Sophia Yat Mei Lee, Ying Chen, and Churen Huang. 2010. A text-driven rule-based system for emotion cause detection. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 45–53.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Irene Russo, Tommaso Caselli, Francesco Rubino, Ester Boldrini, and Patricio Martínez-Barco. 2011. Emocause: an easy-adaptable approach to emotion cause contexts. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 153–160.

Kaisong Song, Ling Chen, Wei Gao, Shi Feng, Daling Wang, and Chengqi Zhang. 2016. Persentiment: A personalized sentiment classification system for microblog users. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 255–258.

Kaisong Song, Wei Gao, Shi Feng, Daling Wang, Kamfai Wong, and Chengqi Zhang. 2017. Recommendation vs sentiment analysis: A text-driven latent factor model for rating prediction with cold-start awareness. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2744–2750.

Yang Wang, Shi Feng, Daling Wang, Yifei Zhang, and Ge Yu. 2016. Context-aware chinese microblog sentiment classification with bidirectional lstm. In *Asia-Pacific Web Conference*, pages 594–606.

Ruifeng Xu, Chengtian Zou, Yanzhen Zheng, Xu Jun, Lin Gui, Bin Liu, and Xiaolong Wang. 2013. A new emotion dictionary based on the distinguish of emotion expression and emotion cognition. *Journal of Chinese Information Processing*, 27(6):82–90.

Modeling Empathy and Distress in Reaction to News Stories

Sven Buechel *†3 **Anneke Buffone** *1 **Barry Slaff** 1 **Lyle Ungar** 1,2 **João Sedoc** 1,2

1 Positive Psychology Center, University of Pennsylvania
2 Computer & Information Science, University of Pennsylvania
3 JULIE Lab, Friedrich-Schiller-Universität Jena

`https://wwbp.org` 1,2 `https://julielab.de` 3

Abstract

Computational detection and understanding of empathy is an important factor in advancing human-computer interaction. Yet to date, text-based empathy prediction has the following major limitations: It underestimates the psychological complexity of the phenomenon, adheres to a weak notion of ground truth where empathic states are ascribed by third parties, and lacks a shared corpus. In contrast, this contribution presents the first publicly available gold standard for empathy prediction. It is constructed using a novel annotation methodology which reliably captures empathy assessments by the writer of a statement using multi-item scales. This is also the first computational work distinguishing between multiple forms of empathy, empathic concern, and personal distress, as recognized throughout psychology. Finally, we present experimental results for three different predictive models, of which a CNN performs the best.

1 Introduction

Over two decades after the seminal work by Picard (1997) the quest of *Affective Computing*, to ease the interaction with computers by giving them a sense of how emotions shape our perception and behavior, is still far from being fulfilled. Undoubtedly, major progress has been made in NLP, with sentiment analysis being one of the most vivid and productive areas in recent years (Liu, 2015).

However, the vast majority of contributions has focused on *polarity prediction*, typically only distinguishing between positive and negative feeling or evaluation, usually in social media postings or product reviews (Rosenthal et al., 2017; Socher et al., 2013). Only very recently, researchers started exploring more sophisticated models of human emotion on a larger scale (Wang et al., 2016; Abdul-Mageed and Ungar, 2017; Mohammad and Bravo-Marquez, 2017a; Buechel and Hahn, 2017, 2018a,b). Yet such approaches, often rooted in psychological theory, also turned out to be more challenging in respect to annotation and modeling (Strapparava and Mihalcea, 2007).

Surprisingly, one of the most valuable affective phenomena for improving human-machine interaction has received surprisingly little attention: *Empathy*. Prior work focused mostly on *spoken dialogue*, commonly addressing conversational agents, psychological interventions, or call center applications (McQuiggan and Lester, 2007; Fung et al., 2016; Pérez-Rosas et al., 2017; Alam et al., 2017).

In contrast, to the best of our knowledge, only three contributions (Xiao et al., 2012; Gibson et al., 2015; Khanpour et al., 2017) previously addressed *text-based* empathy prediction[1] (see Section 4 for details). Yet, all of them are limited in three ways: (a) neither of their corpora are available leaving the NLP community without shared data, (b) empathy ratings were provided by others than the one actually experiencing it which qualifies only as a weak form of ground truth, and (c) their notion of empathy is quite basic, falling short of current and past theory.

---

* These authors contributed equally to this work. Anneke Buffone designed and supervised the crowdsourcing task and the survey described in Section 2, and provided psychological background knowledge. Sven Buechel was responsible for corpus creation, data analysis, and modeling. The technical set-up of the crowdsourcing task and the survey was done jointly by both first authors.

†Work conducted while being at the University of Pennsylvania.

[1] Psychological studies commonly distinguish between *state* and *trait* empathy. While the former construct describes the amount of empathy a person experiences as a direct result of encountering a given stimulus, the latter refers to how empathetic one is on average and across situations. This studies exclusively addresses *state empathy*. For a contribution addressing *trait empathy* from an NLP perspective, see Abdul-Mageed et al. (2017).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4758–4765
Brussels, Belgium, October 31 - November 4, 2018.

In this contribution we present the first publicly available gold standard for text-based empathy prediction. It is constructed using a novel annotation methodology which reliably captures empathy assessments via multi-item scales. The corpus as well as our work as a whole is also unique in being—to the best of our knowledge—the first computational approach differentiating *multiple types of empathy*, empathic concern and personal distress, a distinction well recognized throughout psychology and other disciplines.[2]

2 Corpus Design and Methodology

Background. Most psychological theories of empathic states are focused on reactions to negative rather than positive events. Empathy for positive events remains less well understood and is thought to be regulated differently (Morelli et al., 2015). Thus we focus on empathetic reactions to need or suffering. Despite the fact that everyone has an immediate, implicit understanding of empathy, research has been vastly inconsistent in its definition and operationalization (Cuff et al., 2016). There is agreement, however, that there are multiple forms of empathy (see below). The by far most widely cited state empathy scale is Batson's Empathic Concern – Personal Distress Scale (Batson et al., 1987), henceforth *empathy* and *distress*.

Distress is a self-focused, negative affective state that occurs when one feels upset due to witnessing an entity's suffering or need, potentially via "catching" the suffering target's negative emotions. Empathy is a warm, tender, and compassionate feeling for a suffering target. It is other-focused, retains self-other separation, and is marked by relatively more positive affect (Batson and Shaw, 1991; Goetz et al., 2010; Mikulincer and Shaver, 2010; Sober and Wilson, 1997).

Selection of News Stories. Two research interns (psychology undergraduates) collected a total of 418 articles from popular online news platforms, selected to likely evoke empathic reactions, after being briefed on the goal and background of this study. These articles were then used to elicit empathic responses in participants.

Acquiring Text and Ratings. The corpus acquisition was set up as a crowdsourcing task on `MTurk.com` pointing to a `Qualtrics.com` questionnaire. The participants completed background measures on demographics and personality, and then proceeded to the main part of the survey where they read a random selection of five of the news articles. After reading each of the articles, participants were asked to rate their level of empathy and distress before describing their thoughts and feelings about it in writing.

In contrast to previous work, this set-up allowed us to acquire empathy scores of the actual *writer* of a text, instead of having to rely on an external evaluation by third parties (often student assistants with background in computer science). Arguably, our proposed annotation methodology yields more appropriate gold data, yet also leads to more variance in the relationship between linguistic features and empathic state ratings. That is because each rating reflects a single individual's feelings rather than a more stable average assessment by multiple raters. To account for this, we use *multi-item scales* as is common practice in psychology. I.e., participants give ratings for multiple items measuring the same construct (e.g., empathy) which are then averaged to obtain more reliable results. As far as we know, this is the first time that multi-item scales are used in sentiment analysis.[3]

In our case, participants used Batson's Empathic Concern – Personal Distress Scale (see above), i.e, rating 6 items for empathy (e.g., *warm, tender, moved*) and 8 items for distress (e.g., *troubled, disturbed, alarmed*) using a 7-point scale for each of those (see Appendix for details). After rating their empathy, participants were asked to share their feelings about the article as they would with a friend in a private message or with a group of friends as a social media post in 300 to 800 characters. Our final gold standard consists of these *messages* combined with the numeric ratings for empathy and distress.

In sum, 403 participants completed the survey. Median completion time was 32 minutes and each participant received 4 USD as compensation.

Post-Processing. Each message was manually reviewed by the authors. Responses which deviated from the task description (e.g., mere copying from the articles at display) were removed (31 responses, 155 messages), leading to a total 1860 messages in our final corpus. Gold ratings for empathy and distress were derived by averaging the respective items of the two multi-item scales.

---

[2] Data and code are available at: `https://github.com/wwbp/empathic_reactions`

[3] Here, we use *sentiment* as an umbrella term subsuming semantic orientation, emotion, as well as highly related concepts such as empathy.

	E	**D**	**Message**
(1)	4.8	3.1	*I'm sorry to hear that about Dakota's parents. Even when you are adult it must be hard to see your parents splitting up. No one wants that to happen and it's unfortunate that her parents couldn't work it out. I hope they are able to still remain civil around the kids and family. Just because it didn't work romantically doesn't mean it won't work at all.*
(2)	4.0	5.5	*Here's an article about crazed person who murdered two unfortunate women overseas. Life is crazy. I can't imagine what the families are going through. Having to go to or being forced into sex work is bad enough, but for it to end like this is just sad. It feels like there's no place safe in this world to be a woman sometimes.*
(3)	1.0	1.3	*I just read an article about some chowder-head who used a hammer and a pick ax to destroy Donald Trump's star on the Hollywood walk of fame. Wow, what a great protest. You sure showed him. Good job. Lol, can you believe this garbage? Who has such a hollow and pathetic life that they don't have anything better to do with their time than commit petty vandalism because they dislike some politician? What a dingus.*

Table 1: Illustrative examples from our newly created gold standard with ratings for empathy (**E**) and distress (**D**).

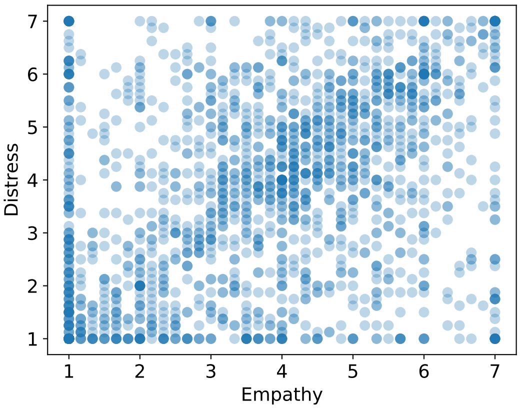

Figure 1: Scatter plot of the bivariate distribution of empathy and distress ratings.

3 Corpus Analysis

For a first impression of the language of our new gold standard, we provide illustrative examples in Table 1. The participant in Example (1) displays higher empathy than distress, (2) displays higher distress than empathy, and (3) shows neither empathic state, but employs sarcasm, colloquialisms and social-media-style acronyms to express lack of emotional response to the article. As can be seen, the language of our corpus is diverse and authentic, featuring many phenomena of natural language which render its computational understanding difficult, thus constituting a sound but challenging gold standard for empathy prediction.

Token Counts. We tokenized the 1860 messages using NLTK tools (Bird, 2006). In total, our corpus amounts to $173,686$ tokens. Individual message length varies between 52 and 198 tokens, the median being 84. See Appendix for details.

Rating Distribution. Figure 1 displays the bivariate distribution of empathy and distress ratings. As can be seen both target variables have a clear linear dependence, yet show only a moderate Pearson correlation of $r{=}.451$, similar to what was found in prior research (Batson et al., 1987, 1997). This finding supports that the two scales capture distinct affective phenomena and underscores the importance of our decision to describe empathic states in terms of *multiple* target variables, constituting a clear advancement over previous work. Both kinds of ratings show good coverage over the full range of the scales.

Reliability of Ratings. Since each message is annotated by only one rater, its author, typical measures of inter-rater agreement are not applicable. Instead, we compute *split-half reliability* (SHR), a standard approach in psychology (Cronbach, 1947) which also becomes increasingly popular in sentiment analysis (Mohammad and Bravo-Marquez, 2017a; Buechel and Hahn, 2018a). SHR is computed by splitting the ratings for the individual scale items (e.g., *warm*, *tender*, etc. for empathy) of all participants randomly into two groups, averaging the individual item ratings for each group and participant, and then measuring the correlation between both groups. This process is repeated 100 times with random splits, before again averaging the results. Doing so for empathy and distress, we find very high[4] SHR values of $r{=}.875$ and .924, respectively.

4 Modeling Empathy and Distress

In this section, we provide experimental results for modeling empathy and distress ratings based on the participants' messages (see Section 2). We examine three different types of models, varying in

---

[4] For a comparison against previously reported SHR values for different emotional categories, see Mohammad and Bravo-Marquez (2017b).

design complexity. Distinct models were trained for empathy and distress prediction.

First, ten percent of our newly created gold standard were randomly sampled to be used in development experiments. Then, the main experiment was conducted using 10-fold cross-validation (CV), providing each model with identical train-test splits to increase reliability. The dev set was excluded for the CV experiment.

Model performance is measured in terms of Pearson correlation r between predicted values and the human gold ratings. Thus, we phrase the prediction of empathy and distress as regression problems.

The input to our models is based on word embeddings, namely the publicly available FastText embeddings which were trained on Common Crawl ($\approx$600B tokens) (Bojanowski et al., 2017; Mikolov et al., 2018).

Ridge. Our first approach is Ridge regression, an ℓ^2-regularized version of linear regression. The centroid of the word embeddings of the words in a message is used as features (embedding centroid). The regularization coefficient α is automatically chosen from $\{1, .5, .1, ..., .0001\}$ during training.

FFN. Our second approach is a Feed-Forward Net with two hidden layers (256 and 128 units, respectively) with ReLU activation. Again, the embedding centroid is used as features.

CNN. The last approach is a Convolutional Neural Net.[5] We use a single convolutional layer with filter sizes 1 to 3, each with 100 output channels, followed by an average pooling layer and a dense layer of 128 units. ReLUs were used for the convolutional and again for the dense layer.

Both deep learning models were trained using the Adam optimizer (Kingma and Ba, 2015) with a fixed learning rate of 10^{-3} and a batch size of 32. We trained for a maximum of 200 epochs yet applied early stopping if the performance on the validation set did not improve for 20 consecutive epochs. We applied dropout with probabilities of .2, .5 and .5 on input, dense and pooling layers, respectively. Moreover ℓ^2 regularization of .001 was applied to the weights of conv and dense layers. Word embeddings were not updated.

The results are provided in Table 2. As can be seen, all of our models achieve satisfying performance figures ranging between r=.379 and .444, given the assumed difficulty of the task (see Section 3). On average over the two target variables, the CNN performs best, followed by Ridge and the FFN. While the CNN significantly outperforms the other models in every case, the differences between Ridge and the FFN are not statistically significant for either empathy or distress.[6] The improvements of the CNN over the other two approaches are much more pronounced for distress than for empathy. Since only the CNN is able to capture semantic effects from composition and word order, our data suggest that these phenomena are more important for predicting distress, whereas lexical features alone already perform quite well for empathy.

	Empathy	**Distress**	**Mean**
Ridge	.385	.410	.398
FFN	.379	.401	.390
CNN	**.404***	**.444***	**.424***

Table 2: Model performance for predicting empathy and distress in Pearson's r; with row-wise mean; best result per column in bold, significant ($p < .05$) improvement over other models marked with '*'.

Discussion. In comparison to closely related tasks such as emotion prediction (Mohammad and Bravo-Marquez, 2017a) our performance figures for empathy and distress prediction are generally lower. However, given the small amount of previous work for the problem at hand, we argue that our results are actually quite strong. This becomes obvious, again, in comparison with emotion analysis where early work achieved correlation values around r=.3 at most (Strapparava and Mihalcea, 2007). Yet state-of-the-art performance literally doubled over the last decade (Beck, 2017), in part due to much larger training sets.

Comparison to the limited body of previous work in text-based empathy prediction is difficult for a number of reasons, e.g., differences in domain, evaluation metric, as well as methodology and linguistic level of annotation. Khanpour et al. (2017) annotate and model empathy in online health communities on the *sentence*-level, whereas the instances in our corpus are much longer and comprise multiple sentences. In contrast to our work, they treat empathy prediction as a classification problem. Their best performing model, a CNN-LSTM, achieves an F-score of .78. Gibson

---

[5] Recurrent models did not perform well during development due to high sequence length.

[6] We use a two-tailed t-test for paired samples based on the results of the individual CV runs; $p < .05$.

et al. (2015) predict therapists' empathy in motivational interviews. Each therapy session transcript received one numeric score. Thus, each prediction is based on much more language data than our individual messages comprise. Their best model achieves a Spearman rank correlation of .61 using n-gram and psycholinguistic features.

Our contribution goes beyond both of these studies by, first, enriching empathy prediction with personal distress and, second, by annotating and modeling the empathic state actually felt by the writer, instead of relying on external assessments.

5 Conclusion

This contribution was the first to attempt empathy prediction in terms of *multiple* target variables, empathic concern and personal distress. We proposed a novel annotation methodology capturing empathic states actually felt by the author of a statement, instead of relying on third-party assessments. To ensure high reliability in this single-rating setting, we employ multi-item scales in line with best practices in psychology. Hereby we create the first publicly available gold standard for empathy prediction in written language, our survey being set-up and supervised by an expert psychologist. Our analysis shows that the data set excels with high rating reliability and an authentic and diverse language, rich of challenging phenomena such as sarcasm. We provide experimental results for three different predictive models, our CNN turning out superior.

Acknowledgments

Sven Buechel would like to thank his doctoral advisor Udo Hahn, JULIE Lab, for funding his research visit at the University of Pennsylvania.

References

Muhammad Abdul-Mageed, Anneke Buffone, Hao Peng, Johannes C Eichstaedt, and Lyle H Ungar. 2017. Recognizing pathogenic empathy in social media. In *ICWSM 2017 — Proceedings of the 11th International Conference on Web and Social Media*, pages 448–451, Montreal, Canada, May 15–18, 2017.

Muhammad Abdul-Mageed and Lyle H. Ungar. 2017. EmoNet: Fine-grained emotion detection with gated recurrent neural networks. In *ACL 2017 — Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, long papers, pages 718–728, Vancouver, British Columbia, Canada, July 30 – August 4, 2017.

Firoj Alam, Morena Danieli, and Giuseppe Riccardi. 2017. Annotating and modeling empathy in spoken conversations. *Computer Speech & Language*, pages 40–61.

C Daniel Batson, Jim Fultz, and Patricia A Schoenrade. 1987. Distress and empathy: Two qualitatively distinct vicarious emotions with different motivational consequences. *Journal of personality*, 55(1):19–39.

C Daniel Batson, Karen Sager, Eric Garst, Misook Kang, Kostia Rubchinsky, and Karen Dawson. 1997. Is empathy-induced helping due to self–other merging? *Journal of personality and social psychology*, 73(3):495.

C Daniel Batson and Laura L Shaw. 1991. Evidence for altruism: Toward a pluralism of prosocial motives. *Psychological inquiry*, 2(2):107–122.

Daniel Beck. 2017. Modelling representation noise in emotion analysis using gaussian processes. In *IJCNLP 2017 — Proceedings of the 8th International Joint Conference on Natural Language Processing*, volume 2, short papers, pages 140–145, Taipei, Taiwan, November 27 – December 1, 2017.

Steven Bird. 2006. NLTK: The natural language toolkit. In *COLING-ACL 2006 — Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, volume 4, interactive presentation sessions, pages 69–72, Sydney, Australia, July 17–21, 2006.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomáš Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5(1):135–146.

Sven Buechel and Udo Hahn. 2017. EmoBank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. In *EACL 2017 — Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2, short papers, pages 578–585, Valencia, Spain, April 3–7, 2017.

Sven Buechel and Udo Hahn. 2018a. Emotion representation mapping for automatic lexicon construction (mostly) performs on human level. In *COLING 2018 — Proceedings of the 27th International Conference on Computational Linguistics*, volume 1, technical papers, pages 2892–2904, Santa Fe, New Mexico, USA, August 20–26, 2018.

Sven Buechel and Udo Hahn. 2018b. Word emotion induction for multiple languages as a deep multitask learning problem. In *NAACL-HLT 2018 — Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, volume 1, long papers, pages 1907–1918, New Orleans, Louisiana, USA, June 1–6, 2018.

Lee J Cronbach. 1947. Test reliability: Its meaning and determination. *Psychometrika*, 12(1):1–16.

Benjamin MP Cuff, Sarah J Brown, Laura Taylor, and Douglas J Howat. 2016. Empathy: a review of the concept. *Emotion Review*, 8(2):144–153.

Mark H Davis. 1980. *Interpersonal Reactivity Index*. Edwin Mellen Press.

ED Diener, Robert A Emmons, Randy J Larsen, and Sharon Griffin. 1985. The satisfaction with life scale. *Journal of Personality Assessment*, 49(1):71–75.

Pascale Fung, Anik Dey, Farhad Bin Siddique, Ruixi Lin, Yang Yang, Yan Wan, and Ho Yin Ricky Chan. 2016. Zara the supergirl: An empathetic personality recognition system. In *NAACL 2016 — Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, volume 3, demonstrations, pages 87–91, San Diego, California, USA, June 12–17, 2016.

James Gibson, Nikolaos Malandrakis, Francisco Romero, David C Atkins, and Shrikanth S Narayanan. 2015. Predicting therapist empathy in motivational interviews using language features inspired by psycholinguistic norms. In *Interspeech 2015 — Proceedings of the 16th Annual Conference of the International Speech Communication Association*, pages 1947–1951, Dresden, Germany, September 6–10, 2015.

Jennifer L Goetz, Dacher Keltner, and Emiliana Simon-Thomas. 2010. Compassion: an evolutionary analysis and empirical review. *Psychological Bulletin*, 136(3):351.

Samuel D Gosling, Peter J Rentfrow, and William B Swann Jr. 2003. A very brief measure of the big-five personality domains. *Journal of Research in Personality*, 37(6):504–528.

Hamed Khanpour, Cornelia Caragea, and Prakhar Biyani. 2017. Identifying empathetic messages in online health communities. In *IJCNLP 2017 — Proceedings of the 8th International Joint Conference on Natural Language Processing*, volume 2, short papers, pages 246–251, Taipei, Taiwan, November 27 – December 1, 2017.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR 2015 — Proceedings of the 3rd International Conference on Learning Representations*, pages 1–15, San Diego, California, USA, May 7–9, 2015.

Bing Liu. 2015. *Sentiment Analysis: Mining Opinions, Sentiments and Emotions*. Cambridge University Press.

Scott W McQuiggan and James C Lester. 2007. Modeling and evaluating empathy in embodied companion agents. *International Journal of Human-Computer Studies*, 65(4):348–360.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *LREC 2018 — Proceedings of the 11th International Conference on Language Resources and Evaluation*, pages 52–55, Miyazaki, Japan, May 7-12, 2018.

M. Mikulincer and P. R. Shaver, editors. 2010. *Prosocial motives, emotions, and behavior: The better angels of our nature*. American Psychological Association.

Saif Mohammad and Felipe Bravo-Marquez. 2017a. WASSA-2017 shared task on emotion intensity. In *WASSA 2017 — Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis @ EMNLP*, pages 34–49, Copenhagen, Denmark, September 8, 2017.

Saif M. Mohammad and Felipe Bravo-Marquez. 2017b. Emotion intensities in tweets. In **SEM 2017 — Proceedings of the 6th Joint Conference on Lexical and Computational Semantics*, pages 65–77, Vancouver, British Columbia, Canada, August 3–4, 2017.

Sylvia A Morelli, Matthew D Lieberman, and Jamil Zaki. 2015. The emerging study of positive empathy. *Social and Personality Psychology Compass*, 9(2):57–68.

Verónica Pérez-Rosas, Rada Mihalcea, Kenneth Resnicow, Satinder Singh, and Lawrence An. 2017. Understanding and predicting empathic behavior in counseling therapy. In *ACL 2017 — Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, long papers, pages 1426–1435, Vancouver, British Columbia, Canada, July 30 – August 4, 2017.

R. W. Picard. 1997. *Affective Computing*. MIT Press.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 Task 4: Sentiment analysis in Twitter. In *SemEval 2017 — Proceedings of the 11th International Workshop on Semantic Evaluation @ ACL*, pages 502–518, Vancouver, British Columbia, Canada, August 3–4, 2017.

H Andrew Schwartz, Salvatore Giorgi, Maarten Sap, Patrick Crutchley, Lyle Ungar, and Johannes Eichstaedt. 2017. DLATK: Differential language analysis toolkit. In *EMNLP 2017 — Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, volume 2, system demonstrations, pages 55–60, Copenhagen, Denmark, September 7–11, 2017.

Elliott Sober and David Sloan Wilson. 1997. Unto others: The evolution of altruism. *Harvard University*.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP 2013 — Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 18–21, 2013.

Carlo Strapparava and Rada Mihalcea. 2007. SemEval 2007 Task 14: Affective text. In *SemEval 2007 — Proceedings of the 4th International Workshop on Semantic Evaluations @ ACL 2007*, pages 70–74, Prague, Czech Republic, June 23–24, 2007.

Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional CNN-LSTM model. In *ACL 2016 — Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, short papers, pages 225–230, Berlin, Germany, August 7–12, 2016.

Bo Xiao, Dogan Can, Panayiotis G Georgiou, David Atkins, and Shrikanth S Narayanan. 2012. Analyzing the language of therapist empathy in motivational interview based psychotherapy. In *APSIPA 2012 — Proceedings of the 2012 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–4, Hollywood, California, USA, December 3–6, 2012.

A Supplemental Material

Details on Stimulus and Instructions

Before being used in our survey, the selected news articles were categorized by the research interns who gathered them in terms of their intensity of suffering (major or minor), cause of suffering (political, human, nature or other), patient of suffering (humans, animals, environment, or other) and scale of suffering (individual or mass). Research interns also provided a short list of key words for each article. This additional information was gathered to examine the influence of these factors on empathy elicitation and modeling performance in later studies.

At the beginning of the survey participants completed background items covering general demographics (including age, gender, and ethnicity), the most commonly used *trait* empathy scale, the Interpersonal Reactivity Index (Davis, 1980), a brief assessment of the Big 5 personality traits (Gosling et al., 2003), life satisfaction (Diener et al., 1985), as well as a brief measure of generalized trust.

After reading each of the articles, participants rated their level of empathic concern and personal distress using multi-item scales. **Figure 2** shows a cropped screenshot of the survey hosted on `Qualtrics.com`. The first six items (*warm, tender, sympathetic, softhearted, moved*, and *compassionate*) refer to empathy. The last eight items (*worried, upset, troubled, perturbed, grieved, disturbed, alarmed*, and *distressed*) refer to distress.

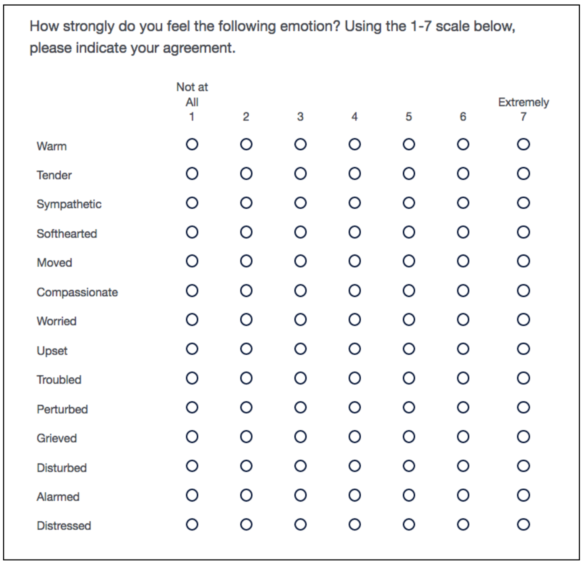

How strongly do you feel the following emotion? Using the 1-7 scale below, please indicate your agreement.

	Not at All 1	2	3	4	5	6	Extremely 7
Warm	○	○	○	○	○	○	○
Tender	○	○	○	○	○	○	○
Sympathetic	○	○	○	○	○	○	○
Softhearted	○	○	○	○	○	○	○
Moved	○	○	○	○	○	○	○
Compassionate	○	○	○	○	○	○	○
Worried	○	○	○	○	○	○	○
Upset	○	○	○	○	○	○	○
Troubled	○	○	○	○	○	○	○
Perturbed	○	○	○	○	○	○	○
Grieved	○	○	○	○	○	○	○
Disturbed	○	○	○	○	○	○	○
Alarmed	○	○	○	○	○	○	○
Distressed	○	○	○	○	○	○	○

Figure 2: Multi-item scales for empathic concern and personal distress.

After completing the rating items, participants were instructed to describe their reactions in writing as follows: *Now that you have read this article, please write a message to a friend or friends about your feelings and thoughts regarding the article you just read. This could be a private message to a friend or something you would post on social media. Please do not identify your intended friend(s) — just write your thoughts about the article as if you were communicating with them. Please use between 300 and 800 characters.*

Further Corpus Analyses

The word clouds in **Figure 3** and **Figure 4** show 1-grams of our corpus which correlate significantly (Benjamini-Hochberg corrected $p < .05$) with high empathy and high distress ratings, respectively. In the word clouds, larger size indicates higher correlation and the color scale, gray-blue-red, indicates word frequency, dark red being most prevalent. The Differential Language Analysis Toolkit (Schwartz et al., 2017) was utilized for this analysis. As can be seen, the word clouds display high face-validity, giving further evidence for the soundness of our acquisition methodology.

Figure 3: Word cloud of high empathy 1-grams.

Figure 4: Word cloud of high distress 1-grams.

Figure 5 displays the distribution of the message length of our corpus in tokens. As can be seen the majority of messages contain between 60 and 100 tokens. Yet outliers go up to almost 200. The introduction of a character cap for the writing task proved successful in comparison to a pilot study where this measure has not been in place. In the latter case, the maximum number of tokens was nearly twice as high due to even stronger outliers.

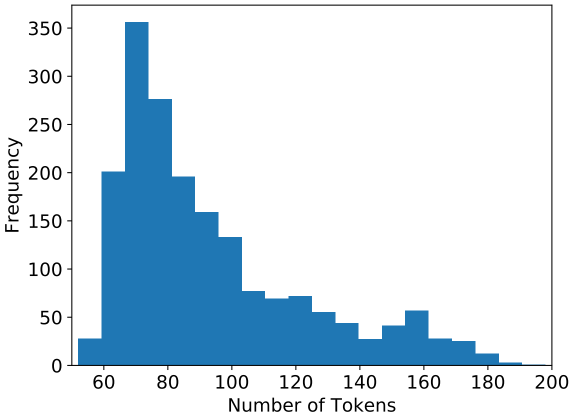

Figure 5: Histogram of message length in our corpus.

Interpretable Emoji Prediction via Label-Wise Attention LSTMs

Francesco Barbieri◇ **Luis Espinosa-Anke**♠ **Jose Camacho-Collados**♠
Steven Schockaert♠ **Horacio Saggion**◇
◇ Large Scale Text Understanding Systems Lab, TALN, UPF, Barcelona, Spain
♠School of Computer Science and Informatics, Cardiff University, UK

◇`{francesco.barbieri,horacio.saggion}@upf.edu,`
♠`{espinosa-ankel,camachocolladosj,schockaerts1}@cardiff.ac.uk`

Abstract

Human language has evolved towards newer forms of communication such as social media, where emojis (i.e., ideograms bearing a visual meaning) play a key role. While there is an increasing body of work aimed at the computational modeling of emoji semantics, there is currently little understanding about what makes a computational model represent or predict a given emoji in a certain way. In this paper we propose a label-wise attention mechanism with which we attempt to better understand the nuances underlying emoji prediction. In addition to advantages in terms of interpretability, we show that our proposed architecture improves over standard baselines in emoji prediction, and does particularly well when predicting infrequent emojis.

1 Introduction

Communication in social media differs from more standard linguistic interactions across a wide range of dimensions. Immediacy, short text length, the use of pseudowords like *#hashtags* or *@mentions*, and even metadata such as user information or geolocalization are essential components of social media messages. In addition, the use of *emojis*, small ideograms depicting objects, people and scenes (Cappallo et al., 2015), are becoming increasingly important for fully modeling the underlying semantics of a social media message, be it a product review, a *tweet* or an *Instagram* post. Emojis are the evolution of character-based emoticons (Pavalanathan and Eisenstein, 2015), and are extensively used, not only as sentiment carriers or boosters, but more importantly, to express ideas about a myriad of topics, e.g., mood (😘), food (🍲), sports (🏃) or scenery (🏝).

Emoji modeling and prediction is, therefore, an important problem towards the end goal of properly capturing the intended meaning of a social media message. In fact, emoji prediction, i.e., given a (usually short) message, predict its most likely associated emoji(s), may help to improve different NLP tasks (Novak et al., 2015), such as information retrieval, generation of emoji-enriched social media content or suggestion of emojis when writing text messages or sharing pictures online. It has furthermore proven to be useful for sentiment analysis, emotion recognition and irony detection (Felbo et al., 2017). The problem of emoji prediction, albeit recent, has already seen important developments. For example, Barbieri et al. (2017) describe an LSTM model which outperforms a logistic regression baseline based on word vector averaging, and even human judgement in some scenarios.

The above contributions, in addition to emoji similarity datasets (Barbieri et al., 2016; Wijeratne et al., 2017) or emoji sentiment lexicons (Novak et al., 2015; Wijeratne et al., 2016; Kimura and Katsurai, 2017; Rodrigues et al., 2018), have paved the way for better understanding the semantics of emojis. However, our understanding of what exactly the neural models for emoji prediction are capturing is currently very limited. What is a model prioritizing when associating a message with, for example, positive (😁), negative (😠) or patriotic (🇺🇸) intents? A natural way of assessing this would be to implement an attention mechanism over the hidden states of LSTM layers. Attentive architectures in NLP, in fact, have recently received substantial interest, mostly for sequence-to-sequence models (which are useful for machine translation, summarization or language modeling), and a myriad of modifications have been proposed, including additive (Bahdanau et al., 2015), multiplicative (Luong et al., 2015) or self (Lin et al., 2017) attention mechanisms.

However, standard attention mechanisms only tell us which text fragments are considered impor-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4766–4771
Brussels, Belgium, October 31 - November 4, 2018.

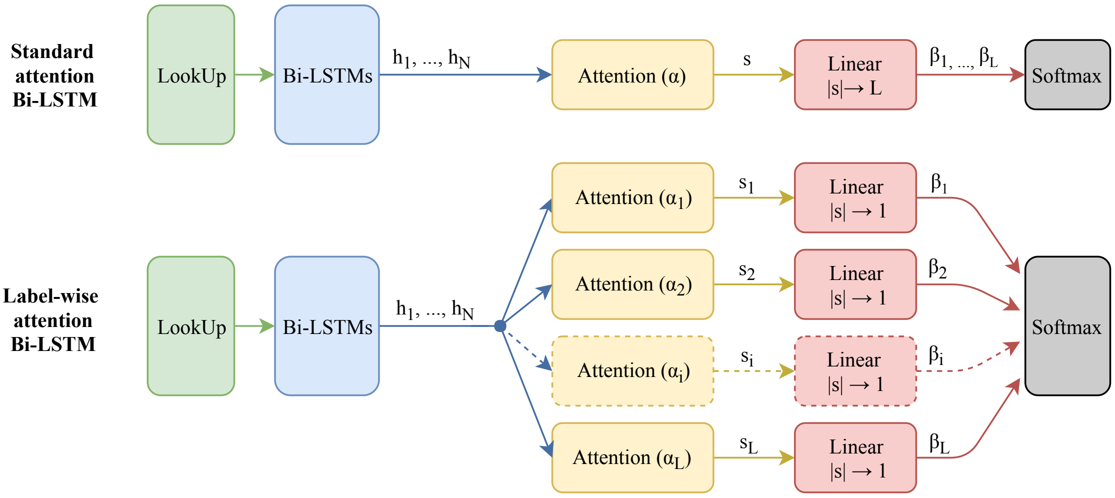

Figure 1: A classic attention network (top), and our attentive label-wise network (bottom), with a specific attention module for each label.

tant for the overall prediction distribution. While emoji prediction has predominantly been treated as a multi-class classification problem in the literature, it would be more informative to analyze which text fragments are considered important *for each individual emoji*. With this motivation in mind, in this paper we put forward a *label-wise* mechanism that operates over each label during training. The resulting architecture intuitively behaves like a batch of binary mini-classifiers, which make decisions over one single emoji at a time, but without the computational burden and risk of overfitting associated with learning separate LSTM-based classifiers for each emoji.

Our contribution in this paper is twofold. First, we use the proposed label-wise mechanism to analyze the behavior of neural emoji classifiers, exploiting the attention weights to uncover and interpret emoji usages. Second, we experimentally compare the effect of the label-wise mechanism on the performance of an emoji classifier. We observed a performance improvement over competitive baselines such as `FastText` (`FT`) (Joulin et al., 2017) and `Deepmoji` (Felbo et al., 2017), which is most noticeable in the case of infrequent emojis. This suggests that an attentive mechanism can be leveraged to make neural architectures more sensitive to instances of underrepresented classes.

2 Methodology

Our base architecture is the Deepmoji model (Felbo et al., 2017), which is based on two stacked word-based bi-directional LSTM recurrent neural networks with skip connections between the first and the second LSTM. The model also includes an attention module to increase its sensitivity to individual words during prediction. In general, attention mechanisms allow the model to focus on specific words of the input (Yang et al., 2016), instead of having to memorize all the important features in a fixed-length vector. The main architectural difference with respect to the typical attention is illustrated in Figure 1.

In Felbo et al. (2017), attention is computed as follows:

$$z_i = w_a h_i + b_a$$
$$\alpha_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$
$$s = \sum_{j=1}^{N} \alpha_j h_j$$

Here $h_i \in \mathbb{R}^d$ is the hidden representation of the LSTM corresponding to the i^{th} word, with N the total number of words in the sentence. The weight vector $w_a \in \mathbb{R}^d$ and bias term $b_a \in \mathbb{R}$ map this hidden representation to a value that reflects the importance of this state for the considered classification problem. The values $z_1, ..., z_n$ are then normalized using a softmax function, yielding the attention weights α_i. The sentence representation s is defined as a weighted average of the vectors h_i. The final prediction distribution is then defined as follows:

$$\beta_l = w_{f,l} s + b_{f,l}$$

$$p_l = \frac{e^{\beta_l}}{\sum_{r=1}^{L} e^{\beta_r}}$$

where $w_{f,l} \in \mathbb{R}^d$ and $b_{f,l}$ define a label-specific linear transformation, with β_l reflecting our confidence in the l^{th} label and L is the total number of labels. The confidence scores β_l are then normalized to probabilities using another softmax operation. However, while the above design has contributed to better emoji prediction, in our case we are interested in understanding the contribution of the words of a sentence for each label (i.e., emoji), and not in the whole distribution of the target labels. To this end, we propose a label-wise attention mechanism. Specifically, we apply the same type of attention, but repeating it $|L|$ (number of labels) times, where each attention module is reserved for a specific label l:

$$\begin{aligned} z_{i,l} &= w_{a,l} h_i + b_{a,l} \\ \alpha_{i,l} &= \frac{e^{z_{i,l}}}{\sum_{j=1}^{N} e^{z_{j,l}}} \\ s_l &= \sum_{j=1}^{N} \alpha_{j,l} h_j \\ \beta_l &= w_{f,l} s_l + b_{f,l} \\ p_l &= \frac{e^{\beta_l}}{\sum_{r=1}^{L} e^{\beta_r}} \end{aligned}$$

3 Evaluation

This section describes the main experiment w.r.t the performance of our proposed attention mechanism, in comparison with existing emoji prediction systems. We use the data made available in the context of the SemEval 2018 Shared Task on Emoji Prediction (Barbieri et al., 2018). Given a tweet, the task consists of predicting an associated emoji from a predefined set of 20 emoji labels. We evaluate our model on the English split of the official task dataset. We also show results from additional experiments in which the label space ranged from 20 to 200 emojis. These *extended* experiments are performed on a corpus of around 100M tweets geolocalized in the United States and posted between October 2015 and May 2018.

Models. In order to put our proposed label-wise attention mechanism in context, we compare its performance with a set of baselines: (1) `FastText` (Joulin et al., 2017) (`FT`), which was the official baseline in the SemEval task; (2) 2 stacked Bi-LSTMs (`2-BiLSTMs`) without attention; and (3) 2 stacked Bi-LSTMs with standard attention (`2-BiLSTMs`$_a$) (Felbo et al., 2017). Finally, we denote as `2-BiLSTMs`$_l$ our proposed label-wise attentive Bi-LSTM architecture.

Lab	Syst	**F1**	**A@1**	**A@5**	**CE**
20*	`FastText`	30.97	42.57	72.45	4.56
	`2-BiLSTM`	33.52	45.76	75.54	3.88
	`2-BiLSTM`$_a$	**34.11**	**46.11**	75.68	3.86
	`2-BiLSTM`$_l$	33.51	45.94	**76.02**	**3.82**
50	`FastText`	18.04	22.33	48.13	14.27
	`2-BiLSTM`	19.07	25.35	53.38	9.37
	`2-BiLSTM`$_a$	19.83	25.52	53.51	9.35
	`2-BiLSTM`$_l$	**20.08**	**25.64**	**53.77**	**9.26**
100	`FastText`	16.25	20.29	42.65	26.04
	`2-BiLSTM`	17.44	**23.01**	**47.46**	15.24
	`2-BiLSTM`$_a$	17.56	22.77	46.93	15.51
	`2-BiLSTM`$_l$	**17.92**	22.80	47.41	**15.17**
200	`FastText`	13.31	18.80	38.99	51.06
	`2-BiLSTM`	16.16	21.05	42.64	24.68
	`2-BiLSTM`$_a$	16.30	21.13	42.50	24.60
	`2-BiLSTM`$_l$	**16.91**	**21.39**	**43.35**	**23.73**

Table 1: Experimental results of the two baselines, as well as single and label-wise attention modifications to the "vanilla" `2-BiLSTM` model.

Results. Table 1 shows the results of our model and the baselines in the emoji prediction task for the different evaluation splits. The evaluation metrics used are: **`F1`**, Accuracy@k (**`A@k`**, where $k \in \{1, 5\}$), and Coverage Error (**`CE`**[1]) (Tsoumakas et al., 2009). We note that the latter metric is not normally used in emoji prediction settings. However, with many emojis being "near synonyms" (in the sense of being often used almost interchangeably), it seems natural to evaluate the performance of an emoji prediction system in terms of how far we would need to go through the predicted emojis to recover the true label. The results show that our proposed `2-BiLSTMs`$_l$ method outperforms all baselines for F1 in three out of four settings, and for CE in all of them. In the following section we shed light on the reasons behind this performance, and we try to understand how these predictions were made.

---

[1] CE is computed as the average number of labels that need to be in the predictions for all true labels to be predicted.

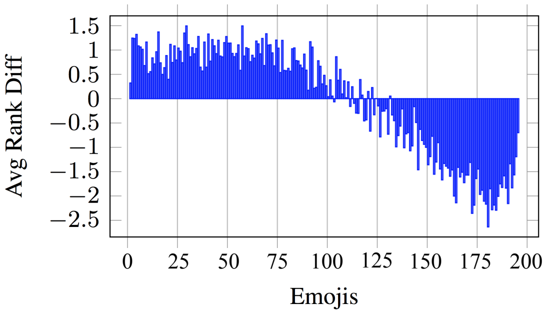

Figure 2: Difference in rank distributions. The x-axis represents emoji labels, ranked from most to least frequent. Lower scores indicate a higher average rank predicted by our proposed label-wise attention mechanism.

4 Analysis

By inspecting the predictions of our model, we found that the label-wise attention mechanism tends to be less heavily biased towards the most frequent emojis. This is reflected in the lower coverage error results in all settings, and becomes more noticeable as the number of labels grows. We verified this by computing the average difference between ranked predictions of the two attentive models in the 200-label setting (Figure 2). We can observe a sudden switch at more or less the median emoji, after which the label-wise attention model becomes increasingly accurate (relative to the standard attention model). This can be explained by the fact that infrequent emojis tend to be more situational (used in specific contexts and leaving less room for ambiguity or interchangeability), which the label-wise attention mechanism can take advantage of, as it explicitly links emojis with highly informative words. Let us illustrate this claim with a case in which the label-wise attention model predicts the correct emoji, unlike its single-attention counterpart:

a friendship is built over time , but sisterhood is given automatically. *Gold:* 👯

For the above example[2], the predictions of the single attention model were all linked to the *general* meaning of the message, that is love and friendship, leading it to predict associated emojis (❤️, 💕 and 💗), failing to capture the most relevant bit of information. On the other hand, our proposed model "picks on" the word *sisterhood*, and with

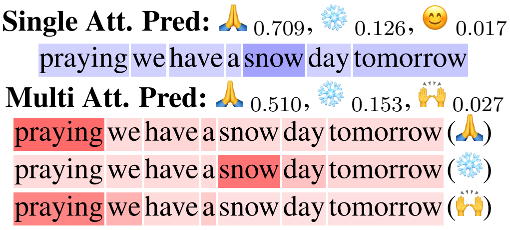
Single Att. Pred: 🙏 $_{0.709}$, ❄️ $_{0.126}$, 😊 $_{0.017}$

praying we have a snow day tomorrow

Multi Att. Pred: 🙏 $_{0.510}$, ❄️ $_{0.153}$, 🙌 $_{0.027}$

praying we have a snow day tomorrow (🙏)

praying we have a snow day tomorrow (❄️)

praying we have a snow day tomorrow (🙌)

Figure 3: Attention weights α and α_l of single and label-wise attentive models. *Gold*: 🙏.

the added context of the surrounding words, ranks the gold label[3] in 4th position, which would be a true positive as per **A@5**.

Let us explore what we argue are interesting cases of emoji usage (ranging from highly explicit to figurative or situtational intent). Figure 3 shows how the word (*praying*) and emojis such as 🙏 and 🙌 are strongly correlated. In addition, the bond between the word *snow* and the ❄️ emoji is also indisputable. However, a perhaps more surprising example is displayed in Figure 4, which is a negative example. Here, the ✔️ emoji was predicted with rank 1, and we see it being strongly associated with the ordinal *second*, suggesting that the model assumed this was some kind of "ticked enumeration" of completed tasks, which is indeed regular practice in Twitter. Finally, we found it remarkable that the ambiguous nature of the word *boarding* is also reflected in two different emojis being predicted with high probability (❄️ and ✈️), each of them showcasing one of the word's senses.

As an additional exploratory analysis, we computed statistics on those words with the highest average attention weights associated with one single emoji. One interesting example is the 🌲 emoji, which shows two clear usage patterns: one literal (a tree) and one figurative (*christmas* and *holidays*). Finally, as a final (and perhaps thought-provoking) finding, the highest attention weights associated to the 🏈 emoji were given to the words *game*, *boys* and *football*, in that order. In other words, the model relies more on the word *boys* than on the actual description of the emoji. This is in line with a previous study that showed how the current usage of emojis in Twitter is in some cases associated with gender stereotypes (Barbieri and Camacho-Collados, 2018).

---

[2] The highlights show the α_l attention weights of 👯.

[3] Which is among the 10% most infrequent emojis in the dataset.

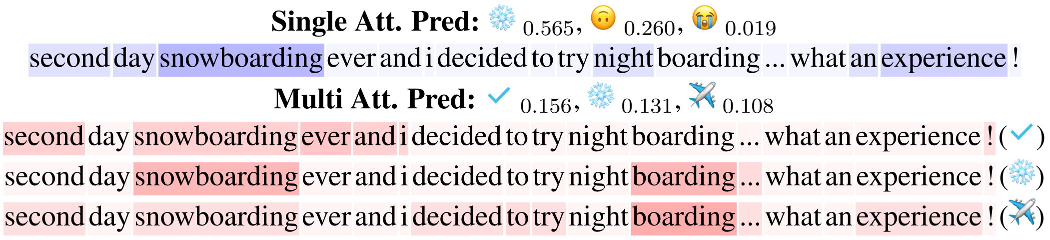
Single Att. Pred: ❄️ 0.565, 🙃 0.260, 😭 0.019

second day snowboarding ever and i decided to try night boarding ... what an experience !

Multi Att. Pred: ✓ 0.156, ❄️ 0.131, ✈️ 0.108

second day snowboarding ever and i decided to try night boarding ... what an experience ! (✓)

second day snowboarding ever and i decided to try night boarding ... what an experience ! (❄️)

second day snowboarding ever and i decided to try night boarding ... what an experience ! (✈️)

Figure 4: Attention weights α and α_l of single and label-wise attentive models. *Gold*: 😂.

5 Conclusion

In this paper we have presented a neural architecture for emoji prediction based on a label-wise attention mechanism, which, in addition to improving performance, provides a degree of interpretability about how different features are used for predictions, a topic of increasing interest in NLP (Linzen et al., 2016; Palangi et al., 2017). As we experimented with sets of emoji labels of different sizes, our proposed label-wise attention architecture proved especially well-suited for emojis which were infrequent in the training data, making the system less biased towards the most frequent. We see this as a first step to improve the robustness of recurrent neural networks in datasets with unbalanced distributions, as they were shown not to perform better than well-tuned SVMs on the emoji predicion task (Çöltekin and Rama, 2018).

As for future work, we plan to apply our label-wise attention mechanism to understand other interesting linguistic properties of human-generated text in social media, and other multi-class or multi-label classification problems.

Finally, code to reproduce our experiments and additional examples of label-wise attention weights from input tweets can be downloaded at `https://fvancesco.github.io/label_wise_attention/`.

Acknowledgments

F. Barbieri and H. Saggion acknowledge support from the TUNER project (TIN2015-65308-C5-5-R, MINECO/FEDER, UE). Luis Espinosa-Anke, Jose Camacho-Collados and Steven Schockaert have been supported by ERC Starting Grant 637277.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 105–111.

Francesco Barbieri and Jose Camacho-Collados. 2018. How gender and skin tone modifiers affect emoji semantics in twitter. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 101–106. Association for Computational Linguistics.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa-Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. SemEval-2018 Task 2: Multilingual Emoji Prediction. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, United States. Association for Computational Linguistics.

Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. 2016. What does this emoji mean? a vector space skip-gram model for Twitter emojis. In *Proc. of LREC 2016*.

Spencer Cappallo, Thomas Mensink, and Cees GM Snoek. 2015. Image2emoji: Zero-shot emoji prediction for visual media. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1311–1314. ACM.

Çağrı Çöltekin and Taraka Rama. 2018. Tübingen-oslo at semeval-2018 task 2: Svms perform better than rnns in emoji prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 32–36, New Orleans, Louisiana. Association for Computational Linguistics.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *European Chapter of the Association for Computational Linguistics*, Valencia, Spain.

Mayu Kimura and Marie Katsurai. 2017. Automatic construction of an emoji sentiment lexicon. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 1033–1036. ACM.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *PloS one*, 10(12):e0144296.

Hamid Palangi, Paul Smolensky, Xiaodong He, and Li Deng. 2017. Deep learning of grammatically-interpretable representations through question-answering. *arXiv preprint arXiv:1705.08432*.

Umashanthi Pavalanathan and Jacob Eisenstein. 2015. Emoticons vs. emojis on Twitter: A causal inference approach. *arXiv preprint arXiv:1510.08480*.

David Rodrigues, Marília Prada, Rui Gaspar, Margarida V Garrido, and Diniz Lopes. 2018. Lisbon emoji and emoticon database (leed): Norms for emoji and emoticons in seven evaluative dimensions. *Behavior research methods*, pages 392–405.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2009. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer.

Sanjaya Wijeratne, Lakshika Balasuriya, Amit Sheth, and Derek Doran. 2016. Emojinet: Building a machine readable sense inventory for emoji. In *International Conference on Social Informatics*, pages 527–541. Springer.

Sanjaya Wijeratne, Lakshika Balasuriya, Amit Sheth, and Derek Doran. 2017. A semantics-based measure of emoji similarity. In *Proceedings of the International Conference on Web Intelligence*, pages 646–653. ACM.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

A Tree-based Decoder for Neural Machine Translation

Xinyi Wang[1], **Hieu Pham**[1,2], **Pengcheng Yin**[1], **Graham Neubig**[1]

`{xinyiw1,hyhieu,pcyin,gneubig}@cs.cmu.edu`
[1]Language Technology Institute, Carnegie Mellon University, Pittsburgh, PA 15213
[2]Google Brain, Mountain View, CA 94043

Abstract

Recent advances in Neural Machine Translation (NMT) show that adding syntactic information to NMT systems can improve the quality of their translations. Most existing work utilizes some specific types of linguistically-inspired tree structures, like constituency and dependency parse trees. This is often done via a standard RNN decoder that operates on a linearized target tree structure. However, it is an open question of what specific linguistic formalism, if any, is the best structural representation for NMT. In this paper, we (1) propose an NMT model that can naturally generate the topology of an arbitrary tree structure on the target side, and (2) experiment with various target tree structures. Our experiments show the surprising result that our model delivers the best improvements with balanced binary trees constructed *without* any linguistic knowledge; this model outperforms standard *seq2seq* models by up to 2.1 BLEU points, and other methods for incorporating target-side syntax by up to 0.7 BLEU.[1]

1 Introduction

Most NMT methods use sequence-to-sequence (*seq2seq*) models, taking in a *sequence* of source words and generating a *sequence* of target words (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). While *seq2seq* models can *implicitly* discover syntactic properties of the source language (Shi et al., 2016), they do not explicitly model and leverage such information. Motivated by the success of adding syntactic information to Statistical Machine Translation (SMT) (Galley et al., 2004; Menezes and Quirk, 2007; Galley et al., 2006), recent works have established that explicitly leveraging syntactic information can improve NMT quality, either through syntactic encoders (Li et al., 2017; Eriguchi et al., 2016), multi-task learning objectives (Chen et al., 2017; Eriguchi et al., 2017), or direct addition of syntactic tokens to the target sequence (Nadejde et al., 2017; Aharoni and Goldberg, 2017). However, these syntax-aware models only employ the standard decoding process of *seq2seq* models, *i.e.* generating one target word at a time. One exception is Wu et al. (2017), which utilizes two RNNs for generating target dependency trees. Nevertheless, Wu et al. (2017) is specifically designed for dependency tree structures and is not trivially applicable to other varieties of trees such as phrase-structure trees, which have been used more widely in other works on syntax-based machine translation. One potential reason for the dearth of work on syntactic decoders is that such parse tree structures are not friendly to recurrent neural networks (RNNs).

In this paper, we propose *TrDec*, a method for incorporating tree structures in NMT. TrDec simultaneously generates a target-side tree topology and a translation, using the partially-generated tree to guide the translation process (§ 2). TrDec employs two RNNs: a *rule RNN,* which tracks the topology of the tree based on rules defined by a Context Free Grammar (CFG), and a *word RNN,* which tracks words at the leaves of the tree (§ 3). This model is similar to neural models of tree-structured data from syntactic and semantic parsing (Dyer et al., 2016; Alvarez-Melis and Jaakkola, 2017; Yin and Neubig, 2017), but with the addition of the word RNN, which is especially important for MT where fluency of transitions over the words is critical.

TrDec can generate any tree structure that can be represented by a CFG. These structures include linguistically-motivated *syntactic* tree representations, *e.g.* constituent parse trees, as well as *syntax-free* tree representations, *e.g.* balanced binary trees (§ 4). This flexibility of TrDec allows us to com-

---

[1] Our code is available at `https://github.com/cindyxinyiwang/TrDec_pytorch`.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4772–4777
Brussels, Belgium, October 31 - November 4, 2018.

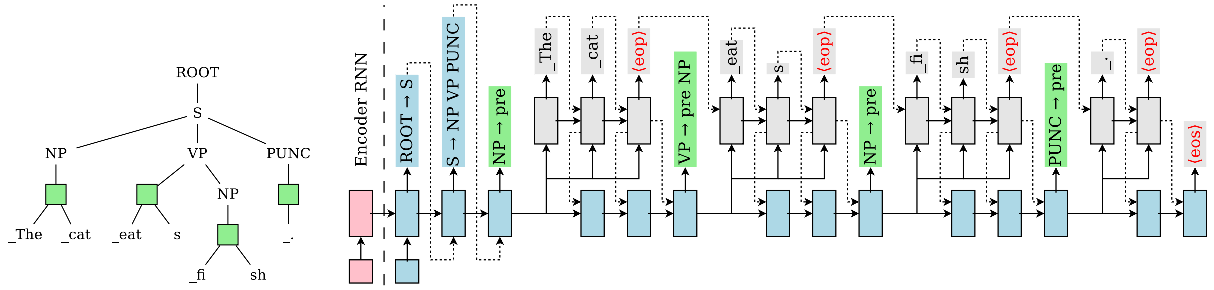

Figure 1: An example generation process of TrDec. *Left:* A target parse tree. The green squares represent preterminal nodes. *Right:* How our RNNs generate the parse tree on the left. The blue cells represent the activities of the rule RNN, while the grey cells represent the activities of the word RNN. ⟨eop⟩ and ⟨eos⟩ are the *end-of-phrase* and *end-of-sentence* tokens. Best viewed in color.

pare and contrast different structural representations for NMT.

In our experiments (§ 5), we evaluate TrDec using both syntax-driven and syntax-free tree representations. We benchmark TrDec on three tasks: Japanese-English and German-English translation with medium-sized datasets, and Oromo-English translation with an extremely small dataset. Our findings are surprising – TrDec performs well, but it performs the best with balanced binary trees constructed *without* any linguistic guidance.

2 Generation Process

TrDec simultaneously generates the target sequence and its corresponding tree structure. We first discuss the high-level generation process using an example, before describing the prediction model (§ 3) and the types of trees used by TrDec (§ 4).

Fig. 1 illustrates the generation process of the sentence "`_The _cat _eat s _fi sh _.`", where the sentence is split into subword units, delimited by the underscore "`_`" (Sennrich et al., 2016). The example uses a syntactic parse tree as the intermediate tree representation, but the process of generating with other tree representations, *e.g.* syntax-free trees, follows the same procedure.

Trees used in TrDec have two types of nodes: *terminal nodes*, *i.e.* the leaf nodes that represent subword units; and *nonterminal nodes*, *i.e.* the nonleaf nodes that represent a span of subwords. Additionally, we define a *preterminal node* to be a nonterminal node whose children are all terminal nodes. In Fig. 1 *Left*, the green squares represent preterminal nodes.

TrDec generates a tree in a top-down, left-to-right order. The generation process is guided by a CFG over target trees, which is constructed by taking all production rules extracted from the trees of all sentences in the training corpus. Specifically, a *rule RNN* first generates the top of the tree structure, and continues until a preterminal is reached. Then, a *word RNN* fills out the words under the preterminal. The model switches back to the *rule RNN* after the *word RNN* finishes. This process is illustrated in Fig. 1 *Right*. Details are as follows:

Step 1. The source sentence is encoded by a sequential RNN encoder, producing the hidden states.

Step 2. The generation starts with a derivation tree with only a `Root` node. A *rule RNN,* initialized by the last encoder hidden state computes the probability distribution over all CFG rules whose left hand side (LHS) is `Root`, and selects a rule to apply to the derivation. In our example, the rule RNN selects `ROOT` $\mapsto$ `S`.

Step 3. The rule RNN applies production rules to the derivation in a top-down, left-to-right order, expanding the current *opening nonterminal* using a CFG rule whose LHS is the opening nonterminal. In the next two steps, TrDec applies the rules `S` $\mapsto$ `NP VP PUNC` and `NP` $\mapsto$ `pre` to the opening nonterminals `S` and `NP`, respectively. Note that after these two steps a preterminal node `pre` is created.

Step 4a. Upon seeing a preterminal node as the current opening nonterminal, TrDec switches to using a *word RNN*, initialized by the last state of the encoder, to populate this empty preterminal with phrase tokens, similar to a *seq2seq* decoder. For example the subword units `_The` and `_cat` are generated by the word RNN, ending with a special *end-of-phrase* token, *i.e.* ⟨eop⟩.

Step 4b. While the word RNN generates subword units, the rule RNN also updates its own hidden states, as illustrated by the blue cells in Fig. 1 *Right*.

Step 5. After the word RNN generates ⟨eop⟩, TrDec switches back to the rule RNN to continue generating the derivation from where the tree left off. In our example, this next stage is the opening nonterminal node `VP`. From here, TrDec chooses the rule `VP` $\mapsto$ `pre NP`.

TrDec repeats the process above, intermingling the rule RNN and the word RNN as described, and halts when the rule RNN generates the *end-of-sentence* token $\langle$eos$\rangle$, completing the derivation.

3 Model

We now describe the computations during the generation process discussed in § 2. At first, a source sentence $\mathbf{x}$, which is split into subwords, is encoded using a standard bi-directional Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997). This bi-directional LSTM outputs a set of hidden states, which TrDec will reference using an attention function (Bahdanau et al., 2015).

As discussed, TrDec uses two RNNs to generate a target parse tree. In our work, both of these RNNs use LSTMs, but with different parameters.

Rule RNN. At any time step t in the rule RNN, there are two possible actions. If at the previous time step $t-1$, TrDec generated a CFG rule, then the state $\mathbf{s}_t^{\text{tree}}$ is computed by:

$$\mathbf{s}_t^{\text{tree}} = \text{LSTM}([\mathbf{y}_{t-1}^{\text{CFG}}; \mathbf{c}_{t-1}; \mathbf{s}_p^{\text{tree}}; \mathbf{s}_t^{\text{word}}], \mathbf{s}_{t-1}^{\text{tree}})$$

where $\mathbf{y}_{t-1}^{\text{CFG}}$ is the embedding of the CFG rule at time step $t-1$; $\mathbf{c}_{t-1}$ is the context vector computed by attention at $\mathbf{s}_{t-1}^{\text{tree}}$, *i.e.* input feeding (Luong et al., 2015); $\mathbf{s}_p^{\text{tree}}$ is the hidden state at the time step that generates the parent of the current node in the partial tree; $\mathbf{s}_t^{\text{word}}$ is the hidden state of the most recent time step before t that generated a subword (note that $\mathbf{s}_t^{\text{word}}$ comes from the word RNN, discussed below); and $[\cdot]$ denotes a concatenation.

Meanwhile, if at the previous time step $t-1$, TrDec did not generate a CFG rule, then the update at time step t must come from a subword being generated by the word RNN. In that case, we also update the rule RNN similarly by replacing the embedding of the CFG rule with the embedding of the subword.

Word RNN. At any time step t, if the word RNN is invoked, its hidden state $\mathbf{s}_t^{\text{word}}$ is:

$$\mathbf{s}_t^{\text{word}} = \text{LSTM}([\mathbf{s}_p^{\text{tree}}; \mathbf{w}_{t-1}; \mathbf{c}_{t-1}], \mathbf{s}_{t-1}^{\text{word}}),$$

where $\mathbf{s}_p^{\text{tree}}$ is the hidden state of rule RNN that generated the CFG rule above the current terminal; $\mathbf{w}_{t-1}$ is the embedding of the word generated at time step $t-1$; and $\mathbf{c}_{t-1}$ is the attention context computed at the previous word RNN time step $t-1$.

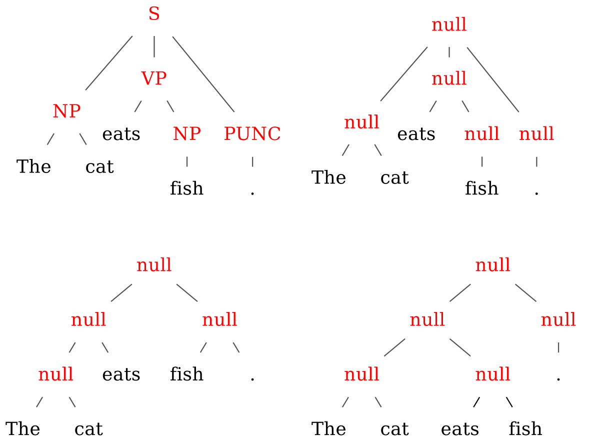

Figure 2: An example of four tree structures (Details of preterminals and subword units omitted for illustration purpose).

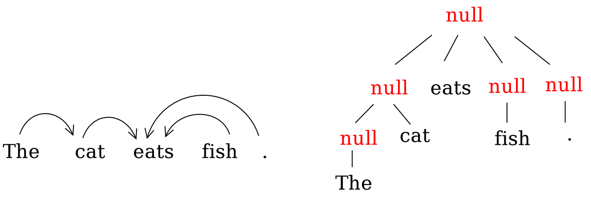

Figure 3: Conversion of a dependency tree for TrDec. *Left:* original dependency tree. *Right:* after conversion.

Softmax. At any step t, our softmax logits are $\mathbf{W} \cdot \tanh\,[\mathbf{s}_t^{\text{tree}}, \mathbf{s}_t^{\text{word}}]$, where $\mathbf{W}$ varies depending on whether a rule or a subword unit is needed.

4 Tree Structures

Unlike prior work on syntactic decoders designed for utilizing a specific type of syntactic information (Wu et al., 2017), TrDec is a flexible NMT model that can utilize any tree structure. Here we consider two categories of tree structures:

Syntactic Trees are generated using a third-party parser, such as Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007). Fig. 2 *Top Left* illustrates an example constituency parse tree. We also consider a variation of standard constituency parse trees where all of their nonterminal tags are replaced by a null tag, which is visualized in Fig. 2 *Top Right*.

In addition to constituency parse trees, TrDec can also utilize dependency parse trees via a simple procedure that converts a dependency tree into a constituency tree. Specifically, this procedure creates a parent node with null tag for each word, and then attaches each word to the parent node of its head word while preserving the word order. An example of this procedure is provided in Fig. 3.

Balanced Binary Trees are syntax-free trees constructed without any linguistic guidance. We use two slightly different versions of binary trees. Version 1 (Fig. 2 *Bottom Left*) is constructed by recursively splitting the target sentence in half and creating left and right subtrees from the left and right halves of the sentence respectively. Version 2 (Fig. 2 *Bottom Right*), is constructed by applying Version 1 on a list of nodes where consecutive words are combined together. All tree nodes in both versions have the null tag. We discuss these construction processes in more detail in Appendix A.1.

In the experiments detailed later, we evaluated TrDec with four different settings of tree structures: 1) the fully syntactic constituency parse trees; 2) constituency parse trees with null tags; 3) dependency parse trees; 4) a concatenation of both version 1 and version 2 of the binary trees, (which effectively doubles the amount of the training data and leads to slight increases in accuracy).

5 Experiments

Datasets. We evaluate TrDec on three datasets: 1) the KFTT (ja-en) dataset (Neubig, 2011), which consists of Japanese-English Wikipedia articles; 2) the IWSLT2016 German-English (de-en) dataset (Cettolo et al., 2016), which consists of TED Talks transcriptions; and 3) the LORELEI Oromo-English (or-en) dataset[2], which largely consists of texts from the Bible. Details are in Tab. 1. English sentences are parsed using Ckylark (Oda et al., 2015) for the constituency parse trees, and Stanford Parser (de Marneffe et al., 2006; Chen and Manning, 2014) for the dependency parse trees. We use byte-pair encoding (Sennrich et al., 2016) with 8K merge operations on ja-en, 4K merge operations on or-en, and 24K merge operations on de-en.

Dataset	Train	Dev	Test
ja-en	405K	1166	1160
de-en	200K	1024	1333
or-en	6.5K	358	359

Table 1: # sentences in each dataset.

Baselines. We compare TrDec against three baselines: 1) *seq2seq*: the standard *seq2seq* model with attention; 2) CCG: a syntax-aware translation model that interleaves Combinatory Categorial Grammar (CCG) tags with words on the target side of a *seq2seq* model (Nadejde et al., 2017); 3) CCG-null: the same model with CCG, but all syntactic tags are replaced by a null tag; and 4) LIN: a standard *seq2seq* model that generates linearized parse trees on the target side (Aharoni and Goldberg, 2017).

Results. Tab. 2 presents the performance of our model and the three baselines. For our model, we report the performance of TrDec-con, TrDec-con-null, TrDec-dep, and TrDec-binary (settings 1,2,3,4 in § 4). On the low-resource or-en dataset, we observe a large variance with different random seeds, so we run each model with 6 different seeds, and report the mean and standard deviation of these runs. TrDec-con-null and TrDec-con achieved comparable results, indicating that the syntactic labels have neither a large positive nor negative impact on TrDec. For ja-en and or-en, syntax-free TrDec outperforms all baselines. On de-en, TrDec loses to CCG-null, but the difference is not statistically significant ($p > 0.1$).

Model	**ja-en**	**de-en**	**or-en** (mean ± std)
seq2seq	21.10	32.26	10.90 ± 0.57
CCG	22.44	32.84	12.55 ± 0.60
CCG-null	21.31	**33.10**	11.96 ± 0.57
LIN	21.55	31.79	12.66 ± 0.61
TrDec-con	21.59	31.93	11.43 ± 0.58
TrDec-con-null	22.72	31.21	11.35 ± 0.55
TrDec-dep	21.41	31.23	8.40 ± 0.5
TrDec-binary	$\mathbf{23.14}^{*}$	32.65	$\mathbf{13.10}^{**} \pm \mathbf{0.61}$

Table 2: BLEU scores of TrDec and other baselines. Statistical significance is indicated with $*$ ($p < 0.05$) and $**$ ($p < 0.001$), compared with the best baseline.

Length Analysis. We performed a variety of analyses to elucidate the differences between the translations of different models, and the most conclusive results were through analysis based on the length of the translations. First, we categorize the ja-en test set into buckets by length of the reference sentences, and compare the models for each length category. Fig. 4 shows the gains in BLEU score over *seq2seq* for the tree-based models. Since TrDec-con outperforms TrDec-dep for all datasets, we only focus on TrDec-con for analyzing TrDec's performance with syntactic trees. The relative performance of CCG decreases on long sentences. However, TrDec, with both parse trees and syntax-free binary trees, delivers more improvement on longer sentences. This indicates that TrDec is bet-

---

[2] LDC2017E29

ter at capturing long-term dependencies during decoding. Surprisingly, TrDec-binary, which does not utilize any linguistic information, outperforms TrDec-con for all sentence length categories.

Second, Fig. 5 shows a histogram of translations by the length difference between the generated output and the reference. This provides an explanation of the difficulty of using parse trees. Ideally, this distribution will be focused around zero, indicating that the MT system is generating translations about the same length as the reference. However, the distribution of TrDec-con is more spread out than TrDec-binary, which indicates that it is more difficult for TrDec-con to generate sentences with appropriate target length. This is probably because constituency parse trees of sentences with similar number of words can have very different depth, and thus larger variance in the number of generation steps, likely making it difficult for the MT model to plan the sentence structure a-prior before actually generating the child sentences.

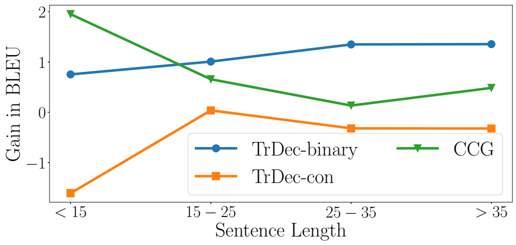

Figure 4: The gains of BLEU score over *seq2seq*.

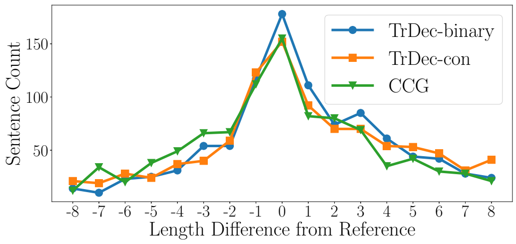

Figure 5: Distribution of length difference from reference.

6 Conclusion

We propose TrDec, a novel tree-based decoder for NMT, that generates translations along with the target side tree topology. We evaluate TrDec on both linguistically-inspired parse trees and synthetic, syntax-free binary trees. Our model, when used with synthetic balanced binary trees, outperforms CCG, the existing state-of-the-art in incorporating syntax in NMT models.

The interesting result that syntax-free trees outperform their syntax-driven counterparts elicits a natural question for future work: how do we better model syntactic structure in these models? It would also be interesting to study the effect of using source-side syntax together with the target-side syntax supported by TrDec.

Acknowledgements

This material is based upon work supported in part by the Defense Advanced Research Projects Agency Information Innovation Office (I2O) Low Resource Languages for Emergent Incidents (LORELEI) program under Contract No. HR0011-15-C0114, and the National Science Foundation under Grant No. 1815287. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *ACL*.

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured decoding with doubly recurrent neural network. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

M. Cettolo, J. Niehues, S. Stuker, L. Bentivogli, R. Cattoni, and M. Federico. 2016. The iwslt 2016 evaluation campaign. In *IWSLT*.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *ACL*.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *ACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *ACL*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *ACL*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *ACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *ACL*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *NAACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Neural Computations*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Junhui Li, Xiong Deyi, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *ACL*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Arul Menezes and Chris Quirk. 2007. Using dependency order templates to improve generality in translation. In *WMT*.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Predicting target language CCG supertags improves neural machine translation. In *WMT*.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Ckylark: A more robust pcfg-la parser. In *NAACL Software Demonstration*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *ACL*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *ACL*.

Greedy Search with Probabilistic N-gram Matching for Neural Machine Translation

Chenze Shao[1,2] **Yang Feng**[1,2*] **Xilin Chen**[1,2]
[1]Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[2]University of Chinese Academy of Science
`{shaochenze18z,fengyang,xlchen}@ict.ac.cn`

Abstract

Neural machine translation (NMT) models are usually trained with the word-level loss using the teacher forcing algorithm, which not only evaluates the translation improperly but also suffers from exposure bias. Sequence-level training under the reinforcement framework can mitigate the problems of the word-level loss, but its performance is unstable due to the high variance of the gradient estimation. On these grounds, we present a method with a differentiable sequence-level training objective based on probabilistic n-gram matching which can avoid the reinforcement framework. In addition, this method performs greedy search in the training which uses the predicted words as context just as at inference to alleviate the problem of exposure bias. Experiment results on the NIST Chinese-to-English translation tasks show that our method significantly outperforms the reinforcement-based algorithms and achieves an improvement of 1.5 BLEU points on average over a strong baseline system.

1 Introduction

Neural machine translation (NMT) (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014) has now achieved impressive performance (Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017; Hassan et al., 2018; Chen et al., 2018; Lample et al., 2018) and draws more attention. NMT models are built on the encoder-decoder framework where the encoder network encodes the source sentence to distributed representations and the decoder network reconstructs the target sentence form the representations word by word.

Currently, NMT models are usually trained with the word-level loss (i.e., cross-entropy) under the teacher forcing algorithm (Williams and Zipser, 1989), which forces the model to generate translation strictly matching the ground-truth at the word level. However, in practice it is impossible to generate translation totally the same as ground truth. Once different target words are generated, the word-level loss cannot evaluate the translation properly, usually under-estimating the translation. In addition, the teacher forcing algorithm suffers from the *exposure bias* (Ranzato et al., 2015) as it uses different inputs at training and inference, that is ground-truth words for the training and previously predicted words for the inference. Kim and Rush (2016) proposed a method of sequence-level knowledge distillation, which use teacher outputs to direct the training of student model, but the student model still have no access to its own predicted words. Scheduled sampling(SS) (Bengio et al., 2015; Venkatraman et al., 2015) attempts to alleviate the exposure bias problem through mixing ground-truth words and previously predicted words as inputs during training. However, the sequence generated by SS may not be aligned with the target sequence, which is inconsistent with the word-level loss.

In contrast, sequence-level objectives, such as BLEU (Papineni et al., 2002), GLEU (Wu et al., 2016), TER (Snover et al., 2006), and NIST (Doddington, 2002), evaluate translation at the sentence or n-gram level and allow for greater flexibility, and thus can mitigate the above problems of the word-level loss. However, due to the non-differentiable of sequence-level objectives, previous works on sequence-level training (Ranzato et al., 2015; Shen et al., 2016; Bahdanau et al., 2016; Wu et al., 2016; He et al., 2016; Wu et al., 2017; Yang et al., 2017) mainly rely on reinforcement learning algorithms (Williams, 1992; Sutton et al., 2000) to find an unbiased gradient estimator for the gradient update. Sparse rewards in this situation often cause the high variance of gradient estimation, which consequently leads to unstable

---

*Corresponding Author

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4778–4784
Brussels, Belgium, October 31 - November 4, 2018.

training and limited improvements.

Lamb et al. (2016); Gu et al. (2017); Ma et al. (2018) respectively use the discriminator, critic and bag-of-words target as sequence-level training objectives, all of which are directly connected to the generation model and hence enable direct gradient update. However, these methods do not allow for direct optimization with respect to evaluation metrics.

In this paper, we propose a method to combine the strengths of the word-level and sequence-level training, that is the direct gradient update without gradient estimation from word-level training and the greater flexibility from sequence-level training. Our method introduces probabilistic n-gram matching which makes sequence-level objectives (e.g., BLEU, GLEU) differentiable. During training, it abandons teacher forcing and performs greedy search instead to take into consideration the predicted words. Experiment results show that our method significantly outperforms word-level training with the cross-entropy loss and sequence-level training under the reinforcement framework. The experiments also indicate that greedy search strategy indeed has superiority over teacher forcing.

2 Background

NMT is based on an end-to-end framework which directly models the translation probability from the source sentence $\boldsymbol{x}$ to the target sentence $\hat{\boldsymbol{y}}$:

$$P(\hat{\boldsymbol{y}}|\boldsymbol{x}) = \prod_{j=1}^{T} p(\hat{y}_j|\hat{\boldsymbol{y}}_{<\boldsymbol{j}}, \boldsymbol{x}, \theta), \qquad (1)$$

where T is the target length and θ is the model parameters. Given the training set $D = \{\mathrm{X}^{\mathrm{M}}, \mathrm{Y}^{\mathrm{M}}\}$ with M sentences pairs, the training objective is to maximize the log-likelihood of the training data as

$$\begin{aligned} \boldsymbol{\theta} &= \arg\max_{\theta}\{\mathcal{L}(\theta)\} \\ \mathcal{L}(\theta) &= \sum_{m=1}^{M}\sum_{j=1}^{l^m} \log(p(\hat{y}_j^m|\hat{\boldsymbol{y}}_{<\boldsymbol{j}}^m, \boldsymbol{x}^m, \theta)), \end{aligned} \qquad (2)$$

where the superior m indicates the m-th sentence in the dataset and l^m is the length of m-th target sentence.

In the above model, the probability of each target word $p(\hat{y}_j^m|\hat{\boldsymbol{y}}_{<\boldsymbol{j}}^{\boldsymbol{m}}, \boldsymbol{x}^{\boldsymbol{m}}, \theta)$ is conditioned on the previous target words. The scenario is that in the training time, the teacher forcing algorithm is employed and the ground truth words from the target sentence are fed as context, while during inference, the ground truth words are not available and the previous predicted words are instead fed as context. This discrepancy is called *exposure bias*.

3 Model

3.1 Sequence-Level Objectives

Many automatic evaluation metrics of machine translation, such as BLEU, GLEU and NIST, are based on the n-gram matching. Assuming that $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are the output sentence and the ground truth sentence with length T and T' respectively, the count of an n-gram $\boldsymbol{g} = (g_1, \ldots, g_n)$ in sentence $\boldsymbol{y}$ is calculated as

$$\mathrm{C}_{\boldsymbol{y}}(\boldsymbol{g}) = \sum_{t=0}^{T-n}\prod_{i=1}^{n} 1\{g_i = y_{t+i}\}, \qquad (3)$$

where $1\{\cdot\}$ is the indicator function. The matching count of the n-gram $\boldsymbol{g}$ between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ is given by

$$\mathrm{C}_{\boldsymbol{y}}^{\hat{\boldsymbol{y}}}(\boldsymbol{g}) = \min\left(\mathrm{C}_{\boldsymbol{y}}(\boldsymbol{g}), \mathrm{C}_{\hat{\boldsymbol{y}}}(\boldsymbol{g})\right). \qquad (4)$$

Then the precision p_n and the recall r_n of the predicted n-grams are calculated as follows

$$p_n = \frac{\sum_{\boldsymbol{g}\in\boldsymbol{y}} \mathrm{C}_{\boldsymbol{y}}^{\hat{\boldsymbol{y}}}(\boldsymbol{g})}{\sum_{\boldsymbol{g}\in\boldsymbol{y}} \mathrm{C}_{\boldsymbol{y}}(\boldsymbol{g})}, \qquad (5)$$

$$r_n = \frac{\sum_{\boldsymbol{g}\in\boldsymbol{y}} \mathrm{C}_{\boldsymbol{y}}^{\hat{\boldsymbol{y}}}(\boldsymbol{g})}{\sum_{\boldsymbol{g}\in\hat{\boldsymbol{y}}} \mathrm{C}_{\hat{\boldsymbol{y}}}(\boldsymbol{g})}. \qquad (6)$$

BLEU, the most widely used metric for machine translation evaluation, is defined based on the n-gram precision as follows

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), \qquad (7)$$

where BP stands for the brevity penalty and w_n is the weight for the n-gram. In contrast, GLEU is the minimum of recall and precision of 1-4 grams where 1-4 grams are counted together:

$$\mathrm{GLEU} = \min(p_{1\text{-}4}, r_{1\text{-}4}). \qquad (8)$$

3.2 probabilistic Sequence-Level Objectives

In the output sentence $\boldsymbol{y}$, the prediction probability varies among words. Some words are translated by the model with high confidence while some words are translated with high uncertainty.

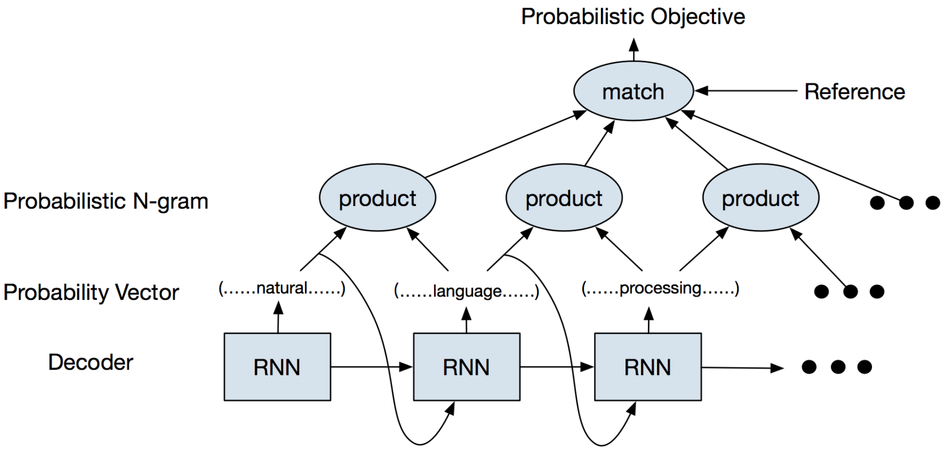

Figure 1: The overview of our model with greedy search. At each decoding step, the predicted word which has the highest probability in the probability vector is selected as context and fed into the RNN, and meanwhile this word and its probability are also used to calculate the probabilistic n-gram count.

However, when calculating the count of n-grams in Eq.(3), all the words in the output sentence are treated equally, regardless of their respective prediction probabilities.

To give a more precise description of n-gram counts which considers the variety of prediction probabilities, we use the prediction probability $p(y_j|\boldsymbol{y_{<j}},\boldsymbol{x},\theta)$ as the count of word y_j, and correspondingly the count of an n-gram is the product of these probabilistic counts of all the words in the n-gram, not *one* anymore. Then the probabilistic count of $\boldsymbol{g} = (g_1,\ldots,g_n)$ is calculated by summing over the output sentence $\boldsymbol{y}$ as

$$\widetilde{\mathrm{C}}_{\boldsymbol{y}}(\boldsymbol{g}) = \sum_{t=0}^{T-n}\prod_{i=1}^{n} 1\{g_i = y_{t+i}\}\cdot p(y_{t+i}|\boldsymbol{y_{<t+i}},\boldsymbol{x},\theta). \tag{9}$$

Now the probabilistic sequence-level objective can be got by replacing $\mathrm{C}_{\boldsymbol{y}}(\boldsymbol{g})$ with $\widetilde{\mathrm{C}}_{\boldsymbol{y}}(\boldsymbol{g})$ (the tilde over the head indicates the probabilistic version) and keeping the rest unchanged. Here, we take BLEU as an example and show how the probabilistic BLEU (denoted as P-BLEU) is defined. From this purpose, the matching count of n-gram $\boldsymbol{g}$ in Eq.(4) is modified as follows

$$\widetilde{\mathrm{C}}_{\boldsymbol{y}}^{\hat{\boldsymbol{y}}}(\boldsymbol{g}) = \min(\widetilde{\mathrm{C}}_{\boldsymbol{y}}(\boldsymbol{g}), \mathrm{C}_{\hat{\boldsymbol{y}}}(\boldsymbol{g})). \tag{10}$$

and the predict precision of n-grams changes into

$$\tilde{p}_n = \frac{\sum_{\boldsymbol{g}\in\boldsymbol{y}} \widetilde{\mathrm{C}}_{\boldsymbol{y}}^{\hat{y}}(\boldsymbol{g})}{\sum_{\boldsymbol{g}\in\boldsymbol{y}} \widetilde{\mathrm{C}}_{\boldsymbol{y}}(\boldsymbol{g})}. \tag{11}$$

Finally, the probabilistic BLEU (P-BLEU) is defined as

$$\text{P-BLEU} = \mathrm{BP}\cdot\exp(\sum_{n=1}^{N} w_n \log \tilde{p}_n), \tag{12}$$

Probabilistic GLEU (P-GLEU) can be defined in a similar way. Specifically, we denote the probabilistic precision of n-grams as P-Pn. The probabilistic precision is more reasonable than recall since the denominator in Eq.(11) plays a normalization role, so we modify the definition in Eq.(8) and define P-GLEU as simply the probabilistic precision of 1-4 grams.

The general probabilistic loss function is:

$$\mathcal{L}(\theta) = -\sum_{m=1}^{M} \mathcal{P}(\boldsymbol{y}^m, \hat{\boldsymbol{y}}^m), \tag{13}$$

where $\mathcal{P}$ represents the probabilistic sequence-level objectives, and $\boldsymbol{y}^m$ and $\hat{\boldsymbol{y}}^m$ are the predicted translation and the ground truth for the m-th sentence respectively. The calculation of the probabilistic objective is illustrated in Figure 1. This probabilistic loss can work with decoding strategies such as greedy search and teacher forcing. In this paper we employ greedy search rather than teacher forcing so as to use the previously predicted words as context and alleviate the exposure bias problem.

System	Dev(MT02)	MT03	MT04	MT05	MT06	AVG
BaseNMT	36.72	33.95	37.44	33.96	33.09	34.61
MRT	37.17	34.89	37.90	34.62	33.78	35.30
RF	37.13	34.66	37.69	34.55	33.74	35.16
P-BLEU	37.26	34.54	38.05	34.30	34.11	35.25
P-GLEU	37.44	34.67	38.11	34.24	34.58	35.40
P-P2	**38.03**	**35.45**	**39.30**	**35.10**	**34.59**	**36.11**

Table 1: Results on NIST Chinese-to-English Translation Task. **AVG** = average BLEU scores for test sets. The bold number indicates the highest score in the column.

4 Experiment

4.1 Settings

We carry out experiments on Chinese-to-English translation.[1] The training data consists of 1.25M pairs of sentences extracted from LDC corpora[2]. Sentence pairs with either side longer than 50 were dropped. We use NIST 2002 (MT 02) as the validation set and NIST 2003-2006 (MT 03-08) as the test sets. We use the case insensitive 4-gram NIST BLEU score (Papineni et al., 2002) for the translation task.

We apply our method to an attention-based NMT system (Bahdanau et al., 2014) implemented by Pytorch. Both source and target vocabularies are limited to 30K. All word embedding sizes are set to 512, and the sizes of hidden units in both encoder and decoder RNNs are also set to 512. All parameters are initialized by uniform distribution over $[-0.1, 0.1]$. The mini-batch stochastic gradient descent (SGD) algorithm is employed to train the model with batch size of 40. In addition, the learning rate is adjusted by adadelta optimizer (Zeiler, 2012) with $\rho = 0.95$ and $\epsilon = 1e$-6. Dropout is applied on the output layer with dropout rate of 0.5. The beam size is set to 10.

4.2 Performance

Systems We first pretrain the baseline model by maximum likelihood estimation (MLE) and then refine the model using probabilistic sequence-level objectives, including P-BLEU, P-GLEU and P-P2 (probabilistic 2-gram precision). In addition, we reproduce previous works which train the NMT model through minimum risk training (MRT) (Shen et al., 2016) and REINFORCE algorithm (RF) (Ranzato et al., 2015). When reproducing their works, we set BLEU, GLEU and 2-gram precision as training objectives respectively and find out that GLEU yields the best performance. In the following, we only report the results with training objective GLEU.

Performance Table 1 shows the translation performance on test sets measured in BLEU score. Simply training NMT model by the probabilistic 2-gram precision achieves an improvement of 1.5 BLEU points, which significantly outperforms the reinforcement-based algorithms. We also test the precision of other n-grams and their combinations, but do not notice significant improvements over P-P2. Notice that our method only changes the loss function, without any modification on model structure and training data.

4.3 Why Pretraining

We use the probabilistic loss to finetune the baseline model rather than training from scratch. This is in line with our motivation: to alleviate the exposure bias and make the model exposed to its own output during training. In the very beginning of the training, the model's translation capability is nearly zero and the generated sentences are often meaningless and do not contain useful information for the training, so it is unreasonable to directly apply the greedy search strategy. Therefore, we first apply the teacher forcing algorithm to pretrain the model, and then we let the model generate the sentences itself and learn from its own outputs.

Another reason favoring pretraining is that pretraining can lower the training cost. The training cost of the introduced probabilistic loss is about three times higher than the cost of cross entropy. Without pretraining, the training time will be much higher than usual. Otherwise, the training cost is acceptable if the probabilistic loss is only for finetuning.

---

[1] Experiment code: https://github.com/ictnlp/GS4NMT

[2] The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

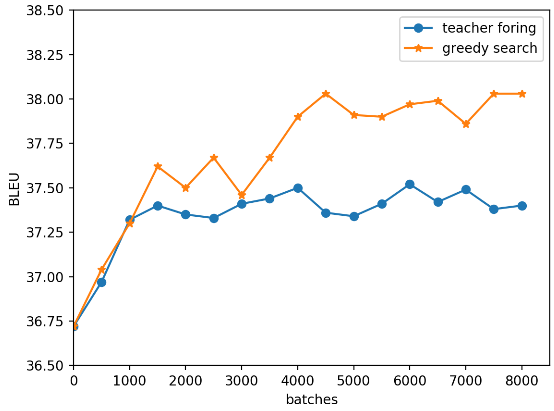

Figure 2: learning curves of different decoding strategies with training objective P-P2.

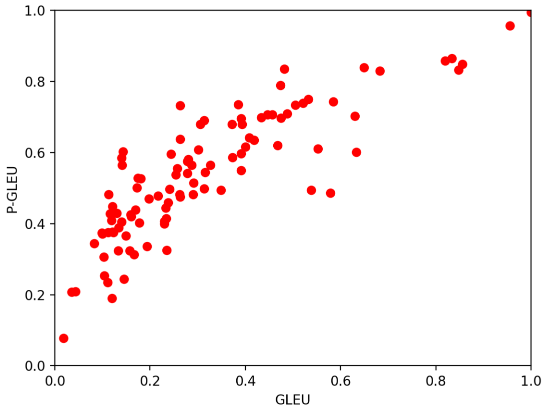

Figure 3: P-GLEU and GLEU scores on 100 pairs of sentences.

4.4 Effect of Decoding Strategy

The probabilistic loss, defined in Eq.(13), is computed from the model output $\boldsymbol{y}$ and reference $\hat{\boldsymbol{y}}$. In this section, we apply two different decoding strategies to generate $\boldsymbol{y}$: 1. teacher forcing, which uses the ground truth as decoder input. 2. greedy search, which feeds the word with maximum probability. By conducting this experiment, we attempt to figure out where the improvements come from: the modification of loss or the mitigation of exposure bias?

Figure 2 shows the learning curves of the two decoding strategies with training objective P-P2. Teacher forcing raises about 0.5 BLEU improvements and greedy search outperform the teacher forcing algorithm by nearly 1 BLEU point. We conclude that the probabilistic loss has its own advantage even when trained by the teacher forcing algorithm, and greedy search is effective in alleviating the exposure bias.

Notice that the greedy search strategy highly relys on the probabilistic loss and can not be conducted independently. Greedy search together with the word-level loss is very similar with the scheduled sampling(SS). However, SS is inconsistent with the word-level loss since the word-level loss requires strict alignment between hypothesis and reference, which can only be accomplished by the teacher forcing algorithm.

4.5 Correlation with Evaluation Metrics

In this section, we explore how the probabilistic objective correlates with the real evaluation metric. We randomly sample 100 pairs of sentences from the training set and compute their P-GLEU and GLEU scores (Wu et al. (2016) indicates that GLEU have better performance in the sentence-level evaluation than BLEU).

Directly computing the correlation between GLEU and P-GLEU gives the correlation coefficient 0.86, which indicates strong correlation. In addition, we draw the scatter diagram of the 100 pairs of sentences in Figure 3 with GLEU as x-axis and P-GLEU as y-axix. Figure 3 shows that P-GLEU correlates well with GLEU, suggesting that it is reasonable to directly train the NMT model with P-GLEU.

5 Conclusion

Word-level loss cannot evaluate the translation properly and suffers from the exposure bias, and sequence-level objectives are usually indifferentiable and require gradient estimation. We propose probabilistic sequence-level objectives based on n-gram matching, which relieve the dependence on gradient estimation and can directly train the NMT model. Experiment results show that our method significantly outperforms previous sequence-level training works and successfully alleviates the exposure bias through performing greedy search.

6 Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work was supported by the National Natural Science Foundation of China (NSFC) under the project NO.61472428 and the project NO. 61662077.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Jiatao Gu, Kyunghyun Cho, and Victor OK Li. 2017. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. *arXiv preprint arXiv:1805.04871*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1683–1692.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. 2015. Improving multi-step prediction of learned time series models. In *AAAI*, pages 3024–3030.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2017. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Exploring Recombination for Efficient Decoding of Neural Machine Translation

Zhisong Zhang[1,*], **Rui Wang**[2,†], **Masao Utiyama**[2], **Eiichiro Sumita**[2] **and Hai Zhao**[1,†]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]National Institute of Information and Communications Technology (NICT)
`zerozones17@gmail.com,`
`{wangrui, mutiyama, eiichiro.sumita}@nict.go.jp,`
`zhaohai@cs.sjtu.edu.cn`

Abstract

In Neural Machine Translation (NMT), the decoder can capture the features of the entire prediction history with neural connections and representations. This means that partial hypotheses with different prefixes will be regarded differently no matter how similar they are. However, this might be inefficient since some partial hypotheses can contain only local differences that will not influence future predictions. In this work, we introduce recombination in NMT decoding based on the concept of the "equivalence" of partial hypotheses. Heuristically, we use a simple n-gram suffix based equivalence function and adapt it into beam search decoding. Through experiments on large-scale Chinese-to-English and English-to-Germen translation tasks, we show that the proposed method can obtain similar translation quality with a smaller beam size, making NMT decoding more efficient.

1 Introduction

Recently, end-to-end Neural Machine Translation (NMT) models (Sutskever et al., 2014; Bahdanau et al., 2015) have achieved notable success. A remarkable characteristic of NMT is that the decoder, which is typically implemented using Recurrent Neural Network (RNN), can capture the features of the entire decoding history. This model does not depend on any independence assumptions and treats sequences with different prefixes as totally different hypotheses. However, many of the NMT output sequences are quite similar and they typically contain only local differences that do not influence future decoding significantly.

Src	有 消息 说 , 这 两 个 城市 的 工人 已经 成立 了 独立 工会 .
Ref	some sources said that the workers in these two *cities* have established ...
Output	according to some sources , the workers of these two *cities* have(-0.075) already(-0.331) set(-0.536) up(-0.001) ...
	according to some sources , workers of these two *cities* have(-0.073) already(-0.248) set(-0.783) up(-0.001) ...
	it has been reported that the workers of these two *cities* have(-0.058) already(-0.414) set(-0.608) up(-0.001) ...

Table 1: Example of similar partial hypotheses in beam search. The hidden layers of the partial hypotheses ending with "*cities*" correspond to the nodes boxed in Figure 1 (only three hypotheses are listed for brevity). The negative log probabilities calculated by the model for the words predicted after "*cities*" are given in parentheses.

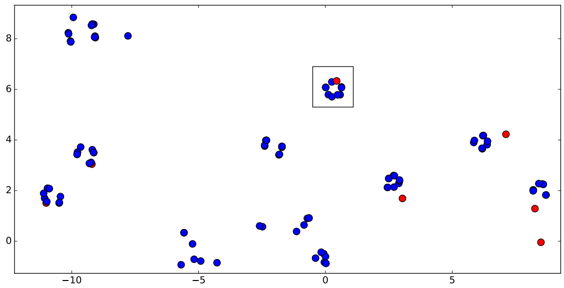

Figure 1: t-SNE visualization (Maaten and Hinton, 2008) of the recurrent hidden layer vectors for partial hypotheses for the example in Table 1. Reference and prediction hypotheses are presented as red and blue nodes, respectively. The nodes inside the box represent the hidden features of partial hypotheses ending with "*cities*".

Table 1 and Figure 1 present an example of such pattern of local differences in NMT decoding. As shown in Table 1, the three partial hypotheses that

---

[*] Zhisong Zhang was a graduate student at SJTU and a research intern at NICT when conducting this work. This work is partially supported by the program "Promotion of Global Communications Plan: Research, Development, and Social Demonstration of Multilingual Speech Translation Technology" of MIC, Japan. Hai Zhao was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. 61672343 and No. 61733011), Key Project of National Society Science Foundation of China (No. 15-ZDA041), The Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04). Thanks a lot for the helpful discussions with Kehai Chen.

[†] Co-corresponding authors

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4785–4790
Brussels, Belgium, October 31 - November 4, 2018.

Algorithm 1 Merging for Beam Search.

```
Require: list of sorted candidates $C$; beam size $k$;
    equivalence function $Eq$.
Ensure: list of candidates surviving in the beam: $C'$.
 1: $C'$ = [ ]
 2: # Scan according to the sorted order.
 3: for $c$ in $C$:
 4:     merge_flag = False
 5:     # Check previous surviving states for merging.
 6:     for $s$ in $C'$:
 7:         # Check with candidate merger states.
 8:         for $s'$ in sequence($s$):
 9:             if $Eq(c, s')$ and score($c$)<score($s'$):
10:                 merge_flag = True
11:     # Pruning by the merger.
12:     if not merge_flag:
13:         $C'$.append($c$)
14:     # Pruning by the beam size.
15:     if len($C'$) >= $k$:
16:         break
17: return $C'$
```

end with "*cities*" share similar patterns. Firstly, as shown in Figure 1, their hidden layer features are close in the latent space. Moreover, for future predictions, the model predicts identical sequences and gives similar scores for them. Although going through different paths, these partial hypotheses appear to be similar or likely equivalent.

Intuitively, for efficiency, we do not need to expand all of these partial hypotheses (states) since they have similar future predictions. In fact, this corresponds to the idea of hypothesis recombination (also known as state merging, which will be used interchangeably) from traditional Phrase-Based Statistical Machine Translation (PBSMT) (Koehn et al., 2003). Given a method to find mergeable states, we can employ recombination in NMT decoding as well.

In this paper, we adopt the mechanism of recombination in NMT decoding based on the definition of "equivalence" of partial hypotheses. Heuristically, we try a simple n-gram suffix based equivalence function and apply it to beam search without adding any neural computation cost. Through experiments on two large-scale translation tasks, we show that it can help to make the decoding more efficient.

Most recent NMT studies have focused on model improvement (Luong et al., 2015; Tu et al., 2016b; Gehring et al., 2017; Vaswani et al., 2017), and only a few have studied the search problem directly. For example, Khayrallah et al. (2017) and Stahlberg et al. (2016) explored searching on lattices generated by traditional Statistical Machine Translation (SMT). In addition, Freitag and Al-Onaizan (2017) investigated different beam search pruning strategies; however, they primarily focused on pruning candidates locally. (Niehues et al., 2017) analyzed the effects of modeling and searching, but focused on re-ranking analysis. Rather than considering candidates from other model's k-best lists, we focus on the own exploration space of a single NMT model and provide a method for more efficient searching.

2 Method

For state merging, "equivalence" should be defined from the aspect of future predictions: states with the same predictions in the future decoding process can be regarded as equivalent. We use an equivalence function $Eq(s_1, s_2)$ to denote that the two states s_1 and s_2 can be regarded as equivalent.

With the concept of equivalence, we can build the method of recombination over it. There are mainly two problems to solve:

1. How to merge states given function Eq? (§2.1)
2. How to obtain this equivalence function? (§2.2)

2.1 Search with Merging

To adopt an equivalence function $Eq(s_1, s_2)$ to merge states in a search process, we need to specify the logic of the merging mechanism. Here, without loss of generality, we specifically focus on the typical beam search.

We adopt merging in NMT beam search with a simple method: retaining the word-level search process and adding a state merger when pruning the beam at each time step. Algorithm 1 shows the proposed merging-enhanced pruning method.

Ordinary beam search only prunes candidates based on beam size (Lines 15-16), while the proposed method adds a merger to prune extra equivalent states (Lines 6-10). To manage the merging process, candidate list C are ordered[1] by model score and considered in turn. When checking equivalence for one candidate state c, we consider all current-step surviving states and their previous-step antecedences. We include previous-step states, because equivalent states may have different sequence lengths and thus not be in the same beam-search step. In Line 8, we define "sequence" as a function of obtaining the possible states that

---

[1] In plain beam search, the candidates may not need to be sorted. We use a local selector to make the sorting efficient: a local k-best selector is first applied on each previous-step candidate states, making the size of the candidate list at most $k * k$ rather than $k * |V|$, where $|V|$ is the vocabulary size.

can merge the current candidate c. If a candidate state c is not merged with any higher-ranked state, it is added to the surviving list C' (Line 13) and can possibly merge the lower-ranked ones later.

When deciding whether to merge, we also consider a criterion on model scores: we only merge state c when its score is lower than s'. Since we also consider previous-step states with different sequence lengths, a length reward λ is added for this comparison of partial hypotheses: $\text{score}(s) = \sum_{y \in s} \lambda + \log \text{p}(y)$. We also attempted length normalization, but found it performed slightly worse.

The merged partial hypotheses can be stored, and by assuming that their future predictions will be the same as their mergers, a lattice-like translation graph can be obtained. We can further extract k-best list from this structure using another beam-search on the lattice (also with length reward when comparing partial hypotheses). Note that this beam search process can be fast, since we reuse the model scores from previous search and no extra neural computations will be included.

2.2 Equivalence Function

Finding an exact equivalence function for NMT is difficult, because future predictions relies on the features from the entire previous sequence and any different sequences are not the same according to the NMT model. Here, we consider a n-gram suffix based heuristic approximation for this problem.

We adopt an approximate equivalence function:

$$\begin{aligned} Eq'(s_1, s_2) \;\equiv\; & s_1.suffix(n) = s_2.suffix(n) \\ & \wedge |s_1.length - s_2.length| < r \end{aligned}$$

Here, $suffix(n)$ represents the n-gram suffix of the sequence of a state, and r is the threshold for the length different of the two states.

This definition of equivalence only considers a subset of state features, which are inspired by PBSMT. In PBSMT, different sequences could lead to states with identical features based on n-gram suffix, and these states are exactly equivalent. Although this is not the case for NMT, the subset may encodes important and relevant features.

Although this function is simple and brings extra approximation, it has the merit of efficiency. In Algorithm 1, we can store the n-gram features of the surviving states in a hash-map and replace the for-loop checking (Line 6-10) with hashing, making the extra time-complexity O(1) for each state. During experiments, we found the extra cost brought by feature matching is far less than the cost of original neural computation.

3 Experiments and Analysis

The proposed method was evaluated on two translation tasks: NIST Chinese-English (Zh-En) and WMT English-German (En-De). For Zh-En, the training set comprised 1.4M sentences pairs from LDC corpora. NIST 02 was selected as the development set and NIST 03 to 06 were used for testing. For En-De, 4.5M WMT training data were utilized, the concatenation of newstest 2012 and 2013 was adopted as the development set, and newstest 2014 to 2016 were adopted as the test set.

We implemented[2] an attentional RNN-based NMT model and its decoder in Python with the DyNet toolkit (Neubig et al., 2017). All the experiments were carried out on one P100 GPU. For Zh-En, we set the vocabulary size of both sides to 30K, and for En-De, we adopted 50K BPE operations (Sennrich et al., 2016). The evaluation metric was tokenized BLEU (Papineni et al., 2002) calculated by `multi-bleu.perl`. Detailed settings can be found in the supplementary material.

We added a local threshold pruner to exclude unlikely words whose probabilities were less than 10% of the highest and adopted length normalization for final hypotheses ranking. For comparing partial hypotheses, the length reward λ was set to 1.0 and 0.4 for Zh-En and En-De, respectively. For the equivalence function, we utilized a suffix of 4-gram and a length difference threshold r of 2.

These hyper-parameters were set by preliminary experiments. For the length difference threshold r, we found that relatively small r like 1 or 2 was better than larger ones, which is reasonable since if the merged hypotheses differs too much in length, there are higher chances that they covered different information. For n-gram suffix, we found smaller n-grams made more bad merges and 4-gram is a reasonably good choice, slightly larger ones gave slightly worse results and also less chances of recombination.

3.1 Results

Figure 2 show the results of various beam sizes on the concatenation of all test sets. Separate results are given in the supplementary material.

As shown by the speed curves, merging adds little extra cost (less than 10%) to decoding at

---

[2] https://github.com/zzsfornlp/znmt-merge

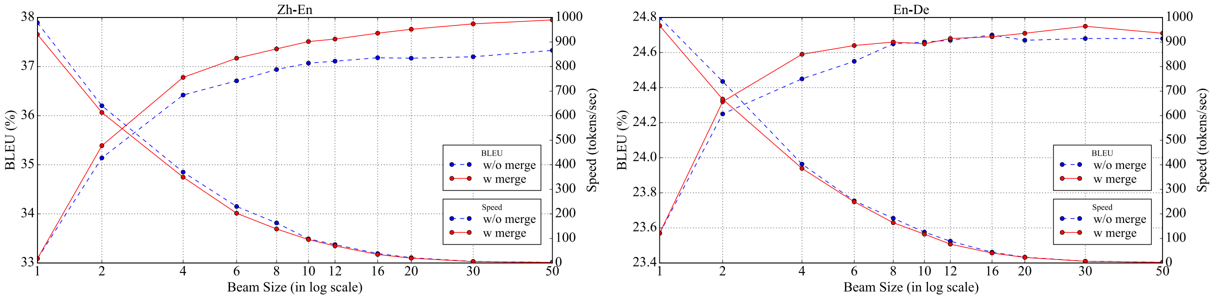

Figure 2: Translation quality and speed of Zh-En and En-De test sets (5453 sentences by concatenating NIST 03 to 06 and 8171 sentences by concatenating newstest 2014 to 2016, respectively).

the same beam size. Moreover, since bringing no extra neural computations, the proposed merging mechanism is transparent to neural architectures and easy to adopt. In our experiments, we used batched decoding on GPU and merging did not influence the efficiency of this implementation.

For translation quality, the results indicate that the proposed methods can yield improvements at various beam sizes for Zh-En and small beam sizes for En-De. Moreover, in some way, merging can make the search more efficient. For example, in both datasets, merge-enhanced searchers with beam-size 6 can obtain comparable or better results compared to those of ordinary searchers with beam-size 12 (on BLEU, 37.17 vs. 37.11 for Zh-En, 24.64 vs. 24.67 for En-De). As for decoding speed, the one of beam-size 6 can be more than twice of the one of beam-size 12 (over 200 tokens/second vs. around 100 tokens/second). That is to say, with merging, we can achieve similar translation quality with a smaller beam size, which leads to higher decoding speed.

The results show that for large beam sizes, expanding explored search space by increasing beam size or adopting merging helps more in Zh-En than En-De. A possible explanation for this is that in NIST Zh-En dataset, each source sentences has four references for evaluation, which encourages the diversity brought by expanding reached search space. In Table 2, we compare the BLEU scores with multiple and single references on several beam sizes, and the single-reference results does not always increase along the beam size like the multiple ones. The En-De dataset also has only one reference and is similar to this case.

The results also show that expanding explored search space does not always bring improvements. This concerns more on modeling than searching and corresponds with previous findings on the relations between NMT searching and modeling (Tu et al., 2016a; Niehues et al., 2017; Li et al., 2018).

Ref \ Beam	10	16	30	50
Multi-Ref	37.07	37.17	37.19	37.32
Single-Ref	18.23	18.29	18.23	18.26

Table 2: Comparisons of multi- and single-reference BLEU scores of NIST 03-06 with "w/o merging". "Multi-Ref" uses all four references, and "Single-Ref" only uses the second one, whose evaluations disagree most with "Multi-ref".

The potential of the proposed method might be better realized with improved NMT models.

3.2 Analysis

We further analyzed the merge-enhanced search process. For these analyses, we mainly checked decoding with a beam size of 10 on Zh-En dataset.

Frequency of Merging First, we investigated how often recombination occurs and how much it expands the explored output space. For a beam size of 10, with influences from the local pruner and the proposed merger, the average expanding size is 7.60 for each step, and the average number of merger-pruned partial hypotheses is 0.61 per step (22.5 per sentence). This indicates that a partial hypothesis is recombined in every two steps. The output translation graph can hold much more output space than the original k-best list, and we found that on average the possible output sequences were averagely 200 times the beam size. Figure 3 shows an example of the output translation graph.

Merging and Similarity of Hidden States It is nearly impossible to explore such a large space with an exact NMT model; thus, we depend on the assumption that merged hypotheses have nearly the same features. To evaluate this assumption, we calculated the similarity between the hidden layers of the merged partial hypotheses. Among the 122772 merge points in 5453 Zh-En sentences, the average cosine similarity (in range $[-1, 1]$) was 0.986, which indicates that the recombinations are reasonable. In addition, we tried adding simple cosine similarity constraints (using another

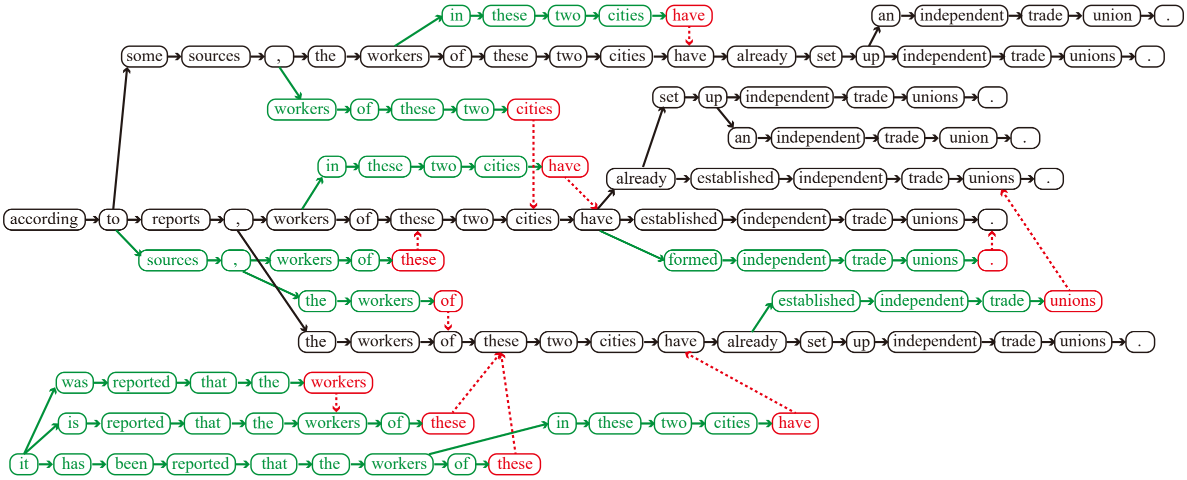

Figure 3: An example output translation graph. The red nodes and dashed arrows indicate merge points.

Beam	w/ merge	w/o merge
1	1.6% / 90.6%	1.6% / 90.6%
4	8.0% / 51.3%	5.2% / 54.8%
10	29.1% / 15.7%	–
16	44.5% / 11.8%	28.2% / 5.6%
30	64.6% / 20.3%	56.4% / 20.2%

Table 3: Comparisons of prediction model scores between different searching settings and a basic setting, which is "Beam=10, w/o merge". The pattern "$a\%$ / $b\%$" means that compared with the basic setting, $a\%$ of the sentences get higher model scores and $b\%$ get lower ones. For the rest (1-$a\%$-$b\%$), they give identical predictions.

threshold) in the equivalence function, however, we found that this does not bring obvious additional benefits.

Effects of Merging We further conducted comparisons between the predictions of ordinary and merge-enhanced beam search. First, we investigated the model scores of their predictions. As shown in Table 3, we selected "Beam=10, no merge" as the basic setting, and compared the predictions of other settings with it. Overall, the merge-enhanced searcher can obtain higher model score predictions, which suggests its stronger search ability, because the goal of searching is to return hypotheses with higher model scores.

Moreover, we tried a re-ranking experiment on 100-best lists with 4-checkpoint-model-ensemble, and only found similar slight improvements for plain and merge-enhanced search. Nevertheless, since merge-enhanced search can obtain a output translation graph, we expect that the graph can contain more diverse hypotheses.

To verify this, we compared the oracle BLEU scores within the reached space. To extract oracle hypotheses from the translation graphs, we simply adopted approximate Partial BLEU Oracle (Dreyer et al., 2007; Sokolov et al., 2012). Merge-based searcher could obtain an oracle score of 47.83, while ordinary beam searcher could only get 42.57. Only by increasing the beam size up to 100 could the ordinary beam searcher achieve a better result of 48.74. This indicates that recombination helps to touch more output space.

4 Conclusion and Discussion

In this work, 1) we show that decoding with heuristic recombination can obtain similar translation qualities with smaller beam sizes, thus increasing efficiency, and, 2) we empirically explore the decoding process and analyze the influences of recombination from various aspects.

Although the improvements brought by recombination depend on careful refinements of the model, this concerns more on modeling, since the goal of decoding is to find hypotheses with higher model scores. The potential of recombination may be further realized by improving how the output sequences are modeled. Another interesting topic will be the combination with SMT or extra larger language models (Wang et al., 2013, 2014).

For the equivalence function, there can also be extensions. For example, a model-based equivalence function can be trained by using the neural features (hidden layers in RNN). However, model-based equivalence functions may bring extra neural computation cost and be harder to efficiently implemented. In this work, we focus on the merging mechanism and leave the study of equivalence function for future work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Markus Dreyer, Keith Hall, and Sanjeev Khudanpur. 2007. Comparing reordering constraints for smt using efficient bleu oracle computation. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 103–110, Rochester, New York.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 123–135, Vancouver, Canada.

Huda Khayrallah, Gaurav Kumar, Kevin Duh, Matt Post, and Philipp Koehn. 2017. Neural lattice search for domain adaptation in machine translation. In *Proceedings of IJCNLP*, pages 20–25, Taipei, Taiwan.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54.

Yanyang Li, Tong Xiao, Yinqiao Li, Qiang Wang, Changming Xu, and Jingbo Zhu. 2018. A simple and effective approach to coverage-aware neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 292–297.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2017. Analyzing neural mt search and model performance. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 11–17, Vancouver.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, Berlin, Germany.

Artem Sokolov, Guillaume Wisniewski, and Francois Yvon. 2012. Computing lattice bleu oracle scores for machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 120–129, Avignon, France.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 299–305, Berlin, Germany.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2016a. Neural machine translation with reconstruction. In *Proceedings of AAAI*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016b. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 76–85, Berlin, Germany.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

Rui Wang, Masao Utiyama, Isao Goto, Eiichro Sumita, Hai Zhao, and Bao-Liang Lu. 2013. Converting continuous-space language models into n-gram language models for statistical machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2014. Neural network based bilingual language model growing for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.

Has Machine Translation Achieved Human Parity?
A Case for Document-level Evaluation

Samuel Läubli[1] **Rico Sennrich**[1,2] **Martin Volk**[1]

[1]Institute of Computational Linguistics, University of Zurich
{laeubli,volk}@cl.uzh.ch

[2]School of Informatics, University of Edinburgh
rico.sennrich@ed.ac.uk

Abstract

Recent research suggests that neural machine translation achieves parity with professional human translation on the WMT Chinese–English news translation task. We empirically test this claim with alternative evaluation protocols, contrasting the evaluation of single sentences and entire documents. In a pairwise ranking experiment, human raters assessing adequacy and fluency show a stronger preference for human over machine translation when evaluating documents as compared to isolated sentences. Our findings emphasise the need to shift towards document-level evaluation as machine translation improves to the degree that errors which are hard or impossible to spot at the sentence-level become decisive in discriminating quality of different translation outputs.

1 Introduction

Neural machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015) has become the de-facto standard in machine translation, outperforming earlier phrase-based approaches in many data settings and shared translation tasks (Luong and Manning, 2015; Sennrich et al., 2016; Cromieres et al., 2016). Some recent results suggest that neural machine translation "approaches the accuracy achieved by average bilingual human translators [on some test sets]" (Wu et al., 2016), or even that its "translation quality is at human parity when compared to professional human translators" (Hassan et al., 2018). Claims of human parity in machine translation are certainly extraordinary, and require extraordinary evidence.[1] Laudably, Hassan et al. (2018) have released their data publicly to allow external validation of their claims. Their claims are further strengthened by the fact that they follow best practices in human machine translation evaluation, using evaluation protocols and tools that are also used at the yearly Conference on Machine Translation (WMT) (Bojar et al., 2017), and take great care in guarding against some confounds such as test set selection and rater inconsistency.

However, the implications of a statistical tie between two machine translation systems in a shared translation task are less severe than that of a statistical tie between a machine translation system and a professional human translator, so we consider the results worthy of further scrutiny. We perform an independent evaluation of the professional translation and best machine translation system that were found to be of equal quality by Hassan et al. (2018). Our main interest lies in the evaluation protocol, and we empirically investigate if the lack of document-level context could explain the inability of human raters to find a quality difference between human and machine translations. We test the following hypothesis:

> A professional translator who is asked to rank the quality of two candidate translations on the document level will prefer a professional human translation over a machine translation.

Note that our hypothesis is slightly different from that tested by Hassan et al. (2018), which could be phrased as follows:

> A bilingual crowd worker who is asked to directly assess the quality of candidate translations on the sentence level will prefer a professional human translation over a machine translation.

---

[1] The term "parity" may raise the expectation that there is evidence for equivalence, but the term is used in the definition of "there [being] no statistical significance between [two outputs] for a test set of candidate translations" by Hassan et al. (2018). Still, we consider this finding noteworthy given the strong evaluation setup.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4791–4796
Brussels, Belgium, October 31 - November 4, 2018.

As such, our evaluation is not a direct replication of that by Hassan et al. (2018), and a failure to reproduce their findings does not imply an error on either our or their part. Rather, we hope to indirectly assess the accuracy of different evaluation protocols. Our underlying assumption is that professional human translation is still superior to neural machine translation, but that the sensitivity of human raters to these quality differences depends on the evaluation protocol.

2 Human Evaluation of Machine Translation

Machine translation is typically evaluated by comparing system outputs to source texts, reference translations, other system outputs, or a combination thereof (for examples, see Bojar et al., 2016a). The scientific community concentrates on two aspects: adequacy, typically assessed by bilinguals; and target language fluency, typically assessed by monolinguals. Evaluation protocols have been subject to controversy for decades (e. g., Van Slype, 1979), and we identify three aspects with particular relevance to assessing human parity: granularity of measurement (ordinal vs. interval scales), raters (experts vs. crowd workers), and experimental unit (sentence vs. document).

2.1 Related Work

Granularity of Measurement Callison-Burch et al. (2007) show that ranking (*Which of these translations is better?*) leads to better inter-rater agreement than absolute judgement on 5-point Likert scales (*How good is this translation?*) but gives no insight about how much a candidate translation differs from a (presumably perfect) reference. To this end, Graham et al. (2013) suggest the use of continuous scales for direct assessment of translation quality. Implemented as a slider between 0 (*Not at all*) and 100 (*Perfectly*), their method yields scores on a 100-point interval scale in practice (Bojar et al., 2016b, 2017), with each raters' rating being standardised to increase homogeneity. Hassan et al. (2018) use source-based direct assessment to avoid bias towards reference translations. In the shared task evaluation by Cettolo et al. (2017), raters are shown the source and a candidate text, and asked: *How accurately does the above candidate text convey the semantics of the source text?* In doing so, they have translations produced by humans and machines rated independently, and parity is assumed if the mean score of the former does not significantly differ from the mean score of the latter.

Raters To optimise cost, machine translation quality is typically assessed by means of crowdsourcing. Combined ratings of bilingual crowd workers have been shown to be more reliable than automatic metrics and "very similar" to ratings produced by "experts"[2] (Callison-Burch, 2009). Graham et al. (2017) compare crowdsourced to "expert" ratings on machine translations from WMT 2012, concluding that, with proper quality control, "machine translation systems can indeed be evaluated by the crowd alone." However, it is unclear whether this finding carries over to translations produced by NMT systems where, due to increased fluency, errors are more difficult to identify (Castilho et al., 2017a), and concurrent work by Toral et al. (2018) highlights the importance of expert translators for MT evaluation.

Experimental Unit Machine translation evaluation is predominantly performed on single sentences, presented to raters in random order (e. g., Bojar et al., 2017; Cettolo et al., 2017). There are two main reasons for this. The first is cost: if raters assess entire documents, obtaining the same number of data points in an evaluation campaign multiplies the cost by the average number of sentences per document. The second is experimental validity. When comparing systems that produce sentences without considering document-level context, the perceived suprasentential cohesion of a system output is likely due to randomness and thus a confounding factor. While incorporating document-level context into machine translation systems is an active field of research (Webber et al., 2017), state-of-the-art systems still operate at the level of single sentences (Sennrich et al., 2017; Vaswani et al., 2017; Hassan et al., 2018). In contrast, human translators can and do take document-level context into account (Krings, 1986). The same holds for raters in evaluation campaigns. In the discussion of their results, Wu et al. (2016) note that their raters "[did] not necessarily fully understand each randomly sampled sentence sufficiently" because it was provided with no context. In such setups, raters cannot reward textual cohesion and coherence.

---

[2] "Experts" here are computational linguists who develop MT systems, who may not be expert translators.

2.2 Our Evaluation Protocol

We conduct a quality evaluation experiment with a 2×2 mixed factorial design, testing the effect of source text availability (adequacy, fluency) and experimental unit (sentence, document) on ratings by professional translators.

Granularity of Measurement We elicit judgements by means of pairwise ranking. Raters choose the better (with ties allowed) of two translations for each item: one produced by a professional translator (HUMAN), the other by machine translation (MT). Since our evaluation includes that of human translation, it is reference-free. We evaluate in two conditions: adequacy, where raters see source texts and translations (*Which translation expresses the meaning of the source text more adequately?*); and fluency, where raters only see translations (*Which text is better English?*).

Raters We recruit professional translators, only considering individuals with at least three years of professional experience and positive client reviews.

Experimental Unit To test the effect of context on perceived translation quality, raters evaluate entire documents as well as single sentences in random order (i. e., context is a within-subjects factor). They are shown both translations (HUMAN and MT) for each unit; the source text is only shown in the adequacy condition.

Quality Control To hedge against random ratings, we convert 5 documents and 16 sentences per set into spam items (Kittur et al., 2008): we render one of the two options nonsensical by shuffling its words randomly, except for 10 % at the beginning and end.

Statistical Analysis We test for statistically significant preference of HUMAN over MT or vice versa by means of two-sided Sign Tests. Let a be the number of ratings in favour of MT, b the number of ratings in favour of HUMAN, and t the number of ties. We report the number of successes x and the number of trials n for each test, such that $x = b$ and $n = a + b$.[3]

2.3 Data Collection

We use the experimental protocol described in the previous section for a quality assessment of Chinese to English translations of news articles. To this end, we randomly sampled 55 documents and 2×120 sentences from the WMT 2017 test set. We only considered the 123 articles (documents) which are native Chinese,[4] containing 8.13 sentences on average. Human and machine translations (REFERENCE-HT as HUMAN, and COMBO-6 as MT) were obtained from data released by Hassan et al. (2018).[5]

The sampled documents and sentences were rated by professional translators we recruited from ProZ:[6] 4 native in Chinese (2), English (1), or both (1) to rate adequacy, and 4 native in English to rate fluency. On average, translators had 13.7 years of experience and 8.8 positive client reviews on ProZ, and received US$ 188.75 for rating 55 documents and 120 sentences.

The averages reported above include an additional translator we recruited when one rater showed poor performance on document-level spam items in the fluency condition, whose judgements we exclude from analysis. We also exclude sentence-level results from 4 raters because there was overlap with the documents they annotated, which means that we cannot rule out that the sentence-level decisions were informed by access to the full document. To allow for external validation and further experimentation, we make all experimental data publicly available.[7]

3 Results

In the adequacy condition, MT and HUMAN are not statistically significantly different on the sentence level ($x{=}86$, $n{=}189$, $p{=}.244$). This is consistent with the results Hassan et al. (2018) obtained with an alternative evaluation protocol (crowdsourcing and direct assessment; see Section 2.1). However, when evaluating entire doc-

---

[3] Emerson and Simon (1979) suggest the inclusion of ties such that $x = b + 0.5t$ and $n = a + b + t$. This modification has no effect on the significance levels reported in this paper.

[4] While it is common practice in machine translation to use the same test set in both translation directions, we consider a direct comparison between human "translation" and machine translation hard to interpret if one is in fact the original English text, and the other an automatic translation into English of a human translation into Chinese. In concurrent work, Toral et al. (2018) expand on the confounding effect of evaluating text where the target side is actually the original document.

[5] `http://aka.ms/Translator-HumanParityData`

[6] `https://www.proz.com`

[7] `https://github.com/laeubli/parity`

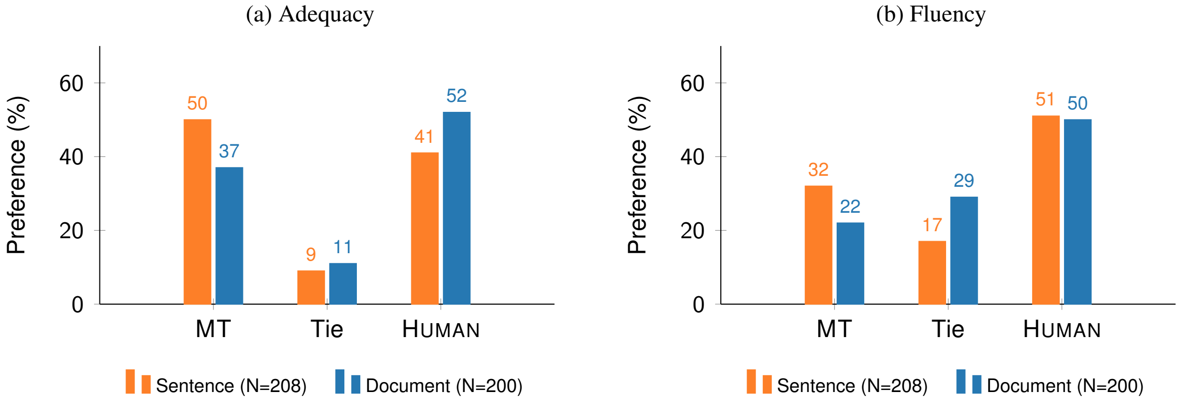

Figure 1: Raters prefer human translation more strongly in entire documents. When evaluating isolated sentences in terms of adequacy, there is no statistically significant difference between HUMAN and MT; in all other settings, raters show a statistically significant preference for HUMAN.

uments, raters show a statistically significant preference for HUMAN ($x=104$, $n=178$, $p<.05$). While the number of ties is similar in sentence- and document-level evaluation, preference for MT drops from 50 to 37 % in the latter (Figure 1a).

In the fluency condition, raters prefer HUMAN on both the sentence ($x=106$, $n=172$, $p<.01$) and document level ($x=99$, $n=143$, $p<.001$). In contrast to adequacy, fluency ratings in favour of HUMAN are similar in sentence- and document-level evaluation, but raters find more ties with document-level context as preference for MT drops from 32 to 22 % (Figure 1b).

We note that these large effect sizes lead to statistical significance despite modest sample size. Inter-annotator agreement (Cohen's κ) ranges from 0.13 to 0.32 (see Appendix for full results and discussion).

4 Discussion

Our results emphasise the need for suprasentential context in human evaluation of machine translation. Starting with Hassan et al.'s (2018) finding of no statistically significant difference in translation quality between HUMAN and MT for their Chinese–English test set, we set out to test this result with an alternative evaluation protocol which we expected to strengthen the ability of raters to judge translation quality. We employed professional translators instead of crowd workers, and pairwise ranking instead of direct assessment, but in a sentence-level evaluation of adequacy, raters still found it hard to discriminate between HUMAN and MT: they did not show a statistically significant preference for either of them.

Conversely, we observe a tendency to rate HUMAN more favourably on the document level than on the sentence level, even within single raters. Adequacy raters show a statistically significant preference for HUMAN when evaluating entire documents. We hypothesise that document-level evaluation unveils errors such as mistranslation of an ambiguous word, or errors related to textual cohesion and coherence, which remain hard or impossible to spot in a sentence-level evaluation. For a subset of articles, we elicited both sentence-level and document-level judgements, and inspected articles for which sentence-level judgements were mixed, but where HUMAN was strongly preferred in document-level evaluation. In these articles, we do indeed observe the hypothesised phenomena. We find an example of lexical coherence in a 6-sentence article about a new app "微信挪车", which HUMAN consistently translates into "WeChat Move the Car". In MT, we find three different translations in the same article: "Twitter Move Car", "WeChat mobile", and "WeChat Move". Other observations include the use of more appropriate discourse connectives in HUMAN, a more detailed investigation of which we leave to future work.

To our surprise, fluency raters show a stronger preference for HUMAN than adequacy raters (Figure 1). The main strength of neural machine translation in comparison to previous statistical approaches was found to be increased fluency, while adequacy improvements were less clear (Bojar et al., 2016b; Castilho et al., 2017b), and we expected a similar pattern in our evaluation. Does this indicate that adequacy is in fact a strength of

MT, not fluency? We are wary to jump to this conclusion. An alternative interpretation is that MT, which tends to be more literal than HUMAN, is judged more favourably by raters in the bilingual condition, where the majority of raters are native speakers of the source language, because of L1 interference. We note that the availability of document-level context still has a strong impact in the fluency condition (Section 3).

5 Conclusions

In response to recent claims of parity between human and machine translation, we have empirically tested the impact of sentence and document level context on human assessment of machine translation. Raters showed a markedly stronger preference for human translations when evaluating at the level of documents, as compared to an evaluation of single, isolated sentences.

We believe that our findings have several implications for machine translation research. Most importantly, if we accept our interpretation that human translation is indeed of higher quality in the dataset we tested, this points to a failure of current best practices in machine translation evaluation. As machine translation quality improves, translations will become harder to discriminate in terms of quality, and it may be time to shift towards document-level evaluation, which gives raters more context to understand the original text and its translation, and also exposes translation errors related to discourse phenomena which remain invisible in a sentence-level evaluation.

Our evaluation protocol was designed with the aim of providing maximal validity, which is why we chose to use professional translators and pairwise ranking. For future work, it would be of high practical relevance to test whether we can also elicit accurate quality judgements on the document-level via crowdsourcing and direct assessment, or via alternative evaluation protocols. The data released by Hassan et al. (2018) could serve as a test bed to this end.

One reason why document-level evaluation widens the quality gap between machine translation and human translation is that the machine translation system we tested still operates on the sentence level, ignoring wider context. It will be interesting to explore to what extent existing and future techniques for document-level machine translation can narrow this gap. We expect that this will require further efforts in creating document-level training data, designing appropriate models, and supporting research with discourse-aware automatic metrics.

Acknowledgements

We thank Xin Sennrich for her help with the analysis of translation errors. We also thank Antonio Toral and the anonymous reviewers for their helpful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*, San Diego, CA.

Ondrej Bojar, Christian Federmann, Barry Haddow, Philipp Koehn, Matt Post, and Lucia Specia. 2016a. Ten years of WMT evaluation campaigns: Lessons learnt. In *Proceedings of the LREC 2016 Workshop "Translation Evaluation – From Fragmented Tools and Data Sets to an Integrated Ecosystem"*, pages 27–34, Portorož, Slovenia.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of WMT*, pages 169–214, Copenhagen, Denmark.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016b. Findings of the 2016 Conference on Machine Translation (WMT16). In *Proceedings of WMT*, pages 131–198, Berlin, Germany.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using amazon's mechanical turk. In *Proceedings of EMNLP*, pages 286–295, Singapore.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (Meta-) evaluation of machine translation. In *Proceedings of WMT*, pages 136–158, Prague, Czech Republic.

Sheila Castilho, Joss Moorkens, Federico Gaspari, Iacer Calixto, John Tinsley, and Andy Way. 2017a. Is Neural Machine Translation the New State of the Art? *The Prague Bulletin of Mathematical Linguistics*, 108:109–120.

Sheila Castilho, Joss Moorkens, Federico Gaspari, Rico Sennrich, Vilelmini Sosoni, Yota Georgakopoulou, Pintu Lohar, Andy Way, Antonio Valerio Miceli Barone, and Maria Gialama. 2017b. A Comparative Quality Evaluation of PBSMT and NMT using Professional Translators. In *Proceedings of MT Summit*, Nagoya, Japan.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuitho Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 Evaluation Campaign. In *Proceedings of IWSLT*, pages 2–14, Tokyo, Japan.

Fabien Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto University participation to WAT 2016. In *Proceedings of WAT*, pages 166–174, Osaka, Japan.

John D. Emerson and Gary A. Simon. 1979. Another Look at the Sign Test When Ties Are Present: The Problem of Confidence Intervals. *The American Statistician*, 33(3):140–142.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous Measurement Scales in Human Evaluation of Machine Translation. In *Proceedings of the 7th Linguistic Annotation Workshop & Interoperability with Discourse*, pages 33–41, Sofia, Bulgaria.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2017. Can machine translation systems be evaluated by the crowd alone? *Natural Language Engineering*, 23(1):3–30.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving Human Parity on Automatic Chinese to English News Translation. *Computing Research Repository*, arXiv:1803.05567.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of EMNLP*, pages 1700–1709, Seattle, WA.

Aniket Kittur, Ed H. Chi, and Bongwon Suh. 2008. Crowdsourcing User Studies with Mechanical Turk. In *Proceedings of CHI*, pages 453–456, Florence, Italy.

Hans P. Krings. 1986. *Was in den Köpfen von Übersetzern vorgeht*. Gunter Narr, Tübingen, Germany.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domains. In *Proceedings of IWSLT*, Da Nang, Vietnam.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The University of Edinburgh's Neural MT Systems for WMT17. In *Proceedings of WMT*, pages 389–399, Copenhagen, Denmark.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of WMT*, pages 368–373, Berlin, Germany.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of NIPS*, pages 3104–3112, Montreal, Canada.

Antonio Toral, Sheila Castilho, Ke Hu, and Andy Way. 2018. Attaining the Unattainable? Reassessing Claims of Human Parity in Neural Machine Translation. In *Proceedings of WMT*, Brussels, Belgium.

Georges Van Slype. 1979. Critical study of methods for evaluating the quality of machine translation. Research report BR 19142 prepared for the Commission of the European Communities, Bureau Marcel van Dijk, Brussels, Belgium.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proceedings of NIPS*, pages 5998–6008, Long Beach, CA.

Bonnie Webber, Andrei Popescu-Belis, and Jörg Tiedemann, editors. 2017. *Proceedings of the Third Workshop on Discourse in Machine Translation*. Copenhagen, Denmark.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *Computing Research Repository*, arXiv:1609.08144.

Automatic Reference-Based Evaluation of Pronoun Translation Misses the Point

Liane Guillou[*]
University of Edinburgh
School of Informatics
Scotland, United Kingdom
lguillou@inf.ed.ac.uk

Christian Hardmeier[*]
Uppsala University
Dept. of Linguistics & Philology
Uppsala, Sweden
christian.hardmeier@lingfil.uu.se

Abstract

We compare the performance of the APT and AutoPRF metrics for pronoun translation against a manually annotated dataset comprising human judgements as to the correctness of translations of the PROTEST test suite. Although there is some correlation with the human judgements, a range of issues limit the performance of the automated metrics. Instead, we recommend the use of semi-automatic metrics and test suites in place of fully automatic metrics.

1 Introduction

As the general quality of machine translation (MT) increases, there is a growing interest in improving the translation of specific linguistic phenomena. A case in point that has been studied in the context of both statistical (Hardmeier, 2014; Guillou, 2016; Loáiciga, 2017) and neural MT (Bawden et al., 2017; Voita et al., 2018) is that of pronominal anaphora. In the simplest case, translating anaphoric pronouns requires the generation of corresponding word forms respecting the grammatical constraints on agreement in the target language, as in the following English-French example, where the correct form of the pronoun in the second sentence varies depending on which of the (equally correct) translations of the word *bicycle* was used in the first:

(1) a. I have a bicycle. **It** is red.
b. J'ai un vélo. **Il** est rouge. [ref]
c. J'ai une bicyclette. **Elle** est rouge. [MT]

However, the problem is more complex in practice because there is often no $1:1$ correspondence between pronouns in two languages. This is easily demonstrated at the corpus level by observing that the number of pronouns varies significantly across languages in parallel texts (Mitkov and Barbu, 2003), but it tends to be difficult to predict in individual cases.

In general MT research, significant progress was enabled by the invention of automatic evaluation metrics based on reference translations, such as BLEU (Papineni et al., 2002). Attempting to create a similar framework for efficient research, researchers have proposed automatic reference-based evaluation metrics specifically targeting pronoun translation: AutoPRF (Hardmeier and Federico, 2010) and APT (Miculicich Werlen and Popescu-Belis, 2017). We study the performance of these metrics on a dataset of English-French translations and investigate to what extent automatic evaluation based on reference translations provides insights into how well an MT system handles pronouns. Our analysis clarifies the conceptual differences between AutoPRF and APT, uncovering weaknesses in both metrics, and investigates the effects of the alignment correction heuristics used in APT. By using the fine-grained PROTEST categories of pronoun function, we find that the accuracy of the automatic metrics varies across pronouns of different functions, suggesting that certain linguistic patterns are captured better in the automatic evaluation than others. We argue that fully automatic wide-coverage evaluation of this phenomenon is unlikely to drive research forward, as it misses essential parts of the problem despite achieving some correlation with human judgements. Instead, semi-automatic evaluation involving automatic identification of correct translations with high precision and low recall appears to be a more achievable goal. Another more realistic option is a test suite evaluation with a very limited scope.

2 Pronoun Evaluation Metrics for MT

Two reference-based automatic metrics of pronoun translation have been proposed in the literature.

---

[*] Both authors contributed equally.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4797–4802
Brussels, Belgium, October 31 - November 4, 2018.

The first (Hardmeier and Federico, 2010) is a variant of precision, recall and F-score that measures the overlap of pronouns in the MT output with a reference translation. It lacks an official name, so we refer to it as *AutoPRF* following the terminology of the DiscoMT 2015 shared task (Hardmeier et al., 2015). The scoring process relies on a word alignment between the source and the MT output, and between the source and the reference translation. For each input pronoun, it computes a *clipped count* (Papineni et al., 2002) of the overlap between the aligned tokens in the reference and the MT output. The clipped count of a given word is defined as the number of times it occurs in the MT output, limited by the number of times it occurs in the reference translation. The final metric is then calculated as the precision, recall and F-score based on these clipped counts.

Miculicich Werlen and Popescu-Belis (2017) propose a metric called *Accuracy of Pronoun Translation (APT)* that introduces several innovations over the previous work. It is a variant of accuracy, so it counts, for each source pronoun, whether its translation can be considered correct, without considering multiple alignments. Since word alignment is problematic for pronouns, the authors propose an heuristic procedure to improve alignment quality. Finally, it introduces the notion of pronoun equivalence, assigning partial credit to pronoun translations that differ from the reference translation in specific ways deemed to be acceptable. In particular, it considers six possible cases when comparing the translation of a pronoun in MT output and the reference. The pronouns may be: (1) *identical*, (2) *equivalent*, (3) *different/incompatible*, or there may be no translation in: (4) the MT output, (5) the reference, (6) either the MT output or the reference. Each of these cases may be assigned a weight between 0 and 1 to determine the level of *correctness*.

3 The PROTEST Dataset

We study the behaviour of the two automatic metrics using the PROTEST test suite (Guillou and Hardmeier, 2016). The test suite comprises 250 hand-selected personal pronoun tokens taken from the *DiscoMT2015.test* dataset of TED talk transcriptions and translations (Hardmeier et al., 2016) and annotated according to the ParCor guidelines (Guillou et al., 2014). It is structured according to a linguistic typology motivated by work on functional grammar by Dik (1978) and Halliday (2004). Pronouns are first categorised according to their *function*:

anaphoric: I have a bicycle. **It** is red.
event: He lost his job. **It** was a shock.
pleonastic: **It** is raining.
addressee reference: **You**'re welcome.

They are then subcategorised according to morphosyntactic criteria, whether the antecedent is a group noun, whether the ancedent is in the same or a different sentence, and whether an addressee reference pronoun refers to one or more specific people (deictic) or to people in general (generic).

Our dataset contains human judgements on the performance of nine MT systems on the translation of the 250 pronouns in the PROTEST test suite. The systems include five submissions to the DiscoMT 2015 shared task on pronoun translation (Hardmeier et al., 2015) – four phrase-based SMT systems AUTO-POSTEDIT (Guillou, 2015), UU-HARDMEIER (Hardmeier et al., 2015), IDIAP (Luong et al., 2015), UU-TIEDEMANN (Tiedemann, 2015), a rule-based system ITS2 (Loáiciga and Wehrli, 2015), and the shared task baseline (also phrase-based SMT). Three NMT systems are included for comparison: LIMSI (Bawden et al., 2017), NYU (Jean et al., 2014), and YANDEX (Voita et al., 2018).

Manual evaluation was conducted using the PROTEST graphical user interface and accompanying guidelines (Hardmeier and Guillou, 2016). The annotators were asked to make judgements (correct/incorrect) on the translations of the pronouns and antecedent heads whilst ignoring the correctness of other words (except in cases where it impacted the annotator's ability to make a judgement). The annotations were carried out by two bilingual English-French speakers, both of whom are native speakers of French. Our human judgements differ in important ways from the human evaluation conducted for the same set of systems at DiscoMT 2015 (Hardmeier et al., 2015), which was carried out by non-native speakers over an unbalanced data sample using a gap-filling methodology. In the gap-filling task annotators are asked to select, from a predefined list (including an uninformative catch-all group "other"), those pronouns that could fill the pronoun translation slot. Unlike in the PROTEST evaluation, the pronoun translations were obscured in the MT output. This avoided priming the annotators with the output of the candidate translation, but it occasionally caused valid translations to be rejected because they were missed by the annotator.

4 Accuracy versus Precision/Recall

There are three ways in which APT differs from AutoPRF: the scoring statistic, the alignment heuristic in APT, and the definition of pronoun equivalence.

APT is a measure of accuracy: It reflects the proportion of source pronouns for which an acceptable translation was produced in the target. AutoPRF, by contrast, is a precision/recall metric on the basis of clipped counts. Hardmeier and Federico (2010) motivate the use of precision and recall by pointing out that word alignments are not $1 : 1$, so each pronoun can be linked to multiple elements in the target language, both in the reference translation and in the MT output. Their metric is designed to account for all linked words in such cases.

To test the validity of this argument, we examined the subset of examples of 8 systems in our English-French dataset[1] giving rise to a clipped count greater than one[2] and found that these examples follow very specific patterns. All 143 cases included exactly one personal pronoun. In 99 cases, the additional matched word was the complementiser *que* ‘that’. In 31 and 4 cases, respectively, it was a form of the auxiliary verbs *avoir* ‘to have’ and *être* ‘to be’. One example matched both *que* and a form of *être*. Two had reflexive pronouns, and one an imperative verb form. With the possible exception of the two reflexive pronouns, none of this seems to be relevant to pronoun correctness. We conclude that it is more reasonable to restrict the counts to a single pronominal item per example. With this additional restriction, however, the recall score of AutoPRF becomes equivalent to a version of APT without equivalent pronouns and alignment correction. We therefore limit the remainder of our study to APT.

5 Effects of Word Alignment

APT includes an heuristic alignment correction procedure to mitigate errors in the word alignment between a source-language text and its translation (reference or MT output). We ran experiments to assess the correlation of APT with human judgements, with and without the alignment correction heuristics.

Score	APT-A		APT-B		PRO-TEST
Alig. corr.	+	−	+	−	
Reference	1.000	1.000	1.000	1.000	0.920
BASELINE	**0.544**	**0.536**	**0.574**	**0.566**	0.660
IDIAP	0.496	0.496	0.528	0.528	0.660
UU-TIED.	0.532	0.532	0.562	0.562	0.680
UU-HARD.	0.528	0.520	0.556	0.548	0.636
POSTEDIT	0.492	0.492	0.532	0.532	0.668
ITS2	0.436	0.428	0.462	0.454	0.472
LIMSI	0.364	0.364	0.388	0.388	0.576
NYU	0.424	0.420	0.456	0.452	0.616
YANDEX	**0.544**	**0.536**	0.570	0.562	**0.796**

Table 1: Comparison of APT scores with human judgements over the PROTEST test suite

Table 1 displays the APT results in both conditions and the proportion of pronouns in the PROTEST test suite marked as correctly translated. For better comparison with the PROTEST test suite results, we restricted APT to the pronouns in the test suite. We used two different weight settings:[3] **APT-A** uses weight 1 for *identical* matches and 0 for all other cases. **APT-B** uses weight 1 for *identical* matches, 0.5 for *equivalent* matches and 0 otherwise.

There is little difference in the APT scores when we consider the use of alignment heuristics. This is due to the small number of pronouns for which alignment improvements are applied for most systems (typically 0–12 per system). The exception is the ITS2 system output for which 18 alignment improvements are made. For the following systems we observe a very small increase in APT score for each of the two weight settings we consider, when alignment heuristics are applied: UU-HARDMEIER (+0.8), ITS2 (+0.8), BASELINE (+0.8), YANDEX (+0.8), and NYU (+0.4). However, these small improvements are not sufficient to affect the system rankings. It seems, therefore, that the alignment heuristic has only a small impact on the validity of the score.

To assess differences in correlation with human judgment for pairs of APT settings, we run Williams’s significance test (Williams, 1959; Graham and Baldwin, 2014). The test reveals that differences in correlation between the various configurations of APT and human judgements are not statistically significant ($p > 0.2$ in all cases).

---

[1] Excluding the YANDEX system, which was added later.

[2] A clipped count greater than one for a given pronoun translation indicates that the MT output and the reference translation aligned to this pronoun overlap in more than one token.

[3] Personal recommendation by Lesly Miculicich Werlen.

	c1	c2	Pearson	Spearman
With alignment heuristics	1	0	0.848	0.820
	1	0.5	0.853	0.815
Without alignment heuristics	1	0	0.850	0.820
	1	0.5	0.855	0.811

Table 2: Correlation of APT and human judgements

Category	**APT Cases**			**Human Assess.**		**Disagreement**		
	1	2	3	✓	✗	Dis.	Ex.	%
Anaphoric								
intra sbj *it*	130	13	73	156	60	47 /	216	21.8
intra nsbj *it*	59	1	31	77	14	19 /	91	20.9
inter sbj *it*	104	21	111	142	94	63 /	236	26.7
inter nsbj *it*	21	0	7	8	20	13 /	28	**46.4**
intra *they*	131	0	95	154	72	37 /	226	16.4
inter *they*	126	0	108	129	105	47 /	234	20.1
sg *they*	57	0	66	83	40	58 /	123	**47.2**
group *it/they*	47	0	41	64	24	31 /	88	35.2
Event *it*	145	42	94	185	96	60 /	281	21.4
Pleonastic *it*	171	54	52	243	34	46 /	277	16.6
Generic *you*	117	0	70	186	1	69 /	187	**36.9**
Deictic sg *you*	95	0	47	140	2	45 /	142	31.7
Deictic pl *you*	91	0	7	97	1	6 /	98	6.1
Total	1,294	131	802	1,664	563	541 /	2,227	24.3

Table 3: Number of pronouns marked as correct/incorrect in the PROTEST human judgements, as identical (1), equivalent (2), and incompatible (3) by APT, and the percentage of disagreements, per category (Disagree [Dis.] / Examples [Ex.])

6 Metric Accuracy per Category

Like Miculicich Werlen and Popescu-Belis (2017), we use Pearson's and Spearman's correlation coefficients to assess the correlation between APT and our human judgements (Table 2). Although APT does correlate with the human judgements over the PROTEST test suite, the correlation is weaker than that with the DiscoMT gap-filling evaluations reported in Miculicich Werlen and Popescu-Belis (2017). A Williams significance test reveals that the difference in correlation (for those systems common to both studies) is not statistically significant ($p > 0.3$). Table 1 also shows that the rankings induced from the PROTEST and APT scores are rather different. The differences are due to the different ways in which the two metrics define pronoun correctness, and the different sources against which correctness is measured (reference translation vs. human judgement).

We also study how the results of APT (with alignment correction) interact with the categories in PROTEST. We consider a pronoun to be measured as *correct* by APT if it is assigned case 1 (identical) or 2 (equivalent). Likewise, a pronoun is considered *incorrect* if it is assigned case 3 (incompatible). We compare the number of pronouns marked as correct/incorrect by APT and by the human judges, ignoring APT cases in which no judgement can be made: no translation of the pronoun in the MT output, reference or both, and pronouns for which the human judges were unable to make a judgement due to factors such as poor overall MT quality, incorrect word alignments, etc. The results of this comparison are displayed in Table 3.

At first glance, we can see that APT disagrees with the human judgements for almost a quarter (24.3%) of the assessed translations. The distribution of the disagreements over APT cases is very skewed and ranges from 8% for case 1 to 32% for case 2 and 49% for case 3. In other words, APT identifies correct pronoun translations with good precision, but relatively low recall. We can also see that APT rarely marks pronouns as equivalent (case 2).

Performance for anaphoric pronouns is mixed. In general, there are three main problems affecting anaphoric pronouns (Table 4). 1) APT, which does not incorporate knowledge of anaphoric pronoun antecedents, does not consider pronoun-antecedent head agreement so many *valid alternative translations involving personal pronouns* are marked as incompatible (i.e. incorrect, case 3), but as correct by the human judges. Consider the following example, in which the pronoun *they* is deemed correctly translated by the YANDEX system (according to the human judges) as it agrees in number and grammatical gender with the translation of the antecedent *extraits (clips)*. However, the pronoun translation *ils* is marked as incorrect by APT as it does not match the translation in the reference (*elles*).

SOURCE: so what these two **clips** show is not just the devastating consequence of the disease, but **they** also tell us something about the shocking pace of the disease…

YANDEX: donc ce que ces deux **extraits**[masc.,pl.] montrent n'est pas seulement la conséquence dévastatrice de la maladie, mais **ils**[masc. pl.] nous disent aussi quelque chose sur le rythme choquant de la maladie…

REFERENCE: ce que ces deux **vidéos**[fem.,pl.] montrent, ce ne sont pas seulement les conséquences dramatiques de cette maladie, **elles**[fem. pl.] nous montrent aussi la vitesse fulgurante de cette maladie…

2) Substitutions between pronouns are governed by much more complex rules than the simple pronoun equivalence mechanism in APT. For example, the dictionary of pronouns used in APT lists *il* and *ce* as equivalent. However, while *il* can often replace *ce* as a pleonastic pronoun in French, it has a much stronger tendency to be interpreted as anaphoric, rendering pleonastic use unacceptable if there is a salient masculine antecedent in the context. 3) APT does not consider the use of *impersonal pronouns* such as *c'* in place of the feminine personal pronoun *elle* or the plural forms *ils* and *elles*.

Category	**V**	**E**	**I**	**O**
Anaphoric				
intra-sent. subj. *it*	22	9	8	8
intra-sent. non-subj. *it*	16	–	1	2
inter-sent. subj. *it*	35	6	22	–
inter-sent. non-subj. *it*	–	–	–	13
intra-sent. *they*	25	–	3	9
inter-sent. *they*	22	–	3	22
singular *they*	40	–	–	18
group *it/they*	21	–	–	10
Event it	–	16	–	44
Pleonastic it	–	11	–	35

V: Valid alternative translation **I**: Impersonal translation
E: Incorrect equivalence **O**: Other

Table 4: Common cases of disagreement for anaphoric, pleonastic, and event reference pronouns

As with anaphoric pronouns, APT incorrectly marks some *pleonastic* and *event* translations as equivalent, in disagreement with the human judges. Other common errors arise from 1) the use of alternative translations marked as incompatible (i.e. incorrect) by APT but correct by the human judges, for example *il* (personal) in the MT output when the reference contained the impersonal pronoun *cela* or *ça* (30 cases for pleonastic, 7 for event), or 2) the presence of *il* in both the MT output and reference marked by APT as identical but by the human judges as incorrect (3 cases for pleonastic, 15 event).

Some of these issues could be addressed by incorporating knowledge of pronoun function in the source language, of pronoun antecedents, and of the wider context of the translation surrounding the pronoun. However, whilst we might be able to derive language-specific rules for some scenarios, it would be difficult to come up with more general or language-independent rules. For example, *il* and *ce* can be anaphoric or pleonastic pronouns, but *il* has a more *referential* character. Therefore in certain constructions that are strongly pleonastic (e.g. clefts) only *ce* is acceptable. This rule would be specific to French, and would not cover other scenarios for the translation of pleonastic *it*. Other issues include the use of pronouns in impersonal constructions such as *il faut* [one must/it takes] in which evaluation of the pronoun requires consideration of the whole expression, or transformations between active and passive voice, where the perspective of the pronouns changes.

7 Conclusions

Our analyses reveal that despite some correlation between APT and the human judgements, fully automatic wide-coverage evaluation of pronoun translation misses essential parts of the problem. Comparison with human judgements shows that APT identifies good translations with relatively high precision, but fails to reward important patterns that pronoun-specific systems must strive to generate. Instead of relying on fully automatic evaluation, our recommendation is to emphasise high precision in the automatic metrics and implement semi-automatic evaluation procedures that refer negative cases to a human evaluator, using available tools and methods (Hardmeier and Guillou, 2016). Fully automatic evaluation of a very restricted scope may still be feasible using test suites designed for specific problems (Bawden et al., 2017).

Acknowledgements

We would like to thank our annotators, Marie Dubremetz and Miryam de Lhoneux, for their many hours of painstaking work, Lesly Miculicich Werlen for providing APT results for the DiscoMT 2015 systems, Elena Voita, Sébastien Jean, Stanislas Lauly and Rachel Bawden for providing the NMT system outputs, and the three anonymous reviewers. The annotation work was funded by the European Association for Machine Translation. The work carried out at The University of Edinburgh was funded by the ERC H2020 Advanced Fellowship GA 742137 SEMANTAX and a grant from The University of Edinburgh and Huawei Technologies. The work carried out at Uppsala University was funded by the Swedish Research Council under grant 2017-930.

References

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2017. Evaluating discourse phenomena in neural machine translation. *CoRR*, abs/1711.00513.

Simon C. Dik. 1978. *Functional Grammar*. Amsterdam, North-Holland.

Yvette Graham and Timothy Baldwin. 2014. Testing for significance of increased correlation with human judgment. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 172–176, Doha, Qatar. Association for Computational Linguistics.

Liane Guillou. 2015. Automatic post-editing for the DiscoMT pronoun translation task. In *Proceedings of the Second Workshop on Discourse in Machine Translation*, pages 65–71, Lisbon, Portugal. Association for Computational Linguistics.

Liane Guillou. 2016. *Incorporating Pronoun Function into Statistical Machine Translation*. Ph.D. thesis, Edinburgh University, Department of Informatics.

Liane Guillou and Christian Hardmeier. 2016. PROTEST: A test suite for evaluating pronouns in machine translation. In *Proceedings of the Eleventh Language Resources and Evaluation Conference*, LREC 2016, pages 636–643, Portorož, Slovenia.

Liane Guillou, Christian Hardmeier, Aaron Smith, Jörg Tiedemann, and Bonnie Webber. 2014. ParCor 1.0: A parallel pronoun-coreference corpus to support statistical MT. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, LREC 2014, pages 3191–3198, Reykjavik, Iceland. European Language Resources Association (ELRA).

Michael A. K. Halliday. 2004. *An introduction to functional grammar*, 3rd edition. Hodder Arnold London.

Christian Hardmeier. 2014. *Discourse in Statistical Machine Translation*. Ph.D. thesis, Uppsala University, Department of Linguistics and Philology.

Christian Hardmeier and Marcello Federico. 2010. Modelling pronominal anaphora in statistical machine translation. In *Proceedings of the 7th International Workshop on Spoken Language Translation*, IWSLT 2010, pages 283–289, Paris, France.

Christian Hardmeier and Liane Guillou. 2016. A graphical pronoun analysis tool for the protest pronoun evaluation test suite. *Baltic Journal of Modern Computing*, (2):318–330.

Christian Hardmeier, Preslav Nakov, Sara Stymne, Jörg Tiedemann, Yannick Versley, and Mauro Cettolo. 2015. Pronoun-focused MT and cross-lingual pronoun prediction: Findings of the 2015 DiscoMT shared task on pronoun translation. In *Proceedings of the Second Workshop on Discourse in Machine Translation*, DiscoMT 2015, pages 1–16, Lisbon, Portugal.

Christian Hardmeier, Jörg Tiedemann, Preslav Nakov, Sara Stymne, and Yannick Versely. 2016. DiscoMT 2015 Shared Task on Pronoun Translation. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *ArXiv e-prints*, 1412.2007.

Sharid Loáiciga and Eric Wehrli. 2015. Rule-based pronominal anaphora treatment for machine translation. In *Proceedings of the Second Workshop on Discourse in Machine Translation*, pages 86–93, Lisbon, Portugal. Association for Computational Linguistics.

Sharid Loáiciga. 2017. *Pronominal anaphora and verbal tenses in machine translation*. Ph.D. thesis, Université de Genève.

Ngoc Quang Luong, Lesly Miculicich Werlen, and Andrei Popescu-Belis. 2015. Pronoun translation and prediction with or without coreference links. In *Proceedings of the Second Workshop on Discourse in Machine Translation*, pages 94–100, Lisbon, Portugal. Association for Computational Linguistics.

Lesly Miculicich Werlen and Andrei Popescu-Belis. 2017. Validation of an automatic metric for the accuracy of pronoun translation (APT). In *Proceedings of the Third Workshop on Discourse in Machine Translation (DiscoMT)*. Association for Computational Linguistics (ACL).

Ruslan Mitkov and Catalina Barbu. 2003. Using bilingual corpora to improve pronoun resolution. *Languages in Contrast*, 4(2):201–211.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia (Pennsylvania, USA). ACL.

Jörg Tiedemann. 2015. Baseline models for pronoun prediction and pronoun-aware translation. In *Proceedings of the Second Workshop on Discourse in Machine Translation*, pages 108–114, Lisbon, Portugal. Association for Computational Linguistics.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Evan J. Williams. 1959. *Regression Analysis*, volume 14. Wiley, New York.

FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation

Xu Han[1,*] **Hao Zhu**[1,*] **Pengfei Yu**[2,*] **Ziyun Wang**[1,†,*]
Yuan Yao[1] **Zhiyuan Liu**[1,‡] **Maosong Sun**[1]
`http://zhuhao.me/fewrel`
Institute for Artificial Intelligence
State Key Laboratory of Intelligent Technology and Systems
[1]Department of CST, [2]Department of EE, Tsinghua University, Beijing, China
`{hanxu17,zhuhao15,yupf15,yy18}@mails.tsinghua.edu.cn,`
`ziyunw@nyu.edu, {lzy,sms}@tsinghua.edu.cn`

Abstract

We present a Few-Shot Relation Classification Dataset (FewRel), consisting of $70,000$ sentences on 100 relations derived from Wikipedia and annotated by crowdworkers. The relation of each sentence is first recognized by distant supervision methods, and then filtered by crowdworkers. We adapt the most recent state-of-the-art few-shot learning methods for relation classification and conduct thorough evaluation of these methods. Empirical results show that even the most competitive few-shot learning models struggle on this task, especially as compared with humans. We also show that a range of different reasoning skills are needed to solve our task. These results indicate that few-shot relation classification remains an open problem and still requires further research. Our detailed analysis points multiple directions for future research. All details and resources about the dataset and baselines are released on `http://zhuhao.me/fewrel`.

Supporting Set	
(A) capital_of	(1) *London* is the capital of *the U.K.* (2) *Washington* is the capital of *the U.S.A.*
(B) member_of	(1) *Newton* served as the president of *the Royal Society*. (2) *Leibniz* was a member of *the Prussian Academy of Sciences*.
(C) birth_name	(1) *Samuel Langhorne Clemens*, better known by his pen name *Mark Twain*, was an American writer. (2) *Alexei Maximovich Peshkov*, primarily known as *Maxim Gorky*, was a Russian and Soviet writer.
Test Instance	
(A) or (B) or (C)	*Euler* was elected a foreign member of *the Royal Swedish Academy of Sciences*.

Table 1: An example for a 3 way 2 shot scenario. Different colors indicate different entities, blue for head entity, and red for tail entity.

1 Introduction

Relation classification (RC) is an important task in NLP, aiming to determine the correct relation between two entities in a given sentence. Many works have been proposed for this task, including kernel methods (Zelenko et al., 2002; Mooney and Bunescu, 2006), embedding methods (Gormley et al., 2015), and neural methods (Zeng et al., 2014). The performance of these conventional models heavily depends on time-consuming and labor-intensive annotated data, which make themselves hard to generalize well. Adopting distant supervision is a primary approach to alleviate this problem for RC (Mintz et al.; Riedel et al.; Hoffmann et al., 2011; Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016), which heuristically aligns knowledge bases (KBs) and text to automatically annotate adequate amounts of training instances. We evaluate the model proposed by Lin et al. (2016), which is followed by the recent state-of-the-art methods (Zeng et al., 2017; Ji et al., 2017; Huang and Wang, 2017; Wu et al., 2017; Liu et al., 2017; Feng et al., 2018; Zeng et al., 2018), on the benchmark dataset NYT-10 (Riedel et al.). Though it achieves promising results on common relations, the performance of a relation drops dramatically when its number of training instances decrease. About 58% of the relations in NYT-10 are long-tail with fewer than 100 instances. Furthermore, distant supervision suffers from the wrong labeling problem, which makes it harder to classify long-tail relations. Hence, it is necessary to study training RC models with insufficient training instances.

We formulate RC as a few-shot learning task in

---

[*] The first four authors contribute equally. The order is determined by dice rolling.
[†] Z. Wang is now at New York University.
[‡] Correspondence author.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4803–4809
Brussels, Belgium, October 31 - November 4, 2018.

this paper, which requires models capable of handling classification task with a handful of training instances, as shown in Table 1. Many efforts have devoted to few-shot learning. The early works (Caruana, 1995; Bengio, 2012; Donahue et al., 2014) apply transfer learning methods to fine-tune pre-trained models from the common classes containing adequate instances to the uncommon classes with only few instances. Then metric learning methods (Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017) have been proposed to learn the distance distributions among classes. Similar classes are adjacent in the distance space. The metric methods also take advantage of non-parametric estimation to make models efficient and general. Recently, the idea of meta-learning is proposed, which encourages the models to learn fast-learning abilities from previous experience and rapidly generalize to new concepts. Many meta-learning models (Ravi and Larochelle, 2017; Santoro et al., 2016; Finn et al., 2017; Munkhdalai and Yu, 2017) achieve the state-of-the-art results on several few-shot benchmarks.

Though meta-learning methods develop fast, most of these works evaluate on two popular datasets, Omniglot (Lake et al., 2015) and mini-ImageNet (Vinyals et al., 2016). Both the datasets concentrate on image classification. Many works in NLP mainly focus on the zero-shot/semi-supervised scenario (Xie et al., 2016; Ma et al., 2016; Carlson et al., 2009), which incorporate extra information to classify objects never appearing in the training sets. However, the few-shot scenario needs models to classify objects with few instances without any extra information. Recently, Yu et al. (2018) propose a multi-metric method for few-shot text classification. However, there lack systematic researches about adopting few-shot learning for NLP tasks. We propose FewRel: a new large-scale supervised **Few**-shot **Rel**ation Classification dataset. To address the wrong labeling problem in most distantly supervised RC datasets, we apply crowd-sourcing to manually remove the noise.[i]

Besides constructing the dataset, we systematically implement the most recent state-of-the-art few-shot learning methods and adapt them for RC. We conduct a detailed evaluation for all these models on our dataset. Though the state-of-the-art few-shot learning methods have much lower results than humans on our challenging dataset, they significantly outperform the vanilla RC models, indicating that incorporating few-shot learning is promising and needs further research. In summary, our contribution is three-fold:

(1) We formulate RC as a few-shot learning task, and propose a new large supervised few-shot RC dataset.

(2) We systematically adapt the most recent state-of-the-art few-shot learning methods for RC, which may further benefit other NLP tasks.

(3) We conduct a comprehensive evaluation of few-shot learning methods on our dataset, which indicates some promising research directions for RC.

2 FewRel Dataset

In this section, we describe the process of creating FewRel in detail. The whole procedure can be divided into two steps: (1) We create a large candidate set of sentences aligned to relations via distant supervision. (2) We ask human annotators to filter out the wrong labeled sentences for each relation to finally achieve a clean RC dataset.

2.1 Distant Supervision

For the first step, We use Wikipedia as the corpus[ii] and Wikidata as the KB. Wikidata is a large-scale KB where many entities are already linked to Wikipedia articles. The articles in Wikipedia also contain anchors linking to each other. Thus it is convenient to align sentences in Wikipedia articles to KB facts in Wikidata. We also employ entity linking technique to extract more unanchored entities in articles. We first adopt named entity recognition via spaCy[iii] to find possible entity mentions, then match each mention with the name of an entity in KBs, and link the mention to the entity if successfully matched.

For each sentence s in Wikipedia articles containing head and tail entities e_1 and e_2, if there exists a Wikidata statement (e_1, e_2, r) meaning e_1 and e_2 have the relation r, we denote the (s, e_1, e_2, r) tuple as an instance and add it to the candidate set. Empirically, many instances of a given relation contain the same entity pair. For

---

[i] Many previous works, such as (Roth et al., 2013; Luo et al., 2017; Xin et al., 2018) have worked on automatically removing noise from distantly supervision. Instead, we use crowd-sourcing methods to achieve a high accuracy.

[ii] We use whole Wikipedia articles as corpus, not just the first sentence.

[iii] `https://spacy.io/`

such relation, classifiers may prefer memorizing the entity pairs in the training instances rather than grasping the sentence semantics. Therefore, in the candidate set of each relation, we only keep 1 instance for each unique entity pair. Finally, we remove relations with fewer than 1000 instances, and randomly keep 1000 instances for the rest of the relations. As a result, we get a candidate set of 122 relations and 122, 000 instances.

2.2 Human Annotation

Next, we invite some well-educated annotators to filter the raw data on a platform similar to Amazon MTurk developed by ourselves. The platform presents each annotator with one instance each time, by showing the sentence, two entities in the sentence, and the corresponding relation labeled by distant supervision. The platform also provides the name of the entities and relation in Wikidata accompanied with the detailed description of that relation. Then the annotator is asked to judge whether the relation could be deduced only from the sentence semantics. We also ask the annotator to mark an instance as negative if the sentence is not complete, or the mention is falsely linked with the entity.

Relations are randomly assigned to annotators from the candidate set, and each annotator will consecutively annotate 20 instances of the same relation before switching to next relation. To ensure the labeling quality, each instance is labeled by at least two annotators. If the two annotators have disagreements on this instance, it will be assigned to a third annotator. As a result, each instance has at least two same annotations, which will be the final decision. After the annotation, we remove relations with fewer than 700 positive instances. For the remaining 105 relations, we calculate the inter-annotator agreement for each relation using the free-marginal multirater kappa (Randolph, 2005), and keep the top 100 relations.

2.3 Dataset Statistics

The final FewRel dataset consists of 100 relations, each has 700 instances. A full list of relations, including their names and descriptions, is provided in Appendix A.2. The average number of tokens in each sentence is 24.99, and there are 124, 577 unique tokens in total. Following recent meta-learning tasks (Vinyals et al., 2016), which use separate sets of classes for training and testing, we use 64, 16, and 20 relations for training, validation, and testing respectively. Table 2 provides a comparison of our FewRel dataset to two other popular few-shot classification datasets, Omniglot and mini-ImageNet. Table 3 provides a comparison of FewRel to the previous RC datasets, including SemEval-2010 Task 8 dataset (Hendrickx et al., 2009), ACE 2003-2004 dataset (Strassel et al., 2008), TACRED dataset (Zhang et al., 2017), and NYT-10 dataset (Riedel et al., 2010). While some RC datasets contain instances with no relations (negative), we ignore such instances for comparison.

Dataset	#cls.	#inst./cls	#insts.
Omniglot	1, 623	20	32, 460
mini-ImageNet	100	600	60, 000
FewRel	100	700	70, 000

Table 2: Comparison of FewRel with Omniglot and mini-ImageNet.

Dataset	#cls.	#insts.
SemEval-2010 Task 8	9	6, 674
ACE 2003-2004	24	16, 771
TACRED	42	21, 784
NYT-10	57	143, 391
FewRel	100	70, 000

Table 3: Comparison of FewRel with existing RC datasets. Note that negative (no relation) instances in some datasets are ignored.

3 Experiments

We conduct comprehensive evaluations of vanilla RC models with simple strategies such as finetune or kNN on our new dataset. We also evaluate the recent state-of-the-art few-shot learning methods.

3.1 Task Formulation

In few-shot relation classification, we intend to obtain a function $F : (\mathcal{R}, \mathcal{S}, x) \mapsto y$. Here $\mathcal{R} = \{r_1, \ldots, r_m\}$ defines the relations that the instances are classified into. $\mathcal{S}$ is a support set

$$\begin{aligned} \mathcal{S} = \{&(x_1^1, r_1), (x_1^2, r_1), \ldots, (x_1^{n_1}, r_1), \\ &\ldots, \\ &(x_m^1, r_m), (x_m^2, r_m), \ldots, (x_m^{n_m}, r_m)\} \end{aligned} \tag{1}$$

including n_i instances for each relation $r_i \in \mathcal{R}$. For relation classification, a data instance x_i^j is a sentence accompanied with a pair of entities. The query data x is an unlabeled instance to classify, and $y \in \mathcal{R}$ is the prediction of x given by F.

Model	5 Way 1 Shot	5 Way 5 Shot	10 Way 1 Shot	10 Way 5 Shot
Finetune (CNN)	44.21 ± 0.44	68.66 ± 0.41	27.30 ± 0.28	55.04 ± 0.31
Finetune (PCNN)	45.64 ± 0.62	57.86 ± 0.61	29.65 ± 0.40	37.43 ± 0.42
kNN (CNN)	54.67 ± 0.44	68.77 ± 0.41	41.24 ± 0.31	55.87 ± 0.31
kNN (PCNN)	60.28 ± 0.43	72.41 ± 0.39	46.15 ± 0.31	59.11 ± 0.30
Meta Network (CNN)	64.46 ± 0.54	80.57 ± 0.48	53.96 ± 0.56	69.23 ± 0.52
GNN (CNN)	66.23 ± 0.75	81.28 ± 0.62	46.27 ± 0.80	64.02 ± 0.77
SNAIL (CNN)	67.29 ± 0.26	79.40 ± 0.22	53.28 ± 0.27	68.33 ± 0.25
Prototypical Network (CNN)	69.20 ± 0.20	84.79 ± 0.16	56.44 ± 0.22	75.55 ± 0.19
Human performance	92.22 ± 5.53	-	85.88 ± 7.40	-

Table 4: Accuracies (%) of all models on FewRel under four different settings.

In recent research on few-shot learning, N way K shot setting is widely adopted. We follow this setting for the few-shot relation classification problem. To be exact, for N way K shot learning

$$N = m = |\mathcal{R}|, K = n_1 = \ldots = n_m \qquad (2)$$

3.2 Experiment Settings

We consider four types of few-shot tasks in our experiments: 5 way 1 shot, 5 way 5 shot, 10 way 1 shot, 10 way 5 shot. Under this setting, we evaluate different few-shot training strategies and state-of-the-art few-shot learning methods built upon two widely used instance encoders, CNN (Zeng et al., 2014) and PCNN (Zeng et al., 2015).

For both CNN and PCNN, the sentence is first represented to the input vectors by transforming each word into concatenation of word embeddings and position embeddings. In CNN, the input vectors pass a convolution layer, a max-pooling layer, and a non-linear activation layer to get the final output sentence embedding. PCNN is a variant of CNN, which replaces the max-pooling operation with a piecewise max-pooling operation.

To evaluate this two vanilla models in few-shot RC task, we first consider two training strategies, namely Finetune and kNN. For the Finetune baseline, it learns to classify all relations on the training set with CNN/PCNN, and tune parameters on the support set. We only tune the parameters of output layer, and keep other parameters unchanged. For the kNN baseline, it also jointly classifies all relations during training, while at the test time, it uses the neural networks to embed all the instances and then adopts k-nearest-neighbor (kNN) to classify the test instances.

By adapting them to relation classification, we also evaluate four recently proposed few-shot learning methods, including Meta Network (Munkhdalai and Yu, 2017), GNN (Satorras and Estrach, 2018), SNAIL (Mishra et al., 2018), and Prototypical Network (Snell et al., 2017). We describe briefly about these baselines in Sec. 3.3. If you are familiar with these methods, you can safely skip that subsection. The hyperparameters of each model are selected via grid search against the validation set.

Human performance is also evaluated under 5 way 1 shot setting and 10 way 1 shot setting. A human labeler is given $5/10$ instances from different relations and one extra test instance. Human labelers are asked to decide which relation the test instance belongs to. Note that these labelers are not provided the name of the relations and any extra information. Since 5 way 5 shot and 10 way 5 shot settings are easier, we only evaluate performance of 5 way 1 shot and 10 way 1 shot.

3.3 Baselines of Few-shot Learning Models

Meta Network Meta Network (Munkhdalai and Yu, 2017) is a meta learning algorithm utilizing a high level *meta learner* on top of the traditional classification model, or *base learner*, to supervise the training process. The weights of base learner are divided into two groups, fast weights and slow weights. Fast weights are generated by the meta learner, whereas slow weights are simply updated by minimizing classification loss. The fast weights are expected to help the model generalize to new tasks with very few training instances.

GNN GNN (Satorras and Estrach, 2018) tackles the few-shot learning problem by considering each supporting instance or query instance as a node in the graph. For those instances in the support sets, label information is also embedded into the corresponding node representations. Graph neural networks are then employed to propagate the information between nodes. A query instance is expected to receive information from support sets in order to make the classification. In our adaption, while the instances are encoded by CNNs, labels are represented by one-hot encoding.

SNAIL SNAIL (Mishra et al., 2018) is a meta learning model that utilizes temporal convolutional neural networks and attention modules for fast learning from past experience. SNAIL arranges all the supporting instance-label pairs into a sequence and appends the query instance behind them. Such an order agrees with the temporal order of learning process where we learn information by reading supporting instances before making predictions for unlabeled instances. Temporal convolution (a 1-D convolution) is then performed along the sequence to aggregate information across different time steps and a causally masked attention model is used over the sequence to aggregate useful information from former instances to latter ones.

Prototypical Networks Prototypical Network (Snell et al., 2017) is a few-shot classification model based on the assumption that for each class there exists a prototype. The model tries to find the prototypes for classes from supporting instances, and compares the distance between the query instance and each prototype under certain distance metric. Prototypical network learns a embedding function u to embed each class's instances, and computes each prototype by averaging over all the output embeddings of instances in the support set S that are labeled with the corresponding class.

4 Result Analysis and Future Work

We report evaluation results in Table 4. From our preliminary experiments, PCNN with few-shot learning methods perform 3-10 percentages worse than CNN, therefore only CNN results are shown in our experimental results. From the results, we observe that integrating few-shot learning methods into CNN significantly outperforms CNN/PCNN with finetune or kNN, which means adapting few-shot learning methods for RC is promising. However, there are still huge gaps between their performance and humans', which means our dataset is a challenging testbed for both relation classification and few-shot learning.

Sentence	Reasoning
Chris Bohjalian graduated from *Amherst College* Summa Cum Laude, where he was a member of the Phi Beta Kappa Society.	Simple Pattern
James Alty obtained a 1st class honours (Physics) at *Liverpool University*.	Commonsense Reasoning
He was a professor at *Reed College*, where he taught *Steve Jobs*, and replaced Lloyd J. Reynolds as the head of the calligraphy program.	Logical Reasoning
He and *Cesare Borgia* were thought to be close friends since childhood, going on to accompany one another during their studies at the *University of Pisa*.	Co-reference Reasoning

Table 5: Examples from relation "educated_at". Different colors indicate different entities, blue for head entity, and red for tail entity.

In this paper, we propose a new large and high quality dataset, FewRel, for few-shot relation classification task. This dataset provides a new point of view for RC, and also a new benchmark for few-shot learning. Through the evaluation of different few-shot learning methods, we find even the best model performs much worse than humans, which suggests there is still large space for few-shot learning methods to improve.

The most challenging characteristic of our dataset is the diversity in expressing the same relation. We provide some examples from FewRel in Table 5, showing different reasoning modes needed for classifying some instances. Future researches may consider incorporating commonsense knowledge or improved causal modules.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (NSFC No. 61572273, 61532010). This work is also funded by the Natural Science Foundation of China (NSFC) and the German Research Foundation (DFG) in Project Crossmodal Learning, NSFC 61621136008 / DFC TRR-169. Hao Zhu is supported by Tsinghua University Initiative Scientific Research Program. We thank all annotators for their hard work. We also thank all members from Tsinghua NLP Lab for their strong support for annotator recruitment.

References

Yoshua Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML*.

Andrew Carlson, Justin Betteridge, Estevam R Hruschka Jr, and Tom M Mitchell. 2009. Coupling semi-supervised learning of categories and relations. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing*, pages 1–9. Association for Computational Linguistics.

Rich Caruana. 1995. Learning many related tasks at the same time with backpropagation. In *Proceedings of NIPS*.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of ICML*.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of ICML*.

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of EMNLP*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SemEval@ACL*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*.

Yi Yao Huang and William Yang Wang. 2017. Deep residual learning for weakly-supervised relation extraction. In *Proceedings of EMNLP*.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of AAAI*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *Proceedings of ICML*.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of EMNLP*.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 430–439.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *COLING*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *Proceedings of ICLR*.

Raymond J Mooney and Razvan C Bunescu. 2006. Subsequence kernels for relation extraction. In *Proceedings of NIPS*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *Proceedings of ICML*.

Justus J Randolph. 2005. Free-marginal multirater kappa (multirater κfree): an alternative to fleiss' fixed-marginal multirater kappa. In *Proceedings of JLIS*.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *Proceedings of ICLR*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*.

Sebastian Riedel, Limin Yao, and Andrew D McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*.

Benjamin Roth, Tassilo Barth, Michael Wiegand, and Dietrich Klakow. 2013. A survey of noise reduction methods for distant supervision. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 73–78. ACM.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of ICML*.

Victor Garcia Satorras and Joan Bruna Estrach. 2018. Few-shot learning with graph neural networks. In *Proceedings of ICLR*.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of NIPS*.

Stephanie Strassel, Mark A. Przybocki, Kay Peterson, Zhiyi Song, and Kazuaki Maeda. 2008. Linguistic resources and evaluation techniques for evaluation of cross-document automatic content extraction. In *Proceedings of LREC*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Proceedings of NIPS*.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of EMNLP*.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*.

Ji Xin, Hao Zhu, Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Put it back: Entity typing with language model enhancement. In *Proceedings of EMNLP*.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1206–1215.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. *JMLR*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Incorporating relation paths in neural relation extraction. In *Proceedings of EMNLP*.

Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Large scaled relation extraction with reinforcement learning. In *Proceedings of AAAI*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of EMNLP*.

A strong baseline for question relevancy ranking

Ana V. González-Garduño and Isabelle Augenstein and Anders Søgaard
Dpt. of Computer Science
University of Copenhagen
`{ana|augenstein|soegaard}@di.ku.dk`

Abstract

The best systems at the SemEval-16 and SemEval-17 community question answering shared tasks – a task that amounts to *question relevancy ranking* – involve complex pipelines and manual feature engineering. Despite this, many of these still fail at beating the IR baseline, i.e., the rankings provided by Google's search engine. We present a strong baseline for question relevancy ranking by training a simple multi-task feed forward network on a bag of 14 distance measures for the input question pair. This baseline model, which is fast to train and uses only language-independent features, outperforms the best shared task systems on the task of retrieving relevant previously asked questions.

1 Introduction

Community question-answer fora are great resources, collecting answers to frequently and less-frequently asked questions on specific topics, but these are often not moderated and contain many irrelevant answers. Community Question Answering (CQA), cast as a question relevancy ranking problem, was the topic of two shared tasks at SemEval 2016-17. This is a non-trivial retrieval task, typically evaluated using mean average precision (MAP). We present a strong baseline for this task, on par with or surpassing state-of-the-art systems.

The English subtasks of the SemEval CQA (Nakov et al., 2015, 2017) consist of Question-Question Similarity, Question-Comment Similarity, and Question-External Comment Similarity. In this study, we focus on the core subtask of Question-Question similarity, defined as follows: Given a question, rank other relevant questions by their relevancy to that question. This proved to be a difficult task in both SemEval-16 and SemEval-17 as it is the one with the least amount of data available. The baseline was the ranking retrieved by performing a Google search, which proved to be a strong baseline beating a large portion of the systems submitted.

Contribution Our baseline is a simple multi-task feed-forward neural network taking distance measures between pairs of questions as input. We use a question-answer dataset as auxiliary task; but we also experiment with datasets for pairwise classification tasks such as natural language inference and fake news detection. This simple, easy-to-train model is on par or better than state-of-the-art systems for question relevancy ranking. We also show that this simple model outperforms a more complex model based on recurrent neural networks.

2 Our Model

We present a simple baseline model for question relevancy ranking.[1] It is a deep feed-forward network with a hidden layer that is shared with an auxiliary task model. The input to the network is extremely simple and consists of five distance measures of the input question-question pair. §2.1 discusses these distance measures, and how they relate. §2.2 introduces the multi-task learning architecture that we propose.

2.1 Features

We use four similarity metrics and three sentence representations (averaged word embeddings, binary unigram vectors, and trigram vectors). The **cosine distance** between the sentence representations of query x and query y is

$$\frac{\sum_i \mathbf{x}_i \mathbf{y}_i}{\sqrt{\sum_i \mathbf{x}^2} + \sqrt{\sum_i \mathbf{y}^2}}$$

---

[1] Code available at http://anavaleriagonzalez/FAQ_rank.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4810–4815
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

The **Manhattan distance** is

$$\sum_i |\mathbf{x}_i - \mathbf{y}_i|$$

The **Bhattacharya distance** is

$$-\ln(\sum_i \sqrt{\mathbf{x}_i \mathbf{y}_i})$$

and is a measure of divergence, and the **Euclidean distance** is

$$\sqrt{\sum_i (\mathbf{x}_i - \mathbf{y}_i)^2}$$

Note that the squared Euclidean distance is proportional to cosine distance and Manhattan distance. The Bhattacharya and Jaccard metrics, on the other hand, are sensitive to the number of types in the input (the ℓ_1 norm of the vector encodings). So, for example, *only* the cosine, Euclidean, and Manhattan distances will be the same for

$$\mathbf{x} = \langle 1,1,0,0,1,0,1,1,0,1 \rangle, \mathbf{y} = \langle 0,0,1,0,1,0,0,0,1,1 \rangle$$

and

$$\mathbf{x} = \langle 0,0,0,0,0,1,0,0,1,1 \rangle, \mathbf{y} = \langle 1,1,1,1,0,0,0,0,0,1 \rangle$$

The **Jaccard index** is the only metric that can only be applied to two of our representations, unigrams and n-grams: It is defined over m-dimensional *binary* (indicator) vectors and therefore not applicable to averaged embeddings. It is defined as

$$\frac{\mathbf{x} \cdot \mathbf{y}}{m}$$

We represent each query pair by these 14 numerical features.

2.2 MTL Architecture

Our architecture is a simple feed-forward, multi-task learning (MTL) architecture. Our architecture is presented in Figure 1 and is a Multi-Layer Perceptron (MLP) that takes a pair of sequences as input. The sequences can be sampled from the main task or the auxiliary task. The MLP has one shared hidden layer, a task-specific hidden layer and, finally, a task-specific classification layer for each output. The hyper-parameters, after doing grid search, optimizing performance on the validation data, are given in Figure 2.

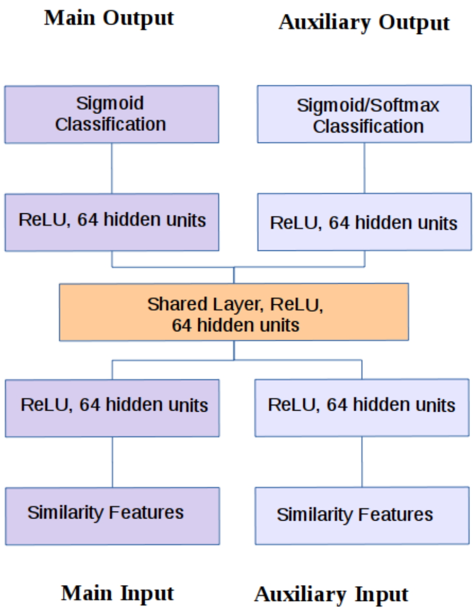

Figure 1: The architecture of the multi task learning MLP

2.3 LSTM baseline

We compare our MLP ranker to a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) model. It takes two sequences inputs: sequence 1 and sequence 2, and a stack of three bidirectional LSTM layers, which encode sequence 1 and sequence 2, respectively. The outputs are then concatenated, to enable representing the differences between the two sequences. Instead of relying only on this presentation (Bowman et al., 2015; Augenstein et al., 2016), we also concatenate our distance features and feed everything into our MLP ranker described above.

3 Datasets

For our experiments, we use data from SemEval shared tasks, but we also take advantage of potential synergies with other existing datasets for classification of sentence pairs. Below we present the datasets used for our main and auxiliary tasks. We provide some summary statistics for each dataset in Table 3.

SemEval 2016 and 2017 As our main dataset we use the queries from SemEval's subtask B which consists of an original query and 10 possibly related queries. As an auxiliary task, we use the data from subtask A, which is a question-related comment ranking task.

Hyperparameter	Best Value	Tested Values
Num. of Epochs	100	10, 20, 40, 60, 80, 100
Batch Size	100	10, 50, 100
Learning rate	0.001	0.001, 0.01, 0.1, 0.2, 0.3
Momentum	0.9	0.0, 0.2, 0.4, 0.6, 0.8, 0.9
Dropout	0.02	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

Figure 2: Hyper-parameters

	EN-SemEval-16				EN-SemEval-17			
Model	MAP	MRR	AvgRec	Acc	MAP	MRR	AvgRec	Acc
STL-MLP (our baseline)	76.21	84.17	88.85	77.86	47.12	51.44	81.95	63.30
STL-BiLSTM-SIM	73.42	82.02	87.93	76.35	46.85	51.20	81.00	68.64
IR baseline	74.75	83.79	88.30	-	41.85	46.42	77.59	-
Random baseline	46.98	50.96	67.92	-	29.81	33.02	62.65	-
SemEval-Best	76.70	83.02	90.31	76.57	47.22	50.07	82.60	52.39
MTL - English Only - QA Aux								
MTL-MLP	**77.61****	**84.29****	**90.33****	78.99**	47.66**	**52.41****	82.37**	71.70**
MTL-BiLSTM-SIM	71.25	79.54	86.52	75.26	43.58	49.16	79.36	50.40
MTL - FNC Aux								
MTL-MLP	76.83	84.05	90.29	**79.30**	47.33**	49.80**	**83.63****	73.06**
MTL-BiLSTM-SIM	71.70	80.83	87.35	73.29	48.15*	51.90*	81.59*	64.00*
MTL - NLI Aux								
MTL-MLP	77.06**	83.98**	89.97**	78.57**	48.00**	51.56**	82.91**	70.11**
MTL-BiLSTM-SIM	75.22**	83.02**	89.14**	77.71	**48.56***	52.05*	82.22*	71.36*
MTL - ALL Aux								
MTL-MLP	75.90**	84.17**	89.59**	78.51**	47.45**	51.12**	82.90**	**73.10****

Table 1: The results show that learning pairwise classification tasks simultaneously with the main task leads to improvements over the baselines and the best SemEval systems. We show results for three auxiliary tasks, Question-Comment relevancy prediction, Fake News detection and Natural Language Inference. The asterisks for the MTL results represent the significance of the improvements over the STL systems with ** representing a p-value of < 0.01 and * representing a p-value between 0.01 and 0.05

Natural Language Inference Natural Language Inference (NLI), consists in predicting ENTAILMENT, CONTRADICTION or NEUTRAL, given a hypothesis and a premise. We use the MNLI dataset as opposed to the SNLI data (Bowman et al., 2015; Nangia et al., 2017), since it contains different genres. Our model is not built to be a strong NLI system; we use the similarity between premise and hypothesis as a weak signal to improve the generalization on our main task.

Fake News Challenge The Fake News Challenge[2] (FNC) was introduced to combat misleading and false information online. This task has been used before in a multi-task setting as a way to utilize general information about pairwise relations (Augenstein et al., 2018). Formally, the FNC task consists in, given a headline and the body of text which can be from the same news article or not, classify the stance of the body of text relative to what is claimed in the headline. There are four labels:

- AGREES: The body of the article is in agreement with the headline
- DISAGREES: The body of the article is in disagreement with the headline
- DISCUSSES: The body of the article does not take a position
- UNRELATED: the body of the article discusses a different topic

We include fake news detection as a weak auxiliary signal that can lead to better generalization of our question-question ranking model.

3.1 Evaluation

We evaluate our performance on the main task of question relevancy ranking using the official

---

[2] http://www.fakenewschallenge.org/

Dataset	Train	Dev	Test
SemEval 16	2000	500	700
SemEval 17	-	-	880
FNC	49k	-	-
MNLI	433k	-	-

Figure 3: Size of datasets used for the experiments. Here we present the full size of FNC and MultiNLI training sets, however for our MTL experiments we used a random sample of the same size of the train, test and dev sets of the SemEval data as auxiliary data. The SemEval 17 shared task uses the same train and dev set as SemEval 16

SemEval-2017 Task 3 evaluation scripts (Nakov et al., 2017). The scripts provide a variety of metrics; however, in accordance with the shared task, we report Mean Average Precision (MAP) (the official metric for the SemEval 2016 and 2017 shared tasks); Mean Reciprocal Rank (MRR), which has being thoroughly used for IR and QA; Average Recall; and, finally, the accuracy of predicting relevant documents.

4 Results

The results from our experiments are shown in Table 1. We present the official metric from the SemEval task, as well as other common metrics. For the SemEval-16 data, our multitask MLP architecture with a question-answer auxiliary task performed best on all metrics, except accuracy, where the multi-task MLP using all auxiliary tasks performed best. We outperform the winning systems of both the SemEval 2016 and 2017 campaigns. In addition, our improvements from single-task to multi-task are significant ($p < 0.01$). We also outperform the official IR baseline used in the SemEval 2016 and 2017 shared tasks. We discuss the STL-LSTM-SIM results in §5. Furthermore, in Table 2, we show the performance of our models when training on feature combinations, while in Table 3, we present an ablation test where we remove one feature at a time.

Learning curve In Figure 4, we also present our learning curves for the development set when incrementally increasing the training set size. We observe that when using an auxiliary task, the learning is more stable across training set size.

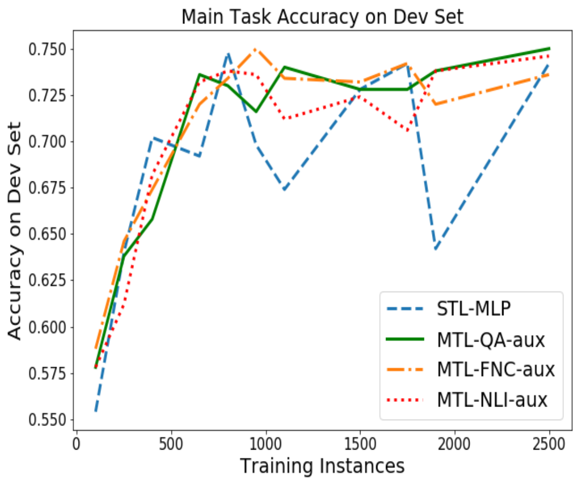

Figure 4: Learning curves for single- and multi-task learning. All MTL models outperform the STL model with very few training samples, and learning curves are more stable for MTL models than for STL.

	STL		MTL	
FEATURE SET	MAP	ACC	MAP	ACC
Unigrams	72.20	74.21	**73.37**	**74.9**
MeanEmb	**67.00**	71.28	65.39	**72.29**
Trigrams	**72.8**	77.8	72.00	**78.2**
Unigrams+Ngrams	73.62	77.00	**74.01**	**78.60**
Unigrams+MeanEmb	62.06	72.60	**70.68**	**74.14**
Emb+Trigrams	66.00	75.01	**72.25**	**77.48**

Table 2: Performance on development set of SemEval-16 when training *only* on certain feature combinations. MTL uses QA as auxiliary data.

5 Discussion

For the SemEval shared tasks on CQA, several authors used complex recurrent and convolutional neural network architectures (Severyn and Moschitti, 2015; Barrón-Cedeno et al., 2016). For example, Barrón-Cedeno et al. used a convolutional neural network in combination with feature vectors representing lexical, syntactic, and semantic similarity as well as tree kernels. Their performance was slightly lower than the best system (SemEval-Best for 2016 in Table 1). The best system used lexical and semantic similarity measures in combination with a ranking model based on support vector machines (SVMs) (Filice et al., 2016; Franco-Salvador et al., 2016). Both systems are harder to implement and train than the model we propose here. For SemEval-17, Franco-Salvador et al. (2016), the winning team used

FEATURE REMOVED		MAP	ACC
COSINE	unigram	69.25	**74.99**
	trigram	69.93	76.28
	embedding	69.11	76.40
MANHATTAN	unigram	70.33	76.71
	trigram	69.29	76.28
	embedding	66.90	75.28
BHATTACHARYA	unigram	70.83	75.85
	trigram	71.14	77.50
	embedding	71.72	77.28
EUCLIDEAN	unigram	71.43	76.57
	trigram	**65.55**	76.14
	embedding	70.60	75.57
JACCARD	unigram	67.41	75.70
	trigram	69.27	75.98

Table 3: We perform an ablation test, where we remove one feature at a time and report performance on development data of our single-task baseline. We observe that our baseline suffers most from removing the Euclidean distance over trigrams and the cosine similarity over unigrams. Note also that the Jaccard index over unigrams seems to carry a strong signal, albeit a very simple feature.

distributed representations of words, knowledge graphs and frames from FrameNet (Baker et al., 1998) as some of their features, and used SVMs for ranking.

For a more direct comparison, we also train a more expressive model than the simple MTL-based model we propose. This architecture is based on bi-directional LSTMs (Hochreiter and Schmidhuber, 1997). For this model, we input sequences of embedded words (using pre-trained word embeddings) from each query into independent BiLSTM blocks and output a vector representation for each query. We then concatenate the vector representations with the similarity features from our MTL model and feed it into a dense layer and a classification layer. This way we can evaluate the usefulness of the flexible, expressive LSTM network directly (as our MTL model becomes an ablation instance of the full, more complex architecture). We use the same dropout regularization and SGD values as for the MLP. Tuning all parameters on the development data, we do not manage to outperform our proposed model, however. See lines MTL-LSTM-SIM in Table 1 for results.

6 Conclusion

We show that simple feature engineering, combined with an auxiliary task and a simple feedforward neural architecture is appropriate for a small dataset and manages to beat the baseline and the best performing systems for the Semeval task of question relevancy ranking. We observe that introducing pairwise classification tasks leads to significant improvements in performance and a more stable model. Overall, our simple model introduces a new strong baseline which is particularly useful when there is a lack of labeled data.

Acknowledgments

The first author of this paper is funded by a BotXO PhD Award;[3] the last author by an ERC Starting Grant. We gratefully acknowledge the support of the NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

Isabelle Augenstein, Tim Rocktschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance Detection with Bidirectional Conditional Encoding. In *EMNLP*, pages 876–885. The Association for Computational Linguistics.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. In *NAACL*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley Framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Alberto Barrón-Cedeno, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad Al-Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 task 3: Answer and question selection for question answering on Arabic and English fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 896–903.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KELP at SemEval-2016 task 3: Learning semantic relations between questions and answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1116–1123.

---

[3] http://www.botxo.co/

Marc Franco-Salvador, Sudipta Kar, Thamar Solorio, and Paolo Rosso. 2016. Uh-prhlt at semeval-2016 task 3: Combining lexical and semantic-based features for community question answering. *Proceedings of SemEval*, pages 814–821.

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

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. Semeval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 269–281.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R Bowman. 2017. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. *arXiv preprint arXiv:1707.08172*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

Learning Sequence Encoders for Temporal Knowledge Graph Completion

Alberto García-Durán[1] **Sebastijan Dumančić**[*2] **Mathias Niepert**[1]

[1]NEC Labs Europe, Germany
{alberto.duran, mathias.niepert}@neclab.eu

[2]KU Leuven, Belgium
sebastijan.dumancic@cs.kuleuven.be

Abstract

Research on link prediction in knowledge graphs has mainly focused on static multi-relational data. In this work we consider temporal knowledge graphs where relations between entities may only hold for a time interval or a specific point in time. In line with previous work on static knowledge graphs, we propose to address this problem by learning latent entity and relation type representations. To incorporate temporal information, we utilize recurrent neural networks to learn time-aware representations of relation types which can be used in conjunction with existing latent factorization methods. The proposed approach is shown to be robust to common challenges in real-world KGs: the sparsity and heterogeneity of temporal expressions. Experiments show the benefits of our approach on four temporal KGs. The data sets are available under a permissive BSD-3 license[1].

1 Introduction

Knowledge graphs (KGs) are used to organize, manage, and retrieve structured information. The incompleteness of most real-world KGs has stimulated research on predicting missing relations between entities. A KG is of the form $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is a set of entities and, $\mathcal{R}$ is a set of relation types or predicates. One can represent $\mathcal{G}$ as a set of triples of the form (subject, predicate, object), denoted as $(\mathtt{s}, \mathtt{p}, \mathtt{o})$. The link prediction problem seeks the most probable completion of a triple $(\mathtt{s}, \mathtt{p}, ?)$ or $(?, \mathtt{p}, \mathtt{o})$ (Nickel et al., 2016). We focus on temporal KGs where some triples are augmented with time information and the link prediction problem asks for the most probable completion given time information. More formally, a temporal KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ is a KG where facts can also have the form (subject, predicate, object, timestamp) or (subject, predicate, object, time predicate, timestamp), in addition to $(\mathtt{s}, \mathtt{p}, \mathtt{o})$ triples. For instance, facts such as (Barack Obama, born, US, 1961) or (Barack Obama, president, US, occursSince, 2009-01) express temporal information about the facts associated with Barack Obama. While the former expresses that a relation type occurred at a specific point in time, the latter expresses an (open) time interval using the time predicate "occursSince." The latter example also illustrates a common challenge posed by the heterogeneity of time expressions due to variations in language and serialization standards.

Most approaches to link prediction are characterized by a scoring function that operates on the entity and relation type embeddings of a triple (Bordes et al., 2013; Yang et al., 2014; Guu et al., 2015). Learning representations that carry temporal information is challenging due to the sparsity and irregularities of time expressions. It is possible, however, to turn time expressions into sequences of tokens expressing said temporal information. Moreover, character-level architectures for language modeling (Zhang et al., 2015; Kim et al., 2016) operate on characters as atomic units to derive word embeddings. Inspired by these models, we propose a method to incorporate time information into standard embedding approaches for link prediction. We learn time-aware representations by training a recursive neural network with sequences of tokens representing the time predicate and the digits of the timestamp, if they exist. The last hidden state of the recurrent network is combined with standard scoring functions from the KG completion literature.

---

[*] Work done while interning at NEC Labs Europe

[1] https://github.com/nle-ml/mmkb

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4816–4821
Brussels, Belgium, October 31 - November 4, 2018.

2 Related Work

Reasoning with temporal information in knowledge bases has a long history and has resulted in numerous temporal logics (van Benthem, 1995). Several recent approaches extend statistical relational learning frameworks with temporal reasoning capabilities (Chekol et al., 2017; Chekol and Stuckenschmidt, 2018; Dylla et al., 2013).

There is also prior work on incorporating temporal information in knowledge graph completion methods. Jiang et al. (2016) capture the temporal ordering that exists between some relation types as well as additional common-sense constraints to generate more accurate link predictions. Esteban et al. (2016) introduce a prediction model for link prediction that assumes that changes to a KG are introduced by incoming events. These events are modeled as a separate event graph and used to predict the existence of links in the future. Trivedi et al. (2017) model the occurrence of a fact as a point process whose intensity function is influenced by the score assigned to the fact by an embedding function. Leblay and Chekol (2018) develop scoring functions that incorporate time representations into a TransE-type scoring function. Prior work has also incorporated numerical but non-temporal entity information for knowledge base completion (Garcia-Duran and Niepert, 2017).

Contrary to all previous approaches, we encode sequences of temporal tokens with an RNN. This facilitates the encoding of relation types with temporal tokens such as “since,” ”until,” and the digits of timestamps. Moreover, the RNN encoding provides an inductive bias for parameter sharing among similar timestamps (e.g., those occurring in the same century). Finally, our method can be combined with all existing scoring functions.

3 Time-Aware Representations

Embedding approaches for KG completion learn a scoring function f that operates on the embeddings of the subject $\mathbf{e}_s$, the object $\mathbf{e}_o$, and the predicate $\mathbf{e}_p$ of the triples. The value of this scoring function on a triple (s, p, o), $f(s,p,o)$, is learned to be proportional to the likelihood of the triples being true. Popular examples of scoring functions are

- TRANSE (Bordes et al., 2013)

$$f(s,p,o) = ||\mathbf{e}_s + \mathbf{e}_p - \mathbf{e}_o||_2. \quad (1)$$

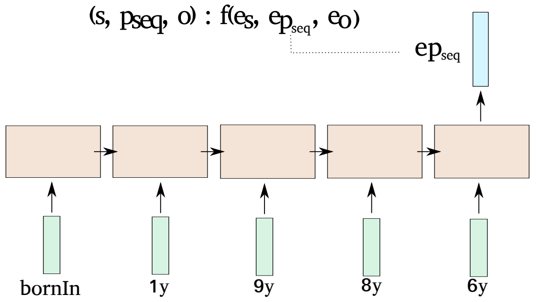

Figure 1: Learning time-aware representations.

- DISTMULT (Yang et al., 2014):

$$f(s,p,o) = (\mathbf{e}_s \circ \mathbf{e}_o)\mathbf{e}_p^T, \quad (2)$$

where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^d$ are the embeddings of the *subject* and *object* entities, $\mathbf{e}_p \in \mathbb{R}^d$ is the embedding of the relation type *predicate*, and $\circ$ is the element-wise product. These scoring functions do not take temporal information into account.

Given a temporal KG where some triples are augmented with temporal information, we can decompose a given (possibly incomplete) timestamp into a sequence consisting of some of the following *temporal tokens*

$$\overbrace{0 \cdot 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9}^{year}$$

$$\overbrace{01 \cdot 02 \cdot 03 \cdot 04 \cdot 05 \cdot 06 \cdot 07 \cdot 08 \cdot 09 \cdot 10 \cdot 11 \cdot 12}^{month}$$

$$\overbrace{0 \cdot 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9}^{day}$$

Hence, temporal tokens have a vocabulary size of 32. Moreover, for each triple we can extract a sequence of *predicate tokens* that always consists of the relation type token and, if available, a temporal modifier token such as “since” or “until.” We refer to the concatenation of the predicate token sequence and (if available) the sequence of temporal tokens as the predicate sequence p_{seq}. Now, a temporal KG can be represented as a collection of triples of the form (s, p_{seq}, o), wherein the predicate sequence may include temporal information. Table 1 lists some examples of such facts from a temporal KG and their corresponding predicate sequence. We use the suffix y, m and d to indicate whether the digit corresponds to year, month or day information. It is these sequences of tokens that are used as input to a recurrent neural network.

3.1 LSTMs for Time-Encoding Sequences

A long short-term memory (LSTM) is a neural network architecture particularly suited for modeling sequential data. The equations defining an

Fact	Predicate Sequence
(Barack Obama, country, US)	[country]
(Barack Obama, born, US, 1961)	[born, $1y$, $9y$, $6y$, $1y$]
(Barack Obama, president, US, since, 2009-01)	[president, since, $2y$, $0y$, $0y$, $9y$, $01m$]

Table 1: Facts and their corresponding predicate sequence.

LSTM are

$$\begin{aligned}\mathbf{i} &= \sigma_g(\mathbf{h}_{n-1}\mathbf{U}_i + \mathbf{x}_n\mathbf{W}_i)\\ \mathbf{f} &= \sigma_g(\mathbf{h}_{n-1}\mathbf{U}_f + \mathbf{x}_n\mathbf{W}_f)\\ \mathbf{o} &= \sigma_g(\mathbf{h}_{n-1}\mathbf{U}_o + \mathbf{x}_n\mathbf{W}_o)\\ \mathbf{g} &= \sigma_c(\mathbf{h}_{n-1}\mathbf{U}_g + \mathbf{x}_n\mathbf{W}_g)\\ \mathbf{c}_n &= \mathbf{f}\circ\mathbf{c}_{n-1} + \mathbf{i}\circ\mathbf{g}\\ \mathbf{h}_n &= \mathbf{o}\circ\sigma_h(\mathbf{c}_n)\end{aligned} \tag{3}$$

where $\mathbf{i}$, $\mathbf{f}$, $\mathbf{o}$ and $\mathbf{g}$ are the input, forget, output and input modulation gates, respectively. $\mathbf{c}$ and $\mathbf{h}$ are the cell and hidden state, respectively. All vectors are in $\mathbb{R}^h$. $\mathbf{x}_n \in \mathbb{R}^d$ is the representation of the n-th element of a sequence. In this paper we set $h = d$. σ_g, σ_c and σ_h are activation functions.

Each token of the input sequence p_{seq} is first mapped to its corresponding d-dimensional embedding via a linear layer and the resulting sequence of embeddings used as input to the LSTM. Each predicate sequence of length N is represented by the last hidden state of the LSTM, that is, $\boxed{\mathbf{e}_{p_{seq}} = \mathbf{h}_N}$. The predicate sequence representation, which carries time information, can now be used in conjunction with *subject* and *object* embeddings in standard scoring functions. For instance, temporal-aware versions of TRANSE and DISTMULT, which we refer to as TA-TRANSE and TA-DISTMULT, have the following scoring function for triples (s, p_{seq}, o):

$$\text{TA-TRANSE: } f(s, p_{seq}, o) = ||\mathbf{e}_s + \mathbf{e}_{p_{seq}} - \mathbf{e}_o||_2$$
$$\text{TA-DISTMULT: } f(s, p_{seq}, o) = (\mathbf{e}_s \circ \mathbf{e}_o)\mathbf{e}^T_{p_{seq}}.$$

All parameters of the scoring functions are learned jointly with the parameters of the LSTMs using stochastic gradient descent.

The advantages of character level models to encode time information for link prediction are: (1) the usage of digits and modifiers such as “since” as atomic tokens facilitates the transfer of information across similar timestamps, leading to higher efficiency (e.g. small vocabulary size); (2) at test time, one can obtain a representation for a timestamp even though it is not part of the training set; (3) the model can use triples with and without temporal information as training data. Figure 1 illustrates the generic working of our approach.

4 Experiments

We conducted experiments on four different KG completion data sets where a subset of the facts are augmented with time information.

4.1 Datasets

Integrated Crisis Early Warning System (ICEWS) is a repository that contains political events with a specific timestamp. These political events relate entities (e.g. countries, presidents...) to a number of other entities via logical predicates (e.g. ’Make a visit’ or ’Express intent to meet or negotiate’). Additional information can be found at `http://www.icews.com/`. The repository is organized in dumps that contain the events that occurred each year from 1995 to 2015. We created two temporal KGs out of this repository, i) a short-range version that contains all events in 2014, and ii) a long-range version that contains all events occurring between 2005-2015. We refer to these two data sets as ICEWS 2014 and ICEWS 2005-15, respectively. Due to the large number of entities we selected a subset of the most frequently occurring entities in the graph and all facts where both the *subject* and *object* are part of this subset of entities. We split the facts into training, validation and test in a proportion of 80%/10%/10%, respectively. The protocol for the creation of these data sets is identical to the onw followed in previous work (Bordes et al., 2013). To create YAGO15K, we used FREEBASE15K (Bordes et al., 2013) (FB15K) as a blueprint. We aligned entities from FB15K to YAGO (Hoffart et al., 2013) with SAMEAS relations contained in a YAGO dump[2], and kept all facts involving those entities. Finally, we augment this collection of facts with time information from the “yagoDateFacts”[3] dump. Contrary to the

---

[2] /yago-naga/yago3.1/yagoDBpediaInstances.ttl.7z

[3] /yago-naga/yago3.1/yagoDateFacts.ttl.7z

Data set	YAGO15k	ICEWS '14	ICEWS 05-15	Wikidata
Entities	15,403	6,869	10,094	11,134
Relationships	34	230	251	95
#Facts	138,056	96,730	461,329	150,079
#Distinct TS	198	365	4,017	328
Time Span	1513-2017	2014	2005-2015	25-2020
Training	110,441 [29,381]	72,826 [72,826]	368,962 [368,962]	121,422 [121,422]
Validation	13,815 [3,635]	8,941 [8,941]	46,275 [46,275]	14,374 [14,374]
Test	13,800 [3,685]	8,963 [8,963]	46,092 [46,092]	14,283 [14,283]

Table 2: Statistics of the data sets. TS stands for timestamps. The number of facts with time information is in brackets.

	YAGO15k				Wikidata			
	MRR	MR	Hits@10	Hits@1	MRR	MR	Hits@10	Hits@1
TTransE	**32.1**	578	51.0	23.0	48.8	80	80.6	33.9
TransE	29.6	614	46.8	22.8	31.6	**50**	65.9	18.1
DistMult	27.5	578	43.8	21.5	31.6	77	66.1	18.1
TA-TransE	**32.1**	564	**51.2**	**23.1**	48.4	79	**80.7**	32.9
TA-DistMult	29.1	**551**	47.6	21.6	**70.0**	198	78.5	**65.2**

Table 3: Results (filtered setting) of the temporal knowledge graph completion experiments for the data sets YAGO15k and Wikidata. The best results are written bold.

	ICEWS 2014				ICEWS 2005-15			
	MRR	MR	Hits@10	Hits@1	MRR	MR	Hits@10	Hits@1
TTransE	25.5	148	60.1	7.4	27.1	181	61.6	8.4
TransE	28.0	**122**	63.7	9.4	29.4	84	66.3	9.0
DistMult	43.9	189	67.2	32.3	45.6	90	69.1	33.7
TA-TransE	27.5	128	62.5	9.5	29.9	**79**	66.8	9.6
TA-DistMult	**47.7**	276	**68.6**	**36.3**	**47.4**	98	**72.8**	**34.6**

Table 4: Results (filtered setting) of the temporal knowledge graph completion experiments for the data sets ICEWS 2014 and ICEWS 2005-15. The best results are written bold.

ICEWS data sets, YAGO15k does contain temporal modifiers; namely, 'occursSince' and 'occursUntil'. Contrary to previous work (Leblay and Chekol, 2018), all facts maintain time information in the same level of granularity as one can find in the original dumps these data sets come from.

We also experimented with the temporal facts from the Wikidata data set[4] extracted in (Leblay and Chekol, 2018). Only information regarding the year is available for these facts, since the authors discarded information of finer granularity. All facts are framed in a time interval (i.e. they contain the temporal modifiers 'occursSince' and 'occursUntil'). Facts annotated with a single point-in-time are associated with that time-point as start and end time. Due to the large number of entities of this data set, which hinders the computation of standard KG completion metrics, we selected a subset of the most frequent entities and kept all facts where both the *subject* and *object* are part of this subset of entities. This set of filtered facts was split into training, validation and test in the same proportion as before.

Table 2 lists some statistics of the temporal KGs. All four data sets, with their corresponding training, validation, and test splits are available at `https://github.com/nle-ml/mmkb`.

4.2 General Set-up

We evaluate various methods by their ability to answer completion queries where i) all the arguments of a fact are known except the *subject* entity, and ii) all the arguments of a fact are known except the *object* entity. For the former we replace the subject by each of the KBs entities $\mathcal{E}$ in turn, sort the triples based on the scores returned by the different methods, and computed the rank of the correct entity. We repeated the same process for the second completion task and average the results.

---

[4] http://staff.aist.go.jp/julien.leblay/datasets

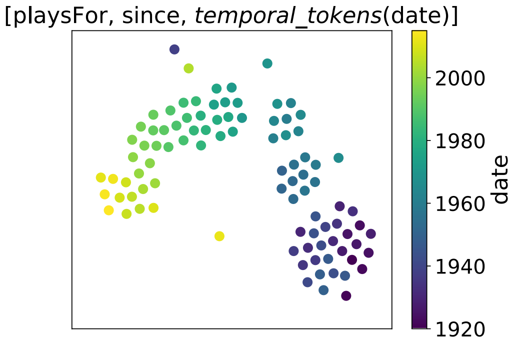

Figure 2: T-SNE visualization of the embeddings learned for the predicate sequence p_{seq} = [playsFor, occursSince, *date*], where *date* corresponds to the date token sequence.

This is standard procedure in the KG completion literature. We also report the filtered setting as described in (Bordes et al., 2013). The mean of all computed ranks is the Mean Rank (lower is better) and the fraction of correct entities ranked in the top n is called hits@n (higher is better). We also compute the Mean Reciprocal Rank (higher is better) which is less susceptible to outliers.

Recent work (Leblay and Chekol, 2018) evaluates different approaches for performing link prediction in temporal KGs. The approach that learns independent representations for each timestamp and use these representations as translation vectors, similarly to (Bordes et al., 2013), leads to the best results. This approach is called Vector-based TTransE, though for the shake of simplicity in the paper we refer to it as TTransE. We compare our approaches TA-TransE and TA-distMult against TTransE, and the standard embedding methods TransE and distMult. For all approaches, we used ADAM (Kingma and Ba, 2014) for parameter learning in a mini-batch setting with a learning rate of 0.001, the categorical cross-entropy (Kadlec et al., 2017) as loss function and the number of epochs was set to 500. We validated every 20 epochs and stopped learning whenever the MRR values on the validation set decreased. The batch size was set to 512 and the number of negative samples to 500 for all experiments. The embedding size is d=100. We apply dropout (Srivastava et al., 2014) for all embeddings. We validated the dropout from the values $\{0, 0.4\}$ for all experiments. For TA-TransE and TA-distMult, the activation gate σ_g is the sigmoid function; σ_c and σ_h were chosen to be linear activation functions.

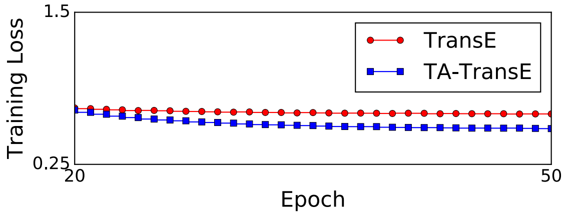

Figure 3: Training loss in YAGO15k. TA-TransE's ability to learn from time information leads to a lower loss.

4.3 Results

Table 3 and 4 list the results for the KG completion tasks. TA-TransE and TA-distMult systematically improve TransE and distMult in MRR, hits@10 and hits@1 in almost all cases. Mean rank is a metric that is very susceptible to outliers and hence these improvements are not consistent. TTransE learns independent representations for each timestamp contained in the training set. At test time, timestamps unseen during training are represented by null vectors. This explains that TTransE is only competitive in YAGO15k, wherein the number of distinct timestamps is very small (see #Distinct TS in Table 2) and thus enough training examples exist to learn robust timestamp embeddings. TTransE's performance is similar to that of TA-TransE, our time-aware version of TransE, in Wikidata. Similarly, TTransE can learn robust timestamp representations because of the small number of distinct timestamps of this data set.

Figure 3 shows a comparison of the training loss of TransE and TA-TransE for YAGO15k. Under the same set-up, TA-TransE's ability to learn from time information leads to a training loss lower than that of TransE.

Figure 2 shows a t-SNE (Maaten and Hinton, 2008) visualization of the embeddings learned for the predicate sequence p_{seq} = [playsFor, occursSince, *date*], where *date* corresponds to the date token sequence. This illustrates that the learned relation type embeddings carry temporal information.

5 Conclusions

We propose a digit-level LSTM to learn representations for time-augmented KG facts that can be used in conjunction with existing scoring functions for link prediction. Experiments in four temporal knowledge graphs show the effectiveness of the approach.

References

J. van Benthem. 1995. Handbook of logic in artificial intelligence and logic programming (vol. 4). chapter Temporal Logic, pages 241–350. Oxford University Press, Inc., New York, NY, USA.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Melisachew Wudage Chekol, Giuseppe Pirrò, Joerg Schoenfisch, and Heiner Stuckenschmidt. 2017. Marrying uncertainty and time in knowledge graphs. In *AAAI*, pages 88–94.

Melisachew Wudage Chekol and Heiner Stuckenschmidt. 2018. Rule Based Temporal Inference. In *Technical Communications of the 33rd International Conference on Logic Programming (ICLP 2017)*, volume 58, pages 4:1–4:14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Maximilian Dylla, Iris Miliaraki, and Martin Theobald. 2013. A temporal-probabilistic database model for information extraction. *Proc. VLDB Endow.*, 6(14):1810–1821.

C. Esteban, V. Tresp, Y. Yang, S. Baier, and D. Krompa. 2016. Predicting the co-evolution of event and knowledge graphs. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 98–105.

Alberto Garcia-Duran and Mathias Niepert. 2017. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *arXiv preprint arXiv:1709.04676*.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Tingsong Jiang, Tian Yu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. 2016. Towards time-aware knowledge graph completion. In *COLING*.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. *arXiv preprint arXiv:1705.10744*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, WWW '18. International World Wide Web Conferences Steering Committee.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3462–3471, International Convention Centre, Sydney, Australia. PMLR.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Learning multi-relational semantics using neural-embedding models. *arXiv preprint arXiv:1411.4072*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Similar but not the Same: Word Sense Disambiguation Improves Event Detection via Neural Representation Matching

Weiyi Lu[†] **and Thien Huu Nguyen**[#‡]
[†] Computer Science Department, New York University, USA
[#] Montreal Institute for Learning Algorithms, University of Montreal, Canada
[‡] Department of Computer and Information Science, University of Oregon, USA
`weiyi.lu@nyu.edu,thien@cs.uoregon.edu`

Abstract

Event detection (ED) and word sense disambiguation (WSD) are two similar tasks in that they both involve identifying the classes (i.e. event types or word senses) of some word in a given sentence. It is thus possible to extract the knowledge hidden in the data for WSD, and utilize it to improve the performance on ED. In this work, we propose a method to transfer the knowledge learned on WSD to ED by matching the neural representations learned for the two tasks. Our experiments on two widely used datasets for ED demonstrate the effectiveness of the proposed method.

1 Introduction

An important aspect of natural language processing involves understanding events mentioned in text. Towards this end, event detection (ED) is the task of locating event triggers (usually verbs or nouns) within a given text, and classifying them among a given set of event types. This task remains challenging due to the inherent ambiguity and flexibility of natural languages. The current state-of-the-art methods for ED have involved applying deep learning (DL) models to automatically extract feature representations of the text, and then treating the task as a classification problem (Chen et al., 2015; Nguyen and Grishman, 2015b).

The major intuition in this paper is that the task of ED is closely related to the task of word sense disambiguation (WSD) whose datasets can help to improve the performance of the DL models for ED. This is due to the goal of WSD to determine the sense of a word within a particular context, given a set of possible senses that the word can take on. Our intuition is based on the two following aspects:

(i) Similar Context Modeling: Given a word in a context/sentence, both ED and WSD models need to select/predict a correct label in a list of candidate labels for the word. For WSD, the candidate labels are the possible senses (e.g, sense ids in WordNet) that the word of interest can have, while for ED, they are the set of predetermined event types (e.g, the event subtypes in the ACE 2005 dataset[1]). Consider the word "*fired*" in the following sentence as an example:

The boss ***fired*** *his secretary today.*

For WSD, there are 12 possible senses for the verb "*fire*" in WordNet in which the correct label for the word "*fired*" in this case is the sense id "*fire%2:41:00::*" (i.e, "*terminate the employment of*"). The ED task in the ACE 2005 dataset, on the other hand, involves 33 possible event subtypes with "*End-Position*" as the correct event subtype/label for the word "*fired*" in our example.

In order to make such label predictions, both ED and WSD need to model the word itself and its context (i.e, the words "*fired*", "*boss*", and "*secretary*" in the example). This similar modeling allows the same DL model to be adopted for both ED and WSD, facilitating the use of WSD data to improve the feature representations for ED via parameter/representation tying.

(ii) Close Semantic Consideration: As there are some overlaps between the semantic differentiation in WSD and ED, the knowledge/information from WSD about a particular word in a context can help to make a better prediction for that word in ED. For instance, in the example above, the knowledge from WSD that the word "*fired*" is referring to a termination of employment would clearly help ED to identify "*End-Position*" as the correct event type (rather than the incorrect event type "*Attack*") for "*fired*" in this case.

How can we exploit this intuition to improve the performance of the DL models for ED with WSD

---

[1] `https://www.ldc.upenn.edu/collaborations/past-projects/ace`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4822–4828
Brussels, Belgium, October 31 - November 4, 2018.

data? In this work, we propose a novel method based on representation matching to transfer the knowledge learned from the WSD data to the DL models for ED. In particular, two separate deep learning models are employed to model the context for WSD and ED. The two models share the network architecture, but involve different parameters that are specific to the tasks. We then transfer the knowledge from the WSD network to the ED network by ensuring that the feature representations learned by the two networks on the same contexts are similar to each other.

We demonstrate the effectiveness of the proposed method on two widely used datasets for ED. To the best of our knowledge, this is the first work to study the transfer learning/multi-task learning methods for WSD and ED with DL.

2 Model

We consider the typical setting where we have two separate datasets $D^{wsd} = \{W_i^{wsd}, p_i^{wsd}, y_i^{wsd}\}$ for WSD and $D^{ed} = \{W_i^{ed}, p_i^{ed}, y_i^{ed}\}$ for ED. Here, W_i^{ed} is the i-the sentence of D^{ed}, p_i^{ed} is the index of the word of interest for event type prediction in W_i^{ed}, and y_i^{ed} is the corresponding event type label. The same conventions apply for $W_i^{wsd}, p_i^{wsd}, y_i^{wsd}$. Also, let Y^{wsd} and Y^{ed} be the label sets for WSD and ED respectively (i.e, $y_i^{wsd} \in Y^{wsd}$ and $y_i^{ed} \in Y^{ed}$). Our goal is to transfer the knowledge learned from the D^{wsd} dataset to improve the performance of the ED models trained on the D^{ed} dataset (multi-task learning).

In the following, we will first describe the deep learning architectures to transform the sentences W in the datasets D^{wsd} and D^{ed} into representation vectors. We only focus on the deep learning architectures proposed for ED in the literature to achieve compatible comparisons for ED. The proposed multi-task learning method for ED with the WSD dataset will follow.

2.1 Computing the Feature Representations

Consider a sentence W in the datasets D^{wsd} or D^{ed} that is represented as a sequence of tokens $W = [w_0, w_1, \ldots, w_t]$. Let p be the index of the word of interest in this sentence. The context for w_p in W is constructed by taking the word itself, the n preceding words, and the n following words (padding or truncating when necessary). The tokens in the context are re-indexed to form an instance $V = [v_0, v_1, \ldots, v_n, \ldots, v_{2n-1}, v_{2n}]$, where v_n corresponds to w_p in W.

Encoding

The first step to prepare the instance V for the deep learning models is to map each token v_j in V into two real-valued vectors, which are then concatenated to form a vector representation x_j for v_j (Nguyen and Grishman, 2015b; Chen et al., 2015):

1. The word embedding of v_j obtained by looking up the token v_j in the pre-trained word embedding table (Mikolov et al., 2013a).

2. The position embedding vector for v_j: obtained by looking up the relative distance $j - n$ of v_j with respect to the token of interest v_n in a position embedding table (randomly initialized) (Chen et al., 2015; Nguyen and Grishman, 2015a).

It is important to note that, different from the prior works (Nguyen and Grishman, 2015b; Liu et al., 2017), we do not include the entity type label of each token into its representation. This is a more realistic setting for our work as the golden entity mentions do not always exist in practice, especially for the datasets in WSD.

Once each token v_j is converted into the representation vector x_j, the instance V becomes a sequence of vectors $X = [x_0, x_1, \ldots, x_n, \ldots, x_{2n-1}, x_{2n}]$ that would be fed into the one of the following deep learning models to learn a feature representation R for V.

Typical Deep Learning Models for ED

1. *CNN*: This is the convolutional neural networks in(Nguyen and Grishman, 2015b; Chen et al., 2015). It features convolution operations that are performed over the k consecutive vectors (k-grams) in X and followed by a max-pooling layer to generate the representation vector R for V. Multiple window values k are used to enhance the coverage of the model over the hidden k-grams in the context.

2. *NCNN* (Nguyen and Grishman, 2016d): This model is similar to *CNN*. The only difference is instead of running the convolution over the k consecutive vectors, *NCNN* convolutes over the k arbitrarily non-consecutive k vectors in V. This helps *NCNN* to explicitly model the non-consecutive words in the context to improve ED.

3. *BiRNN*: This is the bidirectional recurrent neural network (RNN) for event extraction in (Nguyen et al., 2016a). The model is

composed of two recurrent neural networks (RNN), where one runs forward and the other runs backward through the input sequence V. The hidden vectors produced by the two networks are then concatenated at each position in the context. The vector at the position of n for the word of interest is used as the representation vector R for V. Due to the property of RNN, R encodes the information over the whole input V with a greater focus on v_n.

4. *CNN+BiRNN*: In this model (Feng et al., 2016), X is passed through both a *CNN* and a *BiRNN* whose results are concatenated to produce the hidden representation R for ED. The expectation is to take advantage of the modeling abilities from both the *CNN* and *BiRNN* architectures for ED.

In practice, the representation vector R (obtained from one of the deep learning models above) is also concatenated with the word embeddings of the tokens surrounding the token of interest w_n to improve its expressiveness (Chen et al., 2015; Nguyen and Grishman, 2016d). We would use this extended version when we refer to R in the following.

In the final step, the representation vector R is fed into a feed-forward neural network followed by a softmax layer to perform predictions for ED and WSD.

For convenience, we denote the whole process that a DL model M is used to compute the representation vector R for the input sentence W with the token index p of interest as: $R = M(W, p)$.

2.2 Multi-task Learning Models

The previous section has described the deep learning methods that can be employed to train the models for ED and WSD separately. This section presents our proposed method to transfer the knowledge from the WSD dataset to improve the performance for ED.

A typical method for transfer learning/multi-task learning in NLP is to alternate the training process for the parameter-shared models of the related tasks (possibly with different datasets) (Guo et al., 2016; Li et al., 2015; Liu et al., 2016). For instance, in (Guo et al., 2016), the authors use the same deep learning model to learn the feature representations for the text inputs of two related tasks. This is then followed by task-specific output layers to perform the corresponding tasks. Note that the two tasks in (Guo et al., 2016) are provided with two different datasets of different text inputs, thereby being similar to the setting we consider in this work. In order to learn the parameters for this model, in each iteration, (Guo et al., 2016) select one of the tasks with some probabilities, sample a mini-batch of examples in the dataset of the chosen task, and update the model parameters using the objective function specific to the chosen task. Consequently, the model parameters for feature representation learning are updated at every iteration while only the model parameters in the output layer for the chosen task are updated at the current iteration.

It has been demonstrated in (Guo et al., 2016) that the alternating method (called *ALT*) is more effective than pre-training the network on a related task and fine-tuning it on the expected task. We thereby consider *ALT* as the baseline for multi-task learning in our work. However, we argue that this baseline is not effective enough to transfer the knowledge from the WSD dataset to ED in our case. This stems from its employment of a single DL model to induce the representations for the text inputs in both tasks. In our case of WSD and ED, although there are some overlap between the semantic differentiation of the two tasks, the labels in the WSD datasets (i.e, the sense ids) tend to be more fine-grained and exhaustive than those in ED. For instance, for the word "*fire*", there might be 12 WSD labels for it in WordNet while the number of possible event types for "*fire*" in the ACE 2005 dataset is only 2 (i.e, "*End-Position*" and "*Attack*"). Eventually, if a single DL model is used to compute the representations for the text inputs in both WSD and ED, the model would suffer from a confusion to distinguish such subtlety in the semantic differentiation.

In order to overcome this issue, we propose to employ two versions M^{wsd} and M^{ed} of the same DL model (with different model parameters) to compute the feature representations for WSD and ED respectively. We then transfer the knowledge from M^{wsd} to M^{ed} by encouraging the representations generated by the two versions M^{wsd} and M^{ed} on the same text inputs to be similar. Formally, let (W^t, p^t, y^t) be an example in the D^{wsd} or D^{ed} dataset ($t \in \{wsd, ed\}$). Also, let R^{wsd} and R^{ed} be the representations for (W^t, p^t) induced by M^{wsd} and M^{ed} respectively:

$$R^{wsd} = M^{wsd}(W^t, p^t), R^{ed} = M^{ed}(W^t, p^t)$$

Such representation vectors are then followed by a task-specific output layer F^t (i.e, feed-forward neural networks followed by a softmax layer) to compute the probability distribution over the possible labels for (W^t, p^t): $P^t(Y^t|R^t) = F^t(R^t)$ where Y^t is the label set for the t task.

If the two models M^{wsd} and M^{ed} were trained separately, the objective function for the t task for the current example would be the negative log-likelihood: $C^t(W^t, p^t, y^t) = -\log P^t(y^t|R^t)$. In this work, instead of just optimizing this objective, we optimize the joint function:

$$C^t(W^t, p^t, y^t) = -\log P^t(y^t|R^t) + \lambda \frac{1}{d_R} \sum_{i=0}^{d_R} \left(R_i^{wsd} - R_i^{ed} \right)^2$$

where λ is a trade-off parameter and d_R is the dimension of the representation vectors.

The second term in the joint objective function enforces that the feature representations learned by M^{wsd} and M^{ed} on the same input context (W^t, p^t) are close to each other ($t \in \{wsd, ed\}$). One the one hand, this representation matching schema helps the two models to communicate to each other so the knowledge from one model can be passed to the other one. On the other hand, the use of two separate models leaves a flexibility for the models to induce the task-specific structures.

Presumably, the objective function (2.2) can simultaneously improve the performance for both tasks of consideration. However, in our case of ED and WSD, it turns out this mechanism actually worsen the performance of the WSD models that were trained separately. We attribute this to the fact that the semantic differentiation in ED is more coarse-grained that that of WSD, causing the ineffectiveness of the datasets for ED to improve WSD performance. Eventually, we will just focus on the ED performance in the experiments.

3 Experiments

3.1 Parameters and Datasets

We use the Semcor dataset (Miller et al., 1994) as the dataset for WSD in this work. This dataset was extracted from the Brown Corpus, and manually annotated with WordNet senses. We evaluate the models on two different datasets for ED:

1. **ACE 2005**: This dataset has 33 event subtypes. We use the same data split with the prior work (Chen et al., 2015; Nguyen and Grishman, 2015b). In particular, 40 newswire documents are used for testing, 30 other documents are reserved for validation, and the 529 remaining documents form the training data.
2. **TAC 2015**: This dataset was released in the Event Nugget Detection Evaluation of the 2015 Text Analysis Conference (TAC) (Mitamura et al., 2015). It comes with 38 event subtypes. We follow the data split in the official evaluation to achieve compatible comparison. As TAC 2015 does not have a development set, we use the best parameters tuned on ACE 2005 for the experiments with TAC 2015.

We use the pre-trained word embeddings provided by (Nguyen and Grishman, 2016d). For *CNN*, *NCNN* and *CNN+BiRNN*, we employ filter sizes of $\{2, 3, 4, 5\}$ with 300 filters for each size as in (Nguyen and Grishman, 2015b), while Gated Recurrent Units (Cho et al., 2014) with 300 hidden units are applied in *BiRNN* and *CNN+BiRNN* (as do (Nguyen and Grishman, 2016d)). For the other parameters, the best values suggested by the development data include: a dropout rate of 0.5, a feed-forward neural network with one hidden layer of 1200 hidden units for the output layers, and the penalty rate λ of 0.01 for both CNN and BiRNN, 0.6 for NCNN, and 0.7 for CNN+BiRNN in the proposed transfer learning method (called *MATCHING*). For simplicity, the same hyper-parameters are used for the two versions of the same network architecture in the *MATCHING* method. We utilize Adadelta (Zeiler, 2012) with back-propagation to train the models in this work.

3.2 Experiments

In this section, we compare the proposed *MATCHING* method with the transfer learning baseline *ALT* in (Guo et al., 2016) and the separate training mechanism for ED (called *SEPARATE*) employed in the previous work for ED (Chen et al., 2015; Nguyen and Grishman, 2015b). Note that in the *SEPARATE* method, the models are only trained on the datasets for ED without utilizing any transfer learning techniques with external datasets. We report the performance when each of the DL methods in Section 2.1 is used as the network to learn the feature representations for ED and WSD.

Tables 1 and 2 present the performance (i.e, F1 scores) of the models on the ACE 2005 and TAC 2015 datasets respectively. The first observation is that the proposed transfer learning method *MATCHING* is consistently better than the baseline method *ALT* across different deep learning models and datasets with large performance gap. This is significantly with $p < 0.05$ and confirms our hypothesis in Section 2.2 about the advantage of the proposed *MATCHING* over the alternating training method *ALT* for ED and WSD. In fact, the performance of the *ALT* method is even worse than the traditional *SEPARATE* method also over different network architectures and datasets. Consequently, training a single deep learning model on a combination of ED and WSD data (as in *ALT*) does not automatically enable the model to learn to exploit the similar structures of the two tasks. In contrast, it hinders the model's ability to effectively extract hidden representations for ED.

Comparing *MATCHING* and *SEPARATE*, we see that *MATCHING* helps to improve *SEPARATE* with respect to difference choices of the DL models. The performance improvement is significant for *CNN* and *BiRNN* on ACE 2005 and for all the models on TAC 2015. Such results demonstrate the effectiveness of the WSD dataset for ED and the ability of the proposed method *MATCHING* to promote knowledge transferring between WSD and ED to improve ED performance.

Regarding the best reported performance, our best performance on ACE (i.e, 71.2% with *CNN*) is comparable with the recent state-of-the-art performance (i.e, Table 1). However, we note that such work heavily relies on the manual annotation of the entity mentions in the documents. Our current work do not employ such information to better reflect the realistic setting. For the TAC 2015 dataset, our best performance is 60.7% with *CNN+BiRNN* although the performance of the other models is also very close. This performance is better than the best performance that has been reported on the TAC 2015 (i.e, Table 2).

Method	CNN	BiRNN	NCNN	CNN+BiRNN
SEPARATE	67.6	67.6	69.3	68.1
ALT	65.1	66.4	65.0	65.2
MATCHING	**71.2**	**69.0**	**69.6**	**68.3**
(Nguyen and Grishman, 2016d)			71.3*	
(Liu et al., 2017)			71.9*	
(Liu et al., 2018)			72.4*	
(Nguyen and Grishman, 2018a)			73.1*	

Table 1: Performance on the ACE 2005 dataset. * indicates the use of entity mention annotation.

Method	CNN	BiRNN	NCNN	CNN+BiRNN
SEPARATE	57.6	59.4	58.3	58.0
ALT	57.6	54.9	48.5	57.5
MATCHING	**60.0**	**60.4**	**60.0**	**60.7**
TAC TOP (Mitamura et al., 2015)			58.4*	
(Nguyen and Grishman, 2018a)			58.8*	

Table 2: Performance on the TAC 2015 dataset. * indicates the use of entity mention annotation.

4 Related Work

Prior works on ED include statistical models with manual feature engineering(Ahn, 2006; Ji and Grishman, 2008; Hong et al., 2011; Li et al., 2013; Venugopal et al., 2014; Li et al., 2015), followed by neural network models, such as CNNs (Nguyen and Grishman, 2015b; Chen et al., 2015; Nguyen et al., 2016b,e; Chen et al., 2017), RNNs (Nguyen et al., 2016a; Jagannatha and Yu, 2016), and attention-based methods (Liu et al., 2017; Nguyen and Nguyen, 2018b).

A similar trend exists in methods proposed for WSD, with feature based methods (Miller et al., 1994; Zhong and Ng, 2010; Taghipour and Ng, 2015) succeeded recently by deep learning methods (Yuan et al., 2016; Raganato et al., 2017).

For multi-task learning in NLP, methods have been proposed for jointly modeling structured prediction tasks (Hatori et al., 2012; Li et al., 2011; Bohnet and Nivre, 2012; Henderson et al., 2013; Lluís et al., 2013; Duong et al., 2015), and for sequence-to-sequence problems (Dong et al., 2015; Luong et al., 2015; Liu et al., 2016; Klerke et al., 2016). The prior work to solve multiple NLP tasks using an unified architecture includes (Collobert and Weston, 2008; Guo et al., 2016).

5 Conclusion

We present a method that improves the performance of deep learning models for ED by training two different versions of the same network architecture for ED and WSD, while encouraging the knowledge transfer between the two versions via representation matching. The proposed method produces better results across a variety of deep learning models.

References

David Ahn. 2006. The stages of event extraction. In *ACL*.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *EMNLP-CoNLL*.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *ACL*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL-IJCNLP*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL-IJCNLP*.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *ACL*.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *ACL*.

Jiang Guo, Wanxiang Che, Haifeng Wang, Ting Liu, and Jun Xu. 2016. A unified architecture for semantic role labeling and relation classification. In *COLING*.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *ACL*.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational linguistics*, 39(4):949–998.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *ACL*.

Abhyuday N Jagannatha and Hong Yu. 2016. Bidirectional rnn for medical event detection in electronic health records. In *NAACL*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. *arXiv preprint arXiv:1604.03357*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with abstract meaning representation. In *Proceedings of ACL-IJCNLP Workshop on Computing News Storylines (CNewS)*.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *EMNLP*.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018. Event detection via gated multilingual attention mechanism. In *AAAI*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *ACL*.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *AAAI*.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *ACL*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR*.

George A Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G Thomas. 1994. Using a semantic concordance for sense identification. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics.

Teruko Mitamura, Zhengzhong Liu, and Eduard Hovy. 2015. Overview of tac kbp 2015 event nugget track. In *TAC*.

Minh Nguyen and Thien Huu Nguyen. 2018b. Who is killed by police: Introducing supervised attention for hierarchical lstms. In *COLING*.

Thien Huu Nguyen, , Adam Meyers, and Ralph Grishman. 2016e. New york university 2016 system for kbp event nugget: A deep learning approach. In *TAC*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *NAACL*.

Thien Huu Nguyen, Lisheng Fu, Kyunghyun Cho, and Ralph Grishman. 2016b. A two-stage approach for extending event detection to new types via neural networks. In *Proceedings of the 1st ACL Workshop on Representation Learning for NLP (RepL4NLP)*.

Thien Huu Nguyen and Ralph Grishman. 2015a. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Thien Huu Nguyen and Ralph Grishman. 2015b. Event detection and domain adaptation with convolutional neural networks. In *ACL-IJCNLP*.

Thien Huu Nguyen and Ralph Grishman. 2016d. Modeling skip-grams for event detection with convolutional neural networks. In *EMNLP*.

Thien Huu Nguyen and Ralph Grishman. 2018a. Graph convolutional networks with argument-aware pooling for event detection. In *AAAI*.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *EMNLP*.

Kaveh Taghipour and Hwee Tou Ng. 2015. One million sense-tagged instances for word sense disambiguation and induction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *EMNLP*.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. *arXiv preprint arXiv:1603.07012*.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. In *CoRR, abs/1212.5701*.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL*.

Learning Word Representations with Cross-Sentence Dependency for End-to-End Co-reference Resolution

Hongyin Luo
CSAIL, MIT
`hyluo@mit.edu`

James Glass
CSAIL, MIT
`glass@mit.edu`

Abstract

In this work, we present a word embedding model that learns cross-sentence dependency for improving end-to-end co-reference resolution (E2E-CR). While the traditional E2E-CR model generates word representations by running long short-term memory (LSTM) recurrent neural networks on each sentence of an input article or conversation separately, we propose linear sentence linking and attentional sentence linking models to learn cross-sentence dependency. Both sentence linking strategies enable the LSTMs to make use of valuable information from context sentences while calculating the representation of the current input word. With this approach, the LSTMs learn word embeddings considering knowledge not only from the current sentence but also from the entire input document. Experiments show that learning cross-sentence dependency enriches information contained by the word representations, and improves the performance of the co-reference resolution model compared with our baseline.

1 Introduction

Co-reference resolution requires models to cluster mentions that refer to the same physical entities. The models based on neural networks typically require different levels of semantic representations of input sentences. The models usually need to calculate the representations of word spans, or mentions, given pre-trained character and word-level embeddings (Turian et al., 2010; Pennington et al., 2014) before predicting antecedents. The mention-level embeddings are used to make co-reference decisions, typically by scoring mention pairs and making links (Lee et al., 2017; Clark and Manning, 2016a; Wiseman et al., 2016). Long short-term memories (LSTMs) are often used to encode the syntactic and semantic information of input sentences.

Articles and conversations include more than one sentences. Considering the accuracy and efficiency of co-reference resolution models, the encoder LSTM usually processes input sentences separately as a batch (Lee et al., 2017). The disadvantage of this method is that the models do not consider the dependency among words from different sentences, which plays a significant role in word representation learning and co-reference predicting. For example, pronouns are often linked to entities mentioned in other sentences, while their initial word vectors lack dependency information. As a result, a word representation model cannot learn an informative embedding of a pronoun without considering cross-sentence dependency in this case.

It is also problematic if we encode the input document considering cross-sentence dependency and treat the entire document as one sentence. An input article or conversation can be too long for a single LSTM cell to memorize. If the LSTM updates itself for too many steps, gradients will vanish or explode (Pascanu et al., 2013), and the co-reference resolution model will be very difficult to optimize. Regarding the entire input corpus as one sequence instead of a batch also significantly increases the time complexity of the model.

To solve the problem that traditional LSTM encoders, which treat the input sentences as a batch, lack an ability to capture cross-sentence dependency, and to avoid the time complexity and difficulties of training the model concatenating all input sentences, we propose a cross-sentence encoder for end-to-end co-reference (E2E-CR). Borrowing the idea of an external memory module from Sukhbaatar et al. (2015), an external memory block containing syntactic and semantic information from context sentences is added to the standard LSTM model. With this context memory block, the proposed model is able to encode

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4829–4833
Brussels, Belgium, October 31 - November 4, 2018.

input sentences as a batch, and also calculate the representations of input words by taking both target sentences and context sentences into consideration. Experiments showed that this approach improved the performance of co-reference resolution models.

2 Related Work

2.1 Co-reference Resolution

A popular method of co-reference resolution is mention ranking (Durrett and Klein, 2013). Reading each mention, the model calculates co-reference scores for all antecedent mentions, and picks the mention with the highest positive score to be its co-reference. Many recent works are based on this approach. Durrett and Klein (2013) designed a set of feature templates to improve the mention-ranking model. Peng et al. (2015) proposed a mention-ranking model by jointly learning mention heads and co-references. Clark and Manning (2016a) proposed a reinforcement learning framework for the mention ranking approach. Based on similar ideas but without using parsing features, the authors of Lee et al. (2017) proposed the current state-of-the-art model which uses neural networks to embed mentions and calculate mention and antecedent scores. Lee et al. (2018) applied ELMo embeddings (Peters et al., 2018) to improve within-sentence dependency modeling and word representation learning. Wiseman et al. (2016) and Clark and Manning (2016b) proposed models using global entity-level features.

2.2 Language Representation Learning

Distributed word embeddings has been used as the basic unit of language representation for over a decade (Bengio et al., 2003). Pre-trained word embeddings, for example GloVe (Pennington et al., 2014) and Skip-Gram (Mikolov et al., 2013) are widely used as the input of natural language processing models.

Long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) are widely used for sentence modeling. A single-layer LSTM network was applied in the previous state-of-the-art co-reference model (Lee et al., 2017) to generate word and mention representations. To capture dependency of longer distances, Campos et al. (2017) proposed a recurrent model that outputs hidden states by skipping input tokens.

Recently, memory networks (Sukhbaatar et al., 2015) have been applied in language modeling (Cheng et al., 2016; Tran et al., 2016). Applying an attention mechanism on memory cells, memory networks allow the model to focus on significant words or segments for classification and generation tasks. Previous works have shown that applying memory blocks in LSTMs also improves long-distance dependency extraction (Yogatama et al., 2018).

3 Learning Cross-Sentence dependency

To improve the word representation learning model for better co-reference resolution performance, we propose two word representation models that learn cross-sentence dependency.

3.1 Linear Sentence Linking

Instead of treating the entire input document as separate sentences and encode the sentences as a batch with an LSTM, the most direct way to consider cross-sentence dependency is to initialize LSTM states with the encodings of adjacent sentences. We name this method linear sentence linking (LSL).

In LSL, we encode input sentences with a 2-layer bidirectional LSTM. Give input sentences $[s_1, s_2 \dots s_n]$, the outputs of the first layer are $[[\overrightarrow{s}_1; \overleftarrow{s}_1], [\overrightarrow{s}_2; \overleftarrow{s}_2], \dots [\overrightarrow{s}_n; \overleftarrow{s}_n]]$. In the second LSTM layer, the initial state of the forward LSTM of s_i is initialized as

$$\overrightarrow{S}_i = [\overrightarrow{c}_0^2; [\overrightarrow{s}_{i-1}; \overleftarrow{s}_{i-1}]]$$

while the backward state is initialized as

$$\overleftarrow{S}_i = [\overleftarrow{c}_0^2; [\overrightarrow{s}_{i-1}; \overleftarrow{s}_{i-1}]]$$

where c_0^i stands for the initial cell of the i-th layer, and x stands for the final output of the LSTMs in first layer. We then concatenate the outputs of the forward and backward LSTMs in the second layer as the word representations for co-reference prediction.

3.2 Attentional Sentence Linking

It is difficult for LSTMs to embed enough information about a long sentence into a low-dimensional distributed vector. To collect richer knowledge from neighbor sentences, we propose a long short-term recurrent memory module and an attention mechanism to improve sentence linking.

To describe the architecture of the proposed model, we focus on adjacent input sentences s_{i-1}

and s_i. We present the input embeddings of the j-th word in the i-th sentence with $x_{i,j}$.

3.2.1 Long Short-Term Memory RNNs

To solve the traditional recurrent neural networks, Hochreiter and Schmidhuber (1997) proposed the LSTM architecture. The detail of recurrent state updating in LSTMs $h_t = f_{lstm}(x_t, h_{t-1}, c_{t-1})$ is shown in following equations.

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
c_t &= f_t \odot c_{t-1} + i_t \odot tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}$$

where x_t is the input embedding and h_t is the output representation of the t-th word.

3.2.2 LSTMs with Cross-Sentence Attention

We design an LSTM module with cross-sentence attention for capturing cross-sentence dependency. We name this method attentional sentence linking (ASL). Considering input word $x_{i,t}$ in the i-th sentence and all words from the previous sentence $X_{i-1} = [x_{i-1,1}, x_{i-1,2}, \dots, x_{i-1,m}]$, we regard the matrix X_{i-1} as an external memory module and calculate an attention on its cells, where each cell contains a word embedding.

$$\alpha_j = \frac{e^{c_j}}{\sum_k e^{c_k}} \tag{1}$$

$$c_k = f_c([x_{i,t}; h_{t-1}; x_{i-1,k}]^T) \tag{2}$$

With the attention distribution $\boldsymbol{\alpha}$, we can get a vector summarizing related information from s_{i-1},

$$v_{i-1} = \sum_j \alpha_j \cdot x_{i-1,j} \tag{3}$$

The model decides if it needs to pay more attention on the current input or cross-sentence information with a context gate.

$$g_t = \sigma(f_g([x_{i,t}; h_{t-1}; v_{i-1}]^T)) \tag{4}$$

$$\hat{x}_{i,t} = g_t \cdot x_{i,t} + (1 - g_t) \cdot v_{i-1} \tag{5}$$

$\sigma(\cdot)$ stands for the Sigmoid function. The word representation of the target word is calculated as

$$h_{i,t} = f_{lstm}(\hat{x}_{i,t}, h_{i,t-1}, c_{i,t-1}) \tag{6}$$

where f_{lstm} stands for standard LSTM update described in section 3.2.1.

3.3 Co-reference Prediction

In this work, we apply the mention-ranking end-to-end co-reference resolution (E2E-CR) model proposed by Lee et al. (2017) for co-reference prediction. The word representations applied in E2E-CR model is formed by concatenating pre-trained word embeddings and the outputs of LSTMs. In our work, we represent words by concatenating pre-trained word embeddings and the outputs of LSL- and ASL-LSTMs.

4 Experiments

We train and evaluate our model on the English corpus of the CoNLL-2012 shared task (Pradhan et al., 2012). We implement our model based on the published implementation of the baseline E2E-CR model (Lee et al., 2017) [1]. Our implementation is also available online for reproducing the results reported in this paper [2]. In this section, we first describe our hyperparameter setup, and then show the experimental results of previous work and our proposed models.

4.1 Model and Hyperparameter Setup

In practice, the LSTM modules applied in our model have 200 output units. In ASL, we calculate cross-sentence dependency using a multilayer perceptron with one hidden layer consisting of 150 hidden units. The initial learning rate is set as 0.001 and decays 0.001% every 100 steps. The model is optimized with the Adam algorithm (Kingma and Ba, 2014). We randomly select up to 40 continuous sentences for training if the input is too long. In co-reference prediction, we select 250 candidate antecedents as our baseline model.

4.2 Experiment Results and Discussion

We evaluate our model on the test set of the CoNLL-2012 shared task. The performance of previous work and our model are shown in Table 1. We mainly focus on the average F1 score of MUC, B^3, and CEAF metrics. Comparing with the baseline model that achieved 67.2% F1 score, the ASL model improved the performance by 0.6% and achieved 67.8% average F1. Experiments

---

[1] `https://github.com/kentonl/e2e-coref`

[2] `https://github.com/luohongyin/coatt-coref`

Models	MUC Prec.	MUC Rec.	MUC F1	B^3 Prec.	B^3 Rec.	B^3 F1	Ceafe Prec.	Ceafe Rec.	Ceafe F1	Avg. F1
Wiseman et al. (2016)	77.5	69.8	73.4	66.8	57.0	61.5	62.1	53.9	57.7	64.2
Clark and Manning (2016b)	78.9	69.8	74.0	70.1	57.0	62.9	62.5	55.8	59.0	65.3
Clark and Manning (2016a)	79.2	70.4	74.6	69.9	58.0	63.4	63.5	55.5	59.2	65.7
Lee et al. (2017)	78.4	73.4	75.8	68.6	61.8	65.0	62.7	59.0	60.8	67.2
E2E-CR + LSL	81.0	71.5	76.0	72.6	59.4	65.3	65.0	57.5	61.0	67.4
E2E-CR + ASL	79.2	73.7	**76.4**	69.4	62.1	**65.6**	64.0	58.9	**61.4**	**67.8**

Table 1: Experimental results of previous models and cross-sentence dependency learning models on the CoNLL-2012 shared task.

- I remember receiving **an SMS like this one** last year before it snowed since snowfall would affect road conditions in Beijing to a large extent. - Uh-huh . However, **it** did not give people such a special feeling as it did this time.
- **Reporters** are tired of the usual stand ups. - **They** want to be riding on a train or walking in the rain or something to get attention .
- Planned terrorist bombing that ripped a 20 x 40 - foot hole in the **Navy destroyer USS Cole** in the Yemeni port of Aden. - **The ship** was there for refueling.
- Yemeni authorities claimed they have detained **over 70 people** for questioning. - **These** include some Afghan - Arab volunteers.

Table 2: Examples predictions of the ASL model and the baseline model.

show that the models that consider cross-sentence dependency significantly outperform the baseline model, which encodes each sentence from the input document separately.

Experiments also indicated that the ASL model has better performance than the LSL model, since it summarizes extracts context information with an attention mechanism instead of simply viewing sentence-level embeddings. This gives the model a better ability to model cross-sentence dependency.

Examples for comparing the performance of the ASL model and the baseline are shown in Table 2. Each example contains two continuous sentences with co-references distritubed in different sentences. Underlined spans in bold are target mentions and annotated co-references. Spans in green are ASL predictions, and spans in red are baseline predictions. A prediction on "-" means that no mention is predicted as a co-reference.

Table 2 shows that the baseline model, which does not consider cross-sentence dependency, has difficulty in learning the semantics of pronouns whose co-references are not in the same sentence. The pretrained embeddings of pronouns are not informative enough. In the first example, "it" is not semantically similar with "SMS" in GloVe without any context, and in this case, "it" and "SMS" are in different sentences. As a result, if reading this two sentences separately, it is hard for the encoder to represent "it" with the semantics of "SMS". This difficulty makes the co-reference resolution model either prediction a wrong antecedent mention, or cannot find any co-reference.

However, with ASL, the model learns the semantics of pronouns with an attention to words in other sentences. With the proposed context gate, ASL takes knowledge from context sentences if local inputs are not informative enough. Based on word represents enhanced with cross-sentence dependency, the co-reference scoring model can make better predictions.

5 Conclusion and Future Work

We proposed linear and attentional sentence linking models for learning word representations that captures cross-sentence dependency. Experiments showed that the embeddings learned by proposed models successfully improved the performance of the state-of-the-art co-reference resolution model, indicating that cross-sentence dependency plays an important role in semantic learning in articles and conversations consists of multiple sentences. It worth exploring if our model can improve the performance of other natural language processing applications whose inputs contain multiple sentences, for example, reading comprehension, dialog generation, and sentiment analysis.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. 2017. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Kevin Clark and Christopher D Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*.

Kevin Clark and Christopher D Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. *arXiv preprint arXiv:1606.01323*.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Haoruo Peng, Kai-Wei Chang, and Dan Roth. 2015. A joint framework for coreference resolution and mention head detection. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 12–21.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. *arXiv preprint arXiv:1601.01272*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning global features for coreference resolution. *arXiv preprint arXiv:1604.03035*.

Dani Yogatama, Yishu Miao, Gabor Melis, Wang Ling, Adhiguna Kuncoro, Chris Dyer, and Phil Blunsom. 2018. Memory architectures in recurrent neural network language models. In *International Conference on Learning Representations. https://openreview. net/forum*.

Word Relation Autoencoder for Unseen Hypernym Extraction Using Word Embeddings

Hong-You Chen, Cheng-Syuan Lee and **Keng-Te Liao**
Dept of Computer Science & Information Engineering
National Taiwan University
{b03902128}, {b03902009}, {d05922001}@ntu.edu.tw

Shou-De Lin
Dept of Computer Science & Information Engineering
National Taiwan University
Research Center for Information Technology, Academia Sinica
sdlin@csie.ntu.edu.tw

Abstract

Lexicon relation extraction given distributional representation of words is an important topic in NLP. We observe that the state-of-the-art projection-based methods cannot be generalized to handle unseen hypernyms. We propose to analyze it in the perspective of pollution, that is, the predicted hypernyms are limited to those appeared in training set. We propose a word relation autoencoder (WRAE) model to address the challenge and construct the corresponding indicator to measure the pollution. Experiments on several hypernym-like lexicon datasets show that our model outperforms the competitors significantly.

1 Introduction

This paper discusses the inference of relations between words. For the hypernym *beer IsA drink*, , denoted as *IsA(x, y)*, *beer* is the hyponym x and *drink* serves as the hypernym y. Relation lexicons are precious resource for NLP systems, while constructing the semantic graphs such as WordNet (Fellbaum, 1998) and ConceptNet (Speer and Havasi, 2012) requires expensive human efforts for labeling.

Recently, researchers have started working on extracting word relations based on pre-trained word embedding without the need of an existing corpus, thanks to the success of distributional word representation models such as GloVe (Pennington et al., 2014).

Query	Answer
$beef \rightarrow meat \approx crab \rightarrow ?$	seafood
$tiger \rightarrow zoo \approx dolphin \rightarrow ?$	aquarium
$paint \rightarrow artist \approx book \rightarrow ?$	writer
$japan \rightarrow asia \approx italy \rightarrow ?$	europe

Table 1: Unseen relation extraction examples for IsA, AtLocation, CreatedBy, and PartOf (top row to bottom row) in ConceptNet. The answers are not appeared in training.

Comparing with hypernym classification models (Lenci and Benotto, 2012; Weeds et al., 2014; Levy et al., 2015; Vylomova et al., 2016) that take a pair of entities *(x,y)* as inputs and output a binary decision about the existence of relation, there has been less work focusing on hypernym extraction task. It is a challenging task to automatically extract all possible hypernyms of a given hyponym query, especially the unlabeled ones, from the vocabularies.

Classification-based models are not applicable for this task because the complexity of inference is $\mathcal{O}(V)$, where V is the size of vocabulary that often scales to billions.

Among the existing solutions, projection-based methods (Fu et al., 2014; Yamane et al., 2016; Espinosa-Anke et al., 2016; Ustalov et al., 2017) emphasize on hypernym extraction which intuitively represent a relation as $y - x$ according to the linear structure of word embedding. By directly learning a linear mapping Φ between two words such that $x\Phi = y$, the prediction $\hat{y}$ can be obtained with nearest neighbor search for $x\Phi$ in the word embedding space. Moreover, the potential candidates of y are not required to be seen in advance so that the method can be used to predict unseen hypernym directly.

Fu et al. (2014) further observe the existence of

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4834–4839
Brussels, Belgium, October 31 - November 4, 2018.

cluster structures in relation representation $y - x$ and propose to learn a piecewise linear mapping such that $x\Phi_k = y$ for each cluster C_k. Their experiments show that domain clustering on training offset is very useful for hypernym identification.

However, we observe that each cluster contains very few distinct hypernyms. For instance, about 83% of the clusters contain fewer than 5 hypernyms for ConceptNet-IsA in our experiments. Hypernyms can be seen as the collections of related word pairs, e.g., *IsA(dog, animal)*, *IsA(cat, animal)*, *IsA(horse, animal)*, ... etc. The piecewise projection matrices can hardly learn the inference between hyponyms and hypernyms but only *memorize* some words which serve as the hypernyms in the training data. Inevitably, the state-of-the-art models using piecewise projection learning face generalization problem and fail to predict unseen hypernyms correctly.

We design a novel **Word Relation Autoencoder (WRAE)** framework, which adopts the conditional autoencoder structure ($x \rightarrow r \rightarrow x'$) that encodes hyponyms and reconstructs itself by decoding from $r = y - x$. The weights of encoder are further tied with decoder which is imposed to learn how to separate the hypernym and the hyponym from the relation vectors and extract the hyponym x with the intention to optimize reconstruction loss, thus effectively mitigates the mentioned generalization problem.

We summarize our main contributions as follows: (1) We propose a novel, yet more general scenario for relation extraction to handle unseen hypernyms. (2) We propose an intuitive pollution indicator that allows us to empirically measure whether the model learns the inference between a relation pair or not. (3) We propose a novel Word Relation Autoencoder (WRAE) which can effectively reduce pollution. We conduct thorough experiments to show that our model outperforms the competitors, and can be applied to other hypernym-like relations.

2 Related Work

Fu et al. (2014) first apply projection learning for generalized hypernym extraction by learning a linear transformation from a hyponym word embedding to the corresponding hypernym word vector. They further conduct piecewise projection learning, i.e., learning a projection matrix for each cluster and harvest significant improvements by first applying k-means clustering. They perform training with stochastic gradient descent methods, implying good potential for attaching different regularizers for optimization. Several recent works also follow the schema as the one proposed by Espinosa-Anke et al. (2016) and operate the similar model at the sense level and took advantage of domain clustering to discover hypernyms through domain adaption between different topics. Yamane et al. (2016) focus on improving the performance through better cluster assignments by learning clustering and the projections jointly. Ustalov et al. (2017) propose several regularization terms in addition to the original loss function (Fu et al., 2014) using extra synonym pairs or the asymmetric property of hypernym. Nayak (2015) provides detailed technical studies on piecewise projection models.

Our work differs from all of them, as we emphasize on the setting that all hyponyms and hypernyms in testing vocabulary are not seen in training.

3 Model Formulation

3.1 Piecewise Projection

Piecewise projection learning (Fu et al., 2014) serves as our baseline. The objective is to learn a relation transform from x to y on training pairs (x, y). Piecewise projection matrix Φ_k is learned separately for each cluster, after applying k-means clustering on the offset of training using $y - x$ between each pair.

$$\min_{\Phi_k} \frac{1}{|C_k|} \sum_{(x,y)\in C_k} \|x\Phi_k - y\|_2^2 , \qquad (1)$$

where C_k represents the size of the k^{th} cluster. In addition, we also examine a simple solution of L2-penalized projection learning model which imposes a L2 constraint on Φ in Equation 1, i.e., $\alpha \|\Phi_k\|_2^2$.

3.2 Word Relation Autoencoder (WRAE)

Our model takes the form of an autoencoder. As shown in Equation 2,

$$\min_{\Phi_k} \frac{1}{|C_k|} \sum_{(x,y)\in C_k} \|x - x\Phi_k\Phi_k^*\|_2^2 , \qquad (2)$$

where $x\Phi_k = y - x$. Here we adopt the simplifying trick (Kodirov et al., 2017a) to tie with the constraint (Ranzato et al., 2008) $\Phi^* = \Phi^T$. Note that the L2-norm regularization term is not necessary for WRAE to avoid overfitting since the constraint of $\Phi^* = \Phi^T$ guarantees $\|\Phi\|_2^2$ cannot be large otherwise the reconstruction loss will be bad. Also, the learning process is more efficient.

To release the constraint of $x\Phi_k = y - x$, the objective can be further split into two terms:

$$\min_{\Phi_k} \frac{1}{|C_k|} \sum_{(x,y)\in C_k} (\|x\Phi_k - (y-x)\|_2^2 + \lambda \left\|(y-x)\Phi_k^T - x\right\|_2^2), \tag{3}$$

where λ is a weighting constant.

We find that learning relation mapping from $x \to (y-x)$ instead of $x \to y$ effectively mitigates the pollution problem (Lazaridou et al., 2015). A prediction is said to be polluted if the nearest neighbor of predicted $\hat{y}$ matches a hypernym appeared in training set. The operation fundamentally solves the cause of pollution since each pair of input and output becomes $(x, y-x)$ instead of (x, y). Unlike projecting to a small number of target y, the target $y-x$ obviously differs from pair to pair thus avoiding simply overfitting the lexicons.

Conceptually, WRAE learns to extract the hyponym x from the relation vectors $r = y - x$ to optimize on the reconstruction loss. By encouraging the projection to learn the relationship between a word pair, WRAE effectively mitigates the mentioned generalized problem.

Our model is related to Semantic Autoencoder (SAE) (Kodirov et al., 2017b). With the latent relation directly associates with input x, WRAE can be regarded as a special conditional SAE where the condition is the input itself and is incorporated into the middle layer.

Relation	#Pair	#Head	#Tail
IsA	78073	21714	62455
AtLocation	39916	10100	11311
PartOf	14231	9784	5519
CreatedBy	503	385	414

Table 2: Relations from ConceptNet. For a relation pair $x \to y$, x is the *head* and y is the *tail*.

4 Experiments

4.1 Setup

Different from the experimental setup in previous works (Fu et al., 2014; Ustalov et al., 2017; Yamane et al., 2016) that do not assume the candidate hypernyms are unseen, in our experiments the vocabulary sets for training and testing are completely disjoint, i.e., all vocabularies in testing are not seen in training at all.

To further examine the generality of our model, we collect several hypernym-like relations listed in Table 2 from ConceptNet semantic graph. Considering the property of these relations, we treat the head and tail words of a pair as the x and y for our models similar to hyponym and hypernym, respectively. Examples are in Table 1.

We split the datasets with ratios 0.7, 0.2, and 0.1 for training, testing, and validation, respectively. For all results, we report the mean of 30 random splits. We test two different settings, one uses k-means clustering and one does not ($k = 1$). We tune the number of cluster k unsupervisedly with the Silhouette score (Rousseeuw, 1987) on validation. The projection matrices are optimized with the Adam method (Kingma and Ba, 2014) with learning rate = $1e^{-3}$. We adopt the GloVe (Pennington et al., 2014) 300d pre-trained word embeddings[1] which are trained on 6B token corpus (Wikipedia 2014 + Gigaword 5) with 400,000 words.

4.2 Evaluation Metrics

Hit Rate

To evaluate the precision of returned hypernyms, we follow Ustalov et al. (2017) and Kodirov et al. (2017a) using the hit rate measure (Frome et al., 2013). We also adopt area under curve (AUC) measure which computes the averaged area under the $l - 1$ trapezoids of hit@l to take the ranks of ground truth into consideration:

$$AUC_l = \frac{1}{2(l-1)} \sum_{i=1}^{l-1} (hit@i + hit@(i+1)), \tag{4}$$

Soft Pollution

To evaluate the degree of pollution of the extracted hypernym, we adopt a metric similar to Lazaridou et al. (2015). A prediction is said to be polluted if the nearest neighbor of predicted $\hat{y}$ matches a hypernym appears in the training set, noted as a binary function $pol_1(\hat{y})$.

However, it is possible that ground truth unseen hypernyms are be very close to some seen hypernyms in Y_{train} in real cases. We take ground truths

---

[1] https://nlp.stanford.edu/projects/glove/

Relation	**Hit Rate**			**Pollution**		**Hit Rate**			**Pollution**	
Model	hit_{10}	hit_{30}	AUC_{30}	pol_1	pol_{30}^{soft}	hit_{10}	hit_{30}	AUC_{30}	pol_1	pol_{30}^{soft}
IsA*				**k=1**				**k=25**		
Proj.	.110	.157	.107	.715	.223	.090	.137	.087	.721	.232
Proj.+L2	.112	.172	.108	.712	.223	.090	.139	.088	.719	.231
WRAE-Y	.120	.190	.110	.691	.220	.096	.146	.089	.720	.230
WRAE†	**.124**	**.194**	**.122**	**.602**	**.191**	**.164**	**.249**	**.160**	**.124**	**.034**
AtLocation				**k=1**				**k=25**		
Proj.	.075	.149	.103	.707	.115	.131	.220	.149	.782	.094
Proj.+L2	.083	.161	.110	.693	.113	.129	.222	.149	.796	.096
WRAE-Y	.086	.166	.127	.699	.112	.129	.236	.153	.782	.094
WRAE†	**.122**	**.224**	**.152**	**.409**	**.063**	**.148**	**.261**	**.174**	**.191**	**.024**
CreatedBy				**k=1**				**k=10**		
Proj.	.016	.040	.026	.625	.226	.054	.103	.059	.819	.591
Proj.+L2	.016	.044	.030	.624	.227	.050	.099	.048	.818	.592
WRAE-Y	.021	.060	.031	.586	.151	.057	.103	.052	.819	.591
WRAE†	**.070**	**.131**	**.095**	**.191**	**.067**	**.142**	**.243**	**.156**	**.071**	**.048**
PartOf				**k=1**				**k=45**		
Proj.	.335	.434	.341	.660	.187	.260	.405	.294	.796	.224
Proj.+L2	.340	.439	.344	.660	.188	.263	.407	.292	.793	.224
WRAE-Y	.342	.449	.350	.644	.182	.267	.411	.297	.793	.222
WRAE†	**.355**	**.464**	**.370**	**.546**	**.149**	**.437**	**.539**	**.440**	**.117**	**.038**

Table 3: Performance on ConceptNet relation dataset. †: all results pass the hypothesis test against the other models with $p < 0.01$. *: for IsA, we report hit@5 and hit@10. For hit rates, the higher the better. For pollution, the lower the better.

into consideration:

$$pol_l^{soft}(\hat{y}, y) = \rho \cdot pol_1(\hat{y}), y \in Y_{test},$$
$$\rho = \begin{cases} 1, & if \quad NN_l(y) \cap Y_{train} = \phi, \\ 2^{\frac{n-1}{l-1}} - 1, & otherwise, \end{cases} \tag{5}$$

where n is for the top n nearest neighbors (from 1 to l) of y that appears in Y_{train} and ρ is a factor term exponentially decreases from 1 to 0 along with the increase of n therefore provides a smoother estimation. With pollution indications, one can understand to what degree the model suffers from overfitting on the seen examples. ϕ is the empty set. Note that it is reasonable to set l equal for both hit rate and soft pollution.

4.3 Results: Unseen General Hypernym-Like Relation Extraction

We report two sets of results for all models, one with clustering and one without (k = 1). As shown in Table 3, WRAE outperforms the competitors significantly with and without clustering. The naive application of Equation 2 which set $x\Phi_k = y$, denoted as *WRAE-Y*, consistently ranks second. The $y - x$ operation is crucial to avoid pollution thus guarantees the generalization power of the mapping. Apply simple L2-norm regularization Equation 1 for *Proj.*, denoted as *Proj.+L2*, only slightly improves the performance. The results in Table 3 supports our hypothesis that Proj. models deteriorate significantly for larger k, due to lack of training examples for hypernyms in each cluster. We prove that WRAE is effective against pollution. The role of regularizer is important for decoders to optimize towards better objective.

The negative effects derived from pollution impact accuracy. We observe severe pollution problem in simple projection learning. Take *IsA* as example, in $k = 1$ group the pol_1 is about 71% for Proj., which implies about two of out of three returned predictions are data points from training data. Our WRAE reduces the pollution pol_1 to 60% and 12% after clustering. The improvement on accuracy supports that pollution indication reflects the inherent overfitting problem. In general, results are consistent with our claims that pollution can be viewed as valid negative indicators.

Across the board, the performance should benefit from domain clustering if pollution is handled properly as the experiments showed.

5 Conclusion

We present an unseen hypernym extraction framework and analyze the pollution problem with this setup. Consequently we argue that only by using unseen candidates in evaluation can truly test whether the model learns the true relation representation, instead of being polluted by the seen training examples. Future work includes relation discovery, which is to identify new relations besides hypernyms in an unsupervised manner.

Acknowledgments

This material is based upon work supported by Taiwan Ministry of Science and Technology (MOST) under grant number 106-2218-E-002 - 042 (machine learning related).

References

Luis Espinosa-Anke, José Camacho Collados, Claudio Delli Bovi, and Horacio Saggion. 2016. Supervised Distributional Hypernym Discovery via Domain Adaptation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 424–435.

C Fellbaum. 1998. *WordNet: An Electronic Lexical Database: Bradford Book*. Cambridge, MA: MIT Press.

Andrea Frome, Gs Corrado, and Jonathon Shlens. 2013. Devise: A deep visual-semantic embedding model. *Advances in neural information processing systems*, pages 1–11.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1199–1209. Baltimore, Maryland.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs.LG]*.

Elyor Kodirov, Tao Xiang, and Shaogang Gong. 2017a. Semantic Autoencoder for Zero-Shot Learning. In *Computer Vision and Pattern Recognition*. Computer Vision Foundation.

Elyor Kodirov, Tao Xiang, and Shaogang Gong. 2017b. Semantic autoencoder for zero-shot learning.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and Pollution: Delving into Cross-Space Mapping for Zero-Shot Learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 270–280. Association for Computational Linguistics.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 75–79, Stroudsburg, PA, USA. Association for Computational Linguistics.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976. Association for Computational Linguistics.

Neha Nayak. 2015. Learning hypernymy over word embeddings. Technical report, Stanford Univ.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543.

Marc'aurelio Ranzato, Y lan Boureau, and Yann Lecun. 2008. Sparse Feature Learning for Deep Belief Networks. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pages 1185–1192. Curran Associates, Inc.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Robert Speer and Catherine Havasi. 2012. Representing General Relational Knowledge in ConceptNet 5. In *International conference on language resources and evaluation (LREC)*, pages 3679–3686.

Dmitry Ustalov, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. 2017. Negative Sampling Improves Hypernymy Extraction Based on Projection Learning. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 543–550, Valencia, Spain. Association for Computational Linguistics.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1671–1682, Berlin, Germany. Association for Computational Linguistics.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to Distinguish Hypernyms and Co-Hyponyms. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 2249–2259, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Josuke Yamane, Tomoya Takatani, Hitoshi Yamada, Makoto Miwa, and Yutaka Sasaki. 2016. Distributional hypernym generation by jointly learning clusters and projections. In *Proceedings of 26th International Conference on Computational Linguistics (COLING 2016)*, pages 1871–1879, Osaka, Japan. The COLING 2016 Organizing Committee.

Refining Pretrained Word Embeddings Using Layer-wise Relevance Propagation

Akira Utsumi
Department of Informatics & Artificial Intelligence eXploration Research Center
The University of Electro-Communications, Chofu, Tokyo 182-8585, Japan
`utsumi@uec.ac.jp`

Abstract

In this paper, we propose a simple method for refining pretrained word embeddings using layer-wise relevance propagation. Given a target semantic representation one would like word vectors to reflect, our method first trains the mapping between the original word vectors and the target representation using a neural network. Estimated target values are then propagated backward toward word vectors, and a relevance score is computed for each dimension of word vectors. Finally, the relevance score vectors are used to refine the original word vectors so that they are projected into the subspace that reflects the information relevant to the target representation. The evaluation experiment using binary classification of word pairs demonstrates that the refined vectors by our method achieve the higher performance than the original vectors.

1 Introduction

The recent success of neural NLP is partially but largely due to the development of word embedding techniques (Goldberg, 2017). Although a considerable number of studies have been made on training word embeddings from distributional information of language (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017; Nickel and Kiela, 2017), one recent research trend is to refine or fine-tune pretrained word embeddings. One promising approach is the use of other information such as multimodal information (Bruni et al., 2014; Kiela et al., 2014; Kiela and Clark, 2015; Kiela et al., 2015a; Silberer et al., 2017) and language resources (Faruqui et al., 2015; Faruqui and Dyer, 2015; Kiela et al., 2015b; Rothe and Schütze, 2017; Yu and Dredze, 2014). Other refinement methods include task-specific embeddings (Bolukbasi et al., 2016; Yu et al., 2017) and the selective use of multiple embeddings (Bollegala et al., 2017; Kiela et al., 2018).

In this paper, we propose a different approach to refining pretrained word embeddings so that word vectors reflect the information relevant for a specific knowledge. Our method utilizes layer-wise relevance propagation (Bach et al., 2015; Samek et al., 2017), which has been proposed as a general framework for decomposing predictions of modern AI systems, in particular deep learning systems. The basic idea of layer-wise relevance propagation is to quantitatively measure the contribution of each fragment of an input vector (e.g., a single pixel of an image) to the prediction as a relevance score. Using relevance scores, our method projects word vectors into the subspace that better reflects the target knowledge. The assumption underlying our approach is that the information for any given target knowledge is contained in pretrained word embeddings. Our method attempts to make the best use of the information contained in word vectors by estimating the importance in reflecting a target knowledge.

To the best of our knowledge, this paper is the first to employ the technique of layer-wise relevance propagation for refining word embeddings. Our method can be applied to word vectors $\boldsymbol{x}$ trained by any word embedding method. This implies that our method does not compete with other refinement methods, but they are complementary; it can be used for word vectors refined by other methods. In addition, our method can refine word vectors for any target knowledge $\boldsymbol{y}$, from a single binary value to a structured representation, as long as a function $\boldsymbol{y} = f(\boldsymbol{x})$ can be learned.

2 Method for Refining Word Vectors

Our method comprises the following three steps: (1) it trains a prediction function from a pretrained word vector to a target representation; (2) computes a relevance score for each dimension of the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4840–4846
Brussels, Belgium, October 31 - November 4, 2018.

word vector; and (3) projects word vectors into the subspace using the relevance scores. In this section, these three steps are explained in detail.

2.1 Training the Prediction Function

Given pairs of an input word vector $\boldsymbol{x}^{(i)}$ to be refined and a target knowledge representation $\boldsymbol{y}^{(i)}$ for a word $w^{(i)}$, the proposed method trains a function $\boldsymbol{y}^{(i)} = f(\boldsymbol{x}^{(i)})$. In this paper, we use a neural network as a learning method, but other learning methods such as linear transformation and SVM can also be used. Note that a scalar value or a class label can be used as a target representation $\boldsymbol{y}^{(i)}$.

2.2 Computing Relevance Scores

This step derives an explanation of the prediction in terms of input variables, namely the importance of each dimension of a word vector $\boldsymbol{x}^{(i)}$ for the prediction $\hat{\boldsymbol{y}}^{(i)} = f(\boldsymbol{x}^{(i)})$. In layer-wise relevance propagation, the score of the correct prediction $\hat{y}_j^{(i)}$ is redistributed backward using relevance propagation rules. By repeatedly applying propagation rules, it assigns a relevance score $r_k^{(i,j)}$ to each dimension $x_k^{(i)}$ of a word vector $\boldsymbol{x}^{(i)}$. As a result, a relevance score vector $\boldsymbol{r}^{(i,j)}$ is obtained for each word vector $\boldsymbol{x}^{(i)}$ and target dimension $y_j^{(i)}$.

Among a number of propagation rules (Bach et al., 2015), we use the "alpha-beta" rule for multilayer neural networks. The relevance score $R_i^{(l)}$ of the i-th unit $u_i^{(l)}$ in the l-th layer is a function of upper-layer relevances $R_j^{(l+1)}$ defined by:

$$R_{i\leftarrow j}^{(l,l+1)} = R_j^{(l+1)} \cdot \left(\alpha \frac{z_{ij}^{+}}{\sum_i z_{ij}^{+}} + \beta \frac{z_{ij}^{-}}{\sum_i z_{ij}^{-}} \right) \quad (1)$$

$$R_i^{(l)} = \sum_j R_{i\leftarrow j}^{(l,l+1)} \quad (2)$$

$$z_{ij}^{(l,l+1)} = x_i^{(l)} w_{ij}^{(l,l+1)} \quad (3)$$

where $x_i^{(l)}$ is an activation of the unit $u_i^{(l)}$, $w_{ij}^{(l,l+1)}$ is a weight connecting $u_i^{(l)}$ to $u_j^{(l+1)}$, and z_{ij}^{+} and z_{ij}^{-} denote the positive and negative part of $z_{ij}^{(l,l+1)}$. As a result, relevance scores $r_k^{(i,j)}$ of the word vector $\boldsymbol{x}^{(i)}$ and the target dimension $y_j^{(i)}$ are obtained as relevance scores $R_k^{(1)}$ of the input layer. The parameters α and β denote the importance of positive and negative evidence for predicting a target representations and should be chosen such that $\alpha + \beta = 1$. In this paper, we assume that positive and negative evidence equally contributes to the prediction and thus set $\alpha = \beta = 0.5$.

2.3 Projecting Word Vectors into a Subspace

The basic idea of projection is that n-dimensional word vectors are projected into m-dimensional vectors whose relevance scores are more than or equal to a threshold θ_R.

First, for a target dimension j of $\boldsymbol{y}$, relevance score vectors are averaged over words relevant to the target dimension as follows:

$$\overline{\boldsymbol{r}}^{(j)} = g_2\left(\frac{\sum_{w_i \in V_j} g_1(\boldsymbol{r}^{(i,j)})}{|V_j|} \right) \quad (4)$$

$$\{g_1(\boldsymbol{x})\}_i = \begin{cases} x_i & (x_i \geq \theta_{R_1}) \\ 0 & (\text{otherwise}) \end{cases}$$

$$\{g_2(\boldsymbol{x})\}_i = \begin{cases} \frac{x_i}{\max_i x_i} & (\frac{x_i}{\max_i x_i} \geq \theta_{R_2}) \\ 0 & (\text{otherwise}) \end{cases}$$

where V_j is a set of words $w^{(i)}$ such that $\hat{y}_j^{(i)} \geq \theta_T$. The functions g_1 and g_2 are used for downplaying irrelevant dimensions. For example, the target knowledge is the property of *Visually dark* and $V_{visually\_dark}$ is {*chocolate, crow, night*}. By averaging relevance score vectors of these words, we obtain the mean relevance vector $\overline{\boldsymbol{r}}^{(visually\_dark)}$ that represents the importance of word vector dimension in predicting whether a given word has the property of *Visually dark*.

Finally, using the mean relevance vector $\overline{\boldsymbol{r}}^{(j)}$, word vectors $\boldsymbol{x}_i$ is transformed into vectors $\boldsymbol{z}_i^{(j)}$ of a subspace for the target dimension. This is achieved by weighting $\boldsymbol{x}_i$ by component-wise multiplication of $\boldsymbol{x}_i$ and $\overline{\boldsymbol{r}}^{(j)}$ and removing the dimensions of zero relevance. Formally, the projection is defined by the n by m projection matrix $\boldsymbol{T}^{(j)}$ as follows:

$$\boldsymbol{z}_i^{(j)} = \boldsymbol{x}_i \boldsymbol{T}^{(j)} \quad (5)$$

$$T_{ik}^{(j)} = \begin{cases} \overline{r}_i^{(j)} & (\overline{r}_i^{(j)} > 0 \text{ and it is the } k\text{-th} \\ & \ \text{nonzero dimension of } \overline{\boldsymbol{r}}^{(j)}) \\ 0 & (\text{otherwise}) \end{cases} \quad (6)$$

3 Evaluation Experiment

In order to justify the effectiveness of the proposed method, we conducted an evaluation experiment using binary classification of word pairs.

Corpus: All word vectors were trained on the Corpus of Contemporary American English

Domain	Properties
Vision	Vision, Bright, Dark, Color, Pattern, Large, Small, Motion, Biomotion, Fast, Slow, Shape, Complexity, Face, Body
Somatic	Touch, Temperature, Texture, Weight, Pain
Audition	Audition, Loud, Low, High, Sound, Music, Speech
Gustation	Taste
Olfaction	Smell
Motor	Head, UpperLimb, LowerLimb, Practice
Spatial	Landmark, Path, Scene, Near, Toward, Away, Number
Temporal	Time, Duration, Long, Short
Causal	Caused, Consequential
Social	Social, Human, Communication, Self
Cognition	Cognition
Emotion	Benefit, Harm, Pleasant, Unpleasant, Happy, Sad, Angry, Disgusted, Fearful, Surprised
Drive	Drive, Needs
Attention	Attention, Arousal

Table 1: 65 properties in Binder et al.'s (2016) dataset

(COCA), which includes 0.56G word tokens. Words that occurred less than 30 times in the corpus were ignored, resulting in the vocabulary of 108,230 words. Three context windows of size 3, 5, and 10 were used for training.

Word embedding: We used two representative models, namely skip-gram with negative sampling (SGNS) (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). We trained 100-, 200- and 300-dimensional word vectors from the corpus.

Target knowledge representation: We used Binder et al.'s (2016) brain-based semantic vectors of 535 words as a target representation. [1] This representation comprises 65 properties in Table 1, which are based entirely on functional divisions in the human brain. Each word is represented as a 65-dimensional vector and each dimension corresponds to one of these properties. Each value of the brain-based vectors represents the salience of the corresponding property, which is calculated as a mean salience rating on a 7-point scale ranging from 0 to 6. Because these properties are based on not only perceptual properties but also a variety of other properties such as affective, social, and cognitive ones, this dataset is suitable for evaluation.

Refining word vectors: The prediction function f was trained using a three-layer neural network comprising an input layer for n-dimensional word vectors, one hidden layer with $n/2$ sigmoid units, and a linear output layer. The parameters θ_T, θ_{R_1} and θ_{R_2} for projection were estimated using 10-fold cross-validation and grid search. [2]

Bright, Dark, Color, Pattern, Large, Small, Motion, Fast, Slow, Shape, Temperature, Texture, Weight, Loud, Sound, Taste, Smell, Fearful

Table 2: 18 properties in CSLB dataset

Task: We used a binary classification task of judging whether a pair of words is similar or not with respect to each property of Table 1. For example, *night* and *chocolate* should be judged as similar with respect to the property of *Dark*, while *night* and *ice* should be judged as dissimilar with respect to that property. For each property, we chose 10 words with the highest salience and 10 words with the lowest salience from the vocabulary of brain-based vectors, and generated 45 high-salience word pairs and 100 pairs of high-salience and low-salience words. Note that we did not consider low-score word pairs because it does not make sense to ask whether words (e.g., *peace* and *wit*) that do not have a property (e.g., *Dark*) are similar with respect to that property.

To confirm the generality of our method, we also generated another evaluation dataset for untrained words (i.e., words not included in Binder et al.'s vocabulary) using CSLB concept property norms of 638 words (Devereux et al., 2014). [3] After removing words contained in Binder et al.'s vocabulary, we chose properties that were closely related to Binder et al.'s properties and possessed by at least 10 words. As a result, the generated dataset contained 18 properties listed in Table 2, because the property norm mainly includes perceptual and functional properties.

Binary classification was carried out by computing cosine similarity between vectors of paired words and classifying the n highest pairs into similar pairs. Hence, the classification performance was measured by average precision.

4 Results

Table 3 shows mean average precisions across 65 properties for the original word embeddings (Orig) and the refined embeddings by our method (Refn). The asterisk indicates that the mean average precision of the refined vectors is signifi-

---

[1] http://www.neuro.mcw.edu/semanticrepresentations.html

[2] The range in grid search was $[3.0, 4.5]$ with a step size of 0.1 for θ_T, $[0.0, 0.02n]$ with a step size of $0.001n$ for θ_{R_1} of n hundred word vector dimension, and $[0.0, 0.7]$ with a step size of 0.05 for θ_{R_2}.

[3] https://cslb.psychol.cam.ac.uk/propnorms

		SGNS		GloVe	
`win`	`dim`	Orig	Refn	Orig	Refn
10	300	75.3	**78.6***	67.4	**70.4***
10	200	75.9	**79.3***	67.8	**73.7***
10	100	76.1	**77.0***	68.7	**71.6***
5	300	75.4	**78.8***	67.7	**71.8***
5	200	75.6	**79.4***	68.0	**73.9***
5	100	77.2	**78.3**	68.9	**70.1**
3	300	75.5	**79.3***	67.6	**71.5***
3	200	76.5	**77.9***	68.2	**70.8***
3	100	77.4	**79.0***	68.4	**71.2***

Table 3: Mean average precision for Binder et al.'s (2016) dataset

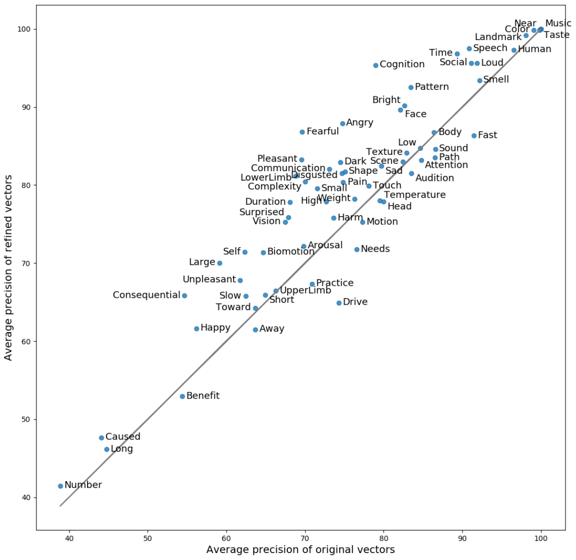

Figure 1: A scatterplot of average precision of the original versus refined vectors for 65 properties in the case of SGNS with `win`$=5$ and `dim`$=200$. The diagonal reference line $y = x$ indicates that the original and refined vectors have equal precision.

cantly higher than that of the original vectors by Wilcoxon signed-rank test ($p < .05$). For all word embeddings, the refined vectors achieved higher mean average precision than the original ones. Furthermore, in almost all cases, the improvement is statistically significant. This result demonstrates that the proposed method is successful in refining word embeddings so that vector similarity better reflects the target knowledge.

Figure 1 depicts the difference of average precision between the original word vectors and the refined vectors for each target property. Most of the properties are plotted above the diagonal reference line, indicating that these properties are better represented by the refined vectors. Note that properties plotted below the diagonal line, for which refined word vectors yielded lower precision than the original vectors, are sensorimotor or spatiotemporal properties. This result is consistent with Utsumi's (2018) finding that these kinds of knowledge are less likely to be encoded in word vectors.

		SGNS		GloVe	
`win`	`dim`	Orig	Refn	Orig	Refn
10	300	57.9	**60.4***	56.8	**58.5**
10	200	58.0	**59.3**	56.3	**56.6**
10	100	**58.8**	58.3	55.9	**56.0**
5	300	58.8	**61.1***	56.4	**59.3***
5	200	58.5	**62.6***	55.6	**56.8**
5	100	58.9	**60.9**	**56.5**	55.5
3	300	58.5	**58.8**	55.2	**55.6**
3	200	58.9	**59.3**	**54.5**	54.1
3	100	**59.3**	58.8	53.5	**54.4**

Table 4: Mean average precision for CSLB property norm dataset

Table 4 shows the result of binary classification for CSLB property norm dataset. In most cases, the refined vectors of untrained words also yielded better performance than the original vectors. In some cases, however, refinement did not improve the performance. One of the reasons for this failure would be that a small set of vocabulary words in Binder et al.'s (2016) dataset is not enough for the subspace to generalize to untrained words.

To confirm whether the projected subspace better reflects the target knowledge than the original space, we visualize both spaces using MDS in Figure 2. Although all 535 words are embedded into the two-dimensional space, Figure 2 only shows words used in binary classification task, namely words with the 10 highest salience (denoted by red dots) and 20 lowest salience for a given property. As shown in Figure 2, our method refines the vectors of salient words to be more similar in the subspace, while preserving the other similarity of words.

5 Related Work

Prior work on word embedding refinement can be classified into general purpose refinement and specific target refinement. Many existing studies have attempted to refine word vectors to improve the performance of general-purpose similarity computation. These studies generally re-

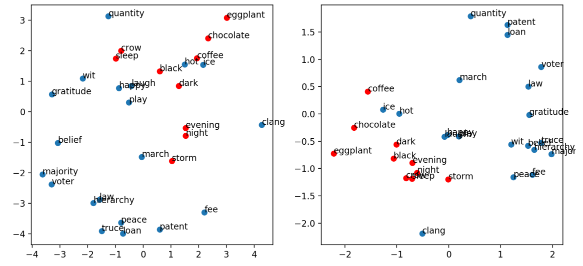

(a) Property *Dark*

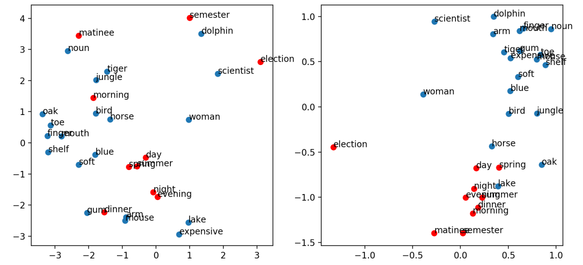

(b) Property *Time*

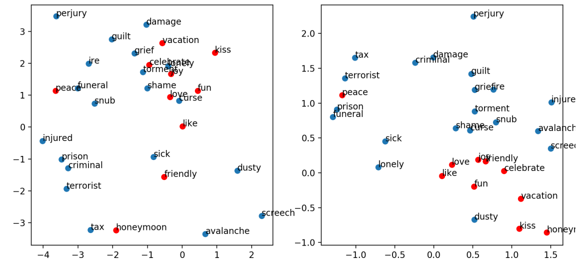

(c) Property *Pleasant*

Figure 2: Two-dimensional MDS visualization of the original space (trained by SGNS with `win`$=5$ and `dim`$=200$) and the projected subspace ($\theta_T = 3.0$, $\theta_{R_1} = 0.026$ and $\theta_{R_2} = 0.10$). Left: Original space, Right: Projected subspace

fine word vectors by solving an optimization problem whose objective function reflects the similarity obtained by language resources, such as WordNet (Faruqui et al., 2015; Yu and Dredze, 2014; Rothe and Schütze, 2017), Freebase (Rothe and Schütze, 2017), Paraphrase Database (Faruqui et al., 2015; Yu and Dredze, 2014), free association norm (Kiela et al., 2015b), and dictionary (Wang et al., 2015). Our method differs from them in that it is proposed for specific target refinement. In other words, the refined vectors by general purpose refinement method can be further refined to extract a specific knowledge by our method.

Most prior studies for specific purpose refinement propose a method specialized for a specific task such as sentiment analysis (Labutov and Lipson, 2013; Tang et al., 2016; Yu et al., 2017) and lexical entailment (Mrkšić et al., 2016; Vulić and Mrkšić, 2018). On the other hand, our method refines word vectors for a specific knowledge or task, but it is not specialized for a knowledge or task.

Rothe et al. (2016) and Rothe and Schütze (2016) are conceptually similar to our approach; their method refines word vectors for a specific knowledge but it is not specialized for a certain task. The merit of our method is that any types of representation can be used as a target, while their method is limited to binary labels. Furthermore, while their method learns an orthogonal transformation of pretrained word vectors by directly optimizing the objective function, our method can project word vectors to a subspace independent of training method for a prediction function.

6 Conclusion

In this paper, we propose a method for refining pretrained word vectors using layer-wise relevance propagation. We demonstrated that the proposed method can refine word vectors so that they better reflect the target knowledge. One of our motivation is to make embeddings more interpretable and useful. In other studies (Utsumi, 2015, 2018), we have analyzed the internal knowledge encoded in text-based word embeddings, while this study is the first step toward a general method for utilizing the internal knowledge of word embeddings.

In future work, we have to modify the refinement method by relevance propagation to be more effective by exploring the mechanism of how the internal knowledge of word vectors is extracted by multilayer neural networks and examining the effectiveness of other relevance propagation methods. It would also be vital for future work to explore efficient combinations with other refinement methods using language resources.

Acknowledgments

This research was supported by JSPS KAKENHI Grant Numbers JP15H02713 and SCAT Research Grant.

References

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140.

Jeffrey R. Binder, Lisa L. Conant, Colin J. Humphries, Leonardo Fernandino, Stephen B. Simons, Mario

Aguilar, and Rutvik H. Desai. 2016. Toward a brain-based componential semantic representation. *Cognitive Neuropsychology*, 33(3–4):130–174.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Danushka Bollegala, Kohei Hayashi, and Ken-ichi Kawarabayashi. 2017. Learning linear transformations between counting-based and prediction-based word embeddings. *PLoS ONE*, 12(9):e0184544.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems 29*, pages 4349–4357.

Elia Bruni, Nam K. Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Jounral of Artificial Intelligence Research*, 49:1–47.

Barry J. Devereux, Lorraine K. Tyler, Jeroen Geertzen, and Billi Randall. 2014. The Centre for Speech, Language and the Brain (CSLB) concept property norms. *Behavior Research Methods*, 46:1119–1127.

Manaal Faruqui, Jesse Dodge, Kumar Sujay Jauhar, Chris Dyer, Eduard Hovy, and A. Noah Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 464–469.

Yoab Goldberg. 2017. *Neural Network Methods for Natural Language Processing*. Morgan & Claypool Publishers.

Douwe Kiela, Luana Bulat, and Stephen Clark. 2015a. Grounding semantics in olfactory perception. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 231–236.

Douwe Kiela and Stephen Clark. 2015. Multi- and cross-modal semantics beyond vision: Grounding in auditory perception. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2461–2470.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015b. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 835–841.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Context-attentive embeddings for improved sentence representations. *arXiv:1804.07983 [cs.CL]*.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 489–493.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at the International Conference on Learning Representation (ICLR)*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, BlaiseThomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148.

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6338–6347.

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

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777.

Sascha Rothe and Hinrich Schütze. 2016. Word embedding calculus in meaningful ultradense subspaces. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 512–517.

Sascha Rothe and Hinrich Schütze. 2017. Autoextend: Combining word embeddings with semantic resources. *Computational Linguistics*, 43(3):593–617.

Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv:1708.08296 [cs.AI]*.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2017. Visually grounded meaning representations. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 39(11):2284–2297.

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28:496–509.

Akira Utsumi. 2015. A complex network approach to distributional semantic models. *PLoS ONE*, 10(8):e0136277.

Akira Utsumi. 2018. A neurobiologically motivated analysis of distributional semantic models. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society (CogSci2018)*, pages 1147–1152.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising word vectors for lexical entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1134–1145.

Tong Wang, Abdel-rahman Mohamed, and Graeme Hirst. 2015. Learning lexical embeddings with syntactic and lexicographic knowledge. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 458–463.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 534–539.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 545–550.

Learning Gender-Neutral Word Embeddings

Jieyu Zhao **Yichao Zhou** **Zeyu Li** **Wei Wang** **Kai-Wei Chang**
University of California, Los Angeles
{jyzhao, yz, zyli, weiwang, kwchang}@cs.ucla.edu

Abstract

Word embedding models have become a fundamental component in a wide range of Natural Language Processing (NLP) applications. However, embeddings trained on human-generated corpora have been demonstrated to inherit strong gender stereotypes that reflect social constructs. To address this concern, in this paper, we propose a novel training procedure for learning gender-neutral word embeddings. Our approach aims to preserve gender information in certain dimensions of word vectors while compelling other dimensions to be free of gender influence. Based on the proposed method, we generate a Gender-Neutral variant of GloVe (GN-GloVe). Quantitative and qualitative experiments demonstrate that GN-GloVe successfully isolates gender information without sacrificing the functionality of the embedding model.

1 Introduction

Word embedding models have been designed for representing the meaning of words in a vector space. These models have become a fundamental NLP technique and have been widely used in various applications. However, prior studies show that such models learned from human-generated corpora are often prone to exhibit social biases, such as gender stereotypes (Bolukbasi et al., 2016; Caliskan et al., 2017). For example, the word "programmer" is neutral to gender by its definition, but an embedding model trained on a news corpus associates "programmer" closer with "male" than "female".

Such a bias substantially affects downstream applications. Zhao et al. (2018) show that a coreference resolution system is sexist due to the word embedding component used in the system. This concerns the practitioners who use the embedding model to build gender-sensitive applications such as a resume filtering system or a job recommendation system as the automated system may discriminate candidates based on their gender, as reflected by their name. Besides, biased embeddings may implicitly affect downstream applications used in our daily lives. For example, when searching for "computer scientist" using a search engine, as this phrase is closer to male names than female names in the embedding space, a search algorithm using an embedding model in the backbone tends to rank male scientists higher than females', hindering women from being recognized and further exacerbating the gender inequality in the community.

To alleviate gender stereotype in word embeddings, Bolukbasi et al. (2016) propose a post-processing method that projects gender-neutral words to a subspace which is perpendicular to the gender dimension defined by a set of gender-definition words.[1] However, their approach has two limitations. First, the method is essentially a pipeline approach and requires the gender-neutral words to be identified by a classifier before employing the projection. If the classifier makes a mistake, the error will be propagated and affect the performance of the model. Second, their method completely removes gender information from those words which are essential in some domains such as medicine and social science (Back et al., 2010; McFadden et al., 1992).

To overcome these limitations, we propose a learning scheme, Gender-Neutral Global Vectors (GN-GloVe) for training word embedding models with protected attributes (e.g., gender) based on GloVe (Pennington et al., 2014).[2] GN-GloVe represents protected attributes in certain dimen-

---

[1] Gender-definition words are the words associated with gender by definition (e,g., mother, waitress); the remainder are gender-neutral words.

[2] The code and data are released at `https://github.com/uclanlp/gn_glove`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4847–4853
Brussels, Belgium, October 31 - November 4, 2018.

sions while neutralizing the others during training. As the information of the protected attribute is restricted in certain dimensions, it can be removed from the embedding easily. By jointly identifying gender-neutral words while learning word vectors, GN-GloVe does not require a separate classifier to identify gender-neutral words; therefore, the error propagation issue is eliminated. The proposed approach is generic and can be incorporated with other word embedding models and be applied in reducing other societal stereotypes.

Our contributions are summarized as follows: 1) To our best knowledge, GN-GloVe is the first method to learn word embeddings with protected attributes; 2) By capturing protected attributes in certain dimensions, our approach ameliorates the interpretability of word representations; 3) Qualitative and quantitative experiments demonstrate that GN-GloVe effectively isolates the protected attributes and preserves the word proximity.

2 Related Work

Word Embeddings Word embeddings serve as a fundamental building block for a broad range of NLP applications (dos Santos and Gatti, 2014; Bahdanau et al., 2014; Zeng et al., 2015) and various approaches (Mikolov et al., 2013b; Pennington et al., 2014; Levy et al., 2015) have been proposed for training the word vectors. Improvements have been made by leveraging semantic lexicons and morphology (Luong et al., 2013; Faruqui et al., 2014), disambiguating multiple senses (Šuster et al., 2016; Arora et al., 2018; Upadhyay et al., 2017), and modeling contextualized information by deep neural networks (Peters et al., 2018). However, none of these works attempts to tackle the problem of stereotypes exhibited in embeddings.

Stereotype Analysis Implicit stereotypes have been observed in applications such as online advertising systems (Sweeney, 2013), web search (Kay et al., 2015), and online reviews (Wallace and Paul, 2016). Besides, Zhao et al. (2017) and Rudinger et al. (2018) show that coreference resolution systems are gender biased. The systems can successfully predict the link between "the president" with male pronoun but fail with the female one. Rudinger et al. (2017) use pointwise mutual information to test the SNLI (Bowman et al., 2015) corpus and demonstrate gender stereotypes as well as varying degrees of racial, religious, and age-based stereotypes in the corpus. A temporal analysis about word embeddings (Garg et al., 2018) captures changes in gender and ethnic stereotypes over time. Researchers attributed such problem partly to the biases in the datasets (Zhao et al., 2017; Yao and Huang, 2017) and word embeddings (Garg et al., 2017; Caliskan et al., 2017) but did not provide constructive solutions.

3 Methodology

In this paper, we take GloVe (Pennington et al., 2014) as the base embedding model and gender as the protected attribute. It is worth noting that our approach is general and can be applied to other embedding models and attributes. Following GloVe (Pennington et al., 2014), we construct a word-to-word co-occurrence matrix X, denoting the frequency of the j-th word appearing in the context of the i-th word as $X_{i,j}$. $w, \tilde{w} \in \mathbb{R}^d$ stand for the embeddings of a center and a context word, respectively, where d is the dimension.

In our embedding model, a word vector w consists of two parts $w = [w^{(a)}; w^{(g)}]$. $w^{(a)} \in \mathbb{R}^{d-k}$ and $w^{(g)} \in \mathbb{R}^k$ stand for neutralized and gendered components respectively, where k is the number of dimensions reserved for gender information.[3] Our proposed gender neutralizing scheme is to reserve the gender feature, known as "protected attribute" into $w^{(g)}$. Therefore, the information encoded in $w^{(a)}$ is independent of gender influence. We use $v_g \in \mathbb{R}^{d-k}$ to denote the direction of gender in the embedding space. We categorize all the vocabulary words into three subsets: male-definition Ω_M, female-definition Ω_F, and gender-neutral Ω_N, based on their definition in WordNet (Miller and Fellbaum, 1998).

Gender Neutral Word Embedding Our minimization objective is designed in accordance with above insights. It contains three components:

$$J = J_G + \lambda_d J_D + \lambda_e J_E, \qquad (1)$$

where λ_d and λ_e are hyper-parameters.

The first component J_G is originated from GloVe (Pennington et al., 2014), which captures the word proximity:

$$J_G = \sum_{i,j=1}^{V} f(X_{i,j}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{i,j} \right)^2 .$$

---

[3] We set $k = 1$ in this paper.

Here $f(X_{i,j})$ is a weighting function to reduce the influence of extremely large co-occurrence frequencies. b and $\tilde{b}$ are the respective linear biases for w and $\tilde{w}$.

The other two terms are aimed to restrict gender information in $w^{(g)}$, such that $w^{(a)}$ is neutral. Given male- and female-definition seed words Ω_M and Ω_F, we consider two distant metrics and form two types of objective functions.

In J_D^{L1}, we directly minimizing the negative distances between words in the two groups:

$$J_D^{L1} = -\left\| \sum_{w \in \Omega_M} w^{(g)} - \sum_{w \in \Omega_F} w^{(g)} \right\|_1 .$$

In J_D^{L2}, we restrict the values of word vectors in $[\beta_1, \beta_2]$ and push $w^{(g)}$ into one of the extremes:

$$J_D^{L2} = \sum_{w \in \Omega_M} \left\| \beta_1 \boldsymbol{e} - w^{(g)} \right\|_2^2 + \sum_{w \in \Omega_F} \left\| \beta_2 \boldsymbol{e} - w^{(g)} \right\|_2^2 ,$$

where $\boldsymbol{e} \in \mathcal{R}^k$ is a vector of all ones. β_1 and β_2 can be arbitrary values, and we set them to be 1 and -1, respectively.

Finally, for words in Ω_N, the last term encourages their $w^{(a)}$ to be retained in the null space of the gender direction v_g:

$$J_E = \sum_{w \in \Omega_N} \left(v_g^T w^{(a)} \right)^2 ,$$

where v_g is estimating on the fly by averaging the differences between female words and their male counterparts in a predefined set,

$$v_g = \frac{1}{|\Omega'|} \sum_{(w_m, w_f) \in \Omega'} (w_m^{(a)} - w_f^{(a)}),$$

where Ω' is a set of predefined gender word pairs.

We use stochastic gradient descent to optimize Eq. (1). To reduce the computational complexity in training the wording embedding, we assume v_g is a fixed vector (i.e., we do not derive gradient w.r.t v_g in updating $w^{(a)}, \forall w \in \Omega'$) and estimate v_g only at the beginning of each epoch.

4 Experiments

In this section, we conduct the following qualitative and quantitative studies: 1) We visualize the embedding space and show that GN-GloVe separates the protected gender attribute from other latent aspects; 2) We measure the ability of GN-GloVe to distinguish between gender-definition words and gender-stereotype words on a newly annotated dataset; 3) We evaluate GN-GloVe on standard word embedding benchmark datasets and show that it performs well in estimating word proximity; 4) We demonstrate that GN-Glove reduces gender bias on a downstream application, coreference resolution.

We compare GN-GloVe with two embedding models, GloVe and Hard-GloVe. GloVe is a widely-used model (Pennington et al., 2014), and we apply the post-processing step introduced in (Bolukbasi et al., 2016) to reduce gender bias in GloVe and name it after Hard-GloVe. All the embeddings are trained on *2017 English Wikipedia dump* with the default hyper-parameters decribed in (Pennington et al., 2014). When training GN-GloVe, we constrain the value of each dimension within $[-1, 1]$ to avoid numerical difficulty. We set λ_d and λ_e both to be 0.8. In our preliminary study on development data, we observe that the model is not sensitive to these parameters. Unless other stated, we use J_D^{L1} in the GN-GloVe model.

Separate protected attribute First, we demonstrate that GN-GloVe preserves the gender association (either definitional or stereotypical associations) in $w^{(g)}$[4]. To illustrate the distribution of gender information of different words, we plot Fig. 1a using $w^{(g)}$ for the x-axis and a random value for the y-axis to spread out words in the plot. As shown in the figure, the gender-definition words, e.g. "waiter" and "waitress", fall far away from each other in $w^{(g)}$. In addition, words such as "housekeeper" and "doctor" are inclined to different genders and their $w^{(g)}$ preserves such information.

Next, we demonstrate that GN-GloVe reduces gender stereotype using a list of profession titles from (Bolukbasi et al., 2016). All these profession titles are neutral to gender by definition. In Fig. 1b and Fig. 1c, we plot the cosine similarity between each word vector $w^{(a)}$ and the gender direction v_g (i.e., $\frac{w^T v_g}{\|w\| \|v_g\|}$). Result shows that words, such as "doctor" and "nurse", possess no gender association by definition, but their GloVe word vectors exhibit strong gender stereotype. In contrast, the gender projects of GN-GloVe word vectors $w^{(a)}$ are closer to zero. This demonstrates

---

[4] We follow the original GloVe implementation using the summation of word vector and context vector to represent a word. Therefore, the elements of the word vectors are constrained in [-2, 2]

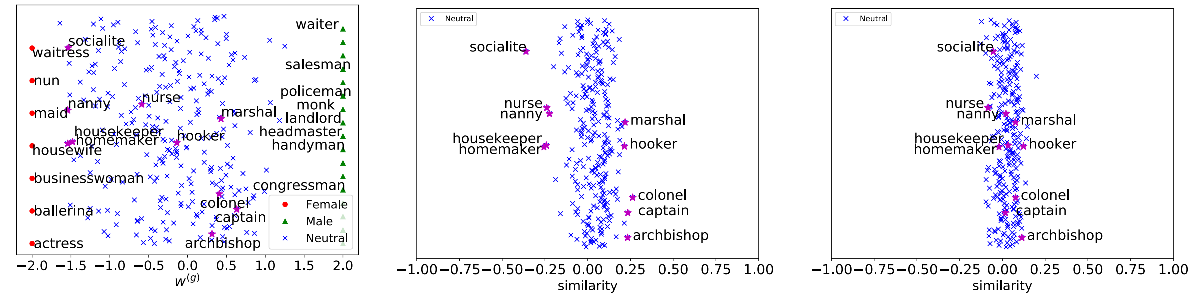

(a) $w^{(g)}$ dimension for all the professions

(b) Gender-neutral profession words projected to gender direction in GloVe

(c) Gender-neutral profession words projected to gender direction in GN-GloVe

Figure 1: Cosine similarity between the gender direction and the embeddings of gender-neutral words. In each figure, negative values represent a bias towards female, otherwise male.

the gender information has been substantially diminished from $w^{(a)}$ in the GN-GloVe embedding.

We further quantify the gender information exhibited in the embedding models. For each model, we project the word vectors of occupational words into the gender sub-space defined by "he-she" and compute their average size. A larger projection indicates an embedding model is more biased. Results show that the average projection of GloVe is 0.080, the projection of Hard-GloVe is 0.019, and the projection of Gn-Glove is 0.052. Comparing with GloVe, GN-GloVe reduces the bias by 35%. Although Hard-GloVe contains less gender information, we will show later GN-GloVe can tell difference between gender-stereotype and gender-definition words better.

Gender Relational Analogy To study the quality of the gender information present in each model, we follow SemEval 2012 Task2 (Jurgens et al., 2012) to create an analogy dataset, *SemBias*, with the goal to identify the correct analogy of "he - she" from four pairs of words. Each instance in the dataset consists of four word pairs: a gender-definition word pair (Definition; e.g., "waiter - waitress"), a gender-stereotype word pair (Stereotyp; e.g., "doctor - nurse") and two other pairs of words that have similar meanings (None; e.g., "dog - cat", "cup - lid")[5]. We consider 20 gender-stereotype word pairs and 22 gender-definition word pairs and use their Cartesian product to generate 440 instances. Among the 22 gender-definition word pairs, there are 2 word pairs that are not used as a seed word during the training. To test the generalization ability of the model, we generate a subset of data (SemBias (subset)) of 40 instances associated with these 2 pairs.

Dataset	Embeddings	Definition	Stereotype	None
SemBias	GloVe	80.2	10.9	8.9
	Hard-Glove	84.1	6.4	9.5
	GN-GloVe	97.7	1.4	0.9
SemBias (subset)	GloVe	57.5	20	22.5
	Hard-Glove	25	27.5	47.5
	GN-GloVe	75	15	10

Table 1: Percentage of predictions for each category on gender relational analogy task.

Table 1 lists the percentage of times that each class of pair is on the top based on a word embedding model (Mikolov et al., 2013c). GN-GloVe achieves 97.7% accuracy in identifying gender-definition word pairs as an analogy to "he - she". In contrast, GloVe and Hard-GloVe makes significantly more mistakes. On the subset, GN-GloVe also achieves significantly better performance than Hard-Glove and GloVe, indicating that it can generalize the gender pairs on the training set to identify other gender-definition word pairs.

Word Similarity and Analogy In addition, we evaluate the word embeddings on the benchmark tasks to ensure their quality. The word similarity tasks measure how well a word embedding model captures the similarity between words comparing to human annotated rating scores. Embeddings are tested on multiple datasets: WS353-ALL (Finkelstein et al., 2001), RG-65 (Rubenstein and Goodenough, 1965), MTurk-287 (Radinsky et al., 2011), MTurk-771 (Halawi et al., 2012), RW (Luong et al., 2013), and MEN-TR-3k (Bruni et al., 2012)

---

[5] The pair is sampled from the list of word pairs with "SIMILAR: Coordinates" relation annotated in (Jurgens et al., 2012). The original list has 38 pairs. After removing gender-definition word pairs, 29 are left.

Embeddings	Analogy		Similarity					
	Google	MSR	WS353-ALL	RG-65	MTurk-287	MTurk-771	RW	MEN-TR-3k
GloVe	**70.8**	**45.8**	62.0	75.3	64.8	64.9	37.3	72.2
Hard-GloVe	**70.8**	**45.8**	61.2	74.8	64.4	64.8	37.3	72.2
GN-GloVe-L1	68.9	43.7	**62.8**	74.1	66.2	**66.2**	**40.0**	**74.5**
GN-GloVe-L2	68.8	43.6	62.5	**76.4**	**66.8**	65.6	39.3	74.4

Table 2: Results on the benchmark datasets. Performance is measured in accuracy and in Spearman rank correlation for word analogy and word similarity tasks, respectively.

datasets. The analogy tasks are to answer the question "*A* is to *B* as *C* is to _?" by finding a word vector w that is closest to $w_A - w_B + w_C$ in the embedding space. Google (Mikolov et al., 2013a) and MSR (Mikolov et al., 2013c) datasets are utilized for this evaluation. The results are shown in Table 2, where the suffix "-L1" and "-L2" of GN-GloVe stand for the GN-GloVe using J_D^{L1} and J_D^{L2}, respectively. Compared with others, GN-GloVe achieves a higher accuracy in the similarity tasks and its analogy score slightly drops indicating that GN-GloVe is capable of preserving proximity among words.

Coreference Resolution Finally, we investigate how the gender bias in word embeddings affects a downstream application, such as coreference resolution. Coreference resolution aims at clustering the denotative noun phrases referring to the same entity in the given text. We evaluate our models on the Ontonotes 5.0 (Weischedel et al., 2012) benchmark dataset and the WinoBias dataset (Zhao et al., 2018).[6] In particular, the WinoBias dataset is composed of pro-stereotype (PRO) and anti-stereotype (ANTI) subsets. The PRO subset consists of sentences where a gender pronoun refers to a profession, which is dominated by the same gender. Example sentences include "The CEO raised the salary of the receptionist because he is generous." In this sentence, the pronoun "he" refers to "CEO" and this reference is consistent with societal stereotype. The ANTI subset contains the same set of sentences, but the gender pronoun in each sentence is replaced by the opposite gender. For instance, the gender pronoun "he" is replaced by "she" in the aforementioned example. Despite the sentence is almost identical, the gender pronoun now refers to a profession that is less represented by the gender. Details about the dataset are in (Zhao et al., 2018).

Embeddings	OntoNotes-test	PRO	ANTI	Avg	Diff
GloVe	66.5	76.2	46.0	61.1	30.2
Hard-Glove	66.2	70.6	54.9	62.8	15.7
GN-GloVe	66.2	72.4	51.9	62.2	20.5
GN-GloVe(w_a)	65.9	70.0	53.9	62.0	16.1

Table 3: F1 score (%) on the coreference system.

We train the end-to-end coreference resolution model (Lee et al., 2017) with different word embeddings on OntoNote and report their performance in Table 3. For the WinoBias dataset, we also report the average (Avg) and absolute difference (Diff) of F1 scores on two subsets. A smaller Diff value indicates less bias in a system. Results show that GN-GloVe achieves comparable performance as Glove and Hard-GloVe on the OntoNotes dataset while distinctly reducing the bias on the WinoBias dataset. When only the $w^{(a)}$ potion of the embedding is used in representing words, GN-GloVe($w^{(a)}$) further reduces the bias in coreference resolution.

5 Conclusion and Discussion

In this paper, we introduced an algorithm for training gender-neutral word embedding. Our method is general and can be applied in any language as long as a list of gender definitional words is provided as seed words (e.g., gender pronouns). Future directions include extending the proposed approach to model other properties of words such as sentiment and generalizing our analysis beyond binary gender.

Acknowledgement

This work was supported by National Science Foundation Grant IIS-1760523. We would like to thank James Zou, Adam Kalai, Tolga Bolukbasi, and Venkatesh Saligrama for helpful discussion, and anonymous reviewers for their feedback.

---

[6] Specifically, we conduct experiments on the Type 1 version.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2018. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association of Computational Linguistics*, 6:483–495.

Sudie E Back, Rebecca L Payne, Annie N Simpson, and Kathleen T Brady. 2010. Gender and prescription opioids: Findings from the national survey on drug use and health. *Addictive behaviors*, 35(11):1001–1007.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2017. Word embeddings quantify 100 years of gender and ethnic stereotypes. *arXiv preprint arXiv:1711.08412*.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM.

David A Jurgens, Peter D Turney, Saif M Mohammad, and Keith J Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 356–364. Association for Computational Linguistics.

Matthew Kay, Cynthia Matuszek, and Sean A Munson. 2015. Unequal representation and gender stereotypes in image search results for occupations. In *Human Factors in Computing Systems*, pages 3819–3828.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *EMNLP*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Anna C McFadden, George E Marsh, Barrie Jo Price, and Yunhan Hwang. 1992. A study of race and gender bias in the punishment of school children. *Education and treatment of children*, pages 140–146.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

George Miller and Christiane Fellbaum. 1998. Wordnet: An electronic lexical database.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 74–79.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *NAACL*.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Simon Šuster, Ivan Titov, and Gertjan Van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. *arXiv preprint arXiv:1603.09128*.

Latanya Sweeney. 2013. Discrimination in online ad delivery. *Queue*, 11(3):10.

Shyam Upadhyay, Kai-Wei Chang, Matt Taddy, Adam Kalai, and James Zou. 2017. Beyond bilingual: Multi-sense word embeddings using multilingual context. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 101–110.

Byron C Wallace and Michael J Paul. 2016. jerk or judgemental? patient perceptions of male versus female physicians in online reviews.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Nianwen Xue, Martha Palmer, Jena D Hwang, Claire Bonial, et al. 2012. Ontonotes release 5.0.

Sirui Yao and Bert Huang. 2017. Beyond parity: Fairness objectives for collaborative filtering. In *Advances in Neural Information Processing Systems*, pages 2925–2934.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *arXiv preprint arXiv:1707.09457*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. *NAACL*.

Learning Concept Abstractness Using Weak Supervision

Ella Rabinovich[▲][*] **Benjamin Sznajder**[†] **Artem Spector**[†] **Ilya Shnayderman**[†]
Ranit Aharonov[†] **David Konopnicki**[†] **Noam Slonim**[†]
[†]IBM Research
▲ Dept. of Computer Science, University of Toronto
{benjams, artems, ilyashn, ranita, davidko, noams}@il.ibm.com
ella@cs.toronto.edu

Abstract

We introduce a weakly supervised approach for inferring the property of abstractness of words and expressions in the complete absence of labeled data. Exploiting only minimal linguistic clues and the contextual usage of a concept as manifested in textual data, we train sufficiently powerful classifiers, obtaining high correlation with human labels. The results imply the applicability of this approach to additional properties of concepts, additional languages, and resource-scarce scenarios.

1 Introduction

During the last decades, the influence of psycholinguistic properties of words on cognitive processes has become a major topic of scientific inquiry. Among the most studied psycholinguistic attributes are concreteness, familiarity, imagery, and average age of acquisition. *Abstractness* (the opposite of concreteness) quantifies the degree to which an expression denotes an entity that can be directly perceived by human senses.

Word abstractness ratings were first collected by Spreen and Schulz (1966) and Paivio et al. (1968), and made available in the MRC database (Coltheart, 1981) for 4,292 English words. Since its release, this database has stimulated research in a wide range of linguistic tasks, as well as artificial intelligence and cognitive studies. Despite their evident usefulness, resources providing abstractness ratings are relatively rare and of limited size. Here, we address the task of *automatically* inferring the abstractness rating of a concept by applying a *weakly supervised* approach that exploits minimal linguistic clues.

Studies on derivational morphological processes indicate that word meaning is often entailed by its morphology. As an example, word suffixation by *-ant* or *-ent* is used to denote a person, as in *assistant*, while the suffix *-hood* yields nouns meaning "condition of being", as in *childhood*. A wide range of word-formation processes was described by Huddleston and Pullum (2002); in particular, the authors detail categories of suffixes that are used to derive words, broadly perceived as abstract, e.g., *-ism* as in *feminism*, or *-ness* as in *agreeableness*.

Concept abstractness indicators are also likely to be manifested in its contextual usage. Consider the two sentences below, each embedding abstract and concrete words – one describing *feminism* and the other *screwdriver* – respectively:

> *Second- and third-wave* ***feminism*** *in China involved a reexamination of women's roles during the communist revolution and other reform movements, and new discussions about whether women's equality has been fully achieved.*

> *Many* ***screwdriver*** *handles are not smooth and often not round, but have bumps or other irregularities to improve grip and to prevent the tool from rolling when on a flat surface.*

We hypothesize that the immediate neighborhood of a word as reflected in embedding sentences captures the signal of abstractness. In the examples above, several potential clues for the degree of word abstractness are underlined.

Correspondingly, we propose a method for inferring the degree of abstractness of concepts in the complete absence of labeled data, by exploiting (1) a minimal set of morphological word-formation clues; and (2) a text corpus for learning the context in which words tend to appear.

We demonstrate that this method allows us to infer the abstractness ratings of unigram, bigram and trigram Wikipedia concepts (titles) – the task that, to the best of our knowledge, was only addressed through manual labeling so far (Brysbaert et al., 2014). The main contribution of this work is, therefore, in the proposal and evaluation of a

---

[*] Work done while the author was at IBM Research.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4854–4859
Brussels, Belgium, October 31 - November 4, 2018.

weakly supervised methodology for inferring the abstractness rating of concepts, potentially applicable to additional languages. The suggested approach may also be applicable for predicting other word and concept properties, when those are manifested in both morphology and context. Finally, we release a dataset of 300K Wikipedia concepts automatically rated for their degree of abstractness, and additional 1500 unigram, bigram and trigram concepts annotated with both manual and predicted scores.[1]

2 Related work

A large body of research addressed the relations of word abstractness and cognitive processes (Connell and Lynott, 2012; Gianico-Relyea and Altarriba, 2012; Oliveira et al., 2013; Nishiyama, 2013; Paivio, 2013; Barber et al., 2013). Computational investigation of word abstractness and concreteness has been a prolific field of recent research, laying out an empirical foundation for the theoretically motivated hypotheses on the characteristics of these properties. A ranker trained on psycholinguistic features extracted from the MRC database (in combination with other features) reached first place in the English Lexical Simplification task at SemEval 2012 (Jauhar and Specia, 2012). Hill and Korhonen (2014) achieved state-of-the-art performance in Semantic Composition and Semantic Modification prediction by including concreteness in the set of features used by the model.

Along the years, several works extended the seed MRC dataset by employing various *supervised* machine learning techniques, further utilizing the extended dataset for tasks of lexical simplification (Paetzold and Specia, 2016b,a), cross-lingual metaphor detection (Tsvetkov et al., 2013), literal and metaphorical sense identification (Turney et al., 2011), as well as readability assessment of Brazilian Portuguese (dos Santos et al., 2017). Feng et al. (2011) exploited word attributes from WordNet, properties extracted from the CELEX database, and Latent Semantic Analysis over a large text corpus for building a linear regression model predicting abstractness rate; the model accounted for 64% variance of human annotations.

A comprehensive survey of psycholinguistic and memory research on word concreteness is presented in Brysbaert et al. (2014) (BWK), who conducted a large-scale manual annotation of concreteness ratings for over 40K concepts, further used by Rothe et al. (2016) to infer concreteness ratings for the whole Google News lexicon. To the best of our knowledge, our work is the first attempt to automatically infer the property of concept abstractness in the complete absence of labeled data.

3 Predicting concept abstractness

3.1 Abstractness indicators

Nominalization is a word-formation process that involves the formation of nouns from bases of other classes by means of affixation. As an example, a derivational suffix can be added to an adjective (capable+*ity* for *capability*) or a verb (react+*tion* for *reaction*) to create a noun. Various word-formation processes often enrich words with meaning associated with certain semantic grouping. Huddleston and Pullum (2002) detail nominalization processes that serve to form nouns denoting a “state” or “condition of being”, which in turn are broadly associated with abstractness. As such, the suffixes *-ety, -ity* and *-ness* carry over the general meaning of “quality or state of being” and the suffix *-ism* is used to form nouns denoting a range of doctrines, beliefs and movements (Huddleston and Pullum, 2002). Additional suffixes that tend to form English nouns with high degree of abstractness include *-ance, -ence, -ation, -ution, -dom, -hood, -ship* and *-y*.

3.2 Dataset

We used the English Wikipedia[2] article titles as a proxy for retrieving frequently used single- and multi-word expressions, thereby associating over 5M Wikipedia titles with concepts.

Training data We chose two abstractness signals, manifested by the suffixes *-ism* and *-ness*, representing different types of abstract meanings. We extracted 1,040 potentially abstract unigram Wikipedia titles suffixed by either of the two (the positive class). The – admittedly noisy – concrete (negative) class was generated by randomly selecting the same number of unigram concepts from the complementary set of titles.In both cases, we set a threshold[3] on the frequency of a concept in the

---

[1] The datasets are available for download at `https://www.research.ibm.com/haifa/dept/vst/debating_data.shtml`

[2] We used the Wikipedia May 2017 dump.

[3] The minimum of 20 occurrences for a concept.

corpus, and filtered out non-alphabetic unigrams and unigrams containing special characters. We assessed the quality of the positive and negative weakly-labeled training unigrams by manual annotation of their level of abstractness, obtaining abstractness prior of 93% in the set of presumably abstract concepts, and concreteness prior of 81% for the opposite class.

Given this set of weakly-labeled positive and negative concepts, we randomly selected a set of Wikipedia sentences that include any of these concepts (equally split by positive and negative unigrams), to be used in the training phase, while limiting sentence length to the range of 10 to 70 tokens. This step resulted in about 400K train sentences in each class, 800K in total. The final preprocessing phase involved masking a sentence concept with a generic token, aiming to prevent the classifier from training on the concept itself, and instead training on its contextual usage.

Evaluation data A randomly selected set of 1500 Wikpedia concepts (with the minimum of 500 occurrences per concept), split equally between unigrams, bigrams and trigrams, and distinct from the training set, was used for testing prediction. We henceforth refer to this set of concepts as the *evaluation set*. Each of these concepts was manually annotated for abstractness on the 1–7 scale by seven in-house labelers, using an adaptation of the guidelines by Spreen and Schulz (1966) to the multi-word scenario:

> *Words or phrases may refer to persons, places and things that can be seen, heard, felt, smelled or tasted or to more abstract concepts that cannot be experienced by our senses. The purpose of this task is to rate a list of concepts with respect to "concreteness" in terms of sense-experience. Any expression that refers to objects, materials or persons should receive a high concreteness rating; any expression that refers to an abstract concept that cannot be experienced by the senses should receive a low concreteness rating. Concrete concepts typically have physical or concrete existence, while abstract do not. Think of the concepts "onion" and "nationalism" – "onion" can be experienced by our senses and therefore should be rated as concrete (1); "nationalism" cannot be experienced by the senses as such and therefore should be rated as abstract (7).*

Word polysemy is a common challenge in tasks related to lexical semantics. As such, our perception of the concreteness rate of the concept *bank* may vary depending on whether a financial institution or a river bank is concerned. While we could not avoid this issue altogether (since working with pre-trained word representations that do not carry disambiguation information), we ensured that all in-house labelers annotated the same word sense by providing them with Wikipedia definition of the most frequent sense of a concept.

The final abstractness score was computed as the average over individual annotations. The average pairwise weighted Kappa agreement[4] on the entire set of 1500 concepts was 0.65.

3.3 Classification models

We hypothesize that words that share similar degree of abstractness tend to share certain similarities in their contextual usage; that, in contrast to concepts that exhibit opposite abstractness rate. Indeed, a statistical significance test applied to the (weak) positive and negative training data (Section 3.2) reveals markers such as {*parish, movement, century, spiritual, life, doctrine, nature, regime*} sharing excessive frequency in sentences containing abstract concepts. The very essence of this phenomenon is captured by distributed word representations (Mikolov et al., 2013; Pennington et al., 2014), a.k.a. word embeddings, learned based on the contextual usage of words. We therefore trained three classifiers, each exploiting different language properties, as described below.

Naive Bayes (NB) Using solely word counts in textual data, we used a simple probabilistic Naive Bayes classifier, with a bag-of-words feature set extracted from the 800K sentences containing positive and negative training concepts. Given a sentence containing a test concept, its degree of abstractness was defined as the posterior probability assigned by the classifier. Aiming at robust classification, we retrieved 500 sentences containing each test concept from the corpus. Consequently, the final abstractness score of a concept was calculated by averaging the predictions assigned by the classifier to individual sentences.

Nearest neighbor We used the nearest neighbors algorithm, specifically, its *radius-based* version (NN-RAD), using the pre-trained GloVe embeddings (Pennington et al., 2014). This classifier

---

[4] We used the implementation in `http://scikit-learn.org`, with "quadratic" scheme.

estimates the degree of concept abstractness given only its distributional representation.

The abstractness score of a test concept was computed by the ratio of its abstract neighbors to the total number of concepts within the predefined radius, where the entire set of neighbors is limited to the concepts in the weakly-labeled training set. The proximity threshold (radius) was set to 0.25, w.r.t. the cosine similarity between two embedding vectors.[5] Multi-word concepts were subject to more careful processing, where the classifier computed a multi-word concept representation as an average of representations of its individual words, and further estimated the abstractness score of the obtained embedding. In case that one of a concept constituents was not found in embeddings, we excluded the concept from computation.

RNN Aiming at exploiting both embeddings and textual data, we utilized a bidirectional recurrent neural network (RNN) with one layer of forward and backward LSTM cells. Each cell has width of 128, and is wrapped by a dropout wrapper with keep probability 0.85. An attention layer was created in order to weigh words according to their proximity to the train/test concept. The output of the LSTM cells is passed to the attention layer which reduces it to the size of 100. The output of the attention layer is passed to a fully connected layer which produces the final prediction of the abstractness level of a concept. GloVe embeddings with 300 dimensions were used as word representations. Given a set of sentences containing a test concept, its final abstractness score was computed by applying the averaging procedure described for the Naive Bayes classifier.

4 Results

We demonstrate that trained models discover linguistic patterns associated with abstract meaning (beyond those known at training), and furthermore yield abstractness scores that correlate significantly with human annotations.

4.1 Revealing abstractness markers

We automatically scored 100K unigram Wikipedia concepts for abstractness with all classifiers and extracted the set of suffixes that share excessive frequency in the top-k abstract concepts using the statistical proportion test. More specifically, we applied the test to the exhaustive list of all three-character English suffixes (e.g., *-aaa, -aab*), counting their occurrences in the subset of concepts with the highest abstractness scores[6] (the population under test) and in the remainder (the background). Our hypothesis was that suffixes associated with abstract meaning in the literature will be over-represented in the population of concepts ranked as abstract by the classifiers. The top-10 suffixes, scored by their statistical significance p-value[7] were {***-ism***, *-ity, -ion, -sis, -ics,* ***-ess***, *-phy, -nce, -ogy, -ing*} – suffixes broadly associated with abstractness in the literature (where all suffixes but two are distinct from the training data). The underlying concept examples included {*illegalism, modernity, antireligion, henosis, politics, lawlessness, ecosophy, conscience, ideology, enabling*} – words broadly perceived as abstract.

4.2 Abstractness rating

Table 1 presents a few examples of abstract and concrete concepts, as identified by manual annotation, along with their abstractness score as predicted by the RNN classifier (Section 3.3).

abstract		concrete	
concept	score	concept	score
marxism	0.972	plywood	0.000
islamophobia	0.969	Wiltshire	0.000
affirmative action	0.844	moonlight	0.058
absolute monarchy	0.842	convoy	0.112
sincerity	0.836	gadget	0.120

Table 1: Examples of concepts found as abstract/concrete (above/below the average score of 0.5) via manual annotation, along with their score as predicted by RNN.

Table 2 presents the Pearson correlation between the abstractness scores as assigned by the classifiers and the manual annotations over the evaluation set. We also present the correlation of scores produced by our classifiers to the set of Wikipedia concepts from the manually annotated MRC database (MRC-seed, Section 1), and to the set of 5883 noun concepts[8] from manually annotated BWK dataset (Brysbaert et al., 2014).

Evidently, the best results are obtained by the RNN classifier, yielding up to 0.740 correlation

---

[5] The radius was tuned on the set of 500 unigrams.

[6] We used the set of 18% highest ranked concepts – the fraction of abstract concepts in a sample population, as estimated by manual labeling.

[7] In all cases the obtained p-value was practically zero.

[8] Only concepts that can be mapped to a corresponding Wikipedia page were considered.

test set	Naive Bayes	NN-RAD	RNN
BWK	**0.657**	0.622	0.634
MRC-seed	**0.674**	0.576	0.669
1-grams	0.679	0.638	**0.740**
2-grams	0.565	0.515	**0.666**
3-grams	0.412	0.467	**0.490**

Table 2: Correlation of abstractness scores assigned by the classifiers to manual annotations.

with human annotations. Notably, the simple Naive Bayes, utilizing only textual data, yields results of reasonable quality; the broad implication of this outcome lies in the potential applicability of this approach to resource-scarce scenarios where high quality word embeddings are not available. Interestingly, while using Google word2vec embeddings (instead of Glove) yielded similar results, utilizing fastText pre-trained representations (Joulin et al., 2016) obtained more accurate ranking, e.g., the NN-RAD classifier yielded correlation of 0.688 for the BWK dataset, compared to 0.622 obtained using Glove (Table 2). We attribute this improvement to the fact that fastText embeddings better capture morphological word properties and cover more extensive vocabulary.

The relatively low correlation obtained with trigram concepts can be explained by the inherent complexity introduced by the multi-word scenario, challenging still further the subjective human perception of abstractness. While inter-labeler agreement for unigrams and bigrams was 0.72 and 0.66, respectively, it only reached 0.54 for trigrams, supporting the aforementioned hypothesis.

4.3 Varying the size of a test set

How many sentences containing a test concept suffice for a reliable prediction? We address this question by limiting the number of (randomly chosen) sentences used for rating. While the correlation obtained by RNN with 500 sentences containing a test concept reached 0.740 (Table 2), as little as 10, and even 5 sentences yielded correlation of 0.706 and 0.675, respectively, implying the efficiency and effectiveness of the presented approach in the availability of only little data. The plot in Figure 1 presents the correlation of the RNN and NB classifiers to label as function of number of (randomly sampled) sentences used for evaluation. Each such experiment (e.g., using 1, 5, 10 sentences) was averaged over 50 runs; the average correlation to label, as well as standard deviation, are plotted on the chart. The constant correlation yield by the (text-independent) NN-RAD algorithm is illustrated by the vertical line.

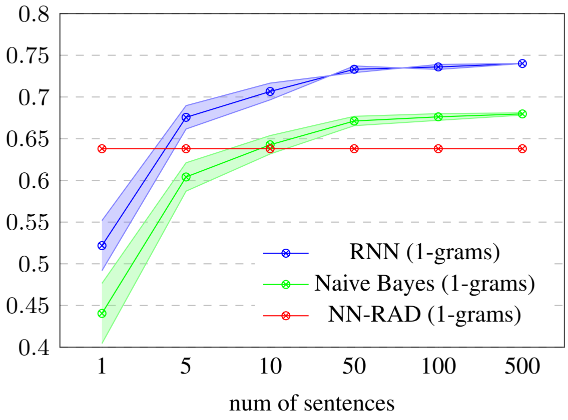

Figure 1: Average correlation (and standard deviation) to manual annotation as function of number of sentences used for evaluation.

4.4 Comparison to supervised models

Tsvetkov et al. (2013) used supervised learning algorithm to propagate abstractness scores to words using pre-trained word representations. Utilizing vector elements as features, they trained a supervised classifier, and predicted the degree of abstractness for unseen words. Abstractness rankings from the MRC database were used as a training set, and the classifier predictions were binarized into abstract-concrete boolean indicators using predefined thresholds. The authors obtained 94% accuracy when tested on held-out data.

5 Conclusions

We presented a weakly supervised approach for inferring the degree of concept abstractness. Our results demonstrate that a minimal morphological signal and a textual corpus are sufficient to train classifiers that yield relatively accurate predictions, that in turn can be used to unravel additional linguistic patterns indicative of the same property. Our future plans include exploring the value of the proposed methodology with other languages and additional properties.

Acknowledgments

We are grateful to Dafna Sheinwald and Shuly Wintner for much advise and helpful suggestions. We also thank our anonymous reviewers for their constructive feedback.

References

Horacio A Barber, Leun J Otten, Stavroula-Thaleia Kousta, and Gabriella Vigliocco. 2013. Concreteness in word processing: ERP and behavioral effects in a lexical decision task. *Brain and language*, 125(1):47–53.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior research methods*, 46(3):904–911.

Max Coltheart. 1981. The MRC psycholinguistic database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.

Louise Connell and Dermot Lynott. 2012. Strength of perceptual experience predicts word processing performance better than concreteness or imageability. *Cognition*, 125(3):452–465.

Shi Feng, Zhiqiang Cai, Scott A Crossley, and Danielle S McNamara. 2011. Simulating human ratings on word concreteness. In *Proceedings of the FLAIRS Conference*.

Jennifer L Gianico-Relyea and Jeanette Altarriba. 2012. Word concreteness as a moderator of the tip-of-the-tongue effect. *The Psychological Record*, 62(4):763–776.

Felix Hill and Anna Korhonen. 2014. Concreteness and subjectivity as dimensions of lexical meaning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 725–731.

Rodney Huddleston and Geoffrey K. Pullum. 2002. *The cambridge grammar of English*. Cambridge University Press.

Sujay Kumar Jauhar and Lucia Specia. 2012. UOW-SHEF: Simplex–lexical simplicity ranking based on contextual and psycholinguistic features. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 477–481. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in neural information processing systems*, pages 3111–3119.

Ryoji Nishiyama. 2013. Dissociative contributions of semantic and lexical-phonological information to immediate recognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 39(2):642–648.

J Oliveira, MV Perea, V Ladera, and P Gamito. 2013. The roles of word concreteness and cognitive load on interhemispheric processes of recognition. *Laterality: Asymmetries of Body, Brain and Cognition*, 18(2):203–215.

Gustavo Paetzold and Lucia Specia. 2016a. Collecting and exploring everyday language for predicting psycholinguistic properties of words. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1669–1679.

Gustavo Paetzold and Lucia Specia. 2016b. Inferring psycholinguistic properties of words. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 435–440.

A Paivio. 2013. Dual coding theory, word abstractness, and emotion: a critical review of kousta et al.(2011). *Journal of experimental psychology.*, 142(1):282.

Allan Paivio, John C Yuille, and Stephen A Madigan. 1968. Concreteness, imagery, and meaningfulness values for 925 nouns. *Journal of experimental psychology*, 76(1p2):1.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532–1543.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. *arXiv preprint arXiv:1602.07572*.

Leandro B. dos Santos, Magali Sanches Duran, Nathan Siegle Hartmann, Arnaldo Candido, Gustavo Henrique Paetzold, and Sandra Maria Aluisio. 2017. A lightweight regression method to infer psycholinguistic properties for Brazilian Portuguese. In *International Conference on Text, Speech, and Dialogue*, pages 281–289. Springer.

Otfried Spreen and Rudolph W Schulz. 1966. Parameters of abstraction, meaningfulness, and pronunciability for 329 nouns. *Journal of Verbal Learning and Verbal Behavior*, 5(5):459–468.

Yulia Tsvetkov, Elena Mukomel, and Anatole Gershman. 2013. Cross-lingual metaphor detection using common semantic features. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 45–51.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 680–690. Association for Computational Linguistics.

Word Sense Induction with Neural biLM and Symmetric Patterns

Asaf Amrami † and **Yoav Goldberg** † ‡
† Computer Science Department, Bar Ilan University, Israel
‡ Allen Institute for Artificial Intelligence
`{asaf.amrami, yoav.goldberg}@gmail.com`

Abstract

An established method for Word Sense Induction (WSI) uses a language model to predict probable substitutes for target words, and induces senses by clustering these resulting substitute vectors.

We replace the ngram-based language model (LM) with a recurrent one. Beyond being more accurate, the use of the recurrent LM allows us to effectively query it in a creative way, using what we call *dynamic symmetric patterns*. The combination of the RNN-LM and the dynamic symmetric patterns results in strong substitute vectors for WSI, allowing to surpass the current state-of-the-art on the SemEval 2013 WSI shared task by a large margin.

1 Introduction

We deal with the problem of *word sense induction* (WSI): given a target lemma and a collection of within-sentence usages it, cluster the usages (**instances**) according to the different senses of the target lemma. For example, for the sentences:

(a) We spotted a large ***bass*** in the ocean.

(b) The ***bass*** player did not receive the acknowledgment she deserves.

(c) The black sea ***bass***, is a member of the wreckfish family.

We would like to cluster (a) and (c) in one group and (b) in another.[1] Note that some mentions are ambiguous. For example, (d) matches both the music and the fish senses:

(d) ***Bass*** scales are the worst.

This calls for a *soft clustering*, allowing to probabilistically associate a given mention to two senses.

The problem of WSI has been extensively studied with a series of shared tasks on the topic (Agirre and Soroa, 2007; Manandhar et al., 2010; Jurgens and Klapaftis, 2013), the latest being SemEval 2013 Task 13 (Jurgens and Klapaftis, 2013). Recent state-of-the-art approaches to WSI rely on generative graphical models (Lau et al., 2013; Wang et al., 2015; Komninos and Manandhar, 2016). In these works, the sense is modeled as a latent variable that influences the context of the target word. The later models explicitly differentiate between local (syntactic, close to the disambiguated word) and global (thematic, semantic) context features.

Substitute Vectors Baskaya et al. (2013) take a different approach to the problem, based on *substitute vectors*. They represent each instance as a distribution of possible substitute words, as determined by a language model (LM). The substitute vectors are then clustered to obtain senses.

Baskaya et al. (2013) derive their probabilities from a 4-gram language model. Their system (AI-KU) was one of the best performing at the time of SemEval 2013 shared task. Our method is inspired by the AI-KU use of substitution based sense induction, but deviate from it by moving to a recurrent language model. Besides being more accurate, this allows us to further improve the quality of the derived substitutions by the incorporation of *dynamic symmetric patterns.*

BiLM Bidirectional RNNs were shown to be effective for word-sense disambiguation and lexical substitution tasks (Melamud et al., 2016; Yuan et al., 2016; Raganato et al., 2017). We adopt the

---

[1] This example shows *homonymy*, a case where the same word form has two distinct meaning. A more subtle case is *polysemy*, where the senses share some semantic similarity. In "She played a low bass note", the sense of *bass* is related to the sense in (b), but distinct from it. The WSI task we tackle in this work deals with both cases.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4860–4867
Brussels, Belgium, October 31 - November 4, 2018.

ELMo biLM model of Peters et al. (2018), which was shown to produce very competitive results for many NLP tasks. We use the pre-trained ELMo biLM provided by Peters et al. (2018).[2] However, rather than using the LSTM state vectors as suggested in the ELMo paper, we opt instead to use the predicted word probabilities. Moving from continuous and opaque state vectors to discrete and transparent word distributions allows far better control of the resulting representations (e.g. by sampling, re-weighting and lemmatizing the words) as well as better debugging opportunities.

As expected, the move to the neural biLM already outperforms the AI-KU system, and matches the previous state-of-the-art. However, we observe that the substitute vectors do not take into account the disambiguated word itself. We find that this often results in noisy substitutions. As a motivating example, consider the sentence "the doctor recommends *oranges* for your health". Here, *running* is a perfectly good substitution, as the "fruitness" of the target word itself isn't represented in the context. We would like the substitutes word distribution representing the target word to take both kinds of information—the context as well as the target word—into account.

Dynamic Symmetric Patterns Our main proposal incorporates such information. It is motivated by Hearst patterns (Hearst, 1992; Widdows and Dorow, 2002; Schwartz et al., 2015), and made possible by neural LMs. Neural LMs are better in capturing long-range dependencies, and can handle and predict unseen text by generalizing from similar contexts. Conjunctions, and in particular the word *and*, are known to combine expressions of the same kind. Recently, Schwartz et al. (2015) used conjunctive symmetric patterns to derive word embeddings that excel at capturing word similarity. Similarly, Kozareva et al. (2008) search for doubly-anchored patterns including the word *and* in a large web-corpus to improve semantic-class induction. The method of Schwartz et al. (2015) result in context-independent embeddings, while that of Kozareva et al. (2008) takes some context into account but is restricted to exact corpus matches and thus suffers a lot from sparsity.

We make use of the rich sequence representation capabilities of the neural biLM to derive *context-dependent symmetric pattern substitutions*. Relying on the generalization properties of neural language models and the abundance of the "X and Y" pattern, we present the language model with a dynamically created incomplete pattern, and ask it to predict probable completion candidates. Rather than predicting the word distribution following *the doctor recommends* __, we instead predict the distribution following *the doctor recommends oranges and* __. This provides substantial improvement, resulting in state-of-the-art performance on the SemEval 2013 shared task.

The code for reproducing the experiments and our analyses is available at `https://github.com/asafamr/SymPatternWSI`.

2 Method

Given a target word (lemma and its part-of-speech pair), together with several sentences in which the target word is used (instances), our goal is to cluster the word usages such that each cluster corresponds to a different sense of the target word. Following the SemEval 2013 shared task and motivating example (d) from the introduction, we seek a soft (probabilistic) clustering, in which each word instance is assigned with a probability of belonging to each of the sense-clusters.

Our algorithm works in three stages: (1) We first associate each instance with a probability distribution over in-context word-substitutes. This probability distribution is based on a neural biLM (section 2.1). (2) We associate each instance with k representatives, each containing multiple samples from its associated word distributions (section 2.3). (3) Finally, we cluster the representatives and use the hard clustering to derive a soft-clustering over the instances (section 2.4).

We use the pre-trained neural biLM as a black-box, but use linguistically motivated processing of both its input and its output: we rely on the generalization power of the biLM and query it using *dynamic symmetric patterns* (section 2.2); and we lemmatize the resulting word distributions.

Running example In what follows, we demonstrate the algorithm using a running example of inducing senses from the word *sound*, focusing on the instance sentence:

*I liked the **sound** of the harpsichord.*

2.1 biLM Derived Substitutions

We follow the ELMo biLM approach (Peters et al., 2018) and consider two separately trained

---

[2] We thank the ELMo team for sharing the pre-trained models.

language models, a forward model trained for predicting $p_{\rightarrow}(w_i|w_1, ..., w_{i-1})$ and a backward model $p_{\leftarrow}(w_i|w_n, ..., w_{i+1})$. Rather than combining the two models' predictions into a single distribution, we simply associate the target word with two distributions, one from $p_{\rightarrow}$ and one from $p_{\leftarrow}$. For convenience, we use $LM_{\rightarrow}(w_1 w_2 ... w_{i-1}\_\_)$ to denote the distribution $p_{\rightarrow}(w_i|w_1, ..., w_{i-1})$ and $LM_{\leftarrow}(\_\_w_{i+1} w_{i+2} ... w_n)$ to denote $p_{\leftarrow}(w_i|w_n, ..., w_{i+1})$.

Context-based substitution In the purely context-based setup (the one used in the AI-KU system) we represent the target word *sounds* by the two distributions:

$LM_{\rightarrow}$(`<s>` I liked the __)
$LM_{\leftarrow}$(__ of the harpsichord `</s>`)

The resulting top predictions from each distribution are:
{*idea:0.12, fact:0.07, article: 0.05, guy: 0.04, concept: 0.02*} and
{*sounds:0.04, version: 0.03, rhythm: 0.03, strings: 0.03, piece: 0.02*} respectively.

2.2 Dynamic Symmetric Patterns

As discussed in the introduction, conditioning solely on context is ignoring valuable information. This is evident in the resulting word distributions. We use the coordinative symmetric pattern *X and Y* in order to produce a substitutes vector incorporating both the word and its context. Concretely, we represent a target word w_i by $p_{\rightarrow}(w'|w_1, ..., w_i, \texttt{and})$ and $p_{\leftarrow}(w'|w_n, ..., w_i, \texttt{and})$. For our running example, this translates to:

$LM_{\rightarrow}$(`<s>` I liked the sound and __)
$LM_{\leftarrow}$(__ and sound of the harpsichord . `</s>`)

with resulting top words: {*feel: 0.15, felt: 0.11, thought: 0.07, smell: 0.06, sounds: 0.05*} and {*sight: 0.16, sounds: 0.11, rhythm: 0.04, tone: 0.03, noise: 0.03*}.

The distributions predicted using the *and* pattern exhibit a much nicer behavior, and incorporate global context (resulting in sensing related substitutes) as well as local and syntactic information that resulting from the target word itself. Table 1 compares the context-only and symmetric-pattern substitutes for two senses of the word *sound*.

2.3 Representative Generation

To perform fuzzy clustering, we follow AI-KU and associate each instance with k representatives, but deviate in the way the representatives are generated. Specifically, each representative is a set of size 2ℓ , containing ℓ samples from the forward distribution and ℓ samples from the backward distribution. In the symmetric pattern case above, a plausible representative, assuming $\ell = 2$, would be: {*feel, sounds, sight, rhythm*} where two words were predicted by each side LM. In this work, we use $\ell = 4$ and $k = 20$.

2.4 Sense Clustering

After obtaining k representatives for each of the n word instances, we cluster the nk representatives into distinct senses and translate this hard-clustering of representatives into a probabilistic clustering of the originating instances.

Hard-clustering of representatives Let V be the vocabulary obtained from all the representatives. We associate each representative with a sparse $|V|$ dimensional bag-of-features vector, and arrange the representatives into a $nk \times |V|$ matrix M where each row corresponds to a representative. We now cluster M's rows into senses. We found it is beneficial to transform the matrix using TF-IDF. Treating each representative as a document, TF-IDF reduces the weight of uninformative words shared by many representatives. We use agglomerative clustering (cosine distance, average linkage) and induce a fixed number of clusters.[3] We use `sklearn` (Pedregosa et al., 2011) for both TF-IDF weighting and clustering.

Inducing soft clustering over instances After clustering the representatives, we induce a soft-clustering over the instances by associating each instance j to sense i based on the proportion of representatives of j that are assigned to cluster i.

2.5 Additional Processing

Lemmatization The WSI task is defined over lemmas, and some target words have morphological variability within a sense. This is especially common with verb tenses, e.g., "I **booked** a flight" and "I am **booking** a flight". As the conjunctive

---

[3] In this work, we use 7 clusters, which roughly matches the number of senses for each target word in the corpus. Dynamically selecting the number of clusters is left for future work. The effect of changing the number of clusters is explored in the supplementary material.

Context Only		Symmetric Pattern	
Forward dist.	Backward dist.	Forward dist.	Backward dist.
This is a ***sound*** idea, I like it.			
sad 0.02	bad 0.12	welcome 0.09	funny 0.10
great 0.02	good 0.09	practical 0.03	beautiful 0.05
huge 0.02	great 0.06	comprehensive 0.03	fun 0.04
very 0.02	wonderful 0.05	light 0.02	simple 0.04
lesson 0.02	nice 0.04	balanced 0.02	interesting 0.03
I liked the ***sound*** of the harpsichord			
idea 0.12	sounds 0.04	feel 0.15	sight 0.16
fact 0.07	version 0.03	felt 0.11	sounds 0.11
article 0.05	rhythm 0.03	thought 0.07	rhythm 0.04
guy 0.04	strings 0.03	smell 0.06	tone 0.03
concept 0.02	piece 0.03	sounds 0.05	noise 0.03

Table 1: Predicted substitutes for two senses of sound, for context-only and the symmetric-pattern approaches.

symmetric pattern favors morphologically-similar words, the resulting substitute vectors for these two sentences will differ, each of them agreeing with the tense of its source instance. To deal with this, we lemmatize the predictions made by the language model prior to adding them to the representatives. Such removal of morphological inflection is straightforward when using the word distributions but much less trivial when using raw LM state vectors, further motivating our choice of working with the word distributions. The substantial importance of the lemmatization is explored in the ablation experiments in the next section, as well as in the supplementary material.

Distribution cutoff and bias Low ranked LM prediction tend to become noisier. We thus consider only the top 50 word predicted by each LM, re-normalizing their probabilities to sum to one. Additionally, we ignore the final bias vector during prediction (words are predicted via $softmax(Wx)$ rather than $softmax(Wx + b)$). This removes unconditionally probable (frequent) words from the top LM predictions.

3 Experiments and Results

We evaluate our method on the SemEval 2013 Task 13 dataset (Jurgens and Klapaftis, 2013), containing 50 ambiguous words each with roughly 100 in-sentence instances, where each instance is soft-labeled with one or more WordNet senses.

Experiment Protocol Due to the stochastic nature of the algorithm, we repeat each experiment 30 times and report the mean scores together with the standard deviation.

Evaluation metrics We follow previous work (Wang et al., 2015; Komninos and Manandhar, 2016) and evaluate on two measures: ***Fuzzy Normalized Mutual Information (FNMI)*** and ***Fuzzy B-Cubed (FBC)*** as well as their geometric mean (**AVG**).

Systems We compare against three graphical-model based systems which, as far as we know, represent the current state of the art: **MCC-S** (Komninos and Manandhar, 2016), **Sense-Topic** (Wang et al., 2015) and **unimelb** (Lau et al., 2013). We also compare against the **AI-KU** system. Wang et al. also present a method for dataset enrichment that boosted their model performance. We didn't use the suggested methods and compare ourselves to the vanilla settings, but report the enrichment numbers as well.

Results Table 2 summarizes the results. Our system using symmetric patterns outperforms all other setups with an AVG score of 25.4, establishing a new state-of-the-art on the task.

Ablation and analysis We perform ablations to explore the contribution of the different components (Symmetric Patterns (SP), Lemmatization (LEM) and TF-IDF re-weighting). Figure (1) shows the results for the entire dataset (ALL, top), as well as broken-down by part-of-speech. All components are beneficial and are needed for obtaining the best performance in all cases. However, their relative importance differs across parts-of-speech. Adjectives gain the most from the use of the dynamic symmetric patterns, while nouns gain the least. For verbs, the lemmatization is

Model	FNMI	FBC	AVG
Original task dataset			
Ours	**11.26** ± 0.48	57.49 ± 0.23	**25.43** ± 0.48
MCC-S	7.62	55.6	20.58
Sense-Topic (SW)	7.14	55.4	19.89
Sense-Topic	6.96	53.5	19.30
AI-KU	6.5	39.0	15.92
unimelb	6.0	48.3	17.02
With data enrichment			
Sense-Topic (AAC)	9.39	**59.1**	23.56
Sense-Topic (AUC)	9.74	54.5	23.04

Table 2: Evaluation Results on the SemEval 2013 Task 13 Dataset. SW: Embeddings similarity based feature weighting. AAC: Extending instance sentences from their traced source. AUC: Adding similar sentences from the dataset originating corpus. We report our mean scores over 30 runs ± standard deviation

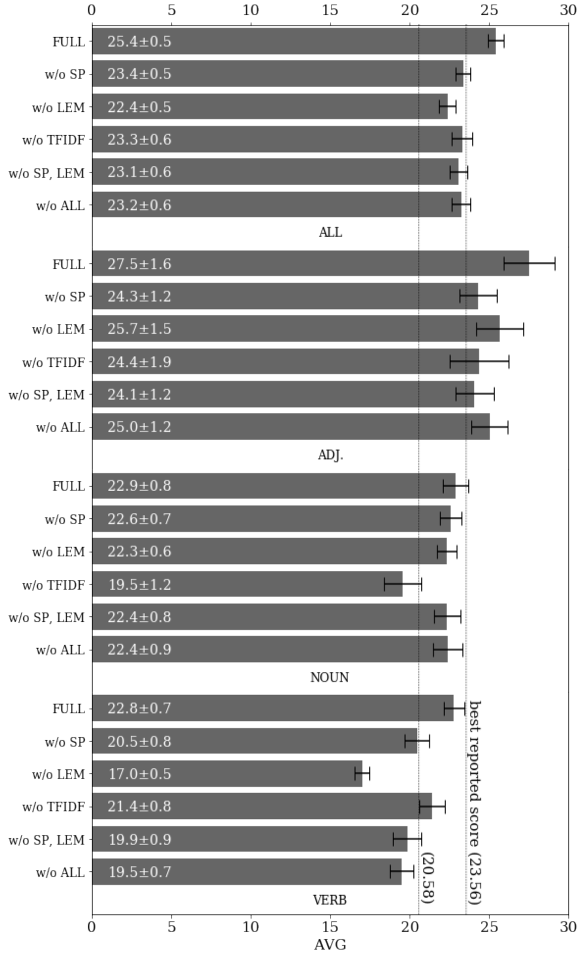

Figure 1: Ablation break down by part of speech, each part of speech was averaged across run. Bars are mean of means and error bars are standard deviations.

crucial for obtaining good performance, especially when symmetric patterns are used: using symmetric patterns without lemmatization, the mean score drops to 17.0. Lemmatization without symmetric patterns achieves a higher mean score of 20.5, while using both yields 22.8. Finally, for nouns it is the TF-IDF scoring that plays the biggest role.

4 Conclusions

We describe a simple and effective WSI method based on a neural biLM and a novel dynamic application of the *X and Y* symmetric pattern. The method substantially improves on the state-of-the-art. Our results provide further validation that RNN-based language models contain valuable semantic information.

The main novelty in our proposal is querying the neural LM in a creative way, in what we call *dynamic symmetric patterns*. We believe that the use of such dynamic symmetric patterns (or more generally *dynamic Hearst patterns*) will be beneficial to NLP tasks beyond WSI.

In contrast to previous work, we used discrete predicted word distributions rather than the continuous RNN states. This paid off by allowing us to inspect and debug the representation, as well to control it in a meaningful way by injecting linguistic knowledge in the form of lemmatization, and by distributional cutoff and TF-IDF re-weighing. We encourage others to consider using explicit, discrete representations when appropriate.

Acknowledgments The work was supported in part by the Israeli Science Foundation (grant number 1555/15 and the German Research Foundation via the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

Eneko Agirre and Aitor Soroa. 2007. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 7–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Osman Baskaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. Ai-ku: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 300–306.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

David Jurgens and Ioannis Klapaftis. 2013. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 290–299.

Alexandros Komninos and Suresh Manandhar. 2016. Structured generative models of continuous features for word sense induction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3577–3587.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056, Columbus, Ohio. Association for Computational Linguistics.

Jey Han Lau, Paul Cook, and Timothy Baldwin. 2013. unimelb: Topic modelling-based word sense induction. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 307–311.

Suresh Manandhar, Ioannis P Klapaftis, Dmitriy Dligach, and Sameer S Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 63–68. Association for Computational Linguistics.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 258–267.

Jing Wang, Mohit Bansal, Kevin Gimpel, Brian D Ziebart, and T Yu Clement. 2015. A sense-topic model for word sense induction with unsupervised data enrichment. *Transactions of the Association of Computational Linguistics*, 3(1):59–71.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. *arXiv preprint arXiv:1603.07012*.

Supplementary Material

Statistics of the SemEval 2013 Task 13 Dataset

SemEval 2013 Task 13 consists of 50 targets, each has a lemma and a part of speech (20 verbs, 20 nouns and 10 adjectives). We use the dataset only for evaluation. Most targets have around 100 labeled instances (sentences containing a usage of the target in its designated part of speech together with one or more WordNet senses assigned by human labeler). Exceptions are the targets of trace.n and book.v which have 37 and 22 labeled instances accordingly. Leaving out the two anomalous targets mentioned above we are left with 4605 instances from 48 targets: 19 verb, 19 noun and 10 adjective targets. We note that the small size of the dataset should make one cautious to draw quick conclusions, yet, our results seem to be consistent.

Effect of the Choice of Number of Clusters

An important statistic of the dataset is the number of senses per target. The average number of senses per target in the dataset is 6.94 (stdev:2.71). Breaking down by part of speech, the means and standard deviations of target senses are: verbs: 5.90 (±1.37), nouns: 7.32 (±2.21), adjectives: 7.11 (±3.54). In this work we follow this statistic and always look for 7 clusters. Figure 2 shows the accuracy as a function of the number of clusters. While 7 clusters indeed produces the highest scores, all numbers in the range 4 to 15 produce state-of-the-art results. We leave the selection of per-instance number of clusters to future work.

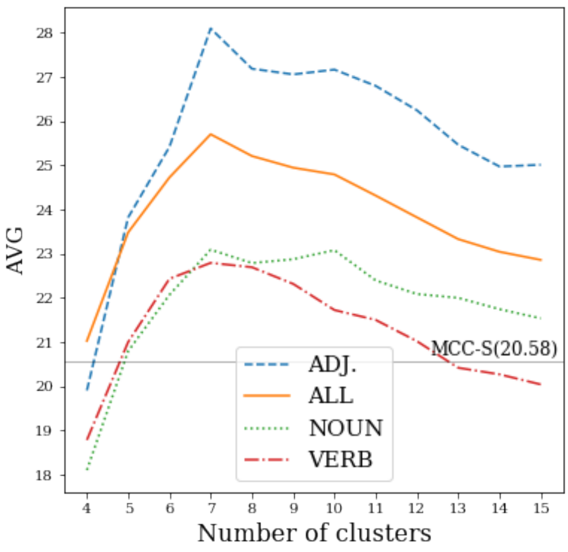

Figure 2: AVG score by number of clusters.

Figure 2 also tells us our system is better at inducing senses for adjectives, at least according to task score.

The Importance of Lemmatization

The ablation results in the paper indicate that for verbs, using symmetric patterns without lemmatization yields poor results. We present the analysis the motivated our use of lemmatization. Consider the samples from the biLM with and without symmetric patterns, for the instance *It was when I was a high-school student that I* ***became*** *convinced of this fact for the first time.*

fw LM, no SP:	didn, write, 'd, learnt, start
bw LM, no SP:	seem, be, grow, be, be
fw LM, with SP:	went, got, started, wasn, loved
bw LM, with SP:	1990s, decade, 1980s, afterwards, changed

Another sentence, in another tense: *The issue will* ***become*** *more pressing as an estimated 40,000 to 50,000 Chinese, mostly unskilled, come to settle each year.*

fw LM, no SP:	be, be, remain, likely, be
bw LM, no SP:	becoming, grown becoming, much, becomes
fw LM, with SP:	remains, remain, which, continue, how
bw LM, with SP:	rising, overseas, booming, abroad, expanded

When using the symmetric patterns, the predicted verbs tend to share the tense of the target word.

This results in targets of different tenses having nearly distinct distributions, even when the targets share the same sense, splitting the single sense cluster to two (or more) tense clusters. We quantify this intuition by computing the correlation between tense and induced clusters (senses), as given by the Normalized Mutual Information (NMI). We measure NMI between verb instance tense in sentence and their most probable induced cluster in the different settings, as well as the NMI of the verb instances and the gold clusters. Table 3 summarize the results. We see that in the gold clusters there is indeed very little correlation (0.15) between the the tense and the sense. When using SP but not lemmatization (w/o LEM), the correlation is substantially higher (0.67). When not using either lemmatization of SP (w/o LEM and SP) the correlation is 0.27, much closer to the gold one. Performing explicit lemmatization naturally reduces the correlation with tense, and using the full model (Final model) results in a correlation to 0.22, close to the gold number of 0.15.

Some Failure Modes of Dynamic Symmetric Patterns

While the use of dynamic symmetric patterns improves performance and generally produces good substitutes for contextualized words, we also identify some failure modes and unexpected behavior.

Common phrases involving conjunctions Some target words have a strong prior to appear in common phrases involving a conjunction, causing the strong local pattern to override context-based hints. For example, when the LM is asked to complete ... *state and* __, its prior on *church* makes it a very probable completion, regardless of context and sense. This phenomena motivated our use TF-IDF for weighing of too common words. Relatedly, a common completion for symmetric patterns is the word *then*, as *and then* is a very common phrase. This completion even ignores the target word and could be troublesome if a global, cross-lemma, clustering is attempted.

Multi word phrases substitutes Sometime the LM does interpret the *and* as a trigger for a symmetric relation, but on a chunk extending beyond the target word. For example, when presented with the query *The human heart not only makes heart sounds and* __, the forward LM predicted in its top twenty suggestions the word *muscle*, followed by a next-word prediction of *movements*. That is, the symmetry extends beyond "sounds" to the phrase "heart sounds" which could be substitutes by "muscle movements". We didn't specifically address this in the current work, but note that restricting the prediction to agree with the target word on part-of-speech and plurality may help in mitigating this. Furthermore, this suggests an exciting direction for moving from single words towards handling of multi-word units.

Settings	NMI (mean ± STD)
Gold labels	0.15 ± 0.07
Final model	0.22 ± 0.12
w/o SP	0.19 ± 0.08
w/o TFIDF	0.18 ± 0.07
w/o LEM	**0.67± 0.12**
w/o LEM and SP	0.26 ± 0.09
w/o ALL	0.24 ± 0.08

Table 3: Correlation between tense and sense. NMI is averaged on all verbs, using best matching sense. SP: Symmetric Patterns, LEM: Lemmatizing predictions, ALL: LEM, SP, TFIDF. The bold line show symmetric patterns without lemmatization excessively correlates tense and sense and provides additional validation to our hypothesis, suggesting its essential to lemmatizate when symmetric patterns are used.

InferLite: Simple Universal Sentence Representations from Natural Language Inference Data

Jamie Ryan Kiros
Google Brain Toronto
`kiros@google.com`

William Chan
Google Brain Toronto
`williamchan@google.com`

Abstract

Natural language inference has been shown to be an effective supervised task for learning generic sentence embeddings. In order to better understand the components that lead to effective representations, we propose a lightweight version of InferSent (Conneau et al., 2017), called InferLite, that does not use any recurrent layers and operates on a collection of pre-trained word embeddings. We show that a simple instance of our model that makes no use of context, word ordering or position can still obtain competitive performance on the majority of downstream prediction tasks, with most performance gaps being filled by adding local contextual information through temporal convolutions. Our models can be trained in under 1 hour on a single GPU and allows for fast inference of new representations. Finally we describe a semantic hashing layer that allows our model to learn generic binary codes for sentences.

1 Introduction

Distributed representations of words have become immensely successful as the building blocks for deep neural networks applied to a wide range of natural language processing tasks (Pennington et al., 2014). Learning representations of sentences, however, has largely been done in a task-dependent way. In recent years, a growing body of research has emerged for learning general purpose sentence embeddings. These methods aim to learn a universal encoding function that can map arbitrary sentences into vectors which can then be applied to downstream prediction tasks without fine-tuning. Much of the motivation behind this work is to mimic the successful use of feature transfer in computer vision.

Recently, Conneau et al. (2017) showed that a bidirectional LSTM with max pooling trained to perform Natural Language Inference (NLI), called InferSent, outperforms several other encoding functions on a suite of downstream prediction tasks. This method could match or outperform existing models that learns generic embeddings in an unsupervised setting, often requiring several days or weeks to train (Kiros et al., 2015). However, a better understanding of what properties induce a useful generic embedding remains illusive.

In this work we propose a lightweight version of InferSent, called InferLite. InferLite deviates from InferSent in that it does not use any recurrent connections and can generalize to multiple pre-trained word embeddings. Our method uses a controller to dynamically weight embeddings for each word followed by max pooling over components to obtain the final sentence representation. Despite its simplicity, our method obtains performances on par with InferSent (Conneau et al., 2017) when using Glove representations (Pennington et al., 2014) as the source of pre-trained word vectors. To our surprise, the majority of evaluations can be done competitively without any notion of context, word ordering or position. For tasks where this is useful, much of the performance gap can be made up through a stack of convolutional layers to incorporate local context. Finally, we describe a semantic hashing layer that allows our model to be extended to learning generic binary vectors. The final result is a method that is both fast at training and inference and offers a strong baseline for future research on general purpose embeddings.

1.1 Why learn lightweight encoders?

Our proposed model naturally raises a question: why consider lightweight sentence encoders? If a generic encoder only needs to be trained once, why would training times be relevant? We argue our direction is important for two reasons. One is inference speed. With a lightweight encoder, we can encode millions of sentences efficiently without requiring extensive computational resources. The appendix includes inference speeds of our models. The second, perhaps more importantly, is to gain a better understanding of what prop-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4868–4874
Brussels, Belgium, October 31 - November 4, 2018.

erties lead to high quality generic embeddings. When models take several days or weeks to train, an ablation analysis becomes prohibitively costly. Since our models can be trained quickly, it allows for a more extensive analysis of architectural and data necessities. Moreover, we include an ablation study in the appendix that shows even innocent or seemingly irrelevant model decisions can have a drastic effect on performance. Such observations could not be observed when models take orders of magnitude longer to train.

2 Related Work

A large body of work on distributional semantics have considered encoding phrase and sentence meaning into vectors e.g. (Mitchell and Lapata, 2008; Grefenstette et al., 2013; Paperno et al., 2014). The first attempt at using neural networks for learning generic sentence embeddings was Kiros et al. (2015), who proposed a sequence-to-sequence extension of the skip-gram model but applied at the sentence level. This method was taught to encode a sentence and predict its neighbours, harnessing a large collection of books for training (Zhu et al., 2015). A similar approach, FastSent, was proposed by (Hill et al., 2016) which replaced the RNN encoder of skip-thoughts with word embedding summation. Methods using RNN encoders tend to perform poorly on STS evaluations, as shown by Wieting et al. (2015). Arora et al. (2017) showed a simple weighted bag of words with the first principal component subtracted, can be competitive on many sentencing encoding tasks.

Attempts to learn generic encoders with discriminative objectives were considered by Nie et al. (2017) and Logeswaran and Lee (2018) who replaced the decoder of skip-thoughts with classification tasks based on discourse relations and prediction of target sentences from an encoded candidate. All of the above methods relied on a large corpus of unlabelled data. Conneau et al. (2017) showed that similar or improved performance can be obtained using NLI datasets as a source of supervisory information. The state of the art sentence encoders utilize multi-task learning (Subramanian et al., 2018) by training an encoder to simultaneously do well on a collection of tasks such as NLI, next sentence prediction and translation.

The use of gating for selecting word representations has been considered in previous work. Yang et al. (2017) introduced a method for choosing between word and character embeddings while Kiela et al. (2018) describe a contextual gating method for word embedding selection. Gating has also been widely applied to multimodal fusion (Arevalo et al., 2017; Wang et al., 2018b; Kiros et al., 2018).

Feature	dataset	dim	method
Glove	Common Crawl	300	
News	Google News	500	CBOW
Query	Google Search	800	CBOW

Table 1: Comparison of word representations used.

Our work is also related to recent methods that induce contextualized word representations (McCann et al., 2017; Peters et al., 2018) as well as pre-training language models for task-dependent fine-tuning (Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018). We differ from these approaches in that we aim to infer a transferable sentence vector without any additional fine-tuning.

3 Method

Our method operates on a collection of pre-trained word representations and is then trained on the concatenation of SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018) datasets as in Conneau et al. (2017). Table 1 summarizes the properties of the embeddings we consider. At a high level, our method takes as input a collection of embeddings for each word and learns a gated controller to decide how to weight each representation. After encoding each word in a sentence, the sentence embedding is obtained by max pooling the transformed word representations. Unlike Subramanian et al. (2018), which learn a shared encoder in a multi-task setting, we instead fix the prediction task to NLI but use embeddings obtained from alternative tasks. Figure 1 illustrates our model.

We begin by defining notation. Suppose we are given a sentence of words $S = w^1, \ldots, w^T$ which we would like to encode into a vector. Let K be the number of embedding types (e.g. Glove, News, Query) and let E^k denote the word embedding matrix for type k. Define $E^c = [E^1; \ldots; E^K]$ to be the concatenation of word embedding matrices of all K types.

We break our model description into four modules: Encoder, Controller, Fusion and Reduction. In the appendix we include an ablation study that analyzes the effect of our design choices.

Encoder. The encoder computes $M + 1$ layers $H_0^k, \ldots, H_M^k$ for $k = 1, \ldots, K$ embedding types.

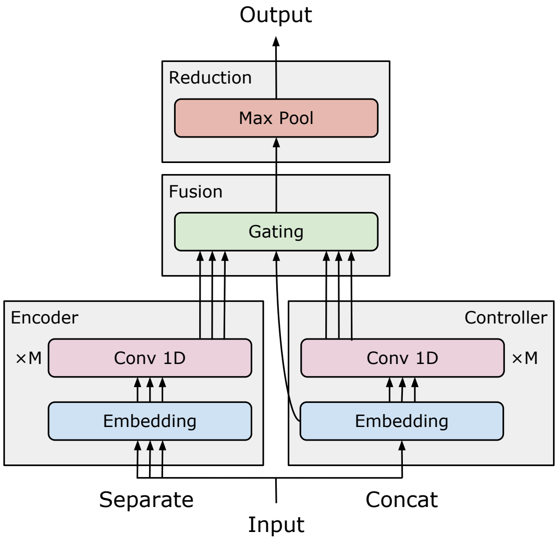

Figure 1: Illustration of the InferLite model. Separate and Concat refer to the embedding types used.

The first layer is computed as:

$$H_0^k = \phi_{0,h}(W_{0,h}^k E^k + b_{0,h}^k) \tag{1}$$

where $W_{0,g}^k E^k$ is a time distributed matrix multiply [1] and $\phi_{0,h}$ is the activation function. Each subsequent layer is given by:

$$H_i^k = \phi_{i,h}(W_{i,h}^k * H_{i-1}^k + b_{i,h}^k) \tag{2}$$

where $*$ denotes the 1-D convolution operator that preserves dimensions. Note that if the convolutional filter length is 1, the model reduces to a bag-of-words encoder. We use ReLU activation functions for $\phi_{i,h}$ where $i = 0, \ldots, M-1$ and a tanh activation for the last layer $\phi_{M,h}$.

Controller. The controller first computes a shared layer G_0^c along with M heads $G_1^k, \ldots, G_M^k$ for $k = 1, \ldots, K$ embedding types. The first layer is computed as:

$$G_0^c = \phi_{0,g}(W_{0,g}^c E^c + b_{0,g}^c) \tag{3}$$

where $W_{0,g}^c E^c$ is a time distributed matrix multiply and $\phi_{0,g}$ is the activation function. Define $G_0^k = G_0^c, k = 1, \ldots, K$. Each subsequent layer is given by:

$$G_i^k = \phi_{i,g}(W_{i,g}^k * G_{i-1}^k + b_{i,g}^k) \tag{4}$$

We use ReLU activation functions for $\phi_{i,g}$ where $i = 0, \ldots, M-1$ and a sigmoid activation for the last layer $\phi_{M,g}$.

Fusion. The fusion layer combines the encoder and controller layers as follows:

$$\begin{aligned} F' &= \left(\sum_{k=1}^{K} H_M^k \odot G_M^k \right) + G_0^c &(5) \\ F &= \phi_f(W_f F' + b_f) &(6) \end{aligned}$$

where $\odot$ denotes a component-wise product, $W_f F'$ is a time distributed matrix multiply, ϕ_f is a ReLU activation function and G_0^c is added as a skip connection. In the appendix we demonstrate that the added skip connection is crucial to the success of the model.

Reduction. The final reduction operation simply applies max pooling across tokens:

$$s = \text{maxpool}\{F\}_T \tag{7}$$

resulting in a sentence vector s. This resulting vector corresponds to the embedding for which we evaluate all downstream tasks with.

For training on NLI, we follow existing work and compute the concatenation of the embeddings of premise and hypothesis sentences along with their componentwise and absolute difference (Conneau et al., 2017). This joint vector is fed into a 2 hidden layer feedforward network with ReLU activations, followed by a softmax layer to predict whether the sentence pairs are neutral, entailed or contradictory. After training on NLI, the weights of the model are frozen and used for encoding new sentences.

3.1 Relationship to other work

Our model shares similarities to two other works, namely the gated convolutional layers from Dauphin et al. (2016); Gehring et al. (2017) and van den Oord et al. (2016). There are three main differences: 1) we generalize to multiple embedding types 2) we only apply gating at the end of the last layer as a way of weighting all embedding types (instead of each layer) and 3) we use a skip connection from the controller's transformed input to the fusion layer. We note that our encoder module can be reduced to the gated convolutional encoder in van den Oord et al. (2016) if we use one embedding type, remove the time distributed layers and only use a single convolutional layer.

3.2 Semantic hashing

We can augment a semantic hashing (Salakhutdinov and Hinton, 2009) layer to InferLite as a way of learning binary codes for sentences. Binary codes allow for efficient storage and retrieval

---

[1] Sometimes referred to as a "translation layer" (see https://github.com/Smerity/keras_snli).

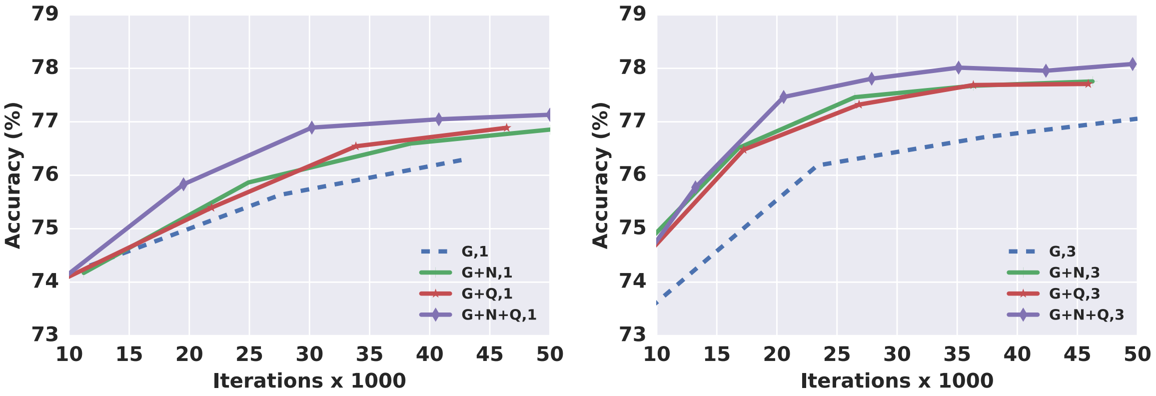

(a) NLI: no context (Conv length 1).

(b) NLI: with context (Conv length 3).

Figure 2: NLI accuracy using models that (a) have no context (convolution length of 1) and b) local context (convolution length of 3). Performance is reported on the concatenation of SNLI and MultiNLI development sets. **G** stands for Glove, **N** stands for News and **Q** stands for Query embeddings.

over massive corpora. To do this, we append the following layer:

$$h(s) = \sigma\left(\frac{\mathrm{LN}(W_x s + b_x)}{\tau}\right) \quad (8)$$

where LN is Layer Normalization (Ba et al., 2016), σ is the sigmoid activation and τ is a temperature hyperparameter. We initialize $\tau = 1$ at the beginning of training and exponentially decay τ towards 0 over the course of training. At inference time, we threshold at 0.5 to obtain codes. We found Layer Normalization was important for obtaining good codes as otherwise many dead units would form. In the appendix we include downstream performance results for 256, 1024 and 4096-bit codes. The combination of fast inference and efficient storage allows InferLite to be an effective generic encoder for large-scale retrieval and similarity search.

4 Experiments

We use the SentEval toolkit (Conneau and Kiela, 2018) for evaluating our sentence embeddings. All of our models are trained to optimize performance on the concatenation of SNLI and MultiNLI, using the concatenated development sets for early stopping. We use 4096-dimensional embeddings as in Conneau et al. (2017). We consider encoders that use convolutional filters of length 1 (no context) or length 3 (local context), with a stack of $M = 3$ convolutional layers. All word embeddings are pre-trained, normalized to unit length and held fixed during training. Full hyperparameter details are included in the appendix, including an ablation study comparing the effect of the choice of M.

We first analyze performance of our model on NLI prior to evaluating our models on downstream tasks. Figure 2 shows development set accuracy on NLI for models with and without context, using various feature combinations. Here we observe that a) using local context improves NLI performance and b) adding additional embedding types leads to improved performance.

Tables 2 and 3 show results on downstream evaluation tasks. Here several observations can be made. First note the effectiveness of the basic (glove,1) model, which is essentially a deep bag-of-unigram encoder. We also observe our models outperform all previous bag of words baselines. Next we observe that adding local context helps significantly on MR, CR, SST2 and TREC tasks. Furthermore, fusing embeddings from query and news models matches or improves performance over a glove-only model on 12 out of 15 tasks. Our (glove+news+query,3) model is best on 5 tasks and is a generally strong performer across all evaluations. Finally observe that our models significantly improves over previous work on STS tasks.

Next we compare training times of our models to previous work. All of our models can be trained in one GPU hour or less. QuickThoughts and InferSent can be trained on the order of a day while Multitask requires 1 week of training. This demonstrates the trade-off of these approaches.

In the appendix we include results from several other experiments including COCO image-sentence retrieval, downstream performance of InferLite with semantic hashing and results on 10 probing tasks introduced in Conneau et al. (2018). We also do an extensive ablation study

Model	MR	CR	SUBJ	MPQA	SST2	TREC	MRPC	time (h)
Glove BOW (Conneau et al., 2017)	78.7	78.5	91.6	87.6	79.8	83.6	72.1/80.9	0
USE-D (Cer et al., 2018)	74.5	81.0	92.7	85.4	77.6	91.2		
ST-LN (Ba et al., 2016)	79.4	83.1	93.7	89.3	82.9	88.4		~ 720
DisSent (Nie et al., 2017)	80.1	84.9	93.6	90.1	84.1	93.6	75.0/	
InferSent (Conneau et al., 2017)	81.1	86.3	92.4	90.2	84.6	88.2	76.2/83.1	< 24
USE-T (Cer et al., 2018)	81.4	87.4	93.9	87.0	85.4	92.5		
QuickThoughts (Logeswaran and Lee, 2018)	82.4	86.0	**94.8**	90.2	**87.6**	92.4	76.9/84.0	~ 24
Multitask (Subramanian et al., 2018)	**82.5**	**87.7**	94.0	**90.9**	83.2	**93.0**	**78.6/84.4**	~ 168
glove,1	79.6	82.2	92.0	89.5	83.0	88.2	75.5/82.7	~ 0.3
glove+news,1	79.0	82.7	92.1	89.8	83.7	89.2	76.9/83.7	~ 0.5
glove+query,1	79.0	83.2	92.2	89.6	83.3	89.4	75.8/83.0	~ 0.5
glove+news+query,1	78.8	82.2	92.0	89.6	83.0	89.2	76.7/83.5	~ 0.7
glove,3	80.9	84.1	92.4	89.6	85.8	90.0	76.5/83.4	~ 0.5
glove+news,3	80.4	84.8	91.9	89.7	86.3	89.8	77.0/83.9	~ 0.7
glove+query,3	80.1	85.6	92.2	89.4	84.7	91.0	76.4/83.3	~ 0.7
glove+news+query,3	80.4	85.6	92.2	89.9	85.0	91.2	76.1/83.3	~ 1

Table 2: Comparison of embedding methods on downstream evaluations. Each set of results is a) bag-of-words b) RNN and Transformer c) ours, filter length 1 and d) ours, filter length 3. Last column is training time in hours.

Model	SICK-R	SICK-E	STSB	STS12	STS13	STS14	STS15	STS16
Glove BOW (Conneau et al., 2017)	80.0	78.6		52.5	42.3	54.2	52.7	
GloVe + WR (Arora et al., 2017)	86.0	84.6		56.2	56.6	68.5	71.1	
ST-LN (Ba et al., 2016)	85.8	79.5		30.8	24.8	31.4	31.0	
InferSent (Conneau et al., 2017)	88.4	86.3	75.8/75.5	59.2	58.9	69.6	71.3	71.4
Multitask (Subramanian et al., 2018)	**88.8**	**87.8**	**78.9/78.6**	60.6	54.7	65.8	74.2	66.4
glove,1	88.3	85.9	78.1/78.0	62.4	60.4	71.6	**74.6**	70.3
glove+news,1	88.5	86.7	78.0/78.1	63.0	58.8	71.2	74.2	70.2
glove+query,1	88.5	86.0	77.1/77.1	63.1	56.8	70.9	74.1	70.3
glove+news+query,1	88.6	85.9	77.7/78.0	**63.9**	58.1	70.9	73.6	69.8
glove,3	88.1	85.5	78.4/78.3	61.9	**61.3**	71.7	74.5	71.2
glove+news,3	88.5	86.6	77.5/77.4	61.7	59.5	71.0	73.9	**71.6**
glove+query,3	88.6	86.5	78.1/78.3	61.8	**61.3**	**71.8**	74.3	70.1
glove+news+query,3	88.7	87.2	77.1/77.1	63.1	58.6	71.7	73.8	70.1

Table 3: Comparison of embedding methods on downstream evaluations. Each sets of results are a) bag-of-words b) RNN encoders c) ours, filter length 1 d) ours, filter length 3. Best results bolded. Our best results underlined.

of model components and illustrate gate activation values qualitatively for sentences from the (glove+news+query,3) model.

4.1 Limitations

We also experimented with additional embedding types, including Picturebook (Kiros et al., 2018), knowledge graph and neural machine translation based embeddings. While adding these embeddings improved performance on NLI, they did not lead to any performance gains on downstream tasks. This is in contrast to Subramanian et al. (2018) who showed adding additional tasks in a multi-task objective led to better downstream performance. This demonstrates the limitations of solely using NLI as an objective, even if we transfer embeddings from additional tasks.

5 Future Work

In future work, we would like to explore using contextualized word embeddings, such as CoVe (McCann et al., 2017) and ELMo (Peters et al., 2018), as input to our models as opposed to non-contextualized representations. We also intend to evaluate on additional benchmark tasks such as GLUE (Wang et al., 2018a), explore using the learned word representations as contextualized embeddings and perform downstream fine-tuning.

Acknowledgments

The authors would like to thank Geoffrey Hinton, George Dahl and the EMNLP anonomous reviewers for their valuable feedback.

References

John Arevalo, Thamar Solorio, Manuel Montes y Gomez, and Fabio A. Gonzalez. 2017. Gated Multimodal Units for Information Fusion. In *arXiv:1702.01992*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *ICLR*.

Jimmy Ba, Jamie Kiros, and Geoffrey Hinton. 2016. Layer Normalization. In *arXiv:1607.06450*.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal Sentence Encoder. In *arXiv:1803.11175*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. In *arXiv:1312.3005*.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An Evaluation Toolkit for Universal Sentence Representations. In *arXiv:1803.05449*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *EMNLP*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. In *arXiv:1612.08083*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *arXiv:1705.03122*.

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *arXiv:1301.6939*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *NAACL*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Context-Attentive Embeddings for Improved Sentence Representations. In *arXiv:1804.07983*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Jamie Kiros, William Chan, and Geoffrey Hinton. 2018. Illustrative Language Understanding: Large-Scale Visual Grounding with Image Search. In *ACL*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-Thought Vectors. In *NIPS*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, Lawrence Zitnick, and Piotr Dollr. 2014. Microsoft COCO: Common Objects in Context. In *ECCV*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *ICLR*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL: HLT*.

Allen Nie, Erin D Bennett, and Noah D Goodman. 2017. Dissent: Sentence representation learning from explicit discourse relations. In *arXiv:1710.04334*.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. 2016. Conditional image generation with pixelcnn decoders. In *NIPS*.

Denis Paperno, Marco Baroni, et al. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *unpublished*.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. In *International Journal of Approximate Reasoning*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. In *JMLR*.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *ICLR*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018a. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *arXiv:1804.07461*.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2018b. Learning Multimodal Word Representation via Dynamic Fusion Methods. In *AAAI*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. In *arXiv:1511.08198*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *NAACL*.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. 2017. Words or Characters? Fine-grained Gating for Reading Comprehension. In *ICLR*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*.

Similarity-Based Reconstruction Loss for Meaning Representation

Olga Kovaleva, Anna Rumshisky, Alexey Romanov
Department of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854
`{okovaleva,arum,aromanov}@cs.uml.edu`

Abstract

This paper addresses the problem of representation learning. Using an autoencoder framework, we propose and evaluate several loss functions that can be used as an alternative to the commonly used cross-entropy reconstruction loss. The proposed loss functions use similarities between words in the embedding space, and can be used to train any neural model for text generation. We show that the introduced loss functions amplify semantic diversity of reconstructed sentences, while preserving the original meaning of the input. We test the derived autoencoder-generated representations on paraphrase detection and language inference tasks and demonstrate performance improvement compared to the traditional cross-entropy loss.

1 Introduction

Natural language processing (NLP) tasks that use an encoder-decoder architecture tend to rely on the cross-entropy reconstruction loss to generate the target output. A great majority of deep learning models used at present for state-of-the-art machine translation, question answering, summarization, and dialogue generation employ this type of architecture.

The standard cross-entropy loss penalizes the model whenever it fails to produce the exact word from the ground truth data used for training. However, in many NLP tasks that deal with generating text from semantic representation, recovering the exact word is not necessarily optimal, and often generating a near-synonym or just a semantically close word is nearly as good or even better from the point of view of model performance. Consider a situation when a decoder model generates a word by sampling from a softmax over the vocabulary-sized final layer to produce an output. Since cross-entropy loss forces a model to generate the exact words corresponding to those in the input text, the model will be penalized when semantically close but distinct outputs are generated. This is clearly undesirable in many cases when the exact output is not required.

In this paper, we introduce and experiment with a series of distance-based reconstruction losses. Using an auto-encoder derived representation of sentence meaning, we test their impact on model performance in several tasks that require building a semantic representation, including paraphrase detection and entailment / inference. We show that the loss functions that take into account distributional similarity between the word embeddings of the generated output and the ground truth tokens lead to a substantial improvement in performance on such tasks in an unsupervised setting.

2 Related Work

The encoder-decoder setting was first used in deep learning by Sutskever et al. (2014) and has been successfully adapted to a problem of representation learning since then. To date, numerous approaches based on the encoder-decoder idea have been suggested for unsupervised feature extraction from textual data.

Cer et al. (2018) modify the Transformer architecture (Vaswani et al., 2017) originally suggested for machine translation to produce sentence embeddings that target transfer learning to other NLP tasks. Arora et al. (2016) claim that sentence representation as simple weighted averaging of word vectors beats more sophisticated recurrent network-based models. McCann et al. (2017) show that adding machine translation-learned vectors to models designed for other NLP tasks improves their performance. Nangia et al. (2017) in RepEval-2017 report that in-sentence attention and biLSTM-based models extract represen-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4875–4880
Brussels, Belgium, October 31 - November 4, 2018.

tation of meaning from text reasonably well. Logeswaran and Lee (2018) and Kiros et al. (2015) change the problem of learning sentence representations to a classification task for predicting context sentences. Subramanian et al. (2018) demonstrate that sharing the same sentence encoder across different tasks leads to performance improvements.

All the listed works, however, propose methods that either develop task-specific architectures, or use large corpora of labeled data to learn embeddings at a sentence level. Unlike the mentioned papers, the simple modification we propose does not require data annotation and can be used with any state-of-the-art neural models for text generation.

Surprisingly, we have not found other work that uses the proposed idea despite its simplicity.

3 Experiments

The objective of a classic autoencoder is to minimize the difference between the given input $\mathbf{X}$ and the reconstructed output $\hat{\mathbf{X}}$.

$$\mathcal{L}(\mathbf{X}, g(f(\mathbf{X}))) \tag{1}$$

where f is the encoder function and g is the decoder function. We propose and compare several modifications of distance-based losses, that apply different penalties to the model depending on the similarity of the produced words to the targets in the embedding space.

- *Weighted similarity loss*

$$\mathcal{L} = -\sum_{i=1}^{V} \text{sim}(y_t, y_i) p_i \tag{2}$$

where p_i is the softmax probability over vocabulary size, $-1 \leq \text{sim} \leq 1$ is the similarity between the tokens embeddings vectors, y_t and y_i are the ground-truth token and the predicted token, respectively, and V is the total vocabulary size. Intuitively, this loss encourages the model to produce high probabilities for words that are close to the target word. In the present experiments, we use cosine as the similarity measure.

- *Weighted cross-entropy loss*

$$\mathcal{L} = -\sum_{i=1}^{V} \text{sim}(y_t, y_i) \log p_i \tag{3}$$

Here the optimization function can be seen as the "weighted" cross-entropy, meaning that every ground-truth token is represented with similarities to other words in the vocabulary rather than with a traditional one-hot-encoding scheme. The schematic illustration of the true label encoding for the weighted similarity and weighted cross-entropy loss functions is shown in Figure 1 (right).

- *Soft label loss*

$$\mathcal{L} = -\sum_{i=1}^{V} y_i^* logp_i \tag{4}$$

This cost function is similar to the previous one in terms of true label y_i representation: we encode ground-truth tokens as their similarities across the vocabulary, but we consider only the top N closest words in the vocabulary and normalize the similarities so that they add up to one $\sum_{i=1}^{V} y_i^* = 1$. Essentially, the loss function can be interpreted as cross-entropy with soft targets. We vary N from 3 to 10 in our experiments. We also exclude common English stop-words from soft target encoding, i.e. we apply a regular cross entropy loss for reconstructing of these words. The schematic illustration is given in Figure 1 (center).

$$y_i^* = \begin{cases} \frac{\text{sim}(y_t, y_i)}{\sum_{j=1}^{N} \text{sim}(y_t, y_j)}, & y_i \in \text{top N} \\ 0, & y_i \notin \text{top N} \end{cases} \tag{5}$$

We use pre-trained fastText (Bojanowski et al., 2016) word vectors to compute similarities between words. Note that the more recently proposed ELMo embeddings (Peters et al., 2018), for example, can not be used in our case, since they are context-dependent, which means that similarities between individual words can not be precomputed.

To find out how the proposed loss functions affect the quality of the derived representations, we trained several autoencoder models using the regular cross-entropy, as well as the three variants of the similarity-based reconstruction loss described above. In these experiments, we use the Yelp restaurant reviews dataset (Shen et al., 2017). This dataset was originally introduced for a sentiment classification task and consists of 600K sentences.

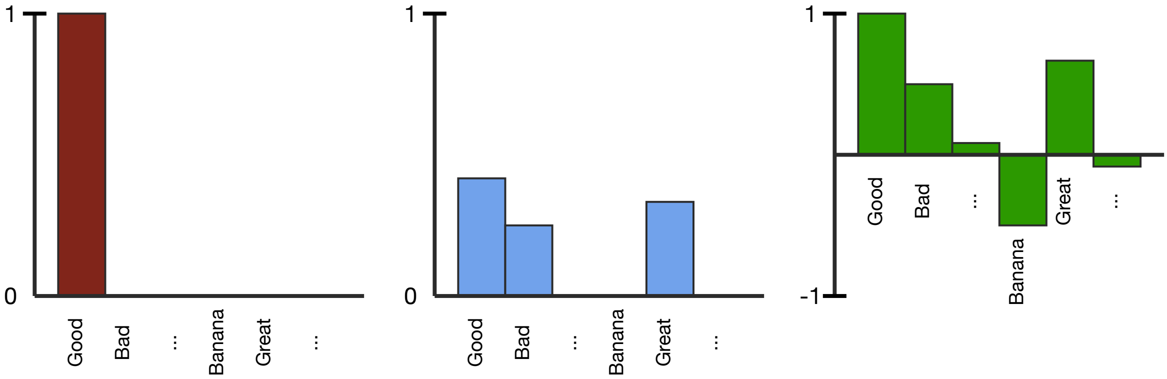

Figure 1: Schematic illustration of true-label encoding using the standard cross-entropy loss (left), soft label loss for $N = 3$ (center) and weighted similarity/weighted cross-entropy loss (right). All the three examples "good", "great" and "bad" are close in the embedding space, since they appear in similar contexts. Note that all the soft labels add up to 1, while weighted similarity labels for the third loss can vary in the range from -1 to 1.

Model	MSRP		SNLI	SICK-E
	F1	Acc	Acc	Acc
Cross-entropy (vanilla AE)	79.0	66.9	44.8	56.8
Soft label, $N = 3$	77.6	67.1	57.8	71.8
Soft label, $N = 5$	79.1	67.3	57.2	71.6
Soft label, $N = 10$	77.9	66.5	57.9	**72.4**
Weighted similarity	77.5	65.6	**69.1**	56.6
Weighted cross-entropy	**79.4**	**68.2**	57.2	70.2

Table 1: Results of transfer learning tasks performance of the proposed autoencoder models. All the models are trained on the Yelp reviews dataset with the use of fastText pre-trained word embeddings.

Our autoencoder model is implemented using the PyTorch deep learning framework (Paszke et al., 2017). In our architecture, both the encoder and the decoder are implemented as single layer LSTMs, each with the hidden size of 256 units. We divide our dataset into train/dev/test splits in 70/10/20 ratio. The resulting vocabulary size of the training dataset is 9.5K tokens. For our training, we use the Adam optimizer (Kingma and Ba, 2014) with the learning rate that varies depending on the tested loss between 0.001 and 0.0001.

We test our learned representations using the SentEval toolkit (Conneau et al., 2017). SentEval is an open-source Python library for evaluating sentence embeddings on a diverse set of language tasks. This toolkit provides a cluster of downstream tasks taken from various competitions such as SemEval as well as a set of probing tasks. In current paper, we focus on the paraphrase detection task using the Microsoft Research Paraphrase Corpus (MSRP) (Dolan et al., 2004), as well as the inference/entailment tasks using the Stanford Natural Language Inference corpus (SNLI) (Bowman et al., 2015) and the SICK-Entailment dataset from SemEval-2014 (Marelli et al., 2014). We selected these tasks because they seem to be likely to benefit from capturing word-level semantic similarity. Table 1 shows the scores averaged over three (3) runs.

4 Discussion

We find that almost all of the proposed loss functions outperform the vanilla autoencoder trained with cross-entropy on all three tasks (see Table 1). The only exception is the weighted similarity loss function. Compared to the logarithm-based losses, this loss applies softer penalties when the ground-truth tokens are predicted to have lower probabilities. We conclude that the non-linearity introduced by a logarithm function contributes to more efficient training.

Among the models we tested, the best scores were achieved by the weighted cross-entropy loss for MSRP (68.2%), the weighted similarity loss for SNLI (69.1%) and by the soft label loss for

Configuration	Autoencoder outputs	
	Input sentence	Reconstructed sentence
Cross-entropy	you can trust this business	you can trust this business
Soft label, $N = 3$	the taste was so good	the flavor was so good
Soft label, $N = 5$	her tone was incredibly rude	her attitude was incredibly unprofessional
Soft label, $N = 10$	a very nice spot for a quiet lunch	a very nice slot for a tranquil lunchtime
Weighted similarity	once again the staff were wonderful	that so that service and great
Weighted cross-entropy	great breakfast option	great food place

Table 2: Sample autoencoder reconstruction outputs for the tested loss configurations.

SICK-E (72.4%). We observe that for the paraphrase task, all the soft label losses behaved similarly, while for the inference/entailment, increasing the number of neighbors improved performance.

In order to better understand how different modifications of the soft label loss affected model performance on transfer tasks, we conducted some additional experiments. Specifically, we investigated the effects of (a) varying the number N of word neighbors used to compute the loss function, and (b) removing the normalization factor by getting rid of the denominator in Eq. 5 (i.e. soft label similarities no longer sum up to 1). Note that when $N = 1$, the soft label loss becomes identical to cross-entropy. When the normalization factor is removed, having $N = V$ makes the soft label loss identical to the weighted similarity loss.

We found that the normalization factor slightly reduced the accuracy for all of the three tasks (see Figure 2). Interestingly, we have not established a universal tendency for the optimal choice of N: for the language inference tasks, the best accuracy was achieved at N close to 10, while for the paraphrasing task the suitable choice for N was in the range of 3-5.

The performance figures obtained for each loss are well illustrated by the quality of the reconstructed examples in Table 2. The standard cross-entropy, as expected, aims at the accurate word-by-word reconstruction of the input sentence. The autoencoder with our least successful weigthed similarity loss function manages to learn most frequent corpus-specific words (e.g. "great service"), but the overall meaning is not conveyed well. he rest of the models succeed in reconstructing synonyms at the word-level. This results in a slightly different expression style (e.g. "her tone was incredibly rude" becomes "her attitude was incredibly unprofessional"), but the overall meaning is reconstructed correctly.

Obviously, the quality of the generated representations depends to a large extent on the selection of pre-trained word embeddings. The related drawback that we observe in our choice of the fastText vectors is that the target ground-truth tokens can be replaced with word inflections as well as with antonyms, which in certain cases can change the meaning of the sentence to the opposite.

For a subset of configurations, we conducted exploratory testing on additional tasks, including different subsets of STS and SICK-Relatedness data. For nearly all tasks tested, we recorded better performance compared to cross-entropy, with the minimum relative gain being 1%. The only performance reduction was in plagiarism detection, which may be expected to favor exact replication.

Although the scores we obtained are below the state-of-the-art for the considered tasks, our goal was to demonstrate that in a traditional encoder-decoder setting, which is extensively used for a number of NLP problems, the proposed loss functions beat the conventional cross-entropy. The major advantage of our proposal is that it is very simple and highly generalizable, i.e. without a sophisticated model architecture, our model is able to produce diversified outputs and can be easily integrated in any existing encoder-decoder architectures.

5 Conclusion

In this paper, we introduced the loss functions that leverage word-level distributional similarity between the generated output and the ground truth. Compared to the representations learned by a

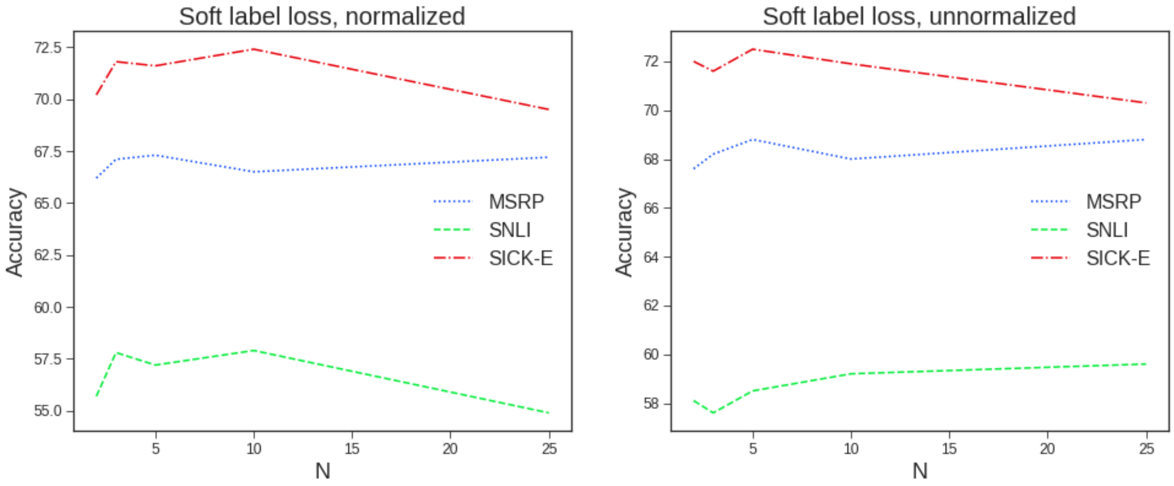

Figure 2: The effect of the soft label loss modifications on task performance. N is the number of closest words neighbors used to compute the loss function.

vanilla autoencoder, the proposed reconstruction loss variants show substantially improved performance on several semantic representation tasks. Further, relative to the conventional cross-entropy, the tested loss variants produce more diverse output while preserving the underlying semantics.

We focused on the autoencoder architecture which requires no pre-annotated data to generate the representations. The major benefit of our proposal is that the proposed loss functions can be plugged directly into any NLP model that generates text word-by-word and that may benefit from more diverse output. The potential applications of our proposal therefore include any of the common NLP problems where language diversity is desirable, including conversational agents, paraphrase generation and text summarization.

The next step for this work is to evaluate the performance of the proposed loss functions in state-of-the-art models for the NLP tasks that leverage sentence-level semantic representation, such as the ones we explored in the present study. Further experiments with the proposed loss functions are also needed to evaluate the effects of different word embedding models on the quality of derived representations.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R Bowman. 2017. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. *arXiv preprint arXiv:1707.08172*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

What can we learn from Semantic Tagging?

Mostafa Abdou[†⋆], **Artur Kulmizev**[⋆], **Vinit Ravishankar**[§],
Lasha Abzianidze[⋆], and **Johan Bos**[⋆]
abdou@di.ku.dk, a.kulmizev@student.rug.nl, vinit.ravishankar@gmail.com,
{l.abzianidze, johan.bos}@rug.nl
CLCG, University of Groningen[⋆]
CoAStaL DIKU, University of Copenhagen[†]
LTG, University of Oslo[§]

Abstract

We investigate the effects of multi-task learning using the recently introduced task of semantic tagging. We employ semantic tagging as an auxiliary task for three different NLP tasks: part-of-speech tagging, Universal Dependency parsing, and Natural Language Inference. We compare full neural network sharing, partial neural network sharing, and what we term the *learning what to share* setting where negative transfer between tasks is less likely. Our findings show considerable improvements for all tasks, particularly in the *learning what to share* setting, which shows consistent gains across all tasks.

1 Introduction

Multi-task learning (MTL) is a recently resurgent approach to machine learning in which multiple tasks are simultaneously learned. By optimising the multiple loss functions of related tasks at once, multi-task learning models can achieve superior results compared to models trained on a single task. The key principle is summarized by Caruana (1998) as "MTL improves generalization by leveraging the domain-specific information contained in the training signals of related tasks". Neural MTL has become an increasingly successful approach by exploiting similarities between Natural Language Processing (NLP) tasks (Collobert and Weston, 2008; Søgaard and Goldberg, 2016; Plank et al., 2016). Our work builds upon Bjerva et al. (2016), who demonstrate that employing semantic tagging as an auxiliary task for Universal Dependency (McDonald et al., 2013) part-of-speech tagging can lead to improved performance.

The objective of this paper is to investigate whether learning to predict lexical semantic categories can be beneficial to other NLP tasks. To achieve this we augment single-task models (ST)[1] with an additional classifier to predict semantic tags and jointly optimize for both the original task and the auxiliary semantic tagging task. Our hypothesis is that learning to predict semantic tags as an auxiliary task can improve performance of single-task systems. We believe that this is, among other factors, due to the following:

- Providing the main task's model with a useful inductive bias, encouraging it to prefer representations that lead to semantically plausible hypotheses over those that are not.
- Putting the focus of the main task model's attention on features that actually matter by providing additional evidence for the relevance or irrelevance of those features.
- Reducing the risk of overfitting by minimizing the model's *Rademacher Complexity*[2] Representations which are learned for multiple tasks have been shown to generalize better (Baxter et al., 2000).

We test our hypothesis on three disparate NLP tasks: (i) Universal Dependency part-of-speech tagging (UPOS), (ii) Universal Dependency parsing (UD DEP), a complex syntactic task; and (iii) Natural Language Inference (NLI), a complex task requiring deep natural language understanding.

2 Background and Related work

2.1 Semantic Tagging

Semantic tagging (Bjerva et al., 2016; Abzianidze and Bos, 2017) is the task of assigning language-neutral semantic categories to words. It is designed to overcome a lack of semantic information syntax-oriented part-of-speech tagsets, such as the

---

[1] We replicate models which perform at or close to the state-of-the-art. Our choice of models is based on replicability.

[2] The ability to fit random noise.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4881–4889
Brussels, Belgium, October 31 - November 4, 2018.

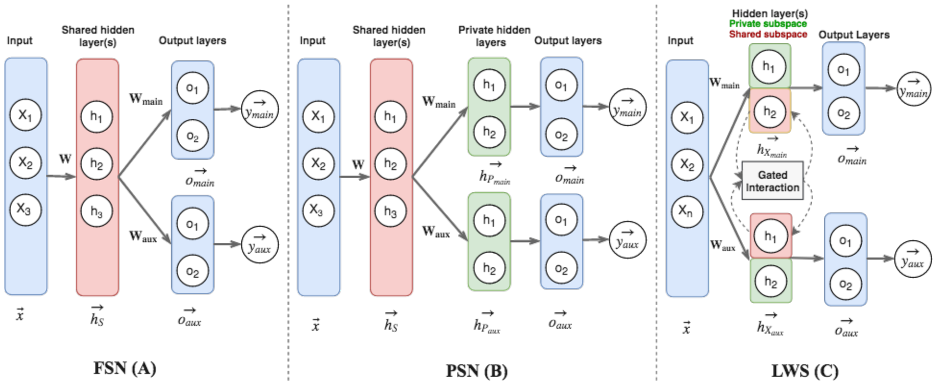

Figure 1: Our three multi-task learning settings: (A) fully shared networks, (B) partially shared networks, and (C) *Learning What to Share*. Layers are mathematically denoted by vectors and the connections between them, represented by arrows, are mathematically denoted by matrices of weights. S indicates a shared layer, P a private layer, and X a layer with shared and private subspaces.

Penn Treebank tagset(Marcus et al., 1993), usually have. Such tagsets exclude important semantic distinctions, such as negation and modals, types of quantification, named entity types, and the contribution of verbs to tense, aspect, or event.

The semantic tagset is language-neutral, abstracts over part-of-speech and named-entity classes, and includes fine-grained semantic information. The tagset consists of 80 semantic tags grouped in 13 coarse-grained classes. The tagset originated in the Parallel Meaning Bank (PMB) project (Abzianidze et al., 2017), where it contributes to compositional semantics and cross-lingual projection of semantic representations. Recent work has highlighted the utility of the tagset as a conduit for evaluating the semantics captured by vector representations (Belinkov et al., 2018), or employed it in an auxiliary tagging task (Bjerva et al., 2016), as we do in this work.

2.2 Learning What to Share

Recently, there has been an increasing interest in the development of models which are trained to learn what to (and what not to) share between a set of tasks, with the general aim of preventing negative transfer when the tasks are not closely related (Meyerson and Miikkulainen, 2017; Ruder et al., 2017; Lu et al., 2017; Misra et al., 2016). Our *Learning What to Share* setting is based on this idea and closely related to Liu et al. (2016)'s shared layer architecture.

Specifically, a layer $\vec{h}_X$ which is shared between the main task and the auxiliary task is split into two subspaces: a shared subspace $\vec{h}_{X_S}$ and a private subspace $\vec{h}_{X_P}$. The interaction between the shared subspaces is modulated via a sigmoidal gating unit applied to a set of learned weights, as seen in Equations (1) and (2) where $\vec{h}_{X_{S(main)}}$ and $\vec{h}_{X_{S(aux)}}$ are the main and auxiliary tasks' shared layers, $W_{a \to m}$ and $W_{m \to a}$ are learned weights, and σ is a sigmoidal function.

$$\vec{h}_{X_{S(main)}} = \vec{h}_{X_{S(main)}} \sigma(\vec{h}_{X_{S(aux)}} W_{a \to m}) \quad (1)$$

$$\vec{h}_{X_{S(aux)}} = \vec{h}_{X_{S(aux)}} \sigma(\vec{h}_{X_{S(main)}} W_{m \to a}) \quad (2)$$

Unlike Liu et al. (2016)'s Shared-Layer Architecture, in our setup each task has its own shared subspace rather than one common shared layer. This enables the sharing of different parameters in each direction (i.e., from main to auxiliary task and from auxiliary to main task), allowing each task to choose what to learn from the other, rather than having "one shared layer to capture the shared information for all the tasks" as in Liu et al. (2016).

3 Multi-Task Learning Settings

We implement three neural MTL settings, shown in Figure 1. They differ in the way the network's parameters are shared between the tasks:

- **Fully shared network (FSN):** All hidden layers are entirely shared among the tasks, each task has a separate output layer. The transformation of our input vector $\vec{x}$ into a shared hidden layer $\vec{h}_S$ is described by Equation (3):

$$\vec{h}_S = \sigma(\vec{x}W) \quad (3)$$

- **Partially shared network (PSN):** A subset of hidden layers is shared among the tasks;

each task has at least one private hidden layer and a separate output layer. The transformation of a *shared* hidden layer $\vec{h}_S$ into *private* hidden layers, denoted by $\vec{h}_{P_{(main)}}$ and $\vec{h}_{P_{(aux)}}$ is described in Equations (4) and (5).

$$\vec{h}_{P_{(main)}} = \sigma(\vec{h_S} W_{(main)}) \tag{4}$$

$$\vec{h}_{P_{(aux)}} = \sigma(\vec{h_S} W_{(aux)}) \tag{5}$$

- **Learning What to Share (LWS):** Each task has a dedicated set of hidden layers. For sharing, a hidden layer is split into a shared subspace and a private subspace. A gating unit modulates the transfer of information between the shared subspaces as shown in Equations (1) and (2).

4 Data

In the UPOS tagging experiments, we utilize the UD 2.0 English corpus (Nivre et al., 2017) for the POS tagging and the semantically tagged PMB release 0.1.0 (sem-PMB)[3] for the MTL settings. Note that there is no overlap between the two datasets. Conversely, for the UD DEP and NLI experiments there is a complete overlap between the datasets of main and auxiliary tasks, i.e., each instance is labeled with both the main task's labels and semantic tags. We use the Stanford POS Tagger (Toutanova et al., 2003) trained on sem-PMB to tag the UD corpus and NLI datasets with semantic tags, and then use those assigned tags for the MTL settings of our dependency parsing and NLI models. We find that this approach leads to better results when the main task is only loosely related to the auxiliary task. The UD DEP experiments use the English UD 2.0 corpus, and the NLI experiments use the SNLI (Bowman et al., 2015) and SICK-E[4] datasets (Marelli et al., 2014). The provided train, development, and test splits are used for all datasets. For sem-PMB, the silver and gold parts are used for training and testing respectively.

5 Experiments

We run four experiments for each of the four tasks (UPOS, UD DEP, SNLI, SICK-E), one using the ST model and one for each of the three MTL settings. Each experiment is run five times, and the average of the five runs is reported. We briefly describe the ST models and refer the reader to the original work for further details due to a lack of space.[5] For reproducibility, detailed diagrams of the MTL models for each task and their hyperparameters can be found in Appendix A.

5.1 Universal Dependency POS Tagging

Our tagging model uses a basic contextual one-layer bi-LSTM (Hochreiter and Schmidhuber, 1997) that takes in word embeddings and produces a sequence of recurrent states which can be viewed as contextualized representations. The recurrent r_n state from the bi-LSTM corresponding to each time-step t_n is passed through a dense layer with a softmax activation to predict the token's tag.

In each of the MTL settings a softmax classifier is added to predict a token's semantic tag and the model is then jointly trained on the concatenation of the sem-PMB and UPOS tagging data to minimize the sum of softmax cross-entropy losses of both the main (UPOS tagging) and auxiliary (semantic tagging) tasks.

5.2 Universal Dependency Parsing

We employ a parsing model that is based on Dozat and Manning (2016). The model's embeddings layer is a concatenation of randomly initialized word embeddings[6] and character-based word representations added to pre-trained word embeddings, which are passed through a 4-layer stacked bi-LSTM. Unlike Dozat and Manning (2016), our model jointly learns to perform UPOS tagging and parsing, instead of treating them as separate tasks. Therefore, instead of tag embeddings, we add a softmax classifier to predict UPOS tags after the first bi-LSTM layer. The outputs from that layer and the UPOS softmax prediction vectors are both concatenated to the original embedding layer and passed to the second bi-LSTM layer. The output of the last bi-LSTM is then used as input for four dense layers with a ReLU activation, producing four vector representations: a word as a dependent seeking its head; a word as a head seeking all its dependents; a word as a dependent deciding on its label; a word as head deciding on the labels of its dependents. These representations are then passed to biaffine and affine softmax classifiers to produce a fully-connected labeled probabilistic dependency graph (Dozat and Manning,

---

[3] `http://pmb.let.rug.nl/data.php`

[4] SICK-E refers to the entailment part of the SICK dataset.

[5] This applies to UD DEP and NLI only, as the POS tagger is not based on any one particular work.

[6] This replaces the holistic word embeddings for frequent words in Dozat and Manning (2016).

2016). Finally, a non-projective maximum spanning tree parsing algorithm (Chu, 1965; Edmonds, 1967) is used to obtain a well-formed dependency tree.[7]

Similarly to UPOS tagging, an additional softmax classifier is used to predict a token's semantic tag in each of the MTL settings, as both tasks are jointly learned. In the FSN setting, the 4-layer stacked bi-LSTM is entirely shared. In the PSN setting the semantic tags are predicted from the second layer's hidden states, and the final two layers are devoted to the parsing task. In the LWS setting, the first two layers of the bi-LSTM are split into a private bi-LSTM$_{private}$ and a shared bi-LSTM$_{shared}$ for each of the tasks with the interaction between the shared subspaces being modulated via a gating unit. Then, two bi-LSTM layers that are devoted to parsing only are stacked on top.

5.3 Natural Language Inference

We base our NLI model on Chen et al. (2017)'s Enhanced Sequential Inference Model which uses a bi-LSTM to encode the premise and hypothesis, computes a soft-alignment between premise and hypothesis' representations using an attention mechanism, and employs an inference composition bi-LSTM to compose local inference information sequentially.[8] The MTL settings are implemented by adding a softmax classifier to predict semantic tags at the level of the encoding bi-LSTM, with rest of the model unaltered. In the FSN setting, the hidden states of the encoding bi-LSTM are directly passed as input to the softmax classifier. In the PSN setting an earlier bi-LSTM layer is used to predict the semantic tags and the output from that is passed on to the encoding bi-LSTM which is stacked on top. This follows Hashimoto et al. (2016)'s hierarchical approach. In the LWS setting, a bi-LSTM layer with private and shared subspaces is used for semantic tagging and for the ESIM model's encoding layer. In all MTL settings, the bi-LSTM used for semantic tagging is pre-trained on the sem-PMB data.

6 Results and Discussion

Results for all tasks are shown in Table 1. In line with Bjerva et al. (2016)'s findings, the FSN setting leads to an improvement for UPOS tagging. POS tagging, a sequence labeling task, can be seen as the most closely related to semantic tagging, therefore negative transfer is minimal and the full sharing of parameters is beneficial. Surprisingly, the FSN setting also leads improvements for UD DEP. Indeed, for UD DEP, all of the MTL models outperform the ST model by increasing margins. For the NLI tasks, however, there is a clear degradation in performance.

The PSN setting shows mixed results and does not show a clear advantage over FSN for UPOS and UD DEP. This suggests that adding task-specific layers after fully-shared ones does not always enable sufficient task specialization. For the NLI tasks however, PSN is clearly preferable to FSN, especially for the small-sized SICK-E dataset where the FSN model fails to adequately learn.

Model	SNLI	SICK-E	UPOS	UD DEP
ST	87.01	81.30	92.12	80.24 / 84.87
FSN	84.96	56.69	92.95	81.03 / 85.54
PSN	87.08	77.92	92.34	80.92 / 85.81
LWS	**87.51**	**84.57**	**95.54**	**81.39 / 86.00**

Table 1: Results for single-task models (ST), fully-shared networks (FSN), partially-shared networks (PSN), and *learning what to share* (LWS). All scores are reported as accuracy, except UD DEP for which we report LAS/UAS F_1 score.

As a sentence-level task, NLI is functionally dissimilar to semantic tagging. However, it is a task which requires deep understanding of natural language semantics and can therefore conceivably benefit from the signal provided by semantic tagging. Our results demonstrate that it is possible to leverage this signal given a selective sharing setup where negative transfer can be minimized. Indeed, for the NLI tasks, only the LWS setting leads to improvements over the ST models.[9] The improvement is larger for the SICK-E task which has a much smaller training set and therefore stands to learn more from the semantic tagging signal. For all tasks, it can be observed that the LWS models outperform the rest of the models. This is in line with our expectations with the findings from previous work (Ruder et al., 2017; Liu et al.,

---

[7] This is recommended but not implemented by Dozat et al. (2017).

[8] We do not implement the additional tree-LSTM model used in Chen et al. (2017) as we focus on the effect of MTL with semantic tagging rather than on absolute performance.

[9] Demonstrative examples of the SNLI models' outputs can be found in Appendix B.

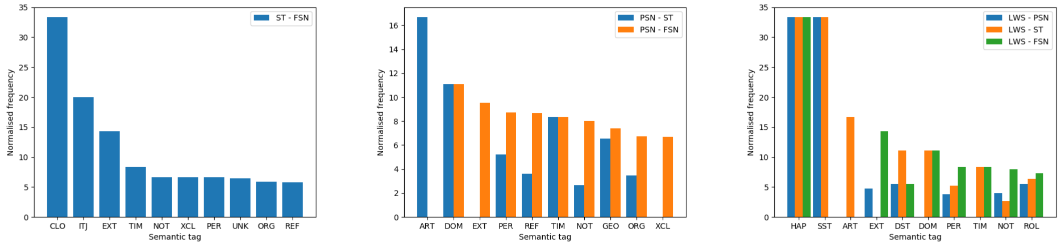

(a) Single-task network (b) Partially shared network (c) Learning what to share

Figure 2: Normalized semantic tag frequencies for all six sets of sentences. `X - Y` denotes the set of sentences correctly classified by model X but misclassified by model Y.

2016) that selective sharing outperforms full network and partial network sharing.

7 Analysis

In addition to evaluating performance directly, we attempt to qualify how semtags affect performance with respect to each of the SNLI MTL settings[10].

7.1 Qualitative analyses

The fact that NLI is a sentence-level task, while semantic tags are word-level annotations presents a difficulty in measuring the effect of semantic tags on the systems' performance, as there is no one-to-one correspondence between a correct label and a particular semantic tag. We therefore employ the following method in order to assess the contribution of semantic tags. Given the performance ranking of all our systems — $FSN < ST < PSN < LWS$ — we make a pairwise comparison between the output of a superior system S_{sup} and an inferior system S_{inf}. This involves taking the pairs of sentences that every S_{sup} classifies correctly, but some S_{inf} does not. Given that FSN is the worst performing system and, as such, has no 'worse' system for comparison, we are left with six sets of sentences: ST-FSN, PSN-FSN, PSN-ST, LWS-PSN, LWS-ST, and LWS-FSN. To gain insight as to where a given system S_{sup} performs better than a given S_{inf}, we then sort each comparison sentence set by the frequency of semtags predicted therein, which are normalized by dividing by their frequency in the full SNLI test set.

We notice interesting patterns, visible in Figure 2. Specifically, PSN appears markedly better at sentences with named entities (`ART`, `PER`, `GEO`, `ORG`) and temporal entities (`DOM`) than both ST and the FSN. Marginal improvements are also observed for sentences with negation and reflexive pronouns. The LWS setting continues this pattern, with additional improvements observable for sentences with the `HAP` tag for names of events, `SST` for subsective attributes, and the `ROL` tag for role nouns.

7.2 Contribution of semantic tagging

To assess the contribution of the semantic tagging auxiliary task independent of model architecture and complexity we run three additional SNLI experiments — one for each MTL setting — where the model architectures are unchanged but the auxiliary tasks are assigned no weight (i.e. do not affect the learning). The results confirm our previous findings that, for NLI, the semantic tagging auxiliary task only improves performance in a selective sharing setting, and hurts it otherwise: i) the FSN system which had performed below ST improves to equal it and ii) the PSN and LWS settings both see a drop to ST-level performance.

8 Conclusions

We present a comprehensive evaluation of MTL using a recently proposed task of semantic tagging as an auxiliary task. Our experiments span three types of NLP tasks and three MTL settings. The results of the experiments show that employing semantic tagging as an auxiliary task leads to improvements in performance for UPOS tagging and UD DEP in all MTL settings. For the SNLI tasks, requiring understanding of phrasal semantics, the selective sharing setup we term *Learning What to Share* holds a clear advantage. Our work offers a generalizable framework for the evaluation of the utility of an auxiliary task.

---

[10] We also provide a per-label report of the standard precision and recall metrics in Appendix B.

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247. Association for Computational Linguistics.

Lasha Abzianidze and Johan Bos. 2017. Towards universal semantic tagging. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS 2017) – Short Papers*, pages 1–6, Montpellier, France.

Jonathan Baxter et al. 2000. A model of inductive bias learning. *J. Artif. Intell. Res.(JAIR)*, 12(149-198):3.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3531–3541. The COLING 2016 Organizing Committee.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. 2017. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5334–5343.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 92–97.

Elliot Meyerson and Risto Miikkulainen. 2017. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. *arXiv preprint arXiv:1711.00108*.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003.

Joakim Nivre, Lars Ahrenberg Zeljko Agic, et al. 2017. Universal dependencies 2.0. lindat/clarin digital library at the institute of formal and applied linguistics, charles university, prague.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 231–235.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

A MTL setting Diagrams, Preprocessing, and Hyperparameters

UPOS Tagging

Figure 3a shows the three MTL models used for UPOS. All hyperparameters were tuned with respect to loss on the English UD 2.0 UPOS validation set. We trained for 20 epochs with a batch size of 128 and optimized using Adam (Kingma and Ba, 2014) with a learning rate of 0.0001. We weight the auxiliary semantic tagging loss with $\lambda = 0.1$. The pre-trained word embeddings we used are GloVe embeddings (Pennington et al., 2014) of dimension 100 trained on 6 billion tokens of Wikipedia 2014 and Gigaword 5. We applied dropout and recurrent dropout with a probability of 0.3 to all bi-LSTMs.

UD DEP

Figure 3b shows the three MTL models for UD DEP. We use the gold tokenization. All hyperparameters were tuned with respect to loss on the English UD 2.0 UD validation set. We trained for 15 epochs with a batch size of 50 and optimized using Adam with a learning rate of $2e-3$. We weight the auxiliary semantic tagging loss with λ = 0.5. The pre-trained word embeddings we use are GloVe embeddings of dimension 100 trained on 6 billion tokens of Wikipedia 2014 and Gigaword 5. We applied dropout with a probability of 0.33 to all bi-LSTM, embedding layers, and non-output dense layers.

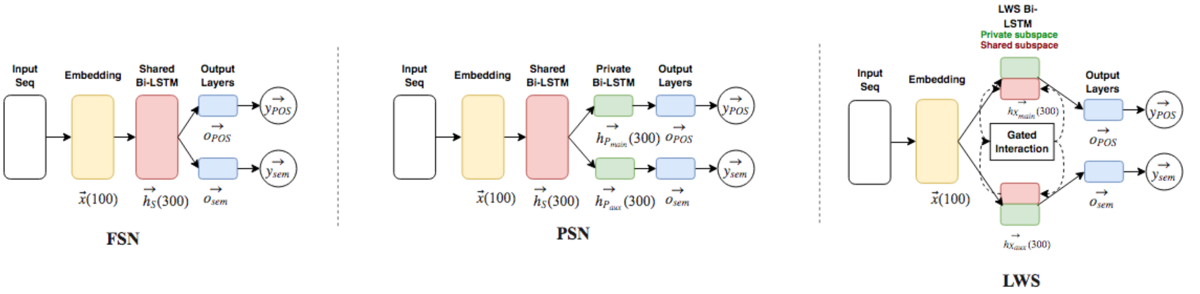

(a) UPOS

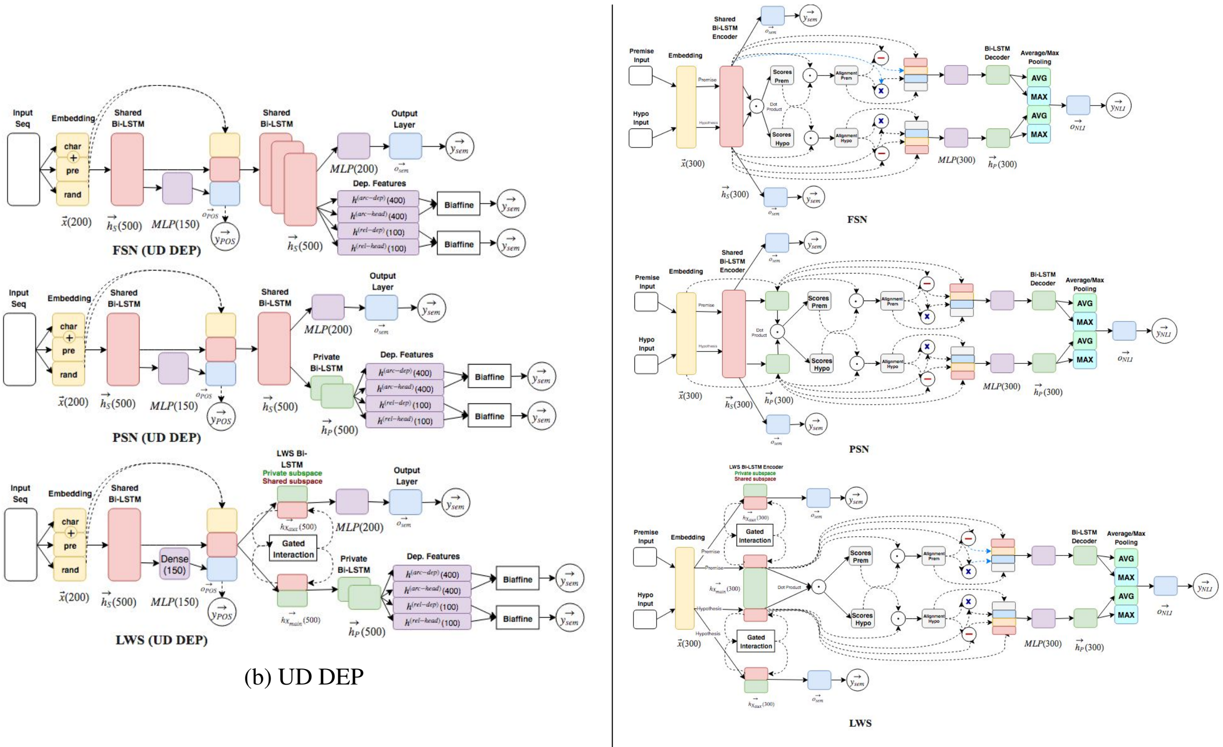

(b) UD DEP

(c) NLI

Figure 3: The three MTL settings for each task. Layers dimensions are displayed in brackets.

NLI

Figure 3c shows the three MTL models for NLI. All hyperparameters were tuned with respect to loss on the SNLI and SICK-E validation datasets (separately). For the SNLI experiments, we trained for 37 epochs with a batch size of 128. For the SICK-E experiments, we trained for 20 epochs with a batch size of 8. Note that the ESIM model was designed for the SNLI dataset, therefore performance is non-optimal for SICK-E. For both sets of experiments: we optimized using Adam with a learning rate of 0.00005; we weight the auxiliary semantic tagging loss with $\lambda = 0.1$; the pre-trained word embeddings we use are GloVe embeddings of dimension 300 trained on 840 billion tokens of Common Crawl; and we applied dropout and recurrent dropout with a probability of 0.3 to all biLSTM, and non-output dense layers.

B SNLI model output analysis

Table 2 shows demonstrative examples from the SNLI test set on which the *Learning What to Share* (LWS) model outperforms the single-task (ST) model. The examples cover all possible combinations of entailment classes. Table 3 explains the relevant part of the semantic tagset. Table 4 shows the per-label precision and recall scores.

Tag category	Semantic tag with examples
Anaphoric	`DEF`: definite; *the, lo*^{IT}*, der*^{DE}
	`HAS`: possessive pronoun; *my, her*
Attribute	`COL`: colour; *red, crimson, light_blue, chestnut_brown*
	`QUC`: concrete quantity; *two, six_million, twice*
	`IST`: intersective; *open, vegetarian, quickly*
	`REL`: relation; *in, on, 's, of, after*
Unnamed entity	`CON`: concept; *dog, person*
Logical	`ALT`: alternative & repetitions; *another, different, again*
	`DIS`: disjunction & exist. quantif.; *a, some, any, or*
Discourse	`SUB`: subordinate relations; *that, while, because*
Events	`ENS`: present simple; *we walk, he walks*
	`EPS`: past simple; *ate, went*
	`EXG`: untensed progessive; *is running*
	`EXS`: untensed simple; *to walk, is eaten, destruction*
Tense & aspect	`NOW`: present tense; *is skiing, do ski, has skied, now*

Table 3: The list of semantic tags found in Table 2.

Model	**Label**		
	Entailment	Contradiction	Neutral
FSN	80.64/93.23	91.64/83.63	83.97/77.63
ST	84.86/91.54	90.10/88.04	84.74/79.71
PSN	84.08/92.70	91.17/88.63	85.96/79.15
LWS	84.45/92.87	91.74/88.91	85.95/79.65

Table 4: Per-label precision (left) and recall (right) for all models.

Premise-hypothesis pairs	**ST**	**LWS/GOLD**
P: The$^{\text{DEF}}$ gentleman$^{\text{CON}}$ is$^{\text{NOW}}$ speaking$^{\text{EXS}}$ while$^{\text{SUB}}$ the$^{\text{DEF}}$ others$^{\text{ALT}}$ are$^{\text{NOW}}$ listening$^{\text{EXS}}$ **H**: The$^{\text{DEF}}$ man$^{\text{CON}}$ is$^{\text{NOW}}$ being$^{\text{EXS}}$ given$^{\text{EXS}}$ respect$^{\text{CON}}$	N	E
P: Men$^{\text{CON}}$ wearing$^{\text{EXG}}$ hats$^{\text{CON}}$ walk$^{\text{EXS}}$ on$^{\text{REL}}$ the$^{\text{DEF}}$ street$^{\text{CON}}$ **H**: The$^{\text{DEF}}$ men$^{\text{CON}}$ having$^{\text{EXS}}$ hats$^{\text{CON}}$ on$^{\text{REL}}$ their$^{\text{HAS}}$ head$^{\text{CON}}$	C	E
P: Three$^{\text{QUC}}$ men$^{\text{CON}}$ in$^{\text{REL}}$ orange$^{\text{IST}}$ suits$^{\text{CON}}$ are$^{\text{NOW}}$ doing$^{\text{EXG}}$ street$^{\text{CON}}$ repairs$^{\text{CON}}$ at$^{\text{REL}}$ night$^{\text{CON}}$ **H**: Three$^{\text{QUC}}$ men$^{\text{CON}}$ in$^{\text{REL}}$ orange$^{\text{IST}}$ suits$^{\text{CON}}$ escaped$^{\text{EPS}}$ from$^{\text{REL}}$ prison$^{\text{CON}}$	N	C
P: A$^{\text{DIS}}$ toddler$^{\text{CON}}$ sits$^{\text{ENS}}$ on$^{\text{REL}}$ a$^{\text{DIS}}$ stone$^{\text{CON}}$ wall$^{\text{CON}}$ surrounded$^{\text{EXS}}$ by$^{\text{REL}}$ fallen$^{\text{EXS}}$ leaves$^{\text{CON}}$ **H**: An$^{\text{DIS}}$ child$^{\text{CON}}$ is$^{\text{NOW}}$ throwing$^{\text{EXG}}$ stones$^{\text{CON}}$ at$^{\text{REL}}$ a$^{\text{DIS}}$ leaf$^{\text{CON}}$ wall$^{\text{CON}}$	E	C
P: An$^{\text{DIS}}$ old$^{\text{IST}}$ shoemaker$^{\text{CON}}$ in$^{\text{REL}}$ his$^{\text{HAS}}$ factory$^{\text{CON}}$ **H**: The$^{\text{DEF}}$ shoemaker$^{\text{CON}}$ is$^{\text{NOW}}$ wealthy$^{\text{IST}}$	C	N
P: A$^{\text{DIS}}$ kid$^{\text{CON}}$ slides$^{\text{CON}}$ down$^{\text{IST}}$ a$^{\text{DIS}}$ yellow$^{\text{COL}}$ slide$^{\text{CON}}$ into$^{\text{REL}}$ a$^{\text{DIS}}$ swimming$^{\text{CON}}$ pool$^{\text{CON}}$ **H**: The$^{\text{DEF}}$ kid$^{\text{CON}}$ is$^{\text{NOW}}$ playing$^{\text{EXS}}$ at$^{\text{REL}}$ the$^{\text{DEF}}$ waterpark$^{\text{CON}}$	E	N

Table 2: Examples of the entailment problems from SNLI which are incorrectly classified by the ST model but correctly classified by the LWS model. Automatically assigned semantic tags are in superscript.

Conditional Word Embedding and Hypothesis Testing via Bayes-by-Backprop

Rujun Han
Information Sciences Institute
University of Southern California
rujunhan@usc.edu

Michael Gill
Center for Data Science
New York University
mzgill@nyu.edu

Arthur Spirling
Center for Data Science
New York University
arthur.spirling@nyu.edu

Kyunghyun Cho
Center for Data Science
New York University
CIFAR Global Scholar
kyunghyun.cho@nyu.edu

Abstract

Conventional word embedding models do not leverage information from document metadata, and they do not model uncertainty. We address these concerns with a model that incorporates document covariates to estimate conditional word embedding distributions. Our model allows for (a) hypothesis tests about the meanings of terms, (b) assessments as to whether a word is near or far from another conditioned on different covariate values, and (c) assessments as to whether estimated differences are statistically significant.

1 Introduction

Whether a word's meaning varies across contexts has become a major focus of NLP, linguistics, and social science research in recent years. For example, since the early 20th century, the word "gay" has evolved from describing an emotion to being more aligned with sexual orientation (Hamilton et al., 2016b). Popular word embedding techniques (e.g., Mikolov et al., 2013a; Pennington et al., 2014) have proven useful for analyzing language evolution. But to use these models for such research, scholars often divide a corpus into distinct training sets (e.g., train independent language models on different decades of text) and compare model output across specifications in an *ad hoc* way (Garg et al., 2018). Such splitting inhibits many within- and across-word comparisons, since embeddings are only comparable within a given model. Additionally, most methods ignore the *variance* of words, mechanically treating words equally regardless of the volatility, or uncertainty, in their meanings. If one inspects semantics with only point estimates of embeddings, it is hard to tell whether embeddings represent meaningful traits or are simply noise in the data.

We address these concerns in three ways. First, we estimate a vector for each distinct value of the document covariates, using a multilayer perceptron (MLP) with a non-linear activation function. Second, we parametrize the covariance matrix of each embedding vector explicitly in the model, adopting the Bayes-by-Backprop algorithm (Blundell et al., 2015). Third, we utilize Hotelling T^2 statistics (Hotelling, 1931) to assess whether estimated differences in word vectors are statistically differentiable under a null χ^2 distribution (Ito, 1956). To our knowledge, no prior work evaluates word embeddings with this statistical framework.

2 Related Work

Drift Analysis using Word Embeddings There are several ways to measure drifts in word meanings. Hamilton et al. (2016c) propose the use of cosine similarities of words in different contexts to detect changes. Hamilton et al. (2016b) provide an alternative measure based on the distance of words from their nearest neighbors. Rudolph and Blei (2018) analyze absolute drift of words using Euclidean distance in (two discrete) slices of data. All of these methods compute the word distance based only on the point (i.e., mean) estimates of the word embeddings.

Conditional Word Embedding Rudolph and Blei (2018) estimate dynamic Bernoulli embeddings (DBE), extending the exponential family embedding (Rudolph et al., 2016) generalization of Mikolov et al. (2013a), to learn conditional word embeddings over time. Their amortized approach builds a separate neural network that transforms a global word vector into a covariate-specific vector, and is closely related to our approach in this paper. However, a noticeable omission in their model is that they do not explicitly model parameter covariance or uncertainty.

Word Embedding with Uncertainty Vilnis and McCallum (2017) earlier proposed an energy-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4890–4895
Brussels, Belgium, October 31 - November 4, 2018.

based learning framework in which each word is represented as a multivariate Gaussian distribution with a diagonal covariance. The energy function is defined by the divergence (e.g., KL) between two Gaussian embeddings, and the margin ranking loss (Weston et al., 2011) is minimized. A related model is the Bayesian skip-gram in Brazinskas et al. (2017), which posits a generative model where words are associated with multivariate Gaussian latent variables that generate context words. The parameters of those prior distributions over the multivariate Gaussian latent variables are estimated by maximizing the variational lowerbound, and act as word embeddings.

These works replace mean estimates of embeddings with Gaussian distributions, similar to our proposal here. However, they arrive at this differently; Vilnis and McCallum (2017) from the energy-based learning (LeCun et al., 2006), and Brazinskas et al. (2017) from generative modeling. We provide yet another angle: via (approximate) Bayesian neural networks.

3 Conditional Word Embedding

Adopting Bayes-by-Backprop for Estimation Given a tuple of a word v, a covariate x and a context word v_c, we define the conditional log-probability as

$$\log p(v_c|v,x) = \theta_{v|x}^{\top}\theta_{v_c}^{c} - \log \sum_{v'_c \in V} \exp\left(\theta_{v|x}^{\top}\theta_{v'_c}^{c}\right),$$

where $\theta_{v|x}$ and $\theta_{v_c}^{c}$ are the conditional word embedding of v given x and the context embedding of v_c, respectively. V is the vocabulary of all unique words. To avoid the expensive computation of the partition function, we use negative sampling (Mikolov et al., 2013b), which stochastically approximates the log-probability above by:

$$\begin{aligned}\log p(v_c|v,x) \approx & \log \sigma(\theta_{v|x}^{\top}\theta_{v_c}^{c}) \\ & + \frac{1}{M}\sum_{m=1}^{M} \log(1-\sigma(\theta_{v|x}^{\top}\theta_{v_c^m}^{c})),\end{aligned} \tag{1}$$

where $v_c^m \in V$ is the m-th negative sample drawn from a unigram distribution estimated from D.

We define a prior distribution over each parameter θ to be a scaled mixture of two Gaussians, as suggested by Blundell et al. (2015):

$$\begin{aligned}\log p(\theta_i) = & \log\left(u\mathcal{N}(\theta_i|0,\sigma_1^2)\right. \\ & \left.+(1-u)\mathcal{N}(\theta_i|0,\sigma_2^2)\right),\end{aligned} \tag{2}$$

where σ_1, σ_2 and u are the hyperparameters.

As exactly marginalizing out the parameters $\theta_{\cdot}$ and $\theta_{\cdot}^{c}$ is not scalable, we maximize the variational lowerbound of the marginal probability. To do so, we introduce a variational posterior $q(\theta|\phi)$ parametrized by its own parameter set ϕ. Then, the variational lowerbound is defined as $-\mathcal{F}(\theta,D) = \mathbb{E}_q\left[\log p(D|\theta)\right] - \mathrm{KL}(q(\theta)\|p(\theta))$, where $\log p(D|\theta) = \sum_{(v,x,v_c)\in D} \log p(v_c|v,x)$ in our case. This is stochastically approximated by

$$\begin{aligned}-\mathcal{F}(\theta,D) \approx & \frac{1}{M}\sum_{m=1}^{M} \log p(D|\theta^{(m)}) \\ & - \log q(\theta^{(m)}|\phi) + \log p(\theta^{(m)}),\end{aligned} \tag{3}$$

where $\theta^{(m)}$ is the m-th sample from the variational posterior q (Blundell et al., 2015) via the Gaussian reparametrization in Kingma and Welling (2013). We formulate the variational posterior as a multivariate Gaussian with diagonal covariance.

We use stochastic gradient descent (SGD) to minimize $\mathcal{F}$ with respect to the variational parameters ϕ. At each SGD step, we compute the gradient of the following per-example cost given an example $(v, v_c, x) \in D$:

$$\begin{aligned}f(\theta,(v,v_c,x)) \approx & -\log p(v_c|v,x) + \log q(\tilde{\theta}_{v|x}|\phi) \\ & + \log q(\tilde{\theta}_{v_c}^{c}) + \log q(\tilde{\theta}_{v'_c}^{c}) - \log p(\tilde{\theta}),\end{aligned}$$

where $\tilde{\theta}$ is a single sample from the approximate posterior, and $\log p(v_c|v,x)$ and $\log p(\tilde{\theta})$ are from Eqs. (1)–(2). We then estimate the (approximate) posterior distribution of each conditional word embedding $\theta_{v|x}$ rather than its point estimate, by minimizing $\mathcal{F}$. See Sec. A of the supplementary material for the detailed steps for computing the per-example cost.

Parametrized Conditional Word Embedding An issue with the approach described so far is the number of parameters grows linearly in the size of the vocabulary and in the number of covariate partitions, i.e., $O(|V| \times |C|)$, where C is the set of all partitions. This effectively excludes any potential sharing of structures underlying words across different covariate values and decreases the number of examples per parameter. To avoid this issue, we use a single parametrized function to compute the variational parameters ϕ of each conditional word embedding $\theta_{v|x}$.

For each covariate-word $v|x$, there are two variational parameters $\mu_{v|x}$ and $\sigma_{v|x}$. We use an MLP without any hidden layer and tanh output layer, i.e., the affine transformation followed by pointwise tanh, that takes as input both a global word vector $\mu_v^{(v)}$ and a covariate vector $\mu_x^{(x)}$ and outputs $\mu_{v|x}$, i.e., $\mu_{v|x} = f_\psi(\left[\mu_v^{(v)}; \mu_x^{(x)}\right])$, where ψ is the parameters of this mean-transformation network. The diagonal covariance $\sigma_{v|x}$ is parametrized as $\sigma_{v|x} = \log(1 + \exp(\rho_v))$, where ρ_v is a parameter shared across all covariate configurations. We then minimize $\mathcal{F}$ w.r.t. these parameters ψ, $\left\{\mu_v^{(v)}, \rho_v\right\}_{v \in V}$ and $\left\{\mu_x^{(x)}\right\}_{x \in C}$.

This approach of parametrized conditional word embeddings significantly reduces the number of parameters from $O(|V| \times |C|)$ to $O(|V| + |C|)$, while maintaining posterior uncertainty of the estimated conditional word embedding $\theta_{v|x}$.

4 Divergences for Word Embeddings

As we estimate the approximate posterior uncertainty of conditional word vectors, we can estimate richer relations between vectors (e.g., KL) in addition to more common comparisons (e.g., cosine or Euclidean distance). Moreover, we can explicitly test for whether two vectors are (un)likely to have the same mean in the population. Below, we introduce how Hotelling's T^2 may be used for word-drift or across-word hypothesis testing.

Hotelling's T^2 Statistic We use the estimated posterior mean vector $\mu_{v|x}$ and the diagonal covariance vector $\sigma_{v|x}$ of two word-covariate pairs $v|x_i$ and $v|x_j$ to compute the T^2 statistic, as if they were estimates from two sets of samples: $T^2 = (\mu_i - \mu_j)^\top \mathrm{diag}(s)^{-1}(\mu_i - \mu_j)$. The pooled (diagonal) covariance s of word pairs is computed by $s = \frac{(n_i-1)\cdot\sigma_i^2+(n_j-1)\cdot\sigma_j^2}{n_i+n_j-2}$, where n_i and n_j are the numbers of occurrences of $v|x_i$ and $v|x_j$ in D, respectively.[1] Unlike other divergence measures, this T^2 statistic explicitly takes into account the frequencies of the word-covariate pairs.

Under general conditions, e.g., D is large, the sampling distribution of T^2 converges to a χ_d^2 distribution (Ito, 1956) with d equal to the embedding dimensionality. This allows us to statistically test such a null hypothesis as $\mathrm{Diff}(v_i|x, v_j|x) = 0$ and $\mathrm{Diff}(v|x_i, v|x_j) = 0$.

5 Application: Political Speech in UK

Data We use U.K. Parliament speech records from 1935-2012 as our training data (Rheault et al., 2016). Our conditioning variable of interest is the decade in which a speech occurred. More details are in Sec. B of the supplementary file.

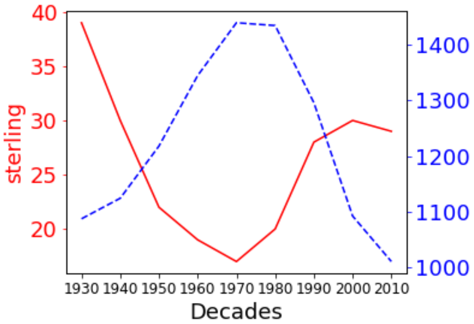

Figure 1: The ranks of "sterling" (solid line) and "pound" (dotted line) w.r.t. "currency" across the decades according to KL divergence.

Model and Learning For each word in the corpus, we consider six surrounding words as its context. The size of embedding is set to 100. We use six negative samples to compute Eq. (1). We use Adagrad (Duchi et al., 2011) with the initial learning rate 0.05 for learning.[2] For other hyperparameters, see the supplementary material. We refer to our approach by BBP. For comparison, we also train analogous DBE embeddings using code from the authors.

6 Result and Analysis

Impact of Covariates To demonstrate how document covariates influence conditional word embeddings, we compare the vector for "currency" against "sterling" and "pound" according to the KL divergence in each decade, which is shown in Fig. 1. In each time period we report the ranking of each w.r.t "currency". Here, we observe that pivotal points for both "sterling" and "pound" occur in the 1970s, which coincides with the moment the UK began to abandon the 'sterling area' (Part III in Schenk, 2010). As such, this financial policy appears to have encouraged semantic drift of the word "pound" towards "currency". See Sec. D in the supplementary material for more details.

We also show a few more examples in Figure 2 and Figure 3 from the Dictionary Induction section below.

Dictionary Induction As a quantitative comparison between the proposed approach and the DBE, we take a dictionary of (British) political terms by Laver and Garry (2000) and look at the

---

[1] T^2 is valid only when $n_i > 1$ and $n_j > 1$.

[2] https://github.com/rhan1207/ConditionalEmbeddings

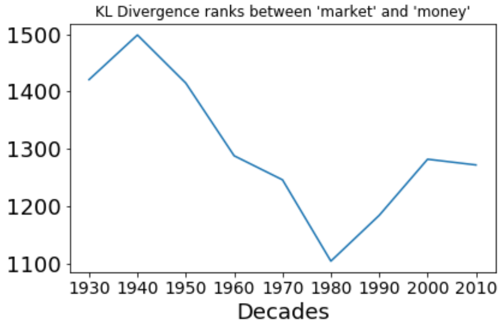

Figure 2: The ranks between "market" and "money" across the decades according to KL divergence.

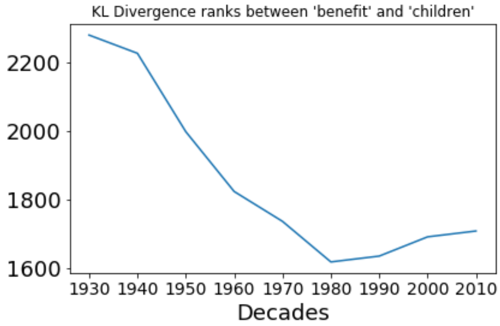

Figure 3: The ranks between "benefit" and "children" across the decades according to KL divergence.

average pair-wise, directional rank in each category ("pro-state", "con-state" and "neutral-state"). We only consider the 2,000 most frequent words in the vocabulary and embeddings with the covariate (decade) set to 2000s. We observed that the proposed model using KL divergence has significantly smaller average pair-wise ranks in "pro-state" (4052 vs. 5047) and "con-state" (2578 vs. 3758) while performs slightly worse than DBE in "neutral-state" category (5414 vs. 5031) suggesting that the proposal approach can cluster words from similar semantic group into closer neighbors than DBE.

Furthermore, we pick 5 most frequent words from "pro-state" and "con-state" and show their average pair-wise rankings and percentile in Table 1. Out of 25K words, our proposed model is able to rank most chosen words within top 10% percentile.

Statistical Word Drift Analysis Our BBP approach permits meaningful downstream hypothesis tests of word drift, i.e, Diff$(v|x_i, v|x_j) = 0$, and across-word similarity, i.e., Diff$(v_i, v_j) = 0$. Among the 2,000 most frequent words in our sample, we perform hypothesis tests of word drifts, comparing vectors from the 1940s against those from the 2000s. We compare results from BBP against the top-100 estimated drifts via DBE. We first observe that most of the top-ranked words by L2 distance in the DBE model are not statistically significant. With the p-value threshold of $\alpha = 0.1$, only eleven words were deemed to have had significant drift, including "council", "labour", "european" and "defence". Sec. E of the supplement includes entire lists of this drift analysis.

Pro-state			Con-state		
Words	Ranks	Pctl	Words	Ranks	Pctl
benefit	1437	5.7	market	1783	7.1
children	2432	9.8	money	1623	6.5
education	716	2.9	own	2852	11.4
health	996	4.0	private	1670	6.7
transport	4247	17.0	value	1693	6.8

Table 1: Average pair-wise rankings of most frequent words in "pro-state" and "con-state" from a British political dictionary.

	DBE	No Covariance		Covariance	
Words	Ranks	L2	cosine	KL	T^2
uk	1	1.60	0.81	61.4	99.7
eu	2	1.58	0.84	44.6	89.2
war	6	1.52	0.85	48.4	96.8
council	8	1.66	0.84	71.0	142.0^{**}
labour	15	1.63	0.82	62.4	124.8^{*}

Table 2: Top word drifts selected based on DBE model and estimated by BBP. * and ** indicate p-value ≤ 0.05 and 0.01, respectively.

In Table 1, we show results from five illustrative tests, drawn from the top-100 word drifts estimated by the DBE model. We report words' drift ranks in DBE against their corresponding L2 distance, cosine similarity, KL divergence and Hotelling T^2 using the embeddings estimated in our BBP model. Based on the distance metrics that ignore the covariance matrix, these words do not appear to change much over time as their cosine similarities are fairly large and their L2 distances are relatively small with little variation across the five words. This suggests their mean vectors are projected into close space between 1940s and 2000s. However, by taking into account their uncertainty, we observe greater variation in both KL divergence and T^2 statistic. For example, "council" has the eighth largest drift in DBE by L2, but shows the largest T^2 statistic among the five words and is statistically significant at $\alpha = 0.01$. So too, the largest DBE drift ("uk") is insignificant once you take into account the covariance structure.

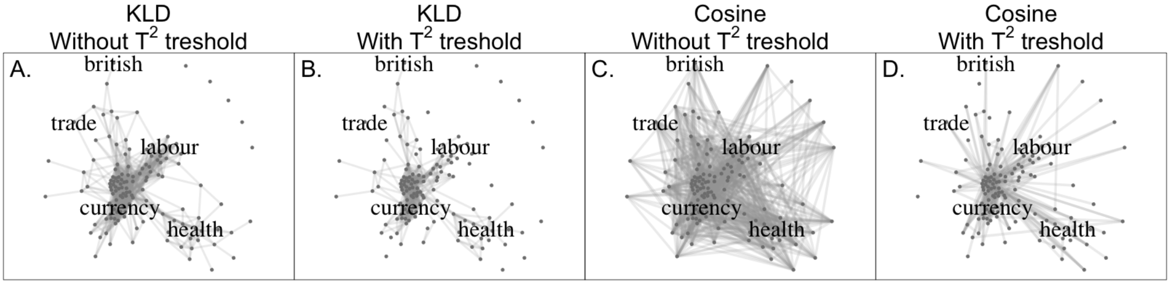

Figure 4: Semantic Graphs with KLD vs. Cosine Similarity

Cosine Similarity vs. KL Divergence In contrast to cosine distance, our proposed method allows computation of the KLD between two vectors that takes into account their covariance. Figure 2 presents semantic graphs estimated in the spirit of Hamilton et al. (2016a). The set of words is given by the union set of the 10 nearest neighbors, measured by cosine similarity and KLD, for the five seed words: "currency", "british", "health", "trade" and "labour". This results in 130 unique words including the seed words and we compute their pair-wise KLD matrix, W_{KL} and pairwise cosine similarity matrix, W_{cos}. We convert W_{KL} to a symmetric matrix as $W'_{KL} = (W_{KL} + W^T_{KL})/2$. Both W_{KL} and W_{cos} have dimensions of 130×130.

Edge weights in Figures 2.A and 2.B are computed by taking a sigmoid transformation of normalized entries in W'_{KL}, i.e., $\sigma(normalize(w'_{KL_{i,j}}))$. Edge weights in 2.C and 2.D are computed by $\arccos(w_{cos_{i,j}})$, following Hamilton et al. (2016a). Edges with weights below 90th percentiles are dropped for visual clarity. Note that with the same number of edges being eliminated, the KLD charts appear more clustered around seed words, implying that incorporating covariance matrix creates useful segregation of words within local contexts; graphs constructed via cosine similarity seem to disperse edge weights in a more diffuse manner.

T^2-based Significance In the context of uncertainty-aware word embeddings, we can use the T^2 statistic to filter out additional words from a nearest neighbor set. For instance, in Figure 2.B and 2.D, we drop edges for word pairs that fall below the 90th percentile of computed T^2 statistics. Filtering with Hotelling T^2 results in more sparse semantic graphs.

7 Conclusion

We proposed an uncertainty-aware conditional word embedding model that combines two ideas; (1) variational Bayesian learning for estimating parameter uncertainty, and (2) structured embeddings conditioned on covariates. This provides a principled direction to investigate hypothesis tests of word vectors in various forms. We evaluated various aspects of the proposed approach on U.K. Parliament speech records from 1935-2012. We believe the proposed approach will serve as a more rigorous tool in social science and other domains.

8 Acknowledgments

KC thanks the support by eBay, TenCent, NVIDIA and CIFAR. RH thanks the support by MINDS research group at Information Sciences Institute of University of California.

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. *arXiv*, 1505.05424.

Arthur Brazinskas, Serhii Havrylov, and Ivan Titov. 2017. Embedding words as distributions with a bayesian skip-gram model. *arXiv*, 1711.11027.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences (Preprint)*, pages 1–10.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016a. Inducing domain-specific sentiment lexicons from unlabeled corpora. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016b. Cultural shift or linguistic drift? comparing two computational measures of semantic change. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 2116. NIH Public Access.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016c. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv*, 1605.09096v4.

Harold Hotelling. 1931. The generalization of student's ratio. *The Annals of Mathematical Statistics*, 2(3):360–378.

Koichi Ito. 1956. Asymptotic formulae for the distribution of hotelling's generalized t_0^2 statistic. *The Annals of Mathematical Statistics*, 27(4):1091–1105.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Michael Laver and John Garry. 2000. Estimating policy positions from political texts. *American Journal of Political Science*, pages 619–634.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. 2006. A tutorial on energy-based learning. *Predicting structured data*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *arXiv*, 1310.4546.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ludovic Rheault, Kaspar Beelen, Christopher Cochrane, and Graeme Hirst. 2016. Measuring emotion in parliamentary debates with automated textual analysis. *PloS one*, 11(12):e0168843.

Maja Rudolph and David Blei. 2018. Dynamic embeddings for language evolution. In *WWW 2018: The 2018 Web Conference*, volume April 23–27, 2018, pages 1003–1011.

Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. 2016. Exponential family embeddings. In *Advances in Neural Information Processing Systems*, pages 478–486.

Catherine R Schenk. 2010. *The decline of sterling: managing the retreat of an international currency, 1945–1992*. Cambridge University Press.

Luke Vilnis and Andrew McCallum. 2017. Word representations via gaussian embedding. *arXiv*, 1412.6623.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, volume 11, pages 2764–2770.

Classifying Referential and Non-referential *It* Using Gaze

Victoria Yaneva, Le An Ha, Richard Evans, and Ruslan Mitkov
Research Institute in Information and Language Processing,
University of Wolverhampton, UK
`{v.yaneva, ha.l.a, r.j.evans, r.mitkov}@wlv.ac.uk`

Abstract

When processing a text, humans and machines must disambiguate between different uses of the pronoun *it*, including non-referential, nominal anaphoric or clause anaphoric ones. In this paper, we use eye-tracking data to learn how humans perform this disambiguation. We use this knowledge to improve the automatic classification of *it*. We show that by using gaze data and a POS-tagger we are able to significantly outperform a common baseline and classify between three categories of *it* with an accuracy comparable to that of linguistic-based approaches. In addition, the discriminatory power of specific gaze features informs the way humans process the pronoun, which, to the best of our knowledge, has not been explored using data from a natural reading task.

1 Introduction

Anaphora resolution is both one of the most important and one of the least developed tasks in Natural Language Processing (Lee et al., 2016). A particularly difficult case for anaphora resolution systems is the pronoun *it*, as it may refer to a specific noun phrase or an entire clause, or it may even refer to nothing at all, as in sentences 1 - 3 below[1].

1. *"I couldnt say exactly, sir, but it wasnt tea-time by a long way."* (Pleonastic *it* (non-referential)).
2. *Now, as to this quarrel. When was the first time you heard of it?* (Nominal anaphoric)
3. *You have been with your mistress many years, is it not so?* (Clause anaphoric[2].)

This phenomenon is not specific only to English; pronouns that can be used both referentially and non-referentially exist in a variety of language groups such as the pronoun *'het'* in Dutch, *'det'* in Danish, *'ello'* in Spanish, *'il'* in French, etc.

In NLP, there has been active research in the area of automatic classification of *it* during the past four decades but the issue is far from being solved (Section 2). According to corpus statistics, the pronoun it is by far the most frequently used pronoun in the English language (Li et al., 2009), and as recently as 2016, incorrect classification of different cases of *it* and their antecedents is highlighted as one of the major reasons for the failure of question-answering systems (Lee et al., 2016). In the state of the art, the best approaches to classifying *it* achieve between 2% and 15% improvement over a majority baseline when assigning examples of the pronoun to more than two classes.

In contrast with the extensive research in NLP, very little is known about the way humans approach the disambiguation of *it*. To the best of our knowledge, Foraker and McElree (2007) is the only study on this subject that uses online measures of reading. It proves empirically that the pronoun *it* is resolved more slowly and less accurately than gendered pronouns due to its ambiguity. So far there have been no studies investigating the subject using natural reading data as opposed to artificially created controlled sentences.

In this paper we approach these two problems as one and hypothesise that obtaining information on the way humans disambiguate the pronoun can improve its automatic classification.

We propose a new method for classifying the pronoun *it* that does not rely on linguistic processing. Instead, the model leverages knowledge about the way in which humans disambiguate the pronoun based on eye tracking data. We

---

[1] Extracted from the GECO corpus (Cop et al., 2016)

[2] Some authors also distinguish other, less-common types of the pronoun it such as proaction, cataphoric, discourse topic, and idiomatic, among others (Evans, 2001).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4896–4901
Brussels, Belgium, October 31 - November 4, 2018.

show that by using gaze data and a POS tagger we are able to achieve classification accuracy of three types of *it* that is comparable to the performance of linguistic-based approaches and that outperforms a common baseline with a statistically significant difference. In addition, examining the discriminatory power of specific gaze features provides valuable information about the human processing of the pronoun. We make our data, code and annotation available at `https://github.com/victoria-ianeva/It-Classification`. The GECO eye-tracking corpus is available at `http://expsy.ugent.be/downloads/geco/`.

2 Related Work

Gaze data in NLP While eye-movement data has been traditionally used to gain understanding of the cognitive processing of text, it was recently applied to a number of technical tasks such as part-of-speech tagging (Barrett et al., 2016), detection of multi-word expressions (Rohanian et al., 2017; Yaneva et al., 2017), sentence compression (Klerke et al., 2016), complex-word identification (Štajner et al., 2017), and sentiment analysis (Rotsztejn, 2018). Eye movements were also shown to carry valuable information about the reader and were used to detect specific conditions affecting reading such as autism (Yaneva et al., 2018, 2016) and dyslexia (Rello and Ballesteros, 2015). The motivation behind these approaches is two-fold. First, eye tracking is already making its way into everyday use with interfaces and devices that feature eye-tracking navigation (e.g. Windows Eye Control[3]). Second, linguistic annotation by gaze is faster than traditional annotation techniques, does not require trained annotators, and provides a language-independent approach that can be applied to under-resourced languages (Barrett et al., 2016). This is particularly interesting for the case of non-referential pronouns, as the phenomenon exists in different languages.

Classification of *it* The majority of the machine learning approaches to classifying the pronoun *it* in different languages are based on linguistic features capturing token, syntactic and semantic context. Different papers report varying majority baseline metrics (between 50% and 75%) depending on the annotated corpora, and an improvement over the majority baselines of between 2% and 15% for classification of more than two classes of *it* (Loáiciga et al., 2017; Uryupina et al., 2016; Lee et al., 2016; Müller, 2006; Boyd et al., 2005; Hoste et al., 2007; Evans, 2001). For example, Loáiciga et al. (2017) train a bidirectional recurrent neural network (RNN) to classify three classes of *it* in the ParCor corpus (Guillou et al., 2014) and compare its performance to a feature-based maximum entropy classifier. The RNN achieves accuracy of 62% compared to a majority baseline of 54% and is significantly outperformed by the linguistic-feature classifier which obtained 68.7% accuracy. Lee et al. (2016) compare several statistical models and report 75% accuracy for four-class classification over a majority baseline of around 62% using linguistic features and a stochastic adaptive gradient algorithm. They also report that experimenting with word embeddings did not lead to more accurate classification. So far, approaches using linguistic features still represent the state of the art in the classification of *it*.

3 Data

Corpus: The eye-tracking data was extracted from the GECO corpus (see Cop et al. (2016) for full corpus specifications) which is the largest and most recent eye-tracking corpus for English at present. The text of the corpus is a novel by Agatha Christie entitled "The Mysterious Affair at Styles", the English version of which contains 54,364 tokens and 5,012 unique types. The entire novel was read by 14 native English undergraduates from the University of Southampton using an eye-tracker with a sampling rate of 1 kHz.

Annotation scheme: A total of 1,052 instances of *it* were found in the corpus[4]. Each of the instances of *it* was annotated by two annotators and assigned to one of three categories: *Pleonastic, Nominal anaphoric* and *Clause anaphoric*, following the scheme used by Lee et al. (2016). The annotators were free to view as much of the previous text as necessary to decide on a label. The inter-annotator agreement for the three categories was $\kappa = 0.636$, $p < 0.0005$, indicating substantial agreement between the annotators. This number corresponds to a percentage agreement of 77.47% for the three categories and is comparable to the

---

[3] https://support.microsoft.com/en-gb/help/4043921/windows-10-get-started-eye-control

[4] This number does not include the possessive pronoun *its*. There are also several tokenisation errors in the GECO corpus (e.g. "it?..ah," and "it...the" misidentified as single tokens.). These cases were excluded.

	Annotat. 1	Annotat. 2	Final
Pleonastic	339 (33%)	406 (38%)	272 (33%)
Nom. anaph.	492 (46%)	527 (50%)	453 (56%)
Clause anaph.	221 (21%)	119 (11%)	89 (11%)

Table 1: Annotation categories

	Prev.	“It”	Next	It + Next
Early	61.1	60.3	61.7	61
Medium	60.9	60.5	60.6	61.2
Late	58.9	60.2	61	61.4

Table 2: Weighted F1 scores for an ablation study for different gaze feature groups over the Previous and Next word baseline (60.4)

81% agreement reported in Lee et al. (2016). We perceived adjudication between cases of disagreement (237 instances) to be extremely arbitrary, so those cases were excluded rather than resolved. Examples of such arbitrary cases include:

- *”Sit down here on the grass, do. It’s ever so much nicer.”* (nominal vs. clause anaphoric)
- *It’s a jolly good life taking it all round...if it weren’t for that fellow Alfred Inglethorp!”* (pleonastic vs. clause anaphoric)

The distribution of each class of the retained data by annotator is presented in Table 1. We make the full annotations of both annotators available.

4 Experiments

Overview In order to test the extent to which gaze data can help the classification of different cases of *it*, we trained and compared three separate classifiers. The first classifier is based on gaze features, the second one is based on linguistic features and finally, we trained a combined classifier using both gaze and linguistic features. We compared the performance of these classifiers to a majority baseline of 55.7 and to another baseline obtained by using the tokens surrounding the pronoun (previous and next word) as features (60.4). We also experimented with adding word embeddings[6] for the surrounding tokens as features. While the full exploration of word embeddings for the classification of *it* remains outside of the scope of this work, it would be interesting to explore whether the embeddings add value to the models by encoding information that was not otherwise captured.

		Ling	Gaze	Comb
EARLY	First_Run_Fixation_Count		*	
	First_Run_Fixation_%			*
	First_Fixation_Duration			
	First_Fixation_Visited_Count			†*
	First_Fix_Progressive			†*
MEDIUM	Second_Run_Fixation_Count			
	Second_Run_Fixation_%			*
	Second_Fixation_Duration		*	†*
	Second_Fixation_Run		*	*
	Gaze_Duration			†
LATE	Third_Run_Fixation_Count			
	Third_Run_Fixation_%			†
	Third_Fixation_Duration			†*
	Third_Fixation_Run		*	
	Last_Fixation_Duration		*	†*
	Last_Fixation_Run			
	Go_Past_Time			†
	Selective_Go_Past_Time			†
	Fixation_Count			†
	Fixation_%		*	†
	Total_Reading_Time			
	Total_Reading_Time_%		*	*
	Trial_Fixation_Count			*
	Trial_Total_Reading_Time			
	Spillover			
	Skip		*	
LINGUISTIC	Word_position	+		+
	# Preceding_NPs_in_sentence			
	# Preceding_NPs_in_paragraph			
	# Following_NPs_in_sentence			
	# NPs_in_the_sentence			
	# NPs_in_the_paragraph			+
	# Following_adject._in_sentence			
	Previous_verb	+		
	Following_adjective	+		+
	Following _verb	+		+
	POS in posit. L4, L3, L2, L1	+	+[5]	+
	POS in posit. R4, R3, R2, R1	+	+	+
	# Following_complementisers	+		+
	An_adjective_before_the_next_NP	+		+
	Words_until_next_complementiser	+		+
	Words_until_next_infinitive	+		+
	Words_until_next_preposition			+
	Words_until_next_ing_verb	+		+
	A_compl._before_the_next_NP			
	Immediately_preceding_preposit.			
BASIC	Previous_word	+	+	+
	Next_word	+	+	+
	Word_length		+	+
	Punctuation			+

Table 3: List of features and their inclusion in the different models. + refers to linguistic data, * to added values for the *It* + *Next* region, and †to gaze features for the previous word region. The features that do not have marks in the last three columns were not retained in any of the three best models.

Gaze features We use the gaze features as provided in GECO and we average the data from all 14 readers per token. We extract gaze data for each case of *it*, as well as for the preceding and following word. The full list of gaze features used in the

---

[5] Except L4 and L3

[6] 300-dimensional vectors from Google News obtained through word2vec: https://code.google.com/archive/p/word2vec/

experiments can be seen in Table 3.

Different eye-tracking measures (usually divided into *early* and *late*) are indicative of different aspects of cognitive processing. Early gaze measures such as *First_Fixation_Duration* give information about the early stages of lexical access and syntactic processing, while late measures such as *Total_Reading_Time* or *Number_of_Fixations* give information about processes such as textual integration and disambiguation (see Rayner et al. (2012) for a review). In Table 3, the distinction between *Early, Medium* and *Late* gaze features is mainly based on the run during which the fixations were made (i.e. whether the eyes were passing through the text for the first, second or third time). For each run we report count measures, percentage measures (as part of the trial) and duration measures (in milliseconds). Additional late features reported include *Last_Fixation_Duration* and *Last_Fixation_Run* (the run during which the last fixation in a given region occurred), *Total_Reading_Time* (in msec and %), and *Trial_Fixation_Count* (the overall number of fixations within the trial). *Go_Past_Time* refers to the summation of all fixation durations on the current word during the first pass. *Spillover* refers to the duration of the first fixation made on the next word after leaving the current word in the first run. Finally, a word is considered skipped if no fixation occurred during the first run (*Skip*). A complete legend explaining each feature can be found within the corpus metadata.

An ablation study on the contributions of individual groups of gaze features towards the classification of *it* is presented in Table 2.

Linguistic features We implemented a set of features originally proposed by Evans (2001) and subsequently used extensively in the studies presented in Section 2. In terms of features and categories of *it*, the study by Evans (2001) is the most fine-grained one we could find, classifying 7 categories of *it* with 69% accuracy. The set of features from Evans (2001) is presented in Table 3. These features synthesize information based on corpus studies of the pronoun *it* and thus aim to capture positional, part-of-speech and proximity information, as well as specific patterns of usage. For example, Evans (2001) notes that pleonastic pronouns rarely appear immediately after a prepositional word and that complementisers or adjectives often follow pleonastic instances. Another pattern that distinguishes the pleonastic use of *it* is associated with certain sequences of elements such as 'adjective + noun phrase' and 'complementiser + noun phrase' (Evans, 2001). Therefore, the linguistic features proposed by Evans (2001) and used in our experiments make possible the utilization of corpus-based knowledge for the automatic classification of *it*.

	P	**R**	**F1**
Baselines			
Majority baseline			55.7
Previous + Next word	62.1	63.9	60.4
Embeddings			
Prev. + Next Embed.	63.1	64.4	62.1
Linguistic models			
Full feature set	63.4	66	63.2
Best linguistic	**66.7**	**68.8**	**66.1***
Best linguistic + Embed.	**66.9**	**68.8**	**67.2***
Gaze-based models			
Basic + POS	63.3	64.5	62.2
Select. Gaze + Basic	65.8	66.8	64.2
Select. Gaze + Basic + POS	**66.6**	**67.9**	**65.6***
S. Gaze + Bas. + POS + Embed.	**66.3**	**68.8**	**66.7***
Combined model			
Best Gaze + Ling	**71**	**71.5**	**68.8***
Best Gaze + Ling. + Embed.	67.5	69.3	671*

Table 4: Precision, Recall and Weighted F1 for the various classifiers. The * symbol marks statistical significance compared to the baseline model of Previous + Next Word (60.4).

Classification We use simple logistic regression as implemented in WEKA (Hall et al., 2009) with 10-fold cross validation and a random seed parameter 20. Since logistic regression is an interpretable method, we are able to assess the performance of individual features and gain insight into the psycholinguistic processing of the pronoun.

We experimented with gaze features for the individual words but as gaze data is inherently very noisy, we found that smoothing the features by adding the ones that correspond to the pronoun and the next word stabilized the results. Adding the gaze features for the previous word significantly reduced the performance but using them separately in a model with the added *It* + *Next* features maximised our results. For the role of individual features in the models see Table 3.

In order to account for class imbalance we compute and report a weighted F1 score, as opposed to the harmonic mean between precision and recall. First, the F1 for each class is weighted by multiplying it by the number of instances in the class. Then the F1 scores for all classes are summed up and divided by the total number of instances. The resulting weighted F1 score is lower than the traditionally reported mean F1 score, but it represents the effects of class imbalance more accurately.

5 Results and Discussion

The results from the classification experiments presented in Table 4 have implications for language processing by both humans and machines.

From the NLP perspective, the potential of gaze data to not only improve but also, to a certain extent, substitute text processing approaches is an exciting new frontier. Our results show that the gaze-based classifier performs on par with the one using linguistic features and both of them perform significantly better[7] than the baseline of 60.4. An improvement of 13% over the majority baseline is achieved when combining the two but this difference is not a significant improvement over the individual best classifiers. A possible reason for this is that the gaze data and linguistic features encode similar information about the disambiguation of *it* and adding them together leads to overlap instead of complementation. In all three classifiers, the clause-anaphoric class was consistently predicted with lowest accuracy (Table 5), which is not surprising given that it only accounts for 11% of the retained data. In line with the observation of Lee et al. (2016) (Section 2), the embeddings do not show a stable contribution. In our case, this is likely related to the small amount of data, to which we add 300 dimensions per word.

Overall, the improvement achieved by the classifiers is comparable with the current state-of-the-art (Section 2). It is important to note that this is the first study to use text from the domain of literature and that this may have influenced the extraction of the linguistic features. At the same time, literature can be regarded as a more challenging domain than the declarative texts used in previous research, owing to the creative use of language.

From a psycholinguistic perspective, we provide evidence that, indeed, the three classes of *it* are processed differently. We observe that medium and late gaze features related to disambiguation are more discriminative than the early ones. For example, measures such as first fixation duration were not included in any of the models, while revisits as late as the third run (the third time the eyes pass over the region of interest) occurred in these regions and provided a strong signal. Particularly useful features were the durations of the second and last fixations, as well as the information about the run (pass) during which they occur.

NomAnaph	ClauseAnaph	Pleon	
395	2	56	NomAnaph
53	11	25	ClauseAnaph
93	3	176	Pleon

Table 5: Confusion matrix for the best combined model (Weighted F1 = 68.8)

The significance of these features in the best classifiers somewhat contradicts the ablation study presented in Table 2. According to that table, early processing features for the preceding and next words are expected to outperform the late ones. A possible explanation for this are predictability and spillover effects, as the pronoun *it* is both highly predictable and easy to skip, because of its high frequency and shortness. Indeed, the gaze features from the it-region itself are not as useful as the ones from the surrounding words.

The results from this study showed that: i) gaze features encode information about the way humans disambiguate the pronoun *it*, ii) that this information partially overlaps with the information carried by linguistic features, and that iii) gaze can be used for automatic classification of the pronoun with accuracy comparable to that of linguistic-based approaches. In our future work we will attempt to identify specific patterns of cognitive processing for the individual classes, as well as explore factors related to the readers.

6 Conclusion

We presented the first study on the use of gaze data for disambiguating categories of *it*, exploring a wide range of gaze and linguistic features. The model based on gaze features and part-of-speech information achieved accuracy similar to that of the linguistic-based model and state-of-the-art systems, without the need for text processing. Late gaze features emerged as the most discriminative ones, with disambiguation effort indicators as late as third pass revisits.

---

[7] Gaze + Basic + POS: $p = 0.029$, 95% CI (0.509; 9.858) ; Best Linguistic: $p = 0.0017$, 95% CI (1.01; 10.34); Best Gaze + Ling: $p = 0.0004$, 95% CI (3.75; 12.99). The CI indicate the difference in %

References

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

Adriane Boyd, Whitney Gegg-Harrison, and Donna Byron. 2005. Identifying non-referential it: a machine learning approach incorporating linguistically motivated patterns. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, pages 40–47. Association for Computational Linguistics.

Uschi Cop, Nicolas Dirix, Denis Drieghe, and Wouter Duyck. 2016. Presenting GECO: An eyetracking corpus of monolingual and bilingual sentence reading. *Behavior research methods*, pages 1–14.

Richard Evans. 2001. Applying machine learning toward an automatic classification of it. *Literary and linguistic computing*, 16(1):45–58.

Stephani Foraker and Brian McElree. 2007. The role of prominence in pronoun resolution: Active versus passive representations. *Journal of Memory and Language*, 56(3):357–383.

Liane Guillou, Christian Hardmeier, Aaron Smith, Jörg Tiedemann, and Bonnie Webber. 2014. Parcor 1.0: A parallel pronoun-coreference corpus to support statistical mt. In *9th International Conference on Language Resources and Evaluation (LREC), MAY 26-31, 2014, Reykjavik, ICELAND*, pages 3191–3198. European Language Resources Association.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Veronique Hoste, Iris Hendrickx, and Walter Daelemans. 2007. Disambiguation of the neuter pronoun and its effect on pronominal coreference resolution. In *International Conference on Text, Speech and Dialogue*, pages 48–55. Springer.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. *arXiv preprint arXiv:1604.03357*.

Timothy Lee, Alex Lutz, and Jinho D Choi. 2016. Qa-it: Classifying non-referential it for question answer pairs. In *Proceedings of the ACL 2016 Student Research Workshop*, pages 132–137.

Yifan Li, Petr Musilek, Marek Reformat, and Loren Wyard-Scott. 2009. Identification of pleonastic it using the web. *Journal of Artificial Intelligence Research*, 34:339–389.

Sharid Loáiciga, Liane Guillou, and Christian Hardmeier. 2017. What is it? disambiguating the different readings of the pronoun it. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1325–1331.

Christoph Müller. 2006. Automatic detection of non-referential it in spoken multi-party dialog. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Keith Rayner, Alexander Pollatsek, Jane Ashby, and Charles Clifton Jr. 2012. *Psychology of reading*. Psychology Press.

Luz Rello and Miguel Ballesteros. 2015. Detecting readers with dyslexia using machine learning with eye tracking measures. In *Proceedings of the 12th Web for All Conference*, page 16. ACM.

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

Jonathan Rotsztejn. 2018. Learning from cognitive features to support natural language processing tasks. Master's thesis, ETH Zurich.

Sanja Štajner, Victoria Yaneva, Ruslan Mitkov, and Simone Paolo Ponzetto. 2017. Effects of lexical properties on viewing time per word in autistic and neurotypical readers. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 271–281.

Olga Uryupina, Mijail Kabadjov, and Massimo Poesio. 2016. Detecting non-reference and non-anaphoricity. In *Anaphora Resolution*, pages 369–392. Springer.

Victoria Yaneva, Le An Ha, Sukru Eraslan, Yeliz Yesilada, and Ruslan Mitkov. 2018. Detecting autism based on eye-tracking data from web searching tasks. In *Proceedings of the Internet of Accessible Things*. ACM.

Victoria Yaneva, Shiva Taslimipoor, Omid Rohanian, and Le An Ha. 2017. Cognitive processing of multiword expressions in native and non-native speakers of english: Evidence from gaze data. In *International Conference on Computational and Corpus-Based Phraseology*, pages 363–379. Springer.

Victoria Yaneva, Irina P Temnikova, and Ruslan Mitkov. 2016. A corpus of text data and gaze fixations from autistic and non-autistic adults. In *Proceedings of the 10th Edition of the Language Resources and Evaluation Conference (LREC)*.

State-of-the-art Chinese Word Segmentation with Bi-LSTMs

Ji Ma **Kuzman Ganchev** **David Weiss**
Google AI Language
{maji,kuzman,djweiss}@google.com

Abstract

A wide variety of neural-network architectures have been proposed for the task of Chinese word segmentation. Surprisingly, we find that a bidirectional LSTM model, when combined with standard deep learning techniques and best practices, can achieve better accuracy on many of the popular datasets as compared to models based on more complex neural-network architectures. Furthermore, our error analysis shows that out-of-vocabulary words remain challenging for neural-network models, and many of the remaining errors are unlikely to be fixed through architecture changes. Instead, more effort should be made on exploring resources for further improvement.

1 Introduction

Neural networks have become ubiquitous in natural language processing. For the word segmentation task, there has been a growing body of work exploring novel neural network architectures for learning useful representation and thus better segmentation prediction (Pei et al., 2014; Ma and Hinrichs, 2015; Zhang et al., 2016a; Liu et al., 2016; Cai et al., 2017; Wang and Xu, 2017).

We show that properly training and tuning a relatively simple architecture with a minimal feature set and greedy search achieves state-of-the-art accuracies and beats more complex neural-network architectures. Specifically, the model itself is a straightforward stacked bidirectional LSTM (Figure 1) with just two input features at each position (character and bigram). We use three widely recognized techniques to get the most performance out of the model: pre-trained embeddings (Yang et al., 2017; Zhou et al., 2017), dropout (Srivastava et al., 2014), and hyperparameter tuning (Weiss et al., 2015; Melis et al., 2018). These results have important ramifications for further model development. Unless best practices are followed, it is difficult to compare the impact of modeling decisions, as differences between models are masked by choice of hyperparameters or initialization.

In addition to the simpler model we present, we also aim to provide useful guidance for future research by examining the errors that the model makes. About a third of the errors are due to annotation inconsistency, and these can only be eliminated with manual annotation. The other two thirds are those due to out-of-vocabulary words and those requiring semantic clues not present in the training data. Some of these errors will be almost impossible to solve with different model architectures. For example, while 抽象概念 (abstract concept) appears as one word at test time, any model trained only on the MSR dataset will segment it as two words: 抽象 (abstract) and 概念 (concept), which are seen in the training set 28 and 90 times, respectively, and never together. Thus, we expect that iterating on model architectures will give diminishing returns, while leveraging external resources such as unlabeled data or lexicons is a more promising direction.

In sum, this work contributes two significant pieces of evidence to guide further development in Chinese word segmentation. First, comparing different model architectures requires careful tuning and application of best practices in order to obtain rigorous comparisons. Second, iterating on neural architectures may be insufficient to solve the remaining classes of segmentation errors without further efforts in data collection.

2 Model

Our model is relatively simple. Our approach uses long short-term memory neural networks architectures (LSTM) since previous work has found success with these models (Chen et al., 2015; Zhou et al., 2017, inter alia). We use two features: uni-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4902–4908
Brussels, Belgium, October 31 - November 4, 2018.

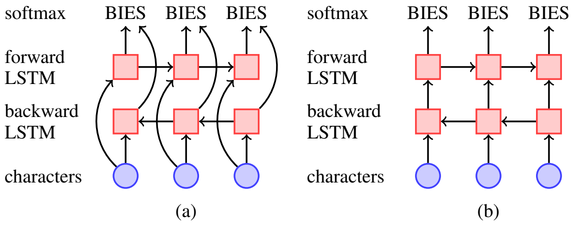

Figure 1: Bi-LSTM models: (a) non-stacking, (b) stacking. Blue circles are input (char and char bigram) embeddings. Red squares are LSTM cells. BIES is a 4-way softmax.

grams and bi-grams of characters at each position. These features are embedded, concatenated, and fed into a stacked bidirectional LSTM (see Figure 1) with two total layers of 256 hidden units each. The softmax layer of the bi-LSTM predicts Begin/Inside/End/Single tags encoding the relationship from characters to segmented words.

In the next sections we describe the best practices we used to achieve state-of-the-art performance from this architecture. Note that all of these practices and techniques are derived from related work, which we describe.

Recurrent Dropout. Contrary to the recommendation of Zaremba et al. (2014), we apply dropout to the recurrent connections of our LSTMs, and we see similar improvements when following the recipe of Gal and Ghahramani (2016) or simply sample a new dropout mask at every recurrent connection.

Hyperparameters. We use the momentum-based averaged SGD procedure from (Weiss et al., 2015) to train the model, with few additions. We normalized each gradient to be at most unit norm, and used asynchronous SGD updates to speed up training time. For each configuration we evaluated, we trained different settings of a manually tuned hyperparameter grid, varying the initial learning rate, learning rate schedule, and input and recurrent dropout rates. We fixed the momentum parameter $\mu = 0.95$. The full list of hyperparameters is given in Table 2. We show the impact of this tuning procedure in Table 7, which we found was crucial to measure the best performance of the simple architecture.

Pretrained Embeddings. Pre-training embedding matrices from automatically gathered data is a powerful technique that has been applied to many NLP problems for several years (e.g. Collobert et al. (2011); Mikolov et al. (2013)). We pretrain the character embeddings and character-bigram embeddings using wang2vec[1] (Ling et al., 2015), which modifies word2vec by incorporating character/bigram order information during training. Note that this idea has been used in segmentation previously by Zhou et al. (2017), but they also augment the contexts by adding the predictions of a baseline segmenter as an additional context. We experimented with both treating the pretrained embeddings as constants or fine-tuning on the particular datasets.

	Train	Development	Test
AS	4,903,564	546,017	122,610
CTIYU	1,309,208	146,422	40,936
MSR	2,132,480	235,911	106,873
CTB6	641,368	59,954	81,578
CTB7	950,138	59,954	81,578
PKU	994,822	115,125	104,372
UD	98,608	12,663	12,012

Table 1: Statistics of training, development and test set.

Other Related Work. Recently, a number of different neural network based models have been proposed for word segmentation task. One common approach is to learn word representation through the characters of that word. For example, Liu et al. (2016) runs bi-directional LSTM over characters of the word candidate and then concatenate bi-directional LSTM outputs at both end points. Cai et al. (2017) adopts a gating mechanism to control relative importance of each character in the word candidate.

Besides modeling word representation directly, sequential labeling is another popular approach. For instance, Zheng et al. (2013) and Pei et al. (2014) predict the label of a character based context of a fixed sized local window. Chen et al. (2015) extends the approach by using LSTMs to capture potential long distance information. Both Chen et al. (2015) and Pei et al. (2014) use a transition matrix to model interaction between adjacent tags. Zhou et al. (2017) conduct rigorous comparison and show that such transition matrix rarely improves accuracy. Our model is similar to Zhou et al. (2017), except that we stack the backward LSTM on top of the forward one, which improves accuracy as shown in later section.

Our model is also trained via a simple maximum likelihood objective. In contrast, other state-

---

[1] `https://github.com/wlin12/wang2vec`

	AS	CITYU	CTB6	CTB7	MSR	PKU	UD
Liu et al. (2016)	—	—	95.9	—	97.3	**96.8**	
Yang et al. (2017)	95.7	96.9	96.2	—	97.5	96.3	—
Zhou et al. (2017)	—	—	96.2	—	97.8	96.0	—
Cai et al. (2017)	—	95.6	—	—	97.1	95.8	—
Kurita et al. (2017)	—	—	—	96.2	—	—	—
Chen et al. (2017)	94.6	95.6	96.2	—	96.0	94.3	—
Qian and Liu (2017)	—	—	—	—	—	—	94.6
Wang and Xu (2017)	—	—	—	—	98.0	96.5	—
Ours (fix embedding)	**96.2**	**97.2**	**96.7**	**96.6**	97.4	96.1	**96.9**
Ours (update embedding)	96.0	96.8	96.3	96.0	**98.1**	96.1	96.0

Table 2: The state of the art performance on different datasets. For Kurita et al. (2017) and Chen et al. (2017) we report their best systems (segpos+dep and Model-I-ADV respectively). †Not directly comparable to the rest of the table due to the usage of an external dictionary. Our bolded results are significantly better ($p < 0.05$ bootstrap resampling) except on MSR.

	AS	CITYU	CTB6	CTB7	MSR	PKU	UD
Liu et al. (2016)	—	—	94.6	—	94.8	94.9	—
Zhou et al. (2017)	—	—	94.9	—	97.2	95.0	—
Cai et al. (2017)	95.2	95.4	—	—	97.0	**95.4**	—
Wang and Xu (2017)	—	—	—	—	96.7	94.7	—
Ours	**95.5**	**95.7**	**95.5**	**95.6**	**97.5**	**95.4**	**94.6**

Table 3: Performance of recent neural network based models without using pretrained embeddings. Our model's wins are statsitically significantly better than prior work ($p < 0.05$ bootstrap resampling), except on PKU.

of-the-art models use a non-greedy approach to training and inference, e.g. Yang et al. (2017) and Zhang et al. (2016b).

3 Experiments

Data. We conduct experiments on the following datasets: Chinese Penn Treebank 6.0 (CTB6) with data split according the official document; Chinese Penn Treebank 7.0 (CTB7) with recommended data split (Wang et al., 2011); Chinese Universal Treebank (UD) from the Conll2017 shared task (Zeman et al., 2017) with the official data split; Dataset from SIGHAN 2005 bake-off task (Emerson, 2005). Table 1 shows statistics of each data set. For each of the SIGHAN 2005 dataset, we randomly select 10% training data as development set. We convert all digits, punctuation and Latin letters to half-width, to handle full/half-width mismatch between training and test set. We train and evaluate a model for each of the dataset, rather than train one model on the union of all dataset. Following Yang et al. (2017), we convert AS and CITYU to simplified Chinese.

3.1 Main Results

Table 2 contains the state-of-the-art results from recent neural network based models, together with the performance of our model. Table 3 contains results achieved without using any pretrained embeddings.

Our model achieves the best results among NN models on 6/7 datasets. In addition, while the majority of datasets work the best if the pretrained embedding matrix is treated as constant, the MSR dataset is an outlier: fine-tuning embeddings yields a very large improvement. We observe that the likely cause is a low OOV rate in the MSR evaluation set compared to other datasets.

3.2 Ablation Experiments

To see which decisions had the greatest impact on the result, we performed ablation experiments on the holdout sets of the different corpora. Starting with our proposed system[2], we remove one decision, perform hyperparameter tuning, and see the change in performance. The results are summarized in Table 6. Negative numbers in Table 6 correspond to decreases in performance for the ablated system. Note that although each of the components help performance on average, there are cases where we observe no impact. For example using recurrent dropout on AS and MSR rarely af-

---

[2] Based on development set accuracy, we keep the pretrained embedding fixed for all datasets except MSR and AS.

Parameter	Values
Char embedding size	[64]
Bigram embedding size	[16, 32, 64]
Learning rate	[0.04, 0.035, 0.03]
Decay steps	[32K, 48K, 64K]
Input dropout rate	[0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6]
LSTM dropout rate	[0.1, 0.2, 0.3, 0.4]

Table 4: Hyperparameter settings.

	AS	CITYU	CTB6	CTB7	MSR	PKU	UD
OOV %	4.2	7.5	5.6	5.0	2.7	3.6	12.4
Recall % (random embedding)	65.7	75.1	73.4	74.1	71.0	66.0	81.1
Recall % (pretrain embedding)	70.7	87.5	85.4	85.6	80.0	78.8	89.7

Table 5: Test set OOV rate, together with OOV recall achieved with randomly initialized and pretrained embeddings, respectively.

fects accuracy.

We next investigate how important the hyperparameter tuning is to this ablation. In the main result, we tuned each model separately for each dataset. What if instead, each model used a single hyperparameter configuration for all datasets? In Table 7, we compare fully tuned models with those that share hyperparameter configurations across dataset for three settings of the model. We can see that hyperparameter tuning consistently improves model accuracy across all settings.

3.3 Error Analysis

In order to guide future research on Chinese word segmentaion, it is important to understand the types of errors that the system is making. To get a sense of this, we randomly selected 54 and 50 errors from the CTB-6 and MSR test set, respectively. We then manually analyzed them.

The model learns to remember words it has seen, especially for high frequency words. It also learns the notion of prefixes/suffixes, which aids predicting OOV words, a major source of segmentation errors (Huang and Zhao, 2007). Using pretrained embeddings enables the model to expand the set of prefixes/suffixes through their nearest neighbors in the embedding spaces, and therefore further improve OOV recall (on average, using pretrained embeddings contributes to 10% OOV recall improvement, also see Table 5 for more details).

Nevertheless, OOV remains challenging especially for those that can be divided into words frequently seen in the training data, and most (37 out of 43) of the oversegmentation errors are due to this. For instance, the model incorrectly segmented the OOV word 抽象概念 (abstract concept) as 抽象 (abstract) and 概念 (concept). 抽象 and 概念 are seen in the training set for 28 times and 90 times, respectively. Unless high coverage dictionaries are used, it is difficult for any supervised model to learn not to follow this trend in the training data.

In addition, the model sometimes struggles when a prefix/suffix can also be a word by itself. For instance, 权 (right/power) frequently serves as a suffix, such as 管理权 (right of management), 立法权 (right of legislation) and 终审权 (right of final judgment). When the model encounters 下放 (delegate/transfer) 权(power), it incorrectly merges them together.

Similarly, the model segments 居 (in/at) + 中 (middle) as 居中 (in the middle), since the training data contains words such as 居首 (in the first place) and 居次 (in the second place). This example also hints at the ambiguity of word delineation in Chinese, and explains the difficulty in keeping annotations consistent.

As another example, 县 is often attached to another proper noun to become a new word, e.g., 高雄 (Kaohsiung) + 县 becomes 高雄县 (county of Kaohsiung), 新竹(Hsinchu) + 县 becomes 新竹县 (county of Hsinchu). When seeing 银行县支行 (bank's county branch), which should be 银行 (bank) + 县支行 (county branch), the model outputs 银行县 + 支行 (i.e. a county named bank).

System	AS	CITYU	CTB6	CTB7	MSR	PKU	UD	Average
This work	98.03	98.22	97.06	97.07	98.48	97.95	97.00	97.69
-LSTM dropout	+0.03	-0.33	-0.31	-0.24	+0.04	-0.29	-0.76	-0.35
-stacked bi-LSTM	-0.13	-0.20	-0.15	-0.14	-0.17	-0.17	-0.39	-0.27
-pretrain	-0.13	-0.23	-0.94	-0.74	-0.45	-0.27	-2.73	-0.78

Table 6: Ablation results on development data. Top row: absolute performance of our system. Other rows: difference relative to the top row.

System	Fully tuned	Avg
This work	97.69	97.49
-Stacked	97.41	97.16
-Pretraining	96.90	96.81

Table 7: Hyperparameter ablation experiments. “Fully tuned” indicates per-system tuning for each dataset. “Avg” is the best setting when averaging across datasets.

corpus	tokens	inconsistency %
AS	4,903,564	1.31
CITYU	1,309,208	0.62
CTB6	641,368	1.27
CTB7	950,138	1.64
MSR	2,132,480	0.28
PKU	994,822	0.53
UD	98,608	0.46

Table 8: Automatically computed inconsistency in the corpus training data. See text for methodology.

Fixing the above errors requires semantic level knowledge such as ‘Bank’ (银行) is unlikely to be the name of a county (县), and likewise, transfer power (下放权) is not a type of right (权).

Previous work (Huang and Zhao, 2007) also pointed out that OOV is a major obstacle to achieving high segmentation accuracy. They also mentioned that machine learning approaches together with character-based features are more promising in solving OOV problem than rule based methods. Our analysis indicate that learning from the training corpus alone can hardly solve the above mentioned errors. Exploring other sources of knowledge is essential for further improvement. One potential way to acquire such knowledge is to use a language model that is trained on a large scale corpus (Peters et al., 2018). We leave this to future investigation.

Unfortunately, a third (34 out of 104) of the errors we have looked at were due to annotation inconsistency. For example, 建筑系 (Department of Architecture) is once annotated as 建筑 (Architecture) + 系 (Department) and once as 建筑系 under exactly the same context 建筑系教授喻肇青 (Zhaoqing Yu, professor of Architecture). 高新技术 (advanced technology) is annotated as 高 (advanced) + 新 (new) + 技术 (technology) for 37 times, and is annotated as 高新 (advanced and new) + 技术 (technology) for 19 times.

In order to augment the manual verification we performed above, we also wrote a script to automatically find inconsistent annotations in the data. Since this is an automatic script, it cannot distinguish between genuine ambiguity and inconsistent annotations. The heuristic we use is the following: for all word bigrams in the training data, we see if they also occur as single words or word trigrams. We ignore the dominant analysis and count the number of occurrences of the less frequent analyses and report this number as a fraction of the number of tokens in the corpus. Table 8 shows the results of running the script. We see that the AS corpus is the least consistent (according to this heuristic) while MSR is the most consistent. This might explain why both our system and prior work have relatively low performance on AS even though this has the largest training set. By contrast results are much stronger on MSR, and this might be in part because it is more consistently annotated. The ordering of corpora by inconsistency roughly mirrors their ordering by accuracy.

4 Conclusion

In this work, we showed that further research in Chinese segmentation must overcome two key challenges: (1) rigorous tuning and testing of deep learning architectures and (2) more effort should be made on exploring resources for further performance gain.

References

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 608–615. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206, Lisbon, Portugal. Association for Computational Linguistics.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1193–1203, Vancouver, Canada. Association for Computational Linguistics.

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

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc.

Chang-ning Huang and Hai Zhao. 2007. Chinese word segmentation: A decade review. 21(3):8.

Shuhei Kurita, Daisuke Kawahara, and Sadao Kurohashi. 2017. Neural joint model for transition-based chinese syntactic analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1204–1214. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, Denver, Colorado. Association for Computational Linguistics.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. *CoRR*, abs/1604.05499.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate linear-time chinese word segmentation via embedding matching. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1733–1743, Beijing, China. Association for Computational Linguistics.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 293–303. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Xian Qian and Yang Liu. 2017. A non-dnn feature engineering approach to dependency parsing – fbaml at conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 143–151. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Chunqi Wang and Bo Xu. 2017. Convolutional neural network with word embeddings for chinese word segmentation. *CoRR*, abs/1711.04411.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 309–317, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 323–333.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016a. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016b. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 421–431.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657. Association for Computational Linguistics.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, XIN-YU DAI, and Jiajun Chen. 2017. Word-context character embeddings for chinese word segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 760–766, Copenhagen, Denmark. Association for Computational Linguistics.

Sanskrit Sandhi Splitting using $seq2(seq)^2$

Rahul Aralikatte
IBM Research

Neelamadhav Gantayat
IBM Research

Naveen Panwar
IBM Research

{rahul.a.r, neelamadhav, naveen.panwar}@in.ibm.com

Anush Sankaran
IBM Research
anussank@in.ibm.com

Senthil Mani
IBM Research
sentmani@in.ibm.com

Abstract

In Sanskrit, small words (morphemes) are combined to form compound words through a process known as *Sandhi*. Sandhi splitting is the process of splitting a given compound word into its constituent morphemes. Although rules governing word splitting exists in the language, it is highly challenging to identify the location of the splits in a compound word. Though existing Sandhi splitting systems incorporate these pre-defined splitting rules, they have a low accuracy as the same compound word might be broken down in multiple ways to provide syntactically correct splits.

In this research, we propose a novel deep learning architecture called Double Decoder RNN (DD-RNN), which (i) predicts the location of the split(s) with 95% accuracy, and (ii) predicts the constituent words (learning the Sandhi splitting rules) with 79.5% accuracy, outperforming the state-of-art by 20%. Additionally, we show the generalization capability of our deep learning model, by showing competitive results in the problem of Chinese word segmentation, as well.

1 Introduction

Compound word formation in Sanskrit is governed by a set of deterministic rules following a well-defined structure described in *Pāṇini's Aṣṭādhyāyī*, a seminal work on Sanskrit grammar. The process of merging two or more morphemes to form a word in Sanskrit is called *Sandhi* and the process of breaking a compound word into its constituent morphemes is called *Sandhi splitting*. In Japanese, *Rendaku* ('sequential voicing') is similar to Sandhi. For example, *'origami'* consists of *'ori'* (paper) + *'kami'* (folding), where *'kami'* changes to *'gami'* due to *Rendaku*.

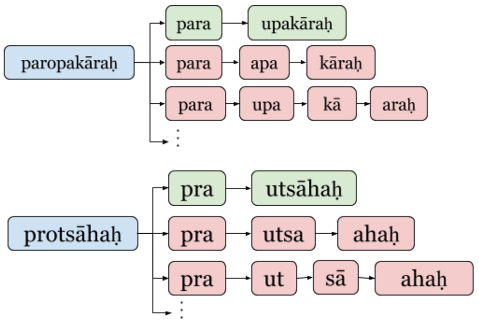

Figure 1: Different possible splits for the word *paropakāraḥ* and *protsāhaḥ*, provided by a standard Sandhi splitter.

Learning the process of sandhi splitting for Sanskrit could provide linguistic insights into the formation of words in a wide-variety of Dravidian languages. From an NLP perspective, automated learning of word formations in Sanskrit could provide a framework for learning word organization in other Indian languages, as well (Bharati et al., 2006). In literature, past works have explored sandhi splitting (Gillon, 2009) (Kulkarni and Shukl, 2009), as a rule based problem by applying the rules from *Aṣṭādhyāyī* in a brute force manner. Consider the example in Figure 1 illustrating the different possible splits of a compound word *paropakāraḥ*. While the correct split is *para* + *upakāraḥ*, other forms of splits such as, *para* + *apa* + *kāraḥ* are syntactically possible while semantically incorrect[1]. Thus, knowing all the rules of splitting is insufficient and it is essential to identify the location(s) of split(s) in a given compound word.

In this research, we propose an approach for au-

---

[1] Different syntactic splits given by one of the popular Sandhi splitters: `https://goo.gl/0M5CPS` and `https://goo.gl/JHnpJw`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4909–4914
Brussels, Belgium, October 31 - November 4, 2018.

tomated generation of split words by first learning the potential split locations in a compound word. We use a deep bi-directional character RNN encoder and two decoders with attention, ***seq*2(*seq*)^2**. The accuracy of our approach on the benchmark dataset for split location prediction is 95% and for split words prediction is 79.5% respectively. To the best of our knowledge, this is the first research work to explore deep learning techniques for the problem of Sanskrit Sandhi splitting, along with producing state-of-art results. Additionally, we show the performance of our proposed model for Chinese word segmentation to demonstrate the model's generalization capability.

2 *seq*2(*seq*)^2: Model Description

In this section, we present our double decoder model to address the Sandhi splitting problem. We first outline the issues with basic deep learning architectures and conceptually highlight the advantages of the double decoder model.

2.1 Issues with standard architectures

Consider an example of splitting a sequence *abcdefg* as *abcdx* + *efg*. The primary task is to identify *d* as the split location. Further, for a given location *d* in the character sequence, the algorithm should take into account (i) the context of character sequence *abc*, (ii) the immediate previous character *c*, (iii) the immediate succeeding character *e*, to make an effective split. For such sequence learning problems, RNNs have become the most popular deep learning model (Pascanu et al., 2013) (Sak et al., 2014).

A basic RNN encoder-decoder model (Cho et al., 2014) with LSTM units (Hochreiter and Schmidhuber, 1997), similar to a machine translation model, was trained initially. The compound word's characters is fed as input to the encoder and is translated to a sequence of characters representing the split words ('+' symbol acts as a separator between the generated split words). However, the model did not yield adequate performance as it encoded only the context of the characters that appeared before the potential split location(s). Though we tried making the encoder bi-directional (referred to as B-RNN), the model's performance only improved marginally. Adding global attention (referred to as B-RNN-A) to the decoder enabled the model to attend to the characters surrounding the potential split location(s) and improved the split prediction performance, making it comparable with some of the best performing tools in the literature.

2.2 Double Decoder RNN (DD-RNN) model

The critical part of learning to split compound words is to correctly identify the location(s) of the split(s). Therefore, we added a two decoders to our bi-directional encoder-decoder model: (i) *location decoder* which learns to predict the split locations and (ii) *character decoder* which generates the split words. A compound word is fed into the encoder character by character. Each character's embedding x_i is passed to the encoders LSTM units. There are two LSTM layers which encode the word, one in forward direction and the other backward. The encoded context vector e_i is then passed to a global attention layer.

In the first phase of training, only the *location decoder* is trained and the *character decoder* is frozen. The character embeddings are learned from scratch in this phase along with the attention weights and other parameters. Here, the model learns to identify the split locations. For example, if the inputs are the embeddings for the compound word *protsāhaḥ*, the location decoder will generate a binary vector $[0, 0, 1, 0, 0, 0, 0, 0, 0]$ which indicates that the split occurs between the third and fourth characters. In the second phase, the *location decoder* is frozen and the *character decoder* is trained. The encoder and attention weights are allowed to be fine-tuned. This decoder learns the underlying rules of Sandhi splitting. Since the attention layer is already pre-trained to identify potential split locations in the previous phase, the *character decoder* can use this context and learn to split the words more accurately. For example, for the same input word *protsāhaḥ*, the *character decoder* will generate [p, r, a, +, u, t, s, ā, h, a, ḥ] as the output. Here the character *o* is split into two characters *a* and *u*.

In both the training phases, we use negative log likelihood as the loss function. Let X be the sequence of the input compound word's characters and Y be the binary vector which indicates the location of the split(s) in the first phase and the true target sequence of characters which form the split words in the second phase. If $Y = y_1, y_2, ..., y_n$, then the loss function is defined as:

$$loss = -\sum_{i=1}^{|Y|} \log P(y_i | y_{i-1}, \cdots, y_1, X)$$

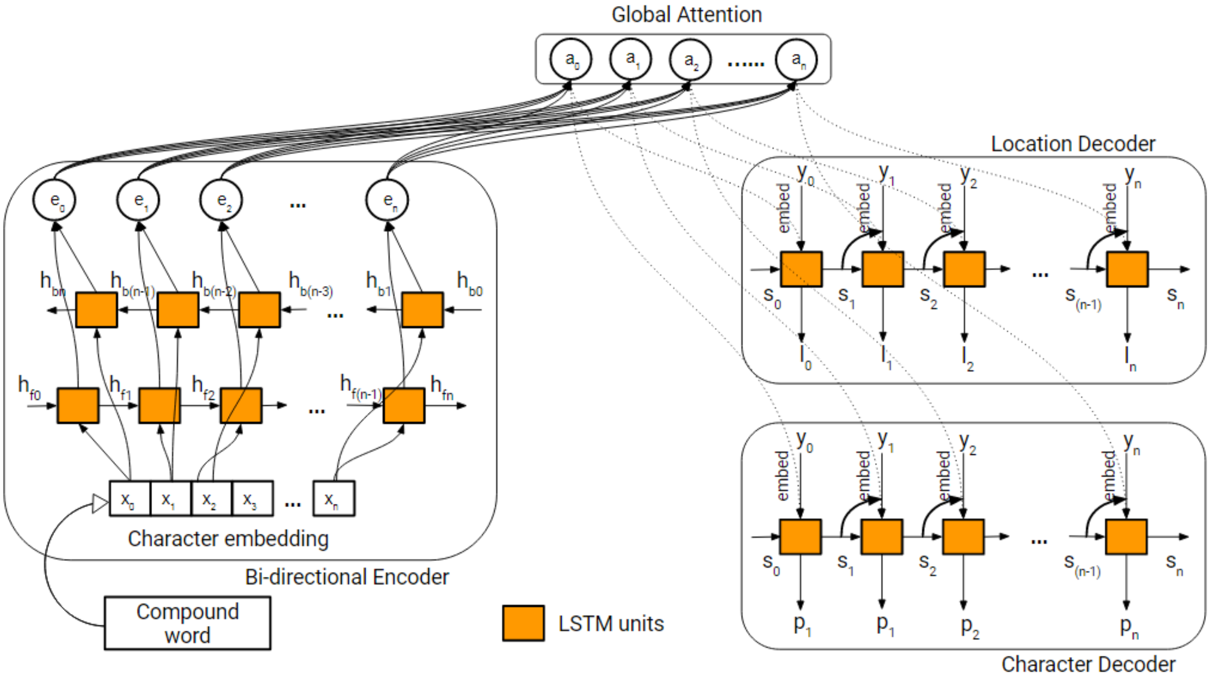

Figure 2: The bi-directional encoder and decoders with attention

We evaluate the DD-RNN and compare it with other tools and architectures in Section 4.

2.3 Implementation details

The architecture of the DD-RNN is shown in Figure 2. We used a character embedding size of 128. The bi-directional encoder and the two decoders are 2 layers deep with 512 LSTM units in each layer. A dropout layer with $p = 0.3$ is applied after each LSTM layer. The entire network is implemented in Torch [2].

Of the 71,747 words in our benchmark dataset, we randomly sampled 80% of the data for training our models. The remaining 20% was used for testing. We used stochastic gradient descent for optimizing the model parameters with an initial learning rate of 1.0. The learning rate was decayed by a factor of 0.5 if the validation perplexity did not improve after an epoch. We used a batch size of 64 and trained the network for 10 epochs on four Tesla K80 GPUs. This setup remains the same for all the experiments we conduct.

3 Existing Datasets and Tools

In this section, we briefly introduce various Sanskirt Sandhi datasets and splitting tools available in literature. We also discuss the tools' drawbacks and the major challenges faced while creating such tools.

Datasets: The *UoH corpus*, created at the University of Hyderabad[3] contains 113,913 words and their splits. This dataset is noisy with typing errors and incorrect splits. The recent *SandhiKosh corpus* (Shubham Bhardwaj, 2018) is a set of 13,930 annotated splits. We combine these datasets and heuristically prune them to finally get 71,747 words and their splits. The pruning is done by considering a data point to be valid only if the compound word and it's splits are present in a standard Sanskrit dictionary (Monier-Williams, 1970). We use this as our benchmark dataset and run all our experiments on it.

Tools: There exist multiple Sandhi splitters in the open domain such as (i) *JNU splitter* (Sachin, 2007), (ii) *UoH splitter* (Kumar et al., 2010) and (iii) *INRIA sanskrit reader companion* (Huet, 2003) (Goyal and Huet, 2013). Though each tool addresses the splitting problem in a specialized way, the general principle remains constant. For a given compound word, the set of all rules are applied to every character in the word and a large potential candidate list of word splits is obtained. Then, a morpheme dictionary of Sanskrit words is used with other heuristics to remove infeasible

---

[2] `http://torch.ch/`

[3] Available at: `http://sanskrit.uohyd.ac.in/Corpus/`

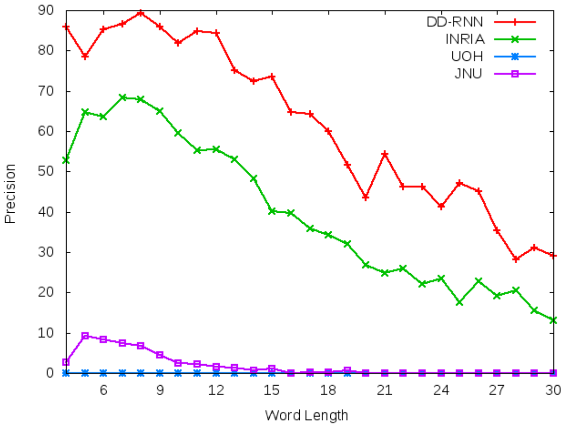

Figure 3: Top-1 split prediction accuracy comparison of different publicly available tools with DD-RNN

word split combinations. However, none of the approaches address the fundamental problem of identifying the location of the split before applying the rules, which will significantly reduce the number of rules that can be applied, hence resulting in more accurate splits.

4 Evaluation and Results

We evaluate the performance of our DD-RNN model by: (i) comparing the **split prediction** accuracy with other publicly available sandhi splitting tools, (ii) comparing the **split prediction** accuracy with other standard RNN architectures such as RNN, B-RNN, and B-RNN-A, and (iii) comparing the **location prediction** accuracy with the RNNs used for Chinese word segmentation (as they only predict the split locations and do not learn the rules of splitting)

4.1 Comparison with publicly available tools

The tools discussed in Section 3 take a compound word as input and provide a list of all possible splits as output (UoH and INRIA splitters provide weighted lists). Initially, we compared only the top prediction in each list with the true output. This gave a very low precision for the tools as shown in Figure 3. Therefore, we relaxed this constraint and considered an output to be correct if the true split is present in the top ten predictions of the list. This increased the precision of the tools as shown in Figure 4 and Table 1.

Even though DD-RNN generates only one output for every input, it clearly out-performs the other publicly available tools by a fair margin.

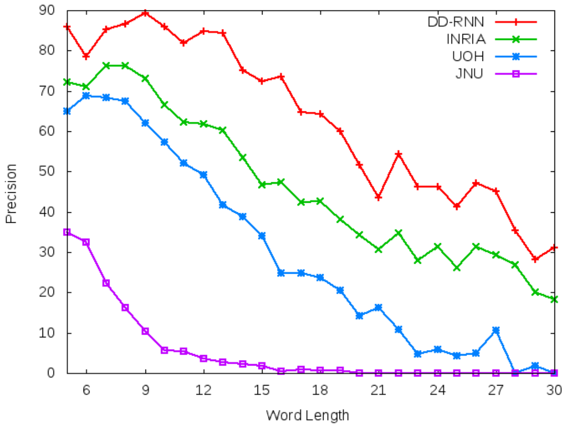

Figure 4: Split prediction accuracy comparison of different publicly available tools (Top-10) with DD-RNN (Top-1)

	Accuracy (%)	
Model	**Location Prediction**	**Split Prediction**
JNU (Top 10)	-	8.1
UoH (Top 10)	-	47.2
INRIA (Top 10)	-	59.9
RNN	79.10	56.6
B-RNN	84.62	58.6
B-RNN-A	88.53	69.3
DD-RNN	**95.0**	**79.5**
LSTM-4	70.2	-
GRNN-5	67.7	-

Table 1: Location and split prediction accuracy of all the tools and models under comparison

4.2 Comparison with standard RNN architectures

To compare the performance of DD-RNN with other standard RNN architectures, we trained the following three models to generate the split predictions on our benchmark dataset: (i) unidirectional encoder and decoder without attention (RNN), (ii) bi-directional encoder and decoder without attention (B-RNN), and (iii) bi-directional encoder and decoder with attention (B-RNN-A)

As seen from the middle part of Table 1, the DD-RNN performs much better than the other architectures with an accuracy of **79.5%**. It is to be noted that B-RNN-A is the same as DD-RNN without the *location decoder*. However, the accuracy of DD-RNN is 14.7% more than that the B-RNN-A and consistently outperforms B-RNN-A on almost all word lengths (Figure 5). This in-

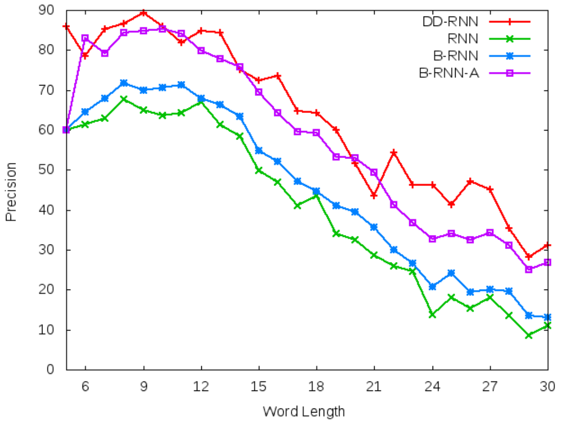

Figure 5: Split prediction accuracy comparison of different variations of RNN on words of different lengths

dicates that the attention mechanism of DD-RNN has learned to better identify the split location(s) due to its pre-training with the *location decoder*.

4.3 Comparison with similar works

(Reddy et al., 2018) propsed a *seq2seq* model using RNN with attention to tackle the Sandhi problem. Their model is similar to B-RNN-A and is outperformed by our proposed DD-RNN by $\tilde{6}$.47%. We also compared our proposed DD-RNN with a uni-directional LSTM with a depth of 4 (Chen et al., 2015b) (**LSTM-4**) and a Gated Recursive Neural Network with a depth of 5 (Chen et al., 2015a) (**GRNN-5**). These models were used to get state of the art results for Chinese word segmentation and their source code is made available online.[4] Since these models can only predict the location(s) of the split(s) and cannot generate the split words themselves, we used the location prediction accuracy as the metric. We trained these models on our benchmark dataset and the results are shown in Table 1. DD-RNN's precision is 35.3% and 40.3% better than LSTM-4 and GRNN-5 respectively. Conversely, we trained the DD-RNN for the Chinese word segmentation task to test the generalizability of the model. Since there are no morphological changes during segmentation in Chinese, the character decoder is redundant and the model collapses to simple *seq2seq*. We used the PKU dataset which is also used in (Chen et al., 2015b) & (Chen et al., 2015a) and obtained an accuracy of 64.25% which is comparable to the results of other standard models.

To summarize, we have used our benchmark dataset to compare the DD-RNN model with existing publicly available Sandhi splitting tools, other RNN architectures and models used for Chinese word segmentation task. Among the existing tools, the INRIA splitter gives the highest split prediction accuracy of 59.9%. Among the standard RNN architectures, B-RNN-A performs the best with a split prediction accuracy of 69.3%. LSTM-4 performs the best among the Chinese word segmentation models with a location prediction accuracy of 70.2%. DD-RNN outperforms all the models both in location and split predictions with 95% and 79.5% accuracies, respectively.

5 Research Impact

This work can be foundational to other Sanskrit based NLP tasks. Let us consider translation as an example. In Sanskrit, arbitrary number of words can be joined together to form a compound word. Literary works, especially from the *Vedic* era often contain words which are a concatenation of three or more simpler words. Presence of such compound words will increase the vocabulary size exponentially and hinder the translation process. However, as a pre-processing step, if all the compound words are split before training a translation model, the number of unique words in the vocabulary reduces which will ease the learning process.

6 Conclusion

In this research, we propose a novel double decoder RNN architecture with attention for Sanskrit Sandhi splitting. Learning such a model would provide further insights into the fundamental linguistic word formation rules of the language. A deep bi-directional encoder is used to encode the character sequence of a Sanskrit word. Using this encoded context vector, a *location decoder* is first used to learn the location(s) of the split(s). Then the *character decoder* is used to generate the split words. We evaluate the performance of the proposed approach on the benchmark dataset in comparison with other publicly available tools, standard RNN architectures and with prior work which tackle similar problems in other languages. As future work, we intend to tackle the harder *Samasa* problem which requires semantic information of a word in addition to the characters' context.

---

[4] `https://github.com/FudanNLP`

References

Akshar Bharati, Amba P Kulkarni, and V Sheeba. 2006. Building a wide coverage sanskrit morphological analyser: A practical approach. In *The First National Symposium on Modelling and Shallow Parsing of Indian Languages, IIT-Bombay*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In *ACL (1)*, pages 1744–1753.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*, pages 1197–1206.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Brendan S. Gillon. 2009. *Tagging Classical Sanskrit Compounds*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Pawan Goyal and Gérard Huet. 2013. Completeness analysis of a sanskrit reader. In *International Symposium on Sanskrit Computational Linguistics*, pages 130–171.

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

Gérard Huet. 2003. Towards computational processing of sanskrit. In *International Conference on Natural Language Processing (ICON)*. Citeseer.

Amba Kulkarni and Devanand Shukl. 2009. Sanskrit morphological analyser: Some issues. *Indian Linguistics*, 70(1-4):169–177.

Anil Kumar, Vipul Mittal, and Amba Kulkarni. 2010. *Sanskrit Compound Processor*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Monier Monier-Williams. 1970. *Sanskrit-English Dictionary*.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Vikas Reddy, Amrith Krishna, Vishnu Dutt Sharma, Prateek Gupta, Vineeth M. R, and Pawan Goyal. 2018. Building a word segmenter for sanskrit overnight. *CoRR*, abs/1802.06185.

Kumar Sachin. 2007. Sandhi splitter and analyzer for sanskrit (with reference to ac sandhi). *M. Phil. degree at SCSS, JNU (submitted, 2007)*.

Hasim Sak, Andrew W Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, pages 338–342.

Rahul Garg Sumeet Agarwal Shubham Bhardwaj, Neelamadhav Gantayat. 2018. Sandhikosh: A benchmark corpus for evaluating sanskrit sandhi tools. In *11th International Conference on Language Resources and Evaluation*.

Unsupervised Neural Word Segmentation for Chinese via Segmental Language Modeling

Zhiqing Sun
Peking University
`1500012783@pku.edu.cn`

Zhi-Hong Deng
Peking University
`zhdeng@pku.edu.cn`

Abstract

Previous traditional approaches to unsupervised Chinese word segmentation (CWS) can be roughly classified into discriminative and generative models. The former uses the carefully designed goodness measures for candidate segmentation, while the latter focuses on finding the optimal segmentation of the highest generative probability. However, while there exists a trivial way to extend the discriminative models into neural version by using neural language models, those of generative ones are non-trivial. In this paper, we propose the segmental language models (SLMs) for CWS. Our approach explicitly focuses on the segmental nature of Chinese, as well as preserves several properties of language models. In SLMs, a context encoder encodes the previous context and a segment decoder generates each segment incrementally. As far as we know, we are the first to propose a neural model for unsupervised CWS and achieve competitive performance to the state-of-the-art statistical models on four different datasets from SIGHAN 2005 bakeoff.

1 Introduction

Unlike English and many other languages, Chinese sentences have no explicit word boundaries. Therefore, Chinese Word Segmentation (CWS) is a crucial step for many Chinese Natural Language Processing (NLP) tasks such as syntactic parsing, information retrieval and word representation learning (Grave et al., 2018).

Recently, neural approaches for supervised CWS are attracting huge interest. A great quantities of neural models, e.g., tensor neural network (Pei et al., 2014), recursive neural network (Chen et al., 2015a), long-short-term-memory (RNN-LSTM) (Chen et al., 2015b) and convolutional neural network (CNN) (Wang and Xu, 2017), have been proposed and given competitive results to the best statistical models (Sun, 2010). However, the neural approaches for unsupervised CWS have not been investigated.

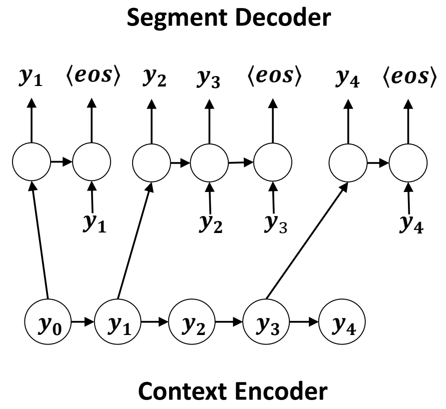

Figure 1: A Segmental Language Model (SLM) works on $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2\mathbf{y}_3\mathbf{y}_4$ with the candidate segmentation $\mathbf{y}_1$, $\mathbf{y}_{2:3}$ and $\mathbf{y}_4$, where $\mathbf{y}_0$ is an additional start symbol which is kept same for all sentences.

Previous unsupervised approaches to CWS can be roughly classified into discriminative and generative models. The former uses carefully designed goodness measures for candidate segmentation, while the latter focuses on designing statistical models for Chinese and finds the optimal segmentation of the highest generative probability.

Popular goodness measures for discriminative models include Mutual Information (MI) (Chang and Lin, 2003), normalized Variation of Branching Entropy (nVBE) (Magistry and Sagot, 2012) and Minimum Description Length (MDL) (Magistry and Sagot, 2013). There is a trivial way to extend these statistical discriminative approaches, because we can simply replace the n-gram language models in these approaches by neural language models (Bengio et al., 2003). There may

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4915–4920
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

exists other more sophisticated neural discriminative approaches, but it is not the focus of this paper.

For generative approaches, typical statistical models includes Hidden Markov Model (HMM) (Chen et al., 2014), Hierarchical Dirichlet Process (HDP) (Goldwater et al., 2009) and Nested Pitman-Yor Process (NPY) (Mochihashi et al., 2009). However, none of them can be easily extended into a neural model. Therefore, neural generative models for word segmentation are remaining to be investigated.

In this paper, we proposed the Segmental Language Models (SLMs), a neural generative model that explicitly focuses on the segmental nature of Chinese: SLMs can directly generate segmented sentences and give the corresponding generative probability. We evaluate our methods on four different benchmark datasets from SIGHAN 2005 bakeoff (Emerson, 2005), namely PKU, MSR, AS and CityU. To our knowledge, we are the first to propose a neural model for unsupervised Chinese word segmentation and achieve competitive performance to the state-of-the-art statistical models on four different datasets.[1]

2 Segmental Language Models

In this section, we present our segmental language models (SLMs). Notice that in Chinese NLP, characters are the atom elements. Thus in the context of CWS, we use “character” instead of “word” for language modeling.

2.1 Language Models

The goal of language modeling is to learn the joint probability function of sequences of characters in a language. However, This is intrinsically difficult because of the curse of dimensionality. Traditional approaches obtain generalization based on n-grams, while neural approaches introduce a distributed representation for characters to fight the curse of dimensionality.

A neural Language Model (LM) can give the conditional probability of the next character given the previous ones, and is usually implemented by a Recurrent Neural Network (RNN):

$$\mathbf{h}_t = f(\mathbf{y}_{t-1}, \mathbf{h}_{t-1}) \tag{1}$$

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = g(\mathbf{h}_t, \mathbf{y}_t) \tag{2}$$

where $\mathbf{y}_t$ is the distributed representation for the t^{th} character and $\mathbf{h}_t$ represents the information of the previous characters.

2.2 Segmental Language Models

Similar to neural language modeling, the goal of segmental language modeling is to learn the joint probability function of the segmented sequences of characters. Thus, for each segment, we have:

$$\hat{p}(\mathbf{y}_t^{(i)}|\mathbf{y}_{1:t-1}^{(i)}, \mathbf{y}^{(1:i-1)}) = g(\mathbf{h}_t^{(i)}, \mathbf{y}_t^{(i)}) \tag{3}$$

where $\mathbf{y}_t^{(i)}$ is the distributed representation for the t^{th} character in the i^{th} segment and $\mathbf{y}^{(1:i-1)}$ is the previous segments. And the concatenation of all segments $\mathbf{y}_{1:T_i}^{(i)}$ is exactly the whole sentence $\mathbf{y}_{1:T}$, where T_i is the length of the i^{th} segment $\mathbf{y}^{(i)}$, T is the length of the sentence $\mathbf{y}$.

Moreover, we introduce a context encoder RNN to process the character sequence $\mathbf{y}^{(1:i-1)}$ in order to make $\mathbf{y}_t^{(i)}$ conditional on $\mathbf{y}^{(1:i-1)}$. Specifically, we initialize $\mathbf{h}_0^{(i)}$ with the context encoder's output of $\mathbf{y}^{(1:i-1)}$.

Notice that although we have an encoder and the segment decoder g, SLM is not an encoder-decoder model. Because the content that the decoder generates is not the same as what the encoder provides.

Figure 1 illustrates how SLMs work with a candidate segmentation.

2.3 Properties of SLMs

However, in unsupervised scheme, the given sentences are not segmented. Therefore, the probability for SLMs to generate a given sentence is the joint probability of all possible segmentation:

$$\begin{aligned} p(\mathbf{y}_{1:T}) &= \sum_{T_1,T_2,\dots} \prod_i \hat{p}(\mathbf{y}_{1:T_i}^{(i)}) \\ &= \sum_{T_1,T_2,\dots} \prod_i \prod_{t=1}^{T_i+1} \hat{p}(\mathbf{y}_t^{(i)}|\mathbf{y}_{0:t-1}^{(i)}) \end{aligned} \tag{4}$$

where $\mathbf{y}_{T_i+1}^{(i)} = \langle eos \rangle$ is the **end of segment** symbol at the end of each segment, and $\mathbf{y}_0^{(i)}$ is the context representation of $\mathbf{y}^{(1:i-1)}$.

Moreover, for sentence generation, SLMs are able to generate arbitrary sentences by generating

---

[1] Our implementation can be found at `https://github.com/Edward-Sun/SLM`

segments one by one and stopping when generating **end of sentence** symbol $\langle EOS \rangle$. In addition, the time complexity is linear to the length of the generated sentence, as we can keep the hidden state of the context encoder RNN and update it when generating new words.

Last but not least, it is easy to verify that SLMs preserve the probabilistic property of language models:

$$\sum_i P(\mathbf{s}_i) = 1 \tag{5}$$

where $\mathbf{s}_i$ enumerates all possible sentences.

In summary, the segmental language models can perfectly substitute vanilla language models.

2.4 Training and Decoding

Similar to language model, the training is achieved by maximizing the training corpus log-likelihood:

$$L = -\log p(\mathbf{y}_{1:T}) \tag{6}$$

Luckily, we can compute the loss objective function in linear time complexity using dynamic programming, given the initial condition that $p(\mathbf{y}_{1:0}) = 1$:

$$p(\mathbf{y}_{1:n}) = \sum_{k=1}^{K} p(\mathbf{y}_{1:n-k}) \hat{p}(\mathbf{y}_{n-k+1:n}) \tag{7}$$

where $p(\cdot)$ is the joint probability of all possible segmentation, $\hat{p}(\cdot)$ is the probability of one segment and K is the maximal length of the segments.

We can also find the segmentation with maximal probability (namely, decoding) in linear time using dynamic programming in the similarly way with $\bar{p}(\mathbf{y}_{1:0}) = 1$:

$$\bar{p}(\mathbf{y}_{1:n}) = \max_{k=1}^{K} \bar{p}(\mathbf{y}_{1:n-k}) \hat{p}(\mathbf{y}_{n-k+1:n}) \tag{8}$$

$$\delta(\mathbf{y}_{1:n}) = \arg\max_{k=1}^{K} \bar{p}(\mathbf{y}_{1:n-k}) \hat{p}(\mathbf{y}_{n-k+1:n}) \tag{9}$$

where $\bar{p}$ is the probability of the best segmentation and δ is used to trace back the decoding.

3 Experiments

3.1 Experimental Settings and Detail

We evaluate our models on SIGHAN 2005 bakeoff (Emerson, 2005) datasets and replace all the punctuation marks with $\langle punc \rangle$, English characters with $\langle eng \rangle$ and Arabic numbers with $\langle num \rangle$ for all text and only consider segment the text between punctuations. Following Chen et al. (2014) , we use both training data and test data for training and only test data are used for evaluation. In order to make a fair comparison with the previous works, we do not consider using other larger raw corpus.

F1 score	**PKU**	**MSR**	**AS**	**CityU**
HDP	68.7	69.9	-	-
HDP + HMM	75.3	76.3	-	-
ESA	77.8	80.1	78.5	76.0
NPY-3	-	80.7	-	81.7
NPY-2	-	80.2	-	**82.4**
nVBE	80.0	**81.3**	76.6	76.7
Joint	81.1	81.7	-	-
SLM-2	**80.2**	78.5	79.4	78.2
SLM-3	79.8	79.4	**80.3**	80.5
SLM-4	79.2	79.0	79.8	79.7

Table 1: Main results on SIGHAN 2005 bakeoff datasets with previous state-of-the-art models (Chen et al., 2014; Wang et al., 2011; Mochihashi et al., 2009; Magistry and Sagot, 2012)

We apply word2vec (Mikolov et al., 2013) on Chinese Gigaword corpus (LDC2011T13) to get pretrained embedding of characters.

A 2-layer LSTM (Hochreiter and Schmidhuber, 1997) is used as the segment decoder and a 1-layer LSTM is used as the context encoder.

We use stochastic gradient decent with a minibatch size of 256 and a learning rate of 16.0 to optimize the model parameters in the first 400 steps, then we use Adam (Kingma and Ba, 2014) with a learning rate of 0.005 to further optimize the models. Model parameters are initialized by normal distributions as Glorot and Bengio (2010) suggested. We use a gradient clip $= 0.1$ and apply a dropout with dropout rate $= 0.1$ to the character embedding and RNNs to prevent over-fit.

The standard word precision, recall and F1 measures (Emerson, 2005) are used to evaluate segmentation performance.

3.2 Results and Analysis

Our final results are shown in Table 1, which lists the results of several previous state-of-the-art methods[2], where we mark the best results in

---

[2] Magistry and Sagot (2012) evaluated their nVBE on the training data, and the joint model of Chen et al. (2014) combine HDP+HMM and is initialized with nVBE, so in principle these results can not be compared directly.

F1 score	PKU	MSR	AS	CityU
SLM-4	79.2	79.0	79.8	79.7
SLM-4*	81.9	83.0	81.0	81.4
SLM-4†	**87.5**	84.3	**84.2**	**86.0**
SLM-4†*	87.3	**84.8**	83.9	85.8

Table 2: Results of SLM-4 incorporating ad hoc guidelines, where † represents using additional 1024 segmented setences for training data and * represents using a rule-based post-processing

boldface. We test the proposed SLMs with different maximal segment length $K = 2, 3, 4$ and use "SLM-K" to denote the corresponding model. We do not try $K > 4$ because there are rare words that consist more than 4 characters.

As can be seen, it is hard to predict what choice of K will give the best performance. This is because the exact definition of what a word remains hard to reach and different datasets follow different guidelines. Zhao and Kit (2008) use cross-training of a supervised segmentation system in order to have an estimation of the consistency between different segmentation guidelines and the average consistency is found to be as low as 85 (f-score). Therefore, this can be regarded as a top line for unsupervised CWS.

Table 1 shows that SLMs outperform previous best discriminative and generative models on PKU and AS datasets. This might be due to that the segmentation guideline of our models are closer to these two datasets.

Moreover, in the experiments, we observe that Chinese particles often attach other words, for example, "的" following adjectives and "了" following verbs. It is hard for our generative models to split them apart. Therefore, we propose a rule-based post-processing module to deal with this problem, where we explicitly split the attached particles from other words.[3] The post-processing is applied on the results of "SLM-4". In addition, we also evaluate "SLM-4" using the first 1024 sentences of the segmented training datasets (about 5.4% of PKU, 1.2% of MSR, 0.1% of AS and 1.9% of CityU) for training, in order to teach "SLM-4" the corresponding ad hoc segmentation guidelines. Table 2 shows the results.

We can find from the table that only 1024 guideline sentences can improve the performance of "SLM-4" significantly. While rule-based post-processing is very effective, "SLM-4†" can outperform "SLM-4*" on all the four datasets. Moreover, performance drops when applying the rule-based post-processing to "SLM-4†" on three datasets. These indicate that SLMs can learn the empirical rules for word segmentation given only a small amount of training data. And these guideline data can improve the performance of SLMs naturally, superior to using explicit rules.

Error	SLM-2	SLM-3	SLM-4
Insertion	7866	4803	3519
Deletion	3855	7518	8851

Table 3: Statistics of insertion errors and deletion errors that SLM-K produces on PKU dataset

3.3 The Effect of the Maximal Segment Length

The maximal segment length K represents the prior knowledge we have for Chinese word segmentation. For example $K = 3$ represents that there are only unigrams, bigrams and trigrams in the text. While there do exist words that contain more than four characters, most of the Chinese words are unigram or bigram. Therefore, K denotes a trade-off between the accuracy of short words and long words.

Specifically, we investigate two major segmentation problems that might affect the accuracy of word segmentation performance, namely, insertion errors and deletion errors. An insertion error insert a segment in a word, which split a correct word. And an deletion error delete the segment between two words, which results in a composition error (Li and Yuan, 1998). Table 3 shows the statistics of different errors on PKU of our model with different K. We can observe that insertion error rate decrease with the increase of K, while the deletion error rate increase with the increase of K.

We also provide some examples in Table 4, which are taken from the results of our models. It clearly illustrates that different K could result in different errors. For example, there is an insertion error on "反过来" by SML-2, and a deletion error on "促进" and "了" by SLM-4.

4 Related Work

Generative Models for CWS Goldwater et al. (2009) are the first to proposed a generative model for unsupervised word segmentation. They

---

[3] The rules we use are listed in the appendix at `https://github.com/Edward-Sun/SLM`.

Model	Example
SLM-2	而 这些 制度 的 完善 反 过来 又 促进 了 检察 人员 执法 水平 的 进一 步 提高
SLM-3	而 这些 制度 的 完善 反过来 又 促进了 检察 人员 执法 水平 的 进一步 提高
SLM-4	而 这些 制度 的 完善 反过 来 又 促进了 检察 人员 执法 水平 的进一步 提高
Gold	而 这些 制度 的 完善 反过来 又 促进 了 检察 人员 执法 水平 的 进一步 提高

Table 4: Examples of segmentation with different maximal segment length K

built a nonparametric Bayesian bigram language model based on HDP (Teh et al., 2005). Mochihashi et al. (2009) proposed a Bayesian hierarchical language model using Pitman-Yor (PY) process, which can generate sentences hierarchically. Chen et al. (2014) proposed a Bayesian HMM model for unsupervised CWS inspired by the character-based scheme in supervised CWS task, where the hidden state of charaters are set to $\{\mathbf{S}\text{ingle}, \mathbf{B}\text{egin}, \mathbf{E}\text{nd}, \mathbf{M}\text{iddle}\}$ to represents their corresponding positions in the words. The segmental language model is not a neural extension of the above statistical models, as we model the segments directly.

Segmental Sequence Models Sequence modeling via segmentations has been well investigated by Wang et al. (2017), where they proposed the Sleep-AWake Network (SWAN) for speech recognition. SWAN is similar to SLM. However, SLMs do not have sleep-awake states. And SLMs predict the following segment given the previous context while SWAN tries to recover the information in the encoded state. Therefore, the key difference is that SLMs are unsupervised language models while SWANs are supervised seq2seq models. Thereafter, Huang et al. (2017) successfully apply SWAN in their phrase-based machine translation. Another related work in machine translation is the online segment to segment neural transduction (Yu et al., 2016), where the model is able to capture unbounded dependencies in both the input and output sequences. Kong (2017) also proposed a Segmental Recurrent Neural Network (SRNN) with CTC to solve segmental labeling problems.

5 Conclusion

In this paper, we proposed a neural generative model for fully unsupervised Chinese word segmentation (CWS). To the best of knowledge, this is the first neural model for CWS. Our segmental language model is an intuitive generalization of vanilla neural language models that directly modeling the segmental nature of Chinese. Experimental results show that our models achieve competitive performance to the previous state-of-the-art statistical models on four datasets from SIGHAN 2005. We also show the improvement of incorporating ad hoc guidelines into our segmental language models. Our future work may include the following two directions.

- In this work, we only consider the sequential segmental language modeling. In the future, we are interested in build a hierarchical neural language model like the Pitman-Yor process.
- Like vanilla language models, the segmental language models can also provide useful information for semi-supervised learning tasks. It would also be interesting to explore our models in the semi-supervised schemes.

Acknowledgements

This work is supported by the National Training Program of Innovation for Undergraduates (URTP2017PKU001). We would also like to thank the anonymous reviewers for their helpful comments.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Jason S Chang and Tracy Lin. 2003. Unsupervised word segmentation without dictionary. *ROCLING 2003 Poster Papers*, pages 355–359.

Miaohong Chen, Baobao Chang, and Wenzhe Pei. 2014. A joint model for unsupervised chinese word segmentation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 854–863, Doha, Qatar. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In *Proceedings of the*

53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1744–1753, Beijing, China. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206, Lisbon, Portugal. Association for Computational Linguistics.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 133, pages 123–133.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. 2009. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

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

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2017. Computer science ¿ computation and language towards neural phrase-based machine translation. *arxiv.org/abs/1706.05565*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lingpeng Kong. 2017. *Neural Representation Learning in Linguistic Structured Prediction*. Ph.D. thesis, Google Research.

Haizhou Li and Baosheng Yuan. 1998. Chinese word segmentation. In *Proceedings of the 12th Pacific Asia Conference on Language, Information and Computation*, pages 212–217.

Pierre Magistry and Benoît Sagot. 2012. Unsupervized word segmentation: the case for mandarin chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 383–387. Association for Computational Linguistics.

Pierre Magistry and Benoît Sagot. 2013. Can mdl improve unsupervised chinese word segmentation? In *Sixth International Joint Conference on Natural Language Processing: Sighan workshop*, page 2.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 100–108. Association for Computational Linguistics.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 293–303, Baltimore, Maryland. Association for Computational Linguistics.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *Coling 2010: Posters*, pages 1211–1219, Beijing, China. Coling 2010 Organizing Committee.

Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2005. Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems*, pages 1385–1392.

Chong Wang, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. 2017. Sequence modeling via segmentations. *arXiv preprint arXiv:1702.07463*.

Chunqi Wang and Bo Xu. 2017. Convolutional Neural Network with Word Embeddings for Chinese Word Segmentation. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*.

Hanshi Wang, Jian Zhu, Shiping Tang, and Xiaozhong Fan. 2011. A new unsupervised approach to word segmentation. *Computational Linguistics*, 37(3):421–454.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. *arXiv preprint arXiv:1609.08194*.

Hai Zhao and Chunyu Kit. 2008. An empirical comparison of goodness measures for unsupervised chinese word segmentation with a unified framework. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

LemmaTag: Jointly Tagging and Lemmatizing for Morphologically Rich Languages with BRNNs

Daniel Kondratyuk[†] and **Tomáš Gavenčiak**[‡] and **Milan Straka**[†] and **Jan Hajič**[†]
Charles University
Faculty of Mathematics and Physics
[†]Institute of Formal and Applied Linguistics, [‡]Department of Applied Mathematics
`dankondratyuk@gmail.com, gavento@kam.mff.cuni.cz, {straka,hajic}@ufal.mff.cuni.cz`

Abstract

We present LemmaTag, a featureless neural network architecture that jointly generates part-of-speech tags and lemmas for sentences by using bidirectional RNNs with character-level and word-level embeddings. We demonstrate that both tasks benefit from sharing the encoding part of the network, predicting tag subcategories, and using the tagger output as an input to the lemmatizer. We evaluate our model across several languages with complex morphology, which surpasses state-of-the-art accuracy in both part-of-speech tagging and lemmatization in Czech, German, and Arabic.

1 Introduction

Morphologically rich languages are often difficult to process in many NLP tasks (Tsarfaty et al., 2010). As opposed to analytical languages like English, morphologically rich languages encode diverse sets of grammatical information within each word using inflections, which convey characteristics such as case, gender, and tense. The addition of several inflectional variants across many words dramatically increases the vocabulary size, which results in data sparsity and out-of-vocabulary (OOV) issues.

Due to these issues, morphological part-of-speech (POS) tagging and lemmatization are heavily used in NLP tasks such as machine translation (Fraser et al., 2012) and sentiment analysis (Abdul-Mageed et al., 2014). In morphologically rich languages, the POS tags typically consist of multiple morpho-syntactic subcategories providing additional information (see Figure 1). Closely related to POS tagging is lemmatization, which involves transforming each word to its root or dictionary form. Both tasks require context-sensitive awareness to disambiguate words with the same form but different syntactic or semantic features and behavior. Furthermore, lemmatization of a word form can benefit substantially from the information present in morphological tags, as grammatical attributes often disambiguate word forms using context (Müller et al., 2015).

We address context-sensitive POS tagging and lemmatization using a neural network model that jointly performs both tasks on each input word in a given sentence.[1] We train the model in a supervised fashion, requiring training data containing word forms, lemmas, and POS tags. In addition, we incorporate the ideas from Inoue et al. (2017) to optionally allow the network to predict the subcategories of each tag to improve accuracy. Our model is related to the work of Müller et al. (2015), which use conditional random fields (CRF) to jointly tag and lemmatize words for morphologically rich languages. The idea of jointly predicting several dimensions of categories has been explored prior to this work, for example, joint morphological and syntactic analysis (Bohnet et al., 2013) or joint parsing and semantic role labeling (Gesmundo et al., 2009).

Our model consists of three parts: (1) The **shared encoder**, which creates an internal representation for every word based on its character se-

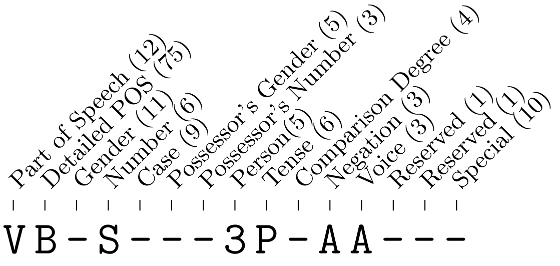

Figure 1: The tag components of the PDT Czech treebank with the numbers of valid values. Around 1500 different tags are in use in the PDT.

---

[1] The code for this project is available at `https://github.com/hyperparticle/LemmaTag`

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4921–4928
Brussels, Belgium, October 31 - November 4, 2018.

quence and the sentence context. We adopt the encoder architecture of Chakrabarty et al. (2017), utilizing character-level (Heigold et al., 2017) and word-level embeddings (Mikolov et al., 2013b; Santos and Zadrozny, 2014) processed through several layers of bidirectional recurrent neural networks (BRNN/BiRNN) (Schuster and Paliwal, 1997; Chakrabarty et al., 2017). (2) The **tagger decoder**, which applies a fully-connected layer to the outputs of the shared encoder to predict the POS tags. (3) The **lemmatizer decoder**, which applies an RNN sequence decoder to the combined outputs of the shared encoder and tagger decoder, producing a sequence of characters that predict each lemma (similar to Bergmanis and Goldwater (2018)).

The main advantages over other proposed models are: (i) The model is featureless, requiring little to no text preprocessing or morphological analysis postprocessing. (ii) The model shares the word embeddings, character embeddings, and RNN encoder weights in the tagger and lemmatizer, improving both tagging and lemmatization accuracy while reducing the number of parameters required for both tasks. (iii) The model predicts tag subcategories and provides the output of the tagger as features for the input of the lemmatizer, further improving accuracy.

We evaluate the accuracy of our model in POS tagging and lemmatization across several languages: Czech, Arabic, German, and English. For each language, we also compare the performance of a fully separate tagger and lemmatizer to the proposed joint model. Our results show that our joint model is able to improve the accuracy for both tasks, and achieves state-of-the-art performance in both POS tagging and lemmatization in Czech, German, and Arabic, while closely matching state-of-the-art performance for English.

2 The Joint LemmaTag Model

Given a sequence of words in a sentence $w_1, \ldots, w_k$, the task of the model is to produce a sequence of associated tags $t_1, \ldots, t_k$ and lemmas $\ell_1, \ldots, \ell_k$. For a word w_i at position i, we denote $c_{i,1}, c_{i,2} \ldots c_{i,m_i}$ to be the sequence of characters that make up w_i, where m_i indicates the length of the word string at position i. Analogously, we define $l_{i,1}, \ldots l_{i,\lambda_i}$ to be the sequence of characters that make up the lemma ℓ_i.

Our proposed model (shown in Figures 2 and 3) is split into three parts: the shared encoder, the tagger, and the lemmatizer. The initial layers of the model are shared between the tagger and lemmatizer, encoding the words, characters, and context in a given sentence. The encoder then passes its outputs to two networks, which perform a classification task to predict tags by the tagger and a sequence prediction task to output lemmas (character-by-character) in the lemmatizer.

2.1 Shared Encoder

In the encoder shown in Figure 2, each character $c_{i,1}, c_{i,2} \ldots c_{i,m_i}$ of a word w_i is indexed into an embedding layer to produce fixed-length embedded vectors representing each character. These vectors are further passed into a layer of BRNNs composed of gated recurrent units (GRU) (Cho et al., 2014) producing outputs $\mathbf{e}_1^c, \ldots, \mathbf{e}_m^c$, and whose final states are concatenated to produce the character-level embedding $\mathbf{s}_i^c$ of the word. Similarly, we index w_i into a word-level embedding layer to compute vector $\mathbf{e}_i^b$. Then we sum these results to produce the final word embedding $\mathbf{e}_i^w = \mathbf{s}_i^c + \mathbf{e}_i^b$.

We repeat this process independently for all the words in the sentence and feed the resulting sequence $\mathbf{e}_1^w \ldots \mathbf{e}_k^w$ into another two BRNN layers composed of long short-term memory units (LSTM) with residual connections. This produces word-level outputs $\mathbf{o}_1^w, \ldots \mathbf{o}_k^w$ that encode sentence-level context for each word (we ignore the final hidden states).

2.2 Tagger

The task of the tagger is to predict a tag $t_i \in \mathcal{T}$ given a word w_i and its context, where $\mathcal{T}$ is a set of possible tags. As explained the introduction, morphologically rich languages typically subdivide tags further into several subcategories $t_i = (t_{i,1}, \ldots, t_{i,\tau})$, where $t_{i,j} \in \mathcal{T}_j$, the j-th subcategory. See Figure 1 for an illustration taken from the Czech PDT tagset where $\tau = 15$.

Having the encoded words of a sentence available, the tagger consists of a fully-connected layer with $|\mathcal{T}|$ neurons whose input is the output of the word feature RNN $\mathbf{o}_i^w$ (see figure 2). This layer produces the logits $\mathbf{t}_i$ of the tag values and the predictions t_i as the maximum-likelihood value (i.e., softmax).

To obtain the information about categorical nature of each tag, we also predict every category $t_{i,j}$ of the tag independently (if they exist in the

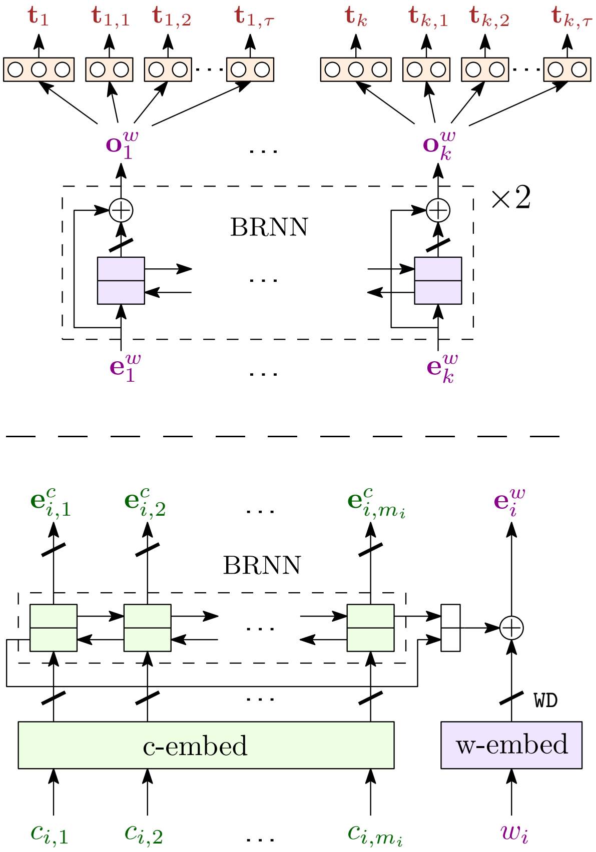

Figure 2: *Bottom:* Word-level encoder. The characters of every input word are embedded with a look-up table (c-embed) and encoded with a BRNN. The outputs $\mathbf{e}_{i,j}^c$ are used in decoder attention, and the final states are summed with the word-level embedding (w-embed) to produce $\mathbf{e}_i^w$. `WD` denotes word dropout.
Top: Sentence-level encoder and tag classifier. Two BRNN layers with residual connections act on the embedded words $\mathbf{e}_i^w$ of a sentence, providing context. The output of the tag classification are the logits for both the whole tags $\mathbf{t}_i$ and their components $\mathbf{t}_{i,j}$.
Both: Thick slanted lines denote training dropout.

dataset) with τ dense layers similar to Inoue et al. (2017). The j-th layer has $|\mathcal{T}_j|$ neurons and outputs the logits $\mathbf{t}_{i,j}$ for the category values. While these values are trained for, their value is not used in tag prediction. All tag values $\mathbf{T}_i = (\mathbf{t}_i, \mathbf{t}_{i,1} \dots, \mathbf{t}_{i,\tau})$ are concatenated into a flat vector and fed into the lemmatizer as an additional set of potentially useful features.

2.3 Lemmatizer

The task of the lemmatizer is to produce a sequence of characters $l_{i,1}, \dots, l_{i,\lambda_i}$ and the lemma length λ_i for each lemma ℓ_i. We use a recurrent sequence decoder, a setup typical of many sequence-to-sequence (seq2seq) tasks such as in neural machine translation (Sutskever et al., 2014).

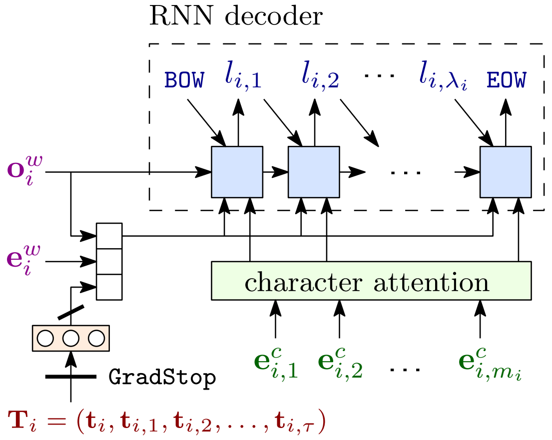

Figure 3: Lemma decoder, consisting of a standard seq2seq autoregressive decoder with Luong attention on character encodings, and with additional inputs of processed tagger features $\mathbf{T}_i$, embeddings $\mathbf{e}_i^w$ and sentence-level outputs $\mathbf{o}_i^w$. Gradients from the lemmatizer are stopped from flowing into the tagger (denoted `GradStop`).

The lemmatizer consists of a recurrent LSTM layer whose initial state is taken from word-level output $\mathbf{o}_i^w$ and whose inputs consist of three parts. The first part is the embedding of the previous output character (initially a *beginning-of-word* character `BOW`).

The second part is a character-level attention mechanism (Bahdanau et al., 2014) on the outputs of the character-level BRNN $\mathbf{e}_{i,1}^c, \dots, \mathbf{e}_{i,m_i}^c$. We employ the multiplicative attention mechanism described in Luong et al. (2015), which allows the LSTM cell to compute an attention vector that selectively weights character-level information in $\mathbf{e}_{i,j}^c$ at each time step j based on the input state of the LSTM cell.

The third and final part of the RNN input allows the network to receive the information about the embedding of the word, the surrounding context of the sentence, and the output of the tagger. This output is the same for all time steps of a lemma and is a concatenation of the following: the output of the encoder $\mathbf{o}_i^w$, the embedded word $\mathbf{e}_i^w$ and processed tag features $\mathbf{T}_i^f$. The tag features are obtained by projecting the concatenated outputs of the tagger $\mathbf{T}_i$ through a fully connected layer with ReLU activation. During training, we do not pass the gradients back through $\mathbf{T}_i$ to prevent the distortion of the tagger output.

The decoder performs greedy decoding to predict the character outputs. It runs until it produces the *end-of-word* character `EOW` or reaches a character limit of $m_i + 10$.

2.4 Loss Function

We define the final loss function as the weighted sum of the losses of the tagger and the lemmatizer:

$$L(\hat{y}, y) = \boldsymbol{\alpha} L(\hat{\mathbf{y}}^t, \mathbf{y}^t) + \beta L(\hat{\mathbf{y}}^\ell, \mathbf{y}^\ell)$$

where $\mathbf{y}$ are the predicted outputs, $\hat{\mathbf{y}}$ the expected outputs, $\mathbf{y}^t$, the tag components and $\mathbf{y}^\ell$ are the lemma characters. The tagger and lemmatizer losses are separately computed as the softmax cross entropy of the output logits. The weight hyperparameters $\boldsymbol{\alpha}, \beta$ scale the training losses so that the subtag and lemmatizer losses do not overpower the unfactored tag predictor gradients. The vector $\boldsymbol{\alpha}$ contains $\tau + 1$ weights: one for the whole tag and one for every component.[2]

3 Experiments

In this section, we show the outcomes of evaluation when running our joint tagger and lemmatizer and compare with the current state of the art in Czech, German, Arabic, and English datasets. Additionally, we evaluate the lemmatizer and tagger separately to compare the relative increase in tagging and lemmatization accuracy.

3.1 Datasets

Our datasets consist of the Czech Prague Dependency Treebank (PDT) (Hajič et al., 2006, 2018), the German TIGER corpus (Brants et al., 2004), the Universal Dependencies Prague Arabic Dependency Treebank (UD-PADT) (Hajic et al., 2004), the Universal Dependencies English Web Treebank (UD-EWT) (Silveira et al., 2014), and the WSJ portion of the English Penn Treebank (tags only) (Marcus et al., 1993). In all datasets, we use the tags specific to their respective language. Of these datasets, only Czech and Arabic provide subcategorical tags, and we use unfactored tags for the rest. See Table 1 for tagger and lemmatizer accuracies.

Note that the PDT dataset disambiguates lemmas with the same textual representation by appending a number as lemma sense indicator. For example, the dataset contains disambiguated lemmas `moc-1` (as *power*) and `moc-2` (as *too much*). About 17.5% of the PDT tokens have such sense-disambiguated lemmas. LemmaTag predicts the lemmas including the senses and the accuracies in Table 1 take that into account. Ignoring the sense ambiguity, the lemmatization accuracy of the joint LemmaTag model is 98.94% for Czech-PDT.

3.2 Hyperparameters

We use loss weights $\boldsymbol{\alpha}_0 = 1.0$ for the whole tags, $\boldsymbol{\alpha}_{1,\ldots,\tau} = 0.1$ for the tag component losses and $\beta = 0.5$ for the lemmatizer loss.[3] The RNNs and word embedding tables have dimensionality 768 except for character-level embeddings and the character-level RNN, which are of dimension 384. The fully-connected layer whose inputs are $\mathbf{T}_i$ is of dimension 256.

We train the models for 40 epochs with random permutations of training sentences and batches of 16 sentences. The starting learning rate is $\eta = 0.001$ and we scale this by 0.25 at epochs 20 and 30 to increase accuracy. We train the network using the lazy variant of the Adam optimizer (Kingma and Ba, 2014), which only updates accumulators for variables that appear in the current batch (TensorFlow, 2018), with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. We clip the global gradient norm to 3.0 to reduce the risk of exploding gradients.

To prevent the tagger from overfitting, we devise several strategies for regularization. We apply dropouts with rate 0.5 as indicated in Figures 2 and 3. The word dropout (`WD`) replaces 25% of words by the unknown token `<unk>` to force the network to rely more on context, combatting data sparsity issues. Lastly, we employ label smoothing (Pereyra et al., 2017) which is a way to prevent the network from being too confident in any one class. The label smoothing parameter is set to 0.1 for the tagger logits (both whole tags and the tag components).

Note that we did not perform any complex hyperparameter search. For additional information on real-world performance and additional techniques which have not improved evaluation accuracy, see Appendix A.

4 Conclusion

The evaluation results show that performing lemmatization and tagging jointly by sharing encoder parameters and utilizing tag features is

---

[2] If no components are available, $\tau = 0$.

[3] These are reasonable values to prevent gradients from overpowering one another. The lemmatizer tends to influence the tagger heavily.

Approach	**Czech-PDT***		**German-TIGER**		**Arabic-PADT***		**Eng-EWT**		**Eng-WSJ**
	tag	lem	tag	lem	tag	lem	tag	lem	tag
LemmaTag (sep)	96.83	98.02	98.96	98.84	95.03	96.07	**95.50**	97.03	97.59
LemmaTag (joint)	**96.90**	**98.37**	**98.97**	**99.05**	**95.21**	**96.08**	95.37	**97.53**	N/A
SoTA results	$\mathit{95.89}^{a}_{+}$	$\mathit{97.86}^{b}_{+}$	$\mathit{98.04}^{c}_{+}$	$\mathit{98.24}^{c}_{+}$	$\mathit{91.68}^{d}_{+}$	$\mathit{92.60}^{e}$	$\mathit{93.90}^{e}$	$\mathit{96.90}^{e}$	$\boldsymbol{97.78}^{f}_{+}$

Table 1: Final accuracies on the test sets comparing the LemmaTag architecture as described (joint), LemmaTag neither sharing the encoder nor providing tagger features (sep), and the state-of-the-art results (SoTA). The state-of-the-art results are taken from the following papers: *(a)* Hajič et al. (2009), *(b)* Straková et al. (2014), *(c)* Eger et al. (2016), *(d)* Inoue et al. (2017), *(e)* Straka et al. (2016), *(f)* Ling et al. (2015). The results marked with a plus$_+$ use additional resources apart from the dataset, and datasets marked with a star* indicate the availability of subcategorical tags.

mutually beneficial in morphologically rich languages. We have shown that incorporating these ideas results in excellent performance, surpassing state-of-the-art in Czech, German, and Arabic POS tagging and lemmatization by a substantial margin, while closely matching state-of-the-art English POS tagging accuracy.

However, in languages with weak morphology such as English (and German to a lesser extent), sharing the encoder parameters may even hurt the performance of the tagger. We believe this is a consequence of tags correlating less with word-level morphology, and more with sentence-level syntax in morphologically poor languages. Lemma prediction could benefit from the syntactic information in the tags, but the tag predictions rely more on syntactic structure (i.e., word order) rather than on root forms of individual words which could be ambiguous.

There are some possible performance improvements and additional metrics which we leave for future work. For simplicity, one improvement we intentionally left out is the use of additional data. We can incorporate word2vec (Mikolov et al., 2013a) or ELMo (Peters et al., 2018) word representations, which have shown to reduce out-of-domain issues and provide semantic information (Eger et al., 2016). A second improvement is to integrate information from a morphological dictionary to resolve certain ambiguities (Hajič et al., 2009; Inoue et al., 2017). A third improvement can be to replace the seq2seq lemmatizer decoder with a classifier that chooses a corresponding edit tree to modify (reduce) the word form to its lemma (Chakrabarty et al., 2017). A fourth possible improvement would be to experiment with the Transformer model (Vaswani et al., 2017), which utilizes non-recurrent multi-headed self-attention and has been shown to achieve state-of-the-art performance in several related sequence tasks (Dehghani et al., 2018). Lastly, we would like to evaluate LemmaTag on a wider range of languages, e.g., on the Universal Dependencies (Nivre et al., 2016) languages and treebanks which employ lemmatization, and to analyze the use of different types of POS tags in the model.

The code we used for LemmaTag is available at `https://github.com/hyperparticle/LemmaTag`.

Acknowledgments

The work described herein has been supported by the City of Prague under the "OP PPR" program, project No. CZ.07.1.02/0.0/0.0/16_023/0000108 and it has been using language resources developed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

Tomáš Gavenčiak has been supported by Czech Science Foundation (GACR) project 17-10090Y "Network optimization". Daniel Kondratyuk has been supported by the Erasmus Mundus program in Language & Communication Technologies (LCT).

References

Muhammad Abdul-Mageed, Mona Diab, and Sandra Kübler. 2014. Samar: Subjectivity and sentiment analysis for arabic social media. *Computer Speech & Language*, 28(1):20–37.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with lematus. In

Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), volume 1, pages 1391–1400.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richrd Farkas, Filip Ginter, and Ja n Haji. 2013. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on language and computation*, 2(4):597–620.

Abhisek Chakrabarty, Onkar Arun Pandit, and Utpal Garain. 2017. Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1481–1491.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Steffen Eger, Rüdiger Gleim, and Alexander Mehler. 2016. Lemmatization and morphological tagging in german and latin: A comparison and a survey of the state-of-the-art. In *LREC*.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling inflection and word-formation in smt. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–674. Association for Computational Linguistics.

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 37–42, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jan Hajič, Eduard Bejček, Alevtina Bémová, Eva Buráňová, Eva Hajičová, Jiří Havelka, Petr Homola, Jiří Kárník, Václava Kettnerová, Natalia Klyueva, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Petr Pajas, Jarmila Panevová, Lucie Poláková, Magdaléna Rysová, Petr Sgall, Johanka Spoustová, Pavel Straňák, Pavlína Synková, Magda Ševčíková, Jan Štěpánek, Zdeňka Urešová, Barbora Vidová Hladká, Daniel Zeman, Šárka Zikánová, and Zdeněk Žabokrtský. 2018. Prague dependency treebank 3.5. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL, `http://hdl.handle.net/11234/1-2621`), Faculty of Mathematics and Physics, Charles University.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, Magda Ševčíková-Razímová, and Zdeňka Urešová. 2006. Prague Dependency Treebank 2.0 (PDT 2.0). LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Jan Hajič, Jan Raab, Miroslav Spousta, et al. 2009. Semi-supervised training for the averaged perceptron pos tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 763–771. Association for Computational Linguistics.

Jan Hajic, Otakar Smrz, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. 2004. Prague arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 505–513.

Go Inoue, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Joint prediction of morphosyntactic categories for fine-grained arabic part-of-speech tagging exploiting tag dictionary information. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 421–431.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Milan Straka, Jan Hajic, and Jana Strakov. 2016. Udpipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Jana Straková, Milan Straka, and Jan Hajič. 2014. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

TensorFlow. 2018. tf.contrib.opt.lazyadamoptimizer: Class lazyadamoptimizer. TensorFlow documentation from tensorflow.org.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (spmrl): what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

A Appendix

A.1 GPU Performance

We ran all the tests on an NVIDIA GTX 1080 Ti GPU. The joint LemmaTag training takes about 3 hours for Arabic PADT, 4.5 hours for English EWT, 12 hours for German TIGER, and 22 hours for Czech PDT. The separate models take about 50% more time. After training, the lemma and tag predictions of 219,000 test tokens of the Czech PDT take about 100 seconds.

A.2 Other Techniques

We briefly summarize some of the additional techniques we have tried but which do not improve the results. While some of those techniques do help on smaller models or earlier in the training, the effect on the fully trained network seems to be marginal or even detrimental.

Separate sense prediction. Instead of predicting the sense disambiguation with the lemmatizer (Czech only), we tried to predict sense as an additional classification problem with one dense layer based on $\mathbf{o}_i^w$ and $\mathbf{T}_i$, but it seems to perform slightly worse (0.2%).

Beam search decoder. We have implemented a beam search decoder for the lemmatizer instead of the standard greedy one, but the improvement was marginal (around 0.01%).

Variational dropout. While the dropouts in the LemmaTag are completely random, variational dropout erases the same channels across the time steps of the RNN. While this generally improves training in convolutional networks and RNNs, we saw no significant difference.

Layer normalization. Layer normalization applied to the encoding RNNs did not bring significant gain and also slowed down the training.

Recovering Missing Characters in Old Hawaiian Writing

Brendan Shillingford[*1,2] and **Oiwi Parker Jones**[*1]
University of Oxford[1], DeepMind[2]
`brendan.shillingford@cs.ox.ac.uk`,
`oiwi.parkerjones@wolfson.ox.ac.uk`

Abstract

In contrast to the older writing system of the 19th century, modern Hawaiian orthography employs characters for long vowels and glottal stops. These extra characters account for about one-third of the phonemes in Hawaiian, so including them makes a big difference to reading comprehension and pronunciation. However, transliterating between older and newer texts is a laborious task when performed manually. We introduce two related methods to help solve this transliteration problem automatically. One approach is implemented, end-to-end, using finite state transducers (FSTs). The other is a hybrid deep learning approach, which approximately composes an FST with a recurrent neural network language model.

1 Introduction

From 1834 to 1948, more than 125,000 newspaper pages were published in the Hawaiian language (Nogelmeier, 2010). Yet by 1981, many expected this once flourishing language to die (Benton, 1981). Hawaiian has since defied expectations and experienced the beginnings of a remarkable recovery (Warner, 2001; Wilson and Kamanā, 2001). However much of the literary inheritance that is contained in the newspapers has become difficult for modern Hawaiians to read, since the newspapers were written in an orthography that failed to represent about one-third of the language's phonemes. This orthography, which we will refer to as the *missionary orthography*, excluded Hawaiian phonemes that did not have equivalents in American English (see Schütz, 1994), including Hawaiian's long vowels /iː eː aː oː uː/ and glottal stop /ʔ/. By contrast, the *modern Hawaiian orthography*, an innovation of Pukui and Elbert's Hawaiian dictionary (Pukui and Elbert, 1957), presents a nearly perfect, one-to-one mapping between graphemes and phonemes (see Appendix A.1). The process of manual transliteration from missionary to modern Hawaiian orthography is extremely labor intensive. Yet the cultural benefits are so great that hundreds of pages of newspaper-serials have already been transliterated by hand, such as Nogelmeier's new edition of the epic tale of *Hiʻiakaikapoliopele*, the volcano goddess's sister (Hoʻoulumāhiehie, 2007). Critically important as such efforts are to the continued revitalization of this endangered language, they are still only an introduction to the material that could be translated for a modern Hawaiian audience.

In this paper, we propose to automate, or semi-automate, the transliteration of old Hawaiian texts into the modern orthography. Following a brief review of related work (Section 2), we begin by describing a dataset of modern Hawaiian (Section 3). In Section 4, we present two methods for recovering missing graphemes (and hence phonemes) from the missionary orthography. The first composes a series of weighted FSTs; the second approximately composes a FST with a recurrent neural network language model (RNNLM) using a beam search procedure. Both approaches require only modern Hawaiian texts for training, which are much more plentiful than parallel corpora. Section 5 reports the results of our transliteration experiments using a simulated parallel corpus, as well as two 19th century newspaper articles for which we also have modern Hawaiian transcriptions. Being based on FSTs, both approaches are modular and extensible. We observe useful and promising results for both of our methods, with the best results obtained by the hybrid FST-RNNLM. These results showcase the strength of combining established hand-engineering methods with deep learning in a smaller data regime, with practical applications for an endangered language.

---

[*] Authors contributed equally.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4929–4934
Brussels, Belgium, October 31 - November 4, 2018.

2 Related work

Many of the themes that we address relate to existing literature. For example, Hajič et al. (2000) and Scannell (2014) have written on machine translation (MT) for closely related languages and on multilingual text normalization. Though language-relatedness makes MT easier (Kolovratník et al., 2010), state-of-the-art techniques such as neural machine translation (NMT) have not performed well for languages with little data (Östling and Tiedemann, 2017). So while the Hawaiian transliteration problem could be cast as an instance of MT or of NMT, we chose to sidestep the scarcity of parallel data by not considering such approaches.

Hybrid approaches that combine expert knowledge for well-understood structures with deep learning for data-plentiful subproblems offer rich opportunities for data-efficient modelling. Prior work has combined FSTs with RNNs, although not using the approximate FST-to-RNN composition algorithm that we introduce here (in Appendix A.4). For example, Sproat and Jaitly (2016) used an FST to restrict the search space when decoding from an RNN and Rastogi et al. (2016) incorporated RNN information into an FST.

3 Data

3.1 Missionary & modern orthography

The primary difference between the missionary and modern Hawaiian orthographies is that the missionary orthography does not encode long vowels or the glottal stop (see Appendix A.1). For example, the following Hawaiian phrases were recorded by a 19th-century German traveller in the missionary orthography: *Ua oia au*, *E ue ae oe ia Ii*, *E ao ae oe ia ia* (Chamisso, 1837, p. 7). In the modern orthography these become: *Ua ʻō ʻia au* ‘I am speared’, *E uē aʻe ʻoe iā ʻĪʻī* ‘You must weep for ‘Īʻī (a person)’, and *E aʻo aʻe ʻoe iā ia* ‘You teach him’ (Elbert and Pukui, 1979, p. 3).

We can convert text in the modern Hawaiian orthography *backward* chronologically to an approximate missionary orthography by mapping each glottal stop ⟨ʻ⟩ to the empty string ε, and each long vowel, e.g. ⟨ā ē ī ō ū⟩, to its corresponding short vowel, ⟨a e i o u⟩. As a first approximation, we may treat mappings from the modern-to-missionary orthographies as unambiguously many-to-one; thus there is information loss. We will return to secondary differences between the orthographies in Section 6. To illustrate, the following four words in the modern orthography all map to the same missionary string *aa*: *aʻa* (root), *ʻaʻa* (brave), *ʻaʻā* (crumbly lava rock), and *ʻāʻā* (stutter).

The *forward* mapping from missionary-to-modern orthographies is one-to-many. Thus the missionary string *aa* could map to *aʻa*, *ʻaʻa*, *ʻaʻā*, or *ʻāʻā*. The *transliteration problem* we address here seeks to discover how we can use context to recover the information not present in the missionary orthography that modern Hawaiian orthography retains.

Source	Chars	Words
Ulukau(160 texts)	6,518,451	1,334,451
Hiʻiakaikapoliopele	1,272,935	259,947
Wikipedia	577,794	10,221
Total	8,369,180	1,604,619

Figure 1: Modern data sources and their sizes.

3.2 Data sources

We draw on three sources for modern Hawaiian text: the main text of *Hiʻiakaikapoliopele* (Hoʻoulumāhiehie, 2007), 160 short texts from *Ulukau: The Hawaiian Electronic Library*, and the full Hawaiian Wikipedia (see Figure 1).[1]

For evaluation, we simulate a missionary-era version of the modern texts using the backward mapping described above. In addition, we evaluated our models on a couple of 19th century newspaper samples for which we have parallel missionary-era and modern text. Both simulated and real parallel corpora will be described in Section 5.

4 Models

We can frame the task of transliterating from missionary-to-modern Hawaiian orthographies as a sequence transduction problem. Many deep learning approaches (e.g. Sutskever et al., 2014; Graves, 2012) are not easily applicable to this task since we do not have a sufficiently large dataset of parallel texts. Instead, we focus on approaches that mix hand-designed finite state transducers with trained language models, including deep learning approaches like RNNLMs (Mikolov et al., 2010).

---

[1] *Ulukau: The Hawaiian Electronic Library*: `http://ulukau.org/`, Hawaiian Wikipedia: `https://haw.wikipedia.org/`. Both accessed 19 May 2018.

4.1 Finite state transducers

Our initial approach represents the mapping from missionary to modern orthography using a composition of (weighted) FSTs. For a thorough review of FSTs, see Mohri (1997).

First, we construct a finite state acceptor, I, from the input text. Here we construct a trivial chain-shaped acceptor that accepts only the input text. Each symbol in the input text is represented by a state which emits this symbol on a single transition that moves to the next state. The transition emitting the final symbol in the string leads to the sole accepting state.

Second, we construct a missionary-to-modern orthography conversion FST which we call C, which models potential orthography changes that can occur when transliterating from the missionary to modern Hawaiian orthography. For example, two non-deterministic transitions introduce an optional long-vowel map (a : ā) and (a : a). Another transition inserts glottal stops: (ϵ : ‘). By capturing the orthographic changes we know to occur, the composition $I \circ C$ produces a large set of candidates to be narrowed using the language model.

Third, we use the modern Hawaiian text from Section 3.2 to construct and evaluate a number of character-level n-gram language models, of various combinations of order and Katz backoff and Kneser-Ney (KN) smoothing (Katz, 1987; Kneser and Ney, 1995); see Appendix A.5 for details. N-gram language models can be expressed as weighted FSTs. We denote the n-gram or weighted FST language model as G. Character-level models are used as we wanted to generalize to out-of-vocabulary words, which we expected to occur frequently in a small corpus like the one we have for Hawaiian.

Finally, we use this model to infer modern orthography given a piece of text in missionary orthography as input, then compose the FSTs to form the *search graph* FST: $S = I \circ C \circ G$. The minimum cost path through S gives the predicted modern orthography. Of these n-gram-based approached, we found the Kneser-Ney-based models to perform best; these approaches are called FST-C-NGRAM-KN and FST-C_{wb}-NGRAM-KN.

We circumvent the lack of a large, non-simulated parallel corpus by training the language model exclusively on text in the modern Hawaiian orthography. In turn, the orthographic transliteration FST C produces candidates which are disambiguated by the language model. The result is finally evaluated against the ground-truth modern text.

Although the orthographic transliteration model is an approximation, and thus not exhaustive, it embodies an explicit and interpretable representation that can be easily extended independently of the rest of the model. To illustrate how the approach can be extended, we constructed a variant C_{wb} (where wb stands for word boundary). C_{wb} optionally inserts a space after each vowel using an additional arc that maps (ϵ : *space*), as diagrammed in Appendix A.2. This variant is able to model some changes in Hawaiian's word-boundary conventions (Wilson, 1976), such as *alaila* becoming *a laila* which demarcates the preposition *a* ‘until’ and noun *laila* ‘then’. We employ this variant to predict modern equivalents from 19th century newspaper samples in Section 5. Pseudocode summarizing this method is shown in Appendix A.3. Example predictions can be found in Appendix A.6.

4.2 FSTs with LSTM language models

As an alternative approach, we combined the FST C in the previous section with an RNNLM (Mikolov et al., 2010). RNNLMs often generalize better than n-gram models.

An RNN is a neural network that models temporal or sequential data, by iterating a function mapping a state and input to a new state and output. These can be stacked to form a deep RNN. For language modelling, each step of the final RNN layer models a word or character sequence via $p(w_1, \ldots, w_n) = \prod_{i=1}^{n} p(w_i | w_{1:i-1})$ and can be trained by maximum likelihood. Recent language modeling work has typically used the long short-term memory (LSTM) unit (Hochreiter and Schmidhuber, 1997) for its favorable gradient propagation properties. All RNNs in this paper are LSTMs.

Our goal is to replace the n-gram language model in the end-to-end FST approach with an RNNLM. While the minimum cost path through an FST can be computed exactly as done in the previous section, it is not straightforward to compose the relation defined by an FST with an arbitrary one like that defined by an RNNLM. A minimum cost path through the composition of the FST and the RNNLM can be defined as a path (i.e. label sequence) that minimizes the sum of the FST path cost and the RNNLM cost.

We can approximately find a minimum cost path of the composition of the two models by a breadth-first search over the FST graph, or using a beam search, as follows. At any particular iteration, consider a single beam element. The beam element holds the current FST and RNN states, and the path taken through the FST so far. We follow each possible arc from the current FST state, each producing a new child beam element, and feed the output symbol into the RNN (unless it is ϵ). There may be duplicate beam elements due the nondeterminicity of the FST; in this case, the lower cost edge wins. We sort by the sum of the FST and RNN costs, keep the lowest-cost K, and then proceed to the next iteration. If a beam element is on an accepting state of the FST, it is kept as-is between iterations. Detailed pseudocode is provided in Appendix A.4.

In the following we will refer to the hybrid models as FST-RNNLM—or as FST-RNNLM-C and FST-RNNLM-C_{wb} if we want to distinguish between which FST we used. Similarly, the FST-only models will be referred to as FST-C and FST-C_{wb}, with suffixes denoting what kind of n-gram and smoothing were used. For example, FST-C-7GRAM-KN denotes a FST-only model with an 7-gram language model and Kneser-Ney smoothing. Details of the language models trained can be found in Appendix A.5.

5 Results

Evaluation. Since we were unable to find a sufficiently large corpus of parallel texts in the missionary and modern Hawaiian orthographies, we instead used a corpus of modern Hawaiian texts (*ground-truth*) as summarized in Section 3.2 and Figure 1. Note that training the n-gram and RNN language models required only this modern corpus.

To evaluate the accuracy of our approaches, we derived a synthetic parallel corpus from these modern Hawaiian texts. We also used a small but real parallel corpus, based on two 19th century newspaper texts and their hand-edited modern equivalents.

Simulated parallel corpus. To produce a simulated parallel corpus (*input-missionary*), we systematically reduced the orthography in the modern texts using the backward mapping described in Section 3.1. We then applied the two approaches described in Section 4, with the aim of recovering the information lost.

We evaluated the predicted modern text (*predictions*) by computing

$$\text{CERR} = \frac{d(\text{prediction}, \text{ground-truth})}{d(\text{input-missionary}, \text{ground-truth})},$$

where d denotes character-level edit distance. This is a modification of character error rate, normalized by the distance of the input and target rather than by the length of the target. We note that CERR may be high even when the predictions are very accurate as $d(\text{input-missionary}, \text{ground-truth})$ is small when the text is similar in both orthographies.

Table 1 reports the results of the approaches we described in Section 4. Out of the Kneser-Ney n-gram models, we found that the FST-C-9GRAM-KN and the version modelling word boundaries (FST-C_{wb}-9GRAM-KN) to perform best on the synthetic parallel corpus and newspapers, respectively. C_{wb} was not applied to the synthetic parallel corpus as we did not model word splitting. The hybrid models (FST-RNNLM) outperformed all FST-only approaches.

Real parallel corpus (newspaper texts). Not content to evaluate the model on simulated missionary orthography, we also evaluated it on two newspaper texts, using selections originally published in 1867 and 1894 for which we had 19th century and manually-edited modern equivalents. The newspaper selections discuss *Kahahana*, one of the last kings of Oʻahu (Kamakau and Perreira, 2002), and *Uluhaimalama*, a garden party and secret political gathering, held after the deposition of Hawaiʻi's last queen (Pukui et al., 2006). Unlike the synthetic missionary corpus evaluation where we did not model word splitting, we found that replacing C with C_{wb} on the newspaper texts significantly improved the output, especially on the FST-RNNLM model. Thus, we found the word-splitting hybrid model (FST-RNNLM-C_{wb}) to be the best performing model overall (Table 1).

6 Conclusions and future work

With this paper we introduced a new transliteration problem to the field, that of mapping between old and new Hawaiian orthographies—where the modern Hawaiian orthography represents linguistic information that is missing from older missionary-era texts. One difficulty of this problem is that there is a limited amount of Hawaiian data, making data-hungry solutions like end-to-end deep learning

	LM perplexity		Transliteration performance (%CERR)		
Transliteration model	Valid.	Test	Corpus	Newspaper 1	Newspaper 2
FST-(C/C_{wb})-7GRAM-KN	3.07	3.13	27.3%	50.1% / 38.7%	52.0% / 47.5%
FST-(C/C_{wb})-9GRAM-KN	2.95	3.02	26.6%	50.7% / 39.3%	52.5% / 47.2%
FST-(C/C_{wb})-11GRAM-KN	2.94	3.02	27.8%	53.9% / 41.3%	54.1% / 48.7%
FST-RNNLM-(C/C_{wb})	**2.65**	**2.69**	**16.3%**	47.2% / **34.3%**	49.8% / **41.2%**

Table 1: Performance (%CERR). Slash-separated pairs denote FSTs incapable/capable of inserting word boundaries, respectively; see Section 4. The -KN suffix denotes Kneser-Ney smoothing. The data from Section 3.2 is used for evaluating the modern-orthography language model perplexity, and "Corpus" evaluates test-set transliteration performance from the synthetic missionary text back to the original modern text.

Input	Ua lawe ola ia o Keawehano imua o Kahekili, a ua hai aku o Kapohu...
Prediction	Ua lawe ola ʻia ʻo Keawehano i mua o Kahekili, a ua haʻi aku ʻo Kapohu...
Ground-truth	Ua lawe ola ʻia ʻo Keawehano i mua o Kahekili, a ua haʻi aku ʻo Kapohū...

Figure 2: An example of (missionary input, predicted modern text, ground-truth), from each newspaper. Note the correctly split word in the second example. Incorrect characters, which are quite rare, are shown as red and underlined. More sample predictions can be found in Appendix A.6.

unlikely to work. To solve the transliteration problem, we therefore proposed two models: the first was implemented end-to-end using weighted FSTs; the second was a hybrid deep learning approach that combined an FST and an RNNLM. Both models gave promising results, but the hybrid approach, which allowed us to use a more powerful recurrent neural network-based language model despite our dataset's small size, performed best. Factoring a problem like ours into one part that can be modelled exactly using expert domain knowledge and into another part that can be learned directly from data using deep learning is not novel; however it is a promising research direction for data-efficient modelling. To our knowledge, this paper is the first to describe a procedure to compose an FST with an RNN by approximately performing beam search over the FST.

While the role of the RNNLM part of the hybrid approach may be obvious, the FST component plays an important role too. For example, the hand-designed FST component can be replaced without needing to retrain the RNNLM. We tried to showcase this modularity by constructing two FSTs which we referred to as C and C_{wb}, where only the latter allowed the insertion of spaces. Future work could extend the FST to model orthographic changes suggested by an error analysis of the current model's predictions (see Appendix A.6). These errors motivate new mappings for consonant substitutions like (r : l) and (s : k) observed in loanword adaptations (e.g. *rose* ⇒ *loke*). The error analysis also motivates mappings to delete spaces ($\sqcup : \epsilon$) and to handle contractions, like *na'lii* ⇒ *nā aliʻi*. We could further incorporate linguistic knowledge of Hawaiian into the FST, which tells us, for example, that a consonant is typically followed by a vowel (Parker Jones, 2010). Additional improvements to the hybrid model might be obtained by increasing the amount of modern Hawaiian text used to train the RNNLM. One way to do this would be to accelerate the rate at which missionary-era Hawaiian texts are modernized. To this end, we hope that the present models will be used within the Hawaiian community to semi-automate, and thereby accelerate, the modernization of old Hawaiian texts.

Acknowledgments

We are grateful to M. Puakea Nogelmeier for providing an electronic copy of *Hiʻiakaikapoliopele* (Hoʻoulumāhiehie, 2007).

References

Richard A Benton. 1981. *The flight of the Amokura: Oceanic languages and formal education in the South Pacific*. New Zealand Council for Educational Research, Wellington.

Adelbert von Chamisso. 1837. *Über die Hawaiische Sprache, Vorgelegt der Königlichen Academie der Wissenschaften zu Berlin am 12, Januar, 1837*. Weidmann, Leipzig.

Samuel H. Elbert and Mary Kawena Pukui. 1979. *Hawaiian Grammar*. University of Hawaiʻi Press, Honolulu.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Jan Hajič, Jan Hric, and Vladislav Kuboň. 2000. Machine translation of very close languages. In *ANLC '00 Proceedings of the sixth conference on Applied natural language processing*, pages 7–12.

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

Hoʻoulumāhiehie. 2007. *Ka Moʻolelo o Hiʻiakaikapoliopele*. Awaiaulu Press, Honolulu. Edited by M. Puakea Nogelmeier.

Samuel Manaiakalani Kamakau and Hiapo Perreira. 2002. Ka moʻolelo o Kahahana, māhele 1. *Ka Hoʻoilina*, 1(1):102–121.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

David Kolovratník, Natalia Klyueva, and Ondřej Bojar. 2010. Statistical machine translation between related and unrelated languages. In *Proceedings of the Conference on Theory and Practice of Information Technologies (ITAT-09)*, pages 31–36.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech*, pages 1045–1048.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311.

M. Puakea Nogelmeier. 2010. *Mai Paʻa i ka Leo: Historical Voice in Hawaiian Primary Materials: Looking Forward and Listening Back*. Bishop Museum Press, Honolulu.

Robert Östling and Jörg Tiedemann. 2017. Neural machine translation for low-resource languages. *Computing Research Repository*, arXiv:1708.05729. Version 1.

Oiwi Parker Jones. 2010. *A computational phonology and morphology of Hawaiian*. Ph.D. thesis, University of Oxford.

Oiwi Parker Jones. 2018. Illustrations of the IPA: Hawaiian. *Journal of the International Phonetic Association*, 48:103–115.

Mary Kawena Pukui and Samuel H. Elbert. 1957. *Hawaiian-English Dictionary*. University of Hawaiʻi Press, Honolulu.

Mary Kawena Pukui, Holo Hoʻopai, Oiwi Parker Jones, and Keao NeSmith. 2006. No ka mahiʻai ʻana, māhele 6. *Ka Hoʻoilina*, 5(1):2–23.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighted finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633.

Kevin Scannell. 2014. Statistical models for text normalization and machine translation. In *Proceedings of the First Celtic Language Technology Workshop*, pages 33–40.

Albert J. Schütz. 1994. *The Voices of Eden: A history of Hawaiian language studies*. University of Hawaiʻi Press, Honolulu.

Richard Sproat and Navdeep Jaitly. 2016. RNN approaches to text normalization: A challenge. *Computing Research Repository*, arXiv:1611.00068. Version 2.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Sam L. Noʻeau Warner. 2001. The movement to revitalize Hawaiian language and culture. In Leanne Hinton and Kenneth Hale, editors, *The Green Book of Language Revitalization in Practice*, pages –144. Academic Press, San Diego, CA.

William H. Wilson. 1976. Standardized Hawaiian orthography. Manuscript, University of Hawaiʻi.

William H. Wilson and Kauanoe Kamanā. 2001. "Mai loko mai o ka ʻiʻini: Proceeding from a dream": The ʻAha Pūnana Leo connection in Hawaiian language revitalization. In Leanne Hinton and Kenneth Hale, editors, *The Green Book of Language Revitalization in Practice*, pages 147–176. Academic Press, San Diego, CA.

When data permutations are pathological: the case of neural natural language inference

Natalie Schluter and Daniel Varab
Department of Computer Science
IT University
Copenhagen, Denmark
`{natschluter, djam}@itu.dk`

Abstract

Consider two competitive machine learning models, one of which was considered state-of-the art, and the other a competitive baseline. Suppose that by just permuting the examples of the training set, say by reversing the original order, by shuffling, or by mini-batching, you could report substantially better/worst performance for the system of your choice, by multiple percentage points. In this paper, we illustrate this scenario for a trending NLP task: Natural Language Inference (NLI). We show that for the two central NLI corpora today, the learning process of neural systems is far too sensitive to permutations of the data. In doing so we reopen the question of how to judge a good neural architecture for NLI, given the available dataset and perhaps, further, the soundness of the NLI task itself in its current state.

1 Introduction

There is increased interest today in the detection of information quality: whether a statement is true or false, or equivalently, whether one statement (the premise) is entailed by, contradicts or has no relation to another statement (the hypothesis). This is the Natural Language Inference task. The timely development of the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) and more recently the Multi-Genre NLI (MULTI-NLI) corpus (Williams et al., 2017) has lead to a steady increase in contributions to research on NLI. Recent research, however, indicates these central datasets to be trivially annotated to a large degree (Gururangan et al., 2018). In this paper, we give evidence of an unrelated problem.

Deep neural network approaches provide the state-of-the-art today for the tasks. We show a pathological sensitivity of these systems to permutations of the training set. This calls into question the soundness of the task in its current state and corresponding development and benchmarking of NLI neural systems.

Given the approximate iterative optimisation methods required to induce models, a common practice for statistical learning is to first randomly shuffle the training set. This serves to offset unwanted bias due to accidental ordering of the examples (for example, with respect to time or class). Probably because of this, there is very little literature or understanding of the effect of order among the training examples–strict ordering of examples has simply been known to be undesirable. However, as neural network approaches increasingly dominate in performance across many NLP tasks, the notion of random shuffling has become overshadowed by that of computational efficiency. This seems to lead back to experiment parameters from Sutskever et al. (2014)'s work, which demonstrates that grouping examples of similar sentence length into batches results in a halved training time. But what is the effect on accuracy? Indeed, a curiosity of neural network models induced over the NLI datasets is the ease in acquiring rather instable results as a result of simple example order permutations of the training set.

In this paper, we present an investigation into these questions. In particular, we consider two simple but competitive neural network topologies in order to investigate the effect of training example order from these datasets on performance. One of these achieves close to state-of-art results over SNLI and state-of-the-art for MULTI-NLI (for simple systems, Cf. Section 4.1), while the other is a simpler variant of the first, characterised by fewer parameters. Out of the standard deep learning, standard machine learning and other plausi-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4935–4939
Brussels, Belgium, October 31 - November 4, 2018.

ble orderings of the dataset, we show that only the original ordering of the training examples leads to state-of-the-art model induction. We also show that the gap in performance between the two neural architectures described here generally drops over all other permutations.

2 NLI task and datasets

The NLI task input is two sentences, $\mathbf{a} = (a_1, \ldots, a_{l_a})$ and $\mathbf{b} = (b_1, \ldots, b_{l_b})$ of lengths l_a and l_b respectively. Each a_i (resp. b_j) for $i \in [l_a]$ (resp. $j \in [l_b]$) corresponds to the word embedding with dimension d for the ith (resp. jth) word. The task dataset consists of labeled pairs of sentences, $\{\mathbf{a^{(n)}}, \mathbf{b^{(n)}}, \mathbf{y^{(n)}}\}_{n=1}^{N}$, where $y^{(n)} \in \{\text{entailment}, \text{contradiction}, \text{neutral}\}$ is the class.

We use two central datasets for our study here: SNLI and MULTI-NLI (Bowman et al., 2015; Williams et al., 2017), containing over 570K and 433K sentence pairs respectively.

3 Our neural network architectures

For the research presented in this paper, we choose two related and relatively simple neural network architectures corresponding roughly to Parikh et al. (2016) and a simplified version of Chen et al. (2017) consisting of five components.[1] Note that we also downloaded Chen et al. (2017)'s code[2] and ran it over the datasets; since it did not perform with the same accuracy reported in the paper for our run (87.77% instead of the reported 88%), and since it took several hours longer than our implementation to run (+7 hours longer), we concentrated on our own implementation. We also attempted to run Gong et al. (2017)'s system[3]; for our runs, the system halted without completion after several hours. Moreover, there are reports on the difficulty in getting the architecture to achieve the accuracy score of 88% reported in the original paper, Mirakyan et al. (2018) reporting that their re-implementation could only achieve 86.38%. This is significantly worst-performing than our best system, which we present now.

(1) Pre-projection. To compensate approximately for not updating the original embeddings during learning, we first carry out a preliminary projection of the embeddings, to the same dimensions using a feed-forward network.

(2) Embedding projection. We further project embeddings via either a simple feed-forward (FF) with a ReLU activation function or a bidirectional LSTM (BiLSTM) layer. The result is then sent to the attention component. This corresponds precisely therefore to Parikh et al. (2016)'s computationally efficient approximation of the vector product before soft-alignment.

(3) Attention. The attention mechanism, first introduced by Bahdanau et al. (2015), is based on a matrix of all-pairs scores between the elements of two sequences $\mathbf{a_i}$ and $\mathbf{b_j}$:[4]

$$e_{ij} := F'(\mathbf{a_i}, \mathbf{b_j}) := F(\mathbf{a_i})^T F(\mathbf{b_j})$$

We follow (Parikh et al., 2016) and later attention-based models for NLI, by representing the importance of $\mathbf{a_i}$ with respect to $\mathbf{b}$ as the normalised sum

$$\beta_i := \sum_{j=1}^{l_b} \frac{exp(e_{ij})}{\sum_{k=1}^{l_b} exp(e_{ik})} b_j.$$

The result is concatenated with $\mathbf{a_i}$, to create $[\mathbf{a_i}, \beta_i]$ which is then projected down to original embedding dimension. The same is done for $\mathbf{b_j}$ with respect to $\mathbf{a}$.

(4) Aggregation. Following the attention mechanism we aggregate over words for a sentence representation. Either we sum over the attended word vectors Parikh et al. (2016) or use the final state of a single LSTM layer (Chen et al., 2017)[5].

(5) Prediction. Finally we feed a vector concatenation of both sentence vectors as input to a component consisting of three feed-forward layers with dropout and regularisation, followed by a linear softmax layer for prediction.

Instantiated architectures. The two topologies we adopt for this study consist of the above components with embedding projection and aggregation as follows:

- FF/SUM corresponding to the embedding projection instantiated with a feed-forward network and the aggregation carried out through vector summation, and

---

[1] We make the systems publicly available `https://github.com/natschluter/nli-data-permutations`.

[2] `https://github.com/lukecq1231/nli`

[3] `https://github.com/YichenGong/Densely-Interactive-Inference-Network`

[4] Note that we also experimented with $e_{ij} := F'(\mathbf{a_i}, \mathbf{b_j})$ where F' was a component-wise multiplication and F' a feed-forward network with ReLU activations. However, we observed no benefits to doing so performance-wise, and substantial loss in computational efficiency.

[5] For Chen et al. (2017), aggregation consists of several layers of BiLSTMS as opposed to our single LSTM layer.

- Bi/LSTM corresponding to the embedding projection instantiated with a BiLSTM and the aggregation carried out via an LSTM.

Other hyperparameters. We use 300 dimensional GloVe embeddings trained on the Common Crawl 840B tokens dataset (Pennington et al., 2014), which remain fixed during training. Out of vocabulary (OOV) words are represented as zero vectors.[6] We use a 0.2 dropout rate and L2 regularisation, applied in all feed-forward layers. We optimise the network using categorical cross-entropy loss and employ the RMSprop optimizer with ρ set to 0.9, a 0.001 learning rate, with a batch size of 512 and use early stopping over the development set after no improvement in accuracy after 4 epochs.

State-of-the-art for simple systems. The models are **simple** in that no information above word embeddings is taken as input, for example, no POS-tags or syntactic relations (See Section 4.1). Our Bi/LSTM system also currently sets the state-of-the-art for simple systems on the MULTI-NLI dataset (See Section 4.1).

Proj-Agg	split	SNLI accuracy	MULTI-NLI accuracy
FF/ SUM	test	85.72	71.58
	test-mis	-	70.63
	dev	85.64	72.29
	dev-mis	-	72.80
Bi/ LSTM	test	**87.12** (+1.4)	**75.58** (+4.0)
	test-mis	- -	**74.33** (+3.7)
	dev	**87.53** (+1.89)	**76.43** (+4.14)
	dev-mis	- -	**76.04** (+3.24)

Table 1: Performance in accuracy of networks, by projection (Proj-) and aggregation (Agg) components.

4 Related work

Previous work is related either in terms of the neural architectures for NLI (Section 4.1), or in terms of work on training data permutations in learning (Section 4.2).

4.1 State-of-the-art NLI

There are different types of neural network systems in the literature with respect to the simplicity of input data required for modeling and interdependence of the internal modules. In this work, we only consider simple system approaches that use only word embeddings (no character representations, POS-tags, word-position, syntactic tree, external resources, etc.) and consist only of interdependent modules (not ensembles). We make no claims regarding linguistic or ensemble-complex systems, but make the straightforward hypothesis that the conclusions presented here can be extended to cover more complex frameworks as well, especially given that our systems are strongly competitive or even better performing than two highly complex state-of-the-art neural systems (Cf. Section 3).

For simple systems, the state-of-the-art is currently set by Sha et al. (2016) at 87.5%, on SNLI. They use a standard BiLSTM to read the premise, and propose a Bi-rLSTM to read the hypothesis. Their proposed rLSTM–“re-read” LSTM unit–takes the attention vector of one sentence as an inner state while reading the other sentence. The output of the standard BiLSTM is also taken as the general input of the bidirectional rLSTM. The currently published next best performing simple system, Parikh et al. (2016) at 86.3% accuracy, introduced use of the attention mechanism for the NLI task, the way it is generally being used today.[7]

4.2 Example permutation in learning

Morishita et al. (2017) explored the effect of mini-batching on the learning of Neural Machine Translation models, carrying out their experiments on two datasets (two language-pairs). In particular, they studied the strategies of (1) sorting by length of the source sentence, (2) target sentence, or (3) both, among other things. They empirically compare their efficiency on two translation tasks and find that some strategies in wide use are not necessarily optimal for accuracy and convergence-wise. In contrast to the work described here however, one of the sorting strategies produced best results, though no comparison was made with the original ordering of examples. By contrast, we show that it is by making non-canonical (semi-non-random) orderings of the data that best results are achieved in NLI, for the two available datasets.

---

[6] We experimented with alternative OOV representations, such as the the mean vector of most frequent words and random vectors, with insignificant effects.

[7] Parikh et al. (2016) also have a result with intra-sentential attention that performs with 86.8% accuracy. However this attention model is dependent on word positional information.

		SNLI			**MULTI-NLI**	
permutation	FF/SUM	Bi/LSTM	diff	FF/SUM	Bi/LSTM	diff
orig	85.64	87.53	1.89	72.29	76.43	4.14
orig-r	82.48	83.92	1.43	66.71	69.71	3.0
shuffle-once (5 runs)	82.88	84.15	1.27	72.39	75.86	3.47
	($\sigma = 0.19$)	($\sigma = 0.18$)		($\sigma = 0.08$)	($\sigma = 0.18$)	
shuffle-epoch	83.09	84.69	1.60	72.15	75.65	3.50
conf	83.35	83.96	0.61	67.12	69.29	2.17
conf-r	81.4	83.95	2.55	66.93	69.25	2.32
prem	83.0	83.74	0.74	67.56	70.11	2.55
prem-r	82.68	83.49	0.81	66.6	69.7	3.1
hypo	82.61	83.82	1.22	67.71	70.03	2.31
hypo-r	82.42	83.68	1.26	66.97	69.77	2.8
lengths	82.74	83.69	0.96	67.11	70.04	2.92
lengths-r	82.65	83.55	0.9	66.93	69.93	3.01

Table 2: Performance of the FF/SUM and Bi/LSTM neural networks for the training example permutations, evaluated over the development set, along with the difference (diff) in performance between the two architectures.

5 Experiments

Data permutations. We consider the following original and sorted orderings of the training set examples to learn our models.

orig: The original order in which the dataset is currently distributed.

prem: Sorting by increasing premise length.

hypo: Sorting by increasing hypothesis length.

lengths: Sorting by increasing premise + hypothesis length.

conf: Sorting by increasing score, where the scores for each example are generating in training Bi/LSTM over orig.

shuffle-once: Randomly shuffle once before training, averaged over 5 runs.

shuffle-epoch: Randomly shuffle on each epoch during training, for a single run.

In addition to each of these, we consider the reversal of the order (indicated by the suffix **-r**.)

In order to not exhaust the test set, we generate the results over data permutations on the development set. These are given in Table 2.

Discussion. For both datasets, we observe that all other training example permutations result in a substantial drop in performance, by approximately 3-4% on SNLI and 1-6% on MULTI-NLI. Even a simple reversal of the original order leads to a substantial drop in performance. Shuffling consistently the data provides the strongest alternative training conditions to the original ordering.

Moreover, the difference in performance of the two separate architectures is generally much lower on all other permutations of the training data, calling into question the significance of the more complex components. These observations apply to both randomly shuffling (as advised in statistical learning practise) and ordering the data by length (as advised in deep learning for NLP practise).

For an analysis of the sorting permutations, we looked into whether hypothesis sentence lengths differed by class to such an extent that the dataset became sorted by class label. We cannot include the results here due to space constraints. However, we observed that ordering by class results in small (and quite interesting) drops in performance, so long as the original order is preserved. If we first randomly shuffle the examples before ordering them by class, similar drops in performance result to those in Table 2.

6 Conclusions

We have shown that models induced over the SNLI and MULTI-NLI datasets are greatly affected by the permutation of the training data instances at hand: recommended statistical learning or deep learning engineering strategies for ordering the training examples result in significantly and even substantially worse performance over these datasets. Our models are simple (no information over word embedding representations of sentences and no ensembles), but strongly competitive with (or better performing than) both SOTA (re-)implementations of much more complex neural systems. We make the straightforward hypothesise that these observations will extend to more complex models; we leave this to be verified by future work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2015. A large annotated corpus for learning natural language inference .

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*. ACL, Vancouver, BC, Canada.

Yichen Gong, Heng Luo, and Jian Zhange. 2017. Natural language inference over interaction space. Technical report. ArXiv:1709.04348 [cs.CL].

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL, New Orleans, Louisiana, USA.

Martin Mirakyan, Karen Hambardzumyan, and Hrant Khachatrian. 2018. Natural Language Inference over Interaction Space: ICLR 2018 Reproducibility Report. *ArXiv e-prints* .

Makoto Morishita, Yusuke Oda, Graham Neubig, Koichiro Yoshino, Katsuhito Sudoh, and Satoshi Nakamura. 2017. An empirical study of mini-batch creation strategies for neural machine translation. In *The First Workshop on Neural Machine Translation (NMT)*. Vancouver, Canada.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proc of EMNLP*. Austin, Texas, USA.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read lstm unit for textual entailment recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 2870–2879.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*.

A. Williams, N. Nangia, and S. R. Bowman. 2017. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. *ArXiv e-prints* .

Bridging Knowledge Gaps in Neural Entailment via Symbolic Models

Dongyeop Kang[1] **Tushar Khot**[2] **Ashish Sabharwal**[2] **Peter Clark**[2]
[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[2]Allen Institute for Artificial Intelligence, Seattle, WA, USA
`{dongyeok}@cs.cmu.edu` `{tushark,ashishs,peterc}@allenai.org`

Abstract

Most textual entailment models focus on lexical gaps between the premise text and the hypothesis, but rarely on knowledge gaps. We focus on filling these knowledge gaps in the Science Entailment task, by leveraging an external structured knowledge base (KB) of science facts. Our new architecture combines standard neural entailment models with a knowledge lookup module. To facilitate this lookup, we propose a fact-level decomposition of the hypothesis, and verifying the resulting sub-facts against both the textual premise and the structured KB. Our model, NSnet, learns to aggregate predictions from these heterogeneous data formats. On the SciTail dataset, NSnet outperforms a simpler combination of the two predictions by 3% and the base entailment model by 5%.

1 Introduction

Textual entailment, a key challenge in natural language understanding, is a sub-problem in many end tasks such as question answering and information extraction. In one of the earliest works on entailment, the PASCAL Recognizing Textual Entailment Challenge, Dagan et al. (2005) define entailment as follows: text (or premise) P entails a hypothesis H if *typically* a human reading P would infer that H is *most likely* true. They note that this informal definition is “based on (and assumes) common human understanding of language as well as *common background knowledge*”.

While current entailment systems have achieved impressive performance by focusing on the language understanding aspect, these systems, especially recent neural models (e.g. Parikh et al., 2016; Khot et al., 2018), do not directly address the need for filling knowledge gaps by leveraging common background knowledge.

Figure 1 illustrates an example of P and H from SciTail, a recent science entailment dataset (Khot et al., 2018), that highlights the challenge of *knowledge gaps*—sub-facts of H that aren’t stated in P but are universally true. In this example, an entailment system that is strong at filling lexical gaps may align *large blood vessel* with *major artery* to help conclude that P entails H. Such a system, however, would equally well—but incorrectly—conclude that P entails a hypothetical variant H’ of H where *artery* is replaced with *vein*. A typical human, on the other hand, could bring to bear a piece of background knowledge, that aorta is a major artery (not a vein), to break the tie.

P: The aorta is a large blood vessel that moves blood away from the heart to the rest of the body.

H (entailed): Aorta is the major artery carrying recently oxygenated blood away from the heart.

H’ (not entailed): Aorta is the major vein carrying recently oxygenated blood away from the heart.

Figure 1: Knowledge gap: Aorta is a major artery (not a vein). *Large blood vessel* soft-aligns with *major artery* but also with *major vein*.

Motivated by this observation, we propose a new entailment model that combines the strengths of the latest neural entailment models with a structured knowledge base (KB) lookup module to bridge such knowledge gaps. To enable KB lookup, we use a fact-level decomposition of the hypothesis, and verify each resulting sub-fact against both the premise (using a standard entailment model) and against the KB (using a structured scorer). The predictions from these two modules are combined using a multi-layer “aggregator” network. Our system, called NSnet, achieves 77.9% accuracy on SciTail, substantially improving over the baseline neural entailment model, and comparable to the structured entailment model proposed by Khot et al. (2018).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4940–4945
Brussels, Belgium, October 31 - November 4, 2018.

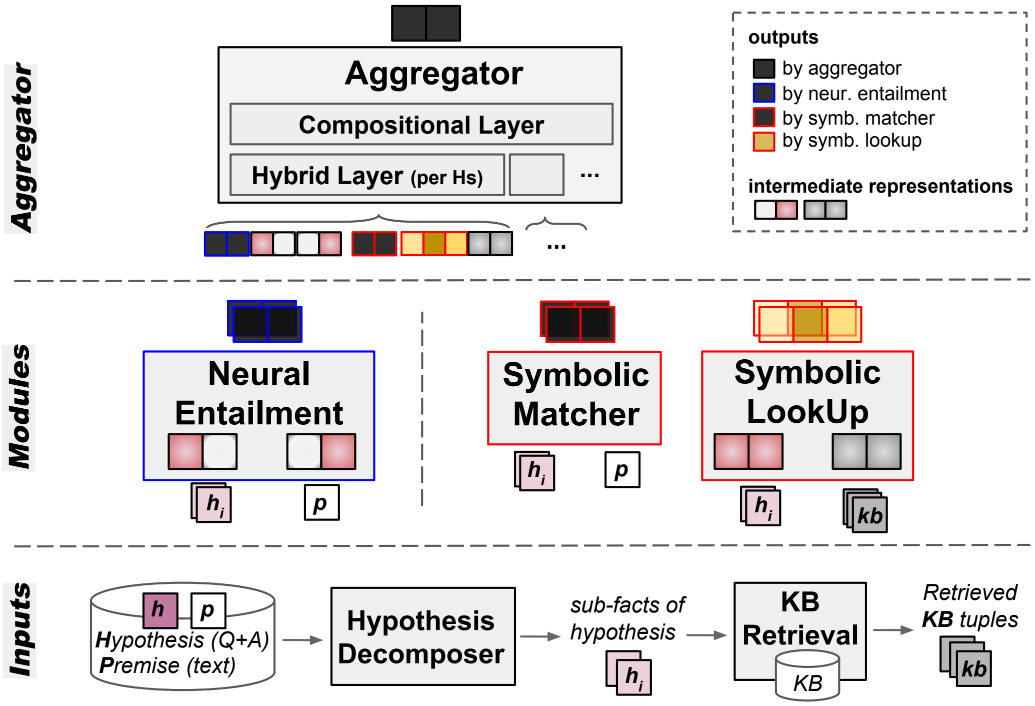

Figure 2: Neural-symbolic learning in NSnet. The bottom layer has QA and their supporting text in SciTail, and the knowledge base (KB). The middle layer has three modules: Neural Entailment (blue) and Symbolic Matcher and Symbolic Lookup (red). The top layer takes the outputs (black and yellow) and intermediate representation from the middle modules, and hierarchically trains with the final labels. All modules and aggregator are jointly trained in an end-to-end fashion.

2 Neural-Symbolic Learning

A general solution for combining neural and symbolic modules remains a challenging open problem. As a step towards this, we present a system in the context of neural entailment that demonstrates a successful integration of the KB lookup model and simple overlap measures, opening up a path to achieve a similar integration in other models and tasks. The overall system architecture of our neural-symbolic model for textual entailment is presented in Figure 2. We describe each layer of this architecture in more detail in the following sub-sections.

2.1 Inputs

We decompose the hypothesis and identify relevant KB facts in the bottom "inputs" layer (Fig. 2).

Hypothesis Decomposition: To identify knowledge gaps, we must first identify the facts stated in the hypothesis $h = (h_1, h_2..)$. We use ClausIE (Del et al., 2013) to break h into sub-facts. ClausIE tuples need not be verb-mediated and generate multiple tuples derived from conjunctions, leading to higher recall than alternatives such as Open IE (Banko et al., 2007). [1]

Knowledge Base (KB): To verify these facts, we use the largest available clean knowledge base for the science domain (Dalvi et al., 2017), with 294K simple facts, as input to our system. The knowledge base contains subject-verb-object (SVO) tuples with short, one or two word arguments (e.g., hydrogen; is; element). Using these simple facts ensures that the KB is only used to fill the basic knowledge gaps and not directly prove the hypothesis irrespective of the premise.

KB Retrieval: The large number of tuples in the knowledge base makes it infeasible to evaluate each hypothesis sub-fact against the entire KB. Hence, we retrieve the top-100 relevant knowledge tuples, K', for each sub-fact based on a simple Jaccard word overlap score.

---

[1] While prior work on question answering in the science domain has successfully used Open IE to extract facts from sentences (Khot et al., 2017), one of the key reasons for errors was the lossy nature of Open IE.

2.2 Modules

We use a Neural Entailment model to compute the entailment score based on the premise, as well as two symbolic models, Symbolic Matcher and Symbolic Lookup, to compute entailment scores based on the premise and the KB respectively (middle layer in Fig. 2).

Neural Entailment We use a simple neural entailment model, Decomposable Attention (Parikh et al., 2016), one of the state-of-the-art models on the SNLI entailment dataset (Bowman et al., 2015). However, our architecture can just as easily use any other neural entailment model. We initialize the model parameters by training it on the Science Entailment dataset. Given the sub-facts from the hypothesis, we use this model to compute an entailment score $n(h_i, p)$ from the premise to each sub-fact h_i.

Symbolic Matcher In our initial experiments, we noticed that the neural entailment models would often either get distracted by similar words in the distributional space (false positives) or completely miss an exact mention of h_i in a long premise (false negatives). To mitigate these errors, we define a Symbolic Matcher model that compares exact words in h_i and p, via a simple asymmetric bag-of-words overlap score:

$$m(h_i, p) = \frac{|h_i \cap p|}{|p|}$$

One could instead use more complex symbolic alignment methods such as integer linear programming (Khashabi et al., 2016; Khot et al., 2017).

Symbolic Lookup This module verifies the presence of the hypothesis sub-fact h_i in the retrieved KB tuples K', by comparing the sub-fact to each tuple and taking the maximum score. Each field in the KB tuple kb_j is scored against the corresponding field in h_i (e.g., subject to subject) and averaged across the fields. To compare a field, we use a simple word-overlap based Jaccard similarity score, $Sim(a, b) = \frac{|a \cap b|}{|a \cup b|}$. The lookup match score for the entire sub-fact and kb-fact is:

$$Sim_f(h_i, kb_j) = \left(\sum_k Sim(h_i[k], kb_j[k]) \right) / 3$$

and the final lookup module score for h_i is:

$$l(h_i) = \max_{kb_j \in K'} Sim_f(h_i, kb_j)$$

Note that the Symbolic Lookup module assesses whether a sub-fact of H is universally true. Neural models, via embeddings, are quite strong at mediating between P and H. The goal of the KB lookup module is to complement this strength, by verifying universally true sub-facts of H that may not be stated in P (e.g. "aorta is a major artery" in our motivating example).

2.3 Aggregator Network

For each sub-fact h_i, we now have three scores: $n(h_i, p)$ from the neural model, $m(h_i, p)$ from the symbolic matcher, and $l(h_i)$ from the symbolic lookup model. The task of the Aggregator network is to combine these to produce a single entailment score. However, we found that using only the final predictions from the three modules was not effective. Inspired by recent work on skip/highway connections (He et al., 2016; Srivastava et al., 2015), we supplement these scores with intermediate, higher-dimensional representations from two of the modules.

From the Symbolic Lookup model, we use the representation of each sub-fact $h_i^{enc} = Enc(h_i)$ obtained by averaging word embeddings (Pennington et al., 2014) and individual similarity scores over the top-100 KB tuples $emb_i = [\ldots, Sim_f(h_i, kb_j), \ldots]$. From the neural entailment model, we use the intermediate representation of both the sub-fact of hypothesis and premise text from the final layer (before the softmax computation), $n^v(h_i, p) = [v_1; v_2]$.

We define a hybrid layer that takes as input a simple concatenation of these representation vectors from the different modules:

$$\begin{aligned} in(h_i, p) =& [h_i^{enc}; l(h_i); m(h_i, p); n(h_i, p); \\ & emb_i; n^v(h_i, p)] \end{aligned}$$

The hybrid layer is a single layer MLP for each sub-fact h_i that outputs a sub-representation $out_i = MLP(in(h_i, p))$. A compositional layer then uses a two-layer MLP over a concatenation of the hybrid layer outputs from different sub-facts, $\{h_1, \ldots, h_I\}$, to produce the final label,

$$label = \text{MLP}([out_1; out_2; \cdots out_I])$$

Finally, we use the cross-entropy loss to train the Aggregator network jointly with representations in the neural entailment and symbolic lookup models, in an end-to-end fashion. We refer to this entire architecture as the NSnet network.

Table 1: Entailment accuracies on the SciTail dataset. NSnet substantially improves upon its base model and marginally outperforms DGEM.

Entailment Model	**Valid.**	**Test**
Majority classifier	63.3	60.3
DecompAttn (Base model)	73.1	74.3
DecompAttn + HypDecomp	71.8	72.7
DGEM	**79.6**	77.3
ENSEMBLE (this work)	75.2	74.8
NSnet (this work)	77.4	**77.9**

To assess the effectiveness of the aggregator network, we also use a simpler baseline model, ENSEMBLE, that works as follows. For each sub-fact h_i, it combines the predictions from each model using a probabilistic-OR function, assuming the model score P_m as a probability of entailment. This function computes the probability of at least one model predicting that h_i is entailed, i.e. $P(h_i) = 1 - \Pi_m(1 - P_m)$ where $m \in n(h_i, p), m(h_i, p), l(h_i)$. We average the probabilities from all the facts to get the final entailment probability.[2]

3 Experiments

We use the SciTail **dataset**[3] (Khot et al., 2018) for our experiments, which contains 27K entailment examples with a 87.3%/4.8%/7.8% train/dev/test split. The premise and hypothesis in each example are natural sentences authored independently as well as independent of the entailment task, which makes the dataset particularly challenging. We focused mainly on the SciTail dataset, since other crowd-sourced datasets, large enough for training, contained limited linguistic variation (Gururangan et al., 2018) leading to limited gains achievable via external knowledge.

For **background knowledge**, we use version v4 of the aforementioned Aristo TupleKB[4] (Dalvi et al., 2017), containing 283K basic science facts. We compare our proposed models to Decomposed Graph Entailment Model (DGEM) (Khot et al., 2018) and Decomposable Attention Model (DecompAttn) (Parikh et al., 2016).

3.1 Results

Table 1 summarizes the validation and test accuracies of various models on the SciTail dataset. The DecompAttn model achieves 74.3% on the test set but drops by 1.6% when the hypotheses are decomposed. The ENSEMBLE approach uses the same hypothesis decomposition and is able to recover 2.1% points by using the KB. The end-to-end NSnet network is able to further improve the score by 3.1% and is statistically significantly (at p-value 0.05) better than the baseline neural entailment model. The model is marginally better than DGEM, a graph-based entailment model proposed by the authors of the SciTail dataset We show significant gains over our base entailment model by using an external knowledge base, which are comparable to the gains achieved by DGEM through the use of hypothesis structure. These are orthogonal approaches and one could replace the base DecompAttn model with DGEM or more recent models (Tay et al., 2017; Yin et al., 2018).

Table 2: Ablation: Both Symbolic Lookup and Symbolic Matcher have significant impact on NSnet performance.

	Valid.	**Test**
NSnet	77.39	77.94
- Symbolic Matcher	76.46	74.73 (**-3.21%**)
- Symbolic Lookup	75.95	75.80 (**-2.14%**)
- Both	75.10	73.98 (**-3.96%**)

In Table 2, we evaluate the impact of the Symbolic Matcher and Symbolic Lookup module on the best reported model. As we see, removing the symbolic matcher, despite its simplicity, results in a 3.2% drop. Also, the KB lookup model is able to fill some knowledge gaps, contributing 2.1% to the final score. Together, these symbolic matching models contribute 4% to the overall score.

3.2 Qualitative Analysis

Figure 3 shows few randomly selected examples in test set. The first two examples show cases when the symbolic models help to change the neural alignment's prediction (F) to correct prediction (T) by our proposed ENSEMBLE or NSnet models. The third question shows a case where the NSnet architecture learns a better combination of the neural and symbolic methods to correctly identify the entailment relation while ENSEMBLE fails to do so.

---

[2] While more intuitive, performing an AND aggregation resulted in worse performance (cf. Appendix **??** for details).

[3] `http://data.allenai.org/scitail`

[4] `http://data.allenai.org/tuple-kb`

Table 3: Few randomly selected examples in the test set between symbolic only, neural only, Ensemble and NSnet inference. The symbolic only model shows its the most similar knowledge from knowledge base inside parenthesis. The first two example shows when knowledge helps fill the gap where neural model can't. The third example shows when NSnet predicts correctly while Ensemble fails.

Premise: plant cells possess a cell wall , animals never .

Hypothesis: a cell wall is found in a plant cell but not in an animal cell .

Sub-fact of hypothesis	**neural only**	**symbolic only**	**Ensemble**	**NSnet**
a cell wall is found in a plant cell but not in an animal cell	F(0.47)	T(0.07) (cell is located in animal)	T(0.50)	-
Prediction (true label: **T** (entail))	**F**	**T**	**T**	**T**

Premise: the pupil is a hole in the iris that allows light into the eye .

Hypothesis: the pupil of the eye allows light to enter .

Sub-fact of hypothesis	**neural only**	**symbolic only**	**Ensemble**	**NSnet**
the pupil of the eye allows light to enter	F(0.43)	T(0.12), (light enter eye)	T(0.50)	-
Prediction (true label: **T** (entail))	**F**	**T**	**T**	**T**

Premise: binary fission in various single-celled organisms (left) .

Hypothesis: binary fission is a form of cell division in prokaryotic organisms that produces identical offspring .

Sub-facts of hypothesis	**neural only**	**symbolic only**	**Ensemble**	**NSnet**
binary fission is a form of cell division in prokaryotic organisms	F(0.49)	T(0.07) (binary fission involve division)	T(0.52)	-
binary fission is a form	F(0.63)	T(0.1) (phase undergo binary fission)	T(0.66)	-
a form of cell division in prokaryotic organisms produces identical offspring	F(0.46)	T(0.05) (cell division occur in tissue)	T(0.48)	-
Prediction (true label: **T** (entail))	**F**	**T**	**F**	**T**

4 Related Work

Compared to neural only (Bowman et al., 2015; Parikh et al., 2016) or symbolic only (Khot et al., 2017; Khashabi et al., 2016) systems, our model takes advantage of both systems, often called neural-symbolic learning (Garcez et al., 2015). Various neural-symbolic models have been proposed for question answering (Liang et al., 2016) and causal explanations (Kang et al., 2017). We focus on end-to-end training of these models specifically for textual entailment.

Contemporaneous to this work, Chen et al. (2018) have incorporated knowledge-bases within the attention and composition functions of a neural entailment model, while Kang et al. (2018) generate adversarial examples using symbolic knowledge (e.g., WordNet) to train a robust entailment model. We focused on integrating knowledge-bases via a separate symbolic model to fill the knowledge gaps.

5 Conclusion

We proposed a new entailment model that attempts to bridge knowledge gaps in textual entailment by incorporating structured knowledge base lookup into standard neural entailment models. Our architecture, NSnet, can be trained end-to-end, and achieves a 5% improvement on SciTail over the baseline neural model used here. The methodology can be easily applied to more complex entailment models (e.g., DGEM) as the base neural entailment model. Accurately identifying the subfacts from a hypothesis is a challenging task in itself, especially when dealing with negation. Improvements to the fact decomposition should further help improve the model.

Acknowledgments

We thank the anonymous reviewers as well as Luke Zettlemoyer, Matt Gardner, Oyvind Tafjord, Julia Hockenmaier and Eduard Hovy for their comments and suggestions.

References

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. 2018. Natural language inference with external knowledge. In *ACL*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *MLCW*.

Bhavana Dalvi, Niket Tandon, and Peter Clark. 2017. Domain-targeted, high precision knowledge extraction. *TACL*.

Corro Luciano Del, Rainer Gemulla, and Rainer Gemulla. 2013. ClausIE: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366. ACM.

A dAvila Garcez, Tarek R Besold, Luc De Raedt, Peter Földiak, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luis C Lamb, Risto Miikkulainen, and Daniel L Silver. 2015. Neural-symbolic learning and reasoning: contributions and challenges. In *Proceedings of the AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches, Stanford*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *NAACL (Short Papers)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen, and Eduard Hovy. 2017. Detecting and explaining causes from text for a time series event. In *EMNLP*.

Dongyeop Kang, Tushar Khot, Ashish Sabharwal, and Eduard Hovy. 2018. Adventure: Adversarial training for textual entailment with knowledge-guided examples. In *ACL*, Melbourne, Australia.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *IJCAI*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. In *ACL*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. *AAAI*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*.

Wenpeng Yin, Dan Roth, and Hinrich Schütze. 2018. End-task oriented textual entailment via deep exploring inter-sentence interactions. In *ACL*.

The BQ Corpus: A Large-scale Domain-specific Chinese Corpus For Sentence Semantic Equivalence Identification

Jing Chen[†], **Qingcai Chen**[#][*], **Xin Liu**[†], **Haijun Yang**[‡], **Daohe Lu**[‡], **Buzhou Tang**[†]
[†#]Shenzhen Calligraphy Digital Simulation Technology Lab,
Harbin Institute of Technology, Shenzhen, China
[‡]WeBank Inc.
[†]`{mcdh.chenjing,hit.liuxin,tangbuzhou}@gmail.com`
[#]`qingcai.chen@hit.edu.cn`
[‡]`{navyyang,leslielu}@webank.com`

Abstract

This paper introduces the Bank Question (BQ) corpus, a Chinese corpus for sentence semantic equivalence identification (SSEI). The BQ corpus contains 120,000 question pairs from 1-year online bank custom service logs. To efficiently process and annotate questions from such a large scale of logs, this paper proposes a clustering based annotation method to achieve questions with the same intent. First, the deduplicated questions with the same answer are clustered into stacks by the Word Mover's Distance (WMD) based Affinity Propagation (AP) algorithm. Then, the annotators are asked to assign the clustered questions into different intent categories. Finally, the positive and negative question pairs for SSEI are selected in the same intent category and between different intent categories respectively. We also present six SSEI benchmark performance on our corpus, including state-of-the-art algorithms. As the largest manually annotated public Chinese SSEI corpus in the bank domain, the BQ corpus is not only useful for Chinese question semantic matching research, but also a significant resource for cross-lingual and cross-domain SSEI research. The corpus is available in public[1].

1 Introduction

As the semantic matching task, sentence semantic equivalence identification (SSEI) is a fundamental task of natural language processing (NLP) in question answering (QA), automatic customer service and chat-bots. In customer service systems, two questions are defined as semantically equivalent if they convey the same intent or they could be answered by the same answer. Because of rich expressions in natural languages, SSEI is really a challenging NLP task.

Compared with other NLP tasks, the lack of large-scale SSEI corpora is one of the biggest obstacles for SSEI algorithm development. To address this issue, several corpora have been provided in recent years, including the Microsoft Research Paraphrase (MSRP) Corpus (Dolan et al., 2004; Dolan and Brockett, 2005), the Twitter Paraphrase Corpus (PIT-2015 corpus) (Xu et al., 2014, 2015), the Twitter URL corpus (Lan et al., 2017) and the Quora dataset [2].

In the early stage, the MSRP corpus was used to validate paraphrase identification algorithms based on a set of linguistic features (Kozareva and Montoyo, 2006; Mihalcea et al., 2006; Rus et al., 2008). Then, MSRP was also used to validate the deep models within a long duration. The deep convolutional neural networks (DCNNs), recurrent neural networks (RNNs), and their variants, such as Arc-I, Arc-II and BiMPM etc., have been developed and verified on it, even though it contains only thousands of sentence pairs (Hu et al., 2014; Yin and Schütze, 2015; Wang et al., 2016, 2017). Until 2015, the SemEval 2015 released a larger corpus, the PIT-2015 corpus for paraphrase and semantic similarity identification tasks. On this corpus, participants adopted SVM classifiers, logistic regression models, referential translation machines (RTM) and neural networks (Xu et al., 2015). In 2017, a large-scale SSEI corpus named Quora was released, which greatly boost the development of deep matching algorithms. Tomar et al. (2017) proposed a variant of the decomposable attention model. Gong et al.(2018) proposed a Densely Interactive Inference Network (DIIN) by hierarchically extracting semantic features from interaction space. However, the Quora corpus comes from social network sites. Consider-

---

[*] Corresponding author

[1] http://icrc.hitsz.edu.cn/Article/show/175.html

[2] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[3] 57,037 pairs out of them are manually labeled.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4946–4951
Brussels, Belgium, October 31 - November 4, 2018.

Corpus	Language	Source	Scale	Sentence Length	pos:neg
MSRP	English	news	5801 sentence pairs	18.9 words	2.05:1
PIT-2015 corpus	English	tweets	18,762 sentence pairs	11.9 words	-
The Twitter URL corpus	English	tweets	676,050 sentence pairs[3]	15 words	1:4.93
The Quora dataset	English	Quora	404,290 question pairs	11.1 words	1:1.71
The Bank Question corpus	Chinese	bank	120,000 question pairs	11.9 words	1:1

Table 1: The comparison of public corpus related to paraphrase or semantic similarity.

ing the request for specific domains and real context in SSEI, there still lacks corpora from different domains and corpora with features of non-English languages.

In this paper, we present a large-scale Chinese SSEI corpus constructed from real bank customer service logs. The main contributions of this paper include: 1) we present a large-scale domain-specific Chinese SSEI corpus, which contains 120,000 manually annotated sentence pairs; 2)we propose the Affinity Propagation (AP) (Frey and Dueck, 2007) clustering based method for SSEI corpus construction from a large number of sentences; 3)we provide the benchmark performance of 5 representative algorithms on our corpus. Hopefully, these contributions are useful in promoting the research on Chinese SSEI methods and the transferring methods for cross languages or cross domains.

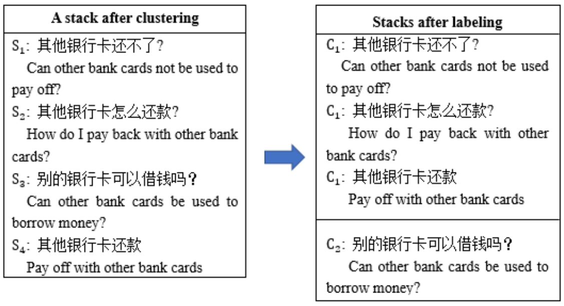

A stack after clustering

S_1: 其他银行卡还不了?
Can other bank cards not be used to pay off?
S_2: 其他银行卡怎么还款?
How do I pay back with other bank cards?
S_3: 别的银行卡可以借钱吗？
Can other bank cards be used to borrow money?
S_4: 其他银行卡还款
Pay off with other bank cards

Stacks after labeling

C_1: 其他银行卡还不了?
Can other bank cards not be used to pay off?
C_1: 其他银行卡怎么还款?
How do I pay back with other bank cards?
C_1: 其他银行卡还款
Pay off with other bank cards

C_2: 别的银行卡可以借钱吗？
Can other bank cards be used to borrow money?

Figure 1: The examples from clustering to labeling

2 The Bank Question Corpus

2.1 Features of the Bank Corpus

As the first domain-specific large-scale Chinese SSEI corpus, the Bank Corpus contains 120,000 question pairs. It is split into three parts: 100,000 pairs for training, 10,000 pairs for validation, and 10,000 pairs for test. There is no sentence overlap among training, validation and test sets. The last line in Table 1 shows the main features of our corpus. We also highlight features of the most popular SSEI corpus from line 1 to 4 in Table 1. The further analysis of the Bank Corpus will be shown in following sections.

2.2 Construction of the Bank Corpus

The original data came from the 1-year customer service logs with more than 20 millions of questions provided by a Chinese bank. To manually annotate so many questions is unimaginable, so we conducted three steps to get the SSEI corpus, including the clustering of questions, the intent-based annotation and the combination of semantic equivalent question pairs.

Grouping and Clustering At first, two definitions are given: a set of questions replied by the same answer is called a *group*; The clusters generated by an automatic clustering algorithm in a group is called a *stack*. Here, a stack is a subset of a group with questions have the same intent. First, the users' questions were divided into *groups* by their respective answers. The de-duplication is then executed on each group. Next, we used the Word Mover's Distance (WMD) (Kusner et al., 2015) based Affinity Propagation (AP) clustering algorithm to split the questions within each group into multiple question *stacks*. After filtering some emoijs and sentences which are standard answers from the questions, we finally got 799 distinct groups and selected total of 55724 questions from all groups for annotation.

Annotation We adopted two steps to annotate the question stacks. First, we recruited 12 annotators to categorize questions in each clustered stack into different intent classes. Here, if the questions express the same intent, we think that they belong to an intent class. For each labeled stack, the classes of intent are the same. If a question is chit-chat or it can not be combined with other questions into an intent class, it will be put into a specific class called ”other”. Second, the experts related to this specific domain were requested to check and correct the annotated intent classes. After annotation, we got 953 groups and 18002 questions. Among the questions, 16680

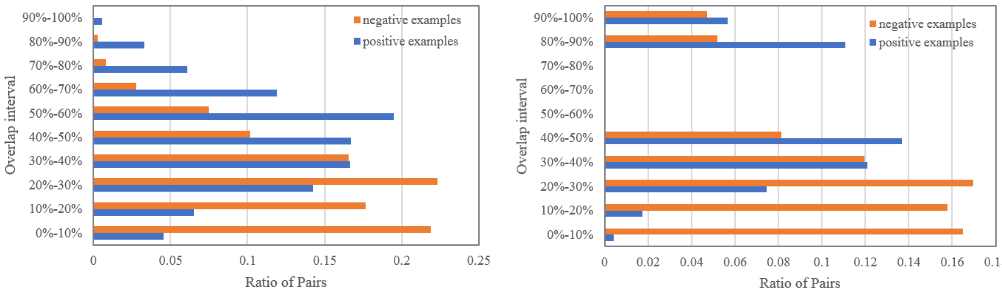

(a) The overlap distribution of our corpus (b) The overlap distribution of the Quora corpus

Figure 2: The overlap distribution of positive and negative data

Examples			
most common:	一次性还完可以继续借吗 Can I continue to borrow money when I repay the loan at one time	一笔借款提前还清还可以借款吗 Can I borrow money again when a loan be pay off ahead of time	1
low overlap:	这是什么公司的产品 What's the company of the product	企业名称 the enterprise name	1
most common:	我几号还款 When should I repay my loan	今天的还款，可以推迟么? Can I postpone today's repayment?	0
high overlap:	能不能取消申请 Can I cancel the application	能不能取消，重新申请 Can I cancel the application and reapply	0

Table 2: The most common examples, the positive example with low overlap and the negative example with high overlap. Chinese sentences are original-form examples and English sentences below them are their corresponding translations.

questions are with meaningful intents and can be used to create semantic pairs. There are average 9 stacks in each group. The annotation process from clustering examples to labeled examples is shown as Figure 1. From the clustering results, we find the clustering algorithm cluster the word ”借(borrow)” and the word ”还(pay back)” together. Actually, they convey different intentions and we need to distinguish them.

Generation Based on the labeled stacks, we combine the questions in each stack which have the same intention to create the positive question pairs, and select questions from different stacks in each group which have different intentions to create the negative question pairs.

2.3 Quality of the Corpus

To verify the quality of the corpus, we analyze the word overlap (Dolan et al., 2004) distribution and the PINC (Paraphrase In N-gram Changes) (Kim, 2014) distribution in the positive pairs and the negative pairs respectively.

The overlap is defined as the number of common words between two sentences divided by the average length of them. As shown in Figure 2, the overlap ratio of positive samples on the intervals appears a normal distribution on our new corpus, while the Quora corpus has no examples on overlaps between 50% and 80%. The positive question pairs with overlap ratio below 50% account for 58.67% and the negative question pairs with overlap ratio above 50% account for 11.36%. Here, we just give some examples with the largest ratio among the overlap intervals, some positive examples with low overlap and some negative examples with high overlap as shown in Table 2. For example, the positive question pair (“这是什么公司的产品(What's the company of the product)” and “企业名称(the enterprise name)”) expresses the same intention while they have low overlap, where “什么公司(what company)” has the same meaning as “企业名称(the enterprise name)”. The negative pair(“能不能取消申请(Can I cancel the application)” and “能不能取消，重新申请(Can I cancel the application and reapply)”) is different only on the word “重新(again)” while they nearly convey the contrary meaning. The statistics and examples indicate that except common examples we also have some difficult examples especially the positive pairs with low overlap and our new corpus is meaningful for research on learning methods.

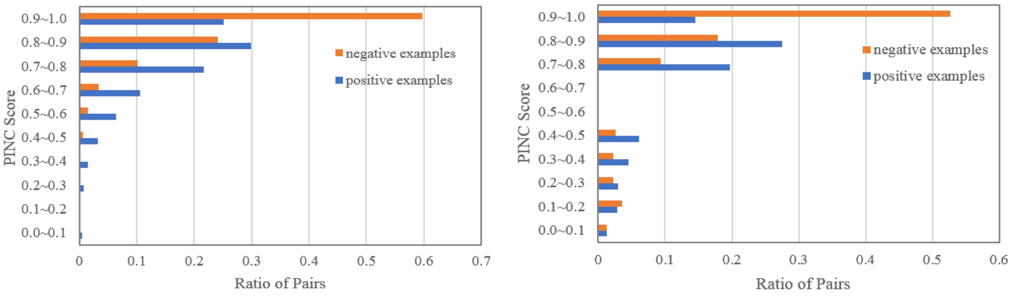

(a) The PINC distribution of our corpus

(b) The PINC distribution of the Quora corpus

Figure 3: The PINC score distribution of positive and negative data

PINC (Kim, 2014) is a score of n-gram differences to measure lexical dissimilarity of sentence pairs. In essence, it is the inverse of BLEU (Papineni et al., 2002). As shown in Figure 3, most of the PINC scores on our corpus are between 0.7 and 1.0 which reveals that our corpus contains more lexical dissimilar question pairs. It contains rich expressions for the same user intention and it is challenging for machine learning methods to identify the semantic equivalence of the question pairs automatically.

3 Semantic Equivalence Identification

For this new public corpus, we provide a benchmark on the question semantic equivalence identification task to better understand its characteristic and provide further evidence for its value.

3.1 Models

Text-CNN (Kim, 2014) is a typical Convolutional Neural Network (CNN) model for sentence classification. We respectively feed each sentence of the question pair into the model with 300-dimensional word vectors and concatenate the sentence representation for SSEI. Here, we trained the word vectors on our new corpus by gensim[4].

BiLSTM (Graves and Schmidhuber, 2005) is an variant of RNN which considers both long and short dependency in context from forward and backward. We use the same structure but constitute the CNN with BiLSTM to model the sentence representation.

BiMPM (Wang et al., 2017) is a bilateral multi-perspective matching model of well performance for natural language sentence matching. The model uses the BiLSTM to learn the sentence representation, matches two sentences from two directions and multi-perspectives, aggregates the matching results with BiLSTM and finally predicts through a fully connected layer.

DIIN (Gong et al., 2018) is a Densely Interactive Inference Network (DIIN) for Natural Language Inference (NLI). It hierarchically extracts semantic features from interaction space to achieve the high-level understanding of sentence pairs. It achieves the state-of-the-art performance on large-scale NLI copora and Quora corpus.

3.2 Results and Discussion

The benchmark performance on our new corpus is shown in Table 3. The performace on the Quora corpus is shown in Table 4. The random method achieves 50.43% which indicates that our new corpus is balanced and meets the basic requirements for SSEI model research.

The TF-IDF method just models the surface features of sentences according to the vocabulary frequency. It can not learn the dependency features in the word sequences and the synonym or near-synonym according to the word meanings. Therefore, it performs not so well, which indicates that the new corpus can not be learned by simple surface features and the deep semantic relationsn need to be mined by deep models.

Here, we use four deep neural network models to verify the new constructed corpus, including two basic and representative models (Text-CNN and BiLSTM) and two latest and well-used models (BiMPM and DIIN) which perform well on the natural language sentence matching task. The results show that the BiLSTM model can learn the dependency features between words in the sentences better than the Text-CNN model. The Accuracy of BiMPM is 81.85% and that of DIIN is 81.41%. Compared with the performance on the Quora corpus, the performance on the BQ corpus

---

[4] https://radimrehurek.com/gensim

Models	Precision	Recall	F1	Accuracy
Random	50.43	50.56	50.49	50.43
TF-IDF	64.68	60.94	62.75	63.83
Text-CNN	67.77	70.64	69.17	68.52
BiLSTM	75.04	70.46	72.68	73.51
BiMPM	**82.28**	**81.18**	**81.73**	**81.85**
DIIN	81.58	81.14	81.36	81.41

Table 3: The comparison of BQ corpus related to paraphrase or semantic similarity.

Models	Precision	Recall	F1	Accuracy
Random	50.73	51.84	51.29	50.72
TF-IDF	63.66	85.16	72.85	68.24
Text-CNN	82.89	71.58	76.82	78.38
BiLSTM	83.44	73.96	78.41	79.62
BiMPM (Wang et al., 2017)	—	—	—	88.17
DIIN (Gong et al., 2018)	—	—	—	**89.06**

Table 4: The comparison of Quora corpus related to paraphrase or semantic similarity.

is lower, which reveals that our new corpus is challenging for semantic matching model research.

4 Conclusion and Future Work

In this paper, we present a large-scale Chinese corpus for question semantic equivalence identification in the bank domain. The construction procedure and benchmark performance are given. To the best of our knowledge, this corpus is the largest manually annotated public Chinese SSEI corpus in the bank domain. Compared with existing corpora, it is of high quality and challenging, and is hopefully useful for research on SSEI, cross-lingual and cross-domain learning.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by Natural Science Foundation of China (Grant No. 61473101, 61573118), Strategic Emerging Industry Development Special Funds of Shenzhen(Grant No. JCYJ20170307150528934, JCYJ20160531192358466) and the joint project foundation of WeBank.

References

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Brendan J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science*, 315(5814):972–976.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. *ICLR*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Zornitsa Kozareva and Andrés Montoyo. 2006. Paraphrase identification on the basis of supervised machine learning techniques. In *Advances in Natural Language Processing, International Conference on Nlp, Fintal 2006, Turku, Finland, August 23-25, 2006, Proceedings*, pages 524–533.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *International Conference on International Conference on Machine Learning*, pages 957–966.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of The 2017 Conference on*

Empirical Methods on Natural Language Processing (EMNLP), pages 1235–1245. Association for Computational Linguistics.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, Usa*, pages 775–780.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Vasile Rus, Philip M. Mccarthy, Mihai C. Lintean, Danielle S. Mcnamara, and Arthur C. Graesser. 2008. Paraphrase identification with lexico-syntactic graph subsumption. In *International Florida Artificial Intelligence Research Society Conference, May 15-17, 2008, Coconut Grove, Florida, Usa*, pages 201–206.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural paraphrase identification of questions with noisy pretraining. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP, Copenhagen, Denmark, September 7, 2017*, pages 142–147.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4144–4150.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1340–1349.

Wei Xu, Chris Callison-Burch, and Bill Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). In *International Workshop on Semantic Evaluation*, pages 1–11.

Wei Xu, Alan Ritter, Chris Callisonburch, William B Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from twitter. pages 435–448.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Interpreting Recurrent and Attention-Based Neural Models: a Case Study on Natural Language Inference

Reza Ghaeini, Xiaoli Z. Fern, Prasad Tadepalli
School of Electrical Engineering and Computer Science, Oregon State University
1148 Kelley Engineering Center, Corvallis, OR 97331-5501, USA
`{ghaeinim, xfern, tadepall}@eecs.oregonstate.edu`

Abstract

Deep learning models have achieved remarkable success in natural language inference (NLI) tasks. While these models are widely explored, they are hard to interpret and it is often unclear how and why they actually work. In this paper, we take a step toward explaining such deep learning based models through a case study on a popular neural model for NLI. In particular, we propose to interpret the intermediate layers of NLI models by visualizing the saliency of attention and LSTM gating signals. We present several examples for which our methods are able to reveal interesting insights and identify the critical information contributing to the model decisions.

1 Introduction

Deep learning has achieved tremendous success for many NLP tasks. However, unlike traditional methods that provide optimized weights for human understandable features, the behavior of deep learning models is much harder to interpret. Due to the high dimensionality of word embeddings, and the complex, typically recurrent architectures used for textual data, it is often unclear how and why a deep learning model reaches its decisions.

There are a few attempts toward explaining/interpreting deep learning-based models, mostly by visualizing the representation of words and/or hidden states, and their importances (via saliency or erasure) on shallow tasks like sentiment analysis and POS tagging (Bahdanau et al., 2014; Denil et al., 2014; Li et al., 2016; Arras et al., 2017; Li et al., 2017; Rei and Søgaard, 2018). In contrast, we focus on interpreting the gating and attention signals of the intermediate layers of deep models in the challenging task of Natural Language Inference. A key concept in explaining deep models is saliency, which determines what is critical for the final decision of a deep model. So far, saliency has only been used to illustrate the impact of word embeddings. In this paper, we extend this concept to the intermediate layer of deep models to examine the saliency of attention as well as the LSTM gating signals to understand the behavior of these components and their impact on the final decision.

We make two main contributions. First, we introduce new strategies for interpreting the behavior of deep models in their intermediate layers, specifically, by examining the saliency of the attention and the gating signals. Second, we provide an extensive analysis of the state-of-the-art model for the NLI task and show that our methods reveal interesting insights not available from traditional methods of inspecting attention and word saliency.

In this paper, our focus was on NLI, which is a fundamental NLP task that requires both understanding and reasoning. Furthermore, the state-of-the-art NLI models employ complex neural architectures involving key mechanisms, such as attention and repeated reading, widely seen in successful models for other NLP tasks. As such, we expect our methods to be potentially useful for other natural understanding tasks as well.

2 Task and Model

In NLI (Bowman et al., 2015), we are given two sentences, a premise and a hypothesis, the goal is to decide the logical relationship (*Entailment*, *Neutral*, or *Contradiction*) between them.

Many of the top performing NLI models (Ghaeini et al., 2018b; Tay et al., 2018; Peters et al., 2018; McCann et al., 2017; Gong et al., 2017; Wang et al., 2017; Chen et al., 2017), are variants of the ESIM model (Chen et al., 2017), which we choose to analyze in this paper. ESIM reads the sentences independently using LSTM at first, and then applies attention to align/contrast

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4952–4957
Brussels, Belgium, October 31 - November 4, 2018.

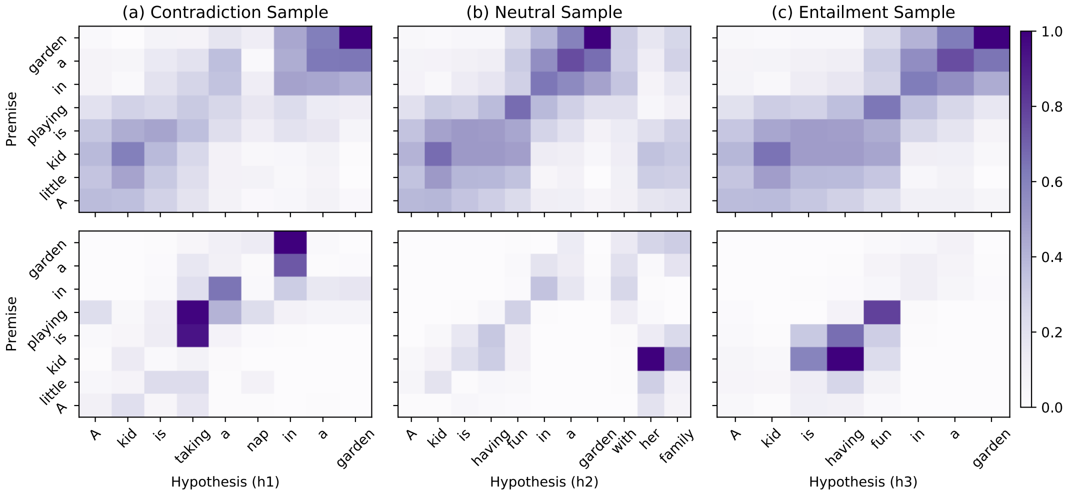

Figure 1: Normalized attention and attention saliency visualization. Each column shows visualization of one sample. Top plots depict attention visualization and bottom ones represent attention saliency visualization. Predicted (the same as Gold) label of each sample is shown on top of each column.

the sentences. Another round of LSTM reading then produces the final representations, which are compared to make the prediction. Detailed description of ESIM can be found in the Supplementary Materials.

Using the SNLI (Bowman et al., 2015) data, we train two variants of ESIM, with dimensionality 50 and 300 respectively, referred to as ESIM-50 and ESIM-300 in the remainder of the paper.

3 Visualization of Attention and Gating

In this work, we are primarily interested in the internal workings of the NLI model. In particular, we focus on the attention and the gating signals of LSTM readers, and how they contribute to the decisions of the model.

3.1 Attention

Attention has been widely used in many NLP tasks (Ghaeini et al., 2018a; Dhingra et al., 2017; Bahdanau et al., 2014) and is probably one of the most critical parts that affects the inference decisions. Several pieces of prior work in NLI have attempted to visualize the attention layer to provide some understanding of their models (Ghaeini et al., 2018b; Parikh et al., 2016). Such visualizations generate a heatmap representing the similarity between the hidden states of the premise and the hypothesis (Eq. 3 of the Supplementary Materials). Unfortunately the similarities are often the same regardless of the decision.

Let us consider the following example, where the same premise *“A kid is playing in the garden”*, is paired with three different hypotheses:

h1: *A kid is taking a nap in the garden*
h2: *A kid is having fun in the garden with her family*
h3: *A kid is having fun in the garden*

Note that the ground truth relationships are Contradiction, Neutral, and Entailment, respectively.

The first row of Fig. 1 shows the visualization of normalized attention for the three cases produced by ESIM-50, which makes correct predictions for all of them. As we can see from the figure, the three attention maps are fairly similar despite the completely different decisions. The key issue is that the attention visualization only allows us to see how the model aligns the premise with the hypothesis, but does not show *how such alignment impacts the decision.* This prompts us to consider the saliency of attention.

3.1.1 Attention Saliency

The concept of saliency was first introduced in vision for visualizing the spatial support on an image for a particular object class (Simonyan et al., 2013). In NLP, saliency has been used to study the importance of words toward a final decision (Li et al., 2016) .

We propose to examine the saliency of attention. Specifically, given a premise-hypothesis pair and the model’s decision y, we consider the similarity between a pair of premise and hypothesis hidden states e_{ij} as a variable. The score of the decision $S(y)$ is thus a function of e_{ij} for all i and j. The saliency of e_{ij} is then defined to be $|\frac{\partial S(y)}{\partial e_{ij}}|$.

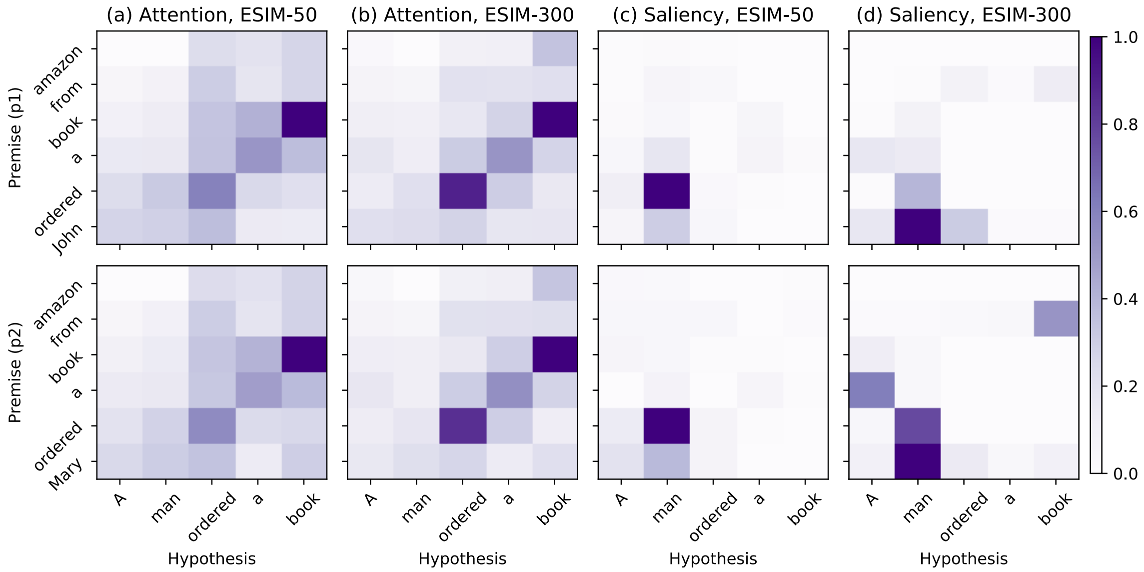

Figure 2: Normalized attention and attention saliency visualizations of two examples (p1 and p2) for ESIM-50 (a) and ESIM-300 (b) models. Each column indicates visualization of a model and each row represents visualization of one example.

The second row of Fig. 1 presents the attention saliency map for the three examples acquired by the same ESIM-50 model. Interestingly, the saliencies are clearly different across the examples, each highlighting different parts of the alignment. Specifically, for h1, we see the alignment between "is playing" and "taking a nap" and the alignment of "in a garden" to have the most prominent saliency toward the decision of Contradiction. For h2, the alignment of "kid" and "her family" seems to be the most salient for the decision of Neutral. Finally, for h3, the alignment between "is having fun" and "kid is playing" have the strongest impact toward the decision of Entailment.

From this example, we can see that by inspecting the attention saliency, we effectively pinpoint which part of the alignments contribute most critically to the final prediction whereas simply visualizing the attention itself reveals little information.

3.1.2 Comparing Models

In the previous examples, we study the behavior of the same model on different inputs. Now we use the attention saliency to compare the two different ESIM models: ESIM-50 and ESIM-300.

Consider two examples with a shared hypothesis of *"A man ordered a book"* and premise:

p1: *John ordered a book from amazon*

p2: *Mary ordered a book from amazon*

Here ESIM-50 fails to capture the gender connections of the two different names and predicts Neutral for both inputs, whereas ESIM-300 correctly predicts Entailment for the first case and Contradiction for the second.

In the first two columns of Fig. 2 (column a and b) we visualize the attention of the two examples for ESIM-50 (left) and ESIM-300 (right) respectively. Although the two models make different predictions, their attention maps appear qualitatively similar.

In contrast, columns 3-4 of Fig. 2 (column c and d) present the attention saliency for the two examples by ESIM-50 and ESIM-300 respectively. We see that for both examples, ESIM-50 primarily focused on the alignment of "ordered", whereas ESIM-300 focused more on the alignment of "John" and "Mary" with "man". It is interesting to note that ESIM-300 does not appear to learn significantly different similarity values compared to ESIM-50 for the two critical pairs of words ("John", "man") and ("Mary", "man") based on the attention map. The saliency map, however, reveals that the two models use these values quite differently, with only ESIM-300 correctly focusing on them.

3.2 LSTM Gating Signals

LSTM gating signals determine the flow of information. In other words, they indicate how LSTM reads the word sequences and how the information from different parts is captured and combined. LSTM gating signals are rarely analyzed, possibly

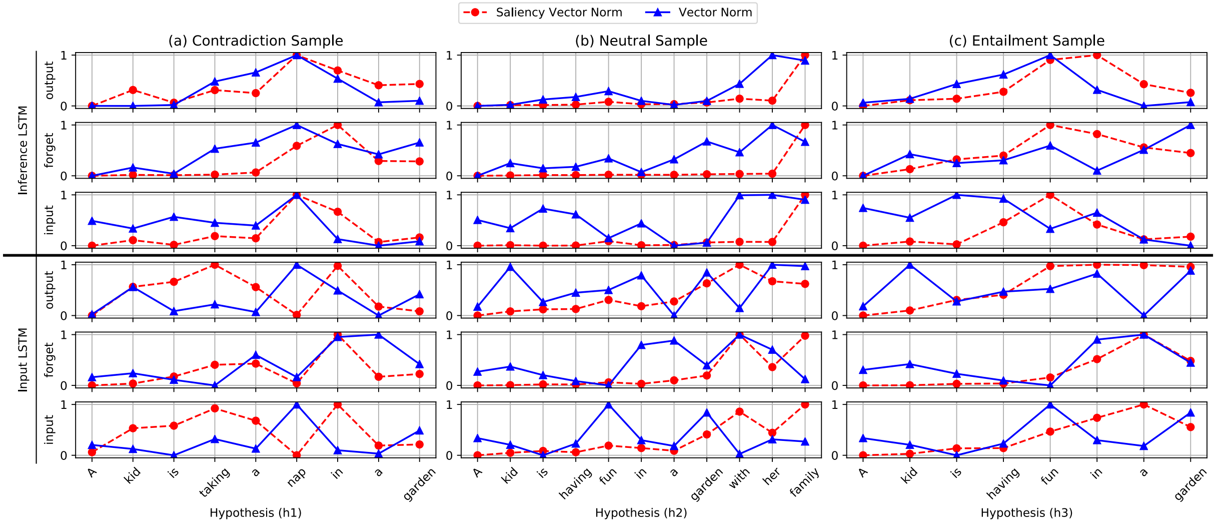

Figure 3: Normalized signal and saliency norms for the input and inference LSTMs (forward) of ESIM-50 for three examples. The bottom (top) three rows show the signals of the input (inference) LSTM. Each row shows one of the three gates (input, forget and output).

due to their high dimensionality and complexity. In this work, we consider both the gating signals and their saliency, which is computed as the partial derivative of the score of the final decision with respect to each gating signal.

Instead of considering individual dimensions of the gating signals, we aggregate them to consider their norm, both for the signal and for its saliency. Note that ESIM models have two LSTM layers, the first (input) LSTM performs the input encoding and the second (inference) LSTM generates the representation for inference.

In Fig. 3 we plot the normalized signal and saliency norms for different gates (input, forget, output)[1] of the Forward input (bottom three rows) and inference (top three rows) LSTMs. These results are produced by the ESIM-50 model for the three examples of Section 3.1, one for each column.

From the figure, we first note that the saliency tends to be somewhat consistent across different gates within the same LSTM, suggesting that we can interpret them jointly to identify parts of the sentence important for the model's prediction.

Comparing across examples, we see that the saliency curves show pronounced differences across the examples. For instance, the saliency pattern of the Neutral example is significantly different from the other two examples, and heavily concentrated toward the end of the sentence ("with her family"). Note that without this part of the sentence, the relationship would have been Entailment. The focus (evidenced by its strong saliency and strong gating signal) on this particular part, which presents information not available from the premise, explains the model's decision of Neutral.

Comparing the behavior of the input LSTM and the inference LSTM, we observe interesting shifts of focus. In particular, we see that the inference LSTM tends to see much more concentrated saliency over key parts of the sentence, whereas the input LSTM sees more spread of saliency. For example, for the Contradiction example, the input LSTM sees high saliency for both "taking" and "in", whereas the inference LSTM primarily focuses on "nap", which is the key word suggesting a Contradiction. Note that ESIM uses attention between the input and inference LSTM layers to align/contrast the sentences, hence it makes sense that the inference LSTM is more focused on the critical differences between the sentences. This is also observed for the Neutral example as well.

It is worth noting that, while revealing similar general trends, the backward LSTM can sometimes focus on different parts of the sentence (e.g., see Fig. 8 of the Supplementary Materials), suggesting the forward and backward readings provide complementary understanding of the sentence.

---

[1] We also examined the memory cell but it shows very similar behavior with the output gate and is hence omitted.

4 Conclusion

We propose new visualization and interpretation strategies for neural models to understand how and why they work. We demonstrate the effectiveness of the proposed strategies on a complex task (NLI). Our strategies are able to provide interesting insights not achievable by previous explanation techniques. Our future work will extend our study to consider other NLP tasks and models with the goal of producing useful insights for further improving these models.

References

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 159–168.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1657–1668.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, visualising and summarising documents with a single convolutional neural network. *CoRR*, abs/1406.3830.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1832–1846.

Reza Ghaeini, Xiaoli Z. Fern, Hamed Shahbazi, and Prasad Tadepalli. 2018a. Dependent gated reading for cloze-style question answering. *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3330–3345.

Reza Ghaeini, Sadid A. Hasan, Vivek V. Datla, Joey Liu, Kathy Lee, Ashequl Qadir, Yuan Ling, Aaditya Prakash, Xiaoli Z. Fern, and Oladimeji Farri. 2018b. Dr-bilstm: Dependent reading bidirectional LSTM for natural language inference. *NAACL HLT 2018, The 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *CoRR*, abs/1709.04348.

Jiwei Li, Xinlei Chen, Eduard H. Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 681–691.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6297–6308.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Marek Rei and Anders Søgaard. 2018. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 293–302.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. A compare-propagate architecture with alignment factorization for natural language inference. *CoRR*, abs/1801.00102.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4144–4150.

Towards Semi-Supervised Learning for Deep Semantic Role Labeling

Sanket Vaibhav Mehta[*] **Jay Yoon Lee**[*] **Jaime Carbonell**

School of Computer Science,
Carnegie Mellon University,
Pittsburgh, PA
`{svmehta,jaylee,jgc}@cs.cmu.edu`

Abstract

Neural models have shown several state-of-the-art performances on Semantic Role Labeling (SRL). However, the neural models require an immense amount of semantic-role corpora and are thus not well suited for low-resource languages or domains. The paper proposes a semi-supervised semantic role labeling method that outperforms the state-of-the-art in limited SRL training corpora. The method is based on explicitly enforcing syntactic constraints by augmenting the training objective with a syntactic-inconsistency loss component and uses SRL-unlabeled instances to train a joint-objective LSTM. On CoNLL-2012 English section, the proposed semi-supervised training with 1%, 10% SRL-labeled data and varying amounts of SRL-unlabeled data achieves +1.58, +0.78 F1, respectively, over the pre-trained models that were trained on SOTA architecture with ELMo on the same SRL-labeled data. Additionally, by using the syntactic-inconsistency loss on inference time, the proposed model achieves +3.67, +2.1 F1 over pre-trained model on 1%, 10% SRL-labeled data, respectively.

1 Introduction

Semantic role labeling (SRL), a.k.a shallow semantic parsing, identifies the arguments corresponding to each clause or proposition, i.e. its semantic roles, based on lexical and positional information. SRL labels non-overlapping text spans corresponding to typical semantic roles such as Agent, Patient, Instrument, Beneficiary, etc. This task finds its use in many downstream applications such as question-answering (Shen and Lapata, 2007), information extraction (Bastianelli et al., 2013), machine translation, etc.

Several SRL systems relying on large annotated corpora have been proposed (Peters et al., 2018; He et al., 2017), and perform relatively well. A more challenging task is to design an SRL method for low resource scenarios (e.g. rare languages or domains) where we have limited annotated data but where we may leverage annotated data from related tasks. Therefore, in this paper, we focus on building effective systems for low resource scenarios and illustrate our system's performance by simulating low resource scenarios for English.

SRL systems for English are built using large annotated corpora of verb predicates and their arguments provided as part of the PropBank and OntoNotes v5.0 projects (Kingsbury and Palmer, 2002; Pradhan et al., 2013). These corpora are built by adding semantic role annotations to the constituents of previously-annotated syntactic parse trees in the Penn Treebank (Marcus et al., 1993). Traditionally, SRL relies heavily on using syntactic parse trees either from shallow syntactic parsers (chunkers) or full syntactic parsers and Punyakanok et al. shows significant improvements by using syntactic parse trees.

Recent breakthroughs motivated by end-to-end deep learning techniques (Zhou and Xu, 2015; He et al., 2017) achieve state-of-the-art performance without leveraging any syntactic signals, relying instead on ample role-label annotations. We hypothesize that by leveraging syntactic structure while training neural SRL models, we may achieve robust performance, especially for low resource scenarios. Specifically, we propose to leverage syntactic parse trees as hard constraints for the SRL task i.e., we explicitly enforce that the predicted argument spans of the SRL network must agree with the spans implied by the syntactic parse of the sentence via scoring function in the training objective. Moreover, we present a semi-supervised learning (SSL) based formulation, wherein we leverage syntactic parse trees for SRL-unlabeled data to build effective SRL for low

---

[*] Equal contribution, name order decided by coin flip.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4958–4963
Brussels, Belgium, October 31 - November 4, 2018.

resource scenarios.

We build upon the state-of-the-art SRL system by (Peters et al., 2018; He et al., 2017), where we formulate SRL as a BIO tagging problem and use multi-layer highway bi-directional LSTMs. However, we differ in terms of our training objective. In addition to the log-likelihood objective, we also include syntactic inconsistency loss (defined in Section 2.3) which quantifies the hard constraint (spans implied by syntactic parse) violations in our training objective. In other words, while training our model, we enforce the outputs of our system to agree with the spans implied by the syntactic parse of the sentence as much as possible. In summary, our contributions to low-resource SRL are:

1. A novel formulation which leverages syntactic parse trees for SRL by introducing them as hard constraints while training the model.
2. Experiments with varying amounts of SRL-unlabeled data that point towards semi-supervised learning for low-resource SRL by leveraging the fact that syntactic inconsistency loss does not require labels.

2 Proposed Approach

We build upon an existing deep-learning approach to SRL (He et al., 2017). First we revisit definitions introduced by (He et al., 2017) and then discuss about our formulation.

2.1 Task definition

Given a sentence-predicate pair $(\mathbf{x}, v)$, SRL task is defined as predicting a sequence of tags $\mathbf{y}$, where each y_i belongs to a set of BIO tags (Ω). So, for an argument span with semantic role ARG$_i$, B-ARG$_i$ tag indicates that the corresponding token marks the beginning of the argument span and I-ARG$_i$ tag indicates that the corresponding token is inside of the argument span and O tag indicates that token is outside of all argument spans. Let $n = |\mathbf{x}| = |\mathbf{y}|$ be the length of the sentence. Further, let *srl-spans*$(\mathbf{y})$ denote the set of all argument spans in the SRL tagging $\mathbf{y}$. Similarly, *parse-spans*$(\mathbf{x})$ denotes the set of all unlabeled parse constituents for the given sentence $\mathbf{x}$. Lastly, SRL-labeled/unlabeled data refers to sentence-predicate pairs with/without gold SRL tags.

2.2 State-of-the-Art (SOTA) Model

He et al. proposed a deep bi-directional LSTM to learn a locally decomposed scoring function conditioned on the entire input sentence- $\sum_{i=1}^{n} \log p(y_i|\mathbf{x})$. To learn the parameters of a network, the conditional negative log-likelihood $\mathcal{L}(\mathbf{w})$ of a sample of training data $\mathcal{T} = \{\mathbf{x}^{(t)}, \mathbf{y}^{(t)}\}_{\forall t}$ is minimized, where $\mathcal{L}(\mathbf{w})$ is

$$\mathcal{L}(\mathbf{w}) = -\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \sum_{i=1}^{|\mathbf{y}|} \log p(y_i|\mathbf{x};\mathbf{w}). \quad (1)$$

Since Eq.(1) does not model any dependencies between the output tags, the predicted output tags tend to be structurally inconsistent. To alleviate this problem, (He et al., 2017) searches for the best $\hat{\mathbf{y}}$ over the space of all possibilities (Ω^n) using the scoring function $f(x, y)$, which incorporates log probability and structural penalty terms. The details of scoring function is on Appendix Eq.(7).

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}'\in\Omega^n} \mathbf{f}(\mathbf{x}, \mathbf{y}') \quad (2)$$

2.3 Structural Constraints

There are different types of structural constraints: BIO, SRL and syntactic constraints. BIO constraints define valid BIO transitions for sequence tagging. For example, B-ARG0 cannot be followed by I-ARG1. SRL constraints define rules on the role level and has three particular constraints: unique core roles (U), continuation roles (C) and reference roles (R) (Punyakanok et al., 2008). Lastly, syntactic constraints state that *srl-spans*$(\mathbf{y})$ have to be subset of *parse-spans*$(\mathbf{x})$.

(He et al., 2017) use BIO and syntactic constraints at decoding time by solving Eq.(2) where $f(x, y)$ incorporates those constraints and report that SRL constraints do not show significant improvements over the ensemble model. In particular, by using syntactic constraints, (He et al., 2017) achieves up to +2 F1 score on CoNLL-2005 dataset via A* decoding.

Improvements of SRL system via use of syntactic constraints is consistent with other observations (Punyakanok et al., 2008). However, all previous works enforce syntactic consistency only during decoding step. We propose that enforcing syntactic consistency during training time would also be beneficial and show the efficacy experimentally on Section 3.3.

Enforcing Syntactic Consistency To quantify syntactic inconsistency, we define *disagreeing-spans*$(\mathbf{x}, \mathbf{y}) = \{span_i \in \text{srl-spans}(\mathbf{y}) \mid span_i \notin \text{parse-spans}(\mathbf{x})\}$. Further, we define *disagreement* *rate*, $d(\mathbf{x},\mathbf{y}) \in [0,1]$, and *syntactic inconsistency score*, $s(\mathbf{x},\mathbf{y}) \in [-1,1]$, as follows:

$$d(\mathbf{x},\mathbf{y}) = \frac{|\text{disagreeing-spans}(\mathbf{x},\mathbf{y})|}{|\text{srl-spans}(\mathbf{y})|} \quad (3)$$

$$s(\mathbf{x},\mathbf{y}) = 2 \times d(\mathbf{x},\mathbf{y}) - 1 \quad (4)$$

Syntactic Inconsistency Loss (SI-Loss) For a given $(\mathbf{x}, v)$, let us consider $\hat{\mathbf{y}}^{(t)}$ to be the best possible tag sequence (according to Eq.(2) during epoch t of model training. Ideally, if our model is syntax-aware, we would have $d(\mathbf{x},\hat{\mathbf{y}}^{(t)}) = 0$ or $r(\mathbf{x},\hat{\mathbf{y}}^{(t)}) = 1$. We define a loss component due to syntactic inconsistency (*SI-Loss*) as follows:

$$\text{SI-Loss} = s(\mathbf{x},\hat{\mathbf{y}}^{(t)}) \sum_{i=1}^{|\hat{\mathbf{y}}^{(t)}|} \log p(\hat{y}_i^{(t)}|\mathbf{x};\mathbf{w}^{(t)}) \quad (5)$$

During training, we want to minimize SI-Loss.

2.4 Training with Joint Objective

Based on Eq.(1), a supervised loss, and Eq.(5), the SI-Loss, we propose a joint training objective. For a given sentence-predicate pair $(\mathbf{x}, v)$ and SRL tags $\mathbf{y}$, our joint training objective (at epoch t) is defined as:

$$\text{Joint loss} = -\alpha_1 \sum_{i=1}^{|\mathbf{y}|} \log p(y_i|\mathbf{x};\mathbf{w})$$
$$+\alpha_2\, r(\mathbf{x},\hat{\mathbf{y}}^{(t)}) \sum_{i=1}^{|\hat{\mathbf{y}}^{(t)}|} \log p(\hat{y}_i^{(t)}|\mathbf{x};\mathbf{w}^{(t)}) \quad (6)$$

Here, α_1 and α_2 are weights (hyper-parameters) for different loss components and are tuned using a development set. During training, we minimize joint loss - i.e., negative log-likelihood (or cross-entropy loss) and syntactic inconsistency loss.

2.5 Semi-supervised learning formulation

In low resource scenarios, we have limited labeled data and larger amounts of unlabeled data. The obvious question is how to leverage large amounts of unlabeled data for training accurate models. In context of SRL, we propose to leverage SRL-unlabeled data in terms of parse trees.

Observing Eq.(5), one can notice that our formulation of SI-Loss is only dependent upon model's predicted tag sequence $\hat{\mathbf{y}}^{(t)}$ at a particular time point t during training and the given sentence and it does not depend upon gold SRL tags. We leverage this fact in our SSL formulation to compute SI-loss from SRL-unlabeled sentences.

Model/ Legend	Test F1	Average disagreement rate (%)
B100	84.40	14.69
B10	**78.56**	**17.01**
B1	**67.28**	**21.17**
J100	84.75 (+0.35)	14.48 (−1.43%)
J10	79.09 (+0.53)	16.25 (−4.47%)
J1	68.02 (+0.74)	20.49 (−3.21%)

Table 1: Comparison of baseline models (**B**) with the models trained with joint objective (**J**).

Let *sup-s* be a batch of SRL-labeled sentences and *usup-s* be a batch SRL-unlabeled sentences only with parse information. In SSL setup, we propose to train our model with joint objective where *sup-s* only contributes to supervised loss Eq.(1) and *unsup-s* contributes in terms of syntactic inconsistency objective Eq.(5) and combine them according to Eq.(6) to train them with joint loss.

3 Experiments

3.1 Dataset

We evaluate our model's performance on span-based SRL dataset from CoNLL-2012 shared task (Pradhan et al., 2013). This dataset contains gold predicates as part of the input sentence and also gold parse information corresponding to each sentence which we use for defining hard constraints for SRL task. We use standard train/development/test split containing 278K/38.3K/25.6K sentences. Further, there is approx. 10% disagreement between gold SRL-spans and gold parse-spans (we term these as noisy syntactic constraints). During training, we do not preprocess data to handle these noisy constraints but for the analysis related to enforcing syntactic constraints during inference, we study both cases: with and without noisy constraints. [1]

3.2 Model configurations

For the SOTA system proposed in (Peters et al., 2018), we use code from Allen AI[2] to implement our approach. We follow their initialization and training configurations. Let **B**$\underline{X}$**, J**$\underline{X}$ denote model trained with $\underline{X}$% of the SRL-labeled data with cross-entropy and joint training objective, re-

---

[1] We preprocessed data for inference by simply deleting syntactic parse trees for the sentences where we have disagreement and perform standard Viterbi decoding for those sentences and note that this preprocessing scheme was never used during training.

[2] https://github.com/allenai/allennlp

Base Model/ Legend	Test F1	Average disagreement rate (%)
B10	78.56	17.01
B10-SI1x	78.84 (+0.28)	16.17 (−4.94%)
B10-SI5x	78.67 (+0.11)	16.47 (−3.17%)
B10-SI10x	78.76 (+0.20)	16.09 (−5.4%)
B1	67.28	21.17
B1-SI1x	67.67 (+0.39)	20.14 (−4.87%)
B1-SI5x	67.74 (+0.46)	19.93 (−5.86%)
B1-SI10x	67.71 (+0.43)	20.16 (−4.77%)

Table 2: Training with **SI-loss** for varying sizes of SRL-unlabeled data on top of the pre-trained baseline models (B1, B10 on Table 1).

Base Model/ Legend	Test F1	Average disagreement rate (%)
B10	78.56	17.01
B10-further	78.86 (+0.3)	16.25 (−2.06%)
B10-J1x	79.23 (+0.67)	16.03 (−5.76%)
B10-J5x	79.25 (+0.69)	16.01 (−5.88%)
B10-J10x	**79.34 (+0.78)**	**15.88 (-6.64%)**
B1	67.28	21.17
B1-further	67.76 (+0.48)	20.75 (−1.98%)
B1-J1x	68.45 (+1.17)	19.57 (−7.56%)
B1-J5x	**68.86 (+1.58)**	**19.38 (-8.46%)**
B1-J10x	68.61 (+1.33)	19.29 (−8.88%)

Table 3: Training with joint objective (**J**) on top of the baseline models (**B1**, **B10** from Table 1), with the same SRL-labeled data used to train the baseline models and with varying sizes of SRL-unlabeled data.

Decoding	B10	B10-J10x	B1	B1-J5x
Viterbi	78.56	79.34	67.28	68.86
	Noisy syntactic constraints			
A*	72.95 (-5.61)	73.57 (-5.77)	63.77 (-3.51)	64.73 (-4.13)
Gradient-based	79.7 (+1.41)	80.21 (+0.87)	69.85 (+2.57)	**70.95** **(+2.1)**
	Noise-free syntactic constraints			
A*	78.87 (+0.31)	79.51 (+0.17)	67.97 (+0.69)	68.97 (+0.11)
Gradient-based	80.18 (+1.62)	**80.66** **(+1.32)**	69.97 (+2.69)	70.94 (+2.08)

Table 4: Comparison of different decoding techniques: Viterbi, A* (He et al., 2017) and gradient based inference (Lee et al., 2017) with noisy and noise-free syntactic constraints[1]. Note that the (+/-) F1 are reported w.r.t Viterbi decoding on the same column.

spectively. B$\underline{X}$-SI$\underline{U}$x and B$\underline{X}$-J$\underline{U}$x denote model trained with SI-loss and Joint loss, respectively, on the pre-trained B$\underline{X}$ model where $\underline{X} \times \underline{U}$ amount of SRL-unlabeled data were used for further training. To satisfy BIO constraints, we run Viterbi decoding by default for inference.

3.3 Results

We are interested in answering following questions. (Q1) how well does the baseline model produce syntactically consistent outputs, (Q2) does our approach actually enforce syntactic constraints, (Q3) does our approach enforce syntactic constraints without compromising the quality of the system, (Q4) how well does our SSL formulation perform, especially in low-resource scenarios, and lastly (Q5) what is the difference in using the syntactic constraints in training time compared to using it at decoding time. To answer (Q1-2) favorably we report average disagreement rate computed over test split. To answer (Q3-5), we report overall F1-scores on CoNLL-2012 test set (using standard evaluation script). For experiments using SRL-unlabeled data, we report average results after running multiple experiments with different random samples of it.

Does training with joint objective help? We trained 3 models with random 1%, 10% and whole 100% of the training set with joint objective (α_1 = α_2 = 0.5). For comparison, we trained 3 SOTA models with the same training sets. All models were trained for max 150 epochs and with a patience of 20 epochs. Table 1 reports the results of this experiment. We see that models trained with joint objective (JX) improve over baseline models (BX), both in terms of F1 and average disagreement rate. These improvements provide evidence for answering (Q1-3) favorably. Further, gains are more in low resource scenarios because by training models jointly to satisfy syntactic constraints helps in better generalization when trained with limited SRL corpora.

Does SSL based training work for low-resource scenarios? To enforce syntactic constraints via SI-loss on SRL-unlabeled data, we further train pre-trained model with two objectives in SSL set up: (a) SI-loss (Table 2) and (b) joint objective (Table 3)

For experiment (a), we use square loss, $\beta \| W - W_{\text{pre-train}} \|^2$ regularizer to keep the model W close to the pre-trained model $W_{\text{pre-train}}$ to avoid catastrophic forgetting (β set to 0.005). We optimize with SGD with learning rate of 0.01, $\alpha_2 = 1.0$, patience of 10 epochs. We see that with SI-loss improvements are significant in terms of average disagreement rate as compared to F1.

For experiment (b), we train B1 and B10 with joint objective in SSL set-up (as discussed in Section 2.5). We use SGD with learning rate of 0.05, $\alpha_1 = \alpha_2 = 1.0$ and patience of 10 epochs. We report +1.58 F1 and +0.78 F1 improvement over

B1 and B10, trained with 5% and 100% SRL-unlabeled data, respectively. Note that we cannot achieve these improvements with simply fine-tunning B$\underline{X}$ with supervised loss, as seen with B$\underline{X}$-further on Table 3. This provides evidence to answer (Q4) favorably. In general, the performance gains increase as the size of the SRL-unlabeled data increases.

Is it better to enforce syntactic consistency on decoding or on training time? To answer (Q5), we conducted three experiments: using syntactic constraints on (a) inference only, i.e. structured prediction, (b) training only, and (c) both training and inference steps. For the structured prediction, we consider A* decoding, as used in (He et al., 2017) and gradient-based inference (Lee et al., 2017), which optimizes loss function similar to SI-loss on Eq.(5) per example basis. If neither A* decoding nor gradient-based inference is used, we use Viterbi algorithm to enforce BIO constraints. The performance is the best (bold on Table 4) when syntactic consistency is enforced both on training and inference steps, +3.67, +2.1 F1 score improvement over B1 and B10 respectively, and we conclude that the effort of enforcing syntactic consistency on inference time is complementary to the same effort on training time. However, note that the overall performance increases as the benefit from enforcing syntactic consistency with SSL is far greater compared to marginal decrement on structured prediction.

While syntactic constraints help both train and inference, injecting constraints on train time is far more robust compared to enforcing them on decoding time. The performance of the structured prediction drops rapidly when the noise in the parse information is introduced (x column of Table 4). On the other hand, SSL was trained on CoNLL2012 data where about 10% of the gold SRL-spans do not match with gold parse-spans and even when we increase noise level to 20% the performance drop was only around 0.1 F1 score.

4 Related Work

The traditional approaches for SRL (Pradhan et al., 2005; Koomen et al., 2005) constituted of cascaded system with four subtasks: pruning, argument identification, argument labeling, and inference. Recent approaches (Zhou and Xu, 2015; He et al., 2017) proposed end-to-end system for SRL using deep recurrent or bi-LSTM-based architecture with no syntactic inputs and have achieved SOTA results on English SRL. Lastly, (Peters et al., 2018) proposed ELMo, a deep contextualized word representations, and improved the SOTA English SRL by 3.2 F1-points.

Even on the end-to-end learning, inference still remains as a separate subtask and would be formalized as a constrained optimization problem. To solve this problem ILP (Punyakanok et al., 2008), *A* algorithm* (He et al., 2017) and gradient-based inference (Lee et al., 2017) were employed. Further, all of these works leveraged syntactic parse during inference and was never used during training unless used as a cascaded system.

To the best of our knowledge, this work is the first attempt towards SSL span-based SRL model. Nonetheless, there were few efforts in SSL in dependency-based SRL systems (Fürstenau and Lapata, 2009; Deschacht and Moens, 2009; Croce et al., 2010). (Fürstenau and Lapata, 2009) proposed to augment the dataset by finding similar unlabeled sentences to already labeled set and annotate accordingly. While interesting, the similar augmentation technique is harder to apply to span-based SRL as one requires to annotate the whole span. (Deschacht and Moens, 2009; Croce et al., 2010) proposed to leverage the relation between words by learning latent word distribution over the context, i.e. language model. Our paper also incorporates this idea by using ELMo as it is trained via language model objective.

5 Conclusion and Future Work

We presented a SI-loss to enforce SRL systems to produce syntactically consistent outputs. Further, leveraging the fact that SI-loss does not require labeled data, we proposed a SSL formulation with joint objective constituting of SI-loss and supervised loss together. We show the efficacy of the proposed approach on low resource settings, +1.58, +0.78 F1 on 1%, 10% SRL-labeled data respectively, via further training on top of pre-trained SOTA model. We further show the structured prediction can be used as a complimentary tool and show performance gain of +3.67, +2.1 F1 over pre-trained model on 1%, 10% SRL-labeled data, respectively. Semi-supervised training from the scratch and examination of semi-supervised setting on large dataset remains as part of the future work.

References

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2013. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, pages 65–69.

Danilo Croce, Cristina Giannone, Paolo Annesi, and Roberto Basili. 2010. Towards open-domain semantic role labeling. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 237–246, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koen Deschacht and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 21–29, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hagen Fürstenau and Mirella Lapata. 2009. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 220–228, Stroudsburg, PA, USA. Association for Computational Linguistics.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 473–483.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*. European Language Resources Association (ELRA).

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 181–184. Association for Computational Linguistics.

Jay Yoon Lee, Michael Wick, Sanket Vaibhav Mehta, Jean-Baptiste Tristan, and Jaime Carbonell. 2017. Gradient-based inference for networks with output constraints. *arXiv preprint arXiv:1707.08608v2*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 217–220. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1127–1137.

Identifying Domain Adjacent Instances for Semantic Parsers

James Ferguson
University of Washington
jfferg@cs.washington.edu

Janara Christensen
Google Inc.

Edward Li
Google Inc

Edgar Gonzàlez
Google Inc

{jchristensen,gavinbelson,edgargip}@google.com

Abstract

When the semantics of a sentence are not representable in a semantic parser's output schema, parsing will inevitably fail. Detection of these instances is commonly treated as an out-of-domain classification problem. However, there is also a more subtle scenario in which the test data is drawn from the same domain. In addition to formalizing this problem of *domain-adjacency*, we present a comparison of various baselines that could be used to solve it. We also propose a new simple sentence representation that emphasizes words which are unexpected. This approach improves the performance of a downstream semantic parser run on in-domain and domain-adjacent instances.

Air Travel Domain	
Example In-Domain Predicates	
buyTicket	Buy ticket LGA to SFO on 3/12
flightStatus	What's the status of my SF flight?
switchFlight	Change it to the 8am SFO flight
cancelFlight	Cancel my flight to SFO
awardTravel	I want to fly to SFO with miles
Example Domain-Adjacent Predicates	
changeSeat	Change my seat to 23A
milesUpgrade	Upgrade my flight with my miles
arrivalGate	Gate that my SFO flight arrives at
mileageStatus	What's my miles status
Example Out-of-Domain Predicates	
transferMoney	Transfer $200 to checking
addTimer	Add a timer for 3 minutes
restaurantSearch	Thai restaurants in SF
scheduleMeeting	Set up a 9am meeting with Amy

Figure 1: In this example, an **air travel** semantic parser is trained on data containing in-domain predicates. A test instance cannot be parsed correctly if it contains any domain-adjacent or out-of-domain predicates.

1 Introduction

Semantic parsers map text to logical forms, which can then be used by downstream components to fulfill an action. Consider, for example, a system for booking air travel, in which a user provides natural language input, and a downstream subsystem is able to make or cancel flight reservations. Users of the system typically have a general understanding of its purpose, so the input will revolve around the correct topic of **air travel**. However, they are unlikely to know the limits of the system's functionality, and may provide inputs for which the expected action is beyond its capabilities, such as asking to change seats on a flight reservation. Because the logical schema is designed with fulfillment in mind, no logical form can capture the semantics of these sentences, making it impossible for the parser to generate a correct parse. Any output the parser generates will cause unintended actions to be executed downstream. For example, asking to change seats might be misparsed and executed as changing flights. Instead, the parser should identify that this input is beyond its scope so the condition can be handled.[1] In this paper, we formalize this pervasive problem, which we call *domain-adjacent instance identification*.

While this task is similar to that of identifying *out-of-domain* input instances (e.g., **banking** with respect to **air travel**), it is much more subtle — the instances come from roughly the same domain as the parser's training examples, and thus use very similar language. Domain adjacency is a property with respect to the parser's output schema, independent of the data used to train it.

In this paper, we formalize this task, and propose a simple approach for representing sentences in which words are weighted by how likely they are to differentiate between in-domain and domain-adjacent instances. Note that while this approach can also be applied to out-of-domain

---

[1] In the final page of the paper, we suggest a few immediate downstream system behaviors when a domain-adjacent instance is identified, but others have investigated the related problem of teaching the system new behavior (Azaria et al., 2016).

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4964–4969
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

instances, in this paper we are interested in its performance on domain-adjacent instances. We describe an evaluation framework for this new task and, finally, evaluate our proposed method against a set of baselines, comparing performance on the domain-adjacent classification problem and a downstream semantic parsing task.

2 Problem Setting

A semantic parser can be seen as a function φ that maps sentences x in a natural language $\mathcal{L}$ to logical forms $y \in \mathcal{Y}$. Assuming the existence of an oracle parser $\hat{\varphi}$, the problem we propose in this paper is that of determining, for a given test instance x, whether it belongs to the *domain* Φ of $\hat{\varphi}$, i.e., if its semantics can be encoded in the schema $\mathcal{Y}$.

In real-world usage, the input sentences x will be generated by a human user, who associates the capabilities of the parser to a particular topic (e.g., **air travel**). Thus most of the $x \in \mathcal{L} \setminus \Phi$ will share topic with the $\hat{x} \in \Phi$. Because of the similarity between these x and $\hat{x}$, we call this task *identification of domain-adjacent instances*.

3 Approach

Our goal is to identify input instances whose semantics are not representable in the parser's output schema, and we assume only an in-domain dataset is available at training time. Our approach is based on determining similarity to these training instances. We split the task in two parts: 1) encode the sentences to a compact representation that preserves the needed information, and 2) given these representations, identify which sentences are so dissimilar that they are unlikely to be parseable with any schema that covers the training set.

3.1 Sentence Representation

Among recent work in distributional semantics, averaging the word vectors to represent a sentence (Wieting et al., 2016; Adi et al., 2017) has proven to be a simple and robust approach. However, we have an intuition that words which are *unexpected* in their context given the training data may be a strong signal that an instance is domain-adjacent. To incorporate this signal, we propose a weighted average, in which the weight corresponds to how unexpected the word is in its context. For example, given in-domain predicates from Figure 1, in the domain-adjacent sentence "Upgrade my flight to SFO with my miles", *upgrade* should receive a much higher weight than *flight* or *SFO*.

Our weighting scheme is as follows: We use the cosine distance between the expected ($\bar{v}_i$) and the actual ($\hat{v}_i$) domain-specific word embedding at a given position (i) in a sentence to compute its weight: $w_i = 1 - cos(\bar{v}_i, \hat{v}_i)$. The expected word embedding is computed using the context embeddings, $\bar{v}_i = \sum_{j=i-c, j\neq i}^{i+c} \hat{v}_j$, where $\hat{v}_j$ is a domain-specific word embedding, in a window of size c around position i. Intuitively, w_i represents how surprising the word is in the context.

Since our training set is too small to directly learn domain-specific embeddings, we learn a mapping from general pre-trained embeddings. We train a continuous bag-of-words model (Mikolov et al., 2013) in which we pass pre-trained embeddings (v_i) instead of 1-hot vectors, as input to the embedding layer. The layer thus learns a mapping from pre-trained to domain-specific embeddings ($\hat{v}_i$). We use this mapping to compute new embeddings for words that are missing from the training set. Only words that do not have pre-trained embeddings are ignored.

Finally, for a sentence with n words, we take the weighted average of the pre-trained embeddings of the words in the sentence, using the weights from above: $S = (\sum_{i=1}^{n} w_i v_i) \,/\, (\sum_{i=1}^{n} w_i)$.

This approach assigns high weight to words that differ significantly from what is expected based on the training data. By combining these weights with the pre-trained word embeddings, we allow the model to incorporate external information, improving generalization beyond the training set.

3.2 Domain-Adjacent Model

A number of techniques can be applied to predict whether a sentence is domain-adjacent from its continuous representation. Of the methods we tried, we found k-nearest neighbors (Angiulli and Pizzuti, 2002) to perform best: to classify a sentence, we calculate the average cosine distance between its embedding and its k nearest neighbors in the training data, and label it domain-adjacent if this value is greater than some threshold. This simpler model relies more heavily on the external information brought in by pre-trained word embeddings, while more complex models seem to overfit to the training data.

Basketball	*numGamesPlayed*
Blocks	*length*
Calendar	*startTime*
Housing	*size*
Publications	*venue*
Recipes	*preparationTime*
Restaurants	*starRating*
Social	*educationStartDate*, *employmentEndDate*

Table 1: Predicates excluded from training and considered domain-adjacent. Domains have 5-20 predicates.

4 Evaluation

In this section, we introduce an evaluation framework for this new task. We consider training and test sets from a single domain, with only the latter containing domain-adjacent instances. Test instances are classified individually, and we measure performance on in-domain/domain-adjacent classification and semantic parsing.

4.1 Dataset and Semantic Parser

We simulate this setting by adapting the OVERNIGHT dataset (Wang et al., 2015). This dataset is composed of queries drawn from eight domains, each having a set of seven to eighteen distinct semantic predicates. Queries consist of a crowd-sourced textual sentence and its corresponding logical form containing one or more of these domain-specific semantic predicates.

For each domain, we select a set of predicates to exclude from the logical schema (see Table 1), and remove all instances containing these predicates from the training set (since they are now domain-adjacent). We then train a domain-adjacent model and semantic parser on the remaining training data and attempt to identify the domain-adjacent examples in the test data. We use the train/test splits from Wang et al. (2015). In all experiments, we use the SEMPRE parser (Berant et al., 2013).

4.2 Baselines

Because this is a novel task, and results are not comparable to previous work, we report results from a variety of baseline systems. The first baseline, CONFIDENCE, identifies instances as domain-adjacent if the semantic parser's confidence in its predictions is below some threshold.

The remaining baselines follow the two-part approach from Section 3. AUTOENCODER is inspired by Ryu et al. (2017)'s work on identifying *out-of-domain* examples. For the sentence representation, this method uses a bi-LSTM with self-attention, trained to predict the semantic predicates, and concatenates the final hidden state from each direction as the sentence representation. An autoencoder is used as the domain-adjacent classifier.

The remaining methods use the nearest neighbor model discussed in Section 3.2. For sentence representations, we include baselines drawn from different neural approaches. In CBOW, we simply average the pre-trained word embeddings in the sentence. In CNN, we train a two-layer CNN with a final softmax layer to predict the semantic predicates for a sentence. We concatenate the mean pooling of each layer as the sentence representation. In LSTM, we use the same sentence representation as in AUTOENCODER, with the nearest neighbor domain-adjacent model. Finally, SURPRISE is the approach presented in Section 3.1.

4.3 Direct Evaluation

We first directly evaluate the identification of domain-adjacent instances: Table 2 reports the area under a receiver operating characteristic curve (AUC) for the considered models (Fawcett, 2006). SURPRISE generally performs the best on this evaluation; and, in general, the simpler models tend to perform better, suggesting that more complex approaches tune too much to the training data.

Qualitatively, for domains where the SURPRISE model performs better, it places higher weight on words we would consider important for distinguishing domain-adjacent sentences. For example in “show me recipes with longer preparation times than rice pudding” from **Recipes**, “longer” and “preparation” have the highest weights. In **Social**, there are two in-domain predicates (*employmentStartDate* and *educationEndDate*) which use very similar wording to those that are domain-adjacent, making it difficult to isolate surprising words. The weights in this domain seem to instead emphasize unusual wordings such as “soonest” in “employees with the soonest finish date”.

4.3.1 Ablation Analysis

In order to determine the contribution of each one of the components of SURPRISE, we performed an ablation analysis comparing the following modifications of the method: CBOW, as described above, using an unweighted average of pre-trained embeddings; FREQUENCY, using a

	Basketball	Blocks	Calendar	Housing	Publications	Recipes	Restaurants	Social
Autoencoder	0.801	0.479	0.766	0.781	0.874	0.722	0.581	0.774
Confidence	0.660	0.738	0.697	0.648	0.631	0.651	0.730	0.573
CBOW	0.743	0.782	0.662	0.910	0.884	0.670	0.911	0.675
CNN	**0.916**	0.654	**0.862**	0.792	0.908	0.505	0.840	**0.813**
LSTM	0.826	0.571	0.741	0.912	0.827	0.487	0.593	0.754
Surprise	0.755	**0.827**	0.817	**0.933**	**0.978**	**0.758**	**0.941**	0.545

Table 2: AUC for domain-adjacent instance identification, using kNN as the domain-adjacent model.

	Basketball	Blocks	Calendar	Housing	Publications	Recipes	Restaurants	Social
CBOW	0.743	0.782	0.662	0.910	0.884	0.670	0.911	**0.675**
Frequency	0.656	0.703	0.771	0.884	0.887	0.667	0.834	0.591
PreTrained	0.612	0.636	0.512	0.819	0.842	0.526	0.858	0.538
Surprise	**0.755**	**0.827**	**0.817**	**0.933**	**0.978**	**0.758**	**0.941**	0.545

Table 3: AUC for domain-adjacent instance identification, using ablated versions of Surprise with kNN.

weighted average of pre-trained embeddings, with weights based on inverse document frequency in the training set; PreTrained, using the surprise schema but with weights determined using pre-trained embeddings; and the full Suprise as presented above. Each approach adds one component (weighting, surprise-based weights, and domain-specific embeddings) with respect to the previous one.

The results of the experiment are shown in Table 3. We can see that Frequency performs slightly worse than CBOW and PreTrained performs even worse than that. We can conclude that the combination of the weighting schema and the tuned vectors is what makes Suprise effective.

4.4 Downstream Task Evaluation

We next evaluate how including the domain-adjacent predictions affects the performance of a semantic parser. In a real setting, when the semantic parser is presented with domain-adjacent input that is beyond its scope, the correct behavior is to label it as such so that it can be handled properly by downstream components. To simulate this behavior, we set the gold parse for domain-adjacent instances to be an empty parse, and automatically assign an empty parse to any instance that is identified as domain-adjacent. We report accuracy of the semantic parser with 20% domain-adjacent test data. We include two additional models: NoFilter, in which nothing is labeled domain-adjacent, and Oracle, in which all the domain-adjacent instances are correctly identified. For each baseline requiring a threshold, we set it such that 3% of the instances in the dev set would be marked as domain-adjacent (intuitively, this represents the error-tolerance of the system).

Table 4 shows the results for this experiment. In general, the relative performance is similar to that in the direct evaluation (e.g. Surprise tends to do well on most domains, but performs poorly on Basketball and Social in particular). However, in this evaluation, misclassifying an instance as domain-adjacent if the semantic parser would have accurately parsed it is worse than misclassifying the instance if the semantic parser could not have accurately parsed it. For example, in Social we can thus infer that Surprise is marking some instances as domain-adjacent that would otherwise be accurately parsed as the performance there is actually worse than for NoFilter.

5 Related Work

Domain-adjacency identification is a new task, but relatively little effort has been devoted to even the related task of identifying out-of-domain instances (i.e., from completely separate domains) for semantic parsers. Hakkani-Tur et al. (2015) approached the problem by clustering sentences based on shared subgraphs in their general semantic parses; Ryu et al. (2017) classify sentences with autoencoder reconstruction error.

Prior distributional semantics work to create compact sentential representations generated specific embeddings for downstream tasks (Kalchbrenner et al., 2014; Kim, 2014; Socher et al., 2013). Recently, work has focused on domain-

	Basketball	Blocks	Calendar	Housing	Publications	Recipes	Restaurants	Social
NOFILTER	0.358	0.294	0.617	0.461	0.511	0.570	0.626	0.355
ORACLE	0.558	0.494	0.817	0.661	0.711	0.770	0.826	0.555
AUTOENCODER	0.413	0.268	0.581	0.447	0.463	0.530	0.543	0.417
CONFIDENCE	0.389	0.306	0.644	0.472	0.525	0.568	0.665	0.360
CBOW	0.344	0.295	0.634	0.515	0.621	**0.575**	0.722	0.358
CNN	**0.452**	0.324	0.674	0.488	0.573	0.570	0.605	**0.446**
LSTM	0.385	0.314	0.622	0.495	0.581	0.547	0.612	0.363
SURPRISE	0.356	**0.371**	**0.679**	**0.570**	**0.668**	0.554	**0.764**	0.345

Table 4: Accuracy for a semantic parser evaluated on a test set in which 20% is domain adjacent.

independent embeddings, learned without downstream task supervision. Kiros et al. (2015), Hill et al. (2016), and Kenter et al. (2016) learn representations by predicting the surrounding sentences. Wieting et al. (2016) use paraphrases as supervision. Mu et al. (2017) represent sentences by the low-rank subspace spanned by the embeddings of the words in them; Arora et al. (2017) use a weighted average of word embeddings, with their projection onto the first principal component across all sentences in the corpus removed.

Another relatively sparse area of related work is handling the domain-adjacent instances once they have been identified. The simplest thing to do is to return a generic error. For user-facing applications, one such message might state that the system can't handle that specific query. Azaria et al. (2016) approach this problem by having the user break down the domain-adjacent instance into a sequence of simpler textual instructions and then attempting to map those to known logical forms.

6 Conclusion

Identifying domain-adjacent instances is a practical issue that can improve downstream semantic parsing precision, and thus provide a smoother and more reliable user experience. In this paper, we formalize this task, and introduce a novel sentence embedding approach which outperforms baselines. Future work includes exploring alternative ways of incorporating information outside of the given training set and experimenting with various combinations of semantic parsers and upstream domain-adjacency models. Another area of future research is how the underlying system should recover when domain-adjacent instances are detected.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proceedings of ICLR 2017*.

Fabrizio Angiulli and Clara Pizzuti. 2002. Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 15–27. Springer.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of ICLR 2017*.

Amos Azaria, Jayant Krishnamurthy, and Tom M. Mitchell. 2016. Instructable intelligent personal agent. In *Proceedings of AAAI 2016*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP 2013*.

Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874.

Dilek Hakkani-Tur, Yun-Cheng Ju, Geoff Zweig, and Gokhan Tur. 2015. Clustering novel intents in a conversational interaction system with semantic parsing. In *Proceedings of INTERSPEECH 2015*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of NAACL-HLT 2016*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL 2014*.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese CBOW: Optimizing word embeddings for sentence representations. In *Proceedings of ACL 2016*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP 2014*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of NIPS 2015*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR 2013*.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. Representing sentences as low-rank subspaces. In *Proceedings of ACL 2017*.

Seonghan Ryu, Seokhwan Kim, Junhwi Choi, Hwanjo Yu, and Gary Geunbae Lee. 2017. Neural sentence embedding using only in-domain sentences for out-of-domain sentence detection in dialog systems. *Pattern Recognition Letters*, 88(C):26–32.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP 2013*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of ACL 2015*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proceedings of ICLR 2016*.

Mapping natural language commands to web elements

Panupong Pasupat **Tian-Shun Jiang** **Evan Liu** **Kelvin Guu** **Percy Liang**
Stanford University
`{ppasupat,jiangts,evanliu,kguu,pliang}@cs.stanford.edu`

Abstract

The web provides a rich, open-domain environment with textual, structural, and spatial properties. We propose a new task for grounding language in this environment: given a natural language command (e.g., "click on the second article"), choose the correct element on the web page (e.g., a hyperlink or text box). We collected a dataset of over 50,000 commands that capture various phenomena such as functional references (e.g. "find who made this site"), relational reasoning (e.g. "article by john"), and visual reasoning (e.g. "top-most article"). We also implemented and analyzed three baseline models that capture different phenomena present in the dataset.

1 Introduction

Web pages are complex documents containing both structured properties (e.g., the internal tree representation) and unstructured properties (e.g., text and images). Due to their diversity in content and design, web pages provide a rich environment for natural language grounding tasks.

In particular, we consider the task of mapping natural language commands to web page elements (e.g., links, buttons, and form inputs), as illustrated in Figure 1. While some commands refer to an element's text directly, many others require more complex reasoning with the various aspects of web pages: the text, attributes, styles, structural data from the document object model (DOM), and spatial data from the rendered web page.

Our task is inspired by the semantic parsing literature, which aims to map natural language utterances into actions such as database queries and object manipulation (Zelle and Mooney, 1996; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Berant et al., 2013; Misra et al., 2015; Andreas and Klein, 2015). While these actions usually act on an environment with a fixed and known schema, web pages contain a larger variety of structures, making the task more open-ended. At the same time, our task can be viewed as a reference game (Golland et al., 2010; Smith et al., 2013; Andreas and Klein, 2016), where the system has to select an object given a natural language reference. The diversity of attributes in web page elements, along with the need to use context to interpret elements, makes web pages particularly interesting.

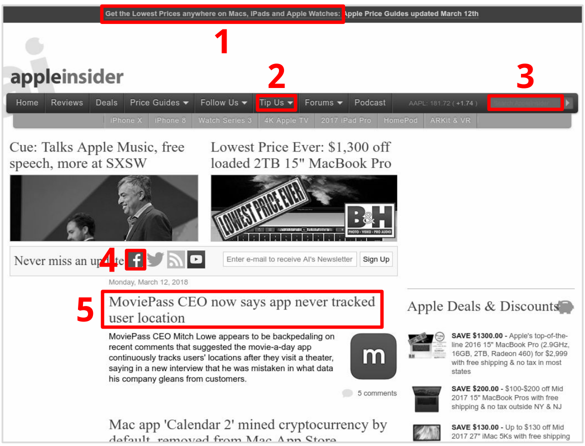

Cue: Talks Apple Music, free speech, more at SXSW

Lowest Price Ever: $1,300 off loaded 2TB 15" MacBook Pro

Monday, March 12, 2018

MoviePass CEO now says app never tracked user location

MoviePass CEO Mitch Lowe appears to be backpedaling on recent comments that suggested the movie-a-day app continuously tracks users' locations after they visit a theater, saying in a new interview that he was mistaken in what data his company gleans from customers.

Mac app 'Calendar 2' mined cryptocurrency by default, removed from Mac App Store

Apple Deals & Discounts

SAVE $1300.00 - Apple's top-of-the-line 2016 15" MacBook Pro (2.9GHz, 16GB, 2TB, Radeon 460) for $2,999 with free shipping & no tax in most states

SAVE $200.00 - $100-$200 off Mid 2017 15" MacBook Pros with free shipping & no tax outside NY & NJ

SAVE $130.00 - Up to $130 off Mid 2017 27" iMac 5Ks with free shipping

1: click on apple deals	2: send them a tip
3: enter iphone 7 into search	4: follow on facebook
5: open most recent news update	

Figure 1: Examples of natural language commands on the web page `appleinsider.com`.

Identifying elements via natural language has several real-world applications. The main one is providing a voice interface for interacting with web pages, which is especially useful as an assistive technology for the visually impaired (Zajicek et al., 1998; Ashok et al., 2014). Another use case is browser automation: natural language commands are less brittle than CSS or XPath selectors (Hammoudi et al., 2016) and could generalize across different websites.

We collected a dataset of over 50,000 natural language commands. As seen in Figure 1, the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4970–4976
Brussels, Belgium, October 31 - November 4, 2018.

Phenomenon	Description	Example	Amount
substring match	The command contains only a substring of the element's text (after stemming).	"view internships with energy.gov" → "Careers & Internship" link	7.0 %
paraphrase	The command paraphrases the element's text.	"click sign in" → "Login" link	15.5 %
goal description	The command describes an action or asks a question.	"change language" → a clickable box with text "English"	18.0 %
summarization	The command summarizes the text in the element.	"go to the article about the bengals trade" → the article title link	1.5 %
element description	The command describes a property of the element.	"click blue button"	2.0 %
relational reasoning	The command requires reasoning with another element or its surrounding context.	"show cookies info" → "More Info" in the cookies warning bar, not in the news section	2.5 %
ordinal reasoning	The command uses an ordinal.	"click on the first article"	3.5 %
spatial reasoning	The command describes the element's position.	"click the three slashes at the top left of the page"	2.0 %
image target	The target is an image (no text).	"select the favorites button"	11.5 %
form input target	The target is an input (text box, check box, drop-down list, etc.).	"in the search bar, type testing"	6.5 %

Table 1: Phenomena present in the commands in the dataset. Each example can have multiple phenomena.

commands contain many phenomena, such as relational, visual, and functional reasoning, which we analyze in greater depth in Section 2.2. We also implemented three models for the task based on retrieval, element embedding, and text alignment. Our experimental analysis shows that functional references, relational references, and visual reasoning are important for correctly identifying elements from natural language commands.

2 Task

Given a web page w with elements $e_1, \dots, e_k$ and a command c, the task is to select the element $e \in \{e_1, \dots, e_k\}$ described by the command c. The training and test data contain (w, c, e) triples.

2.1 Dataset

We collected a dataset of 51,663 commands on 1,835 web pages. To collect the data, we first archived home pages of the top 10,000 websites[1] by rendering them in Google Chrome. After loading the dynamic content, we recorded the DOM trees and the geometry of each element, and stored the rendered web pages. We filtered for web pages in English that rendered correctly and did not contain inappropriate content. Then we asked crowdworkers to brainstorm different actions for each web page, requiring each action to reference exactly one element (of their choice) from the filtered list of interactive elements (which include visible links, inputs, and buttons). We encouraged the workers to avoid using the exact words of the elements by granting a bonus for each command that did not contain the exact wording of the selected element. Finally, we split the data into 70% training, 10% development, and 20% test data. Web pages in the three sets do not overlap.

The collected web pages have an average of 1,051 elements, while the commands are 4.1 tokens long on average.

2.2 Phenomena present in the commands

Apart from referring to the exact text of the element, commands can refer to elements in a variety of ways. We analyzed 200 examples from the training data and broke down the phenomena present in these commands (see Table 1).

Even when the command directly references the element's text, many other elements on the page also have word overlap with the command. On average, commands have word overlap with 5.9 leaf elements on the page (not counting stop words).

3 Models

3.1 Retrieval-based model

Many commands refer to the elements by their text contents. As such, we first consider a simple retrieval-based model that uses the command as a search query to retrieve the most relevant element based on its TF-IDF score.

Specifically, each element is represented as a bag-of-tokens computed by (1) tokenizing and stemming its text content, and (2) tokenizing the

---

[1] `https://majestic.com/reports/majestic-million`

attributes (id, class, placeholder, label, tooltip, aria-text, name, src, href) at punctuation marks and camel-case boundaries. When computing term frequencies, we downweight the attribute tokens from (2) by a factor of α. We use $\alpha = 3$ tuned on the development set for our experiments.

The document frequencies are computed over the web pages in the training dataset. If multiple elements have the same score, we heuristically pick the most prominent element, i.e., the one that appears earliest in the pre-order traversal of the DOM hierarchy.

3.2 Embedding-based model

A common method for matching two pieces of text is to embed them separately and then compute a score from the two embeddings (Kiros et al., 2015; Tai et al., 2015). For a command c and elements $e_1, \ldots, e_k$, we define the following conditional distribution over the elements:

$$p\left(e_i \mid c\right) \propto \exp\left[s(f(c), g(e_i))\right]$$

where s is a scoring function, $f(c)$ is the embedding of c, and $g(e_i)$ is the embedding of e_i, described below. The model is trained to maximize the log-likelihood of the correct element in the training data.

Command embedding. To compute $f(c)$, we embed each token of c into a fixed-dimensional vector and take an average[2] over the token embeddings. (The token embeddings are initialized with GloVe vectors.)

Element embedding. To compute $g(e)$, we embed the properties of e, concatenate the results, and then apply a linear layer to obtain a vector of the same length as $f(c)$. Figure 2 shows an example of the properties that the model receives. The properties include:

- *Text content.* We apply the command embedder f on the text content of e. As the text of most elements of interest (links, buttons, and inputs) are short, we find it sufficient to limit the text to the first 10 tokens to save memory.

- *Text attributes.* Several attributes (aria, title, tooltip, placeholder, label, name) usually contain natural language. We concatenate their values and then apply the command embedder f on the resulting string.

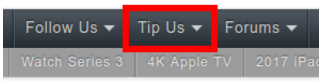

```
<a class="dd-head" id="tip-link"
 href="submit_story/">Tip Us</a>
```

Text content: tip us
String attributes: tip link dd head
Visual features: location = (0.53, 0.08)
visible = true

Figure 2: Example of properties used to compute the embedding $g(e)$ of the element e.

- *String attributes.* We tokenize other string attributes (tag, id, class) at punctuation marks and camel-case boundaries. Then we embed them with separate lookup tables and average the resulting vectors.

- *Visual features.* We form a vector consisting of the coordinates of the element's center (as fractions of the page width and height) and visibility (as a boolean).

Scoring function. To compute the score $s(f(c), g(e))$, we first let $\hat{f}(c)$ and $\hat{g}(e)$ be the results of normalizing the two embeddings to unit norm. Then we apply a linear layer on the concatenated vector $[\hat{f}(c); \hat{g}(e); \hat{f}(c) \circ \hat{g}(e)]$ (where $\circ$ denotes the element-wise product).

Incorporating spatial context. Context is critical in certain cases; for example, selecting a text box relies on knowing the neighboring label text, and selecting an article based on the author requires locating the author's name nearby. Identifying which related element should be considered based on the command is a challenging task.

We experiment with adding *spatial context* to the model. For each direction $d \in$ {top, bottom, left, right}, we use g to embed a neighboring element $n_d(e)$ directly adjacent to e in that direction. (If there are multiple such elements, sample one; if there is no such element, use a zero vector.) After normalizing the results to get $\hat{g}(n_d(e))$, we concatenate $\hat{g}(n_d(e))$ and $\hat{f}(c) \circ \hat{g}(n_d(e))$ to the linear layer input in the scoring function.

3.3 Alignment-based model

One downside of the embedding-based model is that the text tokens from c and e do not directly interact. Previous works on sentence matching usually employ either unidirectional or bidirectional attention to tackle this issue (Seo et al., 2016; Yin

---

[2] We tried applying LSTM but found no improvement.

et al., 2016; Xiong et al., 2017; Yu et al., 2018). We opt for a simple method based on a single alignment matrix similar to Hu et al. (2014) as described below.

Let $t(e)$ be the concatenation of e's text content and text attributes of e, trimmed to 10 tokens. We construct a matrix $A(c, e)$ where each entry $A_{ij}(c, e)$ is the dot product between the embeddings of the ith token of c and the jth token of $t(e)$. Then we apply two convolutional layers of size 3×3 on the matrix, apply a max-pooling layer of size 2×2, concatenate a tag embedding, and then apply a linear layer on the result to get a 10-dimensional vector $h(c, e)$.

We apply a final linear layer on $h(c, e)$ to compute a scalar score, and then train on the same objective function as the encoding-based model. To incorporate context, we simply concatenate the four vectors $h(c, n_d(e))$ of the neighbors $n_d(e)$ to the final linear layer input.

4 Experiments

We evaluate the models on *accuracy*, the fraction of examples that the model selects the correct element. We train the neural models using Adam (Kingma and Ba, 2014) with initial learning rate 10^{-3}, and apply early stopping based on the development set. The models can choose any element that is visible on the page at rendering time.

The experimental results are shown in Table 2. Both neural models significantly outperform the retrieval model.

4.1 Ablation analysis

To measure the importance of each type of information in web pages, we perform an ablation study where the model does not observe one of the following aspects of the elements: text contents, attributes, and spatial context.

Unsurprisingly, the results in Table 2 show that text contents are the most important input signal. However, attributes also play an important role in both the embedding and alignment models. Finally, while spatial context increases alignment model performance, the gain is very small, suggesting that incorporating appropriate contexts to the model is a challenging task due to the variety in the types of context, as well as the sparsity of the signals.

Model	Accuracy (%)
retrieval	36.55
embedding	56.05
no texts	23.62
no attributes	55.43
no spatial context	58.87
alignment	50.74
no texts	15.94
no attributes	48.51
no spatial context	50.66

Table 2: Accuracies of the models and their ablations.

Error Type	Embed	Align
Fail to match strings	26.8%	11.6%
Incorrectly match strings	3.8%	14.2%
Fail to understand paraphrases	8.9%	7.9%
Fail to understand descriptions	12.1%	17.4%
Fail to perform reasoning	15.9%	13.7 %
Select a less prominent element	19.8%	24.8%
Noisy annotation	12.7%	10.5%

Table 3: Error breakdowns of the embedding and alignment models on 100 examples. The embedding model handles implicit descriptions well, while the alignment model excels at string matching.

4.2 Error analysis

To get a better picture of how the models handle different phenomena, we analyze the predictions of the embedding-based and alignment-based models on 100 development examples where at least one model made an error. The errors, summarized in Table 3, are elaborated below:

Fail to match strings. Many commands simply specify the text content of the element (e.g., "click customised garages" → the link with text "Customised Garages, Canopies & Carports"). The encoding model, which encodes the whole command as a single vector, occasionally fails to select the element with partially matching texts. In contrast, the alignment model explicitly models text matching, and thus is better at this type of commands.

Incorrectly match strings. Due to its reliance on text matching, the alignment model struggles when many elements share substrings with the command (e.g., "shop for knitwear" when many elements contain the word "shop"), or when an element with a matching substring is not the correct target (e.g., "get the program" → the "Download" link, not the "Microsoft developer program" link).

Fail to understand descriptions. As seen in Table 1, many commands indirectly describe the elements using paraphrases, goal descriptions, or properties of the elements. The encoding model, which summarizes various properties of the elements as an embedding vector, is better at handling these commands than the alignment model, but still makes a few errors on harder examples (e.g., "please close this notice for me" → the "X" button with hidden text "Hide").

Fail to perform reasoning. For the most part, the models fail to handle relational, ordinal, or spatial reasoning. The most frequent error mode is when the element is a text box, and the command uses nearby label as the reference. While a few text boxes have semantic annotations which the model can use (e.g., tooltip or aria attributes), many web pages do not provide such annotations. To handle these cases, a model would have to identify the label of the text box based on logical or visual contexts.

Other errors. Apart from the annotation noise, occasionally multiple elements on the web page satisfy the given command (e.g., "log in" can match any "Sign In" button on the web page). In these cases, the annotation usually gives the most *prominent* element among the possible candidates. To provide a natural interface for users, the model should arguably learn to predict such prominent elements instead of more obscure ones.

5 Related work and discussion

Mapping natural language to actions. Previous work on semantic parsing learns to perform actions described by natural language utterances in various environments. Examples of such actions include API calls (Young et al., 2013; Su et al., 2017; Bordes and Weston, 2017), database queries (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2007; Berant et al., 2013; Yih et al., 2015), navigation (Artzi and Zettlemoyer, 2013; Janner et al., 2018), and object manipulation (Tellex et al., 2011; Andreas and Klein, 2015; Guu et al., 2017; Fried et al., 2018).

For web pages and graphical user interfaces, there are previous works on using natural language to perform computations on web tables (Pasupat and Liang, 2015; Zhong et al., 2017) and submit web forms (Shi et al., 2017). Our task is similar to previous works on interpreting instructions on user interfaces (Branavan et al., 2009, 2010; Liu et al., 2018). While their works focuses on learning from distant supervision, we consider shallower interactions but on a much broader domain.

Previous work also explores the reverse problem of generating natural language description of objects (Vinyals et al., 2014; Karpathy and Fei-Fei, 2015; Zarriaiß and Schlangen, 2017). We hope that our dataset could also be useful for exploring the reverse task of describing actions on web pages.

Reference games. In a reference game, the system has to select the correct object referenced by the given utterance (Frank and Goodman, 2012). Previous work on reference games focuses on a small number of objects with similar properties, and applies pragmatics to handle ambiguous utterance (Golland et al., 2010; Smith et al., 2013; Çelikyilmaz et al., 2014; Andreas and Klein, 2016; Yu et al., 2017). Our task can be viewed as a reference game with several challenges: higher number of objects, diverse object properties, and the need to interpret objects based on their contexts.

Interacting with web pages. Automated scripts are used to interact with web elements. While most scripts reference elements with logical selectors (e.g., CSS and XPath), there have been several alternatives such as images (Yeh et al., 2009) and simple natural language utterances (Soh, 2017). Some other interfaces for navigating web pages include keystrokes (Spalteholz et al., 2008), speech (Ashok et al., 2014), haptics (Yu et al., 2005), and eye gaze (Kumar et al., 2007).

6 Conclusion

We presented a new task of grounding natural language commands on open-ended and semi-structured web pages. With different methods of referencing elements, mixtures of textual and non-textual element attributes, and the need to properly incorporate context, our task offers a challenging environment for language understanding with great potential for real-world applications.

Our dataset and code are available at `https://github.com/stanfordnlp/phrasenode`. Reproducible experiments are available on the CodaLab platform at `https://worksheets.codalab.org/worksheets/0x0097f249cd944284a81af331093c3579/`.

Acknowledgments

This work was supported by NSF CAREER Award under No. IIS-1552635 and an Amazon Research Award.

References

J. Andreas and D. Klein. 2015. Alignment-based compositional semantics for instruction following. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Andreas and D. Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1182.

Y. Artzi and L. Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)*, 1:49–62.

V. Ashok, Y. Borodin, S. Stoyanchev, Y. Puzis, and I. V. Ramakrishnan. 2014. Wizard-of-Oz evaluation of speech-driven web browsing interface for people with vision impairments. In *Web for All Conference*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A. Bordes and J. Weston. 2017. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations (ICLR)*.

S. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 82–90.

S. Branavan, L. Zettlemoyer, and R. Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Association for Computational Linguistics (ACL)*, pages 1268–1277.

A. Çelikyilmaz, Z. Feizollahi, D. Z. Hakkani-Tür, and R. Sarikaya. 2014. Resolving referring expressions in conversational dialogs for natural user interfaces. In *Empirical Methods in Natural Language Processing (EMNLP)*.

D. L. Chen and R. J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 859–865.

M. Frank and N. D. Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336:998–998.

D. Fried, J. Andreas, and D. Klein. 2018. Unified pragmatic models for generating and following instructions. In *North American Association for Computational Linguistics (NAACL)*.

D. Golland, P. Liang, and D. Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 410–419.

K. Guu, P. Pasupat, E. Z. Liu, and P. Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Association for Computational Linguistics (ACL)*.

M. Hammoudi, G. Rothermel, and P. Tonella. 2016. Why do record/replay tests of web applications break? *IEEE International Conference on Software Testing, Verification and Validation*.

B. Hu, Z. Lu, H. Li, and Q. Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems (NIPS)*.

M. Janner, K. Narasimhan, and R. Barzilay. 2018. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics (TACL)*, 6.

A. Karpathy and L. Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3128–3137.

D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, R. Urtasun, A. Torralba, and S. Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems (NIPS)*.

M. Kumar, A. Paepcke, and T. Winograd. 2007. Eyepoint: practical pointing and selection using gaze and keyboard. In *Conference on Human Factors in Computing Systems (CHI)*.

E. Z. Liu, K. Guu, P. Pasupat, T. Shi, and P. Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*.

D. K. Misra, K. Tao, P. Liang, and A. Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Association for Computational Linguistics (ACL)*.

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*.

M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv*.

T. Shi, A. Karpathy, L. Fan, J. Hernandez, and P. Liang. 2017. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning (ICML)*.

N. J. Smith, N. D. Goodman, and M. C. Frank. 2013. Learning and using language via recursive pragmatic reasoning about other agents. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3039–3047.

K. Soh. 2017. TagUI: RPA / CLI tool for automating user interactions. `https://github.com/kelaberetiv/TagUI`.

L. Spalteholz, K. F. Li, N. Livingston, and F. Hamidi. 2008. Keysurf: a character controlled browser for people with physical disabilities. In *World Wide Web (WWW)*.

Y. Su, A. H. Awadallah, M. Khabsa, P. Pantel, M. Gamon, and M. J. Encarnación. 2017. Building natural language interfaces to web apis. In *Conference on Information and Knowledge Management (CIKM)*.

K. S. Tai, R. Socher, and C. D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Association for Computational Linguistics (ACL)*.

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2014. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*.

C. Xiong, V. Zhong, and R. Socher. 2017. Dynamic coattention networks for question answering. In *International Conference on Learning Representations (ICLR)*.

T. Yeh, T. Chang, and R. Miller. 2009. Sikuli: using GUI screenshots for search and automation. In *User Interface Software and Technology (UIST)*.

W. Yih, M. Chang, X. He, and J. Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Association for Computational Linguistics (ACL)*.

W. Yin, H. Schütze, B. Xiang, and B. Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics (TACL)*, 4.

S. Young, M. Gašić, B. Thomson, and J. D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. In *Proceedings of the IEEE*, 5, pages 1160–1179.

A. W. Yu, D. Dohan, M. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations (ICLR)*.

L. Yu, H. Tan, M. Bansal, and T. L. Berg. 2017. A joint speaker-listener-reinforcer model for referring expressions. In *Computer Vision and Pattern Recognition (CVPR)*.

W. Yu, R. Kuber, E. Murphy, P. Strain, and G. McAllister. 2005. A novel multimodal interface for improving visually impaired people's web accessibility. *Virtual Reality*, 9.

M. Zajicek, C. Powell, and C. Reeves. 1998. A web navigation tool for the blind. In *International ACM Conference on Assistive Technologies*.

S. Zarriaß and D. Schlangen. 2017. Obtaining referential word meanings from visual and distributional information: Experiments on object naming. In *Association for Computational Linguistics (ACL)*.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 678–687.

V. Zhong, C. Xiong, and R. Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Wronging a Right: Generating Better Errors to Improve Grammatical Error Detection

Sudhanshu Kasewa and **Pontus Stenetorp** and **Sebastian Riedel**
{sudhanshu.kasewa.16, p.stenetorp, s.riedel}@ucl.ac.uk
University College London

Abstract

Grammatical error correction, like other machine learning tasks, greatly benefits from large quantities of high quality training data, which is typically expensive to produce. While writing a program to automatically generate realistic grammatical errors would be difficult, one could learn the distribution of naturally-occurring errors and attempt to introduce them into other datasets. Initial work on inducing errors in this way using statistical machine translation has shown promise; we investigate cheaply constructing synthetic samples, given a small corpus of human-annotated data, using an off-the-rack attentive sequence-to-sequence model and a straight-forward post-processing procedure. Our approach yields error-filled artificial data that helps a vanilla bi-directional LSTM to outperform the previous state of the art at grammatical error detection, and a previously introduced model to gain further improvements of over 5% $F_{0.5}$ score. When attempting to determine if a given sentence is synthetic, a human annotator at best achieves 39.39 F_1 score, indicating that our model generates mostly human-like instances.

1 Introduction

There is an ever-growing number of people learning English as a second language; providing them with quick feedback to facilitate their learning is a crucial, labour-intensive endeavour. Part of this process is identifying and correcting grammatical errors, and several computational techniques have been developed to automate it (Rozovskaya and Roth, 2014; Junczys-Dowmunt and Grundkiewicz, 2016). For example, given an erroneous sentence *"I wanted to **goes** to the beach"*, the grammatical error correction task is to output the valid sentence *"I wanted to **go** to the beach"*. The task can be cast as a two-stage process, detection and correction, which can either be performed sequentially (Yannakoudakis et al., 2017), or jointly (Napoles and Callison-Burch, 2017).

Automated error correction performance is arguably still too low for practical consideration, perhaps limited by the amount of training data (Rei et al., 2017). High quality annotations are expensive to procure, and foreign language learners and commercial entities may feel uncomfortable granting access to their data. Instead, one could attempt to supplement existing manual annotations with synthetic instances. Such artificial samples are beneficial only when they share structure with the true distribution from which human errors are generated. Generative Adversarial Networks (Goodfellow et al., 2014) could be used for this purpose, but they are difficult to train, and require a large collection of sentences that are incorrect. One might attempt self-training (McClosky et al., 2006), where new instances are generated by applying a trained model to unannotated data, using high-confidence predictions as ground truth labels. However, in such a scheme, the expectation is that the unlabelled text already contains errors, which is not usually the case for most freely available text such as Wikipedia articles as they strive towards correctness.

In place of using machine translation (MT) to *correct* grammatical mistakes (Yuan and Felice, 2013; Junczys-Dowmunt and Grundkiewicz, 2014; Yuan and Briscoe, 2016), one might consider swapping the input and output streams, and instead learn to *induce errors* into error-free text, for the purpose of creating a synthetic training dataset (Felice and Yuan, 2014). Recently, Rei et al. (2017) used a statistical MT (SMT) system to induce errors into error-free text. Building on this work, and leveraging recent advances in neural MT (NMT), we used an off-the-shelf attentive sequence-to-sequence model (Britz et al., 2017), eliminating the need of specialised soft-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4977–4983
Brussels, Belgium, October 31 - November 4, 2018.

ware such as a phrase-table generator, decoder, and part-of-speech tagger. We created multiple synthetic datasets from in-domain and out-of-domain sources, and found that stochastic token sampling, and pruning redundant and low-likelihood sentences, were helpful in generating meaningful corruptions. Using the artificial samples thus generated, we improved upon detection results with simply a vanilla bi-directional LSTM (Hochreiter and Schmidhuber, 1997). Using a more powerful model, we established new state-of-the-art results, that improve on previously published $F_{0.5}$ scores by over 5%. Additionally, we confirm that our generated instances are human-like, as an annotator identifying generated sentences achieved a maximum F_1 score of 39.39.

2 Related work

In computer vision, images are blurred, rotated, or otherwise deformed inexpensively to create new training instances (Wang and Perez, 2017), because such manipulation does not significantly alter the image semantics. Similar coarse processes do not work in NLP since mutating even a single letter or a word can change a sentence's meaning, or render it nonsensical. Nonetheless, Vinyals et al. (2015) employed a kind of self-training where they use noisy predictions for unlabelled instances output by existing state-of-the-art parsers as ground-truth labels, and improved syntactic parsing performance. Sennrich et al. (2016) synthesised training instances by round-trip-translating a monolingual corpus with weaker versions of an NMT learner, and used them to improve the translation. Bouchard et al. (2016) developed an efficient algorithm to blend generated and true data for improving generalisation.

Grammar correction is a well-studied task in NLP, and early systems were rule-based pattern recognisers (Macdonald, 1983) and dictionary-based linguistic analysis engines (Richardson and Braden-Harder, 1988). Later systems used statistical approaches, addressing specific kinds of errors such as article insertion (Knight et al., 1994) and spelling correction (Golding and Roth, 1996). Most recently, architectural innovations in neural sequence labelling (Rei et al., 2016; Rei, 2017) raised error detection performance through improved ability to process unknown words and jointly learning a language model.

Early efforts for artificial error generation included generating specific types of errors, such as mass noun errors (Brockett et al., 2006) and article errors (Rozovskaya and Roth, 2010), and leveraging linguistic information to identify error patterns and transfer them onto grammatically correct text (Foster and Andersen, 2009; Yuan and Felice, 2013). Imamura et al. (2012) investigated methods to generate pseudo-erroneous sentences for error correction in Japanese. Recently, Rei et al. (2017) corrupted error-free text using SMT to create training instances for error detection.

3 Neural error generation

To learn to introduce errors, we use an off-the-shelf attentive sequence-to-sequence neural network (Bahdanau et al., 2014). Given an input sequence, the *encoder* generates context vectors for each token. Then, the *attention mechanism* and the *decoder* work in tandem to emit a distribution over the target vocabulary. At every decoder time-step, the encoder context vectors are scored by the attention mechanism, and a weighted sum is supplied to the decoder, along with its propagated internal state and last output symbol.

Corruption: Tokens from this distribution are sampled at every decoder time-step, either by $argmax$ (AM), which emits the most likely word, or by a stochastic alternative such as *temperature sampling* (TS) as $argmax$ cannot be relied on to generate rare words. A temperature parameter $\tau > 0$ sharpens or softens the distribution:

$$\tilde{p}_i = \boldsymbol{f}_\tau(p)_i = \frac{p_i^{\frac{1}{\tau}}}{\sum_j p_j^{\frac{1}{\tau}}}$$

where i are the components of the probability distribution corresponding to words in the vocabulary. As one interpolates τ from 0 to 1, the behaviour of $\tilde{p}$ transitions from *argmax* to p, controlling the diversity of the generated tokens.

The sentence generated by TS might be a low probability sequence from the joint conditional distribution $P(\mathbf{v}|\mathbf{u})$, where $\mathbf{u}$ is the input sentence and $\mathbf{v}$ is the output sentence. One way around this is to use *beam search* (BS), which checks the likelihood of every possible continuation of a sentence fragment, and maintains a list of the n best translations generated up to the current time-step. AM, TS, and BS are indicative of the trade-off between increasing levels of model flexibility at the cost of computation; we compare them to assess whether

Original	Corruption
She promised to turn over a new leaf. At the moment I'm in Spain.	She promissed to turn over a new leaf. During the moment I'm in Spain.

Table 1: Example sentences generated by our NMT pipeline.

Data augmentation strategy	Model	FCE (dev)	FCE	CoNLL1	CoNLL2
Rei et al. (2017) FCE$_{PAT}$ + EVP$_{PAT}$	SL	–	47.8	19.5	28.5
Rei et al. (2017) FCE$_{SMT}$ + EVP$_{SMT}$	SL	–	48.4	19.7	28.4
Rei et al. (2017) FCE$_{SMT+PAT}$ + EVP$_{SMT+PAT}$	SL	–	49.1	21.9	30.1
None	BiLSTM	47.9	43.6	16.6	24.3
FCE$_{TS}$	BiLSTM	51.2	47.1	19.7	28.9
EVP$_{BS}$	BiLSTM	52.1	50.1	20.8	29.0
SW$_{TS}$	BiLSTM	51.5	50.6	24.2	31.7
FCE$_{AM+TS}$+EVP$_{AM+TS}$	BiLSTM	52.3	50.4	22.1	30.8
None	SL	52.5	48.2	17.4	25.5
FCE$_{TS}$	SL	54.8	49.9	20.9	29.2
EVP$_{BS}$	SL	55.2	54.6	23.3	31.4
SW$_{TS}$	SL	53.8	52.7	26.8	34.3
FCE$_{AM+TS}$+EVP$_{AM+TS}$	SL	**56.9**	54.6	25.1	33.0
FCE$_{AM+TS}$+EVP$_{AM+TS}$+SW$_{AM+TS}$	SL	56.5	**55.6**	**28.3**	**35.5**

Table 2: $F_{0.5}$ scores on various tests contrasted with published results and unaugmented baseline models.

the additional computations were helpful in creating high-quality synthetic instances.

Post-processing: Original and corrupted sentences are aligned at a word-level using Levenshtein distance. Using the minimal alignment, words in the corrupted sentence are labelled *correct*, 'c', or *incorrect*, 'i', as follows:

If the word is not aligned with itself, then 'i'. *Else*, if following a gap, then 'i', as at this point a human reader would notice that there is a word missing in the sentence. *Else*, if it is the last word, but it is not aligned to the last word of the source sentence, then 'i', as a human would realise that this sentence ends abruptly, *Else*, 'c'.

These token-labelled corrupted sentences now form an artificial dataset for training an error detector. Duplicate instances and corrupted sentences with more than 5 errors were dropped to remove noise from the downstream training.

4 Experiments

We evaluated our approach on the First Certificate of English (FCE) error detection dataset (Rei and Yannakoudakis, 2016), as well as on two human-annotated test sets (CoNLL1, CoNLL2) from the CoNLL 2014 shared task (Ng et al., 2014). The CoNLL data sets pose a unique challenge; as they are different in style and domain from FCE, we have no matching training data. We compared the effect of different neural generation procedures (AM, TS, BS) and contrasted the downstream performance of a bidirectional LSTM with an elaborate sequence labeller.

4.1 Implementation details

NMT training and corruption: We minimally modified the open source implementation[1] of Britz et al. (2017) to implement TS and BS.[2] We trained our NMT with a single-layered encoder and decoder with cell size 256, on the parallel corpus version of FCE (Yannakoudakis et al., 2011), with early stopping after the FCE development set score dropped consistently for 20 epochs. We introduced errors into three datasets: FCE itself (450K tokens), the English Vocabulary Profile or EVP (270K tokens) and a subset of Simple Wikipedia or SW (8.4M tokens); of these, FCE and EVP were both used in artificial error generation via SMT and pattern extraction (PAT) by Rei et al. (2017), enabling us to make a fair experimental comparison. Ten corrupted versions using each of AM, TS ($\tau = 0.05$) and BS were sampled for FCE and EVP corruptions, while one sufficed for SW. The theoretical time complexity of BS is O(bn) for each sentence, where b is num-

---

[1] `https://github.com/google/seq2seq`

[2] `https://github.com/skasewa/wronging`

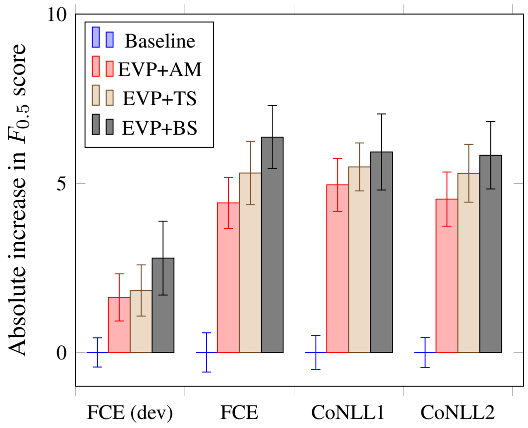

Figure 1: Improvements using three different methods of generation.

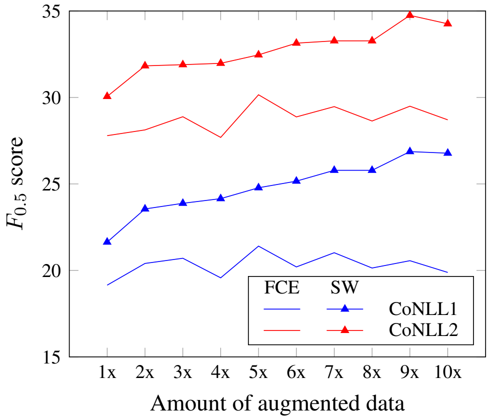

Figure 2: Training with increasing amounts of corrupted data from FCE and SW.

ber of candidates, and n is the maximum length of a sentence. Empirically, BS with $b = 11$ took a factor of 11.3 more time than AM. Examples of generated errors are provided in Table 1.

Error detection: We compare two error detection models: a vanilla bi-directional LSTM (BiLSTM) (Schuster and Paliwal, 1997), and the state-of-the-art sequence labeller (SL) neural network used by Rei et al. (2017). These models were trained on the binary-labelled FCE training set augmented with the corrupted instances. Wherever no model is explicitly stated, the SL model was used. During training, we alternate between the annotated FCE dataset and the synthetic collection. This alternating protocol prevents overfitting on FCE; once it shifts back, it reinforces connections made from the helpful synthetic corruptions while forgetting about the noisy ones.

4.2 Results

The results for our baselines and data augmentation strategies can be found in Table 2. Augmented with our NMT generated data, even our vanilla downstream BiLSTM outperforms the SMT+PAT artificial error augmentation approach of Rei et al. (2017), indicating that our process better generalises the error information in the source dataset. Using the more powerful SL network bests the previous state of the art by over 5% on the FCE test. Most intriguingly, we note a significant improvement for the CoNLL tests using corruptions from out-of-domain SW. Figure 2 illustrates how we gain performance on these tests with increasing amounts of corrupted SW, which does not hold true for corrupted FCE. This shows that we were able to induce useful errors into a corpus with a large unseen vocabulary and different syntactic biases, and this in turn proved valuable for detecting errors in a third domain, suggesting that our method can transfer learned distributions across stylistic genres.

Using EVP as a standard source, Figure 1 illustrates the variance of the different sampling methods. All generation methods yield corruptions that significantly improve test performance, with instances sampled by beam-search consistently outperforming the alternatives.

5 Discussion

5.1 Error distribution

The original FCE dataset was annotated using the error taxonomy specified in Nicholls (2003), and contains 75 unique error codes. We annotated samples of EVP corrupted by all three sampling methods, at a reduced resolution, to compare the distribution of errors across FCE and the synthetic corpora. These are presented in Table 3.

At a high level, NMT generates errors more often among more common parts-of-speech, favouring errors in verbs and nouns, rather than in adverbs and conjunctions. It did not make spelling errors as often as in the source dataset; this is likely because it only observed the specific spelling errors present in FCE, and as the vocabulary is restricted to that dataset, it does not encounter those words as frequently in EVP, and thus rarely makes the same spelling mistakes.

Additionally, the differences in these distributions can partially be attributed to the implicit differences between us and the annotators of FCE.

Spelling	FCE	AM	TS	BS
Spelling errors	11	1	1	4

Part-of-speech	FCE	AM	TS	BS
Verb	34	16	26	16
Preposition	18	16	10	14
Determiner	16	7	6	10
Noun	13	36	35	43
Pronoun	7	3	3	1
Adverb	5	5	3	5
Adjective	3	15	16	12
Conjunction	2	2	2	1
Quantifier	1	0	0	0

Remedy Type	FCE	AM	TS	BS
Replacement	49	35	34	32
Inclusion	23	30	27	35
Removal	14	33	36	32
Word form	9	2	2	1
Word order	5	0	0	0

Table 3: Error distribution across FCE and manually annotated samples of artificial data. Spelling errors are a % of all errors, while Part-of-speech, and Remedy Type are compared within their own categories to sum to 100%.

5.2 Comparison with human errors

To check if the synthetic instances passed for human-like, we mixed 50 generated sentences among an equal number of actual ungrammatical instances from FCE-dev and tasked a human evaluator to identify the artificial statements, in a simple Turing-style test. We created three such sets, one for each of our sampling techniques, and the test subject aimed to identify synthetic samples with high confidence. Results of this test are presented in Table 4.

The high precision but low recall scores suggest that while it is still possible to spot some corruptions that are quite clearly artificial, the bulk of our samples do not betray their synthetic nature and are indistinguishable from naturally occurring erroneous sentences. In order to fairly compare our work with earlier results, we intended to conduct such a test for sentences generated by the SMT of Rei et al. (2017). Unfortunately, we were only able to source corruptions of FCE-train via this method; therefore, we decided not to perform this test as its results cannot be compared to ours.

	AM	TS	BS
Precision	81.25	63.63	50.00
Recall	26.00	28.00	14.00
F1	39.39	38.89	22.22

Table 4: Results of a Turing-style test, where a subject was asked to distinguish between real and fake sentences, sampled from each of the different generated corpora.

6 Conclusions and future work

We presented a novel data augmentation technique for grammatical error detection using neural machine translation to learn the distribution of language-learner errors, and induce such errors into grammatically correct text. We explored several different variants of sampling to improve the quality of our synthetic errors. After creating artificial training instances with an off-the-shelf NMT, we bettered previous state-of-the-art results on the canonical test with even a basic BiLSTM, and established a new state of the art using a stronger model. Additionally, we demonstrated that we were able to leverage corruptions of an out-of-domain dataset to set new benchmarks on separate, also out-of-domain tests, without specifically optimising for either.

Our work indicates that neural error generation warrants further investigation with different datasets and architectures, both for error detection and error correction. Among possible future work is using generative adversarial networks as corruption engines, and developing better sequence alignment methods. Some preliminary results with simple corruptions using word substitution and word dropout (Iyyer et al., 2015) appear to be promising, and may feature as components of a future corruption system. Finally, one could use such artificial error-prone corpora as source text for self-training an error detection system.

Acknowledgements

We thank Marek Rei and Mariano Felice for granting access to their data and code. Also, we would like to thank the anonymous reviewers and Johannes Welbl for valuable feedback and discussions. This work was supported by an Allen Distinguished Investigator Award.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Guillaume Bouchard, Pontus Stenetorp, and Sebastian Riedel. 2016. Learning to generate textual data. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1608–1616, Austin, Texas. Association for Computational Linguistics.

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive Exploration of Neural Machine Translation Architectures. *ArXiv e-prints*.

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting esl errors using phrasal smt techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Sydney, Australia. Association for Computational Linguistics.

Mariano Felice and Zheng Yuan. 2014. Generating artificial errors for grammatical error correction. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–126, Gothenburg, Sweden. Association for Computational Linguistics.

Jennifer Foster and Øistein E Andersen. 2009. Generrate: generating errors for use in grammatical error detection. In *Proceedings of the fourth workshop on innovative use of nlp for building educational applications*, pages 82–90. Association for Computational Linguistics.

Andrew R Golding and Dan Roth. 1996. Applying winnow to context-sensitive spelling correction. *arXiv preprint cmp-lg/9607024*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

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

Kenji Imamura, Kuniko Saito, Kugatsu Sadamitsu, and Hitoshi Nishikawa. 2012. Grammar error correction using pseudo-error sentences and domain adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 388–392. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The amu system in the conll-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 25–33, Baltimore, Maryland. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, Austin, Texas. Association for Computational Linguistics.

Kevin Knight, Ishwar Chander, Matthew Haines, Vasileios Hatzivassiloglou, Eduard Hovy, Masayo Iida, Steve K Luk, Akitoshi Okumura, Richard Whitney, and Kenji Yamada. 1994. Integrating knowledge bases and statistics in mt. *arXiv preprint cmp-lg/9409001*.

Nina H Macdonald. 1983. Human factors and behavioral science: The unix writer's workbench software: Rationale and design. *Bell Labs Technical Journal*, 62(6):1891–1908.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

Courtney Napoles and Chris Callison-Burch. 2017. Systematically adapting machine translation for grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 345–356, Copenhagen, Denmark. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and elt. In *Proceedings of the Corpus Linguistics 2003 conference*, volume 16, pages 572–581.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the*

55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2121–2130, Vancouver, Canada. Association for Computational Linguistics.

Marek Rei, Gamal Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318, Osaka, Japan. The COLING 2016 Organizing Committee.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 287–292, Copenhagen, Denmark. Association for Computational Linguistics.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1191, Berlin, Germany. Association for Computational Linguistics.

Stephen D. Richardson and Lisa C. Braden-Harder. 1988. The experience of developing a large-scale natural language text processing system: Critique. In *Second Conference on Applied Natural Language Processing*.

Alla Rozovskaya and Dan Roth. 2010. Training paradigms for correcting errors in grammar and usage. In *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*, pages 154–162. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2014. Building a state-of-the-art grammatical error correction system. *Transactions of the Association for Computational Linguistics*, 2:419–434.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Jason Wang and Luis Perez. 2017. The effectiveness of data augmentation in image classification using deep learning. Technical report, Technical report.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

Helen Yannakoudakis, Marek Rei, Øistein E. Andersen, and Zheng Yuan. 2017. Neural sequence-labelling models for grammatical error correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2795–2806, Copenhagen, Denmark. Association for Computational Linguistics.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

Zheng Yuan and Mariano Felice. 2013. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 52–61, Sofia, Bulgaria. Association for Computational Linguistics.

Modeling Input Uncertainty in Neural Network Dependency Parsing

Rob van der Goot
University of Groningen
`r.van.der.goot@rug.nl`

Gertjan van Noord
University of Groningen
`g.j.m.van.noord@rug.nl`

Abstract

Recently introduced neural network parsers allow for new approaches to circumvent data sparsity issues by modeling character level information and by exploiting raw data in a semi-supervised setting. Data sparsity is especially prevailing when transferring to non-standard domains. In this setting, lexical normalization has often been used in the past to circumvent data sparsity. In this paper, we investigate whether these new neural approaches provide similar functionality as lexical normalization, or whether they are complementary.

We provide experimental results which show that a separate normalization component improves performance of a neural network parser even if it has access to character level information as well as external word embeddings. Further improvements are obtained by a straightforward but novel approach in which the top-N best candidates provided by the normalization component are available to the parser.

1 Introduction

Recently, neural network dependency parsers (Chen and Manning, 2014; Dyer et al., 2015; Kiperwasser and Goldberg, 2016) obtained state-of-the-art performance for dependency parsing. These parsers incorporate character level information (de Lhoneux et al., 2017a; Ballesteros et al., 2015; Nguyen et al., 2017) and can more easily exploit raw text in a semi-supervised setup. These new methods are especially beneficial for words not occurring in the training data. In practice, such unseen words often are spelling mistakes, or alternative spellings of known words. In more classical parsing models, these unseen words were usually clustered using ad-hoc rules. For non-standard domains, the number of unseen words is much larger. To minimize the degradation in performance, lexical normalization is often used. Lexical normalization is the task of converting non-standard input to a more standard format. Previous work has shown that this is beneficial, in particular for parsing social media data (Foster, 2010; Zhang et al., 2013; van der Goot and van Noord, 2017b).

This leads to the question whether normalization is indeed no longer required for these modern character-based neural network parsers, or whether normalization is capable of solving problems beyond the scope of this type of neural network parsers.

Our main contributions are:

- We show that using normalization as pre-processing improves parser performance for non-standard language, even if pre-trained embeddings and character level information are used.
- We propose a novel technique to exploit the top-N candidates provided by the normalization component, and we show that this technique leads to a further increase in parser performance.
- A treebank containing non-standard language is created to evaluate the effect of normalization on parser performance. The treebank consists of 10,005 tokens annotated with lexical normalization and Universal Dependencies (Nivre et al., 2017). The treebank has been made publicly available.

2 Related Work

Early work on parser adaptation focused on relatively canonical domains, like biomedical data (McClosky and Charniak, 2008). More recently, there has been an increasing interest in parsing of the notoriously noisy domain of social media. A lot of previous work is orthogonal to our

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4984–4991
Brussels, Belgium, October 31 - November 4, 2018.

Original word	new		pix		comming		tomoroe	
Cand. 1 (p1)	new	(0.95)	pix	(0.79)	coming	(0.57)	tomorrow	(0.54)
Cand. 2 (p2)	news	(0.03)	selfies	(0.08)	comming	(0.43)	tomoroe	(0.39)
Cand. 3 (p3)	knew	(0.01)	pictures	(0.06)	combing	(<0.01)	tomorrow's	(0.02)

Table 1: Output of the normalization model for the example sentence "new pix comming tomoroe" including candidate probabilities. Only the top-3 candidates are shown here.

approach, as it focuses on adaptation of the training data (Foster et al., 2011; Khan et al., 2013; Kong et al., 2014; Blodgett et al., 2018). In the remainder of this section we will shortly review work which evaluated the effect of normalization on dependency parsing.

Zhang et al. (2013) tune a normalization model for the parsing task, and show performance improvement on a silver treebank obtained from manually normalized data. Daiber and van der Goot (2016) use an existing normalization model as pre-processing for a graph-based dependency parser, and show a small but significant performance improvement. In the shared task of parsing the web (Petrov and McDonald, 2012) held at SANCL 2012, some teams used a simple rule-based normalization, but the effect on final performance remained untested.

Baldwin and Li (2015) examined the theoretical impact of different normalization actions on parsing performance. To this end they use manual normalization. They show that edits beyond the word level can also be crucial for parsing (e.g. insertion of copulas and subjects). However, these are difficult to obtain automatically.

Note that all this previous work, except for Blodgett et al. (2018), is based on traditional feature-based dependency parsers, whereas we focus on neural network parsing.

3 Method

In this section we will first shortly review the two models we will combine: a lexical normalization model and a neural network parser. Then we describe how they can be combined.

3.1 Normalization

In this work we use an existing normalization model: MoNoise (van der Goot and van Noord, 2017a)[1]. This model is based on the observation that normalization requires a variety of different replacement actions. For these different actions, different modules are used to generate candidates, including: the Aspell spell checker[2], word embeddings and a lookup list generated from the training data. Features from these generation modules are complemented with N-gram features from canonical data and non-canonical data. A random forest classifier is used to score and rank the candidates. In this work, we use the top-N candidates and convert the confidence scores of the classifier to probabilities. An example of this output is shown in Table 1.

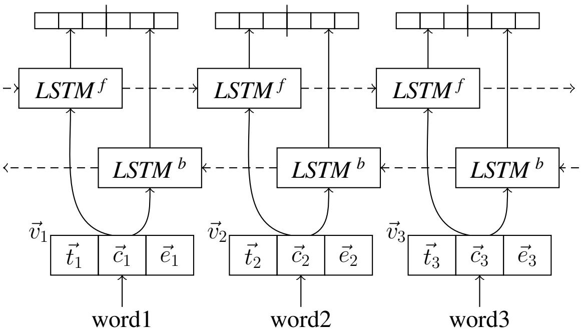

Figure 1: Overview of the conversion of input words to vectors which are used in the shift-reduce algorithm.

We train MoNoise on 2,577 tweets annotated with normalization by Li and Liu (2014), which only contains word-word replacements. In our inital experiments, we noted that the normalization model wrongfully normalized some words due to the different tokenization in the treebank (e.g. "ca n't"), because these do not occur in the normalization data. We manually created a list of exceptions, which are not considered for normalization process.

3.2 Neural Network Parser

As a starting point, we use the shift-reduce UUParser 2.0 (de Lhoneux et al., 2017b; Kiperwasser and Goldberg, 2016). This parser uses the Arc-Hybrid Transition system (Kuhlmann et al., 2011). Words are first converted to continu-

---

[1] `https://bitbucket.org/robvanderg/monoise`

[2] `www.aspell.net`

ous vectors, which are then processed through a Bidirectional Long-Short Term Memory network (BiLSTM) (Graves and Schmidhuber, 2005) before they are passed on to the parsing algorithm. The decision whether to shift, reduce or swap is made by a multi-layer perceptron with one hidden layer. The BiLSTM is trained jointly with the parsing objective, so that the vectors are optimized for the parsing task.

Figure 1 shows an overview of how the input words are converted to vectors which are used in the shift-reduce algorithm. We denote the vector used as input to the BiLSTM for word i by $\vec{v}_i$. This vector is a concatenation of three vectors which are derived from the input word. $\vec{t}_i$ is optimized on the training data, $\vec{c}_i$ is the result of a separate BiLSTM ran over the characters of word i and $\vec{e}_i$ is the external vector; it is obtained from external embeddings which are trained on huge amounts of raw texts. In this work we use the same word embeddings as used by the normalization model (van der Goot and van Noord, 2017a), which are trained on 760,744,676 tweets using word2vec (Mikolov et al., 2013).

3.3 Adaptation Strategy

Notation We use $\vec{w}_0 ... \vec{w}_n$ to represent the vectors of the original words of a sentence. The vectors of the normalization candidates are represented by $\vec{n}_{ij}$, where i is the index of the original word in the sentence, and j is the rank of the candidate. The corresponding probability as given by the normalization model is p_{ij}. We use $\vec{g}_i$ for the vector of the manual normalization of word i

Our baseline setup is to simply use the vector of the original word:

$$\text{ORIG:} \quad \vec{v}_i = \vec{w}_i$$

The most straightforward use of normalization is to use the best normalization sequence as input to the parser. In our setup, this means that we use the vector of the best normalization candidate for each position:

$$\text{NORM:} \quad \vec{v}_i = \vec{n}_{i0}$$

To give more information to the parser, we will exploit the top-n candidates of the normalization model. The vectors of the top-N candidates are merged using linear interpolation:

$$\text{INTEGRATED:} \quad \vec{v}_i = \sum_{j=0}^{n} p_{ij} * \vec{n}_{ij}$$

An interesting property of this integration approach is that it does not influence the size of the search space, so the effect on complexity of the parsing algorithm is negligible. The only extra runtime compared to ORIG originates from running the normalization model.

Finally, we include a theoretical upperbound of the effect of normalization, which uses manually annotated normalization:

$$\text{GOLD:} \quad \vec{v}_i = \vec{g}_i$$

4 Data

To test the effect of normalization, we need a treebank containing non-standard language, preferably with a corresponding training treebank from a more standard domain. Since the existing treebanks are not noisy enough (Foster et al., 2011; Kaljahi et al., 2015)[3] or do not have a corresponding training treebank in the same annotation format (Kong et al., 2014; Daiber and van der Goot, 2016) we annotate a small treebank for development and testing purposes[4]. We choose to use the Universal Dependencies 2.1 annotation format (Nivre et al., 2017), since the annotation efforts on the the English Web Treebank (Silveira et al., 2014) provide suitable training data. This treebank already contains web specific phenomena like URL's, E-Mail addresses and emoticons, so we do not have to create special annotation guidelines and the parser can learn these phenomena from the training data.

Our treebank consists of tweets, taken from Li and Liu (2015). The tweets in this dataset originate from two sources: the LexNorm corpus (Han and Baldwin, 2011), which was originally annotated with normalization, and a corpus originally annotated with POS tags (Owoputi et al., 2013). Li and Liu (2015) complemented this annotation for both datasets, so that they both have a normalization layer and a POS layer. To avoid overfitting on a specific filtering or time-frame we use the data collected by Owoputi et al. (2013) as development data and LexNorm as test data. We only keep the tweets which are still available on Twitter, resulting in a dataset of 305 development and

---

[3] Kaljahi et al. (2015) only normalize 3.6%, and we manually normalized the development data from Foster et al. (2011), were even less words were in need of normalization.

[4] It should be noted that two other suitable Twitter treebanks in the UD format where created in parallel to our treebank (Liu et al., 2018; Blodgett et al., 2018), which were released after submission of this paper.

327 test tweets (10,005 tokens in total). It should be noted that these corpora were filtered to contain domain-specific phenomena and non-standard language, and thus provide an ideal testbed for our experiments but are not representative of the whole Twitter domain.

Tokenization and normalization are first re-annotated, because the Universal Dependencies format requires treebank specific tokenization. To avoid parser bias, dependency relations are annotated from scratch. For more details on annotation decisions for domain-specific structures, we refer to the appendix.

MoNoise reaches 90% accuracy on the word level for the normalization task for our development data. In this dataset, 18% of all words are in need of normalization, so a baseline which simply copies the original words would reach an accuracy of 82%. The most common mistakes made by MoNoise are due to treebank specific normalizations, like ‘na’ ↦ go. However, these also occur in the training treebank, so normalization is not crucial.

5 Evaluation

In this section, we first use the development data to compare the effect of the different normalization settings with the use of character level information and external embeddings. Secondly, we confirm our main results on the test set. Thirdly, we test if our model is sensitive to over-normalization on standard data. Finally, we perform some analysis to examine why normalization is beneficial. All scores reported in this section are obtained using the CoNLL 2017 evaluation script (Zeman et al., 2017). In Section 5.1 the results are the average over ten runs, using a different seed for the BiLSTM and the shuffling of the training data. In the remainder of this section, the best model is used to simplify interpretation. The parser is trained using default settings (de Lhoneux et al., 2017b).

In our initial experiments, it became apparent that the parser often considered a username mention or retweet in the beginning of the tweet as root, resulting in a propagation of errors. Because we want to exclude any influences from this simple construction, we added an heuristic to our parser which exclude usernames and retweets in the beginning of a tweet, and connects them to the root after parsing. We use this heuristic in all experiments.

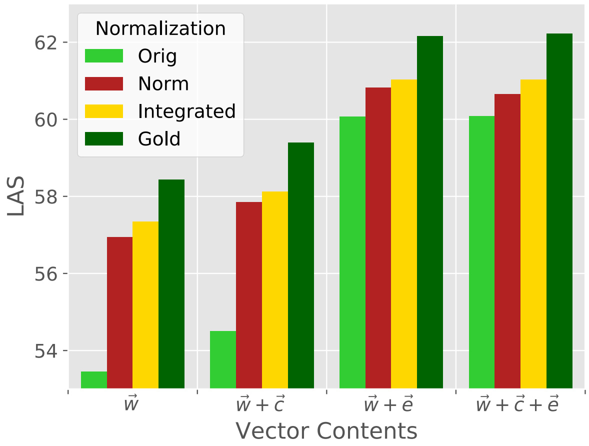

Figure 2: The effect of normalization on LAS for the different parsing models on the development data.

5.1 Normalization Strategies

The results of the different parser and normalization settings on the development data are plotted in Figure 2. Using external embeddings ($\vec{e}$) results in a much bigger performance improvement compared to using character level information ($\vec{c}$). Adding character level embeddings on top of external embeddings only leads to a very minor improvement. This can partly be explained by the coverage of 98.4% of the embeddings on the development data.

In the settings without external embeddings, the direct use of normalization (NORM) results in a improvement of approximately 3 LAS points. However, when external embeddings are included the improvement becomes more than twice as small, indicating that the approaches target some common issues, but are also complementary to each other. When external embeddings and normalization are already used, the character level embeddings slightly harm performance. Integration of the normalization (INTEGRATED) consistently results in a slightly higher LAS compared to direct normalization. Interestingly, gold normalization still performs substantially better compared to automatic normalization.

5.2 Test Data

Table 2 shows the results of the parser with external embeddings and character embeddings (using the best seed from the development data), for the different normalization strategies on the test data. These results confirm the observations on the development data: normalization helps on top of ex-

Model	UAS	LAS
ORIG	69.63	59.64
NORM	70.51	61.76*
INTEGRATED	70.62	62.30*
GOLD	70.71	62.33

Table 2: LAS scores for the Twitter test data. *Statistically significant compared to the previous row at $P < 0.05$ using a paired t-test.

ternal embeddings, and integrating normalization results in an even higher score. In contrast to the development data, the integrated approach almost reaches the theoretical upper bound of gold normalization on the test data. However, since this is only the case on the test data, not too strong conclusions can be drawn from this result. The performance difference between the datasets is probably partly due to the differences in filtering[5]. Interestingly, integrating normalization is especially beneficial for the LAS, meaning that it is most useful for choosing the type of relation.

5.3 Robustness

As stated in Section 4, our development and test data is filtered to be very non-standard. However, it is undesirable to have a parser that performs bad on more standard texts. Hence, we also tested performance on the English Web Treebank development set. This dataset also consists of data from the web, however, it contains much less words in need of normalization; MoNoise normalizes less than 0.5% of all words. We compared the performance using no normalization (ORIG) versus our INTEGRATED approach, which showed a very minor performance improvement from 81.42 to 81.43 LAS. This is a direct effect of the normalization model giving high probabilities to the original words on this more canonical data.

5.4 Analysis

To gain insights into which constructions are parsed better when using normalization, we compared the predictions of the vanilla parser with our NORM and INTEGRATED methods on the development data. Starting with NORM, the first observation is that the incoming arcs of the words which are normalized are responsible for 44.1% of all improvements, whereas the outgoing arcs are responsible for 17.6% of al improvements. So, the direct context of the normalized words is responsible for only 61.7% of all improvements. Considering the type of syntactic constructions for which parsing improved, it is hard to identify trends, because the improvements are based on the output of the normalization model, which normalizes a wide variety of words. One clearly influential effect of using normalization, was that the parser improved upon finding the root. When multiple unknown words occured in the beginning of a sentence, the vanilla parser often failed at identifying the root, which improved considerably after normalizing.

For the INTEGRATED method, almost all the improvements made by NORM remained. On top of these, some additional improvements where made. Manual inspection revealed that these improvements often originated from a non-standard word, for which the correct normalization was ranked high. This then leads to improvements for the non-standard word as well as its context. In some cases, even incorrect normalization candidates lead to performance improvements. For example for 'Gma', where the normalization model ranked the original word first, but 'mom' second. Even though 'grandma' is the correct normalization, 'mom' occurs in similar contexts, and is much easier for the parser to process.

6 Conclusion

We showed that normalization can improve performance of a neural network parser, even when making use of character level information and external word embeddings. Integrating multiple normalization candidates into the parser leads to an even larger performance increase. Normalization has shown to be complementary to external embeddings, in contrast to character embeddings, which add no additional information. Our experiments revealed that our approach is robust, and it does not harm performance on more canonical data. However, when comparing our approach to the theoretical upperbound of using gold normalization, we saw that on different datasets the performance gain is of a different magnitude. Furthermore, we release a dataset containing 636 tweets annotated with both normalization and Universal Dependencies. The data and all code to reproduce the results in this paper is available at: `https://bitbucket.org/robvanderg/normpar`

---

[5] Even when using the best seed on the development data, INTEGRATED results in two-thirds of the performance improvement compared to GOLD.

Acknowledgements

We would like to thank Gosse Bouma for help with the data annotation, and Antonio Toral, Barbara Plank and the anonymous reviewers for feedback on the paper. This work is part of the 'Parsing Algorithms for Uncertain Input' project sponsored by the Nuance Foundation.

References

Tyler Baldwin and Yunyao Li. 2015. An in-depth analysis of the effect of text normalization in social media. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 420–429, Denver, Colorado. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal. Association for Computational Linguistics.

Su Lin Blodgett, Johnny Wei, and Brendan O'Connor. 2018. Twitter Universal Dependency parsing for African-American and mainstream American English. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Joachim Daiber and Rob van der Goot. 2016. The Denoised Web Treebank: Evaluating dependency parsing under noisy input conditions. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jennifer Foster. 2010. "cba to check the spelling": Investigating parser performance on discussion forum posts. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 381–384, Los Angeles, California. Association for Computational Linguistics.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. From news to comment: Resources and benchmarks for parsing the language of web 2.0. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 893–901, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Rob van der Goot and Gertjan van Noord. 2017a. MoNoise: Modeling noise using a modular normalization system. *Computational Linguistics in the Netherlands Journal*, 7:129–144.

Rob van der Goot and Gertjan van Noord. 2017b. Parser adaptation for social media by integrating normalization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 491–497, Vancouver, Canada. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 368–378, Portland, Oregon, USA. Association for Computational Linguistics.

Rasoul Kaljahi, Jennifer Foster, Johann Roturier, Corentin Ribeyre, Teresa Lynn, and Joseph Le Roux. 2015. Foreebank: Syntactic analysis of customer support forums. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1341–1347, Lisbon, Portugal. Association for Computational Linguistics.

Mohammad Khan, Markus Dickinson, and Sandra Kuebler. 2013. Does size matter? text and grammar revision for parsing social media data. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 1–10, Atlanta, Georgia. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference* *on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012, Doha, Qatar. Association for Computational Linguistics.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 673–682, Portland, Oregon, USA. Association for Computational Linguistics.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017a. From raw text to Universal Dependencies – look, no tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies.*, Vancouver, Canada.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017b. Arc-hybrid non-projective dependency parsing with a static-dynamic oracle. In *Proceedings of the The 15th International Conference on Parsing Technologies (IWPT).*, Pisa, Italy.

Chen Li and Yang Liu. 2014. Improving text normalization via unsupervised model and discriminative reranking. In *Proceedings of the ACL 2014 Student Research Workshop*, pages 86–93, Baltimore, Maryland, USA. Association for Computational Linguistics.

Chen Li and Yang Liu. 2015. Joint POS tagging and text normalization for informal text. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1263–1269.

Yijia Liu, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider, and Noah A. Smith. 2018. Parsing tweets into Universal Dependencies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 965–975. Association for Computational Linguistics.

David McClosky and Eugene Charniak. 2008. Self-training for biomedical parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 101–104. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*.

Dat Quoc Nguyen, Mark Dras, and Mark Johnson. 2017. A novel neural network model for joint POS tagging and graph-based dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 134–142.

Joakim Nivre, Zeljko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çağr Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, John Lee, Phuong Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Luong Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Robert Ostling, Lilja Ovrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Simková, Kiril Simov, Aaron Smith, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jonathan North Washington, Mats Wirén, Tak-sum Wong, Zhuoran Yu, Zdeněk Zabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal Dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics ('UFAL), Faculty of Mathematics and Physics, Charles University.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia. Association for Computational Linguistics.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, volume 59.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağr Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld, and Yunyao Li. 2013. Adaptive parser-centric text normalization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1159–1168, Sofia, Bulgaria. Association for Computational Linguistics.

Parameter sharing between dependency parsers for related languages

Miryam de Lhoneux[1*] **Johannes Bjerva**[2] **Isabelle Augenstein**[2] **Anders Søgaard**[2]
[1]Department of Linguistics and Philology
Uppsala University
Uppsala, Sweden
[2] Department of Computer Science
University of Copenhagen
Copenhagen, Denmark

Abstract

Previous work has suggested that parameter sharing between transition-based neural dependency parsers for related languages can lead to better performance, but there is no consensus on what parameters to share. We present an evaluation of 27 different parameter sharing strategies across 10 languages, representing five pairs of related languages, each pair from a different language family. We find that sharing transition classifier parameters always helps, whereas the usefulness of sharing word and/or character LSTM parameters varies. Based on this result, we propose an architecture where the transition classifier is shared, and the sharing of word and character parameters is controlled by a parameter that can be tuned on validation data. This model is linguistically motivated and obtains significant improvements over a mono-lingually trained baseline. We also find that sharing transition classifier parameters helps when training a parser on unrelated language pairs, but we find that, in the case of unrelated languages, sharing too many parameters does not help.

1 Introduction

The idea of sharing parameters between parsers of related languages goes back to early work in cross-lingual adaptation (Zeman and Resnik, 2008), and the idea has recently received a lot of interest in the context of neural dependency parsers (Duong et al., 2015; Ammar et al., 2016; Susanto and Lu, 2017). Modern neural dependency parsers, however, use different sets of parameters for representation and scoring, and it is not clear what parameters it is best to share.

The Universal Dependencies (UD) project (Nivre et al., 2016), which is seeking to harmonize the annotation of dependency treebanks across languages, has seen a steady increase in languages that have a treebank in a common standard. Many of these languages are low resource and have small UD treebanks. It seems interesting to find out ways to leverage the wealth of information contained in these treebanks, especially for low resource languages.

In this paper, we evaluate 27 different parameter sharing strategies. We focus on a particular transition-based neural dependency parser (de Lhoneux et al., 2017a,b), which performs close to the state of the art. This parser has three sets of parameters: i) the parameters of a character-based one-layer, bidirectional LSTM; ii) the parameters of a word-based two-layer, bidirectional LSTM; iii) and the parameters of a multi-layered perceptron (MLP) with a single hidden layer. The two first sets are for learning to represent configurations; the third for selecting the next transition. We consider all combinations of sharing these sets of parameters; and in addition, we consider two ways of sharing each set of parameters, namely with and without a prefixed language embedding. The latter enables partial, soft sharing. In sum, we consider all 3^3 combinations of no sharing, hard sharing and soft sharing of the three sets of parameters. We evaluate the 27 multilingual parsers on 10 languages from the UD project, representing five pairs of related languages, each pair from a different language family. We repeat the experiment with the same set of languages, but using pairs of unrelated languages.

Contributions This paper is, to the best of our knowledge, the first to evaluate different parameter sharing strategies for exploiting synergies between neural dependency parsers of related languages. We evaluate the different strategies on 10 languages, representing five different language families. We find that sharing (MLP) transition

---

[*] Work carried out during a stay at the University of Copenhagen.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4992–4997
Brussels, Belgium, October 31 - November 4, 2018.

Lang	Tokens	Family	Word order
ar	208,932	Semitic	VSO
he	161,685	Semitic	SVO
et	60,393	Finnic	SVO
fi	67,258	Finnic	SVO
hr	109,965	Slavic	SVO
ru	90,170	Slavic	SVO
it	113,825	Romance	SVO
es	154,844	Romance	SVO
nl	75,796	Germanic	No dom. order
no	76,622	Germanic	SVO

Table 1: Dataset characteristics

classifier parameters always helps, whereas the usefulness of sharing LSTM parameters depends on the language pair. This reflects the intuition that the transition classifier learns hierarchical structures that are likely to transfer across languages, based on parser configurations that abstract away from several linguistic differences. The similarity of the input to character- and word-level LSTMs, on the other hand, will vary depending on the phonological and morphosyntactic similarity of the languages in question. Motivated by this observation, we propose an architecture with hard-wired transition classifier parameter sharing, but in which sharing of LSTM parameters is tuned. The novel architecture significantly outperforms our monolingual baseline on our set of 10 languages. We additionally investigate parameter sharing of unrelated languages.

2 The Uppsala dependency parser

The Uppsala parser (de Lhoneux et al., 2017a,b) consists of three sets of parameters; the parameters of the character-based LSTM, those of the word-based LSTM, and the parameters of the MLP that predicts transitions. The character-based LSTM produces representations for the word-based LSTM, which produces representations for the MLP. The Uppsala parser is a transition-based parser (Kiperwasser and Goldberg, 2016), adapted to the Universal Dependencies (UD) scheme,[1] and using the arc-hybrid transition system from Kuhlmann et al. (2011) extended with a SWAP transition and a static-dynamic oracle, as described in de Lhoneux et al. (2017b). The SWAP transition is used to generate non-projective dependency trees (Nivre, 2009).

For an input sentence of length n with words $w_1, \dots, w_n$, the parser creates a sequence of vectors $x_{1:n}$, where the vector x_i representing w_i is the concatenation of a word embedding and the final state of the character-based LSTM after processing the characters of w_i. The character vector $ch(w_i)$ is obtained by running a (bi-directional) LSTM over the characters ch_j $(1 \leq j \leq m)$ of w_i. Each input element is represented by the word-level, bi-directional LSTM, as a vector $v_i = \textsc{BiLstm}(x_{1:n}, i)$. For each configuration, the feature extractor concatenates the LSTM representations of core elements from the stack and buffer. Both the embeddings and the LSTMs are trained together with the model.

A configuration c is represented by a feature function $\phi(\cdot)$ over a subset of its elements. For each configuration, transitions are scored by a classifier, in this case an MLP, and $\phi(\cdot)$ is a concatenation of BiLSTM vectors on top of the stack and the beginning of the buffer. The MLP scores transitions together with the arc labels for transitions that involve adding an arc. In practice, we use two interpolated MLPs, one which only scores the transitions, and one which scores transitions together with the arc label. For simplicity, we refer to that interpolated MLP as the MLP.

3 Parameter sharing

Since our parser has three basic sets of model parameters, we consider sharing all combinations of those three sets. We also introduce two ways of sharing, namely, with or without the addition of a vector representing the language. This language embedding enables the model, in theory, to learn what to share between the two languages in question. Since for all three model parameter sets, we now have three options – not sharing, sharing, or sharing in the context of a language embedding – we are left with $3^3 = 27$ parameter sharing strategies; see Table 2.

In the setting where we do not share (✗) word parameters (**W**), we construct a different word lookup table and a different word-level BiLSTM for each language. In the setting where we do hard parameter sharing (✓) of word parameters, we only construct one lookup table and one word BiLSTM for the languages involved. In the setting where we do soft sharing (ID) of word pa-

---

[1] `http://universaldependencies.org/`

Model	**C**	**W**	**S**	ar	he	es	it	et	fi	nl	no	hr	ru	Av
Mono				76.3	80.2	83.7	83.3	70.4	70.8	77.3	80.8	76.8	82.3	78.2
Language-best				76.6	80.6	84.4	84.8	72.8	72.9	79.6	82.1	78.0	82.9	79.5
Best	✗	✓	ID	76.3	80.3	84.2	84.5	72.1	72.5	78.8	81.4	77.6	82.8	79.1
Char	✓	✗	✗	76.4	80.3	84.3	84.0	72.3	71.0	78.3	81.3	77.0	82.3	78.7
Word	✗	✓	✗	76.3	79.9	83.9	84.4	72.4	71.3	77.4	80.7	76.9	82.5	78.6
State	✗	✗	✓	76.6	80.3	84.0	83.7	71.5	72.9	78.3	81.5	77.4	82.8	78.9
	⋮													
All	✓	✓	✓	76.2	80.1	84.0	84.2	72.1	71.4	78.7	81.1	77.0	82.5	78.7
Soft	ID	ID	ID	76.3	79.9	84.1	84.4	72.1	71.3	79.6	81.4	77.1	82.5	78.9

Table 2: Performance on development data (LAS; in %) across select sharing strategies. Mono is our single-task baseline; Language-best is using the best sharing strategy for each language (as evaluated on development data); Best is the overall best sharing strategy, across languages; Char shares only the character-based LSTM parameters; Word shares only the word-based LSTM parameters; All shares all parameters. ✓ refers to hard sharing, ID refers to soft sharing, using an embedding of the language ID and ✗ refers to not sharing.

rameters, we share those parameters, and in addition, concatenate a language embedding l_i representing the language of word w_i to the vector of the word w_i at the input of the word BiLSTM: $x_i = e(w_i) \circ ch(w_i) \circ l_i$. Similarly for character parameters (**C**), we construct a different character BiLSTM and one character lookup for each language (✗), create those for all languages and share them (✓) or share them and concatenate a (randomly initialized) language embedding l_i representing the language of word w_i at the input of the character BiLSTM (ID): $ch_j = e(ch_j) \circ l_i$. At the level of configuration or parser states (**S**), we either construct a different MLP for each language (✗), share the MLP (✓) or share it and concatenate a language embedding l_i representing the language of word w_i to the vector representing the configuration, at the input of the MLP (ID): $c = \phi(\cdot) \circ l_i$.

4 Experiments

Language pairs We use 10 languages in our experiments, representing five language pairs from different language families. Our two Semitic languages are Arabic and Hebrew. These two languages differ in that Arabic tends to favour VSO word order whereas Hebrew tends to use SVO, but are similar in their rich transfixing morphology. Our two Finno-Ugric languages are Estonian and Finnish. These two languages differ in that Estonian no longer has vowel harmony, but share a rich agglutinative morphology. Our two Slavic languages are Croatian and Russian. These two languages differ in that Croatian uses gender in plural nouns, but otherwise share their rich inflectional morphology. Our two Romance languages are Italian and Spanish. These two languages differ in that Italian uses a possessive adjective with a definite article, but share a fairly strict SVO order. Finally, our two Germanic languages are Dutch and Norwegian. These two languages differ in morphological complexity, but share word ordering features to some extent.

Datasets For all 10 languages, we use treebanks from the Universal Dependencies project. The dataset characteristics are listed in Table 1. To keep the results comparable across language pairs, we down-sample the training set to the size of the smallest of our languages, Hebrew: we randomly sample 5000 sentences for each training set. Note that while this setting makes the experiment somewhat artificial and will probably overestimate the benefits that can be obtained from sharing parameters when using larger treebanks, we find it interesting to see how much low resource languages can benefit from parameter sharing, as explained in the introduction.

Baselines and systems This is an evaluation paper, and our results are intended to explore a space of sharing strategies to find better ways of sharing parameters between dependency parsers of related languages. Our baseline is the Uppsala parser trained monolingually. Our systems are parsers trained bilingually by language pair where we share subsets of parameters between the languages in the pair, and we report on what sharing strategies seem superior across the 10 languages that we consider.

Implementation details A flexible implementation of parameter strategies for the Uppsala parser was implemented in Dynet.[2] We make the code publicly available.[3]

5 Results and discussion

Our results on development sets are presented in Table 2. We use labeled attachment score (LAS) as our metric for evaluating parsers. Table 2 presents numbers for a select subset of the 27 sharing strategies. The other results can be found in the supplementary material. Our main observations are: **(i)** that, generally, and as observed in previous work, *multi-task learning helps*: all different sharing strategies are on average better than the monolingual baselines, with minor (0.16 LAS points) to major (0.86 LAS points) average improvements; and **(ii)** that sharing the MLP seems to be overall a better strategy than not sharing it: the 10 best strategies share the MLP. Whereas the usefulness of sharing the MLP seems to be quite robust across language pairs, the usefulness of sharing word and character parameters seems more dependent on the language pairs. This reflects the linguistic intuition that character- and word-level LSTMs are highly sensitive to phonological and morphosyntactic differences such as word order, whereas the MLP learns to predict less idiosyncratic, hierarchical relations from relatively abstract representations of parser configurations.

Based on this result, we propose a model (OURS) where *the MLP is shared and the sharing of word and character parameters is controlled by a parameter that can be set on validation data.* Results are given in Table 3. We obtain a 0.6 LAS improvement on average and our proposed model is significantly better than the monolingual baseline with $p < 0.01$. Significance testing is performed using a randomization test, with the script from the CoNLL 2017 Shared Task.[4]

	W	**C**	OURS	MONO	δ
ar	✗	✗	77.2	77.1	0.1
es	ID	✓	84.3	83.8	0.5
et	✗	ID	71.4	70.5	0.8
fi	✗	✗	71.6	71.6	0.1
he	✓	✗	80.0	79.8	0.3
hr	✓	✗	77.9	78.0	-0.1
it	ID	✓	85.0	84.0	1.0
nl	ID	✓	75.5	74.1	1.4
no	✗	ID	81.1	80.1	1.0
ru	✓	✗	83.5	82.7	0.8
av.			78.8	78.2	0.6

Table 3: LAS on the test sets of the best of 9 sharing strategies and the monolingual baseline. δ is the difference between OURS AND MONO.

6 Unrelated languages

We repeated the same set of experiments with unrelated language pairs. We hypothesise that parameter sharing between unrelated language pairs will be less useful in general than with related language pairs. However, it can still be useful, it has been shown previously that unrelated languages can benefit from being trained jointly. For example, Lynn et al. (2014) have shown that Indonesian was surprisingly particularly useful for Irish.

The results are presented in Table 4. The table only presents part of the results, the rest can be found in the supplementary material. As expected, there is much less to be gained from sharing parameters between unrelated pairs. However, it is possible to improve the monolingual baseline by sharing some of the parameters. In general, sharing the MLP is still a helpful thing to do. It is most helpful to share the MLP and optionally one of the two other sets of parameters. Results are close to the monolingual baseline when everything is shared. Sharing word and character parameters but not the MLP hurts accuracy compared to the monolingual baseline.

7 Related work

Previous work has shown that sharing parameters between dependency parsers for related languages can lead to improvements (Duong et al., 2015; Ammar et al., 2016; Susanto and Lu, 2017). Smith et al. (2018) recently found that sharing parameters using the same parser as in this paper

---

[2] `https://github.com/clab/dynet`

[3] `https://github.com/coastalcph/uuparser`

[4] `https://github.com/udapi/udapi-python/blob/master/udapi/block/eval/conll17.py`

Model	C	W	S	he	no	fi	hr	ru	es	it	et	nl	ar	Av
Mono				80.2	80.8	70.8	76.8	82.3	83.7	83.3	70.4	77.3	76.3	78.2
Language-best				80.5	81.5	71.9	77.6	82.9	84.0	84.3	72.5	78.7	76.5	78.9
Best	✗	✗	✓	80.3	81.5	71.9	77.6	82.7	84.0	83.8	72.5	78.7	76.3	78.9
Worst	ID	ID	✗	79.8	80.6	69.2	76.7	81.4	83.8	83.2	69.4	76.6	76.0	77.7
Char	✓	✗	✗	80.1	80.9	71.4	76.8	82.9	83.9	84.3	70.9	78.0	76.5	78.6
Word	✗	✓	✗	79.6	80.9	71.9	76.9	82.2	83.7	83.8	70.9	77.0	76.4	78.3
All	✓	✓	✓	80.5	80.9	69.8	76.6	82.3	83.7	84.0	70.6	77.4	76.2	78.2
Soft	ID	ID	ID	79.8	80.5	70.1	76.6	82.1	83.9	83.8	70.6	77.2	76.3	78.1

Table 4: Performance on development data (LAS; in %) across select sharing strategies for unrelated languages. Mono is our single-task baseline; Language-best is using the best sharing strategy for each language (as evaluated on development data); Best and Worst are the overall best and worst sharing strategy across languages; Char shares only the character-based LSTM parameters; Word shares only the word-based LSTM parameters; All shares all parameters. ✓ refers to hard sharing, ID refers to soft sharing, using an embedding of the language ID and ✗ refers to not sharing.

(soft sharing of word parameters, hard sharing of the rest) improves parsing accuracy when training on related languages, and is especially useful in the low resource case. Similar effects have been observed in machine translation (Dong et al., 2015; Johnson et al., 2017), for example. Most studies have only explored a small number of parameter sharing strategies, however. Vilares et al. (2016) evaluate parsing with hard parameter sharing for 100 language pairs with a statistical parser. Naseem et al. (2012) proposed to selectively share subsets of a parser across languages in the context of a probabilistic parser.

Options we do not explore here are learning the architecture jointly with optimizing the task objective (Misra et al., 2016; Ruder et al., 2017), or learning an architecture search model that predicts an architecture based on the properties of datasets, typically with reinforcement learning (Zoph and Le, 2017; Wong and Gesmundo, 2018; Liang et al., 2018). We also do not explore the option of sharing selectively based on more fine-grained typological information about languages, which related work has indicated could be useful (Bjerva and Augenstein, 2018). Rather, we stick to sharing between languages of the same language families.

The strategies explored here do not exhaust the space of possible parameter sharing strategies. For example, we completely ignore soft sharing based on mean-constrained regularisation (Duong et al., 2015).

8 Conclusions

We present evaluations of 27 parameter sharing strategies for the Uppsala parser across 10 languages, representing five language pairs from five different language families. We repeated the experiment with pairs of unrelated languages. We made several observations: (a) Generally, multi-task learning helps. (b) Sharing the MLP parameters always helps. It helps to share MLP parameters when training a parser on a pair of related languages, and it also helps if the languages are unrelated. (c) Sharing word and character parameters is differently helpful depending on the language. (d) Sharing too many parameters does not help, when the languages are unrelated.

In future work, we plan to investigate what happens when training on more than 2 languages. Here, we focused on a setting with rather small amounts of balanced data. It would be interesting to experiment with using datasets that are not balanced with respect to size. Finally, we have restricted our experiments to a specific architecture, using fixed hyperparameters including word and character embedding dimensions. It would be interesting to experiment with different parsing architectures as well as varying those hyperparameters.

Acknowledgments

We acknowledge the computational resources provided by CSC in Helsinki and Sigma2 in Oslo through NeIC-NLPL (www.nlpl.eu). The second author is partially funded by an AdeptMind scholarship. The last author was funded by an ERC Starting Grant.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. More languages, one parser. In *TACL*.

Johannes Bjerva and Isabelle Augenstein. 2018. From Phonology to Syntax: Unsupervised Linguistic Typology at Different Levels with Language Embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 907–916. Association for Computational Linguistics.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL*.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser. In *Proceedings of ACL*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Vigas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's neural machine translation system. In *TACL*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *TACL*, 4:313–327.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic Programming Algorithms for Transition-Based Dependency Parsers. In *Proceedings of ACL*, pages 673–682, Portland, Oregon, USA.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017a. From raw text to universal dependencies - look, no tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 207–217, Vancouver, Canada.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017b. Arc-Hybrid Non-Projective Dependency Parsing with a Static-Dynamic Oracle. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 99–104, Pisa, Italy.

Jason Liang, Elliot Meyerson, and Risto Miikkulainen. 2018. Evolutionary Architecture Search For Deep Multitask Networks. In *GECCO*.

Teresa Lynn, Jennifer Foster, Mark Dras, and Lamia Tounsi. 2014. Cross-lingual transfer parsing for low-resourced languages: An irish case study. In *Proceedings of the First Celtic Language Technology Workshop*, pages 41–49.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-Stitch Networks for Multi-Task Learning. In *Proceedings of CVPR*.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637. Association for Computational Linguistics.

Joakim Nivre. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of ACL*, pages 351–359, Suntec, Singapore.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. In *CoRR, abs/1705.08142*.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 Treebanks, 34 Models: Universal Dependency Parsing with Multi-Treebank Models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Raymond Hendy Susanto and Wei Lu. 2017. Neural architectures for multilingual semantic parsing. In *ACL*.

David Vilares, Carlos Gómez-Rodríguez, and Miguel A Alonso. 2016. One model, two languages: training bilingual parsers with harmonized treebanks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 425–431.

Catherine Wong and Andrea Gesmundo. 2018. Transfer Learning to Learn with Multitask Neural Model Search. In *ICPR*.

Daniel Zeman and Philip Resnik. 2008. Cross-Language Parser Adaptation between Related Languages. In *IJCNLP*.

Barret Zoph and Quoc Le. 2017. Neural Architecture Search with Reinforcement Learning. In *ICPR*.

Grammar Induction with Neural Language Models: An Unusual Replication

Phu Mon Htut[1]
AdeptMind Scholar
pmh330@nyu.edu

Kyunghyun Cho[1,2]
CIFAR Global Scholar
kyunghyun.cho@nyu.edu

Samuel R. Bowman[1,2,3]
bowman@nyu.edu

[1]Center for Data Science
New York University
60 Fifth Avenue
New York, NY 10011

[2]Dept. of Computer Science
New York University
60 Fifth Avenue
New York, NY 10011

[3]Dept. of Linguistics
New York University
10 Washington Place
New York, NY 10003

Abstract

A substantial thread of recent work on *latent tree learning* has attempted to develop neural network models with parse-valued latent variables and train them on non-parsing tasks, in the hope of having them discover interpretable tree structure. In a recent paper, Shen et al. (2018) introduce such a model and report near-state-of-the-art results on the target task of language modeling, and the first strong latent tree learning result on constituency parsing. In an attempt to reproduce these results, we discover issues that make the original results hard to trust, including tuning and even training on what is effectively the test set. Here, we attempt to reproduce these results in a fair experiment and to extend them to two new datasets. We find that the results of this work are robust: All variants of the model under study outperform all latent tree learning baselines, and perform competitively with symbolic grammar induction systems. We find that this model represents the first empirical success for latent tree learning, and that neural network language modeling warrants further study as a setting for grammar induction.

1 Introduction and Background

Work on *grammar induction* attempts to find methods for syntactic parsing that do not require expensive and difficult-to-design expert-labeled treebanks for training (Charniak and Carroll, 1992; Klein and Manning, 2002; Smith and Eisner, 2005). Recent work on *latent tree learning* offers a new family of approaches to the problem (Yogatama et al., 2017; Maillard et al., 2017; Choi et al., 2018). Latent tree learning models attempt to induce syntactic structure using the supervision from a downstream NLP task such as textual entailment. Though these models tend to show good task performance, they are often not evaluated using standard parsing metrics, and Williams et al. (2018a) report that the parses they produce tend to be no better than random trees in a standard evaluation on the full Wall Street Journal section of the Penn Treebank (WSJ; Marcus et al., 1993).

This paper addresses the Parsing-Reading-Predict Network (PRPN; Shen et al., 2018), which was recently published at ICLR, and which reports near-state-of-the-art results on language modeling and strong results on grammar induction, a first for latent tree models (though they do not use that term). PRPN is built around a substantially novel architecture, and uses convolutional networks with a form of structured attention (Kim et al., 2017) rather than recursive neural networks (Goller and Kuchler, 1996; Socher et al., 2011) to evaluate and learn trees while performing straightforward backpropagation training on a language modeling objective. In this work, we aim to understand what the PRPN model learns that allows it to succeed, and to identify the conditions under which this success is possible.

Their experiments on language modeling and parsing are carried out using different configurations of the PRPN model, which were claimed to be optimized for the corresponding tasks. PRPN-LM is tuned for language modeling performance, and PRPN-UP for (unsupervised) parsing performance. In the parsing experiments, we also observe that the WSJ data is not split, such that the test data is used without parse information for training. This approach follows the previous works on grammar induction using non-neural models where the entire dataset is used for training (Klein and Manning, 2002). However, this implies that the parsing results of PRPN-UP may not be generalizable in the way usually expected of machine learning evaluation results. Additionally, it is not obvious that the model should be able to learn to parse reliably: (1) Since the parser is trained as part of a language model, it makes pars-

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4998–5003
Brussels, Belgium, October 31 - November 4, 2018.

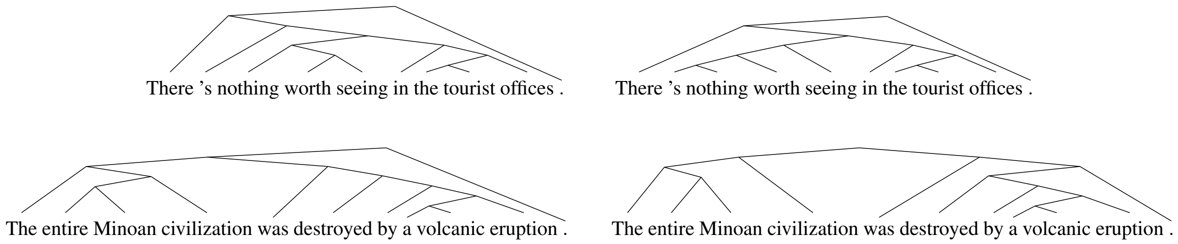

Figure 1: *Left* Parses from PRPN-LM trained on AllNLI. *Right* Parses from PRPN-UP trained on AllNLI (stopping criterion: parsing). We can observe that both sets of parses tend to have roughly reasonable high-level structure and tend to identify noun phrases correctly.

ing decisions greedily and with *no* access to any words to the right of the point where each parsing decision must be made (Collins and Roark, 2004); (2) As RNN language models are known to be insufficient for capturing syntax-sensitive dependencies (Linzen et al., 2016), language modeling as the downstream task may not be well-suited to latent tree learning.

In this replication we train PRPN on two corpora: The full WSJ, a staple in work on grammar induction, and AllNLI, the concatenation of the Stanford Natural Language Inference Corpus (SNLI; Bowman et al., 2015) and the Multi-Genre NLI Corpus (MultiNLI; Williams et al., 2018b), which is used in other latent tree learning work for its non-syntactic classification labels for the task of textual entailment, and which we include for comparison. We then evaluate the constituency trees produced by these models on the WSJ test set, full WSJ10,[1] and the MultiNLI development set.

Our results indicate that PRPN-LM achieves better parsing performance than PRPN-UP on both WSJ and WSJ10 even though PRPN-UP was tuned—at least to some extent—for parsing. Surprisingly, a PRPN-LM model trained on the large out-of-domain AllNLI dataset achieves the best parsing performance on WSJ despite not being tuned for parsing. We also notice that vocabulary size affects the language modeling significantly—the perplexity gets higher as the vocabulary size increases.

Overall, despite the relatively uninformative experimental design used in Shen et al. (2018), we find that PRPN is an effective model. It outperforms all latent tree learning baselines by large margins on both WSJ and MultiNLI, and performs competitively with symbolic grammar induction systems on WSJ10, suggesting that PRPN in particular and language modeling in general are a viable setting for latent tree learning.

2 Methods

PRPN consists of three components: (i) a *parsing network* that uses a two-layer convolution kernel to calculate the *syntactic distance* between successive pairs of words, which can form an indirect representation of the constituency structure of the sentence, (ii) a recurrent *reading network* that summarizes the current memory state based on all previous memory states and the implicit constituent structure, and (iii) a *predict network* that uses the memory state to predict the next token. We refer readers to the appendix and the original work for details.

We do not re-implement or re-tune PRPN, but rather attempt to replicate and understand the results of the work using the author's publicly available code.[2] The experiments on language modeling and parsing are carried out using different configurations of the model, with substantially different hyperparameter values including the size of the word embeddings, the maximum sentence length, the vocabulary size, and the sizes of hidden layers. PRPN-LM is larger than PRPN-UP, with embedding layer that is 4 times larger and the number of units per layer that is 3 times larger. We use both versions of the model in all our experiments.

We use the 49k-sentence WSJ corpus in two settings. To replicate the original results, we re-run an experiment with no train/test split, and for a clearer picture of the model's performance, we run it again with the train (Section 0-21 of WSJ), validation (Section 22 of WSJ), and test (Section 23

---

[1] A standard processed subset of WSJ used in grammar induction in which the sentences contain no punctuation and no more than 10 words.

[2] `https://github.com/yikangshen/PRPN`

Model	Training Data	Stopping Criterion	Vocab Size	Parsing F1				Depth WSJ	Accuracy on WSJ by Tag			
				WSJ10		WSJ			ADJP	NP	PP	INTJ
				μ (σ)	max	μ (σ)	max					
PRPN-UP	AllNLI Train	UP	76k	67.5 (0.6)	68.6	36.9 (0.6)	38.0	5.8	29.3	62.0	31.6	*0.0*
PRPN-UP	AllNLI Train	LM	76k	66.3 (0.8)	68.5	38.3 (0.5)	39.8	5.8	28.7	65.5	32.7	*0.0*
PRPN-LM	AllNLI Train	LM	76k	52.4 (4.9)	58.1	35.0 (5.4)	**42.8**	6.1	**37.8**	59.7	**61.5**	**100.0**
PRPN-UP	WSJ Full	UP	15.8k	64.7 (3.2)	70.9	26.4 (1.7)	31.1	5.8	22.5	47.2	17.9	*0.0*
PRPN-UP	WSJ Full	LM	15.8k	64.3 (3.3)	70.8	26.3 (1.8)	30.8	5.8	22.7	46.6	17.8	*0.0*
PRPN-UP	WSJ Train	UP	15.8k	63.5 (3.5)	70.7	26.2 (2.3)	33.0	5.8	24.8	55.2	18.0	*0.0*
PRPN-UP	WSJ Train	LM	15.8k	62.2 (3.9)	70.3	26.0 (2.3)	32.8	5.8	24.8	54.4	17.8	*0.0*
PRPN-LM	WSJ Train	LM	10k	70.5 (0.4)	71.3	37.4 (0.3)	38.1	5.9	26.2	**63.9**	24.4	*0.0*
PRPN-LM	WSJ Train	UP	10k	66.1 (0.5)	67.2	33.4 (0.8)	35.6	5.9	33.0	57.1	18.3	*0.0*
300D ST-Gumbel	AllNLI Train	NLI	–	–	–	*19.0 (1.0)*	*20.1*	–	*15.6*	*18.8*	*9.9*	59.4
w/o Leaf GRU	AllNLI Train	NLI	–	–	–	22.8 (1.6)	25.0	–	18.9	24.1	*14.2*	51.8
300D RL-SPINN	AllNLI Train	NLI	–	–	–	*13.2 (0.0)*	*13.2*	–	*1.7*	*10.8*	*4.6*	50.6
w/o Leaf GRU	AllNLI Train	NLI	–	–	–	*13.1 (0.1)*	*13.2*	–	*1.6*	*10.9*	*4.6*	50.0
CCM	WSJ10 Full	–	–	–	71.9	–	–	–	–	–	–	–
DMV+CCM	WSJ10 Full	–	–	–	77.6	–	–	–	–	–	–	–
UML-DOP	WSJ10 Full	–	–	–	**82.9**	–	–	–	–	–	–	–
Random Trees	–	–	–	–	34.7	21.3 (0.0)	21.4	5.3	17.4	22.3	16.0	40.4
Balanced Trees	–	–	–	–	–	21.3 (0.0)	21.3	4.6	22.1	*20.2*	*9.3*	55.9
Left Branching	–	–	–	*28.7*	28.7	13.1 (0.0)	13.1	12.4	–	–	–	–
Right Branching	–	–	–	61.7	61.7	16.5 (0.0)	16.5	12.4	–	–	–	–

Table 1: Unlabeled parsing F1 results evaluated on full WSJ10 and WSJ test set broken down by training data and by early stopping criterion. The *Accuracy* columns represent the fraction of ground truth constituents of a given type that correspond to constituents in the model parses. Italics mark results that are worse than the random baseline. Underlining marks the best results from our runs. Results with RL-SPINN and ST-Gumbel are from Williams et al. (2018a), and are evaluated on the full WSJ. We run the model with 5 different random seeds to calculate the average F1. We use the model with the best F1 score to report ADJP, NP, PP, and INTJ. WSJ10 baselines are from Klein and Manning (2002, CCM), Klein and Manning (2005, DMV+CCM), and Bod (2006, UML-DOP). As the WSJ10 baselines are trained using additional information such as POS tags and dependency parser, they are not strictly comparable with the latent tree learning results.

of WSJ) splits. To compare PRPN to the models studied in Williams et al. (2018a), we also retrain it on AllNLI. As the MultiNLI test set is not publicly available, we follow Williams et al. (2018a) and use the development set for testing. The parsing evaluation code in the original codebase does not support PRPN-LM, and we modify it in our experiments only to add this support.

For early stopping, we remove 10k random sentences from the MultiNLI training set and combine them with the SNLI development set to create a validation set. Our AllNLI training set contains 280.5K unique sentences (1.8M sentences in total including duplicate premise sentences), and covers six distinct genres of spoken and written English. We do not remove the duplicate sentences. We train the model for 100 epochs for WSJ and 15 epochs for AllNLI. We run the model five times with random initializations and average the results from the five runs. The generated parses from the trained models with the best F1 scores and the pre-trained model that provides the highest F1 are available online.[3]

3 Experimental Results

Table 2 shows our results for language modeling. PRPN-UP, configured as-is with parsing criterion and language modeling criterion, performs dramatically worse than the standard PRPN-LM (a vs. d and e). However, this is not a fair comparison as the larger vocabulary gives PRPN-UP a harder task to solve. Adjusting the vocabulary of PRPN-UP down to 10k to make a fairer comparison possible, the PPL of PRPN-UP improves significantly (c vs. d), but not enough to match PRPN-LM (a vs. c). We also observe that early stopping on

---

[3] `https://github.com/nyu-mll/PRPN-Analysis`

Model	Training Data	Stopping Criterion	Vocab Size	PPL Median
(a) PRPN-LM	WSJ Train	LM	10k	**61.4**
(b) PRPN-LM	WSJ Train	UP	10k	81.6
(c) PRPN-UP	WSJ Train	LM	10k	92.8
(d) PRPN-UP	WSJ Train	LM	15.8k	112.1
(e) PRPN-UP	WSJ Train	UP	15.8k	112.8
(f) PRPN-UP	AllNLI Train	LM	76k	797.5
(g) PRPN-UP	AllNLI Train	UP	76k	848.9

Table 2: Language modeling performance (perplexity) on the WSJ test set, broken down by training data used and by whether early stopping is done using the parsing objective (UP) or the language modeling objective (LM).

Model	Stopping Criterion	F1 wrt.			Depth
		LB	RB	SP	
300D SPINN	NLI	19.3	36.9	70.2	6.2
w/o Leaf GRU	NLI	21.2	39.0	63.5	6.4
300D SPINN-NC	NLI	19.2	36.2	70.5	6.1
w/o Leaf GRU	NLI	20.6	38.9	64.1	6.3
300D ST-Gumbel	NLI	32.6	37.5	23.7	4.1
w/o Leaf GRU	NLI	30.8	35.6	27.5	4.6
300D RL-SPINN	NLI	95.0	13.5	18.8	8.6
w/o Leaf GRU	NLI	99.1	10.7	18.1	8.6
PRPN-LM	LM	25.6	26.9	45.7	4.9
PRPN-UP	UP	19.4	41.0	46.3	4.9
PRPN-UP	LM	19.9	37.4	**48.6**	4.9
Random Trees	–	27.9	28.0	27.0	4.4
Balanced Trees	–	21.7	36.8	21.3	3.9

Table 3: Unlabeled parsing F1 on the MultiNLI development set for models trained on AllNLI. *F1 wrt.* shows F1 with respect to strictly right- and left-branching (LB/RB) trees and with respect to the Stanford Parser (SP) trees supplied with the corpus; The evaluations of SPINN, RL-SPINN, and ST-Gumbel are from Williams et al. (2018a). SPINN is a supervised parsing model, and the others are latent tree models. Median F1 of each model trained with 5 different random seeds is reported.

parsing leads to incomplete training and a substantial decrease in perplexity (a vs. b and d vs. e). The models stop training at around the 13th epoch when we early-stop on parsing objective, while they stop training around the 65th epoch when we early-stop on language modeling objective. Both PRPN models trained on AllNLI do even worse (f and g), though the mismatch in vocabulary and domain may explain this effect. In addition, since it takes much longer to train PRPN on the larger AllNLI dataset, we train PRPN on AllNLI for only 15 epochs while we train the PRPN on WSJ for 100 epochs. Although the parsing objective converges within 15 epochs, we notice that language modeling perplexity is still improving. We expect that the perplexity of the PRPN models trained on AllNLI could be lower if we increase the number of training epochs.

Turning toward parsing performance, Table 1 shows results with all the models under study, plus several baselines, on WSJ test set and full WSJ10. On full WSJ10, we reproduce the main parsing result of Shen et al. (2018) with their UP model trained on WSJ without a data split. We also find the choice of parse quality as an early stopping criterion does not have a substantial effect and that training on the (unlabeled) test set does not give a significant improvement in performance. In addition and unexpectedly, we observe that PRPN-LM models achieve *higher* parsing performance than PRPN-UP. This shows that any tuning done to separate PRPN-UP from PRPN-LM was not necessary, and more importantly, that the results described in the paper can be largely reproduced by a unified model in a fair setting. Moreover, the PRPN models trained on WSJ achieves comparable results with CCM (Klein and Manning, 2002). The PRPN models are outperformed by DMV+CCM(Klein and Manning, 2005), and UML-DOP(Bod, 2006). However, these models use additional information such as POS and dependency parser so they are not strictly comparable with the PRPN models.

Turning to the WSJ test set, the results look somewhat different: Although the differences in WSJ10 performance across models are small, the same is not true for the WSJ in terms of average F1. PRPN-LM outperforms all the other models on WSJ test set, even the potentially-overfit PRPN-UP model. Moreover, the PRPN models trained on the larger, out-of-domain AllNLI perform better than those trained on WSJ. Surprisingly, PRPN-LM tained on out-of-domain AllNLI achieves the best F1 score on WSJ test set among all the models we experimented, even though its performance on WSJ10 is the lowest of all. This mean that PRPN-LM trained on AllNLI is strikingly good at parsing longer sentences though its performance on shorter sentences is worse than other models. Under all the configurations we tested, the PRPN model yields much better performance than the baselines from Yogatama et al. (2017, called RL-SPINN) and Choi et al. (2018, called ST-Gumbel), despite the fact that the model was tuned exclusively for WSJ10 parsing. This suggests that PRPN is consistently effective at latent tree learning.

We also show detailed results for several specific constituent types, following Williams et al. (2018a). We observe that the accuracy for NP (noun phrases) on the WSJ test set is above 46% (Table 1) for all PRPN models, much higher than any of the baseline models. These runs also perform substantially better than the random baseline in the two other categories Williams et al. (2018a) report: ADJP (adjective phrases) and PP (prepositional phrases). However, as WSJ test set contains only one INTJ (interjection phrases), the results on INTJ are either 0.0% or 100%.

In addition, Table 3 shows that the PRPN-UP models achieve the median parsing F1 scores of 46.3 and 48.6 respectively on the MultiNLI dev set while PRPN-LM performs the median F1 of 45.7; setting the state of the art in parsing performance on this dataset among latent tree models by a large margin. We conclude that PRPN does acquire some substantial knowledge of syntax, and that this knowledge agrees with Penn Treebank (PTB) grammar significantly better than chance.

Qualitatively, the parses produced by most of the best performing PRPN models are relatively balanced (F1 score of 36.5 w.r.t balanced trees) and tend toward right branching (F1 score of 42.0 with respect to balanced trees). They are also shallower than average ground truth PTB parsed trees. These models can parse short sentences relatively well, as shown by their high WSJ10 performance.

For a large proportion of long sentences, most of the best performing models can produce reasonable constituents (Table 1). The best performing model, PRPN-LM trained on AllNLI, achieves the best accuracy at identifying ADJP (adjective phrases), PP (prepositional phrases), and INTJ (interjection phrases) constituents, and a high accuracy on NP (noun phrases). In a more informal inspection, we also observe that our best PRPN-LM and PRPN-UP runs are fairly good at pairing determiners with NPs as we can observe in Figure 1). Although lower level tree constituents appear random in many cases for both PRPN-LM and PRPN-UP, the intermediate and higher-level constituents are generally reasonable. For example, in Figure 1, although the parse for lower level constituents like *The entire Minoan* seem random, the higher-level constituents, such as *The entire Minoan civilization* and *nothing worth seeing in the tourist offices*, are reasonable.

4 Conclusion

In our attempt to replicate the grammar induction results reported in Shen et al. (2018), we find several experimental design problems that make the results difficult to interpret. However, in experiments and analyses going well beyond the scope of the original paper, we find that the PRPN model presented in that work is nonetheless robust. It represents a viable method for grammar induction and the first clear success for latent tree learning with neural networks, and we expect that it heralds further work on language modeling as a tool for grammar induction research.

Acknowledgments

This project has benefited from financial support to SB by Google and Tencent Holdings, and was partly supported by Samsung Electronics (Improving Deep Learning using Latent Structure). We thank Adina Williams, Katharina Kann, Ryan Cotterell, and the anonymous reviewers for their helpful comments and suggestions, and NVIDIA for their support.

References

Rens Bod. 2006. An All-Subtrees Approach to Unsupervised Parsing. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 865–872.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Eugene Charniak and Glen Carroll. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Proceedings of the AAAI Workshop on Statistically-Based NLP Techniques*, page 113.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the Thirty-Second Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI-18)*, volume 2.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, pages 111–118.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96).*

Sepp Hochreiter and Jürgen Schmidhuber. 1996. Long Short Term Memory. *Memory*, (1993):1–28.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, page 128.

Dan Klein and Christopher D. Manning. 2005. Natural language grammar induction with a generative constituent-context model. *Pattern Recognition*, 38(9):1407–1419.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *TACL*, 4:521–535.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised Tree-LSTMs. arXiv preprint 1705.09189.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Noah A. Smith and Jason Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *Proceedings of IJCAI Workshop on Grammatical Inference Applications*, pages 73–82.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Ng, and Chris Manning. 2011. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 129–136.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018a. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics (TACL).*

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018b. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL).*

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to Compose Words into Setences with Reinforcement Learning. *Proceedings of the International Conference on Learning Representations*, pages 1–17.

Data Augmentation via Dependency Tree Morphing for Low-Resource Languages

Gözde Gül Şahin
UKP Lab, Department of Computer Science
Technische Universität Darmstadt
Darmstadt, Germany
sahin@ukp.informatik.tu-darmstadt.de

Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh, Scotland
steedman@inf.ed.ac.uk

Abstract

Neural NLP systems achieve high scores in the presence of sizable training dataset. Lack of such datasets leads to poor system performances in the case low-resource languages. We present two simple text augmentation techniques using dependency trees, inspired from image processing. We "crop" sentences by removing dependency links, and we "rotate" sentences by moving the tree fragments around the root. We apply these techniques to augment the training sets of low-resource languages in Universal Dependencies project. We implement a character-level sequence tagging model and evaluate the augmented datasets on part-of-speech tagging task. We show that crop and rotate provides improvements over the models trained with non-augmented data for majority of the languages, especially for languages with rich case marking systems.

1 Introduction

Most recently, various deep learning methods have been proposed for many natural language understanding tasks including sentiment analysis, question answering, dependency parsing and semantic role labeling. Although these methods have reported state-of-the-art results for languages with rich resources, no significant improvement has been announced for low-resource languages. In other words, feature-engineered statistical models still perform better than these neural models for low-resource languages.[1] Generally accepted reason for low scores is the size of the training data, i.e., training labels being too sparse to extract meaningful statistics.

Label-preserving data augmentation techniques are known to help methods generalize better by increasing the variance of the training data. It has been a common practice among researchers in computer vision field to apply data augmentation, e.g., flip, crop, scale and rotate images, for tasks like image classification (Ciresan et al., 2012; Krizhevsky et al., 2012). Similarly, speech recognition systems made use of augmentation techniques like changing the tone and speed of the audio (Ko et al., 2015; Ragni et al., 2014), noise addition (Hartmann et al., 2016) and synthetic audio generation (Takahashi et al., 2016). Comparable techniques for data augmentation are less obvious for NLP tasks, due to structural differences among languages. There are only a small number of studies that tackle data augmentation techniques for NLP, such as Zhang et al. (2015) for text classification and Fadaee et al. (2017) for machine translation.

In this work, we focus on languages with small training datasets, that are made available by the Universal Dependency (UD) project. These languages are dominantly from Uralic, Turkic, Slavic and Baltic language families, which are known to have extensive morphological case-marking systems and relatively free word order. With these languages in mind, we propose an easily adaptable, multilingual text augmentation technique based on dependency trees, inspired from two common augmentation methods from image processing: **cropping** and **rotating**. As images are cropped to focus on a particular item, we **crop** the sentences to form other smaller, meaningful and focused sentences. As images are rotated around a center, we **rotate** the portable tree fragments around the root of the dependency tree to form a synthetic sentence. We augment the training sets of these low-resource languages via crop and rotate operations. In order to measure the impact of augmentation, we implement a unified character-level sequence tagging model. We systematically

---

[1] For example, in the case of dependency parsing, recent best results from CoNLL-18 parsing shared task can be compared to the results of traditional language-specific models.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 5004–5009
Brussels, Belgium, October 31 - November 4, 2018.

train separate parts-of-speech tagging models with the original and augmented training sets, and evaluate on the original test set. We show that crop and rotate provide improvements over the non-augmented data for majority of the languages, especially for languages with rich case marking system.

2 Method

We borrow two fundamental label-preserving augmentation ideas from image processing: **cropping** and **rotation**. Image cropping can be defined as removal of some of the peripheral areas of an image to focus on the subject/object (e.g., focusing on the flower in a large green field). Following this basic idea, we aim to identify the parts of the sentence that we want to focus and remove the other chunks, i.e., form simpler/smaller meaningful sentences [2]. In order to do so, we take advantage of dependency trees which provide us with links to focuses, such as subjects and objects. The idea is demonstrated in Fig. 1b on the Turkish sentence given in Fig. 1a. Here, given a predicate *(wrote)* that governs a subject *(her father)*, an indirect object *(to her)* and a direct object *(a letter)*; we form three smaller sentences with a focus on the subject (first row in Fig. 1b: her father wrote) and the objects (second and third row) by removing all dependency links other than the focus (with its subtree). Obviously, cropping may cause semantic shifts on a sentence-level. However it preserves local syntactic tags and even shallow semantic labels.

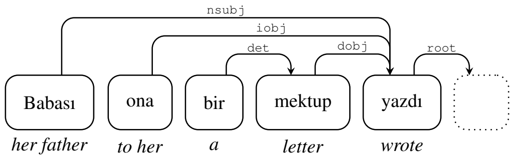

(a) Dependency analysis

(1) Babası yazdı *(Her father he-wrote)*
(2) Ona yazdı *(He-wrote to her)*
(3) Bir mektup yazdı *(He-wrote a letter)*

(b) Sentence Cropping

(1) Babası yazdı bir mektup ona (SVOIO)
(2) Yazdı babası ona bir mektup (VSIOO)
(3) Bir mektup yazdı babası ona (OVSIO)
(4) Ona bir mektup yazdı babası (IOOVS)

(c) Sentence Rotating

Figure 1: Demonstration of augmentation ideas on the Turkish sentence "Babası ona bir mektup yazdı" *(Her father wrote her a letter)*. S: Subject, V: Verb, O:Object, IO: Indirect Object. Arrows are drawn from dependent to head. Both methods are applied to the *Labels of Interest (LOI)*.

Images are rotated around a chosen center with a certain degree to enhance the training data. Similarly, we choose the root as the center of the sentence and rotate the flexible tree fragments around the root for augmentation. Flexible fragments are usually defined by the morphological typology of the language (Futrell et al., 2015). For instance, languages close to analytical typology such as English, rarely have inflectional morphemes. They do not mark the objects/subjects, therefore words have to follow a strict order. For such languages, sentence rotation would mostly introduce noise. On the other hand, large number of languages such as Latin, Greek, Persian, Romanian, Assyrian, Turkish, Finnish and Basque have no strict word order (though there is a preferred order) due to their extensive marking system. Hence, flexible parts are defined as marked fragments which are again, subjects and objects. Rotation is illustrated in Fig. 1c on the same sentence.

In order to investigate the impact of the augmentation, we design a simple sequence tagging model that operates on the character level. Many low-resource languages we deal with in the Experiments section are morphologically rich. Therefore, we use a character-level model to address the rare word problem and to learn morphological regularities among words.

For each sentence s, we produce a label sequence $\vec{l}$, where l_t refers to POS tag for the t-th token. Given g as gold labels and θ as model parameters we find the values that minimize the negative log likelihood of the sequence:

$$\hat{\theta} = \arg\min_{\theta} \left(-\sum_{t=1}^{n} log(p(g_t|\theta, s)) \right) \quad (1)$$

To calculate $p(l_t|\theta, s)$, we first calculate a word embedding, $\vec{w}$, for each word. We consider words as a sequence of characters $c_0, c_1, .., c_n$ and use a bi-LSTM unit to compose the character sequence into $\vec{w}$, as in Ling et al. (2015):

$$\vec{hw_f}, \vec{hw_b} = \text{bi-LSTM}(c_0, c_1, .., c_n) \quad (2)$$

$$\vec{w} = W_f \cdot \vec{hw_f} + W_b \cdot \vec{hw_b} + b \quad (3)$$

---

[2] Focus should not be confused with the grammatical category FOC.

Later, these embeddings are passed onto another bi-LSTM unit:

$$\vec{h_f}, \vec{h_b} = \text{bi-LSTM}(\vec{w_t}) \tag{4}$$

Hidden states from both directions are concatenated and mapped by a linear layer to the label space. Then label probabilities are calculated by a softmax function:

$$p(\vec{l_t}|s,p) = \text{softmax}(W_l \cdot [\vec{h_f}; \vec{h_b}] + \vec{b_l}) \tag{5}$$

Finally the label with the highest probability is assigned to the input.

3 Experiments and Results

We use the data provided by Universal Dependencies v2.1 (Nivre et al., 2017) project. Since our focus is on languages with low resources, we only consider the ones that have less than 120K tokens. The languages without standard splits and sizes less than 5K tokens, are ignored. We use the universal POS tags defined by UD v2.1.

To keep our approach as language agnostic and simple as possible, we use the following universal dependency labels and their subtypes to extract the focus and the flexible fragment: NSUBJ (nominal subject), IOBJ (indirect object), OBJ (indirect object) and OBL (oblique nominal). These dependency labels are referred to as Label of Interest (LOI). The root/predicate may be a phrase rather than a single token. We use the following relations to identify such cases: FIXED, FLAT, COP (copula) and COMPOUND. Other labels such as ADVMOD can also be considered flexible, however are ignored for the sake of simplicity. We enumerate all flexible chunks and calculate all reordering permutations. Keeping the LOI limited is necessary to reduce the number of permutations. We apply reordering only to the first level in the tree. Our method overgeneralizes, to include sequences that are not grammatical in the language in question. We regard the ungrammatical sentences as noise.

Number of possible cropping operations are limited to the number of items that are linked via an LOI to the root. If we call it n, then the number of possible rotations would be $(n+1)!$ since n pieces and the root are flexible and can be placed anywhere in the sentence. To limit the number of rotations, we calculate all possible permutations for reordering the sentence and then randomly pick n of them. Each operation is applied with a certain probability p to each sentence, (e.g., if $p = 1$, n number of crops; if $p = 0.5$ an average of $n/2$ crops will be done).

We use the model in Sec. 2 to systematically train part-of-speech taggers on original and augmented training data sets. To be able measure the impact of the augmentation, all models are trained with the same hyperparameters. All tokens are lowercased and surrounded with special start-end symbols. Weight parameters are uniformly initialized between -0.1 and $+0.1$. We used one layer bi-LSTMs both for character composition and POS tagging with hidden size of 200. Character embedding size is chosen as 200. We used dropout, gradient clipping and early stopping to prevent overfitting for all experiments. Stochastic gradient descent with an initial learning rate as 1 is used as the optimizer. Learning rate is reduced by half if scores on development set do not improve.

Average of multiple runs for 20 languages are given in Fig. 1. Here, Org column refers to our baseline with non-augmented, original training set, where *Imp%* is the improvement over the baseline by the best crop/flip model for that language. It is evident that, with some minor exceptions, *all* languages have benefited from a type of augmentation. We see that the biggest improvements are achieved on Irish and Lithuanian, the ones with the lowest baseline scores and the smallest training sets [3]. Our result on both languages show that both operations reduced the generalization error surprisingly well in the lack of training data.

Tagging results depend on many factors such as the training data size, the source of the treebank (e.g., news may have less objects and subjects compared to a story), and the language typology (e.g., number/type of case markers it uses). In Fig. 2, the relation between the data size and the improvement by the augmentation is shown. Pearson correlation coefficient for two variables is calculated as -0.35.

Indo-European (IE) : **Baltic** and **Slavic** languages are known to have around 7 distinct case markers, which relaxes the word order. As expectedly, both augmentation techniques improve the scores for Baltic (Latvian, Lithuanian) and Slavic (Belarusian, Slovak, Serbian, Ukranian)

---

[3] Although the total size of the Irish dataset is larger than many, the splits are unbalanced. The training set contains 121 trees while the test has 454 trees.

				crop			**rotate**			
#Tokens	**Lang**	**Type**	**Org**	$p = 0.3$	$p = 0.7$	$p = 1$	$p = 0.3$	$p = 0.7$	$p = 1$	*Imp%*
$< 20K$	Lithuanian	*IE, Baltic*	61.51	62.17	66.28	67.64	65.28	66.56	**68.27**	*10.98*
	Belarusian	*IE, Slavic*	83.58	83.87	85.50	85.39	84.33	85.96	**86.11**	*3.03*
	Tamil	*Dravidian*	81.93	81.35	82.78	**84.34**	83.74	83.86	83.61	*2.94*
	Telugu	*Dravidian*	90.78	**90.85**	89.88	90.50	90.36	90.29	89.95	*0.07*
	Coptic	*Egyptian*	**95.17**	94.60	94.74	94.12	95.03	94.65	94.60	*-0.15*
$< 80K$	Irish	*IE, Celtic*	62.75	73.72	75.87	75.42	72.51	**76.35**	76.19	*21.68*
	North Sami	*Uralic, Sami*	86.78	86.04	87.17	87.35	87.85	**88.04**	86.65	*1.45*
	Hungarian	*Uralic, Ugric*	85.94	86.24	86.56	**86.62**	86.49	86.37	86.60	*0.80*
	Vietnamese	*Austro-Asiatic*	75.16	**75.59**	75.32	74.84	75.22	75.15	75.14	*0.57*
	Turkish	*Turkic*	93.49	93.53	93.56	93.89	93.60	93.82	**93.98**	*0.52*
	Greek	*IE, Greek*	95.18	95.32	95.46	**95.54**	95.26	95.22	95.35	*0.38*
	Gothic	*IE, Germanic*	94.38	94.42	94.35	94.44	**94.62**	94.48	94.43	*0.25*
	Old Slavic	*IE, Slavic*	95.36	95.34	95.33	**95.44**	95.17	95.35	94.93	*0.08*
	Afrikaans	*IE, Germanic*	94.91	94.52	94.86	**94.93**	94.73	94.70	94.92	*0.0*
$< 120K$	Latvian	*IE, Baltic*	91.22	91.38	91.77	**91.78**	91.69	91.62	91.76	*0.61*
	Danish	*IE, Germanic*	94.25	94.17	93.96	**94.78**	94.18	94.10	94.21	*0.56*
	Slovak	*IE, Slavic*	91.23	91.17	91.04	91.35	91.53	91.38	**91.58**	*0.38*
	Serbian	*IE, Slavic*	96.14	96.26	96.12	96.17	**96.35**	96.16	96.07	*0.22*
	Ukranian	*IE, Slavic*	94.41	94.33	94.56	94.49	**94.57**	94.38	94.47	*0.17*

Table 1: POS tagging accuracies on UDv2.1 test sets. Best scores are shown with **bold**. Org: Original. p: operation probability. *Imp%*: Improvement over original (Org) by the best model trained with the augmented data.

languages, except for Old Church Slavic (OCS). OCS is solely compiled from bible text which is known to contain longer and passive sentences. We observe that rotation performs slightly better than cropping for Slavic languages. In the presence of a rich marking system, rotation can be considered a better augmenter, since it greatly increases the variance of the training data by shuffling. For **Germanic** (Gothic, Afrikaans, Danish) languages, we do not observe a repeating gain, due to lack of necessary markers.

Uralic and Turkic : Both language types have an extensive marking system. Hence, similar to Slavic languages, both techniques improve the score.

Dravidian : Case system of modern Tamil defines 8 distinct markers, which explains the improved accuracies of the augmented models. We would expect a similar result for Telugu. However Telugu treebank is entirely composed of sentences from a grammar book which may not be expressive and diverse.

4 Related Work

Similar to sentence cropping, Vickrey and Koller (2008) define transformation rules to simplify sentences (e.g., I was not given a chance to eat - I ate) and shows that enrichening training set with simplified sentences improves the results of semantic role labeling. One of the first studies in text augmentation (Zhang et al., 2015), replaces a randomly chosen word with its randomly chosen synonym extracted from a thesaurus. They report improved test scores when a large neural model is trained with the augmented set. Jia and Liang (2016) induce grammar from semantic parsing training data and generate new data points by sampling to feed a sequence to sequence RNN model. Fadaee et al. (2017) chooses low-frequency words instead of a random word, and generate synthetic sentence pairs that contain those rare words.

5 Discussion

Unlike majority of previous NLP augmentation techniques, the proposed methods are meaning-preserving, i.e., they preserve the fundamental meaning of the sentence for most of the tested languages. Therefore can be used for variety of problems such as semantic role labeling, sentiment analysis, text classification. Instead of those problems, we evaluate the idea on the simplest possible task (POS) for the following reasons:

- It gets harder to measure the impact of the idea as the system/task gets complicated due to large number of parameters.

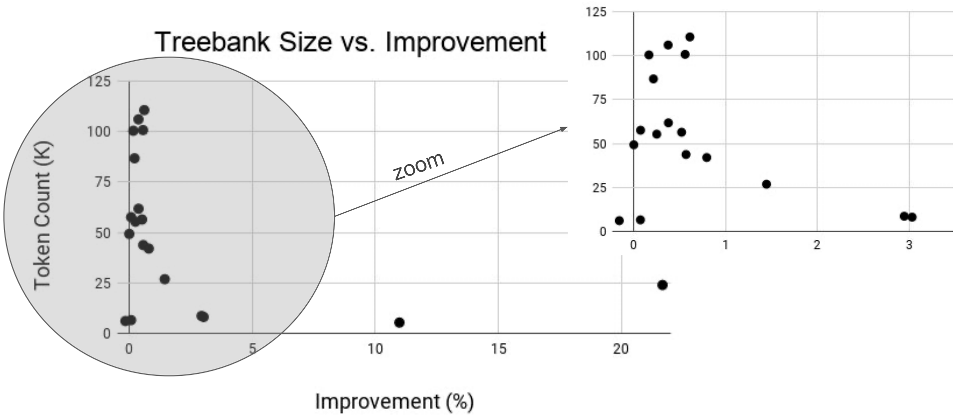

Figure 2: Treebank size versus gain by augmentation

- POS tagging performance is a good indicator of performances of other structured prediction tasks, since POS tags are crucial features for higher-level NLP tasks.

Our research interest was to observe which augmentation technique would improve which language, rather than finding one good model. Therefore we have not used development sets to choose one good augmentation model.

6 Conclusion and Future Work

Neural models have become a standard approach for many NLP problems due to their ability to extract high-level features and generalization capability. Although they have achieved state-of-the-art results in NLP benchmarks with languages with large amount of training data, low-resource languages have not yet benefited from neural models. In this work, we presented two simple text augmentation techniques using dependency trees inspired by image cropping and rotating. We evaluated their impact on parts-of-speech tagging in a number of low-resource languages from various language families. Our results show that:

- Language families with rich case marking systems *(e.g., Baltic, Slavic, Uralic)* benefit both from cropping and rotation. However, for such languages, rotation increases the variance of the data relatively more, leading to slightly better accuracies.

- Both techniques provide substantial improvements over the baseline (non-augmented data) when only a tiny training dataset is available.

This work aimed to measure the impact of the basic techniques, rather than creating the best text augmentation method. Following these encouraging results, method can be improved by (1) considering the preferred chunk order of the language during rotation, (2) taking language-specific flexibilities into account (e.g., Spanish typically allows free subject inversion (unlike object)). Furthermore, we plan to extend this work by evaluating the augmentation on other NLP benchmarks such as language modeling, dependency parsing and semantic role labeling. The code is available at `https://github.com/gozdesahin/crop-rotate-augment`.

7 Acknowledgements

Gözde Gül Şahin was a PhD student at Istanbul Technical University and a visiting research student at University of Edinburgh during this study. She was funded by Tübitak (The Scientific and Technological Research Council of Turkey) 2214-A scholarship during her visit to University of Edinburgh. This work was supported by ERC H2020 Advanced Fellowship GA 742137 SEMANTAX and a Google Faculty award to Mark Steedman. We would like to thank Adam Lopez for fruitful discussions, guidance and support during the first author's visit. We thank to the anonymous reviewers for useful comments and to Ilia Kuznetsov for his valuable feedback.

References

Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Com-*

puter Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012, pages 3642–3649.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 567–573.

Richard Futrell, Kyle Mahowald, and Edward Gibson. 2015. Quantifying word order freedom in dependency corpora. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 91–100. Uppsala University, Uppsala, Sweden.

William Hartmann, Tim Ng, Roger Hsiao, Stavros Tsakalidis, and Richard M Schwartz. 2016. Two-stage data augmentation for low-resourced speech recognition. In *Interspeech*, pages 2378–2382.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 3586–3589.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114.

Wang Ling, Tiago Luis, Luis Marujo, Ramon F Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*, pages 1520–1530.

Joakim Nivre et al. 2017. Universal dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University.

Anton Ragni, Kate M. Knill, Shakti P. Rath, and Mark J. F. Gales. 2014. Data augmentation for low resource languages. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 810–814.

Naoya Takahashi, Michael Gygli, Beat Pfister, and Luc Van Gool. 2016. Deep convolutional neural networks and data augmentation for acoustic event detection. In *Interspeech*, pages 2982–2986.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 344–352.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

How Much *Reading* Does Reading Comprehension Require? A Critical Investigation of Popular Benchmarks

Divyansh Kaushik
Language Technologies Institute
Carnegie Mellon University
`dkaushik@cs.cmu.edu`

Zachary C. Lipton
Tepper School of Business
Carnegie Mellon University
`zlipton@cmu.edu`

Abstract

Many recent papers address *reading comprehension*, where examples consist of (*question*, *passage*, *answer*) tuples. Presumably, a model must combine information from *both* questions and passages to predict corresponding answers. However, despite intense interest in the topic, with hundreds of published papers vying for leaderboard dominance, basic questions about the difficulty of many popular benchmarks remain unanswered. In this paper, we establish sensible baselines for the bAbI, SQuAD, CBT, CNN, and Who-did-What datasets, finding that question- and passage-only models often perform surprisingly well. On 14 out of 20 bAbI tasks, passage-only models achieve greater than 50% accuracy, sometimes matching the full model. Interestingly, while CBT provides 20-sentence passages, only the last is needed for comparably accurate prediction. By comparison, SQuAD and CNN appear better-constructed.

1 Introduction

Recently, *reading comprehension* (RC) has emerged as a popular task, with researchers proposing various end-to-end deep learning algorithms to push the needle on a variety of benchmarks. As characterized by Hermann et al. (2015); Onishi et al. (2016), unlike prior work addressing question answering from general structured knowledge, RC requires that a model extract information from a given, unstructured passage. It's not hard to imagine how such systems could be useful. In contrast to generic text summarization, RC systems could answer targeted questions about specific documents, efficiently extracting facts and insights.

While many RC datasets have been proposed over the years (Hirschman et al., 1999; Breck et al., 2001; Peñas et al., 2011; Peñas et al., 2012; Sutcliffe et al., 2013; Richardson et al., 2013; Berant et al., 2014), more recently, larger datasets have been proposed to accommodate the data-intensiveness of deep learning. These vary both in the source and size of their corpora and in how they cast the prediction problem—as a classification task (Hill et al., 2016; Hermann et al., 2015; Onishi et al., 2016; Lai et al., 2017; Weston et al., 2016; Miller et al., 2016), span selection (Rajpurkar et al., 2016; Trischler et al., 2017), sentence retrieval (Wang et al., 2007; Yang et al., 2015), or free-form answer generation (Nguyen et al., 2016).[1] Researchers have steadily advanced on these benchmarks, proposing myriad neural network architectures aimed at attending to both questions and passages to produce answers.

In this paper, we argue that amid this rapid progress on empirical benchmarks, crucial steps are sometimes skipped. In particular, we demonstrate that the level of difficulty for several of these tasks is poorly characterized. For example, for many RC datasets, it's not reported, either in the papers introducing the datasets, or in those proposing models, how well one can perform while ignoring either the question or the passage. In other datasets, although the passage might consist of many lines of text, it's not clear how many are actually required to answer the question, e.g., the answer may always lie in the first or the last sentence.

We describe several popular RC datasets and models proposed for these tasks, analyzing their performance when provided with question-only (Q-only) or passage-only (P-only) information. We show that on many tasks, the results obtained are surprisingly strong, outperforming many base-

---

[1] We note several other QA datasets (Yang et al., 2015; Miller et al., 2016; Nguyen et al., 2016; Paperno et al., 2016; Clark and Etzioni, 2016; Lai et al., 2017; Trischler et al., 2017; Joshi et al., 2017) not addressed in this paper.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 5010–5015
Brussels, Belgium, October 31 - November 4, 2018.

lines, and sometimes even surpassing the same models, supplied with both questions *and* passages.

We note that similar problems were shown for datasets in *visual question answering* by Goyal et al. (2017) and for *natural language inference* by Gururangan et al. (2018); Poliak et al. (2018); Glockner et al. (2018). Several other papers have discussed the weaknesses of various RC benchamrks (Chen et al., 2016; Lee et al., 2016). We discuss these studies in the paragraphs introducing the corresponding datasets below.

2 Datasets

In the following section, we provide context on each dataset that we investigate and then describe our process for corrupting the data as required by our question- and passage-only experiments.

CBT Hill et al. (2016) prepared a cloze-style (*fill in the blank*) RC dataset by using passages from children's books. In their dataset, each passage consists of 20 consecutive sentences, and each question is the 21st sentence with one word removed. The missing word then serves as the answer. The dataset is split into four categories of answers: Named Entities (NE), Common Nouns (CN), Verbs (V) and Prepositions (P). The training corpus contains over $37,000$ candidates and each question is associated with 10 candidates, POS-matched to the correct answer. The authors established LSTM/embedding-based Q-only baselines but did not present the results obtained by their best model using Q-only or P-only information.

CNN Hermann et al. (2015) introduced the CNN/Daily Mail datasets containing more than 1 million news articles, each associated with several highlight sentences. Also adopting the cloze-style dataset preparation, they remove an entity (answer) from a highlight (question). They anonymize all entities to ensure that models rely on information contained in the passage, vs memorizing characteristics of given entities across examples, and thus ignoring passages. On average, passages contain 26 entities, with over 500 total possible answer candidates. Chen et al. (2016) analyzed the difficulty of the CNN and Daily Mail tasks. They hand-engineered a set of eight features for each entity e (does e occur in the question, in the passage, etc.), showing that this simple classifier outperformed many earlier deep learning results.

Who-did-What Onishi et al. (2016) extracted pairs of news articles, each pair referring to the same events. Adopting the cloze-style, they remove a person's name (the answer) from the first sentence of one article (the question). A model must predict the answer based on the question, together with the other article in the pair (passage). Unlike CNN, Who-did-What does not anonymize entities. On average, each question is associated with 3.5 candidate answers. The authors removed several questions from their dataset to thwart simple strategies such as always predicting the name that occurs most (or first) in the passage.

bAbI Weston et al. (2016) presented a set of 20 tasks to help researchers identify and rectify the failings of their reading comprehension systems. Unlike the datasets discussed so far, the questions in this task are not cloze-style and are synthetically generated using templates. This restricts the diversity in clauses appearing in the passages. Further, this also restricts the dataset vocabulary to just 150 words, in contrast, CNN dataset has a vocabulary made of close to $120,000$ words. Memory Networks with adaptive memory, n-grams and non-linear matching were shown to obtain 100% accuracy on 12 out of 20 bAbI tasks. We note that Lee et al. (2016) previously identified that bAbI tasks might fall short as a measure of "AI-complete question answering", proposing two models based on *tensor product representations* that achieve 100% accuracy on many bAbI tasks.

SQuAD More recently, Rajpurkar et al. (2016) released the Stanford Question Answering Dataset (SQuAD) containing over $100,000$ crowd-sourced questions addressing 536 passages. Each question is associated with a paragraph (passage) extracted from an article. These passages are shorter than those in CNN and Who-did-What datasets. Models choose answers by selecting (varying-length) spans from these passages.

Generating Corrupt Data To void any information in either the questions or the passages, while otherwise leaving each architecture intact, we create corrupted versions of each dataset by assigning either questions randomly, while preserving the correspondence between passage and answer, or by randomizing the passage. For tasks where question-answering requires selecting spans or candidates from the passage, we create passages that contain the candidates in random locations but otherwise consist of random gibberish.

3 Models

In our investigations of the various RC benchmarks, we rely upon the following three recently-proposed models: key-value memory networks, gated attention readers, and QA nets. Although space constraints preclude a full discussion of each architecture, we provide references to the source papers and briefly discuss any implementation decisions necessary to reproduce our results.

Key-Value Memory Networks We implement a Key-Value Memory Network (KV-MemNet) (Miller et al., 2016), applying it to bAbI and CBT. KV-MemNets are based on Memory Networks (Sukhbaatar et al., 2015), shown to perform well on both datasets. For bAbI tasks, the keys and values both encode the passage as a bag-of-words (BoW). For CBT, the key is a BoW-encoded 5-word window surrounding a candidate answer and the value is the candidate itself. We fixed the number of hops to 3 and the embedding size to 128.

Gated Attention Reader Introduced by Dhingra et al. (2017), the Gated Attention Reader (GAR)[2] performs multiple hops over a passage, like MemNets. The word representations are refined over each hop and are mapped by an attention-sum module (Kadlec et al., 2016) to a probability distribution over the candidate answer set in the last hop. The model nearly matches best-reported results on many cloze-style RC datasets, and thus we apply it to *Who-did-What, CNN, CBT-NE* and *CBT-CN*.

QA Net Recently introduced by (Yu et al., 2018), the QA-Net[3] was recently demonstrated to outperform all previous models on the SQuAD dataset[4]. Passages and questions are passed as input to separate encoders consisting of depth-wise separable convolutions and global self-attention. This is followed by a passage-question attention layer, followed by stacked encoders. The outputs from these encoders are used to predict an answer span inside the passage.

4 Experimental Results

bAbI tasks Table 1 shows the results obtained by a Key-Value Memory Network on bAbI tasks by nullifying the information present in either questions or passages. On tasks 2, 7, 13 and 20, P-only models obtain over 80% accuracy with questions randomly assigned. Moreover, on tasks 3, 13, 16, and 20, P-only models match performance of those trained on the full dataset. On task 18, Q-only models achieve an accuracy of 91%, nearly matching the best performance of 93% achieved by the full model. These results show that some of bAbI tasks are easier than one might think.

Children's Books Test On the NE and CN CBT tasks, Q-only KV-MemNets obtain an accuracy close to the *full* accuracy and on the Verbs (V) and Prepositions (P) tasks, Q-only models outperform the full model (Table 2). Q-only Gated attention readers reach accuracy of 50.6% and 54% on Named Entities (NE) and Common Nouns (CN) tasks, respectively, while P-only models reach accuracies of 40.8% and 36.7%, respectively. We note that our models can outperform 16 of the 19 reported results on the NE task in Hill et al. (2016) using Q-only information. Table 3 shows that if we make use of just last sentence instead of all 20 sentences in the passage, our sentence memory based KV-MemNet achieve comparable or better performance *w.r.t* the *full* model on most subtasks.

CNN Table 2, shows the performance of Gated Attention Reader on the CNN dataset. Q-only and P-only models obtained 25.6% and 38.3% accuracies respectively, compared to 77.8% on the true dataset. This drop in accuracy could be due to the anonymization of entities which prevents models from building entity-specific information. Notwithstanding the deficiencies noted by Chen et al. (2016), we found that out CNN, out all the cloze-style RC datasets that we evaluated, appears to be the most carefully designed.

Who-did-What P-only models achieve greater than 50% accuracy in both the strict and relaxed setting, reaching within 15% of the accuracy of the *full* model in the strict setting. Q-only models also achieve 50% accuracy on the relaxed setting while achieving an accuracy of 41.8% on the strict setting. Our P-only model also outperforms all the

---

[2] https://github.com/bdhingra/ga-reader

[3] We use the implementation available at https://github.com/NLPLearn/QANet

[4] At the time of publication, an ensemble of QA-Net models was at the top of the leader board. A single QA-Net was ranked 4th.

	bAbI Tasks 1-10									
Dataset	1	2	3	4	5	6	7	8	9	10
True dataset	**100%**	**100%**	39%	**100%**	**99%**	**100%**	**94%**	**97%**	**99%**	**98%**
Question only	18%	17%	22%	22%	34%	50%	48%	34%	64%	44%
Passage only	53%	86%	**60%**	59%	31%	48%	85%	79%	63%	47%
$\Delta(min)$	−47	−14	+21	−41	−65	−52	−9	−18	−35	−51
	bAbI Tasks 11-20									
	11	12	13	14	15	16	17	18	19	20
True dataset	**94%**	**100%**	**94%**	**96%**	**100%**	**48%**	**57%**	**93%**	**30%**	**100%**
Question only	17%	15%	18%	18%	34%	26%	48%	91%	10%	70%
Passage only	71%	74%	**94%**	50%	64%	**47%**	48%	53%	21%	**100%**
$\Delta(min)$	−23	−26	0	−46	−36	−1	−9	−2	−9	0

Table 1: Accuracy on bAbI tasks using our implementation of the Key-Value Memory Networks

Task	Full	Q-only	P-only	$\Delta(min)$
Key-Value Memory Networks				
CBT-NE	**35.0%**	29.1%	24.1%	−5.9
CBT-CN	**37.6%**	32.4%	24.4%	−5.2
CBT-V	52.5%	**55.7%**	36.0%	+3.2
CBT-P	55.2%	**56.9%**	30.1%	+1.7
Gated Attention Reader				
CBT-NE	**74.9%**	50.6%	40.8%	−17.5
CBT-CN	**70.7%**	54.0%	36.7%	−16.7
CNN	**77.8%**	25.6%	38.3%	−39.5
WdW	**67.0%**	41.8%	52.2%	−14.8
WdW-R	**69.1%**	50.0%	50.6%	−15.6

Table 2: Accuracy on various datasets using KV-MemNets (window memory) and GARs

Task	Complete passage	Last sentence
CBT-NE	22.6%	**22.8%**
CBT-CN	**31.6%**	24.8%
CBT-V	**48.8%**	45.0%
CBT-P	34.1%	**37.9%**

Table 3: Accuracy on CBT tasks using KV-MemNets (sentence memory) varying passage size.

Metric	Full	Q-only	P-only	$\Delta(min)$
EM	**70.7%**	0.6%	10.9%	−59.8
F1	**79.1%**	4.0%	14.8%	−64.3

Table 4: Performance of QANet on SQuAD

suppressed baselines and 5 additional baselines reported by Onishi et al. (2016). We suspect that the models memorize attributes of specific entities, justifying the entity-anonymization used by Hermann et al. (2015) to construct the CNN dataset.

SQuAD Our results suggest that SQuAD is an unusually carefully-designed and challenging RC task. The span selection mode of answering requires that models consider the passage thus the abysmal performance of the Q-only QANet (Table 4). Since SQuAD requires answering by span selection, we construct Q-only variants here by placing answers from all relevant questions in random order, filling the gaps with random words. Moreover, Q-only and P-only models achieve F1 scores of only 4% and 14.8% resp. (Table 4), significantly lower than 79.1 on the proper task.

5 Discussion

We briefly discuss our findings, offer some guiding principles for evaluating new benchmarks and algorithms, and speculate on why some of these problems may have gone under the radar. Our goal is not to blame the creators of past datasets but instead to support the community by offering practical guidance for future researchers.

Provide rigorous RC baselines Published RC datasets should contain reasonable baselines that characterize the difficulty of the task, and specifically, the extent to which questions and passages are essential. Moreover, follow-up papers reporting improvements ought to report performance both on the full task and variations omitting questions and passages. While many proposed technical innovations purportedly work by better matching up information in questions and passages, absent these baselines one cannot tell whether gains come for the claimed reason or if the models just do a better job of passage classification (disregarding questions).

Test that *full context* is essential Even on tasks where both questions and passages are required, problems might appear harder than they really are. On first glance the the length-20 passages in CBT, might suggest that success requires reasoning over all 20 sentences to identify the correct answer to each question. However, it turns out that for some models, comparable performance can be achieved by considering only the last sentence. We recommend that researchers provide reasonable ablations to characterize the amount of context that each model truly requires.

Caution with cloze-style RC datasets We note that cloze-style datasets are often created programatically. Thus it's possible for a dataset to be produced, published, and incorporated into many downstream studies, all without many person-hours spent manually inspecting the data. We speculate that, as a result, these datasets tend be subject to less contemplation of what's involved in answering these questions and are therefore especially susceptible to the sorts of overlooked weaknesses described in our study.

A note on publishing incentives We express some concern that the recommended experimental rigor might cut against current publishing incentives. We speculate that papers introducing datasets may be more likely to be accepted at conferences by omitting unfavorable ablations than by including them. Moreover, with reviewers often demanding *architectural novelty*, methods papers may find an easier path to acceptance by providing unsubstantiated stories about the *reasons* why a given architecture works than by providing rigorous ablation studies stripping out spurious explanations and unnecessary model components. For more general discussions of misaligned incentives and empirical rigor in machine learning research, we point the interested reader to Lipton and Steinhardt (2018) and Sculley et al. (2018).

References

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Eric Breck, Marc Light, Gideon Mann, Ellen Riloff, Brianne Brown, and Pranav Anand. 2001. Looking under the hood: Tools for diagnosing your question answering engine. In *Association for Computational Linguistics (ACL) Workshop on Open-Domain Question Answering*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

Peter Clark and Oren Etzioni. 2016. My computer is an honor student but how intelligent is it? standardized tests as a measure of ai. *AI Magazine*, 37(1):5–12.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Association for Computational Linguistics (ACL)*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Association for Computational Linguistics (ACL)*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Computer Vision and Pattern Recognition (CVPR)*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The Goldilocks principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations (ICLR)*.

Lynette Hirschman, Marc Light, Eric Breck, and John D Burger. 1999. Deep read: A reading comprehension system. In *Association for Computational Linguistics on Computational Linguistics (ACL)*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Association for Computational Linguistics (ACL)*.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Association for Computational Linguistics (ACL)*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. 2016. Reasoning in vector space: An exploratory study of question answering. In *International Conference on Learning Representations (ICLR)*.

Zachary C Lipton and Jacob Steinhardt. 2018. Troubling trends in machine learning scholarship. In *International Conference on Machine Learning (ICML) Machine Learning Debates Workshop*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *Association for Computational Linguistics (ACL)*.

Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Richard Sutcliffe, Caroline Sporleder, Corina Forascu, Yassine Benajiba, and Petya Osenova. 2012. Overview of qa4mre at clef 2012: Question answering for machine reading evaluation.

Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Richard Sutcliffe, Corina Forascu, and Caroline Sporleder. 2011. Overview of qa4mre at clef 2011: Question answering for machine reading evaluation.

A. Poliak, J. Naradowsky, A. Haldar, R. Rudinger, and B. Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Joint Conference on Lexical and Computational Semantics (*SEM)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

D Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. 2018. Winner's curse? on pace, progress, and empirical rigor. In *International Conference on Learning Representations (ICLR) Workshop Track*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems (NIPS)*.

Richard Sutcliffe, Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Corina Forascu, Yassine Benajiba, and Petya Osenova. 2013. Overview of qa4mre main task at clef 2013.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Workshop on Representation Learning for NLP (RepL4NLP)*.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *International Conference on Learning Representations (ICLR)*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations (ICLR)*.

MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling

Paweł Budzianowski[1], **Tsung-Hsien Wen**[2*], **Bo-Hsiang Tseng**[1],
Iñigo Casanueva[1], **Stefan Ultes**[1], **Osman Ramadan**[1] **and Milica Gašić**[1]
[1]Department of Engineering, University of Cambridge, UK, [2]PolyAI, UK
`{pfb30,mg436}@cam.ac.uk`

Abstract

Even though machine learning has become the major scene in dialogue research community, the real breakthrough has been blocked by the scale of data available. To address this fundamental obstacle, we introduce the Multi-Domain Wizard-of-Oz dataset (MultiWOZ), a fully-labeled collection of human-human written conversations spanning over multiple domains and topics. At a size of 10k dialogues, it is at least one order of magnitude larger than all previous annotated task-oriented corpora. The contribution of this work apart from the open-sourced dataset labelled with dialogue belief states and dialogue actions is two-fold: firstly, a detailed description of the data collection procedure along with a summary of data structure and analysis is provided. The proposed data-collection pipeline is entirely based on crowd-sourcing without the need of hiring professional annotators; secondly, a set of benchmark results of belief tracking, dialogue act and response generation is reported, which shows the usability of the data and sets a baseline for future studies.

1 Introduction

Conversational Artificial Intelligence (Conversational AI) is one of the long-standing challenges in computer science and artificial intelligence since the Dartmouth Proposal (McCarthy et al., 1955). As human conversation is inherently complex and ambiguous, learning an open-domain conversational AI that can carry on arbitrary tasks is still very far-off (Vinyals and Le, 2015). As a consequence, instead of focusing on creating ambitious conversational agents that can reach human-level intelligence, industrial practice has focused on building task-oriented dialogue systems (Young et al., 2013) that can help with specific tasks such as flight reservation (Seneff and Polifroni, 2000) or bus information (Raux et al., 2005). As the need of hands-free use cases continues to grow, building a conversational agent that can handle tasks across different application domains has become more and more prominent (Ram et al., 2018).

Dialogues systems are inherently hard to build because there are several layers of complexity: the noise and uncertainty in speech recognition (Black et al., 2011); the ambiguity when understanding human language (Williams et al., 2013); the need to integrate third-party services and dialogue context in the decision-making (Traum and Larsson, 2003; Paek and Pieraccini, 2008); and finally, the ability to generate natural and engaging responses (Stent et al., 2005). These difficulties have led to the same solution of using statistical framework and machine learning for various system components, such as natural language understanding (Henderson et al., 2013; Mesnil et al., 2015; Mrkšić et al., 2017a), dialogue management (Gašić and Young, 2014; Tegho et al., 2018), language generation (Wen et al., 2015; Kiddon et al., 2016), and even end-to-end dialogue modelling (Zhao and Eskenazi, 2016; Wen et al., 2017; Eric et al., 2017).

To drive the progress of building dialogue systems using data-driven approaches, a number of conversational corpora have been released in the past. Based on whether a structured annotation scheme is used to label the semantics, these corpora can be roughly divided into two categories: corpora with structured semantic labels (Hemphill et al., 1990; Williams et al., 2013; Asri et al., 2017; Wen et al., 2017; Eric et al., 2017; Shah et al., 2018); and corpora without semantic labels but with an implicit user goal in mind (Ritter et al., 2010; Lowe et al., 2015). Despite these efforts, aforementioned datasets are usually constrained in one or more dimensions such as missing proper annotations, only available in a limited capacity, lacking multi-domain use cases, or having a negli-

---

[*] The work was done while at the University of Cambridge.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 5016–5026
Brussels, Belgium, October 31 - November 4, 2018.

Metric	**DSTC2**	**SFX**	**WOZ2.0**	**FRAMES**	**KVRET**	**M2M**	**MultiWOZ**
# Dialogues	1,612	1,006	600	1,369	2,425	1,500	**8,438**
Total # turns	23,354	12,396	4,472	19,986	12,732	14,796	**115,424**
Total # tokens	199,431	108,975	50,264	251,867	102,077	121,977	**1,520,970**
Avg. turns per dialogue	14.49	12.32	7.45	**14.60**	5.25	9.86	13.68
Avg. tokens per turn	8.54	8.79	11.24	12.60	8.02	8.24	**13.18**
Total unique tokens	986	1,473	2,142	12,043	2,842	1,008	**24,071**
# Slots	8	14	4	**61**	13	14	25
# Values	212	1847	99	3871	1363	138	**4510**

Table 1: Comparison of our corpus to similar data sets. Numbers in bold indicate best value for the respective metric. The numbers are provided for the training part of data except for FRAMES data-set were such division was not defined.

gible linguistic variability.

This paper introduces the Multi-Domain Wizard-of-Oz (MultiWOZ) dataset, a large-scale multi-turn conversational corpus with dialogues spanning across several domains and topics. Each dialogue is annotated with a sequence of dialogue states and corresponding system dialogue acts (Traum, 1999). Hence, MultiWOZ can be used to develop individual system modules as separate classification tasks and serve as a benchmark for existing modular-based approaches. On the other hand, MultiWOZ has around 10k dialogues, which is at least one order of magnitude larger than any structured corpus currently available. This significant size of the corpus allows researchers to carry on end-to-end based dialogue modelling experiments, which may facilitate a lot of exciting ongoing research in the area.

This work presents the data collection approach, a summary of the data structure, as well as a series of analyses of the data statistics. To show the potential and usefulness of the proposed MultiWOZ corpus, benchmarking baselines of belief tracking, natural language generation and end-to-end response generation have been conducted and reported. The dataset and baseline models will be freely available online.[1]

2 Related Work

Existing datasets can be roughly grouped into three categories: machine-to-machine, human-to-machine, and human-to-human conversations. A detailed review of these categories is presented below.

Machine-to-Machine Creating an environment with a simulated user enables to exhaustively generate dialogue templates. These templates can be mapped to a natural language by either pre-defined rules (Bordes et al., 2017) or crowd workers (Shah et al., 2018). Such approach ensures a diversity and full coverage of all possible dialogue outcomes within a certain domain. However, the naturalness of the dialogue flows relies entirely on the engineered set-up of the user and system bots. This poses a risk of a mismatch between training data and real interactions harming the interaction quality. Moreover, these datasets do not take into account noisy conditions often experienced in real interactions (Black et al., 2011).

Human-to-Machine Since collecting dialogue corpus for a task-specific application from scratch is difficult, most of the task-oriented dialogue corpora are fostered based on an existing dialogue system. One famous example of this kind is the Let's Go Bus Information System which offers live bus schedule information over the phone (Raux et al., 2005) leading to the first Dialogue State Tracking Challenge (Williams et al., 2013). Taking the idea of the Let's Go system forward, the second and third DSTCs (Henderson et al., 2014b,c) have produced bootstrapped human-machine datasets for a restaurant search domain in the Cambridge area, UK. Since then, DSTCs have become one of the central research topics in the dialogue community (Kim et al., 2016, 2017).

While human-to-machine data collection is an obvious solution for dialogue system develop-

---

[1] `http://dialogue.mi.eng.cam.ac.uk/index.php/corpus/`

ment, it is only possible with a provision of an existing working system. Therefore, this chicken (system)-and-egg (data) problem limits the use of this type of data collection to existing system improvement instead of developing systems in a completely new domain. What is even worse is that the capability of the initial system introduces additional biases to the collected data, which may result in a mismatch between the training and testing sets (Wen et al., 2016). The limited understanding capability of the initial system may prompt the users to adapt to simpler input examples that the system can understand but are not necessarily natural in conversations.

Human-to-Human Arguably, the best strategy to build a natural conversational system may be to have a system that can directly mimic human behaviors through learning from a large amount of real human-human conversations. With this idea in mind, several large-scale dialogue corpora have been released in the past, such as the Twitter (Ritter et al., 2010) dataset, the Reddit conversations (Schrading et al., 2015), and the Ubuntu technical support corpus (Lowe et al., 2015). Although previous work (Vinyals and Le, 2015) has shown that a large learning system can learn to generate interesting responses from these corpora, the lack of grounding conversations onto an existing knowledge base or APIs limits the usability of developed systems. Due to the lack of an explicit goal in the conversation, recent studies have shown that systems trained with this type of corpus not only struggle in generating consistent and diverse responses (Li et al., 2016) but are also extremely hard to evaluate (Liu et al., 2016).

In this paper, we focus on a particular type of human-to-human data collection. The Wizard-of-Oz framework (WOZ) (Kelley, 1984) was first proposed as an iterative approach to improve user experiences when designing a conversational system. The goal of WOZ data collection is to log down the conversation for future system development. One of the earliest dataset collected in this fashion is the ATIS corpus (Hemphill et al., 1990), where conversations between a client and an airline help-desk operator were recorded.

- You are traveling to Cambridge and looking forward to try local restaurants.
- You are looking for a **place to stay**. The hotel should be in the type of **hotel** and should be in the **centre**.
- The hotel should **include free wifi** and should have **a star of 4**.
- Once you find the **hotel** you want to book it for **3 people** and **5 nights** starting from **monday**.
- Make sure you get the **reference number**.
- You are also looking for a **restaurant**. The restaurant should serve **australasian** food and should be in the **moderate** price range.
- The restaurant should be **in the same area as the hotel**.
- If there is no such restaurant, how about one that serves **british** food.
- Once you find the **restaurant** you want to book a table for **the same group of people** at **18:30** on **the same day**.
- Make sure you get the **reference number**

Figure 1: A sample task template spanning over three domains - hotels, restaurants and booking.

More recently, Wen et al. (2017) have shown that the WOZ approach can be applied to collect high-quality typed conversations where a machine learning-based system can learn from. By modifying the original WOZ framework to make it suitable for crowd-sourcing, a total of 676 dialogues was collected via Amazon Mechanical Turk. The corpus was later extended to additional two languages for cross-lingual research (Mrkšić et al., 2017b). Subsequently, this approach is followed by Asri et al. (2017) to collect the Frame corpus in a more complex travel booking domain, and Eric et al. (2017) to collect a corpus of conversations for in-car navigation. Despite the fact that all these datasets contain highly natural conversations comparing to other human-machine collected datasets, they are usually small in size with only a limited domain coverage.

3 Data Collection Set-up

Following the Wizard-of-Oz set-up (Kelley, 1984), corpora of annotated dialogues can be gathered at relatively low costs and with a small time effort. This is in contrast to previous approaches (Henderson et al., 2014a) and such WOZ set-up has been successfully validated by Wen et al. (2017) and Asri et al. (2017).

Therefore, we follow the same process to create a large-scale corpus of natural human-human conversations. Our goal was to collect multi-domain dialogues. To overcome the need of relying the data collection to a small set of trusted workers[2], the collection set-up was designed to provide an

---

[2] Excluding annotation phase.

Table 2: Full ontology for all domains in our data-set. The upper script indicates which domains it belongs to. *: universal, 1: restaurant, 2: hotel, 3: attraction, 4: taxi, 5: train, 6: hospital, 7: police.

act type	inform* / request* / select^{123} / recommend/^{123} / not found^{123} request booking info^{123} / offer booking^{1235} / inform booked^{1235} / decline booking^{1235} welcome* /greet* / bye* / reqmore*
slots	address* / postcode* / phone* / name^{1234} / no of choices^{1235} / area^{123} / pricerange^{123} / type^{123} / internet^2 / parking^2 / stars^2 / open hours^3 / departure^{45} destination^{45} / leave after^{45} / arrive by^{45} / no of people^{1235} / reference no.^{1235} / trainID^5 / ticket price^5 / travel time^5 / department^7 / day^{1235} / no of days^{123}

easy-to-operate system interface for the Wizards and easy-to-follow goals for the users. This resulted in a bigger diversity and semantical richness of the collected data (see Section 4.3). Moreover, having a large set of workers mitigates the problem of artificial encouragement of a variety of behavior from users. A detailed explanation of the data-gathering process from both sides is provided below. Subsequently, we show how the crowd-sourcing scheme can also be employed to annotate the collected dialogues with dialogue acts.

3.1 Dialogue Task

The domain of a task-oriented dialogue system is often defined by an ontology, a structured representation of the back-end database. The ontology defines all entity attributes called slots and all possible values for each slot. In general, the slots may be divided into *informable* slots and *requestable* slots. *Informable* slots are attributes that allow the user to constrain the search (e.g., area or price range). *Requestable* slots represent additional information the users can request about a given entity (e.g., phone number). Based on a given ontology spanning several domains, a task template was created for each task through random sampling. This results in single and multi-domain dialogue scenarios and domain specific constraints were generated. In domains that allowed for that, an additional booking requirement was sampled with some probability.

To model more realistic conversations, goal changes are encouraged. With a certain probability, the initial constraints of a task may be set to values so that no matching database entry exists. Once informed about that situation by the system, the users only needed to follow the goal which provided alternative values.

3.2 User Side

To provide information to the users, each task template is mapped to natural language. Using heuristic rules, the task is then gradually introduced to the user to prevent an overflow of information. The goal description presented to the user is dependent on the number of turns already performed. Moreover, if the user is required to perform a sub-task (for example - booking a venue), these sub-goals are shown straight-away along with the main goal in the given domain. This makes the dialogues more similar to spoken conversations.[3] Figure 1 shows a sampled task description spanning over two domains with booking requirement. Natural incorporation of co-referencing and lexical entailment into the dialogue was achieved through implicit mentioning of some slots in the goal.

3.3 System Side

The wizard is asked to perform a role of a clerk by providing information required by the user. He is given an easy-to-operate graphical user interface to the back-end database. The wizard conveys the information provided by the current user input through a web form. This information is persistent across turns and is used to query the database. Thus, the annotation of a belief state is performed implicitly while the wizard is allowed to fully focus on providing the required information. Given the result of the query (a list of entities satisfying current constraints), the wizard either requests more details or provides the user with the adequate information. At each system turn, the wizard starts with the results of the query from the previous turn.

To ensure coherence and consistency, the wizard and the user alike first need to go through the

---

[3] However, the length of turns are significantly longer than with spoken interaction (Section 4.3).

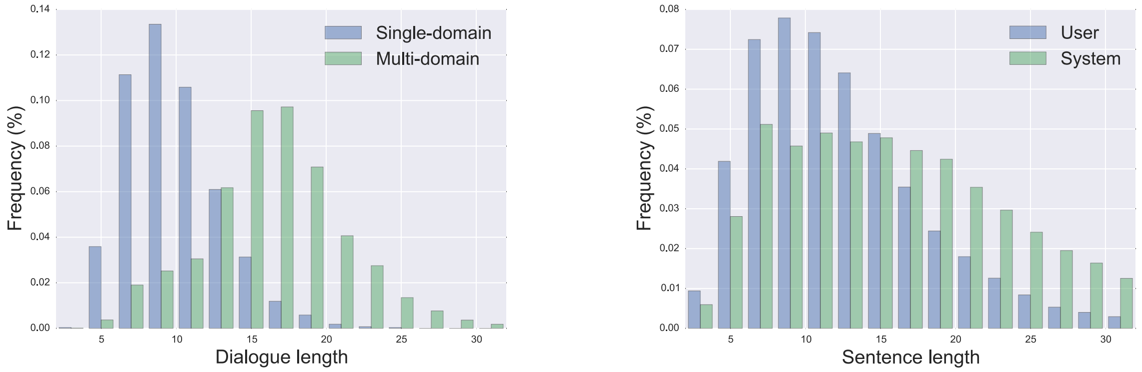

Figure 2: Dialogue length distribution (left) and distribution of number of tokens per turn (right).

dialogue history to establish the respective context. We found that even though multiple workers contributed to one dialogue, only a small margin of dialogues were incoherent.

3.4 Annotation of Dialogue Acts

Arguably, the most challenging and time-consuming part of any dialogue data collection is the process of annotating dialogue acts. One of the major challenges of this task is the definition of a set and structure of dialogue acts (Traum and Hinkelman, 1992; Bunt, 2006). In general, a dialogue act consists of the intent (such as request or inform) and slot-value pairs. For example, the act `inform(domain=hotel,price=expensive)` has the intent *inform*, where the user is informing the system to constrain the search to expensive hotels.

Expecting a big discrepancy in annotations between annotators, we initially ran three trial tests over a subset of dialogues using Amazon Mechanical Turk. Three annotations per dialogue were gathered resulting in around 750 turns. As this requires a multi-annotator metric over a multi-label task, we used Fleiss' kappa metric (Fleiss, 1971) per single dialogue act. Although the weighted kappa value averaged over dialogue acts was at a high level of 0.704, we have observed many cases of very poor annotations and an unsatisfactory coverage of dialogue acts. Initial errors in annotations and suggestions from crowd workers gradually helped us to expand and improve the final set of dialogue acts from 8 to 13 - see Table 2.

The variation in annotations made us change the initial approach. We ran a two-phase trial to first identify set of workers that perform well. Turkers were asked to annotate an illustrative, long dialogue which covered many problematic examples that we have observed in the initial run described above. All submissions that were of high quality were inspected and corrections were reported to annotators. Workers were asked to re-run a new trial dialogue. Having passed the second test, they were allowed to start annotating real dialogues. This procedure resulted in a restricted set of annotators performing high quality annotations. Appendix A contains a demonstration of a created system.

3.5 Data Quality

Data collection was performed in a two-step process. First, all dialogues were collected and then the annotation process was launched. This setup allowed the dialogue act annotators to also report errors (e.g., not following the task or confusing utterances) found in the collected dialogues. As a result, many errors could be corrected. Finally, additional tests were performed to ensure that the provided information in the dialogues match the pre-defined goals.

To estimate the inter-annotator agreement, the averaged weighted kappa value for all dialogue acts was computed over 291 turns. With $\kappa = 0.884$, an improvement in agreement between annotators was achieved although the size of action set was significantly larger.

4 MultiWOZ Dialogue Corpus

The main goal of the data collection was to acquire highly natural conversations between a tourist and a clerk from an information center in a touristic

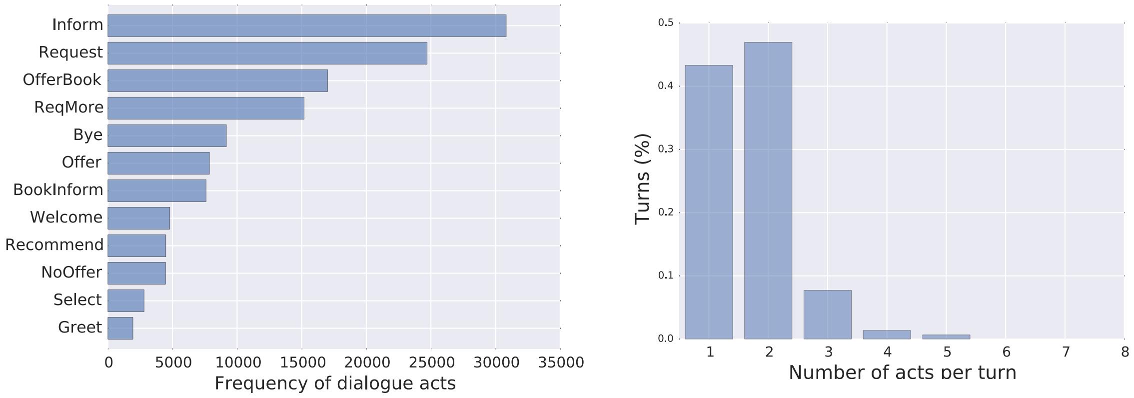

Figure 3: Dialogue acts frequency (left) and number of dialogue acts per turn (right) in the collected corpus.

city. We considered various possible dialogue scenarios ranging from requesting basic information about attractions through booking a hotel room or travelling between cities. In total, the presented corpus consists of 7 domains - *Attraction, Hospital, Police, Hotel, Restaurant, Taxi, Train*. The latter four are extended domains which include the sub-task *Booking*. Through a task sampling procedure (Section 3.1), the dialogues cover between 1 and 5 domains per dialogue thus greatly varying in length and complexity. This broad range of domains allows to create scenarios where domains are naturally connected. For example, a tourist needs to find a hotel, to get the list of attractions and to book a taxi to travel between both places. Table 2 presents the global ontology with the list of considered dialogue acts.

4.1 Data Statistics

Following data collection process from the previous section, a total of $10,438$ dialogues were collected. Figure 2 (left) shows the dialogue length distribution grouped by single and multi domain dialogues. Around 70% of dialogues have more than 10 turns which shows the complexity of the corpus. The average number of turns are 8.93 and 15.39 for single and multi-domain dialogues respectively with $115,434$ turns in total. Figure 2 (right) presents a distribution over the turn lengths. As expected, the wizard replies are much longer - the average sentence lengths are 11.75 and 15.12 for users and wizards respectively. The responses are also more diverse thus enabling the training of more complex generation models.

Figure 3 (left) shows the distribution of dialogue acts annotated in the corpus. We present here a summarized list where different types of actions like *inform* are grouped together. The right graph in the Figure 3 presents the distribution of number of acts per turn. Almost 60% of dialogues turns have more than one dialogue act showing again the richness of system utterances. These create a new challenge for reinforcement learning-based models requiring them to operate on concurrent actions.

In total, $1,249$ workers contributed to the corpus creation with only few instances of intentional wrongdoing. Additional restrictions were added to automatically discover instances of very short utterances, short dialogues or missing single turns during annotations. All such cases were corrected or deleted from the corpus.

4.2 Data Structure

There are $3,406$ single-domain dialogues that include booking if the domain allows for that and $7,032$ multi-domain dialogues consisting of at least 2 up to 5 domains. To enforce reproducibility of results, the corpus was randomly split into a train, test and development set. The test and development sets contain 1k examples each. Even though all dialogues are coherent, some of them were not finished in terms of task description. Therefore, the validation and test sets only contain fully successful dialogues thus enabling a fair comparison of models.

Each dialogue consists of a goal, multiple user and system utterances as well as a belief state and set of dialogue acts with slots per turn. Additionally, the task description in natural language presented to turkers working from the visitor's side is added.

4.3 Comparison to Other Structured Corpora

To illustrate the contribution of the new corpus, we compare it on several important statistics with the DSTC2 corpus (Henderson et al., 2014a), the SFX corpus (Gašić et al., 2014), the WOZ2.0 corpus (Wen et al., 2017), the FRAMES corpus (Asri et al., 2017), the KVRET corpus (Eric et al., 2017), and the M2M corpus (Shah et al., 2018). Figure 1 clearly shows that our corpus compares favorably to all other data sets on most of the metrics with the number of total dialogues, the average number of tokens per turn and the total number of unique tokens as the most prominent ones. Especially the latter is important as it is directly linked to linguistic richness.

5 MultiWOZ as a New Benchmark

The complexity and the rich linguistic variation in the collected MultiWOZ dataset makes it a great benchmark for a range of dialogue tasks. To show the potential usefulness of the MultiWOZ corpus, we break down the dialogue modelling task into three sub-tasks and report a benchmark result for each of them: dialogue state tracking, dialogue-act-to-text generation, and dialogue-context-to-text generation. These results illustrate new challenges introduced by the MultiWOZ dataset for different dialogue modelling problems.

5.1 Dialogue State Tracking

A robust natural language understanding and dialogue state tracking is the first step towards building a good conversational system. Since multi-domain dialogue state tracking is still in its infancy and there are not many comparable approaches available (Rastogi et al., 2017), we instead report our state-of-the-art result on the restaurant subset of the MultiWOZ corpus as the reference baseline. The proposed method (Ramadan et al., 2018) exploits the semantic similarity between dialogue utterances and the ontology terms which allows the information to be shared across domains. Furthermore, the model parameters are independent of the ontology and belief states, therefore the number of the parameters does not increase with the size of the domain itself.[4]

Slot	WOZ 2.0	MultiWOZ (restaurant)
Overall accuracy	96.5	89.7
Joint goals	85.5	80.9

Table 3: The test set accuracies *overall* and for *joint goals* in the restaurant sub-domain.

The same model was trained on both the WOZ2.0 and the proposed MultiWOZ datasets, where the WOZ2.0 corpus consists of 1200 single domain dialogues in the restaurant domain. Although not directly comparable, Table 3 shows that the performance of the model is consecutively poorer on the new dataset compared to WOZ2.0. These results demonstrate how demanding is the new dataset as the conversations are richer and much longer.

5.2 Dialogue-Context-to-Text Generation

After a robust dialogue state tracking module is built, the next challenge becomes the dialogue management and response generation components. These problems can either be addressed separately (Young et al., 2013), or jointly in an end-to-end fashion (Bordes et al., 2017; Wen et al., 2017; Li et al., 2017). In order to establish a clear benchmark where the performance of the composite of dialogue management and response generation is completely independent of the belief tracking, we experimented with a baseline neural response generation model with an *oracle* belief-state obtained from the wizard annotations as discussed in Section 3.3.[5]

Following Wen et al. (2017) which frames the dialogue as a context to response mapping problem, a sequence-to-sequence model (Sutskever et al., 2014) is augmented with a belief tracker and a discrete database accessing component as additional features to inform the word decisions in the decoder. Note, in the original paper the belief tracker was pre-trained while in this work the annotations of the dialogue state are used as an oracle tracker. Figure 4 presents the architecture of the system (Budzianowski et al., 2018).

---

[4] The model is publicly available at `https://github.com/osmanio2/multi-domain-belief-tracking`

[5] The model is publicly available at `https://github.com/budzianowski/multiwoz`

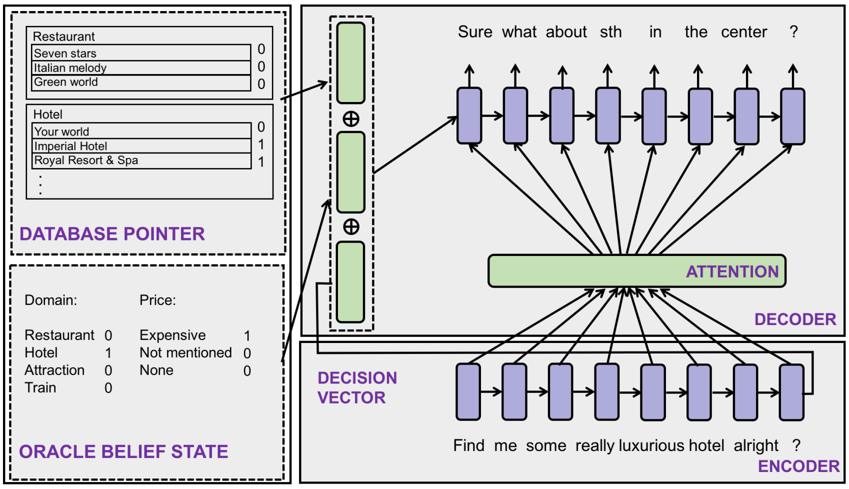

Figure 4: Architecture of the multi-domain response generator. The attention is conditioned on the oracle belief state and the database pointer.

Training and Evaluation Since often times the evaluation of a dialogue system without a direct interaction with the real users can be misleading (Liu et al., 2016), three different automatic metrics are included to ensure the result is better interpreted. Among them, the first two metrics relate to the dialogue task completion - whether the system has provided an appropriate entity (Inform rate) and then answered all the requested attributes (Success rate); while fluency is measured via BLEU score (Papineni et al., 2002). The best models for both datasets were found through a grid search over a set of hyper-parameters such as the size of embeddings, learning rate and different recurrent architectures.

We trained the same neural architecture (taking into account different number of domains) on both MultiWOZ and Cam676 datasets. The best results on the Cam676 corpus were obtained with bidirectional GRU cell. In the case of MultiWOZ dataset, the LSTM cell serving as a decoder and an encoder achieved the highest score with the global type of attention (Bahdanau et al., 2014). Table 4 presents the results of a various of model architectures and shows several challenges. As expected, the model achieves almost perfect score on the Inform metric on the Cam676 dataset taking the advantage of an oracle belief state signal. However, even with the perfect dialogue state tracking of the user intent, the baseline models obtain almost 30% lower score on the Inform metric on the new corpus. The addition of the attention improves the score on the Success metric on the new dataset by less than 1%. Nevertheless, as expected, the best model on MultiWOZ is still falling behind by a large margin in comparison to the results on the Cam676 corpus taking into account both Inform and Success metrics. As most of dialogues span over at least two domains, the model has to be much more effective in order to execute a successful dialogue. Moreover, the BLEU score on the MultiWOZ is lower than the one reported on the Cam676 dataset. This is mainly caused by the much more diverse linguistic expressions observed in the MultiWOZ dataset.

5.3 Dialogue-Act-to-Text Generation

Natural Language Generation from a structured meaning representation (Oh and Rudnicky, 2000; Bohus and Rudnicky, 2005) has been a very popular research topic in the community, and the lack of data has been a long standing block for the field to adopt more machine learning methods. Due to the additional annotation of the system acts, the MultiWOZ dataset serves as a new benchmark for studying natural language generation from a structured meaning representation. In order to verify the difficulty of the collected dataset for the language generation task, we compare it to the SFX dataset (see Table 1), which consists of around 5k dialogue act and natural language sentence pairs. We trained the same Semantically Conditioned Long Short-term Memory network (SC-LSTM) proposed by Wen et al. (2015) on both datasets and used the metrics as a proxy to estimate the difficulty of the two corpora. To make a fair comparison, we constrained our dataset to only the restaurant sub-domain which contains around 25k dia-

	Cam676		MultiWOZ	
	w/o attention	w/ attention	w/o attention	w/ attention
Inform (%)	99.17	99.58	71.29	71.33
Success (%)	75.08	73.75	60.29	60.96
BLEU	0.219	0.204	0.188	0.189

Table 4: Performance comparison of two different model architectures using a corpus-based evaluation.

logue turns. To give more statistics about the two datasets: the SFX corpus has 9 different act types with 12 slots comparing to 12 acts and 14 slots in our corpus. The best model for both datasets was found through a grid search over a set of hyper-parameters such as the size of embeddings, learning rate, and number of LSTM layers.[6]

Table 5 presents the results on two metrics: BLEU score (Papineni et al., 2002) and slot error rate (SER) (Wen et al., 2015). The significantly lower metrics on the MultiWOZ corpus showed that it is much more challenging than the SFX restaurant dataset. This is probably due to the fact that more than 60% of the dialogue turns are composed of at least two system acts, which greatly harms the performance of the existing model.

Metric	SFX	MultiWOZ (restaurant)
SER (%)	0.46	4.378
BLEU	0.731	0.616

Table 5: The test set slot error rate (SER) and BLEU on the SFX dataset and the MultiWOZ restaurant subset.

6 Conclusions

As more and more speech oriented applications are commercially deployed, the necessity of building an entirely data-driven conversational agent becomes more apparent. Various corpora were gathered to enable data-driven approaches to dialogue modelling. To date, however, the available datasets were usually constrained in linguistic variability or lacking multi-domain use cases. In this paper, we established a data-collection pipeline entirely based on crowd-sourcing enabling to gather a large scale, linguistically rich corpus of human-human conversations. We hope that MultiWOZ offers valuable training data and a new challenging testbed for existing modular-based approaches ranging from belief tracking to dialogue acts generation. Moreover, the scale of the data should help push forward research in the end-to-end dialogue modelling.

Acknowledgments

This work was funded by a Google Faculty Research Award (RG91111), an EPSRC studentship (RG80792), an EPSRC grant (EP/M018946/1) and by Toshiba Research Europe Ltd, Cambridge Research Laboratory (RG85875). The authors thank many excellent Mechanical Turk contributors for building this dataset. The authors would also like to thank Thang Minh Luong for his support for this project ad Nikola Mrkšić and anonymous reviewers for their constructive feedback. The data is available at `http://dialogue.mi.eng.cam.ac.uk/index.php/corpus/`.

References

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *Proceedings of SigDial*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ICLR*.

Alan W Black, Susanne Burger, Alistair Conkie, Helen Hastie, Simon Keizer, Oliver Lemon, Nicolas Merigaud, Gabriel Parent, Gabriel Schubiner, Blaise Thomson, et al. 2011. Spoken dialog challenge 2010: Comparison of live and control test results. In *Proceedings of the SIGDIAL 2011 Conference*, pages 2–7. Association for Computational Linguistics.

Dan Bohus and Alexander I Rudnicky. 2005. Sorry, i didn't catch that! - an investigation of non-understanding errors and recovery strategies. In *6th SIGdial workshop on discourse and dialogue*.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *Proceedings of ICLR*.

---

[6] The model is publicly available at `https://github.com/andy194673/nlg-sclstm-multiwoz`

Paweł Budzianowski, Iñigo Casanueva, Bo-Hsiang Tseng, and Milica Gašić. 2018. Towards end-to-end multi-domain dialogue modelling. *Tech. Rep. CUED/F-INFENG/TR.706, University of Cambridge, Engineering Department*.

Harry Bunt. 2006. Dimensions in dialogue act annotation. In *Proc. of LREC*, volume 6, pages 919–924.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Milica Gašić, Dongho Kim, Pirros Tsiakoulis, Catherine Breslin, Matthew Henderson, Martin Szummer, Blaise Thomson, and Steve Young. 2014. Incremental on-line adaptation of pomdp-based dialogue managers to extended domains. In *Interspeech*.

Milica Gašić and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *TASLP*, 22(1):28–40.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania*.

M. Henderson, B. Thomson, and J. Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of SIGdial*.

M. Henderson, B. Thomson, and S. J. Young. 2014b. Word-based Dialog State Tracking with Recurrent Neural Networks. In *Proceedings of SIGdial*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014c. The third dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 324–329. IEEE.

Matthew Henderson, Blaise Thomson, and Steve Young. 2013. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471.

John F Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E Banchs, Jason D Williams, Matthew Henderson, and Koichiro Yoshino. 2016. The fifth dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 511–517. IEEE.

Seokhwan Kim, Luis Fernando DHaro, Rafael E Banchs, Jason D Williams, and Matthew Henderson. 2017. The fourth dialog state tracking challenge. In *Dialogues with Social Robots*, pages 435–449. Springer.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL-HLT*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 733–743.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 285.

J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon. 1955. A proposal for the dartmouth summer research project on artificial intelligence.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017a. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1777–1788.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017b. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the Association of Computational Linguistics*, 5(1):309–324.

Alice H Oh and Alexander I Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 27–32. Association for Computational Linguistics.

Tim Paek and Roberto Pieraccini. 2008. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech communication*, 50(8-9):716–729.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. 2018. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*.

Osman Ramadan, Paweł Budzianowski, and Milica Gašić. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 432–437.

Abhinav Rastogi, Dilek Hakkani-Tur, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. *arXiv preprint arXiv:1712.10224*.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Let's go public! taking a spoken dialog system to the real world. In *Ninth European Conference on Speech Communication and Technology*.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180.

Nicolas Schrading, Cecilia Ovesdotter Alm, Ray Ptucha, and Christopher Homan. 2015. An analysis of domestic abuse discourse on reddit. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2577–2583.

Stephanie Seneff and Joseph Polifroni. 2000. Dialogue management in the mercury flight reservation system. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational Systems - Volume 3*, ANLP/NAACL-ConvSyst '00, pages 11–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

P Shah, D Hakkani-Tur, G Tur, A Rastogi, A Bapna, N Nayak, and L Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 341–351. Springer.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Christopher Tegho, PawełBudzianowski, and Milica Gašić. 2018. Benchmarking uncertainty estimates with deep reinforcement learning for dialogue policy optimisation. In *IEEE ICASSP*.

David R. Traum. 1999. *Foundations of Rational Agency*, chapter Speech Acts for Dialogue Agents. Springer.

David R Traum and Elizabeth A Hinkelman. 1992. Conversation acts in task-oriented spoken dialogue. *Computational intelligence*, 8(3):575–599.

David R Traum and Staffan Larsson. 2003. The information state approach to dialogue management. In *Current and new directions in discourse and dialogue*, pages 325–353. Springer.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. *ACL*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gašić, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. *EACL*.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Steve Young, Milica Gašić, Blaise Thomson, and Jason Williams. 2013. POMDP-based Statistical Spoken Dialogue Systems: a Review. In *Proc of IEEE*, volume 99, pages 1–20.

Linguistically-Informed Self-Attention for Semantic Role Labeling

Emma Strubell[1], **Patrick Verga**[1], **Daniel Andor**[2], **David Weiss**[2] **and Andrew McCallum**[1]
[1]College of Information and Computer Sciences
University of Massachusetts Amherst
{strubell, pat, mccallum}@cs.umass.edu
[2]Google AI Language
New York, NY
{andor, djweiss}@google.com

Abstract

Current state-of-the-art semantic role labeling (SRL) uses a deep neural network with no explicit linguistic features. However, prior work has shown that gold syntax trees can dramatically improve SRL decoding, suggesting the possibility of increased accuracy from explicit modeling of syntax. In this work, we present linguistically-informed self-attention (LISA): a neural network model that combines multi-head self-attention with multi-task learning across dependency parsing, part-of-speech tagging, predicate detection and SRL. Unlike previous models which require significant pre-processing to prepare linguistic features, LISA can incorporate syntax using merely raw tokens as input, encoding the sequence only once to simultaneously perform parsing, predicate detection and role labeling for all predicates. Syntax is incorporated by training one attention head to attend to syntactic parents for each token. Moreover, if a high-quality syntactic parse is already available, it can be beneficially injected at test time without re-training our SRL model. In experiments on CoNLL-2005 SRL, LISA achieves new state-of-the-art performance for a model using predicted predicates and standard word embeddings, attaining 2.5 F1 absolute higher than the previous state-of-the-art on newswire and more than 3.5 F1 on out-of-domain data, nearly 10% reduction in error. On ConLL-2012 English SRL we also show an improvement of more than 2.5 F1. LISA also out-performs the state-of-the-art with contextually-encoded (ELMo) word representations, by nearly 1.0 F1 on news and more than 2.0 F1 on out-of-domain text.

1 Introduction

Semantic role labeling (SRL) extracts a high-level representation of meaning from a sentence, labeling e.g. *who* did *what* to *whom*. Explicit representations of such semantic information have been shown to improve results in challenging downstream tasks such as dialog systems (Tur et al., 2005; Chen et al., 2013), machine reading (Berant et al., 2014; Wang et al., 2015) and translation (Liu and Gildea, 2010; Bazrafshan and Gildea, 2013).

Though syntax was long considered an obvious prerequisite for SRL systems (Levin, 1993; Punyakanok et al., 2008), recently deep neural network architectures have surpassed syntactically-informed models (Zhou and Xu, 2015; Marcheggiani et al., 2017; He et al., 2017; Tan et al., 2018; He et al., 2018), achieving state-of-the art SRL performance with no explicit modeling of syntax. An additional benefit of these end-to-end models is that they require just raw tokens and (usually) detected predicates as input, whereas richer linguistic features typically require extraction by an auxiliary pipeline of models.

Still, recent work (Roth and Lapata, 2016; He et al., 2017; Marcheggiani and Titov, 2017) indicates that neural network models could see even higher accuracy gains by leveraging syntactic information rather than ignoring it. He et al. (2017) indicate that many of the errors made by a syntax-free neural network on SRL are tied to certain syntactic confusions such as prepositional phrase attachment, and show that while constrained inference using a relatively low-accuracy predicted parse can provide small improvements in SRL accuracy, providing a gold-quality parse leads to substantial gains. Marcheggiani and Titov (2017) incorporate syntax from a high-quality parser (Kiperwasser and Goldberg, 2016) using graph convolutional neural networks (Kipf and Welling, 2017), but like He et al. (2017) they attain only small increases over a model with no syntactic parse, and even perform worse than a syntax-free model on out-of-domain data. These works suggest that though syntax has the potential to improve neural network SRL models, we have not

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 5027–5038
Brussels, Belgium, October 31 - November 4, 2018.

yet designed an architecture which maximizes the benefits of auxiliary syntactic information.

In response, we propose *linguistically-informed self-attention* (LISA): a model that combines multi-task learning (Caruana, 1993) with stacked layers of multi-head self-attention (Vaswani et al., 2017); the model is trained to: (1) jointly predict parts of speech and predicates; (2) perform parsing; and (3) attend to syntactic parse parents, while (4) assigning semantic role labels. Whereas prior work typically requires separate models to provide linguistic analysis, including most syntax-free neural models which still rely on external predicate detection, our model is truly end-to-end: earlier layers are trained to predict prerequisite parts-of-speech and predicates, the latter of which are supplied to later layers for scoring. Though prior work re-encodes each sentence to predict each desired task and again with respect to each predicate to perform SRL, we more efficiently encode each sentence only once, predict its predicates, part-of-speech tags and labeled syntactic parse, then predict the semantic roles for all predicates in the sentence in parallel. The model is trained such that, as syntactic parsing models improve, providing high-quality parses at test time will improve its performance, allowing the model to leverage updated parsing models without requiring re-training.

In experiments on the CoNLL-2005 and CoNLL-2012 datasets we show that our linguistically-informed models out-perform the syntax-free state-of-the-art. On CoNLL-2005 with predicted predicates and standard word embeddings, our single model out-performs the previous state-of-the-art model on the WSJ test set by 2.5 F1 points absolute. On the challenging out-of-domain Brown test set, our model improves substantially over the previous state-of-the-art by more than 3.5 F1, a nearly 10% reduction in error. On CoNLL-2012, our model gains more than 2.5 F1 absolute over the previous state-of-the-art. Our models also show improvements when using contextually-encoded word representations (Peters et al., 2018), obtaining nearly 1.0 F1 higher than the state-of-the-art on CoNLL-2005 news and more than 2.0 F1 improvement on out-of-domain text.[1]

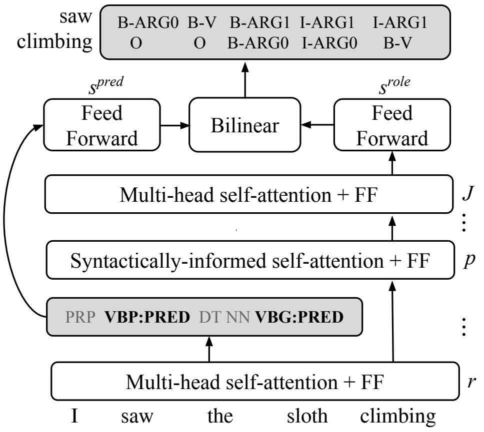

Figure 1: Word embeddings are input to J layers of multi-head self-attention. In layer p one attention head is trained to attend to parse parents (Figure 2). Layer r is input for a joint predicate/POS classifier. Representations from layer r corresponding to predicted predicates are passed to a bilinear operation scoring distinct predicate and role representations to produce per-token SRL predictions with respect to each predicted predicate.

2 Model

Our goal is to design an efficient neural network model which makes use of linguistic information as effectively as possible in order to perform end-to-end SRL. LISA achieves this by combining: (1) A new technique of supervising neural attention to predict syntactic dependencies with (2) multi-task learning across four related tasks.

Figure 1 depicts the overall architecture of our model. The basis for our model is the Transformer encoder introduced by Vaswani et al. (2017): we transform word embeddings into contextually-encoded token representations using stacked multi-head self-attention and feed-forward layers (§2.1).

To incorporate syntax, one self-attention head is trained to attend to each token's syntactic parent, allowing the model to use this attention head as an oracle for syntactic dependencies. We introduce this *syntactically-informed self-attention* (Figure 2) in more detail in §2.2.

Our model is designed for the more realistic setting in which gold predicates are not provided at test-time. Our model predicts predicates and integrates part-of-speech (POS) information into earlier layers by re-purposing representations closer to the input to predict predicate and POS tags us-

---

[1] Our implementation in TensorFlow (Abadi et al., 2015) is available at : `http://github.com/strubell/LISA`

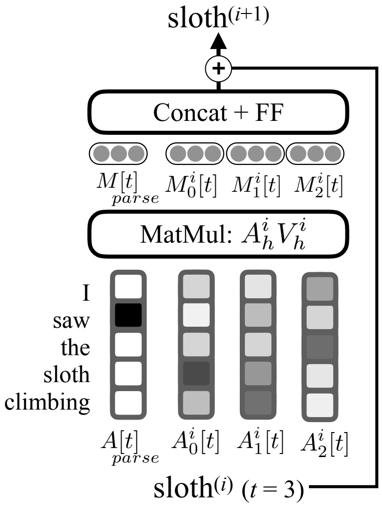

Figure 2: Syntactically-informed self-attention for the query word *sloth*. Attention weights A_{parse} heavily weight the token's syntactic governor, *saw*, in a weighted average over the token values V_{parse}. The other attention heads act as usual, and the attended representations from all heads are concatenated and projected through a feed-forward layer to produce the syntactically-informed representation for *sloth*.

ing hard parameter sharing (§2.3). We simplify optimization and benefit from shared statistical strength derived from highly correlated POS and predicates by treating tagging and predicate detection as a single task, performing multi-class classification into the joint Cartesian product space of POS and predicate labels.

Though typical models, which re-encode the sentence for each predicate, can simplify SRL to token-wise tagging, our joint model requires a different approach to classify roles with respect to each predicate. Contextually encoded tokens are projected to distinct *predicate* and *role* embeddings (§2.4), and each predicted predicate is scored with the sequence's role representations using a bilinear model (Eqn. 6), producing per-label scores for BIO-encoded semantic role labels for each token and each semantic frame.

The model is trained end-to-end by maximum likelihood using stochastic gradient descent (§2.5).

2.1 Self-attention token encoder

The basis for our model is a multi-head self-attention token encoder, recently shown to achieve state-of-the-art performance on SRL (Tan et al., 2018), and which provides a natural mechanism for incorporating syntax, as described in §2.2. Our implementation replicates Vaswani et al. (2017).

The input to the network is a sequence $\mathcal{X}$ of T token representations x_t. In the standard setting these token representations are initialized to pre-trained word embeddings, but we also experiment with supplying pre-trained ELMo representations combined with task-specific learned parameters, which have been shown to substantially improve performance of other SRL models (Peters et al., 2018). For experiments with gold predicates, we concatenate a predicate indicator embedding p_t following previous work (He et al., 2017).

We project[2] these input embeddings to a representation that is the same size as the output of the self-attention layers. We then add a positional encoding vector computed as a deterministic sinusoidal function of t, since the self-attention has no innate notion of token position.

We feed this token representation as input to a series of J residual multi-head self-attention layers with feed-forward connections. Denoting the jth self-attention layer as $T^{(j)}(\cdot)$, the output of that layer $s_t^{(j)}$, and $LN(\cdot)$ layer normalization, the following recurrence applied to initial input $c_t^{(p)}$:

$$s_t^{(j)} = LN(s_t^{(j-1)} + T^{(j)}(s_t^{(j-1)})) \qquad (1)$$

gives our final token representations $s_t^{(j)}$. Each $T^{(j)}(\cdot)$ consists of: (a) multi-head self-attention and (b) a feed-forward projection.

The multi-head self attention consists of H attention heads, each of which learns a distinct attention function to attend to all of the tokens in the sequence. This self-attention is performed for each token for each head, and the results of the H self-attentions are concatenated to form the final self-attended representation for each token.

Specifically, consider the matrix $S^{(j-1)}$ of T token representations at layer $j - 1$. For each attention head h, we project this matrix into distinct key, value and query representations $K_h^{(j)}$, $V_h^{(j)}$ and $Q_h^{(j)}$ of dimensions $T \times d_k$, $T \times d_q$, and $T \times d_v$, respectively. We can then multiply $Q_h^{(j)}$ by $K_h^{(j)}$ to obtain a $T \times T$ matrix of attention weights $A_h^{(j)}$ between each pair of tokens in the sentence. Following Vaswani et al. (2017) we perform scaled dot-product attention: We scale the weights by the inverse square root of their embedding dimension

---

[2] All linear projections include bias terms, which we omit in this exposition for the sake of clarity.

and normalize with the softmax function to produce a distinct distribution for each token over all the tokens in the sentence:

$$A_h^{(j)} = \text{softmax}(d_k^{-0.5} Q_h^{(j)} {K_h^{(j)}}^T) \qquad (2)$$

These attention weights are then multiplied by $V_h^{(j)}$ for each token to obtain the self-attended token representations $M_h^{(j)}$:

$$M_h^{(j)} = A_h^{(j)} V_h^{(j)} \qquad (3)$$

Row t of $M_h^{(j)}$, the self-attended representation for token t at layer j, is thus the weighted sum with respect to t (with weights given by $A_h^{(j)}$) over the token representations in $V_h^{(j)}$.

The outputs of all attention heads for each token are concatenated, and this representation is passed to the feed-forward layer, which consists of two linear projections each followed by leaky ReLU activations (Maas et al., 2013). We add the output of the feed-forward to the initial representation and apply layer normalization to give the final output of self-attention layer j, as in Eqn. 1.

2.2 Syntactically-informed self-attention

Typically, neural attention mechanisms are left on their own to learn to attend to relevant inputs. Instead, we propose training the self-attention to attend to specific tokens corresponding to the syntactic structure of the sentence as a mechanism for passing linguistic knowledge to later layers.

Specifically, we replace one attention head with the deep bi-affine model of Dozat and Manning (2017), trained to predict syntactic dependencies. Let A_{parse} be the parse attention weights, at layer i. Its input is the matrix of token representations $S^{(i-1)}$. As with the other attention heads, we project $S^{(i-1)}$ into key, value and query representations, denoted K_{parse}, Q_{parse}, V_{parse}. Here the key and query projections correspond to $parent$ and $dependent$ representations of the tokens, and we allow their dimensions to differ from the rest of the attention heads to more closely follow the implementation of Dozat and Manning (2017). Unlike the other attention heads which use a dot product to score key-query pairs, we score the compatibility between K_{parse} and Q_{parse} using a bi-affine operator U_{heads} to obtain attention weights:

$$A_{parse} = \text{softmax}(Q_{parse} U_{heads} K_{parse}^T) \qquad (4)$$

These attention weights are used to compose a weighted average of the value representations V_{parse} as in the other attention heads.

We apply auxiliary supervision at this attention head to encourage it to attend to each token's parent in a syntactic dependency tree, and to encode information about the token's dependency label. Denoting the attention weight from token t to a candidate head q as $A_{parse}[t, q]$, we model the probability of token t having parent q as:

$$P(q = \text{head}(t) \mid \mathcal{X}) = A_{parse}[t, q] \qquad (5)$$

using the attention weights $A_{parse}[t]$ as the distribution over possible heads for token t. We define the root token as having a self-loop. This attention head thus emits a directed graph[3] where each token's parent is the token to which the attention A_{parse} assigns the highest weight.

We also predict dependency labels using per-class bi-affine operations between parent and dependent representations Q_{parse} and K_{parse} to produce per-label scores, with locally normalized probabilities over dependency labels y_t^{dep} given by the softmax function. We refer the reader to Dozat and Manning (2017) for more details.

This attention head now becomes an oracle for syntax, denoted $\mathcal{P}$, providing a dependency parse to downstream layers. This model not only predicts its own dependency arcs, but allows for the injection of auxiliary parse information at test time by simply setting A_{parse} to the parse parents produced by e.g. a state-of-the-art parser. In this way, our model can benefit from improved, external parsing models without re-training. Unlike typical multi-task models, ours maintains the ability to leverage external syntactic information.

2.3 Multi-task learning

We also share the parameters of lower layers in our model to predict POS tags and predicates. Following He et al. (2017), we focus on the end-to-end setting, where predicates must be predicted on-the-fly. Since we also train our model to predict syntactic dependencies, it is beneficial to give the model knowledge of POS information. While much previous work employs a pipelined approach to both POS tagging for dependency parsing and predicate detection for SRL, we take a multi-task learning (MTL) approach (Caruana,

---

[3] Usually the head emits a tree, but we do not enforce it here.

1993), sharing the parameters of earlier layers in our SRL model with a joint POS and predicate detection objective. Since POS is a strong predictor of predicates[4] and the complexity of training a multi-task model increases with the number of tasks, we combine POS tagging and predicate detection into a joint label space: For each POS tag TAG which is observed co-occurring with a predicate, we add a label of the form TAG:PREDICATE.

Specifically, we feed the representation $s_t^{(r)}$ from a layer r preceding the syntactically-informed layer p to a linear classifier to produce per-class scores r_t for token t. We compute locally-normalized probabilities using the softmax function: $P(y_t^{prp} \mid \mathcal{X}) \propto \exp(r_t)$, where y_t^{prp} is a label in the joint space.

2.4 Predicting semantic roles

Our final goal is to predict semantic roles for each predicate in the sequence. We score each predicate against each token in the sequence using a bilinear operation, producing per-label scores for each token for each predicate, with predicates and syntax determined by oracles $\mathcal{V}$ and $\mathcal{P}$.

First, we project each token representation $s_t^{(J)}$ to a predicate-specific representation s_t^{pred} and a role-specific representation s_t^{role}. We then provide these representations to a bilinear transformation U for scoring. So, the role label scores s_{ft} for the token at index t with respect to the predicate at index f (i.e. token t and frame f) are given by:

$$s_{ft} = (s_f^{pred})^T U s_t^{role} \tag{6}$$

which can be computed in parallel across all semantic frames in an entire minibatch. We calculate a locally normalized distribution over role labels for token t in frame f using the softmax function: $P(y_{ft}^{role} \mid \mathcal{P}, \mathcal{V}, \mathcal{X}) \propto \exp(s_{ft})$.

At test time, we perform constrained decoding using the Viterbi algorithm to emit valid sequences of BIO tags, using unary scores s_{ft} and the transition probabilities given by the training data.

2.5 Training

We maximize the sum of the likelihoods of the individual tasks. In order to maximize our model's ability to leverage syntax, during training we clamp $\mathcal{P}$ to the gold parse ($\mathcal{P}_G$) and $\mathcal{V}$ to gold predicates $\mathcal{V}_G$ when passing parse and predicate representations to later layers, whereas syntactic head prediction and joint predicate/POS prediction are conditioned only on the input sequence $\mathcal{X}$. The overall objective is thus:

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\Big[\sum_{f=1}^{F}&\log P(y_{ft}^{role} \mid \mathcal{P}_G, \mathcal{V}_G, \mathcal{X})\\ &+ \log P(y_t^{prp} \mid \mathcal{X})\\ &+ \lambda_1 \log P(\text{head}(t) \mid \mathcal{X})\\ &+ \lambda_2 \log P(y_t^{dep} \mid \mathcal{P}_G, \mathcal{X})\Big]\end{aligned} \tag{7}$$

where λ_1 and λ_2 are penalties on the syntactic attention loss.

We train the model using Nadam (Dozat, 2016) SGD combined with the learning rate schedule in Vaswani et al. (2017). In addition to MTL, we regularize our model using dropout (Srivastava et al., 2014). We use gradient clipping to avoid exploding gradients (Bengio et al., 1994; Pascanu et al., 2013). Additional details on optimization and hyperparameters are included in Appendix A.

3 Related work

Early approaches to SRL (Pradhan et al., 2005; Surdeanu et al., 2007; Johansson and Nugues, 2008; Toutanova et al., 2008) focused on developing rich sets of linguistic features as input to a linear model, often combined with complex constrained inference e.g. with an ILP (Punyakanok et al., 2008). Täckström et al. (2015) showed that constraints could be enforced more efficiently using a clever dynamic program for exact inference. Sutton and McCallum (2005) modeled syntactic parsing and SRL jointly, and Lewis et al. (2015) jointly modeled SRL and CCG parsing.

Collobert et al. (2011) were among the first to use a neural network model for SRL, a CNN over word embeddings which failed to out-perform non-neural models. FitzGerald et al. (2015) successfully employed neural networks by embedding lexicalized features and providing them as factors in the model of Täckström et al. (2015).

More recent neural models are syntax-free. Zhou and Xu (2015), Marcheggiani et al. (2017) and He et al. (2017) all use variants of deep LSTMs with constrained decoding, while Tan et al. (2018) apply self-attention to obtain state-of-the-art SRL with gold predicates. Like this work, He et al. (2017) present end-to-end experiments, predicting predicates using an LSTM, and He et al.

---

[4] All predicates in CoNLL-2005 are verbs; CoNLL-2012 includes some nominal predicates.

(2018) jointly predict SRL spans and predicates in a model based on that of Lee et al. (2017), obtaining state-of-the-art predicted predicate SRL. Concurrent to this work, Peters et al. (2018) and He et al. (2018) report significant gains on PropBank SRL by training a wide LSTM language model and using a task-specific transformation of its hidden representations (ELMo) as a deep, and computationally expensive, alternative to typical word embeddings. We find that LISA obtains further accuracy increases when provided with ELMo word representations, especially on out-of-domain data.

Some work has incorporated syntax into neural models for SRL. Roth and Lapata (2016) incorporate syntax by embedding dependency paths, and similarly Marcheggiani and Titov (2017) encode syntax using a graph CNN over a predicted syntax tree, out-performing models without syntax on CoNLL-2009. These works are limited to incorporating partial dependency paths between tokens whereas our technique incorporates the entire parse. Additionally, Marcheggiani and Titov (2017) report that their model does not out-perform syntax-free models on out-of-domain data, a setting in which our technique excels.

MTL (Caruana, 1993) is popular in NLP, and others have proposed MTL models which incorporate subsets of the tasks we do (Collobert et al., 2011; Zhang and Weiss, 2016; Hashimoto et al., 2017; Peng et al., 2017; Swayamdipta et al., 2017), and we build off work that investigates where and when to combine different tasks to achieve the best results (Søgaard and Goldberg, 2016; Bingel and Søgaard, 2017; Alonso and Plank, 2017). Our specific method of incorporating supervision into self-attention is most similar to the concurrent work of Liu and Lapata (2018), who use edge marginals produced by the matrix-tree algorithm as attention weights for document classification and natural language inference.

The question of training on gold versus predicted labels is closely related to learning to search (Daumé III et al., 2009; Ross et al., 2011; Chang et al., 2015) and scheduled sampling (Bengio et al., 2015), with applications in NLP to sequence labeling and transition-based parsing (Choi and Palmer, 2011; Goldberg and Nivre, 2012; Ballesteros et al., 2016). Our approach may be interpreted as an extension of teacher forcing (Williams and Zipser, 1989) to MTL. We leave exploration of more advanced scheduled sampling techniques to future work.

4 Experimental results

We present results on the CoNLL-2005 shared task (Carreras and Màrquez, 2005) and the CoNLL-2012 English subset of OntoNotes 5.0 (Pradhan et al., 2006), achieving state-of-the-art results for a single model with predicted predicates on both corpora. We experiment with both standard pre-trained GloVe word embeddings (Pennington et al., 2014) and pre-trained ELMo representations with fine-tuned task-specific parameters (Peters et al., 2018) in order to best compare to prior work. Hyperparameters that resulted in the best performance on the validation set were selected via a small grid search, and models were trained for a maximum of 4 days on one TitanX GPU using early stopping on the validation set. We convert constituencies to dependencies using the Stanford head rules v3.5 (de Marneffe and Manning, 2008). A detailed description of hyperparameter settings and data pre-processing can be found in Appendix A.

We compare our **LISA** models to four strong baselines: For experiments using predicted predicates, we compare to He et al. (2018) and the ensemble model (**PoE**) from He et al. (2017), as well as a version of our own self-attention model which does not incorporate syntactic information (**SA**). To compare to more prior work, we present additional results on CoNLL-2005 with models given gold predicates at test time. In these experiments we also compare to Tan et al. (2018), the previous state-of-the art SRL model using gold predicates and standard embeddings.

We demonstrate that our models benefit from injecting state-of-the-art predicted parses at test time (**+D&M**) by fixing the attention to parses predicted by Dozat and Manning (2017), the winner of the 2017 CoNLL shared task (Zeman et al., 2017) which we re-train using ELMo embeddings. In all cases, using these parses at test time improves performance.

We also evaluate our model using the gold syntactic parse at test time (**+Gold**), to provide an upper bound for the benefit that syntax could have for SRL using LISA. These experiments show that despite LISA's strong performance, there remains substantial room for improvement. In §4.3 we perform further analysis comparing SRL models using gold and predicted parses.

	Dev			WSJ Test			Brown Test		
GloVe	P	R	F1	P	R	F1	P	R	F1
He et al. (2017) PoE	81.8	81.2	81.5	82.0	83.4	82.7	69.7	70.5	70.1
He et al. (2018)	81.3	81.9	81.6	81.2	83.9	82.5	69.7	71.9	70.8
SA	83.52	81.28	82.39	84.17	83.28	83.72	72.98	70.1	71.51
LISA	83.1	81.39	82.24	84.07	83.16	83.61	73.32	70.56	71.91
+D&M	**84.59**	**82.59**	**83.58**	**85.53**	**84.45**	**84.99**	**75.8**	**73.54**	**74.66**
+*Gold*	*87.91*	*85.73*	*86.81*	—	—	—	—	—	—
ELMo									
He et al. (2018)	84.9	**85.7**	85.3	84.8	**87.2**	86.0	73.9	**78.4**	76.1
SA	85.78	84.74	85.26	86.21	85.98	86.09	77.1	75.61	76.35
LISA	**86.07**	84.64	**85.35**	86.69	86.42	86.55	78.95	77.17	78.05
+D&M	85.83	84.51	85.17	**87.13**	86.67	**86.90**	**79.02**	77.49	**78.25**
+*Gold*	*88.51*	*86.77*	*87.63*	—	—	—	—	—	—

Table 1: Precision, recall and F1 on the CoNLL-2005 development and test sets.

WSJ Test	P	R	F1
He et al. (2018)	84.2	83.7	83.9
Tan et al. (2018)	84.5	85.2	84.8
SA	84.7	84.24	84.47
LISA	84.72	84.57	84.64
+D&M	**86.02**	**86.05**	**86.04**

Brown Test	P	R	F1
He et al. (2018)	74.2	73.1	73.7
Tan et al. (2018)	73.5	74.6	74.1
SA	73.89	72.39	73.13
LISA	74.77	74.32	74.55
+D&M	**76.65**	**76.44**	**76.54**

Table 2: Precision, recall and F1 on CoNLL-2005 with gold predicates.

4.1 Semantic role labeling

Table 1 lists precision, recall and F1 on the CoNLL-2005 development and test sets using predicted predicates. For models using GloVe embeddings, our syntax-free SA model already achieves a new state-of-the-art by jointly predicting predicates, POS and SRL. LISA with its own parses performs comparably to SA, but when supplied with D&M parses LISA out-performs the previous state-of-the-art by 2.5 F1 points. On the out-of-domain Brown test set, LISA also performs comparably to its syntax-free counterpart with its own parses, but with D&M parses LISA performs exceptionally well, more than 3.5 F1 points higher than He et al. (2018). Incorporating ELMo embeddings improves all scores. The gap in SRL F1 between models using LISA and D&M parses is smaller due to LISA's improved parsing accuracy (see §4.2), but LISA with D&M parses still achieves the highest F1: nearly 1.0 absolute F1 higher than the previous state-of-the art on WSJ, and more than 2.0 F1 higher on Brown. In both settings LISA leverages domain-agnostic syntactic information rather than over-fitting to the newswire training data which leads to high performance even on out-of-domain text.

To compare to more prior work we also evaluate our models in the artificial setting where gold predicates are provided at test time. For fair comparison we use GloVe embeddings, provide predicate indicator embeddings on the input and re-encode the sequence relative to each gold predicate. Here LISA still excels: with D&M parses, LISA out-performs the previous state-of-the-art by more than 2 F1 on both WSJ and Brown.

Table 3 reports precision, recall and F1 on the CoNLL-2012 test set. We observe performance similar to that observed on ConLL-2005: Using GloVe embeddings our SA baseline already out-performs He et al. (2018) by nearly 1.5 F1. With its own parses, LISA slightly under-performs our syntax-free model, but when provided with stronger D&M parses LISA out-performs the state-of-the-art by more than 2.5 F1. Like CoNLL-2005, ELMo representations improve all models and close the F1 gap between models supplied with LISA and D&M parses. On this dataset ELMo also substantially narrows the

Dev	P	R	F1
	GloVe		
He et al. (2018)	79.2	79.7	79.4
SA	82.32	79.76	81.02
LISA	81.77	79.65	80.70
+D&M	**82.97**	**81.14**	**82.05**
+*Gold*	*87.57*	*85.32*	*86.43*
	ELMo		
He et al. (2018)	82.1	**84.0**	83.0
SA	84.35	82.14	83.23
LISA	**84.19**	82.56	**83.37**
+D&M	84.09	82.65	83.36
+*Gold*	*88.22*	*86.53*	*87.36*

Test	P	R	F1
	GloVe		
He et al. (2018)	79.4	80.1	79.8
SA	82.55	80.02	81.26
LISA	81.86	79.56	80.70
+D&M	**83.3**	**81.38**	**82.33**
	ELMo		
He et al. (2018)	81.9	**84.0**	82.9
SA	**84.39**	82.21	83.28
LISA	83.97	82.29	83.12
+D&M	84.14	82.64	**83.38**

Table 3: Precision, recall and F1 on the CoNLL-2012 development and test sets. Italics indicate a synthetic upper bound obtained by providing a gold parse at test time.

difference between models with- and without syntactic information. This suggests that for this challenging dataset, ELMo already encodes much of the information available in the D&M parses. Yet, higher accuracy parses could still yield improvements since providing gold parses increases F1 by 4 points even with ELMo embeddings.

4.2 Parsing, POS and predicate detection

Data	Model	POS	UAS	LAS
WSJ	D&M$_E$	—	96.48	94.40
	LISA$_G$	96.92	94.92	91.87
	LISA$_E$	97.80	96.28	93.65
Brown	D&M$_E$	—	92.56	88.52
	LISA$_G$	94.26	90.31	85.82
	LISA$_E$	95.77	93.36	88.75
CoNLL-12	D&M$_E$	—	94.99	92.59
	LISA$_G$	96.81	93.35	90.42
	LISA$_E$	98.11	94.84	92.23

Table 4: Parsing (labeled and unlabeled attachment) and POS accuracies attained by the models used in SRL experiments on test datasets. Subscript G denotes GloVe and E ELMo embeddings.

	Model	P	R	F1
WSJ	He et al. (2017)	94.5	98.5	96.4
	LISA	98.9	97.9	98.4
Brown	He et al. (2017)	89.3	95.7	92.4
	LISA	95.5	91.9	93.7
CoNLL-12	LISA	99.8	94.7	97.2

Table 5: Predicate detection precision, recall and F1 on CoNLL-2005 and CoNLL-2012 test sets.

We first report the labeled and unlabeled attachment scores (LAS, UAS) of our parsing models on the CoNLL-2005 and 2012 test sets (Table 4) with GloVe (G) and ELMo (E) embeddings. D&M achieves the best scores. Still, LISA's GloVe UAS is comparable to popular off-the-shelf dependency parsers such as spaCy,[5] and with ELMo embeddings comparable to the standalone D&M parser. The difference in parse accuracy between LISA$_G$ and D&M likely explains the large increase in SRL performance we see from decoding with D&M parses in that setting.

In Table 5 we present predicate detection precision, recall and F1 on the CoNLL-2005 and 2012 test sets. SA and LISA with and without ELMo attain comparable scores so we report only LISA+GloVe. We compare to He et al. (2017) on CoNLL-2005, the only cited work reporting comparable predicate detection F1. LISA attains high predicate detection scores, above 97 F1, on both in-domain datasets, and out-performs He et al. (2017) by 1.5-2 F1 points even on the out-of-domain Brown test set, suggesting that multi-task learning works well for SRL predicate detection.

4.3 Analysis

First we assess SRL F1 on sentences divided by parse accuracy. Table 6 lists average SRL F1 (across sentences) for the four conditions of LISA and D&M parses being correct or not (**L**±, **D**±). Both parsers are correct on 26% of sentences.

---

[5] spaCy reports 94.48 UAS on WSJ using Stanford dependencies v3.3: `https://spacy.io/usage/facts-figures`

	L+/D+	L–/D+	L+/D–	L–/D–
Proportion	26%	12%	4%	56%
SA	79.29	75.14	75.97	75.08
LISA	79.51	74.33	79.69	75.00
+D&M	79.03	76.96	77.73	76.52
+*Gold*	*79.61*	*78.38*	*81.41*	*80.47*

Table 6: Average SRL F1 on CoNLL-2005 for sentences where LISA (L) and D&M (D) parses were completely correct (+) or incorrect (–).

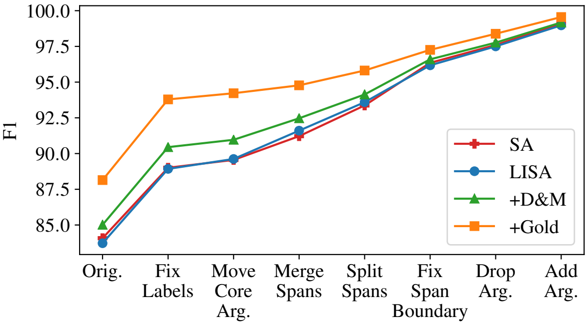

Figure 3: Performance of CoNLL-2005 models after performing corrections from He et al. (2017).

Here there is little difference between any of the models, with LISA models tending to perform slightly better than SA. Both parsers make mistakes on the majority of sentences (57%), difficult sentences where SA also performs the worst. These examples are likely where gold and D&M parses improve the most over other models in overall F1: Though both parsers fail to correctly parse the entire sentence, the D&M parser is less wrong (87.5 vs. 85.7 average LAS), leading to higher SRL F1 by about 1.5 average F1.

Following He et al. (2017), we next apply a series of corrections to model predictions in order to understand which error types the gold parse resolves: e.g. *Fix Labels* fixes labels on spans matching gold boundaries, and *Merge Spans* merges adjacent predicted spans into a gold span.[6]

In Figure 3 we see that much of the performance gap between the gold and predicted parses is due to span boundary errors (*Merge Spans*, *Split Spans* and *Fix Span Boundary*), which supports the hypothesis proposed by He et al. (2017) that incorporating syntax could be particularly helpful for resolving these errors. He et al. (2017) also point out that these errors are due mainly to prepositional phrase (PP) attachment mistakes. We also find this to be the case: Figure 4 shows a breakdown of split/merge corrections by phrase type. Though the number of corrections decreases substantially across phrase types, the proportion of corrections attributed to PPs remains the same (approx. 50%) even after providing the correct PP attachment to the model, indicating that PP span boundary mistakes are a fundamental difficulty for SRL.

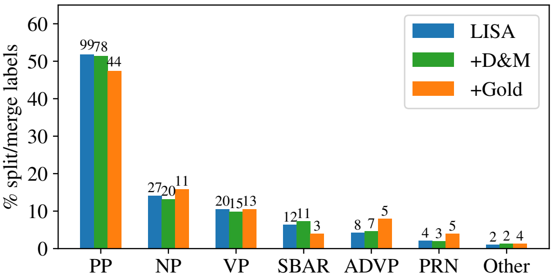

Figure 4: Percent and count of split/merge corrections performed in Figure 3, by phrase type.

5 Conclusion

We present linguistically-informed self-attention: a multi-task neural network model that effectively incorporates rich linguistic information for semantic role labeling. LISA out-performs the state-of-the-art on two benchmark SRL datasets, including out-of-domain. Future work will explore improving LISA's parsing accuracy, developing better training techniques and adapting to more tasks.

Acknowledgments

We are grateful to Luheng He for helpful discussions and code, Timothy Dozat for sharing his code, and to the NLP reading groups at Google and UMass and the anonymous reviewers for feedback on drafts of this work. This work was supported in part by an IBM PhD Fellowship Award to E.S., in part by the Center for Intelligent Information Retrieval, and in part by the National Science Foundation under Grant Nos. DMR-1534431 and IIS-1514053. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

---

[6] Refer to He et al. (2017) for a detailed explanation of the different error types.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow.org*.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *EACL*.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack lstm parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2005–2010.

Marzieh Bazrafshan and Daniel Gildea. 2013. Semantic roles for string to tree machine translation. In *ACL*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Brad Huang, Christopher D. Manning, Abby Vander Linden, Brittany Harding, and Peter Clark. 2014. Modeling biological processes for reading comprehension. In *EMNLP*.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *CoNLL*.

Rich Caruana. 1993. Multitask learning: a knowledge-based source of inductive bias. In *ICML*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *ICML*.

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *Proc. of ASRU-IEEE*.

Jinho D. Choi and Martha Palmer. 2011. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: short papers*, pages 687–692.

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

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Timothy Dozat. 2016. Incorporating nesterov momentum into adam. In *ICLR Workshop track*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970.

W. N. Francis and H. Kučera. 1964. Manual of information to accompany a standard corpus of present-day edited american english, for use with digital computers. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012: Technical Papers*, pages 959–976.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Conference on Empirical Methods in Natural Language Processing*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *ACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 69–78.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations (ICLR)*, San Diego, California, USA.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Thomas N. Kipf and Max Welling. 2017. Semisupervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG Parsing and Semantic Role Labeling. In *EMNLP*.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, volume 30.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *CoNLL*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn TreeBank. *Computational Linguistics – Special issue on using large corpora: II*, 19(2):313–330.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *COLING 2008 Workshop on Cross-framework and Cross-domain Parser Evaluation*.

Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. volume 27, pages 372–376.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30 th International Conference on Machine Learning*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2006. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the Association for Computational Linguistics 43rd annual meeting (ACL)*.

Vasin Punyakanok, Dan Roth, and Wen-Tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1192–1202.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 231–235.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *CoNLL*.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. In *arXiv:1706.09528*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *TACL*, 3:29–41.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *AAAI*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Gokhan Tur, Dilek Hakkani-Tür, and Ananlada Chotimongkol. 2005. Semi-supervised learning for spoken language understanding using semantic role labeling. In *ASRU*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS)*.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *ACL*.

R. J. Williams and D. Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, et al. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Phrase-Based & Neural Unsupervised Machine Translation

Guillaume Lample[†]
Facebook AI Research
Sorbonne Universités
glample@fb.com

Myle Ott
Facebook AI Research
myleott@fb.com

Alexis Conneau
Facebook AI Research
Université Le Mans
aconneau@fb.com

Ludovic Denoyer[†]
Sorbonne Universités
ludovic.denoyer@lip6.fr

Marc'Aurelio Ranzato
Facebook AI Research
ranzato@fb.com

Abstract

Machine translation systems achieve near human-level performance on some languages, yet their effectiveness strongly relies on the availability of large amounts of parallel sentences, which hinders their applicability to the majority of language pairs. This work investigates how to learn to translate when having access to only large monolingual corpora in each language. We propose two model variants, a neural and a phrase-based model. Both versions leverage a careful initialization of the parameters, the denoising effect of language models and automatic generation of parallel data by iterative back-translation. These models are significantly better than methods from the literature, while being simpler and having fewer hyper-parameters. On the widely used WMT'14 English-French and WMT'16 German-English benchmarks, our models respectively obtain 28.1 and 25.2 BLEU points without using a single parallel sentence, outperforming the state of the art by more than 11 BLEU points. On low-resource languages like English-Urdu and English-Romanian, our methods achieve even better results than semi-supervised and supervised approaches leveraging the paucity of available bitexts. Our code for NMT and PBSMT is publicly available.[1]

1 Introduction

Machine Translation (MT) is a flagship of the recent successes and advances in the field of natural language processing. Its practical applications and use as a testbed for sequence transduction algorithms have spurred renewed interest in this topic.

While recent advances have reported near human-level performance on several language pairs using neural approaches (Wu et al., 2016; Hassan et al., 2018), other studies have highlighted several open challenges (Koehn and Knowles, 2017; Isabelle et al., 2017; Sennrich, 2017). A major challenge is the reliance of current learning algorithms on large parallel corpora. Unfortunately, the vast majority of language pairs have very little, if any, parallel data: learning algorithms need to better leverage monolingual data in order to make MT more widely applicable.

While a large body of literature has studied the use of monolingual data to boost translation performance when limited supervision is available, two recent approaches have explored the fully unsupervised setting (Lample et al., 2018; Artetxe et al., 2018), relying only on monolingual corpora in each language, as in the pioneering work by Ravi and Knight (2011). While there are subtle technical differences between these two recent works, we identify several common principles underlying their success.

First, they carefully initialize the MT system with an inferred bilingual dictionary. Second, they leverage strong language models, via training the sequence-to-sequence system (Sutskever et al., 2014; Bahdanau et al., 2015) as a denoising autoencoder (Vincent et al., 2008). Third, they turn the unsupervised problem into a supervised one by automatic generation of sentence pairs via *back-translation* (Sennrich et al., 2015a), i.e., the source-to-target model is applied to source sentences to generate inputs for training the target-to-source model, and vice versa. Finally, they constrain the latent representations produced by the encoder to be shared across the two languages. Empirically, these methods achieve remarkable results considering the fully unsupervised setting; for instance, about 15 BLEU points on the WMT'14 English-French benchmark.

The first contribution of this paper is a model

[†] Sorbonne Universités, UPMC Univ Paris 06, CNRS, UMR 7606, LIP6, F-75005, Paris, France.

[1] https://github.com/facebookresearch/UnsupervisedMT

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 5039–5049
Brussels, Belgium, October 31 - November 4, 2018.

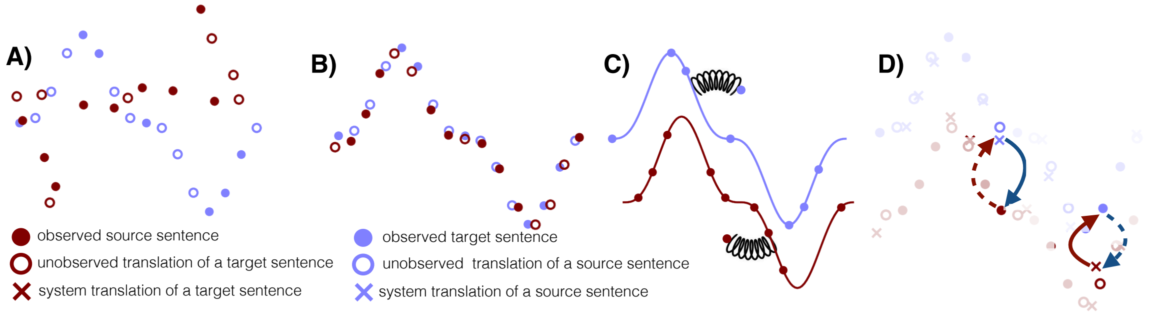

Figure 1: Toy illustration of the three principles of unsupervised MT. **A)** There are two monolingual datasets. Markers correspond to sentences (see legend for details). **B)** First principle: **Initialization**. The two distributions are roughly aligned, e.g. by performing word-by-word translation with an inferred bilingual dictionary. **C)** Second principle: **Language modeling**. A language model is learned independently in each domain to infer the structure in the data (underlying continuous curve); it acts as a data-driven prior to denoise/correct sentences (illustrated by the spring pulling a sentence outside the manifold back in). **D)** Third principle: **Back-translation**. Starting from an observed source sentence (filled red circle) we use the current source → target model to translate (dashed arrow), yielding a potentially incorrect translation (blue cross near the empty circle). Starting from this (back) translation, we use the target → source model (continuous arrow) to reconstruct the sentence in the original language. The discrepancy between the reconstruction and the initial sentence provides error signal to train the target → source model parameters. The same procedure is applied in the opposite direction to train the source → target model.

that combines these two previous neural approaches, simplifying the architecture and loss function while still following the above mentioned principles. The resulting model outperforms previous approaches and is both easier to train and tune. Then, we apply the same ideas and methodology to a traditional phrase-based statistical machine translation (PBSMT) system (Koehn et al., 2003). PBSMT models are well-known to outperform neural models when labeled data is scarce because they merely count occurrences, whereas neural models typically fit hundred of millions of parameters to learn distributed representations, which may generalize better when data is abundant but is prone to overfit when data is scarce. Our PBSMT model is simple, easy to interpret, fast to train and often achieves similar or better results than its NMT counterpart. We report gains of up to +10 BLEU points on widely used benchmarks when using our NMT model, and up to +12 points with our PBSMT model. Furthermore, we apply these methods to distant and low-resource languages, like English-Russian, English-Romanian and English-Urdu, and report competitive performance against both semi-supervised and supervised baselines.

2 Principles of Unsupervised MT

Learning to translate with only monolingual data is an ill-posed task, since there are potentially many ways to associate target with source sentences. Nevertheless, there has been exciting progress towards this goal in recent years, as discussed in the related work of Section 5. In this section, we abstract away from the specific assumptions made by each prior work and instead focus on identifying the common principles underlying unsupervised MT.

We claim that unsupervised MT can be accomplished by leveraging the three components illustrated in Figure 1: (i) suitable initialization of the translation models, (ii) language modeling and (iii) iterative back-translation. In the following, we describe each of these components and later discuss how they can be better instantiated in both a neural model and phrase-based model.

Initialization: Given the ill-posed nature of the task, model initialization expresses a natural prior over the space of solutions we expect to reach, jump-starting the process by leveraging approximate translations of words, short phrases or even sub-word units (Sennrich et al., 2015b). For instance, Klementiev et al. (2012) used a provided bilingual dictionary, while Lample et al. (2018) and Artetxe et al. (2018) used dictionaries inferred in an unsupervised way (Conneau et al., 2018; Artetxe et al., 2017). The motivating intuition is that while such initial “word-by-word” translation may be poor if languages or corpora are not closely related, it still preserves some of the original semantics.

Language Modeling: Given large amounts of monolingual data, we can train language models on both source and target languages. These models express a data-driven prior about how sentences should read in each language, and they improve the quality of the translation models by per-

Algorithm 1: Unsupervised MT

```
Language models: Learn language models $P_s$ and $P_t$ over source and target languages;
Initial translation models: Leveraging $P_s$ and $P_t$, learn two initial translation models, one in each direction: $P^{(0)}_{s\to t}$ and $P^{(0)}_{t\to s}$;
for k=1 to N do
    Back-translation: Generate source and target sentences using the current translation models, $P^{(k-1)}_{t\to s}$ and $P^{(k-1)}_{s\to t}$, factoring in language models, $P_s$ and $P_t$;
    Train new translation models $P^{(k)}_{s\to t}$ and $P^{(k)}_{t\to s}$ using the generated sentences and leveraging $P_s$ and $P_t$;
end
```

forming local substitutions and word reorderings.

Iterative Back-translation: The third principle is back-translation (Sennrich et al., 2015a), which is perhaps the most effective way to leverage monolingual data in a semi-supervised setting. Its application in the unsupervised setting is to couple the source-to-target translation system with a backward model translating from the target to source language. The goal of this model is to generate a source sentence for each target sentence in the monolingual corpus. This turns the daunting unsupervised problem into a supervised learning task, albeit with noisy source sentences. As the original model gets better at translating, we use the current model to improve the back-translation model, resulting in a coupled system trained with an iterative algorithm (He et al., 2016).

3 Unsupervised MT systems

Equipped with the three principles detailed in Section 2, we now discuss how to effectively combine them in the context of a NMT model (Section 3.1) and PBSMT model (Section 3.2).

In the reminder of the paper, we denote the space of source and target sentences by $\mathcal{S}$ and $\mathcal{T}$, respectively, and the language models trained on source and target monolingual datasets by P_s and P_t, respectively. Finally, we denote by $P_{s\to t}$ and $P_{t\to s}$ the translation models from source to target and vice versa. An overview of our approach is given in Algorithm 1.

3.1 Unsupervised NMT

We now introduce a new unsupervised NMT method, which is derived from earlier work by Artetxe et al. (2018) and Lample et al. (2018). We first discuss how the previously mentioned three key principles are instantiated in our work, and then introduce an additional property of the system, the sharing of internal representations across languages, which is specific and critical to NMT. From now on, we assume that a NMT model consists of an encoder and a decoder. Section 4 gives the specific details of this architecture.

Initialization: While prior work relied on bilingual dictionaries, here we propose a more effective and simpler approach which is particularly suitable for related languages.[2] First, instead of considering words, we consider byte-pair encodings (BPE) (Sennrich et al., 2015b), which have two major advantages: they reduce the vocabulary size and they eliminate the presence of unknown words in the output translation. Second, instead of learning an explicit mapping between BPEs in the source and target languages, we define BPE tokens by *jointly* processing both monolingual corpora. If languages are related, they will naturally share a good fraction of BPE tokens, which eliminates the need to infer a bilingual dictionary. In practice, we i) join the monolingual corpora, ii) apply BPE tokenization on the resulting corpus, and iii) learn token embeddings (Mikolov et al., 2013) on the same corpus, which are then used to initialize the lookup tables in the encoder and decoder.

Language Modeling: In NMT, language modeling is accomplished via denoising autoencoding, by minimizing:

$$\begin{aligned}\mathcal{L}^{lm} = \ & \mathbb{E}_{x\sim\mathcal{S}}[-\log P_{s\to s}(x|C(x))] + \\ & \mathbb{E}_{y\sim\mathcal{T}}[-\log P_{t\to t}(y|C(y))] \qquad (1)\end{aligned}$$

where C is a noise model with some words dropped and swapped as in Lample et al. (2018). $P_{s\to s}$ and $P_{t\to t}$ are the composition of encoder and decoder both operating on the source and target sides, respectively.

Back-translation: Let us denote by $u^*(y)$ the sentence in the source language inferred from $y \in \mathcal{T}$ such that $u^*(y) = \arg\max P_{t\to s}(u|y)$. Similarly, let us denote by $v^*(x)$ the sentence in the target language inferred from $x \in \mathcal{S}$ such that $v^*(x) = \arg\max P_{s\to t}(v|x)$. The pairs $(u^*(y), y)$ and $(x, v^*(x)))$ constitute automatically-generated parallel sentences which, following the back-translation principle, can be

---

[2] For unrelated languages, we need to infer a dictionary to properly initialize the embeddings (Conneau et al., 2018).

used to train the two MT models by minimizing the following loss:

$$\begin{aligned}\mathcal{L}^{back} &= \mathbb{E}_{y\sim\mathcal{T}}[-\log P_{s\to t}(y|u^*(y))] + \\ &\quad \mathbb{E}_{x\sim\mathcal{S}}[-\log P_{t\to s}(x|v^*(x))]. \quad (2)\end{aligned}$$

Note that when minimizing this objective function we do not back-prop through the reverse model which generated the data, both for the sake of simplicity and because we did not observe improvements when doing so. The objective function minimized at every iteration of stochastic gradient descent, is simply the sum of $\mathcal{L}^{lm}$ in Eq. 1 and $\mathcal{L}^{back}$ in Eq. 2. To prevent the model from cheating by using different subspaces for the language modeling and translation tasks, we add an additional constraint which we discuss next.

Sharing Latent Representations: A shared encoder representation acts like an interlingua, which is translated in the decoder target language regardless of the input source language. This ensures that the benefits of language modeling, implemented via the denoising autoencoder objective, nicely transfer to translation from noisy sources and eventually help the NMT model to translate more fluently. In order to share the encoder representations, we share all encoder parameters (including the embedding matrices since we perform joint tokenization) across the two languages to ensure that the latent representation of the source sentence is robust to the source language. Similarly, we share the decoder parameters across the two languages. While sharing the encoder is critical to get the model to work, sharing the decoder simply induces useful regularization. Unlike prior work (Johnson et al., 2016), the first token of the decoder specifies the language the module is operating with, while the encoder does not have any language identifier.

3.2 Unsupervised PBSMT

In this section, we discuss how to perform unsupervised machine translation using a Phrase-Based Statistical Machine Translation (PBSMT) system (Koehn et al., 2003) as the underlying backbone model. Note that PBSMT models are known to perform well on low-resource language pairs, and are therefore a potentially good alternative to neural models in the unsupervised setting.

When translating from x to y, a PBSMT system scores y according to: $\arg\max_y P(y|x) = \arg\max_y P(x|y)P(y)$, where $P(x|y)$ is derived from so called "phrase tables", and $P(y)$ is the score assigned by a language model.

Given a dataset of bitexts, PBSMT first infers an alignment between source and target phrases. It then populates phrase tables, whose entries store the probability that a certain n-gram in the source/target language is mapped to another n-gram in the target/source language.

In the unsupervised setting, we can easily train a language model on monolingual data, but it is less clear how to populate the phrase tables, which are a necessary component for good translation. Fortunately, similar to the neural case, the principles of Section 2 are effective to solve this problem.

Initialization: We populate the initial phrase tables (from source to target and from target to source) using an inferred bilingual dictionary built from monolingual corpora using the method proposed by Conneau et al. (2018). In the following, we will refer to phrases as single words, but the very same arguments trivially apply to longer n-grams. Phrase tables are populated with the scores of the translation of a source word to:

$$p(t_j|s_i) = \frac{e^{\frac{1}{T}\cos(e(t_j),We(s_i))}}{\sum_k e^{\frac{1}{T}\cos(e(t_k),We(s_i))}}, \quad (3)$$

where t_j is the j-th word in the target vocabulary and s_i is the i-th word in the source vocabulary, T is a hyper-parameter used to tune the peakiness of the distribution[3], W is the rotation matrix mapping the source embeddings into the target embeddings (Conneau et al., 2018), and $e(x)$ is the embedding of x.

Language Modeling: Both in the source and target domains we learn smoothed n-gram language models using KenLM (Heafield, 2011), although neural models could also be considered. These remain fixed throughout training iterations.

Iterative Back-Translation: To jump-start the iterative process, we use the unsupervised phrase tables and the language model on the target side to construct a seed PBSMT. We then use this model to translate the source monolingual corpus into the target language (back-translation step). Once the data has been generated, we train a PBSMT in supervised mode to map the generated data back to the original source sentences. Next, we perform

---

[3] We set $T = 30$ in all our experiments, following the setting of Smith et al. (2017).

both generation and training process but in the reverse direction. We repeat these steps as many times as desired (see Algorithm 2 in Section A).

Intuitively, many entries in the phrase tables are not correct because the input to the PBSMT at any given point during training is noisy. Despite that, the language model may be able to fix some of these mistakes at generation time. As long as that happens, the translation improves, and with that also the phrase tables at the next round. There will be more entries that correspond to correct phrases, which makes the PBSMT model stronger because it has bigger tables and it enables phrase swaps over longer spans.

4 Experiments

We first describe the datasets and experimental protocol we used. Then, we compare the two proposed unsupervised approaches to earlier attempts, to semi-supervised methods and to the very same models but trained with varying amounts of labeled data. We conclude with an ablation study to understand the relative importance of the three principles introduced in Section 2.

4.1 Datasets and Methodology

We consider five language pairs: English-French, English-German, English-Romanian, English-Russian and English-Urdu. The first two pairs are used to compare to recent work on unsupervised MT (Artetxe et al., 2018; Lample et al., 2018). The last three pairs are instead used to test our PBSMT unsupervised method on truly low-resource pairs (Gu et al., 2018) or unrelated languages that do not even share the same alphabet.

For English, French, German and Russian, we use all available sentences from the WMT monolingual News Crawl datasets from years 2007 through 2017. For Romanian, the News Crawl dataset is only composed of 2.2 million sentences, so we augment it with the monolingual data from WMT'16, resulting in 2.9 million sentences. In Urdu, we use the dataset of Jawaid et al. (2014), composed of about 5.5 million monolingual sentences. We report results on *newstest* 2014 for $en-fr$, and *newstest* 2016 for $en-de$, $en-ro$ and $en-ru$. For Urdu, we use the LDC2010T21 and LDC2010T23 corpora each with about 1800 sentences as validation and test sets, respectively.

We use Moses scripts (Koehn et al., 2007) for tokenization. NMT is trained with 60,000 BPE codes. PBSMT is trained with true-casing, and by removing diacritics from Romanian on the source side to deal with their inconsistent use across the monolingual dataset (Sennrich et al., 2016).

Source	Target	$P(\text{s}\|\text{t})$	$P(\text{t}\|\text{s})$
heureux	happy	0.931	0.986
	delighted	0.458	0.003
	grateful	0.128	0.003
	thrilled	0.392	0.002
	glad	0.054	0.001
Royaume-Uni	Britain	0.242	0.720
	UK	0.816	0.257
	U.K.	0.697	0.011
	United Kingdom	0.770	0.010
	British	0.000	0.002
Union européenne	European Union	0.869	0.772
	EU	0.335	0.213
	E.U.	0.539	0.006
	member states	0.007	0.006
	27-nation bloc	0.410	0.002

Table 1: **Unsupervised phrase table.** Example of candidate French to English phrase translations, along with their corresponding conditional likelihoods.

4.2 Initialization

Both the NMT and PBSMT approaches require either cross-lingual BPE embeddings (to initialize the shared lookup tables) or n-gram embeddings (to initialize the phrase table). We generate embeddings using fastText (Bojanowski et al., 2017) with an embedding dimension of 512, a context window of size 5 and 10 negative samples. For NMT, fastText is applied on the concatenation of source and target corpora, which results in cross-lingual BPE embeddings.

For PBSMT, we generate n-gram embeddings on the source and target corpora independently, and align them using the MUSE library (Conneau et al., 2018). Since learning unique embeddings of every possible phrase would be intractable, we consider the most frequent 300,000 source phrases, and align each of them to its 200 nearest neighbors in the target space, resulting in a phrase table of 60 million phrase pairs which we score using the formula in Eq. 3.

In practice, we observe a small but significant difference of about 1 BLEU point using a phrase table of bigrams compared to a phrase table of unigrams, but did not observe any improvement using longer phrases. Table 1 shows an extract of a French-English unsupervised phrase table, where we can see that unigrams are correctly aligned to bigrams, and vice versa.

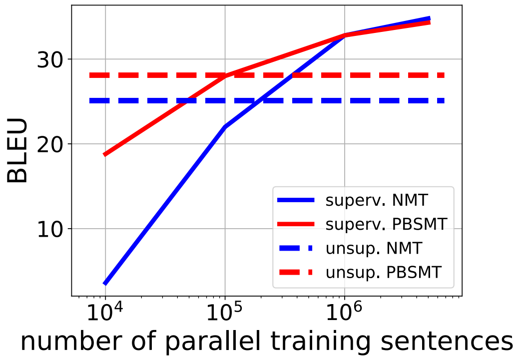

Figure 2: Comparison between supervised and unsupervised approaches on WMT'14 En-Fr, as we vary the number of parallel sentences for the supervised methods.

Model	en-fr	fr-en	en-de	de-en
(Artetxe et al., 2018)	15.1	15.6	-	-
(Lample et al., 2018)	15.0	14.3	9.6	13.3
(Yang et al., 2018)	17.0	15.6	10.9	14.6
NMT (LSTM)	24.5	23.7	14.7	19.6
NMT (Transformer)	25.1	24.2	17.2	21.0
PBSMT (Iter. 0)	16.2	17.5	11.0	15.6
PBSMT (Iter. n)	**28.1**	27.2	17.9	22.9
NMT + PBSMT	27.1	26.3	17.5	22.1
PBSMT + NMT	27.6	**27.7**	**20.2**	**25.2**

Table 2: **Comparison with previous approaches.** BLEU score for different models on the $en - fr$ and $en - de$ language pairs. Just using the unsupervised phrase table, and without back-translation (PBSMT (Iter. 0)), the PBSMT outperforms previous approaches. Combining PBSMT with NMT gives the best results.

4.3 Training

The next subsections provide details about the architecture and training procedure of our models.

4.3.1 NMT

In this study, we use NMT models built upon LSTM (Hochreiter and Schmidhuber, 1997) and Transformer (Vaswani et al., 2017) cells. For the LSTM model we use the same architecture as in Lample et al. (2018). For the Transformer, we use 4 layers both in the encoder and in the decoder. Following Press and Wolf (2016), we share all lookup tables between the encoder and the decoder, and between the source and the target languages. The dimensionality of the embeddings and of the hidden layers is set to 512. We used the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 10^{-4}, $\beta_1 = 0.5$, and a batch size of 32. At decoding time, we generate greedily.

4.3.2 PBSMT

The PBSMT uses Moses' default smoothed n-gram language model with phrase reordering disabled during the very first generation. PBSMT is trained in a iterative manner using Algorithm 2. At each iteration, we translate 5 million sentences randomly sampled from the monolingual dataset in the source language. Except for initialization, we use phrase tables with phrases up to length 4.

4.4 Model selection

Moses' implementation of PBSMT has 15 hyper-parameters, such as relative weighting of each scoring function, word penalty, etc. In this work, we consider two methods to set these hyper-parameters. We either set them to their default values in the toolbox, or we set them using a small validation set of parallel sentences. It turns out that with only 100 labeled sentences in the validation set, PBSMT would overfit to the validation set. For instance, on $en \rightarrow fr$, PBSMT tuned on 100 parallel sentences obtains a BLEU score of 26.42 on *newstest* 2014, compared to 27.09 with default hyper-parameters, and 28.02 when tuned on the 3000 parallel sentences of *newstest* 2013. Therefore, unless otherwise specified, all PBSMT models considered in the paper use default hyper-parameter values, and do not use any parallel resource whatsoever.

For the NMT, we also consider two model selection procedures: an *unsupervised criterion* based on the BLEU score of a "round-trip" translation (source → target → source and target → source → target) as in Lample et al. (2018), and cross-validation using a small validation set with 100 parallel sentences. In our experiments, we found the unsupervised criterion to be highly correlated with the test metric when using the Transformer model, but not always for the LSTM. Therefore, unless otherwise specified, we select the best LSTM models using a small validation set of 100 parallel sentences, and the best Transformer models with the unsupervised criterion.

4.5 Results

The results reported in Table 2 show that our unsupervised NMT and PBSMT systems largely outperform previous unsupervised baselines. We report large gains on all language pairs and directions. For instance, on the $en \rightarrow fr$ task, our unsupervised PBSMT obtains a BLEU score of 28.1, outperforming the previous best result by more than 11 BLEU points. Even on a more complex task like $en \rightarrow de$, both PBSMT and NMT surpass the baseline score by more than 10 BLEU

	en → fr	fr→ en	en→ de	de→ en	en→ ro	ro→ en	en→ ru	ru→ en
Unsupervised PBSMT								
Unsupervised phrase table	-	17.50	-	15.63	-	14.10	-	8.08
Back-translation - Iter. 1	24.79	26.16	15.92	22.43	18.21	21.49	11.04	15.16
Back-translation - Iter. 2	27.32	26.80	17.65	22.85	20.61	22.52	12.87	16.42
Back-translation - Iter. 3	27.77	26.93	17.94	22.87	21.18	22.99	13.13	16.52
Back-translation - Iter. 4	27.84	27.20	17.77	22.68	21.33	23.01	13.37	**16.62**
Back-translation - Iter. 5	**28.11**	27.16	-	-	-	-	-	-
Unsupervised NMT								
LSTM	24.48	23.74	14.71	19.60	-	-	-	-
Transformer	25.14	24.18	17.16	21.00	21.18	19.44	7.98	9.09
Phrase-based + Neural network								
NMT + PBSMT	27.12	26.29	17.52	22.06	21.95	23.73	10.14	12.62
PBSMT + NMT	27.60	**27.68**	**20.23**	**25.19**	**25.13**	**23.90**	**13.76**	**16.62**

Table 3: **Fully unsupervised results.** We report the BLEU score for PBSMT, NMT, and their combinations on 8 directed language pairs. Results are obtained on *newstest* 2014 for $en - fr$ and *newstest* 2016 for every other pair.

points. Even before iterative back-translation, the PBSMT model significantly outperforms previous approaches, and can be trained in a few minutes.

Table 3 illustrates the quality of the PBSMT model during the iterative training process. For instance, the $fr \rightarrow en$ model obtains a BLEU score of 17.5 at iteration 0 – i.e. after the unsupervised phrase table construction – while it achieves a score of 27.2 at iteration 4. This highlights the importance of multiple back-translation iterations. The last rows of Table 3 also show that we get additional gains by further tuning the NMT model on the data generated by PBSMT (PBSMT + NMT). We simply add the data generated by the unsupervised PBSMT system to the back-translated data produced by the NMT model. By combining PBSMT and NMT, we achieve BLEU scores of 20.2 and 25.2 on the challenging $en \rightarrow de$ and $de \rightarrow en$ translation tasks. While we also tried bootstraping the PBSMT model with back-translated data generated by a NMT model (NMT + PBSMT), this did not improve over PBSMT alone.

Next, we compare to fully supervised models. Figure 2 shows the performance of the same architectures trained in a fully supervised way using parallel training sets of varying size. The unsupervised PBSMT model achieves the same performance as its supervised counterpart trained on more than 100,000 parallel sentences.

This is confirmed on low-resource languages. In particular, on $ro \rightarrow en$, our unsupervised PBSMT model obtains a BLEU score of 23.9, outperforming Gu et al. (2018)'s method by 1 point, despite its use of 6,000 parallel sentences, a seed dictionary, and a multi-NMT system combining parallel resources from 5 different languages.

On Russian, our unsupervised PBSMT model obtains a BLEU score of 16.6 on $ru \rightarrow en$, showing that this approach works reasonably well on distant languages. Finally we train on $ur \rightarrow en$, which is both low resource and distant. In a supervised mode, PBSMT using the noisy and out-of-domain 800,000 parallel sentences from Tiedemann (2012) achieves a BLEU score of 9.8. Instead, our unsupervised PBSMT system achieves 12.3 BLEU using only a validation set of 1800 sentences to tune Moses hyper-parameters.

4.6 Ablation Study

In Figure 3 we report results from an ablation study, to better understand the importance of the three principles when training PBSMT. This study shows that more iterations only partially compensate for lower quality phrase table initialization (Left), language models trained over less data (Middle) or less monolingual data (Right). Moreover, the influence of the quality of the language model becomes more prominent as we iterate. These findings suggests that better initialization methods and more powerful language models may further improve our results.

We perform a similar ablation study for the NMT system (see Appendix). We find that back-translation and auto-encoding are critical components, without which the system fails to learn. We also find that initialization of embeddings is very important, and we gain 7 BLEU points compared to prior work (Artetxe et al., 2018; Lample et al., 2018) by learning BPE embeddings over the concatenated monolingual corpora.

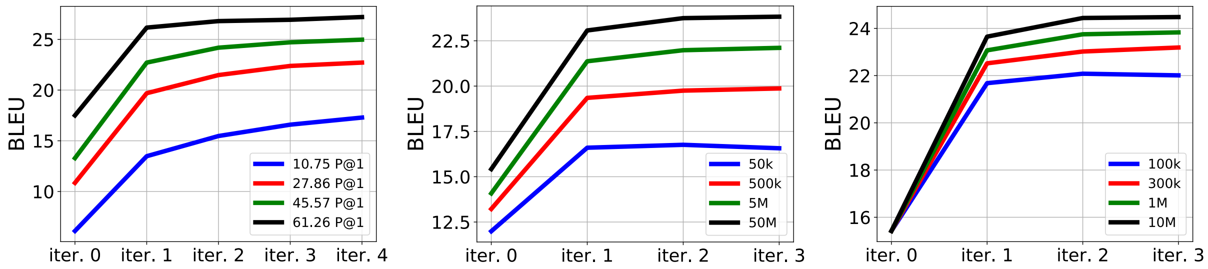

Figure 3: Results with PBSMT on the $fr \rightarrow en$ pair at different iterations. We vary: Left) the quality of the initial alignment between the source and target embeddings (measured in P@1 on the word translation task), Middle) the number of sentences used to train the language models, Right) the number of sentences used for back-translation.

5 Related Work

A large body of literature has studied using monolingual data to boost translation performance when limited supervision is available. This limited supervision is typically provided as a small set of parallel sentences (Sennrich et al., 2015a; Gulcehre et al., 2015; He et al., 2016; Gu et al., 2018; Wang et al., 2018); large sets of parallel sentences in related languages (Firat et al., 2016; Johnson et al., 2016; Chen et al., 2017; Zheng et al., 2017); cross-lingual dictionaries (Klementiev et al., 2012; Irvine and Callison-Burch, 2014, 2016); or comparable corpora (Munteanu et al., 2004; Irvine and Callison-Burch, 2013).

Learning to translate *without* any form of supervision has also attracted interest, but is challenging. In their seminal work, Ravi and Knight (2011) leverage linguistic prior knowledge to reframe the unsupervised MT task as deciphering and demonstrate the feasibility on short sentences with limited vocabulary. Earlier work by Carbonell et al. (2006) also aimed at unsupervised MT, but leveraged a bilingual dictionary to seed the translation. Both works rely on a language model on the target side to correct for translation fluency.

Subsequent work (Klementiev et al., 2012; Irvine and Callison-Burch, 2014, 2016) relied on bilingual dictionaries, small parallel corpora of several thousand sentences, and linguistically motivated features to prune the search space. Irvine and Callison-Burch (2014) use monolingual data to expand phrase tables learned in a supervised setting. In our work we also expand phrase tables, but we initialize them with an inferred bilingual n-gram dictionary, following work from the connectionist community aimed at improving PBSMT with neural models (Schwenk, 2012; Kalchbrenner and Blunsom, 2013; Cho et al., 2014).

In recent years back-translation has become a popular method of augmenting training sets with monolingual data on the target side (Sennrich et al., 2015a), and has been integrated in the “dual learning” framework of He et al. (2016) and subsequent extensions (Wang et al., 2018). Our approach is similar to the dual learning framework, except that in their model gradients are backpropagated through the reverse model and they pretrain using a relatively large amount of labeled data, whereas our approach is fully unsupervised.

Finally, our work can be seen as an extension of recent studies (Lample et al., 2018; Artetxe et al., 2018; Yang et al., 2018) on *fully unsupervised* MT with two major contributions. First, we propose a much simpler and more effective initialization method for related languages. Second, we abstract away three principles of unsupervised MT and apply them to a PBSMT, which even outperforms the original NMT. Moreover, our results show that the combination of PBSMT and NMT achieves even better performance.

6 Conclusions and Future Work

In this work, we identify three principles underlying recent successes in fully unsupervised MT and show how to apply these principles to PBSMT and NMT systems. We find that PBSMT systems often outperform NMT systems in the fully unsupervised setting, and that by combining these systems we can greatly outperform previous approaches from the literature. We apply our approach to several popular benchmark language pairs, obtaining state of the art results, and to several low-resource and under-explored language pairs.

It’s an open question whether there are more effective instantiations of these principles or other principles altogether, and under what conditions our iterative process is guaranteed to converge. Future work may also extend to the semi-supervised setting.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, volume 1, pages 451–462.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In International Conference on Learning Representations (ICLR).

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations (ICLR).

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5:135–146.

Jaime Carbonell, Steve Klein, David Miller, Michael Steinbaum, Tomer Grassiany, and Jochen Frey. 2006. Context-based machine translation. In The Association for Machine Translation in the Americas.

Y. Chen, Y. Liu, Y. Cheng, and V.O.K. Li. 2017. A teacher-student framework for zero-resource neural machine translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder—decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724—–1734.

A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou. 2018. Word translation without parallel data. In International Conference on Learning Representations (ICLR).

O. Firat, B. Sankaran, Y. Al-Onaizan, F.T.Y. Vural, and K. Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal neural machine translation for extremely low resource languages. In Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. arXiv preprint arXiv:1503.03535.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. In arXiv:1803.05567.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In Advances in Neural Information Processing Systems, pages 820–828.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In Proceedings of the Sixth Workshop on Statistical Machine Translation, pages 187–197. Association for Computational Linguistics.

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

Ann Irvine and Chris Callison-Burch. 2013. Combining bilingual and comparable corpora for low resource machine translation. In Proceedings of the eighth workshop on statistical machine translation, pages 262–270.

Ann Irvine and Chris Callison-Burch. 2014. Hallucinating phrase translations for low resource mt. In Proceedings of the Eighteenth Conference on Computational Natural Language Learning, pages 160–170.

Ann Irvine and Chris Callison-Burch. 2016. End-to-end statistical machine translation with zero or small parallel texts. In Journal of Natural Language Engineering, volume 22, pages 517–548.

Pierre Isabelle, Colin Cherry, and George Foster. 2017. A challenge set approach to evaluating machine translation. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2486–2496.

Bushra Jawaid, Amir Kamran, and Ondrej Bojar. 2014. A tagged corpus and a tagger for urdu. In LREC.

M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. In Transactions of the Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Two recurrent continuous translation models. In Conference on Empirical Methods in Natural Language Processing.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Alex Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In Proceedings of the 13th Conference of the European Chapter of the Association for computational Linguistics, Avignon, France. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Ondrej Bojar Chris Dyer, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In Annual Meeting of the Association for Computational Linguistics (ACL), demo session.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In Proceedings of the First Workshop on Neural Machine Translation, pages 28–39.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL), volume 1, pages 48—54.

G. Lample, A. Conneau, L. Denoyer, and M. Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In International Conference on Learning Representations (ICLR).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119.

D.S. Munteanu, A. Fraser, and D. Marcu. 2004. Improved machine translation performance via parallel sentence extraction from comparable corpora. In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL).

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. arXiv preprint arXiv:1608.05859.

S. Ravi and K. Knight. 2011. Deciphering foreign language. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 12–21.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In International Conference on Computational Linguistics, pages 1071—1080.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, pages 376–382.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1715–1725.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. In Proceedings of the First Conference on Machine Translation, pages 371–376.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In Internaltional Conference on Learning Representations.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems, pages 3104–3112.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In Proceedings of the 8th International Conference on Language Resources and Evaluation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. arXiv preprint arXiv:1706.03762.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning, pages 1096–1103.

Yijun Wang, Yingce Xia, Li Zhao, Jiang Bian, Tao Qin, Guiquan Liu, and Tie-Yan Liu. 2018. Dual transfer learning for neural machine translation with marginal distribution regularization. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. Transactions of the Association for Computational Linguistics, pages 339–351.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics.

H. Zheng, Y. Cheng, and Y. Liu. 2017. Maximum expected likelihood estimation for zero-resource neural machine translation. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI), pages 4251–4257.

Author Index

Özbal, Gözde, 3296

Abdou, Mostafa, 4881
Abend, Omri, 738
Abzianidze, Lasha, 4881
Adams, Ethan, 46
Agić, Željko, 614
Agirre, Eneko, 3632
Aharonov, Ranit, 719, 4854
Ailem, Melissa, 1478
Aina, Laura, 2936
Aizawa, Akiko, 4208
Aji, Alham Fikri, 2991
Akhtar, Md Shad, 3402, 3454
Alex, John, 732
Alexandrov, Dimitar, 3528
Alinejad, Ashkan, 3022
Alvarez-Melis, David, 1881
Alzantot, Moustafa, 2890
Amrami, Asaf, 4860
Amsterdamer, Yael, 774
Anderson, Peter, 4610
Andor, Daniel, 2562, 5027
Androutsopoulos, Ion, 1849
Angelidis, Stefanos, 3675
Aono, Yushi, 633
Apidianaki, Marianna, 1752
Aralikatte, Rahul, 4909
Arora, Siddhartha, 2322
Artetxe, Mikel, 3632
Artzi, Yoav, 2667, 4470
Augenstein, Isabelle, 4810, 4992
Auli, Michael, 489
Avvaru, Pravalika, 925

Back, Seohyun, 570, 2131
Bagher Zadeh, AmirAli, 150
Bahar, Parnia, 3009
Baheti, Ashutosh, 3970
Bai, Kun, 751
Bak, JinYeong, 956
Bakalov, Anton, 328
Balakrishnan, Anusha, 2333
Balakrishnan, Avinash, 4524
Balasubramanian, Niranjan, 3560, 3783
Baldridge, Jason, 328, 732
Baldwin, Timothy, 844
Ball, Kelsey, 3084
Baly, Ramy, 3528
Bandaru, Sasi Prasanth, 2550
Banerjee, Suman, 2322
Banner, Edward, 4683
Bansal, Mohit, 125, 1369, 2805, 3992, 4067, 4220
Bansal, Trapit, 2824
Bao, Yujia, 1903
Bapna, Ankur, 3028, 4295
Bar-Ilan, Judit, 774
Barbieri, Francesco, 4766
Barbu, Andrei, 2647
Baroni, Marco, 981
Bartolo, Max, 2087
Barzdins, Guntis, 4535
Barzilay, Regina, 1903, 4694
Bauer, Lisa, 4220
Bayas, Nipun, 3350
Bekoulis, Giannis, 2830
Belding, Elizabeth, 3550
Bellet, Aurélien, 1478
Bengio, Yoshua, 2369
Bennett, Andrew, 2667
Berant, Jonathan, 1619
Berg, Tamara, 1369
Berg-Kirkpatrick, Taylor, 1292, 4024, 4673
Berger, Matthew, 2043
Berzak, Yevgeni, 2647
Bhat, Suma, 1510
Bhatia, Shraey, 844
Bhattacharyya, Chiranjib, 1257
Bhattacharyya, Pushpak, 3454
Bi, Wei, 2769
Bickel, Steffen, 4459
Bilu, Yonatan, 719
Bing, Lidong, 665, 2930, 3154, 3855
Bisazza, Arianna, 2871, 4731
Bisk, Yonatan, 93
Bjerva, Johannes, 4992
Black, Alan W, 708

Blanco, Eduardo, 2278, 4395
Blukis, Valts, 2667
Bogoychev, Nikolay, 2991
Bojanowski, Piotr, 2979
Boleda, Gemma, 2936
Bordes, Antoine, 2775
Borsari, Brian, 696
Bos, Johan, 4881
Bosc, Tom, 1522
Bosselut, Antoine, 57
Botha, Jan A., 732
Bouchacourt, Diane, 981
Bouchard, Guillaume, 2087
Bowman, Samuel, 380, 2475, 4998
Boyd-Graber, Jordan, 1077, 3719
Brarda, Sebastian, 4459
Brendel, William, 1097
Brix, Christopher, 3009
Brokos, George, 1849
Brunato, Dominique, 2690
Budzianowski, Paweł, 5016
Buechel, Sven, 4758
Buffone, Anneke, 1146, 1167, 4758
Burns, Kaylee, 2392, 4035
Butnaru, Andrei M., 1084

Cachola, Isabel, 4405
Cai, Deng, 1064
Cai, Jiaxun, 2401
Cai, Xiaoyan, 4306
Callison-Burch, Chris, 1752
Camacho-Collados, Jose, 294, 4766
Cambria, Erik, 2594, 3402
Can, Duy-Cat, 2266
Cao, James, 1597
Cao, Pengfei, 182
Cao, Qian, 3042
Cao, Yixin, 227
Cao, Yuan, 3028
Caragea, Cornelia, 652, 1160
Carbonell, Jaime, 369, 1691, 3285, 4958
Cardie, Claire, 261, 905
Carey, Kate, 696
Carin, Lawrence, 1829, 1839
Carley, Kathleen, 1091
Carton, Samuel, 3497
Casanueva, Iñigo, 5016
Chaganty, Arun, 82
Chai, Joyce, 2627
Chamberlain, Jon, 1926
Chambers, Nate, 3783
Chan, Hou Pong, 3392
Chan, William, 4868
Chandrasekaran, Arjun, 1036
Chang, Baobao, 1402, 2496, 2857
Chang, Kai-Wei, 2890, 4847
Chang, Ming-Wei, 2442
Chang, Serina, 46
Chang, Shih-Fu, 3992, 4013
Chang, Shiyu, 751, 1903, 1980
Chang, Yi, 3090
Chao, Lidia S., 4449
Chattopadhyay, Prithvijit, 1036
Chaudhary, Aditi, 3285
Chauhan, Dushyant, 3454
Che, Wanxiang, 2422
Chen, Derek, 2333
Chen, Di, 665
Chen, Dongmin, 3890
Chen, Fang, 762
Chen, Francine, 1139
Chen, Haiqing, 479
Chen, Hao, 2109
Chen, Hong, 172
Chen, Hong-You, 4834
Chen, Huajun, 986, 2246
Chen, Jianshu, 414
Chen, Jiaze, 4631
Chen, Jing, 4946
Chen, Jingqiang, 4046
Chen, Jingyuan, 162
Chen, Jun, 4057
Chen, Kevin, 2657
Chen, Liwei, 918
Chen, Liyan, 3208
Chen, Lu, 2780
Chen, Mia, 3028
Chen, Ming, 4340
Chen, Mingda, 215
Chen, Nancy, 2916
Chen, Peng, 1597
Chen, Pin-Yu, 4524
Chen, Qi, 684
Chen, Qingcai, 4946
Chen, Wenhu, 414
Chen, Wenqing, 4545, 4565
Chen, Xi, 986, 2246
Chen, Xilin, 4778
Chen, Xilun, 261
Chen, Xinchi, 1576
Chen, Xinpeng, 162
Chen, Xiuying, 4088
Chen, Xu, 227

Chen, Yan-Ying, 1139
Chen, Yi-Shin, 3687
Chen, Ying, 646
Chen, Yubo, 182, 1267
Chen, Yun, 380, 3622
Chen, Yun-Nung, 271, 3813
Chen, Zhan, 3233
Cheng, Ping, 1861
Cheng, Rui, 1030
Cheng, Runxiang, 3643
Cheng, Xiyao, 646
Cheng, Xueqi, 1030
Cherry, Colin, 4295
Cheung, Alvin, 1643
Cheung, Jackie Chi Kit, 817, 1949, 3739
Chi, Hong, 1597
Chi, Ta Chung, 271
Chinnappa, Dhivya, 2278
Chirkova, Nadezhda, 2910
Chiu, Chung-Cheng, 3708
Cho, Kyunghyun, 250, 380, 1173, 1466, 3622, 4890, 4998
Choi, Eunsol, 607, 2174
Choi, Yejin, 93, 607, 2174
Chollampatt, Shamil, 2528
Choo, Jaegul, 2131
Choudhury, Monojit, 3067
Christensen, Janara, 4964
Chu, Eric, 2638
Chua, Tat-Seng, 162
Clark, Kevin, 1914
Clark, Peter, 57, 2381, 4940
Clematide, Simon, 2877
Coavoux, Maximin, 1
Cocos, Anne, 1752
Cognetta, Marco, 2732
Cohen, Shay B., 1, 1797, 4535
Cohen, William, 2369, 4231
Cohn, Trevor, 2850
Colin, Emilie, 937
Collier, Nigel, 1391, 1959, 2266
Collins, Michael, 3698
Conneau, Alexis, 2475, 5039
Cotterell, Ryan, 458, 4425
Cowley, Stephen, 824
Crawford, Eric, 3739
Crego, Josep, 2967
Cuayahuitl, Heriberto, 570
Cui, Baiyun, 4340
Cui, Lei, 2496
Culkin, Ryan, 944
Currey, Anna, 2961

Dagan, Ido, 774, 2412
Dahl, George, 2942
Dahlmeier, Daniel, 3467
Dai, Andrew, 3844
Dai, Hongliang, 2023
Dai, Hui, 4470
Dai, Zeyu, 3305
Dai, Zihang, 856, 2344
Dalvi, Bhavana, 57
Dambre, Joni, 3275
Danescu-Niculescu-Mizil, Cristian, 2818
Dang, Thanh Hai, 2266
Darrell, Trevor, 1380, 4035
Das, Dipanjan, 305, 732, 798
Dasgupta, Shib Sankar, 1871, 2001
Daume III, Hal, 1818
Davey, Disha, 2033
De La Cruz, Noelia, 1949
de Lhoneux, Miryam, 2711, 4992
De Mattei, Lorenzo, 2690
De Neve, Wesley, 3275
Dehouck, Mathieu, 2864
Del Corro, Luciano, 1043
Deleu, Johannes, 2830
Dell'Orletta, Felice, 2690
Demeester, Thomas, 2830, 3275
Demuynck, Kris, 3275
DeNero, John, 881
Deng, Shumin, 986
Deng, Yuntian, 4098
Deng, Zhi-Hong, 4915
Denis, Pascal, 1478, 2864
Denoyer, Ludovic, 5039
Develder, Chris, 2830
Dhingra, Bhuwan, 4231
Diao, Yufeng, 2507
Ding, Keyu, 2540
Ding, Ran, 830
Ding, Yuanyuan, 3901
Dinu, Liviu P., 652
Dixon, Lucas, 2818
Do Dinh, Erik-Lân, 1412
Dolan, Bill, 3970
Dong, Tiansi, 227
Dong, Yue, 817, 3739
Doshi-Velez, Finale, 2721
Dou, Qing, 2496
Dou, Zi-Yi, 621, 4253
Doval, Yerai, 294
Downey, Doug, 2930

Dragut, Eduard, 1991
Dredze, Mark, 4586
Du, Jiachen, 665, 3154
Du, Jinhua, 2216
Du, Yantao, 2109
Dušek, Ondřej, 3981
Duan, Nan, 1597
Duan, Yuguang, 3793
Dubossarsky, Haim, 1732
Duckworth, Daniel, 2942
Duh, Kevin, 1664
Dumančić, Sebastijan, 4816
Dumitru, Bogdan, 652
Dunietz, Jesse, 1691
Durrett, Greg, 1024, 1455, 4503
Dyer, Chris, 3772

Ebrahimi, Mohammad, 762
Eckert, Kai, 3326
Edunov, Sergey, 489
Eger, Steffen, 4415
Eisenstein, Jacob, 1741, 4360
Eisner, Jason, 1325
Ekbal, Asif, 3402, 3454
Elazar, Yanai, 11
Elgohary, Ahmed, 1077, 2890
Elkahky, Ali, 2562
Elliott, Desmond, 2974
ElSherief, Mai, 3550
Emami, Ali, 1949
Erdem, Aykut, 1358
Erdem, Erkut, 1358
Erk, Katrin, 1455
Espinosa Anke, Luis, 294, 4766
Evans, Richard, 4896

Fadaee, Marzieh, 436
Faddoul, Jean Baptiste, 4459
Fan, Feifan, 3433
Fan, Zhenhua, 172
Fares, Murhaf, 1488
Farhadi, Ali, 559
Faruqui, Manaal, 305, 732, 798
Fearn, Wilson, 824
Feng, Chong, 1018
Feng, Shi, 338, 3719, 4752
Feng, Xiaocheng, 523
Feng, Yang, 4284, 4778
Feng, Yansong, 931, 3433
Feng, Zheng, 4663
Ferguson, James, 4964
Fern, Xiaoli, 999, 4952
Fernández-González, Daniel, 1303
Feuerriegel, Stefan, 576
Field, Anjalie, 3570
Firat, Orhan, 3028, 4295
Ford, Nicolas, 2942
Fornaciari, Tommaso, 671
Foster, George, 4295
Freitag, Markus, 3922
Frey, William, 46
Fu, Jianlong, 1338
Fu, Ruiji, 1543
Fu, Tsu-Jui, 4439
Fukumizu, Kenji, 1763
Fukumoto, Fumiyo, 811
Fung, Pascale, 1133, 2799

Görnerup, Olof, 1425
Gómez-Rodríguez, Carlos, 1303, 1314
Gū, Jetic, 401
Gabay, David, 774
Ganchev, Kuzman, 4902
Ganhotra, Jatin, 3834
Gantayat, Neelamadhav, 4909
Gao, Cuiyun, 3120
Gao, Ge, 607
Gao, Jianfeng, 2298, 3813
Gao, Jun, 2769
Gao, Qiaozi, 2627
Gao, Shen, 4088
Gao, Wei, 338
Gao, Yang, 4120
Gao, Zheng, 1776
Garcia-Duran, Alberto, 4816
Gardent, Claire, 937
Gardner, Matt, 1554
Garg, Saurabh, 3078
Garg, Siddhant, 640
Garrette, Dan, 3084
Gasic, Milica, 5016
Gavenčiak, Tomáš, 4921
Ge, Tao, 2496
Gehrmann, Sebastian, 4098
Gelbukh, Alexander, 3402
Gella, Spandana, 968
Geng, Jiahui, 862
Geng, Xinwei, 523
Gerz, Daniela, 316
Ghaeini, Reza, 4952
Ghosal, Deepanway, 3454
Gildea, Daniel, 2226
Gill, Michael, 4890
Gillblad, Daniel, 1425

Gillick, Daniel, 328
Gimpel, Kevin, 215
Giorgi, Salvatore, 1167, 3560
Glass, James, 3528
Glass, Jim, 4829
Glavaš, Goran, 282, 3326
Godin, Fréderic, 3275
Goldberg, Yoav, 11, 4860
Goldin, Gili, 3591
Gong, Hongyu, 1510
Gong, Jingjing, 1576, 2540
González, Fabio A., 3381
Gonzàlez, Edgar, 4964
Gonzalez, Ana, 4810
Gopnik, Alison, 2392
Goyal, Pawan, 2550
Grangier, David, 489
Grant, Erin, 2392
Grave, Edouard, 2979
Greenberg, Nathan, 2824
Griffiths, Tom, 2392
Grissom II, Alvin, 3719
Grivas, Andreas, 2573
Grossman, Eitan, 1732
Grus, Joel, 57
Grzegorczyk, Karol, 1445
Gu, Jiatao, 3622
Gu, Xiaotao, 1215, 2054
Gu, Yihong, 4642
Guder, Cordula, 4459
Gui, Tao, 2540
Guillou, Liane, 4797
Gulordava, Kristina, 2936
Guo, Daya, 1597
Guo, Hongyu, 2742
Guo, Jiang, 4694
Guo, Jiaqi, 1608
Guo, Tszhang, 751
Guo, Xiaoxiao, 1980
Guo, Zhijiang, 1712
Gupta, Nitish, 2152, 2486
Gupta, Sonal, 2787
Gur, Izzeddin, 1702
Gurevych, Iryna, 1412, 3664, 4120, 4415
Guu, Kelvin, 4970

Höhne, Johannes, 4459
Ha, Le An, 4896
Habash, Nizar, 837
Haffari, Gholamreza, 3048
Hajič, Jan, 4921
Hajishirzi, Hannaneh, 559, 3219, 4620
Haldar, Aparajita, 67
Hales, Emily, 824
Han, George, 757
Han, Jialong, 2517
Han, Jiawei, 1215, 2054
Han, Jingguang, 2216
Han, Rujun, 4890
Han, Xu, 993, 2236, 4803
Han, Yo-Sub, 2732
Haponchyk, Iryna, 2310
Hardmeier, Christian, 3003, 4797
Hardt, Daniel, 659
Hardy, Hardy, 768
Hart, Ethan, 2584
Hartmann, Mareike, 582
Hayati, Shirley Anugrah, 925
Hazarika, Devamanyu, 2594
He, Ben, 4482
He, Daqing, 3164
He, Di, 3602
He, He, 2174, 2333
He, Junxian, 1292
He, Lihong, 1991
He, Luheng, 3219
He, Qing, 3198
He, Ruidan, 3467
He, Shexia, 2401
He, Wei, 3930
He, Xiaodong, 2442
He, Yulan, 3154, 3423
He, Zexue, 1402
He, Zhongjun, 391
Heafield, Kenneth, 2961, 2991
Helcl, Jindřich, 3016
Hellwig, Oliver, 2754
Henao, Ricardo, 1829
Henderson, James, 2947
Hendricks, Lisa Anne, 1380, 4035
Herzig, Jonathan, 1619
Higashinaka, Ryuichiro, 633
Hinami, Ryota, 2605
Hirao, Tsutomu, 4177
Ho, Bo-Jhang, 2890
Hoang, Luong, 1049
Holgate, Eric, 1455, 4405
Hong, Yu, 725
Hopkins, Mark, 1533
Hoshen, Yedid, 469
Hospedales, Timothy, 627
Hou, Lei, 227, 1971
Hou, Wenjun, 646

Hou, Yufang, 1938
Hovy, Dirk, 659, 671, 4383
Hovy, Eduard, 1226, 2344
Htut, Phu Mon, 4998
Hu, Guoping, 1543
Hu, Huang, 3233
Hu, J. Edward, 67
Hu, Junjie, 875
Hu, Minghao, 2077
Hu, Sen, 2098
Hu, Shengli, 1153
Hu, Xiao, 1543
Hu, Zhiyuan, 2742
Hua, Yiqing, 2818
Huang, Binxuan, 1091
Huang, Heyan, 1018, 1247
Huang, Jiaji, 1183
Huang, Liang, 1183, 3054, 3188, 4284
Huang, Lifu, 250, 1018, 4013
Huang, Minghui, 4652
Huang, Ruihong, 3305
Huang, Shanshan, 3477
Huang, Shujian, 621
Huang, Xiaolei, 696
Huang, Xuanjing, 791, 1576, 2540
Huang, Yafang, 2923
Huang, Yen-Hao, 3687
Huang, Zhen, 2077
Hui, Kai, 4482
Hui, Siu Cheung, 1565, 2141, 3443, 4492
Hulden, Mans, 2883
Humeau, Samuel, 2775
Hwa, Rebecca, 1723
Hwang, Mei-Yuh, 3803

Iavarone, Benedetta, 2690
Ikizler-Cinbis, Nazli, 1358
Indurthi, Sathish Reddy, 570, 2131
Inui, Kentaro, 1763, 4208
Ionescu, Radu Tudor, 1084
Iyer, Srinivasan, 1643
Iyyer, Mohit, 2174, 3719, 4743

Jégou, Hervé, 2979
Jaakkola, Tommi, 1881
Jacovi, Michal, 719
Jaidka, Kokil, 1146
Jain, Sarthak, 4683
Jamshid Lou, Paria, 4610
Jebara, Tony, 2298
Jhamtani, Harsh, 4024
Ji, Heng, 250, 1018, 2496, 3992, 4013
Jia, Weijia, 2195
Jia, Yantao, 1267
Jiang, Hui, 4725
Jiang, Jing, 1097
Jiang, Tian-Shun, 4970
Jiang, Xin, 3865
Jiang, Yichen, 4067
Jie, Zequn, 162
Jie, Zhanming, 2431
Jin, Di, 3100
Jin, Hongxia, 1011
Jin, Lifeng, 2721
Jin, Yaohui, 4545, 4565
Johnson, Mark, 4610
Joshi, Rishabh, 1257
Joty, Shafiq, 4196
Joulin, Armand, 2979
Junczys-Dowmunt, Marcin, 2991
Jurafsky, Dan, 82, 3570
Jurgens, David, 33
Jyothi, Preethi, 3078, 4743

Kambhatla, Nishant, 869
Kamigaito, Hidetaka, 4177
Kang, Dongyeop, 4940
Kang, Jaewoo, 565
Kang, Yangyang, 3654
Kann, Katharina, 3254
Kano, Ryuji, 1139
Kantor, Yoav, 719
Karadzhov, Georgi, 3528
Karlekar, Sweta, 2805
Kartsaklis, Dimitri, 1391, 1959
Karuturi, Pradeep, 1097
Kasewa, Sudhanshu, 4977
Kato, Tsuneaki, 594
Katti, Anoop R, 4459
Katz, Boris, 2647
Kaushik, Divyansh, 5010
Kavuluru, Ramakanth, 3132
Ke, Qifa, 3901
Kedzie, Chris, 46, 1818
Keith, Katherine, 4575
Kementchedjhieva, Yova, 458, 582
Khan, Samin, 1128
Khanpour, Hamed, 1160
Khapra, Mitesh M., 2322, 3950
Khashabi, Daniel, 2065
Khot, Tushar, 2381, 4940
Kiela, Douwe, 1466
Kim, Hyunjae, 565
Kim, Jihie, 2131

Kim, Joo-Kyung, 894
Kim, Samuel, 1122
Kim, Young-Bum, 894
Kim, Yunsu, 862
King, Irwin, 3120, 3392
Kiros, Jamie, 4868
Klakow, Dietrich, 4316
Kleinberg, Bennett, 3581
Kliger, Doron, 3570
Knight, Kevin, 250
Knowles, Rebecca, 3034
Ko, Miyoung, 565
Kobayashi, Hayato, 4165
Kobayashi, Sosuke, 1763
Kodner, Jordan, 501
Koehn, Philipp, 3034
Kolchinski, Y. Alex, 1115
Koncel-Kedziorski, Rik, 4620
Kondratyuk, Daniel, 4921
Konopnicki, David, 4854
Konstas, Ioannis, 1643, 3981
Koo, Sangjun, 714
Korhonen, Anna, 282, 316
Kotlerman, Lili, 719
Kovaleva, Olga, 4875
Kozareva, Zornitsa, 804, 887
Kratzwald, Bernhard, 576
Krishna, Amrith, 2550
Krishna, Kalpesh, 4743
Kruschwitz, Udo, 1926
Kryściński, Wojciech, 1808
Kulkarni, Vivek, 3518
Kulmizev, Artur, 4881
Kumar, Anuj, 2787
Kumar, Sandeep, 4002
Kundu, Souvik, 4243
Kurdziel, Marcin, 1445
Kurmi, Vinod Kumar, 4002
Kwiatkowski, Tom, 559
Kwon, Soon Chan, 2732

Läubli, Samuel, 4791
Labaka, Gorka, 3632
Labutov, Igor, 1676
Lai, Guokun, 2344
Lai, Jianhuang, 3602, 3612
Lalor, John, 4711
Lam, Wai, 3855
Lamm, Matthew, 82
Lample, Guillaume, 2475, 5039
Lamprinidis, Sotiris, 659
Lan, Man, 2256
Lan, Xiaohan, 349
Lapata, Mirella, 779, 1554, 1797, 2453, 3675
Lau, Jey Han, 844
Lauscher, Anne, 3326
Lavee, Tamar, 719
Le, Hoang-Quynh, 2266
Le, Quoc, 1914, 3844
Lebanoff, Logan, 3508, 4131
Lee, Cheng-Syuan, 4834
Lee, Fei-Tzin, 46
Lee, Gary Geunbae, 714
Lee, Hung-yi, 4187
Lee, Jason, 1173
Lee, Jay Yoon, 4958
Lee, Jinhyuk, 565
Lee, Kenton, 3772
Lee, Kuang-Chih, 2256
Lee, Lillian, 1277
Lee, Nayeon, 1133
Lee, Wee Sun, 3467
Lee, Yong Jae, 3643
Leeuwenberg, Artuur, 1237
Lei, Jie, 1369
Lei, Tao, 1056, 4470
Lei, Zeyang, 3110
Levin, Lori, 1691, 3285
Levine, Joshua, 2043
Lewis, Mike, 968, 2152, 2787
Lewis, Patrick, 2087
Li, Bo, 1861
Li, Canjia, 4482
Li, Changliang, 3824
Li, Chengjiang, 227
Li, Chenliang, 745
Li, Dongsheng, 2077
Li, Edward, 4964
Li, Hang, 3865
Li, Haoran, 4154
Li, Hui, 3360
Li, Irene, 3911
Li, Jia, 3844
Li, Jian, 2897
Li, Jing, 2517, 3120
Li, Jingyuan, 678
Li, Jiwei, 3970
Li, Jiyi, 811
Li, Juanzi, 227, 1971
Li, Juntao, 3890
Li, Junyi Jessy, 4405
Li, Lei, 4631
Li, Liang, 3824

Li, Lihong, 2298
Li, Linlin, 2401
Li, Muyu, 2985
Li, Peng, 2236
Li, Piji, 3855
Li, Ping, 2119
Li, Qing, 1338
Li, Ruoyu, 3143
Li, Shoushan, 646, 3654
Li, Sujian, 684, 962, 4737
Li, Victor O. K., 3622
Li, Victor O.K., 380
Li, Wei, 1787, 4078, 4514
Li, Weikang, 4514
Li, Wenhao, 3143, 3960
Li, Wenjie, 3154, 4316
Li, Xiangju, 4752
Li, Xiaolin, 4663
Li, Xiujun, 2298, 3813
Li, Xu, 2119
Li, Yi, 1183
Li, Yingming, 4340
Li, Yitong, 1839
Li, Zeyu, 4847
Li, Zhifei, 3803
Li, Zhoujun, 4057
Li, Zichao, 3865
Li, Zifan, 1653, 3911
Li, Zuchao, 2401
liang, di, 2540
Liang, Dong, 1576
Liang, Jennifer, 2357
Liang, Paul Pu, 150
Liang, Percy, 82, 2174, 2333, 4970
Liang, Yingyu, 601
Liao, Keng-Te, 4834
Liao, Lejian, 1018
Liao, Yi, 3855
Libovický, Jindřich, 3016
Lin, Bill Yuchen, 2012, 3477
Lin, Chin-Yew, 3761, 3879
Lin, Dekang, 2109
Lin, Fen, 3198, 4642
Lin, Fengli, 2185
Lin, Hongfei, 2507
Lin, Junyang, 702, 2985, 3940, 4554
Lin, Leyu, 4642
Lin, Qian, 4273
Lin, Shou-de, 4834
Lin, Xi Victoria, 3243
Lin, Yankai, 358
Lin, Zi, 3793
Linzen, Tal, 1192, 4704
Lipton, Zachary C., 2904, 5010
Liu, Changsheng, 1723
Liu, Evan, 4970
Liu, Fei, 2850, 3508, 4131
Liu, Fenglin, 137
Liu, Gongshen, 2401
Liu, Hsien-Chi Toby, 3687
Liu, Jiahua, 2109
Liu, Jing, 3930
Liu, Jingjing, 3813
Liu, Kang, 182, 1267
Liu, Lixing, 696
Liu, Liyuan, 1215, 2054
Liu, Lizhen, 1543
Liu, Qun, 2997, 4284
Liu, Rijia, 2023
Liu, Shaobo, 1030
Liu, Shengping, 182
Liu, Tianshang, 4154
Liu, Tianyi, 2195
Liu, Tianyu, 1402, 3729
Liu, Tie-Yan, 3602, 3612
Liu, Ting, 523, 1543, 1608, 2422
Liu, Xiao, 1247
Liu, Xiaojiang, 1064, 2769
Liu, Xiaozhong, 1776, 3654
Liu, Xin, 4946
Liu, Yahui, 2769
Liu, Yang, 447, 533, 1554
Liu, Yijia, 2422
Liu, Yizhu, 4110
Liu, Yuanxin, 137
Liu, Yuwei, 4652
Liu, Zhengzhong, 1226
Liu, Zhiyuan, 227, 358, 993, 1971, 2236, 3540, 4642, 4803
Liu, Ziyin, 150
Livescu, Karen, 215
Lobacheva, Ekaterina, 2910
Lopez, Adam, 2573
Lou, Jian-Guang, 1608
Louis, Annie, 4350
Lu, Daohe, 4946
Lu, Di, 4013
Lu, Hao, 172
Lu, Wei, 204, 1011, 1712, 2012, 2431
Lu, Weiyi, 4822
Luan, Huanbo, 533
Luan, Yi, 3219

Lukasik, Michal, 3371
Lund, Jeffrey, 824
Luo, Bingfeng, 3233
Luo, Fuli, 1402
Luo, Hongyin, 4829
Luo, Jiebo, 1338
Luo, Kangqi, 2185
Luo, Liangchen, 702
Luo, Xusheng, 2185
Luo, Zhiyi, 3477, 4110
Luo, Zhunchen, 1247
Luong, Minh-Thang, 1914
Luu, Anh Tuan, 1565, 2141, 3443, 4492
Lv, Qingsong, 349
Lv, Xin, 1971
Lynn, Veronica, 3560
Lyu, Michael R., 2897, 3120
Lyu, Yajuan, 1787, 3930, 4078

Müller, Mathias, 4263
Màrquez, Lluís, 4196
Ma, Dehong, 4737
Ma, James, 3911
Ma, Ji, 4902
Ma, Lin, 162
Ma, Mingbo, 3054, 3188
Ma, Shuming, 3729, 4554
Ma, Wei-Yun, 4439
Ma, Xutai, 1664
Ma, Yanjun, 3930
Ma, Zhuang, 3698
Macherey, Wolfgang, 4295
Maddela, Mounica, 3749
Magri, Giorgio, 3265
Mahajan, Vibhuti, 2584
Maharjan, Suraj, 3381
Majumder, Navonil, 3402
Makarov, Peter, 2877
Malca, Rivka, 2700
Mallinson, Jonathan, 2453
Mani, Senthil, 4909
Manning, Christopher D., 82, 1914, 2205, 2369
Mansimov, Elman, 1173
Mansouri Bigvand, Anahita, 869
Mansur, Mairgup, 2857
Mao, Yongyi, 2742
Marshall, Iain, 2837
Marshall, Iain J, 4683
Martins, André F. T., 905
Marujo, Luis, 1097
Marvin, Rebecca, 1192
Masumura, Ryo, 633
Matsumoto, Yuji, 1630
Mayhew, Stephen, 3073
Mazaitis, Kathryn, 4231
Mazare, Pierre-Emmanuel, 2775
McCallum, Andrew, 2824, 5027
McDonald, Ryan, 1849
McKeown, Kathleen, 1818
McKeown, Kathy, 46
Mehta, Sachin, 4620
Mehta, Sanket Vaibhav, 4958
Mei, Qiaozhu, 3497
Mei, Tao, 1338
Melas-Kyriazi, Luke, 757
Meng, Fandong, 4449
Meng, Rui, 3164
Mesgar, Mohsen, 4328
Meyer, Christian M., 4120
Michel, Paul, 543
Miculicich, Lesly, 2947
Mihalcea, Rada, 2594
Mihaylov, Todor, 2381
Mikolov, Tomas, 2979
Miller, Timothy, 2721
Miller, Tristan, 3664
Min, Martin Renqiang, 1839
Miranda, Sebastião, 4535
Mirkin, Shachar, 719
Misra, Dipendra, 2442, 2667
Mitamura, Teruko, 1226
Mitchell, Tom, 425, 1676
Mitkov, Ruslan, 4896
Miura, Yasuhide, 1139
Miwa, Makoto, 2843
Moens, Marie-Francine, 1237
Moghe, Nikita, 2322
Mohit, Mrinal, 2787
Montes, Manuel, 3381
Monz, Christof, 436, 4731
Mooney, Raymond, 1347
Moorthy, Sridhar, 1128
Moosavi, Nafise Sadat, 193
Morency, Louis-Philippe, 150
Mortensen, David R., 3285
Moschitti, Alessandro, 1056, 1070, 2162, 2310
Moshkowich, Guy, 719
Mozes, Maximilian, 3581
Mrkšić, Nikola, 282
Mu, Jiaqi, 1510
Mudgal, Sidharth, 601
Mueller, David, 1024
Mukherjee, Subhabrata, 22

Mukherjee, Tanmoy, 627
Munkhdalai, Tsendsuren, 4711
Murawaki, Yugo, 4371
Myanganbayar, Battushig, 2647
Mysore Gopinath, Abhijith Athreya, 850

Nagata, Masaaki, 4177, 4599
Nagesh, Ajay, 2043
Nakashole, Ndapa, 512
Nakov, Preslav, 1056, 3528, 4196
Nallapati, Ramesh, 830
Namboodiri, Vinay, 4002
Narayan, Shashi, 1, 1797
Nehrdich, Sebastian, 2754
Nema, Preksha, 3950
Nematzadeh, Aida, 2392
Nenkova, Ani, 774
Neubig, Graham, 369, 425, 543, 856, 875, 925, 1292, 3285, 4772
Neumann, Mark, 1499
Ney, Hermann, 862, 3009
Ng, Hwee Tou, 2528, 3467, 4243
Ngo, Gia H, 2916
Nguyen, Minh, 2916
Nguyen, Patrick, 3708
Nguyen, Thien Huu, 4822
Ni, Xiaochuan, 3901
Nicosia, Massimo, 1070
Niculae, Vlad, 905
Nie, Feng, 3879
Niebles, Juan Carlos, 2657
Niepert, Mathias, 3360, 4816
Niklasson, Eyvind, 2667
Nivre, Joakim, 2711
Noraset, Thanapon, 2930
Norouzi, Mohammad, 2942

O'Connor, Brendan, 4575
Oepen, Stephan, 1488
Oh, Alice, 956, 3316
Ohkuma, Tomoko, 1139
Olivier, Raphael, 925
Orbach, Matan, 719
Orr, Walker, 999
Ostendorf, Mari, 3219
Otani, Naoki, 2465
Ott, Myle, 489, 5039
Ouchi, Hiroki, 1630

Padmakumar, Aishwarya, 1347
Pampari, Anusri, 2357
Pan, Jennifer, 3570
Pan, Rong, 3879
Panchal, Vishal, 2033
Panwar, Naveen, 4909
Pappas, Nikolaos, 2947
Parekh, Tanmay, 3078
Parikh, Ankur, 559
Parikh, Devi, 1036
Park, Ji Ho, 2799
Park, Sungjoon, 3316
Parker Jones, Oiwi, 4929
Parmanto, Bambang, 3164
Parthasarathi, Prasanna, 690
Pasunuru, Ramakanth, 125
Pasupat, Panupong, 4970
Patel, Pinal, 2033
Pathak, Parth, 2033
Patro, Badri Narayana, 4002
Patton, Desmond, 46
Paulus, Romain, 1808
Paun, Silviu, 1926
Pavlick, Ellie, 67, 305, 1752
Pei, Jiaxin, 745
Peng, Hao, 1203
Peng, Jian, 1215, 1608, 2357
Peng, Minlong, 2540
Peng, Yuxing, 2077
Peters, Matthew, 1499
Petrak, Johann, 900
Petrochuk, Michael, 554
Pezeshkpour, Pouya, 3208
Pham, Hieu, 856, 4772
Pham, Minh Quang, 2967
Pilehvar, Mohammad Taher, 1391, 1959, 2266
Pineau, Joelle, 690
Ping, Wei, 1183
Pinkal, Manfred, 2616
Pinter, Yuval, 1741
Pitler, Emily, 2562
Plank, Barbara, 614
Platanios, Emmanouil Antonios, 425
Poesio, Massimo, 1926
Pogacar, Ruth, 1128
Pokle, Ashwini, 2657
Poliak, Adam, 67
Polymenakos, Lazaros, 3834
Ponti, Edoardo Maria, 282, 316
Ponza, Marco, 1043
Ponzetto, Simone Paolo, 3326
Poon, Hoifung, 1891
Popat, Kashyap, 22
Poria, Soujanya, 2594, 3402, 3454

Potts, Christopher, 1115
Pou-Prom, Chloé, 2812
Prabhu, Viraj, 1036
Prabhumoye, Shrimai, 708
Pratapa, Adithya, 3067
Prayaga, Sai Suman, 1257
Preoţiuc-Pietro, Daniel, 1167, 4405
Prokhorov, Victor, 1391
Purschke, Christoph, 4383

Qi, Fanchao, 358
Qi, Ji, 3824
Qi, Jianzhong, 1005
Qi, Peng, 2205, 2369
Qian, Jing, 3550
Qin, Bing, 523, 2422
Qin, Guanghui, 3761
Qin, Tao, 3602, 3612
Qiu, Liwei, 2023
Qiu, Xipeng, 1576
Qu, Meng, 3198

Rabinovich, Ella, 3591, 4854
Radev, Dragomir, 1653, 3911
Raghavan, Preethi, 2357
Rai, Pranav, 3664
Raison, Martin, 2775
Rajendran, Janarthanan, 3834
Ram, Dhananjay, 2947
Ramadan, Osman, 5016
Ranzato, Marc'Aurelio, 5039
Rao, Yanghui, 4652
Rappoport, Ari, 738
Rashid, Farzana, 4395
Rastegari, Mohammad, 4620
Ravi, Sujith, 804, 887
Ravikumar, Pradeep, 4524
Ravishankar, Vinit, 4881
Rawlins, Kyle, 4717
Ray, Swayambhu Nath, 1871, 2001
Reichart, Roi, 238, 316, 2700
Reisswig, Christian, 4459
Ren, Liliang, 2780
Ren, Teng, 2054
Ren, Xiang, 1215, 2054
Ren, Xuancheng, 137, 2985, 3940, 4306
Resnick, Paul, 3497
Riedel, Sebastian, 2087, 4977
Rieman, Daniel, 1167
Riesa, Jason, 328
Rieser, Verena, 3981
Rinott, Ruty, 2475
Rios, Annette, 4263
Rios, Anthony, 3132
Ritter, Alan, 3970
Roberts, Angus, 900
Rocktäschel, Tim, 2087
Rodriguez, Pedro, 3719
Rohde, Hannah, 4350
Rohrbach, Anna, 4035
Rohrbach, Marcus, 968
Roman, Shanelle, 3911
Romanov, Alexey, 1586, 4875
Romeo, Salvatore, 1056
Ronen, Hadar, 774
Rong, Shu, 172
Ross, Candace, 2647
Roth, Benjamin, 1109
Roth, Dan, 105, 501, 2065, 2486, 3073
Rouhizadeh, Masoud, 1146
Roy, Deb, 2638
Roy, Scott, 3922
Ruan, Huibin, 725
Ruder, Sebastian, 458
Rudinger, Rachel, 67, 944, 1664, 4717
Rudzicz, Frank, 2812
Rumshisky, Anna, 4875
Rush, Alexander, 757, 1049, 3174, 4098
Russell, Bryan, 1380
Ruths, Derek, 817
Ruvini, Jean David, 2584
Ryu, Seonghan, 714

Søgaard, Anders, 458, 582, 4810, 4992
Sabharwal, Ashish, 2381, 4940
Sachan, Mrinmaya, 425
Sadeh, Norman, 850
Sadiya, Sari, 2627
Saeidi, Marzieh, 2087
Saenko, Kate, 4035
Saggion, Horacio, 4766
Sahin, Gozde Gul, 5004
Sahu, Gaurav, 2550
Salakhutdinov, Ruslan, 2369, 4231
Sammons, Mark, 3073
Sankaran, Anush, 4909
Santra, Bishal, 2550
Saptono, Andi, 3164
Saravia, Elvis, 3687
Sarawagi, Sunita, 640
Sarkar, Anoop, 401, 869, 3022
Satoh, Shin'ichi, 2605
Satuluri, Pavankumar, 2550
Savarese, Silvio, 2657

Schütze, Hinrich, 3060, 3254
Scherer, Stefan, 696
Schiller, Benjamin, 3664
Schluter, Natalie, 2793, 4935
Schmitt, Martin, 1109
Schockaert, Steven, 294, 4766
Schreiber, Konrad, 1109
Schuler, William, 2679, 2721
Schwartz, H. Andrew, 1146, 1167, 3350, 3560
Schwartz, Lane, 2721
Schwartz, Roy, 93, 1203
Schwenk, Holger, 2475
Sedoc, Joao, 4758
Sekine, Satoshi, 4208
Senellart, Jean, 2967
Sennrich, Rico, 2453, 4263, 4791
Seo, Minjoon, 559
Seonwoo, Yeon, 3316
Seppi, Kevin, 824
Sha, Fei, 1478
Sha, Lei, 684
ShafieiBavani, Elaheh, 762
Shah, Darsh, 1056, 4694
Shah, Rushin, 2787
Shain, Cory, 2679
Shang, Jingbo, 1215, 2054
Shang, Lifeng, 3865
Shankar, Shiv, 640
Shao, Chenze, 4778
Shapira, Ori, 774
Shapiro, Pamela, 4425
Sharma, Vishnu Dutt, 2550
Sharma, Yash, 2890
Shatkhin, Max, 2667
Shavarani, Hassan S., 401
Shawe-Taylor, John, 2837
Shechtman, Eli, 1380
Sheinin, Vadim, 918, 931
Shekhar, Leena, 3783
Sheldon, Mike, 2087
Shen, Chenlin, 3654
Shen, Dinghan, 1829, 1839
Shen, Lei, 4284
Shen, Xiaoyu, 4316
Shen, Yikang, 3739
Shi, Chen, 684
Shi, Ge, 1018
Shi, Hanyuan, 3477
Shi, Haoyue, 4631
Shi, Shuming, 2769, 3855, 3890, 4253
Shi, Tianze, 1277
Shieber, Stuart, 3174
Shillingford, Brendan, 4929
Shimura, Kazuya, 811
Shin, Jamin, 2799
Shindo, Hiroyuki, 1630
Shinohara, Yusuke, 633
Shivade, Chaitanya, 1586
Shnayderman, Ilya, 4854
Shore, Todd, 2288
Shpitser, Ilya, 4586
Si, Luo, 2401, 3654
Siahbani, Maryam, 3022
Siddhant, Aditya, 2904
Silberer, Carina, 2616
Silfverberg, Miikka, 2883
Simianer, Patrick, 881
Singh, Gaurav, 2837
Singh, Munindar, 3412
Singh, Sameer, 2087, 3208
Singh, Satinder, 3834
Sinha, Koustuv, 817
Sitaram, Sunayana, 3067
Sivic, Josef, 1380
Skantze, Gabriel, 2288
Skiena, Steve, 3518
Slaff, Barry, 4758
Slonim, Noam, 719, 4854
Smith, Aaron, 2711
Smith, Laura, 1146
Smith, Noah A., 369, 1203, 3772
Socher, Richard, 1808, 3243
Sohrab, Mohammad Golam, 2843
Solorio, Thamar, 3381
Son, Youngseo, 3350
Song, Kaiqiang, 4131
Song, Kaisong, 4752
Song, Linfeng, 2226
Song, Wei, 1543
Song, Xingyi, 900
Song, Yan, 2517, 3120, 3890, 4088
Song, Yangqiu, 2023
Sorensen, Jeffery, 2818
Soto, Alvaro, 2657
Spector, Artem, 4854
Spirling, Arthur, 4890
Srivastava, Mani, 2890
Srivatsan, Akshay, 4673
Stab, Christian, 3664
Staliūnaitė, Ieva, 4350
Stanovsky, Gabriel, 1533, 2412
Steedman, Mark, 5004

Steinheber, Simon, 1109
Stenetorp, Pontus, 4977
Stewart, Ian, 4360
Stolcke, Andreas, 2764
Stone, Peter, 1347
Stoyanov, Veselin, 2475
Straka, Milan, 4921
Strapparava, Carlo, 3296
Strube, Michael, 193, 4328
Strubell, Emma, 5027
Stymne, Sara, 2711
Su, Hui, 4316
Su, Jian, 3443
Su, Jinsong, 447
su, jinsong, 4273
Su, Qi, 2985, 4554
Su, Shang-Yu, 3813
Su, Songwen, 3487
Su, Yu, 414, 1702
Sugawara, Saku, 4208
Sui, Zhifang, 1402, 2496
Sulem, Elior, 738
Suleman, Kaheer, 1949
Sumita, Eiichiro, 4785
Sun, Changlong, 3654
Sun, Changzhi, 2256
Sun, Haitian, 4231
Sun, Jingyuan, 1435
Sun, Le, 4482
Sun, Maosong, 358, 533, 993, 2109, 2236, 3143, 3540, 3960, 4642, 4803
Sun, Mingming, 2119
Sun, Shiliang, 2256
Sun, Weiwei, 3793
Sun, Xiao, 678
Sun, Xingwu, 3930
Sun, Xu, 137, 684, 702, 1103, 2985, 3940, 4306, 4554
Sun, Yibo, 1597
Sun, Yingfei, 4482
Sun, Zhanling, 986
Sun, Zhiqing, 4915
Surdeanu, Mihai, 2043
Suzuki, Jun, 1763, 4599
Swayamdipta, Swabha, 3772
Sznajder, Benjamin, 4854
Szolovits, Peter, 3100

Tadepalli, Prasad, 999, 4952
Takase, Sho, 4599
Talukdar, Partha, 1257, 1871, 2001
Tan, Xu, 3602
Tandon, Niket, 57
Tang, Buzhou, 4946
Tang, Da, 2298
Tang, Duyu, 1597
Tang, Gongbo, 4263
Tang, Qingming, 215
Taniguchi, Motoki, 1139
Tao, Chongyang, 3233, 4088
Taraborelli, Dario, 2818
Tay, Yi, 1565, 2141, 3443, 4492
Teichert, Adam, 944
Tekiroglu, Serra Sinem, 3296
Tenney, Ian, 305
Thain, Nithum, 2818
Thomas, James, 2837
Thomson, Sam, 1203, 3772
Tian, Fei, 3602, 3612
Tomasic, Anthony, 925
Tran, Ke, 4731
Trischler, Adam, 1949
Troiano, Enrica, 3296
Tsai, Chen-Tse, 2065
Tseng, Bo-Hsiang, 5016
Tsvetkov, Yulia, 3570
Tu, Cunchao, 3540
Tu, Zhaopeng, 2897, 2997, 4253, 4449
Tump, Clara, 2871
Tymoshenko, Kateryna, 2162

Ultes, Stefan, 5016
Ungar, Lyle, 1146, 1167, 4758
Upadhyay, Shyam, 501, 2486
Uryupina, Olga, 2310
Utiyama, Masao, 4785
Utsumi, Akira, 4840
Uva, Antonio, 2310

Vázquez, Marynel, 2657
van de Meent, Jan-Willem, 4683
van der Goot, Rob, 4984
van der Vegt, Isabelle, 3581
Van Durme, Benjamin, 67, 944, 1664, 4717
van Hoof, Herke, 3739
van Noord, Gertjan, 4984
van Schijndel, Marten, 4704
Vania, Clara, 2573
Vanmassenhove, Eva, 3003
Varab, Daniel, 4935
Varia, Siddharth, 46
Vashishth, Shikhar, 1257
Velldal, Erik, 1488
Venturi, Giulia, 2690

Verga, Patrick, 2824, 5027
Verma, Rohil, 1122
Vetrov, Dmitry, 2910
Vijayaraghavan, Prashanth, 2638
Vilares, David, 1314
Vincent, Pascal, 1522
Viswanath, Pramod, 1510
Vlachos, Andreas, 768
Volk, Martin, 4791
Voss, Clare, 3992
Vu, Sinh T., 2266
Vu, Thuy-Trang, 3048
Vucetic, Slobodan, 1991
Vulić, Ivan, 282, 316

Wallace, Byron C., 2837, 4683
Wallace, Eric, 3719
Walter, David, 1122
Wan, Dadong, 2216
Wan, Xiaojun, 3793
Wang, Bailin, 204, 1011
Wang, Changhan, 1466
Wang, Cheng, 3360
Wang, Chong, 2298
Wang, Chunqi, 479
Wang, Daling, 338, 4752
Wang, Dingmin, 2517
Wang, Dingquan, 1325
Wang, Dongxu, 1653, 3911
Wang, Feng, 975
Wang, Fu Lee, 4652
Wang, Guanying, 2246
Wang, Hai, 1891
Wang, Houfeng, 137, 684, 3729, 4737
Wang, Jian, 2507
Wang, Jingjing, 3654
Wang, Jinpeng, 3761, 3879
Wang, Le, 4482
Wang, Lei, 1064
Wang, Longyue, 2997
Wang, Lu, 4142
Wang, Oliver, 1380
Wang, Rui, 4785
Wang, Ruoxu, 2246
Wang, Shaonan, 115, 1435
Wang, Shi, 3930
Wang, Sida I., 4470
Wang, Su, 1455
Wang, Wei, 4847
Wang, Weichao, 338
Wang, Weikang, 3803
Wang, Wenhui, 2857
Wang, Wenting, 2256
Wang, William Yang, 414, 1980, 3518, 3550
Wang, Xin, 414
Wang, Xing, 4253
Wang, Xinyi, 856, 4772
Wang, Xuan, 3154
Wang, Xuening, 3761
Wang, Yan, 1064
Wang, Yaushian, 4187
Wang, Yicheng, 4220
Wang, Yining, 2955
Wang, Yizhong, 962
Wang, Yong, 3622
Wang, Yongkun, 4545, 4565
Wang, Yongzhen, 1776
Wang, Yu, 1011
Wang, Yuanzhuo, 1787, 4078
Wang, Yucheng, 791
Wang, Zhichun, 349
Wang, Zhiguo, 918, 931, 2226
Wang, Zijian, 33
Wang, Ziyun, 4803
Washio, Koki, 594
Watcharawittayakul, Sedtawut, 4725
Watson, Daniel, 837
Way, Andy, 2216, 2997, 3003
Webber, Bonnie, 4350
Weber, Noah, 3783
Webster, Kellie, 2562
Wei, Furu, 779, 2077, 2496
Wei, Wan, 2109
Wei, Wei, 3844
Wei, Zhongyu, 791
Weikum, Gerhard, 22, 1043
Weinshall, Daphna, 1732
Weiss, David, 328, 4902, 5027
Wen, Huating, 447
Wen, Tsung-Hsien, 5016
Wen, Weiming, 3487
Wharton, Veronica, 1752
White, Aaron Steven, 67, 4717
Whitehead, Spencer, 3992, 4013
Wieland, Hannah, 1412
Williams, Adina, 2475
Wilson, Shomir, 850
Wintner, Shuly, 3570, 3591
Wiseman, Sam, 1049, 3174
Witbrock, Michael J., 4524
Wojtowicz, Zachary, 4673
Wolf, Lior, 469
Wong, Derek F., 4449

Wong, Raymond, 762
Wood-Doughty, Zach, 4586
Woolley, Joshua, 696
Wu, Chien-Sheng, 1133
Wu, Di, 2507
Wu, Hao, 4711
Wu, Hua, 391
Wu, Junlin, 3687
Wu, Kewen, 2256
Wu, Lijun, 3602, 3612
Wu, Lingfei, 918, 931, 4524
Wu, Lingfeng, 2764
Wu, Minghao, 2850
Wu, Shijie, 4425
Wu, Wei, 3729
wu, wei, 3233
Wu, Xianchao, 3233
Wu, Yonghui, 3028, 3708
Wu, Yu, 4057
Wu, Yuanbin, 2256
Wu, Yuexin, 2465
Wu, Yunfang, 4514
Wuebker, Joern, 881

Xia, Congying, 3090
Xia, Qiaolin, 1402
Xiang, Bing, 830
Xiao, Chaojun, 3540
Xiao, Liqiang, 4545, 4565
Xiao, Xinyan, 1787, 4078
Xie, Haoran, 4652
Xie, Jiateng, 369
Xie, Jinghui, 587
Xie, Jun, 447
Xie, Kaige, 2780
Xie, Qizhe, 2344
Xie, Ruobing, 358, 4642
Xie, Tao, 1608
Xin, Ji, 993
Xin, Yingwei, 2584
Xiong, Caiming, 1808, 3243
Xiong, Chenyan, 1226
Xiong, Deyi, 3042, 4273
Xiong, Wayne, 2764
Xiong, Wenhan, 1980
Xu, Bo, 975, 2507
Xu, Can, 3233
Xu, Fangli, 4524
Xu, Frank F., 3477
Xu, Jiacheng, 4503
Xu, Jiaming, 975
Xu, Jingfang, 533, 2955
Xu, Jingjing, 702, 1103, 3940, 4306
Xu, Jungang, 4482
Xu, Kan, 2507
Xu, Kun, 918, 931, 4524
Xu, Mingbin, 4725
Xu, Ruifeng, 665, 3154
Xu, Ruochen, 2465
Xu, Wei, 3749
Xu, Xinnuo, 3981
Xu, Yang, 725, 1128
Xue, Nianwen, 3339

Yadav, Deshraj, 1036
Yagcioglu, Semih, 1358
Yaghoobzadeh, Yadollah, 3060
Yamada, Makoto, 627
Yan, Jun, 4642
Yan, Rui, 785, 3890, 4088
Yan, Xiaohui, 3090
Yan, Xifeng, 414, 1702
Yan, Zhao, 4057
Yaneva, Victoria, 4896
Yang, Baosong, 2897, 4449
Yang, Bishan, 1676
Yang, Cheng, 3960
Yang, Haijun, 4946
Yang, Hang, 1267
Yang, Jingfeng, 962
Yang, Kai, 1653, 3911
Yang, Liang, 2507
Yang, Min, 3110
Yang, Nan, 2077
Yang, Pengcheng, 1103, 4554
Yang, Shaohua, 2627
Yang, Wei, 1608
Yang, Yang, 3423
Yang, Yi, 912
Yang, Yilin, 3054
Yang, Yiming, 2465
Yang, Zhihao, 2507
Yang, Zhilin, 369, 2369
Yao, Jian, 2769
Yao, Jianmin, 725
Yao, Jin-Ge, 3761, 3879
Yao, Jin-ge, 785
Yao, Qingning, 3911
Yao, Yiqun, 975
Yao, Yuan, 4803
Yasunaga, Michihiro, 1653, 3911
Yates, Andrew, 22, 4482
Yatskar, Mark, 2174
Yavuz, Semih, 1702, 3708

Ye, Hai, 4142
Ye, Jianbo, 3110
Ye, Junting, 3518
Yen, Ian En-Hsu, 4524
Yi, Xiaoyuan, 3143, 3960
Yih, Wen-tau, 57, 1499, 2174, 2442
Yin, Chuandong, 1005
Yin, Jian, 1597
Yin, Pengcheng, 925, 4772
Yin, Wenpeng, 105
Yin, Yongjing, 447
Yokoi, Sho, 1763
Youssef, Paul, 4415
Yu, Dong, 414
Yu, Dongfei, 1338
Yu, Hong, 4711
Yu, Hwanjo, 714
Yu, Jianfei, 1097
Yu, Juntao, 1926
Yu, Kai, 2780
Yu, Licheng, 1369
Yu, Mo, 751, 918, 1903, 1980
Yu, Pengfei, 2236, 4803
Yu, Philip, 3090
Yu, Seunghak, 570, 2131, 2310
Yu, Tao, 1653, 3911
Yu, Xiaodong, 3073
Yu, Xiaoming, 1030
Yu, Zhou, 3487, 3643
Yuille, Alan, 172
Yun, Seongjun, 565
Yvon, François, 2967

Zaheer, Manzil, 4231
Zalmout, Nasser, 837
Zamani, Mohammadzaman, 3560
Zang, Xiaoxue, 2657
Zellers, Rowan, 93
Zeng, Jiali, 447
Zeng, Jichuan, 3120
Zeng, Qi, 702, 4306
Zens, Richard, 3371
Zettlemoyer, Luke, 554, 607, 1499, 1643, 2174, 3772
Zhai, Feifei, 533, 2955
Zhang, Biao, 4273
Zhang, Boliang, 250, 1018
Zhang, Bowen, 1478
Zhang, Chenwei, 3090
Zhang, Dongmei, 1608
Zhang, Dongxiang, 1064
Zhang, Dongyu, 2507
Zhang, Fangfang, 785
Zhang, Haisong, 2517, 3890
Zhang, Han, 3803
Zhang, Helen, 2043
Zhang, Honglun, 4545, 4565
Zhang, Huiji, 4273
Zhang, Ji, 479
Zhang, Jiacheng, 533
Zhang, Jiajun, 115, 391, 2955, 3803, 4154
Zhang, Jun, 2764
Zhang, Lintao, 684
Zhang, Min, 533, 725, 3654
Zhang, Minghua, 4514
Zhang, Ningyu, 986
Zhang, Qi, 2540
Zhang, Richong, 2742
Zhang, Rui, 1005, 1653, 3911
Zhang, Saizheng, 2369
Zhang, Shanshan, 1991
Zhang, Shaowu, 2507
Zhang, Sheng, 944, 1664
Zhang, Soufei, 3110
Zhang, Tong, 2897, 3855, 4253, 4449
Zhang, Wei, 986, 2246
Zhang, Wen, 2246, 4284
Zhang, Xiaoming, 4057
Zhang, Xinbo, 2098
Zhang, Xingxing, 779
Zhang, Xinsong, 2195
Zhang, Xinyuan, 1829
Zhang, Yi, 1103, 4306
Zhang, Yifei, 338, 4752
Zhang, Yiqing, 1005
Zhang, Yu, 349, 4470
Zhang, Yuan, 328
Zhang, Yuchen, 3339
Zhang, Yue, 2226
Zhang, Yuhao, 2205
Zhang, Zhao, 3198
Zhang, Zhe, 3412
Zhang, Zhisong, 4785
Zhang, Zhongfei, 4340
Zhang, Zhuosheng, 2401
Zhang, Zilin, 3911
Zhao, Chen, 1077
Zhao, Dongyan, 3433, 3890, 4088
Zhao, Hai, 2401, 2923, 4785
Zhao, Jianqiang, 447
Zhao, Jieyu, 4847
Zhao, Jinman, 601
Zhao, Jun, 182, 1267

Zhao, Sanqiang, 3164
Zhao, Wei, 3110
Zhao, Yang, 391
Zhao, Yao, 3901
Zhao, Yuanyuan, 3793
Zhao, Zhou, 3110
Zheng, Bo, 2422
Zheng, Renjie, 3188
Zheng, Yinhe, 587
Zhipeng, Guo, 3540
Zhong, Haoxi, 3540
Zhong, Ruiqi, 46
Zhong, Zexuan, 1608
Zhou, Ben, 2065
Zhou, Chunting, 3285
ZHOU, Deyu, 3423
Zhou, Guodong, 725, 3654
Zhou, Hao, 4631
Zhou, Kangyan, 708
Zhou, Ming, 779, 1597, 2077, 2496
Zhou, Mingyang, 3643
Zhou, Wanhao, 2195
Zhou, Yalin, 2246
Zhou, Yaqian, 791
Zhou, Yichao, 4847
Zhou, Yu, 4154
Zhou, Zhi-Hao, 621
Zhu, Hai, 2246
Zhu, Hao, 358, 993, 4642, 4803
Zhu, Junnan, 4154
Zhu, Kenny, 2185, 3477, 4110
Zhu, Qile, 4663
Zhu, Xuan, 587
Zhuang, Fuzhen, 3198
Zhuang, Yimeng, 587
Zhuge, Hai, 4046
Zimmermann, Roger, 2594
Ziser, Yftah, 238
Znotins, Arturs, 4535
Zong, Chengqing, 115, 391, 1435, 2955, 3803, 4154
Zou, Lei, 2098